Network Layer 1

ELEC3222 Computer Networks

Alex Weddell
asw@ecs.soton.ac.uk
The 5-layer Model

- Network is **below** the transport layer and **above** the link layer
- **Provides services** to the Transport layer
  - Packet delivery/switching
- **Relies on services** provided by the Link layer
  - Interconnects between adjacent network nodes
- Responsible for delivering **packets** between endpoints over **multiple links**
Overview

- Packet switching
- Datagrams/Virtual Circuits
- Routing
Store-and-Forward Packet Switching

- **Hosts** send **packets** into the network
- Packets are **forwarded** by **routers**
Connectionless Service - Datagrams

- Packet is forwarded using destination address inside it
  - Different packets may take different paths
Connection-Oriented – Virtual Circuits

- Packet is forwarded along a virtual circuit using tag inside it
- Virtual circuit (VC) is set up ahead of time
# Comparison of Virtual Circuits/Datagrams

<table>
<thead>
<tr>
<th>Issue</th>
<th>Datagram network</th>
<th>Virtual-circuit network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit setup</td>
<td>Not needed</td>
<td>Required</td>
</tr>
<tr>
<td>Addressing</td>
<td>Each packet contains the full source and destination address</td>
<td>Each packet contains a short VC number</td>
</tr>
<tr>
<td>State information</td>
<td>Routers do not hold state information about connections</td>
<td>Each VC requires router table space per connection</td>
</tr>
<tr>
<td>Routing</td>
<td>Each packet is routed independently</td>
<td>Route chosen when VC is set up; all packets follow it</td>
</tr>
<tr>
<td>Effect of router failures</td>
<td>None, except for packets lost during the crash</td>
<td>All VCs that passed through the failed router are terminated</td>
</tr>
<tr>
<td>Quality of service</td>
<td>Difficult</td>
<td>Easy if enough resources can be allocated in advance for each VC</td>
</tr>
<tr>
<td>Congestion control</td>
<td>Difficult</td>
<td>Easy if enough resources can be allocated in advance for each VC</td>
</tr>
</tbody>
</table>
Routing

- **Routing** is the process of discovering network paths
  - Model the network as a graph of nodes and links
  - Decide what to optimize (e.g., fairness vs efficiency)
  - Update routes for changes in topology (e.g., failures)

- **Forwarding** is the sending of packets along a path
Routing

- Optimality Principle
- Shortest Path Algorithm
- Flooding
- Distance Vector Routing
- Link State Routing
- Hierarchical Routing
- Broadcast/Multicast/Anycast Routing
- Routing in Ad-Hoc Networks
The Optimality Principle

- Each portion of a best path is also a best path; the union of them to a router is a tree called the sink tree
  - “Best” means fewest hops in the example

Network

Sink tree of best paths to router B
Shortest Path Algorithm

- **Dijkstra’s algorithm** computes a sink tree on the graph:
  - Each link is assigned a non-negative weight/distance
  - Shortest path is the one with lowest total weight
  - Using weights of 1 gives paths with fewest hops

- **Algorithm:**
  - Start with sink, set distance at other nodes to infinity
  - Relax distance to other nodes
  - Pick the lowest distance node, add it to sink tree
  - Repeat until all nodes are in the sink tree
Shortest Path Algorithm

- A network, and first five steps in computing the shortest paths from A to D. Pink arrows show the sink tree so far.

(a) A network with nodes A, B, C, D, E, F, G, and H. Pink arrows show the sink tree.

(b) The network with updated nodes B(2, A), C(∞, B), E(∞, -), F(∞, -), and D(∞, -).

(c) The network with updated nodes A, B(2, A), C(9, B), E(4, B), and F(∞, -).

(d) The network with updated nodes B(2, A), C(9, B), E(4, B), F(6, E), and D(∞, 1).

(e) The network with updated nodes A, B(2, A), C(9, B), E(4, B), F(6, E), and D(∞, -).

(f) The network with updated nodes B(2, A), C(9, B), E(4, B), F(6, E), and H(8, F).
Shortest Path Algorithm

...  

```c
for (p = &state[0]; p < &state[n]; p++) {
    p->predecessor = -1;
    p->length = INFINITY;
    p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t;
do {
    for (i = 0; i < n; i++)
        if (dist[k][i] != 0 && state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
        }
}
...  
```

Start with the sink, all other nodes are unreachable

Relaxation step. Lower distance to nodes linked to newest member of the sink tree
Shortest Path Algorithm

\[
\begin{align*}
&k = 0; \text{min} = \text{INFINITY}; \\
&\text{for } (i = 0; i < n; i++) \\
&\quad \text{if } (\text{state}[i].\text{label} == \text{tentative} \&\& \text{state}[i].\text{length} < \text{min}) \\
&\quad \quad \text{min} = \text{state}[i].\text{length}; \\
&\quad \quad k = i; \\
&\quad \}\}
&\text{state}[k].\text{label} = \text{permanent}; \\
&\} \text{while } (k \neq s); \\
\end{align*}
\]

Find the lowest distance, add it to the sink tree, and repeat until done.
Flooding

- A simple method to send a packet to all network nodes
- Each node floods a new packet received on an incoming link by sending it out to all of the other links
- Nodes need to keep track of flooded packets to stop the flood; even using a hop limit can blow up exponentially
Distance Vector Routing

• **Distance vector** is a distributed routing algorithm
  – Shortest path computation is split across nodes

• **Algorithm:**
  – Each node knows **distance of links to its neighbors**
  – Each node **advertises** vector of lowest known distances to all neighbors
  – Each node uses received vectors to **update** its own
  – Repeat periodically
Distance Vector Routing

Network

Vectors received at J from Neighbors A, I, H and K

New vector for J

New estimated delay from J

Line

To | A | I | H | K
---|---|---|---|---
A  | 0 | 24 | 20 | 21
B  | 12 | 36 | 31 | 28
C  | 25 | 18 | 19 | 36
D  | 40 | 27 | 8  | 24
E  | 14 | 7  | 30 | 22
F  | 23 | 20 | 19 | 40
G  | 18 | 31 | 6  | 31
H  | 17 | 20 | 0  | 19
I  | 21 | 0  | 14 | 22
J  | 9  | 11 | 7  | 10
K  | 24 | 22 | 22 | 0
L  | 29 | 33 | 9  | 9

JA delay is 8
JI delay is 10
JH delay is 12
JK delay is 6
The “Count-to-Infinity” Problem

- Failures can cause DV to “count to infinity” while seeking a path to an unreachable node

Good news of a path to A spreads quickly

Bad news of no path to A is learned slowly
Link State Routing

- **Link state** is an alternative to distance vector
  - More computation but simpler dynamics
  - Widely used in the Internet (OSPF, ISIS)

- Algorithm:
  - Each node **floods** information about its neighbors in LSPs (Link State Packets)
  - All nodes learn the **full network graph**
  - Each node **runs Dijkstra’s algorithm** to compute the path to take for each destination
Link State Routing

- LSP (Link State Packet) for a node lists neighbors and weights of links to reach them.
Link State Routing

- Seq. number and age are used for reliable flooding
  - New LSPs are acknowledged on the lines they are received and sent on all other lines
  - Example shows the LSP database at router B

<table>
<thead>
<tr>
<th>Source</th>
<th>Seq.</th>
<th>Age</th>
<th>Send flags</th>
<th>ACK flags</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>A  C  F</td>
<td>A  C  F</td>
</tr>
<tr>
<td>A</td>
<td>21</td>
<td>60</td>
<td>0  1  1</td>
<td>1  0  0</td>
</tr>
<tr>
<td>F</td>
<td>21</td>
<td>60</td>
<td>1  1  0</td>
<td>0  0  1</td>
</tr>
<tr>
<td>E</td>
<td>21</td>
<td>59</td>
<td>0  1  0</td>
<td>1  0  1</td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>60</td>
<td>1  0  1</td>
<td>0  1  0</td>
</tr>
<tr>
<td>D</td>
<td>21</td>
<td>59</td>
<td>1  0  0</td>
<td>0  1  1</td>
</tr>
</tbody>
</table>
Hierarchical Routing

- Hierarchical routing reduces the work of route computation but may result in slightly longer paths than flat routing

**Full table for 1A**

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Line</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1B</td>
<td>1B</td>
<td>1</td>
</tr>
<tr>
<td>1C</td>
<td>1C</td>
<td>1</td>
</tr>
<tr>
<td>2A</td>
<td>1B</td>
<td>2</td>
</tr>
<tr>
<td>2B</td>
<td>1B</td>
<td>3</td>
</tr>
<tr>
<td>2C</td>
<td>1B</td>
<td>3</td>
</tr>
<tr>
<td>2D</td>
<td>1B</td>
<td>4</td>
</tr>
<tr>
<td>3A</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>3B</td>
<td>1C</td>
<td>2</td>
</tr>
<tr>
<td>4A</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>4B</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>4C</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>5A</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>5B</td>
<td>1C</td>
<td>5</td>
</tr>
<tr>
<td>5C</td>
<td>1B</td>
<td>5</td>
</tr>
<tr>
<td>5D</td>
<td>1C</td>
<td>6</td>
</tr>
<tr>
<td>5E</td>
<td>1C</td>
<td>5</td>
</tr>
</tbody>
</table>

**Hierarchical table for 1A**

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Line</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1B</td>
<td>1B</td>
<td>1</td>
</tr>
<tr>
<td>1C</td>
<td>1C</td>
<td>1</td>
</tr>
<tr>
<td>2A</td>
<td>1B</td>
<td>2</td>
</tr>
<tr>
<td>2B</td>
<td>1B</td>
<td>3</td>
</tr>
<tr>
<td>2C</td>
<td>1B</td>
<td>3</td>
</tr>
<tr>
<td>2D</td>
<td>1B</td>
<td>4</td>
</tr>
<tr>
<td>3A</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>3B</td>
<td>1C</td>
<td>2</td>
</tr>
<tr>
<td>4A</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>4B</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>4C</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>5A</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>5B</td>
<td>1C</td>
<td>5</td>
</tr>
<tr>
<td>5C</td>
<td>1B</td>
<td>5</td>
</tr>
<tr>
<td>5D</td>
<td>1C</td>
<td>6</td>
</tr>
<tr>
<td>5E</td>
<td>1C</td>
<td>5</td>
</tr>
</tbody>
</table>

- Best choice to reach nodes in 5 except for 5C
Broadcast Routing

- **Broadcast** sends a packet to all nodes
  - RPF (Reverse Path Forwarding): send broadcast received on the link to the source out to all remaining links
  - Alternatively, can build and use sink trees at all nodes
Multicast Routing

- **Multicast** sends to a subset of the nodes called a **group**
  - Uses a different tree for each group and source

**Network with groups 1 & 2**

**Spanning tree from source S**

**Multicast tree from S to group 1**

**Multicast tree from S to group 2**
Anycast Routing

- **Anycast** sends a packet to one (nearest) group member
  - Falls out of regular routing with a node in many places

Anycast routes to group 1

Apparent topology of sink tree to “node” 1
Routing for Mobile Hosts

- Mobile hosts can be reached via a home agent
  - Fixed home agent tunnels packets to reach the mobile host; reply can optimize path for subsequent packets
  - No changes to routers or fixed hosts
Routing in Ad-Hoc Networks

- The network topology changes as wireless nodes move
  - Routes are often made on demand, e.g., AODV (below)

A’s starts to find route to I
A’s broadcast reaches B & D
B’s and D’s broadcast reach C, F & G
C’s, F’s and G’s broadcast reach H & I
Summary

- Packet switching
- Datagrams/Virtual Circuits
- Routing