Cht2.

We have layers to 1. support evolution (easier to update), 2. also to divide the complexity of transmitting packets

A close-up of several words

Description automatically generated

Applications are built on host devices with either a client-server or peer-to-peer structure.

Client Server Architecture:

A diagram of a network

Description automatically generated

Multiple servers for Load Balancing.

A diagram of a computer network

Description automatically generated

Sockets:

-process sends/receives messages to/from its socket (aka its Application Programing Interface (API))

-Application developers can choose the transport protocol and fix a few transport layer parameters

Addressing Processes:

* to receive messages process must have identifier
* host device has unique 32-bit IP address

What transport service does an app need?

* data integrity (reliability) – some apps require 100% reliable data transfer others don’t
* timing (delay/latency)
* throughput
* Security

A screen shot of a computer

Description automatically generated

WEB AND HTTP:

* web pages consist of objects
* objects can be html file, jpeg image, java applet, audio file etc.

-receiver “advertises” free buffer space by including **rwnd** value in TCP header of receiver-to-sender segments

- **RcvBuffer** size set via socket options (typical default is 4096 bytes)

-sender limits amount of unpacked, (“in-flight) data to receiver’s **rwnd** value

-guarantees receive buffer will not overflow

Test Prep

TCP:

reliable package transport (in order)

UDP:

best effort

CRCs

ACks

Stop-and-wait: sender will send a package and wait until receive acknowledges before sending next. If nothing is sent back sender will send same package after a certain amount of time.