

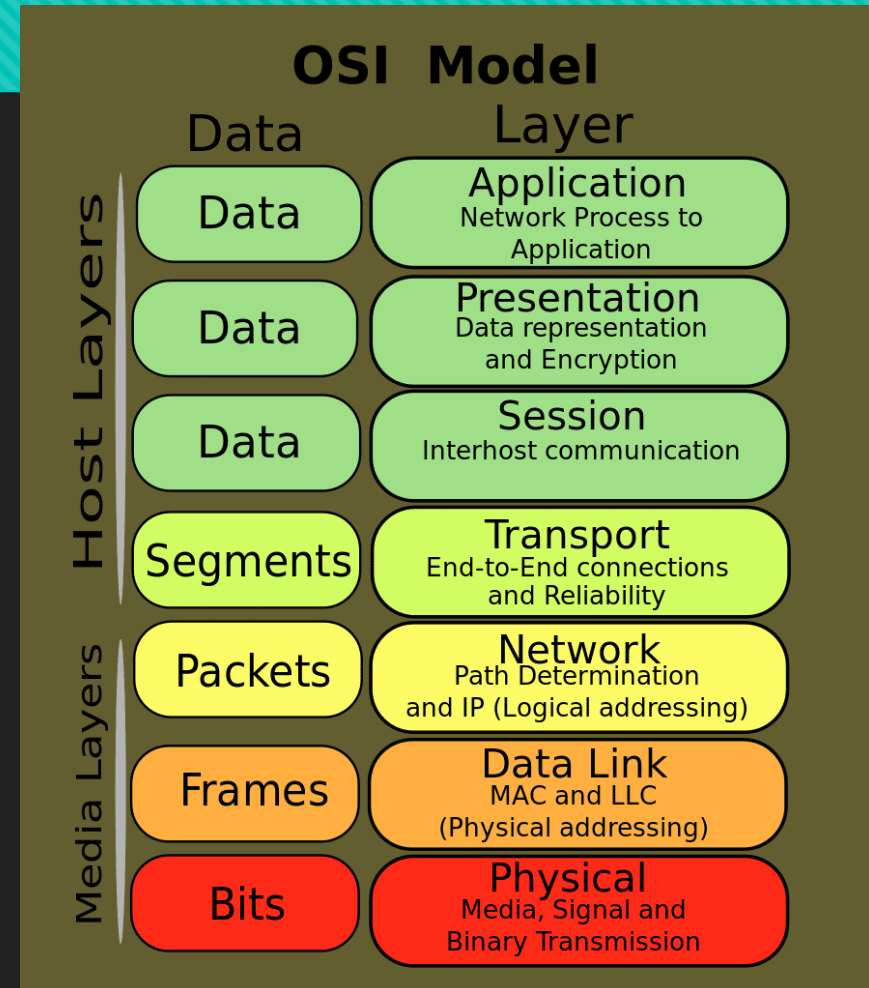
CO1507 Introduction To Networking Transport Layer Protocols

Jonathan Edwards

2024

OSI Model

- Previously we have focussed on the bottom 3 layers of the OSI model:
 - Physical layer
 - Wires, plugs, sockets, voltages, hubs
 - Data-link layer
 - MAC address, switches, data-frames
 - Network layer
 - IP address, routers, packets



'Other' layers

Presentation layer

- File formats

- Encryption

- Etc. not within the scope of this module

Session layer

- Starting, maintaining, and closing "conversations" between client and server

- Many relevant protocols

- Not within the scope of this module

Neither of these layers exist in the TCP/IP model

Transport Layer

- Transport layer is the fourth layer of the OSI model after Physical, data-link, and network layer
- The transport layer in the OSI model is responsible for providing end-to-end communication services for applications.
- It is responsible for segmenting and reassembling data, as well as ensuring reliable data transmission.
- There are two primary transport layer protocols:
 - Transmission Control Protocol (TCP)
 - User Datagram Protocol (UDP).

*Important to note **TCP** and **UDP** *both* part of the **TCP/IP** suite

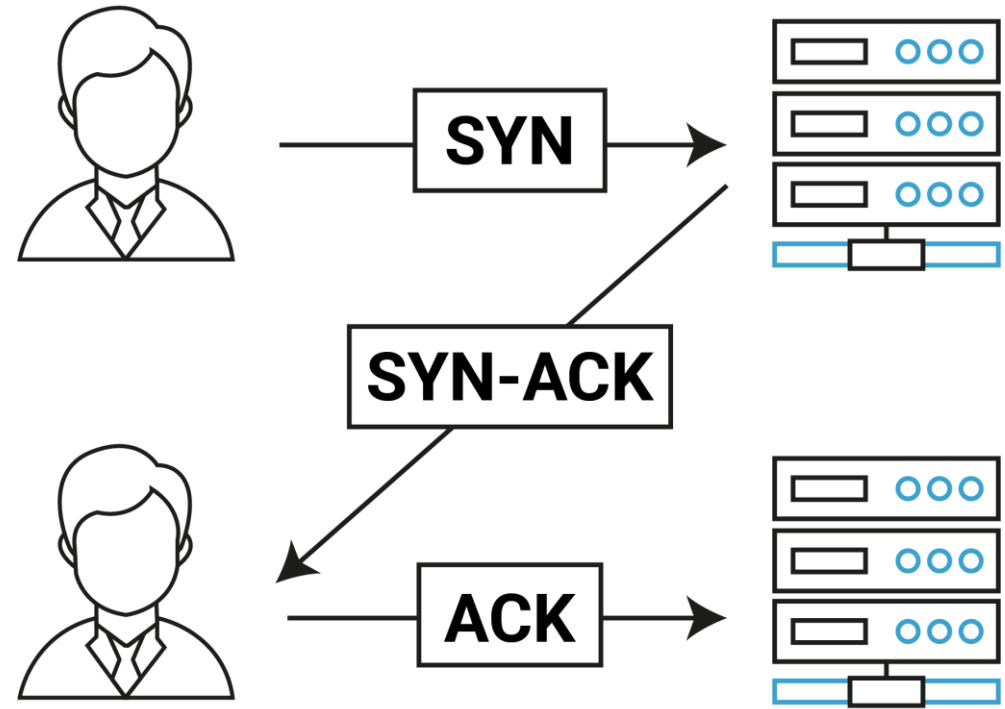
Transport Layer

- The transport layer accomplishes reliable communication by dividing the data received from the application layer into manageable chunks, called segments or datagrams, and adding information to these segments that enables the recipient to reassemble the data correctly.
- Additionally, the transport layer provides
 - Flow control
 - Congestion control
 - Error recovery mechanisms
- This ensures that data is delivered reliably and efficiently.
- It also handles multiplexing and demultiplexing of data streams, allowing multiple applications on a single host to share the same network connection.

TCP

- TCP is a connection-oriented protocol that provides reliable data delivery.
- It establishes a connection between the sender and receiver before transmitting data.
- It uses a three-way handshake process to establish a connection.
- TCP provides flow control, congestion control, and error detection and correction mechanisms.
- It guarantees the delivery of data in the correct order and without any loss or duplication.
- It is commonly used for applications such as file transfers, email, and web browsing.

3-Way Handshake



UDP

- UDP is a connectionless protocol that provides unreliable data delivery.
- It does not establish a connection before transmitting data.
- It is faster and simpler than TCP, but it does not guarantee reliable data delivery.
- Lost packets will NOT be detected or re-sent
- It is commonly used for applications such as video streaming, online gaming, and real-time communication.
- It does not provide flow control, congestion control, or error detection and correction mechanisms.
- It is more suitable for applications where speed is more important than reliability.

Other Transport Layer Protocols

- Other notable transport layer protocols include...
 - Stream Control Transmission Protocol (SCTP): a protocol designed for real-time communications, with features such as multi-homing and multi-streaming.
 - Datagram Congestion Control Protocol (DCCP): a protocol designed for applications that require congestion control, such as streaming media and voice over IP (VoIP)
 - Multipath TCP (MPTCP): a protocol that allows a single connection to use multiple paths simultaneously, providing better throughput and resilience to network failures

Error Detection

Parity Check:

- Parity check is a simple error detection technique that uses a single bit to detect errors.
- It involves adding an extra bit to each byte or word of data to indicate whether the number of 1s in the byte is even or odd.
- For example, if even parity is used and the data byte has an odd number of 1s, a parity bit of 1 is added to make the total number of 1s even.
- The receiver performs the same calculation and checks whether the parity bit matches the expected value.
- If the parity bit does not match, it indicates that an error has occurred during transmission.
- However, parity check can only detect errors and cannot correct them.

Error Detection

CRC Check:

- CRC is a method of detecting errors that occur during transmission of data in a network.
- It is a mathematical function that generates a checksum for a block of data and appends it to the message being transmitted.
- The receiver of the message performs the same calculation and compares the checksum generated by the sender with the one calculated by the receiver.
- If the two checksums do not match, it indicates that an error has occurred during transmission.
- The receiver can then request the sender to retransmit the data.

Error Correction

- TCP/IP, which is the protocol suite used for the Internet, provides error correction mechanisms primarily through the Transmission Control Protocol (TCP) at the transport layer.
- TCP uses a combination of techniques to provide reliable data delivery and error correction. These techniques include...
- Acknowledgment and retransmission:

After a segment of data is sent, TCP waits for an acknowledgement (ACK) from the receiver indicating that the data has been received successfully. If an ACK is not received within a certain time, the sender assumes that the data has been lost and retransmits it.

- Sequence numbers:

TCP numbers each segment of data so that it can be put back together in the correct order when it is received. If a segment is lost or arrives out of order, TCP can use the sequence numbers to detect the error and request retransmission.

Error Correction

- Windowing:

TCP uses a sliding window mechanism to control the flow of data and avoid overwhelming the receiver. The receiver sends back a window size indicating the amount of data it can receive at any given time, and the sender adjusts its transmission rate accordingly. This helps to prevent congestion and reduce the likelihood of errors.

- Checksums:

TCP uses a checksum mechanism to detect errors in the data. Before a segment of data is transmitted, TCP calculates a checksum based on the contents of the segment. When the segment is received, the checksum is recalculated, and if the two values do not match, TCP knows that the data has been corrupted and requests retransmission.