NAT: Network Address Translation

**Motivation:**
- local network uses just one IP address as far as outside world is concerned.
- 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).

**Implementation:**
- NAT router must:
  - outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (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
  - incoming datagrams: replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

**16-bit port-number field:**
- 60,000 simultaneous connections with a single LAN-side address!

**NAT is controversial:**
- routers should only process up to layer 3
- violates end-to-end argument
  - NAT possibility must be taken into account by app designers, eg., P2P applications
- address shortage should instead be solved by IPv6

**ICMP: Internet Control Message Protocol**
- used by hosts & routers to communicate network-level information
- error reporting:
  - unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network layer "above" IP:
  - ICMP messages carried in IP datagrams
- ICMP message type, code plus first 8 bytes of IP datagram causing error

---

**Network Layer**

- **Network Layer 4-1**
- **Network Layer 4-2**
- **Network Layer 4-3**
- **Network Layer 4-4**
- **Network Layer 4-5**
- **Network Layer 4-6**
Traceroute and ICMP

- Source sends series of UDP segments to dest
  - First has TTL = 1
  - Second has TTL = 2, etc.
  - Unlikely port number
- When nth datagram arrives to nth router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0)
  - Message includes name of router & IP address
- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times
- Stepping criterion
- UDP segment eventually arrives at destination host
- Destination returns ICMP “host unreachable” packet (type 3, code 3)
- When source gets this ICMP, stops.

IPv6

- Initial motivation: 32-bit address space soon to be completely allocated.
- Additional motivation:
  - Header format helps speed processing/forwarding
  - Header changes to facilitate QoS
- IPv6 datagram format:
  - Fixed-length 40 byte header
  - No fragmentation allowed

IPv6 Header (Cont)

- Priority: identify priority among datagrams in flow
- Flow Label: identify datagrams in some “flow” (concept of “flow” not well defined)
- Next Header: identify upper layer protocol for data

- Other Changes from IPv4
  - Checksum: removed entirely to reduce processing time at each hop
  - Options: allowed, but outside of header, indicated by “Next Header” field
  - ICMPv6: new version of ICMP
    - Additional message types, e.g. “Packet Too Big”
    - Multicast group management functions

Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneously
  - No “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers
Tunneling

Logical view:

Physical view:

Graph abstraction

Graph: $G = (N, E)$

$N = \{u, v, w, x, y, z\}$

$E = \{(u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)\}$

Graph abstraction: costs

Routing Algorithm classification

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

Static or dynamic?
- Static:
  - routes change slowly over time
- Dynamic:
  - routes change more quickly
  - periodic update
  - in response to link cost changes

A Link-State Routing Algorithm

Dijkstra's algorithm
- net topology, link costs known to all nodes
- accomplished via “link state broadcast”
- 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 dest.'s

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. $v$
- $P(v)$: next hop node along path from source to $v$
- $N^*$: set of nodes whose least cost path definitively known

Interplay between routing, forwarding

Routing algorithm: algorithm that finds least-cost path

Routing algorithm: example

value in arriving packet's header

Remark: Graph abstraction is useful in other network contexts
Example: P2P, where $N$ is set of peers and $E$ is set of TCP connections
Dijsktra’s Algorithm
(compute routes from u to all nodes)

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 /* 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 in $N'$

Dijkstra’s algorithm: example

Step $N' \{D(v) via p(v) \}$
0 $u$ $2,v$ $5,w$ $1,x$ $\infty$
1 $u,v$ $2,v$ $4,x$ $1,x$ $2,x$ $\infty$
2 $u,v,y$ $2,v$ $4,x$ $1,x$ $2,x$ $\infty$
3 $u,v,y,w$ $2,v$ $4,x$ $1,x$ $2,x$ $4,x$
Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:

```
   v
  / \
 /   \
|     |
|   w |
|     |
|     |
\   / \
  y   \
  /     \
  x     \
  /       \
  z       \
```

Resulting forwarding table in u:

<table>
<thead>
<tr>
<th>destination</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>(u,v)</td>
</tr>
<tr>
<td>x</td>
<td>(u,x)</td>
</tr>
<tr>
<td>y</td>
<td>(u,x)</td>
</tr>
<tr>
<td>w</td>
<td>(u,x)</td>
</tr>
<tr>
<td>z</td>
<td>(u,x)</td>
</tr>
</tbody>
</table>

Dijkstra's algorithm, discussion

Algorithm complexity: 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: \(O(n \log n)\)

Oscillations possible:
- e.g., link cost = amount of carried traffic

Initially: recompute routing

... recompute...

... recompute...

... recompute...