Routing

Telematics group
University of Göttingen, Germany
Routing

• Routing principles
  – Link state routing
  – Distance vector routing
• Hierarchical routing
• Homework
• Self-learning: Internet routing protocols

Credits:
  ➢ James Kurose & Keith Ross: Computer Networking(2nd Ed.), Addison-Wesley, 2002
Routing protocol

**Goal:** determine “good” path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:
- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level

• “good” path:
  - typically means minimum cost path
  - other def's possible
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 routing table for that node
- iterative: after k iterations, know least cost path to k dest.’s

Notation:

- \( c(i,j) \): link cost from node i to j. cost infinite 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, that is next v
- \( N \): set of nodes whose least cost path definitively known
Dijsktra’s Algorithm

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

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

<table>
<thead>
<tr>
<th>Step</th>
<th>start 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>AD</td>
<td>2,A</td>
<td>4,D</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>4,E</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph diagram](image_url)
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
Distance Vector Routing Algorithm

iterative:
• continues until no nodes exchange info.
• self-terminating: no “signal” to stop

asynchronous:
• nodes need not exchange info/iterate in lock step!

distributed:
• each node communicates only with directly-attached neighbors

Distance Table data structure
• each node has its own
• row for each possible destination
• column for each directly-attached neighbor to node
• example: in node X, for dest. Y via neighbor Z:

\[
D_X^{(Y,Z)} = \text{distance from } X \text{ to } Y, \text{ via } Z \text{ as next hop} = c(X,Z) + \min_w \{D_Z^{(Y,w)}\}
\]
Distance Table: example

**Cost to destination via**

<table>
<thead>
<tr>
<th>Destination</th>
<th>A</th>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>

\[
D^E_{(C,D)} = c(E,D) + \min_w \{ D^D_{(C,w)} \}
= 2 + 2 = 4
\]

\[
D^E_{(A,D)} = c(E,D) + \min_w \{ D^D_{(A,w)} \}
= 2 + 3 = 5 \text{ loop!}
\]

\[
D^E_{(A,B)} = c(E,B) + \min_w \{ D^B_{(A,w)} \}
= 8 + 6 = 14 \text{ loop!}
\]
Distance table gives routing table

<table>
<thead>
<tr>
<th>Destination</th>
<th>A</th>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>

Outgoing link to use, cost

<table>
<thead>
<tr>
<th>Destination</th>
<th>Outgoing Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A,1</td>
</tr>
<tr>
<td>B</td>
<td>D,5</td>
</tr>
<tr>
<td>C</td>
<td>D,4</td>
</tr>
<tr>
<td>D</td>
<td>D,4</td>
</tr>
</tbody>
</table>

Distance table → Routing table
Distance Vector Routing: overview

Iterative, asynchronous:
  each local iteration caused by:
  • local link cost change
  • message from neighbor: its least cost path change from neighbor

Distributed:
  • each node notifies neighbors only when its least cost path to any destination changes
    – neighbors then notify their neighbors if necessary

Each node:

  wait for (change in local link cost of msg from neighbor)

  recompute distance table

  if least cost path to any dest has changed, notify neighbors
Distance Vector Algorithm:

At all nodes, $X$:

1. Initialization:
2. for all adjacent nodes $v$:
3. $D^{X}(*,v) = \infty$ /* the * operator means "for all rows" */
4. $D^{X}(v,v) = c(X,v)$
5. for all destinations, $y$
6. send $\min_{w} D^{X}(y,w)$ to each neighbor /* $w$ over all $X$'s neighbors */
Distance Vector Algorithm (cont.):

8 loop

9 wait (until I see a link cost change to neighbor V
10 or until I receive update from neighbor V)

11

12 if (c(X,V) changes by d)
13 /* change cost to all dest's via neighbor v by d */
14 /* note: d could be positive or negative */
15 for all destinations y: \( D^X(y,V) = D^X(y,V) + d \)

16

17 else if (update received from V wrt destination Y)
18 /* shortest path from V to some Y has changed */
19 /* V has sent a new value for its \( \min_w DV(Y,w) \) */
20 /* call this received new value is "newval" */
21 for the single destination y: \( D^X(Y,V) = c(X,V) + \text{newval} \)

22

23 if we have a new \( \min_w D^X(Y,w) \) for any destination Y
24 send new value of \( \min_w D^X(Y,w) \) to all neighbors

25

26 forever
Distance Vector Algorithm: example
Distance Vector Algorithm: example

\[ D^X(Y,Z) = c(X,Z) + \min_w \{ D^Z(Y,w) \} \]
\[ = 7 + 1 = 8 \]

\[ D^X(Z,Y) = c(X,Y) + \min_w \{ D^Y(Z,w) \} \]
\[ = 2 + 1 = 3 \]
Distance Vector: link cost changes

Link cost changes:
- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23, 24)

“good news travels fast”
Distance Vector: link cost changes

Link cost changes:
- good news travels fast
- bad news travels slow - “count to infinity” problem!

![Diagram showing distance vector algorithm]

WS 2003/04, fu@informatik.cs.uni-goettingen.de
Distance Vector: poisoned 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)
- will this completely solve count to infinity problem?

![Diagram showing network with nodes X, Y, Z and distances and time changes.](image)
Comparison of LS and DV algorithms

Message complexity

- **LS**: with \( n \) nodes, \( E \) links, \( O(nE) \) msgs sent each
- **DV**: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- **LS**: \( O(n^2) \) algorithm requires \( O(nE) \) msgs
  - may have oscillations
- **DV**: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

- **LS**:
  - node can advertise incorrect *link* cost
  - each node computes only its own table

- **DV**:
  - DV node can advertise incorrect *path* cost
  - each node’s table used by others
    - error propagate thru network
Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network “flat”

... not true in practice

scale: with 200 million destinations:
- can’t store all dest’s in routing tables!
- routing table exchange would swamp links!

administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network
Hierarchical Routing

- aggregate routers into regions, “autonomous systems” (AS)
- routers in same AS run same routing protocol
  - “intra-AS” routing protocol
  - routers in different AS can run different intra-AS routing protocol

**gateway routers**

- special routers in AS
- run intra-AS routing protocol with all other routers in AS
- *also* responsible for routing to destinations outside AS
  - run *inter-AS routing* protocol with other gateway routers
Intra-AS and Inter-AS routing

Gateways:
- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS

inter-AS, intra-AS routing in gateway A.c

network layer
link layer
physical layer

WS 2003/04, fu@informatik.cs.uni-goettingen.de
Intra-AS and Inter-AS routing

- Self-learning: typical Internet routing protocols:
  - inter-AS: BGP
  - intra-AS: RIP, OSPF
Internet AS Hierarchy

Intra-AS border (exterior gateway) routers

Inter-AS interior (gateway) routers
Intra-AS Routing

• Also known as Interior Gateway Protocols (IGP)
• Most common Intra-AS routing protocols:
  – RIP: Routing Information Protocol
  – OSPF: Open Shortest Path First
  – IGRP: Interior Gateway Routing Protocol (Cisco proprietary)
The Internet Network layer

Host, router network layer functions:

- **Routing protocols**
  - path selection
  - RIP, OSPF, BGP

- **IP protocol**
  - addressing conventions
  - datagram format
  - packet handling conventions

- **ICMP protocol**
  - error reporting
  - router “signaling”

Transport layer: TCP, UDP

Network layer

Link layer

physical layer
IP Addressing: introduction

- **IP address**: 32-bit identifier for host, router *interface*
- **interface**: connection between host/router and physical link
  - router’s typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with each interface

```
223.1.1.1 = 11011111 00000001 00000001 00000001
```

WS 2003/04, fu@informatik.cs.uni-goettingen.de
Getting a datagram from source to dest.

IP datagram:

- datagram remains **unchanged**, as it travels source to destination
- addr fields of interest here

### forwarding table in A

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>Nhops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td>223.1.1.4</td>
<td>1</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

WS 2003/04, fu@informatik.cs.uni-goettingen.de
Getting a datagram from source to dest.

Starting at A, send IP datagram addressed to B:

- look up net. address of B in forwarding table
- find B is on same net. as A
- link layer will send datagram directly to B inside link-layer frame
  - B and A are directly connected

<table>
<thead>
<tr>
<th>misc fields</th>
<th>223.1.1.1</th>
<th>223.1.1.3</th>
<th>data</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>Nhops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

WS 2003/04, fu@informatik.cs.uni-goettingen.de
Getting a datagram from source to dest.

Arriving at 223.1.4, destined for 223.1.2.2

- look up network address of E in router’s forwarding table
- E on same network as router’s interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)
Homework

• In „J. Kurose/K. Ross, Computer Networking“, Chapter 4 Problems (pp.409-410) No. 3-5, 7
• Compare and contrast LS and DV algorithms
• What is the difference between routing and addressing?
• Why are different inter-AS and intra-AS protocols used in the Internet?
RIP (Routing Information Protocol)

• Distance vector algorithm
• Included in BSD-UNIX Distribution in 1982
• Distance metric: # of hops (max = 15 hops)
  – Can you guess why?

• Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called advertisement)
• Each advertisement: list of up to 25 destination nets within AS
### RIP: Example

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Num. of hops to dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

Routing table in D
RIP: Example

Advertisement from A to D

Routing table in D

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>z</td>
<td>C</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Num. of hops to dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
</tbody>
</table>

WS 2003/04, fu@informatik.cs.uni-goettingen.de
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

– routes via neighbor invalidated
– new advertisements sent to neighbors
– neighbors in turn send out new advertisements (if tables changed)
– link failure info quickly propagates to entire net
– poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)
RIP Table processing

- RIP routing tables managed by \textit{application-level} process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated
### RIP Table example (continued)

**Router:** `giroflee.eurocom.fr`

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Flags</th>
<th>Ref</th>
<th>Use</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>UH</td>
<td>0</td>
<td>26492</td>
<td>lo0</td>
</tr>
<tr>
<td>192.168.2.</td>
<td>192.168.2.5</td>
<td>U</td>
<td>2</td>
<td>13</td>
<td>fa0</td>
</tr>
<tr>
<td>193.55.114.</td>
<td>193.55.114.6</td>
<td>U</td>
<td>3</td>
<td>58503</td>
<td>le0</td>
</tr>
<tr>
<td>192.168.3.</td>
<td>192.168.3.5</td>
<td>U</td>
<td>2</td>
<td>25</td>
<td>qaa0</td>
</tr>
<tr>
<td>224.0.0.0</td>
<td>193.55.114.6</td>
<td>U</td>
<td>3</td>
<td>0</td>
<td>le0</td>
</tr>
<tr>
<td>default</td>
<td>193.55.114.129</td>
<td>UG</td>
<td>0</td>
<td>143454</td>
<td></td>
</tr>
</tbody>
</table>

- Three attached class C networks (LANs)
- Router only knows routes to attached LANs
- Default router used to “go up”
- Route multicast address: 224.0.0.0
- Loopback interface (for debugging)
OSPF (Open Shortest Path First)

- “open”: publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm

- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)
  - Carried in OSPF messages directly over IP (rather than TCP or UDP)
OSPF “advanced” features (not in RIP)

- **Security**: all OSPF messages authenticated (to prevent malicious intrusion)
- **Multiple same-cost paths** allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set “low” for best effort; high for real time)
- Integrated uni- and **multicast** support:
  - Multicast OSPF (MOSPF) uses same topology database as OSPF
- **Hierarchical** OSPF in large domains.
Hierarchical OSPF
Hierarchical OSPF

- **Two-level hierarchy**: local area, backbone.
  - Link-state advertisements only in area
  - Each node has detailed area topology; only knows direction (shortest path) to nets in other areas.
- **Area border routers**: “summarize” distances to nets in own area, advertise to other Area Border routers.
- **Backbone routers**: run OSPF routing limited to backbone.
- **Boundary routers**: connect to other AS’s.
Inter-AS routing in the Internet: BGP

AS1 (RIP intra-AS routing)

AS2 (OSPF intra-AS routing)

AS3 (OSPF intra-AS routing)

R1

R2

R3

R4

R5
Internet inter-AS routing: BGP

• BGP (Border Gateway Protocol): the de facto standard

• Path Vector protocol:
  – similar to Distance Vector protocol
  – each Border Gateway broadcast to neighbors (peers) *entire path* (i.e., sequence of AS’s) to destination
  – BGP routes to networks (ASs), not individual hosts
  – E.g., Gateway X may send its path to dest. Z:

  \[
  \text{Path (X,Z)} = X,Y_1,Y_2,Y_3,\ldots,Z
  \]
Internet inter-AS routing: BGP

*Suppose:* gateway X send its path to peer gateway W

- W may or may not select path offered by X
  - cost, policy (don’t route via competitors AS), loop prevention reasons.
- If W selects path advertised by X, then:
  \[
  \text{Path (W,Z)} = w, \text{Path (X,Z)}
  \]
- Note: X can control incoming traffic by controlling its route advertisements to peers:
  - e.g., don’t want to route traffic to Z -> don’t advertise any routes to Z
BGP: controlling who routes to you

- A, B, C are provider networks
- X, W, Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

Figure 4.5-BGPnew: a simple BGP scenario
BGP: controlling who routes to you

- A advertises to B the path AW
- B advertises to W the path BAW
- Should B advertise to C the path BAW?
  - No way! B gets no “revenue” for routing CBAW since neither W nor C are B’s customers
  - B wants to force C to route to W via A
  - B wants to route only to/from its customers!
BGP operation

Q: What does a BGP router do?

• Receiving and filtering route advertisements from directly attached neighbor(s).

• Route selection.
  – To route to destination X, which path (of several advertised) will be taken?

• Sending route advertisements to neighbors.
BGP messages

- BGP messages exchanged using TCP.
- BGP messages:
  - **OPEN**: opens TCP connection to peer and authenticates sender
  - **UPDATE**: advertises new path (or withdraws old)
  - **KEEPALIVE**: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION**: reports errors in previous msg; also used to close connection
Why different Intra- and Inter-AS routing?

Policy:
• Inter-AS: admin wants control over how its traffic routed, who routes through its net.
• Intra-AS: single admin, so no policy decisions needed

Scale:
• hierarchical routing saves table size, reduced update traffic

Performance:
• Intra-AS: can focus on performance
• Inter-AS: policy may dominate over performance