



The No-Sweat Guide to Network Topology

Kevin Dooley



ABOUT KEVIN DOOLEY

Kevin has more than 15 years of experience as a network engineer. He has designed and implemented several of the largest and most sophisticated enterprise data networks in Canada and written several highly regarded books on networking for O'Reilly and Associates, including *Designing Large-Scale LANs* and *Cisco IOS Cookbook*. Kevin holds a Ph.D. in theoretical physics and numerous industry certifications.

ABOUT AUVIK NETWORKS

Auvik's cloud-based network management software keeps IT networks around the world running optimally. By automating and simplifying network management, Auvik helps rocket an IT team's efficiency and capacity. With the complete network visibility and control you gain with Auvik, you can truly own the network. Copyright © 2021 Auvik Networks Inc. Some rights reserved.

You may give away or pass along this e-book for free as long as it is left completely intact, unaltered, and is delivered via this PDF file.

CONTENTS

Why Topology Matters	4
What you'll learn in this guide	5
Basic Network Topologies	6
Network layers	6
Ethernet, bus and switches	7
Network Segmentation	9
VLANs and trunks	10
Routing	11
High Availability	13
Device-level high availability	13
Layer 2 network high availability	14
Layer 3 network high availability	15
How to Draw Network Diagrams	16
Common network diagram symbols	16
Drawing Layer 3 diagrams	18
Drawing Layer 2 diagrams	19
Drawing Layer 1 diagrams	20
Drawing combined-layer diagrams	20
How to map an existing network	22
Useful Topologies	26
Single switch	26
Single switch with Internet connection	27
Touchdown segment	29
Three-tier	31
Web-tier	33
Hub-and-spoke	36
Ring	37
Internal firewalls	39
Helpful Resources	41

WHY TOPOLOGY MATTERS

As a network administrator responsible for the care and feeding of a network, it's vitally important you have an extremely detailed understanding of your network topology.

Without this information, even basic troubleshooting can be unnecessarily difficult. You'll find, if you haven't already, that troubleshooting is much easier if you have detailed and up-to-date network documentation.

- When one device is having trouble communicating with another one, you need to know how the packets are supposed to get there.
- You need to know whether there are firewalls in the middle, or whether there are particular bottlenecks where you might be suffering from congestion problems.
- You need to understand any single points of failure where a simple device or link failure could have catastrophic effects.
- Having up to date documentation can save literally hours, and sometimes even days, tracking down and troubleshooting network performance problems. As a network administrator, those are hours and days back to you, to spend time with your family. For your business, this means they're back up and running quickly, reducing the overall cost of downtime. After all, Gartner* has pegged the cost of downtime at \$5,600 per minute - that adds up quick!

There are several other good reasons to maintain your network documentation as well.

If you ever need to change anything, to add a new switch or a new link to a remote office, you need good documentation to understand how the traffic will flow through these new pieces of network. This will help ensure the new network is stable and efficient.

Another important reason to document your network is that you might need to show it to an auditor. PCI auditors always want to see your network topology diagrams. They need to know where the firewalls and servers are and all the different ways that somebody could get into the network.

Even if you never need to submit to this kind of external audit, if there were ever a serious problem with your network and you needed to either defend the design to management or bring in external consultants, having good and current diagrams will always help your case.

There are two ways to obtain a deep understanding of your network.

The first is to be some sort of super genius who can keep it all in her head at once. The trouble with this approach, though, is that it makes you so completely indispensable that you can never be promoted, which would be a shame for somebody so brilliant.

The second way, which has always worked better for me, mostly because I'm not even an intermediate genius, is to document everything.

*source: <https://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/>

WHAT YOU'LL LEARN IN THIS GUIDE

The No Sweat Guide to Network Topology covers two main topics:

- How to document your network topology
- Some of the most common and useful topologies that you'll run into

By the time you get to the final pages, you'll be able to map your network both logically and physically. You'll understand the differences between the different logical and physical layers and how they relate to one another. And you'll know several of the most common network topologies, along with where and when to use them.

A note on mapping methods

You can map your network manually using pencil and paper, or with a drawing program like Visio. You might also use automated software tools to help you. It's obviously a lot easier with the software tools, but even the best software needs a little bit of help.

For example, it's very hard for any software to know how the cables have been run or where the patch panels are. Yet these are often the technical details you need when you're trouble shooting a problem or connecting a new device. On the other hand, automated tools are extremely good at sorting out Layer 2 and 3 connections between devices.

The process of documenting a network is highly personal. I'll show you how I like to do it, and I'll try to explain why I think my methods are useful, but they're certainly not the only way.

Ready? Let's get started.

BASIC NETWORK TOPOLOGIES

This guide uses the concept of network protocol layers in every section, so it's worth taking a moment to explain what they mean.

Network layers

There's a standard seven-layer model called the OSI model that every networking book mentions. The idea is that higher-layer protocols sit on top of lower-layer protocols. But reality is both more complicated and simpler than this model suggests.

Real networks use things like VPN tunnels where a Layer 3 network (or sometimes a Layer 2 network) sits on top of another Layer 3 network with a different logical topology. So although it's important to understand the layers, it's also important not to be too pedantic about them.

When talking about network topology, we're mostly interested in the bottom few layers.

Layer 1 is the physical layer. For network design purposes, this means the things you can touch: the cables and the equipment.

But it technically also includes the electrical and optical signalling properties. Layer 1 defines the properties of the cables that are necessary to carry the signals. Layer 1 also worries about wireless signalling where that's used.

As network designers, we need to make sure we get the right cables and we need to watch out for distance limitations. And of course we need to make sure the right devices are physically connected to one another.

Layer 2 is the logical link layer. Most modern networks are based around some sort of Ethernet. We have everything from 10Mbps to 100Gbps flavors of Ethernet. There are LAN and WAN Ethernets. And even though there are also a lot of other WAN protocols, like DSL and T1, most WAN providers will prefer to deliver it to you using an Ethernet handoff. So those other Layer 2 protocols usually aren't your problem.

The other important Layer 2 protocol to consider is Wi-Fi. I think of Wi-Fi as simply a flavor of Ethernet because it uses very similar signalling. Obviously there are some very important radio frequency (RF) propagation and isolation factors to worry about with Wi-Fi, but those are actually Layer 1, not Layer 2 issues.

Layer 3 is the network layer. Throughout this book that will mean either IPv4 or IPv6. In 2014 or 2015, you'll probably be building your network to support IPv4 though I strongly recommend keeping IPv6 in mind as a future option. Fortunately, a good IPv4 network topology will generally support an IPv6 network perfectly well, although the converse might not be true.

For reasons lost to the mists of history, Layer 2 packets are called frames while layer 3 packets are called packets.

Layer 4 is called the transport layer. It's where things like the familiar Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are defined.

We talk about common protocols like HTTP and HTTPS as operating on TCP port 80 and 443 respectively. DNS is UDP port 53.

Things like TCP, UDP, and ICMP have special IP protocol numbers, defined in the Layer 3 part of the IP packet header. Then these higher layer protocols like HTTP and DNS are defined by their port numbers within the Layer 4 part of the header.

Layers 5 through 7 contain dynamic session information for particular applications. Since this book is focused on network topology, we won't be discussing these top layers.

For reasons lost to the mists of history, Layer 2 packets are called frames while layer 3 packets are called packets.

Ethernet, bus and switched

Ethernet is a **bus architecture**, which means all of the devices are connected to it as peers. In a bus, all communications between all of the attached devices are visible to one another, and only one device can talk at a time.

This is how Ethernet was originally designed, and it's still how Wi-Fi works. However, it's not very efficient. If A wants to talk to B, it has to first wait for C to be finished talking to D. It's a fine system if there aren't many devices on the bus, but as the number of devices grows, they wind up having to wait longer and longer for a chance to talk.

An **Ethernet switch** is a way to achieve the same architecture without all the waiting. In a switched Ethernet architecture, every device talks to just one other device, the switch. The devices are all organized using a small **hub-and-spoke** or **star** connection model.

If A wants to talk to B, it first sends its packet to the switch. The switch reads the destination Ethernet MAC address in the frame received from A, and looks it up in its internal MAC address table to find the interface that holds that address. Then the switch forwards the frame across its own backplane to the destination interface.

The other important advantage to a switch, besides eliminating waiting, is that each individual interface can operate at **full duplex**. Full duplex means the devices connected to a switch can send and receive Ethernet frames at the same time. So not only does the network not have to wait for every other device to stop talking before it sends a frame, it doesn't even need to wait for the switch to stop talking.

Full duplex means the devices connected to a switch can send and receive Ethernet frames at the same time.

A switch is clearly much more complicated than a simple bus, but it also provides much more efficient use of bandwidth. If the backplane is fast enough, the switch can forward Ethernet frames between all connected devices simultaneously in all directions without waiting.

There are two important internal switch architectures. The switch can either operate as a **store-and-forward** or a **cut through** switch.

Store-and-forward means the switch reads each full Ethernet frame and puts the information in a memory buffer before forwarding it to the destination interface.

A cut-through switch, on the other hand, reads only enough of the Ethernet frame's header to know where it needs to go. It then immediately starts forwarding that frame out the destination interface.

Cut-through switches have lower latency (the time it takes to forward a packet from one port to another). However, the other big difference is that when a store-and-forward switch receives a frame with errors, it can drop it, while a cut-through switch has already started transmitting. Fortunately, Layer 2 errors are fairly rare.

NETWORK SEGMENTATION

As we saw in the previous section, breaking a Layer 1 Ethernet bus into a whole lot of switch ports improves network efficiency. Communications on the network can take place simultaneously instead of sequentially.

Segmenting an Ethernet bus into small, switch-port-sized network domains is also important for security reasons. It turns out to be a bad idea that every device can see every packet, even the ones that aren't addressed to it.

Similarly, there are both efficiency and security reasons for segmenting Layer 3 networks.

Let's consider the efficiency reasons. Even with a switched Ethernet network, there are two important classes of frames that are automatically forwarded to every port.

The first are so-called unknowns. These are frames whose destination Ethernet MAC address is not in the switch's MAC address table. In other words, the switch doesn't know where to send the frames, so it has no choice but to forward them out every port, hoping somebody will be able to handle them.

For most types of normal devices, the only way the switch is able to populate its MAC address table is by listening to every port. Each time it sees a new source MAC address in a frame, it puts that MAC address into the table and assigns it to the interface that received the frame.

Devices that are new to the network, and ones that don't talk frequently, won't be in the table. (For basic housekeeping reasons, old MAC addresses are removed from the table if they haven't been seen in a long time.) The switch must "flood" these frames out all interfaces.

The other type of frame that must be forwarded out all interfaces are broadcasts. These are frames sent to the special Ethernet MAC address FF:FF:FF:FF:FF:FF. The most common types of broadcast frames are ARP and DHCP requests.

Address Resolution Protocol (ARP) is a special protocol that allows the end devices on an IP network to find one another. Device A wants to send an IP packet to device B, but it's never sent a packet there before. So it sends out a special broadcast frame in ARP asking for the destination MAC address that corresponds to the IP address. The switch floods this frame to all devices on all ports and the one that owns that IP address responds. Then the IP session can start.

Dynamic Host Configuration Protocol (DHCP) is how devices that are new to the network (or just rebooted) can automatically learn important information like what IP address and mask they should use, and where the DNS server and default gateway are.

There are two important classes of frames that are automatically forwarded to every port.

Naturally, the new device doesn't know anything about the network yet, so it has to use the same broadcast destination MAC address to request this information. The frame gets forwarded out all switch interfaces and the DHCP server responds.

Both ARP and DHCP frames are relatively rare compared to the rest of the application packets flying around a network, but it's important to notice that they get forwarded to everybody. Every device on the network has to receive these frames, look at them, and decide whether they need to do anything.

As the number of devices on a network grows, the number of broadcast frames grows. If you put enough devices on the same network, you start to run into serious efficiency problems once again. That's why we segment the network into smaller broadcast domains using functions like VLANs and routing.

One of the interesting features of IPv6 is that it uses multicast instead of broadcast for these types of functions. Multicast packets are only forwarded out the ports that explicitly say they want to receive them. It's much more efficient because devices don't need to see things that don't interest them.

The second important reason for segmenting your network is security. It's much easier to control communications between devices if the traffic must pass through a common control point where it can be examined. We tend to use special devices called firewalls for these control points.

Although there are firewalls that can operate at Layer 2, these are generally reserved for special situations. Layer 3 segmentation is easier to understand. And as I always say, things that are easier to understand are easier to support.

VLANs and trunks

All modern Ethernet switches have a feature called Virtual Local Area Networks (VLANs). VLANs give you the ability to create groups of interfaces that are able to communicate together.

Devices in the same VLAN can send Ethernet frames directly to one another. If a device in a particular VLAN sends a broadcast frame, it's received by all the other devices in the same VLAN, but not by devices in different VLANs.

Sometimes a VLAN is also called a broadcast domain.

Suppose you have a switch whose interfaces are chopped up into two different VLANs and you want to connect this switch to a second switch that has the same VLANs. You want all of the devices on VLAN 100 on switch A to talk to all of the devices on VLAN 100 on switch B. And you also want the VLAN 200 devices on both switches to talk to one another.

There are two ways to make the connection. The obvious way is to simply configure a port on each switch on VLAN 100 and a different port on each switch on VLAN 200, then connect these ports together.

VLANs give you the ability to create groups of interfaces that are able to communicate together.

This will work, but it doesn't scale well. If you want a lot of VLANs, you won't have very many interfaces left for the devices they connect to. So we use something called a trunk instead.

A trunk is a single connection between two switches that contains a bunch of VLANs. We use a special part of the Ethernet protocol called 802.1Q to make the connection.

The 802.1Q protocol works with a very simple extension of the standard Ethernet protocol that adds VLAN information to the header. Now all of the frames for VLAN 100 are tagged for this VLAN and all of the frames for VLAN 200 are tagged for that VLAN.

Further, if one of the switches has a third VLAN, 300, you could use the same trunk to carry this VLAN to the second switch. Then you could use another trunk to a third switch to carry VLAN 300, even though the one in the middle doesn't have any interfaces configured for VLAN 300.

There's a special VLAN in every trunk called the native VLAN. When a switch receives a frame without 802.1Q VLAN tags on a trunk interface, it assumes the frame was intended for the native VLAN and forwards it accordingly. By default, most switches use VLAN 1 as their native VLAN, but this is configurable.

The trouble with the native VLAN is that, if I can somehow trick the switch and inject an untagged frame on the trunk interface, the frame will wind up on the native VLAN. In principle, you could use this technique to hop packets from one VLAN to another, perhaps violating a security segregation rule.

For this security reason, when it comes to native VLANs, I always follow a simple rule: Leave the native VLAN as VLAN 1 (the default value), then disable VLAN 1 on all switches. This way, even if somebody is able to hop a packet onto the native VLAN, it doesn't do any good.

Routing

There are two ways to interconnect VLANs. You can interconnect them at Layer 2 using a bridge, or you can interconnect them at Layer 3 using a router.

The Ethernet switches we discussed in the previous section are a form of bridge. When you use a bridge, Ethernet-connected devices on one side can send Ethernet frames directly to devices on the other side. If I use a bridge to interconnect two VLANs, then it's as if there's only one VLAN. The devices talk directly.

Routing is a Layer 3 concept that forwards packets based on IP addresses, not Ethernet MAC addresses. To understand routing, we first need to talk about subnetting.

There are two ways to interconnect VLANs: at Layer 2 using a bridge, or at Layer 3 using a router.

Subnets

Every time you create an IP network, you define a subnet. A subnet is simply a way of grouping IP addresses that will be treated together. They'll generally be located on a common VLAN.

Every subnet needs a router of some sort to allow packets to leave the subnet and go to another. The router has a routing table that tells it how to direct packets between subnets.

A subnet is defined using a network number and a mask. The IP address of every device in the subnet has two parts, the network portion and the host portion.

Consider a simple example like 192.168.1.15 with a mask of 255.255.255.0. If you think of the mask in binary notation, each 255 is actually a string of eight 1s and no 0s. Everywhere you see a 1 is part of the network, and everywhere you see a 0 is a host.

In our example mask of 255.255.255.0, the first three bytes (192.168.1) are all network and the last byte (15) is the host portion.



Figure 1 – Subnet mask

All devices in the same subnet can talk to one another directly. In our example, that means everything from 192.168.1.1 to 192.168.1.254 can talk to each other. They can still talk to devices that aren't in this range, but the packets must be sent to the router, which will direct them to the right destination network.

In most cases, you'll define a single **default router** for each subnet. If a device wants to send a packet to 192.168.2.95, for example, it will compare this destination address to its own address and mask and see that the destination address is on a different network.

The only thing the device will do differently when the destination is in another subnet is to put the packet into an Ethernet frame whose destination address is the MAC address of the default router instead of the MAC address of the destination device. Then the router looks in its routing table to see where to send the packet next.

HIGH AVAILABILITY

One of the most important considerations in any network design is **high availability**. The rule of thumb is that for anything important, any single device or any single link should be able to fail without affecting the application.

If you have an application on your network that needs high availability, you need to also make sure the network itself can recover. Nothing works if the network doesn't work.

In practice, of course, the network takes some time to recover from a lot of failure modes.

I used to make a practice of calculating the net failure probability and the expected system availability from different levels of redundancy. I don't do that anymore.

The problem with that way of approaching redundancy is that it assumes most of your failures will be hardware failures. In truth, most failures are caused by human error. A lot are due to software bugs. Even more are the result of configuration errors.

The thing that causes humans to make errors (assuming the humans involved are making an honest effort) is complexity. The more complicated something is, the more likely some body will make a mistake with it.

Complicated software has more bugs. Complicated IT infrastructure has more configuration errors. And complicated environments are harder to troubleshoot when something does go wrong.

All of this is to say that adding extra devices and links with complicated redundancy mechanisms could make your network less stable. However, with that caution, high availability is an essential feature of any mission-critical network.

The more complicated something is, the more likely some body will make a mistake with it.

Device-level high availability

By device-level high availability, I mean the server itself is able to support some sort of high availability. This could take many forms. It could involve redundant power supplies, redundant network connections, or even fully redundant devices. These are all good strategies with different real and theoretical benefits.

The thing most likely to fail in most servers is the power supply. Network devices are somewhat less likely to fail, but if a network device like a switch fails, it affects a lot of devices. Full device redundancy is obviously best of all, particularly if the redundant devices connect to different network switches. We call this N+1 redundancy, meaning you have one more than you need.

Layer 2 network high availability

There are two popular Layer 2 network high availability mechanisms, and dozens of less common ones. The first is link redundancy using Link Aggregation Control Protocol (LACP), and the second is Spanning Tree Protocol (STP).

Link Aggregation Control Protocol

LACP is a way of logically bonding two or more (usually to a maximum of eight) physical links into a single **channel**. In Cisco jargon, this is called EtherChannel.

The two devices interconnected with LACP dynamically load balance traffic among the physical links. There are many different load-balancing algorithms, but they generally involve some mix of Layer 2 and 3 source and destination addresses.

The various load-balancing mechanisms mean that any one session will always take the same physical link. So if you make a channel out of four Gigabit Ethernet links, you get a theoretical maximum of 4Gbps aggregate throughput, but any single session can only be, at most, 1Gbps.

The nice thing about LACP is that it provides redundancy as well as load balancing. If one of the individual links fails, the two devices shift traffic to one of the remaining links automatically, usually in a couple hundred milliseconds.

The biggest problem with LACP is that it's designed to allow connections from one device to one other device. So by itself it doesn't help you if one of the devices fails.

There are a few multi-chassis LACP implementations, however. Cisco has a feature called Virtual Switching System (VSS), which is available only on the large 6500 series switches. The company's Nexus series switches have a different feature called a Virtual Port Channel (VPC) that performs a similar function.

VSS solves the problem of distributing the channel between two switches by putting a very high-speed link between them and treating them as if they were a single switch. The controller module in one switch becomes the brain for both devices.

VPC takes a different and more complicated approach. Once again, it requires a fast link between the switches. Then the two switches exchange state information about the individual links in the channels they share.

The nice thing about LACP is that it provides redundancy as well as load balancing. If one of the individual links fails

Spanning Tree Protocol

The other popular Layer 2 redundancy mechanism is Spanning Tree Protocol. STP works across the whole broadcast domain on all switches. Spanning tree solves two problems: It allows redundant links and redundant switches, and because that would cause loops, it also prevents loops.

Spanning tree solves two problems: It allows redundant links and redundant switches, and because that would cause loops, it also prevents loops.

Spanning tree works by logically rebuilding the switch-to-switch links as a tree. One switch is designated as the **root bridge**, which is the base of the logical tree. Then the switches figure out the shortest paths they can all take to reach the root bridge. Every link that isn't on one of those shortest paths gets disabled to remove the loops.

Spanning tree does all of this using a combination of individual link costs and bridge priorities. Usually you can leave the link costs at their default values and just play with the bridge priorities.

Select the switch you want as the root bridge and give it the lowest priority. Select another switch that should become the root if the first one fails and give it the second lowest priorities.

Setting your root bridge and backup in this way generally gives you good results, but you might have special reasons for wanting to force certain links to be active and others to be disabled. In that case, you can further manipulate the protocol by manipulating port costs and bridge priorities. But that's more advanced network traffic engineering and beyond the scope of this short guide.

Layer 3 network high availability

There are two main ways of getting high availability at Layer 3. On network segments that have a lot of normal end devices like servers or workstations, or even embedded systems like IP phones, printers, or other appliances, the most important thing to make redundant is the default router for the segment.

This is done using a router redundancy protocol such as Hot Standby Routing Protocol (HSRP), Virtual Router Redundancy Protocol (VRRP), or Gateway Load Balancing Protocol (GLBP).

All of these protocols work in a similar way. They all present a virtual IP address that's used by the other devices in the subnet as their default gateway.

The virtual IP address can move around between the different physical routers. This way, if there's a failure of one of the routers or any of their upstream links, another one can automatically take over.

The other important Layer 3 redundancy mechanism is a routing protocol. Routing protocols such as Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and Border Gateway Protocol (BGP) are used to distribute the routing tables among the different routers in the network.

One of the most important things to remember about network high availability mechanisms of all types is that you have to monitor them.

Suppose an automatic failover mechanism has engaged to protect your network and keep everything working transparently following a critical link or device failure. If you don't know the failure has happened, you can't fix it. Then you're left with the mistaken belief that you have a highly available network.

HOW TO DRAW NETWORK DIAGRAMS

Before we discuss specific useful network topologies, I want to talk about network diagrams because I want to use diagrams to illustrate the topology types.

Good network diagrams are not hard to make, but I find them distressingly rare. Even network engineers with years of experience often make diagrams that are jumbled and hard to understand.

The important thing is to be clear in your own mind about what information you're trying to convey. It's better to draw several diagrams that show different aspects of the same network than to try to put everything on one sheet of paper.

It's a good idea to make a separate diagram for each network protocol layer. I usually start with a Layer 3 diagram to show the routing and IP subnets. Then I make a Layer 2 diagram showing the switch connections, trunks, and LACP channels. After that comes a Layer 1 diagram showing physical layouts of the devices.

In more complicated networks, I also do diagrams showing traffic flows, routing protocol distribution mechanisms, VPNs, and other important aspects of the network design.

It's important to draw each of these as separate pictures because they show different things. Combining them only confuses the information and makes the drawing harder to understand.

Common network diagram symbols

There are some useful shorthand symbols to use in network diagrams. These are my rules, and they aren't universal, but I find them useful, particularly for drawing network topologies on a whiteboard.

My Layer 3 network devices are **circles**. My Layer 2 devices are **rectangles**.

Clouds are used to summarize parts of the network that aren't important for the purposes of the diagram. This could mean the Internet or a WAN or it could even mean a collection of internal network segments like user VLANs.

Triangles represent multiplexer devices, which used to be more common in network diagrams but aren't typically seen in enterprise network designs anymore. So I've started to use triangles for IP phones.

Everything else is a **square** or a **rectangle**, intended to represent a generic box.

Each symbol also has a specific marking on it that indicates exactly which type of device it is. When I draw these figures on a whiteboard, a router has an **X** on it and firewalls have a **>**.

If you're creating network diagrams using a drawing program or your map is being automatically generated by a software tool, the symbols will be much more elaborate.




























	Hand-drawn	Visio	Auvik
Cloud			
Firewall			
Terminal			
Switch			
Bridge			
Server			
Router			
Mainframe			
Hub			

Figure 2 – Network diagram symbols

Note that all the diagrams you'll see in this guide are shown as I might draw them with Visio using Visio's symbols. For interest's sake and so you can see the differences, on page 28 I've also included a second version of a diagram, shown as the Auvik tool would map it.

I mention symbols you might use when drawing on a whiteboard because that's where a lot of good network designs start. A group of people gather around the board and figure out what needs to talk to what and how best to accomplish that.

Typically you'll be deploying the network to support some sort of application. And typically it will involve some client and server devices. It might involve an Internet connection and perhaps some firewalls. In a large organization, there could be a separate person responsible for each of these areas, which is why the whiteboard is such a useful place to start.

Drawing Layer 3 diagrams

I always start with Layer 3 diagrams, which show the IP subnets and all Layer 3 network devices like routers, firewalls, and load balancers. The Layer 3 diagram must show all of the important network segments and subnets and how they're interconnected.

Layout is important. I like to show the layout so that it represents the flow of traffic in a broad sense. For example, if I have a bunch of servers being accessed by a group of users, I'll try to put the user network segments on one side of the picture and the servers on the other side.

Similarly, if I want to show how a LAN connects to external networks like the Internet, I group the external networks all on one side or at the top of the picture.

Or, if the point of the picture is to show a WAN with a large number of remote offices connecting to the same network, I'd probably show the connecting WAN in the

middle of the picture and the various remote sites around the edge of the page.

Another layout consideration should be evident in all the diagrams you see in this book: I always draw my network segments either horizontally or vertically. About the only time I use a combination of vertical and horizontal is when I want to show a fundamental difference between the functions of the segments.

For example, I might draw all of my workstation and server segments horizontally, but then draw a special common network management segment vertically down one side of the page. This makes it immediately obvious that the management segment is special.

The Layer 3 diagram should show any high availability mechanisms and redundant network components or redundant paths. It's customary to show router redundancy protocols as an **elongated ellipse** that covers the router links included in the high availability group.

The other important thing about Layer 3 diagrams is that they should only include Layer 3 objects. I don't want to see switches in a Layer 3 diagram, for example. I don't want to see any kind of indication of trunk links on a Layer 3 diagram either.

Layout is important in a Layer 3 diagram. I like to show the layout so that it represents the flow of traffic in a broad sense.

You can show a switch on a Layer 3 diagram only if it's a Layer 3 switch, and then only because it functions as a router. Including Layer 2 objects like a switch in a Layer 3 diagram is confusing, particularly in more complicated pictures.

Another useful thing to put into a Layer 3 diagram is organizational boxes. If there are security zones or interesting groupings of users by function or servers by application, put them together on the picture, put a box around them, and label the box clearly. It's then easy to see the exact network path those users take to reach their servers.

In more complicated network designs, I often use a base Layer 3 diagram showing the VLANs, routers, and firewalls. Then I create several other diagrams to lay over the base diagram. I might have an overlay diagram showing the routing protocol design, another one showing VPNs, and still another showing key application data flows, if that's an important consideration.

Drawing Layer 2 diagrams

Layer 2 diagrams show Layer 2 objects like switches and trunks. They include critical information like which VLANs are included in which trunks, and they show spanning tree parameters like bridge priorities and port costs. In many cases, this is too much information to show easily, so I generally use callout boxes to hold some of the information.

Unlike Layer 3 pictures, Layer 2 diagrams don't need to be laid out in any special way. The most important thing is to keep the picture clear.

If two devices are intended to provide redundancy for one another, then their positions on the page should be related. They should either be located beside one another or in parallel locations on opposite sides of the picture.

If there are different link speeds, they should be indicated in the diagram. I usually show link speed with the thickness of my diagram's connecting lines. The faster the link, the thicker the line.

Sometimes I also use color to indicate special properties of different physical links. For example, I might make fiber optic cables red and copper cables blue. (Technically the cable type is Layer 1 information, but because it doesn't tend to cause confusion in the picture, it's alright to include in your Layer 2 diagram.)

Unlike Layer 3 pictures, Layer 2 diagrams don't need to be laid out in any special way. The most important thing is to keep the picture clear.

Drawing Layer 1 diagrams

I usually use Layer 1 diagrams to show physical connections between devices, but they're also useful for showing cabinet layouts.

Layer 1 diagrams should show port numbers and indicate cable types. In a network that includes many different types of cables, such as fiber optic cables, Category 5 / 6 / 7 copper cabling, and so forth, it's useful to give each cable type a different color.

If there are patch panels, particularly if you want to document how patch panel ports map to device locations and switch port numbers, this information belongs on the Layer 1 diagram.

And if there are different link speeds, you might want to give them different line weights, as described previously for Layer 2 diagrams.

Another type of diagram that's often useful in data center designs is a cabinet layout. It's a diagram that shows exactly what you would see when looking at the front (and sometimes also the back) of the cabinet. A cabinet layout is helpful when you need to tell a remote technician how to find a certain piece of equipment.

Drawing combined-layer diagrams

There's one very special type of diagram in which it's possible to combine Layer 2 and 3 in a single picture. Such a combined diagram is sometimes useful if you have combined Layer 2 and 3 switches and you need to show the relationship between these layers.

The diagram above illustrates a case where showing the relationship between layers is important. The Layer 2 / 3 switch is taking part in redundancy protocols at both Layer 2 and Layer 3.

In the example, four layer 2 / 3 switches are used to interconnect two VLANs. In the inset, which is a pure Layer 3 picture, the arrangement looks highly redundant. But the mixed-layer picture shows you that the switches are in fact cascaded one after the other with no link redundancy at all.

A combined diagram is sometimes useful if you have combined Layer 2 and 3 switches and you need to show the relationship between these layers.

Look in particular at the devices labeled A and B. On the pure Layer 3 diagram, these devices appear to be just one hop apart with four redundant paths. However, the mixed diagram shows that packets from A to B actually have to pass sequentially through all four switches.

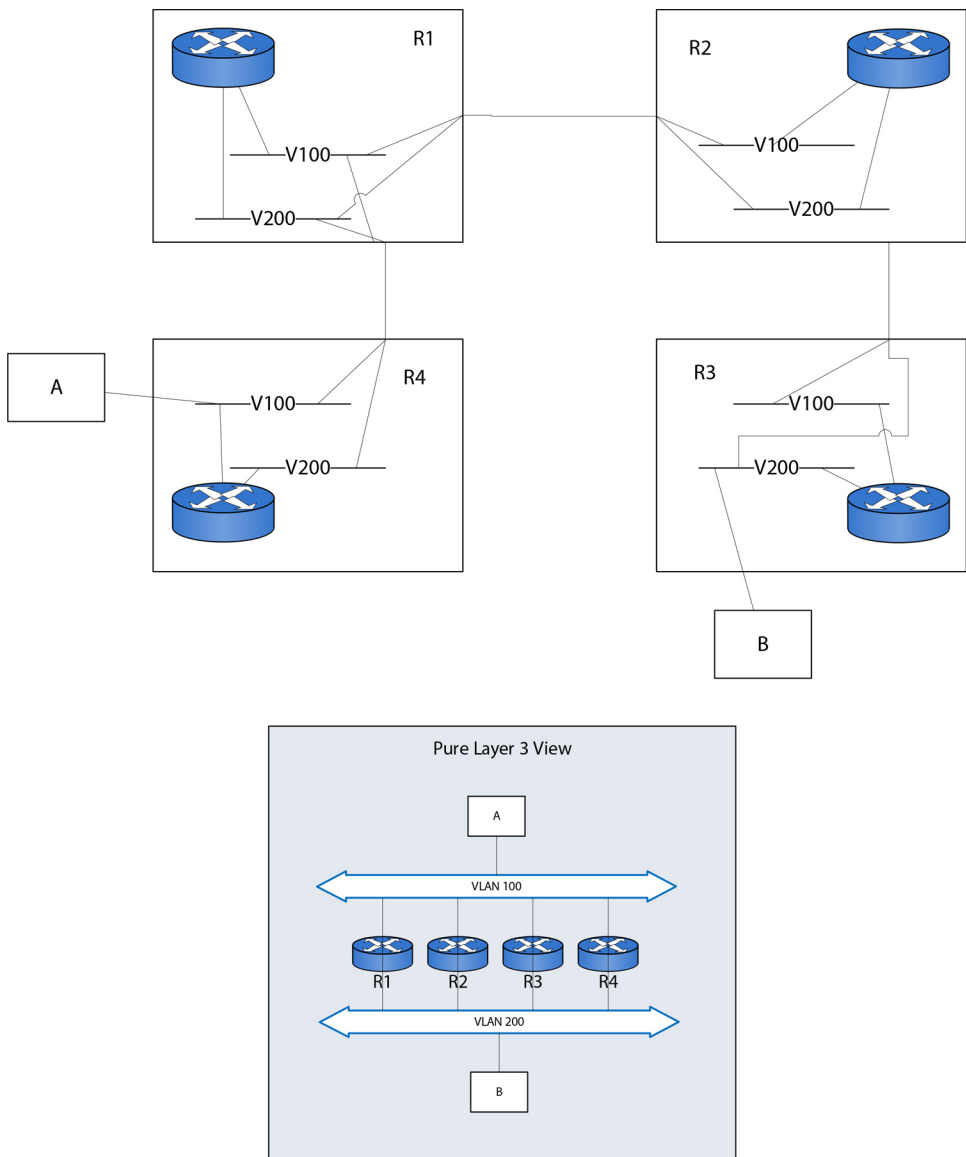


Figure 3 – Combined Layer 2 / 3 diagram

A combined-layer view is also useful when thinking about things like HSRP configuration. Which switch will be the default gateway for each VLAN? And, related to this, will the packets from A to B take the same path as the packets from B to A? None of these details appear in the pure Layer 2 or the pure Layer 3 picture.

Instead, we show the relationship in a combined-layer diagram by drawing boxes for the Layer 2 switch with the VLANs inside it, connected to the Layer 3 router, also inside the switch. The VLANs are connected to trunk interfaces to another Layer 2 / 3 switch.

Note that while this diagram can show the interaction between the layers, it doesn't make either the Layer 2 or Layer 3 network design terribly clear. I'd actually draw all three as separate diagrams, each showing a different important aspect of the network design.

How to map an existing network

Unless you already have a really good set of network documentation, the best place to start learning about network diagrams is by mapping your own network. I always start with pencil and paper, and I expect to draw and redraw the picture many times.

The first time I draw a diagram, it will generally be at Layer 3. I'll log into one device that I think is in the middle or the routing core of my network, and I'll look at its routing table.

On Cisco switches and routers, the command to see the routing table is "show ip route". On Cisco firewalls, it's just "show route". Most other vendors have a similar command.

The output of the "show route" command gives you a list of routes. Each route is defined by a route **prefix**, which defines the subnet range, and a **next hop**, which defines how to get to that subnet. The next hop will either be something like **local** or **connected**, or it will be the IP address of the device that takes care of forwarding packets to the destination.

I draw a small circle to represent the forwarding device, and around it I draw short lines to indicate the locally connected subnets.

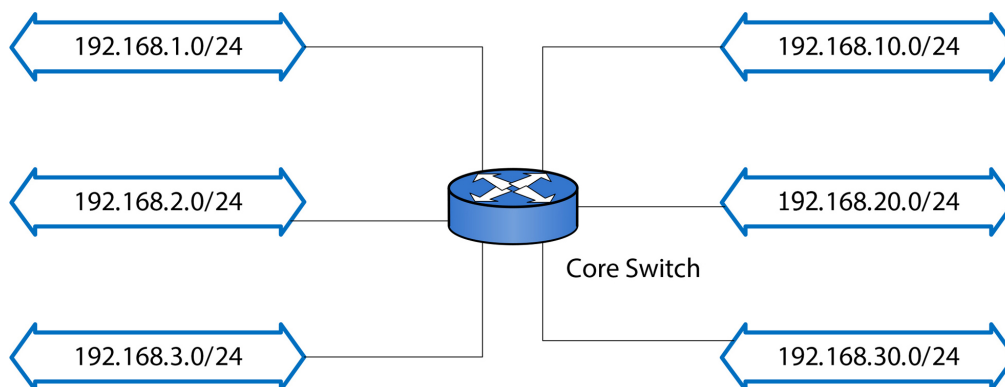


Figure 4 – Step 1 of mapping an existing network

Then I look at all of the next hops that are IP addresses instead of directly connected interfaces. Pretty much by definition, to be useful as a next hop, these devices must be on directly connected networks. Place these devices on the appropriate segments, as in Figure 5.

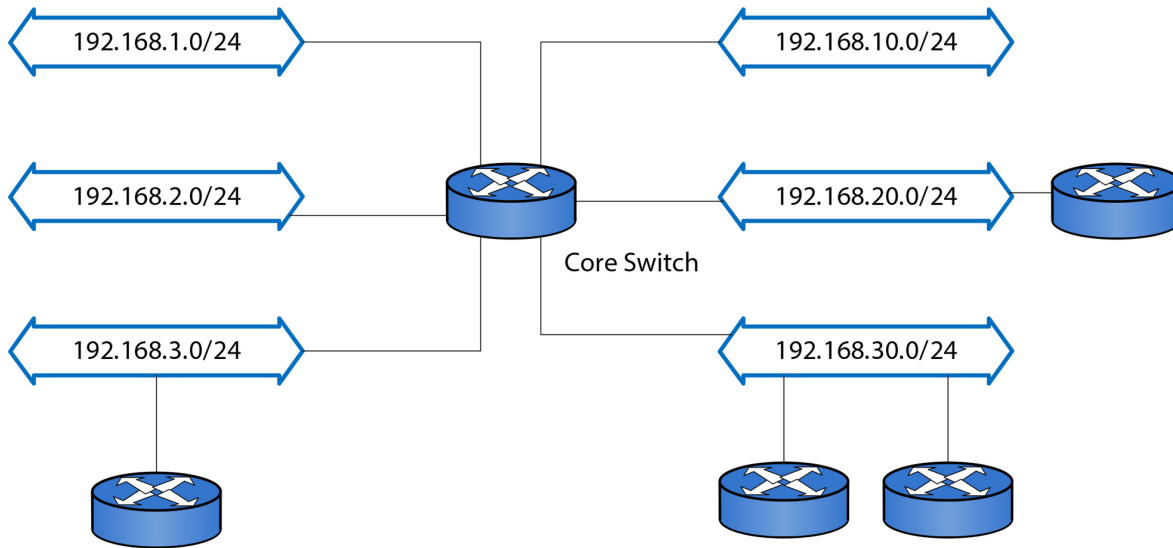


Figure 5 – Step 2 of mapping an existing network

Now you need to log into each of the next hop devices and do the same thing. Soon you'll start seeing devices you've already drawn. Once you've drawn a circle for all of the devices inside your network, you're done.

The next step is to rearrange the networks and the Layer 3 devices until the picture makes sense. I often like to arrange my network diagrams in what I call security order, with the public Internet at the top and the most privileged networks (probably the user segments) at the bottom. Segments that are at a similar security level will be at the same approximate elevation between the top and bottom of the page.

Another common arrangement is to put clients on one side and servers on the other to depict the network in terms of what might be thought of as **traffic flow order**. In this type of arrangement, the segments that are in some sense at a parallel level in terms of traffic flow will be adjacent to one another. In more complicated networks, it's not at all uncommon to have several different views.

Don't worry about fitting your map all on one page. The important thing is that it's readable.

Don't worry about fitting your map all on one page. The important thing is that it's readable.

In fact, for very complicated networks, I usually draw one very high-level network diagram that shows many distinct zones and how those zones are related to one another. Then I have a separate picture for each zone.

After I'm happy with my Layer 3 diagram or diagrams, I move on to Layer 2 and Layer 1. These diagrams are considerably more difficult to do just by logging onto the devices.

You get Layer 2 topology information from looking at your switch configurations and identifying things like trunk interfaces. But not all switch-to-switch links are configured as trunks, and not all trunks are connected to other switches. This is where mapping starts to get a little more complicated.

One useful technique for figuring out Layer 2 next hop devices is to look at the MAC address table. The switch uses the MAC address table to figure out how to forward Layer 2 Ethernet frames. It maps all of the Ethernet MAC addresses that the switch knows about to physical interfaces.

If there's only one MAC address associated with a particular interface, then it's likely (but not definite) that this interface is only connected to a single physical device. If there are many MAC addresses then it's likely (but not definite) that this interface connects to another switch.

But that's a lot of work. If I have to map a network by hand using the MAC address table, I often use the trick of importing the MAC address table into a spreadsheet and sorting it by interface to see which ones appear to connect to more than one MAC.

On Cisco switches, an easier technique is to use Cisco Discovery Protocol (CDP). CDP is a Layer 2 protocol that allows Cisco devices to dynamically learn about one another. If you log into all of your switches and turn on CDP every where, then wait a few minutes for the devices to discover one another, you can get a much better first iteration of your Layer 2 map.

Once again, I pick one device that I think is in some sense central to my network and draw a little rectangle in the middle of the page to represent it. Then I draw similar boxes to represent all of the other switches that I can find by the methods described above.

Mapping a network by hand can be a time-consuming and tedious process.

Once again, this is an iterative process. Log into each of the next hop devices and repeat the procedure, again drawing all of their connections.

Phew! As you can see, mapping a network by hand can be a time-consuming and tedious process.

Even if you're using a **drawing program** like Visio, Gliffy, or LucidChart, you'll still need to manually collect all of the information you'll be plotting. Drawing software only helps you draw.

But here's where **automation tools** come in handy. There are several tools on the market, some free, some rather expensive, that can map your network for you. Each one has its strengths and weaknesses, but each one can help you free up some of your time to focus on higher value tasks. For each component of network topology mapping that you're able to automate, you'll realize additional benefits in terms of time spent focusing on supporting your user's experience.

SolarWinds

The SolarWinds Network Topology Mapper generates maps that are like the first passes I do on paper; generally speaking, they're not all that meaningful. The software doesn't know anything about the functions of different network segments or systems. It also doesn't do a great job of distinguishing between the Layer 2 and Layer 3 functions of the same physical device.

It's also important to remember that the maps SolarWinds generates are essentially snap shots. They don't automatically evolve if the network topology changes. If you want to see how things have changed, you need to run the discovery tool again.

Spiceworks

Spiceworks provides a free network mapping tool that many people find useful. The tool has fewer features than SolarWinds, but it's still quite good at gathering connectivity information for an undocumented network.

netViz, net-viz, and netTerrain

One of the most advanced tools for network mapping used to be netViz, which was a very good package until it was abandoned by CA. Confusingly, there's another Python package called net-viz, which doesn't have a lot of features. Even more confusingly, Graphical Networks has developed a netViz work-alike called netTerrain. netTerrain is useful but requires significant amounts of user input to produce meaningful network maps.

Auvik

Auvik is a particularly useful tool for network mapping. It has several important advantages. It does a much more sensible job of laying out your network topology than most other systems. It also automatically updates your network maps in response to topology changes as they happen. And, it supports more data sources than other network topology mappers, such as pulling from vendor-specific APIs.

Particularly for distributed networks, Auvik can actually produce useful and usable network diagrams right out of the box.

USEFUL TOPOLOGIES

There are several useful network topologies for LAN and WAN construction. Next we'll talk about these designs and where they're best applied.

Single switch

The most basic type of small office network is a single switch. It's probably the fastest and most stable network you can build. But of course, it has the problem of being a massive single point of failure.

In a single-switch network, one switch supports everything: workstations, printers, and servers. The single switch performs the functions of access, aggregation, and network core all at once.

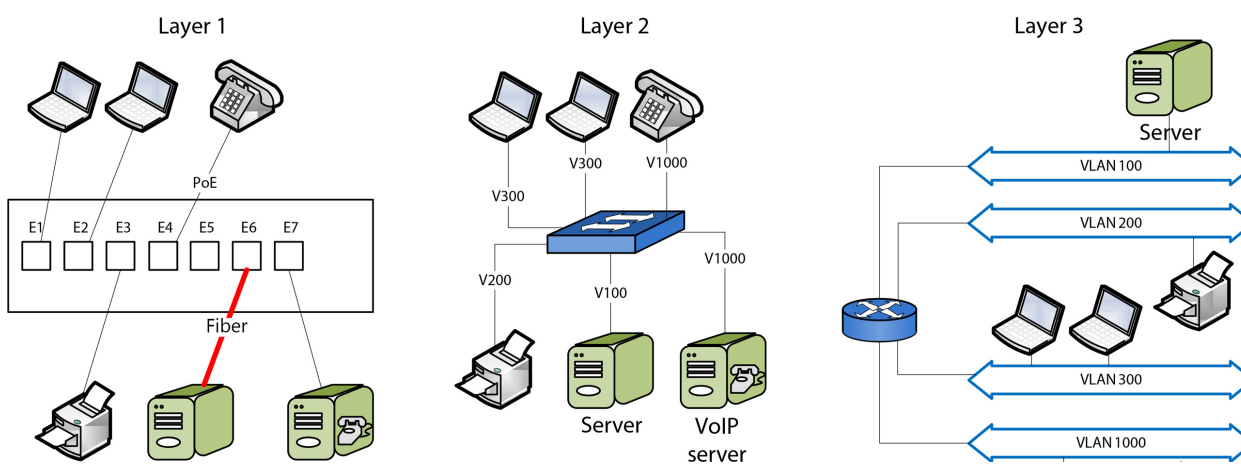


Figure 6 – Single switch small office network topology

For this single-switch network, I've shown separate Layer 1, 2, and 3 diagrams.

- The Layer 1 diagram shows Layer 1 information such as switch ports, media types, and link speeds.
- The Layer 2 diagram shows Layer 2 information, which in this case is just VLAN numbers.
- The Layer 3 picture shows how the VLANs correspond to subnets, and how the VLANs are interconnected.

The really important thing about these diagrams is to notice that the Layer 3 picture shows completely different types of information than the other two pictures. However, you'll probably have noticed that in a simple network like this one there's really no reason not to combine the Layer 1 and 2 information.

The only extra thing that might be in the Layer 1 diagram that doesn't belong in the Layer 2 picture are patch panels and physical layouts. These items may not be relevant in a very small network like this one though.

A single switch topology is useful for

- Small networks with very few devices all located in close proximity to one another.

Avoid when

- You have more devices than will fit on a single switch.
- There are geographical challenges, such as individual devices more than 100m away from the switch.

Single switch with Internet connection

Figure 7 shows a slightly more complicated single-switch network. It's a little different from the previous network because of the inescapable fact that the Internet needs to be treated differently.

The Internet is the most hostile place on Earth. Do not put your external Internet connection on a VLAN on your internal switch.

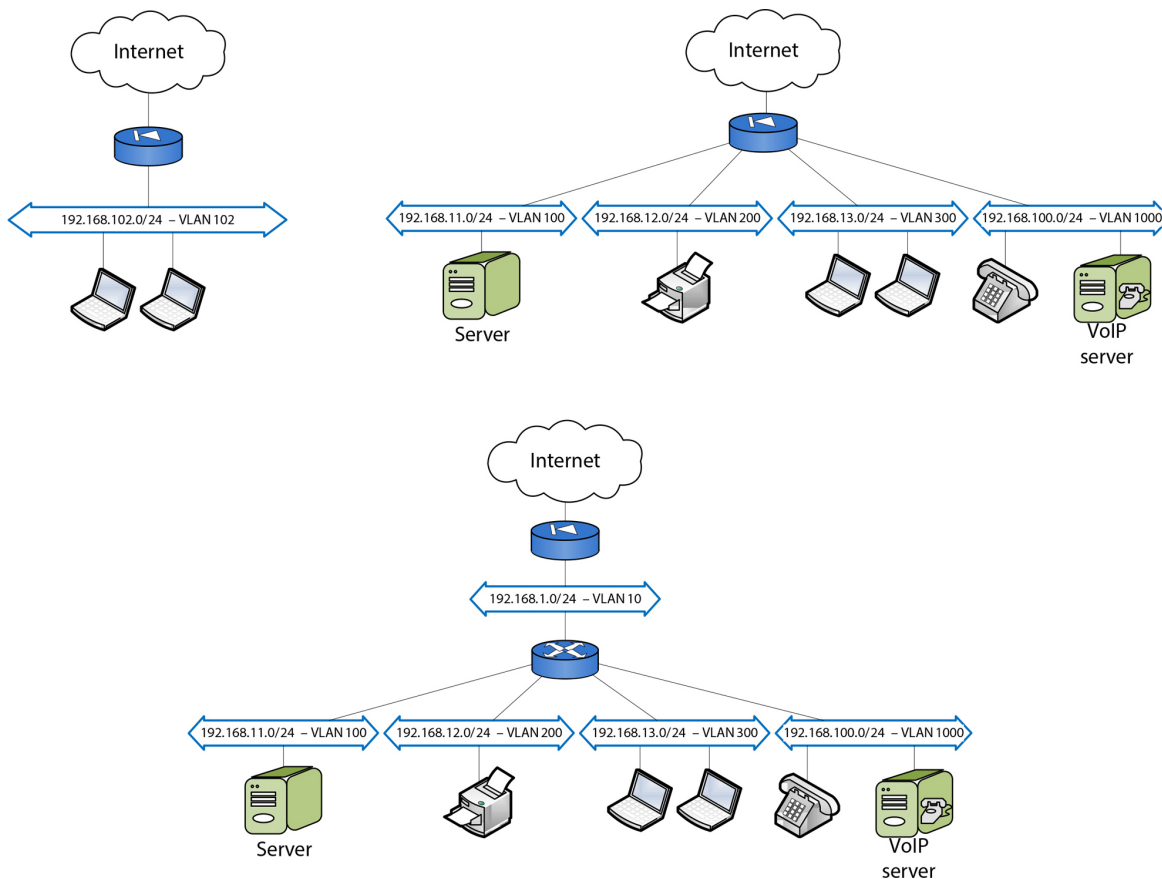


Figure 7 – Small network with Internet connection topology (Visio version)

For interest's sake, here's how the same diagram would look if the Auvik tool had mapped it.

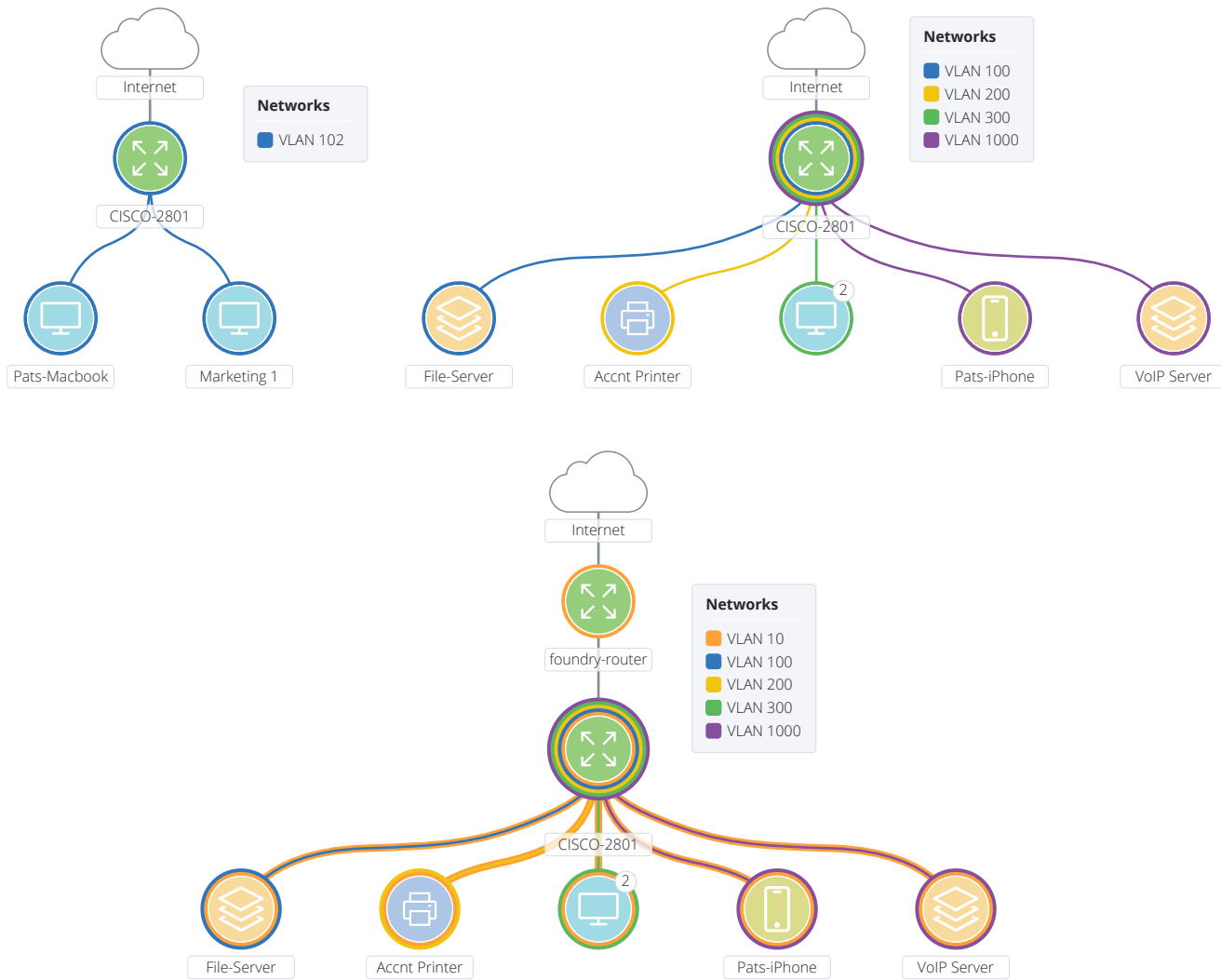


Figure 8 – Small network with Internet connection topology (Auvik version)

For this type of small network design, you really have three choices.

1. Put all the internal devices on one flat segment, together with the inside interface of the Internet firewall.
2. Split the internal network into multiple VLANs, and put every VLAN on a different interface on the firewall.
3. Interconnect the internal segments with the Layer 3 functions of the switch itself. Each of these options has its strengths and weaknesses.

The first option is great if you have a very simple network. It's certainly how most home networks are set up. But if there's any requirement to separate devices, it's not useful.

You might separate the devices onto different segments for security reasons. For example, devices like IP phones and printers that run their own IP stacks but can't run anti-virus software are vulnerable to certain types of attacks, so I like to separate them from sensitive servers and workstations.

The second option, putting each segment onto a different firewall interface, is one way to solve the security problem. It provides good separation of the logical segments. But it has two main drawbacks.

First, it doesn't scale well. If you need to expand the number of segments or even the number of devices, you'll quickly run out of resources on that firewall, both interfaces and throughput.

Second, it can be quite awkward to configure firewall rules that will allow arbitrary Windows server traffic. If you wind up configuring wide open firewall rules between two segments, you probably shouldn't be using a firewall.

So the third option is to use the Layer 3 routing features of the switch to separate segments. You can still put some segments on separate firewall interfaces if you really want the secure separation. The biggest disadvantage with option 3 is that adding Layer 3 switching features makes the switch more expensive.

A single switch with Internet topology is useful for

- Small networks with Internet connections.

Avoid when

- Your network is larger or geographically dispersed.
- You expect your network to grow quickly.
- You want to provide a web site to the external Internet. (In that case, you should really build a proper DMZ.)

Touchdown segment

The touchdown segment is a technique for connecting a LAN to one or more WAN routers. The idea is very simple: Create a separate VLAN and put the other routers on it, then connect this segment to the internal network through either a router or a firewall.

It's a good technique for several reasons.

First, you might have security concerns about the external networks. In this case, the router or firewall that connects the internal network to the touchdown segment can be used as a security control point. It can have access control lists (ACL) or firewall rules to restrict what traffic can enter the internal network from the external routers.

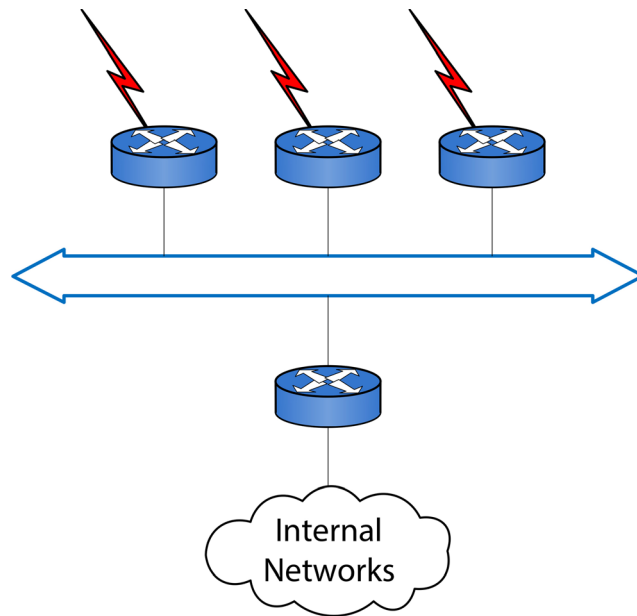


Figure 9 – Touchdown segment topology

Second, it's often necessary to separate the routing protocols. You don't want external networks controlling your internal routing. At the same time, you often do need to learn about the external network's routing, and you similarly need that network to know about at least some of your internal routes. So it makes sense to carefully redistribute and filter the routes in both directions.

Third, related to this routing issue, reaching the external networks through a common point on your network means the router or firewall is an excellent place to summarize routing tables. This greatly simplifies the routing of both internal and external networks, which in turn contributes to better routing convergence times and more stable networks.

Of course, the touchdown segment is also obviously useful for simple reasons of organization. If you can put all of your WAN and external-facing routers in a common corner of your network, it's very easy to handle things like troubleshooting or expanding the network design.

A touchdown segment topology is useful for

- Connecting WAN routers and external networks to an internal LAN

Avoid when

- You're connecting to an extremely large number of external networks.
- The external WAN connections have a very high aggregate bandwidth (in which case the touchdown segment could present a bad congestion point).

Three-tier

The classic three-tier network has a core tier, aggregation tier, and access tier, as shown in Figure 10. You'll often see this kind of topology in textbooks and in Cisco's reference architectures for large corporate networks.

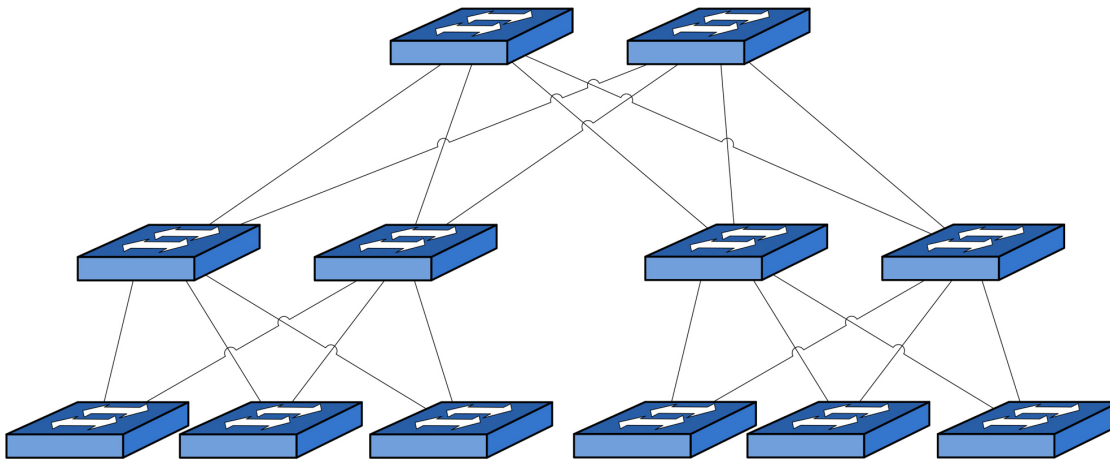


Figure 10 – Three-tier large LAN topology

The main concept of the three-tier network is that the normal end devices, such as workstations and printers, should connect to access-tier switches. Access switches then connect to aggregation switches, and aggregation switches connect to core switches.

Three-tier design is really a Layer 1 concept. There's nothing in this basic design that dictates how VLANs or IP segments should be distributed or isolated. In principle, you could do just about anything you wanted. For example, you could have a particular group of users geographically dispersed among any of the access-tier switches.

The strength of the three-tier network architecture is its scalability. If the access switches each have 48 ports and the aggregation switches also have 48 ports, and there are two core switches also with 48 usable ports, then your network could readily support over 100,000 end devices. You could even increase this further by using larger switches at the core or aggregation tiers, or by expanding the core to include more physical switches.

In practice, if your network is large enough to need this many access ports for end devices, then you probably need to do a lot of segregation. For this reason, it would be unusual to freely distribute VLANs across the network.

Further, if you have this many end devices, you'll probably encounter serious congestion problems if all the traffic between network segments concentrates on the uplinks between the aggregation and core switches.

In Figure 11, I show a subtle modification to the standard three-tier concept. Instead of each switch connecting to one upstream switch, I place two redundant connections at each point. Even the connection between the two core switches is redundant.

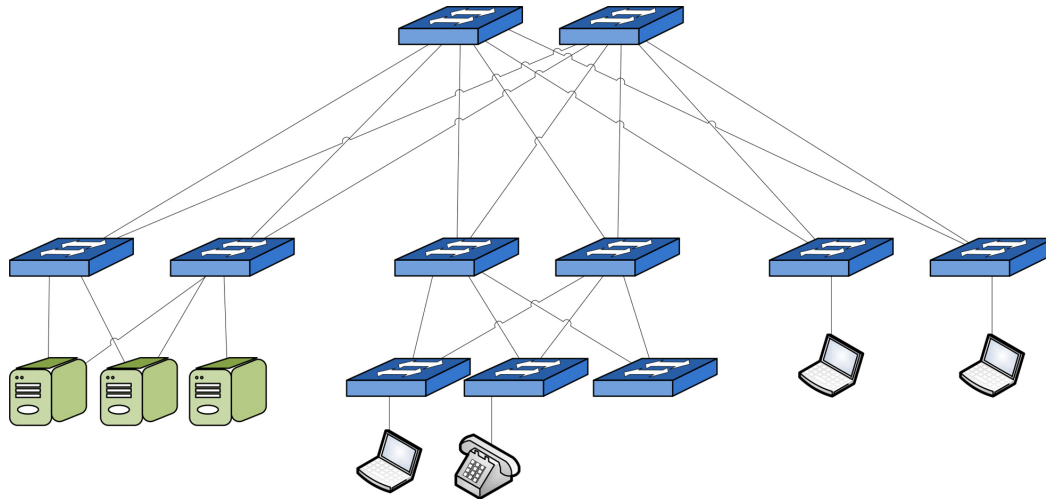


Figure 11 – Modified three-tier topology

The result is that I've treated the aggregation-tier switches in pairs, just like the core switches. You could even opt to have a separate link between the two paired aggregation switches. But doing this would require being careful about your spanning tree configuration, as it might just be another chance to make loops in your network.

In the real world, where scaling to hundreds of thousands of LAN interfaces is rarely required, the three-tier network architecture is used in very limited ways.

You might have a core tier with a few switches connected to an aggregation tier. You might decide to connect critical servers directly to some of the aggregation or even the core switches and a handful of access switches that are really only used for geographical reasons to get connectivity to physically remote groups of users.

That would still be considered a three-tier network, and would likely prove to be more useful.

A three-tier topology is useful for

- Very large corporate networks with many workstations.

Avoid when

- You have a smaller network that doesn't need many end devices.
- You have lots of distinct security zones that need firewalls between them.

Web-tier

Figure 12 shows a classic web-tier network design, a standard DMZ.

A web-tier design doesn't even need to involve web servers. Any application that runs on servers and has a back-end could, in principle, use this model. However, web-based services are an extremely common way of delivering applications.

In this topology, we put a firewall in front of everything. This is the Internet firewall. Its function is to protect the web servers from direct access from the Internet.

In most cases, the firewall will do two things. It will allow access to the web server and only the web server based on its destination IP address. And it will allow access only to the specific HTTP and HTTPS TCP ports (80 and 443) on this web server.

In many cases, it will also do Network Address Translation (NAT) to allow the web server to have a private address but still be accessed from the public Internet.

Inside the external Internet firewall is the network segment holding the web server. The external connections can only reach this web server. In turn, the web server is allowed to send packets through a firewall to an application server on a separate internal segment. And the application server can communicate to a database server on yet another segment, also behind a firewall.

In some DMZ designs the web and application servers might be combined, or the application and database servers might be combined. It's an extremely bad idea, though, to put your real data on the web server unless that data is completely public, like brochure material. Web servers get compromised so often that you have to operate on the assumption it will happen sooner or later.

There are actually a few ways to construct a web-tier network. The simplest is to use the same single firewall for all three server segments as well as the Internet access. This is a simple design, but it requires special attention to firewall rules. One of the critical features of this design is that the servers in each tier can only be accessed from the servers in the next higher tier.

A more complicated web-tier design uses different firewalls to separate the tiers. This requires either putting two firewalls on each LAN segment (Figure 13) or deploying the individual servers with multiple LAN interfaces (Figure 14).

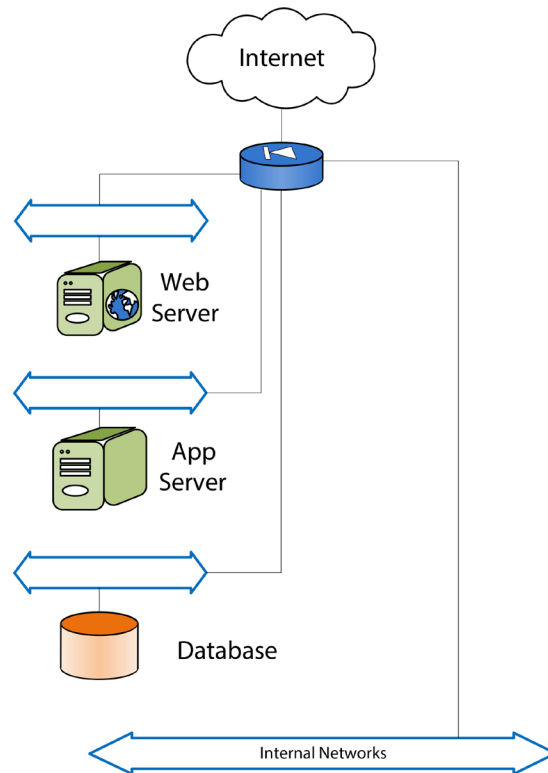


Figure 12 – Web-tier DMZ

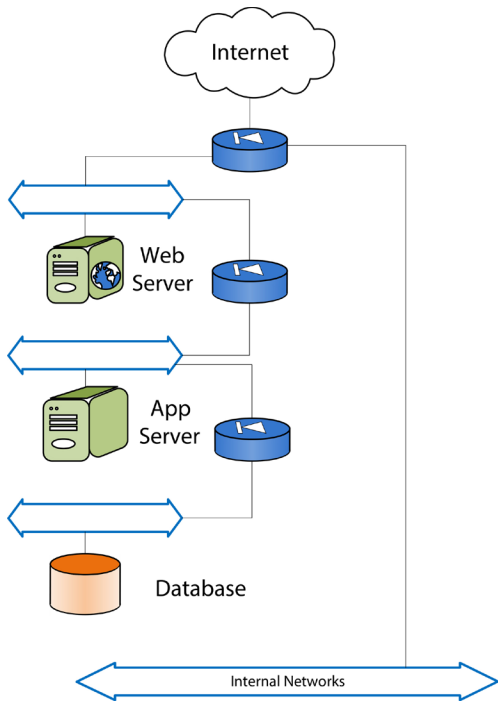


Figure 13 – Web-tier DMZ with multiple firewalls

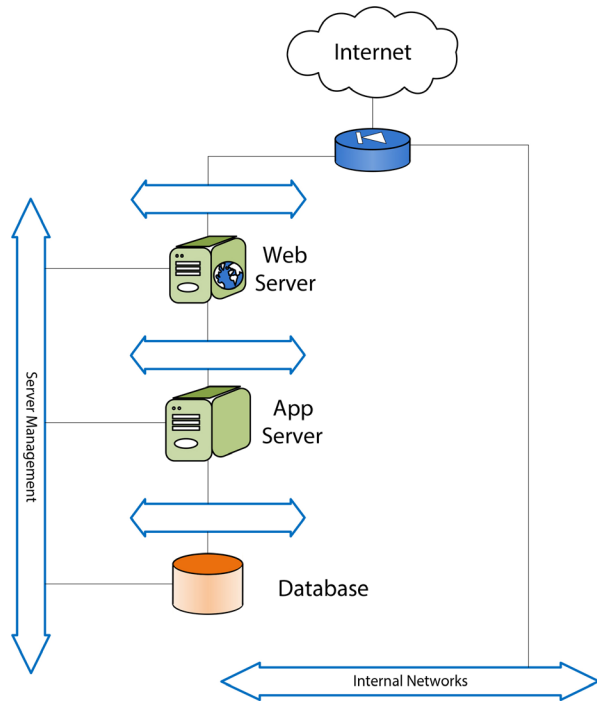


Figure 14 – Web-tier DMZ with multiple server LAN interfaces mask

I personally believe using multiple firewalls is marginally more secure because a single fire wall configuration can be compromised if an attacker is able to somehow subvert the single firewall. Then, because the single firewall connects to all segments, the attacker could gain direct access to the potentially sensitive database.

Conversely, in a multiple firewall configuration, the attacker must compromise the first tier before moving on to the second and third tiers.

The disadvantage to the multiple firewall topology is that it forces greater complexity on the servers. They must now have different routes for traffic going outward to the client than for traffic going inward to the next tier.

Single versus multiple firewalls is definitely a trade-off. You'll have to choose the balance that's right for your environment.

Another important modification to the multiple-tier web model involves adding a load balancer, as shown in Figure 15. I always put the load balancer inside the firewall.

I also like to terminate SSL sessions for HTTPS on the load balancer because it simplifies how certificates are handled for the end user. And I like to do the NAT on the load balancer because load balancers are fundamentally NAT engines. It's better and simpler to do NAT only once rather than first on the external firewall and again on the load balancer.

Finally, there are a lot of cases where you won't have a database at all. Your application might not need or use one.

The important concept of the web-tier design is that sensitive systems should not be accessible from the external network. You need to go through multiple layers to reach them, and you ideally want to build your applications so there's inspection and validation of the data at each stage.

This ensures that an attacker can't just craft a request at the web layer and pass malicious code all the way to the sensitive internal systems.

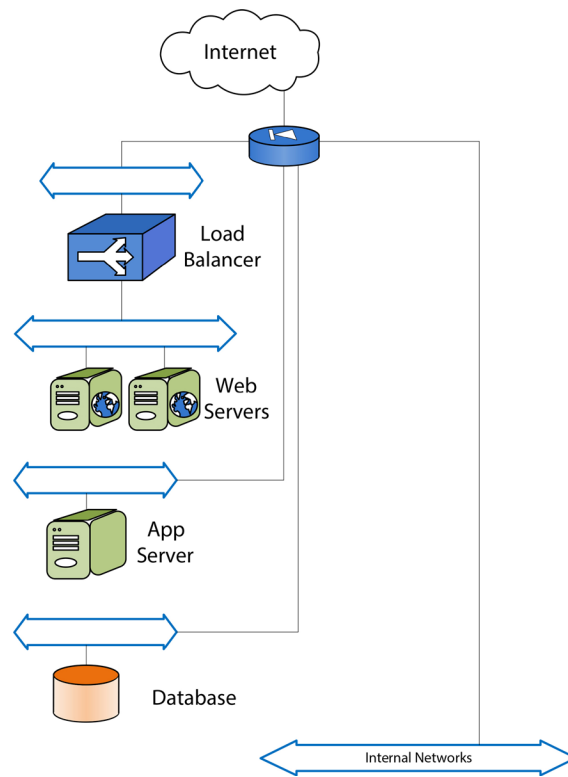


Figure 15 – Web-tier DMZ with load balancermask

A web-tier topology is useful for

- Internet DMZs where you need to provide access to users on an external untrusted network like the Internet.

Avoid when

- Your applications touch only trusted networks.
- Your applications can't be readily separated into tiers.

Hub-and-spoke

Hub-and-spoke networks are similar to the three-tier LANs we discussed earlier. Or, more precisely, three-tier networks are specific examples of a hub-and-spoke topology. Hub-and-spoke networks are also a common WAN topology.

In a basic hub-and-spoke network topology, there's a single central device that connects to several remote devices. Usually these will be Layer 3 connections in a WAN, but they could be Layer 2 trunk connections in a LAN.

Layer 3 hub-and-spoke networks scale very well because you can increase the size of the hub by creating a ring of hub routers as the routing core.

The trouble with this model is that the hub device is a single point of failure for the entire network. So a common variant uses dual redundant hubs. Both single hub and dual hub options are shown in Figure 16.

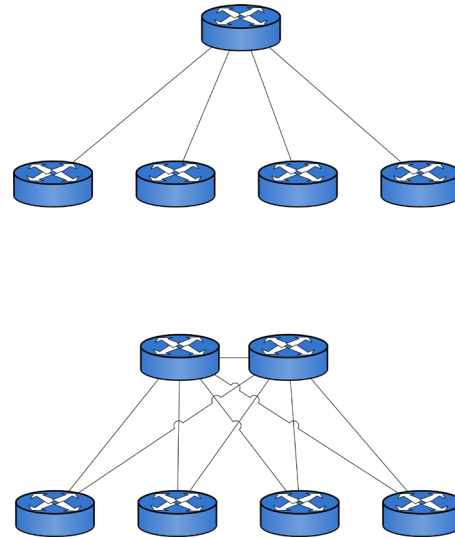


Figure 16 – Single and dual hub-and-spoke topologies

A more typical modern WAN topology is the Multiprotocol Label Switching (MPLS) cloud. It looks somewhat like a hub-and-spoke network design, but in fact most MPLS WANs allow any-to-any communication. This means any of the spoke routers can communicate with any other spoke router, as well as the hub.

The key defining feature of a hub-and-spoke topology is that the spoke nodes are not inter connected to one another. If I can connect the network in a flatter topology, it will generally provide greater reliability overall.

To see why this is, consider the difference between a triangle and two separate remote nodes, each individually connected to a central node. It's clear the triangle topology will be more resilient against different failure modes.

A hub-and-spoke topology is useful for

- Connecting geographically diverse network zones.
- Campus LANs or tall buildings where you need to interconnect many remote areas to a common central data center.

Avoid when

- There's a lot of spoke-to-spoke communication. (For example, this kind of topology wouldn't work well if you have VoIP users calling one another between spokes.)
- The main data center is not located at the hub.

Ring

A ring topology features three or more interconnected switches. The nice thing about a ring topology is that it provides redundancy with a minimum number of links. Any one device or link can fail without disrupting connectivity for any of the others.

I usually make rings in a small part of the network, then connect other switches or routers to the ring using redundant links to two different ring members. I show this design in Figure 17.

There are two particularly useful places to deploy ring structure in networks. One is a ring of access switches and the other is a ring of core switches.

In principle, you could also have a ring of aggregation switches in a three-tier network model. But it's hard to imagine a network large enough to require such a structure yet small enough to not suffer from spanning tree hop count problems.

The old spanning tree rule of thumb is that a network becomes unstable if you have more than 16 hops between any two devices. You'll typically get a high number of hops when the ring contains a dozen or more switches, or in smaller rings when each switch in the ring has many downstream switches.

I sometimes like to use rings of access switches in cases where, usually for reasons of physical geography, I can't run all of the individual access switches all the way back to the core but I still need the redundancy of multiple connections.

For example, I might connect the switch on the first floor to the switch on the second floor, which connects to the switch on the third floor, which in turn connects back to the first floor. Then I separately connect two of these switches to two different switches in my network core for full redundancy without requiring a full mesh and using as few expensive cable runs as possible between the floors.

I also like to use rings of core switches in situations where I need to have more than two core switches or routers.

For example, this strategy was useful in a data center network I designed. In that case, the client had a relatively large number of access switches connected directly to the core switches, but it was more cost-effective to use fixed form-factor core switches than to go to a higher density chassis switch at the core.

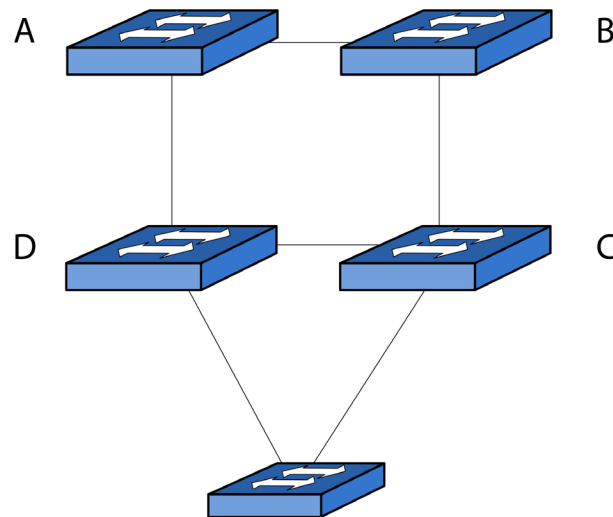


Figure 17 – Ring topology

In the end, we connected several of the core switches together in a ring. The access switches were then connected to the core switches using redundant uplinks to two of the core switches. A similar design is shown Figure 18.

Here, the core switches are interconnected using the highest speed links available. That might be 10Gb or 40Gb Ethernet if the switches support it, or it could be a LACP bundle of links.

The ring in this case is broken up into two groups. Switches A and B are used for server connections in the data center and switches C and D are used for downstream connections to the user access switches. Each downstream access switch has redundant uplinks to the core switches for fault tolerance.

Note that the higher speed links are indicated with thicker lines, a drawing technique we discussed earlier.

A ring topology is useful for

- Interconnecting a group of switches in a redundant topology that uses as few links as possible.
- Situations where you need to save cabling costs due to geographical problems.
- Conserving switch uplink ports when bandwidth is not an issue.

Avoid when

- The number of switching hops is high.
- Your network has high traffic, particularly if there's real-time traffic such as voice or video. The ring topology aggregates data flows onto a small number of backbone links. This can cause congestion, which results in latency and jitter problems.

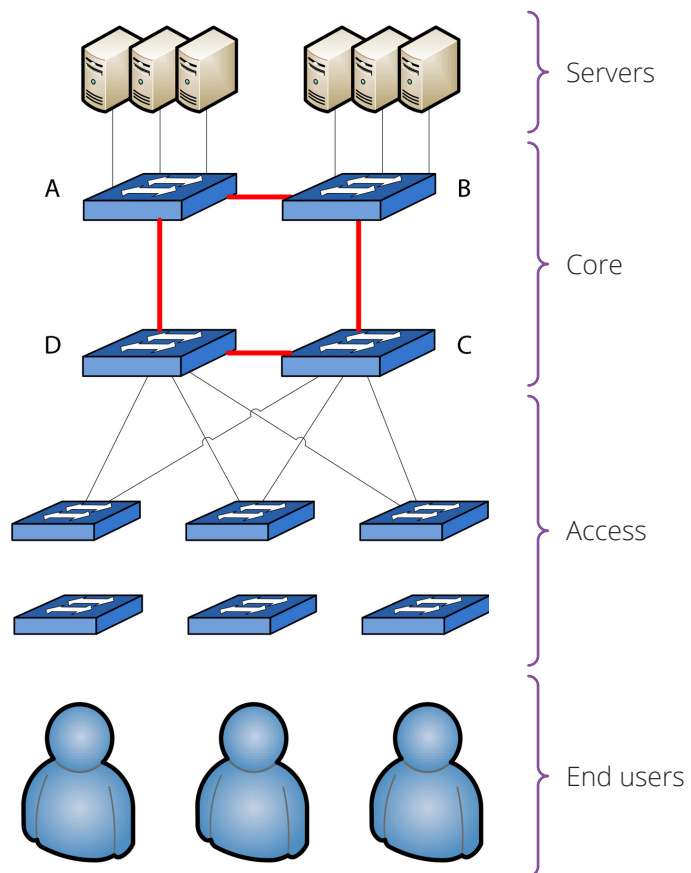


Figure 18 – Backbone rings

Internal firewalls

Security is becoming an increasing vital consideration in network design. I have a lot of clients who deploy firewalls to separate internal security zones within their networks, not just at the perimeter.

There are a lot of reasons to do this. You might have a set of servers containing extremely sensitive data, such as credit card information, private customer information, financial records, human resources data, or really anything that could be sensitive to your business.

It might be legally sensitive, or it might just be the secret sauce that gives your business its competitive advantage. Whatever the reason, access to that data must be controlled, and firewalls are often a good mechanism for controlling access.

Figure 19 shows an example of an internal security zone. In this picture, I show three parallel internal security zones for different applications.

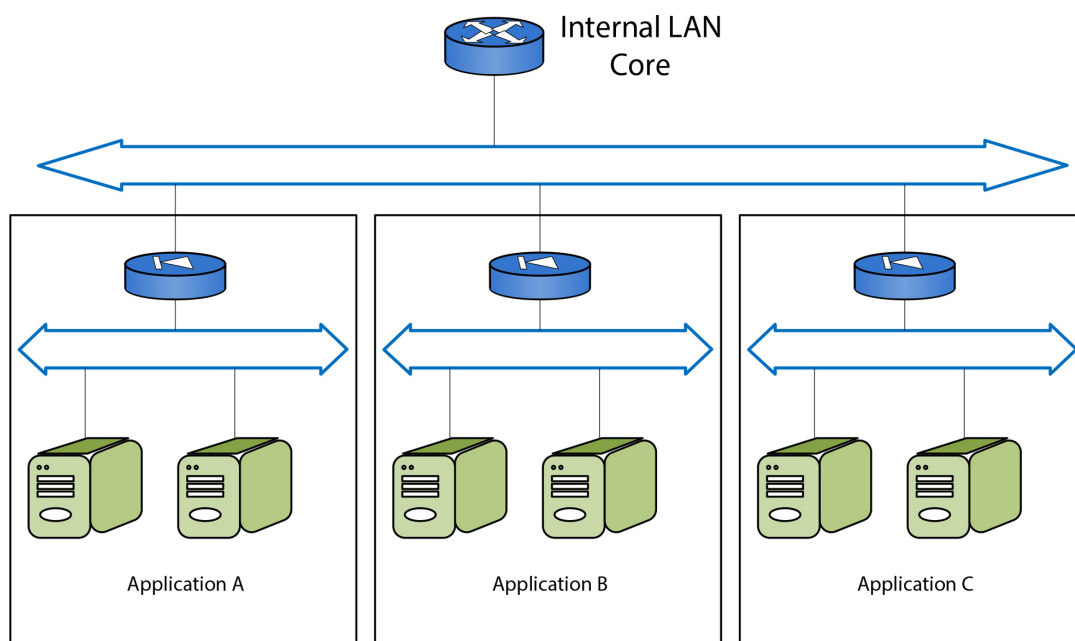


Figure 19 – Internal firewall topology

I like to deploy internal firewalls in a way that's a little different from perimeter firewalls. For perimeter firewalls, I like to have VPN and IDS/IPS inspection features. These features are important when facing the hostile public Internet.

On the other hand, the important thing for an internal firewall is simply controlling access to the applications and servers. So internal firewalls don't need the same web-tier infrastructure design as an Internet DMZ.

Instead, for internal firewall segregated security zones, I like to build the zones in parallel on the same set of switches. If the server environment is virtualized, then I often virtualize the firewalls too. If the firewalls are virtualized, then I give every separate parallel application zone its own separate virtual firewall.

If the server environments are physical, then I generally prefer to use a single common fire wall with one interface on the corporate side and a separate interface in each internal zone.

I tend to avoid using a single common physical firewall, whether virtualized or physical, when I need a lot of throughput into and out of the application zones.

For example, if the amount of network traffic reaching the application inside such a zone is so high that it would require an extremely expensive firewall, then an internal firewall design won't work well. In that case, it might make sense to re-engineer the application so it has a web front-end and put the heavy traffic on an application back-end behind the web server. Then the internal security zone model works very well again.

Internal firewall topology is also not useful when there's a lot of traffic among the different secure application zones. If your applications exchange huge amounts of data, they probably don't want to be separated. In that case it might make sense to put them all in a single secure zone.

An internal firewall topology is useful for

- Protecting sensitive data or applications that are internal to your network, intended for internal consumption, but where the data is so sensitive that it requires special access controls. (The "inside" of the firewall is the secure zone, and the "outside" is the corporate network.)
- Separating software development and test environments from your production data. (The "inside" or secure part of the network is the corporate network and the "outside" is the development or test zone.)

Avoid when

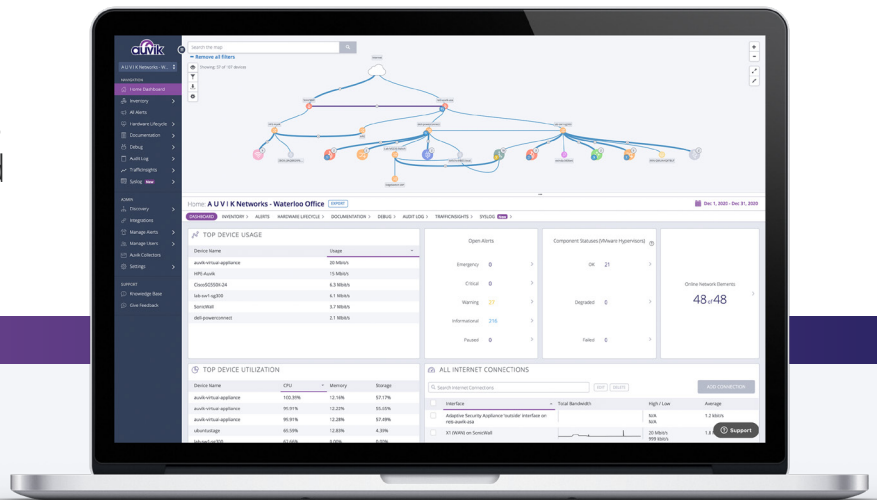
- You need more throughput into and out of the application zones.
- There's a lot of traffic among the different secure application zones.

HELPFUL RESOURCES

In this short guide, we've covered the basics of network design and documentation. If you want to dig deeper, here are some recommended resources.

- An excellent discussion of the theory of TCP/IP can be found in the first volume of *TCP/IP Illustrated (2nd edition)* by W. Richard Stevens. The book was published in 1993 so it's a little dated, but it does a great job of explaining how the protocols work.
- For network topology and network design, there are two books that I like to recommend. The first is *Top-Down Network Design (3rd edition)* by Priscilla Oppenheimer. The second is my own book, *Designing Large-Scale LANs*.
- For larger and more complicated networks, you'll want to dig into the theory and implementation of routing protocols. For this, I really like *Routing TCP/IP (2nd edition)* by Jeff Doyle and Jennifer Carroll. The two-volume set is a tad Cisco-centric but I've never encountered a better book for explaining routing protocols.

Auvik's cloud-based network management software keeps IT networks around the world running optimally



“Auvik has a great ability to

map the networks

and automatically back up the configurations.

It helps us tremendously from a troubleshooting standpoint by pinpointing where to look first.”

- CTO, IT Services Management Company

Interviewed by Forrester for the commissioned *Total Economic Impact Of Auvik's Network Management Solution* study.



To see Auvik in action on your own network, [start a 14-day free trial today.](#)

Sales: 1-226-210-4433