Statistical Multiplexing; Layered Network Architecture

Y. Richard Yang

http://zoo.cs.yale.edu/classes/cs433/

9/5/2013
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, General Packet Radio Service (GPRS)

- **broadcast networks**

- **switched networks**

- **communication networks**

- **circuit switching:** dedicated circuit per call/session:
  - e.g., telephone, GSM High-Speed Circuit-Switched Data (HSCSD)

- **packet switching:** data sent thru network in discrete “chunks”
  - e.g., Internet, General Packet Radio Service (GPRS)
### 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</td>
<td>packet</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

---

- Cannot distinguish

\[ \# f_{k \rightarrow k+1}, \; \# f_{k+1 \rightarrow k} \]

\[ \# b_{k \rightarrow k+1}, \; \# b_{k+1 \rightarrow k} \]
Example: Circuit-Switching Blocking (Busy) Time: Sketch

system state: # of busy lines

\[
p_0 \quad p_1 \quad \ldots \quad p_k \quad p_{k+1} \quad \ldots \quad p_N
\]

at equilibrium (time reversibility) in one unit time:

\[\#(\text{transitions } k \rightarrow k+1) = \#(\text{transitions } k+1 \rightarrow k)\]

\[
p_k \lambda = p_{k+1} (k+1) \mu
\]

\[
p_{k+1} = \frac{1}{k+1} \frac{\lambda}{\mu} p_k = \frac{1}{(k+1)!} \left( \frac{\lambda}{\mu} \right)^{k+1} p_0
\]

\[
p_0 = \frac{1}{1 + \frac{1}{1!} \frac{\lambda}{\mu} + \frac{1}{2!} \left( \frac{\lambda}{\mu} \right)^2 + \ldots + \frac{1}{N!} \left( \frac{\lambda}{\mu} \right)^N}
\]
Example: Packet Switching Delay

- **Four** types of delay at each hop
  - nodal processing delay: check errors & routing
  - queueing: time waiting for its turn at output link
  - transmission delay: time to pump packet onto a link at link speed
  - propagation delay: router to router propagation
- The focus is on **queueing and transmission delay**
Packet Switching Delay

system state: #packets in queue

at equilibrium (time reversibility) in one unit time:

\[ p_k \lambda = p_{k+1} \mu \]

\[ p_{k+1} = \frac{\lambda}{\mu} p_k = \left(\frac{\lambda}{\mu}\right)^{k+1} p_0 = \rho^{k+1} p_0 \]

\[ p_0 = 1 - \rho \]
Example

- Assume requests (packets) come in at a rate of one request per 30 seconds
- Each request (packet) takes on average 20 seconds

- What is the fraction of time that the system is empty?
- What is the chance that a packet newly arrived needs to wait for 3 early packets?
Analysis of Delay (cont’)

- Average queueing delay:
- Transmission delay:
- Queueing + transmission:
Delay

Assume:
- \( R \) = link bandwidth (bps)
- \( L \) = packet length (bits)
- \( a \) = average packet arrival rate (pkt/sec)

queuesing + trans = \( \frac{1}{\mu} \frac{\rho}{1-\rho} + \frac{1}{\mu} \)

Link utilization
(also called traffic intensity)

\[ \text{utilization} : \rho = \frac{a}{R/L} = \frac{La}{R} \]

average queueing delay: \( w = \frac{L}{R} \frac{\rho}{1-\rho} \)

For a demo of M/M/1, see:
http://www.dcs.ed.ac.uk/home/jeh/Simjava/queueing/mm1_q/mm1_q.html
Queueing Delay as a Function of Utilization

Assume:
R = link bandwidth (bps)
L = packet length (bits)
a = average packet arrival rate (pkt/sec)

\[
\text{utilization: } \rho = \frac{a}{R/L} = \frac{La}{R}
\]

- \( \rho \sim 0 \): average queueing delay small
- \( \rho \rightarrow 1 \): delay becomes large
- \( \rho > 1 \): more “work” arriving than can be serviced, average delay infinite!
To Partition, or not to Partition: This is the Question.

Assume:

\[ R = \text{link bandwidth (bps)} \]
\[ L = \text{packet length (bits)} \]
\[ a = \text{average packet arrival rate (pkt/sec)} \]

Setup: \( n \) streams; each stream has an arrival rate of \( a/n \)

Comparison: each stream reserves \( 1/n \) bandwidth or not

- **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 \( R/n \) each
Partition or Not

4pkt/sec 10M

2pk/sec 5M

2pk/sec 5M
**Statistical Multiplexing**

A simple model to compare bandwidth efficiency of
- reservation/dedication (aka circuit-switching) and
- no reservation (aka packet switching)
setup
- a single bottleneck link
- n flows; each flow has an arrival rate of \( \frac{a}{n} \)

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

Assume:
- \( R \) = link bandwidth (bps)
- \( L \) = packet length (bits)
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.
  - Packet header overhead.
  - Per packet processing overhead.
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
Datagram Packet Switching
**Timing Diagram of Datagram Switching**

Transmission time of Packet 1 at Host A

Propagation delay from Host A to Node 1

processing and queueing delay of Packet 1 at Node 2
Virtual-Circuit Packet Switching

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

- Hybrid of circuit switching and datagram switching
  - Hybrid of circuit switching and datagram switching
  - Two implementations:
    1. each packet carries a short tag (virtual-circuit (VC) #); tag determines next hop
    2. Match on packet attributes to find entries

<table>
<thead>
<tr>
<th>Incoming VC#</th>
<th>Outgoing Interface</th>
<th>QoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Virtual-Circuit Switching
Virtual-Circuit Packet Switching

- Three phases
  1. VC establishment
  2. Data transfer
  3. VC disconnect
Timing Diagram of Virtual-Circuit Switching

Host 1

Node 1

Node 2

Host 2

VC establishment

Packet 1

Packet 2

Packet 3

Packet 1

Packet 2

Packet 3

packet delay between Host 1 and Node 1

data transfer

VC termination
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

communication network

- switched network
  - circuit-switched network
  - datagram network
- packet-switched network
- broadcast communication
  - virtual circuit network
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.

```
<table>
<thead>
<tr>
<th>Physical</th>
<th>Datalink</th>
<th>Network</th>
<th>Transport</th>
<th>Application</th>
</tr>
</thead>
</table>

Physical medium
```
Outline

- Admin. and recap
- A taxonomy of communication networks
- Layered network architecture
  - what is layering?
    - why layering?
Why Layering?

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

Dealing with complex systems: explicit structure allows identification of the relationship among a complex system’s pieces
- layered reference model for discussion

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
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.

Diagram:
- Application layer: Telnet, FTP, HTTP
- Transport & Network layer: Ethernet, Fiber optic, Wireless
- Transmission Media layer: Ethernet, Fiber optic, Wireless
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
An Example of Layering

Layer 1
Layer 2
Layer 3
Layer 4
Layer 5

Layer 1/2 interface
Layer 2/3 interface
Layer 3/4 interface
Layer 4/5 interface
Layer 5 protocol
Layer 4 protocol
Layer 3 protocol
Layer 2 protocol
Layer 1 protocol

Physical medium

Host 1
Host 2
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 $\rightarrow$ *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
Layering: *Logical Communication*

**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
  - shared by all upper layer applications → everyone pays for it, even if you do not need it
- 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
  (1) is used by and improves performance of a large number of (current and potential future) applications,
  (2) does not hurt (too much) other applications, and
  (3) 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)
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?
Challenges

- Challenges to build a good (networking) system: find the right balance between:

  - end-to-end arguments
  - performance
  - reuse, interoperability, implementation effort (apply layering concepts)

No universal answer: the answer depends on the goals and assumptions!
Backup Slides
The Design Philosophy of the DARPA Internet
Goals

0. Connect different networks

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: 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)?
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
Support Multiple Types of Service: Questions

- 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