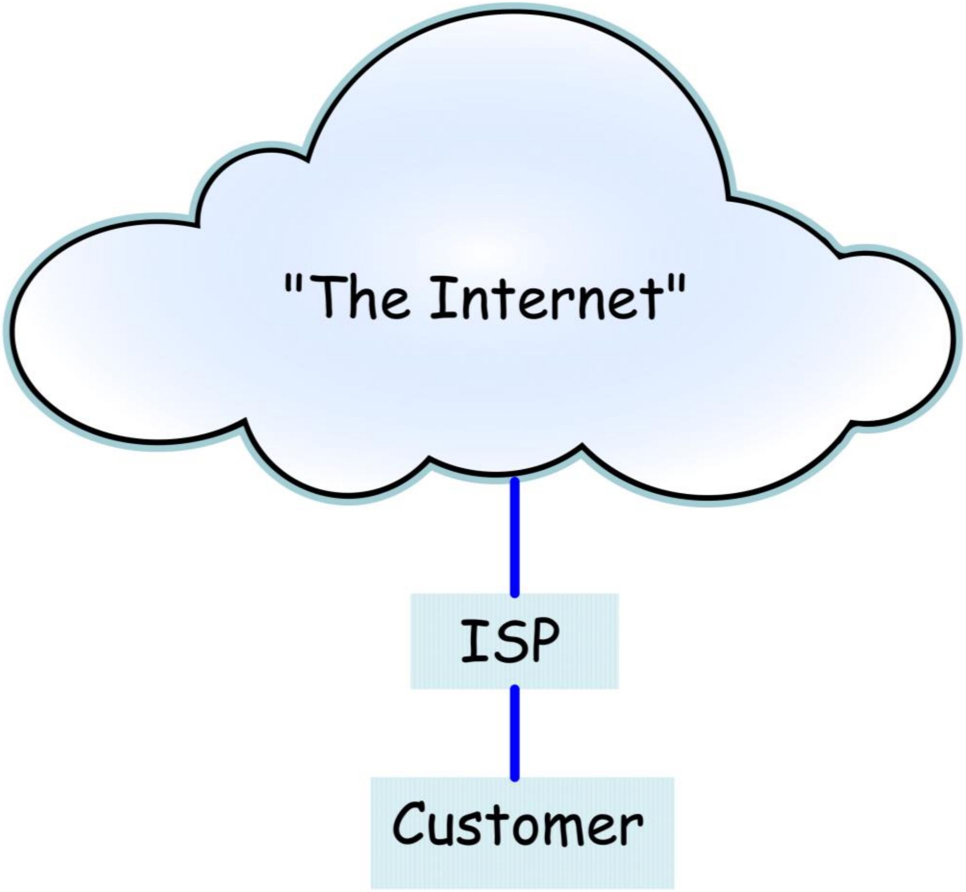
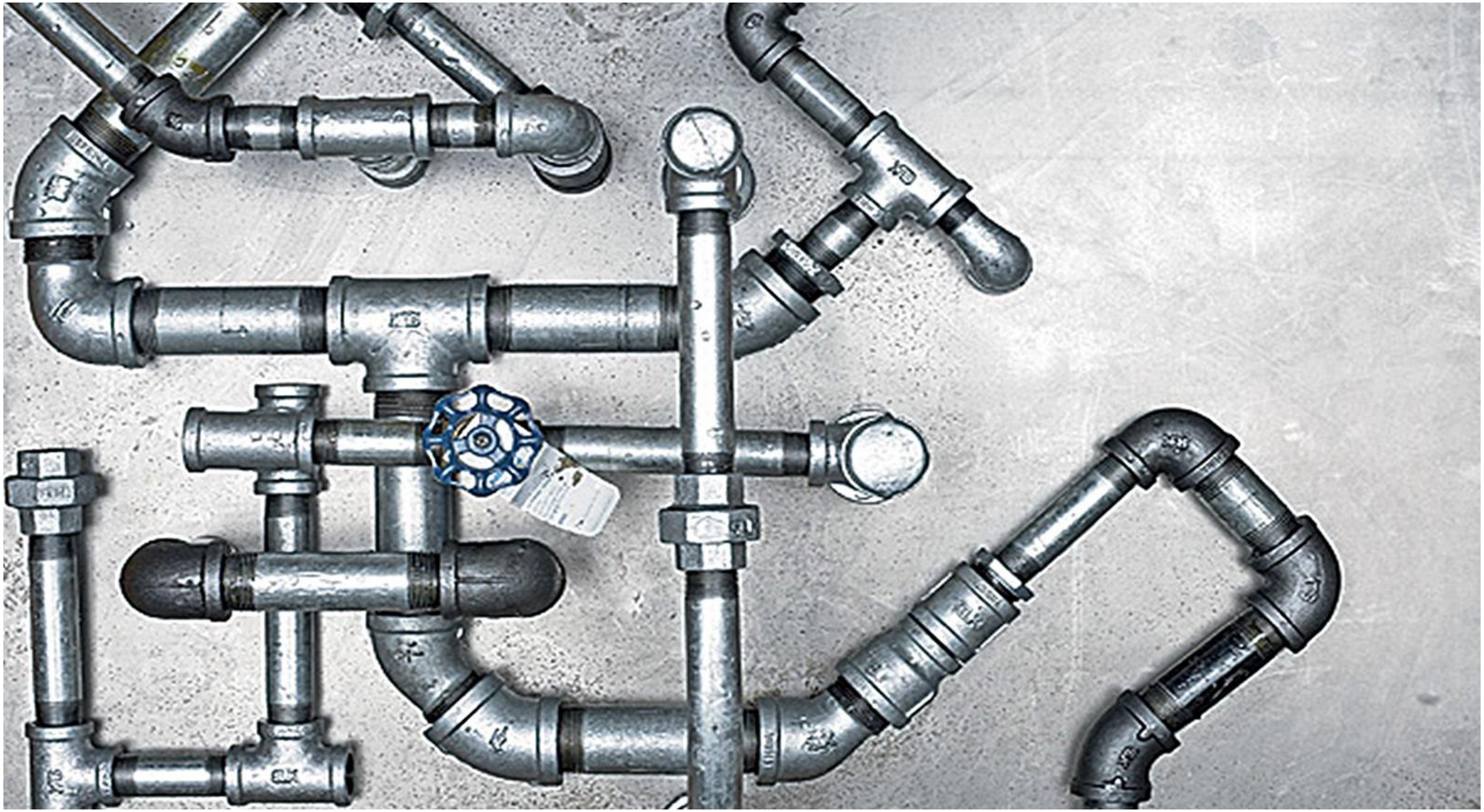
Multihoming Techniques

Customer’s Expectation



But it’s really just…



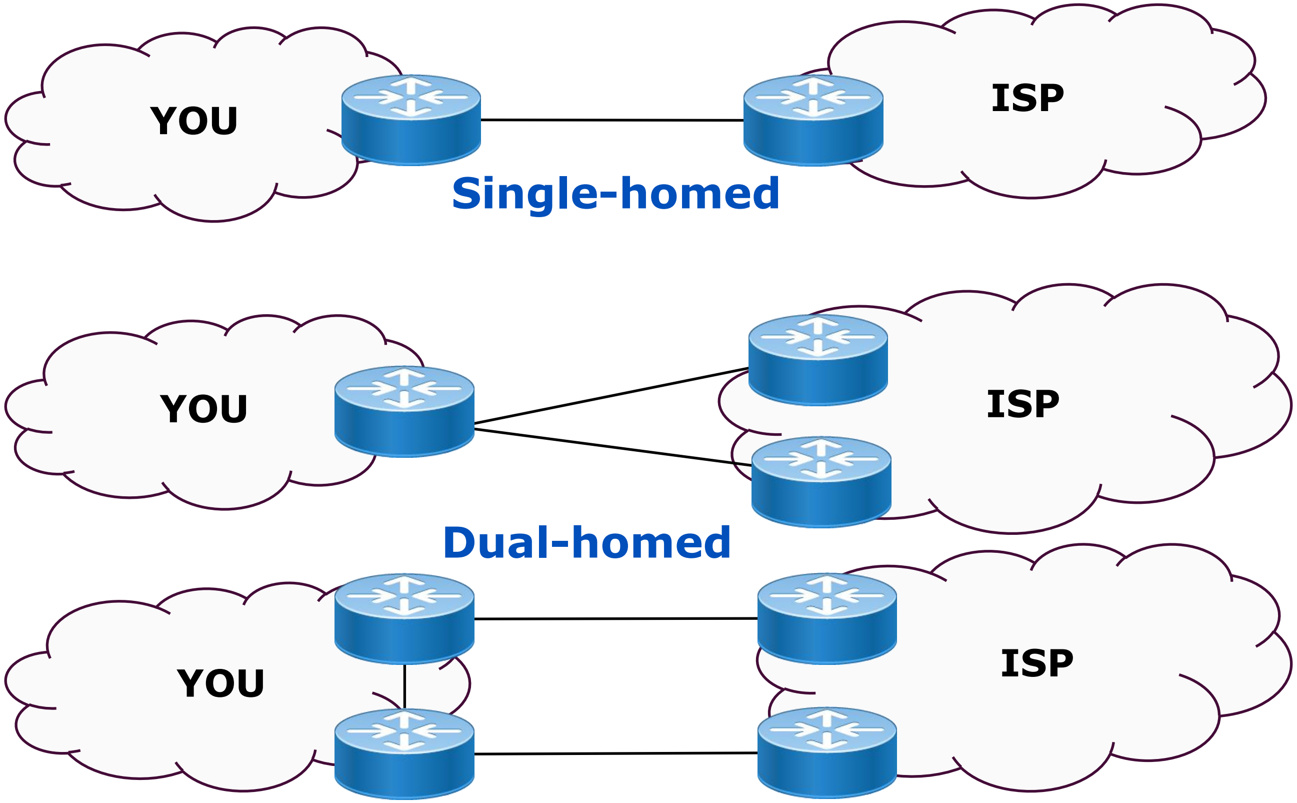
Until this happens

Why Multihome?

- Redundancy
  - Avoid a circuit down taking your business down
  - Multiple ways of doing this
- Performance
  - Maybe provider A’s connectivity is different to provider B’s
  - Asia focus, Europe focus
  - Satellite versus Fiber
- Money
  - Good, Fast, Cheap – Pick Two
  - Buy transit but also peer where you can

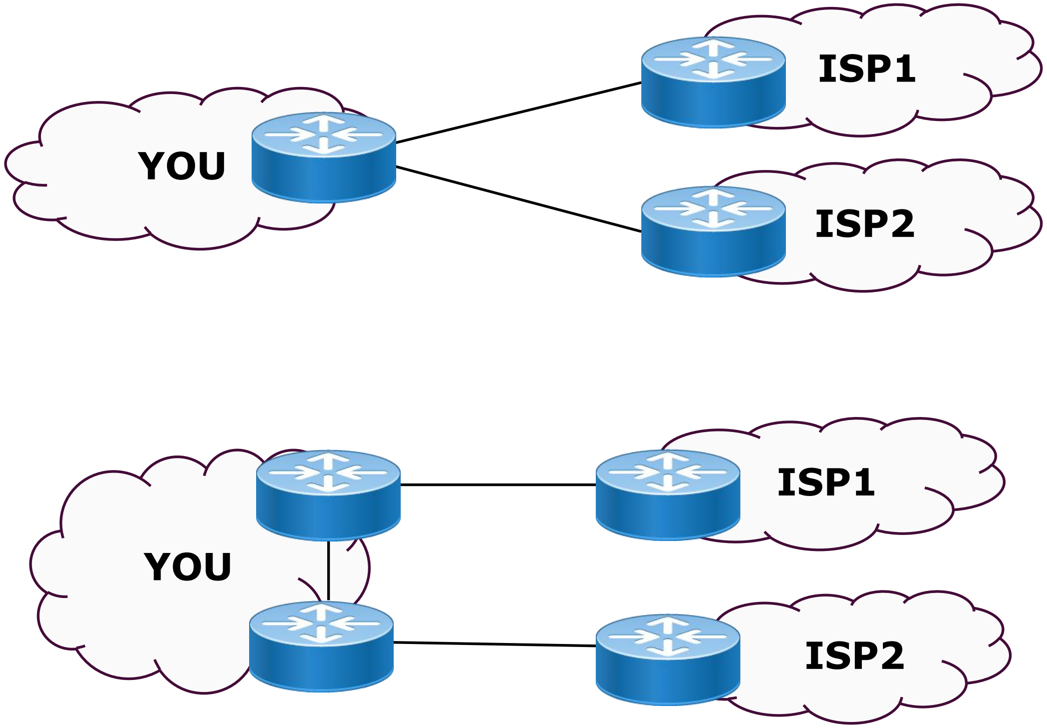
Achieving Redundancy

- More than one path to the same ISP
  - Dual-homed



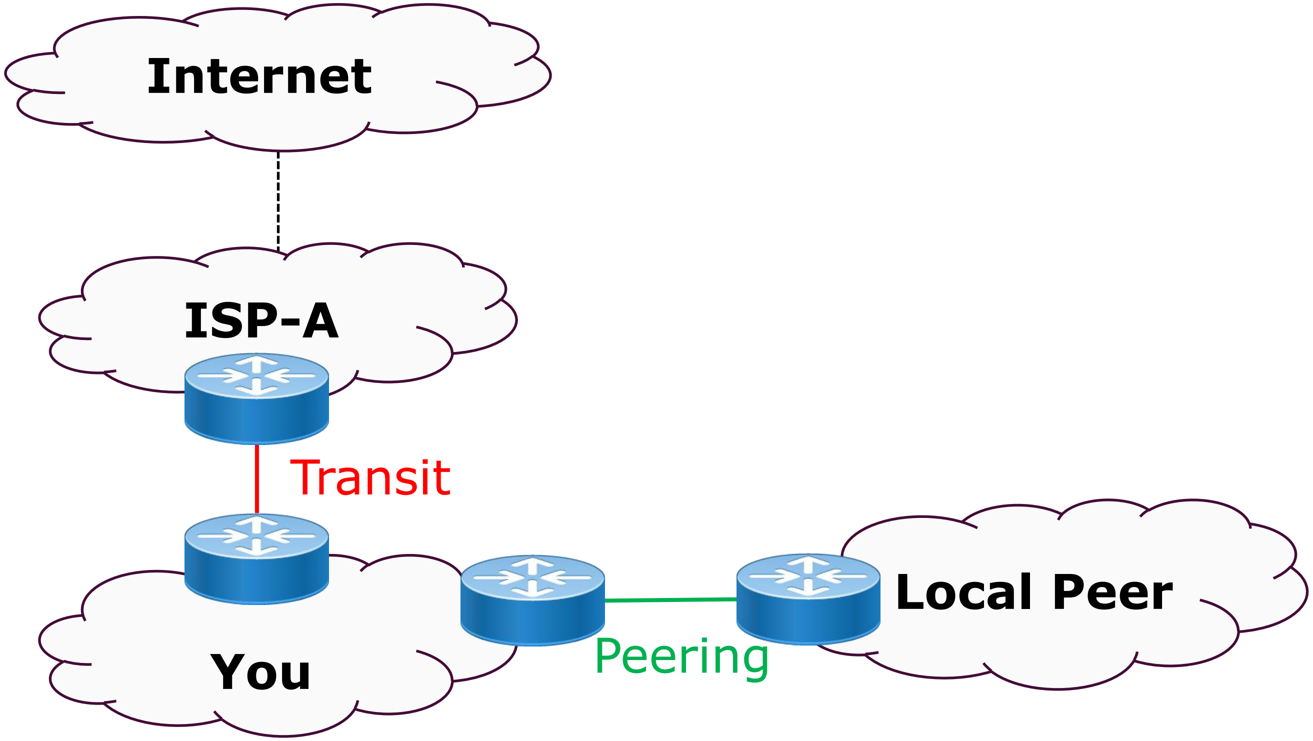
Achieving Redundancy - Multihoming

- More than one upstream ISP
  - Multi-homed



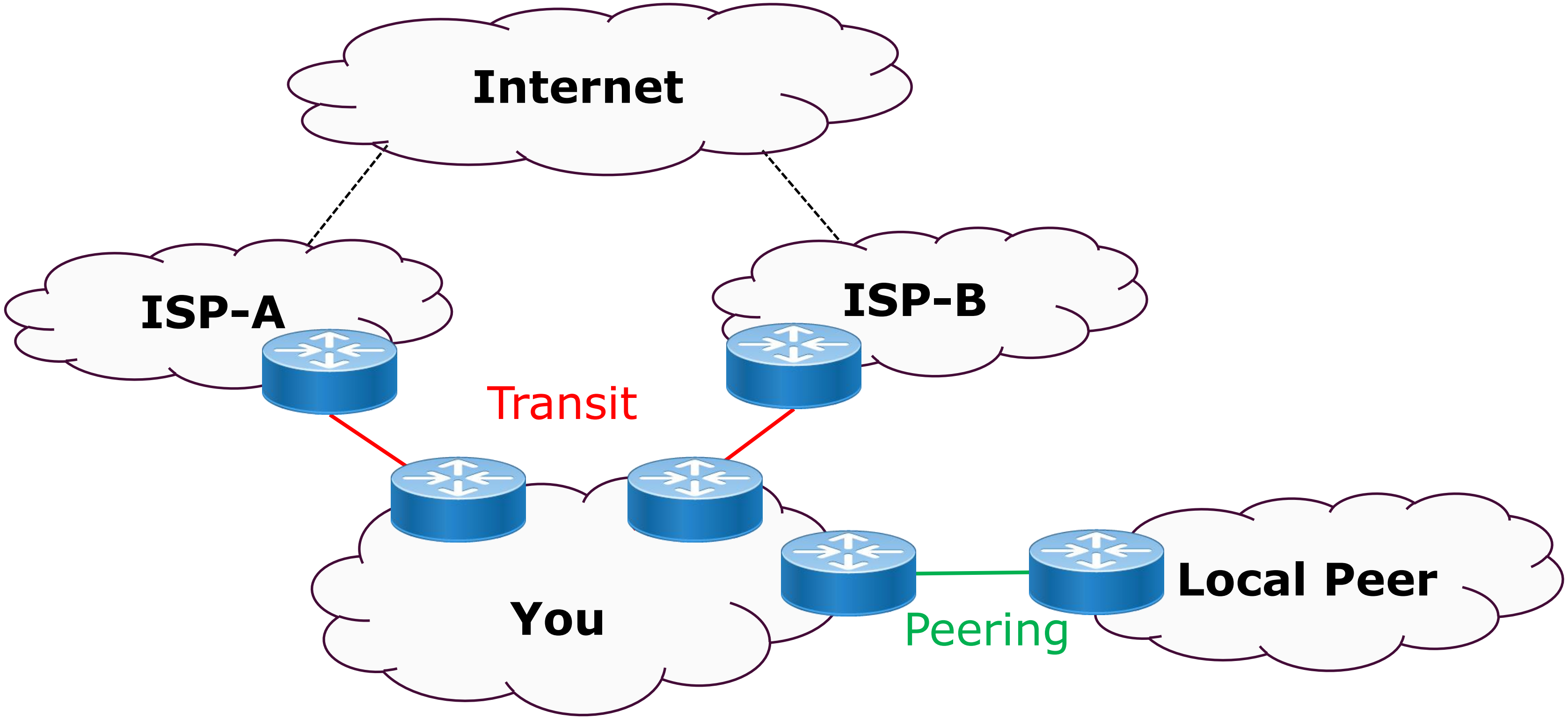
Multihoming – with peering

- One upstream and local peering



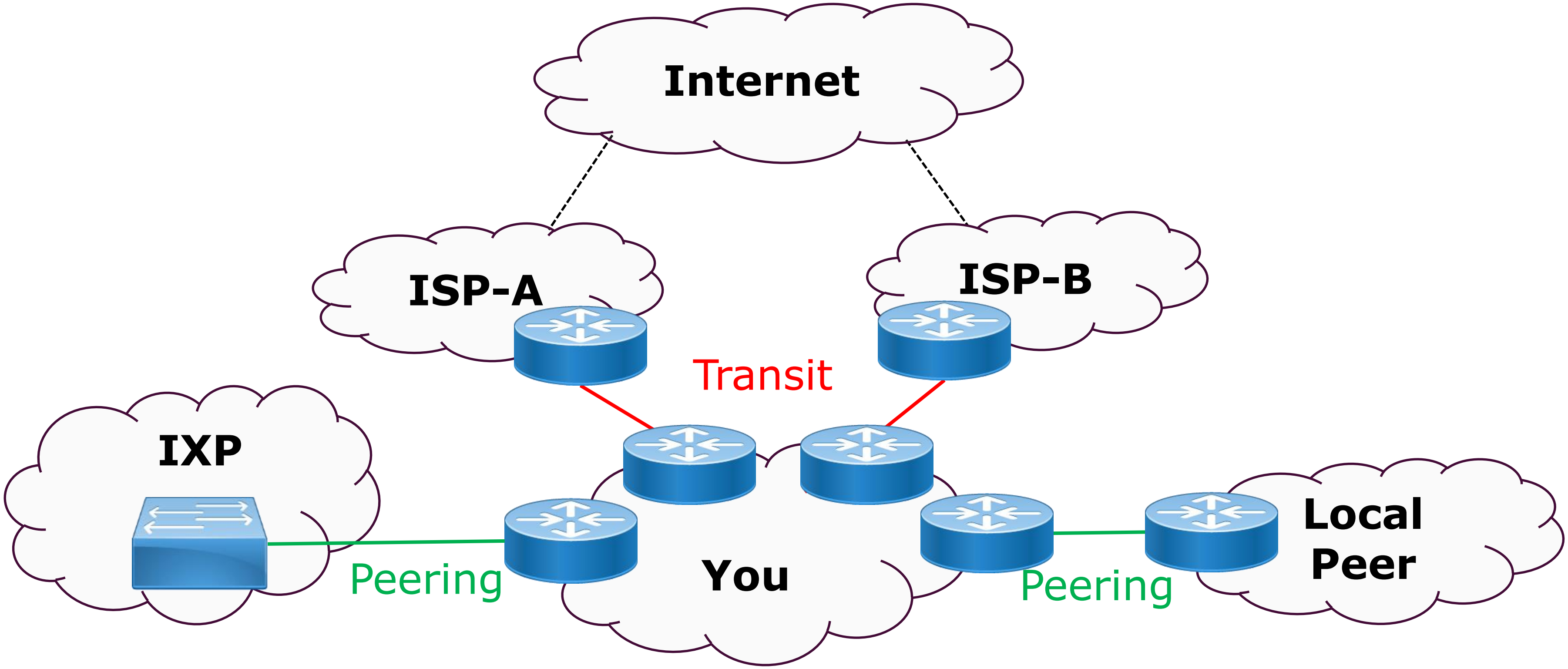
Multihoming

- More than one upstream ISP and local peering



Multihoming

- More than one upstream ISP with local and public peering



Traffic Engineering – Path Control

- Remember
  - the prefixes YOU ACCEPT control how YOU SEND traffic
  - Prefixes YOU ANNOUNCE can influence the (path) INCOMING traffic to you
    - There is no guarantee as the other party has control over sending traffic towards you

Traffic Engineering – Accept

- Just because someone sends you a prefix doesn’t mean you have to accept it

- Use ***Local Preference*** to control the best path of your outgoing traffic
  - it is the first BGP selection criterion!

Define Local Preference rules

- Follow the money
  - Prefer to send traffic to a customer (earns money)
  - If the route isn’t from a customer prefer an announcement seen from a peer (free)
  - Else send to a transit (costs money)
    - Might have a preference between transit providers

- Local preference values should follow this plan
  - Customer > Peer > Transit

Traffic Engineering – Announce

- You can use AS path prepending to try to influence your neighbour’s best path
    - but they may be using Local Preference too

- Often, more specific aggregates can help
  - announce them to the preferred path but not others
    - Remember to send the aggregate to everyone
    - Tag the more specific aggregates with the “no-export” community if possible
  - Be careful that you only do this to YOUR prefixes!

- Use MED when there are multiple links to the same AS
    - to signal a preferred path into your AS (may be ignored without prior negotiation)

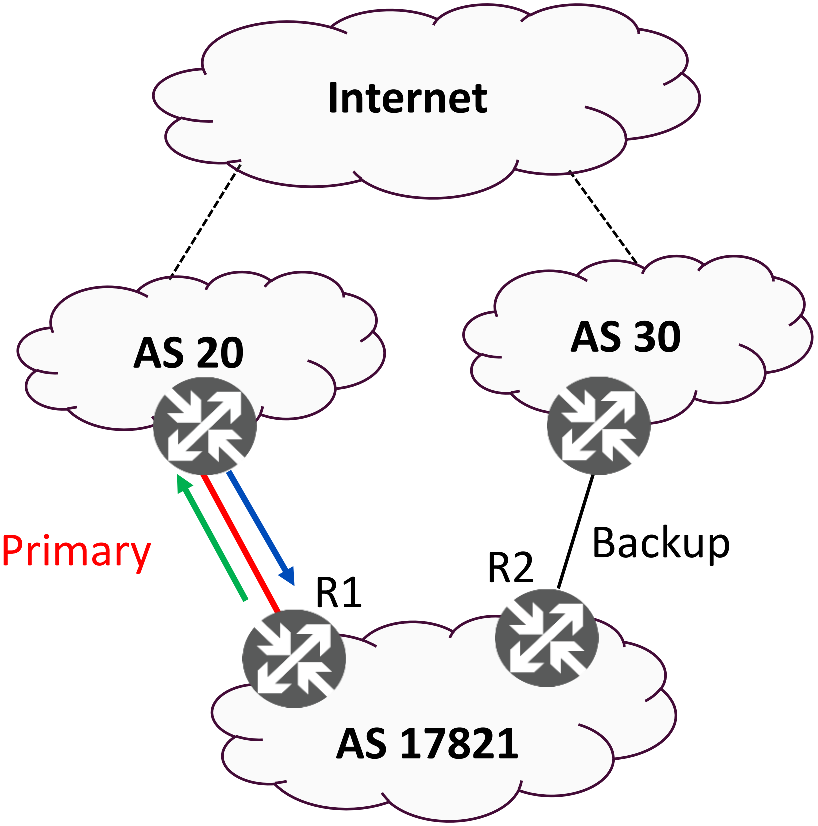
TE (Path control) Attributes

- Inbound Traffic:
  - AS-PATH, MED, Community (sub-aggregates/more specifics)

- Outbound Traffic:
  - Local Preference

Two Upstream – One backup

- Both incoming and outgoing traffic via AS20 (R1)
- AS30 (R2) path to be used only if the link to AS20 fails
  - AS_PATH to control inbound traffic
    - *Prepend outbound on R2*
  - LOCAL_PREF for outgoing traffic
    - *Higher for inbound routes on R1*



Two Upstream – One backup

- Always announce the aggregate on both links!
- **R1** (main link) config:

```
router bgp 17821
 network 61.45.248.0 mask 255.255.248.0
 neighbor 20.20.20.1 remote-as 20
 neighbor 20.20.20.1 prefix-list AGGR out
 neighbor 20.20.20.1 prefix-list DEF in
!
ip prefix-list AGGR permit 61.45.248.0/21
ip prefix-list DEF permit 0.0.0.0/0
!
ip route 61.45.248.0 255.255.248.0 null0
```

Advertise aggregate

Prefix-list applied to outbound routes

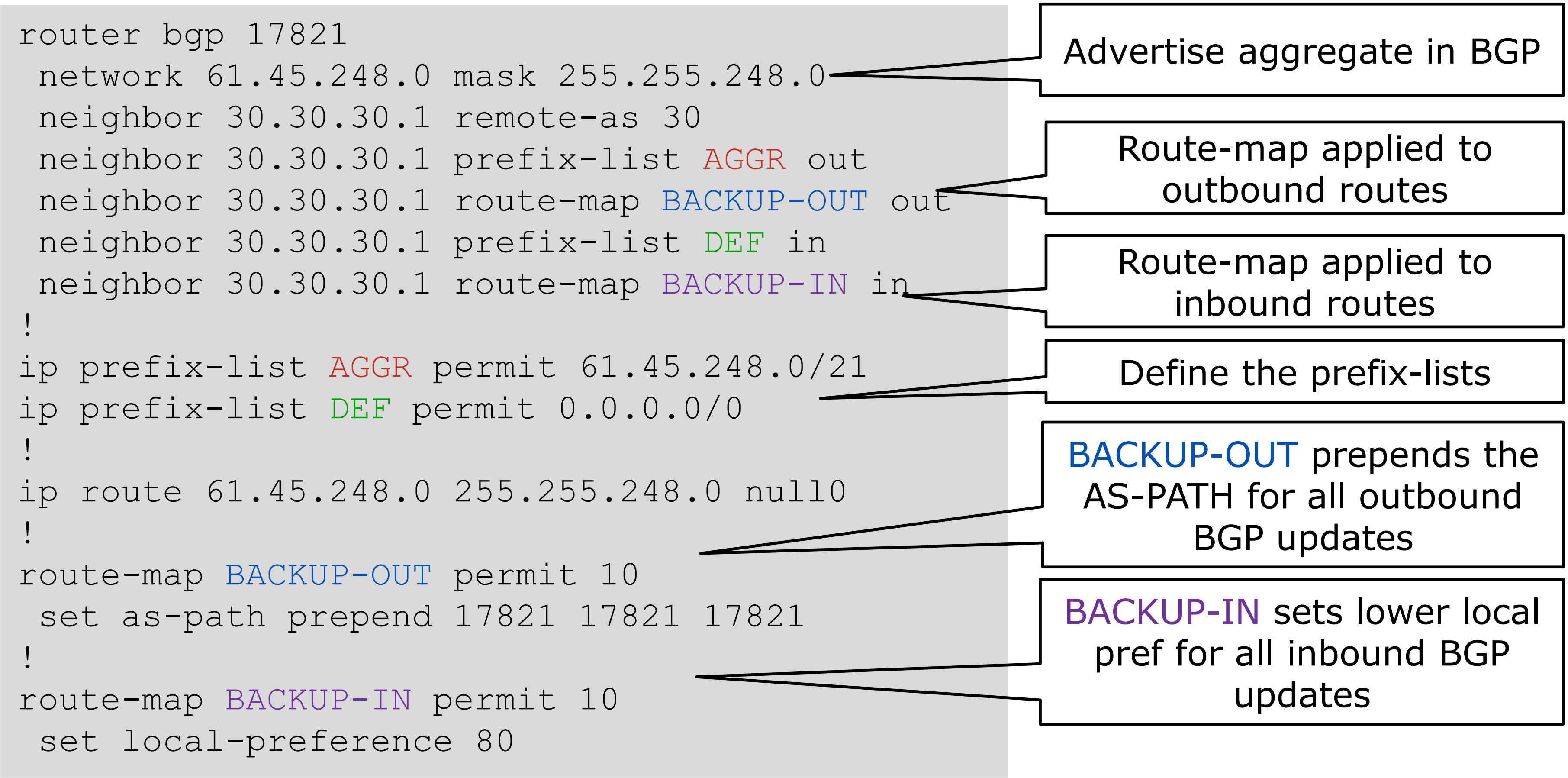
Prefix-list applied to inbound routes

Define the prefix-lists

Aggregate should exist in the routing table (pull-up route)

Two Upstream – One backup

- **R2** (backup) config:



```
router bgp 17821
 network 61.45.248.0 mask 255.255.248.0
 neighbor 30.30.30.1 remote-as 30
 neighbor 30.30.30.1 prefix-list AGGR out
 neighbor 30.30.30.1 route-map BACKUP-OUT out
 neighbor 30.30.30.1 prefix-list DEF in
 neighbor 30.30.30.1 route-map BACKUP-IN in
!
ip prefix-list AGGR permit 61.45.248.0/21
ip prefix-list DEF permit 0.0.0.0/0
!
ip route 61.45.248.0 255.255.248.0 null0
!
route-map BACKUP-OUT permit 10
 set as-path prepend 17821 17821 17821
!
route-map BACKUP-IN permit 10
 set local-preference 80
```

Advertise aggregate in BGP

Route-map applied to outbound routes

Route-map applied to inbound routes

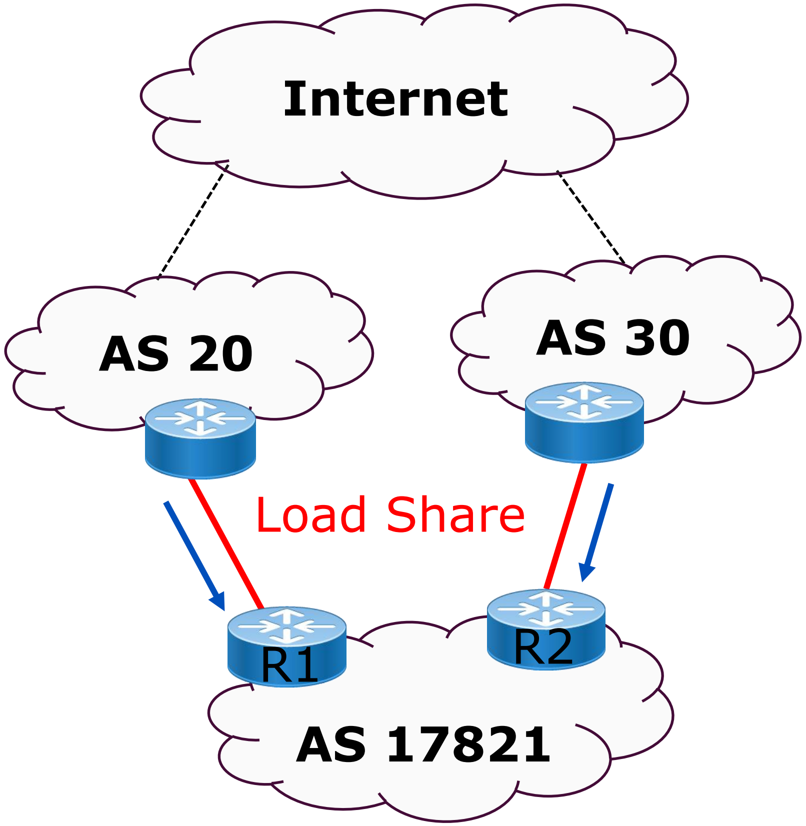
Define the prefix-lists

BACKUP-OUT prepends the AS-PATH for all outbound BGP updates

BACKUP-IN sets lower local pref for all inbound BGP updates

Two Upstream – Load Sharing (Inbound Traffic)

- Announce one sub-aggregate on first, and the other on the second link
  - Always announce the aggregate to both!
- Requires good address planning
  - Customers need to be assigned from both address blocks



Two Upstream – Load Sharing (Inbound Traffic)

- R1 config:

```
router bgp 17821
network 61.45.248.0 mask 255.255.248.0
network 61.45.248.0 mask 255.255.252.0
neighbor 20.20.20.1 remote-as 20
neighbor 20.20.20.1 prefix-list SUB-A out

!
ip prefix-list SUB-A permit 61.45.248.0/21
ip prefix-list SUB-A permit 61.45.248.0/22

!
ip route 61.45.248.0 255.255.248.0 null0
ip route 61.45.248.0 255.255.252.0 null0
```

Advertise both the aggregate and first sub-aggregate in BGP

Advertise sub-aggregate along with the aggregate

Sub-aggregate should exist in the routing table (pull-up route)

Two Upstream – Load Sharing (Inbound Traffic)

- R2 config:

```
router bgp 17821
network 61.45.248.0 mask 255.255.248.0
network 61.45.252.0 mask 255.255.252.0
neighbor 30.30.30.1 remote-as 30
neighbor 30.30.30.1 prefix-list SUB-B out

!
ip prefix-list SUB-B permit 61.45.248.0/21
ip prefix-list SUB-B permit 61.45.252.0/22

!
ip route 61.45.248.0 255.255.248.0 null0
ip route 61.45.252.0 255.255.252.0 null0
```

Advertise both aggregate and second sub-prefix in BGP

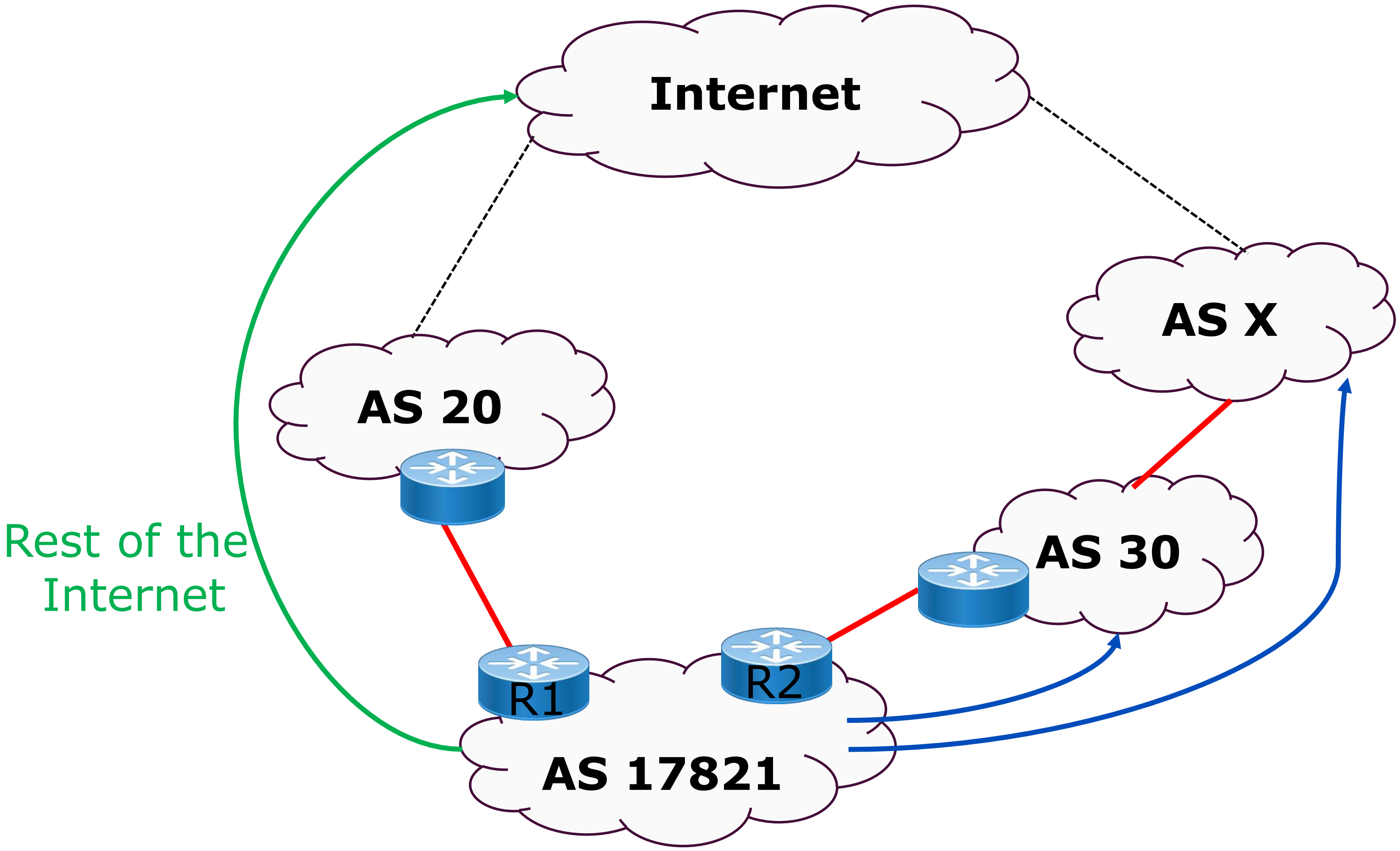
Advertise sub-aggregate along with the aggregate

Sub-aggregate should exist in the routing table (pull-up route)

Load Sharing – Outbound traffic (Full routes)

- Full Internet routes (more memory/CPU)
- Accept full route from both (AS20 and AS30)
  - For routes from AS30 (R2)
    - Higher LOCAL_PREF prefixes originated by AS30 and its immediate neighbors (one AS hop away) – traffic to those go via AS30
    - Lower LOCAL_PREF all other routes – traffic to these go via AS20
  - For routes learned from AS20 (R1)
    - default LOCAL_PREF value

Load Sharing – Outbound traffic (Full routes)



Load Sharing – Outbound traffic (Full routes)

- R1 configuration: nothing doing

```
router bgp 17821
 neighbor 20.20.20.1 remote-as 20
 neighbor 20.20.20.1 prefix-list ALL in
!
ip prefix-list ALL deny <bogons>
ip prefix-list ALL permit 0.0.0.0/0 le 24
!
```

Accept full internet feed
except bogons
(default LOCAL_PREF)

Load Sharing – Outbound traffic (Full routes)

- R2 config:

```
router bgp 17821
neighbor 30.30.30.1 remote-as 30
!
address-family ipv4
 neighbor 30.30.30.1 prefix-list ALL in
 neighbor 30.30.30.1 route-map TWO-HOPS in
!
ip prefix-list ALL deny <bogons>
ip prefix-list ALL permit 0.0.0.0/0 le 24
!
ip as-path access-list 30 permit ^(30_)+$
ip as-path access-list 30 permit ^(30_)+_[0-9]+$
!
route-map TWO-HOPS permit 10
 match as-path 30
 set local-preference 150
route-map TWO-HOPS permit 20
 set local-preference 50
```

Accept full internet feed except bogons

Accept routes local to and received from AS30 (AS-path prepend included)

Only routes from AS30 and its direct neighbor ASes

High-pref AS30 and its neighbor AS originated routes

Low-pref everything else

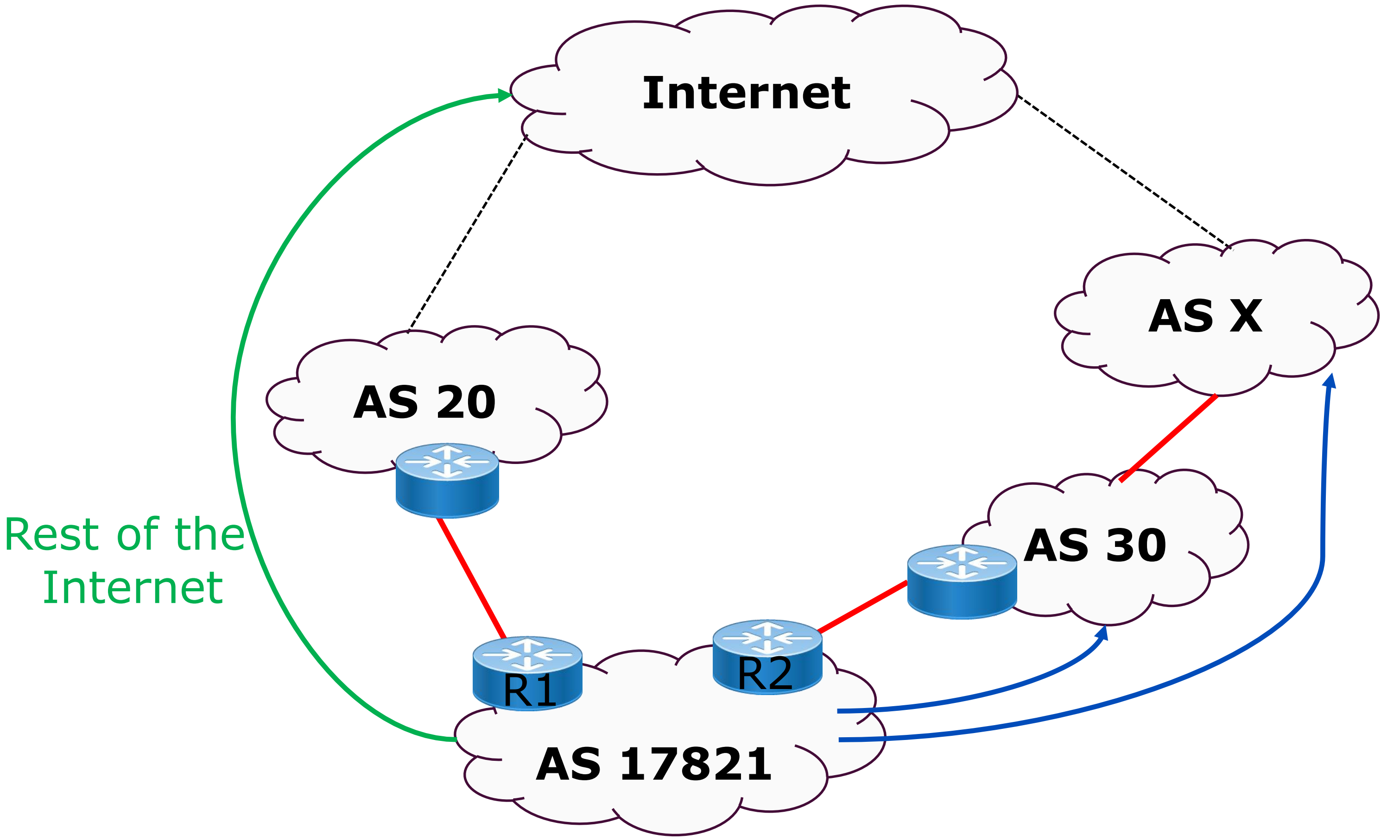
Load Sharing – Outbound Traffic (Partial routes)

- Partial Routes – less HW resources!

- Select some routes for special treatment
- Accept only *default* from AS20
- Default plus routes from AS30 (better connected than AS20)
  - Filter to only accept prefixes originated by AS30 and its neighbor ASes (AS-Path ACLs)
    - Higher LOCAL_PREF those routes
    - Low LOCAL_PREF the default route

  - Traffic to “rest of Internet” via AS 20

Load Sharing – Outbound Traffic (Partial routes)



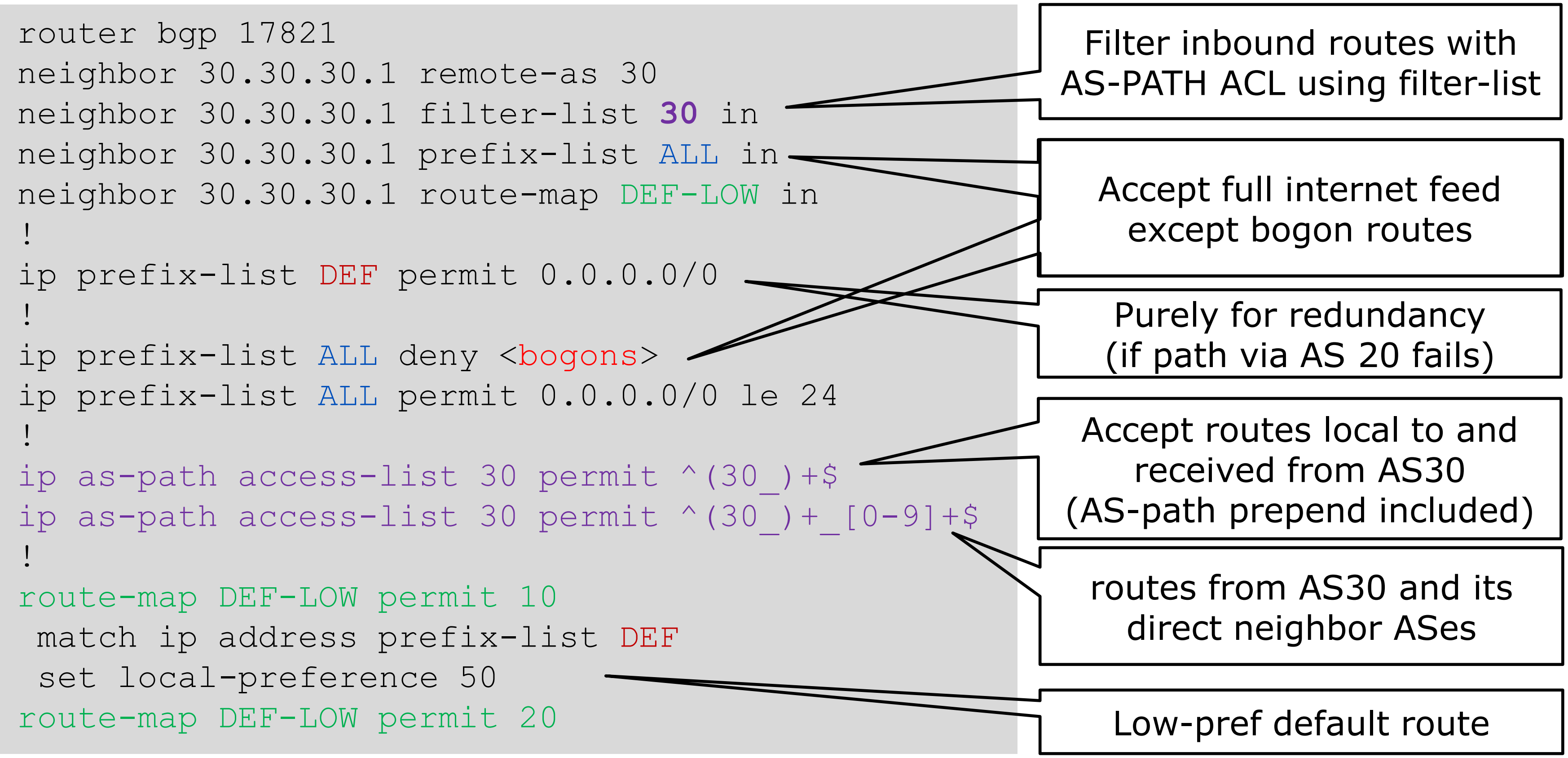
Load Sharing – Outbound Traffic (Partial routes)

- R1 configuration:

```
router bgp 17821
neighbor 20.20.20.1 remote-as 20
neighbor 20.20.20.1 prefix-list DEF in
!
ip prefix-list DEF permit 0.0.0.0/0
!
```

Load Sharing – Outbound Traffic (Partial routes)

- R2 config:



```
router bgp 17821
neighbor 30.30.30.1 remote-as 30
neighbor 30.30.30.1 filter-list 30 in
neighbor 30.30.30.1 prefix-list ALL in
neighbor 30.30.30.1 route-map DEF-LOW in
!
ip prefix-list DEF permit 0.0.0.0/0
!
ip prefix-list ALL deny <bogons>
ip prefix-list ALL permit 0.0.0.0/0 le 24
!
ip as-path access-list 30 permit ^(30_)+$
ip as-path access-list 30 permit ^(30_)+_[0-9]+$
!
route-map DEF-LOW permit 10
 match ip address prefix-list DEF
 set local-preference 50
route-map DEF-LOW permit 20
```

Filter inbound routes with AS-PATH ACL using filter-list

Accept full internet feed except bogon routes

Purely for redundancy (if path via AS 20 fails)

Accept routes local to and received from AS30 (AS-path prepend included)

routes from AS30 and its direct neighbor ASes

Low-pref default route

Using Communities

- Community attribute provides greater flexibility for traffic shaping than prefix-list
  - Simplifies BGP configuration
  - Greater policy control

- Not sent by default to BGP peers
  - Need to explicitly specify (`neighbor x.x.x.x send-community`)

- Can carry policy information
  - Example:
    - `ASN:80  (set local-pref 80)`
    - `ASN:1   (set as-path prepend ASN)`
    - `ASN:888 (set ip next-hop 192.0.2.1 – Cymru bogons)`

COMMUNITY recap

- Used to group prefixes (incoming/outgoing) and apply policies to the communities
  - A prefix can belong to more than one community
- Originally a 32-bit integer
  - Represented as two 16-bit integers [ASN:number]
    - Works well for 2-byte ASN
- With 4-byte ASNs
  - Common to see [private-ASN:number]
  - RFC 8092 (BGP Large Communities): 96-bit integer
    - [32-bit ASN:32-bit:32-bit]

Well-known Communities

- Many well-known communities defined
  - http://www.iana.org/assignments/bgp-well-known-communities/bgp-well-known-communities.xhtml

- The most commonly used:
  - no-export: do not advertise/export to any eBGP peers (RFC1997)
    - only extend to the neighbor AS
  - no-advertise: do not advertise to any BGP peers (RFC1997)
  - no-peer: do not advertise to any bilateral peer (RFC3765)
  - blackhole: null route the prefix (RFC7999)
    - Trigger blackholing
    - Signal neighboring ASes (upstream) to drop any traffic being sent towards this prefix (victim's IP addresses)

Example – IXP Route Server Communities

`0:(IXP-AS)`	Do not announce any peer
`0:(PEER-AS)`	Do not announce prefixes to certain peer
`(IX-AS):(PEER-AS)`	Advertise to a certain peer
`(IX-AS):(IX-AS)`	Advertise prefixes to all peers

Example - Equinix Community Usage

- Default Open, except with AS10, AS20, and AS30

```
router bgp 17821
  neighbor 202.79.197.126 remote-as 24115
  neighbor 202.79.197.126 route-map eqixsg-in in
  neighbor 202.79.197.126 route-map eqixsg-out out

# set default-open community on outbound and restrict
# announcements to AS10, 20 and 30

route-map eqixsg-out permit 10
 set community 24115:24115 0:10 0:20 0:30

# reject routes received from AS 10, 20 and 30
route-map eqixsg-in deny 10
 match as-path 10

ip as-path access-list 10 permit ^(10|20|30)_
```

Using communities for filters

- Set a community when you import a route from a customer or create a static (aggregate) route
  - Use that community to control export to peers & transit
  - Don’t allow peers or transits to set it though

- Now when you add a prefix on a router it will automatically get exported on other routers without updating their prefix lists

Setting Communities (IOS)

```
router bgp 17821
<output omitted>
!
address-family ipv4 unicast
 network 61.45.248.0 mask 255.255.248.0 route-map SET-COMM-AGG
 network 61.45.248.0 mask 255.255.254.0 route-map SET-COMM-4G
 network 61.45.250.0 mask 255.255.254.0 route-map SET-COMM-BB
 network 61.45.252.0 mask 255.255.254.0 route-map SET-COMM-ENT
 network 61.45.254.0 mask 255.255.254.0 route-map SET-COMM-CORP
!
ip route 61.45.248.0 255.255.248.0 null0
ip route 61.45.248.0 255.255.254.0 null0 254
ip route 61.45.250.0 255.255.254.0 null0 254
ip route 61.45.252.0 255.255.254.0 null0 254
ip route 61.45.254.0 255.255.254.0 null0 254
!
```

Setting Communities (IOS)

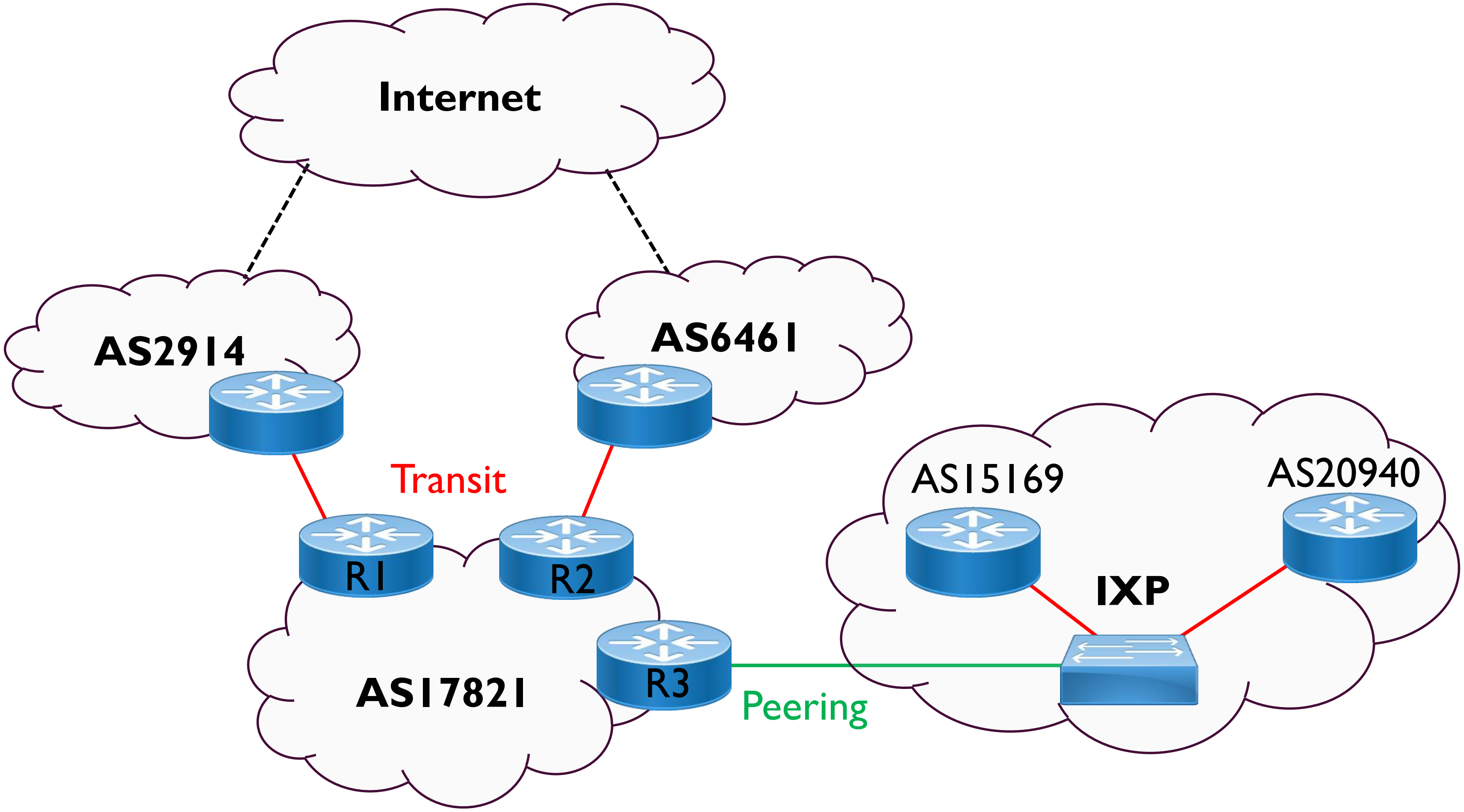
```
route-map SET-COMM-AGG permit 10
 set community 17821:1000
!
route-map SET-COMM-4G permit 10
 set community 17821:1101
!
route-map SET-COMM-BB permit 10
 set community 17821:1102
!
route-map SET-COMM-ENT permit 10
 set community 17821:1103
!
route-map SET-COMM-CORP permit 10
 set community 17821:1104
!
```

Grouping Communities (IOS)

- We can group communities together with community-list:

```
!
ip community-list 20 permit 17821:1000
ip community-list 21 permit 17821:1101
ip community-list 22 permit 17821:1102
ip community-list 23 permit 17821:1103
ip community-list 24 permit 17821:1104
!
```

Two Transits and IXP



Two Transits and IXP – IXP Router

- R3 (IXP) configuration:
  - For both incoming and outgoing traffic - IXP should be the preferred path!

```
router bgp 17821
 neighbor IX-PEERS peer-group
 neighbor 80.81.192.108 remote-as 15169
 neighbor 80.81.192.108 peer-group IX-PEERS
 neighbor 80.81.192.108 description Google
 neighbor 80.81.192.283 remote-as 20940
 neighbor 80.81.192.283 peer-group IX-PEERS
 neighbor 80.81.192.283 description Akamai
!
 address-family ipv4
  neighbor IX-PEERS send-community
  neighbor IX-PEERS remove-private-as
  neighbor IX-PEERS route-map IX-IN in
  neighbor IX-PEERS route-map IX-OUT out
```

Define peer-groups for all IX peers

Add neighbors to the peer group

Define common policies applied to all neighbors on the peer-group
- Send communities
- Remove private ASNs

Apply inbound and outbound routing policies

Two Transits and IXP – IXP Router

- R3 (IXP) configuration (contd..):

```
!
ip community-list 20 permit 17821:1000
ip community-list 21 permit 17821:1101
ip community-list 22 permit 17821:1102
ip community-list 23 permit 17821:1103
ip community-list 24 permit 17821:1104
!
route-map IX-IN permit 10
 set local-preference 250
 set community 17821:<IX-AS> add
!
route-map IX-OUT permit 10
 match community 20 21 22 23 24
 set metric 10
!
```

Define the communities

High-pref routes received from IX peers (outbound traffic via IX)

Define a community for all routes learned via IXP

Send all our prefixes (aggregates and sub-aggregates)

Set lower MED for all routes sent to IX peers (inbound traffic via IX)

Location info via communities

- For Transit/Upstream:
  - Tier-1 ISPs (or ISPs who are run properly) use communities to group their regional prefixes
  - Filter based on those to shape outbound traffic to Internet!
    - Ex: receive US routes from one ISP, and Europe routes from the other
  - Example:
    - https://www.us.ntt.net/support/policy/routing.cfm
    - NTT US – 2914:3000
    - NTT Europe – 2914:3200
    - NTT Asia – 2914:3400
    - NTT South America – 2914:3600

Balancing between two transits

- For Inbound traffic:
  - We can use our sub-prefixes to balance incoming traffic
  - Ex: Advertise half of our routes to one, and the other half to the other
    - keep playing until we reach symmetry!
  - But remember to announce the aggregates to both (REDUNDANCY!)
  - If very small/more specifics, use “no-export” to avoid polluting the global routing table
    - And avoid the wrath of “network police” ☺

Two Transits – Different policies

- R1 configuration:
  - Let us assume NTT (AS2914) as transit here

```
router bgp 17821
 neighbor 29.29.29.1 remote-as 2914
 neighbor 29.29.29.1 description eBGP with NTT
!
 address-family ipv4
  neighbor 29.29.29.1 send-community
  neighbor 29.29.29.1 route-map NTT-IN in
  neighbor 29.29.29.1 route-map NTT-OUT out
!
! We want Asia, US and SA routes
ip community-list 1 permit 2914:3000 !US
ip community-list 1 permit 2914:3400 !AS
ip community-list 1 permit 2914:3600 !SA
ip community-list 2 permit 2914:3200 !EU
```

- Send communities
- Apply inbound and outbound routing policies

Define communities for NTT global routes

- In this example, we will source US and Asia routes from NTT

Two Transits – Different policies

- R1 configuration (contd..):

```
route-map NTT-IN permit 10
 match community 1
 set local-preference 210
route-map NTT-IN permit 20
 set local-preference 50
!
route-map NTT-OUT permit 10
 match community 20
 match community 21
 match community 22
!
```

Route-map to influence outbound traffic

- Set higher LOCAL_PREF for US, Asia, and SA routes (outbound traffic)
- Still lower than IX!

Lower LOCAL_PREF for EU/rest of routes (will prefer the second ISP, but available if that link fails)

Route-map to influence inbound traffic

- Send our aggregate (in case ISP2 fails)
- And half of our sub-prefixes (*can apply no-export)

Two Transits – Different policies

- R2 configuration:
  - Let us assume Zayo (AS6461) as transit here

```
router bgp 17821
 neighbor 64.64.64.1 remote-as 6461
 neighbor 64.64.64.1 description eBGP with Zayo
!
 address-family ipv4
  neighbor 64.64.64.1 send-community
  neighbor 64.64.64.1 route-map ZAYO-IN in
  neighbor 64.64.64.1 route-map ZAYO-OUT out
!
! Zayo Europe routes
ip community-list 3 permit 6461:5996
ip community-list 3 permit 6461:5998
ip community-list 3 permit 6461:5999
! Zayo Global routes
ip community-list 4 permit 6461:5997
```

- Send communities
- Apply inbound and outbound routing policies

Define communities for Zayo global routes

- In this example, we will source EU routes from Zayo

Two Transits – Different policies

- R2 configuration (contd..):

```
route-map ZAYO-IN permit 10
 match community 3
 set local-preference 210
route-map ZAYO-IN permit 20
 set local-preference 50
!
route-map ZAYO-OUT permit 10
 match community 20
 match community 23
 match community 24
!
```

Route-map to influence outbound traffic

- Set higher LOCAL_PREF for EU routes (outbound traffic)
- Still lower than IX!

Lower LOCAL_PREF for global routes (NTT is preferred, but will work if that link fails)

Route-map to influence inbound traffic

- Send our aggregate (in case ISP1 fails), and
- other second-half of our sub-prefixes (*can apply no-export)

Acknowledgement:

- Philip Smith
- Cisco Systems

Questions