Operating Systems Design
15. Client-Server Networking

Paul Krzyzanowski
pxk@cs.rutgers.edu
Some networking terminology
Local Area Network (LAN)

Communications network
- small area (building, set of buildings)
- same, sometimes shared, transmission medium
- high data rate (often): 1 Mbps – 1 Gbps
- Low latency
- devices are peers
  - any device can initiate a data transfer with any other device

Most elements on a LAN are workstations
- endpoints on a LAN are called nodes
Connecting nodes to LANs
Connecting nodes to LANs

Adapter

- expansion slot (PCI, PC Card, USB dongle)
- usually integrated onto main board

Network adapters are referred to as
Network Interface Cards (NICs) or adapters
or Network Interface Component
(since they’re often not cards anymore)
Hubs, routers, bridges

**Hub**
- Device that acts as a central point for LAN cables
- Take incoming data from one port & send to all other ports

**Switch**
- Moves data from input to output port.
- Analyzes packet to determine destination port and makes a virtual connection between the ports.

**Concentrator** or **repeater**
- Regenerates data passing through it

**Bridge**
- Connects two LANs or two segments of a LAN: extends a LAN
- Connection at *data link layer* (layer 2)

**Router**
- Determines the next network point to which a packet should be forwarded
- Connects different types of local and wide area networks at *network layer* (layer 3)
How do nodes share a network?

- Dedicated connection – no sharing: **physical circuit**
- Talk on different frequencies: **broadband**
  - Range of frequencies: **FDM** (Frequency Division Multiplexing)
- Take turns: **baseband**
  - Short fixed time slots: **circuit switching**
    - **TDM** (Time Division Multiplexing)
    - **Circuit switching**: performance equivalent to an isolated connection
  - Variable size time slots: **packet switching**
    - *Statistical multiplexing* for network access
    - Easily support many-to-many communication

- *Packet switching is the dominant means of data communication*
Modes of connection

**Circuit-switching (virtual circuit network)**
- Dedicated path (route) – established at setup
- Guaranteed (fixed) bandwidth – routers commit to resources
- Typically fixed-length packets (cells) – each cell only needs a virtual circuit ID
- Constant latency

**Packet-switching (datagram network)**
- Shared connection; competition for use with others
- Data is broken into chunks called packets
- Each packet contains a destination address
- available bandwidth ≤ channel capacity
- variable latency
Client-Server Networking
What’s in the data?

For effective communication
  – same language, same conventions

For computers:
  – electrical encoding of data
  – where is the start of the packet?
  – which bits contain the length?
  – is there a checksum? where is it? how is it computed?
  – what is the format of an address?
  – byte ordering
Protocols

These instructions and conventions are known as protocols
Protocols

Exist at different levels

understand format of address and how to compute a checksum

humans vs. whales different wavelengths

versus

request web page

French vs. Hungarian
Layering

To ease software development and maximize flexibility:

– Network protocols are generally organized in **layers**
– Replace one layer without replacing surrounding layers
– Higher-level software does not have to know how to format an Ethernet packet

… or even know that Ethernet is being used
Layering

Most popular model of guiding (not specifying) protocol layers is

**OSI reference model**

Adopted and created by ISO

7 layers of protocols
OSI Reference Model: Layer 1

Transmits and receives raw data to communication medium

Does not care about contents

Media, voltage levels, speed, connectors

Deals with representing bits

Examples: USB, Bluetooth, 1000BaseT, Wi-Fi
OSI Reference Model: Layer 2

- Detects and corrects errors
- Organizes data into frames before passing it down. Sequences packets (if necessary)
- Accepts acknowledgements from receiver

Examples: Ethernet MAC, PPP

Data Link

Physical

Deals with frames

© 2014 Paul Krzyzanowski
OSI Reference Model: Layer 2

An **ethernet switch** is an example of a device that works on layer 2. It forwards **ethernet frames** from one host to another as long as the hosts are connected to the switch (switches may be cascaded).

This set of hosts and switches defines the **local area network (LAN)**.
OSI Reference Model: Layer 3

- Relay and route information to destination
- Manage journey of datagrams and figure out intermediate hops (if needed)

Examples: IP, X.25
OSI Reference Model: Layer 4

Provides an interface for end-to-end (application-to-application) communication: sends & receives segments of data. Manages flow control. May include end-to-end reliability.

Network interface is similar to a mailbox.

Examples: TCP, UDP
OSI Reference Model: Layer 5

Services to coordinate dialogue and manage data exchange

Software implemented switch

Manage multiple logical connections

Keep track of who is talking: establish & end communications

Deals with data streams

Examples: HTTP 1.1, SSL
OSI Reference Model: Layer 6

Presentation
Session
Transport
Network
Data Link
Physical

Data representation
Concerned with the meaning of data bits
Convert between machine representations

Deals with objects

Examples: XDR, ASN.1, MIME, JSON, XML

© 2014 Paul Krzyzanowski
OSI Reference Model: Layer 7

Collection of application-specific protocols

- Examples:
  - Email (SMTP, POP, IMAP)
  - File transfer (FTP)
  - Directory services (LDAP)

Deals with services: app-specific protocols

© 2014 Paul Krzyzanowski
IP – Internet Protocol

Born in 1969 as a research network of 4 machines
Funded by DoD’s ARPA

Goal:

*Build an efficient fault-tolerant network that could connect heterogeneous machines and link separately connected networks.*
Internet Protocol

- A set of protocols designed to handle the interconnection of a large number of local and wide-area networks that comprise the Internet
  - IP: network layer – other protocols include TCP, UDP, ICMP, etc.

- The IP layer relies on **routing** from one physical network to another

- At the network layer, IP is connectionless
  - no state needs to be saved at each router

- Survivable design: support multiple paths for data
  - … but packet delivery is not guaranteed!
IP vs. OSI stack

<table>
<thead>
<tr>
<th>OSI layer</th>
<th>IP layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td>Data Link</td>
<td>Network</td>
</tr>
<tr>
<td>Transport</td>
<td>Transport</td>
</tr>
<tr>
<td>Session</td>
<td>Presentation</td>
</tr>
<tr>
<td>Presentation</td>
<td>Application</td>
</tr>
</tbody>
</table>

Middleware is represented in the middle of the OSI stack.
Protocol Encapsulation

At any layer

- The higher level protocol headers are just treated like data
- Lower level protocol headers can be ignored

An ethernet switch or ethernet driver sees this:

```
Ethernet header Ethernet payload CRC
```

A router or IP driver sees this:

```
Ethernet header IP header IP payload CRC
```

A TCP driver sees this:

```
Ethernet header IP header TCP header TCP payload CRC
```

An application sees this:

```
Ethernet header IP header TCP header TCP payload CRC
```
Client – Server Communication
Addressing machines (data link layer)

Each interface on a host has a unique MAC address
- E.g., aramis.rutgers.edu: 48-bit ethernet address = 00:03:ba:09:1b:b0

• This isn’t too interesting to us as programmers
  - We can send ethernet frames to machines on the same LAN
Addressing machines (network layer)

Each interface on a host is given a unique IP address

- The IP address is *not* the network hardware address
  - IP is a *logical network* that overlays & connects physical networks
- IPv4 (still the most common in the U.S.): 32-bit number
  - Example, `cs.rutgers.edu = 128.6.4.2 = 0x80060402`
- IPv6: 128-bit number
  - Example, `cs.rutgers.edu = 0:0:0:0:0:FFFF:128.6.4.2 = ::FFFF:8006:0402`

- Routable across networks
  - We can send IP packets to machines on the Internet
  - **BUT** … we want to talk to applications
Ethernet & IP Reliability

- **Ethernet**
  - LAN connectivity
  - Higher-level protocols (IP) encapsulated inside
  - Unreliable delivery
    - Frames may be lost to congestion, errors, or collision

- **IP**
  - Datagram delivery is also unreliable
  - Frames may be lost due to dropped ethernet frames, errors, congestion, or time-to-live expiration
Address translation

• **Domain name → IP address translation**
  - Domain Name System (DNS)
    • Hierarchical human-friendly names (e.g., www.cs.rutgers.edu)
  - User-level network service to look up IP domain names
  - Cache results to avoid future look-ups
  - The kernel’s network drivers do not handle domain names

• **IP → Ethernet address translation**
  - Address Resolution Protocol (ARP)
  - How does the OS know which ethernet address to use?
  - Broadcast an ARP query and wait for a response
    “Who has 128.6.4.2?”
  - Cache results to avoid future look-ups
Network API

• App developers need access to the network
• A *Network Application Programming Interface (API)* provides this
• Core services provided by the operating system
  – Operating System controls access to resources
• Libraries may handle the rest

• We will only look at IP-based communication
# Programming: connection-oriented protocols

**virtual circuit service (example: TCP)**

- provides illusion of having a dedicated circuit
- messages guaranteed to arrive in-order
- application does not have to address each message

*Not to be confused with virtual circuit networks*

- Which provide constant latency & guaranteed bandwidth
- TCP simulates a virtual circuit network … sort of (except for bandwidth and latency guarantees)

### 1. establish connection
- *dial phone number*

### 2. [negotiate protocol]
- *[decide on a language]*

### 3. exchange data
- *speak*

### 4. terminate connection
- *hang up*
Programming: connectionless protocols

- no call setup
- send/receive data
  (each packet addressed)
- no termination

analogous to mailbox

drop letter in mailbox
  (each letter addressed)

datagram service (example: UDP)
  – client is not positive whether message arrived at destination
  – no state has to be maintained at client or server
  – cheaper but less reliable than virtual circuit service
IP transport layer

IP give us two transport-layer protocols

- **TCP: Transmission Control Protocol**
  - Connection-oriented service
    - Operating system keeps state: simulates a virtual circuit over a datagram network
  - Full-duplex connection: both sides can send messages over the same link
  - Reliable data transfer: the protocol handles retransmission
  - In-order data transfer: the protocol keeps track of sequence numbers

- **UDP: User Datagram Protocol**
  - Connectionless service: lightweight transport layer over IP
  - Data may be lost
  - Data may arrive out of sequence
  - Checksum for corrupt data: operating system drops bad packets
Addressing applications (transport layer)

Communication endpoint at the machine

- **Port number**: 16-bit value

- Port number = transport endpoint
  - Allows application-application communication
  - Identifies a specific data stream

- Some services use well-known port numbers (0 – 1023)
  - IANA: Internet Assigned Numbers Authority (www.iana.org)
  - Also see the file /etc/services

- Ports for proprietary apps: 1024 – 49151

- Dynamic/private ports: 49152 – 65535

- **To communicate with applications, we use a transport layer protocol and an IP address and port number**
The End