2

Understanding Layer 2, 3, and 4 Protocols

While many of the concepts well known to traditional Layer 2 and Layer 3 networking still hold true in content switching applications, the area introduces new and more complex themes that need to be well understood for any successful implementation. Within the discussion of content networking, we will replace terms such as *packets* and *frames* with *sessions* and *transactions* as we move our attention further up the OSI Seven Layer Model. Before we move into these new terms, however, let's look at some standard Layer 2, 3, and 4 networking concepts.

The OSI Seven Layer Model—What *Is* a Layer?

Established in 1947, the International Organization for Standardization (ISO) was formed to bring together the standards bodies from countries around the world. Their definition of the model for Open Systems Interconnection, or OSI, is used to define modes of interconnection between different components in a networking system. This means that the physical method of transport can be designed independently of the protocols and applications running over it. For example, TCP/IP can be run over both Ethernet and FDDI networks, and Novell's IPX and Apple's AppleTalk protocols can both be run over Token Ring networks. These are examples of having independence between the physical network type and the upper layer protocols running across them. Consider also, two TCP/IP-enabled end systems communicating across a multitude of different network types, such as Ethernet, Frame Relay, and ATM. Figure 2–1 shows the OSI Seven Layer Model.

When we talk about Layer 2 and Layer 3 networking, it is these layers that we're referring to, and logically the further up the OSI model we move, the greater intelligence we can use in networking decisions.
Each layer plays its part in moving data from one device to another across a network infrastructure by providing a standard interface to the surrounding layers.

The Application Layer (Layer 7)

The top layer in the stack, the Application layer is where the end-user application resides. Think of the Application layer as the browser application or email client for a user surfing the Web or sending email. Many protocols are defined for use at the Application layer, such as HTTP, FTP, SMTP, and Telnet.

In content switching terms, Layer 7 refers to the ability to parse information directly generated by the user or application in decision making, such as the URL typed by the user in the Web browser. For example, *http://www.foocorp.com* is an example of Application layer data.

The Presentation Layer (Layer 6)

The Presentation layer is used to provide a common way for applications (residing at the Application layer) to translate between data formats or perform encryption and decryption. Mechanisms to convert between text formats such as ASCII and Unicode may be considered part of the Presentation layer, along with compression techniques for image files such as GIF and JPEG.

7	Application Layer
6	Presentation Layer
5	Session Layer
4	Transport Layer
3	Network Layer
2	Data Link Layer
1	Physical Layer

Figure 2–1 The OSI Seven Layer Model.

The Session Layer (Layer 5)

The Session layer coordinates multiple Presentation layer processes communicating between end devices. The Session layer is used by applications at either end of the communication between end devices to tie together multiple Transport layer sessions and provide synchronization between them.

The HTTP protocol can use multiple TCP connections to retrieve objects that make up a single Web page. The Session layer provides application coordination between these separate TCP connections.

The Transport Layer (Layer 4)

The Transport layer is responsible for providing an identifiable and sometimes reliable transport mechanism between two communicating devices. User or application data, having passed through the Presentation and Session layers, will typically be sequenced and checked before being passed down to the Network layer for addressing.

The Transport layer is the first at which we see the concept of packets or datagrams of information that will be transported across the network. TCP, UDP, and ICMP are examples of Layer 4 protocols used to provide a delivery mechanism between end stations. It is also at this layer in the model that applications will be distinguished by information in the Layer 4 headers within the packets. Content switching operates most commonly at this layer by using this information to distinguish between different applications and different users using the same application.

The Network Layer (Layer 3)

Whereas Layer 4 is concerned with *transport* of the packets within a communication channel, the Network layer is concerned with the *delivery* of the packets. This layer defines the addressing structure of the internetwork and how packets should be routed between end systems. The Network layer typically provides information about which Transport layer protocol is being used, as well as local checksums to ensure data integrity. Internet Protocol (IP) and Internet Packet Exchange (IPX) are examples of Network layer protocols.

Traditional Internet routers operate at the Network layer by examining Layer 3 addressing information before making a decision on where a packet should be forwarded. Hardware-based Layer 3 switches also use Layer 3 information in forwarding decisions. Layer 3 routers and switches are not concerned whether the packets contain HTTP, FTP, or SMTP data, but simply where the packet is flowing to and from.

The Data Link Layer (Layer 2)

The Data Link layer also defines a lower level addressing structure to be used between end systems as well as the lower level framing and checksums being used to transmit onto the physical medium. Ethernet, Token Ring, and Frame Relay are all examples of Data Link layer or Layer 2 protocols.

Traditional Ethernet switches operate at the Data Link layer and are concerned with forwarding packets based on the Layer 2 addressing scheme. Layer 2 Ethernet switches are not concerned with whether the packet contains IP, IPX, or AppleTalk, but only with where the MAC address of the recipient end system resides.

The Physical Layer (Layer 1)

As with all computer systems, networking is ultimately about making, moving, and storing 1s and 0s. In networking terms, the Physical layer defines how the user's browser application data is turned into 1s and 0s to be transmitted onto the physical medium. The Physical layer defines the physical medium such as cabling and interface specifications. AUI, 10Base-T, and RJ45 are all examples of Layer 1 specifications.

Putting All the Layers Together

Let's take an example of a Web user visiting the Web site of Foocorp, Inc. Within the browser application, at the Application layer, the user will type in the URL, typically something like *http://www.foocorp.com/*. While this is the only input the *user* will provide the application, there is much more information generated by the browser application itself, including:

- The type of browser being used (e.g., Microsoft Internet Explorer, Netscape)
- The operating system running on the user's machine

- The version of the HTTP protocol being used by the browser
- The language, or languages, supported by the browser (e.g., English, Japanese, etc.)
- Any Presentation layer standards that are supported by the browser, such as compression types, text formats, and file types

In terms of HTTP-based Web browser traffic, these pieces of information can be thought of as the Application, Presentation, and Session layers of the OSI model. They provide not only the raw data input by the user in the application, but also information needed by the application to ensure successful communication with the end system; in this case, a Web server at Foocorp. HTTP information for the Web user would look something like:

```
Hypertext Transfer Protocol

GET / HTTP/1.0\r\n
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg\r\n
Accept-Language: en-gb\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)\r\n
Host: www.foocorp.com\r\n
Connection: Keep-Alive\r\n
\r\n
```

Once this application information has been generated, it can be packaged and passed on to the next layer for transport. HTTP requires a connection-oriented Transport layer protocol to guarantee the delivery of each packet in the session. Transmission Control Protocol (TCP) is used in HTTP applications to ensure this successful packet delivery. Other applications will make use of different Transport layer protocols. TFTP, for example, uses the User Datagram Protocol (UDP) as its Layer 4 transport because it does not require the guaranteed delivery provided by TCP. Routing updates sent between Layer 3 devices can use OSPF, RIP, or BGP as their Layer 4 transport.

At the Transport layer, information about the port numbers, sequence numbers, and checksums are included to provide reliable transport. The Layer 4 headers in our example would look something like:

```
Transmission Control Protocol
    Source port: 3347 (3347)
    Destination port: http (80)
    Sequence number: 52818332
    Next sequence number: 52818709
    Acknowledgement number: 3364222344
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 17520
    Checksum: 0xb043 (correct)
```

Once the Transport layer information has been added to the head of the packet, it is passed to the Network layer for the Layer 3 headers to be appended. The Network layer will include information on the IP addresses of both the client and the end system, and a reference to which Transport layer protocol has been used. The Network layer information is used to ensure the correct delivery path from the client to the end system and the ability for the receiver to identify which Transport layer process the frames should be forwarded to once they arrive. For the Web user example, the Network layer information would look as follows:

```
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Time to live: 128
    Protocol: TCP
    Header checksum: 0x2df9 (correct)
    Source: 192.168.254.201 (192.168.254.201)
         Destination: 216.239.51.101 (216.239.51.101)
```

For transmission across the local, physical network, the frame is then passed to the Data Link layer for the addition of the local physical addresses. In terms of Ethernet, this would be the Ethernet Media Access Control (MAC) address of the user machine and the MAC address of the default gateway router on the Ethernet network. The Layer 2 protocol, such as Ethernet, will also include a reference to which Layer 3 protocol has been used and a checksum to ensure data integrity. For our example, the Layer 2 information might look something like:

```
Ethernet II
    Destination: 00:20:6f:14:58:2f (00:20:6f:14:58:2f)
    Source: 00:30:ab:17:0d:1a (00:30:ab:17:0d:1a)
     Type: IP (0x0800)
```

Figure 2–2 depicts this process of repackaging each layer with new header information at the layer below.

Switching at Different Layers

Now that we've seen examples of different information available within different layers of the OSI model, let's look at how this information can be used to make intelligent traffic forwarding decisions. Before the development of switching, Ethernet relied on broadcast or flooding of packets to all end stations within a network to forward traffic. Ethernet is effectively a shared medium with only one Ethernet end station able to transmit at any time. Combine this with early implementation techniques relying on every end station in an Ethernet network seeing every packet, even if it was not addressed to it, and issues of scalability quickly surface.

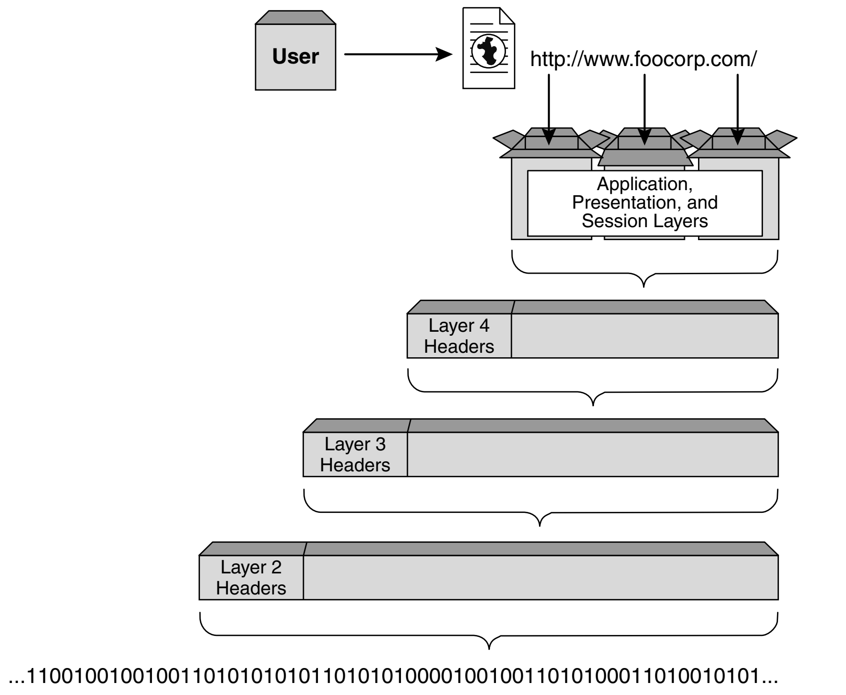
User
http://www.foocorp.com/
Application, Presentation, and Session Layers
Layer 4 Headers
Layer 3 Headers
Layer 2 Headers
...1100100100100110101010101101010100001001001101010001101001010l...

Figure 2–2 Passing data through the seven OSI layers.

Layer 2 Switching

The first implementation of Ethernet or Layer 2 switching uses information in the Ethernet headers to make traffic forwarding decisions. Intelligent switches learn which ports have which end stations attached by recording the Ethernet MAC addresses of packets ingressing the switch. Using this information along with the ability to parse the Layer 2 headers of all packets means that a Layer 2 switch need only forward frames out of ports where it knows the end station to be. For end station addresses that have not yet been learned, frames with unknown destination MAC addresses are flooded out of every port in the switch to force the recipient to reply. This will allow the switch to learn the relevant MAC address, as it will be the *source* address on the reply frame.

Layer 2 switching is implemented along side Layer 3 routing for local area networks to facilitate communication between devices in a common IP subnet. As the information at this layer is relatively limited, the opportunity to configure Layer 2 switches to interpret address information and act upon it in any way other than described previously is generally not required. Many Layer 2 switches will offer the ability to configure intelligent services such as Quality of Service (QoS), bandwidth shaping, or VLAN membership based on the Layer 2 information. Figure 2–3 shows a simplified Layer 2 frame with examples of information that might be used to make switching decisions.

Layer 3 Switching and Routing

Traditional protocol routers work by using information in the Layer 3 headers of Ethernet frames. While routing platforms exist for many different protocols (e.g., IPX, AppleTalk, and DECNet), in TCP/IP terms a router or routing device will typically use the destination IP address in the Layer 3 header to make a forwarding decision. The main advantage of Layer 3 routing in its earliest guises was that it gave the network designer the ability to segregate the network into distinct IP networks and carefully control the traffic and reachability between each.

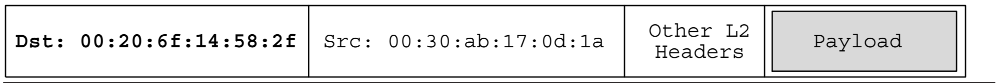

Figure 2–3 Example Layer 2 headers for switching.

Many of the early implementers and pioneers of Layer 3 routing devices used software-based devices as platforms that, while offering a flexible platform for development of the technology, often provided limitations in terms of performance. As Layer 2 switching became more commonplace and the price per port of Ethernet switching systems dropped, manufacturers looked to combine the performance of ASIC-based Layer 2 switching with the functionality and flexibility of Layer 3 routing. Step forward the Layer 3 switch. Layer 3 switches work by examining the destination IP address and making a forwarding decision based on the routing configuration implemented. The destination subnet might be learned via a connected interface, a static route, or a dynamic routing protocol such as RIP, OSPF, or BGP. In all instances, once the Layer 3 switch has examined the frame and compared the destination IP address against the information in its routing database, the destination MAC address is changed and the frame is forwarded through the relevant egress port. For IP frames traversing a Layer 3 device, such as a router or Layer 3 switch, the TTL field in the IP header is also decremented to indicate to end stations and intermediaries that a routing hop has occurred.

It is once we reach the Layer 3 switching environment that configuration for devices become inherently more complex. The administrator must configure the correct routing information to enable basic traffic flow along with the interface IP addresses in each of the subnets to which the Layer 3 switch is attached.

Figure 2–4 shows the typical information used by a Layer 3 switch in making a forwarding decision.

Understanding Layer 4 Protocols

To appreciate the part that a content switch plays in the lifecycle of a user session, it is important to understand the component parts that make up such a session. Many protocols can be considered as Layer 4. Routing protocols such as OSPF, proprietary ones such as EIGRP, redundancy protocols such as the Virtual Router Redundancy Protocol (VRRP), and a host of others such as ICMP,

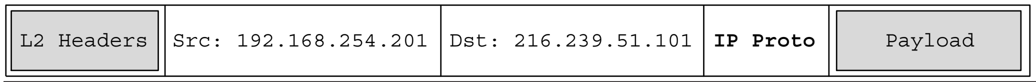

Figure 2–4 Example Layer 3 headers for switching and routing.

IGMP, and IP itself can all be identified by a unique protocol number in the IP header (see Figure 2–5).

The list of IP protocol numbers is administered and controlled by the Internet Assigned Numbers Authority (IANA), and a comprehensive list can be found at *www.iana.org/*. Table 2–1 lists some of the more common IP protocol numbers.

Table 2–1 Some Examples of Common IP Protocol Numbers

IP PROTOCOL NUMBER	LAYER 4 PROTOCOL
1	ICMP—Internet Control Message Protocol
6	TCP—Transmission Control Protocol
17	UDP—User Datagram Protocol
112	VRRP—Virtual Router Redundancy Protocol

Some Layer 4 protocols effectively operate at this layer alone. VRRP, for example, uses Layer 4 headers to transport all information between a series of participating routers in an IP subnet and consequently has no need for upper layer protocol information. Its payload is simply the information contained at Layer 4. Other routing protocols, such as the Border Gateway Protocol (BGP), will use the reliable Layer 4 Transport layer protocol with the BGP routing information and updates carried in the upper layer payloads.

In terms of content switching, the two most commonly understood Layer 4 protocols are TCP and UDP. The majority of the standard Application layer protocols are carried either within TCP or UDP depending on whether there is a requirement for a reliable end-to-end connection. Taking a Web user example, the browser application needs to ensure that all packets are successfully delivered when presenting the user with the desired Web page. The HTTP protocol will

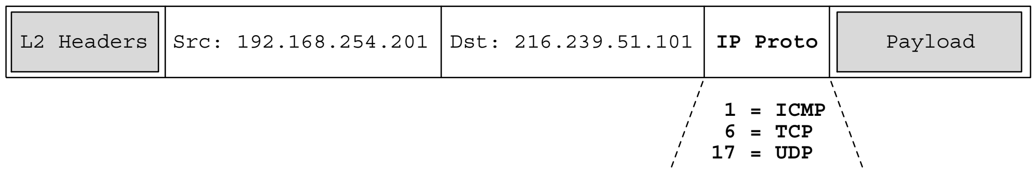

Figure 2–5 Different IP protocol numbers identify which Layer 4 protocol is being used.

therefore rely on TCP as its Transport layer protocol, to guarantee delivery, which in turn will use IP as its delivery mechanism.

Transport Control Protocol (TCP)

As the Layer 3 IP protocol is principally a connectionless and best-efforts delivery mechanism, there is a requirement for many applications to ensure the correctly sequenced delivery of *all* packets within a conversation. Consequently, many applications will use Transport Control Protocol (TCP) at Layer 4 to guarantee successful delivery. TCP has several characteristics built in to ensure this delivery:

- **Checksum**: The TCP header contains a 16-bit data checksum that is computed from all other data elements in the TCP header. The receiving end station uses this checksum to ensure that the packet arrived without corruption.
- **Sequence and acknowledgment numbers**: Each octet of data sent and received by end stations has an associated sequence number associated with it. These sequence numbers are cumulative, whereby a certain sequence number inside the TCP header will be used to indicate that all data up to and including *X* should have been received. Sequence and acknowledgment numbering is used to bring the concept of order to packet delivery over IP.
- **Windowing**: The TCP windowing technique allows two communicating end stations to build on the sequencing and acknowledgments above by removing the need for each sequence of data to be individually acknowledged. In LANs where packet loss is usually minimal, it is far more efficient to allow the sender to transmit several frames of data before an acknowledgment is sent.

Along with these mechanisms, TCP must also be able to uniquely identify each conversation within an internetwork. We've already seen the idea of a TCP port number that is used, among other things, to identify the application process to the high OSI layers during the conversation. Within a TCP conversation, there are in fact two port numbers used: one to identify the sender's listening port and the other to identify the receiver's listening port. Depending on the direction of each individual frame in the conversation, these ports become either the source port or the destination port within the Layer 4 headers.

This combination of source and destination ports, along with the Layer 3 IP addressing, gives TCP the ability to uniquely identify each conversation or session within an internetwork, even in the case of the Internet itself.

The Lifecycle of a TCP Session

Let's put these concepts of addresses, ports, and sequencing numbers together and look at how a conversation between two end stations is initiated, sustained, and terminated. Throughout the following example, we will assume that the client is a PC (10.10.10.10) initiating a connection to a Web server (20.20.20.20).

1. Initiating a Session

Before initiating the session, there are two pieces of information upon which the client must decide. First, in order to identify the session uniquely between itself and the server, it selects a TCP port number to represent the session. This port will be the source port for packets from the client to the server and the destination port for packets from the server to the client. The client will select the source port sequentially on a connection-by-connection basis starting from a value greater than 1024. Port numbers below 1024 are typically referred to as well-known ports and are used to identify well-known applications. Table 2–2 shows some well-known reserved ports as defined by IANA.

Table 2–2 Some Well-Known TCP Port Number Assignments

TCP Port Number	Application
20 and 21	File Transfer Protocol (FTP)
25	Simple Mail Transfer Protocol (SMTP)
23	Telnet
80	HyperText Transfer Protocol (HTTP)

The second element that needs to be decided by the client is the starting sequence number. This will be selected based on an internal 32-bit clock that ensures both randomness and that sequence numbers will not overlap should a lost packet reappear some time after its original transmission. Just as with the

TCP ports used by both the client and server, each side also uses its own sequence numbering to identify where within the session each frame fits.

Once the client has determined these two variables, it is ready to send the first packet of the session and initiate the connection to the server. Using TCP flags, the client will indicate to the server that it wants to initiate a connection by setting the SYN or synchronize flag showing that this is the first pack in the session. In TCP terms, this element is the first packet in what is commonly referred to as the "three-way handshake." This is simply because three packets are exchanged between the client and server to bring the TCP state into that which can transport data. Consequently, no Application layer data is transmitted until at least the fourth packet in the session, a concept which we will see has an important consequence when applied to content switching. Figure 2–6 shows a simplified representation of the three-way handshake to illustrate which side sends which of the packets when a new connection is initiated.

Taking this sequence packet by packet, we can see the importance of the port and sequence numbers in ensuring the reliable transport between the client and server. The first frame from the client to the server initiates the connection by setting the client side port and sequence numbers as shown in Figure 2–7. As we can see, the client chooses a random source port that will be used by the client to identify this session uniquely in cases where it has concurrent sessions to the same server.

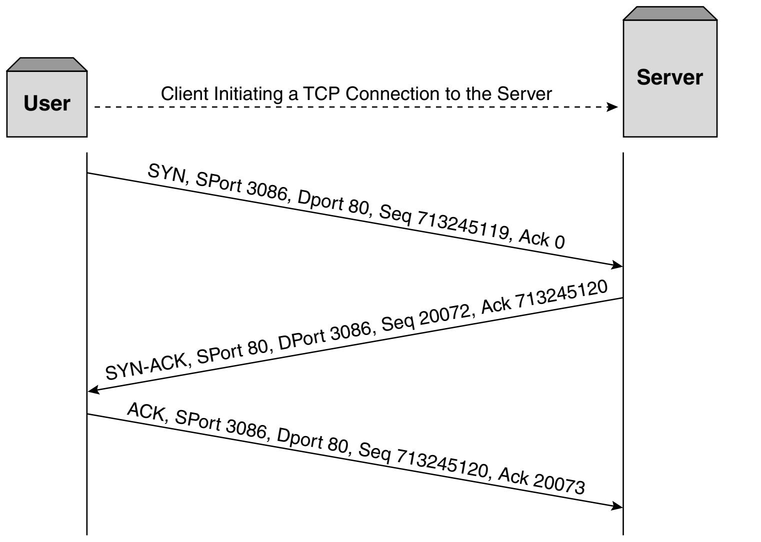
User
Server
Client Initiating a TCP Connection to the Server
SYN, SPort 3086, Dport 80, Seq 713245119, Ack 0
SYN-ACK, SPort 80, DPort 3086, Seq 20072, Ack 713245120
ACK, SPort 3086, Dport 80, Seq 713245120, Ack 20073

Figure 2–6 The TCP three-way handshake.

```
Internet Protocol Headers
    Version: 4
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xb926 (correct)
    Source: 10.10.10.10
    Destination: 20.20.20.20
Transmission Control Protocol Headers

    Source port: 3086 (3086)
    Destination port: http (80)
    Sequence number: 713245119

    Header length: 28 bytes
    Flags: 0x0002 (SYN)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...0 .... = Acknowledgment: Not set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        .... ...0 = Fin: Not set
```

Figure 2–7 The SYN packet sent by the client.

When the server replies, both the SYN and ACK flags are set in the TCP headers to indicate that the server acknowledges the client's connection request. To ensure that each packet can be accounted for, the server will set an acknowledgment number that is equal to the last byte received from the client, relative to the starting sequence number, plus one. In our example, the client started with a sequence number of 713245119 and transmitted no user data, meaning that the server will use an acknowledgment of 713245120.

It is also important to notice the change in source and destination ports depending on which way a particular packet is directed. In our example, the client sends on port 80 and listens on port 3086, whereas the server sends on port 3086 and listens on port 80. Figure 2–8 shows the return packet from the server to the client.

The final packet exchanged during this handshake period is an acknowledgment from the client to the server. This allows the client to correctly acknowledge the sequence numbering used by the server in the previous packet and remove the SYN flag being used to show the start of the session. Once this final

```
Internet Protocol Headers
    Version: 4
    Time to live: 114
    Protocol: TCP (0x06)
    Header checksum: 0x9889 (correct)
    Source: 20.20.20.20
    Destination: 10.10.10.10
Transmission Control Protocol Headers

    Source port: http (80)
    Destination port: 3086 (3086)
    Sequence number: 20072
    Acknowledgement number: 713245120

    Header length: 28 bytes
    Flags: 0x0012 (SYN, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        .... ...0 = Fin: Not set
```

Figure 2–8 The SYN-ACK packet sent by the server.

packet of the handshake has been received, both sides of the connection can move into the *established* state, indicating that the transfer of user or application data can now commence. Figure 2–9 shows this final packet of the handshake. Note that in our example, the client has changed the acknowledgment numbering to match that initiated by the server and has also removed the SYN flag in the TCP header.

2. Data Transfer

Once the connection has moved into the *established* state, data transmission can begin between the two end points. During this state, the ACK flag is always set and the two end stations use the sequence and acknowledgment numbering to track the successful delivery of each segment of data. TCP also employs a number of windowing and buffering techniques to ensure the optimal delivery, retransmission, and buffering of data during this state. The discussions of such techniques are outside of the scope of this book.

```
Internet Protocol Headers
    Version: 4
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xb92c (correct)
    Source: 192.168.254.201 (192.168.254.201)
    Destination: 212.58.226.40 (212.58.226.40)
Transmission Control Protocol Headers

    Source port: 3086 (3086)
    Destination port: http (80)
    Sequence number: 713245120
    Acknowledgement number: 20073

    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Set
        .... ...0 = Fin: Not set
```

Figure 2–9 The final ACK packet of the handshake.

3. Terminating a Session

Unlike the session initiation, the termination of a TCP connection can be initiated from either side. Once one side of the connection decides that it has no more data to transmit, it will set the FIN flag in the TCP header to indicate to the other side that it is ready to terminate the connection. In simple terms, the receiving station will then acknowledge the FIN, by setting the ACK flag, and set its own FIN flag to show that it too is ready to terminate the connection. This series of exchanges results in both sides moving through the *TIME WAIT* state to the *CLOSED* state and the connection is closed.

In some instances, when the client receives the FIN it might still have data to send, in which case it will issue only an ACK back to the closing station. This allows the client to continue sending data until it is complete and then issue a FIN to show that the termination of the session can commence. During this period, the initiator and recipient of the initial FIN are referred to as being in the *FIN WAIT 2* and *CLOSE WAIT* states, respectively. Some applications, such as Web browsers, will often use this type of exchange to leave the connection in a type of half-closed state, thereby allowing the connection to be brought back into use when needed without having to reinitiate the entire connection (see Figure 2–10).

A more detailed description of TCP can be found in RFC 793.

User Datagram Protocol (UDP)

The User Datagram Protocol, or UDP, is the other most commonly used Transport layer protocol found within the Internet. While TCP is designed to provide *connection-oriented* delivery of packets, UDP implements a *connectionless* or *unguaranteed* delivery mechanism that is suitable for a number of upper layer applications. For some applications, the overhead of TCP, such as handshaking, is not required and for these, UDP is best suited.

A comparison between TCP and UDP can be drawn from the world of cellular phones. TCP is similar in nature to a full telephone conversation, whereby you establish a connection to the receiving station by dialing their number, hold a conversation with them using verbal interaction and acknowledgments, and finally terminate the call. UDP is much more akin to SMS or text messaging,

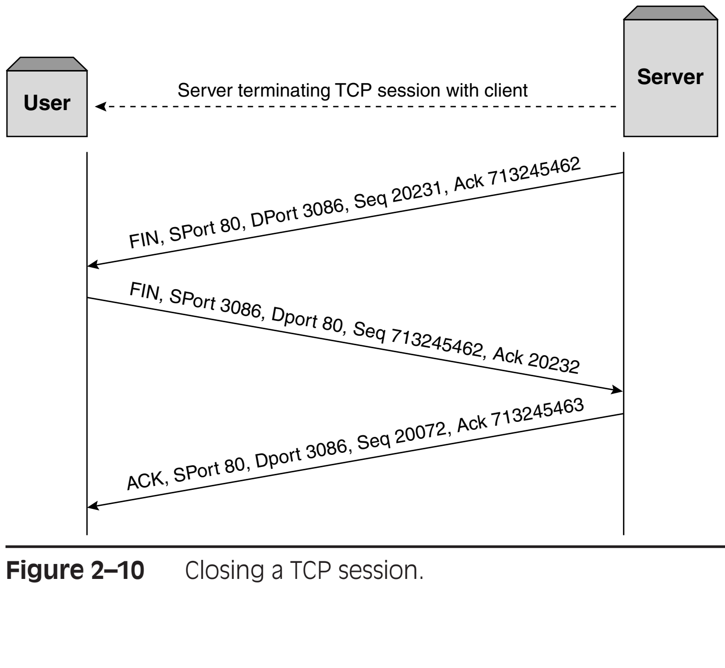
User
Server
Server terminating TCP session with client
FIN, SPort 80, DPort 3086, Seq 20231, Ack 713245462
FIN, SPort 3086, Dport 80, Seq 713245462, Ack 20232
ACK, SPort 80, Dport 3086, Seq 20072, Ack 713245463

Figure 2–10 Closing a TCP session.

whereby you write a message and send it without receiving any acknowledgment of its delivery from anything other than a local call access point.

UDP does share some common characteristics with TCP, as it does implement source and destination ports, to identify application sockets, and a checksum to verify the correct delivery of the layer 4 datagram.

A Simple UDP Data Flow

Let's consider our two example machines again, but this time interacting using UDP rather than TCP. The Domain Name System, or DNS, is one of the most commonly implemented UDP-based applications—our example will consider a client (10.10.10.10) requesting a name resolution from a DNS server (20.20.20.20).

It is important to note that UDP traffic can be both bidirectional, such as the request-response nature of DNS queries, and unidirectional, such as alerts raised through the Simple Network Management Protocol (SNMP). In both instances, the nature of the application determines whether a response is required; UDP simply provides a datagram format for the data between the two end points.

The Request

The first thing you will notice in Figure 2–11 is that the structure of the UDP header is far simpler than that used by TCP. There are only four fields used within the UDP header, to indicate the source and destination ports, the header length, and the checksum. It is clear from this that many of the techniques used by TCP are simply not present in UDP, such as sequencing, handshaking, and flow control.

The Response

As DNS is a bidirectional, request-response application, the frame shown in Figure 2–11 will yield an answer from the DNS server, also carried using UDP. Figure 2–12 shows the response. Note that the source and destination ports are reversed as with TCP, as the client sending the request will be listening and expecting an answer on port 1763.

```
Internet Protocol Headers
    Version: 4
    Time to live: 249
    Protocol: UDP (0x11)
    Header checksum: 0xc8de (correct)
    Source: 20.20.20.20
    Destination: 10.10.10.10
Transmission Control Protocol Headers

    Source port: domain (53)
    Destination port: 1763 (1763)
    Length: 276
    Checksum: 0x04bc (correct)

Domain Name System (response)
    Answers
        www.foo.com: type A, class inet, addr 1.2.3.4
            Name: www.foo.com
            Type: Host address
            Class: inet
            Time to live: 10 minutes
            Data length: 4
            Addr: 1.2.3.4
```

Figure 2–11 A UDP-based DNS query.

This is again a very brief overview of the UDP protocol. A more detailed description is available in RFC 768, available on the IETF Web site.

Virtual Router Redundancy Protocol (VRRP)

The Virtual Router Redundancy Protocol, or VRRP, is inextricably linked with the implementation of content switching, not because it is used by user applications, but because it provides a mechanism to eliminate single points of failure within content switching topologies. VRRP provides a mechanism to group two or more IP addresses, typically representing a routed interface, and make them appear to all surrounding devices as a single logical IP address.

Many of the topologies described later in this book will show how multiple content switches, and other routers, can be deployed to ensure a resilient and fault-tolerant implementation. For this reason, we need to examine the concepts and theory of VRRP in some more detail.

```
Internet Protocol Headers
    Version: 4
    Time to live: 249
    Protocol: UDP (0x11)
    Header checksum: 0xc8de (correct)
    Source: 20.20.20.20
    Destination: 10.10.10.10
Transmission Control Protocol Headers

    Source port: domain (53)
    Destination port: 1763 (1763)
    Length: 276
    Checksum: 0x04bc (correct)

Domain Name System (response)
    Answers
        www.foo.com: type A, class inet, addr 1.2.3.4
            Name: www.foo.com
            Type: Host address
            Class: inet
            Time to live: 10 minutes
            Data length: 4
            Addr: 1.2.3.4
```

Figure 2–12 The UDP-based DNS response.

Layer 2 and 3 Redundancy

Let's consider a network as shown in Figure 2–13. To eliminate a single point of failure for clients on the network accessing the Internet, the network administrator might consider deploying two Internet facing routers, R1 and R2. The client PC on the network will have been configured with a default route; for example, 10.10.10.2 pointing to router R1.

This "hard-coding" of the default gateway IP address into the client's TCP/IP settings presents the network administrator with two challenges when considering resilience:

- Router R1 might fail, leaving the client with a default gateway of an unreachable IP address.
- The client PC will resolve the IP address of the default gateway to the Ethernet address of router R1. This means that even if we replace the hardware of router R1, the client will still not have access to the Internet until its ARP cache has timed out or has been cleared.

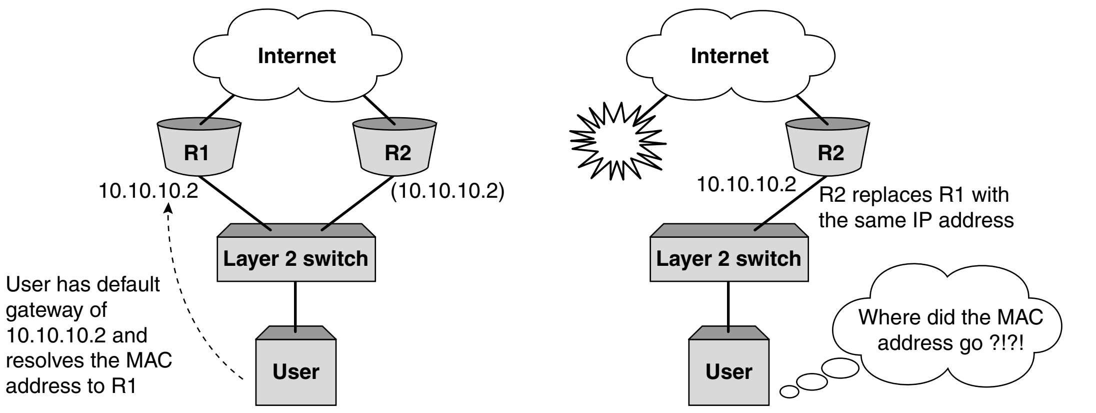
Internet
R1
R2
10.10.10.2
(10.10.10.2)
Layer 2 switch
User has default gateway of 10.10.10.2 and resolves the MAC address to R1
User
Internet
R2
10.10.10.2
R2 replaces R1 with the same IP address
Layer 2 switch
Where did the MAC address go ?!?!
User

Figure 2–13 Example network without VRRP.

It is for these reasons that we need VRRP to provide resilience at both Layer 2, by providing a virtual MAC address, and at Layer 3, by providing a virtual IP address. This virtualization of addresses amongst two or more physical units means that the client or client router will always have a default gateway both in terms of MAC address and IP address.

The Components of VRRP

RFC 2338 defines the following component parts in a network running VRRP:

- **VRRP router**: A router running VRRP. It can participate in one or more virtual routers.
- **Virtual router**: An abstract object managed by VRRP that acts as a default router for hosts on a shared LAN. It consists of a virtual router identifier (VRID) and a set of associated IP address(es) across a common LAN. A VRRP router can back up one or more virtual routers.
- **IP address owner**: The VRRP router that has the virtual router's IP address(es) as real interface address(es). This is the router that, when up, will respond to packets addressed to one of these IP addresses for ICMP pings, TCP connections, and so forth. Other routers that do not have an IP interface equal to the virtual IP address are commonly referred to as an *IP address renter*.
- **Primary IP address**: An IP address selected from the set of real interface addresses. One possible selection algorithm is to always select the first

address. VRRP advertisements are always sent using the primary IP address as the source of the IP packet.

- **Virtual router master**: The VRRP router that is assuming the responsibility of forwarding packets sent to the IP address(es) associated with the virtual router, and answering ARP requests for these IP addresses. Note that if the IP address owner is available, it will always become the master.
- **Virtual router backup**: The set of VRRP routers available to assume forwarding responsibility for a virtual router should the current master fail.
- **VRID**: Configured item in the range 1–255 (decimal). There is no default.
- **Priority**: Priority value to be used by this VRRP router in master election for this virtual router. The value of 255 (decimal) is reserved for the router that owns the IP addresses associated with the virtual router. The value of 0 (zero) is reserved for the master router to indicate that it is releasing responsibility for the virtual router. The range 1–254 (decimal) is available for VRRP routers backing up the virtual router. The default value is 100 (decimal).

VRRP Addressing

Let's take our previous example and expand it now to include VRRP on the two routers, R1 and R2. Assuming that router R1 is configured with the IP address that matches the proposed VRRP address, it will become the *VRRP master* and *VRRP owner*. Router R2 will become the *VRRP backup*.

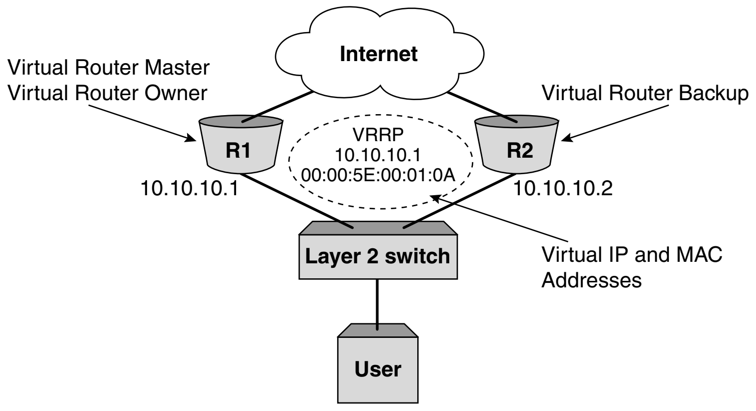

Figure 2–14 VRRP addressing example topology.

The IP address of router 1, 10.10.10.1, is also configured to be the VRRP router address, and this will be used by all clients on the network as a default route to the Internet. If router R1 was to fail, router R2 would take over while preserving the IP address to the clients. To manufacture a resilient Layer 2 MAC address, the following standard is used:

```
00:00:5E:00:01:[VRID]
```

where the virtual router ID is used to make the last byte of the MAC address. In our example, let's assume that a VRID of 10 has been used, giving us the VRRP MAC address of 00:00:5E:00:01:0A. Figure 2–14 shows our implementation with the new VRRP addressing.

VRRP Operation

Now that we have all of the component parts in place, let's look at how the routers operate together to provide a resilient pair. VRRP uses advertisement messages between all participating routers to indicate the health and availability of the current virtual router master. These messages are exchanged using a common multicast destination address of 224.0.0.18, and it is to this address that the current master router will continually advertise to indicate that it is still operational on the network.

In our example topology, during normal operation, router R1 will continually advertise the virtual router ID, the virtual router address, and its priority inside the multicast frame. The source IP address on these advertisements will be the interface on router R1 along with a source MAC address of the virtual MAC address we calculated earlier. The use of this virtual MAC address in these advertisements allows any Layer 2 infrastructure surrounding the VRRP routers—typically Layer 2 switches—to source learn where the common MAC address is currently located.

Now for the interesting part, a router failure. Let's imagine that router R1 experiences a power failure and effectively disappears from the network. In this instance, the following series of events would occur:

1. The master router, R1, would cease sending multicast packets advertising the virtual router.
2. After several missed packets, the standby router, R2, will acknowledge this occurrence by commencing with its own multicast advertisements.

When it does, it will use a source MAC address of the VRRP virtual MAC address, thus informing the attached Layer 2 switch that the MAC address has moved ports.

3. Since the virtual IP address and associated virtual MAC address have now survived the failure of router R1, the client will notice only minimal disruption during the re-election. This period is dependent on the configurable parameters associated with the advertisement intervals, but should typically be no more than 2 to 3 seconds.

VRRP, or variations on it, is commonly implemented in many content switching platforms, and as such it forms an important part of any implementation. More information about VRRP can be found in RFC 2338.

Summary

Many books have been written on the TCP/IP protocol stack and the higher layer applications such as HTTP and FTP that it supports. While it is outside the scope of this book to cover all the details and caveats, this chapter provided sufficient overview of the workings most relevant to content switching. Understanding the concept of a user session—being the total user experience of interacting over a period of time with a resource—and how that maps down the OSI seven-layer model and into the frames, packets, and TCP sessions below is key to understanding and successfully deploying content switching. In Chapter 3, *Understanding Application Layer Protocols*, we'll look at some of the Application layer protocols common to content switching.