F5 STUDY GUIDE

101 – Application Delivery Fundamentals

Eric Mitchell
Channel SE, East US and Federal
F5 Networks

Contents

Overview

Welcome to the F5 Networks 101 - Application Delivery Fundamentals compiled Study Guide. The purpose of this guide is to help you prepare for the F5 101 - Application Delivery Fundamentals exam. The contents of this document are based on the 101 - Application Delivery Fundamentals Blueprint Guide.

This study guide provides students with some of the basic foundational knowledge required to pass the exam.

This study guide is a collection of information and therefore not a completely original work. The majority of the information is compiled from sources that are located on the Internet. All of the information locations are referenced at the top of each topic instead of in an Appendix of this document. This was done to help the reader access the referenced information easier without having to search through a formal appendix. This guide also references a book that should be basic reading for some of the topics on this exam.

The F5 Certified team provides an official 101 - Application Delivery Fundamentals Study Guide to all candidates. The F5 Certified Study Guide is a list of reading material that will help any student build a broad base of general knowledge that can assist in not only their exam success but also in becoming a well-rounded systems engineer. The Resource Guide will be available to the candidate through the certification.f5.com website once they are qualified for the Application Delivery Fundamentals exam.

There are not any pre-requisite to this exam.

This guide was prepared by an F5 employee but is not an official F5 document and is not supported by F5 Networks.

Reading = Knowledge = Power

Printed References

These referenced books are important and should be considered basic reading material for this exam.

(Ref:1) Kozierok, Charles M. 2005. The TCP/IP Guide. No Starch Press, Inc. San Francisco, CA. 94103. ISBN 1-59327-047-X pp 947 -1080

SECTION 1 - OSI

Objective - 1.01 Explain, compare, and contrast the OSI layers

1.01 – Describe the function of each OSI layer

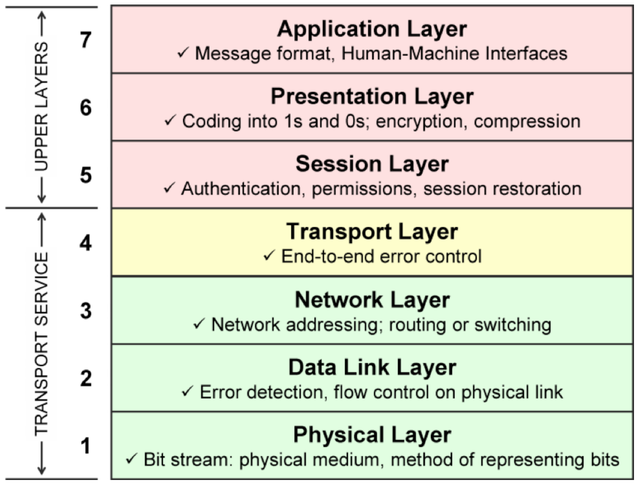
Ref: 1, pp. 168-181.

Networking Basics: Part 17 - The OSI Model

The OSI Model's Seven Layers Defined and Functions Explained

The OSI Model

The term OSI Model is short for Open System Interconnection Basic Reference Model. The OSI Model consists of seven different layers. Each layer of the model is designed so that it can perform a specific task, and facilitates communications between the layer above it and the layer below it. You can see what the OSI Model looks like in the figure below.


UPPER LAYERS
TRANSPORT SERVICE
7
Application Layer
✓ Message format, Human-Machine Interfaces
6
Presentation Layer
✓ Coding into 1s and 0s; encryption, compression
5
Session Layer
✓ Authentication, permissions, session restoration
4
Transport Layer
✓ End-to-end error control
3
Network Layer
✓ Network addressing; routing or switching
2
Data Link Layer
✓ Error detection, flow control on physical link
1
Physical Layer
✓ Bit stream: physical medium, method of representing bits

The Application Layer

The top layer of the OSI model is the Application layer. The first thing that you need to understand about the application layer is that it does not refer to the actual applications that users run. Instead, it provides the framework that the actual applications run on top of.

To understand what the application layer does, suppose that a user wanted to use Internet Explorer to open an FTP session and transfer a file. In this particular case, the application layer would define the file transfer protocol. This protocol is not directly accessible to the end user. The end user must still use an application that is designed to interact with the file transfer protocol. In this case, Internet Explorer would be that application.

The Presentation Layer

The presentation layer does some rather complex things, but everything that the presentation layer does can be summed up in one sentence. The presentation layer takes the data that is provided by the application layer, and converts it into a standard format that the other layers can understand. Likewise, this layer converts the inbound data that is received from the session layer into something that the application layer can understand. The reason why this layer is necessary is because applications handle data differently from one another. In order for network communications to function properly, the data needs to be structured in a standard way.

The Session Layer

Once the data has been put into the correct format, the sending host must establish a session with the receiving host. This is where the session layer comes into play. It is responsible for establishing, maintaining, and eventually terminating the session with the remote host.

The interesting thing about the session layer is that it is more closely related to the application layer than it is to the physical layer. It is easy to think of connecting a network session as being a hardware function, but sessions are established between applications. If a user is running multiple applications, several of those applications may have established sessions with remote resources at any time.

The Transport Layer

The Transport layer is responsible for maintaining flow control. An operating system allows users to run multiple applications simultaneously and it is therefore possible that multiple applications may need to communicate over the network simultaneously. The Transport Layer takes the data from each application, and integrates it all into a single stream. This layer is also responsible for providing error checking and performing data recovery when necessary. In essence, the Transport Layer is responsible for ensuring that all of the data makes it from the sending host to the receiving host.

The Network Layer

The Network Layer is responsible for determining how the data will reach the recipient. This layer handles things like addressing, routing, and logical protocols. Since this series is geared toward beginners, I do not want to get too technical, but I will tell you that the Network Layer creates logical paths, known as virtual circuits, between the source and destination hosts. This circuit provides the individual packets with a way to reach their destination. The Network Layer is also responsible for its own error handling, and for packet sequencing and congestion control.

Packet sequencing is necessary because each protocol limits the maximum size of a packet. The amount of data that must be transmitted often exceeds the maximum packet size. Therefore, the data is fragmented into multiple packets. When this happens, the Network Layer assigns each packet a sequence number. When the data is received by the remote host, that device's Network layer examines the sequence numbers of the inbound packets, and uses the sequence number to reassemble the data and to figure out if any packets are missing. If you are having trouble understanding this concept, then imagine that you need to mail a large document to a friend, but do not have a big enough envelope. You could put a few pages into several small envelopes, and then label the envelopes so that your friend knows what order the pages go in. This is exactly the same thing that the Network Layer does.

The Data Link Layer

The data link layer can be sub divided into two other layers; the Media Access Control (MAC) layer, and the Logical Link Control (LLC) layer. The MAC layer basically establishes the computer's identity on the network, via its MAC address. A MAC address is the address that is assigned to a network adapter at the hardware level. This is the address that is ultimately used when sending and receiving packets. The LLC layer controls frame synchronization and provides a degree of error checking.

The Physical Layer

The physical layer of the OSI model refers to the actual hardware specifications. The Physical Layer defines characteristics such as timing and voltage. The physical layer defines the hardware specifications used by network adapters and by the network cables (assuming that the connection is not wireless). To put it simply, the physical layer defines what it means to transmit and to receive data.

1.01 - Differentiate between the OSI layers

OSI (Open Source Interconnection) 7 Layer Model

OSI Layers

Application (Layer 7)

This layer supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail, and other network software services.

Presentation (Layer 6)

This layer provides independence from differences in data representation (e.g., encryption) by translating from application to network format, and vice versa. This layer formats and encrypts data to be sent across a network, providing freedom from compatibility problems. It is sometimes called the syntax layer.

Transport (Layer 4)

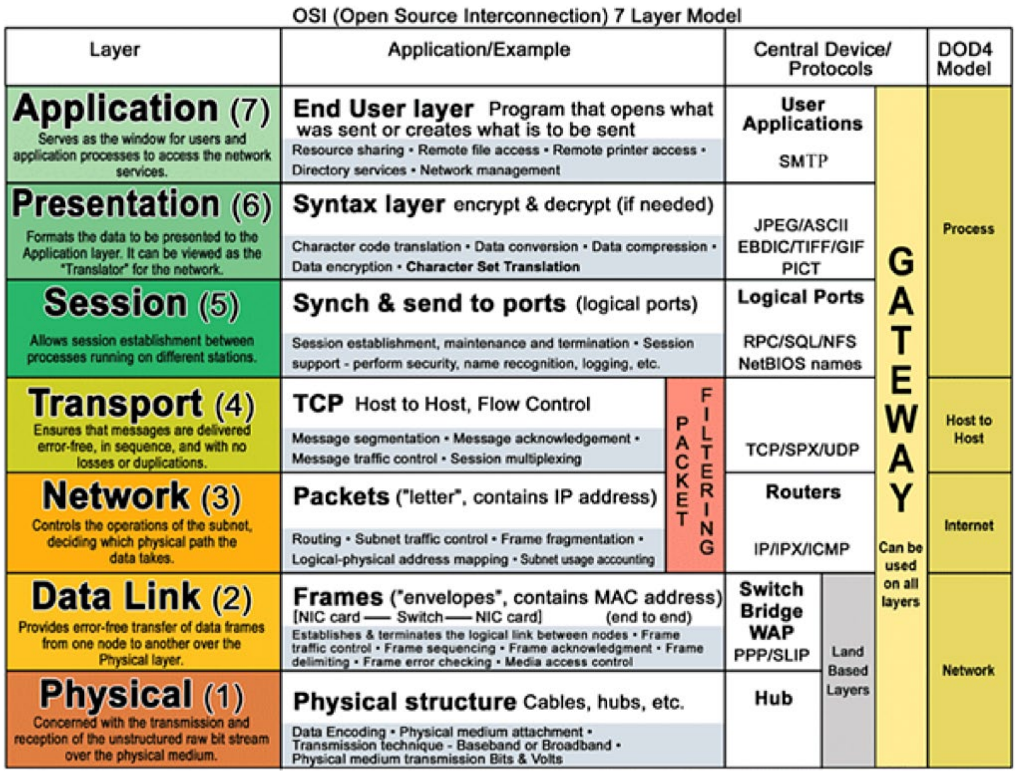
This layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer.

Network (Layer 3)

This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing.

Network (Layer 2)

This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing.



OSI (Open Source Interconnection) 7 Layer Model

Layer	Application/Example		Central Device/ Protocols			DOD4 Model
Application (7) Serves as the window for users and application processes to access the network services.	**End User layer** Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management		**User Applications** SMTP		GATEWAY Can be used on all layers	Process
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	**Syntax layer** encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • **Character Set Translation**		JPEG/ASCII EBDIC/TIFF/GIF PICT			
Session (5) Allows session establishment between processes running on different stations.	**Synch & send to ports** (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.		**Logical Ports** RPC/SQL/NFS NetBIOS names			
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	**TCP** Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	PACKET FILTERING	TCP/SPX/UDP			Host to Host
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	**Packets** ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting		**Routers** IP/IPX/ICMP			Internet
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	**Frames** ("envelopes", contains MAC address) [NIC card — Switch — NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control		**Switch** **Bridge** **WAP** PPP/SLIP	Land Based Layers		Network
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	**Physical structure** Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts		**Hub**			

1.01 - Describe the purpose of the various address types at different OSI layers

OSI layers functional

1. Physical - Hubs, Repeaters, Cables, Optical Fiber, SONET/SDN,Coaxial Cable, Twisted Pair Cable and Connectors
2. Data Link - 802.11 (WLAN), Wi-Fi, WiMAX, ATM, Ethernet, Token Ring, Frame Relay, PPTP, L2TP and ISDN
3. Network - IPv4, IPV6, IPX, OSPF, ICMP, IGMP and ARP
4. Transport - TCP, SPX and UDP
5. Session layer - Logical Ports 21, 22, 23, 80 etc.
6. Presentation layer – SSL, WEP, WPA, Kerberos,
7. Application Layer - DHCP, DNS, FTP, HTTP, IMAP4, NNTP, POP3, SMTP, SNMP, SSH, TELNET and NTP

Objective - 1.02 Explain Protocols and Technologies Specific to the Data Link Layer

1.02 - Explain the purpose of a switch's forwarding database

Forwarding Database

A forwarding database is a table used by a Layer 2 device (switch/bridge) to store the learned MAC addresses of nodes on the attached local broadcast domain/domains (VLANS) and the port (interface) that MAC address was learned on. The MAC addresses are learned transparently as the switch forwards traffic.

How it works

When an Ethernet frame arrives at a Layer 2 device, the Layer 2 device will inspect the source MAC address of the frame and associate it to the port that the frame arrived on in the forwarding database. This simply creates a table that can be cross-referenced for device locations. When the table is populated it allows the Layer 2 device to look at the destination MAC address of the arriving Ethernet frame and find the destination port for that MAC address, to know where to send that specific Ethernet frame. If the FDB table doesn't have any information on that specific MAC address it will flood the Ethernet frame out to all ports in the broadcast domain (VLAN).

1.02 - Explain the purpose and functionality of ARP

Address Resolution Protocol (ARP)

ARP

ARP defines the exchanges between network interfaces connected to an Ethernet media segment in order to map an IP address to a link layer address on demand. Link layer addresses are hardware addresses (although they are not immutable) on Ethernet cards and IP addresses are logical addresses assigned to machines attached to the Ethernet. Link layer addresses may be known by many different names: Ethernet addresses, Media Access Control (MAC) addresses, and even hardware addresses.

Address Resolution Protocol (ARP) exists solely to glue together the IP and Ethernet networking layers. Since networking hardware such as switches, hubs, and bridges operate on Ethernet frames, they are unaware of the higher layer data carried by these frames. Similarly, IP layer devices, operating on IP packets need to be able to transmit their IP data on Ethernets. ARP defines the conversation by which IP capable hosts can exchange mappings of their Ethernet and IP addressing.

ARP is used to locate the Ethernet address associated with a desired IP address. When a machine has a packet bound for another IP on a locally connected Ethernet network, it will send a broadcast Ethernet frame containing an ARP request onto the Ethernet. All machines with the same Ethernet broadcast address will receive this packet. If a machine receives the ARP request and it hosts the IP requested, it will respond with the link layer address on which it will receive packets for that IP address.

Once the requestor receives the response packet, it associates the MAC address and the IP address. This information is stored in the ARP cache.

1.02 – Explain the purpose and functionality of MAC addresses

Ethernet at the Data Link Layer

MAC Addresses

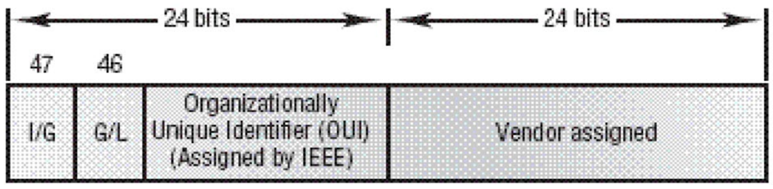
Every network device has a unique physical identity that is assigned by the manufacturing vendor is called MAC address or Ethernet address. The MAC address is also known as the hardware address while the IP address is the logical address of the device. The MAC address is defined in the Hexadecimal format generally. It consists of 6-byte (48 bits) where the first three bytes are used as the identity of the vendor and the last three bytes are used as the node identity. The MAC address works on the MAC sub-layer of the data link layer of the OSI model.

Switches give network managers the ability to increase bandwidth without adding unnecessary complexity to the network. Layer 2 data frames consist of both infrastructure content, such as end user content and MAC

Media Access Control address also known as Ethernet address. At Data Link layer, no modification is required to the MAC address of the data frame when going between like physical layer interfaces, such as from Ethernet to Fast Ethernet. However, changes to Media Access Control (MAC) address of the data frames might occur when bridging between unlike media types such as FDDI and Ethernet or Token Ring and Ethernet.

Switches learn the MAC address and build a table on the base of MAC addressing of the LAN segment called MAC Address Table. The Address Resolution Protocol (ARP) is the protocol that resolves the IP addresses into MAC addresses. RARP, the Reverse Address Resolution Protocol is a reverse of ARP and resolves MAC addresses into IP addresses.

The MAC layer of the Gigabit Ethernet is similar to those of standard Ethernet and Fast Ethernet. Media Access Layer of Gigabit Ethernet should maintain full duplex and half duplex broadcasting. The characteristics of Ethernet, such as collision detection, maximum network diameter, repeater rules, MAC addressing and so forth, will be the same of the Gigabit Ethernet. Support for half duplex Ethernet adds frame bursting and carrier extension, two functions not found in Ethernet and Fast Ethernet.



1.02 - Explain the purpose and functionality of a broadcast domain

Broadcast Domain

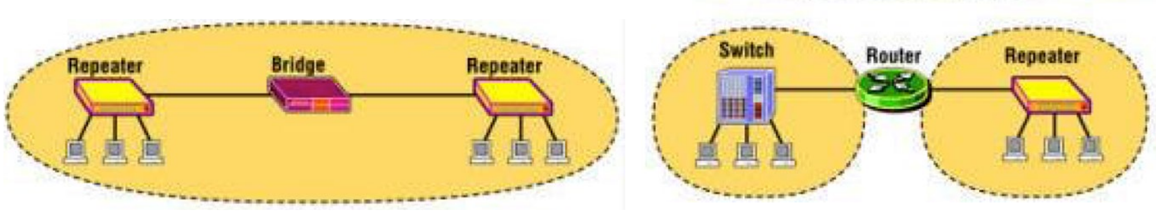
Broadcast Domain

A broadcast domain is a logical part of a network (a network segment) in which any network equipment can transmit data directly to other equipment or device without going through a routing device (assuming the devices share the same subnet and use the same gateway; also, they must be in the same VLAN).

A more specific broadcast domain definition is the area of the computer network that consists of every single computer or network-attached device that can be reached directly by sending a simple frame to the data link layer's broadcast address.

Details on Broadcast Domains

While any layer 2 device is able to divide the collision domains, broadcast domains are only divided by layer 3 network devices such as routers or layer 3 switches. Frames are normally addressed to a specific destination device on the network. While all devices detect the frame transmission on the network, only the device to which the frame is addressed actually receives it. A special broadcast address consisting of all is used to send frames to all devices on the network. The VLAN (Virtual Local Area Network) technology can also create a so-called “virtual” broadcast domain. A network built with switching devices can see each network device as an independent system. These groups of independent systems can be joined into one broadcast domain, even if the computers are not physically connected to each other. This is very useful when administrating large networks where there is the need for better network management and control.



How to Restrict the Broadcast Domain

Since a broadcast domain is the area where broadcasts can be received, routers restrict broadcasts. If a router receives a broadcast signal, it simply drops it. In other words, the edge or border router connected to the Internet will not up-broadcast or will not relay that broadcast message. This is problematic and not foolproof either. Suppose two networks exist that are connected to each other through a router and the first network has a running DHCP server that offers IP addresses to networked systems. On the other side, there is no valid DHCP server running on the second network. Offering IP addresses from the first network’s DHCP server to the second network’s systems can be a difficult task to accomplish since DHCP is a broadcast and the router that joins the networks drops the broadcast traffic. This leaves any DHCP request in the second network unanswered. Many router manufacturers provide capabilities for DHCP forwarding to solve this problem. This can be bypassed by connecting the two networks with a well-configured, Linux-based, purpose oriented software router. That will handle the job properly and prevent further issues.

1.02 - Explain the purpose and functionality of VLANs

Configuring VLANs and VLAN Groups
What is a VLAN?

Virtual Local Area Network (VLAN)

In technical terms, a VLAN is a virtual broadcast domain created inside a switch. Normally, a Layer2 device acts as a single LAN with all ports active in the LAN. In a manageable switch, the switch has the ability to be configured to group any amount of its physical ports into logical VLANs where each is an individual broadcast domain.

Are VLANs required?

It is important to point out that you don't have to configure a VLAN until your network gets so large and has so much traffic that you need one. Many times, people are simply using VLAN's because the network they are working on was already using them.

Another important fact is that, on a Cisco switch, VLAN's are enabled by default and ALL devices are already in a VLAN. The VLAN that all devices are already in is VLAN 1. So, by default, you can just use all the ports on a switch and all devices will be able to talk to one another.

When do I need a VLAN?

You need to consider using VLAN's in any of the following situations:

- You have more than 200 devices on your LAN
- You have a lot of broadcast traffic on your LAN
- Groups of users need more security or are being slowed down by too many broadcasts?
- Groups of users need to be on the same broadcast domain because they are running the same applications. An example would be a company that has VoIP phones. The users using the phone could be on a different VLAN, not with the regular users.
- Or, just to make a single switch into multiple virtual switches.

Why not just subnet my network?

A common question is why not just subnet the network instead of using VLAN's? Each VLAN should be in its own subnet. The benefit that a VLAN provides over a subnetted network is that devices in different physical locations, not going back to the same router, can be on the same network. The limitation of subnetting a network with a router is that all devices on that subnet must be connected to the same switch and that switch must be connected to a port on the router. With a VLAN, one device can be connected to one switch, another device can be connected to another switch, and those devices can still be on the same VLAN (broadcast domain).

How can devices on different VLAN's communicate?

Devices on different VLAN's can communicate with a router or a Layer 3 switch. As each VLAN is its own subnet, a router or Layer 3 switch must be used to route between the subnets.

What is a trunk port?

When there is a link between two switches or a router and a switch that carries the traffic of more than one VLAN, that port is a trunk port. A trunk port must run a special trunking protocol. The protocol used would be Cisco's proprietary Inter- switch link (ISL) or the IEEE standard 802.1q, which is the protocol F5 devices support.

What do VLAN's offer?

VLAN's offer higher performance for medium and large LAN's because they limit broadcasts. As the amount of traffic and the number of devices grow, so does the number of broadcast packets. By using VLAN's you are containing broadcasts. VLAN's also provide security because you are essentially putting one group of devices, in one VLAN, on their own network.

1.02 - Explain the purpose and functionality of link aggregation

Working with Trunks

Introducing trunks

A trunk is a logical grouping of interfaces on the BIG-IP system. When you create a trunk, this logical group of interfaces functions as a single interface. The BIG-IP system uses a trunk to distribute traffic across multiple links, in a process known as ***link aggregation***. With link aggregation, a trunk increases the bandwidth of a link by adding the bandwidth of multiple links together. For example, four fast Ethernet (100 Mbps) links, if aggregated, create a single 400 Mbps link.

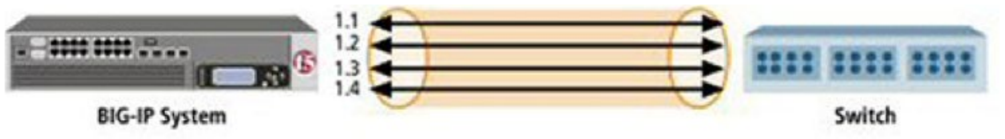
With one trunk, you can aggregate a maximum of eight links. For optimal performance, you should aggregate links in powers of two. Thus, you ideally aggregate two, four, or eight links.

The purpose of a trunk is two-fold: To increase bandwidth without upgrading hardware, and to provide link failover if a member link becomes unavailable.

You can use trunks to transmit traffic from a BIG-IP system to another vendor switch. Two systems that use trunks to exchange frames are known as peer systems.

How do trunks work?

In a typical configuration where trunks are configured, the member links of the trunk are connected through Ethernet cables to corresponding links on a peer system. Figure 9.1 shows an example of a typical trunk configuration with two peers and three member links on each peer.



A primary goal of the trunks feature is to ensure that frames exchanged between peer systems are never sent out of order or duplicated on the receiving end. The BIG-IP system is able to maintain frame order by using the source and destination addresses in each frame to calculate a hash value, and then transmitting all frames with that hash value on the same member link.

The BIG-IP system automatically assigns a unique MAC address to a trunk. However, by default, the MAC address that the system uses as the source and destination address for frames that the system transmits and receives (respectively), is the MAC address of the lowest-numbered interface of the trunk.

The BIG-IP system also uses the lowest-numbered interface of a trunk as a reference link. The BIG-IP system uses the reference link to take certain aggregation actions, such as implementing the automatic link selection policy. For frames coming into the reference link, the BIG-IP system load balances the frames across all member links that the BIG-IP system knows to be available. For frames going from any link in the trunk to a destination host, the BIG-IP system treats those frames as if they came from the reference link.

Finally, the BIG-IP system uses the MAC address of an individual member link as the source address for any LACP control frames.

Overview of LACP

A key aspect of trunks is Link Aggregation Control Protocol, or LACP. Defined by IEEE standard 802.3ad, LACP is a protocol that detects error conditions on member links and redistributes traffic to other member links, thus preventing any loss of traffic on the failed link. On a BIG-IP system, LACP is an optional feature that you can configure.

You can also customize LACP behavior. For example, you can specify the way that LACP communicates its control messages from the BIG-IP system to a peer system. You can also specify the rate at which the peer system sends LACP packets to the BIG-IP system. If you want to affect the way that the BIG-IP system chooses links for link aggregation, you can specify a link control policy.

Objective - 1.03 Explain protocols and apply technologies specific to the network layer

1.03 - Explain the purpose and functionality of IP addressing and subnetting

IP Addressing and Subnetting for New Users

Understanding IP Addresses

An IP address is an address used in order to uniquely identify a device on an IP network. The address is made up of 32 binary bits, which can be divisible into a network portion and host portion with the help of a subnet mask. The 32 binary bits are broken into four octets (1 octet = 8 bits). Each octet is converted to decimal and separated by a period (dot). For this reason, an IP address is expressed in dotted decimal format (for example, `172.16.81.100`). The value in each octet ranges from 0 to 255 decimal, or 00000000 - 11111111 binary.

Here is how binary octets convert to decimal: The right most bit, or least significant bit, of an octet holds a value of 2^0. The bit just to the left of that holds a value of 2^1. This continues until the left-most bit, or most significant bit, which holds a value of 2^7. So if all binary bits were a one, the decimal equivalent would be 255 as shown here:

1	1	1	1	1	1	1	1	
128	64	32	16	8	4	2	1	(128+64+32+16+8+4+2+1=255)

Here is a sample octet conversion when not all of the bits are set to 1.

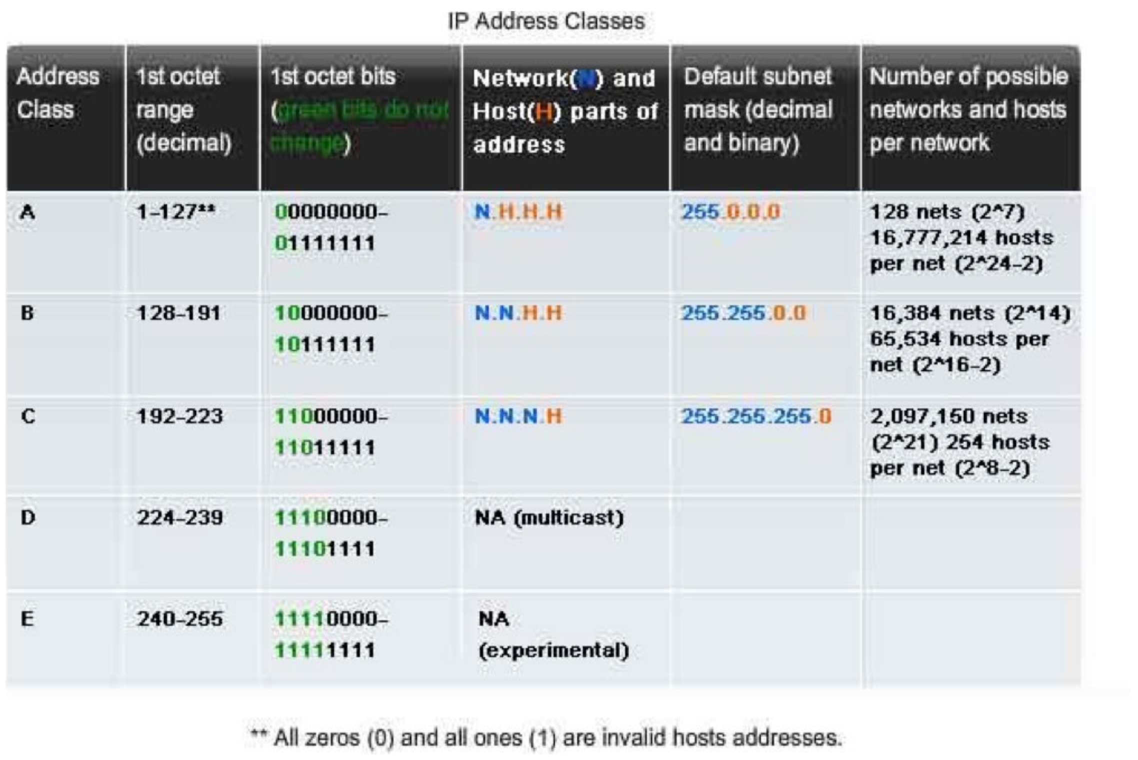
0	1	0	0	0	0	0	1	
0	64	0	0	0	0	0	1	(0+64+0+0+0+0+0+1=65)

This sample shows an IP address represented in both binary and decimal.

10.	1.	23.	19	decimal
00001010.	00000001.	00010111.	00010011	binary

These octets are broken down to provide an addressing scheme that can accommodate large and small networks. There are five different classes of networks, A to E. This document focuses on addressing classes A to C, since classes D and E are reserved and discussion of them is beyond the scope of this document.

> **Note:** Also note that the terms "Class A, Class B and so on" are used in this document to help facilitate the understanding of IP addressing and subnetting. These terms are rarely used in the industry anymore because of the introduction of classless inter-domain routing (CIDR), although CIDR is beyond the scope of this document. Given an IP address, its class can be determined from the three high-order bits. The Figure below shows the significance of the three high order bits and the range of addresses that fall into each class. For informational purposes, Class D and Class E addresses are also shown.



IP Address Classes

Address Class	1st octet range (decimal)	1st octet bits (green bits do not change)	Network(N) and Host(H) parts of address	Default subnet mask (decimal and binary)	Number of possible networks and hosts per network
A	1-127**	00000000-01111111	N.H.H.H	255.0.0.0	128 nets (2^7) 16,777,214 hosts per net (2^24-2)
B	128-191	10000000-10111111	N.N.H.H	255.255.0.0	16,384 nets (2^14) 65,534 hosts per net (2^16-2)
C	192-223	11000000-11011111	N.N.N.H	255.255.255.0	2,097,150 nets (2^21) 254 hosts per net (2^8-2)
D	224-239	11100000-11101111	NA (multicast)		
E	240-255	11110000-11111111	NA (experimental)		

** All zeros (0) and all ones (1) are invalid hosts addresses.

1.03 - Given an IP address and net mask, determine the network IP and the broadcast IP

IP Addressing and Subnetting for New Users

Network Masks

A network mask helps you know which portion of the address identifies the network and which portion of the address identifies the node. Class A, B, and C networks have default masks, also known as natural masks, as shown here:

Class A:	`255.0.0.0`
Class B:	`255.255.0.0`
Class C:	`255.255.255.0`

An IP address on a Class A network that has not been subnetted would have an address/mask pair similar to: 8.20.15.1 255.0.0.0. To see how the mask helps you identify the network and node parts of the address, convert the address and mask to binary numbers.

```
8.20.15.1 = 00001000.00010100.00001111.00000001 255.0.0.0 = 11111111.00000000.00000000.00000000
```

Once you have the address and the mask represented in binary, then identifying the network and host ID is easier. Any address bits that have corresponding mask bits set to 1 represent the network ID. Any address bits that have corresponding mask bits set to 0 represent the node ID.

`8.20.15.1 =`	`00001000.`	`00010100.`	`00001111.`	`00000001`
`255.0.0.0 =`	`11111111.`	`00000000.`	`00000000.`	`00000000`
	Net id	Host id		

Understanding Subnetting

Subnetting allows you to create multiple logical networks that exist within a single Class A, B, or C network. If you do not subnet, you are only able to use one network from your Class A, B, or C network, which is unrealistic.

Each data link on a network must have a unique network ID, with every node on that link being a member of the same network. If you break a major network (Class A, B, or C) into smaller subnets, it allows you to create a network of interconnecting subnets. Each data link on this network would then have a unique network/ sub-network ID. Any device, or gateway, connecting n networks/subnets has n distinct IP addresses, one for each network / sub-network that it interconnects. In order to subnet a network, extend the natural mask using some of the bits from the host ID portion of the address to create a sub-network ID. For example, given a Class C network of 204.17.5.0, which has a natural mask of 255.255.255.0, you can create subnets in this manner:

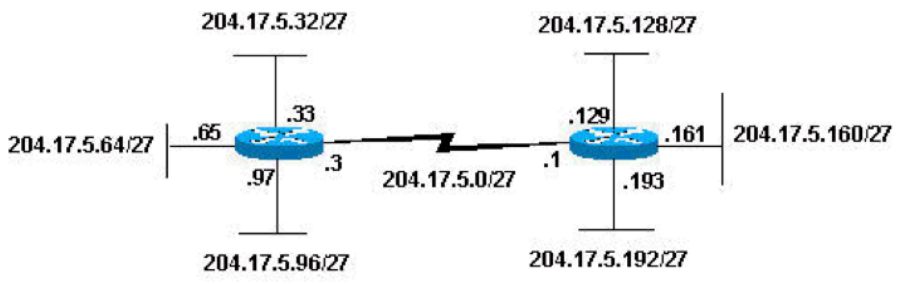
`204.17.5.0 =`	`11001100.`	`00010001.`	`00000101.`	`00000000`
`255.255.255.224 =`	`11111111.`	`11111111.`	`11111111.`	`11100000`
	`--------------`	`--------------`	`--------------`	`------| sub |`

By extending the mask to be 255.255.255.224, you have taken three bits (indicated by "sub") from the original host portion of the address and used them to make subnets. With these three bits, it is possible to create eight subnets. With the remaining five host ID bits, each subnet can have up to 32 host addresses, 30 of which can actually be assigned to a device since host ids of all zeros or all ones are not allowed (it is very important to remember this). So, with this in mind, these subnets have been created.

`204.17.5.0`	`255.255.255.224`	host address range 1 to 30
`204.17.5.32`	`255.255.255.224`	host address range 33 to 62
`204.17.5.64`	`255.255.255.224`	host address range 65 to 94
`204.17.5.96`	`255.255.255.224`	host address range 97 to 126
`204.17.5.128`	`255.255.255.224`	host address range 129 to 158
`204.17.5.160`	`255.255.255.224`	host address range 161 to 190
`204.17.5.192`	`255.255.255.224`	host address range 193 to 222
`204.17.5.224`	`255.255.255.224`	host address range 225 to 254

Note: There are two ways to denote these masks. First, since you are using three bits more than the "natural" Class C mask, you can denote these addresses as having a 3-bit subnet mask. Or, secondly, the mask of `255.255.255.224` can also be denoted as `/27` as there are 27 bits that are set in the mask. This second method is used with CIDR. With this method, one of these networks can be described with the notation pre-fix/length. For example, `204.17.5.32/27` denotes the network `204.17.5.32 255.255.255.224`. When appropriate the prefix/length notation is used to denote the mask throughout the rest of this document.

The network subnetting scheme in this section allows for eight subnets, and the network might appear as:



Notice that each of the routers in the figure is attached to four subnets, one sub-network is common to both routers. Also, each router has an IP address for each subnets to which it is attached. Each sub-network could potentially support up to 30 host addresses.

This brings up an interesting point. The more host bits you use for a subnet mask, the more subnets you have available. However, the more subnets available, the less host addresses available per subnet. For example, a Class C network of `204.17.5.0` and a mask of `255.255.255.224 (/27)` allows you to have eight subnets, each with 32 host addresses (30 of which could be assigned to devices). If you use a mask of `255.255.255.240 (/28)`, the break down is:

204.17.5.0 =	11001100.	00010001.	00000101.	00000000
255.255.255.240 =	11111111.	11111111.	11111111.	11110000
	--------------	--------------	--------------	---------| sub

Since you now have four bits to make subnets with, you only have four bits left for host addresses. So in this case you can have up to 16 subnets, each of which can have up to 16 host addresses (14 of which can be assigned to devices).

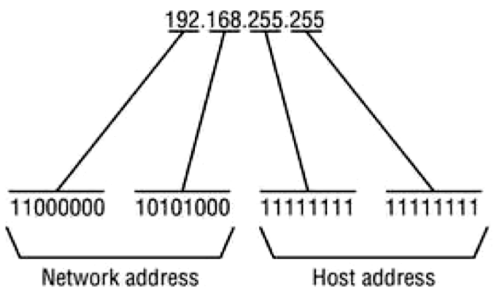
Take a look at how a Class B network might be subnetted. If you have network `172.16.0.0`, then you know that its natural mask is `255.255.0.0` or `172.16.0.0/16`. Extending the mask to anything beyond `255.255.0.0` means you are subnetting. You can quickly see that you have the ability to create a lot more subnets than with the Class C network. If you use a mask of `255.255.248.0 (/21)`, how many subnets and hosts per subnet does this allow for?

204.17.5.0 =	11001100.	00010001.	00000101.	00000000
255.255.255.224 =	11111111.	11111111.	11111111.	11100000
	--------------	--------------	--------------	------| sub |

You are using five bits from the original host bits for subnets. This allows you to have 32 subnets (2^5). After using the five bits for subnetting, you are left with 11 bits for host addresses. This allows each subnet so have 2048 host addresses (2^11), 2046 of which could be assigned to devices.

> **Note:** In the past, there were limitations to the use of a subnet 0 (all subnet bits are set to zero) and all ones subnet (all subnet bits set to one). Some devices would not allow the use of these subnets. Cisco Systems devices allow the use of these subnets when the IP subnet zero command is configured.

A broadcast address is an IP address that targets all systems on a specific subnet instead of single hosts. The broadcast address of any IP address can be calculated by taking the bit compliment of the subnet mask, sometimes referred to as the reverse mask, and then applying it with a bitwise OR calculation to the IP address in question.



Some systems that are derived from BSD use zeros broadcasts instead of ones-broadcasts. This means that when a broadcast address is created, the host area of the IP address is filled while displayed using binary values with zeros instead of ones. Most operating systems use ones broadcasts. Changing systems to use zeros-broadcasts will break some communications in the wrong environments, so the user should understand his/her needs before changing the broadcast address or type.

Math Example

If a system has the IP address 192.168.12.220 and a network mask of `255.255.255.128`, what should the broadcast address for the system be? To do this calculation, convert all numbers to binary values. For bitwise, remember that any two values where at least one value is 1, the result will be 1, otherwise the result is 0.

IP Address:	`11000000.10101000.00001100.11011100`
Reverse Mask:	`00000000.00000000.00000000.01111111`
bitwise OR:	———————————————
Broadcast:	`11000000.10101000.00001100.11111111`

Convert the binary value back to octal and the resulting value is `192.168.12.255`.

1.03 - Given a routing table and a destination IP address, identify which routing table entry the destination IP address will match

The IP Routing Table

Route Tables

Every computer that runs TCP/IP makes routing decisions. The IP routing table controls these decisions. To display the IP routing table on computers running Windows Server 2003 operating systems, you can type "route print" at a command prompt.

The following table shows an example of an IP routing table. This example is for a computer running Windows Server 2003, Standard Edition with one 10 megabit per second (Mbit/s) network adapter and the following configuration:

- IP address: 10.0.0.169
- Subnet mask: 255.0.0.0
- Default gateway: 10.0.0.1

Description	Network destination	Netmask	Gateway	Interface	Metric
Default route	0.0.0.0	0.0.0.0	10.0.0.1	10.0.0.169	30
Loopback net- work	127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
Local network	10.0.0.0	255.0.0.0	10.0.0.169	10.0.0.169	30
Local IP address	10.0.0.169	255.255.255.255	127.0.0.1	127.0.0.1	30
Multicast ad- dresses	224.0.0.0	240.0.0.0	10.0.0.169	10.0.0.169	30
Limited broadcast address	255.255.255.255	255.255.255.255	10.0.0.169	10.0.0.169	1

The routing table is built automatically, based on the current TCP/IP configuration of your computer. Each route occupies a single line in the displayed table. Your computer searches the routing table for an entry that most closely matches the destination IP address.

Your computer uses the default route if no other host or network route matches the destination address included in an IP datagram. The default route typically forwards an IP datagram (for which there is no matching or explicit local route) to a default gateway address for a router on the local subnet. In the previous example, the default route forwards the datagram to a router with a gateway address of `10.0.0.1`.

Because the router that corresponds to the default gateway contains information about the network IDs of the other IP subnets within the larger TCP/IP Internet, it forwards the datagram to other routers until the datagram is eventually delivered to an IP router that is connected to the specified destination host or subnet within the larger network.

The following sections describe each of the columns displayed in the IP routing table: network destination, netmask, gateway, interface, and metric.

Network destination

The network destination is used with the netmask to match the destination IP address. The network destination can range from `0.0.0.0` for the default route through `255.255.255.255` for the limited broadcast, which is a special broadcast address to all hosts on the same network segment.

Gateway

The gateway address is the IP address that the local host uses to forward IP datagrams to other IP networks. This is either the IP address of a local network adapter or the IP address of an IP router (such as a default gateway router) on the local network segment.

Interface

The interface is the IP address that is configured on the local computer for the local network adapter that is used when an IP datagram is forwarded on the network.

Metric

A metric indicates the cost of using a route, which is typically the number of hops to the IP destination. Anything on the local subnet is one hop, and each router crossed after that is an additional hop. If there are multiple routes to the same destination with different metrics, the route with the lowest metric is selected.

1.03 - Explain the purpose and functionality of Routing protocols

IP Routing Protocols

Routing Protocols

A routing protocol is a set of rules or standard that determines how routers on a network communicate and exchange information with each other, enabling them to select best routes to a remote network. Each router has priority knowledge only of networks attached to it directly. A router running routing protocol shares this information first, among immediate neighbors, then throughout the entire network. This way, routers gain insight knowledge of the topology of the network.

Routing protocols perform several activities, including:

- Network discovery
- Updating and maintaining routing tables

The router that sits at the base of a network maintains a routing table, which is a list of networks and possible routes known by the router. The routing table includes network addresses for its own interfaces, which are the directly connected networks, as well as network addresses for remote networks. A remote network is a network that can only be reached by forwarding the packet to another router.

Remote networks are added to the routing table in two ways:

- By the network administrator manually configuring static routes.
- By implementing a dynamic routing protocol.

Routers use Dynamic Routing protocols to share information about the reachability and status of remote networks.

IP Routing Protocols (Dynamic)

There are several dynamic routing protocols for IP. Here are some of the more common dynamic routing protocols for routing IP packets:

- RIP (Routing Information Protocol)
- IGRP (Interior Gateway Routing Protocol)
- EIGRP (Enhanced Interior Gateway Routing Protocol)
- OSPF (Open Shortest Path First)
- IS-IS (Intermediate System-to-Intermediate System)
- BGP (Border Gateway Protocol

Advantages of dynamic routing protocols

- Dynamic routing protocols update and maintain the networks in their routing tables.
- Dynamic routing protocols not only make a best path determination to various networks, they will also determine a new best path if the initial path becomes unusable or there is a change in the topology.

- Routers that use dynamic routing protocols automatically share routing information with other routers and compensate for any topology changes without involving the network administrator.

1.03 - Explain the purpose of fragmentation

IP Fragmentation

Why does fragmentation occur?

Fragmentation happens when a large IP datagram has to travel through a network with a maximum transmission unit (MTU) that is smaller than the size of the IP datagram. If an IP datagram that is bigger than 1500 bytes (typical MTU size) is sent on an Ethernet network, the datagram needs to be fragmented prior to being placed on the network. The network packets are then assembled at the receiving host. Fragmentation can happen at either at the originating host or at an intermediate router.

IP fragmentation can cause excessive retransmissions when fragments encounter packet loss and reliable protocols such as TCP must retransmit all of the fragments in order to recover from the loss of a single fragment.[4] Thus, senders typically use two approaches to decide the size of IP datagrams to send over the network. The first is for the sending host to send an IP datagram of size equal to the MTU of the first hop of the source destination pair. The second is to run the path MTU discovery algorithm,[5] described in RFC 1191, to determine the path MTU between two IP hosts, so that IP fragmentation can be avoided.

1.03 - Given a fragment, identify what information is needed for reassembly

Understanding IP Fragmentation

How are the packets reassembled?

Note that with IP fragmentation, packets are not reassembled until they reach the final destination. It is reassembled at the IP layer at the receiving end. This is make fragmentation and reassembly transparent to the protocol layer (TCP and UDP). If one of the packets is lost, the whole packets need to be transmitted again. Packets are reassembled at the receiving host by associating each fragment with an identical fragment identification number, or frag id for short. The frag ID is actually a copy of the ID field (IP identification number) in the IP header. Besides that, each fragment must carry its “position” or “offset” in the original unfragmented packet. Thus the first fragment will have an offset of 0, since its seat is at the front row and counting starts from 0. Each fragment must also tell the length of data that it carries. This is like the compartments in a train. And finally, each fragment must flag the MF (more fragments) bit if it is not the last fragment.

Fragmenting a Packet

Here is a hypothetical example. Suppose that we want to send a 110 bytes ICMP packet on a network with MTU of 40 (well that's damn small, but this is for illustration purposes). This is a diagram of the original packet:

IP	ICMP	Data
Header		Header
20	8	82 (bytes)

The packet will be fragmented as shown below.

```
+----------------+----------+-----------+

Packet 1 | IP header (20) | ICMP (8) | Data (12) | ID=88, Len=20, Off=0, MF=1

+----------------+----------+-----------+

Packet 2 | IP header (20) | Data (20) | ID=88, Len=20, Off=20, MF=1

+---------------------------------------+

Packet 3 | IP header (20) | Data (20) | ID=88, Len=20, Off=40, MF=1

+---------------------------------------+

Packet 4 | IP header (20) | Data (20) | ID=88, Len=20, Off=60, MF=1

+---------------------------+----------+

Packet 5 | IP header (20) | Data (10) | ID=88, Len=10, Off=80, MF=0

+---------------------------+
```

ID - IP identification number
Len - Data Length (data length does not include IP header)
Off - Offset
MF - More Fragment

Notice that the second packet and subsequent packets contains IP header that is copied from the original packet. There are no ICMP headers, except in the first packet. In a nutshell, the 110 ICMP packet is broke into 5 packet, with total lengths of 40, 40, 40, 40 and 30 bytes each. The ICMP data is broken into lengths of 12, 20, 20, 20, and 10 bytes each.

1.03 - Explain the purpose of TTL functionality

Time to Live

TTL

TTL may be implemented as a counter or timestamp attached to or embedded in the data. Once the prescribed event count or timespan has elapsed, data is discarded. In computer networking, TTL prevents a data packet from circulating indefinitely. In computing applications, TTL is used to improve performance of caching or to improve privacy.

Under the Internet Protocol, TTL is an 8-bit field. In the IPv4 header, TTL is the 9th octet of 20. In the IPv6 header, it is the 8th octet of 40. The maximum TTL value is 255, the maximum value of a single octet. A recommended initial value is 64.

The time-to-live value can be thought of as an upper bound on the time that an IP datagram can exist in an Internet system. The TTL field is set by the sender of the datagram, and reduced by every router on the route to its destination. If the TTL field reaches zero before the datagram arrives at its destination, then the datagram is discarded and an ICMP error datagram (11 - Time Exceeded) is sent back to the sender. The purpose of the TTL field is to avoid a situation in which an undeliverable datagram keeps circulating on an Internet system, and such a system eventually becoming swamped by such “immortals”.

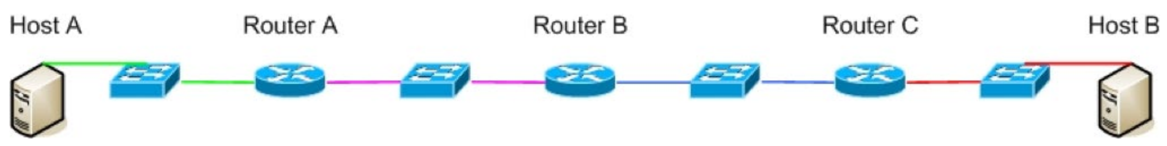
In theory, under IPv4, time to live is measured in seconds, although every host that passes the datagram must reduce the TTL by at least one unit. In practice, the TTL field is reduced by one with every hop. To reflect this practice, the field is renamed hop limit in IPv6.

1.03 - Given a packet traversing a topology, document the source/ destination IP address/MAC address changes at each hop

Packet Traversing a Topology

If Host A wants to talk to host B on the network and there are multiple routed networks between the two devices, can you describe the changes to the packet will look like as it passes through each network device?

Here is an example network and we will discuss the process of the packet traversing the network below.



So as Host A attempts to communicate to Host B (via an application like a browser) this will either be a connection based on a DNS name or an IP address. If it is DNS name, it will resolve the host name to an IP address or if not, it will use the known IP address in the browser path. The operating system will look to see if Host B is on it own local IP subnet. If it were it would look in it's ARP cache to see if it has an entry for Host B's IP address. Since Host B is not on its local IP subnet it will send the traffic to it's default gateway. The packet will not be destined for the IP of the default gateway (in this case Router A's IP on the green subnet) but it will look in ARP cache for the MAC address of the gateway and the packet will look like this as it leaves Host A:

Src MAC = Host A
Dest MAC = DGW Router A
Src IP = Host A
Dest IP = Host B

The default gateway will receive the packet and will process it since it is destine for its MAC Address. Router A will send the packet to the next hop router within the network (based on static routes or routes via routing protocols) and the packet will look like the following:

Src MAC = Router A
Dest MAC = Router B
Src IP = Host A
Dest IP = Host B

Router B will receive the packet and will process it since it is destine for its MAC Address. Router B will send the packet to the next hop router within the network and the packet will look like the following:

Src MAC = Router B
Dest MAC = Router C
Src IP = Host A
Dest IP = Host B

Router C will receive the packet and will process it since it is destine for its MAC Address. Router C will See that the destination IP is on a locally attached subnet and will check its ARP Cache for the MAC address of Host B's IP address. If it has a known MAC Address it will use it and if it does not it will ARP for the IP to add the entry to it's ARP table. Once it knows the MAC address it will send on the packet that will look like the following:

Src MAC = Router C
Dest MAC = Host B
Src IP = Host A
Dest IP = Host B

1.03 - IP version 6 (not in depth on exam)

Understand IPv6 Addresses

IPv6

Increasing the IP address pool was one of the major forces behind developing IPv6. It uses a 128-bit address, meaning that we have a maximum of 2128 addresses available, or 340,282,366,920,938,463,463,374,607,431,768,211,456, or enough to give multiple IP addresses to every grain of sand on the planet. So our friendly old 32-bit IPv4 dotted-quads don't do the job anymore; these newfangled IPs require eight 16-bit hexadecimal colon-delimited blocks. So not only are they longer, they use numbers and letters. At first glance, those huge IPv6 addresses look like impenetrable secret code:

```
2001:0db8:3c4d:0015:0000:0000:abcd:ef12
```

We'll dissect this in a moment and learn that's it not such a scary thing, but first let's look at the different types of IPv6 addressing.

Under IPv4 we have the old familiar unicast, broadcast and multicast addresses. In IPv6 we have unicast, multicast and anycast. With IPv6 the broadcast addresses are not used anymore, because they are replaced with multicast addressing.

IPv6 Unicast

This is similar to the unicast address in IPv4 – a single address identifying a single interface. There are four types of unicast addresses:

- Global unicast addresses, which are conventional, publicly routable address, just like conventional IPv4 publicly routable addresses.
- Link-local addresses are akin to the private, non-routable addresses in IPv4 (`10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16`). They are not meant to be routed, but confined to a single network segment. Link-local addresses mean you can easily throw together a temporary LAN, such as for conferences or meetings, or set up a permanent small LAN the easy way.
- Unique local addresses are also meant for private addressing, with the addition of being unique, so that joining two subnets does not cause address collisions.
- Special addresses are loopback addresses, IPv4-address mapped spaces, and 6-to-4 addresses for crossing from an IPv4 network to an IPv6 network.

If you read about site-local IPv6 addresses, which are related to link-local, these have been deprecated, so you don't need to bother with them.

Multicast

Multicast in IPv6 is similar to the old IPv4 broadcast address a packet sent to a multicast address is delivered to every interface in a group. The IPv6 difference is it's targeted instead of annoying every single host on the segment with broadcast blather, only hosts who are members of the multicast group receive the multicast packets. IPv6 multicast is routable, and routers will not forward multicast packets unless there are members of the multicast groups to forward the packets to. Anyone who has ever suffered from broadcast storms will appreciate this mightily.

Anycast

An anycast address is a single address assigned to multiple nodes. A packet sent to an anycast address is then delivered to the first available node. This is a slick way to provide both load-balancing and automatic failover. The idea of anycast has been = for a long time; it was proposed for inclusion in IPv4 but it never happened.

Several of the DNS root servers use a router-based anycast implementation, which is really a shared unicast addressing scheme. (While there are only thirteen authoritative root server names, the total number of actual servers is considerably larger, and they are spread all over the globe.) The same IP address is assigned to multiple interfaces, and then multiple routing tables entries are needed to move everything along.

IPv6 anycast addresses contain fields that identify them as anycast, so all you need to do is configure your network interfaces appropriately. The IPv6 protocol itself takes care of getting the packets to their final destinations. It's a lot simpler to administer than shared unicast addressing.

Address Dissection

Let's take another look at our example IPv6 address:

`2001:0db8:3c4d:0015:0000:0000:abcd:ef12`
______________|____|__________________
global prefix subnet Interface ID

The prefix identifies it as a global unicast address. It has three parts: the network identifier, the subnet, and the interface identifier.

The global routing prefix comes from a pool assigned to you, either by direct assignment from a Regional Internet Registry like APNIC, ARIN, or RIPE NCC, or more likely from your Internet service provider. The local network administrator controls the subnet and interface IDs.

You'll probably be running mixed IPv6/IPv4 networks for some time. IPv6 addresses must have 128 bits. IPv4 addresses are therefore represented like this:

```
0000:0000:0000:0000:0000:0000:192.168.1.25
```

Eight blocks of 16 bits each are required in an IPv6 address. The IPv4 address occupies 32 bits, so that is why there are only seven colon-delimited blocks.

The localhost address is `0000:0000:0000:0000:0000:0000:0000:0001`.

Naturally we want shortcuts, because these are long and all those zeroes are just dumb-looking. Leading zeroes can be omitted, and contiguous blocks of zeroes can be omitted entirely, so we end up with these:

```
2001:0db8:3c4d:0015:0:0:abcd:ef12

2001:0db8:3c4d:0015::abcd:ef12

::192.168.1.25

::1
```

Objective - 1.04 Explain the features and functionality of protocols and technologies specific to the transport layer

1.04 - Compare/Contrast purpose and functionality of MTU and MSS

MTU

MTU

The MTU is the maximum size of a single data unit (e.g., a frame) of digital communications. MTU sizes are inherent properties of physical network interfaces, normally measured in bytes. The MTU for Ethernet, for instance, is 1500 bytes. Some types of networks (like Token Ring) have larger MTUs, and some types have smaller MTUs, but the values are fixed for each physical technology.

Higher-level network protocols like TCP/IP can be configured with a maximum packet size, a parameter independent of the physical layer MTU over which TCP/IP runs. Unfortunately, many network devices use the terms interchangeably. On both home broadband routers and Xbox Live enabled game consoles, for example, the parameter called MTU is in fact the maximum TCP packet size and not the physical MTU.

In Microsoft Windows, the maximum packet size for protocols like TCP can be set in the Registry. If this value is set too low, streams of network traffic will be broken up into a relatively large number of small packets that adversely affects performance. Xbox Live, for example, requires the value of MTU (packet size) by at least 1365 bytes. If the maximum TCP packet size is set too high, it will exceed the network's physical MTU and also degrade performance by requiring that each packet be subdivided into smaller ones (a process known as fragmentation). Microsoft Windows computers default to a maximum packet size of 1500 bytes for broadband connections and 576 bytes for dialup connections.

Performance problems may also occur if the TCP "MTU" setting on the home broadband router differs from the setting on individual devices connected to it.

TCP Maximum Segment Size (MSS)

MSS

During session connection establishment, two peers, or hosts, engage in negotiations to determine the IP segment size of packets that they will exchange during their communication. The segment size is based on the MSS option (maximum segment size) value set in the TCP SYN (synchronize) packets that the peers exchange during session negotiation. The MSS field value to be used is largely determined by the maximum transmission unit (MTU) of the interfaces that the peers are directly connected to.

About TCP and MSS

The TCP protocol is designed to limit the size of segments of data to a maximum of number of bytes. The purpose for this is to constrain the need to fragment segments of data for transmission at the IP level. The TCP MSS specifies the maximum number of bytes that a TCP packet's data field, or segment, can contain. It refers to the maximum amount of TCP data in a single IP datagram that the local system can accept and reassemble.

A TCP packet includes data for headers as well as data contained in the segment. If the MSS value is set too low, the result is inefficient use of bandwidth; more packets are required to transmit the data. An MSS value that is set too high could result in an IP datagram that is too large to send and that must be fragmented.

Typically a host bases its MSS value on its outgoing interface's maximum transmission unit (MTU) size. The MTU is the maximum frame size along the path between peers. A packet is fragmented when it exceeds the MTU size. Because of variation of the MTU size of the interfaces of hosts in the path taken by TCP packets between two peers, some packets that are within the negotiated MSS size of the two peers might be fragmented but instead are dropped and an ICMP error message is sent to the source host of the packet.

To diminish the likelihood of fragmentation and to protect against packet loss, you can decrease the TCP MSS.

1.04 - Explain the purpose and functionality of TCP

TCP (Transmission Control Protocol)

TCP

TCP is a subset of the Internet protocol suite, which is often called TCP/IP, although the acronym TCP/IP refers to only two of the many protocols in the Internet protocol suite. Still, most people refer to the Internet protocols as TCP/IP and that style is retained here.

TCP is a connection-oriented protocol that provides the flow controls and reliable data delivery services listed next. These services run in the host computers at either end of a connection, not in the network itself. Therefore, TCP is a protocol for managing end-to-end connections. Since end-to-end connections may exist across a series of point-to-point connections, they are often called virtual circuits.

Quick terminology

Connections	Two computers set up a connection to exchange data. The systems synchronize with one another to manage packet flows and adapt to congestion in the network.
Full-duplex operation	A TCP connection is a pair of virtual circuits (one in each direction). Only the two end systems can use the connection.
Error checking	A checksum technique is used to verify that packets are not corrupted.
Sequencing	Packets are numbered so that the destination can reorder packets and determine if a packet is missing.
Acknowledgements	Upon receipt of one or more packets, the receiver returns an acknowledgement (called an "ACK") to the sender indicating that it received the packets. If packets are not ACKed, the sender may retransmit the packets (or terminate the connection if it thinks the receiver has crashed).
Flow control	If the sender is overflowing the receiver by transmitting too quickly, the receiver drops packets. Failed ACKs alert the sender to slow down or stop sending.
Packet recovery services	The receiver can request retransmission of a packet. Also, if packet receipt is not ACKed, the sender will resend the packets.

Reliable data delivery services are critical for applications such as file transfers, database services, transaction processing, and other mission-critical applications in which every packet must be delivered-guaranteed.

While TCP provides these reliable services, it depends on IP to delivery packets. IP is often referred to as an unreliable or best effort service. While it seems odd to build a network that is unreliable, the original Internet architects wanted to remove as many services from the network itself to support fast packet delivery rather than reliability. Routers do not keep track of packets or do anything to ensure delivery. They just forward packets.

The assumption was that end systems would be relatively smart devices with memory and processors. The end devices could handle all the reliability functions rather than the network. This was actually a radical approach at the time, but the implications have been profound. It meant that end systems would become the focus of application development for the Internet, not the network.

In contrast, the telephone network implements an architecture in which end devices (phones) are dumb and the network is supposedly "smart." The only problem with this model is that you can't run applications on your phone that takes advantage of the network. In fact, you are totally dependent on the phone company to deploy new applications (call waiting and caller ID are examples). Compared to the Internet, the phone system is a dinosaur. Consider that the user interface for the Web is a full-color graphical browser, while the interface for the telephone network is a 12-key pad!

While end-systems provide TCP's reliability functions, not all applications need them. For example, there is no need to recover lost packets in a live video stream. By the time they are recovered, the viewer has already seen the barely visible glitch caused by the missing packet. These applications just need speed. So UDP was created to provide an application interface to the network for real-time applications that don't need TCP's extra services. UDP provides a very simple port connection between applications and IP.

TCP Three-way handshake

A TCP Three-way handshake is a method of initializing a Transmission Control Protocol (TCP) session between two hosts on a TCP/IP network. The handshake establishes a logical connection between the hosts by synchronizing the sending and receiving of packets and communicating TCP parameters between the hosts.

How the TCP Three-way Handshake works

All TCP communication is connection oriented. A TCP session must be established before the hosts in the connection exchange data. Packets that are transferred between hosts are accounted for by assigning a sequence number to each packet. An ACK, or acknowledgment, is sent after every packet is received. If no ACK is received for a packet, the packet is re-sent. The three-way handshake ensures that the initial request is acknowledged, that the data is sent, and that the data is acknowledged.

These are the three stages of a TCP three-way handshake:

- The initiating host sends a TCP packet requesting a new session. This packet contains the initiating host's sequence number for the connection. The packet includes information such as a set SYN (synchronization) flag and data about the size of the window buffer on the initiating host.

- The target host sends a TCP packet with its own sequence number and an ACK of the initiating host's sequence number.
- The initiating host sends an ACK containing the target sequence number that it received.

  Note: A similar three-way process is used to terminate a TCP session between two hosts. Using the same type of handshake to end the connection ensures that the hosts have completed their transactions and that all data is accounted for.

1.04 - Explain the purpose and functionality of UDP

What is UDP and How Does It Work?

UDP

UDP stands for User Datagram Protocol. UDP provides an unreliable packet delivery system built on top of the IP protocol. As with IP, each packet is individual and is handled separately. Because of this, the amount of data that can be sent in a UDP packet is limited to the amount that can be contained in a single IP packet. Thus, a UDP packet can contain at most 65507 bytes (this is the 65535-byte IP packet size minus the minimum IP header of 20 bytes and minus the 8-byte UDP header).

UDP packets can arrive out of order or not at all. No packet has any knowledge of the preceding or following packet. The recipient does not acknowledge packets, so the sender does not know that the transmission was successful. UDP has no provisions for flow control--packets can be received faster than they can be used. We call this type of communication connectionless because the packets have no relationship to each other and because there is no state maintained.

The destination IP address and port number are encapsulated in each UDP packet. These two numbers together uniquely identify the recipient and are used by the underlying operating system to deliver the packet to a specific process (application). Each UDP packet also contains the sender's IP address and port number.

One way to think of UDP is by analogy to communications via a letter. You write the letter (this is the data you are sending); put the letter inside an envelope (the UDP packet); address the envelope (using an IP address and a port number); put your return address on the envelope (your local IP address and port number); and then you send the letter.

Like a real letter, you have no way of knowing whether a UDP packet was received. If you send a second letter one day after the first, the second one may be received before the first. Or, the second one may never be received.

1.04 - Explain the purpose and functionality of ports in general

How Internet Infrastructure Works

Protocol Ports

In TCP/IP and UDP networks, a port is an endpoint to a logical connection and the way a client program specifies a specific server program on a computer in a network. There are 65535 available ports per IP address and many of these ports are reserved as well known application ports.

A server makes its services available using numbered ports; one for each service that is available on the server. For example, if a server machine is running a Web server and a file transfer protocol (FTP) server, the Web server would typically be available on port 80, and the FTP server would be available on port 21. Clients connect to a service at a specific IP address and on a specific port number.

Once a client has connected to a service on a particular port, it accesses the service using a specific protocol. Protocols are often text and simply describe how the client and server will have their conversation. Every Web server on the Internet conforms to the hypertext transfer protocol (HTTP).

1.04 - Explain how retransmissions occur

TCP Development

TCP Timeout and Retransmission

The Transmission Control Protocol provides a communication service at an intermediate level between an application program and the Internet Protocol. It provides host-to-host connectivity at the Transport Layer of the Internet model. An application does not need to know the particular mechanisms for sending data via a link to another host, such as the required packet fragmentation on the transmission medium. At the transport layer, the protocol handles all handshaking and transmission details and presents an abstraction of the network connection to the application.

At the lower levels of the protocol stack, due to network congestion, traffic load balancing, or other unpredictable network behavior, IP packets may be lost, duplicated, or delivered out of order. TCP detects these problems, requests retransmission of lost data, rearranges out-of-order data, and even helps minimize network congestion to reduce the occurrence of the other problems. If the data still remains undelivered, its source is notified of this failure. Once the TCP receiver has reassembled the sequence of octets originally transmitted, it passes them to the receiving application. Thus, TCP abstracts the application's communication from the underlying networking details.

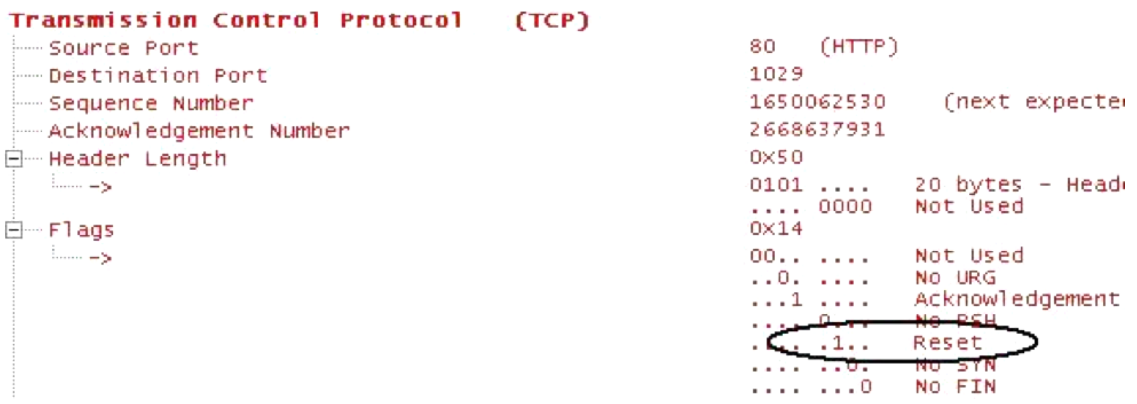
TCP is a reliable stream delivery service that guarantees that all bytes received will be identical with bytes sent and in the correct order. Since packet transfer over many networks is not reliable, a technique known as positive acknowledgment with retransmission is used to guarantee reliability of packet transfers. This fundamental technique requires the receiver to respond with an acknowledgment message as it receives the data. The sender keeps a record of each packet it sends. The sender also maintains a timer from when the packet was sent, and retransmits a packet if the timer expires before the message has been acknowledged. The timer is needed in case a packet gets lost or corrupted.

While IP handles actual delivery of the data, TCP keeps track of the individual units of data transmission, called segments, which a message is divided into for efficient routing through the network. For example, when an HTML file is sent from a web server, the TCP software layer of that server divides the sequence of octets of the file into segments and forwards them individually to the IP software layer (Internet Layer). The Internet Layer encapsulates each TCP segment into an IP packet by adding a header that includes (among other data) the destination IP address. When the client program on the destination computer receives them, the TCP layer (Transport Layer) reassembles the individual segments and ensures they are correctly ordered and error free as it streams them to an application.

1.04 - Explain the purpose and process of a reset

What are TCP RST Packets?

According to RFC 793, which specifies an Option in the Flags portion of the TCP header called Reset (or RST). The Reset bit is designed to allow a station to abort the TCP connection with another station. This can happen for a number of reasons.


```
Transmission Control Protocol    (TCP)
  Source Port                          80   (HTTP)
  Destination Port                     1029
  Sequence Number                      1650062530    (next experte
  Acknowledgement Number               2668637931
  Header Length                        0x50
    ->                                 0101 ....   20 bytes - Head
                                       .... 0000   Not Used
  Flags                                0x14
    ->                                 00.. ....   Not Used
                                       ..0. ....   No URG
                                       ...1 ....   Acknowledgement
                                       .... 0...   No PSH
                                       .... .1..   Reset
                                       .... ..0.   No SYN
                                       .... ...0   No FIN
```

If a station involved in a TCP session notices that it is not receiving acknowledgements for anything it sends, the connection is now unsynchronized, and the station should send a reset. This is a half-open connection where only one side is involved in the TCP session. This cannot work by definition of the protocol.

RST packets are a sign that the TCP connections are half open. One station or the other stopped sending information or ACKs for some reason. There are acceptable times for RST packets, however, if there are a large number of RST packets in a conversation, this is definitely something to troubleshoot. Which side is sending the RST? What is causing it to send the RST? Does this happen right away in the TCP setup, or is it later in the session? If later, is there any reason that the station would abort the session in the middle of the data transfer?

1.04 - Describe various TCP options

TCP Development

TCP Options

The length of this field is determined by the data offset field. Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable). The Option-Kind field indicates the type of option, and is the only field that is not optional. Depending on what kind of option we are dealing with, the next two fields may be set: the Option-Length field indicates the total length of the option, and the Option-Data field contains the value of the option, if applicable. For example, an Option-Kind byte of 0x01 indicates that this is a No-Op option used only for padding, and does not have an Option-Length or Option-Data byte following it. An Option-Kind byte of 0 is the End Of Options option, and is also only one byte. An Option-Kind byte of 0x02 indicates that this is the Maximum Segment Size option, and will be followed by a byte specifying the length of the MSS field (should be 0x04). Note that this length is the total length of the given options field, including Option-Kind and Option-Length bytes. So while the MSS value is typically expressed in two bytes, the length of the field will be 4 bytes (+2 bytes of kind and length). In short, an MSS option field with a value of 0x05B4 will show up as (0x02 0x04 0x05B4) in the TCP options section.

Some options may only be sent when SYN is set. Option-Kind and standard lengths given as (Option-Kind, Option-Length).

- 0 (8 bits) – End of options list
- 1 (8 bits) – No operation (NOP, Padding). This may be used to align option fields on 32-bit boundaries for better performance.
- 2,4,SS (32 bits) – Maximum segment size (see maximum segment size)
- 3,3,S (24 bits) – Window scale (see window scaling for details)

- 4,2 (16 bits) – Selective Acknowledgement permitted. (See selective acknowledgments for details)
- 5,N,BBBB,EEEE,... (variable bits, N is either 10, 18, 26, or 34)- Selective ACKnowledgement (SACK). These first two bytes are followed by a list of 1–4 blocks being selectively acknowledged, specified as 32-bit begin/end pointers.
- 8,10,TTTT,EEEE (80 bits)- Timestamp and echo of previous timestamp (see TCP timestamps for details)

(The remaining options are historical, obsolete, experimental, not yet standardized, or unassigned)

1.04 - Describe a TCP checksum error

TCP/IP Guide

TCP Checksum

The Transmission Control Protocol is designed to provide reliable data transfer between a pair of devices on an IP internetwork. Much of the effort required to ensure reliable delivery of data segments is of necessity focused on the problem of ensuring that data is not lost in transit. But there's another important critical impediment to the safe transmission of data: the risk of errors being introduced into a TCP segment during its travel across the internetwork.

Detecting Transmission Errors Using Checksums

If the data gets where it needs to go but is corrupted and we do not detect the corruption, this is in some ways worse than it never showing up at all. To provide basic protection against errors in transmission, TCP includes a 16-bit Checksum field in its header. The idea behind a checksum is very straightforward: take a string of data bytes and add them all together. Then send this sum with the data stream and have the receiver check the sum. In TCP, a special algorithm is used to calculate this checksum by the device sending the segment; the same algorithm is then employed by the recipient to check the data it received and ensure that there were no errors.

The checksum calculation used by TCP is a bit different than a regular checksum algorithm. A conventional checksum is performed over all the bytes that the checksum is intended to protect, and can detect most bit errors in any of those fields. The designers of TCP wanted this bit error protection, but also desired to protect against other type of problems.

1.04 - Describe how TCP addresses error correction

TCP Checksum for IPv4

TCP Error Correction

Sequence numbers allow receivers to discard duplicate packets and properly sequence reordered packets. Acknowledgments allow senders to determine when to retransmit lost packets.

To assure correctness a checksum field is included. When TCP runs over IPv4, the method used to compute the checksum is defined in RFC 793. The TCP checksum is a weak check by modern standards. Data Link Layers with high bit error rates may require additional link error correction/detection capabilities. The weak checksum is partially compensated for by the common use of a CRC or better integrity check at layer 2, below both TCP and IP, such as is used in PPP or the Ethernet frame. However, this does not mean that the 16-bit TCP checksum is redundant: remarkably, introduction of errors in packets between CRC-protected hops is common, but the end-to-end 16-bit TCP checksum catches most of these simple errors. This is the end-to-end principle at work.

1.04 - Describe how the flow control process occurs

TCP Flow Control

Flow Control

TCP uses an end-to-end flow control protocol to avoid having the sender send data too fast for the TCP receiver to receive and process it reliably. Having a mechanism for flow control is essential in an environment where machines of diverse network speeds communicate. For example, if a PC sends data to a smartphone that is slowly processing received data, the smartphone must regulate the data flow so as not to be overwhelmed.

TCP uses a sliding window flow control protocol. In each TCP segment, the receiver specifies in the receive window field the amount of additionally received data (in bytes) that it is willing to buffer for the connection. The sending host can send only up to that amount of data before it must wait for an acknowledgment and window update from the receiving host.

TCP sequence numbers and receive windows behave very much like a clock. The receive window shifts each time the receiver receives and acknowledges a new segment of data. Once it runs out of sequence numbers, the sequence number loops back to 0.

When a receiver advertises a window size of 0, the sender stops sending data and starts the persist timer. The persist timer is used to protect TCP from a deadlock situation that could arise if a subsequent window size update from the receiver is lost, and the sender cannot send more data until receiving a new window size update from the receiver. When the persist-timer expires, the TCP sender attempts recovery by sending a small packet so that the receiver responds by sending another acknowledgement containing the new window size.

If a receiver is processing incoming data in small increments, it may repeatedly advertise a small receive window. This is referred to as the silly window syndrome, since it is inefficient to send only a few bytes of data in a TCP segment, given the relatively large overhead of the TCP header.

Congestion control

The final main aspect of TCP is congestion control. TCP uses a number of mechanisms to achieve high performance and avoid congestion collapse, where network performance can fall by several orders of magnitude. These mechanisms control the rate of data entering the network, keeping the data flow below a rate that would trigger collapse. They also yield an approximately max-min fair allocation between flows.

Senders infer network conditions between the TCP sender and receiver use acknowledgments for data sent, or lack of acknowledgments. Coupled with timers, TCP senders and receivers can alter the behavior of the flow of data. This is more generally referred to as congestion control and/or network congestion avoidance.

Modern implementations of TCP contain four intertwined algorithms: Slow-start, congestion avoidance, fast retransmit, and fast recovery (RFC 5681).

In addition, senders employ a retransmission timeout (RTO) that is based on the estimated round-trip time (or RTT) between the sender and receiver, as well as the variance in this round trip time. The behavior of this timer is specified in RFC 6298. There are subtleties in the estimation of RTT. For example, senders must be careful when calculating RTT samples for retransmitted packets; typically they use Karn's Algorithm or TCP timestamps (see RFC 1323). These individual RTT samples are then averaged over time to create a Smoothed Round Trip Time (SRTT) using Jacobson's algorithm. This SRTT value is what is finally used as the round-trip time estimate.

Enhancing TCP to reliably handle loss, minimize errors, manage congestion and go fast in very high-speed environments are ongoing areas of research and standards development. As a result, there are a number of TCP congestion avoidance algorithm variations.

Delayed Binding

Delayed binding, also called TCP connection splicing, is the postponement of the connection between the client and the server in order to obtain sufficient information to make a routing decision. Some application switches and routers delay binding the client session to the server until the proper handshakes are complete so as to prevent Denial of Service attacks.

Objective - 1.05 Explain the features and functionality of protocols and technologies specific to the application layer

1.05 - Explain the purpose and functionality of HTTP

Hypertext Transfer Protocol

HTTP Protocol

HTTP functions as a request-response protocol in the client-server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

A web browser is an example of a user agent (UA). Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

HTTP is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of upstream servers to improve response time. Web browsers cache previously accessed web resources and reuses them when possible to reduce network traffic. HTTP proxy servers at private network boundaries can facilitate communication for clients without a globally routable address, by relaying messages with external servers.

HTTP is an application layer protocol designed within the framework of the Internet Protocol Suite. Its definition presumes an underlying and reliable transport layer protocol, and Transmission Control Protocol (TCP) is commonly used. However HTTP can use unreliable protocols such as the User Datagram Protocol (UDP), for example in Simple Service Discovery Protocol (SSDP).

HTTP resources are identified and located on the network by Uniform Resource Identifiers (URIs)—or, more specifically, Uniform Resource Locators (URLs)—using the http or https URI schemes. URIs and hyperlinks in Hypertext Markup Language (HTML) documents form webs of inter-linked hypertext documents.

1.05 - Differentiate between HTTP versions

HTTP History

HTTP versions

The first documented version of HTTP was HTTP V0.9 (1991). Dave Raggett led the HTTP Working Group (HTTP WG) in 1995 and wanted to expand the protocol with extended operations, extended negotiation, richer meta-information, tied with a security protocol which became more efficient by adding additional methods and header fields. RFC 1945 officially introduced and recognized HTTP V1.0 in 1996.

The HTTP WG planned to publish new standards in December 1995 and the support for pre-standard HTTP/1.1 based on the then developing RFC 2068 (called HTTP-NG) was rapidly adopted by the major browser developers in early 1996. By March 1996, pre-standard HTTP/1.1 was supported in Arena, Netscape 2.0, Netscape Navigator Gold 2.01, Mosaic 2.7, Lynx 2.5, and in Internet Explorer 2.0. End-user adoption of the new browsers was rapid. In March 1996, one web hosting company reported that over 40% of browsers in use on the Internet were HTTP 1.1 compliant. That same web hosting company reported that by June 1996, 65% of all browsers accessing their servers were HTTP/1.1 compliant. The HTTP/1.1 standard as defined in RFC 2068 was officially released in January 1997. Improvements and updates to the HTTP/1.1 standard were released under RFC 2616 in June 1999.

Some of the major changes from version 1.0 to version 1.1 were based around request methods. HTTP defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server. The HTTP/1.0 specification defined the GET, POST and HEAD methods and the HTTP/1.1 specification added 5 new methods: OPTIONS, PUT, DELETE, TRACE and CONNECT.

1.05 - Interpret HTTP status codes

HTTP Made Really Easy

Structure of HTTP Transactions

Like most network protocols, HTTP uses the client-server model: An HTTP client opens a connection and sends a request message to an HTTP server; the server then returns a response message, usually containing the resource that was requested. After delivering the response, the server closes the connection (making HTTP a stateless protocol, i.e. not maintaining any connection information between transactions).

The formats of the request and response messages are similar, and English-oriented. Both kinds of messages consist of:

- an initial line,
- zero or more header lines,
- a blank line (i.e. a CRLF by itself), and
- an optional message body (e.g. a file, or query data, or query output).

Put another way, the format of an HTTP message is:

```
<initial line, different for request vs. response>
Header1: value1
Header2: value2
Header3: value3
 <optional message body goes here, like file contents or query data; it can
be many lines long, or even binary data $&*%@!^$@>
```

Initial lines and headers should end in CRLF, though you should gracefully handle lines ending in just LF. (More exactly, CR and LF here mean ASCII values 13 and 10, even though some platforms may use different characters.)

Initial Request Line

The initial line is different for the request than for the response. A request line has three parts, separated by spaces: a method name, the local path of the requested resource, and the version of HTTP being used. A typical request line is:

```
GET /path/to/file/index.html HTTP/1.0
```

Notes:

- GET is the most common HTTP method; it says "give me this resource". Other methods include POST and HEAD—more on those later. Method names are always uppercase.
- The path is the part of the URL after the host name, also called the request URI (a URI is like a URL, but more general).
- The HTTP version always takes the form "HTTP/x.x", uppercase.

Initial Response Line (Status Line)

The initial response line, called the status line, also has three parts separated by spaces: the HTTP version, a response status code that gives the result of the request, and an English reason phrase describing the status code. Typical status lines are:

```
HTTP/1.0 200 OK or HTTP/1.0 404 Not Found
```

Notes:

- The HTTP version is in the same format as in the request line, "HTTP/x.x".
- The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.
- The status code is a three-digit integer, and the first digit identifies the general category of response:
    - 1xx indicates an informational message only
    - 2xx indicates success of some kind
    - 3xx redirects the client to another URL
    - 4xx indicates an error on the client's part
    - 5xx indicates an error on the server's part

The most common status codes are:

200 OK	The request succeeded, and the resulting resource (e.g. file or script output) is returned in the message body.
301	Moved Permanently
302	Moved Temporarily
303 See Other (HTTP 1.1 only)	The resource has moved to another URL (given by the Location: response header), and should be automatically retrieved by the client. This is often used by a CGI script to redirect the browser to an existing file.
403 Forbidden	The request was a valid request, but the server is refusing to respond to it.
404 Not Found	The requested resource doesn't exist.
500 Internal Server Error	An unexpected server error. The most common cause is a server-side script that has bad syntax, fails, or otherwise can't run correctly.
503 Service Unavailable	The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state.

Header Lines

Header lines provide information about the request or response, or about the object sent in the message body. The header lines are in the usual text header format, which is: one line per header, of the form "Header-Name: value", ending with CRLF. It's the same format used for email and news postings.

1.05 - Determine an HTTP request method for a given use case

Other HTTP Methods, Like HEAD and POST

Other HTTP Methods, Like HEAD and POST

Besides GET, the two most commonly used methods are HEAD and POST.

The HEAD Method

A HEAD request is just like a GET request, except it asks the server to return the response headers only, and not the actual resource (i.e. no message body). This is useful to check characteristics of a resource without actually downloading it, thus saving bandwidth. Use HEAD when you don't actually need a file's contents.

The response to a HEAD request must never contain a message body, just the status line and headers.

The POST Method

A POST request is used to send data to the server to be processed in some way, like by a CGI script. A POST request is different from a GET request in the following ways:

- There's a block of data sent with the request, in the message body. There are usually extra headers to describe this message body, like Content-Type: and Content-Length:.
- The request URI is not a resource to retrieve; it's usually a program to handle the data you're sending.
- The HTTP response is normally program output, not a static file.

The most common use of POST, by far, is to submit HTML form data to CGI scripts. In this case, the Content-Type: header is usually application/x-www-form-urlencoded, and the Content-Length: header gives the length of the URL-encoded form data (here's a note on URL-encoding). The CGI script receives the message body through STDIN, and decodes it. Here's a typical form submission, using POST:

```
POST /path/script.cgi HTTP/1.0

From: frog@jmarshall.com

User-Agent: HTTPTool/1.0
```

```
Content-Type: application/x-www-form-urlencoded

Content-Length: 32

home=Cosby&favorite+flavor=flies
```

You can use a POST request to send whatever data you want, not just form submissions. Just make sure the sender and the receiving program agree on the format.

The GET method can also be used to submit forms. The form data is URL-encoded and appended to the request URI.

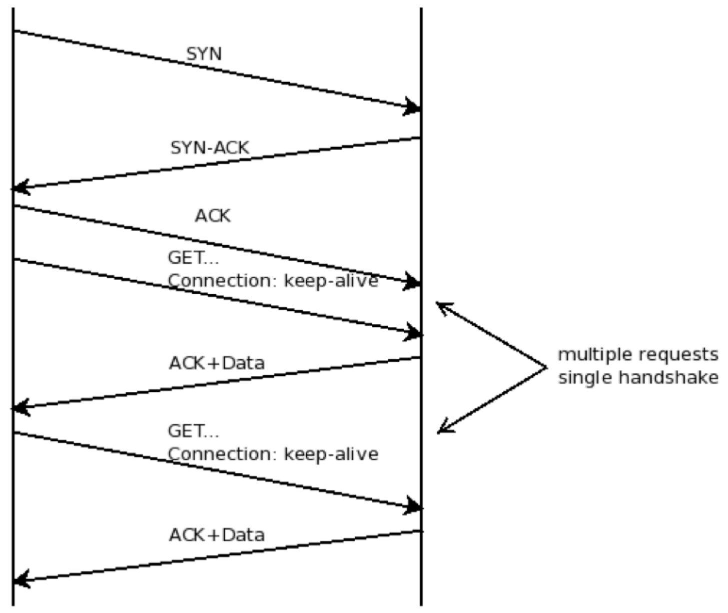
1.05 - Explain the purpose and functionality of HTTP Keep-alives, HTTP headers, DNS, SIP, FTP

HTTP Persistent Connection

HTTP Keep-alives

HTTP keep-alive, also called HTTP persistent connection, or HTTP connection reuse, is the idea of using a single TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new connection for every single request/response pair.

The Keep-Alive header field and the additional information it provides are optional and do not need to be present to indicate a persistent connection has been established.



HTTP Field Names

HTTP Headers

HTTP header fields are components of the header section of request and response messages in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction.

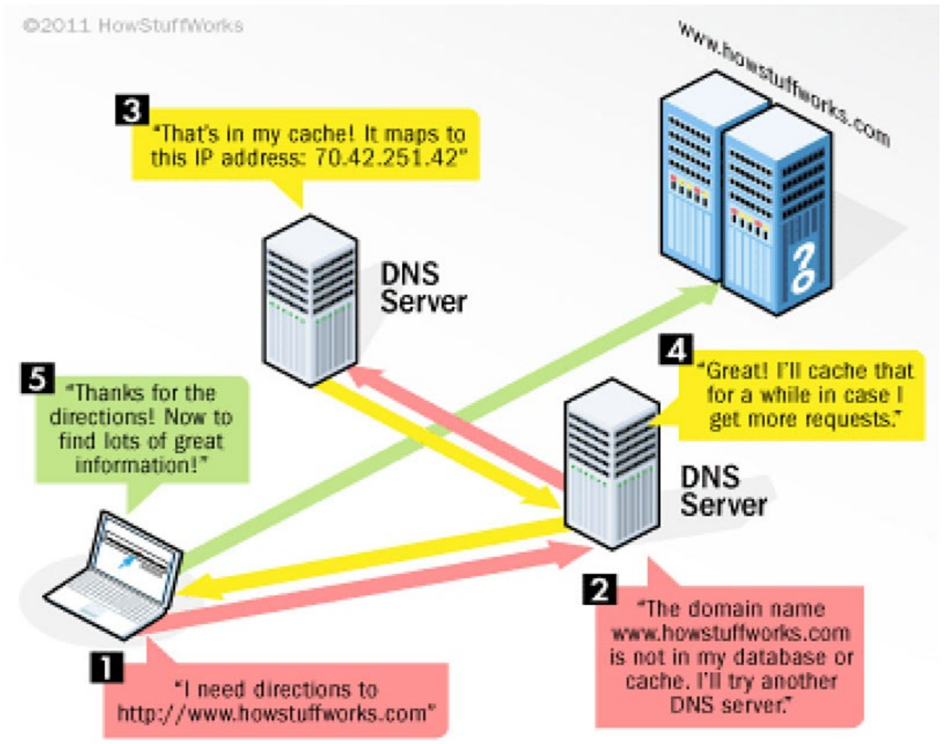
The header fields are transmitted after the request or response line, which is the first line of a message. Header fields are colon-separated name-value pairs in clear-text string format, terminated by a carriage return (CR) and line feed (LF) character sequence. The end of the header section is indicated by an empty field, resulting in the transmission of two consecutive CR-LF pairs. Historically, long lines could be folded into multiple lines; continuation lines are indicated by the presence of a space (SP) or horizontal tab (HT) as the first character on the next line. This folding is now deprecated.

How Domain Name Servers Work

Domain Name System (DNS)

If you've ever used the Internet, it's a good bet that you've used the Domain Name System, or DNS, even without realizing it. DNS is a protocol within the set of standards for how computers exchange data on the Internet and on many private networks, known as the TCP/IP protocol suite. Its basic job is to turn a user-friendly domain name like "howstuffworks.com" into an Internet Protocol (IP) address like 70.42.251.42 that computers use to identify each other on the network. It's like your computer's GPS for the Internet.

Computers and other network devices on the Internet use an IP address to route your request to the site you're trying to reach. This is similar to dialing a phone number to connect to the person you're trying to call. Thanks to DNS, though, you don't have to keep your own address book of IP addresses. Instead, you just connect through a domain name server, also called a DNS server or name server, which manages a massive database that maps domain names to IP addresses.


©2011 HowStuffWorks
www.howstuffworks.com
3 "That's in my cache! It maps to this IP address: 70.42.251.42"
DNS Server
5 "Thanks for the directions! Now to find lots of great information!"
4 "Great! I'll cache that for a while in case I get more requests."
DNS Server
2 "The domain name www.howstuffworks.com is not in my database or cache. I'll try another DNS server."
1 "I need directions to http://www.howstuffworks.com"

Whether you're accessing a Web site or sending e-mail, your computer uses a DNS server to look up the domain name you're trying to access. The proper term for this process is DNS name resolution, and you would say that the DNS server resolves the domain name to the IP address. For example, when you enter "http://www.howstuffworks.com" in your browser, part of the network connection includes resolving the domain name "howstuffworks.com" into an IP address, like 70.42.251.42, for 'HowStuffWorks' Web servers.

You can always bypass a DNS lookup by entering 70.42.251.42 directly in your browser (give it a try). However, you're probably more likely to remember "howstuffworks.com" when you want to return later. In addition, a Web site's IP address can change over time, and some sites associate multiple IP addresses with a single domain name.

Without DNS servers, the Internet would shut down very quickly. But how does your computer know what DNS server to use? Typically, when you connect to your home network, Internet service provider (ISP) or WiFi network, the modem or router that assigns your computer's network address also sends some important network configuration information to your computer or mobile device.

That configuration includes one or more DNS servers that the device should use when translating DNS names to IP address.

How Domain Name Servers Work

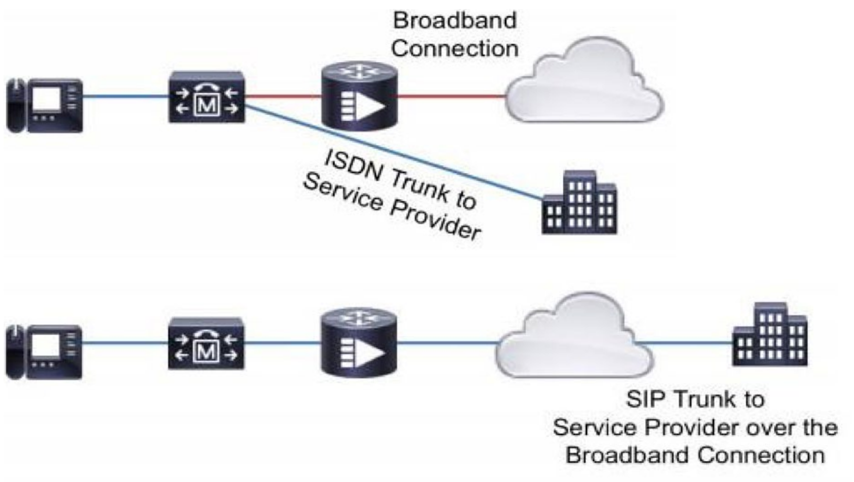
Session Initiation Protocol (SIP)

Session Initiation Protocol (SIP) is a communications protocol used for communicating between different devices on a company network, whether on the LAN, the WAN, or across the Internet. An example of this could be a simple two-way phone conversation, using voice over IP (VoIP) on the LAN or WAN or a SIP trunk across the Internet to a service provider. A SIP trunk provides a new way of connecting to a service provider for incoming and outgoing calls; it is a connection over the Internet instead of a traditional telephone connection such as ISDN.

SIP allows you to take full advantage of applications such as video conferencing, presence, and instant messaging. These applications, and others like them, when working together are known as unified communications. Unified communications opens up a new world of possibilities in how you interact with your customers and prospects, giving them a richer experience when dealing with you and your staff.

SIP provides businesses many benefits over older, proprietary telephony solutions, and best of all, it can save you money.

In the past, connections to the service provider (telephone company) were possible only using a dedicated telephone line, such as an ISDN connection.


Broadband Connection
ISDN Trunk to Service Provider
SIP Trunk to Service Provider over the Broadband Connection

File Transfer Protocol

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet. FTP is built on client-server architecture and uses separate control and data connections between the client and the server. FTP users may authenticate themselves using a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that hides (encrypts) the username and password, and encrypts the content, FTP is often secured with SSL/TLS ("FTPS"). SSH File Transfer Protocol ("SFTP") is sometimes also used instead, but is technologically different.

The first FTP client applications were command-line applications developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix, and Linux operating systems. Dozens of FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into hundreds of productivity applications, such as Web page editors.

1.05 - Differentiate between passive and active FTP

Active FTP vs. Passive FTP, a Definitive Explanation

Active FTP vs. Passive FTP

One of the most commonly seen questions when dealing with firewalls and other Internet connectivity issues is the difference between active and passive FTP and how best to support either or both of them. Hopefully the following text will help to clear up some of the confusion over how to support FTP in a firewalled environment.

The Basics

FTP is a TCP based service exclusively. There is no UDP component to FTP. FTP is an unusual service in that it utilizes two ports, a 'data' port and a 'command' port (also known as the control port). Traditionally these are port 21 for the command port and port 20 for the data port. The confusion begins however, when we find that depending on the mode, the data port is not always on port 20.

Active FTP

In active mode FTP the client connects from a random unprivileged port (N > 1023) to the FTP server's command port, port 21. Then, the client starts listening to port N+1 and sends the FTP command PORT N+1 to the FTP server. The server will then connect back to the client's specified data port from its local data port, which is port 20.

From the server-side firewall's standpoint, to support active mode FTP the following communication channels need to be opened:

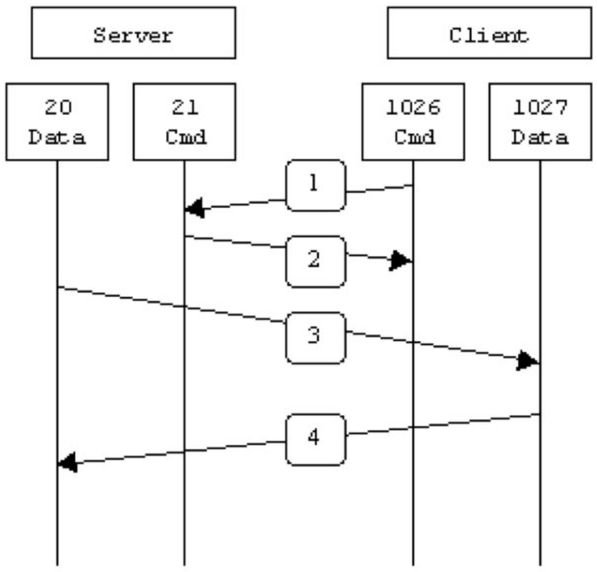
FTP server's port 21 from anywhere (Client initiates connection)

FTP server's port 21 to ports > 1023 (Server responds to client's control port)

FTP server's port 20 to ports > 1023 (Server initiates data connection to client's data port)

FTP server's port 20 from ports > 1023 (Client sends ACKs to server's data port)

When drawn out, the connection appears as follows:



- The client's command port contacts the server's command port and sends the command PORT 1027.
- The server then sends an ACK back to the client's command port
- The server initiates a connection on its local data port to the data port the client specified earlier.
- Finally, the client sends an ACK.

The main problem with active mode FTP actually falls on the client side. The FTP client doesn't make the actual connection to the data port of the server--it simply tells the server what port it is listening on and the server connects back to the specified port on the client. From the client-side firewall this appears to be an outside system initiating a connection to an internal client--something that is usually blocked.

Passive FTP

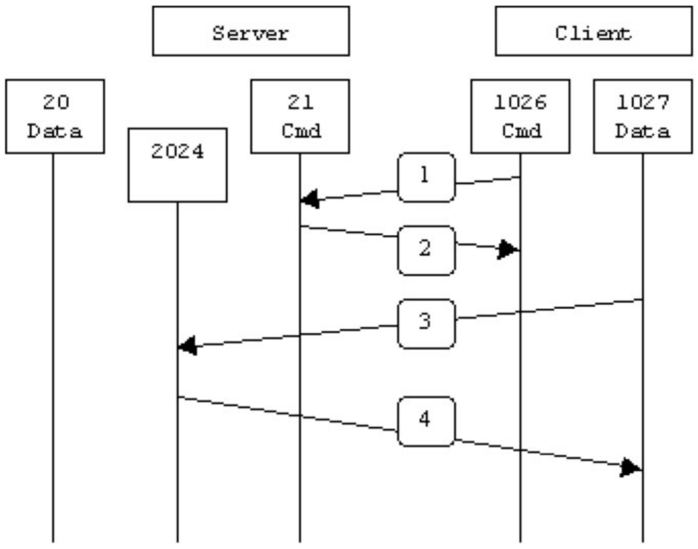
In order to resolve the issue of the server initiating the connection to the client a different method for FTP connections was developed. This was known as passive mode, or PASV, after the command used by the client to tell the server it is in passive mode.

In passive mode FTP the client initiates both connections to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. When opening an FTP connection, the client opens two random unprivileged ports locally (N > 1023 and N+1). The first port contacts the server on port 21, but instead of then issuing a PORT command and allowing the server to connect back to its data port, the client will issue the PASV command. The result of this is that the server then opens a random unprivileged port (P > 1023) and sends P back to the client in response to the PASV command. The client then initiates the connection from port N+1 to port P on the server to transfer data.

From the server-side firewall's standpoint, to support passive mode FTP the following communication channels need to be opened:

- FTP server's port 21 from anywhere (Client initiates connection)
- FTP server's port 21 to ports > 1023 (Server responds to client's control port)
- FTP server's ports > 1023 from anywhere (Client initiates data connection to random port specified by server
- FTP server's ports > 1023 to remote ports > 1023 (Server sends ACKs (and data) to client's data port)

When drawn, a passive mode FTP connection looks like this:



- The client contacts the server on the command port and issues the PASV command.
- The server then replies with PORT 2024, telling the client which port it is listening to for the data connection.

- The client then initiates the data connection from its data port to the specified server data port.
- Finally, the server sends back an ACK to the client's data port.

While passive mode FTP solves many of the problems from the client side, it opens up a whole range of problems on the server side. The biggest issue is the need to allow any remote connection to high numbered ports on the server. Fortunately, many FTP daemons, including the popular WU-FTPD allow the administrator to specify a range of ports, which the FTP server will use.

The second issue involves supporting and troubleshooting clients, which do (or do not) support passive mode. As an example, the command line FTP utility provided with Solaris does not support passive mode, necessitating a third-party FTP client, such as ncftp.

1.05 - Explain the purpose and functionality of SMTP

How E-mail Works

The SMTP Server

Whenever you send a piece of e-mail, your e-mail client interacts with the SMTP server to handle the sending. The SMTP server on your host may have conversations with other SMTP servers to deliver the e-mail. Let's assume that I want to send a piece of e-mail. My e-mail ID is brain, and I have my account on howstuffworks.com. I want to send e-mail to jsmith@mindspring.com. I am using a stand-alone e-mail client like Outlook Express.

When I set up my account at HowStuffWorks, I told Outlook Express the name of the mail server -- mail.howstuffworks.com. When I compose a message and press the Send button, here is what happens:

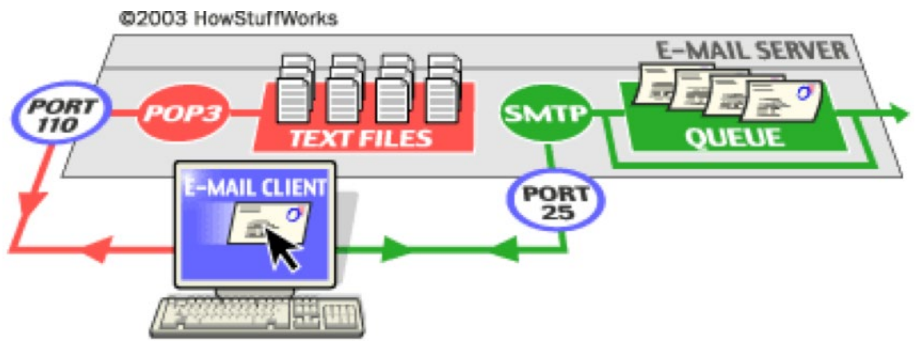
- Outlook Express connects to the SMTP server at mail.howstuffworks.com using port 25.
- Outlook Express has a conversation with the SMTP server, telling the SMTP server the address of the sender and the address of the recipient, as well as the body of the message.
- The SMTP server takes the "to" address (jsmith@mindspring.com) and breaks it into two parts: the recipient name (jsmith) and the domain name (mindspring.com). If the "to" address had been another user at howstuffworks.com, the SMTP server would simply hand the message to the POP3 server for howstuffworks.com (using a little program called the delivery agent). Since the recipient is at another domain, SMTP needs to communicate with that domain.
- The SMTP server has a conversation with a Domain Name Server, or DNS. It says, "Can you give me the IP address of the SMTP server for mindspring.com?" The DNS replies with the one or more IP addresses for the SMTP server(s) that Mindspring operates.

- The SMTP server at howstuffworks.com connects with the SMTP server at Mindspring using port 25. It has the same simple text conversation that my e-mail client had with the SMTP server for HowStuffWorks, and gives the message to the Mindspring server. The Mindspring server recognizes that the domain name for jsmith is at Mindspring, so it hands the message to Mindspring's POP3 server, which puts the message in jsmith's mailbox.

If, for some reason, the SMTP server at HowStuffWorks cannot connect with the SMTP server at Mindspring, then the message goes into a queue. The SMTP server on most machines uses a program called sendmail to do the actual sending, so this queue is called the sendmail queue. Sendmail will periodically try to resend the messages in its queue. For example, it might retry every 15 minutes. After four hours, it will usually send you a piece of mail that tells you there is some sort of problem. After five days, most sendmail configurations give up and return the mail to you undelivered.

The SMTP server understands very simple text commands like HELO, MAIL, RCPT and DATA. The most common commands are:

- HELO - introduce yourself
- EHLO - introduce yourself and request extended mode
- MAIL FROM: - specify the sender
- RCPT TO: - specify the recipient
- DATA - specify the body of the message (To, From and Subject should be the first three lines.)
- RSET - reset
- QUIT - quit the session
- HELP - get help on commands
- VRFY - verify an address
- EXPN - expand an address
- VERB – verbose



1.05 - Explain the purpose and functionality of a cookie

How Internet Cookies Work

How HTTP Cookies Work

A cookie is a piece of text that a Web server can store on a user's hard disk. Cookies allow a Web site to store information on a user's machine and later retrieve it. The pieces of information are stored as name-value pairs.

For example, a Web site might generate a unique ID number for each visitor and store the ID number on each user's machine using a cookie file.

If you use Microsoft's Internet Explorer to browse the Web, you can see all of the cookies that are stored on your machine. The most common place for them to reside is in a directory called `c:/windowscookies`. When I look in that directory on my machine, I find 165 files. Each file is a text file that contains name-value pairs, and there is one file for each Web site that has placed cookies on my machine.

You can see in the directory that each of these files is a simple, normal text file. You can see which Web site placed the file on your machine by looking at the file name (the information is also stored inside the file). You can open each file by clicking on it.

For example, I have visited goto.com, and the site has placed a cookie on my machine. The cookie file for goto.com contains the following information:

```
UserID A9A3BECE0563982D www.goto.com/
```

Goto.com has stored on my machine a single name-value pair. The name of the pair is UserID, and the value is `A9A3BECE0563982D`. The first time I visited goto.com, the site assigned me a unique ID value and stored it on my machine.

(Note that there probably are several other values stored in the file after the three shown above. That is housekeeping information for the browser.)

The vast majority of sites store just one piece of information—a user ID—on your machine. But a site can store many name-value pairs if it wants to.

A name-value pair is simply a named piece of data. It is not a program, and it cannot “do” anything. A Web site can retrieve only the information that it has placed on your machine. It cannot retrieve information from other cookie files, or any other information from your machine.

How do Web sites use cookies?

Cookies evolved because they solve a big problem for the people who implement Web sites. In the broadest sense, a cookie allows a site to store state information on your machine. This information lets a Web site remember what state your browser is in. An ID is one simple piece of state information -- if an ID exists on your machine, the site knows that you have visited before. The state is, “Your browser has visited the site at least one time,” and the site knows your ID from that visit.

Web sites use cookies in many different ways. Here are some of the most common examples:

Sites can accurately determine how many people actually visit the site. It turns out that because of proxy servers, caching, concentrators and so on, the only way for a site to accurately count visitors is to set a cookie with a unique ID for each visitor. Using cookies, sites can determine how many visitors arrive, how many are new versus repeat visitors and how often a visitor has visited. Sites can store user preferences so that the site can look different for each visitor (often referred to as customization). For example, if you visit msn.com, it offers you the ability to “change content/layout/color.” It also allows you to enter your zip code and get customized weather information. When you enter your zip code, the following name-value pair gets added to MSN’s cookie file:

```
WEAT CC=NC%5FRaleigh%2DDurham®ION= www.msn.com/
```

- Since I live in Raleigh, N.C., this makes sense.
- Most sites seem to store preferences like this in the site’s database and store nothing but an ID as a cookie, but storing the actual values in name-value pairs is another way to do it.

E-commerce sites can implement things like shopping carts and “quick checkout” options. The cookie contains an ID and lets the site keep track of you as you add different things to your cart. Each item you add to your shopping cart is stored in the site’s database along with your ID value.

When you check out, the site knows what is in your cart by retrieving all of your selections from the database. It would be impossible to implement a convenient shopping mechanism without cookies or something like them.

In all of these examples, note that what the database is able to store are items you have selected from the site, pages you have viewed from the site, information you have given to the site in online forms, etc. All of the information is stored in the site’s database, and in most cases, a cookie containing your unique ID is all that is stored on your computer.

1.05 – Given a situation in which a client connects to a remote host, explain how the name resolution process occurs

REF 1 p 970 - 982

Name Resolution Process

If a user wants to connect to an application or website on a network, either public or private, they must know where to go to get to the application. As we have been discussing in the prior sections, all of the locations on a network have an IP address that can be connected to for the residing application or service. The challenge is remembering all of the IP addresses of the sites or applications that we need to connect to on a day-to-day basis. It is easier for a human to remember a name than an IP address. For instance it is easier to remember www.google.com than 74.125.229.178. However to use a name instead of an IP address means a record of what names correlate to which IP address has to be maintained and be made available to search by all network attached nodes. The Domain Name System (DNS) is a hierarchical distributed naming system used for this function.

In the DNS name resolution process, for a client to connect to a remote host, quite a few steps take place. Let's go through those steps at a high level.

A user wants to get to the website www.google.com. So he/she opens a browser and types www.google.com in the path bar in the browser. The users system doesn't inherently know the IP address of the website. So it does a DNS lookup of the name and the process is basically like this:

- The user system will look in its local host file for a matching entry for www.google.com. If an entry matches it will use that IP address, and the process ends here.

- If an entry is not found in the local host file, the user system will look to its local name cache for a matching entry for www.google.com. If an entry exists then it will attempt to connect to that address, and the process ends here.

- If there is no record in the user system's cache then the system will request the IP address from its local DNS server for www.google.com.

- If the local DNS server has an entry for www.google.com in its cache then it will respond to the user system with that IP address. The system will put the entry in its cache and connect to the address. The process is then complete.

- If the local DNS server does not have an entry for www.google.com in its cache then the local DNS server will make a call to the root servers in its DNS configuration for the IP address of the Authoritative Name Server for the .com domain. The Local DNS server will then query the Authoritative Name Server of the .com domain for the Authoritative Name Server of the .google.com domain. Once the local DNS server has that IP address, it will query the Authoritative Name Server of the .google.com domain for the A record (IP address) of www.google.com.

- The local DNS server will place that response into its cache and then it will respond to the user system with that IP address. The user system will put the entry in its cache and connect to the address. The process is then complete.

All of these steps will take place extremely fast and only add milliseconds to the process of the connection to the website but for every name that the user system is told to connect to, this process takes place.

1.05 - Explain the purpose and functionality of a URL

URL - Uniform Resource Locator

A uniform resource locator (URL) is a reference to a resource that specifies the location of the resource on a computer network and a mechanism for retrieving it. A URL is a specific type of uniform resource identifier (URI), although many people use the two terms interchangeably. A URL implies the means to access an indicated resource, which is not true of every URI. URLs occur most commonly to reference web pages (HTTP), but are also used for file transfer (FTP), email (mailto), database access (JDBC), and many other applications.

Most web browsers display the URL of a web page above the page in an address bar.

Every HTTP URL consists of the following, in the given order. Several schemes other than HTTP also share this general format, with some variation.

The syntax is:

```
scheme://[user:password@]domain:port/path?query_string#fragment_id
```

Component details:

- The scheme, which in many cases is the name of a protocol (but not always), defines how the resource will be obtained. Examples include HTTP, HTTPS, FTP, file and many others. Although schemes are case-insensitive, the canonical form is lowercase.

- The domain name or literal numeric IP address gives the destination location for the URL. A literal numeric IPv6 address may be given, but must be enclosed in [ ]. e.g. `[db8:0cec::99:123a]`.

- The domain google.com, or its numeric IP address `173.194.34.5`, is the address of Google's website.

- The domain name portion of a URL is not case sensitive since DNS ignores case: `http://en.example.org/ and HTTP://EN.EXAMPLE.ORG/` both open the same page.

- The port number, given in decimal, is optional; if omitted, the default for the scheme is used.

  For example, `http://vnc.example.com:5800` connects to port 5800 of vnc.example.com, which may be appropriate for a VNC remote control session. If the port number is omitted for an http: URL, the browser will connect on port 80, the default HTTP port. The default port for an https: request is 443.

- The path is used to specify and perhaps find the resource requested. This path may or may not describe folders on the file system in the web server. It may be very different from the arrangement of folders on the web server. It is case-sensitive, [14] though it may be treated as case-insensitive by some servers, especially those based on Microsoft Windows.

  If the server is case sensitive and `http://en.example.org/wiki/URL` is correct, then `http://en.example.org/WIKI/URL` or `http://en.example.org/wiki/url` will display an HTTP 404 error page, unless these URLs point to valid resources themselves.

- The query string contains data to be passed to software running on the server. It may contain name/value pairs separated by ampersands, for example `?first_name=John&last_name=Doe`.

- The fragment identifier, if present, specifies a part or a position within the overall resource or document.

  When used with HTML, it usually specifies a section or location within the page, and used in combination with Anchor elements or the "id" attribute of an element, the browser is scrolled to display that part of the page.

The scheme name defines the namespace, purpose, and the syntax of the remaining part of the URL. Software will try to process a URL according to its scheme and context. For example, a web browser will usually dereference the URL `http://example.org:80` by performing an HTTP request to the host at `example.org`, using port number 80.

Other schemes do not follow the HTTP pattern. For example, the mailto scheme only uses valid email addresses. When clicked on in an application, the URL `mailto:bob@example.com` may start an e-mail composer with the address `bob@example.com` in the To field. The tel scheme is even more different; it uses the public switched telephone network for addressing, instead of domain names representing Internet hosts.

SECTION 2 - F5 SOLUTIONS AND TECHNOLOGY

Objective - 2.01 Articulate the role of F5 products

2.01 - Explain the purpose, use, and benefits of APM, LTM, ASM, GTM

BIG-IP Access Policy Manager (APM)

Today, business resources, such as applications and data, are accessed inside and outside the traditional business perimeter. Local and remote employees, partners, and customers often access applications without context or security. A central policy control point delivers access based on context and is critical to managing a scalable, secure, and dynamic environment.

BIG-IP Access Policy Manager (APM) is a flexible, high-performance access and security solution that provides unified global access to your applications and network. By converging and consolidating remote access, LAN access, and wireless connections within a single management interface, and providing easy-to-manage access policies, BIG-IP APM helps you free up valuable IT resources and scale cost-effectively

Provide unified global access

BIG-IP APM protects your public-facing applications by providing policy-based, context-aware access to users while consolidating your access infrastructure. It also provides secure remote access to corporate resources from all networks and devices. BIG-IP APM is the most scalable remote access solution and it is IPv6 ready.

Obtain flexibility, high performance, and scalability

BIG-IP APM is available in three deployment options—as an add-on module for BIG-IP Local Traffic Manager (LTM) for protecting Internet-facing applications, delivered in BIG-IP Edge Gateway for accelerated remote access, and run on BIG-IP LTM Virtual Edition to deliver flexible application access in virtualized environments.

Consolidate and simplify

BIG-IP APM helps you consolidate your infrastructure and simplify access management by providing centralized authentication, authorization, and accounting (AAA) control directly on the BIG-IP system. BIG-IP APM integrates with Oracle Access Manager. It simplifies virtual application deployment by supporting Citrix XenApp and XenDesktop, VMware View, Microsoft Remote Desktop Protocol (RDP), and Java RDP, all in one

WebTop. With BIG-IP APM, you can consolidate and unify elements such as access, security, and policy management to help further reduce costs.

The purpose of the Access Policy Manager is to create a secure access to internal applications by using a single authentication and provide control using a single management interface.

BIG-IP Local Traffic Manager (LTM)

BIG-IP Local Traffic Manager (LTM) turns your network into an agile infrastructure for application delivery. It's a full proxy between users and application servers, creating a layer of abstraction to secure, optimize, and load balance application traffic. This gives you the flexibility and control to add applications and servers easily, eliminate downtime, improve application performance, and meet your security requirements.

Easily deploy applications and ensure availability

BIG-IP LTM gives you the industry's most advanced load balancing and application health monitoring capabilities. F5 iApp enables you to easily deploy and manage applications using user-defined application centric configuration templates. The associated application related statistics provide complete visibility into the health and performance of your apps for faster troubleshooting and resolution of issues.

Accelerate your applications up to 3x

BIG-IP LTM reduces traffic volumes and minimizes the effect of client connection bottlenecks as well as WAN, LAN, and Internet latency to improve application and replication performance up to 3x. You can achieve additional performance increases with BIG-IP Application Acceleration Manager (see section below).

Take control over application delivery

The F5 TMOS platform gives you complete control of the connection, packets, and payload for applications. Using F5's event driven iRules you can customize how you intercept, inspect, transform, and direct inbound and outbound application traffic. The F5 iControl API makes it easy to integrate with third-party management systems. Device Service Clustering provides flexible high-availability scaling and configuration syncing of live application traffic among a cluster of active or standby BIG-IP devices

Reduce servers, bandwidth, and management costs

Advanced TCP connection management, TCP optimization, and server offloading enable you to optimize the utilization of your existing infrastructure—tripling server capacity and reducing bandwidth costs by up to 80 percent. BIG-IP LTM helps you simplify system management by consolidating security, acceleration, and availability in one easy-to-manage platform. By using fewer servers, less bandwidth, less power, and less cooling, while reducing the time spent managing your infrastructure, you can significantly reduce your operational costs.

The purpose of the Local Traffic Manager is to load balance applications in your environment by using advanced TCP connection management, TCP optimization and server offloading and also provides a high security solution. The LTMs iApps functionality is a powerful set of features that enable you to manage application services rather than individual devices and objects.

BIG-IP Application Security Manager (ASM)

F5 BIG-IP Application Security Manager (ASM) is a flexible web application firewall that secures web applications in traditional, virtual, and private cloud environments. BIG-IP ASM provides unmatched web application and website protection, helps secure deployed applications against unknown vulnerabilities, and enables compliance for key regulatory mandates—all on a platform that consolidates application delivery with data center firewall capabilities, and network and application access control.

Deliver comprehensive security

BIG-IP ASM blocks web application attacks in minutes to help protect against a broad spectrum of threats, including the latest distributed denial-of-service (DDoS) and SQL injection attacks. It also helps secure interactive web applications that use the latest coding, such as AJAX widgets and JSON payloads. Advanced vulnerability assessment integrations can scan web applications and BIG-IP ASM patches vulnerabilities in minutes to help protect against web threats. BIG-IP ASM stops hackers and attacks from any location and ensures that legitimate users can access applications.

The purpose of the Application Security Manager is to secure web applications using a certified web application firewall and offer threat assessment and visibility.

BIG-IP Global Traffic Manager (GTM)

F5 BIG-IP Global Traffic Manager (GTM) distributes DNS and user application requests based on business policies, data center and network conditions, user location, and application performance. BIG-IP GTM delivers F5's high-performance DNS Services with visibility, reporting, and analysis; scales and secures DNS responses geographically to survive DDoS attacks; delivers a complete, real-time DNSSEC solution; and ensures global application high availability.

Control and ensure app availability

BIG-IP GTM helps you create a strong disaster recovery and business continuity plan by ensuring that users are always connected to a site where the application is available. In addition to performing comprehensive health checking of the entire infrastructure, BIG-IP GTM minimizes downtime and improves the user experience by determining health at the application layer for every user.

Improve application performance

BIG-IP GTM enables you to send users to a site that will give them the best application experience. It uses a range of different load balancing methods and intelligent monitoring for each specific application and user. Traffic is routed according to your business policies and current network and user conditions. BIG-IP GTM includes an accurate, granular geo-location database, giving you control of traffic distribution based on the user's location. By providing persistence for stateful applications, BIG-IP GTM helps you eliminate broken sessions and corrupted data.

The purpose of the Global Traffic Manager is to ensure availability and access to the applications in your environment by using comprehensive health checks and load balancing methods to determine what site the user should access to get the best application experience.

BIG-IP Application Acceleration Manager (AAM)

F5 BIG-IP Application Acceleration Manager (AAM) overcomes network, protocol, and application issues to help you meet application performance, data replication, and disaster recovery requirements presented by cloud, mobile applications, and video distribution. By offloading your network and servers, BIG-IP AAM decreases the need for additional bandwidth and hardware. Users get fast access to applications, and you gain greater revenue and free up IT resources for other strategic projects.

BIG-IP AAM can optimize a wide variety of protocols delivered to a client browser, a desktop application, or another BIG-IP device, depending on the deployment. Optimizations are divided into data center optimizations, including server and network optimizations, transport optimizations, and application delivery optimizations, including application protocol and web performance optimizations.

The purpose of the Application Acceleration Manager is to overcome WAN latency, maximizes server capacity, and speeds application response times. AAM decreases the need for additional bandwidth and hardware so users get fast access to applications, while you gain greater revenue and free up IT resources.

BIG-IP AFM Datasheet

BIG-IP Advanced Firewall Manager (AFM)

F5 BIG-IP Advanced Firewall Manager (AFM) is a high-performance, stateful, full-proxy network firewall designed to guard data centers against incoming threats that enter the network on the most widely deployed protocols—including HTTP/S, SMTP, DNS, and FTP. By aligning firewall policies with the applications they protect, BIG-IP AFM streamlines application deployment, security, and monitoring. With its scalability, security, and simplicity, BIG-IP AFM forms the core of the F5 application delivery firewall solution.

BIG-IP AFM is the core of the F5 application delivery firewall solution—the first of its kind in the industry, which combines the network firewall with traffic management, application security, user access management, and DNS security. By consolidating the security functions of several BIG-IP modules onto a single platform, the F5 application delivery firewall reduces management complexity and overhead, while still maintaining superior performance and scalability. Building upon BIG-IP Local Traffic Manager, the application delivery firewall has deep application fluency in the most widely deployed enterprise applications. This makes it ideal for protecting Internet-facing data center applications, wherever they reside.

The purpose of the Application Delivery Firewall is to combine the network firewall with anti-DDoS, traffic management, application security, user access management, and DNS security. By integrating these core datacenter features, F5 application delivery firewall reduces management complexity and overhead and is ideal for protecting internet-facing data centers wherever they reside.

Objective - 2.02 Explain the purpose, use, and advantages of iRules

2.02 - Explain the purpose of iRules

DevCentral iRules

What is an iRule?

An iRule is a script that you write if you want to make use of some of the extended capabilities of the BIG-IP that are unavailable via the CLI or GUI. iRules allow you to more directly interact with the traffic passing through the device. Using iRules, you can send traffic not only to pools, but also to individual pool members, ports, or URIs. And directing traffic to a desired pool is only the beginning. You can parse the entire header and payload of the data as it is being passed through the BIG-IP and, at wire speed, execute an entire script of commands on that traffic. The commands at your disposal range from logging to redirecting traffic, from modifying the URI or port to actually rewriting the payload itself.

The iRules you create can be simple or sophisticated, depending on your content-switching needs. The following shows an example of a simple iRule.

This iRule is triggered when a client-side connection has been accepted, causing the LTM system to send the packet to the pool my_pool, if the client's address matches `10.10.10.10`.

Using a feature called the Universal Inspection Engine (UIE), you can write an iRule that searches either a header of a packet, or actual packet content, and then directs the packet based on the result of that search. iRules can also direct packets based on the result of a client authentication attempt.

iRules can direct traffic not only to specific pools, but also to individual pool members, including port numbers and URI paths, either to implement persistence or to meet specific load balancing requirements.

The syntax that you use to write iRules is based on the Tool Command Language (Tcl) programming standard. Thus, you can use many of the standard Tcl commands, plus a robust set of extensions that the LTM system provides to help you further increase load balancing efficiency.

The advantage of iRules is that you extend the capabilities of the BIG-IP that is not available through the CLI or the GUI.

2.02 - Explain the advantages of iRules

DevCentral iRules

How does an iRule work?

To start at the beginning, as it were, an iRule is first and foremost a configuration object, in F5 terms. This means that it is a part of your general bigip.conf along with your pools, virtual servers, monitors, etc. It is entered into the system either via the GUI or CLI, generally speaking. There is also an iRules Editor available for download on DevCentral that is a windows tool for editing and deploying/testing iRules, which can be extremely useful. Unlike most configuration objects, though, an iRule is completely user generated and customizable. An iRule is a script, at its core after all. Regardless of how an iRule gets there, be it UI, CLI or Editor, once an iRule is part of your config, it is then compiled as soon as that configuration is saved.

One of the gross misconceptions about iRules is that, as with most interpreted scripting languages such as TCL, and interpreter must be instantiated every time an iRule is executed to parse the code and process it. This is not true at all, because every time you save your configuration all of your iRules are pre-compiled into what is referred to as “byte code”. Byte code is mostly compiled and has the vast majority of the interpreter tasks already performed, so that TMM can directly interpret the remaining object. This makes for far higher performance and as such increases scalability.

Now that the iRule is saved and pre-compiled, it must then be applied to a virtual server before it can affect any traffic. An iRule that is not applied to a virtual server is effectively disabled, for all intents and purposes. Once you’ve applied an iRule to a given virtual server, however, it will now technically be applied against all traffic passing through that virtual server. Keep in mind though, that this does not necessarily mean that all traffic passing through the virtual server in question will be affected. IRules are most often very selective in which traffic they affect, be it to modify, re-route or otherwise. This is done through both logical constructs within the iRules, but also through the use of events within the iRule itself.

Events are one of the ways in which iRules have been made to be network aware, as a language. An event, which we’ll dig into in much more detail in the next installment of this series, is a way of executing iRules code at a given point in time within the flow of a networking session. If I only want to execute a section of code once for each new connection to the virtual server to which my iRule is applied, I could easily do so by writing some simple code in the appropriate event. Events are also important because they indicate at which point in the proxy chain (sometimes referred to as a hud chain) an iRule executes. Given that BIG-IP is a bidirectional proxy, it is important for iRules to execute on not only the right side of the proxy, but at the right moment in the network flow.

So now you have an iRule added to your configuration, it has been automatically pre-compiled to byte code when the configuration was saved, you have it applied to the appropriate virtual server, and the code within the iRule calls out the desired event in which you want your code to execute; now is when the magic happens, as it were. This is where the massive collection of iRules commands comes into play. From header modification to full on payload replacement to creating a socket connection to an outside system and making a request before processing traffic for your virtual, there are very few limitations to what can be achieved when combining the appropriate series of iRules commands. Those commands are then processed by TMM, which will affect whatever change(s) it needs to the traffic it is processing for the given session, depending on what you've designed your iRule to do. The true power of iRules largely comes into play thanks to the massive array of custom commands that we've built into the language, allowing you to leverage your BIG-IP to the fullest.

2.02 - Given a list of situations, determine which would be appropriate for the use of iRules

DevCentral iRules

When would I use an iRule?

The ideal time to use an iRule is when you're looking to add some form of functionality to your application or app deployment, at the network layer, and that functionality is not already readily available via the built in configuration options in your BIG-IP. Whether it's looking to perform some kind of custom redirect or logging specific information about users' sessions or a vast array of other possibilities, iRules can add valuable business logic or even application functionality to your deployment. iRules have a single point of management, your BIG-IP, as opposed to being distributed to every server hosting whichever application you're trying to modify or affect. This can save valuable management time, and can also be a large benefit in time to deployment. It is often far easier to deploy an iRule or an iRule change than it is to modify your application for a quick fix.

As an example, one of the most common uses of iRules when it was first introduced was to redirect all traffic from HTTP (port 80) to HTTPS (port 443) without affecting either the host or the requested URI for the connection. (See example below) This was and still is a very simple iRule, but it wasn't at the time a feature available in the standard configuration options for BIG-IP.

```
when HTTP_REQUEST {
HTTP::redirect "https://[HTTP::host][HTTP::uri]"
}
```

When would I not use an iRule?

The above example of an HTTP to HTTPS redirect iRule actually depicts perfectly when to not use an iRule, because that functionality was so popular that it has since been added as a profile option directly in the BIG- IP configuration. As such, it is more appropriate and technically higher performance, to use that feature in the profile as opposed to writing an iRule to perform the same task. A general rule of thumb is: Any time you can do something from within the standard config options, profiles, GUI or CLI – do it there first. If you're looking to perform a task that can't be accomplished via the "built-in" means of configuration, then it is a perfect time to turn to iRules to expand the possibilities.

The main reason for this is performance. iRules are extremely high performance as a rule, if written properly. But there is always a slight benefit in performance when you can run functionality directly from built in, core features as opposed to a custom created script, even an iRule. Also, though, it is easier to maintain a feature built into the product through upgrades, rather than re-testing and managing an iRule that could be easily replaced with a few configuration options.

Objective - 2.03 Explain the purpose, use, and advantages of iApps

2.03 - Explain the purpose of iApps

Managing iApp Template Files with iControl

What's an iApp?

An iApp is a user-customizable framework for deploying applications. It consists of three components: Templates, Application Services, and Analytics. An iApp Template is where the application is described and the objects (required and optional) are defined through presentation and implementation language. An iApp Application Service is the deployment process of an iApp Template which bundles the entire configuration options for a particular application together. You would have an iApp Application Service for SharePoint, for example. iApp Analytics include performance metrics on a per-application and location basis. Benefits of using iApp:

- User-customizable
- Easy editing of configurations and cleanup

- Reentrancy
- Configuration encapsulation
- Cradle-to-grave configuration management
- Strictness protects against accidental changes to the configuration
- Copy/Import/Export capability
- Community support for DevCentral hosted templates

2.03 - Explain the advantages of iApps

The Parts of iApps

About iApp Templates

iApps templates create configuration-specific forms used by application services to guide authorized users through complex system configurations. The templates provide programmatic, visual layout and help information. Each new application service uses one of the templates to create a screen with fields and guide the user through the configuration process and creates the configuration when finished.

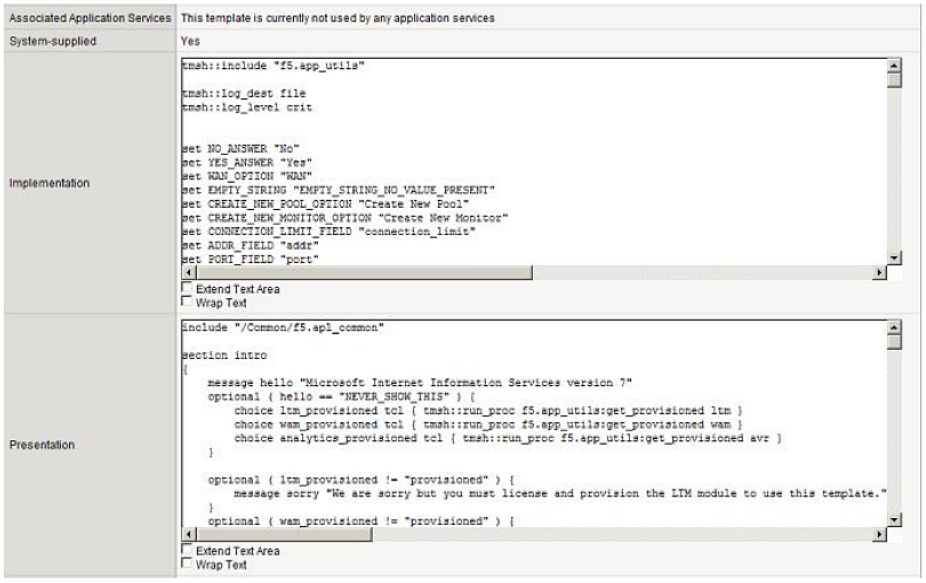
The templates allow users to customize by either modifying an existing template or creating one from scratch. Users can create scratch-built templates using either the iApps Templates screen or any text-editing software

The BIG-IP system comes with several system-supplied templates that you can use as-is to create your application services. You can also use the system-supplied templates as a starting point for your own templates, or you can write templates from scratch using, variously, tmsh and TCL for the back-end template implementation section.

Template sections

Templates have three sections; presentation, implementation, and help.

- The presentation section collects user entries.
- The implementation section uses user entries to build a configuration that will control traffic.
- The help section documents the template and its presentation to users when creating an application service.



Associated Application Services	This template is currently not used by any application services
System-supplied	Yes

Implementation

```
tmsh::include "f5.app_utils"

tmsh::log_dest file
tmsh::log_level crit


set NO_ANSWER "No"
set YES_ANSWER "Yes"
set WAN_OPTION "WAN"
set EMPTY_STRING "EMPTY_STRING_NO_VALUE_PRESENT"
set CREATE_NEW_POOL_OPTION "Create New Pool"
set CREATE_NEW_MONITOR_OPTION "Create New Monitor"
set CONNECTION_LIMIT_FIELD "connection_limit"
set ADDR_FIELD "addr"
set PORT_FIELD "port"
```

Presentation

```
include "/Common/f5.apl_common"

section intro
{
    message hello "Microsoft Internet Information Services version 7"
    optional ( hello == "NEVER_SHOW_THIS" ) {
        choice ltm_provisioned tcl { tmsh::run_proc f5.app_utils:get_provisioned ltm }
        choice wam_provisioned tcl { tmsh::run_proc f5.app_utils:get_provisioned wam }
        choice analytics_provisioned tcl { tmsh::run_proc f5.app_utils:get_provisioned avr }
    }

    optional ( ltm_provisioned != "provisioned" ) {
        message sorry "We are sorry but you must license and provision the LTM module to use this template."
    }
    optional ( wam_provisioned != "provisioned" ) {
```

2.03 - Given a list of situations, determine which would be appropriate for the use of iApps

F5 iApps: Moving Application Delivery Beyond the Network

When do you use an iApp?

As organizations begin moving to more modular cloud and SaaS models, managing applications becomes more important than building infrastructure. Many of the benefits that come from moving to a more agile model are not associated with managing the infrastructure; yet managing application deployments, performance, and availability in cloud and SaaS environments is often difficult because the application is still tied to infrastructure. iApps bind application control, visibility, and management to the infrastructure required to deliver those applications and services beyond the data center.

iApps support the architecture that transforms a network from a static resource comprising isolated components to a unified, flexible, and resilient pool of resources directly associated with an application or service. This enables rapid network deployment, integration, management, and visibility at the application layer. iApps provide complete control over the entire application delivery infrastructure by adapting the network to the application. The resulting quick deployment and single-point capabilities save operational costs. For the first time, organizations can create a common and highly reusable catalogue for security, acceleration, and availability services at a strategic point of control to dramatically increase the organizational agility and efficiency of F5 BIG-IP devices.

With iApps, F5 has created a paradigm shift in how administrators view and manage the network by moving management responsibility from the network components to the application. iApps increase IT agility and efficiency by enabling organizations to manage the security, optimization, availability, health, and performance of not only the ADN devices in the network, but of the mission-critical applications running the business. In this way, iApps create a truly unified Application Delivery Network.

Objective - 2.04 Explain the purpose of and use cases for full proxy and packet forwarding/packet based architectures

2.04 - Describe a full proxy architecture

We often mention that the benefits derived from some application delivery controllers are due to the nature of being a full proxy. And in the same breath we might mention reverse, half, and forward proxies, which makes the technology sound more like a description of the positions on a sports team than an application delivery solution. So what does these terms really mean? Here's the lowdown on the different kinds of proxies in one concise guide.

Proxies

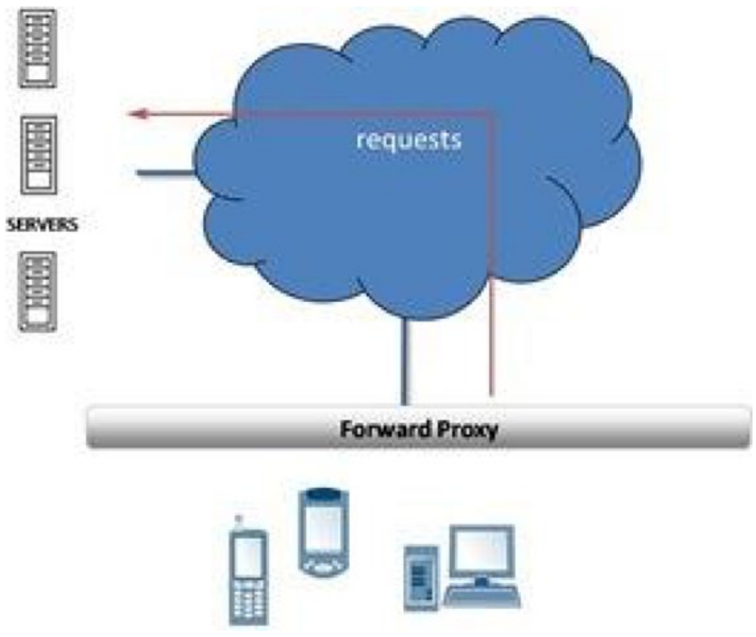
Proxies (often called intermediaries in the SOA world) are hardware or software solutions that sit between the client and the server and do something to requests and sometimes responses. The most often heard use of the term proxy is in conjunction with making Web surfing anonymous. That's because proxies sit between your browser and your desired destination and proxy the connection; that is you talk to the proxy while the proxy talks to the web server and neither you nor the web server know about each other.

Proxies are not all the same. Some are half proxies, some are full proxies; some are forward and some are reverse. Yes, that came excruciatingly close to sounding like a Dr. Seuss book.

Forward Proxies

Forward proxies are probably the most well known of all proxies, primarily because most folks have dealt with them either directly or indirectly. Forward proxies are those proxies that sit between two networks, usually a private internal network and the public Internet. Large service providers have also traditionally employed forward proxies as a bridge between their isolated network of subscribers and the public Internet, such as CompuServe and AOL in days gone by. These are often referred to as "mega-proxies" because they managed such high volumes of traffic.

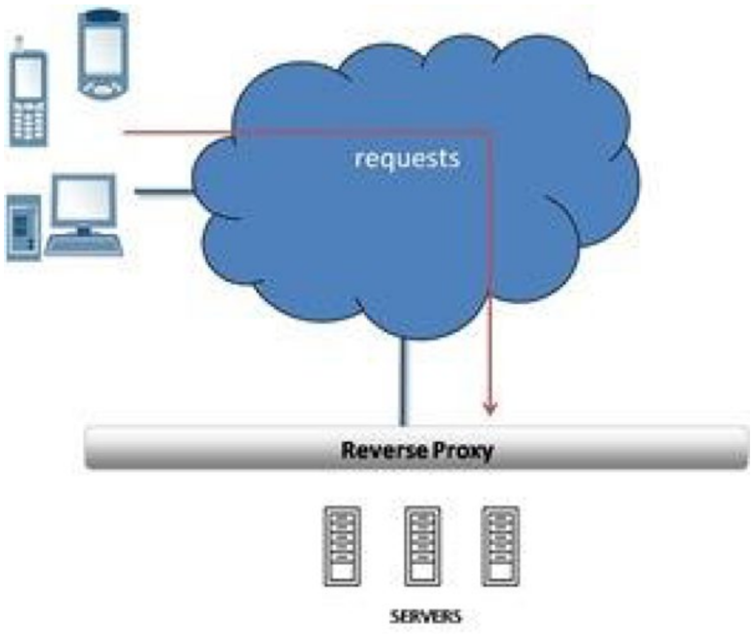
Forward proxies are generally HTTP (Web) proxies that provide a number of services but primarily focus on web content filtering and caching services. These forward proxies often include authentication and authorization as a part of their product to provide more control over access to public content. If you've ever gotten a web page that says "Your request has been denied by blah blah. If you think this is an error please contact the help desk/your administrator" then you've probably used a forward proxy.



Reverse Proxies

A reverse proxy is less well known, generally because we don't use the term anymore to describe products used as such. Load balancers (application delivery controllers) and caches are good examples of reverse proxies. Reverse proxies sit in front of web and application servers and process requests for applications and content coming in from the public Internet to the internal, private network. This is the primary reason for the name "reverse" proxy to differentiate it from a proxy that handles outbound requests.

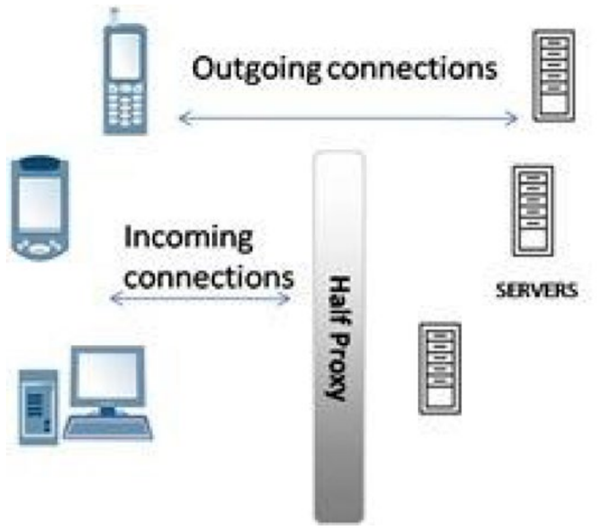
Reverse proxies are also generally focused on HTTP but in recent years have expanded to include a number of other protocols commonly used on the web such as streaming audio (RTSP), file transfers (FTP), and generally any application protocol capable of being delivered via UDP or TCP.



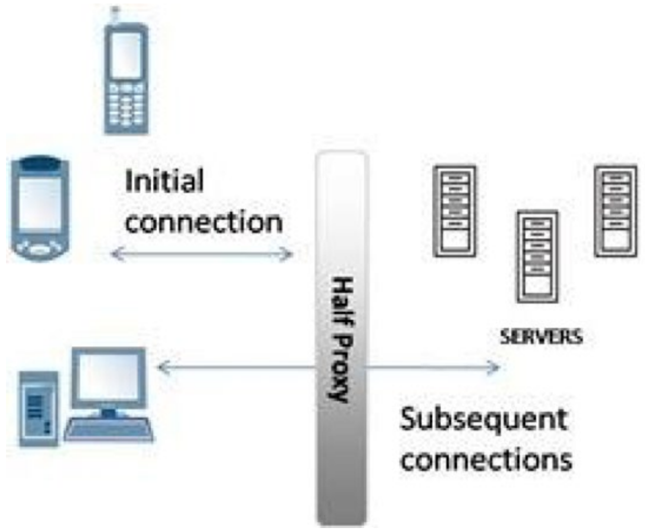
Half Proxies

Half-proxy is a description of the way in which a proxy, reverse or forward, handles connections. There are two uses of the term half-proxy: one describing a deployment configuration that affects the way connections are handled and one that describes simply the difference between a first and subsequent connections.

The deployment-focused definition of half-proxy is associated with a direct server return (DSR) configuration. Requests are proxied by the device, but the responses do not return through the device, but rather are sent directly to the client. For some types of data, particularly streaming protocols, this configuration results in improved performance. This configuration is known as a half-proxy, because only half the connection (incoming) is proxied while the other half, the response, is not.



The second use of the term “half-proxy” describes a solution in which the proxy performs what is known as delayed binding in order to provide additional functionality. This allows the proxy to examine the request before determining where to send it. Once the proxy determines where to route the request, the connection between the client and the server are “stitched” together. This is referred to as a half-proxy because the initial TCP handshaking and first requests are proxied by the solution, but subsequently forwarded without interception.



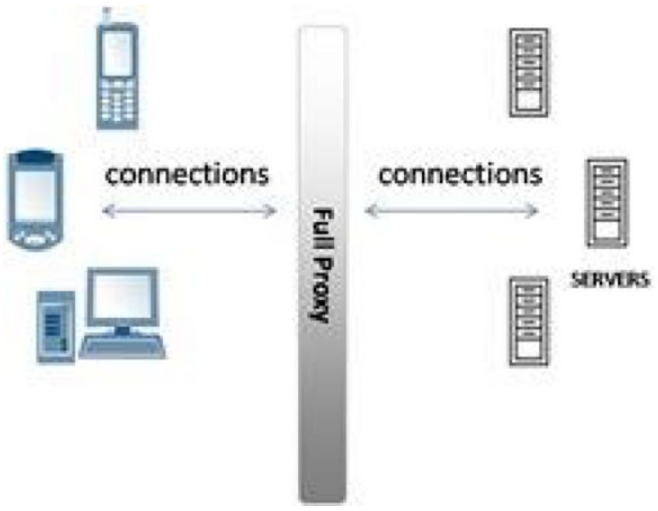
Half proxies can look at incoming requests in order to determine where the connection should be sent and can even use techniques to perform layer 7 inspection, but they are rarely capable of examining the responses. Almost all half-proxies fall into the category of reverse proxies.

Full Proxies

Full proxy is also a description of the way in which a proxy, reverse or forward, handles connections. A full proxy maintains two separate connections - one between itself and the client and one between itself and the destination server. A full proxy completely understands the protocols, and is itself an endpoint and an originator for the protocols. Full proxies are named because they completely proxy connections - incoming and outgoing.

Because the full proxy is an actual protocol endpoint, it must fully implement the protocols as both a client and a server (a packet-based design does not). This also means the full proxy can have its own TCP connection behavior, such as buffering, retransmits, and TCP options. With a full proxy, each connection is unique; each can have its own TCP connection behavior. This means that a client connecting to the full proxy device would likely have different connection behavior than the full proxy might use for communicating with servers. Full proxies can look at incoming requests and outbound responses and can manipulate both if the solution allows it.

Many reverse and forward proxies use a full proxy model today. There is no guarantee that a given solution is a full proxy, so you should always ask your solution provider if it is important to you that the solution is a full proxy.



2.04 - Describe a packet forwarding/packet based architecture

Packet Forwarding

Packet Forwarding

Packet forwarding is the relaying of packets from one network segment to another by nodes in a computer network. A unicast forwarding pattern, typical of many networking technologies including the overwhelming majority of Internet traffic a multicast-forwarding pattern, typical of PIM

A broadcast forwarding pattern, typical of bridged Ethernet

The Network Layer of the OSI Layer is responsible for Packet Forwarding. The simplest forwarding model—unicasting—involves a packet being relayed from link to link along a chain leading from the packet's source to its destination. However, other forwarding strategies are commonly used. Broadcasting requires a packet to be duplicated and copies sent on multiple links with the goal of delivering a copy to every device on the network. In practice, broadcast packets are not forwarded everywhere on a network, but only to devices within a broadcast domain, making broadcast a relative term. Less common than broadcasting, but perhaps of greater utility and theoretical significance, is multicasting, where a packet is selectively duplicated and copies delivered to each of a set of recipients.

Networking technologies tend to naturally support certain forwarding models. For example, fiber optics and copper cables run directly from one machine to another to form a natural unicast media – data transmitted at one end is received by only one machine at the other end. However, as illustrated in the diagrams, nodes can forward packets to create multicast or broadcast distributions from naturally unicast media. Likewise, traditional Ethernet (10BASE5 and 10BASE2, but not the more modern 10BASE-T) are natural broadcast media – all the nodes are attached to a single long cable and a packet transmitted by one device is seen by every other device attached to the cable. Ethernet nodes implement unicast by ignoring packets not directly addressed to them. A wireless network is naturally multicast – all devices within a reception radius of a transmitter can receive its packets. Wireless nodes ignore packets addressed to other devices, but require forwarding to reach nodes outside their reception radius.

At nodes where multiple outgoing links are available, the choice of which, all, or any to use for forwarding a given packet requires a decision making process that, while simple in concept, is sometimes bewilderingly complex. Since a forwarding decision must be made for every packet handled by a node, the total time required for this can become a major limiting factor in overall network performance. Much of the design effort of high-speed routers and switches has been focused on making rapid forwarding decisions for large numbers of packets.

The forwarding decision is generally made using one of two processes: routing, which uses information encoded in a device's address to infer its location on the network, or bridging, which makes no assumptions about where addresses are located and depends heavily on broadcasting to locate unknown addresses. The heavy overhead of broadcasting has led to the dominance of routing in large networks, particularly the Internet; bridging is largely relegated to small networks where the overhead of broadcasting is tolerable. However, since large networks are usually composed of many smaller networks linked together, it would be inaccurate to state that bridging has no use on the Internet; rather, its use is localized.

A network can use one of two different methods to forward packets: store-and-forward or cut through.

TMOS: Redefining the Solution

Packet-based Design

A network device with a packet-based (or packet-by-packet) design is located in the middle of a stream of communications, but is not an endpoint for those communications; it just passes the packets through. Often a device that operates on a packet-by-packet basis does have some knowledge of the protocols flowing through it, but is far from being a real protocol endpoint. The speed of these devices is primarily based on not having to understand the entire protocol stack, short-cutting the amount of work needed to handle traffic. For example, with TCP/IP, this type of device might only understand the protocols well enough to rewrite the IP addresses and TCP ports; only about half of the entire stack.

As networks became more complex and the need for intelligence increased, more advanced packet-based designs began to emerge (including the BIG-IP products from F5). These devices knew TCP/IP well enough to understand both TCP connection setup and teardown, modify TCP/IP headers, and even insert data into TCP streams. Because these systems could insert data into TCP streams and modify the content of the stream, they also had to rewrite the TCP sequence (SEQ) and acknowledgment (ACK) values for packets going back and forth from the client and server. F5's BIG-IP products understood TCP/IP and HTTP well enough to identify individual HTTP requests and could send different requests to different servers, reusing connections the BIG-IP device already had open.

While all of this is possible using a very sophisticated packet-by-packet architecture (BIG-IP devices are some of the most sophisticated of such designs to date), it required a very complex state tracking engine to understand the TCP/IP and HTTP protocols well enough to rewrite header contents, insert data, and maintain its own connections to clients and servers.

Despite this increasing complexity, packet-based designs are still less complex and faster than traditional proxy-based designs, as they have the advantage of only requiring a small percentage of the logic required for a full proxy.

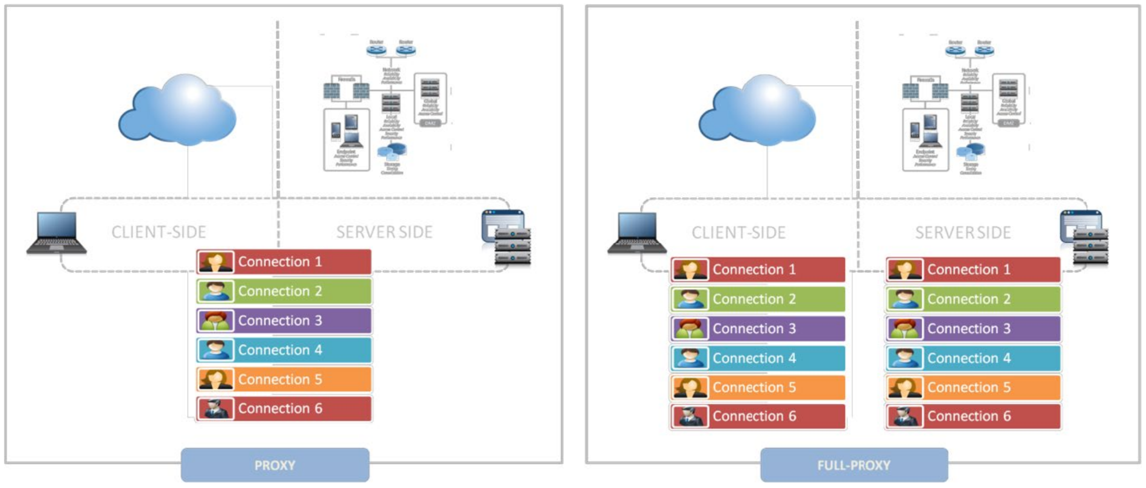
2.04 - Given a list of situations, determine which is appropriate for a full proxy architecture

The Full-Proxy Data Center Architecture

Full proxy architecture – What do they mean?

The reason there is a distinction made between "proxy" and "full-proxy" stems from the handling of connections as they flow through the device. All proxies sit between two entities – in the Internet age almost always "client" and "server" – and mediate connections. While all full-proxies are proxies, the converse is not true. Not all proxies are full-proxies and it is this distinction that needs to be made when making decisions that will impact the data center architecture.

A full-proxy maintains two separate session tables – one on the client-side, one on the server-side. There is effectively an “air gap” isolation layer between the two internal to the proxy, one that enables focused profiles to be applied specifically to address issues peculiar to each “side” of the proxy. Clients often experience higher latency because of lower bandwidth connections while the servers are generally low latency because they’re connected via a high-speed LAN. The optimizations and acceleration techniques used on the client side are far different than those on the LAN side because the issues that give rise to performance and availability challenges are vastly different.



A full-proxy, with separate connection handling on either side of the “air gap”, can address these challenges. A proxy, that may be a full-proxy but more often than not simply uses a buffer-and-stitch methodology to perform connection management, cannot optimally do so. A typical proxy buffers a connection, often through the TCP handshake process and potentially into the first few packets of application data, but then “stitches” a connection to a given server on the back-end using either layer 4 or layer 7 data, perhaps both. The connection is a single flow from end-to-end and must choose which characteristics of the connection to focus on – client or server – because it cannot simultaneously optimize for both.

The second advantage of a full-proxy is its ability to perform more tasks on the data being exchanged over the connection as it is flowing through the component. Because specific action must be taken to “match up” the connection as its flowing through the full-proxy, the component can inspect, manipulate, and otherwise modify the data before sending it on its way on the server-side. This is what enables termination of SSL, enforcement of security policies, and performance-related services to be applied on a per-client, per-application basis.

This capability translates to broader usage in data center architecture by enabling the implementation of an application delivery tier in which operational risk can be addressed through the enforcement of various policies. In effect, we're created a full-proxy data center architecture in which the application delivery tier as a whole serves as the "full proxy" that mediates between the clients and the applications.

The full-proxy data center architecture

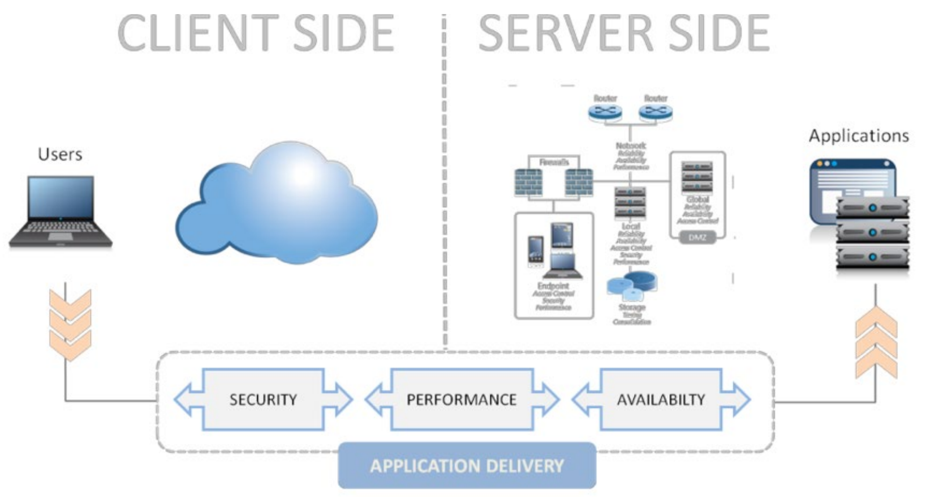
A full-proxy data center architecture installs a digital "air gap" between the client and applications by serving as the aggregation (and conversely disaggregation) point for services. Because all communication is funneled through virtualized applications and services at the application delivery tier, it serves as a strategic point of control at which delivery policies addressing operational risk (performance, availability, security) can be enforced.

A full-proxy data center architecture further has the advantage of isolating end-users from the volatility inherent in highly virtualized and dynamic environments such as cloud computing. It enables solutions such as those used to overcome limitations with virtualization technology, such as those encountered with Pod architectural constraints in VMware View deployments. Traditional access management technologies, for example, are tightly coupled to host names and IP addresses. In a highly virtualized or cloud computing environment, this constraint may spell disaster for either performance or ability to function, or both. By implementing access management in the application delivery tier – on a full-proxy device – volatility is managed through virtualization of the resources, allowing the application delivery controller to worry about details such as IP address and VLAN segments, freeing the access management solution to concern itself with determining whether this user on this device from that location is allowed to access a given resource.

Basically, we're taking the concept of a full-proxy and expanded it outward to the architecture. Inserting an "application delivery tier" allows for an agile, flexible architecture more supportive of the rapid changes today's IT organizations must deal with.

Such a tier also provides an effective means to combat modern attacks. Because of its ability to isolate applications, services, and even infrastructure resources, an application delivery tier improves an organizations' capability to withstand the onslaught of a concerted DDoS attack. The magnitude of difference between the connection capacity of an application delivery controller and most infrastructures (and all servers) gives the entire architecture a higher resiliency in the face of overwhelming connections. This ensures better availability and, when coupled with virtual infrastructure that can scale on-demand when necessary, can also maintain performance levels required by business concerns.

A full-proxy data center architecture is an invaluable asset to IT organizations in meeting the challenges of volatility both inside and outside the data center.



An application delivery tier isolates users from applications and infrastructure (and vice-versa) by transparently applying the appropriate security, performance, and availability policies to inbound and outbound traffic.

2.04 - Given a list of situations, determine which is appropriate for a packet based architecture

TMOS: Redefining the Solution

What is a packet-based design?

A network device with a packet-based (or packet-by-packet) design is located in the middle of a stream of communications, but is not an endpoint for those communications; it just passes the packets through. Often a device that operates on a packet-by-packet basis does have some knowledge of the protocols flowing through it, but is far from being a real protocol endpoint. The speed of these devices is primarily based on not having to understand the entire protocol stack, shortcutting the amount of work needed to handle traffic. For example, with TCP/IP, this type of device might only understand the protocols well enough to rewrite the IP addresses and TCP ports; only about half of the entire stack.

As networks became more complex and the need for intelligence increased, more advanced packet-based designs began to emerge (including the BIG-IP products from F5). These devices knew TCP/IP well enough to understand both TCP connection setup and teardown, how to modify TCP/IP headers, and even insert data into TCP streams.

Because these systems could insert data into TCP streams and modify the content of the stream, they also had to rewrite the TCP sequence (SEQ) and acknowledgment (ACK) values for packets going back and forth from the client and server. F5's BIG-IP products understood TCP/IP and HTTP well enough to identify individual HTTP requests and could send different requests to different servers, reusing connections the BIG-IP device already had open.

While all of this is possible using a very sophisticated packet-by-packet architecture (BIG-IP devices are some of the most sophisticated of such designs to date), it required a very complex state tracking engine to understand the TCP/IP and HTTP protocols well enough to rewrite header contents, insert data, and maintain its own connections to clients and servers. Despite this increasing complexity, packet-based designs are still less complex and faster than traditional proxy-based designs, as they have the advantage of only requiring a small percentage of the logic required for a full proxy.

What is a proxy-based design (full proxy)?

A full-proxy design is the opposite of a packet-by-packet design. Instead of having a minimal understanding of the communications streaming through the device, a full proxy completely understands the protocols, and is itself an endpoint and an originator for the protocols. The connection between a client and the full proxy is fully independent of the connection between the full proxy and the server; whereas in a packet-by-packet design, there is essentially a direct communication channel between the client and the server (although the device in the middle may manipulate the packets going back and forth).

Because the full proxy is an actual protocol endpoint, it must fully implement the protocols as both a client and a server (a packet-based design does not). This also means the full proxy can have its own TCP connection behavior, such as buffering, retransmits, and TCP options. With a full proxy, each connection is unique; each can have its own TCP connection behavior. This means that a client connecting to the full-proxy device would likely have different connection behavior than the full proxy might use for communicating with the backend servers.

Therefore, a full proxy allows for the optimization of every connection uniquely, regardless of the original source and the final destination. Further, a full proxy understands and processes each protocol as a real client or server would, using layers. Using HTTP as an example, first the IP protocol is processed, then TCP, then HTTP; and each layer have no knowledge of the lower layers.

Redefining the Solution

It is common knowledge that proxy-based solutions, or at least the intelligence offered by them, were the ultimate solution. However, the vastly superior performance of packet-by-packet designs more than made up for their limited intelligence. For a while, this was an acceptable trade-off for most enterprise networks. As the need for increased intelligence grows, packet-based solutions are quickly experiencing the same performance restrictions that proxy-based solutions have always suffered from. And the development complexity of packet-based solutions is quickly approaching that of proxy-based designs as well. Despite dramatic increases in hardware and software power, packet-by-packet designs are unable to keep up with the need for intelligence and performance. It is no longer acceptable to have to choose between them. While packet-based solutions had their time, that time is gone. It is now readily apparent that shortcutting intelligence in lieu of performance did not adequately provide a viable solution. The real solution is to build a proxy-based solution with the performance of the packet-based solution.

Objective - 2.05 Explain the advantages and configurations of high availability (HA)

2.05 – F5 High Availability concepts

Configuring High Availability

Single device

When you are running the BIG-IP system as a single device (as opposed to a unit of a redundant system), high availability refers to core services being up and running on that device, and VLANs being able to send and receive traffic.

Redundant devices

When you are running the BIG-IP system as a unit of a redundant system configuration, high availability refers to core system services being up and running on one of the two BIG-IP systems in the configuration. High availability also refers to a connection being available between the BIG-IP system and a pool of routers, and VLANs on the system being able to send and receive traffic.

A redundant system is a type of BIG-IP system configuration that allows traffic processing to continue in the event that a BIG-IP system becomes unavailable. A BIG-IP redundant system consists of two identically configured BIG-IP units. When an event occurs that prevents one of the BIG-IP units from processing network traffic, the peer unit in the redundant system immediately begins processing that traffic, and users experience no interruption in service.

What is failover?

Failover is a process that occurs when one system in a redundant system becomes unavailable, thereby causing the peer unit to assume the processing of traffic originally targeted for the unavailable unit. To facilitate coordination of the failover process, each unit has a unit ID (1 or 2).

An essential element to making failover successful is a feature called configuration synchronization. Configuration synchronization, or ConfigSync, is a process where you replicate one unit's main configuration file on the peer unit. Because data is shared in this way, a unit can process the other unit's traffic when failover occurs.

By default, the way that a BIG-IP unit monitors the status of its peer, in order to detect that failover is required, is through a hard-wired connection between the two BIG-IP units. With proper configuration, however, you can cause each BIG-IP unit to monitor peer status by way of a TCP/IP network connection instead.

2.05 - Explain active/active

Considerations When Configuring BIG-IP Systems in Active-Active Configuration for High Availability Deployments

Understanding active-active redundancy

Network device failures may occur for a wide variety of reasons, resulting in unexpected interruptions to applications and/or services. These unexpected outages may reduce customer satisfaction and confidence, and as a result, may incur loss of revenue vital for most organizations that conduct their businesses online. From this perspective, avoiding outages is critical to these organizations and the first step to ensure application and service availability is to deploy devices in an HA configuration. High availability ensures that applications fail over seamlessly and service continues uninterrupted. You can also perform troubleshooting while the application or service is still functioning.

You can deploy BIG-IP systems in an Active-Active configuration for an HA deployment. Deploying BIG-IP systems in such a manner allows the active devices in the same cluster to process traffic separately during normal operations and the ability to failover to one another when required. However, F5 does not recommend deploying BIG-IP systems in an Active-Active configuration without a standby device in the cluster, due to the potential loss of high availability. Consider the following points:

If each BIG-IP system in the Active-Active configuration can process the application or services adequately and with some additional resources in reserve, a failover from one active device to another active device should be seamless, which allows applications and/or services to continue uninterrupted.

If each BIG-IP system in the Active-Active configuration is running at half or greater capacity, a failover from one active device to another active device may cause the device that is taking over to reach its processing threshold. As a result, the device fails and interrupts the applications and/or services. This condition could continue until both devices are restored to their operational capability.

Starting in BIG-IP 11.0.0, the Device Service Clustering (DSC) feature allows you to configure more than two BIG-IP systems in an HA configuration. The DSC feature also allows you to configure multiple devices as active and one or more devices as standby.

2.05 - Explain active/standby

Creating an Active-Standby Configuration Using the Setup Utility

Understanding active-standby redundancy

An active-standby pair is a pair of BIG-IP devices configured so that one device is actively processing traffic while the other device remains ready to take over if failover occurs. The two devices synchronize their configuration data and can fail over to one another in the event that one of the devices becomes unavailable.

First, you can run the Setup utility on each device to configure base network components (that is, a management port, administrative passwords, and the default VLANs and their associated self IP addresses). Continue running it on each device to establish a trust relationship between the two devices, and create a Sync-Failover type of device group that contains two member devices.

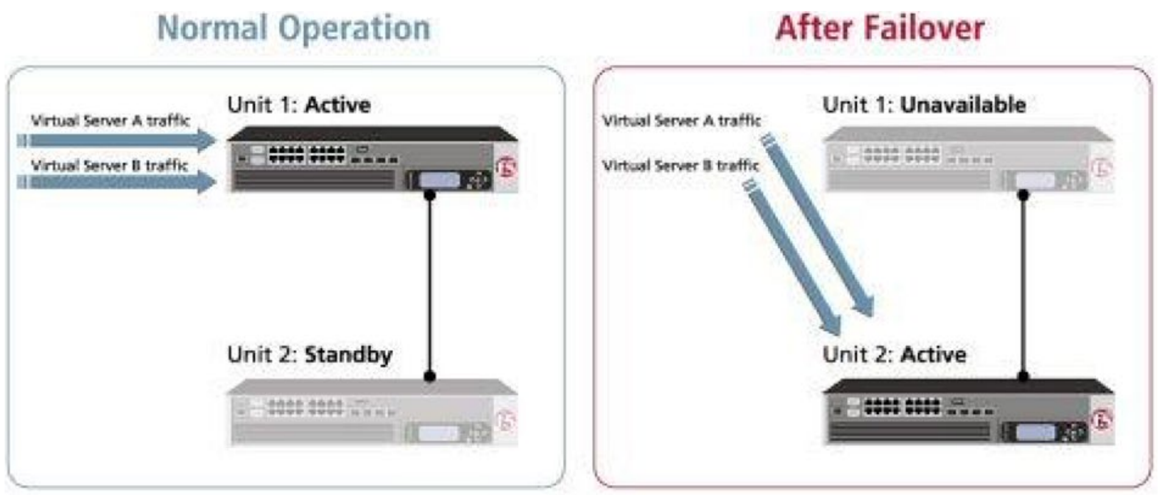
After the Setup utility is run on both devices, each device contains the default traffic group that the BIG-IP system automatically created during setup. A traffic group represents a set of configuration objects (such as floating self IP addresses and virtual IP addresses) that process application traffic. This traffic group actively processes traffic on one of the two devices, making that device the active device. When failover occurs, the traffic group will become active on (that is, float to) the peer BIG-IP device.

By default, the traffic group contains the floating self IP addresses of the default VLANs. Whenever you create additional configuration objects such as self IP addresses, virtual IP addresses, and SNATs, the system automatically adds these objects to the default traffic group.

Understanding failover in active/standby mode

When a redundant system is in active/standby mode, one unit is active, that is, accepting and processing connections on behalf of the redundant system, while the other unit is idle (that is, in a standby state). When failover occurs, the standby unit becomes active, and it normally stays active until failover occurs again, or until you force it into a standby state. Forcing the unit into a standby state automatically causes the other system to become active again, if possible.

For example, you can configure unit 1 to process traffic for virtual servers A and B. The standby unit monitors the active unit, and if communications fail, the standby unit initiates a failover and becomes the active unit. The newly-active unit then begins processing traffic for both virtual servers. You can see an active/standby configuration, first as it behaves normally, and then after failover has occurred, by viewing the figure.



As you can see in the figure, unit 1 is in an active state, and unit 2 is in a standby state. With this configuration, failover causes the following to occur:

- Unit 2 switches to an active state.
- Unit 2 begins processing the connections that would normally be processed by its peer.

When the failed unit becomes available again, you can force a unit to change its state from active to standby or from standby to active, thereby initiating failback. Failback on an active/standby system causes a unit to relinquish any processing that it is doing on behalf of its peer, and return to a standby state. A redundant system in active/standby mode is the most common type of redundant system.

SECTION 3 – LOAD BALANCING ESSENTIALS

Objective - 3.01 Discuss the purpose of, use cases for, and key considerations related to load balancing

3.01 – Explain the purpose of distribution of load across multiple servers

Distribution of Load

The amount of connections and utilization that an application may have coming in from its core base of users can often far exceed the throughput capacity of a single server hosting the application. The need for distributing the inbound requests and processing load of responses across a group of servers running the same application or service is self-evident. The trick to distributing the load is finding the right load balancing solution and then implementing it in the most effective manner. While making sure the functions of the applications are not being broken as users are sent all across the servers in the group.

3.01 – Given an environment, determine the appropriate load balancing algorithm that achieves a desired result

Pools

Local Traffic Manager load balancing methods

There are several load balancing methods available within the BIG-IP system for load balancing traffic to pool members.

Method	Description	When to use
Round Robin	This is the default load balancing method. Round Robin mode passes each new connection request to the next server in line, eventually distributing connections evenly across the array of machines being load balanced.	Round Robin mode works well in most configurations, especially if the equipment that you are load balancing is roughly equal in processing speed and memory.

Method	Description	When to use
Ratio (member) Ratio (node)	Local Traffic Manager distributes connections among pool members or nodes in a static rotation according to ratio weights that you define. In this case, the number of connections that each system receives over time is proportionate to the ratio weight you defined for each pool member or node. You set a ratio weight when you create each pool member or node.	These are static load balancing methods, basing distribution on user-specified ratio weights that are proportional to the capacity of the servers.
Dynamic Ratio (member) Dynamic Ratio (node)	The Dynamic Ratio methods select a server based on various aspects of real-time server performance analysis. These methods are similar to the Ratio methods, except that with Dynamic Ratio methods, the ratio weights are system-generated, and the values of the ratio weights are not static. These methods are based on continuous monitoring of the servers, and the ratio weights are therefore continually changing. **Note:** To implement Dynamic Ratio load balancing, you must first install and configure the necessary server software for these systems, and then install the appropriate performance monitor.	The Dynamic Ratio methods are used specifically for load balancing traffic to RealNetworks RealSystem Server platforms, Windows platforms equipped with Windows Management Instrumentation (WMI), or any server equipped with an SNMP agent such as the UC Davis SNMP agent or Windows 2000 Server SNMP agent.
Fastest (node) Fastest (application)	The Fastest methods select a server based on the least number of current sessions. These methods require that you assign both a Layer 7 and a TCP type of profile to the virtual server. **Note:** If the OneConnect feature is enabled, the Least Connections methods do not include idle connections in the calculations when selecting a pool member or node. The Least Connections methods use only active connections in their calculations.	The Fastest methods are useful in environments where nodes are distributed across separate logical networks.

Method	Description	When to use
Least Connections (member) Least Connections (node)	The Least Connections methods are relatively simple in that Local Traffic Manager passes a new connection to the pool member or node that has the least number of active connections. **Note:** If the OneConnect feature is enabled, the Least Connections methods do not include idle connections in the calculations when selecting a pool member or node. The Least Connections methods use only active connections in their calculations.	The Least Connections methods function best in environments where the servers have similar capabilities. Otherwise, some amount of latency can occur. For example, consider the case where a pool has two servers of differing capacities, A and B. Server A has 95 active connections with a connection limit of 100, while server B has 96 active connections with a much larger connection limit of 500. In this case, the Least Connections method selects server A, the server with the lowest number of active connections, even though the server is close to reaching capacity. If you have servers with varying capacities, consider using the Weighted Least Connections methods instead.

Method	Description	When to use
Weighted Least Connections (member) Weighted Least Connections (node)	Like the Least Connections methods, these load balancing methods select pool members or nodes based on the number of active connections. However, the Weighted Least Connections methods also base their selections on server capacity. The Weighted Least Connections (member) method specifies that the system uses the value you specify in Connection Limit to establish a proportional algorithm for each pool member. The system bases the load balancing decision on that proportion and the number of current connections to that pool member. For example, member_a has 20 connections and its connection limit is 100, so it is at 20% of capacity. Similarly, member_b has 20 connections and its connection limit is 200, so it is at 10% of capacity. In this case, the system select selects member_b. This algorithm requires all pool members to have a non-zero connection limit specified. The Weighted Least Connections (node) method specifies that the system uses the value you specify in the node's Connection Limit setting and the number of current connections to a node to establish a proportional algorithm. This algorithm requires all nodes used by pool members to have a non-zero connection limit specified. If all servers have equal capacity, these load balancing methods behave in the same way as the Least Connections methods. **Note:** If the OneConnect feature is enabled, the Weighted Least Connections methods do not include idle connections in the calculations when selecting a pool member or node. The Weighted Least Connections methods use only active connections in their calculations.	Weighted Least Connections methods work best in environments where the servers have differing capacities. For example, if two servers have the same number of active connections but one server has more capacity than the other, Local Traffic Manager calculates the percentage of capacity being used on each server and uses that percentage in its calculations.
Observed (member) Observed (node)	With the Observed methods, nodes are ranked based on the number of connections. The Observed methods track the number of Layer 4 connections to each node over time and create a ratio for load balancing.	The need for the Observed methods is rare, and they are not recommended for large pools.

Method	Description	When to use
Predictive (member) Predictive (node)	The Predictive methods use the ranking methods used by the Observed methods, where servers are rated according to the number of current connections. However, with the Predictive methods, Local Traffic Manager analyzes the trend of the ranking over time, determining whether a node's performance is currently improving or declining. The servers with performance rankings that are currently improving, rather than declining, receive a higher proportion of the connections.	The need for the Predictive methods is rare, and they are not recommend for large pools.
Least Sessions	The Least Sessions method selects the server that currently has the least number of entries in the persistence table. Use of this load balancing method requires that the virtual server reference a type of profile that tracks persistence connections, such as the Source Address Affinity or Universal profile type. **Note:** The Least Sessions methods are incompatible with cookie persistence.	The Least Sessions method works best in environments where the servers or other equipment that you are load balancing have similar capabilities.
Ratio Least Connections	The Ratio Least Connections methods cause the system to select the pool member according to the ratio of the number of connections that each pool member has active.	

3.01 - Explain the concept of persistence

Persistent and Persistence, What's the Difference?

Persistent and Persistence, What's the Difference?

While the conceptual basis of persistence and persistent are essentially the same, in reality they refer to two different technical concepts.

Both persistent and persistence relate to the handling of connections. The former is often used as a general description of the behavior of HTTP and, necessarily, TCP connections, though it is also used in the context of database connections. The latter is most often related to TCP/HTTP connection handling but almost exclusively in the context of load balancing.

Persistent

Persistent connections are connections that are kept open and reused. The most commonly implemented form of persistent connections is HTTP, with database connections a close second.

Persistent HTTP connections were implemented as part of the HTTP 1.1 specification as a method of improving the efficiency of HTTP in general. Before HTTP 1.1 a browser would generally open one connection per object on a page in order to retrieve all the appropriate resources. As the number of objects in a page grew, this became increasingly inefficient and significantly reduced the capacity of web servers while causing browsers to appear slow to retrieve data. HTTP 1.1 and the Keep-Alive header in HTTP 1.0 were aimed at improving the performance of HTTP by reusing TCP connections to retrieve objects. They made the connections persistent such that they could be reused to send multiple HTTP requests using the same TCP connection.

Similarly, this notion was implemented by proxy-based load-balancers as a way to improve performance of web applications and increase capacity on web servers. Persistent connections between a load-balancer and web servers is usually referred to as TCP multiplexing. Just like browsers, the load-balancer opens a few TCP connections to the servers and then reuses them to send multiple HTTP requests.

Persistent connections, both in browsers and load-balancers, have several advantages:

- Less network traffic due to less TCP setup/teardown. It requires no less than 7 exchanges of data to set up and tear down a TCP connection, thus each connection that can be reused reduces the number of exchanges required resulting in less traffic.

- Improved performance. Because subsequent requests do not need to setup and tear down a TCP connection, requests arrive faster and responses are returned quicker. TCP has built-in mechanisms, for example window sizing, to address network congestion. Persistent connections give TCP the time to adjust itself appropriately to current network conditions, thus improving overall performance. Non-persistent connections are not able to adjust because they are open and almost immediately closed.

- Less server overhead. Servers are able to increase the number of concurrent users served because each user requires fewer connections through which to complete requests.

Persistence

Persistence, on the other hand, is related to the ability of a load-balancer or other traffic management solution to maintain a virtual connection between a client and a specific server.

Persistence was often referred to in the application delivery networking world as “stickiness” while in the web and application server demesne it is called “server affinity”. Persistence ensures that once a client has made a connection to a specific server that subsequent requests are sent to the same server. This is very important to maintain state and session-specific information in some application architectures and for handling of SSL- enabled applications. When the first request is seen by the load-balancer it chooses a server. On subsequent requests the load balancer will automatically choose the same server to ensure continuity of the application or, in the case of SSL, to avoid the compute intensive process of renegotiation. This persistence is often implemented using cookies but can be based on other identifying attributes such as IP address. Load-balancers that have evolved into application delivery controllers are capable of implementing persistence based on any piece of data in the application message (payload), headers, or at in the transport protocol (TCP) and network protocol (IP) layers.

Some advantages of persistence are:

- Avoid renegotiation of SSL. By ensuring that SSL enabled connections are directed to the same server throughout a session, it is possible to avoid renegotiating the keys associated with the session, which is compute and resource intensive. This improves performance and reduces overhead on servers.
- No need to rewrite applications. Applications developed without load balancing in mind may break when deployed in a load-balanced architecture because they depend on session data that is stored only on the original server on which the session was initiated. Load-balancers capable of session persistence ensure that those applications do not break by always directing requests to the same server, preserving the session data without requiring that applications be rewritten.

Summary

So persistent connections are connections that are kept open so they can be reused to send multiple requests, while persistence is the process of ensuring that connections and subsequent requests are sent to the same server through a load-balancer or other proxy device.

Both are important facets of communication between clients, servers, and mediators like load-balancers, and increase the overall performance and efficiency of the infrastructure as well as improving the end-user experience.

Objective - 3.02 Differentiate between a client and server

3.02 - Given a scenario, identify the client/server

Difference Between Client and Server Systems

Differences between Server and Client

Computers are needed in businesses of different sizes. Large computer setups that include networks and mainframes are used in large businesses. A computer network used in these types of businesses has a client- server architecture or two-tier architecture. The main purpose of this architecture is the division of labor, which is required in large organizations.

3.02 - Explain the role of a client

Client

In client- server architecture, the client acts a smaller computer that is used by the employees of the organization in order to perform their day-to-day activities. The employee uses the client computer in order to access the data files or applications stored on the server machine.

The rights authorized to the client machine can be different. Some employees have the access to data files of the organization while other may only access the applications present on the server.

Apart from using the applications and data files, the client machine can also utilize the processing power of the server. In this case, the client computer is plugged-in to the server and the server machine handles all the calculations. In this way, the large processing power of the server can be utilized without any addition of hardware on the client side.

The best example of client- server architecture is WWW or World Wide Web. Here the client is the browser installed on each computer and the information about different pages is stored on the server side from which the client or the user can access it.

3.02 - Explain the role of a server

Server

In client-server environment, the server computer acts as the “brains” of the business. A very large capacity computer is used as a server. There can be a mainframe also as it stores a wide variety of functionalities and data.

Generally, applications and data files are stored on the server computer. Employee computers or workstations access these applications and files across the network. For example, an employee can access company’s data files stored on the server, from his/her client computer.

Difference between client and server

- Client is a smaller computer through which the user accesses the information or application stored on the server whereas server is a powerful computer that stores the data files and applications.
- In some cases, the client may utilize the greater processing power of the server machine.
- In some cases, the client side may have a better graphical user interface or GUI as compared to the server side.

In some cases, employees may access only specific applications from their client machine. Application server is the name given to this type of server. The client-server architecture is fully utilized in this type of environment as employees have to login from their client machine in order to access the application stored on the server. For example, these kinds of applications include graphic design programs, spreadsheets and word processors. The client- server architecture is illustrated in each case.

Apart from the storage medium, the server also acts as a processing power source. The client machines get their processing power from this server source. By doing so, no extra hardware for the client is needed and it utilizes greater processing power of the server.

SECTION 4 – SECURITY

Objective - 4.01 Compare and contrast positive and negative security models

4.01 - Describe the concept of a positive security model

Positive Security Model

Positive Security Model

A “positive” security model (also known as “whitelist”) is one that defines what is allowed, and rejects everything else.

The positive security model can be applied to a number of different application security areas. For example, when performing input validation, the positive model dictates that you should specify the characteristics of input that will be allowed, as opposed to trying to filter out bad input. In the access control area, the positive model is to deny access to everything, and only allow access to specific authorized resources or functions. If you’ve ever had to deal with a network firewall, then you’ve probably encountered this application of the positive model.

4.01 - Describe the concept of a negative security model

Signature & Negative Security

Negative Security Model

A “negative” (or “blacklist”) security model is one that defines what is disallowed, while implicitly allowing everything else.

A negative security model (or misuse based detection) is based on a set of rules that detect attacks rather than allow only valid traffic.

A negative security model is very common to Intrusion Detection and Prevention systems (IDPS). Therefore it is very important to understand what the differences are between the negative security model provided by an IDPS and the negative security model provided by a WAF.

4.01 - Describe the benefits of a positive security model

Benefits of a Positive Security Model

The benefit of using a positive model is that 0-day attacks, will be prevented as well as developer related shortcomings. However, the positive model is susceptible to false positives in that if you don't account for everything that the application needs to function in the policy you will block it. Also if the application changes you will need to build a new custom policy to match the new changes.

4.01 - Describe the benefits of a negative security model

Benefits of a Negative Security Model

The benefit of a negative security model is that it can be deployed rapidly. You do not have to worry with building a complicated policy that has to deal with every transactional function of the application. You are simply blocking any known bad attacks that could happen. You are vulnerable to 0-day attacks of which the pattern of the attack is not already known.

4.01 – Security (Non-Blueprint)

Applied Application Security - Positive and Negative Efficiency

Positive & Negative Security model

After many years of purely negative security provided by anti-virus scanners, IDS/IPS, and antispam engines, it's refreshing to hear that the positive security model—the basis for tried and true security devices like network firewalls and ACLs—is coming back in vogue. Most recently, this positive policy re-emergence has revolved around the Web Application Firewall (WAF) and application security market. Yet with the positive security positioning comeback carries with it a very interesting point of detail: although many in the WAF space argue that the positive model is preferable, nearly all application security providers still rely on a partially negative solution. While acknowledging that a positive security model is the preferable model to secure web applications, many practitioners and vendors advocate a bilateral approach of both positive and negative security.

As the application security market continues to evolve and define itself, there continues to be diverging views on which security methodology is the best option. In reality, enterprise security decisions are highly dependent on many factors, most of which are more business than technology oriented. Implementing an application security solution that is both secure and practical—while still allowing for the fluid nature of protecting dynamic applications—requires taking the best pieces of technology and business analysis and synthesizing them into an effective and efficient security solution.

What Does "Good" Security Cost?

In theory, the best security is impenetrable, but practical security does not function as a control group. In a business environment, security is a multivariate problem. What is the performance of the security? How easy is it to deploy? What impact will adding security have on the cost per transaction? Is it more expensive to build an impenetrable security system or risk covering the cost of a public breach? The quality of the security is always questioned as well, but it's never the only question. Many security-related questions come from the balance sheets, not the security engineers. Approaching security from a technical standpoint alone does not help the business; it hurts it.

Businesses constantly analyze their economic model to generate better operational efficiencies and a greater return on investment; the entire business intelligence market exists for this purpose. The driving force behind any IT security decision is an evaluation of a situation's potential risks versus the investment necessary to circumvent these risks. In the same vein, a business' security efforts should address a business problem; namely, to increase operational efficiencies. Security breaches can mar this efficiency, hurting a company's value, either in real dollars, operational downtime, or loss of customer trust. In truth, the motivation behind every IT decision (including security decisions) is a business decision. This has been stated and fully advocated by Gartner as well:

Jay Heiser, a Gartner vice president, said the fundamental problem with a purely technical approach is that IT security professionals have no understanding of business. Speaking at [the] Gartner IT Security Summit in London, Heiser said businesses must now mature and appoint individuals who understand the complexities of business, rather than the simplicities of security.

When IT decisions become business decisions, blurring the distinction between secure value and business value, theoretical security and applied, or practical, security begin to separate. The theoretical approach places security on a singular plane, untouched by other business factors; applied security is comprised of the measure and level of actual security safeguards and implementations needed to accomplish business goals.

For a security product to be a functional part of a business' IT infrastructure, the product's applied security must be given more attention than the product's theoretical security. A solution that only strives to provide ultimate security will almost always be replaced with a solution designed to apply "good enough" security and increase business efficiencies than one intending to recreate Fort Knox. Theoretical security is a checkbox criteria, applied security becomes more of a buy-and-use criteria. Applied security exists at an equilibrium point between total security (theoretical security) and total functionality (no security). The choice between these two—and what to sacrifice—is made based on the most operationally efficient method for achieving that prescribed balance. To better evaluate the root ROI question when dealing with security products, the next logical question becomes "Which model, positive or negative, provides this equilibrium in the most operationally efficient manner?

Positive vs. Negative Application Security

The two approaches to security most often mentioned in the context of application security—positive and negative are diametrically opposed in all of their characteristic behaviors, but they are structured very similarly. Both positive and negative security approaches operate according to an established set of rules. Access Control Lists (ACLs) and signatures are two implementation examples of positive and negative security rules, respectively.

Positive security moves away from "blocked," end of the spectrum, following an "allow only what I know" methodology. Every rule added to a positive security model increases what is classified as known behavior, and thus allowed, and decreases what is blocked, or what is unknown. Therefore, a positive security model with nothing defined should block everything and relax (i.e., allow broader access) as the acceptable content contexts are defined.

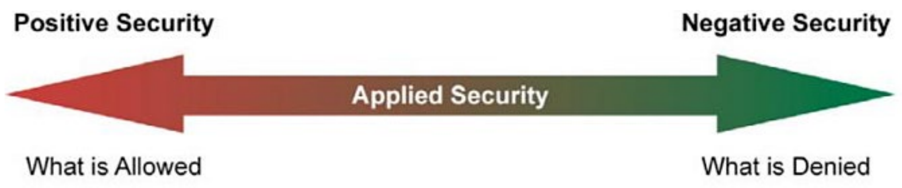
At the opposite end of the spectrum, negative security moves towards "blocked what I know is bad," meaning it denies access based on what has previously identified as content to be blocked, running opposite to the known/allowed positive model. Every rule added to the negative security policy increases the blocking behavior, thereby decreasing what is both unknown and allowed as the policy is tightened. Therefore, a negative security policy with nothing defined would grant access to everything, and be tightened as exploits are discovered. Although negative security does retain some aspect of known data, negative security knowledge comes from a list of very specific repositories of matching patterns. As data is passed through a negative security policy, it is evaluated against individual known "bad" patterns. If a known pattern is matched, the data is rejected; if the data flowing through the policy is unidentifiable, it is allowed to pass. Negative security policies do not take into account how the application works, they only notice what accesses the application and if that access violates any negative security patterns.

Discussions on preferred security methods typically spawn very polarized debates. Tried and true security engineers might ardently argue the merits of the positive security model because it originates from the most "secure" place—"Only allow what I know and expect." Many business pundits would argue that the negative model is the best as it starts in the most "functional" place—

"Block what I know is bad and let everything unknown through." Both groups are correct and yet both opinions become irrelevant when projected onto applied security, because both positive and negative security is theoretical. Applied security falls somewhere in the middle of the spectrum, providing a practical balance. At some point, as the negative approach is tightened, it will take on characteristics of a more positive model, inching towards a more complete security approach.

Likewise, as a positive security model is loosened to accommodate new application behaviors, it will take on some aspects of a more negative approach, such as implementing data pattern matching, to block the more predictable attacks. As a positive policy continues to relax, it will move closer towards complete functionality.

The point at which these two opposing concepts begin to overlap is where applied security starts to take shape.



This "meet in the middle" idea suggests that from an applied security standpoint, both models are capable of achieving the same delicate balance between "security" and "functionality." The difference between these models stems from where each begins and where they collide. This can be as simple as the number of rules required to meet the end goal.

It is clear then that, from an operational efficiency standpoint, the undiluted concepts of neither the positive nor the negative approach intrinsically provides more efficiency than the other. In some cases, the positive approach generates the least number of rules while in other cases the negative approach generates the least. It would also appear that it is the nature of the applied policy and/or the content itself, which might determine the best approach. What then, are the qualities of the policy or content which makes one approach more efficient over the other?

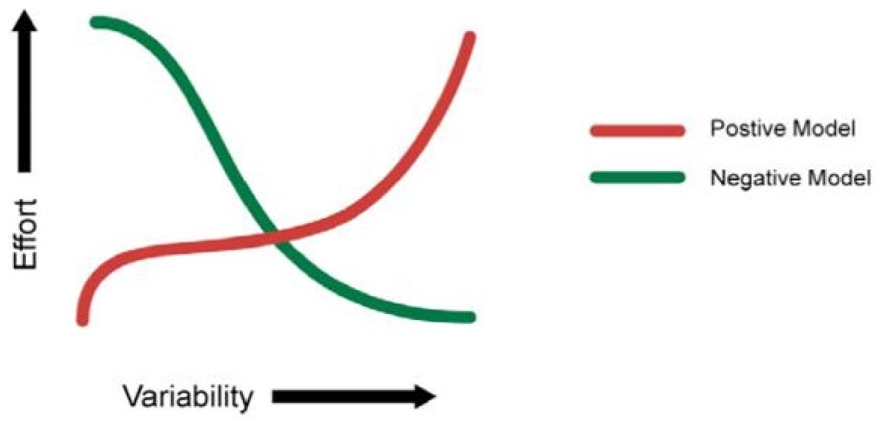
Factors of an Effective Applied Security Model

Implementing successful application security architecture is not as easy as deciding how much negative security and how much positive security to mix together into a hypothetical applied security blender. By design, application security devices have to have some level of application knowledge, such as the type of content delivered by the application, which is accessing any point within the application, and how to map specific policy criteria to this information. Very specific application awareness of this nature is essential in building an efficient applied security policy.

The Effect of Content Variability

Within the scope of application security, Content Variability is a measure of the content that needs to be secured and includes a number of different component pieces: the number of objects, the number of types of content, frequency of content change, and the nature of the content. A site that only has five specific objects is much less variable than a site with 500 specific objects. Within those objects, the cohesiveness of the content type is also a factor; if all 500 objects share a common format, they are less variable than a site with where all 500 objects are unique. Obviously, a site that changes only once a year is much less variable than one that changes daily. Finally, the nature of that content—for example, whether it is dynamically generated or static is a contributing factor. Essentially, variability is a measure of the site complexity. The idea of Content

Variability is a single measurable value based on all of these factors. The variability of the content dictates the amount of effort needed to achieve the prescribed applied security from the chosen model.



As depicted in the diagram, the higher the variability of the content, the easier it is to define a policy using the negative security model. As the complexity of the known content increases, it is easier to describe what isn't allowed rather than what is. Conversely, the opposite effect is true of the positive model; the more variable the site content, more effort is required to define those elements that are allowed. For example, let us assume that we have 10 different types of content within our site out of a possible 100 different types of content known. Because the site exhibits little variability, or is more cohesive, it is much easier to define the 10 allowable types of content than to define the 90 types of restricted content; a positive model is much more appropriate in this case.

On the other hand, if the site is less cohesive, perhaps representing of 90 of the 100 different types, it now becomes more efficient to define the 10 restricted content types than it is to define the 90 allowed ones; thus a negative model is more efficient. Once again, both models are equally successful at producing a desired level of security, but the variability of the content determines which is more efficient in a given scenario. And as we map the concept of content variability back to applied security, it becomes obvious that we will take the necessary aspects from the negative security model and couple those with what is required from the positive model.

The most successful implementation will come from a joint applied security policy, addressing both the security and the business needs at same time.

Rule Specificity

As the content variability affects the ability to create and maintain a security policy, the same is true of the specificity of rules used to build that policy. Rule Specificity conveys the level of detail of the protection mechanism implemented for any particular rule. For example, a rule that blocks Unicode attacks may block them from any application on one end of the spectrum all the way to only protecting Unicode directory traversal attacks against IIS5 on the other end. Depending upon the specificity of a rule, many things may be allowed with a single rule (positive security) or disallowed with a single rule (negative security). But as is the problem with theoretical security, Rule Specificity itself is not an exact science.

A rule that is not specific enough may block too much, creating unnecessary false positives (blocking access that shouldn't be blocked); a rule that is too specific may not block enough, creating false negatives. Content variability also impacts the efficiency of a policy by altering the level of specificity in the rules themselves. As the variability of the content increases, the ability to specifically stipulate what content is or isn't allowed becomes more time consuming. In an ideal world, every rule would be as specific as possible for the particular application it was designed to protect, avoiding false positives and false negatives. Similarly, the level of rule specificity within an application security policy can vary greatly depending on the content variability experienced by the application.

Order of Precedence

A third factor in implementing an efficient applied security policy is the order of precedence: defining which parts of the security policy are enacted before other parts of the policy. This concept is often seen in programmatic search algorithms: "match first" or "match any." Using a combination of negative patterns and positive policy rules with varying degrees of specificity is bound to create many conflicts. In order to arbitrate these conflicts an order of precedence for all rules must be defined and followed for the policy to remain coherent. This is a critical decision point for application security, because the policy must decide if it should implement a more funneled approach (parsing through the policy to weed out what doesn't match) or if it should look for the most restrictive implementation first. Choosing the most specific rule may solve this order of precedence, whether it is positive or negative, and opening up access as data moves further through the policy.

Alternately, the order may be based on implementing a given rule set, for example, all traffic may be pattern matched first and if there are any positive matches, the data is rejected, regardless of which specific pattern was matched. No matter which method is chosen, if the policy is implemented with an incorrect order of precedence, access to the application could be blocked by a policy that tightens first. Likewise, a policy that applies rules too loosely may allow unintended access to the application.

And as precedence is factored into the applied security equation, traffic volumes must also be taken into consideration. A two percent false positive error rate may be an acceptable metric in an applied security policy of an application that handles 100 connections/day, but unacceptable for a 10 million connection/day application. Regardless of the precedence methodology used it should be well defined and easy to follow to make a policy easy to audit and manage.

Conclusion - Best Practices

The problem with a purely positive policy is simply that it's merely the most appropriate model for about half of the situations in which it's deployed. The other half are unnecessarily weighed down by the fact that a negative model would be much more efficient. That is why, as a matter of best practice, every security solution should support a weighted balance of both the positive and negative methodologies. In the strictest sense of the term, negative security provides the best applied security out of the box due to the effort applied by the security vendor before the product is shipped. Focusing on known security vulnerabilities, this will block the most attacks, despite content variability. However, this does not provide security against unknown attacks or allow specific functions to be allowed. For that, positive security is required. To lessen the amount of effort needed for a given application, positive security templates should be provided by the application vendors themselves to complement the negative security.

If the goal of applied security is to reach a pre-defined posture in the most efficient manner, then the choice of model is directly related to the variability of the content itself. Somewhere between total security and total functionality is where the desired applied security level exists, and—theoretically—either security model is capable of achieving this goal. But as stated above, theoretical security can only exist in a vacuum. Applied security is a business choice and concept that moves security into real-world implementations to attain the most efficient, functional method. Neither positive nor negative security models alone can deliver the most economical solution in every situation or environment. Applied together, however—and merged with the business needs and requirements—a holistic view of both approaches can help delineate between theoretical security and applied security, enabling businesses to realize the greatest ROI from any security policy implementation.

Objective - 4.02 Explain the purpose of cryptographic services

4.02 - Describe the purpose of signing

Digital Signature

Purpose of Signing

A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, that the sender cannot deny having sent the message (authentication and non-repudiation), and that the message was not altered in transit (integrity).

Digital signatures employ asymmetric cryptography. In many instances they provide a layer of validation and security to messages sent through a non-secure channel: Properly implemented, a digital signature gives the receiver reason to believe the message was sent by the claimed sender. Digital seals and signatures are equivalent to handwritten signatures and stamped seals. Digital signatures are equivalent to traditional handwritten signatures in many respects, but properly implemented digital signatures are more difficult to forge than the handwritten type. Digital signature schemes, in the sense used here, are cryptographically based, and must be implemented properly to be effective. Digital signatures can also provide non-repudiation, meaning that the signer cannot successfully claim they did not sign a message, while also claiming their private key remains secret; further, some non-repudiation schemes offer a time stamp for the digital signature, so that even if the private key is exposed, the signature is valid. Digitally signed messages may be anything representable as a bitstring: examples include electronic mail, contracts, or a message sent via some other cryptographic protocol.

4.02 - Describe the purpose of encryption

The Purpose of Cryptography

Topic

In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet.

Within the context of any application-to-application communication, there are some specific security requirements, including:

- Authentication: The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)
- Privacy/confidentiality: Ensuring that no one can read the message except the intended receiver.
- Integrity: Assuring the receiver that the received message has not been altered in any way from the original.
- Non-repudiation: A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into ciphertext, which will in turn (usually) be decrypted into usable plaintext.

4.02 - Describe the purpose of certificates and the certificate chains

Chain Certificates

Certificates and Certificate Chains

It all starts with something called a root certificate. The root certificate is generated by a certification authority (CA) and is embedded into software applications. You will find root certificates in Microsoft Windows, Mozilla Firefox, Mac OS X, Adobe Reader, etc. The purpose of the root certificate is to establish a digital chain of trust. The root is the trust anchor.

The presumption is that the application developer has pre-screened the CA, ensured it meets a minimum level of trust and has accepted the CA's root certificate for use. Many application developers, including Adobe, Apple, Mozilla, Microsoft, Opera and Oracle, have root certificate programs. Others rely on the roots provided by the underlying operating system or developer toolkit.

One of the main functions of the root is to issue chain certificates to issuing CAs who are the first link in the chain of trust. Your Web browser will inherently trust all certificates that have been signed by any root that has been embedded in the browser itself or in an operating system on which it relies.

Why do you need an issuing CA? The purpose of the issuing CA is to isolate certificate policy from the root. Issuing CAs can be used to issue many different certificate types: SSL, EV SSL, Code Signing, Secure Email, Adobe CDS, etc. These certificate types are subjected to different requirements and risks, and as such have different certificate policies. The certificates may have different assurance levels such as high, medium and low. Issuing CAs may also be controlled by an organization other than that which controls the root.

The last link of trust is that between the end entity certificate and the issuing CA. In the case of an SSL certificate, the end entity certificate represents the linkage between a website owner and the website domain name. The SSL certificate is installed on the Web server along with the chain certificate. When a user browses to the website protected by the SSL certificate, the browser initiates the verification of the certificate and follows the chain of trust back to the embedded root.

In some cases, the CA may have chosen to issue end entity certificates directly from the root CA. This is an outdated practice; issuing directly from the root increases risk and limits how certificate policy can be managed and enforced.

4.02 - Distinguish between private/public keys

Encryption

Private Key Encryption

Private key encryption is the standard form. Both parties share an encryption key, and the encryption key is also the one used to decrypt the message. The difficulty is sharing the key before you start encrypting the message - how do you safely transmit it?

Many private key encryption methods use public key encryption to transmit the private key for each data transfer session.

If Bob and Alice want to use private key encryption to share a secret message, they would each use a copy of the same key. Bob writes his message to Alice and uses their shared private key to encrypt the message. The message is then sent to Alice. Alice uses her copy of the private key to decrypt the message. Private key encryption is like making copies of a key. Anyone with a copy can open the lock. In the case of Bob and Alice, their keys would be guarded closely because they can both encrypt and decrypt messages.

Public Key Encryption

Public key encryption uses two keys - one to encrypt, and one to decrypt. The sender asks the receiver for the encryption key, encrypts the message, and sends the encrypted message to the receiver. Only the receiver can then decrypt the message - even the sender cannot read the encrypted message.

When Bob wants to share a secret with Alice using public key encryption, he first asks Alice for her public key. Next, Bob uses Alice's public key to encrypt the message. In public key encryption, only Alice's private key can unlock the message encrypted with her public key. Bob sends his message to Alice. Alice uses her private key to decrypt Bob's message.

The thing that makes public key encryption work is that Alice very closely guards her private key and freely distributes her public key. She knows that it will unlock any message encrypted with her public key.

4.02 - Compare and contrast symmetric/asymmetric encryption

Symmetric Encryption

This system uses only private keys, which can be anything from a numerical symbol to a string of random letters. These private keys are used to encode a message, so that only the sender and the recipient of the message who know what the secret key is can "unlock" it and decrypt it. The system works pretty much like two best friends using a decoder ring to send secret messages to each other. The symmetric system's only downside is the potentially unsafe private key transmission via the Internet, where other people can “crack” it and decode the message.

Asymmetric Encryption

As a solution for the not completely safe Symmetric Encryption, there is the Asymmetric Encryption system that uses a pair of keys for added security: a private and a public key. The private key is for yourself and the public key is published online for others to see.

The public key is used to access the encryption code that corresponds to your private key. So, if you were sending an encrypted message to Susan, which you do not want others to see, you would use her public key to encrypt it. She will be able to decrypt it with her own corresponding private key. Likewise, if she sends a message to you, she uses your public key to encrypt the message and you would use your private key to decrypt it.

Objective - 4.03 Describe the purpose and advantages of authentication

4.03 - Explain the purpose of authentication

The Business of Authentication

What Is Authentication?

Authentication is the process of determining if a user or identity is who they claim to be. Authentication is accomplished using something the user knows (e.g. password), something the user has (e.g. security token) or something of the user (e.g. biometric).

The authentication process is based on a measure of risk. High risk systems, applications and information require different forms of authentication that more accurately confirm the user’s digital identity as being who they claim to be than would a low risk application, where the confirmation of the digital identity is not as important from a risk perspective. This is commonly referred to as “stronger authentication”.

Authentication processes are dependent upon identity verification and registration processes. For example, when Jane Doe is hired at an enterprise, she provides the enterprise with information and tokens of who she is (e.g. name, address, driver’s license, birth certificate, a SSN number, a passport, etc.). The enterprise may choose to immediately accept this information or, it may instead chose to run background checks on Jane to see if she is who she claims to be and determine if she has any criminal record. When the checks come back favorably, the enterprise will accept her identity and enter her into their systems. The identity registration process will usually involve issuing Jane with enterprise authentication mechanisms such as id and password, security token, digital certificate and/or registering some of her biometrics.

The authentication process is totally dependents on the identity validation and registration process used for Jane. If Jane presents false tokens, which are accepted by the enterprise, then the person acting as Jane will be positively authenticated every time, even though she is not the real Jane Doe. Authentication security therefore is only as good as the weakest link in the chain.

4.03 - Explain the advantages of single sign on

Password Authentication

Password authentication is the most common method of authentication. It is also the least secure. Password authentication requires the identity to input a user id and a password in order to login. Password length, type of characters used and password duration are password management is now critical concern in enterprises. The ability to easily crack passwords has resulted in high levels of identity theft. As a result, the high risk of passwords means most enterprises now deploy a layered security strategy. A user enters in their id and password for initial login to gain access to only low risk information and applications with other forms of authentication required for higher risk information and applications.

Single Sign On Authentication

Single Sign On (SSO), Reduced Sign On (RSO), or Enterprise Single Sign On (ESSO) is the ability to reduce the number of id's and passwords a user has to remember. In most enterprises, a strong business case can be made to implement single sign on by reducing the number of password related help desk calls. SSO is also the architecture to require stronger forms of authentication for higher risk information and applications. Thus a user may login using their id and password to gain general low risk access to an enterprise. The SSO software enables them to not have to use multiple IDs and passwords. However, when the user tries to access more sensitive information and applications, the single sign on software will require the identity to input stronger authentication such as a security token, a digital certificate and/or a biometric.

4.03 - Explain the concepts of multifactor authentication

Multifactor Authentication (MFA) Definition

Multi-factor Authentication

Multifactor authentication (MFA) is a security system in which more than one form of authentication is implemented to verify the legitimacy of a transaction. The goal of MFA is to create a layered defense and make it more difficult for an unauthorized person to access a computer system or network.

Multifactor authentication is achieved by combining two or three independent credentials: what the user knows (knowledge-based authentication), what the user has (security token or smart card) and what the user is (biometric verification). Single-factor authentication (SFA), in contrast, only requires knowledge the user possesses. Although password-based authentication is well suited for website or application access, it is not secure enough for online financial transactions.

4.03 - Describe the role authentication plays in AAA

Authentication, Authorization, and Accounting (AAA) Definition

Authentication, Authorization, and Accounting (AAA)

Authentication, authorization, and accounting (AAA) is a term for a framework for intelligently controlling access to computer resources, enforcing policies, auditing usage, and providing the information necessary to bill for services. These combined processes are considered important for effective network management and security.

As the first process, authentication provides a way of identifying a user, typically by having the user enter a valid user name and valid password before access is granted. The process of authentication is based on each user having a unique set of criteria for gaining access. The AAA server compares a user's authentication credentials with other user credentials stored in a database. If the credentials match, the user is granted access to the network. If the credentials are at variance, authentication fails and network access is denied.

Following authentication, a user must gain authorization for doing certain tasks. After logging into a system, for instance, the user may try to issue commands. The authorization process determines whether the user has the authority to issue such commands. Simply put, authorization is the process of enforcing policies: determining what types or qualities of activities, resources, or services a user is permitted. Usually, authorization occurs within the context of authentication. Once you have authenticated a user, they may be authorized for different types of access or activity.

The final plank in the AAA framework is accounting, which measures the resources a user consumes during access. This can include the amount of system time or the amount of data a user has sent and/or received during a session. Accounting is carried out by logging of session statistics and usage information and is used for authorization control, billing, trend analysis, resource utilization, and capacity planning activities.

Authentication, authorization, and accounting services are often provided by a dedicated AAA server, or a program that performs these functions. A current standard by which network access servers interface with the AAA server is the Remote Authentication Dial-In User Service (RADIUS).

4.03 - SAML Authentication Not on Blueprint

SAML Authentication

SAML Authentication - What is SAML?

SAML - Security Assertion Markup Language

SAML, developed by the Security Services Technical Committee of "Organization for the Advancement of Structured Information Standards" (OASIS), is an XML-based framework for exchanging user authentication, entitlement, and attribute information. SAML is a derivative of XML. The purpose of SAML is to enable Single Sign-On for web applications across various domains.

Why SAML?

There are four 'drivers' behind the creation of the SAML standard:

Limitations of Browser cookies: Most existing Single-Sign On products use browser cookies to maintain state so that re-authentication is not required. Browser cookies are not transferred between DNS domains. So, if you obtain a cookie from www.abc.com, then that cookie will not be sent in any HTTP messages to www.xyz.com. This could even apply within an organization that has separate DNS domains. Therefore, to solve the Cross-Domain SSO (CDSSO) problem requires the application of different technology. All SSO products solve the CDSSO problem by different techniques.

SSO Interoperability: How products implement SSO and CDSSO are completely proprietary. If you have an organization and you want to perform SSO across different DNS domains within the same organization or you want to perform CDSSO to trading partners, then you will have to use the same SSO product in all the domains.

Web Services: Security within Web Services is still being defined. Most of the focus has been on how to provide confidentiality and authentication/integrity services on an end-to-end basis. The SAML standard provides the means by which authentication and authorization assertions can exchanged between communicating parties.

Federation: The need to simplify identity management across organizational boundaries, allowing users to consolidate many local identities into a single (or at least a reduced set).

Objective - 4.04 Describe the purpose, advantages, and use cases of IPsec and SSL VPN

4.04 - Explain the purpose, advantages, and challenges associated with IPsec

IPsec

IPsec – IP Security

Internet Protocol Security (IPsec) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. IPsec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used in protecting data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host).

Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite, while some other Internet security systems in widespread use, such as Transport Layer Security (TLS) and Secure Shell (SSH), operate in the upper layers at Application layer. Hence, only IPsec protects any application traffic over an IP network. IPsec can automatically secure applications at the IP layer.point environments.

IPSec vs. SSL VPN: Transition Criteria and Methodology

Why should you use IPsec?

IPSec VPNs are best suited for point-to-point access. Open tunneling protects data between two private networks or between IT-managed machines and a private network. IPSec is a perfectly viable solution when a permanent connection is required between two specific locations, for example between a branch or remote office and a corporate headquarters. It can also be used successfully to provide access to a small finite number of remote workers using tightly controlled corporate-issued laptops.

Many existing IPSec implementations can continue to work well for these use cases for which they were originally deployed. IT might consider keeping IPSec in these limited areas and extend remote access to other areas, such as trusted partners or extranet users, via a parallel SSL VPN solution. While a parallel VPN implementation is a viable choice for some enterprises, transitioning all access use cases through a single SSL VPN gateway might ultimately cost less and be easier to manage.

While many organizations still implement IPSec solutions today, however, for secure remote access the momentum has clearly shifted to SSL VPNs. Some organizations replace older versions of IPSec with newer versions that better streamline the provisioning of agents, or provide elements of end point control.

Nevertheless, these augmented IPSec VPNs still may not be as flexible or robust as SSL VPN solutions.

With increased access from unmanaged end point devices, end point control becomes a key risk factor. For managed devices, some IPSec solution providers suggest keeping IPSec and adding a network access control (NAC) solution. However, this greatly adds to the costs and complexity of administering and maintaining a separate appliance to achieve end point control, and still does not provide granular access controls down to the application layer, essentially allowing the remote device to be a node on the network.

Replacing IPsec

The ascendancy of IPSec technology as an innovative remote access solution peaked nearly a decade ago. IPSec VPNs are no longer an effective remote access solution when comparing costs of IT overhead and the desire for granular access controls for highly portal devices with the current demands of an increasingly mobile workforce. With early IPSec implementations, the considerable overhead involved in provisioning, maintaining and supporting dedicated IPSec clients was tolerated because IPSec access tended to be restricted only to relatively few managed-device use cases. In recent years, however, since broadband has become widespread and laptops have become cheaper, there has been greater incentive for IT to deploy more laptops and other mobile devices to more users across the enterprise, increasing the overhead needed to support distributed-client IPSec VPNs. While these devices are more likely to be transported beyond the physical office to be used at home or other remote sites, IPSec still views them as nodes on the network, regardless of location.

Workers are also now accessing corporate resources from more end point devices that are not directly managed by IT, such as home computers, WiFi-enabled laptops, PDAs, smartphones and public kiosks.

While most workers today are not full-time teleworkers, many commonly perform teleworking functions, such as sending and receiving e-mail and attachments from home before or after work hours, on weekends, while on the road or while on vacation. In addition, business partners need limited access to specific network resources, which introduces additional remote access challenges to the IT department in today's world of outsourced supply chains. By providing employees and business partners with wider access to business tools and information, the proliferation of unmanaged end point devices has directly resulted in increased productivity. But it has also greatly increased the complexity for IT in controlling remote access, thereby minimizing the viability of distributed-client IPSec VPNs as an efficient remote access solution.

But still IPsec tunnels are still commonly used in site-to-site communications.

4.04 - Explain the purpose, advantages, and challenges associated with SSL VPN

IPSec vs. SSL VPN: Transition Criteria and Methodology

SSL VPN

SSL is the standard protocol for managing the security of message transmission on the Internet. SSL is a higher-layer security protocol than IPSec, working at the application layer rather than at the network layer. By operating at the application layer, SSL can provide the highly granular policy and access control required for secure remote access. Because SSL is included in all modern browsers, SSL VPNs can empower today's mobile workforce with clientless remote access—while saving IT departments the headache of installing and managing the complexity of IPSec clients. By extending the workplace to home PCs, kiosks,

SSL VPN solutions increase workforce productivity, for users with PDAs, and other unmanaged devices, resulting in a greater return on investment. And by eliminating the need to deploy and support "fat" clients, SSL VPN reduces IT overhead, resulting in a lower total cost of ownership.

An SSL VPN uses SSL to provide end users with authorized and secure access for Web, client/server and file share resources. SSL VPNs deliver user-level authentication, ensuring that only authorized users have access to the specific resources allowed by the company's security policy. SSL VPNs start with providing access via a Web browser, removing the need for IT to provision clients to the end point device. For advanced access, agents may be required but SSL VPNs allow IT to have agents provisioned and activated within the context of the Web browser where Active X or Java based "thin" clients are transparently pushed through the browser, Alternatively, most SSL VPNs allow IT to pre-provision the agents directly to a user's device, allowing the user to directly access the SSL VPN without having to open a Web browser.

Potential Benefits of Transitioning to an SSL VPN:

- Increased productivity: SSL VPNs work in more places, including home PCs, kiosks, PDAs and unmanaged devices over wired and wireless networks.
- Lower costs: SSL VPNs are clientless or use lightweight Web-delivered clients rather than "fat" IPSec clients, reducing management and support calls.
- Broadened security: SSL VPNs provide granular access and end point control to managed and non-managed devices

Why you should transition to SSL VPN

Today's modern mobile workforce demands more secure access to more resources from more remote devices and platforms than ever before. Corporate boundaries are blurring, with partners, vendors and consultants playing as vital a role in daily operations as employees do. These changes suggest the need for an inverted model for the corporate network, evolving from the traditional enclosed-perimeter model to a distributed global network that connects employees, partners and customers over multiple Internet, intranet and VoIP channels. IT managers must now assume that any user and device is a potential risk point, whether the user is accessing remotely or plugged directly into the LAN. Disaster recovery and business continuity initiatives pose additional incentive to provide remote access from any end point location. Policy based granular access control becomes imperative.

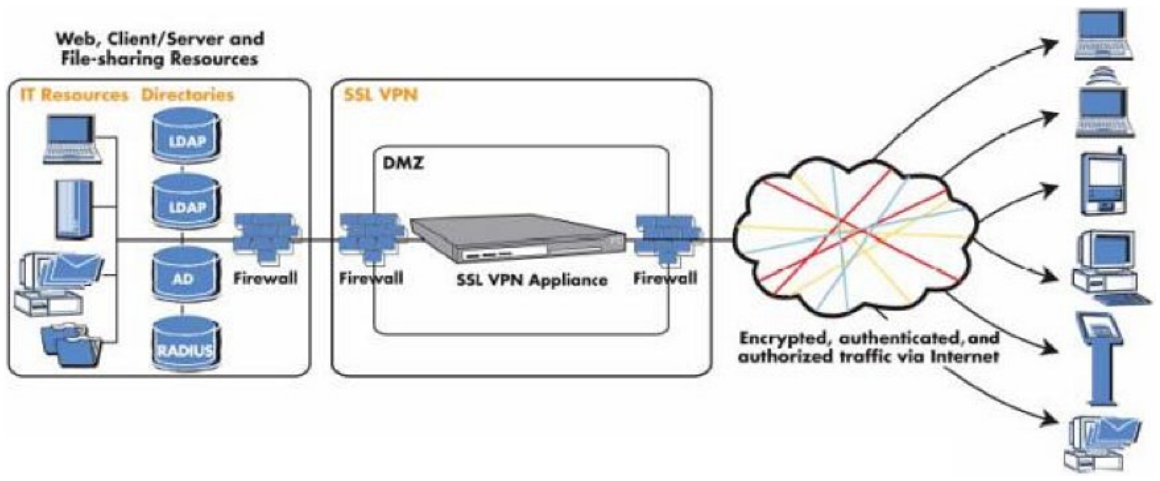
Securing inverted networks with granular access control is an ideal use case for SSL VPN technology. SSL based access control appliances are the key to achieving application access control. SSL VPN solutions can detect what is running on the end point device, protect applications with granular access control based on user identity and device integrity and connect users securely and easily to applications on any device.

Because SSL is part of any Web browser, SSL VPN solutions provide clientless and Web-delivered thin client access that significantly increases the number of points from which employees, partners and customers can access network data. SSL VPN solutions greatly simplify the connection process for mobile 7 users, seamlessly traversing NAT, firewalls and proxy servers. SSL VPN solutions reduce IT support costs, lowering total cost of ownership. SSL VPN clientless access minimizes the IT overhead involved in provisioning, configuring and maintaining an enterprise remote access solution. Alternatively, certain SSL

VPN tunnel solutions provide a complete “in-office” experience by deploying an auto-updating, Web delivered thin client, eliminating the need for direct IT intervention. SSL VPN solutions also streamline administration costs by controlling all access to enterprise resources via a single, centralized gateway.

SSL VPN solutions also provide greater security compared to IPSec. Since SSL is an application layer protocol, an SSL VPN is inherently better suited for securing application-based remote access. SSL VPN solutions provide secure, granular access controls, ensuring that users gain access only to the designated resources or applications specific to their needs and according to security policy.

With SSL VPN solutions, end-user access to any given resource is restricted unless authorized. As a result, SSL VPN technology provides the granular access control that requires all users, regardless of location, to be granted explicit permission to access specific network resources. With SSL VPN technology, access control to applications and networks can be as general or specific as required to meet regulatory compliance and corporate security mandates.


Web, Client/Server and File-sharing Resources
IT Resources
Directories
LDAP
LDAP
AD
RADIUS
Firewall
SSL VPN
DMZ
Firewall
SSL VPN Appliance
Firewall
Encrypted, authenticated, and authorized traffic via Internet

An SSL VPN solution provides secure remote access to specific corporate resources.

4.04 – Given a list of environments/situations, determine which is appropriate for an IPsec solution and which is appropriate for an SSL VPN solution

SSL VPN vs IPSec

When presented with scenario-based questions on which solution is more appropriate IPSec or SSL VPN. Just remember that SSL VPN is the best solution for remote users to access business resources remotely and IPSec is the best solution for tunneling traffic between two business locations.

SECTION 5 – APPLICATION DELIVERY PLATFORMS

Objective - 5.01 Describe the purpose, advantages, use cases, and challenges associated with hardware based application delivery platforms and virtual machines

5.01 - Explain when a hardware based application deliver platform solution is appropriate and when a virtual machine solution is appropriate

Hardware vs. Software

Different situations call for the BIG-IP Virtual Edition and the physical BIG-IP hardware.

The Virtual Edition should be used for discrete workloads on commodity hardware. It also provides flexible and quick deployment options and failure isolation. Virtual Editions can also give you the flexibility to run in proprietary computing environments like Amazon Web Services (AWS) and other public cloud offerings.

Physical hardware provides a number of important benefits and is necessary in many situations. F5 hardware is purposefully built to provide high performance for application delivery. Using an F5 physical platform also offers a single-vendor solution. If a customer is running an F5 Virtual Edition on an HP server and an issue arises, troubleshooting between vendors is more difficult. When F5 hardware runs F5 software, this is avoided.

Additionally, a BIG-IP physical device can run all BIG-IP modules and perform hardware SSL offload and compression. It provides customers with all other features that may be important to their deployment, such as always-on management, certifications, special-purpose FPGAs, and improved security.

5.01 - Explain the purpose, advantages, and challenges associated with hardware based application deliver platform solutions

BIG-IP System Hardware Datasheet

Hardware

BIG-IP ADC appliances can help you simplify your network by offloading servers and consolidating devices, saving management costs as well as power, space, and cooling in the data center.

With the massive performance and scalability of the BIG-IP platform, you can reduce the number of Application Delivery Controllers you need to deliver even the most demanding applications. By offloading computationally intense processes, you can significantly reduce the number of application servers needed.

BIG-IP hardware includes:

- SSL hardware acceleration—Offload costly SSL processing and accelerate key exchange and bulk encryption with best-in-market SSL performance.
- Hardware compression*—Cost-effectively offload traffic compression processing from your servers to improve page load times and reduce bandwidth utilization.
- OneConnect connection pooling—Aggregate millions of TCP requests into hundreds of server-side connections. Increase server capacity and ensure requests are handled efficiently.
- Embedded Packet Velocity Acceleration (ePVA)*—Provide specific application delivery optimizations, support for low latency and tunneling protocols, and denial-of-service (DoS) protection. ePVA uses field-programmable gate array (FPGA) technology tightly integrated with TMOS and software to deliver:
- High performance interconnects between Ethernet ports and processors.
- L4 offload, enabling leading throughput and reduced load on software.
- Hardware-accelerated SYN flood protection.
- More than 65 types of DoS attacks detected and mitigated in hardware.
- Native Financial Information eXchange (FIX) support for message routing and tag substitution while maintaining low latency requirements.

Advantages

The Advantages of F5 BIG-IP Technology

Unique architecture and patented hardware and software innovations from F5 offer unmatched capabilities, including:

F5 ScaleN architecture

ScaleN enables you to scale performance on demand, virtualize, or horizontally cluster multiple BIG-IP devices, creating an elastic Application Delivery Networking infrastructure that can efficiently adapt as your business needs change.

On-demand scaling

Increase capacity and performance with on-demand scaling, where you can simply add more power to your existing infrastructure instead of adding more devices. The latest BIG-IP appliance models can be upgraded to the higher performance model within each series through on-demand software licensing. On- demand licensing enables organizations to right-size application delivery services and support growth without requiring new hardware.

Operational scaling

F5 can virtualize ADC services with a multi-tenant architecture that supports a variety of BIG-IP versions and product modules on a single device. Multi- tenant device virtualization is provided by F5's unique Virtual Clustered Multiprocessing (vCMP) technology, which enables select hardware platforms to run multiple BIG-IP guest instances. Each BIG-IP guest instance looks and acts like a physical BIG-IP device, with a dedicated allocation of CPU, memory, and other resources.

You can further divide each vCMP guest using multi-tenant features such as partitions and route domains, which can isolate configuration and networks on a per-virtual-domain basis. Within each virtual domain, you can further isolate and secure configuration and policies by using a role-based access system for greater administrative control. When combining both route domains/partitions with vCMP guests, F5 provides the highest density multi-tenant virtualization solution that can scale to thousands of virtual ADC (vADC) instances.

This ability to virtualize BIG-IP ADC services means service providers and enterprise users can isolate based on BIG-IP version, enabling departmental or project-based tenancy as well as performance guarantees, while benefiting from managing a single, consolidated application delivery platform and increased utilization.

Application scaling

Increase capacity by adding BIG-IP resources through an all-active approach. With application scaling, you can scale beyond the traditional device pair to eliminate the need for idle and costly standby resources. Application scaling achieves this through two forms of horizontal scale: Application Service Clustering, which focuses on application scalability and high availability, and Device Service Clustering, designed to efficiently and seamlessly scale BIG-IP application delivery services.

Application Service Clustering delivers sub-second failover and comprehensive connection mirroring for a highly available cluster of up to eight devices at the application layer, providing highly available multi-tenant deployments. Workloads can be moved across a cluster of devices or virtual instances without interrupting other services and can be scaled to meet the business demand.

Device Service Clustering can synchronize full device configurations in an all-active deployment model, enabling consistent policy deployment and enforcement across devices—up to 32 active nodes. This ensures a consistent device configuration that simplifies operations.

Challenges

Some of the only challenges with hardware are that it can take longer to acquire for implementations, which can add to time lines in projects, and some public cloud environments do not let you run your own hardware, since it is a strictly virtualized environment.

5.01 - Explain the purpose, advantages, and challenges associated with virtual machines

Virtual Hardware

Virtualization is critical to maintaining an adaptable network and accomplishing the scale, consolidation, and business continuity demanded by today's advanced application infrastructures.

F5 BIG-IP virtual editions (VEs) are virtual application delivery controllers (vADCs) that can be deployed on all leading hypervisors and cloud platforms running on commodity servers. BIG-IP VEs deliver all the same market-leading Software-Defined Application Services (SDAS)—including advanced traffic management, acceleration, DNS, firewall, and access management—that run on F5 purpose-built hardware. VE software images are downloadable and portable between on-premises virtualized data centers, public, and hybrid cloud environments. With BIG-IP virtual editions and F5 BIG-IQ management solutions, you can rapidly provision consistent application services across the data center and into the cloud.

Advantages

Deploy with increased agility

- Quickly and easily spin up, spin down, or migrate application delivery services in and across the data center and public cloud, using instant deployment options as needed.

Achieve automation and orchestration in cloud architectures

- Automate deployment and configuration or integrate with leading orchestration frameworks—in cloud or software-defined networking (SDN) environments through application level templates, REST APIs, and granular programmability.

Optimize application services more efficiently

- Rapidly provision and consolidate application services on your existing servers, unlocking the broadest feature density through flexible licensing models that align to your business needs.

Provide the ultimate in flexibility

- Get the most flexible deployment options in the industry, with support across all major virtualization platforms for both private and public cloud environments.

Challenges

Some of the only challenges with have to do with performance when compared to what dedicated hardware can do. Throughput speeds and volumetric processing of SSL transactions per second does not compare with anything above a 4000 series in hardware.

5.01 - Explain the advantages of dedicated hardware (SSL card, compression card)

The charts below tell the story of Hardware vs. Software, especially in relation to SSL offload with hardware or software.

BIG-IP Virtual Editions Datasheet

Virtual Editions

Available in a range of performance options, F5 virtual editions can be sized and configured to suit the application services required. Maximum performance is based on applicable VE licensed performance ranges and resources (number of CPU cores/memory) allocated.

Performance	Starting	Maximum*
L7 requests per second	3,000	450,000
L4 connections per second	2,000	135,000
Throughput	25 Mbps	10 Gbps**
Maximum connections	1 million	10 million
SSL		
Maximum SSL TPS (1K keys/2K keys)	900/900	12,000/3,550
SSL throughput	23 Mbps	4 Gbps
Software compression		
Maximum software compression throughput	20 Mbps	4 Gbps

BIG-IP System Datasheet

Appliance Hardware Editions

Available in a range of performance options, F5 hardware appliances can be sized and configured to suit the application services required.

Performance	2000s Model	12250v Model
L7 requests per second	212,000	4 million
L4 connections per second	75,000	1.5 million
Throughput	5 Gbps L4/L7	84 Gbps/40 Gbps L4/L7
Maximum connections	5 million	10 million
SSL		
Maximum SSL TPS (2K keys)	2,000	240,000
SSL throughput	4 Gbps*	40 Gbps*
Hardware compression		
Maximum compression throughput	N/A	40 Gbps

VIPRION Datasheet

VIPRION Hardware Editions

Available in a range of performance options, F5 hardware blades for chassis platforms can be sized and configured to suit the application services required.

Performance	2150 blades	4340N blades
L7 requests per second	1 million	2.5 million
L4 connections per second	400,000	1.4 million
Throughput	40Gbps/18Gbps L4/L7	80 Gbps/40 Gbps L4/L7
Maximum connections	24 million	72 million
SSL		
Maximum SSL TPS (2K keys)	10,000	30,000
SSL throughput	4 Gbps*	20 Gbps*
Hardware compression		
Maximum compression throughput	10 Gbps	20 Gbps

Objective - 5.02 Describe the purpose of the various types of advanced acceleration techniques

5.02 - Describe the purpose of TCP optimization

Acceleration 101

Optimizing TCP

Although TCP is ubiquitous today, the protocol has undergone many updates to help overcome limitations that existed in earlier versions. An acceleration device can help optimize TCP by implementing features that may not be present in either a client or server's TCP implementation.

An acceleration device can also decrease the number of server-side TCP connections required to service client requests. Additionally, it can help accelerate HTTP traffic by increasing the number of simultaneous client-side TCP connections a browser can open while downloading a web page.

General TCP Optimizations

Because it operates as a proxy, an acceleration device may be able to implement features missing from a client or server that can help speed application delivery. The acceleration device may be able to leverage optimizations natively supported by particular client or server operating systems and is likely to be able to implement optimizations that are not operating-system specific. The benefit of high speed, high latency WAN connections are that the acceleration device can perform TCP window scaling to improve performance. To overcome packet loss, the acceleration device can implement selective TCP acknowledgements (SACK) and advanced congestion control algorithms to prevent TCP from reducing throughput.

These are only two examples. Some acceleration devices implement hundreds of improvements to TCP in order to help it perform better.

Decreasing Server-side TCP Connections

Reducing server-side connection processing can dramatically improve application performance and reduce the number of servers required to host an application. TCP connection setup and teardown requires significant overhead, particularly for servers. As the number of open server connections increases, maintaining the open connections while simultaneously opening new connections can severely degrade server performance and therefore, user response time.

Although multiple transactions (for example, file transfers) can occur within a single TCP connection, a connection is generally between one client and one server. Normally, a connection closes either when a server reaches a defined transaction limit or when a client has transferred all needed files from that server. Because an acceleration device operates as a proxy, it can aggregate, or “pool,” TCP server-side connections by combining many separate transactions, potentially from many users, through fewer (or one) TCP connections. The acceleration device opens new server-side connections only when necessary, and instead reuses existing connections for requests from other users whenever possible.

Increasing Client-side TCP Connections

By default, most web browsers limit the maximum number of simultaneous HTTP/HTTPS connections that the browser can open to one URL. For example, Microsoft Internet Explorer v7 and below limit the maximum number of simultaneous connections to two per domain. Earlier versions of Firefox limit the browser to eight connections per domain. Given that a web page can contain dozens of objects, this limitation can greatly slow page-loading times.

For example, suppose a user running Internet Explorer v7 requests a page from a web server that returns a response containing a list of the 30 objects that make up the web page. Further assume that that all objects are accessed through the domain, www.example.com. The browser opens two connections to www.example.com, requests one object at a time per connection (the limit imposed by TCP), and then reuses the two connections until all files have been downloaded or the connection reaches the server's transaction limit. If the connection suffers high latency, round trip time is high and download speed can be greatly reduced.

If the server terminates the connection after reaching a pre-defined transaction limit, the browser opens another connection to that URL. This process continues until the page downloads completely. Operating this way needlessly increases the page load time.

Some acceleration devices can "spoof" a browser by modifying the URLs in an HTTP response to speed page downloading. The modified URLs must first be defined in DNS to point to the same IP address. When examining the server response, the modified names appear to the browser to be different servers, so the web browser opens parallel connections to these altered URLs rather than serially downloading the objects from one URL.

5.02 - Describe the purpose of HTTP Keep-alives, caching, compression, and pipelining

Acceleration 101

HTTP Protocol and Web Application Optimizations

HTTP protocol optimizations maintain high user performance levels by optimally tuning each HTTP session. For example, some web applications are unable to return an HTTP 304 status code (Not Modified) in response to a client request rather than returning the entire object. Because an acceleration device proxies connections and caches content, it may be able to note when there is no change to a requested object and return the 304 response instead. This enables the browser to load the content from its own cache, even in conditions where the web application is hard-coded to re-send the object.

Some acceleration devices can additionally examine and change server responses to provide better browser and server performance. For example, some off-the-shelf and custom applications add a no-cache header to some objects, which directs a browser not to cache an object, rather to download the object from the origin web server every time. The purpose of the no-cache header is to ensure a browser always downloads dynamic (changing) data.

However, applications in some cases mark static data like a company logo as being non-cacheable. Some acceleration devices can re-write the server response to mark the object as being cacheable and supply a more realistic expiration date. This feature can help remedy problems with off-the-shelf or custom-developed applications where code cannot easily be modified.

Caching

Caching involves storing data close to users and re-using the data during subsequent requests. Caching usually takes one of three forms. The first is the classic approach taken by web browsers and web applications. In this case, the web application code running on a server instructs a browser to cache an object marked as static for a specific time period. During that time period, the browser reads the object from cache when building a web page until the content expires. The client then reloads the content. Caching prevents the browser from having to waste time and bandwidth by always accessing data from a central site. This is the most common form of caching in use today.

The second form involves deploying an acceleration device in a data center to offload requests for web application content from web servers. This method operates asymmetrically, with the acceleration device caching objects from web servers and delivering them directly to users. Some acceleration devices cache static content only, while some additionally can process HTTP responses, include objects referenced in a response, and send the included objects as a single object to a browser. This not only offloads web server processing but also offloads web browser processing too. A side benefit to this approach is that as the acceleration device is typically in the data center and connected to higher-speed connections, the acceleration device can both assemble the objects from instructions in the HTTP response and deliver them using fewer objects and with fewer transactions.

Operating in this manner, caching can dramatically reduce server TCP and application processing, improve web page loading time, and hence reduce the need to regularly expand the number of web servers required to service an application.

The third form of caching involves using symmetric acceleration devices to cache and serve content to users at the remote site. The remote acceleration device serves content locally whenever possible, which reduces both response time and network utilization. This form of caching can be deployed not only for HTTP, but also for other protocols as well.

Caching has its limitations. First, if the client-side acceleration device serves content regardless of whether it is in contact with its remote peer, the client side device must implement access control to prevent unauthorized access to an object. Second, the client-side device may serve older, stale versions of content that change after the connection between the devices is broken. While this typically is not an issue with static web content, it can have significant impact on files that regularly change. When both issues are addressed, remote caching can greatly improve application performance, especially for web applications and static files used with other applications.

Compression

Compression is one of the oldest acceleration techniques, having been around for decades. GZIP, the most common compression algorithm, is implemented in virtually every web browser and server. Compression algorithms such as GZIP are good at finding small, repeating patterns and reducing the characters required to send them. Besides web servers and browsers, acceleration devices implement compression. This is done for two reasons: first to offload compression overhead from web servers and second, to enable the acceleration device to perform other optimizations that improve performance for an HTTP/HTTPS stream.

Compression can be computationally expensive, especially for algorithms that provide high compression levels. These algorithms are of limited use with high-speed communication, where delays must be minimized to maintain rapid user response times. More effective compression algorithms are therefore limited to low-speed communications where more time is available to perform compression processing without degrading user throughput and hence, response times. Fortunately, compression hardware assist is now available in some acceleration devices that can achieve compression rates in excess of 1 Gbps.

HTTP Pipelining: A Security Risk Without Real Performance Benefits

Pipelining

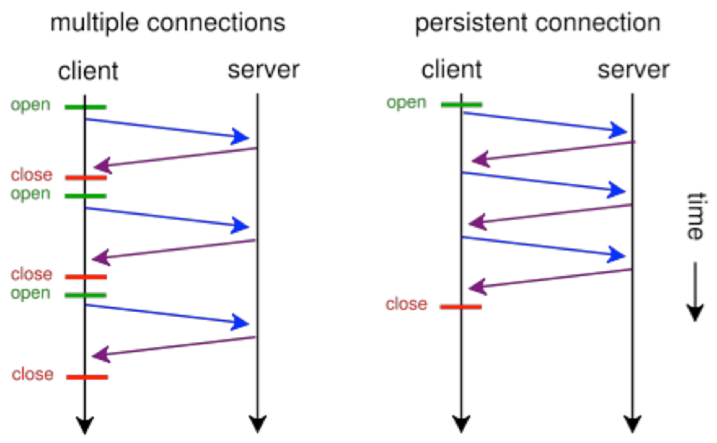
Everyone wants web sites and applications to load faster, and there's no shortage of folks out there looking for ways to do just that. But all that glitters are not gold and not all acceleration techniques actually do all that much to accelerate the delivery of web sites and applications. Worse, some actual incur risk in the form of leaving servers open to exploitation.

A brief history

Back in the day when HTTP was still evolving, someone came up with the concept of persistent connections. See, in ancient times – when administrators still wore togas in the data center – HTTP 1.0 required one TCP connection for every object on a page. That was okay, until pages started comprising ten, twenty, and more objects. So someone added an HTTP header, Keep-Alive, which basically told the server not to close the TCP connection until (a) the browser told it to or (b) it didn't hear from the browser for X number of seconds (a time out). This eventually became the default behavior when HTTP 1.1 was written and became a standard.

I told you it was a brief history.

This capability is known as a persistent connection, because the connection persists across multiple requests. This is not the same as pipelining, though the two are closely related. Pipelining takes the concept of persistent connections and then ignores the traditional request – reply relationship inherent in HTTP and throws it out the window.

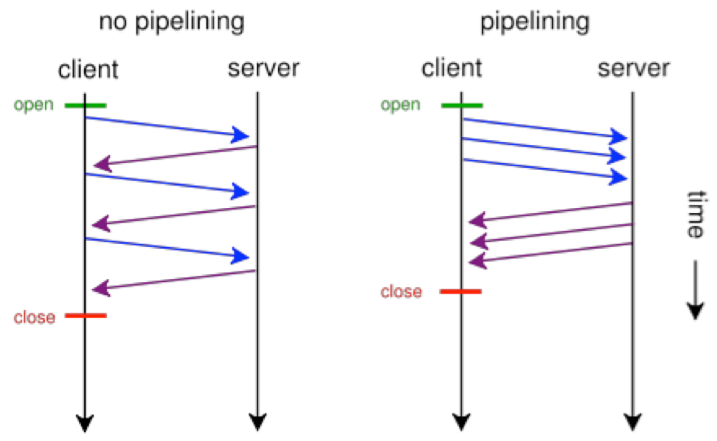


The general line of thought goes like this:

"Whoa. What if we just shoved all the requests from a page at the server and then waited for them all to come back rather than doing it one at a time? We could make things even faster!"

HTTP pipelining

In technical terms, the browser initiates HTTP pipelining by opening a connection to the server and then sending multiple requests to the server without waiting for a response. Once the requests are all sent then the browser starts listening for responses. The reason this is considered an acceleration technique is that by shoving all the requests at the server at once you essentially save the RTT (Round Trip Time) on the connection waiting for a response after each request is sent.



Why it just doesn't matter anymore (and maybe never did)

Unfortunately, pipelining was conceived of and implemented before broadband connections were widely utilized as a method of accessing the Internet. Back then, the RTT was significant enough to have a negative impact on application and web site performance and the overall user-experience was improved by the use of pipelining. Today, however, most folks have a comfortable speed at which they access the Internet and the RTT impact on most web application's performance, despite the increasing number of objects per page, is relatively low.

There is no arguing, however, that some reduction in time to load is better than none. Too, anyone who's had to access the Internet via high latency links can tell you anything that makes that experience faster has got to be a Good Thing. So what's the problem?

The problem is that pipelining isn't actually treated any differently on the server than regular old persistent connections. In fact, the HTTP 1.1 specification requires that a "server MUST send its responses to those requests in the same order that the requests were received." In other words, the requests are return in serial, despite the fact that some web servers may actually process those requests in parallel. Because the server MUST return responses to requests in order that the server has to do some extra processing to ensure compliance with this part of the HTTP 1.1 specification. It has to queue up the responses and make certain responses are returned properly, which essentially negates the performance gained by reducing the number of round trips using pipelining.

Depending on the order in which requests are sent, if a request requiring particularly lengthy processing – say a database query – were sent relatively early in the pipeline, this could actually cause a degradation in performance because all the other responses have to wait for the lengthy one to finish before the others can be sent back.

Application intermediaries such as proxies, application delivery controllers, and general load-balancers can and do support pipelining, but they, too, will adhere to the protocol specification and return responses in the proper order according to how the requests were received. This limitation on the server side actually inhibits a potentially significant boost in performance because we know that processing dynamic requests takes longer than processing a request for static content. If this limitation were removed it is possible that the server would become more efficient and the user would experience non-trivial improvements in performance. Or, if intermediaries were smart enough to rearrange requests such a way that their execution were optimized then we'd maintain the performance benefits gained by pipelining. But that would require an understanding of the application that goes far beyond what even today's most intelligent application delivery controllers are capable of providing.

The silver lining

At this point it may be fairly disappointing to learn that HTTP pipelining today does not result in as significant a performance gain as it might at first seem to offer (except over high latency links like satellite or dial-up, which are rapidly dwindling in usage). But that may very well be a good thing.

As miscreants have become smarter and more intelligent about exploiting protocols and not just application code, they've learned to take advantage of the protocol to "trick" servers into believing their requests are legitimate, even though the desired result is usually malicious. In the case of pipelining, it would be a simple thing to exploit the capability to enact a layer 7 DoS attack on the server in question. Because pipelining assumes that requests will be sent one after the other and that the client is not waiting for the response until the end, it would have a difficult time distinguishing between someone attempting to consume resources and a legitimate request.

Consider that the server has no understanding of a "page". It understands individual requests. It has no way of knowing that a "page" consists of only 50 objects, and therefore a client pipelining requests for the maximum allowed – by default 100 for Apache – may not be seen as out of the ordinary. Several clients opening connections and pipelining hundreds or thousands of requests every second without caring if they receive any of the responses could quickly consume the server's resources or available bandwidth and result in a denial of service to legitimate users.

So perhaps the fact that pipelining is not really all that useful to most folks is a good thing, as server administrators can disable the feature without too much concern and thereby mitigate the risk of the feature being leveraged as an attack method against them.

Pipelining as it is specified and implemented today is more of a security risk than it is a performance enhancement. There are, however, tweaks to the specification that could be made in the future that might make it more useful. Those tweaks do not address the potential security risk, however, so perhaps given that there are so many other optimizations and acceleration techniques that can be used to improve performance that incur no measurable security risk that we simply let sleeping dogs lie.

Conclusion

This document is intended as a study guide for the F5 101 – Application Delivery Fundamentals exam. This study guide is not an all-inclusive document that will guarantee a passing grade on the exam. It is intended to be a living doc and any feedback or material that you feel should be included, to help exam takers better prepare, can be sent to channeleng@f5.com.

Thank you for using this study guide to prepare for the 101 – Application Delivery Fundamentals exam and good luck with your certification goals.

Thanks,
Eric Mitchell
Channel FSE, East US and Federal

F5 Networks, Inc. 401 Elliott Avenue West, Seattle, WA 98119 888-882-4447 f5.com

Americas	Asia-Pacific	Europe/Middle-East/Africa	Japan
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com