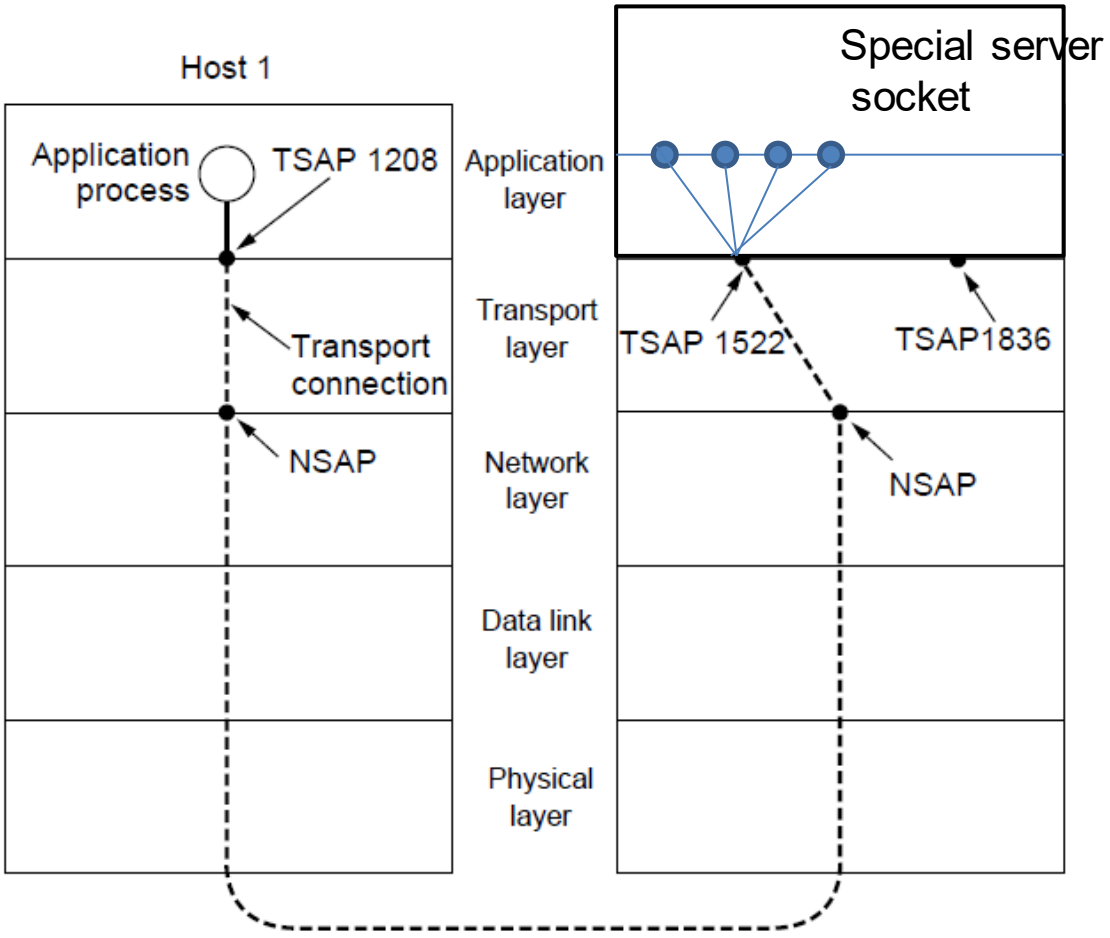
Transport Layer Contd

Internet Technologies
COMP90007

Sockets & Addressing

- Socket library provides a type of multiplexing tool on top of TSAPs to allow servers to service multiple clients



Port Allocations

- Port numbers can range from 0-65535
- Port numbers are regulated by IANA (http://www.iana.org/assignments/port-numbers)
- Ports are classified into 3 segments:
  - Well Known Ports (0-1023)
  - Registered Ports (1024-49151)
  - Dynamic Ports (49152-65535)

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Transport Layer Protocols: UDP

- The **most basic service** is actually connectionless:
  - Called: **User Datagram Protocol (UDP)**
  - Does **not add much to the Network Layer** functionality
  - TCP gives the real-deal for this layer, *reliability*…
  - For UDP: Just remove connection primitives to use it in a program
  - **UDP good for?:**
    - It is used for apps like video streaming/gaming regularly
  - **The reliability issue is left to?:**
    - the application layer… retransmission decisions as well as congestion control

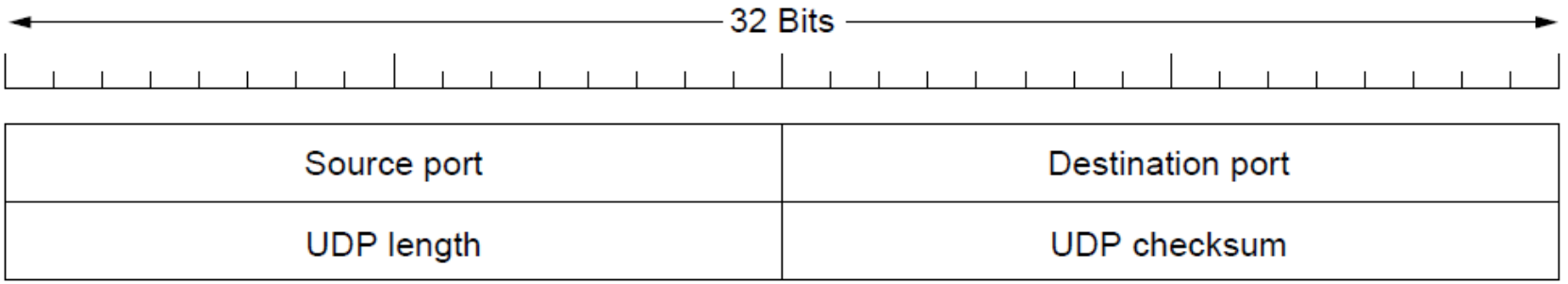
UDP Contd

- Provides a protocol whereby **applications can transmit encapsulated IP datagrams without a connection establishment**
- UDP transmits in segments consisting of an **8-byte header followed by the payload**
- UDP **headers contain source and destination ports**
- Payload is handed to the process which is attached to the particular port at destination

UDP Contd.

- Main advantage of using UDP over raw IP is:
  - the ability to specify ports for source and destination pairs, i.e., **addressing for processes**

- Both source and destination ports are required - destination allows for incoming segments, source allows reply for outgoing segments



Structure of UDP header: It has ports (TSAPs), length and checksum

Strengths and Weaknesses of UDP

- Strengths: provides an IP interface with multiplexing/de-multiplexing capabilities and related transmission efficiencies
- Weaknesses: UDP does not include support for: flow control, error control/retransmission of bad segments

- Conclusion: where applications require a precise level of control over packet flow/error/timing, UDP is a good choice as application layer can make choices

- **Domain Name System over the Internet is a famous user of UDP**

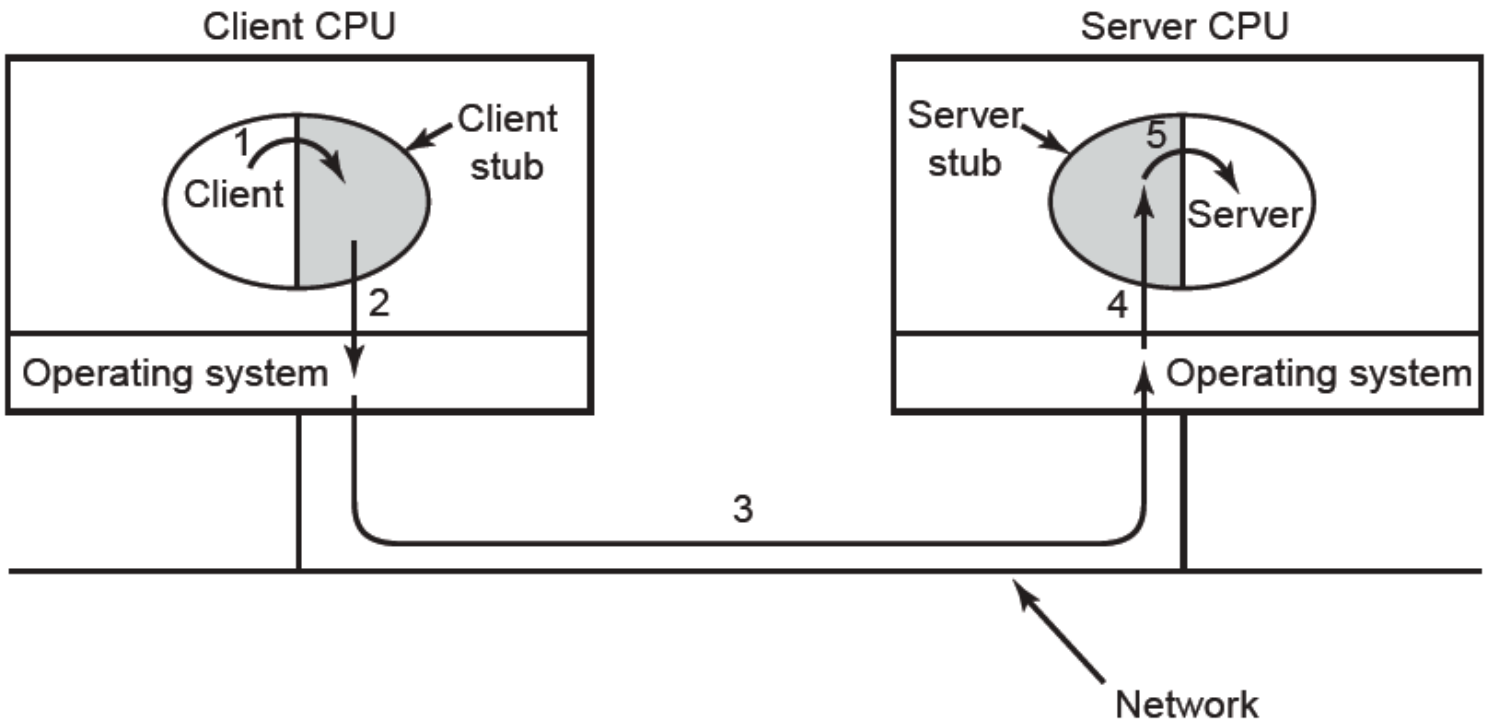
Another Application of UDP: Remote Procedure Call (RPC)

- Sending a message and getting a reply back is analogous to making a function call in programming languages
- Birrell and Nelson modified this to allow programs to call procedures on remote hosts using UDP
  - Remote Procedure Call (RPC)

Remote Procedure Call (RPC)

- To call a remote procedure, the client is bound to a small library (the client stub) that represents the server procedure in the client‘s address space.
- Similarly the server is bound with a procedure called the server stub.
- These **stubs hide the fact that the procedure itself is not local**.

RPC Illustrated



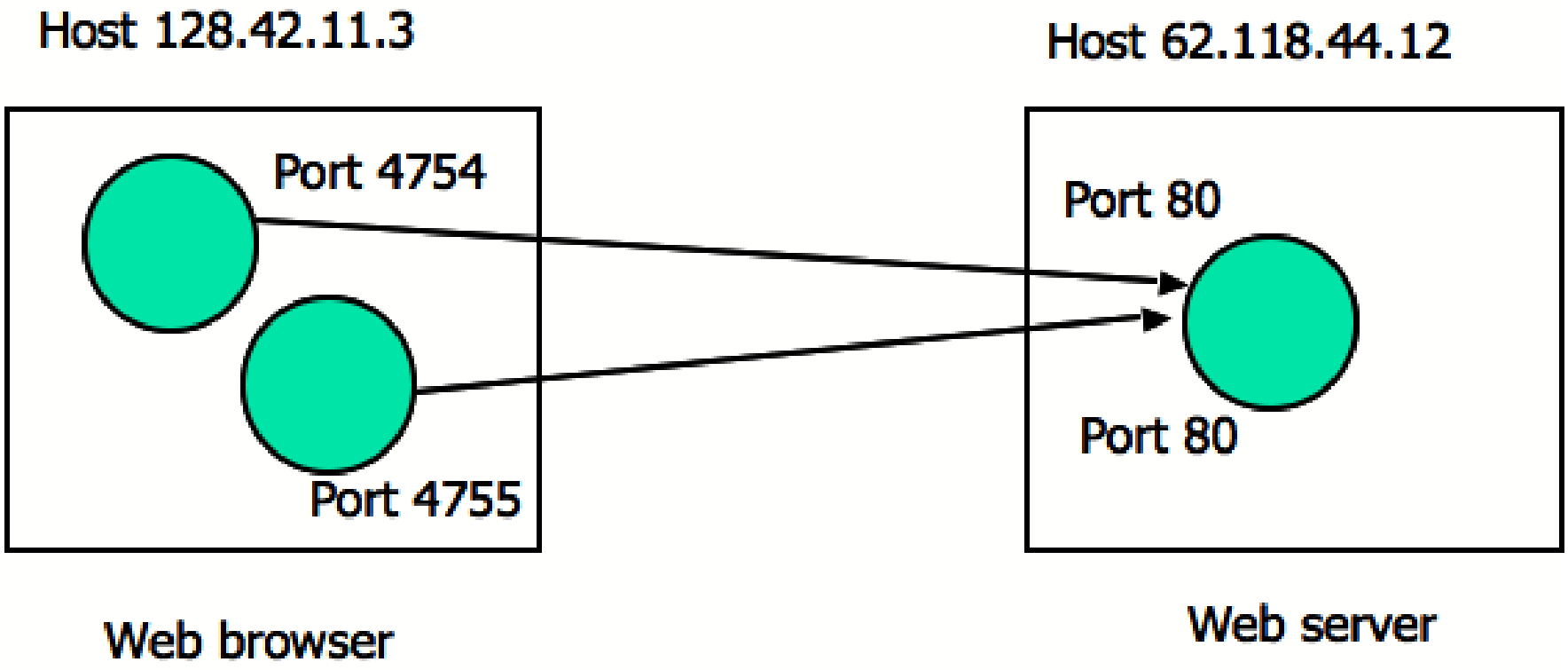
Transmission Control Protocol: TCP Details

- Provides a protocol by which applications can transmit datagrams within a **connection-oriented** framework, thus increasing reliability
- TCP transport entity manages TCP streams and interfaces to the IP layer - can exist in numerous locations (kernel, library, user process)
- **TCP entity** accepts user data streams, and **segments them into pieces < 64KB** (often at a size in order so that the IP and TCP headers can fit into a single Ethernet frame), and sends each piece as a separate datagram
- Recipient TCP entities reconstruct the original byte streams from the encapsulation

The TCP Service Model

- Sender and receiver both create **sockets**, consisting of the IP address of the host and a port number as we saw earlier
- For TCP Service to be activated, **connections must be explicitly established between a socket at a sending host** (src-host, src-port) and a socket at a receiving host (dest-host, dest-port)
- Special one-way server sockets may be used for multiple connections simultaneously

Example


Host 128.42.11.3
Host 62.118.44.12
Port 4754
Port 4755
Port 80
Port 80
Web browser
Web server

Features of TCP Connections

- TCP connections are:
- Full duplex - data in both directions simultaneously
- Point to point - exact pairs of senders and receivers
- Byte streams, not message streams - message boundaries are not preserved
- Buffer options - TCP entity can choose to buffer prior to sending or not depending on the context
  - **TCP_NODELAY in Java**
  - **Socket.setTcpNoDelay(boolean)**

TCP Contd

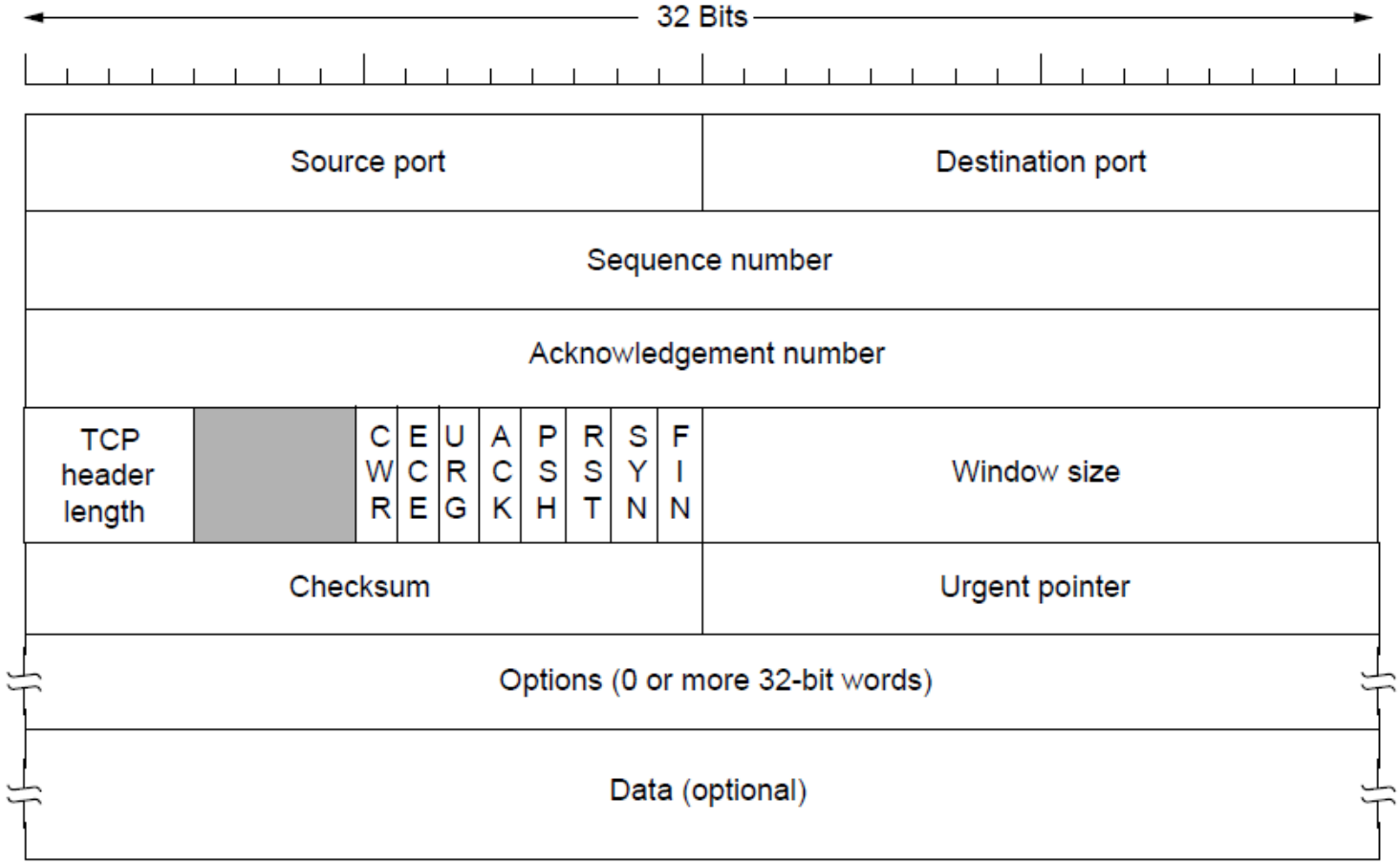
- Data sent between TCP entities in segments - segment has a **20 byte header plus zero or more data bytes**

- TCP entities decide how large segments should be mainly with 2 constraints:
  - **65,515 byte IP payload**
  - **Ethernet unit size - generally 1500 bytes**

- Sliding window - **sender transmits and starts a timer**
  - Receiver sends back an acknowledgement which is the next sequence number expected - if sender's timer expires before acknowledgement, then the sender **transmits the original segment again**

The TCP Segment Header

- TCP header includes addressing (ports), sliding window (seq. / ack. number), flow control (window), error control (checksum) and more



The TCP Segment Header

- ***Source port* and *Destination port*** fields identify the local end points of the connection
- ***Sequence number and Acknowledgement number*** fields perform their usual functions
- ***TCP header* length** tells how many 32-bit words are contained in the TCP header
- ***Window size*** field tells how many bytes may be sent starting at the byte acknowledged
- ***Checksum*** is also provided for extra reliability. It checksums the header, the data
- ***Options*** field provides a way to add extra facilities not covered by the regular header
- ***URG*** is set to 1 if the *Urgent pointer* is in use. The Urgent pointer is used to indicate a byte offset from the current sequence number at which urgent data are to be found

The TCP Segment Header

- ***CWR*** and ***ECE*** are used to signal congestion when *ECN* (Explicit Congestion Notification) is used
- ***ECE*** is set to signal an ECN-Echo to a TCP sender to tell it to slow down when the TCP receiver gets a congestion indication from the network
- ***CWR*** is set to signal Congestion Window Reduced from the TCP sender to the TCP receiver so that it knows the sender has slowed down and can stop sending the ECN-Echo
- The ***ACK*** bit is set to 1 to indicate that the Acknowledgement number is valid. This is the case for nearly all packets. 0 means ignore ACK number field
- ***PSH*** bit indicates PUSHed data. The receiver is hereby kindly requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received

The TCP Segment Header

- The ***RST*** bit is used to abruptly reset a connection that has become confused due to a host crash or some other reason. It is also used to reject an invalid segment or refuse an attempt to open a connection
- The ***SYN*** bit is used to establish connections. The connection request has SYN = 1 and ACK = 0. The connection reply does bear an acknowledgement, so it has SYN = 1 and ACK = 1.
- In essence, the SYN bit is used to **denote both CONNECTION REQUEST and CONNECTION ACCEPTED**, with the ACK bit used to distinguish between those two possibilities.
- The ***FIN*** bit is used to release a connection. It specifies that the sender has no more data to transmit. However, after closing a connection, the closing process may continue to receive data.