Host-to-Host Protocols

by Sophia

WHAT'S COVERED

In this lesson, you will learn about host-to-host (OSI Layer 4) protocols and their functions.

Specifically, this lesson will cover the following:

1. Host-to-Host Layer Protocols

The main purpose of the host-to-host layer, which maps to the transport Layer 4 of the OSI model, is to shield the upper-layer applications from the complexities of the network. This layer acts as if to say to the upper layer, "Just give me your data stream, with any instructions, and I will begin the process of getting your information ready to be sent."

The following sections describe the two protocols at this layer:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

In addition, we will look at some of the key host-to-host protocol concepts as well as the port numbers.

1a. Transmission Control Protocol

Transmission Control Protocol (TCP) takes large blocks of information from an application and breaks them into **segments**, that is, segment headers and data sections, which contain 10 mandatory fields, and an optional extension field. It numbers and sequences each segment so that the destination's TCP process can put the segments back into the order the application intended. After these segments are sent, TCP (on the transmitting

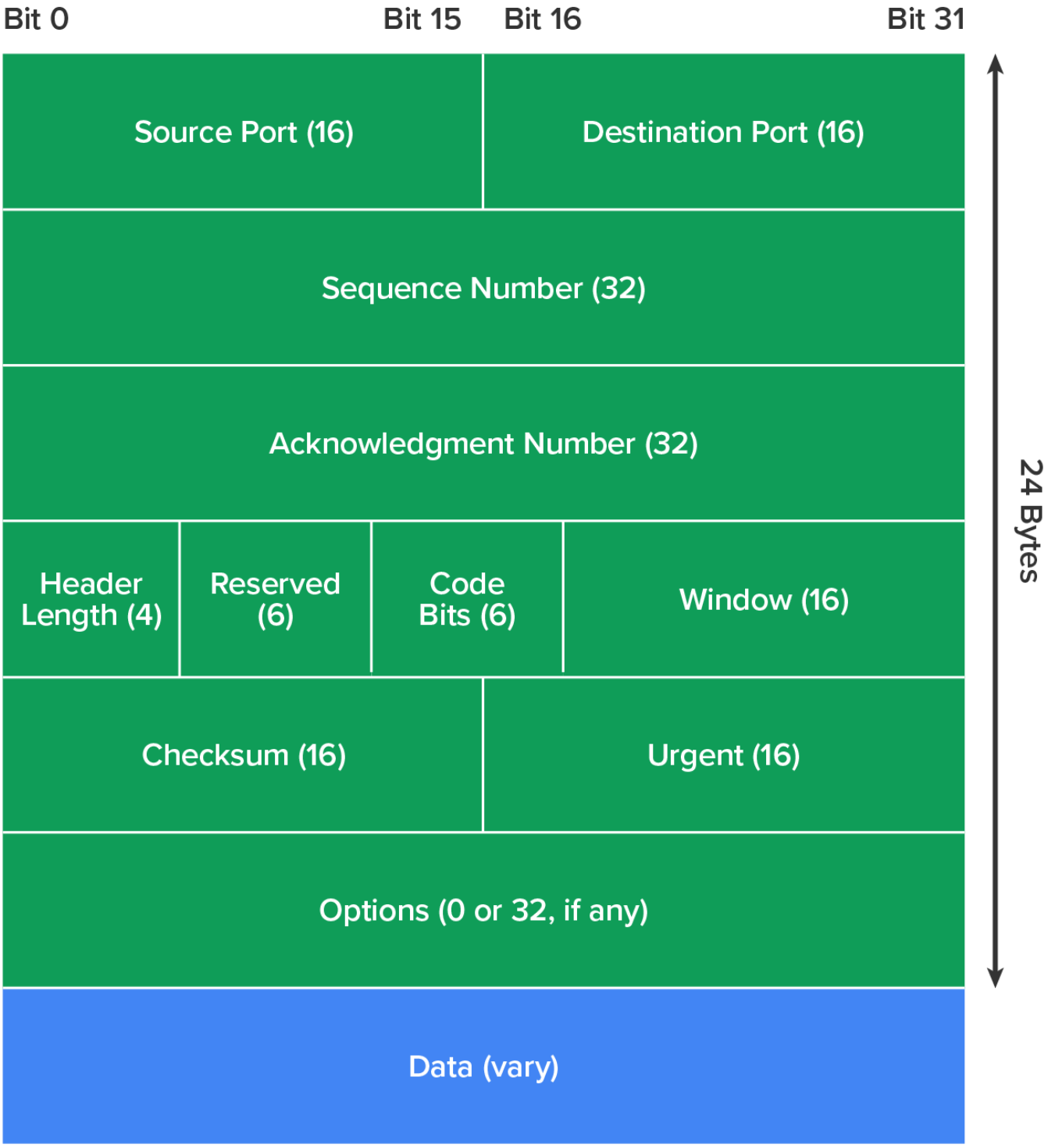
host) waits for an acknowledgment from the receiving end's TCP process, retransmitting those segments that have not been acknowledged.

KEY CONCEPT

Remember that in a reliable transport operation, a device that wants to transmit information sets up a connection-oriented communication with a remote device by creating a session. The transmitting device first establishes a connection-oriented session with its peer system; this session is called a **three-way handshake**. Data are then transferred; when the transfer is complete, a call termination takes place to tear down the virtual circuit.

TCP is a full-duplex, connection-oriented, and reliable protocol that guarantees delivery. TCP is complicated and uses significant network overhead. The application layer sends a data stream down to the protocols in the host-to-host layer, and TCP segments the data stream and prepares it for the internet layer. When the internet layer receives the data stream, it routes the segments as packets through an internetwork. The segments are handed over to the receiving host's host-to-host layer protocol, which rebuilds the data stream and hands it over to the upper-layer protocols.

The diagram below shows the TCP segment format, including the different fields within the TCP header (the first 24 bytes).



TERMS TO KNOW

Transmission Control Protocol (TCP)

A Layer 4 connection-oriented protocol.

Segments

Headers and data sections, which contain 10 mandatory fields, and an optional extension field.

Three-Way Handshake

A SYN, SYN ACK, ACK sequence that sets up a TCP session.

1b. User Datagram Protocol

If you were to compare **User Datagram Protocol (UDP)** with TCP, basically UDP would be the scaled-down economy model, which is sometimes referred to as a **thin protocol**. UDP does not offer the guarantee of delivery that TCP does, but it transports information that does not require reliable delivery using far fewer network resources.

HINT

In some situations, it would definitely be wise for developers to opt for UDP rather than TCP. Do you remember the watchdog SNMP in the application layer? SNMP monitors the network, sending intermittent messages and a fairly steady flow of status updates and alerts, especially when running on a large network. The overhead cost of establishing, maintaining, and closing a TCP connection for each one of those little messages will reduce network performance.

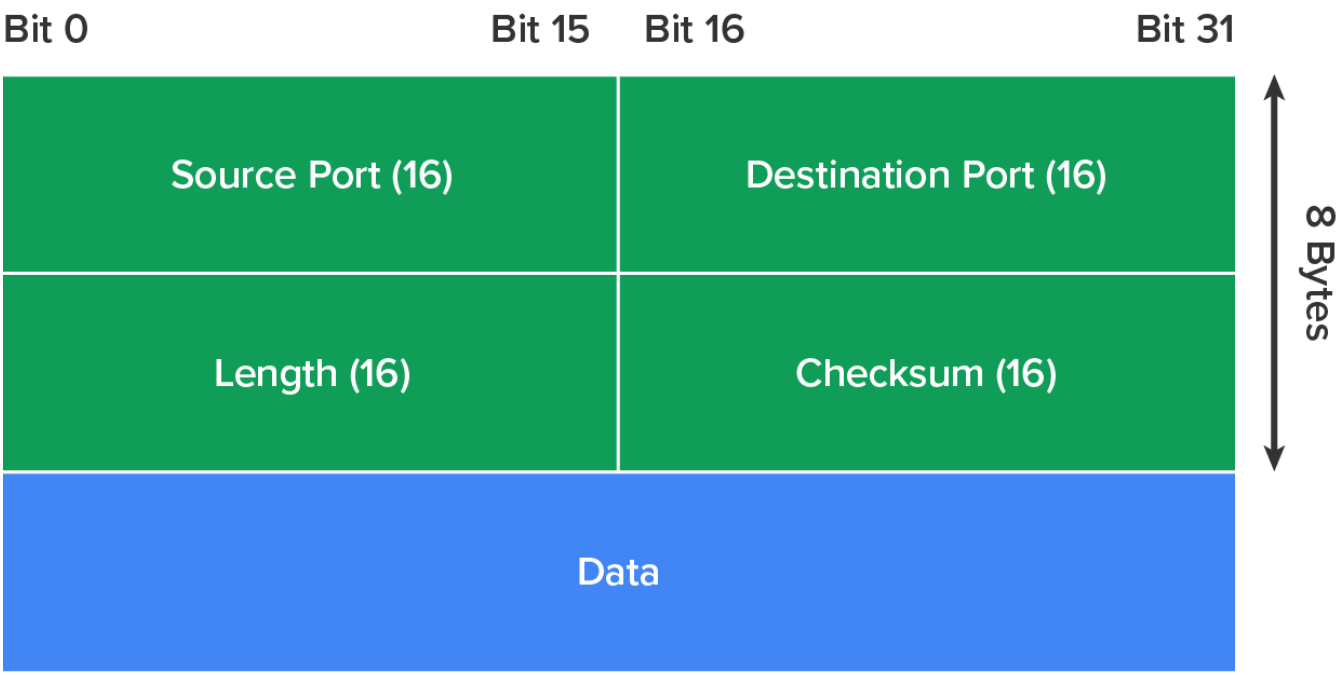
Another situation that calls for UDP over TCP is when reliability is already handled at the process/application layer. DNS handles its own reliability issues, making the use of TCP both impractical and redundant. But, ultimately, it is up to the application developer to decide whether to use UDP or TCP, not the user who wants to transfer data faster.

UDP does *not* sequence the segments and does not care about the order in which the segments arrive at the destination. But, after that, UDP sends the segments off and forgets about them. It does not follow through, check up on them, or even allow for an acknowledgment of safe arrival. Because of this, it is referred to as an **unreliable protocol**. This does not mean that UDP is ineffective, but that it does not handle issues of reliability. Because UDP assumes that the application will use its own reliability method, it does not use any. This gives an application developer a choice when running the IP stack: TCP for reliability or UDP for faster transfers.

KEY CONCEPT

UDP does not create a virtual circuit, nor does it contact the destination before delivering information to it. Because of this, it is also considered a **connectionless protocol**.

The diagram below illustrates UDP's low overhead as compared to TCP's usage. Look at the figure carefully, and you will see that UDP does not use windowing or provide for acknowledgments in the UDP header.



TERMS TO KNOW

User Datagram Protocol (UDP)

A Layer 4 connectionless protocol.

Thin Protocol

A protocol that uses relatively little bandwidth.

Unreliable Protocol

A protocol that does not guarantee delivery.

Connectionless Protocol

The communication between two network endpoints without a prior arrangement in which one network endpoint simply sends a message to the other.

2. Key Concepts of Host-to-Host Protocols

Now that you have seen both connection-oriented (TCP) and connectionless (UDP) protocols in action, it would be good to summarize them here. The table below highlights some of the key concepts that you should keep in mind regarding these two protocols. You should study this table well.

TCP	UDP
Sequenced	Unsequenced
Reliable	Unreliable
Connection-oriented	Connectionless
Virtual circuit	No virtual circuit
High overhead	Low overhead
Acknowledgments	No acknowledgment
Windowing flow control	No wondowing or flow control

EXAMPLE A post office analogy may help you understand how TCP and UDP work.

Using UDP is like sending a postcard. You simply write your message, address the postcard, and mail it. This is analogous to UDP's connectionless orientation. Because the message on the postcard is probably not a matter of life or death, you do not need an acknowledgment of its receipt. Similarly, UDP doesn't involve acknowledgments. Sending a postcard is inexpensive, because it does not require a lot of resources to deliver it. Likewise, UDP has low overhead, which means it does not use a lot of network resources like bandwidth and processing power.

Using TCP is like sending a letter via certified mail with a return receipt requested. The post office tracks the certified mail along each step of the transport and delivery process, and the return receipt acknowledges that the letter has been delivered. Likewise, TCP guarantees delivery by acknowledging the packets received, and the transmitting host resends any unacknowledged packets. TCP has high overhead, which means it uses a lot of network resources like bandwidth and processing power.

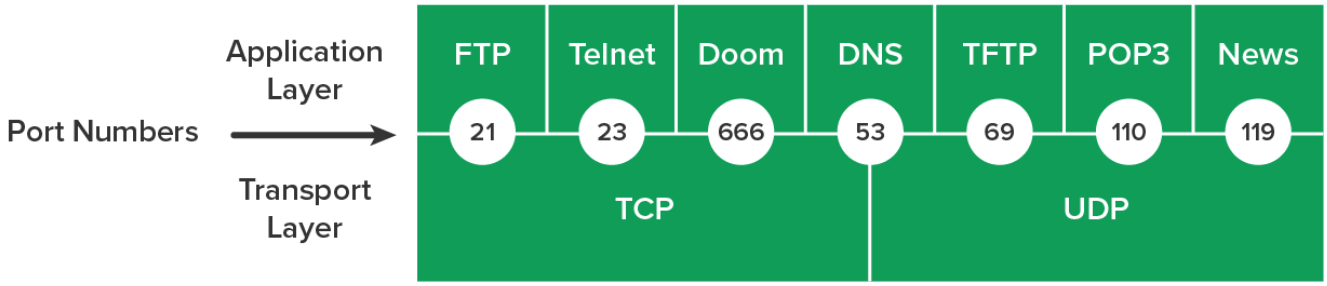
2a. Port Numbers

TCP and UDP use **port numbers** to communicate with the upper layers, because they keep track of different simultaneous conversations originated by or accepted by the local host. Originating source port numbers are

dynamically assigned by the source host and will usually have a value of 1024 or higher. Ports 1023 and below are defined in RFC 3232, which discusses what are called **well-known port numbers**.

Virtual circuits that do not use an application with a well-known port number are assigned port numbers randomly from a specific range instead. These port numbers identify the source and destination application or process in the TCP segment.

The diagram below illustrates how both TCP and UDP use port numbers.



You just need to remember that numbers below 1024 are considered well-known port numbers and are defined in RFC 3232. Numbers 1024 and above are used by the upper layers to set up sessions with other hosts and by TCP as source and destination identifiers in the TCP segment.

> EXAMPLE The table below gives you a list of the typical applications used in the TCP/IP suite, their well-known port numbers, and the transport layer protocols used by each application or process.

Telnet 23	SNMPv1/2 161
SMTP 25	TFTP 69
HTTP 80	DNS 53
FTP 20, 21	BOOTPS/DHCP 67,68
SFTP 22	NTP 123
DNS 53	
HTTPS 443	
SSH 22	
SMB 445	
POP3 110	
IMAP4 143	
RDP 3389	
SNMPv3 161	

TERMS TO KNOW

Port Number

A logical transmission endpoint identified by a 16-bit binary number representing 0–65,536.

Well-Known Port Numbers

TCP or UDP port numbers 0–1023.

SUMMARY

In this lesson, you learned about **host-to-host (OSI Layer 4) protocols**, **key host-to-host concepts**, and their functions, including TCP, UDP, and their respective port numbers. In the next lesson, you will learn about Internet Protocol (IP).

Source: This content and supplemental material has been adapted from CompTIA Network+ Study Guide: Exam N10-007, 4th Edition. Source **Lammle: CompTIA Network+ Study Guide: Exam N10-007, 4th Edition - Instructor Companion Site (wiley.com)**

TERMS TO KNOW

Connectionless Protocol
The communication between two network endpoints without a prior arrangement in which one network endpoint simply sends a message to the other.

Port Number
A logical transmission endpoint identified by a 16-bit binary number representing 0–65,536.

Segments
Headers and data sections, which contain 10 mandatory fields, and an optional extension field.

Thin Protocol
A protocol that uses relatively little bandwidth.

Three-Way Handshake
A SYN, SYN ACK, ACK sequence that sets up a TCP session.

Transmission Control Protocol (TCP)
A Layer 4 connection-oriented protocol.

Unreliable Protocol
A protocol that does not guarantee delivery.

User Datagram Protocol (UDP)
A Layer 4 connectionless protocol.

Well-Known Port Numbers
TCP or UDP port numbers 0–1023.