


Week 10 - Tuesday

Intro to Networks (cont.) & Network Programming


School of Information Technology and Electrical Engineering
The University of Queensland



Week 10

- Tuesday (and continued Friday)
 - Intro to Networks (cont.)
 - Network Programming
- Assignment 3
 - Due Saturday 16 May @ 11pm (plus any available grace days)
 - No submissions after 11pm Thursday 21 May
 - See course website for clarifications and bug fixes
 - Test script released


2



Outline

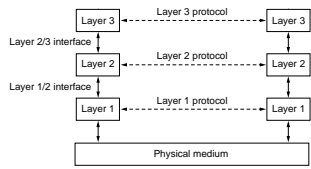
- internet Protocols
- IP, TCP, UDP
- Network APIs
 - Sockets
- C Network Programming
 - Examples to be covered in class
- Credits:
 - Glass and Ables, "UNIX for Programmers and Users"
 - Bryant and O'Halloran, "Computer Systems: A Programmer's Perspective"
 - Rochkind, "Advanced UNIX Programming"
 - Tanenbaum, "Computer Networks"

3




From last week... Protocol Hierarchies

- To reduce design complexity, most networks organized as series of layers or levels
- Purpose of layer
 - Offer **services** to higher layers
 - Abstraction
 - Shield higher layers from implementation details




4



Network Architecture

- Definition: A set of layers and protocols
- Must contain enough information to build
- Some protocol architectures do not specify interfaces, only protocols
 - In such cases interfaces can differ on different machines
- **Protocol Stack**
 - Set of protocols used by a certain system, one protocol per layer

5



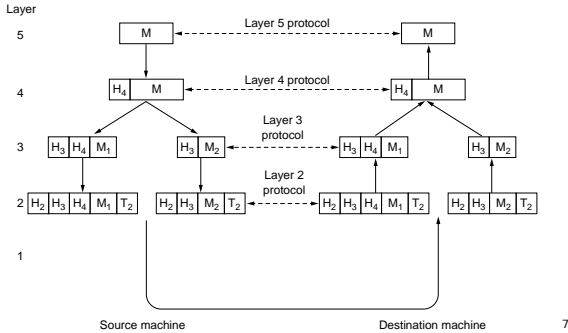
Headers and Enveloping

- Header
 - Additional information attached to message data by a protocol
- Protocol worries about header, not message
- Like placing a message in an envelope
- Layered protocols -> nested envelopes
- Some protocols might add a *tail* also

6

COMP2303
COMP7306

Headers and Enveloping (cont.)



7

COMP2303
COMP7306

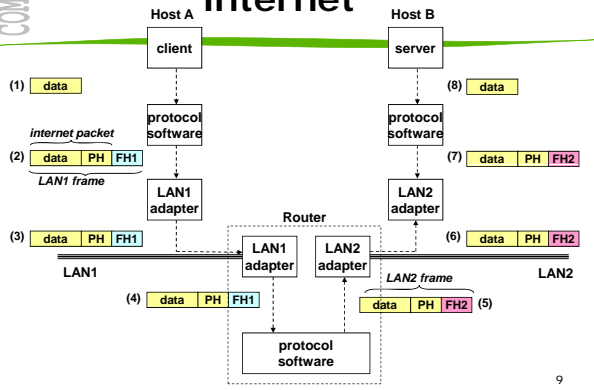
The Notion of an internet Protocol

- How is it possible to send bits across incompatible LANs and WANs?
 - Use an *internet protocol* (i.e., set of rules) that governs how hosts and routers should cooperate when they transfer data from network to network.
- Best example: **IP (Internet Protocol)**
- An internet protocol provides
 - An addressing scheme
 - e.g. IP addresses (130.102.72.5)
 - A delivery mechanism
 - e.g. IP packets (datagrams)

8

COMP2303
COMP7306

Transferring Data Over an internet



9

COMP2303
COMP7306

Global IP Internet

- Most famous example of an internet
- Based on the TCP/IP protocol family
 - IP (Internet protocol)**
 - Provides basic naming scheme and unreliable delivery capability of packets (datagrams) from host-to-host
 - UDP (Unreliable Datagram Protocol)**
 - Uses IP to provide unreliable datagram delivery from process-to-process
 - TCP (Transmission Control Protocol)**
 - Uses IP to provide reliable byte streams from process-to-process over connections
- Accessed via a mix of Unix file I/O and functions from the **sockets interface**

10

COMP2303
COMP7306

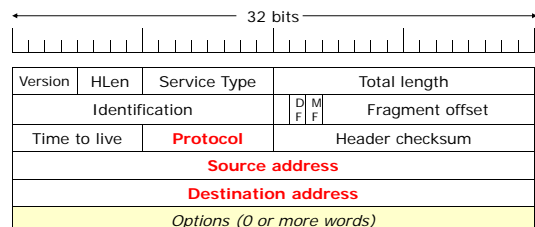
IP (Internet Protocol)

- Connectionless model of delivery
- Best-effort** delivery – no guarantees of success
- Packets**
 - Also called **datagrams**
 - Consists of
 - Header
 - 20 byte fixed part
 - 0-40 byte optional part
 - Body part ("text")

11

COMP2303
COMP7306

Internet Protocol (IP) Header



- We're interested in the **highlighted** header fields

12

COMP2303
COMP7306

IP Header: Protocol

- 8 bits
- Range: 0 to 255
- Identifies higher level protocol to which packet should be passed, e.g.
 - 6 = TCP
 - 17 = UDP
- RFC 1700 – defines protocol numbers

Version	HL	Service Type	Total Length
Identification		Flags/Offset	
Time to live	Protocol	Header checksum	
Source address			
Destination address			
Options (if any are present)			

13

COMP2303
COMP7306

IP Header: Source and Dest. Addresses

- 32 bits each
- Dest. address
 - Used to route packet to intended destination
- Source address
 - Destination can choose whether to receive
 - Destination knows whom to reply to

Version	HL	Service Type	Total Length
Identification		Flags/Offset	
Time to live	Protocol	Header checksum	
Source address			
Destination address			
Options (if any are present)			

14

COMP2303
COMP7306

IP Addresses

- 32-bit IP addresses are often written in **dotted-decimal** notation
 - each of 4 bytes written in decimal
 - e.g. 130.102.2.15
 - This notation used for human consumption
- Some addresses have special meanings
 - e.g., broadcast to all hosts on a particular network
 - More details in week 13

15

COMP2303
COMP7306

What does IP give us?

- Best-effort packet delivery
 - No guarantees ...
- Packets may be
 - dropped
 - reordered
 - duplicated
 - size-limited
 - delayed

16

COMP2303
COMP7306

Short Break

- Stand up and stretch

17

COMP2303
COMP7306

What do Applications Want?

- Not to have to worry about the underlying network (routers etc)
- Messages
 - transported from one host to another
 - guaranteed delivery
 - delivered in same order as sent
 - arbitrary size

18

COMP2303
COMP7306

What do Applications Want? (cont.)

19

COMP2303
COMP7306

Transport Protocols

- We'll look at two transport protocols
 - UDP = User Datagram Protocol
 - TCP = Transmission Control Protocol

20

COMP2303
COMP7306

UDP: User Datagram Protocol

- Connectionless Transport Protocol
- Allows applications to send datagrams to other applications
 - Basically an interface to IP
- No need to establish connection
- 8 byte header + data

21

COMP2303
COMP7306

Why would we want to use UDP?

- Don't care if packet lost
 - e.g. Streaming
- Small messages
 - No connection establishment overhead
- No congestion control
 - Can send things as fast as possible
- Simple implementation
 - No connection state information

22

COMP2303
COMP7306

UDP: Header

← 32 Bits →			
Source port		Destination port	
UDP length		UDP checksum	

- Ports
 - Identify end-points within machines
 - e.g. processes
 - Source port is optional
- UDP Length
 - Includes header
- UDP Checksum
 - For checking if corruption occurs
 - Optional – store 0

23

COMP2303
COMP7306

TCP: Transmission Control Protocol

- Connection-oriented
- Reliable (over unreliable IP internetwork)
- Byte-stream
 - Not message stream
- Full-duplex
- Point-to-point (end-to-end)
 - Not multicast or broadcast
- Each machine has a *TCP transport entity*
 - user process or part of kernel

24

COMP2303
COMP7306

TCP Responsibilities

- Sender
 - Accepts data streams from local processes
 - Breaks data into pieces < 64k bytes
 - Sends each piece as IP datagram
 - Time-out and retransmit if no acknowledgement received
- Receiver
 - IP datagrams containing TCP data are passed to TCP entity for reconstruction
 - Acknowledge receipt
 - Reassemble datagrams in proper sequence

25

COMP2303
COMP7306

TCP Service Model: Sockets

- Sender and Receiver create sockets
- Sockets have addresses (IDs)
 - IP address + port number
- Multiple connections possible through one socket
 - Connections identified by socket ID of each end

26

COMP2303
COMP7306

Port Numbers

- 16 bits: 0 – 65535
- Below 1024
 - Well known ports
 - Reserved for standard services, e.g.
 - 23 - Telnet
 - 21 - FTP
 - 80 - HTTP
 - Look in /etc/services on a UNIX box

27

COMP2303
COMP7306

An Explanation of Ports

- Figure to be drawn in class

28

COMP2303
COMP7306

TCP: Byte Stream

- Message boundaries are not preserved
- Example
 - Sending process writes four 512 byte chunks
 - Receiving process may get
 - one 2048 byte chunk (see below)
 - two 1024 byte chunks
 - many other possibilities
- TCP doesn't care what the bytes are

(a)

(b) 29

COMP2303
COMP7306

TCP Protocol

- TCP entities exchange **segments**
 - Data + 20 byte header, including:
 - Source & destination port numbers
 - Sequence number (for reordering)
 - Acknowledgements
- Size limited by
 - IP packet size
 - Network – maximum transfer unit (MTU)
- Segments with zero data are valid

30

COMP2303
COMP7306

TCP Connections

- Identified by
 - Source Port
 - Source IP address
 - Destination Port
 - Destination IP address

31

COMP2303
COMP7306

TCP/UDP and IP

- Remember: IP underlies both TCP and UDP
- Figure to be drawn in class

32

COMP2303
COMP7306

Break

33

COMP2303
COMP7306

Useful UNIX

34

COMP2303
COMP7306

Client-Server Model

- Most network applications are based on the **client-server model**:
 - A *server* process and one or more *client* processes
 - Server manages some *resource*
 - Server provides *service* by manipulating resource for clients

```

    graph LR
      Client((Client process)) -- "1. Client sends request" --> Server((Server process))
      Server -- "2. Server handles request" --> Resource[(Resource)]
      Server -- "3. Server sends response" --> Client
      Client -- "4. Client handles response" --> Client
    
```

Note: clients and servers are processes running on hosts (can be the same or different hosts).

35

COMP2303
COMP7306

Hardware and Software Organisation of an Internet Application

```

    graph TD
      subgraph Client_Host [Internet client host]
        direction TB
        C[Client] --- TCP_IP[TCP/IP]
        TCP_IP --- NA[Network adapter]
      end
      subgraph Server_Host [Internet server host]
        direction TB
        S[Server] --- TCP_IP[TCP/IP]
        TCP_IP --- NA[Network adapter]
      end
      NA --- GI[Global IP Internet]
      NA --- GI
      C --- SI[Sockets interface system calls]
      SI --- TCP_IP
      TCP_IP --- HI[Hardware interface interrupts]
      HI --- NA
    
```

36

COMP2303
COMP7306

Network Programming APIs

- **API** = Application Programming Interface
 - Set of procedures/functions/methods which provide access to some service
- Service of interest to us: transport
- Many network transport APIs exist
 - NetBIOS
 - TLI
 - XTI
 - Sockets
 - Winsock
 - ...

37

COMP2303
COMP7306

Role of Network Transport APIs

- Hide details of network layer (routing etc.)
- Provide simple, clean interface for application developers
- Ideally, hide protocol details
 - Remember the distinction between services and protocols

38

COMP2303
COMP7306

Sockets

- Introduced in Berkeley UNIX
- Sometimes called UNIX sockets
- Originally C based
 - Many other languages now
- A socket is a **communication endpoint**
 - Associated with a file descriptor in UNIX – can do file I/O on socket
 - Main distinction between regular file I/O and socket I/O is how the application "opens" the socket descriptors
- Many types of sockets
 - We're interested only in Internet sockets

39

COMP2303
COMP7306

Internet Sockets

Two types

- **Stream Sockets** (TCP)
 - Full-duplex byte streams
 - Reliable, connected
- **Datagram Sockets** (UDP)
 - Unreliable, connectionless
 - Limited-size messages
- Third type – Raw sockets
 - Allows direct access to network layer
 - We'll concentrate on the two above

40

COMP2303
COMP7306

Socket Primitives

- **socket(...)**
 - Create new communication end-point
- **bind(...)**
 - Attach a local address to a socket
- **listen(...)**
 - Willing to accept connections, give queue size
- **accept(...)**
 - Wait for a connection attempt to arrive
- **connect(...)**
 - Attempt to establish a connection

41

COMP2303
COMP7306

Socket Primitives (cont.)

- **send(...)** or **write(...)**
 - Send data over the connection
- **recv(...)** or **read(...)**
 - Receive data over the connection
- **sendto(...)**
 - Send datagram
- **recvfrom(...)**
 - Receive datagram
- **close(...)**
 - Release the connection
- **shutdown(...)**
 - Close down one side of connection (or both sides)
- Not all are applicable in all circumstances!

42

COMP2303
COMP7306

Typical Server (Stream based)

- Create socket `socket(...)`
- Bind to address/port `bind(...)`
- Specify willingness to accept connections `listen(...)`
- Block waiting for connection `accept(...)`
 - `accept(...)` returns a new socket
 - Original socket continues to listen
- Deal with request
 - e.g. spawn process or thread `send(...)` or `write(...)`
`recv(...)` or `read(...)`
- Continue

43

COMP2303
COMP7306

Typical Client (Stream based)

- Create socket `socket(...)`
- Connect to server (at a particular address) `connect(...)`
- Send/receive data as necessary `send(...)` or `write(...)`
`recv(...)` or `read(...)`
- Close connection `close(...)`
- Clients don't normally use `bind(...)`
 - don't care what the outgoing port is

44

COMP2303
COMP7306

Typical Datagram Applications

Typical Datagram Receiver

- Create datagram socket `socket(...)`
- Bind to address/port `bind(...)`
- Receive messages `recvfrom(...)`
- Close socket `close(...)`

Typical Datagram Sender

- Create datagram socket `socket(...)`
- Send messages `sendto(...)`
- Close socket `close(...)`

45

COMP2303
COMP7306

Other Programming Languages

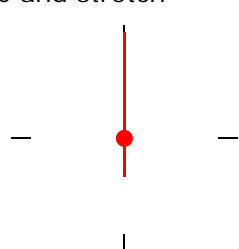
- Sockets interface is available in many programming languages
 - Interface is similar (but not identical) across all of them
- We'll be concentrating on the C sockets interface

46

COMP2303
COMP7306

Short Break

- Stand up and stretch



47

COMP2303
COMP7306

IP Addresses in C

- 32-bit IP addresses are stored in an *IP address struct*
 - IP addresses are always stored in memory in network byte order (big-endian byte order)
 - True in general for any integer transferred in a packet header from one machine to another
 - E.g., the port number used to identify an Internet connection

```

/* Internet address structure */
struct in_addr {
    unsigned int s_addr; /* network byte order (big-endian) */
};
    
```

- Handy network byte-order conversion functions:
 - `htonl()`: convert long int from host to network byte order.
 - `htons()`: convert short int from host to network byte order.
 - `ntohl()`: convert long int from network to host byte order.
 - `ntohs()`: convert short int from network to host byte order.

48

COMP2303
COMP7306

IP Addresses in C (cont.)

- Users (applications) often write IP addresses using dotted decimal notation
 - e.g. IP address 0x8002C2F2 = 128.2.194.242
- Functions for converting between binary IP addresses and dotted decimal strings:
 - `inet_addr(...)`: converts a dotted decimal string to an IP address in network byte order
 - `inet_ntoa(...)`: converts an IP address in network byte order to its corresponding dotted decimal string
 - "n" denotes network representation. "a" denotes application representation

49

COMP2303
COMP7306

Address Handling Code Example

- To be discussed in class*

50

COMP2303
COMP7306

Creating a Socket

- `socket(...)` creates a socket descriptor
 - `AF_INET`: indicates that the socket is associated with Internet protocols
 - `SOCK_STREAM`: selects a reliable byte stream connection (TCP)

```
int fd; /* socket descriptor */
/* Create a TCP socket descriptor */
if ((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
    perror("Socket creation failed");
    exit(1);
}
```

51

COMP2303
COMP7306

Socket Address Structures

- Generic socket address:
 - For address arguments to `connect`, `bind`, and `accept`.
 - Necessary only because C did not have generic (`void *`) pointers when the sockets interface was designed

```
struct sockaddr {
    unsigned short sa_family; /* protocol family */
    char sa_data[14]; /* address data. */
};
```
- Internet-specific socket address:
 - Must cast (`struct sockaddr_in *`) to (`struct sockaddr *`) for `connect`, `bind`, and `accept`

```
struct sockaddr_in {
    unsigned short sin_family; /* address family (always AF_INET) */
    unsigned short sin_port; /* port num in network byte order */
    struct in_addr sin_addr; /* IP addr in network byte order */
    unsigned char sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};
```

COMP2303
COMP7306

Socket Address Structures

```
struct sockaddr {
    unsigned short sa_family; /* protocol family */
    char sa_data[14]; /* address data. */
};

struct sockaddr_in {
    unsigned short sin_family; /* address family (always AF_INET) */
    unsigned short sin_port; /* port num in network byte order */
    struct in_addr sin_addr; /* IP addr in network byte order */
    unsigned char sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};
```

COMP2303
COMP7306

Resources

- Beej's Guide to Network Programming
 - <http://beej.us/guide/bgnet/>
- Glass & Ables, "UNIX for Programmers and Users"
- Rochkind, "Advanced UNIX Programming"
- Bryant and O'Halloran, "Computer Systems: A Programmer's Perspective"
- Other UNIX Programming books...
 - See Reference text list in course profile

54

COMP2303
COMP7306

Coming Up

- Week 10
 - Friday - Network Programming examples
 - Sample client and server
- Week 11
 - Tuesday – Network Applications
 - Friday – Network Programming (cont.)
- Week 12
 - Tuesday – Network Programming (cont.)
 - Ethernet
- Week 13
 - Internet Layer
 - Review

55

56

57

58

59

60