CIS 457 - Data Communications

Nathan Bowman

Images taken from Kurose and Ross book

Application/Transport Interface

When studying networks, we restrict ourselves to focusing on one layer at a time

However, we sometimes need to briefly take a look one layer up or down the protocol stack

In the case of the application layer, we need to know what services are provided by the transport layer

Recall that a distributed application is made of communicating processes

Quite simply, these processes need to

- do stuff
- communicate

The "doing" part is the application layer

From the application layer's view, the job of the tranport layer is to *send messages from one process to another on different hosts*

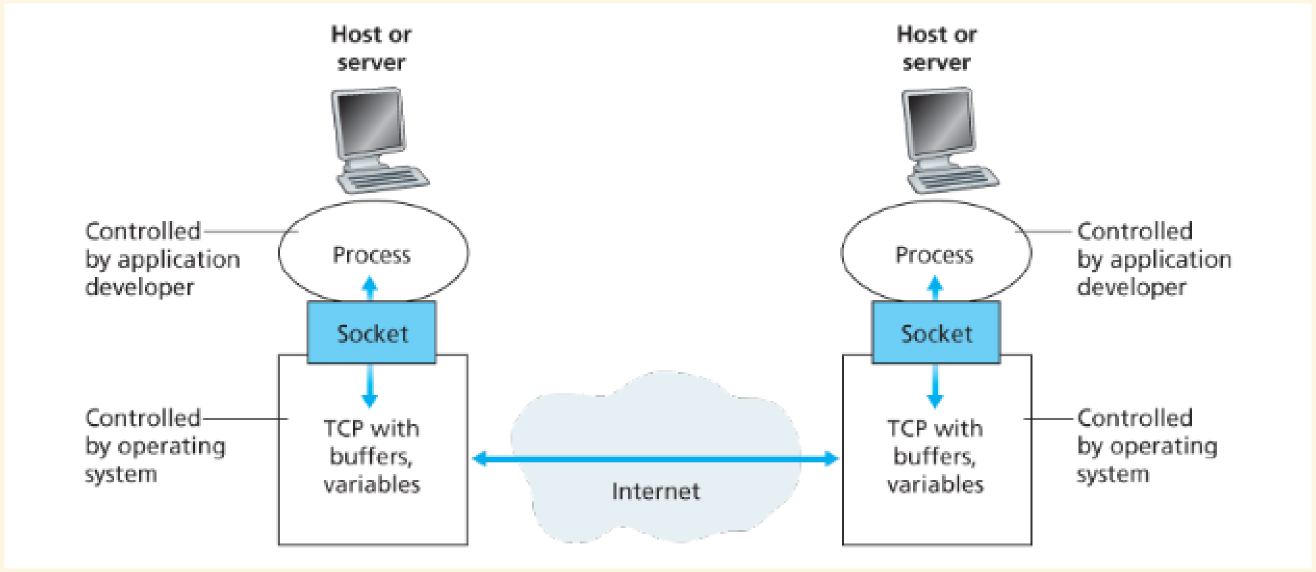
An interface between application and transport layers is called a **socket**

In the abstract, you can think of a socket as a place where the application layer sends a message for the transport layer to deliver

This resembles an API -- the application doesn't care how the message is sent, just that it is done

The receiving application also has a socket; otherwise it would need to know how to receive a transport-layer segment

We will see how to write programs to make use of sockets later


Host or server
Host or server
Controlled by application developer
Process
Process
Controlled by application developer
Socket
Socket
Controlled by operating system
TCP with buffers, variables
Internet
TCP with buffers, variables
Controlled by operating system

To send a message, we must know where it is going

Because messages are sent between *processes*, we need to specify two things to describe our destination:

- location of host running the process
- particular process on that host

Hosts are identified by **IP address**

Processes on a host are identified by **port number**

We will see soon how to get the IP address

For well-known applications, the port numbers are standardized (like much of the rest of the Internet)

For example, web servers by default listen on port 80 and mail servers on port 23

If my browser wants to get a webpage from some server whose IP address it knows, it must send a message to port 80

Sending a message to the correct host but wrong port will fail, because whatever program (if any) receives the message will not know what to do with it

Recapping so far:

- application must know ip address and port of receiving process
- passes that information, along with a message, to transport layer
- transport layer handles work of actually sending message

Last thing we need to consider is what "features" transport layer provides when sending that may be of interest to applications

On the Internet, there are two choices for transport-layer protocol: TCP and UDP

The main difference is whether they provide reliable data transfer to the application layer

TCP *does* provide reliable data transfer

Any message sent over TCP is guaranteed to arrive at the receiving application with all packets intact, unmodified, and in the correct order

How TCP does this is a topic for another day

TCP is also connection-oriented

TCP creates a connection between processes before any application-layer information is sent

From that point, from the perspective of the application, there is essentially a pipe running between the processes into which both processes can send messages at the same time

The connection exists until the processes are finished sending messages and it can be destroyed

UDP, on the other hand, provides none of this

UDP simply sends segments out into the network and hopes that they reach their destination

The segments may arrive out of order or not arrive at all

Transport layer could provide security, such as encryption and endpoint authentication, but it does not on the Internet

Because security is such a concern in modern distributed applications, application-layer solutions for security exist, which we discuss later

One such solution is Secure Sockets Layer (SSL), which functions much like a transport-layer protocol but operates at the application layer

Of note, no transport layer option on the internet provides guarantees about

- throughput or
- timing

When choosing a transport-layer protocol, why would an application designer ever choose UDP?

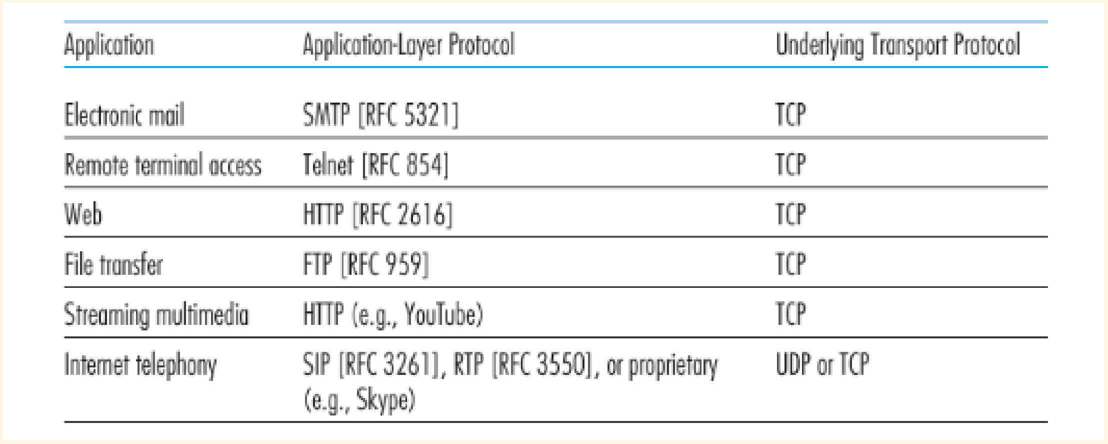
It has less overhead because it provides no features

Some applications, such as internet telephony, can tolerate a few lost packets, so the added overhead of TCP may not be worthwhile

That's all we need to know about the transport layer to start considering applications

The primary application-layer protocols we study will be HTTP (web), SMTP (email), and DNS (directory service)

The next table from your book shows several applications, their protocols, and which transport protocols are used to provide services for them



Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP