NAT: Network Address Translation

- **Motivation:**
  - Local (home) network uses just one IP address as far as outside world view:
    - Range of addresses not needed from ISP: just one IP address 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, visible by outside world (a security plus)
  - Range of addresses within: 10.0.0.0/24

NAT Router Tasks

**Implementation:** NAT router must:

- For outgoing datagrams: replace (source IP address, port #) of every outgoing datagram with (NAT IP address, new port #)
  - Remote clients/servers will respond using (NAT IP address, new port #) as destination addr
- Remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
- For incoming datagrams: replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table
**NAT: network address translation**

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

<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>
</tbody>
</table>

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

---

**Standard Reserved IP Address Blocks for Private Network Use**

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

---

**NAT Controversies?**

- Port numbers are used by NAT to identify hosts (and the process) within the local network - but ports are for addressing processes only not hosts
- Routers should only process packets up to layer 3 (ports associated with app socket)
- Violates end-to-end argument
  - NAT possibility must be taken into account by application designers, e.g., P2P applications
  - Interfering nodes should not modify IP addresses and port numbers
- Address shortage should instead be solved by IPv6
Summary So Far...

- IP Addressing
  - Network prefixes and Subnets
  - IP datagram format
- DHCP – dynamic addressing
  - Obtain: own IP address
  - Subnet mask, DNS server & first-hop router IP addresses
- NAT – network address translation

On to Routing and Routing Algorithms:
  Link-State and Distance-Vector

Recap: Routing v. Forwarding

Routing Notation

Graph: \( G = (N,E) \)

- \( N = \) set of nodes, here nodes = routers
  = \{ u, v, w, x, y, z \}
- \( E = \) set of edges or links
  = \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}
A Link-State Routing Algorithm

Dijkstra’s algorithm
- Computes the shortest paths in a graph by using weights on edges as a measure of distance.
  - Starts with complete information
  - A path with the least number of edges may not be the path with the least weight / least cost.
- Each node has global information on network topology and edge weights
- A ‘Greedy’ algorithm
  - Makes the locally optimum choice, with objective of finding the global optimum

Dijkstra Notation
- $c(x,y)$: link cost from node $x$ to $y$
  - $= \infty$ 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 $v$
- $N^*$: set of nodes whose least cost path definitively known

A Link-State Routing Algorithm

Dijkstra’s algorithm
- Computes least cost paths from one node (‘source’) to all other nodes
  - Determines the forwarding table for that node
- The network topology and link costs are known to all nodes
  - accomplished via "link state broadcast"
    - all nodes have the same information
- The algorithm is iterative: after $k$ iterations, the least cost paths to $k$ destinations are known

http://www.b2.is.tokushima-u.ac.jp/~ikeda/auuri/dijkstra/DijkstraApp.shtml?demo1
Dijkstra’s Algorithm for node ‘u’

1 **Initialization:**
2 \( N' = \{u\} \)
3 for all nodes \( v \)
4 if \( v \) is neighbor to \( u \)
5 then \( D(v) = c(u,v) \)
6 else \( D(v) = \infty \)

---

**Loop**

9 find some \( w \) not yet 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 /* new cost to \( v \) is either old cost to \( v \) or known
14 shortest path cost to \( w \) plus cost from \( w \) to \( v \) */
15 * until all nodes are in set \( N' \)*

---

Dijkstra’s algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>( N' )</th>
<th>( D(v) )</th>
<th>( p(v) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( u )</td>
<td>7, ( u )</td>
<td>5, ( u )</td>
</tr>
<tr>
<td>1</td>
<td>uw</td>
<td>6, ( w )</td>
<td>5, ( u )</td>
</tr>
<tr>
<td>2</td>
<td>uwxt</td>
<td>8, ( w )</td>
<td>11, ( w )</td>
</tr>
<tr>
<td>3</td>
<td>uwxtv</td>
<td>6, ( v )</td>
<td>10, ( w )</td>
</tr>
<tr>
<td>4</td>
<td>uwxy</td>
<td>4, ( y )</td>
<td>4, ( y )</td>
</tr>
<tr>
<td>5</td>
<td>uwxyz</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
</tbody>
</table>

**Notes:**

- Construct shortest path tree by tracing predecessor nodes
- Construct the forwarding table by recording the next hop to the destination node
- What is the forwarding table??

Resulting forwarding table in \( A \):

<table>
<thead>
<tr>
<th>destination</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>(A, B)</td>
</tr>
<tr>
<td>D</td>
<td>(A, D)</td>
</tr>
<tr>
<td>E</td>
<td>(A, D)</td>
</tr>
<tr>
<td>C</td>
<td>(A, D)</td>
</tr>
<tr>
<td>F</td>
<td>(A, D)</td>
</tr>
</tbody>
</table>
Routing Activity

- Each pair, or table, be a different router
- Fill in table on handout using Dijkstra’s algorithm, for your router letter (IP address)
- Create the forwarding table (back side of handout)
- Send datagrams to a distant destination, forwarding the datagrams to the appropriate “next-hop” using your forwarding table.

Link State Example

Use Dijkstra’s algorithm to compute the least-cost-path table for node x, and the forwarding table for x’s router

Final Step: The Forwarding Table
Recap of Routing So Far

- Review routing vs. forwarding
- Questions?

Semester project first thoughts

- Find a topic of interest to you, related to computer networks, that is a current topic that is still being decided, defined or developed
  - Not: What is ethernet?
  - Instead: We propose ______ for net neutrality and this is why.
  - Or: Here is a great new hardware device we propose to do ______

- Write a research proposal for your topic
  - Introduction
  - Background of the topic and previous research
    - Why it is of interest and to whom
    - Why, how, where, etc, is this topic evolving
  - Your new question and/or proposal related to this topic
    - What is your proposal for: a decision, or development, or change, etc, for your topic, and why?
    - What would you do to complete your research proposal (what are your proposed methods, for example)
    - What form would your results take
  - Conclusions

Summary

Routing:
- Routing objectives
- Routing notation
- Routing classification

Still to come...
- Distance Vector algorithm
- Hierarchical structure