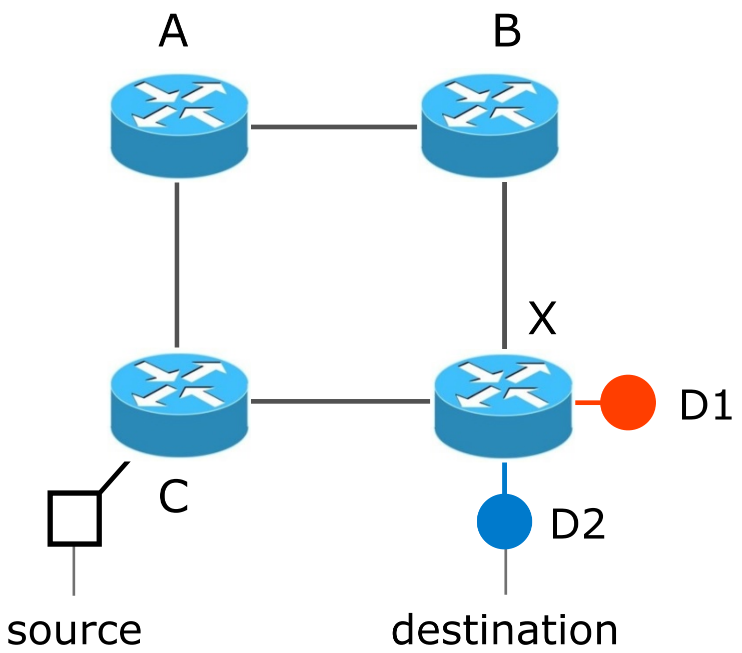
PC Rev 1

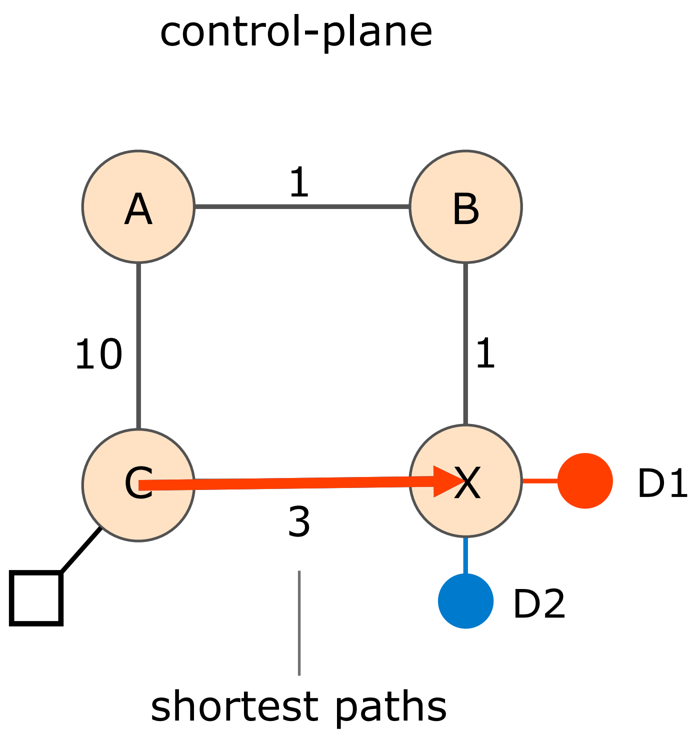
Fibbing, coding, vectoring

April 21, 2016

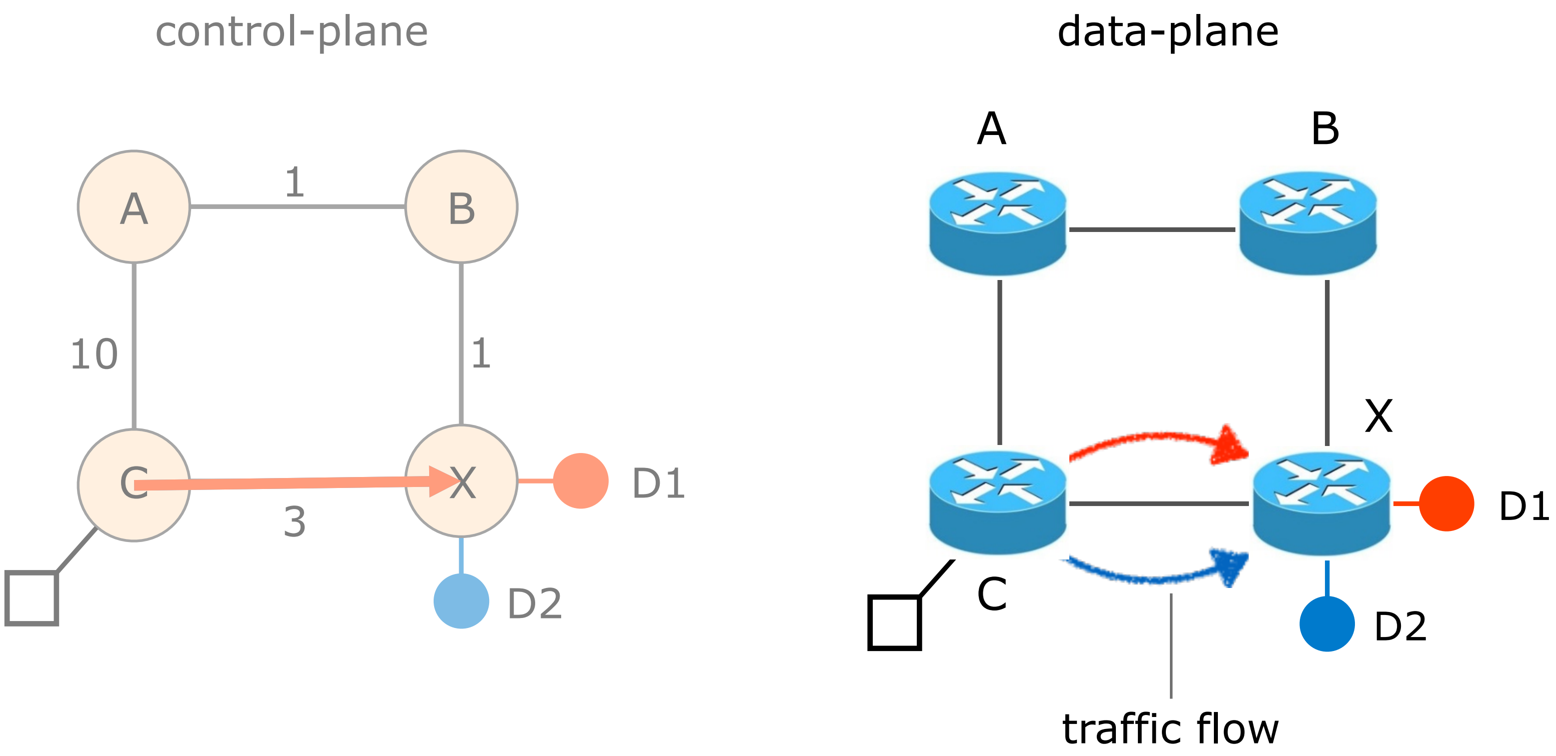
Consider this simple network
(implemented with Cisco routers)



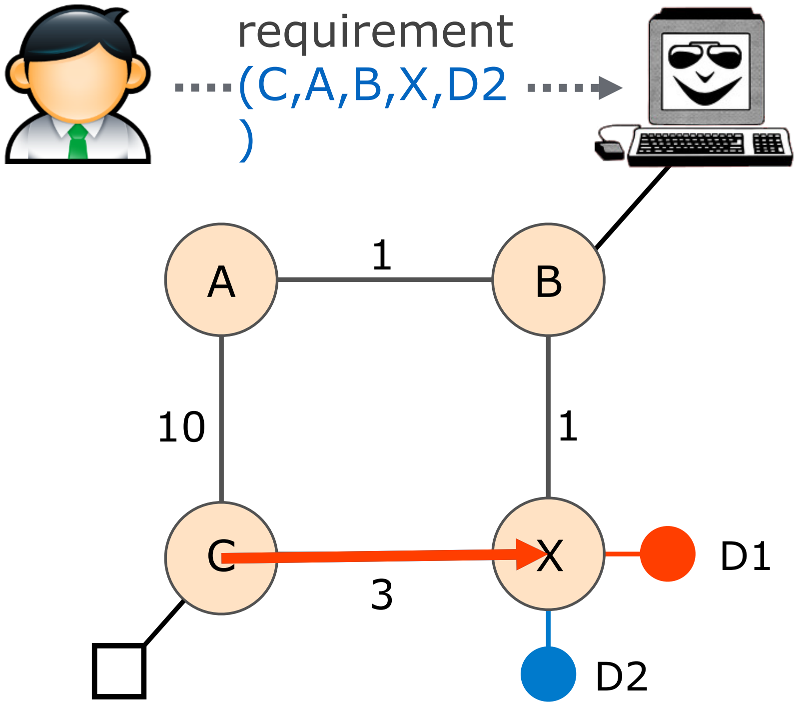
An IGP control-plane computes shortest paths on a shared weighted topology



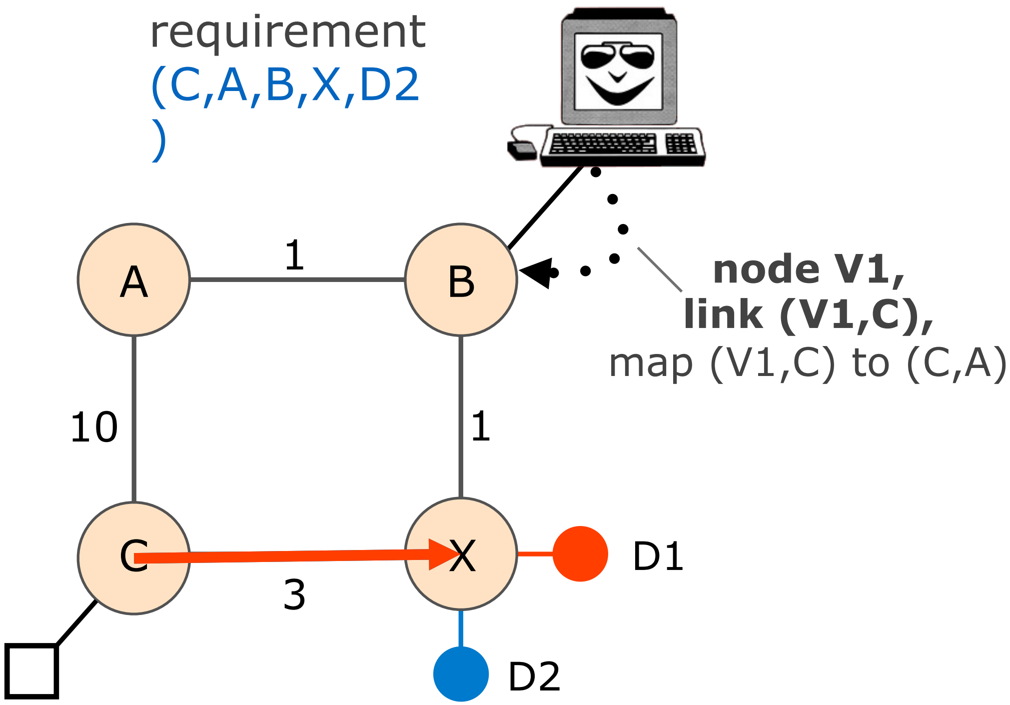
IGP shortest paths are translated into forwarding paths on the data-plane



In Fibbing, operators can ask the controller to modify forwarding paths

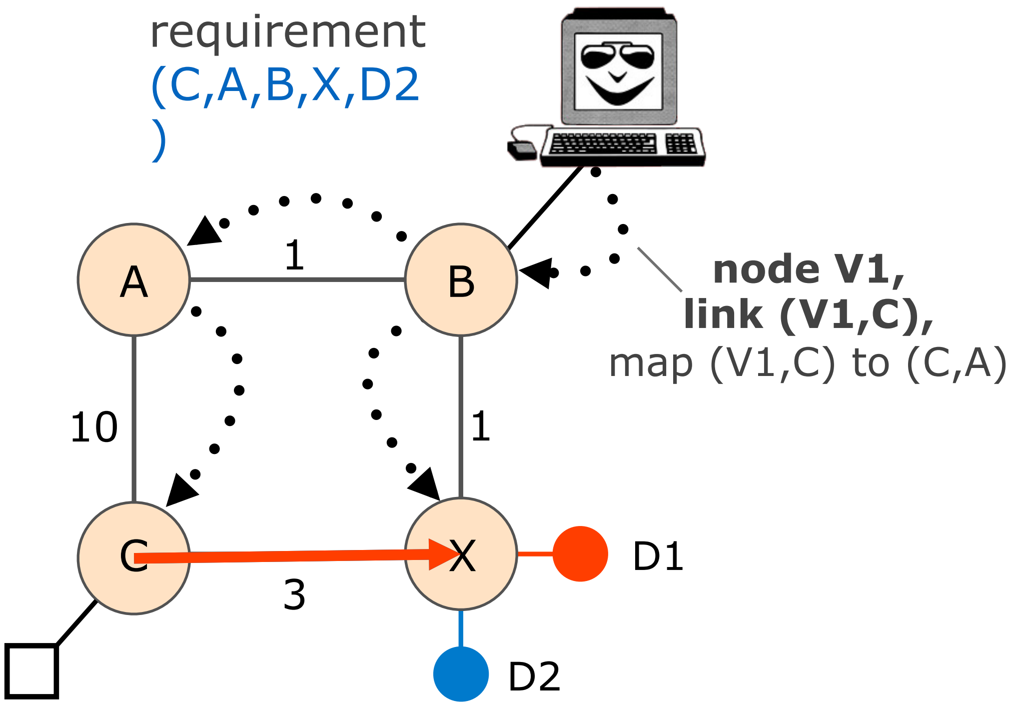


The Fibbing controller injects information on *fake nodes and links* to the IGP control-plane



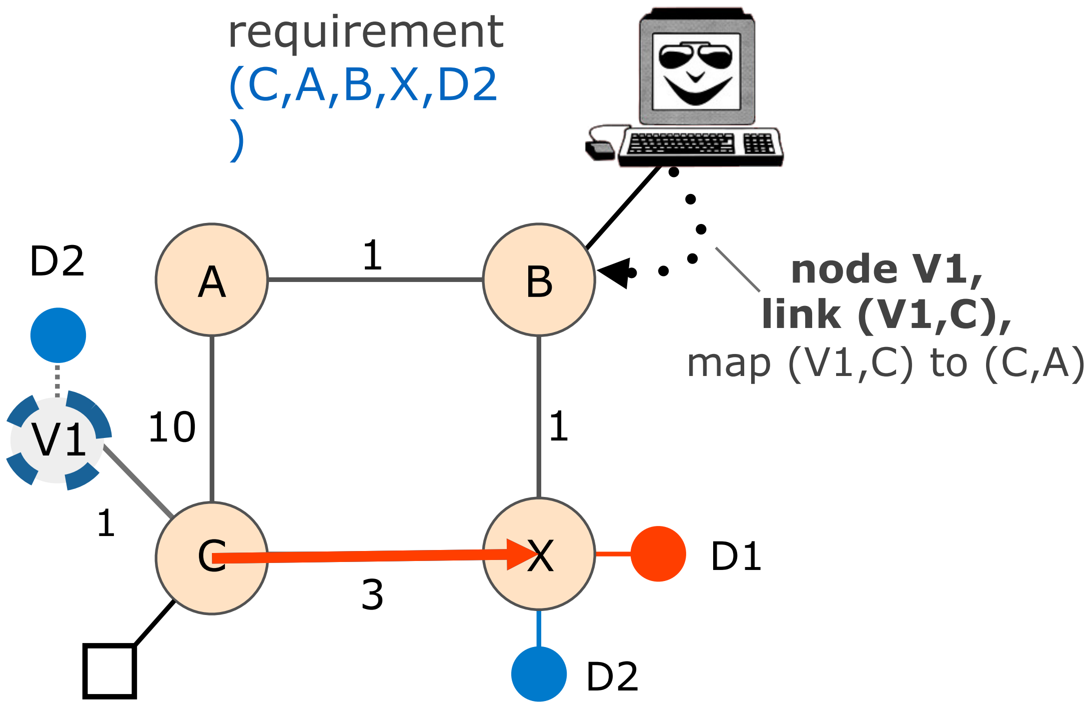
Informations are flooded

to all IGP routers in the network

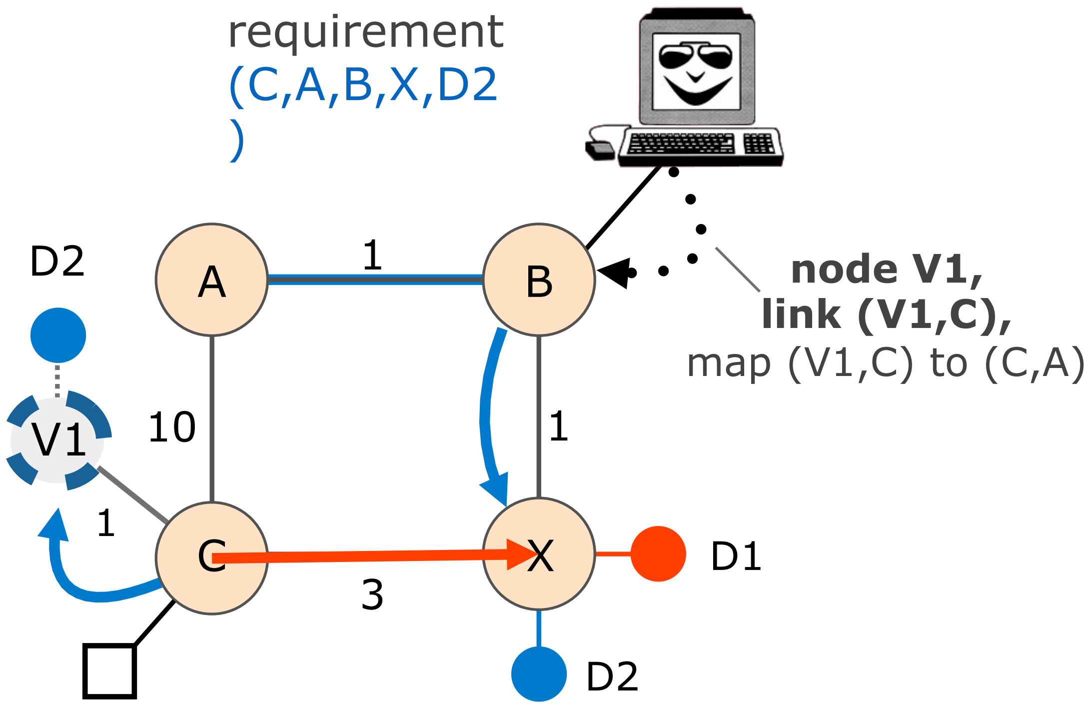


Fibbing messages *augment*

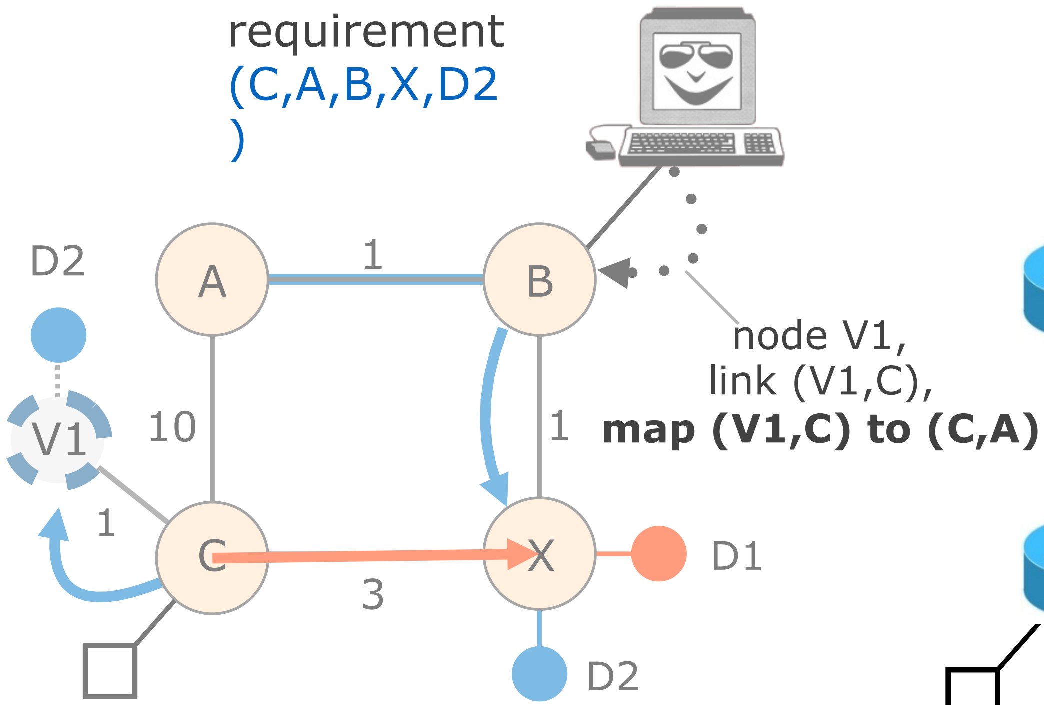
the topology seen by all IGP routers

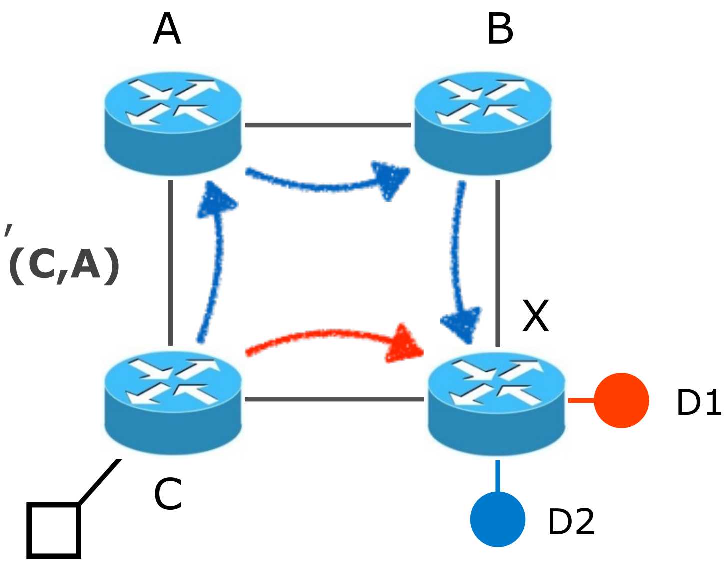


Augmented topologies translate into new control-plane paths



Augmented topologies translate into new *data-plane* paths





Fibbing can program arbitrary per-destination paths

Theorem Any set of forwarding DAGs can be enforced by Fibbing

Fibbing can program arbitrary per-destination paths

Theorem

Any set of forwarding DAGs can be enforced by Fibbing

paths to the same destination do not create loops

By achieving full per-destination control, Fibbing is highly flexible

Theorem Any set of forwarding DAGs can be enforced by Fibbing

- fine-grained traffic steering (middleboxing)
- per-destination load balancing (traffic engineering)
- backup paths provisioning (failure recovery)

Central Control over Distributed Routing

fibbing.net

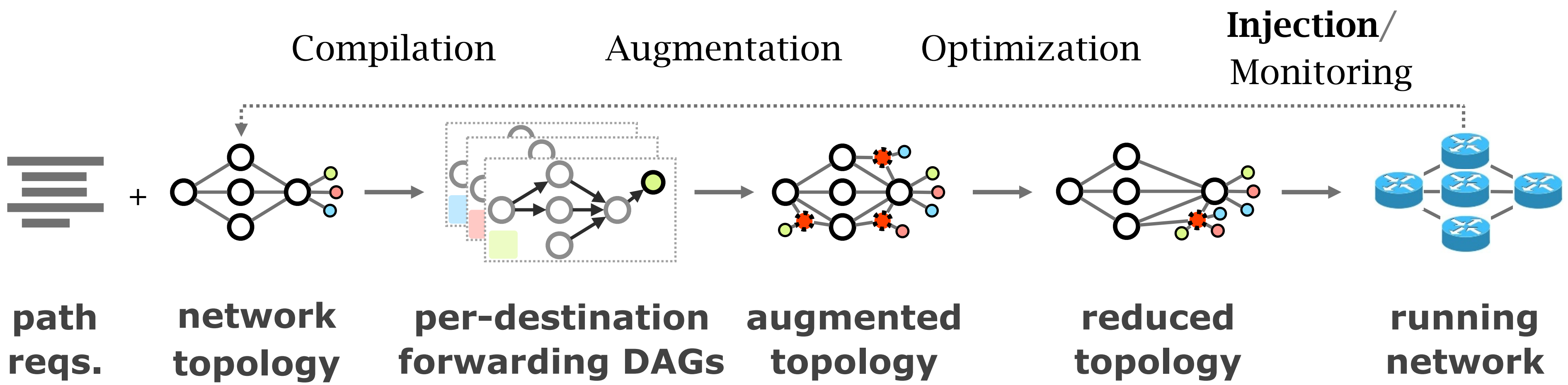
1 Manageability

2 Flexibility

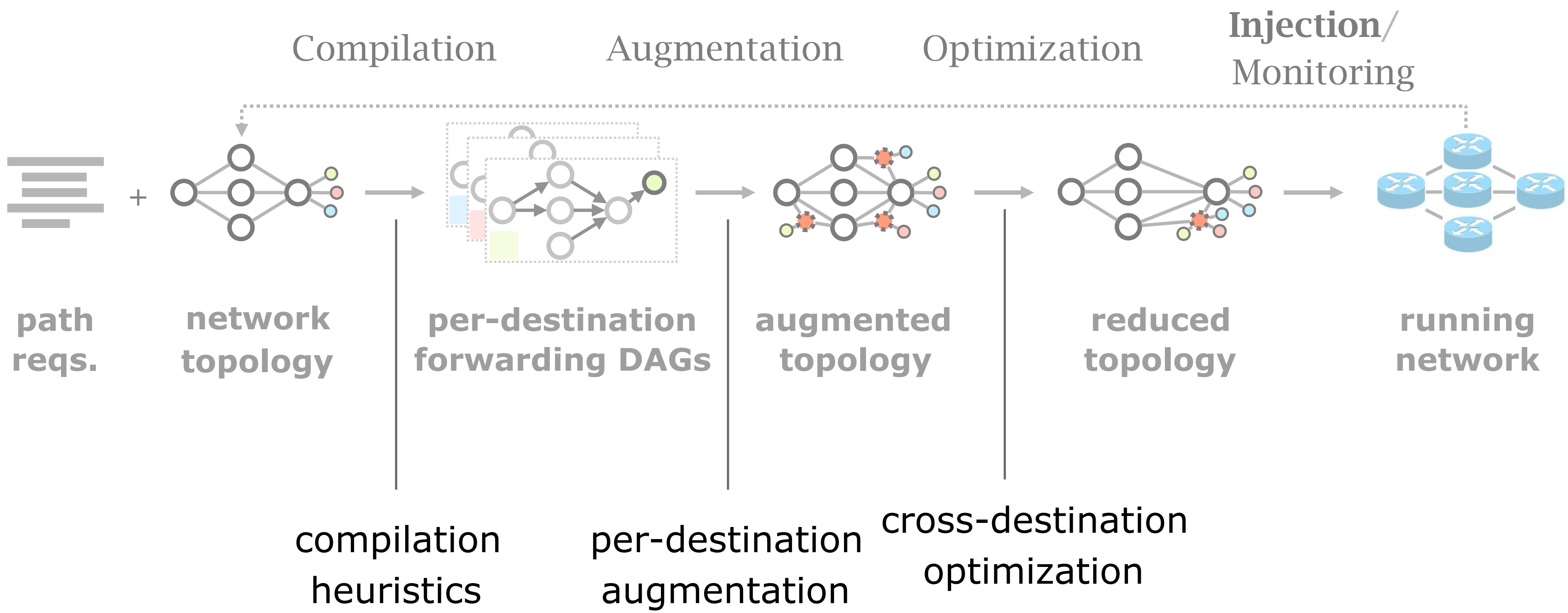
3 Scalability

4 Robustness

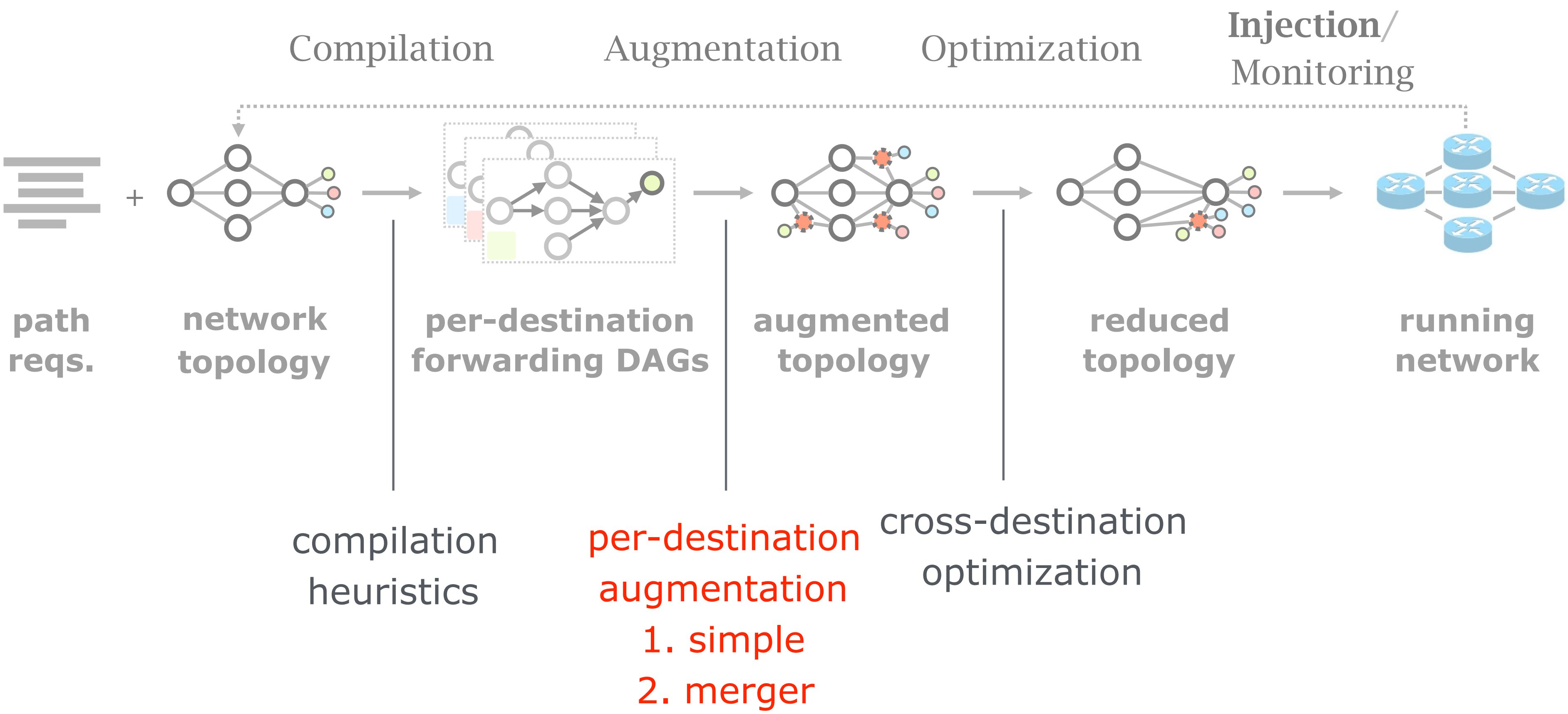
We implemented a Fibbing controller



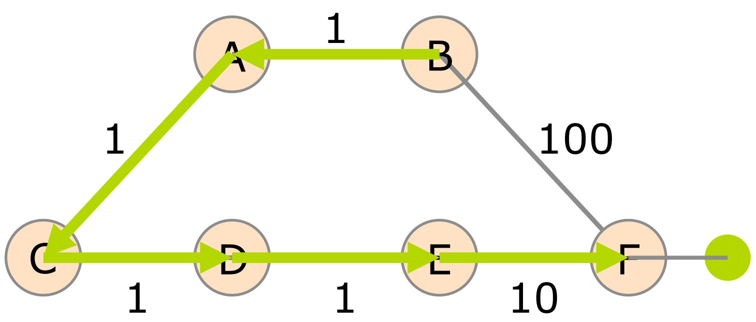
We also propose algorithms to compute augmented topologies of limited size



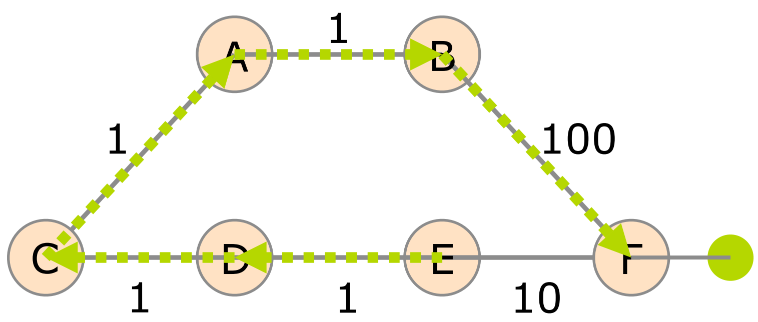
For our Fibbing controller, we propose algorithms to be run in sequence



Consider the following example,
with a drastic forwarding path change

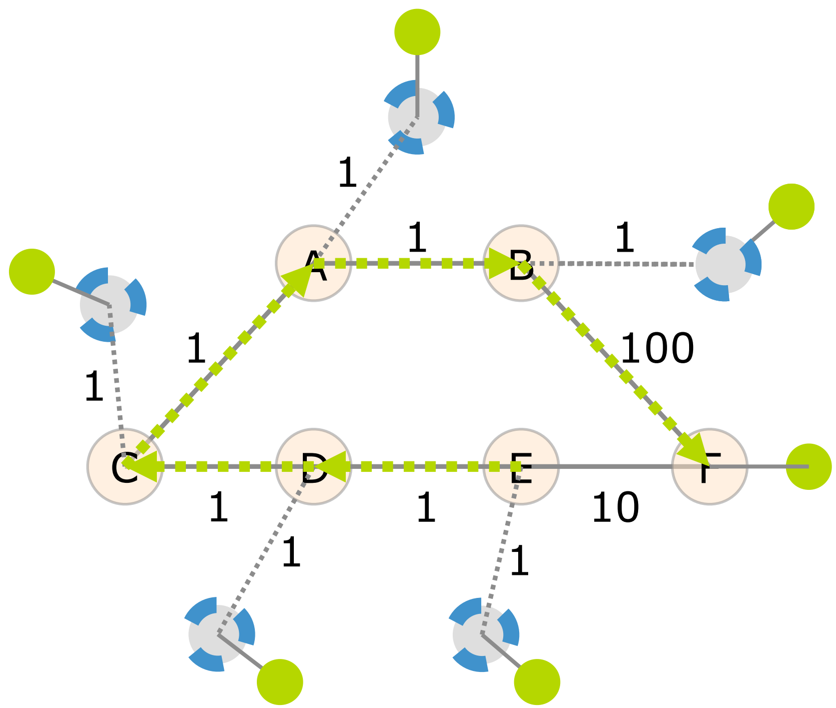


original shortest-path
"down and to the right"



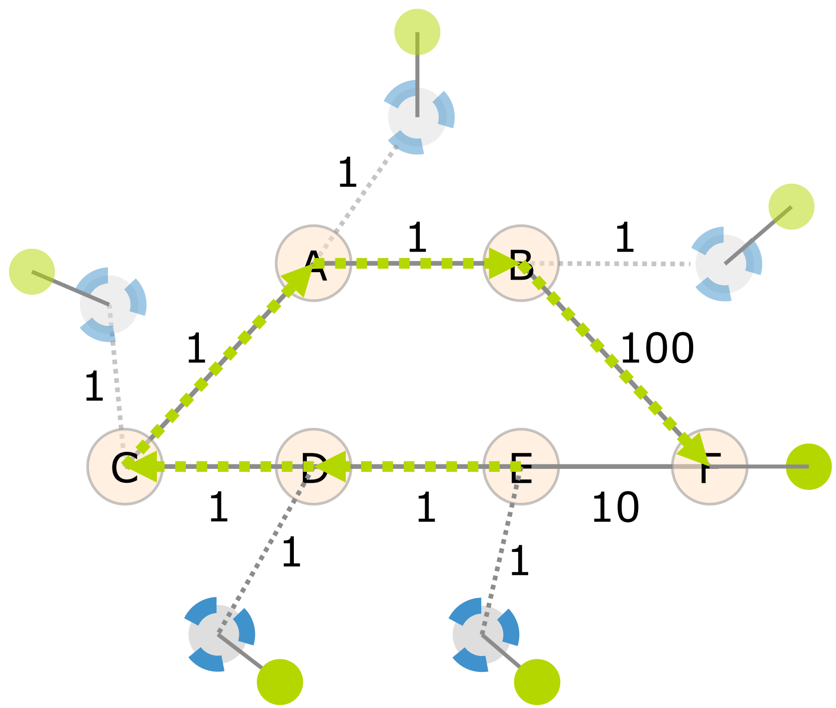
desired shortest-path
"up and to the right"

Simple adds one fake node for every router that has to change next-hop



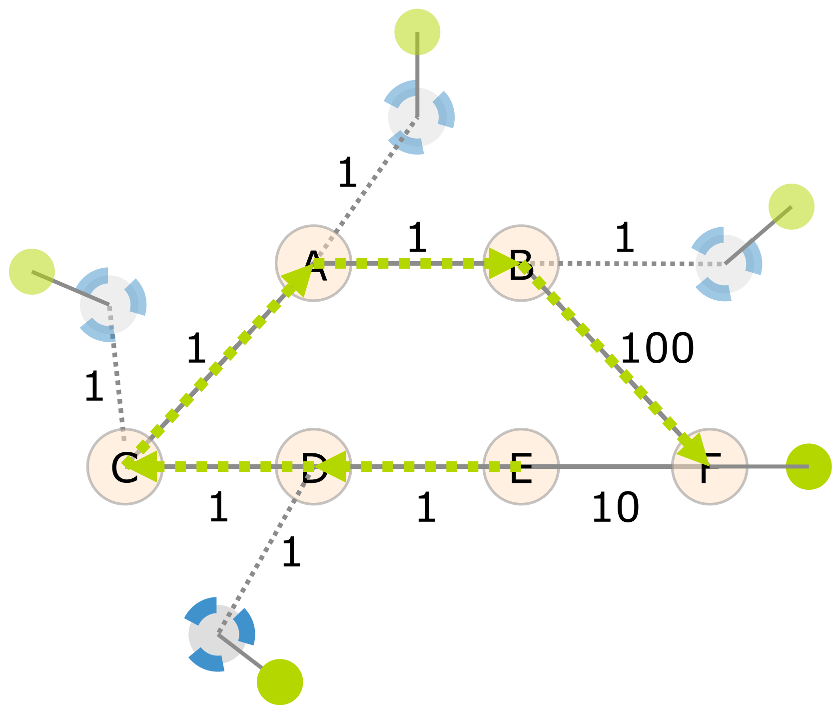
Merger iteratively merges fake nodes

(starting from Simple's output)

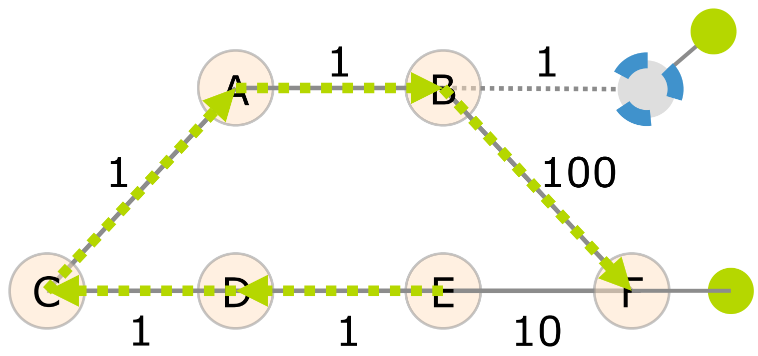


Merger iteratively merges fake nodes

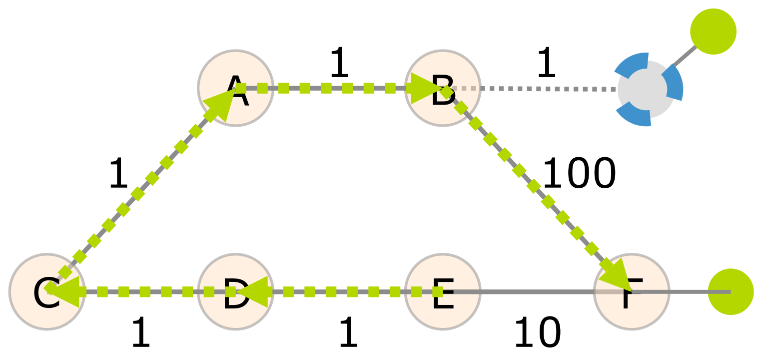
(starting from Simple's output)



This way, Merger programs multiple next-hop changes with a single fake node



This way, Merger programs multiple next-hop changes with a single fake node



Previous SDN solutions (e.g., RCP) cannot do the same

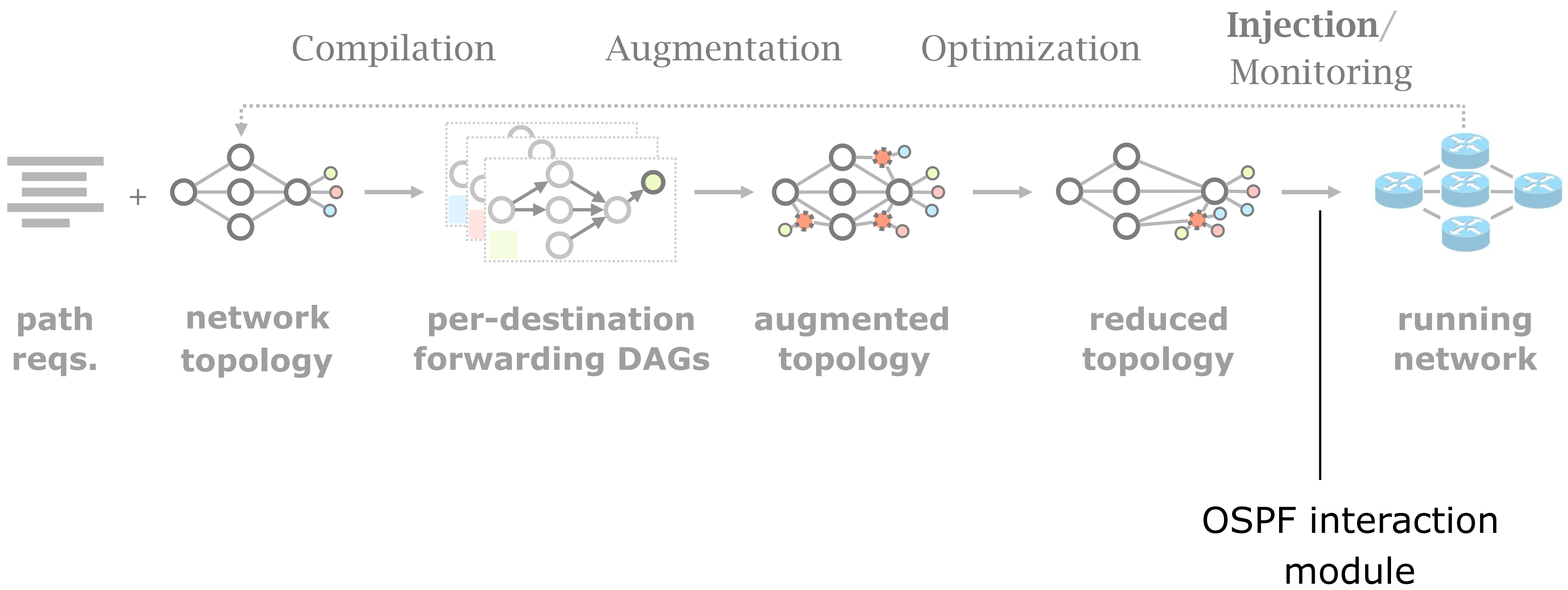
Simple and Merger achieve different trade-offs in terms of time and optimization efficiency

We ran experiments on Rocketfuel topologies,
with at least 25% of nodes changing next-hops

- Simple runs in milliseconds
  Merger takes 0.1 seconds

- Merger reduces fake nodes by up to 50%
  and up to 90% with cross-destination optimization

We implemented the machinery to listen to OSPF and augment the topology



Experiments on real routers show that
Fibbing has very limited impact on routers

# fake nodes	router memory (MB)	
1 000	0.7	
5 000	6.8	
10 000	14.5	
50 000	76.0	
100 000	153	DRAM is cheap

>> # real routers

Experiments on real routers show that Fibbing has very limited impact on routers

# fake nodes	router memory (MB)	
1 000	0.7	
5 000	6.8	
10 000	14.5	
50 000	76.0	
100 000	153	DRAM is cheap

CPU utilization always under 4%

Experiments on real routers show that Fibbing does not impact IGP convergence

Upon link failure, we registered *no difference* in the (sub-second) IGP convergence with

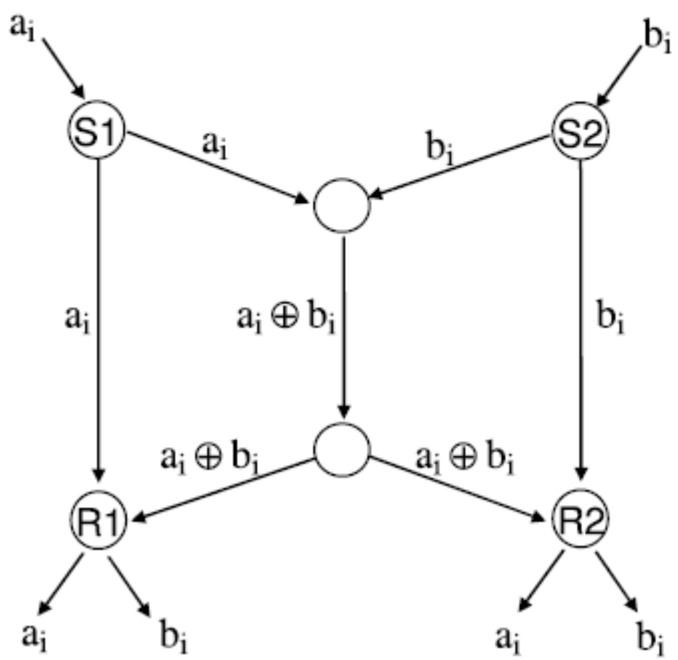
- no fake nodes
- up to 100,000 fake nodes and destinations

Experiments on real routers show that Fibbing achieves fast forwarding changes

# fake nodes	installation time (seconds)	
1 000	0.9	
5 000	4.5	
10 000	8.9	
50 000	44.7	
100 000	89.50	894.50 μs/entry

Network Coding – Background

- Ahlswede et al. – *Butterfly Example* in "Network Information Flow", IEEE Transactions on Information Theory, 2000



Allowing routers to mix the bits in forwarding messages can increase network throughput
(Achieves multicast capacity)

Chronology of Research

- Li et al. – Showed that linear codes are sufficient to achieve maximum capacity bounds (*2003*)
- Koetter and Medard – Polynomial time algorithms for encoding and decoding (*2003*)
- Ho et al. – Extended previous results to a randomized setting (*2003*)
- Studies on wireless network coding began in *2003* as well! (Shows that it was a high interest research area)
- More work on wireless network coding with multicast models (2004)
- Lun et al. – Problem of minimizing communication cost in wireless networks can be formulated linearly (*2005*) – Used multicast model as well!

  So all the previous work was theoretical and assumes multicast traffic.
- Authors introduced the idea of opportunistic coding for wireless environments in 2005

Why is it different?

<u>*They address the common case of unicast traffic, bursty flows and other practical issues.*</u>

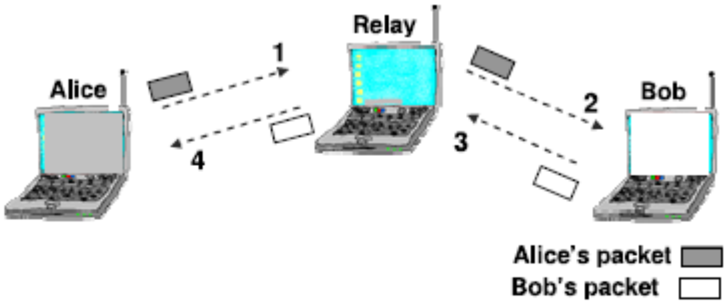
Current Paper

- Explores the utility of network coding in improving the throughput of wireless networks.
- Authors extend the theory of their opportunistic coding architecture (COPE) by application in a practical scenario.
- Presents the first system architecture for wireless network coding.
- Implements the design, creating the first deployment of network coding in a wireless network.
- Studies the performance of COPE.

COPE

- What does being opportunistic mean?

  Each node relies on local information to detect and exploit coding opportunities when they arise, so as to maximize throughput.
- COPE inserts an *opportunistic* coding shim between the IP and MAC layers.
- Enables forwarding of multiple packets in a single transmission.
- Based on the fact that intelligently mixing packets increases network throughput.



(a) Current Approach

(b) COPE

Design Principles:

- *COPE embraces the broadcast nature of the wireless channel.*
- *COPE employs network coding.*

Inside COPE

Term	Definition
Native Packet	A non-encoded packet
Encoded or XOR-ed Packet	A packet that is the XOR of multiple native packets
Nexthops of an Encoded Packet	The set of nexthops for the native packets XOR-ed to generate the encoded packet
Packet Id	A 32-bit hash of the packet's IP source address and IP sequence number
Output Queue	A FIFO queue at each node, where it keeps the packets it needs to forward
Packet Pool	A buffer where a node stores all packets heard in the past T seconds
Coding Gain	The ratio of the number of transmissions required by the current non-coding approach, to the number of transmissions used by COPE to deliver the same set of packets.
Coding+MAC Gain	The expected throughput gain with COPE when an 802.11 MAC is used, and all nodes are backlogged.

COPE incorporates three main techniques:

- Opportunistic Listening
- Opportunistic Coding
- Learning Neighbor State

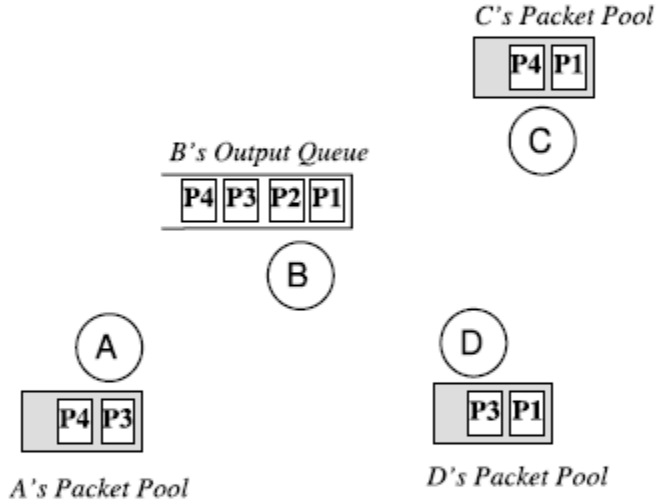
Opportunistic Listening

- Nodes are equipped with omni-directional antennae
- COPE sets the nodes to a promiscuous mode.
- The nodes store the overheard packets for a limited period T (0.5 s)
- Each node also broadcasts *reception reports* to tell it’s neighbors which packets it has stored.

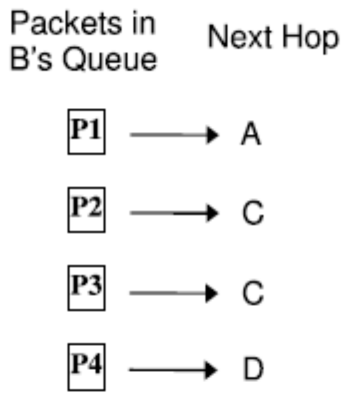
Opportunistic Coding

Rule:

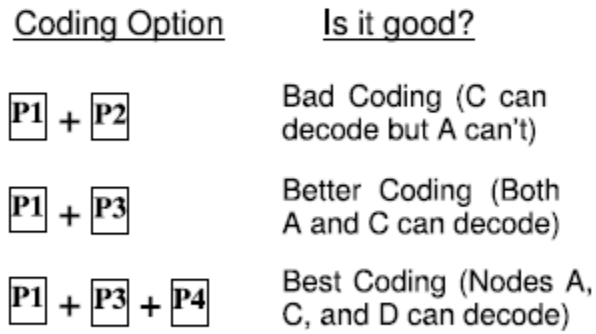
"A node should aim to maximize the number of native packets delivered in a single transmission, while ensuring that each intended next-hop has enough information to decode it's native packet."



(a) B can code packets it wants to send



(b) Nexthops of packets in B's queue



(c) Possible coding options

Issues:

- Unneeded data should not be forwarded to areas where there is no interested receiver, wasting capacity.
- The coding algorithm should ensure that all next-hops of an encoded packet can decode their corresponding native packets.

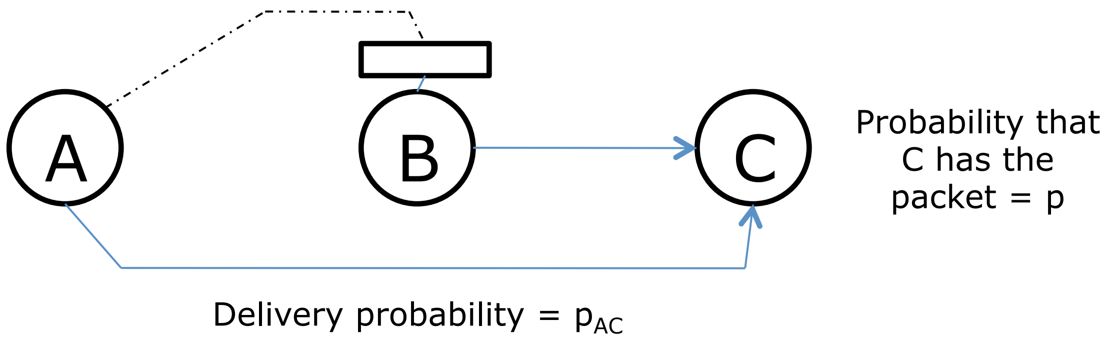
<u>Rule:</u> To transmit n packets $p_1 \ldots p_n$ to n next-hops $r_1 \ldots r_n$, a node can XOR the n packets together only if each next-hop r_i has all n - 1 packets p_j for $j \neq i$

Learning Neighbor State

- A node cannot solely rely on reception reports, and may need to guess whether a neighbor has a particular packet.
- To guess intelligently, we can leverage routing computations.

  The ETX metric computes the delivery probability between nodes and assigns each link a weight of 1/*(delivery_probability)*
- In the absence of deterministic information,

  COPE estimates the probability that a particular neighbor has a packet, as the delivery probability of the link between the packet's previous hop and the neighbor.

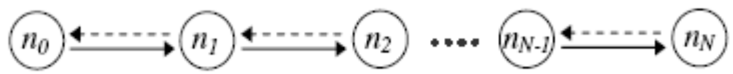


"p increases with p_{AC}"

Understanding COPE's Gains

Coding Gain

- Defined as the ratio of no. of transmissions required without COPE to the no. of transmissions used by COPE to deliver the same set of packets.
- By definition, this number is greater than 1. (4/3 for Alice-Bob Example)
- *Theorem: In the absence of opportunistic listening, COPE's maximum coding gain is 2, and it is achievable.*

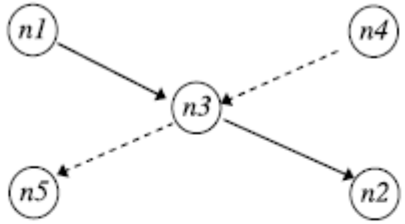


Chain topology; 2 flows in reverse directions.

Coding Gain achievable = 2N/(N+1)

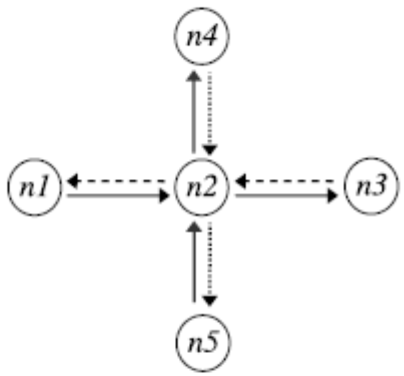
This value tends to 2 as

<u>In the presence of opportunistic listening</u>



“X” topology
2 flows intersecting at n_2.

Achievable Coding Gain = 1.33



Cross topology
4 flows intersecting at n_2

Achievable Coding Gain = 1.6

Understanding COPE’s Gains

Coding + MAC Gain

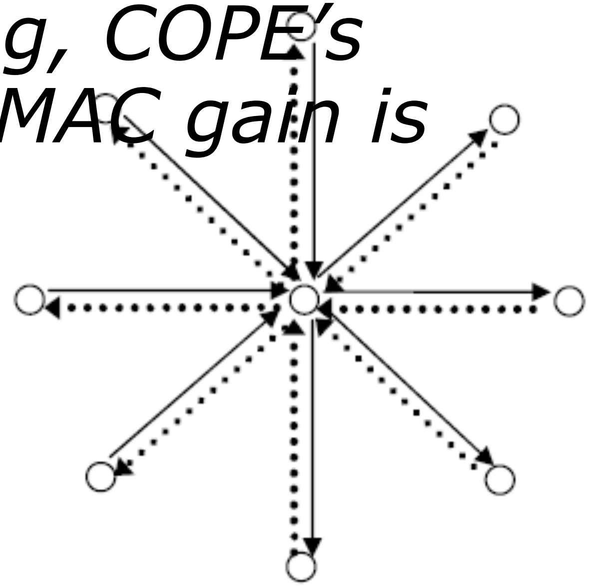
- It was observed that throughput improvement using COPE greatly exceeded the coding gain.
- Since it tries to be fair, the MAC layer divides the bandwidth equally between contending nodes.
- COPE allows the bottleneck nodes to XOR pairs of packets and drain them quicker, increasing the throughput of the network.
- For topologies with a single bottleneck, the Coding + MAC Gain is the ratio if the bottleneck’s draining rate with COPE to it’s draining rate without COPE.

- *<u>Theorem:</u> In the absence of opportunistic listening, COPE's maximum Coding + MAC gain is 2, and it is achievable.*

  Node can XOR at most 2 packets together, and the bottleneck can drain at almost twice as fast, bounding the Coding + MAC Gain at 2.

- *<u>Theorem:</u> In the presence of opportunistic listening, COPE's maximum Coding + MAC gain is unbounded.*

For N edge nodes, the bottleneck node XORs N packets together, and the queue drains N times faster.



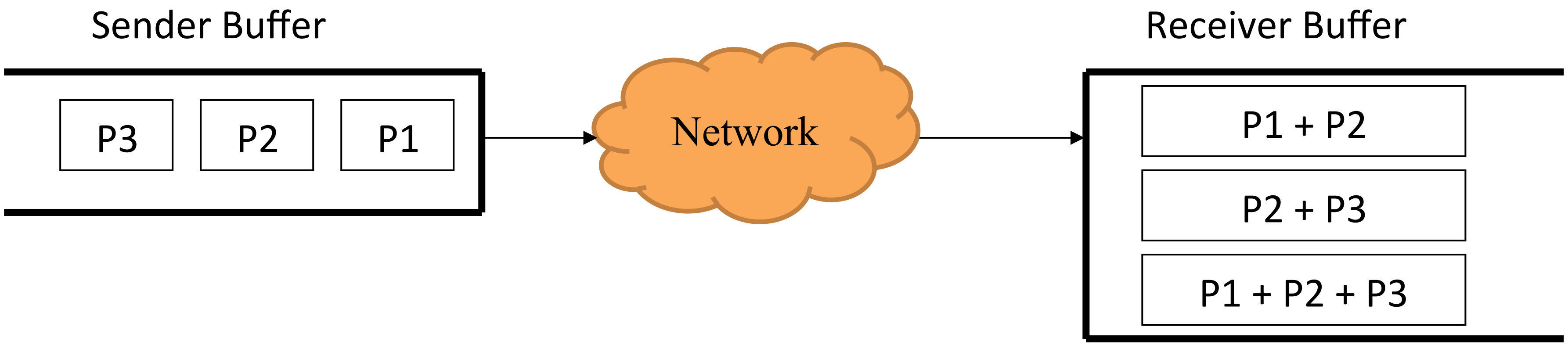
Wheel topology; many flows intersecting at the center node.

- Theoretical gains:

Topology	Coding Gain	Coding+MAC Gain
Alice-and-Bob	1.33	2
"X"	1.33	2
Cross	1.6	4
Infinite Chain	2	2
Infinite Wheel	2	∞

- Important to note that:
  - The gains in practice tend to be lower due to non-availability of coding opportunities, packet header overheads, medium losses, etc.,
  - But COPE does increase actual information rate of the medium far above the bit rate.

The Problem



Can’t acknowledge a packet until you can decode.
Usually, decoding requires a number of packets.
Code / acknowledge over small blocks to avoid delay, manage complexity.

Compare to ARQ

Context: Reliable communication over a (wireless) network of packet erasure channels

ARQ

- Retransmit lost packets
- Low delay, queue size
- Streaming, not blocks
- Not efficient on broadcast links
- Link-by-link ARQ does not achieve network multicast capacity.

Network Coding

- Transmit linear combinations of packets
- Achieves min-cut multicast capacity
- Extends to broadcast links
- Congestion control requires feedback
- Decoding delay: block-based

Goals

- Devise a system that behaves as close to TCP as possible, while masking non-congestion wireless losses from congestion control where possible.
  - Standard TCP/wireless problem.
- Stream-based, not block-based.
- Low delay.
- Focus on wireless setting.
  - Where network coding can offer biggest benefits.
  - Not necessarily a universal solution.

Main Idea : Coding ACKs

- What does it mean to "see" a packet?
- Standard notion: we have a copy of the packet.
  - Doesn't work well in coding setting.
  - Implies must decode to see a packet.
- New definition: we have a packet that will allow us to decode *once enough* useful packets arrive.
  - Packet is useful if linearly independent.
  - When enough useful packets arrive can decode.

Coding ACKs

- For a message of size *n*, need *n* useful packets.
- Each coded packet corresponds to a degree of freedom.
- *Instead of acknowledging individual packets, acknowledge newly arrived degrees of freedom.*

Coding ACKs

Original message : $p_1, p_2, p_3 \ldots$

Coded Packets

$4p_1 + 2p_2 + 5p_3$

c_1	4 2 5 0 0 0 0
c_2	3 1 2 5 0 0 0
c_3	1 2 3 4 1 0 0
c_4	3 3 1 2 1 0 0
c_5	1 2 5 4 5 0 0

$$\begin{pmatrix} 4 & 2 & 5 & 0 & 0 & 0 & 0 \\ 3 & 1 & 2 & 5 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 1 & 0 & 0 \\ 3 & 3 & 1 & 2 & 1 & 0 & 0 \\ 1 & 2 & 5 & 4 & 5 & 0 & 0 \end{pmatrix}$$

Coding ACKs

Original message : $p_1, p_2, p_3\ldots$

Coded Packets

$4p_1 + 2p_2 + 5p_3$

c_1	4 2 5 0 0 0 0
c_2	3 1 2 5 0 0 0
c_3	1 2 3 4 1 0 0
c_4	3 3 1 2 1 0 0
c_5	1 2 5 4 5 0 0

$$\begin{pmatrix} 4 & 2 & 5 & 0 & 0 & 0 & 0 \\ 3 & 1 & 2 & 5 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 1 & 0 & 0 \\ 3 & 3 & 1 & 2 & 1 & 0 & 0 \\ 1 & 2 & 5 & 4 & 5 & 0 & 0 \end{pmatrix}$$

When c_1 comes in, you've "seen" packet 1; eventually you'll be able to decode it. And so on…

Coding ACKs

Original message : $p_1, p_2, p_3 \ldots$

Coded Packets

$4p_1 + 2p_2 + 5p_3$

c_1	4 2 5 0 0 0 0
c_2	3 1 2 5 0 0 0
c_3	1 2 3 4 1 0 0
c_4	3 3 1 2 1 0 0
c_5	1 2 5 4 5 0 0

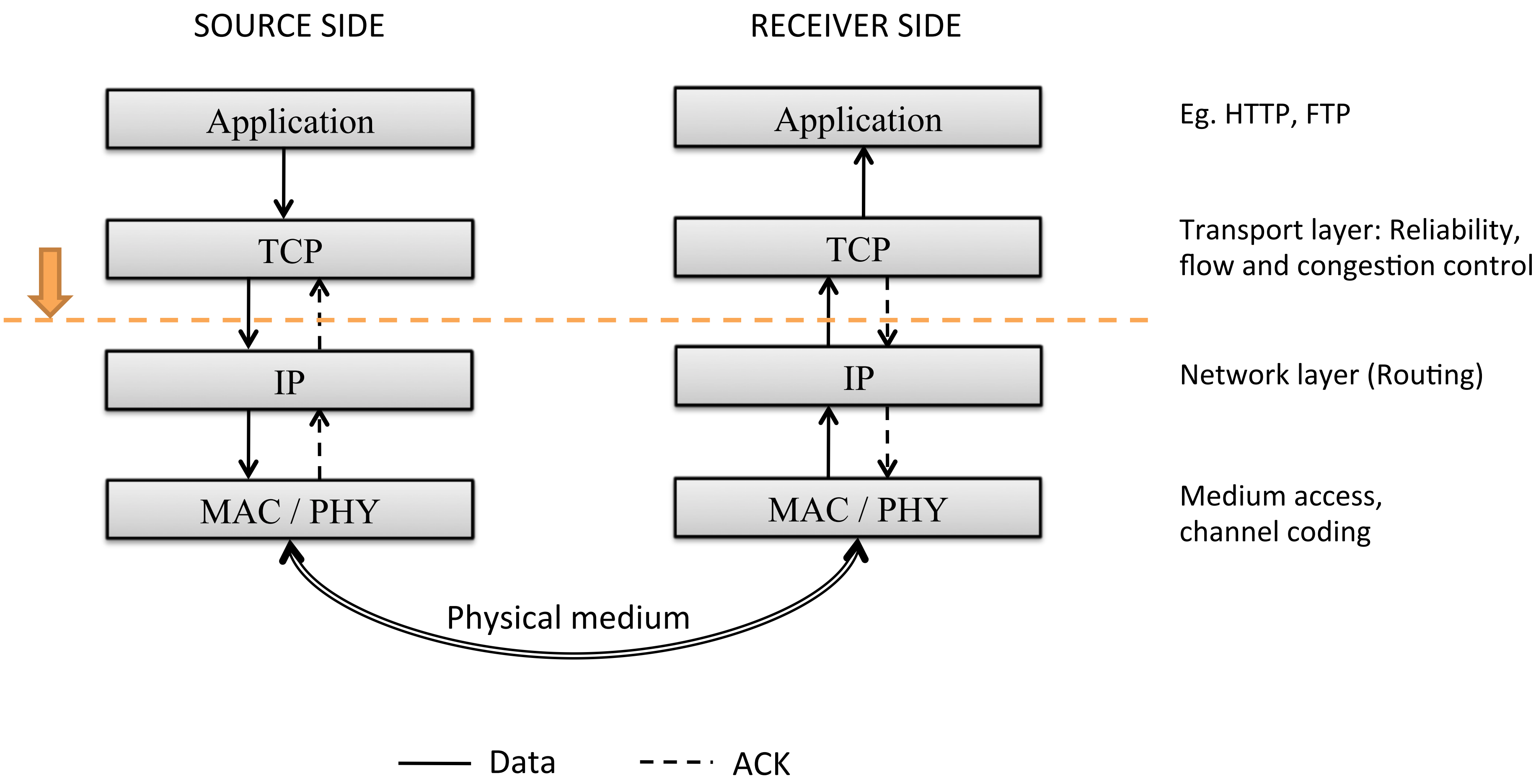
$$\begin{pmatrix} 1 & 4 & 5 & 3 & 0 & 0 & 0 \\ 0 & 1 & 3 & 2 & 6 & 0 & 0 \\ 0 & 0 & 1 & 6 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Use Gaussian elimination as packets arrive to check for a new seen packet.

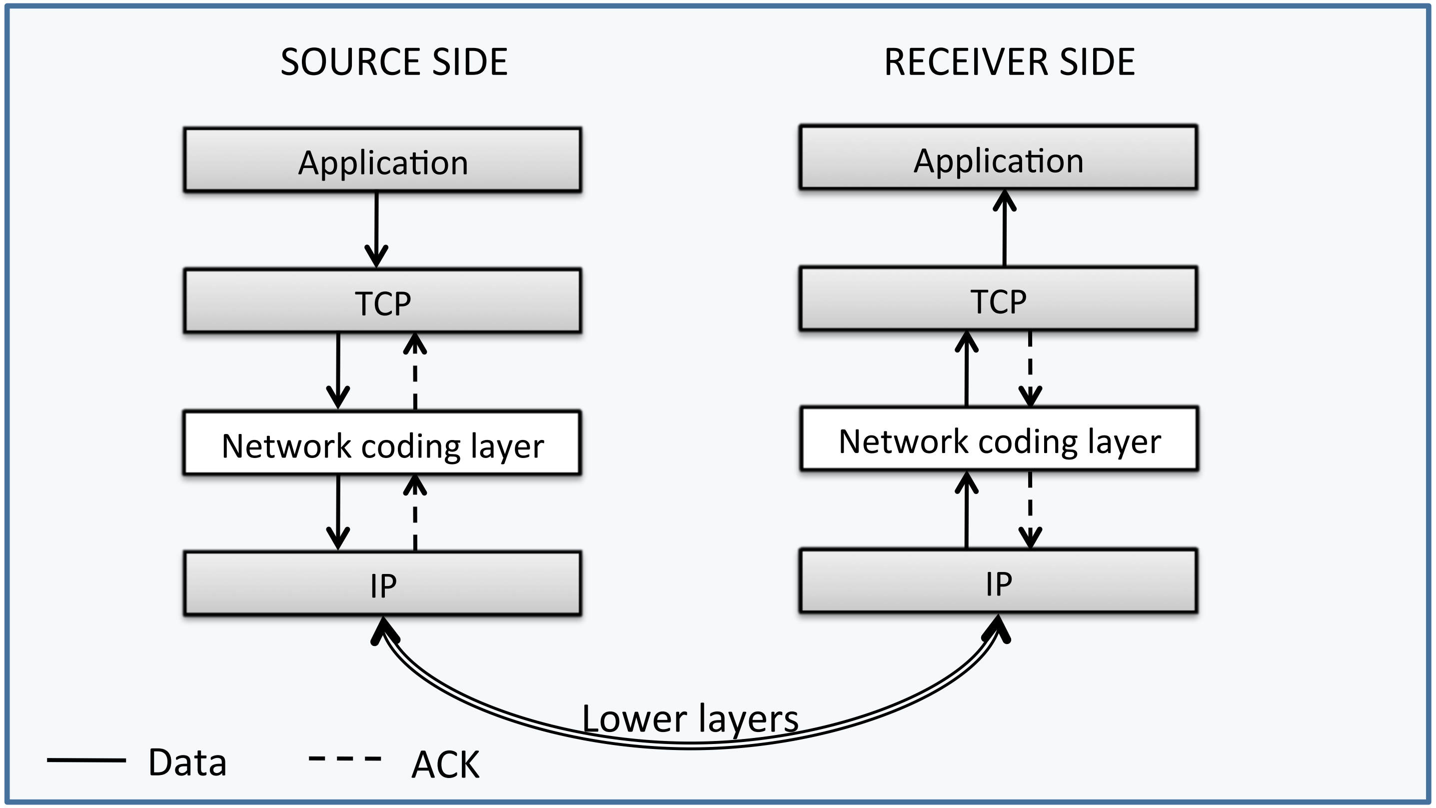
Formal Definition

- A node has *seen* a packet p_k if it can compute a linear combination p_k+q where q is a linear combination of packets with index larger than k.
- When all packets have been seen, decoding is possible.

Layered Architecture



TCP using Network Coding



The Sender Module

- Buffers packets in the current window from the TCP source, sends linear combinations.
- Need for redundancy factor *R.*
  - Sending rate should account for loss rate.
  - Send a constant factor more packets.
  - Open issue : determine *R* dynamically?

Redundancy

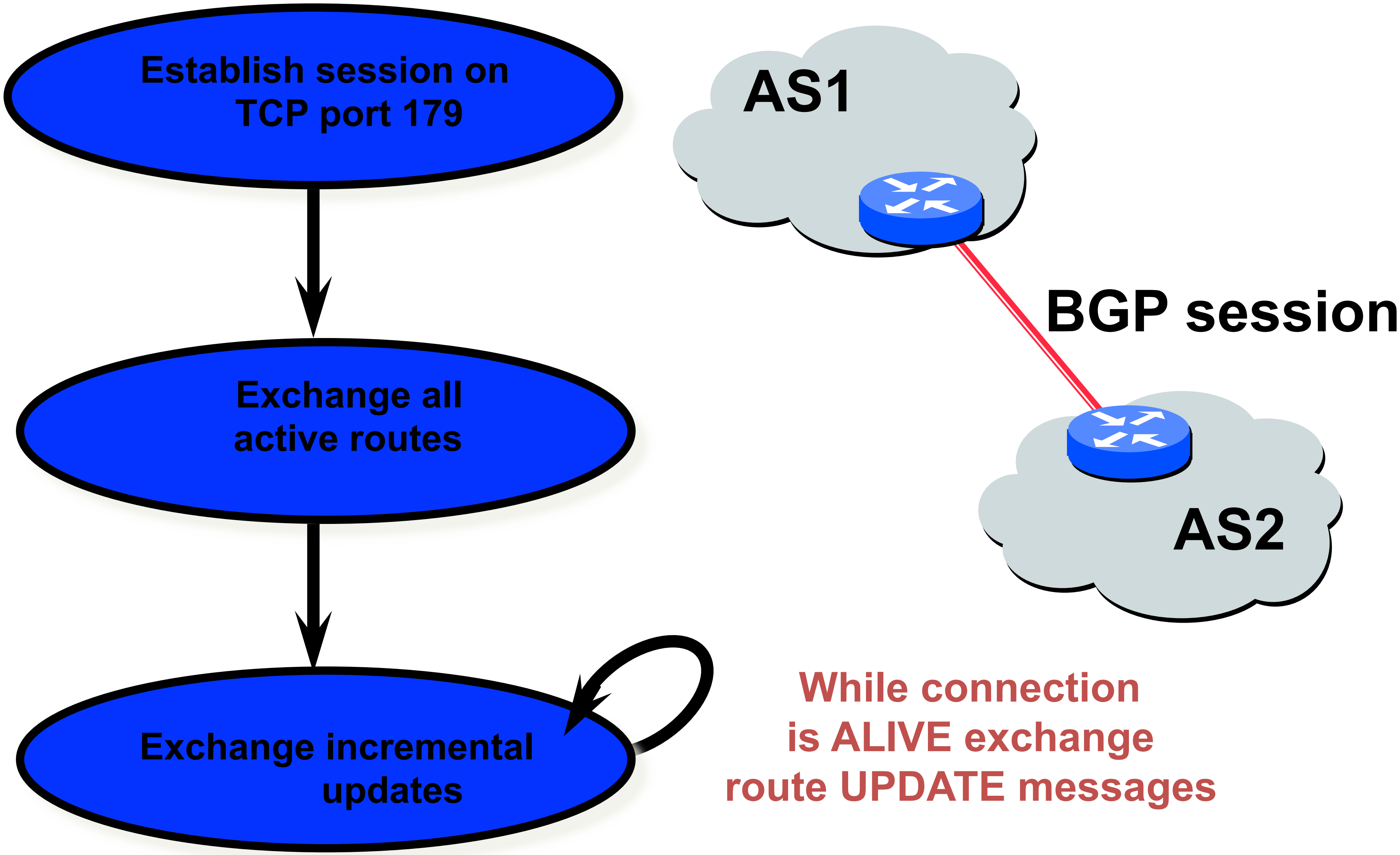
- Too low R
  - TCP times out and backs off drastically.
- Too high R
  - Losses recovered – TCP window advances smoothly.
  - Throughput reduced due to low code rate.
  - Congestion increases.
- Right R is $1/(1-p)$, where p is the loss rate.

BGP-4

- **BGP** = **B**order **G**ateway **P**rotocol
- Is a **Policy-Based** routing protocol
- Is the **de facto EGP** of today's global Internet
- Relatively simple protocol, but configuration is complex and the entire world can see, and be impacted by, your mistakes.

- **1989 : BGP-1 [RFC 1105]**
  - **Replacement for EGP (1984, RFC 904)**
- **1990 : BGP-2 [RFC 1163]**
- **1991 : BGP-3 [RFC 1267]**
- **1995 : BGP-4 [RFC 1771]**
  - **Support for Classless Interdomain Routing (CIDR)**

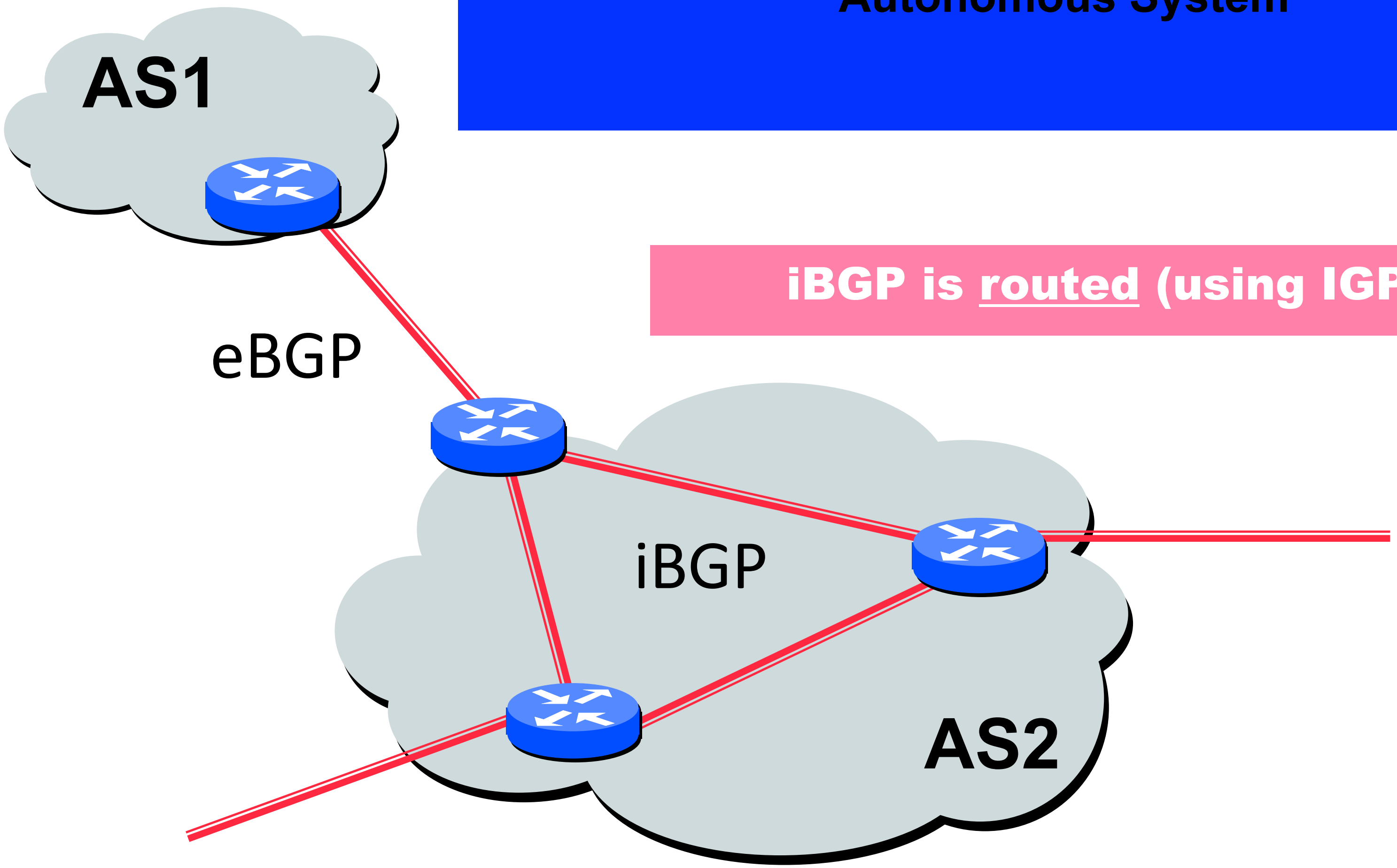
BGP Operations (Simplified)



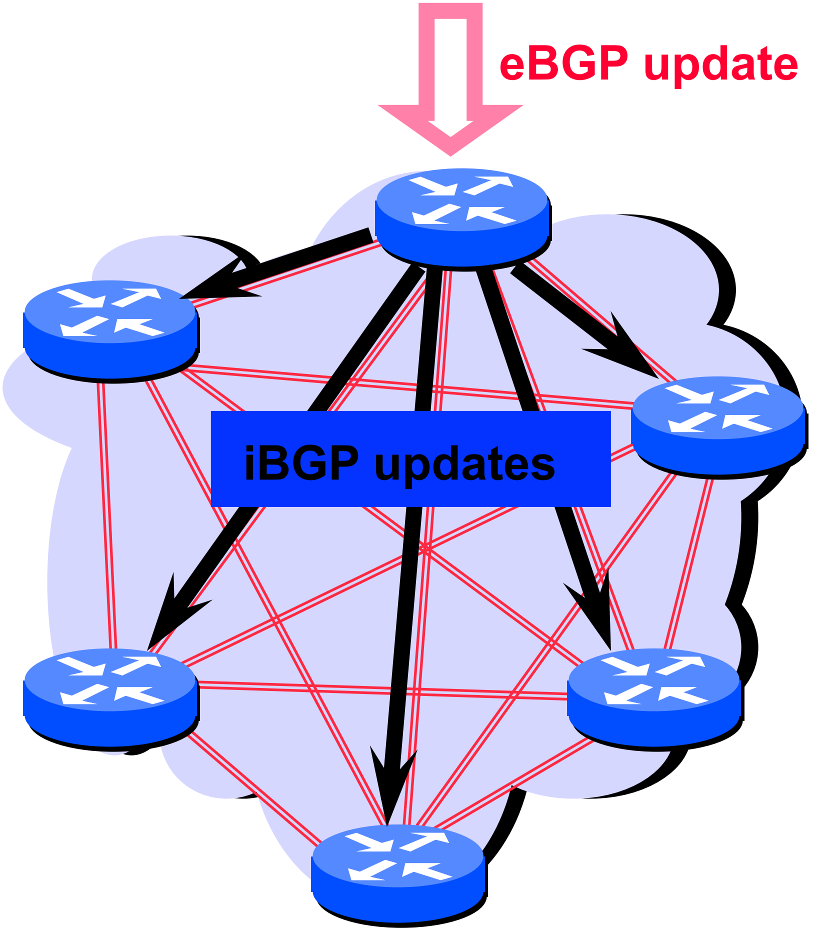
Two Types of BGP Neighbor Relationships

- **External Neighbor (eBGP) in a different Autonomous Systems**
- **Internal Neighbor (iBGP) in the same Autonomous System**

iBGP is routed (using IGP!)



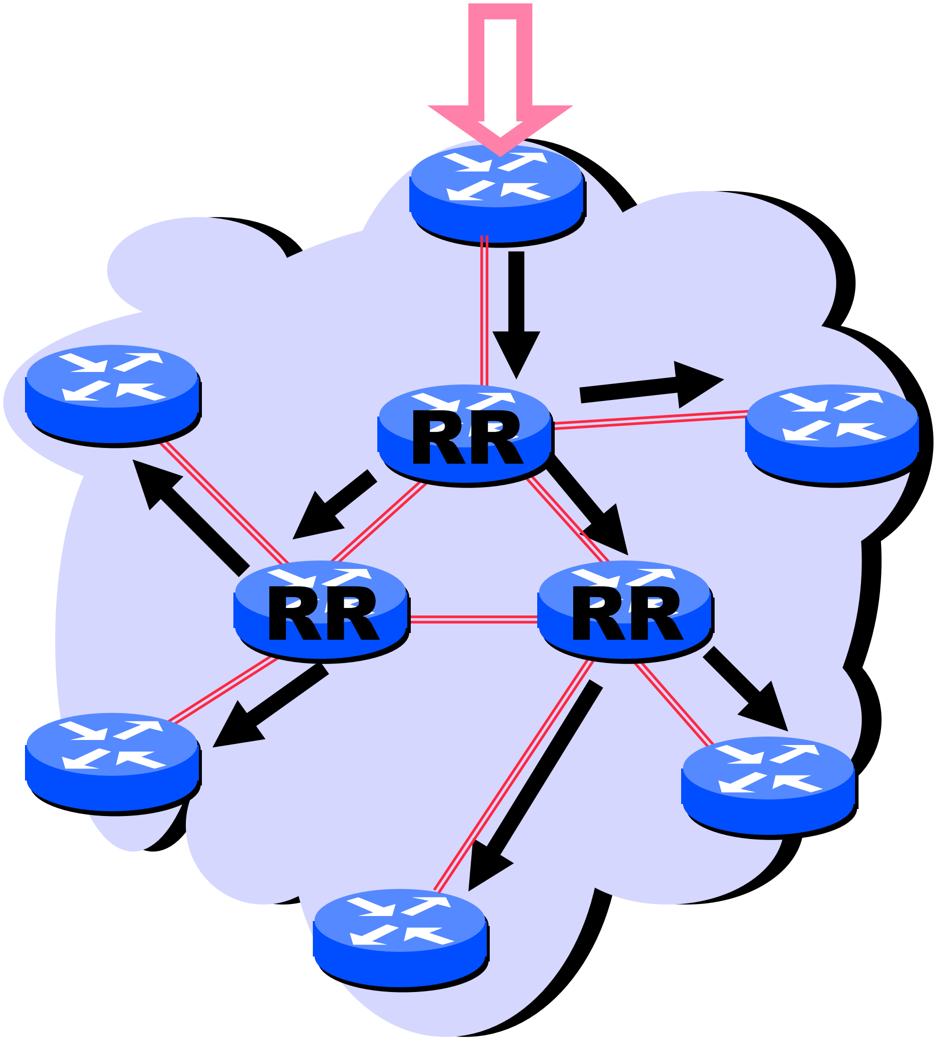
iBGP Mesh Does Not Scale



- N border routers means N(N-1)/2 peering sessions
- Each router must have N-1 iBGP sessions configured
- The addition a single iBGP speaker requires configuration changes to all other iBGP speakers
- Size of iBGP routing table can be order N larger than number of best routes (remember alternate routes!)
- Each router has to listen to update noise from each neighbor

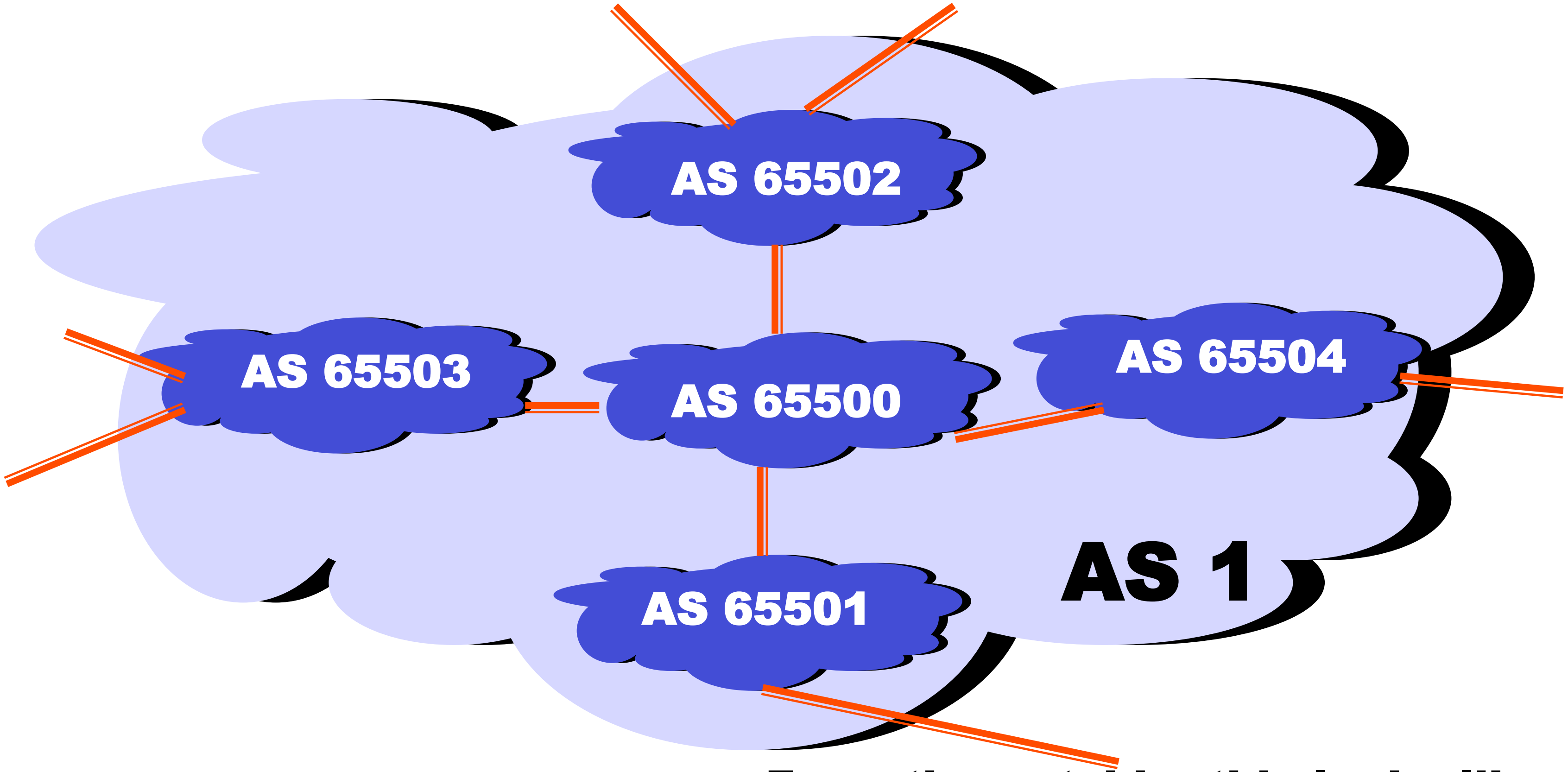
Currently four solutions:
(0) Buy bigger routers!
(1) Break AS into smaller ASes
(2) BGP Route reflectors
(3) BGP confederations

Route Reflectors



- **Route reflectors can pass on iBGP updates to clients**
- **Each RR passes along ONLY best routes**
- **ORIGINATOR_ID and CLUSTER_LIST attributes are needed to avoid loops**

BGP Confederations



From the outside, this looks like AS 1

Confederation eBGP (between member ASes) preserves LOCAL_PREF, MED, and BGP NEXTHOP.

Four Types of BGP Messages

- **Open** : Establish a peering session.
- **Keep Alive** : Handshake at regular intervals.
- **Notification** : Shuts down a peering session.
- **Update** : Announcing new routes or withdrawing previously announced routes.

announcement

=

prefix + attributes values

BGP Attributes

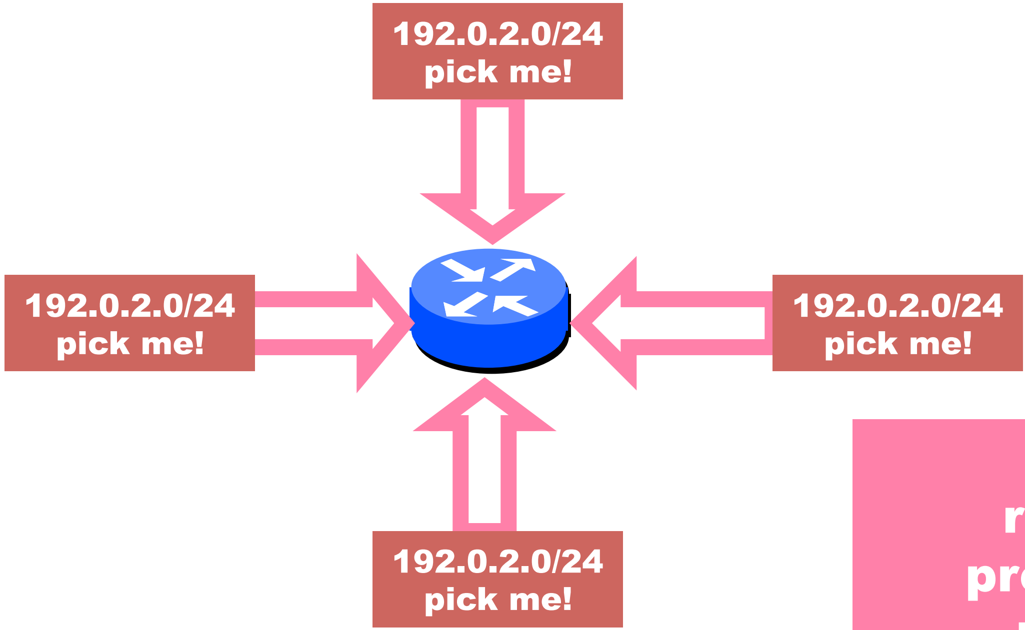
Value	Code	Reference
1	ORIGIN	[RFC1771]
2	AS_PATH	[RFC1771]
3	NEXT_HOP	[RFC1771]
4	MULTI_EXIT_DISC	[RFC1771]
5	LOCAL_PREF	[RFC1771]
6	ATOMIC_AGGREGATE	[RFC1771]
7	AGGREGATOR	[RFC1771]
8	COMMUNITY	[RFC1997]
9	ORIGINATOR_ID	[RFC2796]
10	CLUSTER_LIST	[RFC2796]
11	DPA	[Chen]
12	ADVERTISER	[RFC1863]
13	RCID_PATH / CLUSTER_ID	[RFC1863]
14	MP_REACH_NLRI	[RFC2283]
15	MP_UNREACH_NLRI	[RFC2283]
16	EXTENDED COMMUNITIES	[Rosen]
	...	
255	reserved for development	

From IANA: http://www.iana.org/assignments/bgp-parameters

Most important attributes (values 2–5 and 8–10)

Not all attributes need to be present in every announcement

Attributes are Used to Select Best Routes



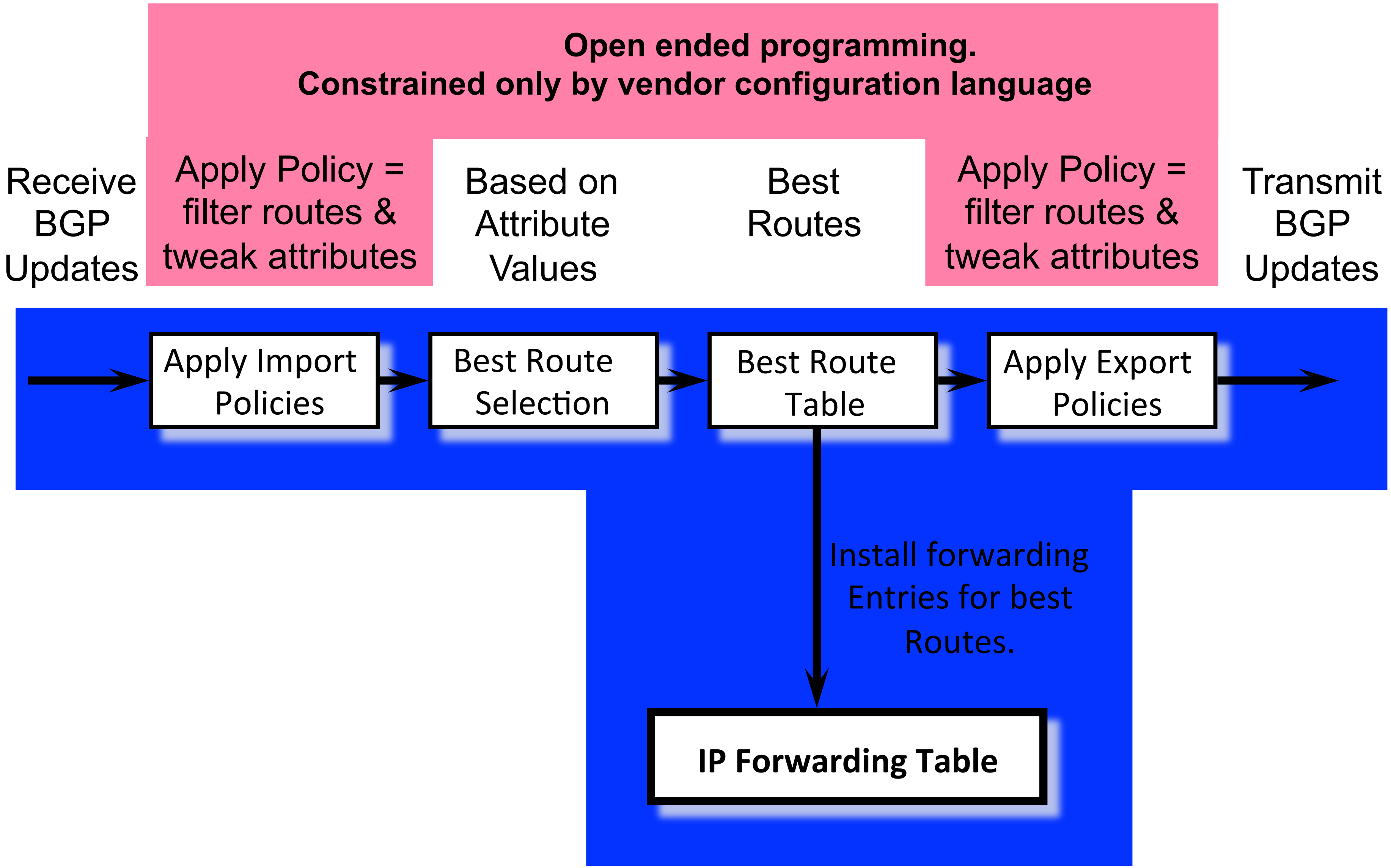
Given multiple routes to the same prefix, a BGP speaker must pick at most <u>one</u> best route

(Note: it could reject them all!)

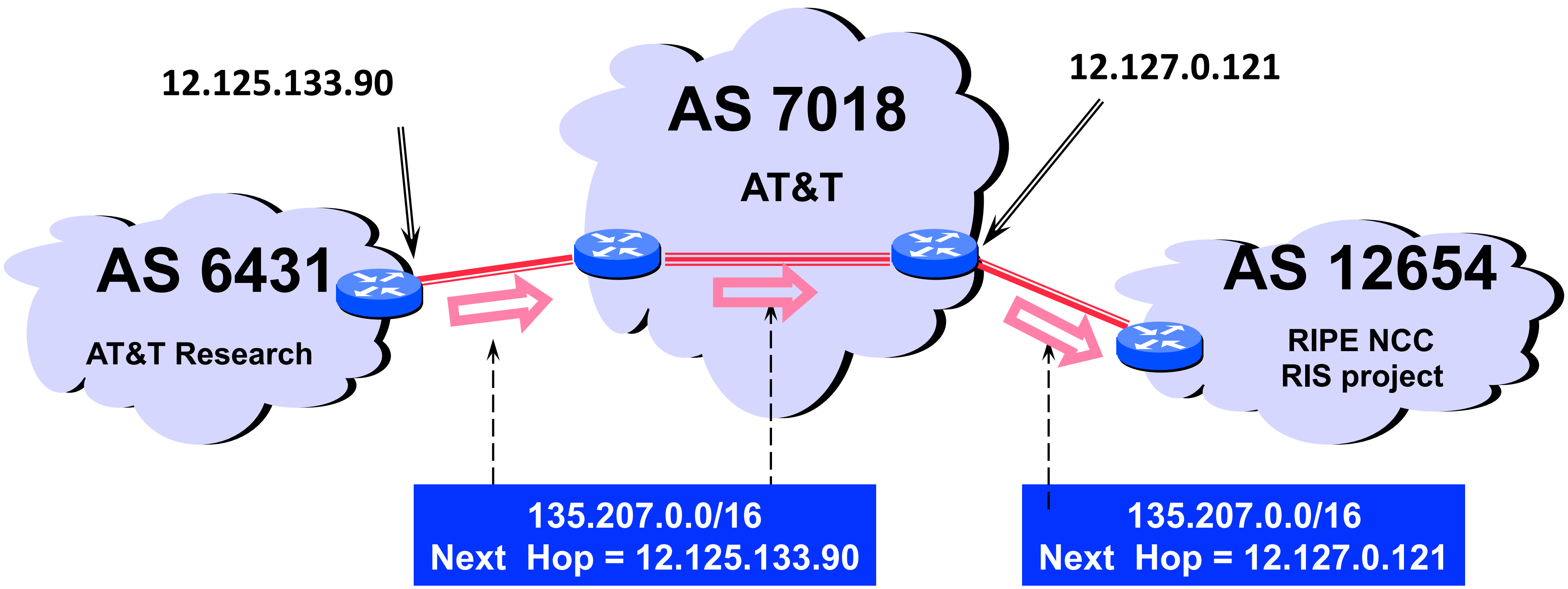
Route Selection Summary

Highest Local Preference	**Enforce relationships**
Shortest ASPATH **Lowest MED** **i-BGP < e-BGP** **Lowest IGP cost to BGP egress**	**traffic engineering**
Lowest router ID	**Throw up hands and break ties**

BGP Route Processing

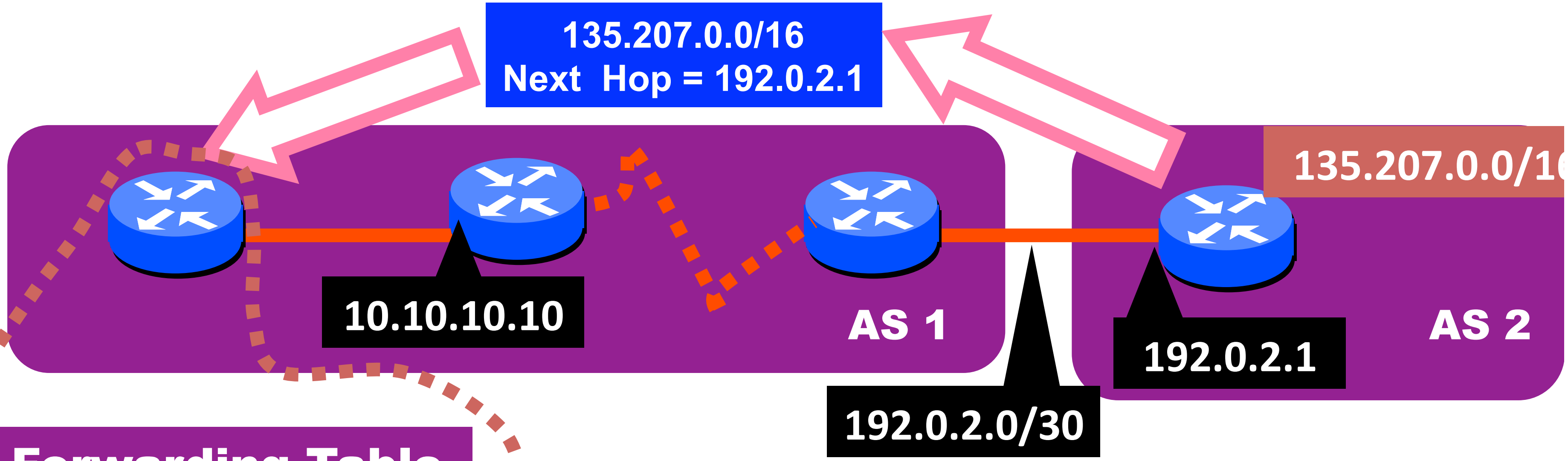


BGP Next Hop Attribute



Every time a route announcement crosses an AS boundary, the Next Hop attribute is changed to the IP address of the border router that announced the route.

Join EGP with IGP For Connectivity



Forwarding Table

destination	next hop
192.0.2.0/30	10.10.10.10

+

EGP

destination	next hop
135.207.0.0/16	192.0.2.1

→

Forwarding Table

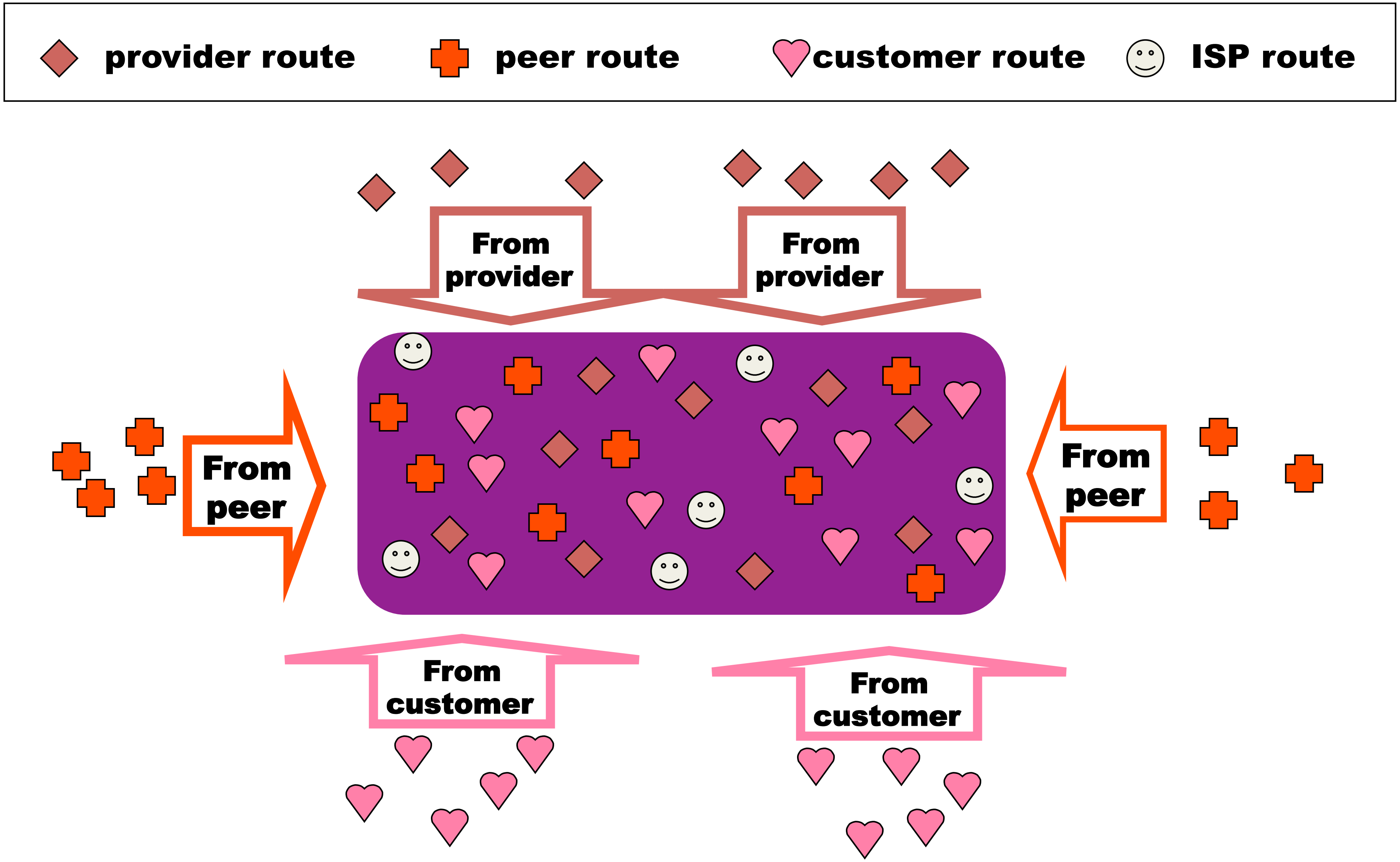
destination	next hop
135.207.0.0/16	10.10.10.10
192.0.2.0/30	10.10.10.10

Implementing Customer/Provider and Peer/Peer relationships

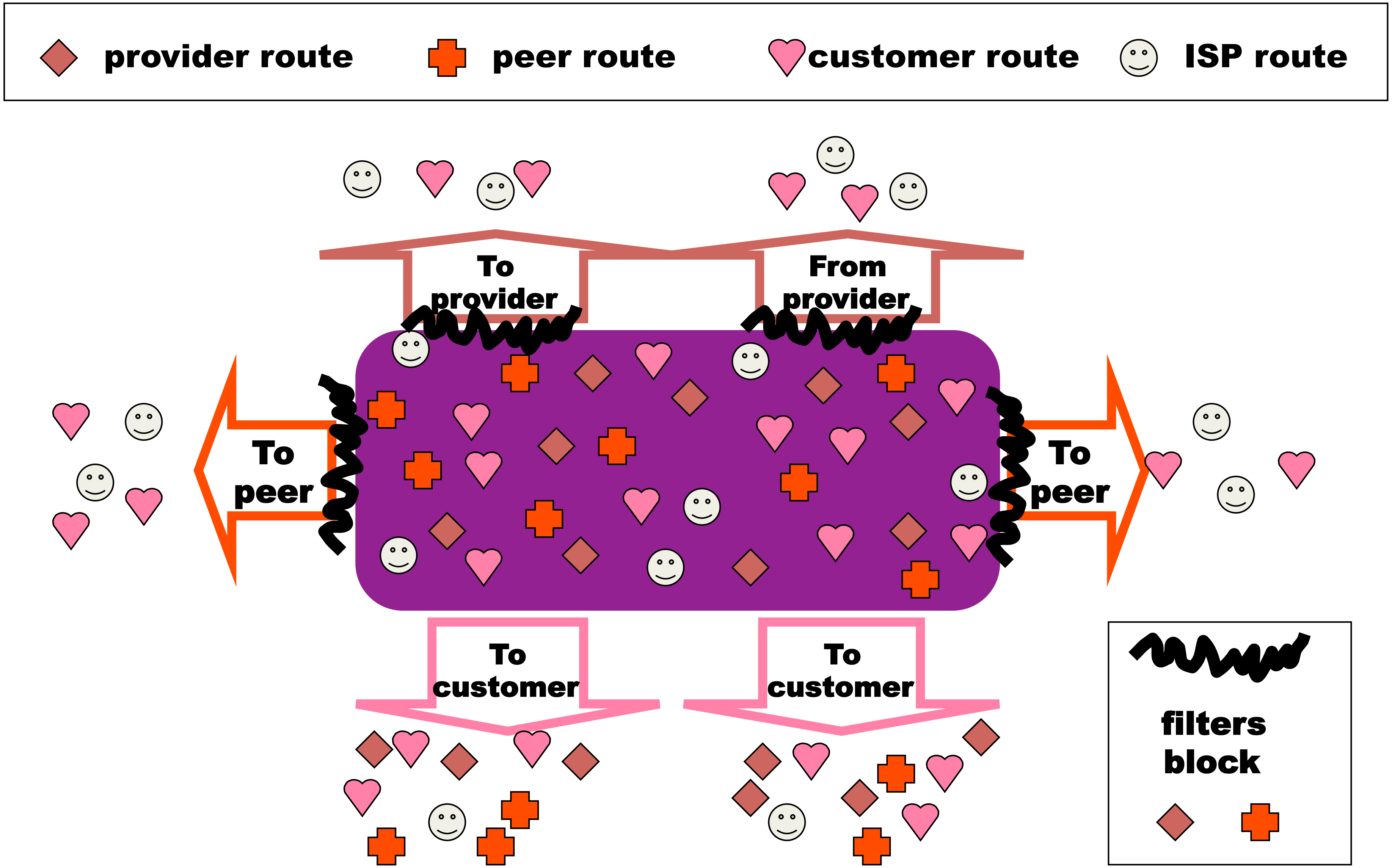
Two parts:

- Enforce transit relationships
  - Outbound route filtering
- Enforce order of route preference
  - provider < peer < customer

Import Routes

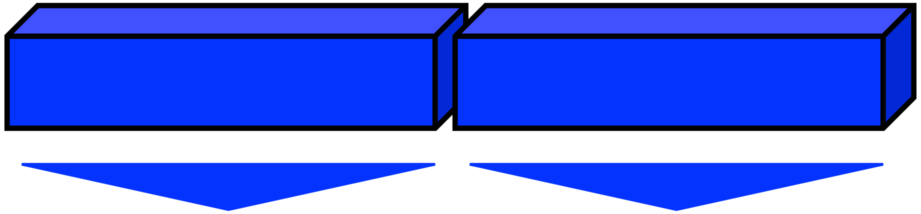


Export Routes



How Can Routes be Colored? BGP Communities!

A community value is 32 bits



By convention, first 16 bits is ASN indicating who is giving it an interpretation

community number

Used for signally within and between ASes

Very powerful BECAUSE it has no (predefined) meaning

Community Attribute = a list of community values.
(So one route can belong to multiple communities)

RFC 1997 (August 1996)

Two reserved communities

no_export = 0xFFFFFF01: don't export out of AS

no_advertise 0xFFFFFF02: don't pass to BGP neighbors

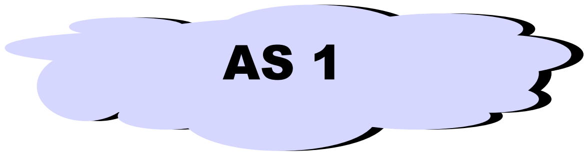
Communities Example

- 1:100
  - Customer routes
- 1:200
  - Peer routes
- 1:300
  - Provider Routes

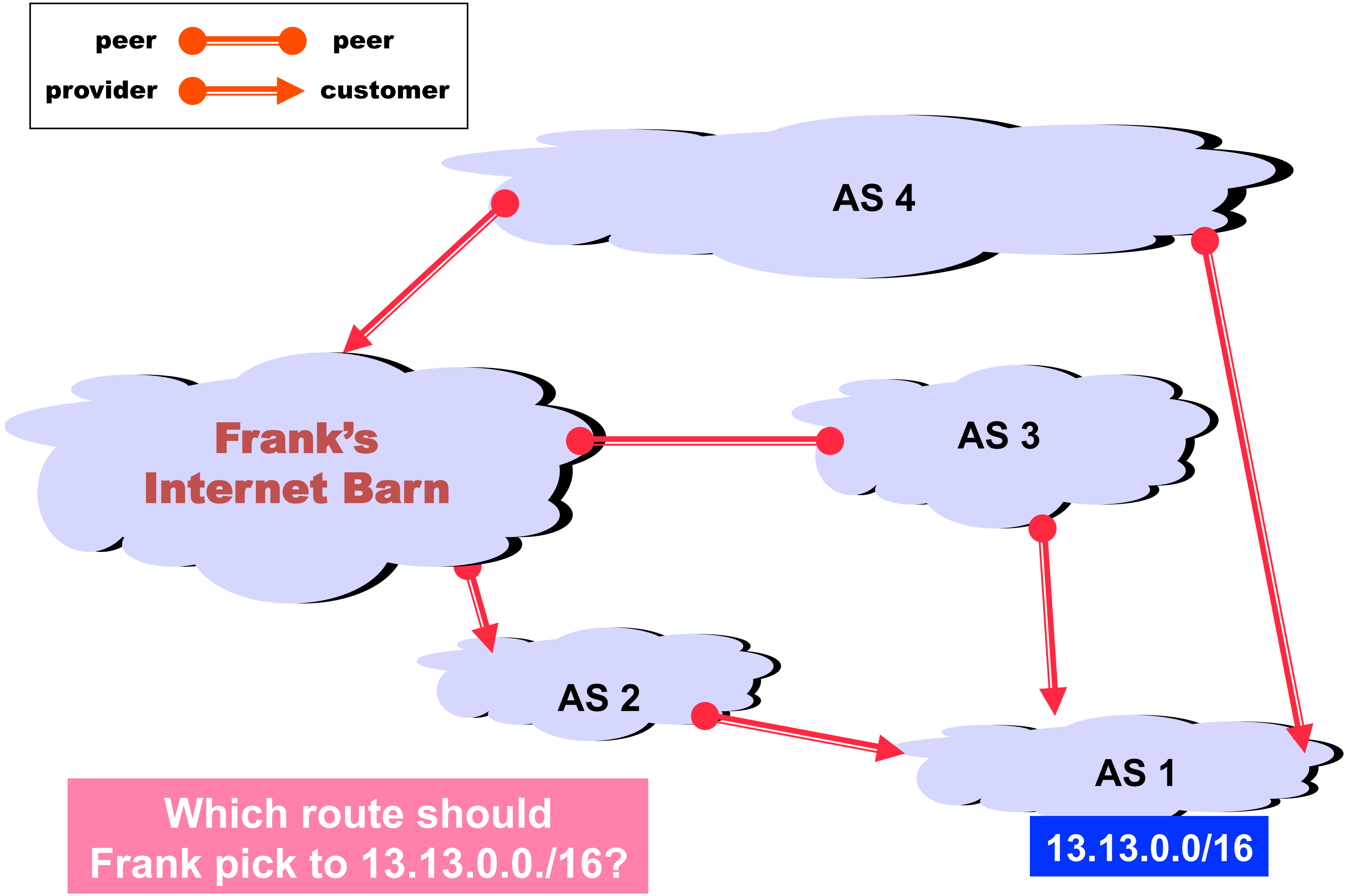
Import

- To Customers
  - 1:100, 1:200, 1:300
- To Peers
  - 1:100
- To Providers
  - 1:100

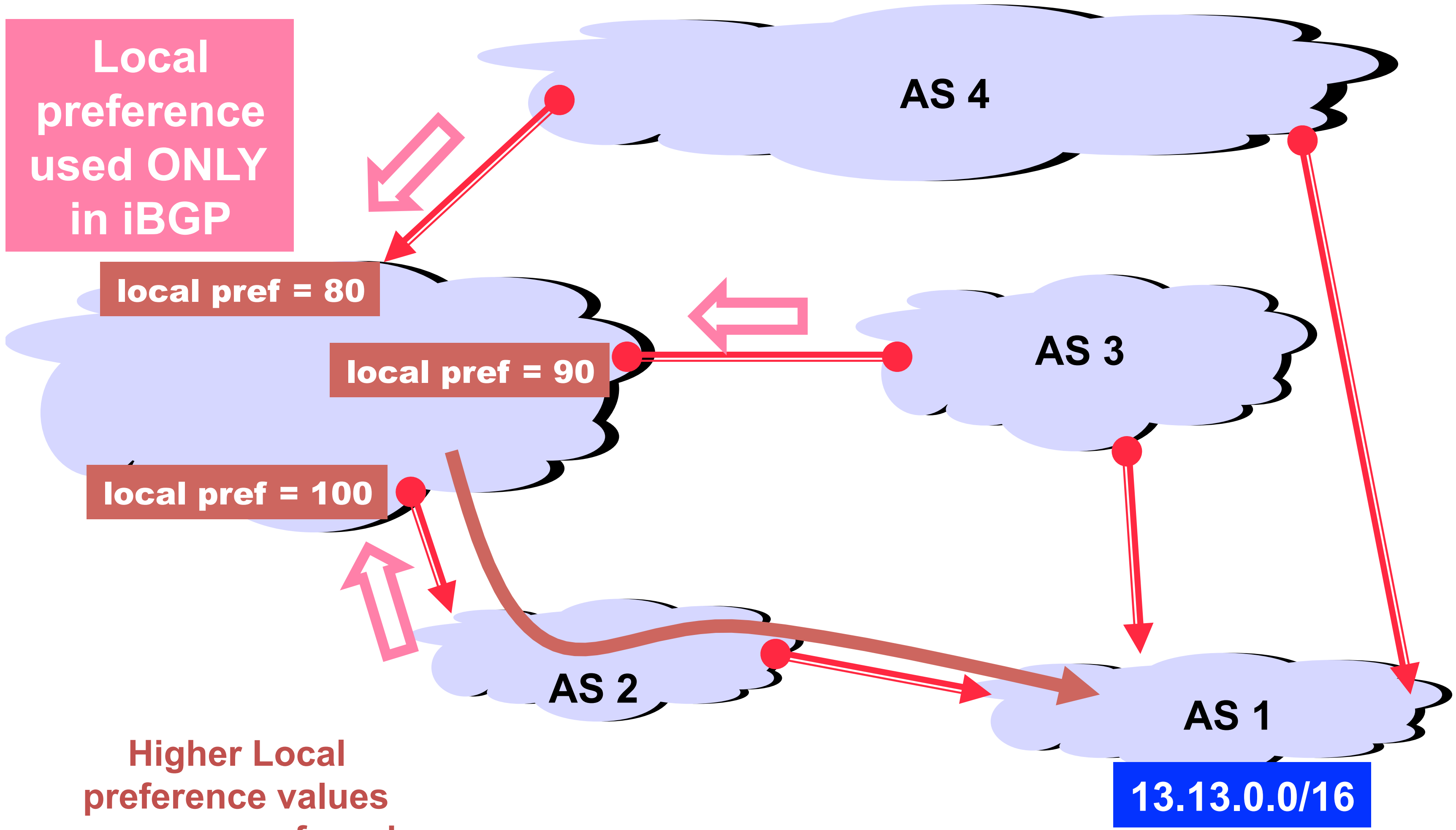
Export



So Many Choices



LOCAL PREFERENCE



Higher Local preference values are more preferred