Overview on Network Routing

Outline:
- Network layer service model
- Routing principles
  - Link state routing

Network Service Model

Q: What service model is needed in transporting packets from sender to receiver?
- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender?

“Analogy”

Q: What service model is needed in air travel (transporting people from location to location)?
- on-time arrival?
- no lost baggage?
- entertainment (free movie)?
- not-too-lousy food?

Virtual Circuits

“source-to-dest path behaves much like telephone circuit”
- call setup for each call before data can flow, explicit teardown afterwards
- each packet carries VC identifier (not destination host ID)
- every router on source-dest path maintains “state” for each passing connection
- link, router resources (bandwidth, buffers) may be allocated to VC
  — to get circuit-like performance

Packet Switching: the Internet model

- no call setup at network layer
- routers: no state about end-to-end connections
  — no network-level concept of “connection”
- packets forwarded using destination host address
  — packets between same source-dest pair may take different paths
Network Layer Service Models

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>best effort</td>
<td>none</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no (inferred via loss)</td>
</tr>
<tr>
<td>Virtual circuit</td>
<td>circuit-like</td>
<td>constant</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>congestion</td>
</tr>
</tbody>
</table>

- Advantage: more efficient ⇒ transmitting more data with the same resources
- Advantage of providing bad service in air travel ⇒ more profit for the airlines
- Internet model being extended for quality of service

Datagram or VC Networks: Why?

Internet
- data exchange among computers
  - "elastic" service, no strict timing requirement
  - "smart" end systems (computers)
    - can adapt, perform control, error recovery
    - simple inside network, complexity at "edge"
  - many link types
    - different characteristics
    - uniform service difficult
- Internet model being extended for quality of service

VC Networks
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - complexity inside network

Outline
- Network layer service model
- Routing principles
  - Link state routing

Routing
- "good" path: typically means minimum cost path
- Graph abstraction for routing algorithms:
  - graph nodes are hosts or routers
  - graph edges are physical links
    - link cost: delay, $ cost, or congestion level

Routing Algorithm Classification
- Global information:
  - all routers have complete topology, link cost info
  - "link state" algorithms
- Decentralized:
  - router knows physically-connected neighbors, link costs to neighbors
  - exchange of info with neighbors, may take many rounds to converge
  - "distance vector" algorithms

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
  - iteratively computes least cost paths from one node ("source") to all other nodes
Dijkstra's Algorithm: Details

1. Initialization:
   - $c(i,j)$: link cost from node $i$ to $j$. Cost infinity 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 dest. $v$.
   - $N$: set of nodes whose least cost path are currently known.

2. $N = \{A\}$
3. for all nodes $v$
   4. if $v$ adjacent to $A$
      5. then $D(v) = c(A,v)$
      6. else $D(v) = \infty$

7. Loop
   8. find $w$ not in $N$ such that $D(w)$ is a minimum
   9. add $w$ to $N$
   10. update $D(v)$ for all $v$ adjacent to $w$ and not in $N$:
       $$D(v) = \min(D(v), D(w) + c(w,v))$$
   11. /* new cost to $v$ is either old cost to $v$ or known shortest path cost to $w$ plus cost from $w$ to $v$ */
   12. until all nodes in $N$

Dijkstra's Algorithm: An Example

<table>
<thead>
<tr>
<th>Step</th>
<th>N</th>
<th>D(B), p(B)</th>
<th>D(C), p(C)</th>
<th>D(D), p(D)</th>
<th>D(E), p(E)</th>
<th>D(F), p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2, A</td>
<td>5, A</td>
<td>1, A</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>1</td>
<td>AO</td>
<td>2, A</td>
<td>4, A</td>
<td>2, D</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2, A</td>
<td>3, E</td>
<td>4, E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3, E</td>
<td>4, E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td>4, E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>N</th>
<th>D(B), p(B)</th>
<th>D(C), p(C)</th>
<th>D(D), p(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2, A</td>
<td>5, A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>AO</td>
<td>2, A</td>
<td>4, A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2, A</td>
<td>3, E</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3, E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td>4, E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dijkstra's Algorithm: Discussion

Algorithm complexity: $n$ nodes
- each iteration: need to check all nodes, $w$, not in $N$
- $n^2(n+1)/2$ comparisons: $O(n^2)$
- using heap to find minimum distance node: $O(n \log n)$

Oscillations possible:
- e.g., link cost = amount of carried traffic
- solution: not run at the same time

Disclaimer

- Parts of the lecture slides contain original work of James Kurose, Larry Peterson, and Keith Ross. The slides are intended for the sole purpose of instruction of computer networks at the University of Rochester. All copyrighted materials belong to their original owner(s).