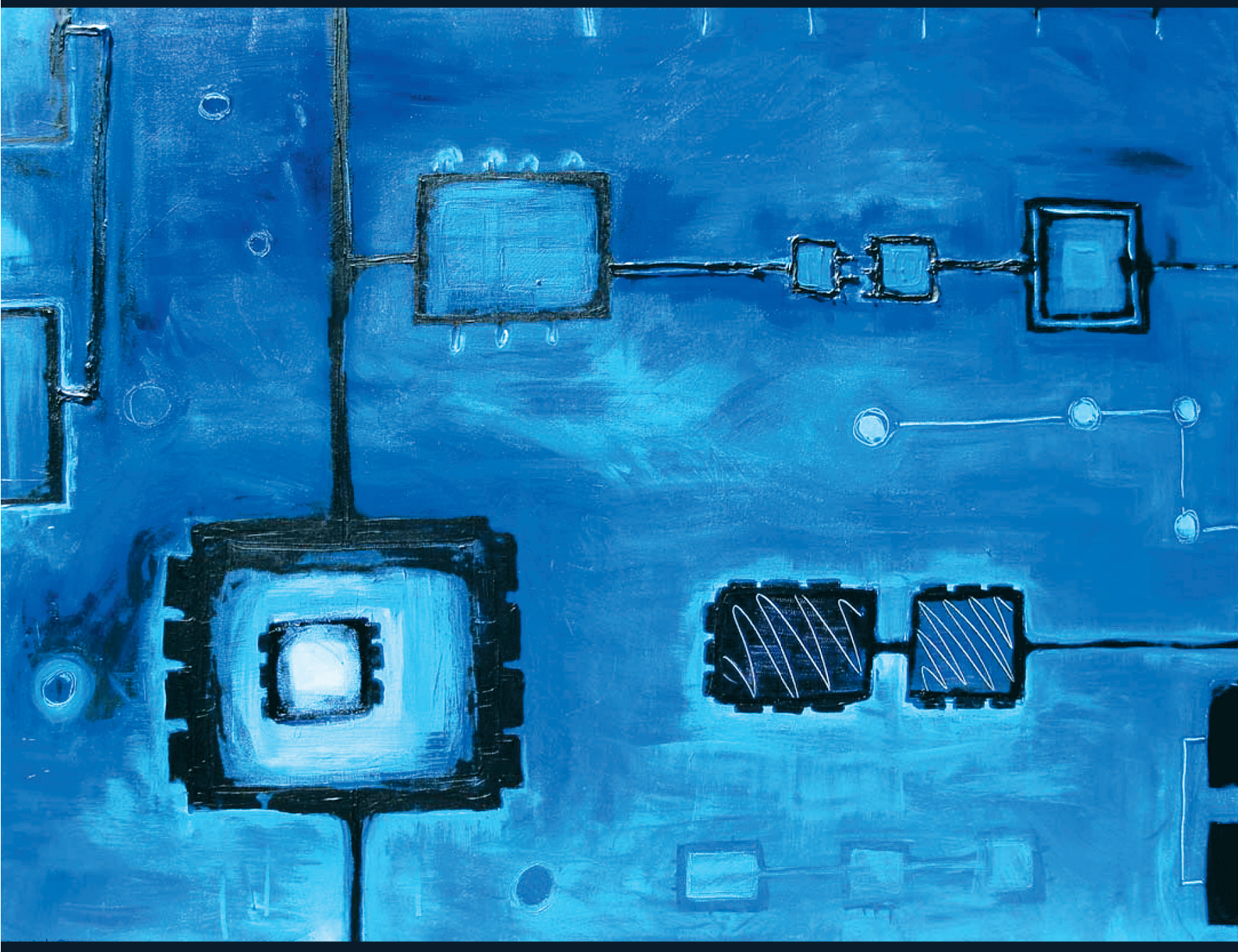
Allied Telesis

LAN Switching and Troubleshooting Guide

Version 1.0



Compiled by

Andrew Riddell

LAN Switching and Troubleshooting Guide

First Edition

Andrew Riddell

About Allied Telesis

Allied Telesis is a world class leader in delivering IP/Ethernet network solutions to the global marketplace. We create innovative, standards-based IP networks that seamlessly connect users with their voice, video and data services. We are an international company headquartered in Japan with major divisions in Europe, Asia and North and South America. Our partners include the world's largest distributors, integrators, solution providers and resellers to assure you receive immediate local service and support.

Allied Telesis has been designing, manufacturing and selling networking products for over 20 years. Our philosophy of producing products of the highest quality, at affordable prices, has resulted in Allied Telesis products being deployed in networks of all types and sizes across the world. Our proven track record of providing solid technology, excellent support and full feature products has allowed Allied Telesis to become the worldwide de-facto standard in many areas of technology. With a portfolio of products that can provide end-to-end networking for both Service Provider, Enterprise and SMB customers, Allied Telesis is the natural choice for many world class organizations.

Please visit us online at http://www.alliedtelesis.com

Printed in New Zealand
First Printing: October 2009
Document Number
C613-04003-00-REV D

ISBN: 978-0-473-15794-4

Co-authors
Andrew Darrell
Michael Moffatt
Cameron Drewett
Wayne Sangster

Editors
Elizabeth Moffatt
Carin van Bolderen
Rebecca Officer

Technical Illustrators
Elizabeth Moffatt
Carin van Bolderen
Rebecca Officer
Angela Maxwell

Cover Designer
Marnie Hart

Introduction

A LAN is a high-speed, data network that supplies connectivity to a group of computers, printers, and other devices that are in close proximity to each other, as in an office building, a school or a home. Network switches work by connecting network paths together and making intelligent decisions about moving traffic in a network. By keeping track of changes in the network, switches reduce congestion and improve speed.

This guide explains the basics of LAN switching and provides a high-level view of the fundamental components and key protocols used to help smooth data flow. It's full of easy to understand definitions, step-by-step instructions, and diagrams to help you manage and troubleshoot a switched LAN.

One of the basic components of a modern network is the VLAN. In this guide, we begin by explaining why and how VLANs evolved. Learn how VLAN membership, tagging, and classification all help provide a flexible, logical network topology.

Then, we move on to the world of multicast traffic. Multicasting provides an efficient way to transmit packets to a group of hosts simultaneously while conserving network bandwidth. We describe the benefits of Loop Protection and Spanning Tree, Ethernet Protection Switching Rings (EPSRing™) and Static and Dynamic Link Aggregation.

Packet switching within an Ethernet switch is essentially quite simple, as you will see from the guide's flowcharts. Dynamic tables make decisions about forwarding or dropping data packets. We take you through a packet processing walk-through explaining this process in simple terms.

Virtual Chassis Stacking (VCStack™) enables you to manage two or more Allied Telesis switches as one single switch. We explain the physical creation and the configuration required on stack members.

Finally, we provide some practical advice on how to go about troubleshooting different issues that arise in networks. Essential reading for network engineers. This guide aims to provide you with useful information in an easily understood manner. Armed with clearly defined strategies, you will be better able to interpret and troubleshoot switched LAN network issues effectively.

Contents

Chapter 1 | VLANs (Virtual LANs)

Introduction

VLANs are a fundamental component of Ethernet LAN switching, as they define the Layer 2 structure of a network, and underpin all other aspects of the network's operation. How data flows through a network is defined by the VLAN structure, and most features of a network: security, resiliency, application, and support are VLAN-aware, and dependant on VLANs for their effective operation. When dealing with issues in a network, the fundamental starting point is understanding both the physical and logical topology of the network. This cannot be done effectively without a very clear understanding of VLANs – why they were developed, the advantages of using them, and how they work.

This chapter focuses on the **nature and operation** of **VLANs** themselves; the interaction of VLANs with other aspects of Ethernet networking are the subject matter of other chapters.

The discussion begins with a definition of what VLANs are, followed by a brief review of how and why they evolved. Next, we examine how VLANs are implemented, and how VLAN information is carried in Ethernet frames. VLAN classification rules are then explained. Although protocol-based and subnet-based VLANs are rarely used, they provide a useful context for fully understanding how a switch allocates VLAN membership to a packet on ingress. Finally, there are some examples of common simple VLAN configurations.

List of terms

VLAN tag

A 32-bit field inside an Ethernet frame that identifies which VLAN a packet is forwarded to. The tag can, optionally, also contain an indicator of what priority to give the packet at times of congestion.

Collision domain

A physical region of a local area network (LAN) in which data collisions can occur.

Broadcast domain

A section of an Ethernet network comprising all the devices that will receive broadcast packets sent by any device in the domain. Separated from the rest of the network by a Layer 3 switch.

VLAN classification

A packet can be allocated VLAN membership based on its protocol, subnet, or port.

VLAN Overview

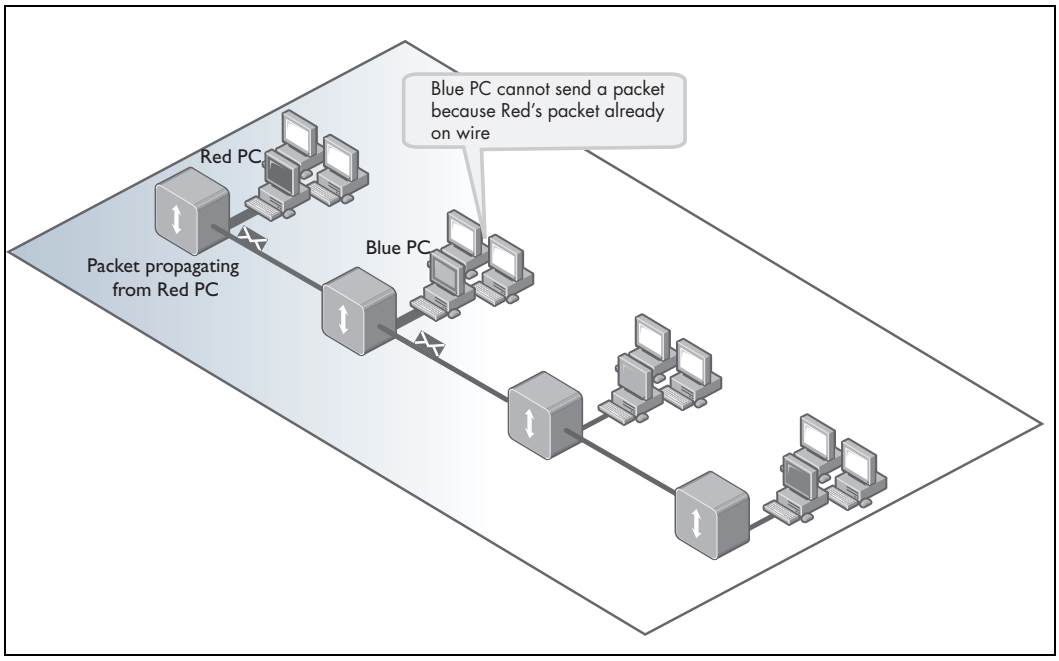
In simple terms, a VLAN is a set of workstations within a LAN that can communicate with each other as though they were on a single, isolated LAN. Among other things, this means that:

- Broadcast packets sent by one of the workstations will reach **all** the others in the VLAN.
- Broadcast packets sent by one of the workstations in the VLAN will not reach any workstations that are not in the VLAN.
- Broadcast packets sent by workstations that are not in the VLAN will never reach workstations that are in the VLAN.
- Workstations within a VLAN can communicate with each other without needing to go through a gateway. For example, IP connections are established by sending ARP messages for the destination IP and sending packets directly to the destination workstation—there is no need to send packets to the IP gateway to be routed to another subnet.
- Workstations within a VLAN can communicate with each other using non-routable protocols.

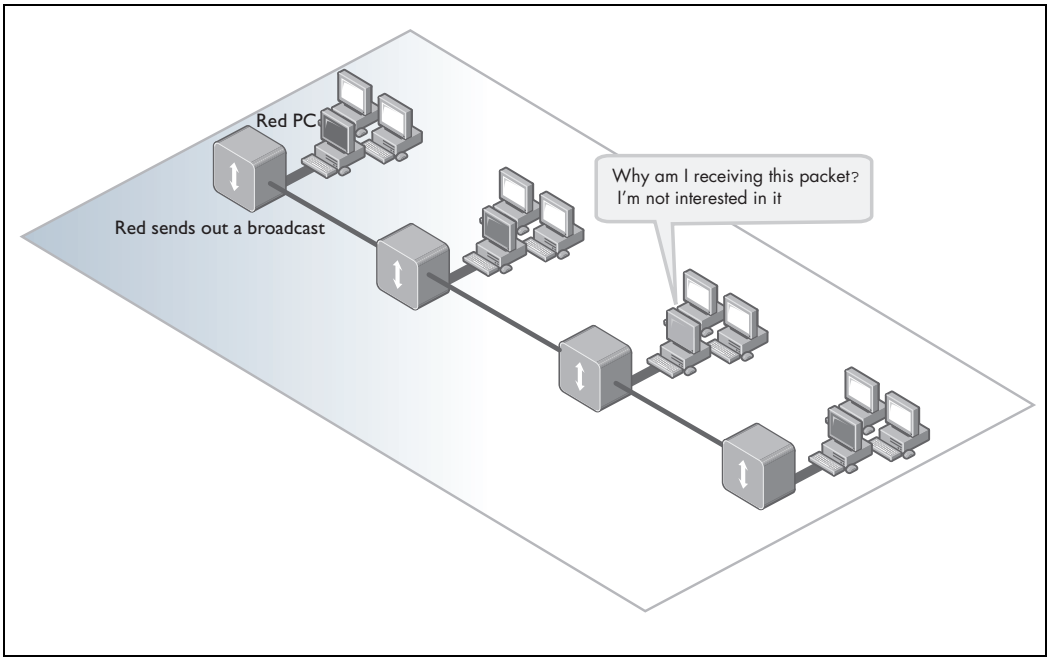
The development of LANs

The basic reason for splitting a network into VLANs is to reduce congestion on a large LAN. To understand this problem, we need to look briefly at how LANs have developed over the years.

Initially LANs were very **flat**—all the workstations were connected to a single piece of coaxial cable, or to sets of chained hubs. In a flat LAN, every packet that any device puts onto the wire gets sent to **every** other device on the LAN. As the number of workstations on the typical LAN grew, they started to become hopelessly congested; there were just too many **collisions**, because most of the time when a workstation tried to send a packet, it would find that the wire was already occupied by a packet sent by some other device.



Also, a good deal of network bandwidth was wasted on the transmission of broadcast and multicast packets to every corner of the network, even if the majority of the workstations on the network did not need to see most of these packets.



The problem of **excessive collisions** lead to the development of **switches to replace hubs**. The significant advantage of switches are that they send unicast packets only onto the segments where they are needed. But, using switches alone could not solve the problem of bandwidth being wasted by broadcasts and multicasts. Instead, reducing the range of broadcast and multicast packets required splitting a LAN into a set of Virtual "broadcast domains", or VLANs.

The idea of using VLANs to segment LANs was not immediately apparent to network developers. Instead, other solutions were used to initially try to solve the problem of excessive collisions. Over time, these solutions left LANs still with the problem of large broadcast areas. The next sections explain this evolution from physically segmented LANs to VLANs:

- "Using routers to segment LANs" on page 4
- "Using switches to segment LANs" on page 4
- "Using VLANs to segment LANs" on page 5

Using routers to segment LANs

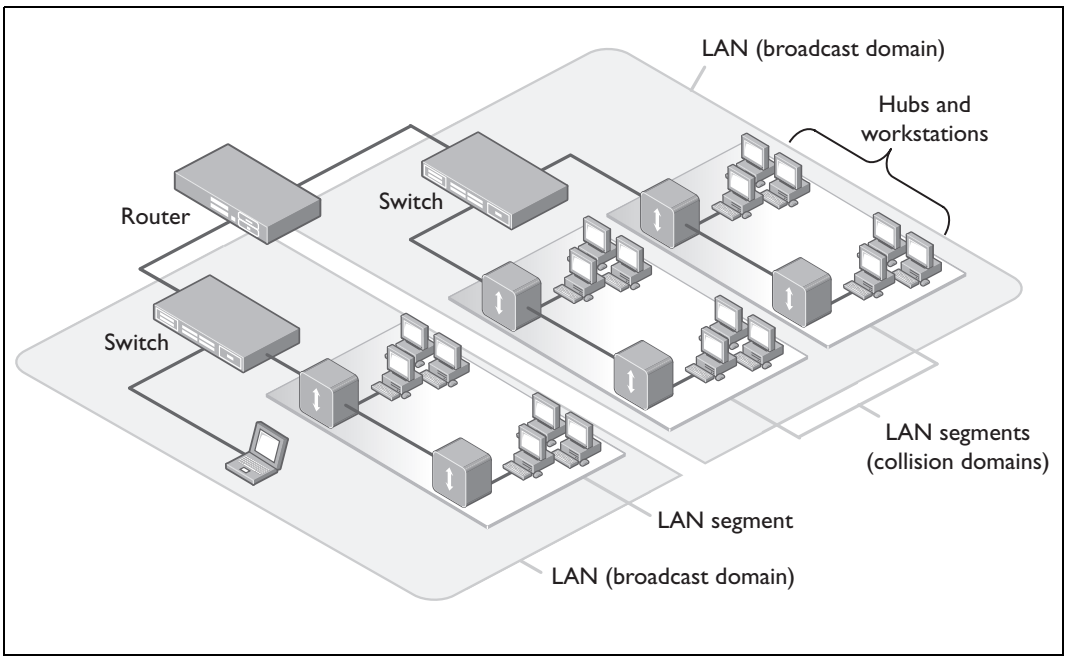
The early solution to the problem of excessive collisions was to segment the network using routers. This split the network into a number of smaller sub-LANs, with fewer workstations on each sub-LAN, and therefore less congestion.

Of course, routable data being sent between sub-LANs would have to be routed, so the Layer 3 addresses would have to be organized so that each sub-LAN had an identifiable set of addresses that could be routed to—such as an IP subnet or an AppleTalk zone. Non-routable protocols would have to be bridged, which is not quite so congestion-reducing, because bridges forward all broadcasts. But, at least for unicast packets, a bridge only forwards packets if it knows that the destination address is not in the originating LAN.

Using switches to segment LANs

As switches became more available, there was a move from chained hubs to a set of hubs connected to a switch. A switch only sends traffic to a given port if the traffic must go to that port. So switches have the effect of reducing congestion at workstations, by stopping the workstations from seeing all the traffic from the other ports of the switch.

A simple switched network, though, still needs routers to set the boundaries of where broadcasts are sent (referred to as “broadcast containment”). The typical LAN was set up as shown in the next diagram:



Domain terminology

The above diagram introduces the concept of a **LAN segment**. This is also referred to as a **collision domain**, because when a device is trying to send a packet, it can only collide with packets sent by other devices on the same segment. Each LAN segment consists of all the devices attached to a single switch port—the switch stops packets from different ports from colliding with each other.

The next level of organisation within the network is referred to as a **broadcast domain**, because if any device within that region sends out a broadcast packet, it will be transmitted to all devices in that region, but not to devices beyond it.

Using VLANs to segment LANs

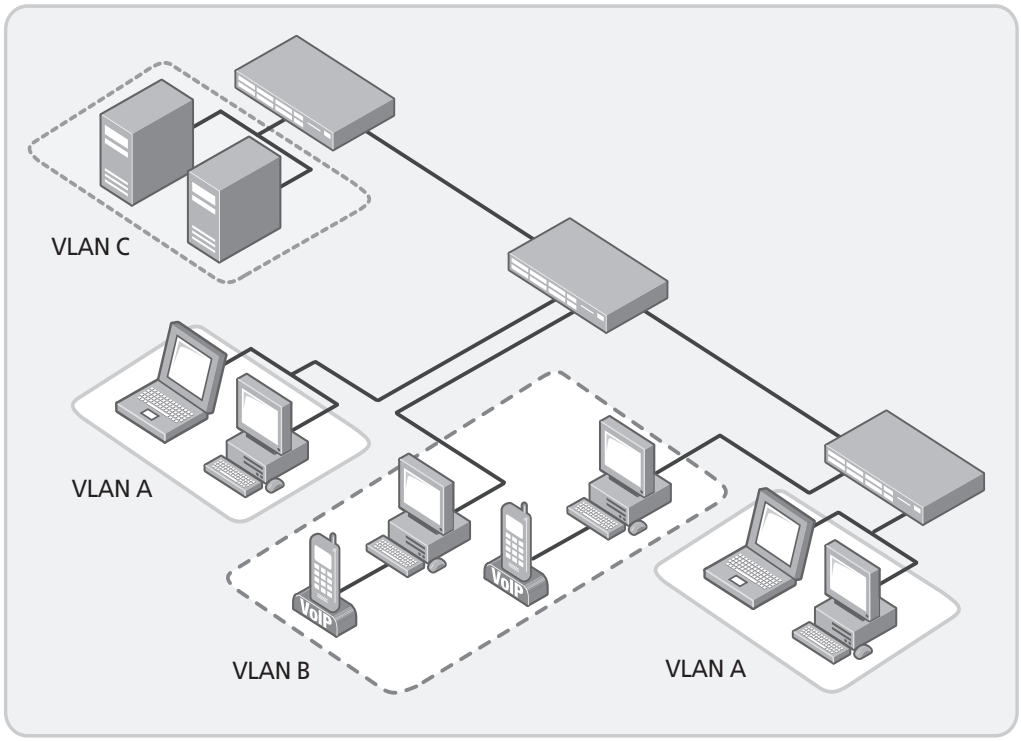
Network routers started to become bottlenecks as LANs became larger, data rates became faster, and users desired greater flexibility. This is because:

- routers typically forward data in software, and so are not as fast as switches
- splitting up a LAN using routers meant that a sub-LAN typically corresponded to a particular physical location. This became limiting when many users had laptops, and wanted to be able to move between buildings, but still have the same network environment wherever they plugged in.

Switch vendors started implementing methods for defining "virtual LANs"—sets of switch ports, usually distributed across multiple switches, that somehow interacted as though they were in a single isolated LAN. This way, workstations could be separated off into separate sub-LANs without being physically divided up by routers.

At about the same time, hubs became less popular and have now been largely replaced by Layer 2 switches. This has made the whole concept of a collision domain somewhat historical. In modern networks, a **collision domain** mostly consists of a single device attached to a Layer 2 switch port, or possibly a PC with something like an IP phone attached to it. With the advent of duplex Ethernet interfaces, simultaneous sending and receiving on the same segment became possible, effectively eliminating collisions in normal circumstances.

The layout of the LAN has become more like the next diagram:



Instead of the LANs corresponding to physical areas divided from each other by routers, there are virtual LANs distributed across the network. For example, all the devices in the various areas labelled "VLAN A" all belong to a single virtual LAN—i.e. a single broadcast domain.

VLANs–a flexible, logical network topology

VLANs enable a malleable virtual topology to be overlaid on top of a physical network topology. The abstraction of the logical Layer 2 topology from the physical network makes it possible to have precise control over which parts of the network different data types may access.

The remarkable advantages of VLANs come from their great flexibility. This flexibility lies in the fact that VLANs are in essence just a set of rules residing within Ethernet switches. These rules assign VLAN membership to packets, and then control where the packets may be forwarded, based on the VLAN they have been assigned to. The rules are logical,

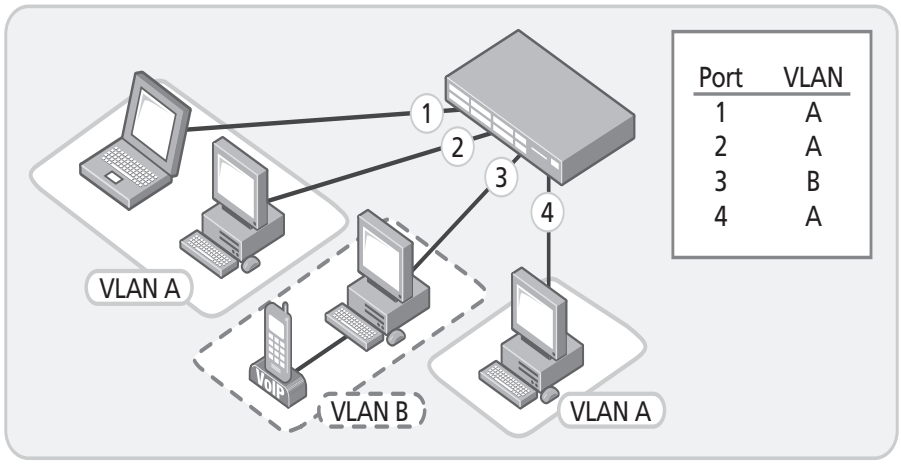
rather than physical, entities, and they act upon packets, rather then upon the devices that originate those packets. As a result, the topology of a VLAN can be quickly and easily altered. Ports, switches, and entire sections of a network can be added to or removed from a VLAN's reach by the simple process of entering a few switch configuration commands.

Advantages of using VLANs

1. **Greater flexibility**. If VLANs are set up the right way, then users can move their desks, or just move around the place with their laptops, and still be within the same VLAN when they plug their device in at the new location. This is much harder when a network is physically divided up by routers.
2. **Ease of partitioning off resources**. Network administrators can put servers or other equipment that they want to restrict access to, into their own VLAN. Then users in other VLANs can be given access selectively.
3. **Scalability**. The broadcast containment provided by VLANs means that new users can be added into the network, in new VLANs, with minimal impact on existing users in existing VLANs.
4. **Performance**. As discussed in the section "Using VLANs to segment LANs" on page 5, routers that forward data in software become a bottleneck as LAN data rates increase. Doing away with the routers removes this bottleneck. Moving to a VLAN model of network segmentation opened the door to the idea of transferring packets between VLANs within a switch (called Layer 3 switching), thereby removing the router bottleneck from the network.

Implementing VLANs

A VLAN is created by associating ports with the VLAN, and creating the criteria for VLAN membership for packets arriving into those ports. By far the most common VLAN membership criterion is port-based–packets arriving on a specific port are always associated with a specific VLAN. On an edge switch, this means that each attached device would associate with a VLAN based on the port it was connected to, as shown in the next diagram.



In this example, all devices connected to a given port automatically become associated with the VLAN to which that port was assigned. This in effect divides a switch up into a set of independent sub-switches.

There are other, very-rarely used VLAN membership criteria, which are discussed in "Protocol and subnet-based VLAN classification" on page 12. However you do not need to know about these criteria to understand the general operation of VLANs in this section.

Using VLAN tagging to distribute a single VLAN across multiple switches

The diagram in "Using VLANs to segment LANs" on page 5 shows an example network where some devices in VLAN A are connected to one switch, while other devices in VLAN A are connected to another switch.

You may be asking “Are these both part of the same VLAN A, or separate VLANs that all happen to be called VLAN A?”

The answer is that they are all parts of the same VLAN—there is a single VLAN A that is spread across two switches.

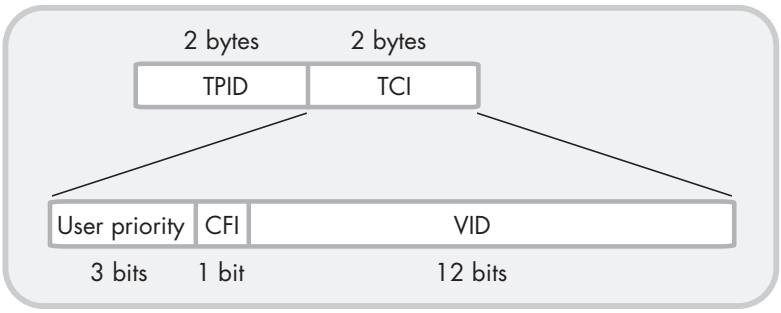
How is this achieved? How does one switch know that when it receives a broadcast packet that it associates to VLAN A that it must also forward that broadcast to other switches?

This can be done in a number of different ways, and in the early days of VLANs, just about every one of these ways was tried. Some vendors had their switches use a proprietary protocol to inform each other of their VLAN tables; some vendors used time-divided multiplexing in which different timeslots were allocated to different VLANs; other vendors used **frame tagging**.

In the end, frame tagging became the accepted standard. In most respects this is a simple and elegant solution. However, it initially had one big downside: it required a fundamental change to format of the Ethernet header. This split the world's Ethernet devices into those that recognized tagged headers and those that did not recognize tagged headers–making a lot of Ethernet equipment obsolete.

How does tagging work?

Tagged headers are comprised of 4 bytes inserted into the header of Ethernet packets. This consists of 2 bytes of Tag Protocol Identifier (TPID) and 2 bytes of Tag Control Information (TCI):



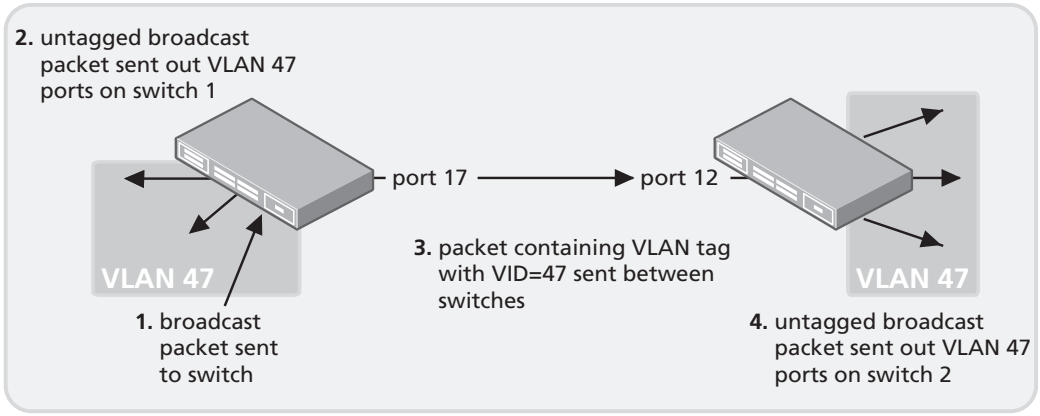
TPID is the tag protocol identifier, which indicates that a tag header is following. The TCI contains the user priority, Canonical Format Indicator (CFI), and the VLAN ID (VID):

- the User priority is a 3-bit field that allows priority information to be encoded in the frame. Eight levels of priority are allowed, where zero is the lowest priority and seven is the highest priority.
- the CFI is a 1-bit indicator that is always set to zero for Ethernet switches. CFI is used for compatibility between Ethernet and Token Ring networks. If a frame received at an Ethernet port has a CFI set to 1, then that frame should not be bridged to an untagged port.
- the VID field contains the identifier of the VLAN. This field is all that is really needed for distributing VLANs across switches—the other fields were added by the IEEE at the same time so that only the one change to reading Ethernet headers was required.

VLAN tagging example

The following example shows how this tag makes it simple to distribute VLANs across switches.

Consider a broadcast packet arriving at a switch port. By some criterion, the packet is associated with VLAN 47 (i.e. a VLAN with VLAN ID=47). Port 17 of this switch is connected to port 12 of another switch that also has some ports in VLAN 47.



The network administrator needs to configure port 17 of switch 1 and port 12 of switch 2 as **tagged** member ports of VLAN 47. This tells:

- switch 1 to send the broadcast out port 17 as a tagged packet, with VID=47 in the tag
- switch 2 to accept that tagged packet and associate it with VLAN 47

The tag makes it very easy for the second switch to know what to do with the packet. When it receives a tagged packet from Switch 1, then Switch 2 can send the packet out all its member ports of VLAN 47, because that is what it does with broadcasts that it has associated with VLAN 47.

The two simple rules of tagging are:

- if a port is a tagged member of a VLAN, then any packets sent out that port by that VLAN must have a tag inserted into the header.
- if a tagged packet arrives in at a port, and the port is a tagged member of the VLAN corresponding to the VID in the packet's tag, then the packet is associated with that VLAN.

With these two simple rules, it is possible to distribute VLANs across multiple switches.

Mixing tagged and untagged packets on the same port

In the previous section, we discussed using tags to indicate the VLAN membership of packets that are transferred from one switch over to another. But, it is also possible that untagged packets will be transported across that link that joins the two switches.

For example, in the diagram on "VLAN tagging example" on page 10, port 17 of switch 1 could be an untagged member of VLAN 56, and port 12 of switch 2 could also be an untagged member of VLAN 56. So, if switch 1 needs to transport VLAN 56 packets over to switch 2, it would send them untagged. For both switches, any untagged packets they receive on the port connected to the other switch are deemed to belong to VLAN 56, despite not having a specific tag to indicate their VLAN membership. So, when untagged packets arrive at port 12 of switch 2, then switch 2 automatically associates these packets with VLAN 56.

Obviously, a port can be an untagged member of only one port-based VLAN, otherwise there would be uncertainty about what VLAN the incoming untagged packets belonged to. (Note however, that the situation is a bit different when subnet and protocol-based VLANs are introduced—see page 12 onwards). This VLAN is often referred to as the **native** VLAN of the port.

Often, you might not want to associate a native VLAN with the port that connects a switch to another switch, so that all packets coming into that port **must** use a VLAN tag to indicate their VLAN membership. This stops the switch from accepting any untagged packets on the port. In AlliedWare Plus, this is achieved by configuring a port to trunk mode and not configuring a native VLAN on it.

Ingress filtering

Consider a port that is connected to a normal workstation. Normal applications on the workstation will never send tagged packets, so there is no requirement for the switch port to accept tagged packets.

But, is there any harm if the port does accept tagged packets if they happen to come along? Well, the answer is **yes**. If the workstation does send tagged packets, then it is very likely doing so for malicious reasons.

To guard against such maliciousness, most switches provide the ability to configure **ingress filtering**. When ingress filtering is applied to a port, **packets will only be accepted into a port if they match the VLAN configuration of that port**. If the port is an untagged member of one VLAN, and nothing else, then only untagged packets will be accepted on the port. If the port is tagged for a set of VLANs, then a tagged packet will be accepted into the port only if it is tagged with the VID of one of the tagged VLANs configured on the port.

We highly recommend that you configure ingress filtering on all switch ports, because there is seldom a good reason for a port to accept packets from VLANs that are not configured on that port. Under AlliedWare Plus ingress filtering is **enabled** on all ports by default.

Protocol and subnet-based VLAN classification

The previous sections have discussed the implementation of port-based VLANs. However, there are other ways of defining VLAN membership. In this section, we will consider two examples of these other types of VLAN:

Protocol-based VLANs

With this method, different protocol types are assigned to different VLANs. For example, IP defines one VLAN, IPX defines another VLAN, Netbeui yet another VLAN, etc.

Protocol	VLAN
IP	1
IPX	2

Subnet-based VLANs

With this method, the VLAN membership is defined by the subnet to which a workstation's IP address belongs.

Subnet	VLAN
23.2.24.0	1
26.21.35.0	2

Understanding VLAN membership

After reading the descriptions for protocol and subnet based VLAN membership, you may be wondering:

- isn't a VLAN normally a set of workstations? How does a protocol specify a workstation? Surely a given workstation can send out packets using different protocols (often at the same time), depending on which applications it is running?

  and

- in the case of a subnet-based VLAN, which VLAN does a workstation belong to when it is not sending IP packets?

When initially learning about VLANs, it is common to think of VLANs as sets of workstations. However, it is important to remember that in essence, a VLAN is **a set of rules residing within a switch**, and these rules assign VLAN membership to **packets**. When using protocol-based and subnet-based VLANs, it is **data streams** that are divided into VLANs, not necessarily whole workstations.

This means that a given workstation can belong to multiple VLANs–it could belong to one subnet-based VLAN when sending IP packets, another protocol-based VLAN when

sending IPX packets, and yet another different port-based VLAN when sending some other protocol.

When analysing the VLAN setup on a network, it is a mistake to ask "what VLAN does this workstation belong to?" Instead, the more meaningful question to ask is "if a packet of such-and-such a protocol arrived at port x of the switch, which VLAN would that packet be associated with?"

In practice, port-based VLAN configurations is often all that a network administrator wants to achieve, and it is common for these administrators to think of the VLANs as sets of workstations. However, once the VLAN configuration on a switch becomes complex, with multiple VLANs of different types all configured on the same port, it is no longer possible to really think about the VLAN from the workstation point of view. It becomes necessary to think of it from the **packet** point of view.

Therefore, it really is vital to think of packets being associated to VLANs when trying to understand VLAN configurations. Any other approach can end in confusion.

This is particularly so when dealing with VLAN-related security issues like VLAN-hopping attacks. The key to VLAN security exploits is the very fact that the VLAN membership of a packet is carried by the packet itself. It is not possible to understand the operation of those attacks without thinking about the ways that the VLAN rules within switches operate on each packet.

Examples of determining a packet's VLAN membership

The previous sections have looked generically at how VLANs are implemented. In this section, we put this generic description into context by examining what happens when certain packets arrive on a switch with multiple VLANs configured.

VLAN setup on the switch

The switch uses the following VLANs:

- ports 1 - 4 of the switch are untagged members of the port-based VLAN 2.
- ports 3 - 6 of the switch are untagged members of the subnet-based VLAN 3, which is configured for the subnet 192.168.1.0/255.255.255.0.
- port 4 is an untagged member of the protocol-based VLAN 4, which is configured for protocols IP and IPX.
- port 5 is a tagged member of VLAN 2
- port 6 is a tagged member of VLAN 4

This switch implementation also has the following rules:

1. Subnet-based VLANs take precedence over protocol-based VLANs, which take precedence over port-based VLANs.
2. If a tagged packet arrives at a port, it is only accepted **if** that port is a tagged member of the VLAN corresponding to the VID in the packet's tag, (i.e. ingress filtering is turned on).

The following table describes the VLAN configuration on the switch. The priority column shows which VLAN membership has precedence for packets that could match more than one VLAN, with 1 being the highest precedence.

Port	VLAN membership	Tagging and Type	Priority
1	VLAN 2	Port - untagged	3
2	VLAN 2	Port - untagged	3
3	VLAN 3	Subnet (192.168.1.0/24) - untagged	1
	VLAN 2	Port - untagged	3
4	VLAN 3	Subnet (192.168.1.0/24) - untagged	1
	VLAN 4	protocol - untagged	2
	VLAN 2	port - untagged	3
5	VLAN 3	Subnet (192.168.1.0/24) - untagged	1
	VLAN 2	port - tagged	3
6	VLAN 3	Subnet (192.168.1.0/24) - untagged	1
	VLAN 4	protocol - tagged	2

Treatment of packets

Now let us look at certain packets arriving at the switch:

▶ **An untagged IPX packet arrives at port 1**

Port 1 is **only** a member of VLAN 2, so the packet will be associated with VLAN 2. The switch will look at the forwarding table for VLAN 2. If the destination MAC address of the packet is in the forwarding table, the packet will be forwarded out the corresponding port in that table entry. If the destination MAC address is not in the forwarding table for VLAN 2, then the packet will be flooded out **all** other ports of VLAN 2. So, it will be sent as an untagged packet out ports 2- 4, and as a tagged packet out port 5.

▶ **An untagged IP packet with source/destination IP address in the 192.168.1.0/255.255.255.0 subnet arrives at port 4**

Port 4 is a member of a subnet-based VLAN 3 configured for the subnet 192.168.1.0/24. So, the packet will be associated to VLAN 3. The switch will look at the forwarding table for VLAN 3. If the destination MAC address of the packet is in the forwarding table, the packet will be forwarded out the corresponding port in that table entry. If the

destination MAC address is not in the forwarding table for VLAN 3, then the packet will be flooded out **all** other ports of VLAN 3. It will be sent as an untagged packet out ports 3, 5, and 6.

▶ **An untagged IP packet with source/destination IP address *not* in the 192.168.1.0/255.255.255.0 subnet arrives at port 4**

Port 4 is a member of a subnet-based VLAN 3 configured for the subnet 192.168.1.0/24, but the packet's addresses are not in that subnet. So, the packet will not be associated with VLAN 3.

The next VLAN type in the precedence order is the protocol-based VLAN. Port 4 is a member of the protocol-based VLAN 4, configured for IP and IPX. As this is an IP packet, it will be associated with VLAN 4. The switch only has one other port in VLAN 4. The packet will be sent as a tagged packet out port 6.

▶ **An untagged AppleTalk packet arrives at port 4**

The AppleTalk packet cannot be associated with the subnet-based or the protocol-based VLANs on port 4, so it is associated with the port-based VLAN (VLAN 2) on the port.

The switch will look at the forwarding table for VLAN 2. If the destination MAC address of the packet is in the forwarding table, the packet will be forwarded out the corresponding port in that table entry. If the destination MAC address is not in the forwarding table for VLAN 2, then the packet will be flooded out **all** other ports of VLAN 2. So, it will be sent as an untagged packet out ports 1-3, and as a tagged packet out port 5.

▶ **A tagged packet with VID=10 arrives at port 5**

Port 4 is an untagged member of the protocol-based VLAN 4, configured for IP and IPX. But, the packet is tagged, so it will be dropped if ingress filtering is enabled.

Port 5 is a tagged member of VLAN 2. But the VID in the packet's tag does not match the VID of the VLAN (2), so the packet is dropped if ingress filtering is enabled.

▶ **A tagged packet with VID=2 arrives at port 5**

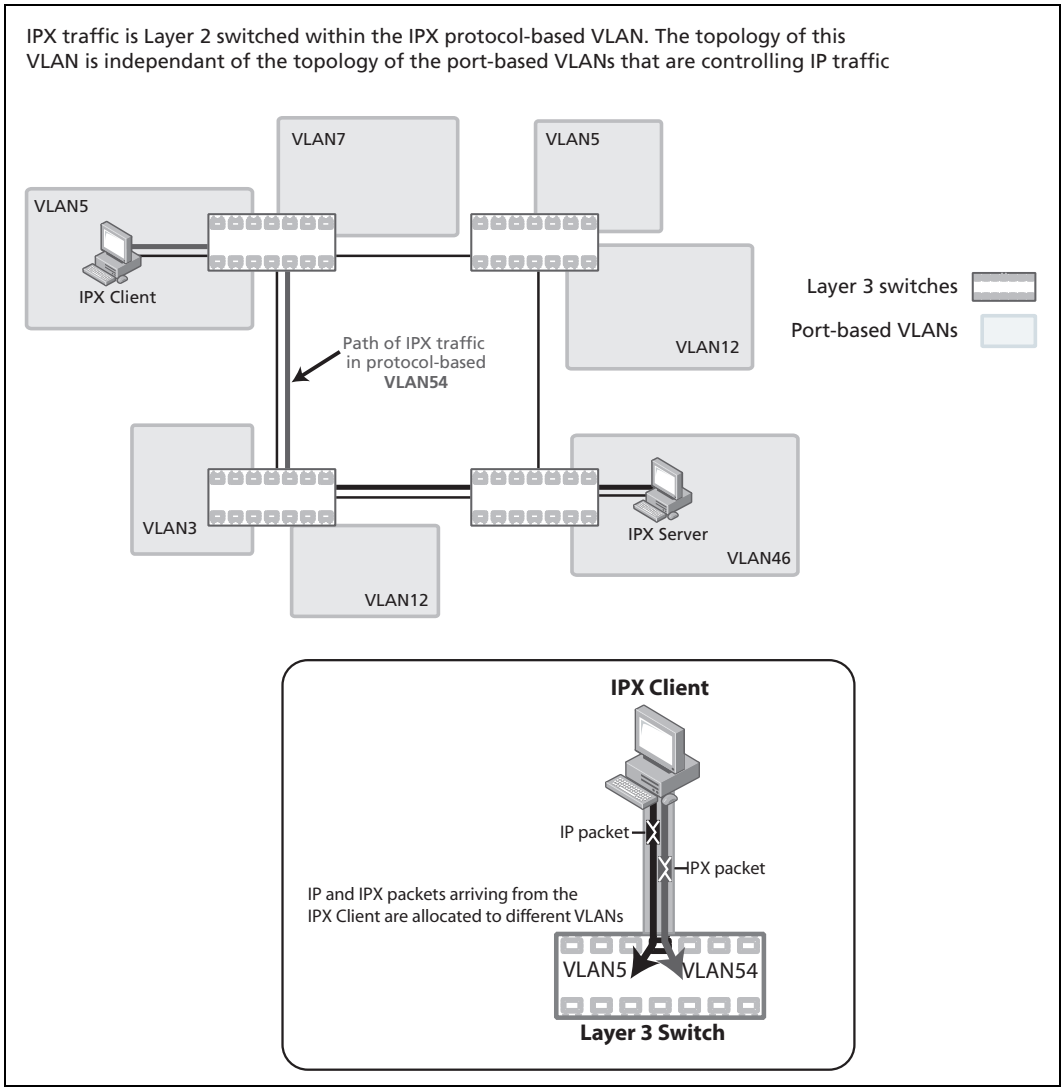
Port 5 is a tagged member of VLAN 2. The VID in the packet's tag matches the VID of the VLAN, so the packet is associated with VLAN 2. The switch will look at the forwarding table for VLAN 2. If the destination MAC address of the packet is in the forwarding table, the packet will be forwarded out the corresponding port in that table entry. If the destination MAC address is not in the forwarding table for VLAN2, then the packet will be flooded out **all** other ports of VLAN 2. So, it will be sent as an untagged packet out ports 1-4.

What is the purpose of non port-based VLANs?

The main purpose of non port-based VLANs is to enable hosts in a network that share some specific characteristic to communicate directly with each other at Layer 2 independently of whatever boundaries are imposed between the devices by the network's port-based VLANs.

For example, consider a case where a network is divided into a number of port-based VLANs, and the vast majority of the traffic on the network is IP. However, a legacy application on the network still uses IPX communication. If the devices that are involved in this IPX communication are in separate port-based VLANs, and the Layer 3 switches in the network are not capable of routing IPX, then there is potentially a problem with enabling this inter-VLAN IPX communication.

This problem can be solved by defining an IPX protocol-base VLAN. With this protocol-based VLAN in place, IPX packets are classified into this VLAN, rather than the local port-based VLAN. With appropriate tagging of uplink ports into the IPX VLAN, an IPX-specific Layer 2 path across the network is created. This enables the IPX communication to cross the port-based VLAN boundaries despite the Layer 3 switches not being capable of IPX routing.

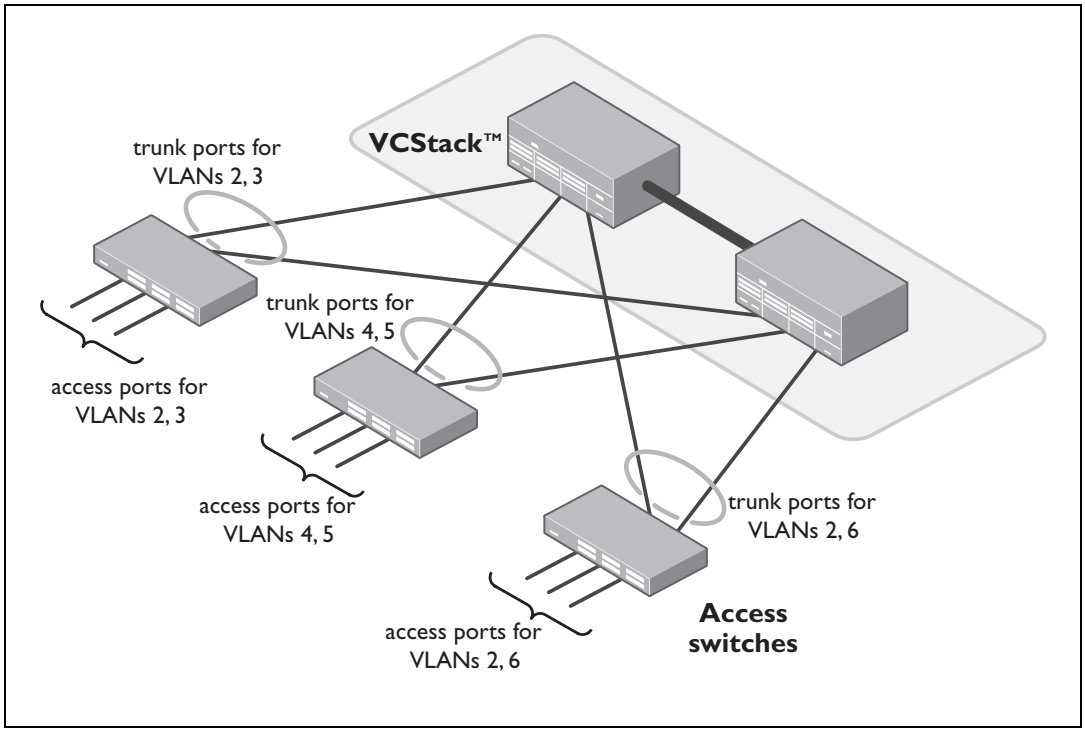


Configuring VLANs using the AlliedWare Plus OS

Static port-based VLANs

By far the most common usage of VLANs is as statically configured port-based VLANs.

The following diagram shows a setup that could provide static separation of the network into VLANs for departments within a business. Edge-facing ports in access switches are configured as Access ports for chosen VLANs, and the ports on links between switches are configured as Trunk ports for the VLANs that are configured on a given access switch.



Configuring an access switch

To configure the typical access switch as shown in the previous diagram, use these AlliedWare Plus OS commands on the switch:

1. Create the VLANs.

Use the commands:

```
awplus(config)#vlan database
awplus(config-vlan)#vlan 2 name accounting
awplus(config-vlan)#vlan 3 name marketing
awplus(config-vlan)#exit
```

2. Add the VLANs to each port.

Use the commands:

```
awplus(config)#interface port1.0.1-1.0.12
awplus(config-if)#switchport access vlan 2
awplus(config)#interface port1.0.13-1.0.24
awplus(config-if)#switchport access vlan 3
awplus(config)#interface port1.1.1
awplus(config-if)#switchport mode trunk
awplus(config-if)#switchport trunk allowed VLAN add 2,3
awplus(config)#interface port1.2.1
awplus(config-if)#switchport mode trunk
awplus(config-if)#switchport trunk allowed VLAN add 2,3
```

Configuring the core switch or VCStack™

Similarly, the VLAN configuration of the stacked core switches is:

1. Create the VLANs.

Use the commands:

```
awplus(config)#vlan database
awplus(config-vlan)#vlan 2 name accounting
awplus(config-vlan)#vlan 3 name marketing
awplus(config-vlan)#vlan 4 name sales
awplus(config-vlan)#vlan 5 name engineering
awplus(config-vlan)#vlan 6 name admin
awplus(config-vlan)#exit
```

2. Add the VLANs to each port.

Use the commands:

```
awplus(config)#interface port1.1.1-2.1.1
awplus(config-if)#switchport mode trunk
awplus(config-if)#switchport trunk allowed VLAN add 2,3
awplus(config)#interface port1.1.2-2.1.2
awplus(config-if)#switchport mode trunk
awplus(config-if)#switchport trunk allowed VLAN add 4,5
awplus(config)#interface port1.1.3-2.1.3
awplus(config-if)#switchport mode trunk
awplus(config-if)#switchport trunk allowed VLAN add 3,6
```

Dynamic VLAN assignment

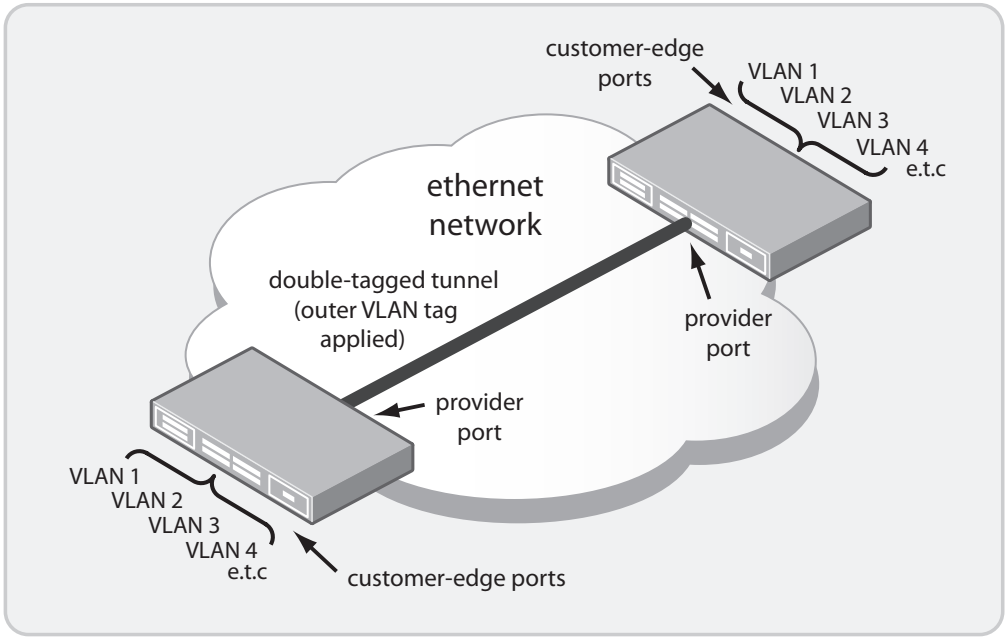
An increasingly popular mode of VLAN usage is **dynamic** VLAN assignment, where the PC-facing ports on the access switches are configured to authenticate connected devices using 802.1x. Once a device is authenticated on a port, the RADIUS server allocates to the switch the VLAN to use for that device. The VLAN assignment by the RADIUS server is based on the identity supplied by the authenticated device, and can also include other factors like Network Access Control (NAC) policy conformance.

Although the details of the configuration and operation of this mode is beyond the scope of this book, it is well worth being aware of, as it is a mode that is rapidly growing in popularity. It enables a user to have a network environment based on their identity, rather than the physical location of their connection. Additionally, it enables a network to provide restricted access to unknown guest users, and to implement a NAC system, in which a user's level of network access is based on their adherence to a security policy.

VLAN double tagging

One other increasingly popular use of VLANs is double tagging. This is a mechanism whereby a whole set of VLANs can be tunnelled across another Ethernet network.

In the following diagram, multiple different VLANs enter the switch at a customer-edge port of the double-tagging VLAN. These VLANs are all encapsulated by double-tagging, and emerge at the provider port of the double-tagging VLAN with the outer tag applied.

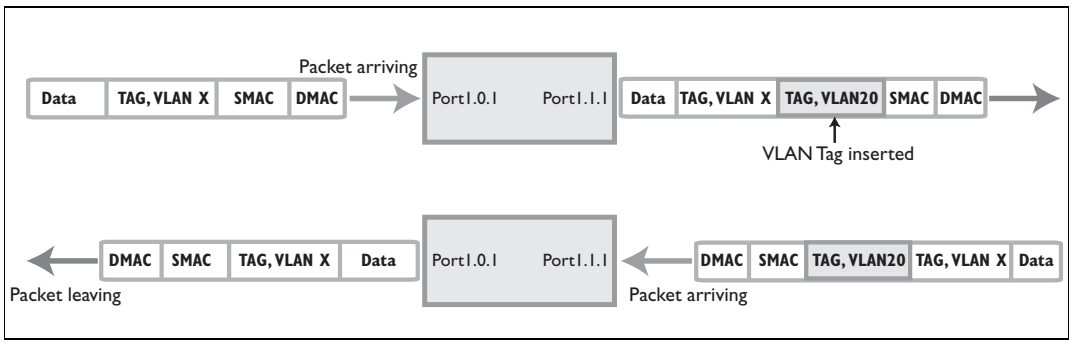


This mechanism enables a set of virtualised networks to share the same physical Ethernet infrastructure.

The **switchport vlan-stacking customer-edge-port** command enables the switch to use VLAN double-tagging, as shown in this example configuration:

```
awplus(config)#vlan database
awplus(config-vlan)#vlan 20 name tunnel20
awplus(config-vlan)#exit
awplus(config)#interface port1.0.1
awplus(config-if)#switchport access vlan 20
awplus(config-if)#switchport vlan-stacking customer-edge-port
awplus(config-if)#interface port1.1.1
awplus(config-if)#switchport mode trunk
awplus(config-if)#switchport trunk allowed vlan add 20
```

The effect of this configuration is that ANY packet arriving into the switch via port 1.0.1, that is switched out port1.1.1, will have an extra VLAN tag inserted into its Ethernet header, containing VID 20. Even if the packet already has a tag when it arrives into port1.0.1, the VLAN20 tag will be added to the packet's Ethernet header, in front of the existing tag, when it is switched out port1.1.1. Conversely, if a packet arrives into the switch on port1.1.1, and its first tag has VID20, and it is to be switched out port1.0.1, then the VLAN20 tag will be stripped out of the packet. The packet will then be switched out port1.0.1 with whatever VLAN tag (if any) that was previously sitting in behind the VLAN20 tag.



Chapter 2 | Managing Multicast Traffic with IGMP and IGMP Snooping

Introduction

Multicasting provides an efficient way to transmit packets to a group of hosts simultaneously while conserving network bandwidth. Unlike unicasting, hosts do not directly communicate with the traffic source, but instead join a multicast group. The network devices between the hosts and the source keep note of which hosts are in a multicast group, and only forward the stream to these hosts.

Internet Group Management Protocol (IGMP) is the mechanism by which hosts can join a multicast group, and through which routers and Layer 3 switches keep track of where to forward multicast streams. Layer 2 networks can also snoop on IGMP messages so that they can continue to forward multicast streams efficiently.

Multicasting is a communication method whose popularity is increasing rapidly. It is particularly well suited to digital distribution of Television and Radio. Therefore, it is important for a network engineer to understand the role that IGMP plays in multicast communication, and how to ensure that IGMP is operating correctly within a network.

List of terms

IGMP Snooper

A device that spies on IGMP messages to create flow efficiencies by ensuring that multicast data streams are only sent to interested ports. A Snooper can decide on the best path to send multicast packets at Layer 2 but does not initiate any IGMP communications.

IGMP Querier or Router

A device in a subnetwork that is the coordinator for all multicast streams and IGMP membership information. Each subnet only has one active querier.

This chapter starts with a brief description of IP multicasting, in order to explain the context within which IGMP operates, and make it clear what a multicast group really is. This is followed by a description of how hosts join, stay in, and leave a multicast group using IGMP, and the changes to the protocol that version 3 introduced. Next, the chapter explains why switches perform IGMP snooping and how IGMP snooping works. The chapter ends with a quick guide to the IGMP troubleshooting information that is available in this book.

Overview of Multicasting

Multicasting is a technique developed to send packets from one location in a network to many other locations without any unnecessary packet duplication. In multicasting, one packet is sent from a source and is replicated as needed in the network to reach as many end-users as necessary.

Multicasting is different from broadcasting; while broadcast packets are sent to every possible receiver, multicast packets need only be forwarded to receivers that want them. The benefit of this technique is bandwidth conservation–it is the most economical technique for sending a packet stream to many locations simultaneously.

The IP addressing for multicast packets works differently from unicast and broadcast packets. A multicast stream sends packets out with a destination IP address that identifies a specific multicast group. It does not at all specify an end host, like unicast; or a whole subnet, like broadcast.

Data transmission methods:

Unicast
Two individual devices hold a conversation just between themselves.

Broadcast
One device sends out data that is intended to be received and processed by every device that it reaches.

Multicast
One device sends out data that is intended to be received and processed by a selected group of the devices it reaches.

This makes multicasting a connectionless process. The server simply sends out its multicast UDP packets, with no idea who will receive them, or whether they are successfully received. It is the hosts that tell the network that they wish to receive a multicast stream, using the Internet Group Management Protocol (IGMP). This is a Layer 3 protocol; however Layer 2 switches can also conserve bandwidth within their LAN by using IGMP snooping to track which hosts require the data stream.

Multicast groups

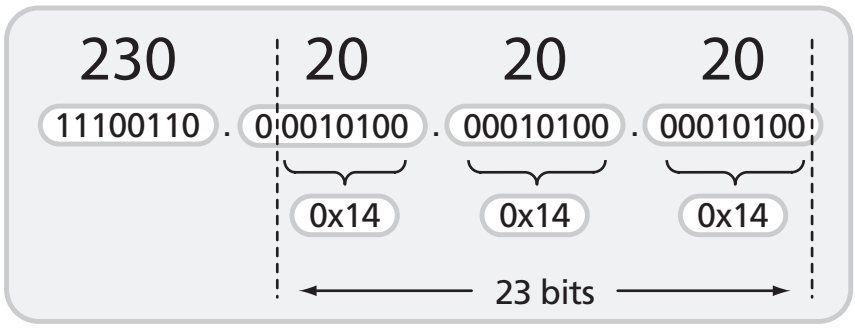
The concept of a group is crucial to multicasting. A group is the set of hosts that wish to receive a particular multicast stream, and is identified by a multicast IP address and matching multicast MAC address. The multicast sender transmits the stream to the group address, and only members of the group can receive the multicast data.

A multicast IP address is within the Class D address range–this is all IP addresses from 224.0.0.0 to 239.255.255.255. They are also referred to as Group Destination Addresses (GDAs). Like all IP addresses, GDAs are a limited resource and there are rules about who may use addresses from which ranges of values. There are even programs that can hand out a group address for a temporary time, and claim it back when the server has finished with it.

A GDA also has an associated multicast MAC address (useful for IGMP snooping). This MAC address always starts with **0100.5e** and is followed by the last 23 bits of the GDA

translated in hex. Each IP multicast packet sent over an Ethernet network must use the destination MAC address that corresponds with the packet's GDA.

The next figure shows how the MAC address is created for the GDA 230.20.20.20. Here you can see the IP address translated to binary, then the last 23 bits translated into 3 hex numbers. Note - 0x14 is the hexadecimal representation of decimal 20.



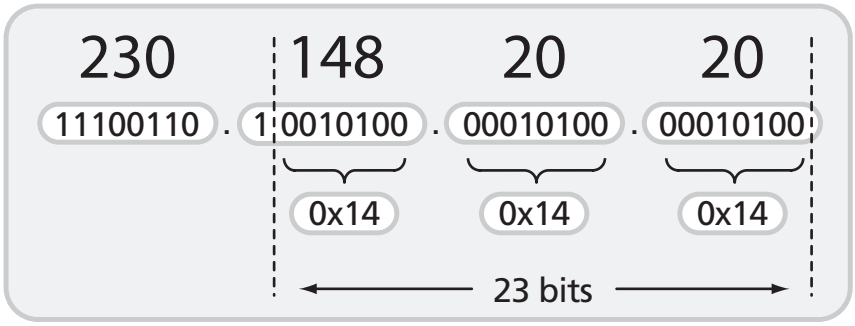
Combine these with the prefix 0100.5e, and the full MAC address corresponding to 230.20.20.20 is:

```
0100.5e14.1414
```

As this example shows, the first byte of the GDA is not reflected in the MAC address at all. This means that any of the following GDAs would also correspond to MAC address 0100. 5e14.1414:

```
224.20.20.20, 225.20.20.20, 226.20.20.20, ...239.20.20.20
```

As well, the first bit of the second byte of the GDA does not contribute to the MAC address. For example, the next diagram shows GDA 230.148.20.20 in binary and then translated to 3 hex numbers from the last 23 bits.

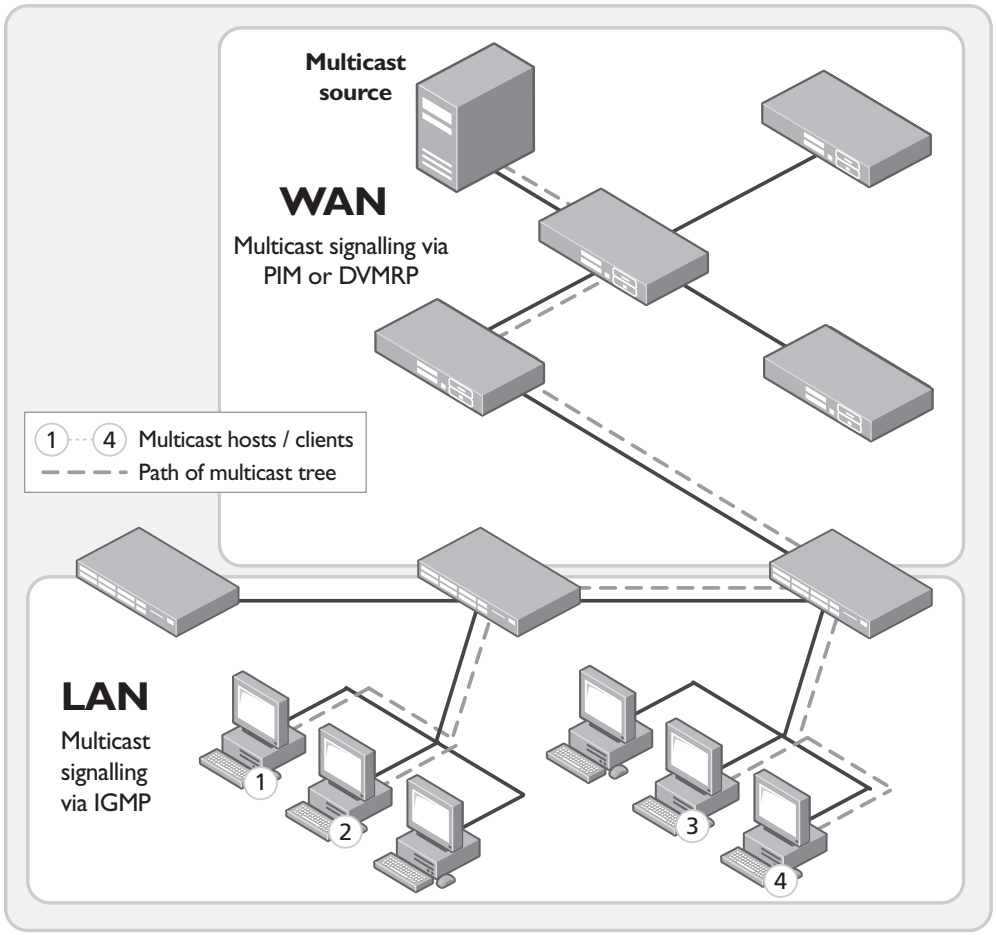


As this example shows, the MAC address corresponding to 230.148.20.20 is also 0100.5e14.1414.

The association between multicast MAC addresses and GDAs is not a one-to-one mapping, but a one-to-many mapping. However, devices can still determine the set of GDAs that are possible for a given MAC address.

Components in a multicast network

There are several protocols and roles required in a multicast network, including IGMP. This section describes the end-to-end process of transporting multicast data through a network to highlight where IGMP fits in this process.



At the two ends of a multicast data transmission are:

- **The source**
  This is typically a server or video encoder. It sends the stream of multicast data out through its network interface. It is unaware of where the recipients of the stream are, or if there are any recipients.
- **The recipients**
  These are storage or display devices, such as PCs, set-top boxes, or security video archivers. The recipients signal their desire to receive a particular multicast stream by sending out IGMP messages requesting the stream.

The role of the network in-between is to deliver the multicast stream to the recipients as efficiently as possible. The devices achieve this by exchanging signalling information between themselves in order to establish a forwarding path along which the multicast stream will flow. Each node informs the next node up the chain that it needs to receive the multicast stream. Once this series of requests reaches the router nearest the multicast source, then that router will start to forward the stream. All the nodes between the source and the recipients are ready to forward the stream, due to their

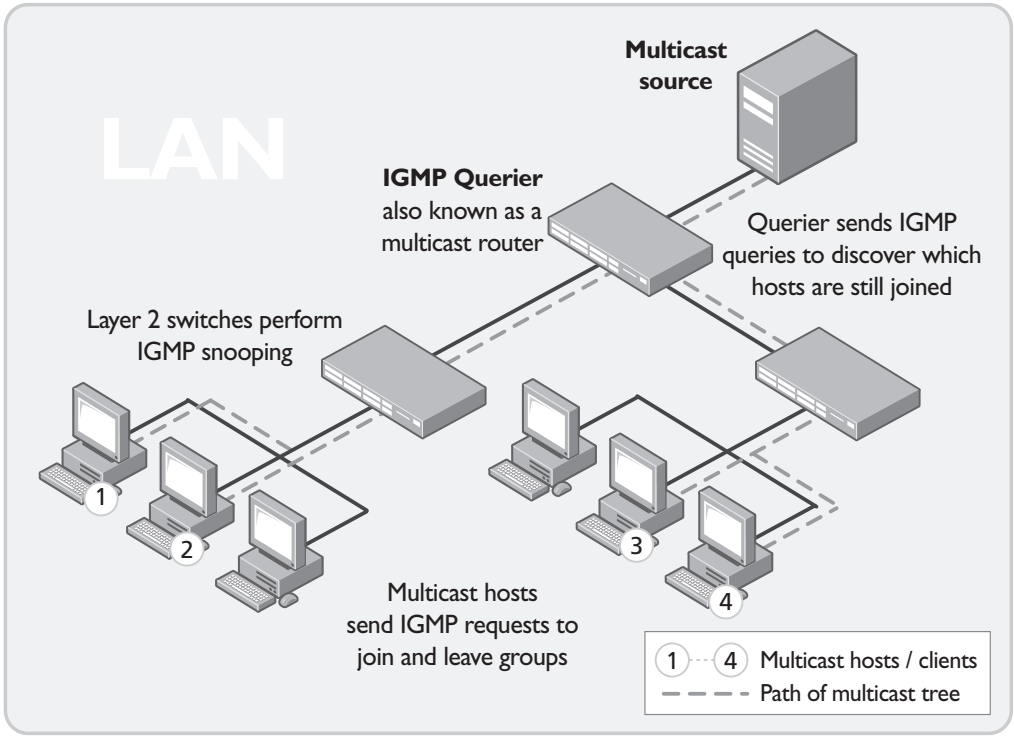
having received the signalled requests. In this way, the stream is efficiently forwarded right through to the recipients.

The type of signalling that the network uses falls into two categories:

- in the local area network where the Recipients are located, the signalling consists of the exchange of IGMP packets.
- as soon as the signalling needs to leave the VLAN containing the recipients and cross into other VLANs and subnets, then a Layer 3 multicasting protocol like PIM or DVMRP is used between the routers in the Layer 3 network.

Layer 3 multicasting is beyond the scope of this book. The focus of this chapter is only on the IGMP signalling within a Layer 2 network.

The next diagram shows a self-contained multicast system all within a single LAN. As it is a Layer 2 LAN it uses IGMP to signal group membership.



In every Layer 2 multicast network, there needs to be a device that is sending IGMP queries into the network. This is essential to maintaining multicast flows once they have been established (see "Staying in the multicast group (Query message)" on page 27 for more information).

Typically, the device that is configured to send the queries is the router that is the gateway from the local network into a Layer 3 network. However, as in the self-contained multicast network shown in the diagram, it does not necessarily need to be performing Layer 3 forwarding of multicast.

The device configured to send the queries is sometimes referred to as the multicast router (or mrouter), and sometimes as the Querier. Note that in this chapter it is referred

to as the Querier, to emphasise its role as a sender of IGMP queries, rather than its role as a Layer 3 forwarding device.

The rest of this chapter will look into the detail of how the IGMP protocol works, and how the switches in a Layer 2 network make use of IGMP to ensure multicast streams are sent only where they need to be sent.

Understanding IGMP

Internet Group Management Protocol (IGMP) is the protocol that hosts use to indicate that they are interested in receiving a particular multicast stream.

There are three versions of IGMP specified by RFCs. All versions use IGMP Membership reports and Query messages. However, these are the major differences:

- Version 1 (IGMPv1) only includes the Membership reports and Query messages. It became obsolete with the introduction of version 2; however, network devices running IGMP v2/3 are still compatible with this version.
- Version 2 (IGMPv2), developed in 1997, introduced a Leave message. This reduced the time it took for a host to leave a multicast group.
- Version 3 (IGMPv3), developed in 2002, allows a host to specify not only which group address it wants to receive, but what source addresses it wants to receive the group from. See "IGMPv3 changes" on page 28 for more information.

Joining a multicast group (Membership report)

When a host wants to receive a stream (referred to as “joining a group”) it sends out an IGMP packet containing the address of the group it wants to join. This packet is called an IGMP Membership report, often referred to as a “join packet”.

This packet is forwarded through the LAN to the local IGMP querier, which is typically a router. (Note that throughout this chapter, the term ‘router’ refers generically to a device that is performing Layer 3 forwarding. It may be a traditional router, or it may be a Layer 3 switch).

Once the querier has received an IGMP join message, it knows to forward the multicast stream to the host. If it is not already receiving the stream, it must tell the devices between itself and the multicast source (which may be some hops away from the querier) that it wishes to receive the stream. As described in "Components in a multicast network" on page 24, this might involve a process of using Layer 3 multicast protocols to signal across a WAN, or it might be as simple as receiving a stream from a locally connected multicast server.

Staying in the multicast group (Query message)

There are two other IGMP message types aside from the join packet. One of these is the Query message.

The Query message is used by a querier to determine whether hosts are still interested in an IGMP group. At certain time intervals (the default is 125 seconds), the querier sends an IGMP query message onto the local LAN. The destination address of the query message is a special "all multicast groups" address. The purpose of this query is to ask "Are there any hosts on the LAN that wish to remain members of multicast groups?"

After receiving an IGMP query, any host that wants to remain in a multicast group must send a new join packet for that group. If a host is a member of more than one group, then it sends a join message for each group it wants to remain a member of.

The querier looks at the responses it receives to its query, and compares these to the list of multicast streams that it is currently registered to forward. If there are any items in that list for which it has not received query responses, it will stop forwarding those streams. Additionally, if it is receiving those streams through a Layer 3 network, it will send a Layer 3 routing protocol message upstream, asking to no longer receiver that stream.

Leaving the multicast group (Leave message)

How a host leaves a group depends on the IGMP version that it is using.

Under IGMP version 1, when a host has finished with a data stream, the local querier continues to send the stream to the host until it sent out the next query message (and received no reply back from the host).

IGMP version 2 introduced the Leave message. This allows a host to explicitly inform its querier that it wants to leave a particular multicast group. When the querier receives the Leave message, it sends out a group-specific query asking whether any hosts still want to remain members of that specific group. If no hosts respond with join messages for that group, then the querier knows that there are no hosts on its LAN that are still members of that group. This means that for that specific group, it can ask to be pruned from the multicast tree.

IGMP version 3 removed the Leave message. Instead a host leaves a group by sending a join message with no source specified.

IGMPv3 changes

IGMP version 3 adds source awareness to the protocol, providing support for implementing Source Specific Multicast (SSM). This allows the host to specify not only which group address it wants to receive, but what source addresses it wants to receive the group from.

The host can specify the list of desired sources in two modes: **include** and **exclude**. When using the include mode, the host gives the list of source IP addresses that will accept a certain multicast group from; when using the exclude mode, the host gives the list of source IP addresses that it will not accept the multicast group from.

IGMP version 3 also adds a few other changes to the protocol, for example:

- the Query packets include a Robustness Variable and Query Interval.
- all Report packets are sent to 224.0.0.22.
- a single Report packet can contain status records of multiple groups.
- a re transmission mechanism was added for Query and Report packets.

Understanding IGMP Snooping

IGMP snooping is a way for Layer 2 switches to reduce the amount of multicast traffic on a LAN. Note that IGMP snooping is a feature that evolved rather than being specified by a particular RFC or IEEE standard. IGMP snooping in the AlliedWare Plus OS is an implementation that actively manages the multicast network by filtering out IGMP join and leave messages to the local querier (reducing the load on the querier). It is compatible with networks running all IGMP versions.

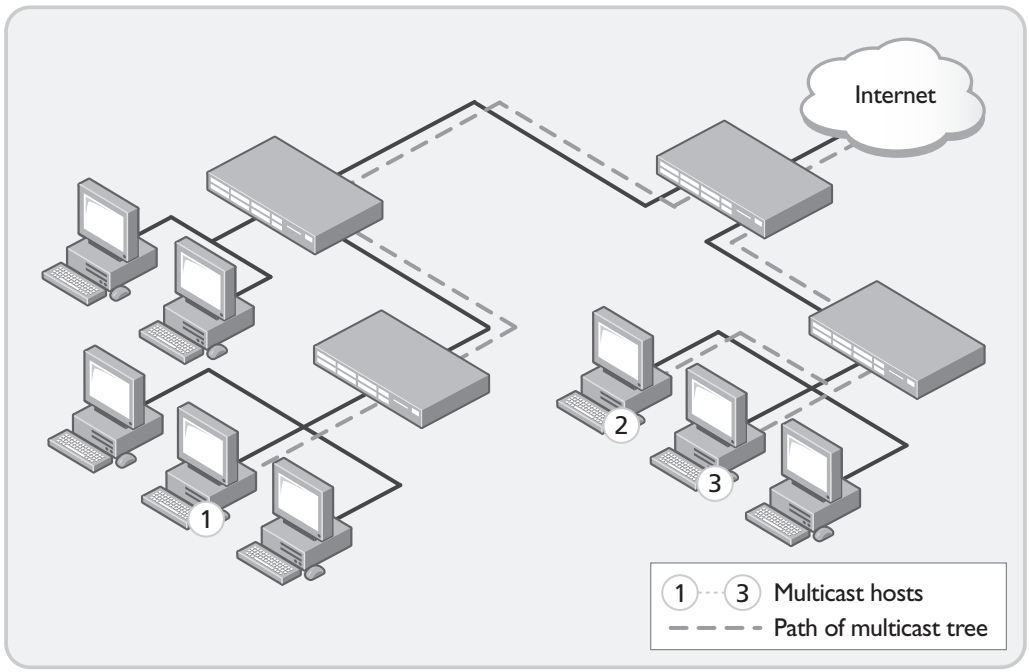
Why IGMP snooping is required

Without IGMP snooping, Layer 2 switches forward on IGMP requests to the local querier, but do not interpret them. When the querier starts sending a multicast stream to the switch, it by default must forward the traffic to all ports (except the port it received the traffic from). This circumvents the purpose of multicasting, which is to make efficient use of bandwidth. For example, if one host requires a multicast stream, then all hosts attached to the switch receive the stream; on a switch with 24 hosts attached, one multicast host would cause the switch to consume 23 times the actual bandwidth required. If another host attached to the switch required a different multicast stream, then the bandwidth use would rise to 46 times what was required.

IGMP snooping provides a solution by allowing the Layer 2 switches to be aware of the IGMP packets that pass through them. The Layer 2 switches “snoop” the packets destined for queriers to track which hosts have asked to join which multicast groups, and which hosts remain in a group. This allows the switch to forward multicast traffic only out the appropriate ports.

The switch will also reduce the number of reports being sent up to the querier. If it sees multiple reports sent for one group, it will forward only one of them.

In the next diagram, the switches are using IGMP snooping to forward a multicast stream only to the three hosts that require it. Without IGMP snooping, every other host in the LAN would also receive the stream.



How IGMP snooping operates

IGMP snooping operates similarly to the multicast protocols. When a switch hears an IGMP report from a host for a given multicast group, the switch adds the host's port number to the list of ports that are listening to the multicast group. When the switch hears an IGMP leave, it removes the host's port from the list, after the completion of the leave process as described on page 27. When there are no hosts listening to a group, the switch informs the local querier to stop sending that group's multicast stream.

IMGP snooping allows query messages to be forwarded to all ports. The hosts that still require the stream respond to the queries by sending reports. The switch intercepts these. Depending on configuration settings, the switch may just forward the reports directly on to the querier, or it may proxy report on behalf of the group, only forwarding on one report for each group.

The switch only snoops IGMP frames–it does not need to snoop any other multicast traffic.

As well as tracking which ports out which to forward multicast groups to hosts, a snooping switch must also identify the ports attached to a path towards multicast queriers.

Learning the Router Port

IGMP snooping will only forward Join requests and Leave messages on ports that it knows head towards the multicast queriers. These ports are referred to as **router ports**. If it does not have at least one router port, it will not forward these messages.

The switch detects router ports by listening for multicast packets that only a querier or router would send. As the only IGMP message received from a querier is a Query message (and this is sent infrequently), it cannot rely on listening solely for IGMP packets. So, the switch listens for other multicast traffic as well, such as:

- PIMv1 hellos sent to 0100.5e00.0002
- PIMv2 hellos sent to 0100.5e00.000d
- DVMRP probes sent to 0100.5e00.0004
- MOSPF messages sent to 0100.5e00.0005 or 06
- RIPv2 messages sent to 0100.5e00.0009
- OSPF messages sent to 0100.5e00.0005 or 06

When the switch receives one of these packet types on a port, it marks the port as a router-connected port, and can forward Join requests and Leave messages through these ports. As well, every possible multicast address is associated as "joined" on this port. This means that any multicast streams received from another port on the switch are sent out through the router-connected ports, ensuring that multicast sources operating from within the LAN are distributed.

Activity when a host joins a group

When the first host in a LAN segment joins a particular multicast group:

1. The host sends an unsolicited IGMP membership report.
2. The switch intercepts the IGMP membership report sent by the host.
3. The switch creates a multicast entry for that group and notes down that the stream for that group should be sent out of the port in which it received the membership report.
4. The switch forwards the IGMP report on to all router ports.
   The multicast querier receives the IGMP report and updates its multicast routing table accordingly, and begins forwarding the stream to the switch.

When a second host joins the same multicast group:

1. The new host sends an unsolicited IGMP membership report.
2. The switch intercepts the IGMP membership report sent by the host.
3. As the multicast entry already exists, the switch simply adds the port to the already existing entry for that multicast group.
4. The switch may or may not forward this second IGMP report to all its router ports, depending on configuration settings.

Activity when a host leaves a group

When a host leaves a group, but the switch still has other host interested the stream:

1. The switch captures the IGMP leave message from the host, noting which port the leave arrived on.
2. If IGMP snooping on the switch is operating in fast-leave mode, then it will just immediately discard this port from the group entry. It will not forward the leave packet to the querier.

Otherwise, the switch will forward the leave message up to the IGMP querier, and wait for the querier to perform a group-specific query on the group that the host is leaving. The switch then waits to see if any IGMP reports are received on that port in response to the query. The period for which the switch waits is determined by values that it snoops from the query packet. These values are the query response interval and the query count. The switch calculates the wait period by multiplying the query response interval by the query count. If no IGMP report for the group is received on the port within that period, then the switch will discard this port from the group entry.

When the last host in the switch's LAN segment leaves the group:

1. The switch captures the IGMP leave message from the host, noting which port the leave arrived on.
2. Regardless of whether IGMP snooping is operating in fast-leave mode or not, it will forward the IGMP leave to the querier, so that the querier will discover that there are no longer any hosts downstream of the switch that are members of the group.
3. If IGMP snooping on the switch is operating in fast-leave mode, then it will immediately discard this port from the group.

Otherwise, the switch will wait for the querier to perform a group-specific query on the group that the host is leaving. The switch then waits for a period equal to the query-response interval specified in the group-specific query. If no IGMP report for the group is received on the port within that period, then the switch will discard this port from the group entry.

Configuring IGMP and IGMP Snooping using the AlliedWare Plus OS

By default, the AlliedWare Plus OS enables IGMP snooping on all VLANs.

Configuring IGMP snooping

You can disable IGMP snooping globally, using the **no ip igmp snooping** command in global configuration mode:

```
awplus(config)#no ip igmp snooping
```

Or, you can disable it on a specific VLAN by using the **no ip igmp snooping** command in interface configuration mode for that VLAN interface:

```
awplus(config)#interface VLAN173
awplus(config-if)#no ip igmp snooping
```

You can statically configure a port as a router port by using the command:

```
awplus(config)#interface VLAN173
awplus(config-if)#ip igmp snooping mrouter interface port1.4.2
```

The mechanism whereby the switch reduces the number of IGMP reports being sent up to the querier by not sending multiple reports for the same group, is enabled by using the command **ip igmp snooping report-suppression** command. This command is entered in interface configuration mode for a VLAN interface, and will only suppress reports received on ports that are members of the VLAN it is configured on.

You can control the IGMP groups that the ports of a given VLAN can learn by applying an access-list that specifies the allowed groups. Use the command:

```
awplus(config)#interface <VLAN-ID>
awplus(config-if)#ip igmp access-group {<1-99>|<word>}
```

Configuring the switch as an IGMP querier

By default the IGMP querier function is disabled by the AlliedWare Plus OS. To configure the switch as an IGMP querier, the function must be enabled on each VLAN that you wish the switch to send queries on, using the **ip igmp** command:

```
awplus(config)#interface <VLAN-ID>
awplus(config-if)#ip igmp
```

By default, the AlliedWare Plus OS will send IGMPv3 queries. The switch can be forced to send IGMPv2 queries with the command:

```
 ip igmp version 2
```

Timers and other parameters associated with the query process are configurable for each interface, using the other IGMP commands available. For example, you can configure the query interval using the command:

```
awplus(config-if)#ip igmp query-interval <2-18000>
```

It is not recommended to change any timers from their default values unless forced by specific circumstances, like installing a switch into a network in which non-default values have already been configured on other equipment.

Monitoring and Troubleshooting IGMP

For advice on troubleshooting, refer to "Troubleshooting Layer 2 multicast issues" on page 124, and "Chapter 10 | Troubleshooting" on page 189

Chapter 3 | Loop Protection using Rapid Spanning Tree Protocol (RSTP)

Introduction

Rapid Spanning Tree Protocol (RSTP) is a loop protection protocol. It allows switches in a multiply-connected, Layer 2 network to create a loop free tree that provides a single unique path between any pair of network nodes. Using RSTP, switches can gather information about each other, negotiate which links to block, and perform ongoing monitoring of the state of the network. This allows you to create a network that has redundant backup paths that can automatically cut over to a backup path when the primary path fails.

List of terms

Port state

Indicates the packet-forwarding status of a port. In RSTP, a port can be in one of 3 states - Discarding, Learning, or Forwarding

Port role

Indicates a port status within the spanning tree topology. In RSTP, ports can have one of 4 roles - Root, Designated, Alternate, or Backup.

Topology change

The movement of a non-edge port to a forwarding state

Root bridge

The one switch that all other switches in the spanning tree use as a point of reference to determine their ports' roles and states.

Port cost

A value indicating the desirability of the port going to a forwarding state. Typically inversely proportional to a port's bandwidth

Because the purpose of RSTP is to protect the network from loops, problems with RSTP can have severe consequences. If too many links are blocked, then parts of the network are inaccessible; if too few links are blocked then loops are created, leading to packet storms. Added to this, the operation of RSTP is relatively complex, and involves a number of operational rules that are specific to particular circumstances. So, the ability to effectively and efficiently debug RSTP issues is an important skill, as debugging often has to be performed under pressure (as the network is down) and requires a very clear understanding of the required behaviour of each switch that is involved in the protocol interaction. This chapter explains how the spanning tree algorithm creates the initial loop-free topology, and how the ports move from blocking to forwarding traffic.

It then describes how the protocol detects an event that will cause a topology change in the network, and how the switches react to the event. Next, it explains how to configure RSTP using the AlliedWare Plus OS; in particular focusing on how to force switches into particular spanning tree roles, and the best-practices for ensuring smooth running of the network. The chapter finishes with details on the types of problems that can occur in RSTP, and the right approaches to troubleshooting those problems.

Overview of Spanning Tree Protocols

The purpose of the Spanning Tree Protocol (STP) is to turn a looped network into a loop-free tree while only blocking the minimum number of ports. As well as ensuring that there are no active forwarding loops in a physically looped network, STP must also ensure that every node in the network still has an active forwarding path to any other node in the network. The protocol needs to block just enough ports in the network to remove the loops without completely cutting off any node. It does this by creating a tree of active forwarding paths over the network.

STP was first standardized in 1990 by the IEEE. Over time, two extensions to STP, initially developed as proprietary extensions, were also standardised by the IEEE: Rapid STP (RSTP) and Multiple STP (MSTP).

This chapter focuses on RSTP, which extended STP to provide faster tree convergence after a topology change. Amongst other optimizations, it includes the following features:

- **PortFast**
  This enables an edge port of the network to bypass the cautious process by which an STP port normally works out whether it should forward data or not.
- **Alternate ports**
  This enables a switch to react quickly if a forwarding port goes down when there is a suitable alternative (blocked) port to take over from the downed port.
- **Negotiated forwarding**
  When a switch wishes to bring a designated port to forwarding, it can actively negotiate this with its neighbor, rather than going through a slower listening phase.

These features mean that RSTP has faster convergence, less network disruption when topology changes occur, and is more tunable to network needs than STP. For example, the PortFast feature allows RSTP to treat any ports that are not connected to other switches, as edge ports, that are forwarding immediately.

Multiple Spanning Tree Protocol enables a set of spanning trees to effectively operate independently on the same set of physical network segments. Each of the almost-independent spanning trees protects a different set of VLANs. While the concept behind MSTP is simple, implementing MSTP on a network can become quite complicated, and is not included in this chapter.

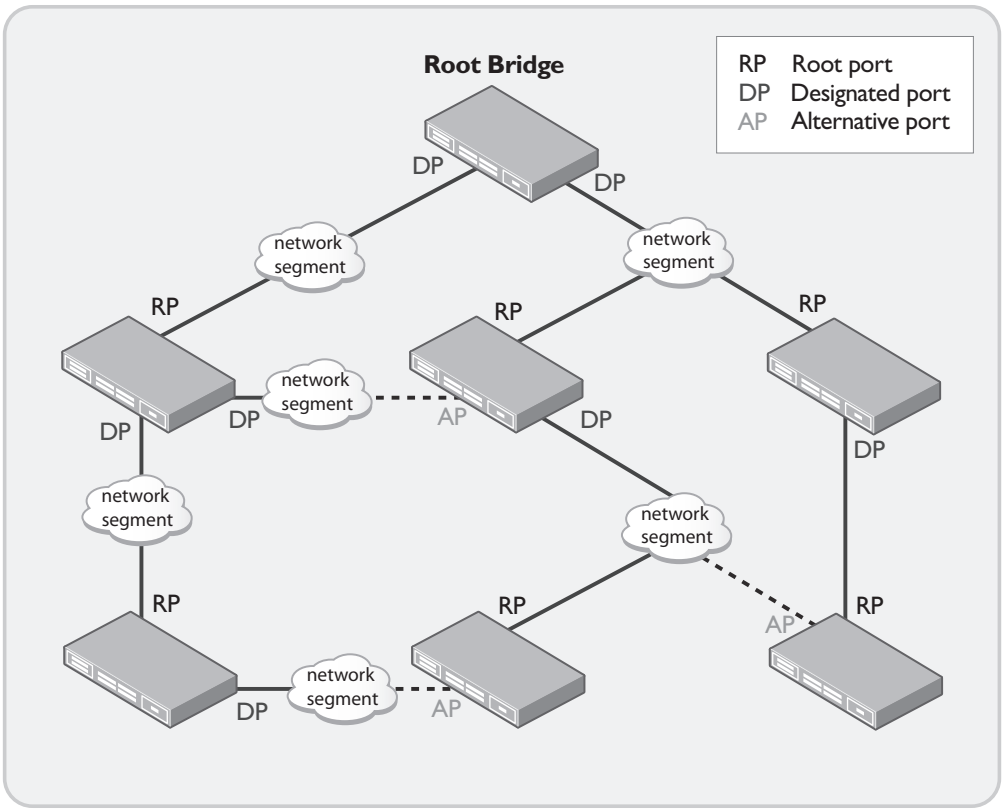
Understanding how trees are negotiated

The following simple set of rules allows the spanning tree protocol to create a loop-free set of forwarding paths that do not leave any network node isolated:

1. One of the switches is elected as the **Root Bridge** for the spanning tree.
2. All ports on the Root Bridge go into the **forwarding** state.
3. All other switches then arrange the spanning tree topology based on their **path cost** to the Root Bridge. The ports are assigned specific roles:
    - the switches work out which of their ports has the 'least cost' path to the Root Bridge. These ports are called **root ports**, and are put into the forwarding state. In essence, root ports are ports heading towards the Root Bridge. Each switch has only one root port at any one time.
    - on direct point-to-point links between pairs of switches, if the port at one end of the link is a root port, then the port on the neighboring switch at the other end of the link is put into the forwarding state. These ports are given the role of **designated** ports. In fact, designated ports are any port in the forwarding state that are not root ports.
    - on links where neither port at either end of the link is a root port, a negotiation occurs to work out which is the 'superior' port (this negotiation will be described in more detail below). This superior port is put into the forwarding state as a **designated port**. The port at the other end of the link is given the role of an **alternate port**–literally, an alternative to the switch's current root port–and is blocked from forwarding. Alternate ports may be quickly transitioned to forwarding if there is a problem with the root port connection to the spanning tree.
    - if multiple ports connect onto a single segment (all connect to a hub, for example), then if any of these ports is a switch's root port, it will be put into forwarding. Among the remaining ports there will be a negotiation to determine the 'superior' port, which will be put into forwarding.

Once all this negotiation is complete, then the protocol has done its job–all the loops have been removed from the network with the least number of blocked ports, and the network is said to have converged.

The following diagram shows an example converged spanning tree.



Bridge Protocol Data Units (BPDUs)

To establish a spanning tree, the switches exchange spanning tree packets, called Bridge Protocol Data Units (BPDUs). These determine which switch is elected as the Root Bridge, and the role of each port within the network (designated, root, or alternate).

When a switch sends a BPDU, it contains the following information that is used to carry out the negotiation:

- **BridgeID**–this value identifies the switch that has sent the BPDU. It is created by combining a spanning tree specific parameter, called the Switch Priority, with the switch MAC address. The BridgeID is an 8-byte number in which the first two bytes are the priority, and the other six bytes are the MAC address.
- **RootBridgeID**–this value identifies the switch that the switch who is sending the BPDU currently believes to be the Root Bridge of the network. Like the BridgeID, this is created by combining the Switch Priority and MAC address of the switch believed to be the Root Bridge.
- **PathCost**–this is a number that represents the logical "cost" of the path from the sending switch to the root bridge. The section "Topology creation" on page 39 describes the process used to calculate the cost.

- **PORTS={port**- a spanning tree specific parameter, that indicates the switch's preference for a port to go into the forwarding state. The BPDU contains the priority value assigned to the port from which the BPDU was transmitted.
- **PortID**–the port number of the port from which the switch transmitted the BPDU.

Root Bridge election

When a switch is first turned on, it assumes that it is the Root Bridge. It sends out BPDUs with its own Priority and MAC address in the RootBridgeID field. But, if it receives BPDUs from other switches that have a lower RootBridgeID value, then it starts sending out that switch's details in the RootBridgeID field of its BPDUs. If it then starts to receive BPDUs with an even lower value, it will start using this new value in its BPDUs.

Eventually, once enough BPDUs have been exchanged, everyone will be in agreement about who the Root Bridge is, and will all be sending that switch's details in the RootBridgeID of their BPDUs. At that point, the Root Bridge has been elected (note that if the Switch Priority is the same on two switches, then the switch with the lowest MAC address becomes the Root Bridge).

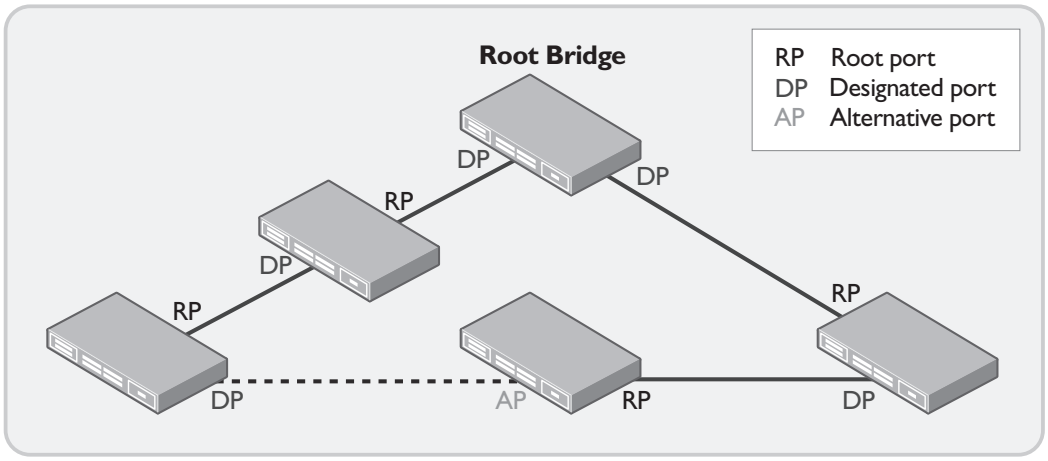
Topology creation

During the BPDU exchanges, the switches also calculate the path cost from each port to the Root Bridge. This calculation is used to determine the topology for the tree, and what roles each port has–either root, designated, alternate, or backup (for information on the backup port role, see "Non-negotiated rapid transitions" on page 44).

The root port for each switch is the port with the lowest path cost to the Root Bridge. The switch calculates this value for a port by taking the root path cost value in the BPDUs received on the port, and adding the path cost value for the port (the mechanism for determining a port's path cost value is described in "Calculating path costs" on page 40). The port with the lowest cost to the Root Bridge becomes the root port. If two ports have the same path cost to the Root Bridge, then the port receiving the BPDUs containing the lowest port priority is chosen. If there is still no clear winner, then the tie-breaker is to choose the port which is receiving BPDUs containing the lowest port number.

The path cost is also advertised by the switch, along with the Root Bridge. Each time the switch receives a BPDU, it checks whether the advertised Root Bridge is better than the one it is advertising, and it checks the path cost value received for that Root Bridge. In this way, each switch will eventually advertise the path cost from itself to the Root Bridge, and will have determined its root port, which always goes into forwarding.

Once a switch has determined its root port, then its other ports are given the correct roles. Normally, any port that is attached to a root port becomes a designated port, and that port goes into port forwarding. However, not all links will have a root port. Often in these cases, the switch at one end of the link will have a lower path cost to root. The port on that switch becomes the designated port on the link, and the port at the other end of the link becomes an alternate port, and goes into blocking.



If both switches on a link have the same path cost to root, the tie-breaker is to compare their BridgeIDs. The switch with the lower BridgeID value wins; its port becomes the designated port on that link, while the port at the other end becomes an alternate port. This case is illustrated in the diagram above.

Calculating path costs

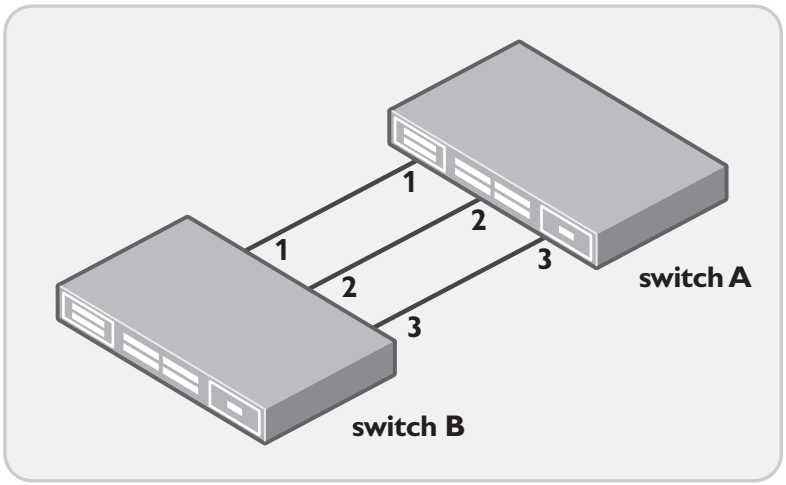
Typically, the path cost on a port is inversely proportional to the bandwidth of the link. The purpose of this is to try to have high-bandwidth links forwarding, and low-bandwidth links discarding. So, the higher the bandwidth on the link attached to a port, the lower the path cost associated with that port. By default, the path costs that RSTP associates with different bandwidths are:

Port speed	Default path cost
Less than 100 Kb/s	200,000,000
1Mbps	20,000,000
10Mbps	2,000,000
100 Mbps	200,000
1 Gbps	20,000
10 Gbps	2,000
100 Gbps	200
1 Tbps	20
10 Tbps	2

If you want the spanning tree to use a path that would not necessarily be chosen when using the default path costs, then you can manually alter the path costs of ports using the **spanning-tree path-cost** command in interface configuration mode, see "Manipulating which ports become root ports" on page 50

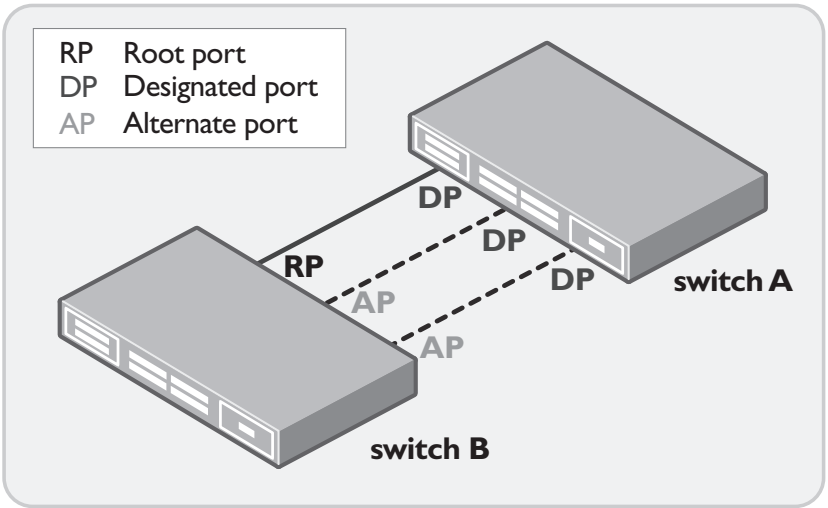
Topology for multiple non-aggregated links

Spanning tree can also negotiate the connection for multiple independent (non-aggregated) links between the same two switches.



In the previous diagram, it is not possible to have all the links forwarding. This problem is resolved by switch B selecting one root port, and making the other ports alternate ports that go into blocking. When selecting the root port, if the path costs for each link are the same, then the switch chooses the port that is receiving BPDUs with the lowest port priority. If there is still no clear winner, then the next tie-breaker is to choose the port receiving the BPDUs containing the lowest port number. Note that the tie-breakers are not the ports' own priority and port number, but the priority and port number in their received BPDUs - i.e. the priority and port number of the ports to which they are connected.

The ports on switch A remain designated ports, and the one attached to the root port will transition to forwarding using the rapid transition process, while the other designated ports will eventually go into forwarding using the slow transition process. The next diagram shows the port roles and link status once the tree has converged.



It is possible to change the choice of which port goes into forwarding, by altering the port priorities. The default priority value for all ports is 128, and the command **spanning-tree priority** (in Interface mode) enables you to alter the priorities. The lower a port's priority, the better the chance that the port it connects to on the neighboring switch has of being chosen to be the Forwarding port. For more information, see "Manipulating which ports become root ports" on page 50

Port state transitions

Within a spanning tree network, ports can be in one of 3 states:

- **Discarding**–the port will send and receive BPDUs, but will drop all other packets.
- **Learning**–the port still does not forward anything but BPDUs, but it does learn forwarding database (FDB) entries from the source MACs of non-BPDU packets it receives.
- **Forwarding**–the port is in a completely operational state; it sends and receives packets of all types.

When a spanning tree negotiation process begins, all the ports on each switch are in the discarding state, and will only exchange BPDUs. As the tree converges, and switches work out which ports should go into a forwarding state, the switches transition those ports from discarding to forwarding.

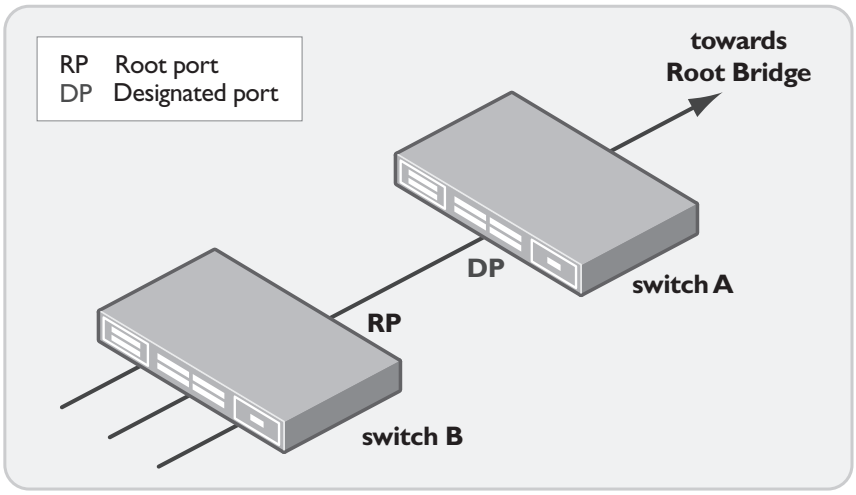
Ports do not transition from discarding to forwarding instantly, as this must be performed in a controlled manner. The switches negotiate the transition by using a Proposal bit in BPDU messages. Depending on the response a switch receives, the ports can transition either quickly or slowly:

- a rapid forwarding transition (described next) occurs when the switch receives a reply to its proposal message.
- a slow forwarding transition (described on page 44) occurs when the switch receives no replies to its proposals.

There are also instances when a switch can rapidly transition to forwarding without requiring any negotiation process. These transitions can only occur on edge, alternate, or backup ports. See the section "Non-negotiated rapid transitions" on page 44 for more information.

Rapid forwarding transition

The rapid forwarding transition requires a brief negotiation between the ports at either end of a link. The following example illustrates this negotiation. In this example, a root port on one switch is connected to a designated port on another switch, as shown in the next diagram.



The sequence of events in a rapid transition are as follow:

1. Both the ports begin in the discarding state. When a designated port is in a discarding state, it sets a bit, called the Proposal bit, in the BPDUs that it sends.
2. When the root port of switch B receives a BPDU with the Proposal bit set, it goes into a state known as "sync". In sync state it ensures that any of its own designated ports are in the discarding state, and then puts its root port into forwarding.
3. Switch B then sends to switch A a special BPDU, called an agreement message. This message is an exact copy of the BPDU that it received from switch A, except that the Proposal bit is unset, and the Agreement bit is set instead.
4. As soon as switch B receives the agreement message from switch A, it immediately puts the receiving port into forwarding.

It is the switch which has the designated port on a link that starts this negotiation, and it only does this once it has completed the same negotiation with the switch upstream of its own root port. So within a spanning tree instance, the Root Bridge is the first switch to begin forwarding, followed by the switches directly connected to it. This means that in the example above, switch B can only begin negotiating forwarding with switches connected to its designated ports once it has negotiated forwarding with switch A.

Slow forwarding transition

If a designated port sends out BPDUs with the proposal bit set, but does not receive any reply, it will fall back to a slow forwarding transition. The process is:

1. The switch listens for replies for a period known as the forwarding delay time (by default this is 15 seconds).
2. If it still has not received a BPDU during this period, the port then goes into the learning state for another period of forwarding delay time.
3. After this second waiting period, the port goes into the forwarding state.

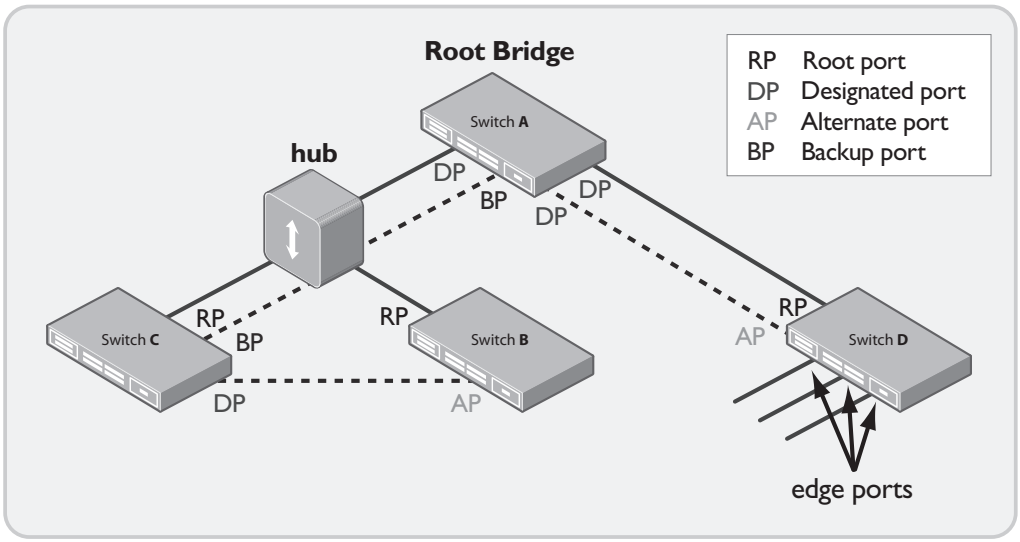
The main reasons why the port would fail to receive replies to its proposal BPDUs would be because the neighbor port is an alternate port, or because the attached device is not configured for RSTP (for example, this would happen if the port was connected to a PC, but the port had not been configured as an edge port).

Non-negotiated rapid transitions

There are three circumstances where a port will make a rapid transition to forwarding without even passing through the Proposal bit negotiation process.

- **Edge ports**
  Ports that are not connected to a switch, but are connected to PCs, printers, or other devices, are called edge ports. Normally, you would explicitly configure a switch so that it is aware of which ports are edge ports. Any port configured as an edge port goes into forwarding as soon as it comes up.
- **Alternate ports**
  This is an alternative to the root port that stays in the discarding state while the root port is active. However, if the root port on switch B (in the next diagram) went down, then switch B would put the alternate port into the forwarding state immediately.
- **Backup ports**
  When a switch has multiple links within a LAN (for example, between itself and a hub), then only one link should remain active, while the other links must go into blocking so as not to create a loop. The ports that are blocked are called backup ports. A switch knows to put a port into the backup port role when it receives BPDUs from itself on that port. If the forwarding port in that set of parallel links goes down, then the lowest numbered of the backup ports will immediately transition to forwarding.

The following diagram shows the relationship between the port roles and link status in an example spanning tree.



Using Hello BPDUs to detect link failures

Once the network has converged, RSTP needs to be able to react to changes in the network topology, so that it continues to maintain an optimal forwarding tree. The key mechanism that detects these changes is the continual sending of hello BPDUs by the switches.

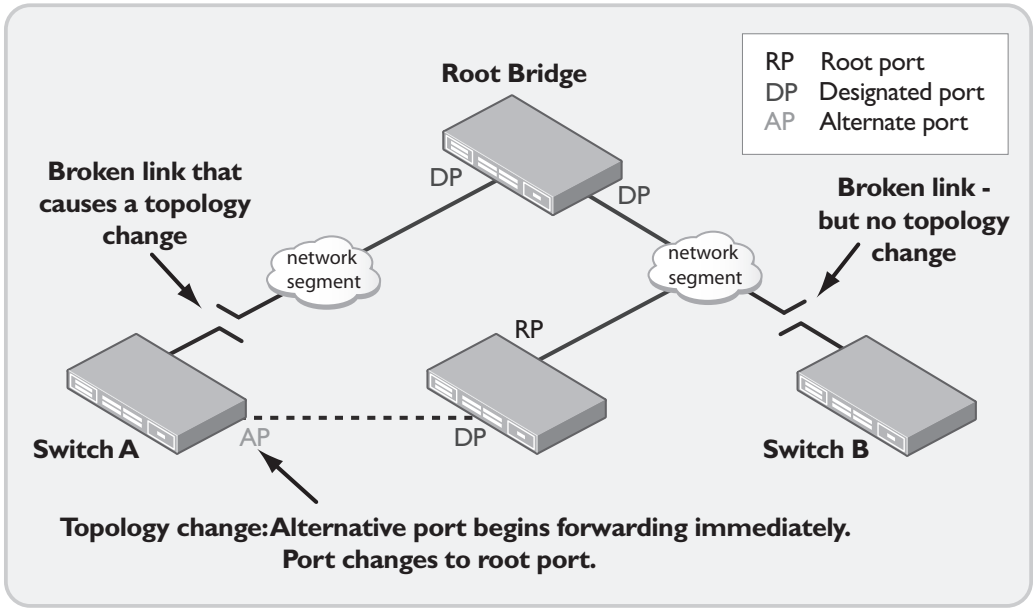
The Root Bridge continues to send hello BPDUs (with a path cost of 0) out all of its active STP ports, at a period known as the Hello Interval. When the neighboring bridges receive the BPDU, they know that the Root Bridge is still active, and transmit hello BPDUs out their designated ports with the correct RootBridgeID and path cost value for their link. The attached switches continue this process of sending hello BPDUs down designated ports, so that each switch in the network receives a BPDU on its root port. These hellos are sent at regular intervals, and confirm for each switch that the tree topology is still the same.

Even if a switch stops receiving hellos on its root port, it will continue to send hellos out its designated ports every Hello Interval. Initially, it will send the original BPDU information it received on the root port. However, if a switch fails to receive any Hello for three successive hello intervals, or if the root port itself goes link down, then it takes action. How this action affects the network depends on the other links that the switch has to the network; if it has no other links, then the switch simply drops off the network; if it has other links to the spanning tree, then the spanning tree as a whole needs to take action. The details of this action are described below.

Reacting to topology changes

A key concept in the RSTP change detection process is a **topology change**. A topology change is defined as a "non-edge port moving to the forwarding state".

Not every link state change is considered a topology change. For example, if a switch has only a single link to the spanning tree and that link goes down, no port transitions to forwarding as a result and therefore this is not deemed a topology change. For example, in the next diagram there are two broken links, but only one topology change.



When a switch detects a topology change (that is, one of its non-edge ports transitions to forwarding), it does two things:

- from its FDB, it flushes all MAC addresses that have been learnt on its non-edge designated ports and its root port.
- it starts a timer (called the TC While timer) that is equal to twice the Hello Time. While this timer is active, all the BPDUs that the switch sends will have the TC bit set. These BPDUs are sent out the root port as well as the designated ports.

The TC bit in the BPDUs lets the switch notify its neighbors that there is a topology change. When each of the neighbors receives a BPDU with the TC flag set, they react as follows:

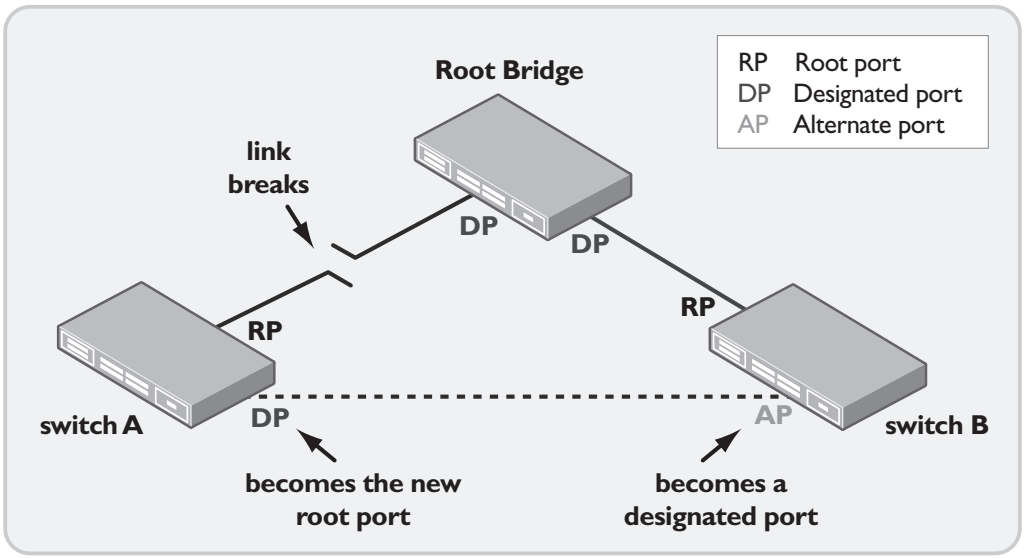
- they clear the MAC addresses learned on all their ports, except the one that receives the BPDU that had the TC flag set.
- they start their own TC While timer and send BPDUs with the TC bit set on all their designated ports and root port.

This system lets the entire spanning tree know that a topology change has occurred. All the switches clear out the details that they had about the path to any given MAC address, and relearn where those paths are in the new topology.

Events that cause a topology change

There are a variety of events that can cause a switch to transition a port to forwarding, and so will stimulate a change in the tree topology. In most cases, the primary event that causes a change is when a switch stops receiving BPDUs on its root port or its root port goes link-down. In these cases:

- if the switch has no other links to the spanning tree instance, then it cannot communicate at all with the tree, and begins considering itself the Root Bridge. However, this will not trigger the topology change process on the network, as no port has moved to forwarding.
- if the switch has an alternate port, it moves this from discarding to forwarding using a non-negotiated rapid forwarding transition. This triggers the topology change process in the network.
- if the switch has a backup port, then this immediately moves to forwarding using a non-negotiated rapid forwarding transition. This triggers the topology change process.
- if the switch has a designated port, but not an alternate port, then a negotiation occurs. The following diagram shows an example network where switch A has a designated port that it must start forwarding successfully out of. However, the alternate port attached to it will drop user traffic unless a negotiation takes place.



In this example, switch A begins generating BPDUs to say "I am root". When switch B receives the BPDU, it sends back a BPDU with the details of the original (and superior) Root Bridge. When switch A receives the BPDUs from switch B, it recognises that there is a better Root Bridge, and changes its designated port to a root port. After that, the root path cost in the BPDUs that switch A sends to switch B will be greater than switch B's own root path cost. Switch B realises that it now has a better root path cost than switch A, and changes its alternate port to a designated port. Once that port transitions to forwarding, the topology change process is triggered across the rest of the network.

The other event that can cause a topology change is the connecting of a new link in the network. The switches on the link will exchange BPDUs on this link, and determine a role

for their port at each end of the link. One or both of the ports will eventually go into forwarding, triggering the topology change process on the network.

Configuring RSTP using the AlliedWare Plus OS

By default, the AlliedWare Plus operating system enables RSTP on all the ports of a switch. However, you will manually need to configure some aspects of RSTP to meet your network's needs.

Forcing a switch to become the Root Bridge

The Root Bridge plays a special role in a spanning tree–it is the one switch that is guaranteed to always have all it ports in the forwarding state; it is the switch from which the hello packets radiate outward; and it is the switch to which all other switches measure their "path cost to root". You typically want to specify which switch will become the Root Bridge, and which switch should take over as root if the original Root Bridge goes down.

To make sure that a specific switch becomes the Root Bridge, you need to ensure that it has the lowest priority value in the network (see the section "Root Bridge election" on page 39 for more information about the election process). By default, all switches have a priority value of 32768 (8000 in hexadecimal notation) and priority values must be multiples of 4096, with zero as the lowest possible priority value. So, the easiest way to force a switch to become the Root Bridge is to set its priority to zero. To do this in the AlliedWare Plus OS, use the global command:

```
awplus(config)#spanning-tree priority 0
```

Do keep in mind, of course, that if more than one switch in the network has priority 0, then the election of root will be based on the MAC addresses of those switches.

If you want to designate another switch as a backup to the Root Bridge, then configure it with the next lowest possible priority value of 4096.

Configuring the edge ports

Once a pair of switches are connected together, and the ports connecting them have gone link-up, they will stay continuously link-up for long periods of time. So, the link topology between switches very rarely changes state.

However, the switch ports at the edge of the network (where the workstations connect) are likely to change state more than once a day. Every time a workstation is turned on or off, or reboots, its link to the network switch will change state between link-up and link-down a few times. State changes on active spanning tree ports usually triggers the topology change process in the spanning tree, which result in switches flushing their forwarding database tables and then flooding packets as they relearn MAC addresses.

This process can disrupt the smooth running of the network, so it is beneficial to the network to stop the edge ports from operating as active spanning tree ports. As well, it is rare for network loops to exist beyond those edge ports as each port is normally connected to just one terminating device.

For these reasons, the **PortFast** feature exists in RSTP. This is a defined setting that is tailored specifically for the requirements of edge ports. It puts the ports into a state where they:

- immediately go into the forwarding state as soon as they go link-up
- do not generate topology change events when they go into forwarding

These ports will still emit hello packets, on the off-chance that a loop is inadvertently formed. If they receive BPDUs then, depending on the port's configuration, they will either become active STP ports or they will shut down.

To enable the PortFast setting on a port, enter the port's configuration mode and use the command:

```
awplus(config-if)#spanning-tree portfast
```

To configure the port to shut down if it receives BPDUs, specify the **bpdu-guard** option:

```
awplus(config-if)#spanning-tree portfast bpdu-guard
```

If you do not use this option, then if the port receives a BPDU it will begin to negotiate spanning tree with the device sending BPDUs.

Protecting against Root Bridge spoofing attacks

Root Bridge spoofing is a data-stealing and denial-of-service attack. The attacker sets up a device that emits STP hello packets with a very low priority to enable their device to get elected as the Root Bridge. Once it is elected, the attacker will be able to see a lot of the data on the network, allowing them to steal that data. Root Bridge spoofing also disrupts the network. To protect against this attack, you can configure specific ports on each switch to shut down if they receive BPDUs with a lower priority than the switch's own Bridge ID. You should configure this setting only on ports that you know should never be root ports on the switch.

To enable this feature on a port, enter the port's configuration mode, and use the command:

```
awplus(config-if)#spanning-tree guard root
```

Manipulating which ports become root ports

A switch determines its root port by looking at the path cost from each port to the Root Bridge, and selecting the port with the lowest combined cost as its root port. If you want one particular port to become the root port, then you can set a low path-cost value on the preferred port and set high path-cost values on the competing ports.

To set the path cost for a port, enter the port's configuration mode, and use the command:

```
awplus(config-if)#spanning-tree path-cost <pathcost>
```

Note that this sets the value for the port only–the full root path cost for a port is calculated by adding this value to the path cost value found in the BPDUs received on this port.

In order to affect the decisions that are made in the 'tie-breaker' situations where a switch has multiple ports with the same lowest cost to root, you can explicitly configure port priorities. To configure the priority value on a port, enter the port's configuration mode, and use the command:

```
awplus(config-if)#spanning-tree priority <priority>
```

It is important to remember that altering the priority on a port does not change its likelihood of becoming a root port; it affects the decision made on the neighbor switch that this port is connected to. The lower the priority on a port, the more likely that the port it is connected to will become the root port for that neighbor switch.

Disabling RSTP on a single port

In some scenarios, it can be useful to disable a spanning tree protocol on a per port basis. There is no specific command to do this in the AlliedWare Plus OS, however, you can use the PortFast setting and BPDU-filtering to achieve the same effect. The sequence of commands are:

1. Enter the configure terminal mode:

   ```
   awplus#configure terminal
   ```

2. Enter the port you want to configure:

   ```
   awplus(config)#interface port1.0.1
   ```

3. Configure the port as an edge port by using the PortFast setting:

   ```
   awplus(config-if)#spanning-tree portfast
   ```

4. Enable the BPDU filter on that port. This filters out any received BPDUs and prevents any from being sent:

   ```
   awplus(config-if)#spanning-tree portfast bpdu-filter enable
   ```

You will get the following warning message on the console:

```
% Warning:Ports enabled with bpdu filter will not send BPDUs and drop
all received BPDUs.
 You may cause loops in the bridged network if you misuse this feature
```

To all intents and purposes, spanning tree is no longer operating on the port. The port transitions quickly to a designated port that is forwarding, and remains in this state.

Monitoring and Troubleshooting RSTP

The following section will help you to better understand and troubleshoot RSTP issues.

Problems that can occur on spanning trees

By far the most common spanning tree problem is a switch failing to receive or process incoming BPDUs for some reason. The reasons are such as:

- if the link stops functioning correctly, or congestion occurs on the link from the switch chip to the CPU.
- some software failure stops the STP software from operating correctly.

When this occurs, the maximum age timer will time out and the switch will decide it is the root, and so will put all of its ports into the forwarding state. If one or more of its ports should be blocking, then this will cause a loop in the network, usually resulting in a network storm.

The next most common spanning tree problem is a bug in a vendor's implementation of the STP algorithm, causing one or more switches to fail to put ports into blocking when they should have been. This also can cause a loop in the network, resulting in a packet storm.

Common configuration **mistakes** are:

- not configuring the workstation-connected ports as edge ports (PortFast ports). This mistake causes topology change notifications to be advertised into the network every time a PC is switched on or off. This results in a lot of unnecessary flooding of traffic that can cause significant congestion at the start or end of the working day (when everyone is turning their PCs on or off).
- connecting a non-STP port (port that does not have STP enabled) into the active STP topology
- setting timers too low in a misguided attempt to create a network that will converge more quickly
- creating a network with a larger diameter than the protocol can support. The diameter is the maximum number of switches between any two points of

attachment of end stations. The IEEE recommends that the maximum diameter of the tree is seven bridges when using the default STP timers.

Another problem that is sometimes seen is a case where too many ports go into blocking, causing a part of the network to be cut off from the rest.

Debugging spanning trees

The recommended steps to investigate any of the issues described are as follows:

▶ Develop an accurate network diagram.

The most important thing to do before embarking on debugging an STP issue is to get an accurate diagram of the exact topology of the network, so that you know exactly where every switch port connects to. This accurate diagram is essential, as the operation of RSTP is intimately connected with the network topology.

Once you have an accurate network diagram, the next debug steps are aimed at determining:

- the location of the Root Bridge
- the location of the blocked ports and the redundant links

This knowledge provides the initial overview of the state that RSTP has put the network into.

▶ Compare the current network topology with the normal topology.

It is always useful to know the location of the Root Bridge switch and the blocking ports in the network's normal state. Discrepancies between the normal state and the state in front of you are excellent clues as to the location of the problem.

The first step in getting this information is to get the spanning tree state of all the active spanning tree ports on all the switches in the network.

The command to get the spanning tree state of all ports on a switch is:

```
show spanning-tree
```

To see the state of a single port, the command is:

```
show spanning-tree interface <port-number>
```

The following diagram shows example output for the command **show spanning-tree interface port1.0.1**:

```
Default: Bridge up - Spanning Tree Enabled
Default: Root Path Cost 0 - Root Port 0 -  Bridge Priority 0
Default: Forward Delay 15 - Hello Time 2 - Max Age 20
Default: Root Id 00000000cd27c0b3
Default: Bridge Id 00000000cd27c0b3
Default: 16 topology change(s)  - last topology change Thu Jul 24
22:00:44 2008

Default: portfast bpdu-filter disabled
Default: portfast bpdu-guard disabled
Default: portfast errdisable timeout disabled
Default: portfast errdisable timeout interval 300 sec
  port1.0.1: Port 5001 - Id 8389 - Role Designated - State Forwarding
  port1.0.1: Designated Path Cost 0
  port1.0.1: Configured Path Cost 20000 - Add type Explicit ref count 1
  port1.0.1: Designated Port Id 8389 - Priority 128 -
  port1.0.1: Root 00000000cd27c0b3
  port1.0.1: Designated Bridge 00000000cd27c0b3
  port1.0.1: Message Age 0 - Max Age 20
  port1.0.1: Hello Time 2 - Forward Delay 15
  port1.0.1: Forward Timer 0 - Msg Age Timer 0 - Hello Timer 1 - topo
change timer 0
  port1.0.1: forward-transitions 3
  port1.0.1: Version Rapid Spanning Tree Protocol - Received RSTP - Send
RSTP
  port1.0.1: No portfast configured - Current  portfast off
  port1.0.1: portfast bpdu-guard  default  - Current portfast bpdu-guard
off
  port1.0.1: portfast bpdu-filter default  - Current portfast bpdu-
filter off
  port1.0.1: no root guard configured     - Current root guard off
  port1.0.1: Configured Link Type point-to-point - Current point-to-
point
```

▶ Check for discrepancies between each switch's understanding of the network.

The things to check for are:

- do all the switches agree on the MAC address of the Root Bridge?
- do both ports at each end of each link in the network agree on which is the designated port?
- do all the switches agree on the timers (hello time, forward delay, max age)?
- are there enough ports in the blocking state? A quick way to get a summary of the states of all the ports on the switch is to use the command:

  ```
  show spanning-tree | grep State
  ```

- is the Root Bridge switch and the blocking ports in the same locations as they are when the network is in its "normal" state?

Those questions are usually enough to take you to where a problem is happening–a switch that incorrectly thinks that it is the Root Bridge; a port that should be blocking but is not; a port that is blocking when it should not be.

But, if the location of the problem is still not evident, then more investigation is required. The things to then go on to investigate are:

BPDU flow

- is the Root Bridge sending out BPDUs at the correct regular intervals?
- are the other bridges forwarding the BPDUs on correctly (through their designated ports)?
- are the switches correctly incrementing the path cost and message age in the BPDUs before forwarding them on?

The possible methods for seeing the BPDUs in the network are:

- sniffing the links in the network.
- using STP packet debugging. You can set packet debugging to display output that is very granular (down to just the packets received or the packets sent on a single port). Packet debugging can also display the packets in decoded form. For example, to set packet debugging to decode STP packets received on port1.0.19, use the command:

  ```
  debug mstp packet rx decode interface port1.0.19
  ```

  Note that although this command uses the keyword **mstp**, it displays debugging output for the RSTP and STP protocols as well the MSTP protocol.

  Some care does need to be taken with STP debugging, as the output can be very verbose, making it difficult to enter the commands to disable the debugging again. For this reason, when you enable terminal monitor, to see the debug, it is advisable to enable terminal monitor for a specified number of seconds, with the command:

  ```
  terminal monitor <seconds>
  ```

Topology changes

One class of Spanning Tree problem is network slowness caused by excessive topology change notifications. This is caused by switches flushing their FDB upon receipt of topology change notifications, causing data flooding that results in network congestion. To check for excessive topology changes:

- look for topology change notifications by examining the BPDUs being exchanged. You can identify the notifications by looking for BPDUs with the TC flag set.
- alternatively, you can look at the following line in the **show spanning-tree** output:

```
16 topology change(s)  - last topology change Thu Jul 24
22:00:44 2008
```

If in doubt - debug all

If none of the above approaches have revealed the location of the problem, then enable all debugging on all the switches at the same time (if possible) for a minute or so, and capture all the output. That will give a snapshot of the complete STP activity in the network for that period. To do this, use the command:

```
debug mstp all
```

For further troubleshooting advice, refer to "Chapter 10 | Troubleshooting" on page 189

Chapter 4 | Static and Dynamic (LACP) Link Aggregation

Introduction

Link aggregation is the process where two or more ports in an Ethernet switch are combined together to operate as a single virtual port. A link aggregation can exist only between a pair of neighboring switches, the ports that are aggregated on one switch must not be connected to unaggregated ports on the other switch. A switch can have multiple link aggregations–to different neighbors, and even to the same neighbor if the network is loop protected.

Link aggregation is a key component in resilient network design, as it both increase the available bandwidth between devices, and provides continuity of connectivity if one link is broken. This makes it a popular feature in LAN networking, and this means that it is important for a LAN support engineer to understand and troubleshoot link aggregation. In particular, understanding how traffic is shared across the links in an aggregation is important as this is often misunderstood, leading to mistaken expectations about how an aggregation should perform.

As well as understanding how link aggregation works, an engineer also needs to know how the AlliedWare Plus operating system treats the link aggregation virtual interfaces. Understanding which commands you should apply to the virtual interface, and which you should apply to the component ports, will help you to avoid configuration errors.

This chapter describes the two mechanisms for link aggregation on Allied Telesis devices (**static** and **dynamic** using LACP), and some generic features of link aggregation–how traffic is shared across the links, and what happens when a port is dropped out of the aggregation. It then explains in detail how Link Aggregation Control Protocol (LACP) works, including the packet structure of LACP messages, and how LACP negotiates and monitors links. Next it explains how aggregated interfaces are treated in devices running the AlliedWare Plus OS. The chapter finishes with details on how to troubleshoot link aggregation issues.

Overview

Link Aggregation is a key component of resilient network design. The benefits of Link Aggregation are:

- **increased bandwidth**

  By aggregating two or more links together, you can increase the bandwidth between neighbor devices. This is effectively additive, two links between neighbors gives up to twice the bandwidth of one link.

- **resiliency**

  Having more than one link to a neighboring device provides connectivity if one of the links breaks. A feature of this resiliency is the speed at which link aggregation reacts to the change–just a matter of milliseconds.

The two methods that you can aggregate links on Allied Telesis switches are **static** aggregation, and **dynamic** aggregation using LACP. The differences between the two methods are described in "Static Aggregation versus Dynamic Aggregation" on page 60. Both static and dynamic aggregation use the same generic method for sharing traffic across the links and deciding how to forward traffic if a port goes link-down, as described in the next two sections.

How aggregated links share traffic

From the point of view of switching, a link aggregation is treated as a single port. For example, learnt MAC addresses are associated with the aggregation, not the individual ports within it; ARP entries are associated with the aggregation; and so on. The Layer 2 and Layer 3 switching processes do not decide which port within the aggregation should transmit a given packet; the switching processes simply deliver the packet to the aggregation, and let a process within the aggregation decide which port to transmit on.

This means that link aggregation needs a process for deciding which port in an aggregation to transmit a packet across. The process must:

- spread traffic across the ports in a fairly even manner, so that the maximum amount of bandwidth is used.
- operate very fast–at wire speed.
- not need to keep a recorded history–for example, it should not need to know what port the last packet was transmitted across.

The process that fulfils these criteria is hashing. Certain fields are extracted from each packet and fed into a hashing algorithm from which a single number results. The number indicates which member port of the aggregation the packet will be transmitted through.

The fields that the algorithm can select from are the source and destination:

- MAC address
- IP address
- TCP port

The switch has multiple hashing algorithms to support variation in the number of ports in a link. A two-port aggregation must use a hashing algorithm that generates a result that is either '1' or '2'. A three-port aggregation must use an algorithm that generates a result that is between 1 and 3; and so on. These algorithms are implemented in the switching hardware and operate very quickly. If the traffic being passed to the aggregation is sufficiently varied (has a variety of source and destination addresses) then this process will result in a relatively even sharing of traffic across the component ports of the aggregation.

Using a hashing algorithm for this process also means that traffic streams with the same destination and source details are always sent down the same link. For example, a specific TCP session between two devices will always have the same MAC address, IP address, and TCP port details, so the algorithm will always send the packets to the same link. This is beneficial to the stream–any variation in the transmission delay of the different links could cause out-of-order packets or jitter, as packets from one link arrived faster than the other link. In most networking equipment, including Allied Telesis devices, you can choose which fields are fed into the hashing algorithm. See the section "Configuring Link Aggregation using the AlliedWare Plus OS" on page 66 for details about changing the fields in the AlliedWare Plus OS.

What happens when a port leaves the aggregation?

A powerful feature of link aggregation is the speed with which it can recover from the loss of a link. As soon as the switch chip detects that a port has gone down (or has been removed by the LACP keep-alive process) it simply needs to change over to using a different hash algorithm to choose the egress port for packets. For example, if a 3-port aggregation loses a port, the chip immediately begins using the 2-port hashing algorithm. Since this process occurs entirely within the switching hardware, it is performed very quickly–within a few milliseconds.

Similarly, when a port rejoins the aggregation, the chip simply begins to use a new hashing algorithm. This means that the aggregation very quickly begins using the newly joined port.

Properties that an aggregation's ports must share

There are a set of properties that a set of ports must have in common before they can form an aggregation. The ports must all:

- have the same speed and duplex settings
- be members of exactly the same set of VLANs
- have the same STP path cost.

Static Aggregation versus Dynamic Aggregation

There are two ways you can aggregate links on Allied Telesis switches:

- **Static aggregation**

  Static aggregation is achieved by explicitly configuring the ports as aggregated links, and must ensure that their peers ports on the neighbor switch are similarly configured. Traffic is shared across whichever of these ports are link-up at any given moment.

- **Dynamic aggregation using LACP**

  Link Aggregation Control Protocol (LACP) is a control protocol that allows you to dynamically aggregate links by negotiating links and checking link communication. With LACP, a set of ports are marked as able to aggregate. The switch then aggregates the ports if they obtain agreement from the neighbor device at the other end of the links. Synchronisation with the neighbor device is regularly checked on each link; if communication is lost on any link, it is dropped out of the aggregation, even if it is still in a link-up state.

  LACP is defined by the IEEE standard 802.3ad.

Choosing an aggregation method

There are distinct **advantages to using LACP** in a complex environment, or over long-distance links:

- LACP enables the switch to identify the neighbor at the far end of each link, and only aggregates ports that are connected to the same neighbor. If you make a cabling mistake and connect some links to the wrong neighbors, then LACP will detect the mistake.
- LACP acts as a keep-alive protocol that can detect a failure somewhere in the end-to-end link between two neighbor switches. For example, if devices like media convertors are present within the link, it is possible for communication on the link to fail without the ports at both ends of the link going down. Static aggregation depends on a port-down event as a signal to drop a port out of the aggregation; the LACP keep-alive mechanism enables a dynamic aggregation to drop out any link on which communication is failing, regardless of whether the attached port goes down.

Static aggregation is best used only in a simple and unchanging environment, where the neighbor switches are close together (such as within the same rack) and you can ensure that the correct ports are connected to each other. In this scenario static aggregation would keep the setup as simple as possible.

Understanding LACP

With **static** link aggregation, there is not a great deal to understand–the switch treats the aggregation as effectively a single port, and uses a hashing algorithm to share data across whichever member ports are link-up. Dynamic link aggregation however requires the communication protocol LACP.

LACP Definition

LACP is a control protocol that automatically detects multiple links between two LACP enabled devices and enables them to use their maximum possible bandwidth by automatically aggregating the links together.

LACP modes and Data Units

Each aggregation link created by LACP is referred to as a dynamic channel group. Ports in a dynamic channel group can be in one of two modes:

- **LACP active** mode

  A port in active mode sends LACP Data Units (LACPDUs) at regular intervals to seek out partners.

- **LACP passive** mode

  A port in passive mode only begins sending out LACPDUs in response to a received LACPDU.

When ports are added to a dynamic channel group, each connected active mode port sends LACPDUs to find any partner devices that also have LACP enabled. If a port receives a reply, the switch uses the reply to help build a map of connected LACP partners and the links that they share. When LACP detects that two or more links are connected to the same partner, and have the same key (the channel group ID), it aggregates them into one logical link.

When you add any further physical links to the same partner system, the links are added to the already existing aggregation (within the hardware limits on each switch).

Ports in passive mode do not actively seek out LACP partners. However, if a passive mode port receives an LACPDU, then the device it is on includes the new link details in a map of connected LACP partners, and begins sending LACP control packets out of the interface.

Details of the protocol exchange

The LACP protocol involves the exchange of LACPDUs between the devices at either end of a link. Each switch refers to itself as the actor and refers to the switch that it is negotiating with as its partner. The information that the switches exchange is shown in the next table.

Items	Description
Port number	The port that the LACPDU was emitted from.
SystemID	A value used to uniquely identify a switch, so that each switch can work out which of its ports are connected to the same neighbor. The SystemID also acts as a priority value, to enable neighboring switches to decide which switch's configuration takes priority if neighbors share more ports than they can aggregate together. See the section "System Priority" on page 65 for more information.
Port key	Used to identify groups of ports as being available to aggregate together. Ports that are in the same dynamic channel group must all have the same value.
Timeout setting	This indicates the rate that the switch expects to receive LACPDUs from its partner.
Port's aggregation setting	Whether the port is allowed to become part of an aggregation, or must stay individual.
Port's synchronisation setting	If the LACP parameters on the device have recently been updated, but the new settings have not yet been committed to the hardware controlling this port, then the synchronisation status will be "out-of-sync"; otherwise it will be "in-sync".
Port's collecting status	Whether or not the port is allowed to receive packets.
Port's distributing status	Whether or not the port is allowed to transmit packets.

Each LACPDU packet contains an **actor** section, in which the switch sends its own values for those parameters, and a **partner** section, in which it sends its current view of the neighbor's values for these parameters. The next output (collected by a packet sniffing program) shows the contents of an LACPDU for a port that the LACP neighbors have not yet established as part of an aggregated link.

```
Link Aggregation Control Protocol
    Slow Protocols subtype: LACP (0x01)
    LACP Version Number: 0x01
    Actor Information: 0x01
    Actor Information Length: 0x14
    Actor System Priority: 32768
    Actor System: Allied_5f:f5:80 (000a.b75f.f580)
    Actor Key: 2
    Actor Port Priority: 32768
    Actor Port: 5
    Actor State: 0xc5 (Activity, Aggregation, Defaulted, Expired)
        .... ...1 = LACP Activity: Yes
        .... ..0. = LACP Timeout: No
        .... .1.. = Aggregation: Yes
        .... 0... = Synchronization: No
        ...0 .... = Collecting: No
        ..0. .... = Distributing: No
        .1.. .... = Defaulted: Yes
        1... .... = Expired: Yes
    Reserved: 000000
    Partner Information: 0x02
    Partner Information Length: 0x14
    Partner System Priority: 0
    Partner System: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Partner Key: 0
    Partner Port Priority: 0
    Partner Port: 0
    Partner State: 0x02 (Timeout)
        .... ...0 = LACP Activity: No
        .... ..1. = LACP Timeout: Yes
        .... .0.. = Aggregation: No
        .... 0... = Synchronization: No
        ...0 .... = Collecting: No
        ..0. .... = Distributing: No
        .0.. .... = Defaulted: No
        0... .... = Expired: No
    Reserved: 000000
```

After the exchange of a few LACPDUs, the neighbors will have agreed on each other's status. They will also have verified that each has correctly recognized the other's status, by checking the values that their neighbor has sent in the partner fields of the LACPDUs.

Once the neighbors have agreed about each other's settings on a given pair of ports, then those ports are said to be synchronised. The devices can then make a decision about whether they can add the ports at each end of the link to an aggregation.

The values exchanged and settings established can be seen for a specific LACP port with the command **show port etherchannel <port>**.

Fine tuning LACP

There are various settings that you can adjust to fine tune the operation of LACP on a switch and in a network. This section describes the common settings to adjust.

Mode

You can change the mode on links between either passive or active mode. See "LACP modes and Data Units" on page 61 for more information about the modes.

We recommend that you **always use active mode on links that you want to aggregate**, as there is no downside to using active mode. However, passive mode can have a downside; if both ends of a link are configured for passive mode, then no LACP negotiation will ever occur on that link.

The main purpose for passive mode is to control accidental loops. Passive mode requires one of the neighbor switches to be in active mode.

LACPDU timeout

You can change the rate that your switch expects to receive LACPDUs from its neighbor to either short or long.

- if the timeout is set to **long**, then the switch expects to receive an LACPDU every 30 seconds. It will time a port out of the aggregation if it does not receive an LACPDU for 90 seconds (this means three consecutive LACPDUs are lost).
- if the timeout is set to **short**, then the switch expects to receive an LACPDU every second. It will time a port out of the aggregation if no LACPDUs are seen for three seconds (this means three consecutive LACPDUs are lost).

The switch indicates its preference using the timeout field in the actor section of its LACPDUs. If that field is set to **1**, then the switch has set the **short** timeout. If the field is set to **0**, then the switch has set the **long** timeout.

Setting the short timeout enables the switch to be more reactive to communication failure on a link. It also does not add much processing overhead to a switch–one packet per second, which is insignificant.

Because of this timeout setting, it is not possible to configure the rate at which the switch sends LACPDUs. A switch must always send LACPDUs at the rate which its neighbor has indicated it expects to receive them.

The interface-mode command **lacp timeout {short|long}** enables the switch to indicate the rate at which it expects to receive LACPDUs from its neighbor.

Port priority

This value deals with the situation where the number of ports that have been configured into a dynamic channel group is larger than the number that the switch hardware can accommodate in a single aggregation. The priority value is used to decide which ports should be included into the aggregation. The higher priority ports (those with a lower priority value) are selected ahead of the lower priority ports.

Excess ports are put into a standby mode, leaving them effectively disabled. However, if a link in their channel group goes down, they will take the place of that link.

The LACP port priority can be specified with the interface-mode command **lacp port-priority**. The default value is 32768 (1 is high).

System priority

LACP neighbors can disagree about port priorities, with one switch configured to leave a certain set of links out of the aggregation, while its neighbor is configured to leave out a different set. This means that LACP requires a mechanism to decide which LACP neighbor's port priority setting is used.

LACP combines the System Priority value with the MAC address of the switch to create a System ID. The switch with the lower System ID becomes the master switch, and the other switch must fall into line with whatever decision the master makes.

The system priority of a switch can be set in global configuration mode, using the command **lacp system-priority**.

Configuring Link Aggregation using the AlliedWare Plus OS

The AlliedWare Plus™ O S makes it very easy for the user to truly treat an aggregation as a single port. Within the AlliedWare Plus OS, there are two types of port aggregation:

- A **static channel group**–this is a static port aggregation
- An **etherchannel**–this is a dynamic aggregation, formed using LACP

Once a static channel group or an etherchannel has been created, you can then treat it as an interface in its own right. The aggregation is given an ID number between 1 and 65535, and the interface that represents the aggregation is given a name according to the following conventions:

- For an **etherchannel**, the interface name is formed by combining the letters po with the ID number. For example, the interface that represents the etherchannel with ID 18 is called po18.
- For a **static channel group**, the interface name is formed by combining the letters **sa** with the ID number. For example, the interface that represents the static channel group with ID 18 is called sa18.

Configuring the aggregation link properties

To configure properties on an aggregation, you enter interface configuration mode on the representative interface. For example, to enter interface configuration mode for etherchannel 16, use the command:

```
awplus(config)#interface po16
```

To enter interface configuration mode for static channel group 14, use the command:

```
awplus(config)#interface sa14
```

You can then configure the following properties for the interface:

- VLAN classifiers and VLAN membership
- thrash-limiting
- switchport mode
- storm-control levels
- MAC access-groups and IP access-groups
- default CoS value and QoS service-policies
- egress-rate-limit (this configures the egress rate limit on each port in the aggregation

Some quite port-specific properties, like speed, duplex, polarity, and WRR scheduling, cannot be configured on a link aggregation.

The AlliedWare Plus OS allows you to configure an aggregated link as:

- an igmp snooping mrouter interface
- the port to which to attach a static IGMP entry
- an EPSR primary or secondary port
- the port to which to attach a static MAC entry
- the egress port associated with a static ARP entry
- the port to which to assign an MSTP instance
- the port on which to set spanning-tree pathcost and port priority

Creating a static aggregation

A link aggregation does not need to be explicitly created before you can add ports to it. The first time a port is configured to belong to a static channel group, then the static channel group (and its virtual interface) are implicitly created.

Ports are collected into an aggregation using a command in the port's interface mode. For example, to add port1.0.1 to the static channel group sa16, use the commands:

```
awplus(config)#interface port1.0.1
awplus(config-if)#static-channel-group 16
```

Creating a dynamic (LACP) aggregation

A link aggregation does not need to be explicitly created before you can add ports to it. The first time a port is configured to belong to an etherchannel, then the etherchannel (and its virtual interface) are automatically created.

To add a port to the etherchannel, enter the port's interface mode, for example:

```
awplus(config)#interface port1.0.2
awplus(config-if)#channel-group 24 mode {active|passive}
```

The mode parameter determines whether the port actively looks for an LACP neighbor. See the section "LACP modes and Data Units" on page 61 for more information.

To change the LACPDU timeout setting, (see "LACPDU timeout" on page 64) use the command:

```
awplus(config)#interface port1.0.4
awplus(config-if)#lacp timeout {short|long}
```

To change the port priority, (see "Port priority" on page 65) use the command:

```
awplus(config)#interface port1.0.4
awplus(config-if)#lacp port-priority <1-65535>
```

The default is 32768. The lower you set the number, the higher a port's priority as a link member.

The system priority (see "System priority" on page 65) is set globally, rather than per interface. Use the command:

```
awplus#configure terminal
awplus(config)#lacp system-priority <1-65535>
```

The default is 32768. Like the port priority, the lower you set the number, the higher the system's priority.

Configuring the hashing algorithm on the switch

The hashing algorithm determines which port in an aggregation a packet is sent to. It is set globally, rather than per interface. You can choose one, two, or all three of the fields that the algorithm process uses, with the command:

```
awplus(config)#platform load-balancing {src-dst-mac|src-dst-
   ip|src-dst-port} [src-dst-mac|src-dst-ip|src-dst-port]
```

The default is **src-dst-mac** and **src-dst-ip**. This command changes the hashing algorithm on all ports for both aggregation methods.

The hashing algorithm process is described in the section "How aggregated links share traffic" on page 58.

Operational capabilities

Channel Group limit	The maximum number of channel **groups** that you can create on a x900 or x600 Series switch is **31**.
Ports within a link	The maximum number of **ports** that can be active members of a given channel group on an x900 or x600 Series switch is **8**.

There is no restriction as to the location of a channel group's member ports within a switch. You can aggregate ports in:

- the first 24 ports of an x900 with ports in a XEM
- different XEMs on an x900 or x908
- different switches within a VCStack

Monitoring and troubleshooting static link aggregations

Static link aggregation does not involve any signalling between the neighbor switches, so there is very little you can monitor to check if the aggregation process is operating correctly.

Incorrect cabling or configuration is the most likely cause of any issue with a static link aggregation. If all the links in the aggregation do not terminate on the same neighbor switch, then the transport of data across the aggregation will probably be unpredictable. Common symptoms of a "split aggregation" (that is, not all links terminating on the same neighbor) are:

- some data flows will operate correctly, and others will not. For example, if Layer 4 fields are included in the link choice algorithm, then split aggregation can lead to a situation where two devices can ping each other, but TCP packets from one are never delivered to the other.
- broadcast packets sent out one port in the aggregation could be sent back to another port in the aggregation, and then sent back out to the originating host. Whilst this will not cause a complete broadcast storm, it will cause a noticeable increase in the amount of broadcast traffic in the VLANs configured over the aggregated link. It will also cause a level of MAC thrashing, where the FDB entries for hosts will oscillate between host-connected ports and the aggregated ports.

To see the overall state of a static aggregation, use the command:

```
show interface <interface name>
```

The interface name is the concatenation of "sa" with the channel ID. For example, to see the overall state of the static aggregation 3, use the command:

```
awplus#show interface sa3
```

This will show the following output:

```
stack#show interface sa3
Interface sa3
  Scope: both
  Link is UP, administrative state is UP
  Thrash-limiting
    Status Not Detected, Action learn-disable, Timeout 1(s)
  Hardware is AGGREGATE
  Description: SB1
  index 4503 metric 1 mtu 1500
  <UP,BROADCAST,RUNNING,MULTICAST>
  VRF Binding: Not bound
    input packets 40955, bytes 3875058, dropped 0, multicast packets
19573
   output packets 30034355, bytes 2108501311, multicast packets 124725
broadcast packets 26932613
```

Monitoring and troubleshooting LACP

Checking link status

The command to see the state of a dynamic channel group is:

```
show etherchannel [<1-65535>]
```

Example

To see *etherchannel interface po23*, use the command: **show etherchannel 23**.

The output is as follows:

```
Aggregator po23 (4623)
  Mac address: 00:15:77:c2:4d:53
  Admin Key: 0023 - Oper Key 0023
  Receive link count: 1 - Transmit link count: 1
  Individual: 0 - Ready: 1
  Partner LAG: 0x8000,00-00-cd-04-f2-e0
   Link: port1.0.12 (5012) disabled
   Link: port1.0.13 (5013) sync: 1
   Link: port1.0.14 (5014) disabled
   Link: port1.0.15 (5015) sync: 0
```

The LACP states of each port in the aggregation are shown by the entry at the end of each **Link** line. The meanings of each state are described in the following table.

LACP state	Description
sync:1	The port is up, and LACP has successfully negotiation with the neighbor at the other end of the link.
sync: 0	The port is up, but LACP has not successfully negotiation with the neighbor at the other end of the link.
disabled	The port is down.

You can see more detailed information about the LACP activity on a given port in the output of the command **show port etherchannel <port>**. The output for the command is shown in the next output example.

```
Link: port1.0.12 (5012)
 Receive machine state: Defaulted
 Periodic Transmission machine state: Slow periodic
 Mux machine state: Waiting
 Actor Information:                      Partner Information:
   Selected ................. Standby     Partner Sys Priority ....... 0x8000
   Physical Admin Key ..... 3            Partner System .. 00-00-00-00-00-00
   Port Key ..... 23                     Port Key ..... 0
   Port Priority ..... 32768             Port Priority ..... 0
   Port Number ..... 5012                Port Number ..... 0
   Mode ..... Active                     Mode ..... Passive
   Timeout ..... Long                    Timeout ..... Long
   Individual ..... Yes                  Individual ..... Yes
   Synchronised ..... No                 Synchronised ..... No
   Collecting ..... No                   Collecting ..... No
   Distributing ..... No                 Distributing ..... No
   Defaulted ..... Yes                   Defaulted ..... Yes
   Expired ..... No                      Expired ..... No
```

Working out why a port has not joined a dynamic aggregation

In the following **show etherchannel** output, port**1.0.12** is port-up but has not joined the aggregation (indicated by sync: 0).

```
Aggregator po23 (4623)
  Mac address: 00:15:77:c2:4d:53
  Admin Key: 0023 - Oper Key 0023
  Receive link count: 1 - Transmit link count: 1
  Individual: 0 - Ready: 1
  Partner LAG: 0x8000,00-90-99-3c-de-c0
   Link: port1.0.12 (5012) sync: 0
   Link: port1.0.13 (5013) sync: 1
```

The two most likely reasons for the port to have failed to join are:

- the port has not been attached to the right neighbor.
- the port has been attached to the right neighbor, but the port to which it is connected does not have LACP enabled.

How to diagnose both these issues follows.

Checking the port is attached to the right neighbor

The easiest way to diagnose this situation is to use the command **show etherchannel port <port number>**, and look for the "Partner System" that is reported.

For example, port1.0.12 and port1.0.13 in the following output are supposed to be connected to the same neighbor device. Port1.0.12 reports a partner with MAC address 0000.cd04.f2e0. However, port1.0.13 is reporting a neighbor with MAC address 0090.993c.dec0

.

```
awplus#show port etherchannel port1.0.12
 Link: port1.0.12 (5012)
 Aggregator: po23 (4623)
 Receive machine state: Current
 Periodic Transmission machine state: Fast periodic
 Mux machine state: Attached
 Actor Information:                    Partner Information:
   Selected ................. Selected     Partner Sys Priority ....... 0x8000
   Physical Admin Key ..... 3     Partner System .. 0000.cd04.f2e0
.
.
.
awplus#show port etherchannel port1.0.13
 Link: port1.0.13 (5013)
 Aggregator: po23 (4623)
 Receive machine state: Current
 Periodic Transmission machine state: Fast periodic
 Mux machine state: Collecting/Distributing
 Actor Information:                    Partner Information:
   Selected ................. Selected     Partner Sys Priority ....... 0x8000
   Physical Admin Key ..... 3     Partner System .. 0090.993c.dec0
.
.
.
```

In this case, it is clear that one of the ports is plugged into the wrong neighbor.

Checking the neighbor has LACP enabled

The easiest way to see if a port is not receiving valid LACP packets from its partner is to use the **show etherchannel port <port number>** command. If the Partner System is reported as 00-00-00-00-00-00, as in the next output, then that is a sign that no valid LACPDUs have been received on that port. Having the Partner Port Key, Port Priority and Port Number all reported as "0" are also indicators of not having received valid LACPDUs.

```
awplus#show port etherchannel port1.0.12
 Link: port1.0.12 (5012)
 Receive machine state: Defaulted
 Periodic Transmission machine state: Slow periodic
 Mux machine state: Waiting
 Actor Information:                      Partner Information:
  Selected ................. Standby     Partner Sys Priority ....... 0x8000
  Physical Admin Key ..... 3            Partner System ..  00-00-00-00-00-00
  Port Key ..... 23                      Port Key ..... 0
  Port Priority ..... 32768              Port Priority ..... 0
  Port Number ..... 5012                 Port Number ..... 0
  Mode ..... Active                      Mode ..... Passive
  Timeout ..... Long                     Timeout ..... Long
  Individual ..... Yes                   Individual ..... Yes
  Synchronised ..... No                  Synchronised ..... No
  Collecting ..... No                    Collecting ..... No
  Distributing ..... No                  Distributing ..... No
  Defaulted ..... Yes                    Defaulted ..... Yes
  Expired ..... No                       Expired ..... No
```

You can extend this diagnostic using the **debug lacp event** command to enable event debugging. If a port is receiving LACPDUs, then the event debug will periodically report having received a frame:

```
11:11:16 awplus LACP[1076]: lacp_handle_packet: link port1.0.12 handle
received frame of len 110
```

If the event debugging shows that the port is not receiving packets, then this indicates that the neighbor port does not have LACP enabled, or is failing to communicate. Note that you should capture two minutes or more of event debugging to give yourself the best chance of seeing a receive event if the neighbor is sending LACPDUs.

If the port is receiving LACPDUs, but the port is not showing valid information for the partner port (seen in the **show etherchannel port <command>** output), then the problem could be a subtle matter in the content of the LACPDU. In this case, it is best to capture about two minutes of LACP packets. Use the commands:

```
debug LACP packet
terminal monitor 120
```

Escalate the issue and send this packet debug output as part of the issue report.

LACP debug commands

The command **debug lacp** provides a set of options for debugging LACP activity:

```
debug lacp {all|cli|event|ha|packet|sync|timer}
```

Debugging option	Description
all	Turns on all debugging.
cli	Echoes commands to console.
event	Echoes events to console.
ha	Echoes high availability events to console.
packet	Echoes packet contents to console.
sync	Echoes synchronization to console.
timer	Echoes timer expiry to console.

These debug modes can show you in real time what is happening in the LACP state machine and negotiation processes. For example, if LACP event debugging is enabled, and a link is connected, you will receive output like the following:

```
LACP[1076]: lacp_update_selected: actor link 5012
LACP[1076]: lacp_update_selected: port - partner:0 actor:1
LACP[1076]: lacp_update_selected: port prio - partner:0 actor:32768
LACP[1076]: lacp_update_selected: key - partner:0 actor:32768
LACP[1076]: lacp_update_selected: system prio - partner:32768 actor:32768
LACP[1076]: lacp_update_selected: aggregation flag - partner:1 actor:1
LACP[1076]: lacp_update_selected: partner system 8000-00-00-00-00-00-00
LACP[1076]: lacp_update_selected: actor system 8000-00:00:cd:04:f2: e0
LACP[1076]: lacp_update_selected: link 5012 unselected
LACP[1076]: lacp_update_ntt: link 5012 sets ntt
LACP[1076]: lacp_record_pdu: actor link id:5012
LACP[1076]: lacp_record_pdu: port - actor:5012 partner:5012
LACP[1076]: lacp_record_pdu: port priority - actor:32768 partner:32768
LACP[1076]: lacp_record_pdu: key - actor:32768 partner:32768
LACP[1076]: lacp_record_pdu: system prio - actor:32768 partner:32768
LACP[1076]: lacp_record_pdu: system actor: 8000-00:15:77:c2:4d:53
LACP[1076]: lacp_record_pdu: system partner: 8000-00:15:77:c2:4d:53
LACP[1076]: lacp_record_pdu: Actor flags - 10100001
LACP[1076]: lacp_record_pdu: PDU partner flags - 11101100
LACP[1076]: lacp_record_pdu: PDU actor flags - 11110001
LACP[1076]: lacp_record_pdu: comparison -> pt:1 pr:1 si:1 sp:1 ky:1 ag:1 sn:1
mn:1
LACP[1076]: lacp_record_pdu: link 5012 synchronized
```

In this output, you can see that the LACP module initially updates itself with the information it has about the port that has just come up, then updates itself with the information that it receives in an LACPDU from the partner. Having decided that all is in order, it declares the port to be synchronised, which means that this switch and the neighbor attached to this port are in agreement about each other's LACP information. It does not necessarily mean that the port has been joined to the aggregation.

Chapter 5 | High Performance Loop Protection Using EPSRing

Introduction

Ethernet Protection Switching Rings (EPSRing™) is a loop-protection technology that is designed to provide the type of rapid recovery–within as little as 50 milliseconds–that is traditionally available only in telephony networks.

EPSRing is one of the technologies, along with 10Gigabit Ethernet and WireSpeed QoS, that has made Ethernet suitable for use in high-performance, high-reliability, service-provider networks. These technologies are driving the transition from Ethernet as just on the Enterprise LAN to 'Ethernet Everywhere'.

Like the recovery mechanisms in telephony networks, EPSR requires that the underlying topology be simple. Whereas Spanning Tree Protocol is able to remove loops from any arbitrary multiply connected topology, EPSR can only support a ring topology. This restriction allows the protocol to operate very simply, and it is therefore very reliable and fast-acting.

You must consider the design and configuration of an EPSR network much more carefully than that of an RSTP network, as EPSR introduces a number of rules that constrain the network design. However once a network is

List of terms

EPSR domain
The EPSR entity that protects a given set of VLANs. In each domain there is one control VLAN, and one or more data VLANs. The domain resides on a set of switches connected in a single ring.

Master node
The controlling node for a domain, responsible for polling the ring state, collecting error messages, and controlling the flow of traffic in the domain.

Transit node
The other switches in the domain. They are very limited in the decisions they can make with regard to port forwarding states. They follow instructions sent from the Master Node.

Enhanced recovery
Allows transit nodes to move ports from blocking to forwarding by querying the master node.

established, troubleshooting EPSR is much simpler than troubleshooting RSTP, as the protocol is less complicated and there are fewer specialised protocol interactions.

This chapter explains the operation of the EPSR protocol and the rules that it imposes on the network design and configuration. It details how the protocol detects and recovers from failures in the ring, and explains how to configure an EPSR ring using the AlliedWare Plus operating system. The chapter ends with advice for troubleshooting EPSR problems–while the simplicity of EPSR means that it is rarely a source of network problems, other factors such as packet corruption or misconfiguration can manifest themselves as problems in EPSR.

Overview

Putting a ring of Ethernet switches at the core of a network is a simple way to increase the network's resilience–such a network is no longer susceptible to a single point of failure. However, the ring must be protected from Layer 2 loops. Traditionally, spanning tree protocols are used to protect rings, but they are relatively slow to recover from link failure. This can create problems for applications that have strict loss requirements, such as voice and video traffic, where the speed of recovery is highly significant.

Allied Telesis advanced Layer 3 switches and iMAPs provide a superior solution to the loop protection requirement, known as Ethernet Protection Switching Ring (EPSR). EPSR enables rings to recover rapidly from link or node failures–within as little as 50milliseconds. This is a much faster recovery time than STP at 30 seconds or even RSTP at 1to 3 seconds.

EPSR is designed to operate on a simple ring topology. By keeping the topology simple, EPSR is able to employ a simple, robust algorithm that reacts remarkably quickly to node or link failures. The mechanism itself is simple–when all nodes and links in the ring are up, EPSR prevents a loop by blocking data transmission across one port in the ring. When a node or link fails, EPSR detects the failure rapidly and responds by unblocking the port so that data can flow around the ring.

Understanding EPSR

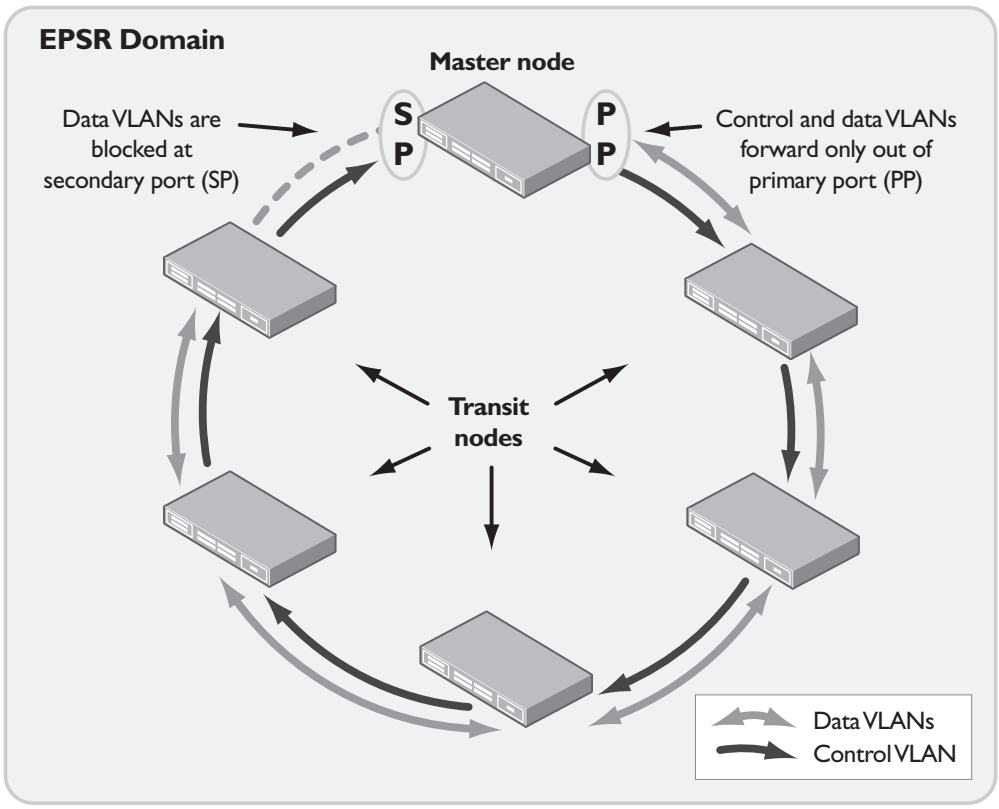
EPSR blocks loops in specified sets of VLANs. All VLANs traversing the ring must be assigned to an EPSR instance, called an **EPSR domain**. As well as normal data VLANs, an EPSR domain has an associated VLAN called the **control VLAN**. This VLAN carries the EPSR control packets for that domain, and nothing else. Each domain has just one control VLAN, but can protect multiple data VLANs.

In each domain, there is a switch given the task of checking for link failures and controlling the traffic flow. This is called the **master node**. When the ring is operating normally, the master node blocks the data VLANs on one of its links to the ring. However, it does not block data from the control VLAN. Instead, the master node uses this VLAN to

check that the ring is still complete by checking that control packets sent out one port are received on the other port. The port that the master transmits the control packets from is called the **primary port**. The port that the master receives its own control packets back through, and also blocks the data VLANs on, is called the **secondary port**.

The other ring members are called **transit nodes**. When an EPSR ring is complete, all a transit node needs to do is forward on the control packets sent by the master node to the next node. However, if a transit node notices a link-down event, it can also generate its own EPSR packet to notify the master node.

The following diagram shows the elements of an EPSR instance, with arrows indicating the flow of traffic.



There is no limit to the number of data VLANs you can assign to a domain (aside from the switch's own maximum VLAN limit). Alternatively, you can have multiple domains in a ring to protect different VLANs–this makes it possible to run totally separate Layer 2 networks, and achieve better load distribution around the ring by configuring different switches to be the master for each ring. Each domain operates independently from the other.

If there are multiple domains on the same physical ring then:

- Each domain requires its own unique control VLAN. No VLAN can be the control VLAN for more than one domain.
- Each domain can have a different switch designated as its master node. There is no requirement that all domains use the same switch as their master node.

- A VLAN can be a data VLAN in more than one domain, so that a switch connected to two separate physical rings (that each have their own domain) can transfer data between the rings. In this scenario, the data VLANs would need to be members of more than one domain on that switch to enable data to pass from one physical ring into the other. However, a data VLAN should not be a member of multiple domains that traverses the same physical ring–doing this could disrupt the network.

When configuring a domain, you must make the ports that are part of the ring (**ring ports**) tagged members of the control VLAN. No other ports on any of the switches should be members of a control VLAN.

As well, the ring ports must be tagged members of each data VLAN associated with a domain. Other ports on the switches can also be members of the data VLANs (otherwise, there would be no way for data to flow into and out of those VLANs). There should be no untagged VLANs on the ring ports.

The following table summarizes the EPSR components.

Component	Description
EPSR domain	The EPSR entity that protects a given set of VLANs. In each domain there is one control VLAN, and one or more data VLANs.
Master node	The controlling node for a domain, responsible for polling the ring state, collecting error messages, and controlling the flow of traffic in the domain.
Transit node	The other nodes in the domain.
Ring port	A port that connects the node to the ring. On the master node, each ring port is either the primary port or the secondary port. On transit nodes, ring ports do not have roles.
Primary port	A ring port on the master node. This port determines the direction of the traffic flow, and is always operational.
Secondary port	A second ring port on the master node. This port remains active, but blocks data transmission in all protected VLANs unless the ring fails. Similar to the blocking port in an STP/RSTP instance.
Control VLAN	The VLAN over which all control messages are sent and received. EPSR never blocks this VLAN.
Data VLAN	A VLAN that the EPSR domain is protecting from loops.

Operation in normal circumstances

When all links in a ring are up, EPSR prevents loops by blocking the data VLANs on the secondary port on the master node. The master node does not need to block any port on the control VLAN because loops never form on the control VLAN. This is because the only data flowing in the control VLAN is EPSR messages, and the master node never forwards any EPSR messages that it receives.

Once you have configured EPSR on the switches, the following process completes the EPSR ring:

1. The master node sends a Health packet.

At regular intervals, known as the Hello Interval, the master node creates an EPSR Health message and sends it out the primary port. By default, the master sends a Health message every second. Each message increments the master node's Transmit: Health counter, shown with the **show epsr count** command.

When the ring is complete, the switch should receive the Health message on its secondary port (having just traversed the ring) within 2 seconds.

2. The transit nodes forward the Health packet through the ring (and to CPU).

The first transit node receives the Health message on one of its two ring ports and, using a hardware filter, sends the message out its other ring port.

Note that transit nodes never generate Health messages, only receive them and forward them with their switching hardware. This does not increment the node's Transmit: Health counter. However, it does increment the Transmit counter in the **show switch port** command on the node.

The hardware filter also copies the Health message to the CPU. This increments the transit node's Receive: Health counter. The CPU processes this message to increment the counter, but does not send the message anywhere because the switching hardware has already done this.

The Health message continues around the rest of the transit nodes, being copied to the CPU of each transit node and simultaneously forwarded in their switching hardware.

3. The master node receives the Health message on its secondary port.

The master node eventually receives the Health message on its secondary port. The master node's hardware filter copies the packet to the CPU (which increments the master node's Receive: Health counter) and does not forward it. Because the master received the Health message on its secondary port, it knows that all links and nodes in the ring are up.

Note that the master node does not send the received Health message out again. If it did, the packet would be continuously flooded around the ring. Also, the master node will only ever send Health messages out its primary port–if this port goes down, it does not send Health messages out of the secondary port.

4. The master node reacts if health packet is not received.

If the master node does not receive the Health message back within 2 seconds, it concludes that the ring must be broken.

Node states

The command **show epsr** displays the node's state for each domain that it is a member of. The possible states are shown below.

Master node states

- **Complete**

  The state when there are no link or node failures on the ring.

- **Failed**

  The state when there is a link or node failure on the ring. This state indicates that the master node received a link down message or that the master node's secondary port failed to receive a Health message.

Transit node states

- **Idle**

  The state when EPSR is first configured, before the master node determines that all links in the ring are up. In this state, both ring ports on the node are blocked for the data VLANs. From this state, the node can move to Links Up or Links Down.

- **Links Up**

  The state when both the node's ring ports are up and forwarding. From this state, the node can move to Links Down.

- **Links Down**

  The state when one or both of the node's ring ports are down. From this state, the node can move to Pre-forwarding.

- **Pre-forwarding**

  The state when both ring ports are up, but one has only just come up and is still blocked to prevent loops. From this state, the transit node can move to Links Up, when the node moves the blocked port to the forwarding state; or to Links Down if another port goes down.

Detecting a ring failure

EPSR uses a fault detection scheme that alerts the ring when a break occurs. The ring then automatically heals itself by sending traffic over a protected reverse path.

EPSR can detect when a transit node or link goes down using the following two methods:

- **Transit node unsolicited fault detection**

  When an EPSR transit node sees one of its ring ports go down, it sends (via the ring port that remains up) a link-down notification directly to the master node. This notifies the master node that the ring is broken and causes it to respond immediately (see "Reacting to a ring failure" on page 81 for a description of the response).

  This explicit notification process is very rapid–it can provide a recovery time of as little as 50 milliseconds.

- **Master node polling fault detection**

  The master node regularly sends Health messages out its primary port to check the condition of the ring. If all links and nodes in the ring are up, the messages arrive back at the master node on its secondary port. If the healthcheck fails to return, then the master node knows that the ring is broken somewhere.

  This acts as a backup mechanism to the link-down notifications from the transit nodes. This detection method is relatively slow compared to the explicit notifications, because it depends on how often the node sends Health messages.

The EPSR Domain figure on page 82 shows the flow of control frames when a there is a break in the ring.

Reacting to a ring failure

EPSR can recover from a failure with the master node, with a transit node, or with a link in the network.

Link or a transit node failure

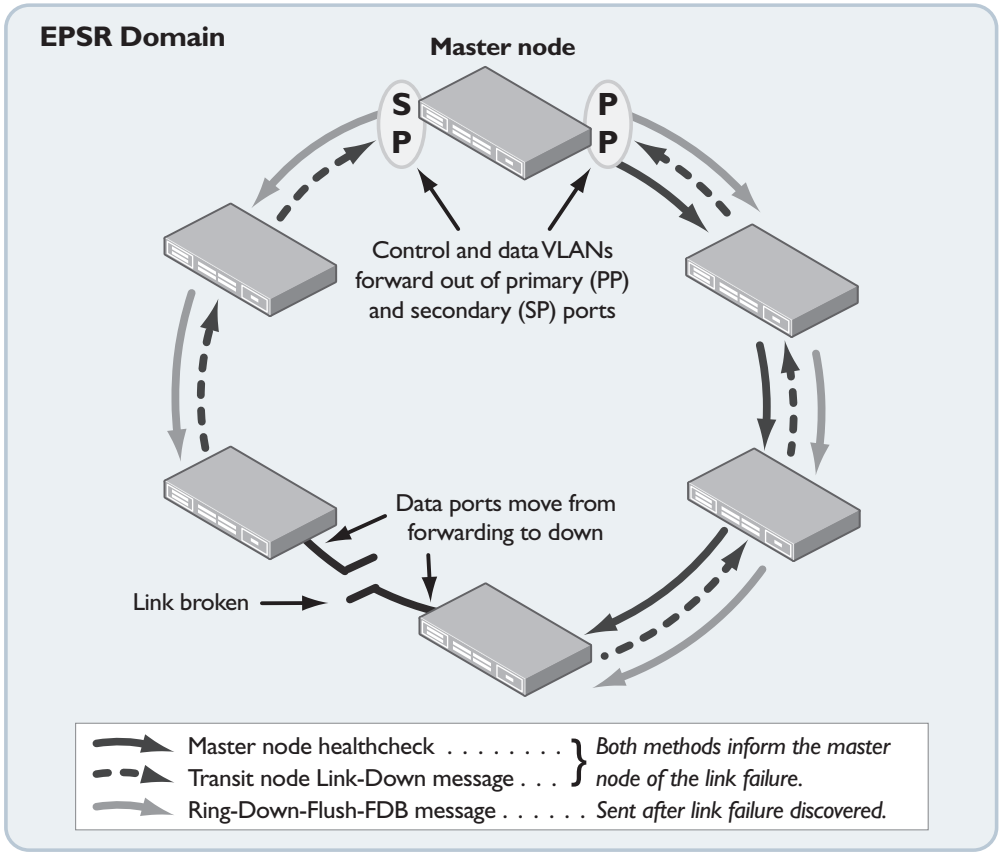
When the master node detects an outage somewhere in the ring, using either detection method, it restores traffic flow by:

1. declaring the ring to be in a Failed state
2. unblocking its secondary port, which enables data VLAN traffic to pass between its primary and secondary ports
3. flushing its own forwarding database (FDB) for the two ring ports
4. sending a message to all the transit nodes, via both its primary and secondary ports

The transit nodes respond to the message by flushing their forwarding databases for each of their ring ports. As the data starts to flow in the ring's new configuration, the nodes (master and transit) re-learn their Layer 2 addresses. During this period, the master node continues to send Health messages over the control VLAN. This situation continues until the faulty link or node is repaired.

For a multi-domain ring, this process occurs separately for each domain within the ring.

The following figure shows the flow of control frames when a there is a break in the ring.



Master node failure

If the master node goes down, the transit nodes simply continue forwarding traffic around the ring–their operation does not change.

The only observable effects on the transit nodes are that:

- they stop receiving Health messages and other messages from the master node.
- the transit nodes that are connected to the master node experience a broken link, and so they send Link-Down messages. If the master node is down, these messages are simply dropped.

Neither of these symptoms affect how the transit nodes forward traffic.

Once the master node recovers, it continues its function as the master node.

Restoring normal operation

Once the failure in the ring is fixed, then the ring must return to its previous state, with the master node blocking its secondary port and all other ports forwarding. The speed at which the restores itself to normal operation depends on whether you have enabled the **enhanced recovery** mechanism on the EPSR domain. This mechanism allows transit nodes to move ports from blocking to forwarding by querying the master node.

Recovery by master node

Once the fault has been fixed, the master node's Health messages traverse the whole ring and arrive at the master node's secondary port. The master node then restores normal conditions by:

1. Declaring the ring to be in a state of Complete.
2. Blocking its secondary port for data VLAN traffic (but not for the control VLAN).
3. Flushing its forwarding database for its two ring ports.
4. Sending a message from its primary port, to all transit nodes.

Recovery by transit nodes with one port down

As soon as the fault has been fixed, the transit nodes on each side of the previously faulty link section detect that link connectivity has returned. They change the state of the newly restored port from down to blocking. The state of the EPSR domain on the transit node moves from Links Down to Pre-Forwarding.

From there, the behaviour of the transit node depends on whether or not it has enhanced recovery enabled:

- when the enhanced recovery mechanism is **disabled**, the transit node simply keeps the newly restored port in the blocking state, and waits for the master node to send a control message. Once the transit nodes receive the message, they flush the forwarding databases for both their ring ports and move the blocking port to the forwarding state, which allows data to flow through the previously blocked ring port.
- when the enhanced recovery mechanism is **enabled**, the transit node sends a request to the master node. If the master node is confident that the transit node can put its port into the forwarding state without causing a loop on the ring, it will send a response to give permission to the transit node to move the port state to forwarding.

In a ring without enhanced recovery enabled, if multiple links in the ring (links not connected to the same transit node) are down, then the ports connected to those links do not return to the forwarding state until the ring is fully restored. So, until you have fixed the final broken link, the ring operates sub-optimally, as the nodes connected to repaired links will have their ports stuck in the blocking state until the ring is complete, and are not able to take advantage of any repaired links.

In both cases (with and without the enhanced recovery mechanism) the transit nodes must wait for some form of notification from the master node before they return the restored ports to the forwarding state. This makes sure the previously-down transit node ports stay blocked until after the master node either blocks its secondary port or uses the healthcheck mechanism to ensure that the ring still has down ports. This cautious approach is necessary to prevent the EPSR domain from ever entering an unprotected state.

Recovery by transit nodes with both ports down

EPSR can also handle double link failures. If both ports on a transit node are down and one port comes up then, even if enhanced recovery is not enabled, the node:

1. Puts the port immediately into the forwarding state and starts forwarding data out that port. It does not need to wait, because the node knows there is no loop in the ring.
2. Remains in the Links Down state.
3. Waits four seconds.
4. If one port is still up and one is still down, the transit node sends a message out the port that is up. This message fakes the message a master node would send when the ring is restored.

Sending this message allows any ports on other transit nodes that are blocking state to move to forwarding traffic in the Links Up state. The timer delay lets the device at the other end of the link that came up configure its port appropriately, so that it is ready to receive the transmitted message.

Note that the master node would not send a message in these circumstances, because the ring is not in a state of Complete. The master node's secondary port remains unblocked.

Transit nodes when the master node is not reachable

The master node is an essential part of the restoration of a port to the forwarding state. However, even if the master node is down, or is disconnected from the ring, the transit nodes can still continue with link restoration processes while the master node is out of contact.

The ring can restore ports to the forwarding state even when the transit nodes cannot contact the master node. The process used is very similar to the double-failure recovery described in the previous section, "Recovery by transit nodes with both ports down" on page 84. When a port on a transit node goes link-up and is put into a blocking state, the transit node starts a timer. The transit node then starts its recovery mechanism–either by waiting for the control message sent when the ring is restored, or sending the Link Forwarding request. If the node does not receive a notification from the master before the timer expires, the transit node concludes that the master node is not reachable, and moves the port to the forwarding state.

Operational capabilities

Ring size

The EPSR protocol does not place any limits on the number of nodes that may be in a ring. The only theoretical limit imposed by the protocol itself is the time that it takes a healthcheck packet to traverse the ring. If the latency within the ring is such that healthcheck packets regularly require more time to traverse the ring than the master's healthcheck timeout, then EPSR will not operate correctly. But, given that the hardware forwarding latency within a switch is very short, it would have to be an incredibly large ring to cause this problem to occur.

The more practical limit is really one of management and monitoring. If you create large rings, then troubleshooting becomes more difficult, and the loss of some links will affect too many users. An average limit that network designers tend to put on their rings is 12 switches, but there are rings of up to 20 switches operating successfully.

Number of domains through a given node

A given node cannot participate in more than 16 EPSR domains.

Number of data VLANs in an EPSR domain

There is no limit on the number of Data VLANs in a single EPSR domain, aside from a switch's own maximum VLAN limit.

Interoperation between implementations

All Allied Telesis equipment that implements EPSR can interoperate in a common ring. This means that you can connect together iMAPs and any EPSR capable Layer 3 switch into a node together. This is regardless of whether the switches run the AlliedWare or AlliedWare Plus operating systems–both are compatible with one another and the iMAPs.

All Allied Telesis equipment that implements EPSR can also interoperate with Extreme Networks equipment running EAPS.

Multiple points of connection between rings

There is a difference in the EPSR implementation on the iMAP and the EPSR implementation in AlliedWare and AlliedWare Plus operating systems.

The iMAPs implement the "common link" feature, which makes it possible to have two separate points of connection between rings, even if the rings share the same set of Data VLANs.

For switches running the AlliedWare and AlliedWare Plus operating systems, it is possible to have two separate points of connection between rings provided the rings have different sets of data VLANs–the switches at the points of connection must use Layer 3 switching between the two rings.

The stackable switches that run the AlliedWare Plus OS can provide a device-resilient connection between rings. By putting a stacked pair of switches at the junction point, and having each ring span the stack (that is, each ring has a ring port on stack member 1 and a ring port on stack member 2) then the connection will be resilient against the loss of either stack member.

EPSR and link aggregation

EPSR supports aggregated links in the ring. This allows you to use aggregated ports as the ring ports on EPSR nodes.

EPSR and spanning tree

An EPSR ring port cannot also participate in a spanning tree. However, any other port on the switch can participate in a spanning tree, and the spanning tree can involve the data VLANs that are protected by an EPSR domain.

EPSR and Q-in-Q VLANs

The data VLANs in an EPSR domain can be VLAN-stacking VLANs. The only restriction is that EPSR ring ports cannot be customer-edge ports of a stacked VLAN; the ring ports must be provider ports of the stacked VLAN.

EPSR and Quality of Service (QoS)

There is no restriction to using QoS with EPSR. However, we recommend that no user data is sent to queue 7. Instead, QoS policies should use queue 6 as the highest queue into which user data is prioritized.

This is because the signalling traffic sent through the control VLAN must have the highest priority. If a domain loses signalling packets, the stability of the ring becomes compromised. Therefore, EPSR signalling messages are automatically sent at the highest priority queue (queue 7) on egress ports. This is to ensure the least possible disruption to EPSR signalling even if the network is congested.

Configuring EPSR using the AlliedWare Plus OS

The following steps describe how to configuring basic EPSR under the AlliedWare Plus operating system. You can find further information about how to configure a variety of different scenarios in the EPSR How To Notes available for both the AlliedWare and AlliedWare Plus™ operating systems. Visit www.alliedtelesis.com/resources/literature/default.aspx to download these documents.

When configuring a domain, you must make the ring ports tagged members of the control VLAN, and the data VLANs associated with the control VLAN. There should be no untagged VLANs on the ring ports. Once you have the ports, VLANs, and other basic properties configured, follow these steps on each node of the domain:

1. Enter the EPSR configuration mode.

Use the command:

```
awplus(config)#epsr configuration
```

2. Create the EPSR domain.

When you create an EPSR domain, you must use different commands to identify the master node and the transit nodes.

When creating the domain on a master node, use the **epsr mode master** command. In this command you must also specify the primary port and the control VLAN:

```
awplus(config-epsr)#epsr <name> mode master controlvlan
   <control-vid> primaryport <port-numbers>
```

(You do not need to specify the secondary port; it is inferred from the configuration.)

On a transit node, use the **epsr mode transit** command to create the domain. You will need to specify the control VLAN:

```
awplus(config-epsr)#epsr <name> mode transit controlvlan
   <control vid>
```

3. Add the data VLANs to the nodes in the EPSR domain.

Specify the data VLANs on both the master node and the transit nodes. Use the command:

```
awplus(config-epsr)#epsr <name> datavlan <data-vid>
```

4. Enable the enhanced recovery mechanism on the domain.

We highly recommended that you enable the enhanced recovery mechanism on the domain. Use the command:

```
awplus(config-epsr)#epsr <name> enhance enable
```

5. Enable the domain.

Use the command:

```
awplus(config-epsr)#epsr <name> state enabled
```

SNMP traps for EPSR

The SNMP MIB for EPSR defines a trap that is sent whenever there is a state change in any ESPR domain on a switch (regardless of whether it is a master node or a transit node).

The information sent in the trap is:
- Node type (Master/Transit)
- EPSR domain name (the name that the user assigns to the domain
- EPSR domain ID (unique number the software automatically assigns to each domain)
- Previous state (Idle/Complete/Failed/LinkUp/LinkDown/PreForward
- Next state (Idle/Complete/Failed/LinkUp/LinkDown/PreForward
- Control VLAN ID
- Primary port interface identifier
- Primary port state (Blocked/Forward)
- Secondary port interface identifier
- Secondary port state (Blocked/Forward)

To configure a switch to send SNMP traps, use the command:

```
snmp-server enable traps epsr
```

Monitoring and Troubleshooting EPSR

Port forwarding problems in data VLANs

The most likely problems to occur in an EPSR ring are:

- nodes **not** forwarding on a port when they should
- nodes forwarding on ports **when they should not**

The recommended steps to investigate either of these issues are as follows:

▶ Find out the states of EPSR domains.

Find out the states of the EPSR domains on the switches in the ring. To do this, use the **show epsr** command.

Here is some example output:

```
EPSR Information
------------------------------------------------------------------------
 Name ....................... r1
 Mode ........................ Master
 Status ...................... Enabled
 State ....................... Failed
 Control Vlan ................ 10
 Data VLAN(s) ................ 20
 Interface Mode .............. Ports Only
 Primary Port ................ port1.0.1
 Primary Port Status ........... Forwarding
 Secondary Port .............. port1.0.2
 Secondary Port Status ......... Down
 Hello Time .................. 1 s
 Failover Time ............... 2 s
 Ring Flap Time .............. 0 s
 Trap ........................ Enabled
 Enhanced Recovery ............ Enabled

 Name ....................... r2
 Mode ........................ Master
 Status ...................... Enabled
 State ....................... Failed
 Control Vlan ................ 110
 Data VLAN(s) ................ 120
 Interface Mode .............. Ports Only
 Primary Port ................ port1.0.1
 Primary Port Status ........... Forwarding
 Secondary Port .............. port1.0.2
 Secondary Port Status ......... Down
 Hello Time .................. 1 s
 Failover Time ............... 2 s
 Ring Flap Time .............. 0 s
 Trap ........................ Enabled
 Enhanced Recovery ............ Enabled
------------------------------------------------------------------------
```

▶ Check the control packets received.

If one or more nodes are not in the state that you expect them to be in, look for reasons why this might be. The first thing to do is to check that the nodes are seeing the EPSR control packets that they should be seeing, and check whether they are reporting significant numbers of invalid packets.

This information is gained by looking at the EPSR counters, as shown in the next figure. Use the **show EPSR counters** command to display these counters.

```
 EPSR Counters
---------------------------------------------------------------------
 Name: r1
  Receive:                                Transmit:
  Total EPSR Packets          345227      Total EPSR Packets      640611
  Health                      332435      Health                  628664
  Ring Up                       2868       Ring Up                 2869
  Ring Down                        0       Ring Down               5752
  Link Down                     3056       Link Down               0
  Link Forward Request          2796       Link Forward Request    0
  Permit Link Foward             885       Permit Link Forward     3326
  Invalid EPSR Packets          3187

 Name: r2
  Receive:                                Transmit:
  Total EPSR Packets          276762      Total EPSR Packets      445689
  Health                      266569      Health                  445109
  Ring Up                          8       Ring Up                 8
  Ring Down                        0       Ring Down               23
  Link Down                       79       Link Down               0
  Link Forward Request           285       Link Forward Request    0
  Permit Link Foward              14       Permit Link Forward     549
  Invalid EPSR Packets          9807
---------------------------------------------------------------------
```

Check that:

- the master node is reporting regular transmission of a health packet on each ESPR domain.
- the transit nodes are receiving the health packets at the same rate as the master is sending them.
- when transit nodes send Link Forward Requests or Link Down notifications, they are received at the master.
- when the master sends Ring Up or Ring Down notifications, they are received at the transit nodes.
- the invalid EPSR packet counters are not incrementing.

EPSR info debugging will also show whether messages are being received at the master or the transit node. Use the command **debug epsr info** to enable EPSR info debugging, followed by the **terminal monitor** command to get the debugging displayed on screen.

If you enable this debugging on the master node, then an event like a ring port going down on another switch should result in the master node reporting that it received link-down notifications, as shown in the following figure.

```
EPSR[1342]: EPSR r1: link down msg from 00-00-cd-25-33-fc
EPSR[1342]: EPSR r2: link down msg from 00-00-cd-25-33-fc
```

▶ Look at the control messages.

If it looks like the nodes are sending or receiving EPSR messages correctly, then the next thing to do is to look at the content of the messages.

You can use the command **debug epsr pkt** to see the content of the EPSR control messages.

```
EPSR[1342]: port1.0.1 Tx:
EPSR[1342]:  00e02b00 00040000 cd28bfe5 8100e06e 005caaaa 0300e02b
EPSR[1342]:  00bb0100 00540a94 00000000 0000cd28 bfe5990b 00400105
EPSR[1342]:  006e0000 00000000 cd28bfe5 00010002 0200a434 00000000
EPSR[1342]: port1.0.1 Tx:
EPSR[1342]:  00e02b00 00040000 cd28bfe5 8100e00a 005caaaa 0300e02b
EPSR[1342]:  00bb0100 0054177e 00000000 0000cd28 bfe5990b 00400105
EPSR[1342]:  000a0000 00000000 cd28bfe5 00010002 020097ae 00000000
```

To see a decoded output of the content of each EPSR control packet, enable message debugging by using the **debug epsr msg** command.

```
EPSR[1342]: port1.0.1 Tx:
EPSR[1342]: ------------------------------------------------------------
EPSR[1342]:           TYPE = HEALTH                     STATE = FAILED
EPSR[1342]:      CTRL VLAN = 110                       SYSTEM = 00-00-
cd-28-bf-e5
EPSR[1342]:     HELLO TIME = 1                      FAIL TIME = 2
EPSR[1342]:      HELLO SEQ = 42226
EPSR[1342]: ------------------------------------------------------------
EPSR[1342]: port1.0.1 Tx:
EPSR[1342]:  ------------------------------------------------------------
-----------
EPSR[1342]:           TYPE = HEALTH                     STATE = FAILED
EPSR[1342]:      CTRL VLAN = 10                        SYSTEM = 00-00-
cd-28-bf-e5
EPSR[1342]:     HELLO TIME = 1                      FAIL TIME = 2
EPSR[1342]:      HELLO SEQ = 39020
EPSR[1342]: ------------------------------------------------------------
```

While looking at the contents of the control packets might not let you solve a problem onsite, capturing some sequences of packets received and sent from ring nodes could be useful information to provide when escalating an issue. In particular, if a fault in the EPSR ring operation seems to be related to a particular event (for example, a port going up or down, or a particular user service starting up), then you should capture a sequence of EPSR packet, message, and info debugging from affected nodes at the time of the event. You can enable these debugging options together–we recommend doing this, as the three debug options will clearly show the events on the network. Try to capture debug output on the master and transit nodes in the ring simultaneously, so that the outputs can be correlated.

The following example output is from a master node with packet, message, and info debugging enabled, and was captured when a port in the ring went link-down. In the output the master node reports receiving a Link-Down message, sending a Ring-Down-Flush-FDB message in response, and changing the ring state from complete to failed.

```
EPSR[1296]: port1.0.2 Rx:
EPSR[1296]:  00e02b00 00040000 cd127808 810003e8 005caaaa 0300e02b
EPSR[1296]:  00bb0100 00543935 00000000 0000cd12 7808990b 00400108
EPSR[1296]:  03e80000 00000000 cd127808 00000000 04000000 00000000
EPSR[1296]: port1.0.2 Rx:
EPSR[1296]: ------------------------------------------------------------
EPSR[1296]:           TYPE = LINK-DOWN                STATE = LINK-DOWN
EPSR[1296]:      CTRL VLAN = 1000                       SYSTEM = 00-00-
cd-12-78-08
EPSR[1296]:     HELLO TIME = 0                       FAIL TIME = 0
EPSR[1296]:      HELLO SEQ = 0
EPSR[1296]: ------------------------------------------------------------
EPSR[1296]: EPSR awplus: link down msg from 00-00-cd-12-78-08
EPSR[1296]: port1.0.2 Tx:
EPSR[1296]:  00e02b00 00040000 cd240331 8100e3e8 005caaaa 0300e02b
EPSR[1296]:  00bb0100 005424c1 00000000 0000cd24 0331990b 00400107
EPSR[1296]:  03e80000 00000000 cd240331 00000000 02000000 00000000
EPSR[1296]: port1.0.2 Tx:
EPSR[1296]: ------------------------------------------------------------
EPSR[1296]:           TYPE = RING-DOWN-FLUSH-FDB       STATE = FAILED
EPSR[1296]:      CTRL VLAN = 1000                      SYSTEM = 00-00-
cd-24-03-31
EPSR[1296]:     HELLO TIME = 0                       FAIL TIME = 0
EPSR[1296]:      HELLO SEQ = 0
EPSR[1296]: ------------------------------------------------------------
EPSR[1296]: Unblock EPSR:awplus port:5002 VLAN:2
EPSR[1296]: EPSR awplus: port 5002 is forwarding
EPSR[1296]: Flush FDB EPSR: awplus  vid: 2
EPSR[1296]: awplus oldState:COMPLETE newState:FAILED
EPSR[1296]: EPSR awplus: ring failed
```

The next example output is from a transit node during the same network event. It reports the receipt of the Ring-Down-Flush-FDB message, and shows the content of the message. This should correlate with the content sent from the master.

```
EPSR[1284]: port1.0.1 Rx:
EPSR[1284]:  00e02b00 00040000 cd240331 810003e8 0058aaaa 0300e02b
EPSR[1284]:  00bb0100 005426d7 00000000 0000cd24 0331990b 00400107
EPSR[1284]:  03e80000 00000000 cd240331 00000000 02000000 00000000
EPSR[1284]: port1.0.1 Rx:
EPSR[1284]: ------------------------------------------------------------
EPSR[1284]:           TYPE = RING-DOWN-FLUSH-FDB       STATE = FAILED
EPSR[1284]:      CTRL VLAN = 1000                      SYSTEM = 00-00-
cd-24-03-31
EPSR[1284]:     HELLO TIME = 0                       FAIL TIME = 0
EPSR[1284]:      HELLO SEQ = 0
EPSR[1284]: ------------------------------------------------------------
```

Flooding problems in the control VLAN

The control VLAN is not blocked on any port. This means that if non-EPSR packets leak into the control VLAN, these packets will loop endlessly around the ring. If this occurs, it will probably result in the network slowing down, as the endless looping of these packets could cause congestion.

The most likely causes of non-EPSR packets entering the control VLAN would be either a misconfiguration or incorrect cabling. Either of these could allow data from an outside source to enter a port configured in an EPSR control VLAN.

The most effective way to determine if non-EPSR data is traversing the control VLAN is to mirror a ring port on an EPSR node, and sniff the data flowing through that port. Then look at the captured data for non-EPSR data tagged with the VID of the control VLAN.

Data looping on an unprotected VLAN

It is vital that there are no VLANs on the ring ports that span the ring but are not part of an EPSR domain. These VLANs would be unprotected and so would loop data, creating congestion on the network.

In particular, it is important to check that there is no native VLAN configured on a ring port. Because it is untagged, a native VLAN cannot be protected by EPSR.

For more advice on troubleshooting EPSR, refer to "Chapter 10 | Troubleshooting" on page 189

Chapter 6 | The Operation of Ethernet Switches

Introduction

The aim of this chapter is to give you an understanding of the general logic of the processing of packets within an Ethernet switch.

The chapter begins by explaining the essential differences between a software-based networking device, like a router, and a hardware-based switch. Then it moves on to show you that the switching of a packet within an Ethernet switch is essentially quite simple. The decisions made in the processing of a packet are represented in a logic diagram, and the steps in that diagram are explained.

The last section of the chapter is a step-by-step consideration of the processes involved in Layer 3 forwarding an ICMP ping between two hosts that were hitherto unknown to the switch.

List of terms

Ingress

Incoming packet process.

Egress

Outgoing packet process.

ASIC

Dedicated switching chip.

Dynamic tables

Sets of registers within the ASIC. The contents of the tables are controlled by the switch software, in response to configuration and to information learnt from network control protocols.

The tables are used for making decisions about forwarding or dropping of data packets.

Overview

To be effective in network monitoring and troubleshooting, it is useful to have a good general understanding of the internal structure of Ethernet switches, and the logic of their data forwarding process. Just before looking at the details of a Layer 3 switch, it is useful to first take a quick look at a software-based router.

Routers - Layer 3 forwarding in software

A router is basically a specialised PC. It has a CPU, some memory, some file storage and devices connected via a high-speed internal communication Bus, typically a PCI Bus. It will have several network interfaces attached, for example:

- Ethernet ports
- ADSL ports
- Serial ports

It may also have a basic Layer 2 switch chip and some switch ports.

When a router receives packets:

- they are sent to the CPU via the internal communication bus.
- the CPU decides what to do with these packets based on a series of dynamic tables in software.
- if a packet is to be forwarded (not filtered), it is then sent to the appropriate interface (via the internal communication bus) and sent out the chosen egress port.

A router's Ethernet ports do **not** have associated MAC tables, as the purpose of MAC tables is to reduce flooding on devices that have multiple ports connecting to the same LAN. A router's Ethernet port is a single point of connection onto a LAN, so there is no need to keep a table of which MACs on the LAN are associated with this port - they all are. Of course, if the router also contains a switch chip, then the switch **ports** will have associated MAC tables.

Switch Overview

A switch is basically a more specialised PC:

- it has a CPU, some memory, some file storage connected via a high-speed Bus, AND a switch chip.
- it will have some number of switch ports, the number being dictated by how many ports are on the switch chip.
- it may have an Ethernet port for management which is connected to the CPU via the high-speed bus.

  Switch chips are also known as ASIC's, packet processors, chips, silicon.

A switch forwards packets:

- packets are received and switched by the switch chip, which is directly connected to the ports.
- the switch chip decides what to do with these packets based on a series of dynamic tables in the chip.
- some packets (like broadcasts, and those addressed to the switch itself) are sent up to the CPU, but the vast majority of packets are dealt with inside the switch chip.

Switch Function

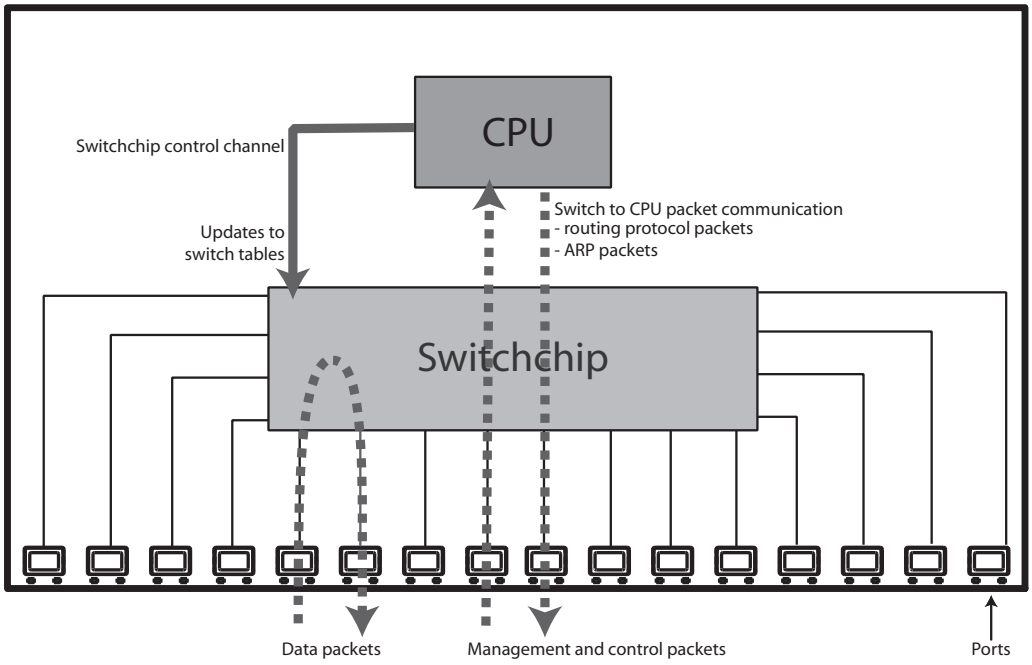
Whilst there is a reasonable amount of complexity in a switch, it is important to bear in mind that on reception of a packet, a switch will take one of only four actions on that packet:

1. Drop the packet.
2. Forward the packet out one port or to the CPU (unicast forwarding).
3. Forward the packet out multiple ports (multicast forwarding).
4. Flood the packet out all ports - this refers to all ports in a VLAN including the CPU (broadcast forwarding).

The decision as to which action to take is made by the switch chip.

Relationship between switch chip and CPU

The CPU, based on current protocol states, will configure the dynamic tables in the switch chip that control these operations.



Packet processing walk-through

To understand the decision points in a generic switch's packet processing, let's step through that processing.

The decision processes can be broken down into three task blocks:

1. **Ingress** process

   This will basically decide if the switch will accept the packet.

   If so, the packet's VLAN membership is then decided.

2. **Forwarding** process

   Having accepted the packet, where will it be sent?

   This is the most complex part of the processing, particularly on a Layer 3 switch. A number of decisions have to be made about the nature of the packet, in order to determine whether it will be forwarded, and what category it belongs to.

   In a Layer 3 switch, that is only Layer 3 switching IPv4 (and not IPv6, IPX, or other network-layer protocols), there are essentially five categories that packets will be assigned to:

   - Layer 2 broadcast
   - Layer 2 unicast
   - Layer 2 multicast
   - Layer 3 unicast
   - Layer 3 multicast

   The forwarding process for Layer 2 **broadcast** packets is very simple - just send it to all ports of the VLAN. But, for the other four categories, the following decisions are controlled by large lookup tables, that are maintained by the CPU, as will be explained in detail in the table management chapters "Chapter 7 | Management of Forwarding Tables - Layer 2" on page 107, and "Chapter 8 | Management of Forwarding Tables - Layer 3" on page 129.

3. **Egress** process

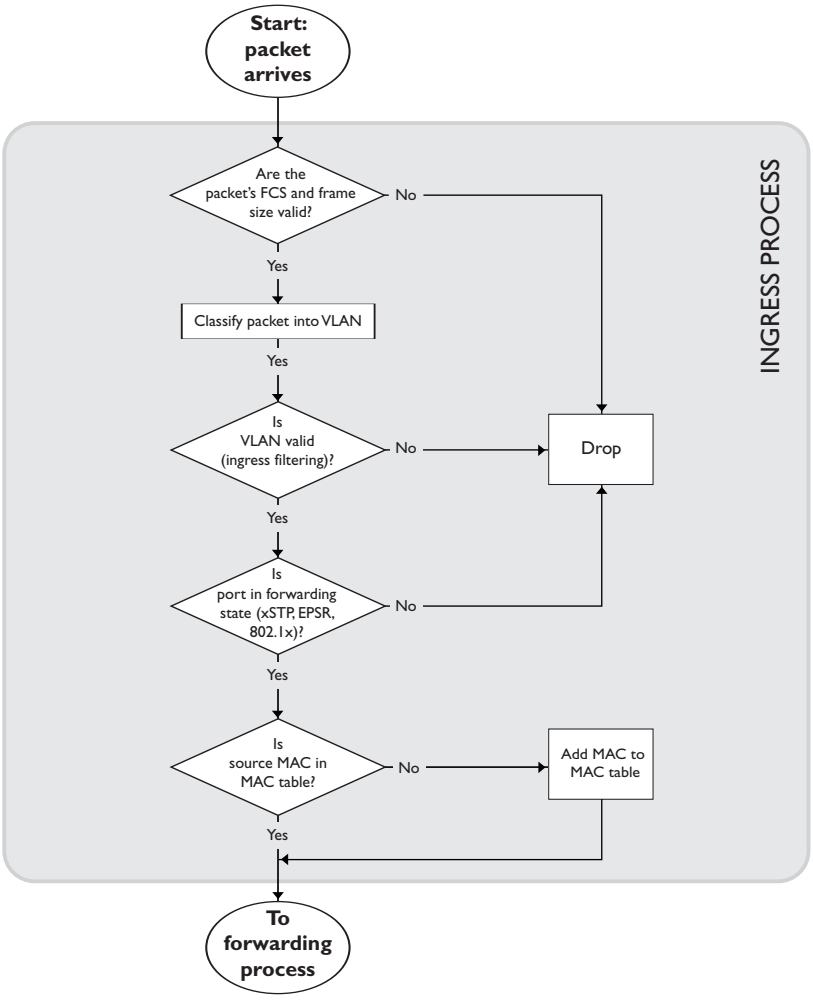
   Apply any filters or Quality of Service actions.

All of these tasks are performed by the switch chip.

Ingress Process

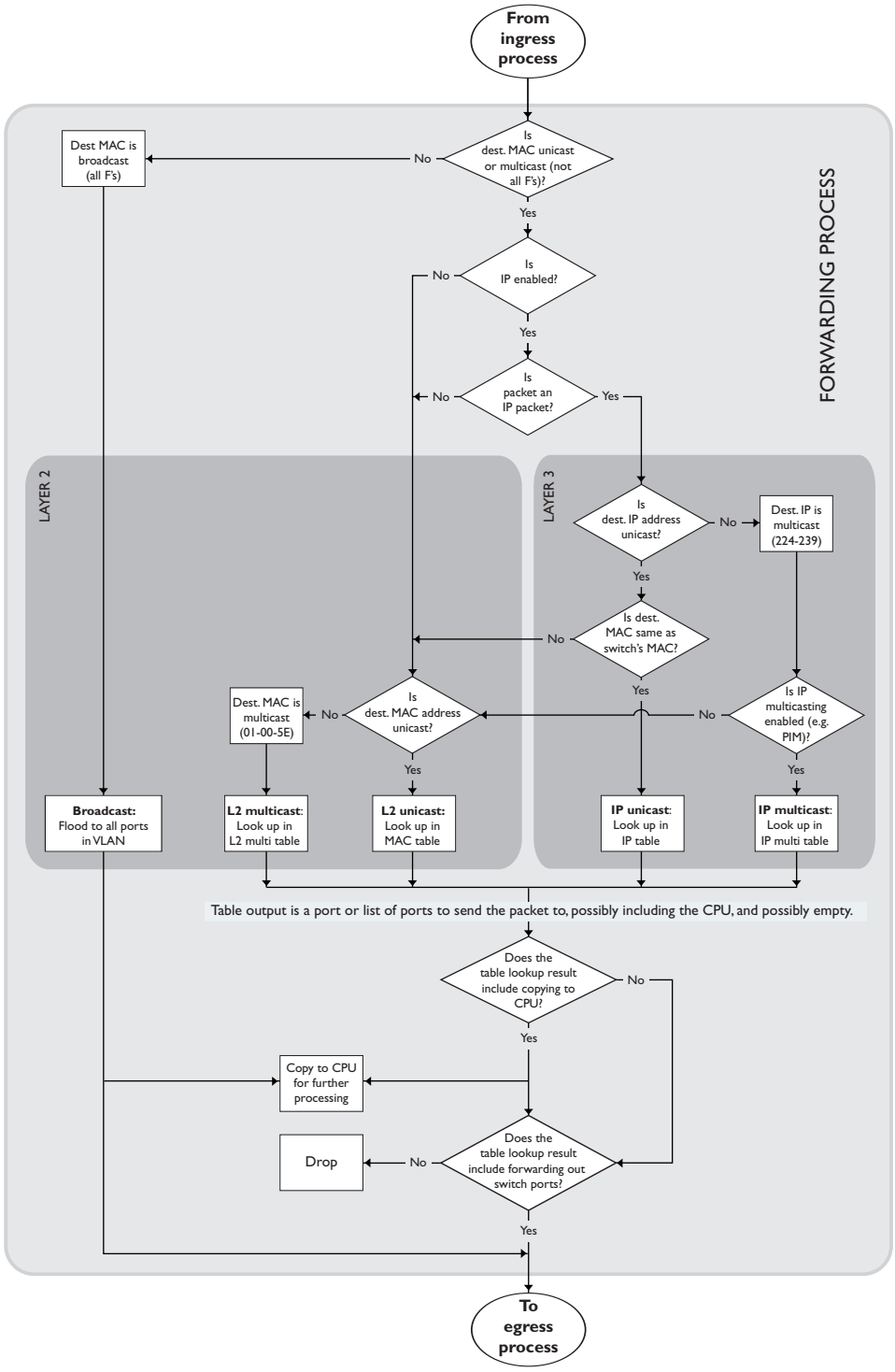
The steps in the Ingress process are:

1. Assuming the port is linked up, a packet is received.
2. The frame format is checked to see if it is valid (including a verification of the FCS)
3. The packet is then classified into a VLAN - all packets must belong to one and only one VLAN. The logic of the VLAN classification process is described in "Implementing VLANs" on page 8
4. The logical port state is checked - do any loop protection or security processes on this port make a ruling that it cannot accept this packet?
5. The source MAC address is added to the MAC table, or if there is already an entry for this MAC address, the ageing field on this entry is updated.



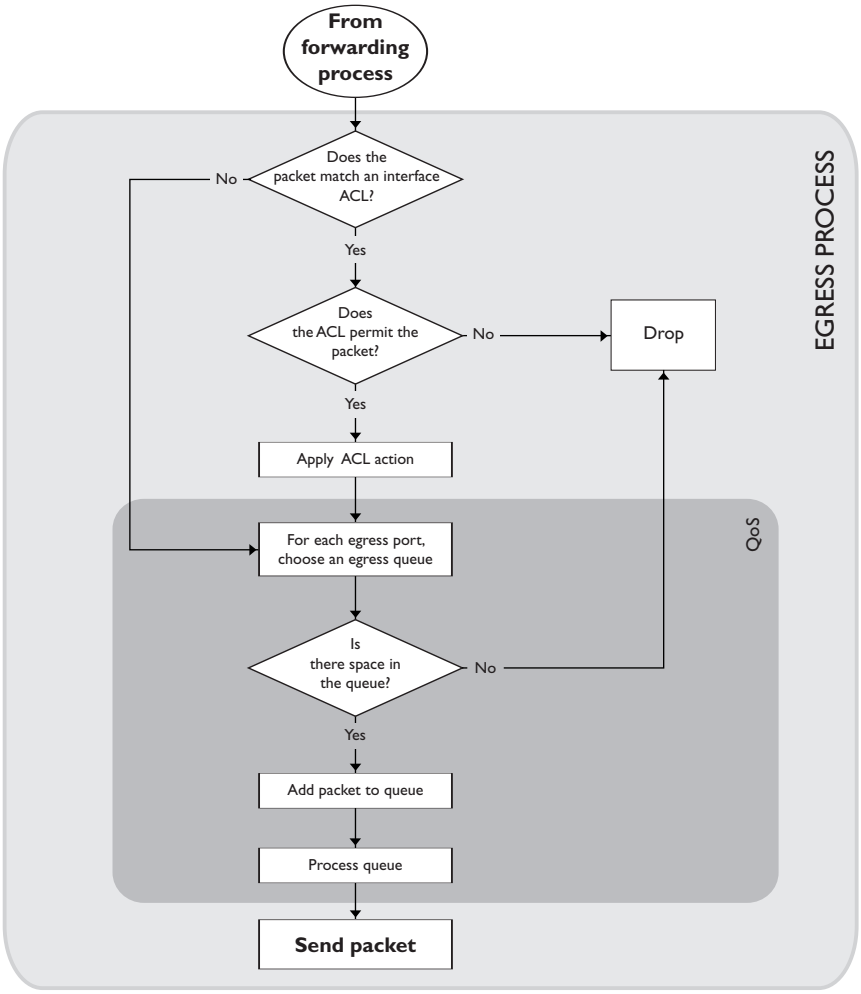
Forwarding Process

The forwarding process decides which port(s), if any, the packet is to be sent to. It does this by looking up the destination MAC or IP address in one of the switching tables (the actual tables can vary between switches). The table entries will specify if the packet is to be dropped, forwarded, or sent/copied to the CPU. If the packet is to be forwarded, it will then be sent to the egress process.



Egress Process

This is a very simplistic view of the egress process. Packets can be dropped due to matching a hardware filter (ACL). They can also be dropped due to congestion, i.e. when there is no space in the egress queue. Once a packet is put into a queue it will be sent, but there could be some delay if there is congestion.



Exceptions or special cases

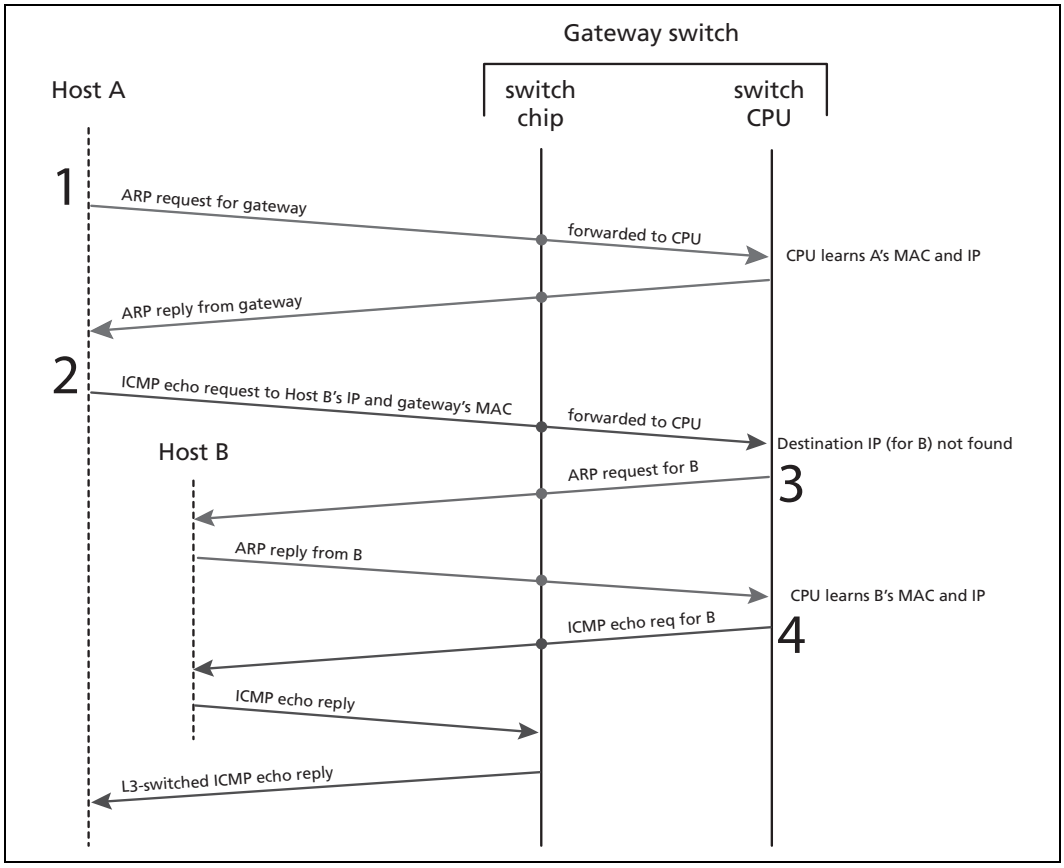
- **IEEE defined BPDUs** - these include xSTP, LACP, 802.1x, LLDP, GARP and Pause frames - i.e. control packets with a dest MAC address starting with 0180.C2.
- These packets are defined from switch-to-switch conversations, and usually are not forwarded, unless the switch is configured as a pass-through for one or other control protocol. Typically, these packets are sent up to the CPU to be processed. If the associated protocol is not operating on the switch, then the CPU will generally drop the frames.
- **Mirroring** - packets can be copied to a specified port.
- **Private VLANs** - flooding rules are changed.
- **Learn limiting** - restricts the number of MAC addresses that can be learnt and forwarded.
- **Rapid MAC movement and loop detection** - these detect an abnormal event on the network and disable packet reception for a period of time.
- **Storm Control** - applies a rate limit to Broadcast and/or Multicast and/or DLFs (Destination Lookup Failures) packets. Those packets that are in excess of the configured rate are discarded.
- **Directed IP broadcasts** - a directed broadcast is a packet that arrives on one IP interface, and its destination IP address is the broadcast IP address for the subnet connected to another IP interface of the switch. By default, such a packet will be dropped. But if broadcast forwarding has been enabled on the egress interface, with the command **ip directed-broadcast**, then the packet will be sent as a broadcast (with destination MAC address FFFF.FFFF.FFFF) out all ports of the egress VLAN.
- **TTL expiry** - all IP packets carry a time-to-live (TTL) field in their headers. The originator of the packet will insert a value into this field, then every Layer 3 forwarding device that the packet passes through will decrement the value by one. If a Layer 3 device finds itself decrementing the TTL of a packet to zero, then it will discard the packet and send an ICMP time-expired message back to the originator of the packet.

  In a Layer 3 switch, the decrementing of the TTL is performed in hardware as part of the Layer 3 forwarding process. When a packet arrives with TTL=1, it is sent to the CPU instead of being forwarded. The CPU, now aware of the timed-out packet, will generate an ICMP time-expired message to send back to the originator.

The end-to-end process of IP forwarding

Having seen the decision processes occurring within the switch chip, let's move the level of abstraction up a bit, and take an overview of the steps involved in establishing an IP flow between two hosts through a Layer 3 switch.

Consider the case that a host on one subnet pings a host on another subnet. A Layer 3 switch is acting as a router between the two subnets. The Layer 3 switch will be the gateway device for both hosts, and the route packets between the hosts.
The following figure shows the packet flow when the one host pings through the switch to the other host for the **first** time.



You can break this packet flow down into four stages:

1. Host A initiates the ping to Host B. Host A recognises that Host B is in a different subnet, so it needs to send the ICMP echo request via its gateway router (the Layer 3 switch). Therefore the destination MAC address of the echo requests must be the MAC address of the gateway. If Host A does not know the MAC address of the gateway (does not have an ARP-cache entry that associates a MAC address with its known gateway IP address), then it sends an ARP request to ask for that MAC address. The switch receives the ARP request from host A, the switch chip forwards the ARP request to the CPU, which sends an ARP reply that tells host A the switch's MAC address. In the process, the switch also learns the MAC and IP address of Host A. This enables the switch to create a MAC-table entry for Host A's MAC address, and a forwarding entry for Host A's IP address in its Layer 3 forwarding table.

In fact, the switch would create entries in at least three tables:

a. **hardware MAC**
in AW+ this is displayed by: **show platform table macaddr**

b. **software ARP**
in AW+ this is displayed by: **show arp**. This gets updated when the CPU receives the ARP request from the switch chip.

c. **hardware IP**
in AW+ this is displayed by: **show platform table IP** . This gets updated because the CPU propagates the software ARP table entry to the hardware table.

2. Now that host A knows the MAC address of the gateway, it sends the ICMP echo request packet. The switch chip receives this, and checks its hardware IP table for the destination address. It does not find the address in the table, and therefore sends the echo request to the CPU. The CPU looks up the best route to the destination address in the IP route table, and determines the egress VLAN. It doesn't know the MAC address of Host B, or which port it is connected to. So it sends an ARP request for Host B. The switch chip floods that ARP request out all ports of the egress VLAN.
3. Host B sends an ARP reply and its details get added to the hardware MAC table, the software ARP table, and the hardware IP table. The switch now has the information it needs in order to send the echo request to Host B, so it does.
4. Host B replies to the echo request. The switch chip forwards the reply directly to Host A. At this point, the switch is Layer 3 switching, because its hardware entries for this particular IP flow are now complete.

Multiple switch chips

The above example considers the case that the switch has only one switch chip. But, it is not uncommon for switches to contain multiple switch chips - each chip being connected to a certain set of external ports. In this section, we will follow the convention of AlliedWare Plus, and refer to these multiple switch chips as switch instances.

So, let us now consider the case that the switch contains multiple switch instances and the two hosts are connected to different instances. Packets with unrecognised addresses are still sent to the CPU and the CPU floods ARP requests out all ports in the relevant VLAN no matter which chip the ports are on.

The significant difference is that a multi-instance switch has multiple hardware tables, which it keeps in sync. When one switch instance learns a new MAC address, it updates its hardware MAC table.

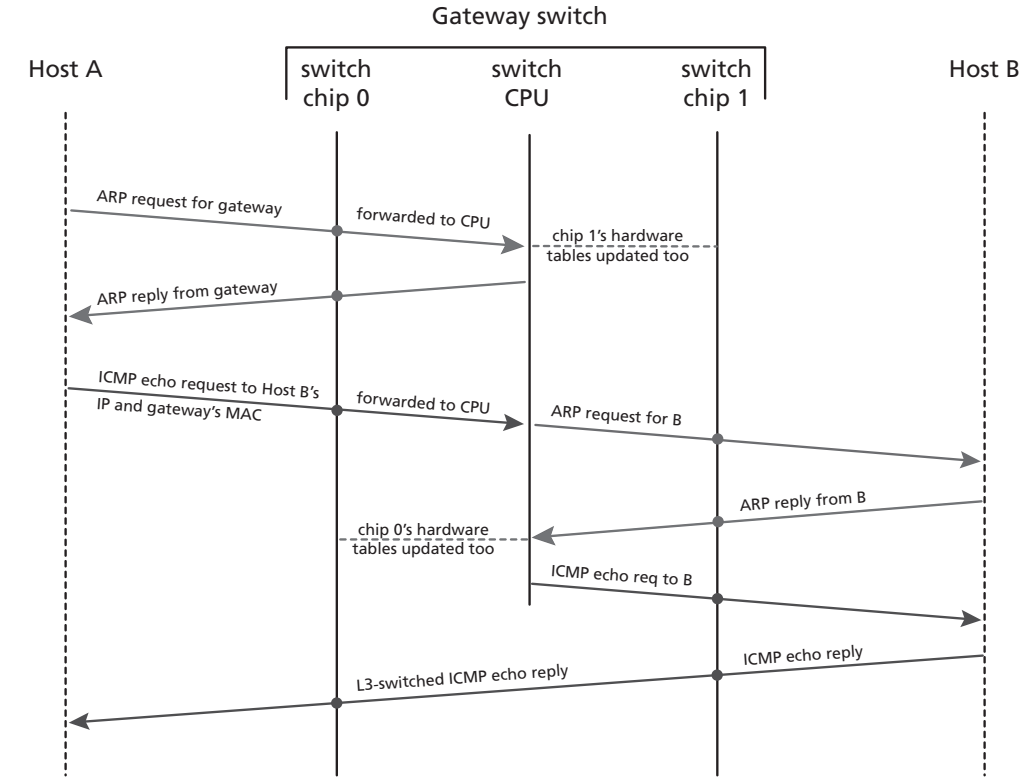
Display with: **show platform table macaddr instance 1.0**, for example.

The switch then updates the hardware MAC table on the other instance(s)

Display with: **show platform table macaddr instance 1.1**, for example.

Similarly, when the switch updates the hardware Layer 3 information, based on the content of ARP packets, it updates the Layer 3 tables of multiple instances, not just a single instance.

The following figure shows the packet flow when host A and host B are attached to ports in two different switch instances (and for simplicity, it assumes that all ports in Host B's VLAN are attached to instance 1). The tables to view at each processing stage are the same as if the ports are in the same instance (see "The end-to-end process of IP forwarding" on page 104). The main difference is that whenever the switch updates one switch chip's hardware tables, it also updates the other chip's tables.



Switch containing multiple switch chips

Chapter 7 | Management of Forwarding Tables - Layer 2

Introduction

When a packet arrives at a switch port, the switching chip makes various decisions about the nature of the packet, and then eventually chooses which forwarding table to use to decide the egress port(s) for that packet. The flowchart used in the section "Forwarding Process" on page 101, shows this decision-making process.

The actual choice of the egress port(s) to which packets are sent is determined by the contents of the forwarding tables. It is very important that the contents of these tables be correct. Very often, network problems are caused by incorrect or missing information in these tables. Therefore, effective network troubleshooting requires a good understanding of the mechanisms by which the switch manages the contents of these tables.

In this chapter, we will look at the tables used in Layer 2 unicast, and Layer 2 multicast. We will look at Layer 3 switching in the next chapter.

Note, throughout this chapter, there are references to commands for displaying hardware tables, and illustrative snippets of hardware table output. These commands and output examples are all taken from the x900 and x908 switches. On other models of switches, like the x600, the commands and the format of the table output may differ slightly. However, the concepts described in this chapter are universal to Allied Telesis x-series switches.

List of terms

HW tables

Hardware tables contain Layer 2 or Layer 3 packet forwarding information. They are the hardware (silicon) cache of the forwarding information previously learnt in software i.e. the CPU.

MAC address learning

A key optimisation in Ethernet switching is that the flooding of unicast traffic is minimized. This is based on switches knowing which port to forward traffic to for given destination MAC addresses. Switches achieve this by the simple process of noting on which ports packets arrive from given MAC addresses, as those will be the ports to which return packets to those MAC addresses will need to be forwarded. This process is referred to as MAC address learning.

The Layer 2 unicast table

The purpose of the Layer 2 unicast table

This is the table that is frequently referred to as the FDB (Forwarding DataBase).

Let us start by thinking about what the purpose of this table is. This is the table that is used to decide which egress port to Layer 2 switch a unicast packet to. But why is it important to choose a particular port to switch the packet to? Why not just send it to all ports? If we did that, then it would be sure to get to its destination one way or another. So why have all this elaborate process in the switching chip simply to narrow the choice down to a single egress port?

The answer is, of course, **performance**. The networking industry has moved from using hubs, which did simply spray **every** packet to **every** port, to switches. Why go to all this effort to work out which individual port to send a packet to? Because the "spray and pray" method of the hub led to terrible network congestion, and therefore low network performance.

More recently, we have become aware that there are good security reasons for keeping strict control over where packets are sent, but the original driving force behind the migration from hubs to switches was simply performance.

So, given that performance is the key reason for the concept of an FDB even to exist, then all the time that we are considering the operation of the FDB, we should remember that it has to operate **very fast**. So, any activity related to the operation of the FDB needs to be efficient.

Unicast table entries

Let us look at a typical FDB entry. In this example, the entry is displayed by using the command **show platform table fdb** on an x900 series switch.

```
FDB table (software MAC table)
...
--------------------------------------------------------------------------
  Index  VLAN MAC                Port/Vidx Status Usage     daRoute
   thrashToken
--------------------------------------------------------------------------
  4584   2   0009.41fd.c059      1.1.1     dyn    MAC       0       10
...
```

There are a number of fields in the entry, but the three important ones: port, MAC address, and VLAN are highlighted. These entries are universal, and will appear in the FDB entries of **any** vendors' switch. The other fields in the entry relate mostly to the internal structure and performance optimisations for the specific ASIC in use.

Fundamentally, the task of populating the Layer 2 unicast table is a matter of associating a port number with every MAC-address/VLAN combination that traverses the switch.

This means that when **any** packet arrives at the switch, destined to some MAC address Y in some VLAN Z, the switch can look into its FDB to immediately choose which port to send that packet to.

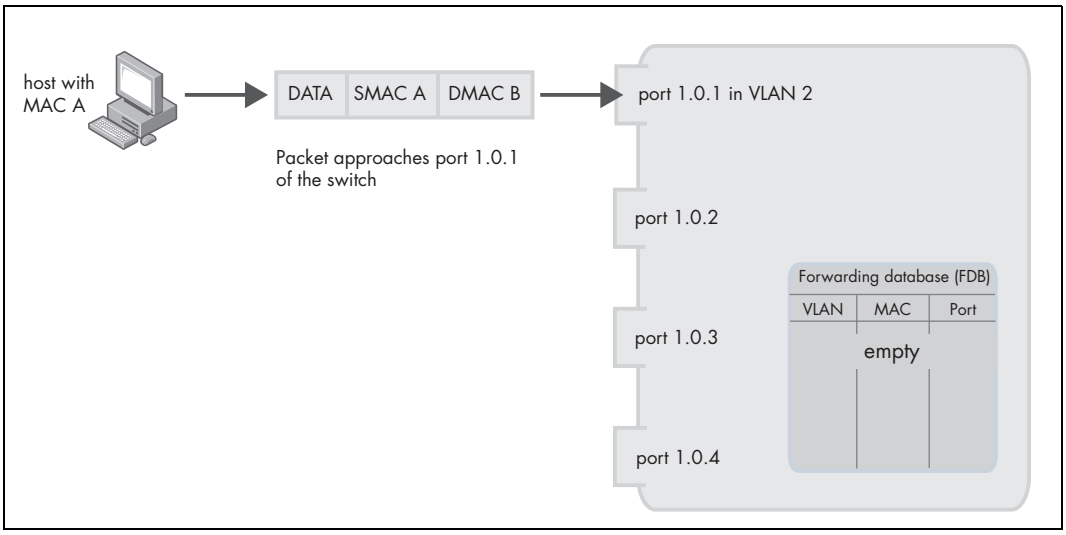
How entries get into the Layer 2 unicast table

When a switch is first powered up, its FDB is, of course, empty. The switch cannot possibly know in advance all the MAC addresses in use in the network, and which VLANs all those MAC addresses reside in. So, it needs to **learn** all the MAC-address/VLAN combinations as packets start to flow through it.

In essence, this learning process is very simple. If the switch sees a packet arrive on VLAN X on port Y with source MAC A, it says "I have now learnt that the host with MAC address A can be reached on VLAN X via port Y. I will store that information in the FDB."

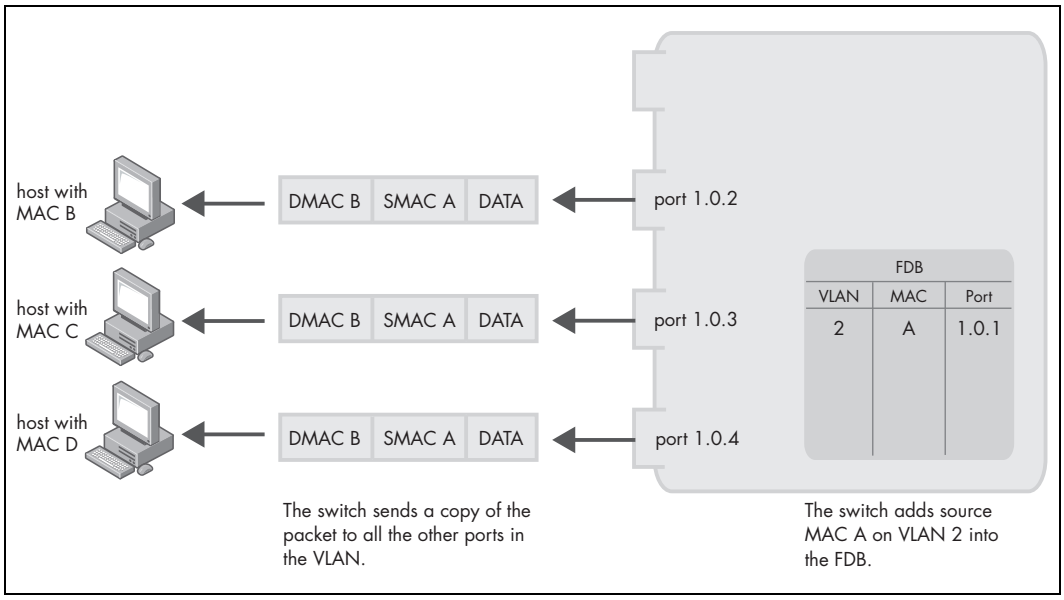
This learning process is essentially such a mechanical process that, in a lot of switches, it is performed entirely in hardware. In such switches, the CPU does not get involved at all. However, in more sophisticated switches (like the AT-9924, x900 series, etc.) security reasons (like 802.1x authentication, port learn limiting, etc.) could easily mean that certain MAC addresses on certain VLANs should **not** be put into the FDB table. The switching hardware cannot have knowledge of such reasons. So, in these more sophisticated switches, the MAC learning is controlled by software. The ASIC sends an interrupt to the CPU to say "I just received a packet whose source MAC/VLAN combination is not already in the FDB. Please examine this information and decide if you want to create a new FDB entry for this". Of course, this software-controlled learning has to operate **fast**, to minimise any impact on network performance that might be caused by a significant delay in the learning process.

The process is illustrated in the following diagrams. These diagrams show what happens when the very first packets arrive at the switch when the FDB is empty.

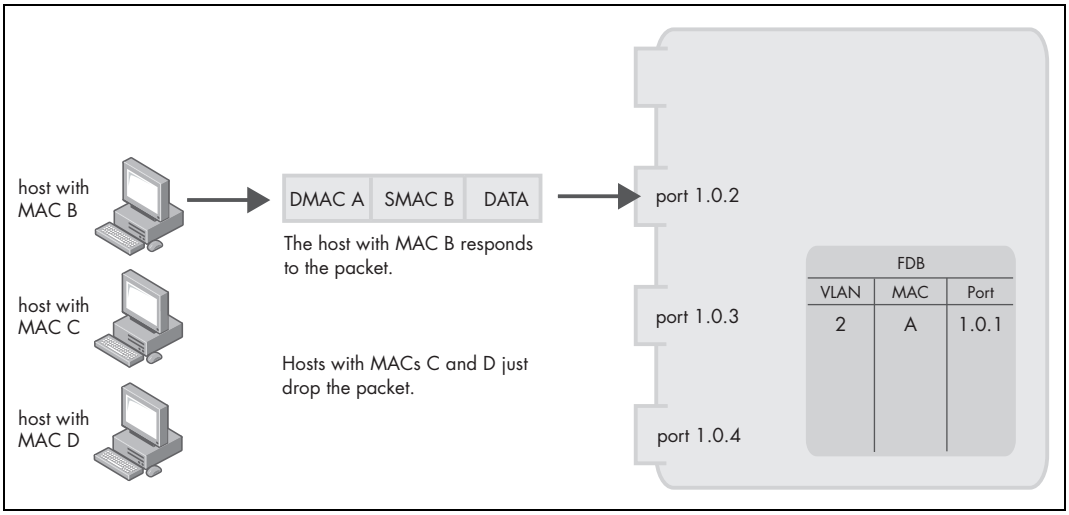


As you can see in the second diagram below, the switch can create an FDB entry based on the source MAC of the packet, but if it has not previously seen the destination MAC of this packet, it will not know exactly where to send the packet. So, it must "flood" the

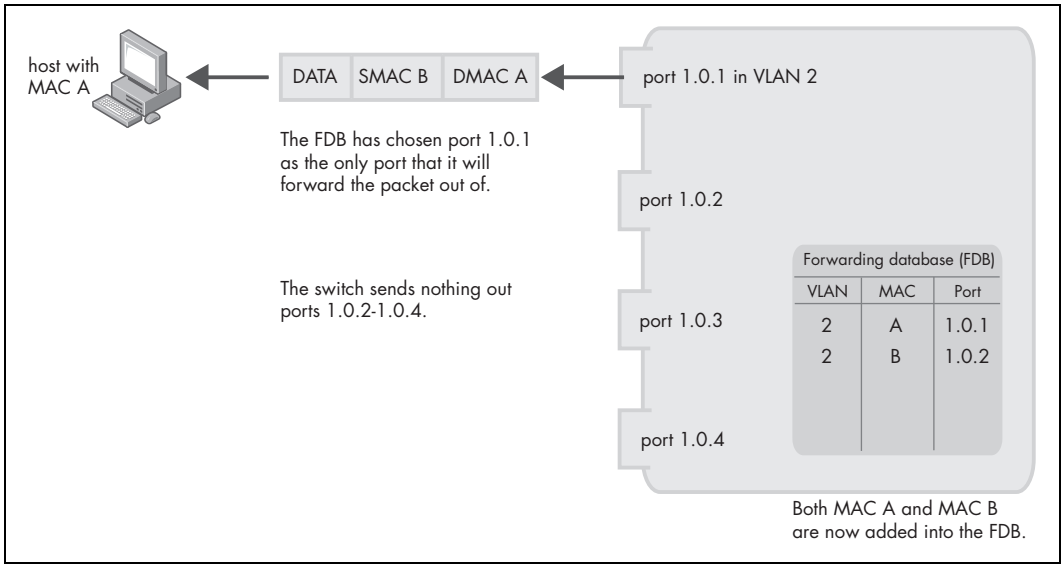
packet to all other ports in the VLAN that the packet arrived on, in the hope that the destination device can be reached via one of these ports.



If the network has been connected up properly, then the packet will reach the destination device, which will reply, as the following diagram shows. All the other devices on the other ports of the VLAN will simply silently discard the packet.



Then, the switch will receive this reply packet, will learn which port that destination device was connected to, **and** will know exactly which port to send this reply packet to (because it has already learnt that MAC A is reached via port1.0.1). The following figure shows this.



Subsequent packets exchanged between hosts A and B will be able to be hardware switched to exactly the right port every time.

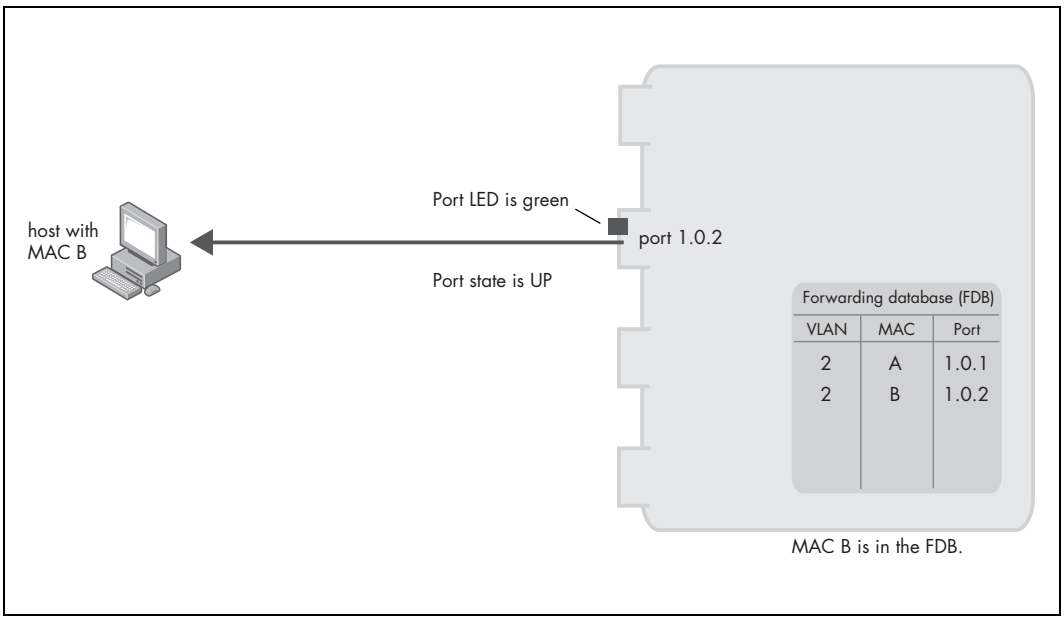
How entries are removed from the Layer 2 unicast table

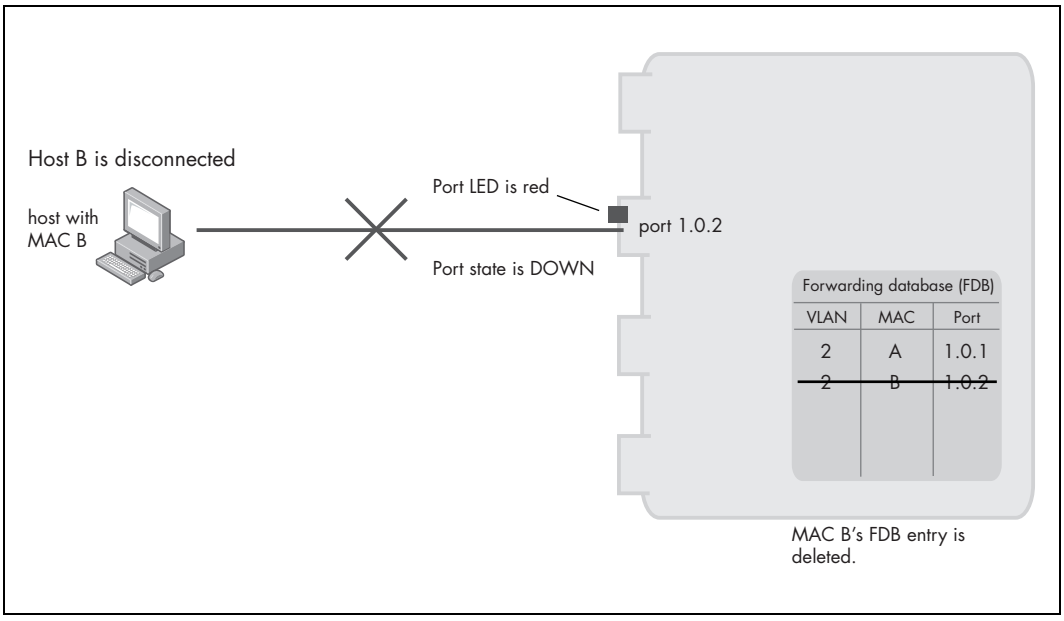
As described above, a switch populates its FDB by learning from the packets that pass through it. If networks were 100% static—no workstations were ever turned off, or moved, or replaced—then the FDB learning process would only have to be performed once after start-up, and the contents of the FDB could remain the same forever.

In reality, networks are never 100% static. The process that manages the FDB needs to account for the inconvenient fact that the world changes from time to time. So, the switch needs a process for removing entries from the FDB. There are three main events that remove entries from the FDB: ports going down, FDB entries aging out, and STP/ EPSR topology changes.

Ports going down

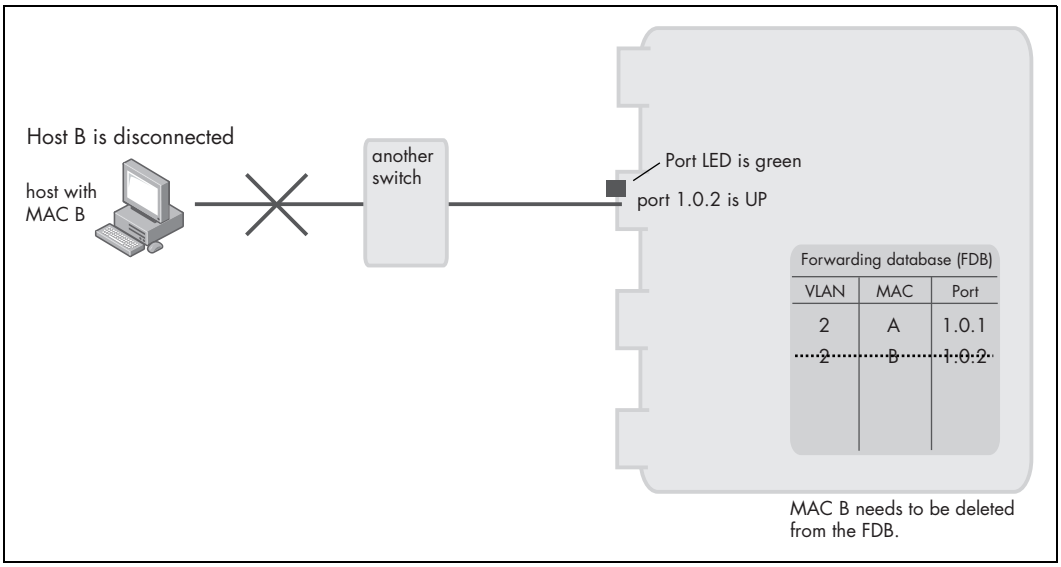
The first process is simply that when a port goes from up to down, any FDB entries associated with that port are deleted, as obviously there is no point in trying to send packets out a port that is down. The following two diagrams show what happens when a host is disconnected.





FDB entries aging out

The second process is that of FDB aging. The switch needs to age FDB entries because not every address to which a switch forwards packets belongs to a host that is directly connected to a port on that switch. So, when one of these further-away hosts is disconnected from the network, how do all the other switches in the network find out about this, and so know to remove that host's entries from their FDBs?



In fact, there is no mechanism that advertises to the whole network that a certain host has been disconnected. Each switch needs to separately perform housework on its own FDB, cleaning out any old no-longer-used entries that are in the table.

Different switches use different mechanisms to perform this housework. But the most common mechanism is the 'hit bit'. With this, every entry in the FDB has a bit associated with it. At a regular periodic interval (known as the aging time), the switch will set this bit to 0 on every FDB entry. Then, whenever the switch receives a packet and the packet's source MAC/VLAN match a particular FDB entry, then the switch sets the hit bit on that entry to 1. (Of course if the switch receives 100,000 packets that all match a given entry, it is really only the receipt of the first packet that results in the hit bit being changed from 0 to 1. For the other 99,999 packets the hit bit simply stays set to 1).

Then, the next time the aging timer goes off, the switch checks through for any hit bits that are still sitting at 0. These indicate FDB entries that correspond to hosts that have not sent any packets since the last time the aging timer went off. The switch deletes these entries, and then sets the hit bit to 0 on all other entries.

You could say "well, maybe the hosts that have not sent packets are still on the network, but have just been quiet for a while." You would be right: there is a risk of the FDB aging process removing entries relating to devices that are just quiet, not disconnected. However, that is not really such a problem; if those hosts start sending packets again later on, the switch will very quickly learn a new entry for them. What is important, though, is that the aging process will definitely remove any entries relating to hosts that **have** been disconnected.

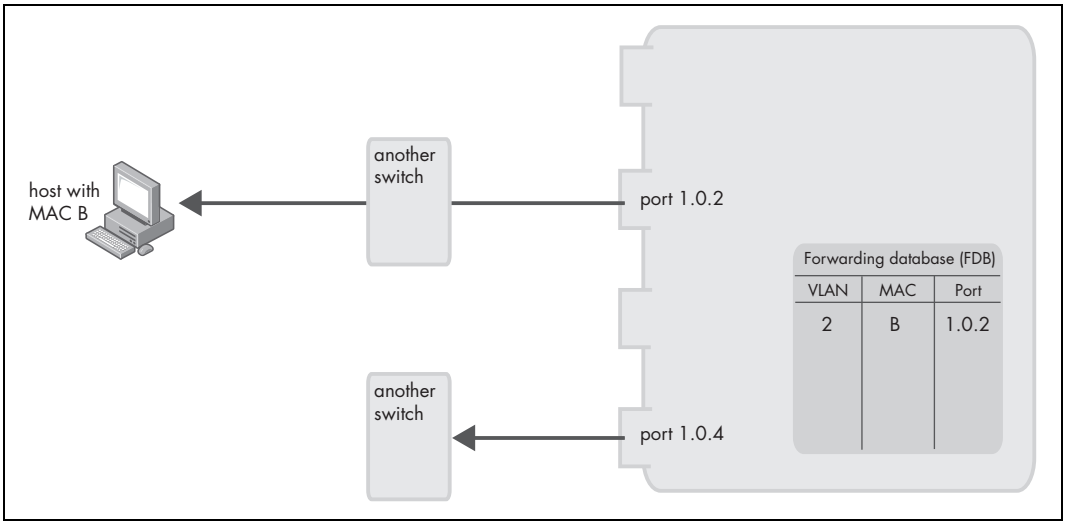
Topology changes

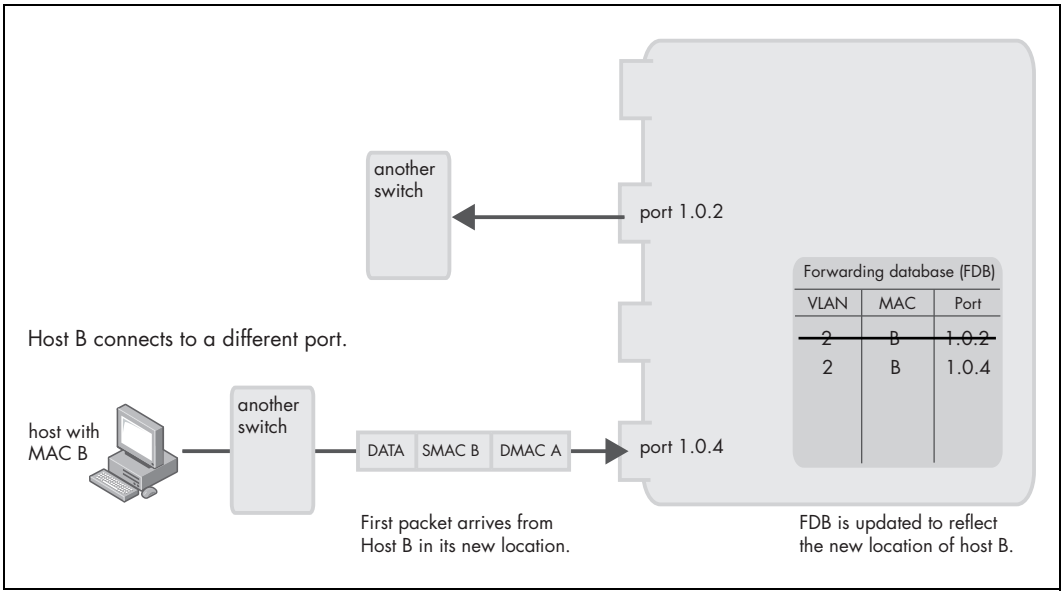
As described on the section "Reacting to topology changes" on page 46, and the section "Reacting to a ring failure" on page 81, topology changes in looped networks will cause entries to be flushed from switches' FDBs. This is because in the new topology, the port via which to reach various hosts in the network may have changed, so the switch needs to forget what it had learnt in the previous topology, and relearn the locations of MAC addresses in the new switch topology.

How entries are updated in the Layer 2 unicast table

Sometimes a remotely connected host will be disconnected from one remote switch and reconnected on another remote switch, all in a period of time less than the switch's FDB aging timer. In that case, the switch will receive a packet on a particular port, even though it has an FDB entry saying that the host is reached via a different port. The following diagrams show this:

.





What should the switch do in this situation? Well, the rule is, that as long as the host is still in the same VLAN, the switch should remove the old information it had learnt about this host, and replace it with the new information the switch has just leant from the newly arrived packet, as the second diagram above shows.

Static FDB entries

In some situations, the switch needs to have certain entries written into the FDB permanently. This is usually for security reasons—so that a switch will never be fooled by an attempt to spoof the MAC address on a different port, and so that traffic for this host is never flooded.

Under AlliedWare Plus, you can configure static entries by using a command like this:

```
mac address-table static 0000.aaaa.bbbb forward interface
   port1.1.1 vlan 2
```

The resulting MAC entry will look like (**show platform table fdb**):

```
-----------------------------------------------------------------------
Index  VLAN MAC         Port/Vidx Status Usage     daRoute thrashToken
-----------------------------------------------------------------------
2264    2   0000.aaaa.bbbb  1.1.1  sta    MAC       0         10
```

Note that the status is marked as **sta** (for static).

A static entry like this will never be removed by the FDB aging process, and will never be overwritten by the FDB updating process. It is nailed down permanently, and can only be removed by explicit user intervention with a command like the following:

```
no mac address-table static 0000.aaaa.bbbb forward interface
   port1.1.1 vlan 2
```

FDB filter entries

It is also possible to explicitly tell the switch to drop packets that are destined for a given MAC address. This is called an FDB filter entry, and is created by using a command like this:

```
mac address-table static 0000.aaaa.cccc discard interface
   port1.1.1 vlan 2
```

CPU entries

It is very common to see FDB entries like the following:

```
-------------------------------------------------------------------
Index  VLAN MAC           Port/Vidx Status Usage  daRoute thrashToken
-------------------------------------------------------------------
  306    2    0000.cd28.5253  CPU     sta    MAC       1       10
```

What is the meaning of these entries that have "port = CPU"? The interesting thing to note about these entries is that typically the MAC address of the entry is the MAC address of the switch itself. And, typically one of these entries exists for every VLAN on the switch on which an IP address has been configured.

The purpose of these entries is to tell the switch what to do when it receives a packet whose destination MAC address is the MAC address of the switch itself. Of course, the switch **should** never learn any entries for its **own** MAC address (because packets should never enter the switch with a source MAC address equal to the switch's own MAC address). So, there needs to be a mechanism to get entries into the FDB that **do** cover the case when a packet arrives destined to the switch's own MAC address. Hence, the software creates these static entries at start-up to say "these packets are destined to **me**, so please send them to the CPU, which will then decide what to do with them".

Typically, these packets will be management packets that are destined to be processed by software modules in the switch (telnet sessions, SNMP messages, pings, ARP replies, etc.).

The Layer 2 multicast table

The purpose of the Layer 2 multicast table

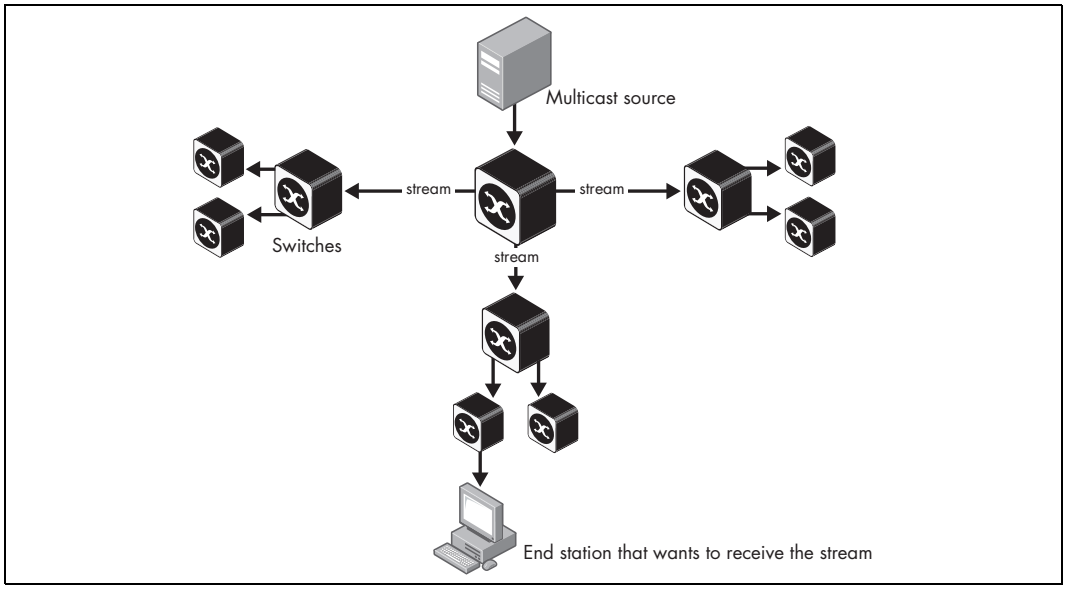
The first thing to make clear is that the Layer 2 multicast table is used exclusively for forwarding IP multicast, not general Layer 2 multicast packets. So, it is only relevant to packets that have a destination MAC address of the form 0100.5exx.xxxx.

To understand the purpose of the Layer 2 multicast table, it is necessary to reflect briefly on IP multicasting, in particular on why IGMP—the Internet Group Management Protocol—was developed. This will provide the background to understanding what the Layer 2 multicast table is aiming to achieve, and why IGMP is so central to the management of the Layer 2 multicast table.

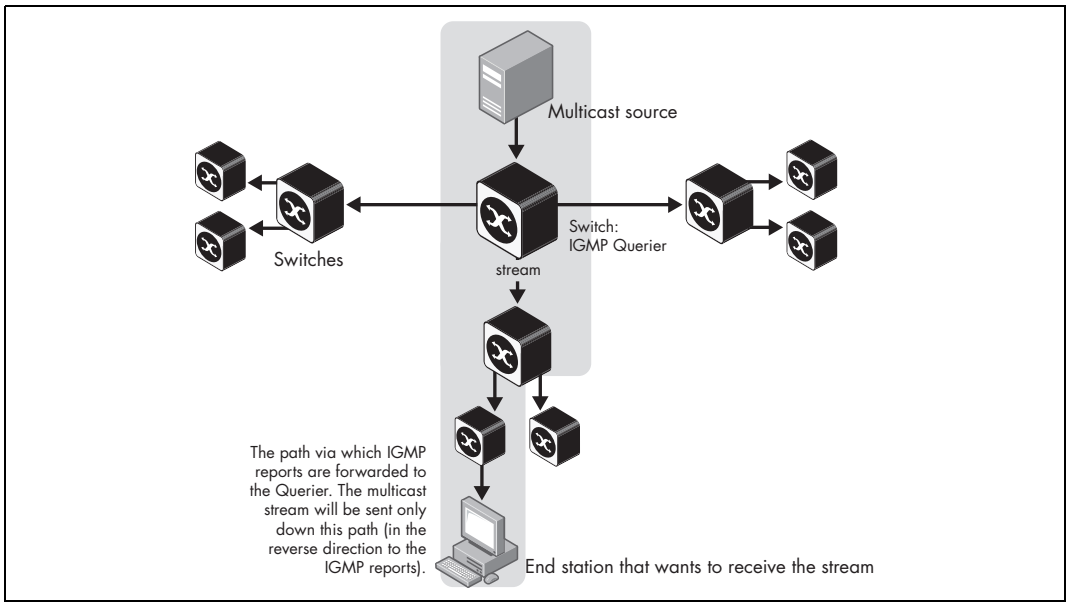
Just as the transition from networking with hubs to networking with switches was driven by the desire to improve network performance, the development of the protocols and processes for handling multicast traffic has also been driven by network performance considerations.

In the absence of the IGMP protocol, a Layer 2 switch would forward an IP multicast packet to all ports in the VLAN that it was received on. In a network where a multicast stream is being forwarded through multiple Layer 2 switches, this could lead to an enormous number of extra packets being forwarded onto LAN segments where they are not required. The purpose of IGMP is to let specific end stations signal their desire to receive a particular multicast stream. It also lets intermediate switches learn exactly which ports they need to forward a given multicast stream to (via IGMP snooping).

The following figure shows an example of a multi-switch network. In the absence of IGMP, multicast streams would be forwarded to numerous LAN segments in this network, even if only one end station wanted to receive the stream.



With IGMP, the switches know exactly where the stream needs to be forwarded, and so send the stream nowhere else.



For an Layer 2 switch, the IGMP snooping process is used to learn which ports given multicast streams should be forwarded to, by watching which ports IGMP reports for given multicast groups arrive on. The operation of IGMP and IGMP snooping are described in more detail in Chapter 2: "Understanding IGMP Snooping" on page 28

Information that is stored in the Layer 2 multicast table

Entries in the Layer 2 multicast table hold three significant pieces of information: MAC address, VLAN, and port list. These are, of course, very similar to the items in an entry in the Layer 2 unicast table (see page 108). The only difference is that the multicast table entries hold a list of egress ports instead of a single egress port.

A typical entry in the Layer 2 multicast table is shown below. In this example, the entry is displayed by using the command **show platform table l2mc** on an x900 series switch.

```
------------------------------------------------------------------------
        Total number of entries = 1
------------------------------------------------------------------------
Index  MAC               VID    MCGroup   CPU_MEM       NumPorts
       PORT_LIST
------------------------------------------------------------------------
0      0100.5e4c.2d03    1       4097           0            2
       Local Ports = 1.0.13, 1.0.19
```

Again, as with the unicast table, there are other ASIC-specific pieces of information stored in the entries, but the essential items used in the packet-forwarding decision are the MAC, VLAN, and port list.

When an IP multicast packet with a given destination MAC arrives at a port that is a member of a given VLAN, then the corresponding entry in the Layer 2 multicast table will show which port(s) to forward this packet to.

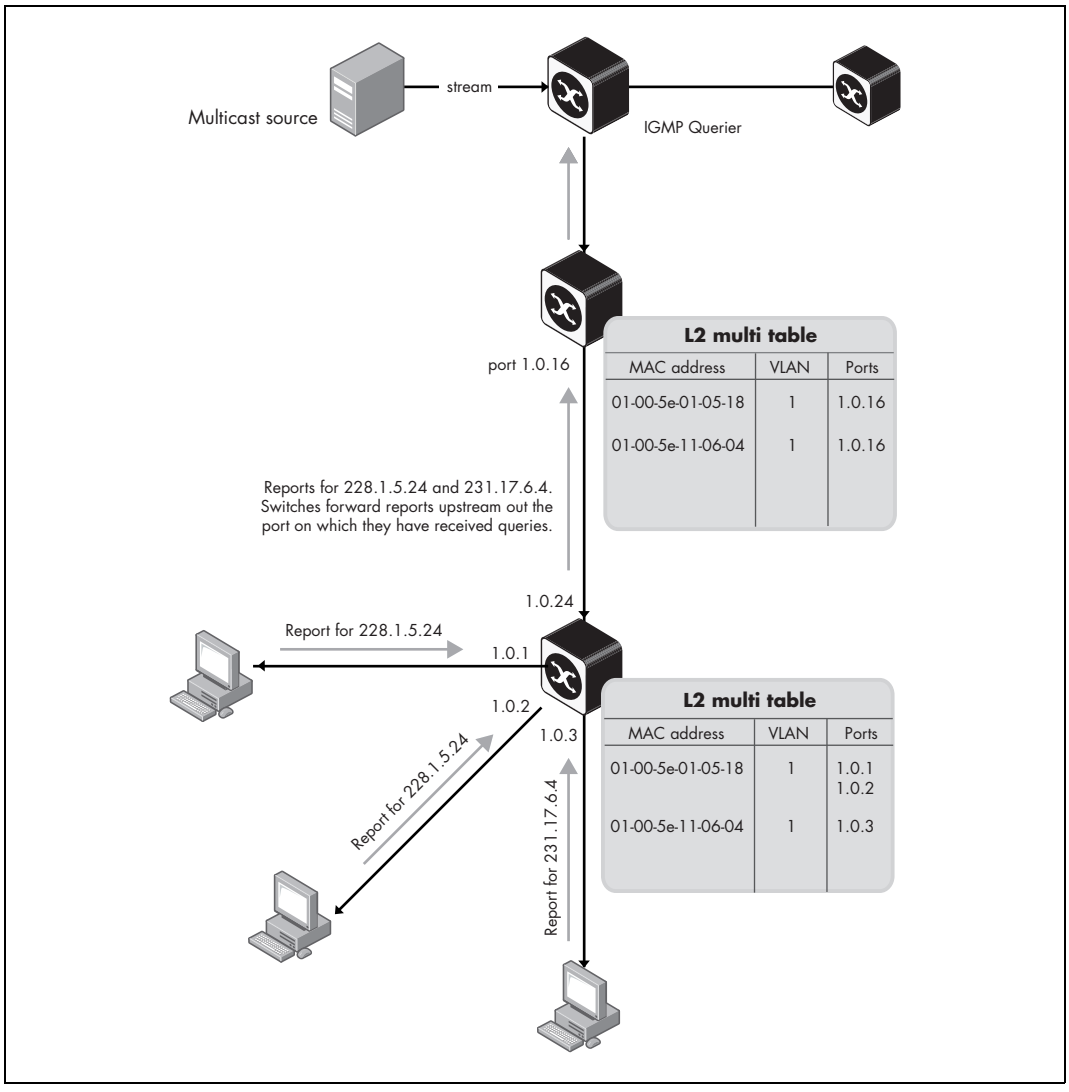
Now, there is one very important difference between the operation of Layer 2 unicast forwarding and Layer 2 multicast forwarding. The difference relates to what the switch does if it cannot find an entry that corresponds to the VLAN/dest-MAC combination of the received packet. In the case of unicast forwarding, the packet would be flooded. In the case of multicast forwarding the packet is dropped.

This is because of the fundamental difference between unicast and multicast communication. Unicast communication is a two-way conversation between two hosts that need to find each other on the network. The switch must facilitate their finding each other, by flooding packets to unknown unicast addresses, so that they can eventually reach the intended destination host. Multicast communication, though, consists of a multicast source sending out traffic streams with no idea of who the recipients of the streams are (if any). The recipients need to inform the **network** (**not** the source) of their desire to receive the streams, by sending IGMP reports. So, if a switch has not received IGMP reports for a particular multicast group (and so, does not have Layer 2 multicast table entries for that group) then there are no downstream recipients who have requested a stream with that group address, so the switch simply does not forward the packets.

How entries get into the Layer 2 multicast table

The populating of the Layer 2 multicast table is completely dependent on IGMP. The receiving IGMP reports is what causes the switch to create entries in the table.

The mechanism is quite simple. If the switch receives a report for group address A on a port in VLAN X, then it will create an entry containing that group address/VLAN/port combination. If an entry already exists for the same VLAN and group, but a different port (or set of ports), then the new port is simply added to the list of ports in that existing entry.



Note in the diagram above, that the switch will also forward the reports up towards the IGMP querier, so that a chain of forwarding ports will be created all the way back up to the querier.

This process is all under software control. The typical mechanism is that if IGMP snooping or an IGMP querier is configured on a unit, then the switch chip is instructed to catch **all** incoming IGMP packets, and send them to the CPU to interpret. The software examines the IGMP packets, and manages the Layer 2 multicast table based on the contents of the IGMP packets.

How entries are removed from the Layer 2 multicast table

There are three mechanisms by which entries are removed from the table, or ports are removed from the port list in an entry.

Aging

When a particular port is added to the egress port list for a particular group address on a particular port, a timer (by default, 250 seconds) is started. Every time an IGMP report is received for that group on **that** port, the timer is reset back to the start. If the timer ever expires, then the port is removed from the list.

IGMP queriers send out periodic queries for all the groups that they have received reports for, to cause listening hosts to send reports in response to the queries. This refreshes the table-entry timers in the switches on the path between the hosts and the querier.

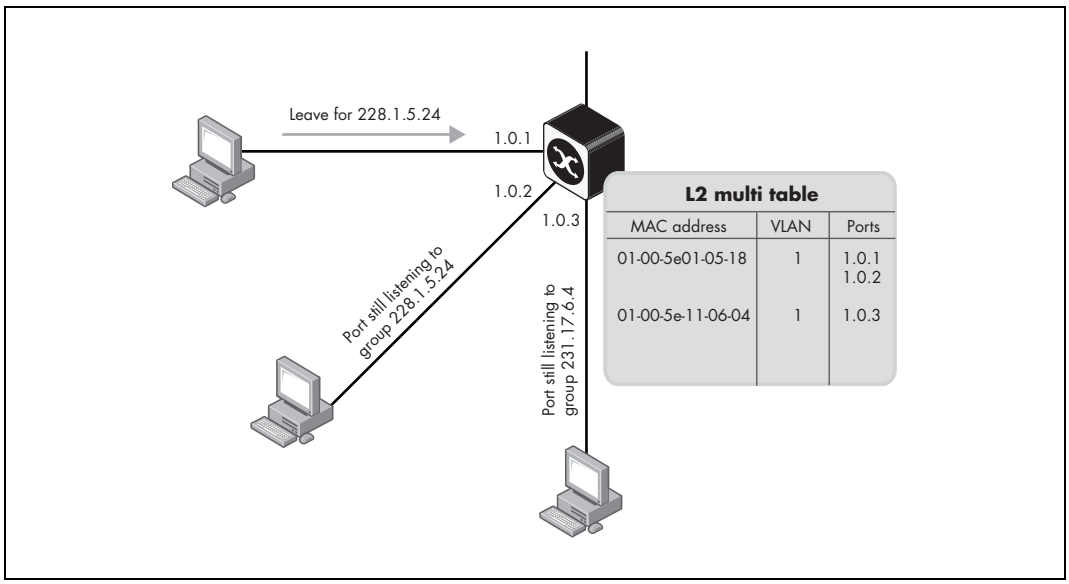
Group aging

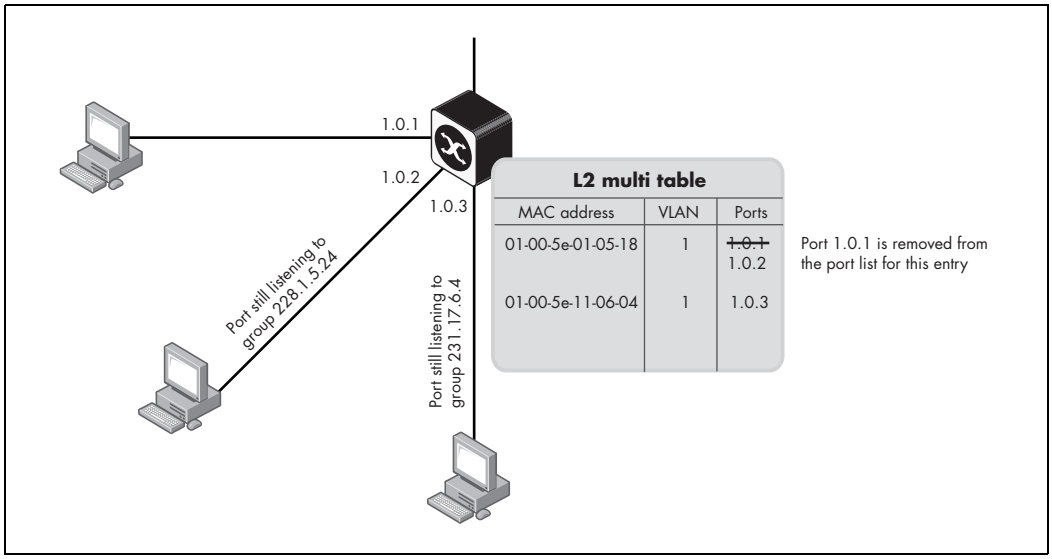
When the last port is removed from the egress port list for a group entry, then: if no more data is arriving for this group, the whole entry is deleted from the table. If data is still arriving for the group, the entry remains, with an empty port list.

IGMP leave messages

The IGMP protocol (from version 2 onwards) includes a packet type known as the "leave".

When a host wishes to inform the network that it no longer desires to receive streams to a given group address, it sends an IGMP leave for that group. When the switch receives the IGMP leave, it will remove the receiving port from the port list for the entry in question.





If this was the only port in the port list for the table entry, then the entire table entry is removed, if no multicast stream, destined to that group, is still arriving at the switch

Speed of removal

The port might not be removed from the port list in the table entry immediately upon receipt of the IGMP leave. Instead, the switch might wait for the querier to send out a query to check whether other hosts are actually still listening to the group on that port.

Whether the port is removed immediately upon receipt of the leave message, or whether the switch waits for the querier to send out a query, depends on whether or not the switch has been configured with IGMP **fast leave** on the port in question. When fast leave is configured, the switch does not wait—it immediately removes the port entry and so immediately stops sending the multicast stream to that 'fast-leave' port.

Fast leave is only suitable for ports that are connected to a single host. i.e. do not have another switch between themselves and the host.

To turn on fast leave on a VLAN, use the following commands:

```
awplus(config)#interface vlan1
awplus(config-if)#ip igmp snooping fast-leave
```

Port down

As with the Layer 2 unicast table, when a port goes down, it is removed from the Layer 2 multicast table. (It is just as difficult to send multicast packets out a down port as it is to send unicast packets out a down port.)

Static entries

Static entries in the Layer 2 multicast table are created by creating static IGMP membership entries, by using the commands:

```
awplus(config)#interface vlan1
awplus(config-if)#ip igmp static-group 228.1.5.24 interface
   port1.0.1
```

Avoiding overlapping group addresses

Another complication is that multicasting is designed to use each packet's group IP address to determine a multicast MAC address to send the packet to. However, as described in the section "Multicast groups" on page 22, multicasting does not have a 1:1 mapping of IP address to MAC address—instead each multicast MAC address corresponds to 32 multicast IP addresses. This means that different multicast IP addresses use the same MAC address.

The MAC address only uses the last 23 bits of the IP address; it ignores the IP's first octet and the first bit of the second octet. Note that all IP multicast MAC addresses start with 01-00-5E.

You need to avoid using multiple IP addresses that have the same MAC address. In practice, this means that **if you use x.0.y.z, then do not use x.128.y.z (or vice versa)**, where x is anything from 224-239, and y and z are the same in each IP address. For example, if y=6 and z=200 then these IP addresses use the same MAC: 224.0.6.200, 224.128.6.200, 225.0.6.200, 225.128.6.200, etc.

To see this in detail, consider 224.0.6.200. This has a multicast MAC of 0100.5E00.06C8, like this:

IP address, decimal:	224. 0. 6. 200
IP address, binary:	11100000 00000000 00000110 11001000
MAC address, binary:	0000000 00000110 11001000
MAC address, hex:	0100.5E00.06C8

Therefore, the following multicast IP addresses will all have the same MAC address as 224.0.6.200, because their last 23 bits are all the same:

IP address, decimal:	IP address, binary:	
224.0.6.200	11100000 0	0000000 00000110 11001000
224.128.6.200	11100000 1	0000000 00000110 11001000
225.0.6.200	11100001 0	0000000 00000110 11001000
225.128.6.200	11100001 1	0000000 00000110 11001000
226.0.6.200	11100010 0	0000000 00000110 11001000
226.128.6.200	11100010 1	0000000 00000110 11001000
227.0.6.200	11100011 0	0000000 00000110 11001000

IP address, decimal:	IP address, binary:	
227.128.6.200	11100011 1	0000000 00000110 11001000
...	...	...
239.0.6.200	11101111 0	0000000 00000110 11001000
239.128.6.200	11101111 1	0000000 00000110 11001000
	Different IPs	The same multicast MAC

Avoid x.0.0.y, x.0.1.y, x.128.0.y, and x.128.1.y

It is particularly important to avoid using any address in the ranges x.0.0.y, x.128.0.y, x.0.1.y, or x.128.1.y (where x is 224-239 and y is 1-254).

This is because x.0.0.y and x.128.0.y will map to the same multicast MAC address as 224.0.0.y. Similarly, x.0.1.y and x.128.1.y will map to the same multicast MAC address as 224.0.1.y. Most addresses in the ranges 224.0.0.y and 224.0.1.y are reserved for contacting all routers, or for routing protocol messages, so they are always flooded out all ports in the relevant VLAN.

Therefore, all addresses in the ranges x.0.0.y, x.128.0.y, x.0.1.y, or x.128.1.y are flooded out every port in the relevant VLAN. Using these addresses can significantly increase multicast traffic in your network and cause cpu performance degradation.

If you are debugging a situation where it seems that certain multicast groups are forwarded when you think they shouldn't be, check whether the choice of group addresses has violated any of the recommendations above. For a detailed list of reserved multicast addresses, see: http://www.iana.org/assignments/multicast-addresses/

Troubleshooting Layer 2 multicast issues

Failure to forward a stream

If a multicast stream is not being forwarded to the ports you think is should be forwarded to, then use the following steps to debug the situation:

1. Check that multicast stream is actually arriving at the upstream side of the switch.

   Usually the best way to do that is to mirror the upstream port to a sniffer port, and use a sniffer to check that the packets of the stream are arriving at that port.

2. Check that the VLAN that the stream is arriving on is the same as the VLAN that you want it transmitted on.

   Is the upstream port in the same VLAN as the downstream port? Are the packets in the stream maybe arriving tagged with some other VID?

3. Are the IGMP report packets arriving at the switch from the client device?

4. Is IGMP snooping correctly interpreting the reports, and creating entries based on them? Are the Layer 2 multicast entries in the correct VLAN? Maybe the reports are coming in tagged with the wrong VID?

You can use IGMP debugging to check that IGMP snooping is receiving the packets:

```
awplus#debug igmp decode
awplus#terminal monitor
% Warning: Console logging enabled
awplus#12:25:29 awplus NSM[1537]: [IGMP-DECODE] : IGMP V2 Membership
    Report, Max
. Rsp. Code 0 on port1.0.19
12:25:29 awplus NSM[1537]: [IGMP-DECODE] : Grp 226.76.45.3 on vlan1
```

To see the details of the IGMP snooping table, you can use the command **show ip igmp groups detail**:

```
awplus#show ip igmp groups detail

Interface:      vlan1                       - this is the VLAN
Group:          226.76.45.3                 - this is the group's address
Uptime:         00:07:39
Group mode:     Exclude (Expires: 00:04:13)
Last reporter:  40.40.40.23
TIB-A Count:    0
TIB-B Count:    0
Source list is empty
Port member list:                           - this is the port list
port1.0.19 - 27 secs
Source list is empty
port1.0.13 - 254 secs
Source list is empty
```

5. Do the entries in the hardware Layer 2 multicast table correctly reflect the IGMP snooping entries?

   Check that the VLAN, address, port-list information in the hardware Layer 2 multicast table entries match the corresponding information in the entries in the IGMP snooping table.

Intermittent loss of the stream

It is quite common for the multicast traffic to be delivered for a while, and then stop arriving, and then maybe start being delivered again for a while, and so on.

Typically, this is due to a mismatch between the ageing timer on the switch, and the query timer on the querier. As explained above, the switch refreshes the entries in its Layer 2 multicast tables when it sees reports, and reports are periodically generated by the receiving hosts in response to the queries that are sent down from the querier.

But, if the time-lapse between the queries being sent down is longer than the aging timer on the switch, then the switch will age out the entries before they can be refreshed by the reports from the hosts. So the multicast will stop when the entries age out. When the query is eventually sent, the hosts will generate reports, and the switch

will create new Layer 2 multicast table entries in response to those reports, and the multicast will start flowing again.

To debug this situation, it is necessary to monitor the IGMP activity going through the switch. When are the queries sent down? When do the reports arrive from the client devices? At what point does the switch delete the entries from its Layer 2 multicast table?

For detailed information about the relationships between different IGMP timers and counters, see "Configurable IGMP timers and counters" in How To Configure IGMP for Multicasting on Routers and Managed Layer 3 Switches. This How To Note is available from www.alliedtelesis.com/resources/literature/howto.aspx.

Stream is forwarded for a few minutes, then stops

A problem that occurs sometimes is that the client device will request a multicast stream, and successfully receive it for about 4 minutes, then the stream stops. If the client requests the stream again, the same thing happens. Typically, this occurs when IGMP queries are not reaching the client. Most multicast client devices will not send further IGMP reports unless they are queried. If the client is not receiving queries, and so, not sending further reports, the snooping switch the client is connected will age out the entry for that group. Possible reasons for the client not receiving IGMP queries are:

- there is no querier in the network.
- some device between the querier and the client is blocking queries.
- the querier is not sending queries on the client's VLAN
- an IGMP version mismatch. The querier may be sending out IGMPv3 queries, but the client only supports IGMP v2.

Multicast being forwarded to ports that it should not be forwarded to

You can be confident that the switch will only forward multicast packets to the ports that the hardware table tells it to forward to (with the exception of addresses of the form x.0.0.y, x.0.1.y, x.128.0.y, x.128.1.y, which are flooded, as explained on page 124).

If you are seeing a stream forwarded to a port that you do not expect it to be forwarded to, then you **should** find an entry in the Layer 2 multicast table telling it to forward the stream to that port, unless IGMP snooping has been disabled (in which case, all multicast groups will be flooded).

The task is to then work out how that Layer 2 multicast table entry got created. It might be that IGMP reports for the group in question **are** actually being received on the port in question, and you just did not realise it. But, if the relevant IGMP reports are not being received, possibly IGMP packets are slightly malformed and the IGMP processing is misinterpreting the group address in the IGMP reports; etc.

The debugging steps are effectively the opposite of those for the "Failure to forward stream" case:

1. Do the entries in the IGMP snooping table match the entries in the Layer 2 multicast table? If not, then IGMP snooping is probably processing the IGMP reports correctly, but there is a bug in the part of the code that finally writes the entries to the hardware Layer 2 multicast table.
2. If the entries in the IGMP snooping table are also incorrect, then maybe there is a bug in the processing of the IGMP reports.

The important thing is to follow through the logic of how Layer 2 multicast table entries are created, and see where the point of inconsistency is occurring.

Chapter 8 | Management of Forwarding Tables - Layer 3

Introduction

In this chapter, we will look at the tables used in unicast Layer 3 switching, and multicast Layer 3 switching.

Note, throughout this chapter, there are references to commands for displaying hardware tables, and illustrative snippets of hardware table output. These commands and output examples are all taken from the x900 and x908 switches. On other models of switches, like the x600, the commands and the format of the table output may differ slightly. However, the concepts described in this chapter are universal to Allied Telesis x-series switches.

List of terms

Layer 3 switching

A Layer 3 switch is an optimized combination of routing software and specialised hardware. The software uses traditional methods (static routing commands, and routing protocols) to build up a table of the best routes to network destinations, and then writes them into a set of registers in the specialised forwarding hardware. The hardware then forwards packets, based on their Layer 3 address content, at very high data rates, using the values that are written into the registers.

Layer 3 multicast forwarding

Just as Layer 3 unicast forwarding is required to transport IP packets from a source device that is in one IP subnet to a destination that is in another subnet, switches also need to transport IP multicast data from one subnet to another. However, the process for deciding which destination subnet(s) to which to forward multicast is significantly different to the decision process for unicast forwarding. The destination IP address in a multicast packet is a multicast group address, that contains no inherent information to identify the subnet(s) it needs to be forwarded to. So, Layer 3 multicasting depends on a signalling protocol that enables switches to find out which subnets contain hosts who have joined given groups.

The Layer 3 unicast table

The purpose of the Layer 3 unicast table

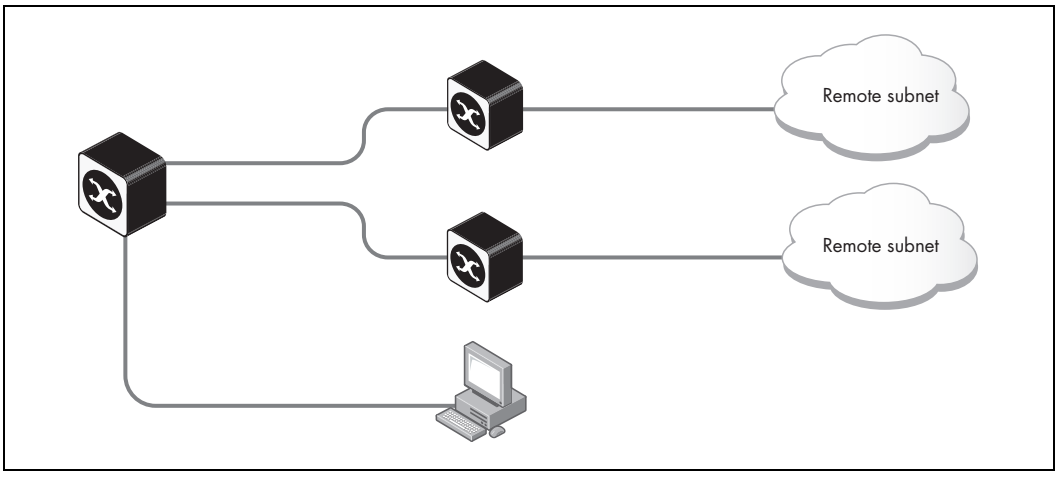
The presence of a Layer 3 switching table in a switch is what makes it a "Layer 3 Switch". The Layer 3 unicast table is a hardware cache of the IP route table, and it enables IP routing to be performed in hardware at wire speed.

Again, performance was the drive behind the development of this functionality. But, it was a different aspect of performance to that which drove the development of Layer 2 switching. In the case of Layer 2 switching, the driver was the reduction of network congestion (with its attendant degradation of network performance). In the case of Layer 3 switching, the driver has the need to increase data throughput through routing devices. Prior to the development of Layer 3 switches, all routing was performed in software based routers. As Ethernet speeds increased (10Mbps to 100Mbps to 1Gbps), the throughput that could be achieved by software routing in even a fast CPU fell well behind the bandwidth available on Ethernet networks. The only way to make the very large performance improvement required to route at multiple gigabit speeds was to develop dedicated hardware routing engines (i.e. switching chips that contain Layer 3 forwarding tables).

Information that is stored in Layer 3 unicast table

There are two classes of entry in the table: route entries and host entries.

Route entries are the routes to remote subnets (i.e. those subnets that are separated from the switch by at least one router). Host entries are for forwarding to individual hosts within directly connected local subnets.



Route entries are effectively hardware copies of the entries in the software routing table, but boiled down to their essentials: destination subnet address, subnet mask, egress port, egress VLAN, and nexthop MAC address.

Host entries are effectively hardware copies of entries in the software ARP table, and contain: destination IP address, egress port, host MAC address, egress VLAN, and an all-1s subnet mask.

So, in fact, the content of a host entry is the same as that of a route entry. It is interesting that, when it comes down to it, the same information is required to Layer 3 switch to a remote subnet as to a locally connected host.

A typical entry in the Layer 3 unicast table is shown below. In this example, the entry is displayed by using the command **show platform table ip** on an x900 series switch.

```
IPv4 Unicast Route Table: Driver Shadow
------------------------------------------------------------------------------
Index  IP address/prefLen NextHop          EnCap Dev Port Tr VID  Mtu   Ttl Cnt
------------------------------------------------------------------------------

4      187.96.78.0     24 00-00-cd-1d-7d-72  0    0   1    -  1    10240 Ena  2

...
------------------------------------------------------------------------------
```

How entries get into the Layer 3 unicast table

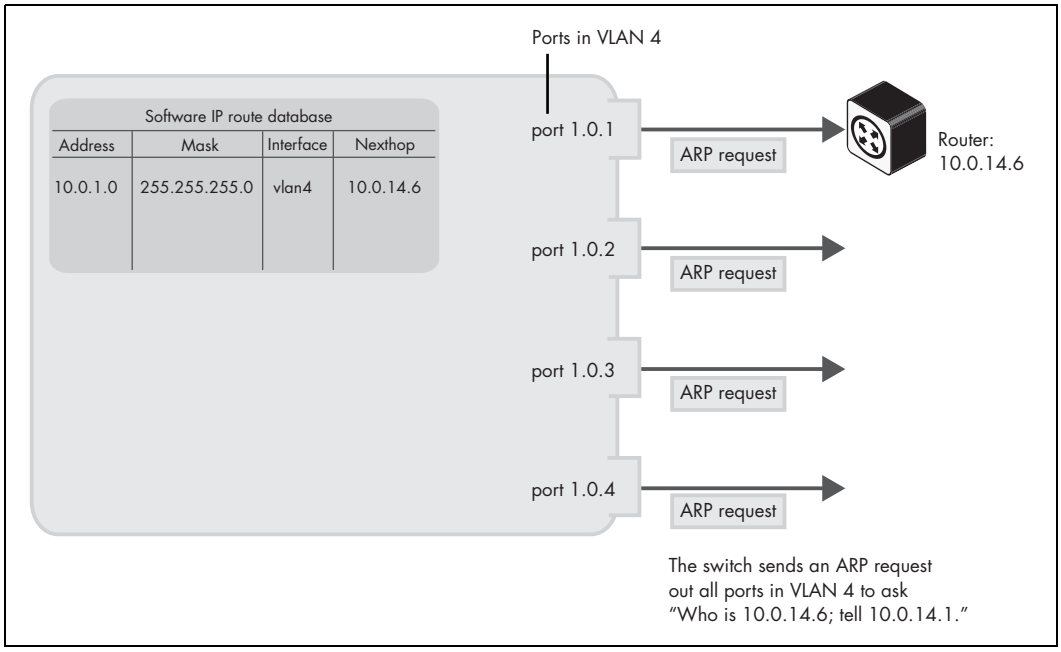
The hardware Layer3 unicast table is a hardware cache of the software routing database. In this document, we are not really going to discuss how routes get into the software routing database. We know that there are all sorts of protocols (OSPF, RIP, BGP, etc.) and filtering processes (Route maps, prefix lists, etc.) and rules (longest prefix match, metrics, preferences etc.) that interact in the process of managing the contents of the software routing database. A discussion of how all those factors interconnect would be quite complex, and is beyond the scope of this document.

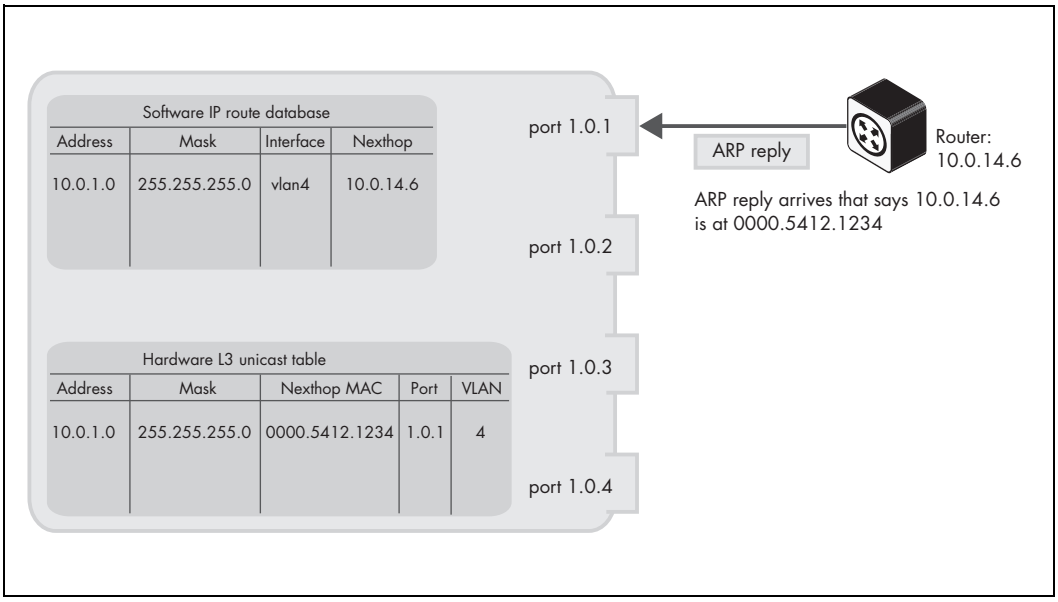
Instead, we are going to consider the question of what decisions and processes govern the action of taking software routing database entries into the hardware Layer 3 unicast table.

Entries in the IP forwarding information base contain some essential pieces of information: destination subnet address, subnet mask, nexthop IP address, egress IP interface. But, interestingly, these are **not** exactly the same pieces of information that reside in an entry in the hardware Layer 3 unicast table. So, it is not possible for the software to simply go through the software forwarding information base, and write each entry into the hardware Layer 3 unicast table. First, it is necessary to translate from **nexthop IP address/egress IP interface** to **nexthop MAC address/egress port**.

The key to this translation is, of course, the ARP protocol. The switch discovers the MAC address corresponding to a given IP by sending out an ARP request. When the ARP reply arrives, the switch discovers the MAC address of the host, **and** finds out which port it is connected to (the port on which the ARP reply arrived).

The following diagrams illustrate how the ARP protocol is used to find out the information required to translate an IP forwarding information base entry into a hardware Layer 3 unicast table entry.





Some switches aim to write IP forwarding information base entries into the hardware Layer 3 unicast table as soon as the route appears in the IP forwarding information base (having been learnt by OSPF, or statically entered from the command line, etc.). Other switches will not cache the route into the hardware table until they need to forward down the route. This is purely a matter of implementation choice; there is no standard that governs when to enter a route into the hardware Layer 3 table.

In summary, the process by which route entries get into the hardware Layer 3 unicast table is:

1. The route arrives into the software IP database (for example, because it is learnt by a routing protocol, or configured from the command line).
2. Either immediately, or when required, the switch translates the nexthop IP address/egress IP interface combination in the IP database entry into a nexthop MAC/egress port combination. The ARP protocol is the key to this translation process.
3. The switch then writes the route into the Layer 3 unicast table.

But, what about host entries? The process relating to these entries is slightly different. Initially, you might think: "well, all hosts within locally connected subnets are reached by a "connected" route entry in the IP forwarding information base, so why not just write that connected route into the hardware Layer 3 table, just like all other routes?" However, it is not quite as simple as that.

Consider, for a moment, what a route entry in the Layer 3 tells the switch. It says "all hosts within this subnet are reached by sending the packets out port X to nexthop MAC Y". But, for a locally connected subnet, that is not true—the different hosts in the subnet are reached via different ports, and the destination MAC address on the packets sent to each host must be the host's own unique MAC address. Just writing a single entry into the Layer 3 unicast table, to cover the whole of a locally connected subnet, simply will not do the job. Each host in the subnet needs its own unique entry, with its own MAC address and the port by which it is reached.

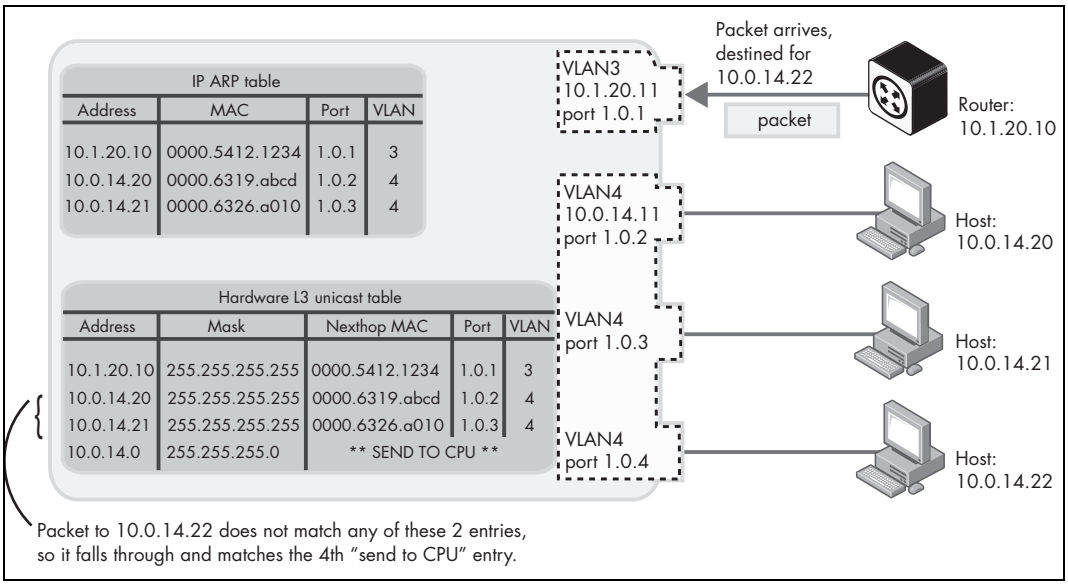
This is typically achieved by having an entry in the Layer 3 unicast table for the connected route, **but** for that entry to have no nexthop MAC or egress port. Instead, packets that match this entry are sent up the CPU to be dealt with.

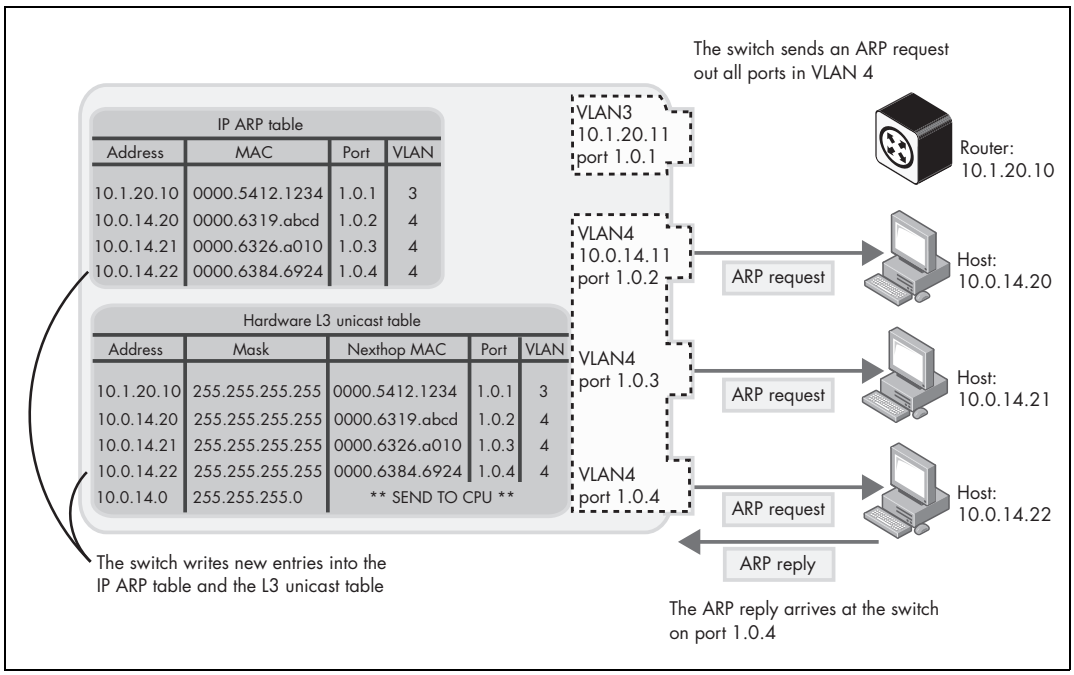
Here is an example of a couple of entries for directly connected subnets, that are marked to send packets to the CPU.

```
IPv4 Unicast Route Table: Driver Shadow
------------------------------------------------------------------------------
Index  IP address/prefLen NextHop          EnCap Dev Port Tr VID  Mtu   Ttl Cnt
------------------------------------------------------------------------------
1      192.168.1.250   24 ** trap to cpu **  -    -   -    -  -    1536  Dis  0
2      192.168.2.200   24 ** trap to cpu **  -    -   -    -  -    1536  Dis  0
```

Each time a packet arrives at the switch, destined to a previously unused host in a local subnet, the packet is sent to the CPU. The software then realises that it needs to do an ARP request for that host address. Upon receiving the ARP reply, the CPU can forward the packet, **and** it now has sufficient information to create a host-specific entry in the hardware Layer 3 unicast table for this host. It then creates the entry, so that subsequent packets destined to that IP address will match that host-specific entry, and will be hardware switched, rather than sent up to the CPU.

The following diagrams illustrate this process for a case where there are already host-specific entries for some addresses in a local subnet, and then a packet arrives destined to an address that is not already covered by a host-specific entry.





How entries are removed from the Layer 3 unicast table

There are plenty of different events that can cause an entry to be removed from the hardware Layer 3 unicast table. But when all these events are considered, it turns out they all fall into just 2 categories. They are events that cause an entry to be removed from either:

- the IP forwarding information base, or
- the IP ARP table

Because the content of the hardware Layer 3 unicast table is just a caching of the information that the software would use to make routing decisions, changes to the hardware table have to be driven by changes in the software tables. No mechanism should exist that causes deletions of entries in the hardware table independently of changes to the software tables.

Let's consider these two categories of events in turn.

Route removed from the IP forwarding information base

It is **imperative** for the content of the Layer 3 unicast table to remain in sync with the content of the IP forwarding information base. So, when a route is removed from the IP route database, if it has a corresponding entry in the hardware Layer 3 unicast table, then that hardware table entry must be removed immediately.

There are all sorts of reasons why a route will be removed from the IP forwarding information base, including the following:

- A static route is deleted by a command.
- A route is aged out by a routing protocol. Protocols like RIP and OSPF have aging mechanisms; if advertisements of a particular route are not seen for a specified period, the route is aged out.
- A route is explicitly withdrawn by a routing protocol. Most routing protocols have a mechanism by which one router can say to another "remove the following route(s) from your route database right now".
- A routing protocol is disabled. If the OSPF protocol, for instance, is administratively disabled, then all routes learnt by OSPF are immediately removed from the route database.
- An egress IP interface goes down.
- A routing protocol neighbor goes down. Protocols like OSPF and BGP regularly probe the UP state of the neighbors from which they are learning routes. If they detect that the neighbor is not responding to the probes, the neighbor is declared DOWN, and all routes learnt from that neighbor are immediately removed from the IP route database.
- A route is replaced by a better-cost route.
- A route is deleted due to a change in route-map configuration.

Entry removed from the IP ARP table

Initially, you might think "ARP table? I thought that the hardware Layer 3 unicast table was caching entries from the IP forwarding information base, so why would the removal of an entry from the ARP table result in the removal of an entry from the hardware Layer 3 unicast table?"

There are **two** reasons.

The **first** is that, as we saw above, the host entries in the hardware Layer 3 unicast table are effectively caching ARP table entries. The diagram illustrated the one-to-one correspondence between ARP table entries and host entries in the hardware Layer 3 unicast table.

If the ARP entry for 10.0.14.22 is removed from the ARP table, then the switch no longer knows the MAC address of that host, and the port via which to reach it. So, the hardware Layer 3 unicast table entry for 10.0.14.22 is no longer valid, and must be removed.

The **second** reason is the key role that ARP plays in the translation from IP forwarding information database entry to route entry in the hardware Layer 3 unicast table. The ARP entry for the nexthop router provides the key for translating the nexthop IP/egress interface to nexthop MAC/egress port. If the ARP entry for the nexthop IP is removed from the IP ARP table, then the switch no longer knows the MAC address of the nexthop device, and the port via which to reach it. So, any hardware Layer 3 unicast table entries using that information are longer valid, and must be removed.

There are a number of reasons why an entry may be removed from the IP ARP table, including the following:

- The entry ages out. As is typical with these tables, the ARP table has an aging mechanism, to remove entries that have not been used for a long time.
- The egress port goes down. We saw above that entries are removed the IP route database when a whole IP interface goes down. But, ARP entries are tied to a single port, so are removed when just that one port goes down.
- The entry is administratively deleted by a command.
- The IP address on the associated IP interface changes.
- The same ARP is learnt on a different port.

ECMP (Equal Cost MultiPath) routing

In a multiply connected network, it is possible to have more than one route to a particular destination subnet. In fact, networks are often deliberately designed to have alternate routes, to provide resiliency in case of the loss of one of the routes.

In some circumstances, a switch can have multiple "equal cost" routes to a certain subnet. "Equal cost" means that all the criteria used to decide on a "best" route are identical in all the routes, in other words, the administrative distance, prefix length, metric, etc. are all the same for all the routes.

In that case, it makes sense for the switch to use all those routes simultaneously, to take advantage of their whole aggregate bandwidth. This process of using multiple routes to the same destination is called ECMP (Equal Cost MultiPath) routing.

Some Layer 3 switching ASICs are able to support ECMP routing by holding multiple entries for the same destination subnet, and using a hashing algorithm to share different traffic flows across the different entries.

The output below shows two entries in the hardware Layer 3 unicast table, both for the same subnet 187.96.78.0/24.

```
awplus#show platform table ip

IPv4 Unicast Route Table: Driver Shadow
-------------------------------------------------------------------------------
Index  IP address/prefLen NextHop          EnCap Dev Port Tr VID  Mtu   Ttl Cnt
-------------------------------------------------------------------------------
...
6      187.96.78.0     24 00-00-cd-1d-7d-72  0    0   1    -  1    10240 Ena  2
7      187.96.78.0     24 00-00-cd-1d-7d-71  0    1   13   -  2    10240 Ena  2

...
-------------------------------------------------------------------------------
```

Blackhole routes

All the discussion up until now has talked about hardware Layer 3 table entries being used to decide where to **forward** packets to. But, it is possible to have an entry that specifically **drops** packets destined to a given subnet. Such an entry results from a "blackhole" route.

Blackhole routing is used for administrative and security reasons, typically when an administrator notices that there is rogue traffic on the network destined to a particular subnet (to which the switch should be expected to be forwarding traffic). The blackhole route can drop that traffic, and stop it propagating any further through the network.

```
awplus#show platform table ip

IPv4 Unicast Route Table: Driver Shadow
-------------------------------------------------------------------------------
Index  IP address/prefLen NextHop          EnCap Dev Port Tr VID  Mtu   Ttl Cnt
-------------------------------------------------------------------------------
...
4      173.72.8.0      24 ** drop **         -    -   -    -  -    1536  Dis  3

...
-------------------------------------------------------------------------------
```

Default route entry

If there is a default route in the IP forwarding information base, then that route can be written to the hardware Layer 3 unicast table, in the normal way, as for any other route.

But, if there is no default route in the IP forwarding information base, then what will the switch do with those packets that don't match any of the other entries in the Layer 3 unicast table? The answer is that if a packet does not match any entry in the table, it is dropped (not flooded).

Typically, though, for a packet that does not match any entry in the hardware table, you will want the CPU to have the last say on what happens with this packet—whether it is dropped silently, whether an ICMP unreachable is sent back to the originating host, whether the event is logged, etc. So, in the absence of an explicit default route, most switches will create a default route entry in the Layer 3 unicast table, with a destination of "send to CPU":

```
awplus#show platform table ip

IPv4 Unicast Route Table: Driver Shadow
-------------------------------------------------------------------------------
Index  IP address/prefLen NextHop          EnCap Dev Port Tr VID  Mtu   Ttl Cnt
-------------------------------------------------------------------------------
...
3      0.0.0.0         0  ** send to cpu **  -    -   -    -  -    1536  Dis  1

...
-------------------------------------------------------------------------------
```

Troubleshooting Layer 3 unicast forwarding issues

There can be many reasons for problems in IP routing. But, the majority of them relate to the management of the software IP forwarding information base. For example, problems where OSPF SPF calculations are incorrect, or BGP attributes are incorrectly interpreted, or RIP metrics are not set correctly are all problems that affect what entries exist in the software IP forwarding information base. Of course, a mistake in the contents of the IP forwarding information base will invariably lead to a corresponding mistake in the contents of the hardware Layer 3 unicast table. But the issue causing the mistake is not specific to the management of the Layer 3 unicast table.

In this section, we will just consider the debugging of problems directly related to the management of the hardware Layer 3 unicast table.

Failing to forward packets to the correct port

As with the other hardware forwarding tables, the switch will just do exactly what the contents of the Layer 3 unicast table tells it to do. If you find that packets are being Layer 3 switched to an incorrect port, or are being dropped when they should be forwarded, the first thing to do is to look at the contents of the hardware Layer 3 unicast table, to find the entry relating to the traffic that is being affected. There **will** be an entry that this traffic matches, even if it is just the **trap all else to the CPU** default entry.

Having found this entry, the next task is to see whether this entry is a correct caching of the current information of the IP forwarding information base and the ARP table. The IP forwarding information base and the ARP table are viewed as follows:

```
awplus#show ip route
Codes: C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       * - candidate default

S       187.96.78.0/24 [1/0] via 192.168.1.100, vlan1
                       [1/0] via 192.168.2.201, vlan2
C       192.168.1.0/24 is directly connected, vlan1
C       192.168.2.0/24 is directly connected, vlan2


awplus#show arp
 IP Address      MAC Address   Interface    Port      Type
192.168.1.100   0000.cd1d.7d72 vlan1      port1.0.1  dynamic
192.168.2.201   0000.cd1d.7d71 vlan2      port1.0.13 dynamic
```

When deciding whether the entry in the hardware Layer 3 table is a correct representation of the current contents of these software tables, it is important to take all factors into consideration:

- Just because a route appears in the IP forwarding information base does not mean that it **must** appear in the hardware Layer 3 table. For example, if the ARP entry for the nexthop address of the route has not yet been learnt, then the route would not necessarily have been written to the hardware table yet. So, it is important to check the IP ARP table to see if there is an entry for the nexthop IP.

- Be careful to check the netmasks on the routes you are comparing. It is possible that two routes have the same destination subnet IP, but different masks. If so, they are not identical routes.
- Similarly, check carefully that the MAC and egress port in an ARP entry exactly match the nexthop MAC and egress port for a route that uses that ARP entry.
- Remember that the Layer 3 table entries for directly connected routes must not have a nexthop MAC and egress port associated with them; they must be catch-all entries for their subnets that send packets the CPU if they match none of the host routes that have been generated for IP addresses in that subnet.

If you are certain that you have found a mismatch between the contents of the Layer 3 unicast table, and the software tables, then that is an important error that needs to be reported, along with all captured information to demonstrate the mismatch. It is very valuable, if possible, to also provide a description of how the switch reached this state—was it when a certain interface went down, was it when a certain RIP neighbor was added, etc. Errors in managing the content of the Layer 3 unicast table can cause significant network problems, but are often very subtle to track down. It is important for the reports of such errors to contain as much relevant information as possible.

Packets being software forwarded instead of hardware forwarded

This is a somewhat different symptom, but usually due to the same class of root cause.

If the entry for a particular destination is missing from the hardware Layer 3 table, then:

- The packets for that destination might match another forwarding entry in the hardware table, and be forwarded to an incorrect port by that entry (which is the problem discussed above),

  or

- A "send to CPU" catch-all default route entry might pick the packets up and send them to the CPU. But, if the IP forwarding information base actually contains a route to the packet's destination, the software will correctly route the packet.

It is this latter case that we are considering in this section.

The symptoms are:

- the packets are being correctly forwarded to their destination
- the CPU is busier than expected

The root cause, though, will be that an error in the management of the hardware Layer 3 unicast table means that the software route that is forwarding the packets is not written into the hardware table.

So, again, the correct approach to debugging is to identify the mismatch between the hardware table and the software tables, and report this fully.

The Layer 3 multicast table

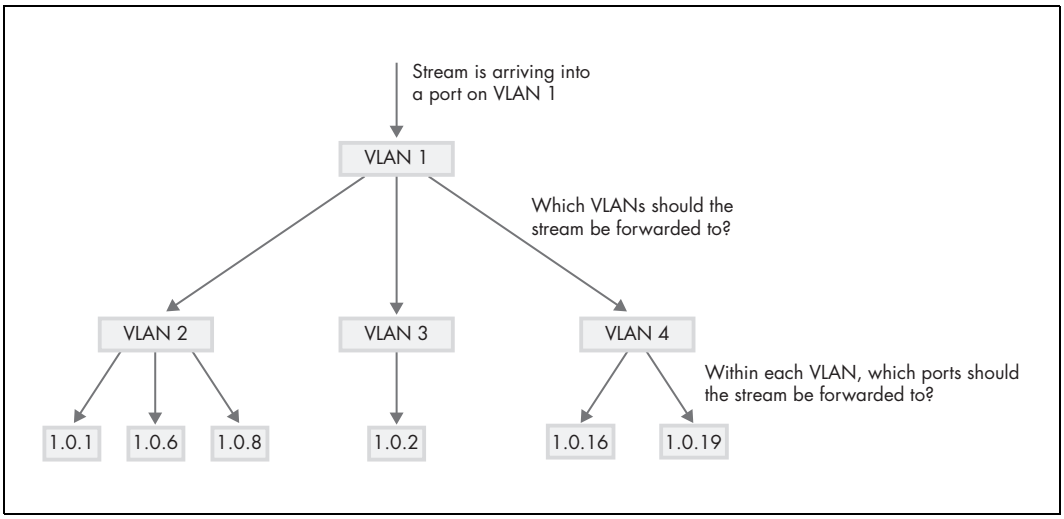
The purpose of the Layer 3 multicast table

The Layer 3 multicast table can be thought of in very much the same terms as the Layer 2 multicast table. It is just an extension of the concept to multiple VLANs. Where the Layer 2 multicast table controls the forwarding of multicast streams within a single VLAN, the Layer 3 table controls the forwarding of multicast streams from ports on one VLAN to ports on other VLANs.

Information that is stored in the Layer 3 multicast table

In fact, the Layer 3 multicast "table" is really a two-tier structure, because two questions need to be asked about the forwarding of the multicast streams:

- Which VLANs should the stream be forwarded to?
- Within each of those destination VLANs, which ports should the stream be forwarded to?



In fact, in many switch chip implementations, the second question is answered by an entry in the Layer 2 multicast table. The Layer 3 multicast table is effectively a list of destination VLANs for the streams, and associated with each VLAN, there is an index to an entry in the Layer 2 multicast table, listing which ports to send that stream to on that VLAN.

The following figures demonstrate this. For each VLAN in the VLAN list of the Layer 3 multicast table entry, the index points to an entry in the Layer 2 multicast table.

Layer 3 multicast table		
Group address	**List of VLANs**	
	VLAN	**L2 multi index**
238.56.124.10	vlan2	473
	vlan3	964
	vlan4	724

Layer 2 multicast table			
Index	**Multicast MAC**	**VLAN**	**Port list**
473	0100.5e38.7c-0a	2	1.0.1, 1.0.6, 1.0.8
.			
724	0100.5e38.7c0a	4	1.0.16, 1.0.19
.			
964	0100.5e38.7c0a	3	1.0.2

An example of the output that shows this information all together is shown below. The first two lines of the table entry relate to the incoming stream, then below that are two lines per downstream VLAN. The "Vidx" value that is highlighted is the index of the entry in the Layer 2 multicast table from which the per-VLAN information was drawn.

```
awplus#show platform table ipmulti

IPv4 Multicast Route Table
-------------------------------------------------------------------------------
Index  GroupIP         SourceIP       cmd ForceCos Prio DP   CntSet  CPU
  EnaTTL CheckRPF RPFIf MLL BottomEn VID Type Vidx TtlThr NoSrcVlan MTU

...
3      227.0.8.16      192.168.207.45  1      0      0   0      2     0(1)
   Ena       1     7     1      0
                                    1    5  4099   1      0       1536
   NumPorts = 1  CpuMem = N  Local Ports = 1.0.21

                          2      0     2    5  4101   1      0       1536
   NumPorts = 1  CpuMem = N  Local Ports = 1.0.13
```

How entries get into the Layer 3 multicast table

For an entry to get into the Layer 3 multicast table, an Layer 3 multicasting protocol like PIM or DVMRP needs to be enabled. Without such a protocol enabled, the switch will not even consider Layer 3 switching multicast traffic. What is more, the Layer 3 multicasting protocol needs to be configured on **all** the IP interfaces that are to be involved in Layer 3 multicast switching.

The act of enabling an Layer 3 multicast protocol is quite straightforward. The following figure is an extract of a configuration file:

```
ip multicast-routing
interface vlan1
 ip address 192.168.1.250/24
 ip igmp
 ip pim sparse-mode
!
interface vlan2
 ip address 192.168.2.200/24
 ip igmp
 ip pim sparse-mode
!
interface vlan3
 ip address 192.168.207.1/24
 ip igmp
 ip pim sparse-mode
!
...
```

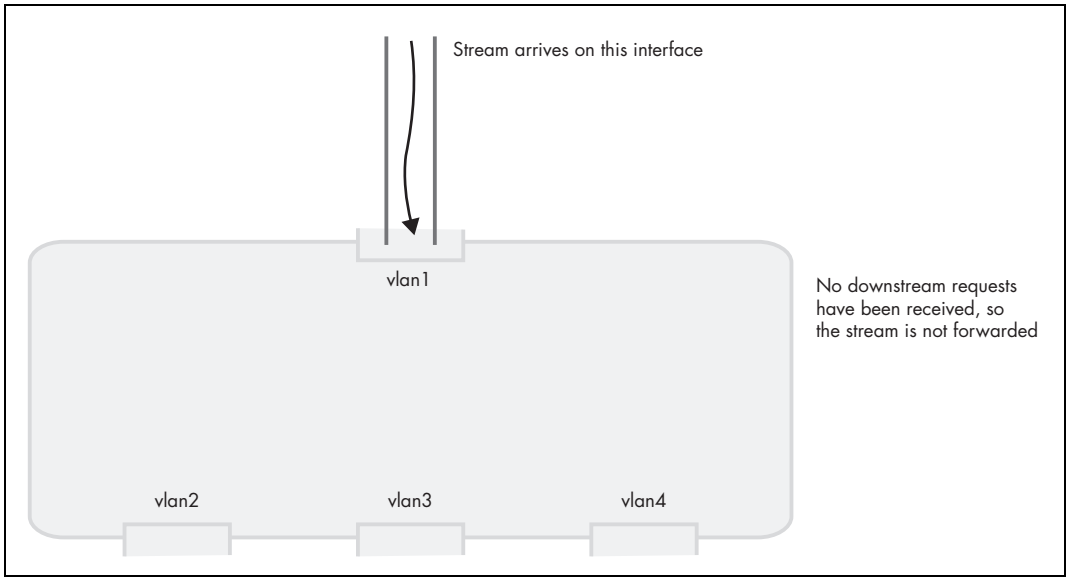
The details of the interaction of the Layer 3 multicasting protocol with other switches are, however, quite complex, and are beyond the scope of this document. For the purposes of the current discussion, we will simply assume that the Layer 3 multicasting protocol can receive notifications from other switches, which ask the switch to send a stream to them, or ask the switch to stop sending a stream. The details of how these notifications are exchanged will not be considered here.

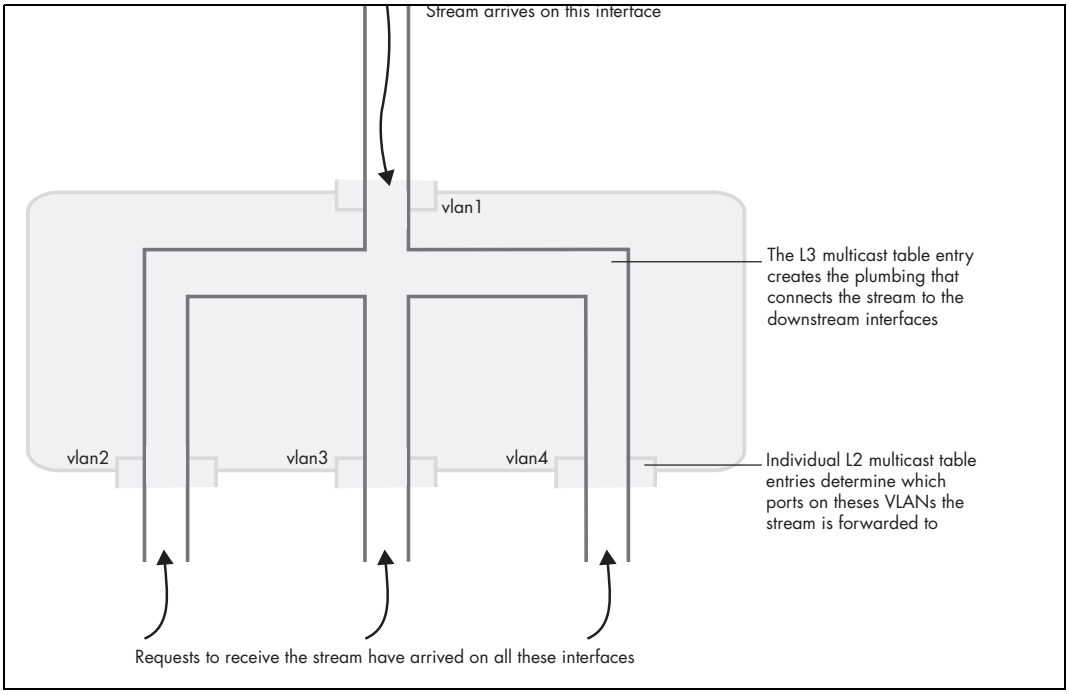
Once an Layer 3 multicast protocol has been enabled on the switch, two things need to occur in order for an Layer 3 multicast entry to be created:

- The switch needs to detect that a multicast stream is arriving on an interface on which an Layer 3 multicast protocol has been enabled
- The switch needs to have received a request to receive that stream, on a different VLAN interface, on which an Layer 3 multicast protocol is also enabled. The request could be an IGMP report from an end host, or it could be a notification from another switch running an Layer 3 multicast protocol.

When these two events have occurred, the switch effectively "plumbs in" a connection between the interface receiving the stream and the interface(s) to which the stream is to be forwarded; then the stream can flow on through, and be forwarded on the relevant ports of those downstream interfaces.

The junction point in this plumbing is the Layer 3 multicast table entry.





How entries are removed from the Layer 3 multicast table

Just as there are two events that are precursors to an entry being put into the Layer 3 multicast table, there are two major ways that entries can be removed from the Layer 3 multicast table: either the stream stops arriving, or downstream receivers stop wanting to receive the stream.

The stream stops arriving

If the stream is no longer arriving at the switch, then there is no point in having an Layer 3 multicast table entry in place to tell the switch which VLANs to forward the stream to.

You might say "yes, but having created the table entry, why not leave it in place just in case that stream starts arriving again?" Well, that would be okay if the Layer 3 multicast table was very large. But, typically the Layer 3 multicast table can hold 1000 - 2000 entries. If we kept the entries for every destination group address the switch had ever seen, then there would be a danger of filling the table. It is prudent to remove entries as soon as they are not needed, to conserve space in the table.

You might also wonder how the CPU knows when a stream has stopped arriving. If the stream is being hardware switched, the CPU will never see those packets, and so, will never know when they have stopped. That is a very good point, and one of the trickier aspects of designing the software that manages the Layer 3 multicast table. To handle this, at regular intervals (for example, once per minute) the software alters each Layer 3 multicast table entry in such a way that it will copy its stream to the CPU. If the CPU then starts receiving packets for that stream, it knows the stream is still arriving, and it keeps that entry. But, if the CPU does not then start receiving packets for that stream, it knows that the stream has stopped arriving, and it removes that entry.

Downstream receiver(s) stop listening

In the discussion of the L2 multicast table (page 117 onwards), we looked at the various events that could cause a port to be removed from the port list in an L2 multicast table entry, and also how an L2 multicast table entry could be removed altogether.

Once an L2 multicast table entry is completely removed for Group X on VLAN Y, that indicates to the software that the Layer 3 multicast table no longer needs to include VLAN Y in the list of VLANs to which it forwards Group X. So, the software would, at that point, remove VLAN Y from the VLAN list in the Layer 3 multicast table entry for Group X.

Of course, if this were the only remaining VLAN listed in the Layer 3 multicast table entry for Group X, then that whole Layer 3 multicast table entry might as well be removed.

It is worth remembering that with Layer 3 multicasting, IGMP is not the only protocol that is being used to indicate the presence of downstream listeners for a given group. There can also be other downstream Layer 3 multicast switches, which can be sending notifications via the Layer 3 multicasting protocol. But the effect on the Layer 3 multicast table is the same, whether the "stop sending the stream to me" notification arrived via an IGMP leave message or via a PIM Prune.

There are some other possible events to bear in mind, that can result in a VLAN being removed from an Layer 3 multicast entry:

- If the Layer 3 multicasting protocol is administratively disabled on a certain VLAN, then that VLAN is no longer participating in Layer 3 multicasting on that switch. That VLAN must be immediately removed from any Layer 3 multicast table entries.
- If the neighbor relationship with a downstream Layer 3 multicast switch goes down, then that is treated as equivalent to that neighbor sending "stop sending me the stream" notifications for **all** the streams that it was receiving.

Troubleshooting Layer 3 multicast table issues

The most common Layer 3 multicast problem is that of a stream not being forwarded when you think it should be.

There are a lot of factors to consider when debugging this problem, but if you work through them methodically, you will invariably be able to narrow down where the problem is.

The steps to work through are:

1. Have you confirmed that the video stream definitely gets delivered to the upstream port it should be arriving on?

   If the stream is arriving, and the software is operating correctly, then the software should indicate that it is seeing the stream.

```
awplus#sh ip pim sparse-mode mroute
IP Multicast Routing Table

(*,*,RP) Entries: 0
(*,G) Entries: 0
(S,G) Entries: 1
(S,G,rpt) Entries: 1
FCR Entries: 0

(192.168.207.45, 227.0.8.16) - this line is the source and dest. IPs of
the stream
RPF nbr: 0.0.0.0
RPF idx: None
SPT bit: 1
Upstream State: NOT JOINED
 Local      ..............................
 Joined     ..............................
 Asserted   ..............................
 Outgoing   ..............................

(192.168.207.45, 227.0.8.16, rpt)
RP: 192.168.2.200
RPF nbr: 0.0.0.0
RPF idx: None
Upstream State: RPT NOT JOINED
 Local      ..............................
 Pruned     ..............................
 Outgoing   ..............................
```

Also, once the software has seen an incoming stream, it will create an entry for this stream in the Layer 3 multicast table, even if it does not believe that it needs to forward this stream anywhere:

```
awplus#show platform table ipmulti

IPv4 Multicast Route Table

Index  GroupIP         SourceIP       cmd ForceCos Prio DP   CntSet  CPU
  EnaTTL CheckRPF RPFIf MLL BottomEn VID Type Vidx TtlThr NoSrcVlan MTU

1      227.0.8.16      192.168.207.45  1      0      0   0      2     0(1)
   Ena        1     7     *** no downstream interfaces ***
```

However, the absence of this indication from the software does not necessarily mean that the stream is not arriving. It could be that the stream is arriving, but is tagged with a VLANID different to the VLAN of the upstream port. Or it could be that the stream is failing the RPF test (see below). If there is no indication from the software, the best way to confirm that the stream is not arriving, is to mirror that upstream port to another port, and sniff (network analyse) on that other port.

If network analysis is not possible, then you can at least confirm whether **some** multicast stream is arriving on the port by looking for incrementing **multicast receive** counters on the port. If that counter is not incrementing, then there is definitely no multicast stream arriving at the port. The counter to look for is highlighted below:

```
awplus#show platform table port port1.0.7 count

 Switch Port Counters
------------------------------------------------------------------------

Port 1.0.7    Ethernet MAC counters:
 Combined receive/transmit packets by size (octets) counters:
  64                          8986623 512 - 1023                          0
  65 - 127                         15 1024 - MaxPktSz                     0
  128 - 255                         0
  256 - 511                         0

 General Counters:
 Receive                            Transmit
  Octets                    575125312 Octets                         19598
  Pkts                        8986334 Pkts                             304
  CRCErrors                         0
  MulticastPkts               8986329 MulticastPkts                    303
  BroadcastPkts                     4 BroadcastPkts                      0
  FlowCtrlFrms                      0 FlowCtrlFrms                       0
  OversizePkts                      0
  Fragments                         0
  Jabbers                           0
  UpsupportOpcode                   0
  UndersizePkts                     0
                                      Collisions                         0
                                      LateCollisions                     0
                                      ExcessivCollsns                    0

 Miscellaneous Counters:
  MAC TxErr                         0
  MAC RxErr                         1
  Drop Events                       0
------------------------------------------------------------------------
```

2. If the stream is not arriving at the switch, and PIM-SM is being used as the multicast routing protocol, the problem could be that the switch has not learnt a Rendezvous Point from which to request the stream. When a switch wishes to receive a stream, it does not necessarily know where that stream is located. Hence, in the PIM-SM protocol, certain routers in the network are designated as the 'known points' from which to request stream for given multicast groups. These routers are called Rendezvous Points. The mapping of which routers are the Rendezvous Points for which blocks of multicast addresses is advertised through the network by a Bootstrap Router. A configuration error, or a failure to receive the information from the Bootstrap Router, could leave a switch in a state where it simply does not know which router to contact to request the multicast stream in question.

   To check whether the switch has learnt an address of a Rendezvous Point for the multicast stream you are investigating, use the command **show ip pim sparse-mode rp mapping**.

3. In order to avoid network loops, Layer 3 multicasting protocols perform a Reverse-path forwarding (RPF) test on incoming multicast streams. The effect of the test is to look at the source IP of the incoming stream, and check whether the route back to that source IP address is out the interface on which the stream is arriving. If this test fails, the stream will not be accepted. Even if the stream is arriving on the port, the software could be refusing to register this stream for this reason.

4. Check the TTL value in the packets in the stream. For multicast packets to be successfully Layer 3 forwarded, they must have a TTL value of greater than 1. Probably the only way to work out the TTL value in the packets is to sniff them.
5. Check whether the IGMP reports or Layer 3 multicast notifications have been received on the downstream interface. The discussion of the L2 multicast table describes what to look for in this regard (page 117 onwards).
6. If the software has registered that the stream is arriving, and it has registered the downstream requests, then it **should** have connected these pieces of information together. This is best seen using the command **show ip mroute**, which will show the incoming and outgoing interfaces for a given multicast stream.

```
awplus#show ip mroute

IP Multicast Routing Table
Flags: I - Immediate Stat, T - Timed Stat, F - Forwarder installed
Timers: Uptime/Stat Expiry
Interface State: Interface (TTL)

(192.168.207.45, 227.0.8.16), uptime 00:03:48, stat expires 00:02:06
Owner PIM-SM, Flags: TF
  Incoming interface: vlan7
  Outgoing interface list:
    vlan2 (1)
    vlan1 (1)
```

The command **show ip pim sparse-mode mroute** can also be used to see that PIM has joined the incoming stream to some outgoing interfaces.

But, it is not so clearly displayed in this output. It is indicated by there being "o" characters in the line "Outgoing o..o...."

```
awplus#show ip pim sparse-mode mroute
IP Multicast Routing Table

(*,*,RP) Entries: 0
(*,G) Entries: 1
(S,G) Entries: 1
(S,G,rpt) Entries: 1
FCR Entries: 0

(*, 227.0.8.16)
RP: 192.168.2.200
RPF nbr: 0.0.0.0
RPF idx: None
Upstream State: JOINED
 Local     i..i..........................
 Joined    ..............................
 Asserted  ..............................
FCR:

(192.168.207.45, 227.0.8.16)
RPF nbr: 0.0.0.0
RPF idx: None
SPT bit: 1
Upstream State: JOINED
 Local     ..............................
 Joined    ..............................
 Asserted  ..............................
 Outgoing  o..o..........................

(192.168.207.45, 227.0.8.16, rpt)
RP: 192.168.2.200
RPF nbr: 0.0.0.0
RPF idx: None
Upstream State: NOT PRUNED
 Local     ..............................
 Pruned    ..............................
 Outgoing  o..o..........................
```

Also, if the software has connected the arriving stream to the downstream requests, then it should have put entries in the Layer 3 multicast table.

One general piece of advice with regard to Layer 3 multicasting: the best Layer 3 multicast protocol to use, if at all possible, is PIM-DM. It does not have PIM-SM's complications of boot-strap router and rendezvous point elections. Also, PIM-DM works on the basis of "I will forward the multicast unless someone tells me not to", whereas PIM-SM works on the basis of "I will not forward the multicast until someone asks me to". Unless the situation absolutely requires PIM-SM, choose PIM-DM.

Chapter 9 | Virtual Chassis Stacking

Introduction

Virtual Chassis Stacking (VCStack™) is the name given to two or more Allied Telesis switches that are configured to operate as a single switch. From a configuration and management point of view, it is as though the switches are just one device with a seamless transition from the ports of one stack member to the ports of the nextWhen configuring a VCStack, there are no limitations on how the ports of one stack member can interact with the ports of another stack member–they can belong to VLANs together, they can belong to port aggregations together, they can mirror to each other, and port-range configuration can span multiple stack members. The stack member ports truly operate as though they all belong to one virtual switch.

The same applies with Layer 2 and Layer 3 switching (both unicast and multicast). The stack operates as a single device and is not perceived by end users, or the traffic itself, to be any more than a single network node.

There are some limitations to the seamlessness of virtual chassis stacking; for example, the file systems on the individual stack members remain discrete.

List of terms

Stack member

An individual switch that is part of a VCStack.

VCStack or **stack**

A group of two or more switches operating as a single switch.

Stack master or **active master**

The switch that manages the stack.

Resiliency Link

An extra, out-of-band, data link between stack members. In the event of loss of communication across the stacking connection, the stack members can determine the status of other members via communication on the resiliency link. This assists the stack members in deciding the correct course of action when communication on the stack is lost.

This chapter explains the physical creation of a VCStack, the configuration required on stack members, and how to monitor the operation of the VCStack. It also provides an understanding of how the stack behaves when a stack member stops responding.

Connecting switches into a stack

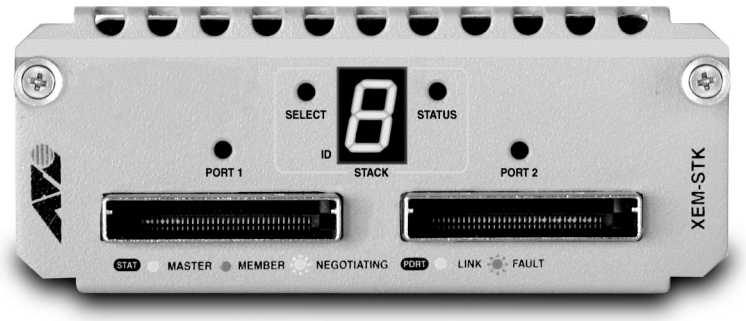
The proprietary high-speed communication protocol that is used over the stacking links requires multiple twisted pairs and a high level of shielding. This means that to stack x-Series switches, specialised cables and connections are required.

The types of cables and connections available are dependant on the type of x-Series switches you are stacking. The stacking options are:

- "Front-port stacking using XEM-STKs on x900 Series switches" on page 152
- "High-speed stacking on SwitchBlade x908 switches" on page 153
- "AT-StackXG slide-in modules on x600 Series switches" on page 154

Front-port stacking using XEM-STKs on x900 Series switches

You can fit the XEM bays on x900 Series switches with a specialised stacking XEM called the XEM-STK.

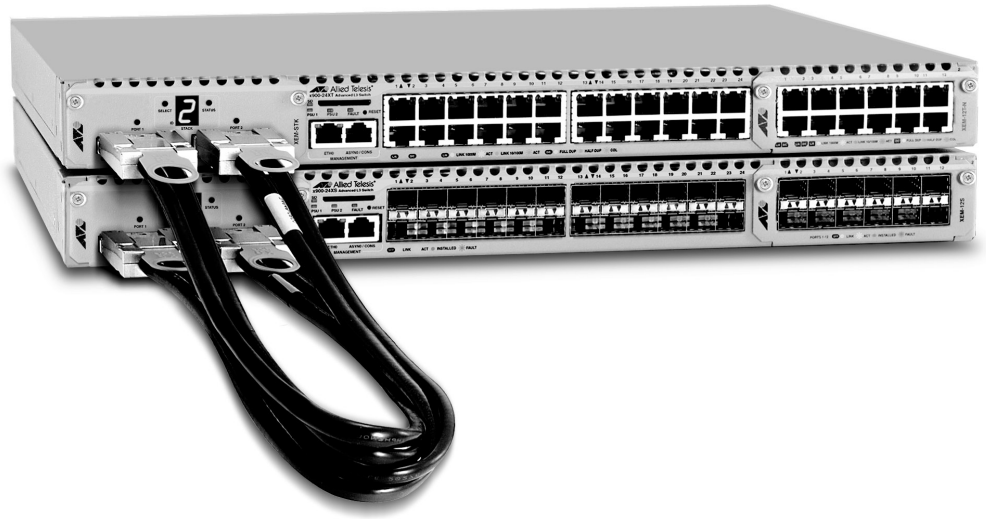


The LED number on the front of the XEM-STK shows the stack ID of the switch that the XEM-STK is installed in. See the section "Identifying each switch with stack IDs" on page 159 for more information about stack IDs.

The specific cable type that connects these XEMs are purchased individually as either 0.5 or 2 metre long cables. The product codes are:

- AT-XEM-STK-CBL0.5
- AT-XEM-STK-CBL2.0

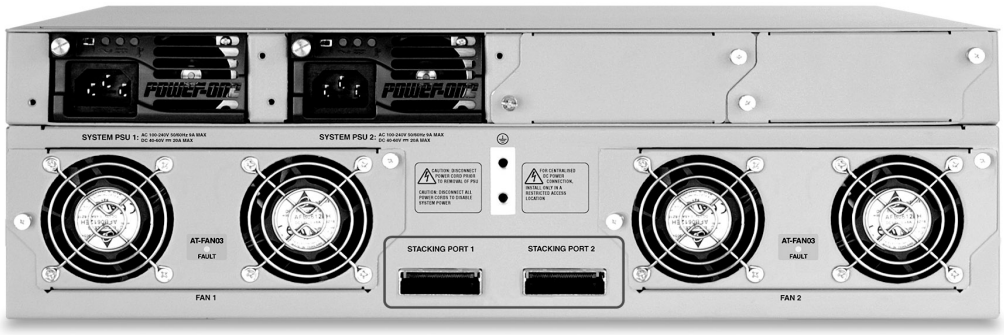
With these XEMs and cables, you can create a stack of up to two x900 Series switches.



Although you can use XEM-STKs with x908 switches, we do not recommend doing so, as high-speed stacking performs significantly better on SwitchBlade® x908 switches. Front-port stacking also uses 2 XEM slots which you could use for other XEM modules.

High-speed stacking on SwitchBlade x908 switches

On the rear of the SwitchBlade x908 chassis, there is a pair of fixed stacking ports.



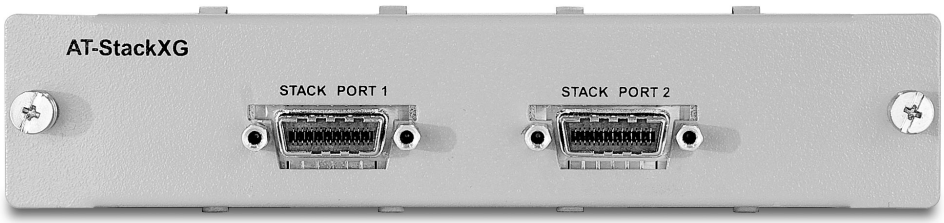
High-Speed stacking requires a specific cable type, different than the cable required for the XEM-STK. The product code is AT-HS-STK-CBL1.0.

You can stack two SwitchBlade x908 switches together using these ports and cables.

Note that the cables are crossed over–port 1 of the top switch is connected to port 2 of the bottom switch, and vice versa.

AT-StackXG slide-in modules on x600 Series switches

On the rear of the x600 chassis you can insert a slide-in module called the AT-StackXG.



The specific cable type that connects the AT-StackXG are purchased as either 0.5 or 1 metre long cables. The product codes are:

- AT-STACKXG/0.5
- AT-STACKXG/1

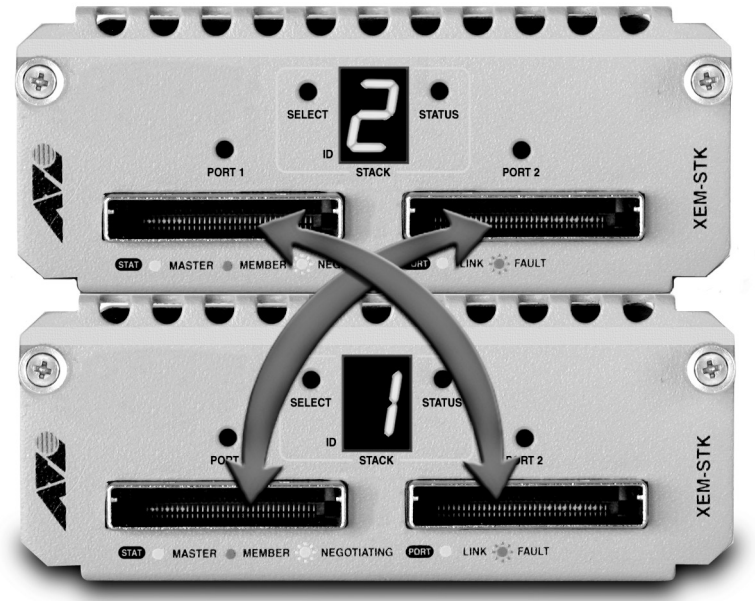
You can stack up to four x600 switches using the AT-StackXG.

Like the other stacking methods, the connections are crossed-over–port 1 on one switch is connected to port 2 on its neighbor and the switches are connected in a ring.

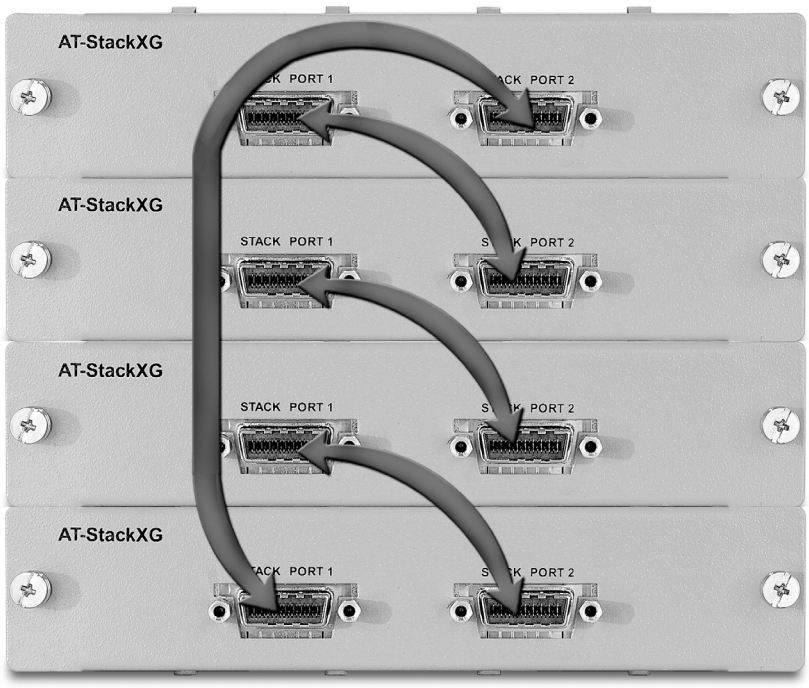
Connecting the cables to the switches

Stacked switches are connected in a ring, with port 1 of one switch connected to port 2 of the next switch. The last switch in the stack then connects to the first switch using port 2 of the first switch and port 1 of the last switch.

In two switch stacks, this means that the connection consists of just a crossed pair of cables. You may also use only one cable to connect the switches, but you will halve the bandwidth between stack members.



The following figure shows how to connect a 4 switch stack of x600 Series switches. Once again, you could connect the switches without one of the cables, but you would halve the bandwidth between stack members.



Restrictions on stacking combinations, connections, and sizes

There are some restrictions to what products and connections you can combine in a single stack. In general:

- different switch families cannot be stacked together
- you cannot combine different stacking methods or cables

x900 Series restrictions

compatible products	x900 Series switches can only stack with other x900 switches. Within the x900 Series: • x900-12XT/S switches can only stack with x900-12XT/S switches • x900-24XS and x900-24XT can stack together
maximum stack size	2 switches.
cable type	AT-XEM-STK-CBL0.5 (0.5 metre length) or AT-XEM-STK-CBL2.0 (2 metre length).

SwitchBlade x908 restrictions

compatible products	A SwitchBlade x908 switch can only stack with another SwitchBlade® x908 switch.
maximum stack size	2 switches.
cable type	For high-speed stacking: AT-HS-STK-CBL1.0.

It does not matter what type of XEMs are installed in each SwitchBlade x908, nor where each XEM is installed in the switch.

The stack cannot contain a mixture of high-speed stacking and XEM stacking. This means that it is not possible to stack two x908 switches together with high-speed stacking, and then connect one of those x908 switches to another x908 using XEM stacking.

x600 Series restrictions

compatible products	x600 Series switches can only stack with other x600 switches. Within the x600 Series, you can stack any model with another model. This means that the x600-24Ts, x600-24Ts/XP, x600-48Ts, and x600-48Ts/XP switches can all stack together without restriction.
maximum stack size	4 switches.
cable type	AT-STACKXG/0.5 (0.5 metre length) or AT-STACKXG/1 (1 metre length).

How the stack communicates

The stack communicates through the stacking cables. You can also configure a resiliency link between stack members, which is used when a failover event occurs on the stack. See the section "Failover and recovery" on page 172 for more information.

One switch controls all switch management on a stack. Which stack member does this is discussed in "Roles of each switch in a stack" on page 158.

The software version, startup configuration, and running configuration are exactly the same on each member of a stack. For information on how the stack synchronises the files, see the section "Software and configuration file synchronisation" on page 170.

The internal communication between stack members is carried out using IP packets sent over the stacking links. This stack management traffic is tagged with a specific VLAN ID and uses IP addresses in a specified subnet. The default is:

- VLAN 4094
- Subnet 192.168.255.0/28

The management traffic is queued to egress queue 7 on the stack link. The section "Configuring the stack" on page 168 discusses the minor restrictions necessary when configuring VLANs, IP subnets, and QoS on the stack.

Roles of each switch in a stack

Each switch in a stack acts in one role at any time. This is either as a **backup member** (also called **stack member**) or a **stack master** (normally as the **active master**). The stack members are controlled by the stack master. All configuration, including updating the switch software, is set on the stack members by the stack master.

The stack master performs a number of tasks that a stack member does not perform:

- it controls all switch management activity.
- it synchronises boot release and configuration files with stack members.
- all routing protocol packets are processed by the stack master. The stack master then transfers any requisite table updates to the stack members.

The stack master also handles communication on behalf of the stack:

- when you telnet or SSH to the stack, you connect to a process running on the stack master.
- when you connect to the asyn port on a stack member, this automatically creates an SSH session to the master. This connection will behave as if you were connected to the asyn port on the master.
- the stack master handles SNMP interactions. It gathers MIB statistics from the stack members, and delivers these statistics in response to SNMP Get requests.

You can still execute some specific commands and manage files on an individual switch in the stack. See the section "Executing commands and managing files on a specific stack member" on page 177 for more information.

When a VCStack is operating normally, the stack master acts as the **active master**. However during some failover and resiliency situations, when a stack member cannot contact the active master, it may act as either a **disabled master**, or **fallback master**. See the section "Failover and recovery" on page 172 for more information about the differences between these stack master roles.

Selecting the active master

The stack members negotiate among themselves as to which switch will become the **active** master. The election of the active master is based on two criteria:

- each stack member's priority setting
- each stack member's MAC address

For each of these criteria, a **lower number** indicates a higher priority. The active master is the switch with the lowest priority value. If multiple switches share the lowest priority value, then the switch with the **lowest MAC address** becomes the active master.

Manually changing the active master

You can choose a specific switch to be active master by changing its priority value. By default, a switch has a stack priority value of 128. The command to change a switch's priority value for stacking is:

```
stack <switch stack ID> priority <0-255>
```

The stack ID is a unique identifier that is assigned to each switch in the stack. See the section "Identifying each switch with stack IDs" on page 159 for more information.

If a switch is already part of a stack, you can still set the priority value on each switch in the stack through the active master. However, if you set the priority value on a stack member lower than the active master's priority value, the active master does not immediately relinquish its role as master. The stack member with the lowest priority will take over as active master only if the current active master leaves the stack (this includes any reboot by the stack master).

If a switch has not yet joined a stack, you can still use the **stack priority** command to change the priority value before connecting it to the stack. However, even if the new stack member has a lower priority value than the active master, it will not take over as active master unless the current active master is removed or rebooted.

Identifying each switch with stack IDs

Each switch in a stack has an ID number, which can be an integer between 1 and 8. The default on each switch is a stack ID of 1. The ID number of a stack member is an important identifier. All commands that are port or switch specific need to use the stack ID to identify which stack member the commands apply to.

The stack ID is also used to manage specific stack members. When you telnet or SSH to the stack, your login session terminates on the active master. Similarly, if you connect to the console port of any stack member, your console login session is sent through to the active master. So, the active master switch is the automatic point of contact for any management session on the stack. If you want to examine something that physically resides on one of the other stack members, such as files in a stack member's flash, or voltage levels on a stack member's power supply, then you can do this by specifying the stack ID of the stack member in commands.

The stack IDs are used frequently in the stack configuration scripts. For example, any command in the configuration script that refers to a physical port will include a stack ID in the port number. The section "Port numbering" on page 168 explains the port numbering scheme used in stacks.

For these reasons, the stack IDs on each switch within a stack are unique and a switch's stack ID normally does not change without user intervention. The only exception to this is when a new switch is connected to a stack and the switch has the same stack ID number as another stack member–the new switch's ID will be automatically renumbered.

There is no connection between stack ID and active master status. The active master switch does not have to be the switch with a Stack ID of 1.

Displaying the stack IDs

To see the stack ID on a switch before it is connected to a stack, use the command:

```
show stack
```

The output will show the current stack ID and any pending change to the stack ID.

```
awplus#show stack
Virtual Chassis Stacking summary information
ID  Pending ID  MAC address        Priority  Role
1   2           00-00-cd-28-bf-e5  128       Active Master
```

You can also use this command to see the stack numbers on a two-switch stack. To match the physical switch with the right stack ID number, look for the **active master** LED on the front panel. Then use the **show stack** command to show all members of the stack. You can match the LED status to the **show** command output.

```
x900#show stack
Virtual Chassis Stacking summary information
ID  Pending ID  MAC address        Priority  Role
1   -           00-00-cd-28-07-c0  0         Active Master
2   -           00-00-cd-28-bf-e5  128       Backup Member
```

The **show stack** command is available on all stacking switches running the AlliedWare Plus operating system. On the front of the XEM-STK there is an indicator that displays the stack ID of the switch that the XEM is installed in.

On x600 Series switches, you can use the command:

```
show stack indicator <1-8>|all [time <1-500>]
```

This causes the **master** LED on the switch to flash in a sequence which indicates the stack ID number. For example, on a four-switch stack with the stack IDs 1, 2, 3, and 4, the switch with stack ID:

- 1 will flash on and off without pausing `**********`
- 2 will flash twice then pause `** ** ** ** **`
- 3 will flash three times then pause `***  ***  ***`
- 4 will flash four times then pause `**** **** ****`

You can extend the time that the **master** LEDs will flash by up to 500 seconds to give you more time to reach the stack's location, by using the optional **time** parameter with the command. By default the LEDs will flash for 5 seconds.

Assigning stack IDs

Stack IDs can be assigned in a number of ways. We recommend only assigning stack IDs when you set up the stack, as a change to stack ID numbers is not automatically reflected in configuration scripts.

You can assign stack IDs to switches before they are part of a stack:

- "Manual assignment on a switch before stacking" on page 161

To assign stack IDs once the stack is created, you can use the following methods:

- "Automatic assignment as a switch joins the stack" on page 162
- "Manual renumbering of a switch after stacking" on page 162
- "Cascade renumbering of the stack" on page 162
- "Renumbering the whole stack using the XEM Select button" on page 163

Manual assignment on a switch before stacking

You can manually assign the stack IDs on switches before you stack them together.

If your switch has never had a stack ID assigned to it, it will have the stack ID of 1. You can assign it a new stack ID with the command:

```
stack (config)#stack 1 renumber <1-8>
```

If the switch has previously been in a stack and was assigned a stack ID, it keeps that stack ID with it, even if it is removed from the stack. The stack ID has become an inherent property of the switch. If you wish to renumber it you will need to specify the current stack ID. For example to renumber the stack ID from 3 to 5, use the command:

```
stack (config)#stack 3 renumber 5
```

The stack ID renumbering does not take effect until the switch is rebooted, so you will receive the following warning as shown in the next figure.

```
awplus(config)#stack 1 renumber 2
Warning: the new ID will not become effective until the stack-member
reboots.
Warning: the boot configuration may now be invalid.
```

Until the reboot occurs, the new stack ID value is noted as the 'pending' stack ID, as shown in the next figure.

```
awplus#show stack
Virtual Chassis Stacking summary information
ID  Pending ID  MAC address        Priority  Role
1   2           0000.cd28.bfe5     128       Active Master
```

Automatic assignment as a switch joins the stack

When a stack is established, the stack will automatically assign a new stack ID to a switch if it has the same stack ID as another member. This means that you can create a stack without previously changing the stack ID numbers from the default of 1. The stack master will be assigned stack ID 1, and the other switches will be automatically assigned other IDs.

Manual renumbering of a switch after stacking

If you want to manually renumber the switches when they are already part of a stack, use the command:

```
stack <1-8> renumber <1-8>
```

You can issue this command from the active master to renumber any switch in the stack. To avoid duplicate IDs, a warning message appears if you assign to a stack member the same ID that is currently assigned to another stack member. However, you can continue to renumber the stack member IDs and fix potential ID duplications. Once you have removed any duplicate IDs, you can reboot the switches with ID changes to implement your changes.

If you do not remove the duplications, then when the stack reboots, the switch with the best stack priority (the lowest priority value) is allocated this ID, and the competing switch is automatically assigned another ID.

Cascade renumbering of the stack

For larger stacks, it is useful to have all the switches numbered in sequence, based on the order of their stack connections. The command that can achieve this is:

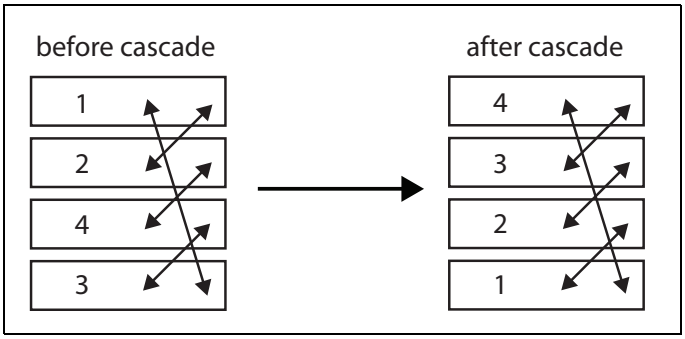
```
stack <1-8> renumber cascade [<1-8>]
```

For example, you can enter the command:

```
stack 3 renumber cascade 1
```

This starts the stack numbering on the switch that currently has Stack ID 3 and gives that switch Stack ID 1. The other switches in the stack are renumbered sequentially based on their connection to the switch that now has the stack ID 1.

The following figure shows the change from this command.



If the second stack ID parameter is not supplied in the command, then the numbering begins from 1. When the numbering process hits the maximum existing stack ID value, it assigns the value 1 to the next switch in the stack.

We recommend using this command to ensures the switches are sequentially numbered, if you did not manually assign the switches with stack IDs before connecting them to the stack. Use the command straight after the stack is first connected.

Renumbering the whole stack using the XEM Select button

On switches connected with XEM-STKs, you can use the **Select** button on the front of the XEM-STK. This achieves the same effect as the **renumber cascade** command. When you press the **Select** button on a XEM-STK, the switch with that XEM-STK is assigned stack ID1, and the other members of the stack are numbered sequentially from there.

Caution with stack ID renumbering

Stack IDs are critical in identifying each physical switch in the port numbering and configuration commands. However, the stack's configuration script is not altered when stack IDs are renumbered. This means that you may lose the connection between configuration commands and the physical switches, and commands could end up being applied to a different switch.

For example, if switch A in the stack currently has stack ID 2, then any commands in the configuration script that refer to stack ID 2 are applied to switch A. If the stack IDs are renumbered, so that stack ID 2 is now allocated to switch B, then the commands in the configuration script that refer to stack ID 2 will now be applied to switch B. If switch A is given a stack ID which is not in the configuration script, than it will not have any configuration applied.

We recommend that you only renumber the stack when it is initially configured, or during a time of major stack reconfiguration.

Steps to set up a VCStack

There are no set rules regarding the order in which stack configuration tasks need to be carried out. However, these steps provide a guideline to help ensure that the stack creation process goes smoothly.

1. Prepare the switches.

Before connecting any of the switches together:

- Ensure that all the switches have the same feature licences installed. If you have purchased feature licences to enable certain features to operate on the stack, then all stack members need to have the licences installed. If some stack members have feature licences installed for features that will not be used on the stack, and other switches do not have those licences installed, then remove those unnecessary licences.
- If you are using XEM stacking, you must install the XEMs into the switches before you can enter any stacking commands. An x900 Series switch will reject any stacking commands until a XEM-STK is installed.

```
x900(config)#stack 1 priority 0
% The unit has no XEM-STK installed, stacking commands are not
allowed
```

- Decide which stack member will be the active master and set its priority to 0, to ensure it becomes the active master. The command is:

  ```
  stack 1 priority 0
  ```

  (The switch, if it has not been stacked before, should have the default stack ID of 1.)

2. Install and power the stack master.

Install and power up the master switch. It will detect that there are no other members in the stack, so it will elect itself master. At bootup, it will output the following messages:

```
Notice: Possible stack, please wait whilst searching for members.
13:35:10 awplus-1 VCS[1164]: Member 1 (0000.cd28.07c0) has become the
Active Master
```

Then, after the switch has booted, the **show stack** command will show a stack with only one member:

```
Virtual Chassis Stacking summary information
ID  Pending ID  MAC address        Priority  Role
1   -           0000.cd28.07c0     0         Active Master
```

3. Install and power the backup member.

Install the next switch, connecting the stacking cable from that switch to the master.

Note: Make sure the stacking cables are crossed over between the stack members. This means that stack port 1 on switch 1 should connect to stack port 2 on switch 2. If this is not done, the stack links will not come up and the stack will not form.

Power up the switch. It will detect that there is already an active master, and so will come up as a backup member. The active master will assign it the first available stack ID.

At bootup, the new stack member outputs the following messages:

```
Notice: Possible stack, please wait whilst searching for members.
01:37:30 awplus-2 VCS[1157]:Member 1 (0000.cd28.07c0) has joined stack
01:37:30 awplus-2 VCS[1157]:Member 1 (0000.cd28.07c0) has become the
Active Master
```

After bootup, the **show stack** command will show that there are 2 switches in the stack:

```
x900#show stack
Virtual Chassis Stacking summary information
ID  Pending ID  MAC address        Priority  Role
1   -           0000.cd28.07c0     0         Active Master
2   -           0000.cd28.bfe5     128       Backup Member
```

The active master will check that the new stack member has the same software version as itself. If the software versions are different, the active master will use the software auto-synchronization mechanism to force the new stack member to run the same software version.

4. Install and power the next backup member.

Repeat step 3 for each of the other switches in the stack, remembering to connect port 2 of each new switch to port 1 of its neighbor. For the last switch added to the stack, connect port 1 of this switch to port 2 of the first installed switch.

5. Confirm that the stack is operating.

Check that the stack links have all come up successfully. This can be done by:

- checking the LEDs on the switches or XEMs
  The port LEDs for all stack members should be green. Port LEDs that are off or flashing amber indicate that the stack is not operating correctly. The master or status LED will be green on the switch that is the stack master.
- using the **show stack detail** command
  This command provides a snapshot of the stack status.

See the section "Checking stack status" on page 179 for more information about LEDs and the **show stack detail** command.

6. Check stack numbering.

If you are not satisfied with the stack IDs that the switches have been automatically assigned, then this is the right moment to remedy that. To change a stack ID, use the command:

```
stack <1-8> renumber <1-8>
```

It is usually a good idea to give the stack ID 1 to the active master, as it is quite natural to associate the lowest ID with the active master switch.

To sequentially renumber all stack members, you can:

- use the **select** button on the XEM-STK
- use the **stack renumber cascade** command
  To renumber the stack members and give stack ID 1 to the active master, use the command:

```
stack <current ID on master switch> renumber cascade 1
```

7. Reboot the switches.

Reboot all the stack members, and check that they all come up with the stack IDs that you are expecting. You can use the command **show stack detail** to check the stack IDs and the status of the neighbor connections.

8. Configure any priority changes.

If there is another switch that you want to be the one that takes over as active master, if the active master goes down, then set its priority to a value lower than the other switches:

```
stack <1-8> priority 10
```

9. Configure the stack as one switch.

You are now ready to configure the stack with port channels, VLANs, IP addresses, and so on. For example, to create a port channel that aggregates ports from two different stack members, you would configure as follows:

```
awplus#configure terminal
awplus (config)#interface port1.0.1
awplus (config-if)#channel-group 1 mode active
awplus (config-if)#interface port2.0.1
awplus (config-if)#channel-group 1 mode active
```

Once you are happy with the stack configuration, make a backup copy of the configuration file.

Steps to replace a stack member

If you need to replace a stack member, use the following steps to achieve a smooth transition.

1. Configure the Stack ID on the replacement switch.

Prepare the replacement switch by configuring it with the same stack ID as the switch that you are replacing.

```
stack (config)#stack <current stack ID> renumber <1-8>
```

2. Configure the feature licences.

Ensure that the replacement switch is configured with the same set of feature licences as the existing stack members.

3. Remove the failed switch.

Unplug the failed switch from the stack.

4. Install the replacement switch.

Connect the stacking cables to the replacement switch. Power up the replacement switch. It will detect that there is already an active master, and so will come up as a stack member.

The active master will check that the new stack member has the same software version as itself. If the software versions are different, the active master will use the software auto-synchronization mechanism to force the new stack member to run the same software version.

The new switch will also receive the startup configuration from the active master. As the replacement switch has been configured with the same stack ID as the replaced switch, it will receive exactly the same configuration as the replaced switch, and will operate exactly as that switch had.

Configuring the stack

The configuration script on a VCStack is shared between all stack members. When configuring the stack, you will need to be aware that the stack uses a specific port numbering scheme and that there are some minor restrictions to VLAN, IP subnet, and QoS configuration. There are also specific triggers available for stacking that you may wish to use. See each section for more detail:

- "Port numbering" on page 168
- "VLAN and IP subnet restrictions" on page 169
- "Quality of Service (QoS) restriction" on page 169
- "Stacking triggers" on page 170

For information about how the stack synchronises the startup configuration and running configuration between stack members, see the section "Software and configuration file synchronisation" on page 170.

Port numbering

In the AlliedWare Plus OS, switchport interfaces are always referenced as portx.y.z. The first number, or 'x', in the three number port identifier is the Stack ID.

For a stand-alone switch that has never been in a stack, the ports are always numbered 1.y.z. When configuring a stack, however, there will be stack members with other stack ID values. To configure ports on these switches, use the stack ID to refer to each physical switch, for example 2.0.1, 2.0.2, 2.0.3, for the first 3 ports on a switch with stack ID 2.

Please note that when a switch is removed from a stack, it maintains its stack ID number. If it is configured as a stand-alone switch, you will need to either change the stack ID back to 1, or use the current stack ID when configuring its ports.

VLAN and IP subnet restrictions

Internal communication between the stack members is carried out using IP packets sent over the stacking links. This stack management traffic is tagged with a specific ID, and uses IP addresses in a specified subnet.

By default, the VLAN and subnet used are:

- VLAN 4094
- Subnet 192.168.255.0/28

You may need to change these values if they clash with a VLAN ID or subnet that is already in use in the network. Use the commands:

```
stack management subnet <ip-address>
stack management vlan <2-4094>
```

Quality of Service (QoS) restriction

The management communication that the stack members exchange over the stacking link is vital for the successful operation of the stack. Be careful to avoid interrupting that communication.

This communication traffic is transmitted on egress queue 7 on the stacking ports.

We recommend that you avoid sending any user traffic into queue 7 on a VCStack. In any QoS configuration on a stack, prioritize traffic only into queues 0-6. Moreover, you should ensure that the CoS-to-Queue map does not automatically use queue 7 for the transmission of packets that are received with a CoS value of 7. To ensure this, change the cos-to-queue map to put packets with a CoS value of 7 into queue 6. Use the command:

```
mls qos map cos-queue 7 to 6
```

Stacking triggers

There are five trigger types defined for particular stack events. Any scripts that you configure for triggers are copied from the active master to all stack members. The triggers are:

```
type master-fail
type stack member join
type stack member leave
type stack xem-stk up
type stack xem-stk down
```

Software and configuration file synchronisation

A VCStack requires that the software version and the configuration files on all stack members are the same. If these files are not the same, then the stack cannot form or operate correctly. To make stack formation easy, all these files are synchronised automatically on the stack by the stack master. The following are synchronised:

- "Software release auto-synchronisation" on page 170
- "Shared running configuration" on page 171
- "Shared startup configuration" on page 171
- "Scripts" on page 172

Software release auto-synchronisation

The VCStack implementation supports a feature called software auto-synchronization. The effect of this feature is that when a new member joins a stack and has a software release that is different to the active master, then the active master's software release is copied onto the new member. The new member then reboots and comes up on that release.

The sequence of events that occurs at startup is shown in the next figure.

```
Notice: Possible stack, please wait whilst searching for members.
01:34:59 awplus-2 VCS[1157]: Member 1 (0000.cd28.07c0) has joined stack
01:34:59 awplus-2 VCS[1157]: Member 1  (0000.cd28.07c0) has become the
Active Master
01:35:02 awplus-2 VCS[1157]: Software incompatibility detected.
01:35:02 awplus-2 VCS[1157]: Auto synchronization starts. Unit will
reboot once that finishes. Please wait..
01:36:50 awplus-2 VCS: The new software r1-5.2.2-0.2-rc1.rel is set as
preferred software and used at the next reboot.
01:36:50 awplus-2 VCS: Restarting system.
```

The software auto-synchronization feature is enabled on all switches by default. You can enable or disable it using the command:

```
(no) stack <1-8> software-auto-synchronization
```

If you disable software auto-synchronisation, you may disrupt a stack forming if the stack members have different software releases. If a new member joins the stack, and is running a software version that is different to the active master, the active master will reject the new member from the stack if it cannot synchronise the software releases.

Note that this feature can result in the new stack member downgrading its software release if the active master is running an older software version.

Shared running configuration

A single, unified running configuration is shared by all the switches in a stack.

This means that the running configuration on each stack member is exactly the same, and contains the configuration information for all stack members. For example, on a two-switch stack, switch A with stack ID 1 will show the configuration for switch B with stack ID 2, as well as its own configuration. If the running configuration on switch B with stack ID 2 is examined, the output will be exactly the same as that produced by switch A.

Shared startup configuration

Once a stack has formed, the startup configuration stored on each stack member is overwritten by the active master's startup configuration. This means all the switches in a stack have exactly the same startup configuration script.

Every time the startup configuration script on the active master is changed, for example when the running config is copied to the startup config, the updated startup script is copied to all the other stack members.

```
stack(config)#boot config-file interop.cfg
VCS synchronizing file across the stack, please wait..
File synchronization with stack member-2 successfully completed
[DONE]
```

It is important that you take this behaviour into account when you first create a stack. If you have carefully crafted a startup configuration on the switch that you intend to be the active master, but some mistake results in a different switch becoming the active master when the stack forms, then the intended startup configuration will be overwritten by something else (assuming the same filename was used for both configurations).

We recommend that you:

- take care when first installing the VCStack to ensure that you configure the stack priority values correctly. If you do not, the wrong stack member may become the

active master on first boot, and overwrite the stack with the wrong startup configuration.

- backup the startup configuration of the intended active master before connecting the switch to the stack. You can make another copy on the switch's flash file system with the command:

```
copy startup_filename.cfg backup_filename.cfg
```

As well as the running configuration and startup configuration, the stack synchronises the **boot backup configuration** file, and the **fallback configuration** file. Like the startup configuration file, these are synchronised from the active master's file system, so the same recommendations apply.

Scripts

Script files stored on the active master's file system are copied to the stack members.

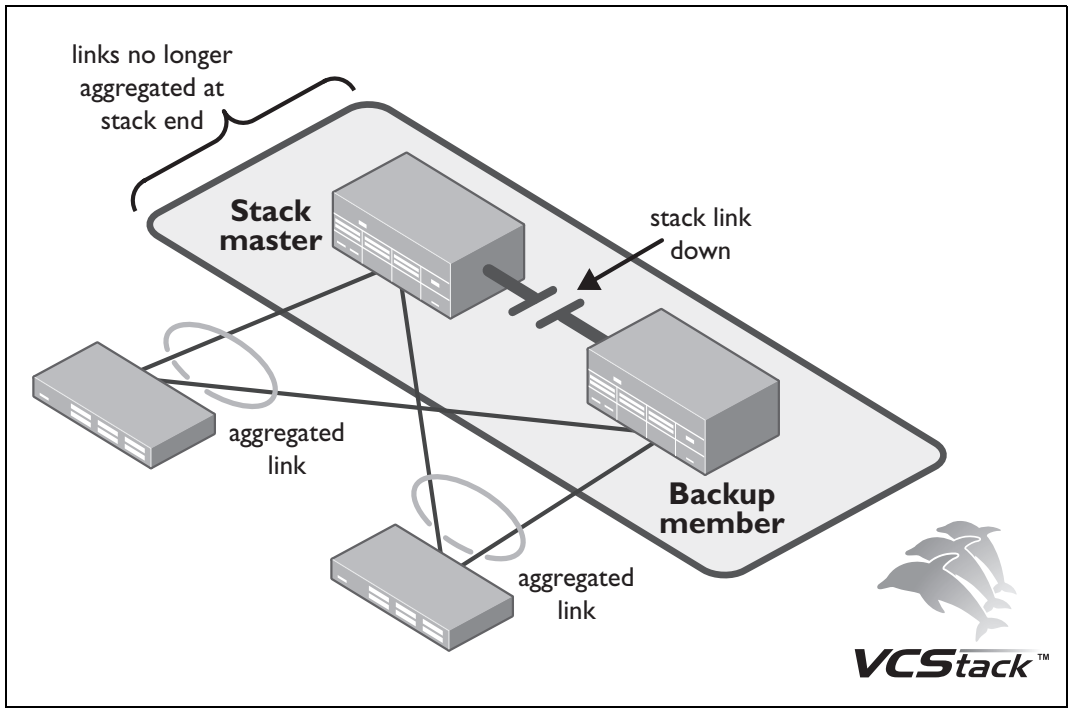
Failover and recovery

A VCStack behaves in a reliable and consistent manner if any component fails, whether the problem is with a stacking link or stack member. However, before looking at how the stack reacts to each of those scenarios, we first need to understand a generic mechanism that is available in Allied Telesis VCStack to help deal with failure scenarios– namely the resiliency link.

The resiliency link feature

If one or more stacking links fail, so that some stack members are no longer in contact with the active master switch, then the other stack members are left with a dilemma. Should they assume that the active master switch has gone down, and re-elect a new active master, or should they assume that the active master is still operating?

This problem is particularly important when a stack has multiple connections to edge switches in the network. In this network scenario, if a stack splits into 2 active sections that are operating independently, then the edge switches will continue to treat their uplink ports as aggregated and share traffic across the links. However, the broken stack link could mean that traffic arriving at some of the stack members cannot reach the intended destination.



So it is necessary to have a backup mechanism through which stack members can check if the active master is still active in case the stack link communication to the active master is lost.

The mechanism provided in Allied Telesis VCStack is called the **resiliency link**. The out-of-band Ethernet management port on the switches can be configured as a **resiliency port**. On 2-switch stacks, a single cable connects these ports to one another. On larger stacks, a hub or switch is used to connect the stack resiliency ports together. The stack members all listen for periodic (one per second) Health Check messages from the master. As long as the active master sends Health Check messages, the other stack members know that the active master is still active.

This means that the stack members can know whether the active master is still operating if they lose contact with the active master through the stacking links.

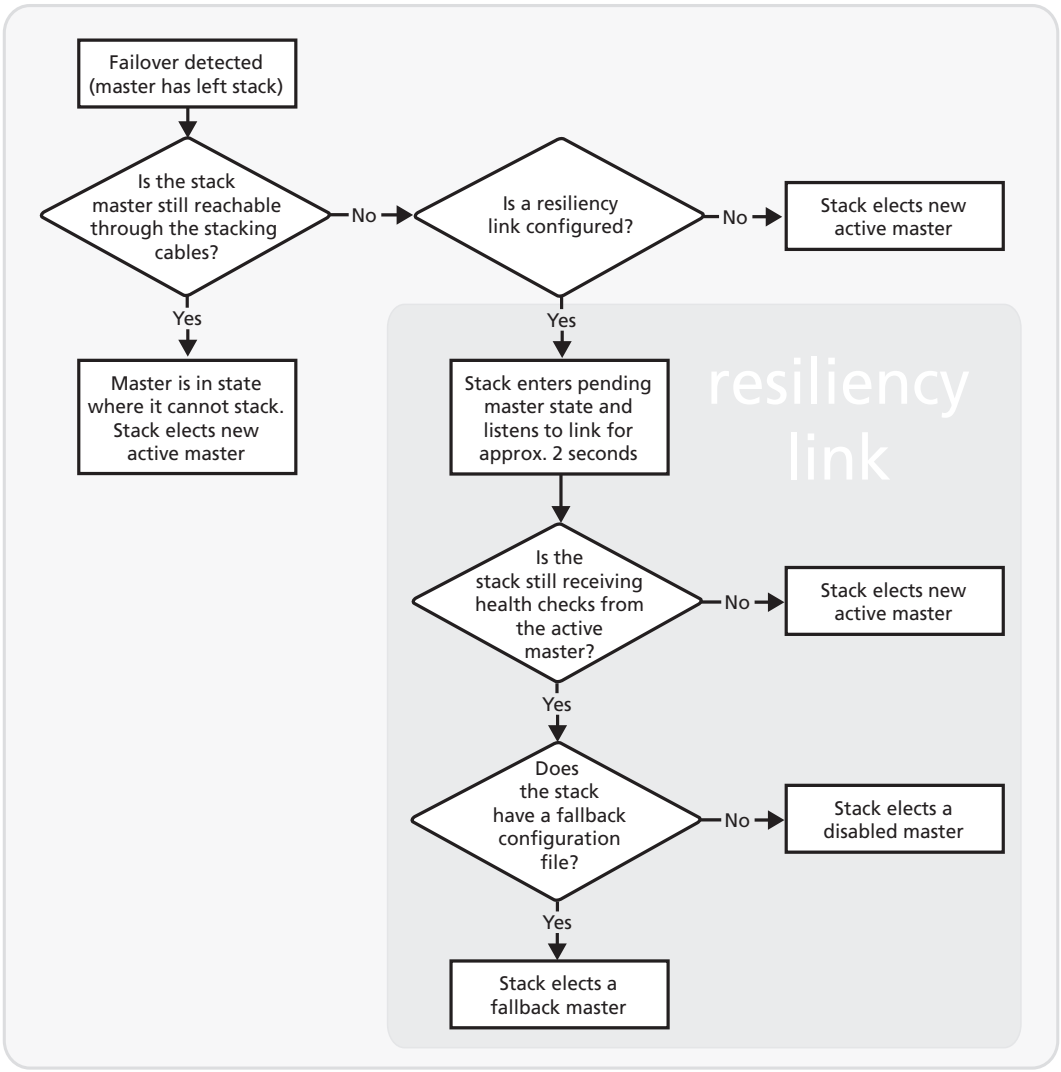
The out-of-band Ethernet port is configured as a resiliency port with the command:

```
stack resiliencylink eth0
```

Note that even if you configure the eth0 port as a resiliency port, you can still use it for out-of-band management.

Recovery from losing stack link communication to the active master

The logic of how stack members react when they lose stack-link communication with the active master is illustrated in the following flowchart.



Stack behaviour without resiliency link

If there is no resiliency link configured and stack members lose stack link communication to the active master, then the stack members assume that the active master is down. So the stack members retain their existing configurations, and elect a new active master.

Stack behaviour with resiliency link

If the resiliency link is configured, then the stack members can listen for health checks from the active master. While listening for the active master, the stack members enter the **pending master** state. In this state they continue forwarding traffic while they wait for a short time period to see whether they are receiving health checks from the active master.

Health check heartbeats are sent only by the Master over the resiliency link.

How the stack then reacts after the time period has elapsed is based on whether it has received health checks:

- If the stack is not receiving health checks from the active master, then the stack members conclude that the active master is down. So the stack members retain their existing configurations, and elect a new active master. Note that if the stack has a resiliency link configured but not physically connected, the stack will also conclude that the active master is down.
- If the stack is receiving health checks from the active master, then the stack members cannot continue to operate with the same configuration as this will cause problems with data flow on the network. So the stack must use a predetermined **fallback configuration** if it is available, or if not they must stop forwarding traffic:
  - If the fallback configuration does exist, then the switches which cannot contact the active master reconfigure themselves with this configuration, and elect a **fallback master** from amongst themselves. This means that the stack effectively splits into two stubs–the stub containing the active master, and the stub containing the fallback master. Both stub stacks operate as independent stacks.
  - If the fallback configuration does not exist, then the stack members which cannot contact the active master disable all their Ethernet switching ports, and elect a **disabled master** from amongst themselves.

The fallback configuration

The fallback configuration is a configuration script that enables stack members that have lost contact with the active master to continue forwarding data, if possible. If you can be confident that some ports on those switches can remain active without causing problems, then the fallback configuration should have these ports active. However, for any ports that are members of aggregated links, the best option is to shutdown these ports in the fallback configuration. The fallback configuration may also configure a unique management IP address for the fallback master stub, so that you can still remotely manage this stack.

The fallback configuration script is set on the stack with the command:

```
stack fallback-config URL
```

Typically, the script will be stored in flash.

In the following example, the stacking cables have been removed from a stack with 2 members. The stack has a fallback configuration. The following figure shows the original active master's state has remained in the active master:

```
stack#show stack
Virtual Chassis Stacking summary information

ID  Pending ID  MAC address        Priority  Role
1   -           0009.41fd.c033     100       Active Master
```

The other stack, consisting of 1 switch in this example, elects a fallback master, as shown in the following figure:

```
stack#show stack
Virtual Chassis Stacking summary information

ID  Pending ID  MAC address        Priority  Role
2   -           0000.cd28.5239     128       Fallback Master
```

Recovery from other failure situations

The following two tables show the recovery options that the stack takes in a variety of different failure situations. **Fallback action** means that the switch reconfigures itself using the fallback configuration script. Where there is no fallback script, the switch disables all its switch ports.

The following table shows the recovery options when the failure has occurred on the stack master.

Event	Master reaction	Backup reaction –with resiliency link	Backup reaction –without resiliency link
Event on master node			
Both stack links removed	no change	fallback action	re-elect master
XEM-STK removed/faulty	no change	fallback action	re-elect master
Software application problem (lock-up or continual crashes)	reboot as backup	re-elect master	re-elect master
Kernel crash or lock-up	frozen	re-elect master	re-elect master
Power down or PSU failure	powered down	re-elect master	re-elect master

This next table shows the recovery options for the stack when the failure has occurred on a backup member. Note that in this table the backup reaction columns are for the backup member who has had the failure event.

Event	Master reaction	Backup reaction –with resiliency link	Backup reaction –without resiliency link
Event on backup member			
Both stack links removed	no change	fallback action	re-elect master
XEM-STK removed/faulty	no change	fallback action	re-elect master
Software application problem (lock-up or continual crashes)	no change	reboot as backup	reboot as backup
Kernel crash or lock-up	no change	frozen	frozen
Power down or PSU failure	no change	powered down	powered down

Repairing a broken stack

When two stub stacks are reconnected, the stack will detect if there is more than one master (active, disabled, or fallback). You will see console messages and logs that report a **duplicate master** was detected. The stack will carry out an election to choose the true active master, based on priorities and MAC addresses. Any losing master will reboot and become a backup member. The switches that have been running on their fallback configuration (most likely those in the same stub as the losing active master) will reboot and come up on the shared configuration pushed to them by the winning active master. When a stack detects a duplicate master, a debug snapshot file is created and stored in flash. The file is called debug-duplicate-master-<*time & date*>.tgz. The debug snapshot does not contain a.core file, as is produced when a process unexpectedly terminates on a switch. Instead, it contains a snapshot of various pieces of information within the software at the time when the stack detected the duplicate master.

Executing commands and managing files on a specific stack member

There are some limited actions that you can do on an individual switch member:

- **Executing commands**
  You can use the **reload** command to reboot an individual switch, or use **show** commands to see the individual physical state of a switch and its file system. Show commands that relate to the ports, counters, or configuration are applicable for the stack only.
- **Managing files**
  You can manage files on an individual switch. Note that some files are synchronised between switches. See the "Software and configuration file synchronisation" section for more information.

Executing commands

If you want to run a command that displays information from a specific stack member, you can prefix the command with the syntax:

```
remote-command <stack ID>
```

For example, to see the full state of all the file systems on the stack member with stack ID 3, use the command:

```
remote-command 3 show file systems
```

To see the processes running on the stack member with stack ID 2, use the command:

```
remote-command 2 show process
```

To use the **show system environment** command on the stack member with stack ID 2, use the command:

```
remote-command 2 show system environment
```

This will display the following output:

```
Stack Environment Monitoring Status

Stack member 2:

Overall Status: Normal

Resource ID: 1  Name: PSU bay 1
ID  Sensor (Units)                  Reading  Low Limit High Limit Status
1   Device Present                      Yes          -          -     Ok
2   PSU Overtemp                         No          -          -     Ok
3   PSU Fan Fail                         No          -          -     Ok
4   PSU Power Output                    Yes          -          -     Ok
5   PSU AC Supply                       Yes          -          -     Ok
```

Additionally, the **reload** command can take a stack ID as an extra parameter. So, to reboot just the stack member with stack ID 4, use the command:

```
reload stack-member 4
```

You will get the following question from the stack:

```
stack#reload stack-member 4
reboot stack member 4 system? (y/n):
```

Managing files

If you wish to perform an action on another stack member's file system, the syntax is:

```
<stack-member-name>/flash:[/]<file name>
```

The **stack-member-name** is not the stack ID, but is an extended hostname formed as:

```
<hostname>-<stack ID>
```

So, if the hostname of the stack is **BlueCore**, then the stack-member-name for switch 2 in the stack is:

```
BlueCore-2
```

If you do not use the *stack-member-name* prefix, then the command refers to a file that resides on the stack master.

You can also use the *stack-member-name* syntax for the directory command. To view the contents of the flash file system on a specific stack member, you can use the syntax:

```
dir <stack-member-name>/flash:/
```

Monitoring and troubleshooting

The physical connection and the signalling communications between stack members are vital to stacking. Because of this, the show commands and debugging facilities for VCStack are oriented around stack port status and stack signalling communications.

Checking stack status

You can check that the stack links have come up successfully by:

- checking the LEDs on the switch or XEM
- using the **show stack detail** command

LEDs

The LEDs on the XEM-STK show the following:

LED	State	Meaning
Port 1 and Port 2	Green	A stacking link is established.
	Amber (flashing slowly)	The link has transmission fault.
	Off	The stacking link is down.
Status	Green	The switch is the stack master.
	Amber	The switch is a backup member.
	Green (flashing)	The stack is selecting a stack master.
	Off	The switch is not a stack member.
Numeric ID	1 to 8	The stack member ID.
	Off	The switch is not a stack member.

The front panel of the SwitchBlade x908 has the following LEDs for monitoring high-speed stacking:

LED	State	Meaning
Port 1 and Port 2	Green	A stacking link is established.
	Amber (flashing slowly)	The link has transmission fault.
	Off	The stacking link is down.
Master	Green	The switch is the stack master.
	Amber	The switch is a backup member.
	Green (flashing)	The stack is selecting a stack master.
	Off	The switch is not a stack member.

The front panel of the x600 Series switch has the following LEDs for monitoring stacking:

LED	State	Meaning
MSTR	Green	The switch is the stack master.
	Off	The switch is a backup member.
1 L/A and 2 L/A	Green	A stacking link is established on that link.
	Green (flashing)	The link is transmitting or receiving data.
	Off	The stacking link is down.
PRES	On	An AT-STACKXG module is correctly installed in the switch.
	Off	There is no AT-STACKXG installed in the switch, or the module is installed incorrectly.

Show stack detail command

The **show stack detail** command provides a snapshot of the stack status. It includes a full summary of the status of all the stack members and the status of their connections to the other member. This allows you to check that all the stack members have active connections to each other and have recognised correctly which neighboring switch is connected to each of their stacking ports.

```
x900#show stack detail
Virtual Chassis Stacking detailed information

Stack Status:
-----------------------------------------------------------------
Normal operation
Operational Status                      Enabled
Management VLAN ID                      4094
Management VLAN subnet address          192.168.255.0

Stack member 1:
-----------------------------------------------------------------
ID                                      1
Pending ID                              -
MAC address                             00-00-cd-28-07-c0
Last role change                        Tue Oct  7 13:35:10 2008
Product type                            x900-12XT/S
Role                                    Active Master
Priority                                0
Host name                               x900
S/W version auto synchronization        On
Fallback config                         Not configured
Resiliency link                         Configured
Port 1.1.1 status                       Learnt neighbor 2
Port 1.1.2 status                       Learnt neighbor 2

Stack member 2:
-----------------------------------------------------------------
ID                                      2
Pending ID                              -
MAC address                             00-00-cd-28-bf-e5
Last role change                        Wed Oct  8 01:37:33 2008
Product type                            x900-12XT/S
Role                                    Backup Member
Priority                                128
Host name                               x900-2
S/W version auto synchronization        On
Fallback config                         Not configured
Resiliency link                         Failed
Port 2.1.1 status                       Learnt neighbor 1
Port 2.1.2 status                       Learnt neighbor 1
```

Stack debug output

The clearest way to receive stack debug output is to enable console logging of VCStack messages. Using this method, you receive just the stack debug messages without the other messages that are seen when you enable terminal monitor.

To enable console logging of VCStack messages, use the command:

```
log console program VCS
```

You must enter the parameter **VCS** in uppercase letters when you enter this command.

This command enables you to receive a record of stack link and communication events. For example if a stack port goes down, and then comes up again, the series of messages output is:

```
16:44:58 x900-2 VCS[1157]: Link down event on stack link 2.1.1
16:44:58 x900-2 VCS[1157]: STK TRACE: Link 2.1.1: N/A
16:44:58 x900-2 VCS[1157]: STK TRACE: Link 2.1.2: --> 1
(0000.cd28.07c0)
16:44:58 x900-2 VCS[1157]: STK TRACE: Stack topology has changed -
updating stack H/W routes for L2 connectivity
16:45:04 x900-2 VCS[1157]: Link up event on stack link 2.1.1
16:45:04 x900-2 VCS[1157]: Beginning neighbor discovery on link 2.1.1
16:45:05 x900-2 VCS[1157]: STK TRACE: < Rxd Hello msg on 2.1.1 (nbr ?):
16:45:05 x900-2 VCS[1157]: STK TRACE: Received topology for neighbor
0000.cd28.07c0, member-ID 1
16:45:05 x900-2 VCS[1157]: STK TRACE: Link 2.1.1: --> 1
(0000.cd28.07c0)
16:45:05 x900-2 VCS[1157]: STK TRACE: Link 2.1.2: --> 1
(0000.cd28.07c0)
16:45:05 x900-2 VCS[1157]: STK TRACE: > Txd Hello msg on 2.1.1 (nbr 1):
16:45:05 x900-2 VCS[1157]: neighbor discovery on link 2.1.1 has
successfully completed
16:45:05 x900-2 VCS[1157]: STK TRACE: Stack topology has changed -
updating stack H/W routes for L2 connectivity
16:45:05 x900-2 VCS[1157]: STK TRACE: < Rxd Hello msg on 2.1.1 (nbr 1):
16:45:05 x900-2 VCS[1157]: STK TRACE: Received topology for neighbor
0000.cd28.07c0, member-ID 1
```

If stack-link communication to a stack member is completely lost, the series of messages output is:

```
16:47:53 x900 VCS[1164]: Link down event on stack link 1.1.1
16:47:53 x900 VCS[1164]: STK TRACE: Link 1.1.1: N/A
16:47:53 x900 VCS[1164]: STK TRACE: Link 1.1.2: --> 2 (0000.cd28.bfe5)
16:47:53 x900 VCS[1164]: STK TRACE: Stack topology has changed -
updating stack H/W routes for L2 connectivity
16:47:53 x900 VCS[1164]: Stack member 2 has become unreachable via
Layer-2 traffic
16:47:53 x900 VCS[1164]: Link down event on stack link 1.1.2
16:47:53 x900 VCS[1164]: Link between members 1 and 2 is down
16:47:53 x900 VCS[1164]: STK TRACE: Link 1.1.1: N/A
16:47:53 x900 VCS[1164]: STK TRACE: Link 1.1.2: N/A
16:47:53 x900 VCS[1164]: STK TRACE: Member 2 (0000.cd28.bfe5) is
leaving the stack
16:47:53 x900 VCS[1164]: STK TRACE: Shutting down access to member 2's
file system
16:47:54 x900 VCS[1164]: STK TRACE: The stack Backup-Master is member 1
16:47:54 x900 VCS[1164]: STK TRACE: Member 2 (0000.cd28.bfe5) state
Backup Member --> Leaving
16:47:54 x900 VCS[1164]: STK TRACE: Now removing member's HA workload
across stack
16:47:54 x900 VCS[1164]: STK TRACE: Stack topology has changed -
updating stack H/W routes for L2 connectivity
16:47:55 x900 VCS[1164]: Member 2 (00-00-cd-28-bf-e5) has left stack
16:47:55 x900 VCS[1164]: STK TRACE: Stopping file replication...
16:47:57 x900 VCS[1164]: STK TRACE: Now assigned HA workload across
stack
16:47:57 x900 VCS[1164]: STK TRACE: Disabling intra-stack
communication interface
```

To see a very detailed log of stacking related events after they have occurred, and other VCStack debug information, use the command:

```
show stack full-debug [<1-8>]
```

Even if you had not been capturing stack log output at the moment an event occurred, you can still retrospectively obtain the logging information by using this command. If you do not specify a stack ID, then each stack member's output is displayed, one after the other. This command produces a large amount of output, as shown in the following figure. The **VCS debug** section contains the detailed log information.

```
Detailed debugging information for stack member 1

VCS host configuration
------------------------------------------------------------
Unit stackable, stack H/W present
VCS mgmt VLAN 4094, subnet 192.168.255.0
Topology: Broken-Ring
IMI config: Normal
neighbor port-1: 3
neighbor port-2: N/A
  ID Mac Address       PP IP             State
> 1  0015.77c2.4d54    0  192.168.255.1  Active Master
  3  0015.77ad.fb09    0  192.168.255.3  Synchronizing*
```

```
VCS Inter-process connectivity configuration
--------------------------------------------------------------------
Type       Lower      Upper      Port Identity              Publication
-------------------------------- ------------------------- -----------
0        16781313   16781313   <1.1.1:2123055098>       2123055099 zone
           16781315   16781315   <1.1.3:2160787450>         2160787451
1         1         1         <1.1.1:2123055099>        2123055100 node
9500      1        1        <1.1.1:2123054932>       2123054933 cluster
           3          3          <1.1.3:2160787202>         2160787203
          100       100       <1.1.1:2123055096>        2123055097 node
          200       200       <1.1.1:2123055097>        2123055098 node
                              <1.1.1:2123054865>        2123054866 node
                              <1.1.1:2123054945>        2123054946 node
                              <1.1.1:2123055040>        2123055041 node
.


Link <multicast-link>
  Window:20 packets
  RX packets:5 fragments:0/0 bundles:0/0
  TX packets:6 fragments:0/0 bundles:0/0
  RX naks:0 defs:0 dups:0
  TX naks:0 acks:0 dups:0
  Congestion bearer:0 link:0  Send queue max:2 avg:1


Link <1.1.1:vlan4094-1.1.3:vlan4094>
  ACTIVE  MTU:1500  Priority:10  Tolerance:3000 ms  Window:50 packets
  RX packets:1 fragments:0/0 bundles:0/0
  TX packets:7 fragments:0/0 bundles:0/0
  TX profile sample:3 packets  average:60 octets
  0-64:100% -256:0% -1024:0% -4096:0% -16354:0% -32768:0% -66000:0%
  RX states:89 probes:43 naks:0 defs:0 dups:0 tos:0
  TX states:89 probes:43 naks:0 acks:0 dups:0
  Congestion bearer:0 link:0  Send queue max:5 avg:0



VCS debug
------------------------------------------------------------
2008 Nov 25 09:02:15 user.info awplus kernel: TIPC: Activated (version
1.6.2 compiled Nov 21 2008 15:56:26)
2008 Nov 25 09:02:15 user.info awplus kernel: TIPC: Started in single
node mode
2008 Nov 25 09:02:17 user.info awplus VCS[1006]: Try to parse /flash/
cr23774.cfg
2008 Nov 25 09:02:17 user.info awplus kernel: TIPC: Started in network
mode
2008 Nov 25 09:02:17 user.info awplus kernel: TIPC: Own node address
<1.1.1>, network identity 4711
2008 Nov 25 09:02:23 user.debug awplus VCS[1027]: STK DEV  : Stacking
topology event counters register is successful.

2008 Nov 25 09:02:23 user.debug awplus VCS[1027]: STK DEV  : Stacking
port 1 topology event counters register is successful.
```

```
2008 Nov 25 09:02:23 user.debug awplus VCS[1027]: STK DEV  : Stacking
port 2 topology event counters register is successful.
2008 Nov 25 09:02:23 user.debug awplus VCS[1027]: STK DEV  : Stacking
topology message counters register is successful.
2008 Nov 25 09:02:23 user.debug awplus VCS[1027]: STK DEV  : Stacking
topology error counters register is successful.


Trace debug
-------------------------------------------------------------
09:02:37 awplus-1 HSL[1252]: Updating stack port 0.1 to block bcast/
mcast flooding
09:02:37 awplus-1 HSL[1252]: Updating stack port 0.2 to block bcast/
mcast flooding
09:02:50 awplus-1 HSL[1257]: Member-3 0015.77ad.fb09 JOINED stack
09:02:50 awplus-1 HSL[1252]: Updating stack port 0.1 to unblock bcast/
mcast flooding
09:02:57 awplus HSL[1257]: Stack member-1 is now Active Master (we are
MASTER)

VCS log messages of interest
-------------------------------------------------------------
2008 Nov 25 09:02:37 user.err awplus HSL[1034]: HSL: ERROR: Interface
5 not found in database
```

Counters

You can obtain detailed counters relating to stack events and signalling packets with the **show counter stack** command. You can use these for tracking down whether signalling packets are being lost, by checking if there are discrepancies between the number sent from one switch and the number received by its neighbor. The event counters make it possible to see if unexpected events have been occurring on the stack. The following figure is an output example.

```
Virtual Chassis Stacking counters

Stack member 1:

Topology Event counters
Units joined          ......... 4
Units left            ......... 3
Links up              ......... 16
Links down            ......... 14
ID conflict           ......... 0
Master conflict       ......... 3
Master failover       ......... 0
Master elected        ......... 1
Master discovered     ......... 0
SW autoupgrades       ......... 0
```

```
Stack Port 1 Topology Event counters
Link up                ......... 14
Link down              ......... 14
Nbr re-init            ......... 0
Nbr incompatible       ......... 0
Nbr 2way comms         ......... 6
Nbr full comms         ......... 16

Stack Port 2 Topology Event counters
Link up                ......... 15
Link down              ......... 14
Nbr re-init            ......... 0
Nbr incompatible       ......... 0
Nbr 2way comms         ......... 10
Nbr full comms         ......... 0
Topology Message counters
Tx Total               ......... 183
Tx Hellos              ......... 133
Tx Topo DB             ......... 0
Tx Topo update         ......... 0
Tx Link event          ......... 14
Tx Reinitialise        ......... 3
Tx Port 1              ......... 85
Tx Port 2              ......... 59
Tx 1-hop transport     ......... 144
Tx Layer-2 transport   ......... 39
Rx Total               ......... 2903
Rx Hellos              ......... 19
Rx Topo DB             ......... 0
Rx Topo update         ......... 0
Rx Link event          ......... 0
Rx Reinitialise        ......... 3
Rx Port 1              ......... 12
Rx Port 2              ......... 10
Rx 1-hop transport     ......... 22
Rx Layer-2 transport   ......... 2881
Rx Total               ......... 2903
Rx Hellos              ......... 19
Rx Topo DB             ......... 0
Rx Topo update         ......... 0
Rx Link event          ......... 0
Rx Reinitialise        ......... 3
Rx Port 1              ......... 12
Rx Port 2              ......... 10
Rx 1-hop transport     ......... 22
Rx Layer-2 transport   ......... 2881

Topology Error counters
Version unsupported    ......... 0
Product unsupported    ......... 0
XEM unsupported        ......... 0
Too many units         ......... 0
Invalid messages       ......... 0
Stack member 2:
Topology Event counters
Units joined           ......... 1
Units left             ......... 0
Links up               ......... 2
Links down             ......... 0
ID conflict            ......... 0
Master conflict        ......... 0
```

```
Master failover       ......... 0
Master elected        ......... 0
Master discovered     ......... 1
SW autoupgrades       ......... 0
Stack Port 1 Topology Event counters
Link up               ......... 1
Link down             ......... 0
Nbr re-init           ......... 0
Nbr incompatible      ......... 0
Nbr 2way comms        ......... 1
Nbr full comms        .........


Stack Port 2 Topology Event counters
Link up               ......... 1
Link down             ......... 0
Nbr re-init           ......... 0
Nbr incompatible      ......... 0
Nbr 2way comms        ......... 1
Nbr full comms        ......... 0

Topology Message counters
Tx Total              ......... 2766
Tx Hellos             ......... 2
Tx Topo DB            ......... 0
Tx Topo update        ......... 0
Tx Link event         ......... 0
Tx Reinitialise       ......... 0
Tx Port 1             ......... 1
Tx Port 2             ......... 1
Tx 1-hop transport    ......... 2
Tx Layer-2 transport  ......... 2764
Rx Total              ......... 9
Rx Hellos             ......... 4
Rx Topo DB            ......... 0
Rx Topo update        ......... 0
Rx Link event         ......... 0
Rx Reinitialise       ......... 0
Rx Port 1             ......... 3
Rx Port 2             ......... 3
Rx 1-hop transport    ......... 6
Rx Layer-2 transport  ......... 3

Topology Error counters
Version unsupported   ......... 0
Product unsupported   ......... 0
XEM unsupported       ......... 0
Too many units        ......... 0
Invalid messages      ......... 0
```

Chapter 10 | Troubleshooting

Introduction

This chapter provides practical advice on how to go about troubleshooting different issues that arise in networks. It gives examples of the kind of information you need to capture, and describes techniques for capturing all relevant information and for helping you or someone else to analyse the captured debug information.

A few different strategies for successful network troubleshooting are presented. Several specific problem scenarios are examined, and the way to troubleshoot these scenarios is discussed.

The final section of the chapter presents some of the powerful troubleshooting tools available in the AlliedWare Plus OS.

List of terms

OSI
A scheme for understanding network communications as a series of layers.

Show tech
A fundamental troubleshooting command that captures a snapshot of the state of a switch.

General techniques for network troubleshooting

When starting on a debugging session, you need to have questions in mind that you are aiming to answer. Questions such as:

- Is the problem widespread across the network, or localised in one area?
- Is the problem due to a configuration error on one or more units in the network?
- Is the problem due to an interop issue between different devices on the network?
- Is there a cabling error in the network?
- Is the problem intermittent or consistent?
- Does a particular sequence of events on the network trigger the problem?
- Is the problem related to some particular data that is flowing in the network?
- Can the problem be narrowed to just one specific device in the network?
- Is the issue due to a hardware failure on some device in the network?
- Is the issue maybe due to a software problem on a device in the network?

It may not be possible to precisely answer these questions during the debugging session (in particular, subtle hardware errors can be hard to distinguish from software bugs). However, it is important to approach the debugging with the desire to answer these questions as a guiding goal.

Often, as a result of your debugging and analysis, you will isolate configuration error(s) and correct them, or you will identify a faulty piece of hardware, and get it replaced, thereby resolving the network issue.

But, there will also be plenty of occasions where the problem cannot be solved there and then (particularly if it is a software bug), and the issue needs to be escalated to another part of the customer support process.

In these cases, the speed with which the issue is eventually resolved can be greatly influenced by the quality of your onsite debugging and analysis, and the quality of your issue escalation report.

Provide clear and complete information

To be effective in getting a full understanding of the problem, and provide a clear report if the issue needs to be escalated, the type of information you need to capture is:

- A **description** of the problem and what you believe to be the **cause** of the issue.
- What troubleshooting steps have been taken to back up your theory of the cause.
- Output of the relevant **show** commands and hardware tables. Capture the device in a working state as well as a non-working state, to enable a comparison between the two states.
- Ethereal captures, the results of communication tests (pings, traceroutes, etc.) if relevant.

- A **description** of any work-around you might have tried.
- A network diagram.
- A full **show tech** output from all relevant devices.

Start a debugging session

It is important to capture enough information (and the right sort of information) so you can analyse your data at a later stage. It is also important to capture information that someone else can use to effectively analyse the situation. This could be so that they can confirm your theory about the root cause of the problem, or, if you were not able to nail down the root cause, so that they can develop their own theories about what the cause might be.

Tip #1 Capture everything

As soon as you attach your terminal emulator to the console port of the first piece of equipment, turn on the logging facility of the terminal emulator. You will capture a lot of (seemingly) irrelevant stuff, but you will not fail to capture the important stuff.

Tip #2 Start by using "show tech-support" to capture the initial state

It is important to approach the problem with an open mind, and start with a general survey of the equipment you are looking at, to give an initial picture of how things were at the start of debugging session. This might be the "network operating OK" state, or the "things going wrong" state, depending on what state the network was in when you started.

Therefore, on every relevant switch, start by running the command **show tech-support.**

This command captures the output of a large number of show commands and saves it in a file on Flash called **tech-support.txt.gz**. If a tech-support file already exists, the switch doesn't override it, instead it creates a new file with the time stamp in the filename, such as: *tech-support-20090807130032.txt.gz*

Make the captured output as useful as possible

It is likely that your debug capture will need to be analysed later, either by someone else or by yourself. Therefore, you need to record your thinking and actions as you go along, to give context to all the debugging that you capture.

Comment your debug capture

It is likely that your debug capture will need to be analysed later, either by someone else or by yourself. Therefore, you need to record your thinking and actions as you go along, to give context to all the debugging that you capture.

This makes it clear:

- **why** you were looking at particular counters etc.
- **how** the captures relate to external events. So often it is vital to know whether a particular state change or counter change seen in the debug capture happened before or after a particular external event. For example, “did that switch port go into STP blocking state before or after port 1.1.2 was unplugged?”

Tip #3 Type your thoughts and actions

Throughout the capture, type (just at the CLI prompt) what you are thinking, and what you are doing. Just write it as a stream-of-consciousness, such as:

```
"Hmmm, that counter value looks strange, let's look at this a bit more
"Right, the ping to the PC on port 1.0.5 just starting failing.
```

Of course, your consciousness might not stream in English. If not, then write the comments in your own language; the important ones can always be translated later.

Writing comments at the CLI prompt will mean that you get error messages from the unit like:

```
awplus#Now I am going to send the IGMP report from the client, and see if
any PIM counters or states change

    % Invalid input detected at '^' marker.
```

This doesn't matter. Anyone analysing the capture will be able to mentally filter out these error messages.

When you see something really significant, you might want to highlight it so it is easy to find later. If you start a comment with a string of !!!!!!!! or ######, it will stand out when you browse or search through the file.

Check whether counters are changing

Often, the investigation of a theory requires looking for what is changing. For example, are the error counters on a particular port increasing, is port A now sending more multicast packets than port B is receiving, are entries in the route ARP table being refreshed, etc.? This brings us to Tip #4.

Tip #4 Perform particular "show" commands a few times in succession.

You might see certain values changing that give a vital clue to what is happening. Or, if you don't notice them, someone analysing the capture after you might.

You can also re-run the command **show tech-support** after any significant event.

Check the rate of change

More subtly, the **rate** at which certain things change can often be significant-for example, how long did it take that unit to make the transition from VRRP master to slave, how long did that route take to age out, etc.? Unfortunately, the outputs of **show** commands are not time-stamped, so sometimes it is necessary to do the "time stamping" yourself.

Tip #5 **If you are investigating matters that are time-related, enter a "show clock" command before each of your other commands.**

Gather enough information

Follow each possible cause logically and in detail. For example, consider an issue where the symptom is that a certain client never receives multicast streams. Possible causes include:

- the client is failing to send the IGMP report, or
- the client is sending the report, but it is being dropped by an intermediate switch, or
- the report is reaching the PIM router, but a multicast route is not being created, or...

To deal with this situation, you would follow each of these possibilities to confirm or deny it. Once you find the problem, you would capture counters, debug command output, ethereal traces, etc. to explore it in detail. For example, if you find that a particular switch in the chain is not seeing the IGMP report, you might find that a switch port error counter increments every time you send the IGMP report, indicating that reports are being corrupted.

Once you believe that you have successfully isolated the root cause of the problem, gather as much evidence as you can to support your theory.

Tip #6 **If you find a sequence of actions that make the problem happen, capture the full sequence (and the evidence that the problem has happened) multiple times-more than twice if possible.**

Often, certain pre-conditions determine whether the problem will or won't occur-for example, the problem still occurs if port 1.0.2 is disconnected, the problem doesn't happen if the ARP entry has been cleared before the route ages out, etc. It is important to capture output to illustrate these pre-conditions. It might be very clear to you that port 1.0.2 is definitely disconnected, but anyone analysing the captures after you cannot be certain of that unless they see output of the show interface brief command that shows port 1.0.2 as down.

Tip # 7 **Capture as much evidence as possible to support any assertions you make about the circumstances in which you see the problem happen.**

You can also re-run the command **show tech-support** after any significant event.

Annotate the output afterwards

If you need to send your capture off to the next layer of support or anyone else, you will want to give them as much guidance as possible as to how to extract the significant pieces of information from your capture. This can be achieved by going through the capture and putting in further comments to spell out more clearly what is happening than you were able to do with the quick comments you typed in while on-site. Or, you might want to copy the really important parts out of the capture and paste them into a separate file, to succinctly illustrate the core of the problem.

Tip #8 **Do some post-processing of the capture file, to save time and effort for anyone who is analysing it after you.**

But, even if you copy the pertinent pieces out of the capture, still send the rest of the capture file anyway. There could be significant gems of information in there that you weren't aware of.

Tip #9 **If possible, send everything that you captured - too much information is always better than too little.**

A structure for thinking about the network - the OSI model

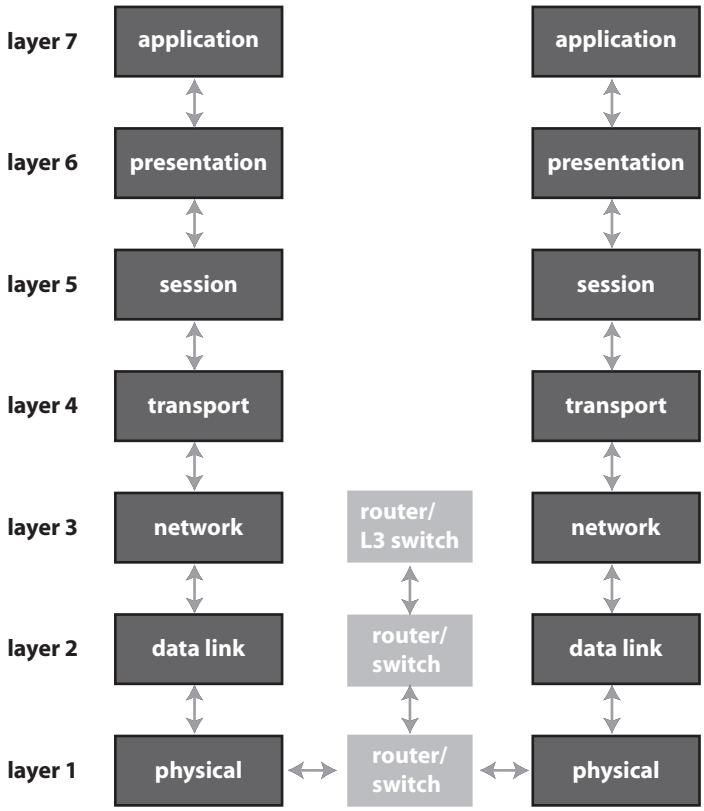
When investigating a network problem, your guiding aim is to understand the root cause of the problem, which might be:

- a cable plugged into the wrong socket
- a misconfiguration of some piece of equipment
- an interoperability problem between two pieces of equipment
- a hardware fault in some piece of equipment
- a bug in the software of some piece of equipment. Of course, it might not always be possible to find the root cause, but looking for it should be the central target of your debugging session.

It is necessary to have a framework within which to work, to provide you with a path to follow in your investigation.

Debugging by using the OSI model as a framework

A very useful framework to follow is the OSI 7-layer model. This model is a good guide to your investigation process when analysing problems, as it is the model that underlies the design of network protocols and equipment.



The following sections give some examples of how to look for problems at each of layers1-3. If you don't find a problem at one layer, then you can move on to the next layer, and look for problems there. Thereby you can move your investigation forward in a logical and methodical manner. This will help you to eliminate different factors that could be causing the problem, and so narrow down on where the problem does lie.

Layer 1 Physical layer: The connection between the routers or switches

At this layer, you can check that links are up. If links are not up, you can investigate why not.

1. Use the **show interface brief** command to see whether the interfaces are up or down.
2. Check the port LEDs to see if there is a connection.
3. Follow cables to check that they are correctly connected.
4. Check that the speed/duplex/autonegotiation configuration on connected ports are compatible.
5. Check that any pluggable components (GBICs, SFPs, XFPs) are compatible.
6. Draw a network diagram. Include the ports, the VLANs, the IP addresses, and how the devices are connected together.

Layer 2 Data link layer: MAC addresses, VLANs and Layer 2 protocols

At this layer, you can investigate whether MAC addresses are learnt in the hardware tables, whether clients are in the correct VLANs, and whether Layer 2 protocols are operating correctly (if they have been configured in that particular network). You can also look for packets being dropped by Layer 2 corruption-detection mechanisms (like FCS).

1. Use the command **show platform table fdb** to display the software Layer 2 forwarding database. Check that MAC addresses are learnt on the ports/VLANs that they should be.
2. Use the commands **show platform table macaddr**, **show platform table vlan**, and **show platform table port** to display the Layer 2 hardware tables. Check that the contents of those tables are as you expect they should be.
3. Use the command **show vlan all** to check the VLAN states and port memberships.
4. If spanning tree is configured, check that it is operating correctly. Has the correct switch been elected as root? Are the correct ports in forwarding/blocking?
5. If EPSR is configured, is the ring state being detected correctly? Are the ports in the correct forwarding/blocking states?
6. If LACP is in operation, is it operating as expected? Have the correct sets of ports been aggregated.
7. Troubleshoot Layer 2 multicasting (IGMP snooping):
    d Use the command **show ip igmp snooping statistics interface <int-name>**, to check that IGMP reports are being seen for the correct groups on the correct ports.

e Use the command **show platform table l2mc** to display the Layer 2 multicasting hardware table.

Layer 3 Network Layer: IP features (IP, ICMP, ARP, DHCP, IGMP, PIM, RIP, OSPF, etc.)

At this layer, you look for problems with Layer 3 connectivity. Are the correct ARPs learnt? Are the correct IP routes learnt? Are the multicast protocols operating correctly, etc.?

1. Check ARP entries and hardware tables:
   a Use the command **show arp** to see the ARP entries.
   b Use the command **show platform table ip** to display the Layer 3 IP hardware table.
2. Use traceroute and ping to check reachability and path.
3. Use the command **show ip route** to check the routes. Use the command **show platform table ip** to see whether the correct routes have been written into the hardware IP table.
4. Troubleshoot RIP/OSPF:
   a Check OSPF neighbor relationships; use the command show **ip ospf neighbor**. Are all the neighbors there?
   b Check the OSPF LSA database and see if LSAs are being received from all the routers in the OSPF area.
   c Look at RIP/OSPF statistics using the commend **show iprip interface** or **show ip OSPF interface**. Are there discards, or authentication failures?
   d Check whether the correct parameters are configured for redistributing static, interface and other routing protocol routes.
5. When the switch is acting as the DHCP server, troubleshoot DHCP:
   a Use the command **show ip dhcp binding** to see which clients the server knows about.
   b Use the command **show ip dhcp server statistics** to see how many DHCP messages the server has sent and received.
   c Get an Ethereal capture from the client when requesting an IP address from the DHCP server.
6. Troubleshoot Layer 3 multicasting (IGMP and PIM Sparse Mode):
   a Use the command **show ip igmp groups** to see a list of the IGMP groups that the switch knows about and the port each group is reached through.
   b Use the command **show ip igmp interface** to see the state and timer values for IGMP on each port and VLAN.
   c Use the commands **show ip pim sparse-mode rp-hash** to see the Rendezvous Point and **show ip pim sparse-mode neighbor** to see the neighbors.
   d Use the command **show ip mroute** to display the multicast routing table.
   e Use the command **show platform table ipmulti** to display the Layer 3 multicasting hardware table.

Troubleshooting some common network problems

In this section we will consider a set of the most common networking problems, and discuss how to troubleshoot them.

Link will not come up

You cable up a connection between two switches, and the ports at each of the cable do not come up. Or, maybe the port at one end comes up, but the other end does not.

The most common reasons for this problem are:

- the cable is faulty.
- the SFPs (if SFPs are being used) at each end of the link are incompatible. One might be LX and the other SX, or one is for 100Mbps and the other for 1Gps, etc.
- the configurations applied to the ports are incompatible - one might be fixed to 100Mbps and the other to 1Gbps.
- one or other port is faulty.

Use logical troubleshooting to step you through each of these probable causes.

Try some different cables

If the problem persists, carefully check the capabilities of the SFPs (if SFPs are being used). Try a different SFP (in case one of the SFPs is faulty).

If the problem does not lie with SFPs, carefully check the configuration on the ports. If in doubt, set the ports to both be autonegotiating, and see if that brings up the link. If the link comes up with both ends autonegotiating, but not with the original config, then the problem could lie with the config - try slight variations of the configs, to see if it is possible to narrow down a set of configs that work, and a set that don't.

Try some different ports at each end

If the problem does not seem to be with configuration, then examine the possibility of a faulty port. Try some different ports at each end, to see if the link comes up when a different port is used.

If none of the above techniques is able to indicate where the problem lies, and the issue is going to be escalated, then some very useful information to accompany the escalation is the status of the PHY registers at each end of the link. The PHY is the hardware component that is involved in link negotiation, and the registers within this hardware component embody the state of the link negotiation. A snapshot of the registers in the PHY chip for a given port can be captured with the command **show platform table port**. An ongoing debug output of changes to these registers as negotiation proceeds can be enabled with the command **debug platform driver phy**.

Two devices in the network can't ping each other

The failure of Layer 3 communication between two devices in the network is most commonly reported as the devices being unable to ping each other.

The problem could be anywhere from the physical layer through to the network layer. The most common reasons for this problem are:

- A physical link failure somewhere in the data path between the devices.
- A misconfiguration at the VLAN or IP level somewhere along the data path between the devices.
- Incorrect cabling (something plugged into the wrong port) somewhere along the data path between the devices.
- Missing route information somewhere along the data path between the devices.

Traceroutes

The best way to start debugging this problem is to use traceroute to narrow down the point in the data path where the problem lies.

Traceroute works by increasing the **time-to-live** (TTL) value of each successive batch of packets sent. The first three packets sent have a TTL value of one (implying that they are not forwarded by the next router and make only a single hop). The next three packets have a TTL value of 2, and so on. When a packet passes through a host, normally the host decrements the TTL value by one, and forwards the packet to the next host. When a packet with a TTL of one reaches a host, the host discards the packet and sends an ICMP time exceeded (type 11) packet to the sender. The traceroute utility uses these returning packets to produce a list of hosts that the packets have traversed on route to the destination.

Traceroutes from each of the devices toward each other should show that traffic is forwarded OK for a certain number of hops at each end. The points identified from each

end which the traceroutes show the data to be lost or misdirected, are the places to start looking for the problem. It could be that the problem location identifies by the traces from each end turns out to be one common location, or it could be that there are problems occurring at multiple locations.

Once the traceroute has pointed you to where to start investigating, the debugging process should work through the possible causes listed above.

Cable problems

Check for cabling problems (links down) in that location.

If the links are all OK, start doing pings or traceroutes from the switch(es) at the problem location towards the end-device that is not reachable from that location. Also, perform pings and traceroutes to other switches along the path towards the unreachable device. Look at where the ping and trace-route traffic is directed.

- is a misconfiguration causing the switch to send the traffic out an incorrect port?
- is the egress VLAN not configured on the correct ports?
- is the IP address on the egress VLAN incorrect?

If the configurations at each hop look correct, then check whether a cabling mistake is actually taking traffic to an incorrect destination.

Routing problems

The problem could very possibly not be a misconfiguration of VLANs and IP interfaces, but could be due to a failure to transfer route information between the switches. If the switch where the problem occurs is reporting that it does not have a route to the destination device, then examine why it does not have this route. Was it a failure to configure a static route on the switch? Is it a failure of another switch to advertise the route to the switch in question?

One possible problem is that a switch does have a route to the destination in software, but the route has not been written into hardware. In this case, the problem switch would be able to ping the destination device, but the switch one hop before the problem switch would not be able to ping the destination device, even though both switches have correct routes into their IP route tables.

If the problem needs to be escalated, then the most important information to capture on the switch(es) at the problem location is the output of **show tech** - it will contain the full set of information to illustrate the state of IP routing on the switch, at software and hardware levels.

Switch reports high CPU utilisation

In normal circumstances, the CPU on a hardware-based switch should not be busy. All the packet-forwarding work is done by the hardware switching chip; the CPU is only used for control-plane activities. The CPU is only performing activities like sending/receiving/processing routing protocol packets; sending/receiving/processing ARPs; maintaining hardware tables, processing management sessions, etc.

If the CPU utilisation on a switch is consistently high (50% or more) then this should be investigated, so that the reason for the high utilisation can be understood and, if necessary, contributing problems resolved.

Sometimes the high CPU utilisation can be due to the switch being required to carry out an unnecessarily high level of 'normal' activity. For example, if the switch has a very large route table (1000+ routes) and is using RIP as its routing protocol, that can put a very high load on the cpu. Or, it could be that some monitoring process in the network is sending a very high level of SNMP requests to the switch.

Find the highly utilised CPU process

To get an idea of which process is using most of the CPU, use the command **show cpu**. This command will display the CPU usage for each process running in the switch.

```
CPU averages:
 1 second: 10%, 20 seconds: 1%, 60 seconds: 1%
System load averages:
 1 minute: 0.00, 5 minutes: 0.20, 15 minutes: 0.18
Current CPU load:
 userspace: 6%, kernel: 3%, interrupts: 0% iowaits: 0%

user processes
==============
  pid name            thrds  cpu%   pri state sleep% runtime
 1343 hostd              1   6.3    20   run     0    39
 1127 hsl               14   5.4    20 sleep     0  2162
 1194 corosync          47   1.8    -2 sleep     0   538
    1 init               1   0.0    20 sleep     0    64
 2607 sh                 1   0.0    20 sleep     0     0
 2608 corerotate         1   0.0    20 sleep     0     0
  796 syslog-ng          1   0.0    20 sleep     0   240
  802 klogd              1   0.0    20 sleep     0     2
  855 inetd              1   0.0    20 sleep     0     1
  865 portmap            1   0.0    20 sleep     0     0
  876 crond              1   0.0    20 sleep     0     0
  962 automount          1   0.0    20 sleep     0    32
 1040 openhpid          10   0.0    20 sleep     0   189
 1062 hpilogd            1   0.0    20 sleep     0     0
 1125 stackd             1   0.0    20 sleep     0    14
 1257 authd              1   0.0    20 sleep     0    43
 1276 bgpd               1   0.0    20 sleep     0    63
 1295 cntrd              1   0.0    20 sleep     0     3
 1321 epsrd              1   0.0    20 sleep     0    32
 1423 imi                1   0.0    20 sleep     0   169
 1448 irdpd              1   0.0    20 sleep     0    37
 1473 lacpd              1   0.0    20 sleep     0    44
```

```
 1501 mstpd               1   0.0    20 sleep     0    48
 1540 nsm                 1   0.0    20 sleep     0   107
 1560 ospfd               1   0.0    20 sleep     0    50
 1581 pdmd                1   0.0    20 sleep     0    36
 1601 pimd                1   0.0    20 sleep     0    45
 1620 ripd                1   0.0    20 sleep     0    48
 1641 ripngd              1   0.0    20 sleep     0    39
 1659 rmond               1   0.0    20 sleep     0    38
 1679 snmpd               1   0.0    20 sleep     0    57
 1699 vrrpd               1   0.0    20 sleep     0    38
 2052 ntpd                1   0.0    20 sleep     0     5
 2158 atlgetty            1   0.0    20 sleep     0     0
 2159 login               1   0.0    20 sleep     0     6
2719 more                1   0.0    20 sleep     0     0
 2718 imish               1   0.0    20 sleep     0     0
 2371 imish               1   0.0    20 sleep     0    23
kernel threads
==============
  pid name             cpu%   pri state sleep% runtime
   79 aio/0             0.0    15 sleep     0     0
    5 events/0          0.0    15 sleep     0     4
   49 kblockd/0         0.0    15 sleep     0     0
    7 khelper           0.0    15 sleep     0     5
  683 kmmcd             0.0    15 sleep     0     0
    3 ksoftirqd/0       0.0    15 sleep     0     0
   78 kswapd0           0.0    15 sleep     0     0
    2 kthreadd          0.0    15 sleep     0     0
    6 linkwatch         0.0    15 sleep     0     0
  980 loop0             0.0     0 sleep     0    45
  689 mpc83xx_spi.0     0.0    15 sleep     0     0
  667 mtdblockd         0.0    15 sleep     0    23
   76 pdflush           0.0    20 sleep     0     0
  726 rpciod/0          0.0    15 sleep     0     0
    4 watchdog/0        0.0  -100 sleep     0     0
  763 jffs2_gcd_mtd0    0.0    30 sleep     0     6
```

Some of the processes that are listed in this output are clearly related to network protocols (bgpd, ripd, sshd, etc.) and some are underlying processes within the operating system (exfx, aisexec, rpc.mountd etc.). If the high level of CPU usage is reported against one of the network protocol related processes, then that might be enough to point you quite quickly to the cause of the problem - a protocol that has been configured to do too much work: logging configured to send log messages to too many syslog hosts, or **rmon** configured with too many probes, etc.

Look for a high level of unnecessary traffic coming up to the CPU

However, most of the time, the investigation will need to continue on to the traffic investigation phase. The fact is that most often the cause of high CPU utilisation on a switch is simply that a lot of packets are coming up to the CPU.

For example, if you see a high CPU utilisation reported for lacpd, it could well indicate that the switch is receiving a high rate of LACP packets for some reason.

Most commonly the process that will be reporting a high CPU utilisation is EXFX or hostd. Which just indicates that a high rate of packets are coming up from the switch chip.

To get a snapshot of the packets coming up to the CPU, use the command:

```
debug platform packet timeout 2
```

That will give you a 2 second snapshot of all the packets coming up to the CPU. Note that it might take well more than 2 seconds (quite possibly some minutes) for the debug output to complete.

The next step is to investigate this debug capture to determine what the bulk of the traffic is, and where it is coming from. The clues provided by the **show cpu** output can be very valuable here. For example, If the process that was reporting high CPU was mstpd, then you should start by looking for STP packets in the debug capture.

If the bulk of the packets in the capture are IP packets or ARP, then you can obtain a more decoded, and finer-grained packet trace by using the command:

```
debug ip packet interface {<interface-name>|all}[address <ip
   address>|verbose|hex|arp|udp|tcp|icmp]
```

This command operates in a manner that is very similar to the popular **tcpdump** utility that is available on numerous computing platforms. With the help of this command, you can obtain a lot of valuable troubleshooting information:

- A textual decoding of the packets.
- By specifying different interfaces in the command, you can narrow down the ingress interface of the heavy packet streams.
- By specifying protocols (ARP, TCP, UDP, ICMP) you can quickly narrow down on the traffic type.

Having identified the traffic type (IP or other) that is over-utilising the CPU, the next step is to work out where it is coming from, and resolve whatever network problem that is causing that unnecessary traffic to be sent to the CPU of the switch.

It might be that some rouge device on the network is flooding out packets that it should not be. It might be that a device is sending a valid data stream that the switch should be hardware switching, but is unable to hardware switch due to a lack of a route or lack of a nexthop ARP entry.

Network slowdown

Typically, slow performance is not experienced across a whole network. Rather, most often what happens is that some or all workstations on the network experience slow performance of applications interacting with one, or a set, of network servers.

Before looking into the network switches for the cause of the slowdown, it is important to first verify that the slow performance is not due to resource overload on the servers or workstations themselves.

If there is good evidence that the cause of the slow performance lies in the switching infrastructure, rather than the workstations or servers, then the next step in troubleshooting is to identify one or more network paths that are experiencing frequent occasions of slow performance. Identify particular workstations and particular servers between which communication is frequently slow, and map out the network path between them - i.e. which ports of which switches do the packets exchanged between the workstation and server pass through?

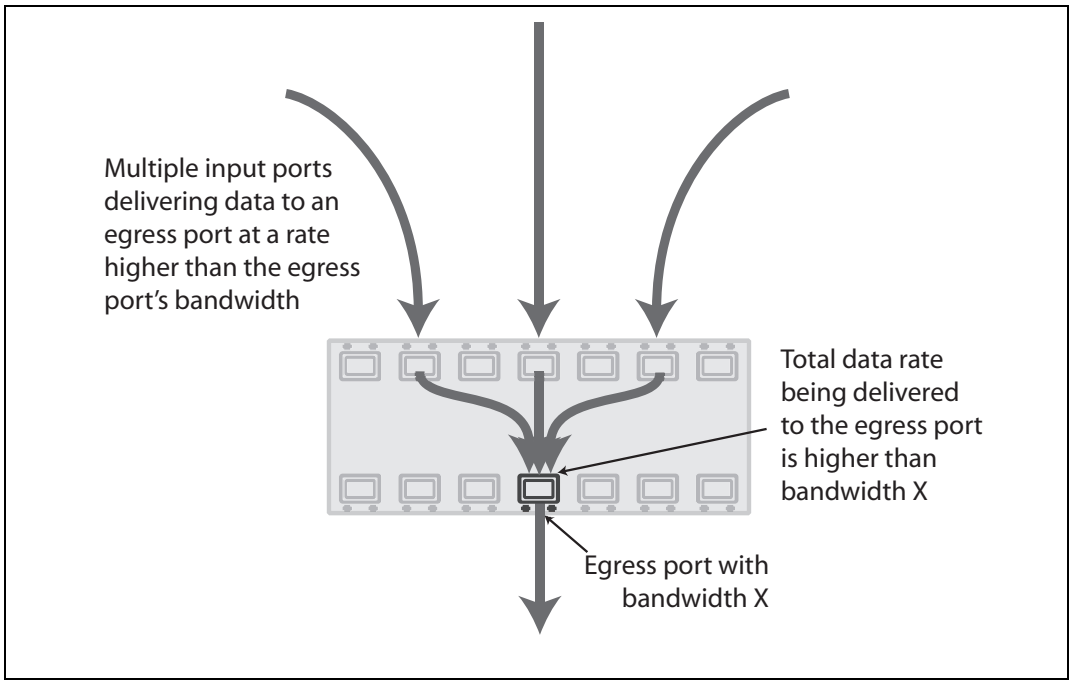
Then, work along the identified network path(s) looking for evidence of congestion, corruption, or collisions, as described in this guide.

Congestion - oversubscribed ports

The first task in debugging a network slow down is to look for ports that are oversubscribed.

Oversubscribed ports

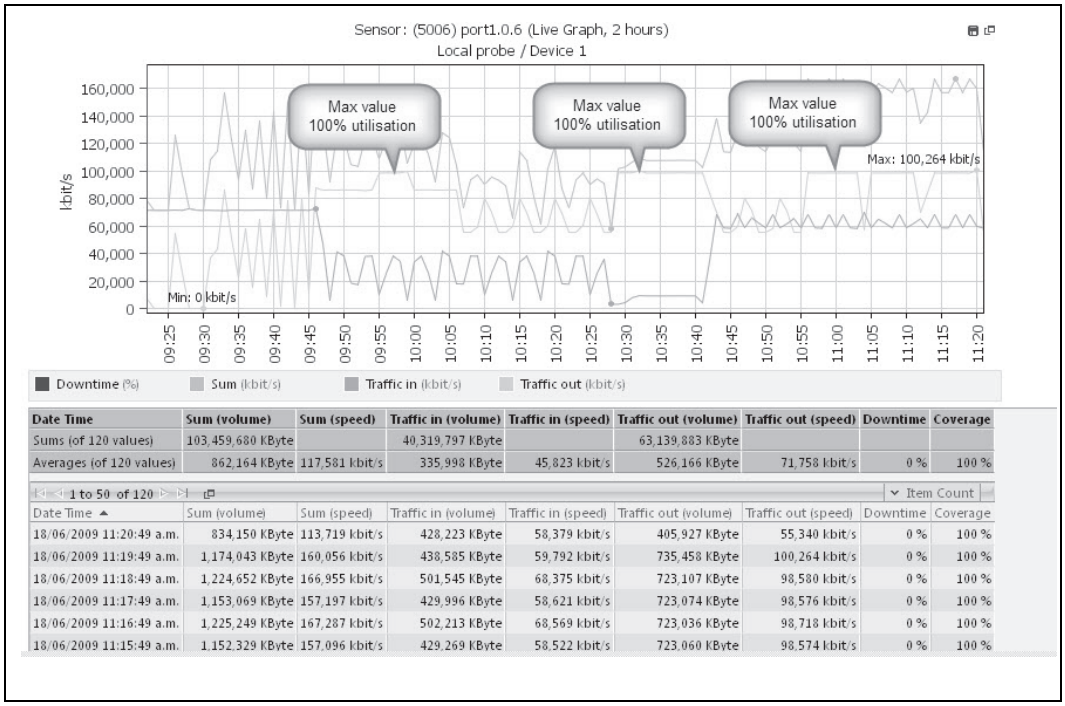
An oversubscribed port is one from which the switch is attempting to deliver data at a rate higher than the port's bandwidth can allow.



When this occurs, some packets must end up being dropped. The users whose packets are being dropped will experience in a slow-down in their file transfers or in the responsiveness of the network-based applications they are running.

An important step in diagnosing network slowdowns is to look for oversubscribed ports in the network. One way to find oversubscribed ports is to use a tool that will report the utilisation of network ports. A variety of such tools exist, most network management packages provide facilities for monitoring port utilisation. The popular traffic graphing tool MRTG is widely used for the task of port utilisation monitoring and graphing. Other commercial tools, similar to MRTG, e.g. PRTG, provide similar port utilisation monitoring and graphing capabilities.

Typically these tools will provide graphs of utilisation over time, similar to the one below:



Date Time	Sum (volume)	Sum (speed)	Traffic in (volume)	Traffic in (speed)	Traffic out (volume)	Traffic out (speed)	Downtime	Coverage
Sums (of 120 values)	103,459,680 KByte		40,319,797 KByte		63,139,883 KByte			
Averages (of 120 values)	862,164 KByte	117,581 kbit/s	335,998 KByte	45,823 kbit/s	526,166 KByte	71,758 kbit/s	0 %	100 %

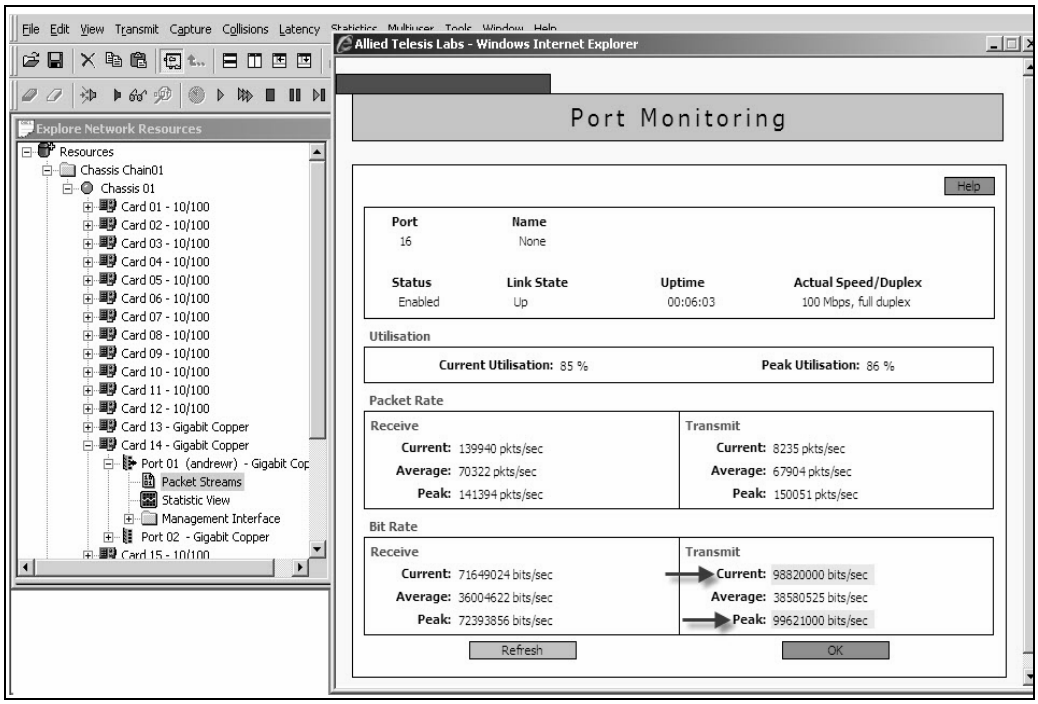
1 to 50 of 120 — Item Count

Date Time	Sum (volume)	Sum (speed)	Traffic in (volume)	Traffic in (speed)	Traffic out (volume)	Traffic out (speed)	Downtime	Coverage
18/06/2009 11:20:49 a.m.	834,150 KByte	113,719 kbit/s	428,223 KByte	58,379 kbit/s	405,927 KByte	55,340 kbit/s	0 %	100 %
18/06/2009 11:19:49 a.m.	1,174,043 KByte	160,056 kbit/s	438,585 KByte	59,792 kbit/s	735,458 KByte	100,264 kbit/s	0 %	100 %
18/06/2009 11:18:49 a.m.	1,224,652 KByte	166,955 kbit/s	501,545 KByte	68,375 kbit/s	723,107 KByte	98,580 kbit/s	0 %	100 %
18/06/2009 11:17:49 a.m.	1,153,069 KByte	157,197 kbit/s	429,996 KByte	58,621 kbit/s	723,074 KByte	98,576 kbit/s	0 %	100 %
18/06/2009 11:16:49 a.m.	1,225,249 KByte	167,287 kbit/s	502,213 KByte	68,569 kbit/s	723,036 KByte	98,718 kbit/s	0 %	100 %
18/06/2009 11:15:49 a.m.	1,152,329 KByte	157,096 kbit/s	429,269 KByte	58,522 kbit/s	723,060 KByte	98,574 kbit/s	0 %	100 %

Periods during which the port is at 100% utilisation are sure signs of the port being oversubscribed.

Using the GUI to monitor port data transmission

Port data transmission rates can also be monitored through the switch's GUI interfaces. If the transmit rate is frequently close to the port's maximum bandwidth, then it is highly likely that congestion is occurring on the port. Note the highlighted **Current** and **Peak** values shown for port 16 in the diagram below.



Checking the state of egress port queues

Another effective way to find ports that are oversubscribed is to look at the state of each switch port's egress queues. If there are packets queued up in a port's egress queues, then the port is oversubscribed.

On x900 and x908 switches, in order to see the state of port egress queues, it is necessary to enable Quality of Service (QoS) counters, using the command:

```
platform enhancedmode {nexthop|qoscounters}
```

followed by a reboot of the switch.

Then, after the switch has rebooted, the egress queue counters can be viewed with the following command:

```
show mls qos interface <interface name> queue-counters
```

This will output a snapshot of the state of the egress queues on the specified port. If some queues regularly have non-zero values for a given port, then that port is oversubscribed.

Data Corruption

If packets are being corrupted, by faulty cabling, electrical interference, or by switch hardware faults, then the corrupted packets will be dropped by the receiving switch. A high rate of corrupt packets will cause a slow down in network performance, as the hosts sending the dropped packets will have to resend them.

To look for evidence of a port receiving corrupt packets, use the command:

```
show platform table port <interface name> counters
```

For example:

```
show platform table port port1.0.18 counters
```

Look for non-zero error counters in the **Receive** column.

If you find a port that is receiving a large rate of errors, as shown in bold in the output below, then that will indicate that either the cable connected to the port is faulty, or the port on the other end of the cable is faulty.

```
 Triple-Auth#sh plat table port port1.0.13 count

 Switch Port Counters
--------------------------------------------------------------------------------

Port port1.0.13 Ethernet MAC counters:
 Combined receive/transmit packets by size (octets) counters:
  64                          29293 1024 - MaxPktSz              4713
  65 - 127                   434884 1519 - 1522                    0
  128 - 255                   66732 1519 - 2047                    0
  256 - 511                    2282 2048 - 4095                    0
  512 - 1023                    248 4096 - 9216                    0

 General Counters:
 Receive                            Transmit
  Octets                   35868103 Octets                  25871611
  Pkts                       323030 Pkts                      215122
  FCSErrors                       0
  MulticastPkts               48940 MulticastPkts                 4
  BroadcastPkts               21054 BroadcastPkts              1309
  PauseMACCtlFrms                 0 PauseMACCtlFrms                0
  OversizePkts                    0
  Fragments                       3
  Jabbers                        21
  UnsupportOpcode                 0
  AlignmentErrors               193
  SymErDurCarrier                 0
  CarrierSenseErr                 0
  UndersizePkts                 743
                                     FrameWDeferrdTx               0
                                     FrmWExcesDefer                0
                                     SingleCollsnFrm               0
                                     MultCollsnFrm                 0
                                     LateCollisions                0
                                     ExcessivCollsns               0
                                     Collisions                    0

 Layer 3 Counters:
  ifInUcastPkts               253036 ifOutUcastPkts            213809
  ifInDiscards                 39423 ifOutErrors                    0
  ipInHdrErrors                    6

 Miscellaneous Counters:
  DropEvents                      0
  ifOutDiscards                   0
  MTUExcdDiscard                  0

--------------------------------------------------------------------------------
```

Collisions

One special case to consider is when a port is reporting a high rate of collisions. On full-duplex ports, there should **never** be any collisions, as sent and received packets are exchanged in different channels, and cannot collide with each other.

If a port is reporting a high rate of collisions, it is highly likely that the port is in half-duplex mode, and the port on the switch at the other end of the link is in full-duplex mode.

As a result, the switch in full-duplex mode will send data even while the half-duplex port is transmitting (which is normal behaviour for a full-duplex port). The half-duplex port, though, will see this situation as causing collisions, and will frequently abort packet sending, and retry. This can significantly reduce throughput. This duplex mismatch commonly occurs if one end of the link has a fixed speed/duplex, and the port at the other end is autonegotiating. The autonegotiating port will not be able to detect the duplex state of the peer port, and will default to half duplex.

The solution to the duplex mismatch problem is to **change the configuration of the ports** to ensure that they will come up with the same duplex state at both ends.

Checking for collisions

Use the following command to check for collisions:

```
show platform table port <port-number> count
```

The command will output a table of counters for a port, among the counters are the number of collisions that have occurred on the port - they are shown in bold in the example below:

```
awplus#show platform table port port1.0.13 count
Switch Port Counters
------------------------------------------------------------------------

Port port1.0.13 Ethernet MAC counters:
 Combined receive/transmit packets by size (octets) counters:
  64                          1084 1024 - MaxPktSz              7040
  65 - 127                    2773 1519 - 1522                     0
  128 - 255                    162 1519 - 2047                     0
  256 - 511                     10 2048 - 4095                     0
  512 - 1023                   106 4096 - 9216                     0
 General Counters:
 Receive                           Transmit
  Octets                 10820753 Octets                     220353
  Pkts                       9484 Pkts                         1691
  UndersizePkts                 0

 <Some output removed for brevity>
  FrameWDeferrdTx               2
                                   FrmWExcesDefer                 0
                                   SingleCollsnFrm                72943
                                   MultCollsnFrm                  24527
                                   LateCollisions                 2694
                                   ExcessivCollsns                571
                                   Collisions                     951313
 Layer 3 Counters:
  ifInUcastPkts              7254 ifOutUcastPkts                490
  ifInDiscards               2147 ifOutErrors                     0
  ipInHdrErrors                 0
```

The definitions of the collision-related counters output by this command are:

Collisions - This is a counter of all collisions that have occurred on that port. A collision being a case of a half-duplex interface detecting an incoming packet at the time it was trying to transmit a packet.

Single Collision Frames, Multiple Collision Frames - These counters indicate how many times the port has experienced a single collision when attempting to transmit a given frame, and how many times it experienced multiple collisions whilst attempting to deliver a given frame.

Late Collision - A late collision occurs when the switch detects an incoming packet after it has already transmitted 64 bytes of its current outgoing packet. In a properly constructed Ethernet network this should never happen, as other hosts on the segment should not start transmitting when one host has already been transmitting for 64-bytes worth of time. However, in cases of a duplex mismatch, late collisions are highly likely, as a port at one end of the link is operating at full duplex. The full-duplex port has no problem with transmitting at the same time as the port at the other end of the link is transmitting, as it considers each switch's transmissions to be in separate logical channels, and therefore unable to collide. The port operating in half-duplex, however, considers both switches to be transmitting in the same logical channel. So if it is well advanced in transmitting a packet when the peer switch starts a transmission, it will experience a late collision.

Excessive Collisions - Each time a port experiences a collision when attempting to transmit, it will pause, and then try again. If the port has 16 attempts to transmit a packet and each results in a collision, then the port increments the ExcessiveCollisions counter, and gives up attempting to deliver the packet. Again, in a case of duplex

mismatch, if the port operating in full duplex is transmitting at a high data rate, it is possible for the port operating in half duplex to experience cases of excessive collisions.

A high rate of STP Topology Change Notifications

There is another relatively common cause of intermittent performance problems in networks that are using spanning tree (any type of spanning tree - STP, RSTP or MSTP) to control redundant network paths. When an active STP port changes link status, the switch will send a Topology Change Notification (TCN) to the other switches in the spanning tree. When those other switches receive the TCNs, they need to flush all or part of their MAC and ARP tables.

Then, of course, those entries need to be re learnt. During the relearning period, all switches in the network will flood almost the packets they receive, as though they were hubs. This will cause a period of congestion, and quite possibly high CPU utilisation on workstations and servers, as they experience a significantly increased rate of packet reception (most of which are not actually intended for them). This all causes a brief slow down of network performance.

If the topology changes keep happening on the network with some frequency for an extended period, then the slow down of network performance will become quite noticeable.

This very problem of frequent topology changes over an extended period is often caused by edge ports on the network not being configured as **portfast**. Edge ports are those ports that connect to workstations or servers, as against those ports that connect to other switches. In normal circumstances, these ports are effectively 'leaves' on the spanning tree, as they connect to end-point devices. Because of this, the spanning tree does not need to be informed about state changes of these ports, as those state changes do not result in changes to the internal topology of the spanning tree.

When portfast is configured on an edge port, then the turning on or off, or rebooting, of the PC connected to the port will not result in TCNs. However, a failure to configure portfast on the edge ports of the network will result in TCNs being generated, and propagated through the whole spanning tree, every time those ports change state. Consequently, at times of the day when many PCs are being turned on or off there will be a noticeable slowing of the network.

Therefore, one avenue of investigation of network slow downs (particularly slow downs at the start and end of the working day) is to check whether the edge ports on the network have been configured for **portfast**.

Other points to note in relation to network slowdowns

High CPU utilisation on switches does not cause network slowdowns

It is important to keep in mind that Ethernet switches forward packets in their switching ASICs. Although the CPU is involved in processing network control protocols, and populating the ASIC forwarding tables, the **actual packet forwarding is performed by the ASICs**.

In fact, the rate at which a switch forwards packets is quite unrelated to how busy its CPU is.

Despite that, it is not uncommon for a switch's CPU to experience high utilisation as a consequence of something that is causing network congestion. In particular, if there is a high level of broadcast packets flowing in the network, that could result in network congestion, and also result in high utilisation of switches' CPUs, as broadcast packets are all sent to the CPU, as well as forwarded.

Whilst switches with high CPU utilisation are not a cause of a network slowdown, the occurrence of high CPU utilisation on the switches at the same time as a network slowdown is well worth investigating, as it could provide important clues as to the actual cause of the slowdown.

In particular, if the high CPU utilisation is (as is most likely) being caused by a high rate of data coming up to the switches' CPUs, then identifying this CPU-directed data could help in identifying what data is causing congestion in the network.

Identifying data causing congestion in the network

Allied Telesis x-series switches provide a variety of methods and commands for obtaining a capture of the packets arriving at the CPU. Here are two useful commands:

Packet snapshot

To get a snapshot of the packets coming up to the CPU, use the command:

```
debug platform packet timeout 2
```

This will give you a 2 second snapshot of all the packets coming up to the CPU. Note that it might take well more than 2 seconds, quite possibly some minutes, for the debug output to complete.

Identify traffic type

The next step is to investigate this debug capture to determine what the bulk of the traffic is, and where it is coming from. If the bulk of the packets in the capture are IP packets or ARP, then you can obtain a more decoded, and finer-grained packet trace by using the command:

```
debug ip packet interface {<interface-name>|all}[address
  <ipaddress>|verbose|hex|arp|udp|tcp|icmp]
```

This command will generate a trace of the packets that are entering and exiting the switch's CPU. With the help of this command, you can obtain a lot of valuable troubleshooting information:

- You can get a textual decoding of the packets.
- If the option 'verbose' is specified, then the debug indicates which interface the packets are arriving/leaving on, so you can determine the ingress interface of the heavy packet streams.
- By specifying protocols (ARP, TCP, UDP, ICMP), you can quickly narrow down on the traffic type.
- Having identified the traffic type (IP or other) that is over-utilising the CPU, the next step is to work out where it is coming from, and this may lead you to whatever network problem that is causing that unnecessary traffic to be sent to the CPU of the switch, and probably also causing network congestion.

Ping can be a useful tool in monitoring for network slowdown

Ping is a simple, effective, way to monitor the round-trip time along particular network paths. By pinging between various pairs of points in the network, it is often possible to identify the point(s) in the network at which packets are being queued or dropped.

But, it is important to have a clear understanding of the best way to make use of ping.

Simply looking at the average round-trip-time of the pings between one device and another does not necessarily provide very useful information about network performance. Typically, the bulk of the round-trip-time of a ping is not due to switch forwarding latency. Rather the bulk of the time is due to the software processing of the ping packet in the sending and receiving devices. **Different networking devices process ping packets differently** - some will treat ping packets as high priority and respond to them very quickly; others will not give such high priority to ping, which will result in a longer round-trip-time.

Consistently high round-trip time for pings to an Ethernet switch does not provide any evidence that the switch is experiencing congestion, or even that its CPU is heavily loaded, it just indicates that this switch does not have a fast turn-around of ping packets.

When using ping to help investigate network slow-downs, it is far more important to look for significant:

- **variations** in ping round-trip times along a given network path
- numbers of pings on a given network path getting **no response**

A significant variation of ping round-trip times, or intermittent loss of ping responses, could be an indicator of network congestion.

Packets sitting in egress queues when ports are oversubscribed can cause variation in ping round-trip times. Similarly, when oversubscribed ports have to drop packets, that can cause intermittent loss of ping responses.

However, be aware that significant variation of ping round-trip times, or intermittent loss of ping responses are not highly reliable indicators of network congestion. These variations could just as easily be caused by variable CPU loading of the target device.

While ping is useful as a simple way to monitor whether there is variation in the loading on the network or the network's hosts, it is too simple to be a useful diagnostic tool.

Once you have reason to believe that there are over-loaded or slow-performing paths in your network, then further investigation needs to gather more precise information than can be gained from ping. You need to move on to looking at port utilisations, packet error counters, etc., as have been described in this document.

Broadcast storm - how to identify and track port counters

One of the most debilitating problems that can happen on a network is a storm. The typical symptom of a storm is that 'the whole network stops'. Nobody on the network can communicate with anyone else on the network.

It can also be a very hard problem to resolve quickly. The common tools for network troubleshooting are often not much help in a storm. LEDs don't really help a lot - almost all the LEDs on all the switches are flashing at an enormous rate. Packet debugging on a switch, or sniffing using ethereal is not much help - the packet traces just show enormous numbers of broadcast packets flying around.

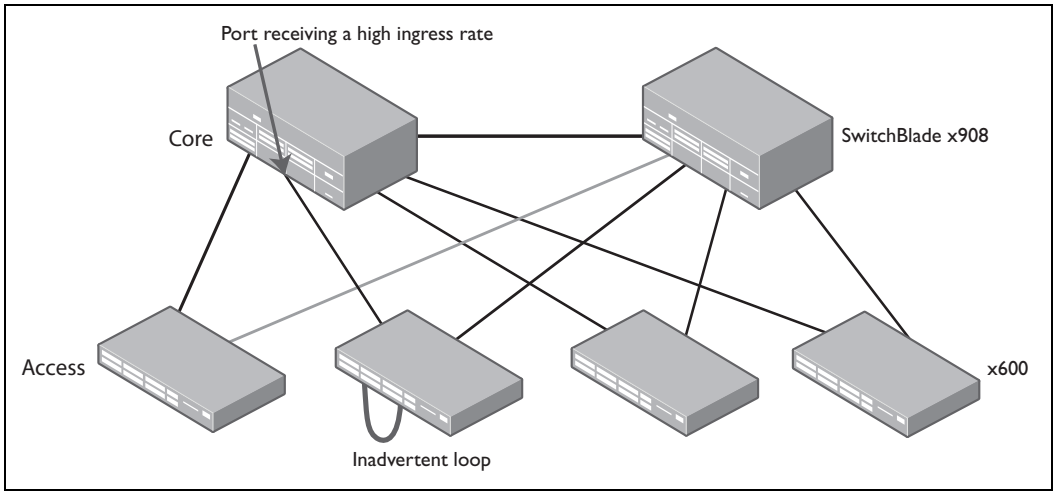
Faced with this mayhem, people will often pull out links and reboot switches until the network finally settles down. Unfortunately, this approach is not necessarily able to achieve a quick resolution, and might result in only a temporary resolution.

The **most effective** way to resolve a network storm is to use **port counters** to lead you to the cause of the storm, as will be described below.

Before looking at the troubleshooting process, let us first consider the main causes of broadcast storms. In fact, there are two main causes:

- An inadvertent loop caused by a cabling mistake
- Failure of the protocol protecting a deliberate resilient loop

In each case, the problem is that there is an unprotected loop in the network, and you need to find where that loop is, and resolve its problem. First, consider the case where an inadvertent loop has been created between two unprotected ports somewhere on the network.



In this case, packets sent down in the direction of that unprotected loop will be sent around the loop and come back up to the core. On the assumption that loop protection is working correctly between the core and aggregation layers, the looped data will only arrive back to the core on a single port.

To find the location of the inadvertent loop, you need to look at the ingress counters on the ports of the core switches, using the command **show platform table port counters**, that creates an output of the form:

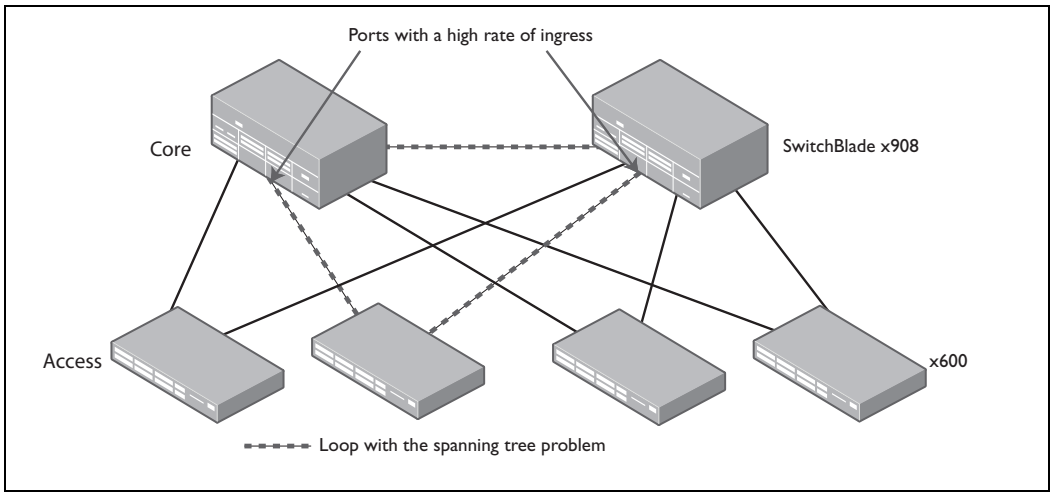
```
Switch Port Counters
------------------------------------------------------------------------------

Port 1.0.2    Ethernet MAC counters:
 Combined receive/transmit packets by size (octets) counters:
  64                          6917797 512 - 1023                    1748145
  65 - 127                   27824406 1024 - MaxPktSz               2269133
  128 - 255                  11721695
  256 - 511                   1789197

 General Counters:
 Receive                             Transmit
  Octets                  2314597957 Octets                   7891231403
  Pkts                      18157119 Pkts                       34113254
  CRCErrors                       0
  MulticastPkts              441051 MulticastPkts               1470792
  BroadcastPkts              669415 BroadcastPkts               8951147
  FlowCtrlFrms                    0 FlowCtrlFrms                      0
  OversizePkts                    0
  Fragments                       0
  Jabbers                         0
  UpsupportOpcode                 0
  UndersizePkts                   0
                                    Collisions                        0
                                    LateCollisions                    0
                                    ExcessivCollsns                   0

 Miscellaneous Counters:
  MAC TxErr                       0
  MAC RxErr                       0
  Drop Events                     0
```

The counter to look at is the **BroadcastPkts** in the Receive column; the item that is bold and underlined in the example above.

Enter the command a few times in a row, and see which port has the **Received BroadcastPkts** counter incrementing quickly. This will be the port to follow in order to find where the inadvertent loop lies. Move to the switch at the other end of the link that is connected to this port. Then, use the command **show platform table port counters** on that switch to find which port(s) are receiving a high rate of broadcast. If this switch is the edge switch, then there should, in fact be two ports that are receiving a high rate of broadcasts - they will be the ports connected to the loop. If this switch is not the edge switch, then move to the switch at the other end of the link connected to the broadcast-receiving port, and continue the process until you reach the edge switch.

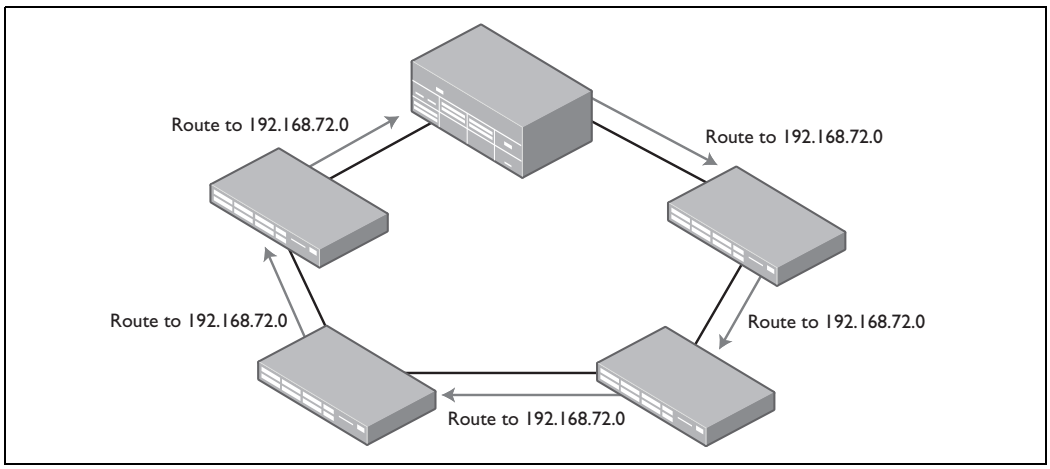
For the case where the mechanism for protecting a redundant link has failed to work, the process is only slightly different. In this case, the unprotected loop could be connected directly to the core switches, in which case, two of the ports that connect the core switches will have a high rate of ingress broadcasts. In fact, all the ports connected to the dotted lines in the diagram below will have a high rate of ingress broadcasts, but in terms of working out the direction to follow in order to find the source of the problem, the two ports indicated by arrows in the diagram are the ones that provide the most useful clue. But, if the unprotected loop is further away from the core, you simply need to follow the same process as above - finding the port that is receiving a high rate of broadcasts, moving to the switch at the other end of that link, and so on until you find the place where the packets are going around a loop.

Having found the problem loop, the task is then a matter of stopping the storm by unplugging a link, and then working out why the loop was unprotected.



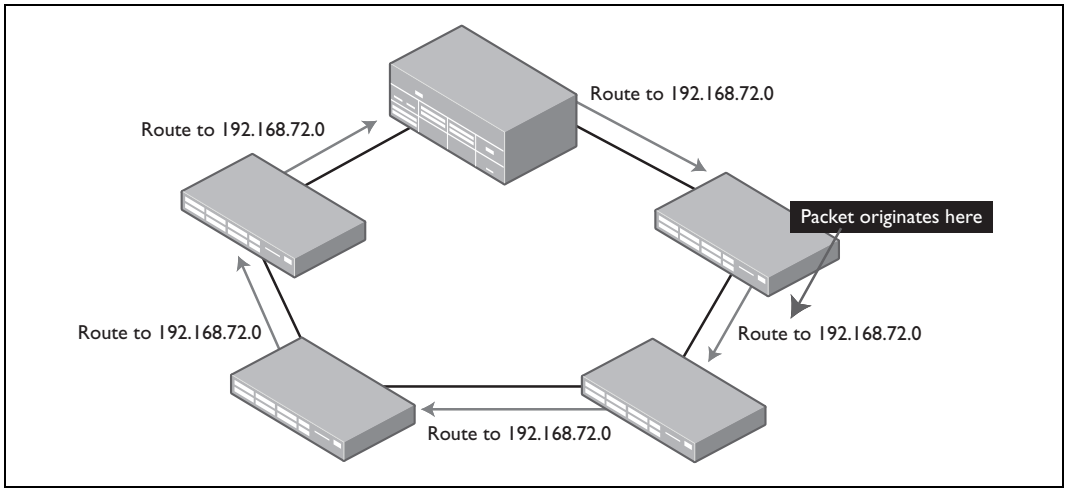
Routing Loop

A routing loop is a problem that can arise due to misconfiguration in a network. Simply, it is a situation whereby the routes that the networks' routers have to a given destination end up taking packets around in a loop.



The result of this problem is, of course, that packets sent to the subnet in question do not reach their destination. If you suspect that a routing loop is occurring in the network, then the first place to start the troubleshooting is to look at the logs of the switches. There are two different log messages that are likely to appear, depending on where the looping packets have originated from - within or outside the loop.

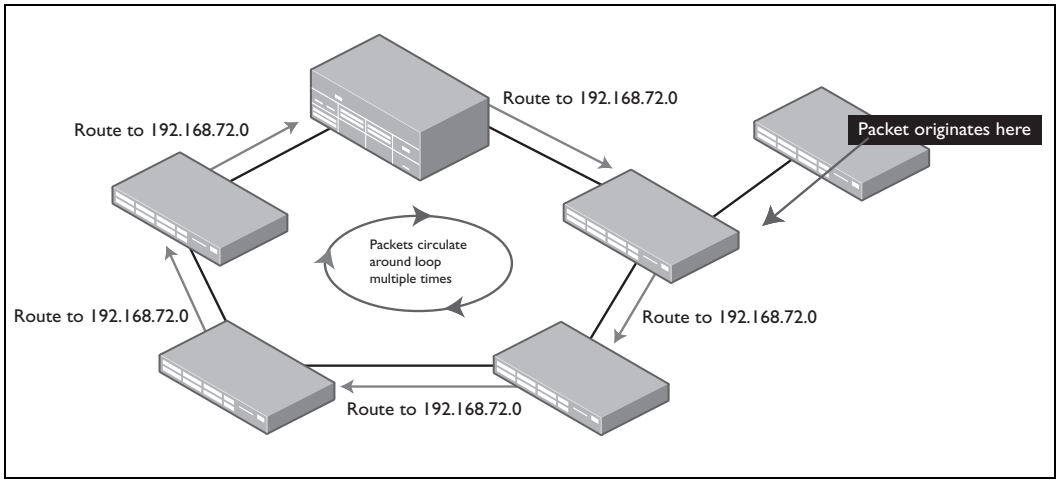
If the packets originate from a device that is **within** the loop, then it will only be able to make one traverse of the loop.



When it gets back to the device from which it originated, that device will notice that the source address of the packet is that device's own IP address. The device will drop the packet (as is standard when a network device receives a packet that purports to come from its own IP address), and reports a log message to say that a packet with an **martian** source address has been received.

```
martian source 172.18.180.254 from 172.18.180.253, on dev vlan180
ll header: ff:ff:ff:ff:ff:ff:00:09:41:fd:c0:15:08:06
```

If the packet originates from a device **outside** of the loop, then the packet will circulate around the loop multiple times, as there is no device in the loop it will recognise its own source IP on the packet.



Eventually, the TTL in the packets will be decremented to 0, and they will be dropped. To see if the switch is decrementing any packets' TTL to 0, enable ICMP packet debugging, with the commands:

```
debug ip packet icmp
terminal monitor
```

If the switch is decrementing any packets' TTL to 0, you will see it outputting messages like:

```
13:35:11.686833 IP <Switch IP address> > 45.45.45.48: ICMP time exceeded
   in-transit, length 54
```

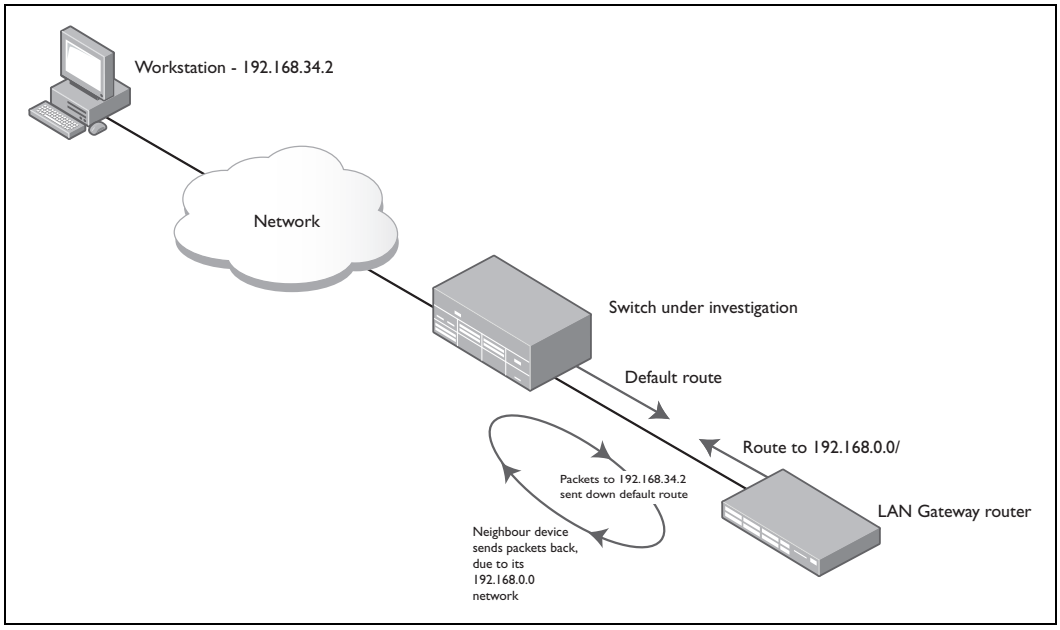
Having seen one or both of the log messages shown above, the next step is to find the destination addresses that these packets are going to, then look for the route(s) to those addresses in each of the switches in the loop; and determine which switch has incorrect route(s). From there it is a matter of working out what misconfiguration or routing protocol error is resulting in the incorrect route(s).

Packets 'bouncing' due to a lost route

This problem is a special, and very common, case of a routing loop.

If a router loses one or more routes, then packets to the destinations covered by those lost routes are likely to be misdirected. Often, the next-best route available for the destinations covered by the lost routes is the default route, so those packets will be directed down the default route.

The next device along the default-route path could be a LAN gateway router, that has summary routes, pointing back to the sending device, that cover the address range being used in the network. If so, then the packets that were sent down the default due to the missing route(s) will simply be sent back to the sending device. The sending device, of course, will simply bounce them back down the default route, and so on.



Eventually, after sufficient such bounces, the TTL in the packets will be exhausted, and they will be sent up to the CPU of the sending device, quite possibly resulting in a **Martian Source** log message.

The best way to debug a situation like this is to note the destination addresses of the packets that are being reported in the Martian Source log messages. Look in the IP route table for the best route to these destination addresses. If the best route is a default route, or summary route, pointing out via the interface on which the Martian Source messages are being reported, then it is highly likely the correct route(s) to those destinations has been lost.

From there, it is a matter of working out why the route(s) has been lost; work out how they should have been learnt (by RIP or OSPF, or by static configuration) and why they are no longer being learnt.

Unexpected reboots

Some software or hardware problems can cause a switch to reboot. Such a reboot should result the creation of a coredump file in Flash. The coredump file is a snapshot of various key pieces of information within the switch at the moment of the reboot. It includes such items as:

- The contents of the function-call stack, and CPU register values.
- The contents of the log files (startup, radius, stats, trace, stacking etc. logs).
- Interrupt data.
- Information on active TCP/UDP sessions.
- Information about internal PCI buses and devices.
- A range of data about internal software processes (queueing, memory usage, environment variables, status, etc. of processes).
- A capture of packets that have recently arrived at the CPU (if packet debugging is enabled).

All this information is zipped up, and stored in Flash. The name given to the file is of the form:

<*process_name*>-<*release_name*>-<*timestamp*>.<*pid*>.tgz

where:

- *process_name* is the process which caused the exception.
- *release_name* is the release image that was running at the time.
- *timestamp* is a unix timestamp (secs since 0:00h, 1 Jan 1970).
- *pid* is the process ID of the thread that caused the exception.

To see what coredump files are present in Flash, use the command **show exception log**.

The contents of the coredump are not really something that can be interpreted by the network engineer investigating onsite. The file should be copied off the switch, and form part of the information provided when the issue is escalated.

Switch locks up or hangs

Very rarely, a hardware or software fault can cause a switch to hang up - i.e. management access to the switch is completely blocked, and it may also cease Layer 2/ Layer 3 forwarding of packets.

Lock-ups can come in different forms:

Hardware fault causing freeze

Sometimes a hardware fault will cause a switch to simply freeze - LEDs stop flashing, no data is forwarded, the switch is quite inactive. In such cases, there is little information that can be gathered on-site to diagnose the fault. The unit simply needs to be sent back to the manufacturer for repair.

Internal communication failure and memory leak

More subtle hang-ups can leave the switch in a state where management access is blocked, but the switch is still forwarding data.

This can be caused by a:

- **Failure of communication between the switch chip and the CPU**. In this case, it is still possible to access the switch command line via the console port, but it cannot be accessed remotely via the switch ports. The switch will continue to Layer 2 switch packets, and quite possibly continue to Layer 3 switch packets (at least for a while until all ARPs age out). The way to verify that communication between the switch chip and the CPU has been lost is to enable packet debugging, which outputs all packets arriving from the switch chip to the CPU, using the command **debug platform packet timeout=1**. If this debug shows no packets arriving, or only corrupt packets arriving, then that is a good indicator of a failure of the communication link between the switch chip and the CPU. Additional information to provide with an escalation of this issue would be the output of the counters for the switch chip-to-CPU communication path, which are output by **show platform count**. It would be best to capture this command a few times in a row, to show which (if any) counters are changing.
- **Memory leak**. If there is a software error which is causing a gradual repletion of the available free memory, then eventually the switch will reach a state where the software cannot operate any more. In this case, the switch command line will probably not even be accessible from the console port, but switching will continue. If the memory leak is gradual, then it is possible to gather useful information about what is happening. After the switch has been rebooted, you can monitor the switch's memory usage over time, to see if some process is gradually accumulating memory. The command to capture is **show memory allocations**, which provides details of the memory allocated to each process running in the software. Capturing output of this command at regular intervals will enable you to see which (if any) process is gradually chewing up memory. If it does become evident that a process is chewing up memory, then the output of the sequence of show memory allocation commands will be valuable information to provide with the issue escalation.

Specific troubleshooting tools in the AlliedWare Plus OS

Using the show tech-support command

This section describes the **show tech-support** command and how to use it. The show tech-support command simply runs a large number of show commands and (by default) writes the output to a zipped file in Flash called tech-support.txt.gz. This saves a large amount of information that is very helpful for anyone who is trying to resolve a switch or network issue.

If desired, you can specify a name for the file instead of allowing the switch to create one automatically. If an output file already exists with that name, a new file name is generated that includes the current time stamp (for example, tech-support-20090807130032.txt gz). The time stamping means that you can run the command repeatedly, for example, every time something significant happens on the network.

The **show tech-support** command is a very useful tool when debugging problems. It should be one of the first tasks you perform when you arrive at a site and start to investigate a problem. Even if there are no problems occurring when you first arrive, run the command anyway, the file will serve as a benchmark for the switch when there are no problems on the network. You can use this output at a later time to compare selected tables when you see a problem occurring.

Command Syntax

The basic syntax for the show tech-support command is simply:

```
show tech-support
```

The full command syntax includes some optional parameters for greater control, and is:

```
show tech-support [all] [bgp] [igmp] [ip] [ospf] [pim] [stack] [stp]
   [system] [outfile <filename>]
```

The optional parameters **bgp**, **igmp**, **ip**, **ospf**, **pim**, **stack**, **stp**, and **system** act like filters, they reduce the number of show commands that the switch runs. The table "Commands that the show tech-support command runs" on page 222 shows the filter parameters that apply for each command. If you do not specify any of these filters, the switch runs all the show commands in the table.

The **outfile** parameter is also optional and specifies a name for the output file. If you do not specify a filename, the switch uses ***tech-support.txt.gz***, and includes the time stamp in the filename if necessary.

Commands that the show tech-support command runs

The x900 series switches and the SwitchBlade® x908 will output the following commands to the file:
Each command is executed in Privileged EXEC mode.

Command	Which filters apply?
###### General debug information ######	
show clock	all
show system	all
show system environment	system
show system pluggable	system
show running-config	all
show startup-config	system
dir all recursive flash:	system
show boot	system
show stack	stack
show stack detail	stack
show cpu	system
show cpu history	system
show memory	system
show users	system
show license	system
show ntp status	system
show ntp associations	**system**
show ntp associations detail	**system**
show log	system
show log permanent	system
show exception log	system
show clock	
show platform table counter	
Wait 5	
show clock	
show platform table counter	
show cpu	
###### Physical Layer information ######	
show interface brief	**ip**
show interface	**ip**
###### VLAN information ######	
show vlan brief	**ip,igmp,bgp,ospf**
###### IP information ######	

Command	Which filters apply?
show ip interface brief	**ip,igmp,bgp,ospf**
show arp	**ip**
show ip route	**ip**
show ip bgp dampening dampened-paths	**bgp**
show ip bgp dampening flap-statistics	**bgp**
show ip bgp dampening parameters	**bgp**
show ip bgp neighbors	**bgp**
show ip bgp paths	**bgp**
show ip bgp summary	**bgp**
show ip ospf	**ospf**
show ip ospf database	**ospf**
show ip ospf interface	**ospf**
show ip ospf neighbor	**ospf**
show ip ospf route	**ospf**
###### ASIC information ######	
show platform	
show clock	
show platform table port counters	
wait 5	
show clock	
show platform table port counters	
###### Hardware tables - General ######	
show platform table vlan	
show platform table macaddr	
show platform table ip	
show platform table ipsw	
show platform table nh	
show platform table nhsw	
show platform table stpgroup	**stp**
show platform table macfull	
show platform table port	
show platform table traffcicond	
show platform table pcl	
show platform table pclaction	
show platform table pclprofile	
show platform table fieldproc	

Command	Which filters apply?
show platform table topology	
show platform table interface	
show platform table outlif	
show platform table fdb	
show platform table ipswcount	
show platform table nhswcount	
###### Hardware tables - Low Level #####	
show platform mem DSCP_TABLE	
show platform mem E2E_HOL_STATUS	
show platform mem EGR_EM_MTP_INDEX	
show platform mem EGR_IM_MTP_INDEX	
show platform mem EGR_L3_INTF	
show platform mem EGR_L3_NEXT_HOP	
;show platform mem EGRESS_MASK	
;show platform mem EGR_VLAN	
;show platform mem EGR_VLAN_STG	
;show platform mem EGR_VLAN_XLATE	
show platform mem EM_MTP_INDEX	
show platform mem FP_COUNTER_TABLE	
show platform mem FP_METER_TABLE	
show platform mem FP_POLICY_TABLE	
show platform mem FP_PORT_FIELD_SEL	
show platform mem FP_RANGE_CHECK	
;show platform mem FP_TCAM	
;show platform mem IFP_PORT_FIELD_SEL	
;show platform mem IM_MTP_INDEX	
;show platform mem ING_L3_NEXT_HOP	
show platform mem IPORT_TABLE	
show platform mem L2_ENTRY	
show platform mem L2_USER_ENTRY	
show platform mem L3_DEFIP	
show platform mem L3_ENTRY_IPV4_MULTICAST	
show platform mem L3_ENTRY_IPV4_UNICAST	
show platform mem L3_ENTRY_IPV6_MULTICAST	
show platform mem L3_ENTRY_IPV6_UNICAST	
show platform mem L3_IPMC	

Command	Which filters apply?
show platform mem LPORT	
show platform mem PORT_MAC_BLOCK	
show platform mem IPMC_GROUP0	
show platform mem IPMC_GROUP1	
show platform mem IPMC_GROUP2	
show platform mem IPMC_GROUP3	
show platform mem IPMC_GROUP4	
show platform mem IPMC_GROUP5	
show platform mem IPMC_GROUP6	
show platform mem IPMC_GROUP7	
show platform mem IPMC_VLAN	
show platform mem MODPORT_MAP	
show platform mem NONUCAST_TRUNK_BLOCK_MASK	
show platform mem PORT	
show platform mem SOURCE_TRUNK_MAP	
show platform mem SRC_MODID_BLOCK	
show platform mem VLAN_STG	
show platform mem TRUNK_BITMAP	
show platform mem PORT_TRUNK_EGRESS	
show platform mem TRUNK_GROUP	
show platform mem VLAN_MAC	
show platform mem VLAN_PROTOCOL	
show platform mem VLAN_SUBNET	
show platform mem VLAN	
show platform mem VLAN_XLATE	
###### IP multicast information ######	
show ip igmp groups	**igmp**
show ip igmp interface	**igmp**
show ip igmp snooping mrouter interface vlan1	**igmp**
show ip pim dense-mode interface detail	**pim**
show ip pim dense-mode mroute	**pim**
show ip pim dense-mode neighbor detail	**pim**
show ip pim dense-mode nexthop	**pim**
show ip pim dense-mode bsr-router	**pim**

show ip pim sparse-mode interface detail	**pim**
show ip pim sparse-mode mroute detail	**pim**
show ip pim sparse-mode neighbor	**pim**
show ip pim sparse-mode nexthop	**pim**
show ip pim sparse-mode rp mapping	**pim**
show platform table l2mc	**igmp**
show platform table ipmulti	**igmp,pim**
show platform table ipmultisw	**igmp,pim**
show platform table mclinklist	**igmp,pim**
###### QoS information ######	
show platform table class	
show platform table cosmarking	
show platform table cosremarking	
###### Link Aggregation information ######	
show lacp sys-id	
show etherchannel	
show etherchannel detai	
show lacp-counter	
show static-channel-group	
show platform table channel-group	
show vrrp	
show spanning tree	**stp**
show clock	**all**
; END	

Extending the show tech-support command

You can add more commands to the list that **show tech-support** runs.

To do this, create a text file called **tech-support.conf** in the directory **flash:/.configs/** and add your extra commands to this file. The switch will run these commands after it has finished running the standard commands.

The rules for adding commands to this file are:

- Each command is executed in Privileged Exec mode.
- You can add any **dir** command and/or any **show** command. (A **wait** command is also available).
- You can add comment lines, which must begin with a semicolon (";"). The switch will ignore comment lines.

- You can add lines that start with a non-alphabetic character (for example, #). These lines are displayed in the output, so you can use them to put comments into the output.
- You can filter commands so that they run when **show tech-support** is entered with a filtering parameter. The available filters are bgp, igmp, ip, ospf, pim, stack, stp, and system. To specify the filter, include it in {} after the command. For example, to specify STP information for ports when the filter is stp (and when there is no filter), enter the line:

```
show spanning-tree interface port1.0.1-1.0.12 {stp}
```

If you do not specify a filter, the line is only processed when the command is entered with no filter or when the filter is **all**. You can set filters on any lines except comment lines (lines that start with a semicolon).

USA Headquarters | 19800 North Creek Parkway | Suite 100 | Bothell | WA 98011 | USA | T: +1 800 424 4284 | F: +1 425 481 3895
European Headquarters | Via Motta 24 | 6830 Chiasso | Switzerland | T: +41 91 69769.00 | F: +41 91 69769.11
Asia-Pacific Headquarters | 11 Tai Seng Link | Singapore | 534182 | T: +65 6383 3832 | F: +65 6383 3830
www.alliedtelesis.com

Connecting The IP World

Allied Telesis™