Locator/ID Separation Protocol (LISP)

Damien Saucez*

INRIA Sophia Antipolis

FRNOG 18, December 2th, 2011

* special thanks to Olivier Bonaventure, Luigi Iannone and Dino Farinacci

Disclaimer

- Not a vendor
- Not an operator
- LISP will not solve all the problems

➡ try to stay with the facts (now)

Agenda

- Why Separating Identifiers from Locators?
- Locator/ID Separation Protocol (LISP)
- LISP Use Case
- Open Questions

Why Separating Identifiers from Locators?

Bad Traffic Engineering

- Outgoing TE is ok
- Incoming TE is not can be hard, only tricks (BGP, DNS, NAT)
- Scalability issues
  - Table size (Memory)
  - Churn (CPU)

The IP Schizophrenia

- The IP addresses currently used by endhosts play two complementary roles
  - **Identifier role**: the IP address identifies (with port) the endpoint of transport flows
  - **Locator role**: the IP address indicates the paths used to reach the endhost
    - these paths are updated by routing protocols after each topology change

The Locator/Identifier Separation

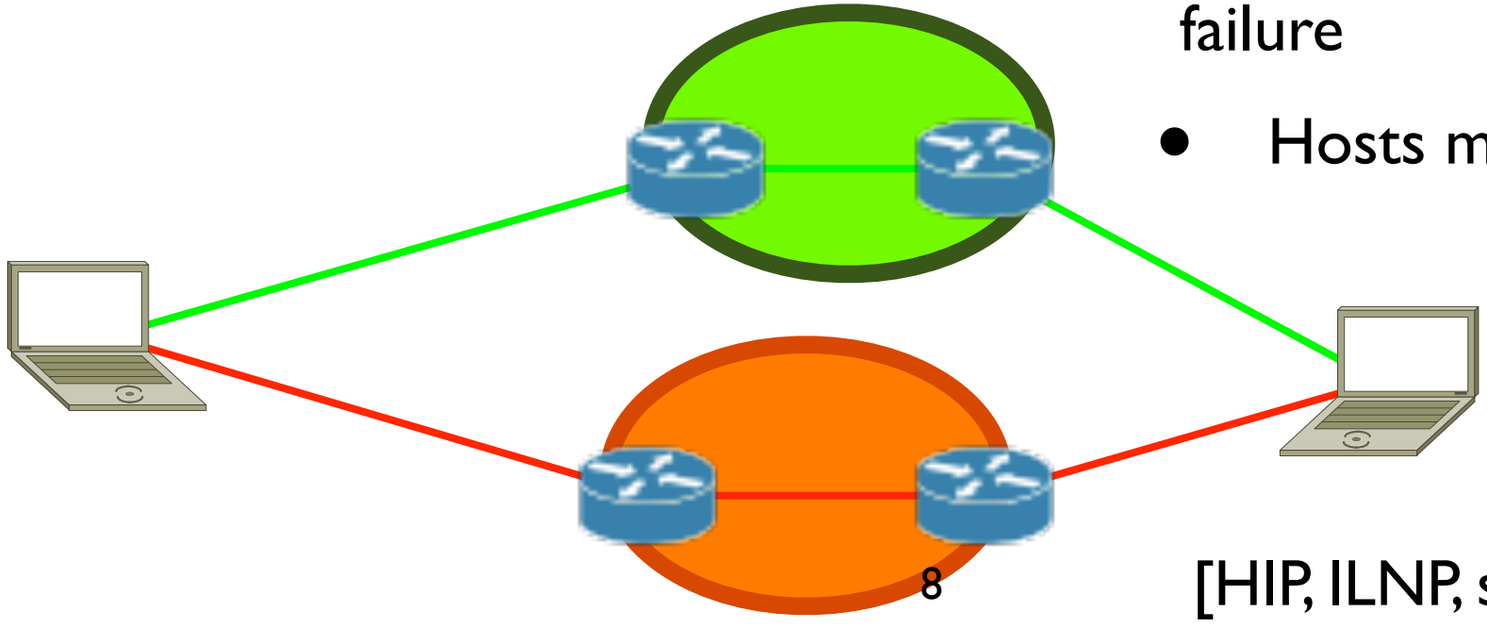
- Today, changing the locator means changing the identifier, breaking the pending flows
- Separating the locator and the identifier roles to avoid breaking the flows
  - Host-based approach
  - Network-based approach

Host-based Loc/ID split

Transport layer

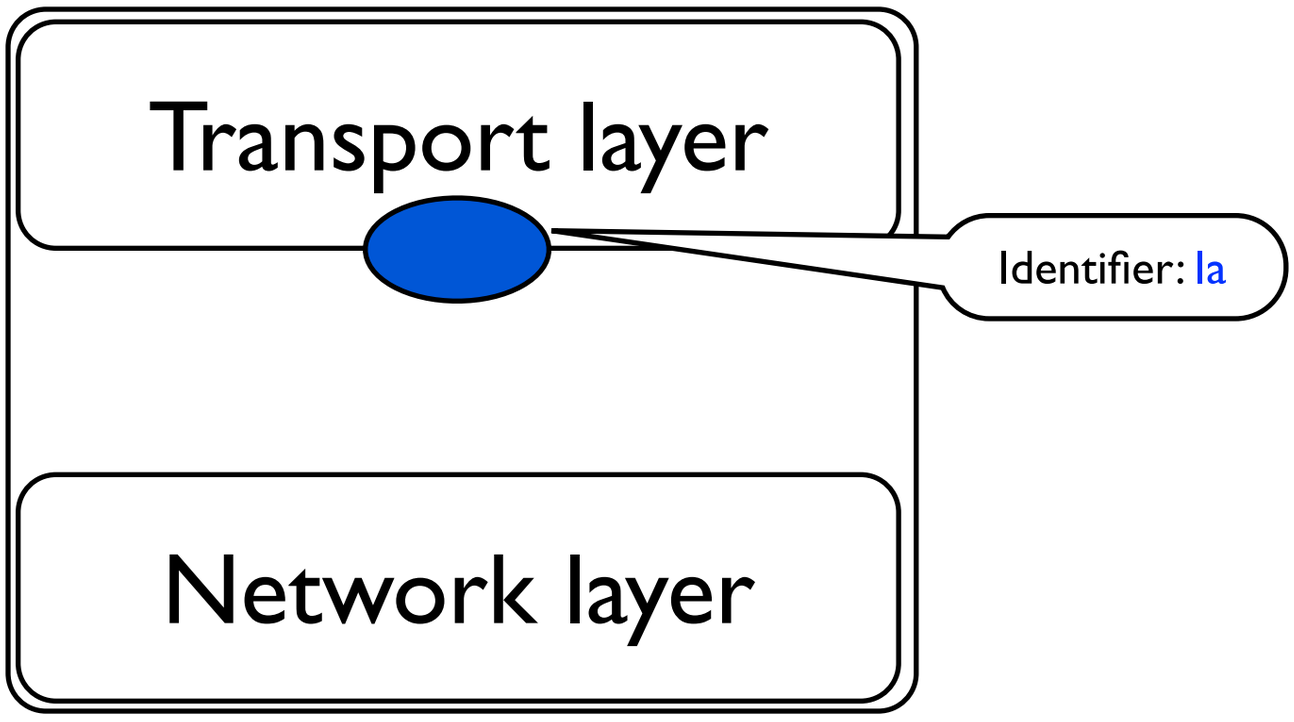
Network layer

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

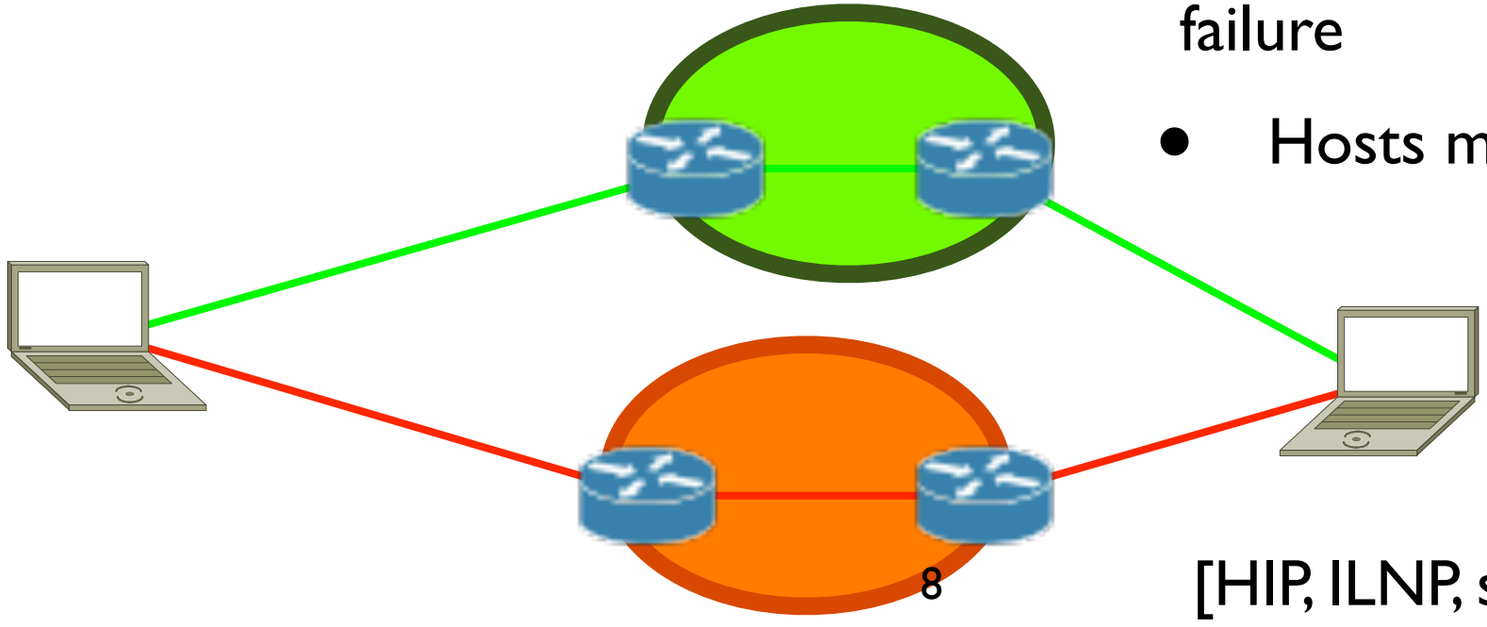


[HIP, ILNP, shim6, Six/One, MPTCP(?)]

Host-based Loc/ID split

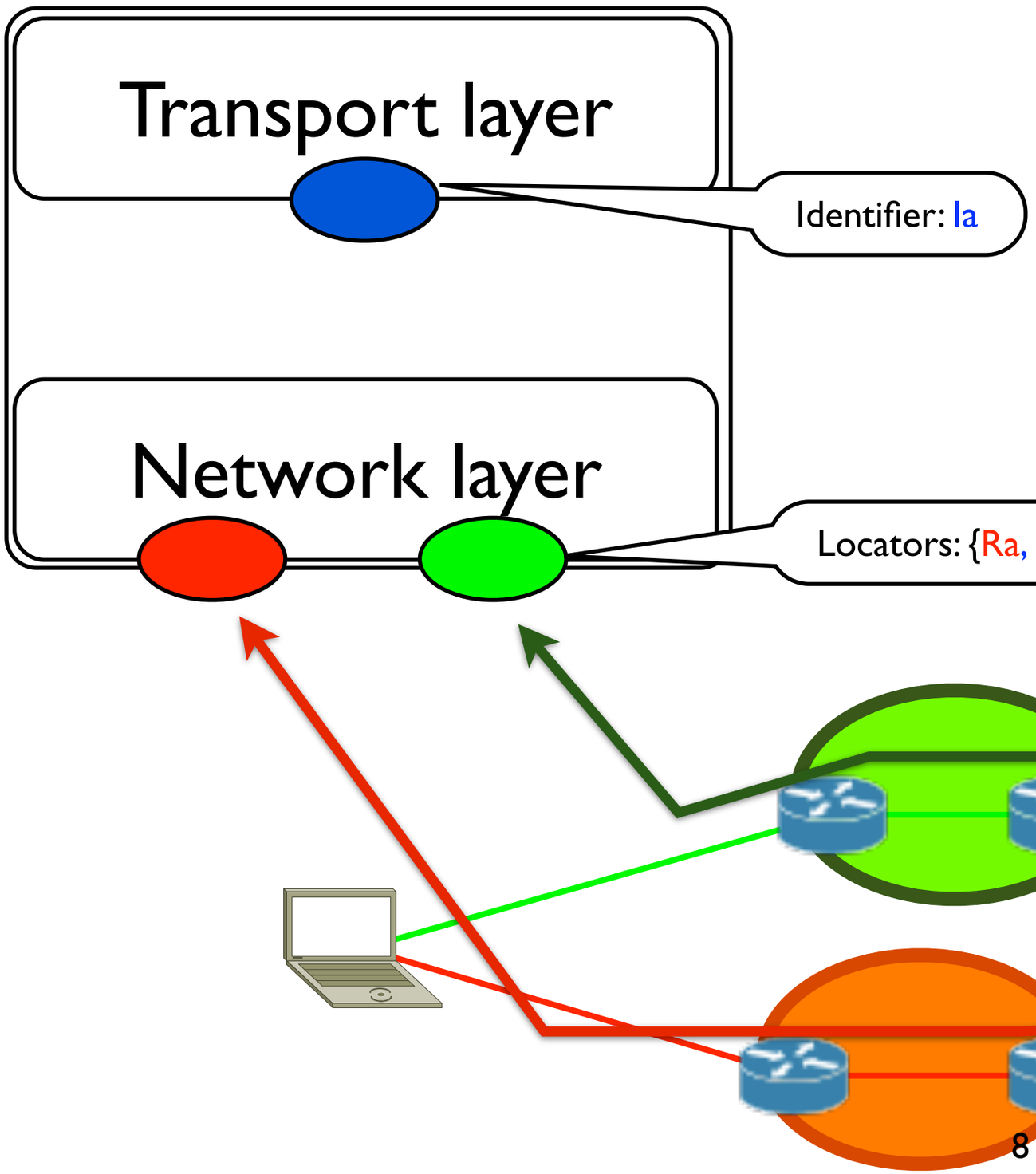


- Roles
  - Translates the packets so that
    - Transport layer only sees the host identifier
    - Network layer only sees locators
  - Manages the set of locators
  - Switches from one locator to another upon move or after link failure
  - Hosts maintain flow state



[HIP, ILNP, shim6, Six/One, MPTCP(?)]

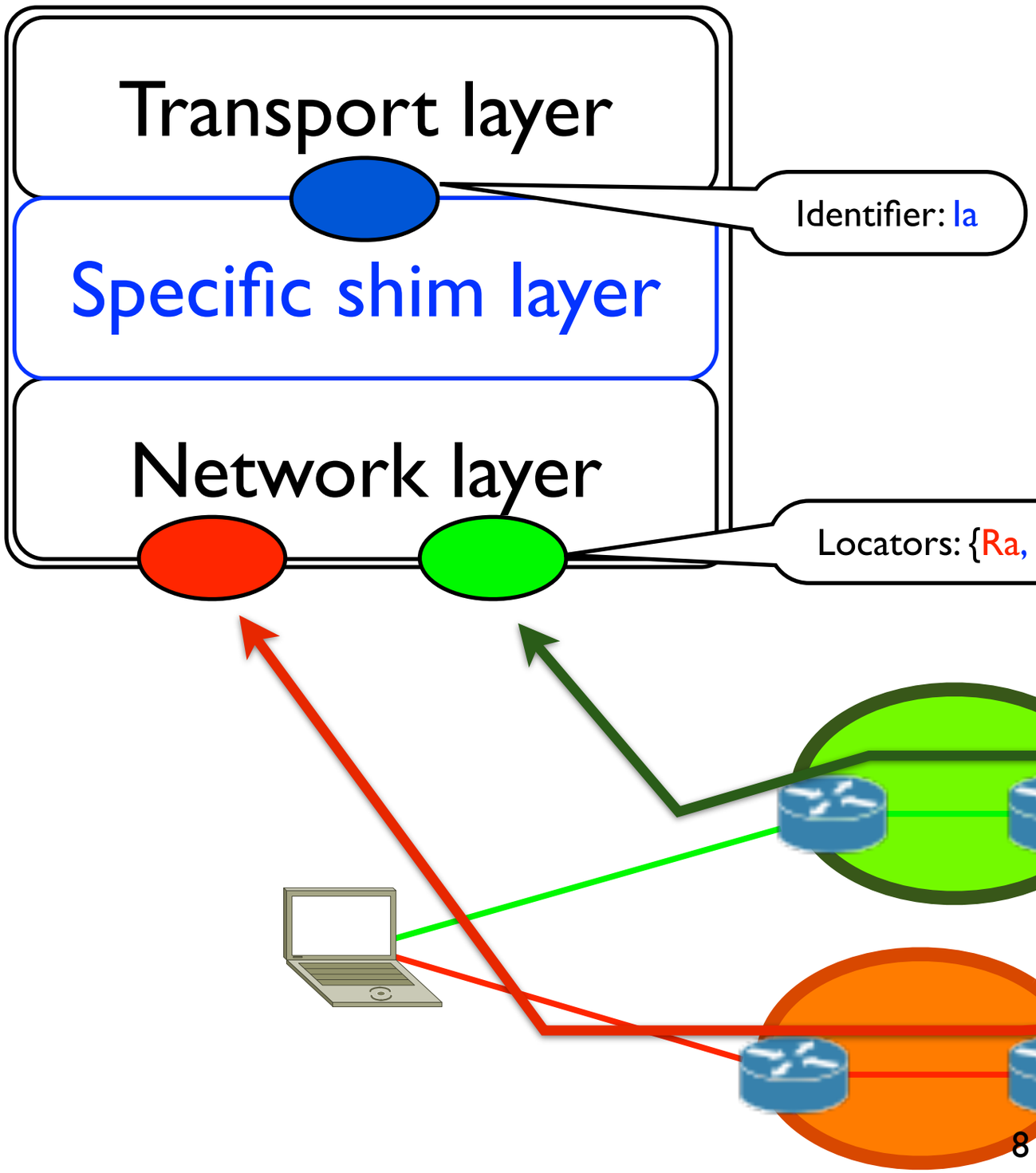
Host-based Loc/ID split



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

[HIP, ILNP, shim6, Six/One, MPTCP(?)]

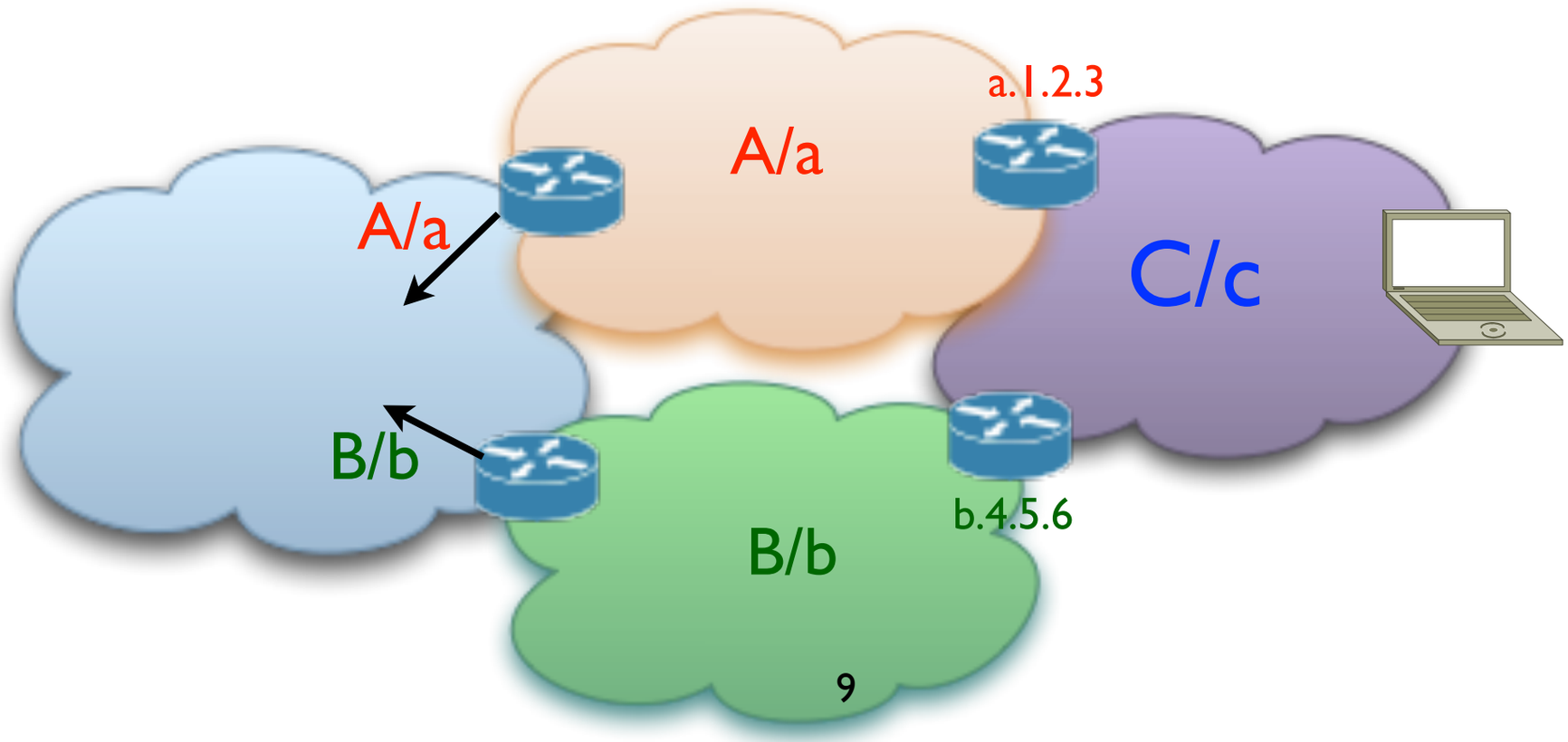
Host-based Loc/ID split



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

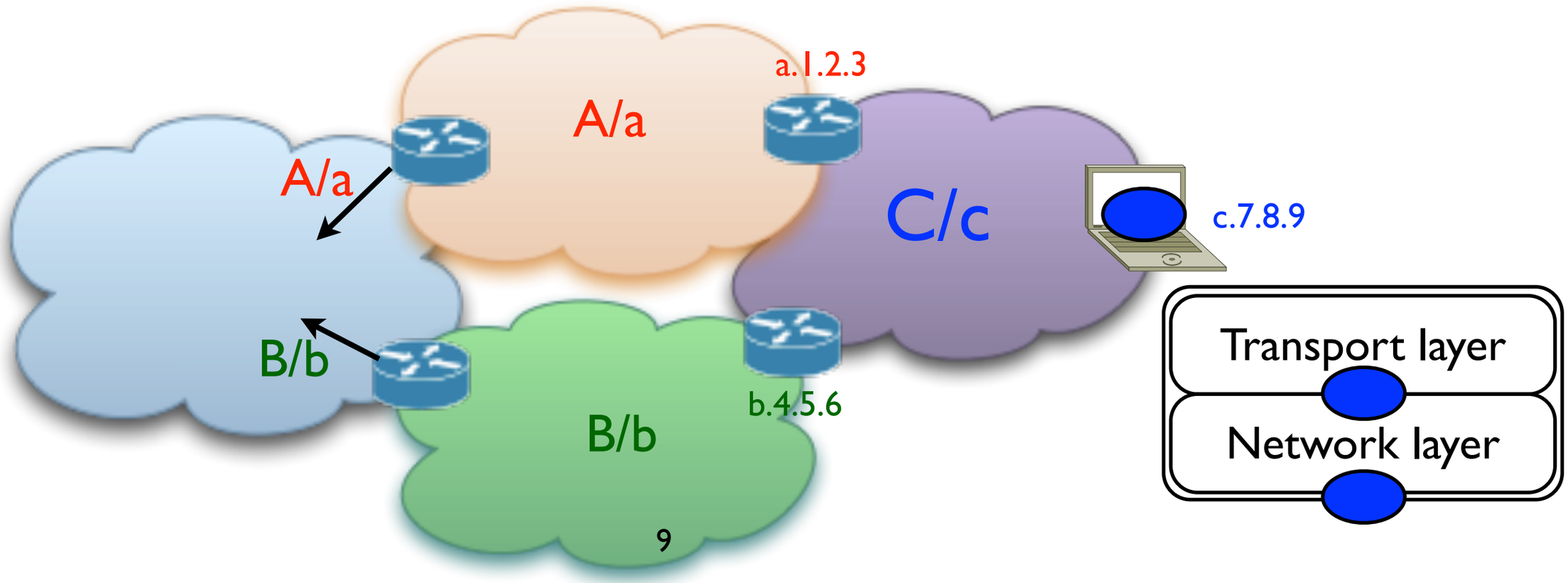
[HIP, ILNP, shim6, Six/One, MPTCP(?)]

Network-based Loc/ID split



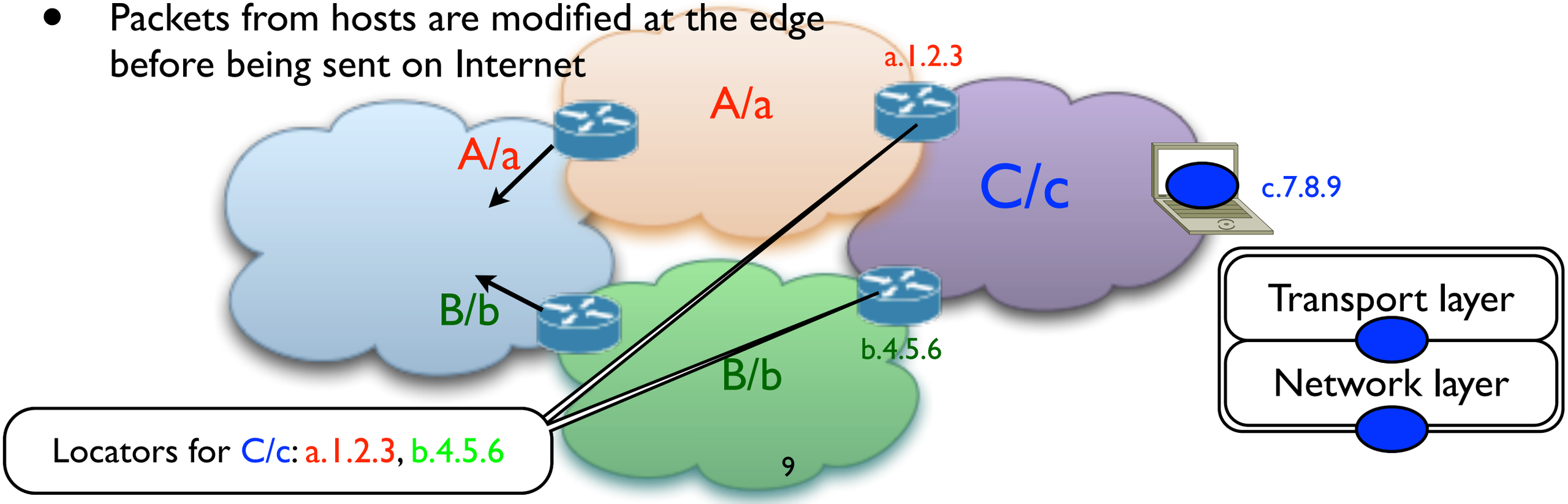
Network-based Loc/ID split

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



Benefits of Identifier and Locator separation*

- Reduction of the DFZ routing table size
- No provider lock-in
- More traffic control and cost-effective multihoming
- Mobility

* in theory

Locator/ID Separation Protocol (LISP)

Design Goals*

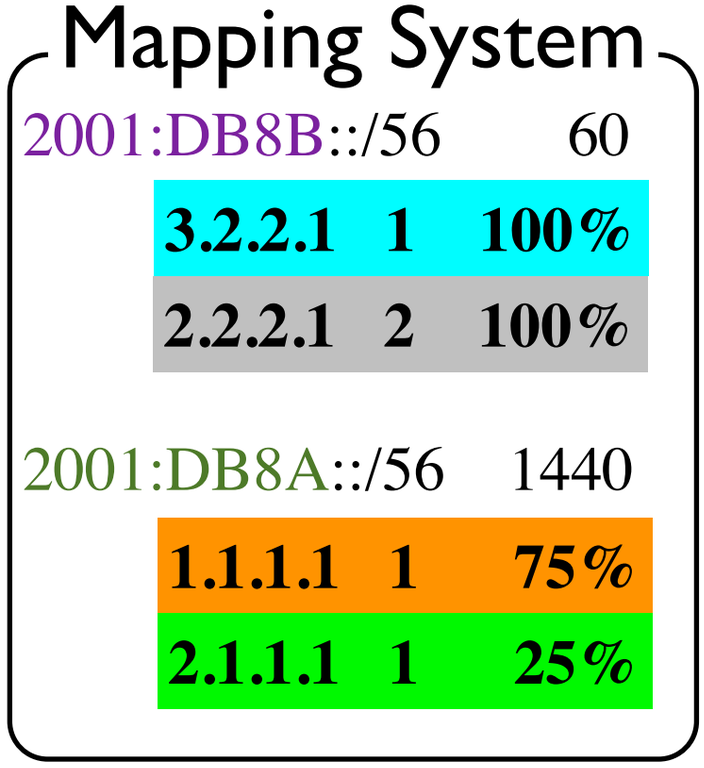
- No end-systems (hosts) changes (1)
- Minimize required changes to the Internet infrastructure (2) and the number of routers which have to be modified (5)
- Avoid or minimize packet loss when EID-to-RLOC mappings need to be performed (7)
- Be incrementally deployable (3)
- No router hardware change (4) and minimize router software changes (6)

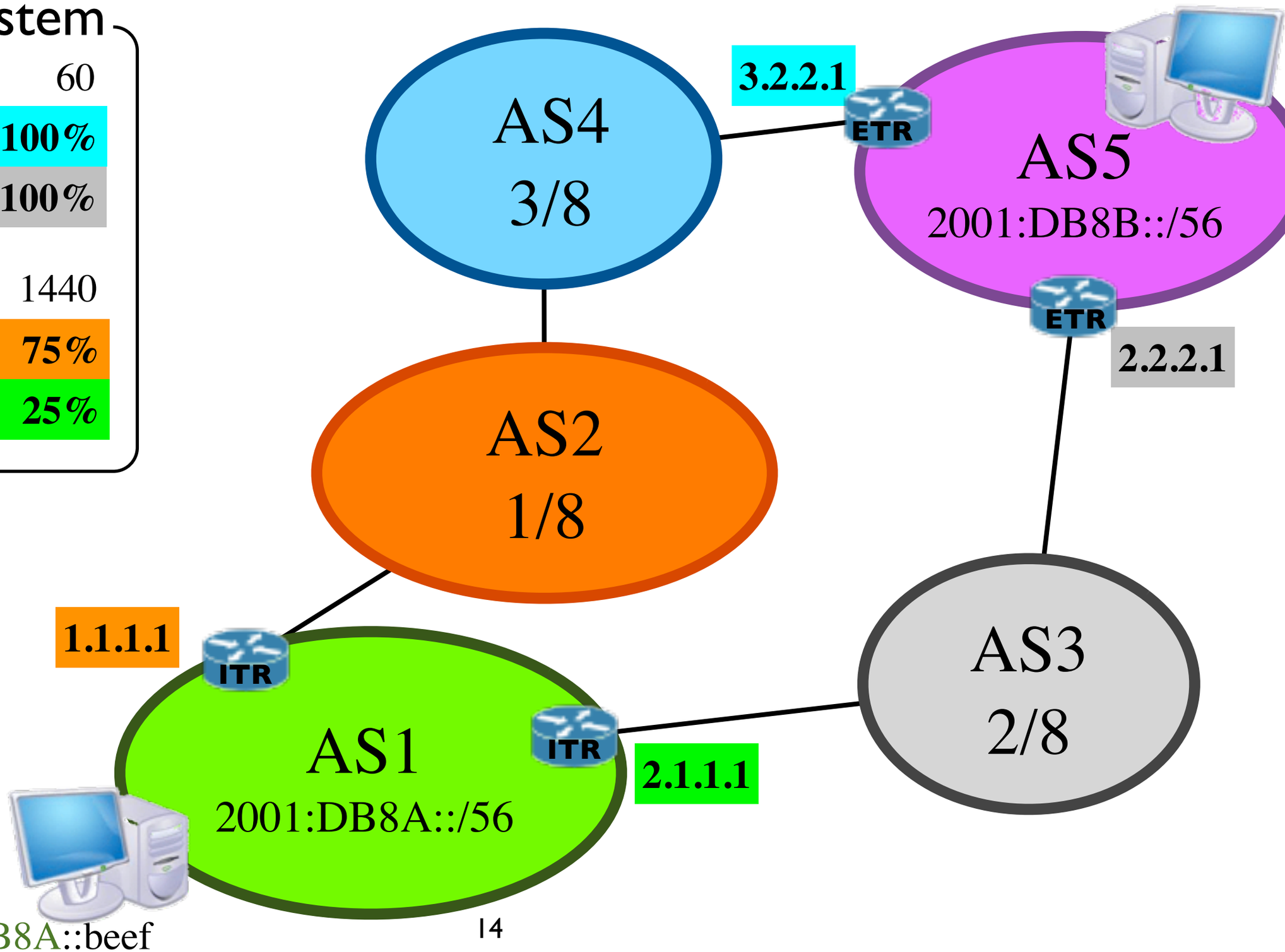
*explicitly stated until -08

LISP Philosophy

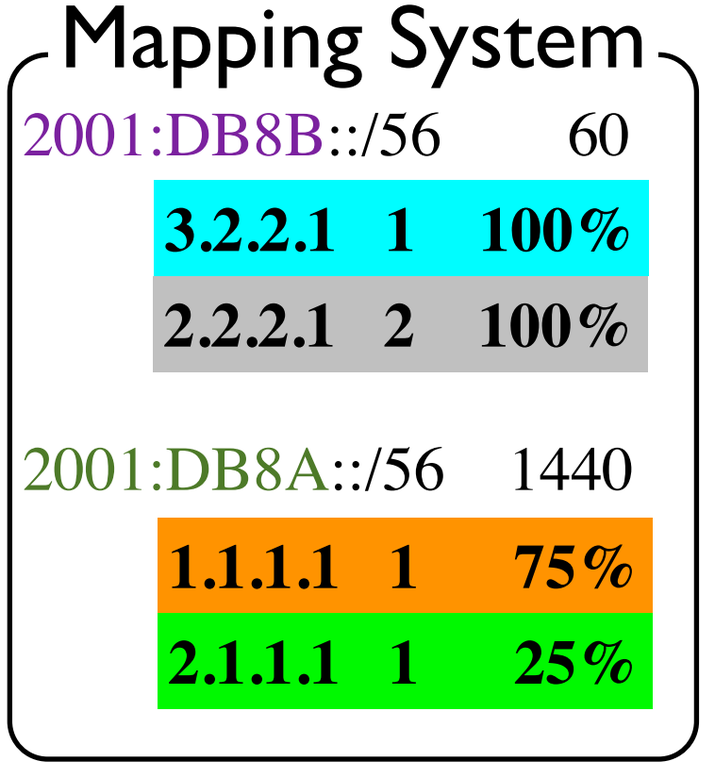
- Split the IP address space in two at the border routers
  - **Endpoint IDentifiers (EID)**
    - identify end-systems and edge routers
    - non-globally routable
    - end systems in a site share the same EID prefix
  - **Routing LOCators (RLOC)**
    - attached to core routers (router interfaces)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

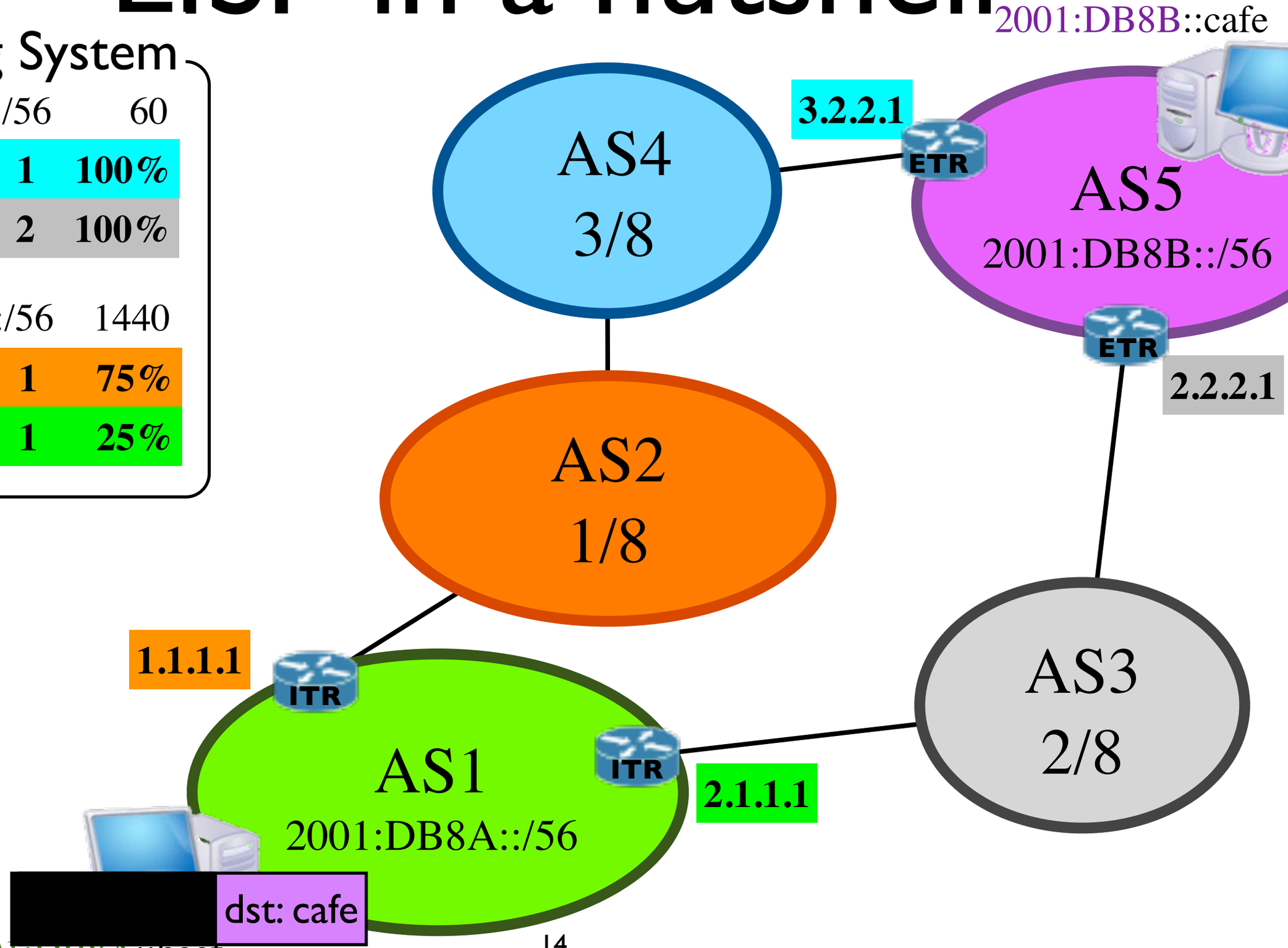
LISP in a nutshell



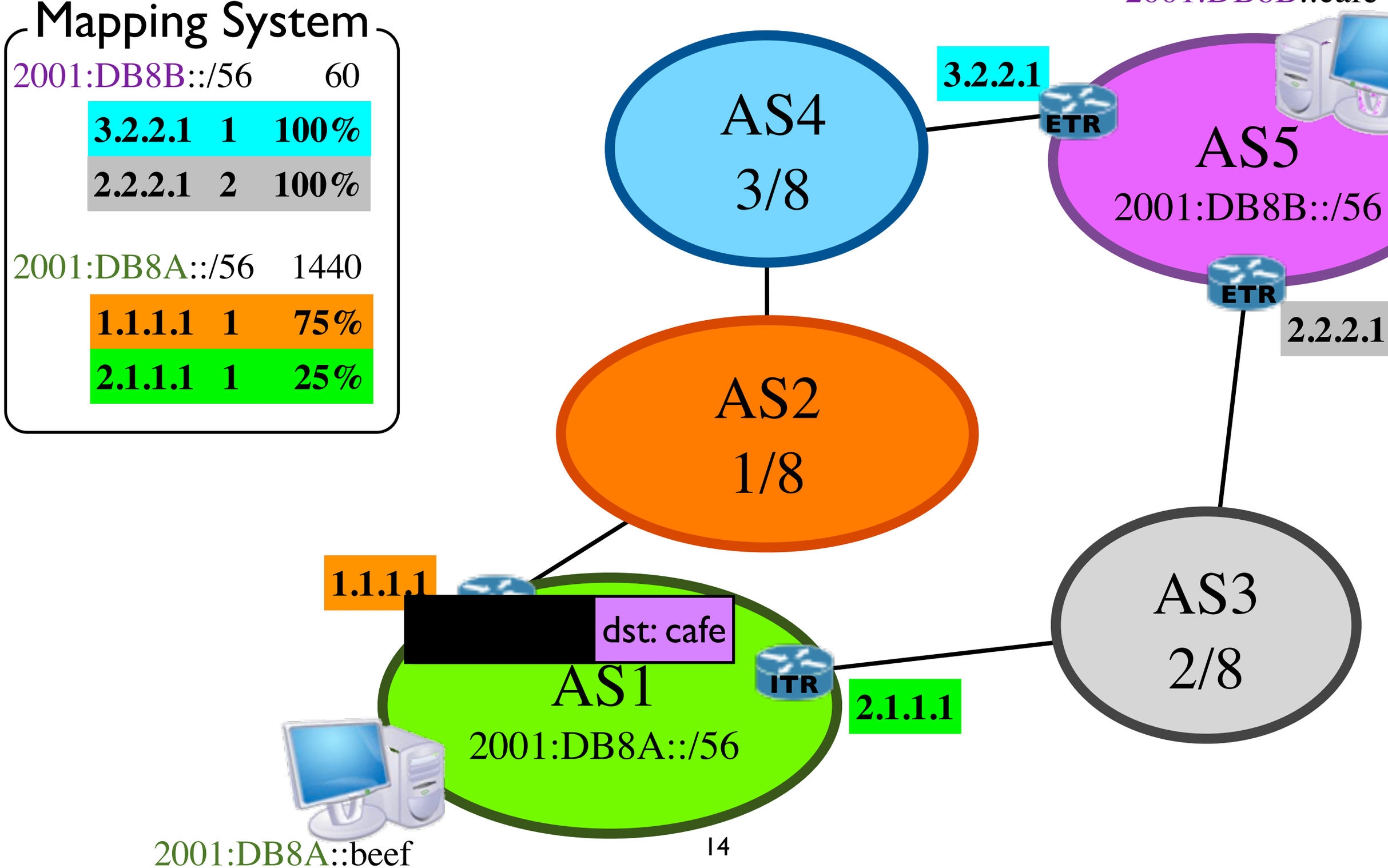


LISP in a nutshell

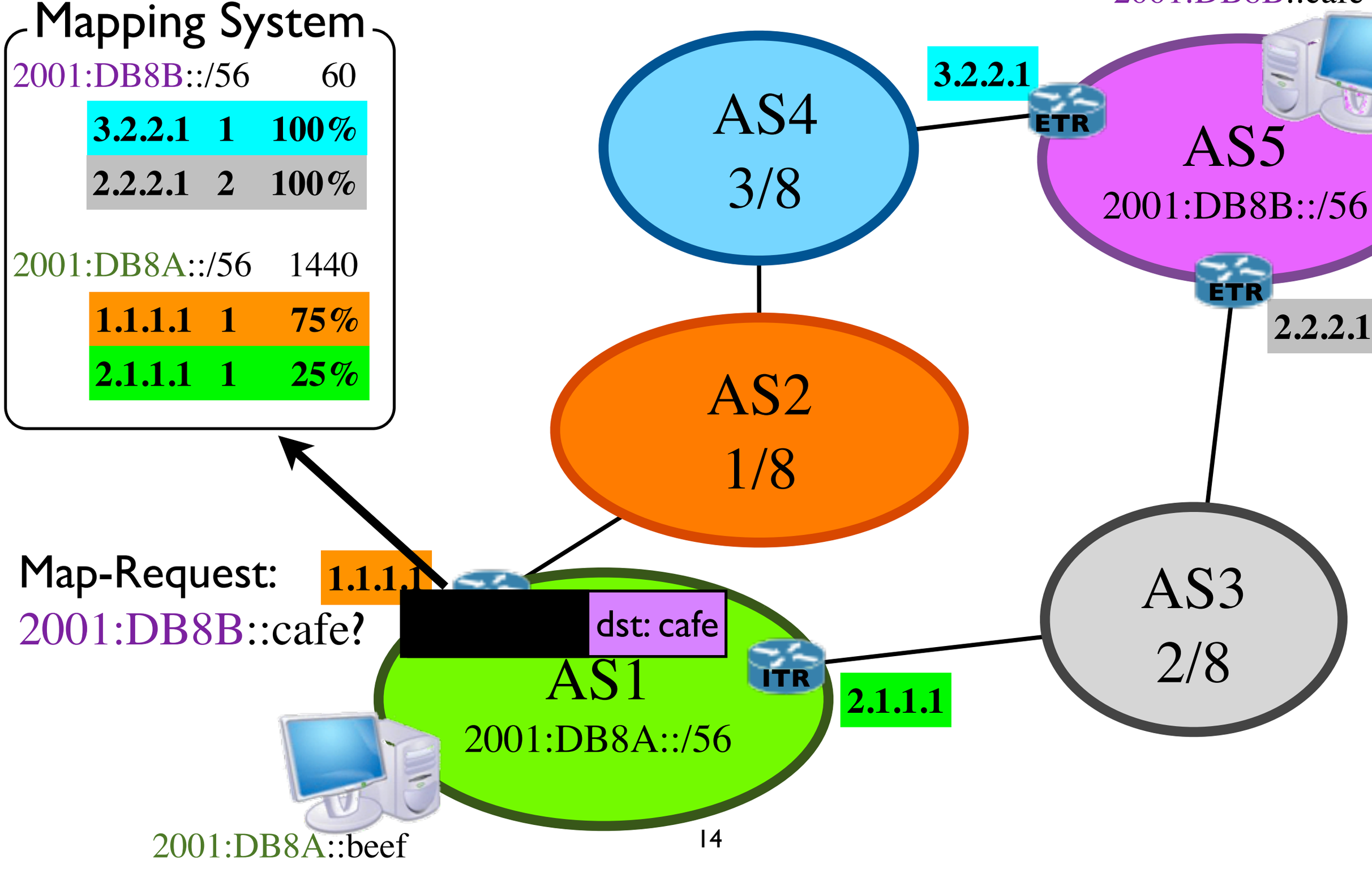




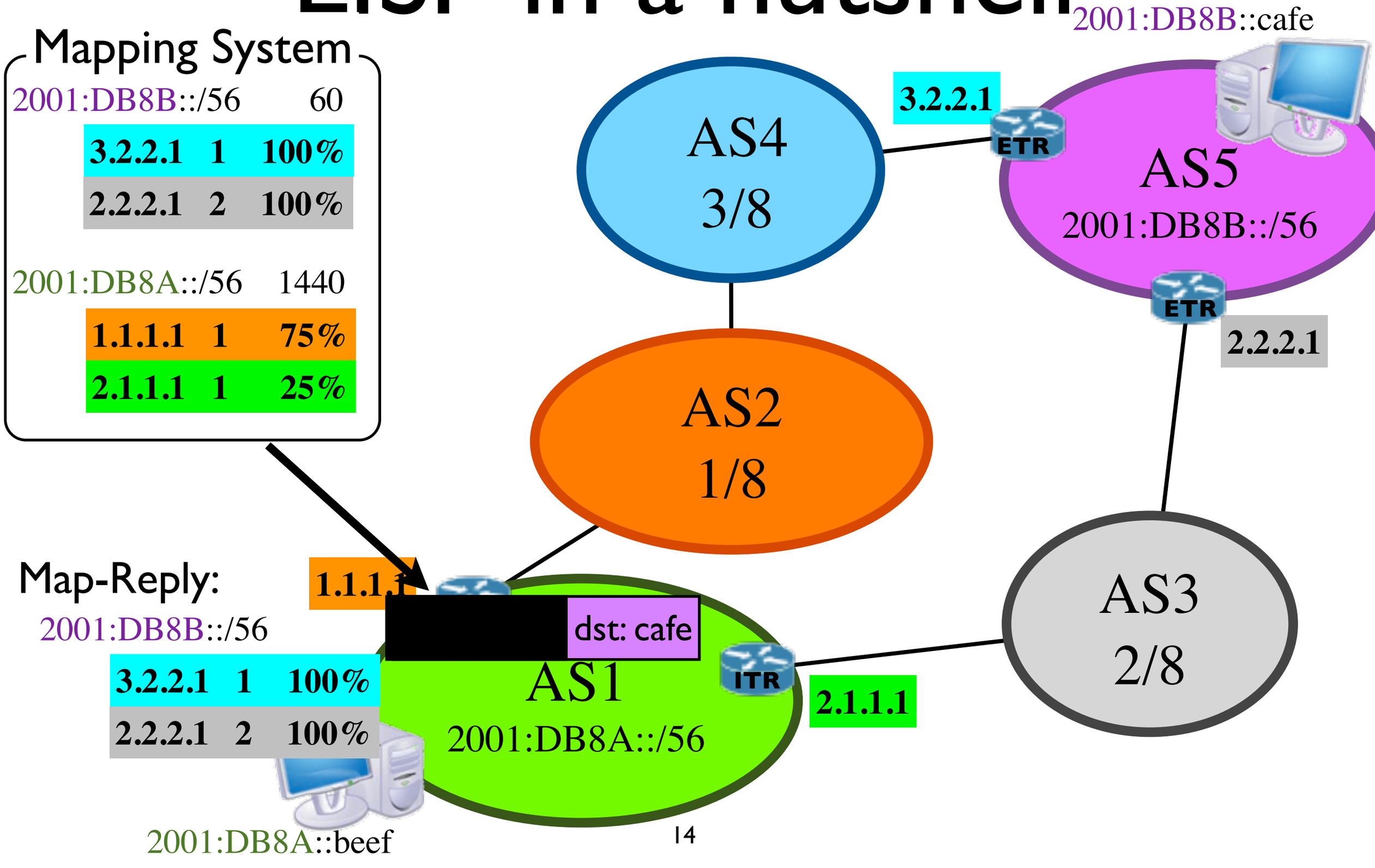
LISP in a nutshell



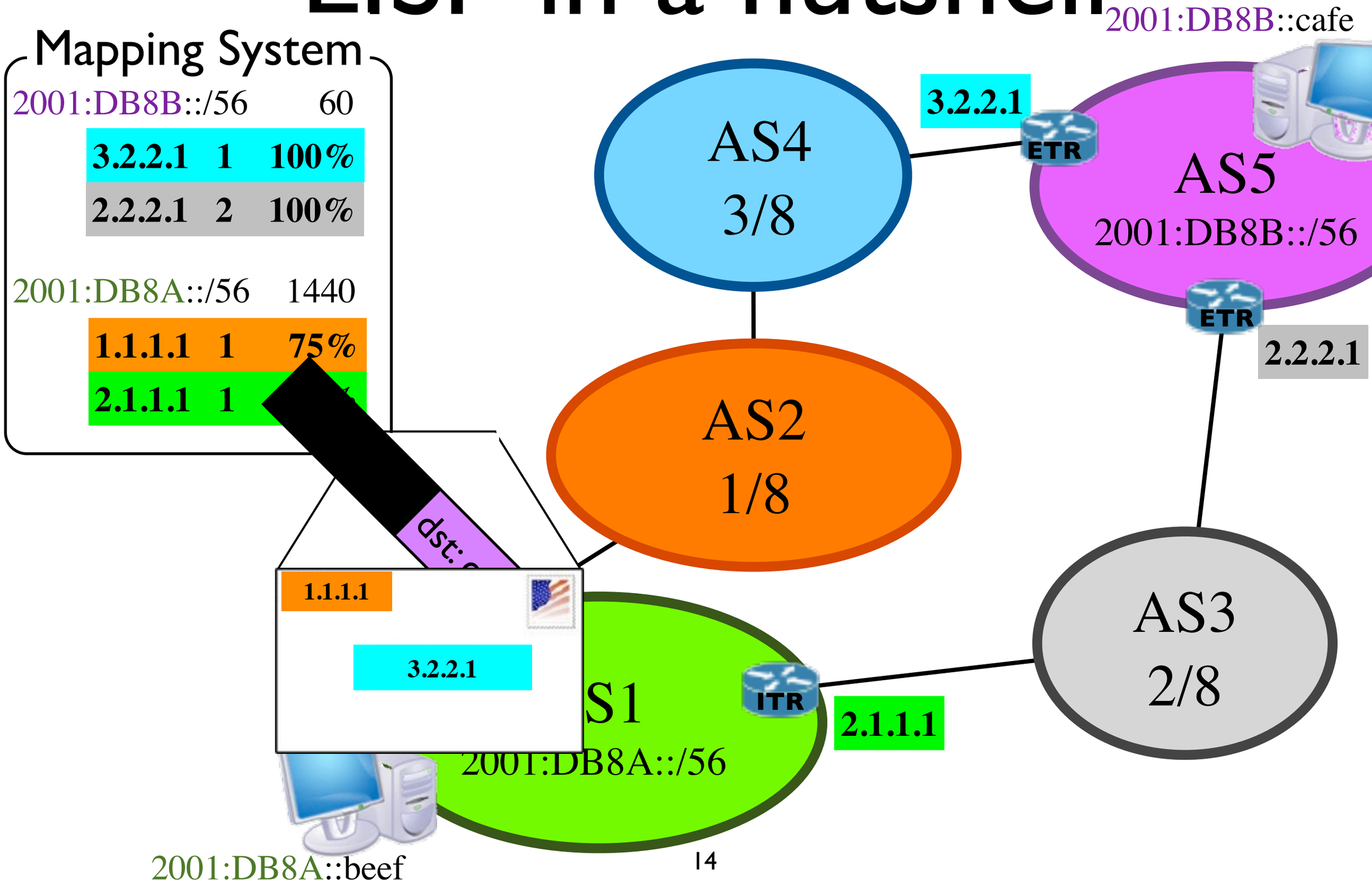
LISP in a nutshell



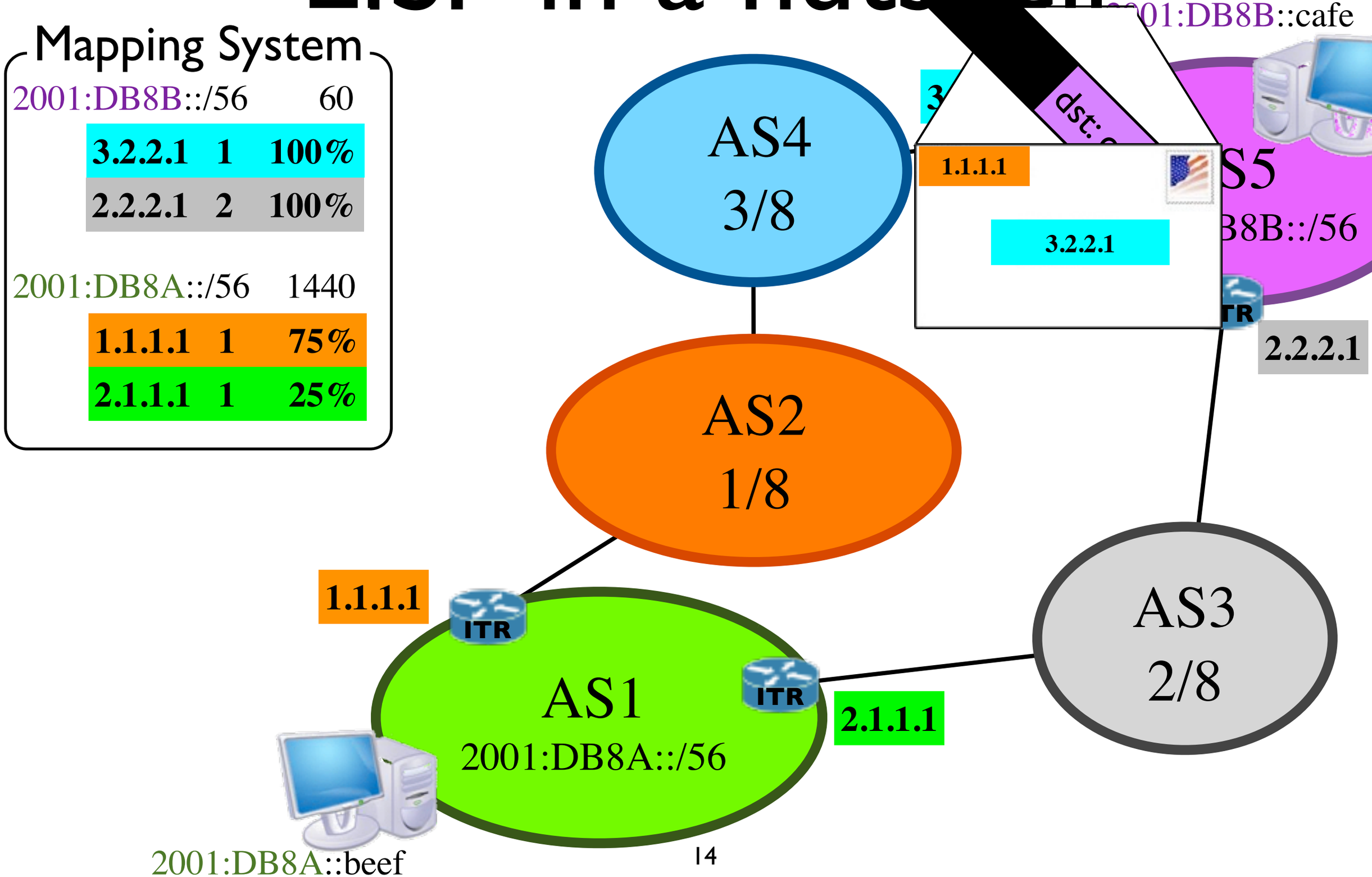
LISP in a nutshell



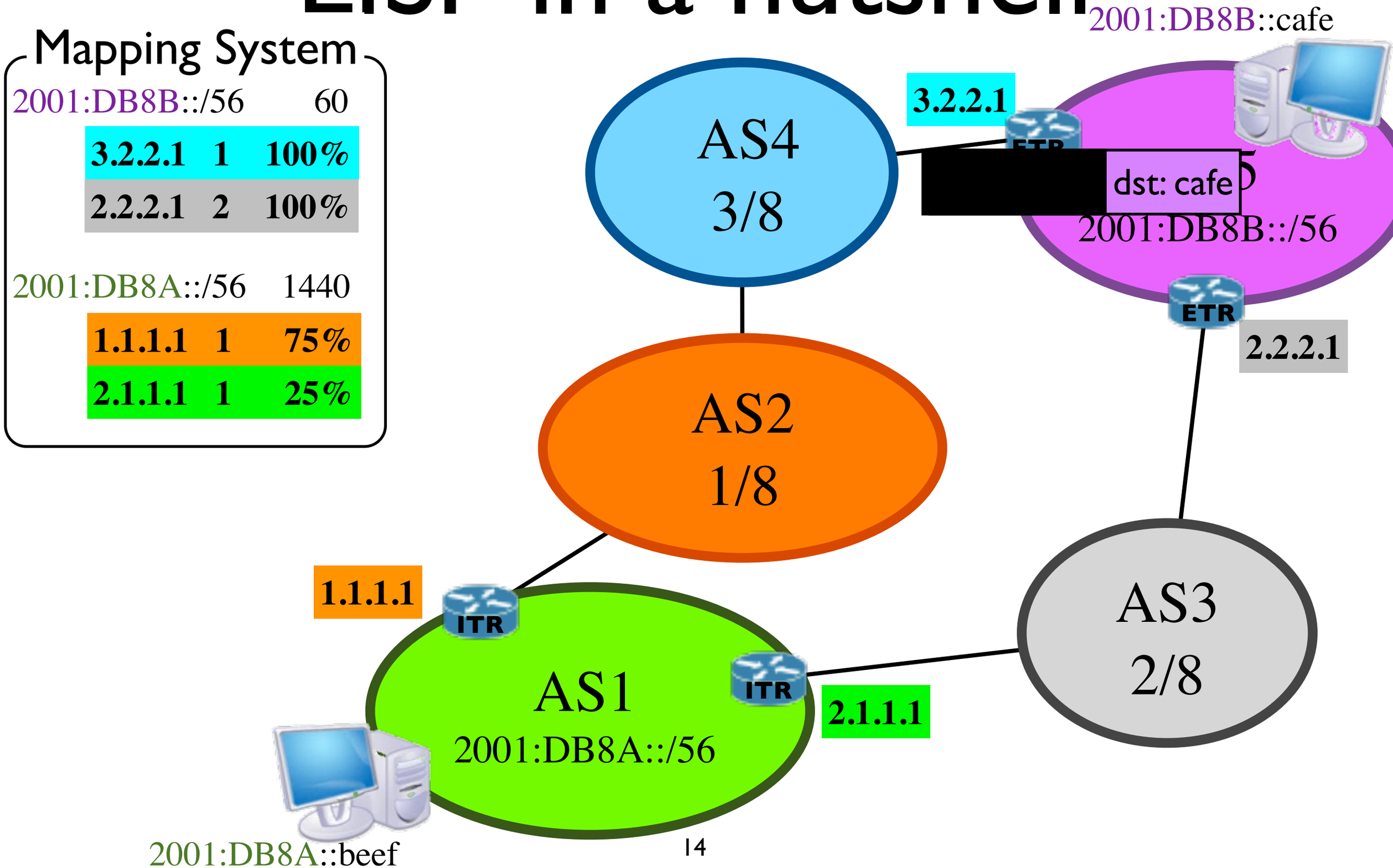
LISP in a nutshell



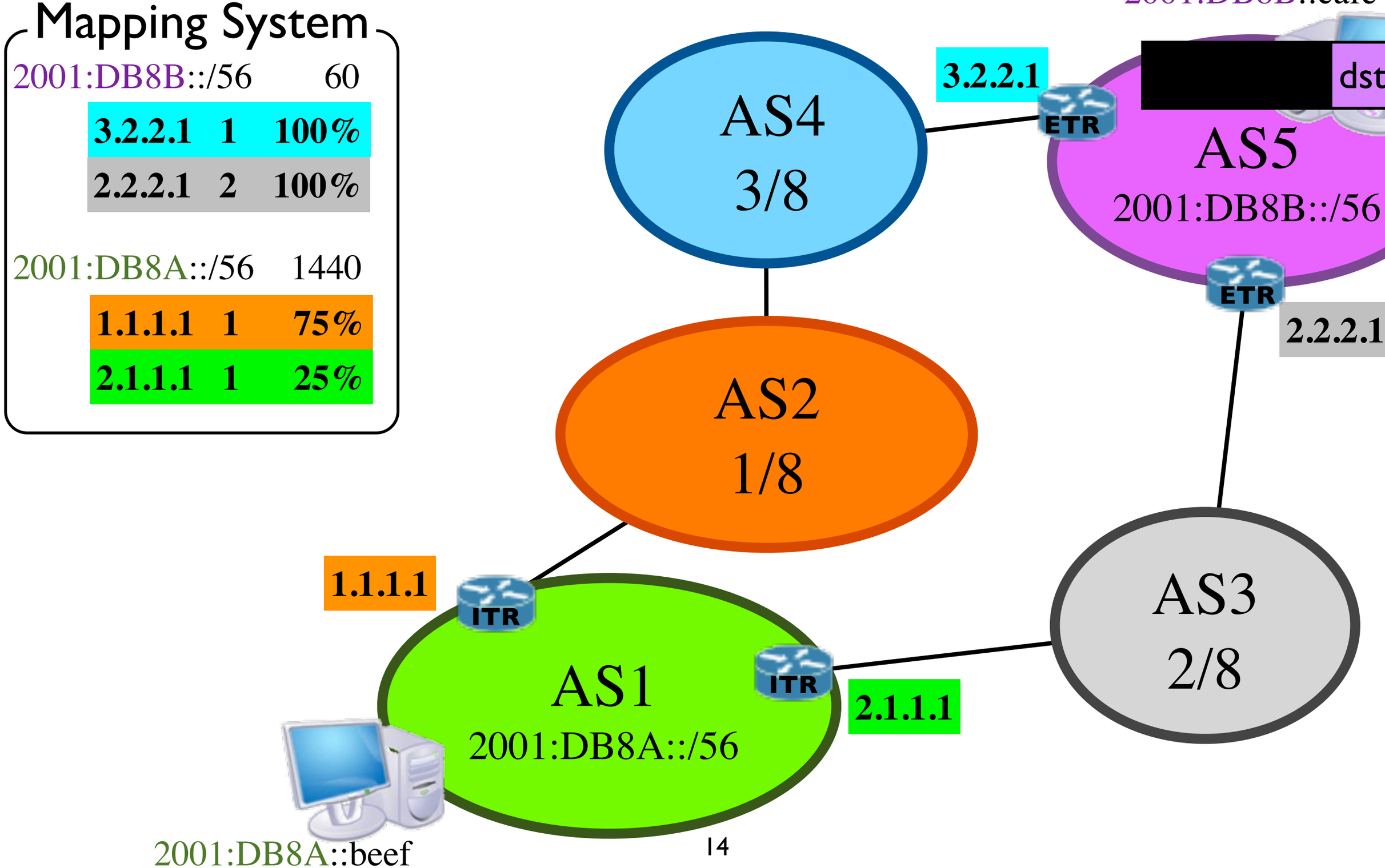
LISP in a nutshell



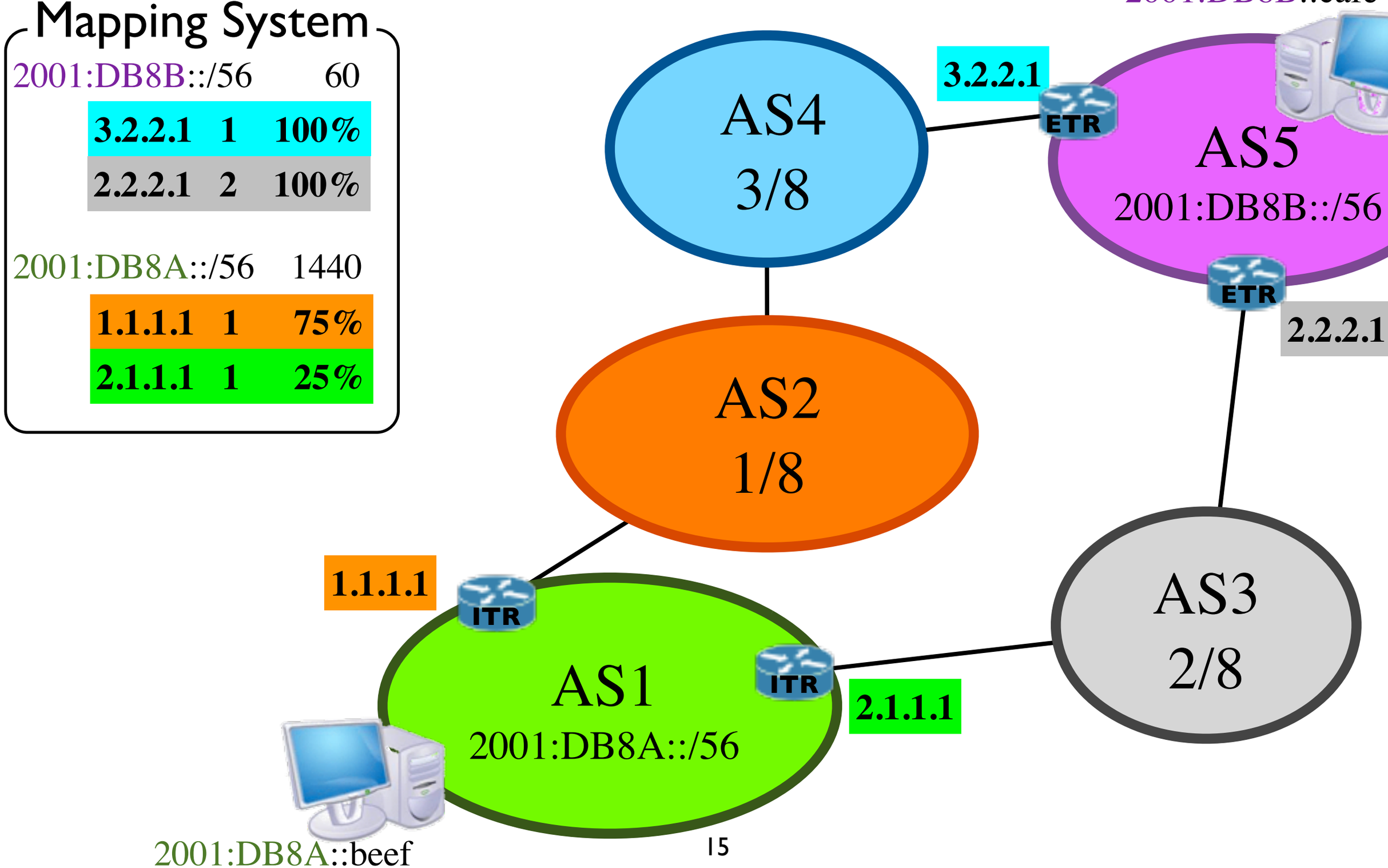
LISP in a nutshell



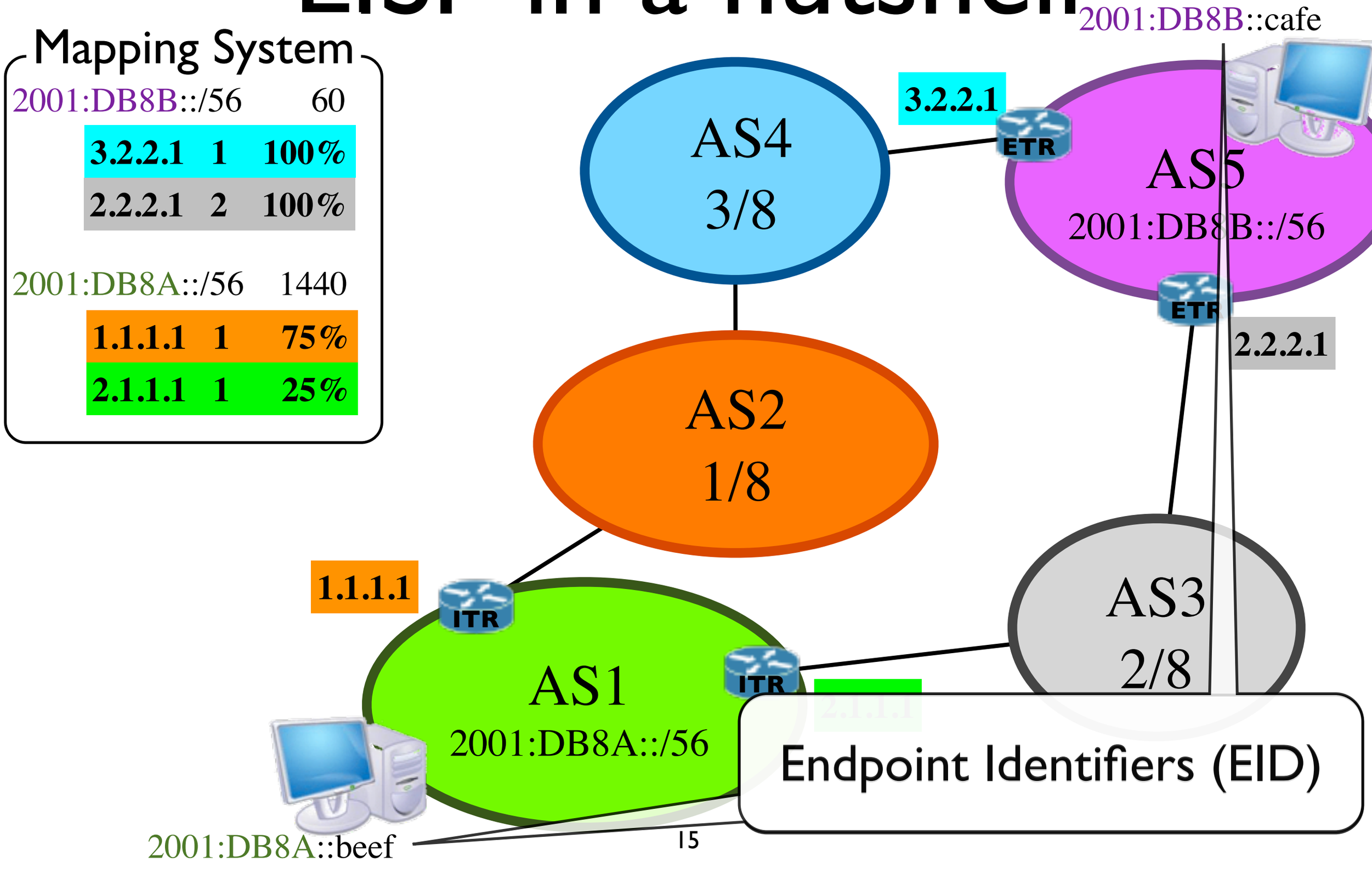
LISP in a nutshell



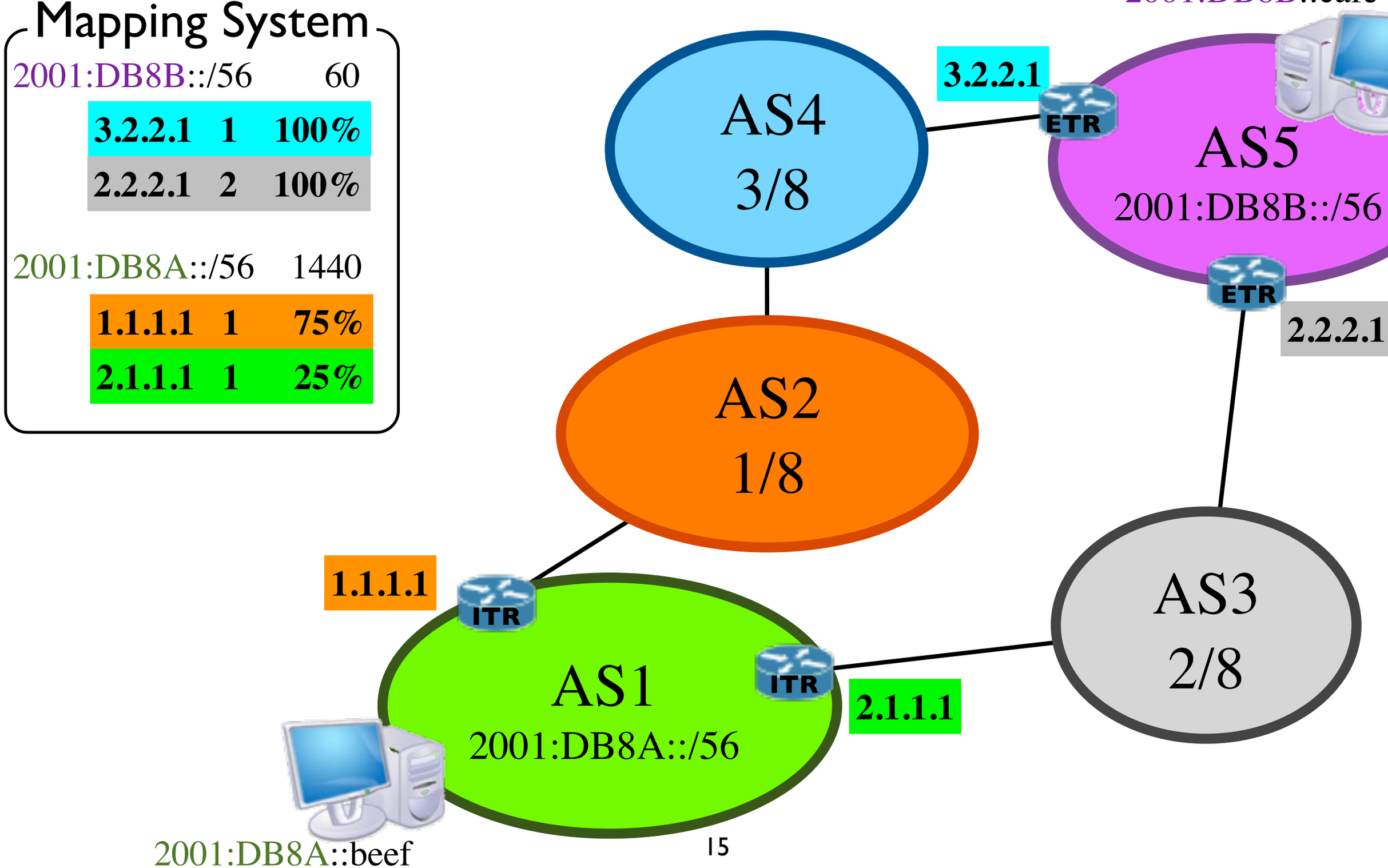
LISP in a nutshell



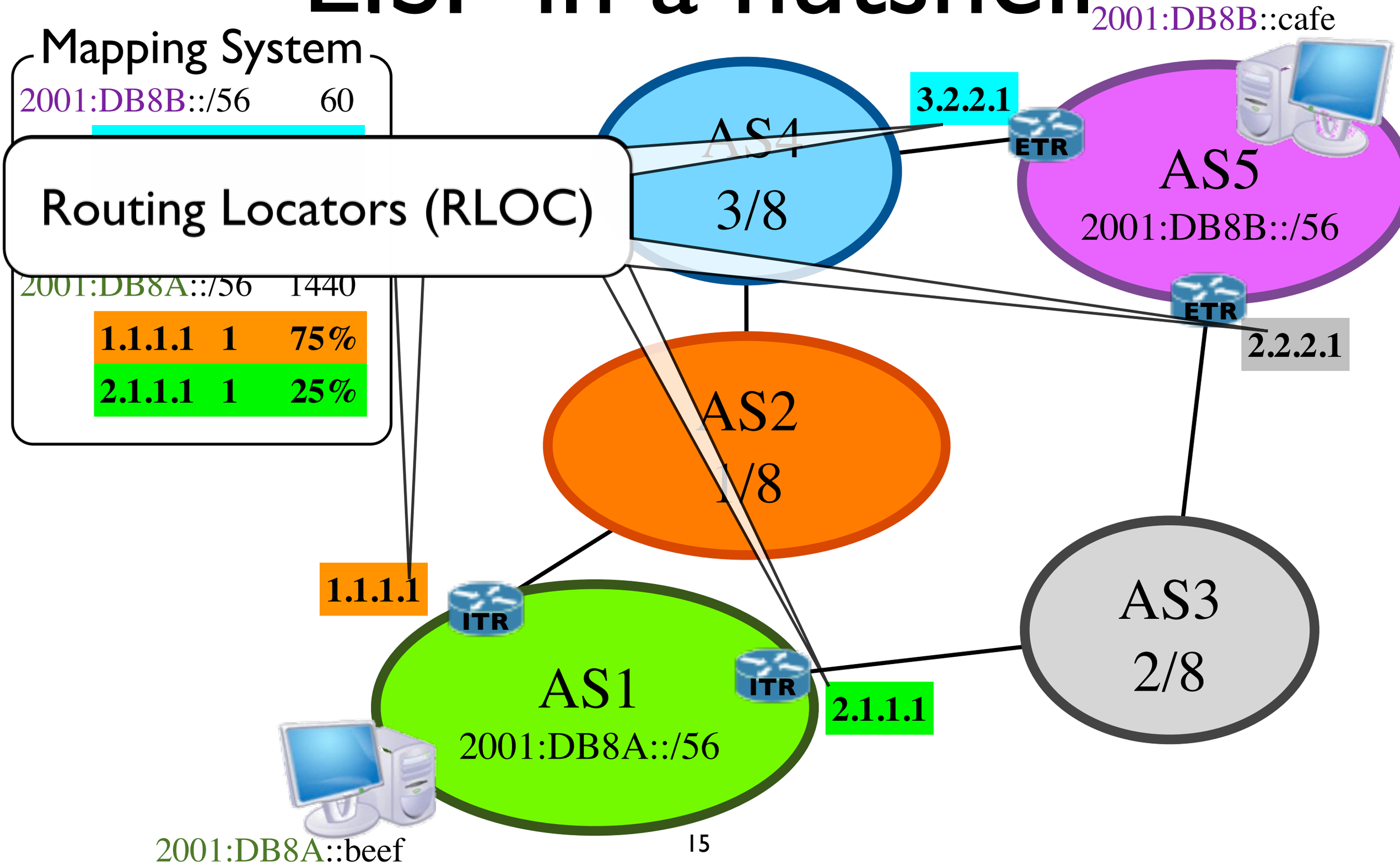
LISP in a nutshell



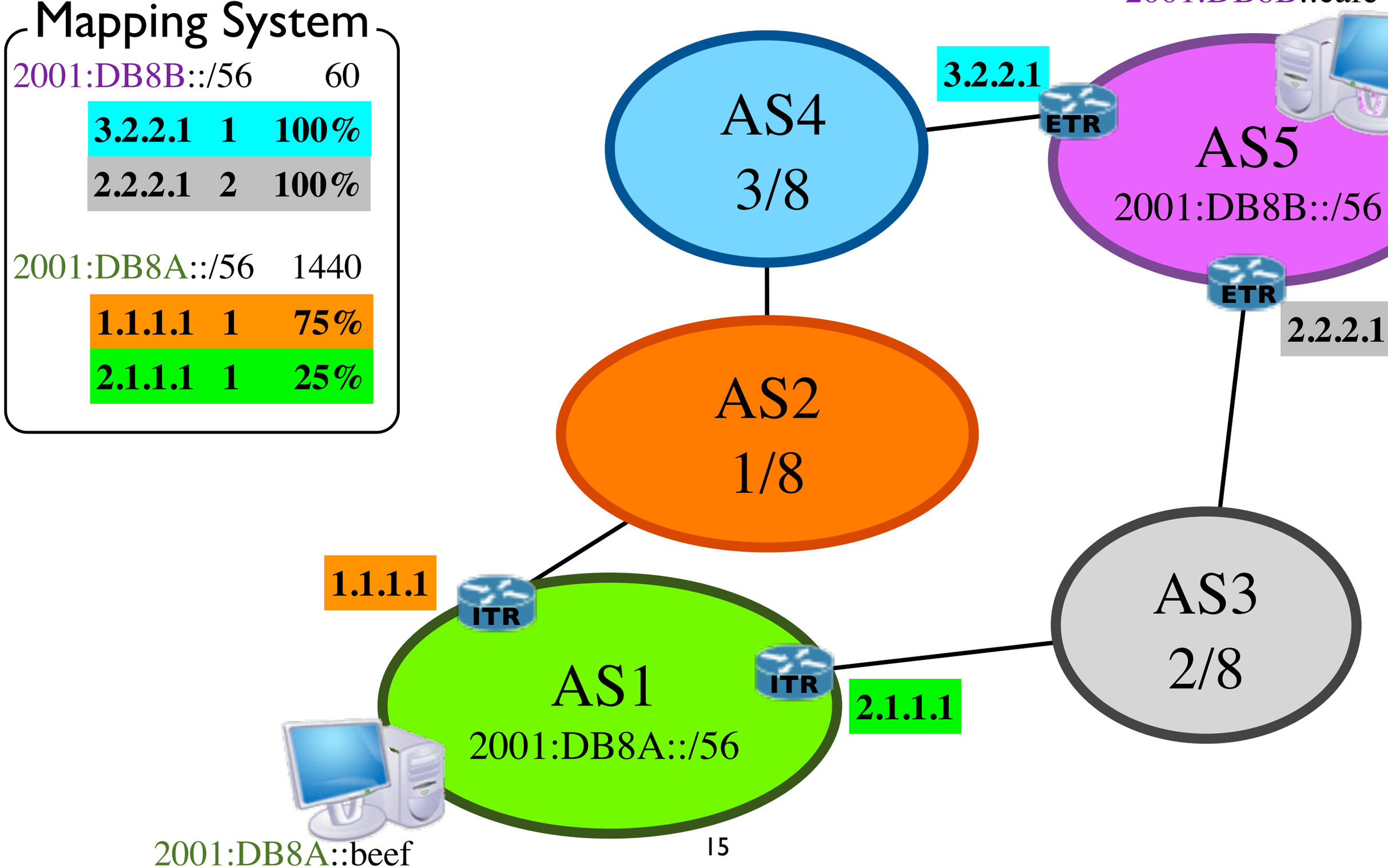
LISP in a nutshell



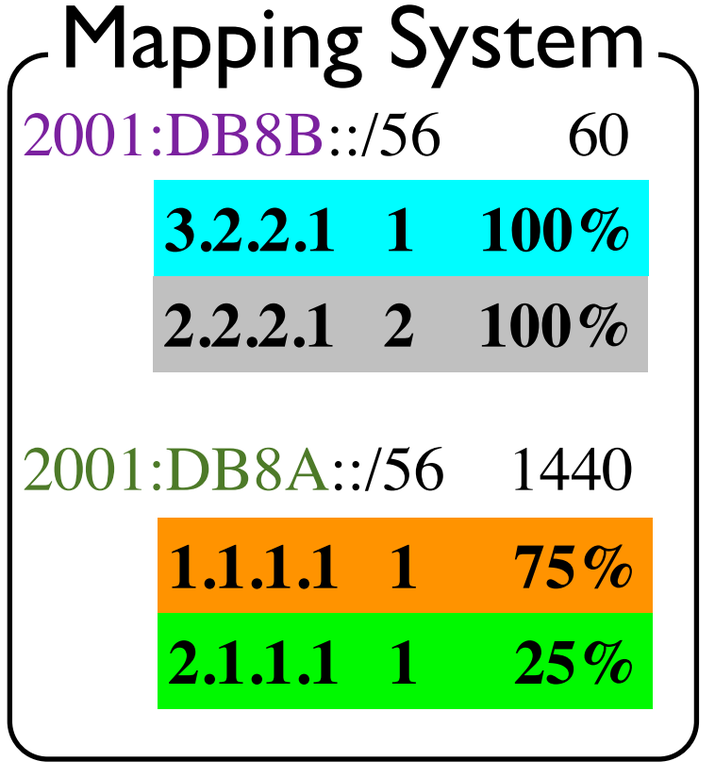
LISP in a nutshell

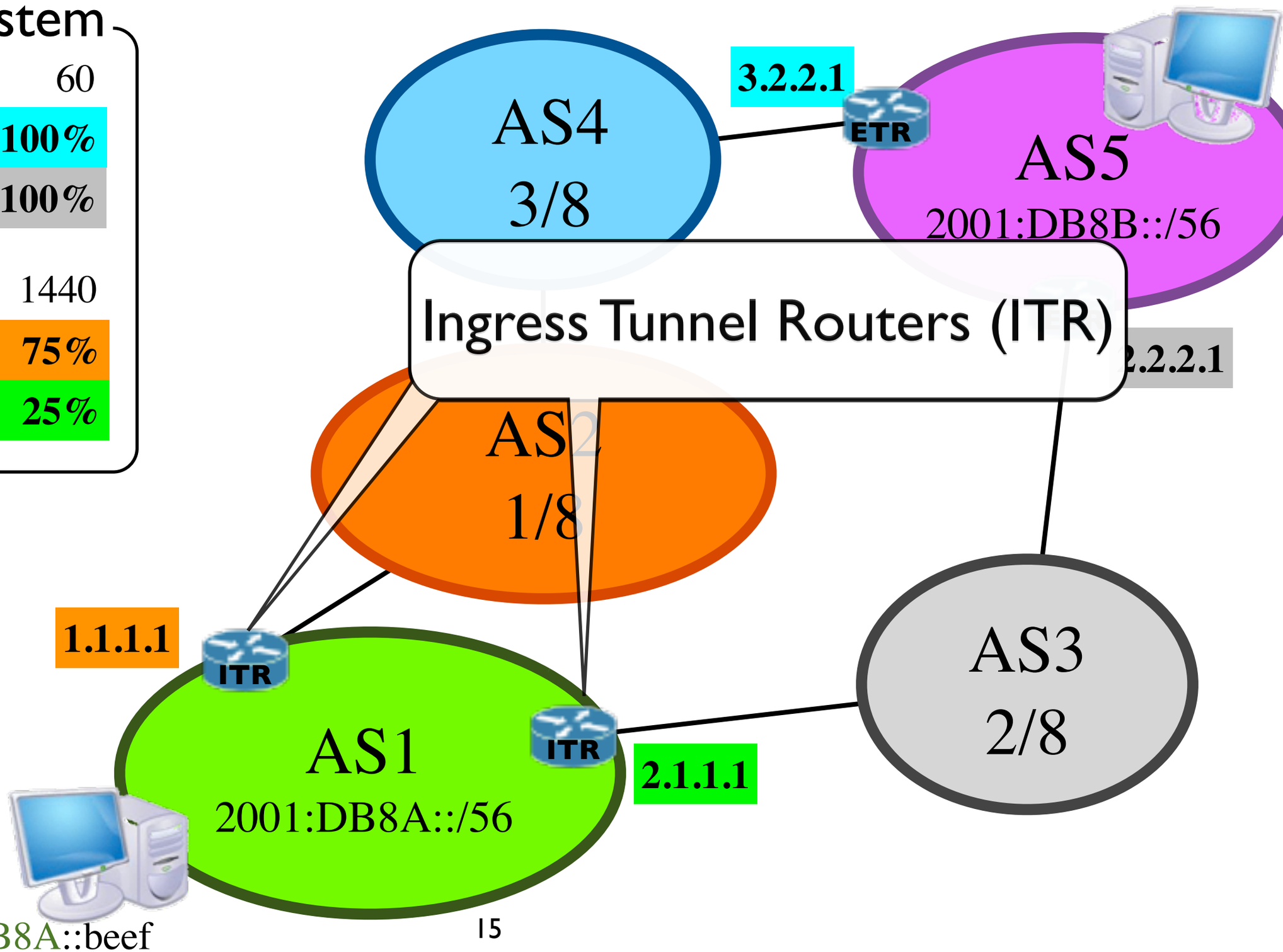


LISP in a nutshell

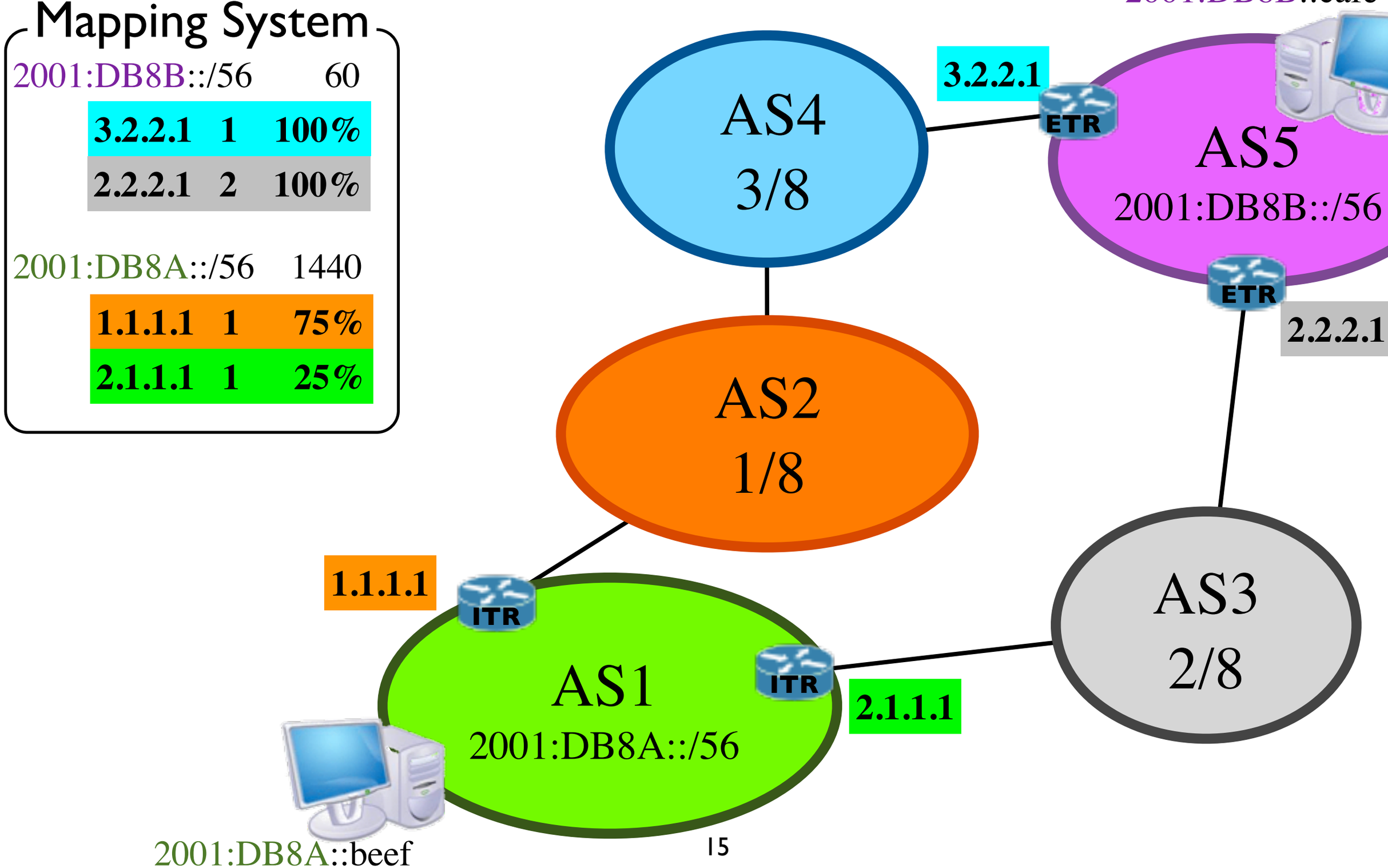


LISP in a nutshell

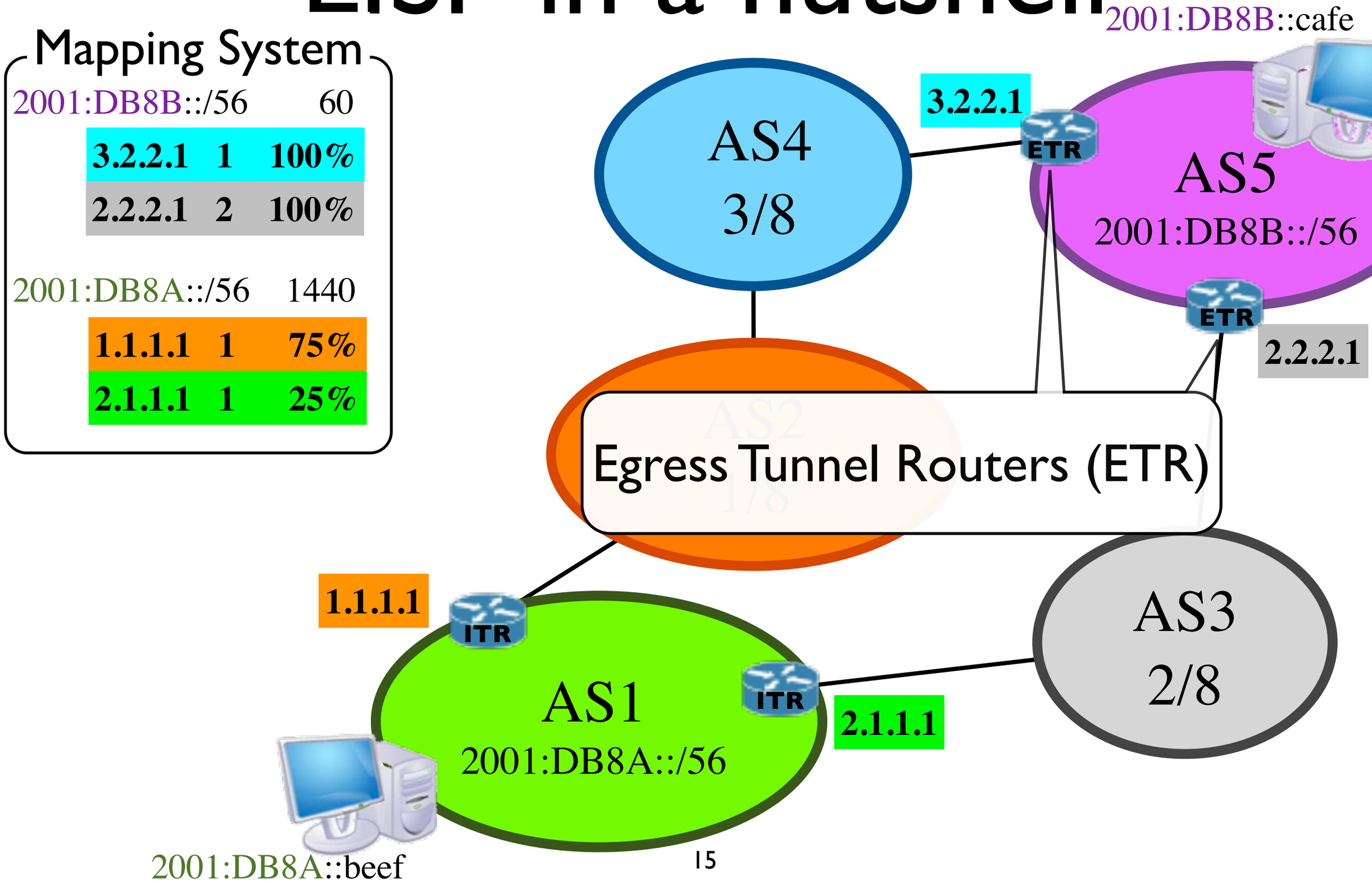




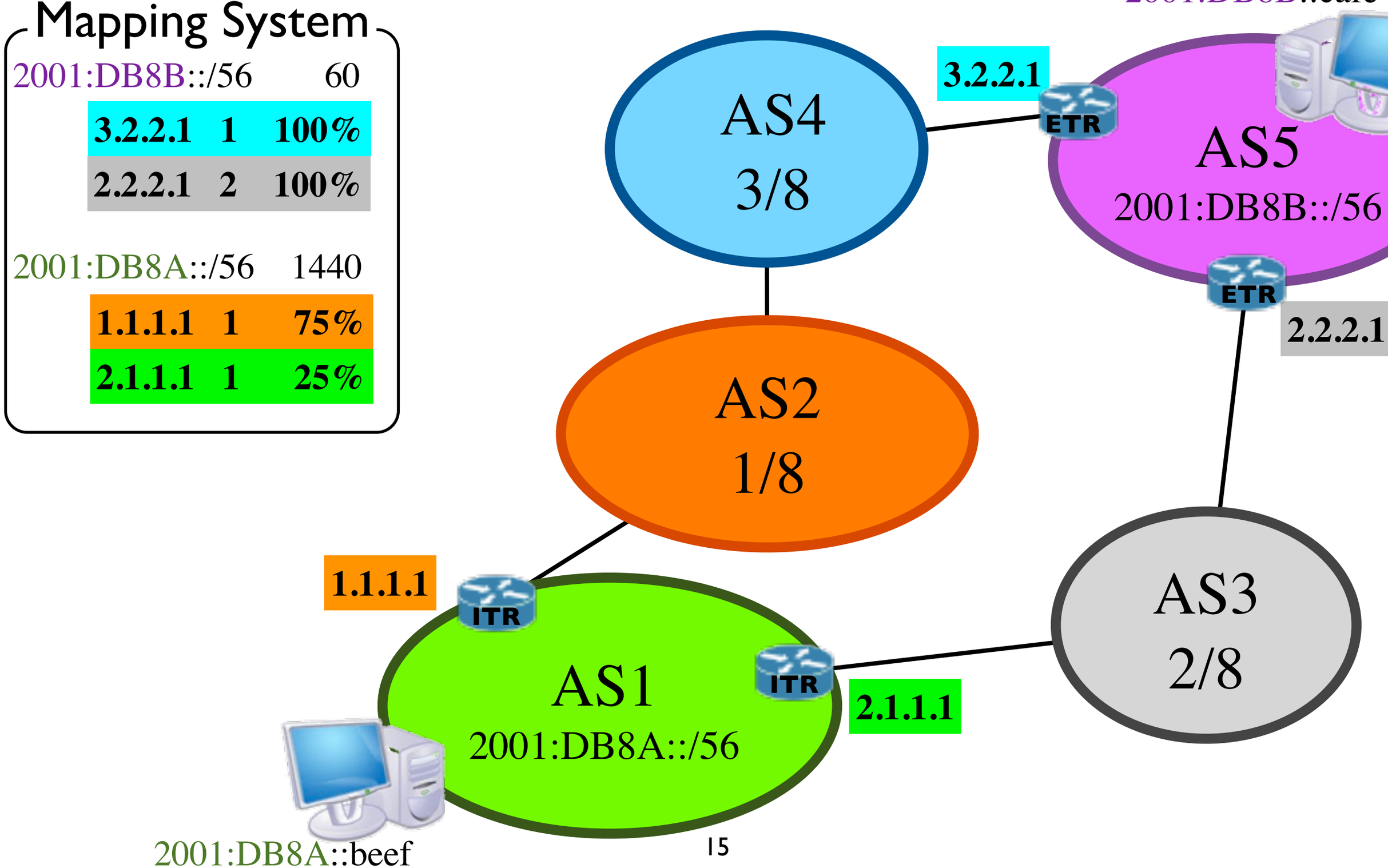
LISP in a nutshell



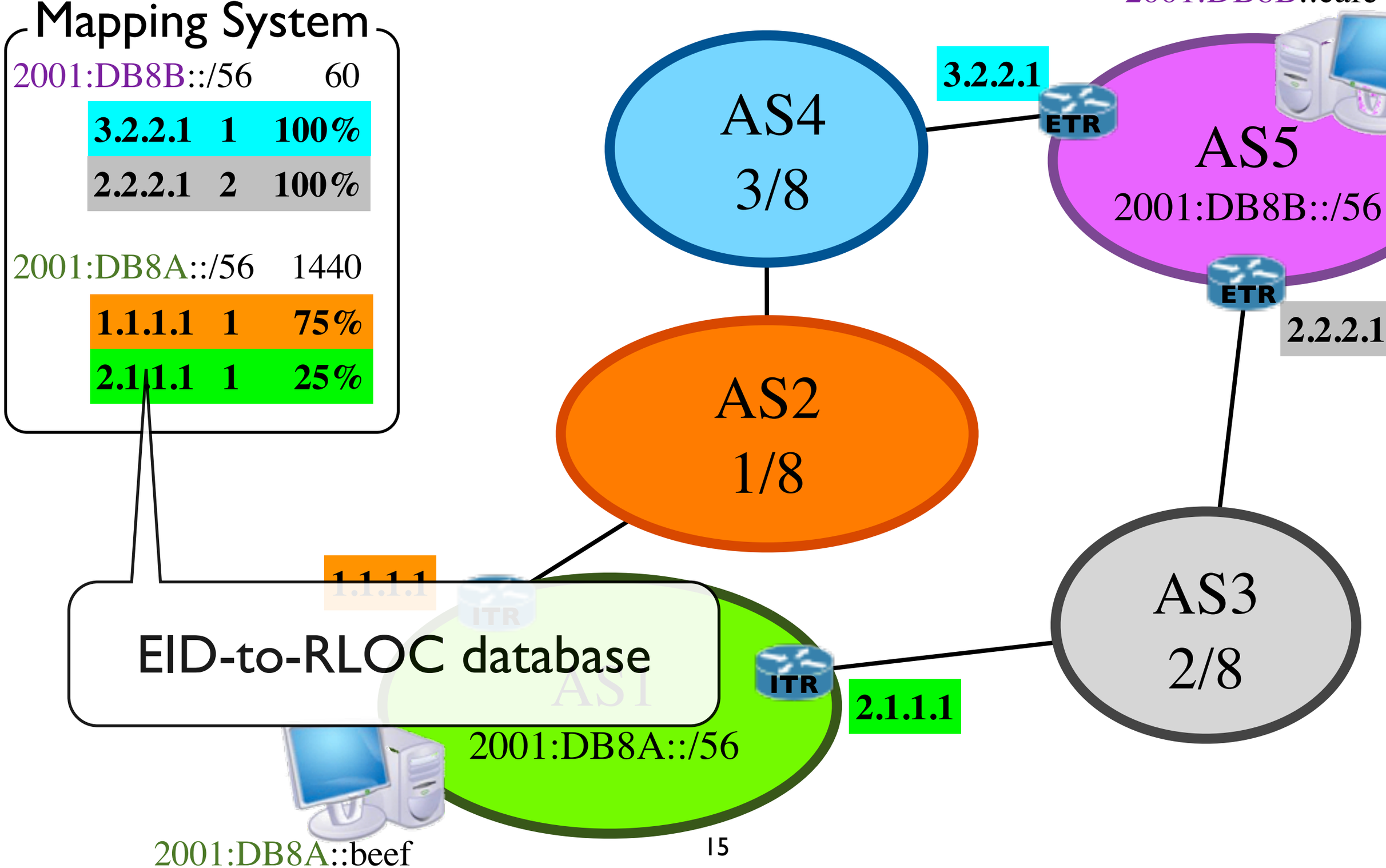
LISP in a nutshell



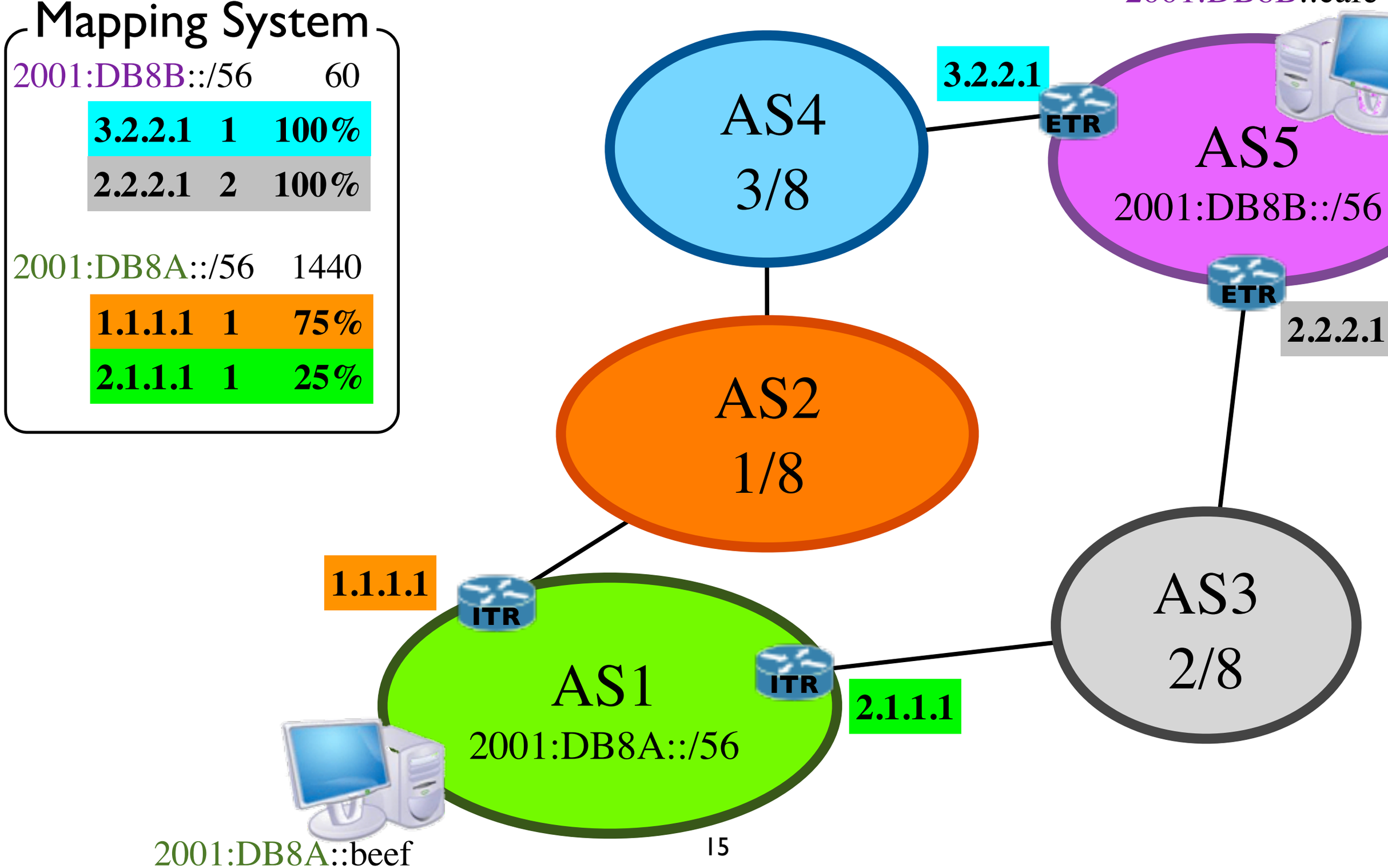
LISP in a nutshell



LISP in a nutshell



LISP in a nutshell



Terminology

- **Ingress Tunnel Router (ITR)**: a router which accepts a packet which addresses are identifiers. 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

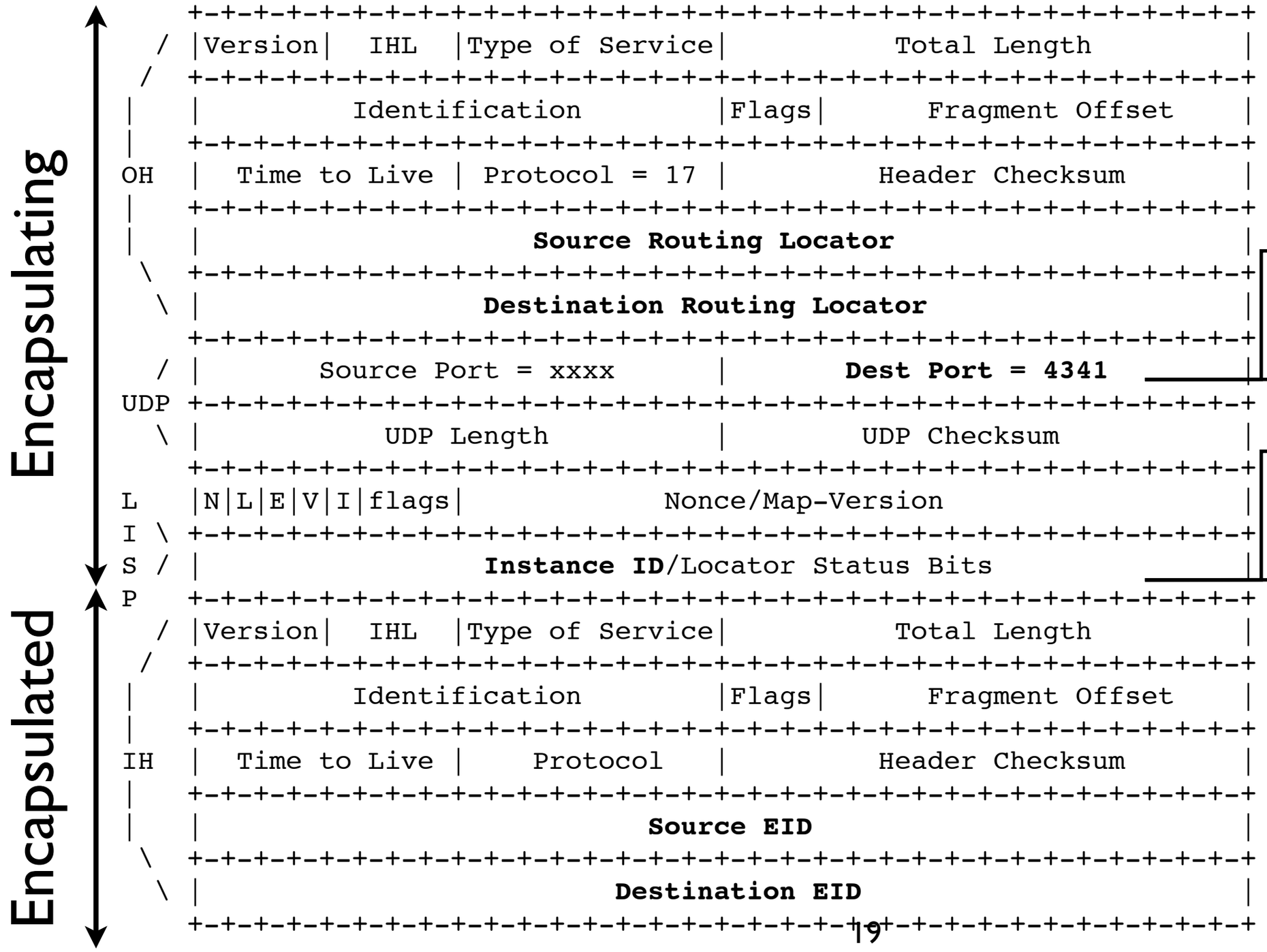
Terminology

- **EID-to-RLOC Database**: a globally distributed database that contains all known EID-prefix to RLOC mappings
- **LISP Cache**: EID-to-RLOC Database stored at the ITR
- **LISP Database**: EID-to-RLOC Database stored at the ETR

Under the hood

Data-plane packets

(IP(UDP:4341(LISP(XXX))))

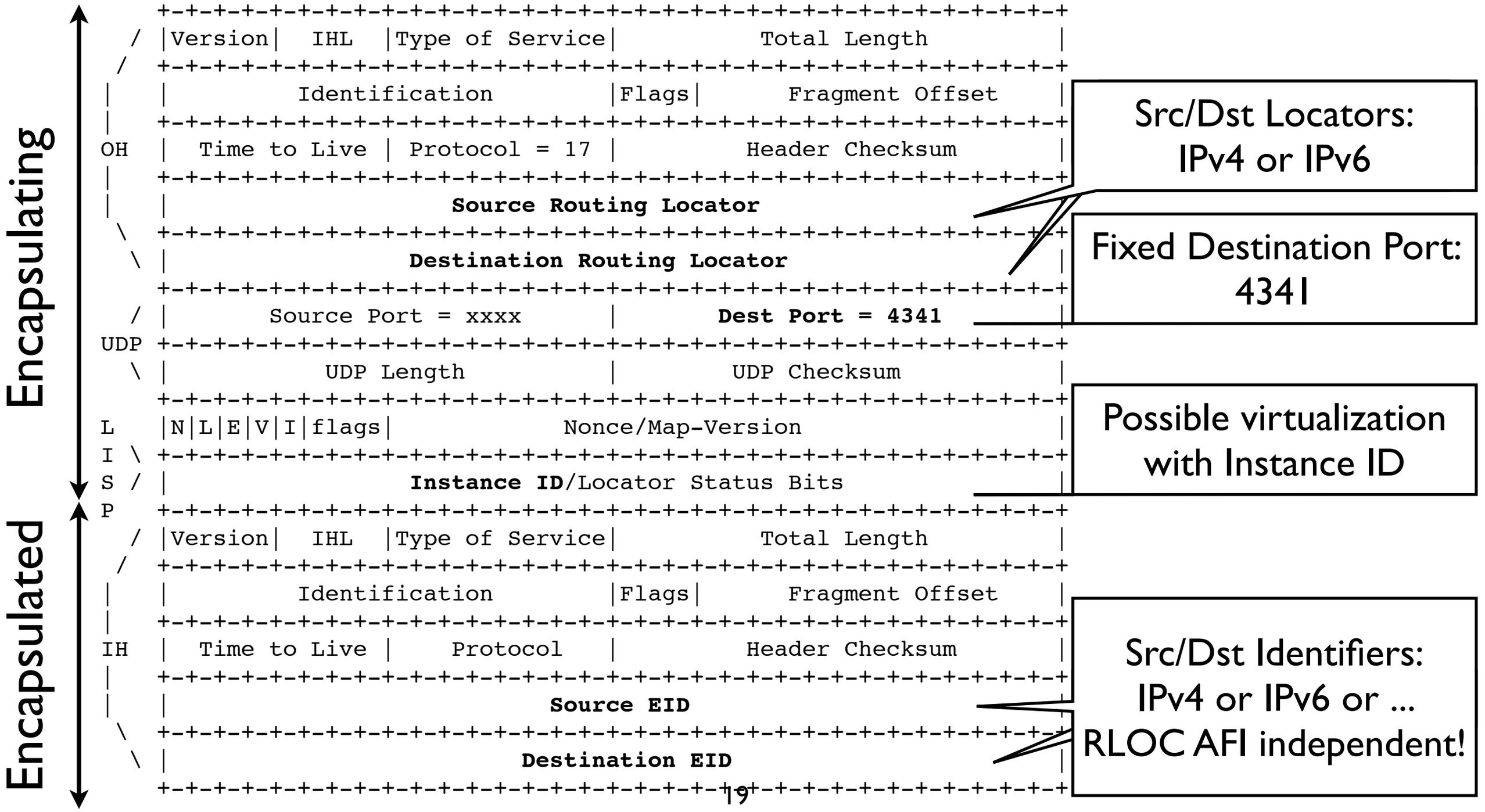


Fixed Destination Port: 4341

Possible virtualization with Instance ID

Data-plane packets

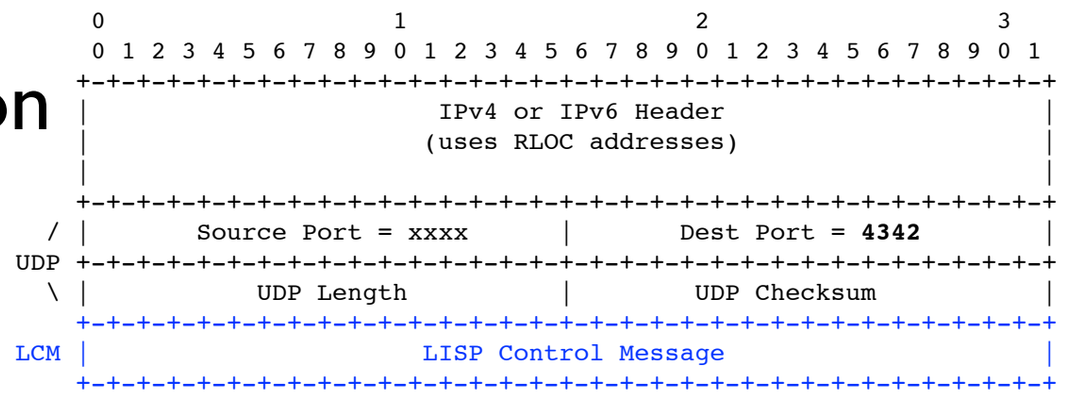
(IP(UDP:4341(LISP(XXX))))



Control Plane Packets

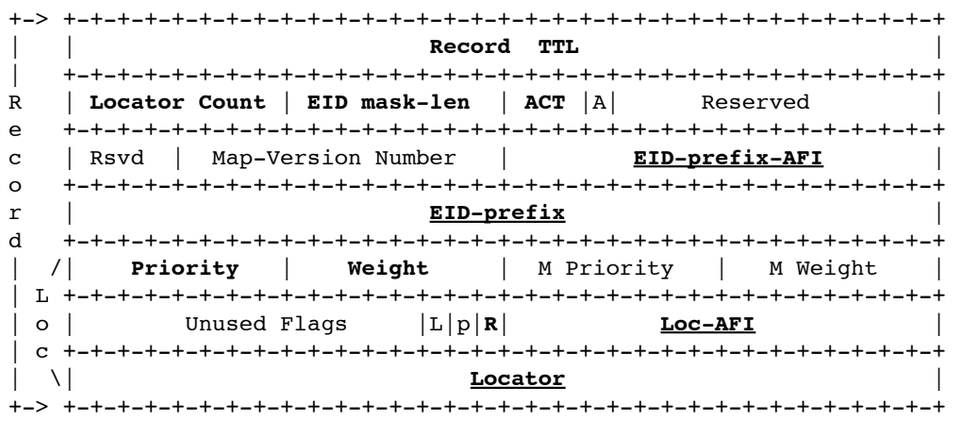
(IP(UDP:4342(LISP_control)))

- Map-Request (ITR => MR/ETR)
  - request for a mapping
- Map-Reply (ETR/MS => ITR)
  - provides the mapping requested by a Map-Request
- Map-Register (ETR => MS)
  - register a mapping to the mapping system
- Map-Notify (MS => ETR)
  - confirm a mapping registration



What is in a mapping?

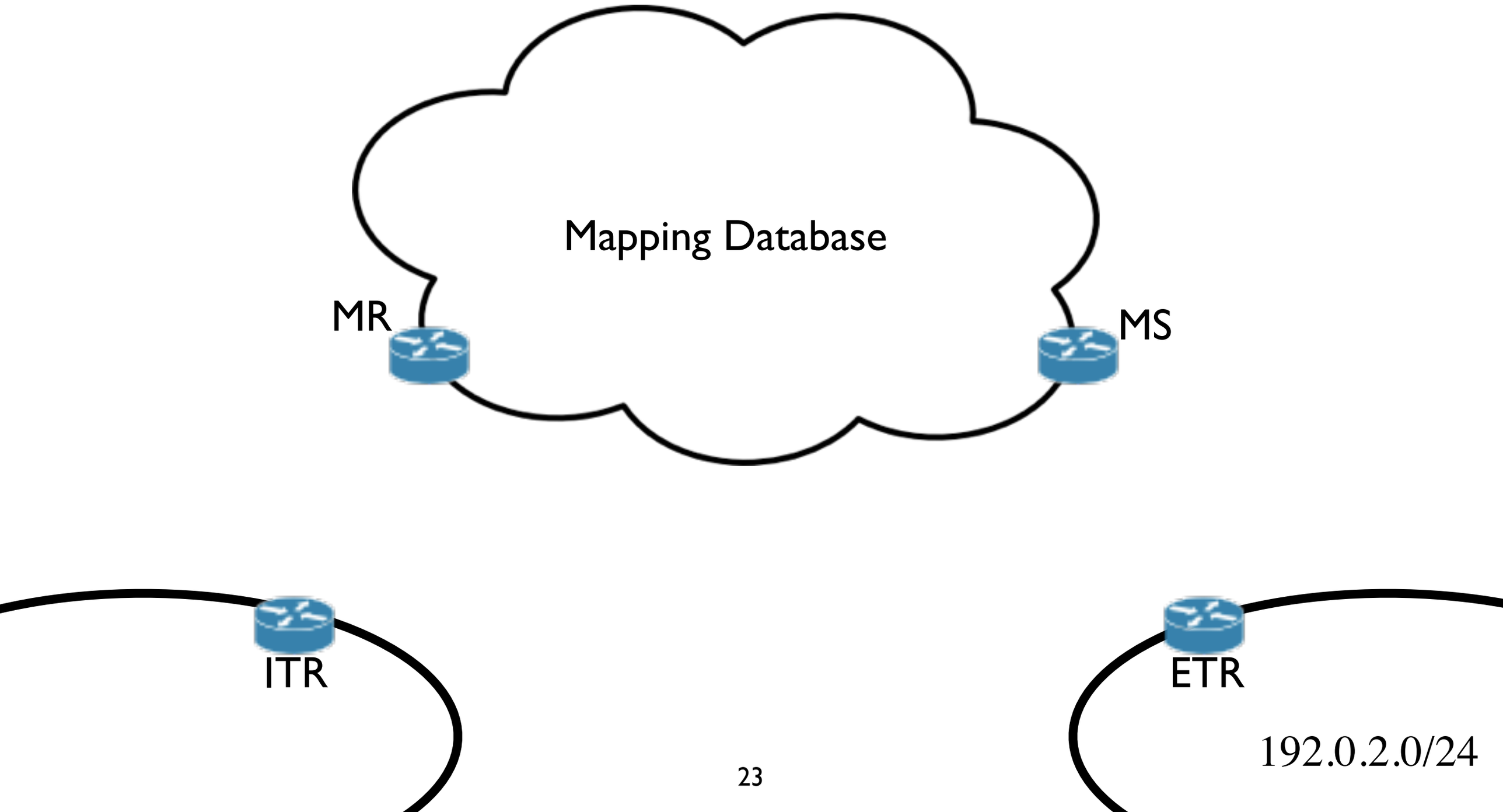
- A mapping associates an EID prefix to a list of RLOC of arbitrary AFI
  - Each RLOC receives a priority and a weight
    - L4-flows are balanced between RLOCs of same priority, proportionally to the weight
- Mappings can be empty to announce holes
- Mappings are time limited



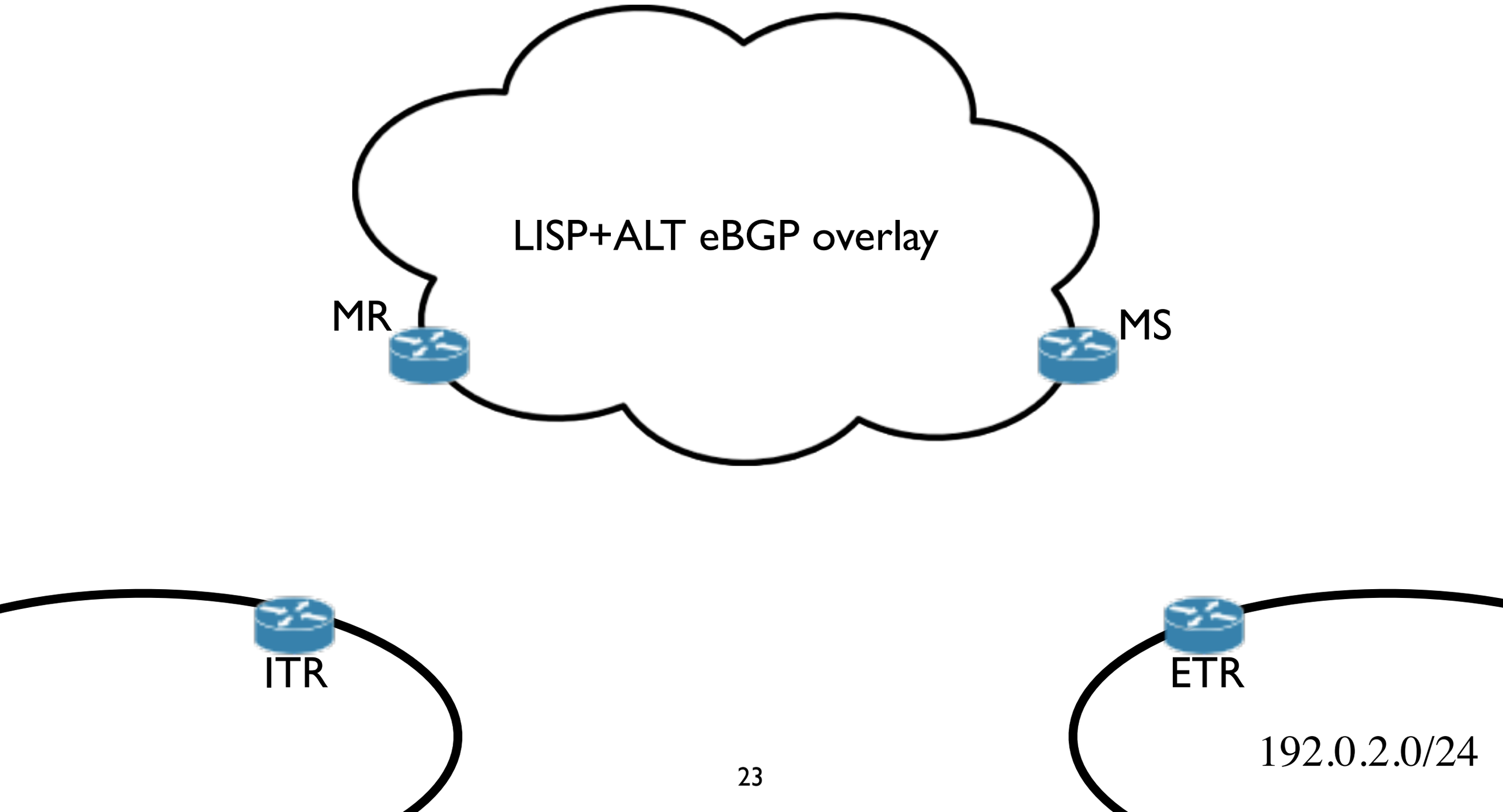
Infrastructure

- Mapping Database modularity
- Interworking of LISP and non LISP networks

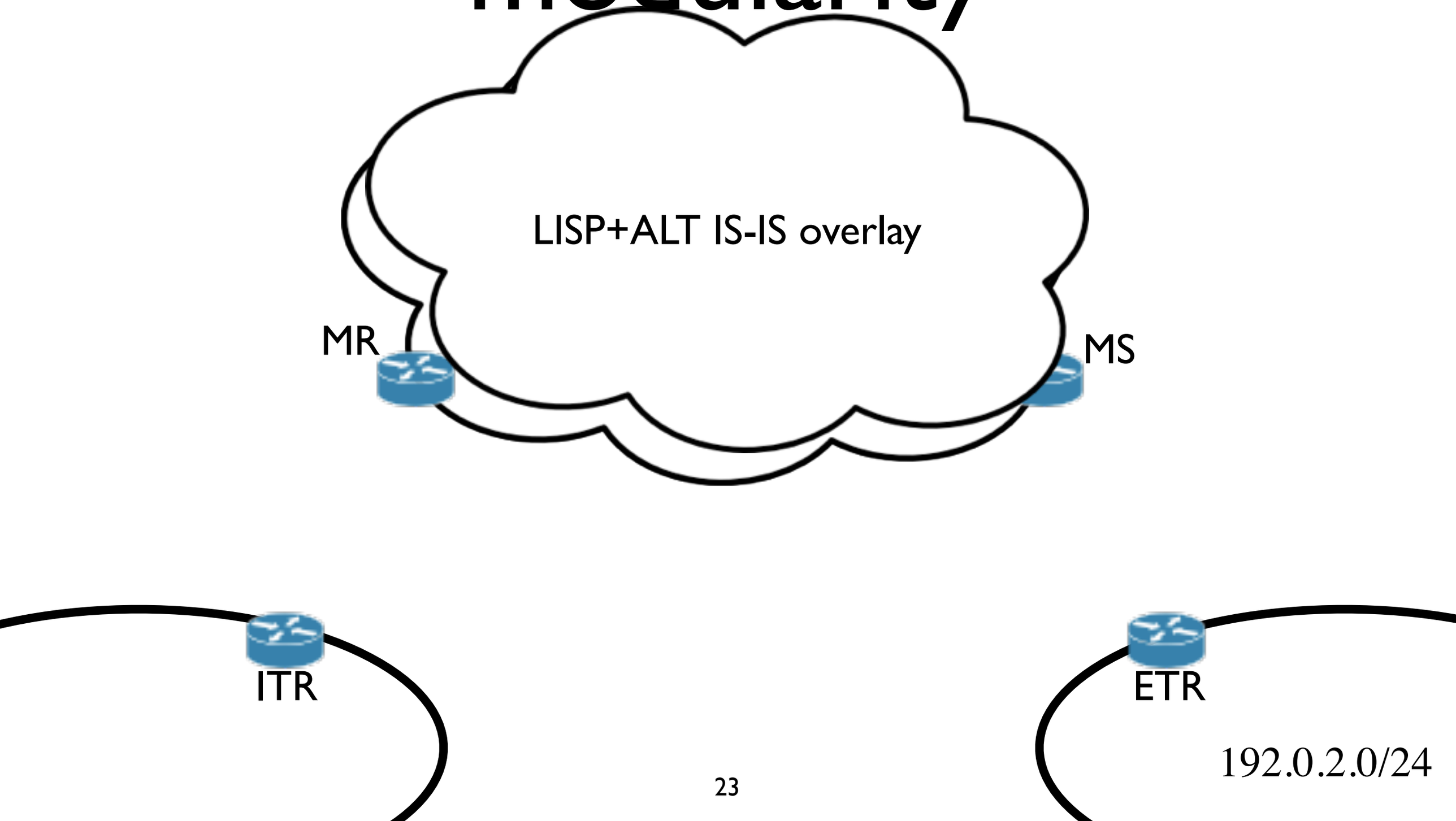
Mapping Database modularity



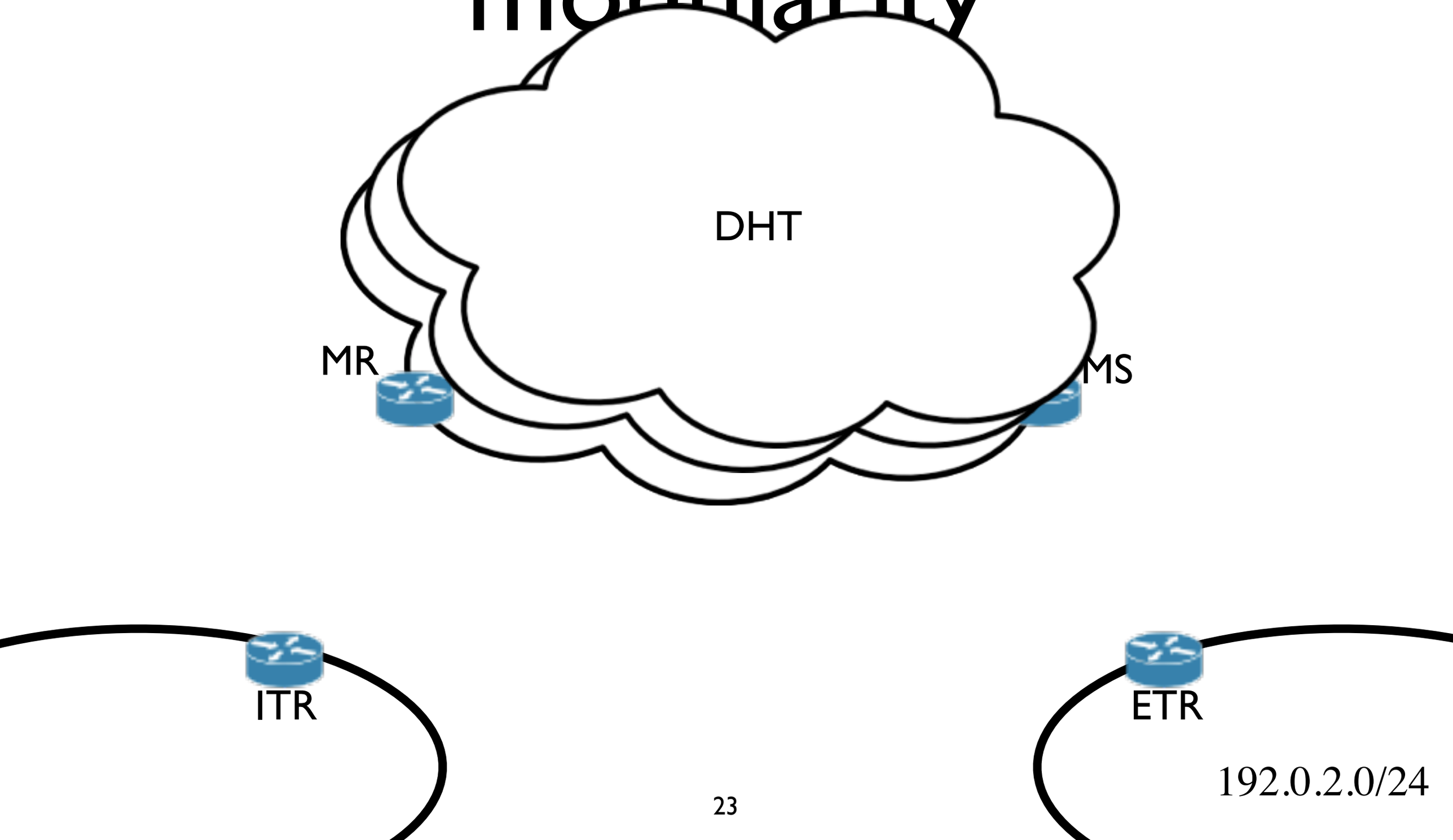
Mapping Database modularity



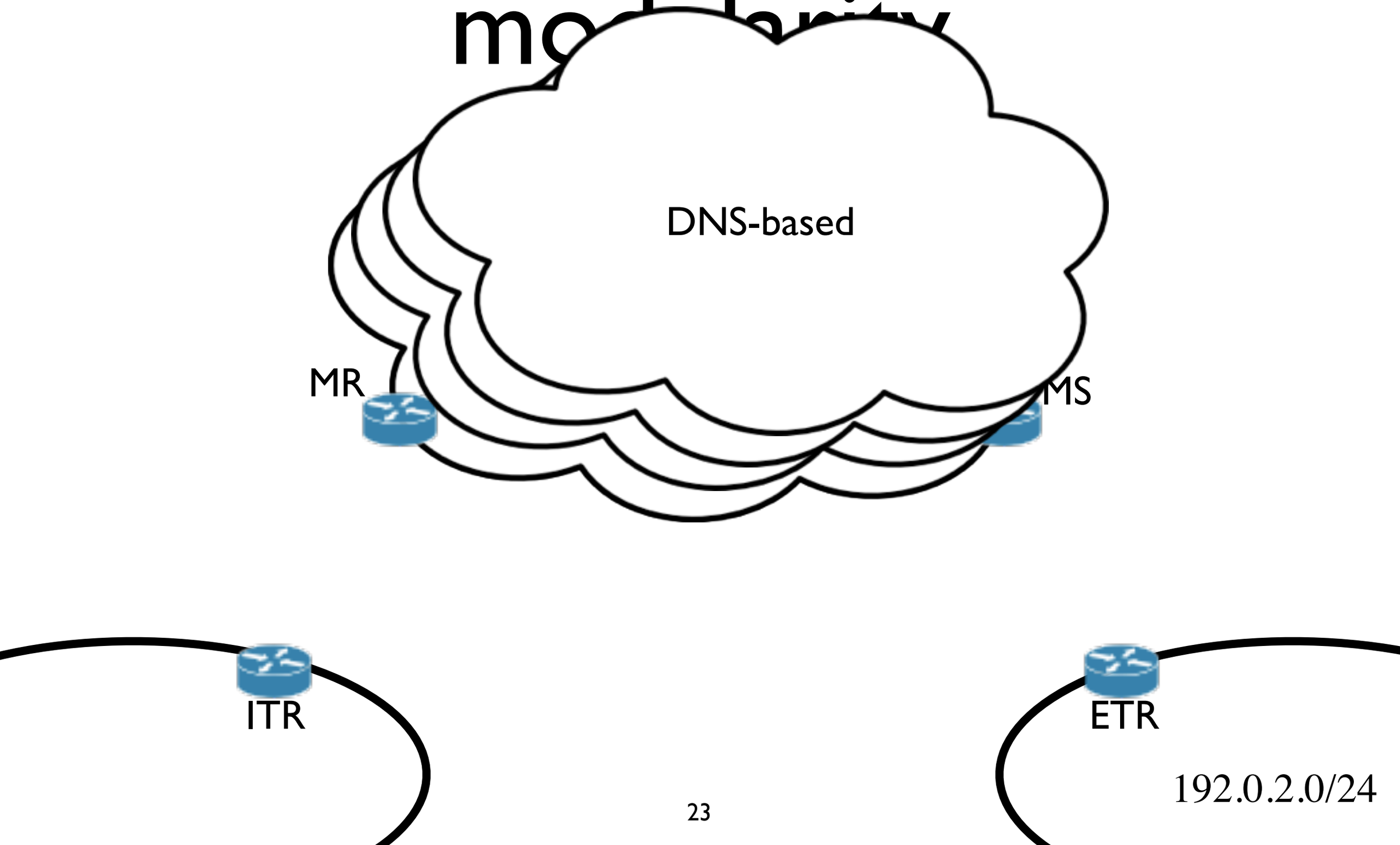
Mapping Database modularity



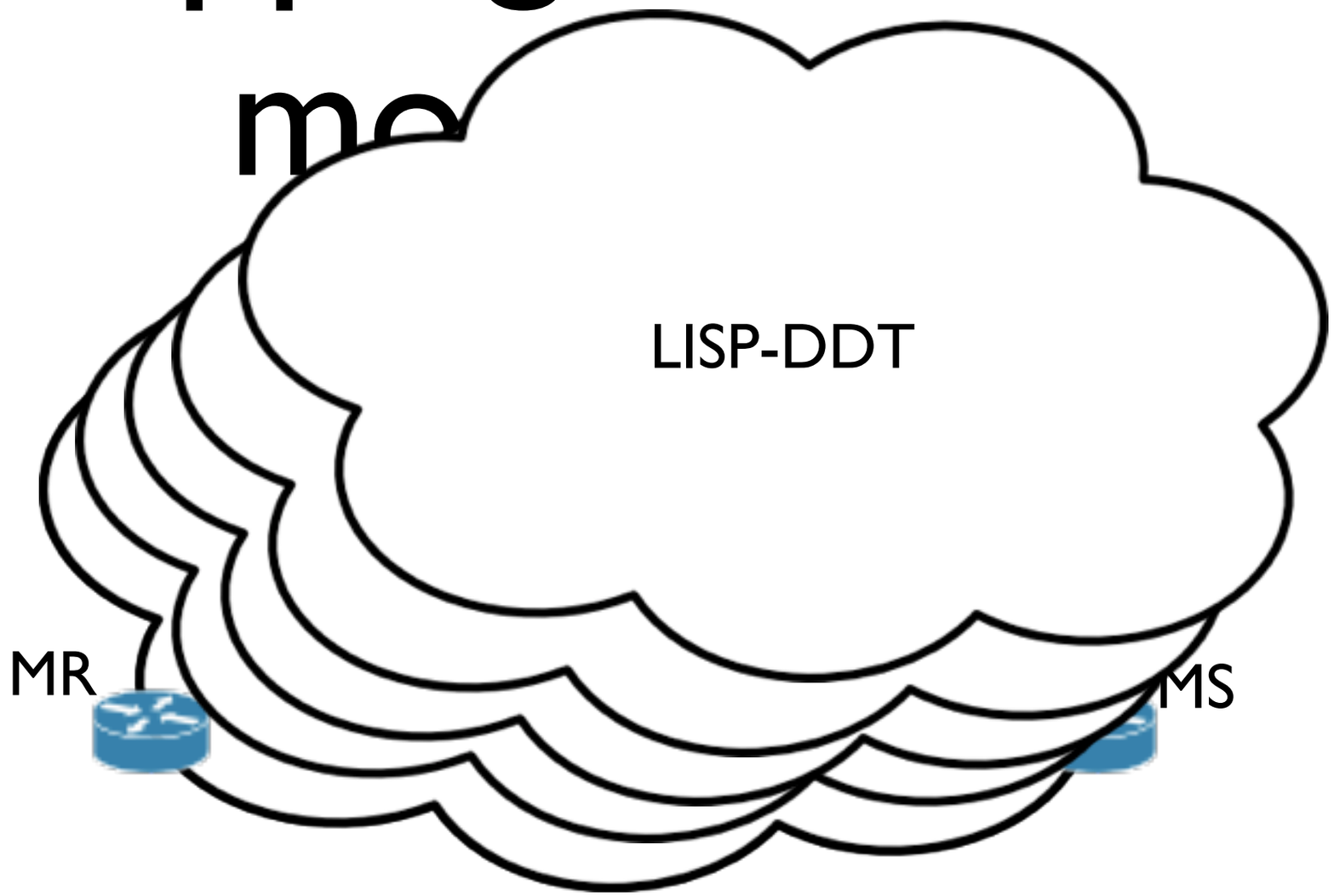
Mapping Database modularity

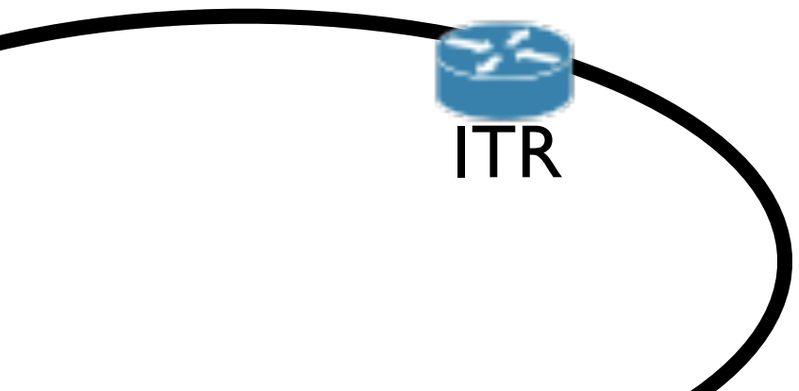


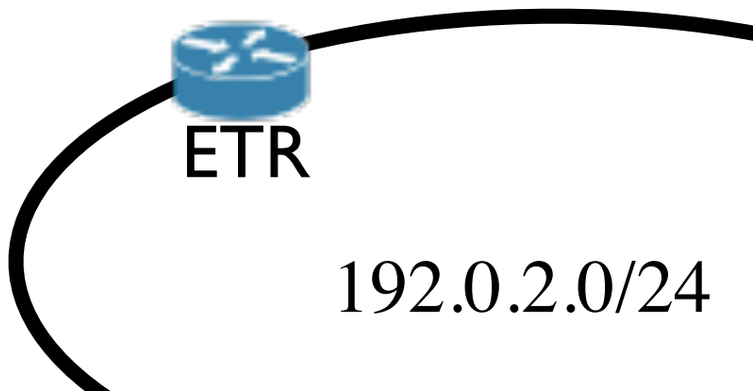
Mapping Database modularity



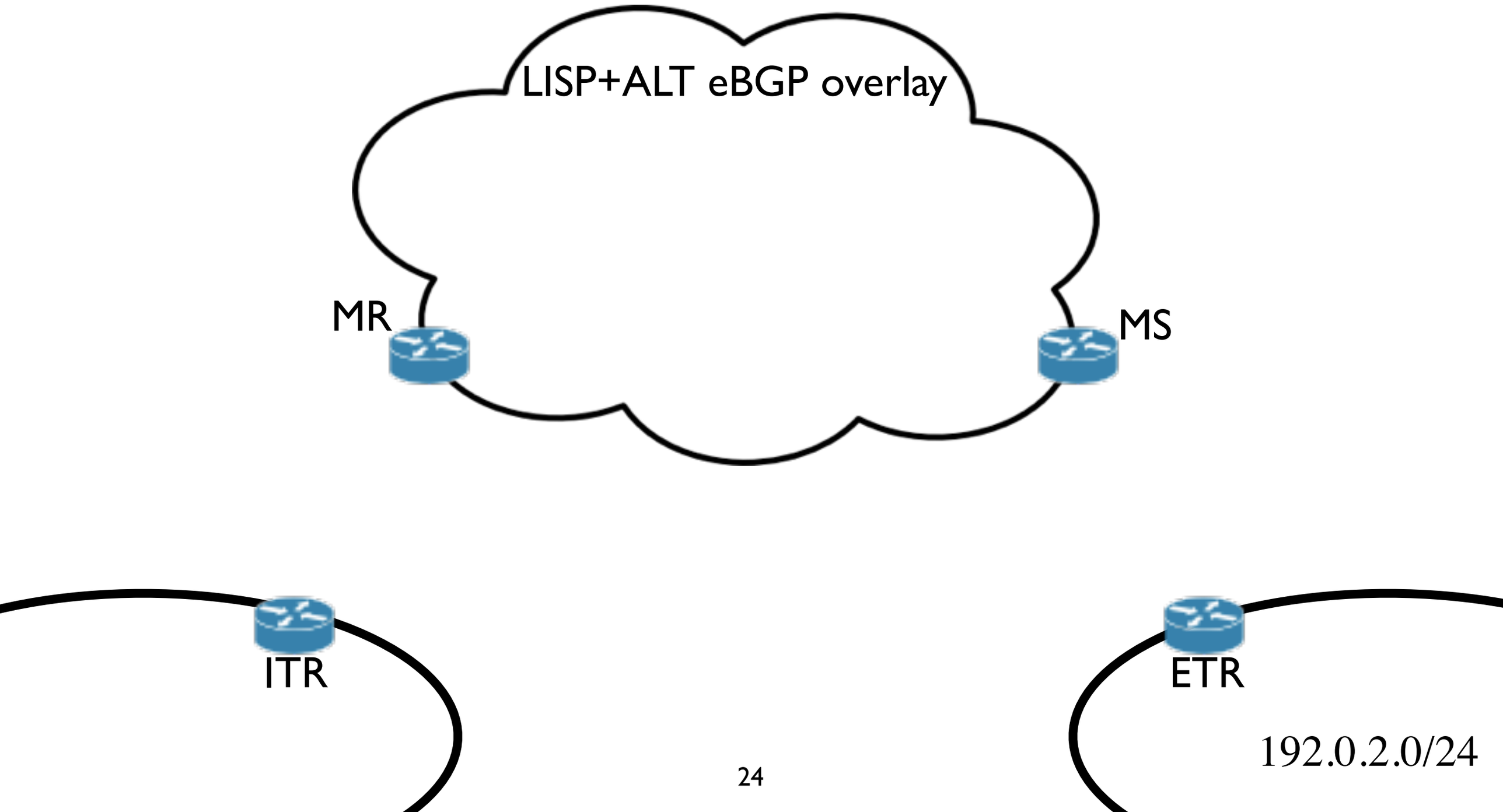
Mapping Database me



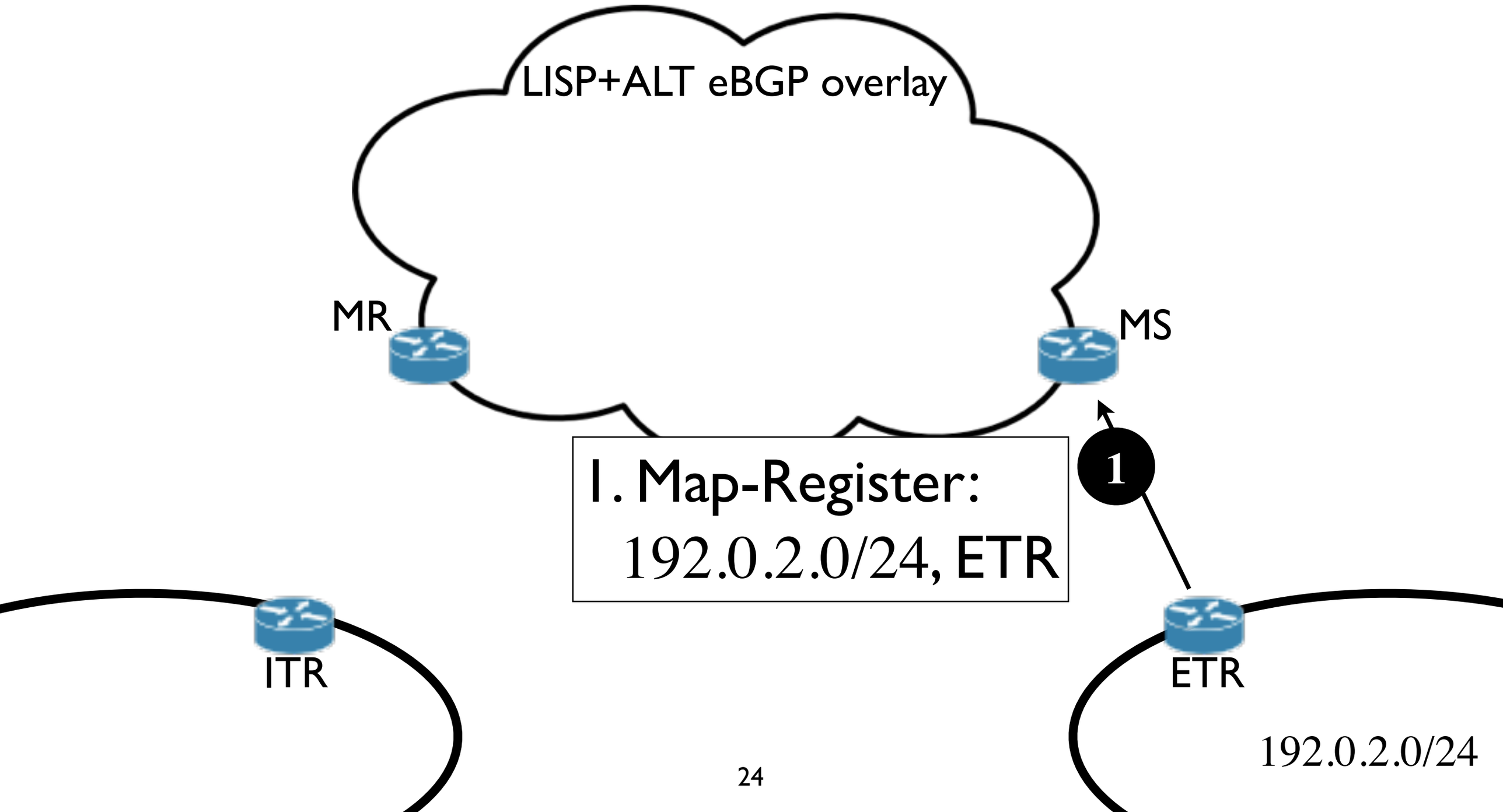




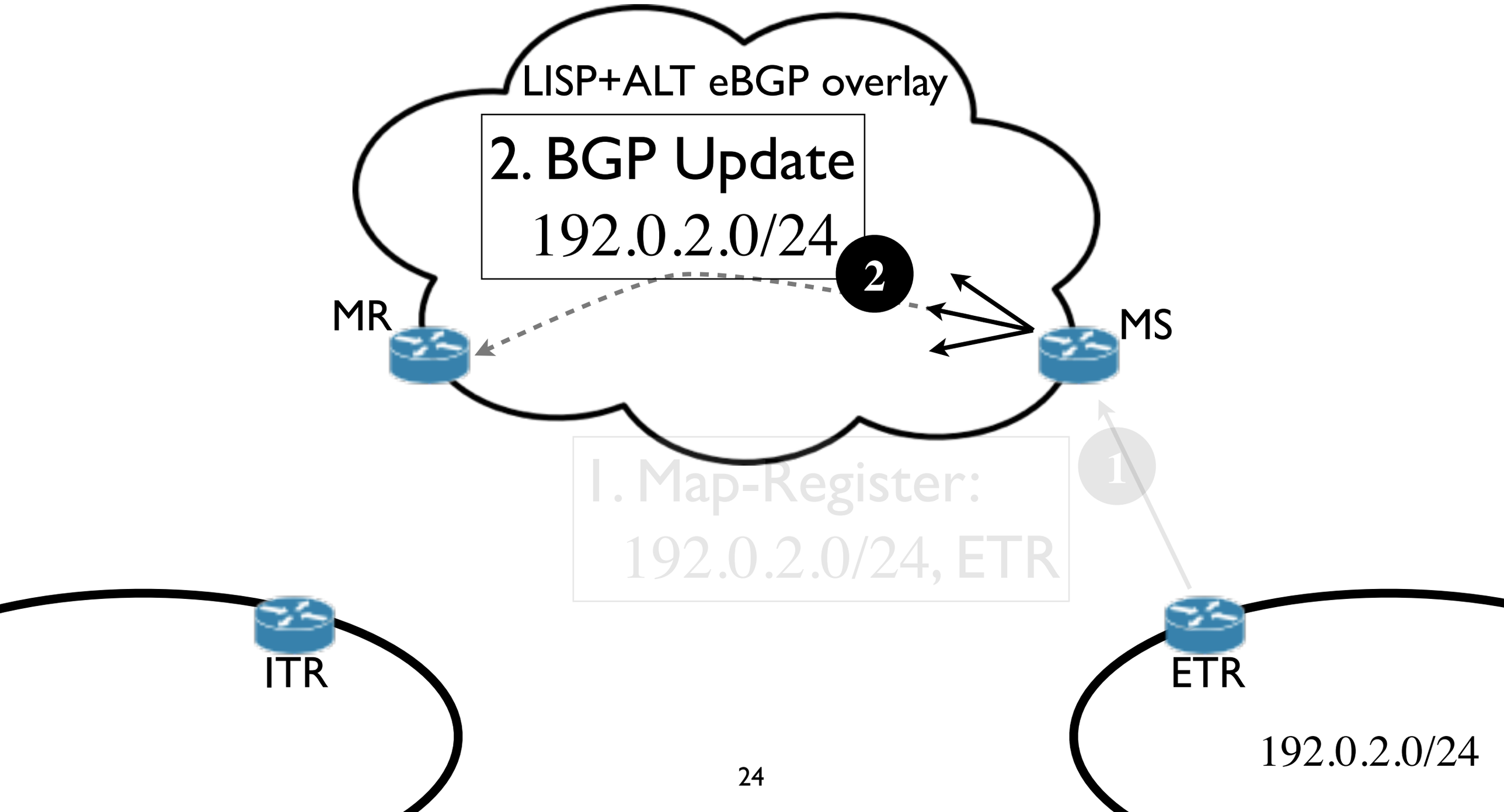
Mapping Registration



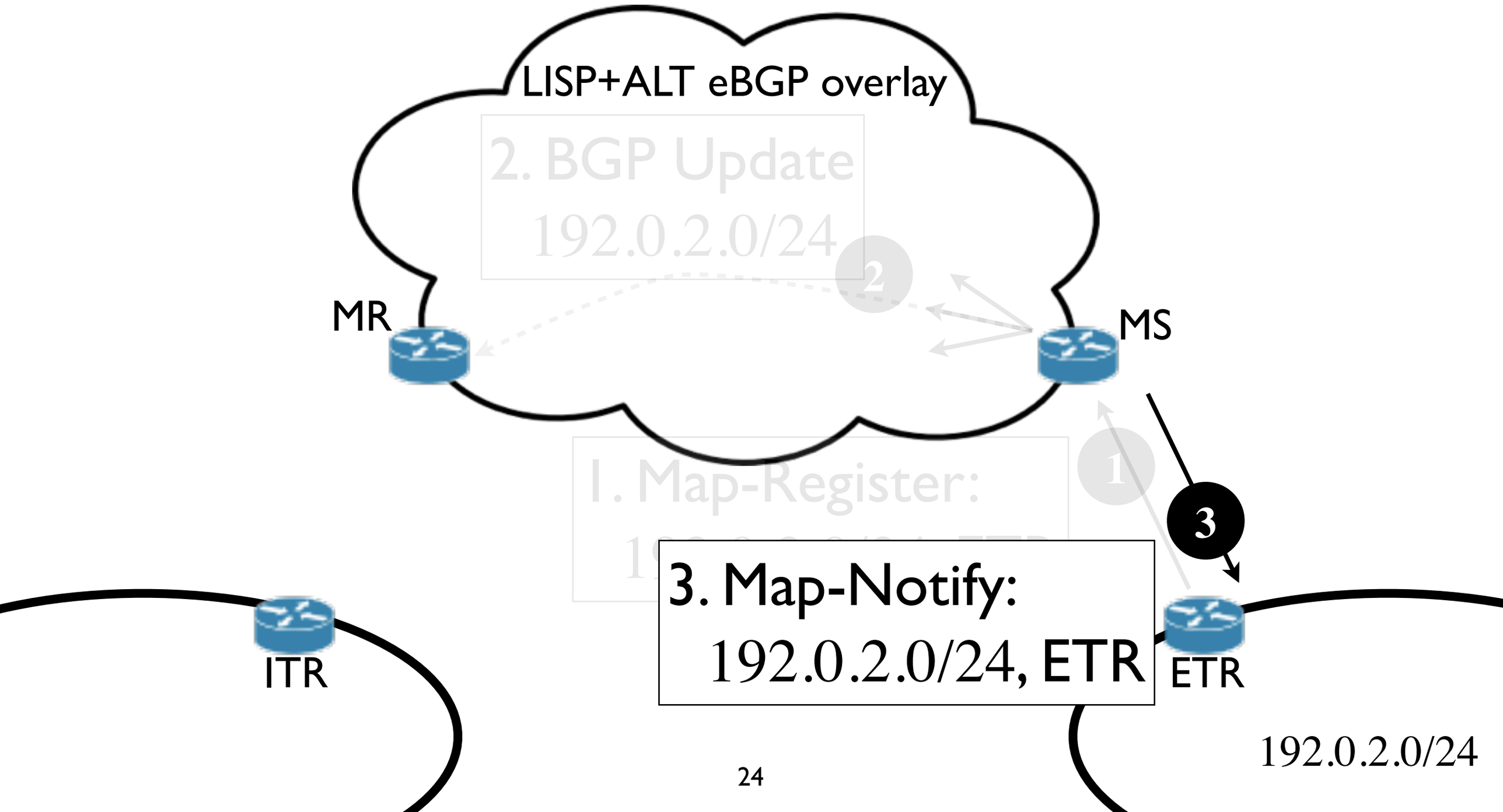
Mapping Registration



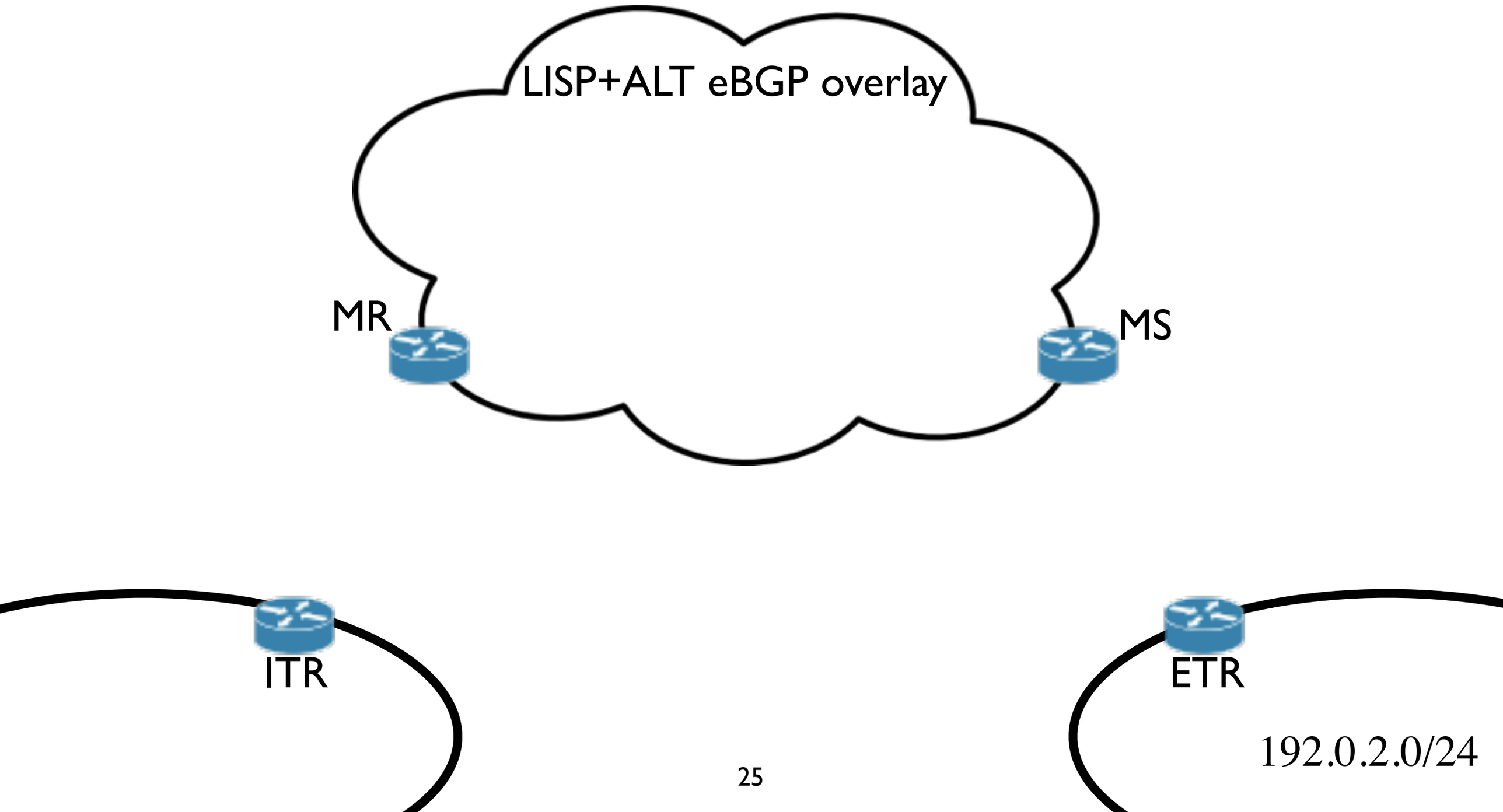
Mapping Registration



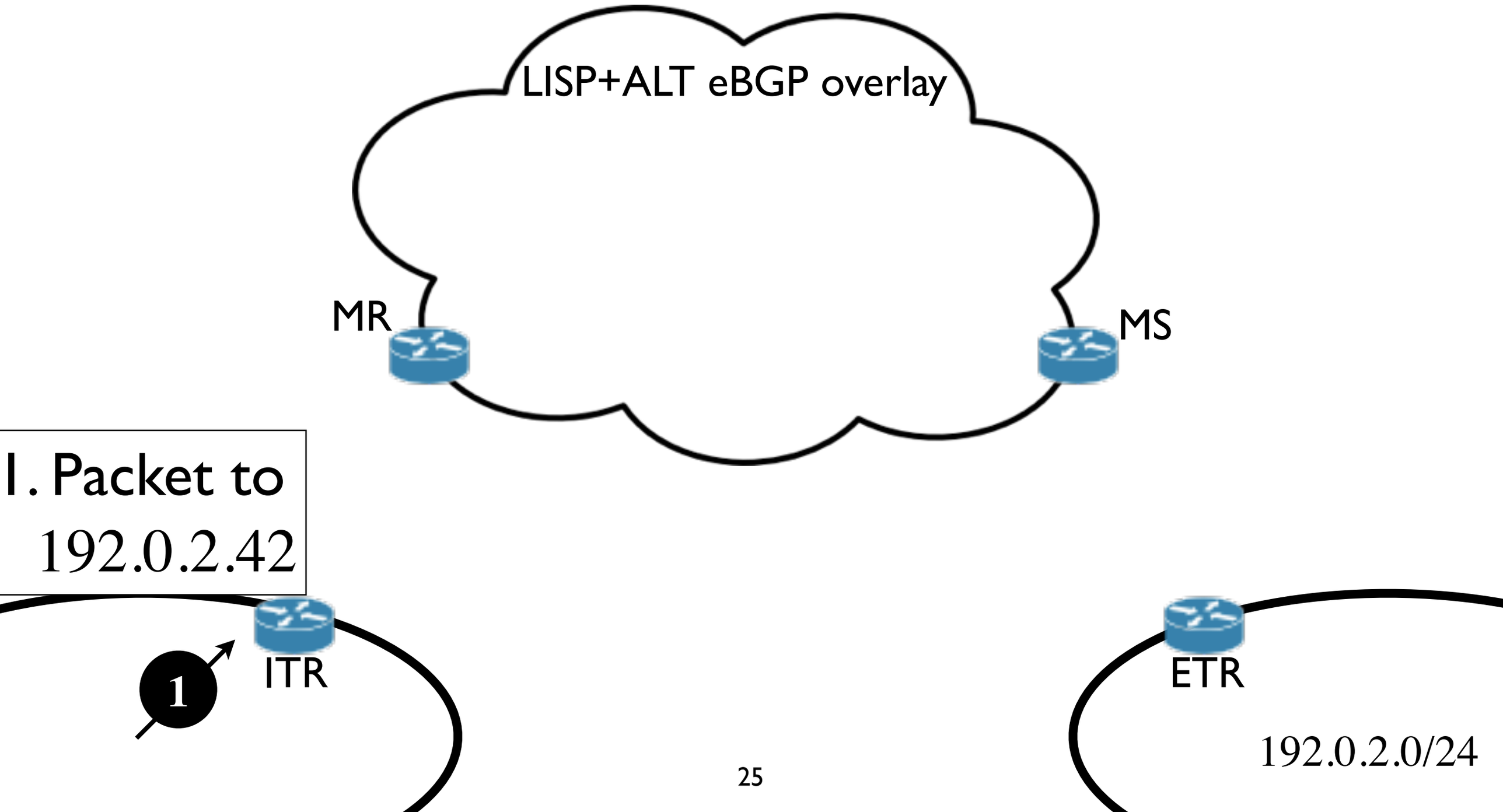
Mapping Registration



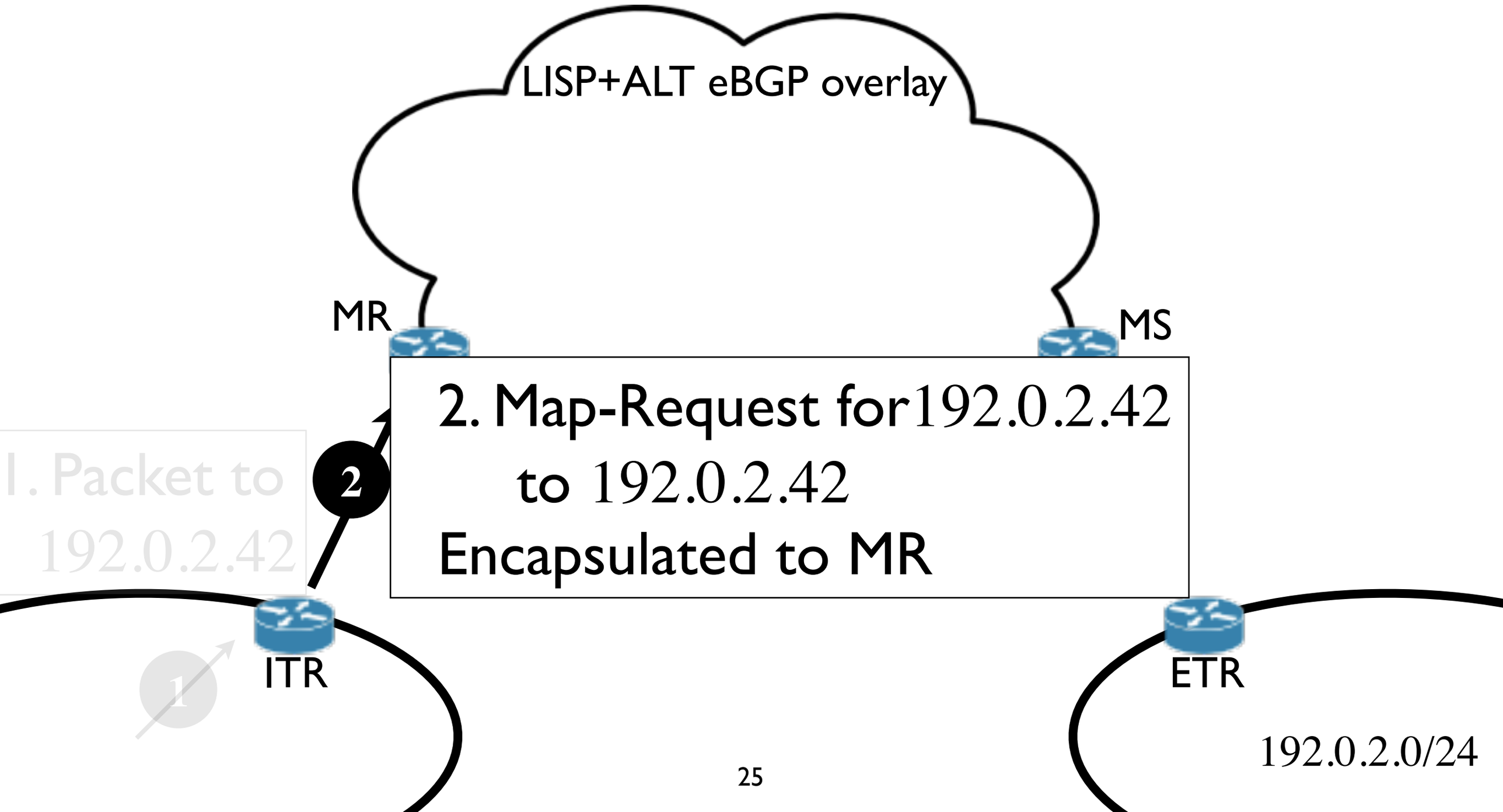
Mapping retrieval



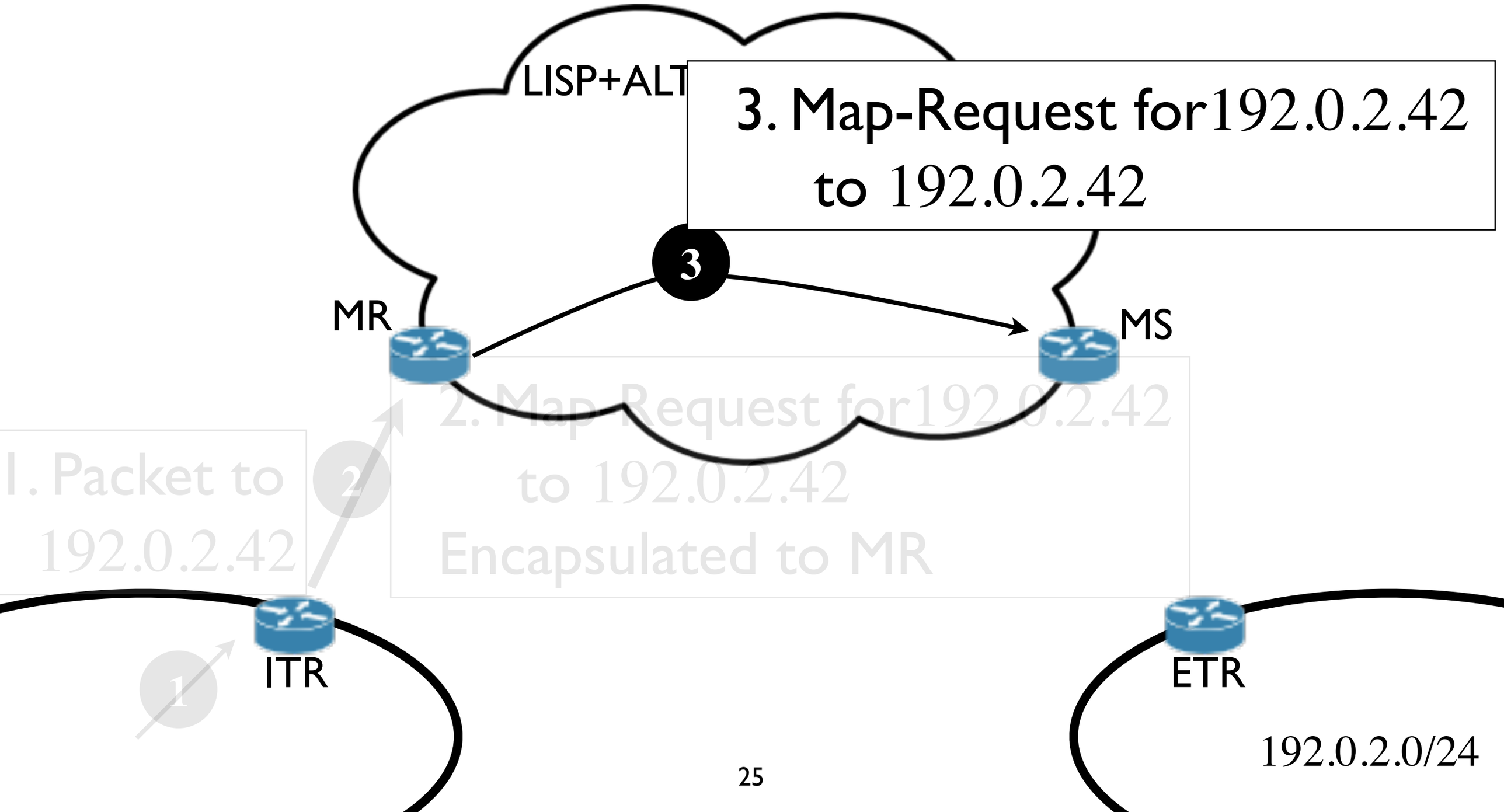
Mapping retrieval



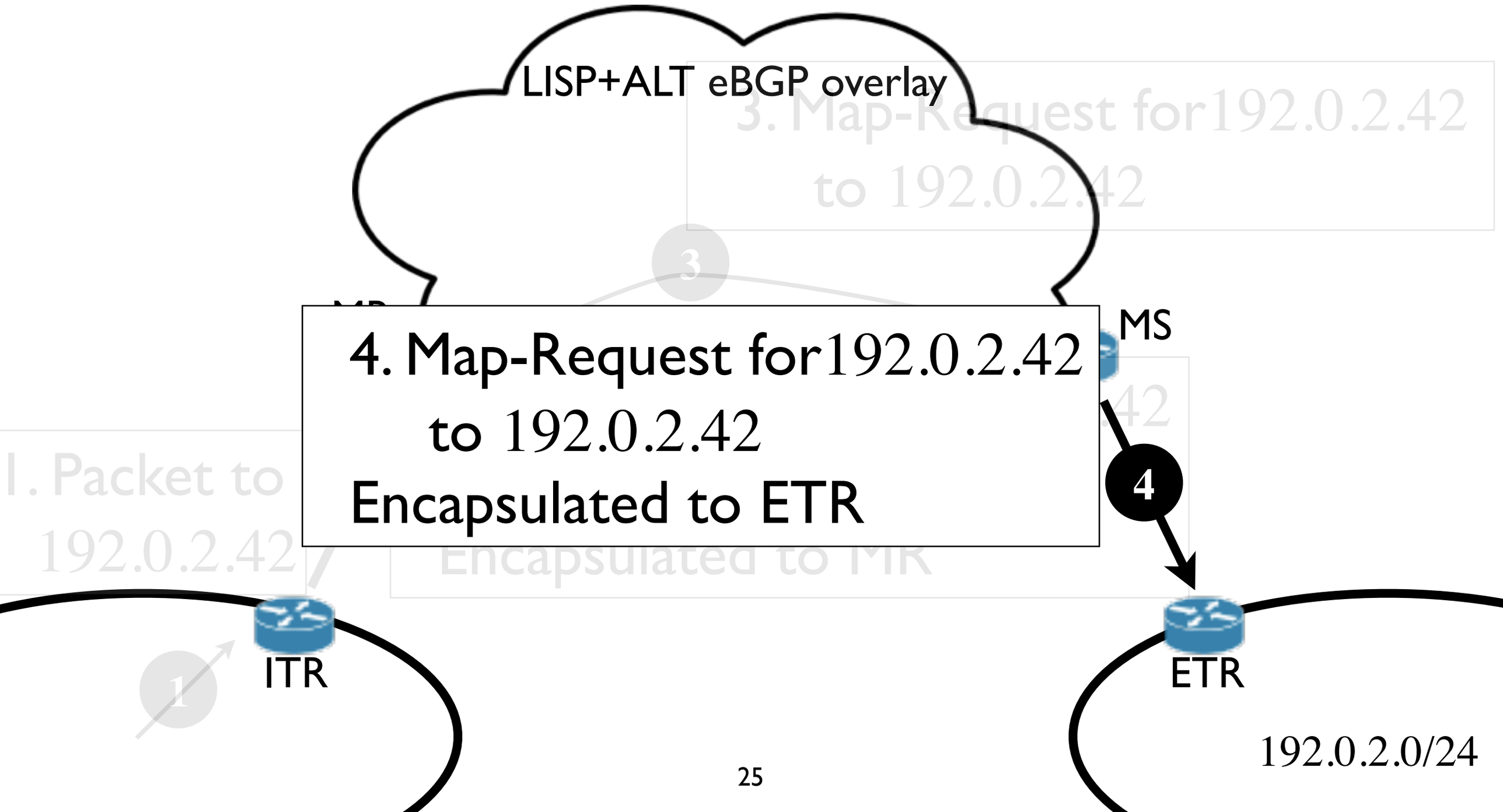
Mapping retrieval



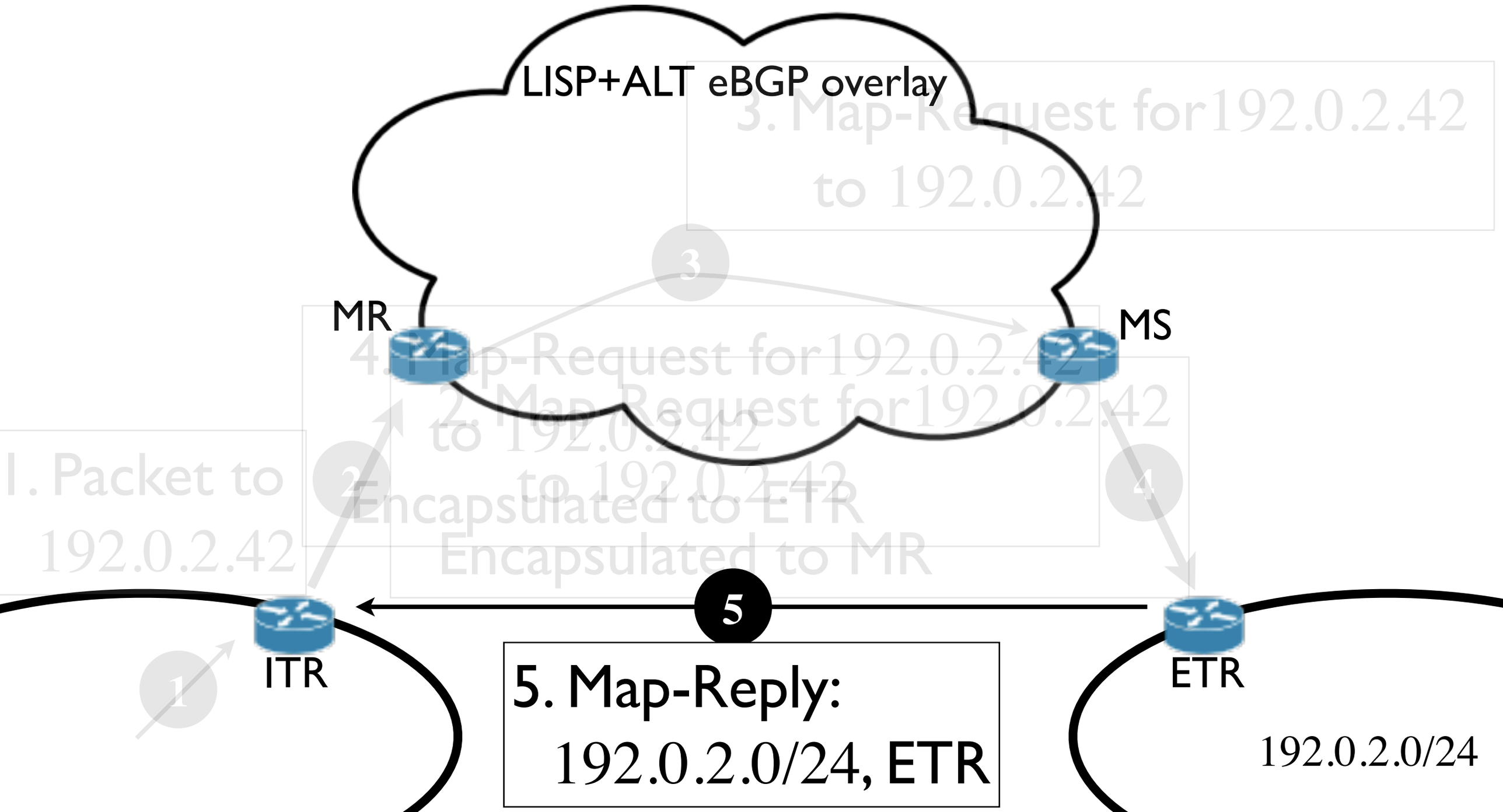
Mapping retrieval



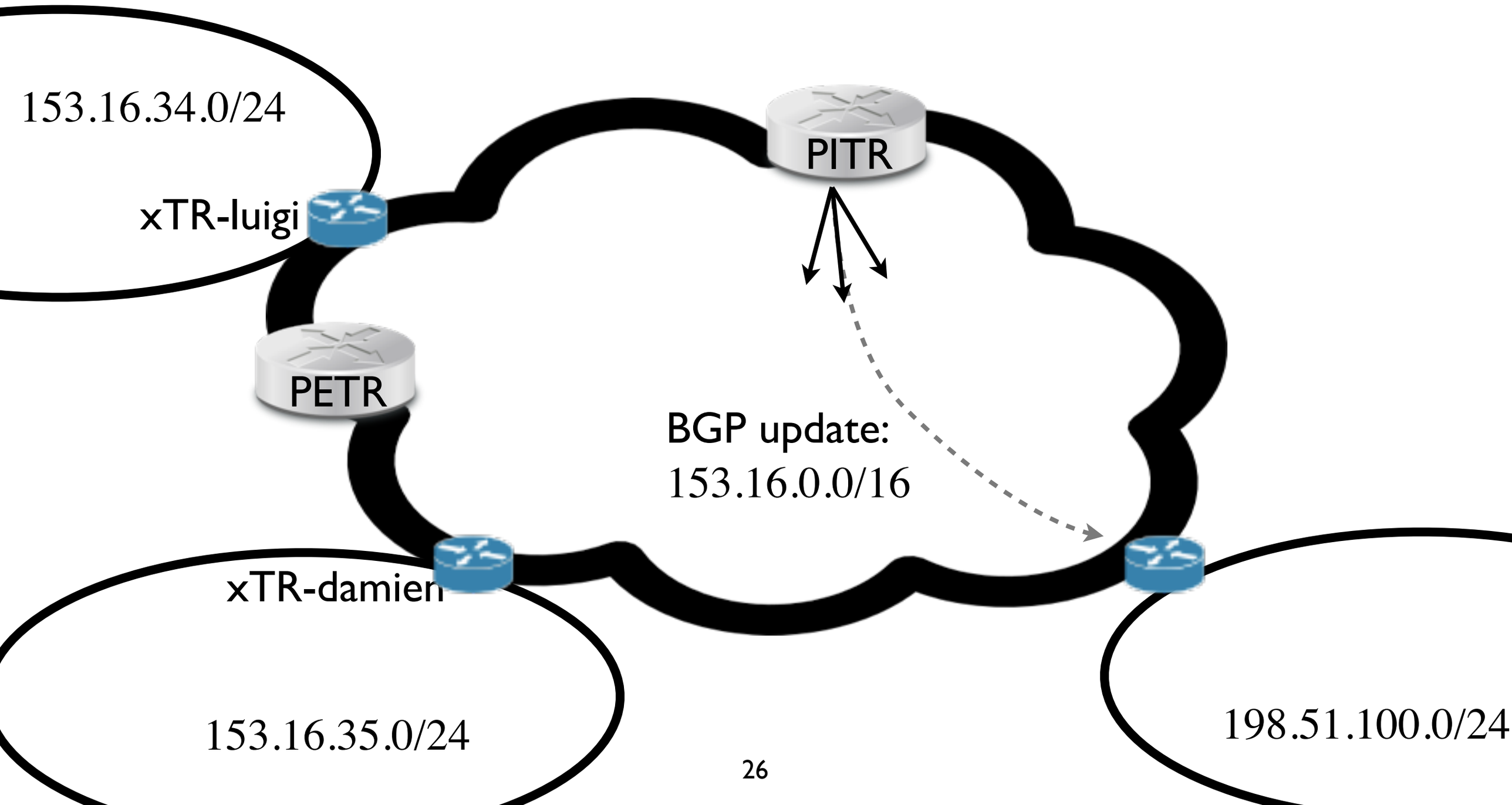
Mapping retrieval



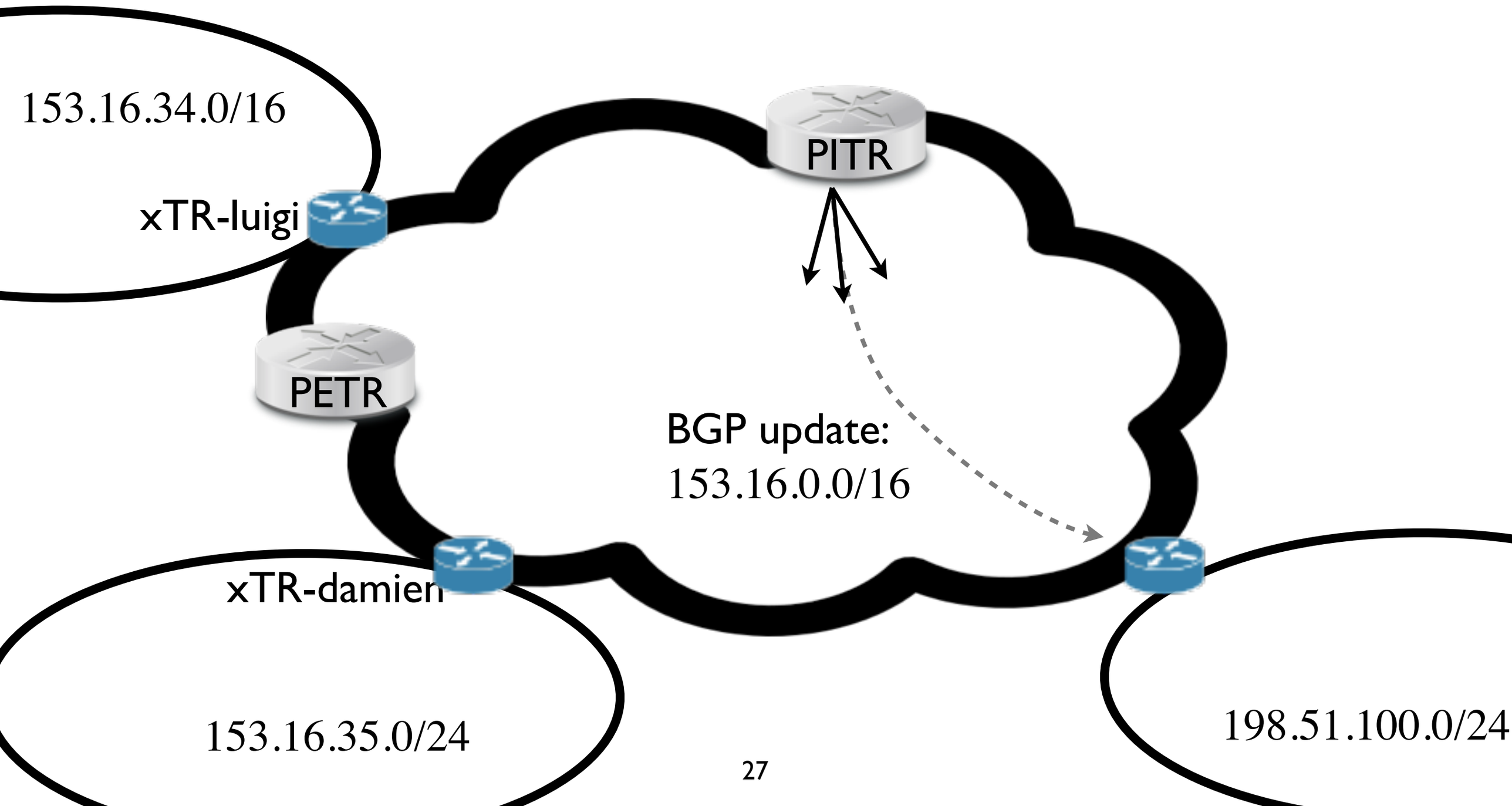
Mapping retrieval



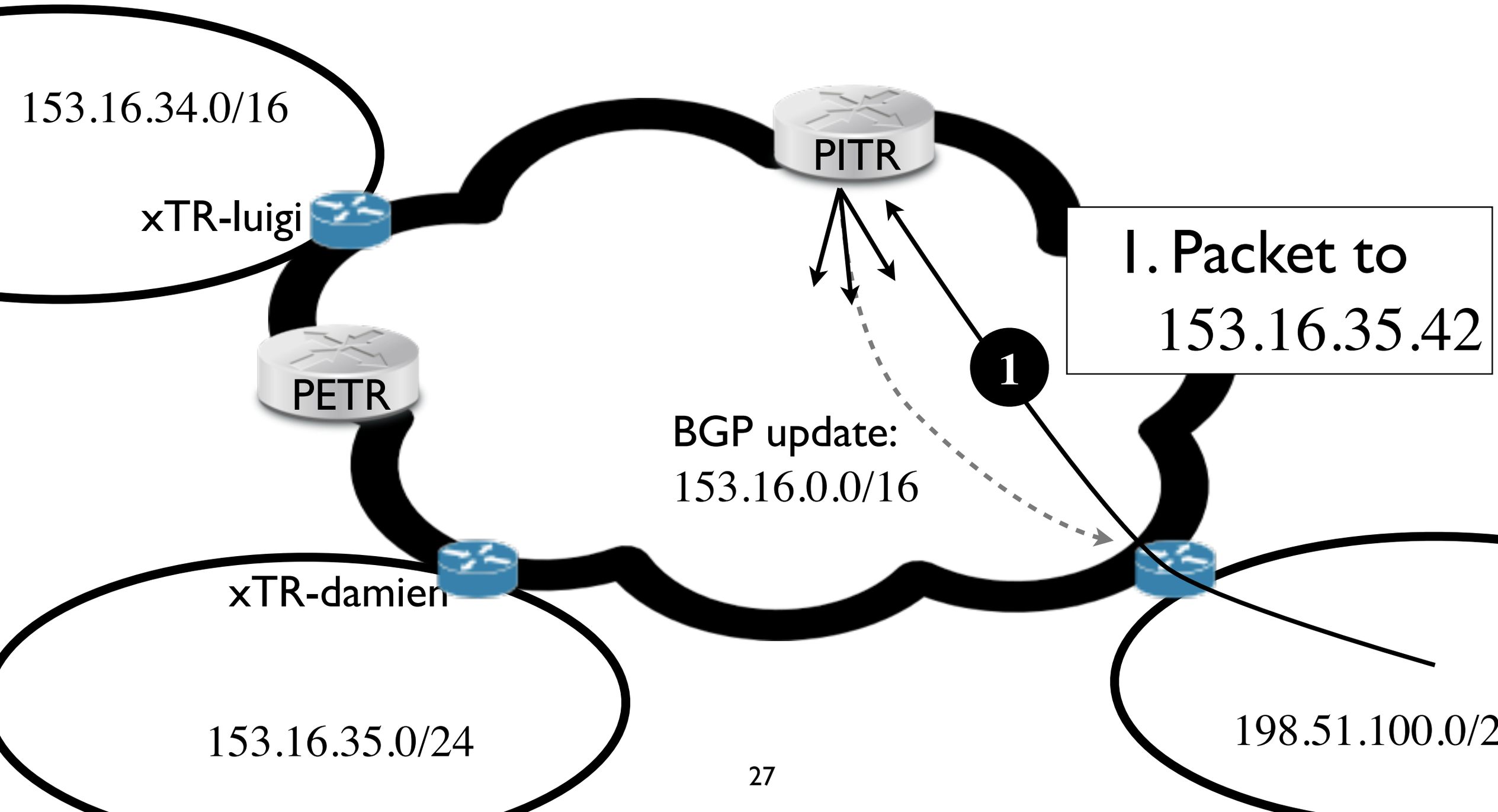
Interworking LISP



Non-LISP to LISP

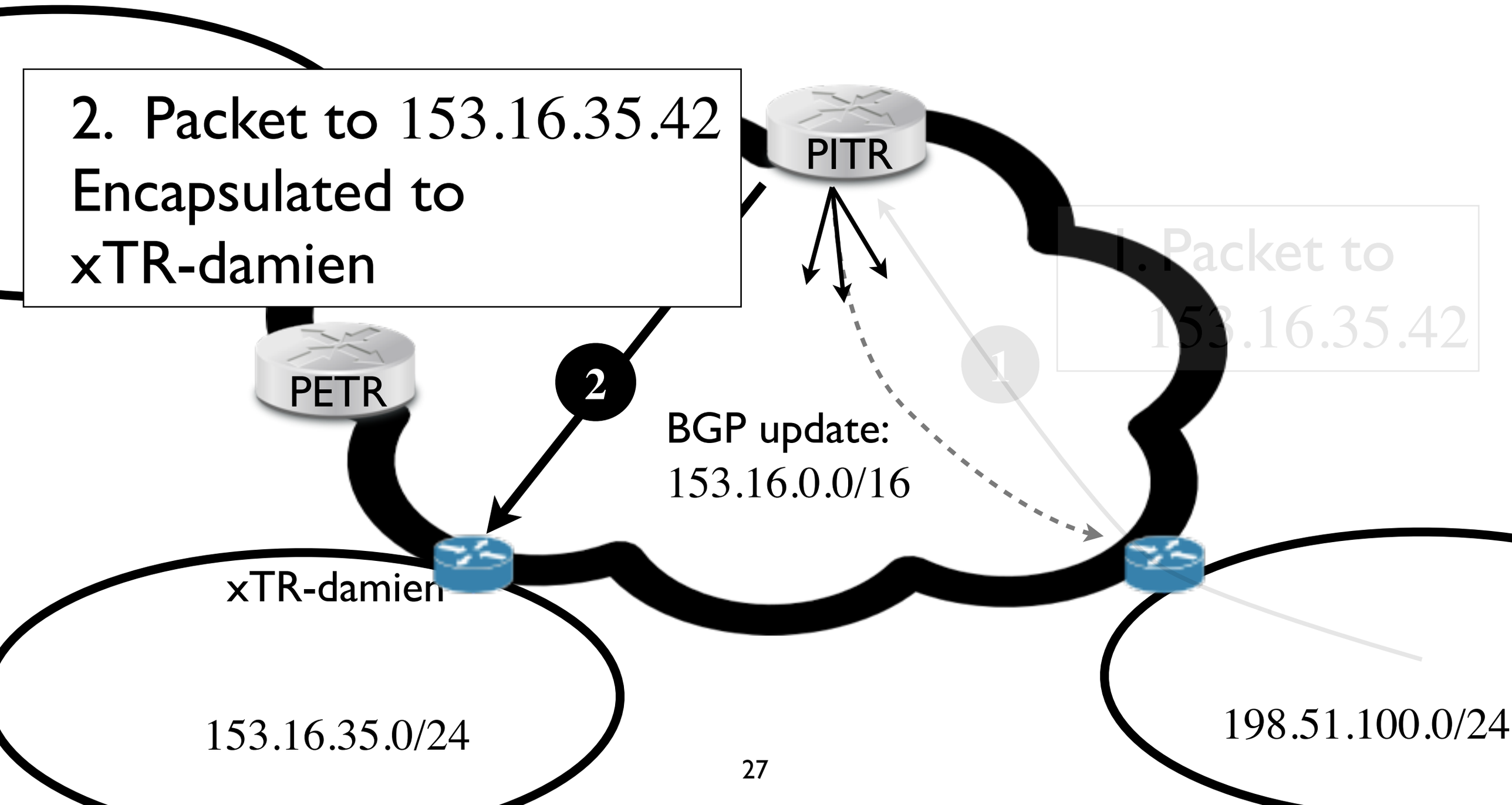


Non-LISP to LISP

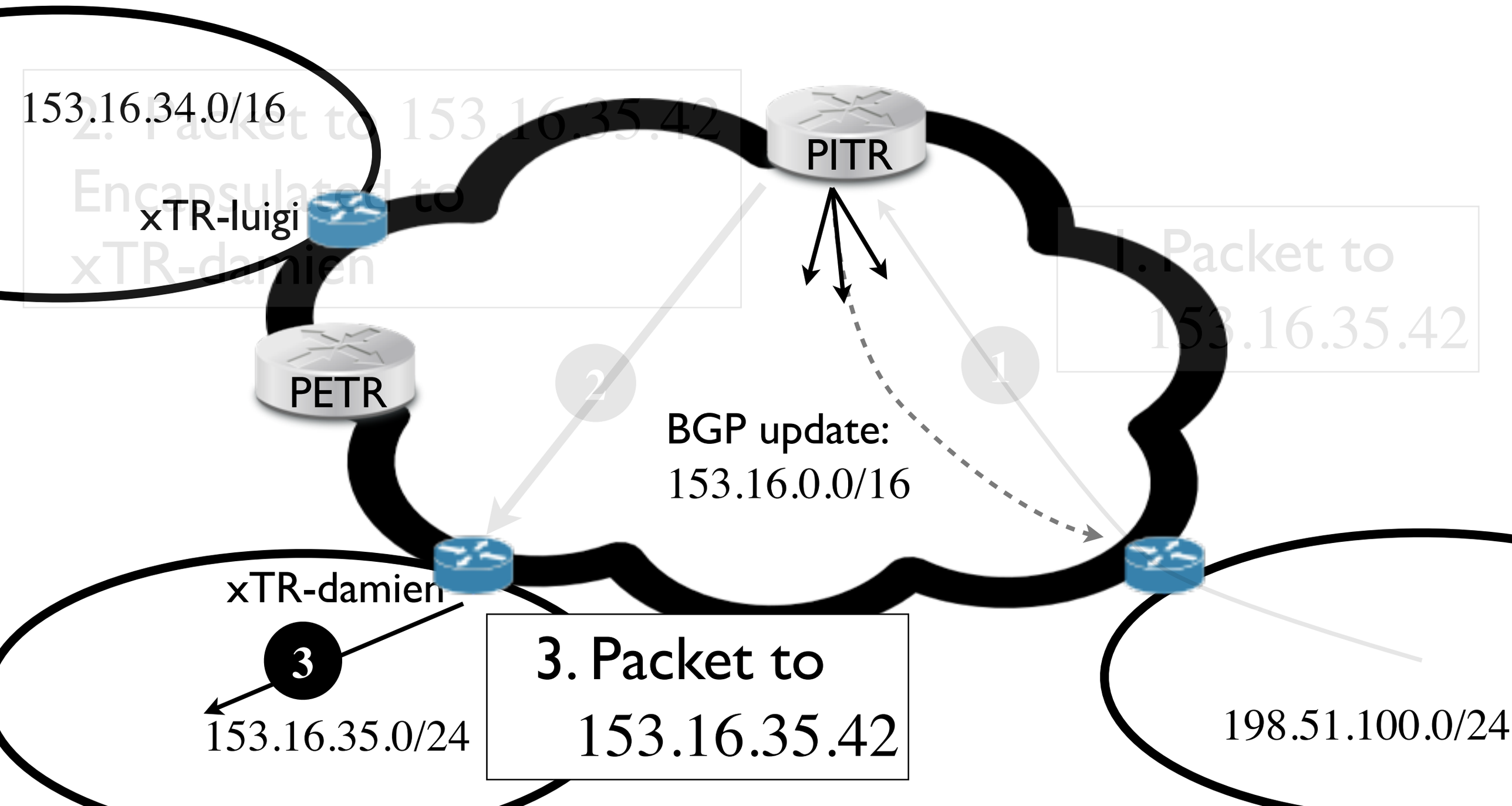


Non-LISP to LISP

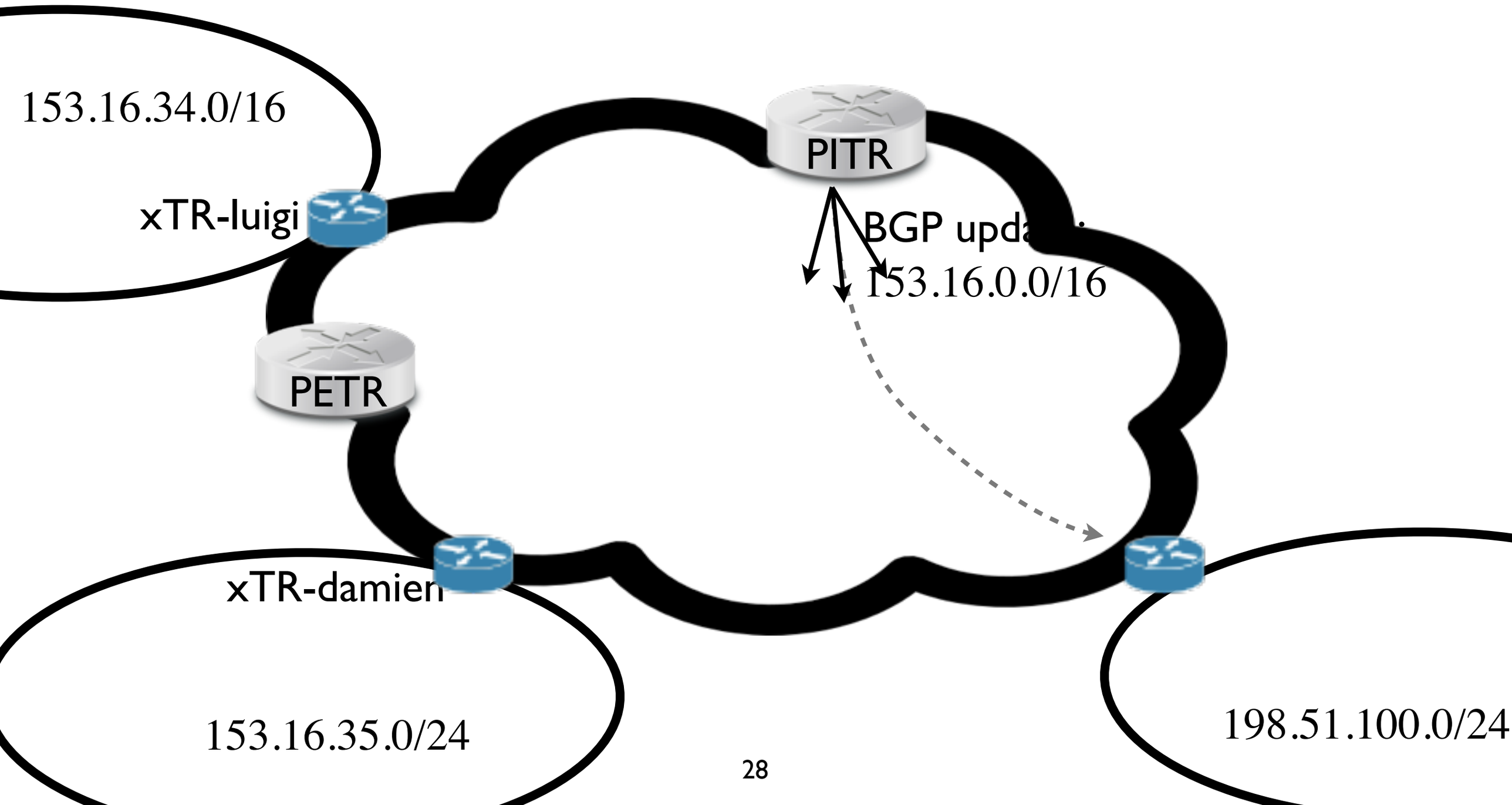
2. Packet to 153.16.35.42 Encapsulated to xTR-damien



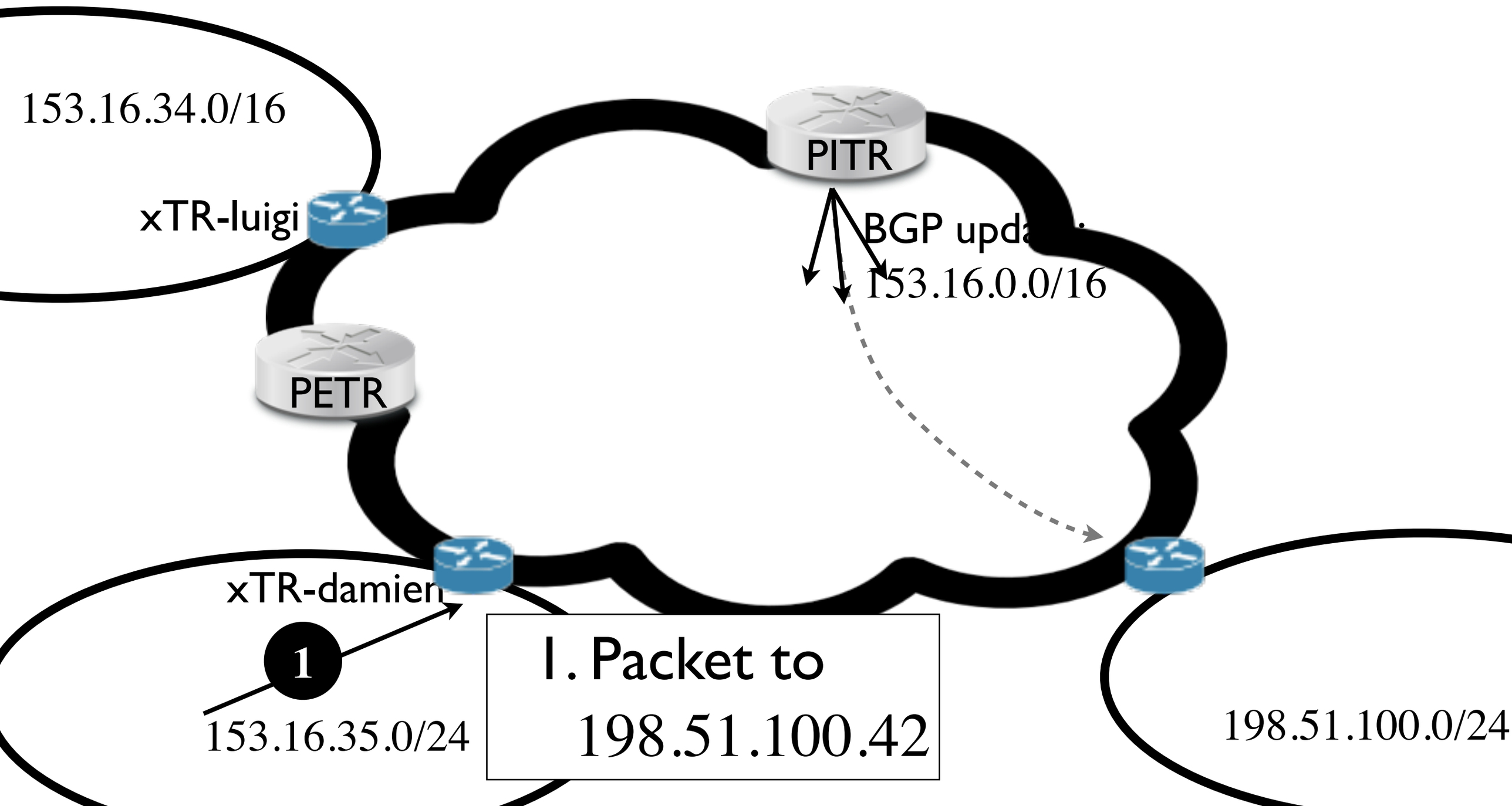
Non-LISP to LISP



LISP to non-LISP

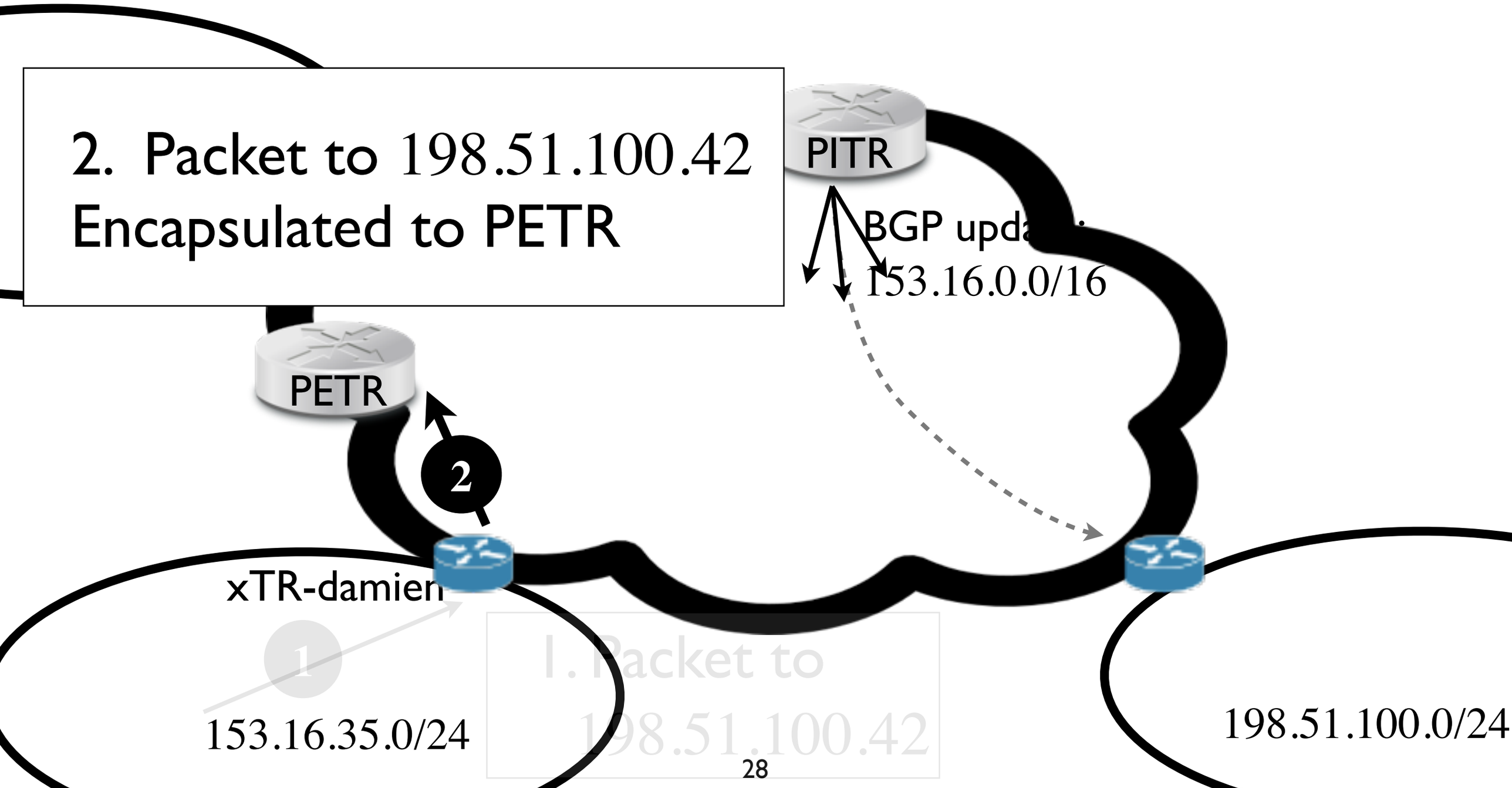


LISP to non-LISP

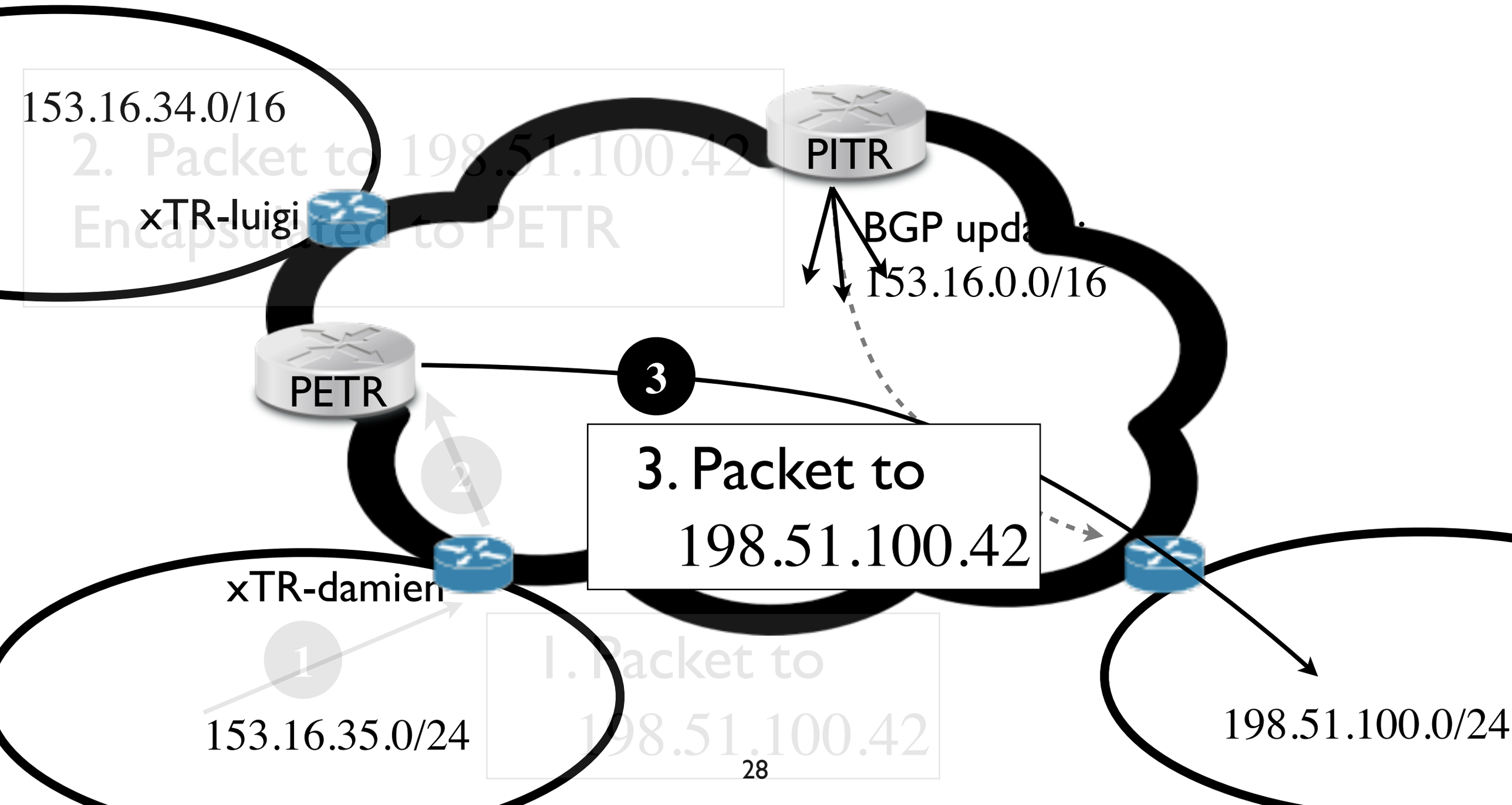


LISP to non-LISP

2. Packet to 198.51.100.42 Encapsulated to PETR

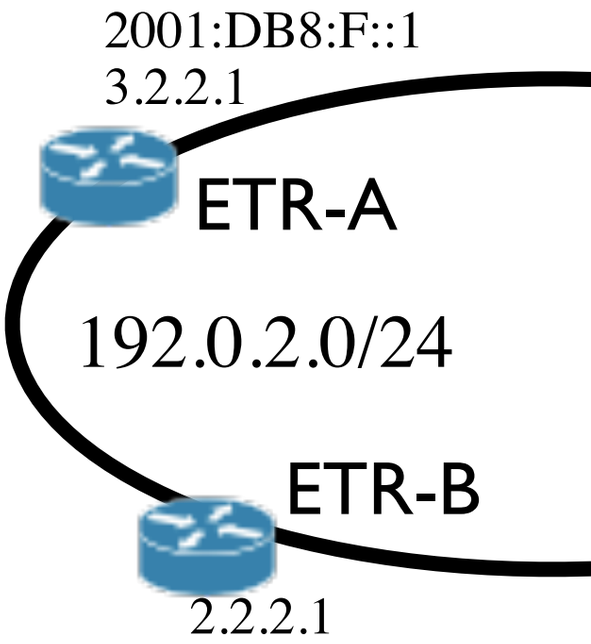


LISP to non-LISP



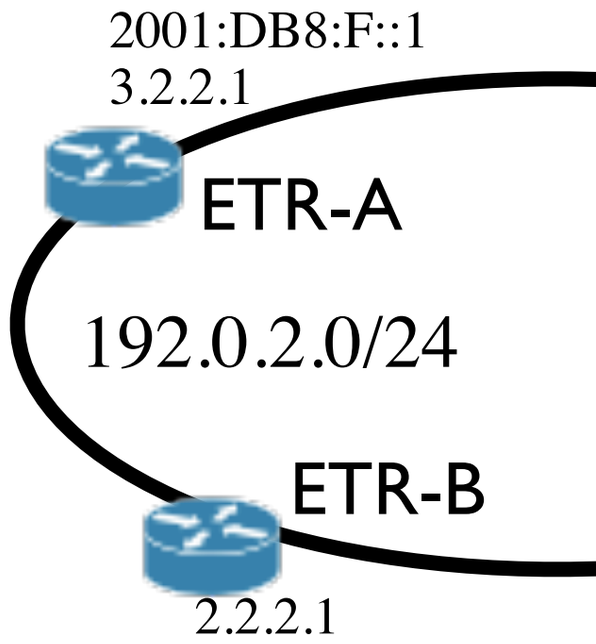
LISP Use Cases

Low OpEx site multihoming



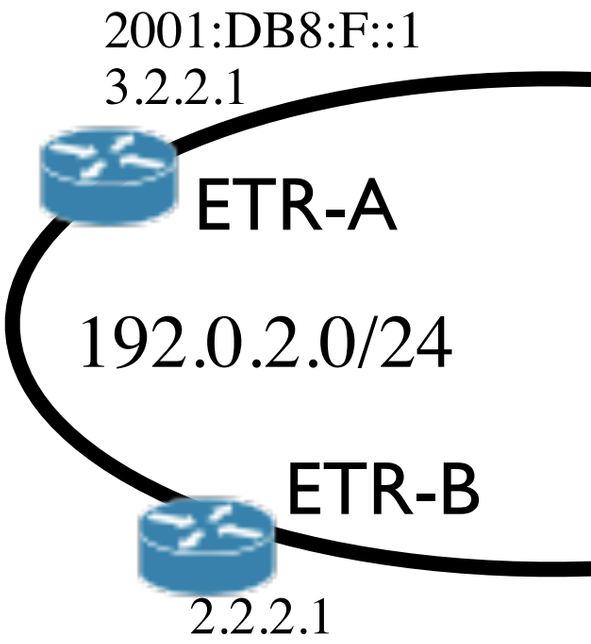
Low OpEx site multihoming

- Basic LISP feature



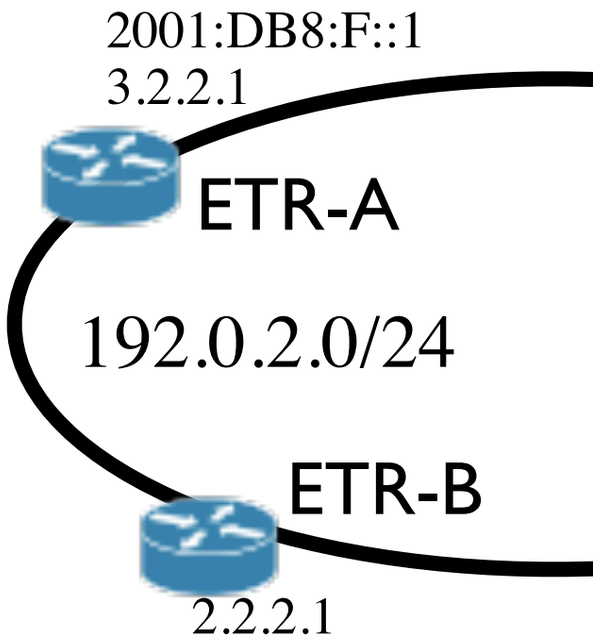
Low OpEx site multihoming

- 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:11, prio 100, weight: 100

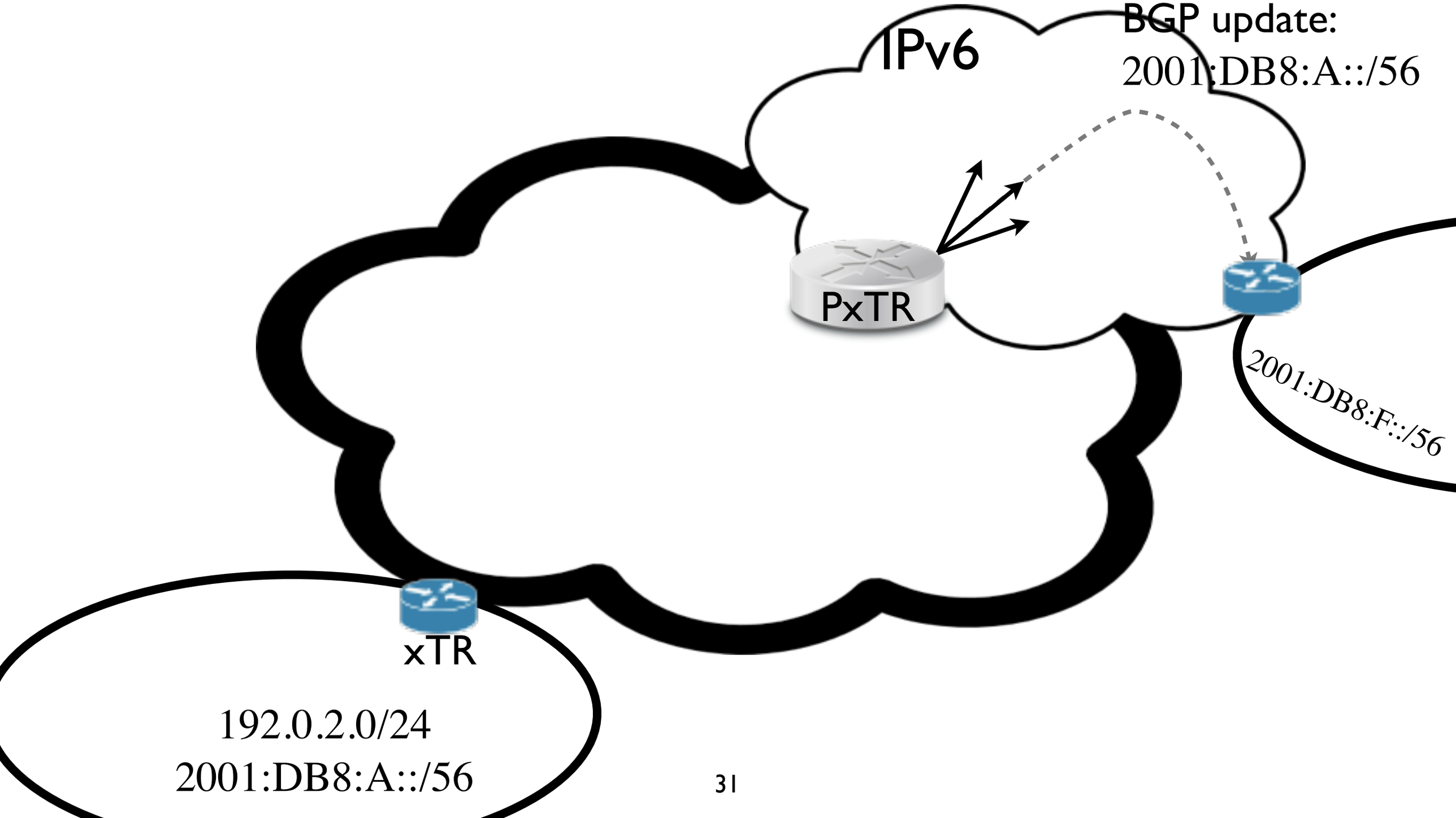


Low OpEx site multihoming

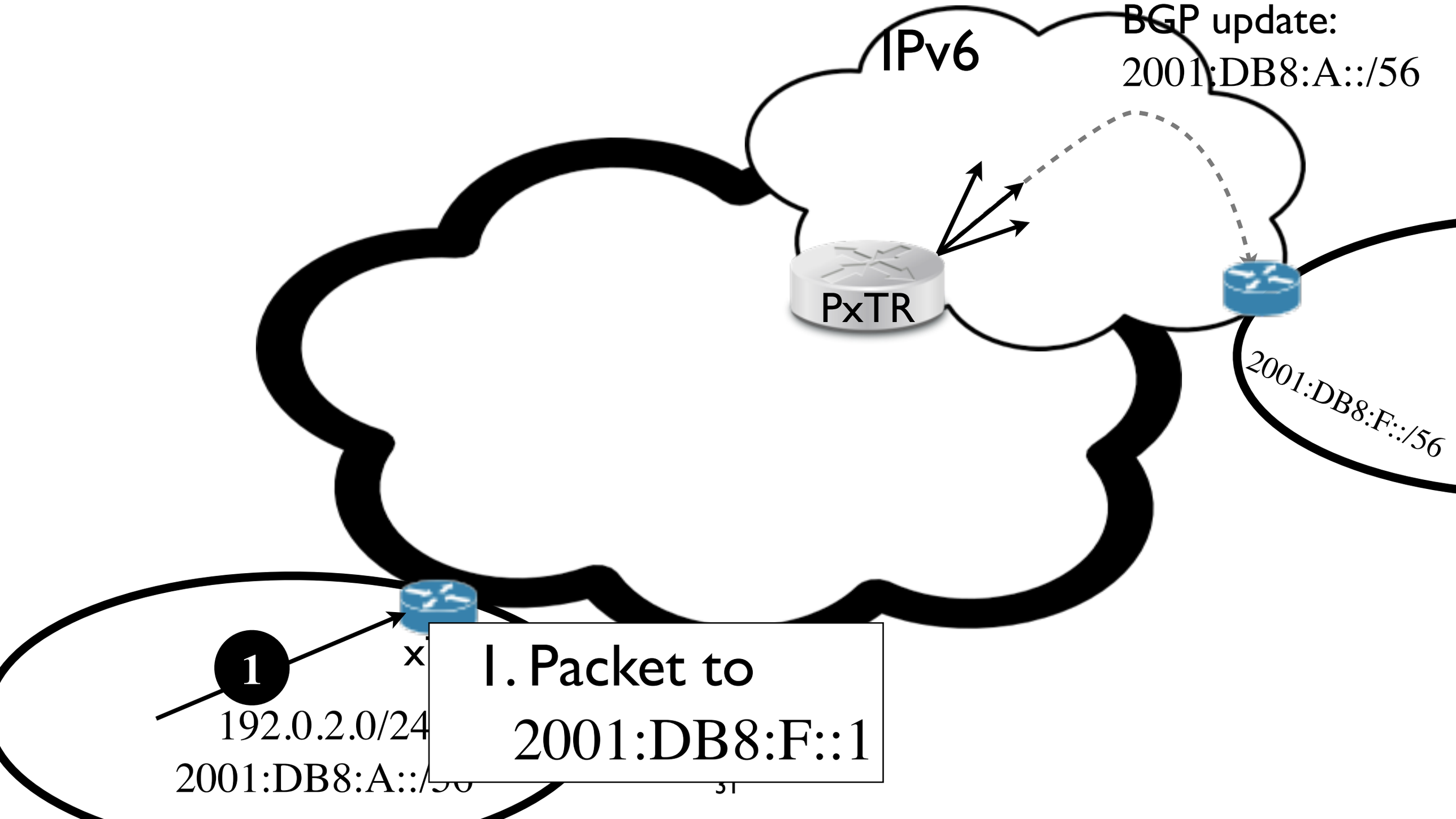
- Basic LISP feature
- ETR-A Primary, ETR-B Backup (IPv6 just in case...)
  - 3.2.2.1, prio: 1, weight: 100
  - 2.2.2.1, prio: 10, weight: 100
  - 2001:DB8:F:11, prio 100, weight: 100
- ETR-A: 60%, ETR-B: 40% (IPv6 just in case...)
  - 3.2.2.1, prio: 1, weight: 60
  - 2.2.2.1, prio: 1, weight: 40
  - 2001:DB8:F:11, prio 99, weight: 100



IPv6/IPv4 coexistence

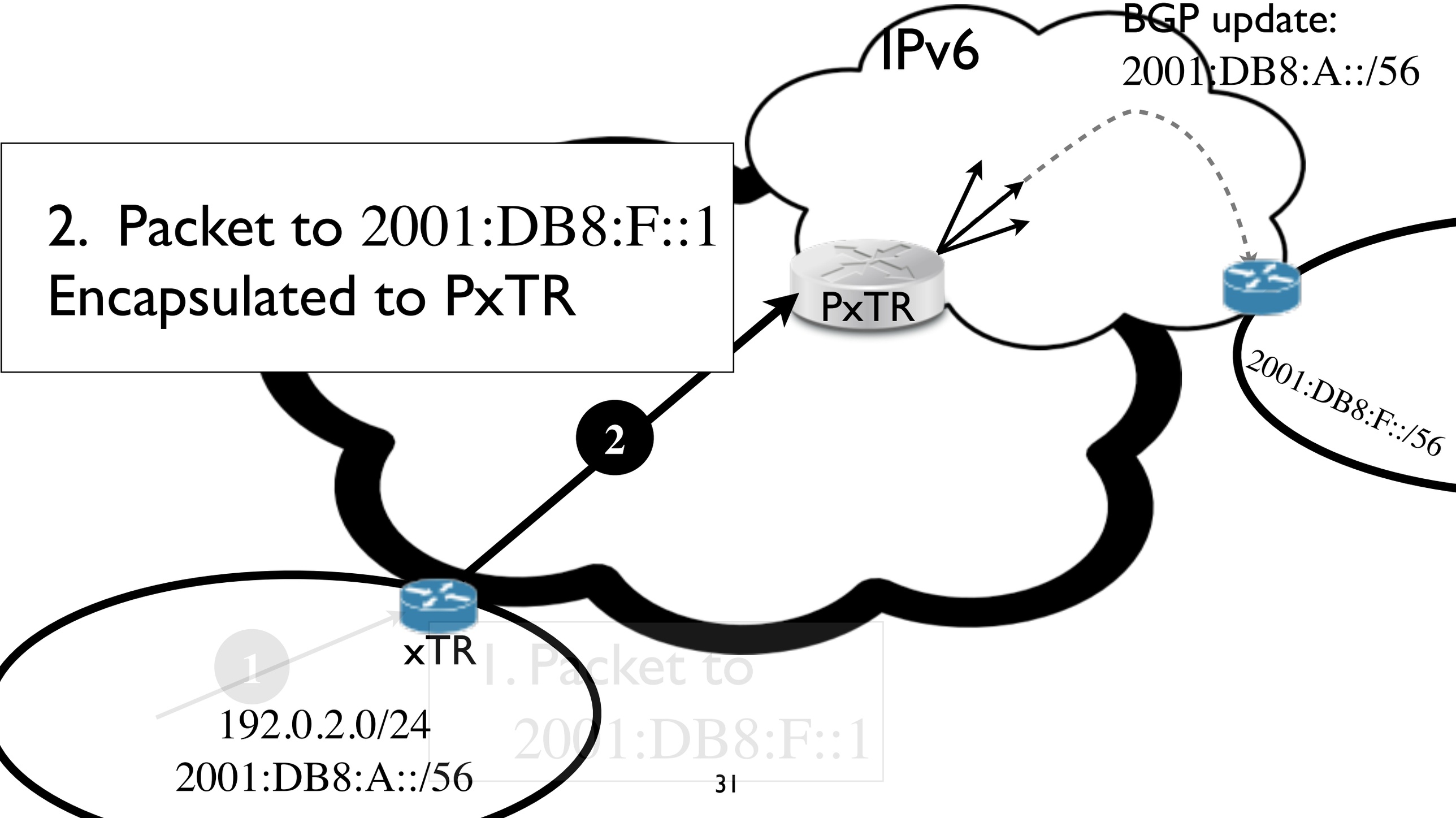


IPv6/IPv4 coexistence

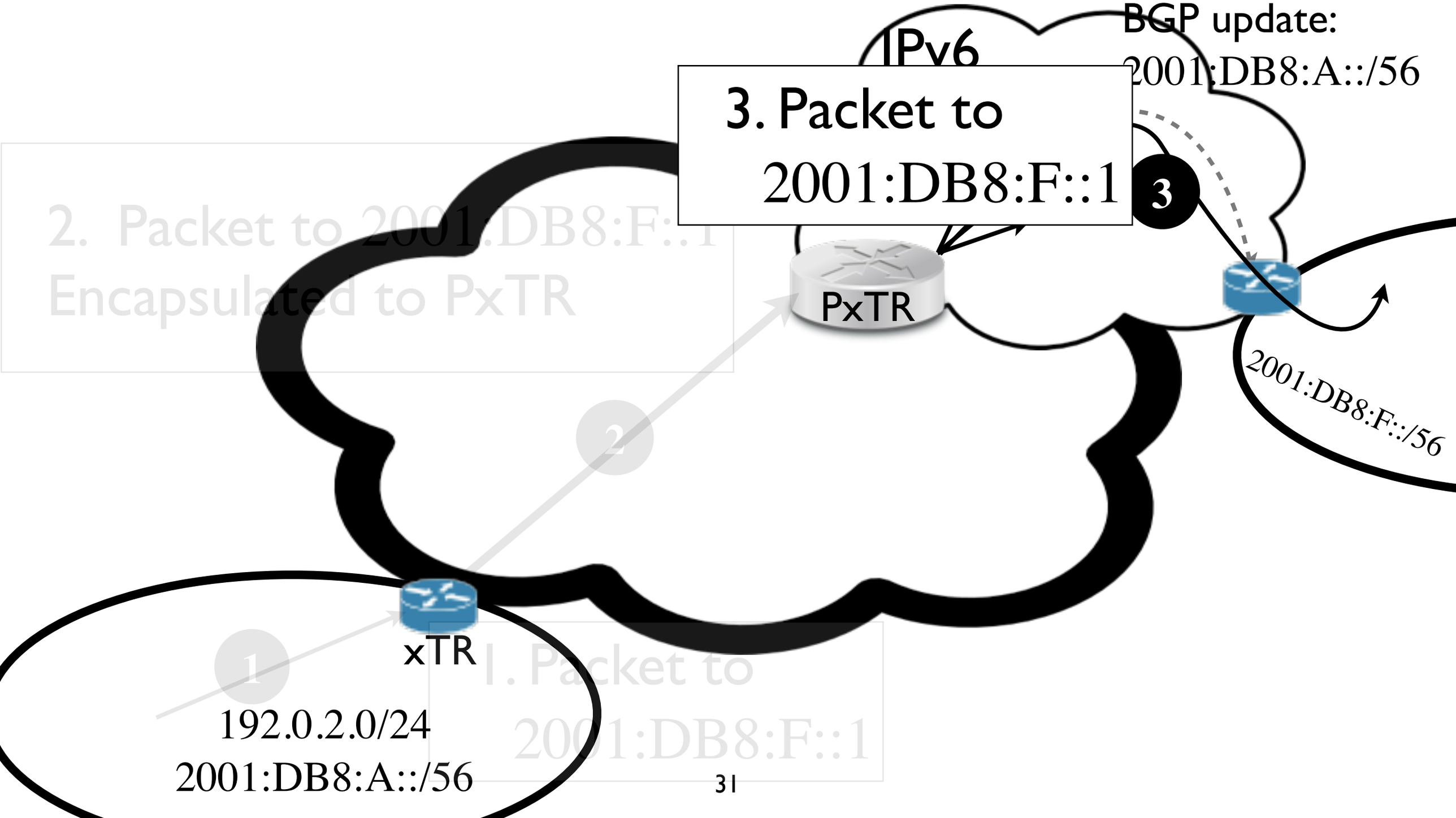


IPv6/IPv4 coexistence

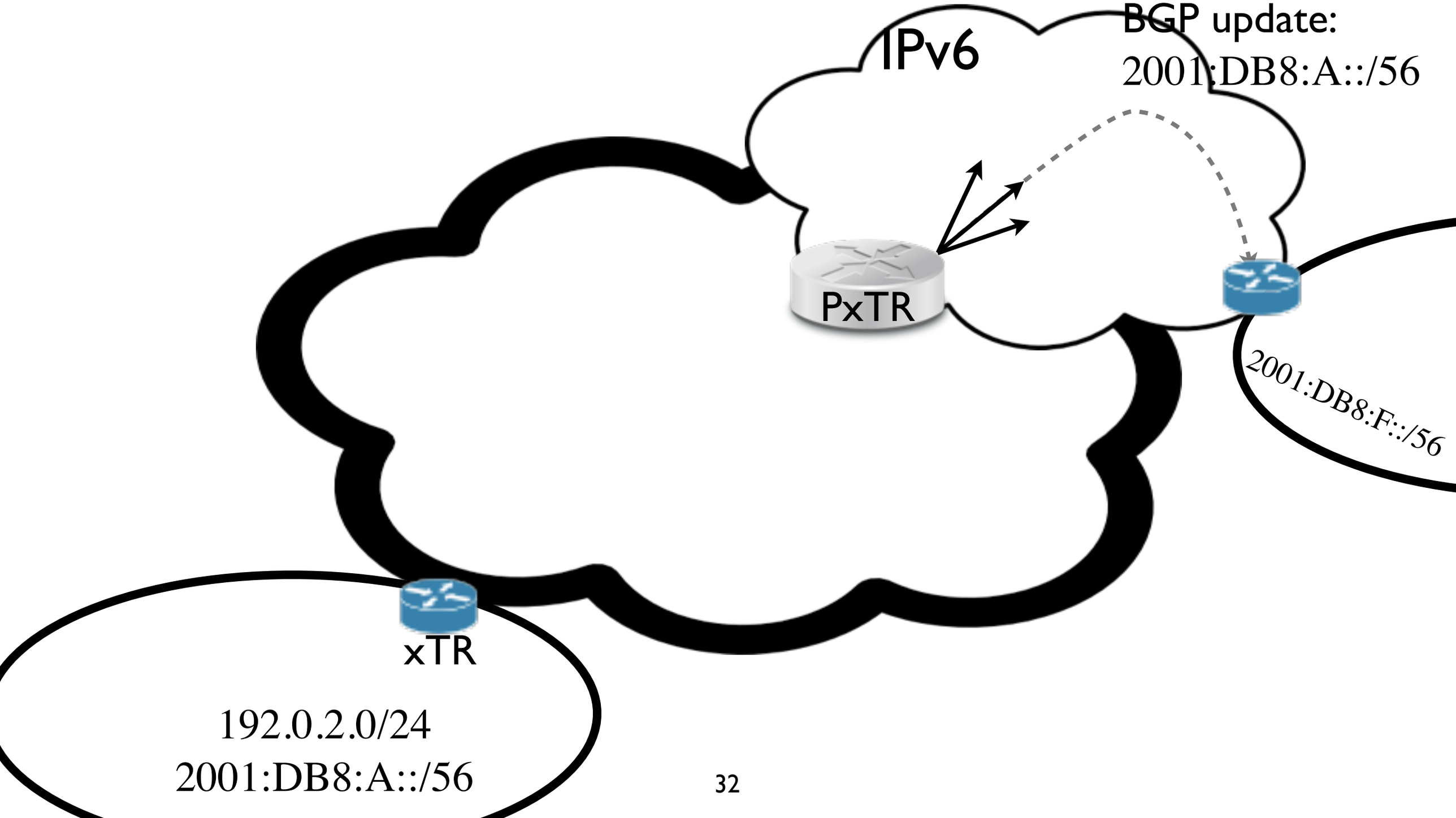
2. Packet to 2001:DB8:F::1 Encapsulated to PxTR



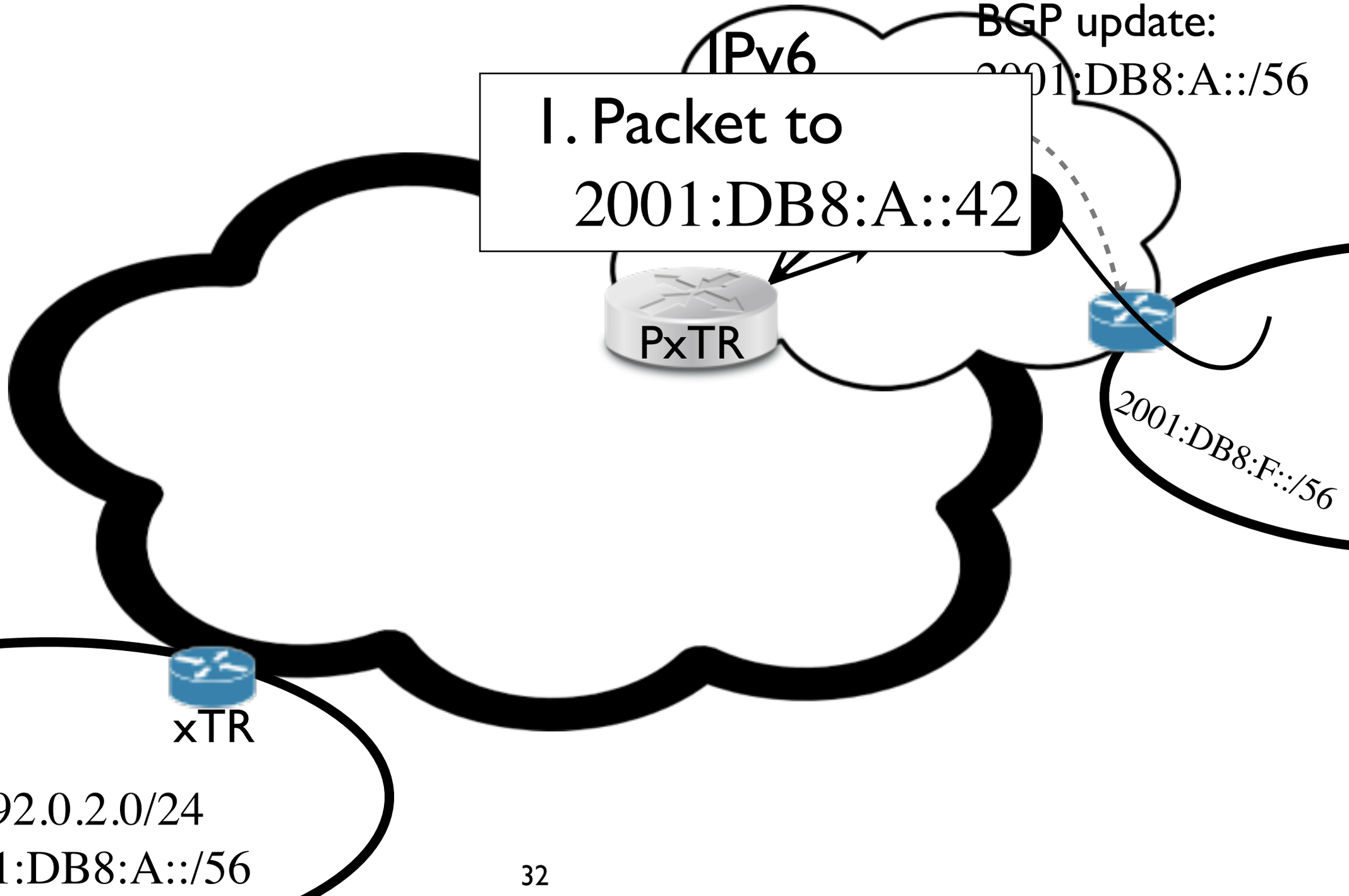
IPv6/IPv4 coexistence



IPv6/IPv4 coexistence

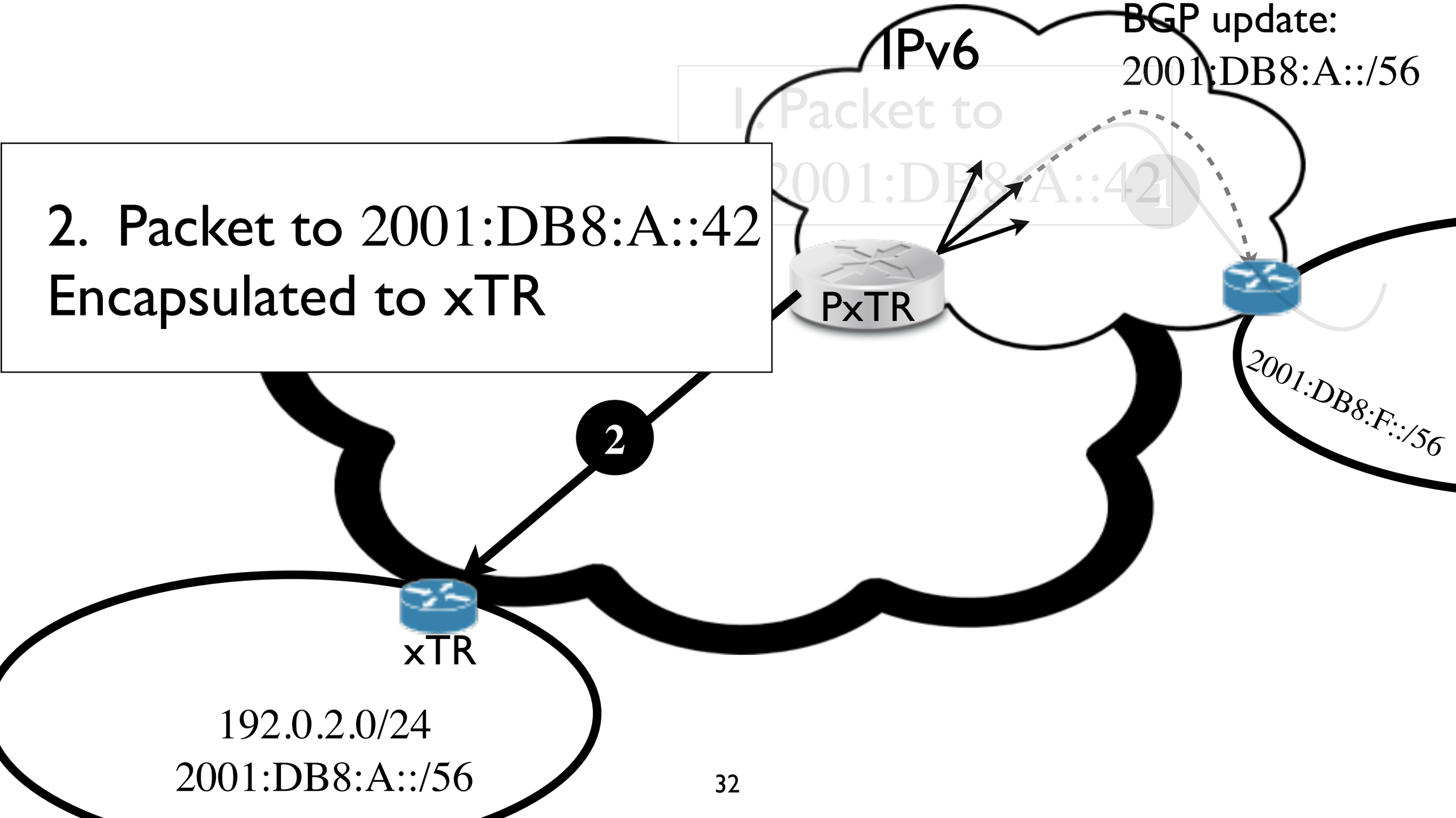


IPv6/IPv4 coexistence

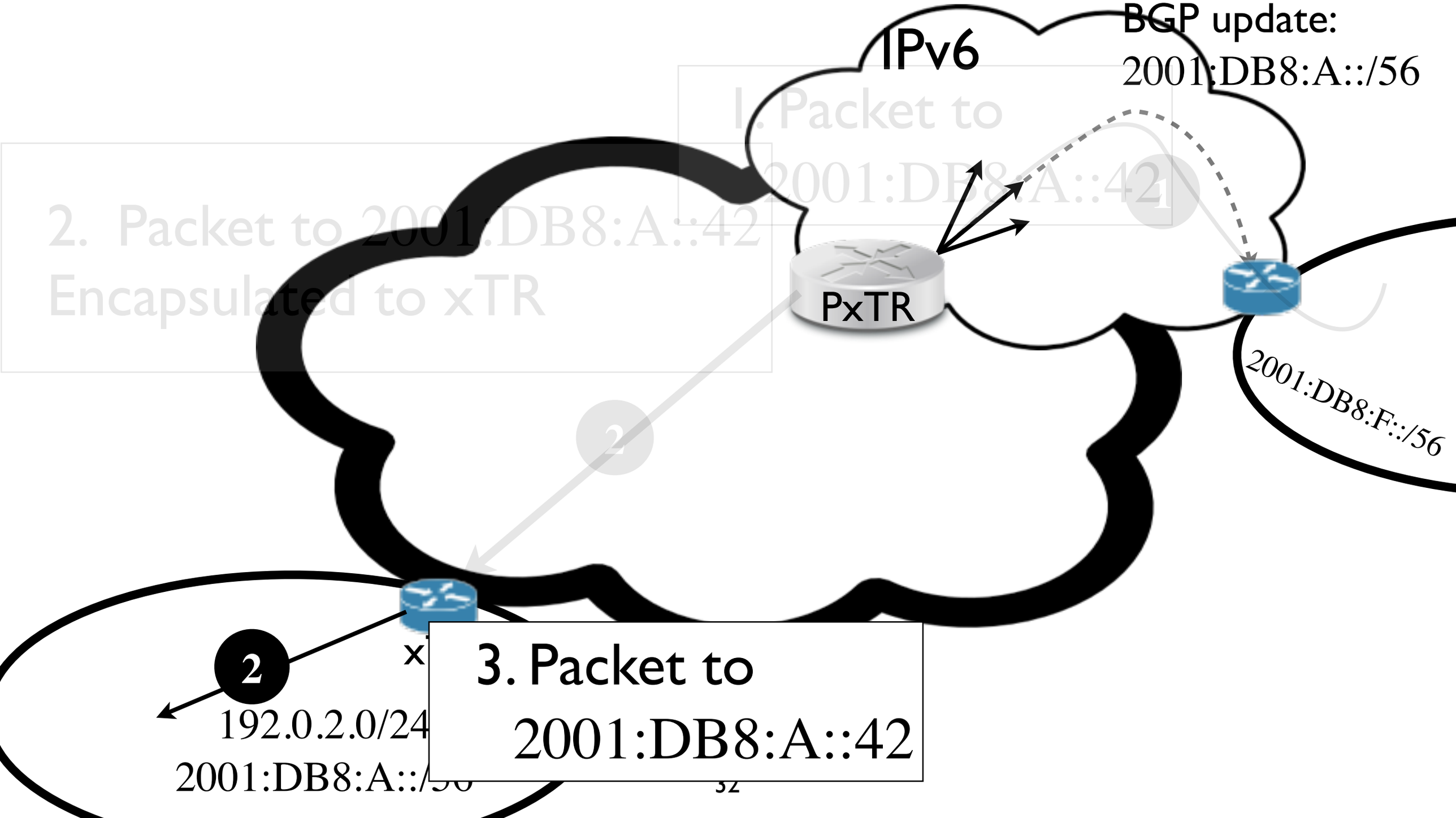


IPv6/IPv4 coexistence

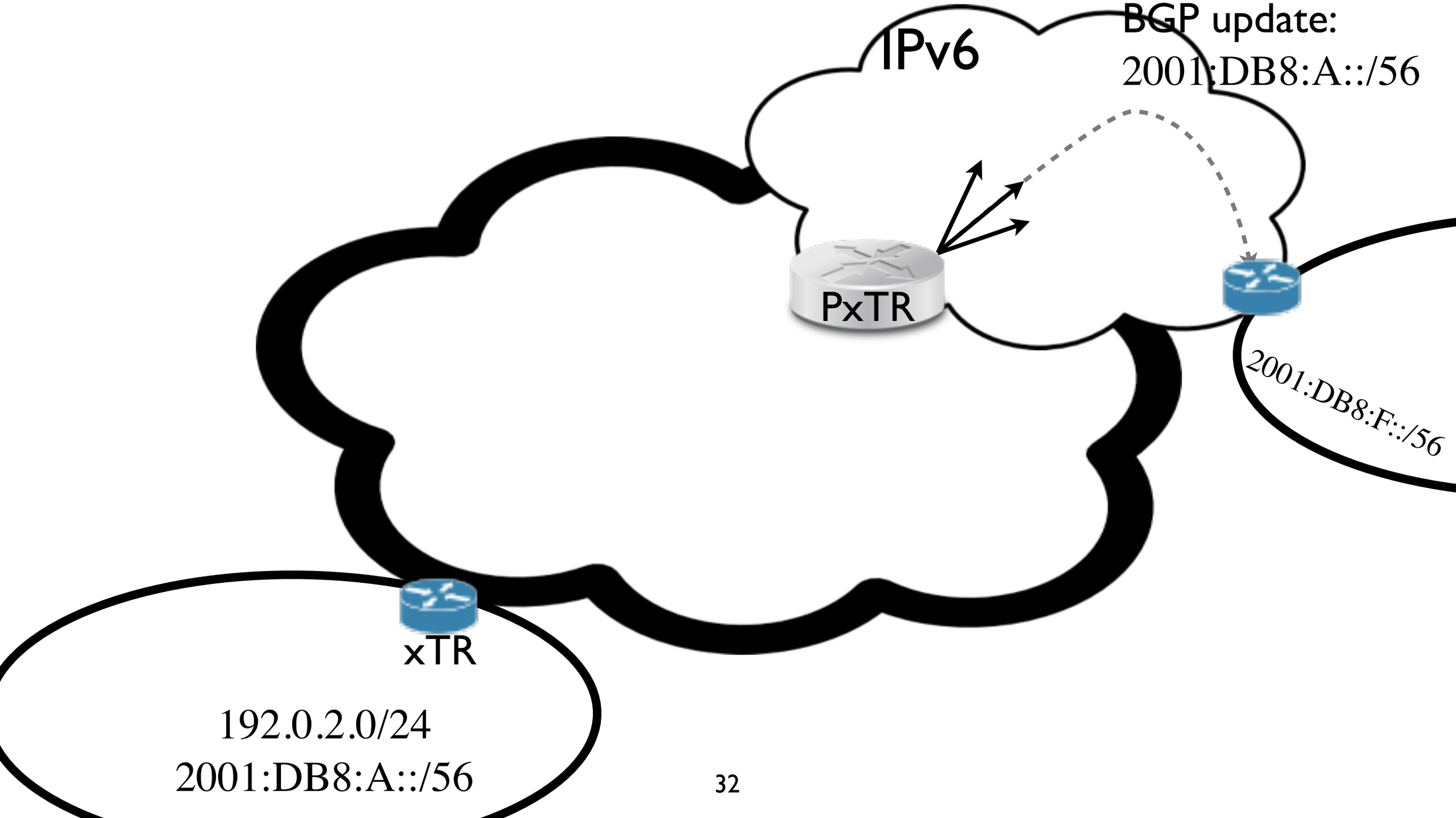
2. Packet to 2001:DB8:A::42 Encapsulated to xTR



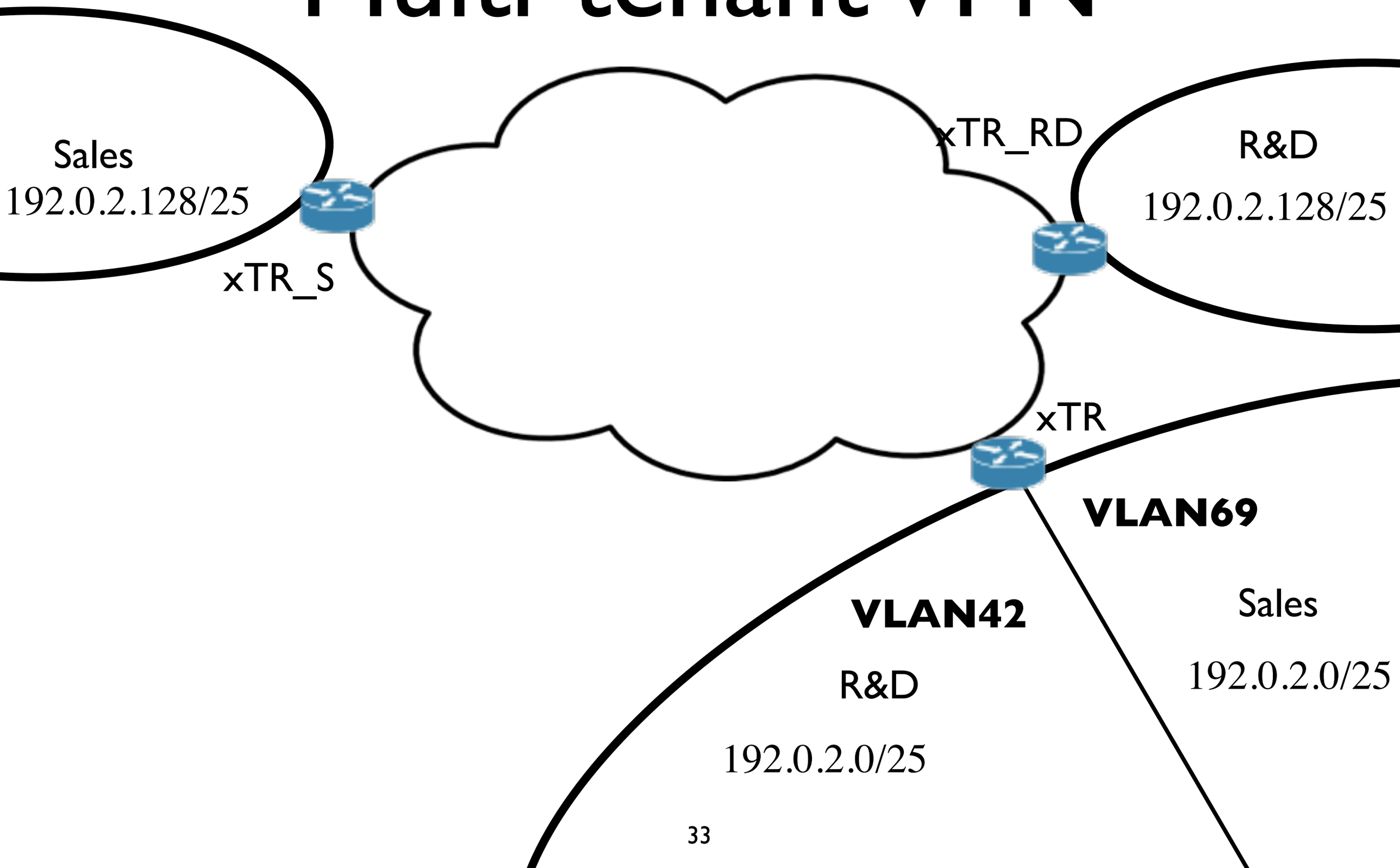
IPv6/IPv4 coexistence



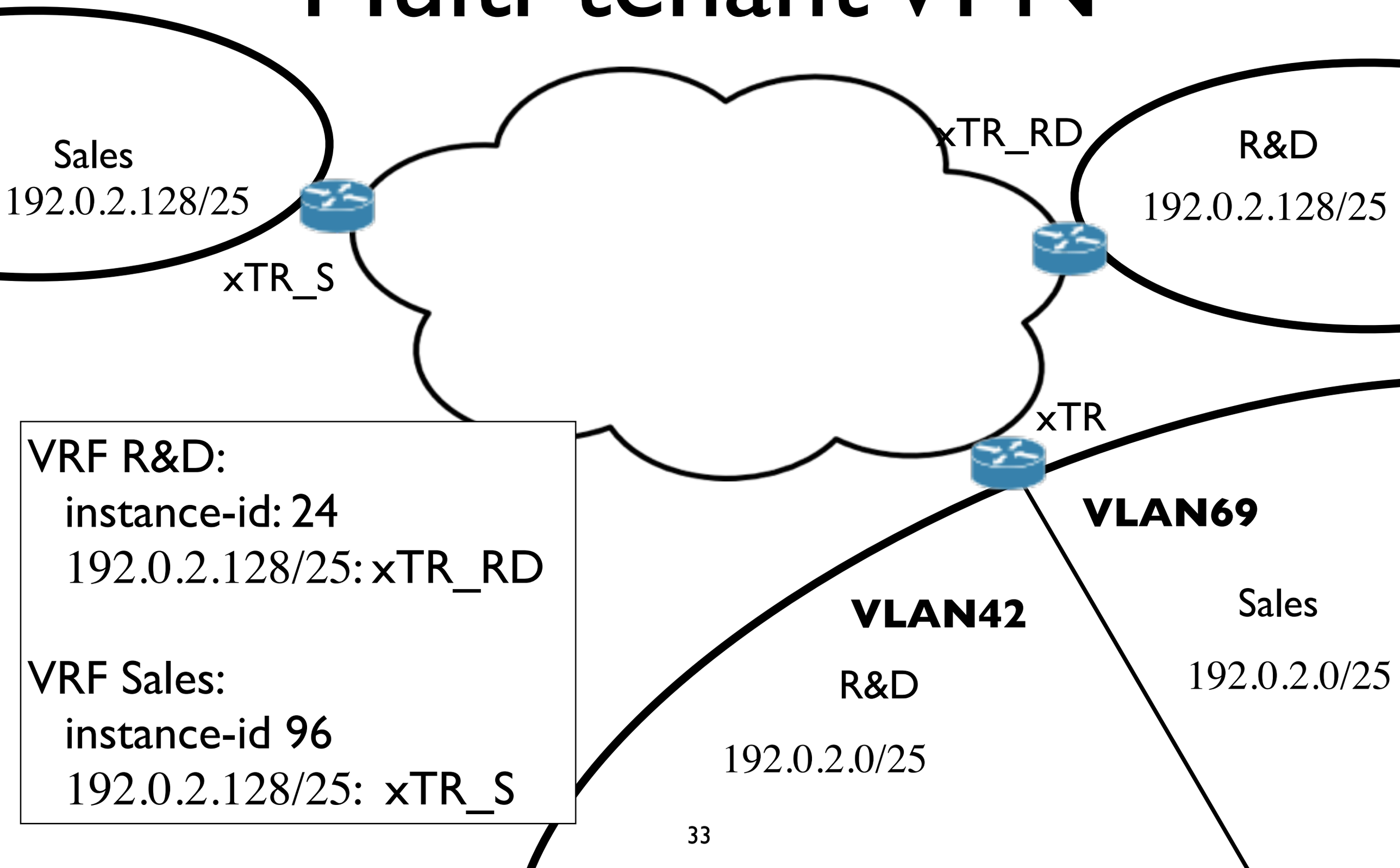
IPv6/IPv4 coexistence



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

1. Packet to 192.0.2.42

1

Sales
192.0.2.128/25

xTR_S

xTR_RD

R&D
192.0.2.128/25

xTR

VLAN69

Sales
192.0.2.0/25

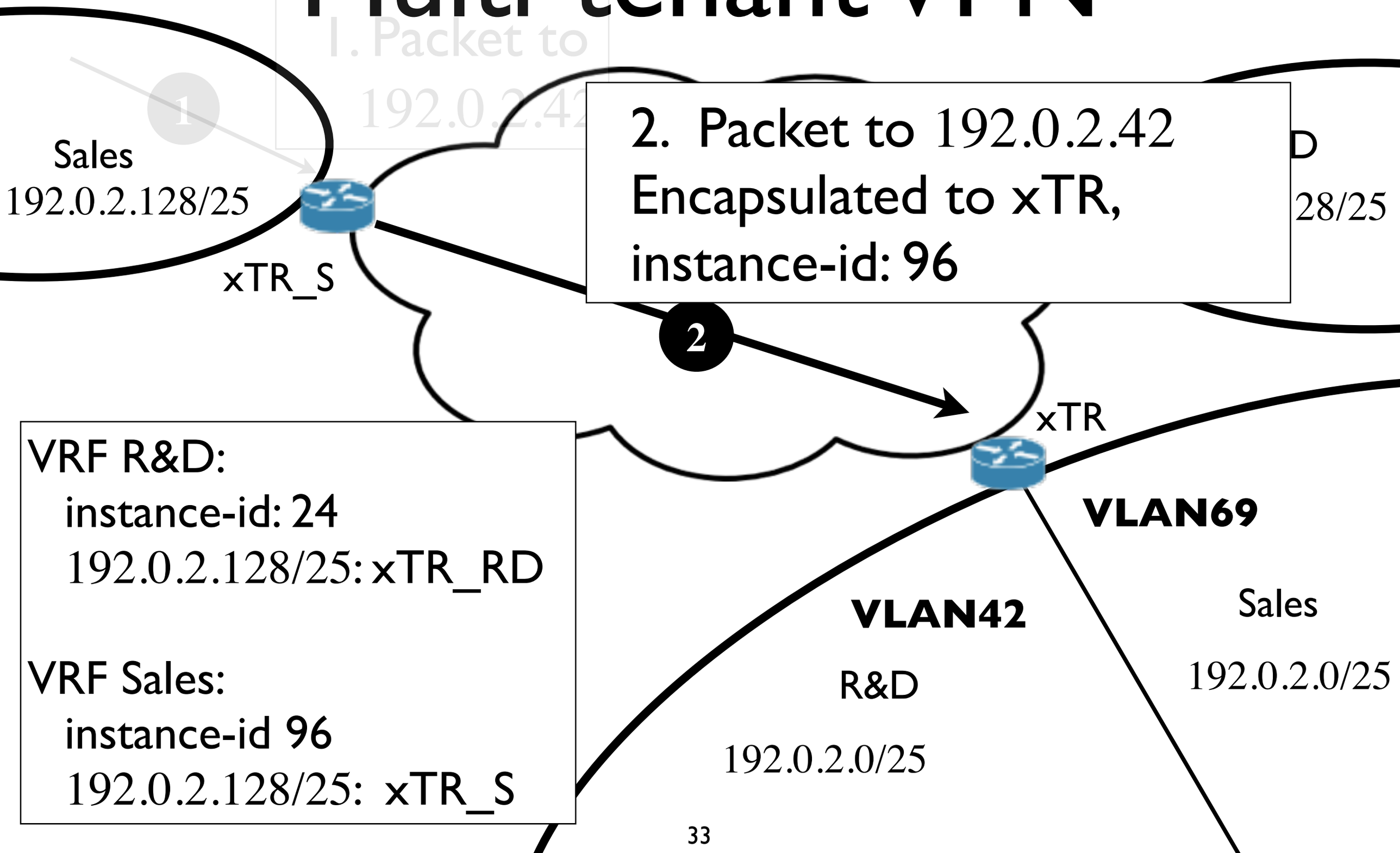
VLAN42

R&D
192.0.2.0/25

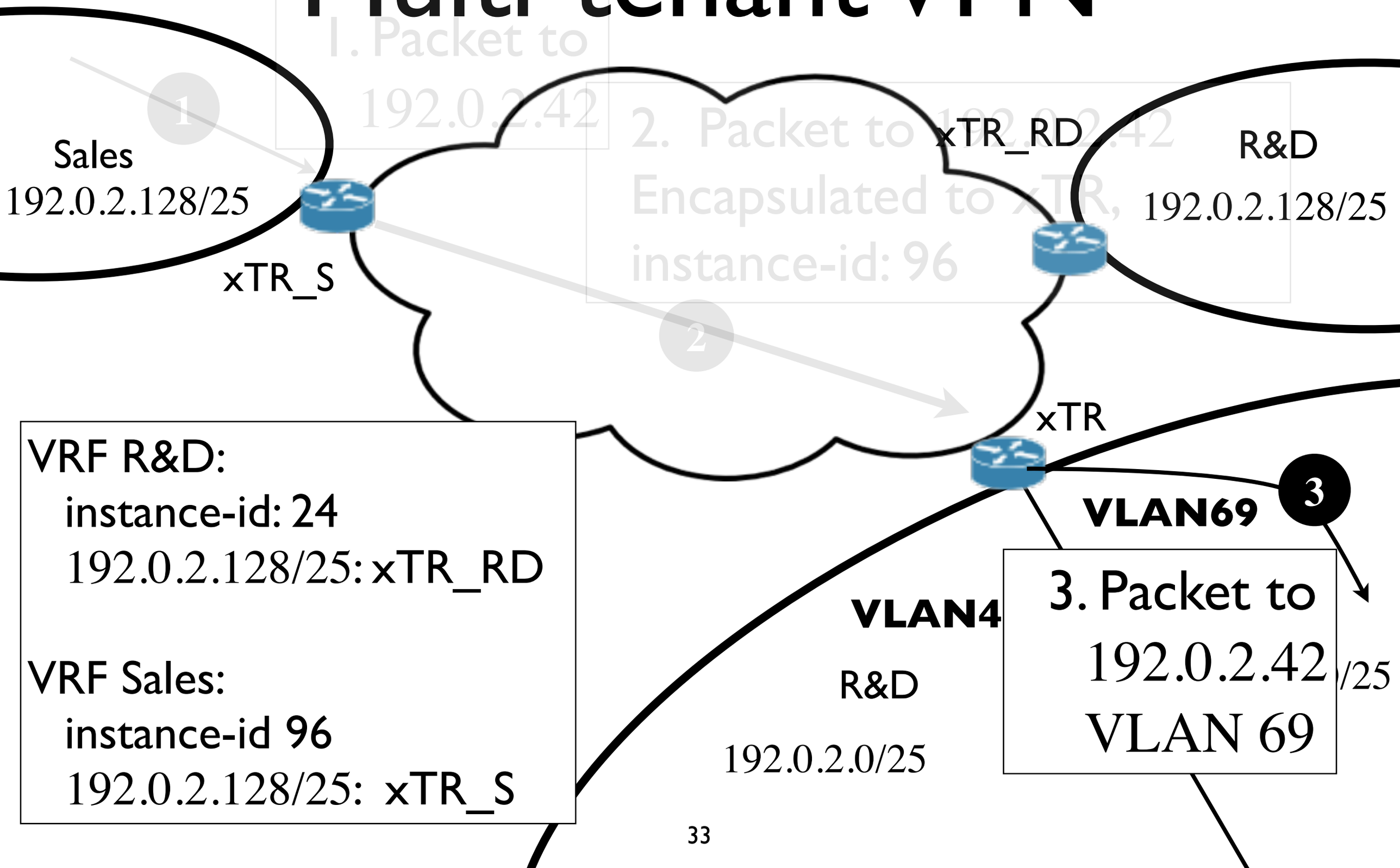
VRF R&D:
instance-id: 24
192.0.2.128/25: xTR_RD

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

Multi-tenant VPN



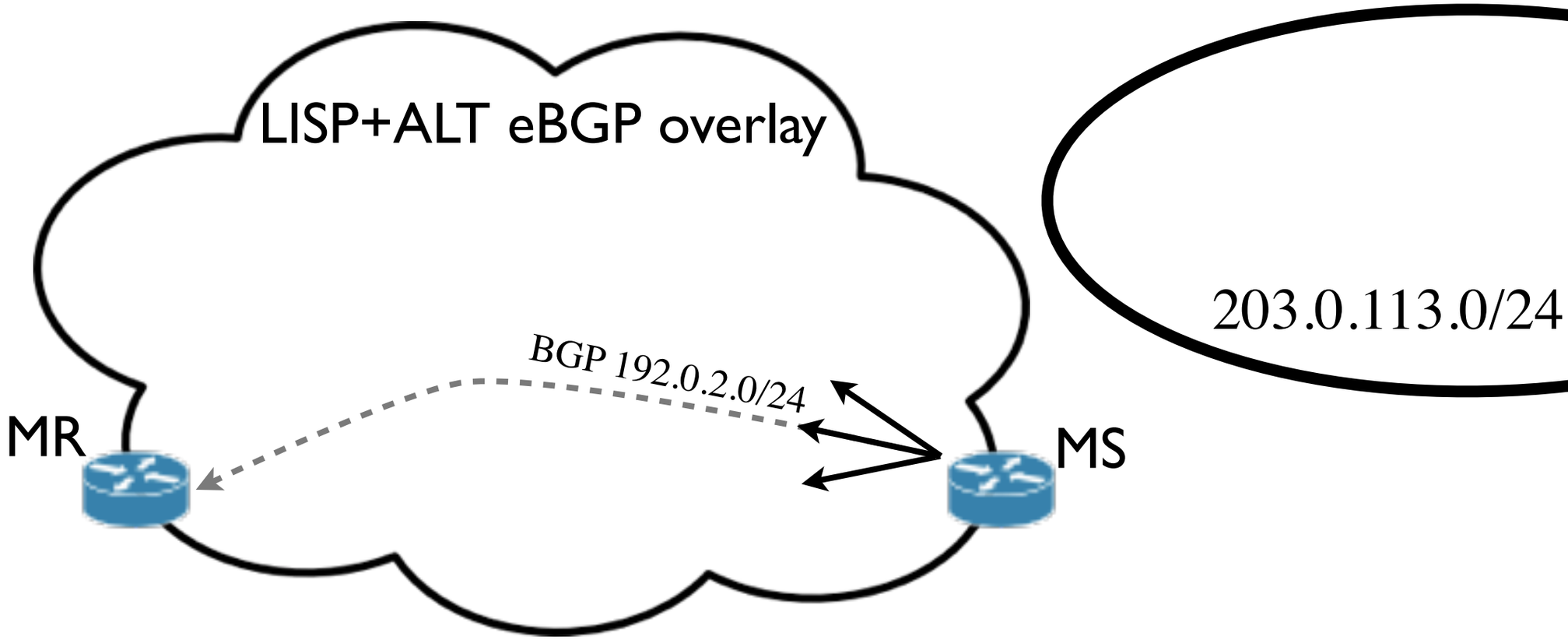
Multi-tenant VPN



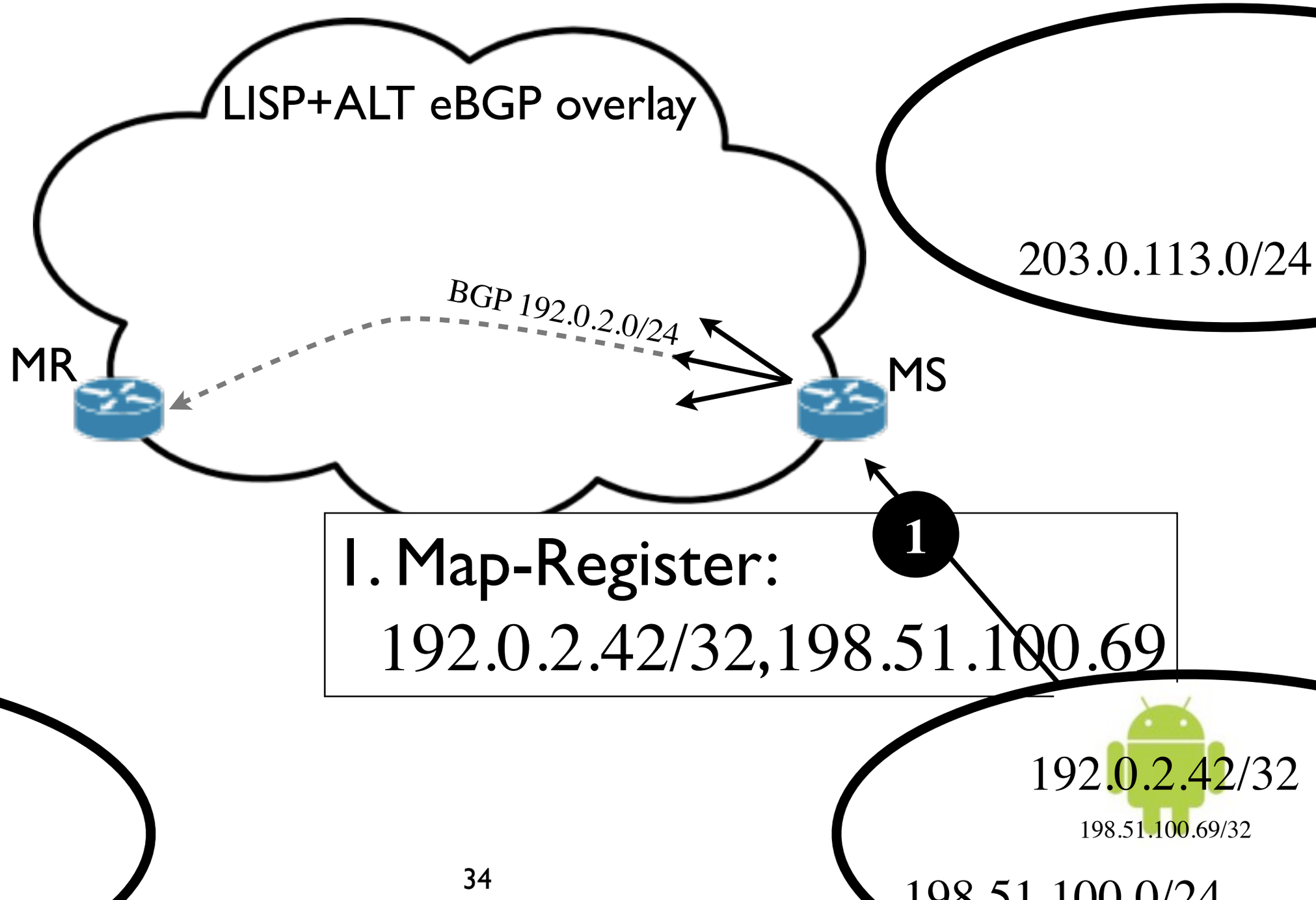
VRF R&D:
- instance-id: 24
- 192.0.2.128/25: xTR_RD

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

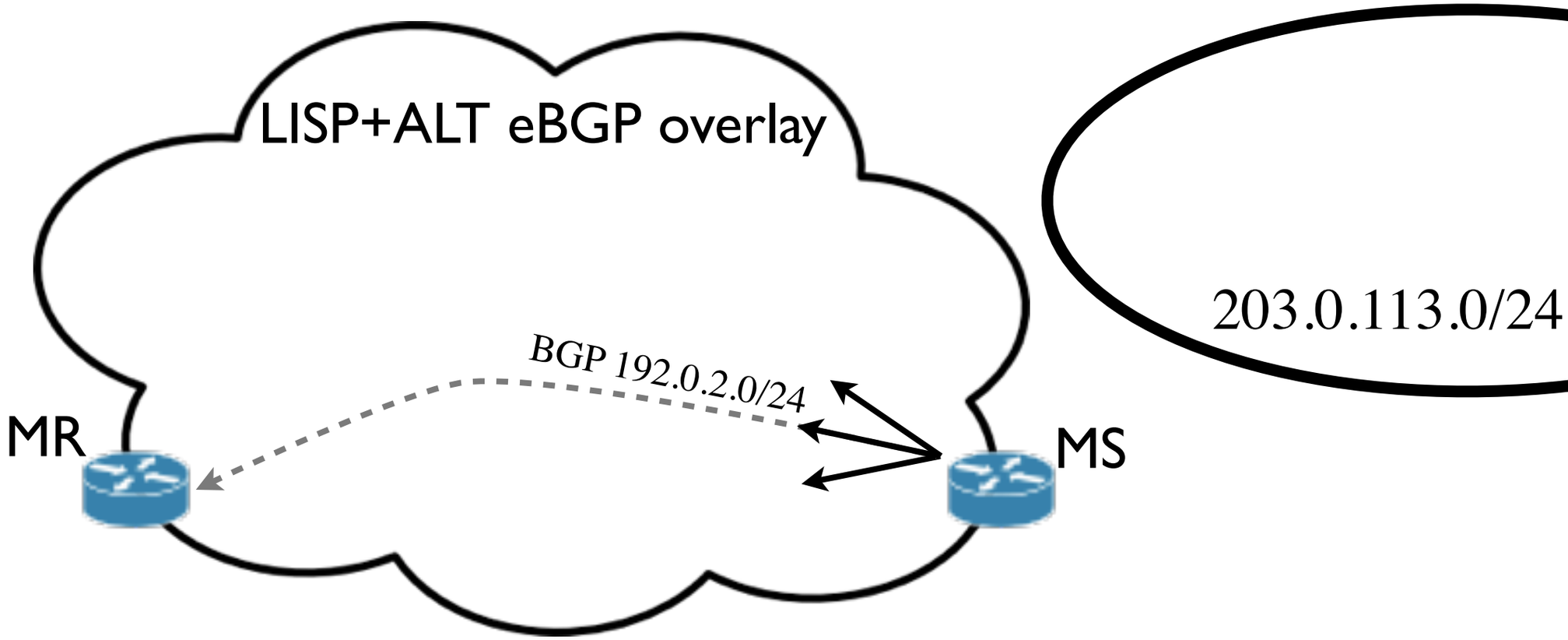
LISP Mobile Nodes



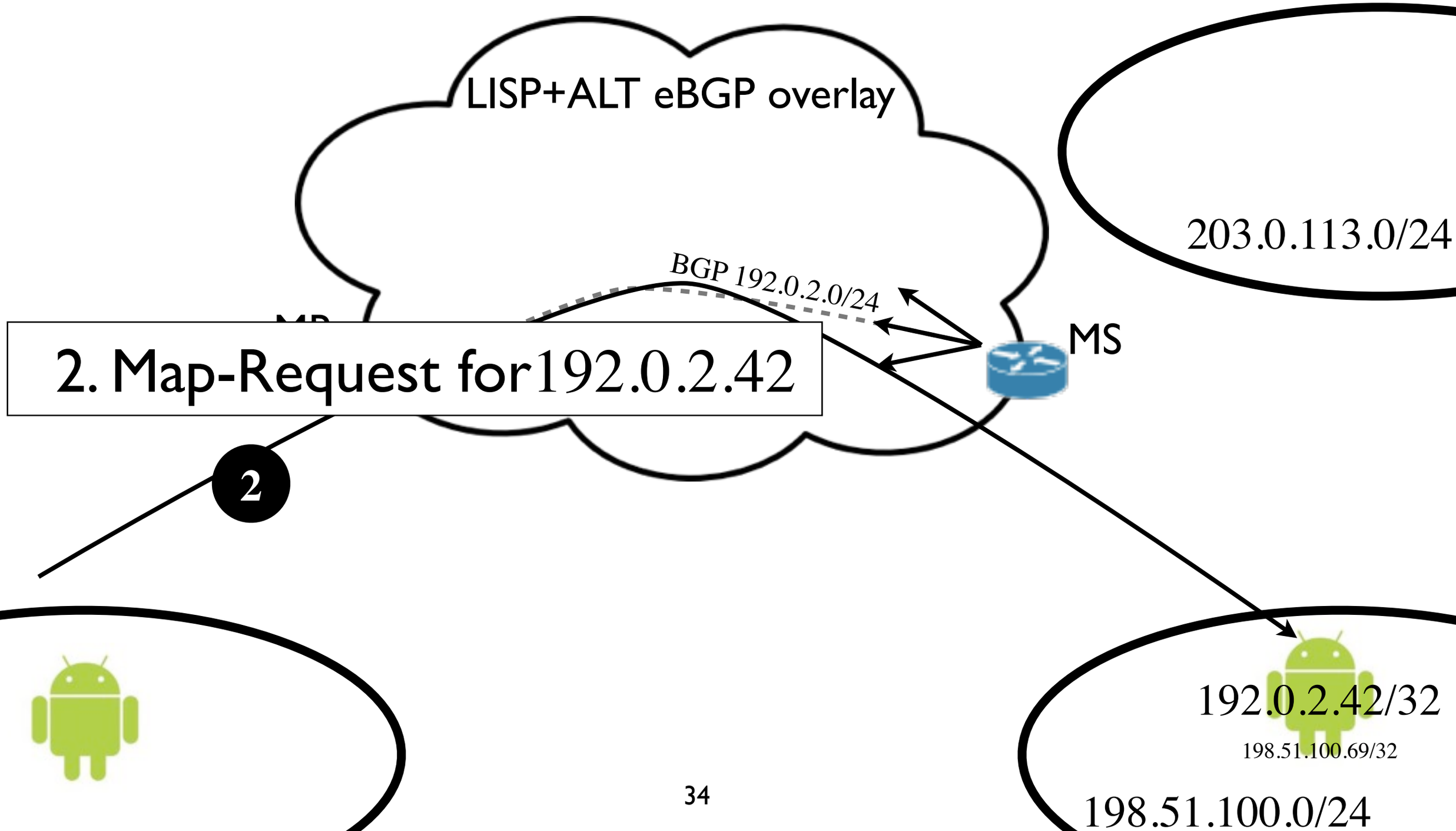
LISP Mobile Nodes



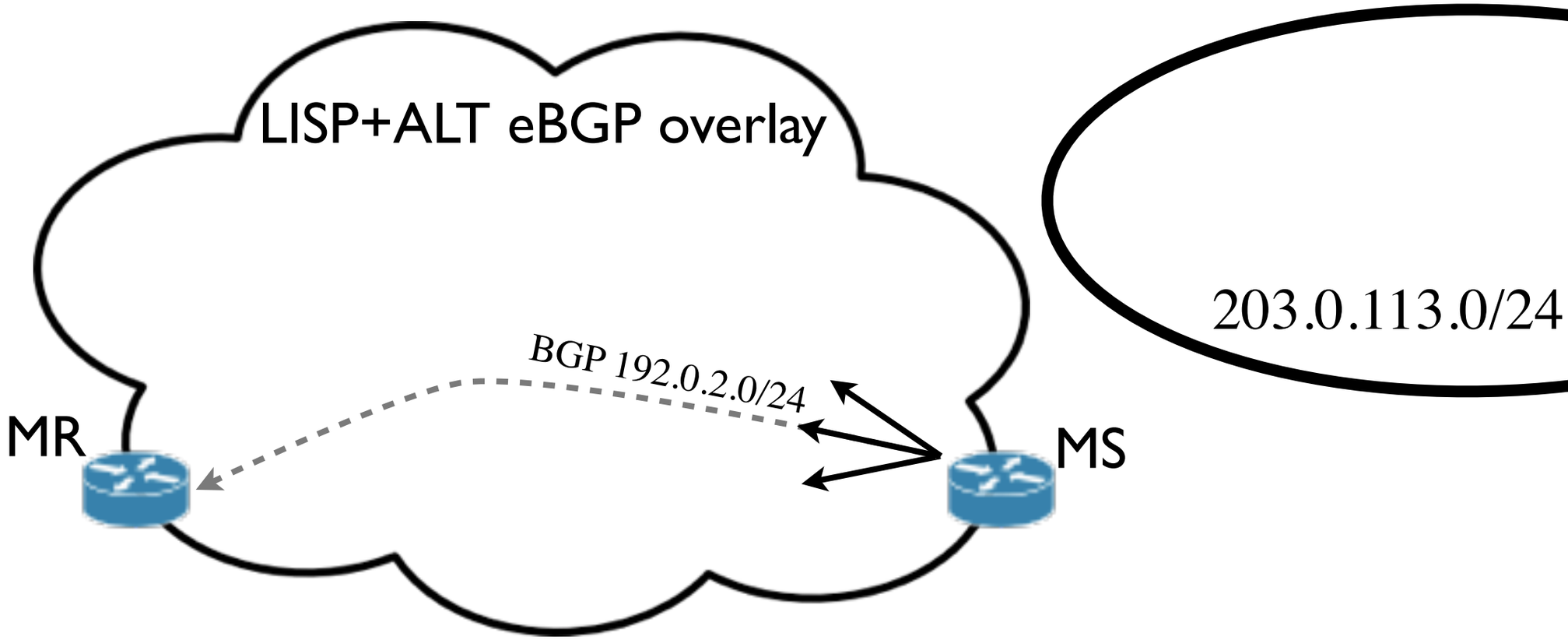
LISP Mobile Nodes



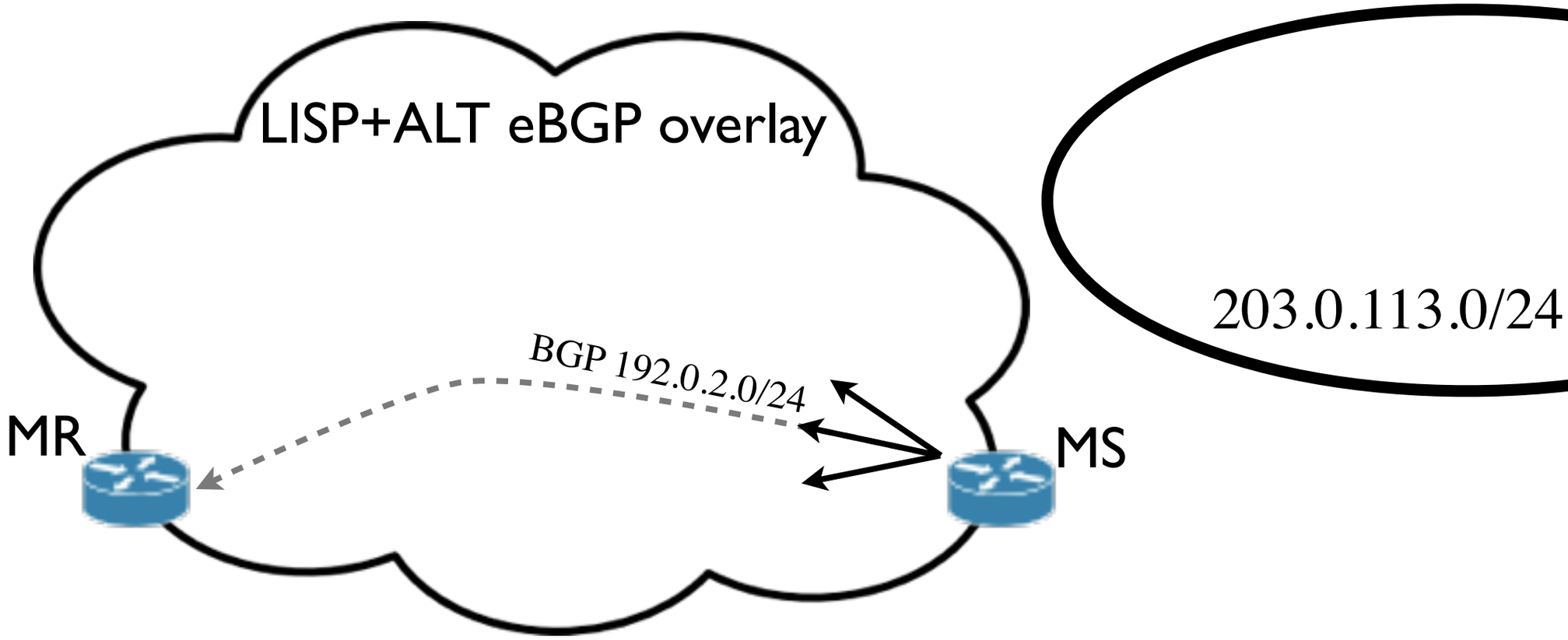
LISP Mobile Nodes

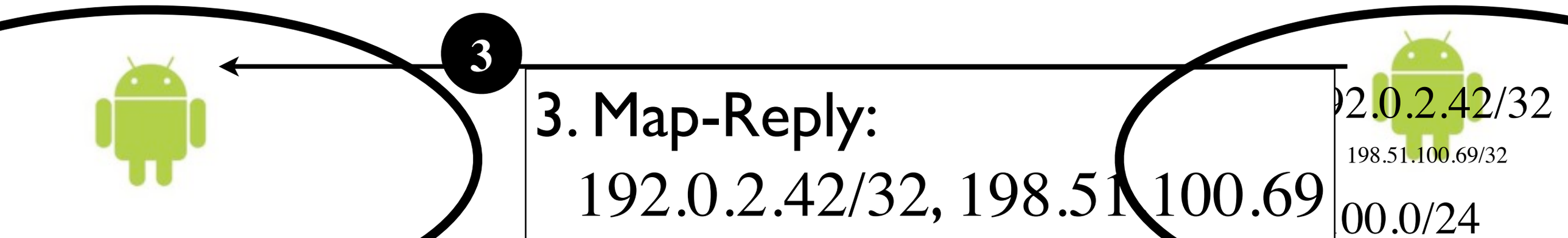


LISP Mobile Nodes

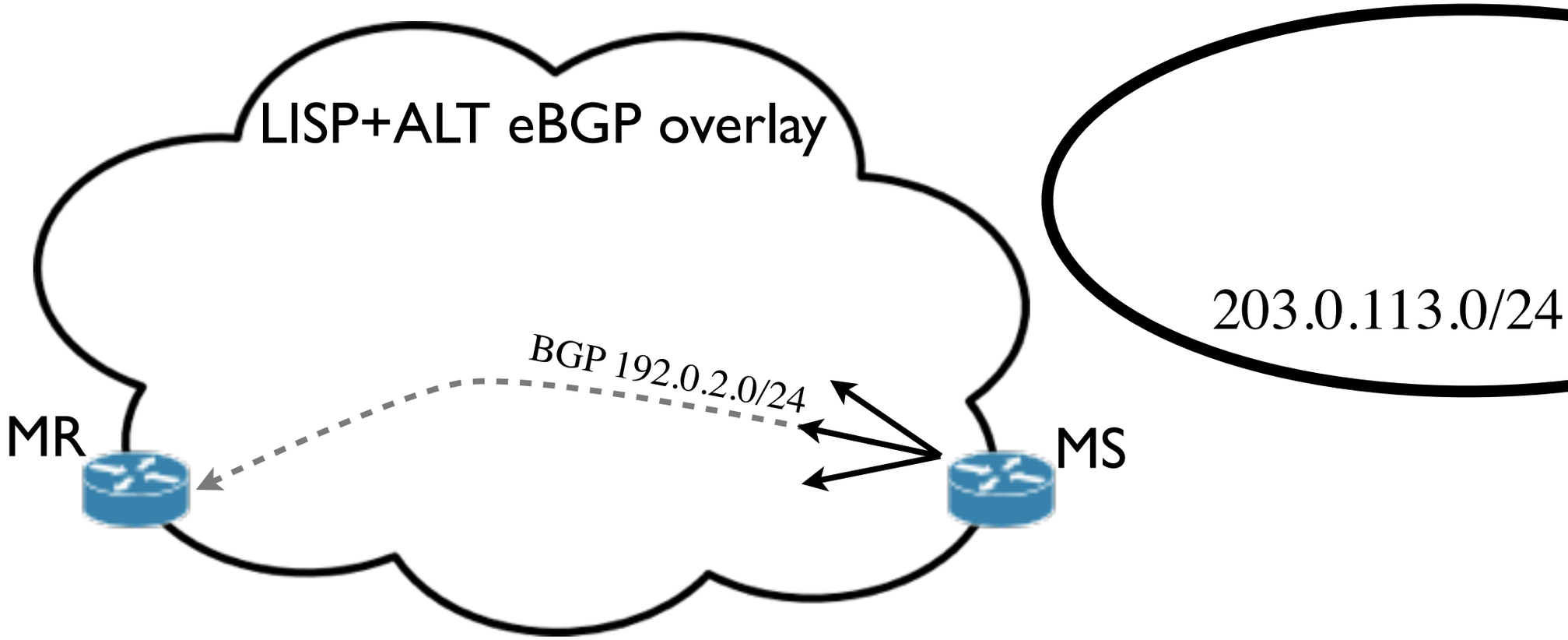


LISP Mobile Nodes

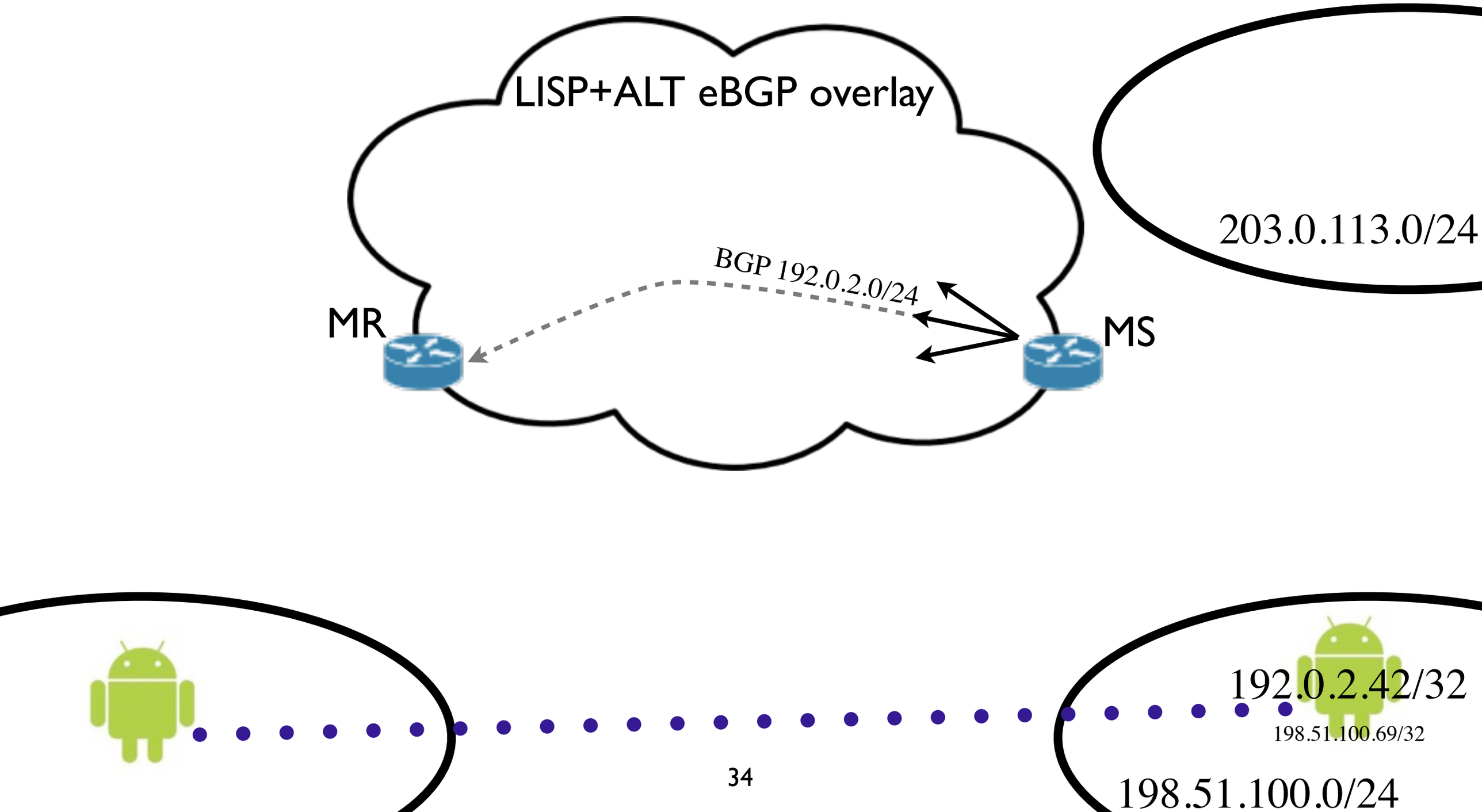




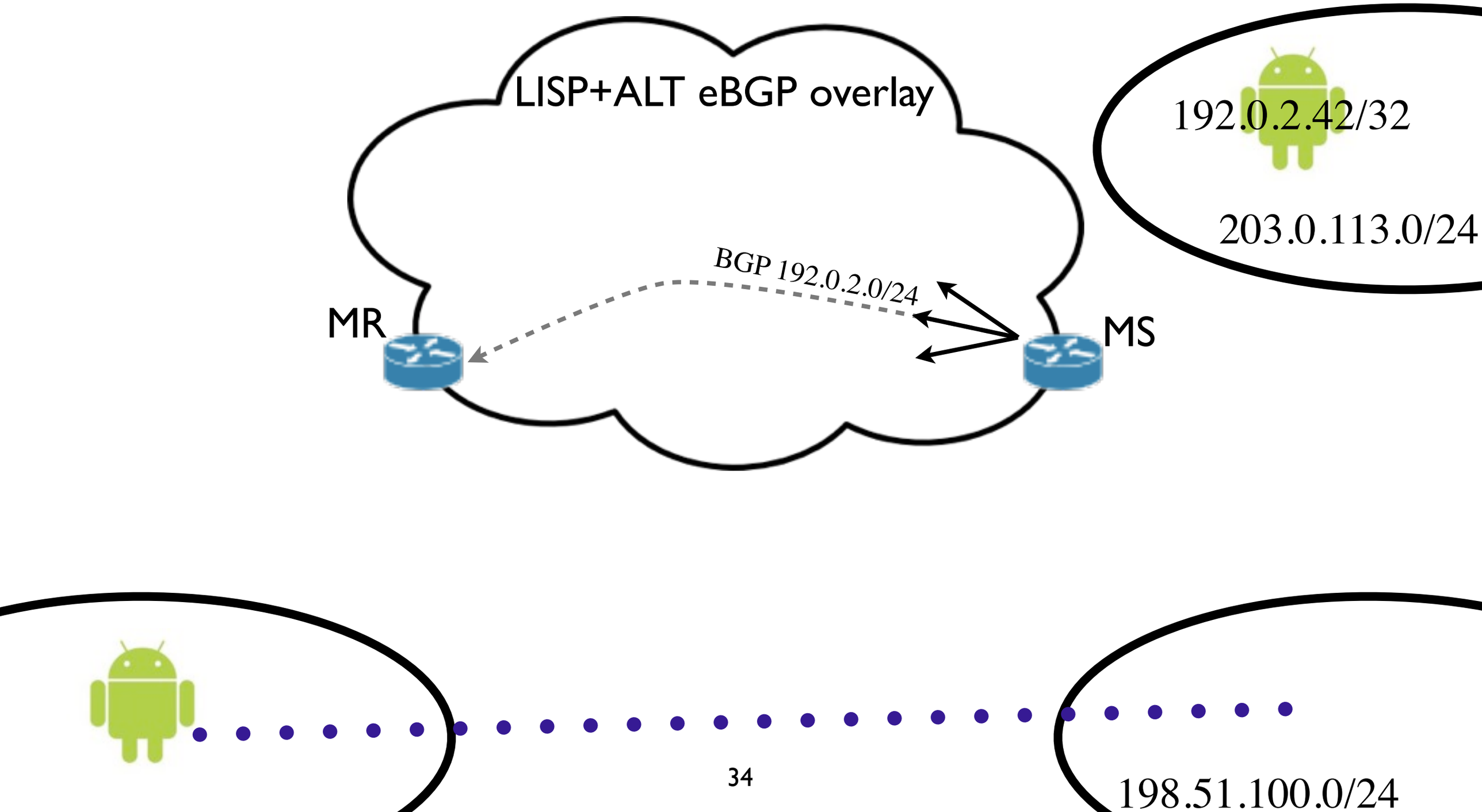
LISP Mobile Nodes



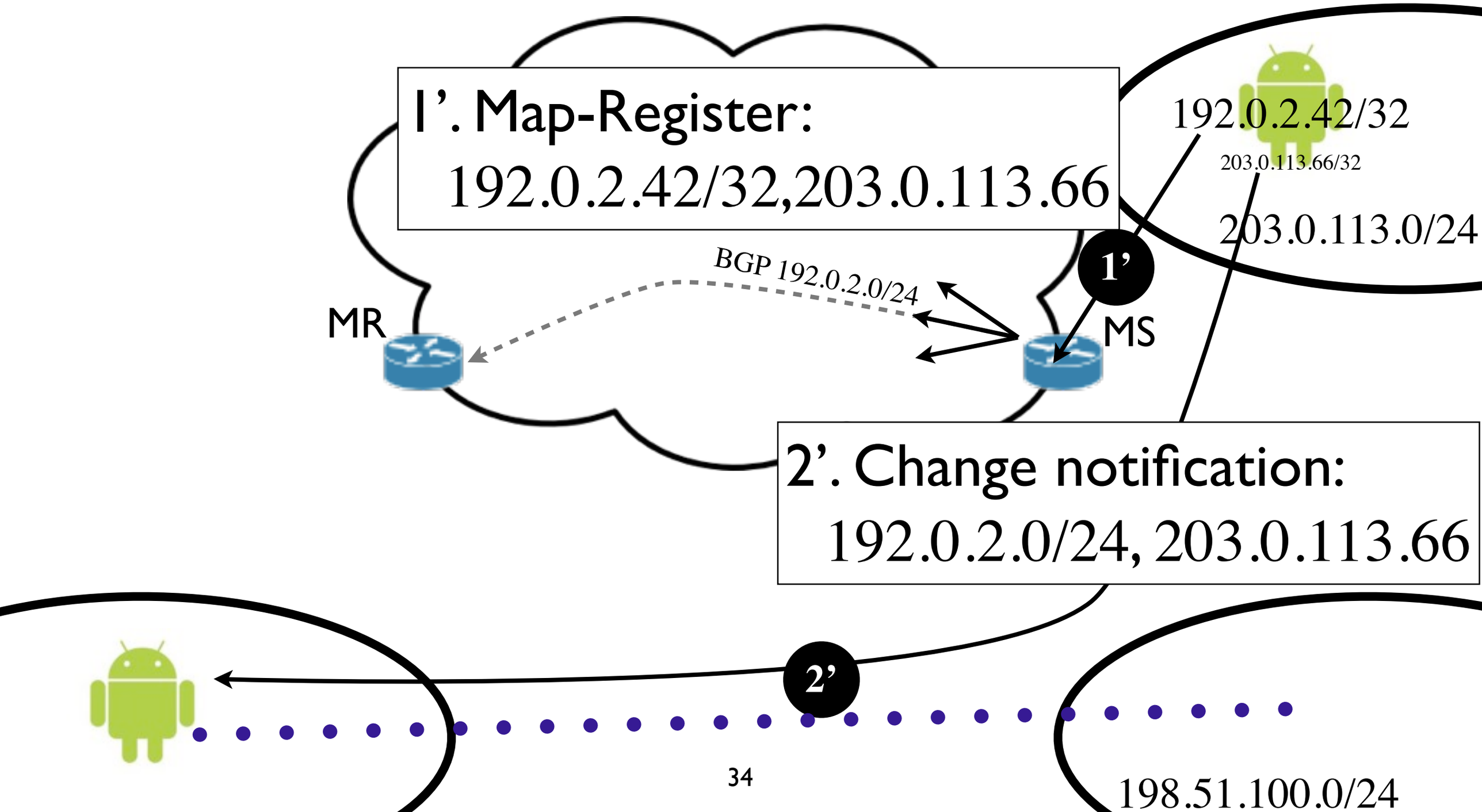
LISP Mobile Nodes



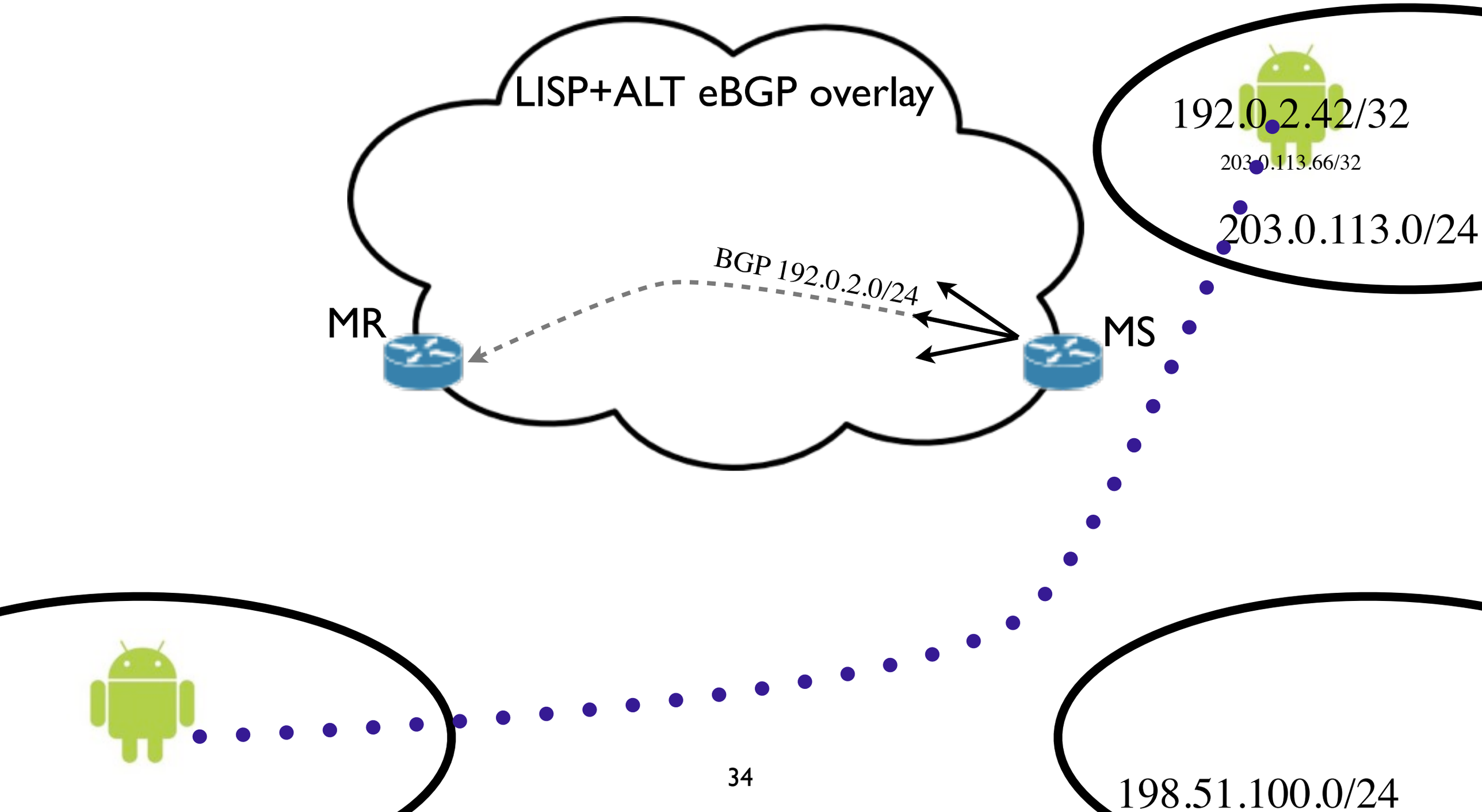
LISP Mobile Nodes



LISP Mobile Nodes



LISP Mobile Nodes



Open Questions

Network Resiliency

- Today, preserving the prefixes reachability is mainly performed locally
- In LISP, the legacy Internet is EID agnostic
  - Recovery only once ITR discover ETR failure
    - ITR periodically probes RLOCs
- Important part of LISP WG' new charter

Network Security

- Who control the mappings control the traffic
  - Denial of service
  - Eavesdropping
  - ...
- Firewalls? DPI?
- Important part of LISP WG' new charter

Conclusion

Summary

- LISP uses map-and-encap to separate IP's ID and locator roles
  - Improve network multi-homing
  - Transparent for the end-users
  - Incrementally deployable
  - Mobility support
- **See you @ 5:55 pm for a configuration hands on (possible to play with live routers)**

Bibliography

- draft-ietf-lisp
- draft-ietf-lisp-alt
- draft-fuller-ddt
- draft-ietf-lisp-ms
- draft-ietf-lisp-map-versioning
- draft-ietf-lisp-mn
- draft-ietf-lisp-lig
- draft-ietf-lisp-threats
- draft-ietf-lisp-sec
- draft-farinacci-lisp-lcaf
- Jackab et al., *LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System*
- Iannone and Bonaventure, *On the cost of caching locacor/ID mappings*
- Kim et al, *A Deep Dive into the LISP Cache and What ISPs Should Know about It*
- Ohmori et al, *Analyses on First Packet Drops of LISP in End-to-End Bidirectional Communications*
- www.lisp4.net

```
?? || /**/
```

Backup

LISP Philosophy 2/2

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

LISP is over UDP

- UDP to traverse firewalls/NAT, limit the impact of ECMP hashing on reordering...
- Source port is random
  - but per-flow source port is recommended
- Destination port is fixed to 4341
- Checksum is important if IPv6 RLOCs

Locator Status Bits

- A vector of 32 bits (L bit set to 1 if LSB is present)
- Each **source** locator is mapped to one position in the vector

```
if locator_status_bit[i] = 1
  RLOC i is reachable
else
  RLOC i is not reachable
```

- Each RLOC has an implicit position in the LSB
- How to set the bit? What is her meaning?

Mapping Systems

NERD: A Not-so-novel EID to RLOC Database

NERD

- The only proposed push model
  - Composed of 4 parts
    - a network database format;
    - a change distribution format;
    - a database retrieval/bootstrapping method;
    - a change distribution method

- Principles
  - An authority computes the mapping database based on the stored registrations
  - The database signed by the authority is stored on servers
  - ITR poll regularly the database servers to update their own mapping database

LISP+ALT

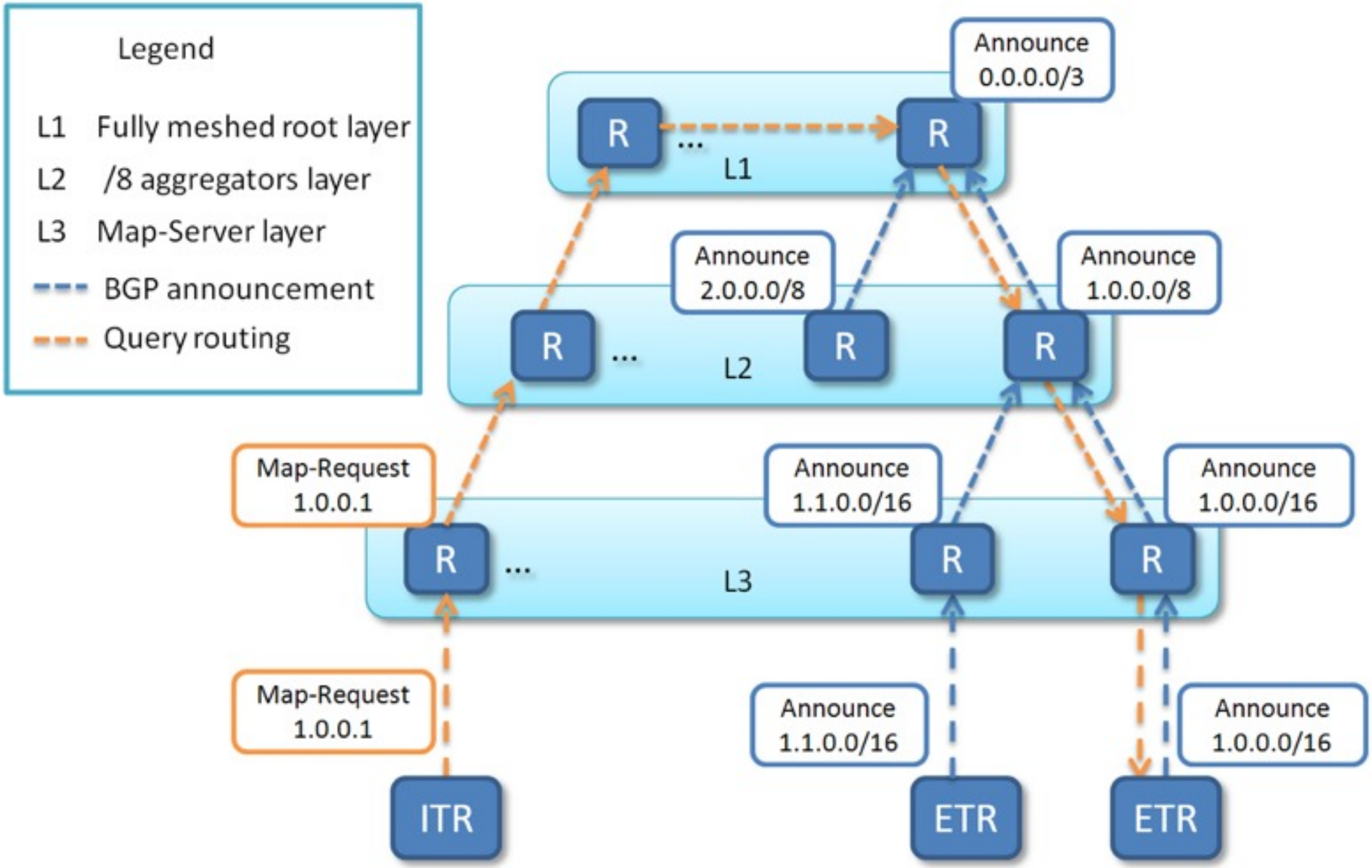
LISP+ALT

- An pull model mapping system
- A mapping mechanism that relies on an alternate topology to distribute mapping requests to mapping servers
  - LISP ITR routers sending mapping request messages to ALT routers
  - ALT routers forward those mapping messages between themselves on an overlay topology built by using GRE tunnels

LISP+ALT

- BGP announces **where** the mappings can be found
- Map-Requests are forwarded on the ALT
- Map-Replies are forwarded on the legacy Internet (directly sent to the ITRs' RLOC)
- ! BGP does not give the mappings !

LISP+ALT


Legend
L1 Fully meshed root layer
L2 /8 aggregators layer
L3 Map-Server layer
BGP announcement
Query routing
Announce 0.0.0.0/3
R
...
L1
R
Announce 2.0.0.0/8
Announce 1.0.0.0/8
R
...
L2
R
R
Map-Request 1.0.0.1
Announce 1.1.0.0/16
Announce 1.0.0.0/16
R
...
L3
R
R
Map-Request 1.0.0.1
Announce 1.1.0.0/16
Announce 1.0.0.0/16
ITR
ETR
ETR

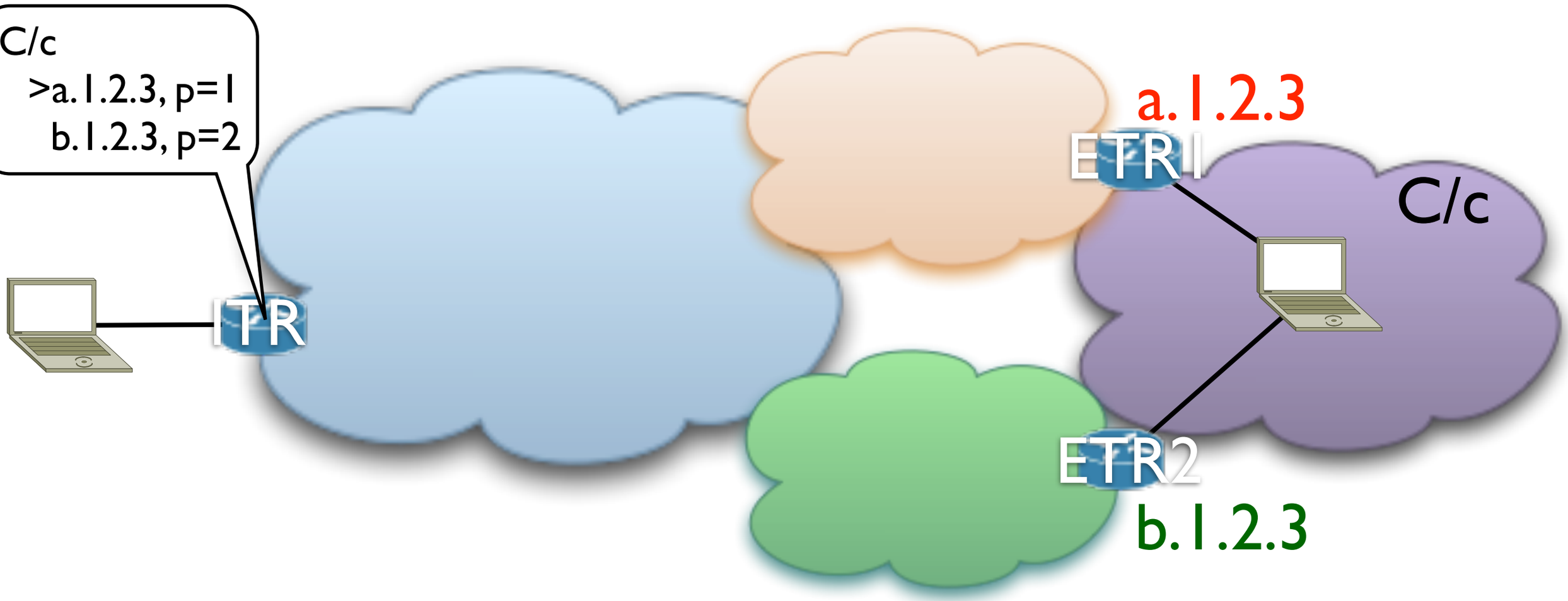
LISP+ALT issues

- Complex system with tunnels, BGP protocol (no discussion about policies), ...
- Still relies on lots of error-prone manual configuration
- Scalability will depend on whether aggregation will be possible
- If mapping requests are lost due to congestion, difficult to diagnose the problem or send them via another path
- Security needs to be studied

LISP+ALT, big question

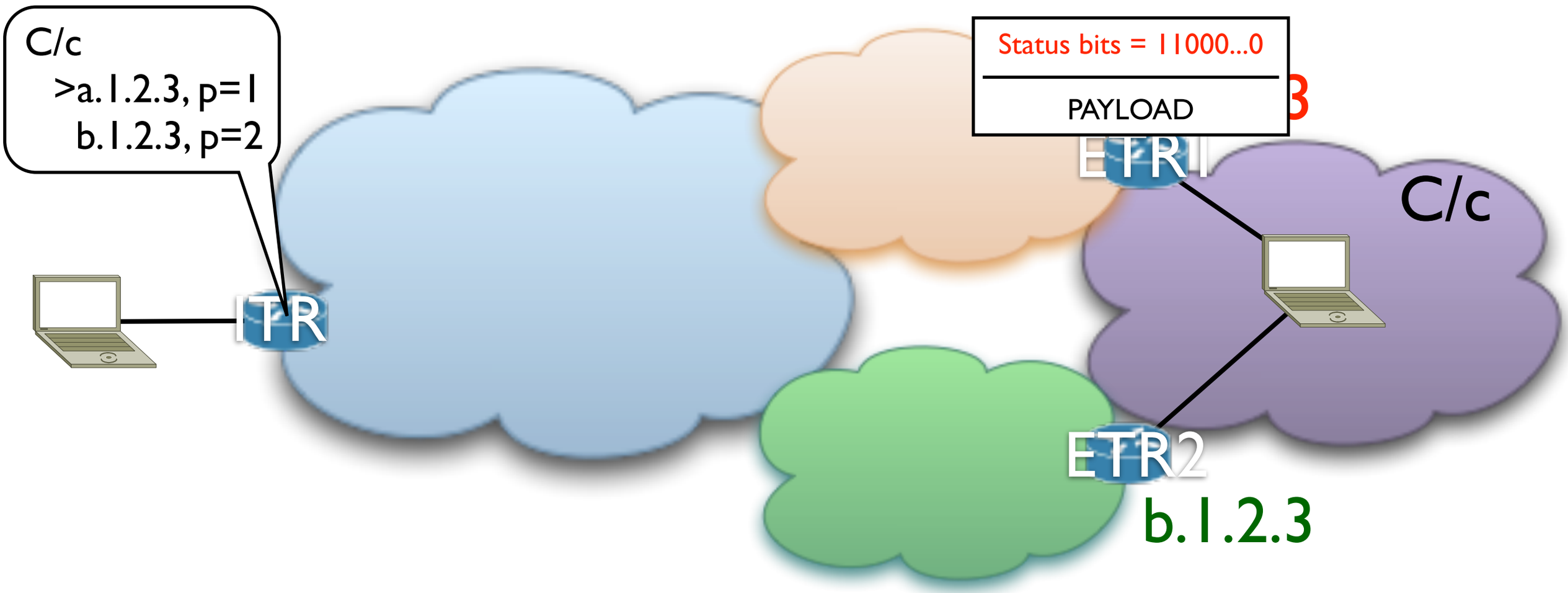
- How to deploy it?
- aggregation vs recovery vs TE
- how to cache the mappings?

Solving the reachability problem with the locator status bits



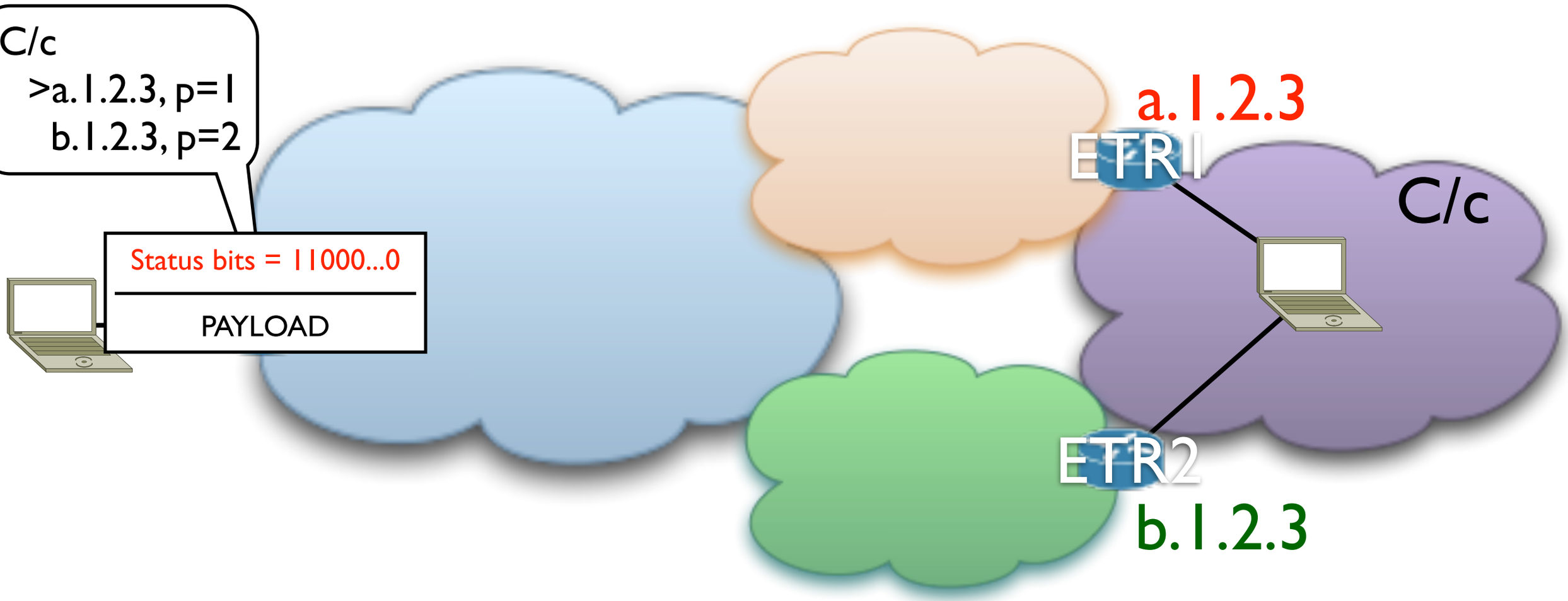
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



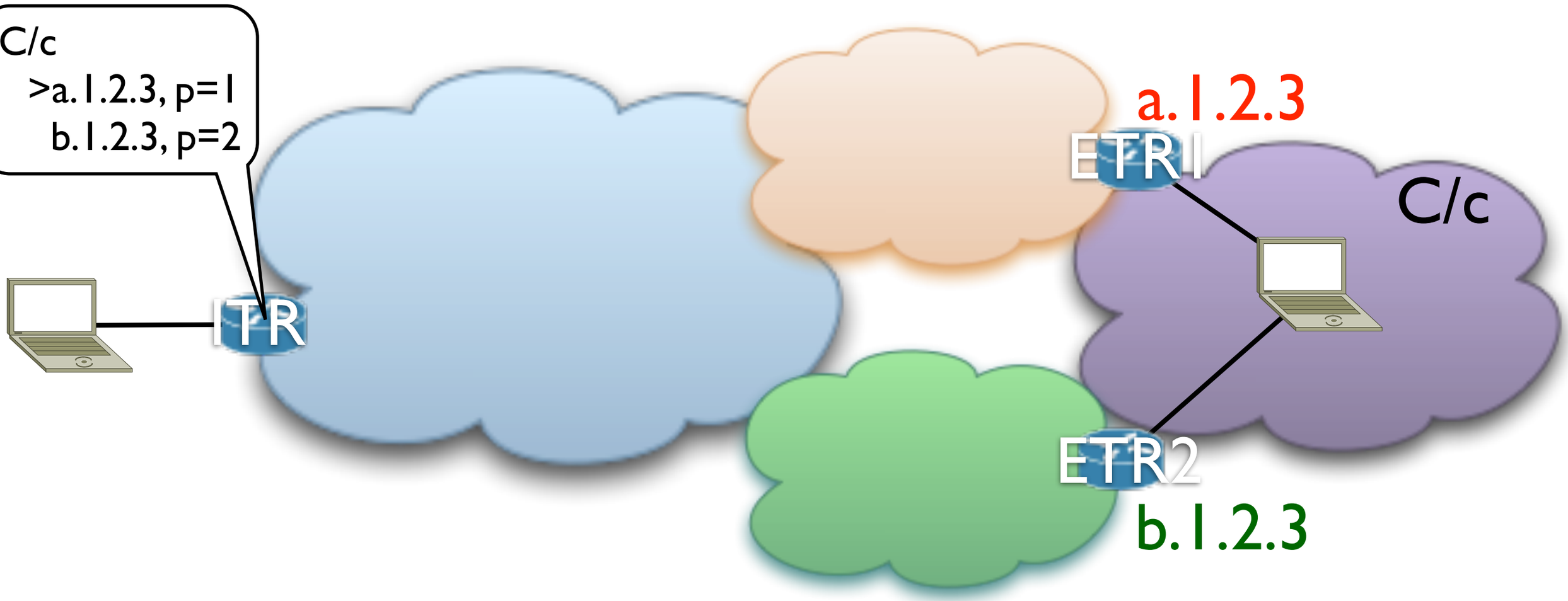
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



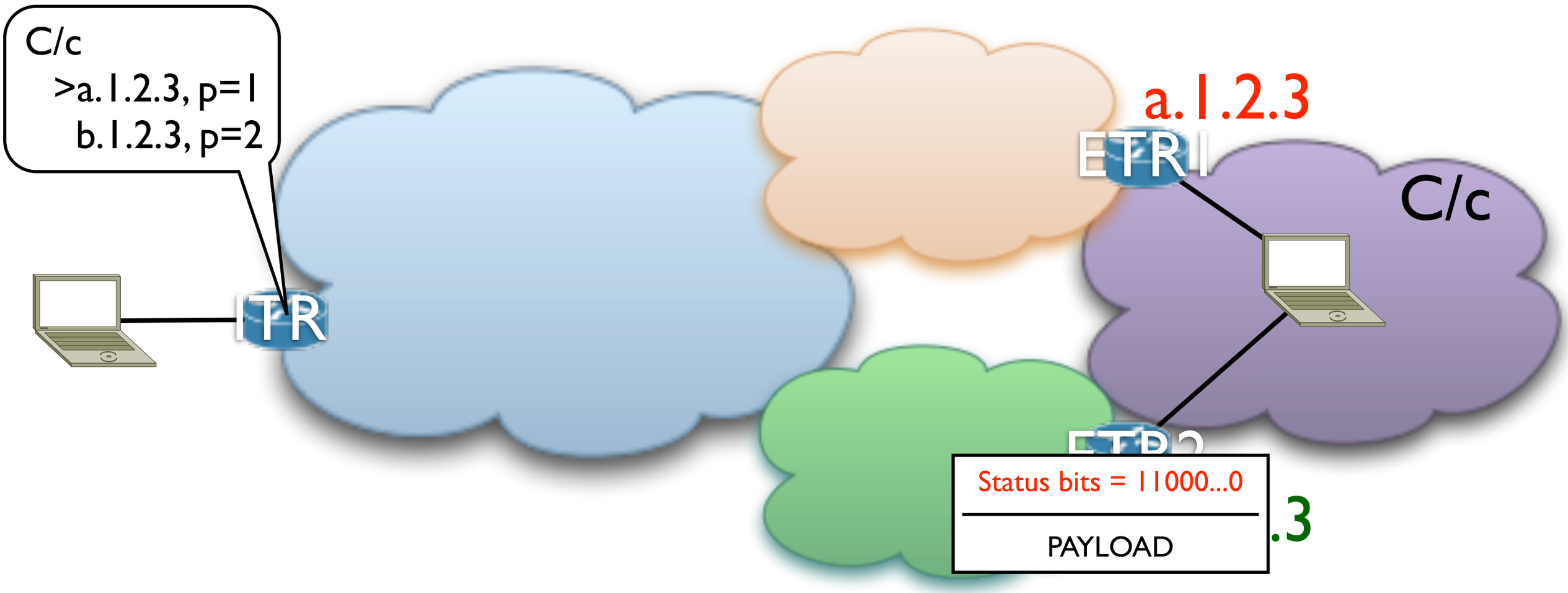
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



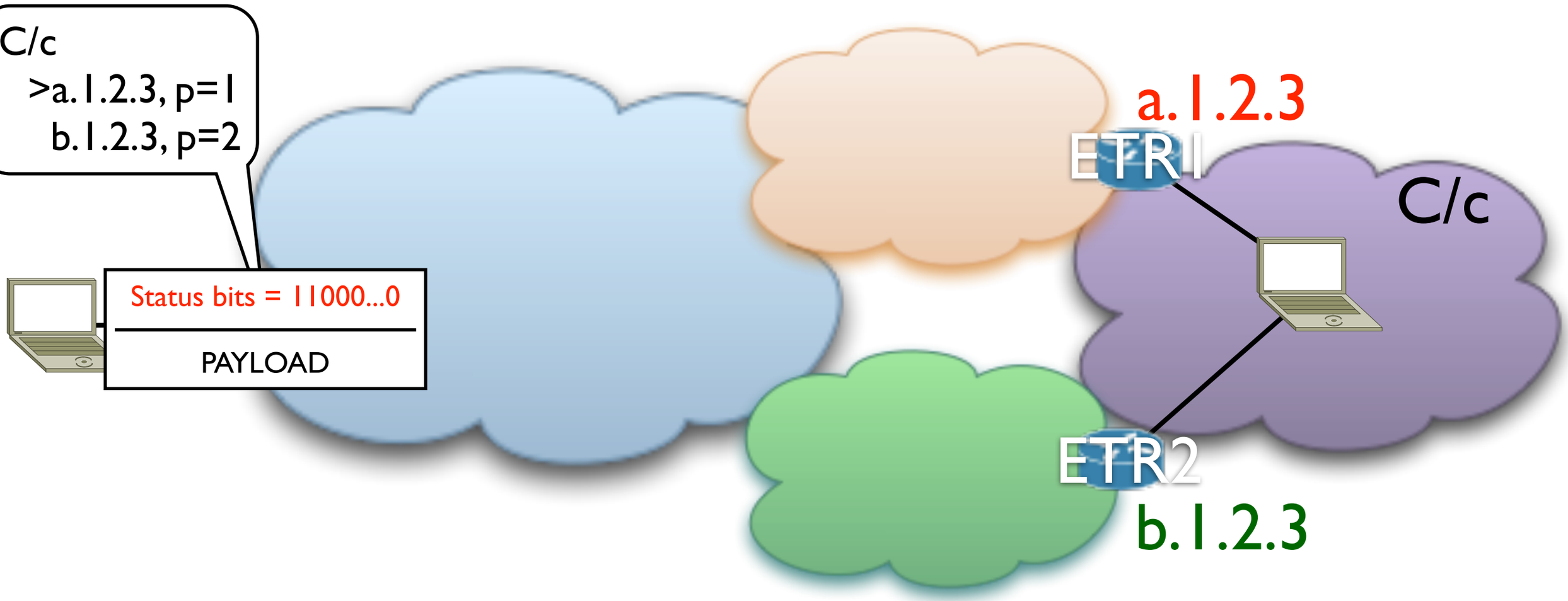
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



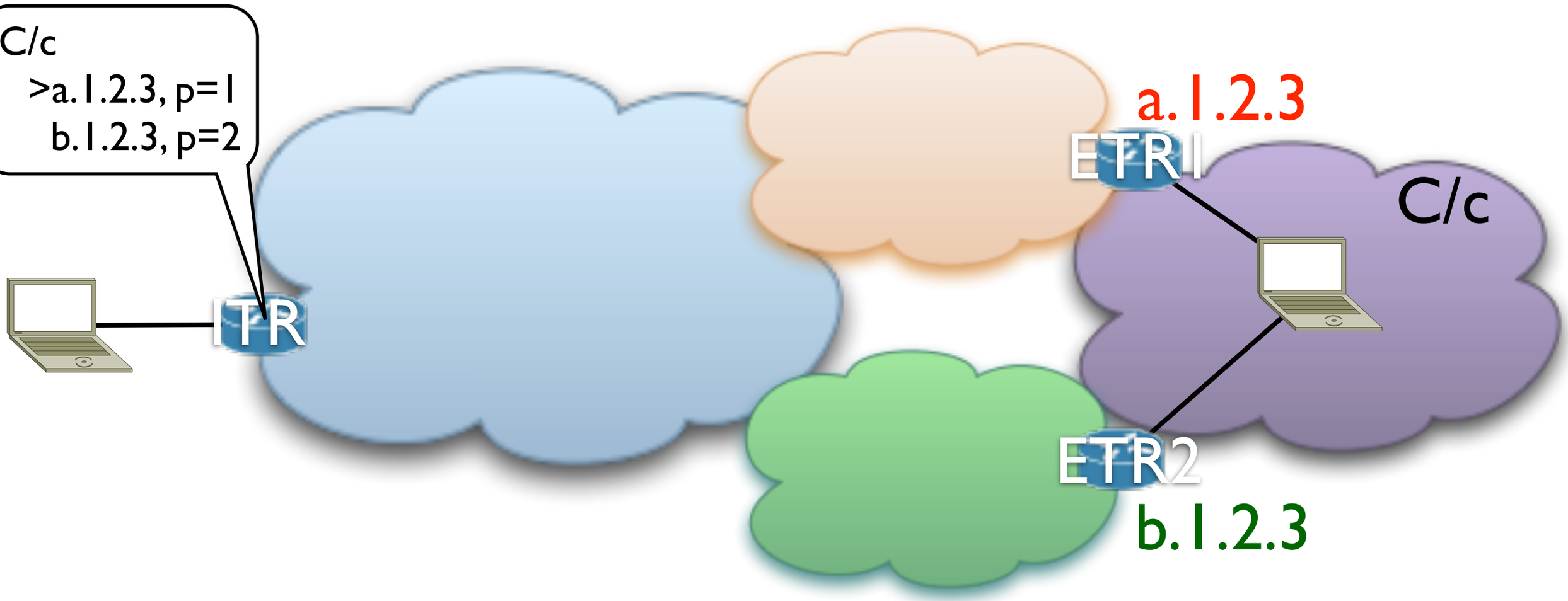
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



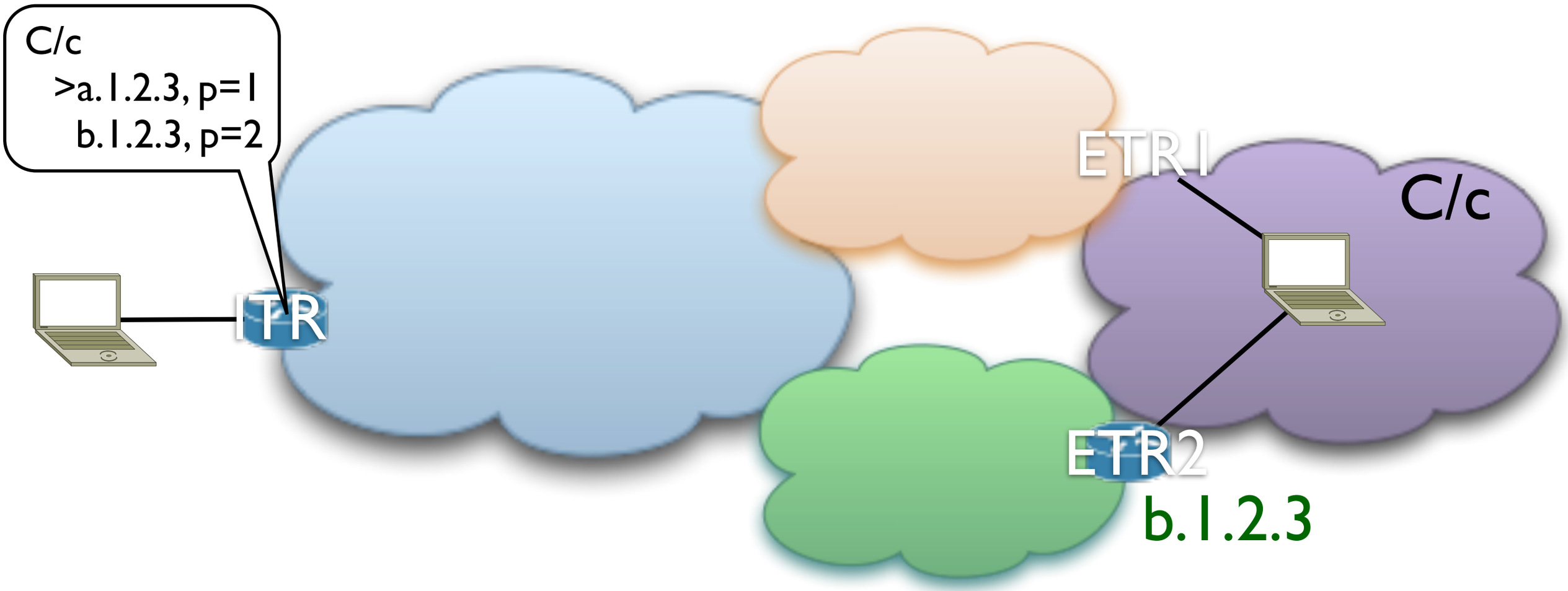
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



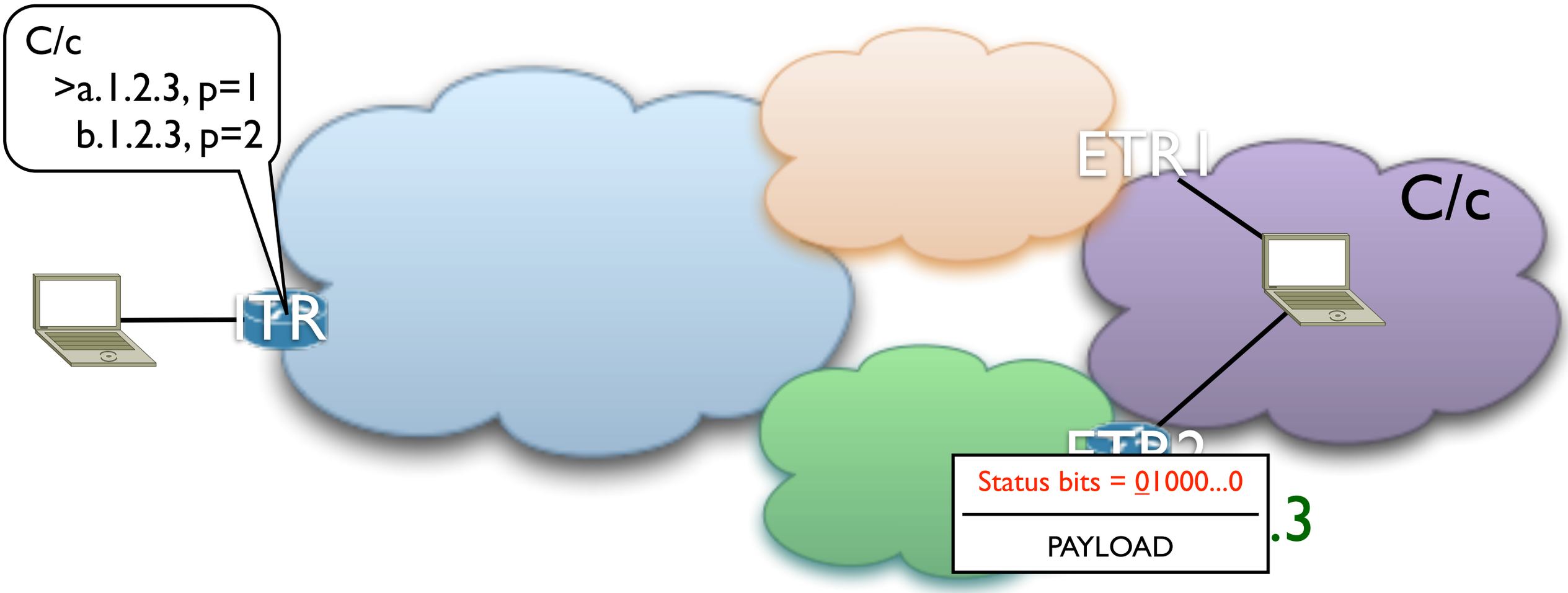
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



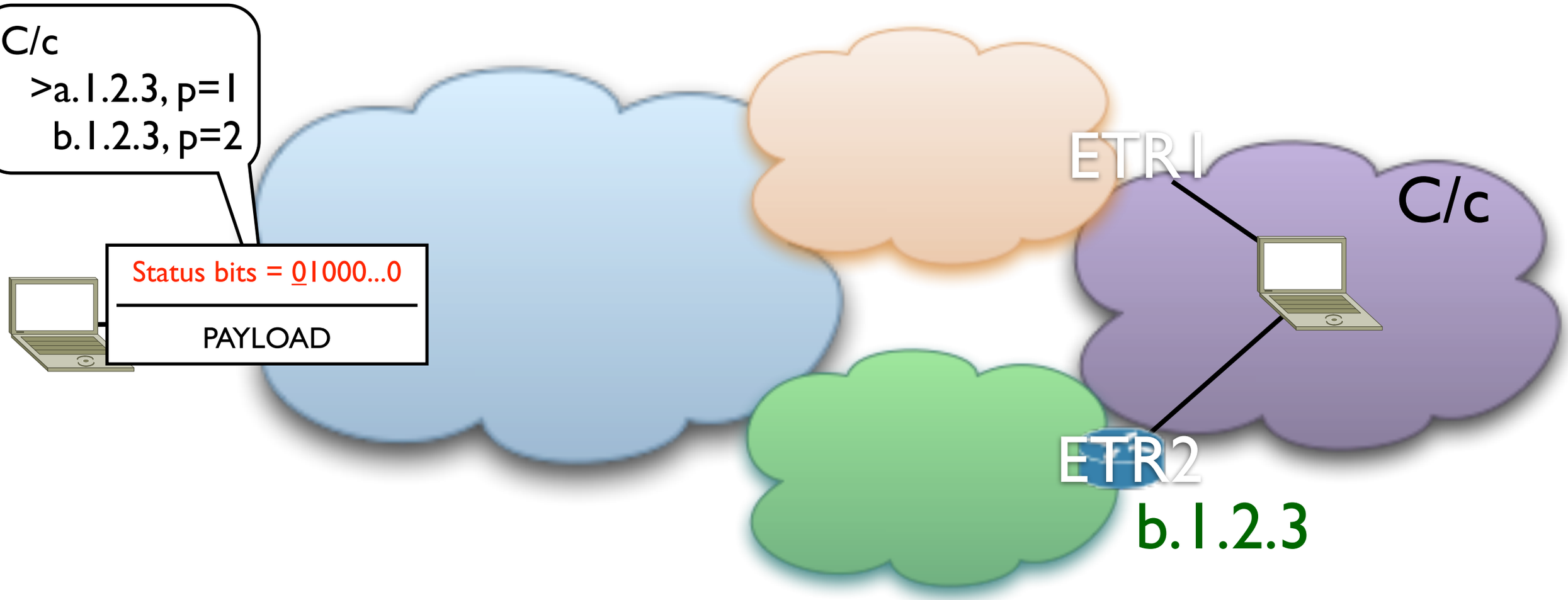
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



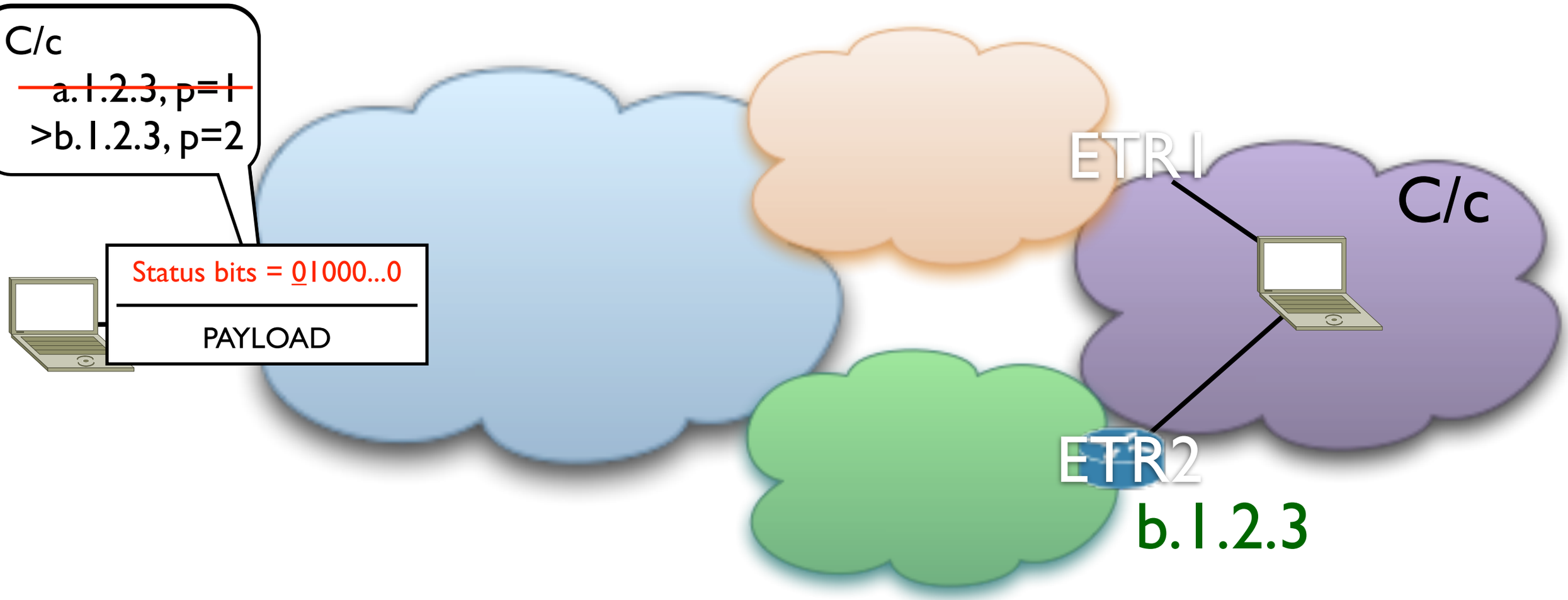
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



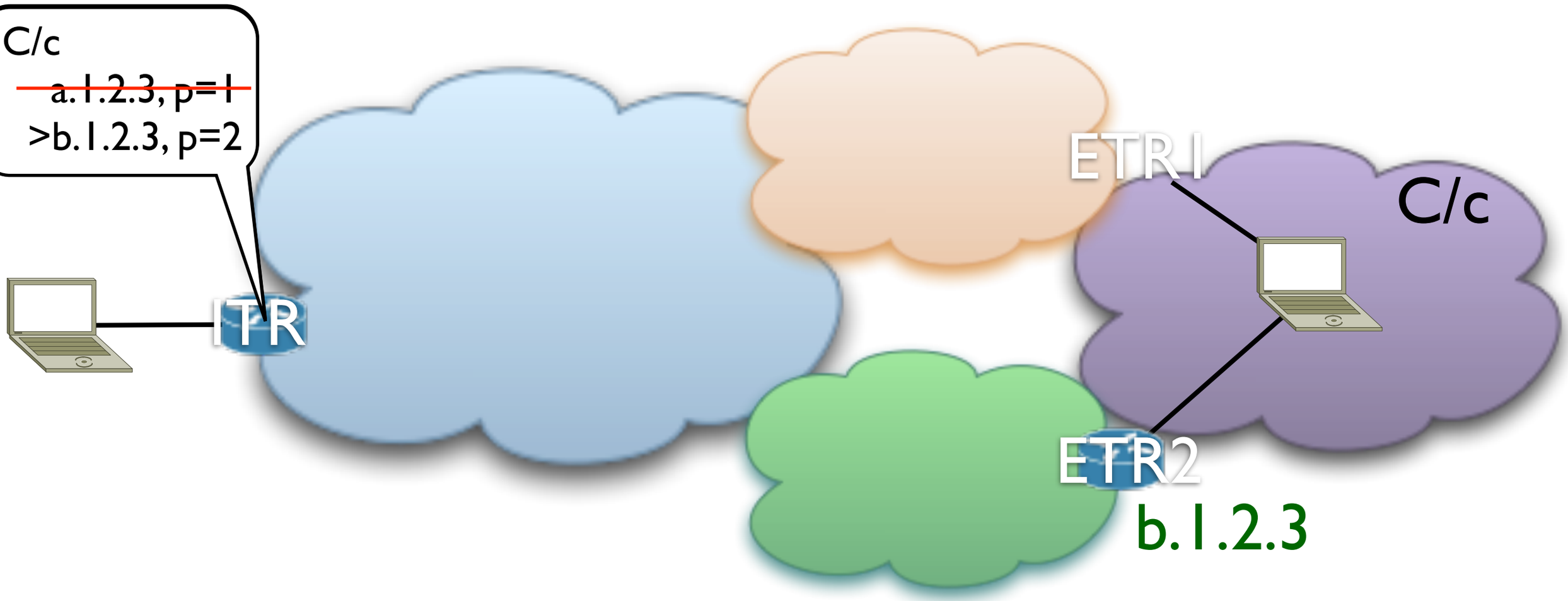
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



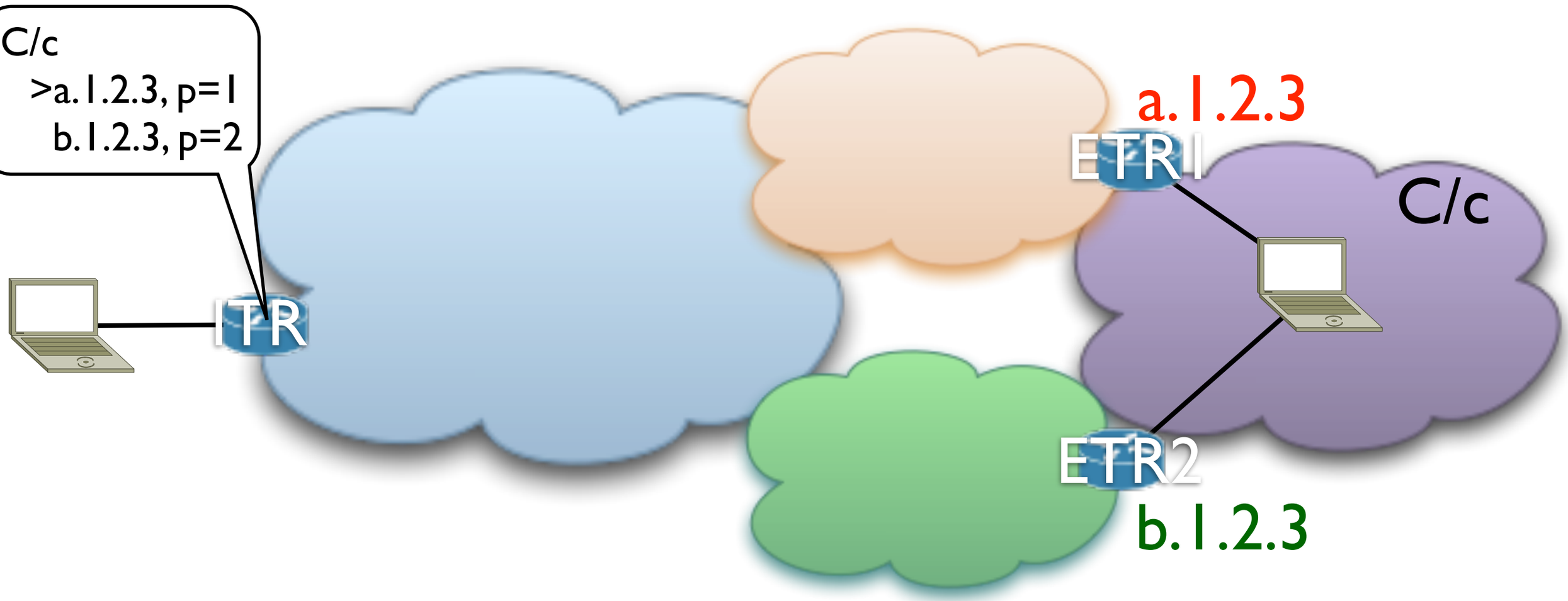
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the locator status bits



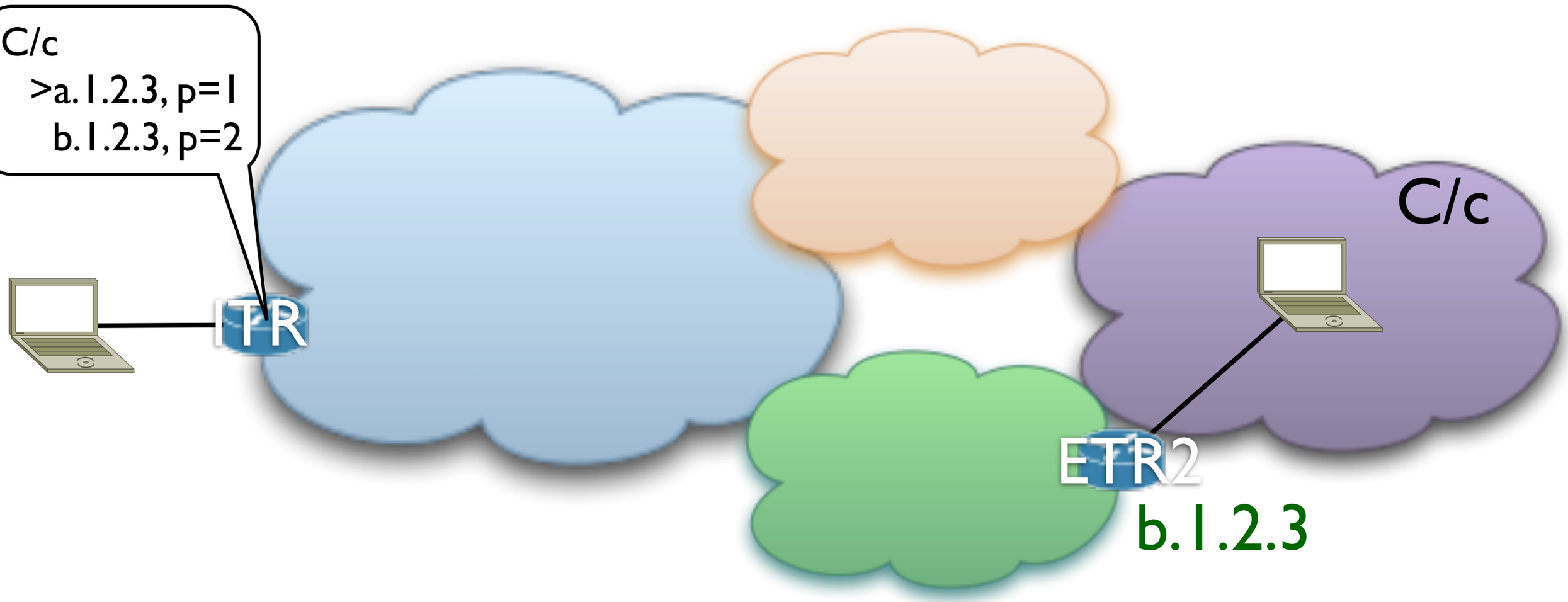
- ETR2 notices the failure and informs all ITRs to which it is sending LISP encapsulated packets by setting the reachability bit of ETR1 to 0

Solving the reachability problem with the SMR bit



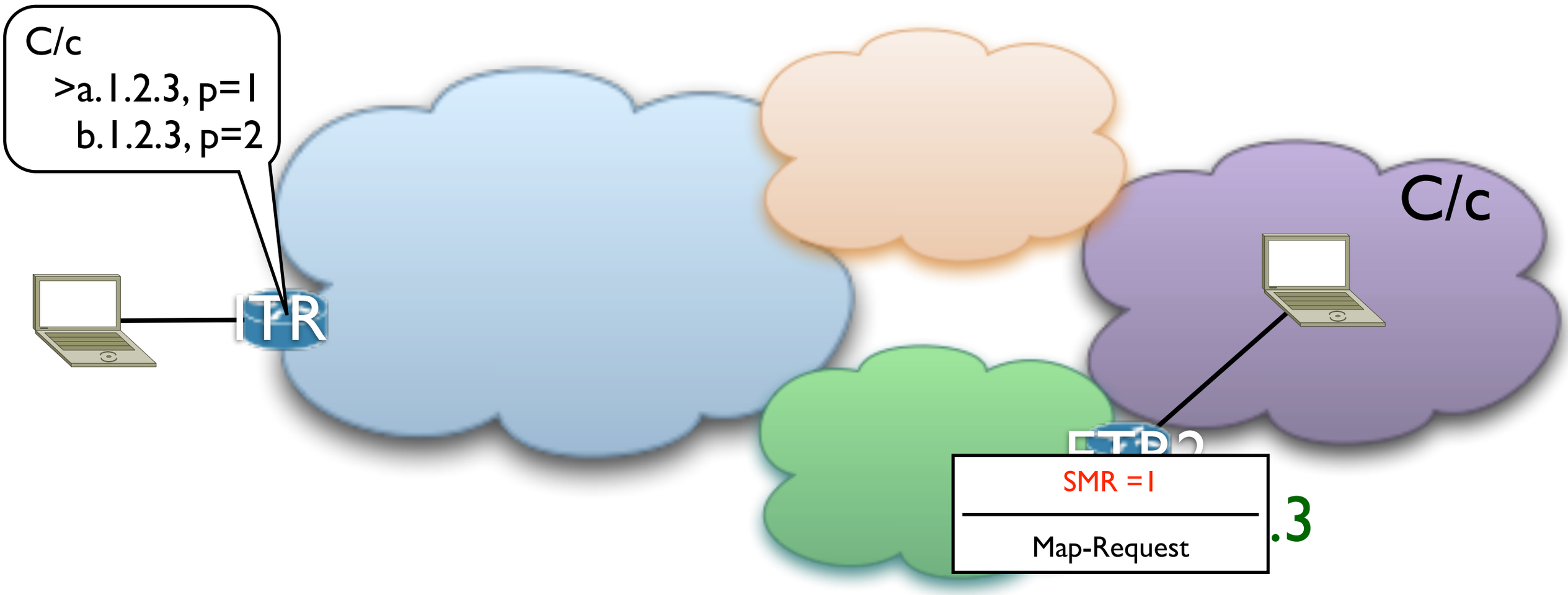
- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



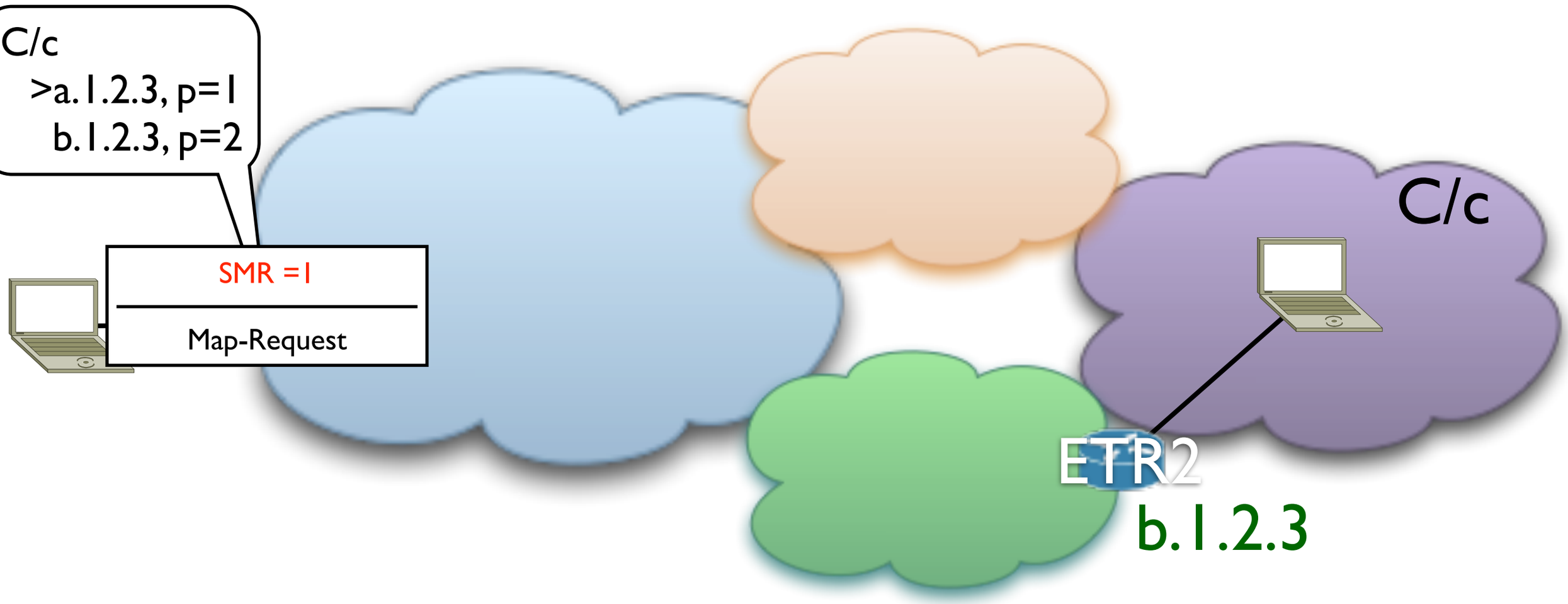
- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



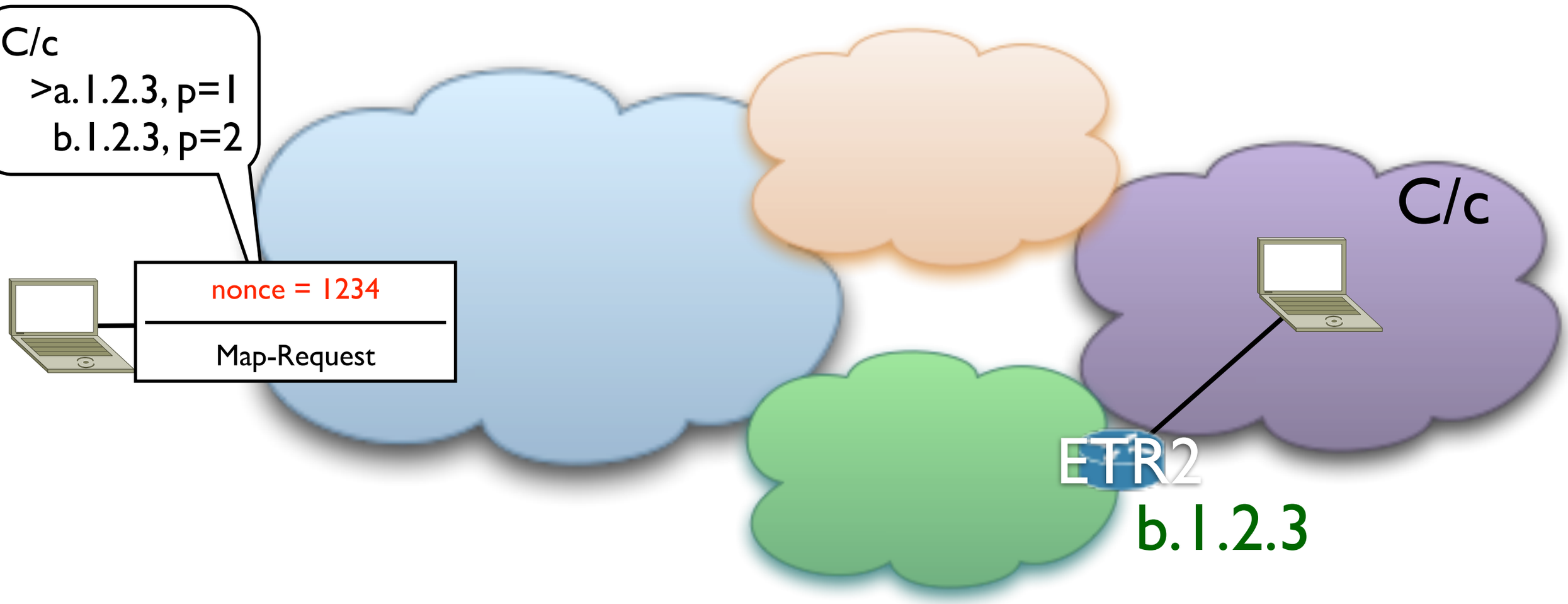
- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



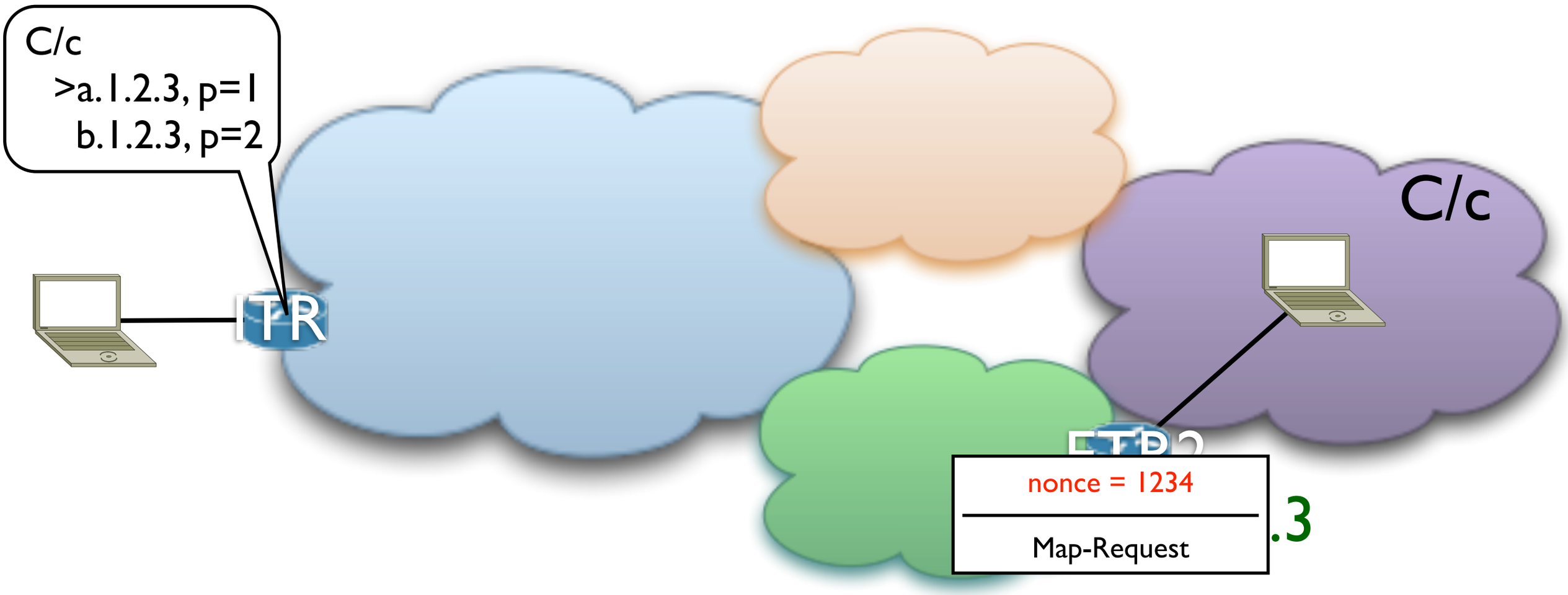
- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



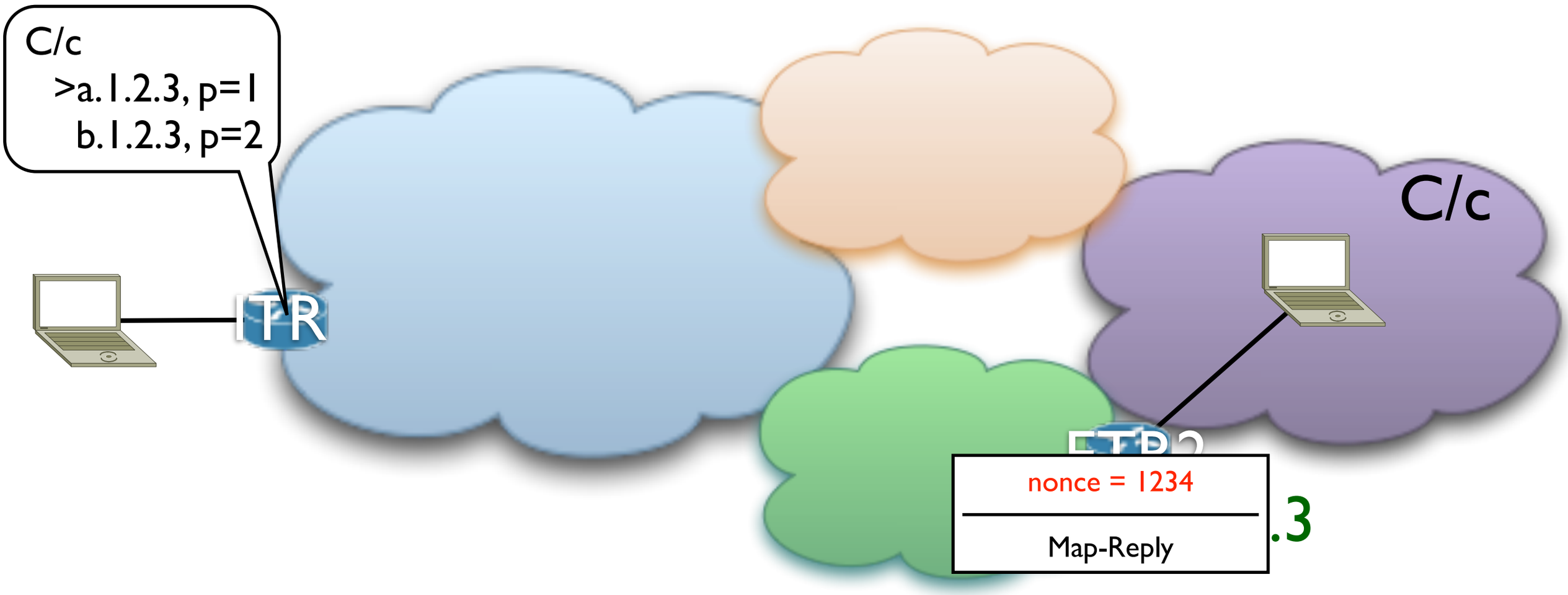
- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



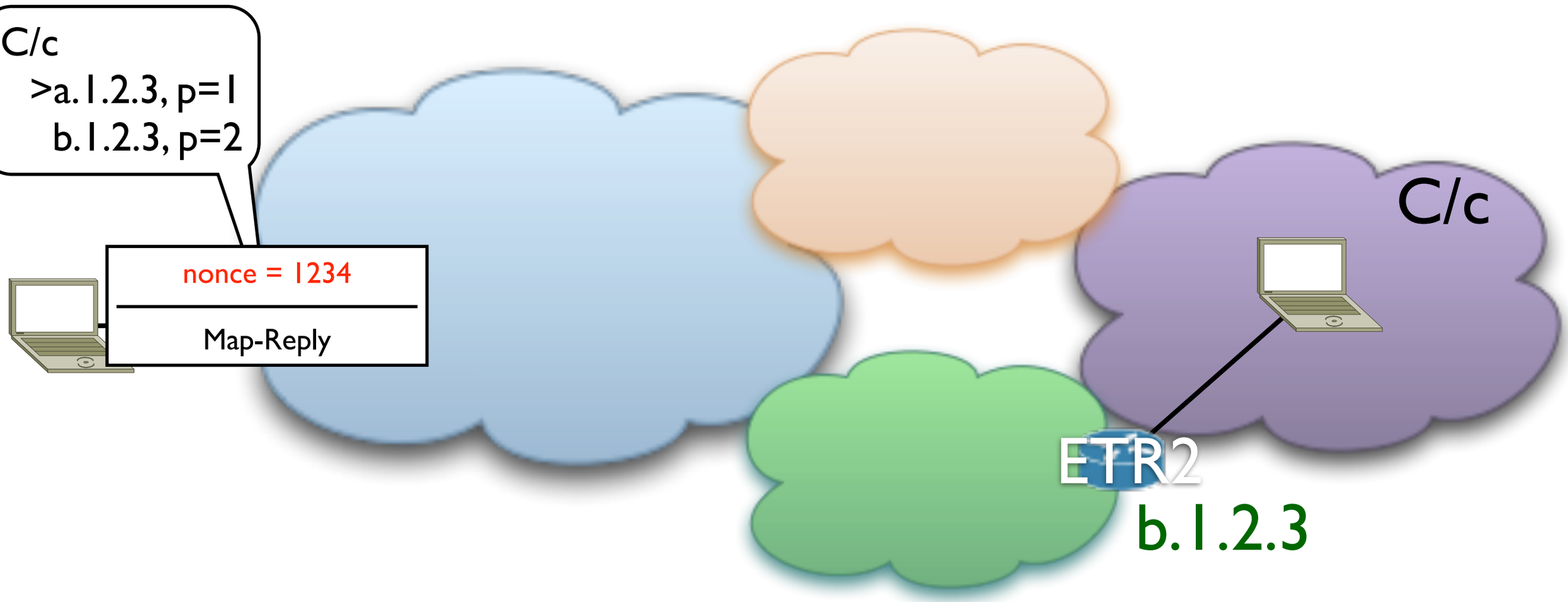
- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



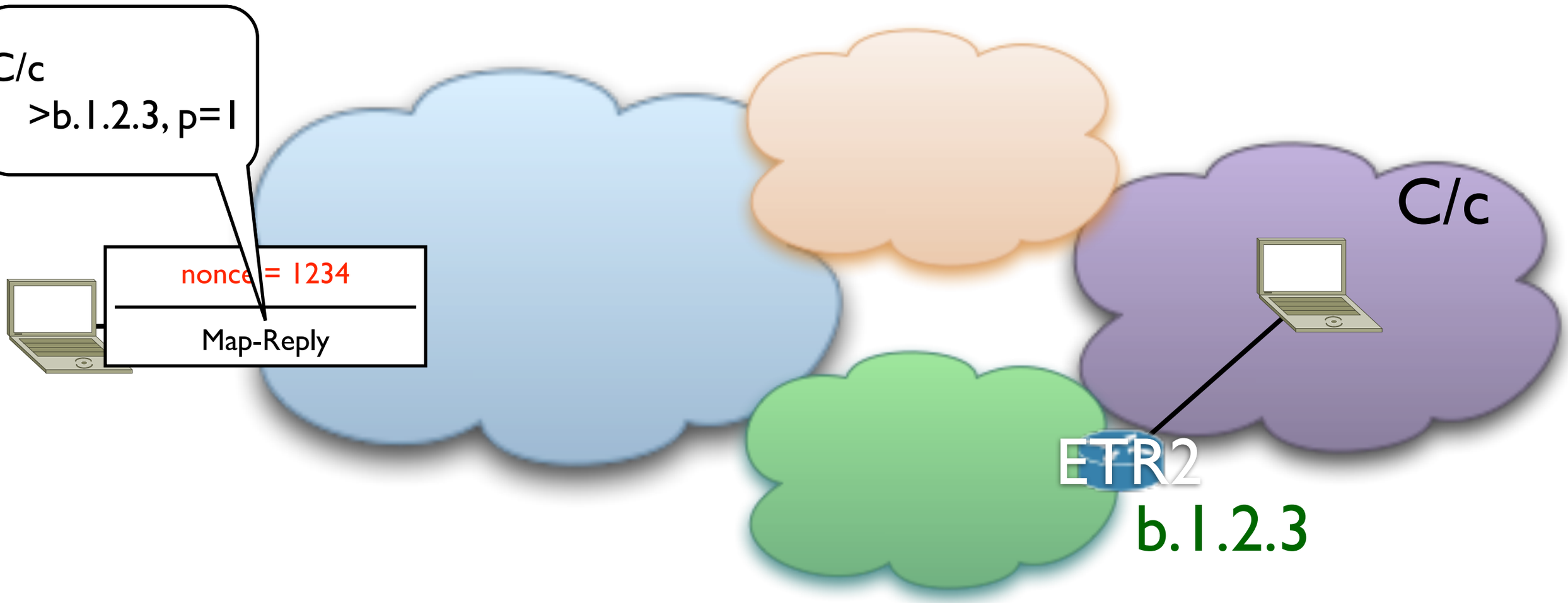
- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



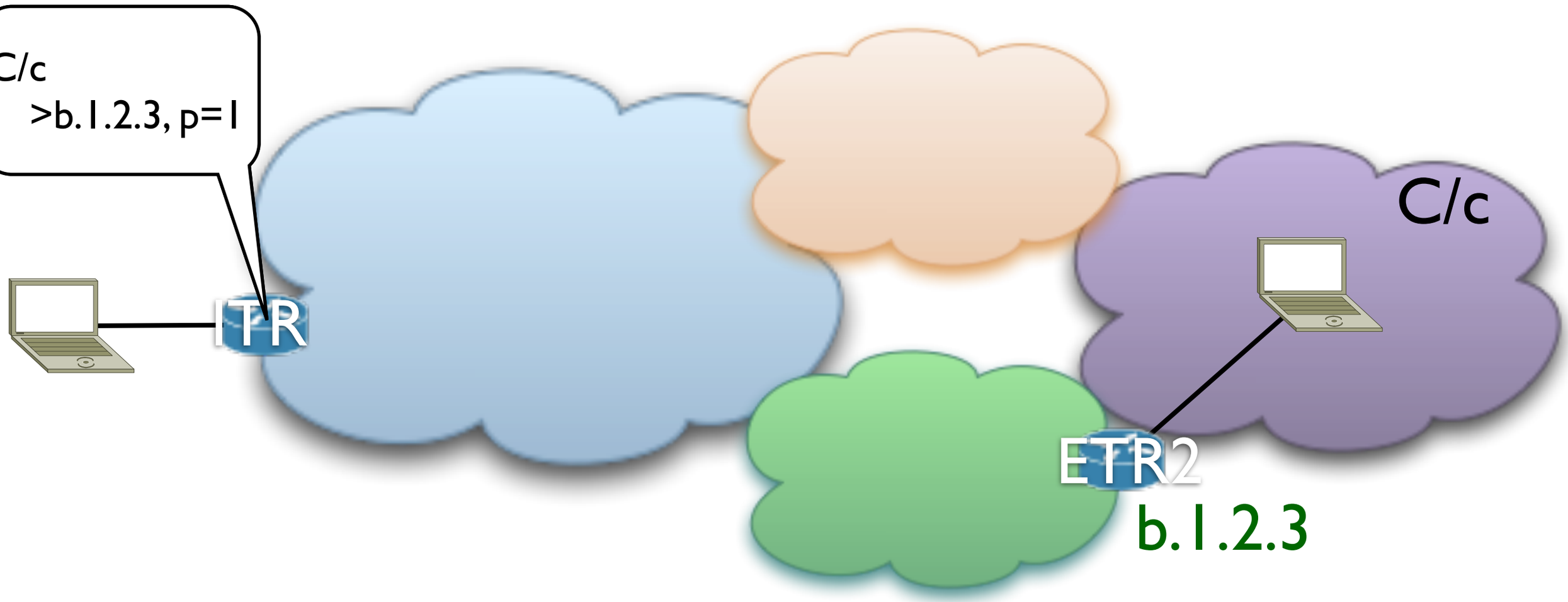
- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

Solving the reachability problem with the SMR bit



- ETR1 has been decommissioned and ETR2 wants that all the sites currently sending encapsulated data to itself update the mapping

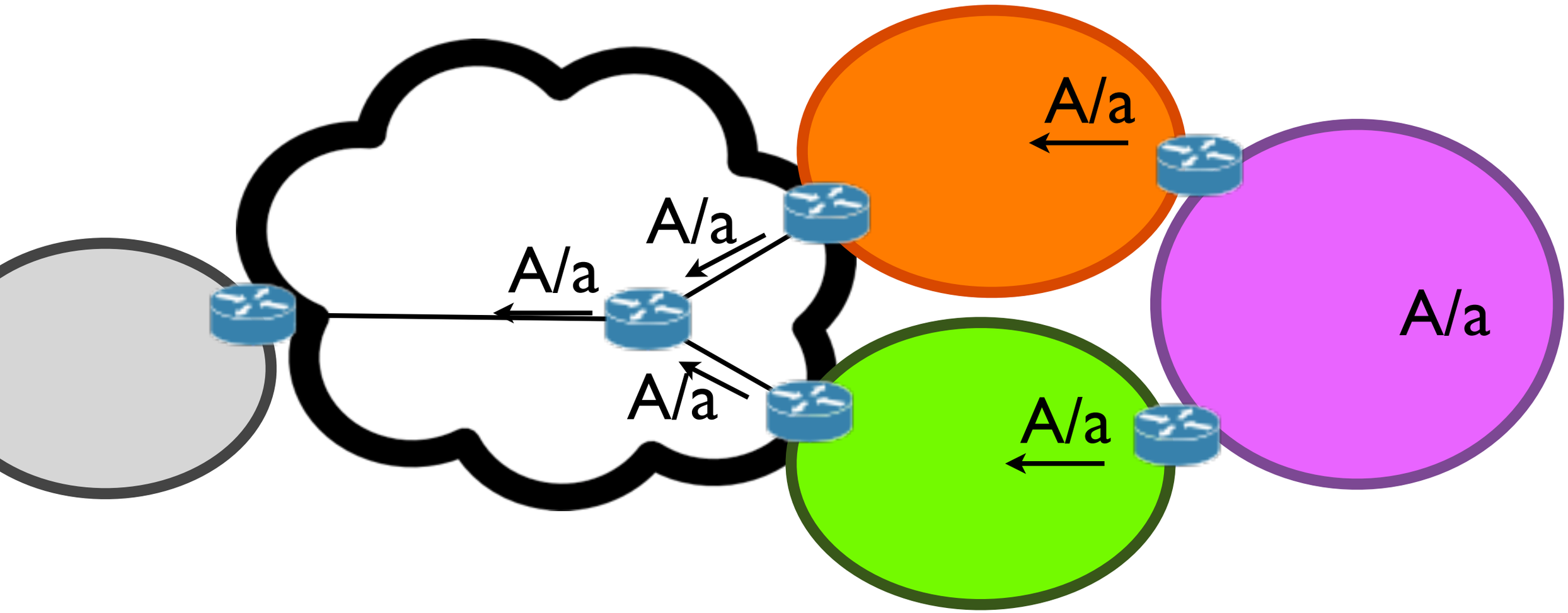
Solving the reachability problem with the Echo-Nonce Algorithm

- xTR *I* wants to know if the RLOC she uses to reach ETR *E* is reachable:
  - generate a nonce *n* when encap to *E*
  - set E=1
- Next time *E* sends a packet to *I*, she sets the nonce to *n*
- If *I* receives the nonce within a given time, she considers the RLOC reachable, otherwise *E* is considered unreachable

The reachability problem

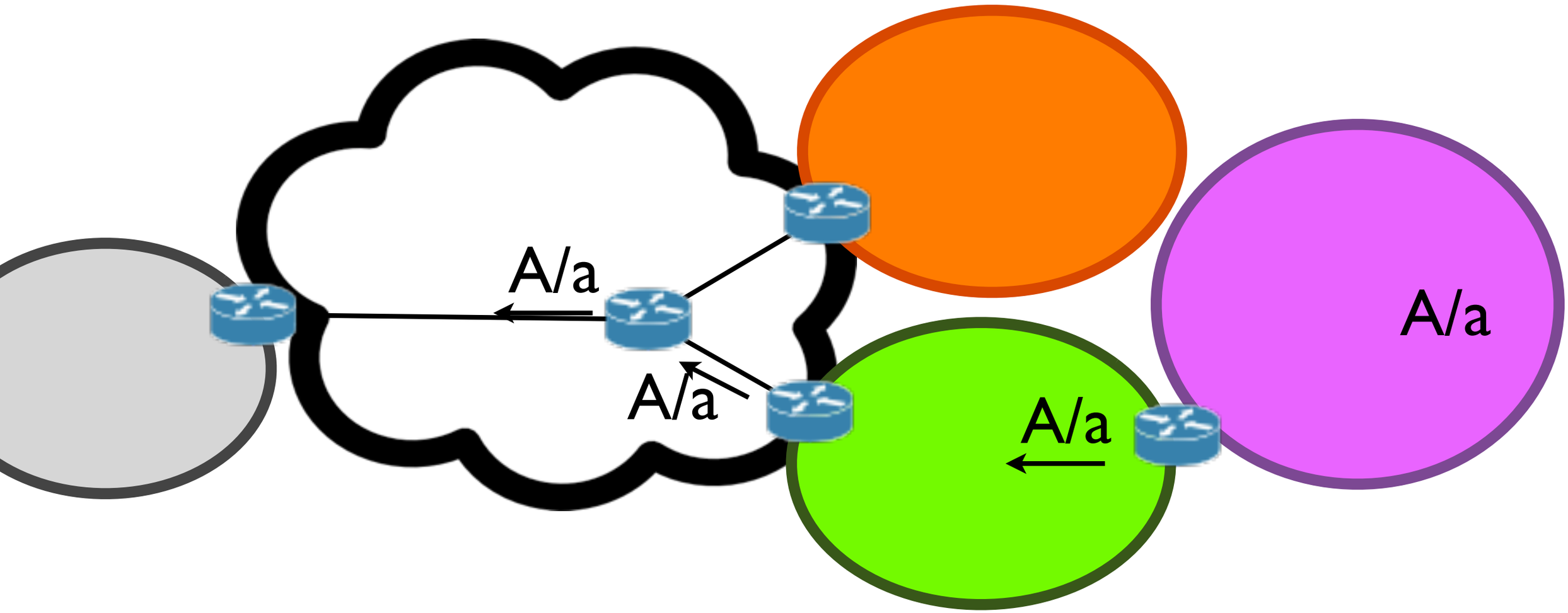
- Today, preserving the prefixes reachability is mainly performed locally
- In LISP, the legacy Internet is EID agnostic
-

The reachability problem in today's Internet



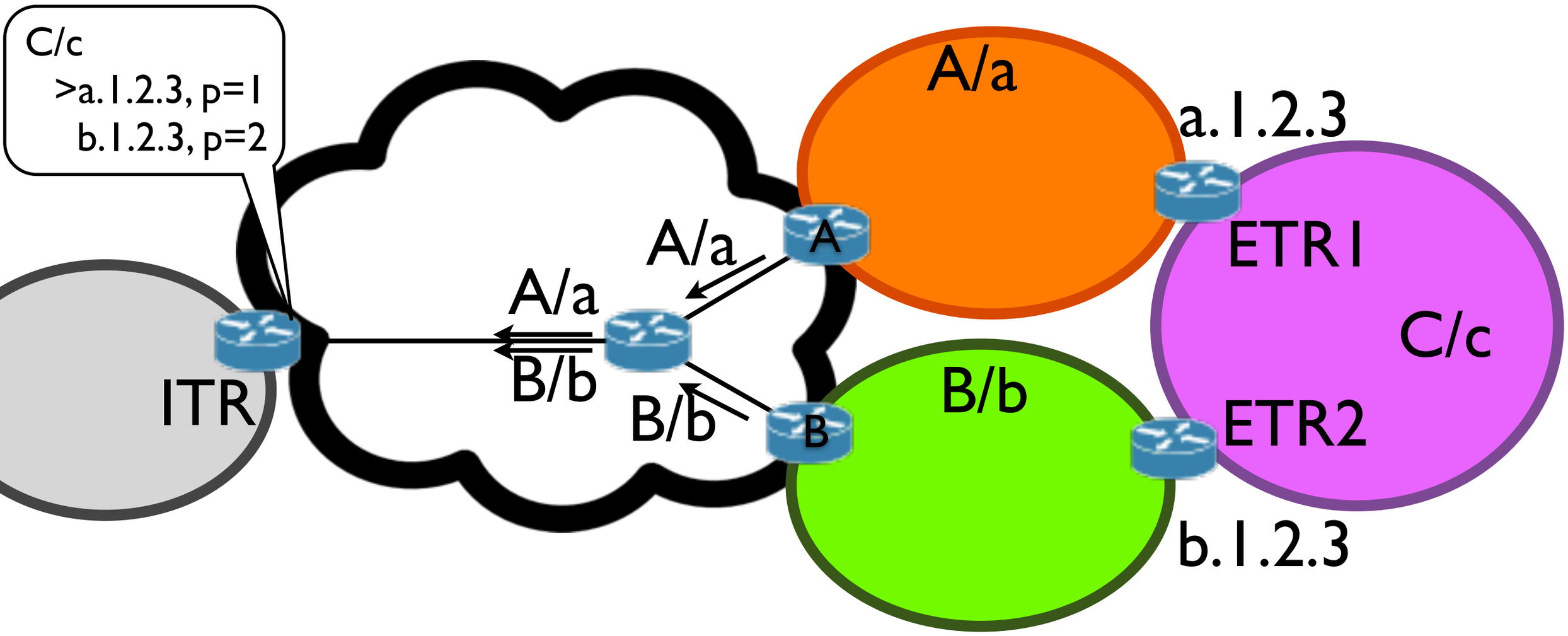
- In today's Internet, routing protocols converge after a link failure to ensure that multihomed prefixes such as *A/a* remain reachable

The reachability problem in today's Internet



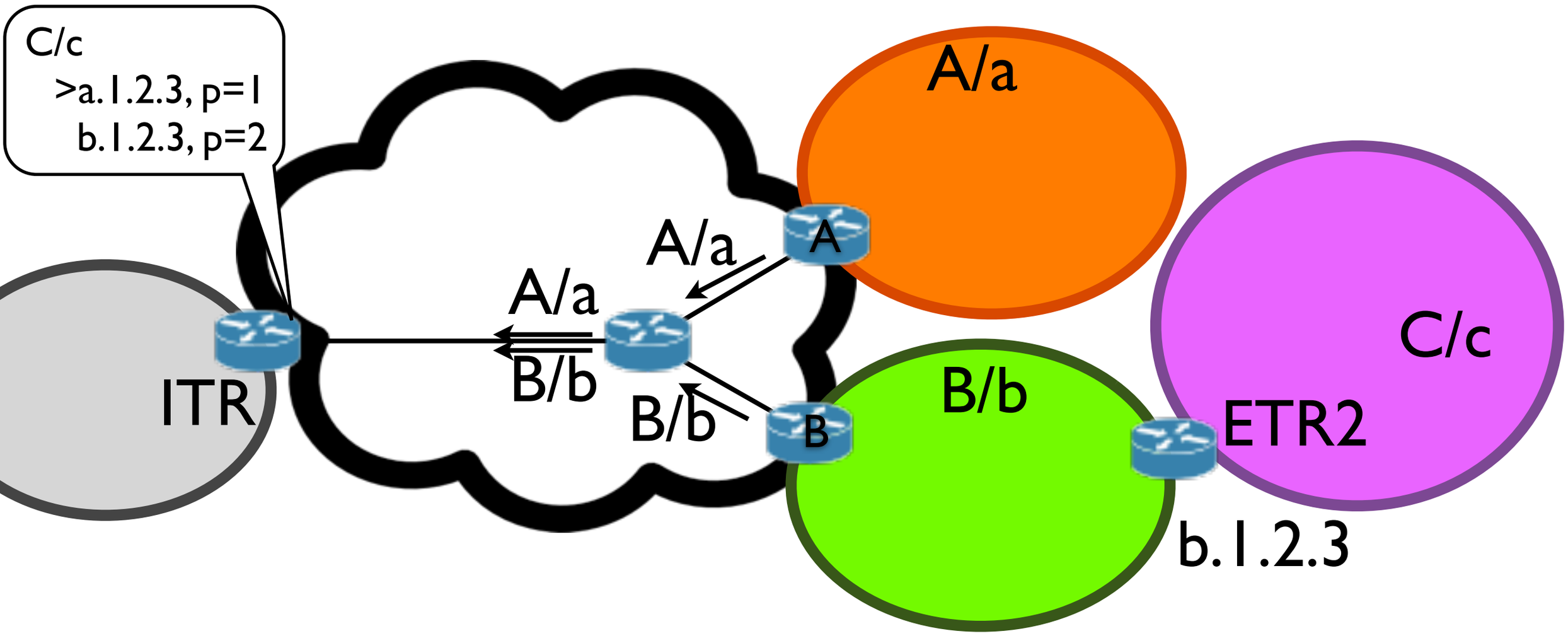
- In today's Internet, routing protocols converge after a link failure to ensure that multihomed prefixes such as *A/a* remain reachable

The reachability problem in a LISP-based Internet



- Upon failure of ETR1, A continues to advertise A/a via BGP
- How can ITR notice that ETR1 failed and that ETR2 should be used instead?

The reachability problem in a LISP-based Internet



- Upon failure of ETR1, A continues to advertise A/a via BGP
- How can ITR notice that ETR1 failed and that ETR2 should be used instead?

Solving the reachability problem with RLOC Probing

- Periodically probe the RLOCs to check their reachability

Host vs Network-based Loc/ID split

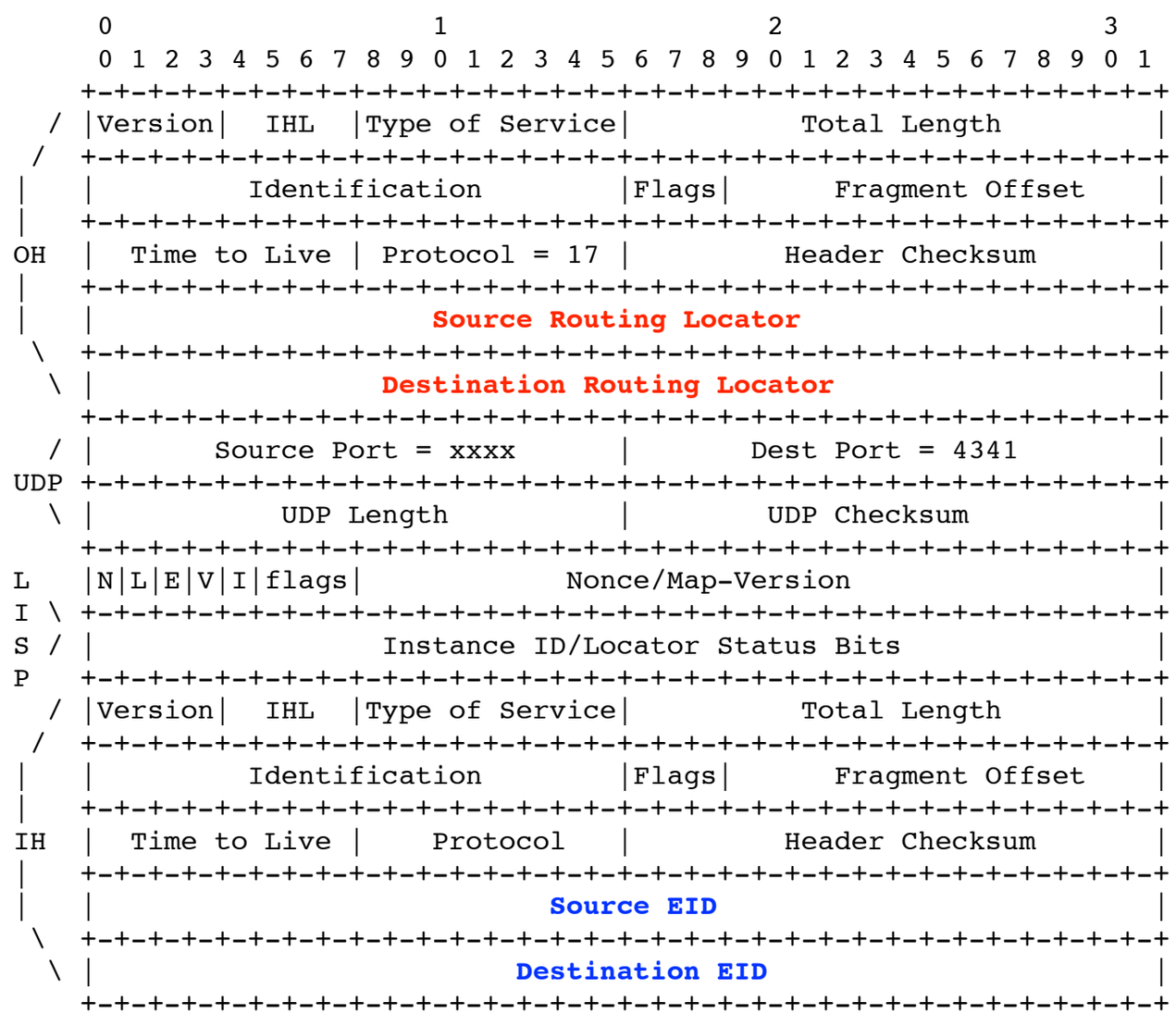
- Both can coexist
- At home, connected directly to the wall
  - Let my ISP do the stuff for me
- In the street, calling with Skype over WIFI&3G
  - Prefer WIFI to 3G when possible

Header details

(IP(UDP(LISP(IP))))

Encapsulating

Encapsulated



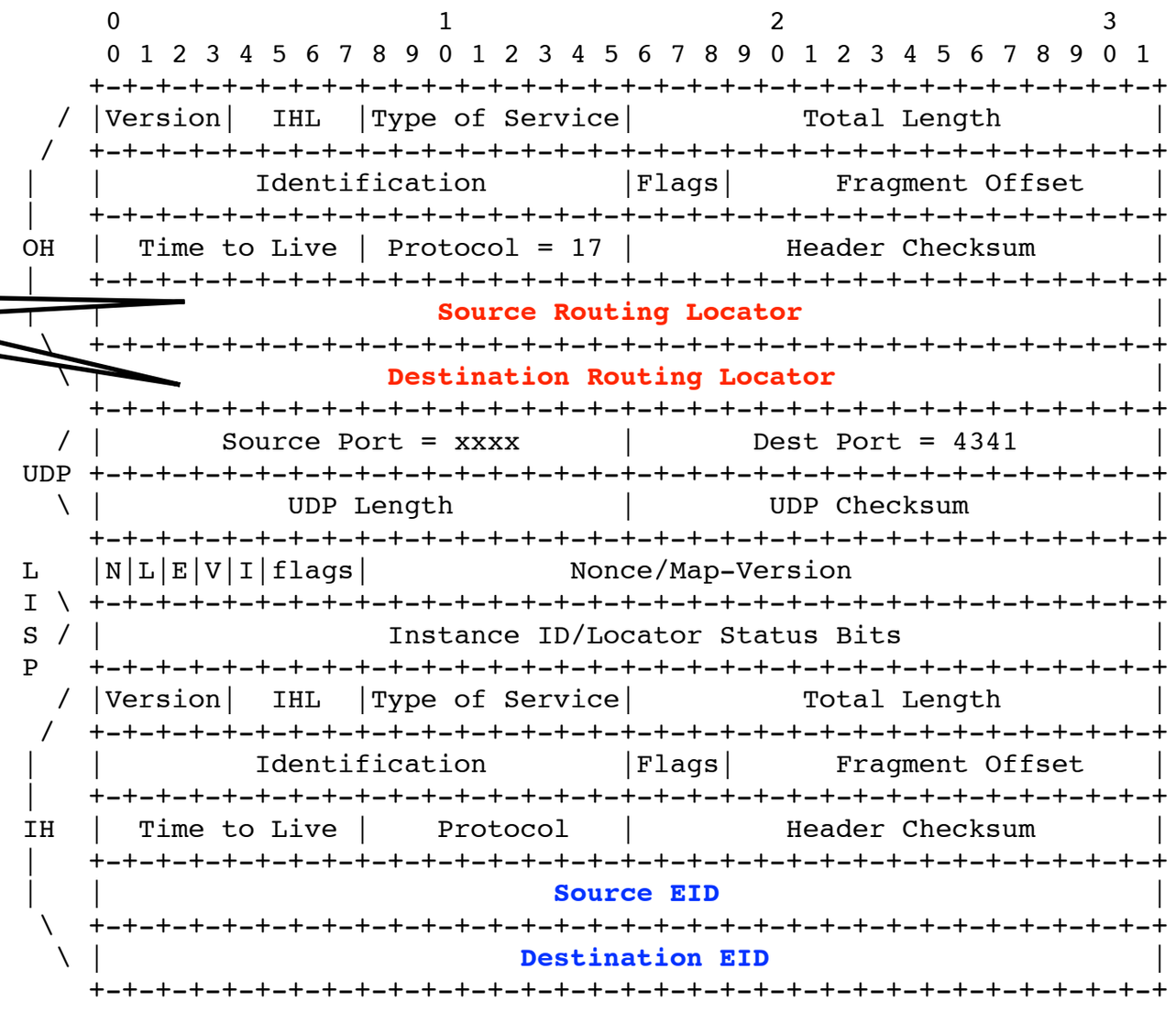
Header details

(IP(UDP(LISP(IP))))

Encapsulating

Encapsulated

Source/Destination Locators



Header details

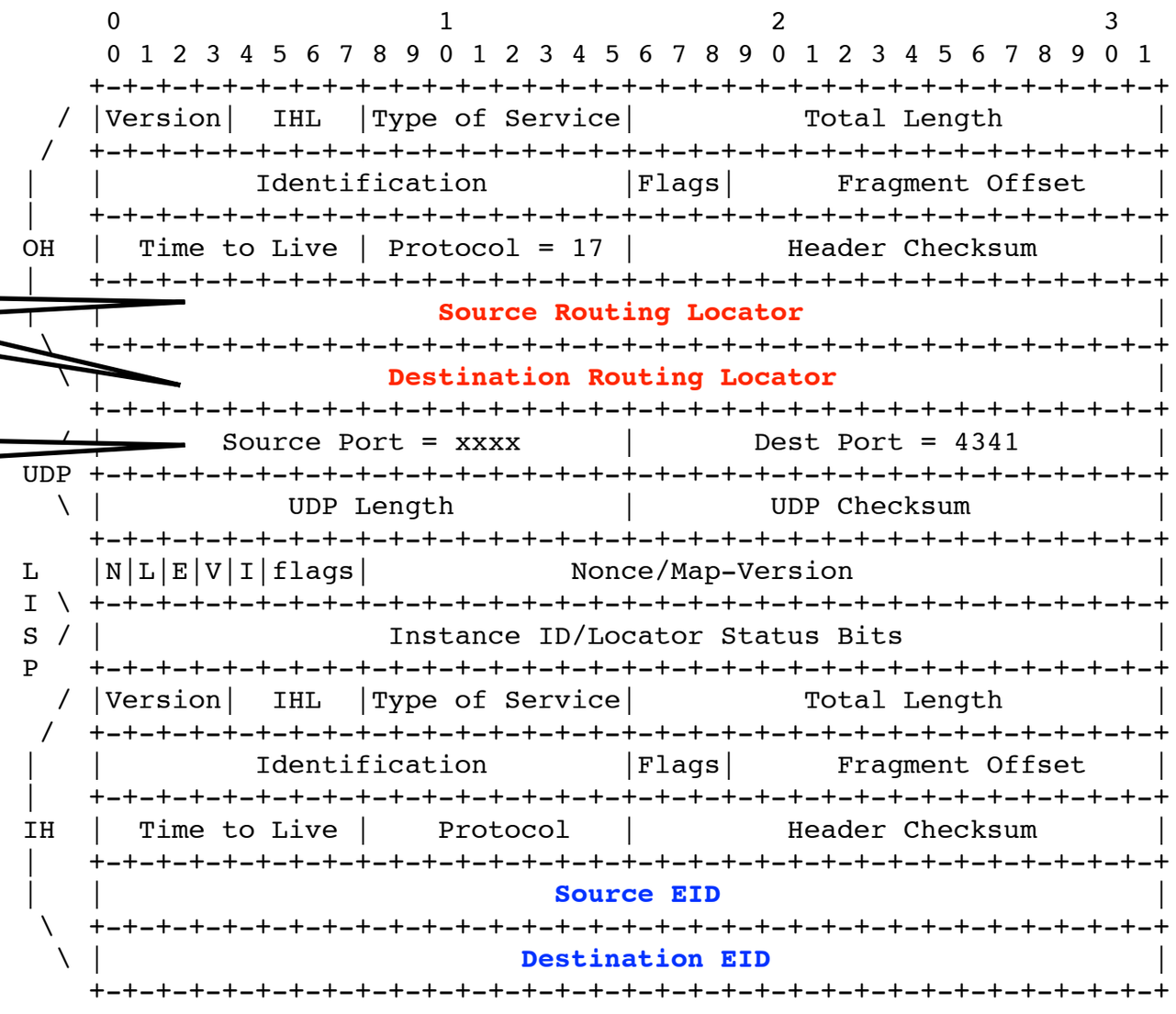
(IP(UDP(LISP(IP))))

Encapsulating

Encapsulated

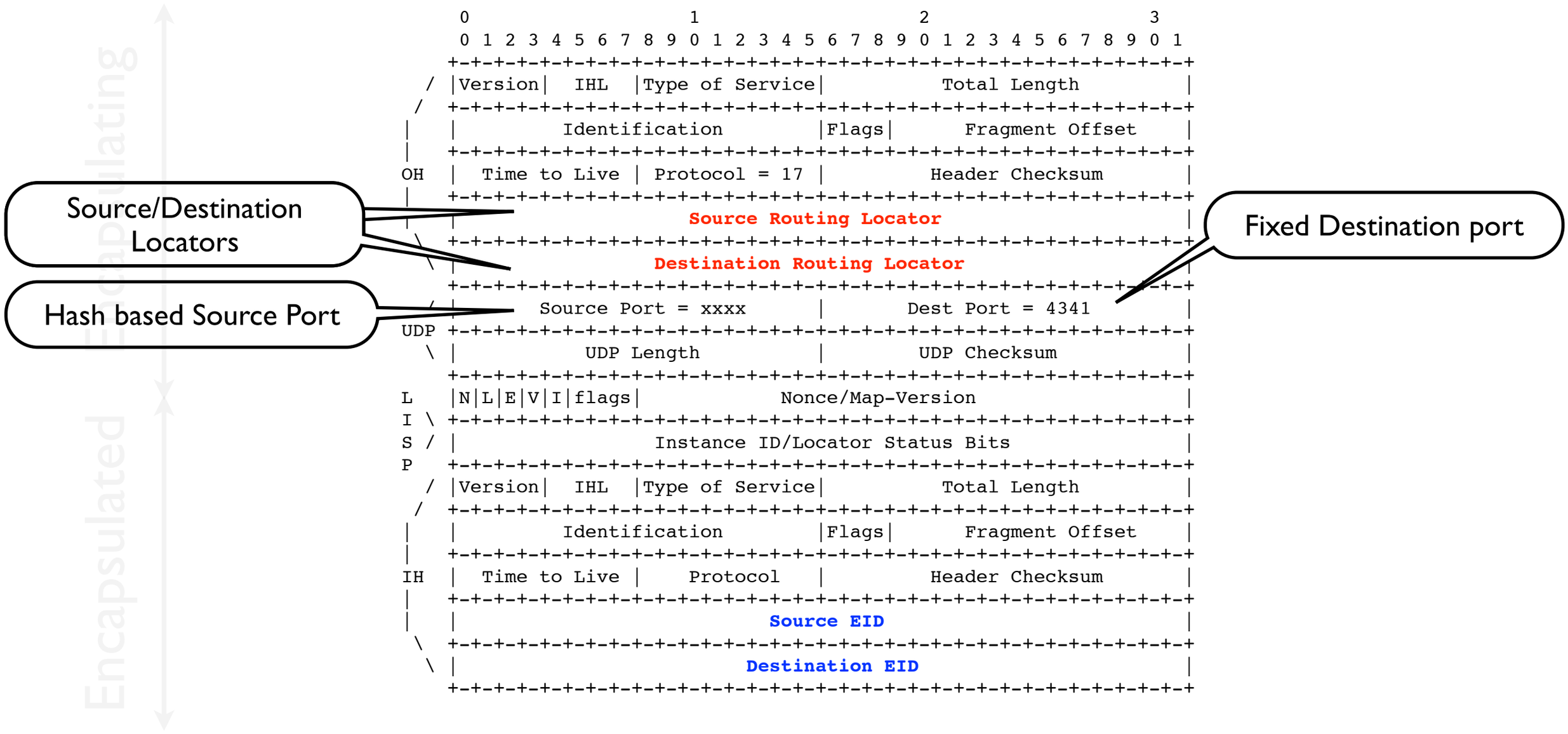
Source/Destination Locators

Hash based Source Port



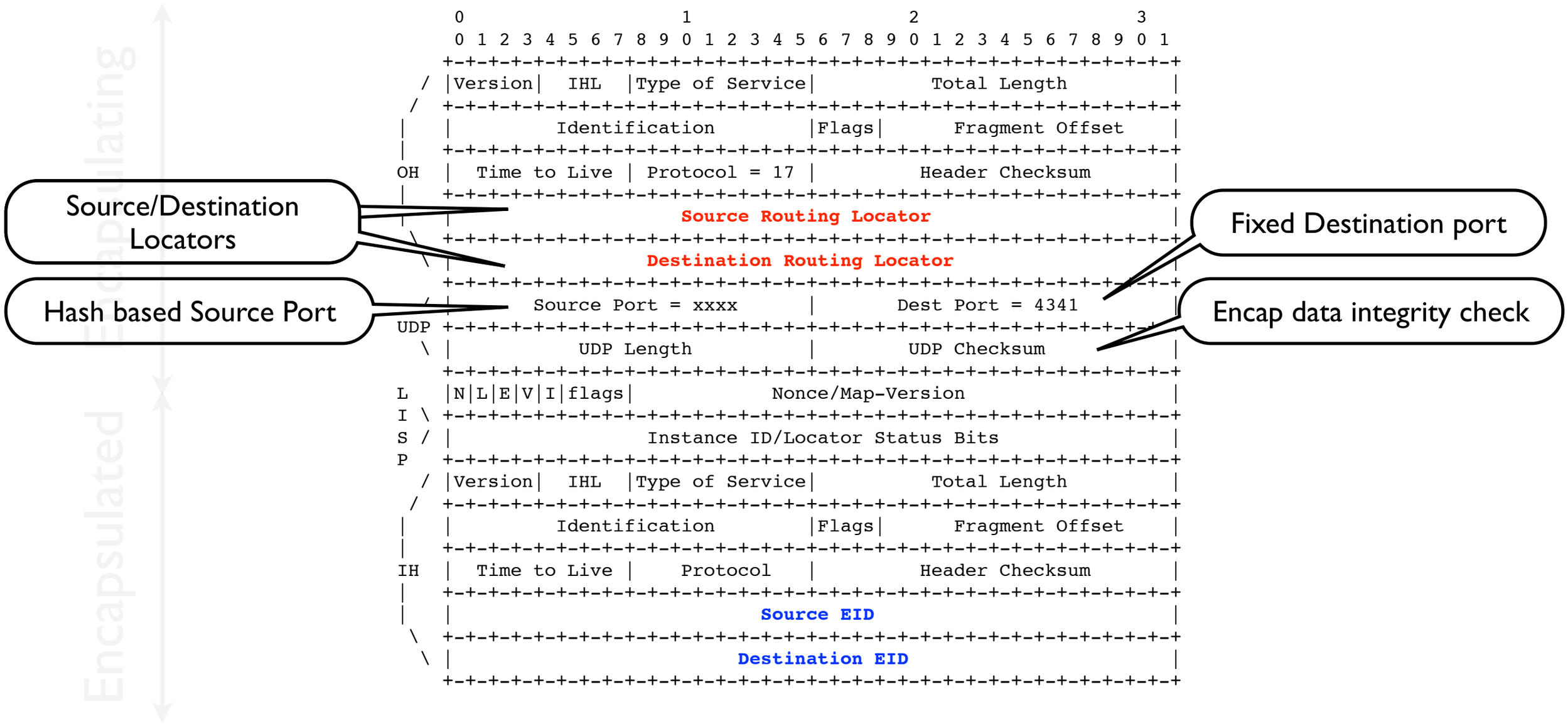
Header details

(IP(UDP(LISP(IP))))



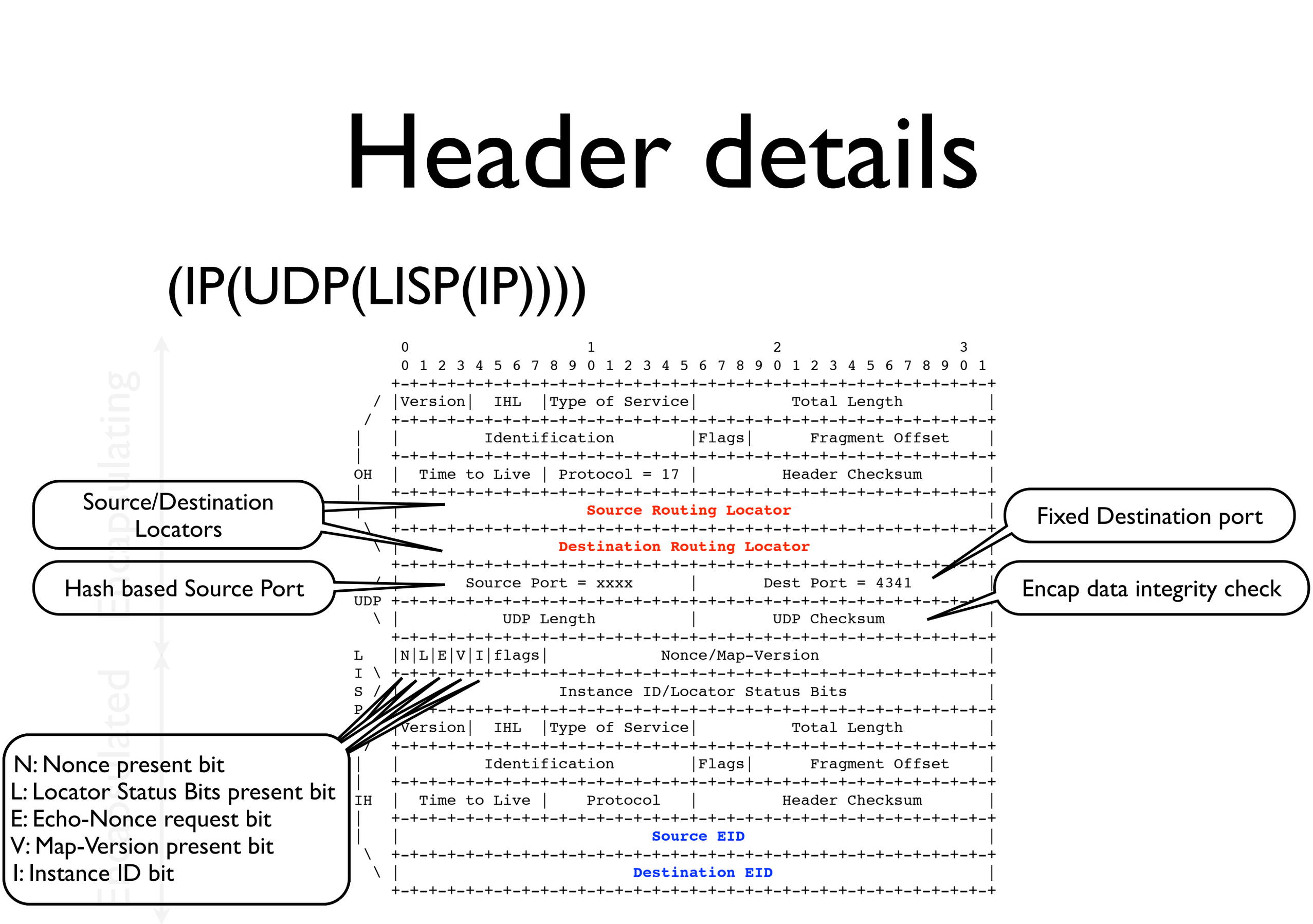
Header details

(IP(UDP(LISP(IP))))



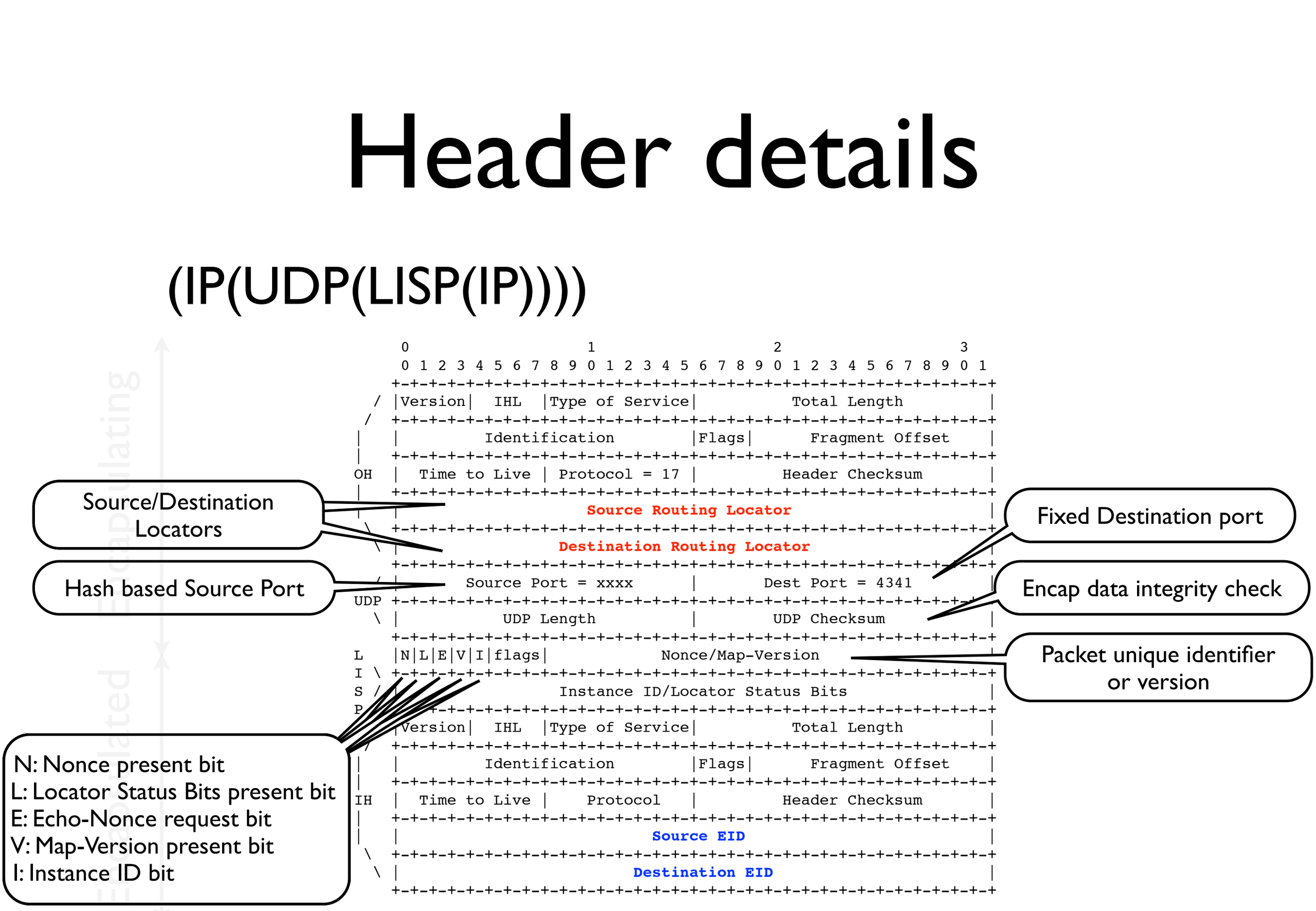
Header details

(IP(UDP(LISP(IP))))



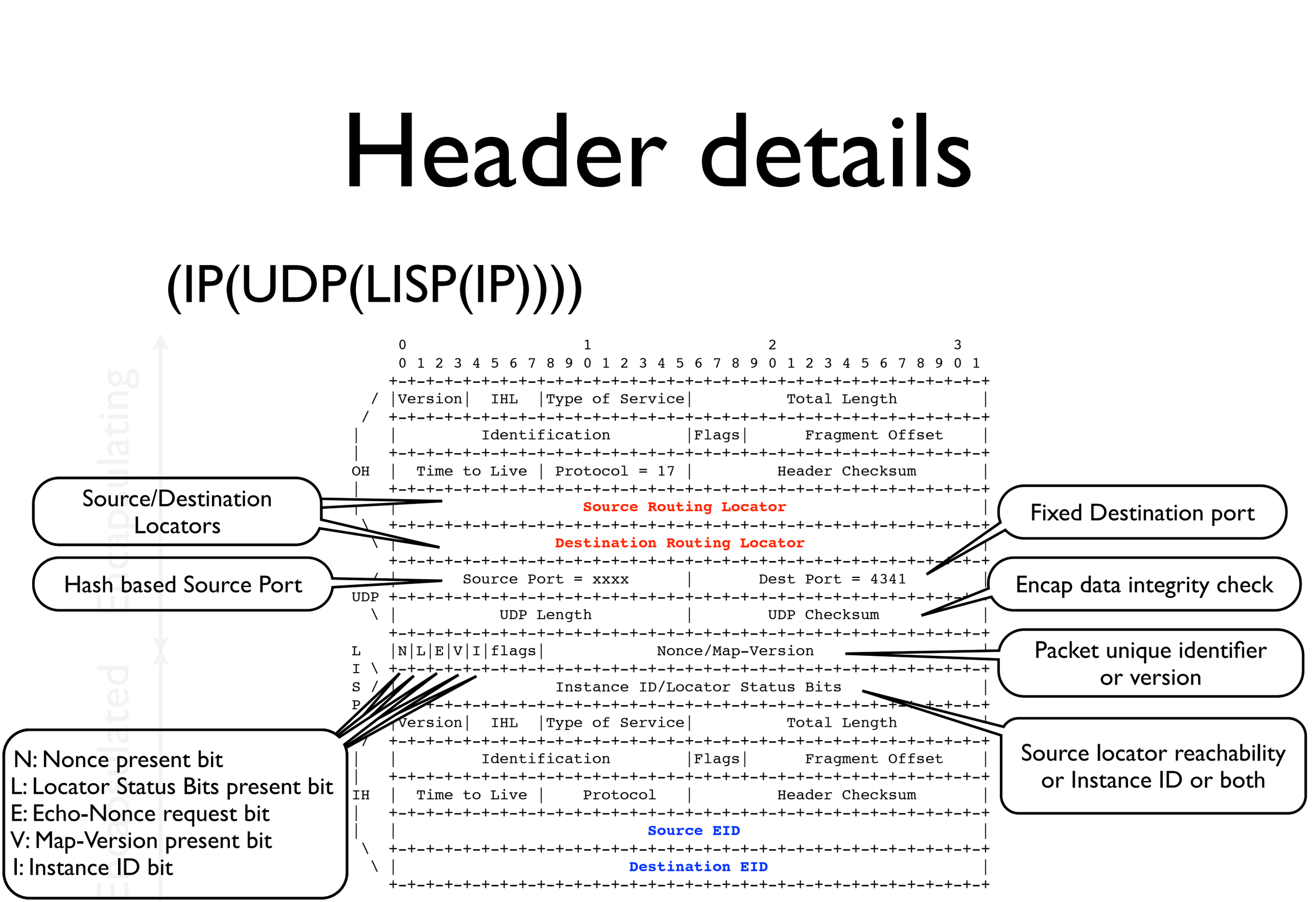
Header details

(IP(UDP(LISP(IP))))



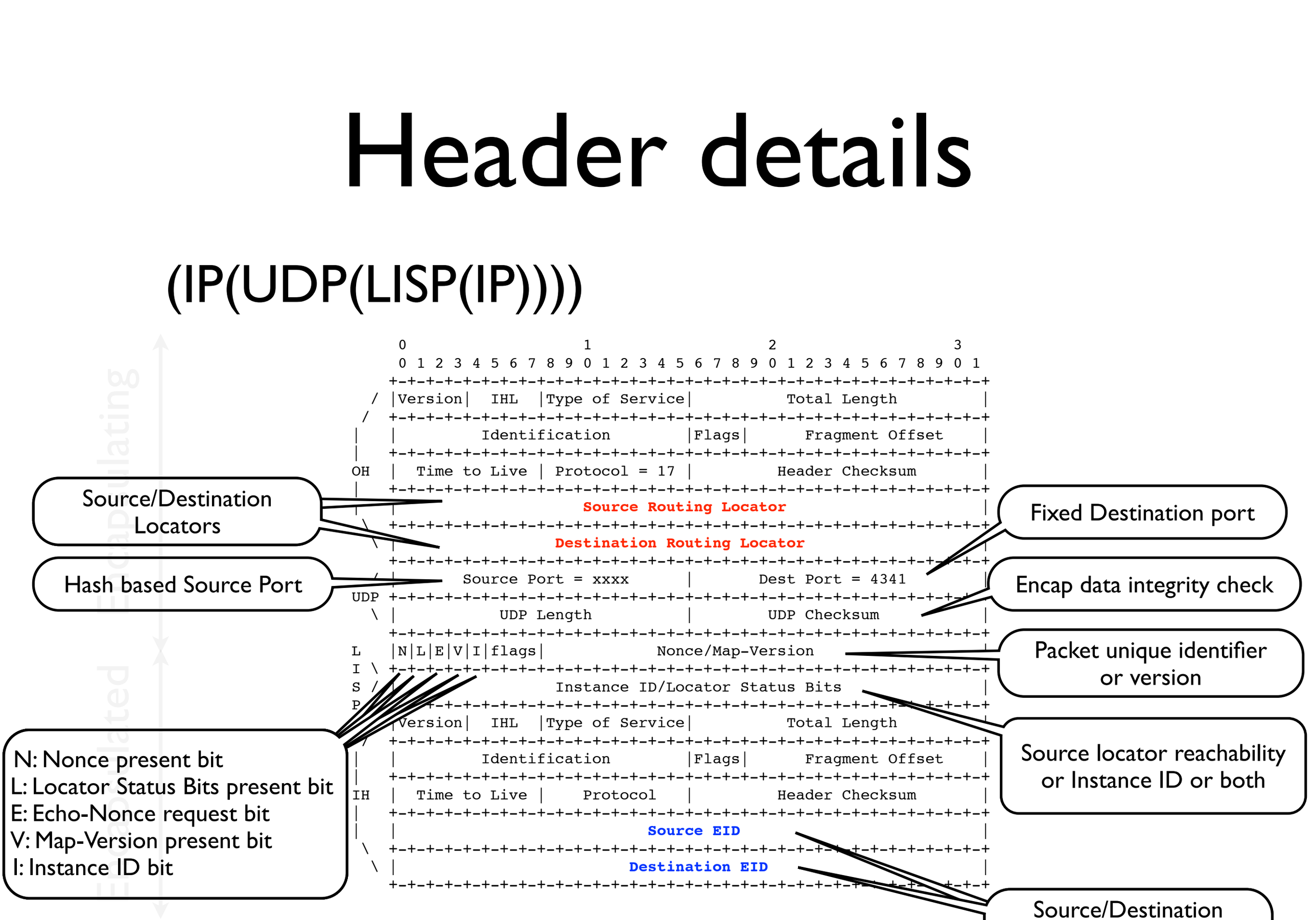
Header details

(IP(UDP(LISP(IP))))



Header details

(IP(UDP(LISP(IP))))



Src/Dst Locators

- Source locator: IP address of the ITR that performed the encapsulation
- Destination locator: IP address of one ETR responsible for the destination EID's prefix
- Locators can be in **IPv4 or IPv6**

Src/Dst EID

- Source EID: IP address of the source end-host
- Destination EID: IP address of destination end-hosts
- EIDs can be in **IPv4 or IPv6**
  - or even L2 with LCAF (WiP)
- *AFI*(RLOC) can be different *AFI*(EID)
  - LISP can be an IPv4/IPv6 transition mechanism (but does not support xAFI)

Interworking of LISP and non LISP networks

- Introduce transition devices
  - Consider the legacy Internet as a big LISP site
- **Proxy-ITR**, encapsulates packets, whatever their source address
  - Advertises coarse-aggregated EID prefixes to BGP
- **Proxy ETR**, decapsulates packets, whatever their destination EID
- **LISP-NAT**, rewriting instead of encapsulation
  - ITR replaces source EID by one of its RLOCs