Mid-term Review

Fall 2017
Internet Protocol Stack

- **Application**: supporting network applications
  - FTP, SMTP, HTTP
- **Transport**: host-host data transfer
  - TCP, UDP
- **Network**: routing of datagrams from source to destination
  - IP, routing protocols
- **Link**: data transfer between neighboring network elements
  - WiFi, Ethernet
- **Physical**: bits "on the wire"
  - Radios, coaxial cable, optical fibers
Physical Layer
Overview of Wireless Transmissions

sender

bit stream

source coding

channel coding

modulation

analog signal

receiver

bit stream

source decoding

channel decoding

demodulation
Shannon Channel Capacity

- The maximum number of bits that can be transmitted per second by a physical channel is:

\[ W \log_2 (1 + \frac{S}{N}) \]

where \( W \) is the frequency range that the media allows to pass through, \( S/N \) is the signal noise ratio.
Signal, Noise, and Interference

• Signal (S)

• Noise (N)
  - Includes thermal noise and background radiation
  - Often modeled as additive white Gaussian noise

• Interference (I)
  - Signals from other transmitting sources

• $\text{SINR} = \frac{S}{N+I}$ (sometimes also denoted as SNR)
Signal Propagation

• Path loss
• Shadowing
• Reflection at large obstacles
• Refraction depending on the density of a medium
• Scattering at small obstacles
• Diffraction at edges
• Fading (depending on the frequency)
Path Loss (Cont.)

- **Free space model**
  \[ P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \]

- **Two-ray ground reflection model**
  \[ P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L} \quad d_c = \frac{(4\pi h_t h_r)}{\lambda} \]

- **Log-normal shadowing**
  \[ P(d)[dB] = \overline{P}(d)[dB] + X_\sigma \]

- **Indoor model**
  \[ P_r(d)[dBm] = P_t(d)[dBm] - 10n \log\left(\frac{d}{d_0}\right) - \left\{ \begin{array}{ll}
    nW * WAF & nW < C \\
    C * WAF & nW \geq C
  \end{array} \right. \]

- **P = 1 mW at d0=1m, what’s Pr at d=2m?**
Multiplexing

• *Goal:* multiple use of a shared medium

• *Multiplexing in 4 dimensions*
Multiplexing

• Goal: multiple use of a shared medium

• Multiplexing in 4 dimensions
  - space \((s_i)\)
  - time \((t)\)
  - frequency \((f)\)
  - code \((c)\)
Modulation and Demodulation

- Digital data
  - 101101001
  - Digital modulation
  - Analog baseband signal
  - Analog modulation
  - Radio carrier
  - Radio transmitter

- Analog demodulation
  - Analog baseband signal
  - Synchronization decision
  - Digital data
  - 101101001
  - Radio receiver
Modulation Schemes
Modulation Schemes

- Amplitude Modulation (AM)
- Frequency Modulation (FM)
- Phase Modulation (PM)
Spread spectrum technology
Spread spectrum technology

- Problem of radio transmission: frequency dependent fading can wipe out narrow band signals for duration of the interference
- Solution: spread the narrow band signal into a broad band signal using a special code

- Side effects:
  - coexistence of several signals without dynamic coordination
  - tap-proof
- Alternatives: Direct Sequence, Frequency Hopping
DSSS
(Direct Sequence Spread Spectrum)

• XOR of the signal with pseudo-random number (chipping sequence)
  - generate a signal with a wider range of frequency: spread spectrum

\[
\begin{align*}
0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{align*}
\]

\[\text{XOR} = 0 1 1 0 0 1 0 1 = t_b \]

\[\begin{align*}
t_b &: \text{ bit period} \\
t_c &: \text{ chip period}
\end{align*}\]
FHSS
(Frequency Hopping Spread Spectrum)

• **Discrete changes of carrier frequency**
  - sequence of frequency changes determined via pseudo random number sequence

• **Two versions**
  - Fast Hopping: several frequencies per user bit
  - Slow Hopping: several user bits per frequency

• **Advantages**
  - frequency selective fading and interference limited to short period
  - simple implementation
  - uses only small portion of spectrum at any time
Link Layer
IEEE 802.11 DCF

• DCF is CSMA/CA protocol
  - Why?
• DCF suitable for multi-hop ad hoc networking
• Optionally uses RTS-CTS exchange to avoid hidden terminal problem
  - Any node overhearing a CTS cannot transmit for the duration of the transfer
• Uses ACK to provide reliability
CSMA/CA

- **Why not CSMA/CD?**
  - Impossible to detect collision using half-duplex radios
  - Hidden terminal
- **Solution: CSMA/CA**
- **Carrier sense**
  - Nodes hearing RTS or CTS stay silent for the duration of the corresponding transmission.
  - Physical carrier sense
    - Carrier sense threshold
  - Virtual carrier sense using Network Allocation Vector (NAV)
    - NAV is updated based on overheard RTS/CTS/DATA/ACK packets
- **Collision avoidance**
  - Once channel becomes idle, the node waits for a randomly chosen duration before attempting to transmit.
IEEE 802.11

Interference “range”

Carrier sense range

Transmit “range”

DATA

A

B

C

D

E

F
IEEE 802.11: CSMA/CA

- DATA packet, followed by ACK.
Hidden Terminal Problem
Hidden Terminal Problem

- B can communicate with both A and C
- A and C cannot hear each other

Problem
- When A transmits to B, C cannot detect the transmission using the carrier sense mechanism
- If C transmits, collision will occur at node B

Solution
- Hidden sender C needs to defer

How does 802.11 solve hidden terminal problem?
Solution for Hidden Terminal Problem: MACA

• When A wants to send a packet to B, A first sends a Request-to-Send (RTS) to B

• On receiving RTS, B responds by sending Clear-to-Send (CTS), provided that A is able to receive the packet

• When C overhears a CTS, it keeps quiet for the duration of the transfer
  - Transfer duration is included in both RTS and CTS
IEEE 802.11

RTS = Request-to-Send

Pretending a circular range
IEEE 802.11

RTS = Request-to-Send

NAV = remaining duration to keep quiet
IEEE 802.11

CTS = Clear-to-Send
IEEE 802.11

CTS = Clear-to-Send

NAV = 8
IEEE 802.11

• DATA packet follows CTS. Successful data reception acknowledged using ACK.
IEEE 802.11
IEEE 802.11

Reserved area

A
B
C
D
E
F
Backoff Interval

• When transmitting a packet, choose a backoff interval in the range $[0, CW]$  
  - $CW$ is contention window
• Count down the backoff interval when medium is idle  
  - Count-down is suspended if medium becomes busy
• Transmit when backoff interval reaches 0
DCF Example

B1 and B2 are backoff intervals at nodes 1 and 2

cw = 31

B1 = 25

B1 = 5

B2 = 20

B2 = 15

B2 = 10
Backoff Interval

• The time spent counting down backoff intervals is a part of MAC overhead

• Important to choose CW appropriately
  - large CW $\Rightarrow$ large overhead
  - small CW $\Rightarrow$ may lead to many collisions (when two nodes count down to 0 simultaneously)
Backoff Interval (Cont.)

- Since the number of nodes attempting to transmit simultaneously may change with time, some mechanism to manage contention is needed.

- IEEE 802.11 DCF: contention window $CW$ is chosen dynamically depending on collision occurrence.
Binary Exponential Backoff in DCF

- When a node fails to receive CTS in response to its RTS, it increases the contention window
  - \( CW \) is doubled (up to an upper bound)
  - More collisions \( \Rightarrow \) longer waiting time to reduce collision

- When a node successfully completes a data transfer, it restores \( CW \) to \( CW_{\text{min}} \)
Network Layer
Routing

Routing protocol

Goal: determine “good” path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level

“good” path:
- typically means minimum cost path
- other def’s possible
Routing Algorithm classification

Global or decentralized information?
Global:
• all routers have complete topology, link cost info
• “link state” algorithms
Decentralized:
• router knows physically-connected neighbors, link costs to neighbors
• iterative process of computation, exchange of info with neighbors
• “distance vector” algorithms

Static or dynamic?
Static:
• routes change slowly over time
Dynamic:
• routes change more quickly
  - periodic update
  - in response to link cost changes
A Link-State Routing Algorithm

Dijkstra’s algorithm

- net topology, link costs known to all nodes
  - accomplished via “link state broadcast”
  - all nodes have same info
- computes least cost paths from one node (‘source”) to all other nodes
  - gives routing table for that node
- iterative: after k iterations, know least cost path to k dest.’s

Notation:

- $c(i,j)$: link cost from node $i$ to $j$. cost infinite if not direct neighbors
- $D(v)$: current value of cost of path from source to dest. $V$
- $p(v)$: predecessor node along path from source to $v$, that is next $v$
- $N$: set of nodes whose least cost path definitively known
Dijsktra’s Algorithm

1 **Initialization:**
2 \( N = \{A\} \)
3 for all nodes \( v \)
4   if \( v \) adjacent to \( A \)
5     then \( D(v) = c(A,v) \)
6   else \( D(v) = \text{infinity} \)
7
8 **Loop**
9   find \( w \) not in \( N \) such that \( D(w) \) is a minimum
10  add \( w \) to \( N \)
11  update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N \):
12    \( D(v) = \text{min}( D(v), D(w) + c(w,v) ) \)
13    /* new cost to \( v \) is either old cost to \( v \) or known shortest path cost to \( w \) plus cost from \( w \) to \( v \) */
14  until all nodes in \( N \)
Dijkstra's algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>start N</th>
<th>D(B),p(B)</th>
<th>D(C),p(C)</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>A</td>
<td></td>
<td></td>
<td></td>
<td>1,A</td>
<td></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>

- **Graph Representation**
  - Nodes: A, B, C, D, E, F
  - Edges: A-B (2), B-C (3), C-D (3), D-E (1), E-F (2), A-D (1)
  - Red edges represent the path chosen in each step.
# Dijkstra's algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>Start N</th>
<th>$D(B), p(B)$</th>
<th>$D(C), p(C)$</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>A</td>
<td>2,A</td>
<td>5,A</td>
<td>1,A</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2,A</td>
<td>4,D</td>
<td>2,D</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2,A</td>
<td>3,E</td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td></td>
<td>3,E</td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td></td>
<td></td>
<td></td>
<td>4,E</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph](image)
Distance Vector Routing Algorithm

**iterative:**
- continues until no nodes exchange info.
- *self-terminating:* no "signal" to stop

**asynchronous:**
- nodes need *not* exchange info/iterate in lock step!

**distributed:**
- each node communicates *only* with directly-attached neighbors

**Each node:**
- *wait* for (change in local link cost of msg from neighbor)
- *recompute* distance table
- if least cost path to any dest has changed, *notify* neighbors
Distance Vector Algorithm: Data Structures

- **Each node** \( x \) **maintains:**
  - For each neighbor \( v \), cost \( c(x,v) \)
  - Node \( x \)'s distance vector: \( D_x = [D_x(y): y \in N] \)
    containing \( x \)'s estimate of cost to all destinations
  - Distance vectors for each neighbor \( v \): \( D_v = [D_v(y): y \in N] \)

- **Basic operation:** Bellman-Ford algorithm
  \[
  D_x(y) = \min_v \{c(x,v) + D_v(y)\} \quad y \in N
  \]
Distance Vector Algorithm:

At all nodes, X:

1 Initialization:
2 For all destinations \( y \in N \):
3 \( D_x(y) = c(x,y) \) /* if \( y \) is not a neighbor, then \( c(x,y) = \infty */:
4 For each neighbor \( w \)
5 \( D_w(y) = \infty \) for all destinations \( y \in N \)
6 For each neighbor \( w \)
7 Send distance vector \( D_x = [D_x(y): y \in N] \) to \( w \)

8 Loop:
9 Wait (until communication from neighbor \( w \))
10 For each \( y \in N \):
11 \( D_x(y) = \min_v \{ c(x,v)+ D_v(y) \} \)
12 If \( D_x(y) \) changes for any destination \( y \)
13 Send distance vector \( D_x = [D_x(y): y \in N] \) to all neighbors
Distance Vector: link cost changes

Link cost changes:
• node detects local link cost change
• updates routing info, recalculates distance vector
• if DV changes, notify neighbors

At time $t_0$, $y$ detects the link-cost change, updates its DV, and informs its neighbors.

At time $t_1$, $z$ receives the update from $y$ and updates its table. It computes a new least cost to $x$ and sends its neighbors its DV.

At time $t_2$, $y$ receives $z$’s update and updates its distance table. $y$’s least costs do not change and hence $y$ does not send any message to $z$. 

“good news travels fast”
Distance Vector: link cost changes
Distance Vector: link cost changes

\[ \text{D}_y(x) = \min \{ c(y,x) + \text{D}_x(x), c(y,z) + \text{D}_z(x) \} \]
\[ = \min \{60+0, \ 1+5\} = 6 \]
Distance Vector: link cost changes

Link cost changes:
- good news travels fast
- bad news travels slow - “count to infinity” problem!
- 44 iterations before algorithm stabilizes

Poissoned reverse:
- If Z routes through Y to get to X:
  - Z tells Y its (Z’s) distance to X is infinite (so Y won’t route to X via Z)
- will this completely solve count to infinity problem?
Comparison of LS and DV algorithms

Message complexity
- **LS**: with \( n \) nodes, \( E \) links, \( O(nE) \) msgs sent each
- **DV**: exchange between neighbors only
  - convergence time varies

Speed of Convergence
- **LS**: \( O(n^2) \) algorithm requires \( O(nE) \) msgs
  - may have oscillations
- **DV**: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?
- **LS**:
  - node can advertise incorrect *link* cost
  - each node computes only its own table
- **DV**:
  - DV node can advertise incorrect *path* cost
  - each node’s table used by others
    - error propagate thru network
Mobile IP

- HA: Home Agent
- MN: Mobile Node
- FA: Foreign Agent
- CN: Correspondent Node
- Internet
- Home network
  - (physical home network for the MN)
- Foreign network
  - (current physical network for the MN)
- Mobile end-system
Data transfer to the mobile system

1. Sender sends to the IP address of MN, HA intercepts packet (proxy ARP)
2. HA tunnels packet to COA, here FA, by encapsulation
3. FA forwards the packet to the MN
Data transfer from the mobile system

1. Sender sends to the IP address of the receiver as usual, FA works as default router.
Network Layer: routing

- Mobile ad hoc networks
- Mesh networks
- Sensor networks
- Delay tolerant networks
Name one difference and one commonality between DSR and ADOV
## Comparison between DSR and AODV

<table>
<thead>
<tr>
<th></th>
<th>DSR</th>
<th>AODV</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Commonality</strong></td>
<td>- Reactive routing</td>
<td>- Distance vector routing</td>
</tr>
<tr>
<td></td>
<td>- Discover routes via flooding</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Obtain routes from route reply</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Send route error upon failure detection</td>
<td>- Proactively delete expired route cache</td>
</tr>
<tr>
<td><strong>Differences</strong></td>
<td>Source routing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Aggressive cache routes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Distance vector routing</td>
<td></td>
</tr>
</tbody>
</table>
Routing in Mesh Networks

• What are ETX and ETT?
Routing in Mesh Networks

- **ETX**
  - The predicted number of data transmissions required to send a packet over a link
  - Expected probability that a transmission is successfully received and acknowledged is $d_f \times d_r$
    - $d_f$ is forward delivery ratio
    - $d_r$ is reverse delivery ratio
  - Each attempt to transmit a packet is a Bernoulli trial, so...

\[
\text{ETX} = \frac{1}{d_f \times d_r}
\]

- **ETT**: extension of ETX
Sample Question 1

• Why do we have protocol layers?
  - Pros
    • Abstraction
    • Easy to reuse
    • Easy to maintain
  - Cons
    • Overhead with layering
    • Less transparent
    • Hard to optimize and troubleshoot
Sample Question 2

• $P = 1000 \text{ mW}$. What is $P$ in dBm?

• $P_2 = 0.01 P_1$. What $P_2/P_1$ in dB?
Sample Question 3

• Why we don’t use CSMA/CD in wireless networks?
Sample Question 4

- DSR is designed for mobile ad hoc networks. If we use it in wireless mesh networks, how well does it perform?