Lecture #4: BGP routing

WPI CS4516 Spring 2019 D term

Instructor: Lorenzo De Carli (ldecarli@wpi.edu)

(slides include material from Christos Papadopoulos, CSU and Craig Shue, WPI)

What is this lecture about?

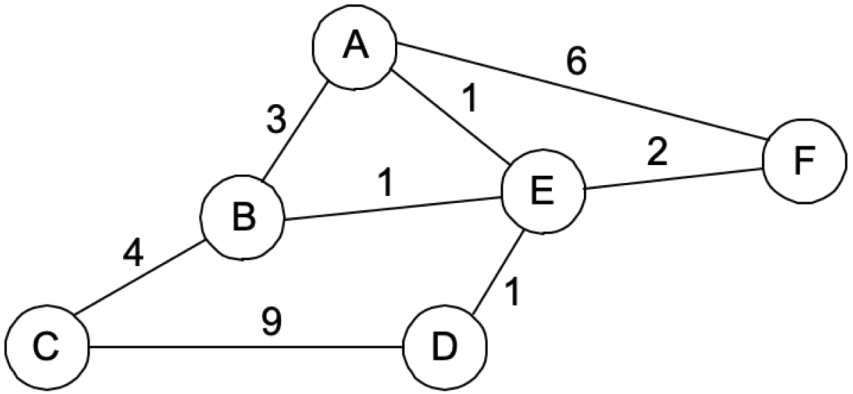
- A **general introduction to routing**
- A **review of the BGP routing protocol**
- A discussion of **BGP security issues**

Forwarding V.S. Routing

- Forwarding: the process of **moving packets from input to output** based on:
  - the forwarding table
  - information in the packet
- Routing: process by which the **forwarding table is built and maintained:**
  - one or more routing protocols
  - procedures (algorithms) to convert routing info to forwarding table

Factors Affecting Routing

- Routing algorithms **view the network as a graph**
- Problem: find **lowest cost path between two nodes**
- Factors
  - static: topology
  - dynamic: load
  - policy



Two Main Approaches

- **DV: Distance-vector protocols**
- LS: Link state protocols

Distance Vector Protocols

- Employed in the early Arpanet
- Distributed next hop computation
  - adaptive
- Unit of information exchange
  - vector of distances to destinations
- Distributed Bellman-Ford Algorithm

Distributed Bellman-Ford

Start Conditions:

Each router starts with a vector of (zero) distances to all directly attached networks

Send step:

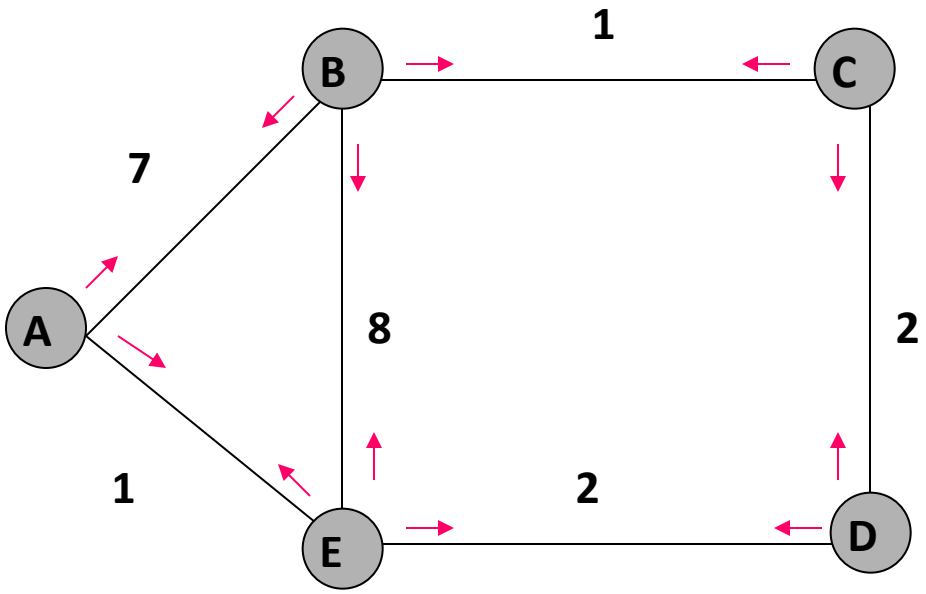
Each router advertises its current vector to all neighboring routers.

Receive step:

Upon receiving vectors from each of its neighbors, router computes its own distance to each neighbor. Then, for every network X, router finds that neighbor who is closer to X than any other neighbor.

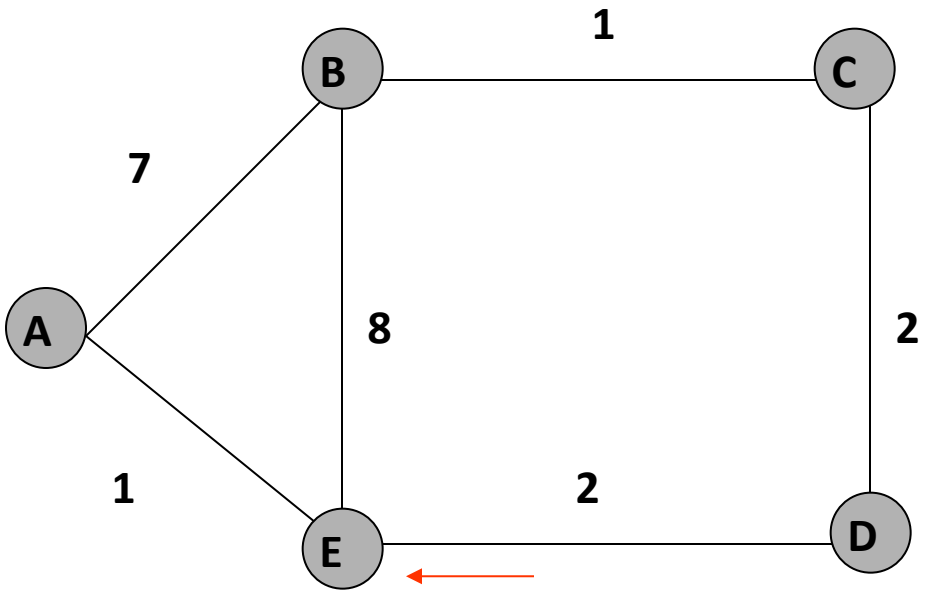
Router updates its cost to X. After doing this for all X, router goes to send step.

Example - Initial Distances



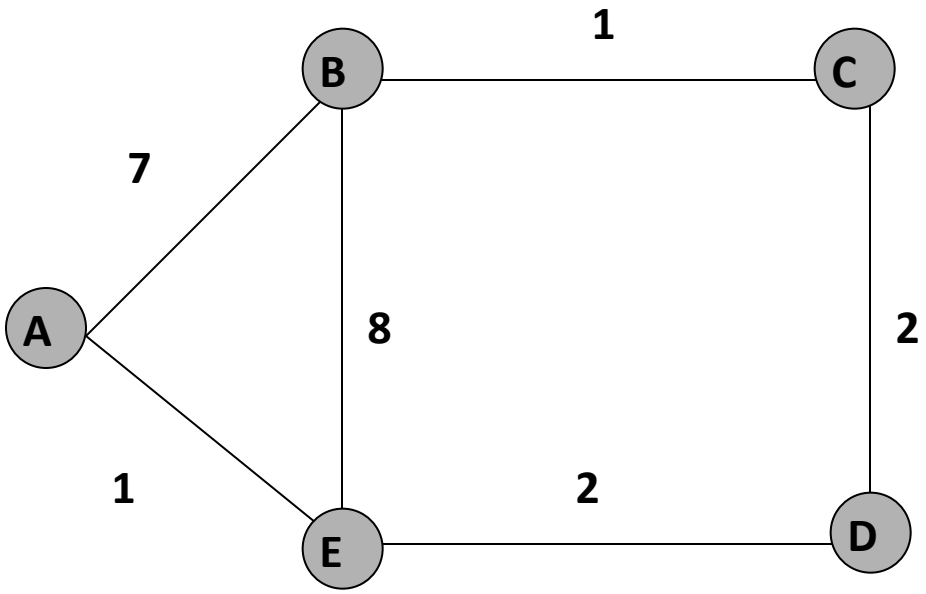
Info at node	Distance to node A	B	C	D	E
A	0	7	~	~	1
B	7	0	1	~	8
C	~	1	0	2	~
D	~	~	2	0	2
E	1	8	~	2	0

E Receives D's Routes



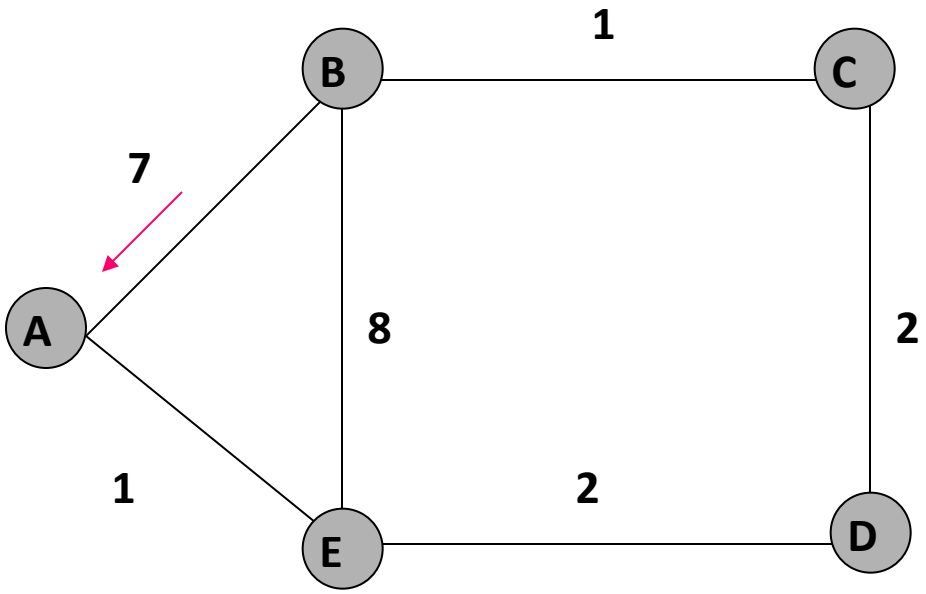
Info at node	Distance to node A	B	C	D	E
A	0	7	~	~	1
B	7	0	1	~	8
C	~	1	0	2	~
D	~	~	2	0	2
E	1	8	~	2	0

E Updates Cost to C



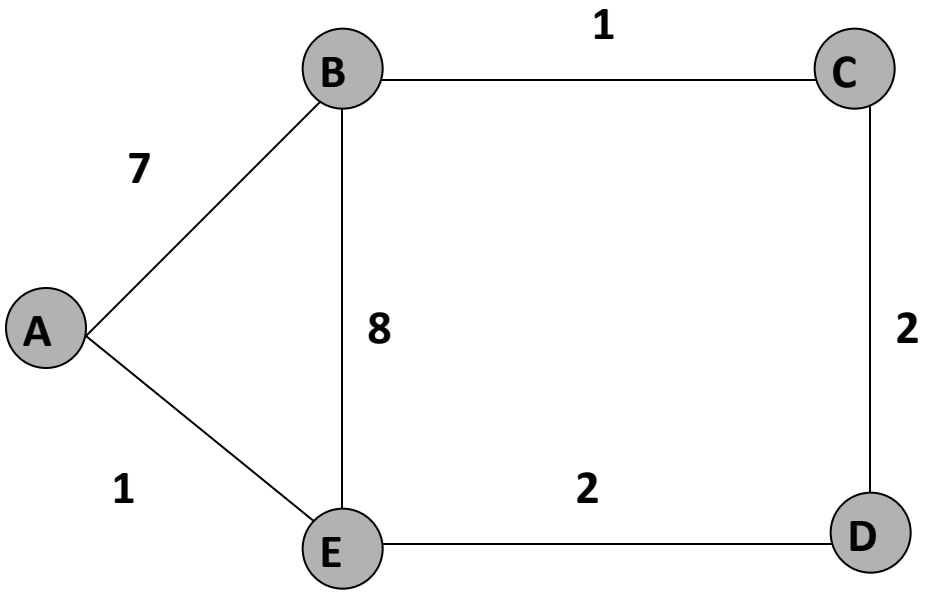
Info at node	Distance to node A	B	C	D	E
A	0	7	~	~	1
B	7	0	1	~	8
C	~	1	0	2	~
D	~	~	2	0	2
E	1	8	4	2	0

A Receives B's Routes



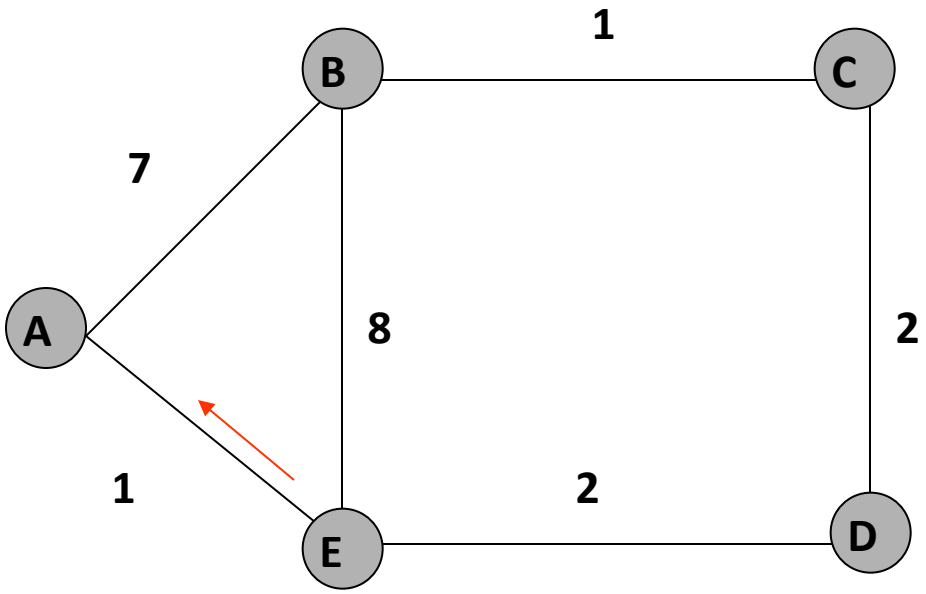
Info at node	Distance to node A	B	C	D	E
A	0	7	~	~	1
B	7	0	1	~	8
C	~	1	0	2	~
D	~	~	2	0	2
E	1	8	4	2	0

A Updates Cost to C



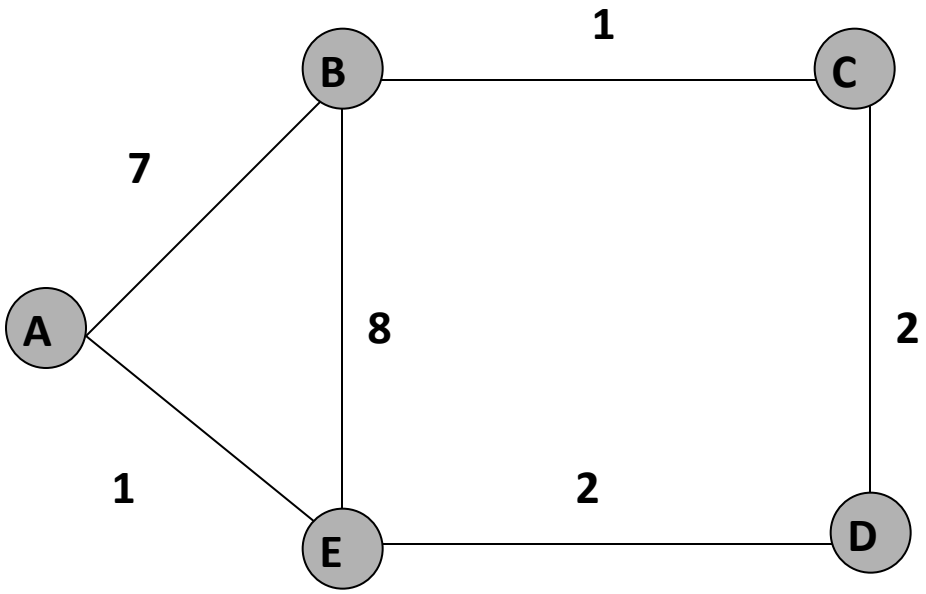
Info at node	Distance to node A	B	C	D	E
A	0	7	8	~	1
B	7	0	1	~	8
C	~	1	0	2	~
D	~	~	2	0	2
E	1	8	4	2	0

A Receives E's Routes



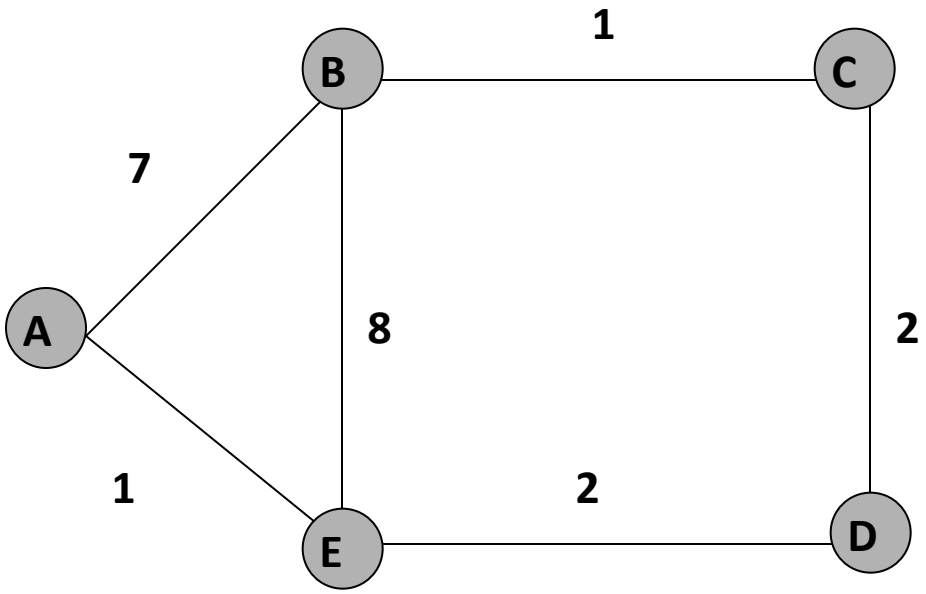
Info at node	Distance to node A	B	C	D	E
A	0	7	8	~	1
B	7	0	1	~	8
C	~	1	0	2	~
D	~	~	2	0	2
E	1	8	4	2	0

A Updates Cost to C and D



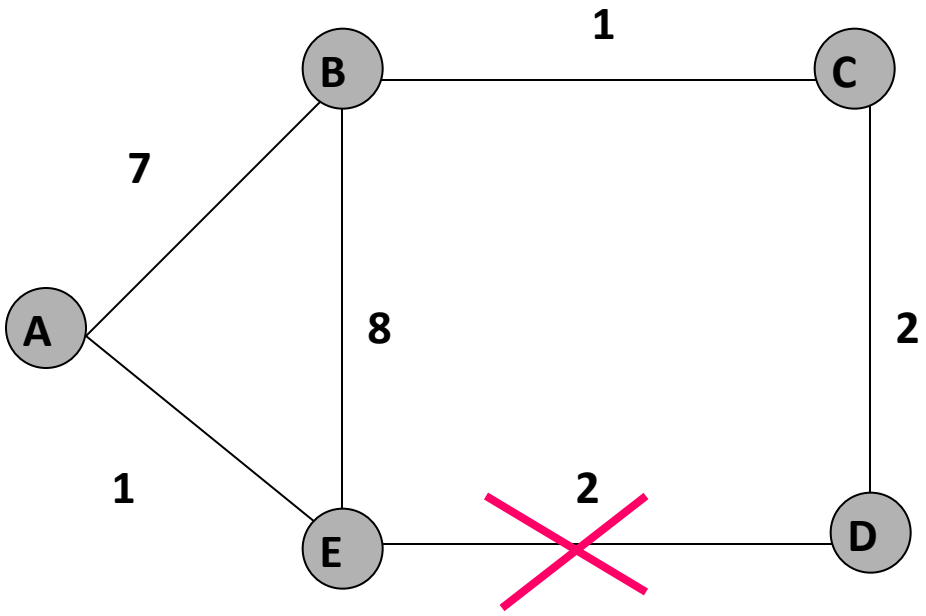
Info at node	Distance to node A	B	C	D	E
A	0	7	5	3	1
B	7	0	1	~	8
C	~	1	0	2	~
D	~	~	2	0	2
E	1	8	4	2	0

Final Distances



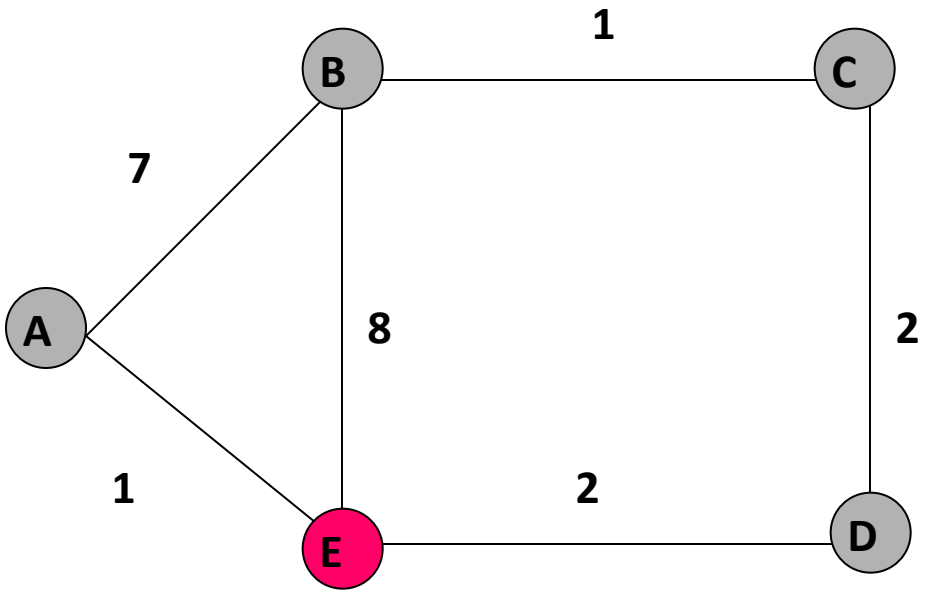
Info at node	Distance to node A	B	C	D	E
A	0	6	5	3	1
B	6	0	1	3	5
C	5	1	0	2	4
D	3	3	2	0	2
E	1	5	4	2	0

Final Distances After Link Failure



Info at node	Distance to node A	B	C	D	E
A	0	7	8	10	1
B	7	0	1	3	8
C	8	1	0	2	9
D	10	3	2	0	11
E	1	8	9	11	0

View From a Node

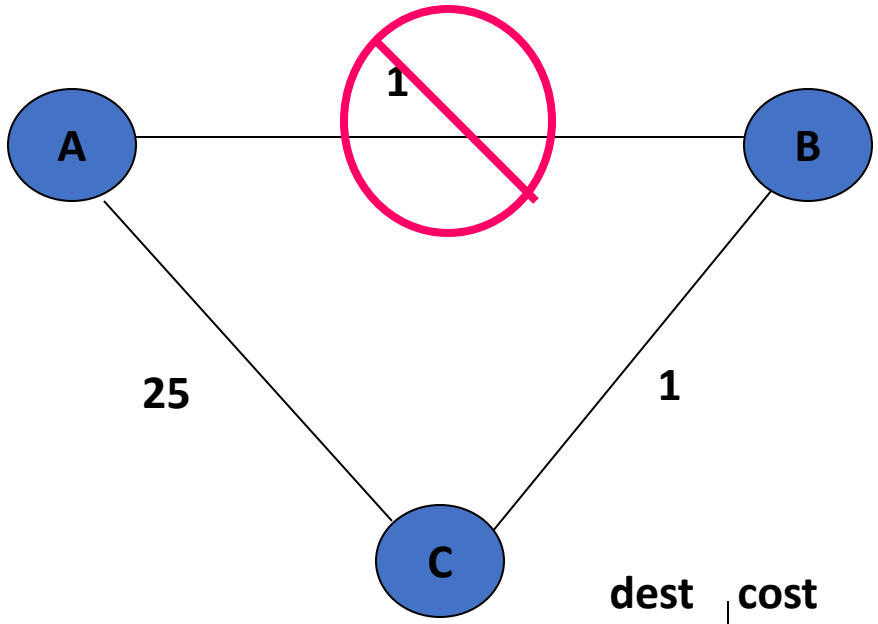


E's routing table

dest	Next hop		
	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

The Bouncing Effect

dest	cost
B	1
C	2

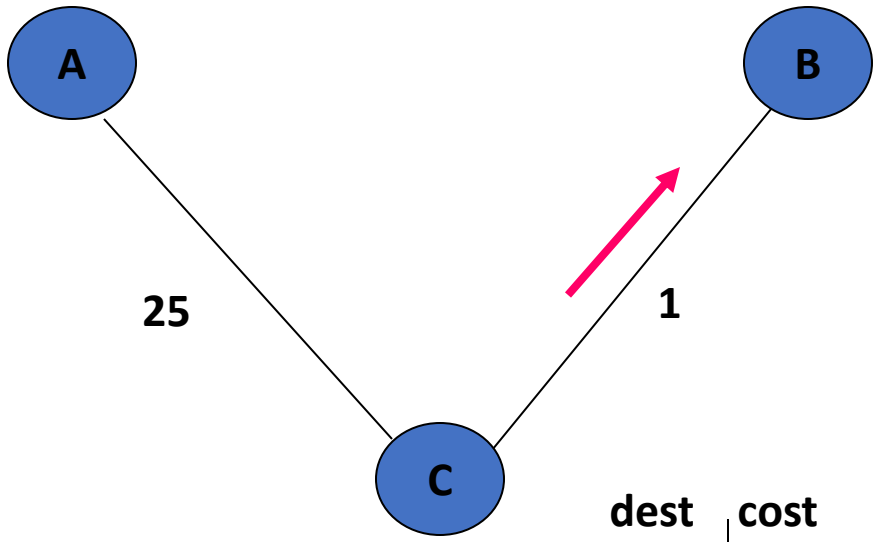


dest	cost
A	1
C	1

dest	cost
A	2
B	1

C Sends Routes to B

dest	cost
B	1
C	2

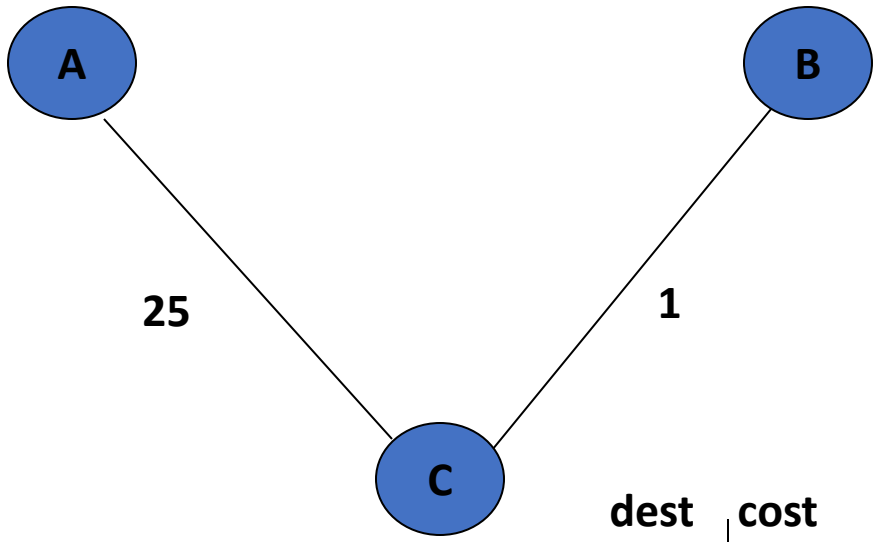


dest	cost
A	~
C	1

dest	cost
A	2
B	1

B Updates Distance to A

dest	cost
B	1
C	2

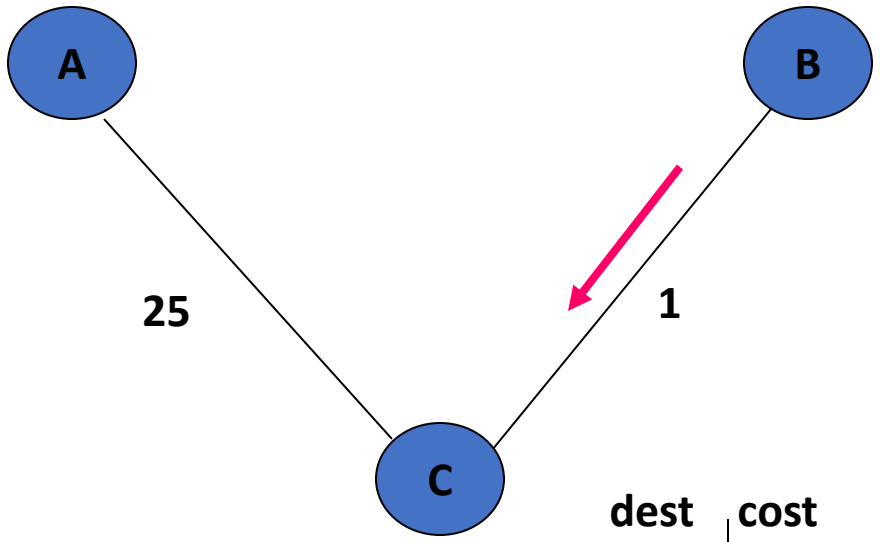


dest	cost
A	3
C	1

dest	cost
A	2
B	1

B Sends Routes to C

dest	cost
B	1
C	2

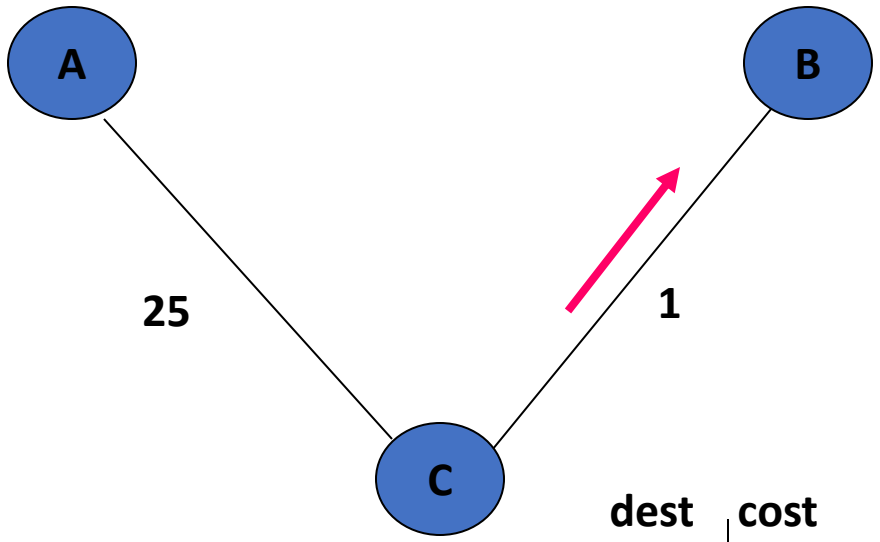


dest	cost
A	3
C	1

dest	cost
A	4
B	1

C Sends Routes to B

dest	cost
B	1
C	2



dest	cost
A	5
C	1

dest	cost
A	4
B	1

How Are These Loops Caused?

- Observation 1:
  - B's metric increases
- Observation 2:
  - C picks B as next hop to A
  - But, the implicit path from C to A includes itself!

Solution 1: Holddowns

- **If metric increases, delay propagating information**
  - in our example, B delays advertising route
  - C eventually thinks B's route is gone, picks its own route
  - B then selects C as next hop
- **Adversely affects convergence**

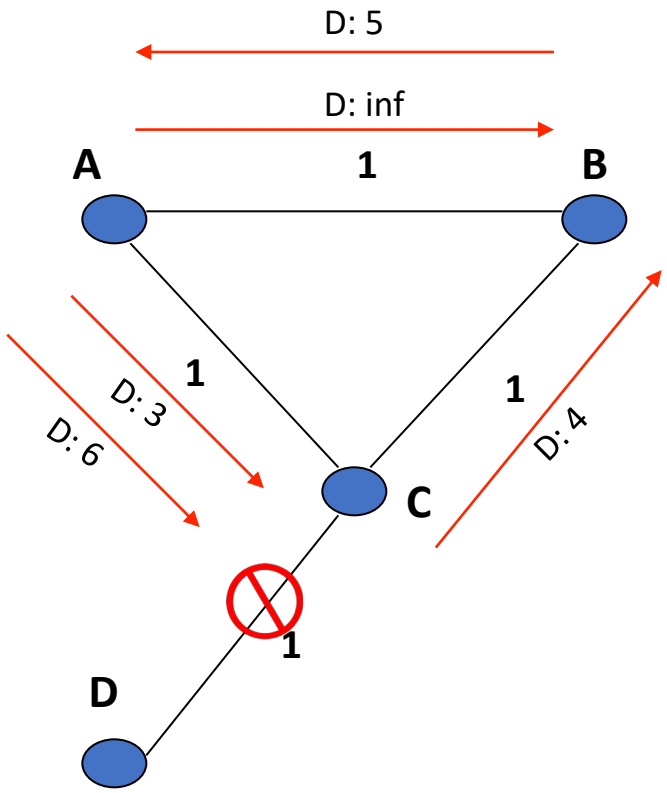
Other “Solutions”

- **Split horizon**
  - B does not advertise route to C
  - In general, **prevents routes from being advertised on the interface through which they were learned**
- Works for two-node loops
  - **does not work for loops with more than two nodes**

Poisoned Reverse

- Implementation/variation of the split horizon concept
- Instead of not advertising a route back to the node from which it was learned…
- …advertise the route as unreachable
- **Same drawback as split horizon**

Example Where Split Horizon Fails



1. When link breaks, C marks D as unreachable and reports that to A and B.
2. Suppose A learns it first. A now thinks best path to D is through B. A reports D unreachable to B and a route of cost=3 to C.
3. C thinks D is reachable through A at cost 4 and reports that to B.
4. B reports a cost 5 to A who reports new cost to C.
5. etc...

Avoiding the Bouncing Effect

Select loop-free paths

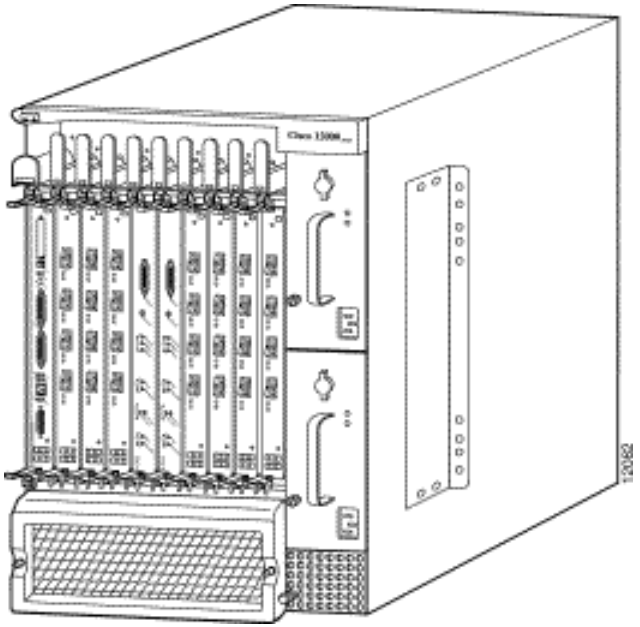
- One way of doing this:
  - each route advertisement carries entire path
  - if a router sees itself in path, it rejects the route

BGP does it this way

Distance Vector in Practice

- RIP and RIP2
  - uses split-horizon/poison reverse
- **BGP**/IDRP
  - **propagates entire path**
  - path also used for effecting policies

BGP and Inter-domain Routing



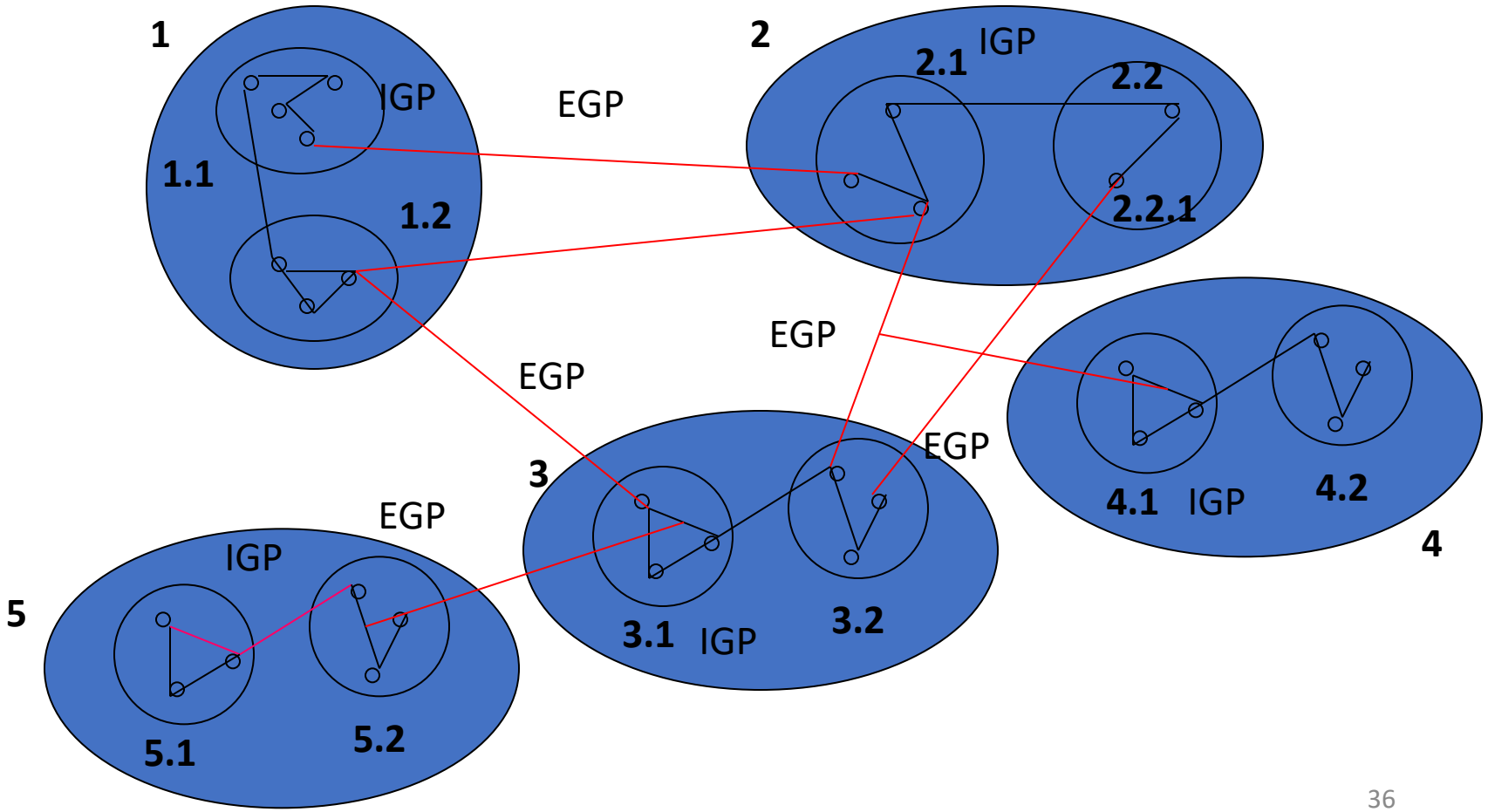
Sources

- John Stewart III: “BGP4 - Inter-domain routing in the Internet”
- RFC1771: main BGP RFC
- RFC1772-3-4: application, experiences, and analysis of BGP
- RFC1965: AS confederations for BGP
- Christian Huitema: “Routing in the Internet”, chapters 8 and 9
- Cisco tutorial on line

Autonomous Systems

- What is an **AS**?
  - a **set of routers under a single technical administration** (A single organization may own multiple ASes)
  - a **set of advertised prefixes**
  - uses an *interior gateway protocol (IGP)* and common metrics to route packets within the AS
  - uses an *exterior gateway protocol (EGP)* to route packets to other AS's
- AS may use multiple IGPs and metrics, but appears as single AS to other AS's

Example


1
IGP
1.1
1.2
EGP
2
IGP
2.1
2.2
2.2.1
EGP
EGP
EGP
3
3.1
IGP
3.2
EGP
4
4.1
IGP
4.2
5
IGP
5.1
5.2

History

- Mid-80s: EGP
  - **reachability** protocol (no shortest path)
  - did not accommodate **cycles** (tree topology)
  - evolved when all networks connected to ARPANET
- Limited size network topology
- Result: **BGP introduced as routing protocol**

Path Vectors

- Each routing update carries the **entire path as a sequence of ASes**
- Loops are detected as follows:
  - when AS gets route **check if AS already in path**
    - if yes, reject route
    - if no, add self and (possibly) advertise route further
- Advantage:
  - metrics are local - **AS chooses path, protocol ensures no loops**

Interconnecting BGP Peers

- BGP uses **TCP** to connect peers (port 179)
- Advantages:
  - BGP much simpler
  - **no need for periodic refresh** - routes are valid until withdrawn, or the connection is lost
  - **incremental updates**
- Disadvantages
  - congestion control on a routing protocol?

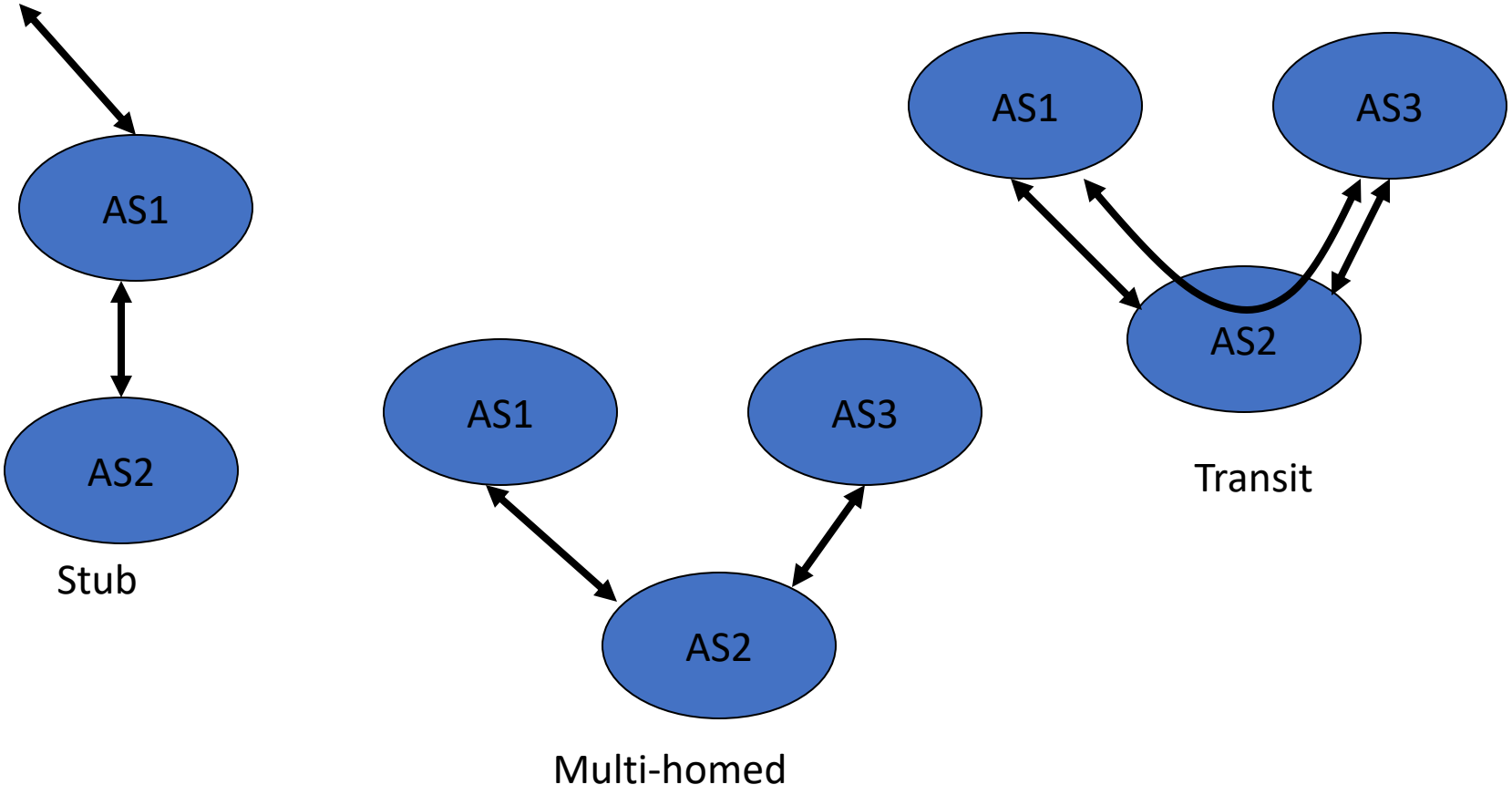
Hop-by-hop Model

- BGP advertises to neighbors **only those routes that it uses**
  - consistent with the hop-by-hop Internet paradigm
  - e.g., AS1 cannot tell AS2 to route to other ASes in a manner different than what AS2 has chosen (need source routing for that)

AS Categories

- **Stub**: an AS that has only a single connection to one other AS - carries only local traffic
- **Multi-homed**: an AS that has connections to more than one AS, but does not carry transit traffic
- **Transit**: an AS that has connections to more than one AS, and carries both transit and local traffic (under certain policy restrictions)

AS Categories



Policy With BGP

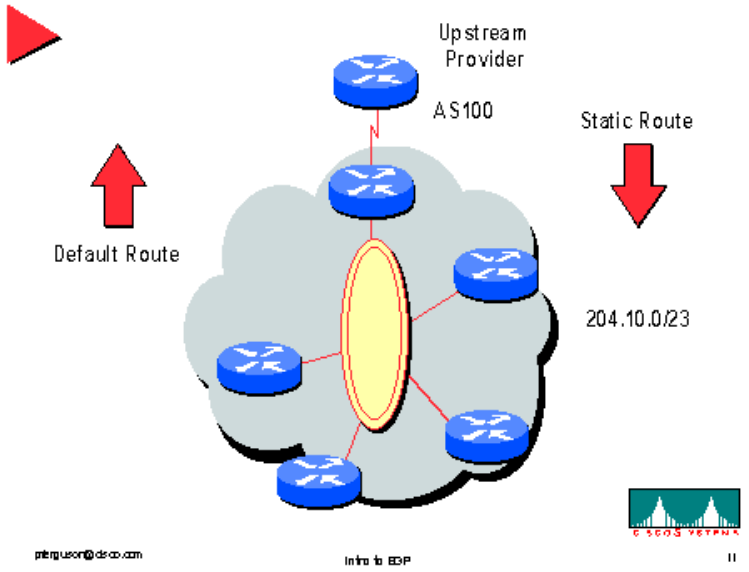
- BGP provides capability for enforcing various policies
- Policies are **not** part of BGP: they are provided to BGP as configuration information
- BGP enforces policies by choosing paths from multiple alternatives and controlling advertisement to other AS's

Examples of BGP Policies

- A multi-homed AS refuses to act as transit
  - limit path advertisement
- A multi-homed AS can become transit for some AS's
  - only advertise paths to some AS's
- An AS can favor or disfavor certain AS's for traffic transit from itself
  - Pick appropriate routes by examining path vectors

BGP Is NOT Needed If:

- Single homed network (stub)
- AS does not provide downstream routing
- AS uses a default route



Path Selection Criteria

- Information based on **path attributes**
- Attributes + external (**policy**) information
- Examples:
  - hop count
  - presence or absence of certain AS
  - path origin
  - link dynamics (flapping, stable)

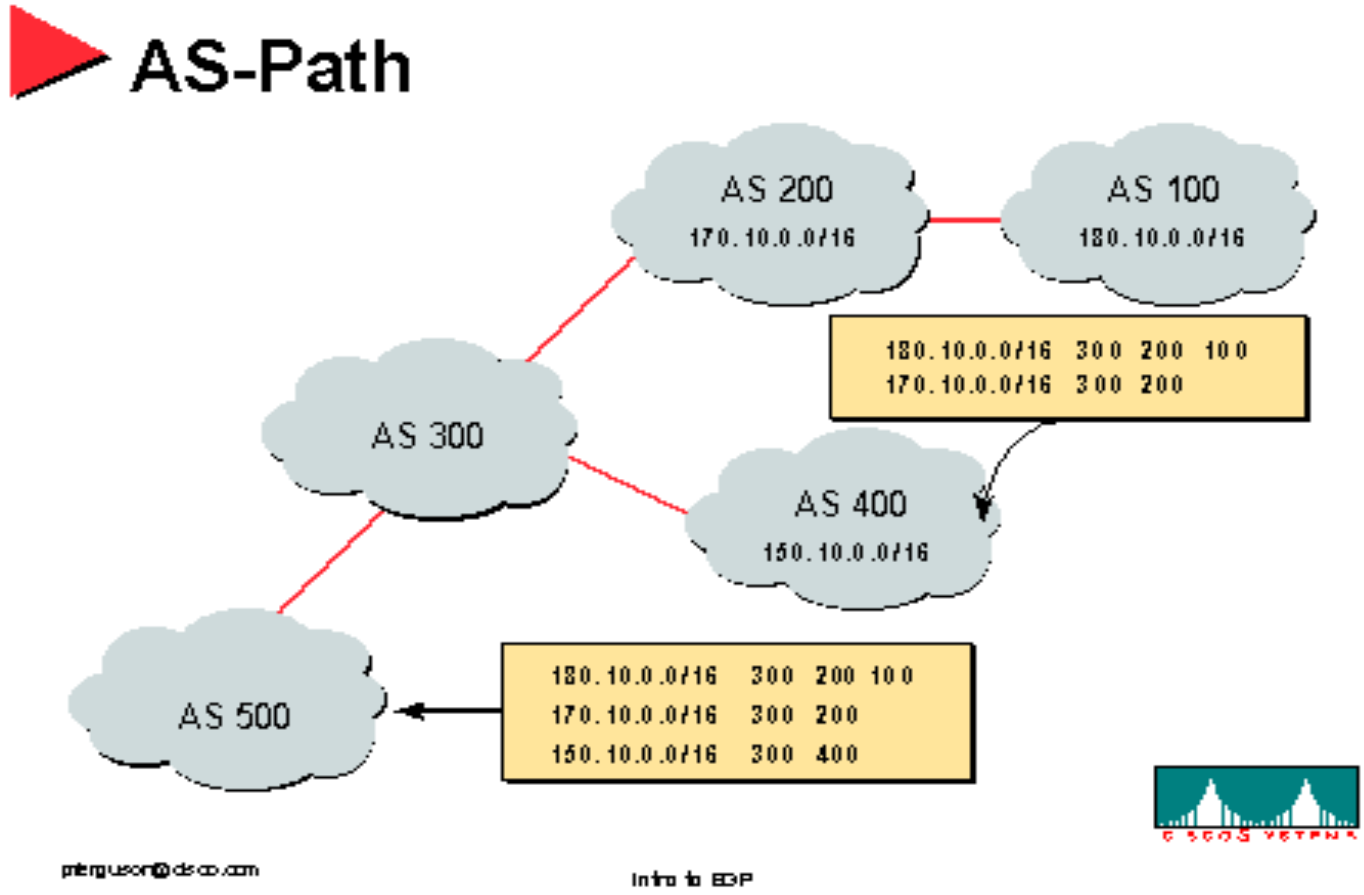
Path Attributes

- Categories (recall flags):
  - well-known mandatory (passed on)
  - well-known discretionary (passed on)
  - optional transitive (passed on)
  - optional non-transitive (if unrecognized, not passed on)
- **Optional attributes** allow for **BGP extensions**
- **Path attributes** are used by routers to select routes
- Many possible attributes; we are only going to discuss one example

AS_PATH Attribute

- Well-known, mandatory attribute
- Important components:
  - **list of traversed AS's**
- If forwarding to internal peer:
  - do not modify AS_PATH attribute
- If forwarding to **external peer**:
  - **prepend self into the path**

AS_PATH Attribute

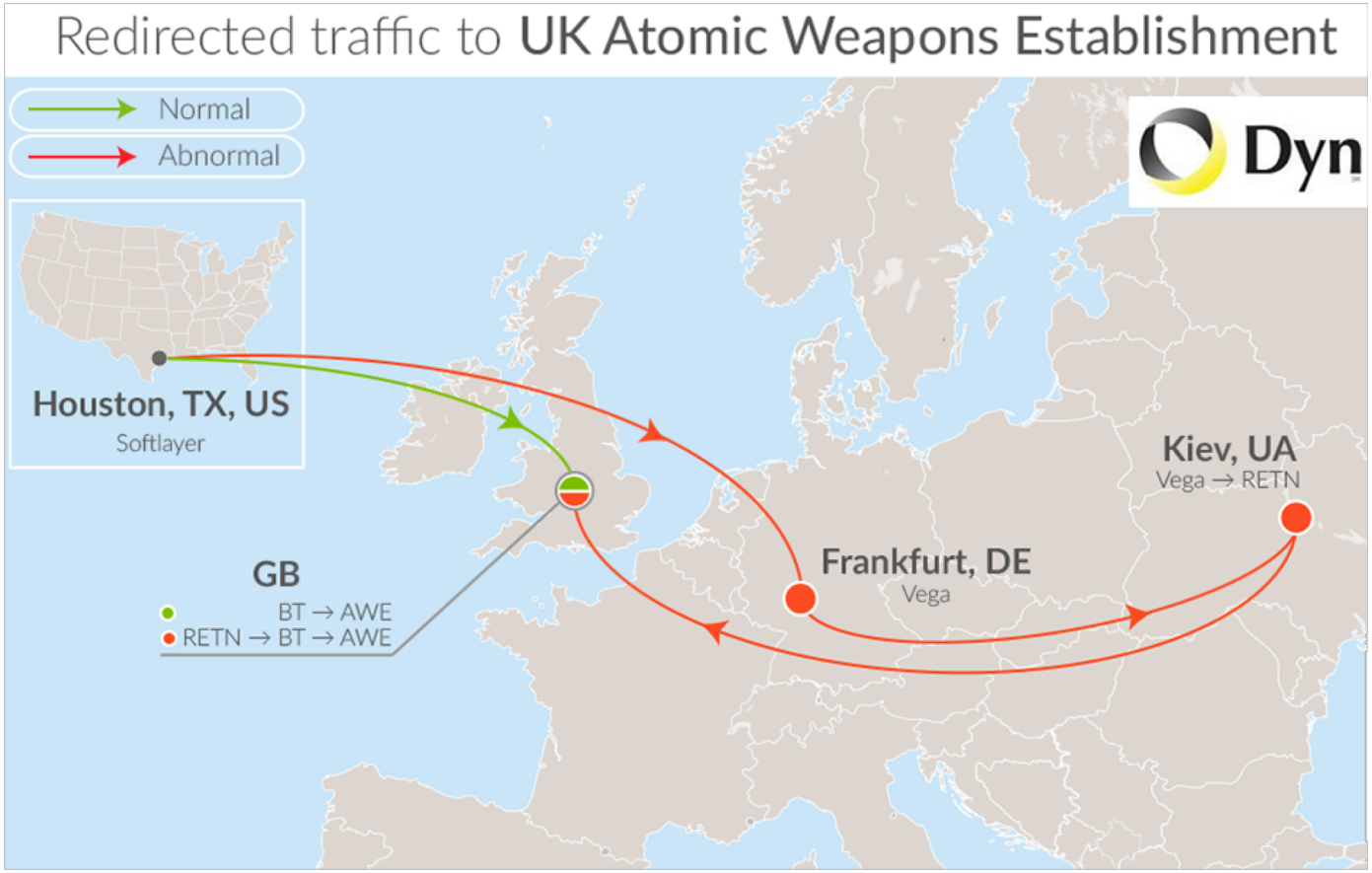


Now, let’s talk about BGP security

Prefix Hijacking

- Just originate a prefix you don’t own
  - I mean, who is to say you don’t own it?
- Originating the same prefix?
  - Your neighbors will prefer you
- Originating a longer prefix?
  - EVERYONE will prefer you
- End-goal is always the same:
  - **Get traffic you are not supposed to route to flow towards your network**

Prefix hijacking is a real problem!



https://dyn.com/blog/uk-traffic-diverted-ukraine/

Prefix hijacking - goals

- Once you have the traffic, you can
  - Blackhole (drop all packets)
  - Impersonate (become the destination)
  - Intercept (man-in-the-middle attacks)
- Usually, unexplained events are of the “intercept” variety
- Blackhole and the such may happen by mistake
  - On 2/24/2008 Pakistan Telecom created a blackhole that made Youtube unreachable for the entire world in an attempt to block access within he country

Prefix hijacking – goals II

- Certain intercept events have fairly transparent motivations

BLEEPINGCOMPUTER

Search Site

NEWS ▾ DOWNLOADS ▾ VIRUS REMOVAL GUIDES ▾ TUTORIALS ▾ DEALS ▾

Home > News > Security > U.S. Payment Processing Services Targeted by BGP Hijacking Attacks

U.S. Payment Processing Services Targeted by BGP Hijacking Attacks

By Lawrence Abrams August 6, 2018 03:01 AM 0

According to a new report, three United States payment processing companies were targeted by BGP hijacking attacks on their DNS servers. These Internet routing attacks were designed to redirect traffic directed at the payment processors to servers controlled by malicious actors who would then attempt to steal the data.

On three separate dates in July, Oracle has stated that they saw what appeared to be BGP hijacks that targeted the DNS servers for U.S. payment processors Datawire, Vantiv, or Mercury Payment Systems.

…or you can just mess with routers

- Generally use a TCP connection
  - May not be a direct link
- General TCP attacks
  - Resets, eavesdropping, integrity, DoS

Adversaries Know the Rules

- Break them to attack
  - Truncate AS path
  - Remove path entries
  - Add path entries (e.g., the victim)
  - Produce wrong origins, MEDs, etc.
    - Multi-exit discriminator: hint to other AS'es on how to reach an AS w/ multiple entry points

Secure BGP (S-BGP)

- First approach, being standardized
- Two **public key infrastructures**
  - Bind IP prefixes to AS numbers
  - Bind AS numbers to organization infrastructure
- Uses public key infrastructure
- Lots of **digital signatures**
  - Costly
- **Route attestations** (signed by each party in the path)

S-BGP Benefits and Costs

- **Benefits**
  - Most comprehensive
  - Origin Authentication
  - Path Authenticity
- **Costs**
  - Storage of attestations
  - A lot of crypto operations

Public Key Infrastructure

- Naming/number authority
  - ICANN, regional authorities
- Certificate containing
  - AS number, public key
  - router DNS name, router ID, AS number, router public key
- Attestations
  - Address binding to AS
  - Route to originating AS

Secure Origin BGP (soBGP)

- Address space attestations (like S-BGP)
- Certificate for pub. key for routers
- New SECURITY message
- Signed local topology (AS and neighbors)
  - Inconsistent updates are dropped
- Avoids signing/verifying each update
  - Instead uses signed topology to filter
- No protection against mid-path alterations

Interdomain Route Validation (IRV)

- Servers in each AS that can attest when needed
  - Not every UPDATE is attested
  - Queries can test a subset of routers
- Keeps complexity out of routers
  - IPSec tunnels? No problem
- Requires functional routing to validate…

Experimental Systems

- Origin Authentication
  - Hash Trees
- Secure Path Vector (SPV)
  - MAC-based approach, one-time signatures
  - Lighter-weight, but vulnerable to collusion
- Pretty Secure BGP (psBGP)
  - Reputation system of other ASes
  - Ownership assertion approach
- MOAS Conflict Detection
- Pretty Good BGP (PGBGP)

Conclusions

- BGP **still not inherently secure**
- But, the **attacks are well-known**
  - And there are a lot of eye watching
- Some filtering helps
- Community of network operators
  - Historical information
  - **Ability to pick up a phone and call**
- **Hard to propose something new**

Appendix: link state

Distance vector is not the only possible approach!

- Another school of thought: **link state**
- In **Distance Vector**, a node sends its knowledge about its state of the network to its neighbors
- In **Link State**, a node sends information about its neighbors to the entire network
  - Example: **Open Shortest Path First** (OSPF). It is an **interior gateway protocol**, so you'll find it inside AS'es but not on the open internet

Link State - Basic Steps

Each node assumed to know state of links to its neighbors

- Step 1: Each node broadcasts its state to all other nodes
- Step 2: Each node locally computes shortest paths to all other nodes from global state

Building Blocks

- **Reliable broadcast mechanism**
  - flooding
- Shortest path tree (SPT) algorithm
  - **Dijkstra's SPT algorithm**

Link State Packets (LSPs)

Periodically, each node creates a Link state packet containing:

- Node ID
- List of neighbors and link cost
- Sequence number
- Time to live (TTL)

Node outputs LSP on all its links

Reliable Flooding

When node i receives LSP from node j:

- If LSP is the most recent LSP from j that i has seen so far, i saves it in database and forwards a copy on all links except link LSP was received on.
- Otherwise, discard LSP.

SPT Algorithm (Dijkstra)

SPT = {*a*}

for all nodes *v*

if *v* adjacent to *a* then *D(v)* = cost (a, v)

else *D(v) = infinity*

Loop

find w not in SPT, where *D(w)* is min

add *w* in SPT

for all *v* adjacent to *w* and not in SPT

D(v) = min (D(v), D(w) + C(w, v))

until all nodes are in SPT

LS v.s. DV

In DV send **everything you know to your neighbors**

In LS **send info about your neighbors to everyone**

- Msg size: small with LS, potentially large with DV
- Msg exchange: LS: O(nE), DV: only to neighbors

LS v.s. DV

Robustness:

- LS can broadcast **incorrect/corrupted LSP**
  - **localized problem**
- DV can **advertise incorrect paths** to all destinations
  - incorrect calculation can **spread to entire network**