Computer Network Fundamentals
Spring 2007

Midterm Review
Andreas Terzis
Midterm

• 3/07 in class

• Format
  – Closed book
  – Bring a calculator

• Material
  – Reading assignments
  – Presentation slides
Internet protocol architecture

- **Applications**: client-server model, HTTP, FTP, SMTP, DNS
- **transport**: deliver data from process to process
  - issues: multiplexing and demultiplexing, reliable delivery and flow control (Seq#, ACK, rxt timer, stop-and-go, go-back-N), congestion control (TCP slow-start)
- **IP**: deliver packets from host to host
  - issues: IP address structure (class-based, CIDR, subnet), packet fragmentation, routing (link-state, distance-vector, multicast, mobile IP)
- **Subnet/link**: delivery data frames between physically connected nodes
  - issues: error checking (CRC), framing, media access (CSMA/CD), LAN interconnect (bridge), address translation
- **physical**: transmitting bit streams (not covered)
  - issues: bit encoding, signal power...
Layered protocol implementation

A protocol defines:
- the format of message exchanged between peer entities
- the actions taken on receipt of the message
Interactions between layers

• When going down layer stack: multiplexing
  – Data from all applications and all protocols eventually going out the same network interface
  – Applications provide the IP address to send to
    • Between subnet and IP, need ARP to find IP->subnet address mapping

• When going up: demultiplexing
  – Pass incoming data up to their corresponding application processes
  – Subnet knows which protocol to pass the data to
  – IP uses “protocol” field in IP header to identify the layer above
  – Transport protocol uses port# to identify appl.’s
Packet switching

- What is packet switching?
  - Data is cut into chunks, sent in a "store-and-forward" way
  - statistical multiplexing → queueing delay, potential losses

- What’s in a packet:
  - header: contains all the information needed for the intended data delivery function

- Why layered network protocol architecture?
  - Divide and conquer
BW-Delay Examples

- $P=1$ Kbyte
  $R = 1$ Gbps
  100 km fiber
  $T=500$ usec
  $P/R = 8$ usec

- $P=1$ Kbyte
  $R = 100$ Mbps
  1 Km fiber
  $T=5$ usec
  $P/R = 80$ usec
Queuing

- The queue has one packet when second packet arrives -> packet has to wait in queue for the first packet to be transmitted

Tx Time: P/R

Tx Time: Q/R

Time
Store and Forward

- Packet is enqueued before being sent
Bandwidth-Delay Product

- **Window based flow control:**
  - Send $W$ bits (window size)
  - Wait for Acks
  - Repeat

- **Throughput = $W/RTT$**
  - $W=64$ Kbytes
  - $RTT=200$ms
  - Throughput = $W/T = 2.6$Mbps
Reliable Transfer Protocols

• Two basic mechanisms
  – (N)ACKs
  – Times

• Performance
  – Stop and Wait
  – Go-Back-N
  – Selective Retransmit
Multiaccess protocols

- Single communication channel shared by multiple nodes
- Only one node can send successfully at a time

**Random access**
- ALOHA, slotted ALOHA
- CSMA/CD
- CSMA/CA

**Controlled access**
- Adaptive to demand
- Static allocation
- TDMA
- FDMA

*CS 344/Fall 07*
Ethernet

- CSMA/CD, collision resolution (exponential backoff)
- Q: How to connect multiple Ethernets
- Q: Compare hubs, bridges and routers
- Q: What’s the function of ARP?
The world according to IP

Various application protocols

transport (end-to-end)

TCP  UDP  (RDP  SCTP)

Various network technologies

IP  internet layer

Ethernet  wireless  FDDI  dialup  ATM
The world according to IP

- All hosts connected to physical networks (subnet)
- All subnets interconnected by IP routers
  - receive and forward packets between subnets
  - at subnet level a router sends/received data in exactly the same way as a host
- IP assigns globally unique addresses to all reachable interfaces (connecting to either hosts or routers)
- datagram delivery between these interfaces
  - routers run routing protocols to figure out the next hop along shortest path to forward each IP packet to its destination
Relation between IP and subnet layer

- IP performs host-to-host packet delivery, possibly through a chain of IP routers
- Subnets do the real work of getting packets from one IP node to the next
- Functions needed
  - Each IP router looks up forwarding table to determine next hop to forward the packet to
  - Address translation from IP address to subnet address
  - Packet encapsulation and decapsulation when crossing the subnet-IP layer boundary
Interconnection by encapsulation

- IP packets are wrapped in a network’s protocol to travel through that network
- A router un-wraps the packet to see its IP destination address
  - on the same network: send to destination directly
  - on a diff. network: send to next hop router
Following an IP packet from source to dest

Source host A first uses subnet mask M to figure out whether dest. host is on the same network

1. Dest. = host B: find B's MAC address, send data
2. dest. = host C, A sends packet to its default router
   - the router strips off Ethernet header, consult its IP forwarding table to find next hop
3. Dest. = D:
IP Fragmentation

- all fragments of the same packet carry the same identifier
- all fragments except the last one have the “MF” bit set
- fragment offset points to the first byte of the fragment

example:

1st fragment: identifier=8FB3; MF=1; offset=0
2nd fragment: identifier=8FB3; MF=1; offset=64
3rd fragment: identifier=8FB3; MF=0; offset=128

```
data=1300B                      IP header

data frag-3  IP Hd  data frag-2  IP Hd  data frag-1  IP Hd
(276B)             (512B)             (512B)
```
IP address structure

• 4 bytes
• Hierarchical! (i.e. not flat, as MAC addresses)
  - network ID
  - host ID
• Where is the boundary between these 2 parts:
  - Class-based address: classes A, B, C
  - Subnetting
  - Classless Inter-Domain Routing (CIDR)
Subnetting

- subnetting: Add another (hidden) level to address hierarchy
  - Subnet is known only at the local site
  - netID includes part of the original host ID portion
- **Subnet mask**: defines portion of the address considered as “network ID” *by the local site*

`\begin{center}
\begin{tikzpicture}
\begin{scope}[every node/.style={font=\small}]
\node (n1) at (0,0) {Network ID};
\node (n2) at (0,-2) {Network-ID host-ID};
\node (n3) at (0,-4) {1111111111111111111111110000000000};
\node (n4) at (0,-6) {Network ID};
\end{scope}
\end{tikzpicture}
\end{center}`
CIDR: Classless InterDomain Routing

- assign network addresses by *blocks of contiguous IP addresses*, in a form of
  \[<\text{IP address}, \text{mask}>\]
  - mask identifies block size, must be power of 2
  - example: *SmartDesign Inc.* got 4 x 2^8 address blocks 192.4.16.0—192.4.19.255,
    \[<192.4.16.0, 255.255.252.0>, \text{or} \ 192.4.16/22\]
Routing

- Intra-domain: use link state or distance vector protocols
- Inter-domain: use path vector protocol
Intra-Domain Routing

- **Link state (e.g., OSPF):**
  - Each router periodically floods neighborhood information to every other node in the network
  - Each router uses the received information to build the complete topology of the network and then compute shortest path using Dijkstra

- **Distance Vector (e.g., RIP)**
  - Each router periodically sends its reachability information (to other nodes in the network) to its neighbors
  - Upon receiving this information each router updates its routing table

- **Q:** What is the difference between the forwarding and the routing tables?
Example: Dijkstra’s Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>start S</th>
<th>( D(B), p(B) )</th>
<th>( D(A), p(A) )</th>
<th>( D(D), p(D) )</th>
<th>( D(E), p(E) )</th>
<th>( D(F), p(F) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>8, C</td>
<td>5, C</td>
<td>3, C</td>
<td>1, E</td>
<td>5, F</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. **Initialization:**
2. \( S = \{C\}; \)
3. for all nodes \( v \)
4. if \( v \) adjacent to \( C \)
5. then \( D(v) = c(C, v); \)
6. else \( D(v) = \infty; \)

\[ ... \]
## Example: Dijkstra’s Algorithm

### Table

<table>
<thead>
<tr>
<th>Step</th>
<th>start S</th>
<th>D(B),p(B)</th>
<th>D(A),p(A)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>8,C</td>
<td>5,C</td>
<td>3,C</td>
<td>1,C</td>
<td>5,C</td>
</tr>
<tr>
<td>1</td>
<td>CE</td>
<td>8,C</td>
<td>5,C</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Loop

8 **Loop**
9 find w not in S s.t. D(w) is a minimum;
10 add w to S;
11 update D(v) for all v adjacent to w and not in S:
12 \[ D(v) = \min(D(v), D(w) + c(w,v)) \];
13 until all nodes in S;
## Example: Dijkstra’s Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>Start S</th>
<th>(D(B),p(B))</th>
<th>(D(A),p(A))</th>
<th>(D(D),p(D))</th>
<th>(D(E),p(E))</th>
<th>(D(F),p(F))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>8,C</td>
<td>5,C</td>
<td>3,C</td>
<td>1,C</td>
<td>5,C</td>
</tr>
<tr>
<td>1</td>
<td>CE</td>
<td>8,C</td>
<td>5,C</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>2</td>
<td>CED</td>
<td>6,D</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Loop

9. find \(w\) not in \(S\) s.t. \(D(w)\) is a minimum;
10. add \(w\) to \(S\);
11. update \(D(v)\) for all \(v\) adjacent to \(w\) and not in \(S\):
    \[
    D(v) = \min( D(v), D(w) + c(w,v) )
    \]
13. until all nodes in \(S\);

---

CS 344/Fall 07 27
Example: Dijkstra’s Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>start S</th>
<th>D(B),p(B)</th>
<th>D(A),p(A)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>8,C</td>
<td>5,C</td>
<td>3,C</td>
<td>1,C</td>
<td>5,C</td>
</tr>
<tr>
<td>1</td>
<td>CE</td>
<td>8,C</td>
<td>5,C</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>2</td>
<td>CED</td>
<td>8,C</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>3</td>
<td>CEDF</td>
<td>6,D</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
</tbody>
</table>

**Loop**

9. find w not in S s.t. D(w) is a minimum;
10. add w to S;
11. update D(v) for all v adjacent to w and not in S:
   \[ D(v) = \min( D(v), D(w) + c(w,v) ) \]
12. until all nodes in S;
### Example: Dijkstra’s Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>start S</th>
<th>D(B),p(B)</th>
<th>D(A),p(A)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>8,C</td>
<td>5,C</td>
<td>3,C</td>
<td>1,C</td>
<td>5,C</td>
</tr>
<tr>
<td>1</td>
<td>CE</td>
<td>8,C</td>
<td>5,C</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>2</td>
<td>CED</td>
<td>8,C</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>3</td>
<td>CEDF</td>
<td>6,C</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>4</td>
<td>CEDFA</td>
<td>5,A</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...  

**Loop**  
9 find w not in S s.t. D(w) is a minimum;  
10 add w to S;  
11 update D(v) for all v adjacent to w and not in S:  
12 \( D(v) = \min( D(v), D(w) + c(w,v) ) \);  
13 *until all nodes in S*;
## Example: Dijkstra’s Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>start S</th>
<th>D(B),p(B)</th>
<th>D(A),p(A)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>8,C</td>
<td>5,C</td>
<td>3,C</td>
<td>1,C</td>
<td>5,C</td>
</tr>
<tr>
<td>1</td>
<td>CE</td>
<td>8,C</td>
<td>5,C</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>2</td>
<td>CED</td>
<td>8,C</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>3</td>
<td>CEDF</td>
<td>6,D</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>4</td>
<td>CEDFA</td>
<td>5,A</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
<tr>
<td>5</td>
<td>CEDFAB</td>
<td>5,A</td>
<td>3,D</td>
<td>2,E</td>
<td>1,C</td>
<td>3,E</td>
</tr>
</tbody>
</table>

... **Loop**

9 find w not in S s.t. D(w) is a minimum;
10 add w to S;
11 update D(v) for all v adjacent to w and not in S:
12 \[ D(v) = \min( D(v), D(w) + c(w,v) ) \];
13 until all nodes in S;