Transport Layer Contd

COMP90007

Internet Technologies

UDP

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

New Code: UDP Example Client…

```
public static void main(String args[]) {
        ….
        DatagramSocket mySocket = new
              DatagramSocket();
        mySocket.send([data,address, etc
              parameters]);
        …
}
```

Server Side: UDP Example

```
public static void main(String args[]) {
        ....
        DatagramSocket server = new
                DatagramSocket(port);
        while (true) {
                server.receive([parameters]);
                ...
        }}
```

[*Multi-threaded version can be done as show in the last lecture…*]

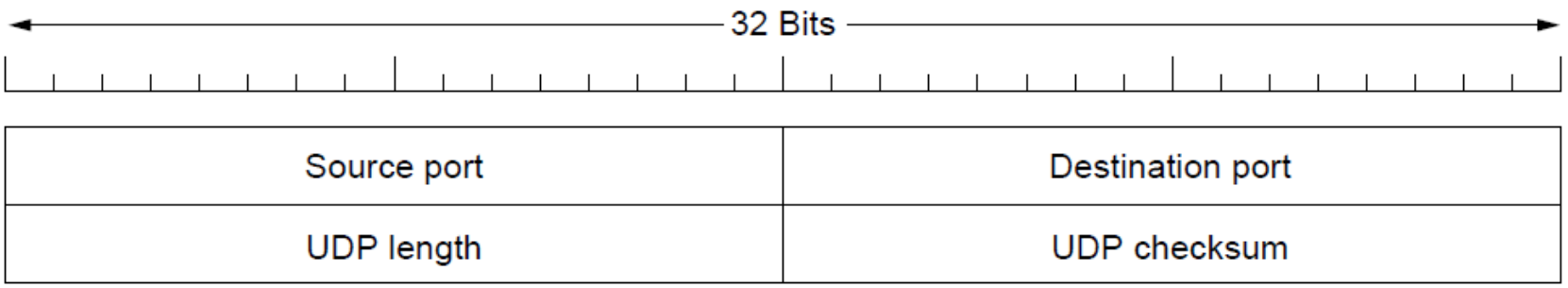
UDP

- 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, why?
  - destination allows for incoming segments, source allows **reply** for outgoing segments



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
  - Is this a real weakness or a choice?
- …where applications **require a precise level of control** over packet flow/error/timing, **UDP is a good choice** as application layer can make choices
- Example use: Domain Name System over the Internet is a famour user of UDP (we will visit DNS later…)

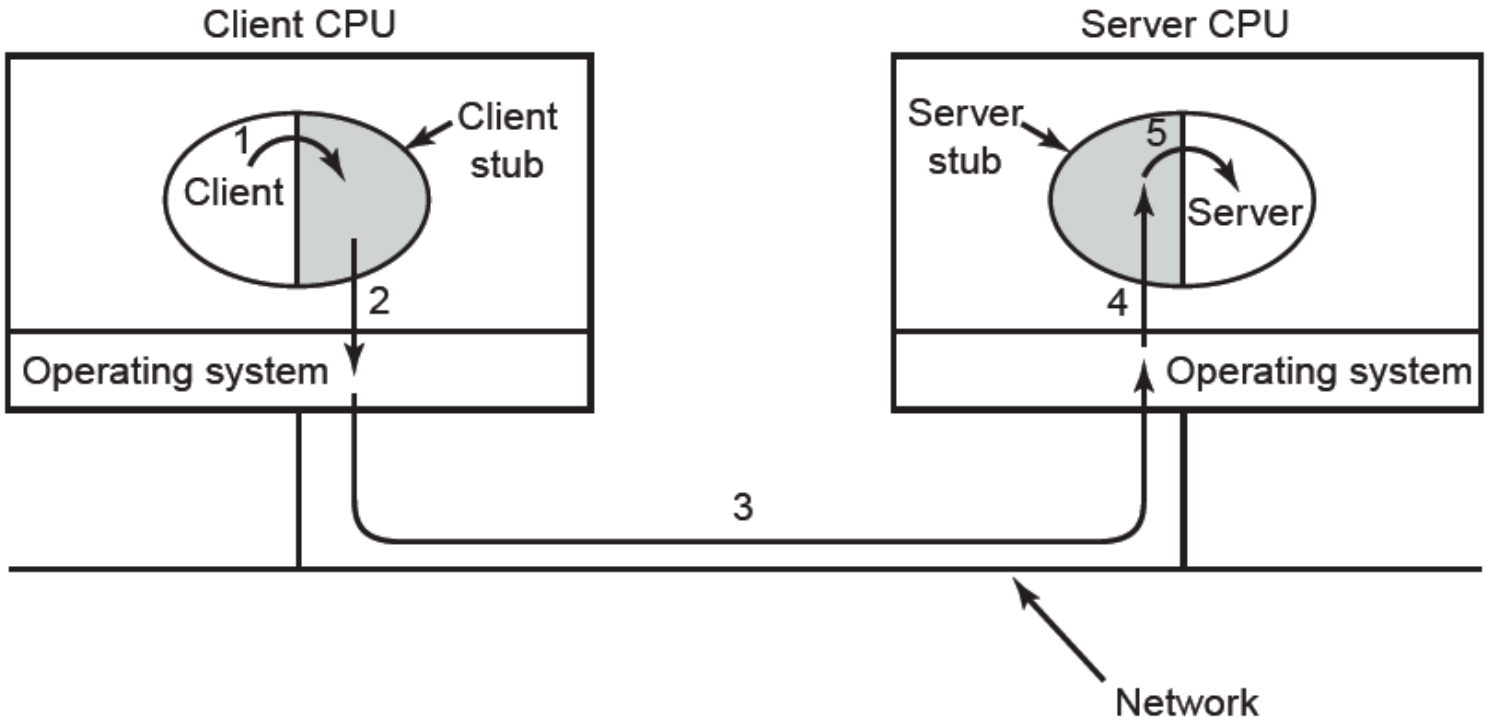
Using UDP: Remote Procedure Call (RPC)

- 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



Looking at Details of Internet: Transmission Control Protocol (TCP)

- Provides a protocol by which applications can **transmit IP datagrams** within a **connection-oriented** framework, thus increasing **reliability** and applications do not have to deal with basics of reliability

- TCP transport entity manages TCP streams and interfaces to the IP layer (can exist in numerous locations i.e., kernel, library, etc)

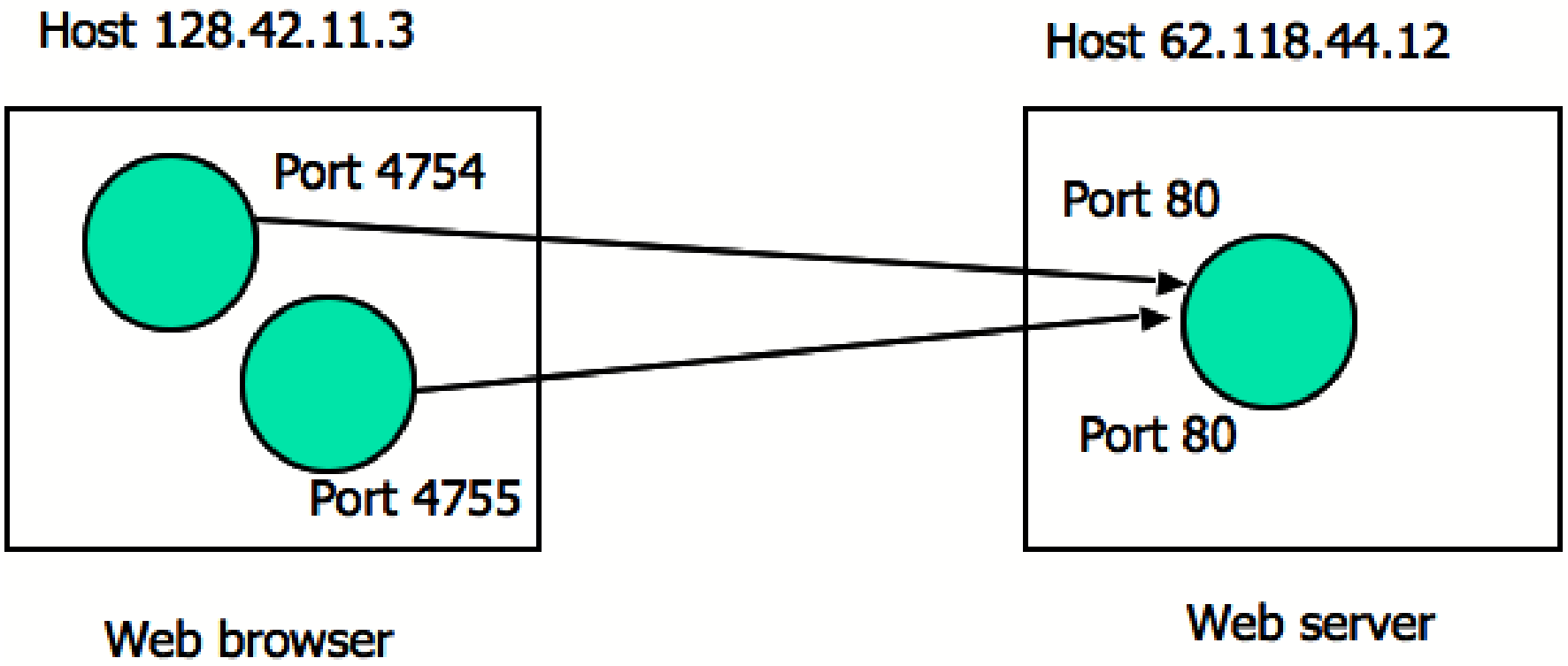
- **TCP entity** accepts user data streams, and **segments them into pieces < 64KB** (often at a size in order so that the IP and TCP headers can fit into a single Ethernet frame), and sends each piece as a separate IP datagram

- Recipient TCP entities reconstruct the byte stream…

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

Port Allocations

- Recall TSAPs
- 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

Socket Library - Multiplexing

- Socket library provides a multiplexing tool on top of TSAPs to allow servers to service multiple clients
- It **simulate** the server using a different port to connect back to the client