

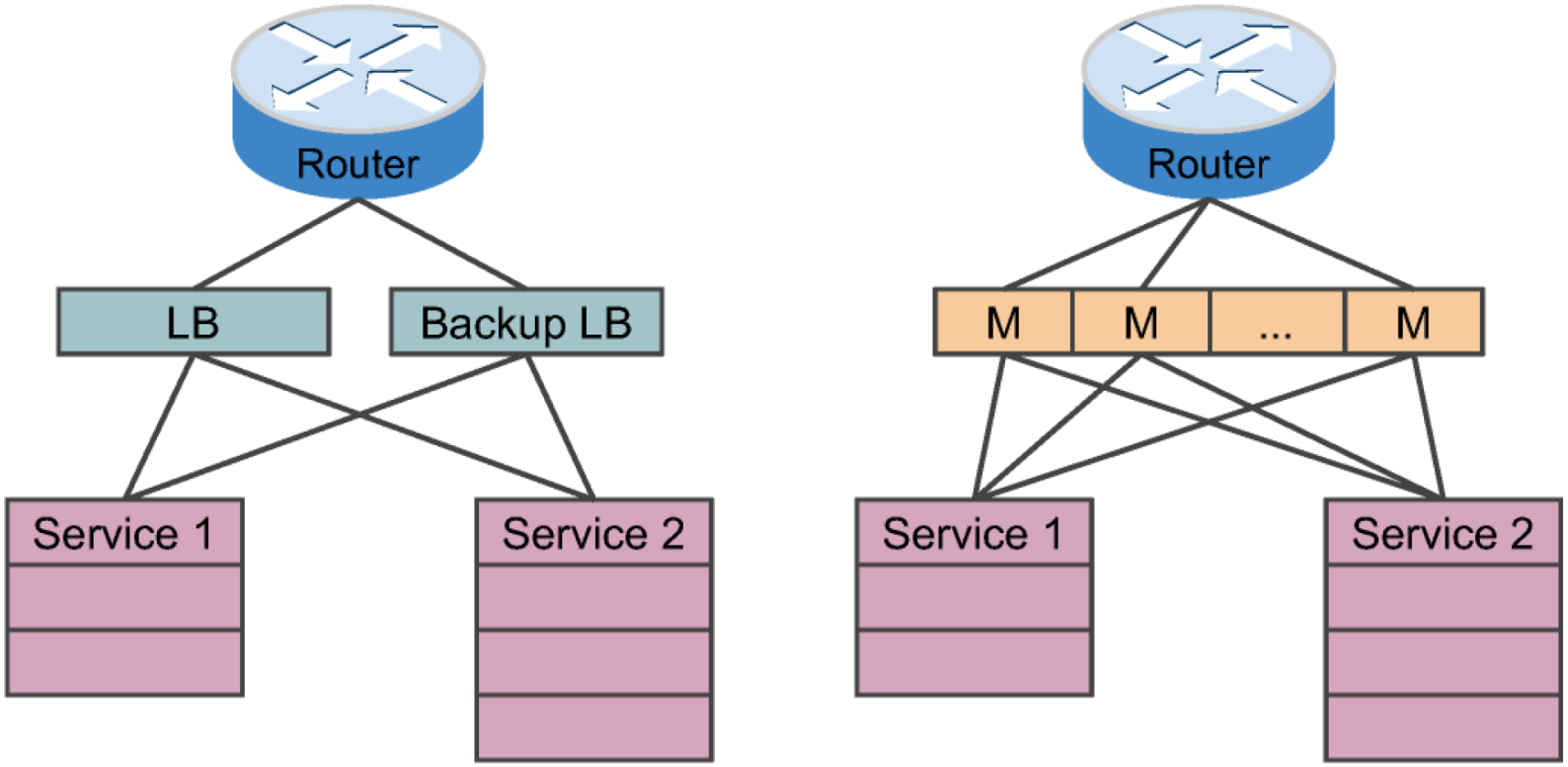
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

HY-559
Infrastructure Technologies for Large-Scale Service-Oriented Systems

Kostas Magoutis
magoutis@csd.uoc.gr
http://www.csd.uoc.gr/~hy559

Network Load Balancing: Google Maglev

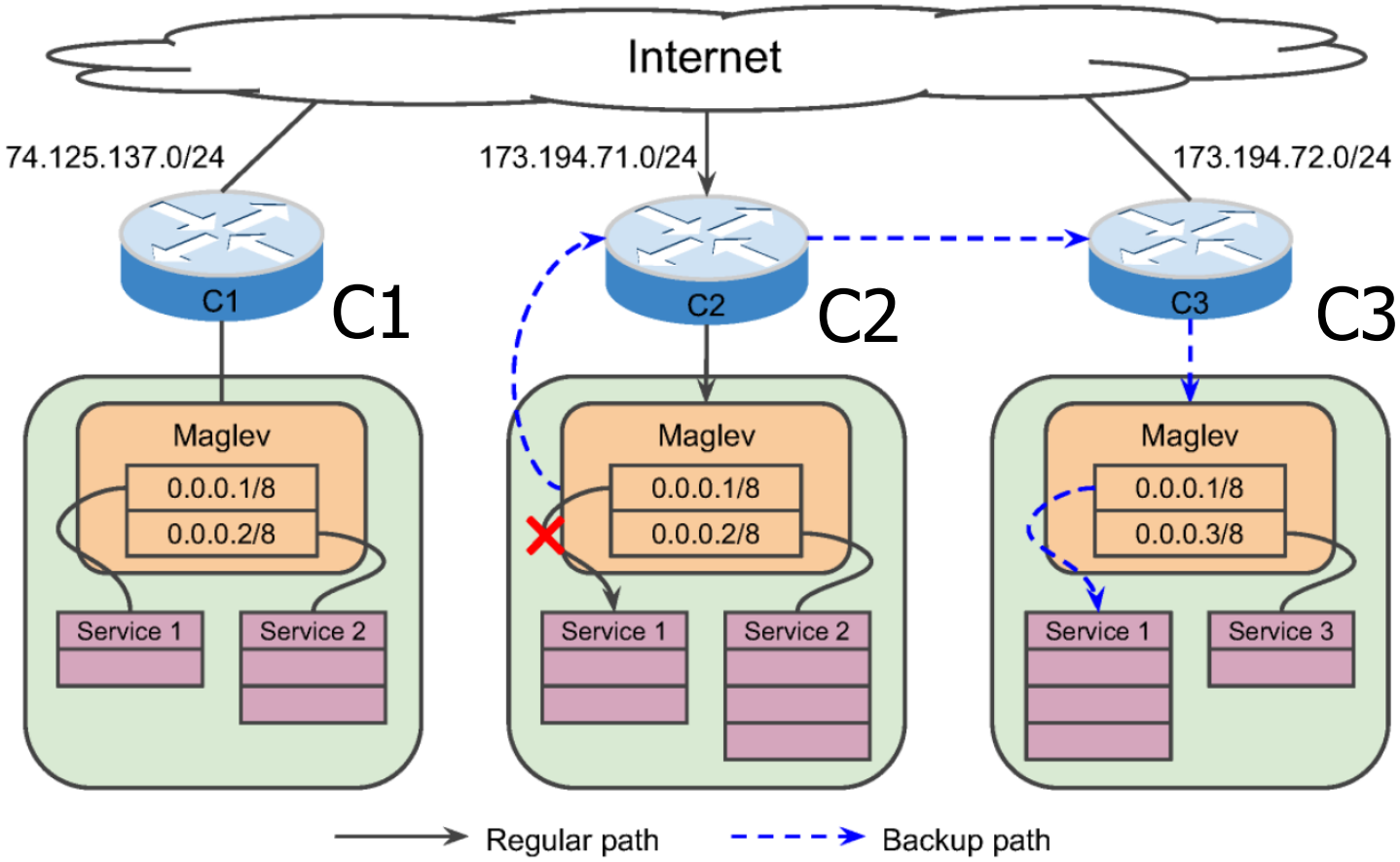
Hardware load balancer and Maglev



Reference: Maglev: A Fast and Reliable Software Network Load Balancer, Usenix NSDI'16, https://dl.acm.org/doi/proceedings/10.5555/2930611

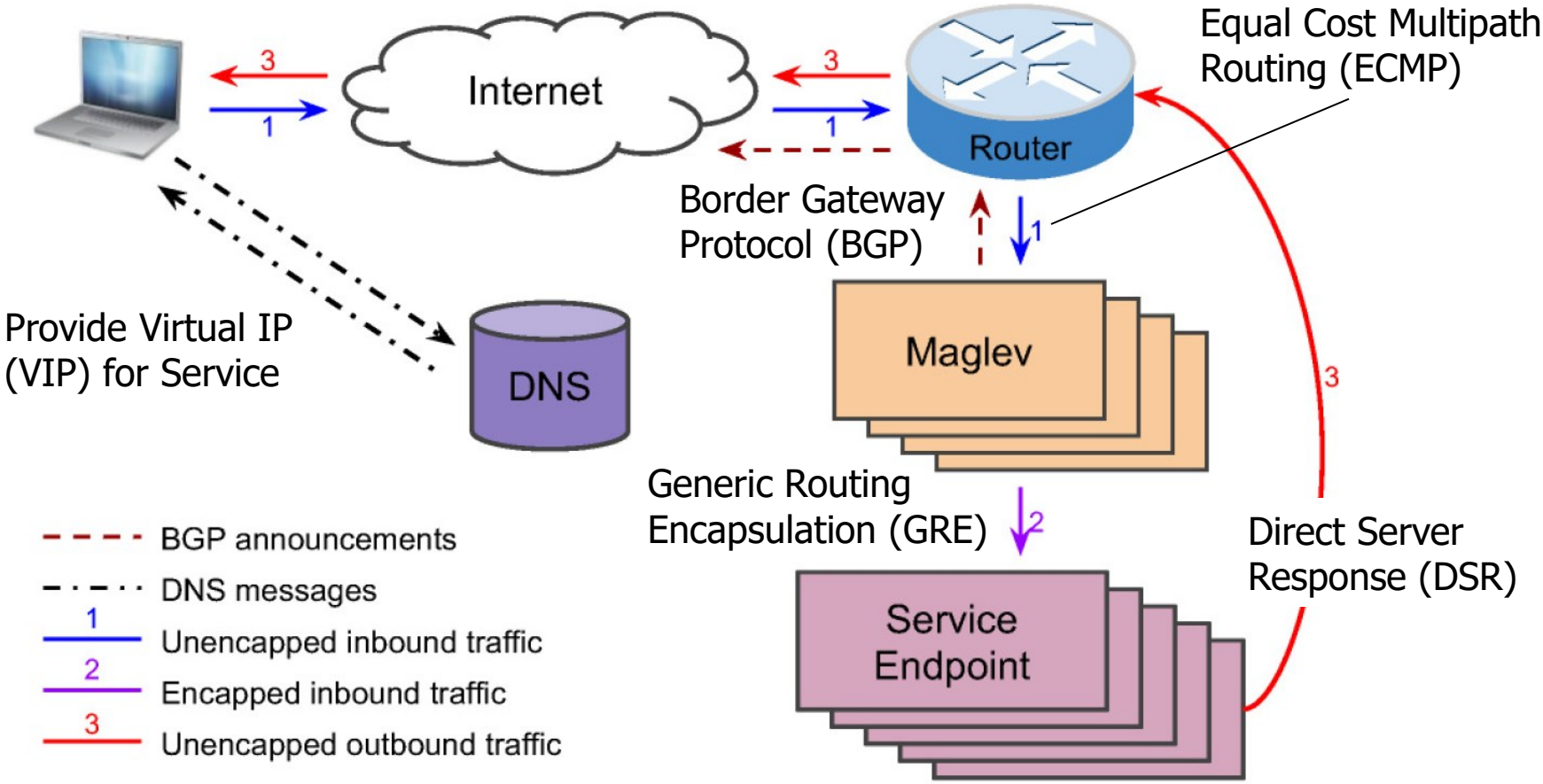
Virtual IP (VIP) assignment

- The same service is configured as different VIPs in different clusters
- The user is directed to one of them by DNS
- Service1 is configured as 74.125.137.1 in C1 and 173.194.71.1 in C2



Maglev packet flow

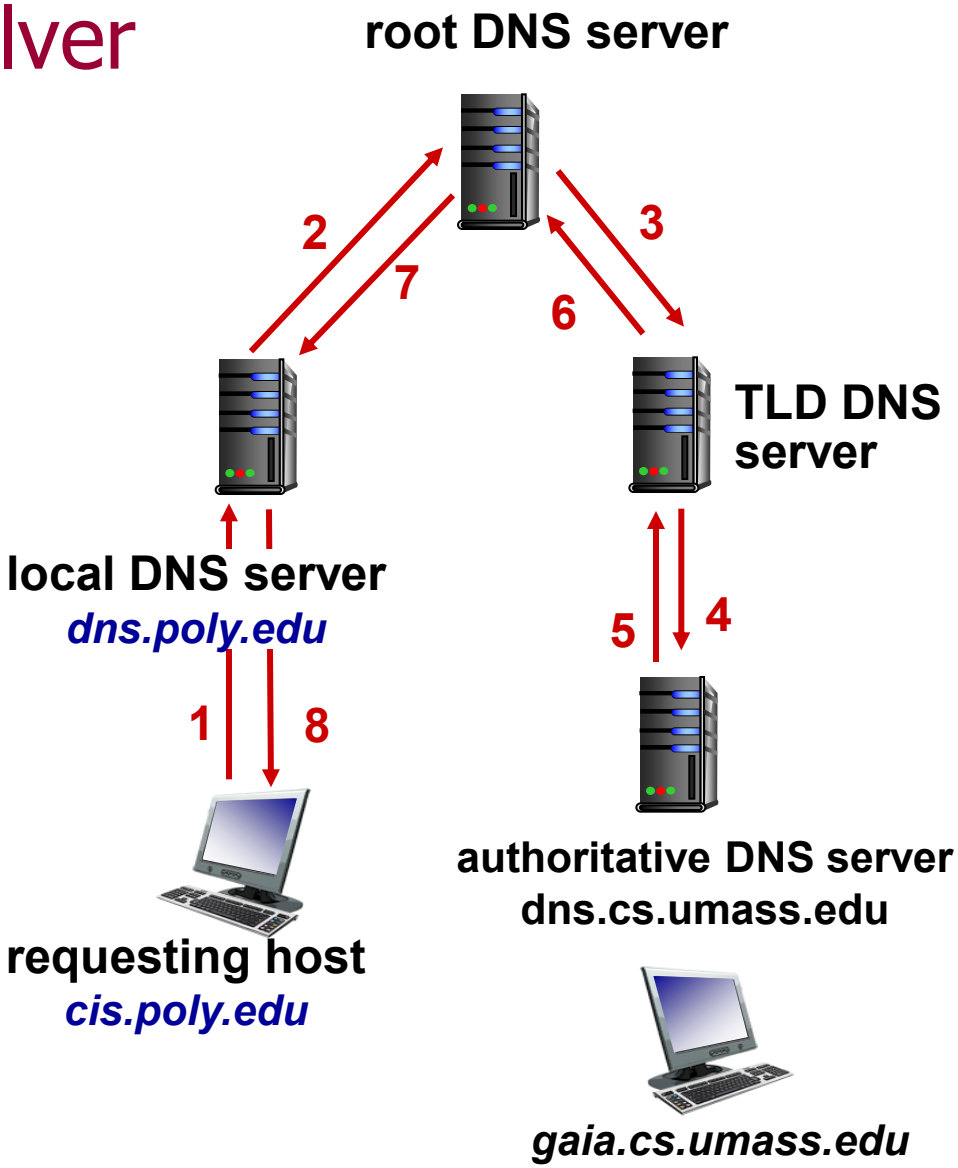
BGP advertises prefix of each cluster e.g., 74.125.137.0/24



Reference: Maglev: A Fast and Reliable Software Network Load Balancer, Usenix NSDI'16, https://dl.acm.org/doi/proceedings/10.5555/2930611

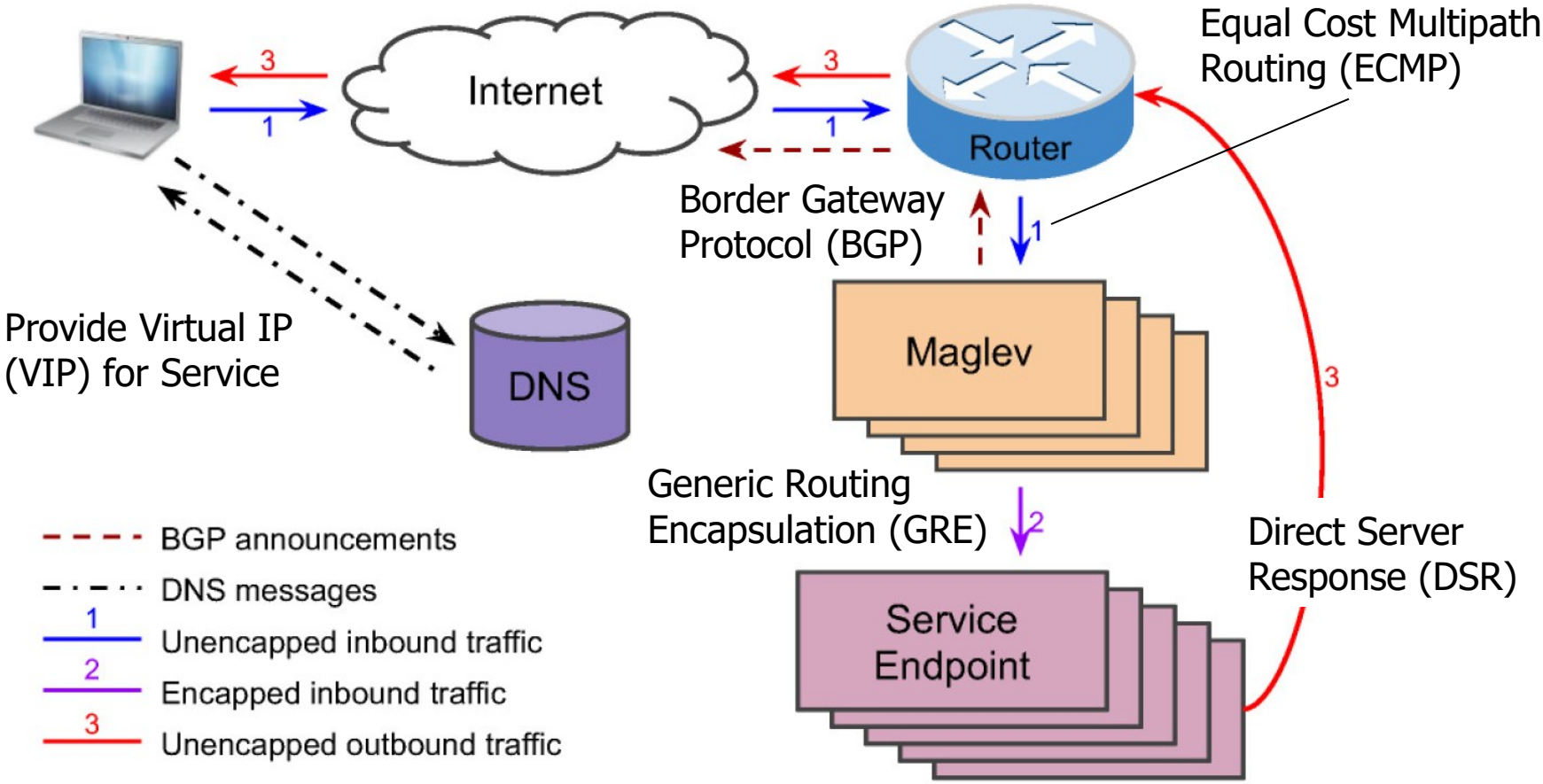
DNS recursive resolver

- EDNS0 extension
  - EDNS Client Subnet
  - Allows a recursive resolver to include the client's IP address or subnet in the query
  - Provide more relevant and geographically closer responses, improving performance and lowering latency for end user



Maglev packet flow

BGP advertises prefix of each cluster e.g., 74.125.137.0/24

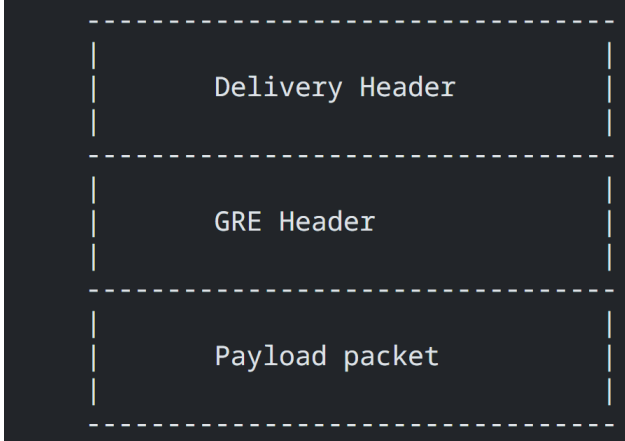


Reference: Maglev: A Fast and Reliable Software Network Load Balancer, Usenix NSDI'16, https://dl.acm.org/doi/proceedings/10.5555/2930611

Generic Routing Encapsulation (GRE)

2. Structure of a GRE Encapsulated Packet

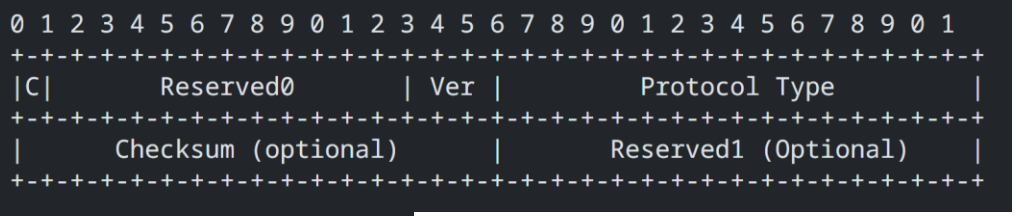
A GRE encapsulated packet has the form:



This specification is generally concerned with the structure of the GRE header, although special consideration is given to some of the issues surrounding IPv4 payloads.

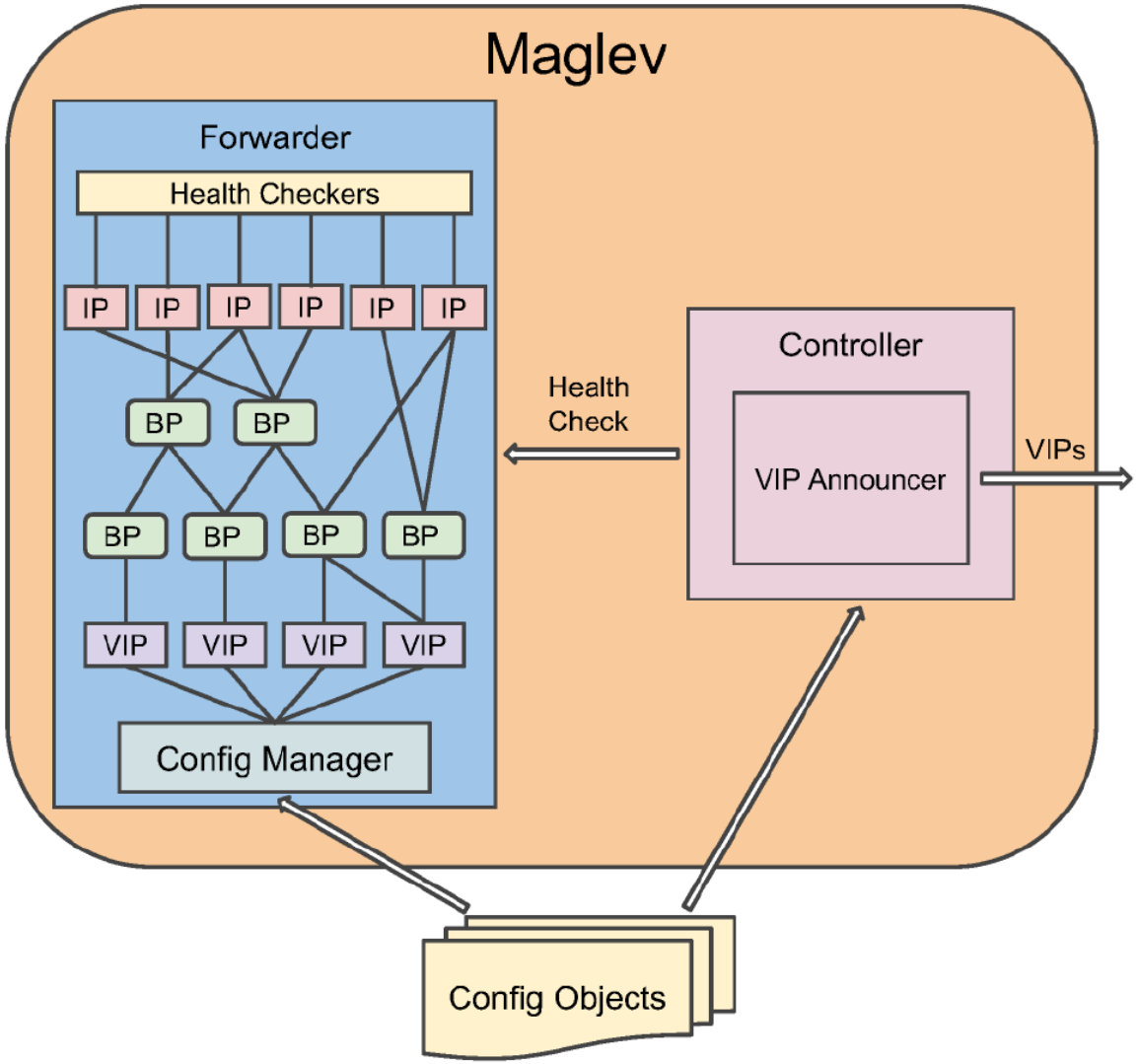
2.1. GRE Header

The GRE packet header has the form:

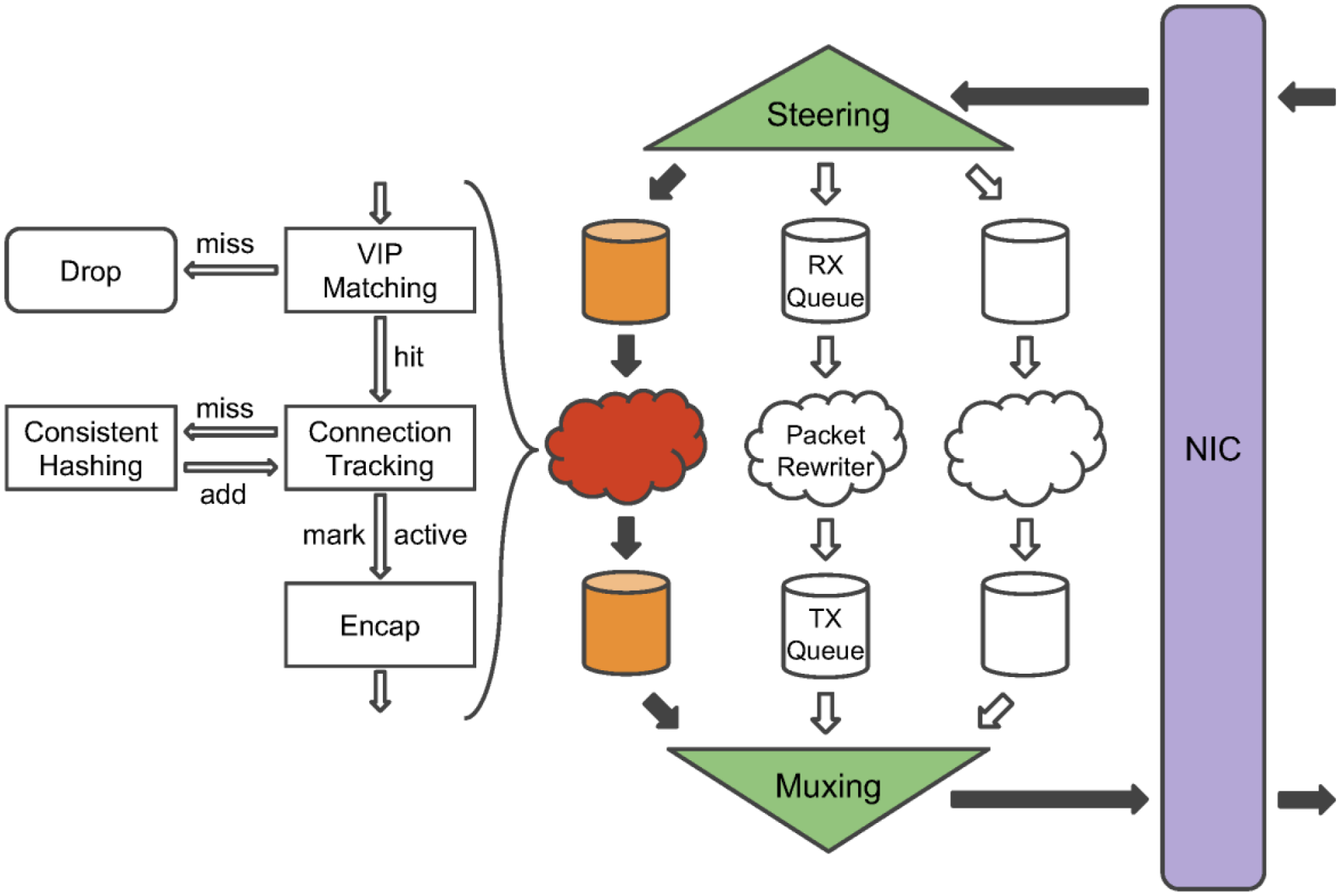


https://datatracker.ietf.org/doc/html/rfc2784

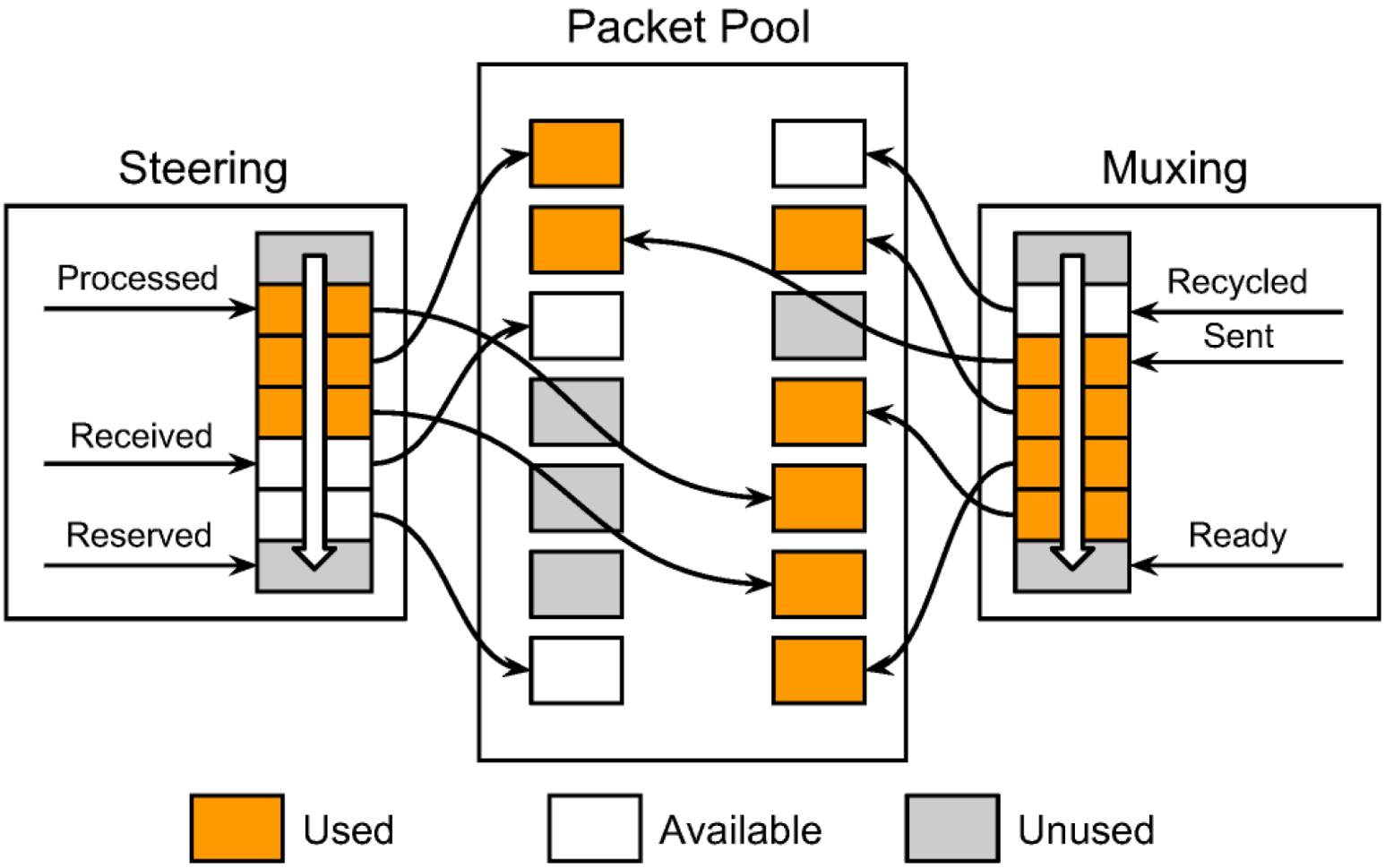
Maglev config



Maglev forwarder structure

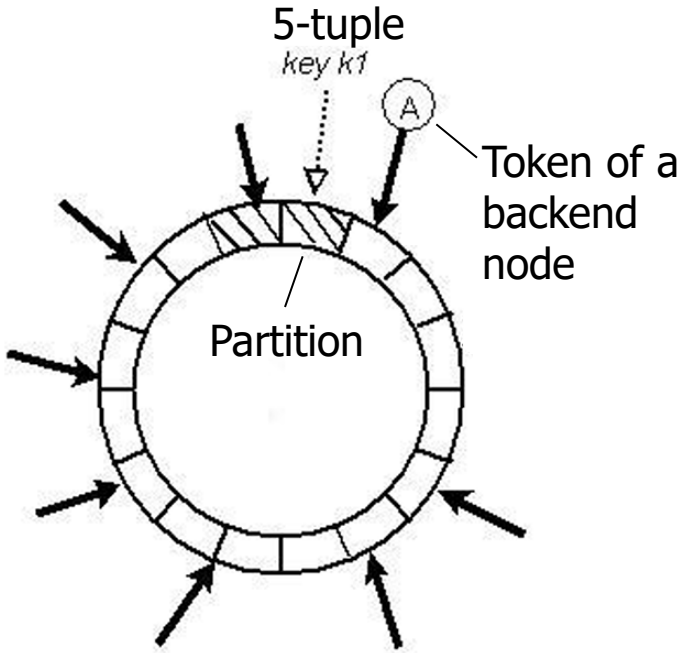


Packet movement in and out of forwarder



Maglev consistent hashing

Consistent hashing

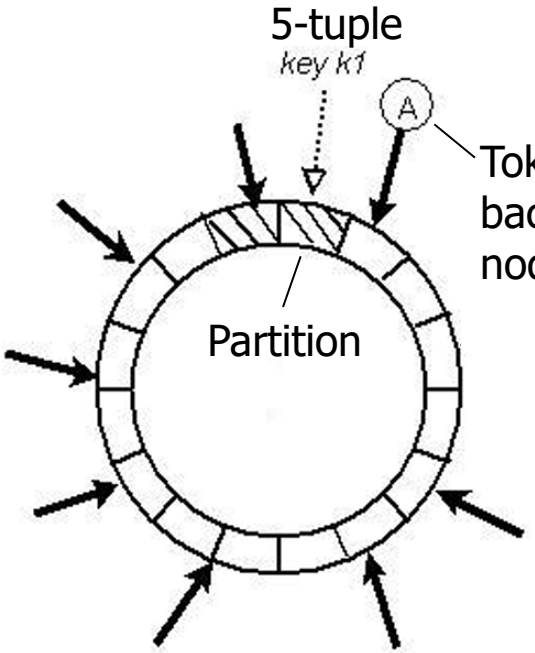


N backend nodes
T random tokens per backend node
M equal-sized partitions
M >> N*T

Maglev consistent hashing

$$offset \leftarrow h_1(name[i]) \mod M$$
$$skip \leftarrow h_2(name[i]) \mod (M-1)+1$$
$$permutation[i][j] \leftarrow (offset + j \times skip) \mod M$$

Consistent hashing



N backend nodes
T random tokens per backend node
M equal-sized partitions
M >> N*T

Per VIP: (3, 4), (0, 2) and (3, 1)

Table 1: A sample consistent hash lookup table.

	$B0$	$B1$	$B2$
0	3	0	3
1	0	2	4
2	4	4	5
3	1	6	6
4	5	1	0
5	2	3	1
6	6	5	2

M

Permutation tables for the backends.

Certain VIPs may have hundreds of backends

M > 100*N to ensure at most a 1% difference in hash space assigned to backends

Maglev consistent hashing

	*B*0	*B*1	*B*2
0	3	0	3
1	0	2	4
2	4	4	5
3	1	6	6
4	5	1	0
5	2	3	1
6	6	5	2

	Before
0	*B*1
1	*B*0
2	*B*1
3	*B*0
4	*B*2
5	*B*2
6	*B*0

1. i=0, next[0]=0, c=permutation[0][0]=3, entry[3]= B0
2. i=1, next[1]=0, c=permutation[1][0]=0, entry[0]= B1
3. i=2, next[2]=0, c=permutation[2][0]=3, entry[3]= (conflict)
4. i=2, next[2]=1, c=permutation[2][1]=4, entry[4]= B2
5. i=0, next[0]=1, c=permutation[0][1]=0, entry[0]= (conflict)
6. i=0, next[0]=2, c=permutation[0][2]=4, entry[4]= (conflict)
7. i=0, next[0]=3, c=permutation[0][3]=1, entry[1]= B0
8. i=1, next[1]=1, c=permutation[1][1]=2, entry[2]= B1
9. i=2, next[2]=2, c=permutation[2][2]=5, entry[5]= B2
10. i=0, next[0]=4, c=permutation[0][4]=5, entry[5]= (conflict)
11. i=0, next[0]=5, c=permutation[0][5]=2, entry[2]= (conflict)
12. i=0, next[0]=6, c=permutation[0][6]=6, entry[6]= B0

Maglev consistent hashing

	*B*0
0	3
1	0
2	4
3	1
4	5
5	2
6	6

*B*2
3
4
5
6
0
1
2

	After
0	*B*0
1	*B*0
2	*B*0
3	*B*0
4	*B*2
5	*B*2
6	*B*2

1. i=0, next[0]=0, c=permutation[0][0]=3, entry[3]= B0
2. i=2, next[2]=0, c=permutation[2][0]=3, entry[3]= (conflict)
3. i=2, next[2]=1, c=permutation[2][1]=4, entry[4]= B2
4. i=0, next[0]=1, c=permutation[0][1]=0, entry[0]= B0
5. i=2, next[2]=2, c=permutation[2][2]=5, entry[5]= B2
6. i=0, next[0]=2, c=permutation[0][2]=4, entry[4]= (conflict)
7. i=0, next[0]=3, c=permutation[0][3]=1, entry[1]= B0
8. i=2, next[2]=3, c=permutation[2][3]=6, entry[6]= B2
9. i=0, next[0]=4, c=permutation[0][4]=5, entry[5]= (conflict)
10. i=0, next[0]=5, c=permutation[0][5]=2, entry[2]= B0

Maglev consistent hashing

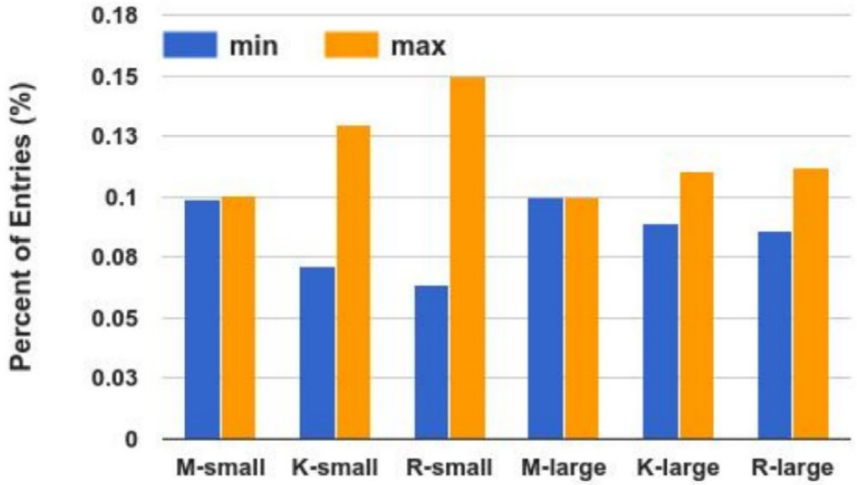
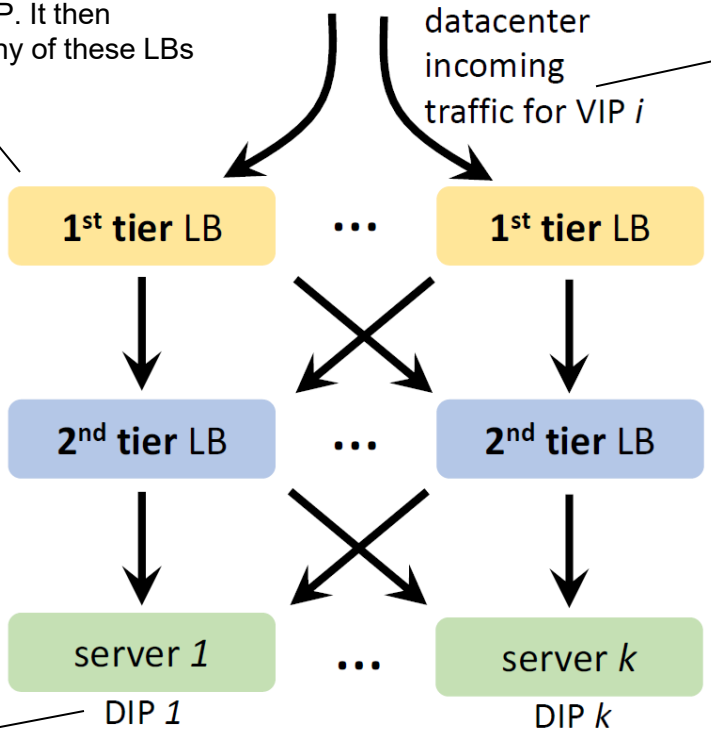


Figure 11: Load balancing efficiency of different hashing methods. M, K and R stand for Maglev, Karger and Rendezvous, respectively. Lookup table size is 65537 for *small* and 655373 for *large*.

- Total number of backends: 1000
- Lookup table size is 65537 (Small) and 655373 (Large)
- Max, min % of table entries per backend for each table size

Summary: Traditional datacenter load balancing architecture

Any 1st tier LB receiving a packet directed to a VIP, performs a look up to fetch the set of 2nd tier LBs responsible for that VIP. It then forwards the packet towards any of these LBs



VIP: Virtual IP

Stateful LBs: Store some information about ongoing connections

Stateless LBs: Do not store any per-connection state

DIP: Direct IP

Figure 1: A traditional datacenter load balancing architecture.

Reference: A High-Speed Load-Balancer Design with Guaranteed Per-Connection-Consistency (Cheetah), NSDI'20, https://dl.acm.org/doi/proceedings/10.5555/3388242

DNS records

RR format: (name, value, type, TTL)

- Type=A
  - name is hostname
  - value is IP address

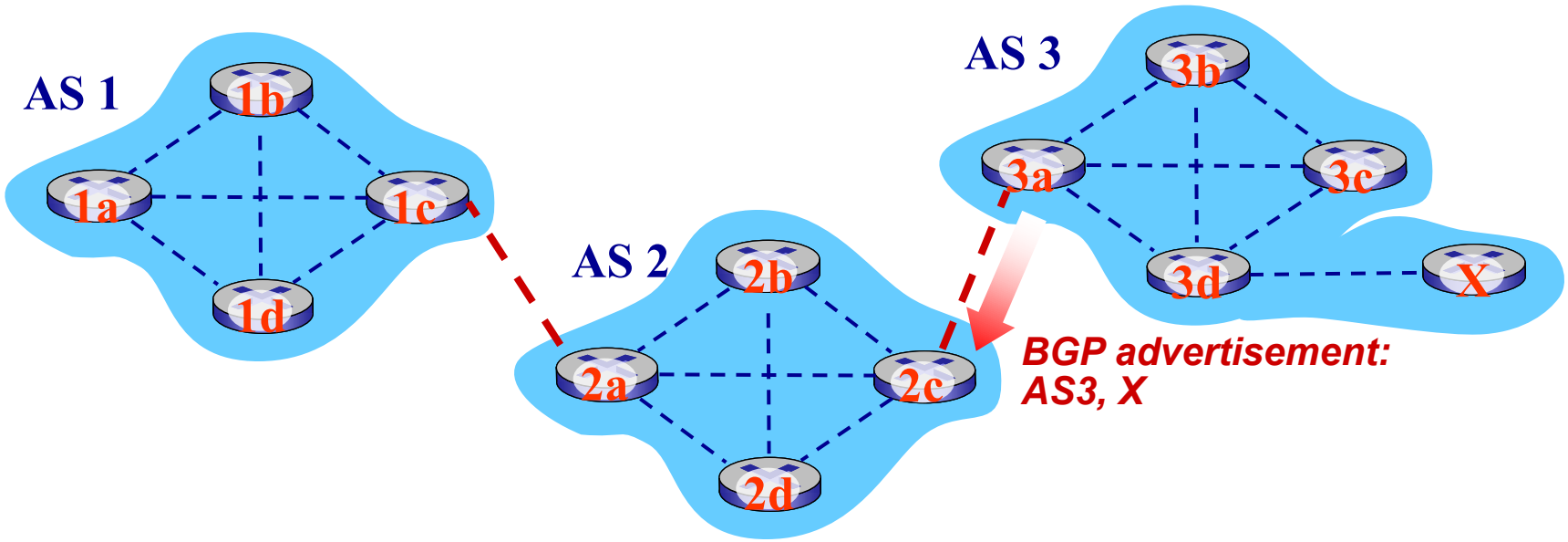
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain

- Type=CNAME
  - name is alias for some "canonical" (real) name
    www.ibm.com is really servereast.backup2.ibm.com
  - value is canonical name

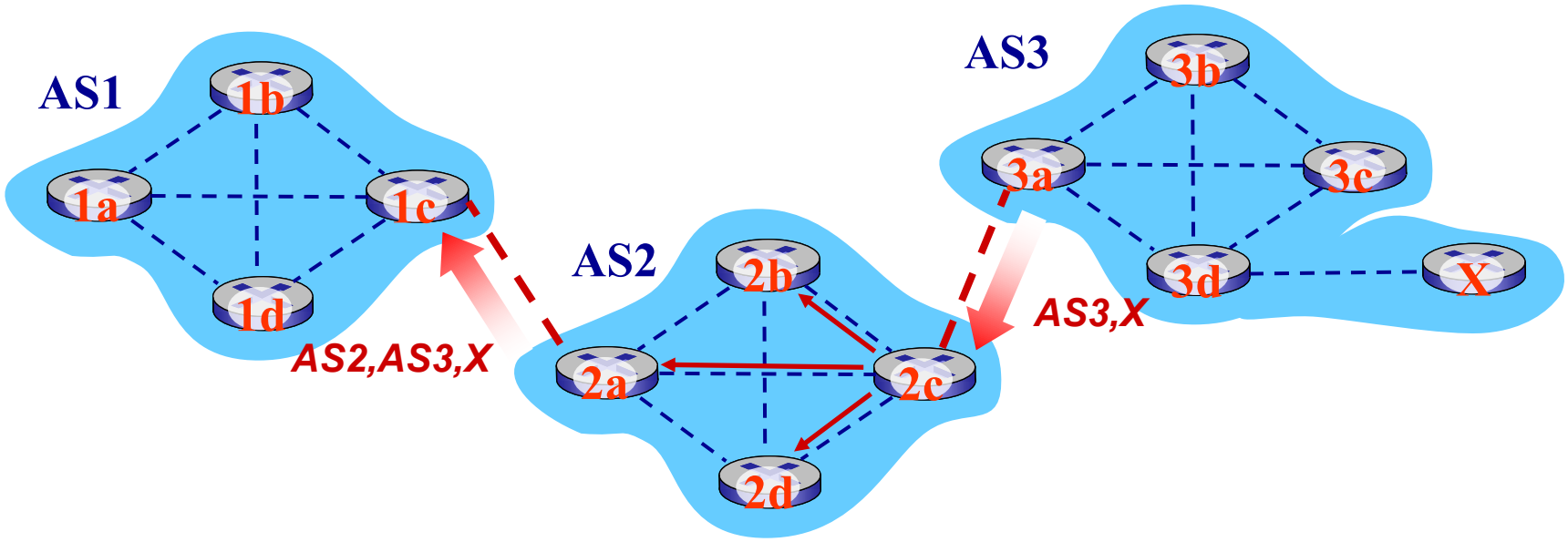
- Type=MX
  - value is name of mail server associated with name

Border Gateway Protocol (BGP) basics

- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- when AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards X



BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

NAT / Reverse NAT

