

multi-homed network

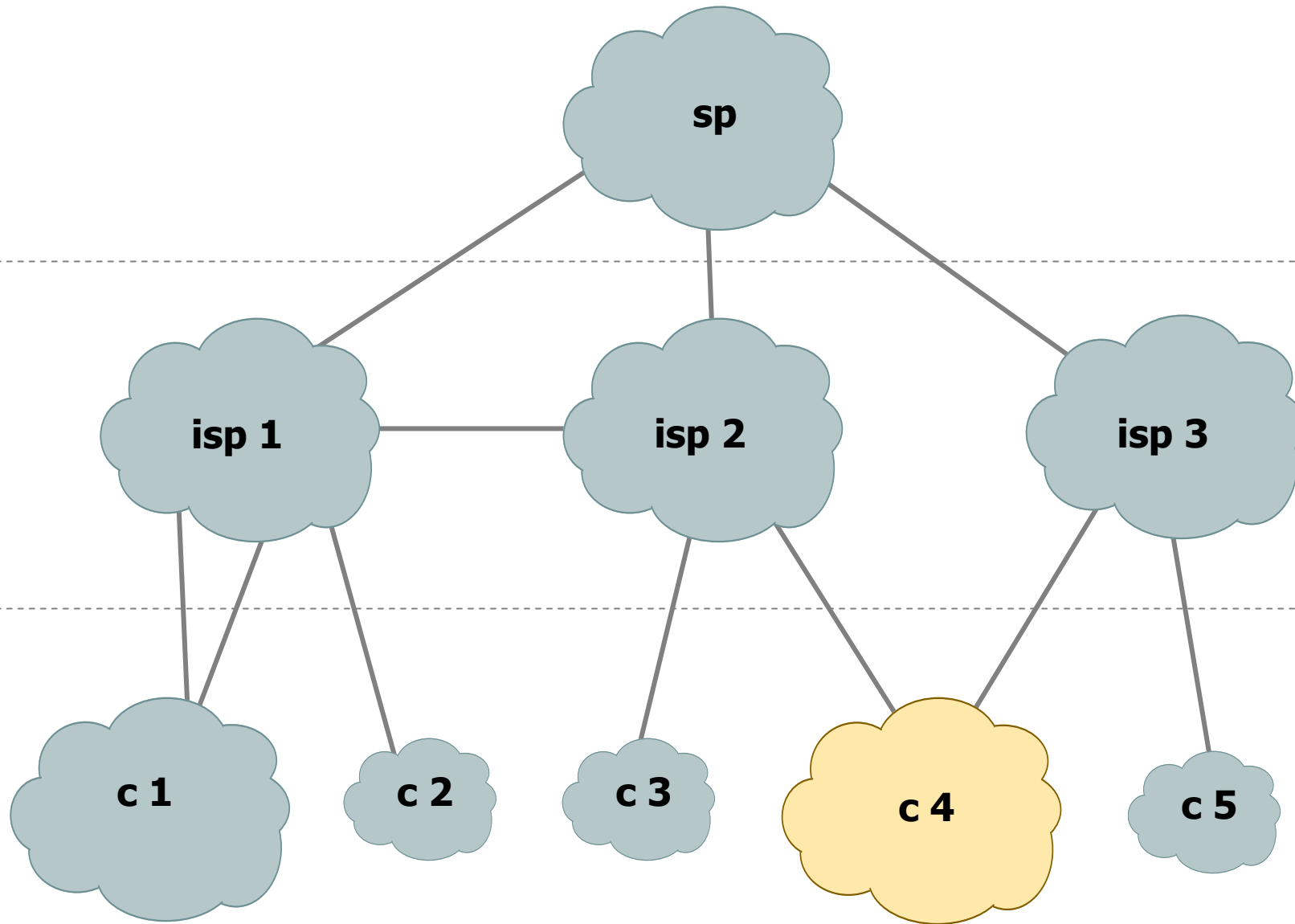
two links to two different isp

# multi-homed network

backbone

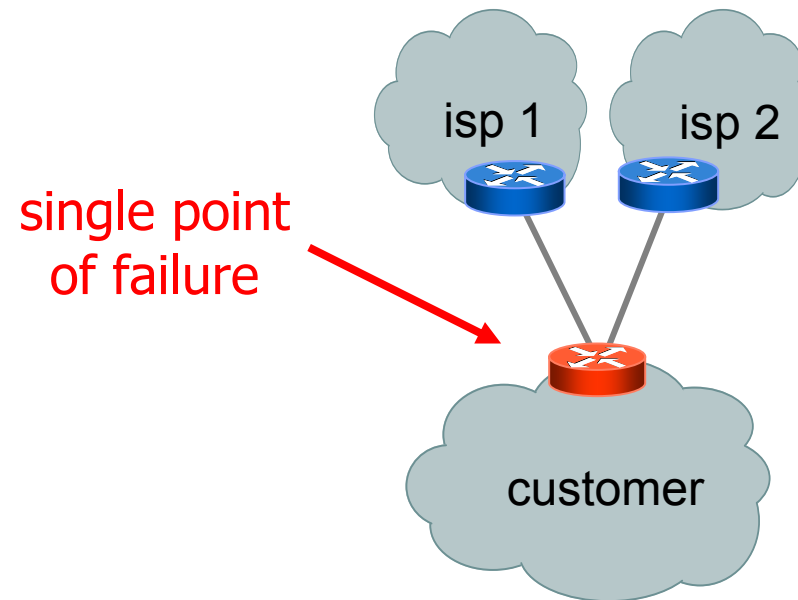
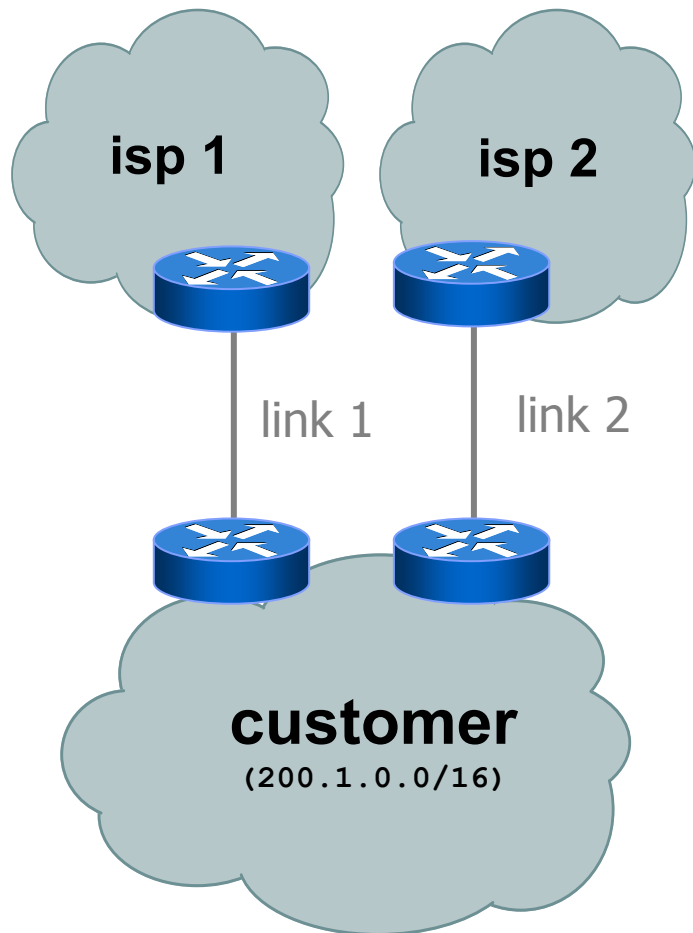
provider

customer



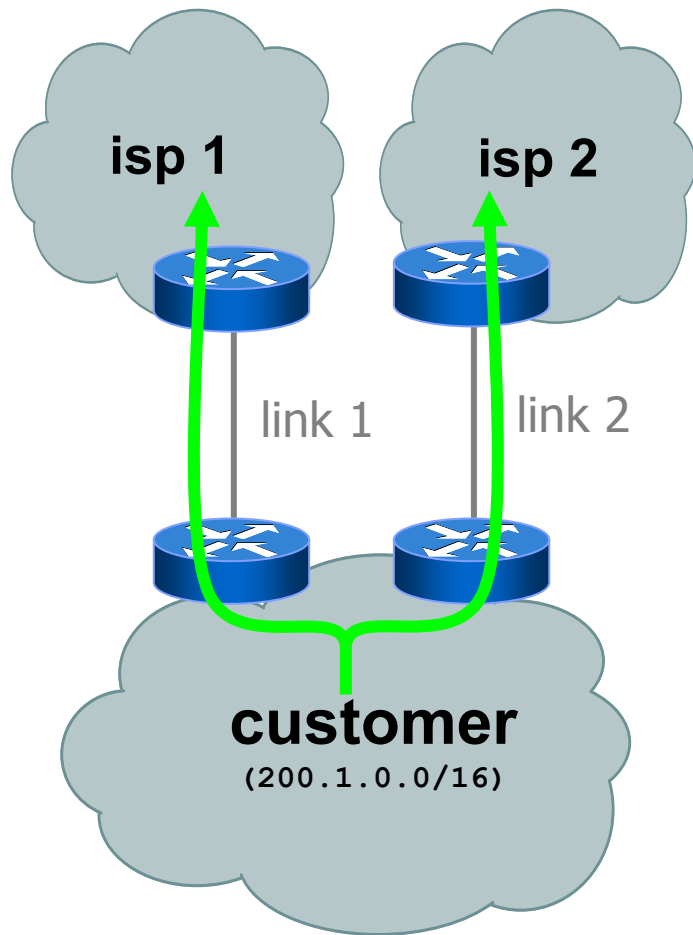
# multi-homed network

- two links to two different providers
- generally two routers are involved in order to avoid single points of failures

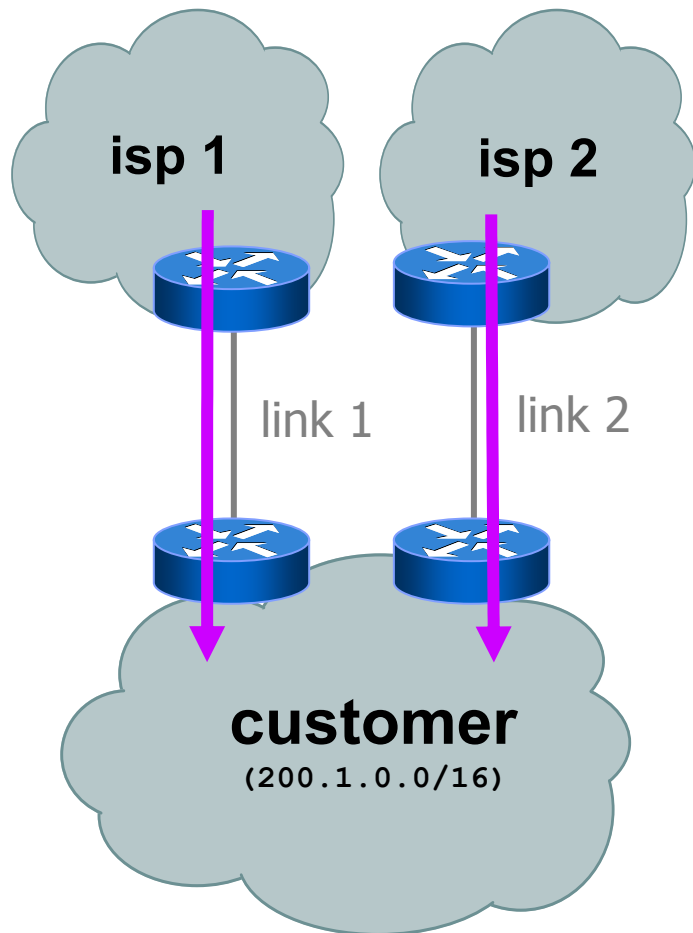


# degrees of freedom

- an outbound packet may be sent through one of the two links in order to reach the internet

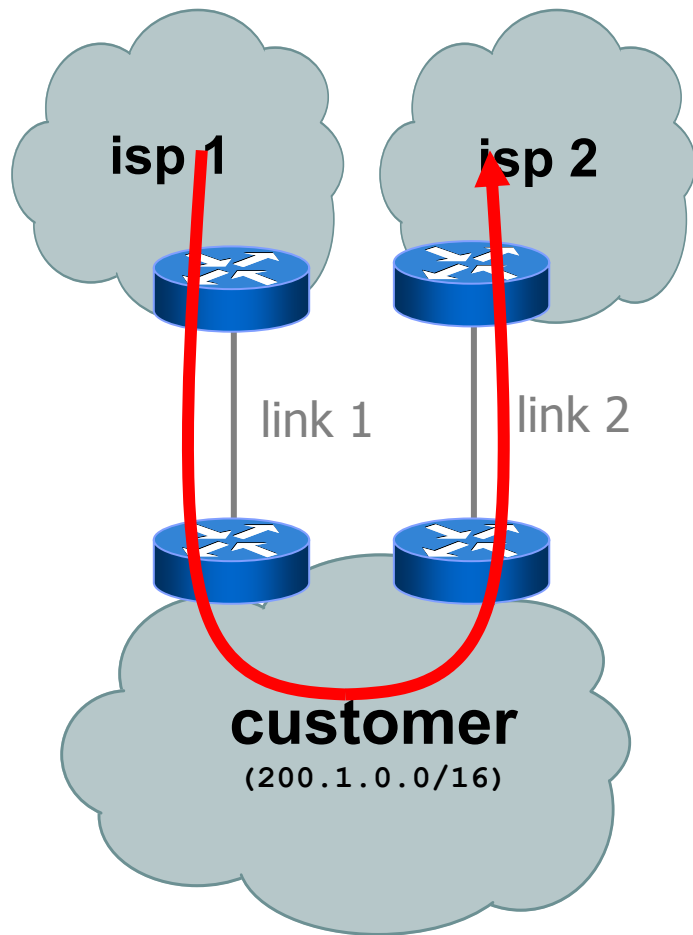


# degrees of freedom



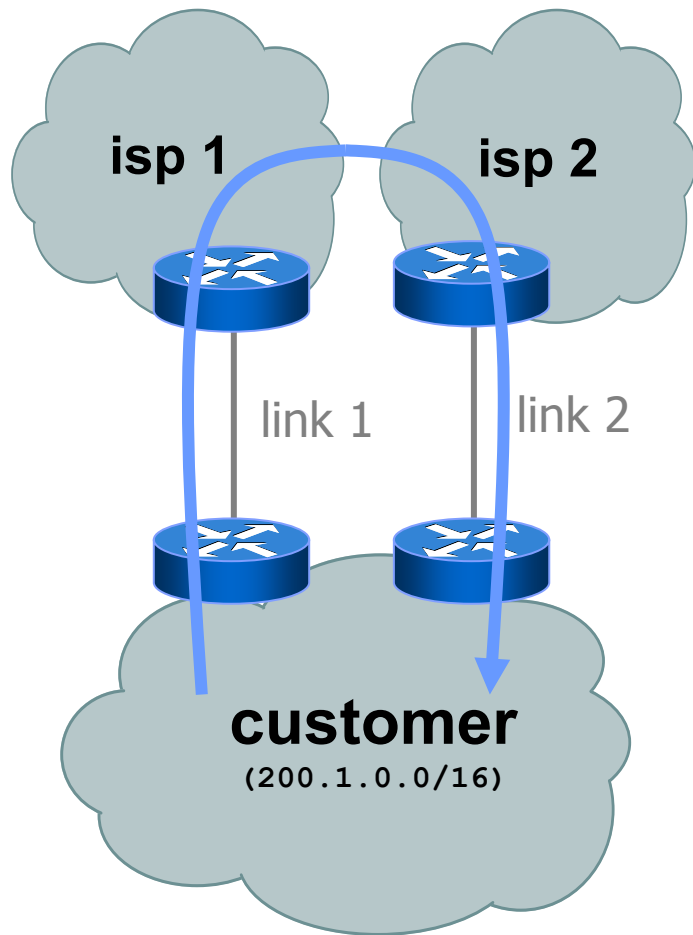
- an outbound packet may be sent through one of the two links in order to reach the internet
- an inbound packet may use any of the two links in order to reach the network

# degrees of freedom



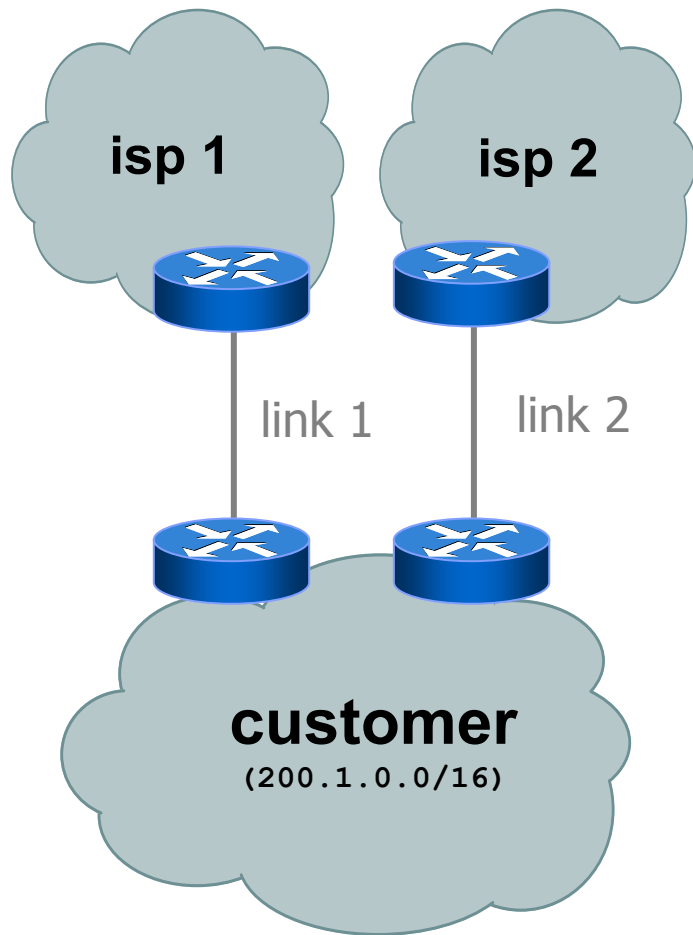
- an outbound packet may be sent through one of the two links in order to reach the internet
- an inbound packet may use any of the two links in order to reach the network
- an internet packet may traverse link 1 and link 2 (or vice versa)

# degrees of freedom



- an outbound packet may be sent through one of the two links in order to reach the internet
- an inbound packet may use any of the two links in order to reach the network
- an internet packet may traverse link 1 and link 2 (or vice versa)
- a local packet may traverse link 1 and link 2 (or vice versa)

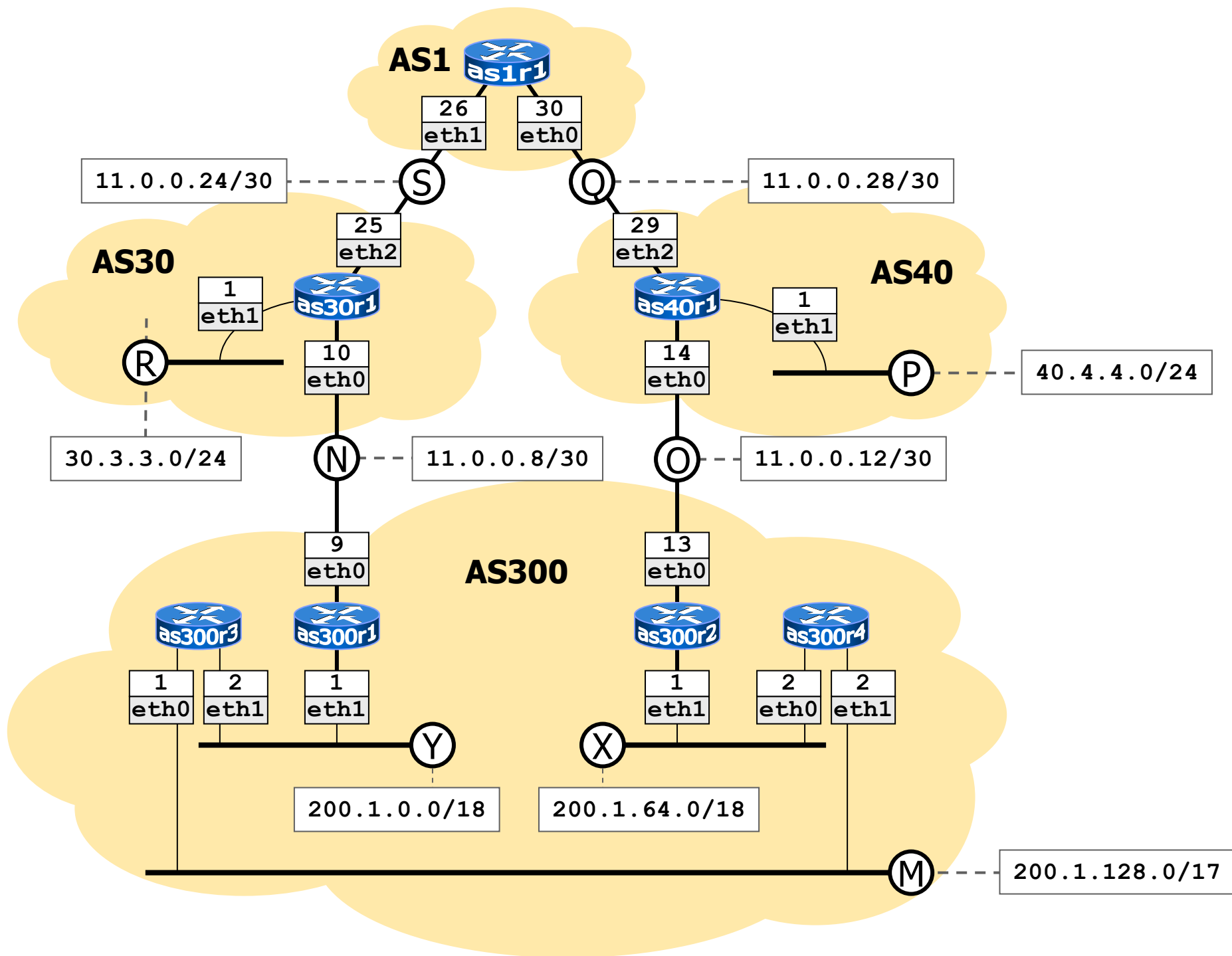
# desired policy: loadsharing



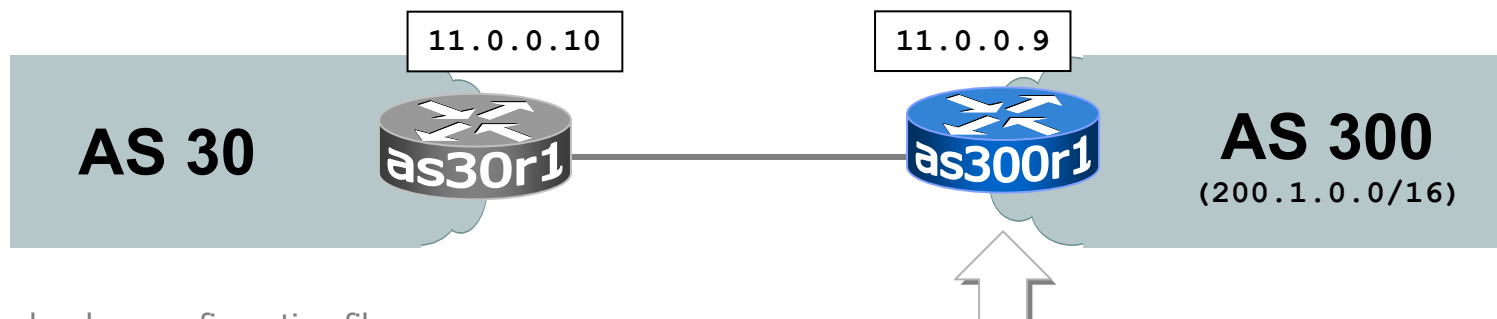
- rule out transit flows
- outbound traffic:
  - half of the internal hosts use link 1
  - the other half uses link 2
- inbound traffic:
  - use link 1 when going to half the internal hosts
  - use link 2 when going to the other half

# using bgp for loadsharing

- announce /16 aggregate on each link
- split /16 and announce as two /17s, one on each link
  - rough loadsharing on inbound traffic
  - assumes equal circuit capacity and even spread of traffic across address block
- vary the split until “perfect” loadsharing achieved
- accept the default from upstream
  - basic outbound loadsharing by nearest exit
  - okay in first approx as most customer traffic is inbound



# router as300r1 configuration



zebra bgp configuration file

```
router bgp 300
network 200.1.0.0/16
network 200.1.0.0/17
!
neighbor 11.0.0.10 remote-as 30
neighbor 11.0.0.10 description Router as30r1
neighbor 11.0.0.10 prefix-list mineOutOnly out
neighbor 11.0.0.10 prefix-list defaultIn in
!
ip prefix-list mineOutOnly permit 200.1.0.0/16
ip prefix-list mineOutOnly permit 200.1.0.0/17
ip prefix-list defaultIn permit 0.0.0.0/0
```

# router as300r2 configuration

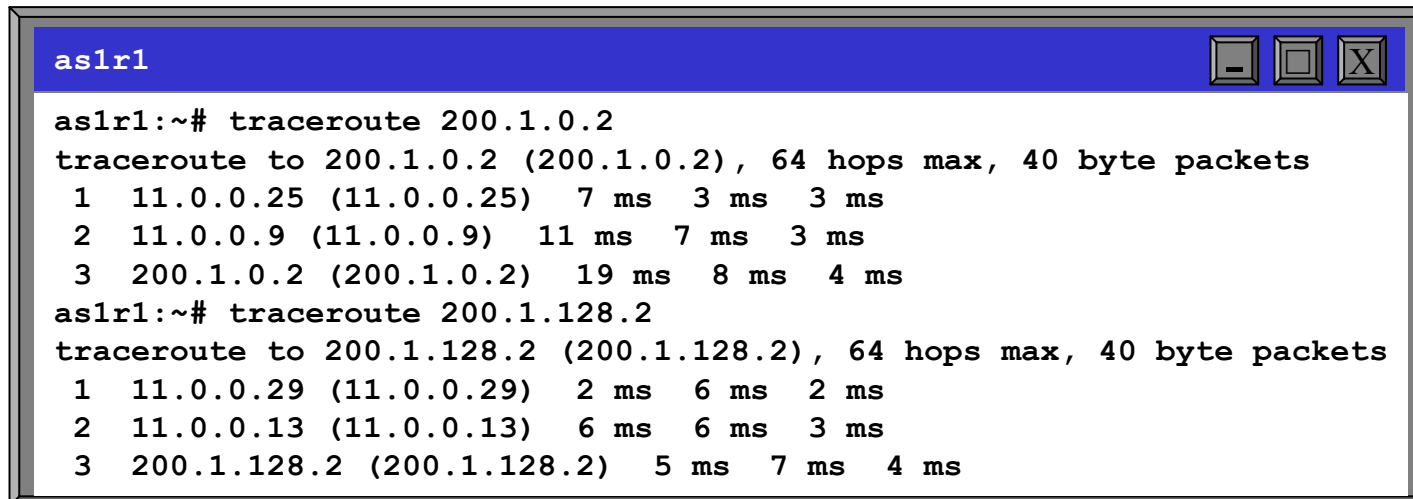


zebra bgp configuration file

```
router bgp 300
network 200.1.0.0/16
network 200.1.128.0/17
!
neighbor 11.0.0.14 remote-as 40
neighbor 11.0.0.14 description Router as40r1
neighbor 11.0.0.14 prefix-list mineOutOnly out
neighbor 11.0.0.14 prefix-list defaultIn in
!
ip prefix-list mineOutOnly permit 200.1.0.0/16
ip prefix-list mineOutOnly permit 200.1.128.0/17
ip prefix-list defaultIn permit 0.0.0.0/0
```

# loadsharing

- experiment loadsharing
  - traceroute from as1r1 to 200.1.0.2 and to 200.1.128.2



```
as1r1
as1r1:~# traceroute 200.1.0.2
traceroute to 200.1.0.2 (200.1.0.2), 64 hops max, 40 byte packets
 1  11.0.0.25 (11.0.0.25)  7 ms  3 ms  3 ms
 2  11.0.0.9 (11.0.0.9)   11 ms  7 ms  3 ms
 3  200.1.0.2 (200.1.0.2)  19 ms  8 ms  4 ms
as1r1:~# traceroute 200.1.128.2
traceroute to 200.1.128.2 (200.1.128.2), 64 hops max, 40 byte packets
 1  11.0.0.29 (11.0.0.29)  2 ms  6 ms  2 ms
 2  11.0.0.13 (11.0.0.13)  6 ms  6 ms  3 ms
 3  200.1.128.2 (200.1.128.2)  5 ms  7 ms  4 ms
```

- check the rip routing inside as300

# backup

- experiment backup

- crash collision domain 0 as follows (as300r2):
  - `bgpd> enable`
  - `bgpd# configure terminal`
  - `bgpd(config)# router bgp 300`
  - `bgpd(config-router)# neighbor 11.0.0.14 shutdown`
- check the routing table of as300r2
- check as1r1 (next slide)

# backup

aslr1

bgpd> show ip bgp

BGP table version is 0, local router ID is 11.0.0.30

Status codes: s suppressed, d damped, h history, \* valid, > best, i - internal

Origin codes: i - IGP, e - EGP, ? - incomplete

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 0.0.0.0	0.0.0.0	0		32768	i
*> 11.0.0.8/30	11.0.0.25	0		0	30 i
*> 11.0.0.12/30	11.0.0.29	0		0	40 i
*> 11.0.0.24/30	0.0.0.0	0		32768	i
*> 11.0.0.28/30	0.0.0.0	0		32768	i
*> 30.3.3.0/24	11.0.0.25	0		0	30 i
*> 40.4.4.0/24	11.0.0.29	0		0	40 i
*> 200.1.0.0/16	11.0.0.25			0	30 300 i
*> 200.1.0.0/17	11.0.0.25			0	30 300 i

Total number of prefixes 9

.....

aslr1:~# traceroute 200.1.128.2

traceroute to 200.1.128.2 (200.1.128.2), 64 hops max, 40 byte packets

1	11.0.0.25 (11.0.0.25)	3 ms	6 ms	2 ms
2	11.0.0.9 (11.0.0.9)	3 ms	4 ms	3 ms
3	200.1.0.2 (200.1.0.2)	13 ms	9 ms	4 ms
4	200.1.128.2 (200.1.128.2)	18 ms	9 ms	5 ms