How does routing work?
Packet-switched Routing

Choosing an optimal path

• According to a cost metric
• Decentralised forwarding
  – each router has full/necessary information
Routing in Packet Switching Networks

• Select route across network between end nodes
• Characteristics required:
  – Correctness
  – Simplicity
  – Robustness vs Stability
  – Fairness vs Optimality
  – Efficiency
Router

- Internetworking device
  - Passes data packets between networks
  - Checks *Network Layer* addresses
  - Uses Routing/forwarding tables

**Two functions:**

1. Routing
2. Forwarding
Router Architecture Overview
Input Port

Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet

Decentralized switching:
• Given destination, lookup output port using routing table
• Goal: complete input port processing at ‘line speed’
Input Port Queuing

• Fabric slower that sum of input ports → queuing
• Delay and loss due to input buffer overflow
• Head-of-the-Line (HOL) blocking: Datagram at front of queue prevents others in queue from proceeding
Priority Scheduling:

- Scheduling discipline may choose among queued datagrams for transmission
Output Port Queuing

- Datagrams’ arrival rate through the switch exceeds the transmission rate of the output line → buffering
- Delay and loss due to output port buffer overflow
Switching Fabrics

ETSF05/ETSF10 - Internet Protocols
Routing Tables and Forwarding Table

- OSPF Domain
- RIP Domain

Forwarding Table Manager

- RIP Process
  - RIP Routing tables
- OSPF Process
  - OSPF Routing tables
- BGP Process
  - BGP Routing tables

Static routing table

OS kernel
Router cache

• Save next hop for packet type (e.g. addr and TOS)
  – Keep packets within a session on the same path
  – Prohibits reordering
  – decreases delay variations

• Works in both directions
  – Reply take the same path as request

• Drawback: for long sessions (e.g. video) session continuity might be broken if link fails (e.g. mobility)

• Typical for user networks
Performance Criteria

• Used for selection of route
• Simplest is to choose “minimum hop”
• Can be generalized as “least cost” routing
• Because “least cost” is more flexible it is more common than “minimum hop”
Flooding

• In Flooding an incoming packet is retransmitted on all outgoing links. A hop counter is used to prevent loops.

• What are the alternatives to find the least cost path.
Best Path: Decision Time and Place

Decision time (when?)

• Packet or virtual circuit (session) basis
• Fixed or dynamically changing

Decision place (where?)

• Distributed - made by each node
  • More complex, but more robust
• Centralized – made by a designated node
• Source – made by source station
Network Information Source and Update Timing

• Routing decisions usually based on knowledge of network, traffic load, and link cost
  – Distributed routing
    • Using local knowledge, information from adjacent nodes, information from all nodes on a potential route
  – Central routing
    • Collect information from all nodes

Issue of update timing

• Depends on routing strategy
• Fixed - never updated
• Adaptive - regular updates
Routing Strategies - Fixed Routing

• Use a **single permanent** route for each source to destination pair of nodes
• Determined using a least cost algorithm
• **Route is fixed**
  – Until a change in network topology
  – Based on expected traffic or capacity
• Advantage is **simplicity**
• Disadvantage is **lack of flexibility**
  – Does not react to network failure or congestion
Routing Strategies - Adaptive Routing

- Used by almost all packet switching networks
- Routing decisions change as conditions on the network change due to failure or congestion
- Requires information about network

Disadvantages
- More complex
- Tradeoff between quality and overhead
- Too quick updates may lead to oscillations
- Too slow updates may lead to outdated information
A cost function describes the cost for transmitting a packet over a link.

The link cost can depend on e.g.:
- Data rate
- Load
- Length
- Transmission media
- etc
Graf

A network can be described by a graph, consisting of nodes \( N \) and edges \( E \) with weights \( w(e) \), i.e. costs.

Example (undirected graph)

\[ N = \{A, B, C, D, E\} \]

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( E )</td>
<td>( w(e) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BD</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BE</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ARPANET Routing Strategies
1st Generation

Distance Vector Routing

• 1969
• Distributed adaptive using estimated delay
  – Queue length used as estimate of delay
• Version of Bellman-Ford algorithm
• Node exchanges delay vector with neighbors
• Update routing table based on incoming information
• Doesn't consider line speed, just queue length and
  responds slowly to congestion
Least cost alg 1
Bellman-Ford

• Find the shortest path from one source node $s$ to the others.
• Let $d(n)$ be the cost from $s$ to $n$

Init:


d(s) = 0 \\
d(n) = \infty, \ n \neq s

for $i = 1$ to $|N| - 1$

for each $n \in N$

\[
d(n) = \min_{u \in N} \{d(u) + w(u, n)\} \quad // \text{Find the shortest path from} \quad \ \ \ // \text{node } u \text{ to node } n \text{ in one step}
\]

• Addition: Keep track of the path!!
Example Bellman-Ford

<table>
<thead>
<tr>
<th>Nod</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>i=1</td>
<td>0</td>
<td>3</td>
<td>∞</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>i=2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>i=3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>i=4</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Net graph as a tree

Distant vector for A when the algorithm converged

<table>
<thead>
<tr>
<th>Nod</th>
<th>Dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
</tr>
</tbody>
</table>
Bellman-Fords algoritm grafiskt

a. General case with three intermediate nodes

\[ D_{xy} = \min\{ (c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}) \} \]

b. Updating a path with a new route

\[ D_{xy} = \min\{ D_{xy}, (c_{xz} + D_{zy}) \} \]

Jmf Stallings kap 19.2
Distance Vector Routing

• Best path info **shared** locally
  – Periodically
  – Upon any change

• Routing tables **updated** for
  – New entries
  – Cost changes
Updating a Routing Table

<table>
<thead>
<tr>
<th>Received from C</th>
<th>To</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

A's modified table:

<table>
<thead>
<tr>
<th>A</th>
<th>4</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>6</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
<td>C</td>
</tr>
</tbody>
</table>

Compare

A's new table:

<table>
<thead>
<tr>
<th>To</th>
<th>Cost Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>—</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
</tr>
</tbody>
</table>

A's old table:

<table>
<thead>
<tr>
<th>To</th>
<th>Cost Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
</tr>
</tbody>
</table>

to C (+2)
Updating Algorithm (Bellman-Ford)

if (advertised destination not in table)
{
    add new entry  // rule #1
}
else if (adv. next hop = next hop in table)
{
    update cost  // rule #2
}
else if (adv. cost < cost in table)
{
    replace old entry  // rule #3
}
Completed Routing Tables

A’s table:

<table>
<thead>
<tr>
<th>To</th>
<th>Cost</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>6</td>
<td>C</td>
</tr>
</tbody>
</table>

D’s table:

<table>
<thead>
<tr>
<th>To</th>
<th>Cost</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>8</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>9</td>
<td>A</td>
</tr>
</tbody>
</table>

C’s table:

<table>
<thead>
<tr>
<th>To</th>
<th>Cost</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

B’s table:

<table>
<thead>
<tr>
<th>To</th>
<th>Cost</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

E’s table:

<table>
<thead>
<tr>
<th>To</th>
<th>Cost</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
I can reach A at cost 1

I can reach A at cost 2

I can reach A at cost 3

I don’t know A, so I will not tell

I can reach A at cost 2

I can reach A at cost 3
Problem: Count to Infinity

I can reach A at cost 2
I can reach A at cost 3
I can reach A at cost 4
I can reach A at cost 5
I can reach A at cost 6
I can reach A at cost 7
Solution: Split Horizon

I can reach A at cost 1

I can’t reach A

I can reach A at cost 2 but I won’t tell 1

I can’t reach A

I can reach A at cost 2 but I won’t tell 1

I can’t reach A
Link-State Routing

• 1979
• Distributed adaptive using delay criterion
  – Using timestamps of arrival, departure and ACK times
• Re-computes average delays every 10 seconds
• Any changes are flooded to all other nodes
• Re-computes routing using Dijkstra’s algorithm
• Good under light and medium loads
• Under heavy loads, little correlation between reported delays and those experienced
Least cost alg 2
Dijkstra

• Find the shortest path from one source node $s$ to the others.
• Let $d(n)$ be the cost from $s$ to $n$

**Init:**

\[
\begin{align*}
    d(s) &= 0 \\
    d(n) &= \infty, \ n \neq s \\
    V &= \emptyset \quad \text{// Visited nodes}
\end{align*}
\]

while $V \subset E$

\[
\begin{align*}
    u &= \arg \min_{u \in V} d(u) \quad \text{// Find least weight to unvisited node} \\
    V &= V \cup u \quad \text{// Add $u$ to visited nodes} \\
    \text{for } n \notin V \\
    d(n) &= \min\{d(n), d(u) + w(u,n)\} \quad \text{// Less cost to go to $n$ via $u$?}
\end{align*}
\]

• Addition: Keep track of the path!!
SPF exempel med graf

Potentiell väg
Permanent väg

A -> B
A -> C
B -> D
B -> E
C -> D
C -> E
D -> A
E -> A

Rotnod
Permanent nod
Preliminär nod
Dijkstra tabell

<table>
<thead>
<tr>
<th>Besökt</th>
<th>L(A)</th>
<th>L(B)</th>
<th>L(C)</th>
<th>L(D)</th>
<th>L(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>${A}$</td>
<td></td>
<td>3:A</td>
<td>$\infty$</td>
<td>1:A</td>
<td>$\infty$</td>
</tr>
<tr>
<td>${A,D}$</td>
<td>2:D</td>
<td></td>
<td>$\infty$</td>
<td></td>
<td>4:D</td>
</tr>
<tr>
<td>${A,D,B}$</td>
<td></td>
<td></td>
<td>3:B</td>
<td></td>
<td>3:B</td>
</tr>
<tr>
<td>${A,D,B,C}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3:B</td>
</tr>
<tr>
<td>${A,D,B,C,E}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Link State Routing

• Local topology info **flooded** globally
  – Periodically
  – Upon any change

• Routing tables **updated** for
  – Link state changes
  – Cost changes
Initial Link State Knowledge

A's states of links:
- A to D: 3
- A to C: 5
- A to B: 2

D's states of links:
- D to A: 3

C's states of links:
- C to A: 2
- C to B: 4
- C to D: 4

B's states of links:
- B to C: 4
- B to E: 3

E's states of links:
- E to C: 4

2012-10-30
Tree Generation Algorithm (Dijkstra)

put yourself to tentative list
while tentative list not empty
{
    pick node which can be reached
    with least cumulative cost
    add it to your tree*
    put its neighbours to tentative list**
    with cumulative costs to reach them
}

*(a.k.a. permanent list)
**(if not already there)
Building a Shortest Path Tree

- After flooding
- Take: A

1. Set root to A and move A to tentative list.
2. Move A to permanent list and add B, C, and D to tentative list.
3. Move C to permanent and add E to tentative list.
4. Move D to permanent list.
5. Move B to permanent list.
6. Move E to permanent list (tentative list is empty).

<table>
<thead>
<tr>
<th>Node</th>
<th>Cost</th>
<th>Next Router</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>—</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>—</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>—</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
<td>C</td>
</tr>
</tbody>
</table>
ARPANET Routing Strategies
3rd Generation

• 1987

• Link cost calculation changed
  – Dampen routing oscillations
  – Reduce routing overhead

• Measure average delay over last 10 seconds and transform into link utilization estimate

• Calculate average utilization based on current value and previous average
  \[ U(n + 1) = \frac{1}{2} \rho(n) + \frac{1}{2} U(n) \]

• Use as link cost a function based on the average utilization
Autonomous Systems (AS)

• Exhibits the following characteristics:
  – Is a set of routers and networks managed by a single organization
  – Consists of a group of routers exchanging information via a common routing protocol
  – Except in times of failure, is connected (in a graph-theoretic sense); there is a path between any pair of nodes
Interior Router Protocol (IRP)

- Interior Gateway Protocol (IGP)

- Shared routing protocols passes routing information between routers **within an AS**
- Custom tailored to specific applications and requirements

- Examples:
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
Exterior Router Protocol (ERP)
Exterior Gateway Protocol (EGP)

• Protocol used to pass routing information between routers in different ASs

• Will need to pass less information than an IRP
  – To transmit a datagram from a host in one AS to a host in another AS, a router in the first system need only determine the target AS and devise a route to get into it
  – Once the datagram enters the target AS, the routers within that system can cooperate to deliver the datagram
  – The ERP is not concerned with details of the route

• Examples:
  – Border Gateway Protocol (BGP)
  – Open Shortest Path First (OSPF)
Figure 19.9  Application of Exterior and Interior Routing Protocols
Routing Algorithms and Protocols

Routing protocols
- Intradomain
  - Distance vector: RIP
  - Link state: OSPF
- Interdomain
  - Path vector: BGP
Distance-Vector Routing

• Requires that each node exchange information with its neighboring nodes
  – Two nodes are said to be neighbors if they are both directly connected to the same network

• Used in the first-generation routing algorithm for ARPANET

• Each node maintains a vector of link costs for each directly attached network and distance and next-hop vectors for each destination

• Routing Information Protocol (RIP) uses this approach
RIP (Routing Information Protocol)

• Included in BSD-UNIX Distribution in 1982

• Distance metric:
  – # of hops (max 15) to destination network

• Distance vectors:
  – exchanged among neighbours every 30 second via Response Message (advertisement)

• Implementation:
  – Application layer protocol, uses UDP/IP
RIP update message

• Contains the whole forwarding table

• Action on reception:
  – Add 1 to cost in received message
  – Change next hop to sending router
  – Apply RIP updating algorithm

• Received update msgs identify neighbours!
RIP: Link Failure and Recovery

- If no advertisement heard after 180 seconds
  - Neighbour/link declared dead
  - Routes via neighbour invalidated (infinite distance = 16 hops)
  - New advertisements sent to neighbours (triggering a chain reaction if tables changed)
  - “Poison reverse” used to prevent count to infinity loops
  - “Good news travel fast, bad news travel slow”
Routing Algorithms and Protocols

Routing protocols

Intradomain
- Distance vector
  - RIP
- Link state
  - OSPF

Interdomain
- Path vector
  - BGP
Link-State Routing

• When a router is initialized, it determines the link cost on each of its network interfaces
• The router then advertises this set of link costs to all other routers in the internet topology, not just neighboring routers
• From then on, the router monitors its link costs
• When there is a significant change, the router advertises its link costs to all other routers

• The OSPF protocol is an example
• The second-generation routing algorithm for ARPANET also uses this approach
Open Shortest Path First (OSPF) Protocol

• RFC 2328 (Request For Comments)
• Used as the interior router protocol in TCP/IP networks
• Computes a route that incurs the least cost based on a user-configurable metric
• Is able to balance loads over multiple equal-cost paths
OSPF (Open Shortest Path First)

- Divides domain into areas
  - Limits flooding for efficiency
  - One ”backbone” area connects all

- Distance metric:
  - Cost to destination network
Areas, Router and Link Types

ETSF05/ETSF10 - Internet Protocols
Graph

Network topology expressed as a graph

• Routers
• Networks
  – Transit, passing data through
  – Stub, not transit
• Edges
  – Direct, router to router
  – Indirect, router to network
Transit network

Stub network

Indirect edge

Direct edge

Point to point

Figure 19.12 Directed Graph of Autonomous System of Figure 19.11

ETSF05/ETSF10 - Internet Protocols
Link State Advertisements

• What to advertise?
  – Different entities as nodes
  – Different link types as connections
  – Different types of cost
Router Link Advertisement

Advertising router with four links

To transient network
Point-to-point
To stub network
Virtual
Network Link Advertisement

• Network is a passive entity
  – It cannot advertise itself
Summary Link to Network

• Done by area border routers
  – Goes through the backbone

Area 1
Flooded by the area border router into the area

Area 2
Flooded by the area border router into the area

Autonomous system

R1
Summary link to network

R2
Summary link to network
External Link Advertisement

- Link to a single network outside the domain
Hello message

• Find neighbours
• Keep contact with neighbours: I am still alive!
• Sent out periodically, typically every 10 seconds
• If no hellos received during holdtime (typically 30 seconds), neighbour declared dead.

• Compare RIP update messages
Routing Algorithms and Protocols

• Interior and Exterior Router Protocols

• Distance vector
  – Bellman-Ford
  – Announce whole table to neighbors
  – RIP

• Link State
  – Dijkstra
  – Announce neighbor connections to whole network
  – OSPF