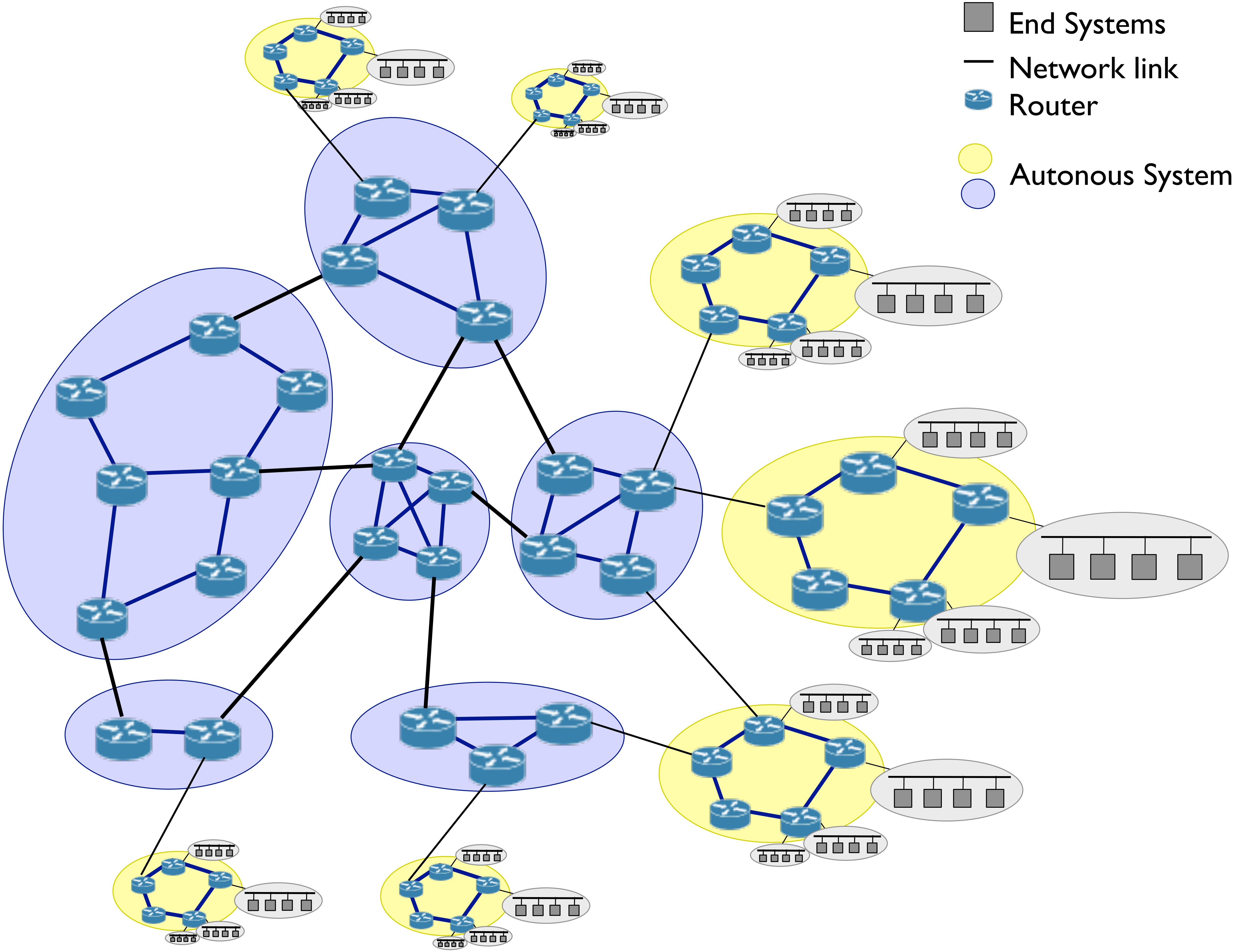
Mechanisms for Interdomain Traffic Engineering with LISP

Damien Saucez

Background

Internet?


amazon.com
YouTube
Innovators: LISP
facebook
Google
ISP Driven informed path selection



Inspired by http://interstices.info/jcms/c_15921/internet-le-conglomerat-des-reseaux

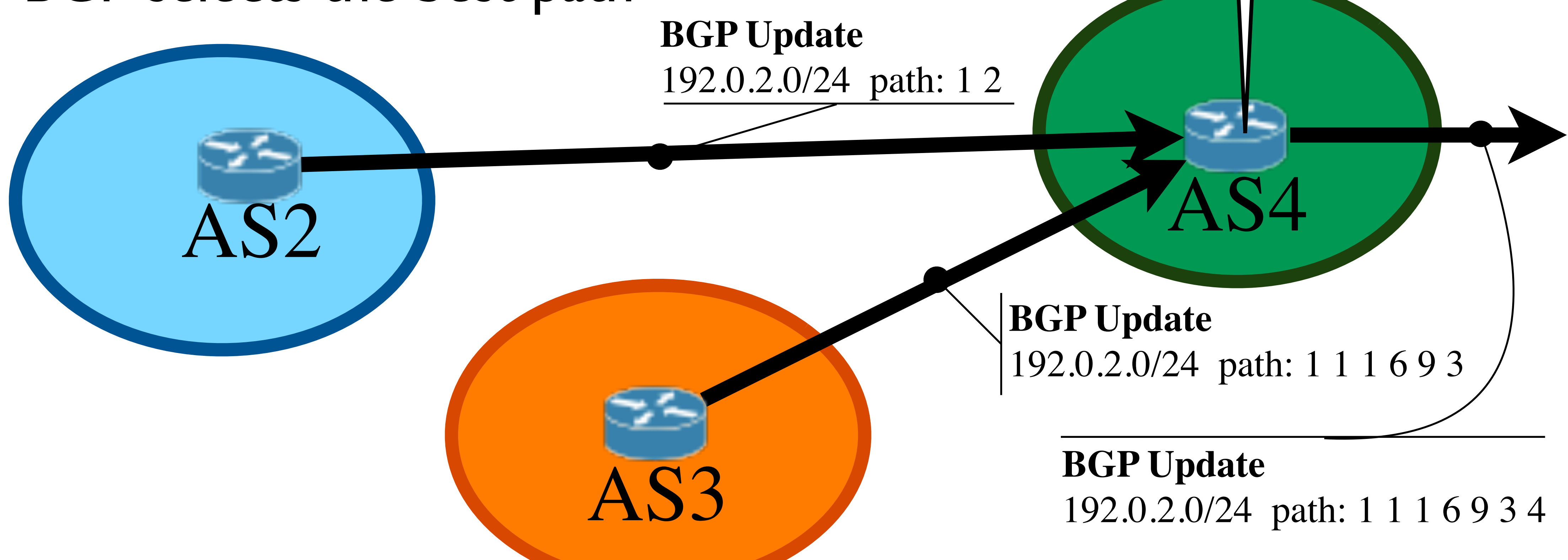
Internet Protocol (IP)

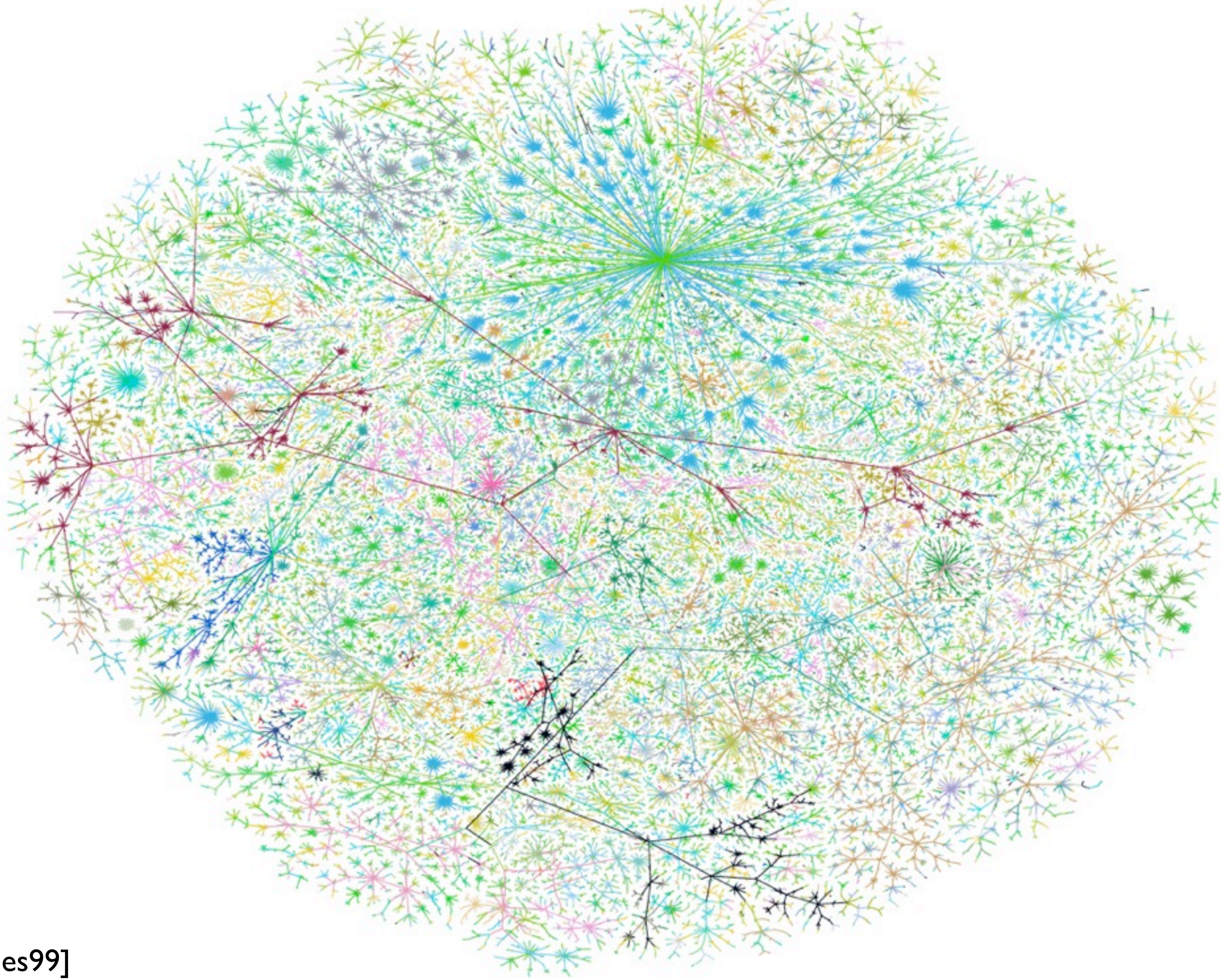
- Specifies the format to respect to exchange data packets on the Internet
- One unique IP address per network interface per device
  - 192.0.2.1 (IPv4), 2001:DB8::0b0:15:900d (IPv6)
- Devices in the same network share the same prefix
  - {192.0.2.1, 192.0.2.254, ...} ∈ 192.0.2.0/24,
{2001:DB8::0b0:15:900d, 2001:DB8::cafe,...} ∈ 2001:DB8::/32
- Complementary role of IP addresses
  - **identifier** role of IP addresses vs **locator** role of IP prefixes

Border Gateway Protocol (BGP)

- BGP is the routing protocol that allows each network on the Internet to signal to other networks what destinations they can reach
  - BGP learns multiple paths to each route
  - BGP selects the best path

Network	Nh	LP	Path
192.0.2.0/24	blue	10	1 2 e
192.0.2.0/24	orange	100	1 1 1 6 9 3 e




[Ches99]

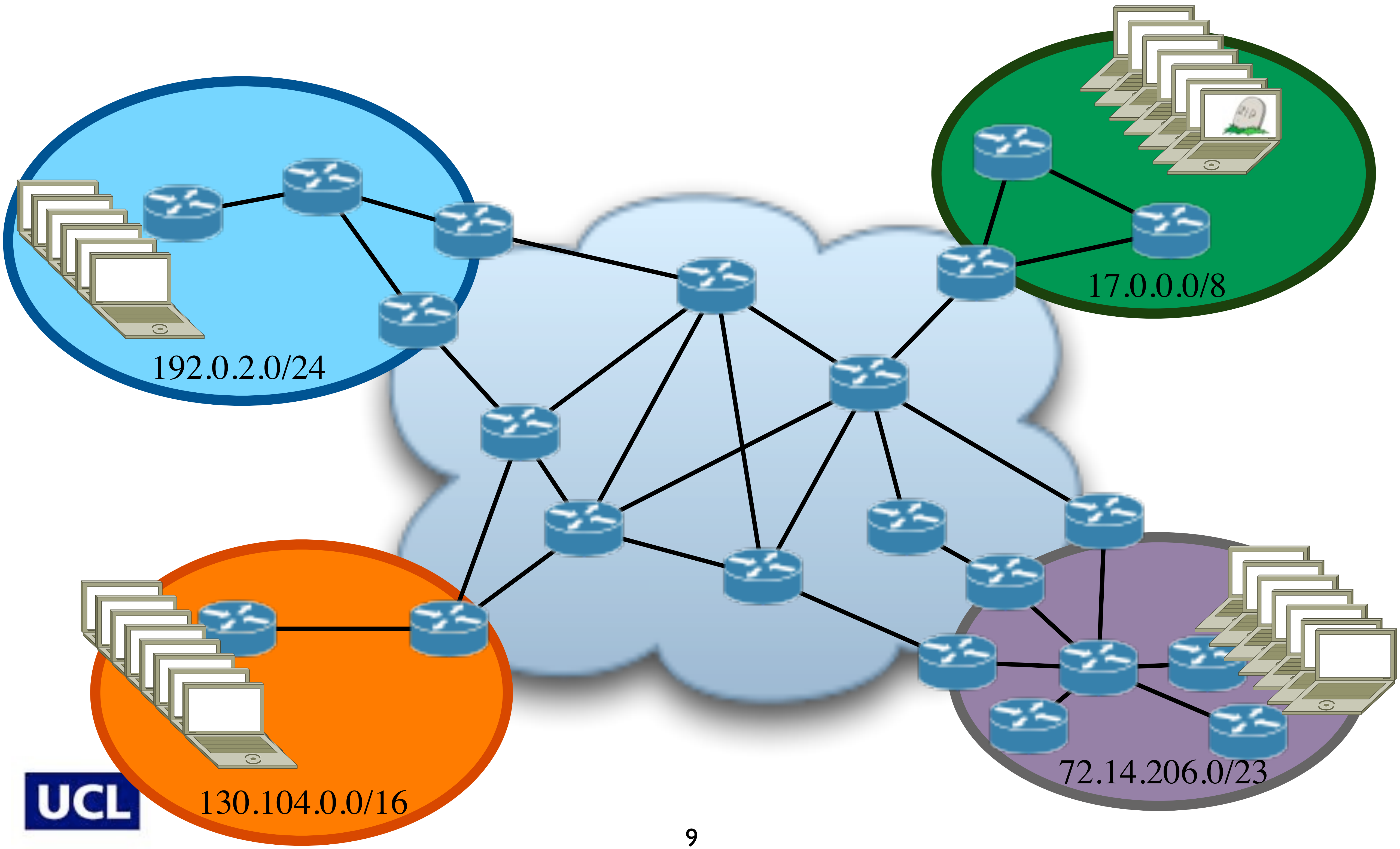
- ~ 35K ASes
  - 75% are multihomed
  - linear increase
- ~ 350K prefixes
  - super-linear increase
- Billions of devices

[Ches99]

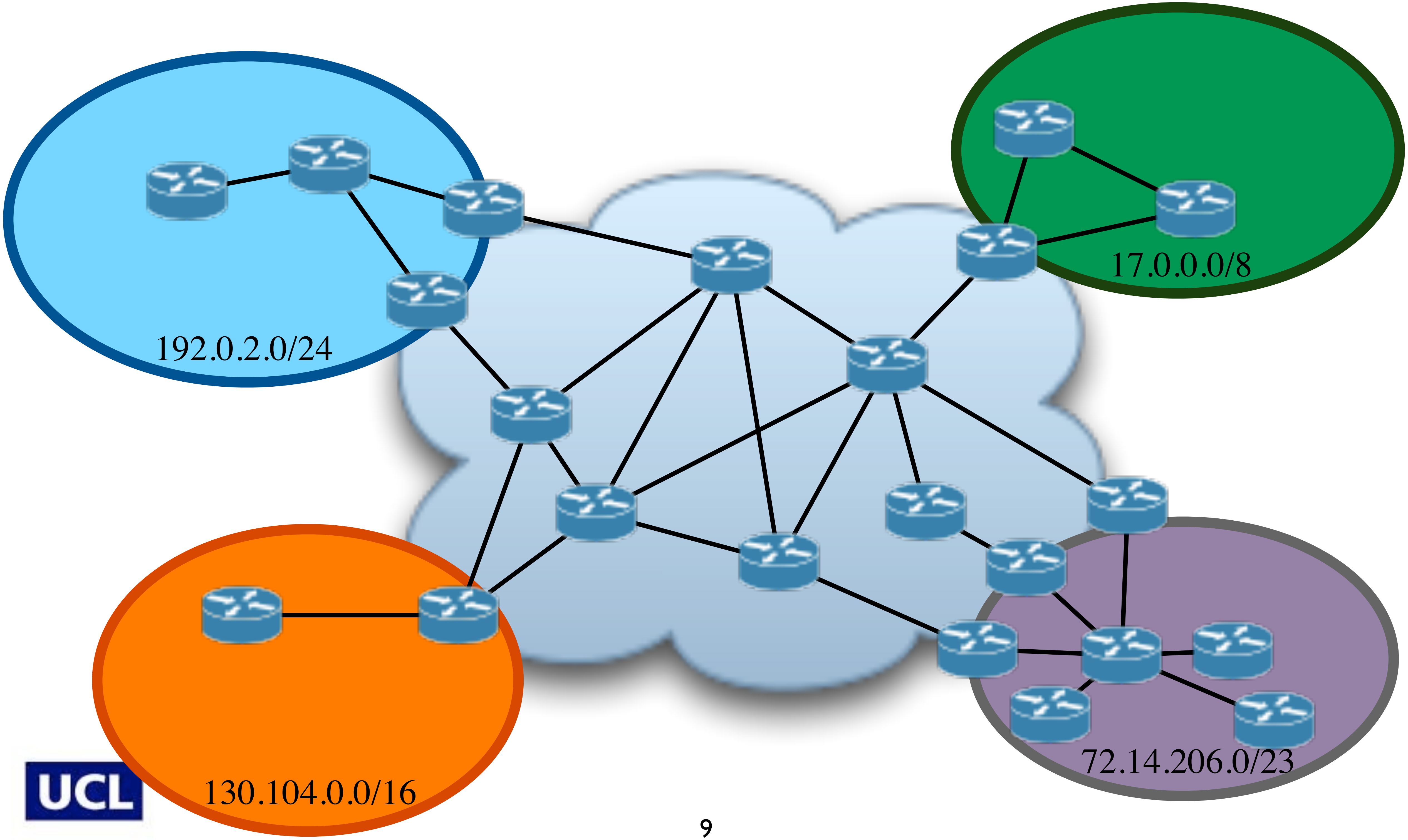
Are the paths equal?

And if they are not, how can we use the best ones?

Methodology

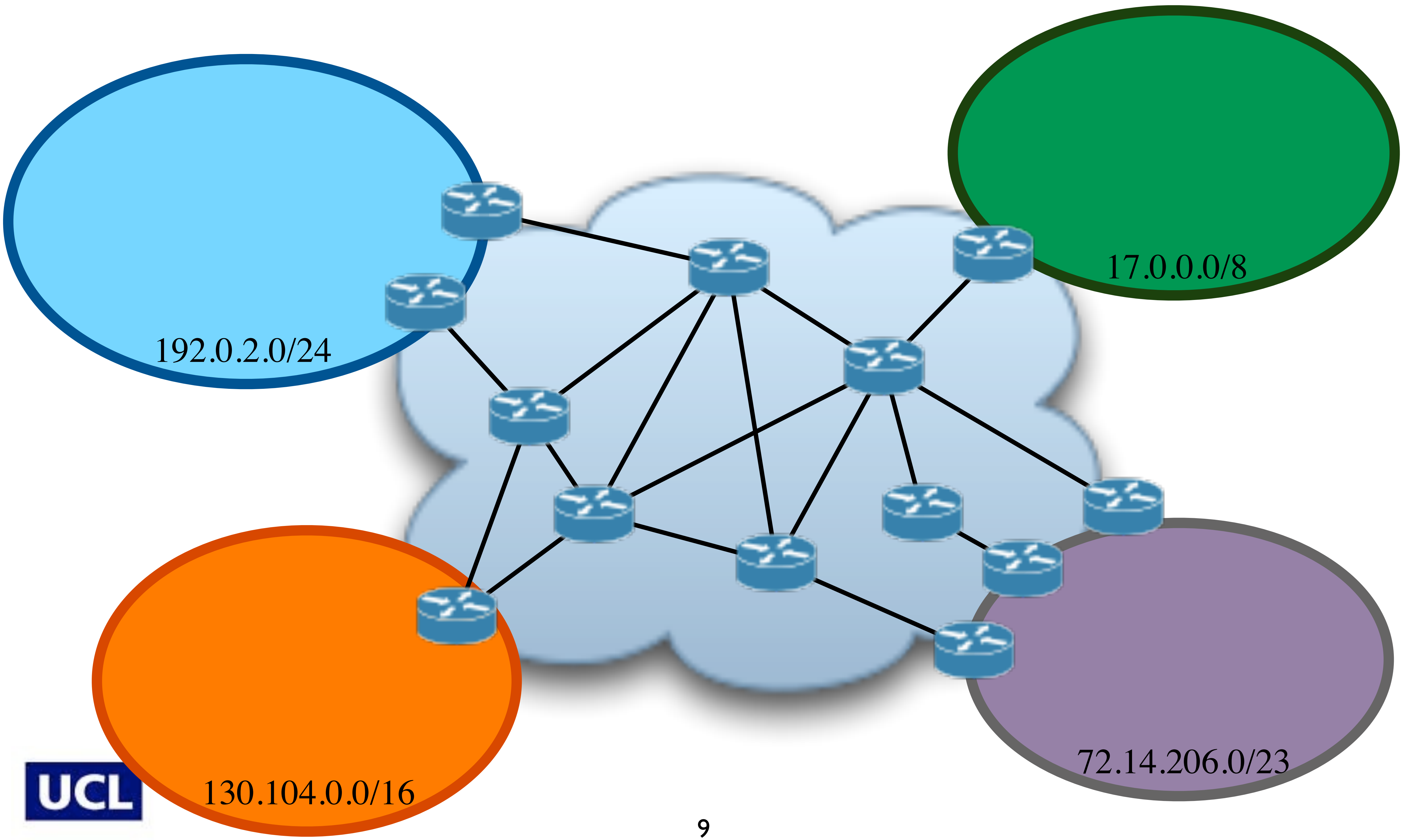


Methodology



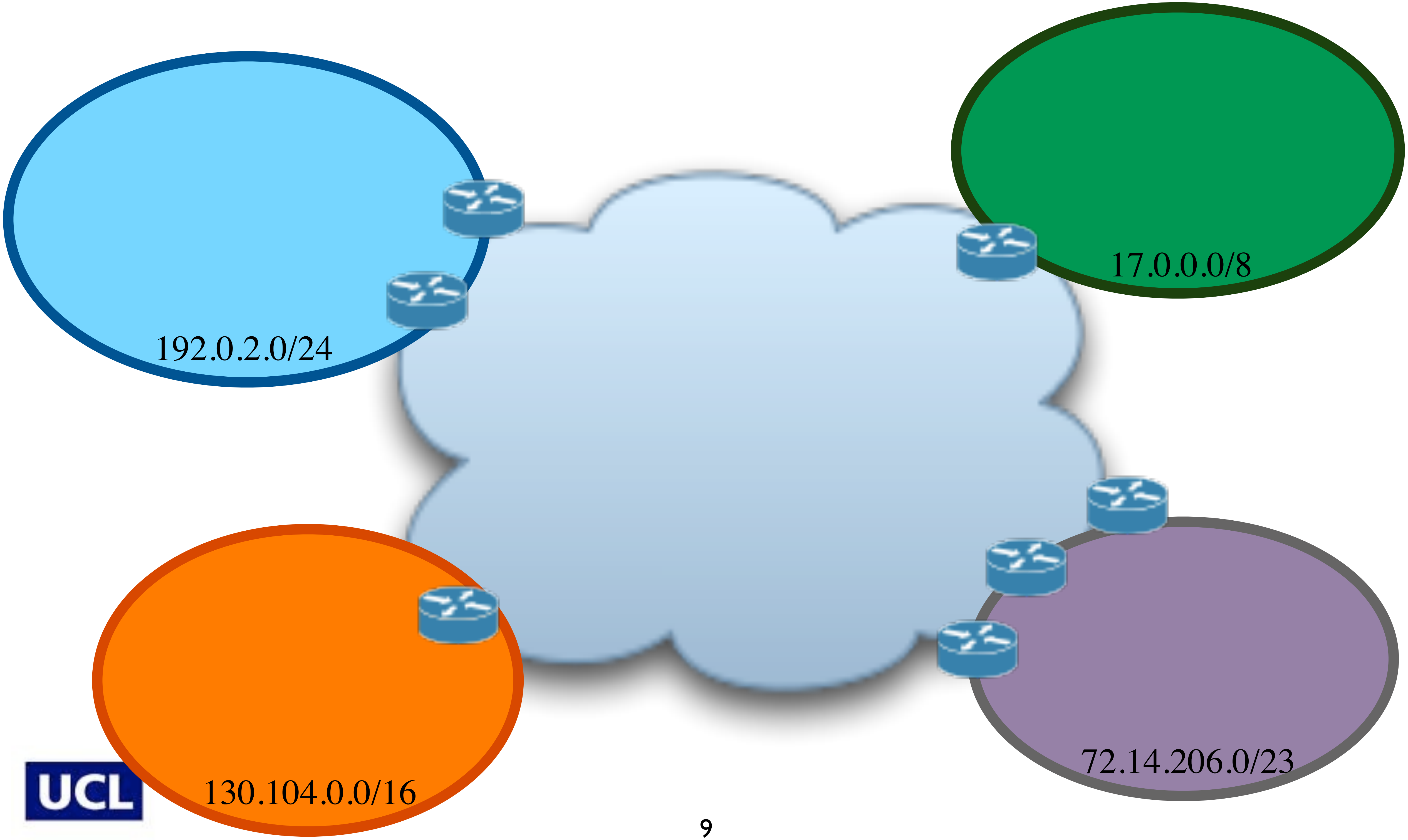
UCL

Methodology



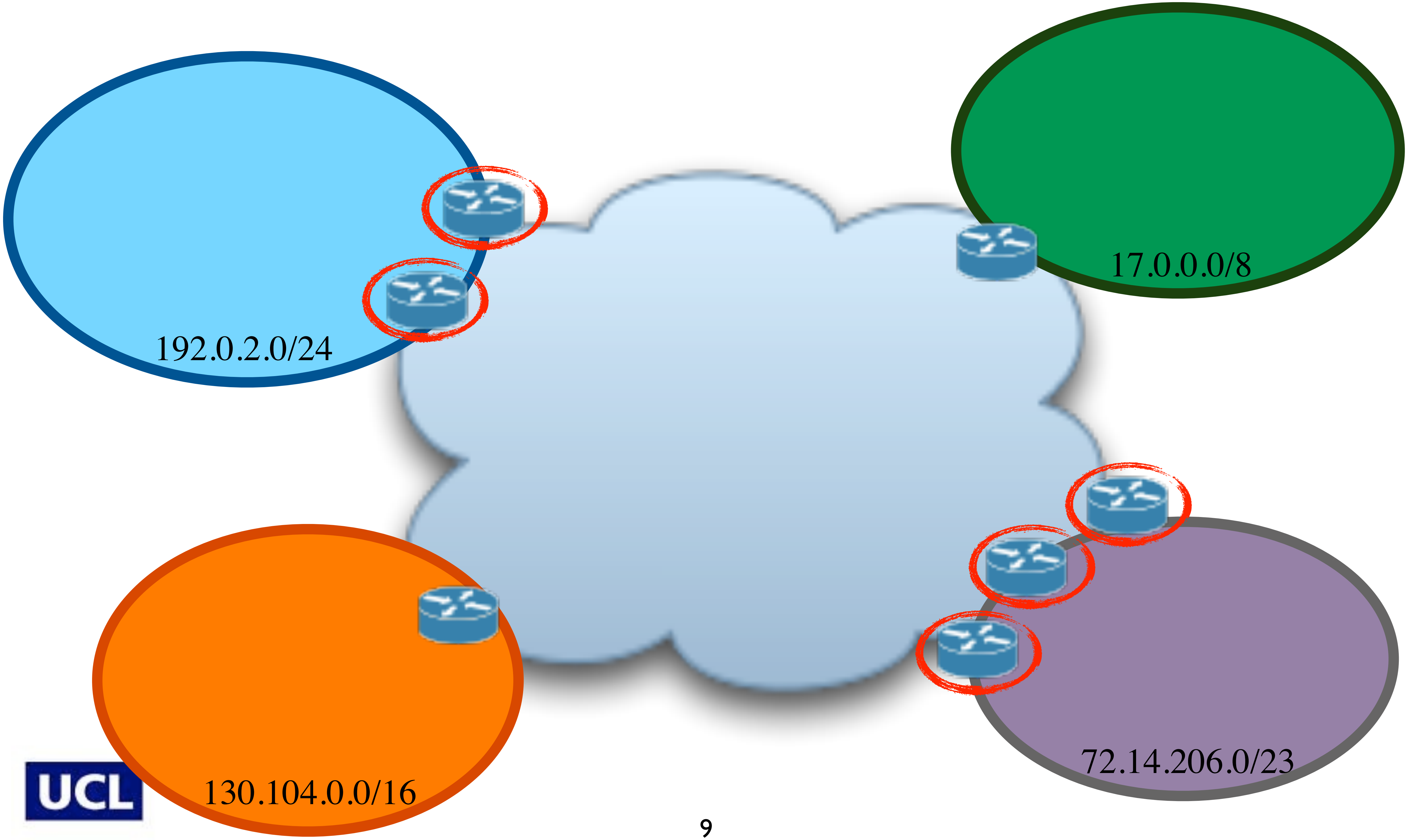
UCL

Methodology

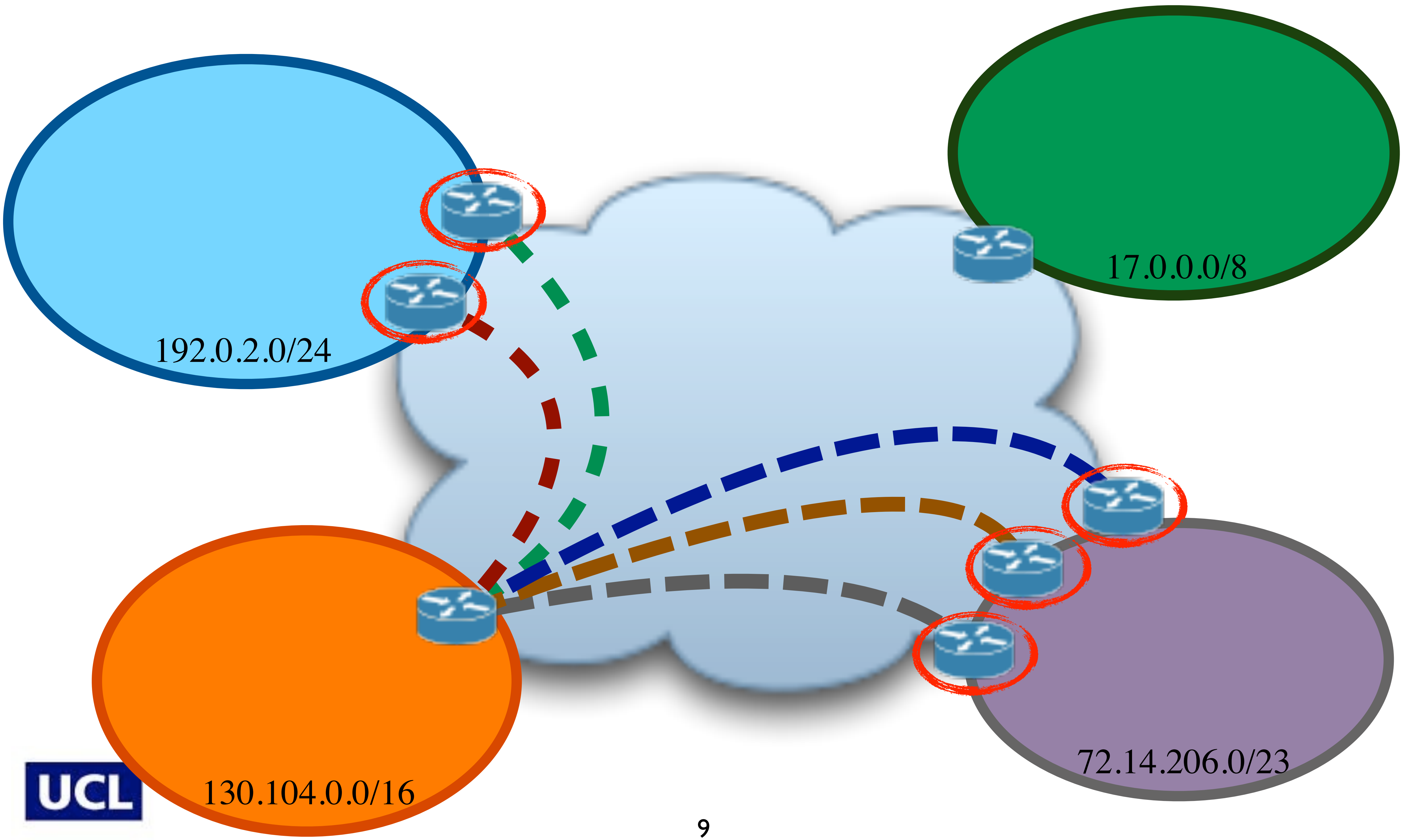


UCL

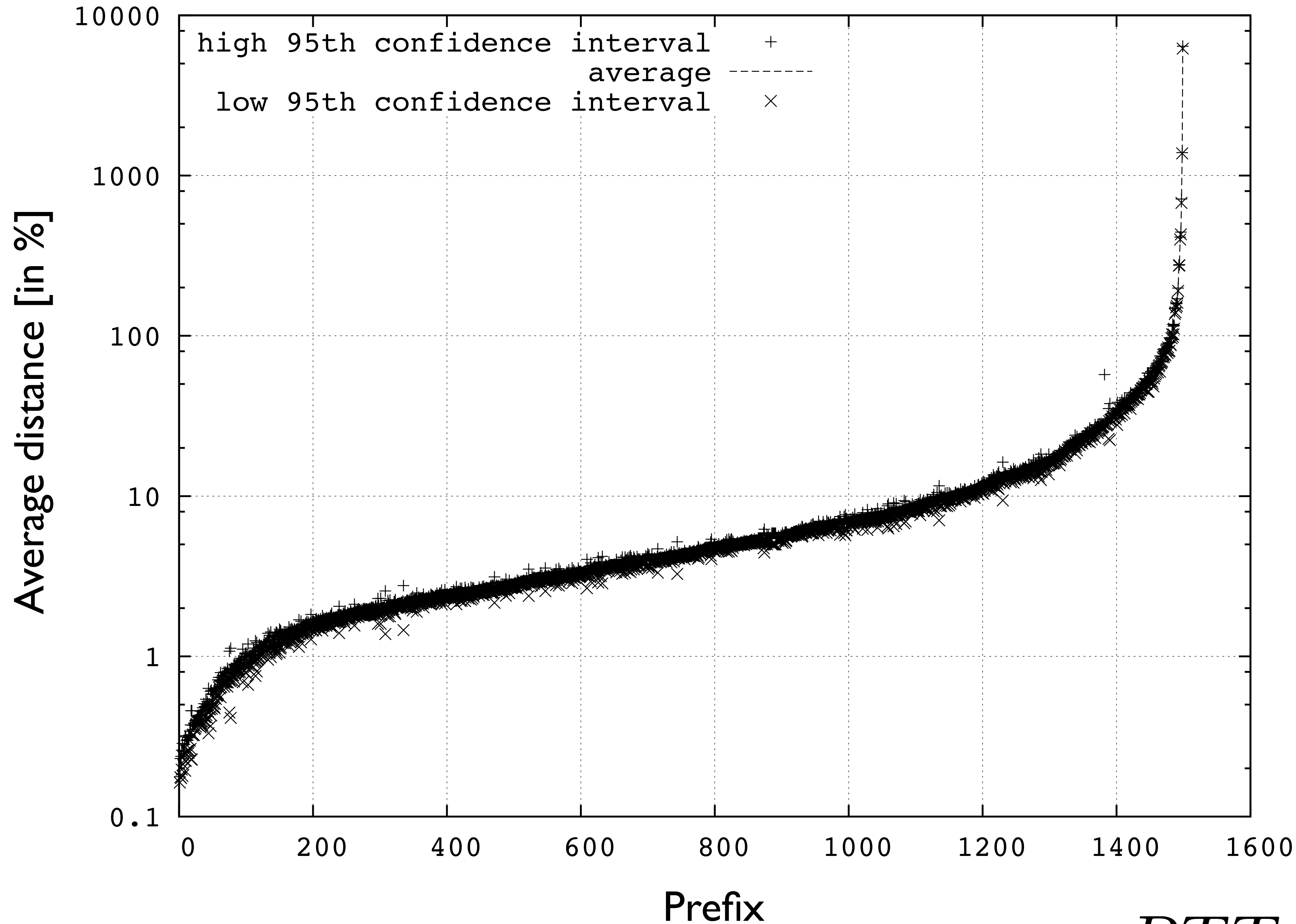
Methodology



Methodology

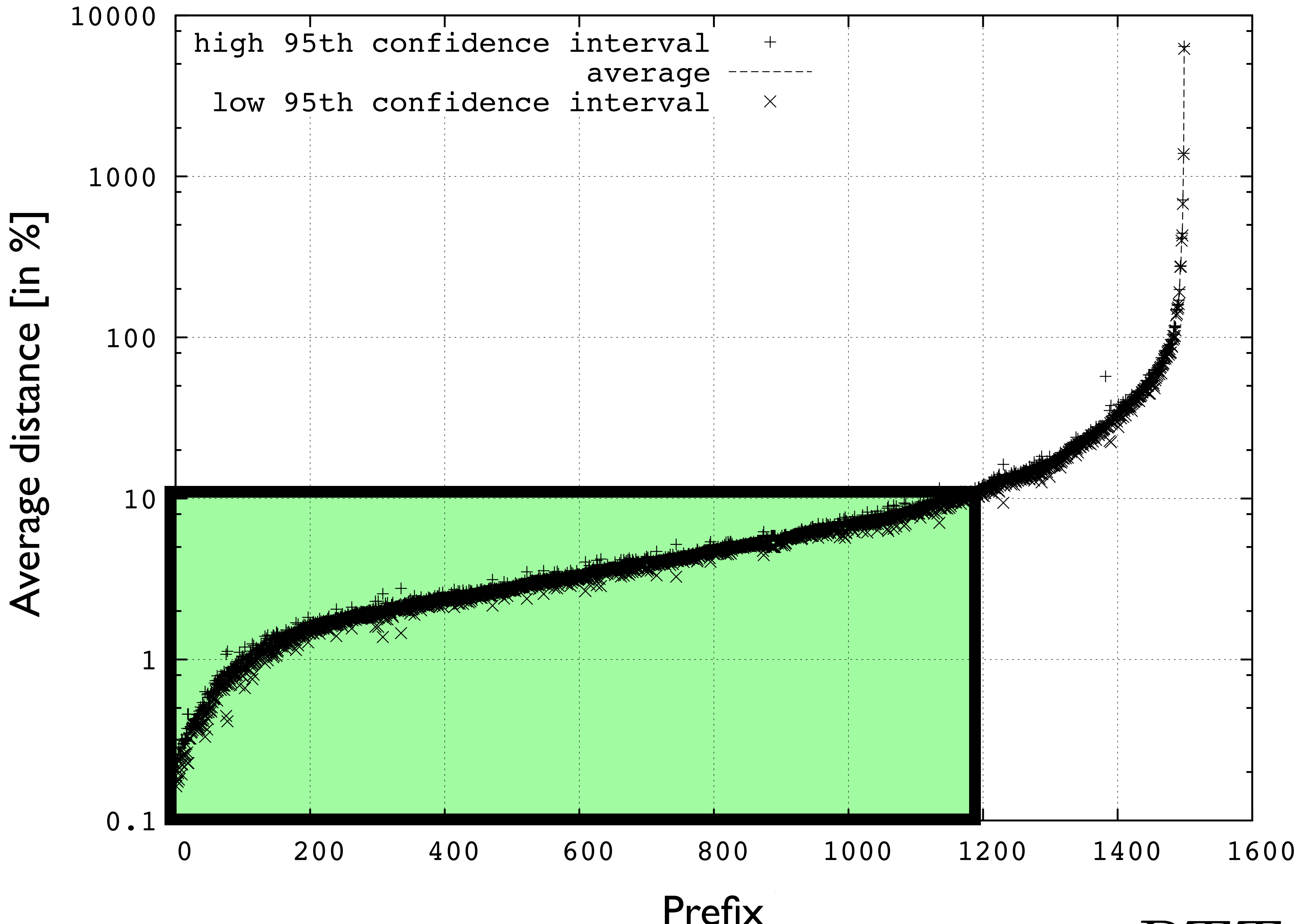


All the paths are not equal



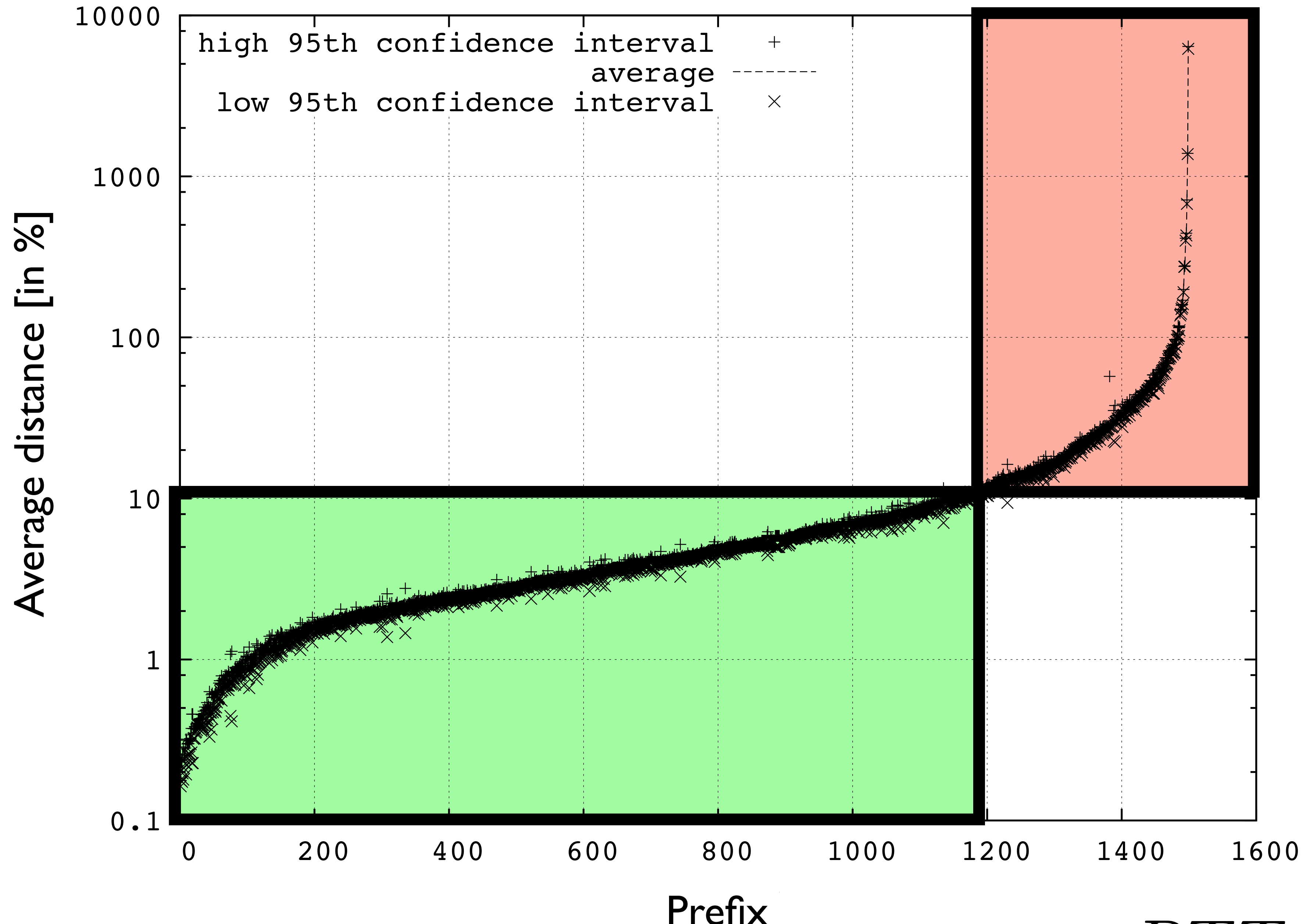
$$\text{distance} = \frac{RTT_i - RTT_{\text{fastest}}}{RTT_{\text{fastest}}}$$

All the paths are not equal



$$\text{distance} = \frac{RTT_i - RTT_{\text{fastest}}}{RTT_{\text{fastest}}}$$

All the paths are not equal



$$\text{distance} = \frac{RTT_i - RTT_{\text{fastest}}}{RTT_{\text{fastest}}}$$

Problems

- Limited incoming traffic control with BGP
- Performance evolve with time
- The IP schizophrenia
  - Change the path? Change the address!
    - and break the data flows...

What do we need to enable performance based interdomain incoming traffic engineering?

What do we need?

What do we need?

- A system that allows one to know which path provides the best performance
  - IDIPS [SDB09,SDIB08,SDB07]

What do we need?

- A system that allows one to know which path provides the best performance
  - IDIPS [SDB09,SDIB08,SDB07]
- A system to control the incoming traffic
  - LISP and LISP-Tree [SV09 ,JCAC+10, ISB11]

Traffic control with the Locator/ID Separation Protocol (LISP)

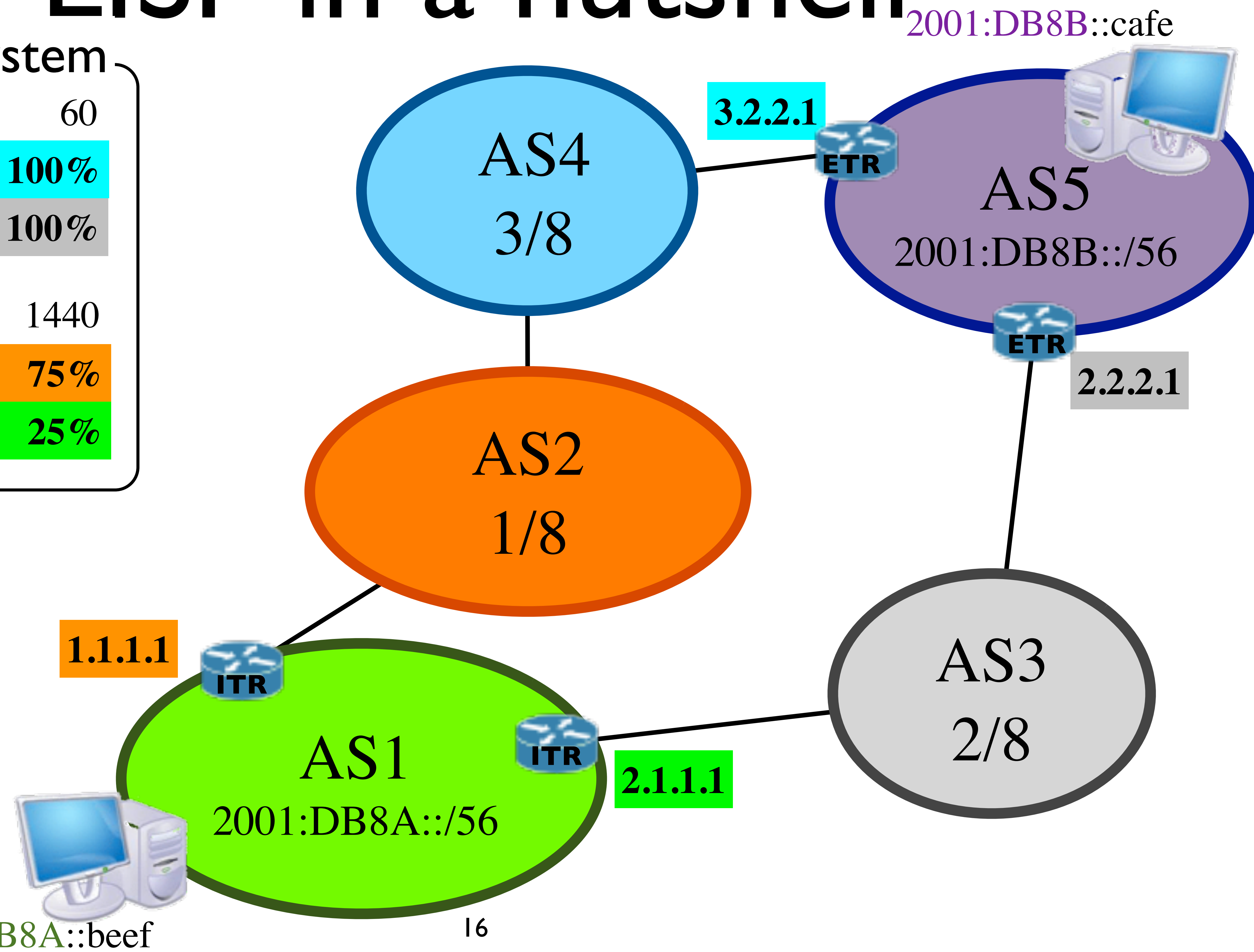
LISP philosophy

- Split the IP address space in two at the border routers
  - Endpoint IDentifiers (EID)
    - identify end-systems and edge routers
    - non-globally routable
    - end systems in a site share the same EID prefix
  - Routing LOCators (RLOC)
    - attached to core routers (router interfaces)
    - globally routable

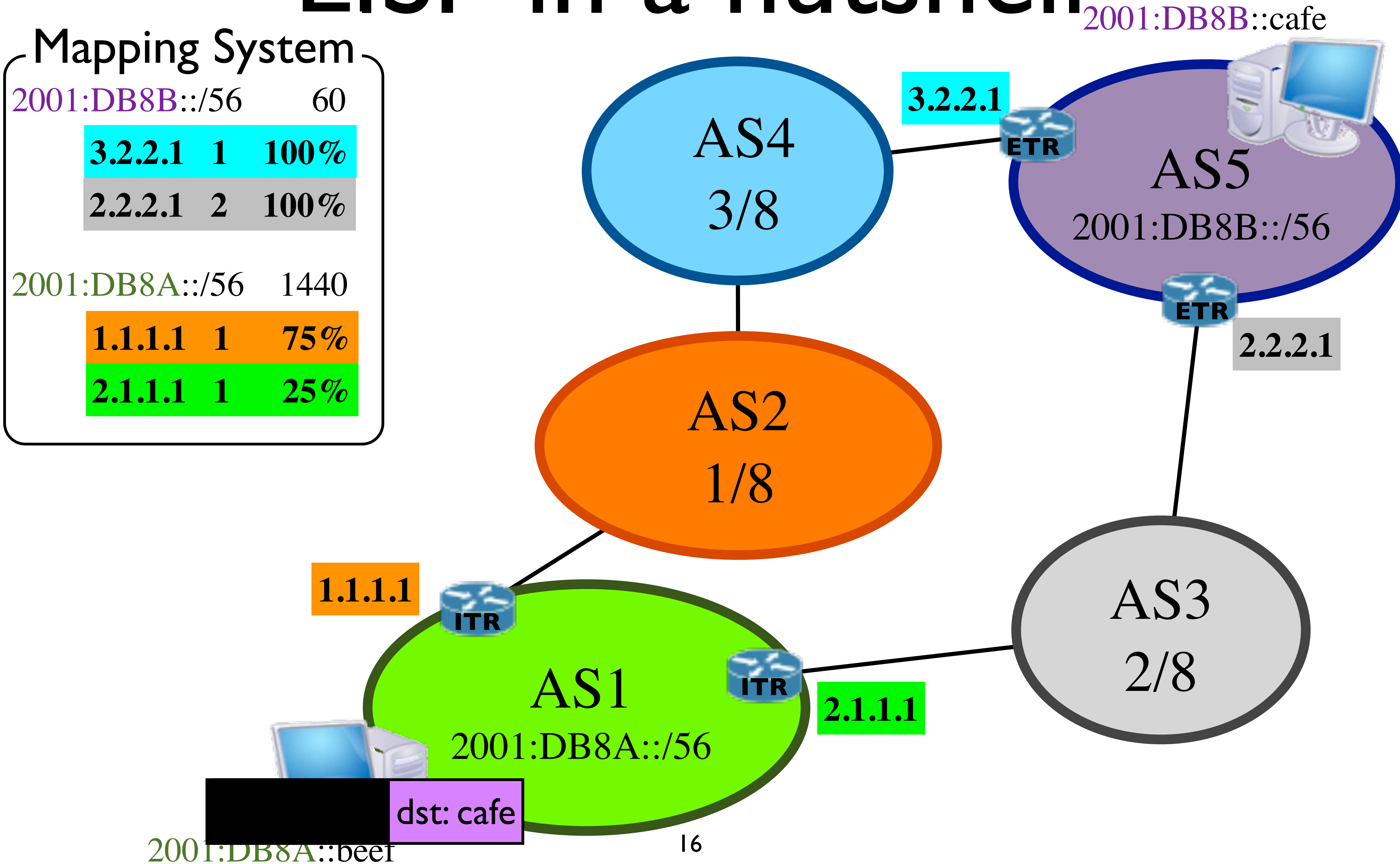
LISP in a nutshell

Mapping System

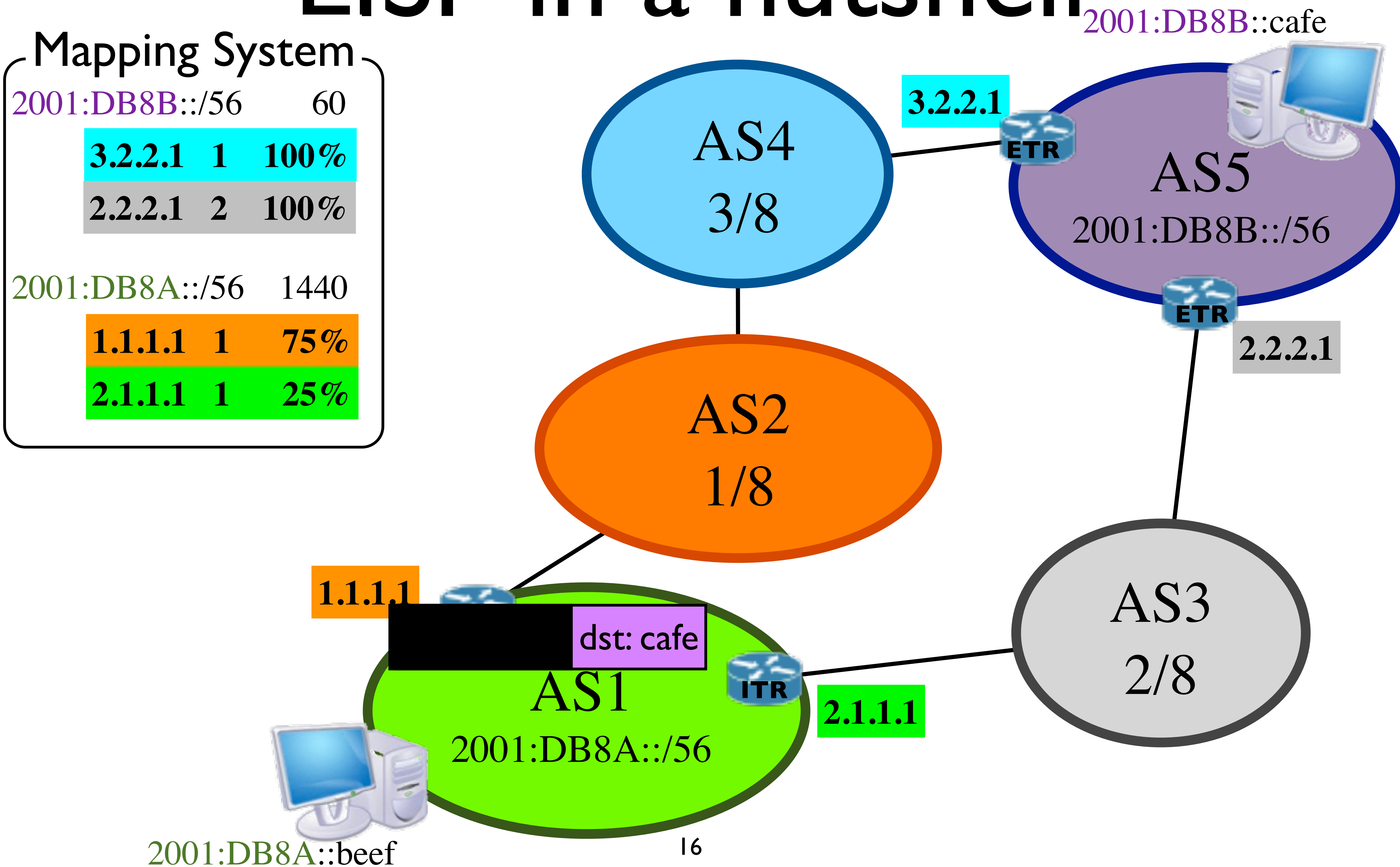
2001:DB8B::/56			60
	3.2.2.1	1	100%
	2.2.2.1	2	100%
2001:DB8A::/56			1440
	1.1.1.1	1	75%
	2.1.1.1	1	25%



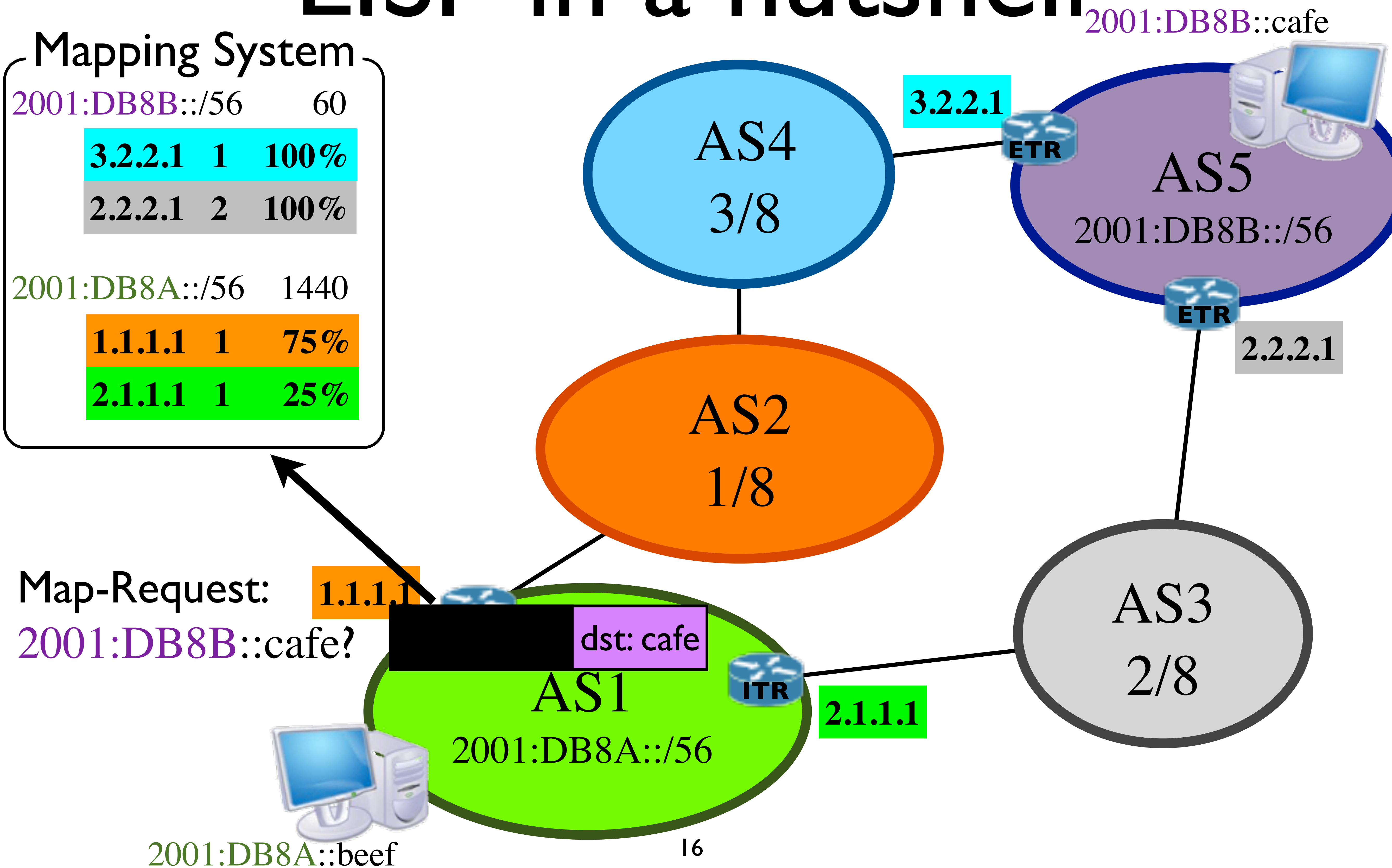
LISP in a nutshell



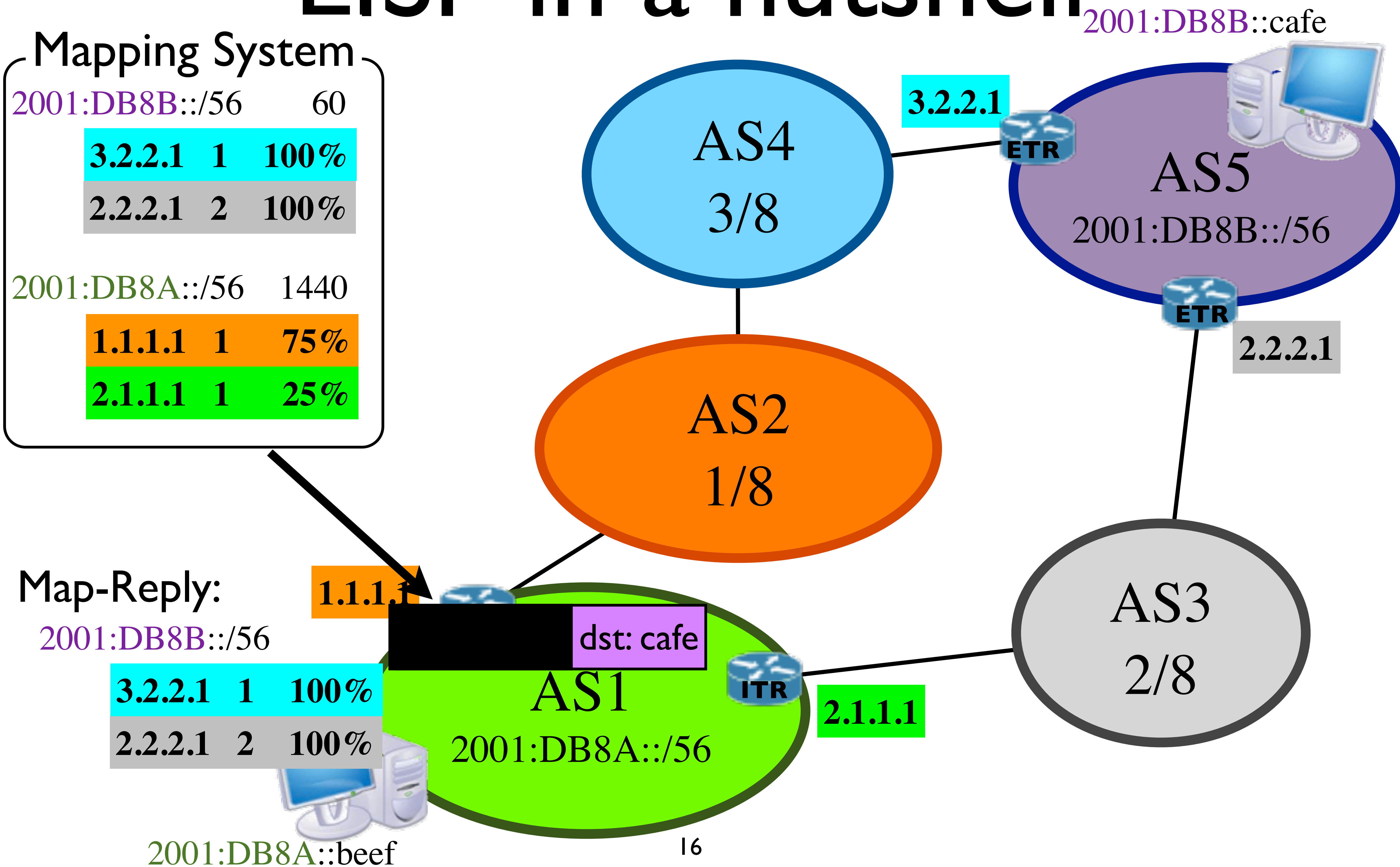
LISP in a nutshell



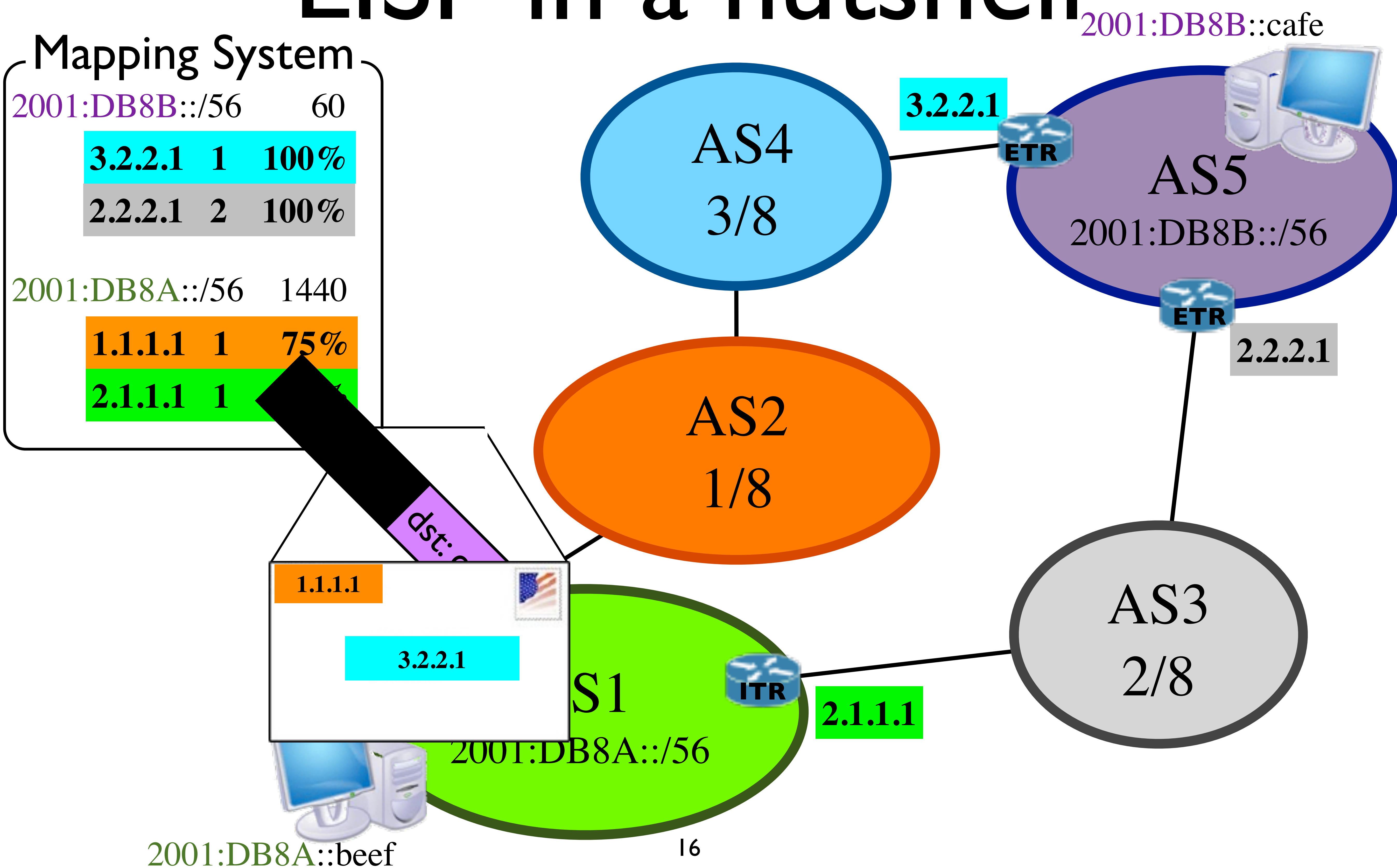
LISP in a nutshell



LISP in a nutshell



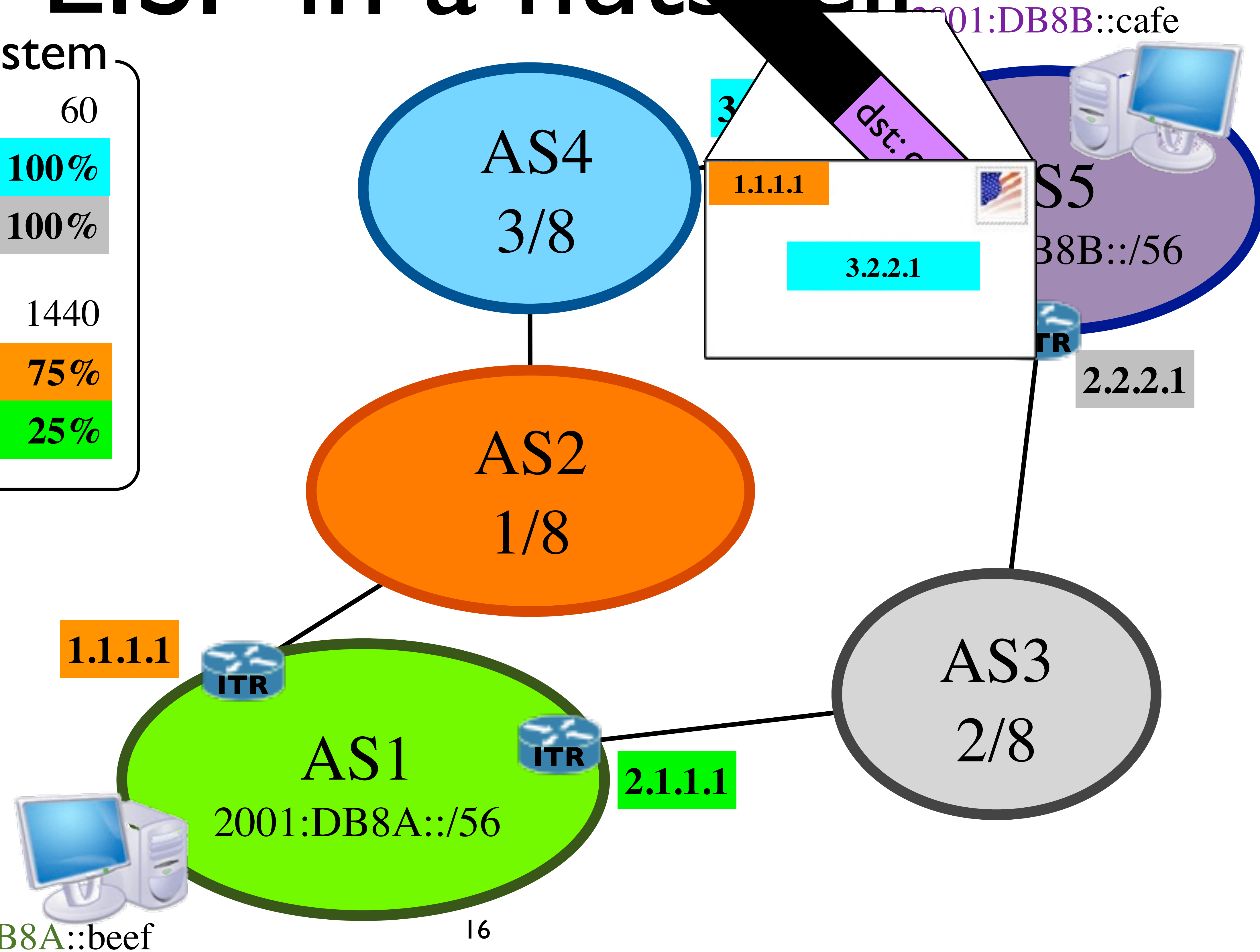
LISP in a nutshell



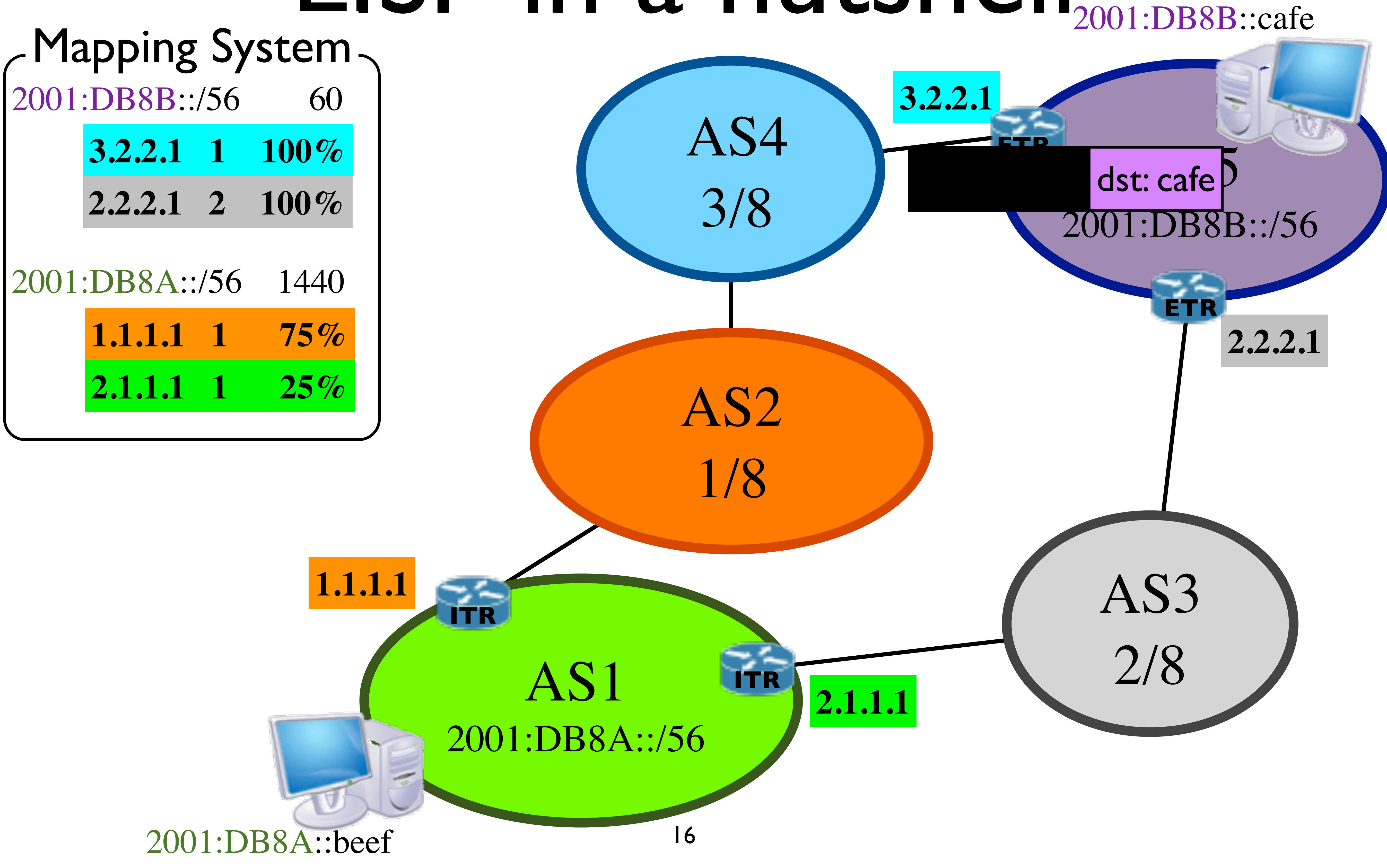
LISP in a nutshell

Mapping System

2001:DB8B::/56			60
	3.2.2.1	**1**	**100%**
	2.2.2.1	**2**	**100%**
2001:DB8A::/56			1440
	1.1.1.1	**1**	**75%**
	2.1.1.1	**1**	**25%**



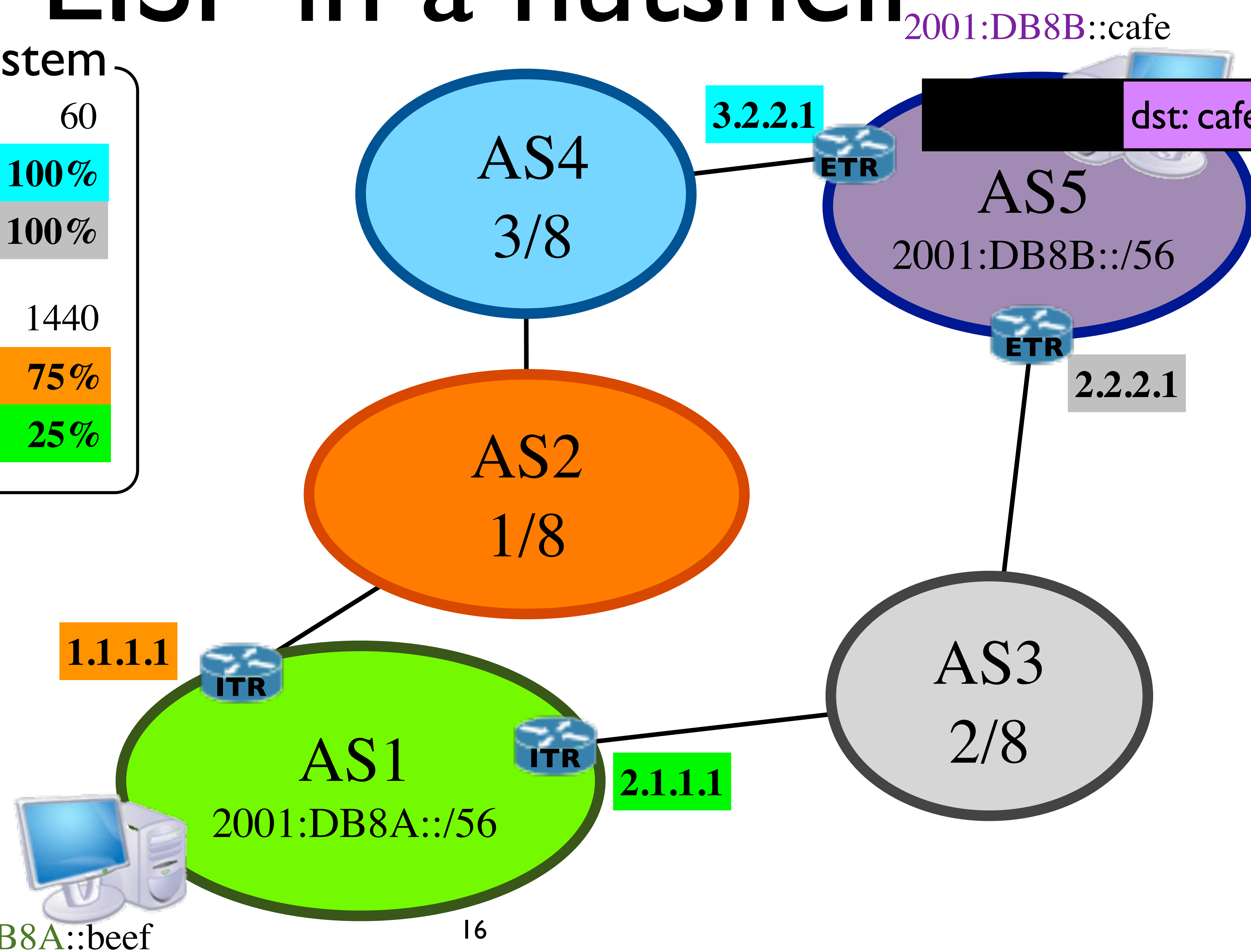
LISP in a nutshell



LISP in a nutshell

Mapping System

2001:DB8B::/56			60
	3.2.2.1	**1**	**100%**
	2.2.2.1	**2**	**100%**
2001:DB8A::/56			1440
	1.1.1.1	**1**	**75%**
	2.1.1.1	**1**	**25%**



Use and abuse of the mappings

- Incoming traffic engineering
  - adapt the priorities, weights and TTL to reflect the traffic engineering needs
- Mapping differentiation
  - provide per requester mappings (i.e.,depending on the source EID)

What should a good mapping system look like?

What should a good mapping system look like?

- A good mapping system must be

What should a good mapping system look like?

- A good mapping system must be
  - scalable

What should a good mapping system look like?

- A good mapping system must be
  - scalable
  - troubleshoot-able and tolerant to fault (isolation)

What should a good mapping system look like?

- A good mapping system must be
  - scalable
  - troubleshoot-able and tolerant to fault (isolation)
  - insensitive to configuration errors (isolation)

What should a good mapping system look like?

- A good mapping system must be
  - scalable
  - troubleshoot-able and tolerant to fault (isolation)
  - insensitive to configuration errors (isolation)
  - securable

LISP-Tree

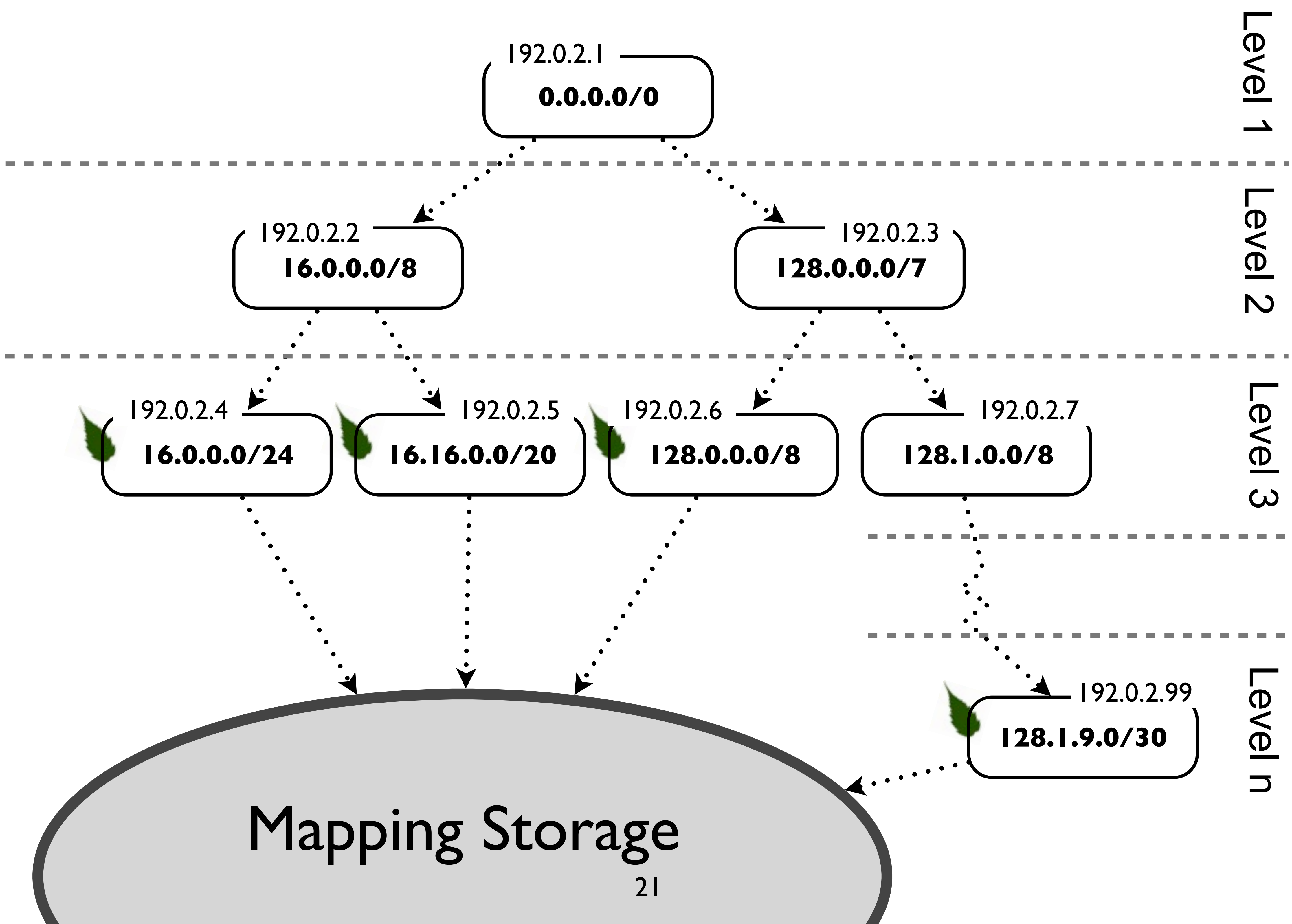
- LISP-Tree is a hierarchical mapping system
  - similar to the DNS
- Two components
  - a storage part to store the mappings
  - a discovery part to discover where the mapping are stored

[JCAC+ 10,SB11]

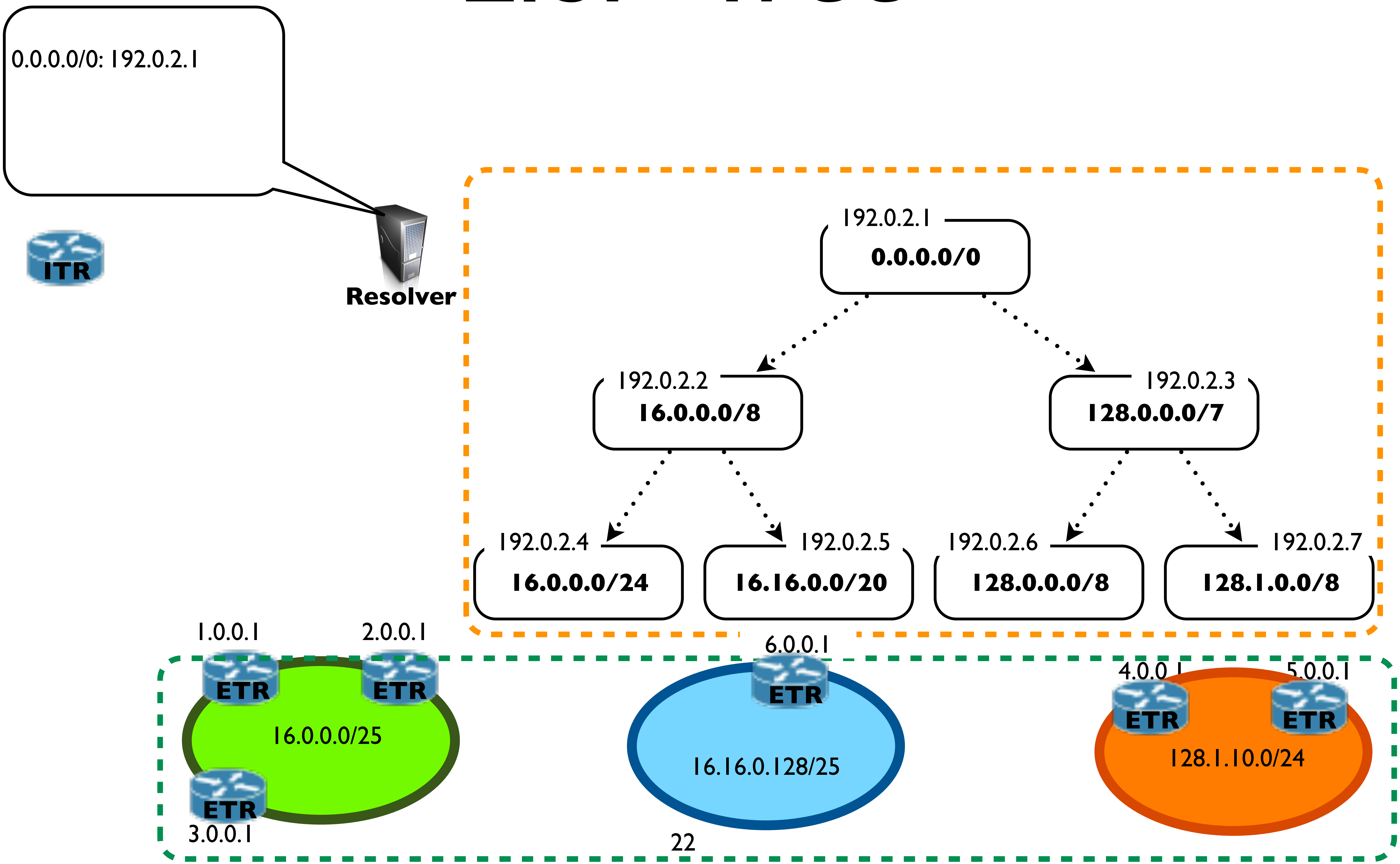
Storage

- Mappings are stored at the ETRs in their EID-to-RLOC Database
- Mappings are registered at the Map-Server
- Compatible with the LISP Specifications [FFML11, FF11]

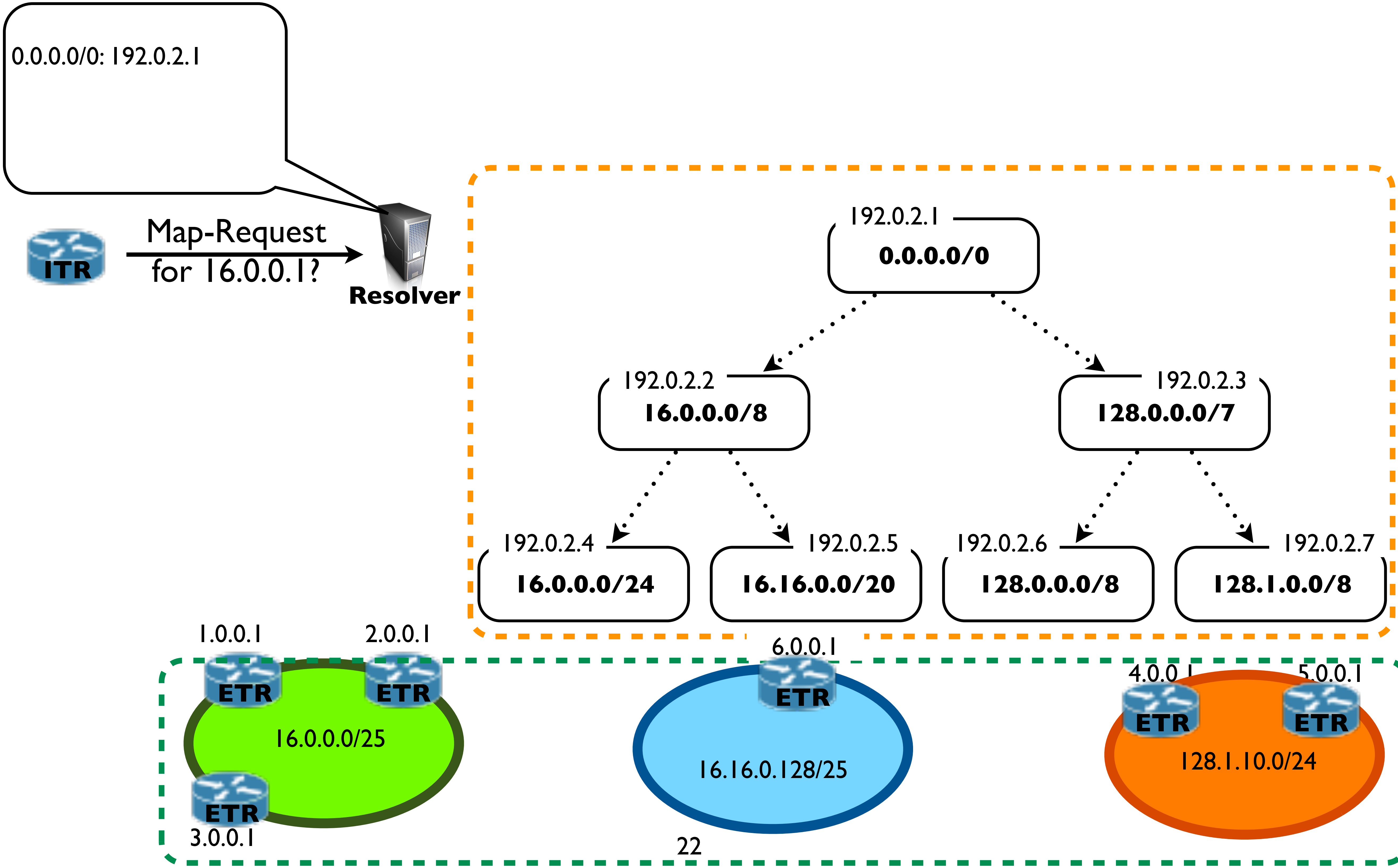
Discovery


192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
192.0.2.99
128.1.9.0/30
Mapping Storage
Level 1
Level 2
Level 3
Level n

LISP-Tree

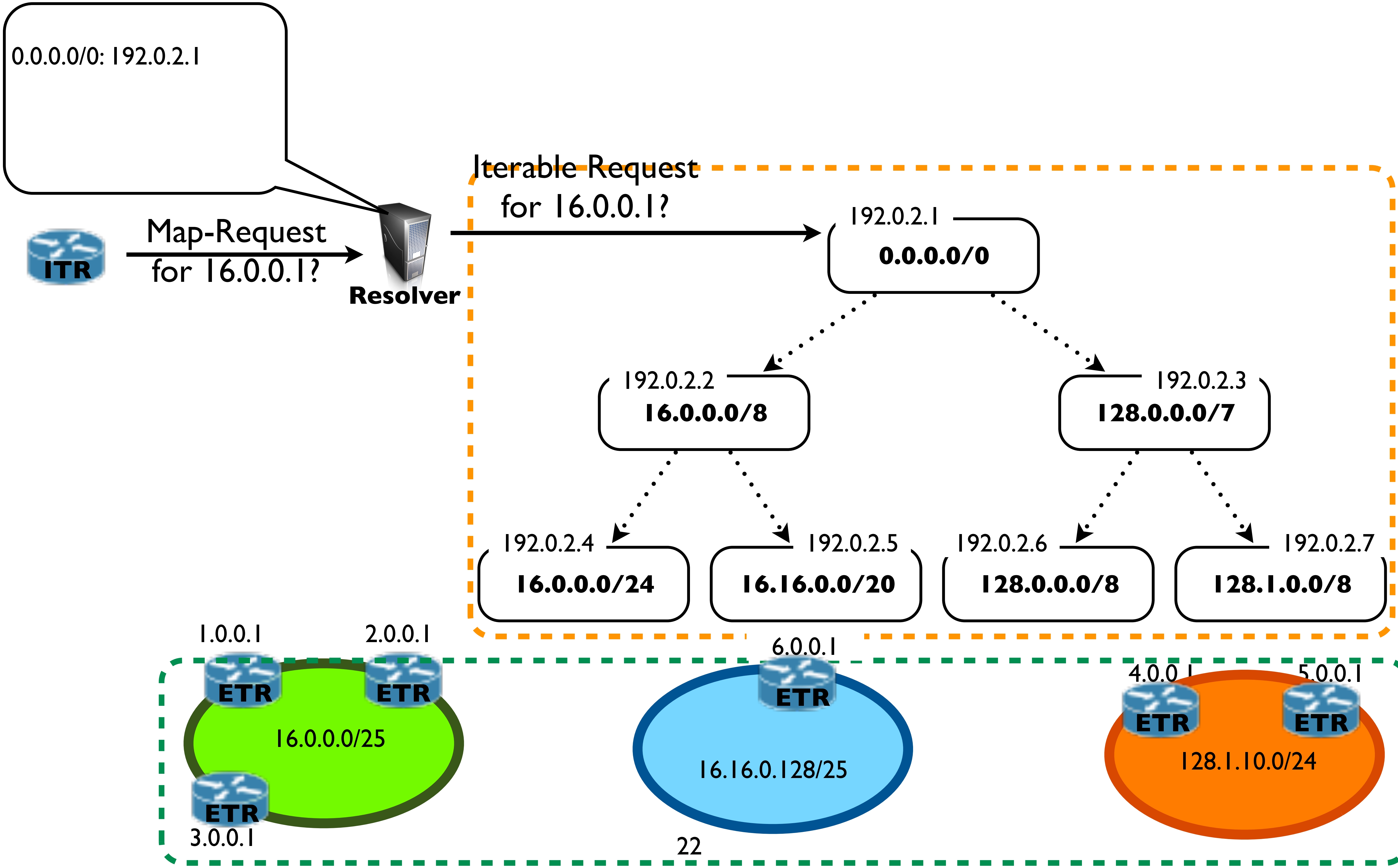

0.0.0.0/0: 192.0.2.1
ITR
Resolver
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree



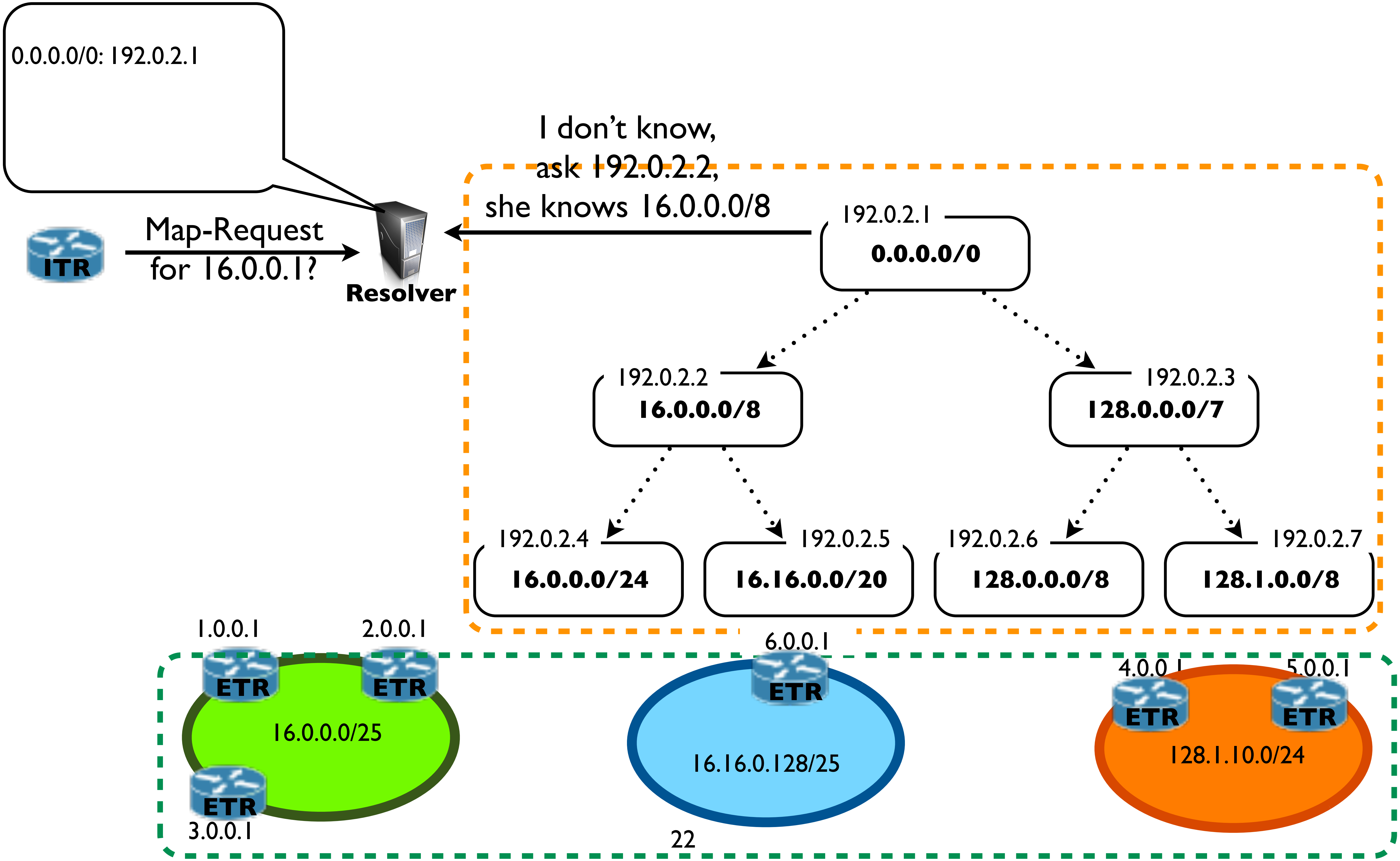
0.0.0.0/0: 192.0.2.1
ITR
Map-Request
for 16.0.0.1?
Resolver
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree

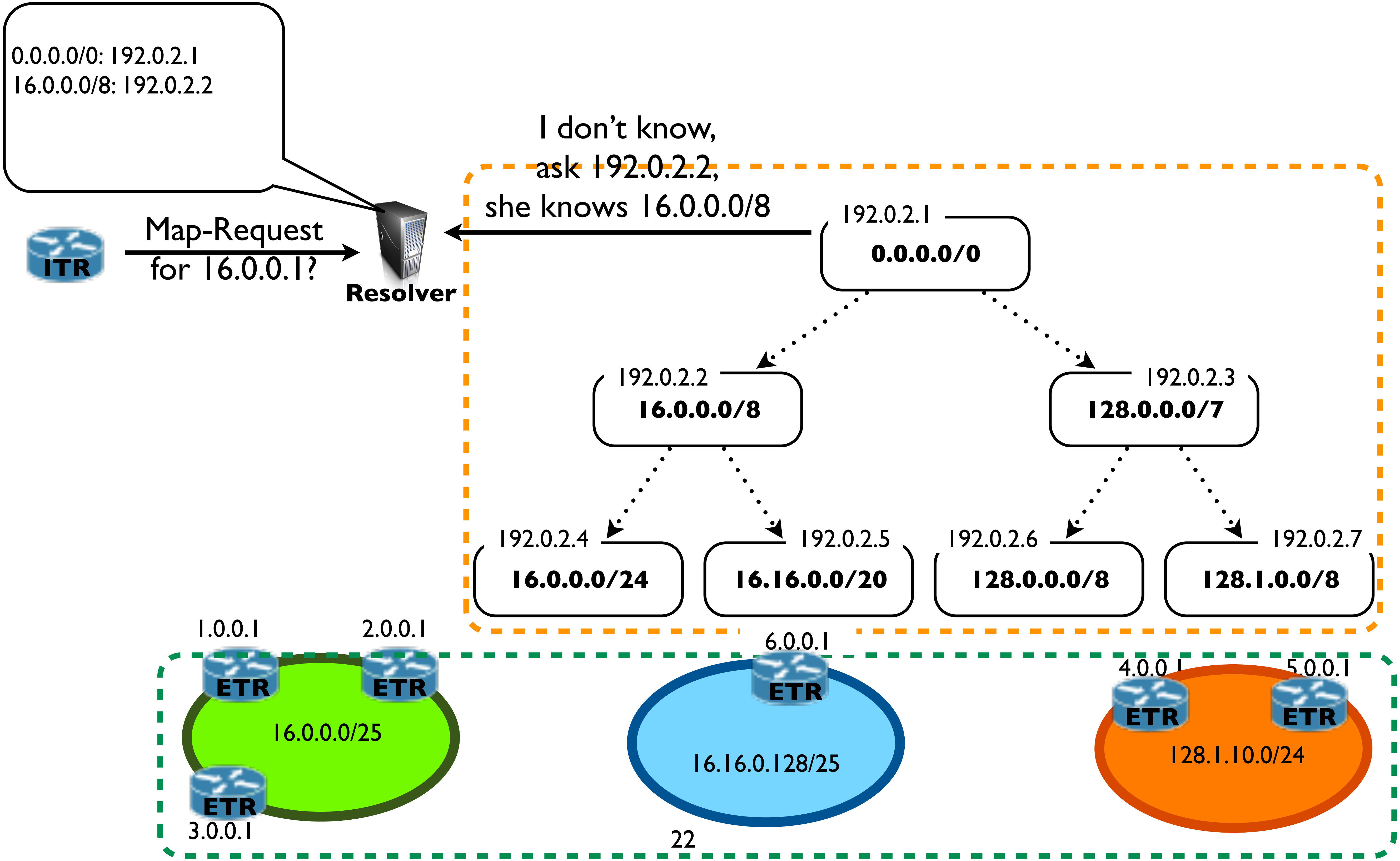


0.0.0.0/0: 192.0.2.1
ITR
Map-Request
for 16.0.0.1?
Resolver
Iterable Request
for 16.0.0.1?
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree

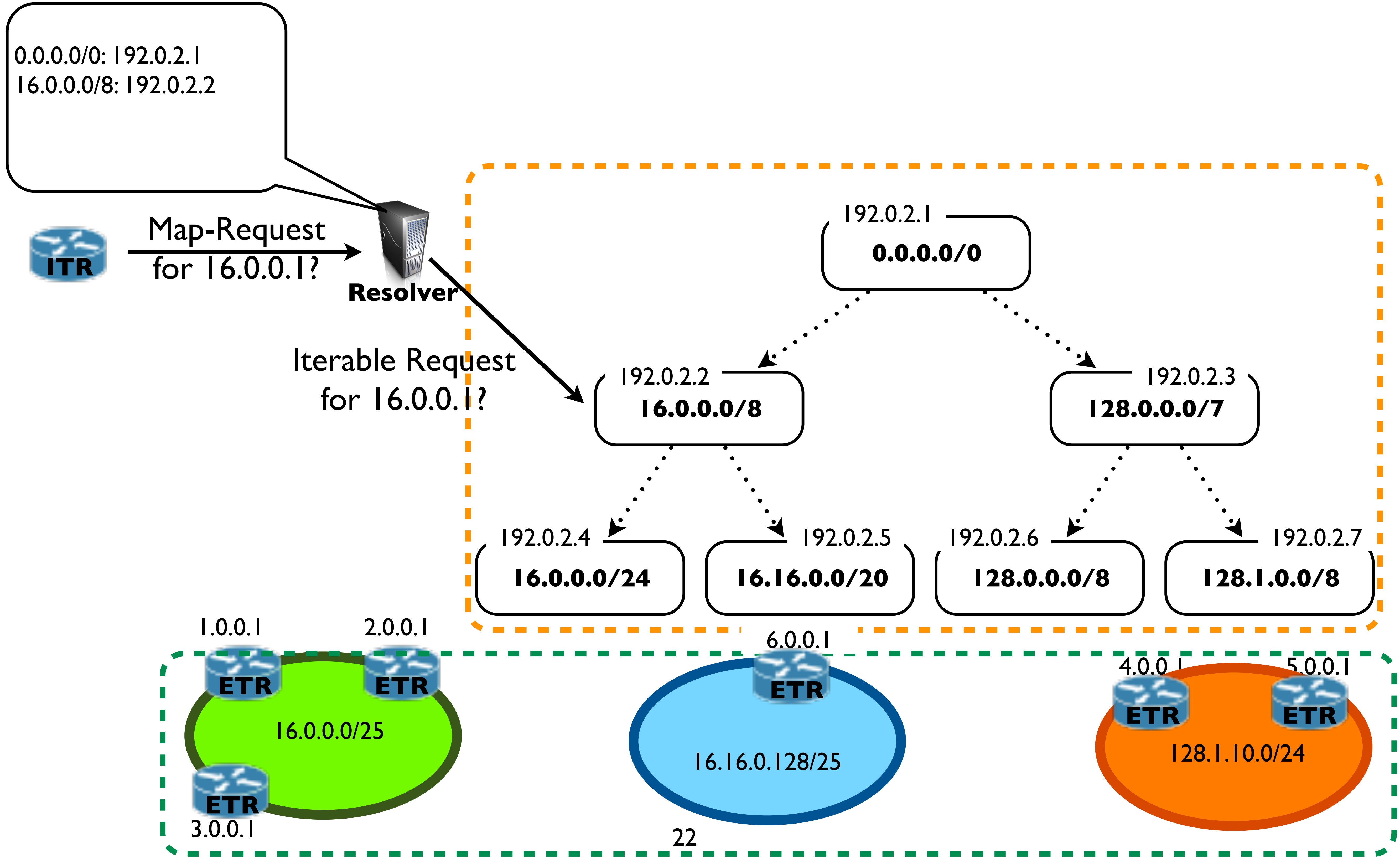

0.0.0.0/0: 192.0.2.1
I don't know,
ask 192.0.2.2,
she knows 16.0.0.0/8
ITR
Map-Request
for 16.0.0.1?
Resolver
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree

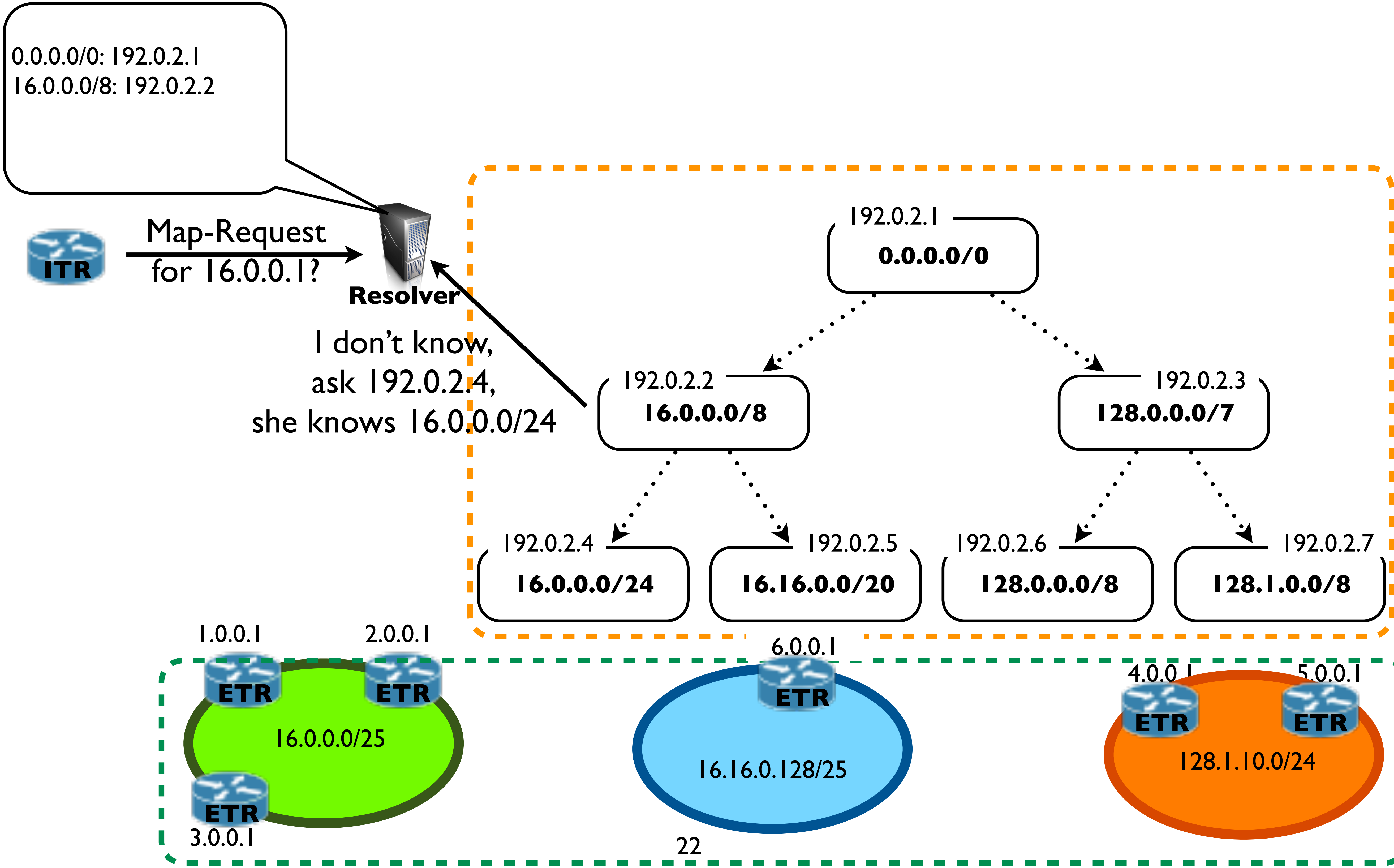


0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
I don't know,
ask 192.0.2.2,
she knows 16.0.0.0/8
Map-Request
for 16.0.0.1?
ITR
Resolver
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
3.0.0.1
ETR
ETR
ETR
16.0.0.0/25
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree

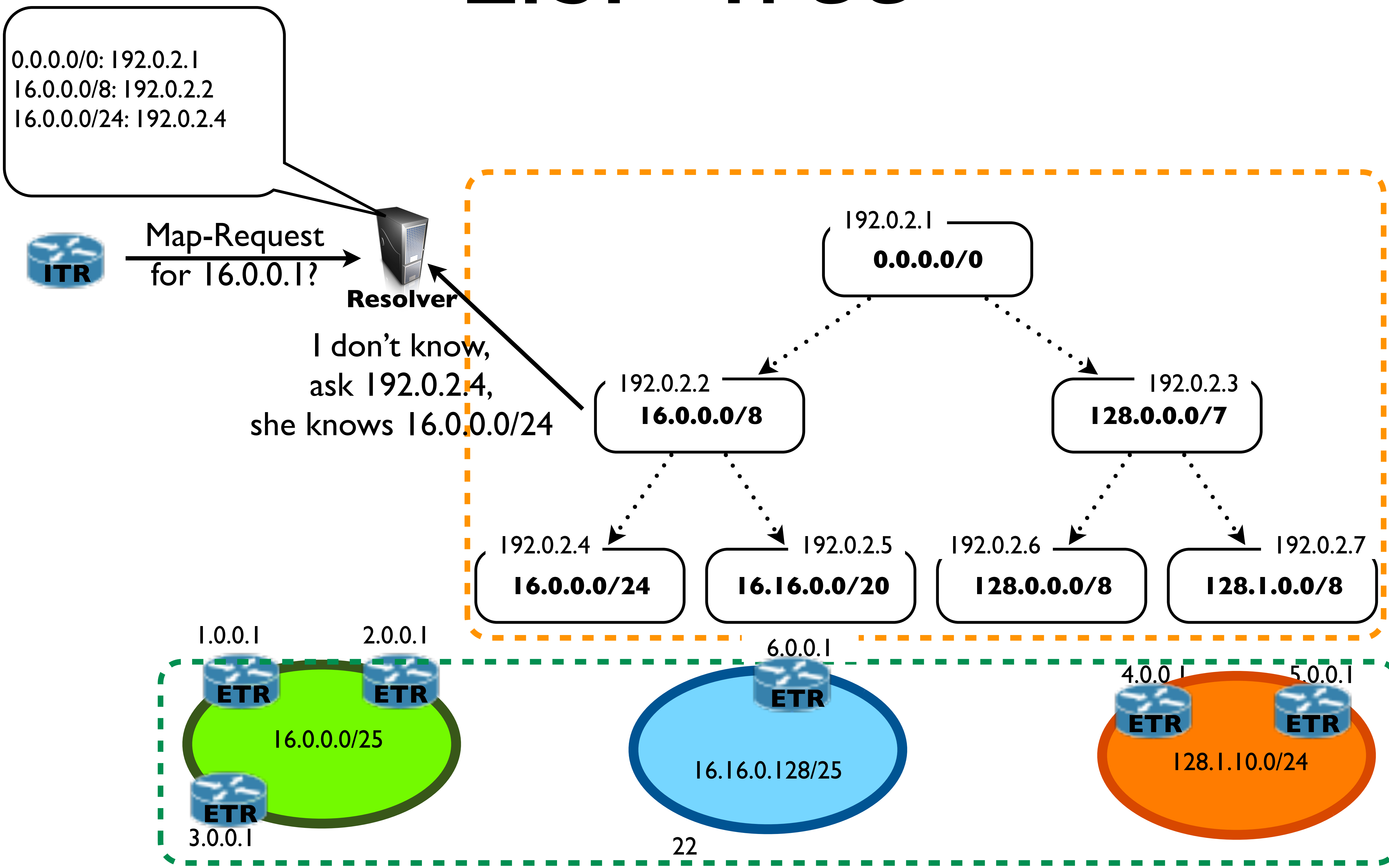

0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
ITR
Map-Request
for 16.0.0.1?
Resolver
Iterable Request
for 16.0.0.1?
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree

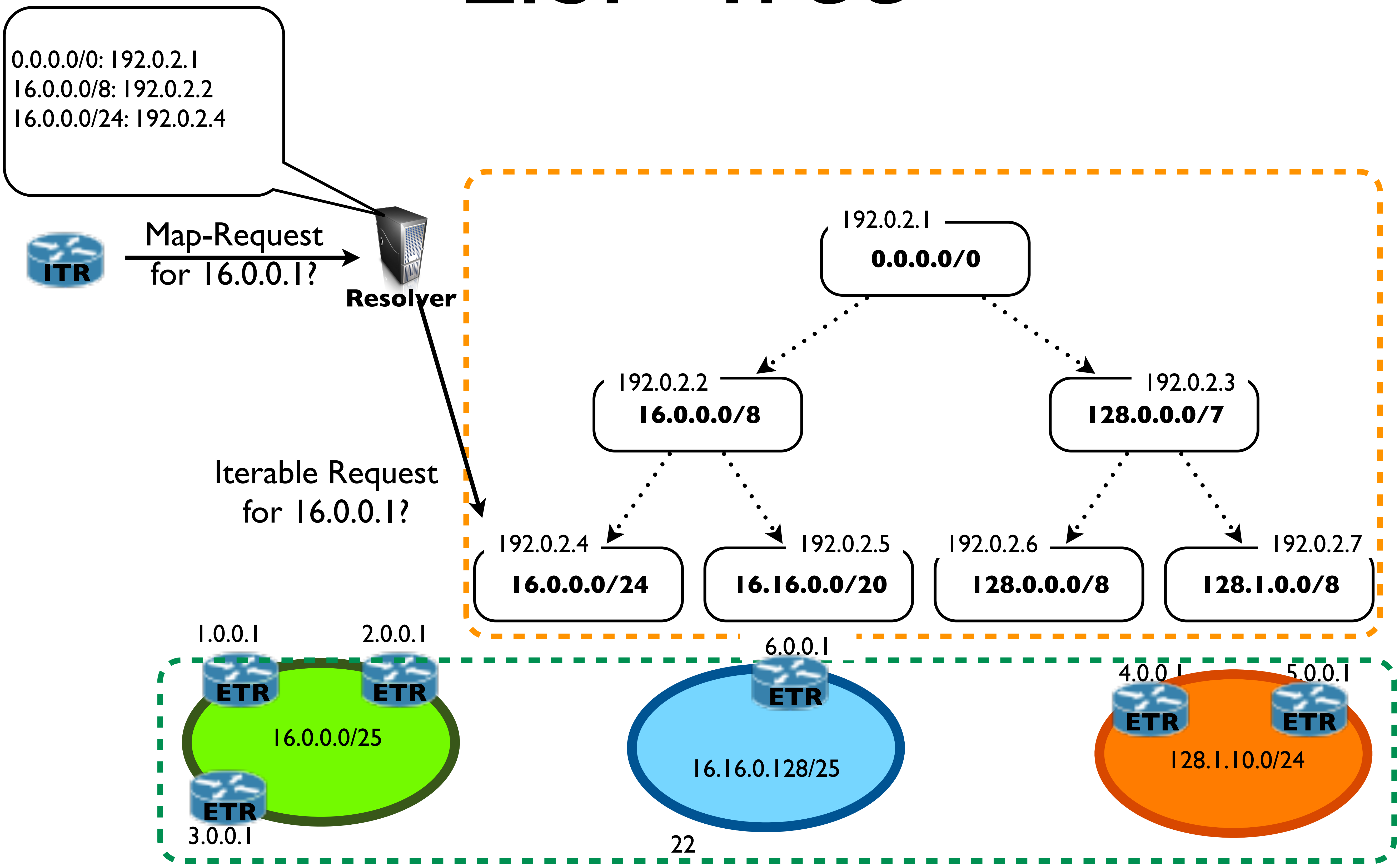


0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
ITR
Map-Request
for 16.0.0.1?
Resolver
I don't know,
ask 192.0.2.4,
she knows 16.0.0.0/24
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree

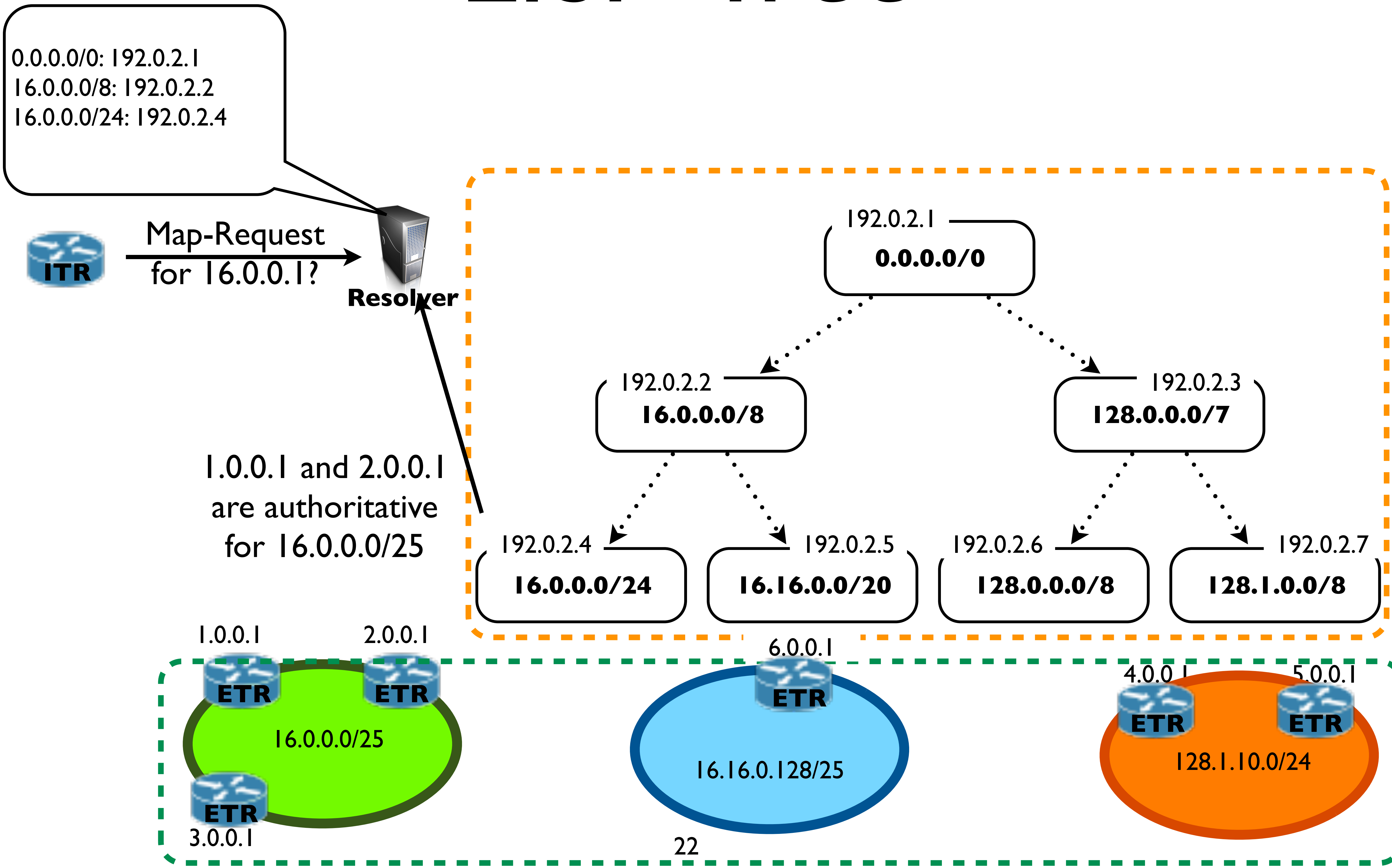

0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
ITR
Map-Request
for 16.0.0.1?
Resolver
I don't know,
ask 192.0.2.4,
she knows 16.0.0.0/24
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree

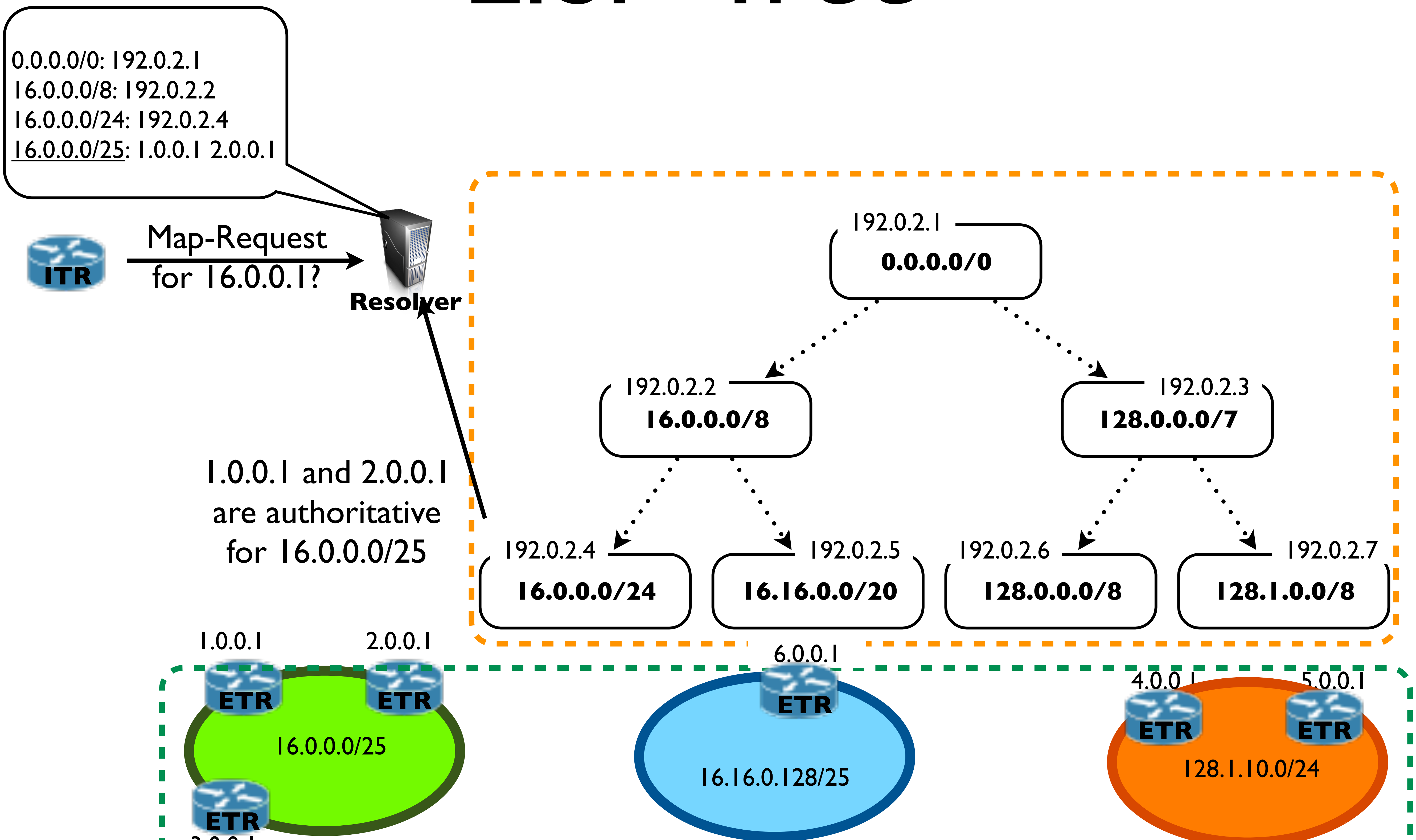


0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
ITR
Map-Request for 16.0.0.1?
Resolver
Iterable Request for 16.0.0.1?
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

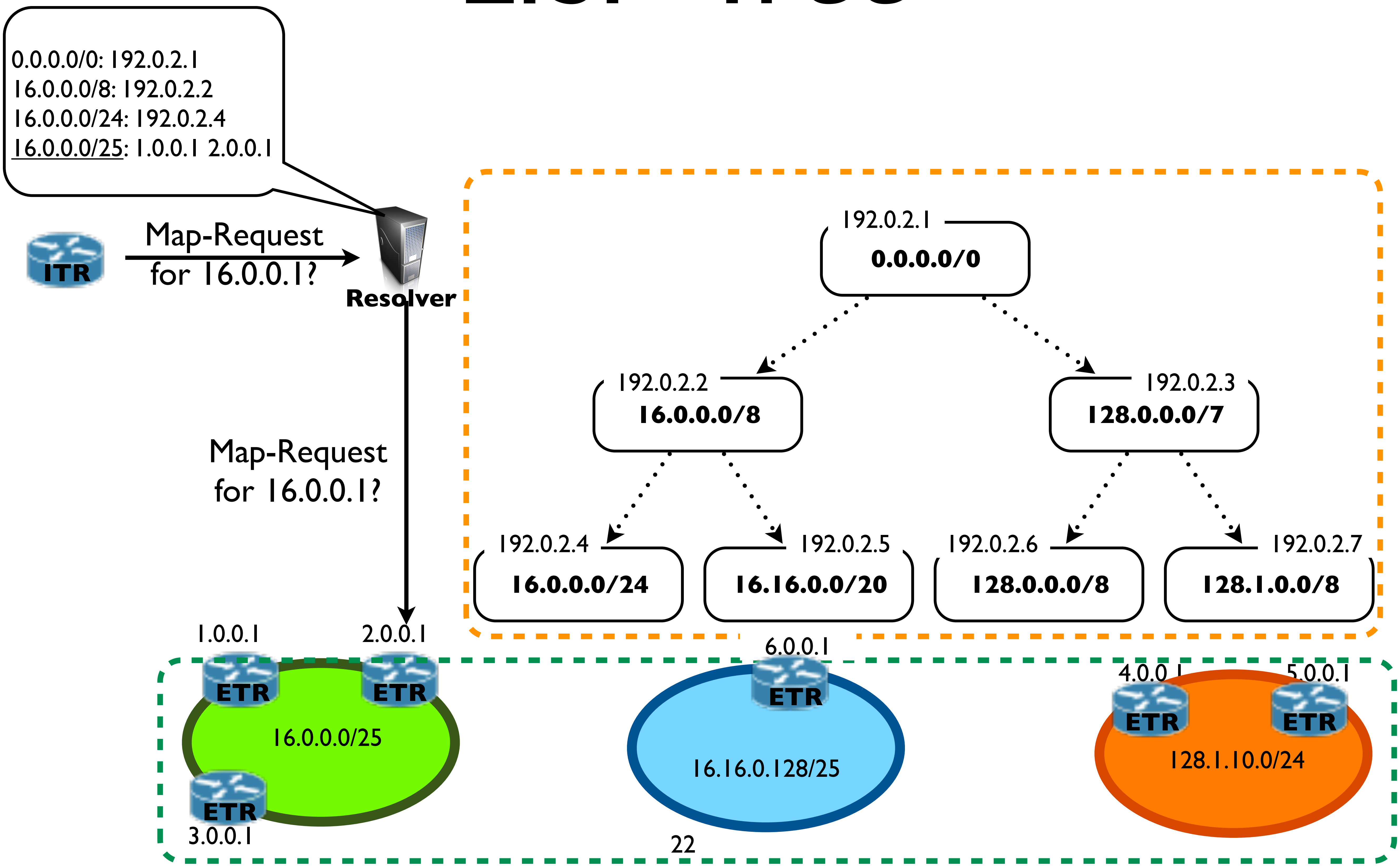
LISP-Tree


0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
ITR
Map-Request
for 16.0.0.1?
Resolver
1.0.0.1 and 2.0.0.1
are authoritative
for 16.0.0.0/25
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

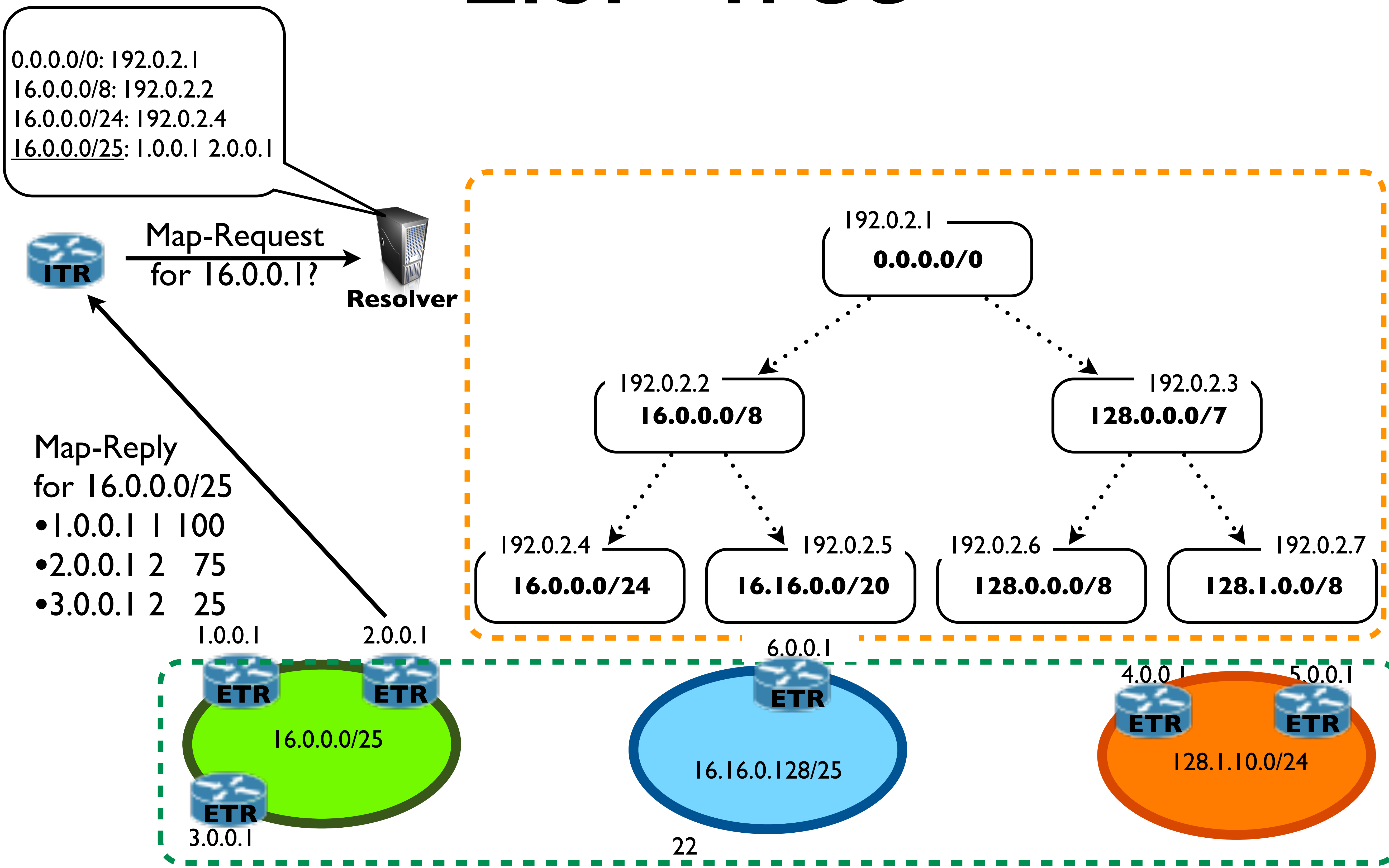
LISP-Tree


0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
16.0.0.0/25: 1.0.0.1 2.0.0.1
ITR
Map-Request
for 16.0.0.1?
Resolver
1.0.0.1 and 2.0.0.1
are authoritative
for 16.0.0.0/25
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree

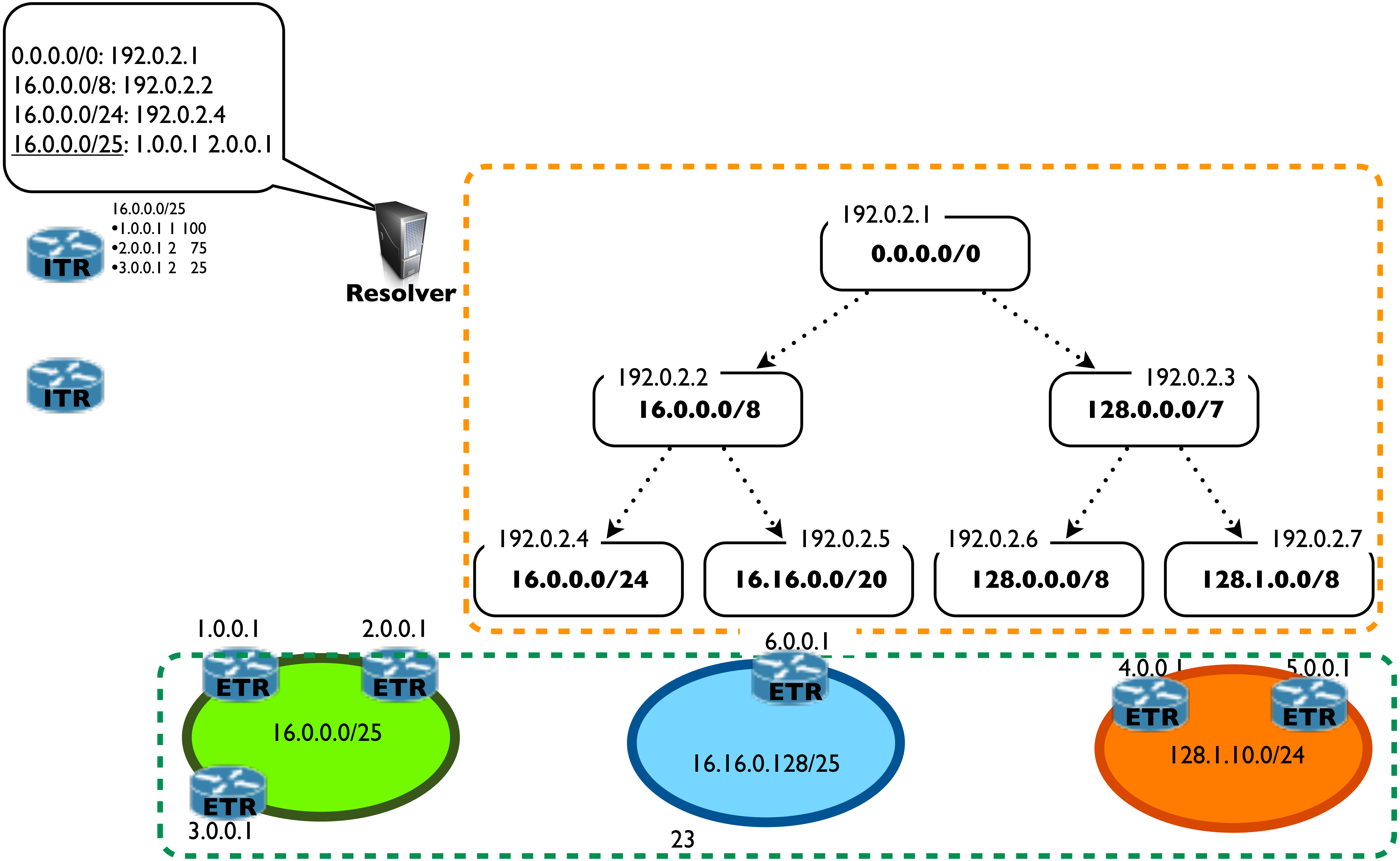

0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
16.0.0.0/25: 1.0.0.1 2.0.0.1
ITR
Map-Request
for 16.0.0.1?
Resolver
Map-Request
for 16.0.0.1?
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree



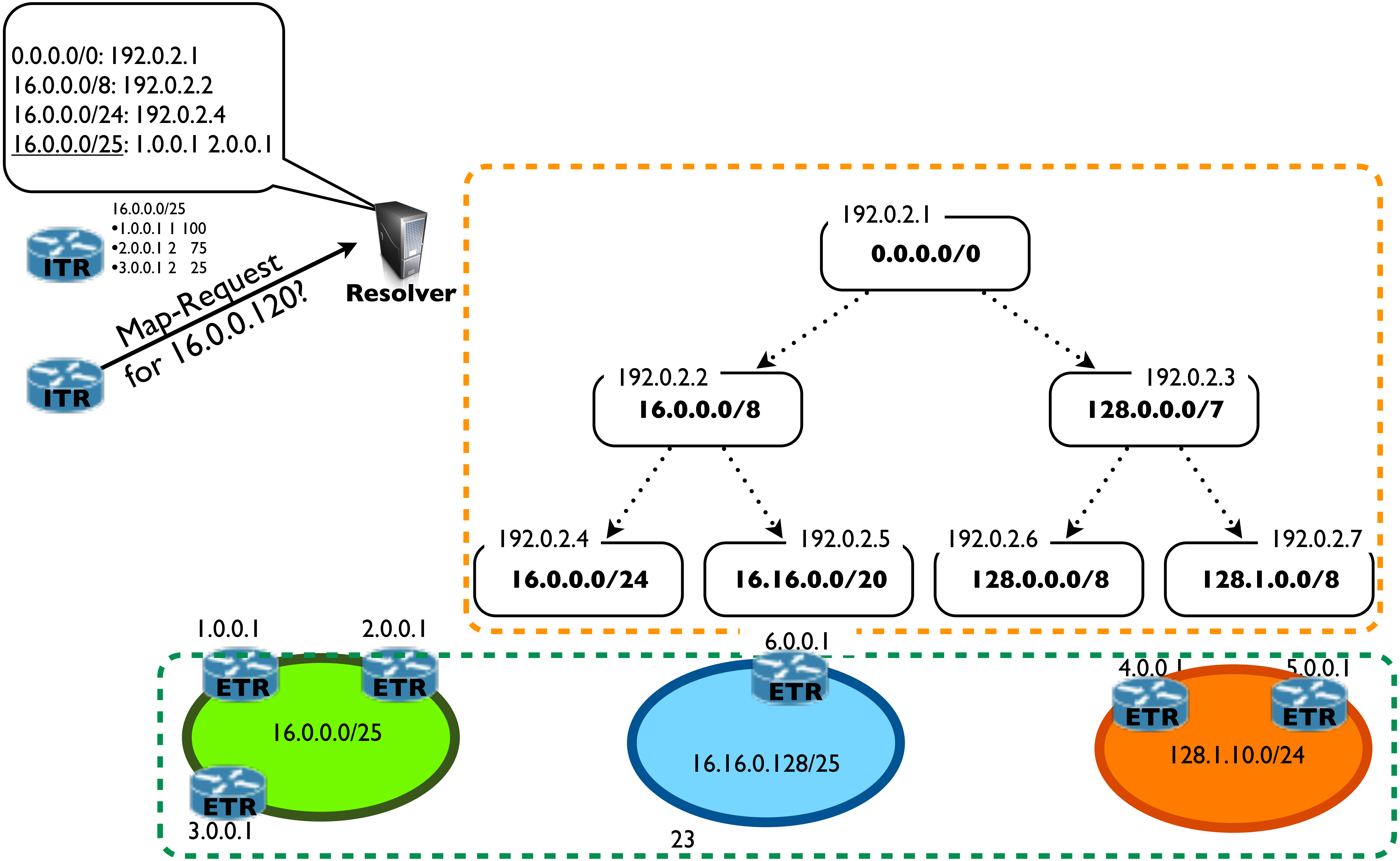
0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
16.0.0.0/25: 1.0.0.1 2.0.0.1
ITR
Map-Request
for 16.0.0.1?
Resolver
Map-Reply
for 16.0.0.0/25
•1.0.0.1 1 100
•2.0.0.1 2 75
•3.0.0.1 2 25
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree



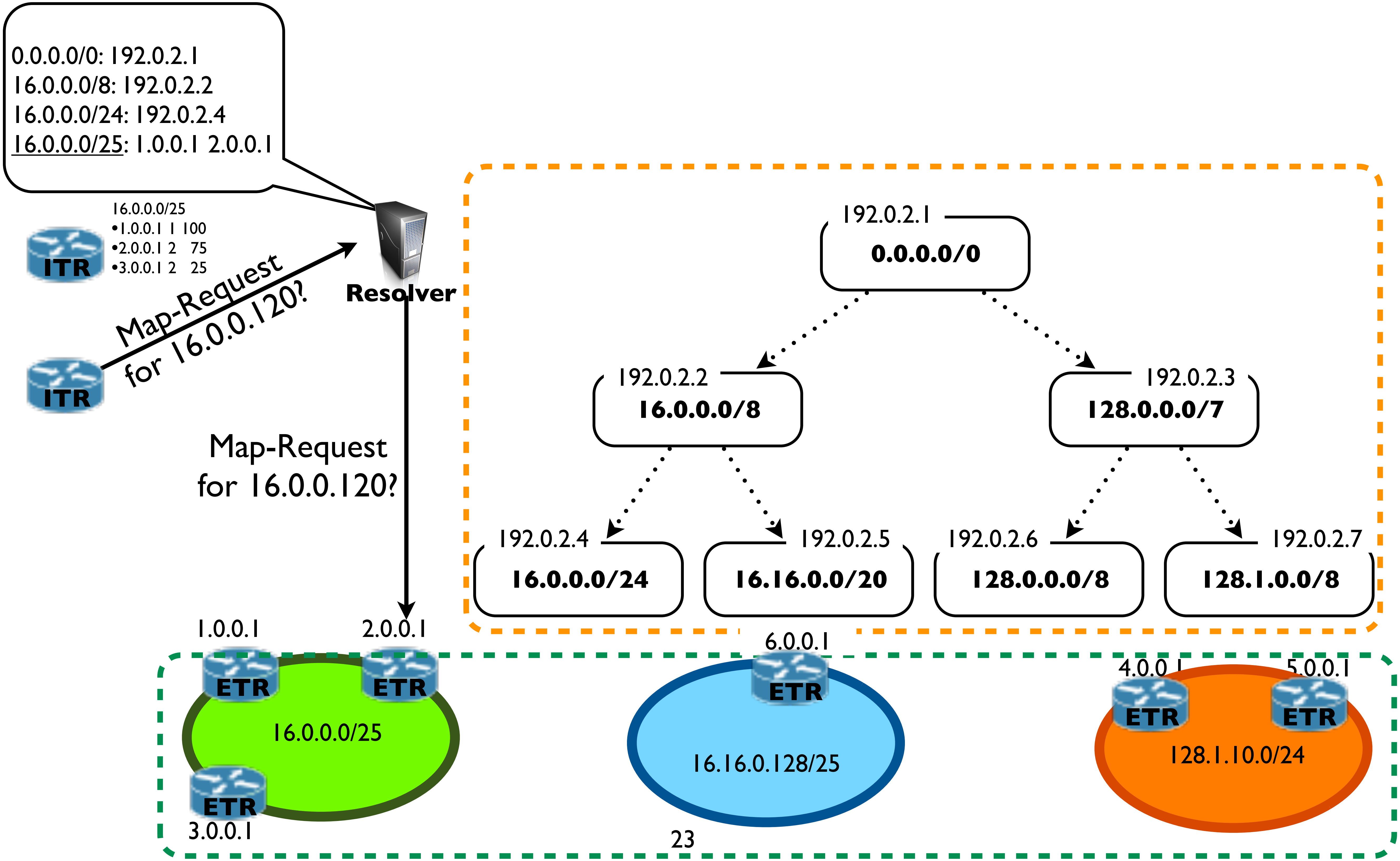
0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
16.0.0.0/25: 1.0.0.1 2.0.0.1
16.0.0.0/25
•1.0.0.1 1 100
•2.0.0.1 2 75
•3.0.0.1 2 25
ITR
ITR
Resolver
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree



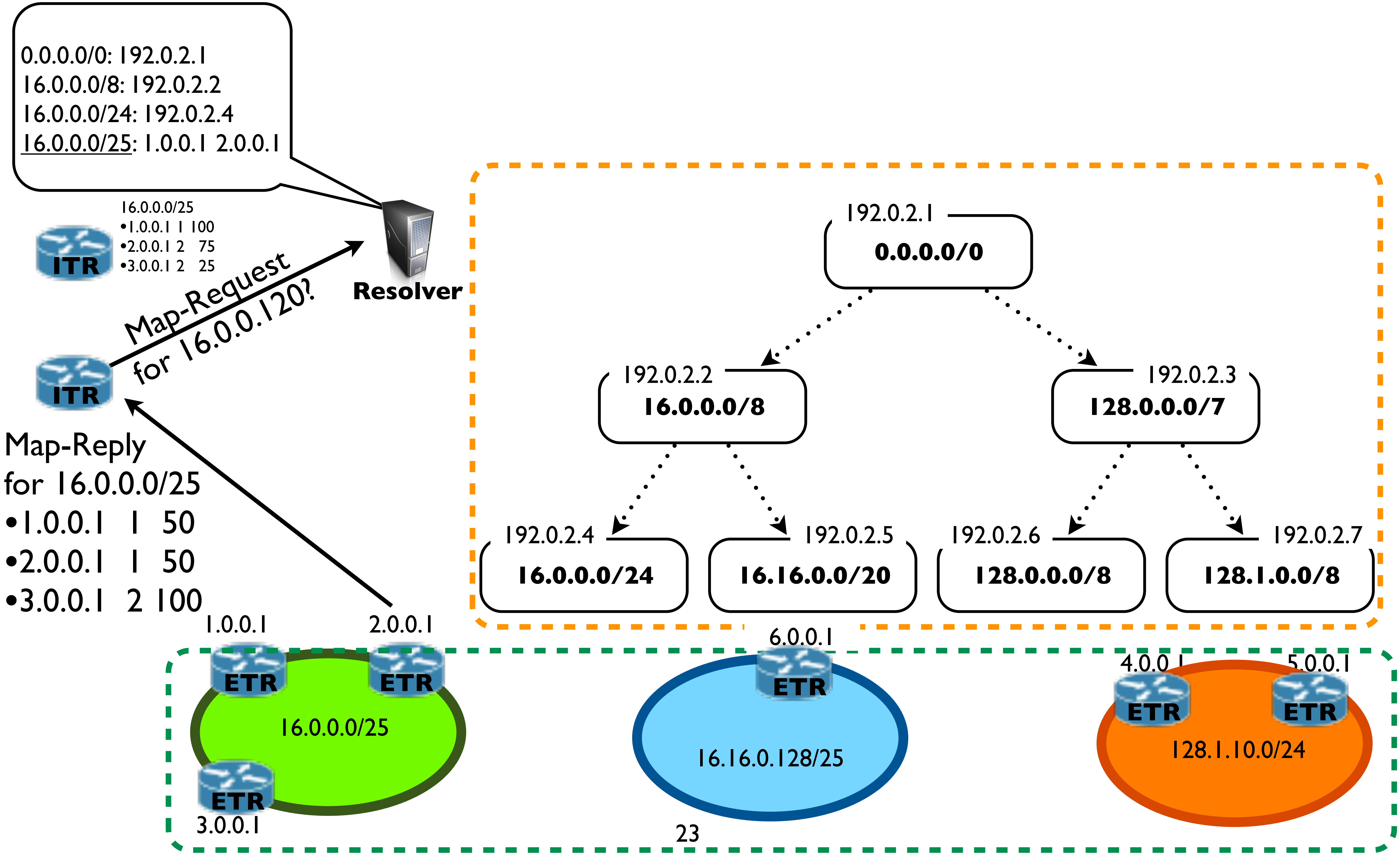
0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
16.0.0.0/25: 1.0.0.1 2.0.0.1
16.0.0.0/25
•1.0.0.1 1 100
•2.0.0.1 2 75
•3.0.0.1 2 25
ITR
ITR
Map-Request for 16.0.0.120?
Resolver
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

LISP-Tree



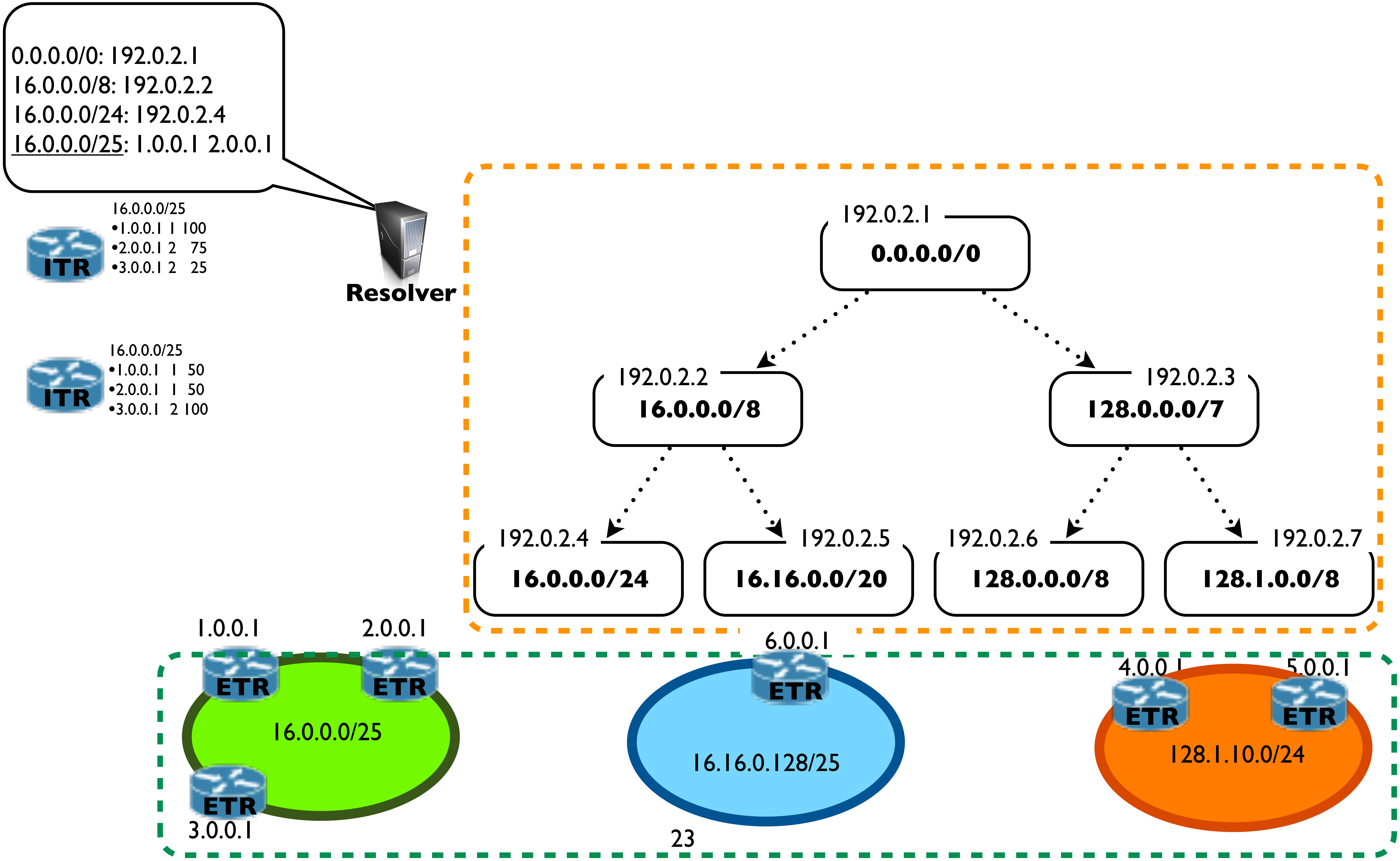
0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
16.0.0.0/25: 1.0.0.1 2.0.0.1
16.0.0.0/25
•1.0.0.1 1 100
•2.0.0.1 2 75
•3.0.0.1 2 25
ITR
ITR
Map-Request for 16.0.0.120?
Resolver
Map-Request for 16.0.0.120?
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
6.0.0.1
4.0.0.1
5.0.0.1
ETR
ETR
ETR
ETR
ETR
ETR
16.0.0.0/25
16.16.0.128/25
128.1.10.0/24
3.0.0.1

LISP-Tree



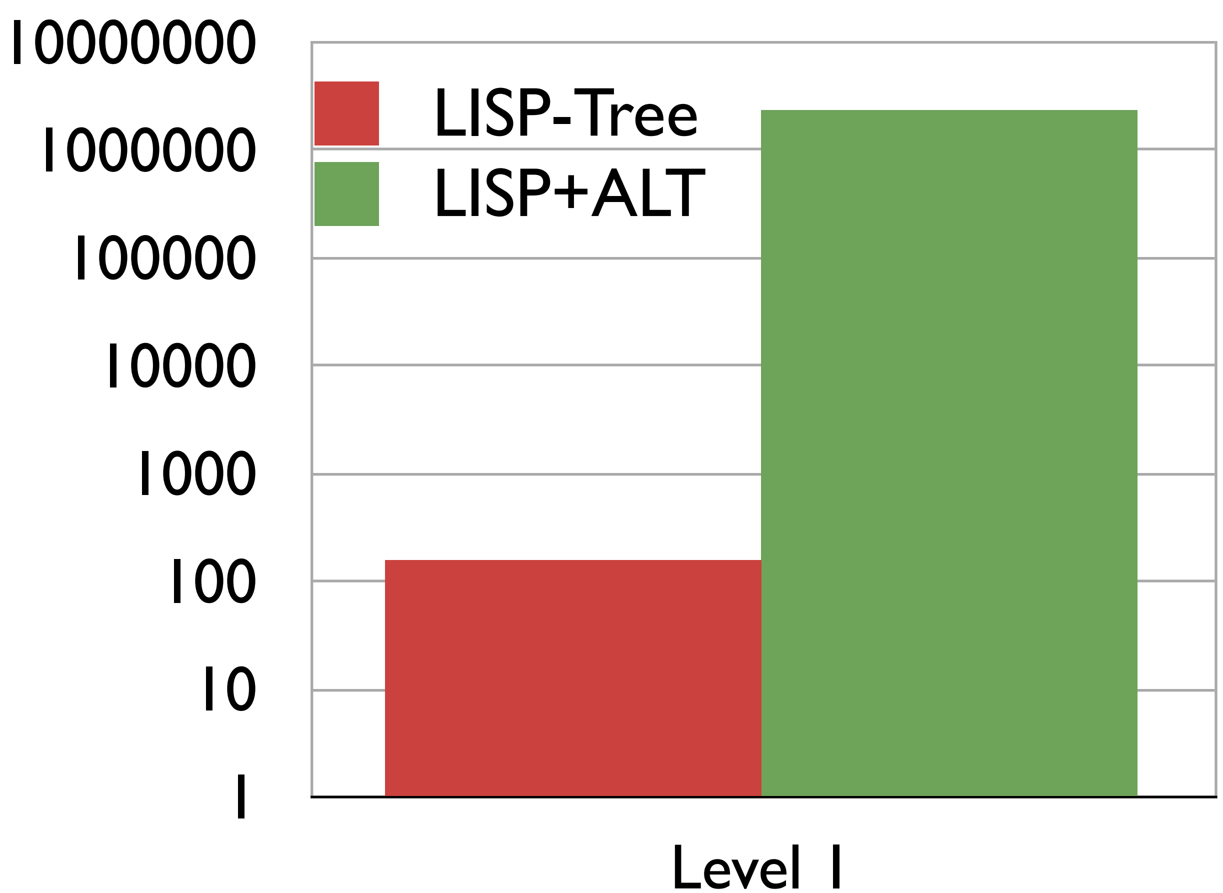
0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
16.0.0.0/25: 1.0.0.1 2.0.0.1
16.0.0.0/25
•1.0.0.1 1 100
•2.0.0.1 2 75
•3.0.0.1 2 25
ITR
Resolver
Map-Request for 16.0.0.120?
ITR
Map-Reply for 16.0.0.0/25
•1.0.0.1 1 50
•2.0.0.1 1 50
•3.0.0.1 2 100
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
6.0.0.1
4.0.0.1
5.0.0.1
ETR
ETR
ETR
ETR
ETR
ETR
16.0.0.0/25
16.16.0.128/25
128.1.10.0/24
3.0.0.1

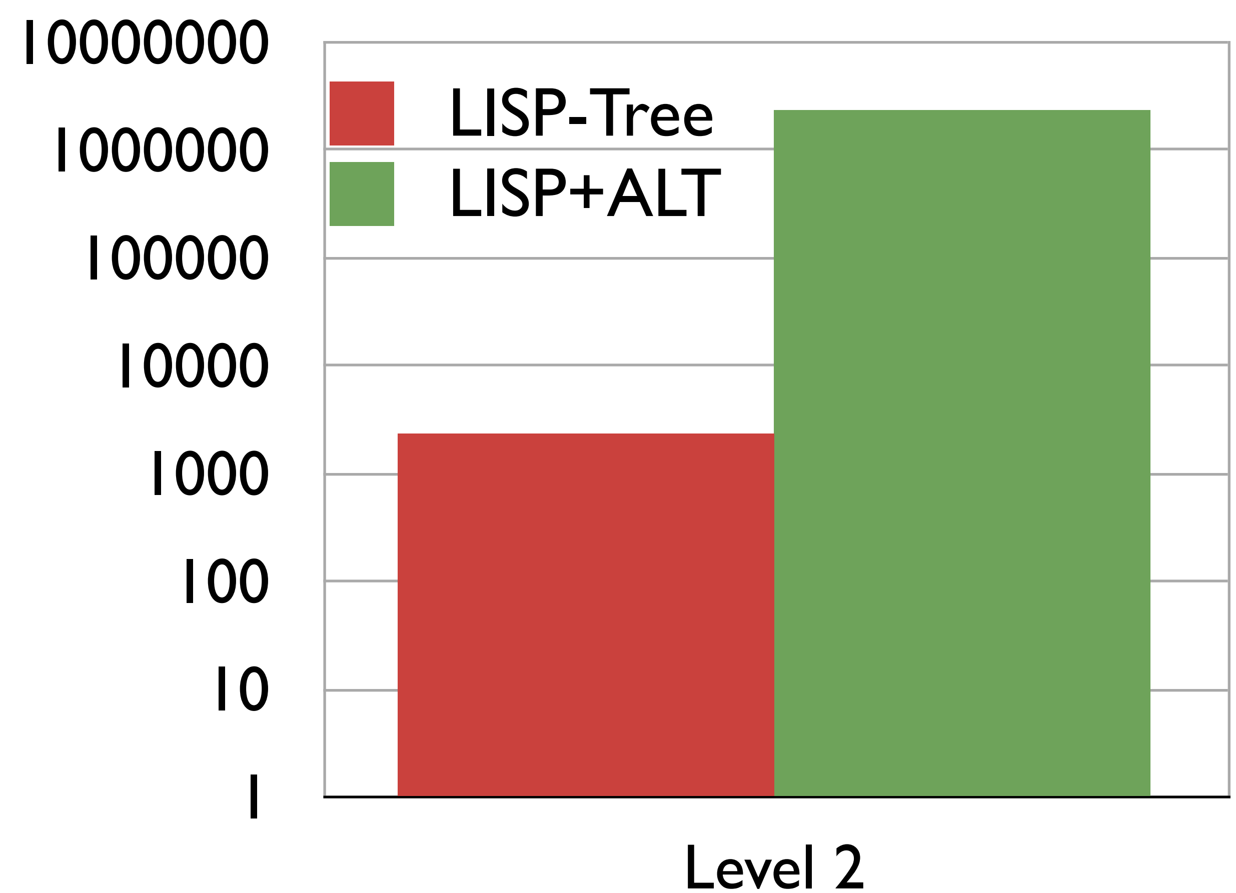
LISP-Tree



0.0.0.0/0: 192.0.2.1
16.0.0.0/8: 192.0.2.2
16.0.0.0/24: 192.0.2.4
16.0.0.0/25: 1.0.0.1 2.0.0.1
16.0.0.0/25
•1.0.0.1 1 100
•2.0.0.1 2 75
•3.0.0.1 2 25
ITR
Resolver
16.0.0.0/25
•1.0.0.1 1 50
•2.0.0.1 1 50
•3.0.0.1 2 100
ITR
192.0.2.1
0.0.0.0/0
192.0.2.2
16.0.0.0/8
192.0.2.3
128.0.0.0/7
192.0.2.4
16.0.0.0/24
192.0.2.5
16.16.0.0/20
192.0.2.6
128.0.0.0/8
192.0.2.7
128.1.0.0/8
1.0.0.1
2.0.0.1
ETR
ETR
16.0.0.0/25
ETR
3.0.0.1
6.0.0.1
ETR
16.16.0.128/25
4.0.0.1
5.0.0.1
ETR
ETR
128.1.10.0/24

Caching makes LISP-Tree scalable





- The higher a LISP-Tree server is in the hierarchy, the less frequently it is queried by a resolver

LISP summary

- LISP relies on the Map-and-Encap mechanism
  - the mappings are used to control the incoming traffic
- LISP-Tree is a scalable mapping system

➡LISP + LISP-Tree make efficient incoming traffic engineering possible

Path Ranking with IDIPS

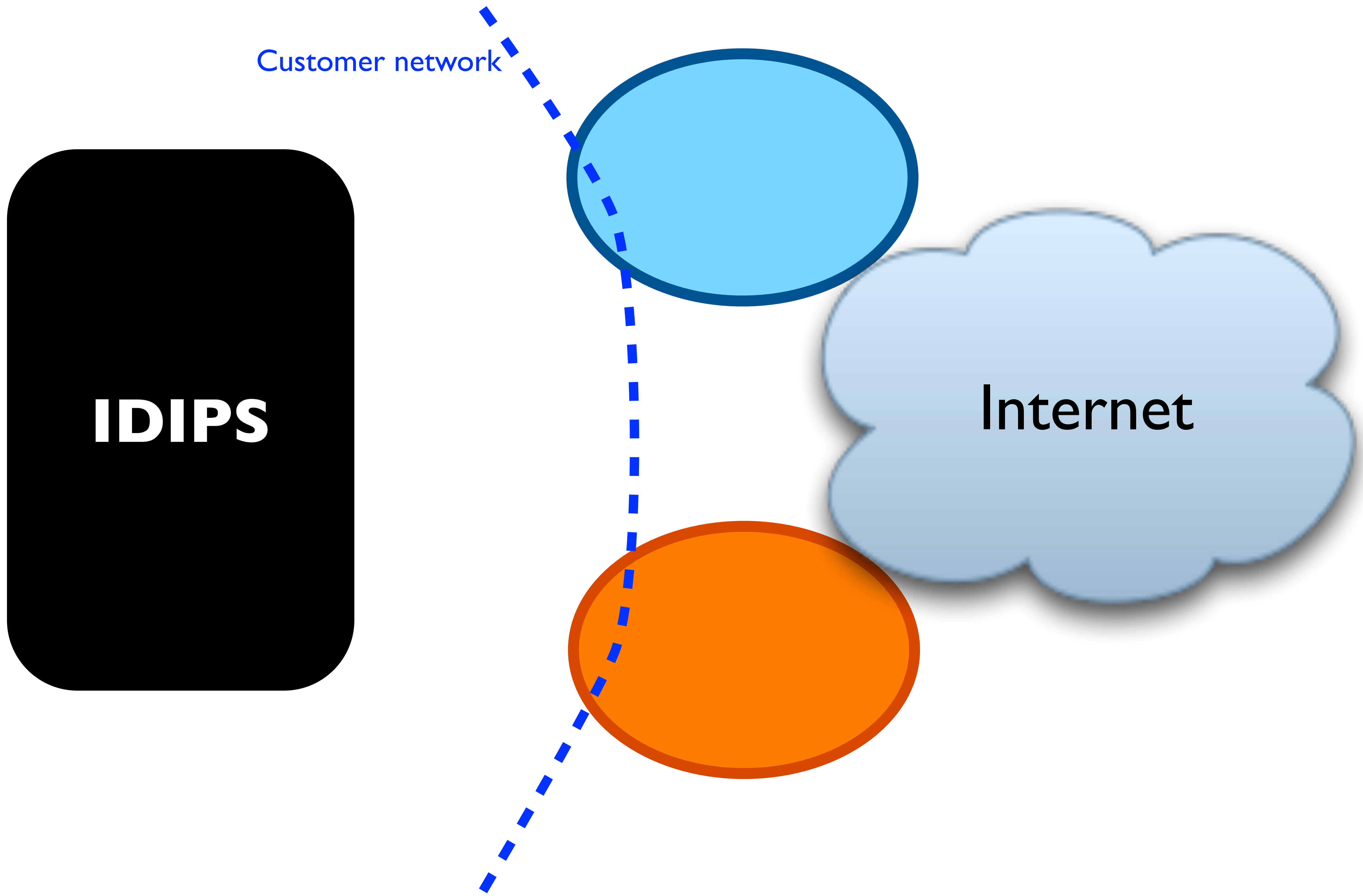
Motivation

- How to determine if a path is better than another?
- The solution must
  - be flexible (i.e., work for any definition of best)
  - not reveal topology or internal policies
  - be scalable

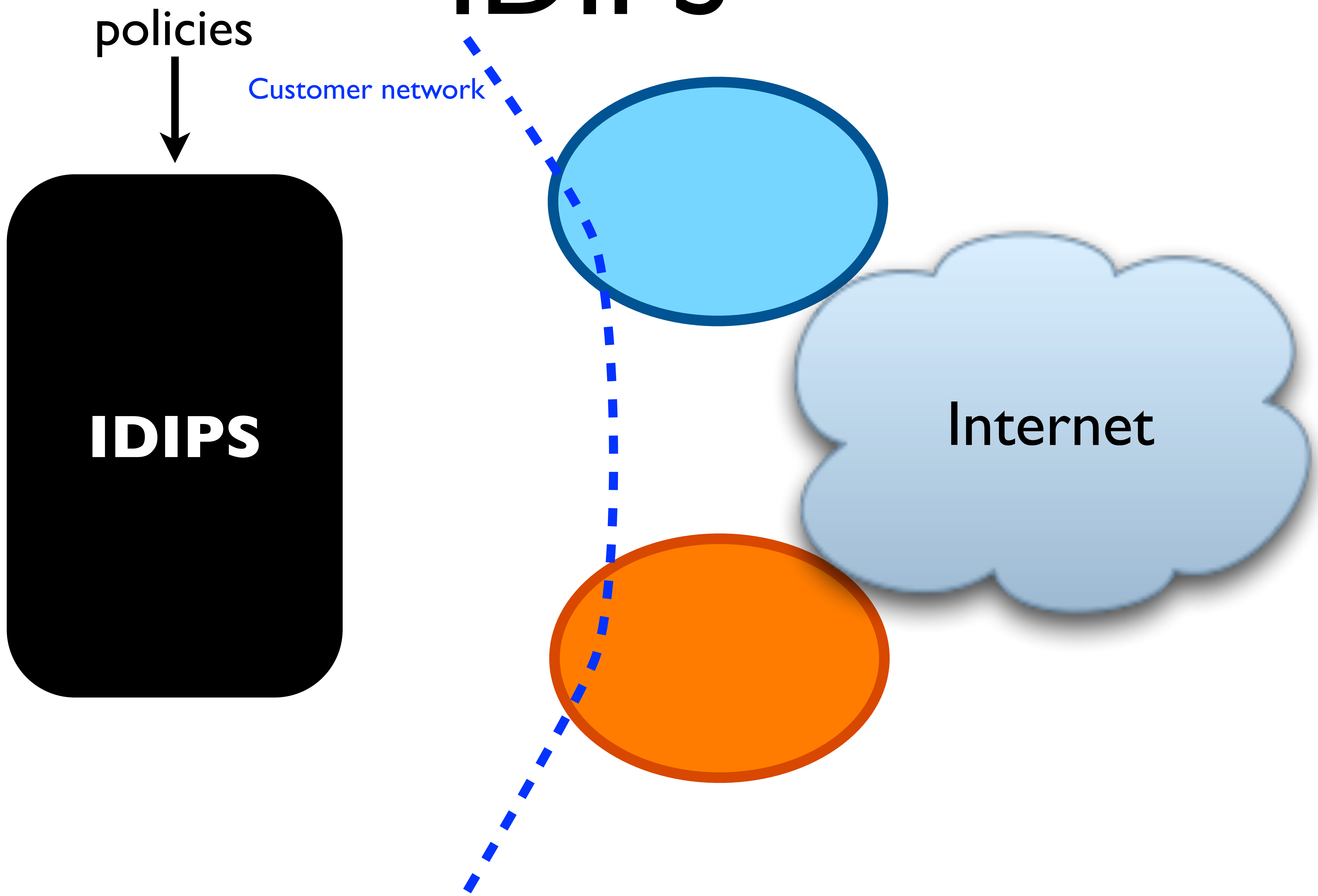
Approach

- We are interested in finding the most effective path, not by its "absolute" performance
- We propose IDIPS (ISP-Driven Informed Path Selection), a tool that
  - ranks the paths to highlight the most efficient
  - measures path performance on behalf of its clients in a scalable way

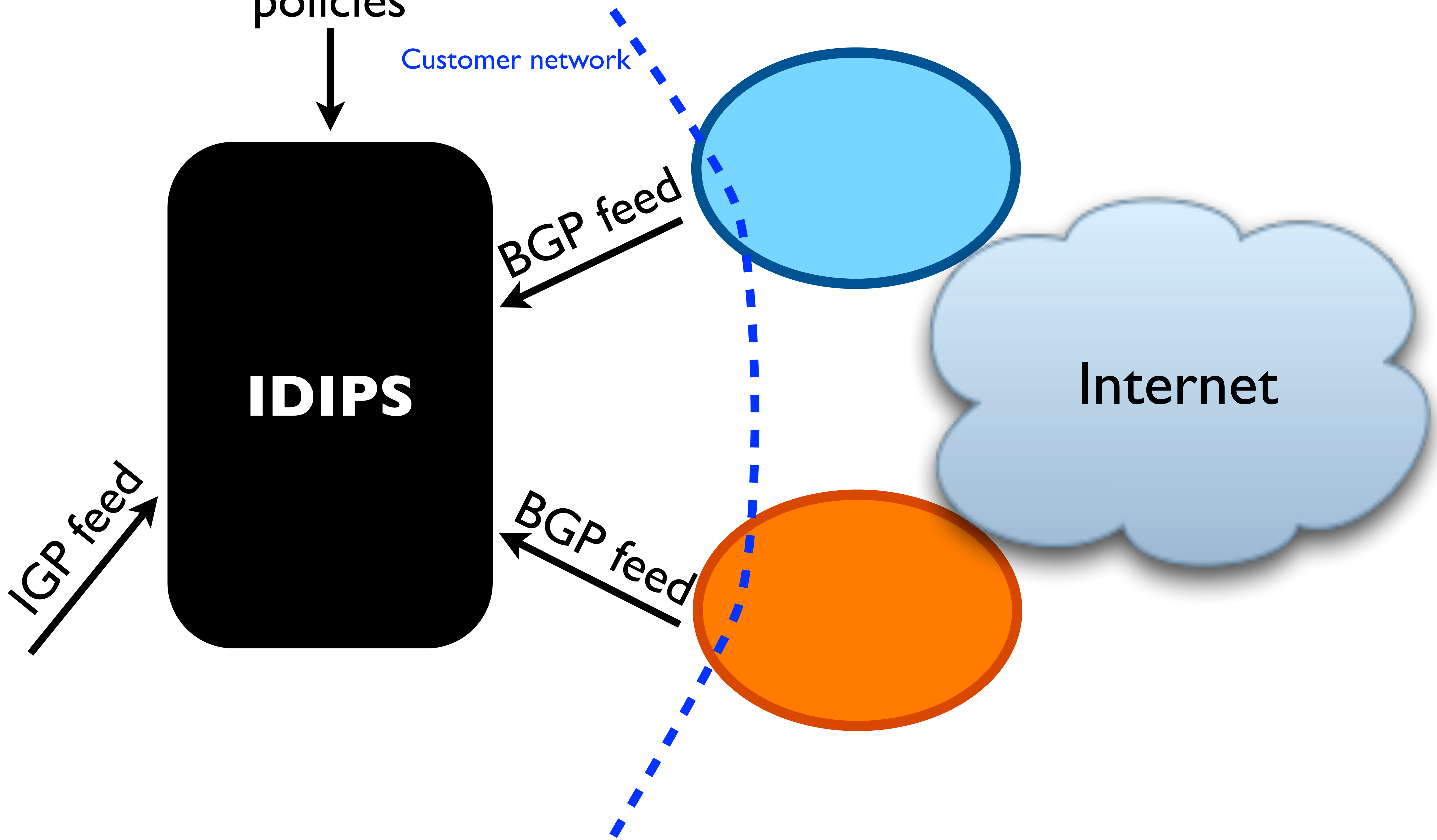
IDIPS



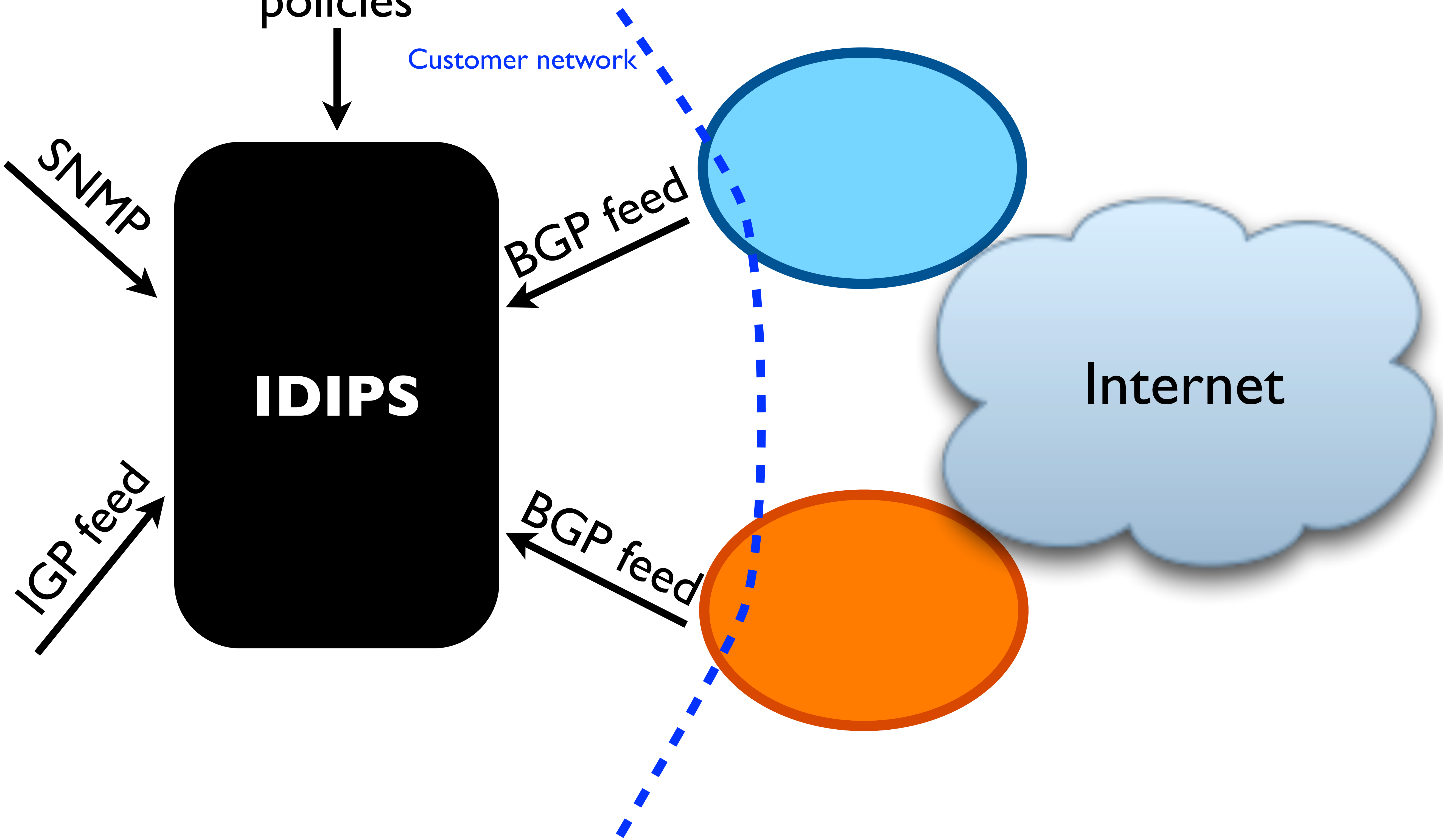
IDIPS



IDIPS

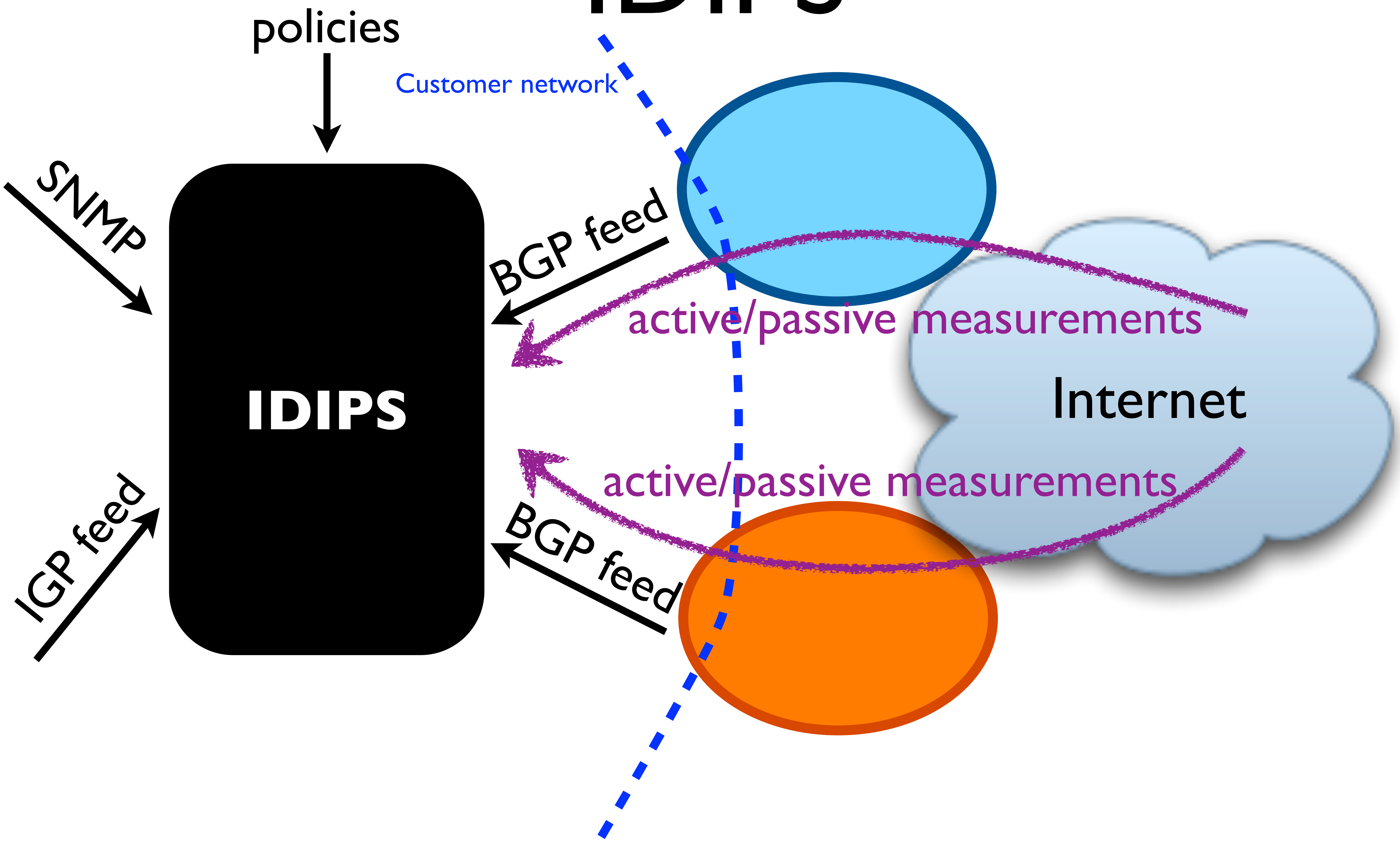

policies
Customer network
BGP feed
IDIPS
Internet
BGP feed
IGP feed

IDIPS



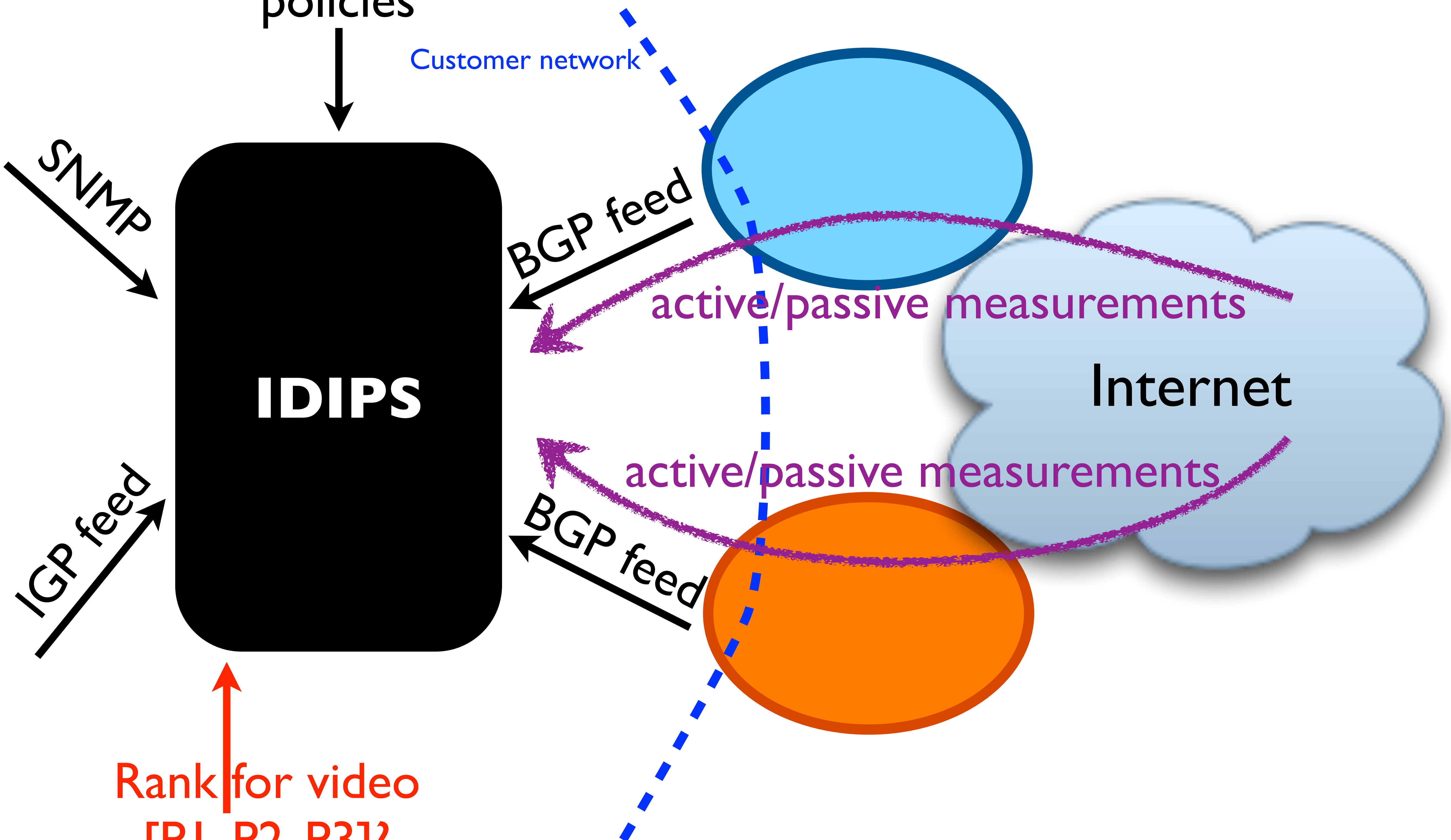
policies
Customer network
SNMP
BGP feed
IDIPS
Internet
IGP feed
BGP feed

IDIPS



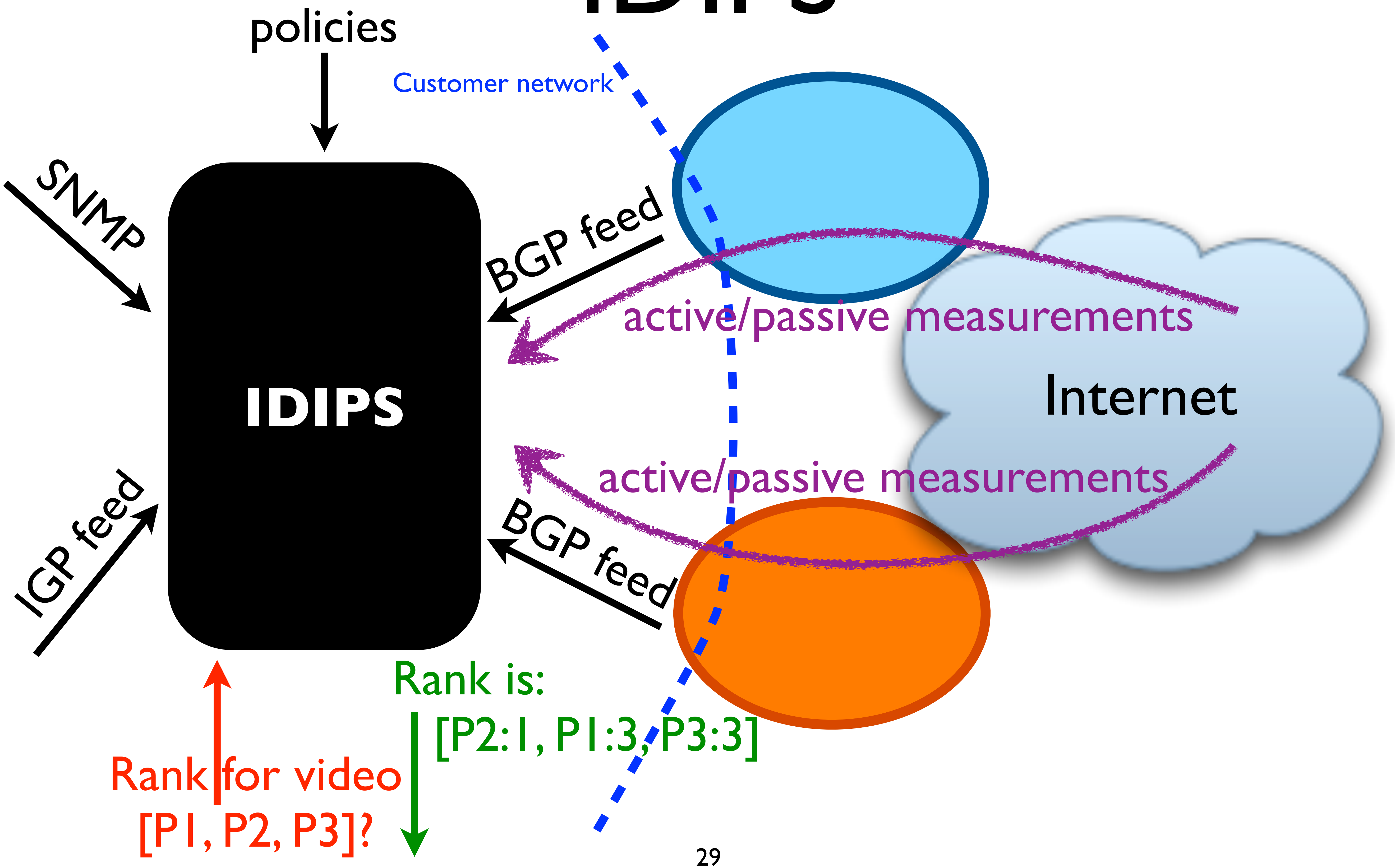
policies
Customer network
SNMP
BGP feed
active/passive measurements
IDIPS
Internet
active/passive measurements
IGP feed
BGP feed

IDIPS



policies
Customer network
SNMP
BGP feed
IDIPS
active/passive measurements
Internet
active/passive measurements
IGP feed
BGP feed
Rank for video
[P1, P2, P3]?

IDIPS


policies
Customer network
SNMP
BGP feed
IDIPS
active/passive measurements
Internet
active/passive measurements
IGP feed
BGP feed
Rank is:
[P2:1, P1:3, P3:3]
Rank for video
[P1, P2, P3]?

Abstraction

- Path performance abstraction: cost
  - positive integer
  - costs can be combined $\text{cost} = \alpha + \sum_i \beta_i \cdot \text{cost}_i^{\theta_i}$
  - costs respect a transitivity relationship
  - the lowest the cost, the better the path
- Objective: minimize the cost

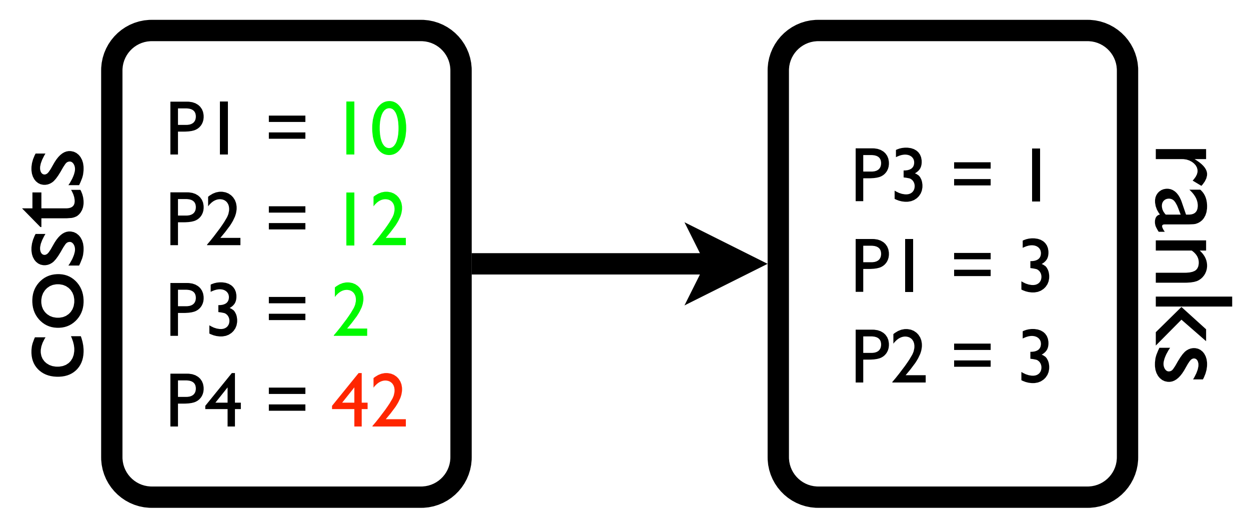
[SDIB08]

Cost function

- Cost functions implement policies in IDIPS
- A cost function computes the cost of a path regarding a given (set of) performance metric(s)
- Example: *prefer a path with a high bandwidth and a low delay but that is as cheaper as possible*

Hide critical information with the rank

- The rank is an abstraction of the cost to hide topology and computation details
- The smaller, the better
  - cost is absolute, rank is relative
  - no transitivity relationship with the ranks



Inside IDIPS

Inside IDIPS

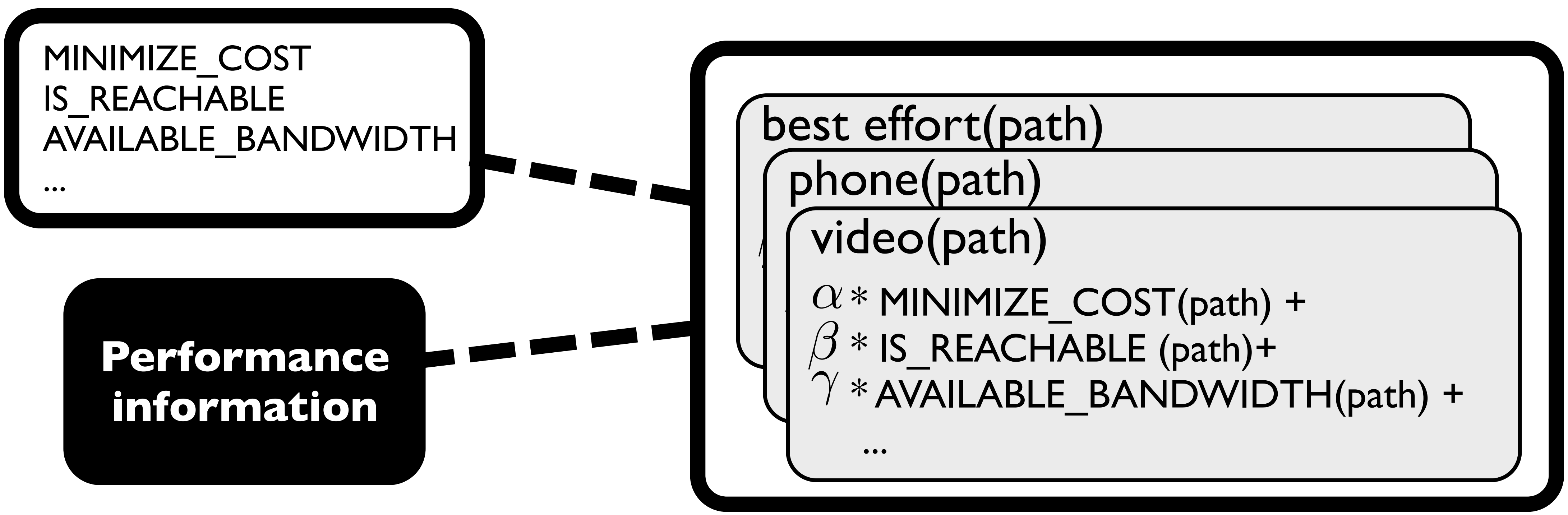
Performance information

Inside IDIPS

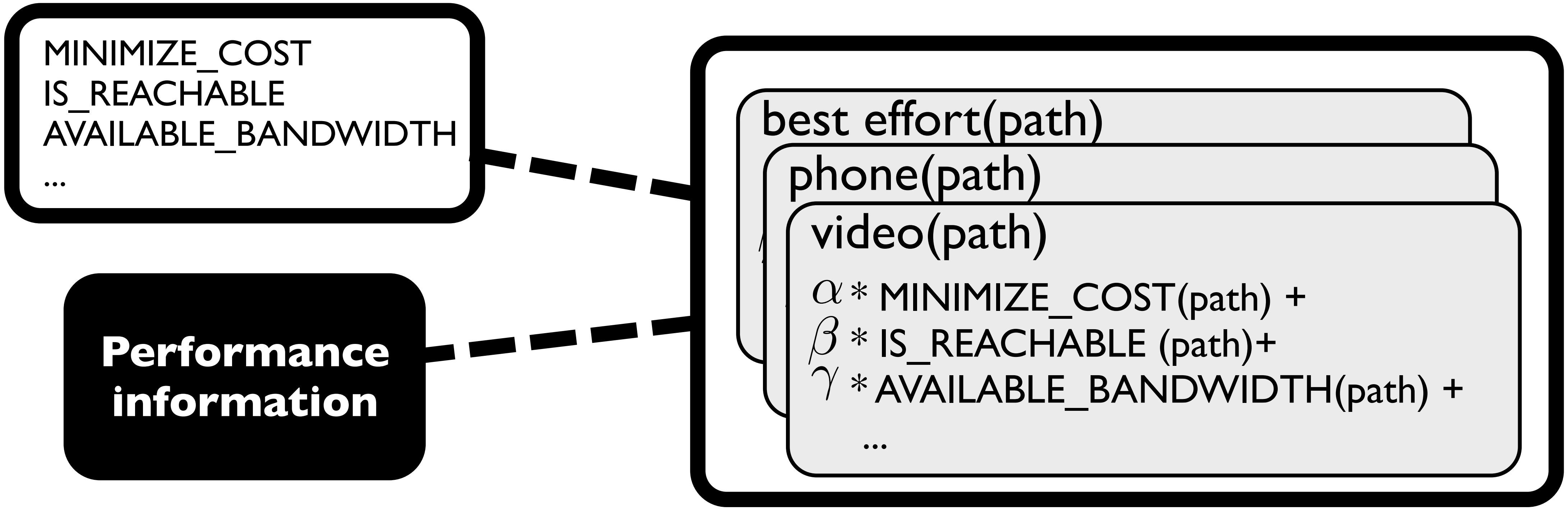
MINIMIZE_COST
IS_REACHABLE
AVAILABLE_BANDWIDTH
...

Performance information

Inside IDIPS

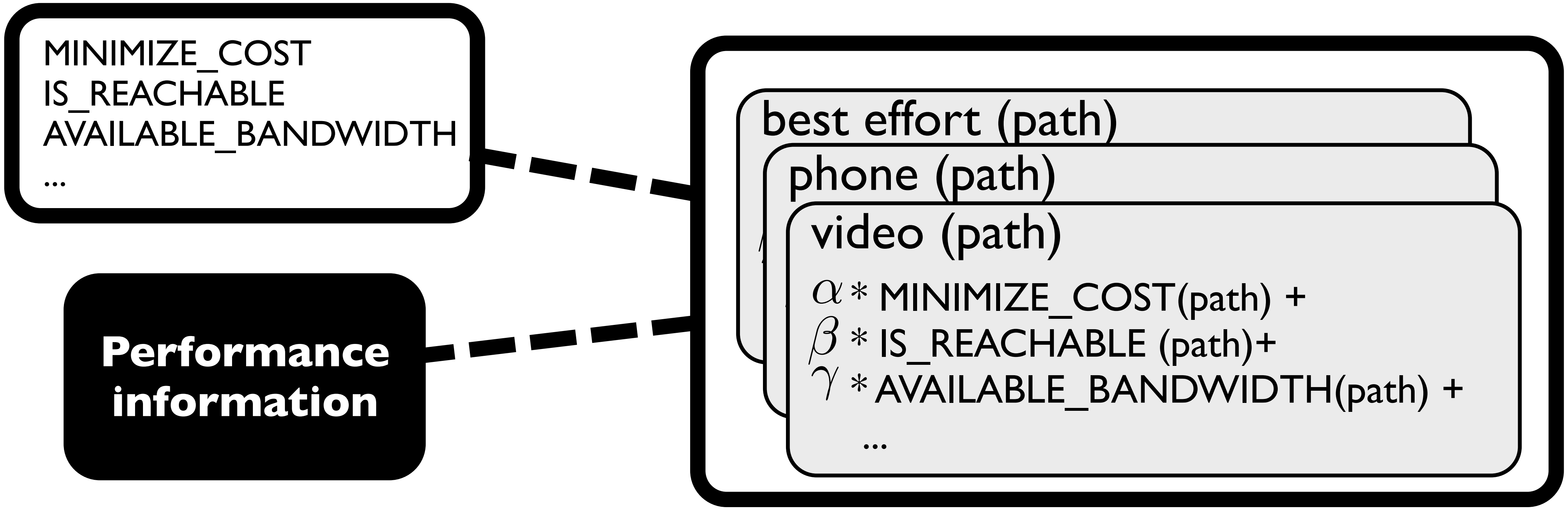


Inside IDIPS



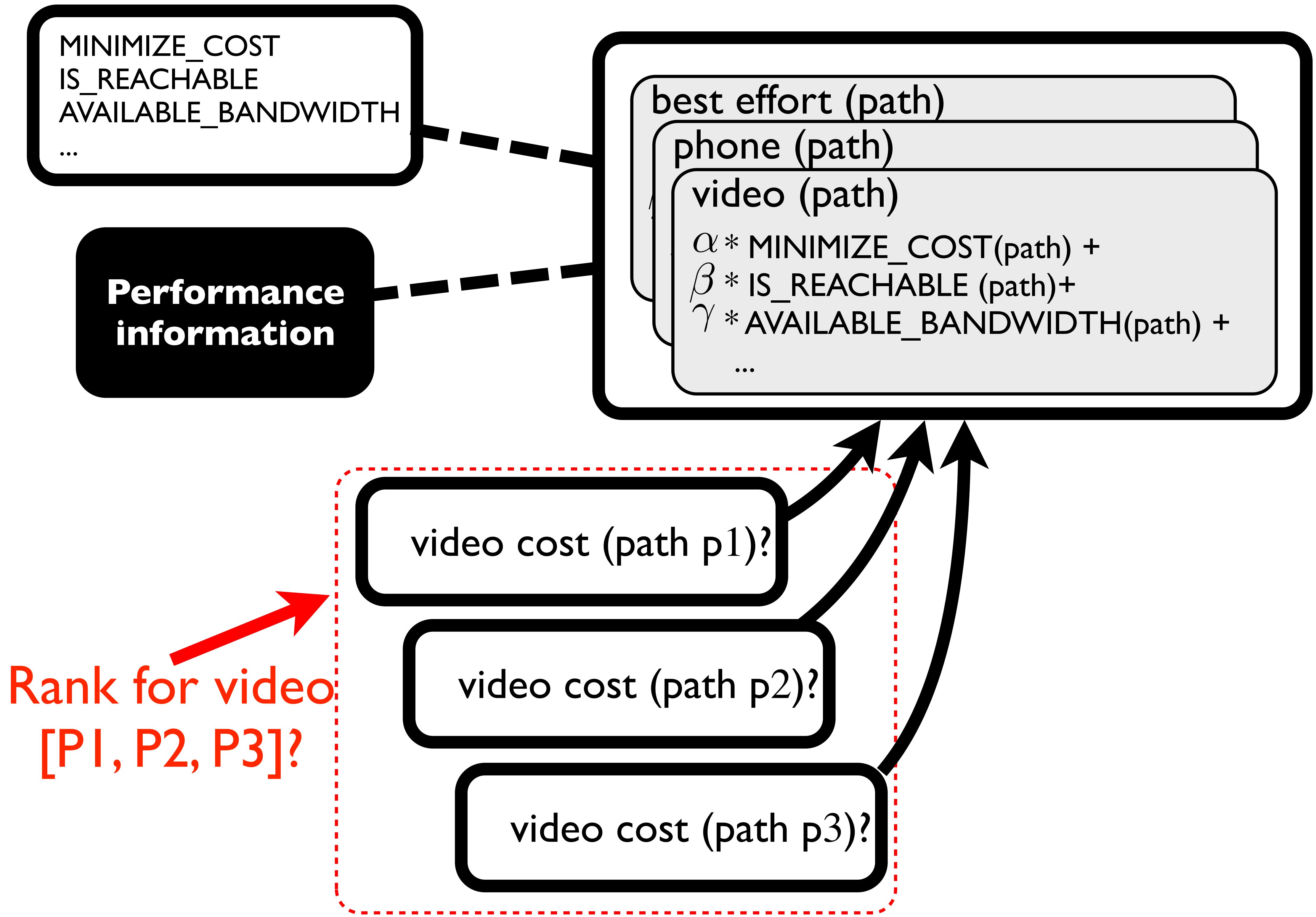
Rank for video
[P1, P2, P3]?

Inside IDIPS

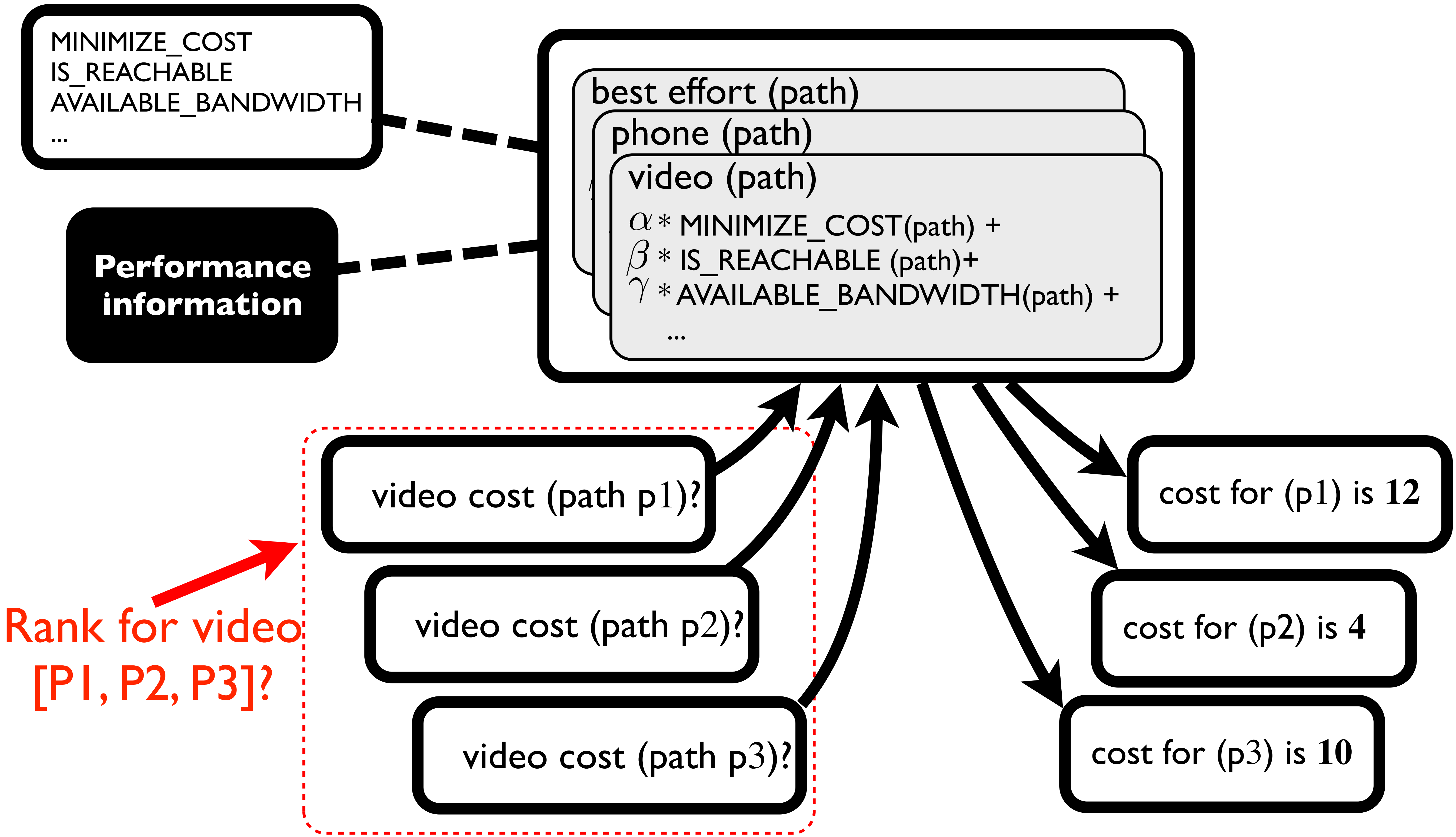


Rank for video
[P1, P2, P3]?

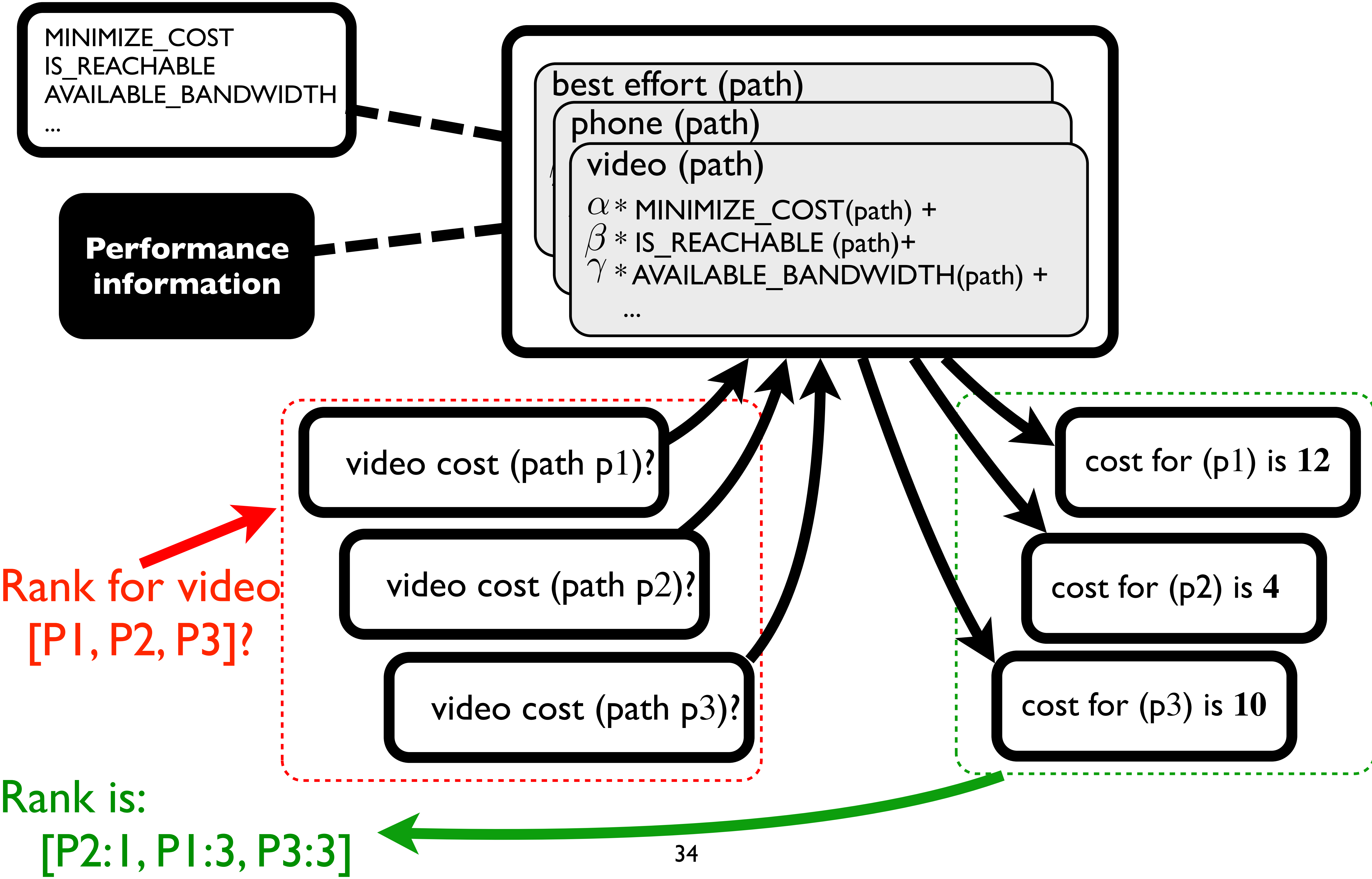
Inside IDIPS



Inside IDIPS



Inside IDIPS

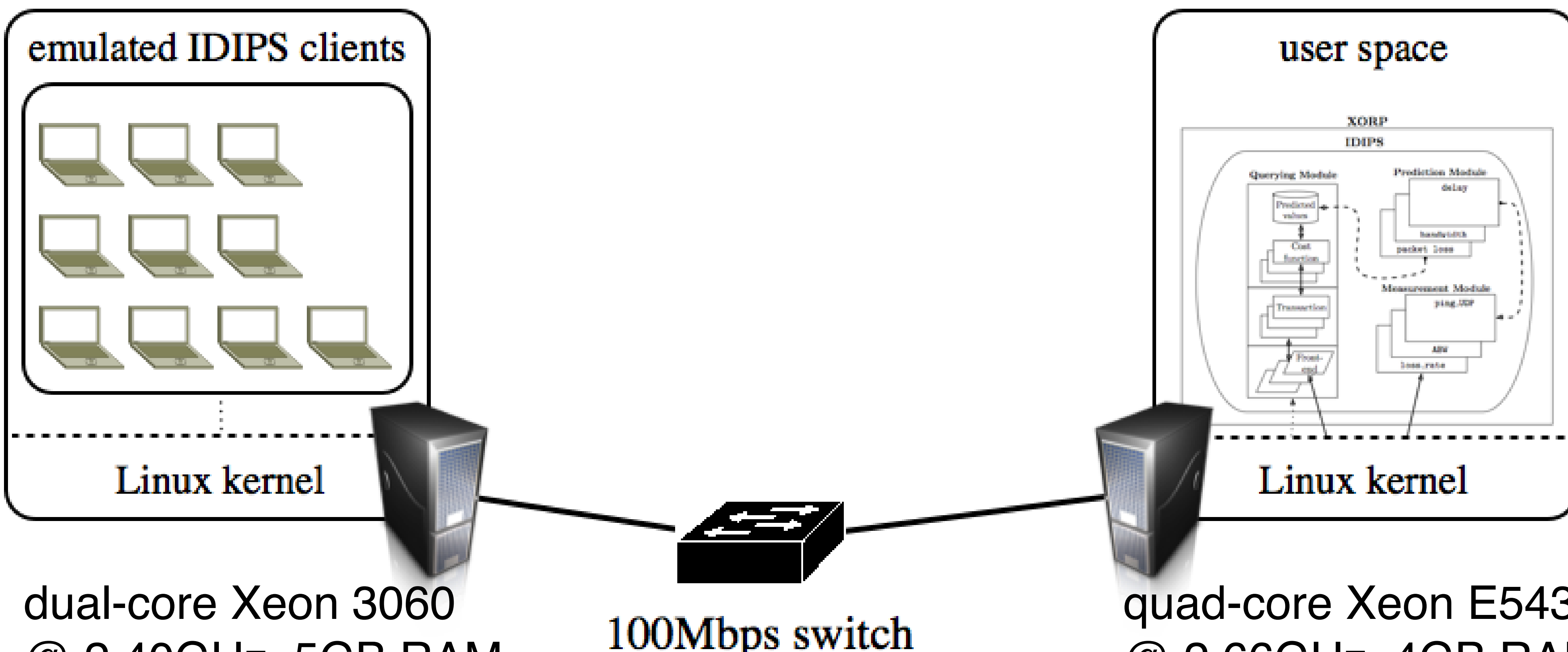


Implementation efforts

- IDIPS is implemented in XORP and is fully functional
  - around 6K lines of code + separate JSON front-end in Perl

[HHK03]

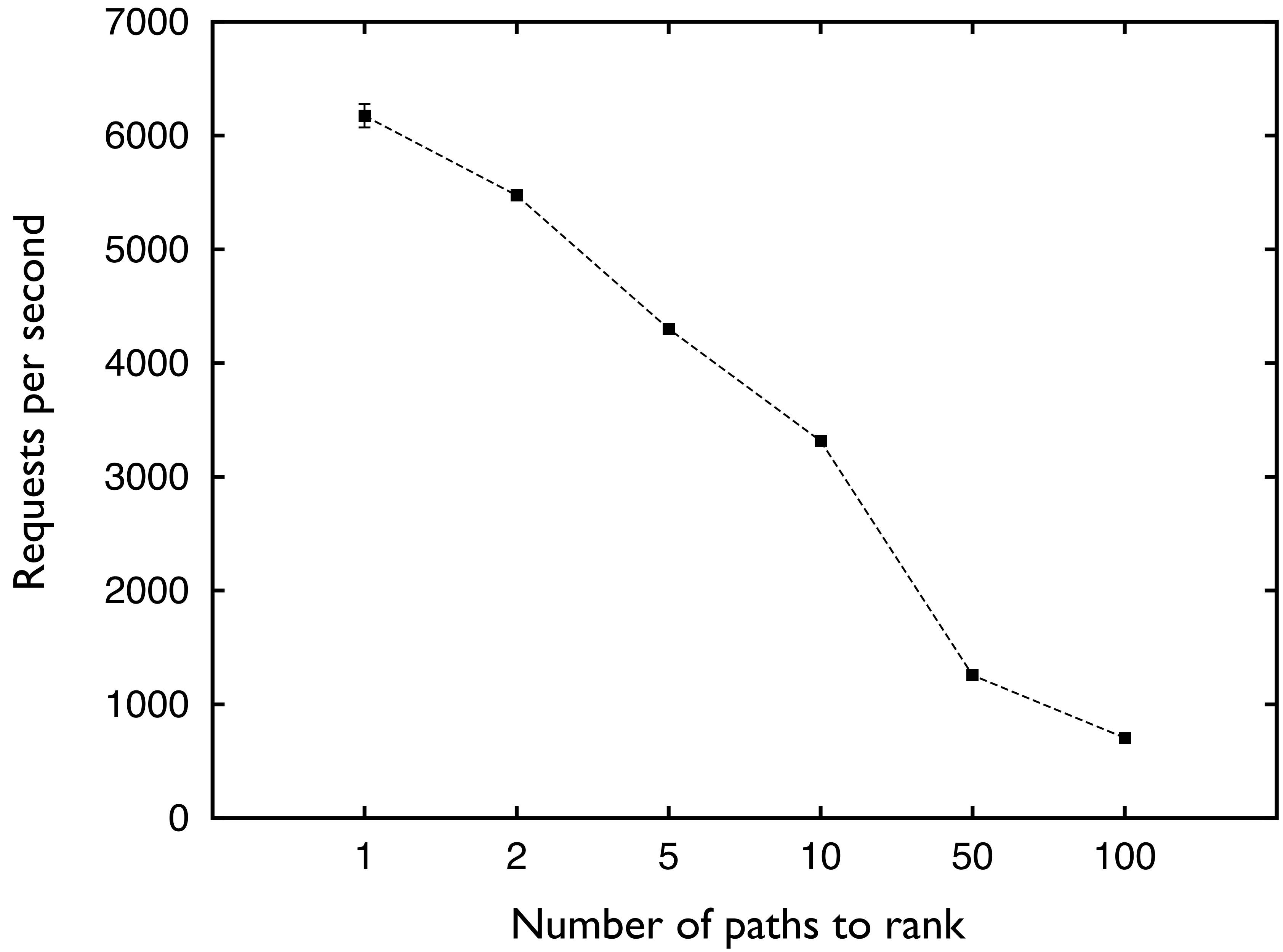
Performance evaluation testbed



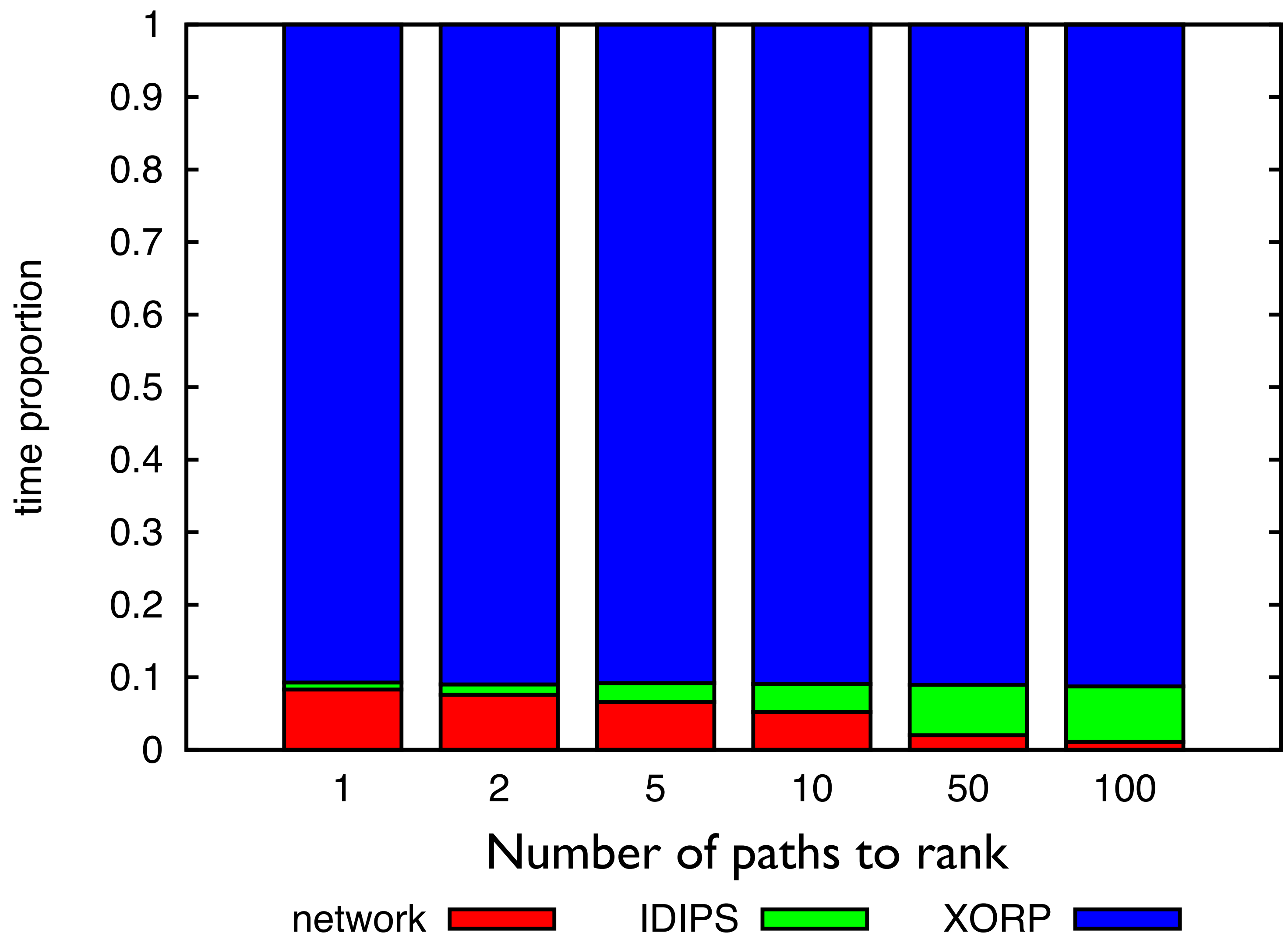
dual-core Xeon 3060
@ 2.40GHz, 5GB RAM

quad-core Xeon E5430
@ 2.66GHz, 4GB RAM

IDIPS can handle hundreds of clients



IDIPS is lightweight, XORP is not



- Up to 90% of the time is lost in XORP, not in IDIPS cost computation
  - the finder is the bottleneck

IDIPS summary

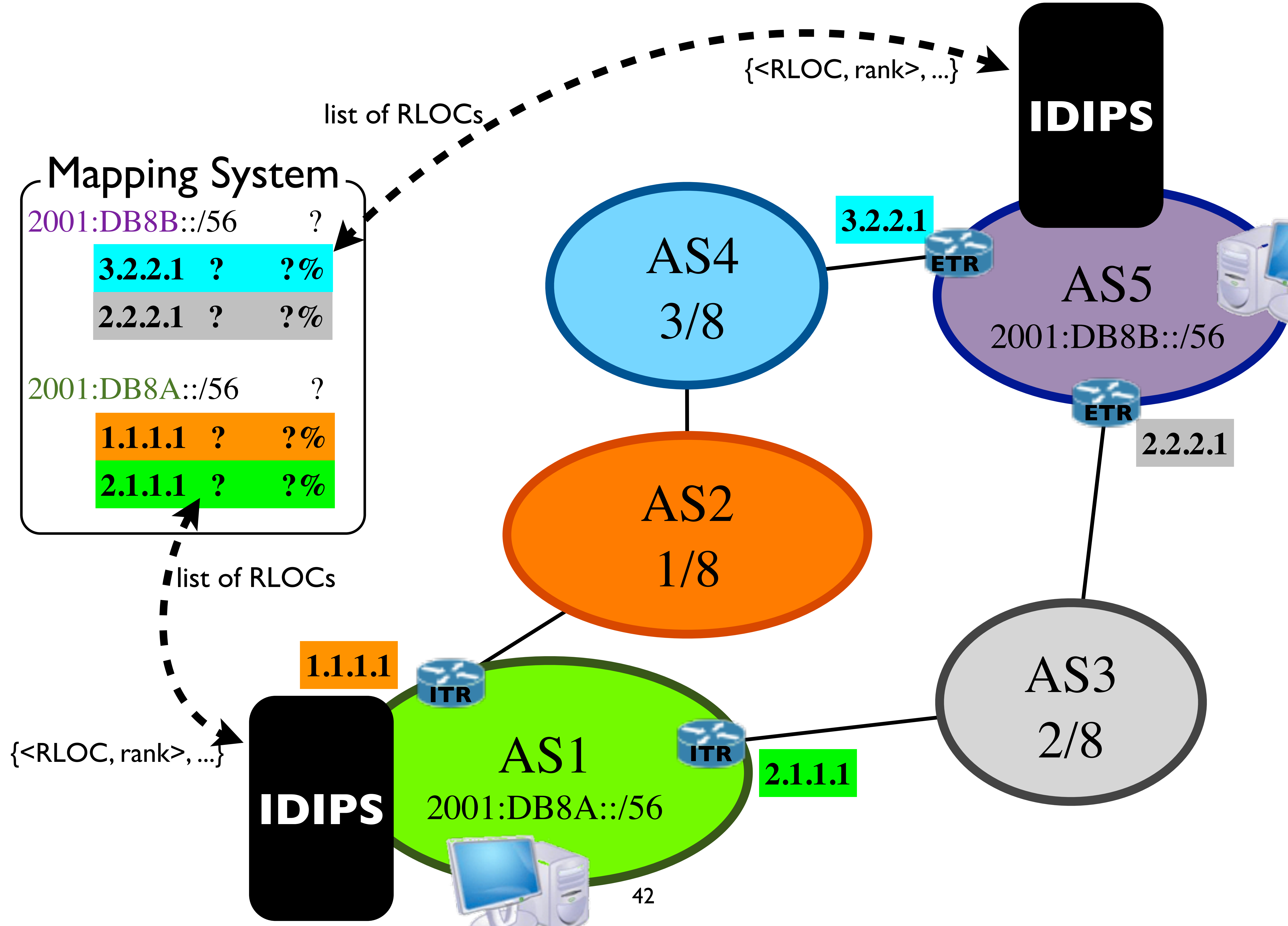
- IDIPS provides a path ranking mechanism
  - scalable
    - architecture
    - measurement reduction with source clustering
  - flexible (compatible with draft-ietf-alto-protocol-09 Sec. 7.7.5.)
    - performance are abstracted
    - cost function to implement high level policies
    - paths are abstraction
      - could be LISP, P2P, MPTCP, shim6...

Put them all together

Performance based incoming traffic engineering

- LISP provides a to do incoming traffic engineering
- IDIPS provides a way to rank paths in order to determine which paths offer the best performance

➡ How to combine both to obtain Performance based incoming traffic engineering



{<RLOC, rank>, ...}
IDIPS
list of RLOCs
Mapping System
2001:DB8B::/56 ?
3.2.2.1 ? ?%
2.2.2.1 ? ?%
2001:DB8A::/56 ?
1.1.1.1 ? ?%
2.1.1.1 ? ?%
list of RLOCs
{<RLOC, rank>, ...}
IDIPS
AS4
3/8
3.2.2.1
ETR
AS5
2001:DB8B::/56
ETR
2.2.2.1
AS2
1/8
AS3
2/8
1.1.1.1
ITR
ITR
2.1.1.1
AS1
2001:DB8A::/56

Conclusion

Conclusion

Conclusion

- Today's inter-domain traffic engineering follows a trial-and-error approach
  - impossible to know how the routes will be propagated
  - path performance are mostly ignored

Conclusion

- Today's inter-domain traffic engineering follows a trial-and-error approach
  - impossible to know how the routes will be propagated
  - path performance are mostly ignored
- How to enable performance based traffic engineering?

Conclusion

Conclusion

- In this thesis we proposed

Conclusion

- In this thesis we proposed
  - to use LISP to efficiently implement incoming traffic engineering

Conclusion

- In this thesis we proposed
  - to use LISP to efficiently implement incoming traffic engineering
    - for that we had to design LISP-Tree a scalable control-plane for LISP

Conclusion

- In this thesis we proposed
  - to use LISP to efficiently implement incoming traffic engineering
    - for that we had to design LISP-Tree a scalable control-plane for LISP
  - IDIPS, a ranking mechanism to efficiently rank paths according to their performance

Conclusion

- In this thesis we proposed
  - to use LISP to efficiently implement incoming traffic engineering
    - for that we had to design LISP-Tree a scalable control-plane for LISP
  - IDIPS, a ranking mechanism to efficiently rank paths according to their performance
  - to combine IDIPS, LISP and LISP-Tree as a mechanism for Interdomain Traffic Engineering

draft-previdi-isis-te-metric-extensions-00 - IS-IS Traffic Engineering (TE) Metric Extensions

http://tools.ietf.org/html/draft-previdi-isis-te-metric-extensions-00

http://www.i...rfc4677.txt | UCL - Public... de la thèse | Detexify LaT...recognition | Network Top...co Systems

Network Working Group
Internet Draft
Intended status: Proposed Standard
Expires: April 2012

S. Previdi
Cisco Systems

S. Giacalone
Thomson Reuters

D. Ward
Juniper Networks

J. Drake
Juniper Networks

A. Atlas
Juniper Networks

C.Filsfils
Cisco Systems

October 10, 2011

IS-IS Traffic Engineering (TE) Metric Extensions
draft-previdi-isis-te-metric-extensions-00.txt

Abstract

In certain networks, such as, but not limited to, financial information networks (e.g. stock market data providers), network performance criteria (e.g. latency) are becoming as critical to data path selection as other metrics.

This document describes extensions to IS-IS TE [RFC5305] such that network performance information can be distributed and collected in a scalable fashion. The information distributed using ISIS TE Express Path can then be used to make path selection decisions based on network performance.

Note that this document only covers the mechanisms with which network

Thank You

?? || /**/

LISP contributions

- Implementing the Locator/ID Separation Protocol: Design and Experience, with Iannone, L. and Bonaventure, O., Computer Networks, March 2011
- LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System, with Jakab, L., Cabelos, A., Coras, F. and Bonaventure, O., JSAC, October 2010
- draft-saucez-lisp-iterable-mapping-00, with Bonaventure, O., IETF draft, October 2010
- LISP-Click: A Click implementation of the Locator/ID Separation Protocol, with Nguyen, V. , 1st Symposium on Click Modular Router, November 2009
- IETF at LISP WG: draft-ietf-lisp-map-versioning, draft-ietf-lisp-sec, draft-ietf-lisp-threats, draft-bonaventure-lisp-preserve-00

IDIPS contributions

- On the Impact of Clustering on Measurement Reduction, with D., Donnet, B. and Bonaventure, O., Networking 2009, May 2009
- Interdomain Traffic Engineering in a Locator/Identifier Separation Context, with Donnet, B., Iannone, L. and Bonaventure, O., Internet Network Management Workshop 2008 (INM), October 2008
- Implementation and Preliminary Evaluation of an ISP-Driven Informed Path Selection, with Donnet, B. and Bonaventure, O., ACM CoNEXT Student Workshop, December 2007
- miscellaneous at IETF ALTO WG: draft-akonjang-alto-proxidor-00, draft-saucez-alto-generalized-alto-00, draft-saucez-idips-00, draft-bonaventure-informed-path-selection-00

Further works

Measurements

- How to measure a cluster (which IP)?
- Measurement prediction
- Auto-adaptive measurement frequency
- Avoid oscillations

LISP

- How to ensure the end-to-end reachability?
- Deploy LISP-Tree in the wild
- Security
- Alternative deployments (enterprises, data-centers...)

Backup

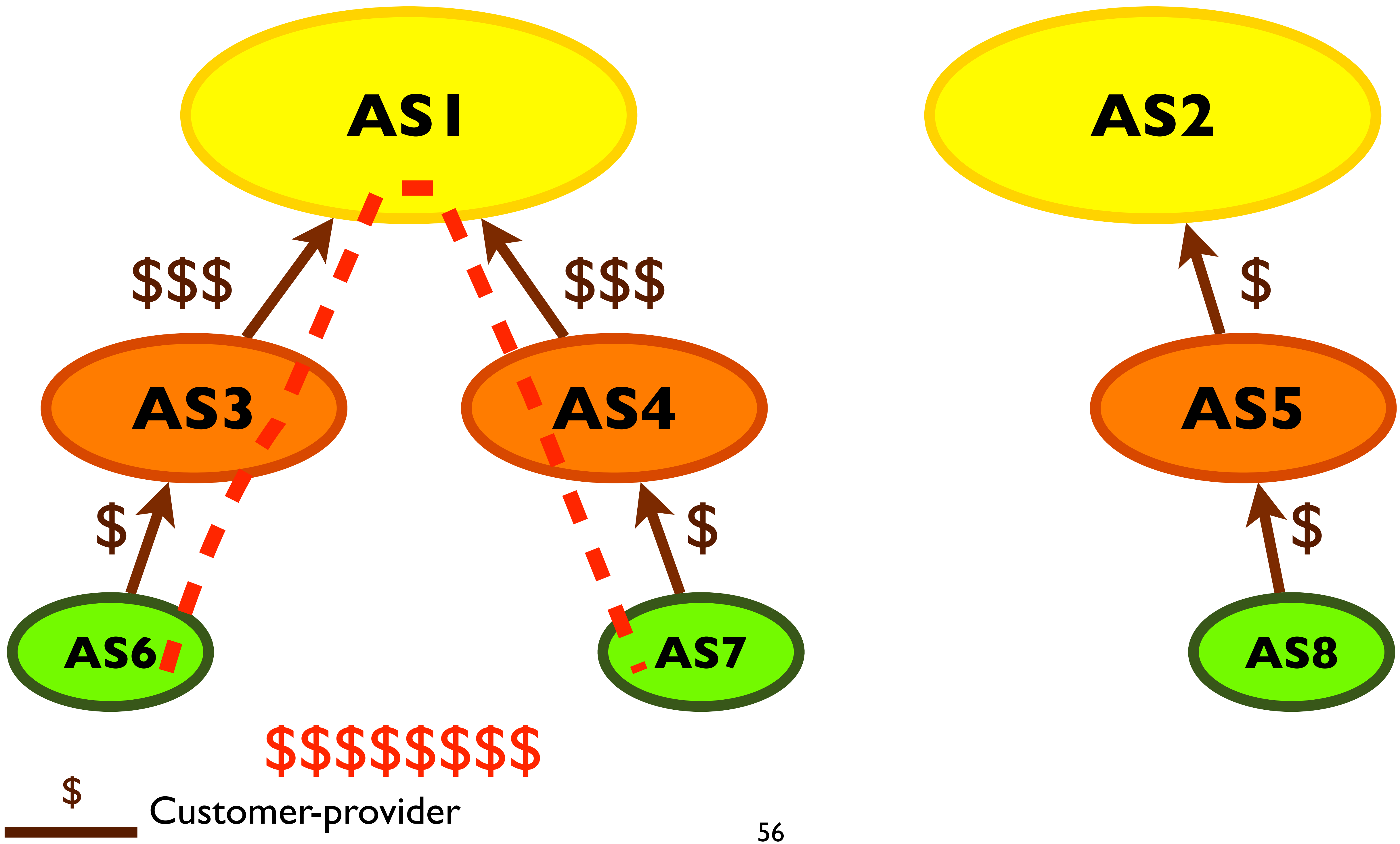
Interdomain Routing

- Goal
  - Allow to transmit data along the best path towards the destination through several transit domains while taking into account the *routing policies* of each domain without knowing the detailed topology of those domains
- The *Border Gateway Protocol* (BGP) is the common protocol between the domains [RLH06]

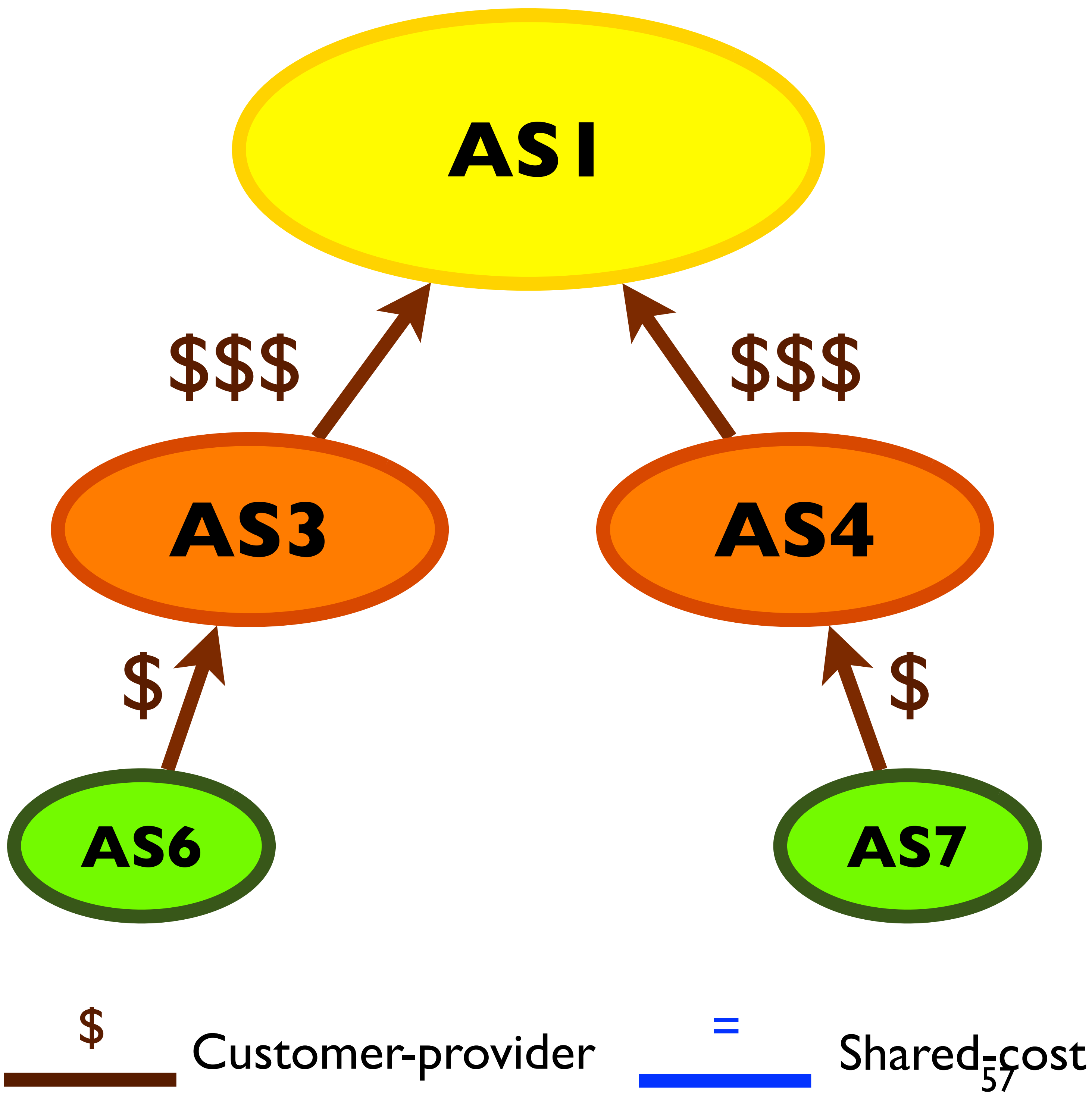
Routing Policies

- In theory, BGP allows each domain to define its own routing policy...
- In practice, there are two common policies:
  - **Customer-provider peering:** customer *c* buy Internet connectivity to provider *p*
  - **Shared-cost peering:** domains *x* and *y* agree to exchange data by using a direct link through an interconnection point

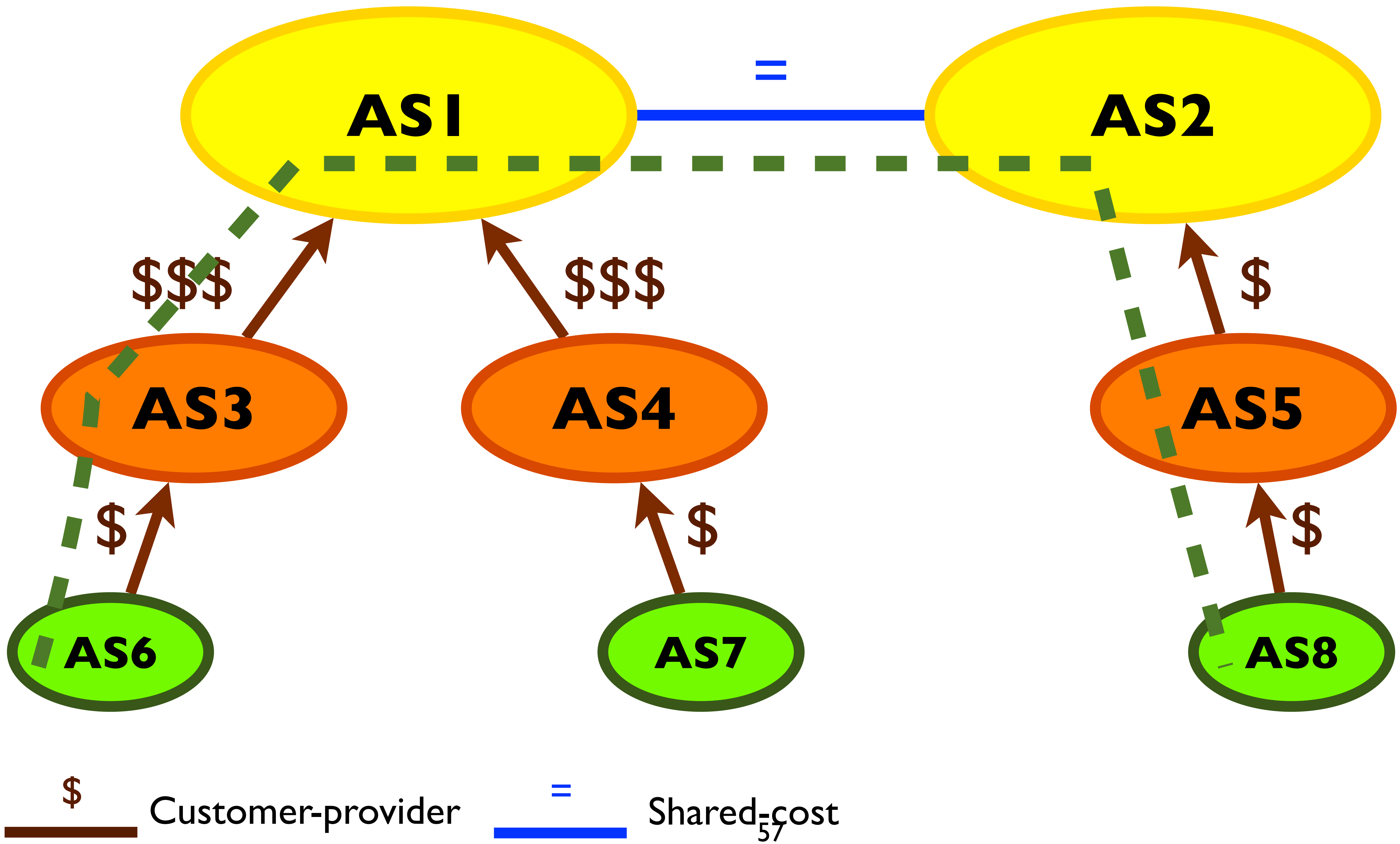
Customer-provider peering



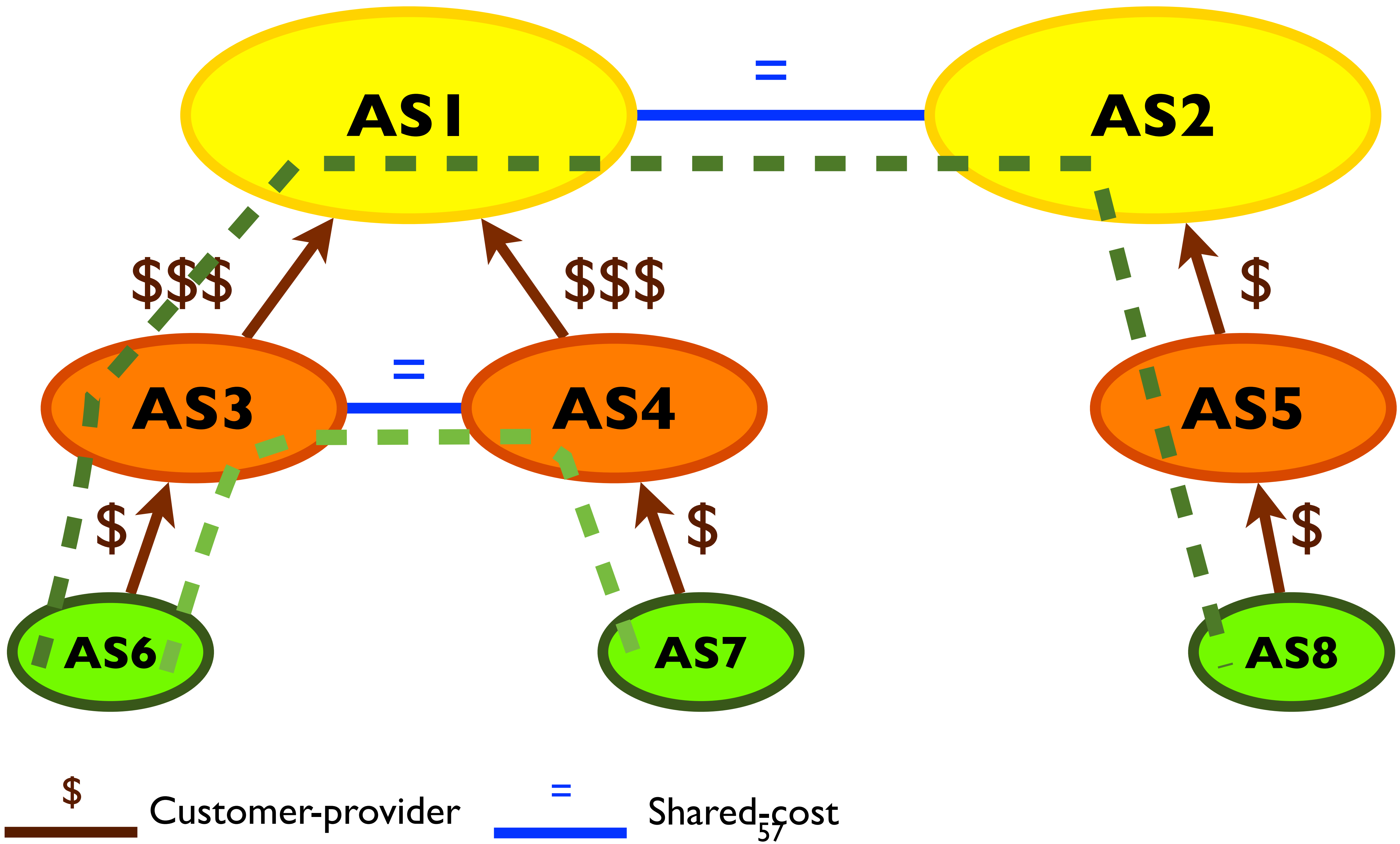
Shared-cost peering



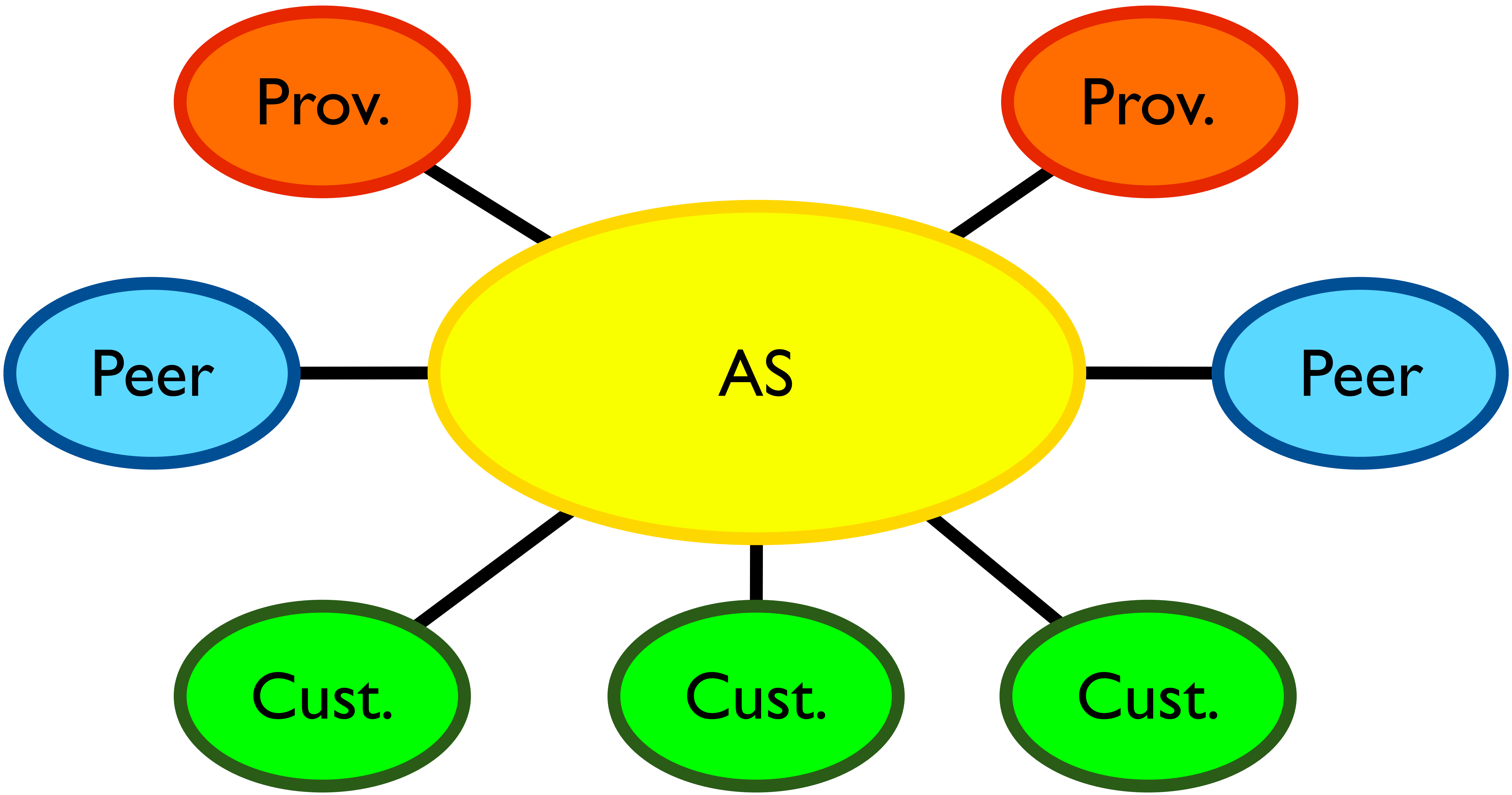
Shared-cost peering



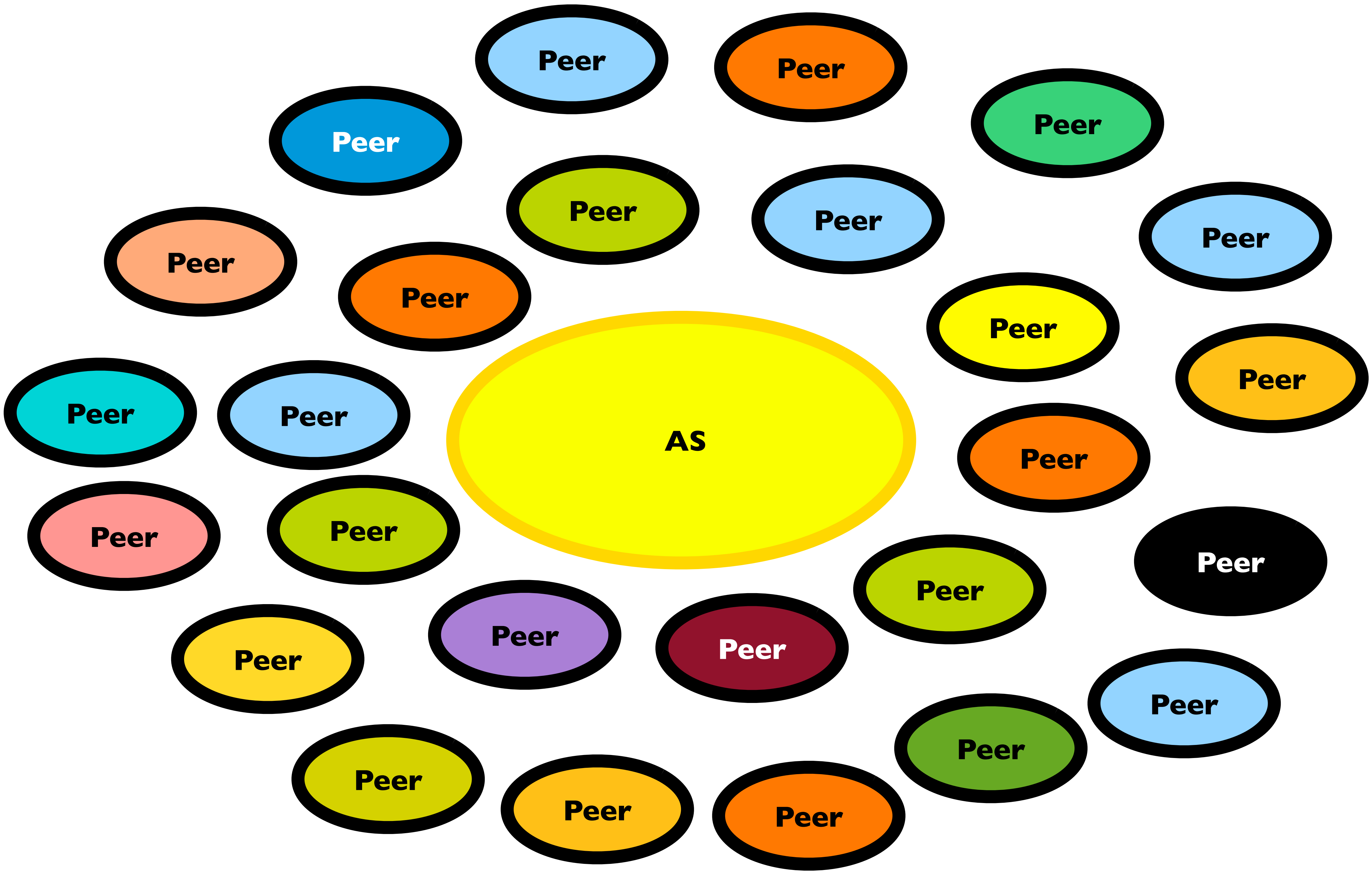
Shared-cost peering



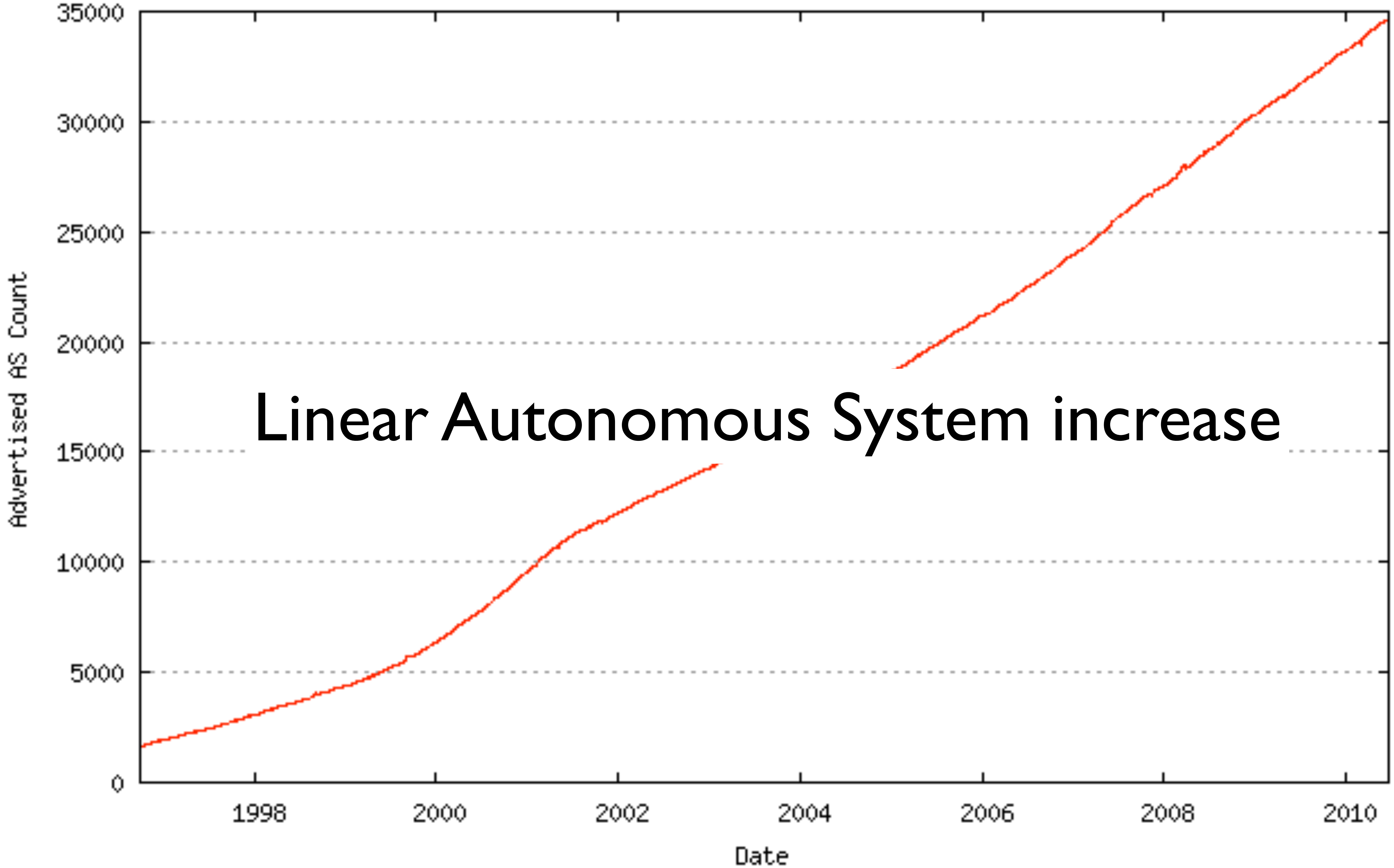
The simple case...



The nightmare...

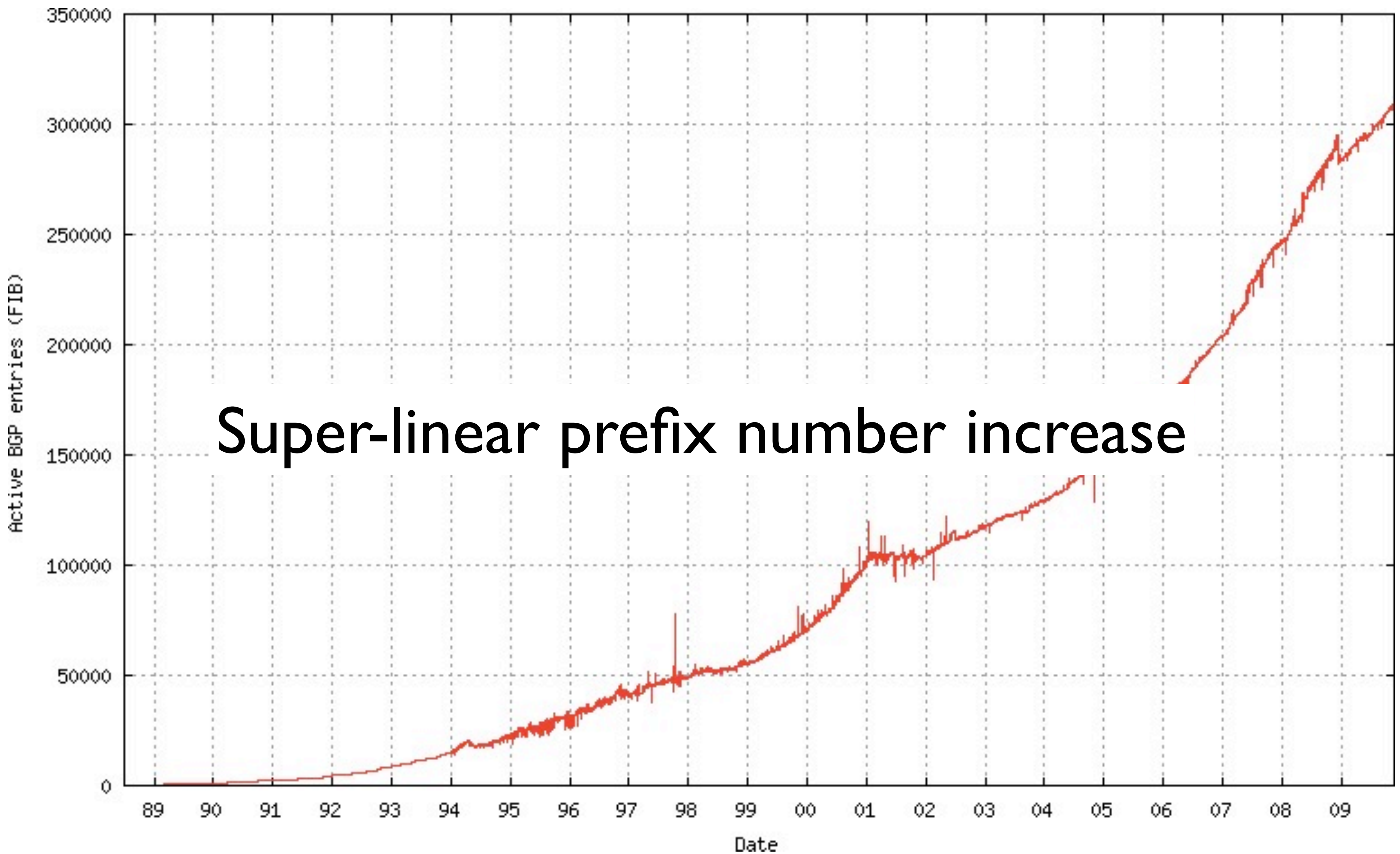


More networks



[http://bgp.potaroo.net]

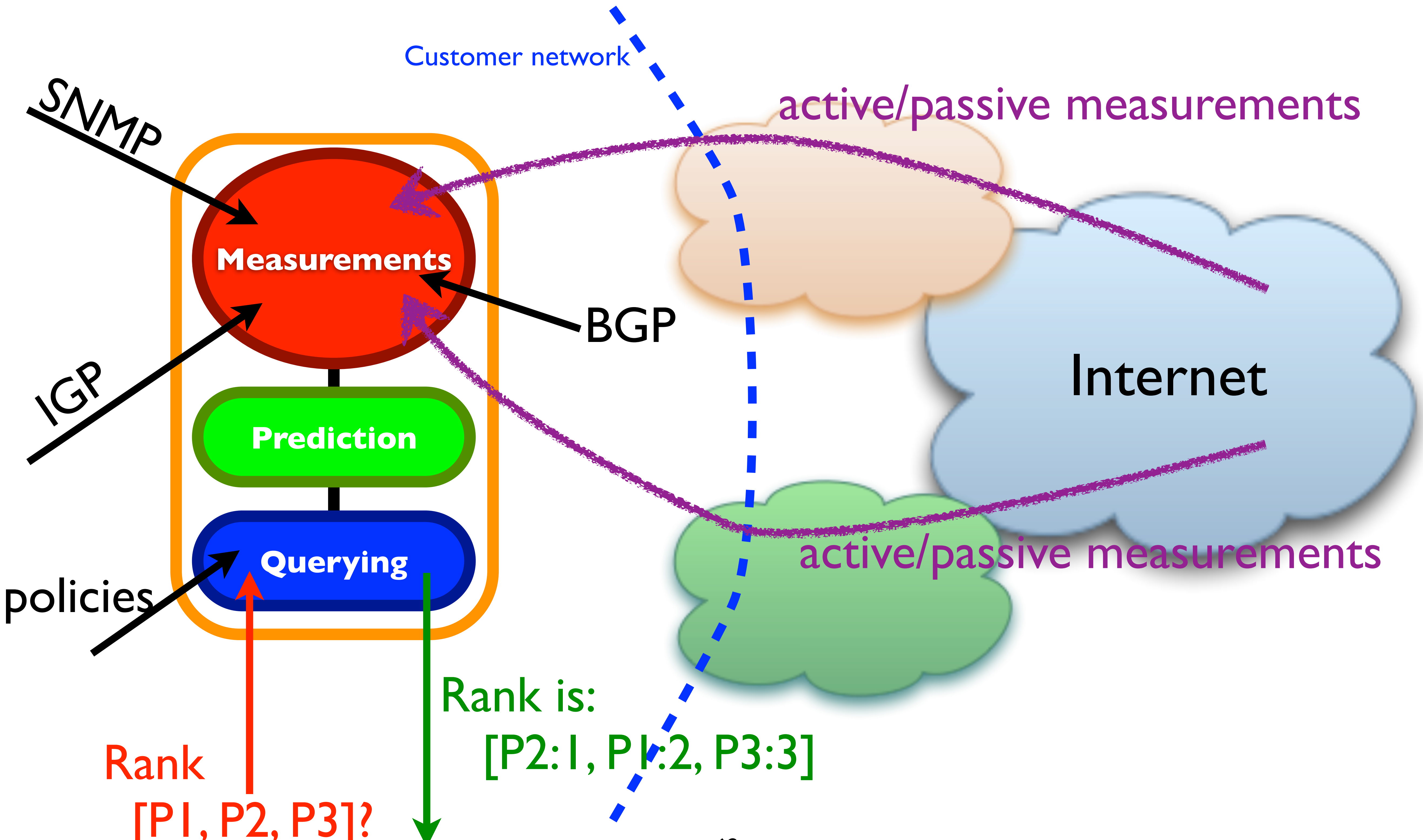
More prefixes



Super-linear prefix number increase

[http://bgp.potaroo.net]

Inside IDIPS



Abstraction

- Path performance abstraction: cost
  - combine performance metric
    - $cost = \sum w_i * cost_i$
- Objective: minimize the cost

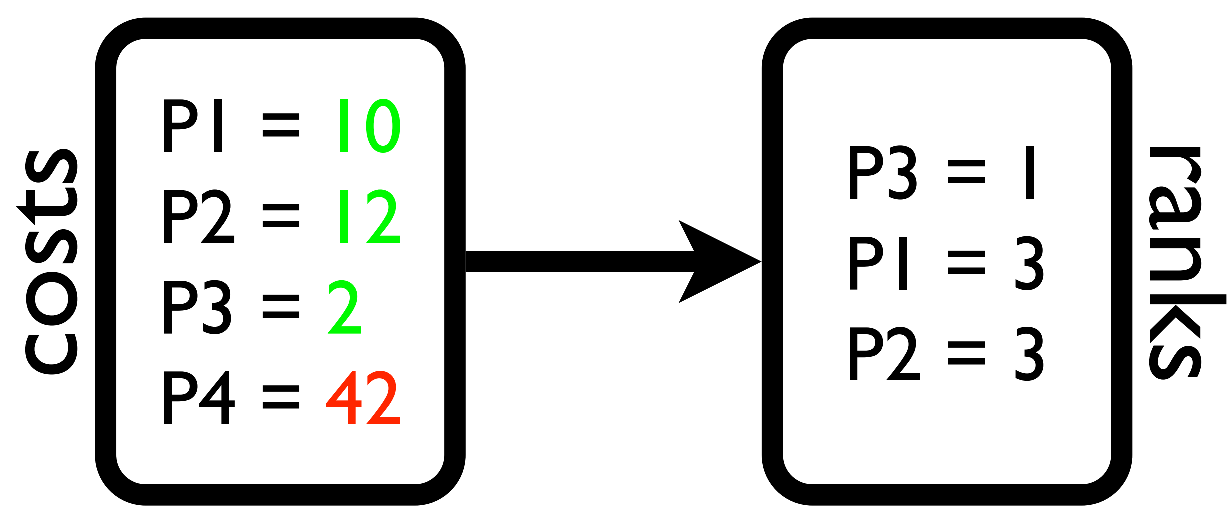
[SDIB08] Saucez, D. et al. *Interdomain Traffic Engineering in a Locator/Identifier Separation Context*. INM'08

Cost function

- Cost functions implement policies in IDIPS
- A cost function computes the cost of a path regarding a given (set of) performance metric(s)
  - input: <*src*, *dst*> pair, i.e., a path
  - output: a cost (a positive integer)
- the lowest the cost, the better the path
- transitivity with cost function relationship

Path ranking

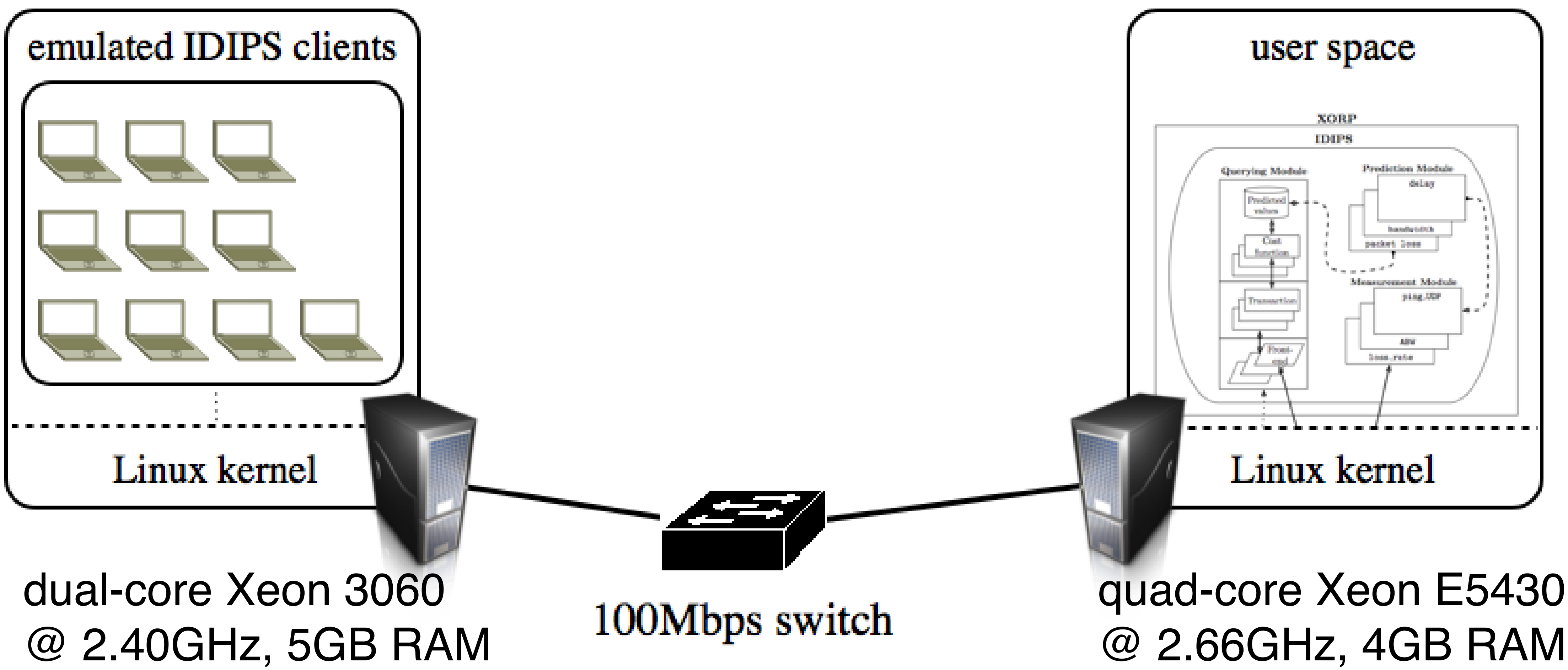
- The rank is an abstraction of the cost to hide topology and computation details
- The smaller, the better
  - cost is absolute, rank is relative
  - cost relationship is transitive, not the ranking



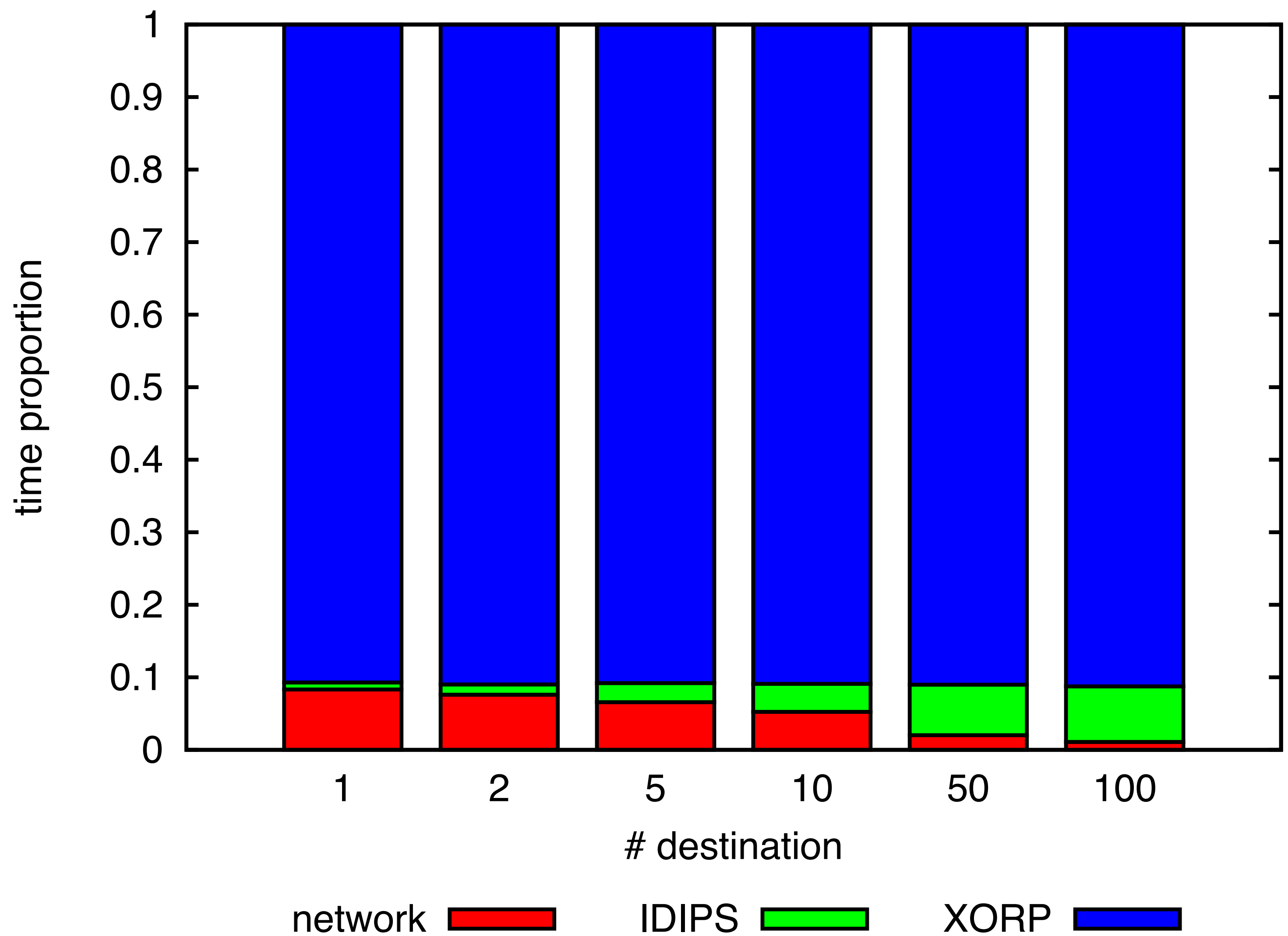
Implementation efforts

- IDIPS is implemented in XORP and is fully functional
  - around 6K lines of code + separate JSON front-end in Perl
- IDIPS implements the "Endpoint Cost Service" specifications of the ALTO protocol (draft-ietf-alto-protocol-09 Sec. 7.7.5.)

Performance evaluation testbed



IDIPS is lightweight, XORP is not



- Up to 90% of the time is lost in XORP, not in IDIPS cost computation
  - the finder is the bottleneck

Note well: path definition

- In this work, an inter-domain path, or **path**, is *defined by the exit point of the source network and the entry point of the destination network considered*
  - anything before or after is not considered [CH10]
  - anything in-between is only statically considered as not controllable or even discoverable
- A path is define by a
  <source, destination> pair
  - typically (but not limited to) IP addresses

[CH10] Cai, X., Heidemann, J.: Understanding block-level address usage in the visible Internet. In: Proc. ACM SIGCOMM, 2010

Note well: focus on inter-domain incoming traffic engineering

Note well: focus on inter-domain incoming traffic engineering

- Intra-domain traffic engineering are already well known and controlled [FRT02, FT00, FT02]

Note well: focus on inter-domain incoming traffic engineering

- Intra-domain traffic engineering are already well known and controlled [FRT02, FT00, FT02]
- Inter-domain outgoing traffic engineering is also well studied [SGD03, QUP+03]

Locator discovery

- Based on Archipelago traceroutes

traceroute to 153.16.35.1 (153.16.35.1), 30 hops max, 40 byte packets
1 CtStevin.sri.ucl.ac.be (130.104.72.222)
2 CtPythagore.sri.ucl.ac.be (130.104.254.229)
3 ge.ar1.lln.belnet.net (193.191.11.9) 0.765 ms
4 10ge.cr1.brueve.belnet.net (193.191.17.133)
5 xe-2-3-0.ldn4nqp1.uk.ip.tdc.net (195.66.224.64)
6 ge3-0-0.sltnxc98.dk.ip.tdc.net (83.88.31.154)
7 tdc-pxtr.rloc.lisp4.net (193.162.145.46)
8 damien-xtr.lisp4.net (153.16.35.1)
9 www.lisp.info.ucl.ac.be (153.16.35.10)

Locator discovery

- Based on Archipelago traceroutes

traceroute to 153.16.35.1 (153.16.35.1), 30 hops max, 40 byte packets

1 CtStevin.sri.ucl.ac.be (130.104.72.222)
2 CtPythagore.sri.ucl.ac.be (130.104.254.229)
— Stub: 130.104/16

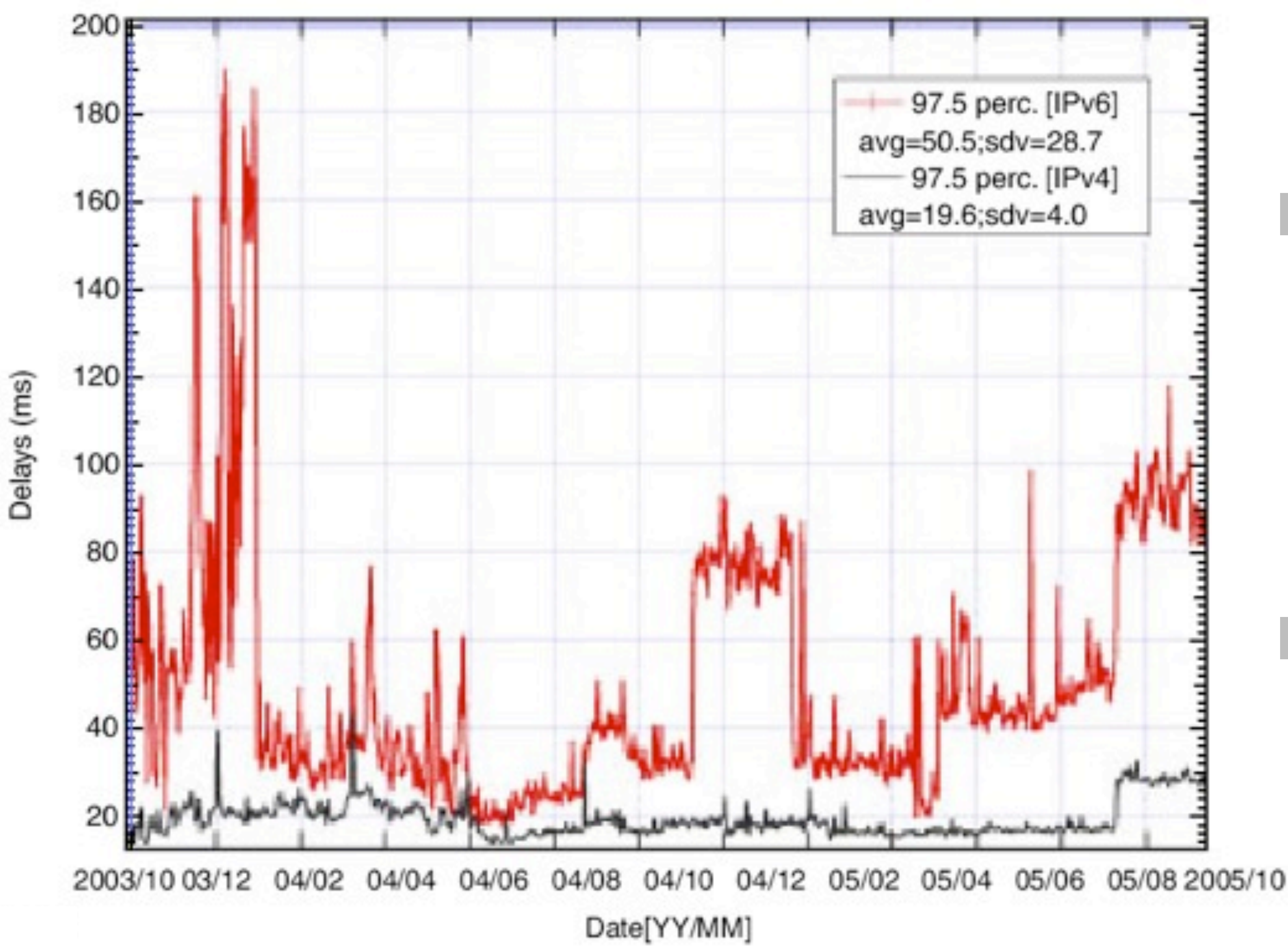
3 ge.ar1.lln.belnet.net (193.191.11.9) 0.765 ms
— 130.104/16's border

4 10ge.cr1.brueve.belnet.net (193.191.17.133)
5 xe-2-3-0.ldn4nqp1.uk.ip.tdc.net (195.66.224.64)
6 ge3-0-0.sltnxc98.dk.ip.tdc.net (83.88.31.154)
— Core

7 tdc-pxtr.rloc.lisp4.net (193.162.145.46)
— 153.16.35/24's border

8 damien-xtr.lisp4.net (153.16.35.1)
9 www.lisp.info.ucl.ac.be (153.16.35.10)
— Stub: 153.16.35/24

Dual Stack



- IPv4 and IPv6 do not provide the same performances
- IPv6 less stable than IPv4

[ZJUM08] X. Zhou et al., IPv6 delay and loss performance evolution, IJCS 2008

Predict performances

- path performance prediction is a Machine Learning (ML) problem
  - input
    - performance measurements
  - output
    - expected performances

ML: 1. pre-process

- observation might have “gaps”
  - transient failures, packet loss...
- data imputation (smooth fit)
  - average,
  - median,
  - k-nearest neighbor
  - ...

ML: 2. predict

- time series analysis
- support vector regression
- hidden Markov model
- ...

ML: 3. refine

- the network changes with the time
- performance index
  - tune learning model' parameters
- used to determine the measurement frequency

The Locator Identifier Separation Protocol (1/2)

- Define a router-based solution where current IP addresses are separated in two different spaces
  - Endpoint IDentifiers (EID)
    - identify end-hosts
    - non-globally routable
    - hosts in a given site are expected to use EIDs in the same prefix
  - Routing LOCators (RLOC)
    - attached to routers (router interfaces)
    - globally routable

The Locator Identifier Separation Protocol (2/2)

- Follows the Map-and-Encap principle
  - a mapping system maps EID prefixes onto site router's RLOCs
  - routers encapsulate the packets received from hosts before sending them towards the destination RLOC
  - routers decapsulate the packets received from the Internet before sending them towards the destination hosts

Evolutionary versus Clean-Slate

- Two possible approaches to make the Internet better
  - clean-slate
    - restart from scratch and avoid errors from the past
  - evolutionary
    - fix the errors from the past

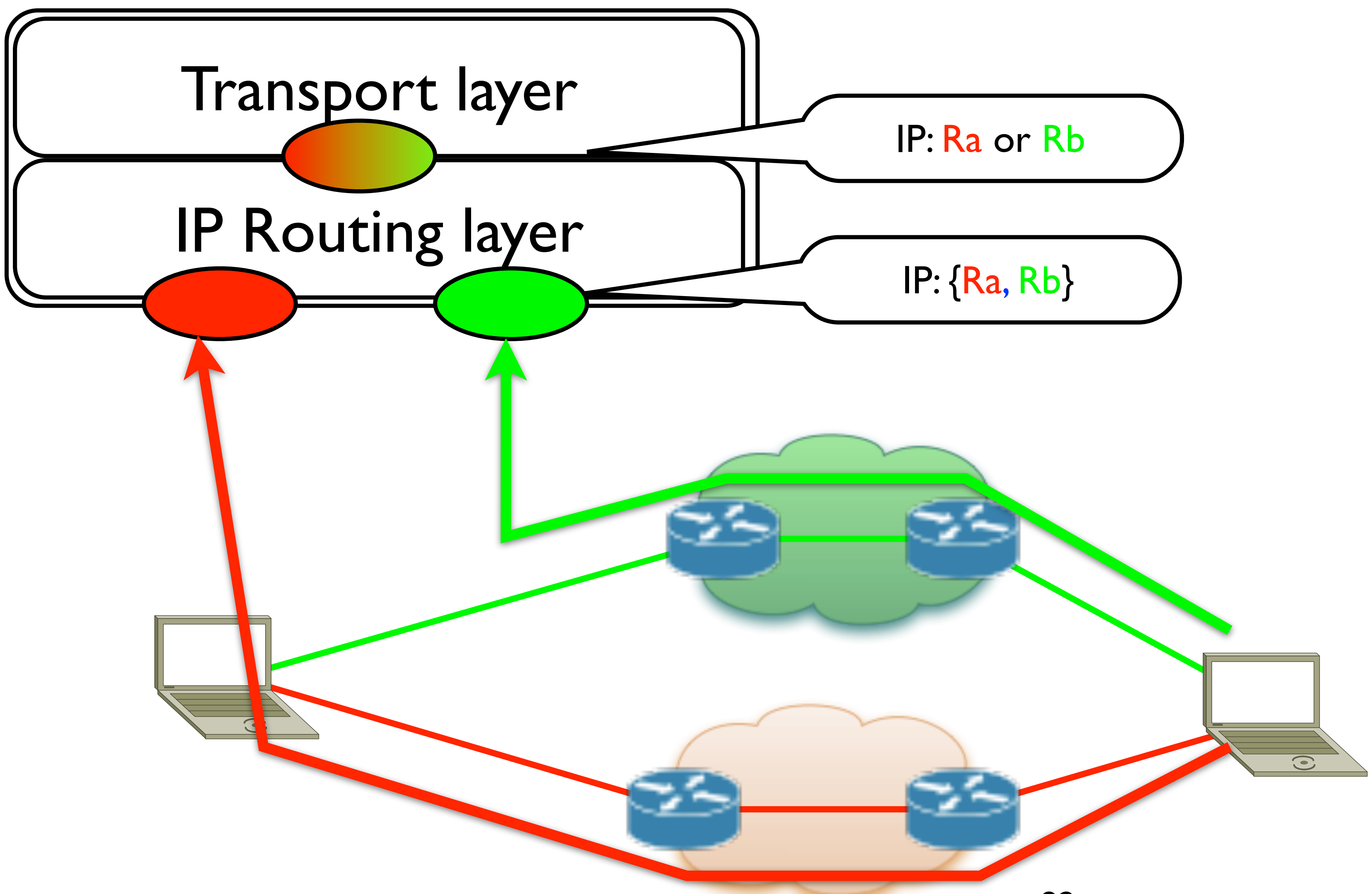
The complementary role of IP addresses

- IP addresses play two complementary roles:
  - identifier to distinguish (with port) the endpoint of transport flows (*who*)
  - locator to reveal the position in the topology (*where*)
    - may change with the topology
- Changing the locator breaks the pending flows

The Locator/Identifier Separation Paradigm

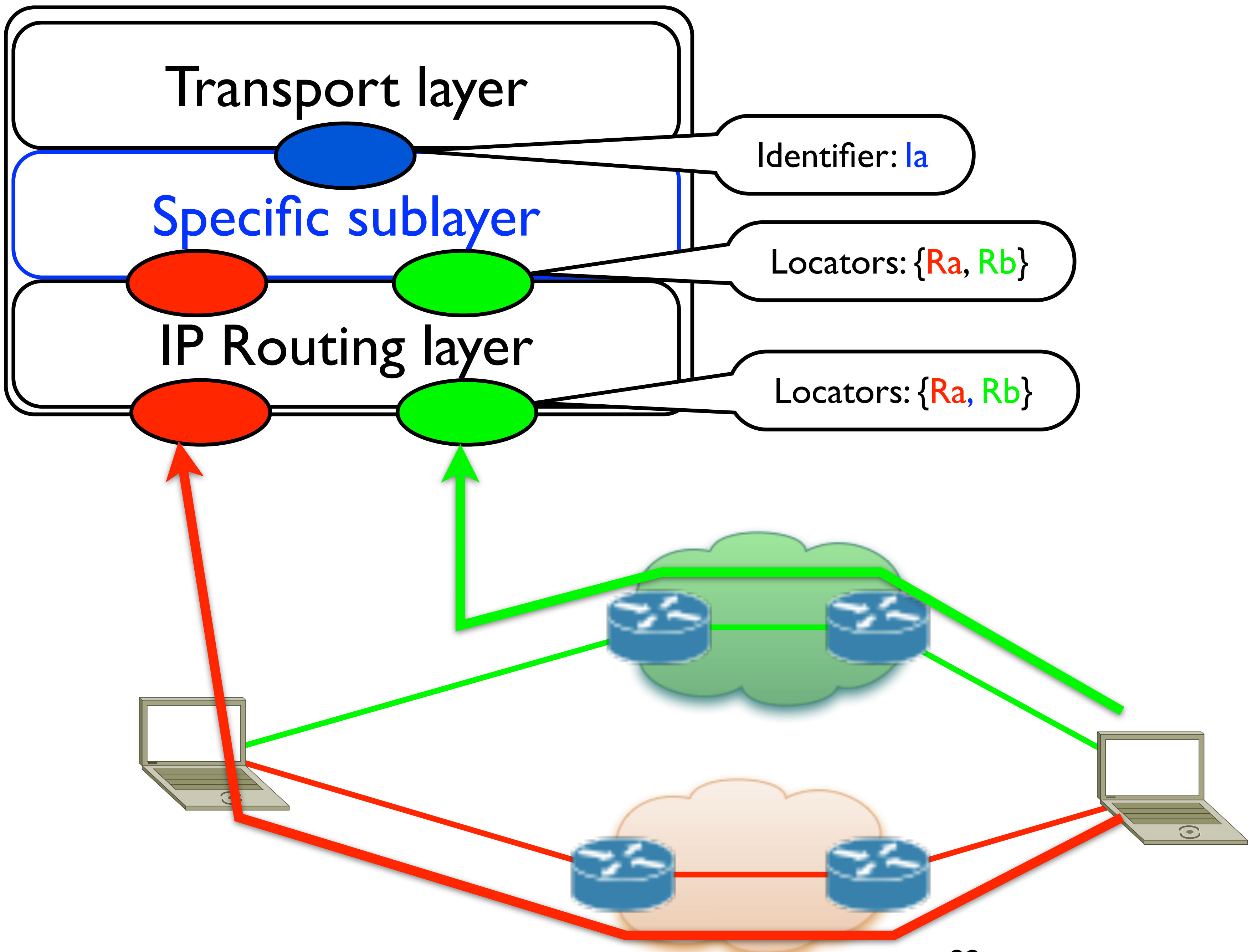
- To keep the flow alive, separate the two roles of IP addresses
- Different levels of separation
  - at the host
  - at the network

Host-Based Locator/ Identifier Separation



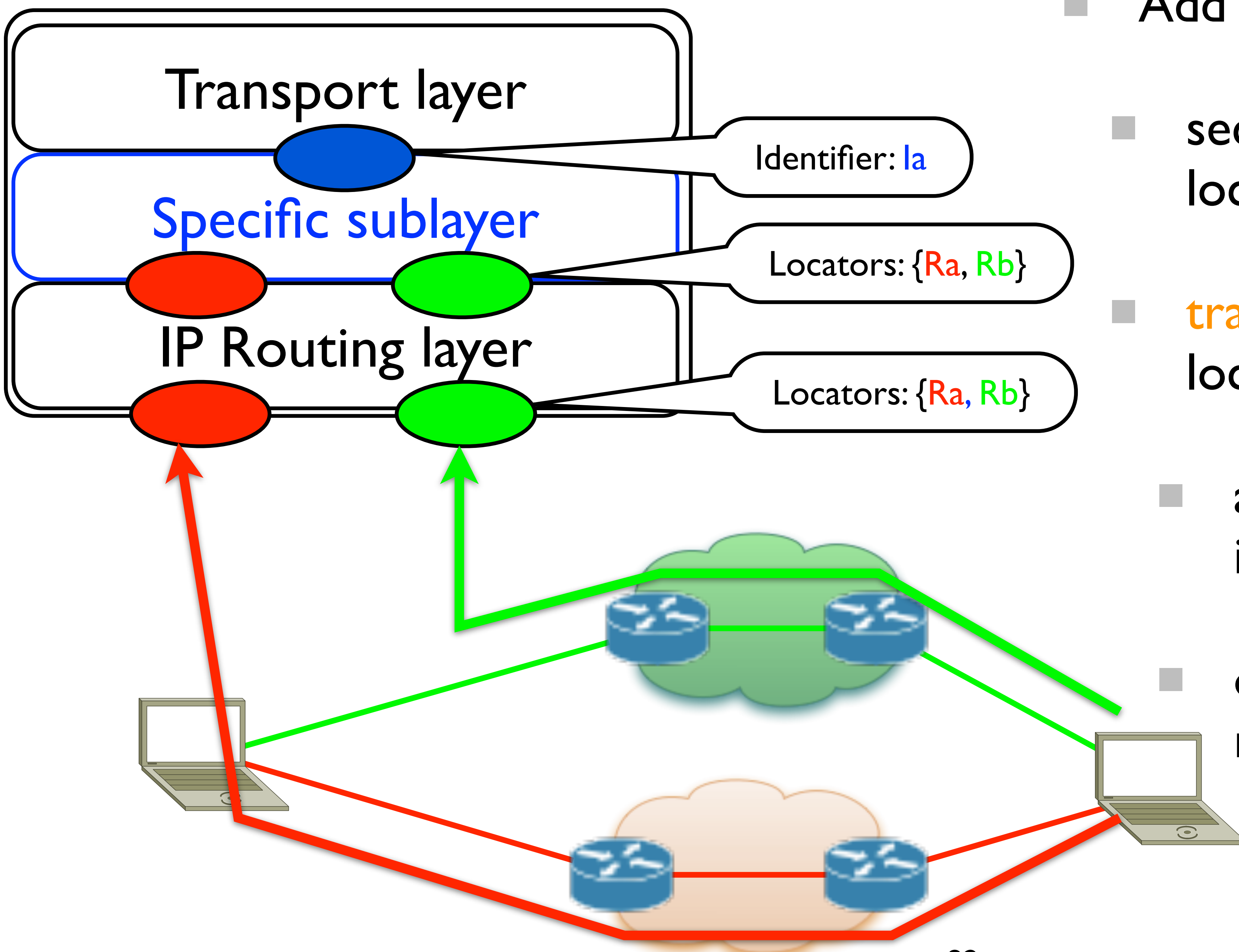
[HIP, ILNP, shim6, Six/One, MPTCP]

Host-Based Locator/ Identifier Separation



[HIP, ILNP, shim6, Six/One, MPTCP]

Host-Based Locator/ Identifier Separation



- Add a shim layer at the host that
  - securely manages the set of locators
  - translates the identifier into locators and vice versa to
    - always exhibit the same host identifier to the transport layer
    - only use IP locators at the routing layer

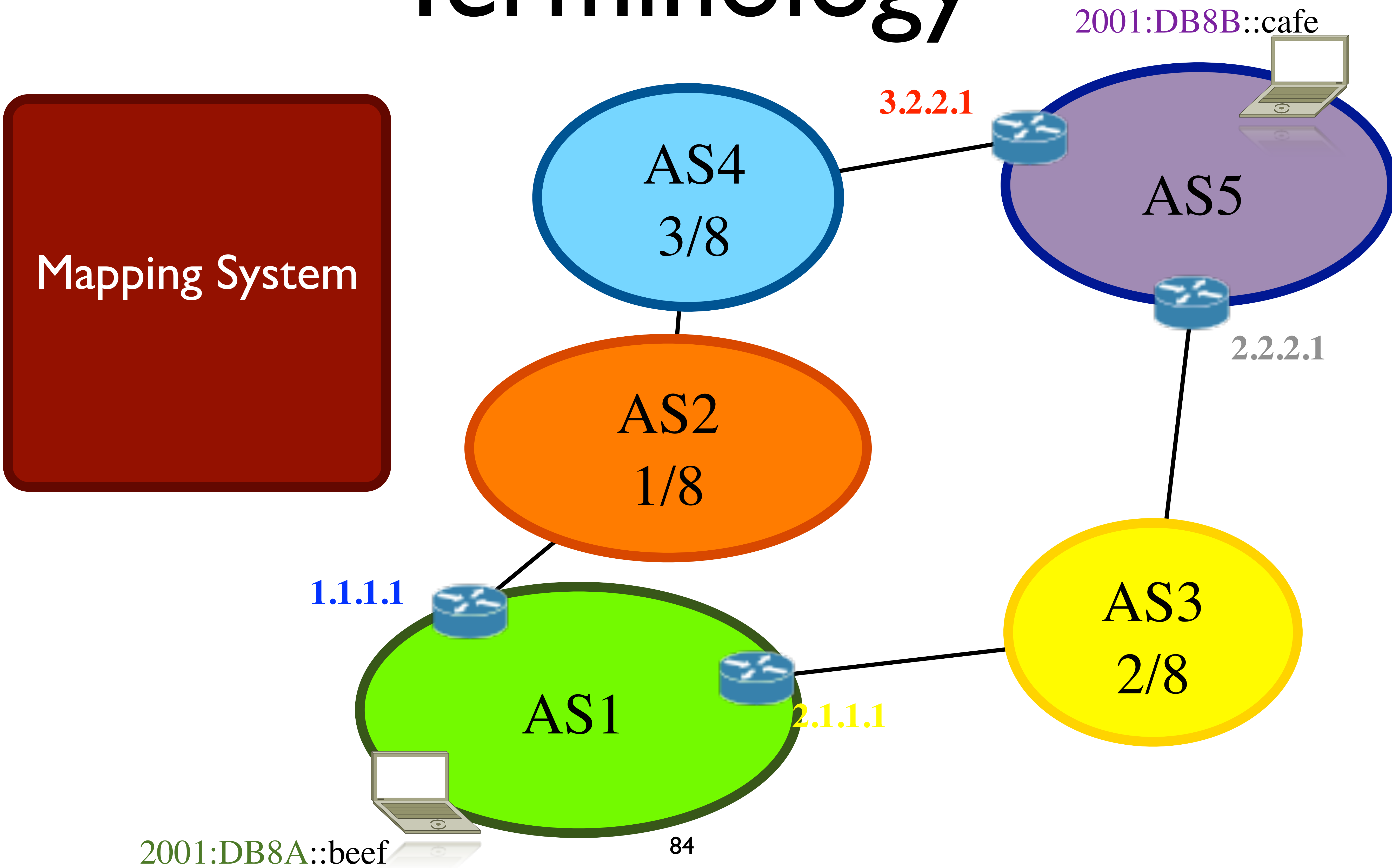
[HIP, ILNP, shim6, Six/One, MPTCP]

Network-Based Locator/ Identifier Separation

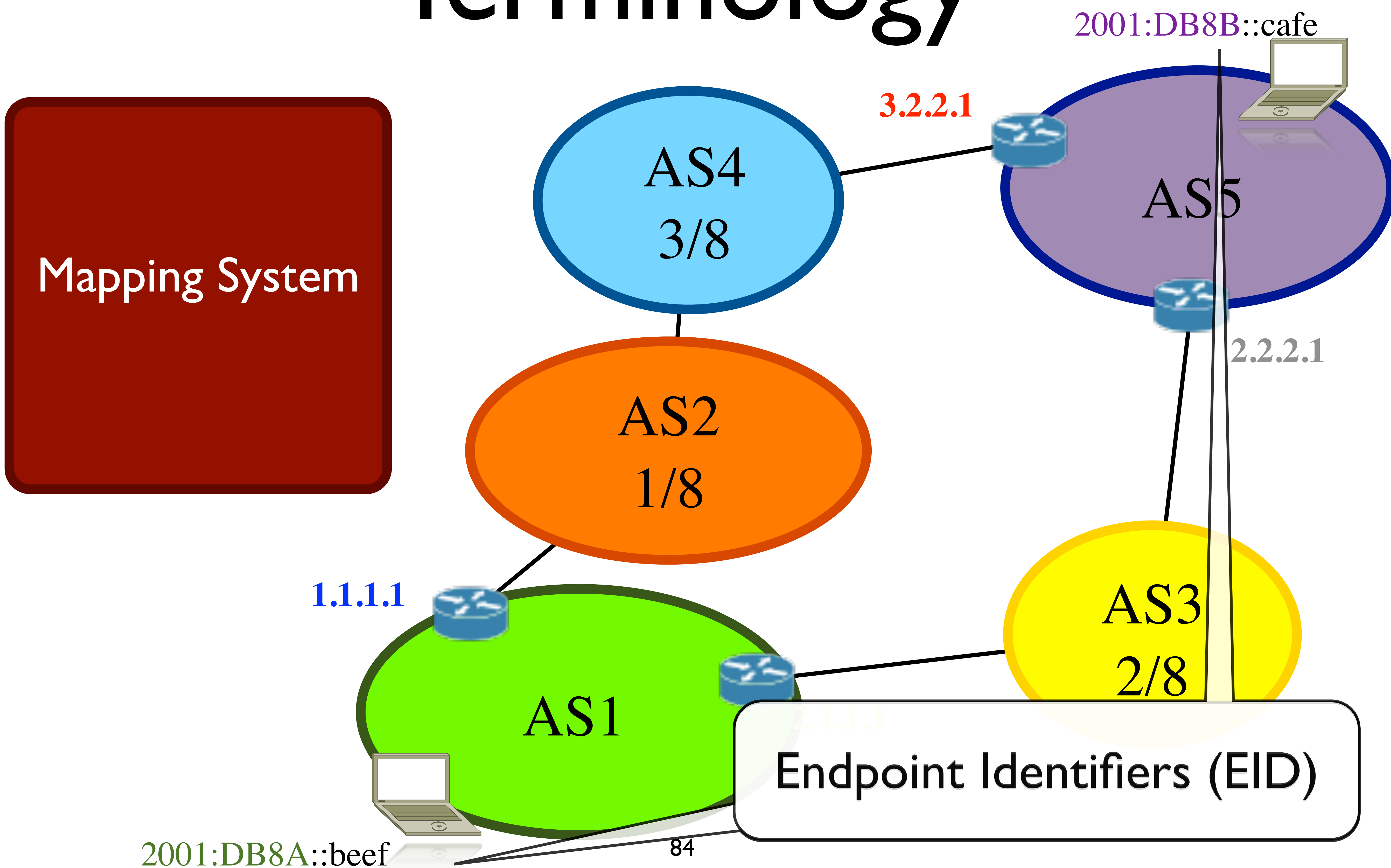
- Each core router owns globally routable addresses used as Routing LOCators (RLOC)
- Each host owns a locally (not globally) routable address used as Endpoint IDentifier (EID)
- The edge/core border routers are in charge of the core/edge separation with the help of three mechanisms to
  - map identifiers onto locators
  - modify packets received from the edge before sending them to the core to route them on top of locators
  - modify packets received from the core before sending them to the edge to route them on top of identifiers

[LISP, GSE, IVIP]

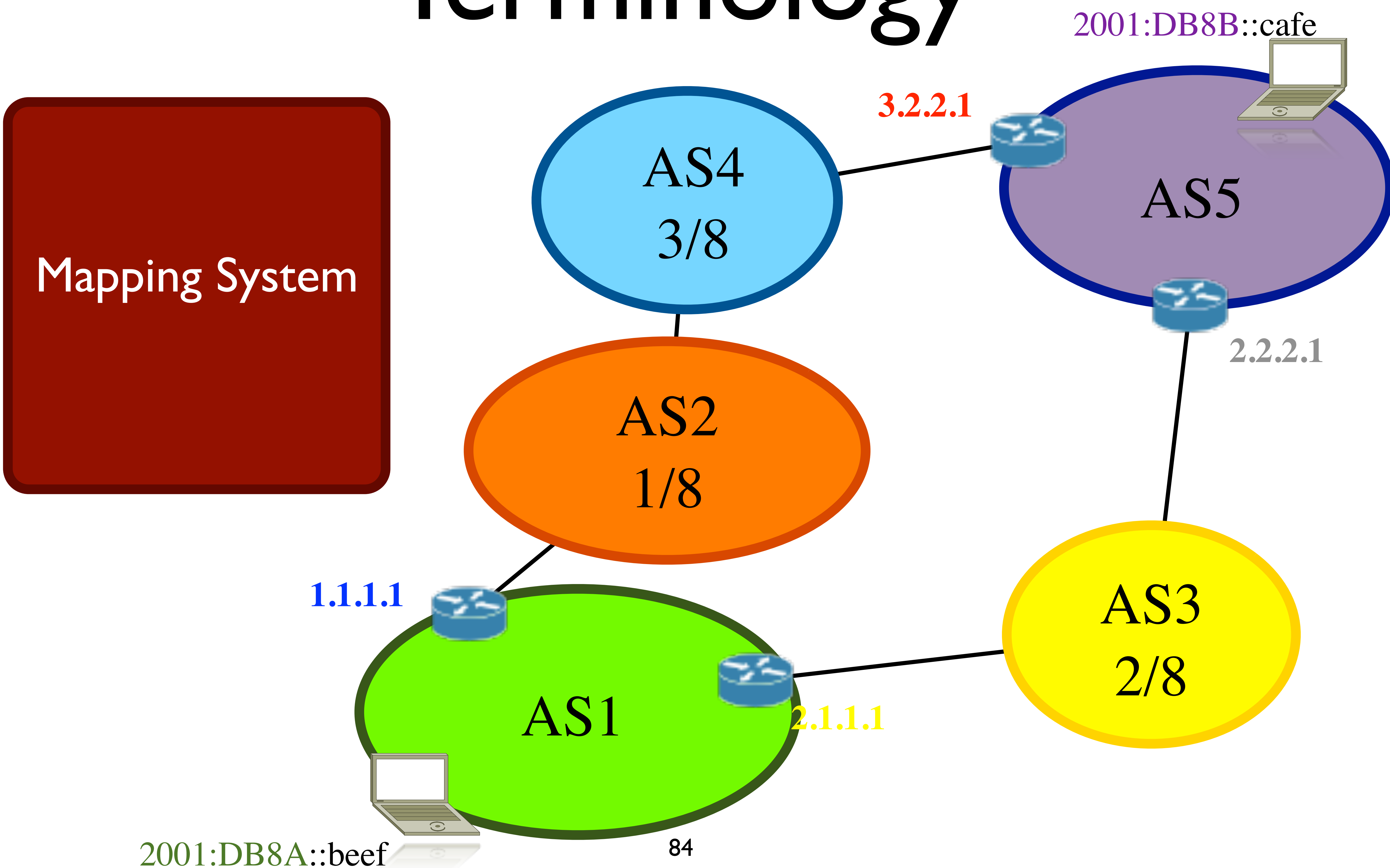
Terminology



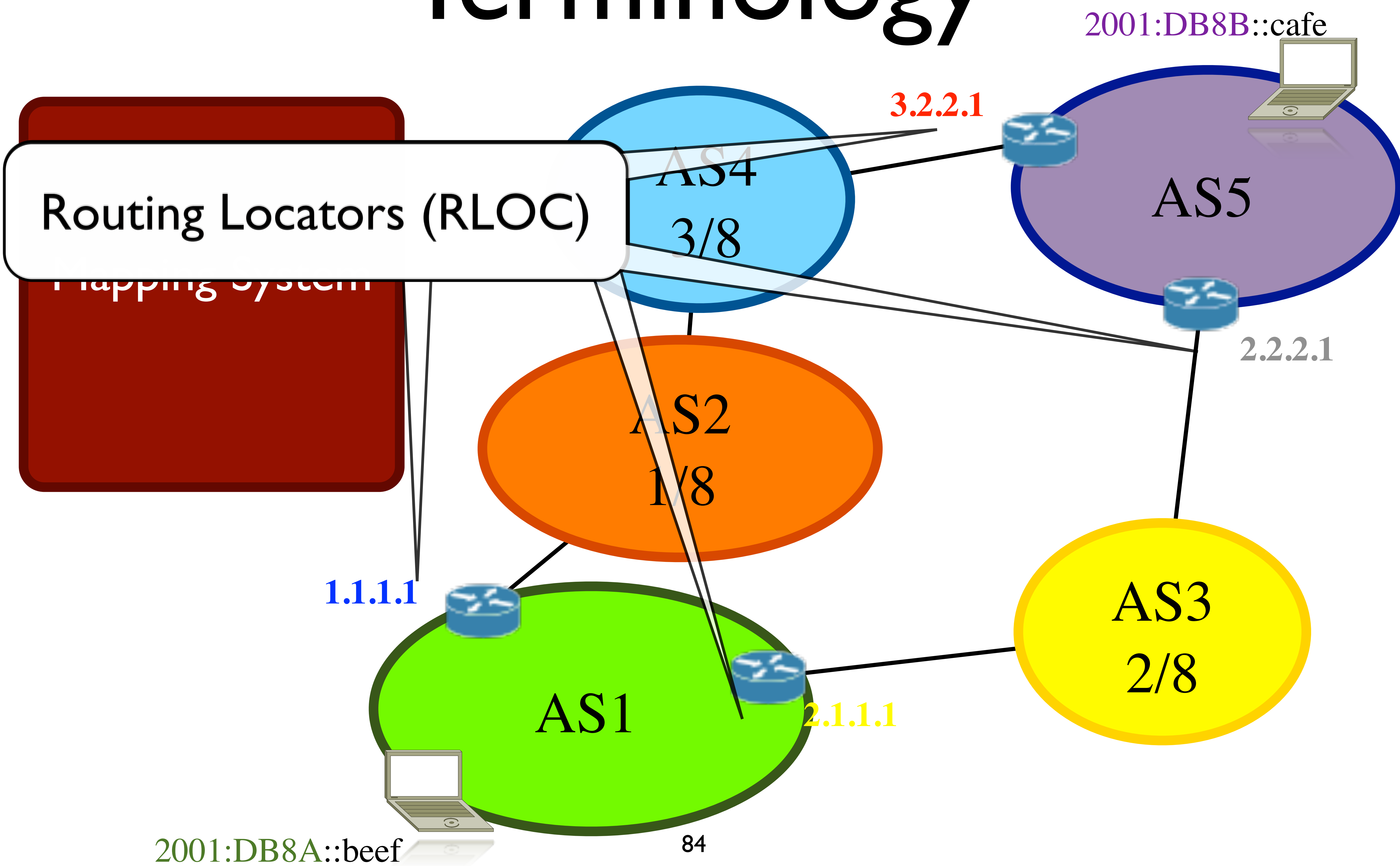
Terminology



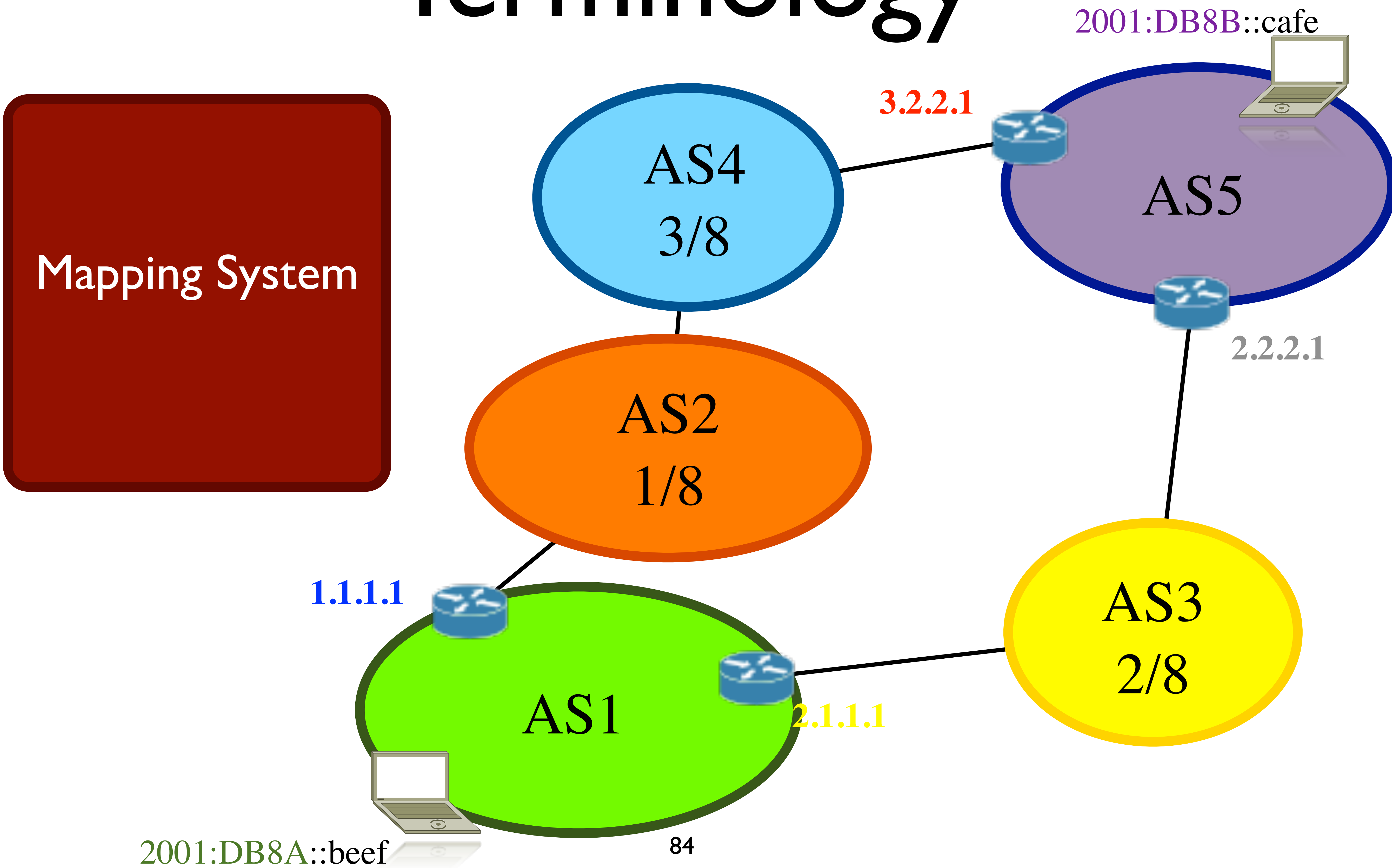
Terminology



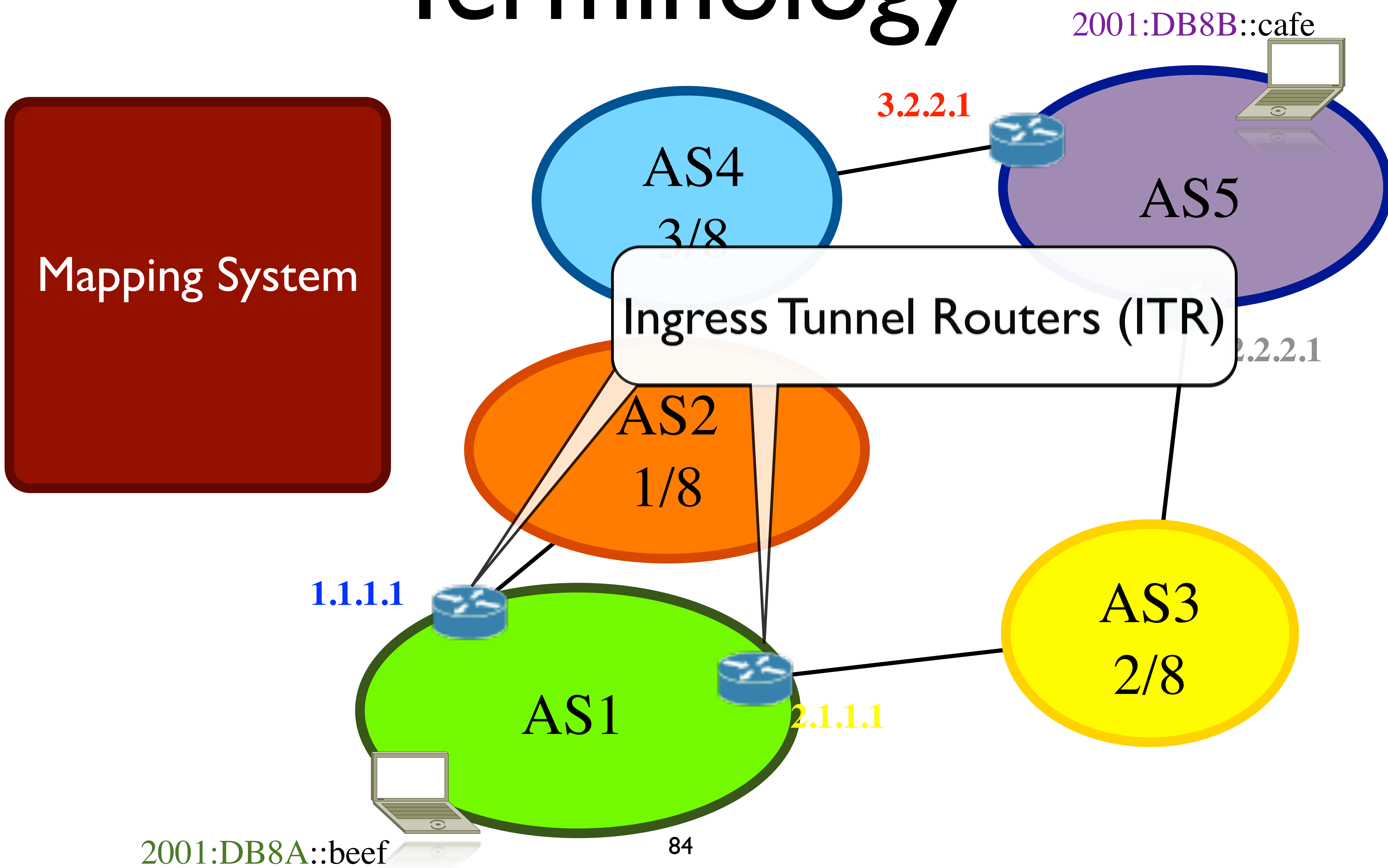
Terminology



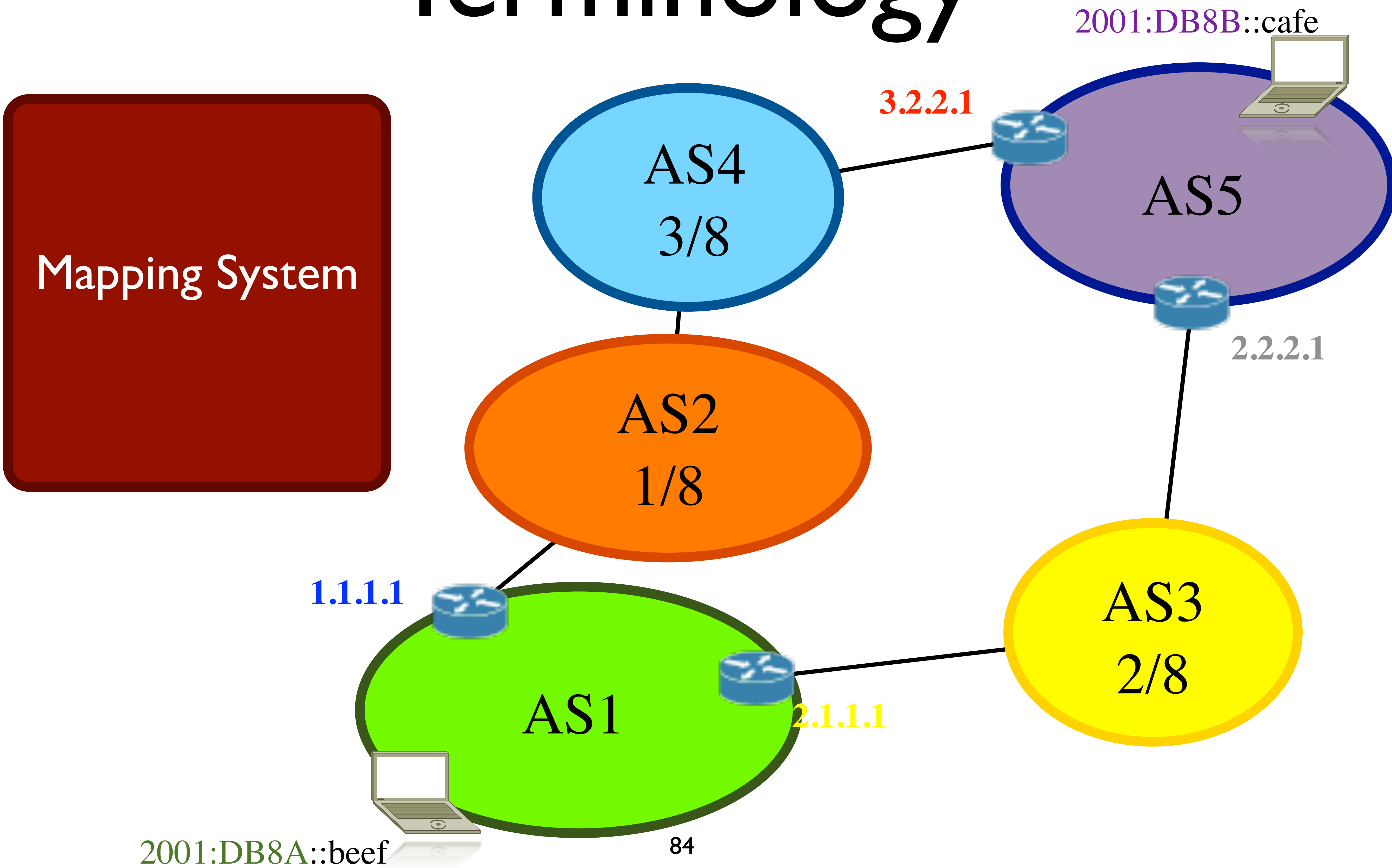
Terminology



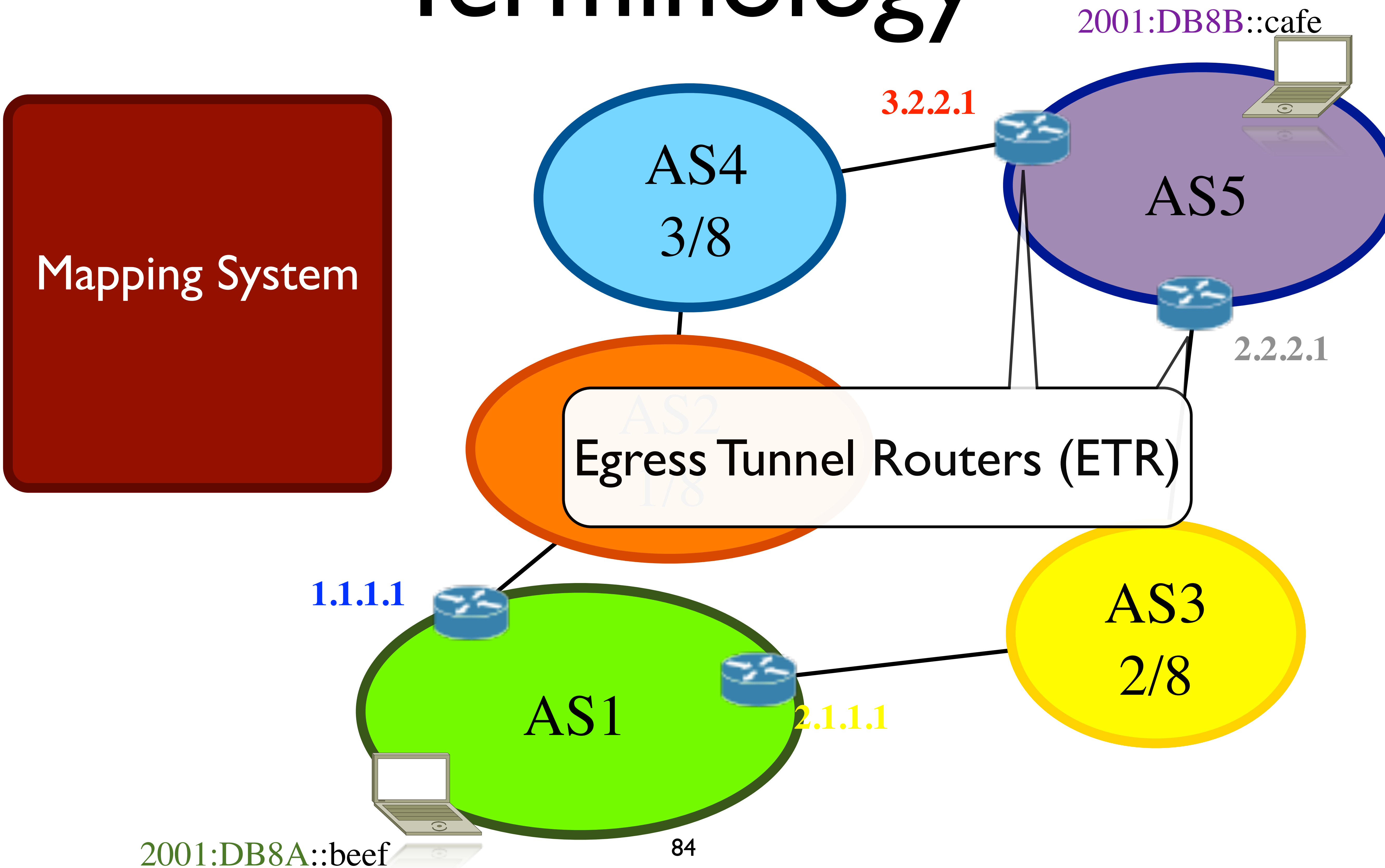
Terminology



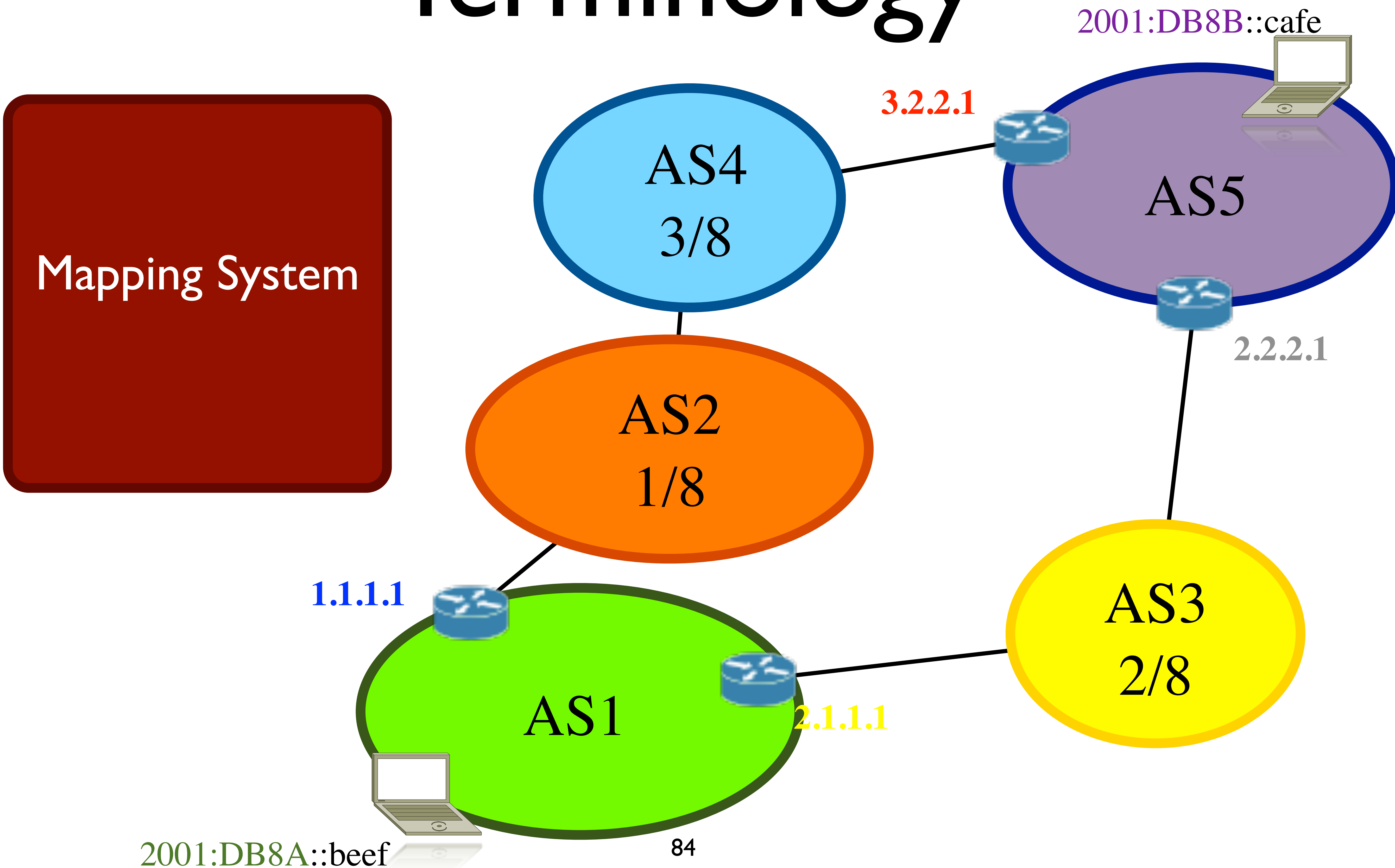
Terminology



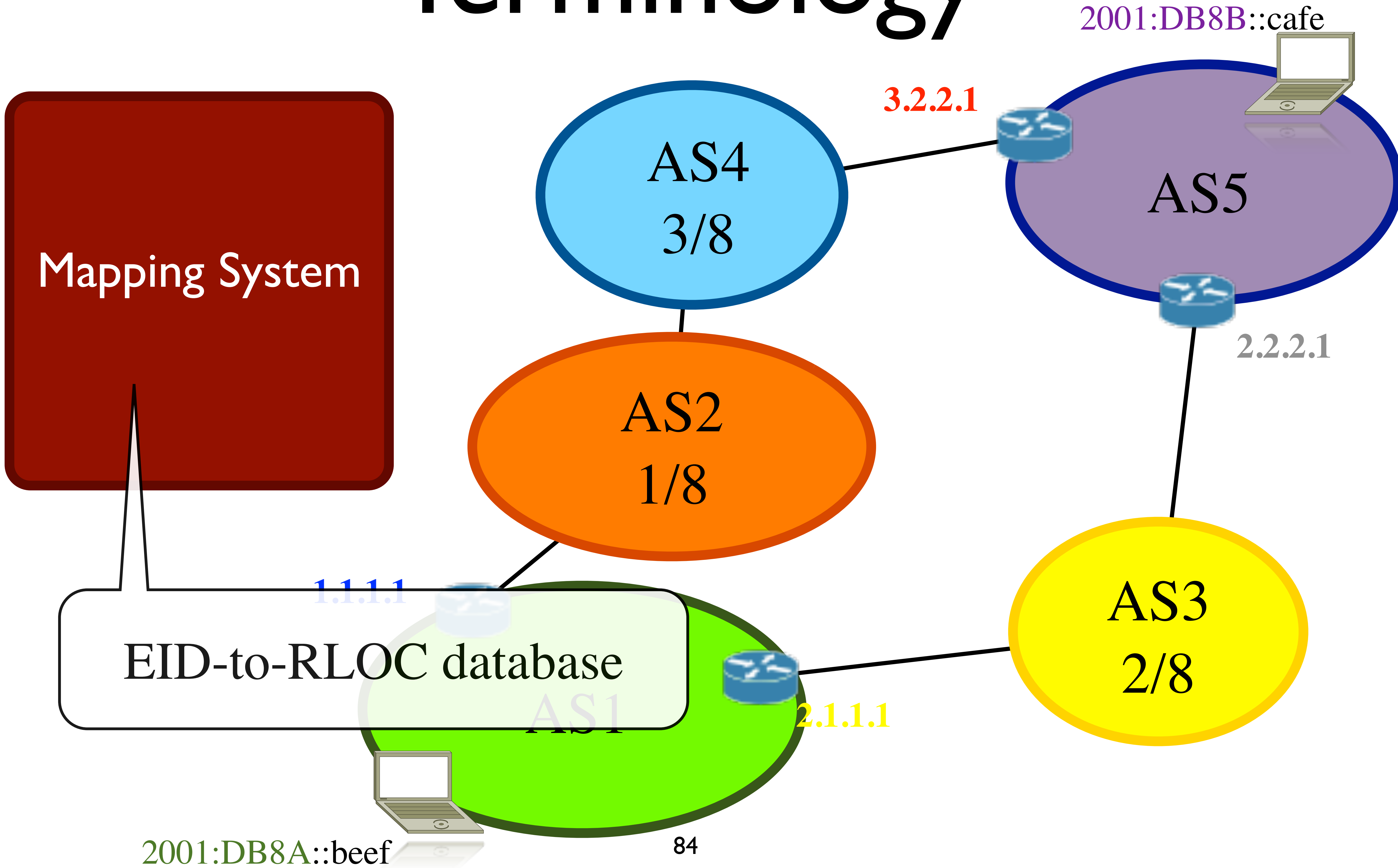
Terminology



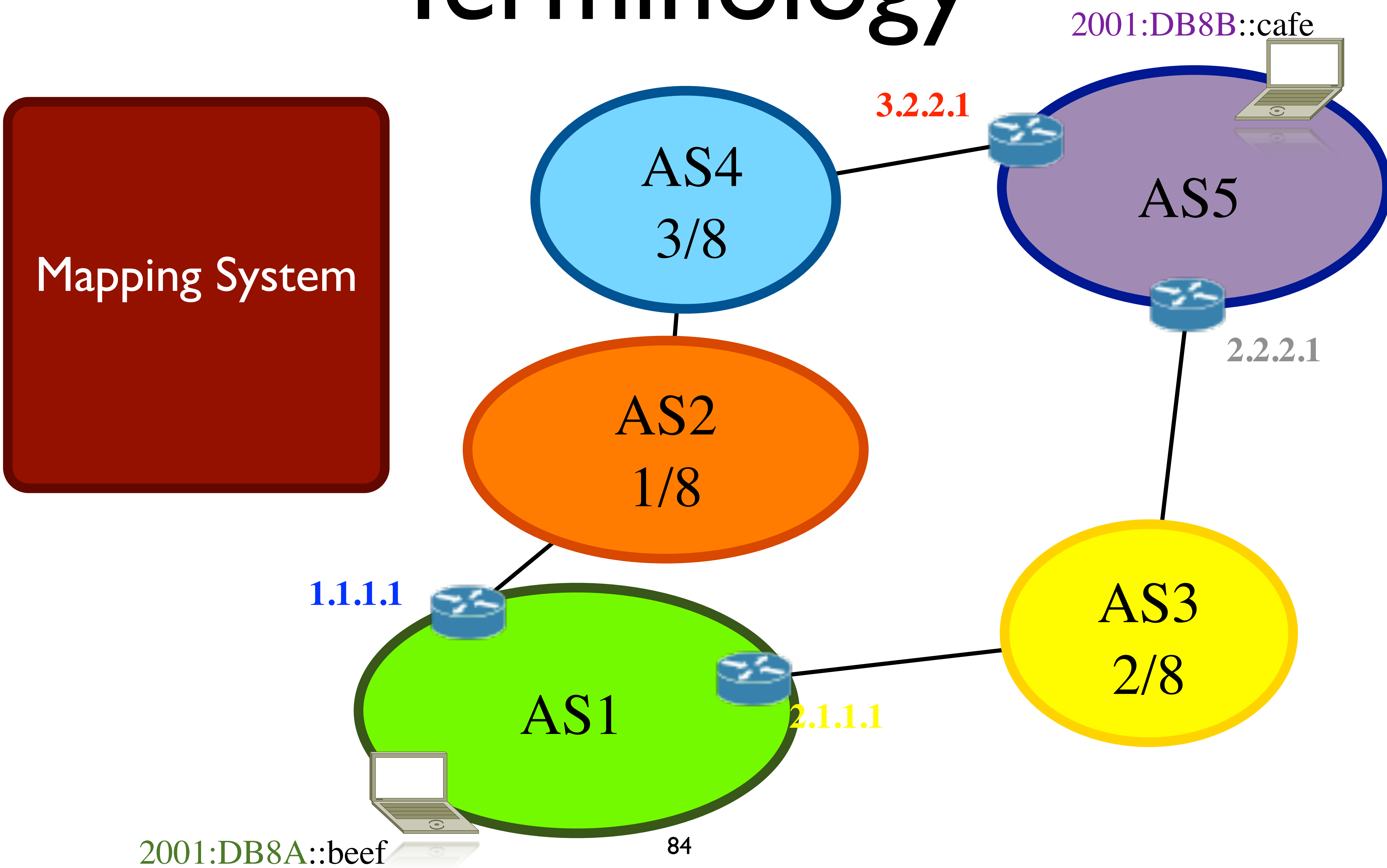
Terminology



Terminology



Terminology

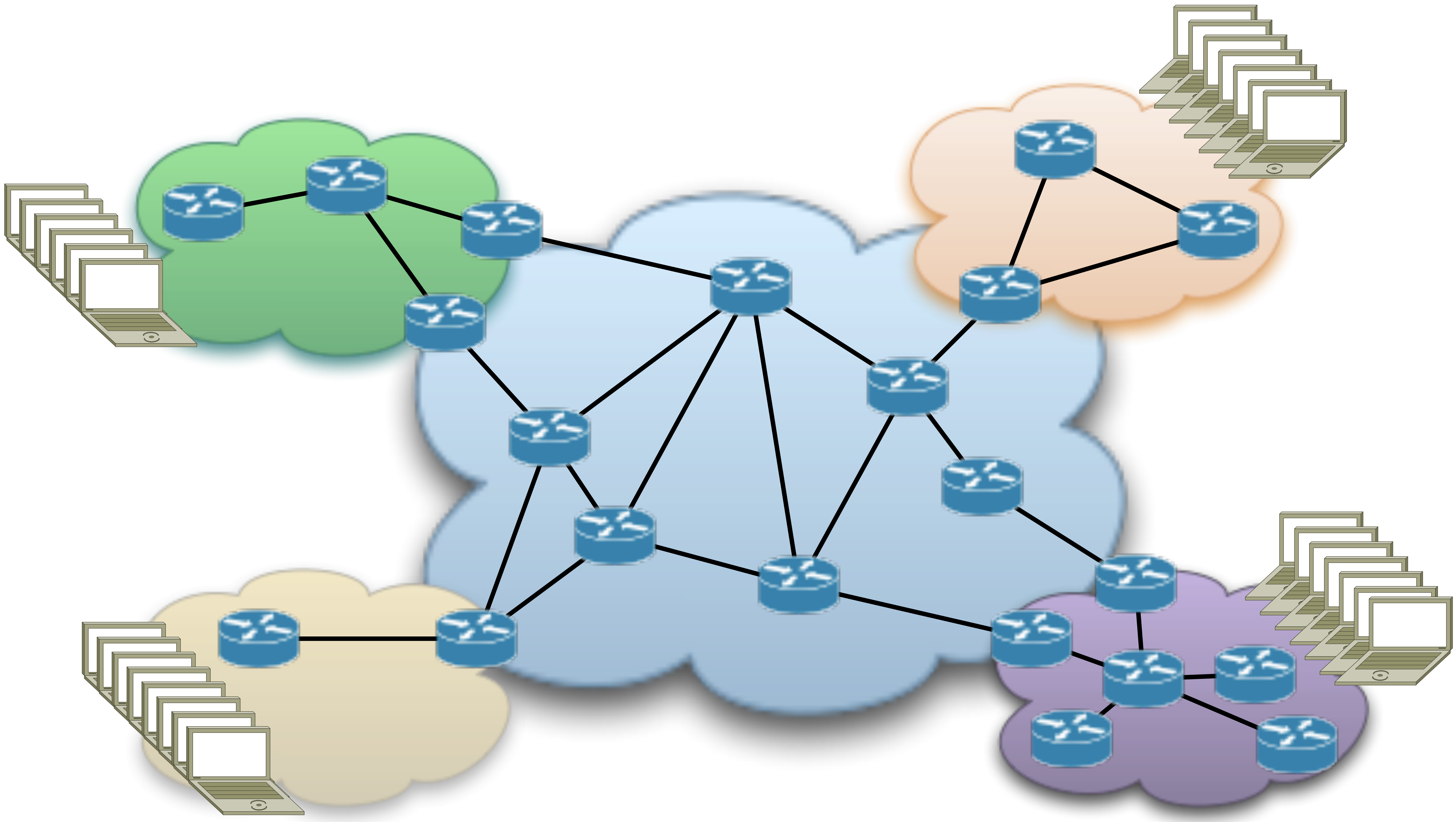


Motivation

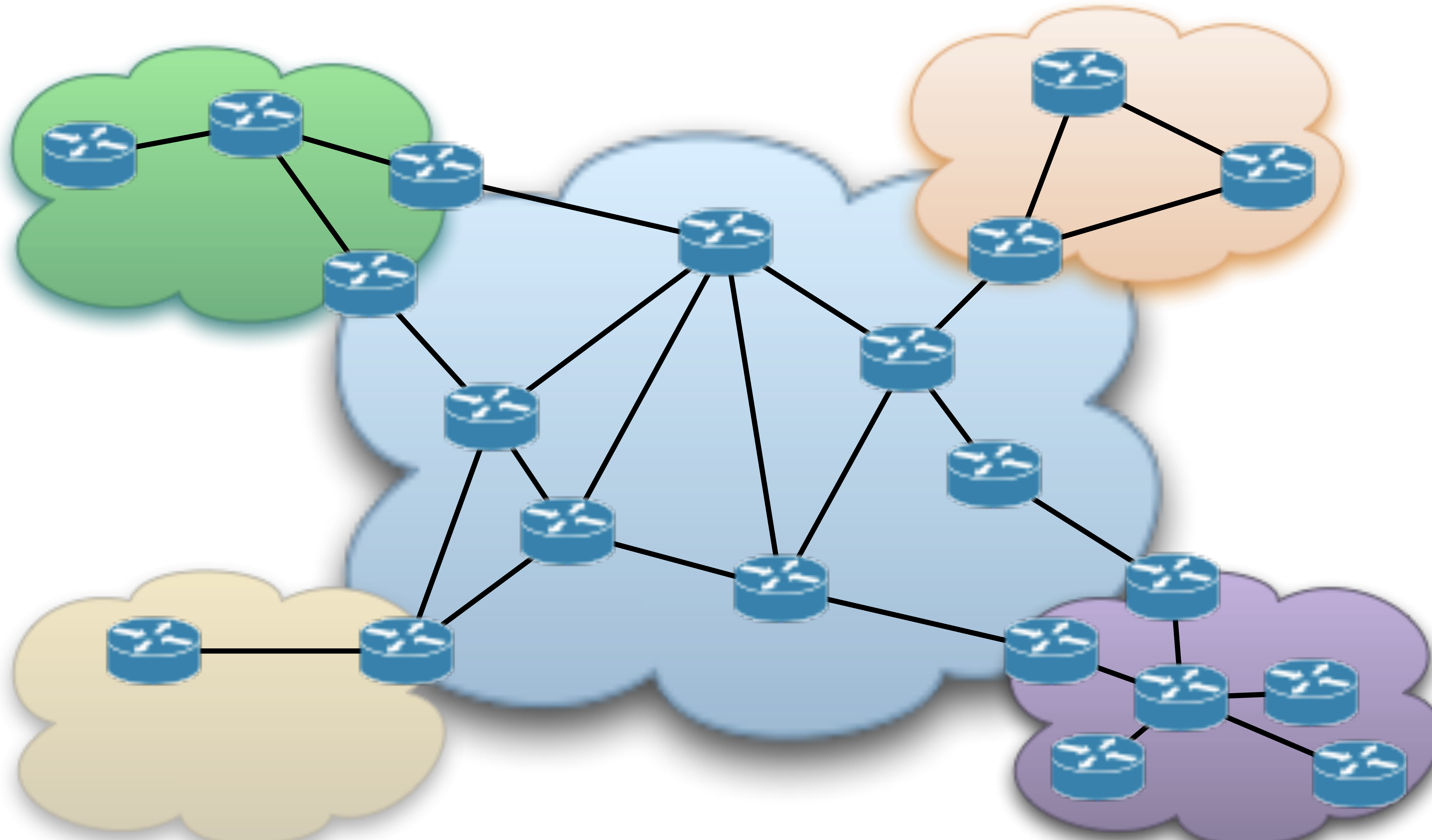
Questions before starting

- Do we have inter-domain path diversity?
  - multihoming
  - multi-connectivity
- Are all the paths equal?

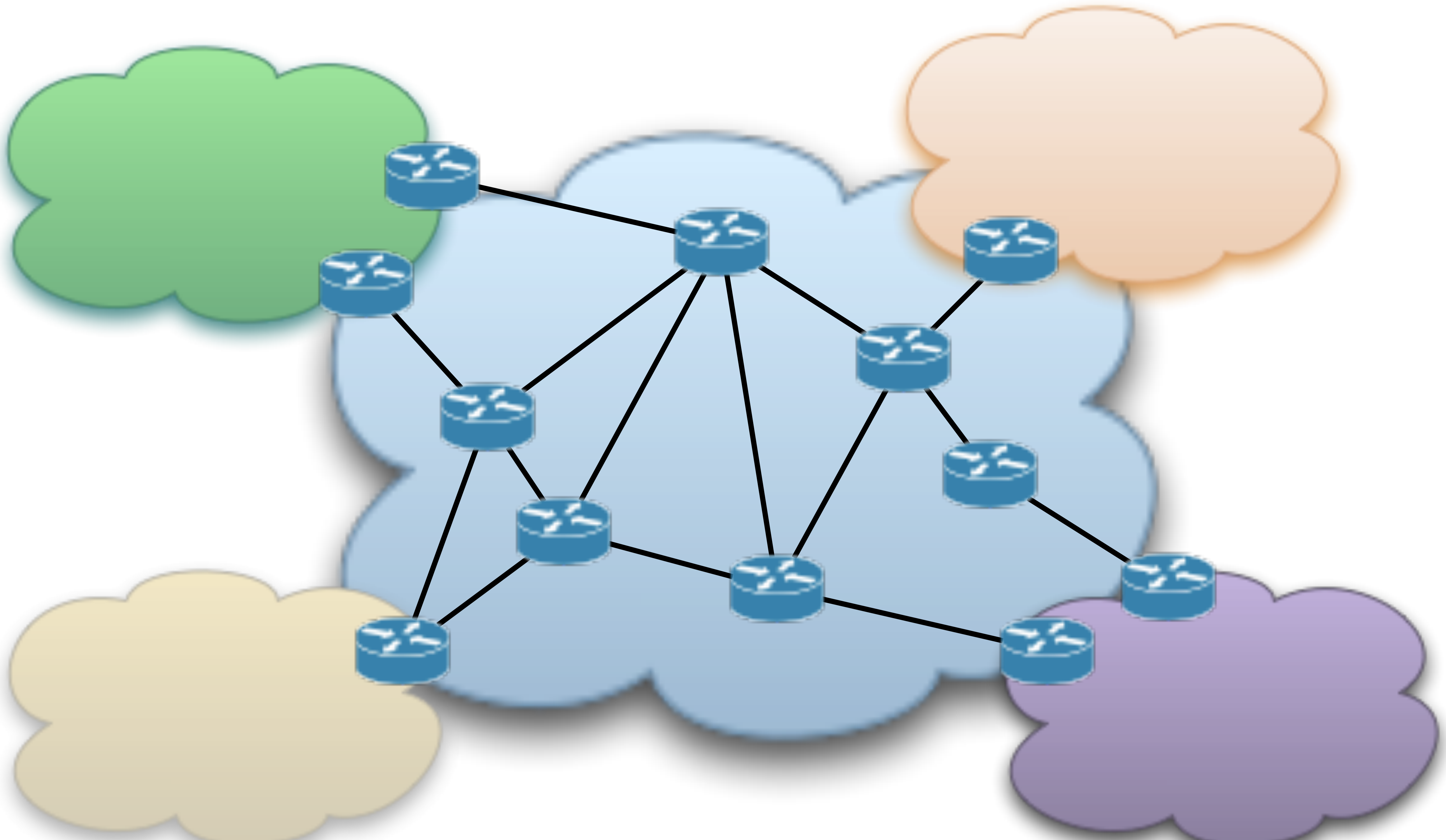
Methodology



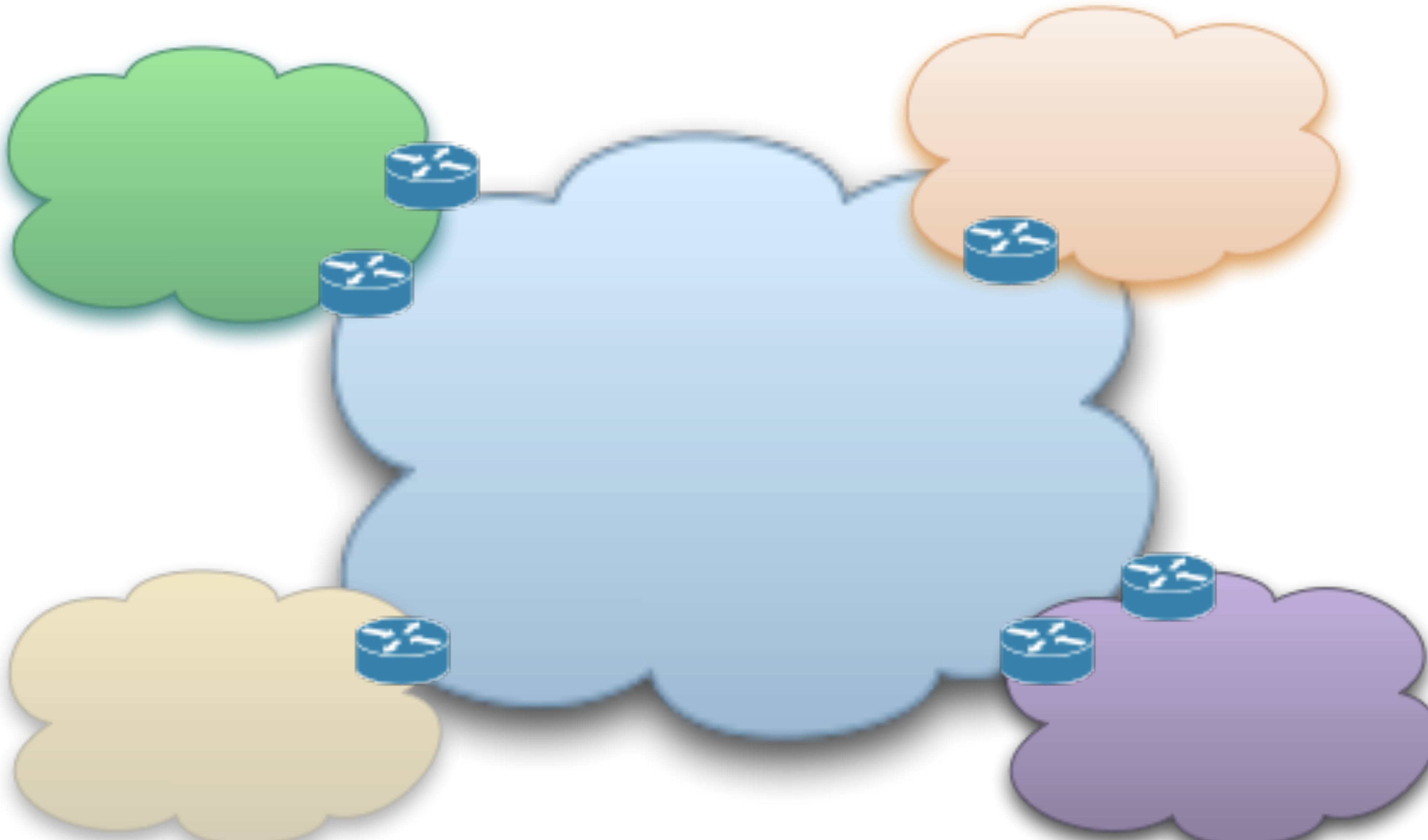
Methodology



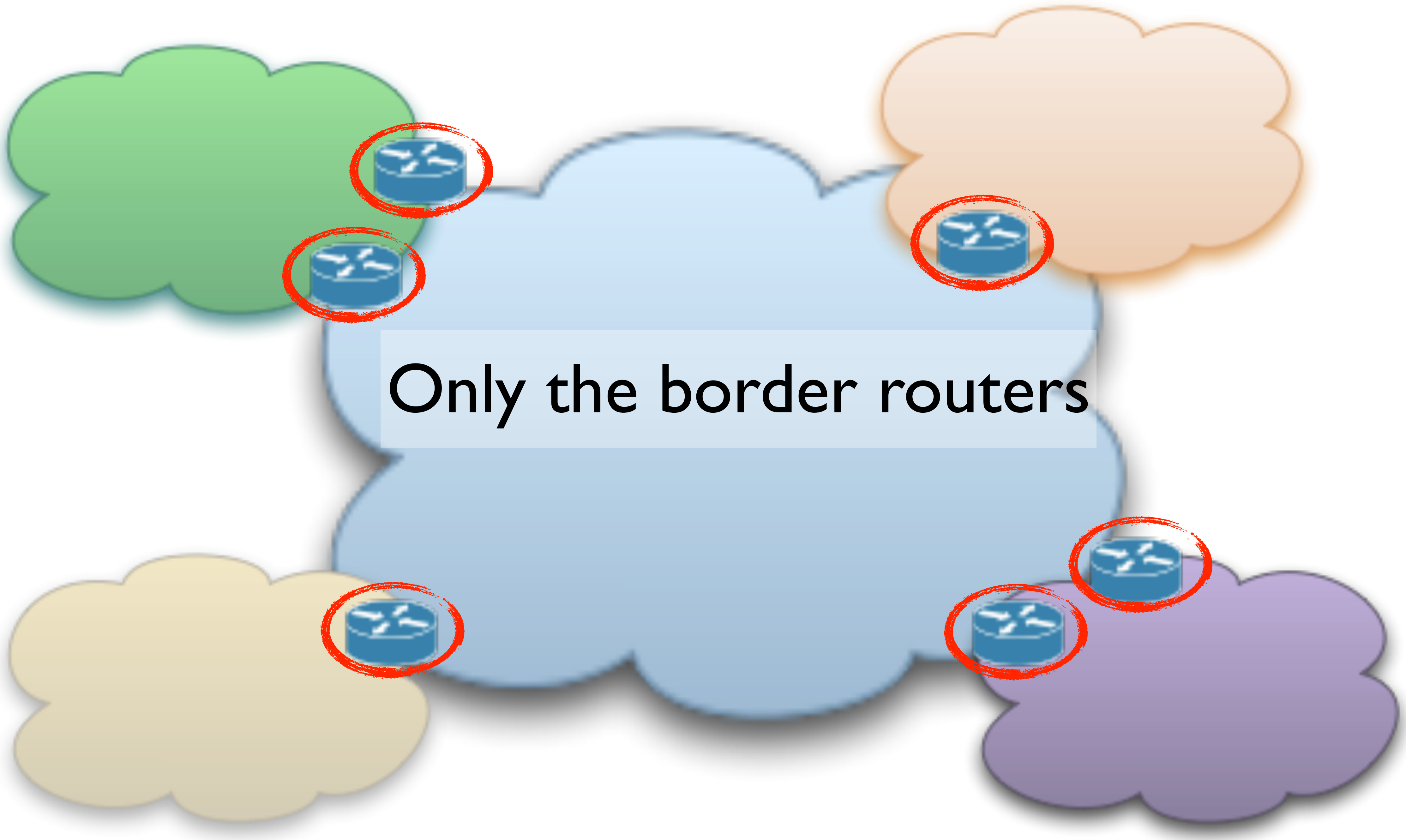
Methodology



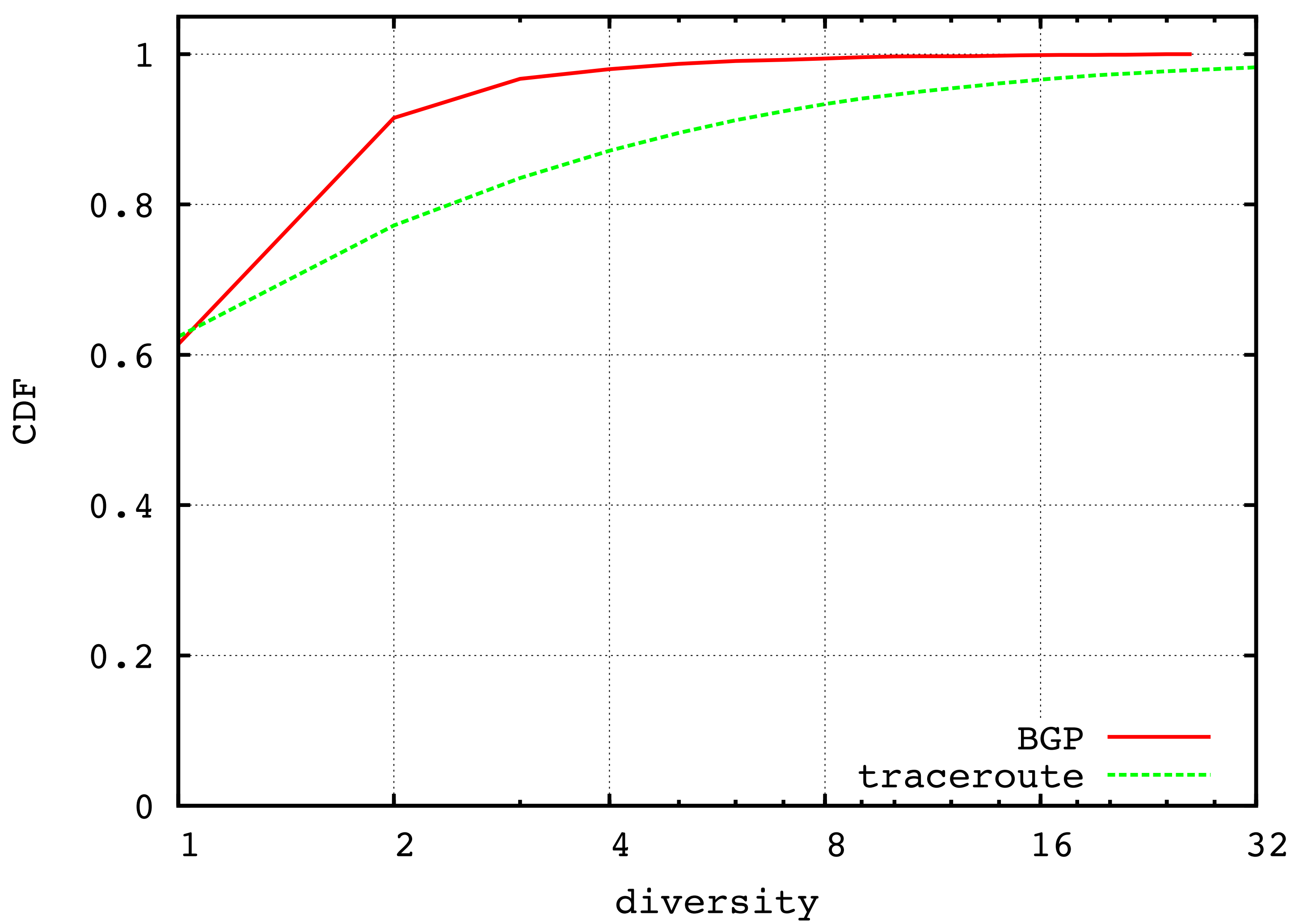
Methodology



Methodology

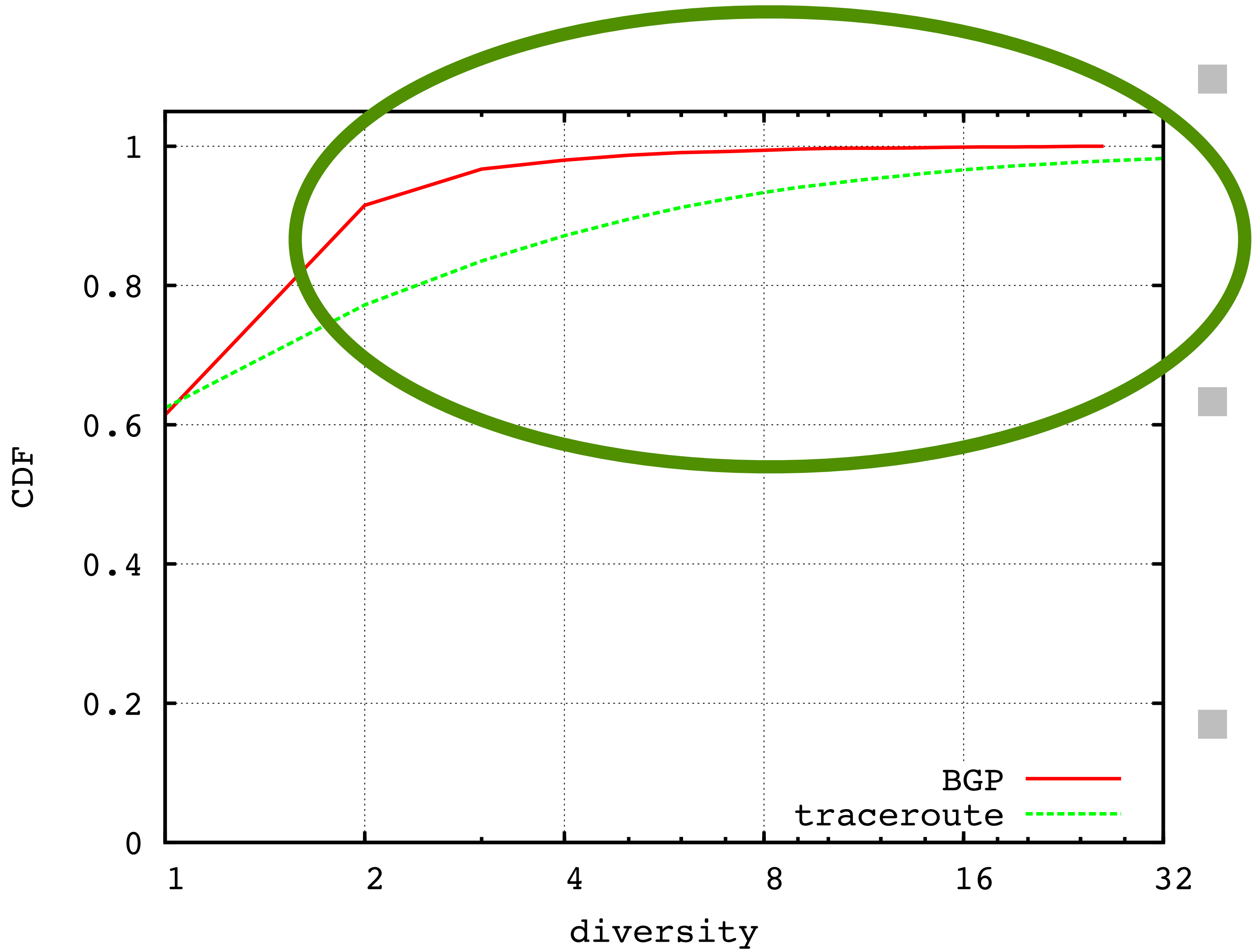


Multi-* is common even at the stubs



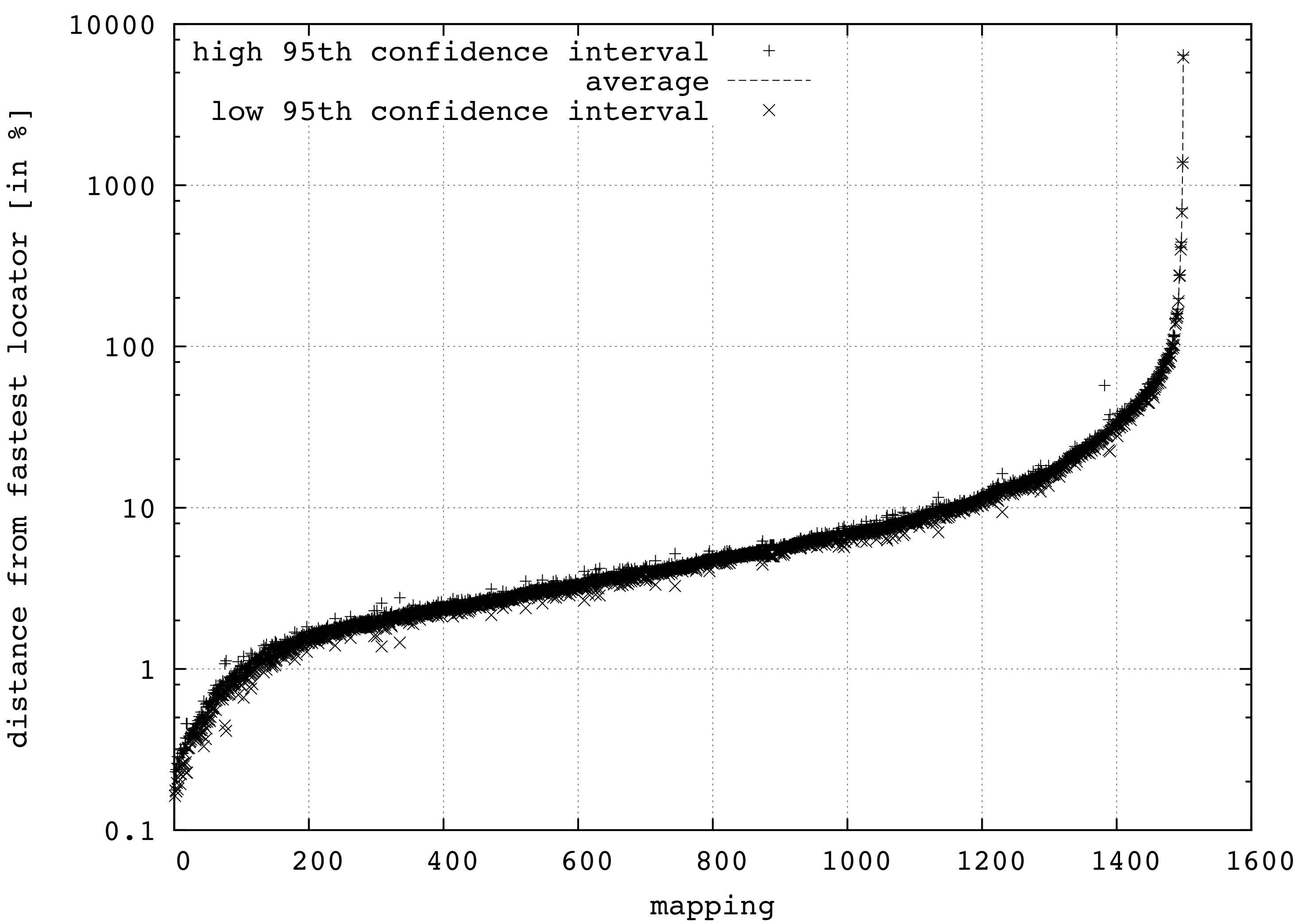
- More than 25% have at least two logically distinct entry points
- Some have even more than hundreds of distinct entry points
- Are the paths to these entry paths equal?

Multi-* is common even at the stubs



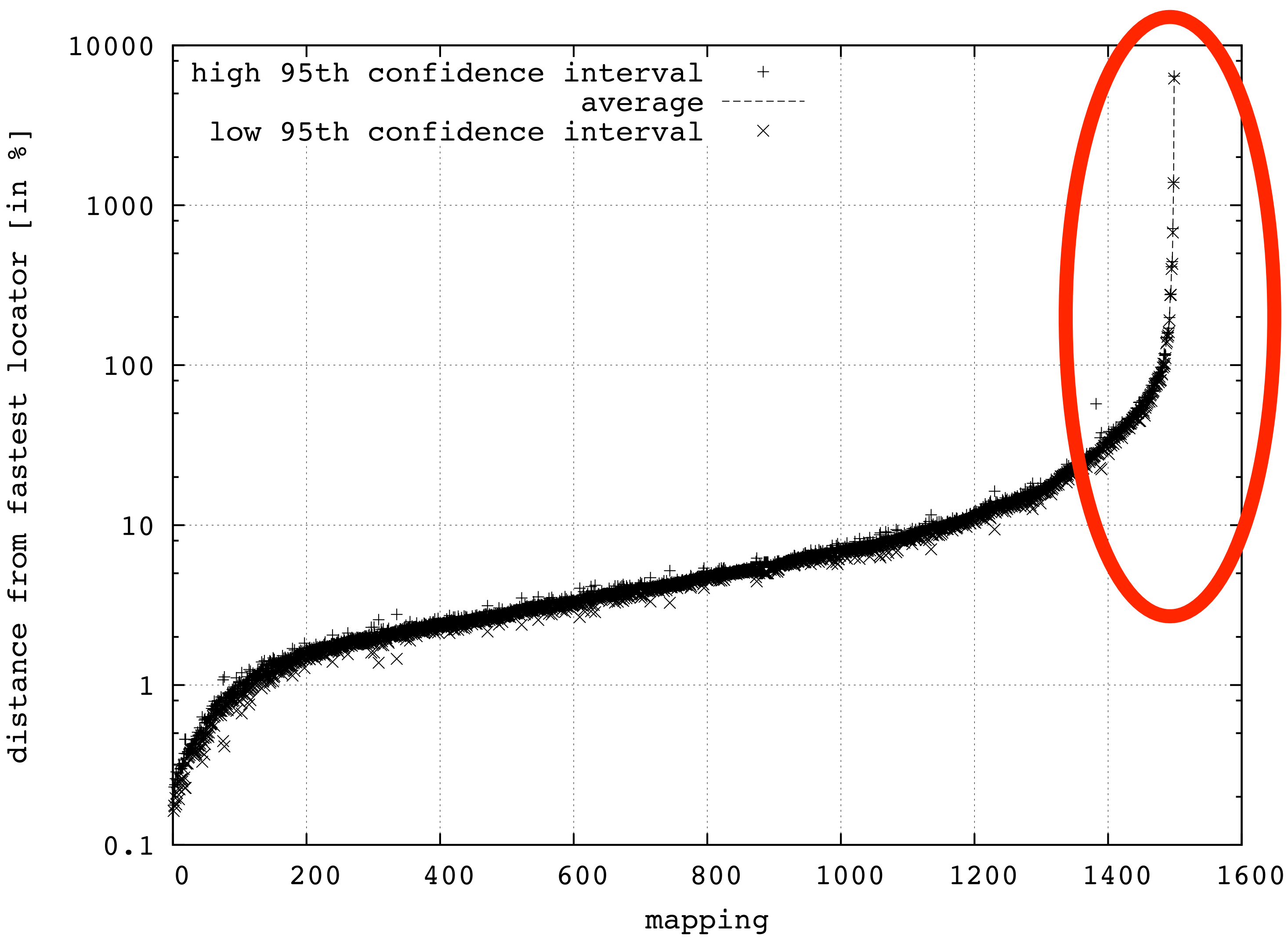
- More than 25% have at least two logically distinct entry points
- Some have even more than hundreds of distinct entry points
- Are the paths to these entry paths equal?

Choosing the wrong path can be very bad



- Slowest paths can be more than 25% slower than the fastest
  - a good path choice can significantly improve the delay

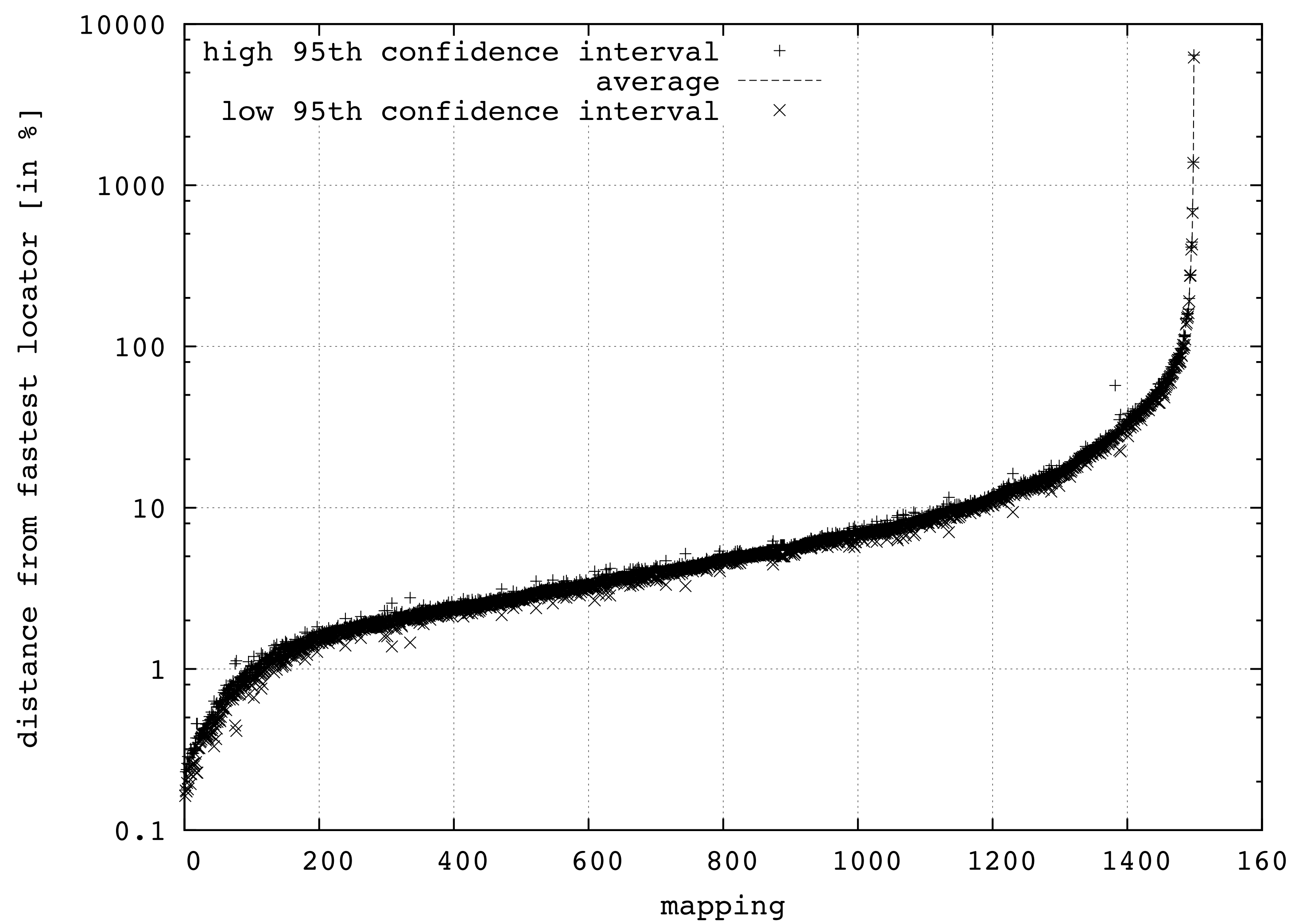
Choosing the wrong path can be very bad



- Slowest paths can be more than 25% slower than the fastest
  - a good path choice can significantly improve the delay

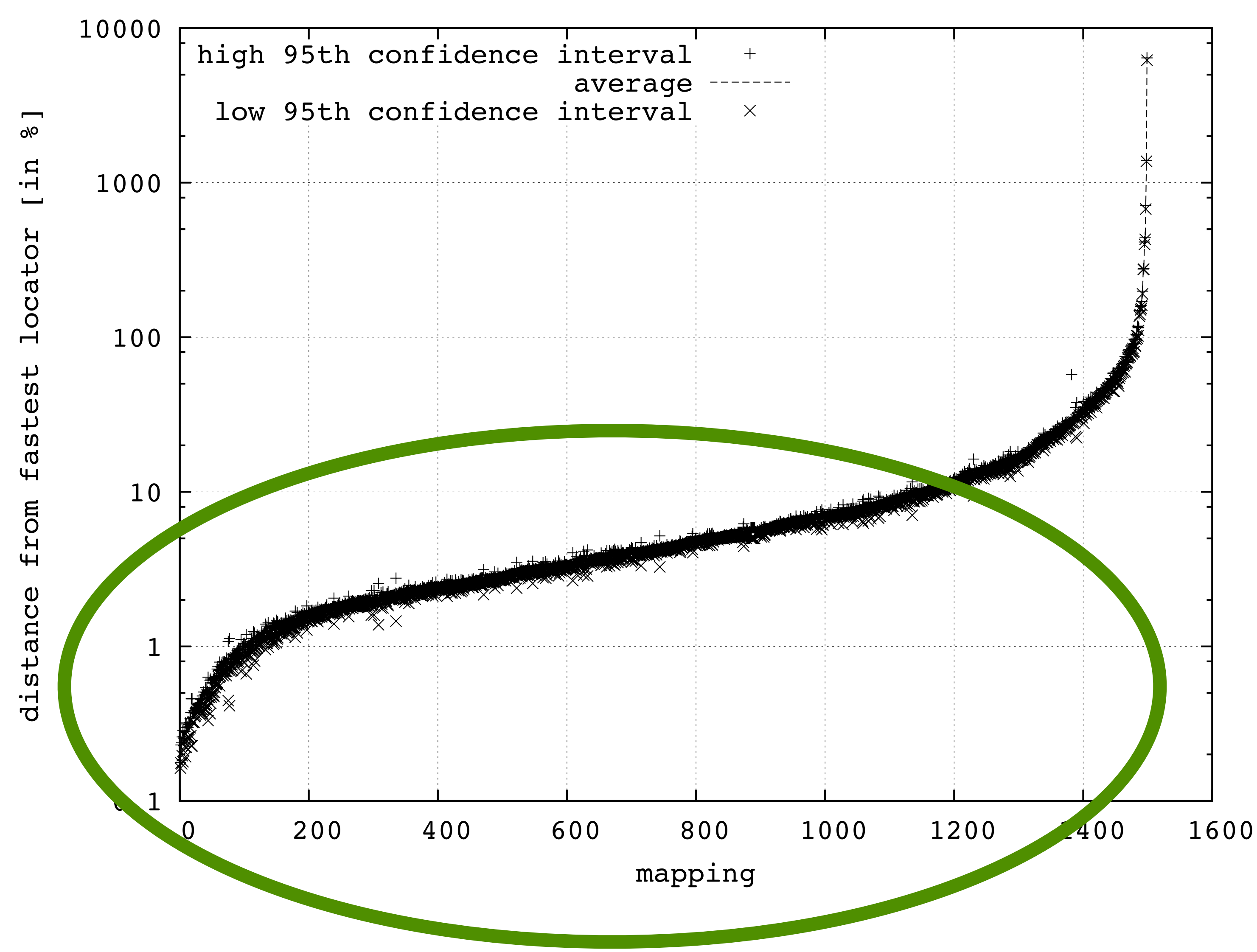
Good inter-domain load balancing is feasible

- Difference between the fastest paths may be less than 10%
  - load balancing with no performance drop (e.g., TCP)

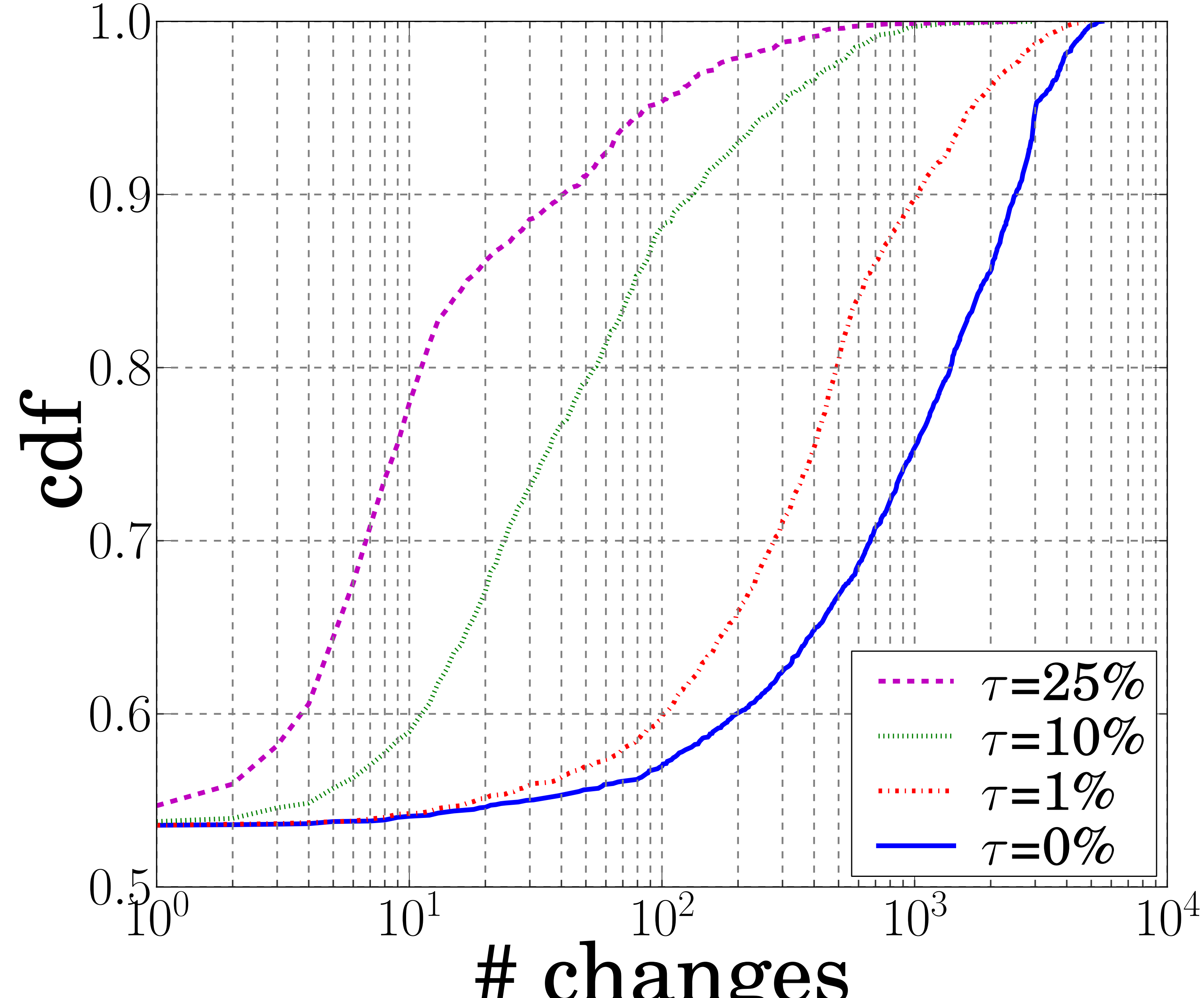


Good inter-domain load balancing is feasible

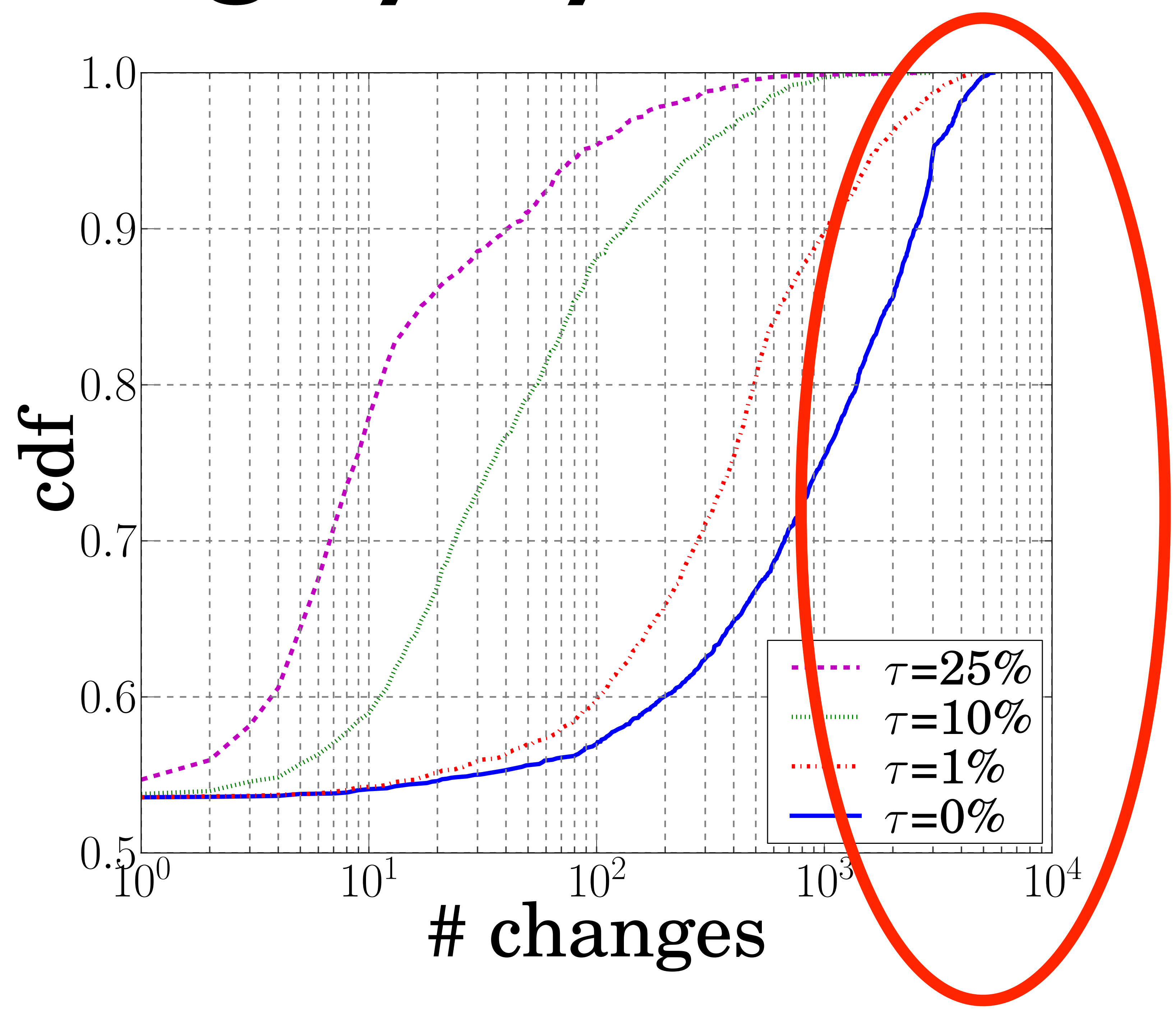
- Difference between the fastest paths may be less than 10%
  - load balancing with no performance drop (e.g., TCP)



Performance can be highly dynamic



Performance can be highly dynamic



Performance based incoming traffic engineering challenge

- We need a service able to
  - measure/predict paths performance
  - decide paths to use
  - enforce source to follow the destination decisions

Requirements

- The incoming TE service must be
  - auto adaptive to network changes
  - flexible for operational policies
  - incrementally deployable on the Internet

Terminology

- Mapping System: a globally decentralized database that contains all know EID-prefix to RLOC mappings and the mechanisms to distribute them
- LISP Cache: EID-to-RLOC Database stored at the ITR
- LISP Database: EID-to-RLOC Database stored at the ETR

LISP Main Design Goals

- Minimize required changes to Internet
- No end-systems (hosts) changes
- Be incrementally deployable
- No router hardware changes
- Minimize router software changes
- Network-Based Locator/Identifier Separation relying on the Map-and-Encap paradigm

Terminology

- Ingress Tunnel Router (*ITR*): a router which accepts a packet containing a single IP header. The router maps the destination address of the packet to an RLOC and prepends a LISP header before forwarding the encapsulated packet.
- Egress Tunnel Router (*ETR*): a router which accepts a LISP encapsulated packet. The router strips the LISP header and forwards the packet based on the next header

Summary

- plenty of paths, with much varying properties
- use the bests!

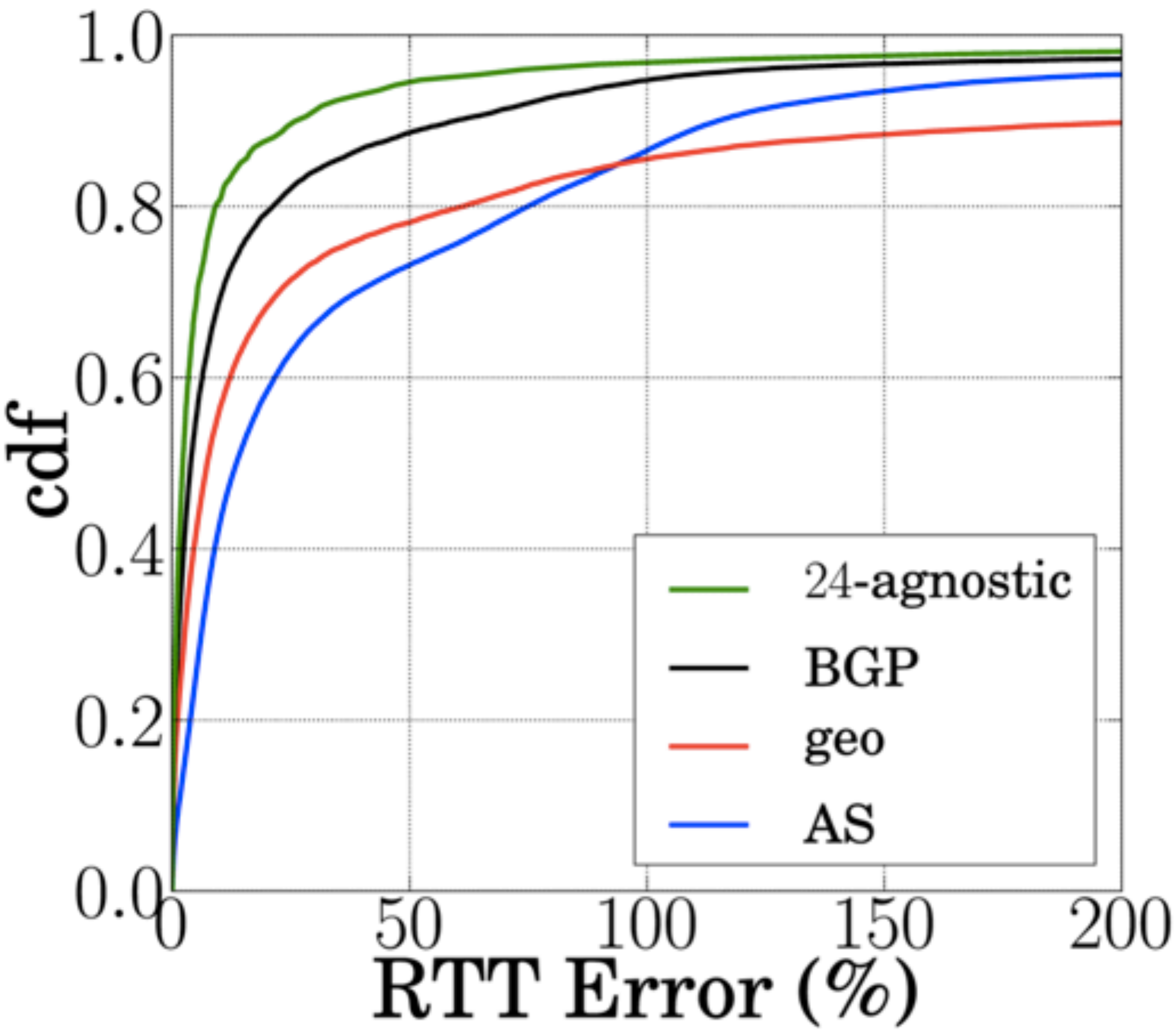
1. predict the performance of each path
2. rank paths towards the same destinations
3. use only the best ranked paths

Cost function example

- Maximize the bandwidth for premium users
- Minimize the cost for standard users
- Maximize night bandwidth for advanced users but minimize day cost
- Always prefer intra-AS paths when possible

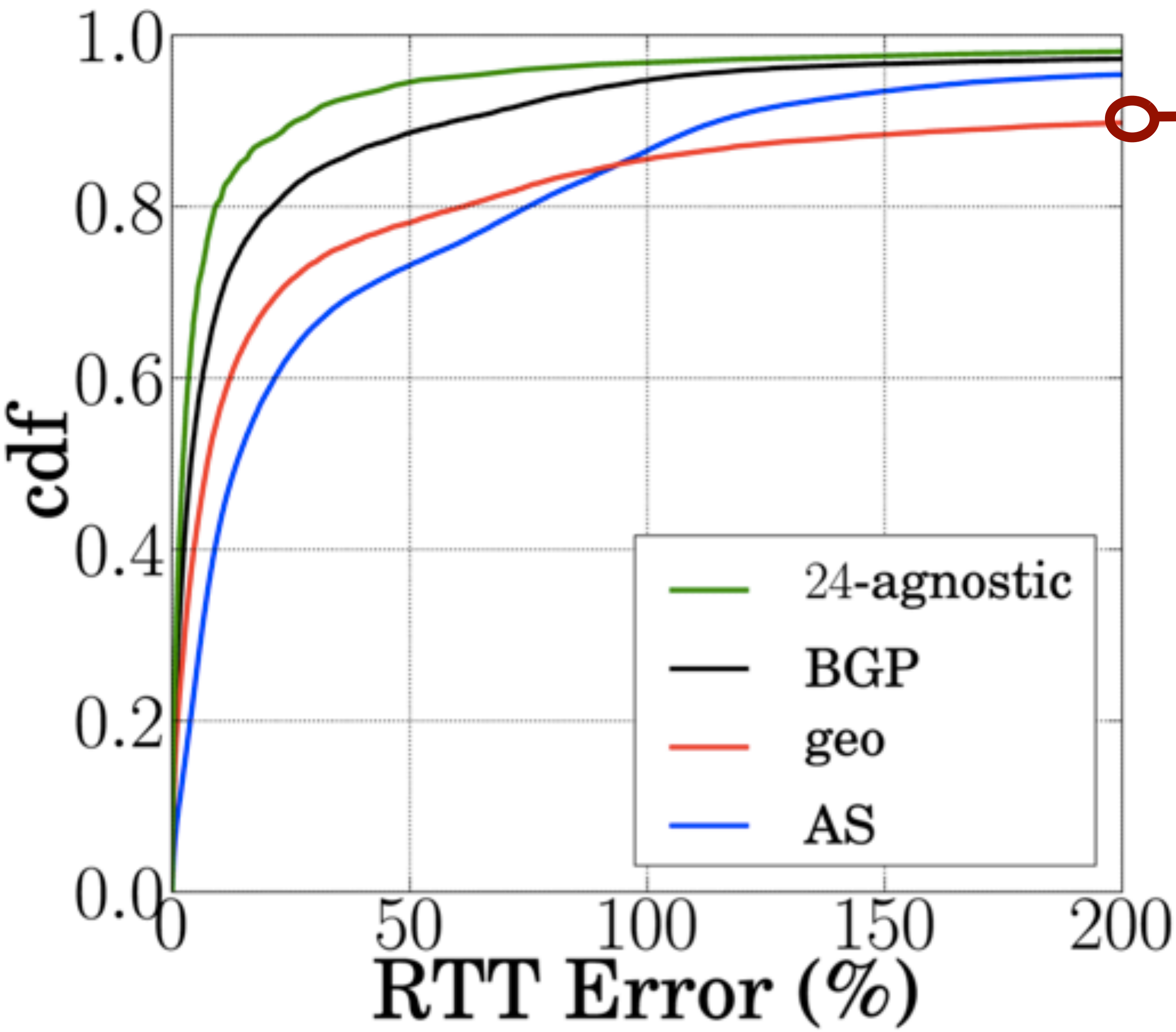
[SDIB08] Saucez, D. et al. Interdomain Traffic Engineering in a Locator/Identifier Separation Context. INM'08

RTT Error



$$e_{ij} = \frac{|m_{ij} - \hat{m}_{ij}|}{m_{ij}}$$

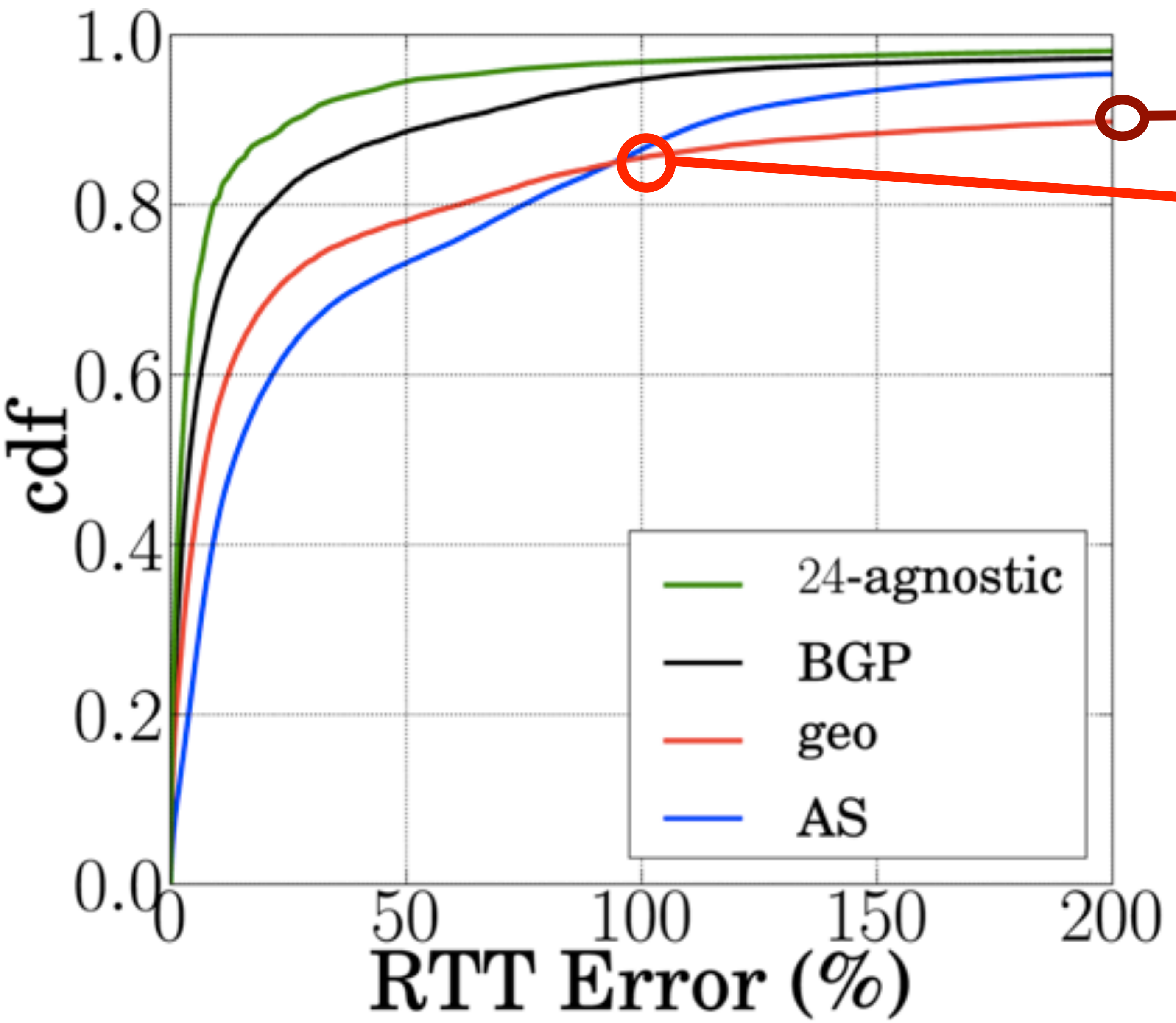
RTT Error



10% with more than 200% error

$$e_{ij} = \frac{|m_{ij} - \hat{m}_{ij}|}{m_{ij}}$$

RTT Error

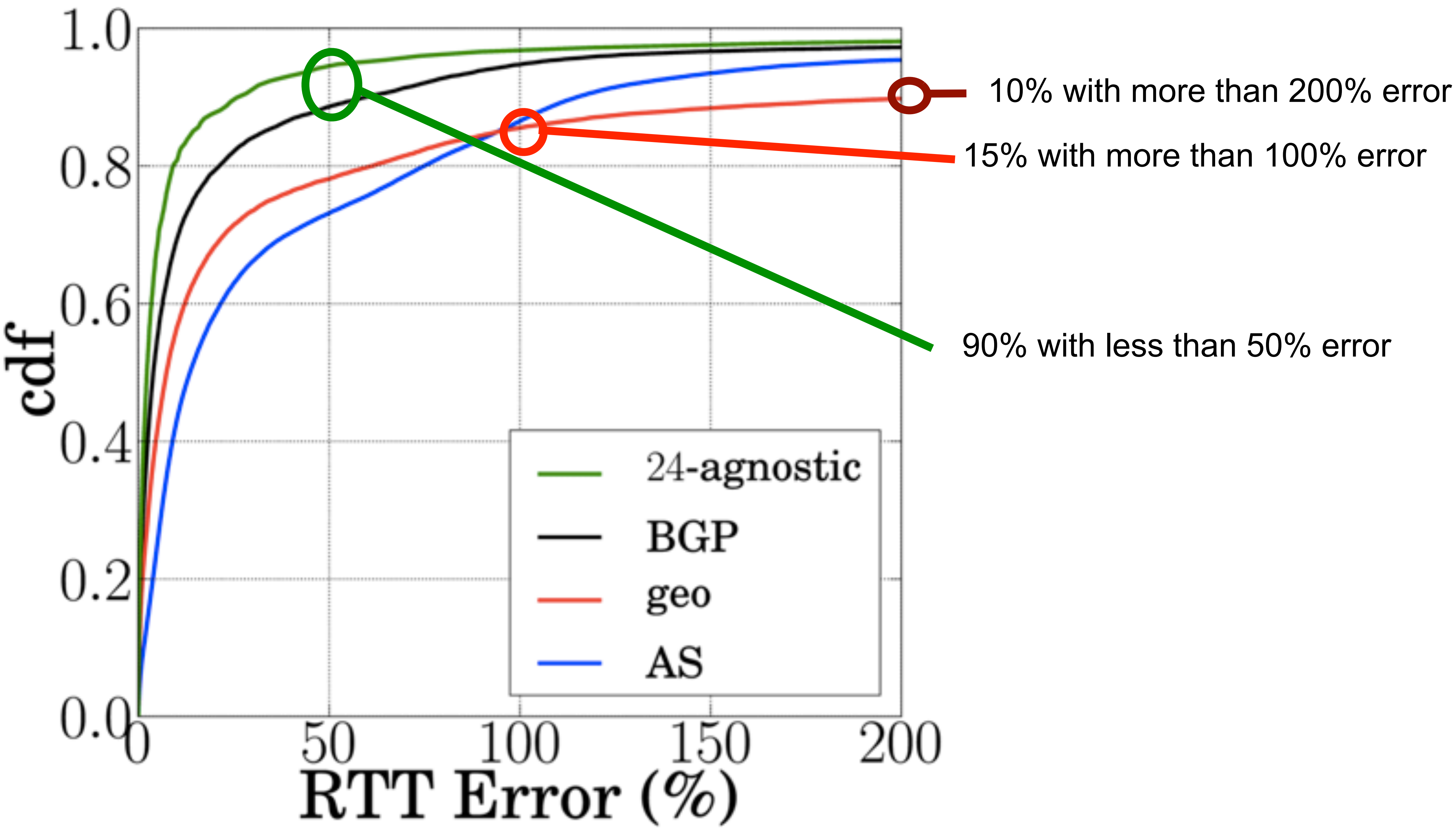


10% with more than 200% error

15% with more than 100% error

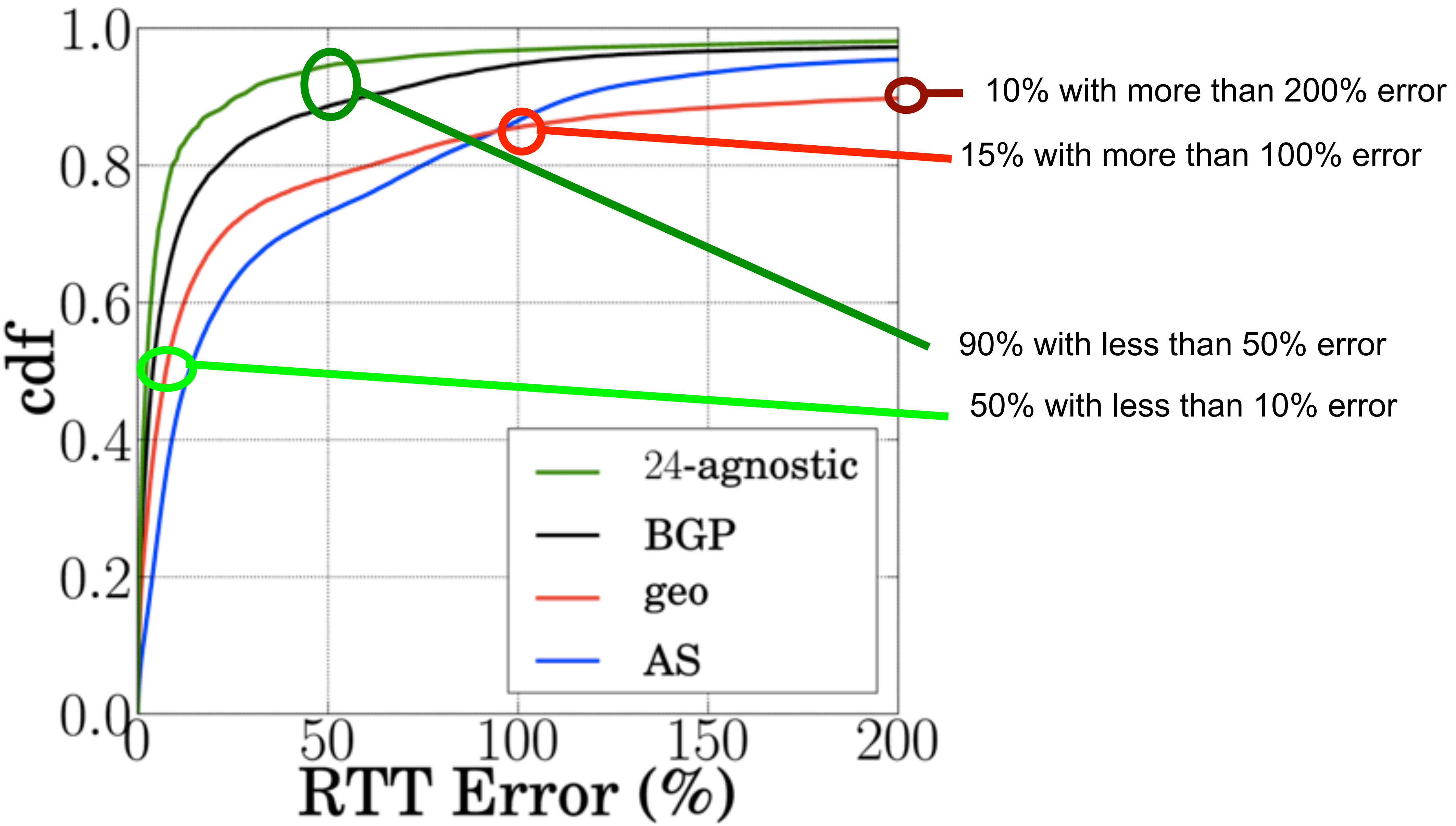
$$e_{ij} = \frac{|m_{ij} - \hat{m}_{ij}|}{m_{ij}}$$

RTT Error



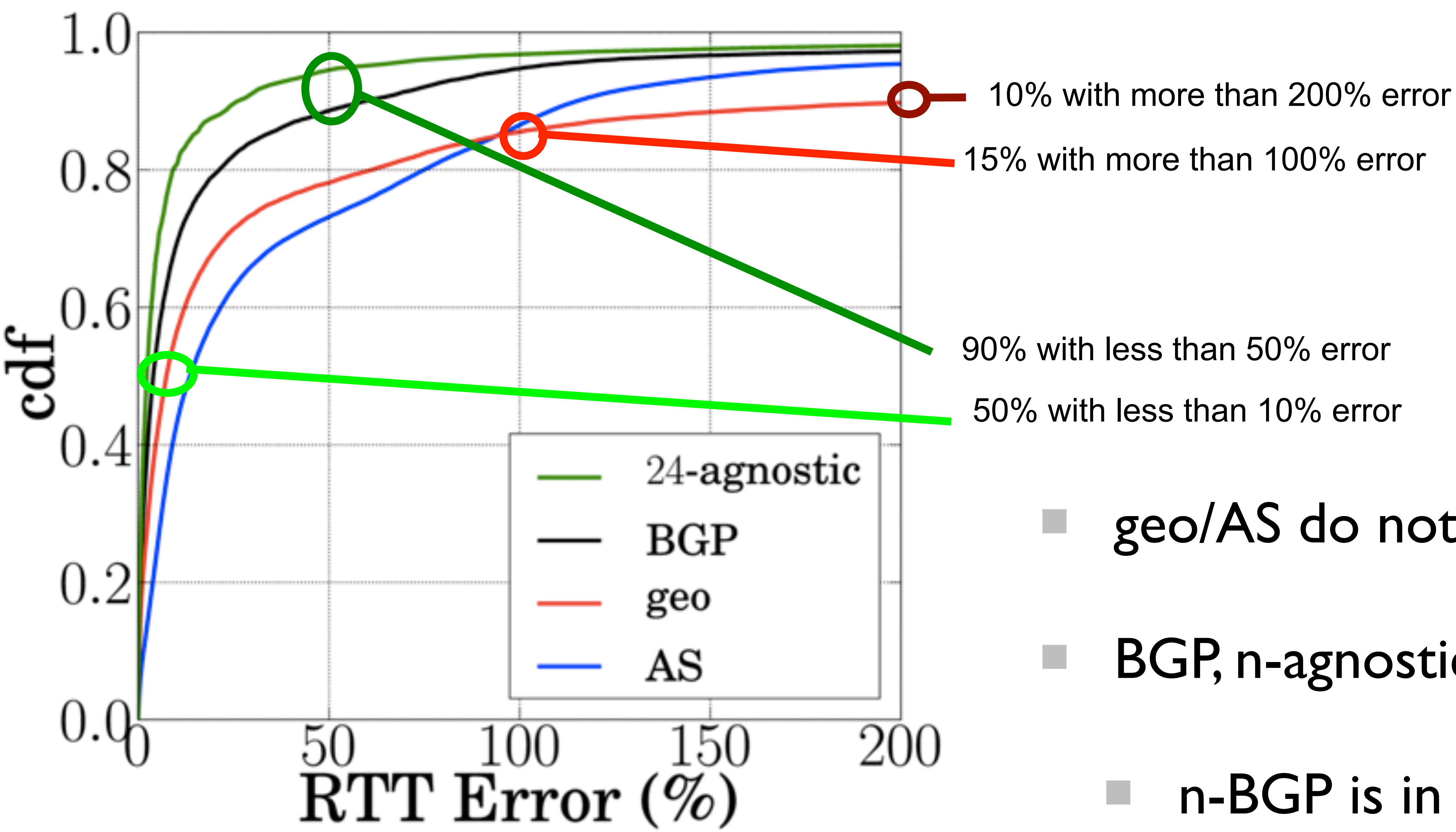
$$e_{ij} = \frac{|m_{ij} - \hat{m}_{ij}|}{m_{ij}}$$

RTT Error



$$e_{ij} = \frac{|m_{ij} - \hat{m}_{ij}|}{m_{ij}}$$

RTT Error



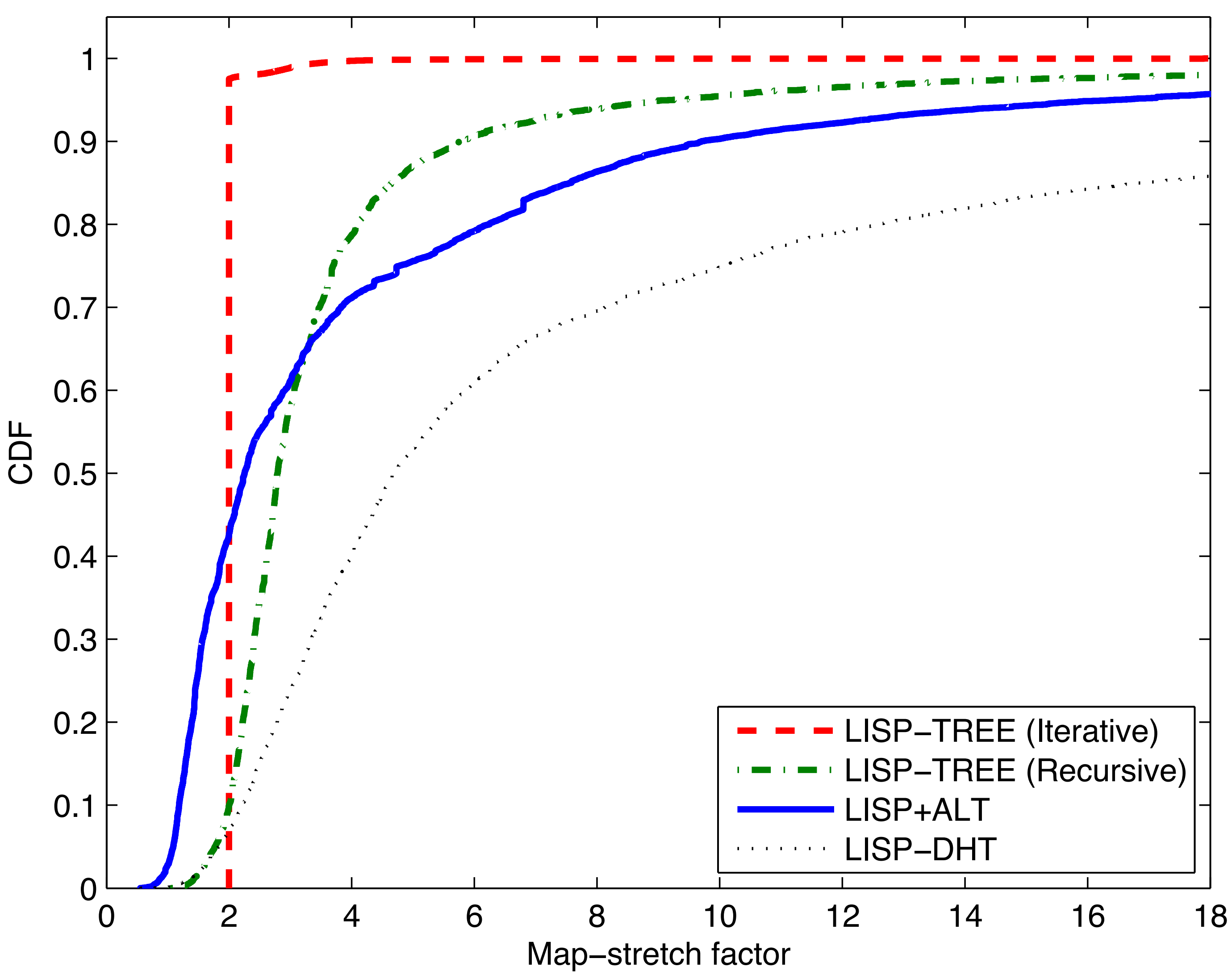
- geo/AS do not work
- BGP, n-agnostic work
  - n-BGP is in between

$$e_{ij} = \frac{|m_{ij} - \hat{m}_{ij}|}{m_{ij}}$$

Evaluation Setup

- One day full NetFlow trace collected on March 23, 2009 at UCL
  - 1,200 million packets
  - 69Mbps on average
- Three level hierarchy (delay via iPlane)
  - level 1 - root
  - level 2 - 256 /8 prefixes
  - level 3 - 112, 233 prefixes in 14,340 PoP
    - ETR are collocated with level 3 LISP-Tree servers
    - Resolver collocated at the ITR site
- ITR: 3 minutes timeout

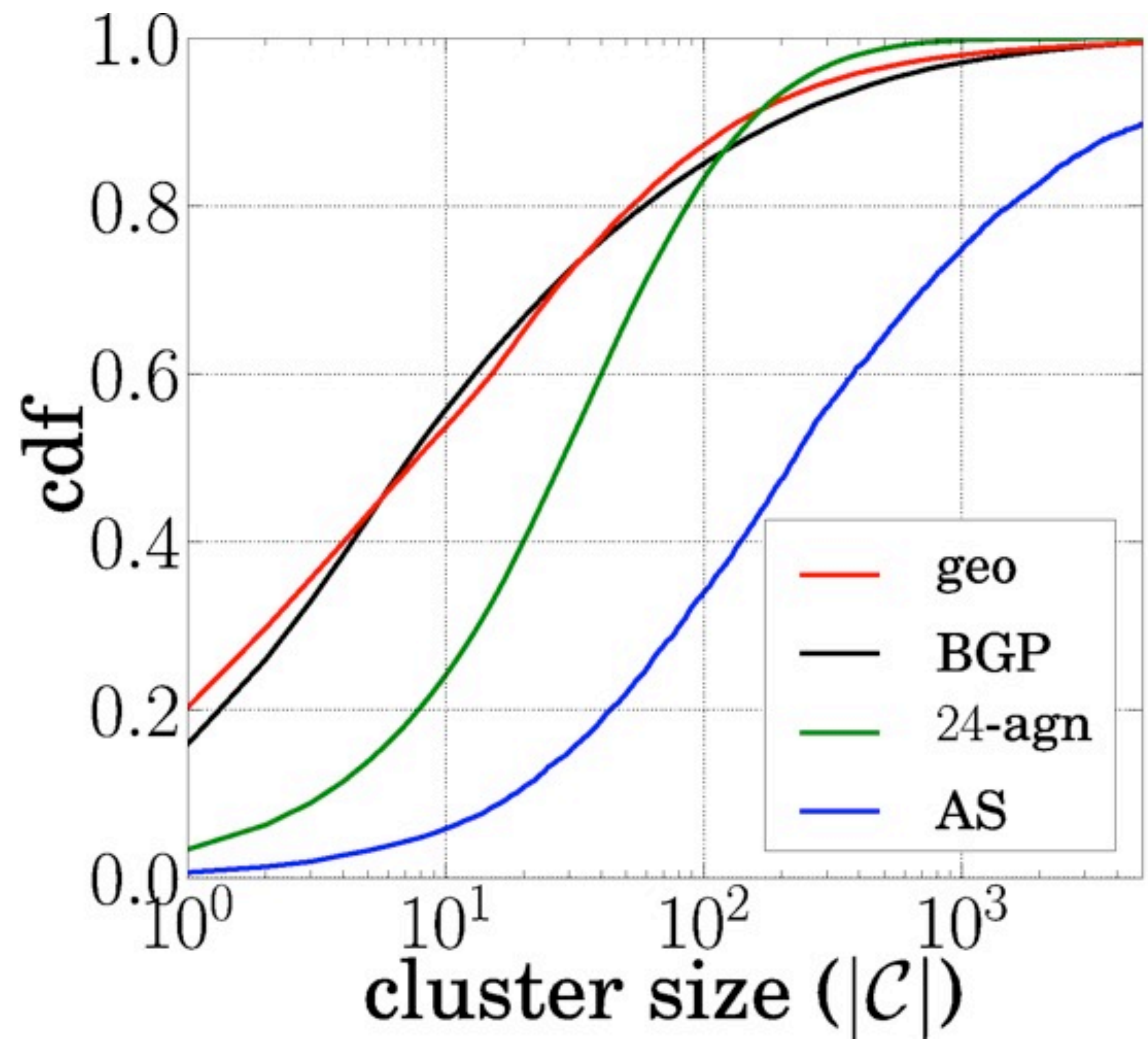
Caching makes mapping retrieval fast



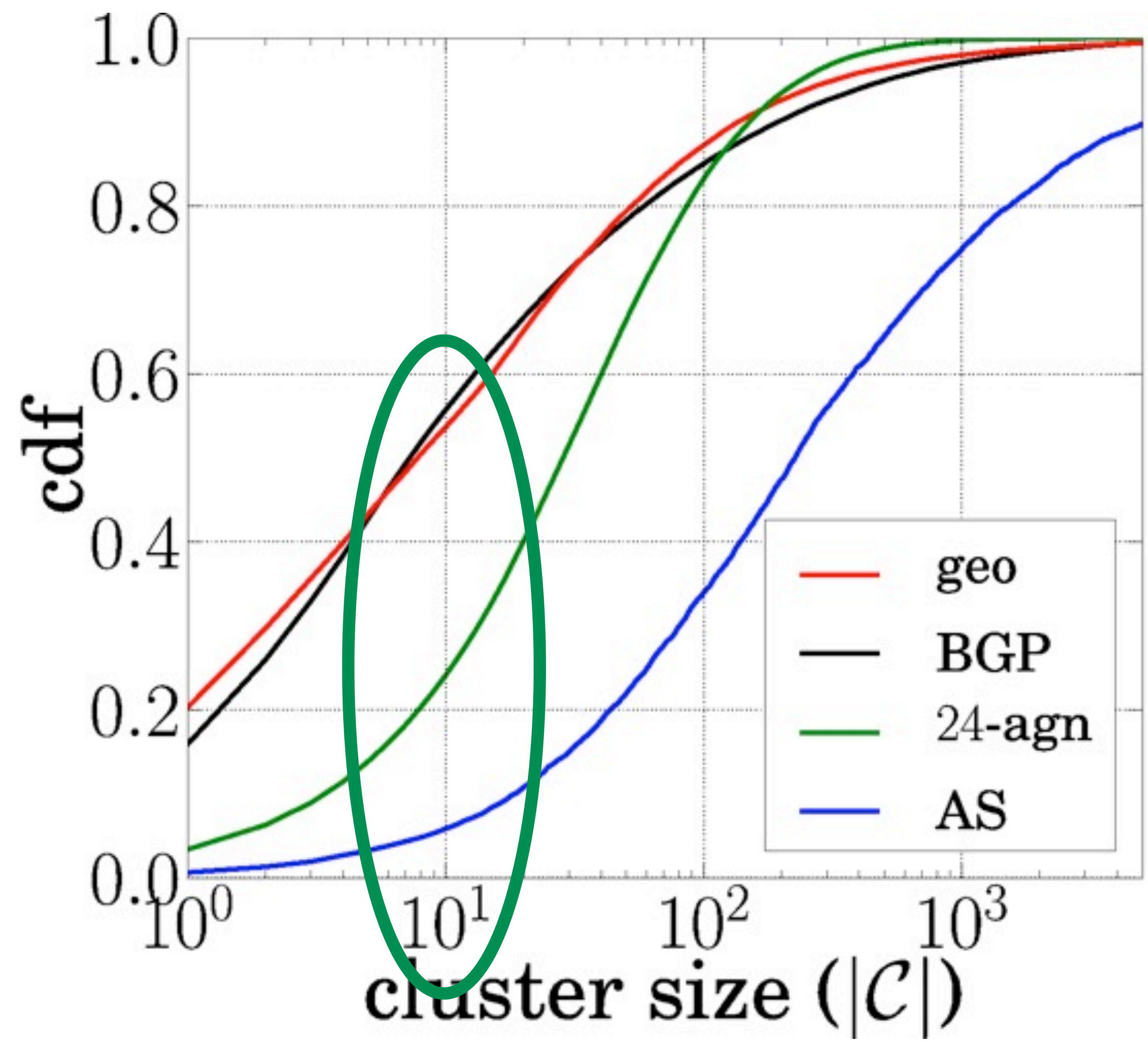
$$\text{Map-stretch} = \frac{\text{total\_time}}{\text{RTT}_{\text{ITR,ETR}}}$$

- LISP+ALT is slow because Map-Requests must always traverse the ALT topology
- LISP-Tree in iterative mode is fast because the tree is not browsed for every Map-Request
  - the resolver caches information about the tree
  - the tree can remain stable for long periods

Clustering effectively reduces the number of measurements

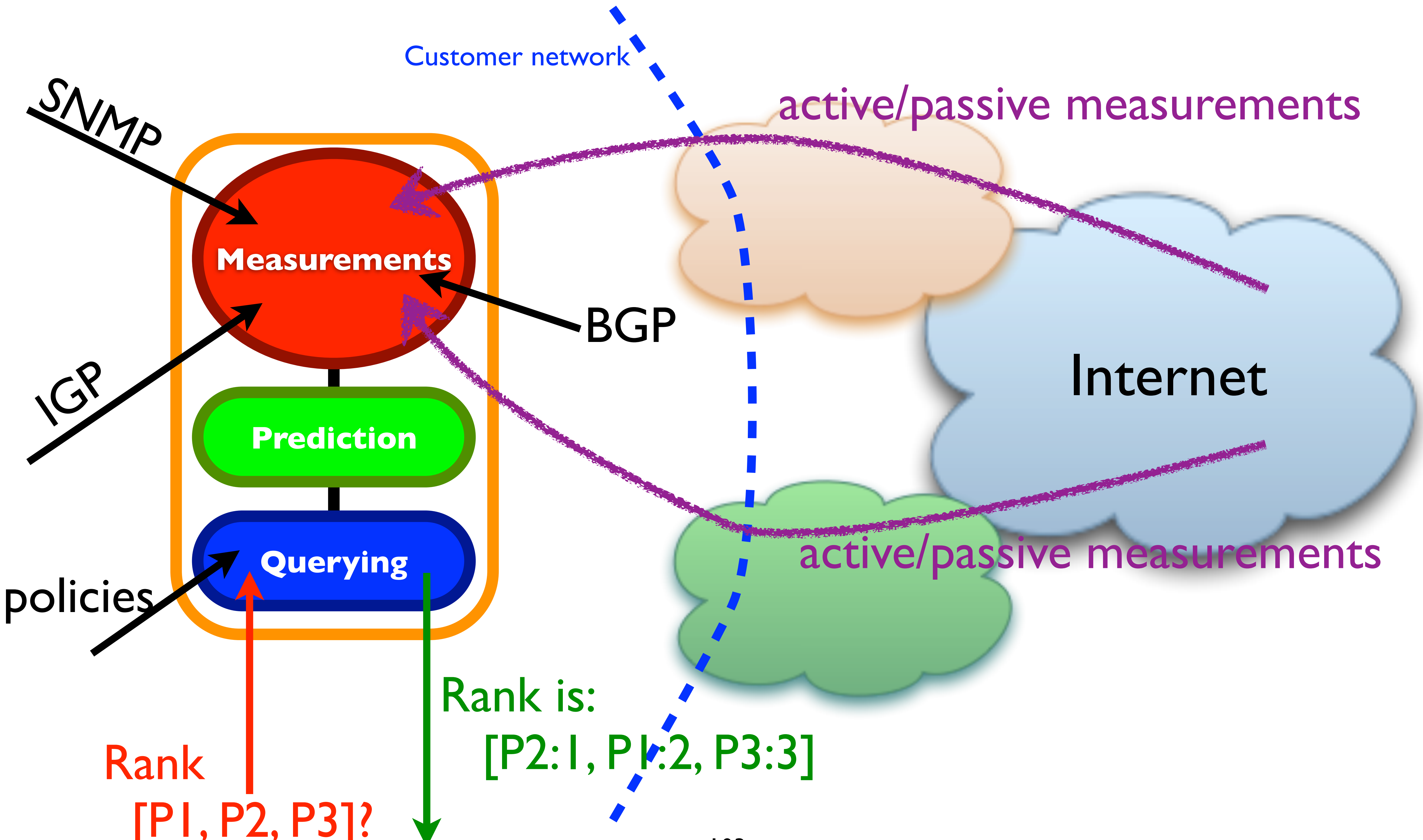


Clustering effectively reduces the number of measurements

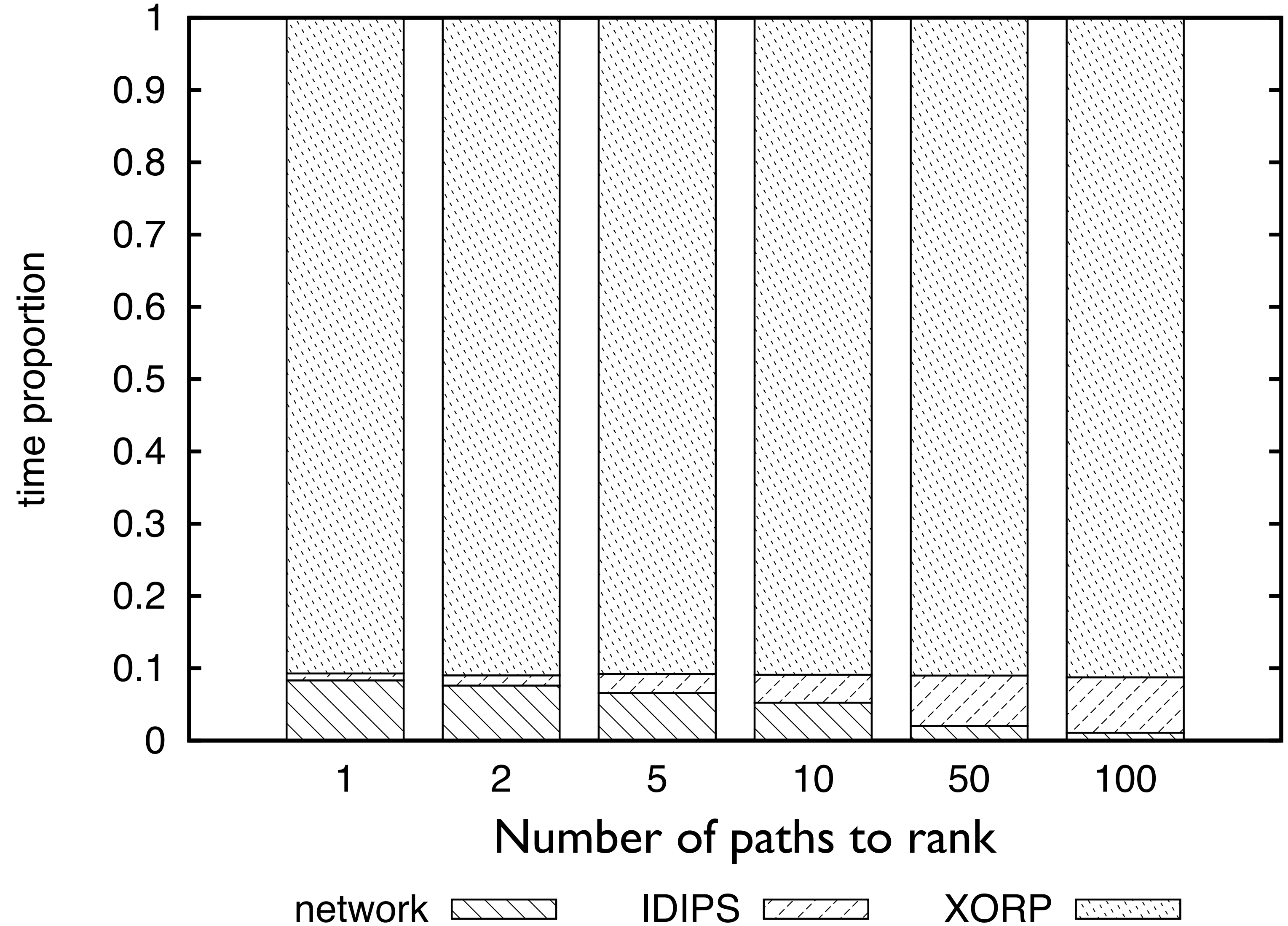


- At least 45% of the clusters cover more than 10 nodes

Inside IDIPS



IDIPS is lightweight, XORP is not

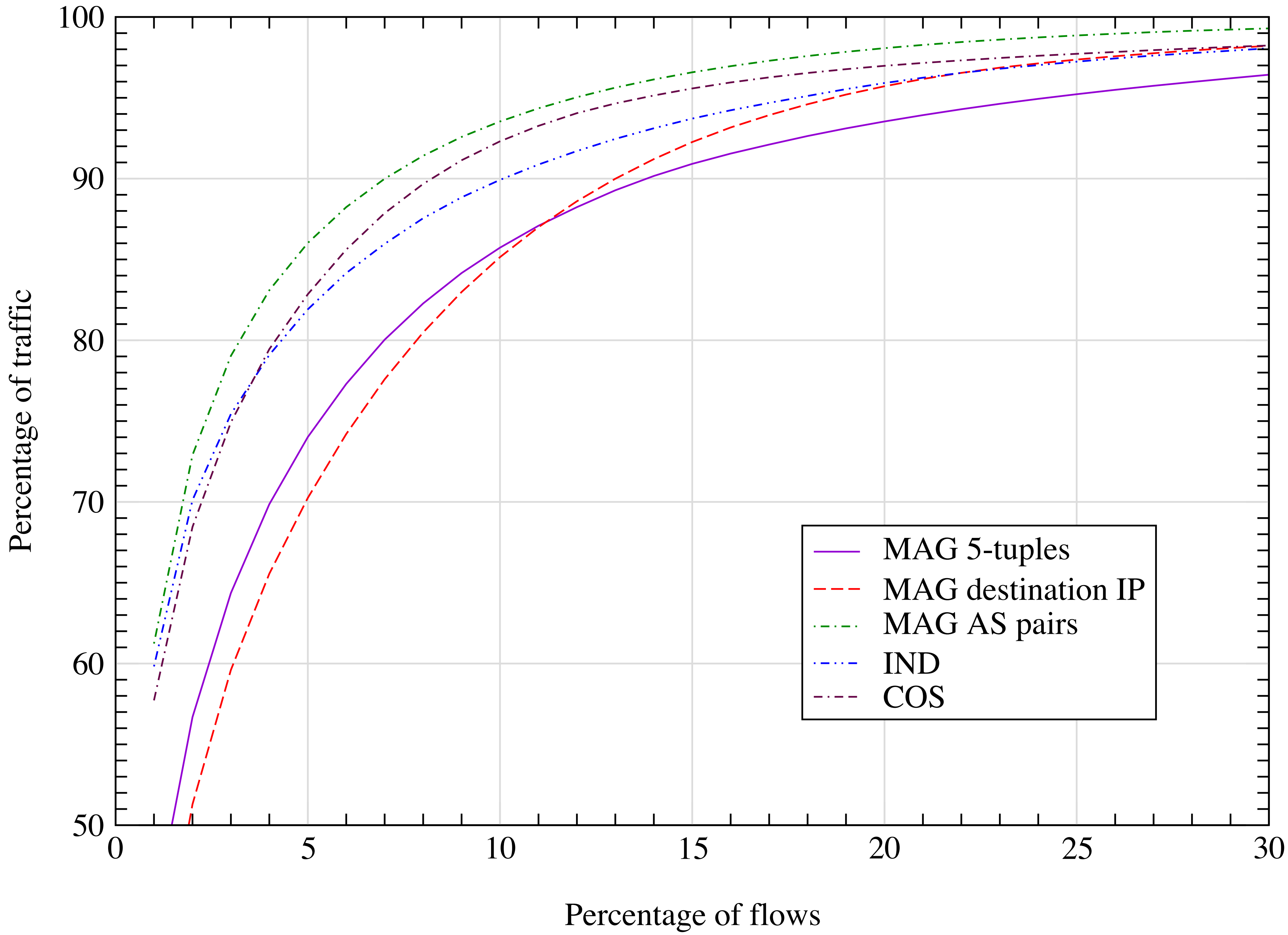


- Up to 90% of the time is lost in XORP, not in IDIPS cost computation
  - the finder is the bottleneck

Measurement reduction

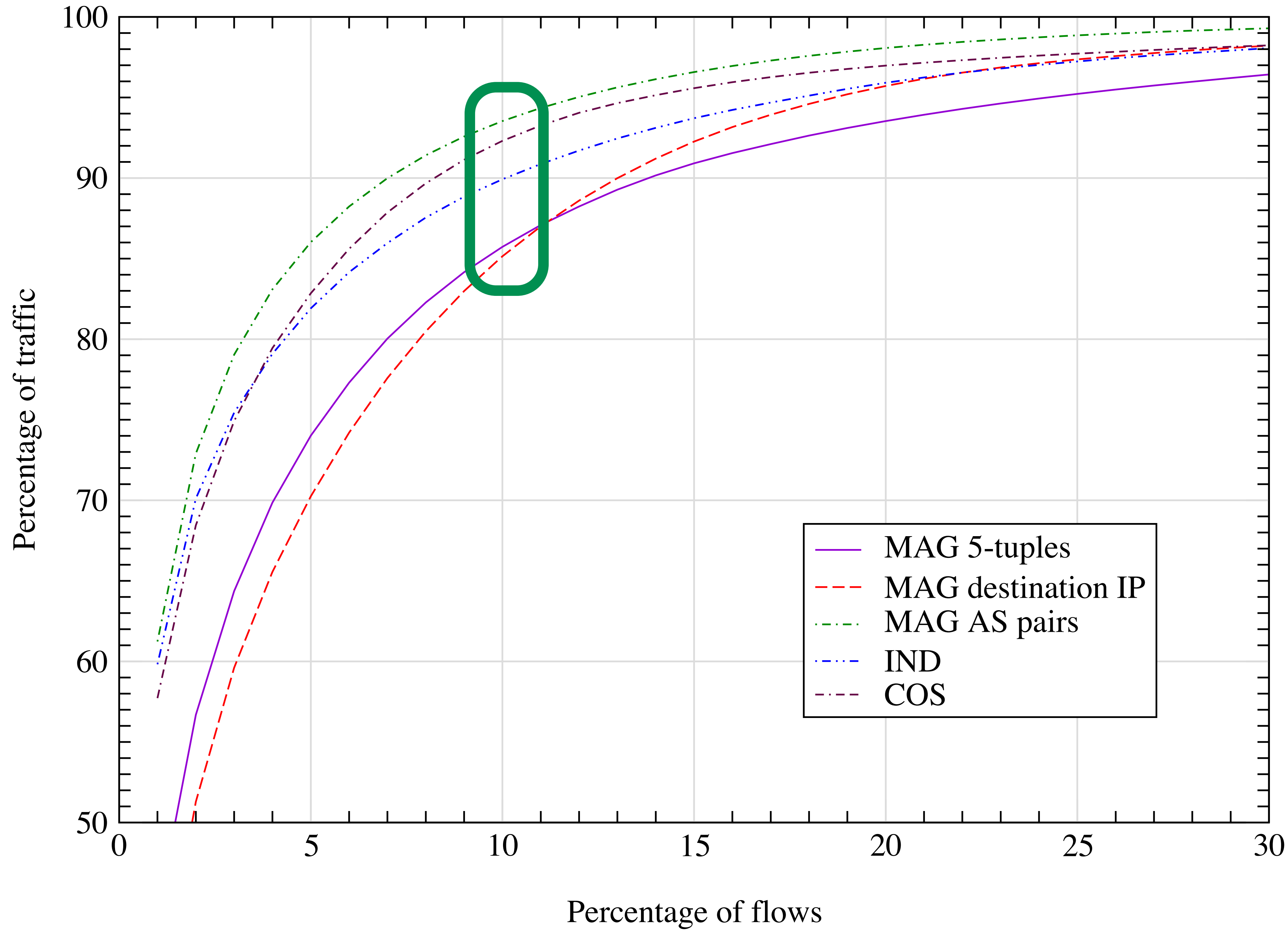
- focus on top talkers

Focus on the top talkers



[EV03] Estan, C. and Varghese, G., *New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice*, ACM TCSystems 21(3) 2003.

Focus on the top talkers



[EV03] Estan, C. and Varghese, G., *New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice*, ACM TCSystems 21(3) 2003.

Performance based Interdomain Incoming Traffic Engineering

Performance based Interdomain Incoming Traffic Engineering

- traffic entering the network and that is originated by another network

Performance based Interdomain Incoming Traffic Engineering

- capacity of controlling the way traffic is entering, transiting or leaving a network

Performance based Interdomain Incoming Traffic Engineering

- *"capabilities of a machine or product, esp. when observed under particular conditions"* [NOAD]

[NOAD] New Oxford American Dictionary

Performance based Interdomain Incoming Traffic Engineering

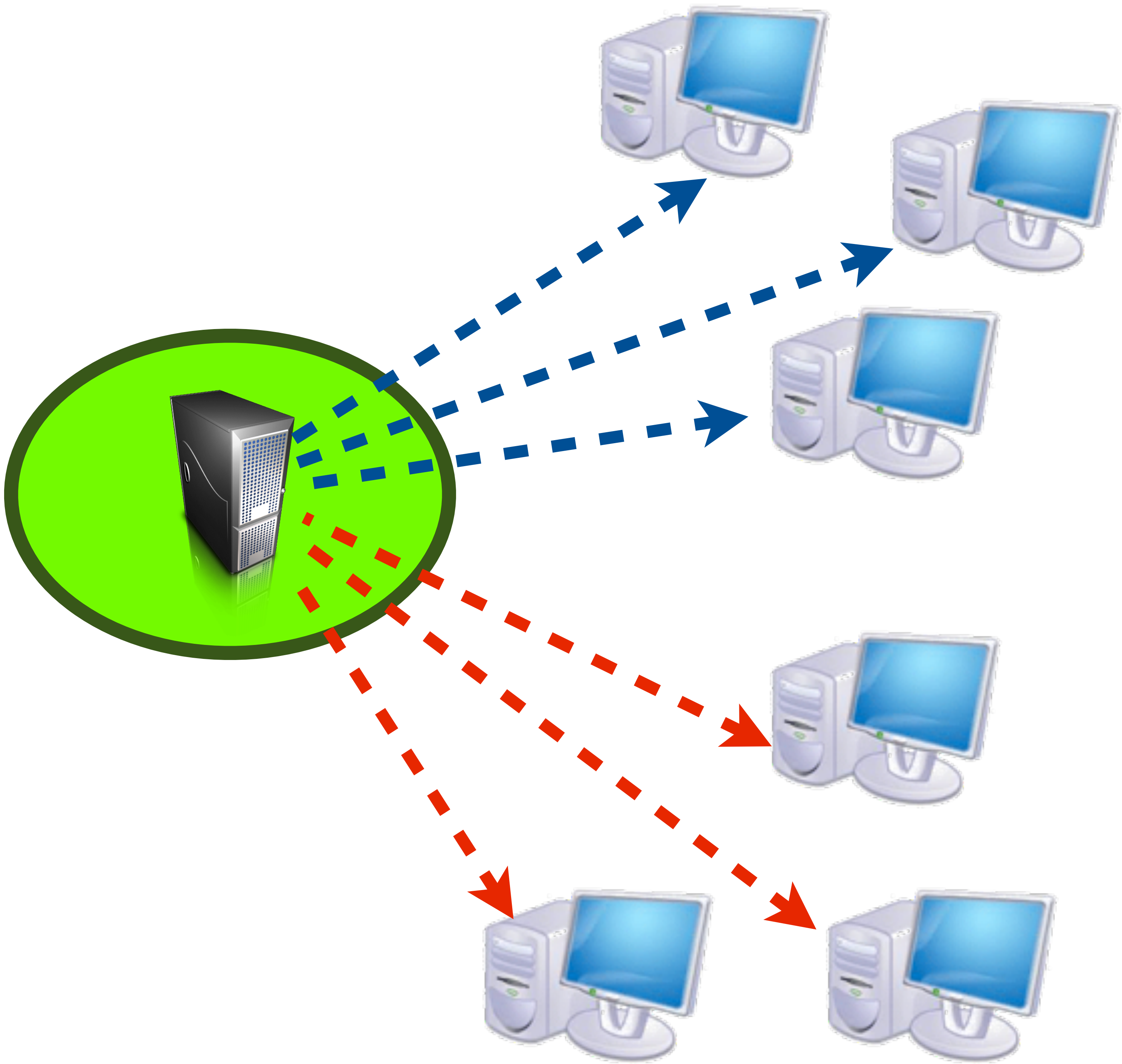
- A network implements performance based inter-domain incoming traffic engineering if it manages to make its inter-domain incoming traffic entering via the links that optimize its efficiency (e.g., minimize delay)

[NOAD] New Oxford American Dictionary

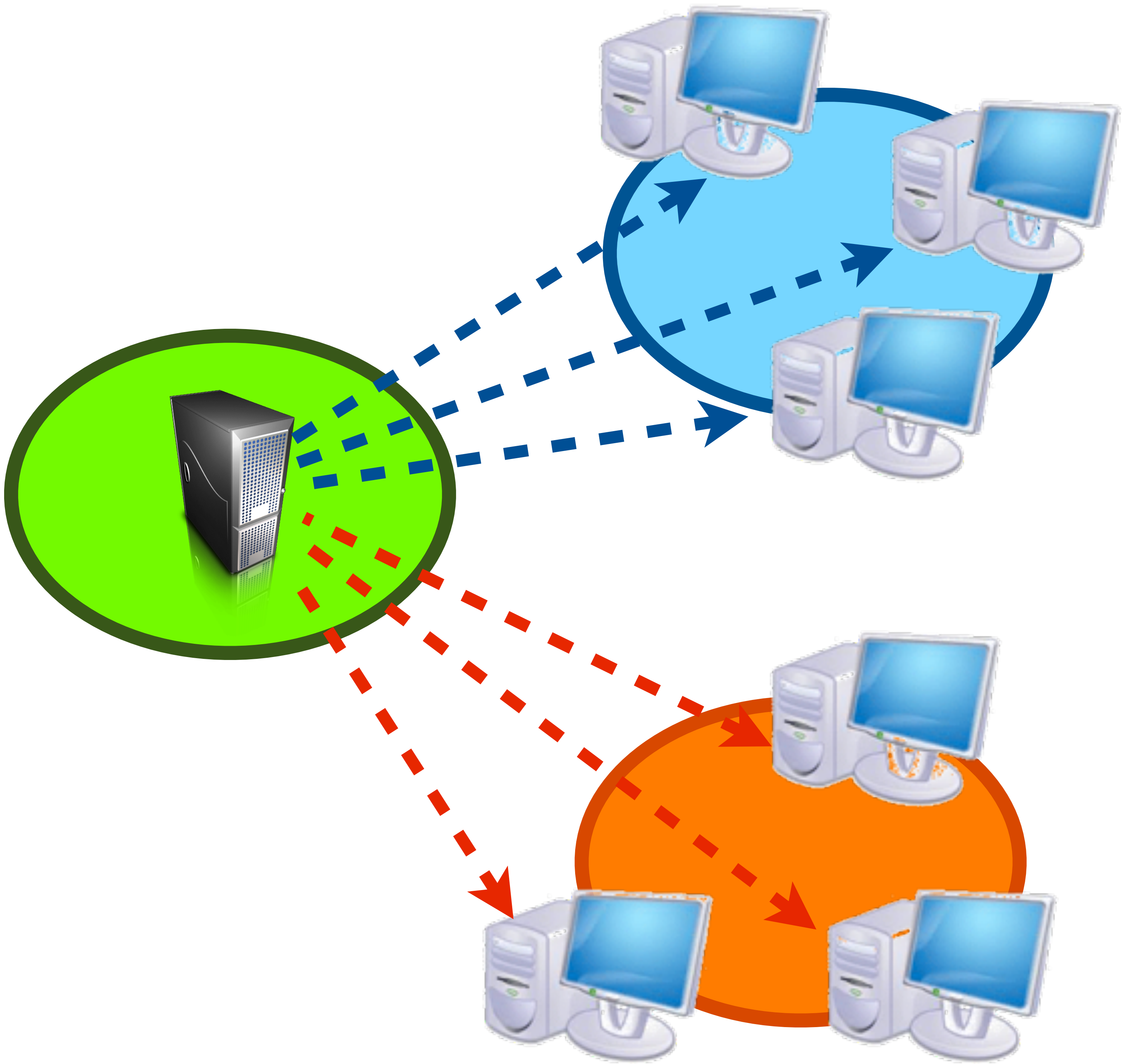
Measurement reduction

- group destinations into clusters

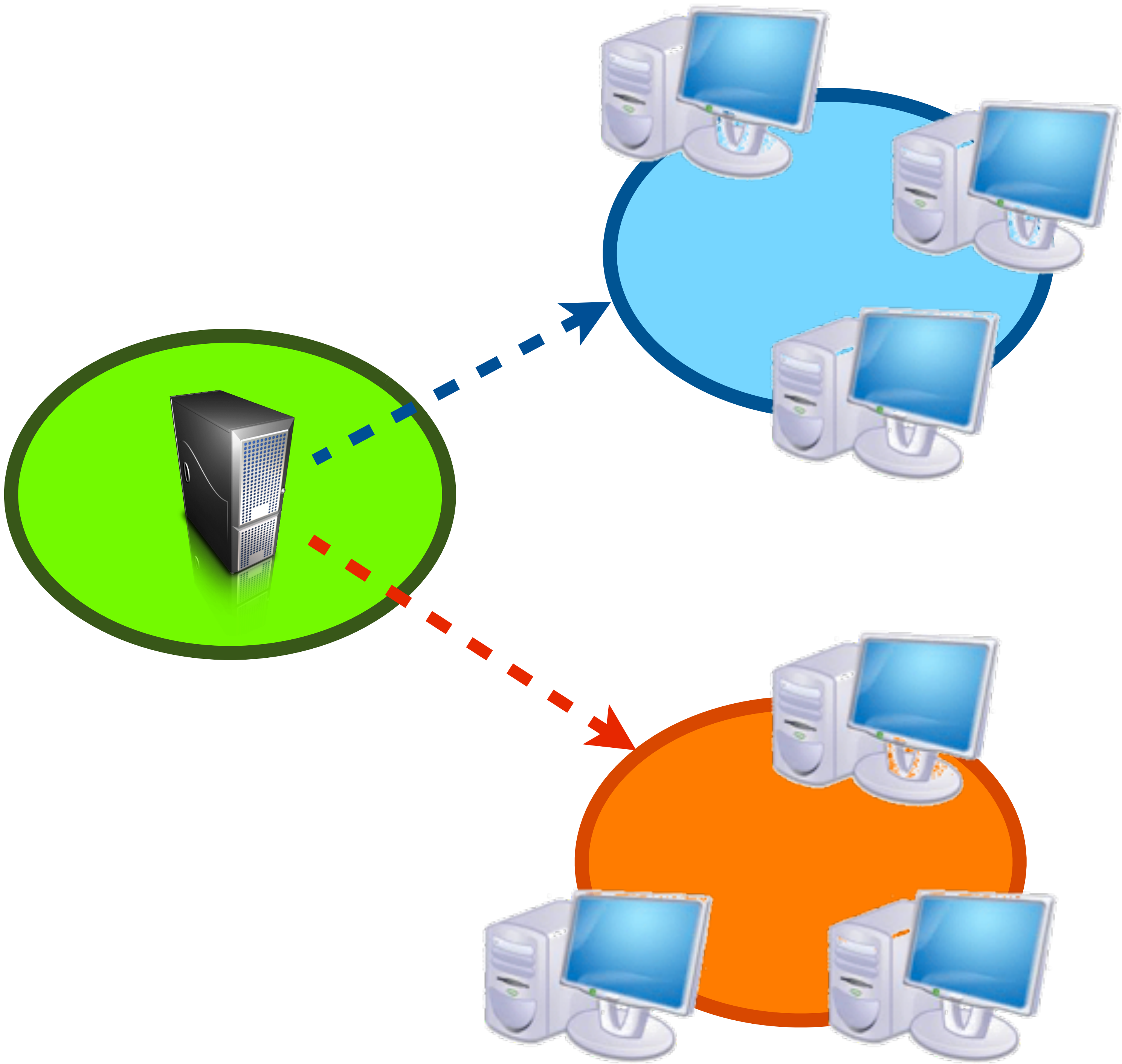
Clustering



Clustering



Clustering



Clustering

- geographic clustering
  - group nodes by city
- AS clustering [KW01]
  - group nodes by autonomous systems
- *n*-agnostic clustering [SPPVS08]
  - group nodes by /n prefixes
- BGP clustering [KW00]
  - group nodes by longest-match BGP prefix
- *n*-BGP clustering [SDB09]
  - group nodes by the more specific prefix of *n*-agnositc clustering and BGP clustering