CS118 Discussion 1B, Week 7

Zhehui Zhang
HAINES A2 / Friday / 12:00pm-1:50pm
Outline

- Network data plane
  - Fragmentation, DHCP, NAT, IPv6, Openflow
- Network control plane
  - Routing
    - Link state routing
    - Distance vector routing
- Project 2
Quick question on fragmentation

- Consider following IP packet

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>TOS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>12345</td>
<td>2400</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>6</td>
<td>checksum</td>
<td></td>
</tr>
<tr>
<td>10.1.1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80.233.250.61</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Assume MTU = 1450 Bytes. Show the header length, total length, identification, flags, fragment offset, TTL, and IP payload size.
Consider following IP packet

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>TOS</th>
<th>2400</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>12345</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>6</td>
<td>checksum</td>
<td></td>
</tr>
<tr>
<td>10.1.1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80.233.250.61</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assume MTU = 1450 Bytes. Show the header length, total length, identification, flags, fragment offset, TTL, and IP payload size.

For the first packet: 5 (20 bytes), 1444 bytes, ID = 12345, MF = 1, Offset = 0, TTL = 25, 1424 bytes.

For the second packet: 5 (20 bytes), 976 bytes, ID = 12345, MF = 0, Offset = 178, TTL = 25, 956 bytes.
DHCP: Dynamic Host Configuration Protocol

- Dynamically allocates the following info to a host
  - IP address on subnet for the host
  - IP address for default router (“first-hop” router)
  - Subnet mask
  - IP address and name for DNS caching resolver
- Allows address reuse
DHCP: operations

• Host broadcasts “DHCP discovery” msg [optional]

• DHCP server responds with “DHCP offer” msg [optional]

• Host requests IP address: “DHCP request” msg

• DHCP server sends address: “DHCP ack” msg

Important example on Chapter 4 slides 45—46!
Quick question

- What information in DHCP will be updated if PC0 move to PC1?
- How about PC0 move to PC3?
NAT (network address translation)

- Depletion of IPv4 addresses — short-term solution
  - IP tunneling?
- Use private IP addresses
- Side-benefit: security
- How to achieve?
  - <public IP:port> — <private IP:port> mapping
Quick question

<table>
<thead>
<tr>
<th>NAT translation table</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAN side addr</td>
</tr>
<tr>
<td>138.76.29.7, 5001</td>
</tr>
<tr>
<td>......</td>
</tr>
</tbody>
</table>

1: host 10.0.0.1 sends packet to 128.119.40, 80
NAT: downside

- Increased complexity
- Single point of failure
- Cannot run services inside a NAT box
  - Why?
IPv6

IPv6 Header Format (RFC 2460)
IPv6/IPv4 differences

- Fixed-length 40 byte header
  - length field excludes header
  - Header Length field eliminated
- Address length: 128 bits
- Priority: usage yet to be finalized
- Flow Label: identify packets in same flow
- Next header: identify upper layer protocol for data
- Options: outside of the basic header, indicated by Next Header field
- Header Checksum: removed
IPv6 address format (optional)

  - Can skip leading zeros of each word: 2607:F010:3f9:0:0:0:4:1
  - Can skip one sequence of zero words (compressed representation), e.g., 2607:f010:3f9::4:1
  - Can leave the last 32 bits in dot-decimal: 2607:f010:3f9::0.4.0.1
  - Can specify a prefix by /length: 2607:f010:3f9::/64
Special IPv6 addresses (optional)

- ::/128 - Unspecified
- ::1/128 - Loopback
- ::ffff:0:0/96 - IP4-mapped address
- 2002::/16 - 6to4
- ff00::/8 - Multicast
- fe80::/10 - Link-Local Unicast
Open flow example

<table>
<thead>
<tr>
<th>match</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Src = 10.3.<em>.</em></td>
<td>forward(3)</td>
</tr>
<tr>
<td>IP Dst = 10.2.<em>.</em></td>
<td></td>
</tr>
</tbody>
</table>

Forward incoming packets on port 2 to host h3 and h4?
Routing: concepts

• Global or decentralized information?
  - global: all routers have complete topology, link cost info
  - algorithm?
Routing: concepts

- Global or decentralized information?
  - global: all routers have complete topology, link cost info
  - “link state” algorithms
Routing: concepts

• 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
    
    • algorithm?
Routing: concepts

• 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
Link state routing

- Dijkstra’s algorithm
  - net topology, link costs known to all nodes
  - computes least cost paths from one node (‘source”) to all other nodes
  - iterative: after k iterations, know least cost path to k destinations
Link state routing: algorithm

1  Initialization:
2    N' = \{u\}
3    for all nodes v
4      if v adjacent to u
5        then D(v) = c(u,v)
6      else D(v) = \infty
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      [Link cost update heuristic from Dijkstra algo.]
13  until all nodes in N'

\( c(x, y) \): link cost from node x to y; \( c(x, y) = \infty \) if not direct neighbors
\( D(v) \): current value of cost of path from source to destination v
\( p(v) \): predecessor node along path from source to v
\( N' \): set of nodes whose least cost path definitively known
Link state routing: algorithm

1 Initialization:
2  \( N' = \{u\} \)
3  for all nodes \( v \)
4    if \( v \) adjacent to \( u \)
5      then \( D(v) = c(u,v) \)
6    else \( D(v) = \infty \)
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    \[ D(v) = \min(D(v), D(w) + c(w,v)) \]
13  until all nodes in \( N' \)

c(x, y): link cost from node \( x \) to \( y \); \( c(x, y) = \infty \) if not direct neighbors
D(v): current value of cost of path from source to destination \( v \)
p(v): predecessor node along path from source to \( v \)
N': set of nodes whose least cost path definitively known
Link state routing: example

• Using link state routing to setup a forwarding table for node $u$
### Let’s work it out

<table>
<thead>
<tr>
<th>$N'$</th>
<th>$D(v), p(v)$</th>
<th>$D(w), p(w)$</th>
<th>$D(x), p(x)$</th>
<th>$D(y), p(y)$</th>
<th>$D(z), p(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$</td>
<td>2, $u$</td>
<td>5, $u$</td>
<td>1, $u$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$ux$</td>
<td>2, $u$</td>
<td>4, $x$</td>
<td></td>
<td>2, $x$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$uxy$</td>
<td>2, $u$</td>
<td>3, $y$</td>
<td></td>
<td></td>
<td>4, $y$</td>
</tr>
<tr>
<td>$uxyv$</td>
<td></td>
<td>3, $y$</td>
<td></td>
<td></td>
<td>4, $y$</td>
</tr>
<tr>
<td>$uxyw$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4, $y$</td>
</tr>
<tr>
<td>$uxyvwz$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Let’s work it out

<table>
<thead>
<tr>
<th>N’</th>
<th>D(v), p(v)</th>
<th>D(w), p(w)</th>
<th>D(x), p(x)</th>
<th>D(y), p(y)</th>
<th>D(z), p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>2, u</td>
<td>5, u</td>
<td>1, u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>ux</td>
<td>2, u</td>
<td>4, x</td>
<td></td>
<td>2, x</td>
<td>∞</td>
</tr>
<tr>
<td>uxy</td>
<td>2, u</td>
<td>3, y</td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyv</td>
<td></td>
<td>3, y</td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyvw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyvwz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Link state routing: complexity

- size: n nodes
- each iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(n \log n)$
Distance vector routing

- Bellman-Ford equation (dynamic programming)
- let 
  - $d_x(y) := \text{cost of least-cost path from } x \text{ to } y$
- then
  - $d_x(y) = ?$
Distance vector routing

- Bellman-Ford equation (dynamic programming)

- let

- $dx(y) := \text{cost of least-cost path from } x \text{ to } y$

- then

- $dx(y) = \min_v \{c(x,v) + dv(y) \}$, $v: \text{neighbors of } x$
Distance vector routing: example

- What’s the cost of least-cost path for $u \rightarrow z$?
Let’s work it out

- clearly:
  - $dv(z) = \text{?}$, $dx(z) = \text{?}$, $dw(z) = \text{?}$
Let’s work it out

- clearly:
  - $dv(z) = 5$, $dx(z) = 3$, $dw(z) = 3$

- According to B-F equation:
  - $du(z) = \min \{ \ ? \}$
Let’s work it out

- clearly:
  - \( dv(z) = 5, \ dx(z) = 3, \ dw(z) = 3 \)

- According to B-F equation:
  - \( du(z) = \min\{c(v, x) + dv(z), c(u, x) + dx(z), c(u, w) + dw(z)\} \)
Let’s work it out

• clearly:
  • $dv(z) = 5$, $dx(z) = 3$, $dw(z) = 3$

• According to B-F equation:
  • $du(z) = \min \{c(u, v) + dv(z), c(u, x) + dx(z), c(u, w) + dw(z)\}$
  • $= \min \{2 + 5, 1 + 3, 5 + 3\} = 4$
Distance vector routing: key idea

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation.
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?
- Can you propose a solution?
- basic idea?

A should not propagate its distance to B!
Distance vector routing: split horizon

• Previous solution idea:
  • split horizon
    • if A reaches C through B, A should not tell B that B can reach C
    • Then B will not attempt to go through A to reach C
  • Are we good?
Distance vector routing: split horizon

• Previous solution idea:
  
  • split horizon
  
  • if A reaches C through B, A should not tell B that B can reach C
  
  • Then B will not attempt to go through A to reach C
  
  • Are we good?
Distance vector routing: poison reverse

- Split horizon + poison reverse
  - if A reaches D through C:
    - A tells C that A’s distance to D is infinite
    - Then C will not attempt to go through A to reach D
  - In practice, infinite == 16 hops
Distance vector routing: poison 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)
## Link State v.s. Distance Vector

<table>
<thead>
<tr>
<th></th>
<th>Link state</th>
<th>Distance vector</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>message complexity</strong></td>
<td>with n nodes, E links, O(nE) msgs sent</td>
<td>exchange between neighbors only (convergence time varies)</td>
</tr>
<tr>
<td><strong>convergence speed</strong></td>
<td>O(n^2) algorithm requires O(nE) msgs</td>
<td>convergence time varies (may be routing loops)</td>
</tr>
<tr>
<td><strong>robustness</strong></td>
<td>node can advertise incorrect link cost; each node computes only its own table</td>
<td>DV node can advertise incorrect path cost; error propagate thru network</td>
</tr>
<tr>
<td><strong>implementation</strong></td>
<td>OSPF</td>
<td>RIP</td>
</tr>
</tbody>
</table>


Summary

• Link-state routing (Dijkstra) algorithm:
  • each node computes the shortest paths to all the other nodes based on the complete topology map

• Distance Vector (Bellman-Ford) routing algorithm:
  • each node computes the shortest paths to all the other nodes based on its neighbors distance to all destinations