Last week on Communication Networks
Internet Protocol and Forwarding

1. IP addresses
   use, structure, allocation

2. IP forwarding
   longest prefix match rule

3. IP header
   IPv4 and IPv6, wire format

source: Boardwatch Magazine
Internet Protocol and Forwarding

1. IP addresses
   use, structure, allocation

IP forwarding
   longest prefix match rule

IP header
   IPv4 and IPv6, wire format
IPv4 addresses are unique 32-bits number associated to a network interface (on a host, a router, ...)

IP addresses are usually written using dotted-quad notation

```
82.130.102.10
```

```
01010010 10000010 01100110 00001010
```
IP addressing is hierarchical, composed of a prefix (network address) and a suffix (host address)
Each prefix has a given length, usually written using a “slash notation”

<table>
<thead>
<tr>
<th>IP prefix</th>
<th>82.130.102.0 /24</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix length</td>
<td>(in bits)</td>
</tr>
</tbody>
</table>
Prefixes are also sometimes specified using an address and a mask

<table>
<thead>
<tr>
<th>Address</th>
<th>82.130.102.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01010010.10000010.01100110. 00000000</td>
</tr>
<tr>
<td></td>
<td>11111111.11111111.11111111. 00000000</td>
</tr>
<tr>
<td>Mask</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>
Routers forward packet to their destination according to the network part, *not* the host part.
Doing so enables to scale the forwarding tables.
ICANN allocates large prefixes blocks to Regional Internet Registries (RIRs)

- ARIN: America
- LACNIC: Latin America
- RIPE NCC: Europe
- APNIC: Asia-Pacific
- AFRINIC: Africa
Internet Protocol and Forwarding

IP addresses
use, structure, allocation

2

IP forwarding
longest prefix match rule

IP header
IPv4 and IPv6, wire format
Routers maintain forwarding entries for each Internet prefix
Provider 2’s Forwarding table

<table>
<thead>
<tr>
<th>IP prefix</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>129.0.0.0/8</td>
<td>IF#2</td>
</tr>
<tr>
<td>129.132.1.0/24</td>
<td>IF#2</td>
</tr>
<tr>
<td>129.132.2.0/24</td>
<td>IF#2</td>
</tr>
<tr>
<td>129.133.0.0/16</td>
<td>IF#3</td>
</tr>
</tbody>
</table>

Diagram:

- Provider 1
  - 129.132.1.0/24
  - 129.132.2.0/24
  - 129.132.4.0/24
  - 129.133.0.0/16

- Provider 2
  - 129.0.0.0/8
  - IF#2
  - IF#3
To resolve ambiguity, forwarding is done along the *most specific* prefix (*i.e.*, the longer one)
Let's say a packet for **129.133.0.1** arrives at Provider 2

> Provider 2 forwards it to IF#3
Internet Protocol and Forwarding

IP addresses
use, structure, allocation

IP forwarding
longest prefix match rule

IP header
IPv4 and IPv6, wire format
<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>header length</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Type of Service</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Total Length</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td>3</td>
<td>Identification</td>
</tr>
<tr>
<td>Flags</td>
<td>3</td>
<td>Flags</td>
</tr>
<tr>
<td>Fragment offset</td>
<td>13</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>Time To Live</td>
<td>3</td>
<td>Time To Live</td>
</tr>
<tr>
<td>Protocol</td>
<td>4</td>
<td>Protocol</td>
</tr>
<tr>
<td>Header checksum</td>
<td>1</td>
<td>Header checksum</td>
</tr>
<tr>
<td>Source IP address</td>
<td>4</td>
<td>Source IP address</td>
</tr>
<tr>
<td>Destination IP address</td>
<td>4</td>
<td>Destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>4</td>
<td>Options (if any)</td>
</tr>
<tr>
<td>Payload</td>
<td>4</td>
<td>Payload</td>
</tr>
</tbody>
</table>
This week on

Communication Networks
Internet routing

http://www.opte.org
traceroute www.google.ch
$\text{traceroute www.google.ch}$

1. rou-etx-1-ee-tik-etx-dock-1 (82.130.102.1)
2. rou-ref-rz-bb-ref-rz-etx (10.10.0.41)
3. rou-fw-rz-ee-tik (10.1.11.129)
4. rou-fw-rz-gw-rz (192.33.92.170)
5. swiix1-10ge-1-4.switch.ch (130.59.36.41)
6. swiez2 (192.33.92.11)
7. swiix2-p1.switch.ch (130.59.36.250)
8. equinix-zurich.net.google.com (194.42.48.58)
9. 66.249.94.157 (66.249.94.157)
10. zrh04s06-in-f24.1e100.net (173.194.40.88)
Internet routing comes into two flavors: 

*intra-* and *inter-domain* routing

- **inter-domain routing**
  - Find paths between networks

- **intra-domain routing**
  - Find paths within a network
Find paths *between* networks
Google can be reached via NEWY, WASH, ATLA, HOUS
Google can be reached via NEWY, WASH, ATLA, HOUS

best exit point

based on money, performance, ...
Find paths within a network
NEWY can be reached via SALT
traceroute www.google.ch

rou-etx-1-ee-tik-etx-dock-1
rou-ref-rz-bb-ref-rz-etx
rou-fw-rz-ee-tik
rou-fw-rz-gw-rz
swiix1-10ge-1-4.switch.ch
swiez2
swiix2-p1.switch.ch
equinix-zurich.net.google.com
66.249.94.157
zrh04s06-in-f24.1e100.net

intra-domain routing
traceroute www.google.ch

rou-etx-1-ee-tik-etx-dock-1
rou-ref-rz-bb-ref-rz-etx
rou-fw-rz-ee-tik
rou-fw-rz-gw-rz
swiix1-10ge-1-4.switch.ch
swiez2
swiix2-p1.switch.ch
equinix-zurich.net.google.com
66.249.94.157
zrh04s06-in-f24.1e100.net

inter-domain routing

inter-domain routing
Internet routing
from here to there, and back

1  Intra-domain routing
   Link-state protocols
   Distance-vector protocols

2  Inter-domain routing
   Path-vector protocols
Internet routing
from here to there, and back

1. Intra-domain routing
   - Link-state protocols
   - Distance-vector protocols

2. Inter-domain routing
   - Path-vector protocols
Intra-domain routing enables routers to compute **forwarding paths** to any internal subnet.

what kind of paths?
Network operators don’t want arbitrary paths, they want **good paths**

<table>
<thead>
<tr>
<th>definition</th>
<th>A good path is a path that minimizes some network-wide metric typically delay, load, loss, cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>approach</td>
<td>Assign to each link a weight <em>(usually static)</em>, compute the <em>shortest-path</em> to each destination</td>
</tr>
</tbody>
</table>
When weights are assigned proportionally to the distance, shortest-paths will minimize the end-to-end delay.
When weights are assigned proportionally to the distance, shortest-paths will **minimize the end-to-end delay** if traffic is such that there is no congestion.
When weights are assigned **inversely proportionally** to each link capacity, **throughput is maximized** if traffic is such that there is no congestion.
Internet routing
from here to there, and back

1

Intra-domain routing

Link-state protocols
Distance-vector protocols

Inter-domain routing
Path-vector protocols
In Link-State routing, routers build a precise map of the network by flooding local views to everyone.

Each router keeps track of its incident links and cost as well as whether it is up or down.

Each router broadcast its own links state to give every router a complete view of the graph.

Routers run Dijkstra on the corresponding graph to compute their shortest-paths and forwarding tables.
Flooding is performed as in L2 learning

Node sends its link-state on all its links

Next node does the same, except on the one where the information arrived
Flooding is performed as in L2 learning, except that it is reliable.

Node sends its link-state on all its links.

Next node does the same, except on the one where the information arrived.

All nodes are ensured to receive the latest version of all link-states.

Challenges include packet loss and out of order arrival.
Flooding is performed as in L2 learning, except that it is reliable

Node sends its link-state on all its links

Next node does the same, except on the one where the information arrived

All nodes are ensured to receive the latest version of all link-states

solutions

ACK & retransmissions

sequence number

time-to-live for each link-state
A link-state node initiate flooding in 3 conditions

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology change</td>
<td>link or node failure/recovery</td>
</tr>
<tr>
<td>Configuration change</td>
<td>link cost change</td>
</tr>
<tr>
<td>Periodically</td>
<td>refresh the link-state information</td>
</tr>
<tr>
<td></td>
<td>every (say) 30 minutes</td>
</tr>
<tr>
<td></td>
<td>account for possible data corruption</td>
</tr>
</tbody>
</table>
Once a node knows the entire topology, it can compute shortest-paths using Dijkstra’s algorithm.
By default, Link-State protocols detect topology changes using software-based beaconing.

Routers periodically exchange “Hello” in both directions (e.g., every 30s).

Trigger a failure after few missed “Hellos” (e.g., after 3 missed ones).

Tradeoffs between:

- detection speed
- bandwidth and CPU overhead
- false positive/negatives
During network changes, the link-state database of each node might differ.

- control-plane consistency
  - necessary
- forwarding validity
  - all nodes have the same link-state database
  - the global forwarding state directs packet to its destination
Inconsistencies lead to transient disruptions in the form of blackholes or forwarding loops
Blackholes appear due to detection delay, as nodes do not immediately detect failure
depends on the timeout for detecting lost hellos
Transient loops appear due to inconsistent link-state databases

Initial forwarding state
C learns about the failure and immediately reroute to E
A loop appears as E isn’t yet aware of the failure
The loop disappears as soon as E updates its forwarding table.
Convergence is the process during which the routers seek to actively regain a consistent view of the network.
Network convergence time depends on 4 main factors

<table>
<thead>
<tr>
<th>factors</th>
<th>time the routers take for…</th>
</tr>
</thead>
<tbody>
<tr>
<td>detection</td>
<td>realizing that a link or a neighbor is down</td>
</tr>
<tr>
<td>flooding</td>
<td>flooding the news to the entire network</td>
</tr>
<tr>
<td>computation</td>
<td>recomputing shortest-paths using Dijkstra</td>
</tr>
<tr>
<td>table update</td>
<td>updating their forwarding table</td>
</tr>
</tbody>
</table>
In practice, network convergence time is mostly driven by table updates

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th>improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>detection</td>
<td>few ms</td>
<td>smaller timers</td>
</tr>
<tr>
<td>flooding</td>
<td>few ms</td>
<td>high-priority flooding</td>
</tr>
<tr>
<td>computation</td>
<td>few ms</td>
<td>incremental algorithms</td>
</tr>
<tr>
<td>table update</td>
<td>potentially, minutes!</td>
<td>better table design</td>
</tr>
</tbody>
</table>
table update  potentially, minutes!  better table design
## R1’s Forwarding Table

<table>
<thead>
<tr>
<th>prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>512k IP prefixes</td>
<td>0</td>
</tr>
</tbody>
</table>

### Providers

**Provider #1 ($$)**
- IP: 203.0.113.1
- MAC: 01:aa

**Provider #2 ($$)**
- IP: 198.51.100.2
- MAC: 02:bb
All 512k entries point to R2
because it is cheaper

R1's Forwarding Table

<table>
<thead>
<tr>
<th>prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
</tr>
<tr>
<td>2</td>
<td>1.0.1.0/16</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.0/8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.99.0.0/24</td>
</tr>
</tbody>
</table>

Provider #1 ($)
IP: 203.0.113.1
MAC: 01:aa

Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb

All 512k entries point to R2 because it is cheaper.
Upon failure of R2, all 512k entries have to be updated

R1’s Forwarding Table

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1.0.0.0/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>2 1.0.1.0/16</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k 100.0.0.0/8</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k 200.99.0.0/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>
Upon failure of R2, all 512k entries have to be updated

### R1’s Forwarding Table

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
</tr>
<tr>
<td>2</td>
<td>1.0.1.0/16</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.0/8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.99.0.0/24</td>
</tr>
</tbody>
</table>

Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb
### R1’s Forwarding Table

<table>
<thead>
<tr>
<th>prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0.0/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>1.0.1.0/16</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>200.99.0.0/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>

Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb
### R1’s Forwarding Table

<table>
<thead>
<tr>
<th>prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1.0.0.0/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>2 1.0.1.0/16</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k 100.0.0.0/8</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k 200.99.0.0/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>

---

**Diagram:**

- **R1**
  - IP: 198.51.100.2
  - MAC: 02:bb

- **R3**

**Provider #2 ($$)**

- IP: 198.51.100.2
- MAC: 02:bb
R1’s Forwarding Table

<table>
<thead>
<tr>
<th>prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24 (02:bb, 1)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.1.0/16 (02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.0/8 (02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.99.0.0/24 (01:aa, 0)</td>
</tr>
</tbody>
</table>

Provider #2 ($$)
IP: 198.51.100.2
MAC: 02:bb
### R1’s Forwarding Table

<table>
<thead>
<tr>
<th>prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24 (02:bb, 1)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.1.0/16 (02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.0/8 (02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.99.0.0/24 (02:bb, 1)</td>
</tr>
</tbody>
</table>

Provider #2 ($$)
- IP: 198.51.100.2
- MAC: 02:bb
How long does it take for ETH routers to converge?

Cisco Nexus 9k
ETH recent routers
25 deployed
Traffic can be lost for several minutes

~2.5 min.
The problem is that forwarding tables are flat.

Entries do not share any information even if they are identical.

Upon failure, all of them have to be updated inefficient, but also unnecessary.
Two universal tricks you can apply to any computer sciences problem

When you need... more flexibility,
you add... a layer of indirection

When you need... more scalability,
you add... a hierarchical structure
When you need... more flexibility,
you add... a layer of indirection
Router Forwarding Table

<table>
<thead>
<tr>
<th>prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0.0/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>1.0.1.0/16</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>100.0.0.0/8</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>200.99.0.0/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>

Replace this...
... with that

Router Forwarding Table

Mapping table

<table>
<thead>
<tr>
<th>prefix</th>
<th>pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
</tr>
<tr>
<td>2</td>
<td>1.0.1.0/16</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.0/8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.99.0.0/24</td>
</tr>
</tbody>
</table>

Pointer table

<table>
<thead>
<tr>
<th>pointer</th>
<th>NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x666</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>
Upon failures, we update the pointer table

Router Forwarding Table

Mapping table

<table>
<thead>
<tr>
<th>prefix</th>
<th>pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1.0.0.0/24</td>
<td>0x666</td>
</tr>
<tr>
<td>2 1.0.1.0/16</td>
<td>0x666</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k 100.0.0.0/8</td>
<td>0x666</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k 200.99.0.0/24</td>
<td>0x666</td>
</tr>
</tbody>
</table>

Pointer table

- pointer: 0x666
- NH: (01:aa, 0)

Ports: port 0, port 1
Here, we only need to do one update

**Router Forwarding Table**

<table>
<thead>
<tr>
<th>prefix</th>
<th>pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0.0/24</td>
<td>0x666</td>
</tr>
<tr>
<td>1.0.1.0/16</td>
<td>0x666</td>
</tr>
<tr>
<td>200.99.0.0/24</td>
<td>0x666</td>
</tr>
</tbody>
</table>

**Pointer table**

<table>
<thead>
<tr>
<th>pointer</th>
<th>NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x666</td>
<td>(02:bb, 1)</td>
</tr>
</tbody>
</table>

**Mapping table**

<table>
<thead>
<tr>
<th>prefix</th>
<th>pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
</tr>
<tr>
<td>2</td>
<td>1.0.1.0/16</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.0/8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.99.0.0/24</td>
</tr>
</tbody>
</table>
Hierarchical table enables to converge within 150ms, independently on the number of prefixes.
Today, two Link-State protocols are widely used: OSPF and IS-IS
Open Shortest Path First

used in many enterprise & ISPs
work on top of IP
only route IPv4 by default
**OSPF**

Open Shortest Path First

**IS-IS**

Intermediate Systems

used mostly in large ISPs
work on top of link-layer
network protocol agnostic
Internet routing
from here to there, and back

1  Intra-domain routing
   Link-state protocols
   Distance-vector protocols

Inter-domain routing
Path-vector protocols
Distance-vector protocols are based on Bellman-Ford algorithm
Let $d_x(y)$ be the cost of the least-cost path known by $x$ to reach $y$. 


Let $d_x(y)$ be the cost of the least-cost path known by $x$ to reach $y$.

Each node bundles these distances into one message (called a vector) that it repeatedly sends to all its neighbors until convergence.
Let $d_x(y)$ be the cost of the least-cost path known by $x$ to reach $y$

Each node bundles these distances into one message (called a vector) that it repeatedly sends to all its neighbors

Each node updates its distances based on neighbors’ vectors:

$$d_x(y) = \min \{ c(x,v) + d_v(y) \} \quad \text{over all neighbors } v$$
Similarly to Link-State, 3 situations cause nodes to send new DVs

- Topology change
- Configuration change
- Periodically

<table>
<thead>
<tr>
<th>Situation</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology change</td>
<td>link or node failure/recovery</td>
</tr>
<tr>
<td>Configuration change</td>
<td>link cost change</td>
</tr>
<tr>
<td>Periodically</td>
<td>refresh the link-state information</td>
</tr>
<tr>
<td></td>
<td>every (say) 30 minutes</td>
</tr>
<tr>
<td></td>
<td>account for possible data corruption</td>
</tr>
</tbody>
</table>
Optimum 1-hop path

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th></th>
<th>B</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
<td>Dst</td>
<td>Cst</td>
</tr>
<tr>
<td>A</td>
<td>0 A</td>
<td></td>
<td>A</td>
<td>4 A</td>
</tr>
<tr>
<td>B</td>
<td>4 B</td>
<td></td>
<td>B</td>
<td>0 B</td>
</tr>
<tr>
<td>C</td>
<td>∞ -</td>
<td></td>
<td>C</td>
<td>∞ -</td>
</tr>
<tr>
<td>D</td>
<td>∞ -</td>
<td></td>
<td>D</td>
<td>3 D</td>
</tr>
<tr>
<td>E</td>
<td>2 E</td>
<td></td>
<td>E</td>
<td>∞ -</td>
</tr>
<tr>
<td>F</td>
<td>6 F</td>
<td></td>
<td>F</td>
<td>1 F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th></th>
<th>D</th>
<th></th>
<th>E</th>
<th></th>
<th>F</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
</tr>
<tr>
<td>A</td>
<td>∞ -</td>
<td></td>
<td>A</td>
<td>∞ -</td>
<td></td>
<td>A</td>
<td>2 A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>∞ -</td>
<td></td>
<td>B</td>
<td>3 B</td>
<td></td>
<td>B</td>
<td>∞ -</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0 C</td>
<td></td>
<td>C</td>
<td>1 C</td>
<td></td>
<td>C</td>
<td>∞ -</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1 D</td>
<td></td>
<td>D</td>
<td>0 D</td>
<td></td>
<td>D</td>
<td>∞ -</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>∞ -</td>
<td></td>
<td>E</td>
<td>∞ -</td>
<td></td>
<td>E</td>
<td>0 E</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>1 F</td>
<td></td>
<td>F</td>
<td>∞ -</td>
<td></td>
<td>F</td>
<td>3 F</td>
<td></td>
</tr>
</tbody>
</table>
Optimum 1-hop path

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
</tr>
<tr>
<td>A</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>F</td>
</tr>
</tbody>
</table>
Optimum 2-hops path

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cost</td>
<td>Hop</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>7</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cost</td>
<td>Hop</td>
<td>Dst</td>
<td>Cost</td>
</tr>
<tr>
<td>A</td>
<td>7</td>
<td>F</td>
<td>A</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>F</td>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>D</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>F</td>
<td>E</td>
<td>∞</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
<td>F</td>
<td>2</td>
</tr>
</tbody>
</table>
Optimum 3-hops path

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>F</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
</tr>
</tbody>
</table>

C

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
<td>Dst</td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td>F</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>F</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>F</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

D

E

F
Let’s consider the convergence process after a link cost change
Consider the following network
Consider the following network leading to the following vectors:

**Y vector**
- **dest.** via
- $X\rightarrow Z$
- $X, 4, 6$

**Z vector**
- **dest.** via
- $X\rightarrow Y$
- $X, 50, 5$

**Y reaches X directly**

**Z reaches X via Y**
$t = 0$

$(X,Y)$ weight changes from 4 to 1

<table>
<thead>
<tr>
<th>time</th>
<th></th>
<th>t=0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Y vector</th>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>X</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Z vector</th>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>X</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
Node detects local cost change, update their vectors, and notify their neighbors if it has changed
$t = 1$

Y updates its vector, sends it to X and Z
t = 2

Z updates its vector, sends it to X and Y

<table>
<thead>
<tr>
<th>t=0</th>
<th>t=1</th>
<th>t=2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y vector</strong></td>
<td><strong>X</strong></td>
<td>4</td>
</tr>
<tr>
<td><strong>dest.</strong></td>
<td><strong>via</strong></td>
<td><strong>X</strong></td>
</tr>
<tr>
<td><strong>Z vector</strong></td>
<td><strong>dest.</strong></td>
<td><strong>via</strong></td>
</tr>
<tr>
<td><strong>X</strong></td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td><strong>dest.</strong></td>
<td><strong>via</strong></td>
<td><strong>X</strong></td>
</tr>
</tbody>
</table>
t = 3
Y updates its vector, sends it to X and Z

\[
\begin{array}{c|c|c}
\text{t=0} & \text{t=1} & \text{t=2} & \text{t=3} \\
\hline
\text{Y vector} & \text{dest.} & \text{via} & \text{dest.} & \text{via} & \text{dest.} & \text{via} \\
X & 4 & Z & X & 1 & 6 & X & 1 & 3 \\
Z & \text{dest.} & \text{via} & X & Y & \text{dest.} & \text{via} & X & Y \\
X & 50 & Y & X & 50 & 2 &
\end{array}
\]
t > 3
no one moves anymore
network has converged!

<table>
<thead>
<tr>
<th>t=0</th>
<th>t=1</th>
<th>t=2</th>
<th>t&gt;3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y vector</strong></td>
<td>dest. via</td>
<td>dest. via</td>
<td>dest. via</td>
</tr>
<tr>
<td></td>
<td>X Z</td>
<td>X Z</td>
<td>X Z</td>
</tr>
<tr>
<td></td>
<td>X 4 6</td>
<td>X 1 6</td>
<td>X 1 3</td>
</tr>
<tr>
<td><strong>Z vector</strong></td>
<td>dest. via</td>
<td>dest. via</td>
<td>dest. via</td>
</tr>
<tr>
<td></td>
<td>X Y</td>
<td>X Y</td>
<td>X Y</td>
</tr>
<tr>
<td></td>
<td>X 50 5</td>
<td>X 50 2</td>
<td>X 50 2</td>
</tr>
</tbody>
</table>
The algorithm terminates after 3 iterations

Good news travel fast!
Good news travel fast!

What about bad ones?
$t = 0$

$(X,Y)$ weight changes from 4 to 60

<table>
<thead>
<tr>
<th></th>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y vector</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>Z vector</td>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>X</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
$t = 1$

Y updates its vector, sends it to X and Z

<table>
<thead>
<tr>
<th>t=0</th>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y vector</td>
<td>dest.</td>
<td>via</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t=1</th>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y updates its vector, sends it to X and Z</td>
<td>dest.</td>
<td>via</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>60</td>
</tr>
</tbody>
</table>

$Z$ vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>X</td>
<td>50</td>
</tr>
</tbody>
</table>
$t = 2$

Z updates its vector, sends it to X and Y

\begin{align*}
\text{Y vector} & : \\
\text{t=0} & : \begin{array}{c} \text{dest.} \\ X \end{array} \begin{array}{c} \text{via} \\ X \end{array} \begin{array}{c} \text{Z} \\ X \end{array} & : X & 4 & 6 \\
\text{t=1} & : \begin{array}{c} \text{dest.} \\ X \end{array} \begin{array}{c} \text{via} \\ X \end{array} \begin{array}{c} \text{Z} \\ X \end{array} & : X & 60 & 6 \\
\text{Z vector} & : \begin{array}{c} \text{dest.} \\ X \end{array} \begin{array}{c} \text{via} \\ X \end{array} \begin{array}{c} \text{Y} \\ X \end{array} & : X & 50 & 5 \\
\text{t=2} & : \begin{array}{c} \text{dest.} \\ X \end{array} \begin{array}{c} \text{via} \\ X \end{array} \begin{array}{c} \text{Y} \\ X \end{array} & : X & 50 & 7
\end{align*}
t = 3

Y updates its vector, sends it to X and Z

t=0  t=1  t=2  t=3

**Y vector**

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>60</th>
<th>6</th>
</tr>
</thead>
</table>

**Z vector**

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>50</th>
<th>5</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>50</th>
<th>7</th>
</tr>
</thead>
</table>
\[ t = 4 \]

\( Z \) updates its vector, sends it to \( X \) and \( Y \)...
t=4

Y vector

... many iterations later ...

Z vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

X 50 9

dest. via

X 60 51

X 50 52
The algorithm terminates after 44 iterations!

Bad news travel slow!
This problem is known as count-to-infinity, a type of routing loop.

Count-to-infinity leads to very slow convergence. What if the cost had changed from 4 to 9999?

Routers don’t know when neighbors use them. Z does not know that Y has switched to use it.

Let’s try to fix that.
Whenever a router uses another one, it will announce it an infinite cost. The technique is known as poisoned reverse.
As Z uses Y to reach X, it announces to Y an infinite cost.
$t = 0$

$(X,Y)$ weight changes from 4 to 60

time

<table>
<thead>
<tr>
<th>vector</th>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$\infty$</td>
</tr>
<tr>
<td>Z</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
t = 1

Y updates its vector, sends it to X and Z

<table>
<thead>
<tr>
<th>t=0</th>
<th>t=1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y vector</strong></td>
<td><strong>Y vector</strong></td>
</tr>
<tr>
<td>dest. via</td>
<td>dest. via</td>
</tr>
<tr>
<td>X Z</td>
<td>X Z</td>
</tr>
<tr>
<td>X 4 ∞</td>
<td>X 60 ∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t=0</th>
<th>t=1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Z vector</strong></td>
<td><strong>Z vector</strong></td>
</tr>
<tr>
<td>dest. via</td>
<td>dest. via</td>
</tr>
<tr>
<td>X Y</td>
<td>X Y</td>
</tr>
<tr>
<td>X 50 5</td>
<td>X 50 5</td>
</tr>
</tbody>
</table>
t = 2

Z updates its vector, sends it to X and Y

t = 0

Y vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
</tr>
</tbody>
</table>

X  4  ∞

Z vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

X  50  5

t = 1

Y vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
</tr>
</tbody>
</table>

X  60  ∞

Z vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

X  50  5

t = 2

Y vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
</tr>
</tbody>
</table>

X  60  ∞

Z vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

X  50  61
At $t = 3$

Y updates its vector, sends it to X and Z

<table>
<thead>
<tr>
<th></th>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>t=0</td>
<td>X</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>50</td>
</tr>
<tr>
<td>t=1</td>
<td>X</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>50</td>
</tr>
<tr>
<td>t=2</td>
<td>X</td>
<td>60</td>
</tr>
<tr>
<td>t=3</td>
<td>X</td>
<td>60</td>
</tr>
</tbody>
</table>
$t = 4$

$Z$ updates its vector, sends it to $X$ and $Y$

$Y$

vector

$Z$

vector

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>$Y$</td>
</tr>
</tbody>
</table>

$X$ 50 $\infty$
\[ t > 4 \]

no one moves

network has converged!

\[ t = 4 \]

\[ t > 4 \]

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X 60</td>
<td>Z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X 50</td>
<td>∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dest.</th>
<th>via</th>
</tr>
</thead>
<tbody>
<tr>
<td>X 50</td>
<td>∞</td>
</tr>
</tbody>
</table>

\[ X \]

\[ Y \]

\[ Z \]
While poisoned reverse solved this case, it does \textbf{not} solve loops involving 3 or more nodes…

see exercise session
Actual distance-vector protocols mitigate this issue by using small “infinity”, e.g. 16
# Link-State vs Distance-Vector routing

<table>
<thead>
<tr>
<th></th>
<th>Message complexity</th>
<th>Convergence speed</th>
<th>Robustness</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Link-State</strong></td>
<td>$O(nE)$ message sent</td>
<td>relatively fast</td>
<td>node can advertise incorrect link cost</td>
</tr>
<tr>
<td></td>
<td>$n$: #nodes</td>
<td></td>
<td>nodes compute their own table</td>
</tr>
<tr>
<td></td>
<td>$E$: #links</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Distance-Vector</strong></td>
<td>between neighbors only</td>
<td>slow</td>
<td>node can advertise incorrect path cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>errors propagate</td>
</tr>
</tbody>
</table>
Internet routing
from here to there, and back

Intra-domain routing
Link-state protocols
Distance-vector protocols

Inter-domain routing
Path-vector protocols
Internet
Internet

A network of *networks*
Internet

Border Gateway Protocol (BGP)
The Internet is a network of networks, referred to as Autonomous Systems (AS)
Each AS has a number (encoded on 16 bits) which identifies it.
BGP is the routing protocol “glueing” the entire Internet together.
Using BGP, ASes exchange information about the IP prefixes they can reach, directly or indirectly.
BGP needs to solve three key challenges: scalability, privacy and policy enforcement.

There is a huge # of networks and prefixes:
700k prefixes, >50,000 networks, millions (!) of routers.

Networks don’t want to divulge internal topologies or their business relationships.

Networks need to control where to send and receive traffic without an Internet-wide notion of a link cost metric.
Link-State routing **does not solve** these challenges

- Floods topology information
- Requires each node to compute the entire path
- Minimizes some notion of total distance

...high processing overhead...works only if the policy is shared and uniform...
Distance-Vector routing is on the right track

**pros**

Hide details of the network topology
nodes determine only “next-hop” for each destination
Distance-Vector routing is on the right track, but not really there yet…

**Pros**
- Hide details of the network topology
- Nodes determine only “next-hop” for each destination

**Cons**
- It still minimizes some common distance
- Impossible to achieve in an inter domain setting
- It converges slowly
- Counting-to-infinity problem
BGP relies on path-vector routing to support flexible routing policies and avoid count-to-infinity.

Key idea: advertise the entire path instead of distances.
BGP announcements carry complete path information instead of distances.
Each AS appends itself to the path when it propagates announcements.

129.132.0.0/16
ETH/UNIZH Camp Net
129.132.0.0/16
Path: 10 40

129.132.0.0/16
Path: 50 10 40

129.132.0.0/16

ETH/UNIZH Camp Net
Complete path information enables ASes to easily detect a loop

ETH sees itself in the path and discard the route
Life of a BGP router is made of three consecutive steps

while true:

■ receives routes from my neighbors
■ select one best route for each prefix
■ export the best route to my neighbors
Each AS can apply local routing policies

Each AS is free to

- select and use any path
  preferably, the cheapest one
always prefer Deutsche Telekom routes over AT&T
always prefer Deutsche Telekom routes over AT&T
Each AS can apply local routing policies

Each AS is free to

- select and use any path
  preferably, the cheapest one

- decide which path to export (if any) to which neighbor
  preferably, none to minimize carried traffic
do not export ETH routes to AT&T
Do not export ETH routes to AT&T
Next week on
Communication Networks

Internet routing policies