ROUTING
AND FORWARDING

Mario Baldi
www.baldi.info
This set of transparencies, hereinafter referred to as slides, is protected by copyright laws and provisions of International Treaties. The title and copyright regarding the slides (including, but not limited to, each and every image, photography, animation, video, audio, music and text) are property of the authors specified on page 1.

The slides may be reproduced and used freely by research institutes, schools and Universities for non-profit, institutional purposes. In such cases, no authorization is requested.

Any total or partial use or reproduction (including, but not limited to, reproduction on magnetic media, computer networks, and printed reproduction) is forbidden, unless explicitly authorized by the authors by means of written license.

Information included in these slides is deemed as accurate at the date of publication. Such information is supplied for merely educational purposes and may not be used in designing systems, products, networks, etc. In any case, these slides are subject to changes without any previous notice. The authors do not assume any responsibility for the contents of these slides (including, but not limited to, accuracy, completeness, enforceability, updated-ness of information hereinafter provided).

In any case, accordance with information hereinafter included must not be declared.

In any case, this copyright notice must never be removed and must be reported even in partial uses.
Routing

Determine a path through the network for packets
Forwarding

- Advance packets through the network
- Includes a routing decision
Forwarding and Routing

1. Routing (proactive)

2. Forwarding (on-the-fly routing)
Proactive Routing

- Independent of actual traffic
- Determine reachable destinations
- Compute best route
- Commonly referred to as “routing”
On-the-fly Routing

- Realized when handling each packet
- Based on local information
- Routing/forwarding table
- Output of proactive routing or signaling
- A.k.a. route
On-the-fly Routing Algorithms

- Routing by Network Address
- Label Swapping
- Source Routing

Each protocol architecture adopts one or more...
Forwarding Phases

- Routing (on-the-fly)
- Output port selection
- Possibly next-hop selection
- Switching: transfer to output port
- Transmission
A Proactive Routing Algorithm Classification

- Non-adaptive algorithms (static)
- Adaptive algorithms (dynamic)
NON-ADAPTIVE ROUTING
Non-adaptive Algorithms

- Fixed Directory routing
- AKA static routing
- Manual configuration
- (Selective) flooding and derivates
Fixed Directory Routing

Routing Table of B
A,E ↔
C,D →

Routing Table of C
A,B ↔
D →

Routing Table of D
A,B,C,E ↔

Routing Table of E
A,B ↔
C,D →

Routing Table of A
B,D →
E,C ↓

Routing - 13
Pros and Cons

- Administrator has full control
- Error prone
- It does not adapt to topology changes
Backup Route Load Balancing

Routing Table of B
A,E,C,D ↔
C,D,E,A →

Routing Table of A
B,D,C,E →
E,C,B,D ↓

Routing Table of C
A,B ↔
D →
E,A,B ↓

Routing Table of E
A,B ↔
C,D,B →

Routing Table of D
A,B,C,E ↔
Static vs. Dynamic

The only way to the rest of network

Multiple alternative routes

Core Dynamic routing

Small routing tables

Edge Static routing

Static vs. Dynamic

Routing - 16
DYNAMIC ROUTING
Adaptive Algorithms

- Centralized Routing
- Isolated Routing
- Distributed Routing
- Distance Vector
- Link State
Centralized Routing

→ Routing Control Center (RCC)
→ Calculates and distributes routing tables
→ Needs information from all nodes
Centralized Routing

- Optimizes performance
- Simplifies troubleshooting
- Significant network load in proximity of RCC
Centralized Routing

- RCC is single point of failure
- RCC is bottleneck
- Not suitable for highly dynamic networks
Isolated Routing

- Each node decides independently
- No exchange of information
- E.g., Backward Learning
- IEEE 802.1D bridges
Distributed Routing

→ Routers co-operate in exchanging connectivity information

→ Each router decides independently, but coherently

Combines advantages of isolated and centralized routing
DISTANCE VECTOR ROUTING ALGORITHM
1. E at dist 0

2. A at dist 0 and E at 1 (from A1)

3. A at 0 and E at 1

4. C at 0, A at 1 (from C1), F at 1 (from C2), and E at 2 (from C1 and C2, same dist)

2. F at 0 and E at 1 (from F1)

3. F at 0 and E at 1
Distance Vector

→ List of reachable destinations (all!)
→ Distance from announcing router
→ Generated by each router
→ Received from neighbors
Sample Scenario

Routing information stored by A

<table>
<thead>
<tr>
<th>Location (A)</th>
<th>Distance Vector (B)</th>
<th>Distance Vector (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, 0</td>
<td>A, 1</td>
<td>A, 1</td>
</tr>
<tr>
<td>B, 0</td>
<td>B, 1</td>
<td></td>
</tr>
<tr>
<td>C, 1</td>
<td>C, 2</td>
<td></td>
</tr>
<tr>
<td>D, 1</td>
<td></td>
<td>D, 0</td>
</tr>
<tr>
<td>E, 1</td>
<td></td>
<td>E, 1</td>
</tr>
</tbody>
</table>
### Distance Vector Merging and Generation

#### Received from line A1

- **A**: 0
- **B**: 1
- **C**: 0
- **D**: 1
- **E**: 1

#### Received from line A2

- **A**: 0
- **B**: 1
- **C**: 2
- **D**: 0
- **E**: 1

#### Table: ROUT. TABLE (A)

<table>
<thead>
<tr>
<th>Loc (A)</th>
<th>DV (B)</th>
<th>DV (D)</th>
<th>DV (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A, 1</td>
<td>A, 1</td>
<td>A, local, 0</td>
</tr>
<tr>
<td>B</td>
<td>B, 0</td>
<td>B, 1</td>
<td>B, 1</td>
</tr>
<tr>
<td>C</td>
<td>C, 1</td>
<td>C, 1</td>
<td>C, 2</td>
</tr>
<tr>
<td>D</td>
<td>D, 1</td>
<td>D, 0</td>
<td>D, 1</td>
</tr>
<tr>
<td>E</td>
<td>E, 1</td>
<td>E, 1</td>
<td>E, 2</td>
</tr>
</tbody>
</table>
**Topology Change**

**Loc (A)**  | **DV (B)**  | **DV (D)**  | **ROUT. TABLE (A)**  | **DV (A)**
---|---|---|---|---
A, 0 | A, 1 | A, 1 | local, 0 | A, 0
E, 0 | B, 1 | B, 1 | A2, 2 | B, 2
C, 1 | C, 2 | C, 2 | A2, 3 | C, 3
D, 1 | D, 0 | D, 0 | A2, 1 | D, 1
E, 1 | E, 1 | E, 1 | A2, 2 | E, 2
Example: Cold Start

A sends its DV

B and D send their DVs

All send their DVs

Routing - 30
Issues

- Several problems
- Black Hole
- Count to infinity
- Bouncing Effect (loop)

Instability
## Count to Infinity

### IS B:

<table>
<thead>
<tr>
<th>Loc (B)</th>
<th>DV (A)</th>
<th>DV (C)</th>
<th>RT (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B, 0</td>
<td>A, 0</td>
<td>A, 2</td>
<td>A, B2 ,3</td>
</tr>
<tr>
<td></td>
<td>P, 1</td>
<td>B, 1</td>
<td>B, loc,0</td>
</tr>
<tr>
<td></td>
<td>C, 2</td>
<td>C, 0</td>
<td>C, B2 ,1</td>
</tr>
</tbody>
</table>

B sends DV

### IS C:

<table>
<thead>
<tr>
<th>Loc (C)</th>
<th>DV (B)</th>
<th>RT (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, 0</td>
<td>A, 1</td>
<td>A, C1 ,2</td>
</tr>
<tr>
<td></td>
<td>B, 0</td>
<td>B, C1 ,1</td>
</tr>
<tr>
<td></td>
<td>C, 1</td>
<td>C, loc,0</td>
</tr>
</tbody>
</table>

C sends DV

**Count to Infinity!**
### Bouncing Effect

**IS B:**

- **Loc (B):**
  - B, 0
  - B, 1
  - C, 2

- **DV (A):**
  - A, 0
  - B, 1
  - C, 0

- **DV (C):**
  - A, 2
  - B, 1
  - C, 0

- **RT (B):**
  - A, B2, 3
  - B, loc, 0
  - C, B2, 1

- **Loc (C):**
  - C, 0
  - B, 0
  - C, 1

- **DV (B):**
  - A, 1
  - B, 0
  - C, 1

- **RT (C):**
  - A, C1, 2
  - B, C1, 1
  - C, loc, 0

B sends its DV.

**IS C:**

- **Loc (C):**
  - C, 0
  - A, 3
  - B, 0
  - C, 1

- **DV (B):**
  - A, 1
  - B, 0
  - C, 1

- **RT (C):**
  - A, C1, 4
  - B, C1, 1
  - C, loc, 0
Issues

- Partial solutions
- Split Horizon
- Path Hold Down
- Route Poisoning
Split Horizon

“If C reaches destination A through B, it is useless for B trying to reach A through C”
Split Horizon

- Prevents loops between two nodes
- Speeds up convergence
- “Personalized” DVs to neighbors
  - DV of C to B does not contain destinations reached through B
- In actual implementations, route has to expire
Split Horizon with Poisonous Reverse

- More aggressive
- No theoretical advantages
- Practically, no need to wait for route expiration
Split Horizon on Mesh

**IS C:**

<table>
<thead>
<tr>
<th>Loc (C)</th>
<th>DV (B)</th>
<th>DV (D)</th>
<th>RT (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, 0</td>
<td>A, 1</td>
<td>A, 2</td>
<td>A, C1, 2</td>
</tr>
<tr>
<td>B, 0</td>
<td>B, 1</td>
<td>D, 0</td>
<td>B, C1, 1</td>
</tr>
<tr>
<td>D, 1</td>
<td>D, 0</td>
<td>D, loc, 0</td>
<td>C, loc, 0</td>
</tr>
</tbody>
</table>

**IS D:**

<table>
<thead>
<tr>
<th>Loc (D)</th>
<th>DV (B)</th>
<th>DV (C)</th>
<th>RT (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D, 0</td>
<td>A, 1</td>
<td>A, 2</td>
<td>A, D1, 2</td>
</tr>
<tr>
<td>B, 0</td>
<td>B, 1</td>
<td>B, 1</td>
<td>B, D1, 1</td>
</tr>
<tr>
<td>C, 1</td>
<td>C, 0</td>
<td>C, D2, 1</td>
<td>C, D2, 1</td>
</tr>
<tr>
<td>D, loc, 0</td>
<td>D, C2, 1</td>
<td>D, loc, 0</td>
<td></td>
</tr>
</tbody>
</table>

**IS B:**

<table>
<thead>
<tr>
<th>Loc (B)</th>
<th>DV (A)</th>
<th>DV (C)</th>
<th>DV (D)</th>
<th>RT (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B, 0</td>
<td>A, 1</td>
<td>C, 1</td>
<td>D, loc, 0</td>
<td></td>
</tr>
<tr>
<td>D, 1</td>
<td>D, 0</td>
<td>C, B2, 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IS C:**

<table>
<thead>
<tr>
<th>Loc (C)</th>
<th>DV (B)</th>
<th>RT (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, 0</td>
<td>B, 0</td>
<td>A, C2, 3</td>
</tr>
<tr>
<td>D, 1</td>
<td>B, 1</td>
<td>B, C1, 1</td>
</tr>
<tr>
<td>D, 0</td>
<td>D, 0</td>
<td>C, loc, 0</td>
</tr>
</tbody>
</table>

**IS D:**

<table>
<thead>
<tr>
<th>Loc (D)</th>
<th>DV (B)</th>
<th>DV (C)</th>
<th>RT (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D, 0</td>
<td>B, 0</td>
<td>A, 2</td>
<td>A, D2, 3</td>
</tr>
<tr>
<td>C, 1</td>
<td>C, 0</td>
<td>B, 1</td>
<td>B, D1, 1</td>
</tr>
<tr>
<td>C, D2, 1</td>
<td>C, D2, 1</td>
<td>C, D2, 1</td>
<td></td>
</tr>
<tr>
<td>D, loc, 0</td>
<td>D, loc, 0</td>
<td>D, loc, 0</td>
<td></td>
</tr>
</tbody>
</table>

B sends its DV
### Split Horizon on Mesh

#### IS C:
(from the previous slide)

<table>
<thead>
<tr>
<th>Loc</th>
<th>DV (B)</th>
<th>DV (D)</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, 0</td>
<td>B, 0</td>
<td>A, 2</td>
<td>A,C2, 3</td>
</tr>
<tr>
<td>D, 1</td>
<td>B, 1</td>
<td>C,loc,0</td>
<td>B,C1, 1</td>
</tr>
</tbody>
</table>

C and D send their DVs

#### IS B:

<table>
<thead>
<tr>
<th>Loc</th>
<th>DV (C)</th>
<th>DV (D)</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B, 0</td>
<td>A, 3</td>
<td>A, 3</td>
<td>A,B3, 4</td>
</tr>
<tr>
<td>C, 0</td>
<td>C, 1</td>
<td>C,loc,0</td>
<td>B,C2, 1</td>
</tr>
<tr>
<td>D, 1</td>
<td>D, 0</td>
<td>D,B3, 1</td>
<td></td>
</tr>
</tbody>
</table>

#### IS C:

<table>
<thead>
<tr>
<th>Loc</th>
<th>DV (B)</th>
<th>DV (D)</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, 0</td>
<td>B, 0</td>
<td>B, 1</td>
<td>B,C1, 1</td>
</tr>
<tr>
<td>D, 1</td>
<td>D, 0</td>
<td>C,loc,0</td>
<td>C,D2, 1</td>
</tr>
</tbody>
</table>

#### IS D:

<table>
<thead>
<tr>
<th>Loc</th>
<th>DV (B)</th>
<th>DV (D)</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>D, 0</td>
<td>B, 0</td>
<td>B, 1</td>
<td>B,D1, 1</td>
</tr>
<tr>
<td>C, 1</td>
<td>C, 0</td>
<td>C,D2, 1</td>
<td></td>
</tr>
<tr>
<td>D, 0</td>
<td>D,loc,0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Path Hold Down

If link L fails, all destinations reachable through link L are considered unreachable for a certain period of time. I.e., no routes to them are computed.
Path Hold Down

- High convergence time for the examined node (even with an alternative path)
- The router that noted the fault may not participate to any loop at least until the timeout of Hold Down timer
Route Poisoning

Routing loops happen when routes have increasing costs

Cost-increasing routes in DVs are not used

Two subsequent advertisements show a cost increase
Route Poisoning

- Effective in more cases compared to the path Hold Down
- Faster convergence time
- Not necessary to wait for the Hold Down timer
- Route is computed when two subsequent DVs confirm the same cost
- It blocks routes whose cost increase may be legitimate
- It is implemented in some protocols as a substitute of the Path Hold Down
The Bottom Line

Routers do not know the network topology

Based on distance vectors B cannot distinguish

A ——— B ——— C

A ——— B ——— C
Advantages

- Simple to implement
- Protocols simple to deploy
- Very little configuration
Shortcomings

- Exponential worst case complexity and convergence time
- $O(n^2)$ to $O(n^3)$
Shortcomings

→ Convergence limited by slower links and routers set pace

→ Complex tuning
Shortcomings

→ Complex troubleshooting
→ Large routing traffic (and storage)

Not suitable for large complex networks
# Path Vector

Eliminates routing loops

**IS A:**

<table>
<thead>
<tr>
<th>Loc (A)</th>
<th>PV (B)</th>
<th>PV (C)</th>
<th>RT (A)</th>
<th>PV (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, 0</td>
<td>A, 1, [B]</td>
<td>A, 1 [C]</td>
<td>A, loc, 0</td>
<td>A, 0, [- ]</td>
</tr>
<tr>
<td></td>
<td>B, 0, [- ]</td>
<td>B, 1, [B]</td>
<td>B, A1, 1</td>
<td>B, 1, [A]</td>
</tr>
<tr>
<td></td>
<td>C, 1, [B]</td>
<td>C, 0, [- ]</td>
<td>C, A2, 1</td>
<td>C, 1, [A]</td>
</tr>
<tr>
<td></td>
<td>D, 2, [B,C]</td>
<td>D, 1, [D]</td>
<td>D, A2, 2</td>
<td>D, 2, [A,C]</td>
</tr>
</tbody>
</table>
LINK STATE ROUTING ALGORITHM
Basic Principles

1. I, node E, connect to A and F
2. I, node A, connect to E and C
3. I, node F, connect to E and C
Basic Principles

- Information on the state of each link
  - Link state
- A local map
  - Sent by each node to all other nodes
- Selective flooding
Basic Principles

→ Nodes build a network map
→ The same on all nodes
→ Each node computes routes on the map
→ Dijkstra (shortest path first) algorithm
Link State Database

![Diagram of a network with nodes A, B, C, D, E and their link state database entries.]

- **LS (A)**: D, 1
- **LS (B)**: A, 1; C, 1; E, 1
- **LS (C)**: B, 1
- **LS (D)**: A, 1; E, 1
- **LS (E)**: B, 1; C, 1; D, 1

Stored by each router.
Dijkstra Algorithm

→ Low complexity
  → $L \cdot \log (N)$

→ $L$: number of links
→ $N$: number of nodes

→ Shortest Path First
Shortest Path First

→ The next node “nearest” to the root is identified

→ Its path is inserted into the routing table
Example

Routing - 57
Rapid Convergence

- LSs spread quickly
- No intermediate processing
Limited Routing Traffic and Storage

- Link states are small
- Fast and efficient neighbor greeting
Other Advantages

→ It rarely generates loops
→ Simple to understand and troubleshoot
→ All nodes have identical databases
Shortcoming

- High implementation complexity
- Selective flooding
- First implementation took several years
- Protocols with complex configuration