Network Layer III

Dmitri Loguinov
Texas A&M University

November 16, 2021
Homework #4 Grading

• Default mode: final grading will use 3 homeworks
  – Homework contribution = (hw1+hw2+hw3) / 3

• Extra-credit option A: use hw4 in place of any previous homework
  – Swapping out hw1, we get (hw4+hw2+hw3) / 3

• Extra-credit option B: add 20% of hw4 to other homeworks
  – (hw1 + hw2 + hw3 + 0.2*hw4) / 3

• Example: hw1 = 21, hw2 = 80, hw3 = 70, hw4 = 60
  – Default = 57, option A = 70, option B = 61

• Example: hw1 = 90, hw2 = 102, hw3 = 90, hw4 = 90
  – Default = option A = 94, option B = 100
NAT: Network Address Translation

All datagrams *leaving* local network have the same single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)
NAT: Network Address Translation

• Local network uses just one IP address as far as the outside world is concerned
  - No need to be allocated a range of addresses from ISP – just one IP address is used for all devices
  - Can change addresses of devices in local network without notifying outside world
  - Can change ISP without changing addresses of devices in local network
  - Devices inside local net not explicitly addressable or visible to outside world (a security plus)

• To see your NAT IP and current NAT port, visit http://ipchicken.com/
NAT: Network Address Translation

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

<table>
<thead>
<tr>
<th>WAN side addr</th>
<th>LAN side addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.76.29.7, 5001</td>
<td>10.0.0.1, 3345</td>
</tr>
<tr>
<td>……</td>
<td>……</td>
</tr>
</tbody>
</table>

3: Reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

1: host 10.0.0.1 sends datagram to 128.194.135.72, 80
NAT: Network Address Translation

• 16-bit port-number field
  – Up to 64K simultaneous connections with a single LAN-side address

• NAT is controversial:
  – Routers should only process up to layer 3
  – Violates the end-to-end argument

• Makes inbound connections difficult
  – Inbound connections needed in P2P and other applications
  – May be overcome by UPnP or manually configuring NAT to route incoming connections to a particular host

• Some believe that address shortage should instead be solved by IPv6
Chapter 4: Roadmap

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
4.5 Routing algorithms
4.6 Routing in the Internet
4.7 Broadcast and multicast routing
ICMP: Internet Control Message Protocol

- Communicates network-level debug information
  - Error reporting: unreachable host, network, port, protocol
  - Echo request/reply (ping)
- Network-layer above IP
  - ICMP msgs carried in IP datagrams (“layer 3.5”)
- ICMP error message
  - Payload contains first 28 bytes of IP pkt causing error

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>router advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
<th>ID</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>32 bits</td>
<td></td>
</tr>
</tbody>
</table>
Traceroute and ICMP

- Source sends series of **UDP** segments to dest
  - First with TTL = 1
  - Second with TTL = 2
  - *Unlikely* port number

- When the \(n\)-th datagram arrives to the \(n\)-th router:
  - Router discards datagram
  - Sends to source a TTL Expired (type 11, code 0)
  - Message includes IP hdr from router & first 28 bytes of original packet

- When ICMP message arrives, source calculates RTT
  - Traceroute does this 3 times per hop

**Stopping criterion**

- UDP segment eventually arrives at destination host
  - Destination returns ICMP “port unreachable” packet (type 3, code 3)
  - When source gets this ICMP, it stops
Chapter 4: Roadmap

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
   - Datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 Routing algorithms
4.6 Routing in the Internet
4.7 Broadcast and multicast routing
IPv6

- Initial motivation: 32-bit address space has been completely allocated
- Additional motivation:
  - Simpler header format helps speed up forwarding
  - Header changes to facilitate QoS and extensions

IPv6 datagram format:
- Fixed-length 40 byte header
- No fragmentation allowed

- Priority of packet (QoS)
- Flow ID (not well defined)
- Upper-layer protocol (e.g., TCP, ICMP) or IPv6 extension header

16-byte IP, e.g., FEBC:A574:382B:23C1:AA49:4592:4EFE:9982
**IPv6 Notes**

- **Checksum:** removed entirely to reduce processing time at each hop
  - Recall that IPv4 checksums the header only (TCP/UDP checksum the entire packet)
- **Options:** allowed, but outside of header, indicated by “Next Header” field
- All routers cannot be upgraded simultaneously
  - How will the network operate with mixed IPv4 / IPv6 routers?
- **Tunneling:** IPv6 carried as payload in IPv4 datagram among IPv4 routers
Q: how does E know the packet has encapsulated IPv6 data?
A: protocol field (often 41)
Chapter 4: Roadmap

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
4.5 Routing algorithms
   - Link state
   - Distance Vector
   - Hierarchical routing
4.6 Routing in the Internet
4.7 Broadcast and multicast routing
Interplay Between Routing and Forwarding

Routing algorithm

Local forwarding table

<table>
<thead>
<tr>
<th>Header Value</th>
<th>Output Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

Value in arriving packet’s header
Graph Abstraction

Graph: $G = (V, E)$

$V = \text{set of routers} = \{u, v, w, x, y, z\}$

$E = \text{set of links} = \{ (u,v), (u,x), (u,w), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$
Graph Abstraction: Costs

- $c(x,y) =$ cost of link $(x,y)$
  - E.g., $c(w,z) = 5$
- Cost options:
  - Could always be 1
  - Could be inversely related to bandwidth or be proportional to congestion
  - Physical distance

Cost of path $(x_1, x_2, x_3, ..., x_p) = c(x_1, x_2) + c(x_2, x_3) + ... + c(x_{p-1}, x_p)$

Question: What’s the least-cost path between $u$ and $z$?

Routing algorithms find least-cost paths
Routing Algorithm Classification

Global or local information?
- **Global**: Routers have complete topology, link cost info
  - “Link state” algorithms
- **Local (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**: Useful when routes change slowly over time
  - Manual or DHCP-based route creation
- **Dynamic**: Routes change more quickly
  - Periodic update in response to link cost changes
Chapter 4: Roadmap

4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What’s inside a router
4.4 IP: Internet Protocol
4.5 Routing algorithms
   - Link state
   - Distance Vector
   - Hierarchical routing
4.6 Routing in the Internet
4.7 Broadcast and multicast routing
Simple Link-State Routing Algorithm

Dijkstra’s algorithm

• Entire network topology and link costs known
  ─ Accomplished via “link state broadcast”
  ─ Eventually, all nodes have same info

• Computes least cost paths from one node (“source”) to all other nodes
  ─ Gives forwarding table for that node

• Iterative: after \( k \) iterations, know least-cost path to \( k \) closest destinations

Notation:

• \( c(x,y) \): link cost from \( x \) to \( y \)
  ─ Cost is \( \infty \) if not direct neighbors

• \( D(v) \): current estimate of the cost from source to destination \( v \)

• \( p(v) \): predecessor of \( v \) along the least-cost path back to source

• \( F \): set of closest nodes whose least-cost path has been finalized (i.e., known for a fact)
**Dijkstra’s Algorithm**

**Initialization:**

\[
F = \{u\}, \quad D(u) = 0 \\
\text{for all nodes } v \neq u \\
\quad \text{if } v \text{ is adjacent to } u \\
\quad \quad D(v) = c(u,v) \\
\quad \text{else} \\
\quad \quad D(v) = \infty \\
\]

\[
do \\
\quad \text{find node } i \text{ not in } F \text{ such that } D(i) \text{ is minimum} \\
\quad \text{add } i \text{ to } F \\
\quad \text{for all } j \text{ adjacent to } i \text{ and not in } F: \\
\quad \quad D(j) = \min(D(j), D(i) + c(i,j)) \\
\quad \text{/* new cost to } j \text{ is either old cost to } j \text{ or known shortest path cost to } i \text{ plus cost from } i \text{ to } j */ \\
\}\text{ while (not all nodes in } F\)

![Diagram of a graph with nodes and edges labeled with costs.](image-url)
**Dijkstra’s Algorithm: Example**

<table>
<thead>
<tr>
<th>Step</th>
<th>$F$</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>0</td>
<td>$u$</td>
<td>$2, u$</td>
<td>$5, u$</td>
<td>$1, u$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>1</td>
<td>$ux$</td>
<td>$2, u$</td>
<td>$4, x$</td>
<td>$2, x$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>$uxy$</td>
<td>$2, u$</td>
<td>$3, y$</td>
<td>$\infty$</td>
<td>$4, y$</td>
<td>$4, y$</td>
</tr>
<tr>
<td>3</td>
<td>$uxyw$</td>
<td>$3, y$</td>
<td>$\infty$</td>
<td>$4, y$</td>
<td>$4, y$</td>
<td>$4, y$</td>
</tr>
<tr>
<td>4</td>
<td>$uxywz$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$4, y$</td>
<td>$4, y$</td>
<td>$4, y$</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The diagram illustrates a network with nodes $u$, $v$, $w$, $x$, $y$, and $z$. The table shows the steps of Dijkstra’s algorithm, where $F$ represents the set of nodes, and $D(v)$ and $p(v)$ represent the distance and predecessor of node $v$, respectively.
Dijkstra’s Algorithm Discussion

Algorithm complexity: $n$ nodes

- Iteration $k$: need to find min of $(n-k)$ costs, visit $d_i$ neighbors
- Naïve implementation: $O(|E| \cdot |V|)$ complexity
- Heap-based implementation: $O(|E| \cdot \log|V|)$

Oscillations possible, but only for traffic-dependent cost:
- e.g., Link cost = amount of carried traffic

![Diagram](image)