L11: Link and Network layer

Dina Katabi
6.033 Spring 2007
http://web.mit.edu/6.033

Some slides are from lectures by Nick McKeown, Ion Stoica, Frans Kaashoek, Hari Balakrishnan, Sam Madden, and Robert Morris
Last lecture: layering of protocols

- Each layer adds/strips off its own header
- Each layer may split up higher-level data
- Each layer multiplexes multiple higher layers
- Each layer is (mostly) transparent to higher layers
Problem:
Deliver data from one end of the link to the other

Need to address:
- Bits $\rightarrow$ Analog $\rightarrow$ Bits
- Framing
- Errors
- Medium Access Control (The Ethernet Paper)
Manchester encoding

- Each bit is a transition
- Allows the receiver to sync to the sender’s clock
Framing

• Receiver needs to detect the beginning and the end of a frame
• Use special bit-pattern to separate frames
  • E.g., pattern could be 1111111 (7 ones)
• Bit stuffing is used to ensure that a special pattern does not occur in the data
  • If pattern is 1111111 → Whenever the sender sees a sequence of 6 ones in the data, it inserts a zero (reverse this operation at receiver)
Error Handling

- **Detection:**
  - Use error detection codes, which add some redundancy to allow detecting errors

- **When errors are detected**
  - **Correction:**
    - Some codes allow for correction
  - **Retransmission:**
    - Can have the link layer retransmit the frame (rare)
  - **Discard:**
    - Most link layers just discard the frame and rely on higher layers to retransmit
Network Layer:
finds a path to the destination and forwards packets along that path

- Difference between routing and forwarding
  - Routing is finding the path
  - Forwarding is the action of sending the packet to the next-hop toward its destination
Forwarding

- Each router has a forwarding table
- Forwarding tables are created by a routing protocol

**Forwarding table at R**

<table>
<thead>
<tr>
<th>Dst. Addr</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
</tr>
</tbody>
</table>
Inside a router

Forwarding Table

Forwarding Decision

Choose Egress

Choose Egress

Choose Egress

Choose Egress
The Internet Protocol (IP)

Protocol Stack

- App
- Transport
- Network
- Link

TCP / UDP

Data

IP

Data

TCP packet

IP packet

Data

Hdr

Data

Hdr
# The IP Header

<table>
<thead>
<tr>
<th>vers</th>
<th>HLen</th>
<th>TOS</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Flags</th>
<th>FRAG Offset</th>
<th>TTL</th>
<th>Protocol</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SRC IP Address</th>
<th>DST IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(OPTIONS)</th>
<th>(PAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Hop count**
Forwarding an IP Packet

- Lookup packet’s DST in forwarding table
  - If known, find the corresponding outgoing link
  - If unknown, drop packet
- Decrement TTL and drop packet if TTL is zero; update header Checksum
- Forward packet to outgoing port
- Transmit packet onto link
And switches today...

- **Alcatel 7670 RSP**
- **Juniper TX8/T640 TX8**
- **Avici TSR**
- **Cisco GSR 12416**
  - 6ft x 2ft x 1.5ft
  - 4.2 kW power
  - 160 Gb/s cap.
- **Lucent 5ESS telephone switch**
The Routing Problem:

- Generate forwarding tables

Goals: No loops, short paths, etc.
THE ARPA NETWORK

DEC 1969

4 NODES

FIGURE 6.2 Drawing of 4 Node Network
(Courtesy of Alex McKenzie)
ARPA NET LOGICAL MAP, MARCH 1977

(Please note that while this map shows the host population of the network according to the best information obtainable, no claim can be made for its accuracy.)

Names shown are IMP names, not necessarily host names.
Path Vector Routing Protocol

- Initialization
  - Each node knows the path to itself

For example, D initializes its paths

<table>
<thead>
<tr>
<th>DST</th>
<th>Link</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>End layer</td>
<td>null</td>
</tr>
</tbody>
</table>
Path Vector

- Step 1: Advertisement
  - Each node tells its neighbors its path to each node in the graph

For example, D receives:

<table>
<thead>
<tr>
<th>From A:</th>
<th>From C:</th>
<th>From E:</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Path</td>
<td>To</td>
</tr>
<tr>
<td>A</td>
<td>null</td>
<td>C</td>
</tr>
</tbody>
</table>
Path Vector

- Step 2: Update Route Info
  - Each node use the advertisements to update its paths

\[
\begin{align*}
\text{D received:} & \quad \text{From A:} & \quad \text{From C:} & \quad \text{From E:} \\
& \quad \text{To} & \quad \text{Path} & \quad \text{To} & \quad \text{Path} & \quad \text{To} & \quad \text{Path} \\
& A \quad & \text{null} & C \quad & \text{null} & E \quad & \text{null} \\
\end{align*}
\]

D updates its paths:

\[
\begin{align*}
\text{DST} & \quad \text{Link} & \quad \text{Path} \\
D & \quad \text{End layer} & \text{null} \\
A & 1 & <A> \\
C & 3 & <C> \\
E & 2 & <E> \\
\end{align*}
\]

Note: At the end of first round, each node has learned all one-hop paths
Path Vector

- Periodically repeat Steps 1 & 2

In round 2, D receives:

<table>
<thead>
<tr>
<th>From A:</th>
<th>From C:</th>
<th>From E:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>To</strong></td>
<td><strong>Path</strong></td>
<td><strong>To</strong></td>
</tr>
<tr>
<td>A</td>
<td>null</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>&lt;D&gt;</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

D updates its paths:

<table>
<thead>
<tr>
<th>DST</th>
<th>Link</th>
<th>Path</th>
<th>DST</th>
<th>Link</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>End layer</td>
<td>null</td>
<td>D</td>
<td>End layer</td>
<td>null</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>&lt;A&gt;</td>
<td>A</td>
<td>1</td>
<td>&lt;A&gt;</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>&lt;C&gt;</td>
<td>C</td>
<td>3</td>
<td>&lt;C&gt;</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>&lt;E&gt;</td>
<td>E</td>
<td>2</td>
<td>&lt;E&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>3</td>
<td>&lt;C, B&gt;</td>
</tr>
</tbody>
</table>

Note: At the end of round 2, each node has learned all two-hop paths
Questions About Path Vector

• How do we avoid permanent loops?

• What happens when a node hears multiple paths to the same destination?

• What happens if the graph changes?
Questions About Path Vector

- How do we ensure no loops?
  - When a node updates its paths, it never accepts a path that has itself

- What happens when a node hears multiple paths to the same destination?
  - It picks the better path (e.g., the shorter number of hops)

- What happens if the graph changes?
  - Algorithm deals well with new links
  - To deal with links that go down, each router should discard any path that a neighbor stops advertising
Hierarchical Routing

- Internet: collection of domains/networks
- Inside a domain: Route over a graph of routers
- Between domains: Route over a graph of domains
- Address consists of “Domain Id”, “Node Id”
Hierarchical Routing

**Advantage**
- Scalable
  - Smaller tables
  - Smaller messages
- Delegation
  - Each domain can run its own routing protocol

**Disadvantage**
- Mobility is difficult
  - Address depends on geographic location
- Sup-optimal paths
  - E.g., in the figure, the shortest path between the two machines should traverse the yellow domain.
Routing: many open issues

• Flat addresses and scalable?

• Routing in multihop WiFi networks?

• Routing in peer-to-peer networks?

• Misconfigurations between domains?