Outline

- Admin. and recap
- A taxonomy of communication networks
- Layered network architecture

Admin.

- Readings from the textbook and additional suggested readings
  - All are highly recommended
  - Some are marked as required

- Assignment one is linked on the schedule page
  - Due Sept. 12, 2013, in class
  - If you type it in, you can upload to classes V2

Recap: A Taxonomy of Comm. Networks

- Circuit-switched networks (e.g. telephone, GSM)
- Packet-switched networks (e.g. Internet)

Recap: Circuit Switching vs. Packet Switching

<table>
<thead>
<tr>
<th></th>
<th>Circuit Switching</th>
<th>Packet Switching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource usage</td>
<td>Use a single partition bandwidth</td>
<td>Use whole link bandwidth</td>
</tr>
<tr>
<td>Reservation/Setup</td>
<td>Need reservation (setup delay)</td>
<td>No reservation</td>
</tr>
<tr>
<td>Resource Contention</td>
<td>Busy signal (session loss)</td>
<td>Congestion (long delay and packet losses)</td>
</tr>
<tr>
<td>Charging</td>
<td>Time packet</td>
<td></td>
</tr>
<tr>
<td>Header</td>
<td>No header</td>
<td>Per packet header</td>
</tr>
<tr>
<td>Fast Path Processing</td>
<td>Fast</td>
<td>Per packet processing</td>
</tr>
</tbody>
</table>

Recap: Queueing Theory

- We are not interested in extremely precise modeling, but want quantitative intuition

- Strategy:
  - Model system state
  - If we know the fraction of time the system spends at each state, we can get answers to some basic questions: how long does a new request need to wait before being served?

- System state changes upon events:
  - Introduce state transition diagram
  - Focus on equilibrium: state trend neither growing nor shrinking
    - One intuitive way to define equilibrium is time reversibility
Equilibrium = Time Reversibility

Example: Circuit-Switching Blocking (Busy) Time: Sketch

Example: Packet Switching Delay

Packet Switching Delay

Example

Analysis of Delay (cont)
Delay

\[ delay = \frac{1}{\mu} + 1 - \frac{1}{\mu} \]

Queueing Delay as a Function of Utilization

\[ \rho \rightarrow 1: \text{delay becomes large} \]
\[ \rho > 1: \text{more "work" arriving than can be serviced, average delay infinite} \]

To Partition, or not to Partition: This is the Question

Case 1 (not reserve): all arrivals into a single queue serving with rate \( R \)

Case 2 (reserve): the arrivals are divided into \( n \) links with rate \( \frac{R}{n} \) each

Statistical Multiplexing

- reservations/dedication (aka circuit-switching) and no reservation (aka packet switching)

- a single bottleneck link

- no reservation: all arrivals into the single link, the queueing delay + transmission delay:

\[ \frac{L}{R} \frac{1}{1 - \rho} \]

- reservation: each flow uses its own reserved (sub)link with rate \( \frac{R}{n} \), the queueing delay + transmission delay:

\[ \frac{L}{R} \frac{1}{1 - \rho} \]

Summary: Packet Switching vs. Circuit Switching

- Advantages of packet switching over circuit switching:
  - most important advantage of packet-switching over circuit switching is statistical multiplexing, and therefore more efficient bandwidth usage.

- Disadvantages of packet switching:
  - potential congestion: packet delay and high loss
  - protocols needed for reliable data transfer, congestion control
  - it is possible to guarantee quality of service (QoS) in packet-switched networks and still gain statistical multiplexing, but it adds much complexity

For a demo of M/M/1, see:
http://www.dcs.ed.ac.uk/home/jeh/Simjava/queueing/mmm1_q/mm1_q.html

For a demo of M/M/1, see:
http://www.dcs.ed.ac.uk/home/jeh/Simjava/queueing/mmm1_q/mm1_q.html
Outline

- Admin. and recap
- A taxonomy of communication networks
  - circuit switched networks
  - packet switched networks
  - circuit switching vs. packet switching
  - datagram and virtual circuit packet switched networks

A Taxonomy of Packet-Switched Networks According to Routing

- Goal: move packets among routers from source to destination
  - we’ll study routing algorithms later in the course

- Two types of packet switching
  - datagram network
    - each packet of a flow is switched independently
  - virtual circuit network
    - all packets from one flow are sent along a pre-established path (= virtual circuit)

Datagram Packet Switching

- Commonly when we say packet switching we mean datagram switching
- Example: IP networks
- Each packet is independently switched
  - each packet header contains complete destination address
  - receiving a packet, a router looks at the packet’s destination address and searches its current routing table to determine the possible next hops, and pick one
- Analogy: postal mail system

Virtual-Circuit Packet Switching

- Example: Multiple Label Packet Switching (MPLS) in IP networks, e.g.,

- Hybrid of circuit switching and datagram switching
  - fixed path determined at virtual circuit setup time, remains fixed thru flow
  - Two implementations:
    1. each packet carries a short tag (virtual-circuit (VC) #): tag determined next hop
    2. Match on packet attributes to find entries

Timing Diagram of Datagram Switching
Virtual-Circuit Switching

Three phases
1. VC establishment
2. Data transfer
3. VC disconnect

Timing Diagram of Virtual-Circuit Switching

Discussion: Datagram Switching vs. Virtual Circuit Switching

What are the benefits of datagram switching over virtual circuit switching?

What are the benefits of virtual circuit switching over datagram switching?

Summary of the Taxonomy of Communication Networks

Summary of Progress

We have seen the hardware infrastructure, the basic communication scheme, a next key question is how to develop the software system.
Outline

- Admin. and recap
- A taxonomy of communication networks
- Layered network architecture
  - what is layering?
  - why layering?
  - how to determine the layers?
- ISO/OSI layering and Internet layering

What is Layering?

- A technique to organize a networked system into a succession of logically distinct entities, such that the service provided by one entity is solely based on the service provided by the previous (lower level) entity.

Why Layering?

- Networks are complex:
  - many “pieces”:
    - hardware
      - hosts
      - routers
      - links of various media
    - software
      - applications
      - protocols

An Example: No Layering

- No layering: each new application has to be re-implemented for every network technology!

An Example: Benefit of Layering

- Introducing an intermediate layer provides a common abstraction for various network technologies

Dealing with complex systems: explicit structure allows identification of the relationship among a complex system’s pieces

Modularization eases maintenance, updating of system:

- change of implementation of a layer’s service transparent to rest of system, e.g., changes in routing protocol doesn’t affect rest of system
**ISO/OSI Concepts**

- ISO - International Standard Organization
- OSI - Open System Interconnection
- Service - says what a layer does
- Interface - says how to access the service
- Protocol - specifies how the service is implemented
  - a set of rules and formats that govern the communications between two or more peers

**An Example of Layering**

![Layering Diagram]

**Protocol Layering and Meta Data**

Each layer takes data from above
- adds header (meta) information to create new data unit
- passes new data unit to layer below

**Layering: Logical Communication**

**E.g.: application**
- provide services to users
- application protocol:
  - send messages to peer
  - for example, HELO, MAIL, FROM, RCPT TO are messages between two SMTP peers

**E.g.: transport**
- take msg from app
- Transport protocol
  - add control info to form "datagram"
  - send datagram to peer
  - wait for peer to ack receipt; if no ack, retransmit
Layering: Physical Communication

Key design issue:
How do you divide functionalities among the layers?

Outline
- Review
- A taxonomy of communication networks
- Layered network architecture
  - what is layering?
  - why layering?
  - how to determine the layers?

The End-to-End Arguments
The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication systems. Therefore, providing that questioned function as a feature of the communications systems itself is not possible.

J. Saltzer, D. Reed, and D. Clark, 1984

What does the End-to-End Arguments Mean?
- The application knows the requirements best, place functionalities as high in the layer as possible
- Think twice before implementing a functionality at a lower layer, even when you believe it will be useful to an application

Example: Where to Provide Reliability?
- Solution 1: the network (lower layer L1) provides reliability, i.e., each hop provides reliability
- Solution 2: the end host (higher layer L2) provides reliability, i.e., end-to-end check and retry
**What are Reasons for Implementing Reliability at Higher Layer?**

- The lower layer cannot completely provide the functionality the receiver has to do the check anyway!
- Implementing it at lower layer increases complexity, cost and overhead at lower layer. The upper layer knows the requirements better and thus may choose a better approach to implement it.

**Are There Reasons Implementing Reliability at Lower Layer?**

- Improve performance, e.g., if high cost/delay... on a local link.
  - improves efficiency
  - reduces delay
- Share common code, e.g., reliability is required by multiple applications.

**Summary: End-to-End Arguments**

- If a higher layer can do it, don’t do it at a lower layer -- the higher the layer, the more it knows about the best what it needs.
- Add functionality in lower layers iff it is used by and improves performance of a large number of (current and potential future) applications, does not hurt (too much) other applications, and does not increase (too much) complexity/overhead.
- Practical tradeoff, e.g., allow multiple interfaces at a lower layer (one provides the function; one does not).

**Examples**

- We used reliability as an example.
  - Assume two layers (L1: network; L2: end-to-end). Where may you implement the following functions?
    - security (privacy of traffic)
    - quality of service (e.g., delay/bandwidth guarantee)
    - congestion control (e.g., not to overwhelm network links or receiver)

**Challenges**

- Challenges to build a good (networking) system: find the right balance between:
  - end-to-end arguments
  - reuse, interoperability, implementation effort (apply layering concepts)

No universal answer: the answer depends on the goals and assumptions!

**Example**

- Consider the presence service in a social networking system: shows which contacts are online (e.g., skype, MSN)
  - implementing by end user’s host app or through a third party service?
Goals

1. Survivability in the face of failure
2. Support multiple types of services
3. Accommodate a variety of networks
4. Permit distributed management of resources
5. Be cost effective
6. Permit host attachment with a low level of effort
7. Be accountable

Survivability in the Face of Failure

- Continue to operate even in the presence of network failures (e.g., link and router failures)
  - as long as the network is not partitioned, two endpoints should be able to communicate, moreover, any other failure (excepting network partition) should be transparent to endpoints
- Decision: maintain state only at end-points (fate-sharing)
  - eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: stateless network architecture

Survivability in the Face of Failure: Questions

- What does the goal mean?
- Why is the goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?

Support Multiple Types of Service

- What does this goal mean?
- Why is the goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?
**Support Multiple Types of Service**
- Add UDP to TCP to better support other types of applications
  - e.g., "real-time" applications
- This was arguably the main reason for separating TCP and IP
- Provide datagram abstraction: lower common denominator on which other services can be built: everything over IP
  - service differentiation was considered (remember ToS?), but this has never happened on the large scale (Why?)

**Support a Variety of Networks: Questions**
- What does the goal mean?
- Why is this goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?

**Support a Variety of Networks**
- Very successful
  - because the minimalist service; it requires from underlying network only to deliver a packet with a "reasonable" probability of success
- ...does not require:
  - reliability
  - in-order delivery
- The mantra: IP over everything
  - Then: ARPANET, X.25, DARPA satellite network...
  - Now: ATM, SONET, WDM...

**Other Goals**
- Permit distributed management of resources
- Be cost effective
- Permit host attachment with a low level of effort
- Be accountable