Internet of the future

Master ESTEL - March 11th 2015

Contact information

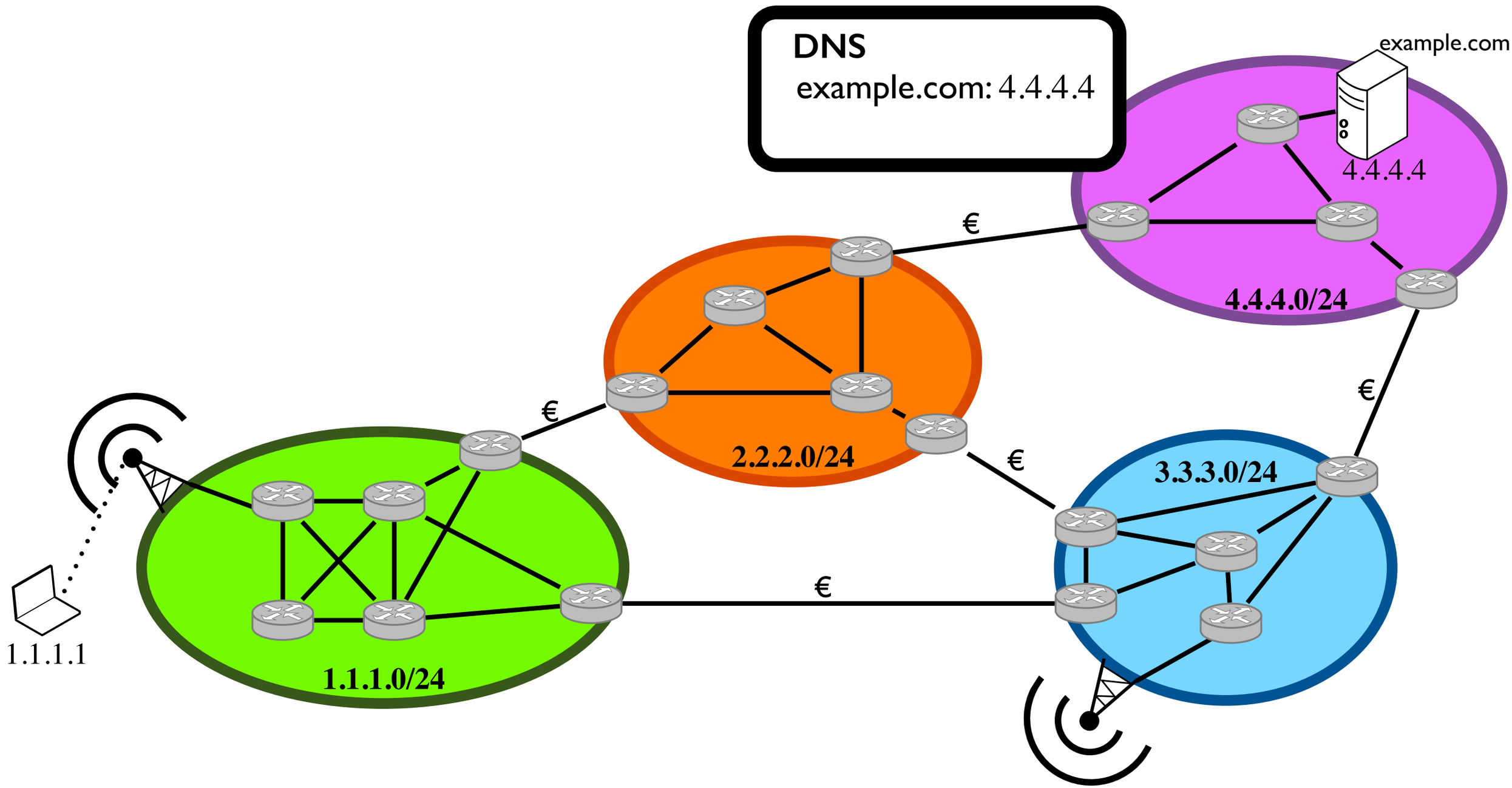
- Damien Saucez
  - Email: damien.saucez@inria.fr
  - Mobile: +32 497 19.34.83
  - Phone: +33 4 89.73.24.18

Internet today

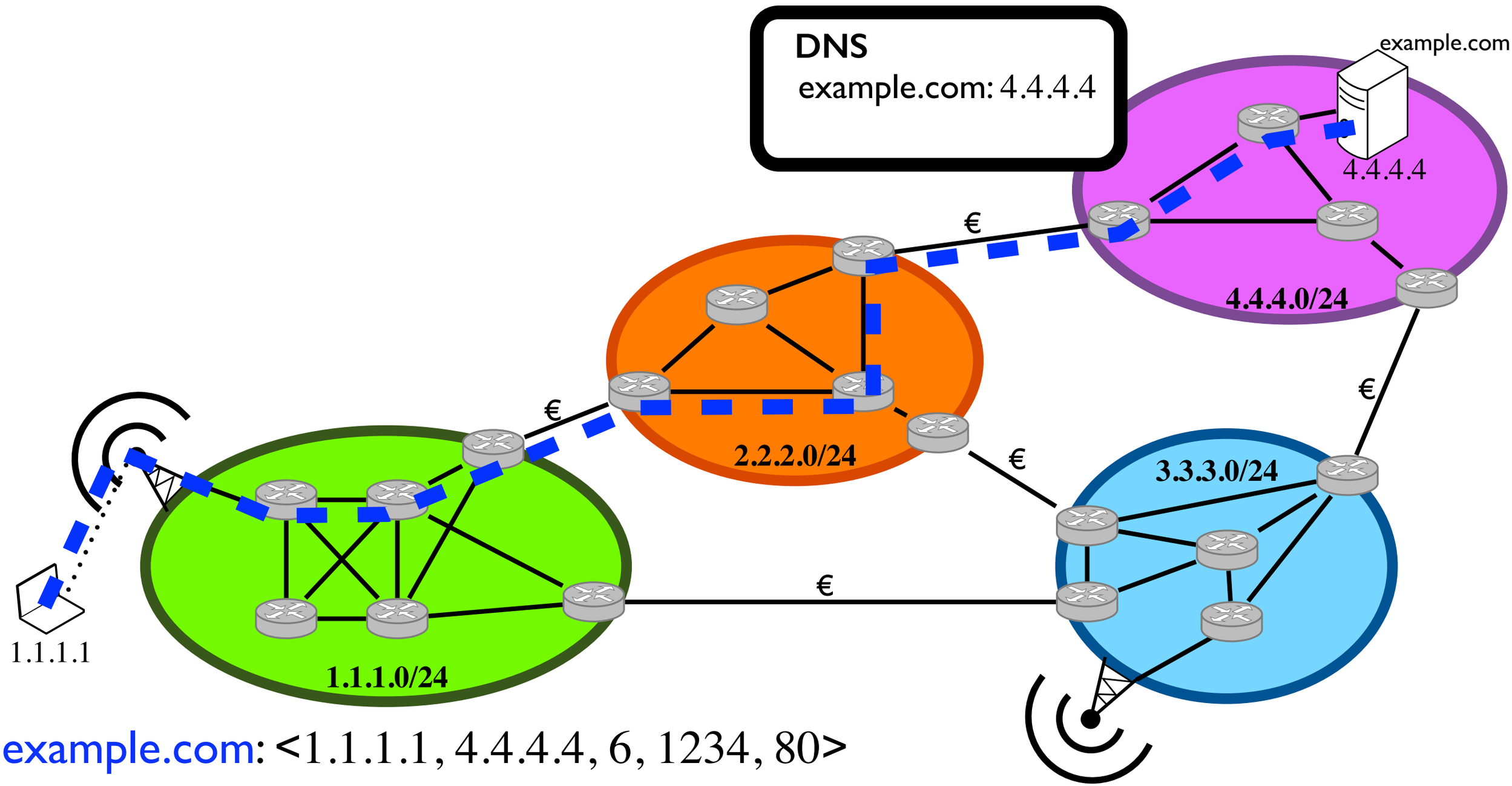
Internet Protocol (IP)

- 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

The curse of IP

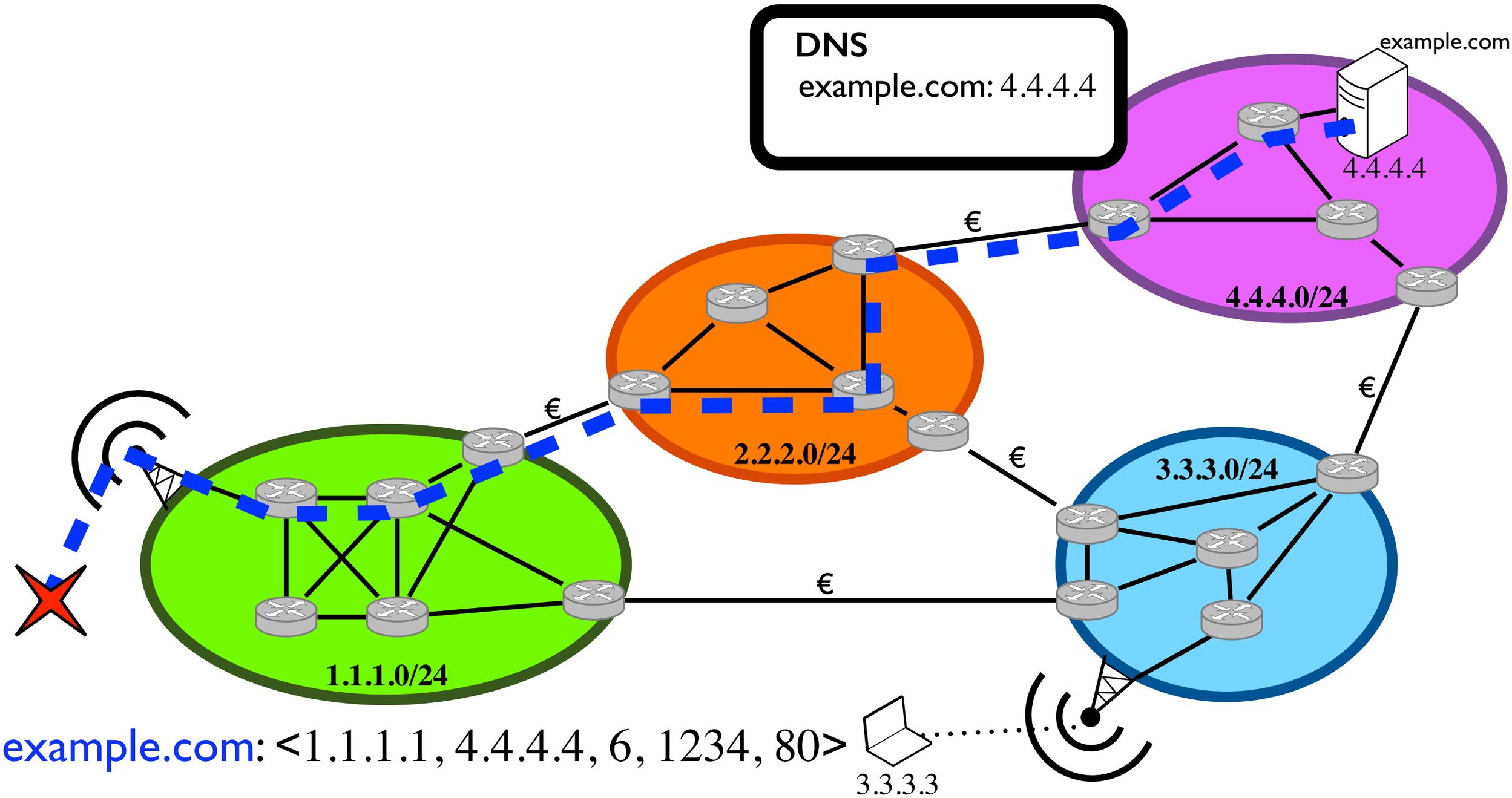


The curse of IP



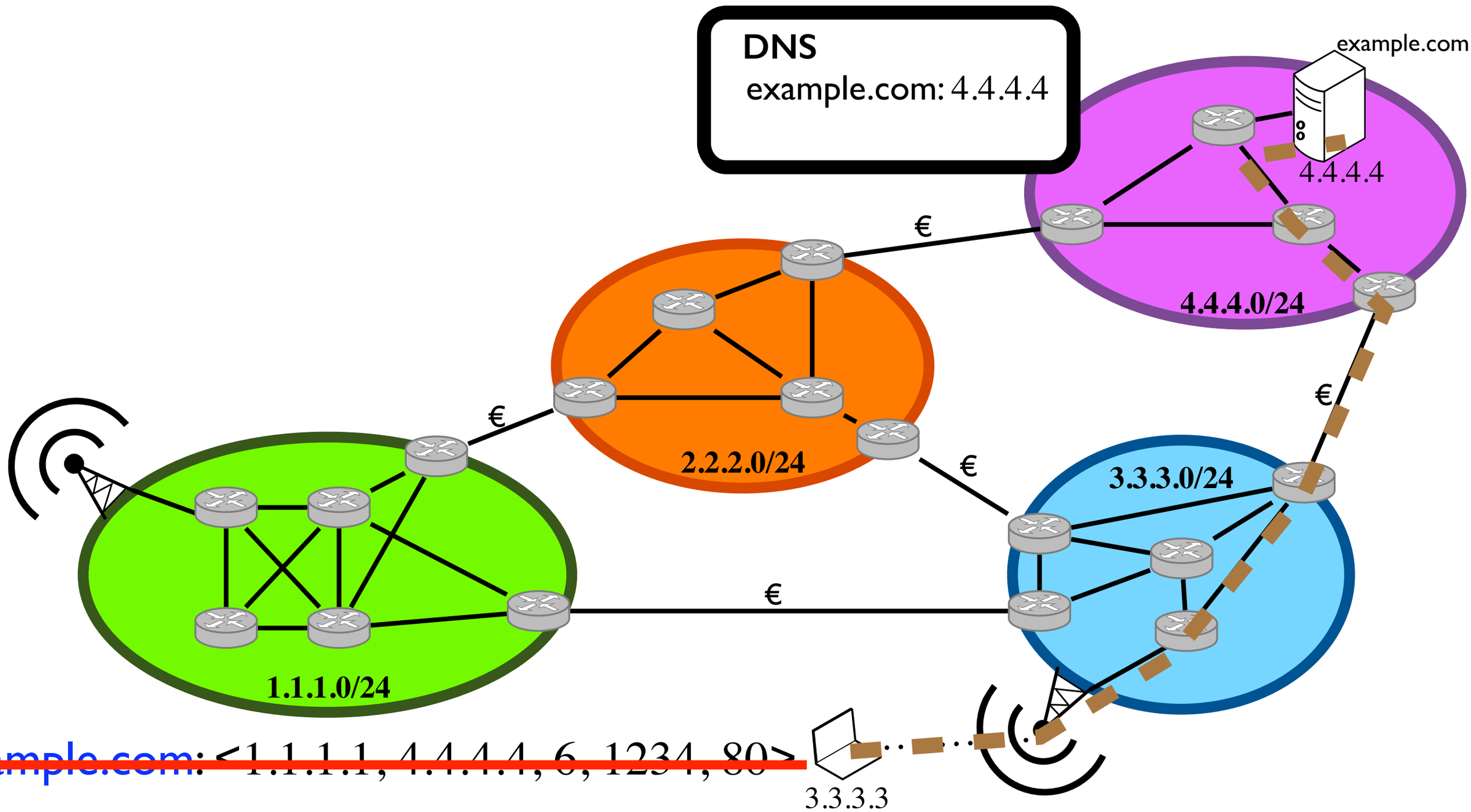
example.com: <1.1.1.1, 4.4.4.4, 6, 1234, 80>

The curse of IP



example.com: <1.1.1.1, 4.4.4.4, 6, 1234, 80>

The curse of IP

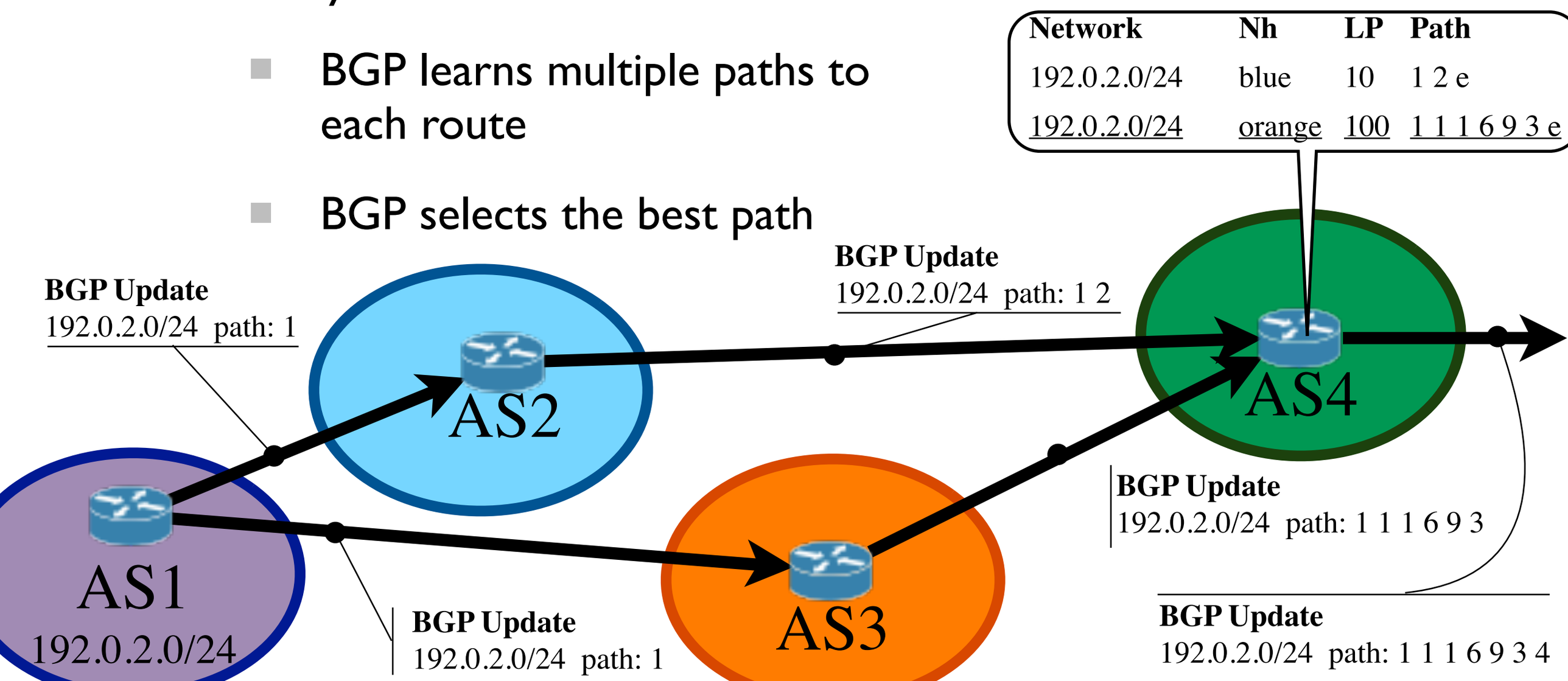


~~example.com: <1.1.1.1, 4.4.4.4, 6, 1234, 80>~~

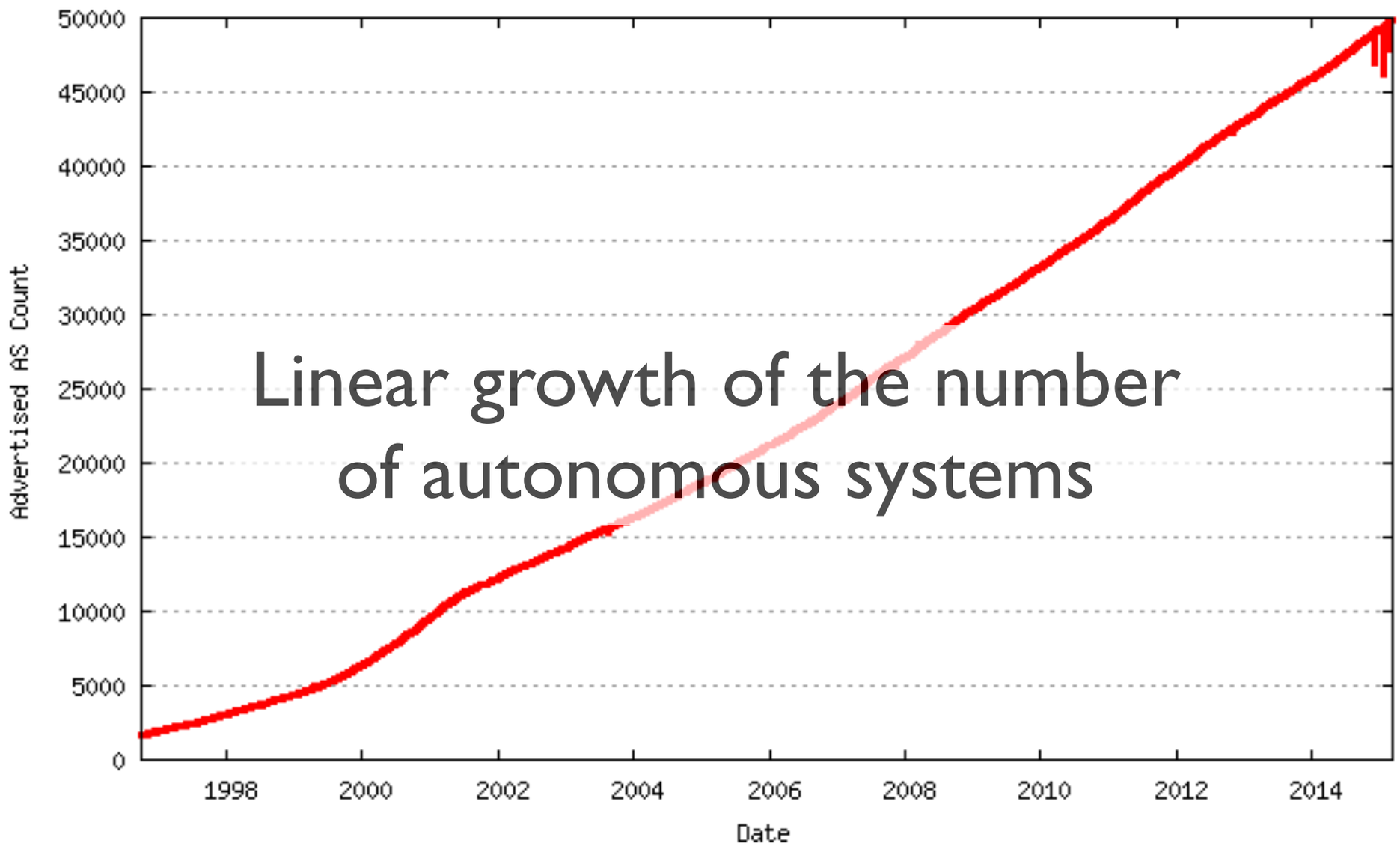
example.com: <3.3.3.3, 4.4.4.4, 6, 5678, 80>

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



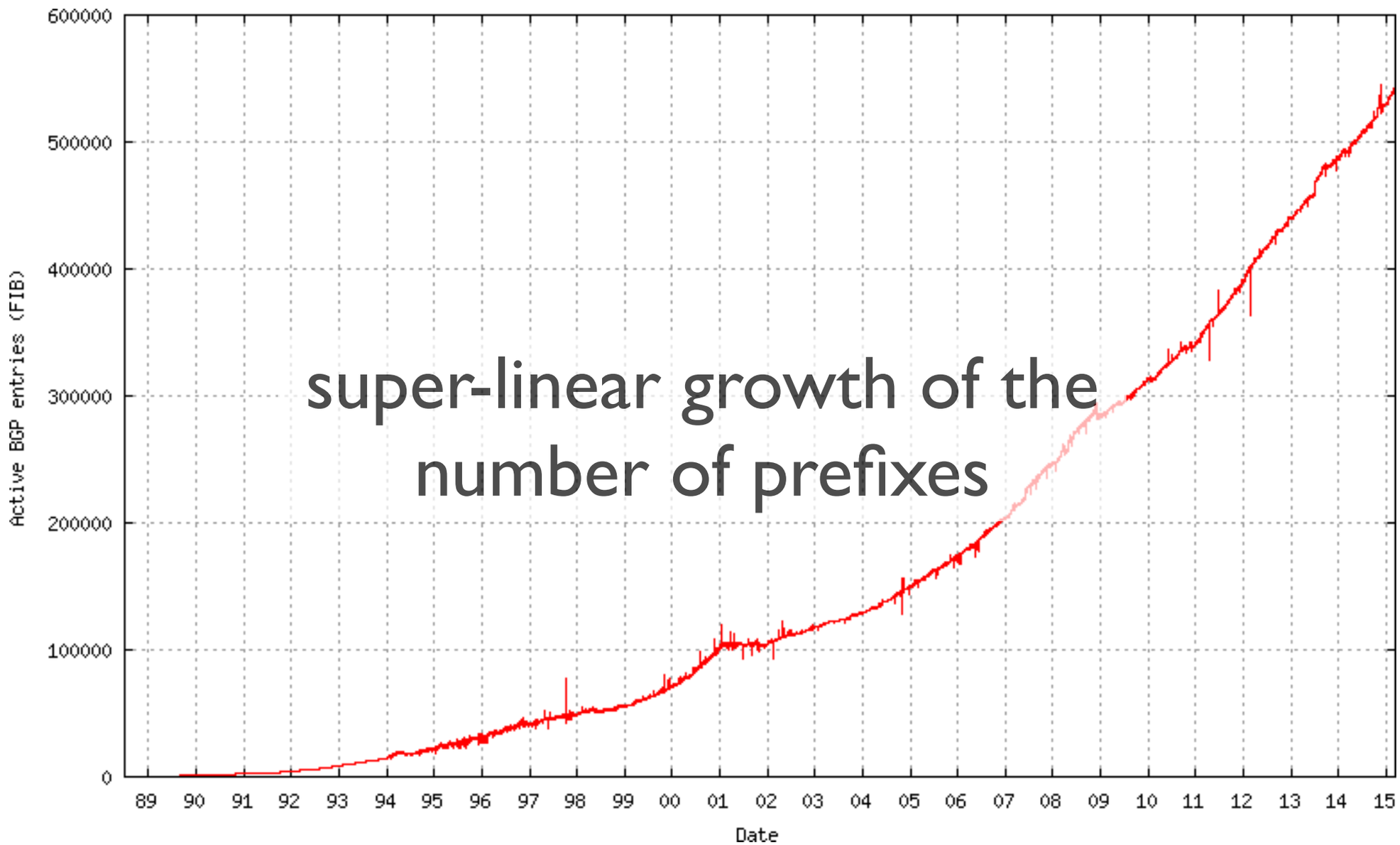
More networks



Linear growth of the number of autonomous systems

[http://bgp.potaroo.net, 03/10/2015]

More prefixes



super-linear growth of the number of prefixes

[http://bgp.potaroo.net, 03/10/2015]

What do prefixes look like?

- 49,806 ASes
- 541,645 prefixes
  - 53% of /24
- 29,945 ASes originate more than one prefix
- after aggregation: 299,462 prefixes

➡ prefixes de-aggregated 1.8 times on average

[http://bgp.potaroo.net, 03/06/2015]

Why so many (small) prefixes?

- Allocation of IP prefixes to sites
  - Initial solution chosen by IANA
    - First come, first served for all qualifying sites

```
130.100.0.0/16 ripencc  adv
130.101.0.0/16 arin     adv
130.102.0.0/16 apnic    adv
130.103.0.0/16 arin     unadv
130.104.0.0/16 ripencc  adv
```

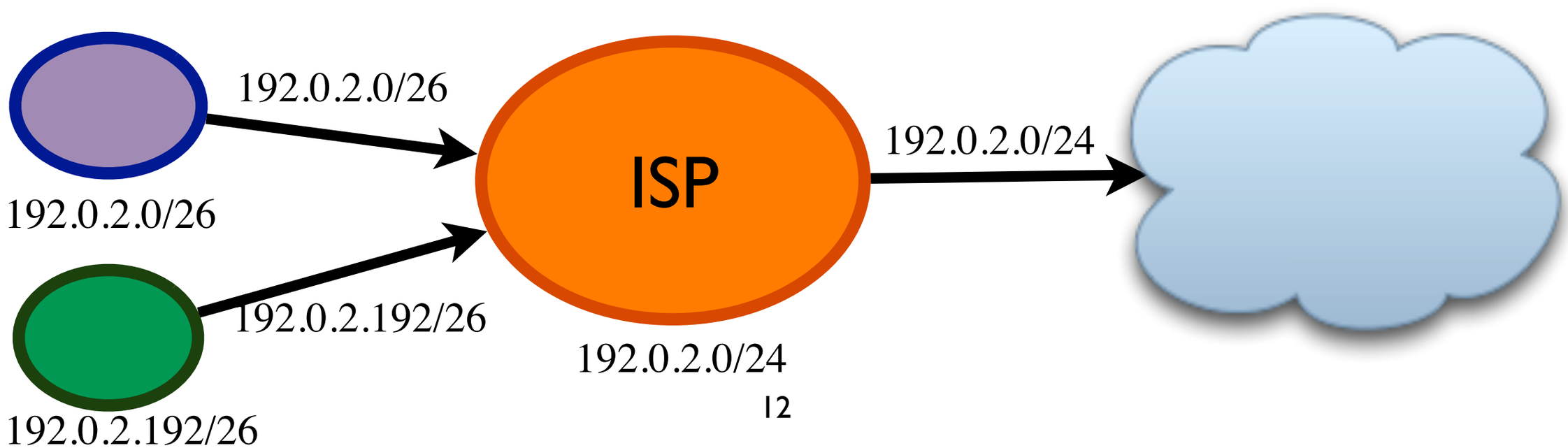
    - Few constraints on which sites qualify for an IP prefix, owned forever
    - Classful network design (/8, /16, /24)
  - Hard to aggregate

Why so many (small) prefixes?

- Current IP prefixes allocation with CIDR
- **Provider Independent** (PI) prefixes
  - Given by RIRs to qualifying sites (i.e., ISPs paying their membership dues to the RIR)
  - Owned by the site forever and can be globally announced

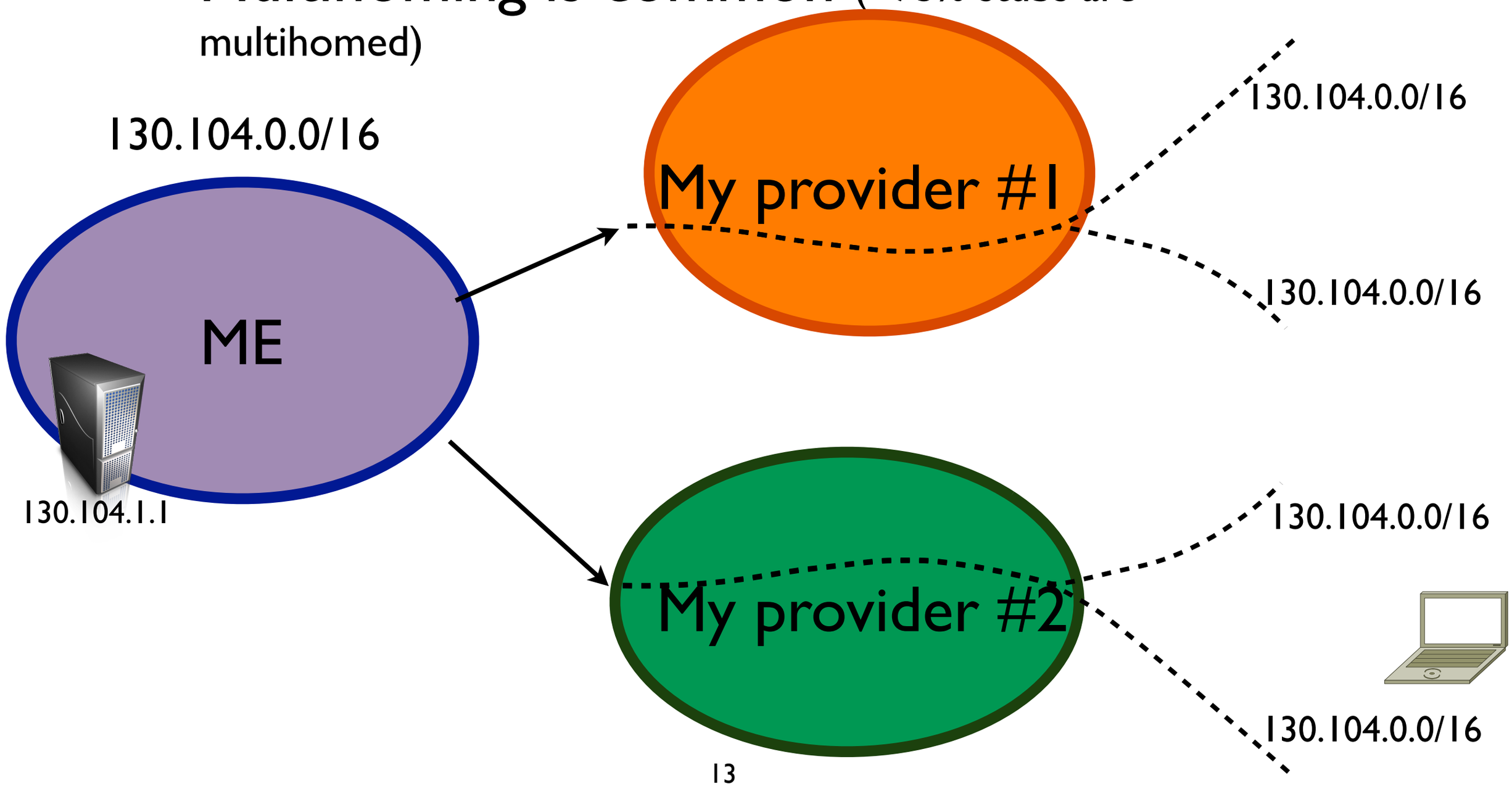
Why so many (small) prefixes?

- **Provider Aggregatable** (PA) prefixes
  - Given by ISPs from their own address block to customers
  - Customers return their prefix if they change ISP
- ... but provider lock-in, renumbering burden...



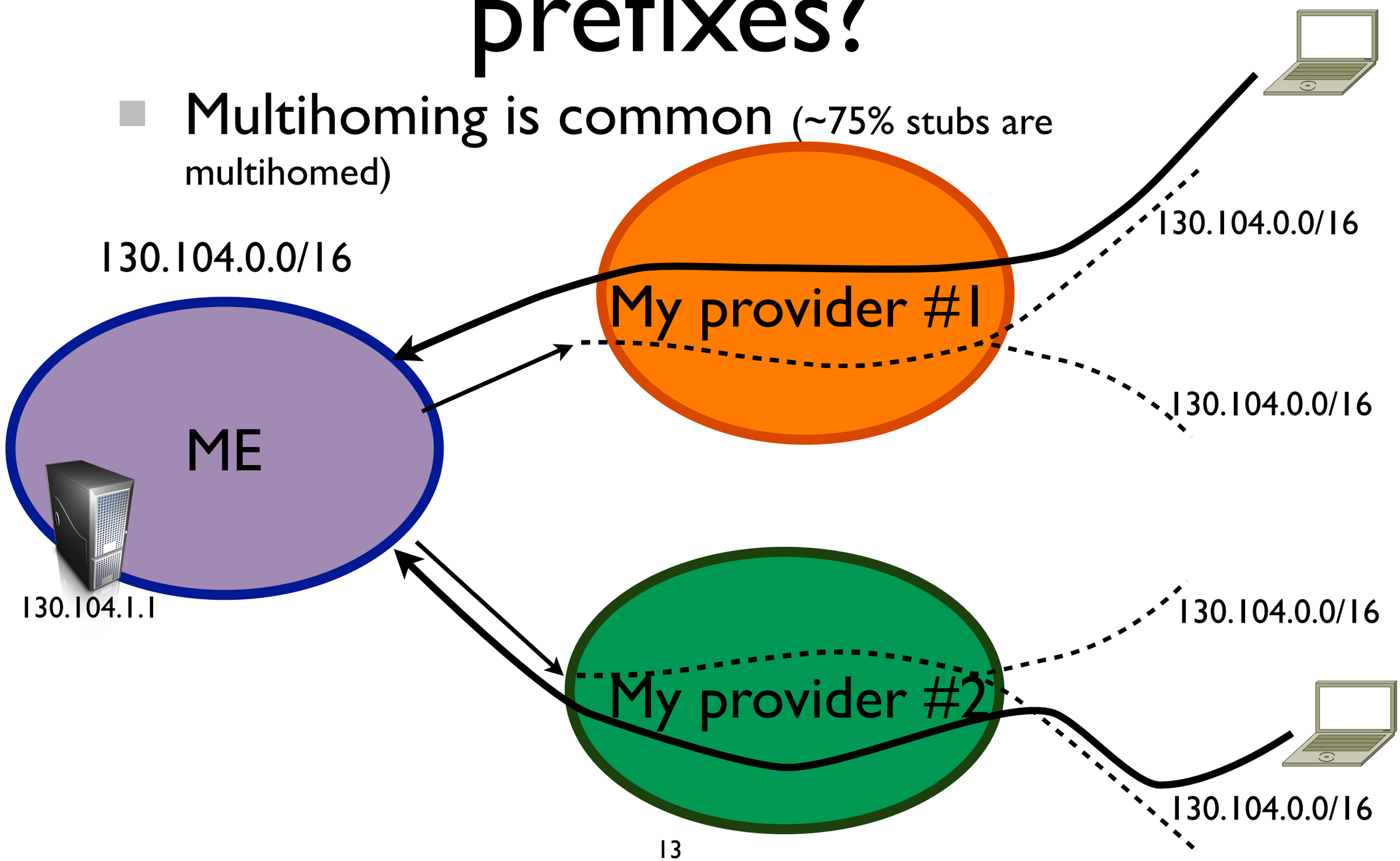
Why so many (small) prefixes?

- Multihoming is common (~75% stubs are multihomed)



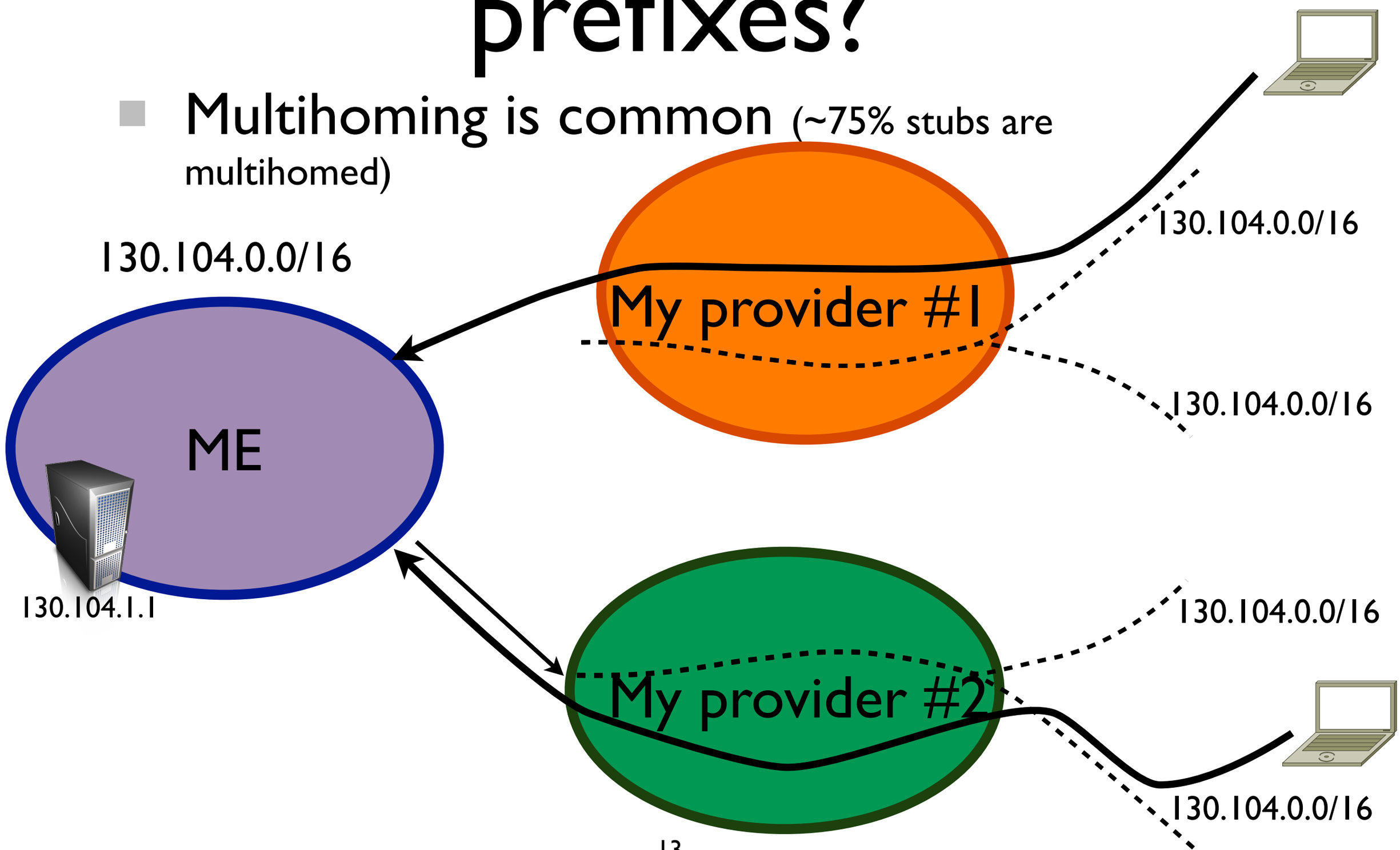
Why so many (small) prefixes?

- Multihoming is common (~75% stubs are multihomed)



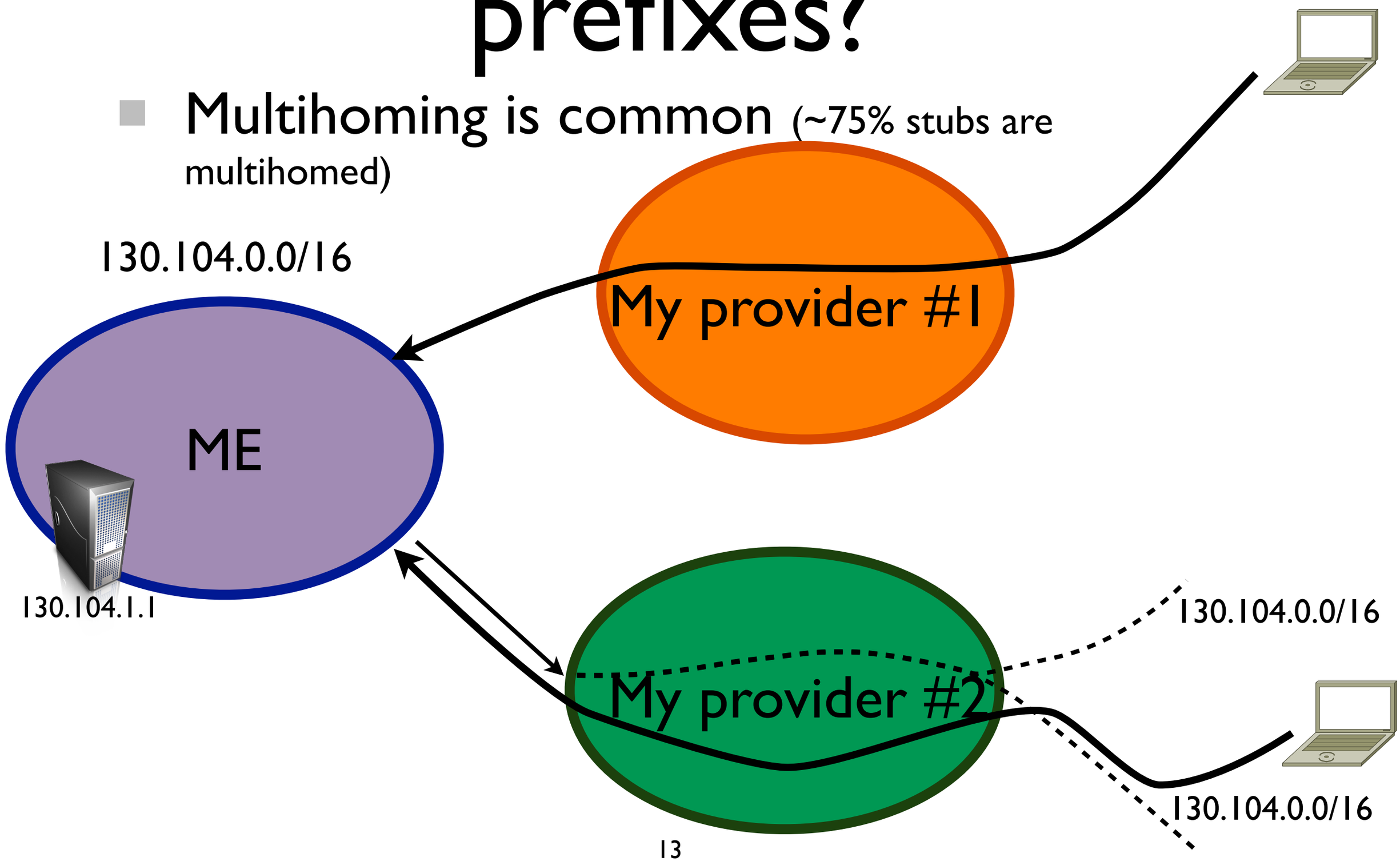
Why so many (small) prefixes?

- Multihoming is common (~75% stubs are multihomed)



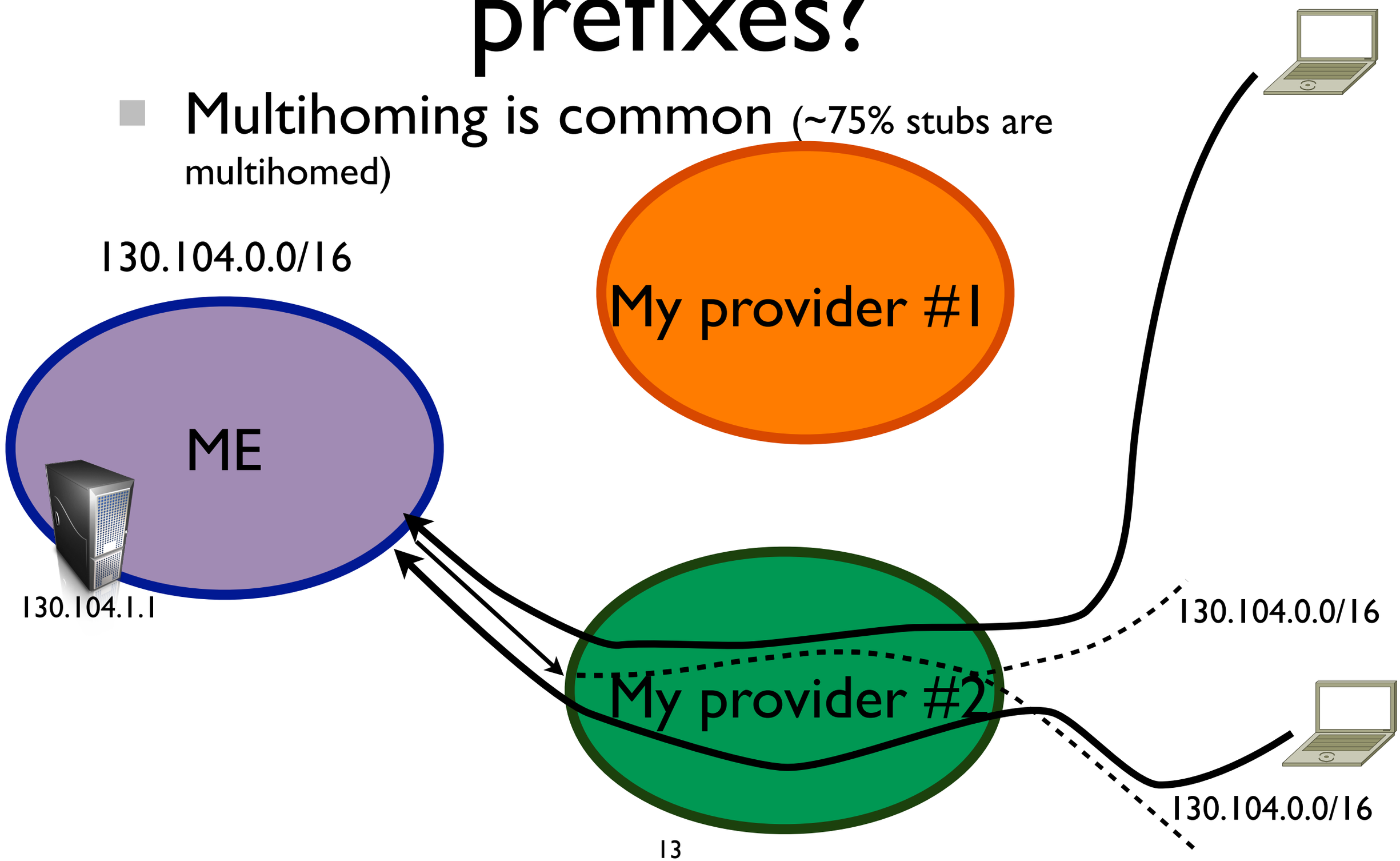
Why so many (small) prefixes?

- Multihoming is common (~75% stubs are multihomed)



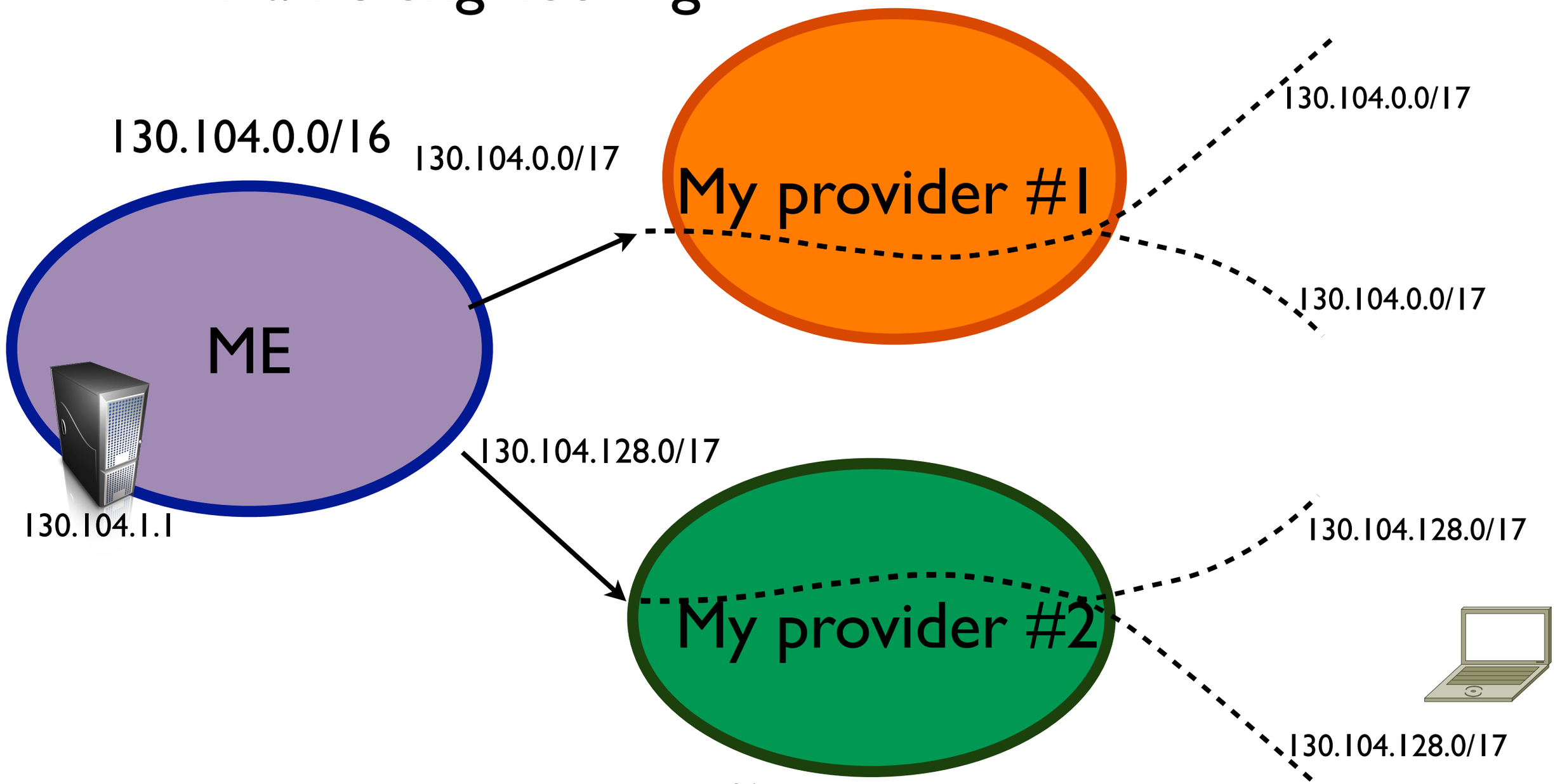
Why so many (small) prefixes?

- Multihoming is common (~75% stubs are multihomed)



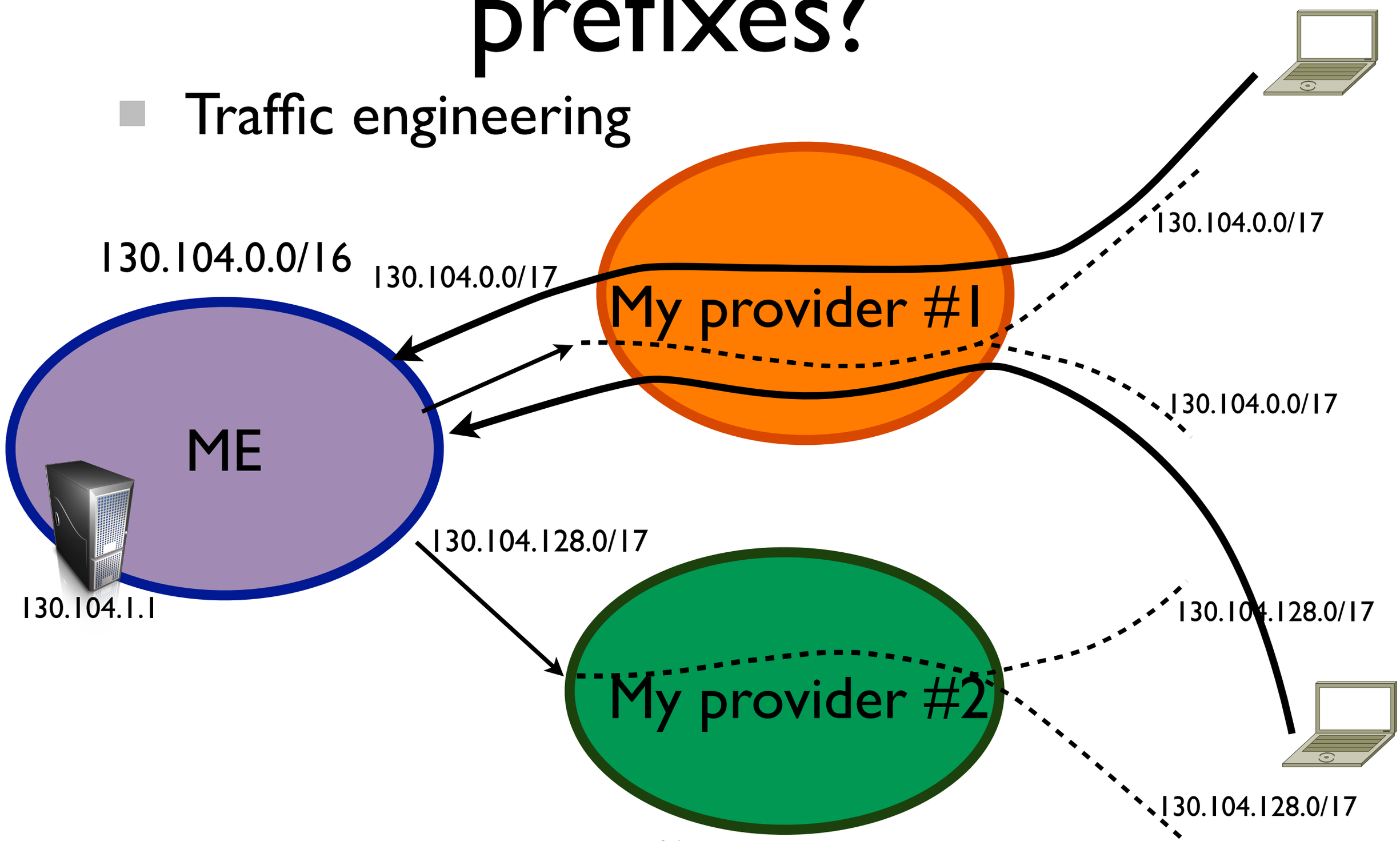
Why so many (small) prefixes?

- Traffic engineering



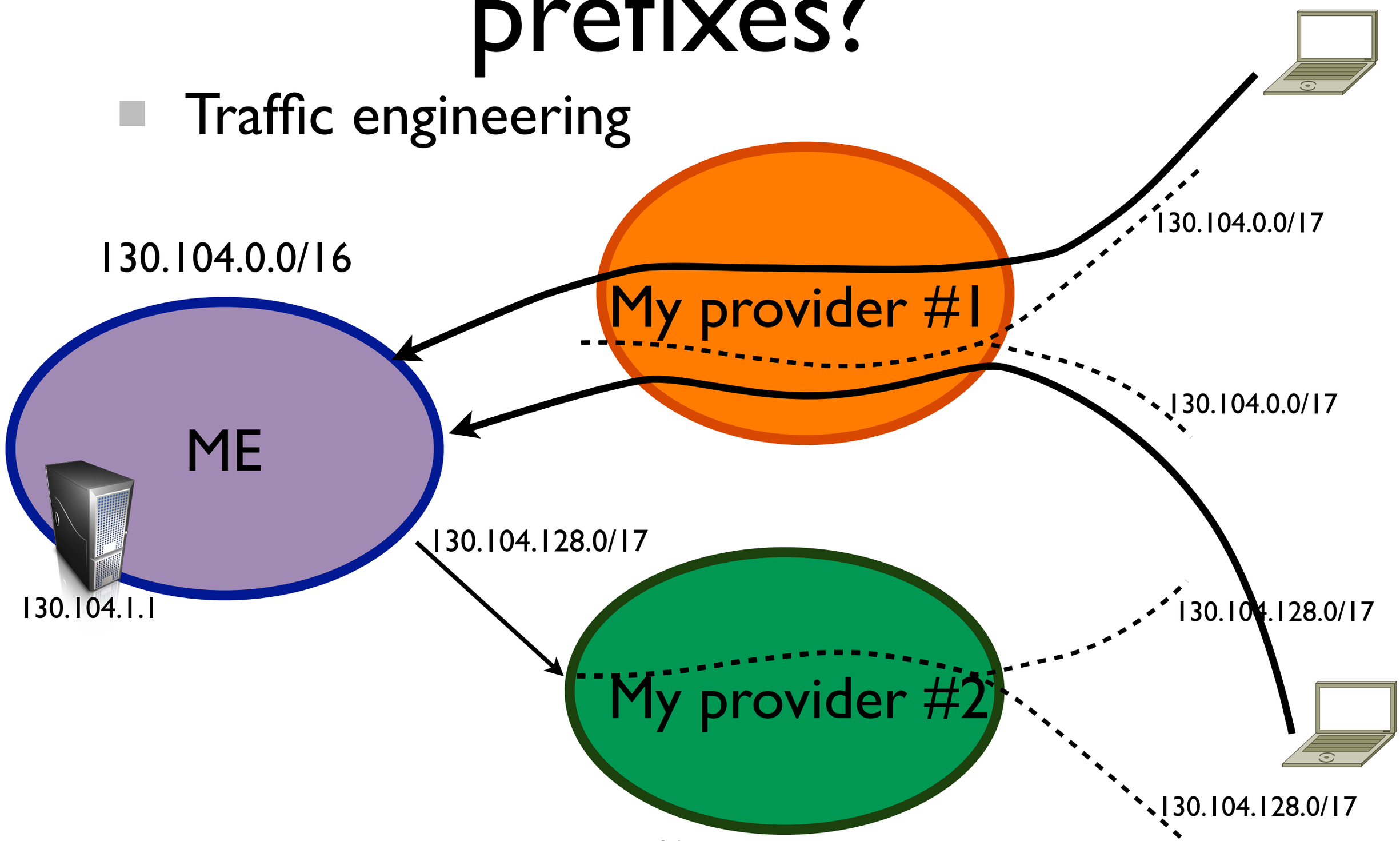
Why so many (small) prefixes?

- Traffic engineering



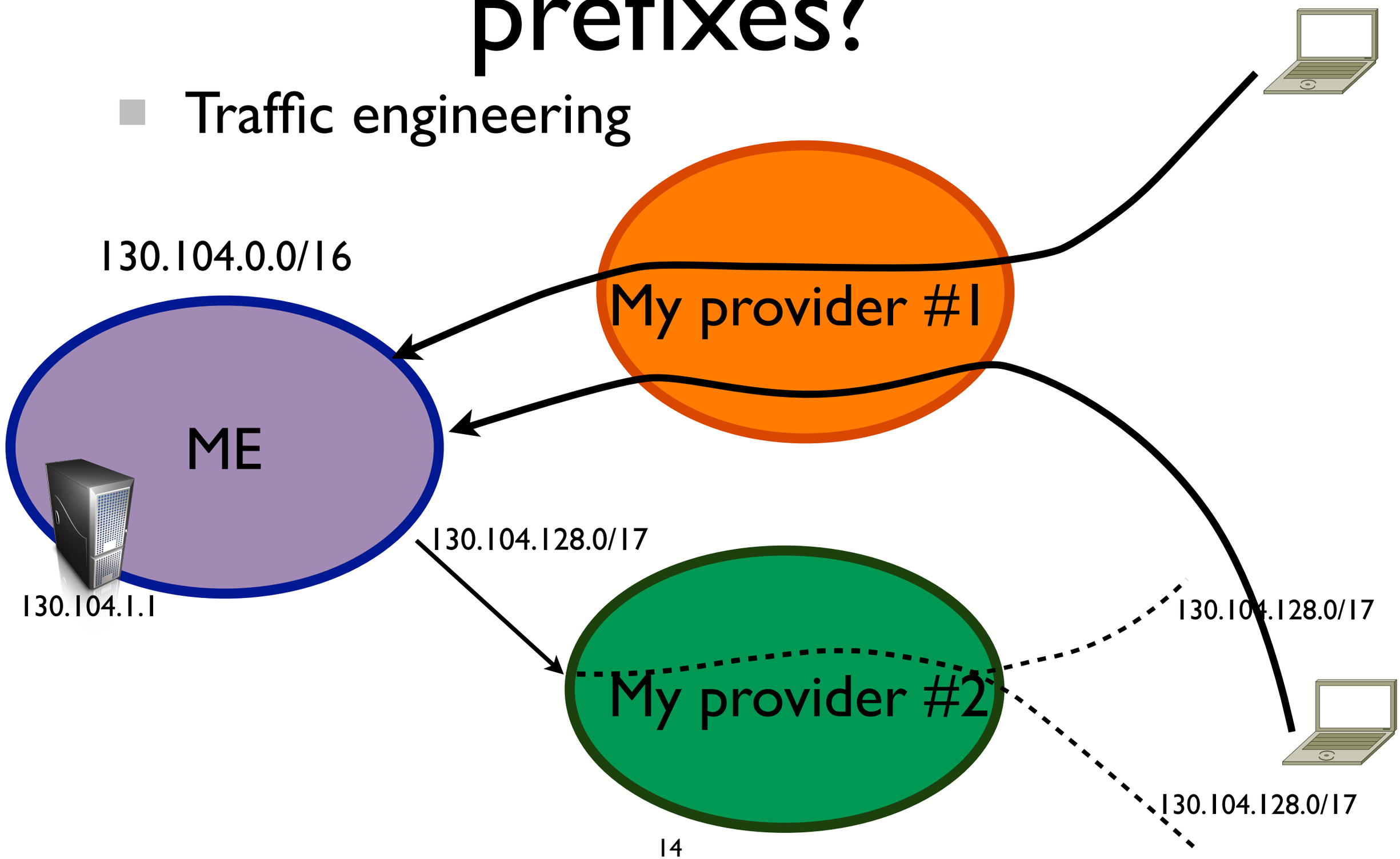
Why so many (small) prefixes?

- Traffic engineering



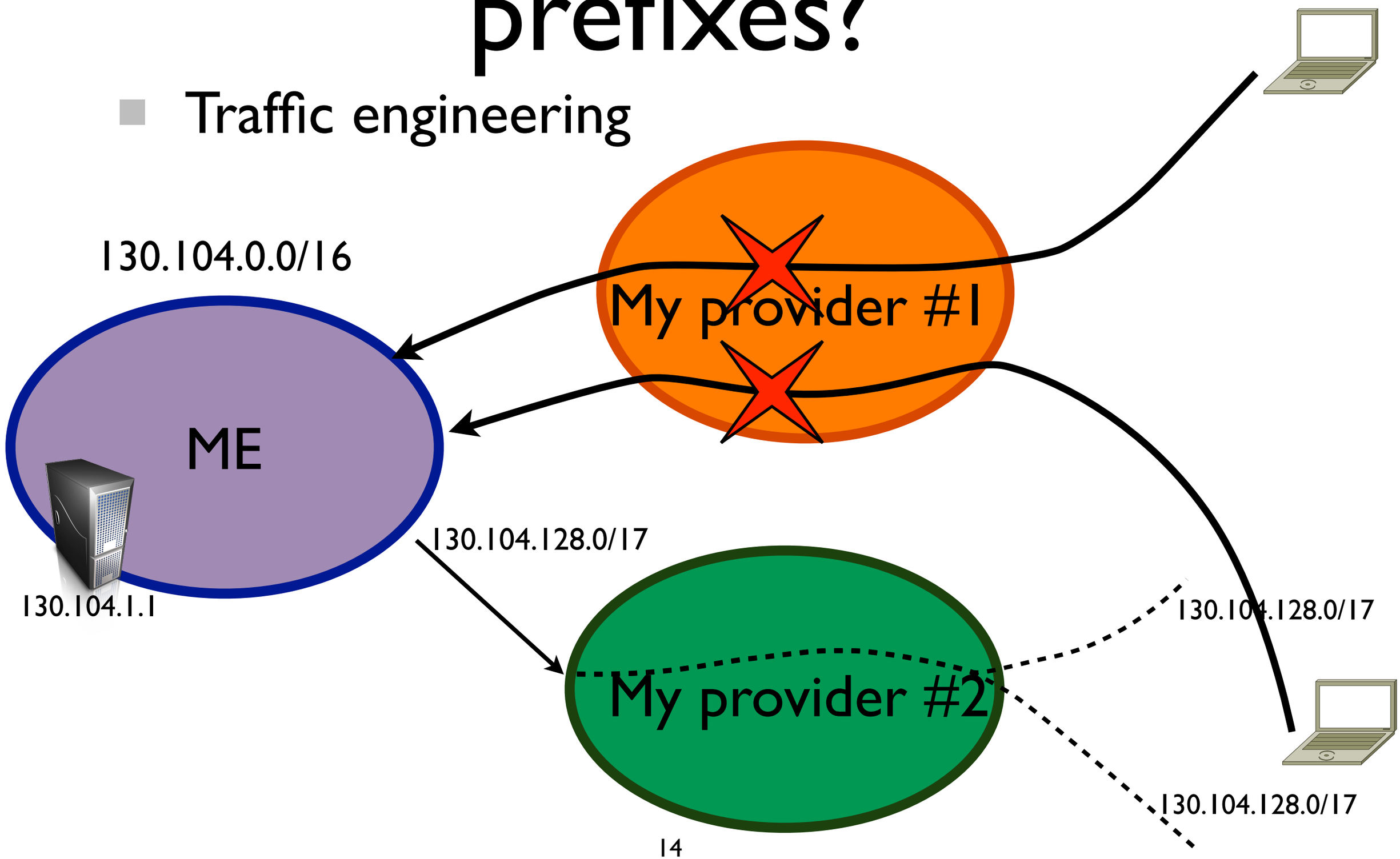
Why so many (small) prefixes?

- Traffic engineering



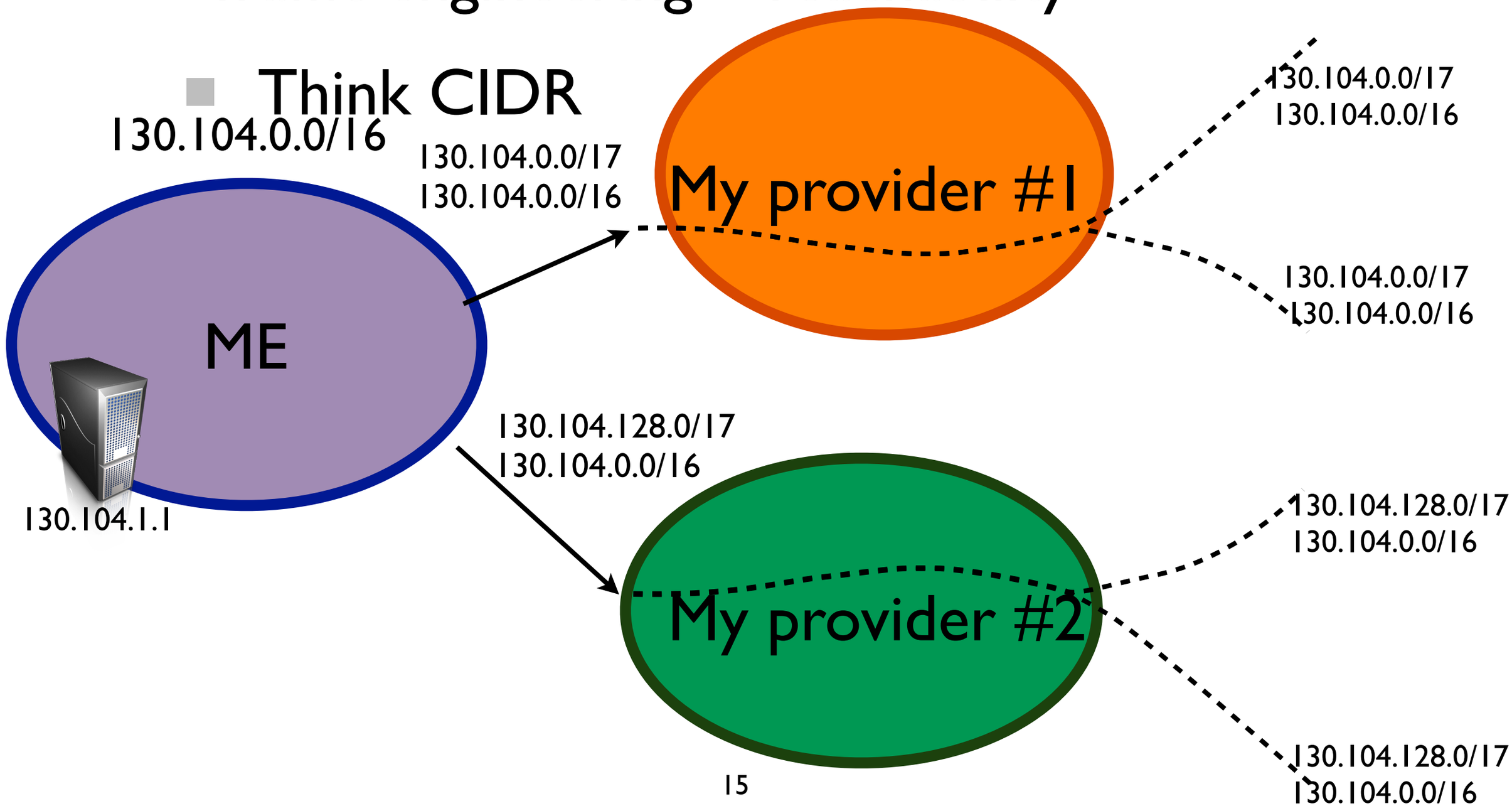
Why so many (small) prefixes?

- Traffic engineering



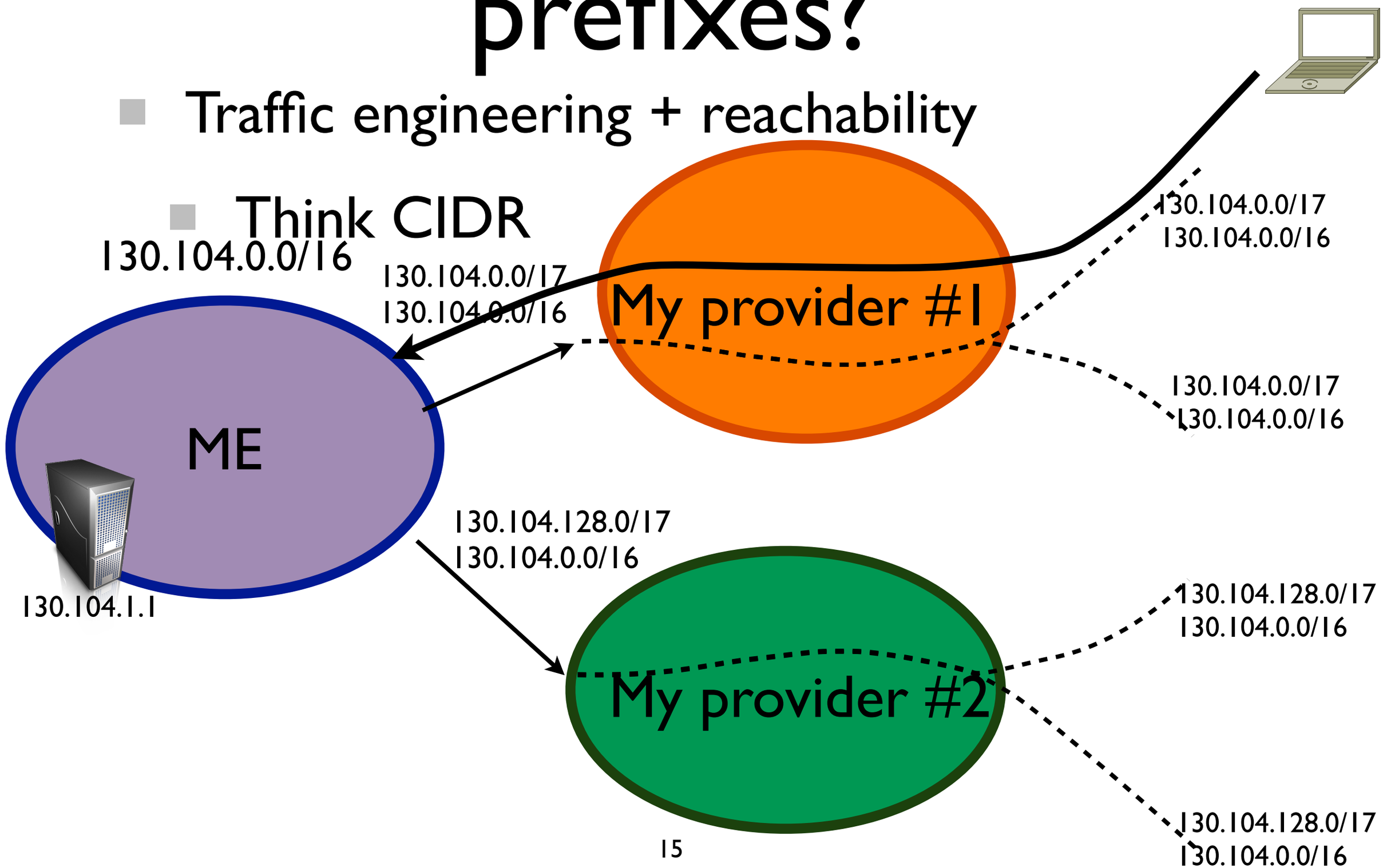
Why so many (small) prefixes?

- Traffic engineering + reachability
  - Think CIDR



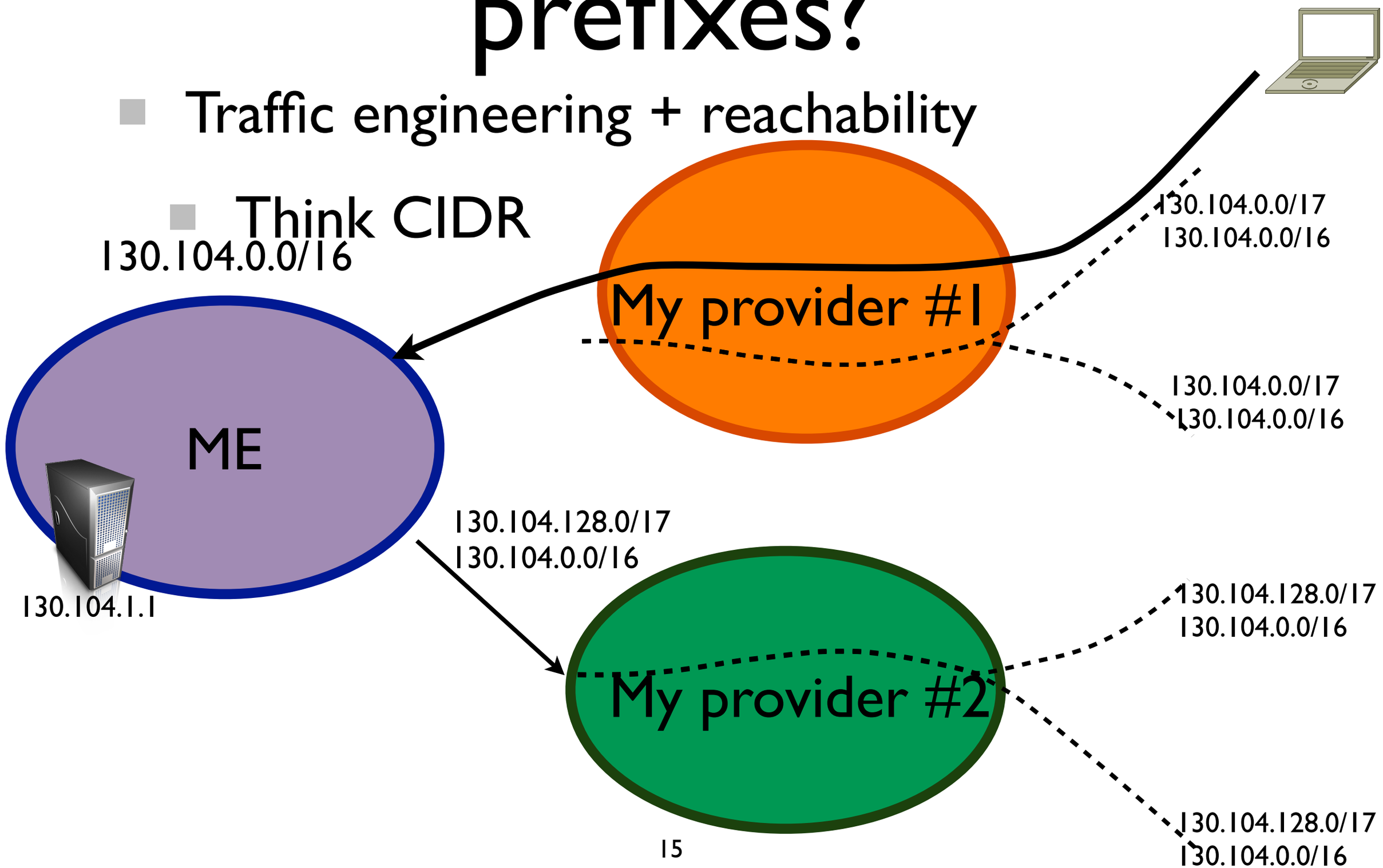
Why so many (small) prefixes?

- Traffic engineering + reachability
  - Think CIDR



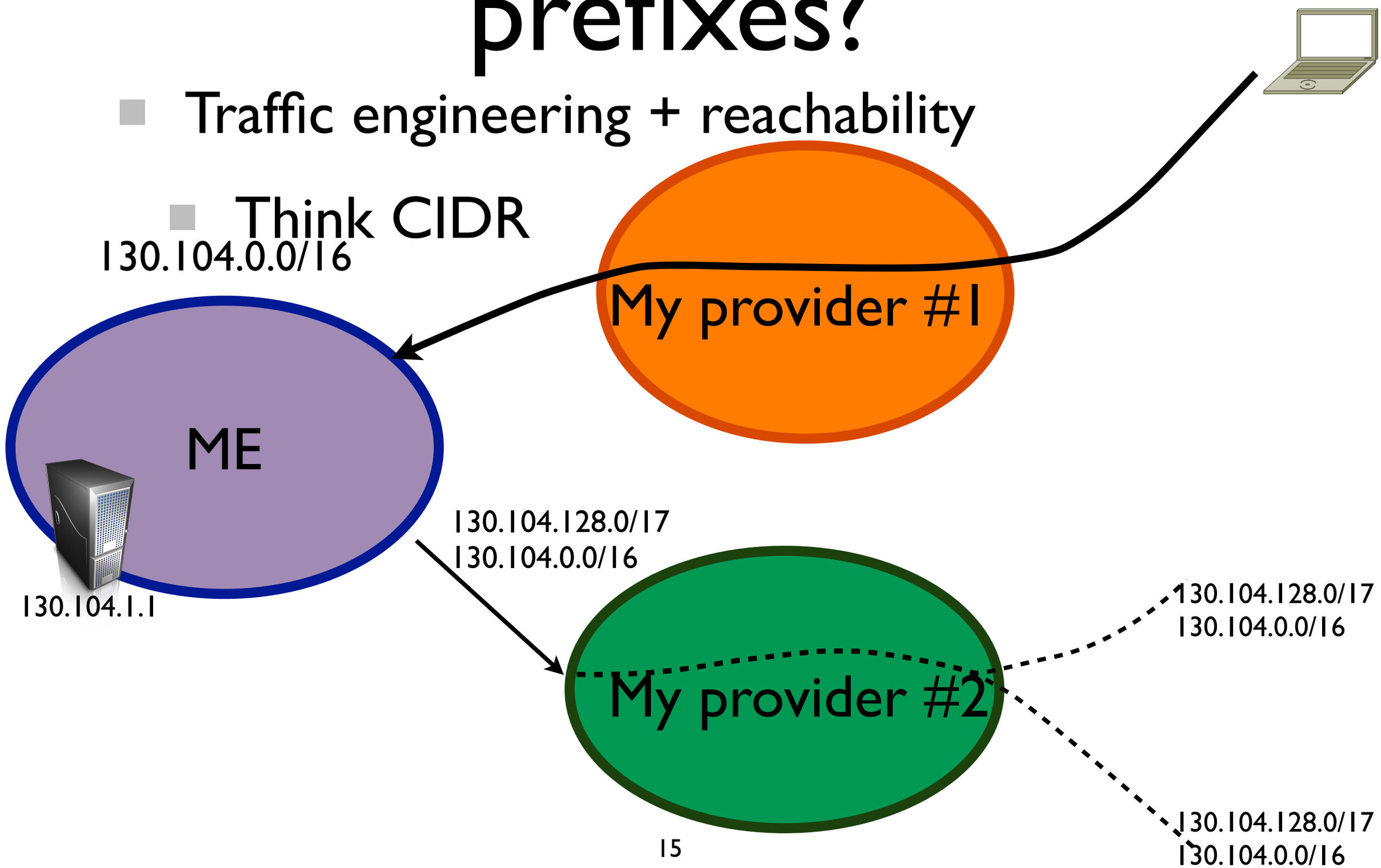
Why so many (small) prefixes?

- Traffic engineering + reachability
  - Think CIDR



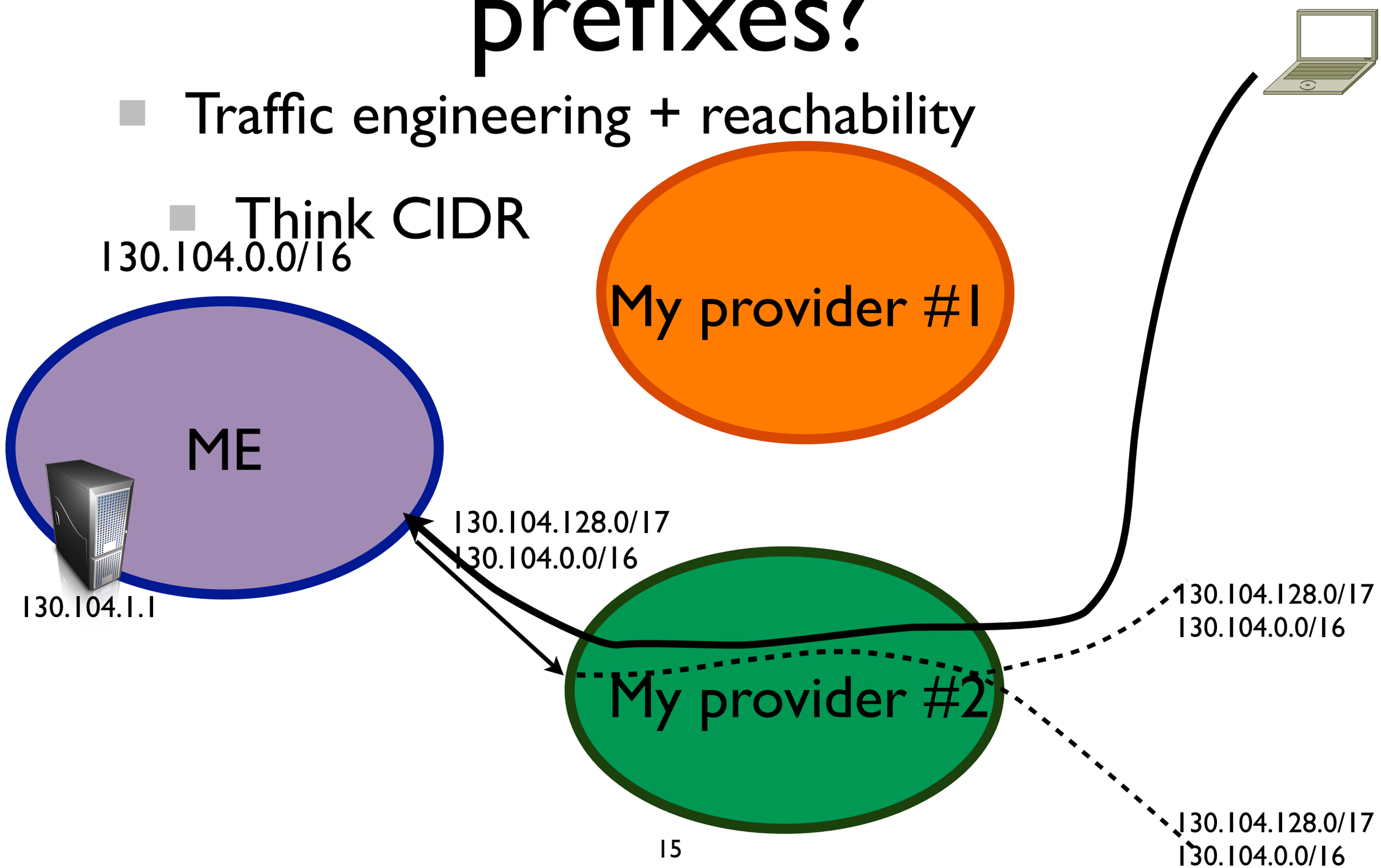
Why so many (small) prefixes?

- Traffic engineering + reachability
  - Think CIDR



Why so many (small) prefixes?

- Traffic engineering + reachability
  - Think CIDR



Routing table

- Prefixes are announced to all the Internet
  - FIB size may become a problem
- Changes are potentially seen by everybody
  - Churn may become a problem
    - ~10 to ~1000+ updates/sec is common

Traffic Engineering

- Outgoing TE is “easy”
- Incoming TE is “hard”
  - Injection of prefixes limited to /24s and prone to aggregation
  - BGP tweaks (prepending&co) are tweaks
  - Hard/costly to have per client TE

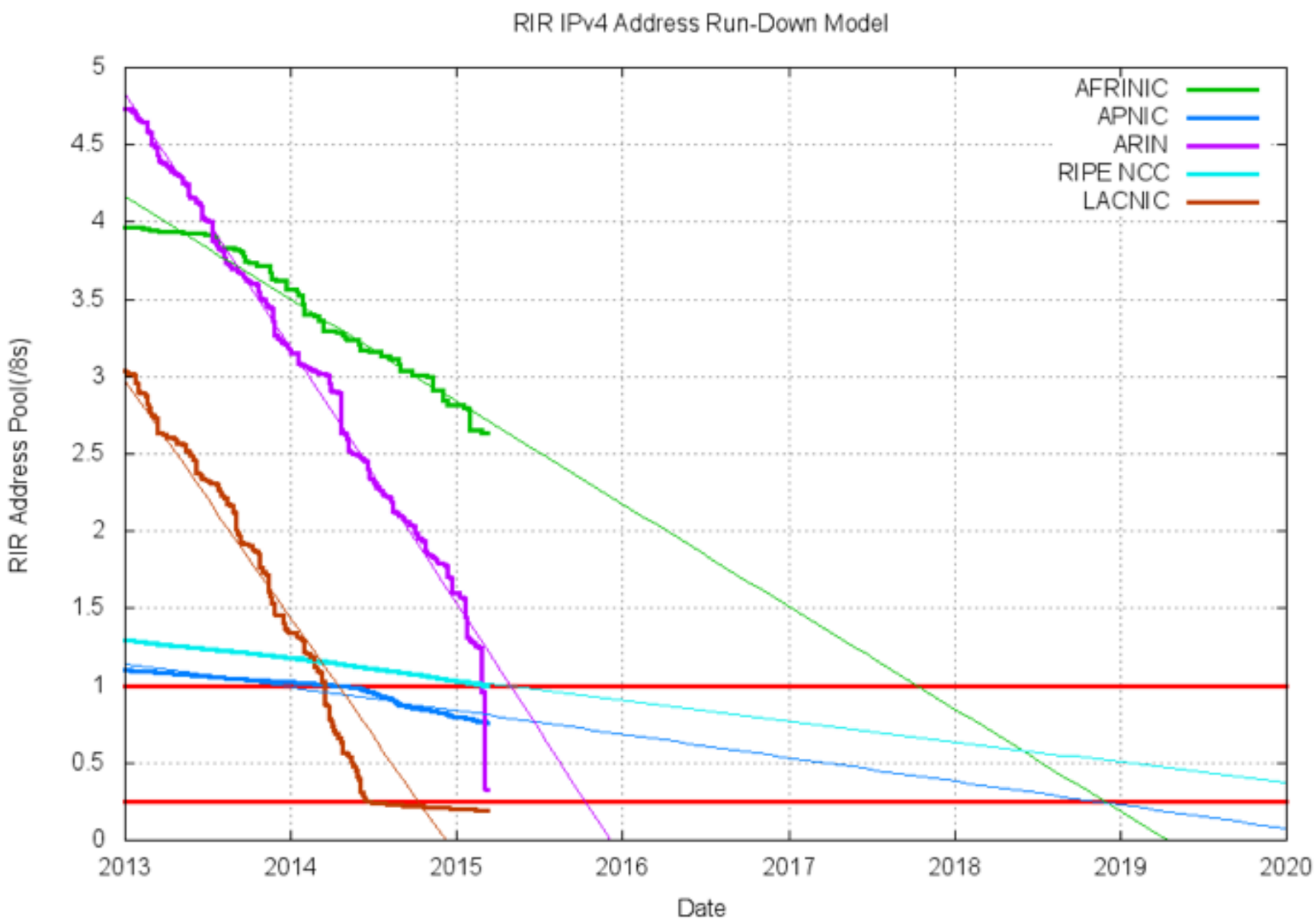
IPv4 exhaustion

- Hum, yes, in top of this we are experiencing IPv4 addresses exhaustion!

- IANA Unallocated Address Pool Exhaustion: 03-Feb-**2011**

Source: http://www.potaroo.net/tools/ipv4/

IPv4 exhaustion

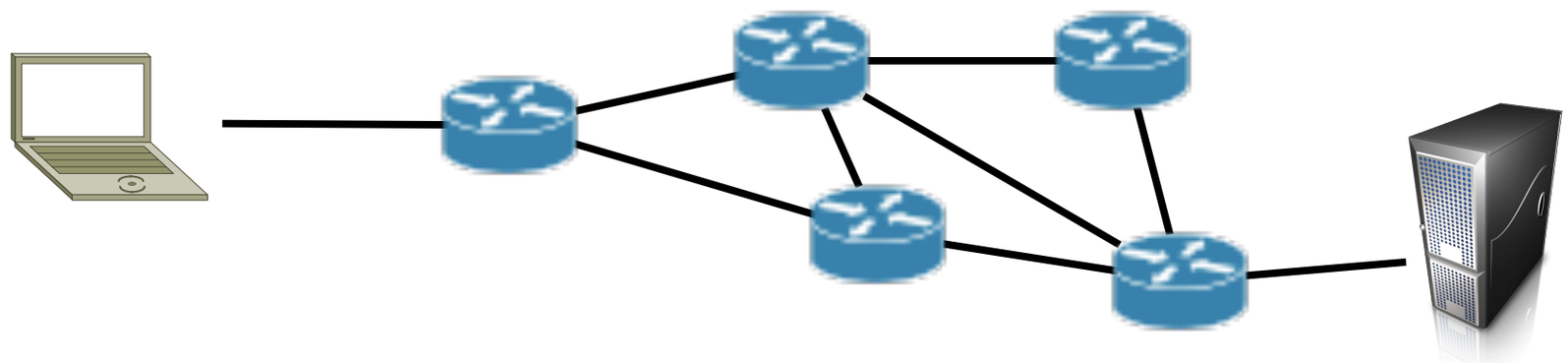


cing

Source: http://www.potaroo.net/tools/ipv4/

Technique vs users

- Communications are between two devices



- Users consume services, from anywhere

Internet is not going so well

Internet is not going so well

- Because of IP schizophrenia
  - The technique is not adapted to usages
  - Mobility is not well supported
  - Really big management overhead
  - IP addresses are becoming as rare as gold!

➡ **How to make the Internet better?**

Internet is not going so well

- Because of IP schizophrenia

Let's fix the Internet

- Really big management overhead
- IP addresses are becoming as rare as gold!

➡ **How to make the Internet better?**

Today

- Network is mostly used (> 99%) to acquire chunks of names data
  - web pages
  - videos
  - torrents
- Retrieving data is not a *conversation*, it is a *dissemination*

Only data matter

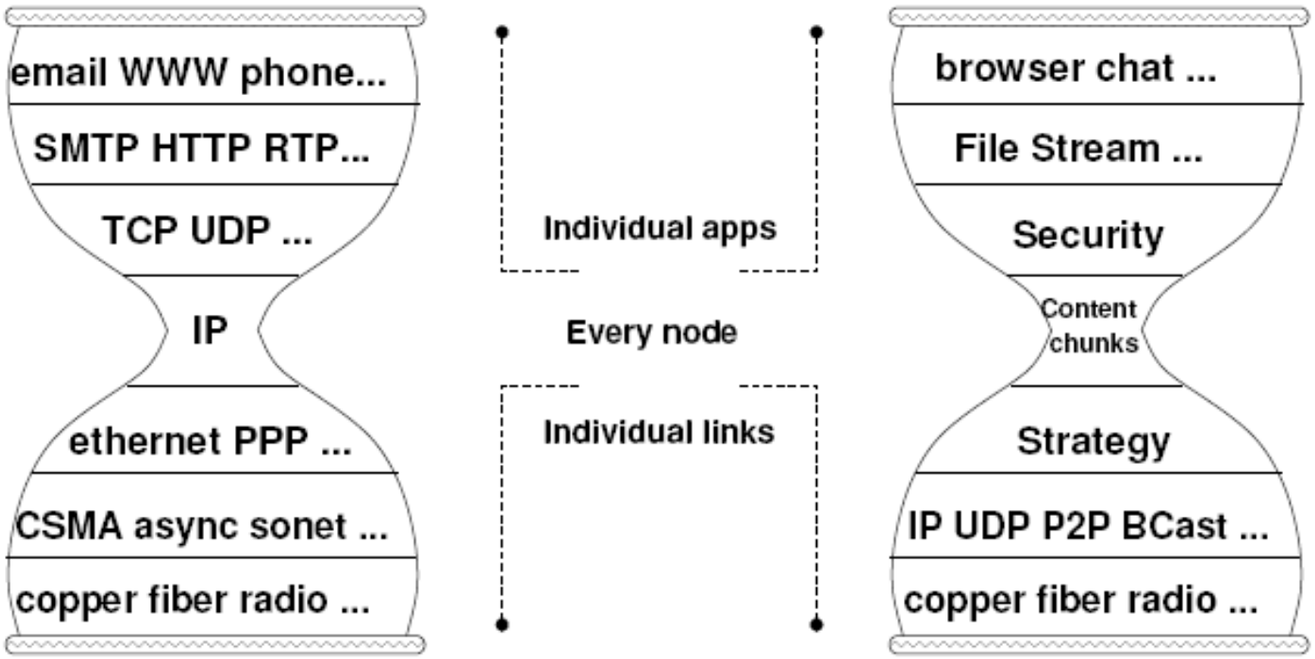
- Conversational protocol can be used to disseminate data but
  - User goal and realisation is the result of compromises and plumbing
  - Security (e.g., SSL) is not adapted as it is the transmission of the data that is protected, not the data itself
    - requires an out-of-band mechanism to verify the data itself
  - Lack of efficiency as the end-to-end path is strictly given by the conversation end-points (client - server)

How to reconcile the two worlds?

- **Clean-slate solutions**
  - Rethink the paradigms
- Evolutionary solutions
  - Enhance current architecture
  - Provide interworking mechanisms

Content-Centric Networking (CCN)

- Shift from location-based to content-based communications
  - Contents become first class citizens in the network



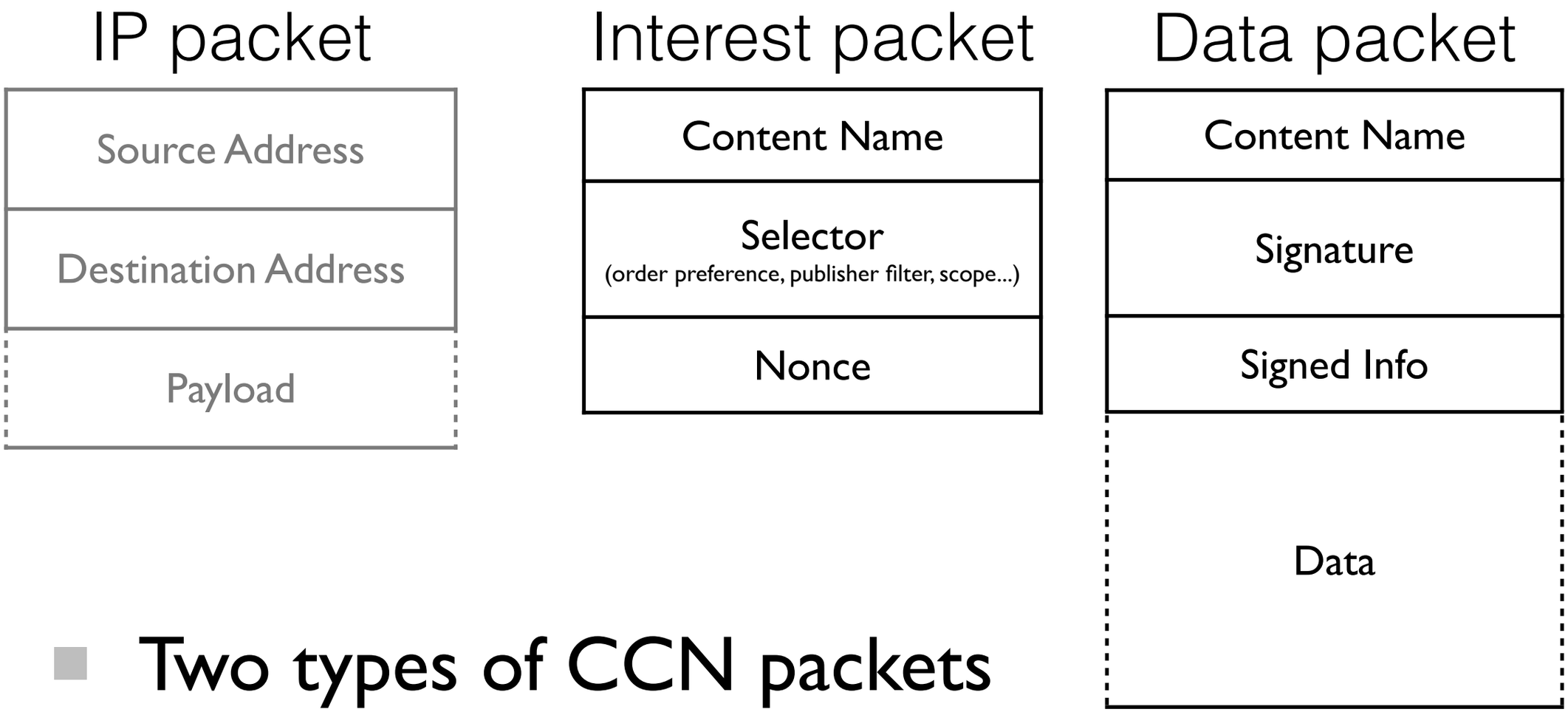
The idea

- Content-Centric Networking (CCN) treats content as a primitive [JST+09]
- Every chunk of data is assigned a name, such that any content can be directly retrieved by its name
- Routers cache chunks of data on-path between consumers and producers

Workflow

- A content consumer (client) asks for content by sending an Interest packet to nodes at its direct neighborhood
- A node that has data that satisfies the interest responds with a Data packet
- Otherwise, the node forwards the Interest packet to its neighbors, and remembers from which neighbors it received the interest

Packets



- Two types of CCN packets
- Packets indicate the what, not the who or the where (neither source nor destination)

What does it change?

- Shift from location to content based communications
- Shift from end-to-end to local communications

What does it change?

- Shift from location to content based communications
- Shift from end-to-end to local communications
- Secure data themselves instead of communication channels

What does it change?

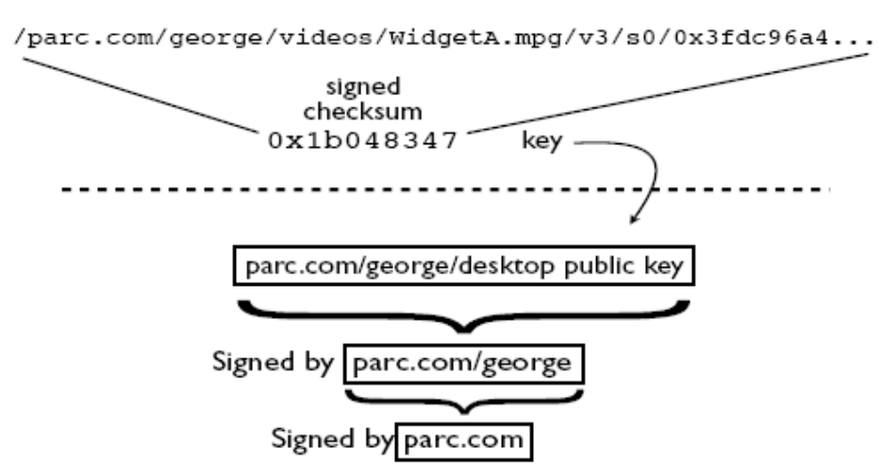
- Shift from location to content based communications
- Shift from end-to-end to local communications
- Secure data themselves instead of communication channels
- Topology is only an optimization

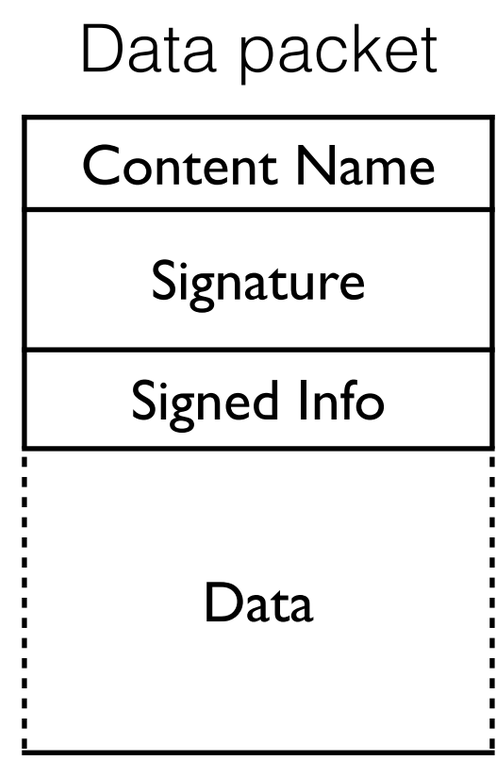
What does it change?

- Shift from location to content based communications
- Shift from end-to-end to local communications
- Secure data themselves instead of communication channels
- Topology is only an optimization
- Contents can be cached anywhere

Secure data themselves instead of communication channels

- Each chunk of data is authenticated with a digital signature
  - the signature is included in the data packet
  - relies on a publick key distribution infrastructure





Topology is only an optimization

- Consumers send their interests over any available communication link
  - Can be a broadcast
- Any node that has a copy of the requested chunk can answer the request by sending the data back
  - Enables multipath communications
  - Enables mobility
  - The closest copy of the chunk can be retrieved

Topology is only an optimization (contd.)

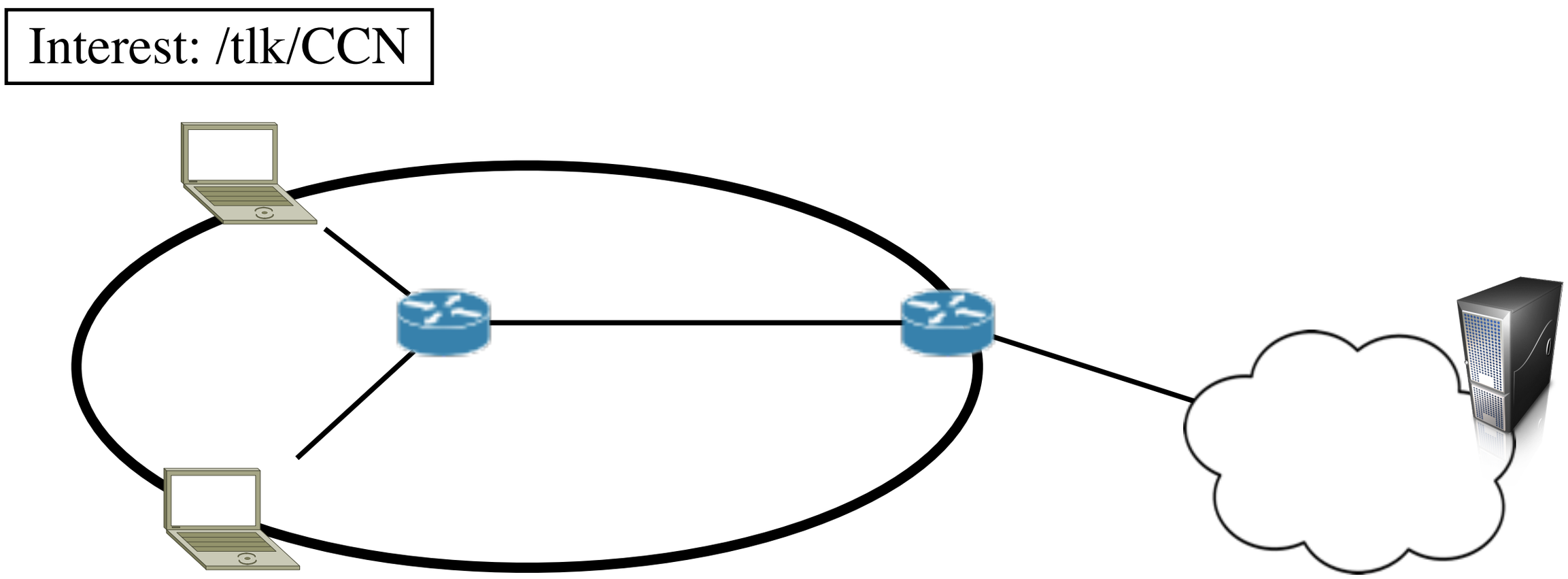
- Communications are independent of data' location
  - No particular action to take if data move to other locations
  - Possible to change network interface at anytime
- Communications are purely driven by the consumer
  - No communication state at the producer side
  - No state migration if data moves to another location

➡ Seamless mobility

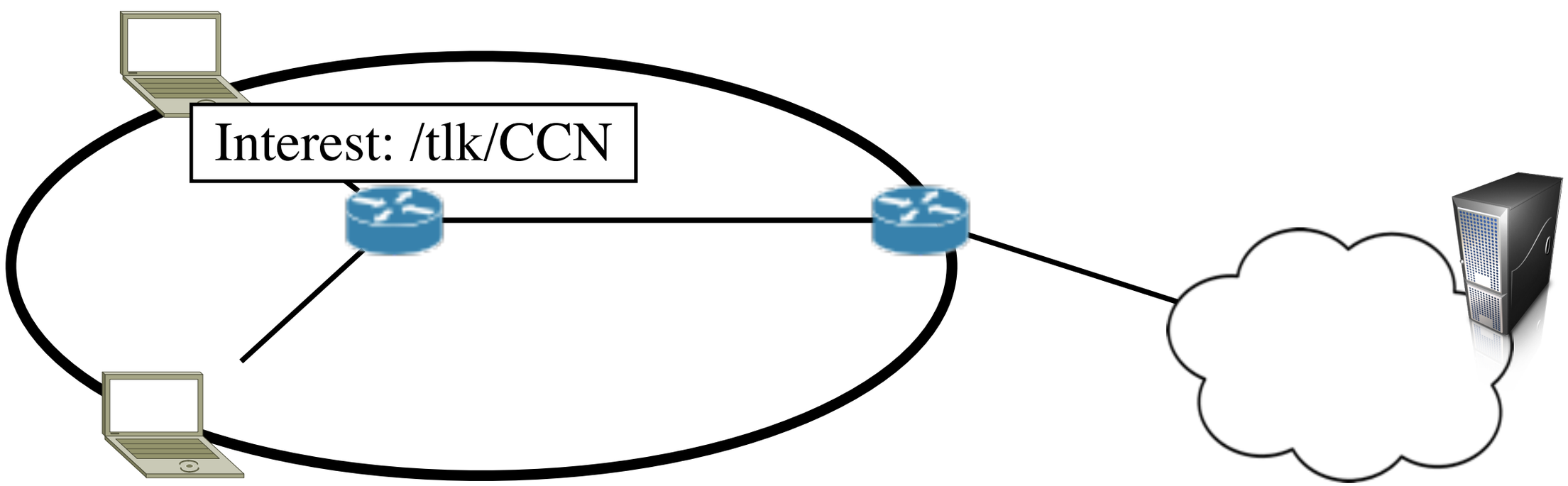
Contents can be cached anywhere

- The communication model does not consider the node that delivers the data
- Data transfer integrity is independent of the node that delivers the data
- Data can be replicated anywhere, at anytime

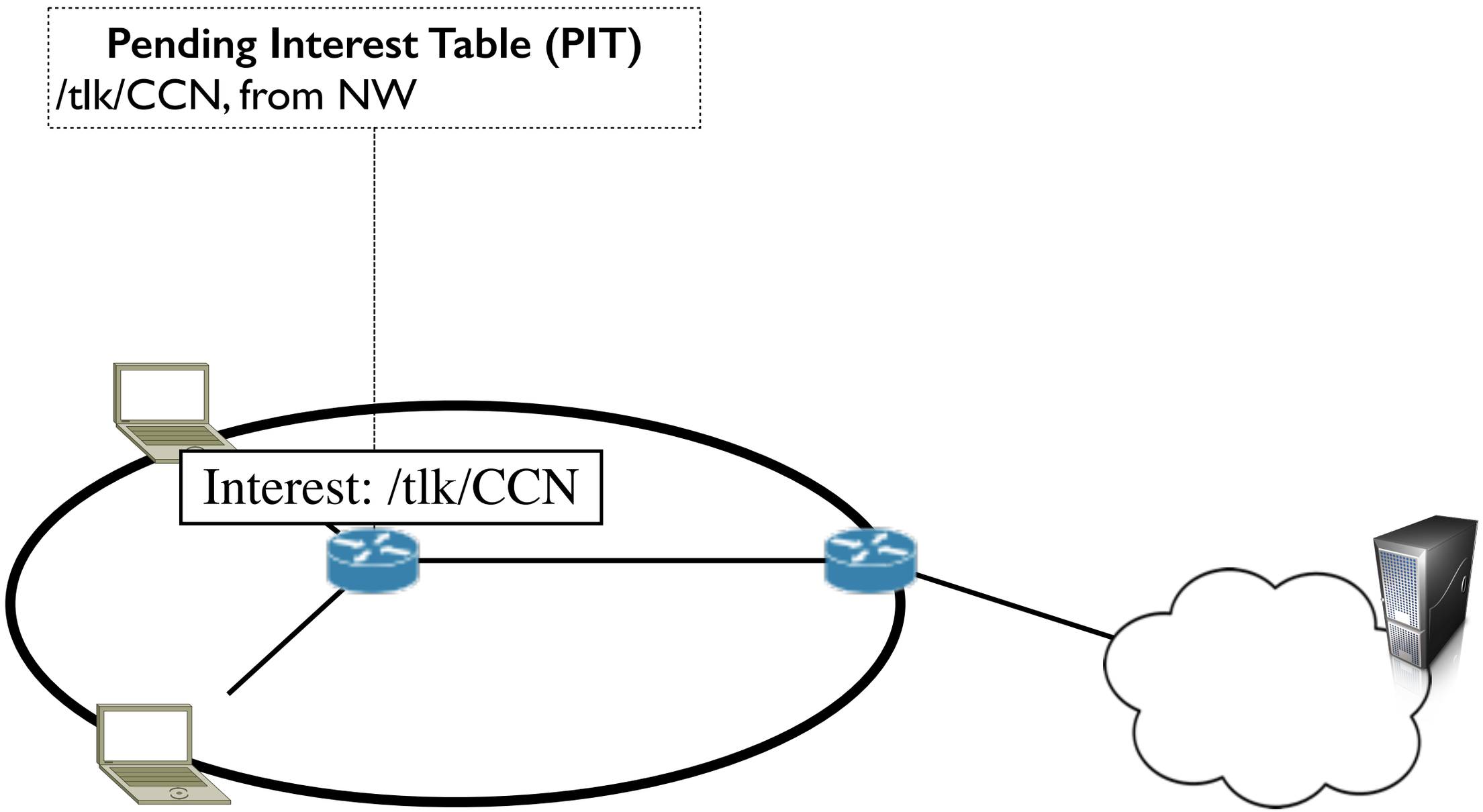
Contents can be cached anywhere (contd.)



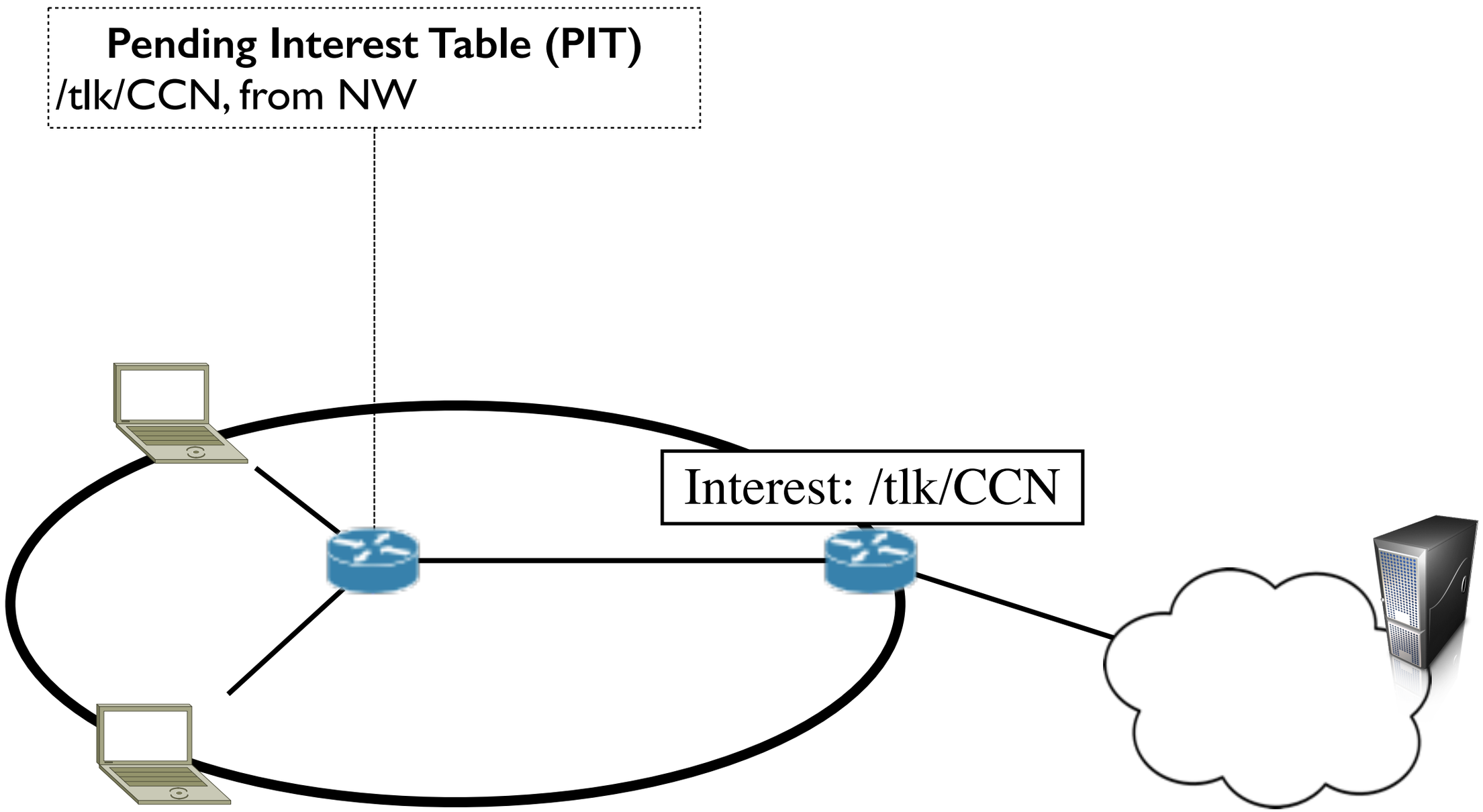
Contents can be cached anywhere (contd.)



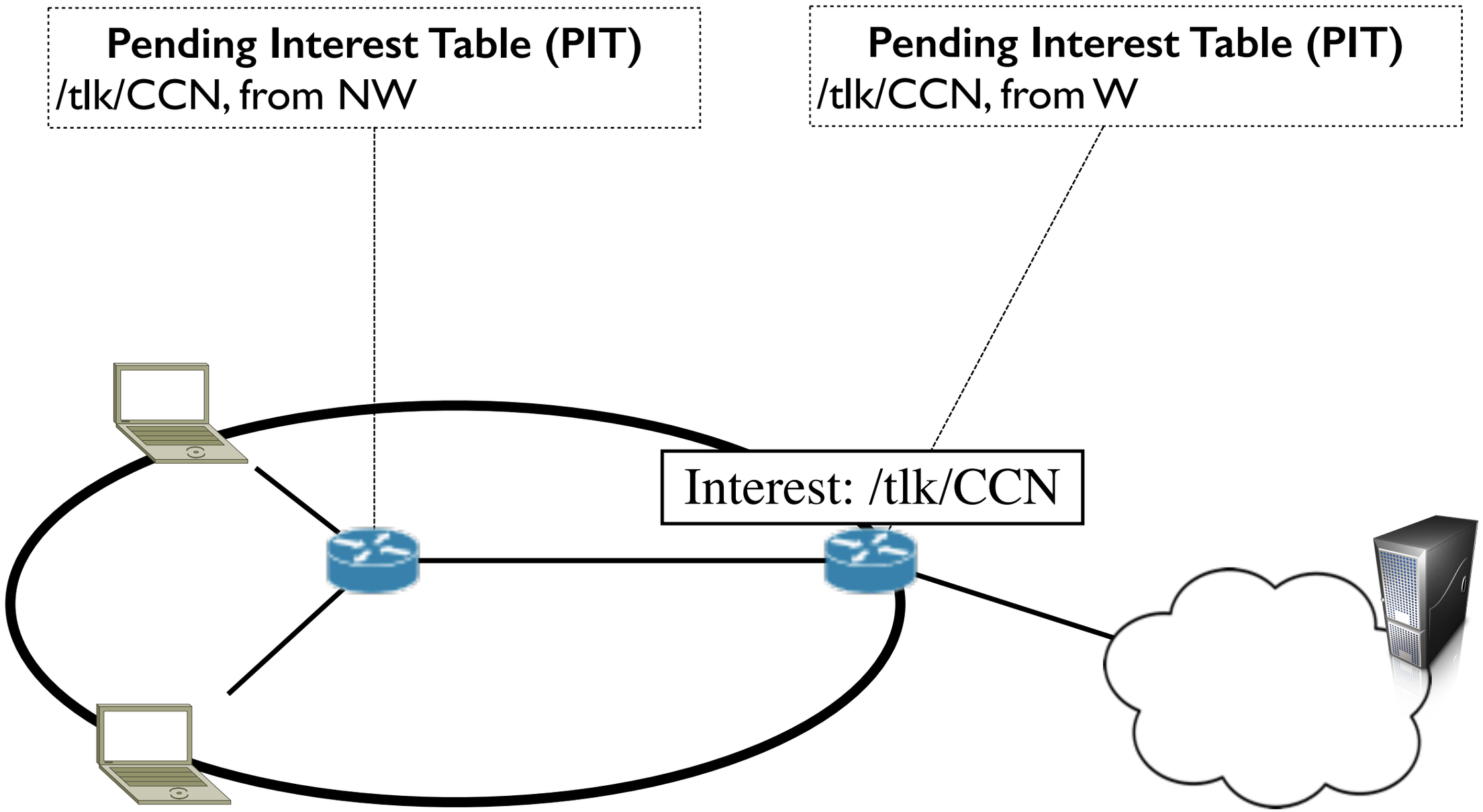
Contents can be cached anywhere (contd.)



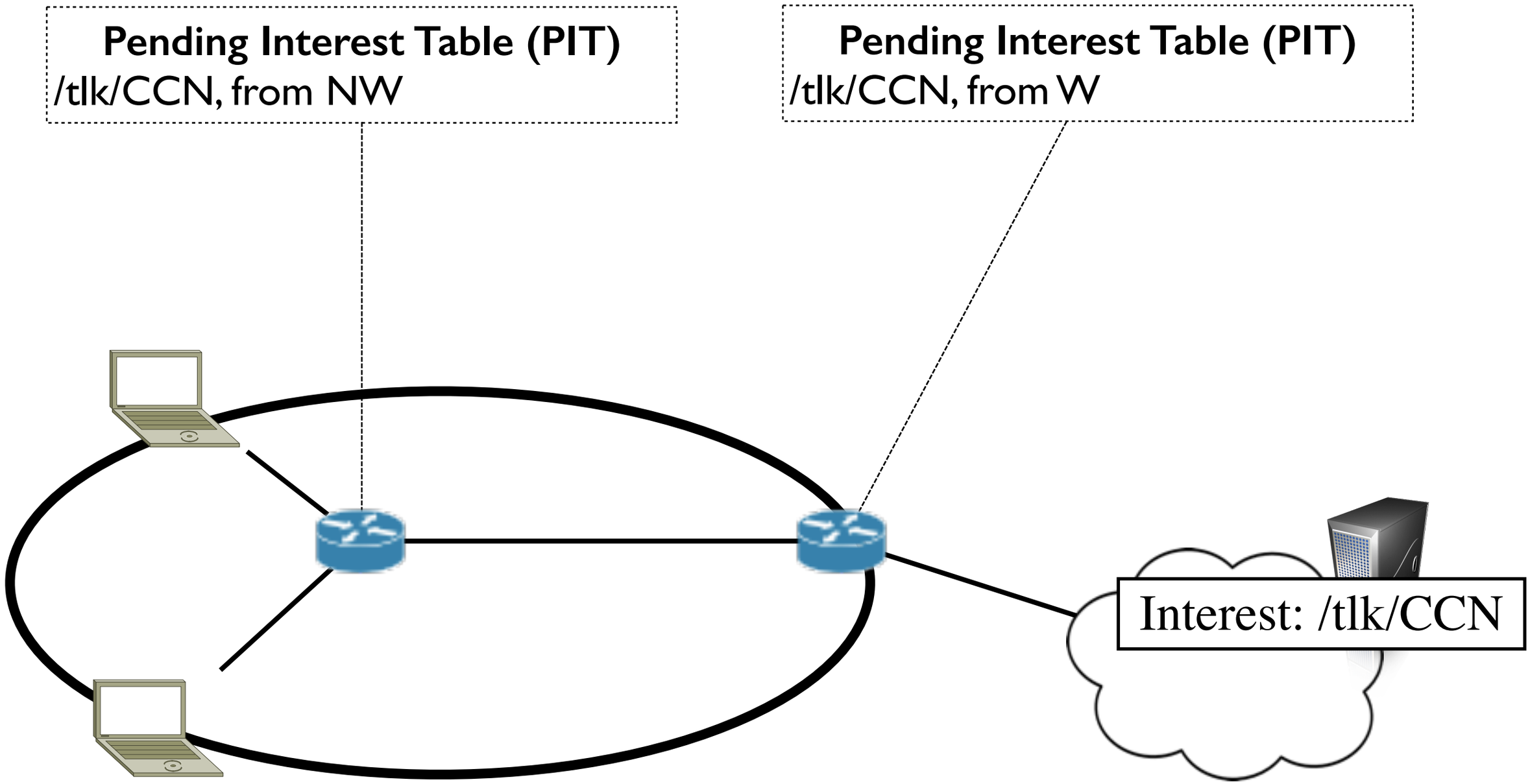
Contents can be cached anywhere (contd.)



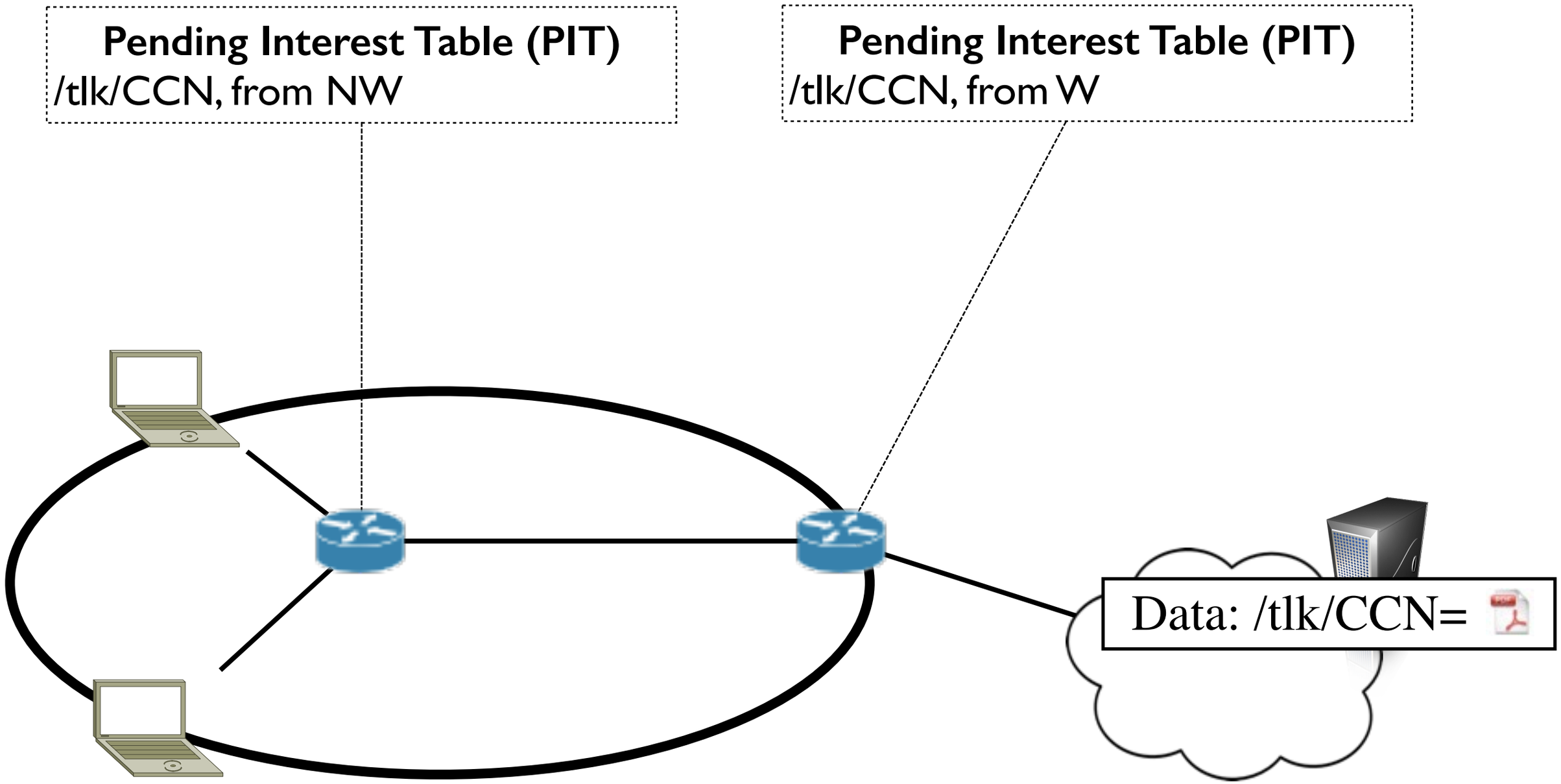
Contents can be cached anywhere (contd.)



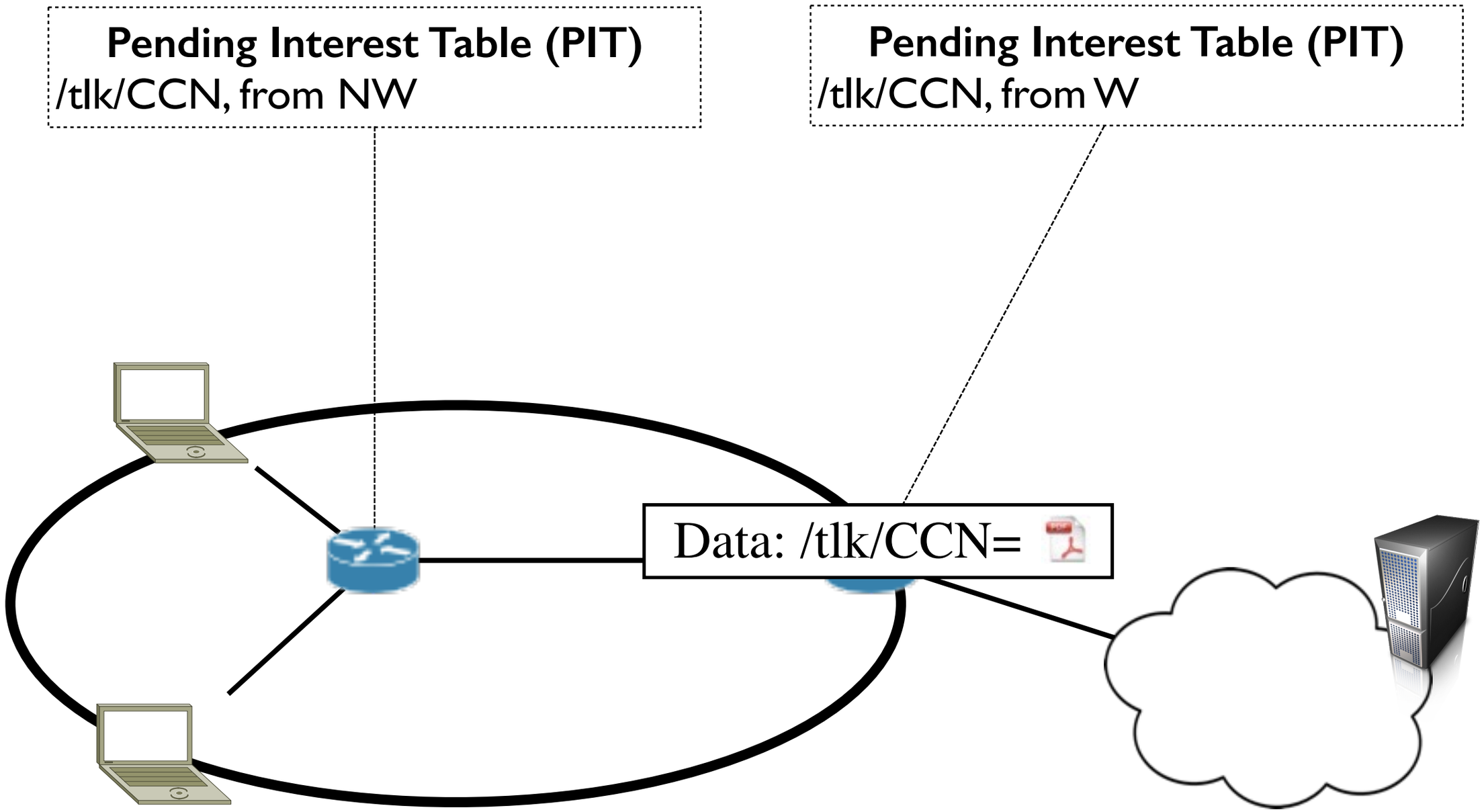
Contents can be cached anywhere (contd.)



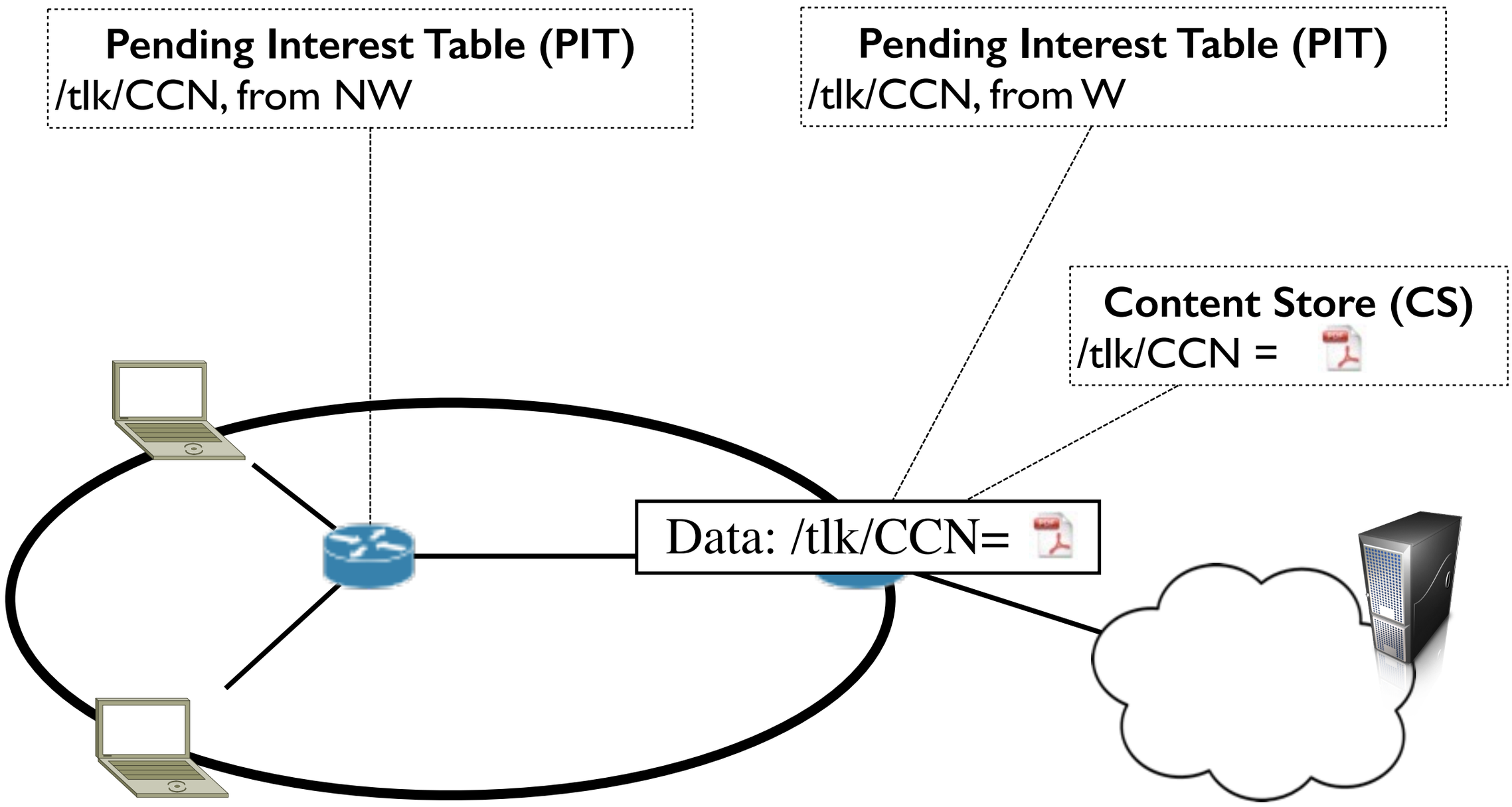
Contents can be cached anywhere (contd.)



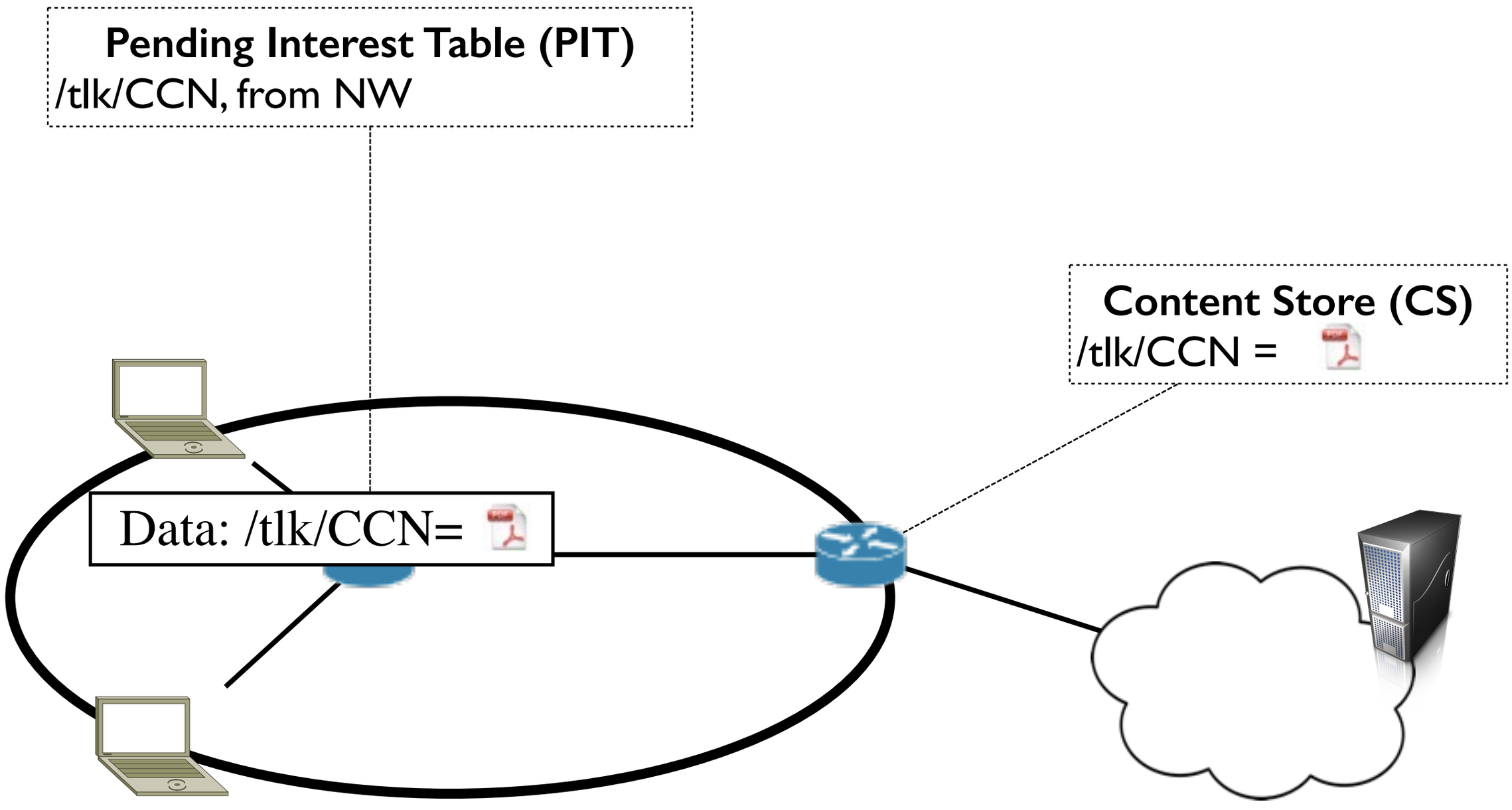
Contents can be cached anywhere (contd.)



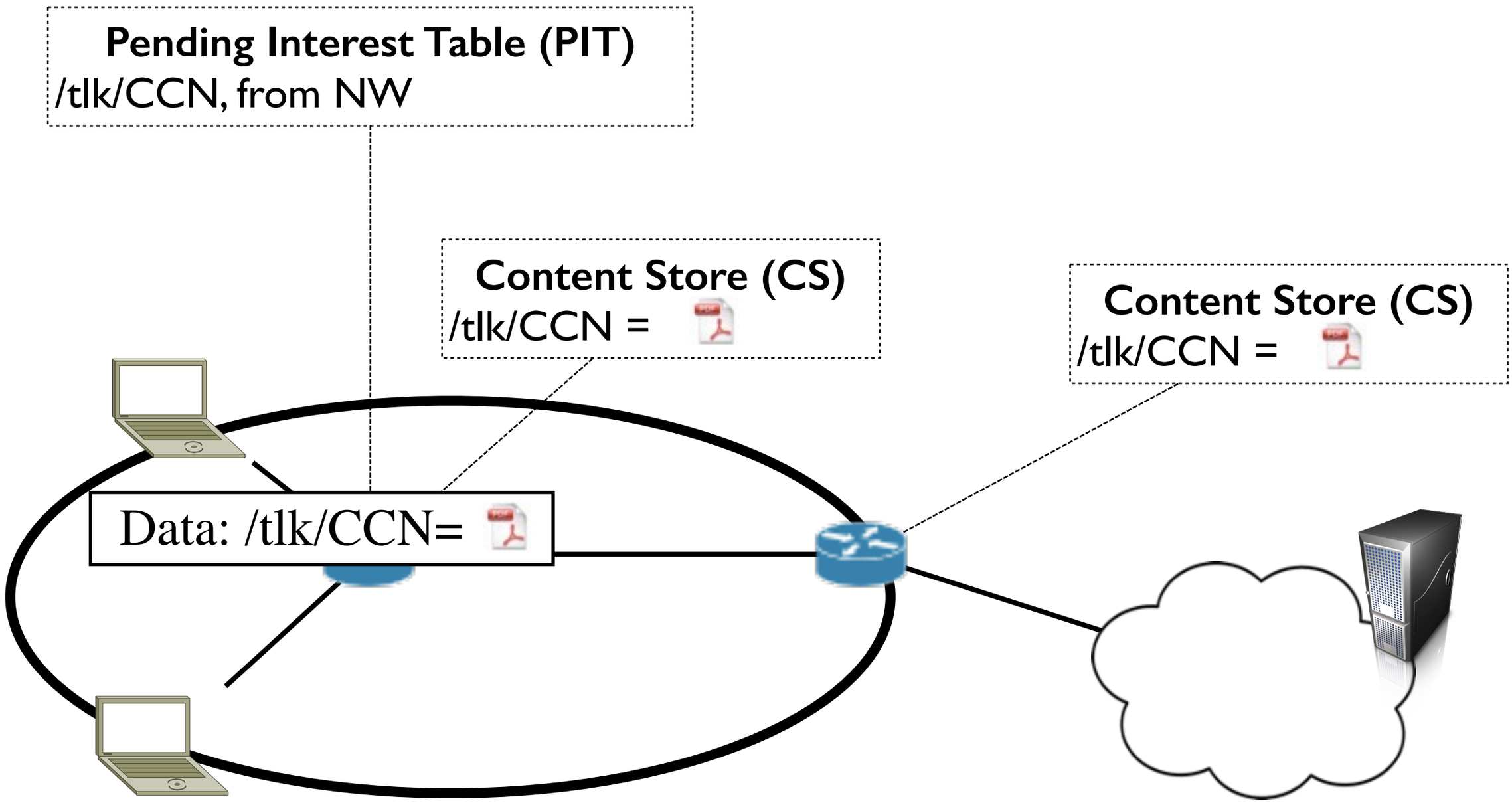
Contents can be cached anywhere (contd.)



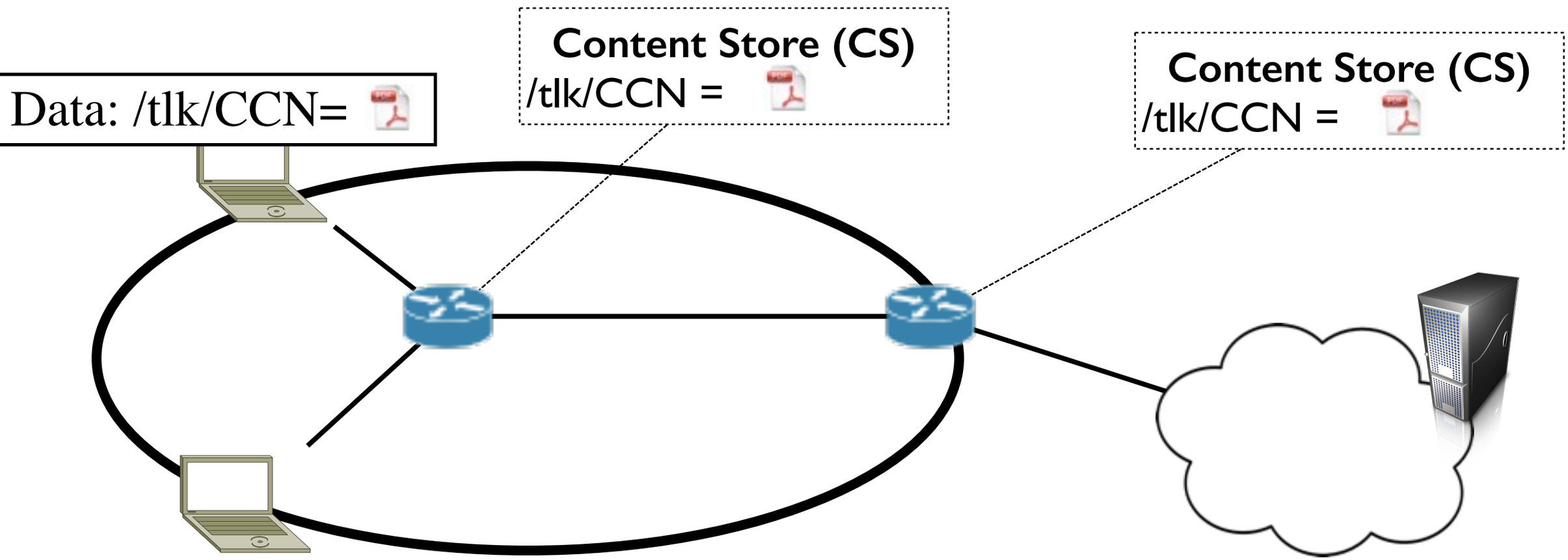
Contents can be cached anywhere (contd.)



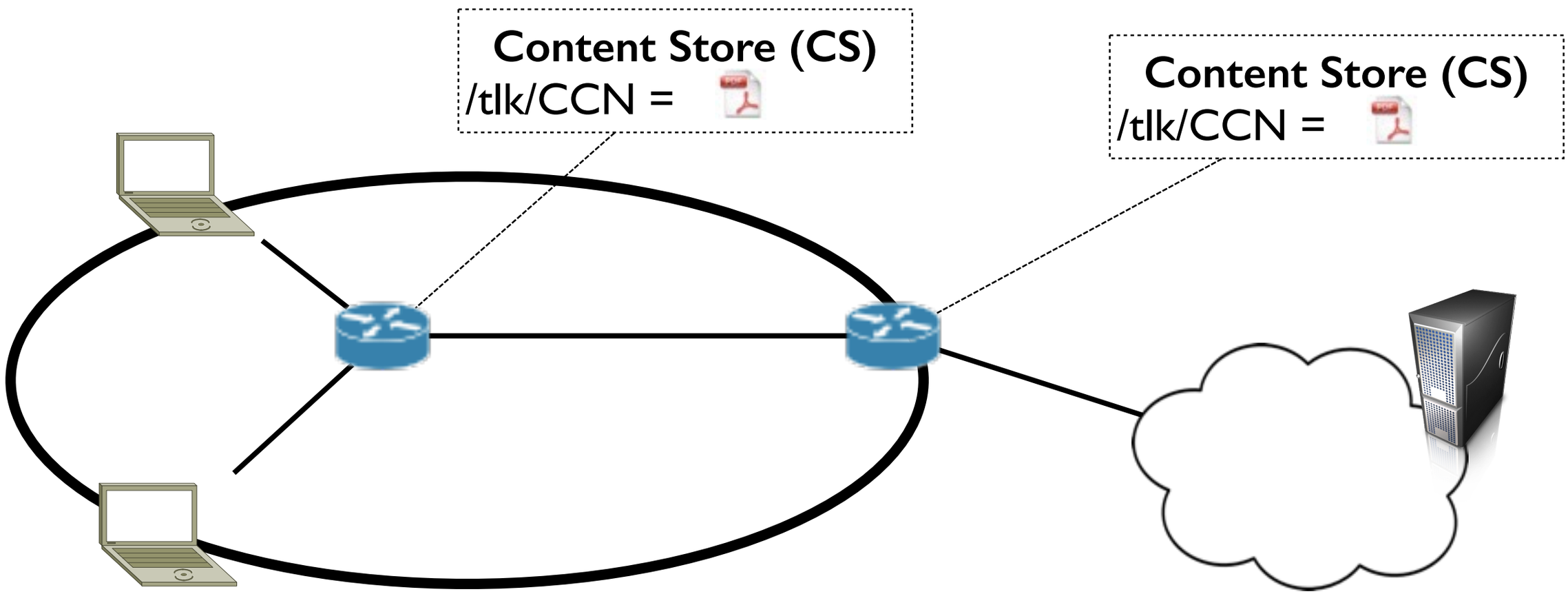
Contents can be cached anywhere (contd.)



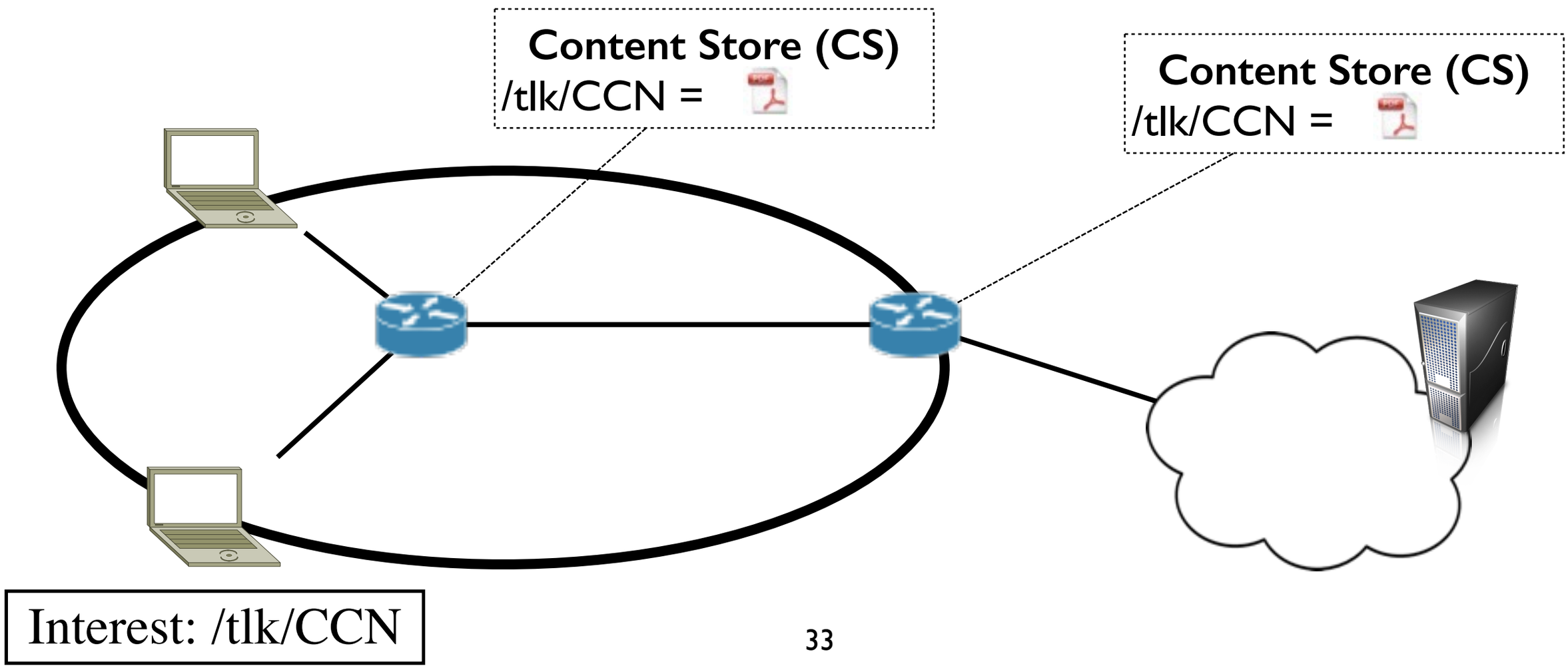
Contents can be cached anywhere (contd.)



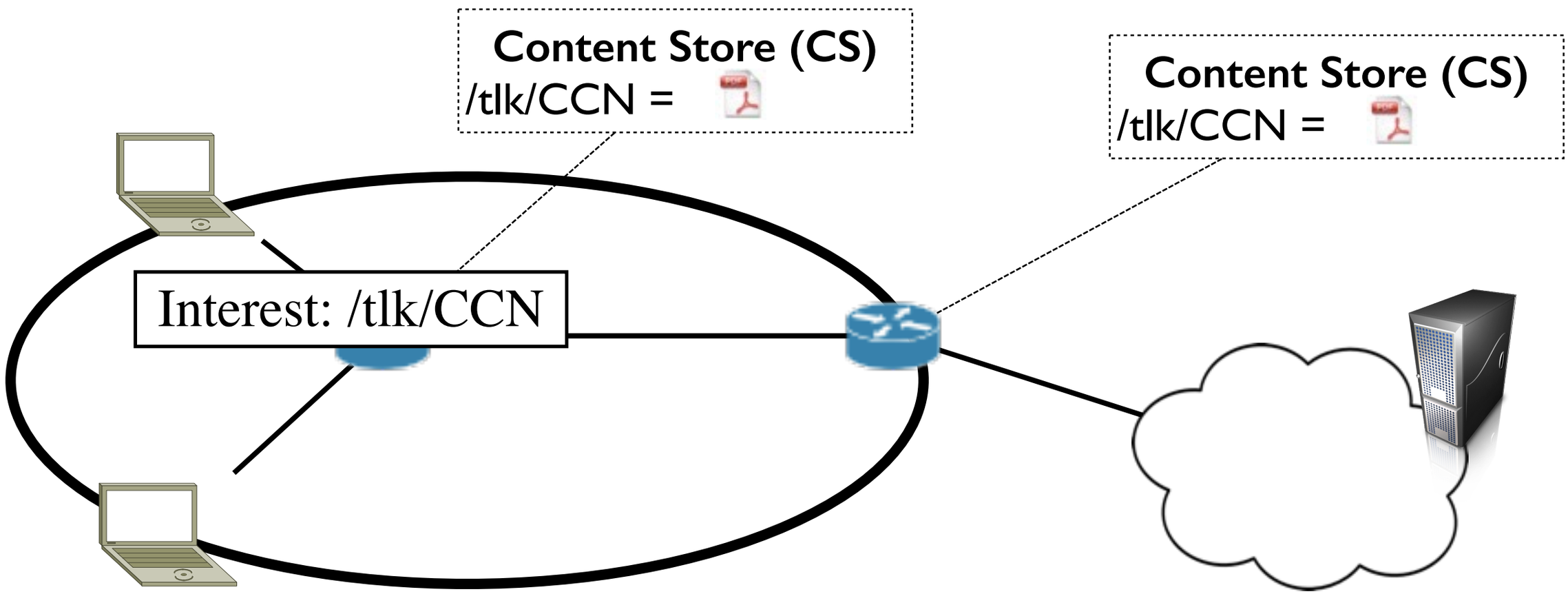
Contents can be cached anywhere (contd.)



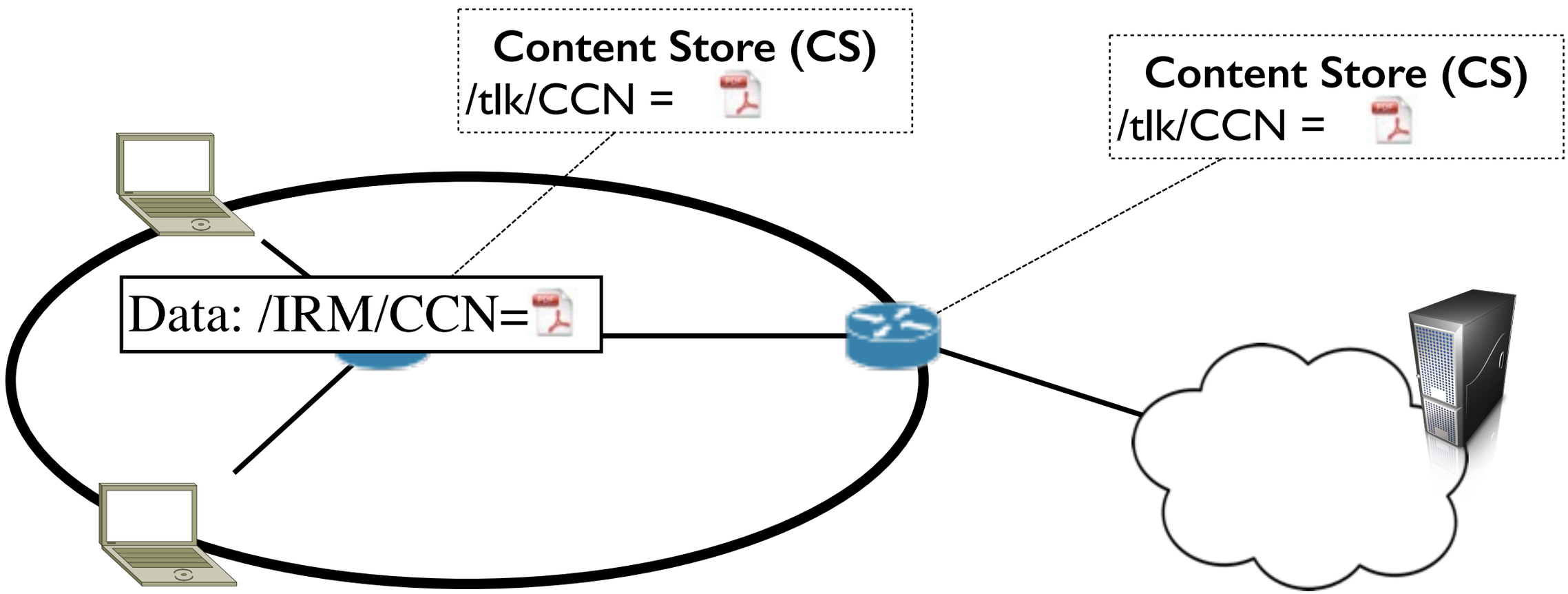
Contents can be cached anywhere (contd.)



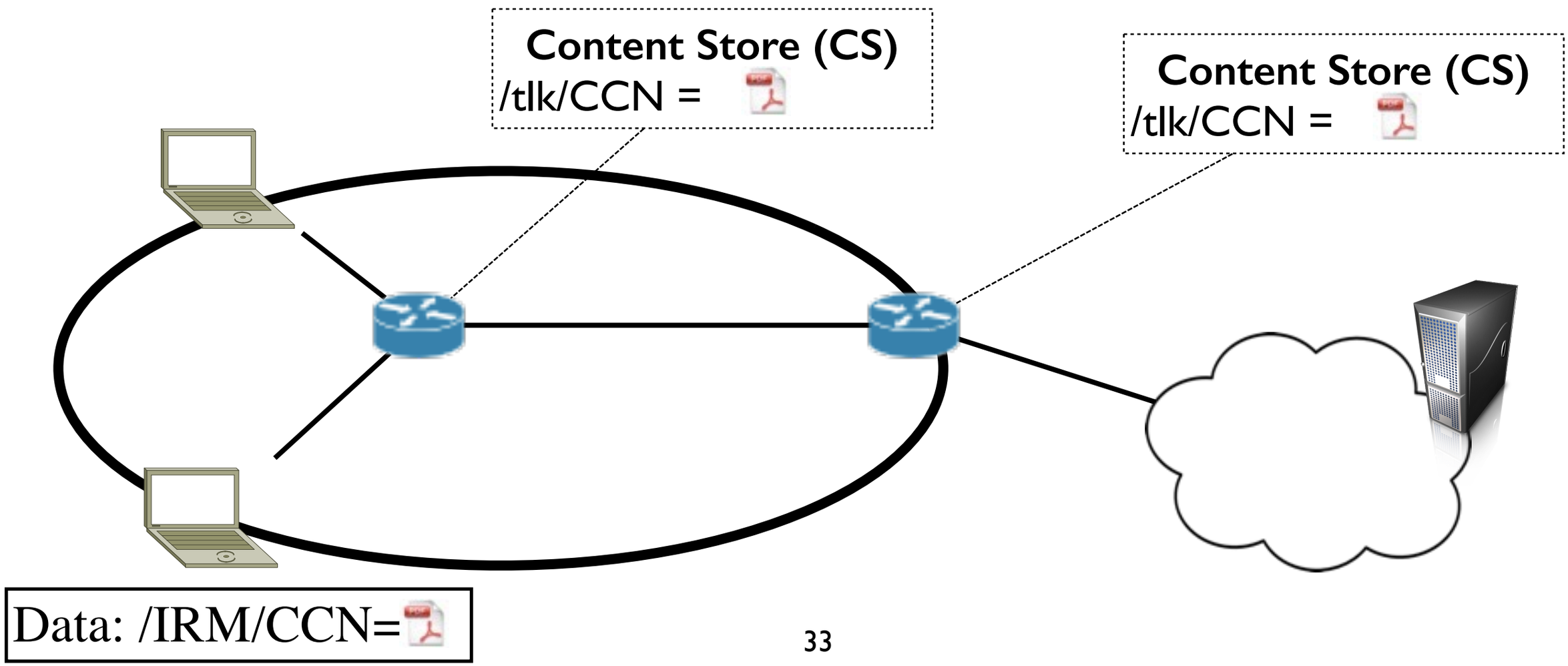
Contents can be cached anywhere (contd.)



Contents can be cached anywhere (contd.)

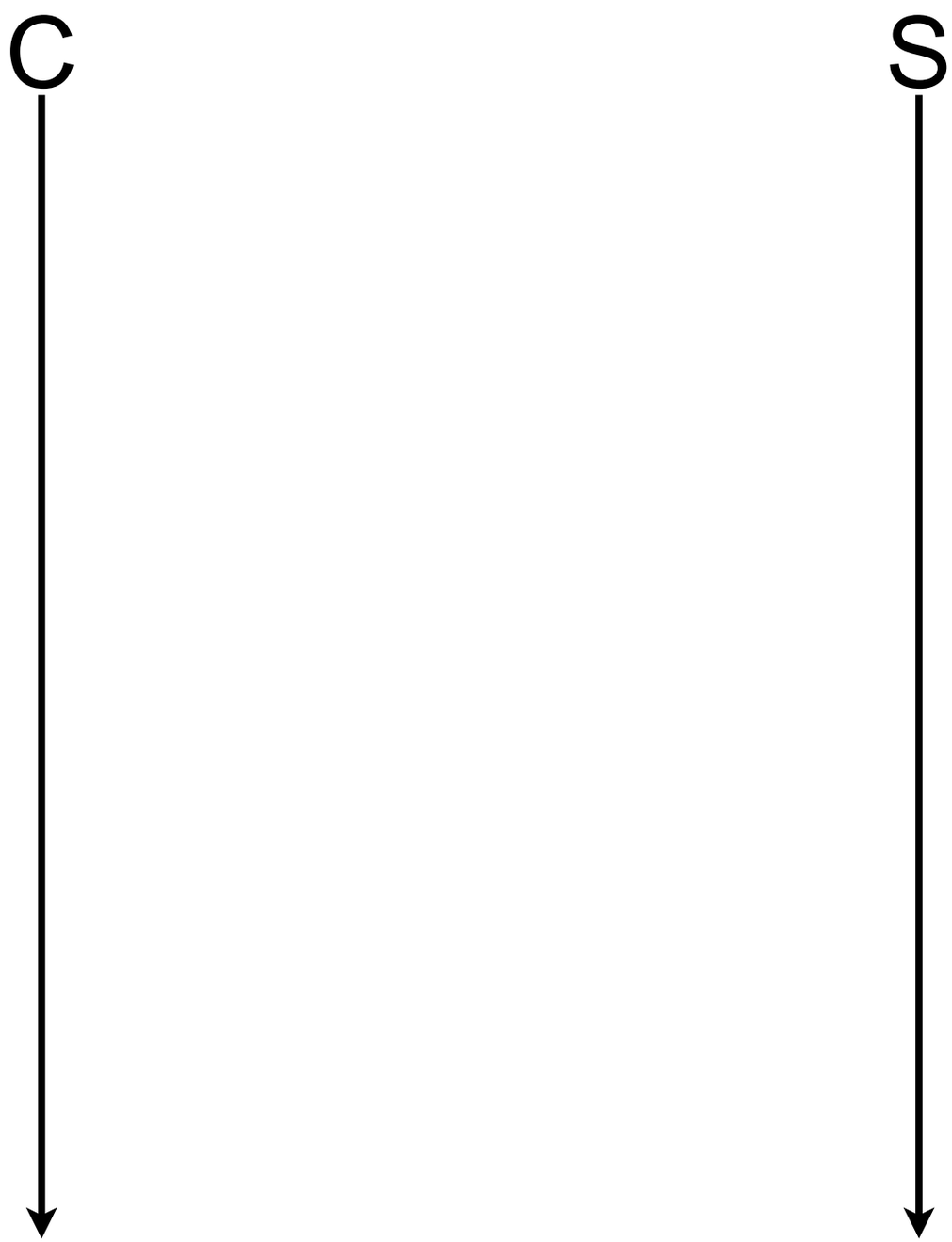


Contents can be cached anywhere (contd.)



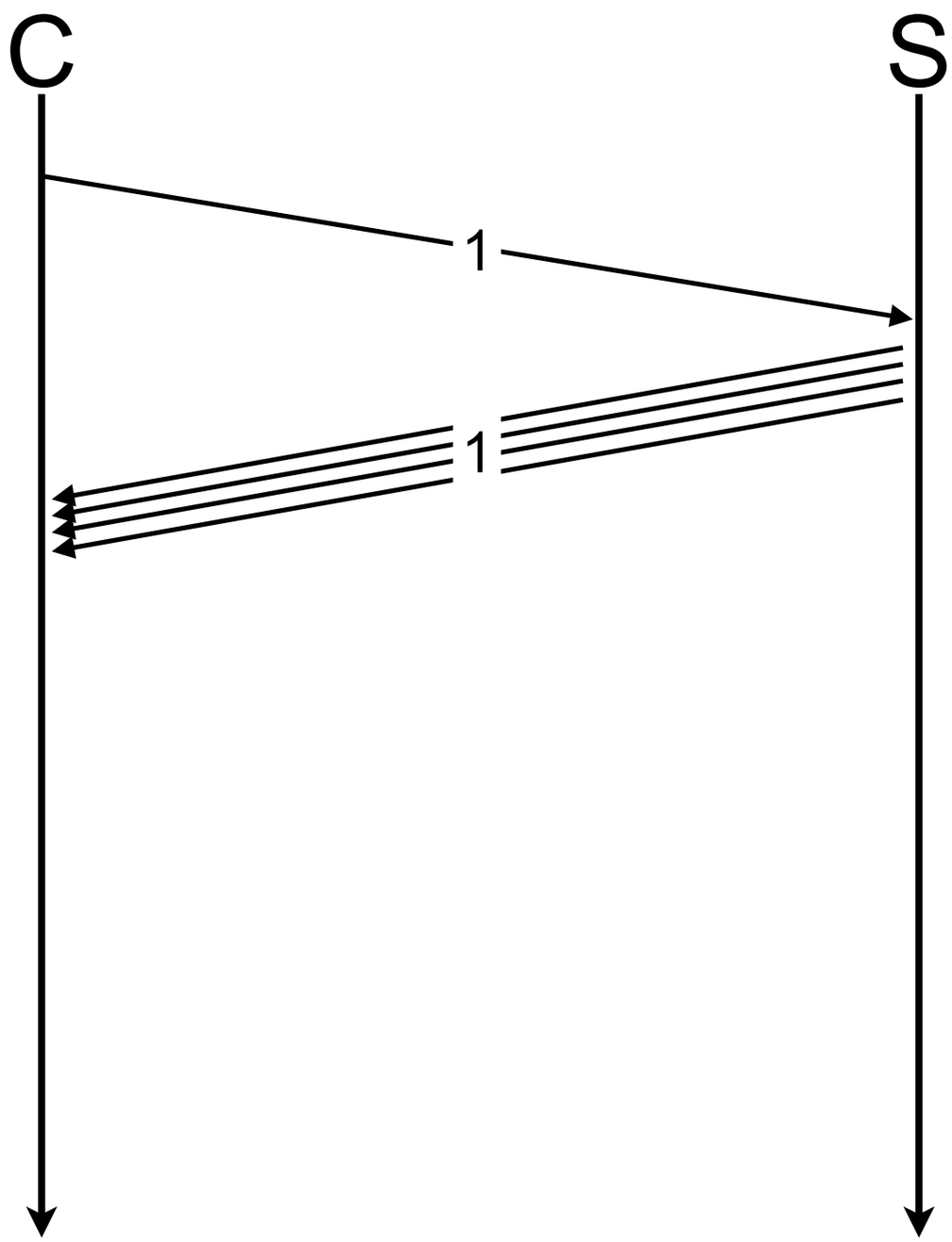
How to transport data?

Pipelining to speedup download time



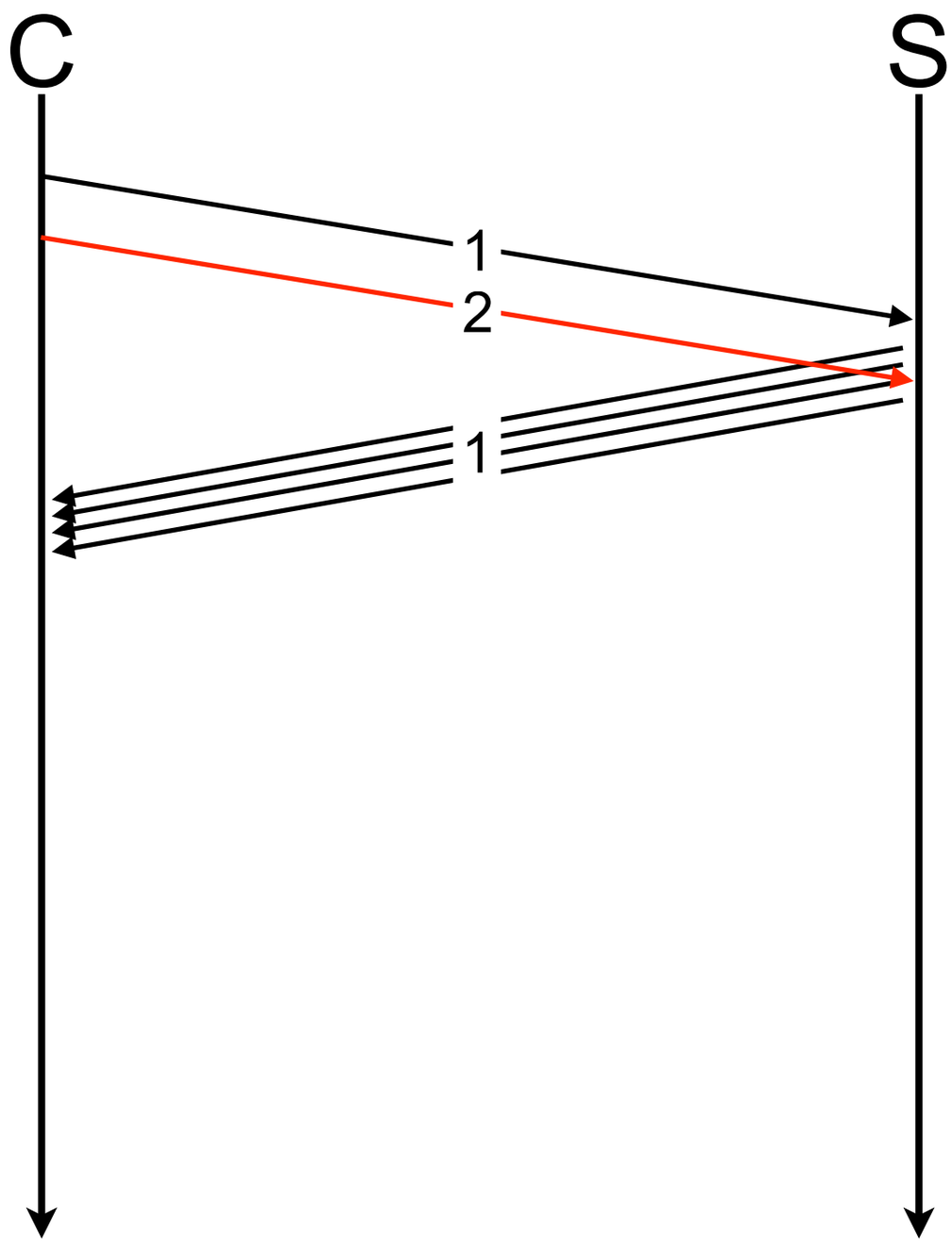
- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT
  - need to maintain a window of pending requests

Pipelining to speedup download time



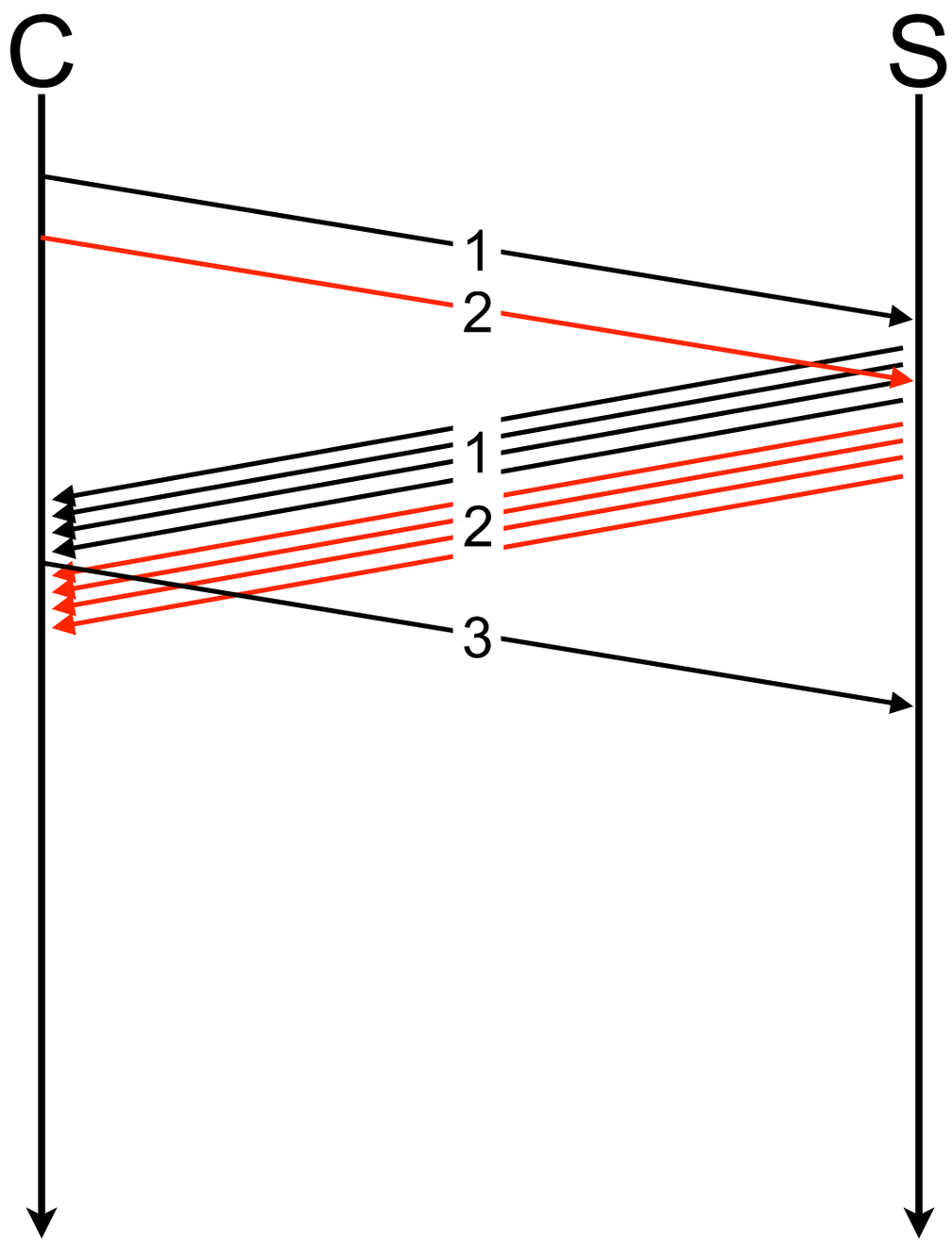
- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT
  - need to maintain a window of pending requests

Pipelining to speedup download time



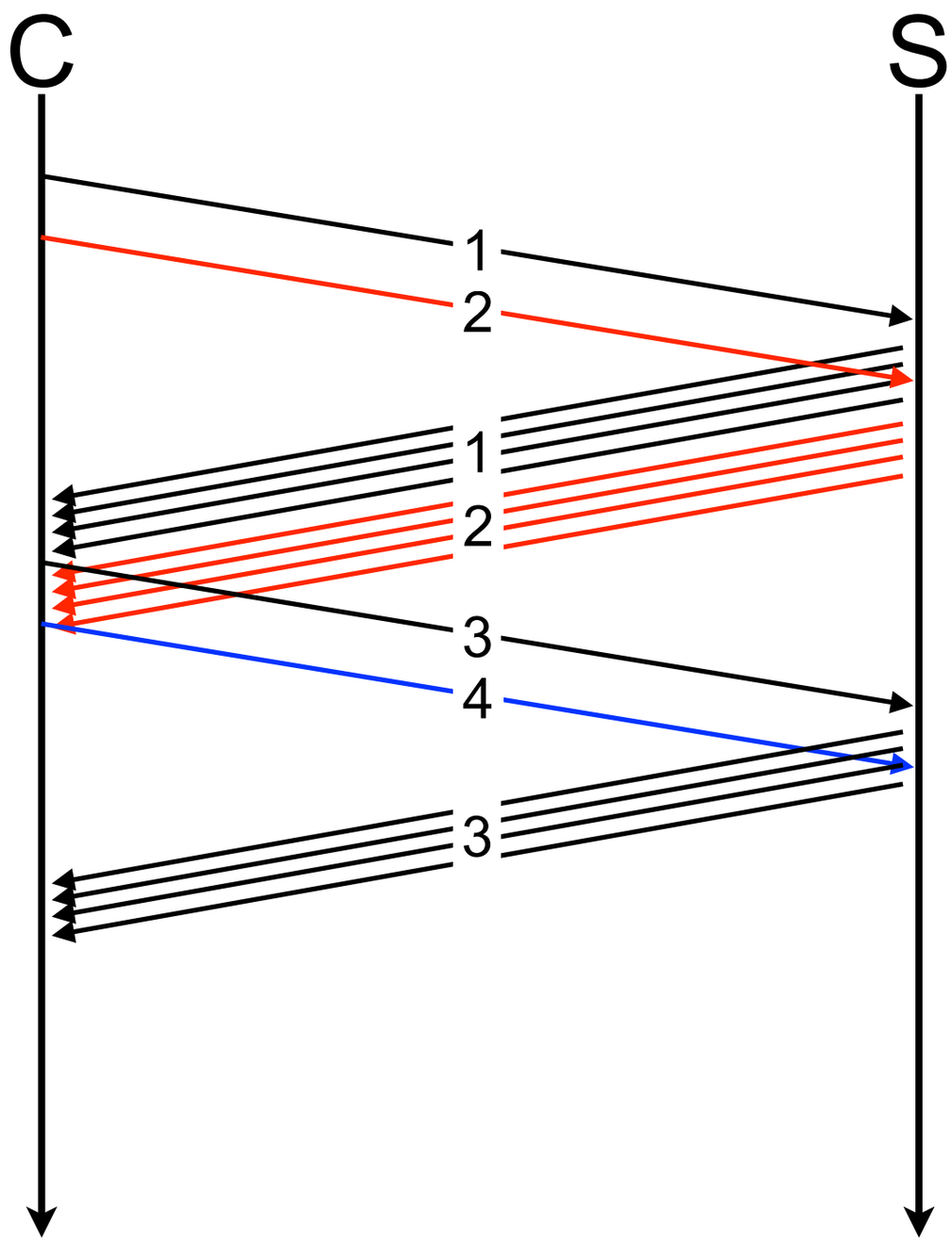
- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT
  - need to maintain a window of pending requests

Pipelining to speedup download time



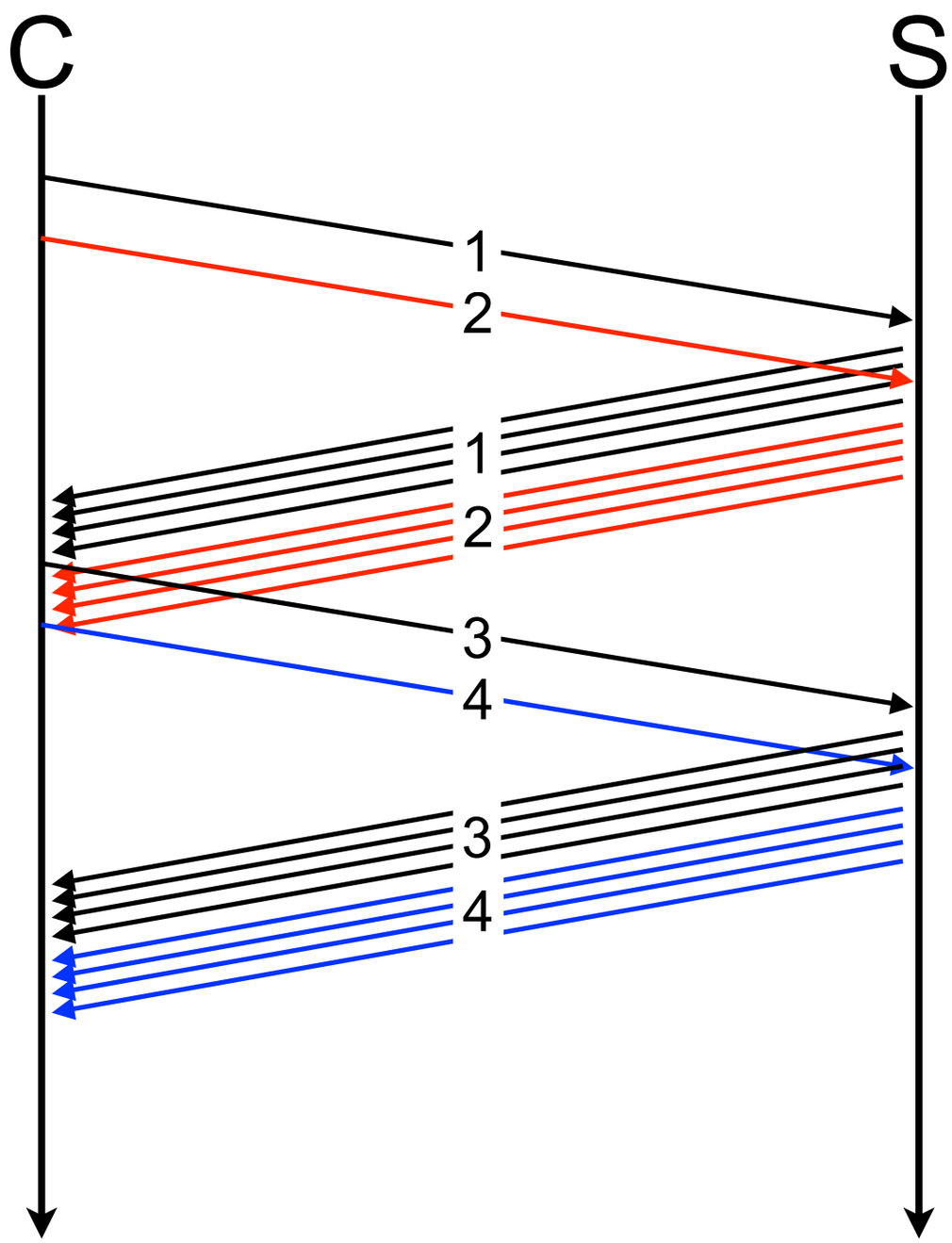
- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT
  - need to maintain a window of pending requests

Pipelining to speedup download time



- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT
  - need to maintain a window of pending requests

Pipelining to speedup download time



- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT
  - need to maintain a window of pending requests

What is AIMD?

- Additive-increase/multiplicative-decrease (AIMD) is the congestion avoidance algorithm of TCP
  - AIMD is used to dynamically determine the congestion window size for a TCP flow
    - `w(t+1) = w(t) + a` when no congestion is detected
    - `w(t+1) = w(t) * m` when congestion is detected

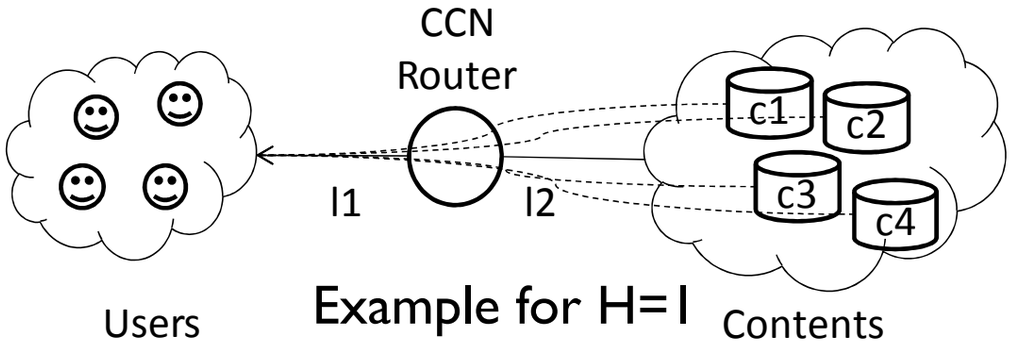
Is caching so transparent?

Is caching so transparent?

- Content-Centric Networking can massively rely on in-network on-path caching
  - most popular contents tend to be cached close to the consumers
  - least popular contents tend to be cached farther
- **What happens if AIMD is used in CCN?**
  - How does on-path caching impact the retrieval time?
  - How does it influence the fairness?
  - How does in-network on-path caching impact server load?

Hypotheses 1/2

- One consumer site initiates ALL the Interests
- One (other) site initiates ALL the Data packets (content producer)
- Consumer and producer sites are connected by a chain of LRU caches of length *H*
- Every link with delay *d*, total delay *H***d*



Example for H=1

Hypothesis 2/2

- Congestion is controlled by and only by the requester with "Additive Increase Multiplicative Decrease" (AIMD)
- Queueing delay is negligible
- Throughput for content c in AIMD given by

$$T(c) = \frac{K}{RTT(c)\sqrt{p(c)}}$$

Hypothesis 2/2

- Congestion is controlled by and only by the requester with "Additive Increase Multiplicative Decrease" (AIMD)
- Queueing delay is negligible
- Throughput for content c in AIMD given by

$$T(c) = \frac{K}{RTT(c)\sqrt{p(c)}}$$

How does in-network on-path caching impact the retrieval time?

- RTT for a content is the RTT to the first node that caches the content
- The average position is given by the hit rate $\omega_j(c)$ of nodes in the chain
- The average delay for *c* is then

$$RTT(c) = d \sum_{i=1}^{H} i\omega_i(c) \prod_{j<i} [1 - \omega_j(c)]$$

How does in-network on-path caching impact the retrieval time?

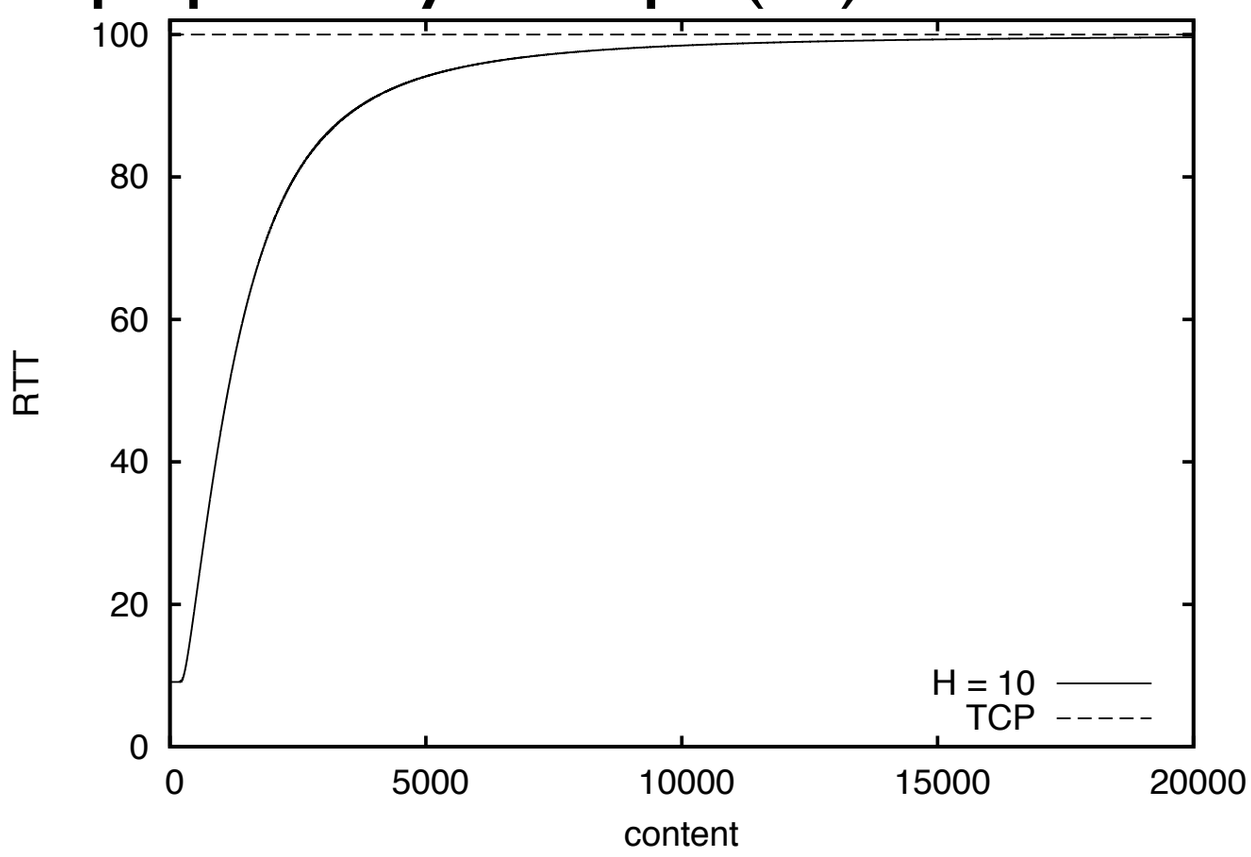
- RTT for a content is the RTT to the first node that caches the content
- The average position is given by the hit rate $\omega_j(c)$ of nodes in the chain
- The average delay for *c* is then

$$RTT(c) = d \sum_{i=1}^{H} i\omega_i(c) \prod_{j<i} [1 - \omega_j(c)]$$

Probability of reaching node *i*

Short RTT for popular contents

- 1,000,000 contents
- 10,000 caching entries total
- Content popularity ~ Zipf (α)



$\alpha = 2$

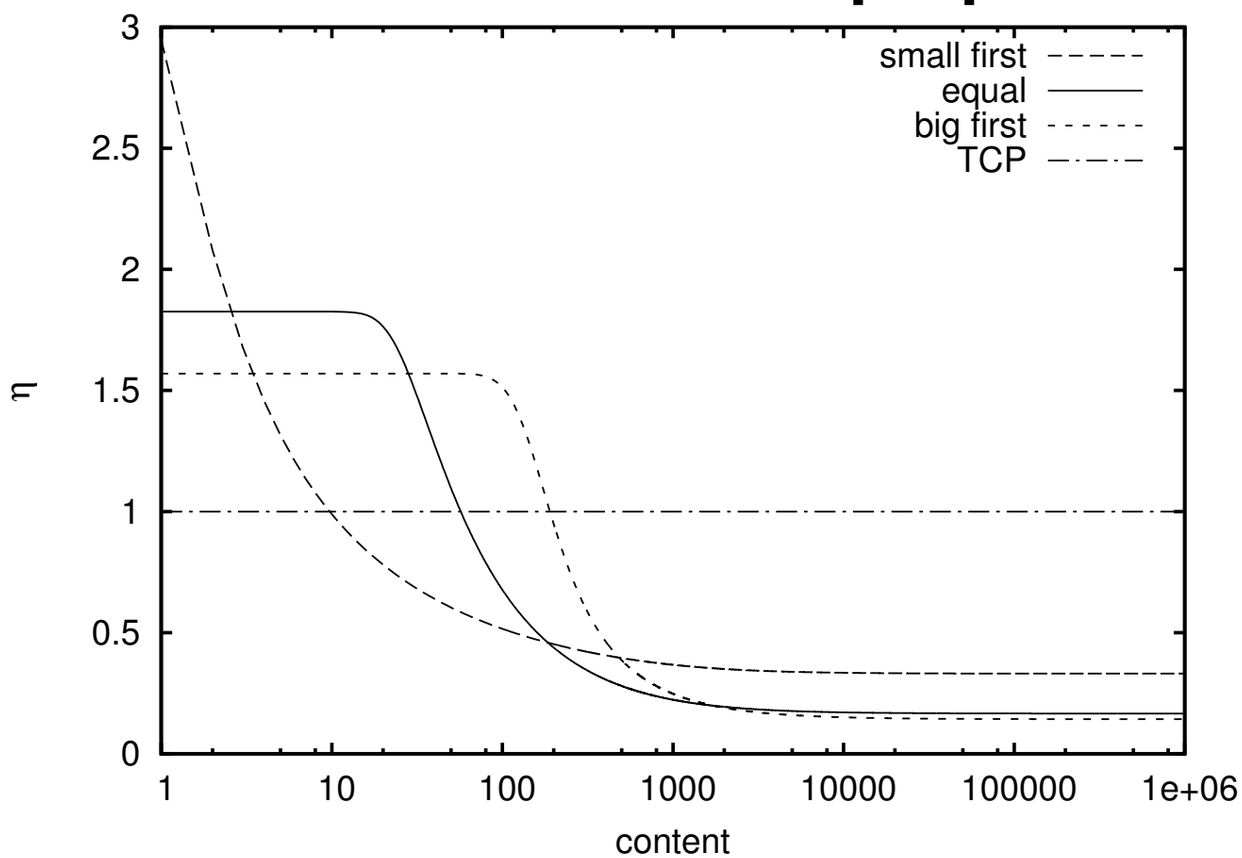
How does it influence the fairness?

- Baseline: TCP (i.e., no cache) throughput
- Is the throughput gain identical for every of the *N* downloads?
- Metric: ratio of throughput with and without cache, for any download *i*

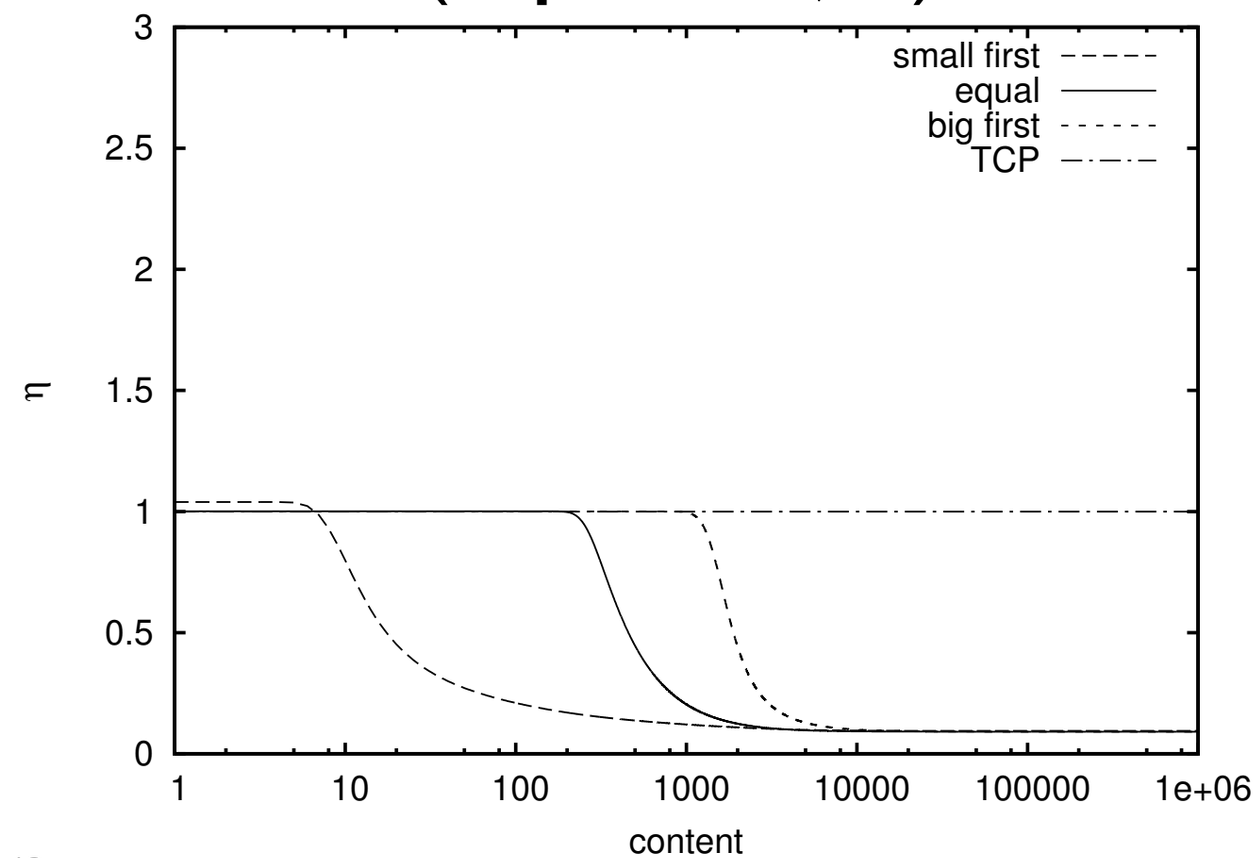
$$\eta(c_i) = \frac{T(c_i)}{\hat{T}(c_i)} \approx \frac{1/RTT(c_i)}{\frac{\sum_{j=1}^{N} 1/RTT(c_j)}{N}}$$

Negative impact for least popular contents

- Individually, very popular contents might not gain that much...
- ... but least populars lose a lot (up to $\frac{1}{H+1}$)



$\alpha = 1.1$



$\alpha = 2$

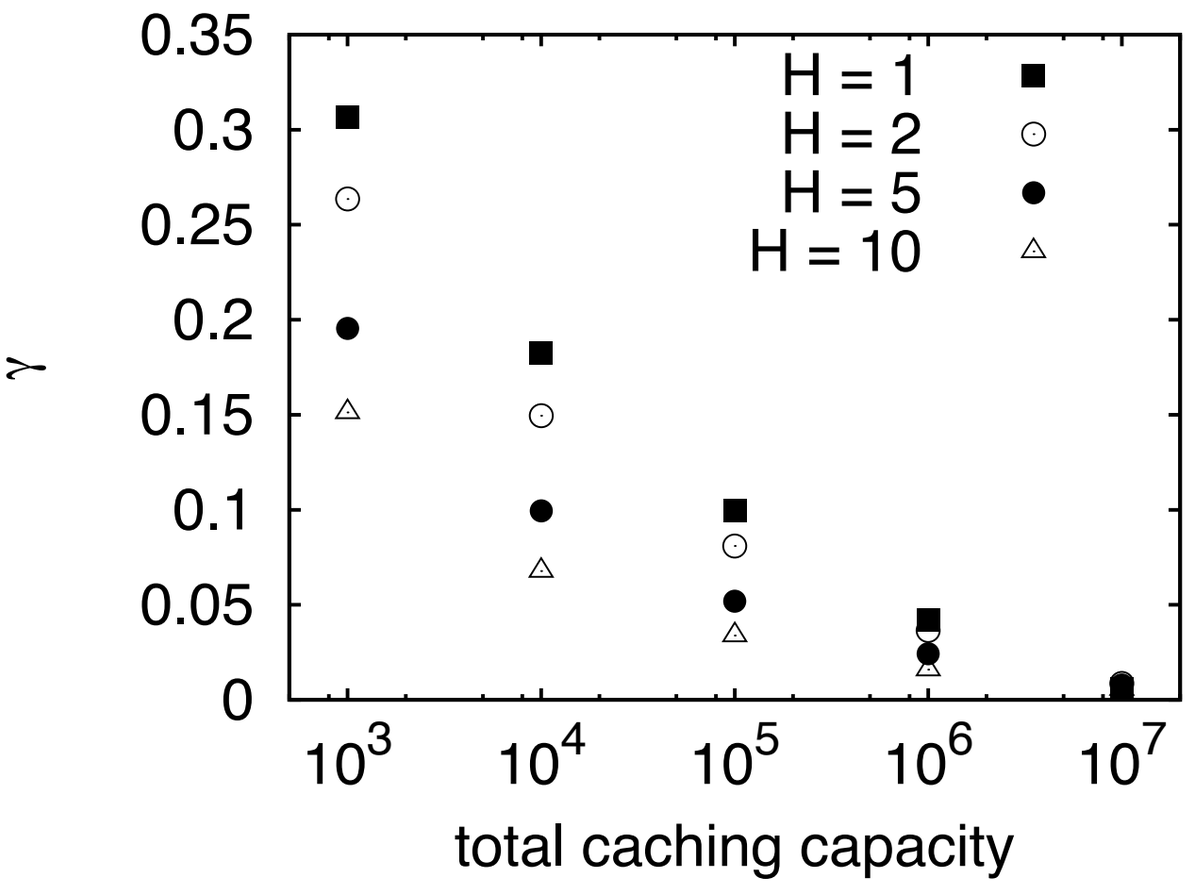
How does in-network on-path caching impact server load?

- Hypothesis: processing cost and data size is the same for every content
- Metric: ratio of server link usage with and without cache, for any content *c*

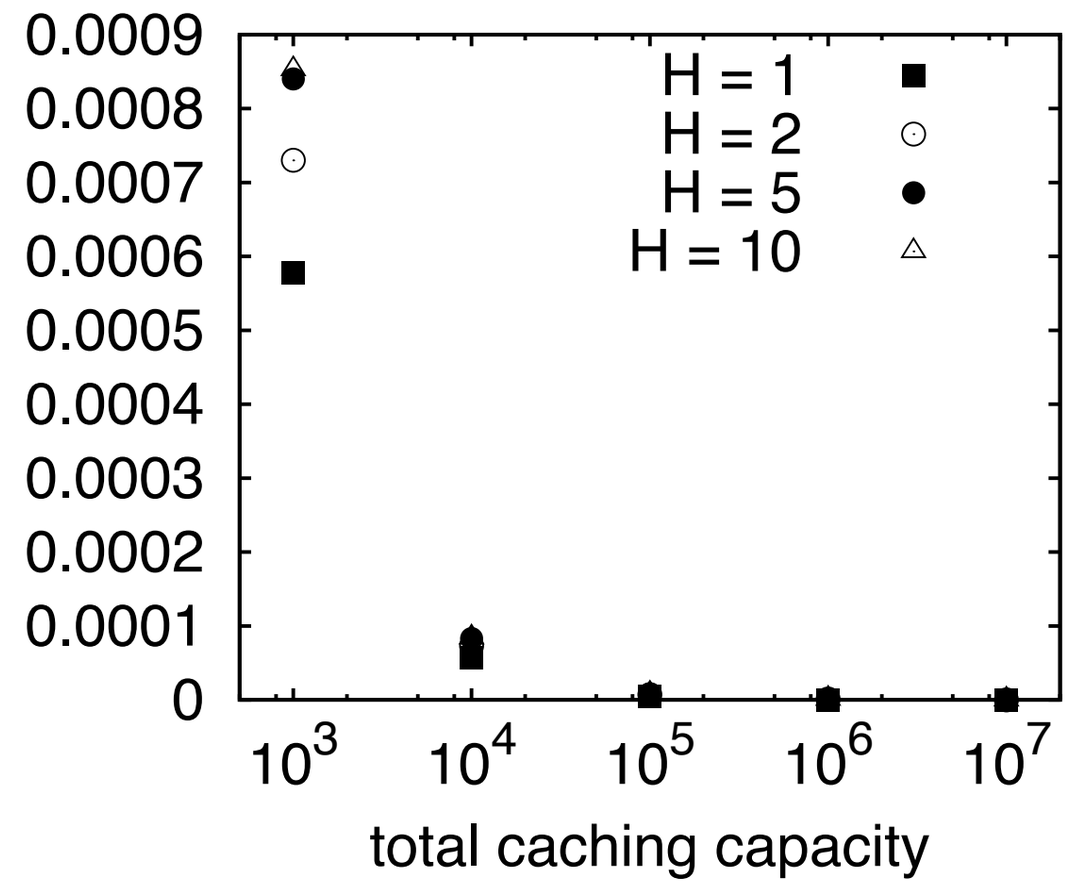
$$\gamma = \frac{\Lambda(l_{H+1})}{\hat{\Lambda}(l_{H+1})} = \frac{\sum_{i=1}^{N} \frac{\prod_{j=1}^{H}[1-\omega_j(c_i)]}{RTT(c_i)}}{\sum_{j=1}^{N} \frac{1}{RTT(c_j)}}$$

Server load is reduced

- Limited impact of the chain length on server load



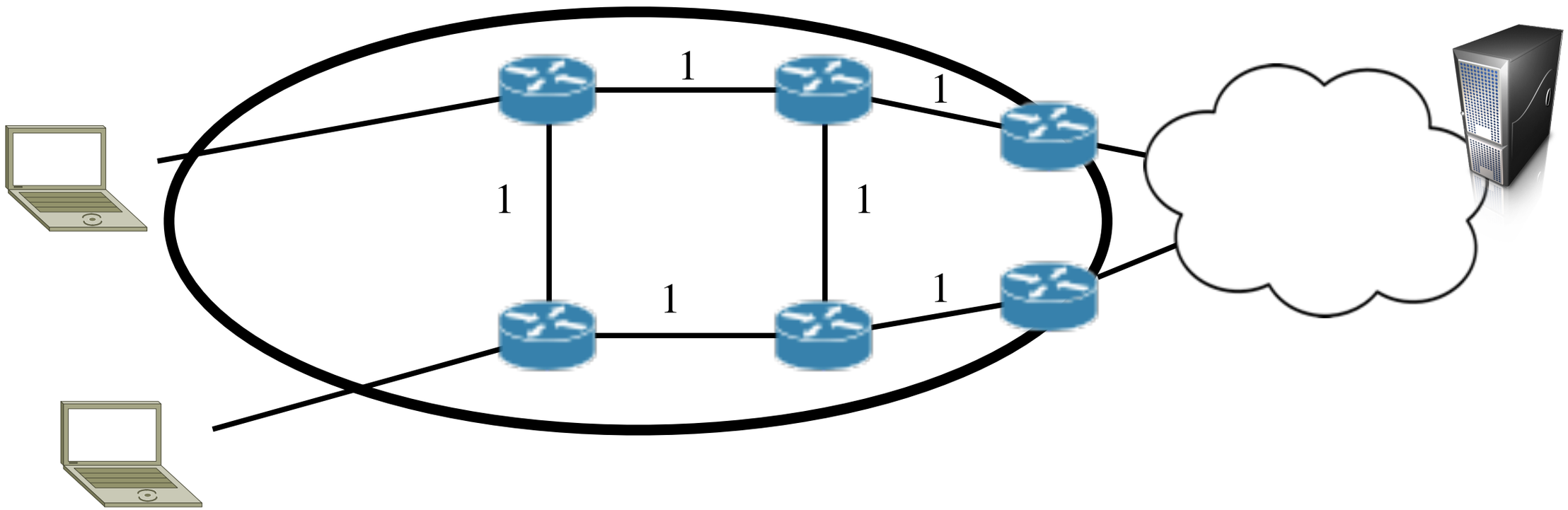
$\alpha = 1.1$



$\alpha = 2$

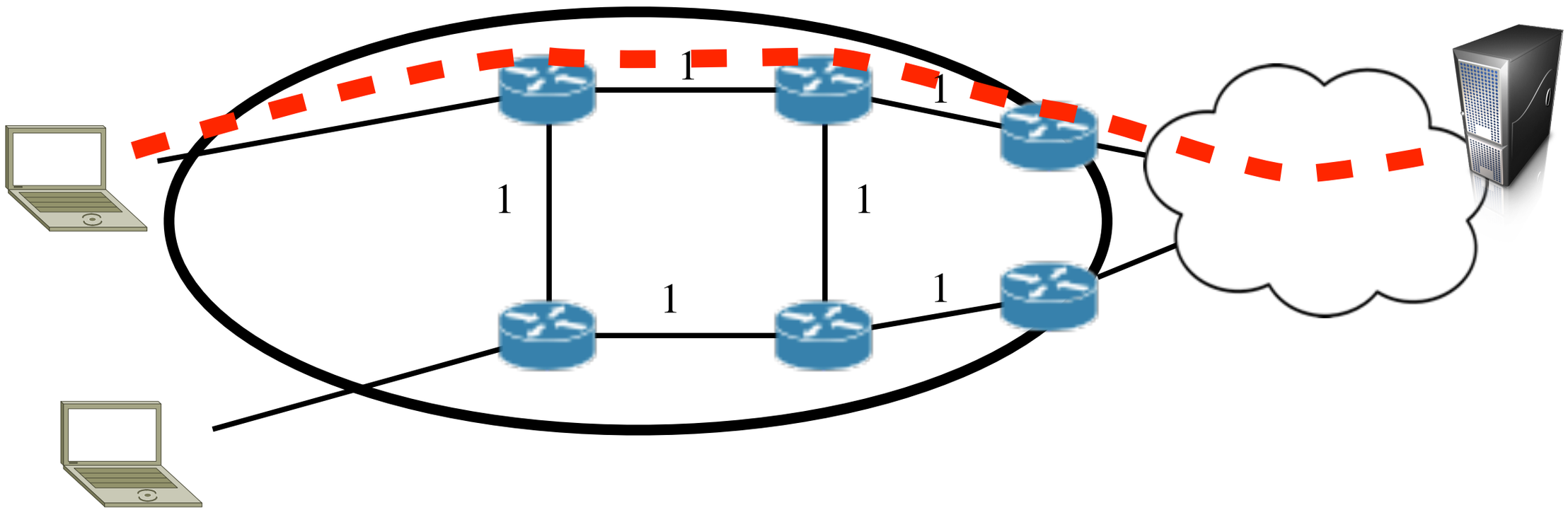
A dive into universal caching

Universal on-path caching is sub-optimal



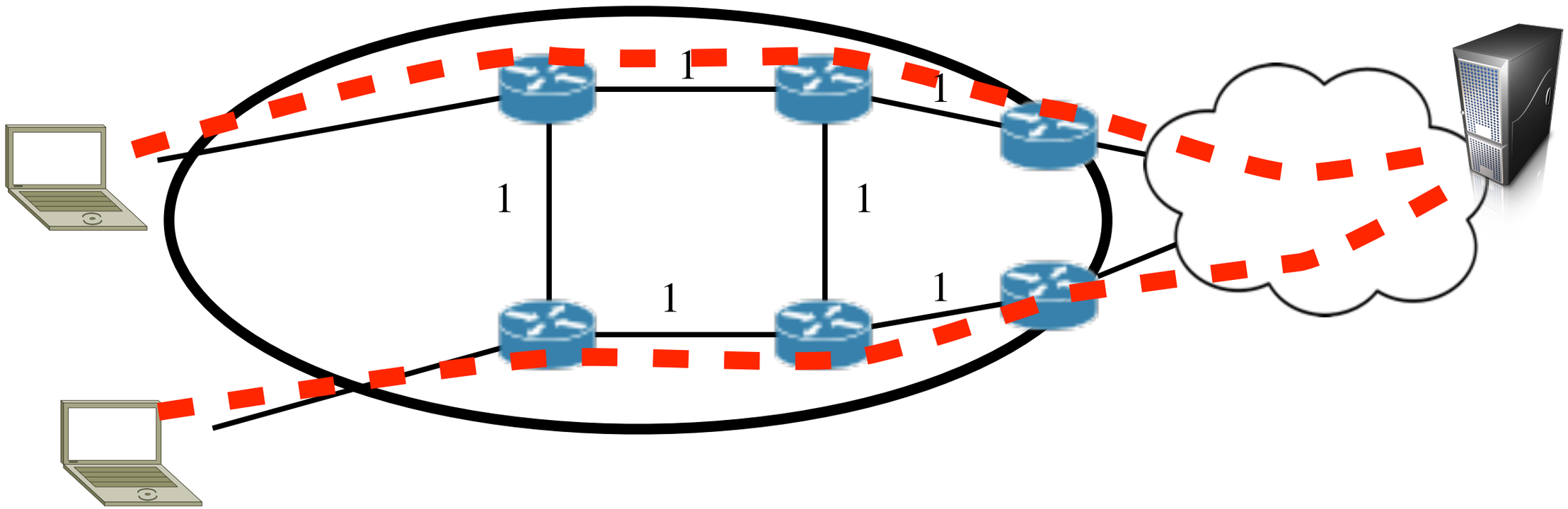
- /tlk/CCN

Universal on-path caching is sub-optimal



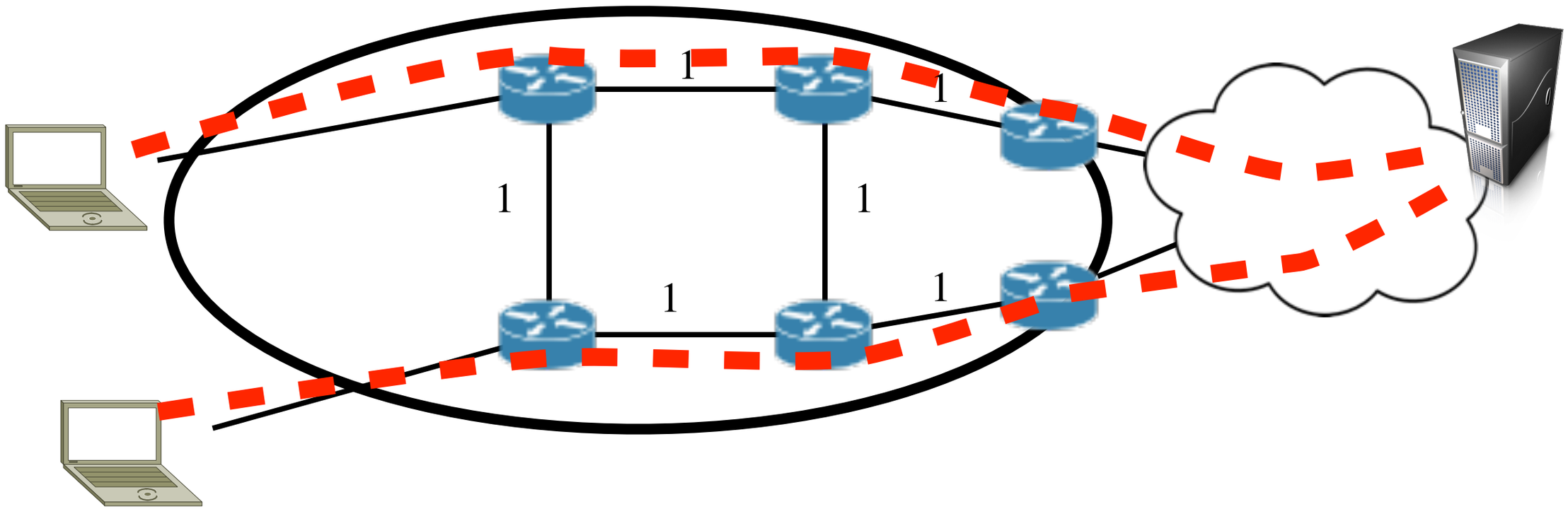
- /tlk/CCN

Universal on-path caching is sub-optimal



- /tlk/CCN

Universal on-path caching is sub-optimal



- Copies are as close as possible to consumers
  - but the network keeps up to 6 copies of the same chunk

/tlk/CCN

Universal on-path caching is sub-optimal (contd.)

- The amount of traffic on the external links of an AS that can cache *N* contents is minimized if the *N* most popular contents are cached
- On-path caching is sub-optimal as contents might be duplicated on different caches:
  - lower hit rates
  - higher delays

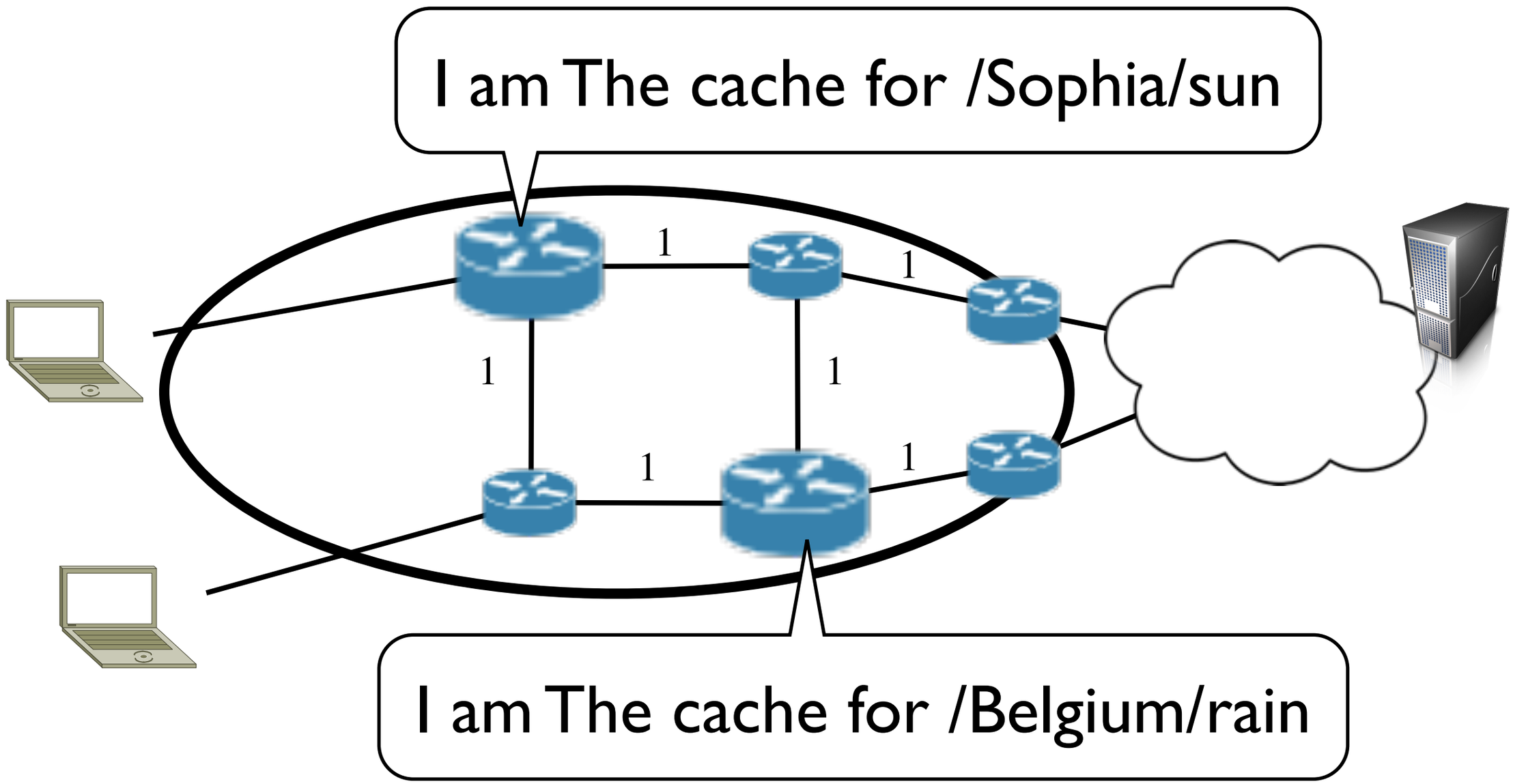
How to perform caching within an AS such that the use of external links is minimized while keeping the AS links' usage below their nominal capacities?

Deflect popular content traffic to optimally located caches

- To avoid content duplication on various caches, each popular content is assigned a specific cache
- A content is stored only on its assigned cached
- Every Interest packet for a given popular content is deflected to its content' assigned cache

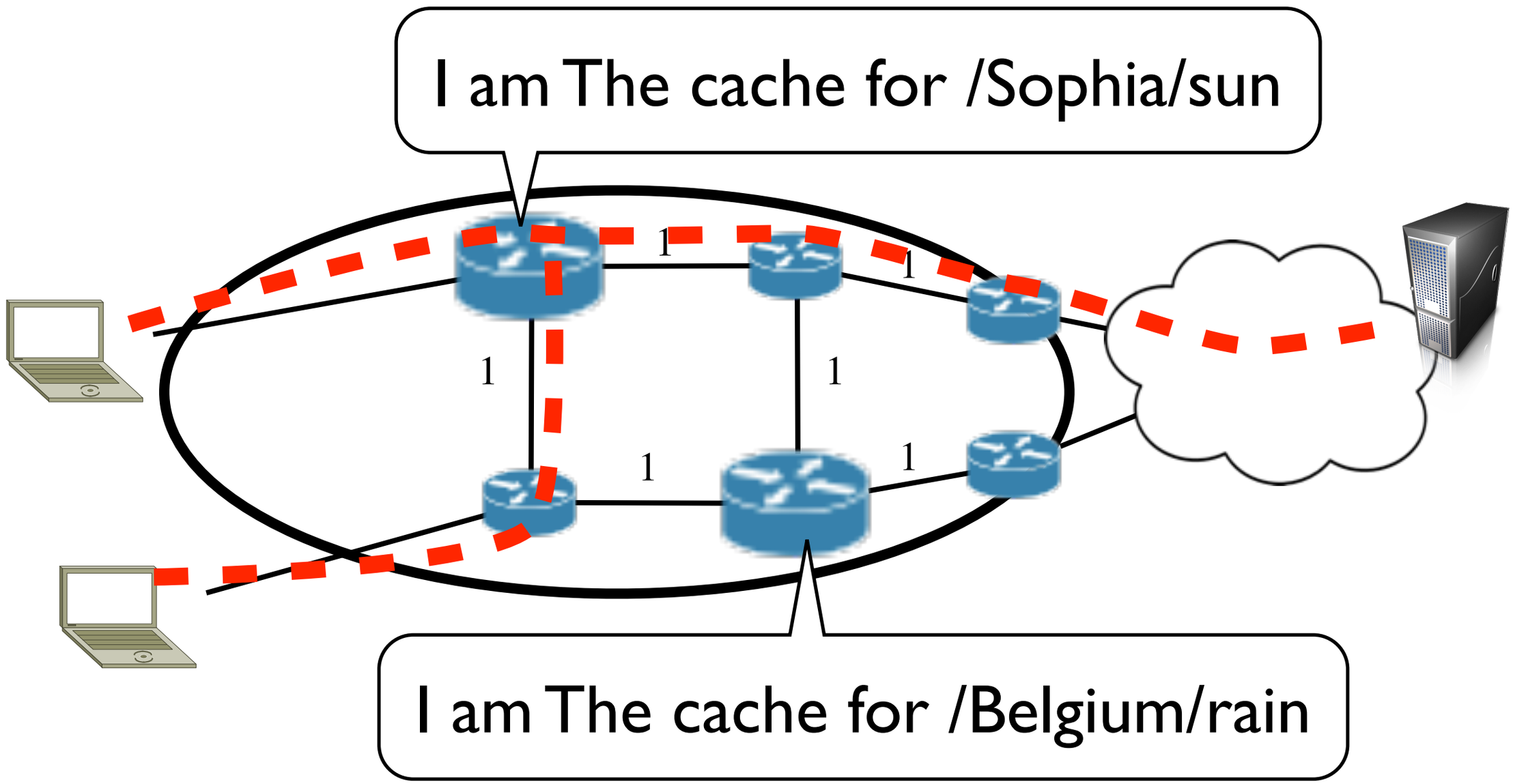
➡ As the shortest path is not followed anymore, it is called off-path caching

Off-path caching to achieve optimality



- /Sophia/sun

Off-path caching to achieve optimality



/Sophia/sun

Where to place contents?

- Ideal placement would be such that
  - contents are not duplicated,
  - popular contents are cached close (delay) to their consumers,
  - cache memory is not overloaded,
  - links are not overloaded.

Optimization problem

- Let A be the "*content (c) to cache (r)*" allocation matrix, with $\sum_{r \in \mathbf{R}} A_{r,c} = 1 \wedge A_{r,c} \in \{0, 1\}$

Optimization problem

- Let A be the "*content (c) to cache (r)*" allocation matrix, with $\sum_{r \in \mathbf{R}} A_{r,c} = 1 \wedge A_{r,c} \in \{0, 1\}$
- Minimize the delay due to deflection

$$\min \sum_{c \in \mathbf{C}} \sum_{e \in \mathbf{E}} \lambda_{c,e} \sum_{r \in \mathbf{R}} A_{r,c} \cdot d_{e,r}$$

Optimization problem

- Let A be the "*content (c) to cache (r)*" allocation matrix, with $\sum_{r \in \mathbf{R}} A_{r,c} = 1 \wedge A_{r,c} \in \{0, 1\}$
- Minimize the delay due to deflection

$$\min \sum_{c \in \mathbf{C}} \sum_{e \in \mathbf{E}} \lambda_{c,e} \sum_{r \in \mathbf{R}} A_{r,c} \cdot d_{e,r}$$

- Do not overload cache memory

$$\sum_{c \in \mathbf{C}} A_{r,c} \leq \text{memory}_r, \qquad \forall r \in \mathbf{R}$$

Optimization problem

- Let A be the "*content (c) to cache (r)*" allocation matrix, with $\sum_{r \in \mathbf{R}} A_{r,c} = 1 \wedge A_{r,c} \in \{0, 1\}$
- Minimize the delay due to deflection

$$\min \sum_{c \in \mathbf{C}} \sum_{e \in \mathbf{E}} \lambda_{c,e} \sum_{r \in \mathbf{R}} A_{r,c} \cdot d_{e,r}$$

- Do not overload cache memory

$$\sum_{c \in \mathbf{C}} A_{r,c} \leq \text{memory}_r, \qquad \forall r \in \mathbf{R}$$

- Do not overload links

$$\sum_{c \in \mathbf{C}^+} \sum_{e \in \mathbf{E}} \lambda_{c,e} \cdot \delta_{l,e,\text{egress}_{c,e}} \leq c_l, \qquad \forall \text{ link } l$$

Optimal content placement and deflection

- The popularity $\lambda_{c,e}$, for every content c, at every edge router e can be determined (e.g., CS, NetFlow, FlowVisor)
- A is constructed by solving the optimization problem
- A is used by the routing system to construct path such that Interest packets are deflected to the appropriate caches

Optimality is complex to achieve

- Optimal placement is complex to compute
  - requires content popularity estimation
  - hard to maintain the optimization problem tractable in some configuration (e.g., large network, large caches...)
  - not adapted to dynamic popularity distribution
- Flow table size is $\mathcal{O}(N)$ with *N* potentially large

Hash function based heuristic

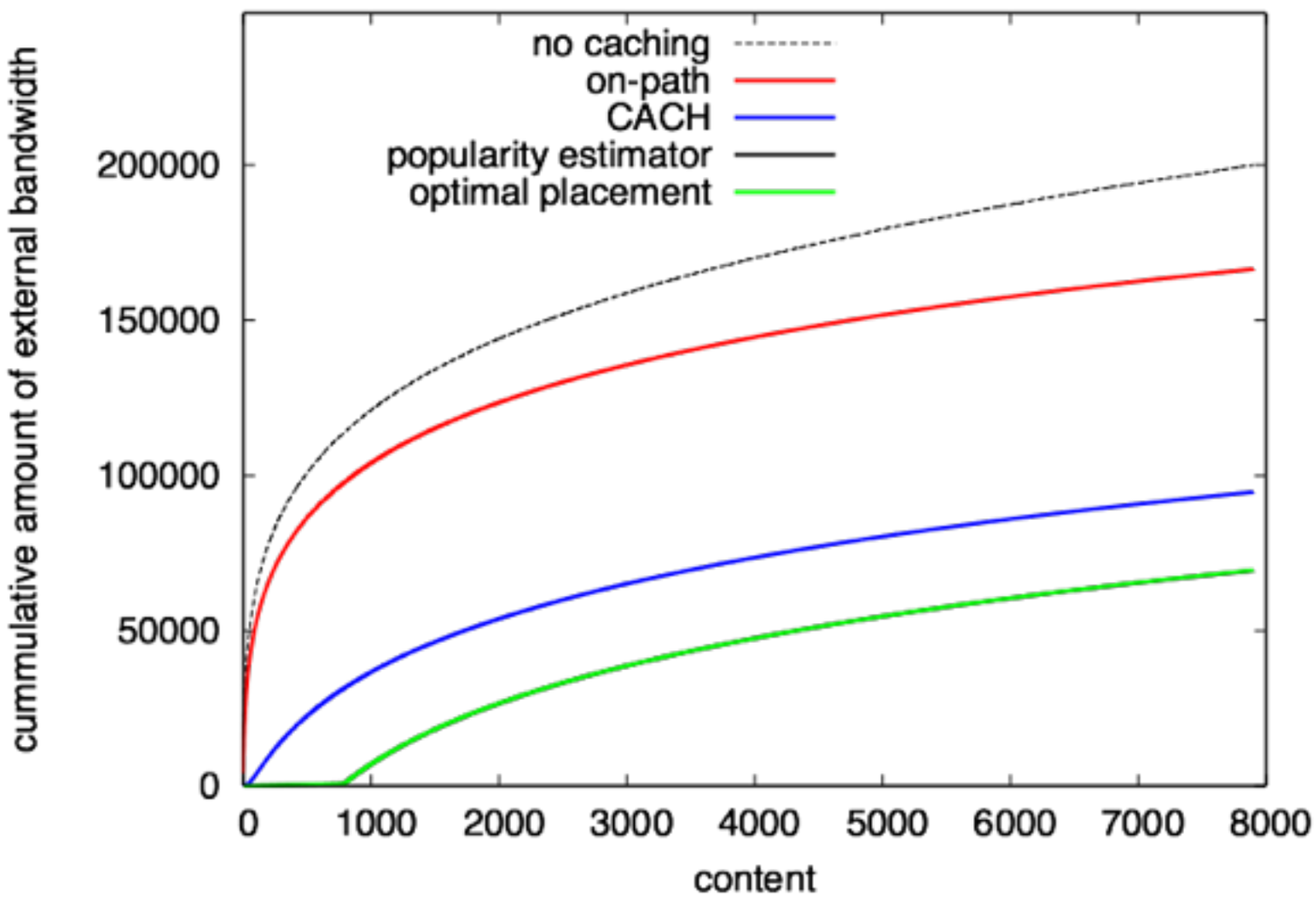
- A heuristic that avoids content duplication, removes the necessity to solve an optimization problem, and maintains flow table size linear with the size of the network $\mathcal{O}(|R|)$
- Caching All Contents by Hashing (CACH) heuristic
  - each edge router maintains the list of routers with caching capability $\mathcal{O}(|R|)$
  - for every Interest packet, hash its content name
  - deflect the Interest packet to the cache pointed out by the hash value

Evaluation

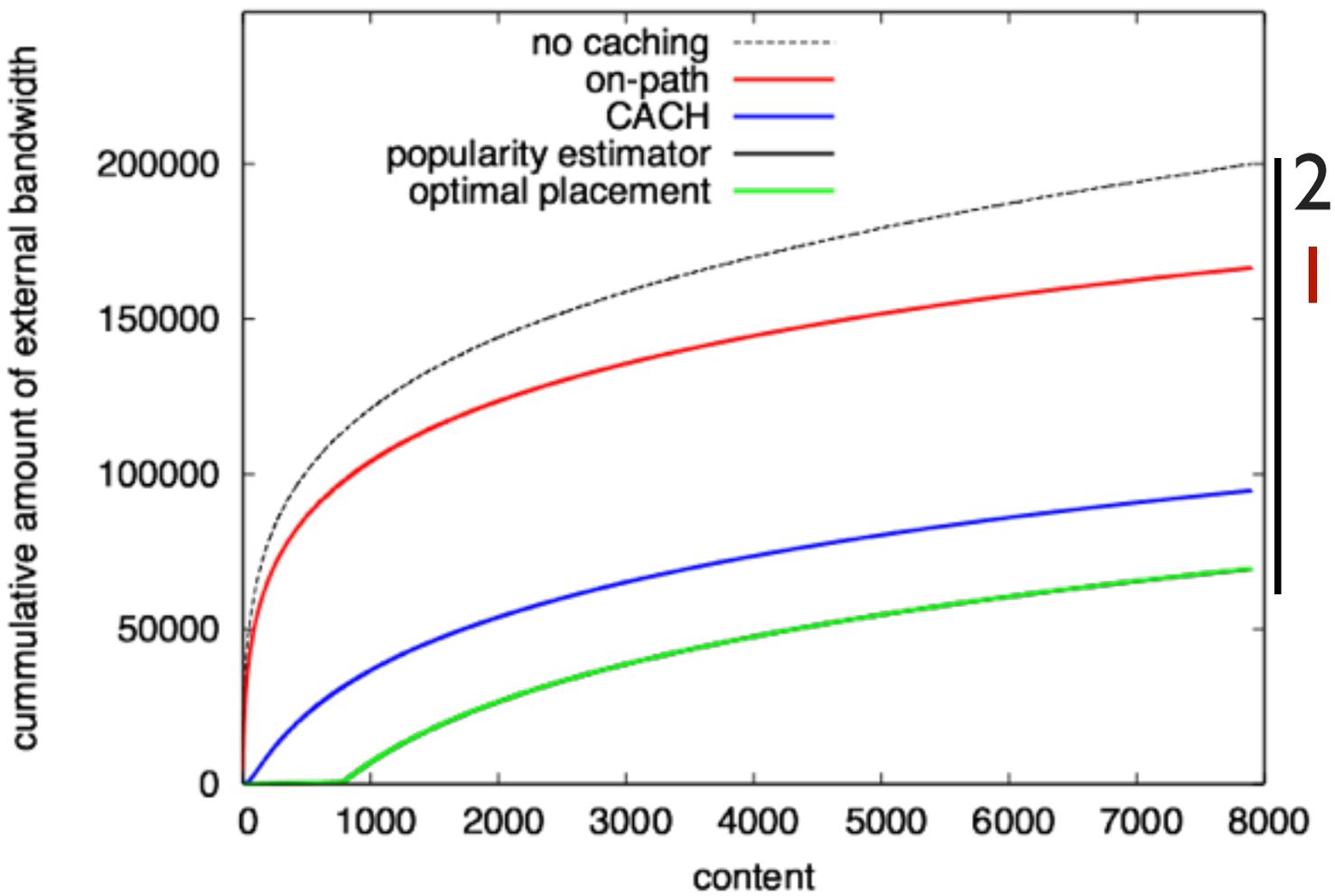
Simulation Setup

- Rocketfuel [SMW02] topology ASN 3967
  - 79 core routers, 44 edge routers (2 per city), 6 peering routers
- LRU caching on edge routers, 10 cache entries per core router
- 150ms peering link delay
- 200,000 Interest packets generated, simulations repeated 11 times
- 7,900 content of Zipf 0.8 [FRR12] popularity distribution

Peering Link Bandwidth Gain



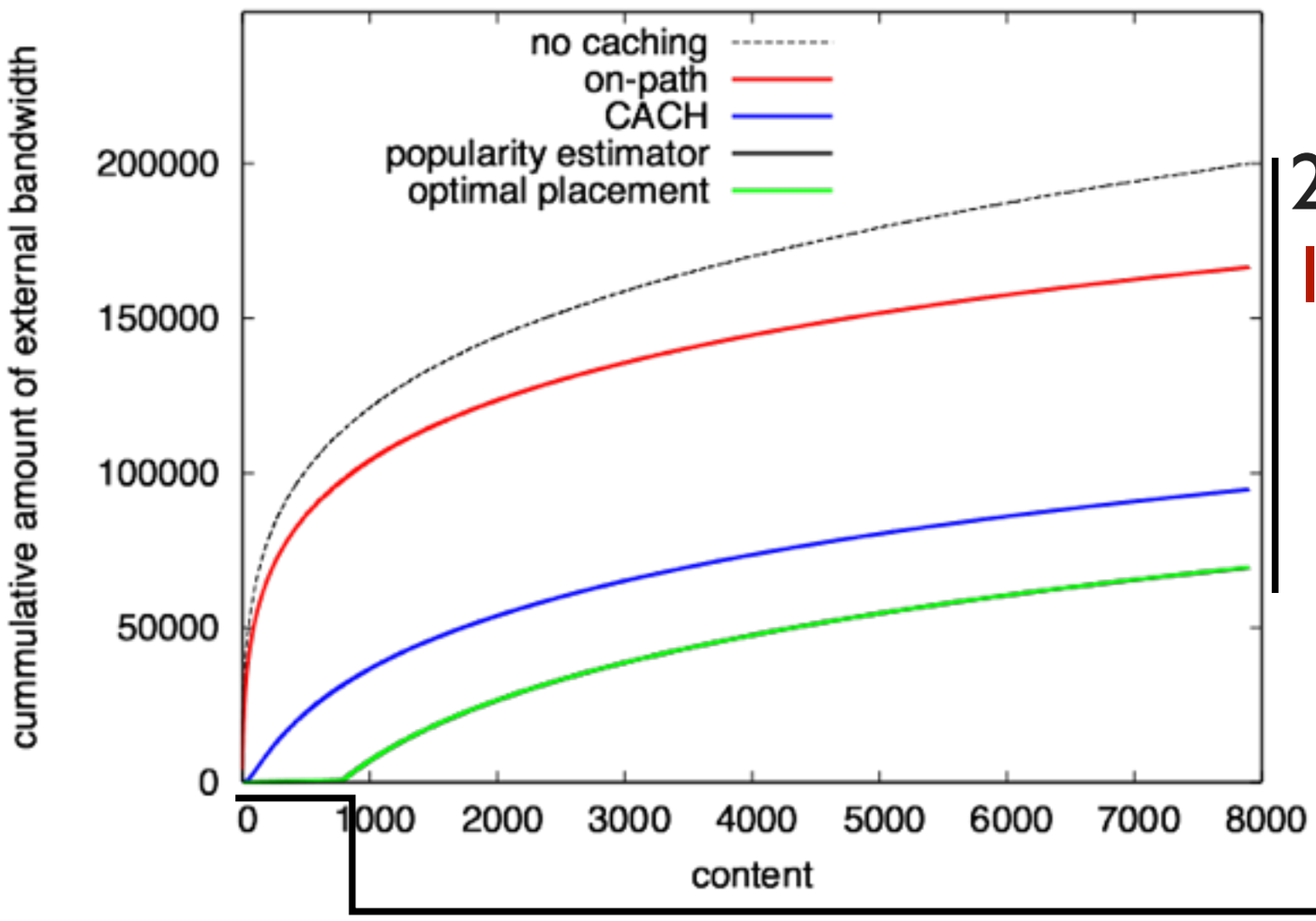
Peering Link Bandwidth Gain



200,000
166,479
94,657
69,509

Peering traffic drops from 83% of the total traffic to 47% and 35%.

Peering Link Bandwidth Gain

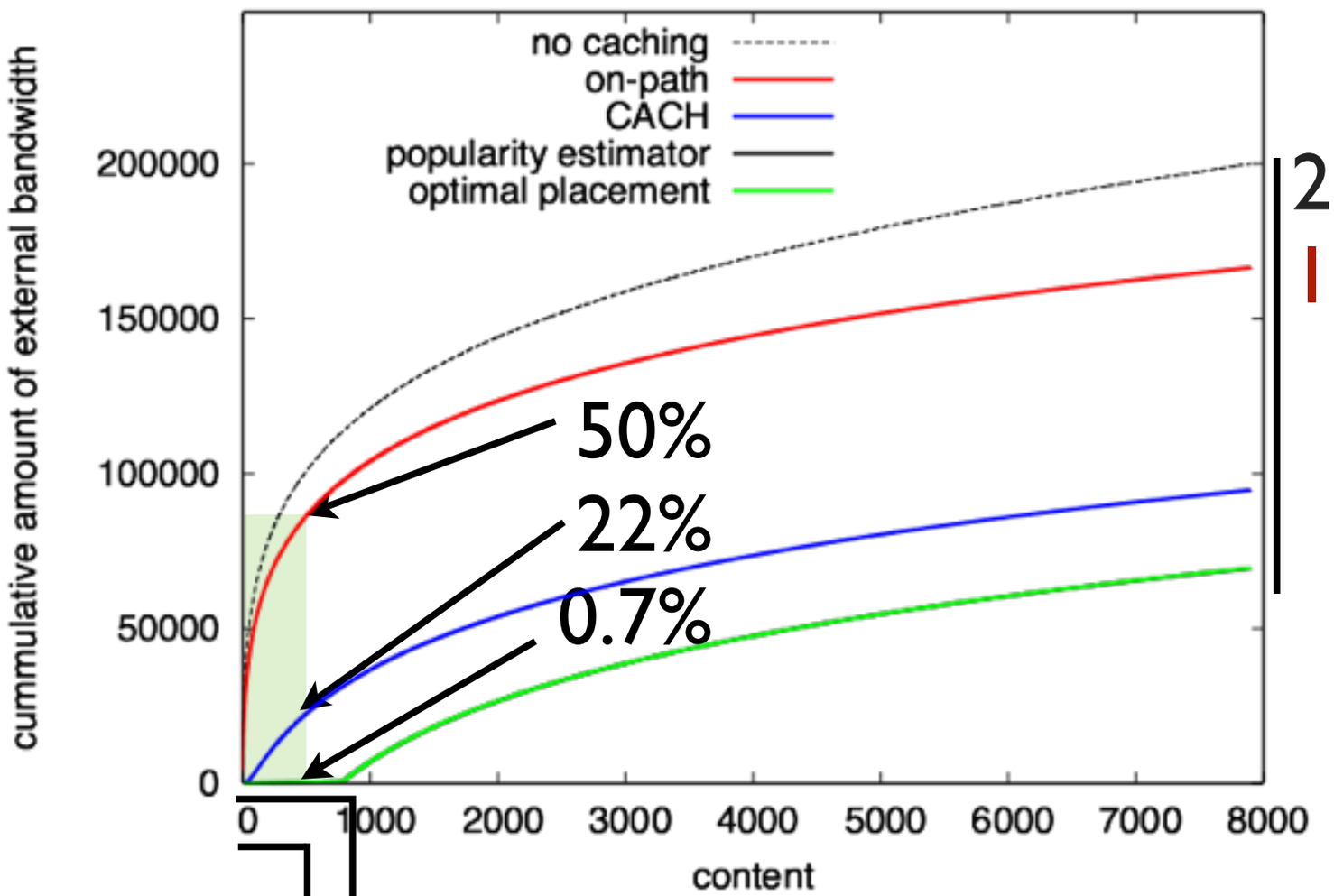


200,000
166,479
94,657
69,509

Peering traffic drops from 83% of the total traffic to 47% and 35%.

Popular contents are always cached

Peering Link Bandwidth Gain



200,000
166,479
94,657
69,509

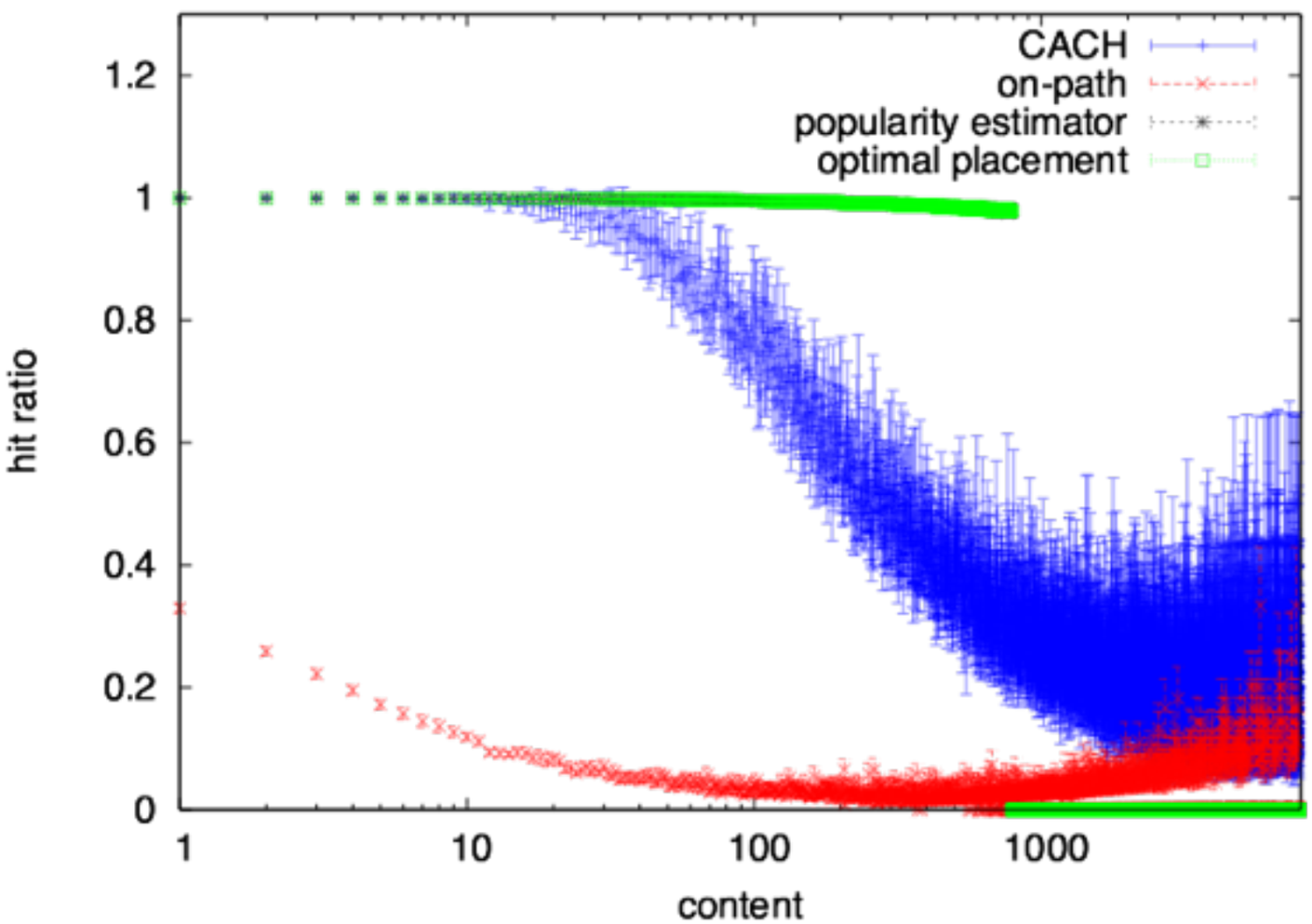
Peering traffic drops from 83% of the total traffic to 47% and 35%.

Popular contents are always cached

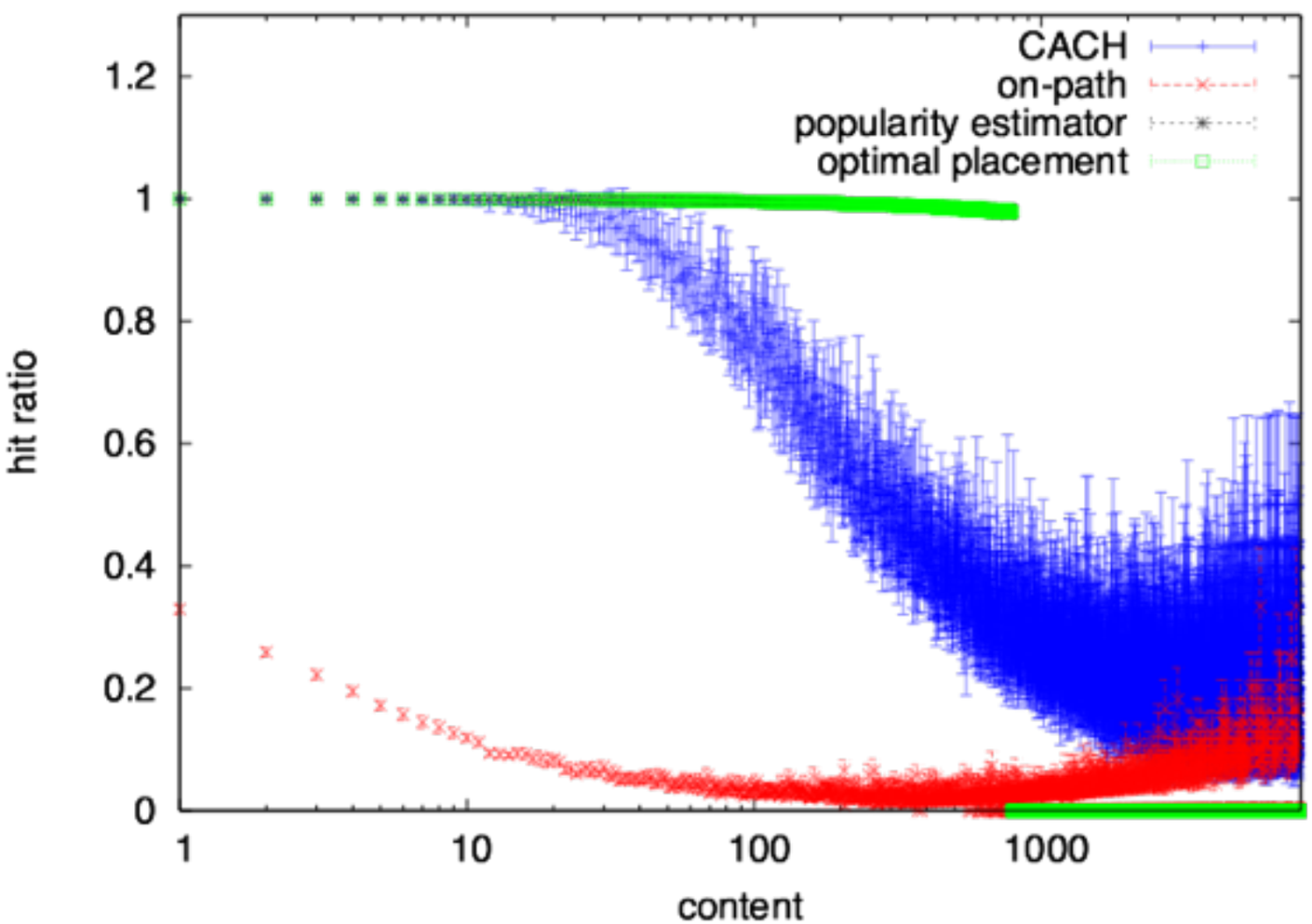
437

The top 5.5% of popular contents accounts for 50% of the peering traffic, while at optimal they account only for 0.7%

Off-path caching improves hit ratio

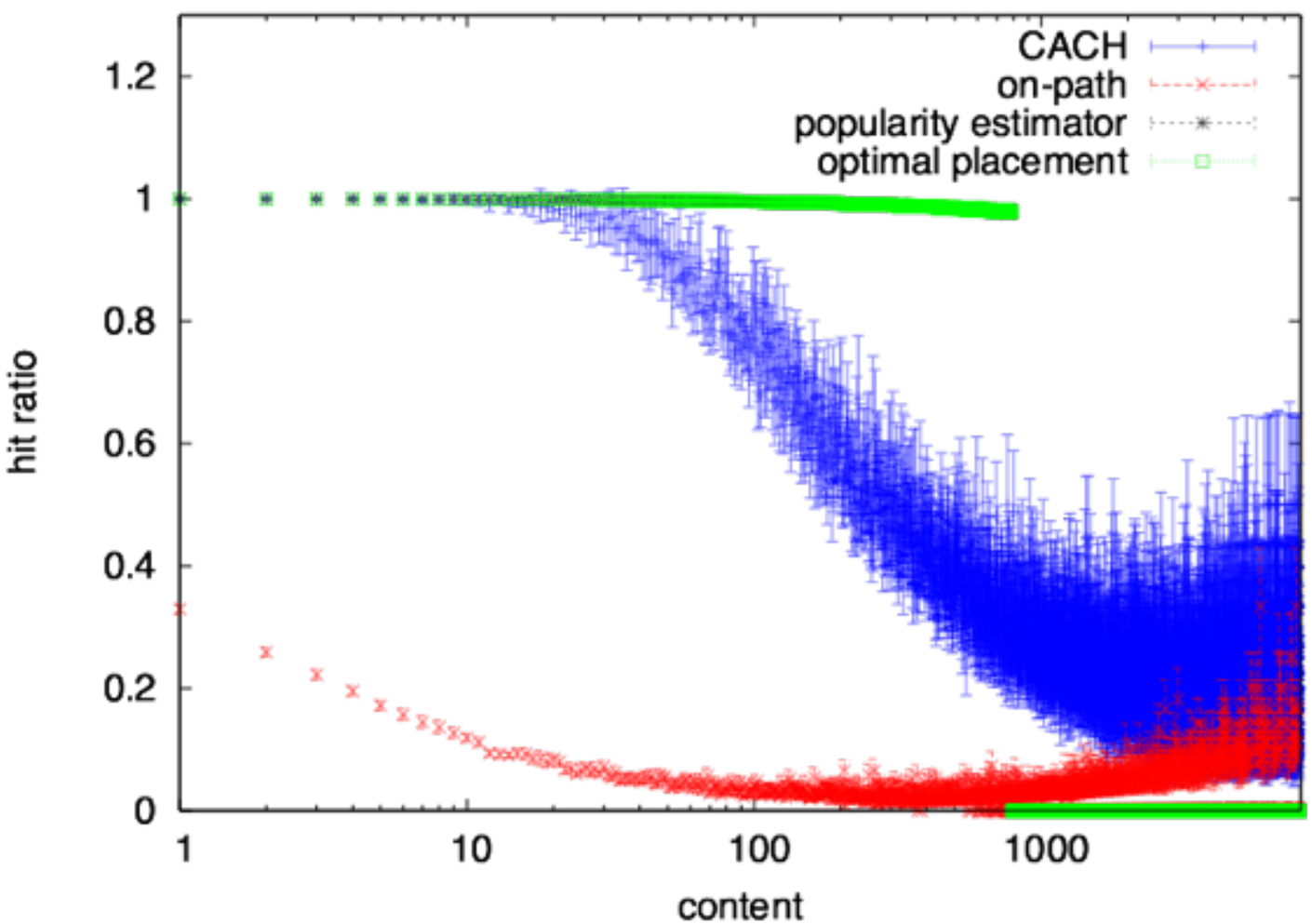


Off-path caching improves hit ratio



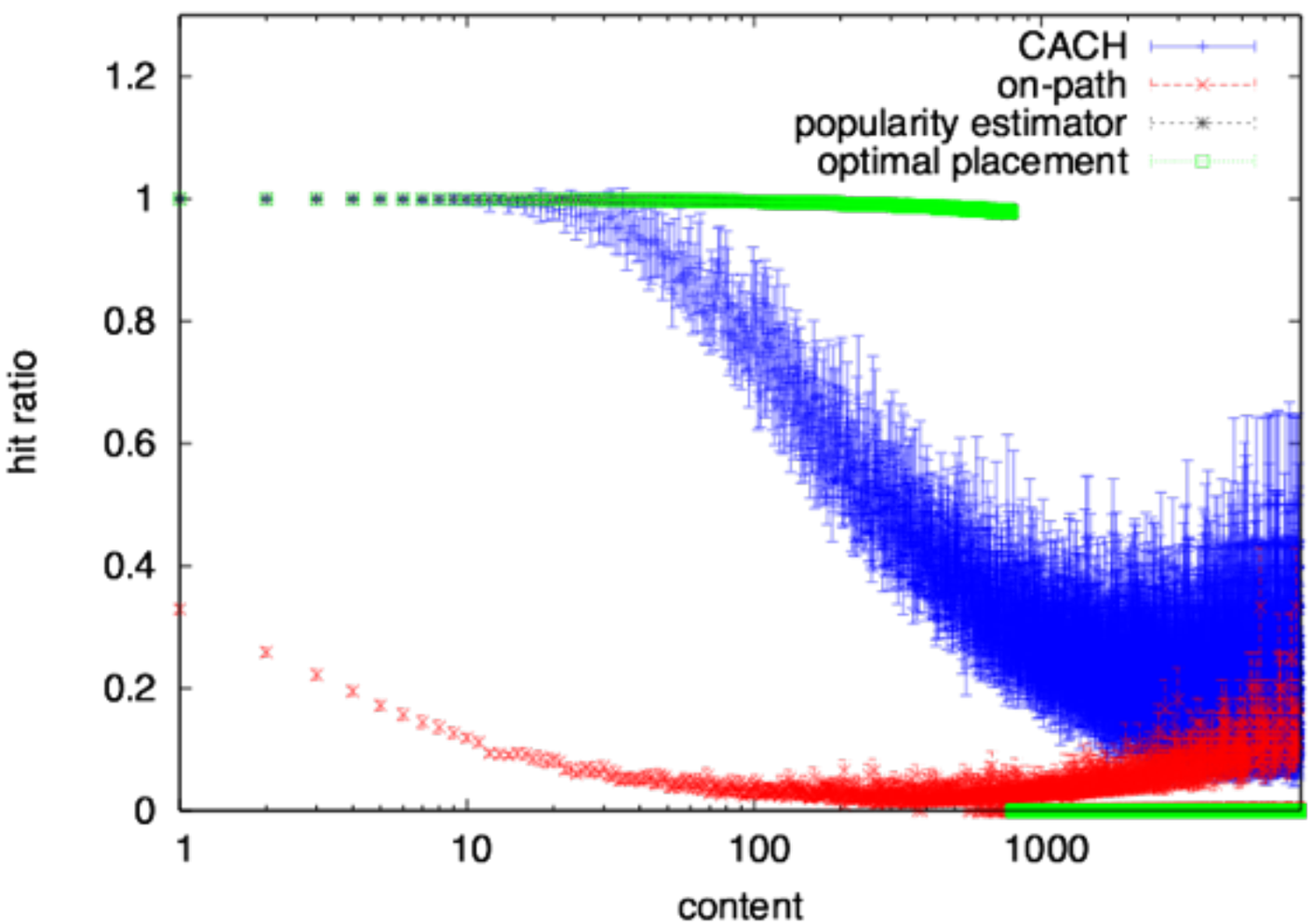
- High hit ratio for popular contents

Off-path caching improves hit ratio



- High hit ratio for popular contents
- The overall hit ratio significantly increases from 17% to 53% and 65%

Off-path caching improves hit ratio



- High hit ratio for popular contents
- The overall hit ratio significantly increases from 17% to 53% and 65%
- What is the impact on delay?

Off-path caching improves retrieval delay

On-path	CACH	Optimal placement
$5.11ms \pm 0.05$	$28.08ms \pm 0.04$	$23.52ms \pm 0.03$

Off-path caching improves retrieval delay

- Once a content is cached, the deflection has a negative impact on the average retrieval delay

On-path	CACH	Optimal placement
$5.11ms \pm 0.05$	$28.08ms \pm 0.04$	$23.52ms \pm 0.03$

- But the overall average retrieval delay is reduced with off-path caching, thanks to a better hit ratio

On-path	CACH	Optimal placement
$154.42ms \pm 0.05$	$119.19ms \pm 0.11$	$84.23ms \pm 0.09$

Information-centric networking is a long term work

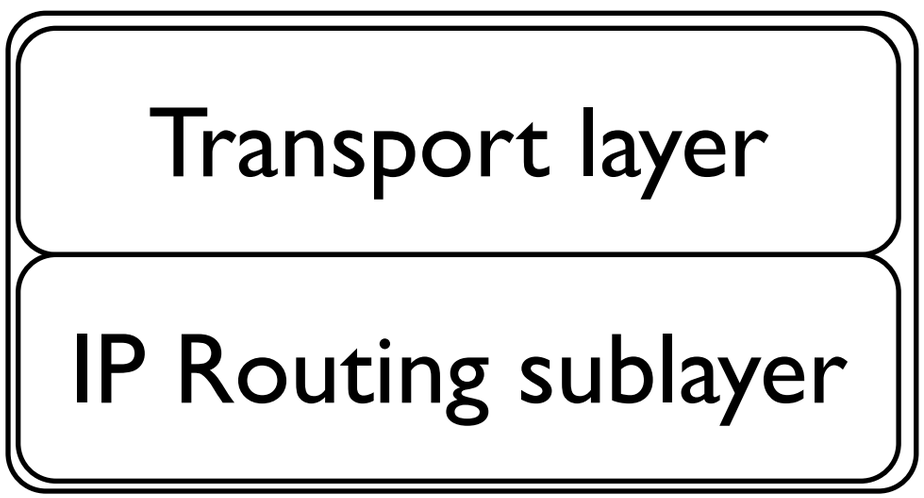
How to reconcile the two worlds?

- Clean-slate solutions
  - Rethink the paradigms
- **Evolutionary solutions**
  - Enhance current architecture
  - Provide interworking mechanisms

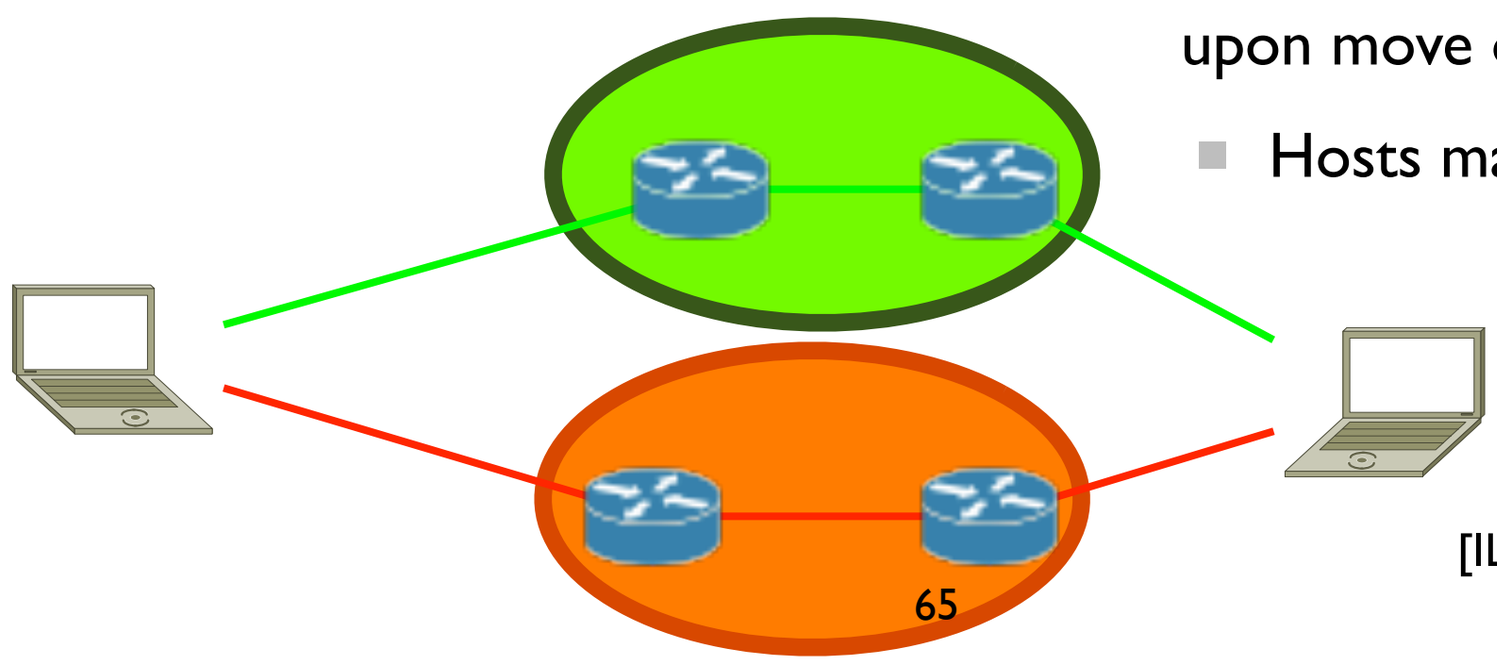
Separating Identifiers from Locators

- Today, changing the locator means changing the identifier, breaking the pending flows
- Separating the locator and the identifier roles to avoid breaking flows
  - Host-based approach
  - Network-based approach

Host-based Loc/ID split

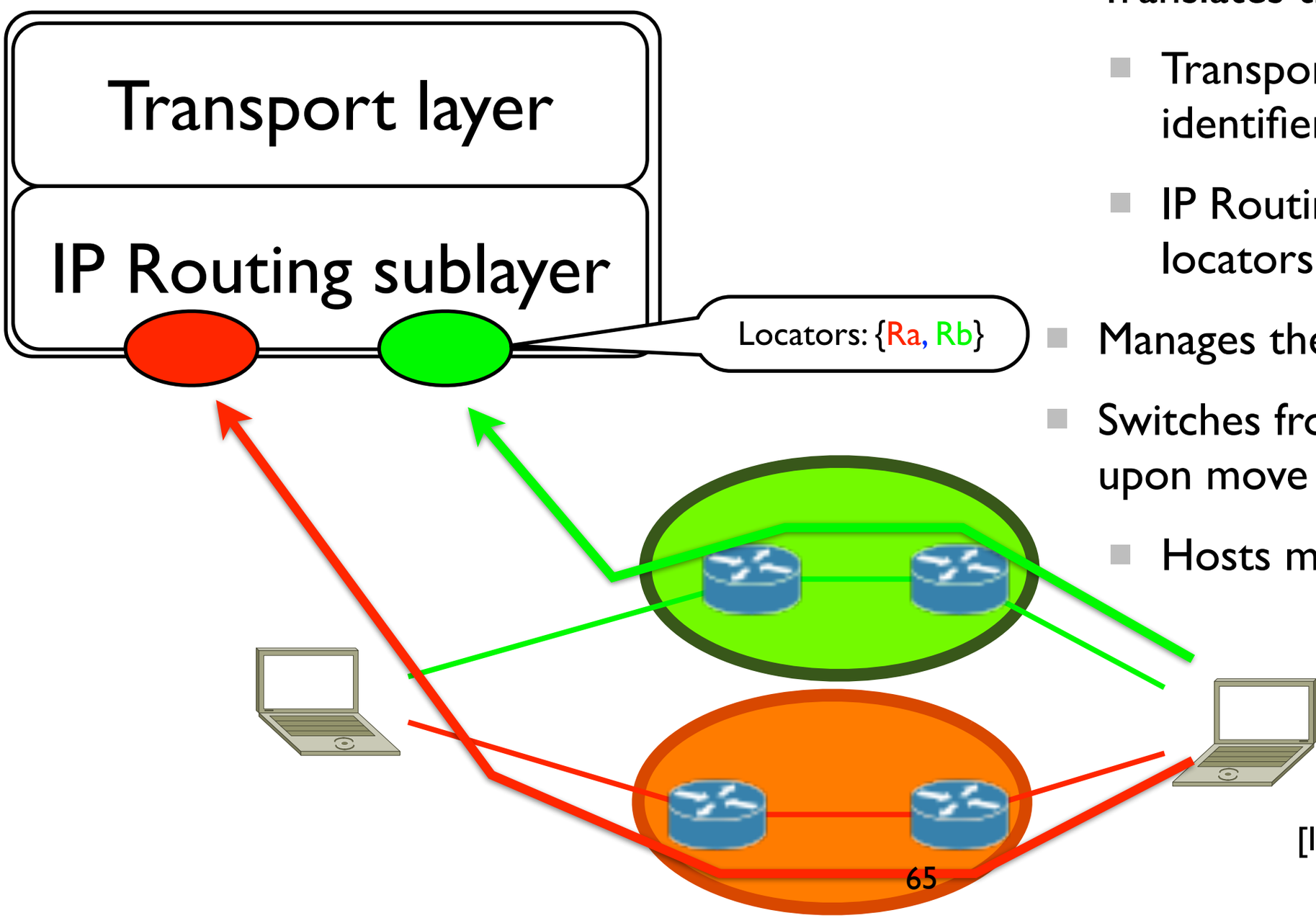


- Roles
  - Translates the packets so that
    - Transport layer only sees the host identifier
    - IP Routing sublayer sees only locators
  - Manages the set of locators
  - Switches from one locator to another upon move or after link failure
    - Hosts maintain some state



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

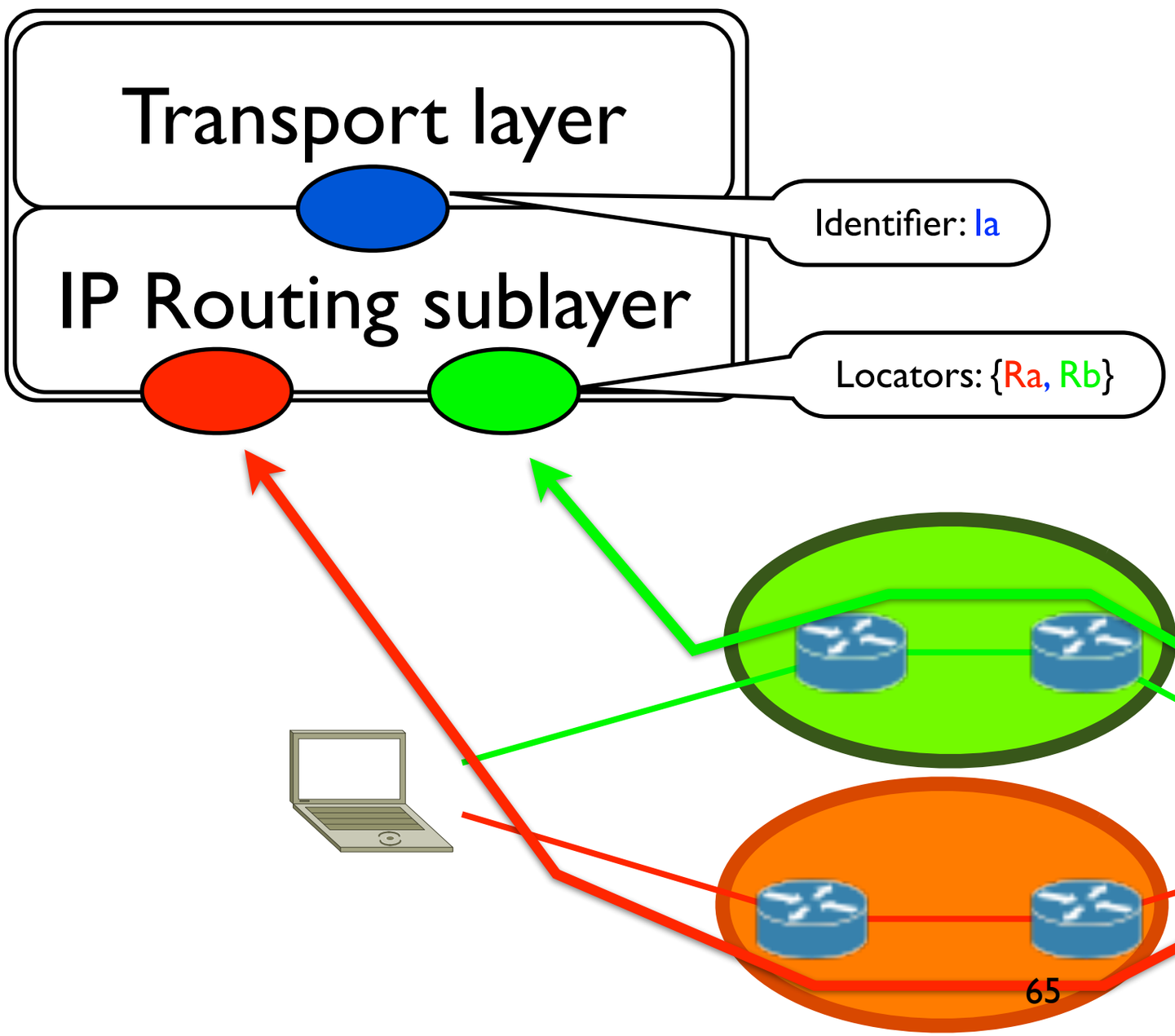
Host-based Loc/ID split



- Roles
  - Translates the packets so that
    - Transport layer only sees the host identifier
    - IP Routing sublayer sees only locators
  - Manages the set of locators
  - Switches from one locator to another upon move or after link failure
    - Hosts maintain some state

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

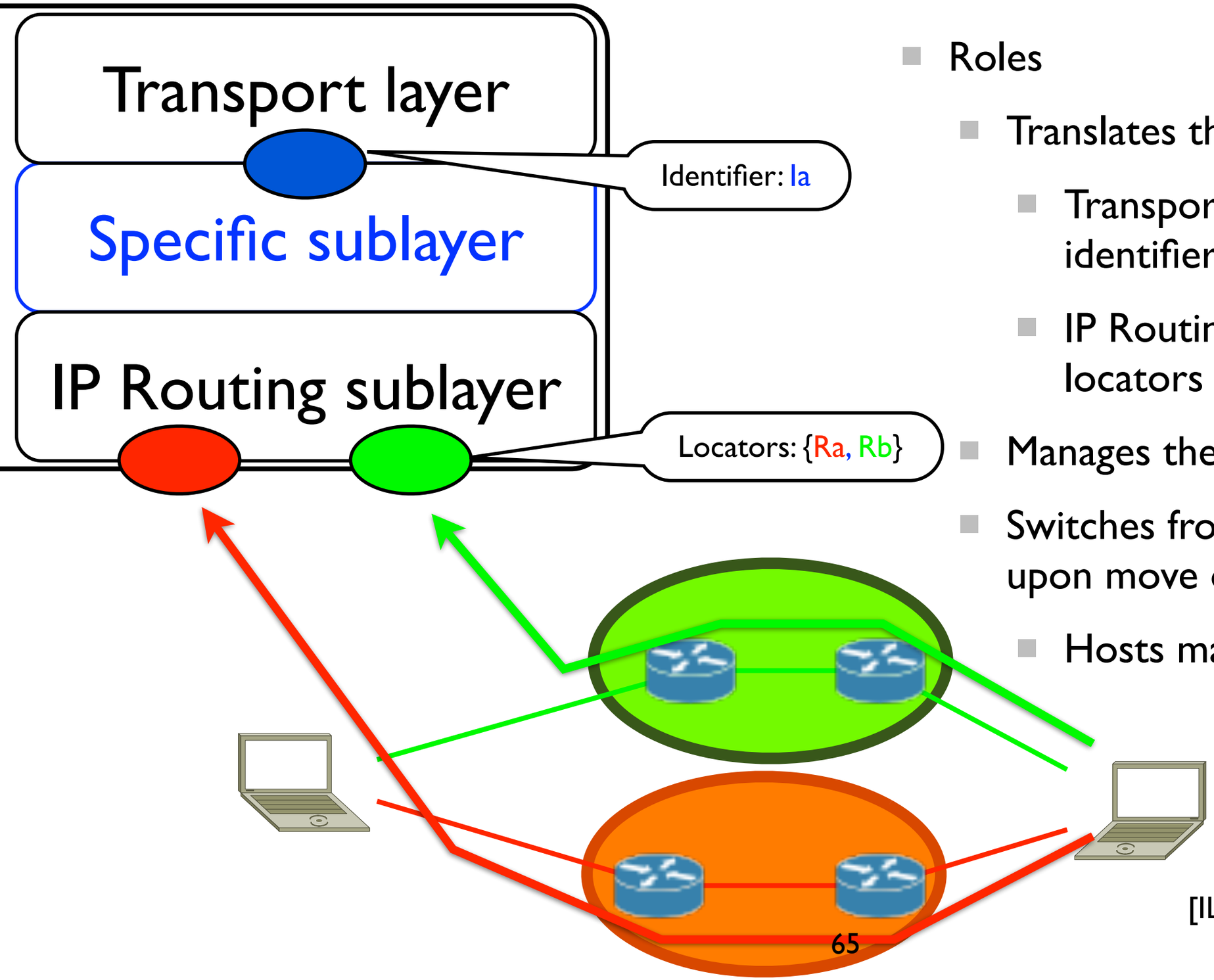
Host-based Loc/ID split



- Roles
  - Translates the packets so that
    - Transport layer only sees the host identifier
    - IP Routing sublayer sees only locators
  - Manages the set of locators
  - Switches from one locator to another upon move or after link failure
    - Hosts maintain some state

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

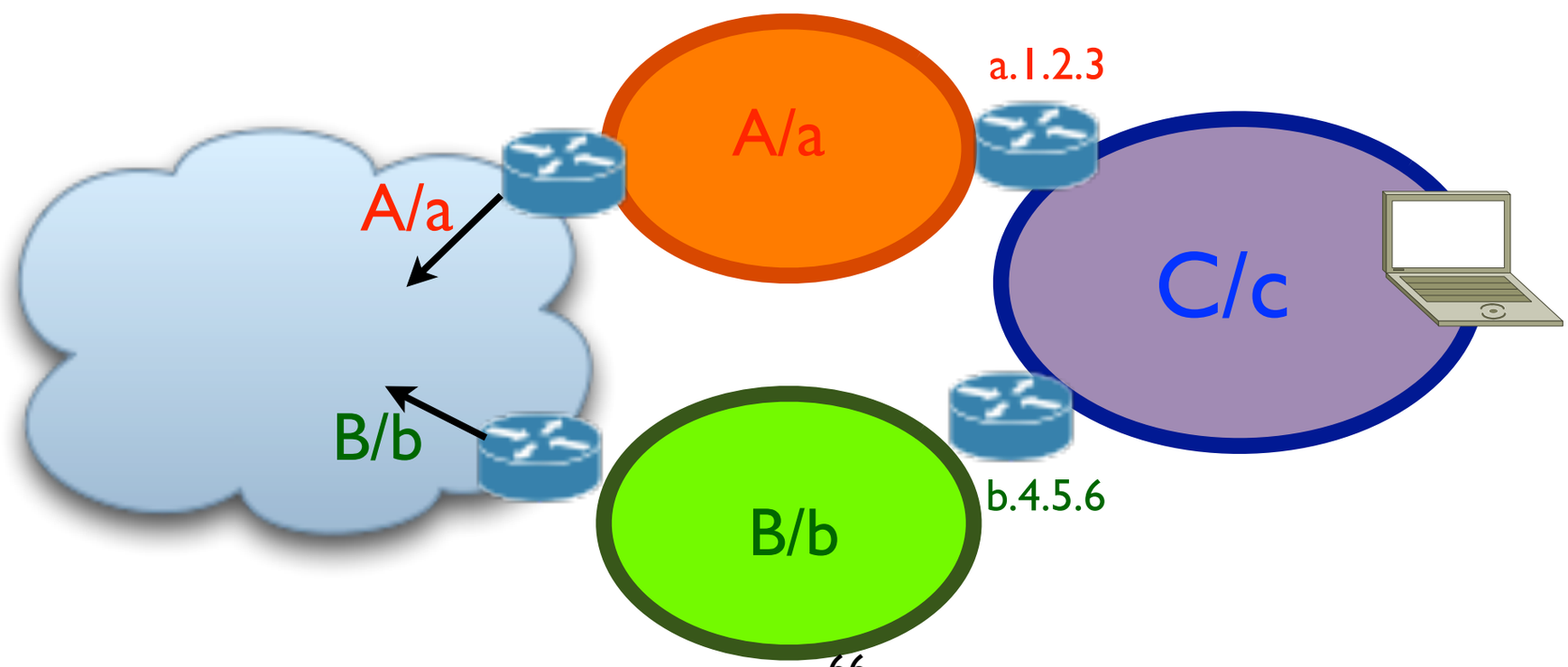
Host-based Loc/ID split



- Roles
  - Translates the packets so that
    - Transport layer only sees the host identifier
    - IP Routing sublayer sees only locators
  - Manages the set of locators
  - Switches from one locator to another upon move or after link failure
    - Hosts maintain some state

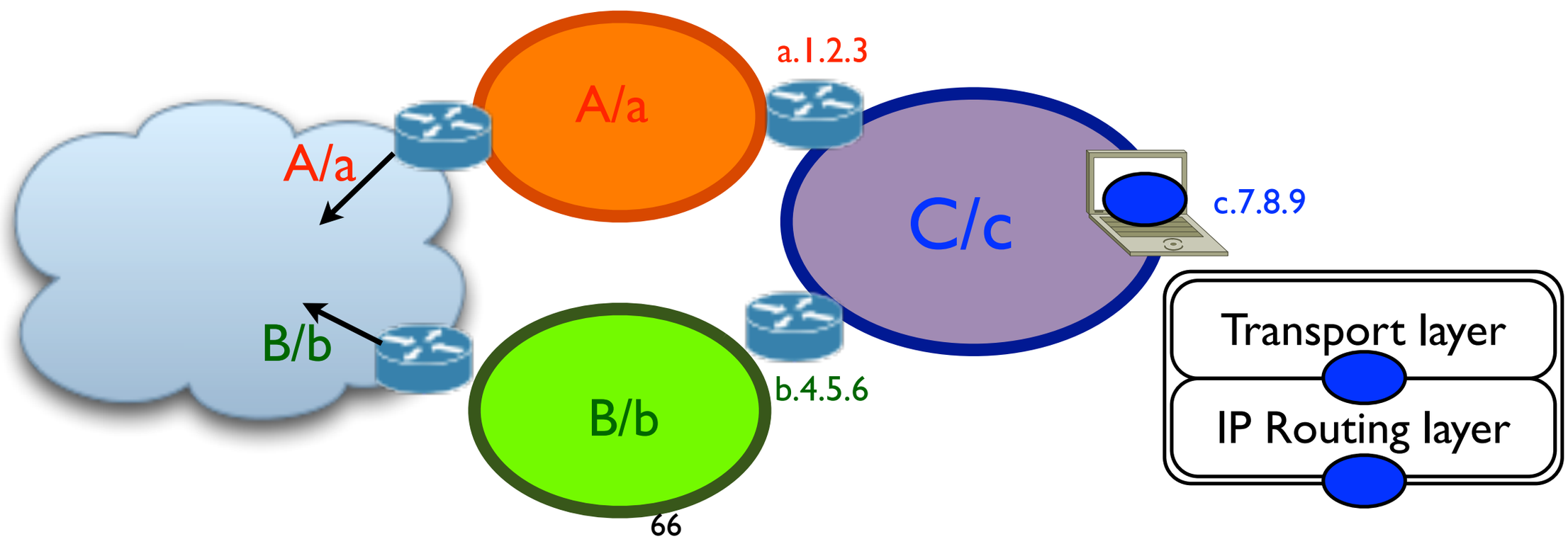
[ILNP,HIP, shim6, Six/One,MPTCP]

Network-based Loc/ID split



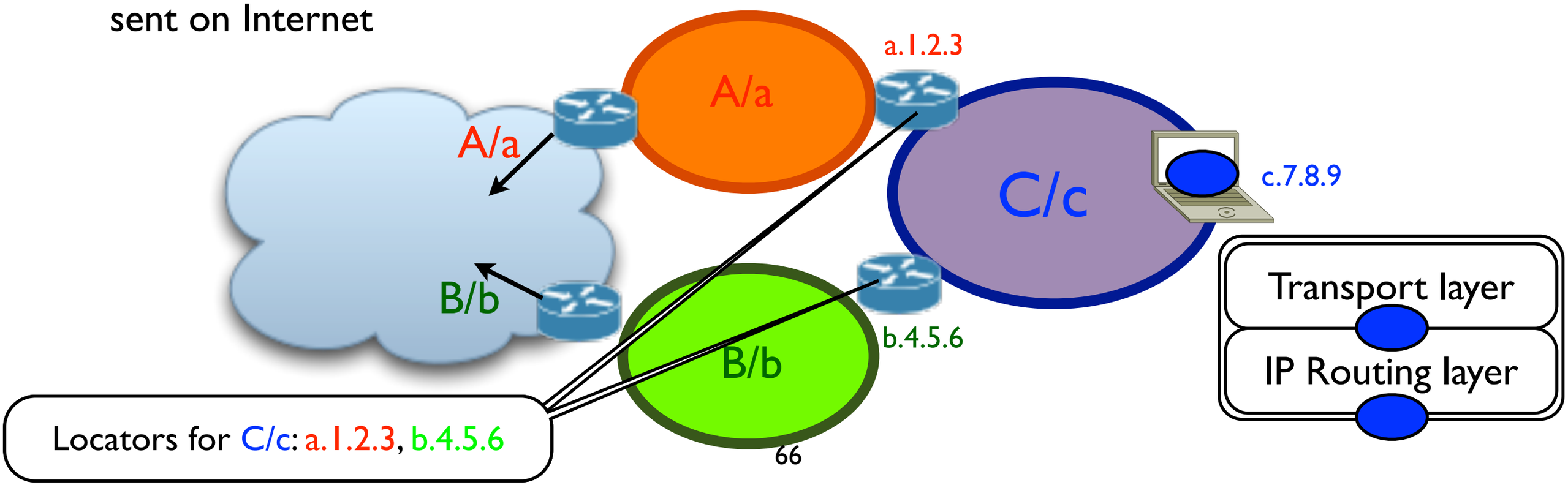
Network-based Loc/ID split

- Host's IP stack unchanged
  - Each host has one stable IP address
  - Used as identifier
  - **Not globally routable**



Network-based Loc/ID split

- Each edge router owns
  - Globally routed addresses used as locators
- **Mapping mechanism** is used to find locators associated to one identifier
- Packets from hosts are modified before being sent on Internet
- Host's IP stack unchanged
  - Each host has one stable IP address
  - Used as identifier
  - **Not globally routable**



Host vs Network-based Loc/ID split

- We need both!
- At work, connected directly to the wall
  - Let my company doing the voodoo for the whole network
- In the street, calling with Skype over WIFI&LTE
  - Prefer WIFI to LTE

The Locator/Identifier Separation Protocol (LISP)

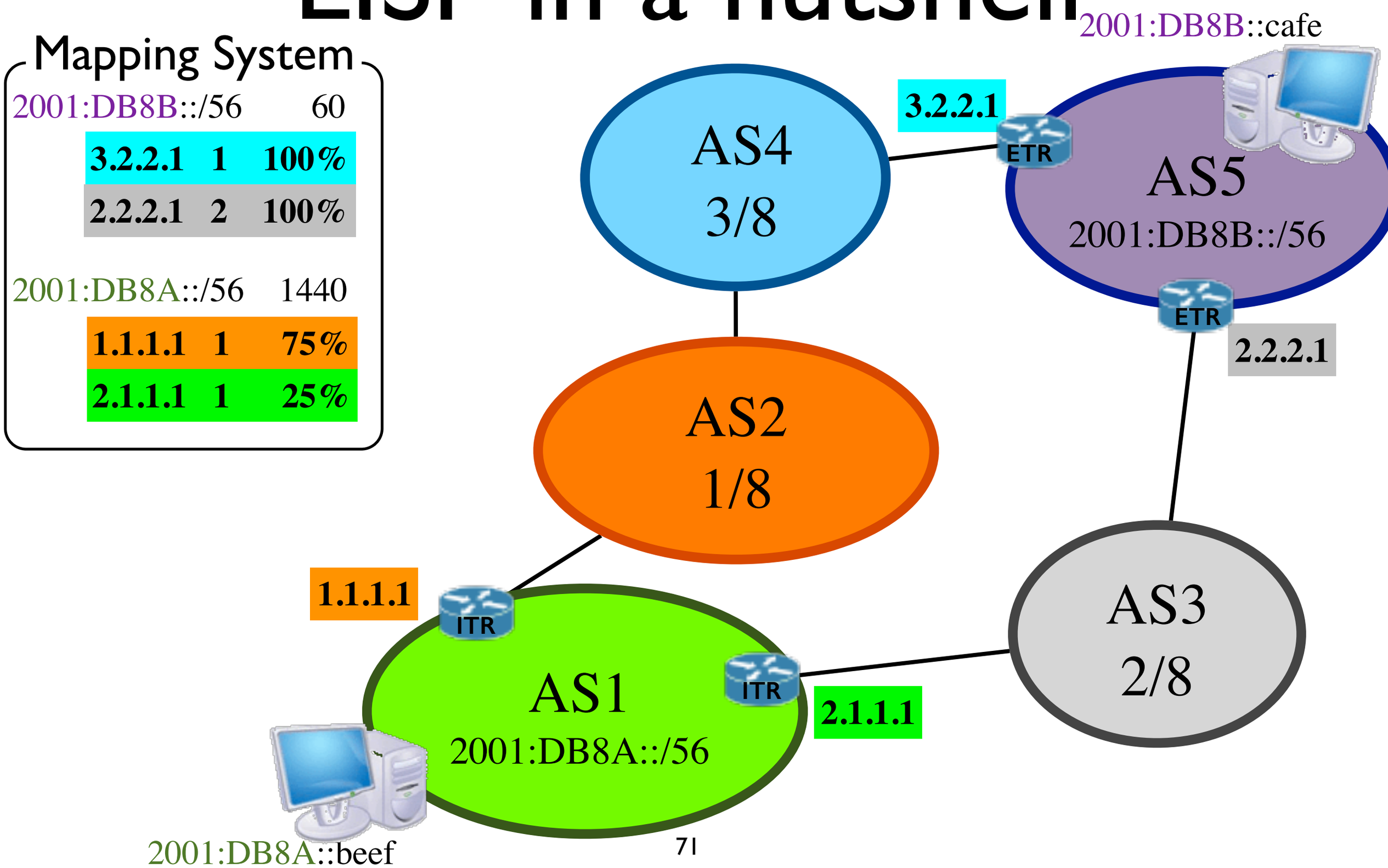
LISP philosophy (1/2)

- 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

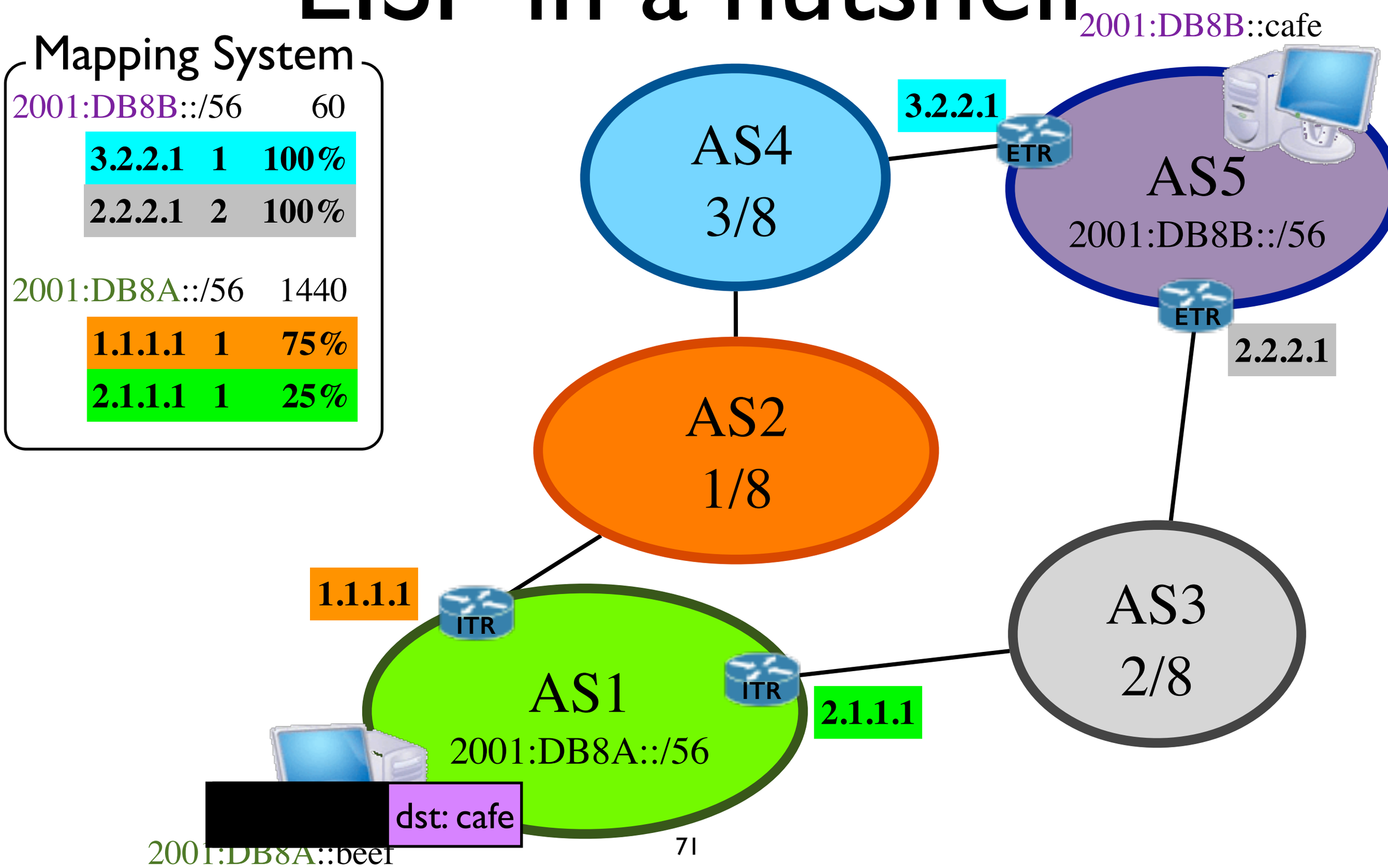
LISP philosophy (2/2)

- Follows the Map-and-Encap principle
  - A **mapping system** maps EID prefixes onto site routers RLOCs
  - Routers **encapsulate** (ITR) packets received from hosts before sending them towards the destination RLOC
  - Routers **decapsulate** (ETR) packets received from the Internet before sending them towards the destination hosts

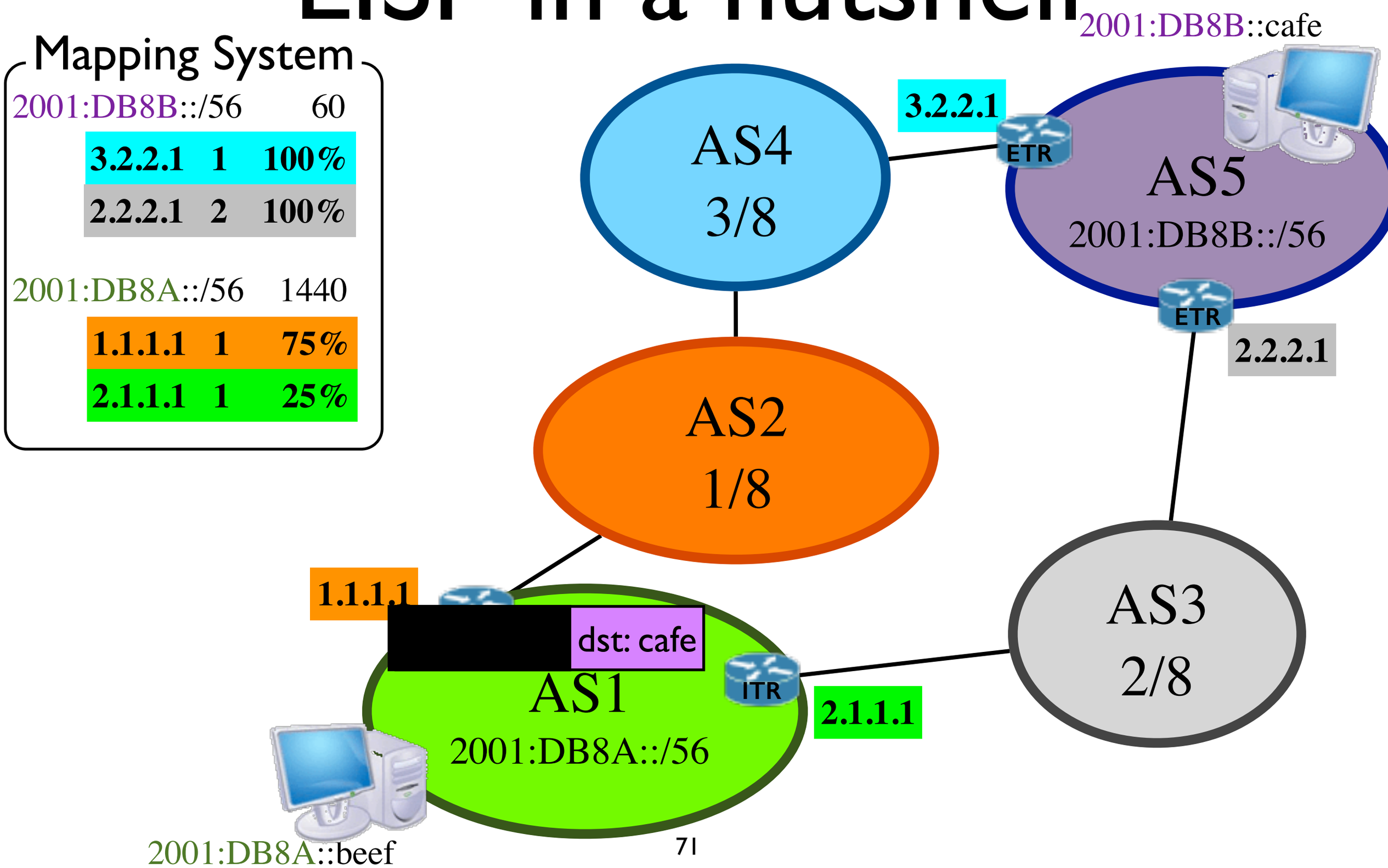
LISP in a nutshell



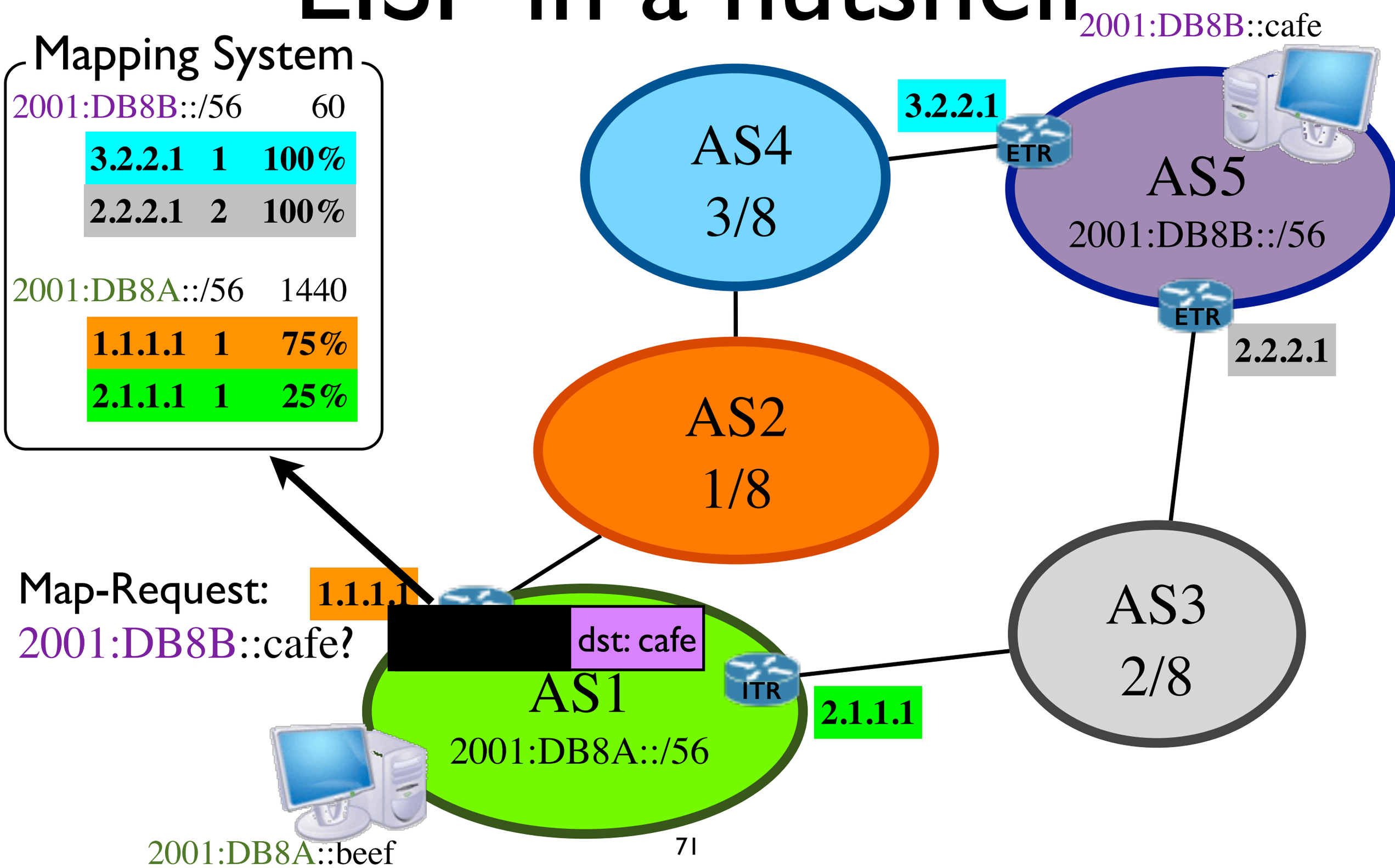
LISP in a nutshell



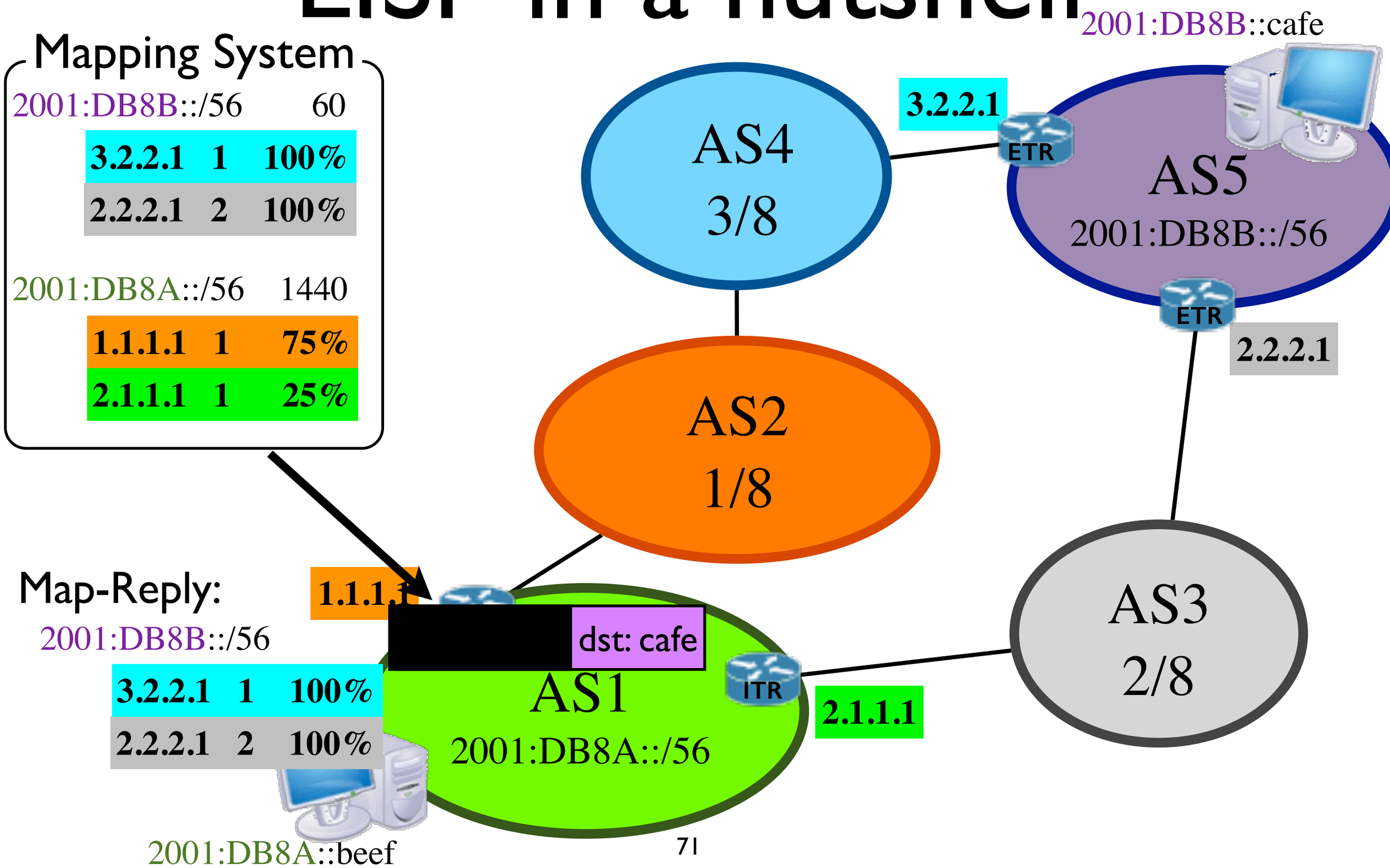
LISP in a nutshell



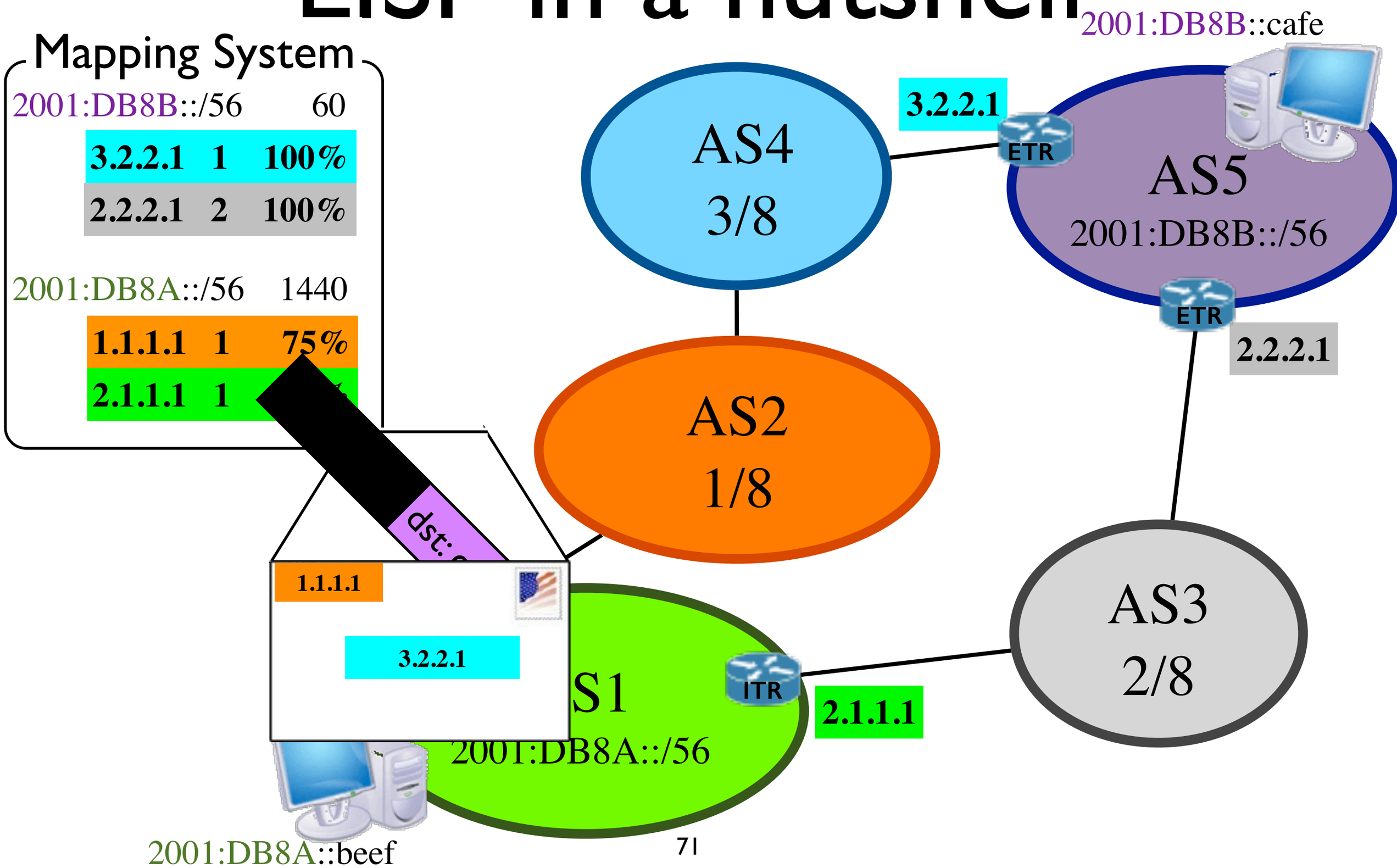
LISP in a nutshell



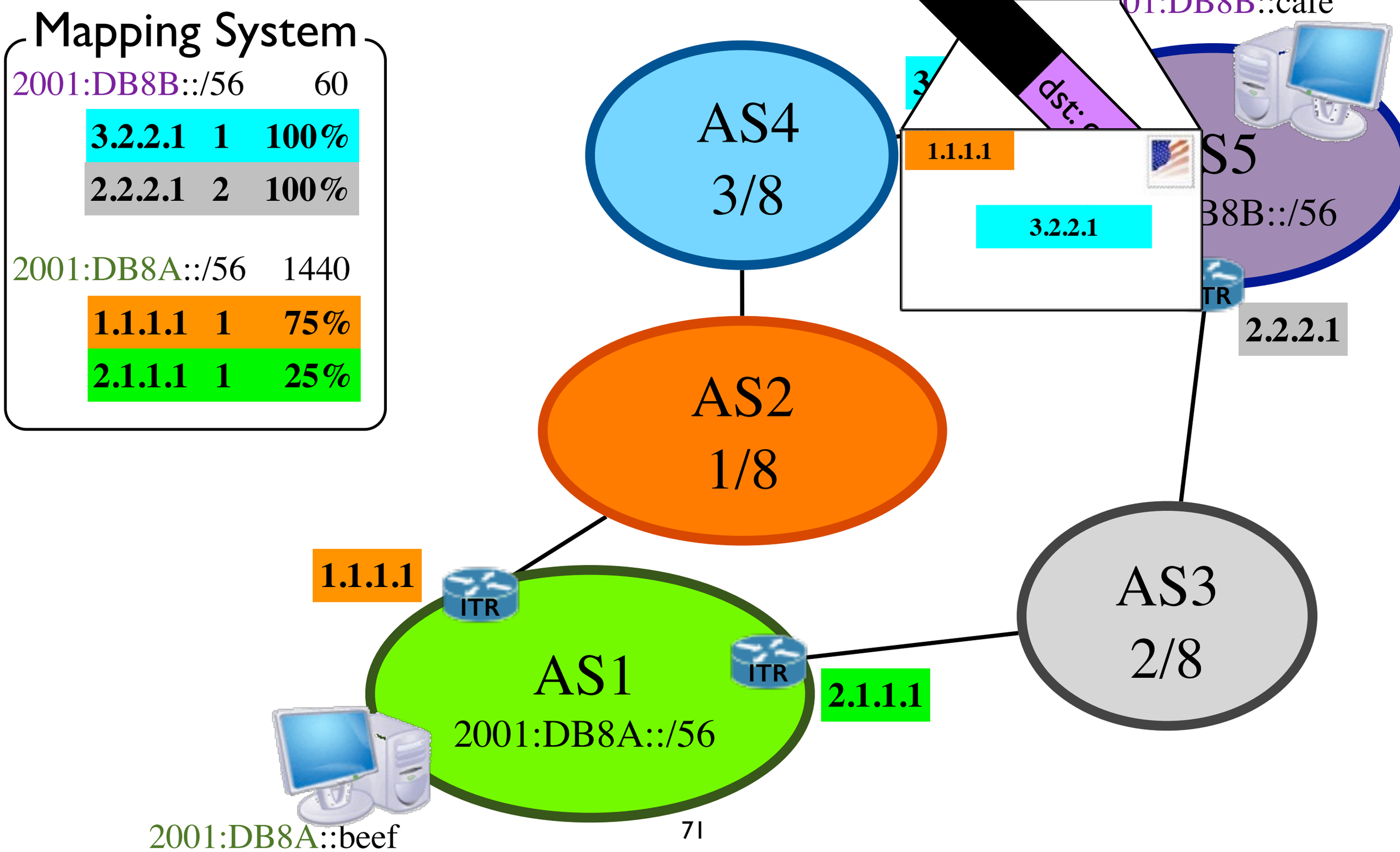
LISP in a nutshell



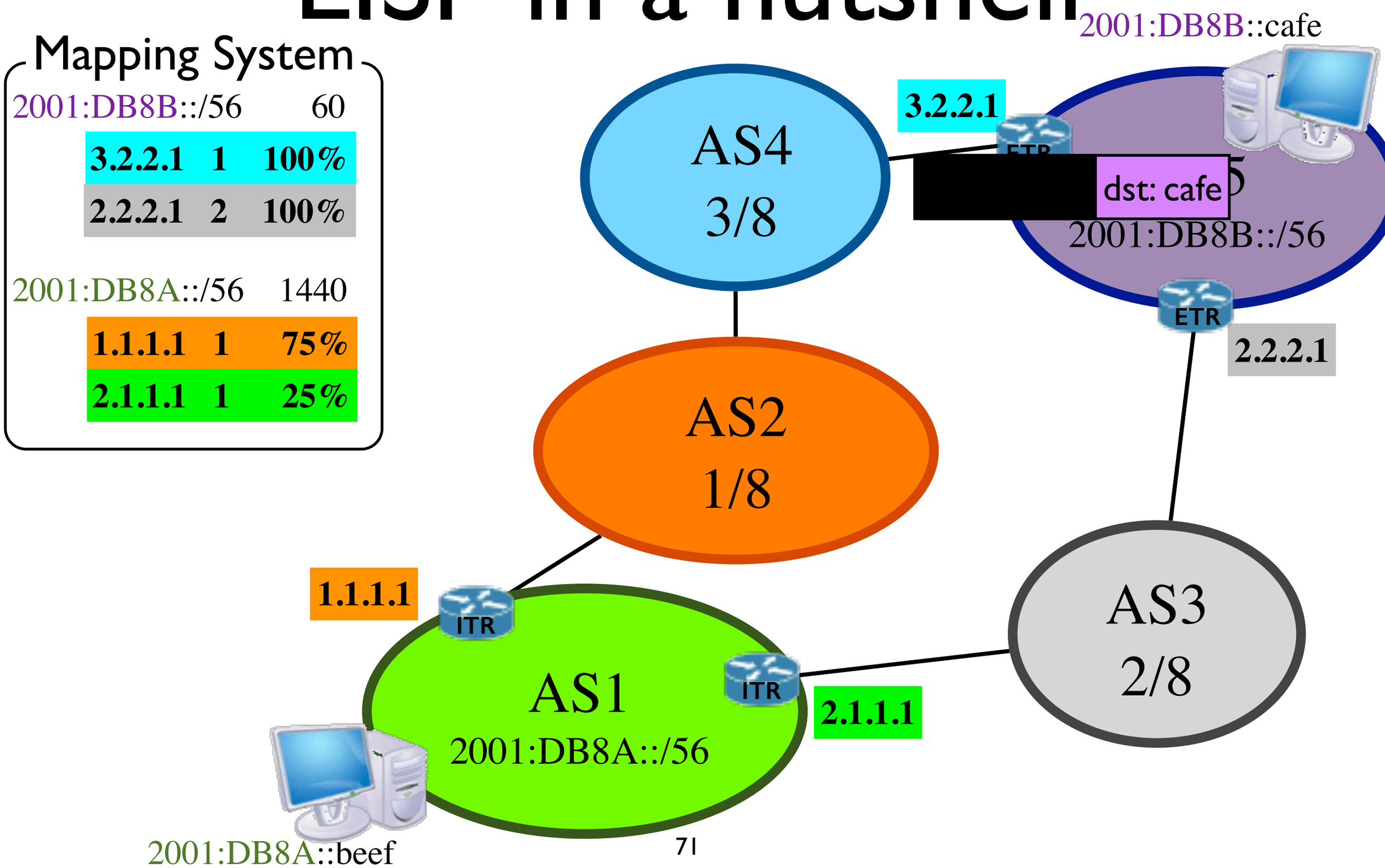
LISP in a nutshell



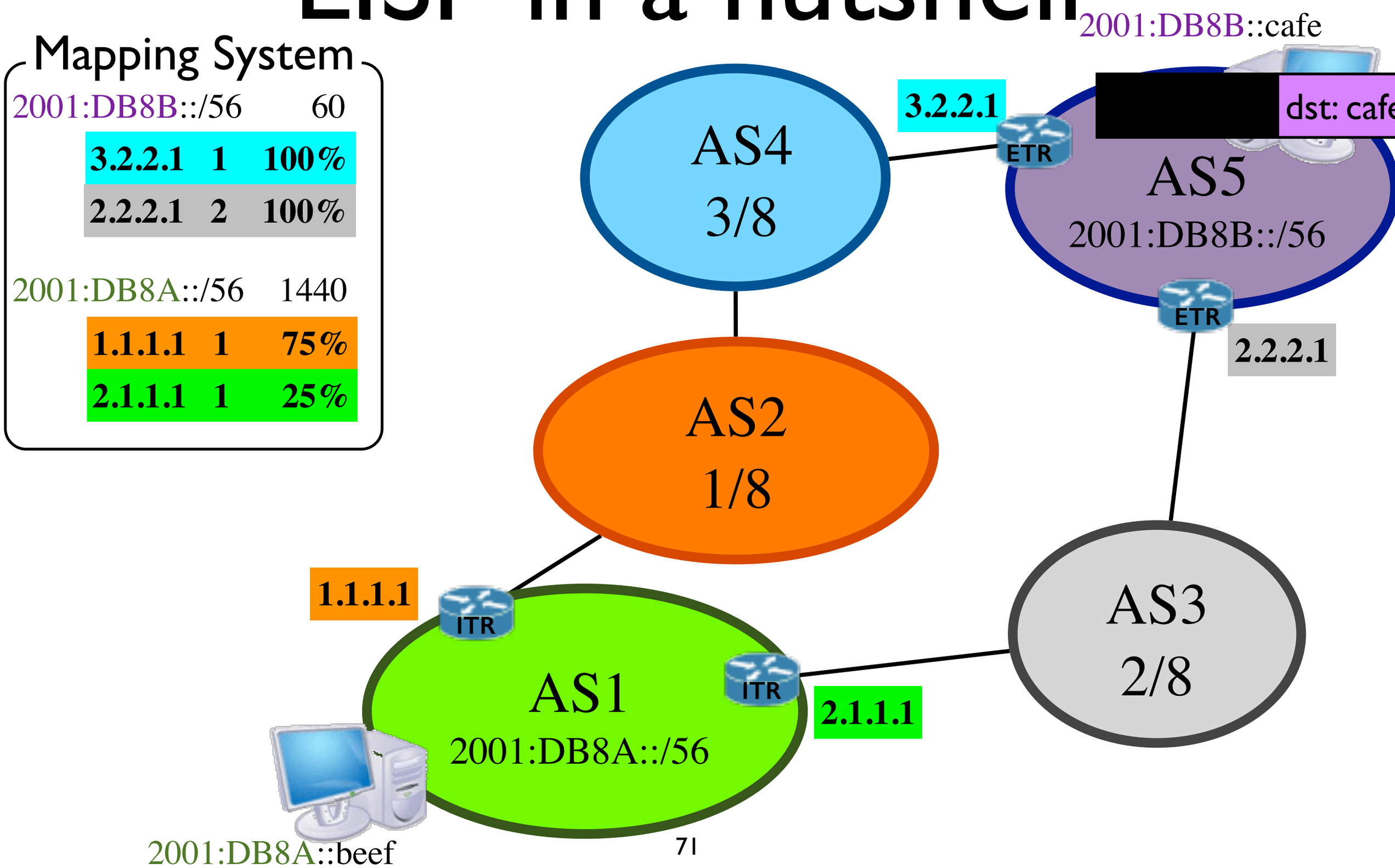
LISP in a nutshell



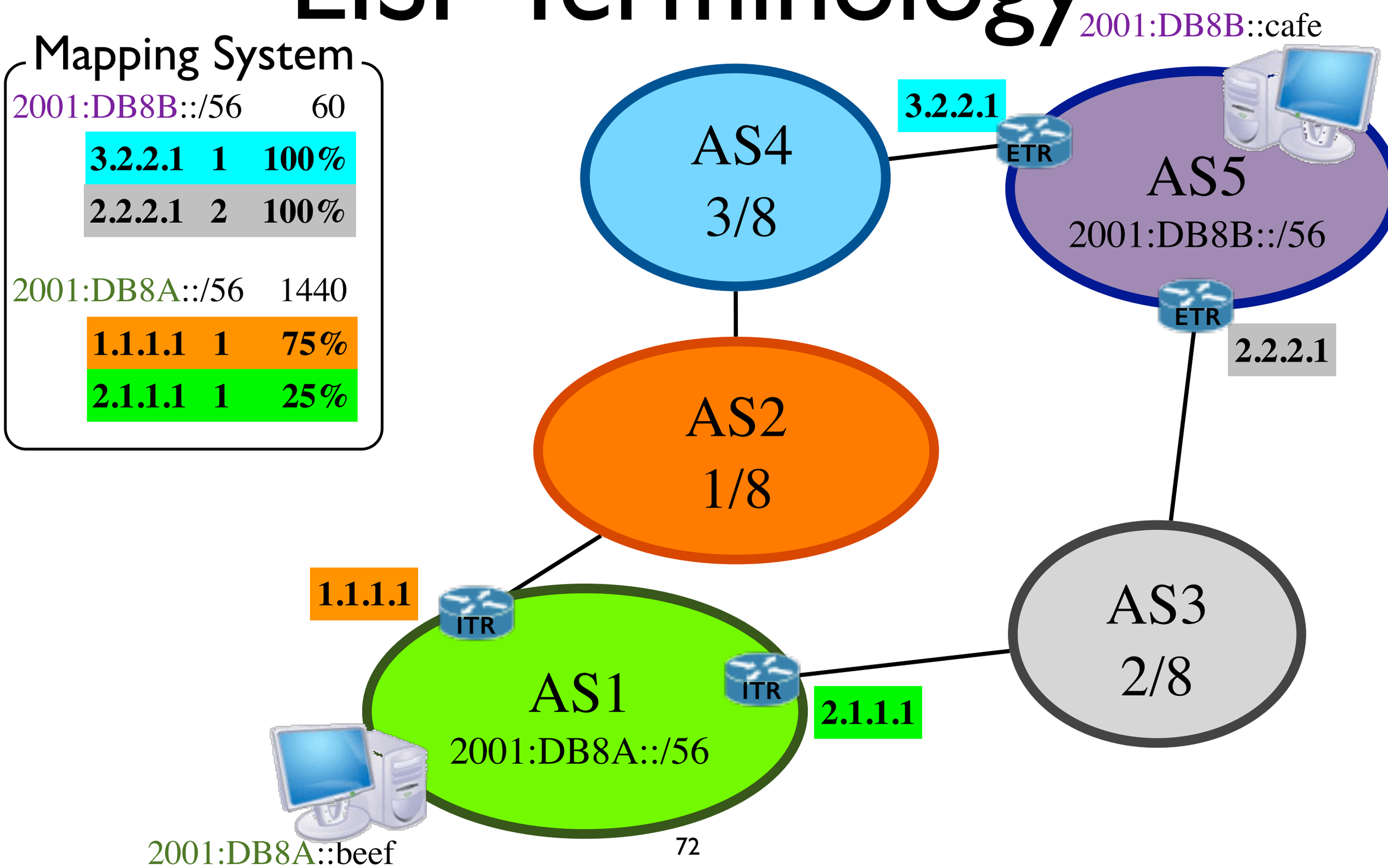
LISP in a nutshell



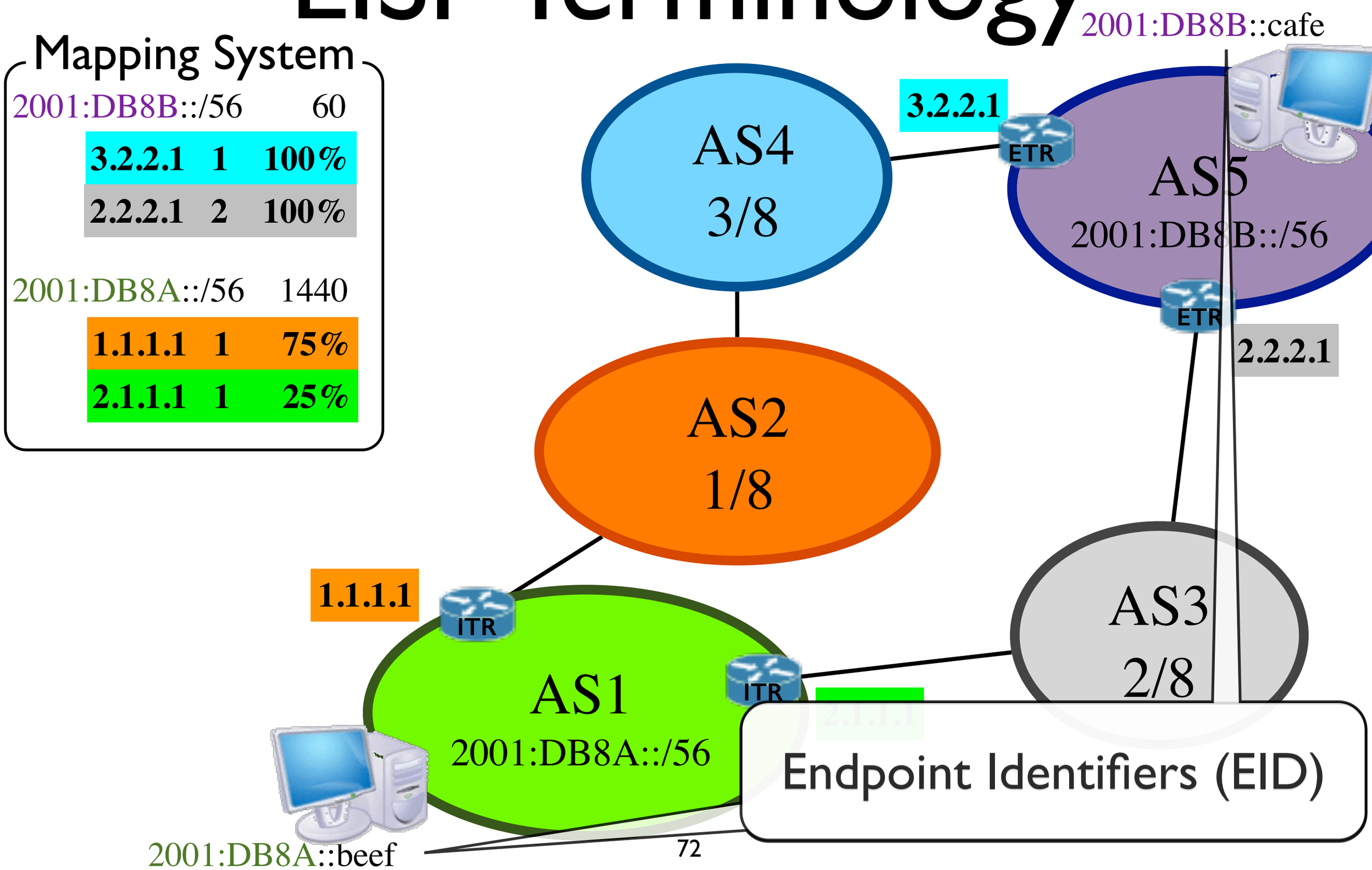
LISP in a nutshell



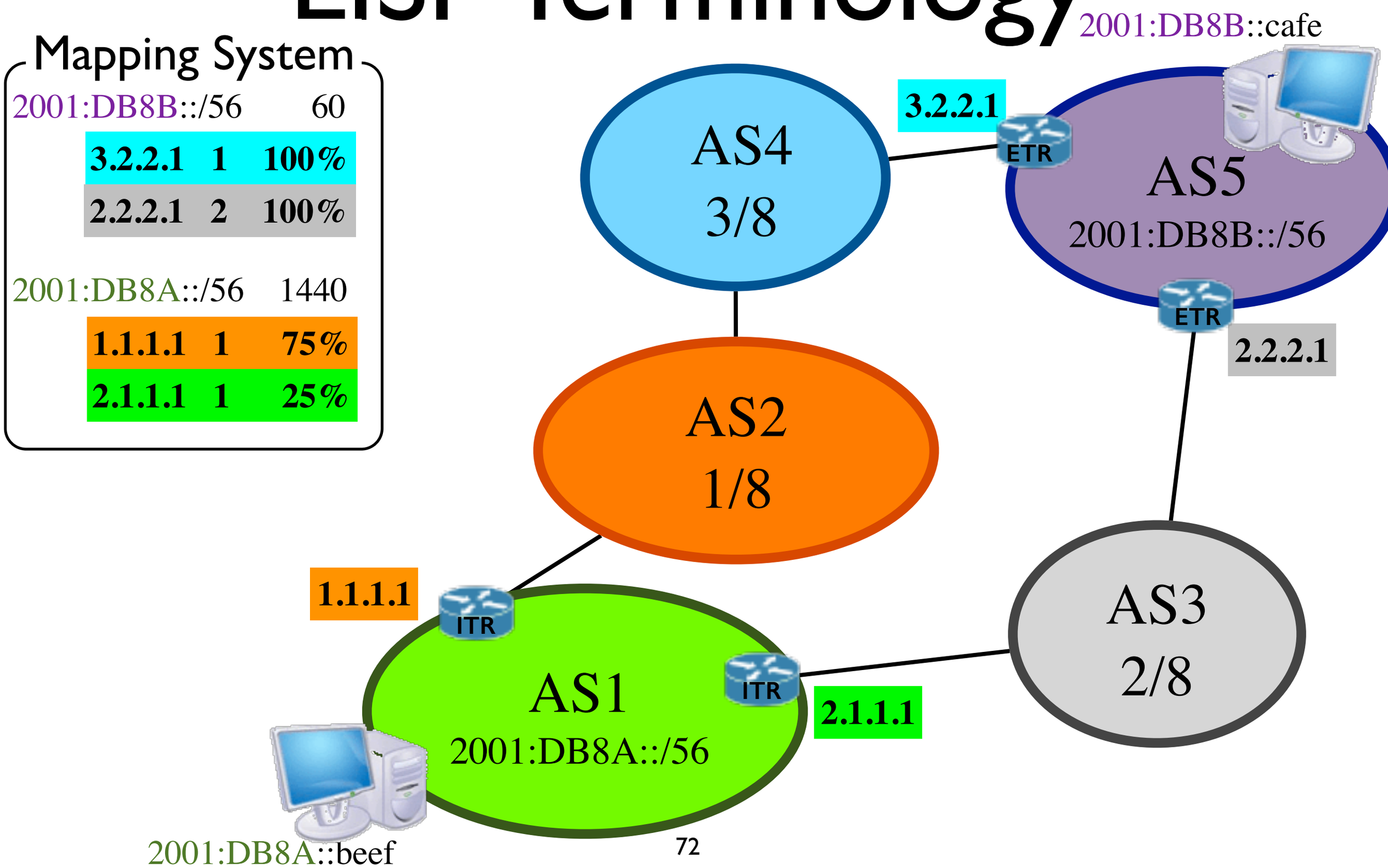
LISP Terminology



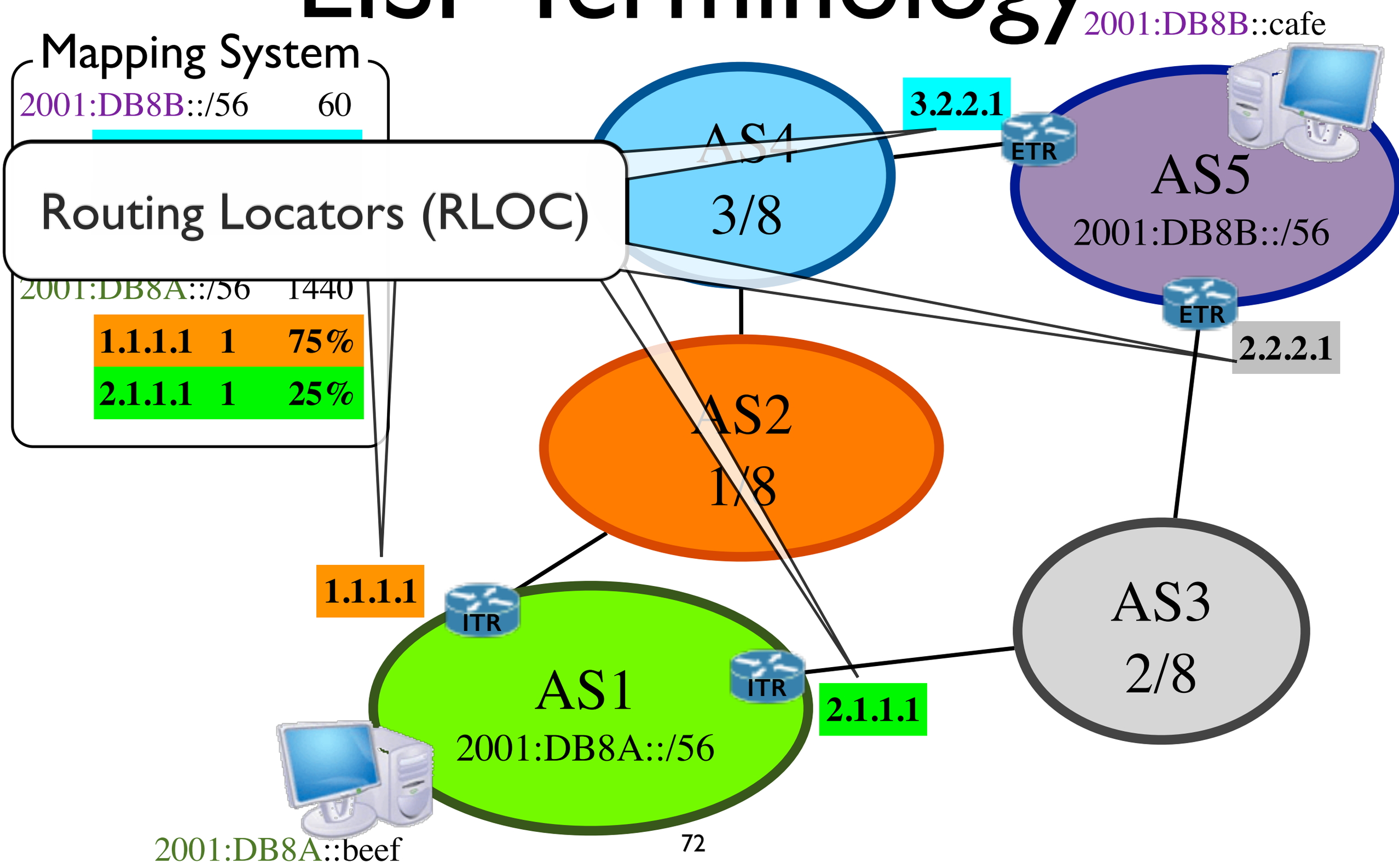
LISP Terminology



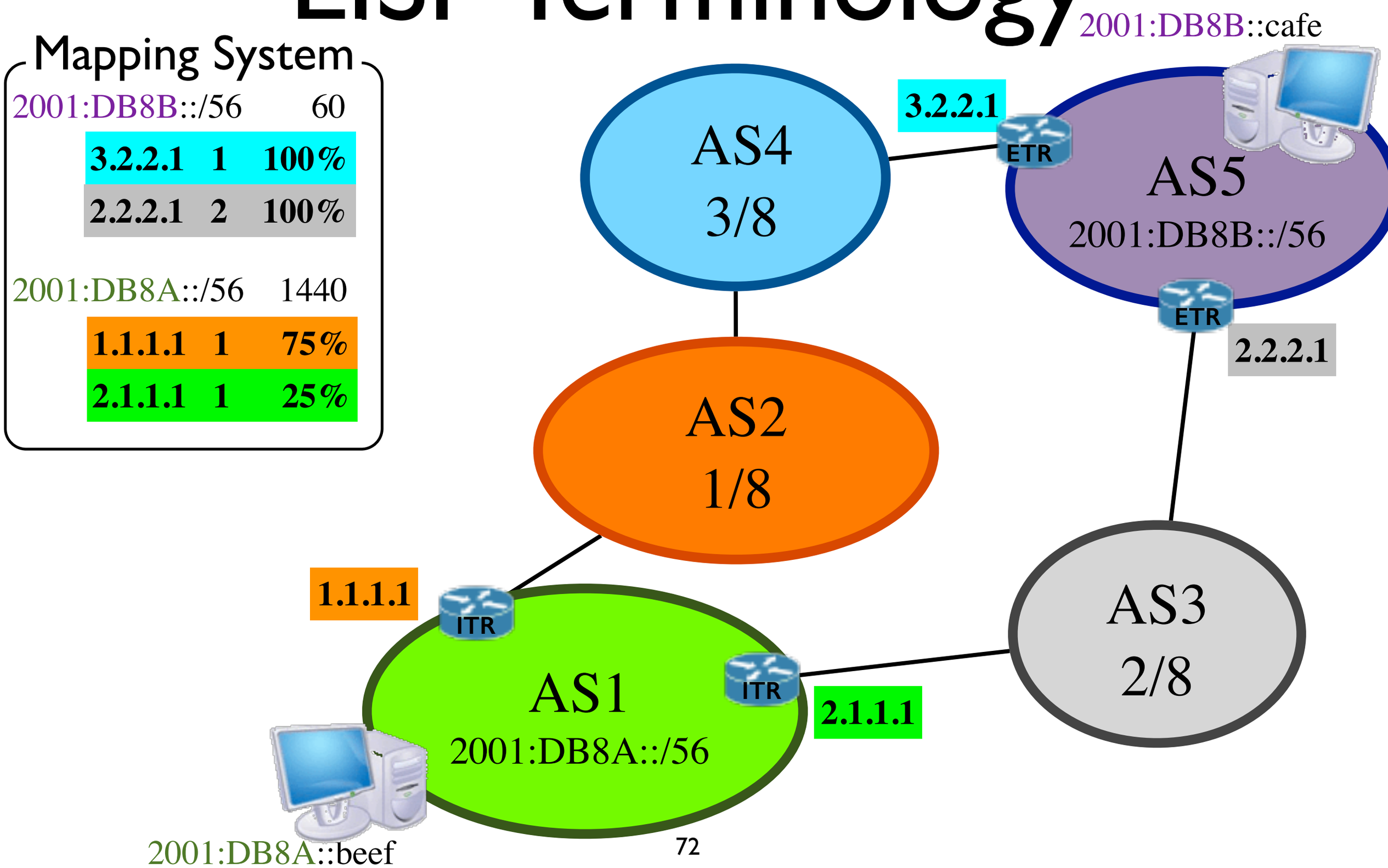
LISP Terminology



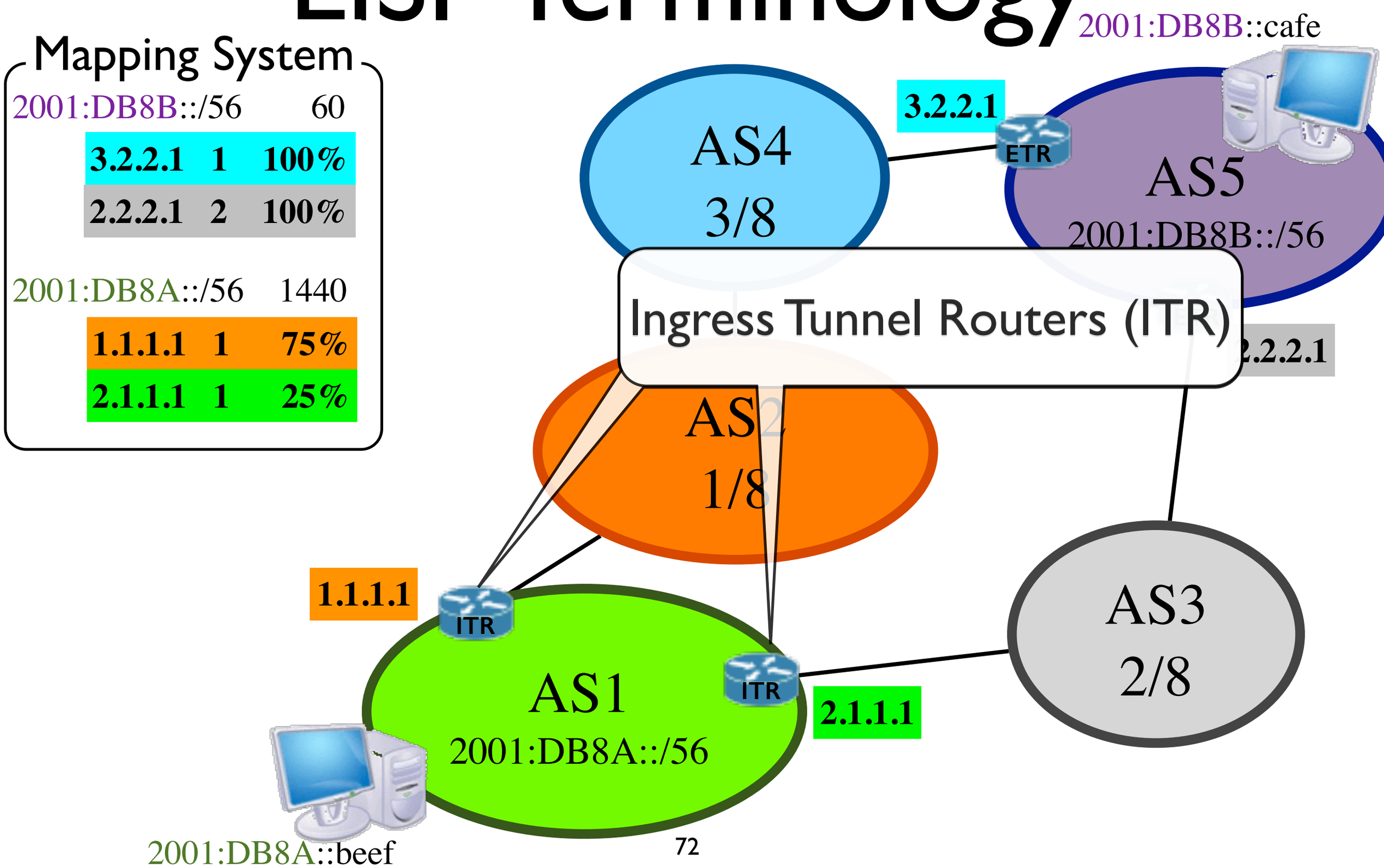
LISP Terminology



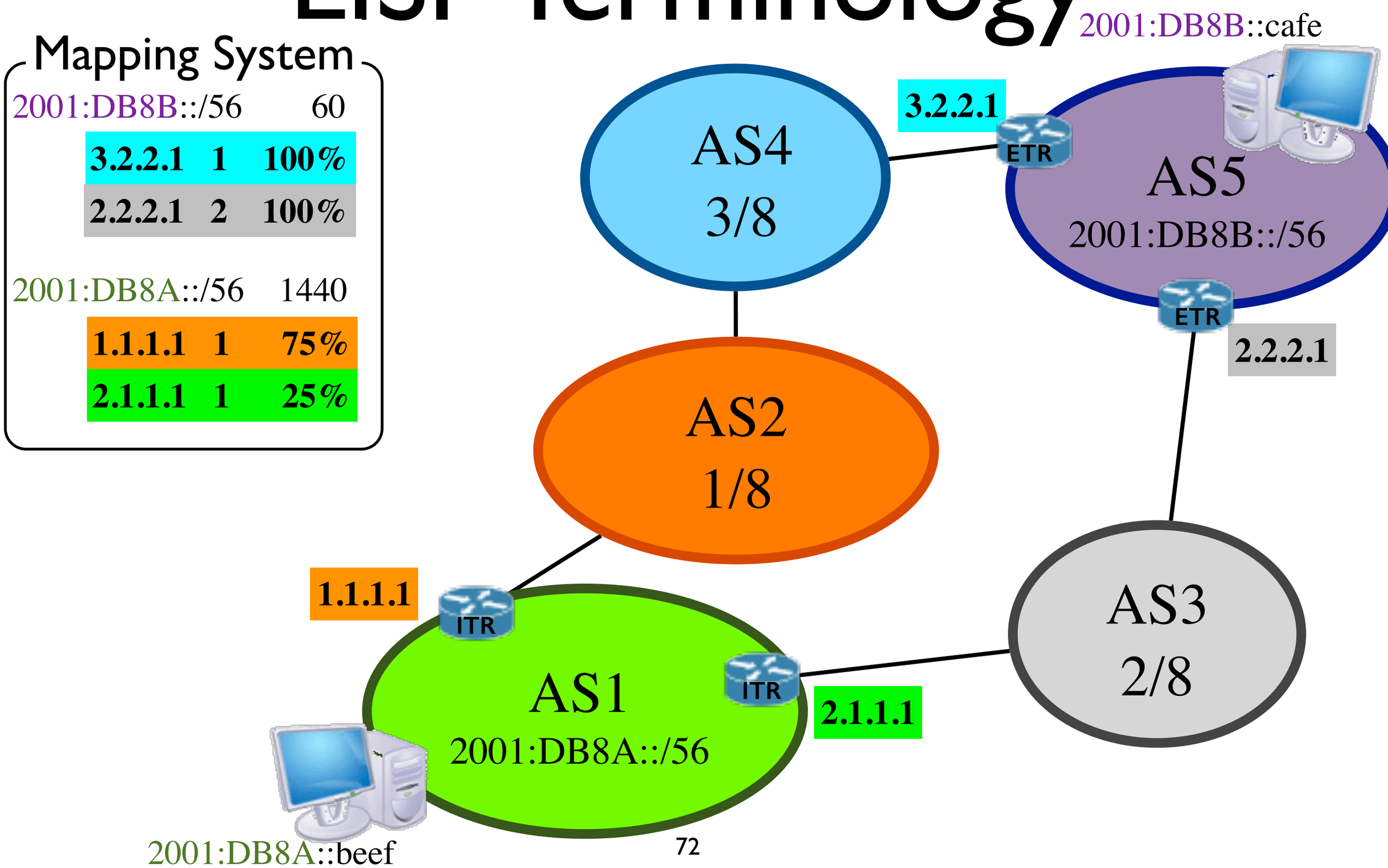
LISP Terminology



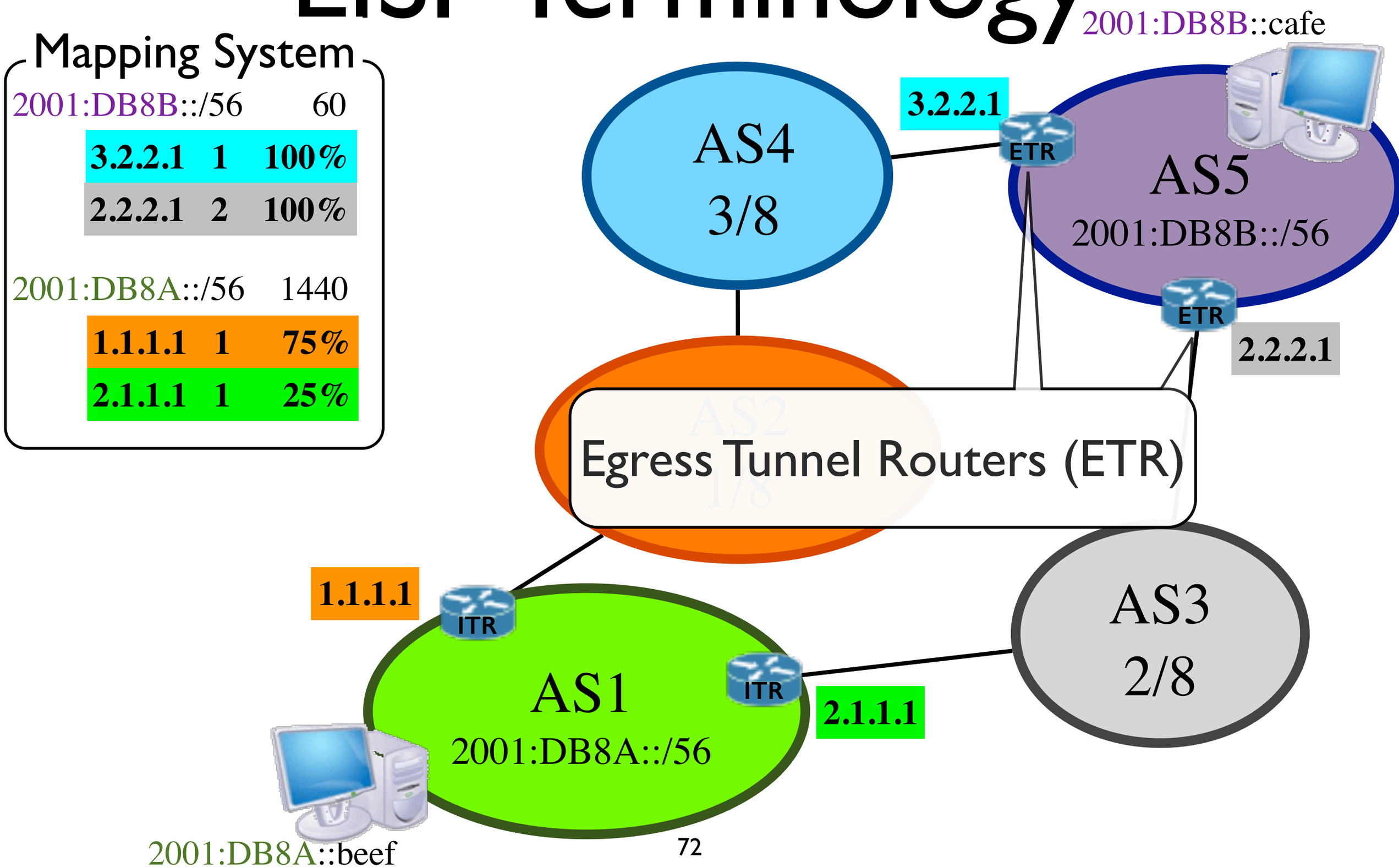
LISP Terminology



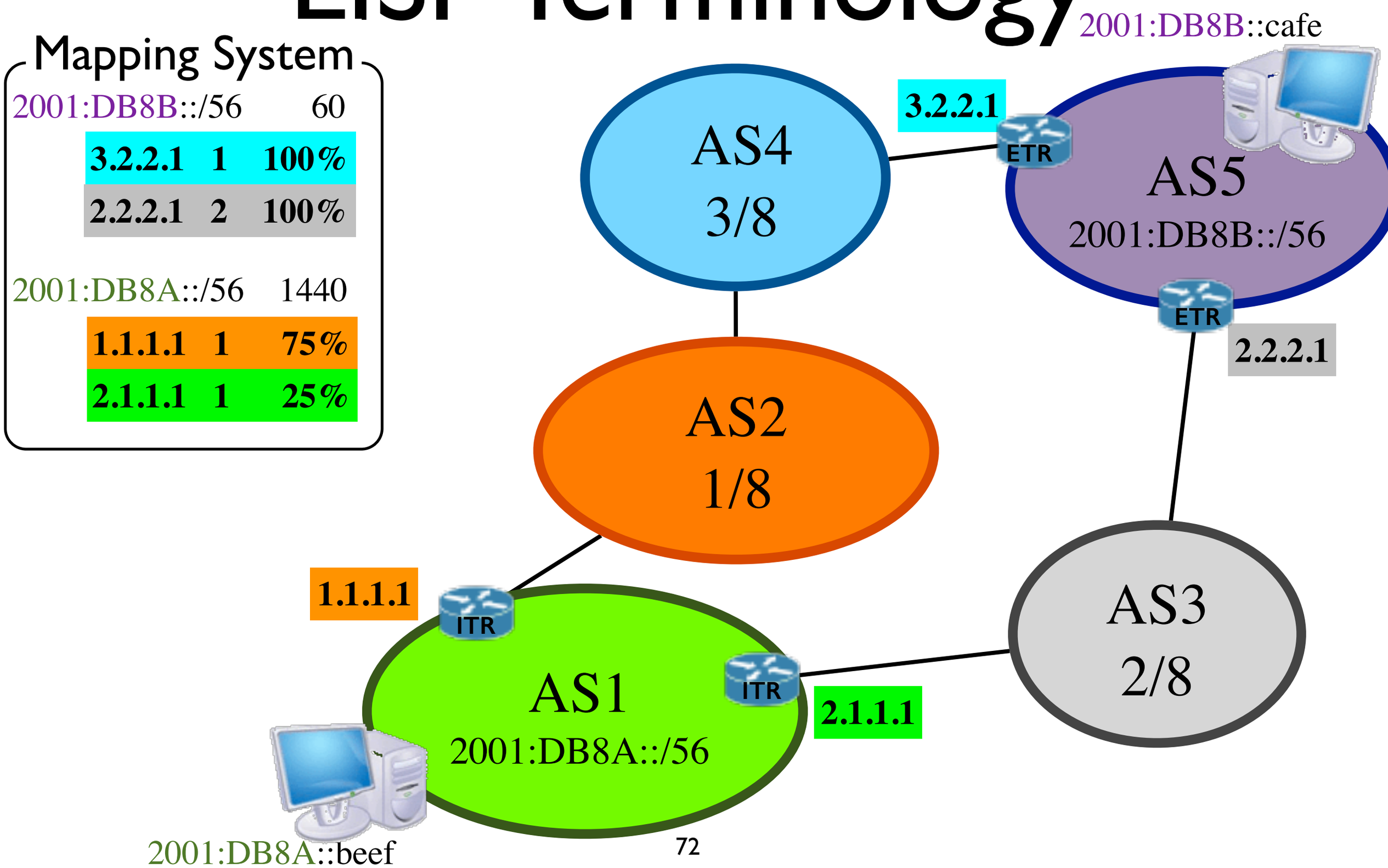
LISP Terminology



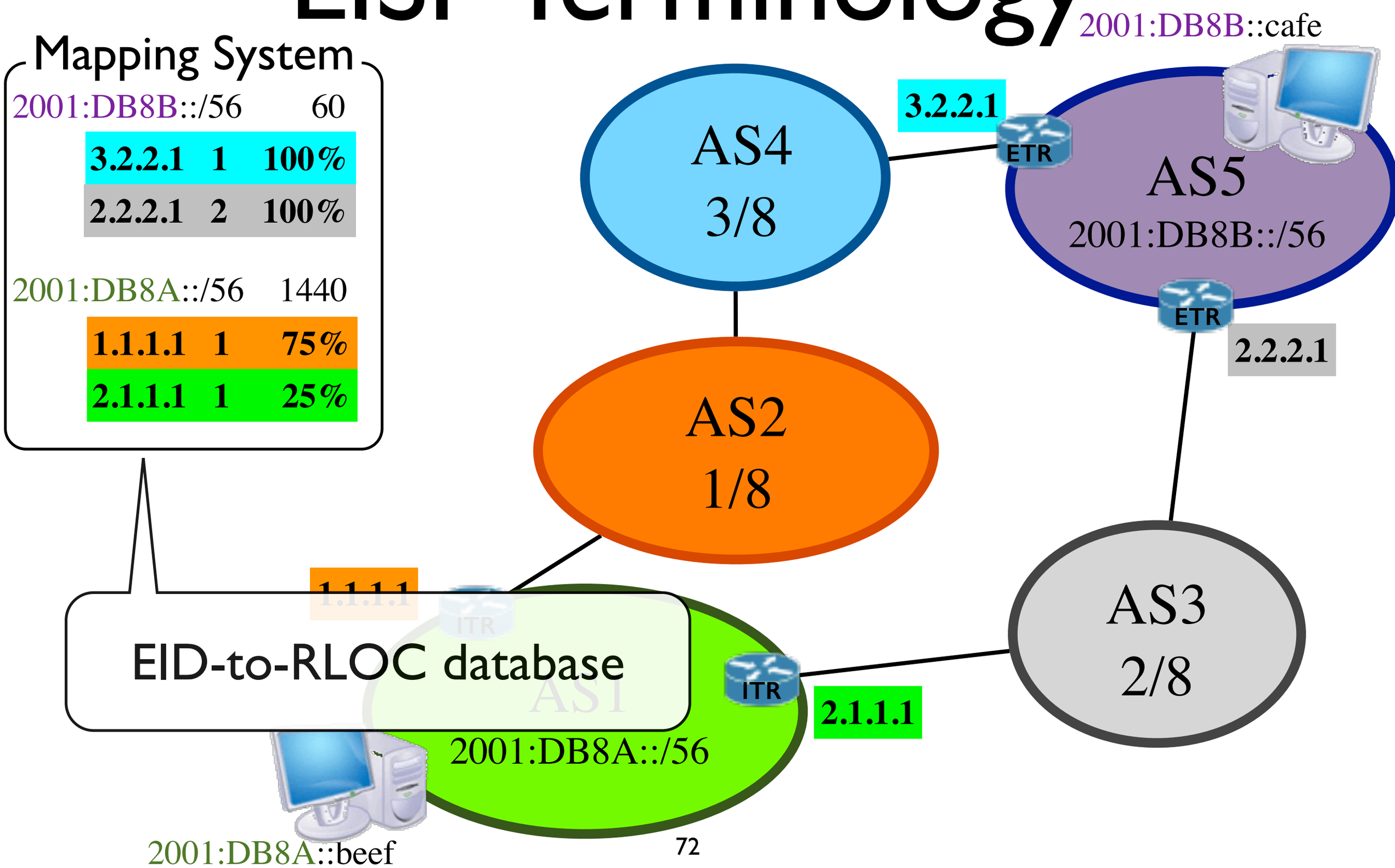
LISP Terminology



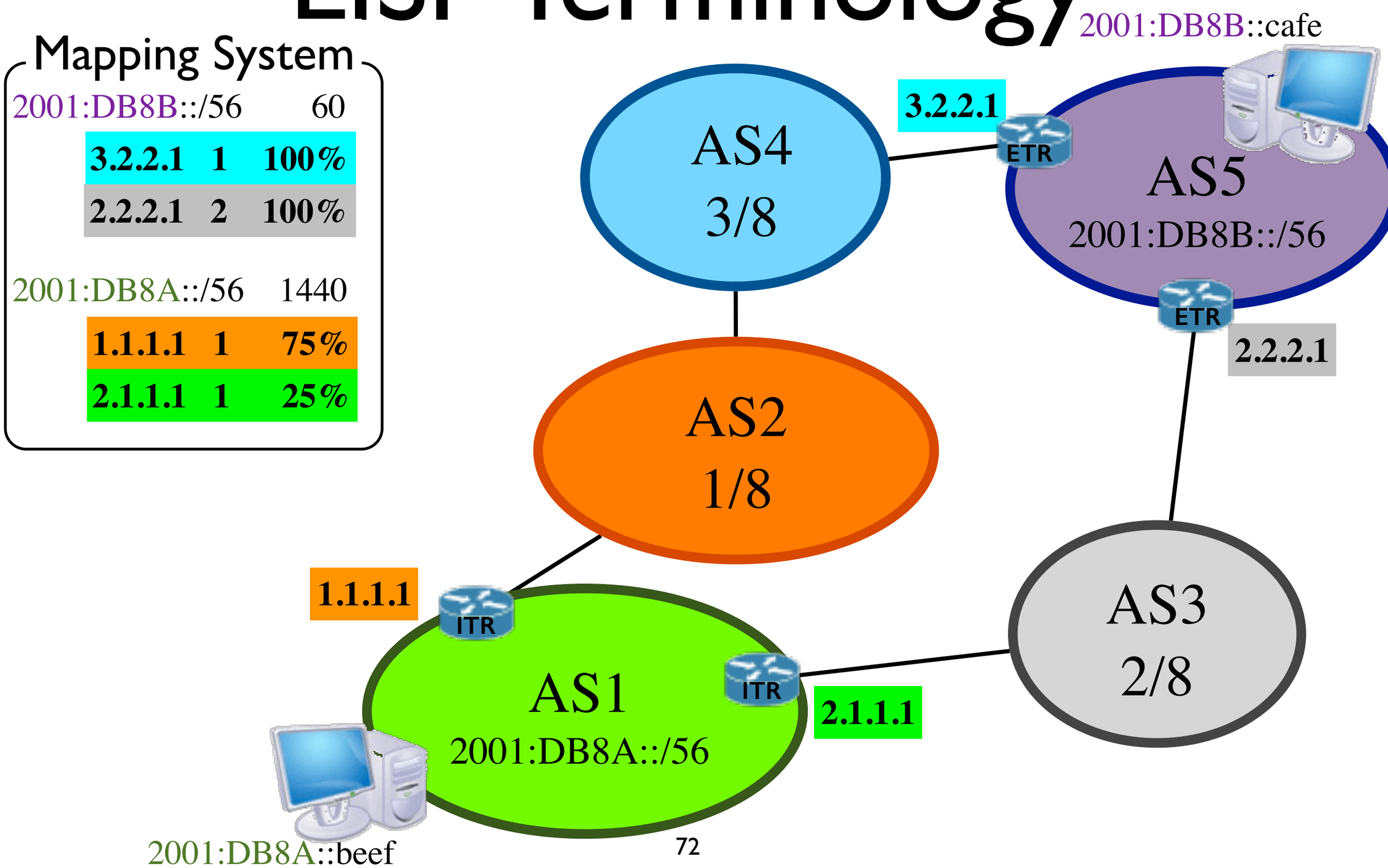
LISP Terminology



LISP Terminology



LISP Terminology



How has LISP been obtained?

LISP Main Design Goals

- No end-systems (hosts) changes
- Minimize required changes to the Internet infrastructure and the number of routers which have to be modified
- No router hardware changes and minimize router software changes
- Be incrementally deployable

No end-systems (hosts) changes

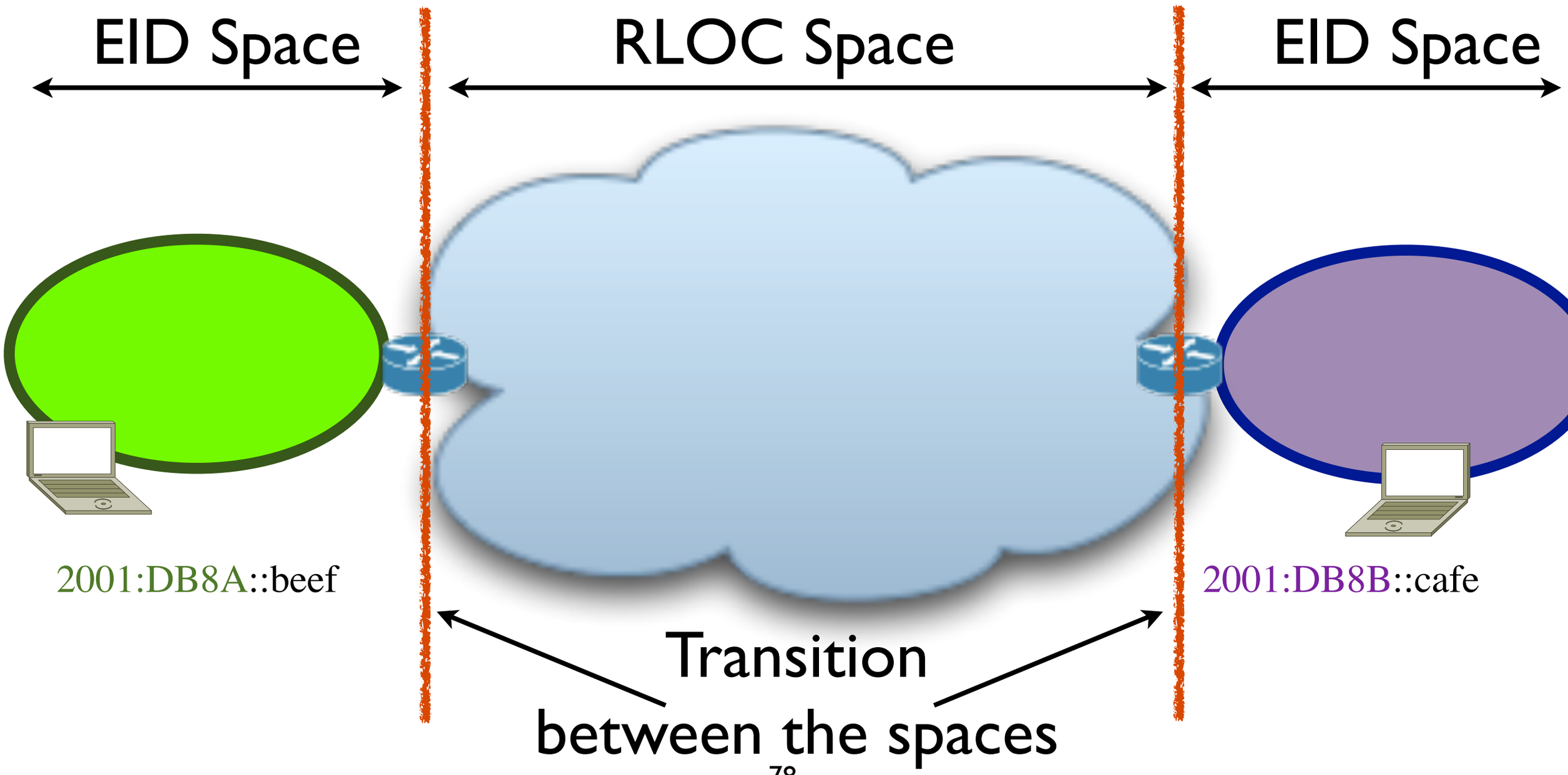
Network-based solution

- Identifiers are pure IP addresses (v4 or v6)
- Work performed by routers
- Encapsulation based protocol (vs rewriting)
  - Packets received at the destination are the same as those sent by the source (no address/port/... modification)

➡ Transparent for end-hosts

Minimize required changes to the Internet infrastructure and the number of routers which have to be modified

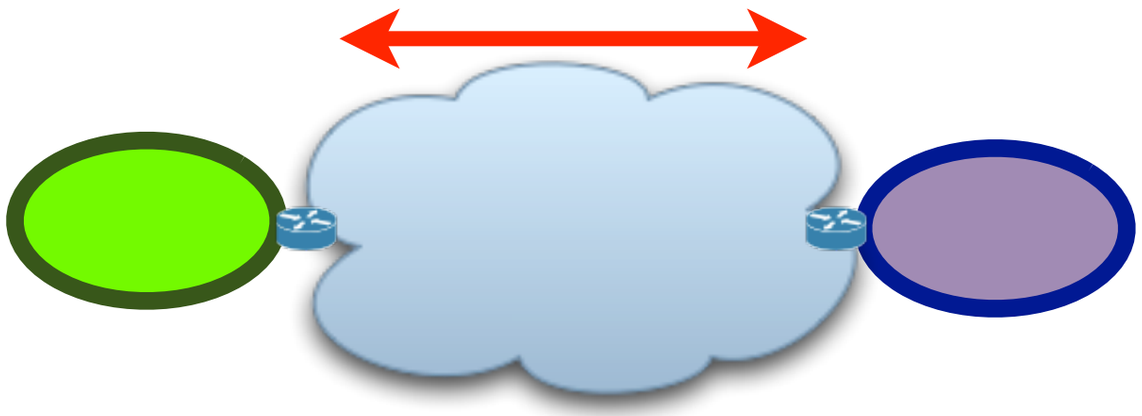
Split the Internet in two spaces



RLOC space

- Composed of the current transit networks
  - Keep IP and BGP as-is
  - RLOCs globally routable
  - EIDs are invisible here

➡ No change in the core



EID space

- Composed of the current stub networks
  - EIDs are routable (IP) within their stub
  - RLOCs and EIDs of other stubs are unknown in the stub
    - IGP with default route(s) to the border routers
    - Packets to distant EIDs simply follow the default route(s)

➡ No change in the stubs

Interactions between the two spaces

- LISP: the glue between the EID and RLOC spaces
- Modify border routers to support LISP functionality (xTR)
  - No change before/after the border
- Border routers do not advertise stub prefixes (EID) anymore to the Internet

Internet is more than routers

- NATs
- Firewalls
- Obscure equipments
- How to be confident the packets will not be lost somewhere?

Internet is more than routers

- NATs
- Firewalls
- Obscure equipments
- How to be confident the packets will not be lost somewhere?

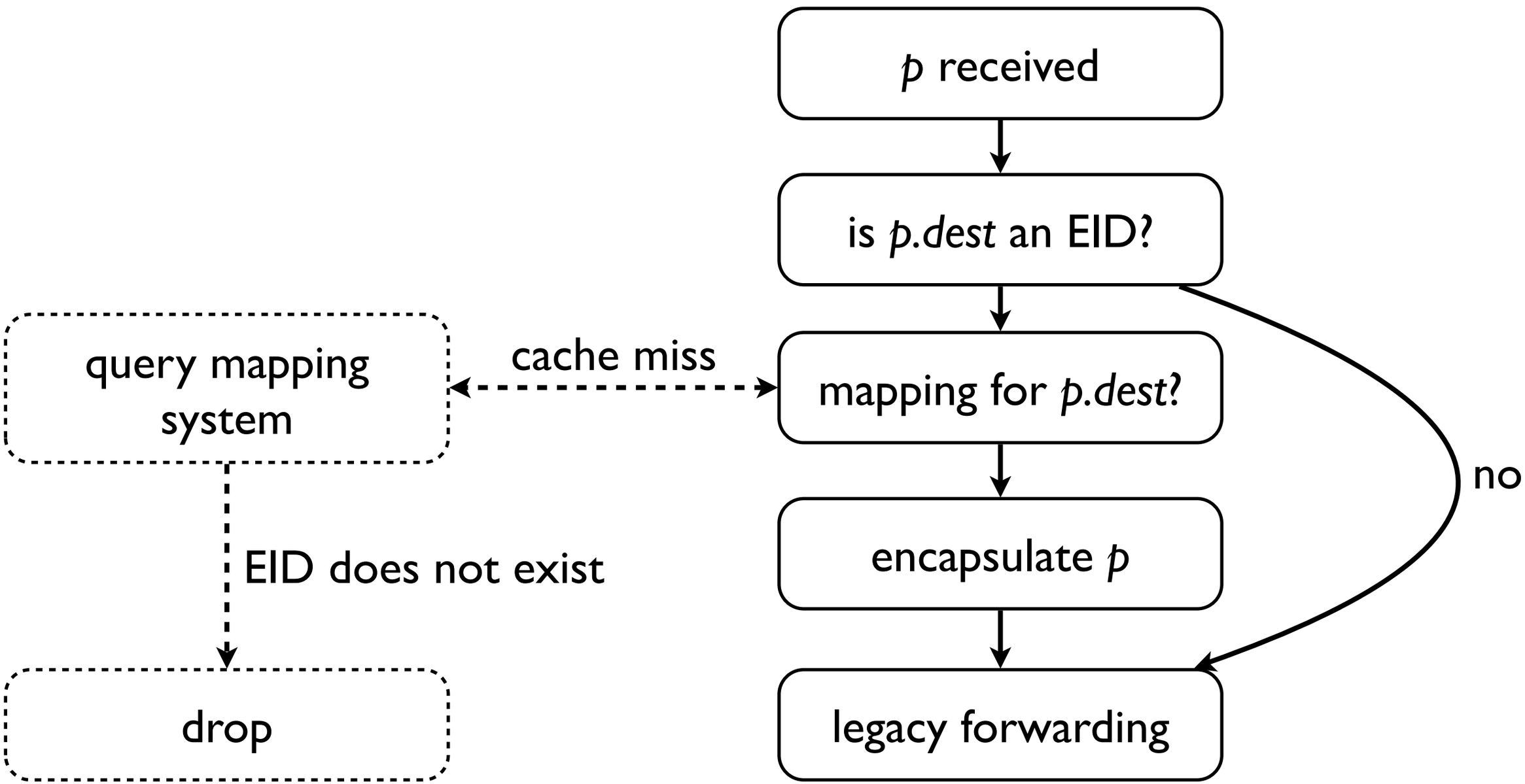
➡ Transport the new protocol over UDP

No router hardware changes and minimize router software changes

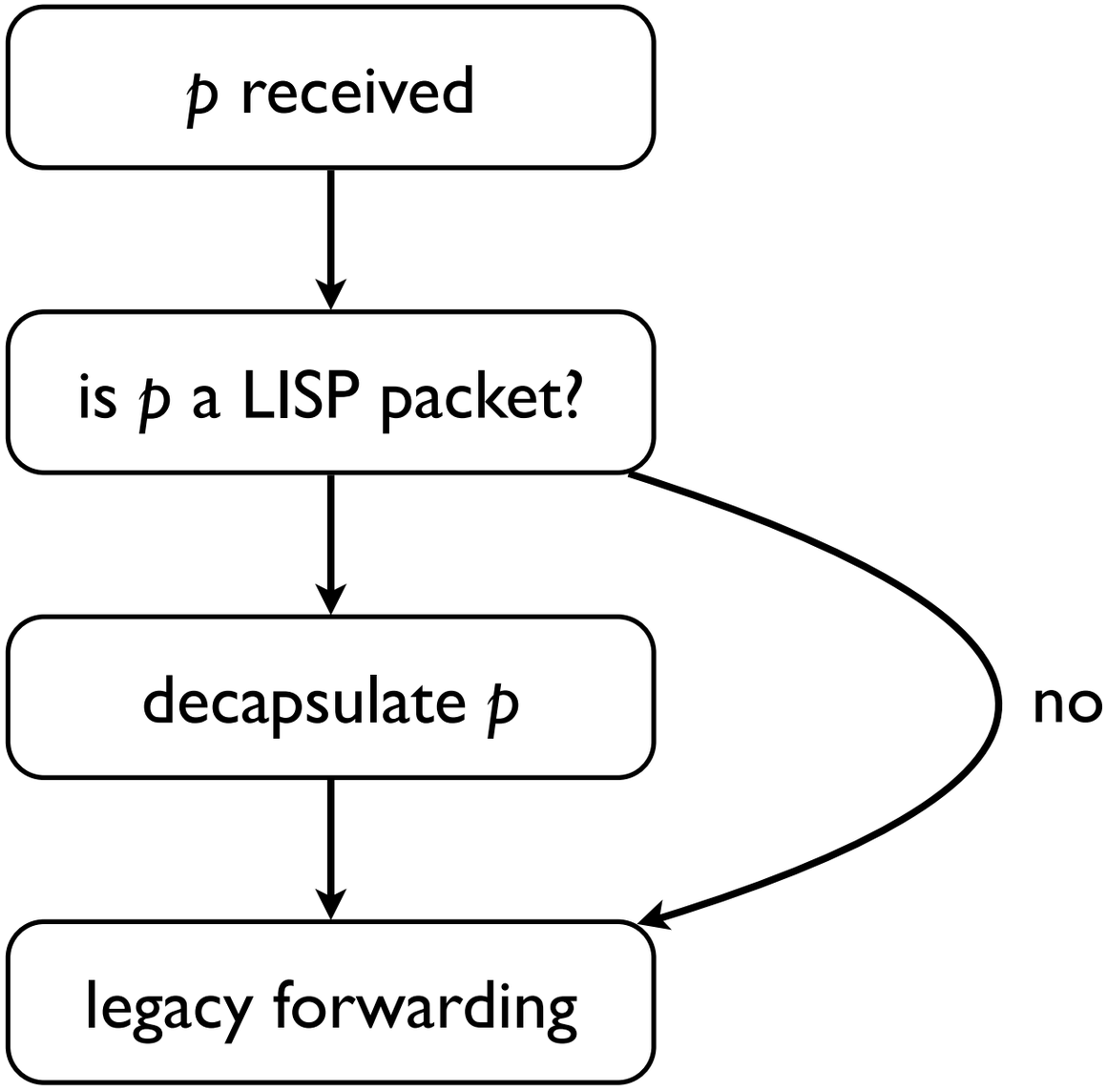
Use IP and UDP

- RLOCs and EIDs are pure IP packets
  - Any router implements IP
- EID prefixes follow CIDR
  - Any router implements longest prefix matching
- Transport LISP over UDP
  - Any router implements UDP

The life of a packet in an ITR



The life of a packet in an ETR



Be incrementally deployable

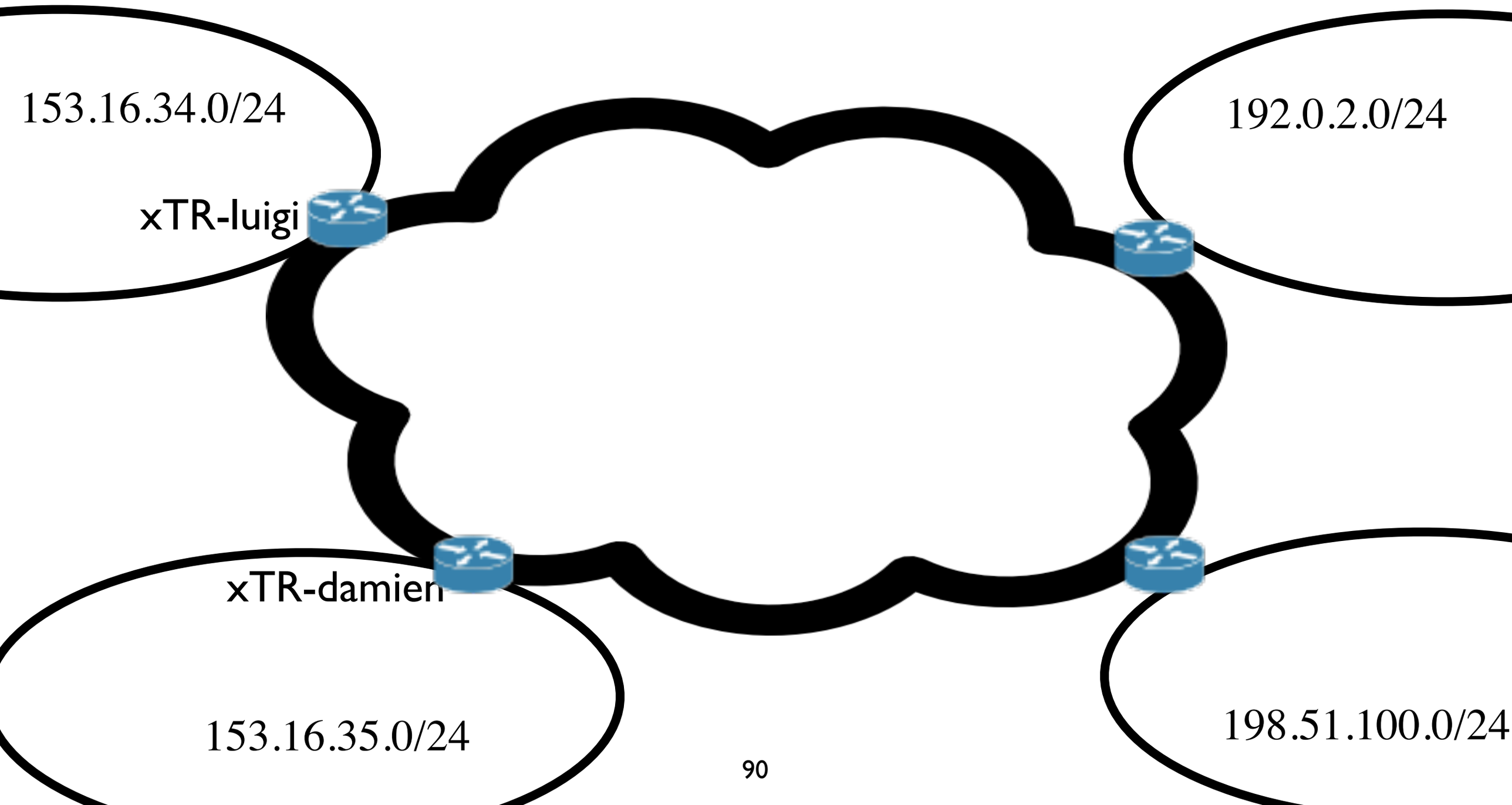
3 challenges

- non-LISP to non-LISP
- non-LISP to LISP
- LISP to non-LISP

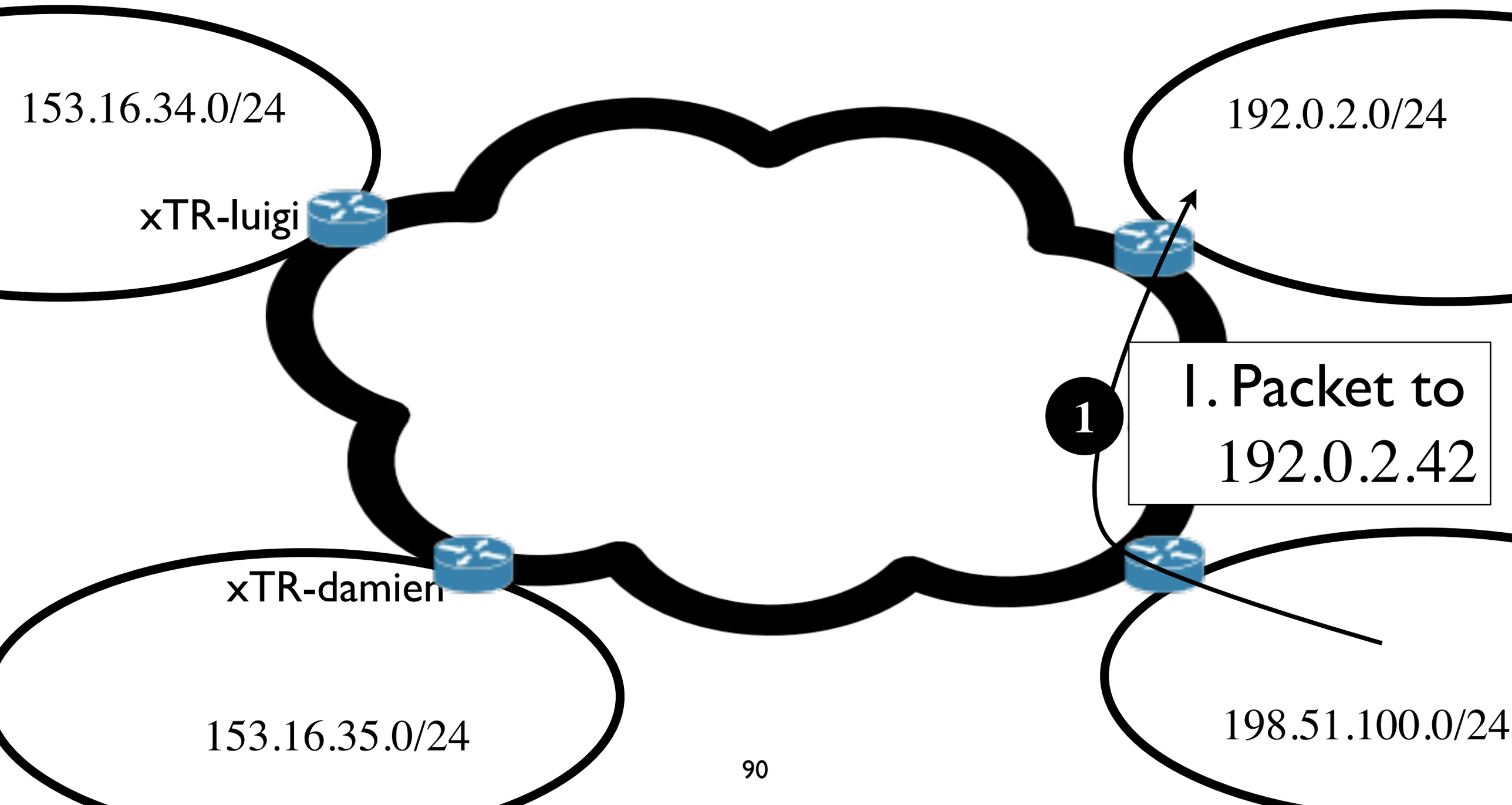
non-LISP to non-LISP

- LISP is not involved
  - Current Internet

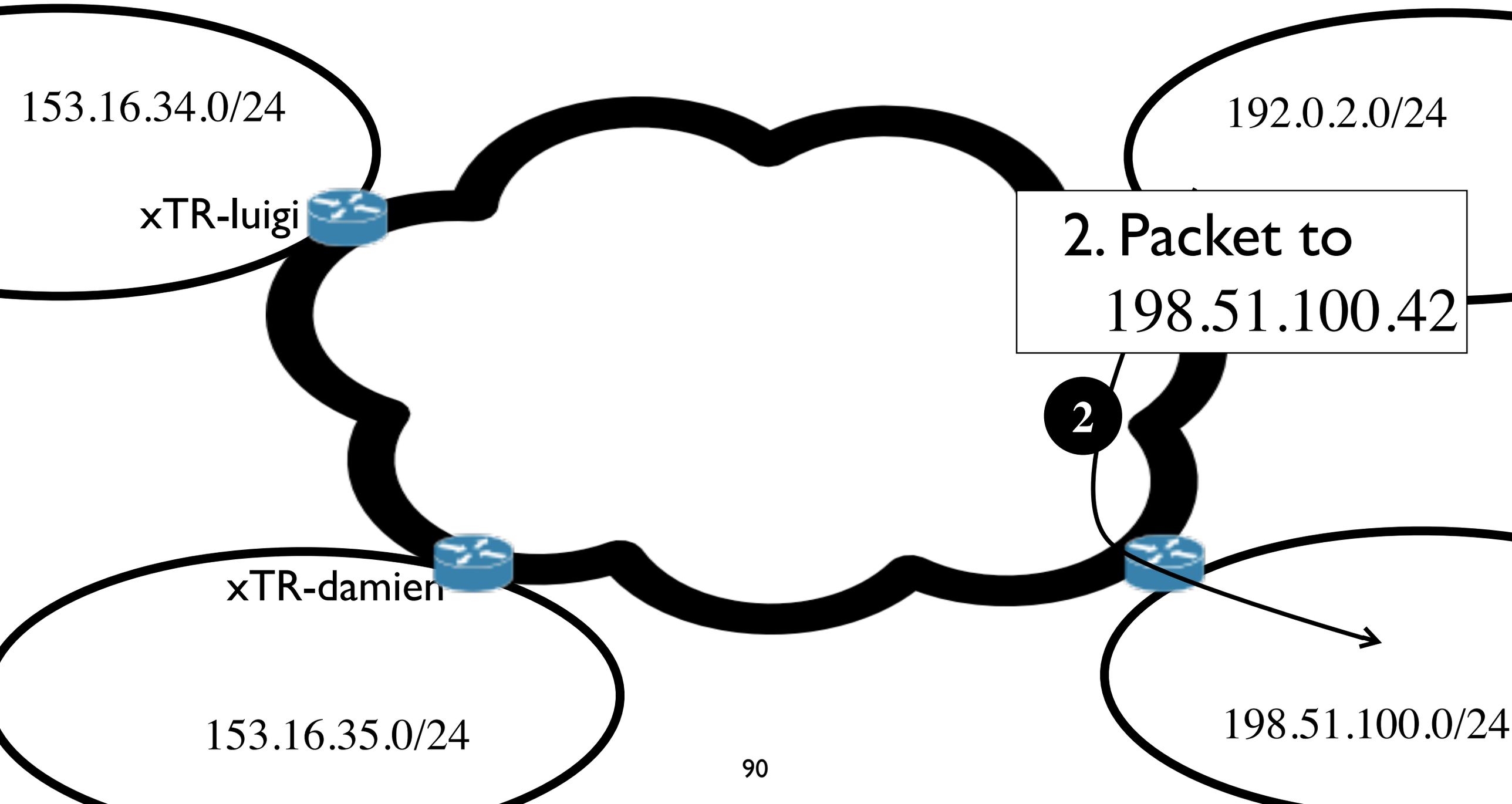
non-LISP to non-LISP



non-LISP to non-LISP



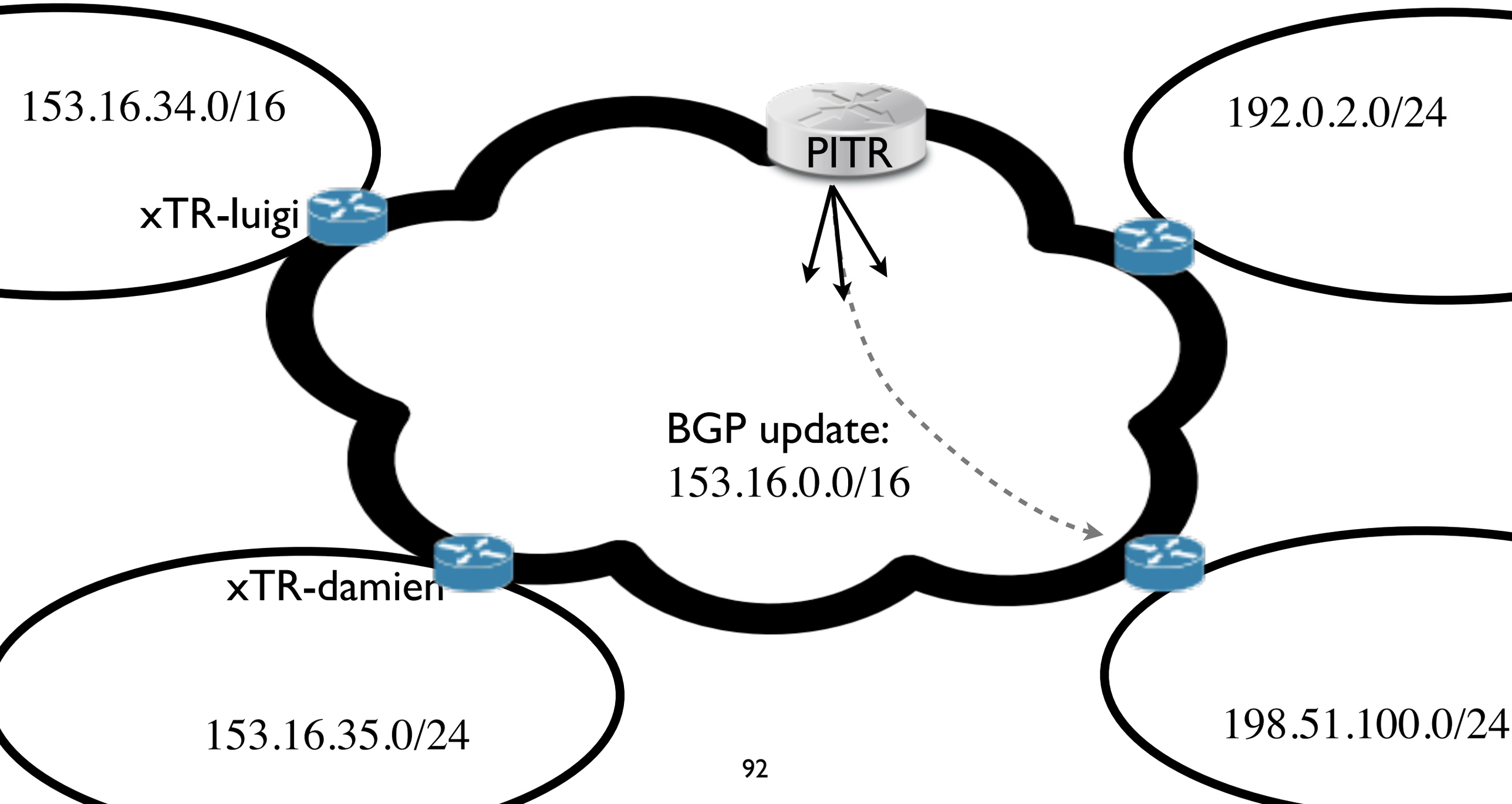
non-LISP to non-LISP



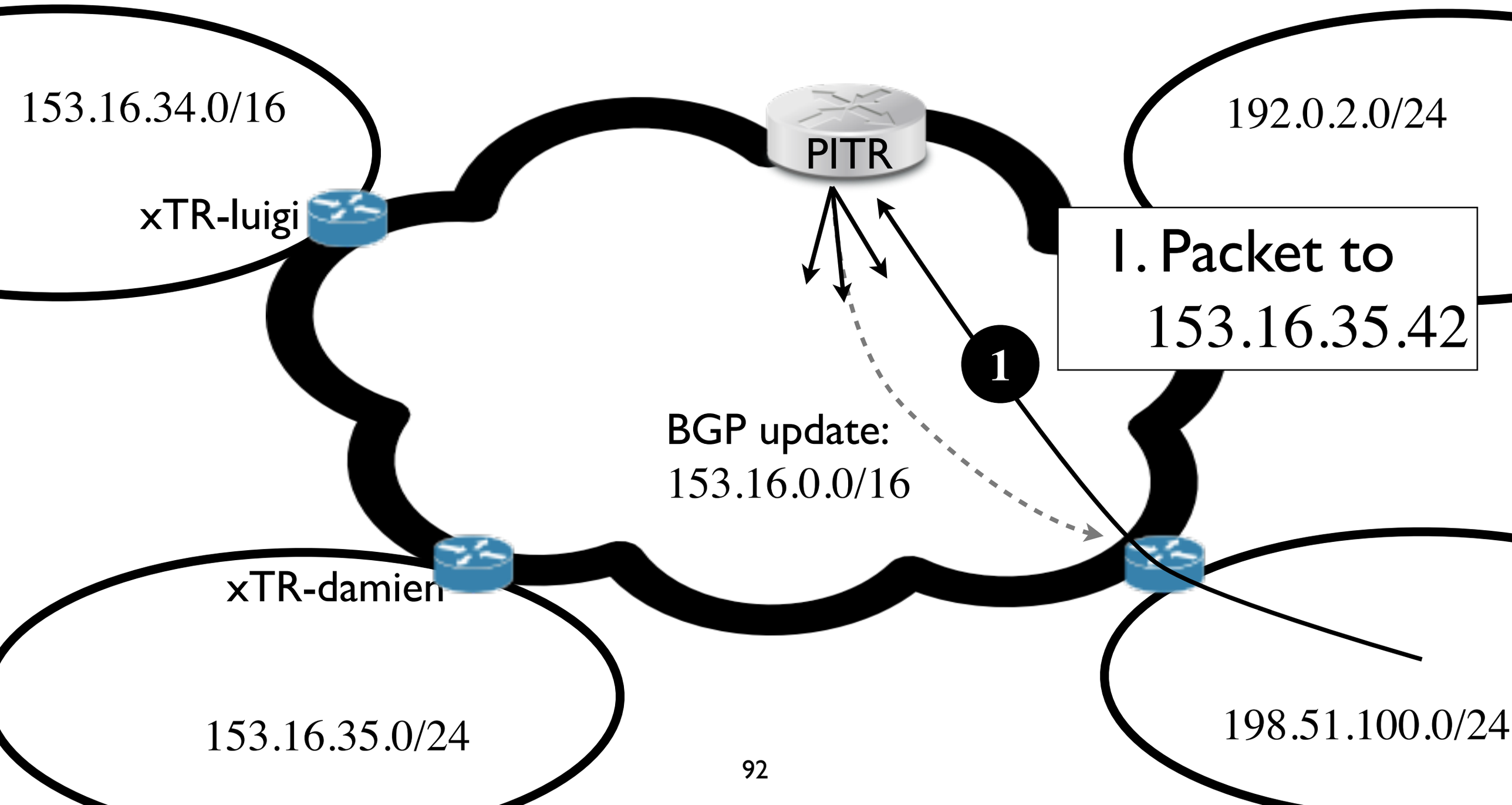
non-LISP to LISP

- Add a Proxy ITR (PITR) middle-box somewhere on the Internet
- PITR originates EID advertisement
  - The EID prefix becomes globally routable (!)
  - The PITR attracts traffic for the EID prefix
- Traffic with destination IP in the EID prefix are natively forwarded to the PITR
- The PITR acts as the ITR on behalf of non-LISP sites

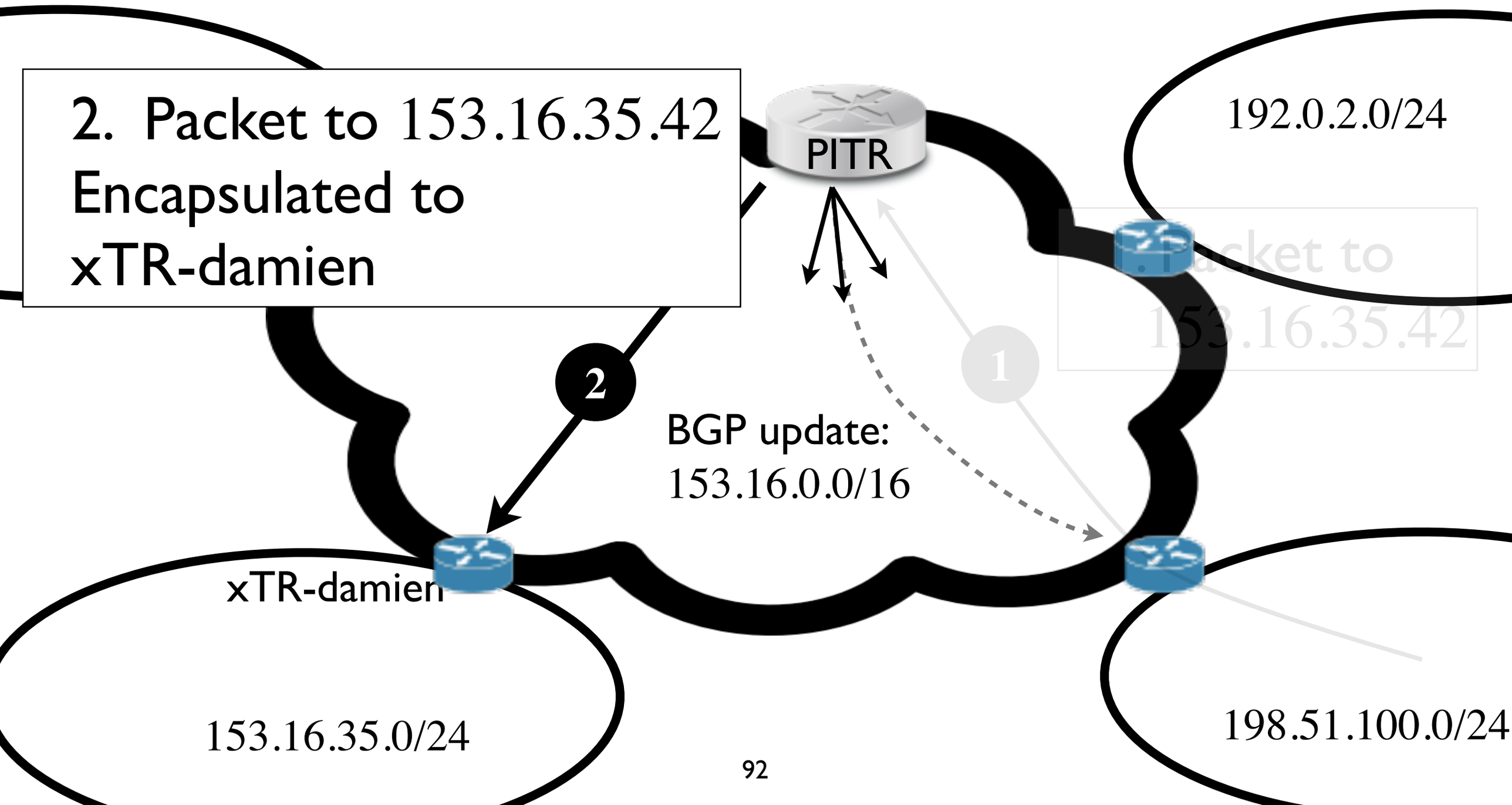
non-LISP to LISP



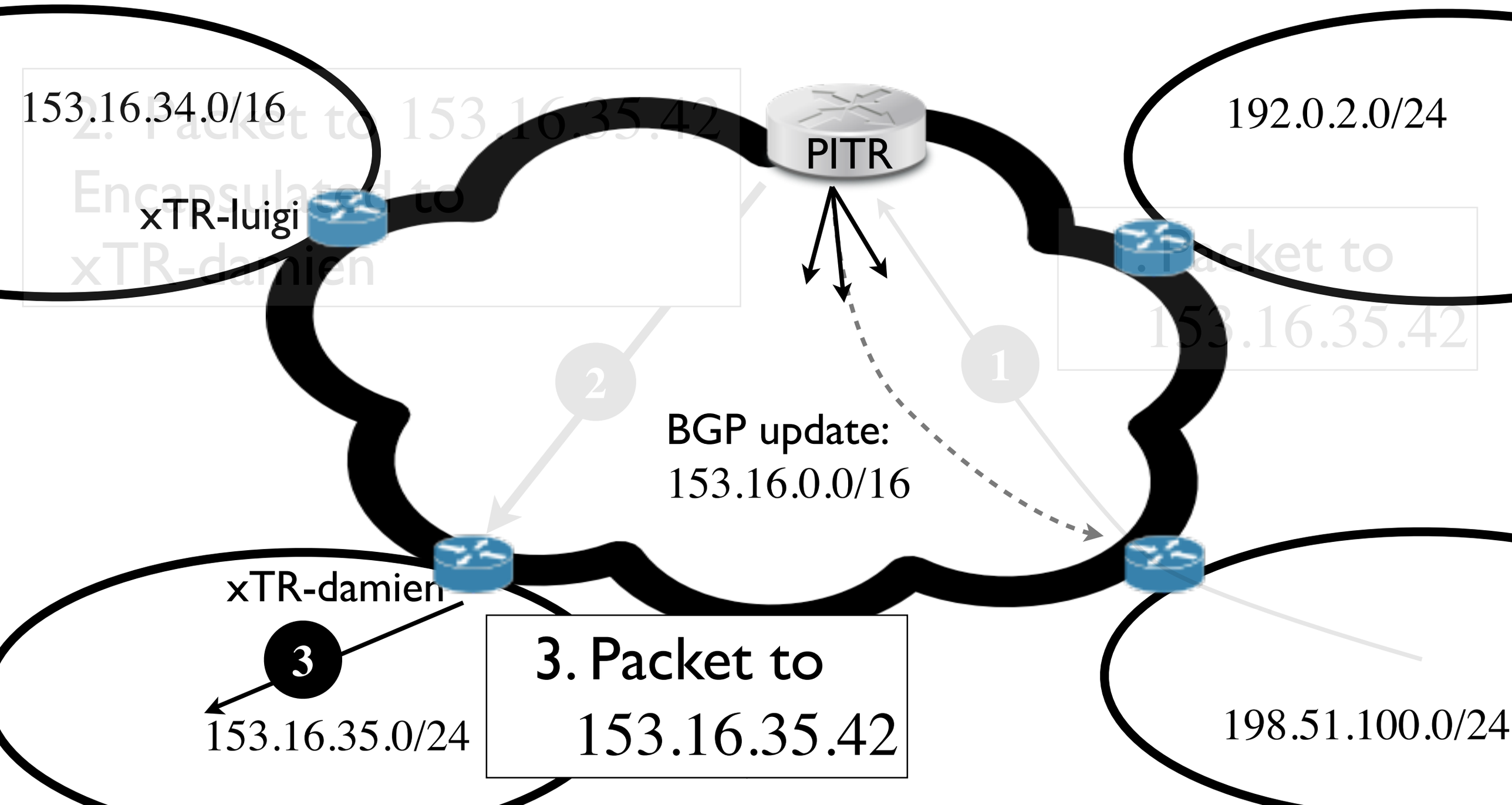
non-LISP to LISP



non-LISP to LISP



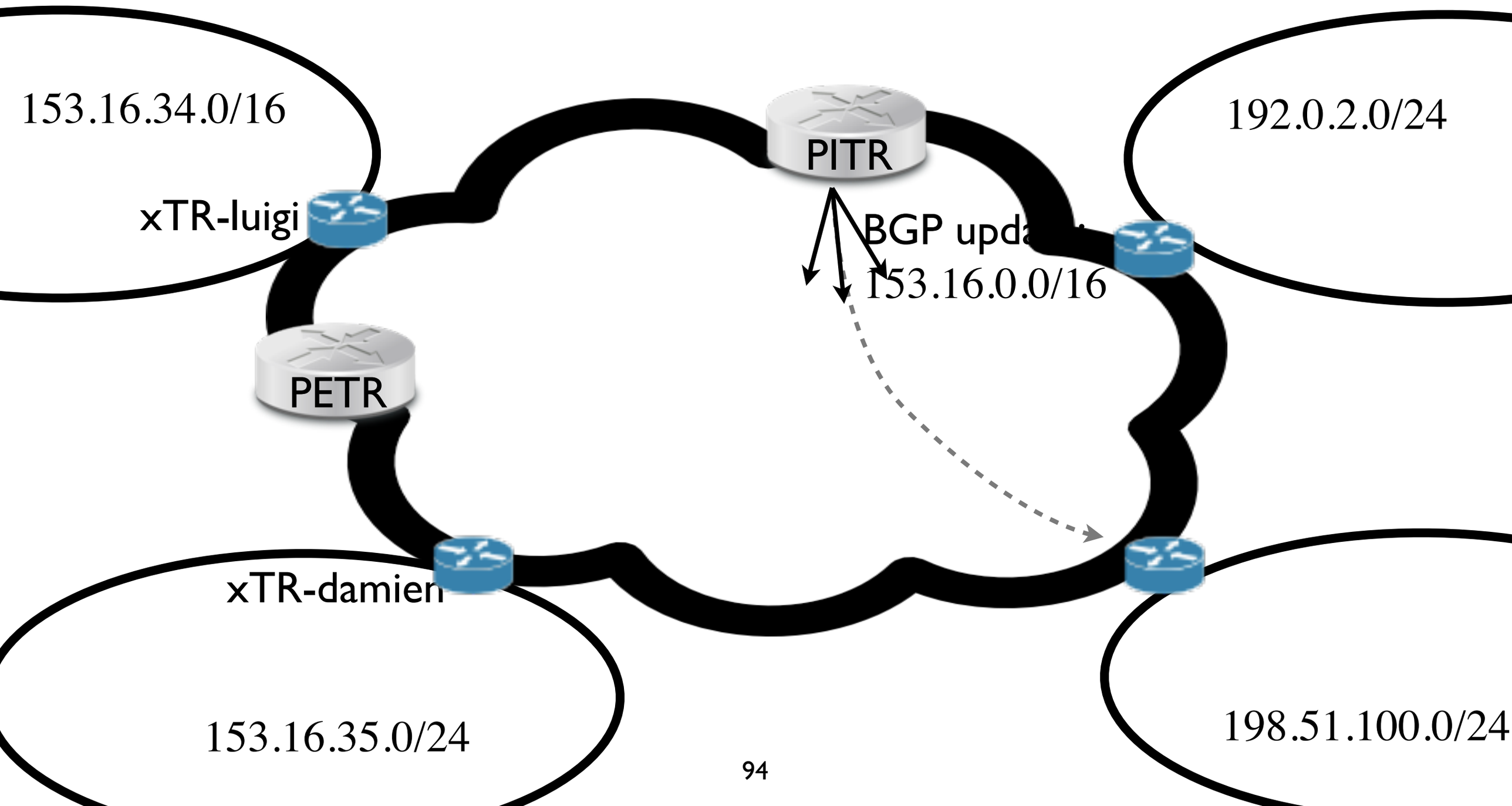
non-LISP to LISP



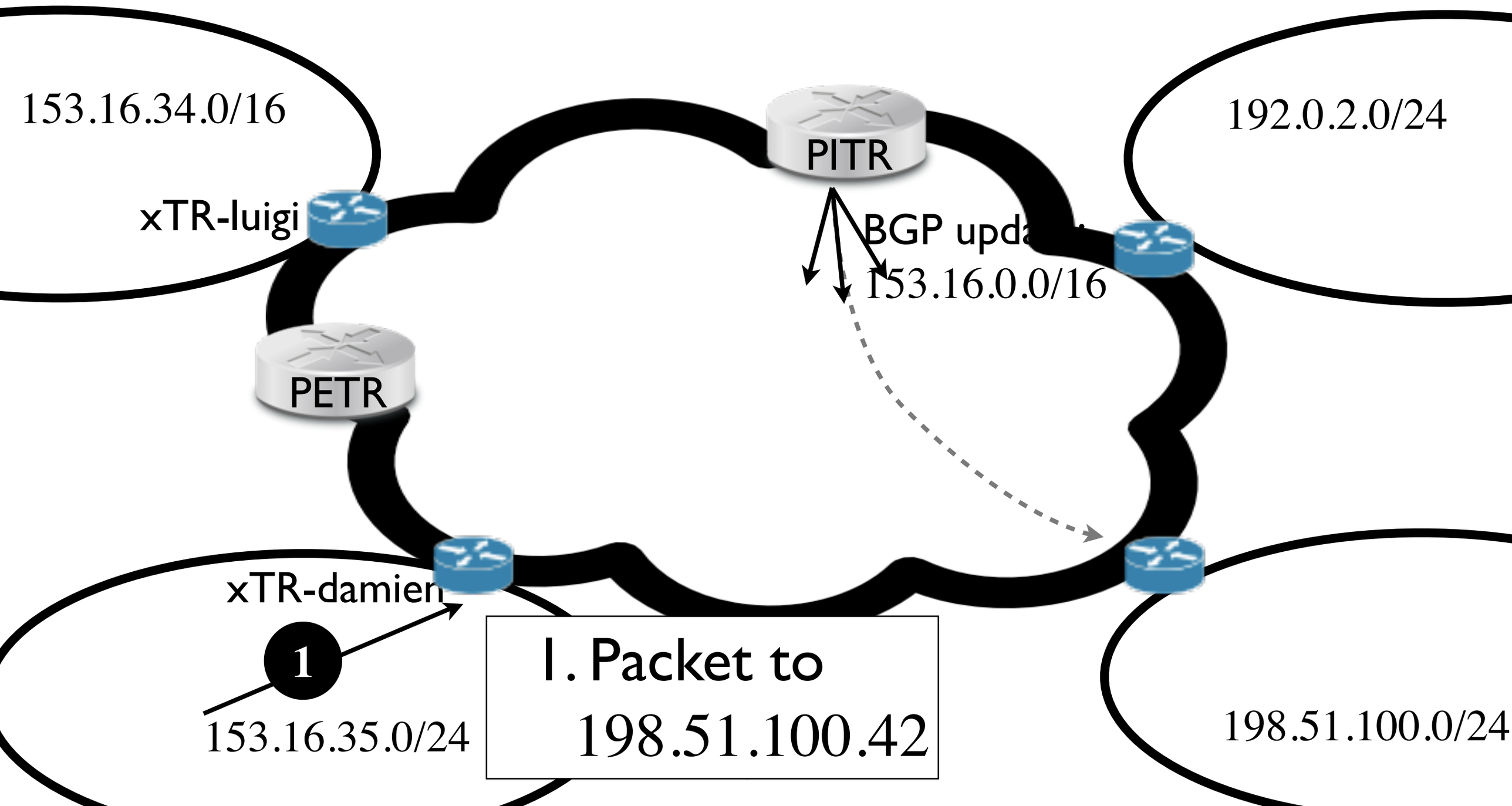
LISP to non-LISP

- Add a Proxy ETR (PETR) middle-box somewhere on the Internet
  - The PETR acts as the ETR on behalf of non-LISP sites
- EID and RLOC space are separated
  - The ITR does not encapsulate if the destination IP is not an EID
- If no PITR for the source, use LISP-NAT
  - LISP-NAT rewrites the non-routable EID source to a routable source and keeps the state for the reverse direction

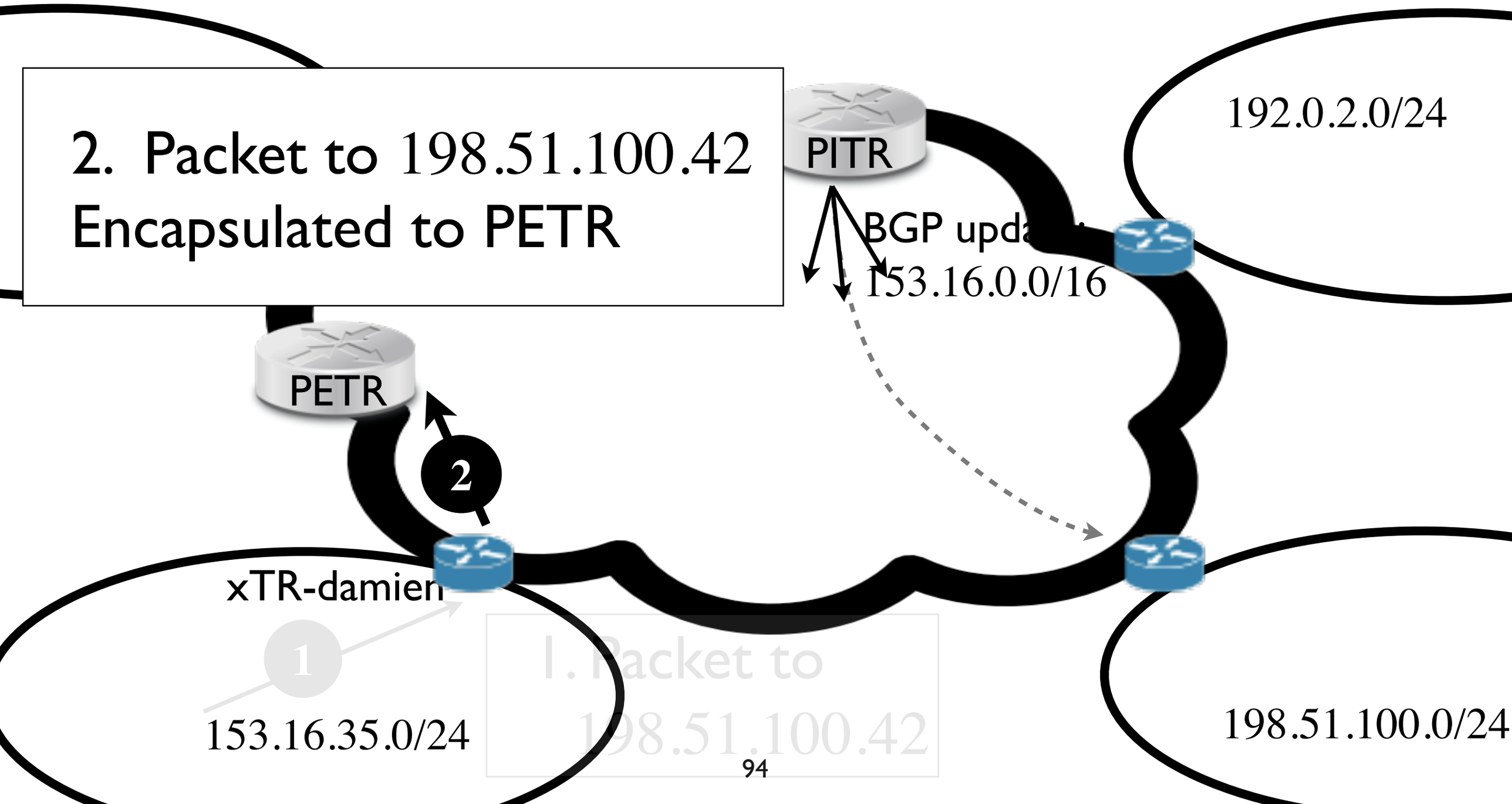
LISP to non-LISP



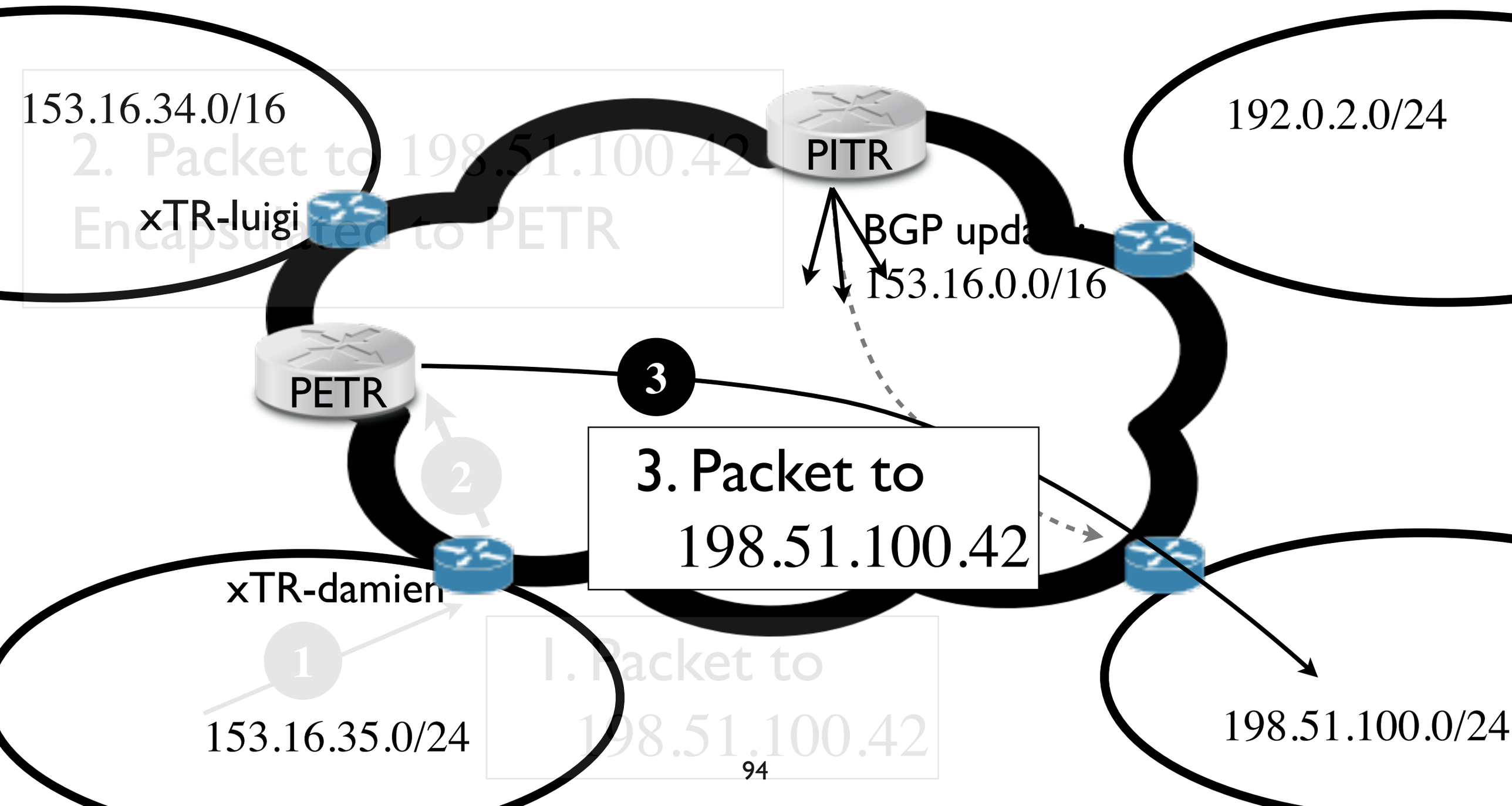
LISP to non-LISP



LISP to non-LISP



LISP to non-LISP



What about enterprise and datacenter networks?

Issues with datacenter and enterprise networks

- Enterprise and datacenter network rapidly become complex to manage with current protocols
  - networks can become very large (tens of thousands devices)
  - the number of features to be implemented at the network is big (e.g., security, traffic optimisation…)
  - network entities are now mobile (e.g., tablets, virtual machines…)
  - complex policies can have to be implemented directly in the network (e.g., users must see the same network wherever they are connected)

Issues with datacenter and enterprise networks (contd.)

- Networks are managed by configuration
  - hard to construct configurations that support all the possible cases
  - each protocol has its own set of configuration
  - impossible to react to sudden changes (e.g., earthquake that disrupt a large portion of the network)
- No abstraction
  - need to know the very details of the topology (e.g., link capacity, IP addresses…)
  - need a deep understanding of the deployed protocols and their interactions

Software Defined Networking

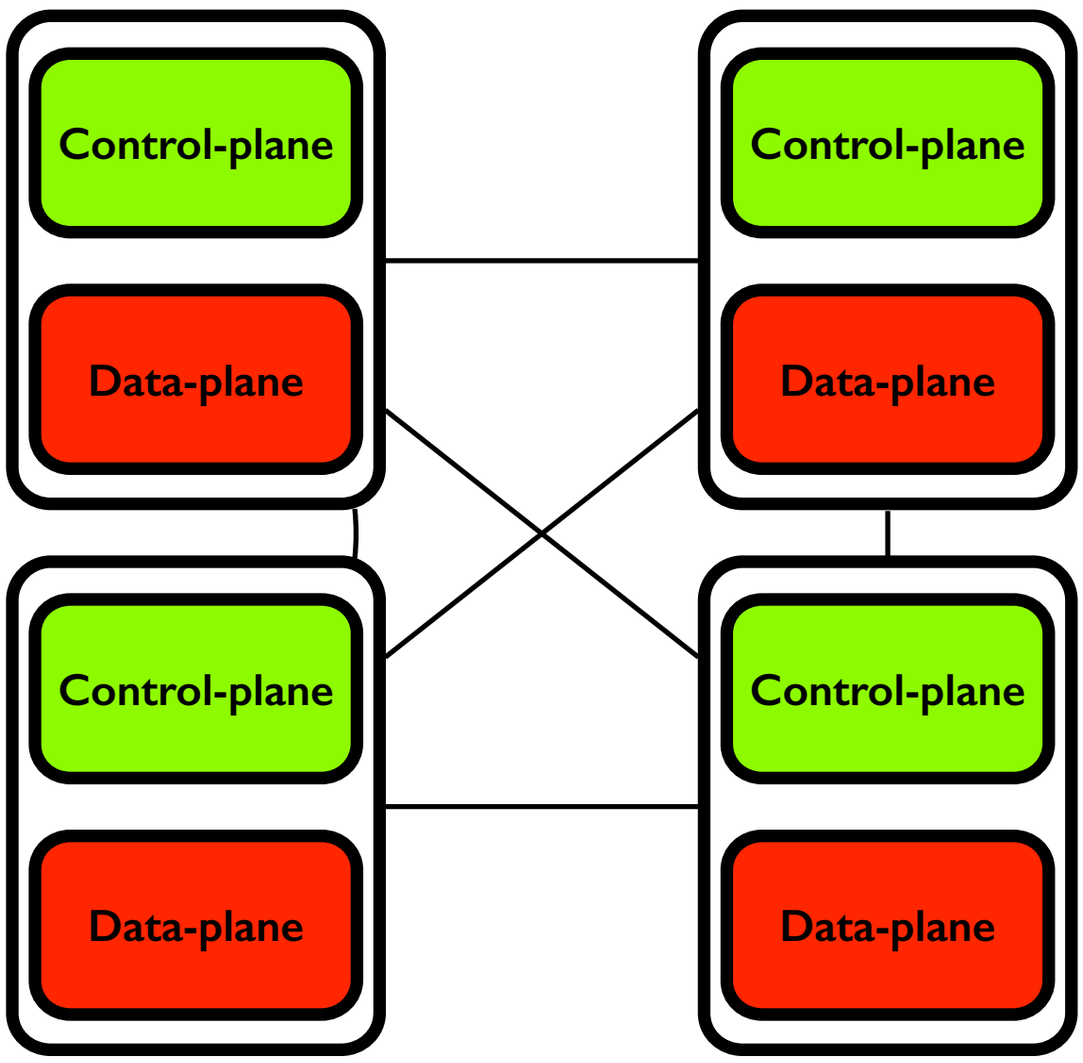
- Concept: conceive the network as a program
  - operators do not configure the network, they program it
  - operators do not interact directly with devices
    - network logic is implemented by humans but network elements are never touched by humans

Software Defined Networking (contd.)

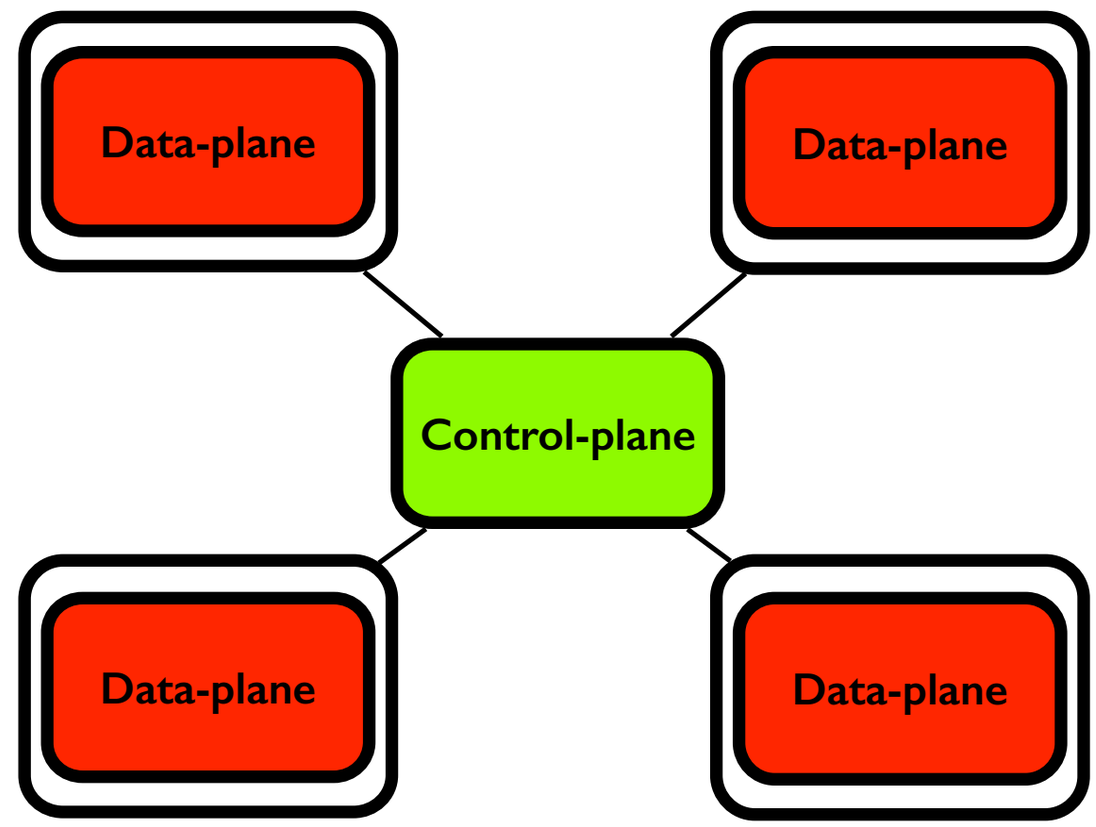
- Programmability of network is reach by decoupling control plane from data plane
- OpenFlow makes this distinction where network elements are only constituted of elementary switches that are remotely commanded by a logically centralised controller

OpenFlow in one slide

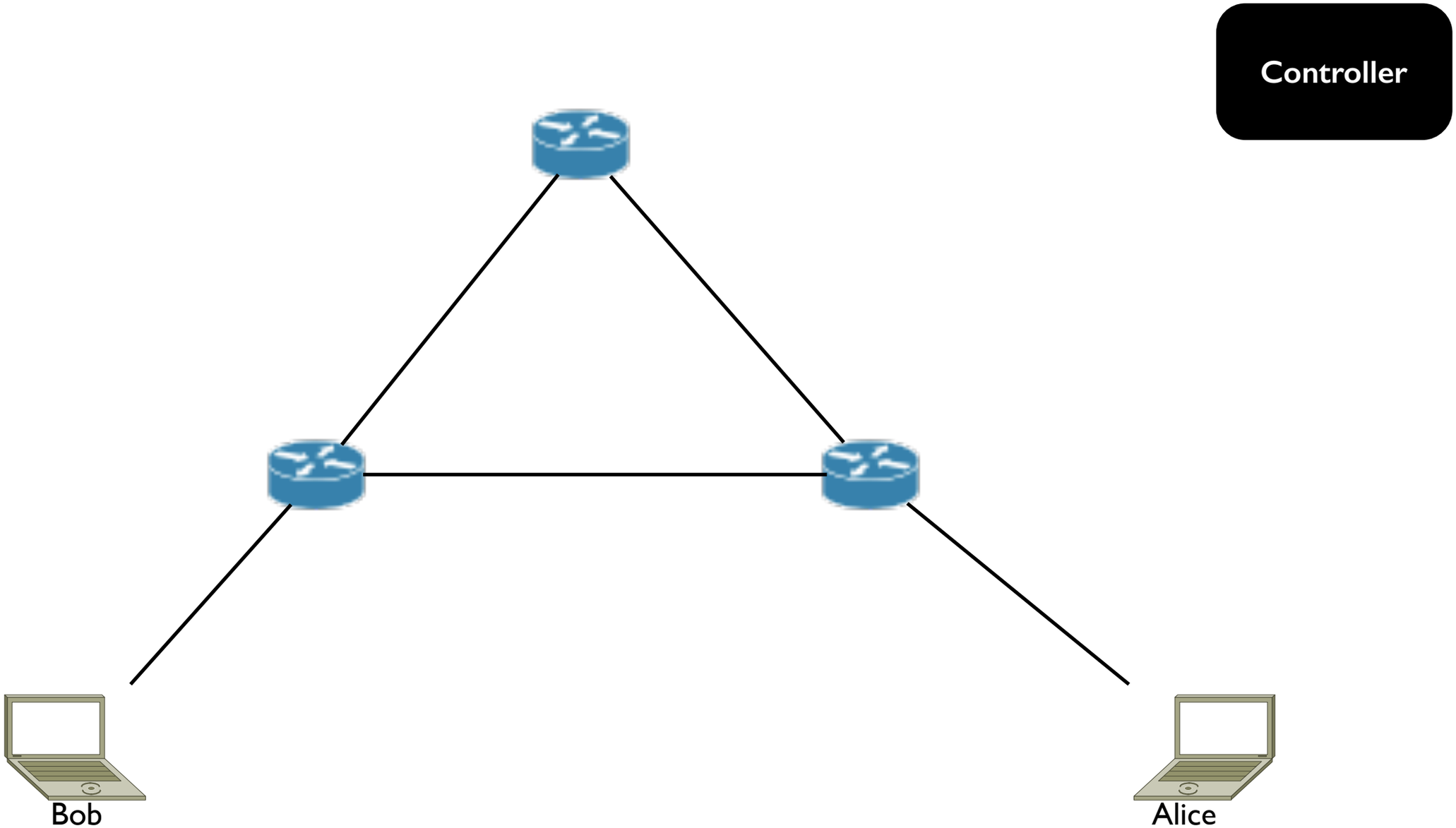
Traditional approach



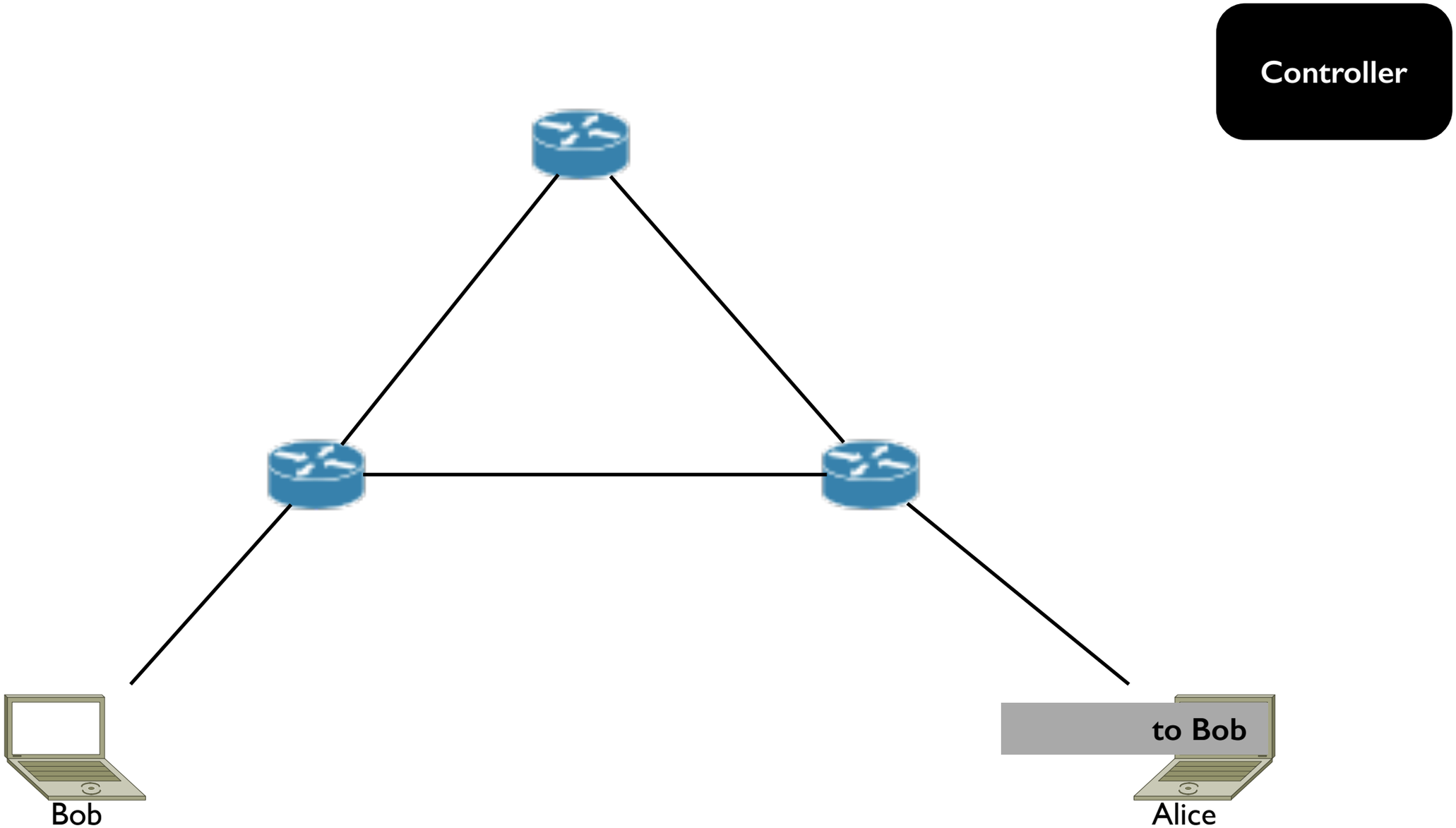
OpenFlow approach



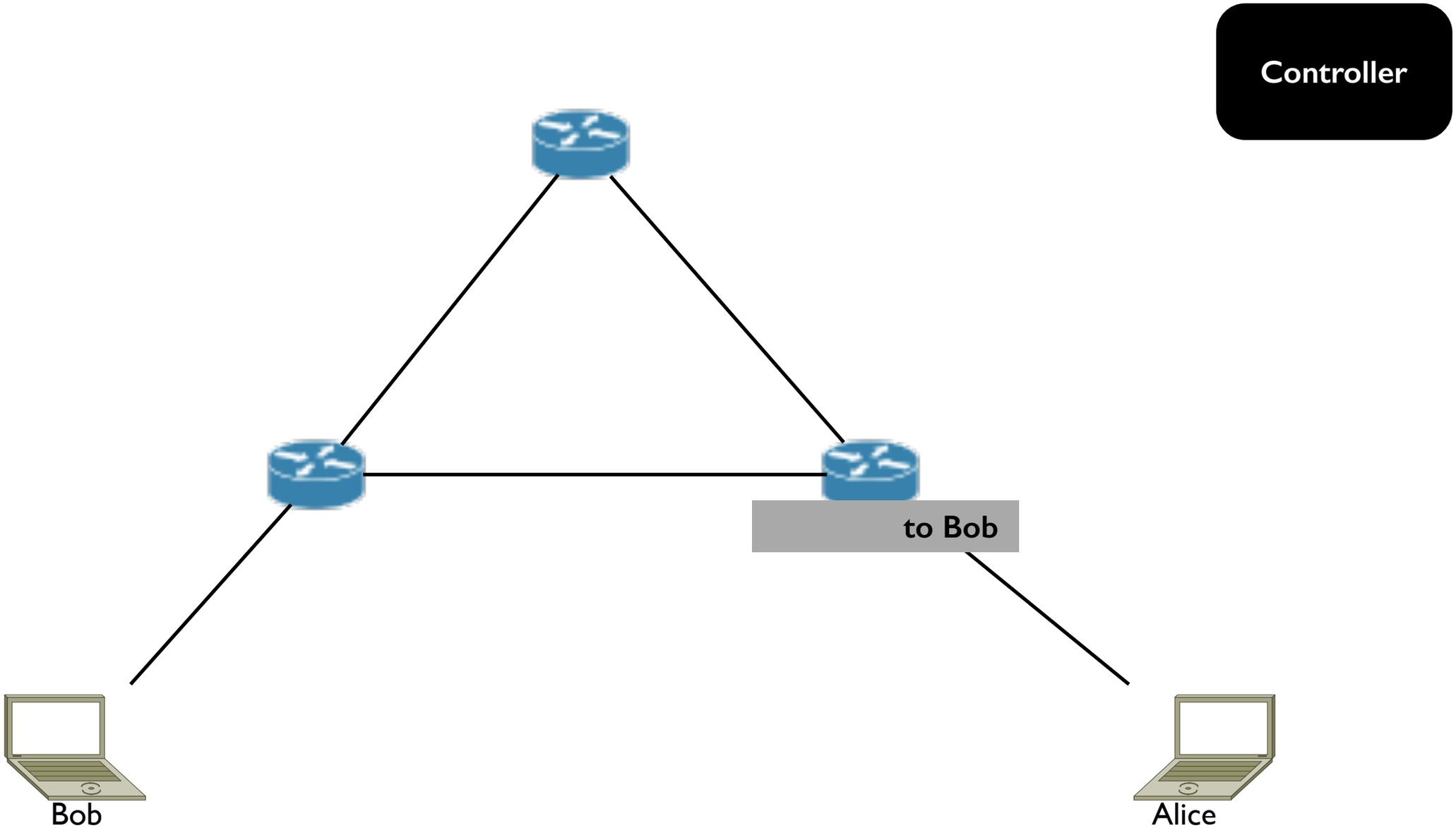
OpenFlow workflow



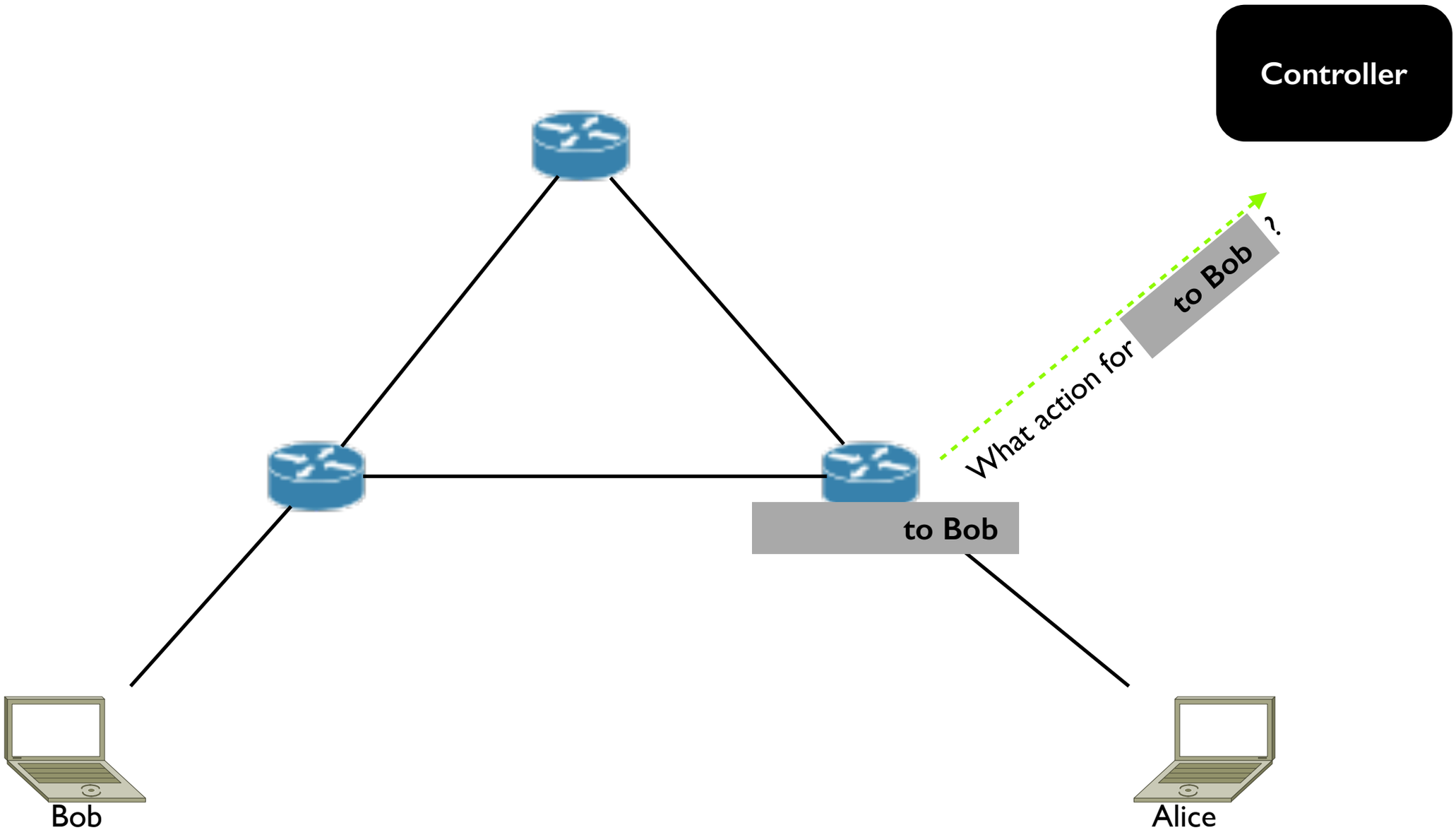
OpenFlow workflow



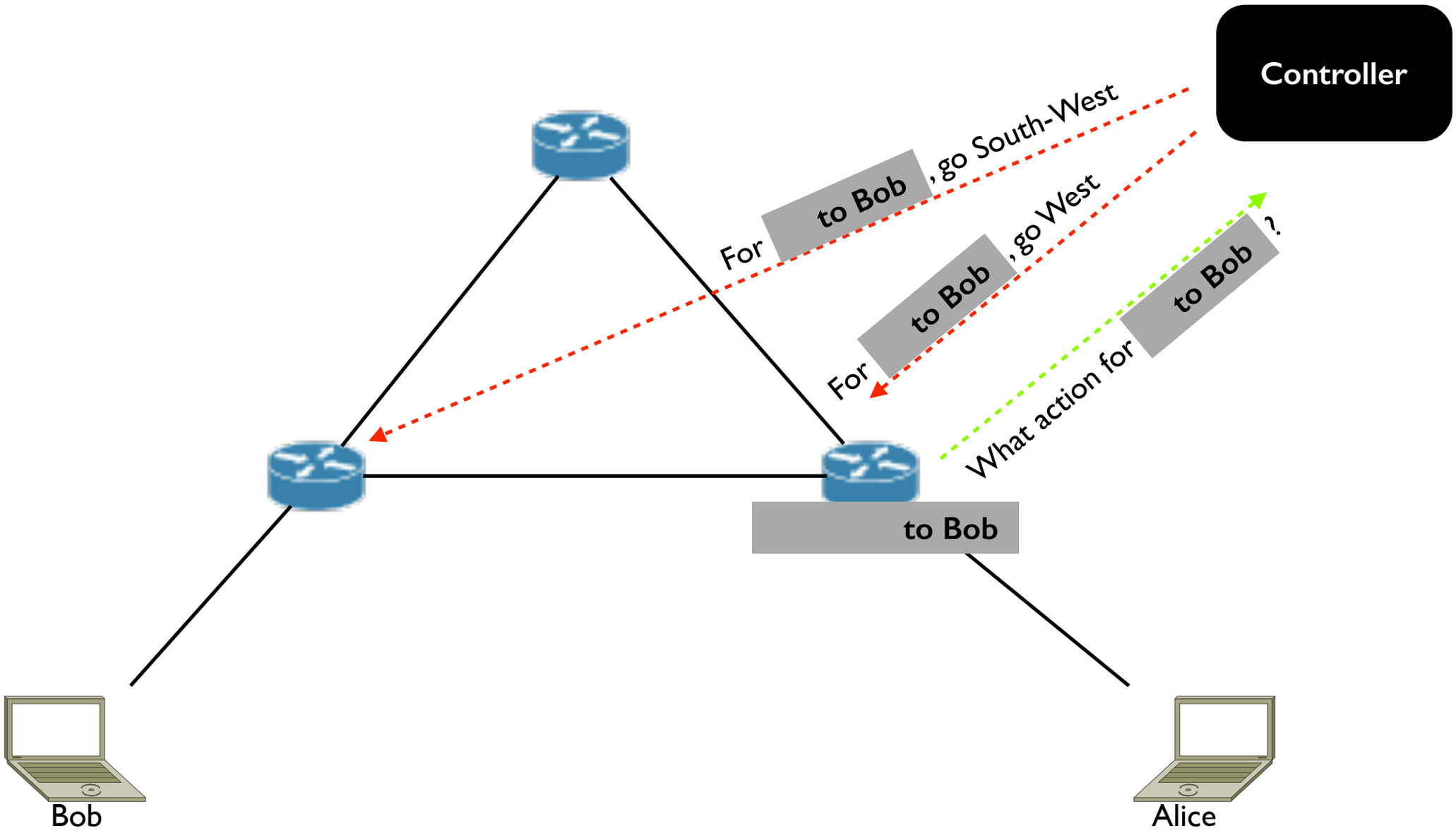
OpenFlow workflow



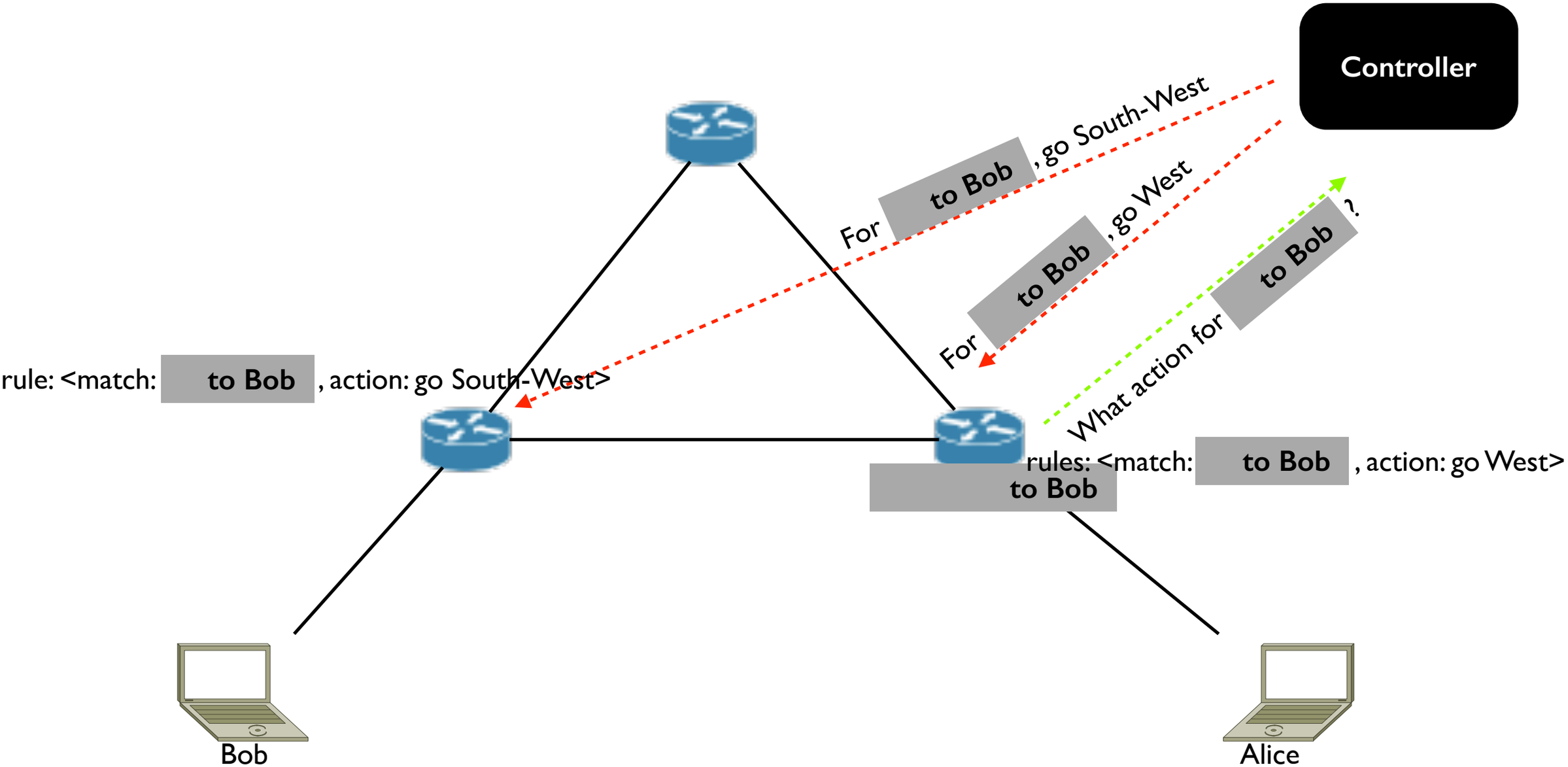
OpenFlow workflow



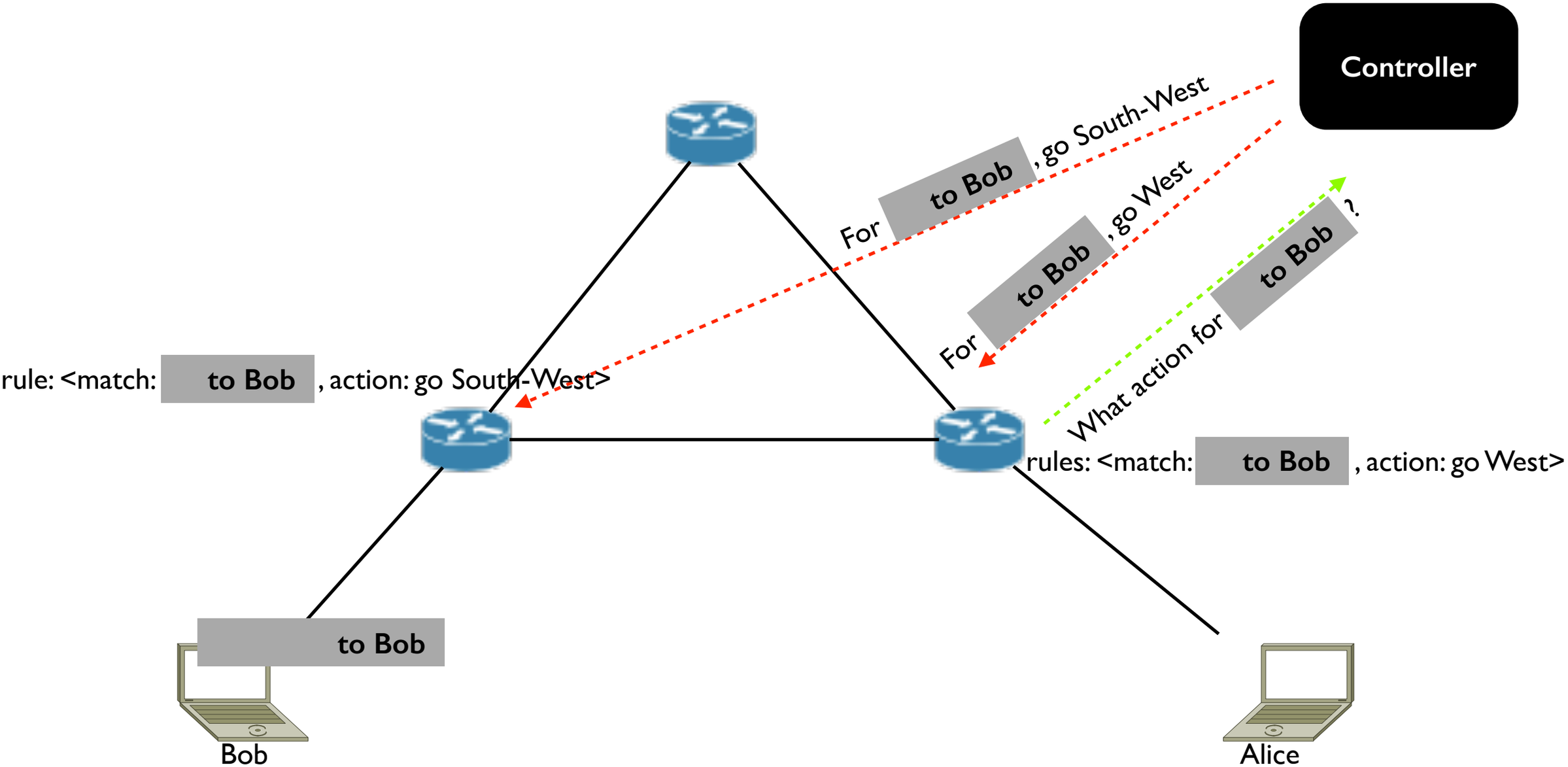
OpenFlow workflow



OpenFlow workflow



OpenFlow workflow



Take away message

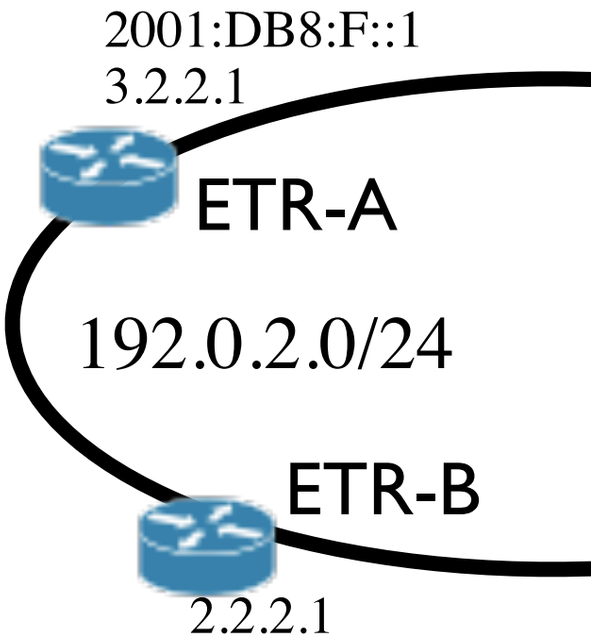
- Separate the roles!
  - locations and identities must be independent
  - control and data planes must be independent

/**/ || ¿¿

Backup

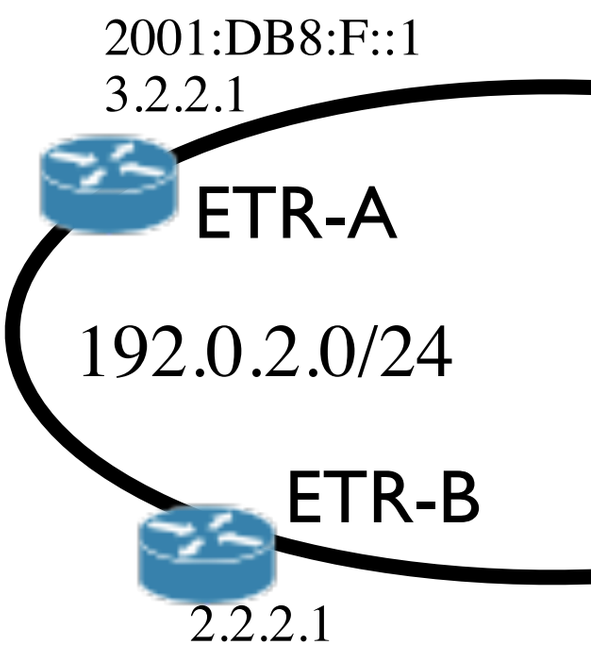
LISP Use Cases

Low OpEx site multihoming

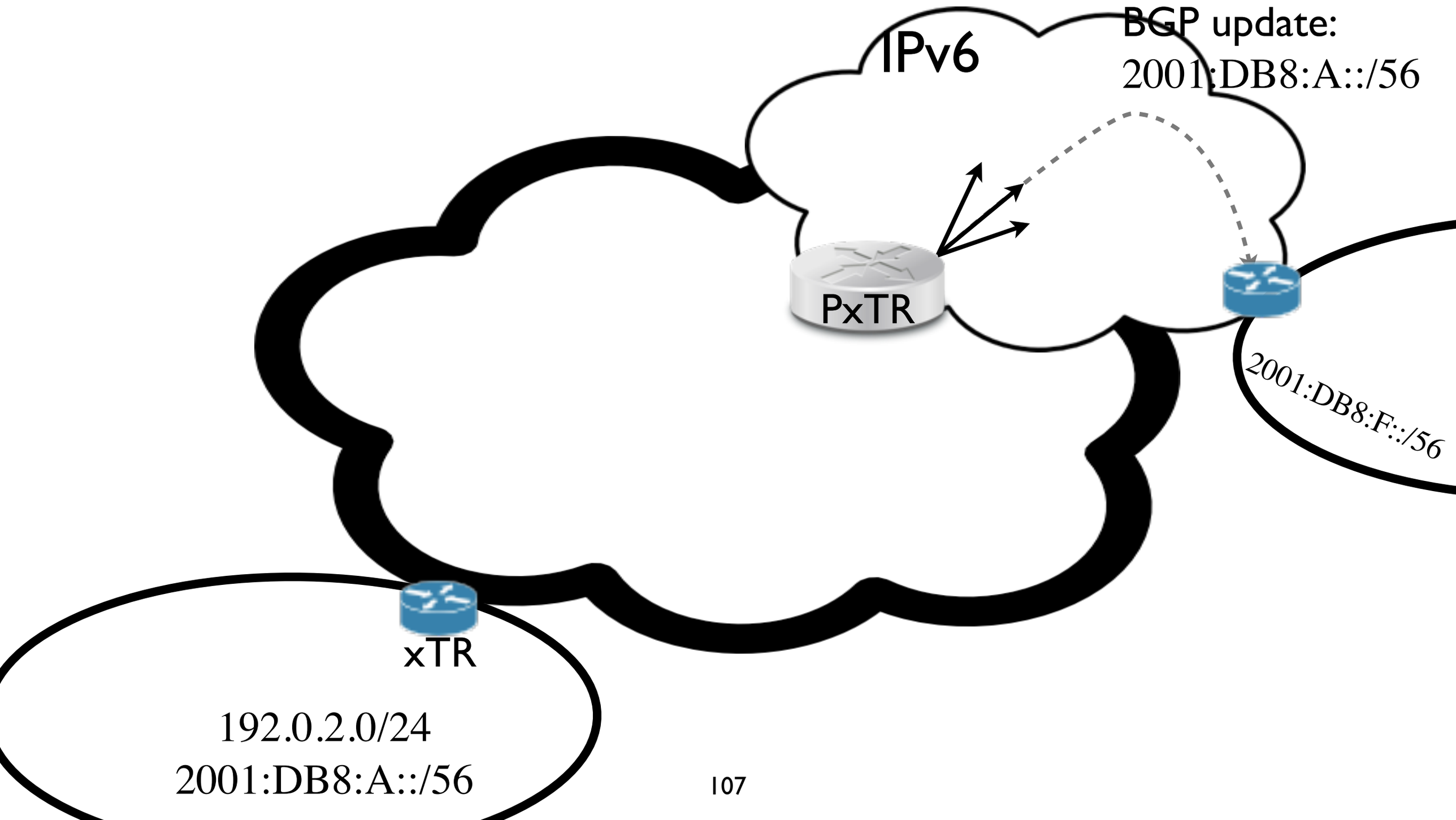


Low OpEx site multihoming

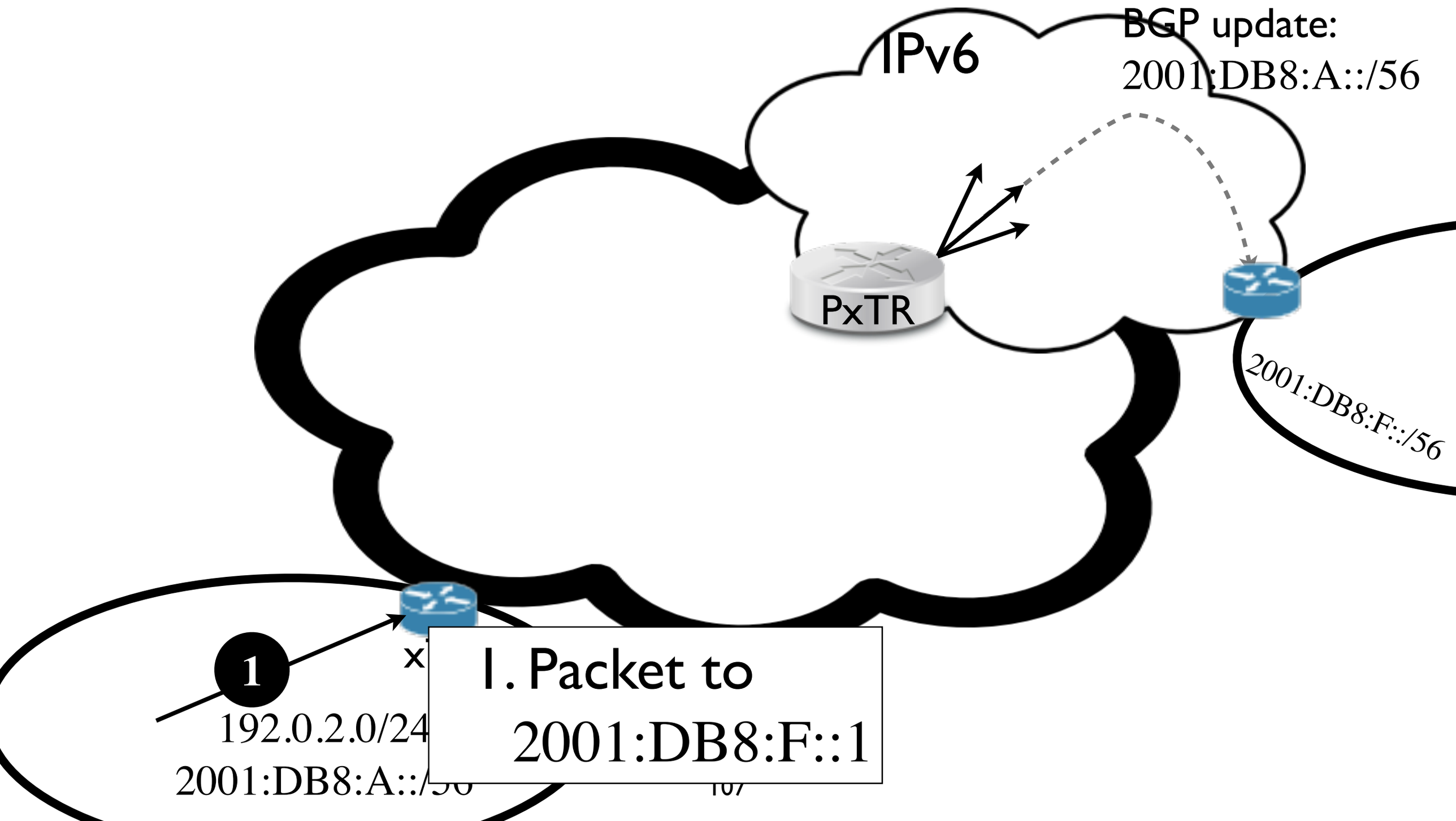
- Basic LISP feature
- ETR-A Primary, ETR-B Backup (IPv6 just in case...)
  - 3.2.2.1, prio: 1, weight: 100
  - 2.2.2.1, prio: 10, weight: 100
  - 2001:DB8:F:1 , prio 100, weight: 100
- ETR-A: 60%, ETR-B: 40% (IPv6 just in case...)
  - 3.2.2.1, prio: 1, weight: 60
  - 2.2.2.1, prio: 1, weight: 40
  - 2001:DB8:F:1, prio 99, weight: 100



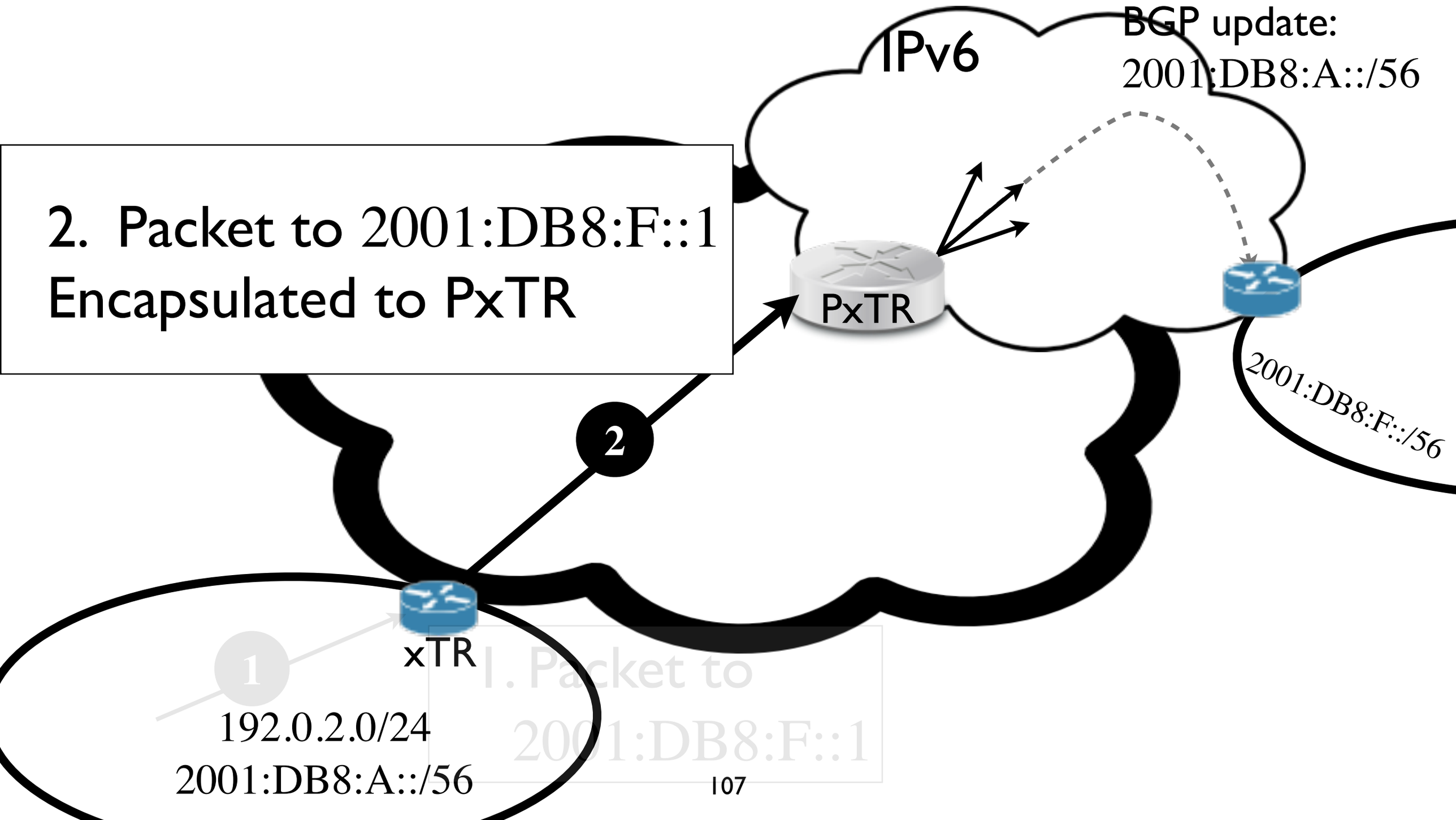
IPv6/IPv4 coexistence



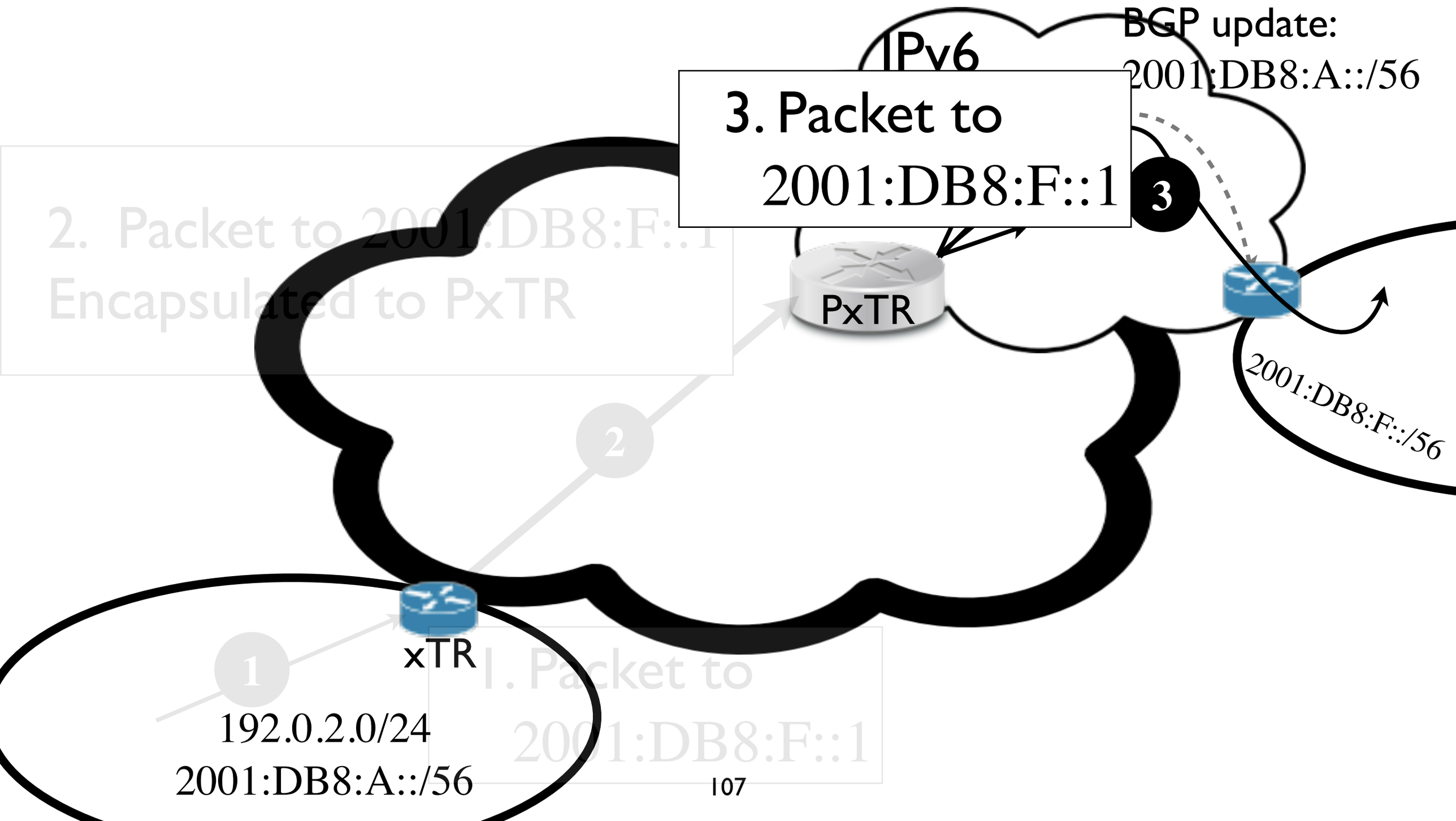
IPv6/IPv4 coexistence



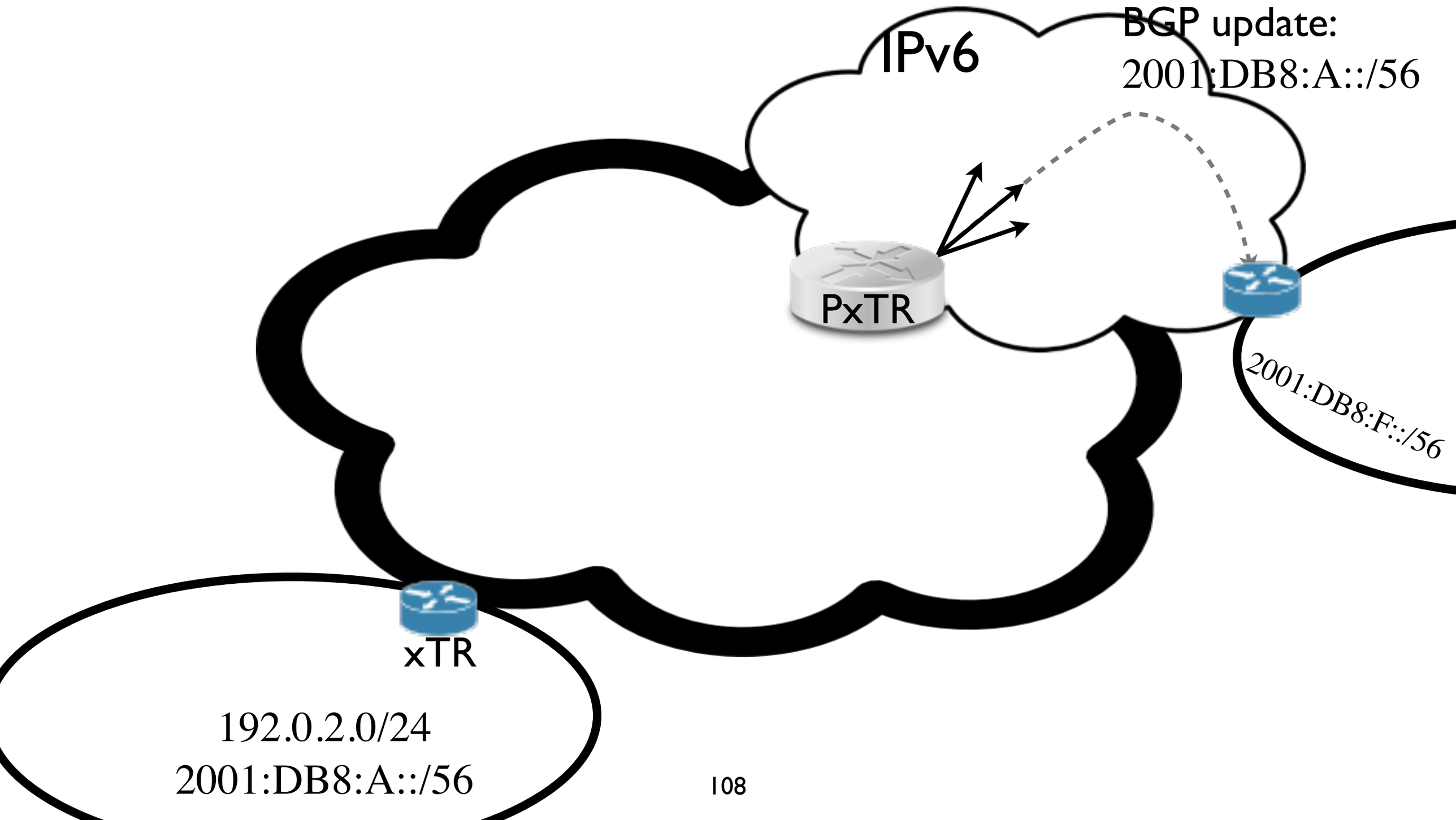
IPv6/IPv4 coexistence



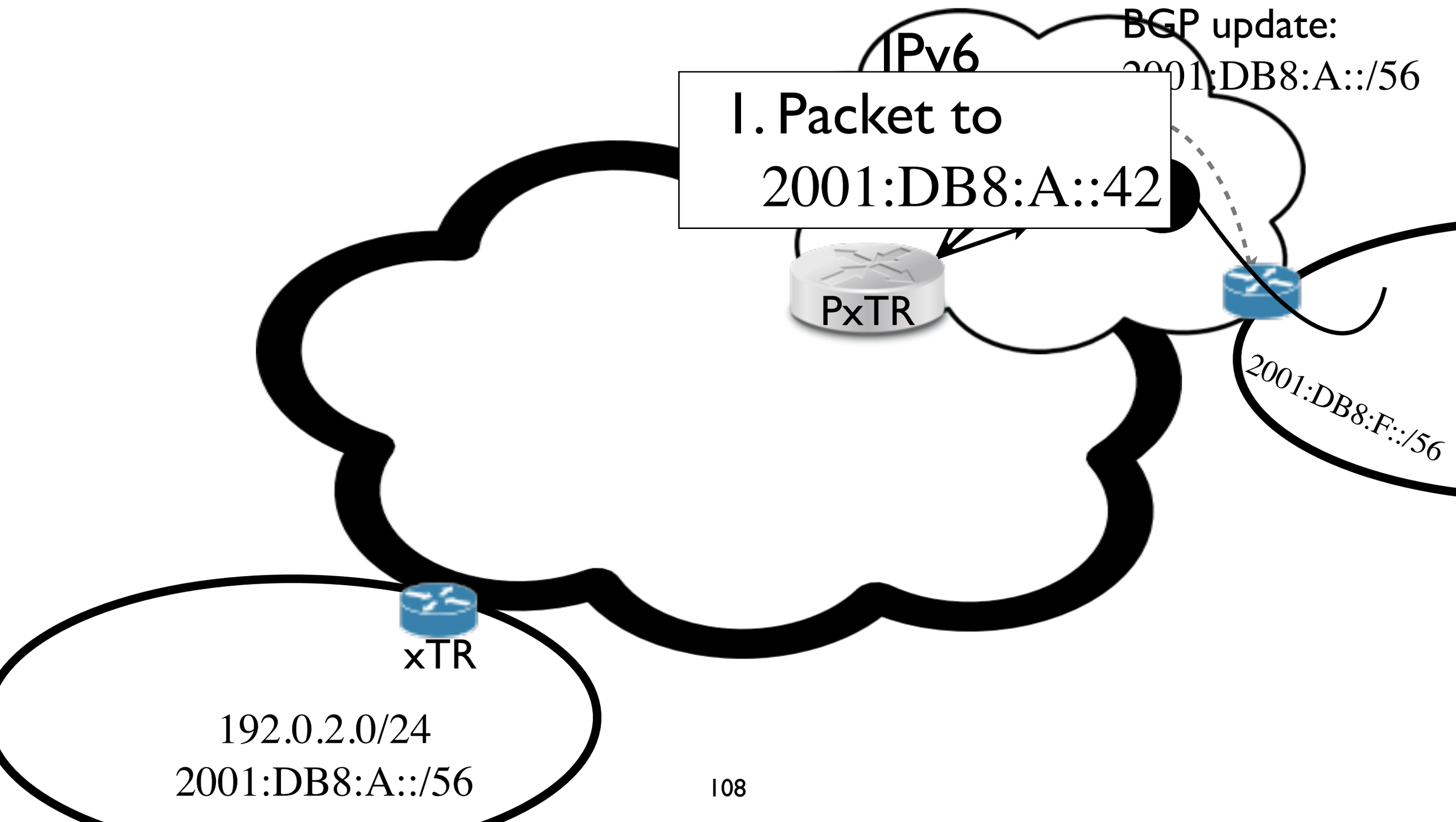
IPv6/IPv4 coexistence



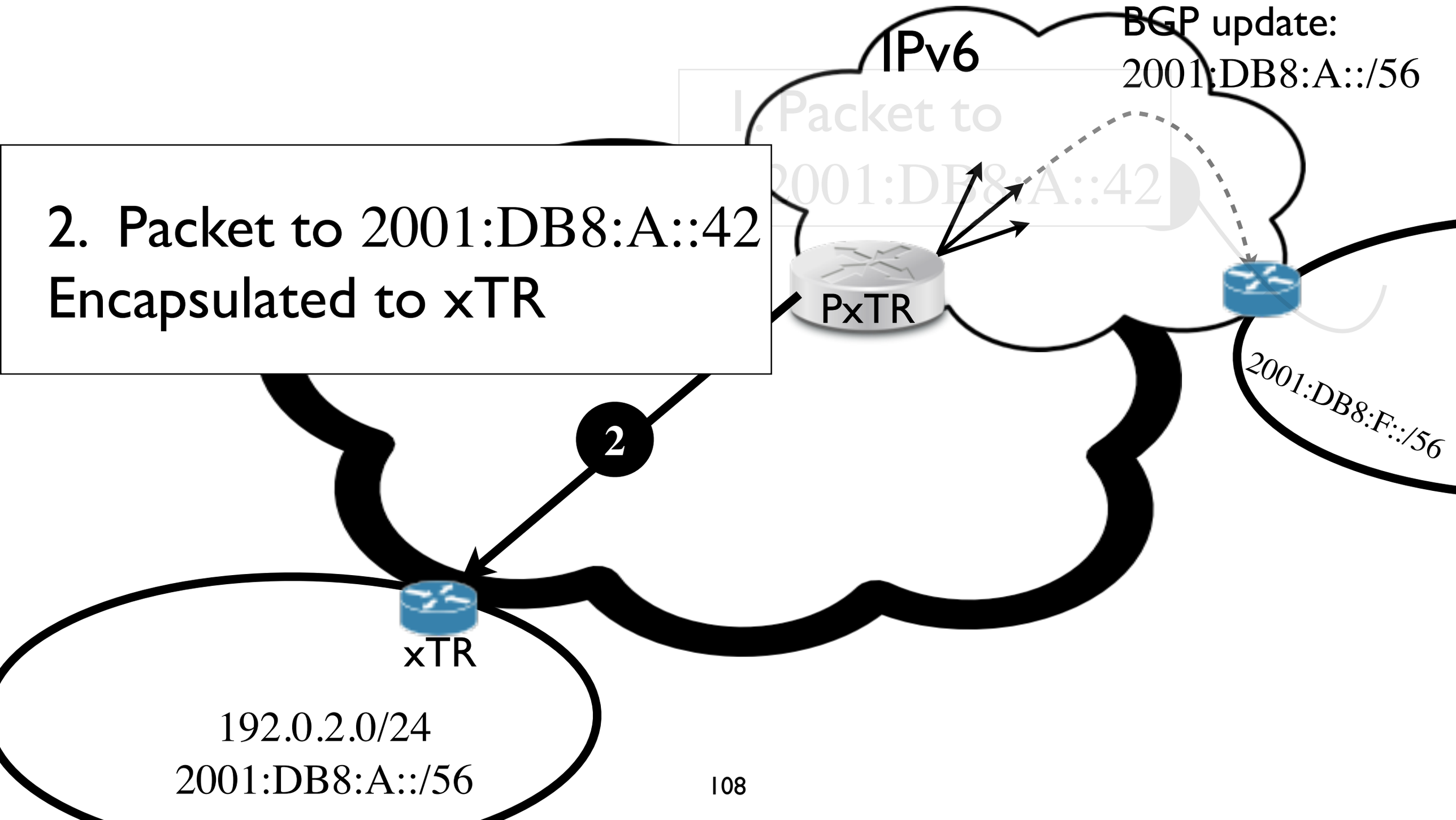
IPv6/IPv4 coexistence



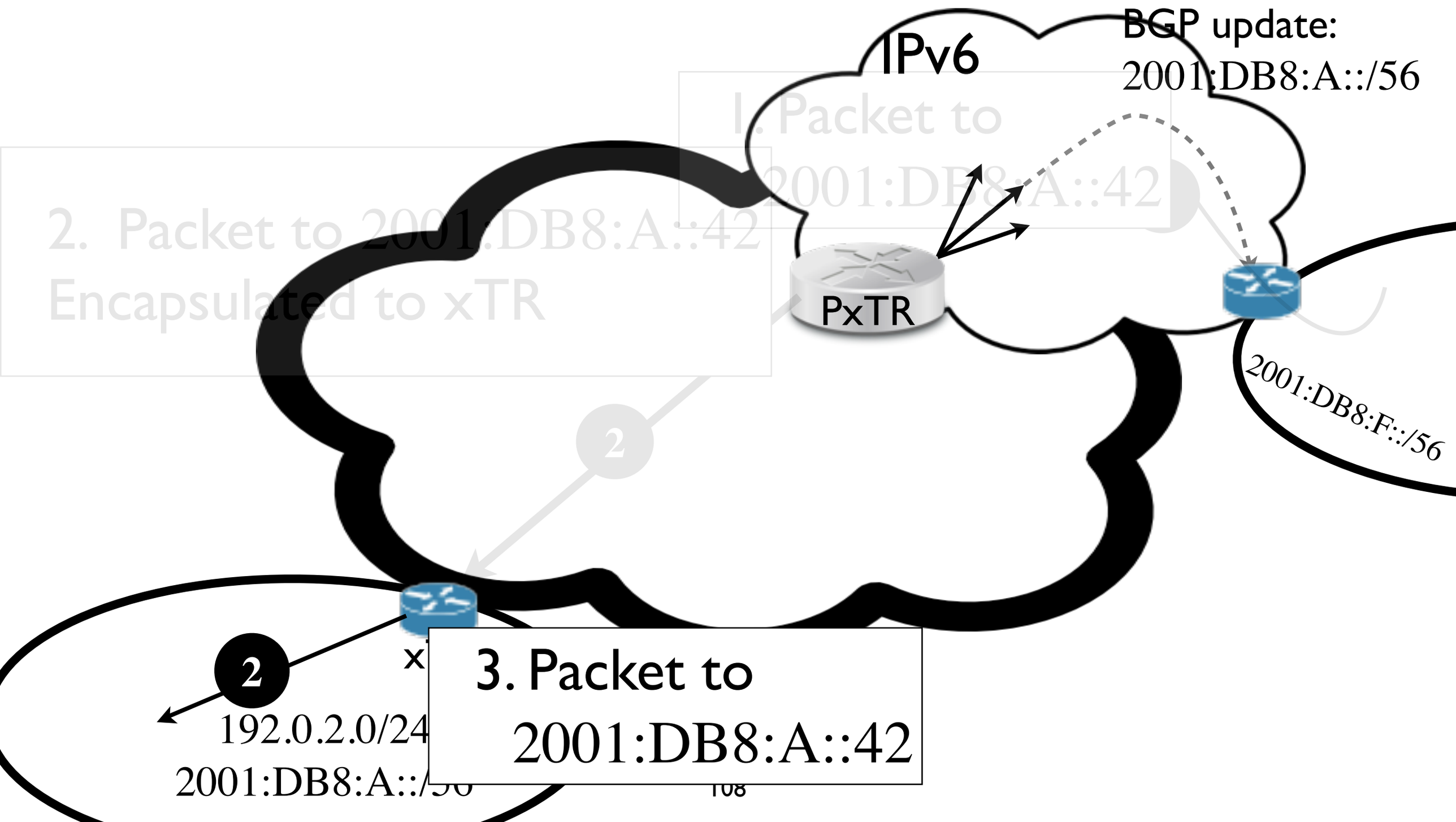
IPv6/IPv4 coexistence



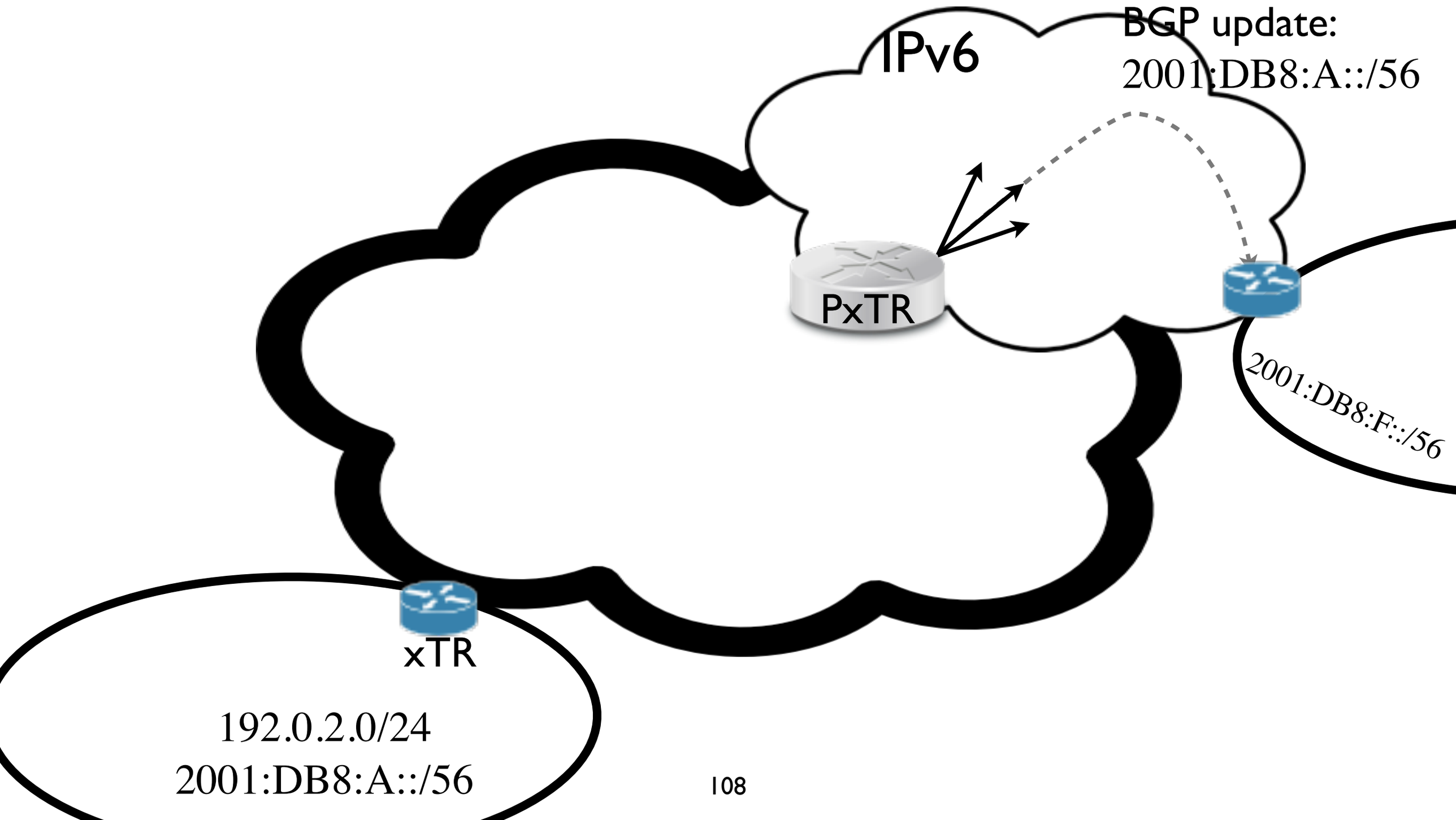
IPv6/IPv4 coexistence



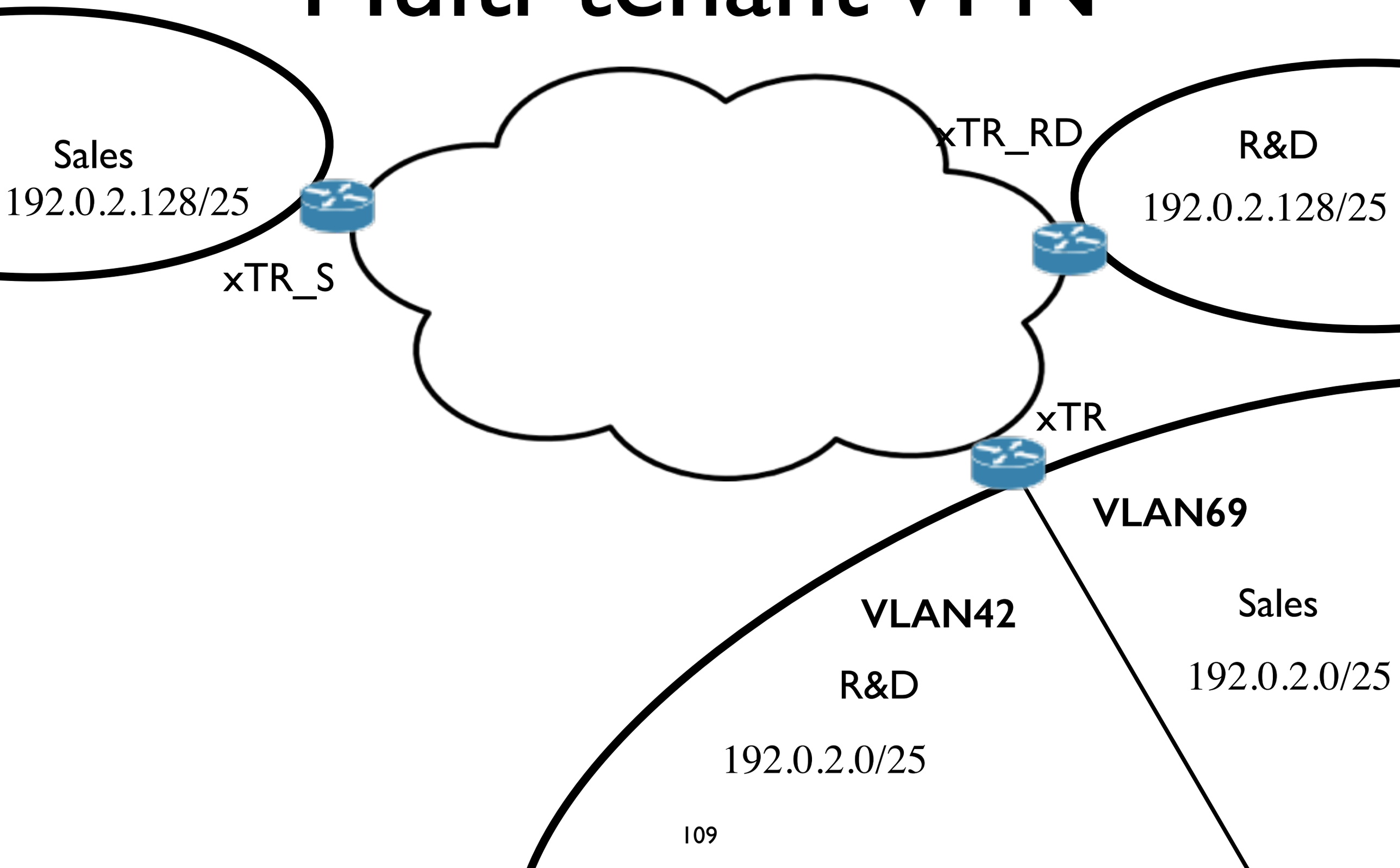
IPv6/IPv4 coexistence



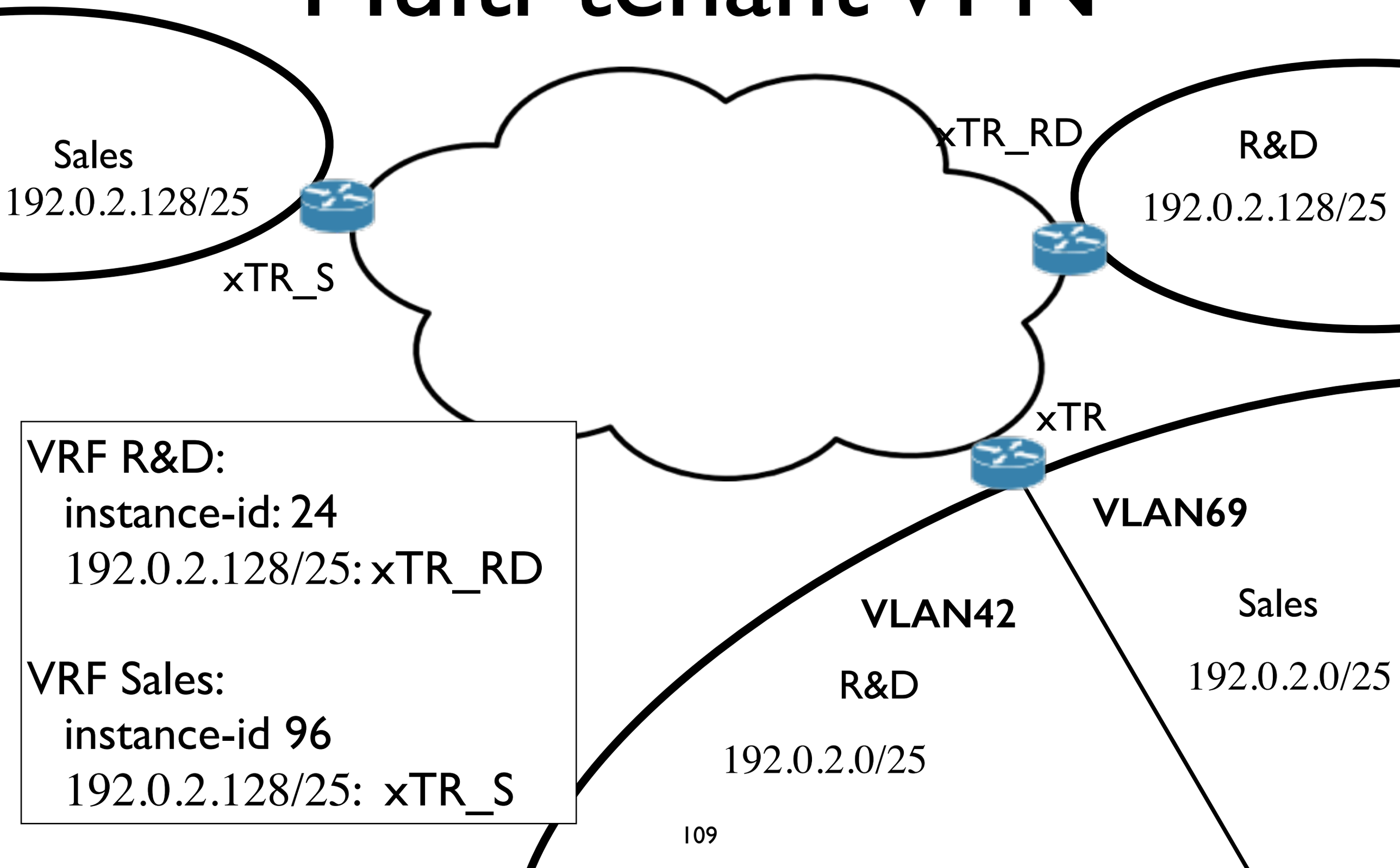
IPv6/IPv4 coexistence



Multi-tenant VPN



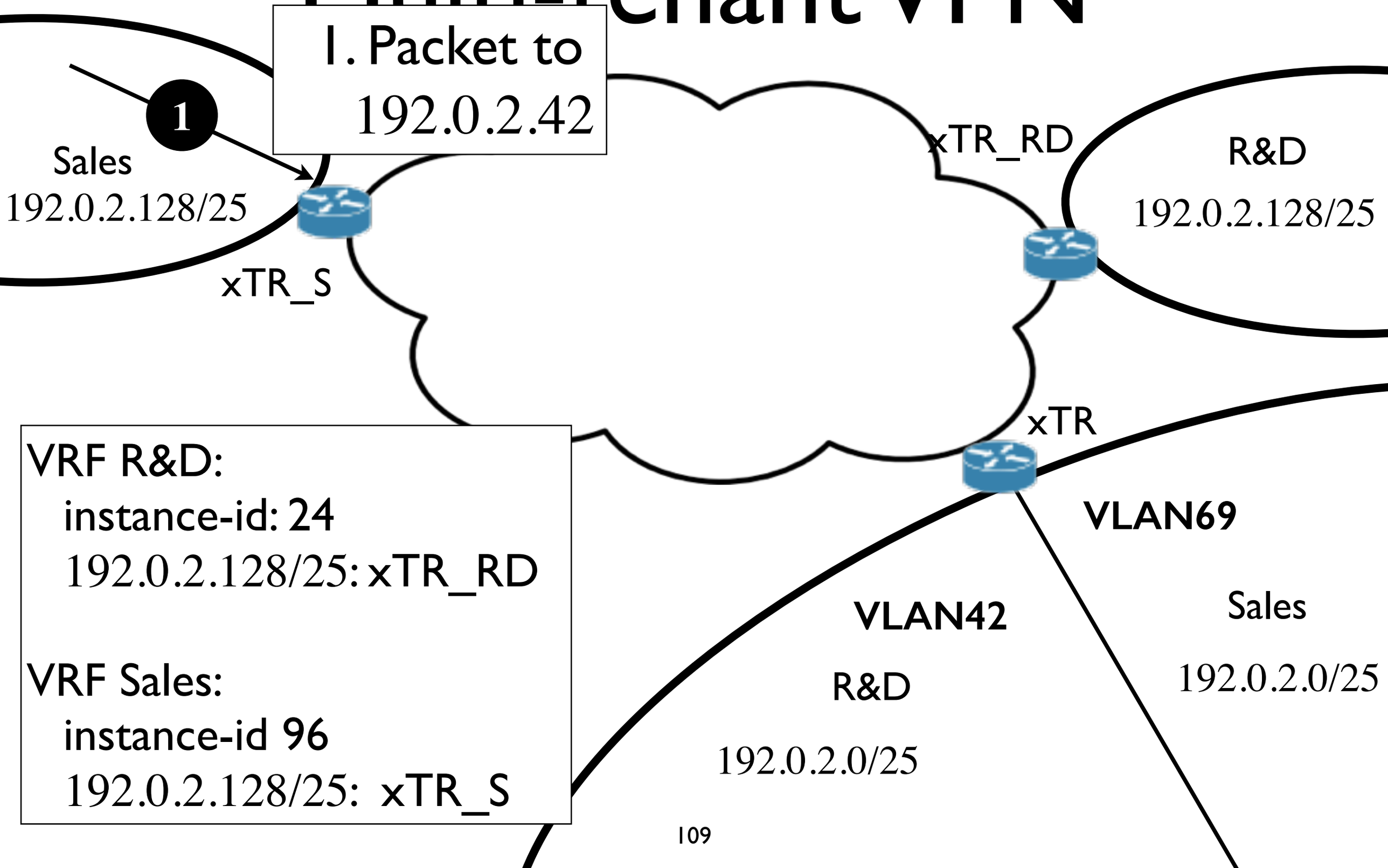
Multi-tenant VPN



VRF R&D:
instance-id: 24
192.0.2.128/25: xTR_RD

VRF Sales:
instance-id 96
192.0.2.128/25: xTR_S

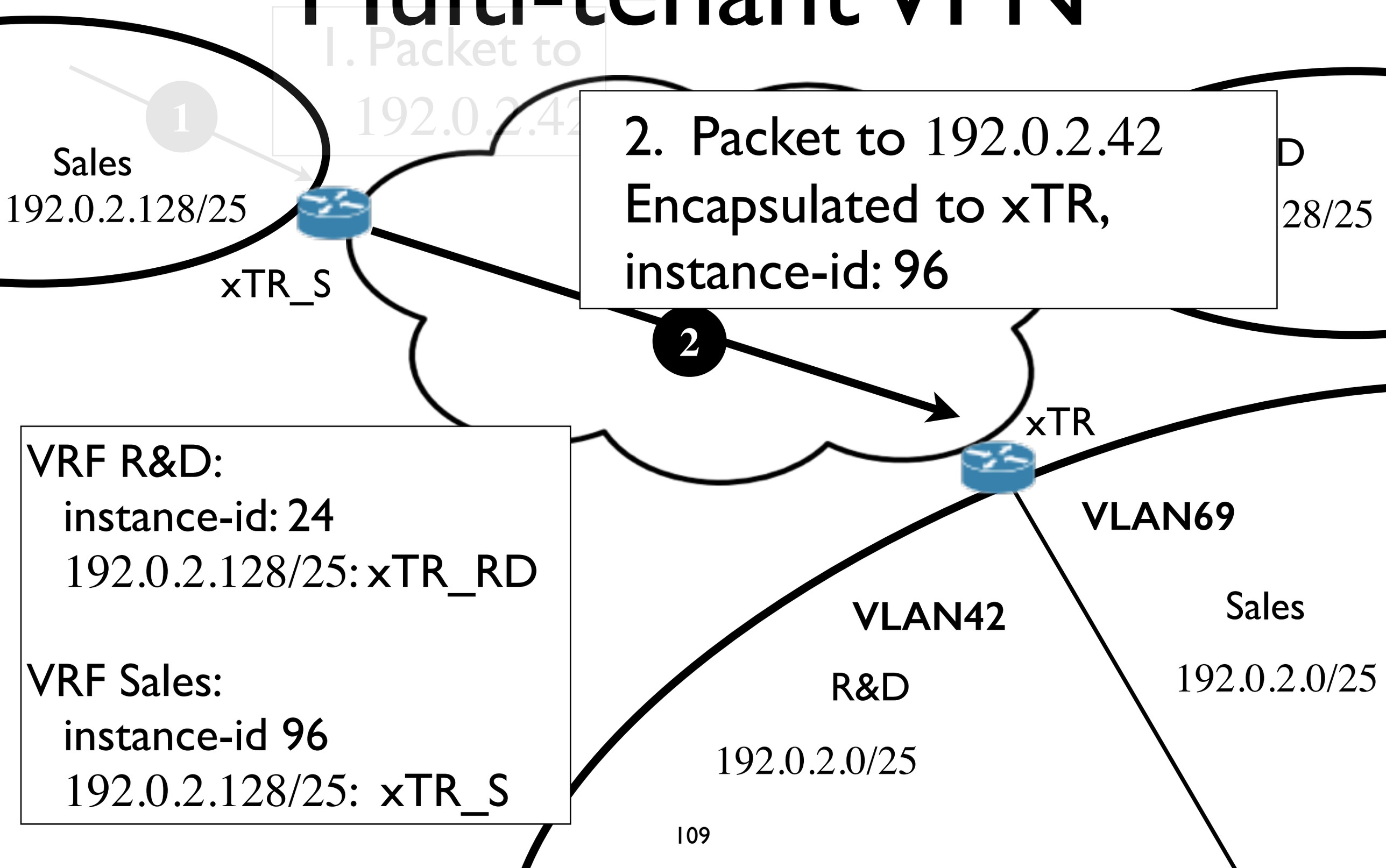
Multi-tenant VPN



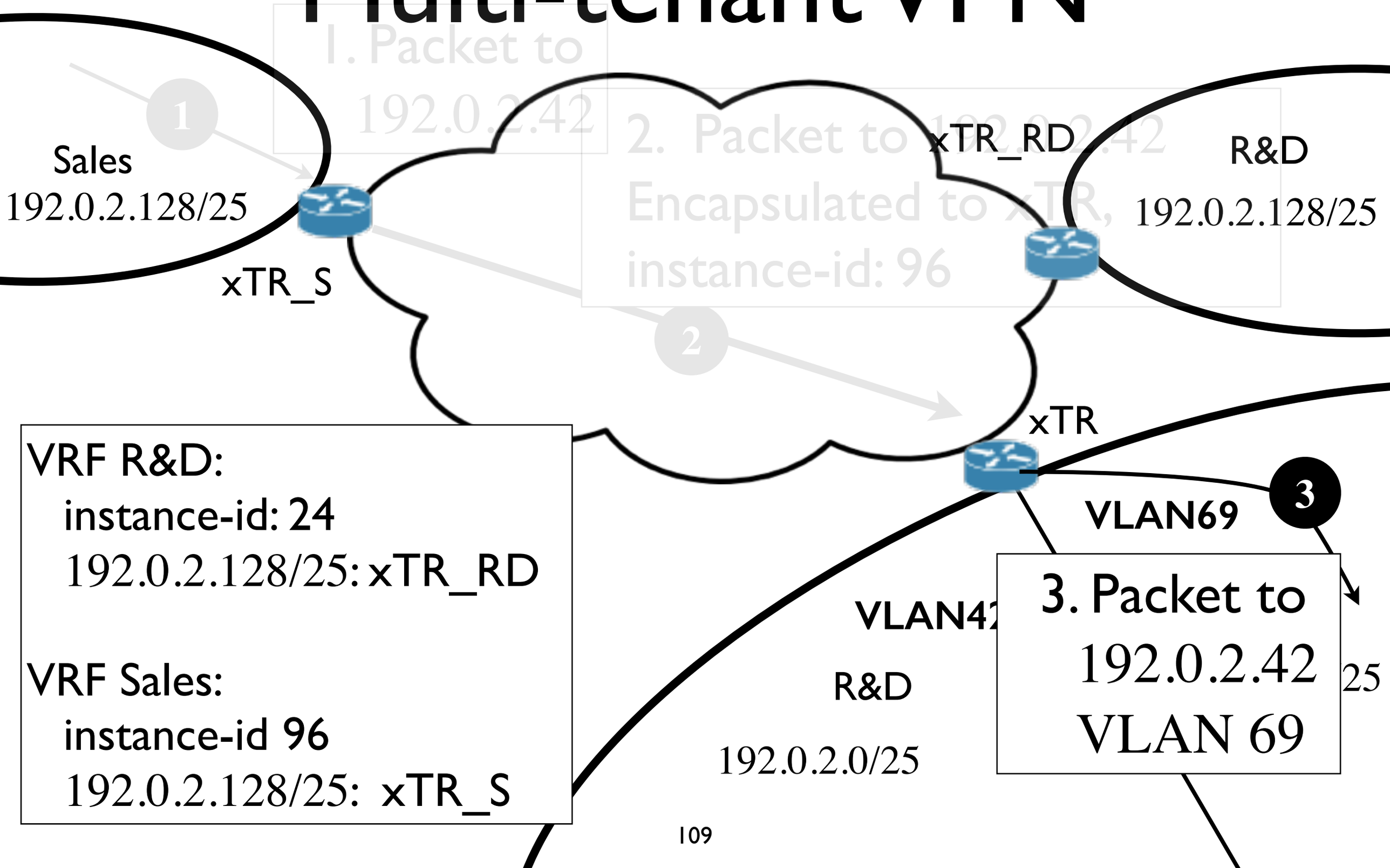
VRF R&D:
instance-id: 24
192.0.2.128/25: xTR_RD

VRF Sales:
instance-id 96
192.0.2.128/25: xTR_S

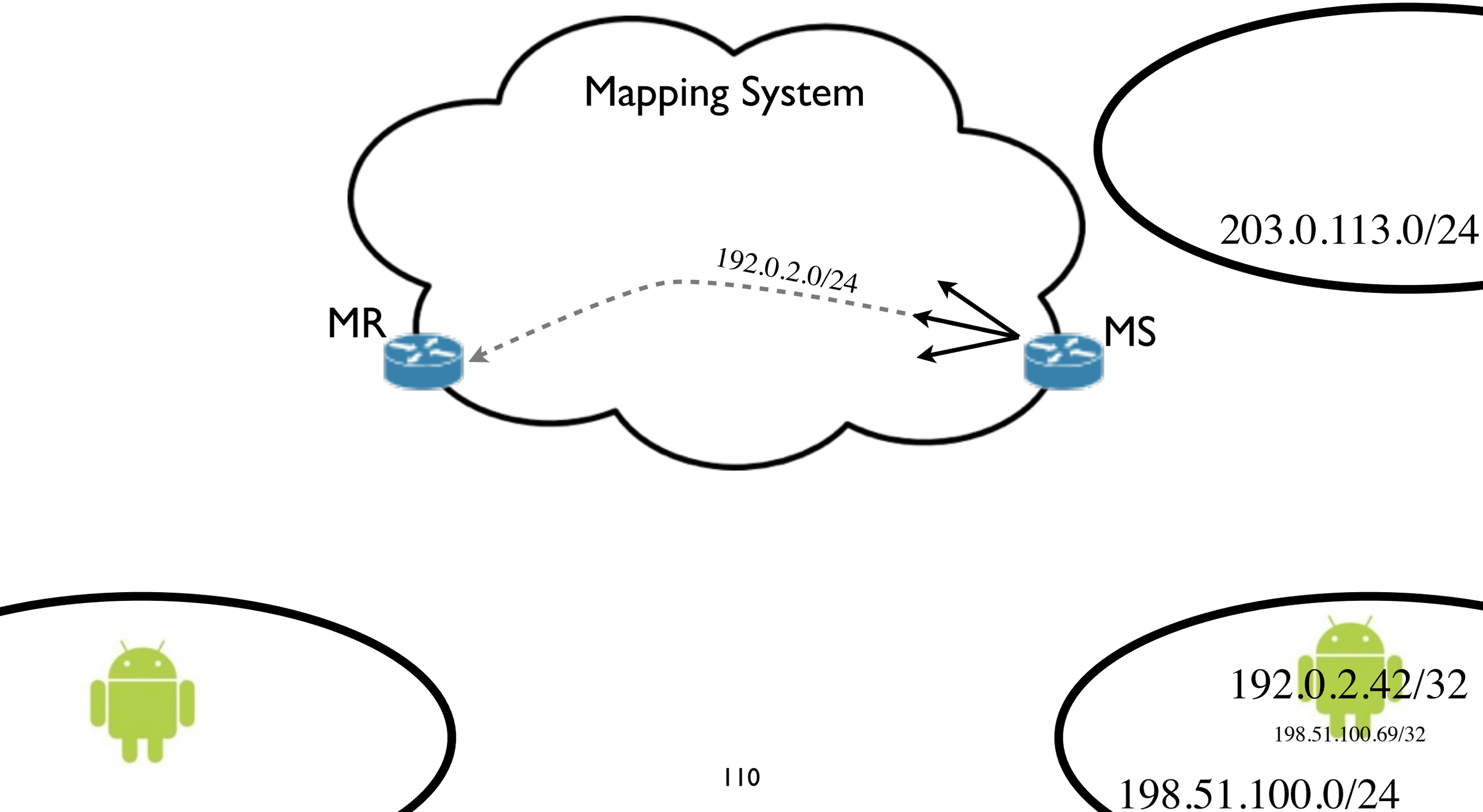
Multi-tenant VPN



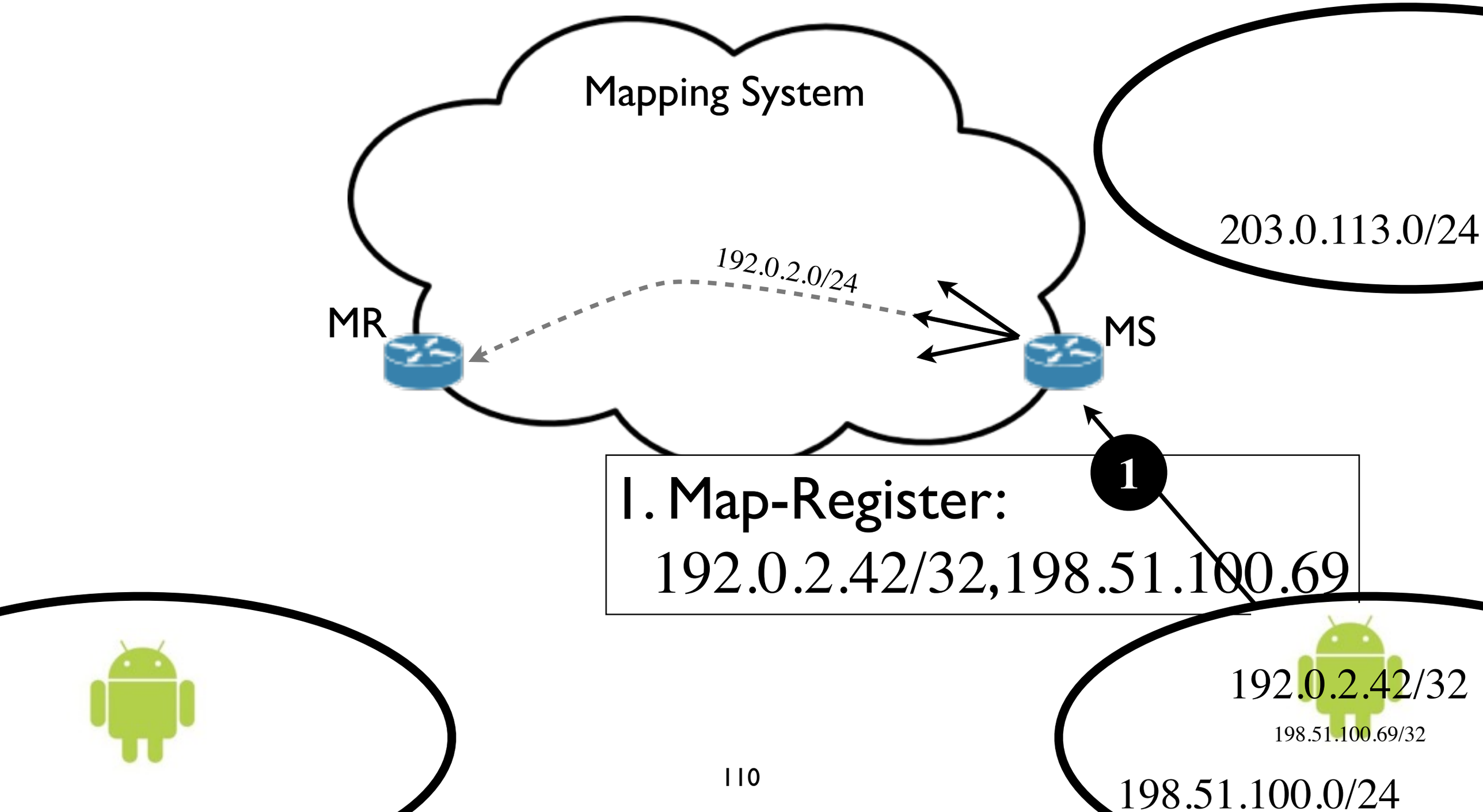
Multi-tenant VPN



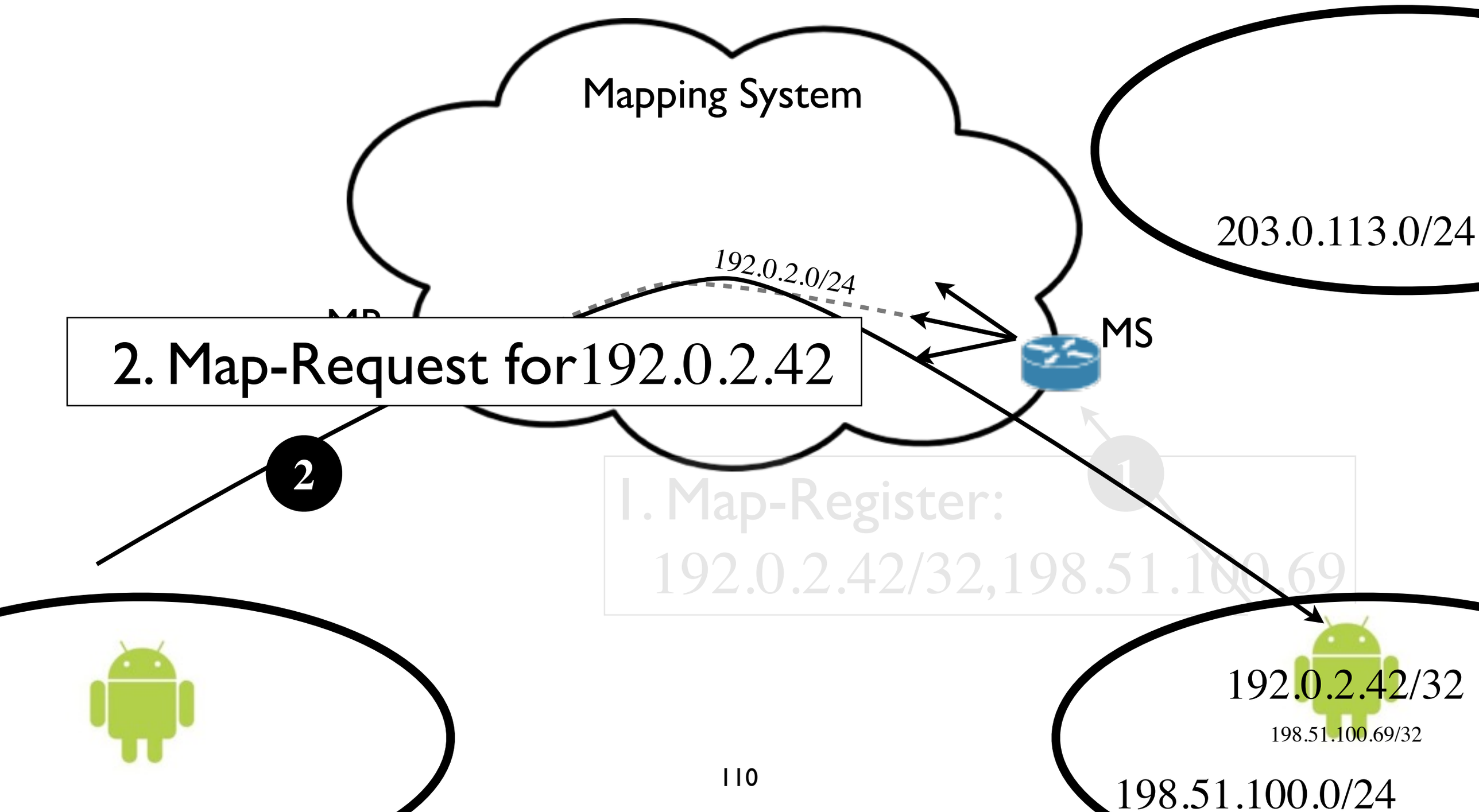
LISP Mobile Nodes



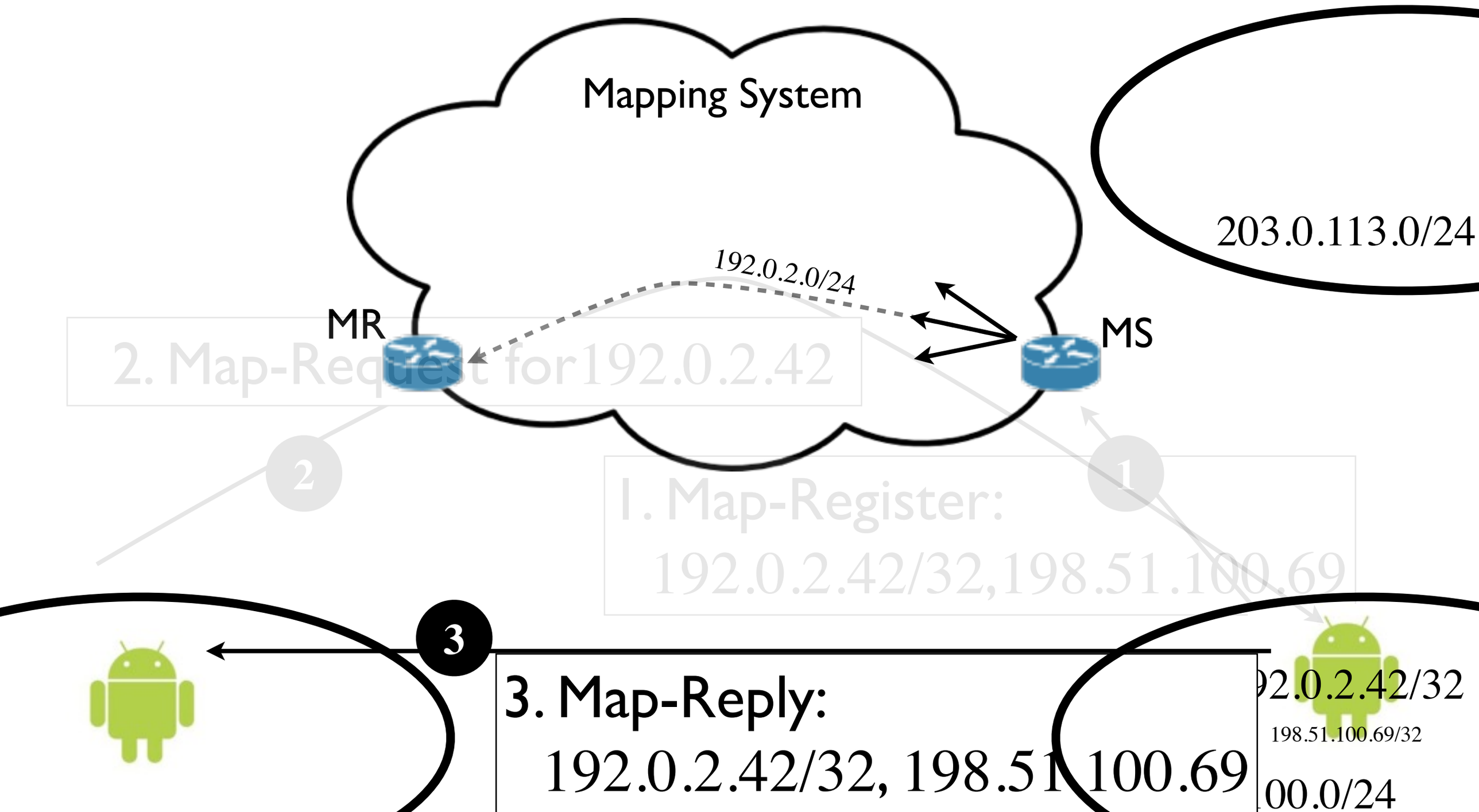
LISP Mobile Nodes



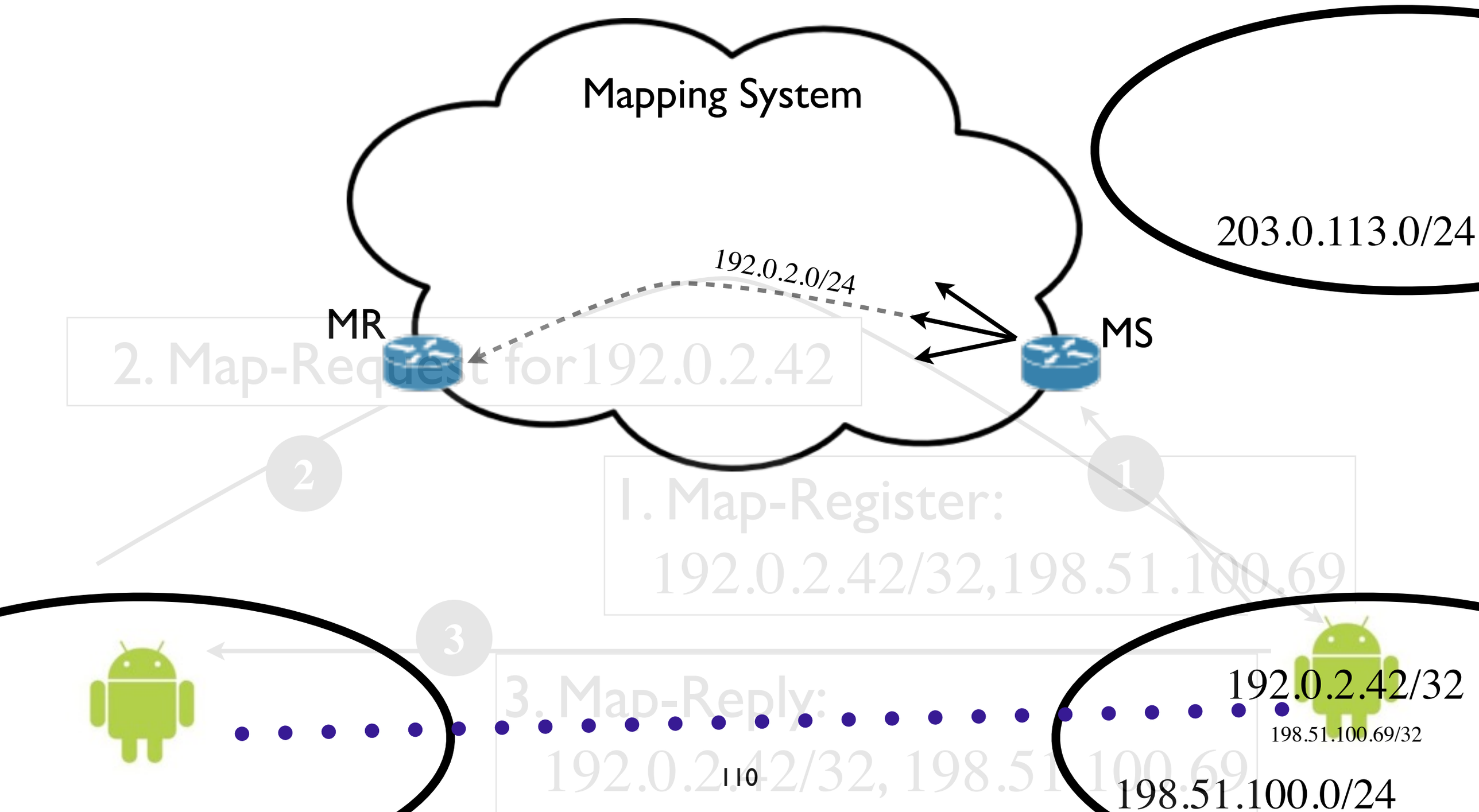
LISP Mobile Nodes



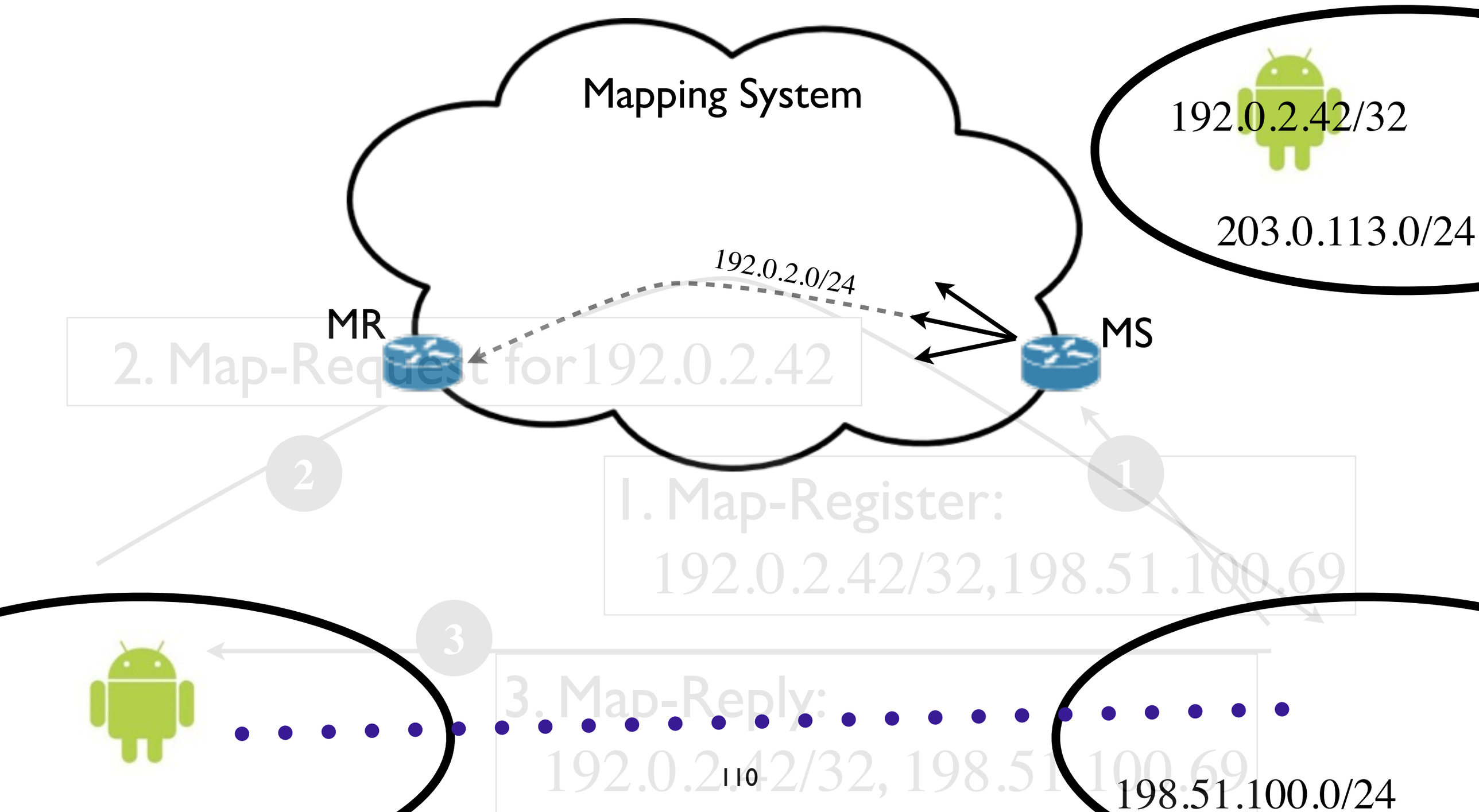
LISP Mobile Nodes



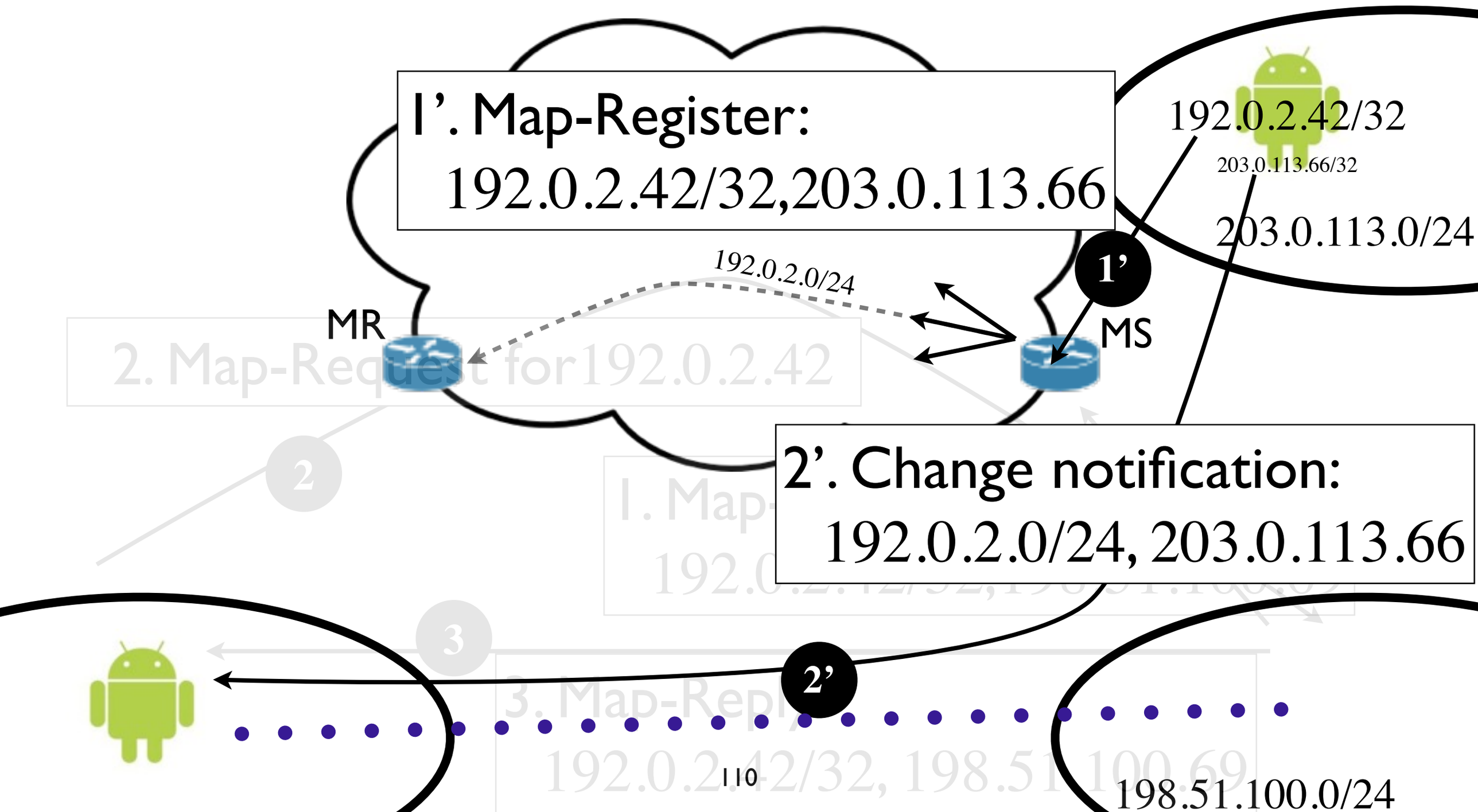
LISP Mobile Nodes



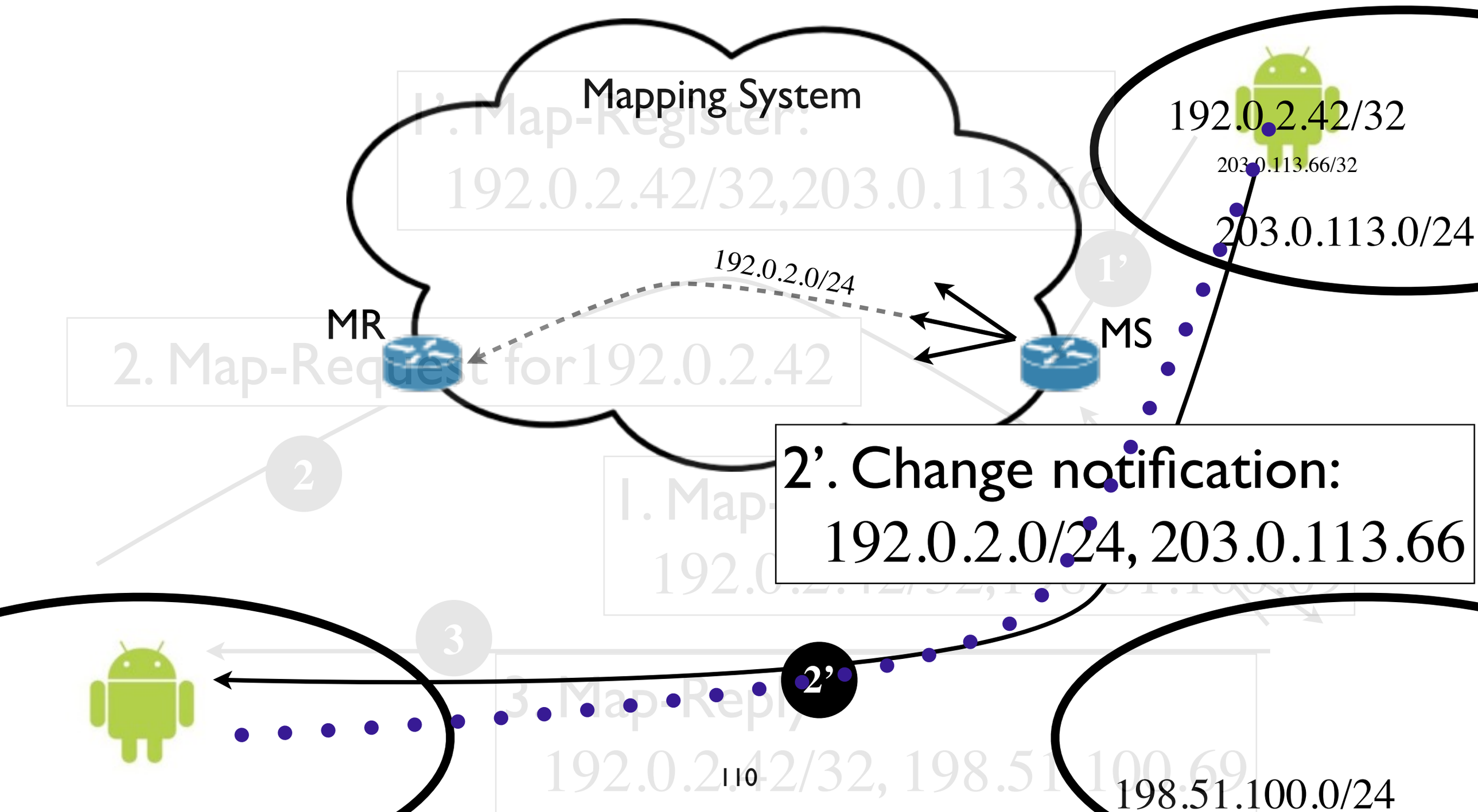
LISP Mobile Nodes



LISP Mobile Nodes



LISP Mobile Nodes



Time for question

- How would you do Virtual Machine (VM) mobility with LISP without changing the hypervisor or the VM itself?
  - think ARP...