ISP and NSP Security Workshop

APRICOT 2013

Day 2

Infrastructure Security

Planes

Planes

- Management Plane
- Control Plane
- Forwarding Plane (or Data Plane)

Management Plane

- Refers to the management of the router
  - Telnet/SSH
  - SNMP
  - FTP/SCP/SFTP
  - NetConf
  - DHCP, RADIUS, TACACS+, NTP, SYSLOG…
- Increasingly, routers offer a separate Ethernet interface that is tied to the management plane only in addition to the serial management interface
  - *Out of Band*
- Management plane can also be in-band
  - Needs to be protected so that only authorised traffic can make it to the management plane
- Typically not designed for high speed packet forwarding – network management ONLY

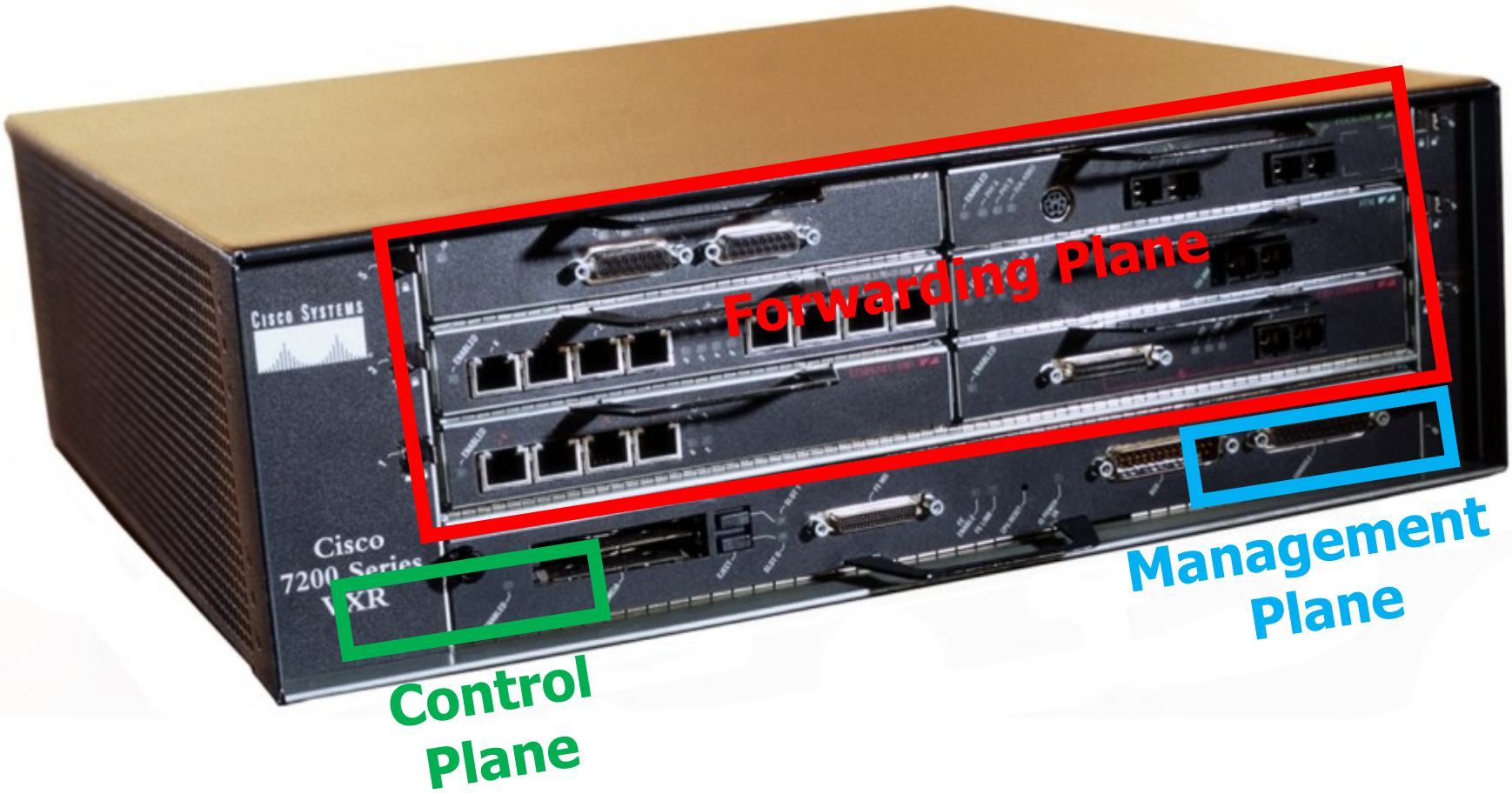
Control Plane

- Refers to the control of routing and forwarding
- A fairly critical function!
  - OSPF
  - ISIS
  - BGP
  - LDP
  - RSVP-TE
- Is always run in-band on the router
- Like the management plane, needs to be protected
- Often, a relatively slow CPU, maybe with a slow bus too

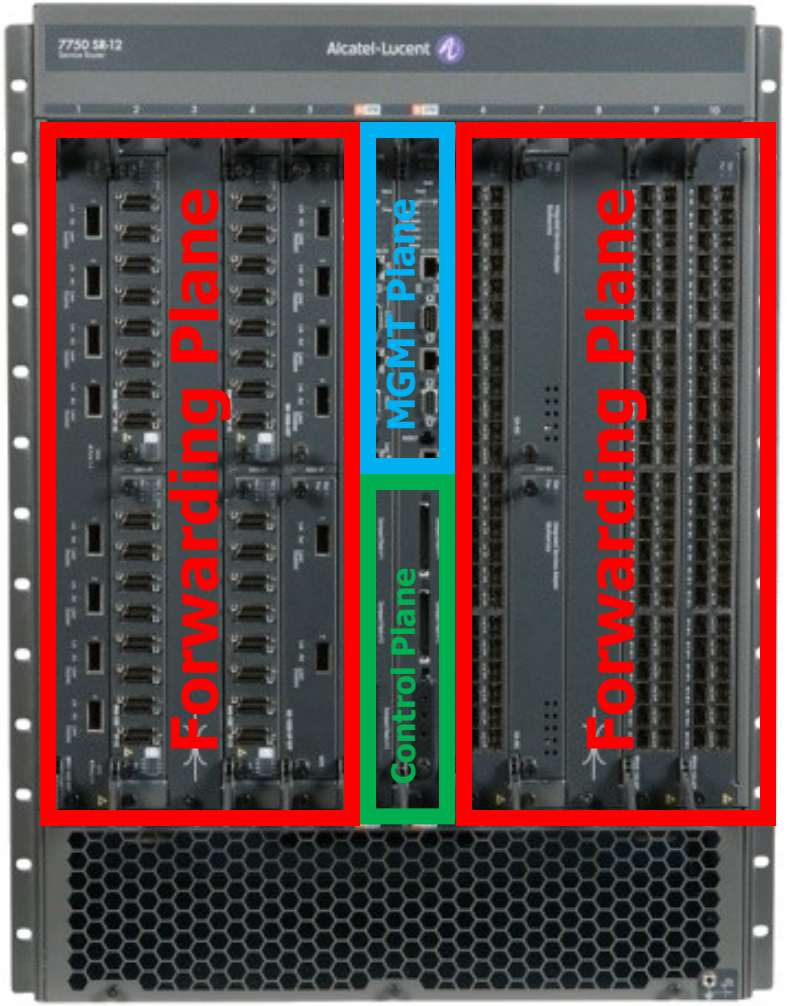
Forwarding Plane

- The actual forwarding of packets
- Also referred to as the data plane
- High speed custom forwarding engines for packet processing
  - Network Processing Units (NPUs), usually FPGA or ASIC
- Generally doesn't need protecting since it's designed to forward packets!

Planes in a router



Planes in a router



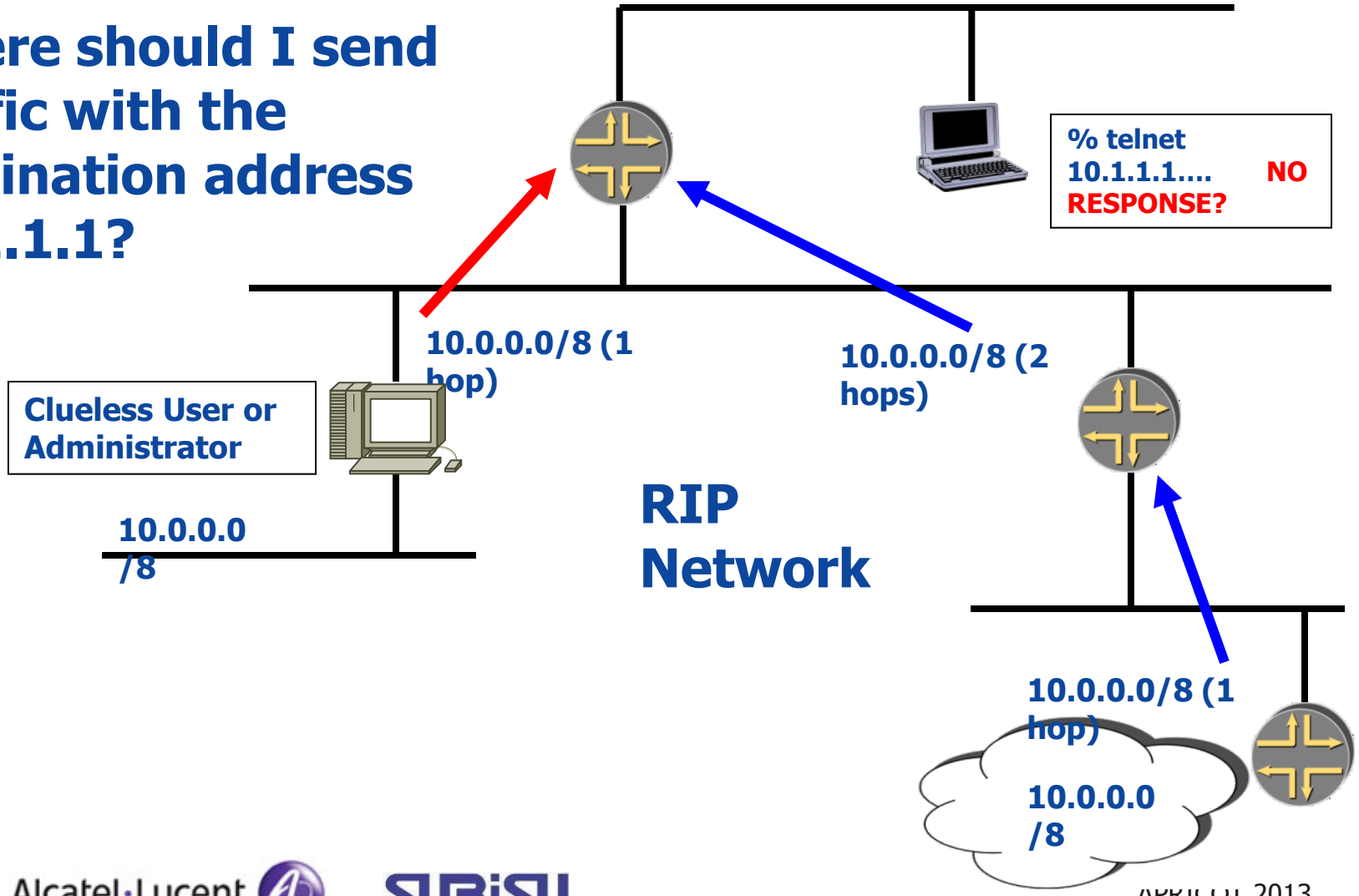
ROUTING SECURITY

The Need for Secure Routing

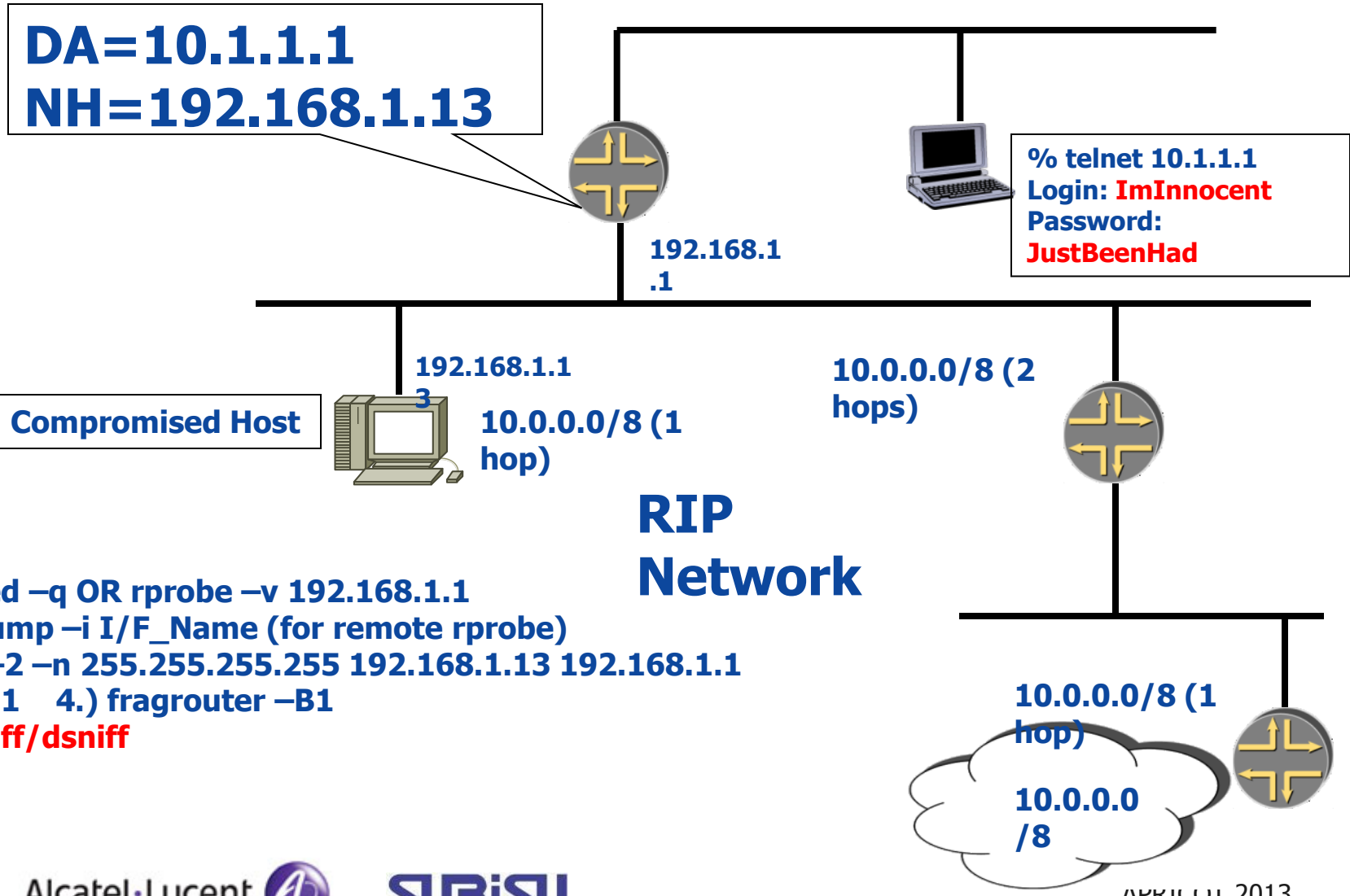
- Q: What is the most important function performed by routers?
  - A: They route!
- Q: What happens when you lose control of this function?
  - A: A compromised router or routing protocol can result in problems that run the gambit from simple loss of connectivity to a compromise of security!

Routing Gone Bad—Example 1

Where should I send traffic with the destination address 10.1.1.1?



Routing Gone Worse—Example 2



Routing Protocol Security

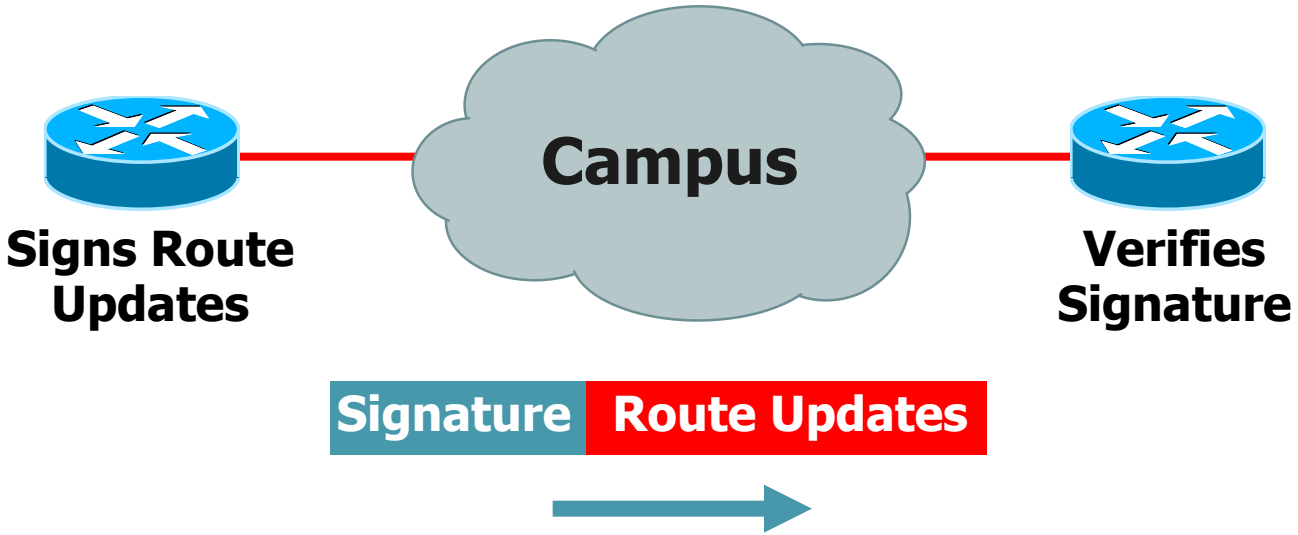
Routing Protocols Can Be Attacked

- Denial of service
- Smokescreens
- False information
- Reroute packets

May Be Accidental or Intentional

Secure Routing—Route Authentication

Configure Routing Authentication



Certifies Authenticity of Neighbor and Integrity of Route Updates

Route Authentication

- Shared key included in routing updates
  - Plain text—protects against accidental problems only
  - Message Digest 5 (MD5)—protects against accidental and intentional problems
- Multiple keys supported
- Supported for BGP, IS-IS, OSPF, RIPv2, and EIGRP
- Update keys before protocol timeout to avoid session bounce
- Often non-implemented
  - "Never seen an attack"
  - "My peer doesn't use it"

Routing Protocol Security

- The case for using routing protocol authentication:
  - RIP is not the only protocol susceptible to this attack
    - OSPF, IS-IS, and BGP are all trusting by nature
  - Implementations of routing protocols are widely available for many operating systems
    - Easy to install on a compromised host
    - Easy to implement by a disgruntled employee
    - Easy to install by a well-meaning—but clueless—insider
  - Secure your routing protocols such that only authenticated neighbors and peers can participate
  - Limit your usage of routing protocols to only the links that must be included

Agenda: Securing RIP

- Routing Protocol Authentication
- ➔ Securing RIP
- Securing OSPF
- Securing IS-IS
- Securing BGP
- Additional Routing Security

RIP Authentication

- RIP authentication only available when using RIPv2
  - Three types of authentication are available:
    - None (default)
    - Simple (clear text)
    - MD5
  - Using MD5 authentication, a trailer is added to each RIPv2 packet
    - Contains a keyed MD5 hash of the packet contents and a shared key
    - Provides integrity, but not availability or confidentiality
  - Two places to configure RIPv2 authentication:
    - Globally, for all RIPv2 neighbors (interfaces)
    - On an individual interface (neighbor level)

RIP Configuration

```
configure terminal
 key chain ripkeys
       key 1
             key-string rip
 interface Ethernet 1/1
       ip rip authentication mode md5
       ip rip authentication key-chain ripkeys
```

In this example, MD5 authentication is enabled for Ethernet 1/1, running RIPv2

Agenda: Securing OSPF

- Routing Protocol Authentication
- Securing RIP
- ➔ Securing OSPF
- Securing IS-IS
- Securing BGP
- Additional Routing Security

OSPF Authentication, Configuration

- Authentication occurs within an individual area
  - Two types are supported: none and MD5
- Each interface requires an authentication key
  - Multiple interfaces can use the same key
- By default, the authentication type is set to none
  - Effectively means no authentication is performed

```
configure terminal
        router ospf <ASN>
        area <ASN> authentication message-digest
        interface eth1/1
                ip ospf message-digest-key 1 md5 <key>
```

MD5 Authentication Configuration

- Using MD5 authentication, a message digest is generated and appended to the end of each OSPF packet
  - Contains a keyed MD5 hash of the packet contents and a shared key
  - Provides *integrity*, but not *availability* or *confidentiality*
- Each interface requires an authentication key
  - Multiple interfaces can use the same key
  - Keys are always encrypted in the configuration

Agenda: Securing IS-IS

- Routing Protocol Authentication
- Securing RIP
- Securing OSPF
- ➔ Securing IS-IS
- Securing BGP
- Additional Routing Security

IS-IS Authentication

- Authentication can occur within multiple places
  - Level 1
  - Level 2
  - Interface
- Three authentication types are supported
  - None (default)
  - Simple
  - MD5
- Using HMAC-MD5 authentication, TLV 10 is included in each IS-IS PDU
  - TLV contains an HMAC-MD5 hash of the packet contents and a shared key
  - Provides integrity, but not availability or confidentiality

Authentication Configuration

- Level authentication affects all IS-IS PDUs
  - Link-state, sequence number, and hello
- Per-interface authentication takes precedence over per-level settings

```
configure terminal
        interface ethernet 1/1
                isis authentication mode <md5 | text> [level-1] [level-2]
                isis key-chain <name-of-key-chain> [level-1] [level-2]
                isis password <password> [level-1] [level-2]
```

OSPF and ISIS Authentication Example

OSPF

```
configure terminal
   router ospf <ASN>
        area 0 authentication message-digest
   interface eth1/1
        ip ospf message-digest-key 1 md5 <key>
```

ISIS

```
configure terminal
   key chain isis-key
    key 1
        key-string <key>
   interface eth1/1
        isis authentication mode md5 [level-1] [level-2]
        isis authentication key-chain lab-key [level-1] [level-2]
```

Agenda: Securing BGP

- Routing Protocol Authentication
- Securing RIP
- Securing OSPF
- Securing IS-IS
- ➔ Securing BGP
- Additional Routing Security

BGP Authentication

- BGP authentication:
  - Two types of authentication available
    - None (default)
    - MD5
  - Using MD5 authentication, an extension is included in each TCP segment
    - Contains a 16-byte MD5 hash of the packet contents and a shared key
    - Provides *integrity*, but not *availability* or *confidentiality*
- Places to apply BGP authentication:
  - For all peers in an individual group
  - For a single peer

BGP Authentication

```
configure terminal
    router bgp <ASN>
       neighbor N.N.N.N password <key>
```

BGP Route Authentication

```
router bgp 200
 no synchronization
 neighbor 4.1.2.1 remote-as 300
 neighbor 4.1.2.1 description Link to
  Excalibur
 neighbor 4.1.2.1 send-community
 neighbor 4.1.2.1 version 4
 neighbor 4.1.2.1 soft-reconfiguration
  inbound
 neighbor 4.1.2.1 route-map Community1 out
 neighbor 4.1.2.1 password 7 q23dc%$#ert
```

BGP Route Authentication

- Works per neighbor or for an entire peer-group
- Two routers with password mismatch:
  - %TCP-6-BADAUTH: Invalid MD5 digest from [peer's IP address]:11004 to [local router's IP address]:179
- One router has a password and the other does not:
  - %TCP-6-BADAUTH: No MD5 digest from [peer's IP address]:11003 to [local router's IP address]:179

LAB

Lab Exercise 1

- Enable OSPF authentication on your two PE routers
  - Use the key apricot
- Check and ensure your OSPF sessions are working and routes are exchanged
- Is everything else still working?

Verifying Authentication

- Authentication information available with the `show ospf interface detail` command
  - Type of authentication is displayed
  - Key ID values shown if appropriate

```
Router16#show ip ospf interface et1/0
Ethernet1/0 is up, line protocol is up
Internet Address 10.0.18.2/30, Area 0
Process ID 80, Router ID 10.0.18.252, Network Type BROADCAST, Cost: 10
 Enabled by interface config, including secondary ip addresses
   Transmit Delay is 1 sec, State DR, Priority 1
   Designated Router (ID) 10.0.18.252, Interface address 10.0.18.2
   Backup Designated router (ID) 10.0.18.251, Interface address 10.0.18.1
   Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
     oob-resync timeout 40     Hello due in 00:00:02
   Supports Link-local Signaling (LLS)
   Index 2/2, flood queue length 0
   Next 0x0(0)/0x0(0)
   Last flood scan length is 1, maximum is 1
   Last flood scan time is 0 msec, maximum is 4 msec
   Neighbor Count is 1, Adjacent neighbor count is 1
     Adjacent with neighbor 10.0.18.251  (Backup Designated Router)
   Suppress hello for 0 neighbor(s)
```

Lab Exercise 2

- Enable BGP authentication on your two PE routers
  - Use the key apricot
- Check and ensure your BGP sessions are working and routes are exchanged
- Is everything else still working?

Verifying BGP Authentication

```
*Feb 20 10:10:09.175: %BGP-5-ADJCHANGE: neighbor 10.0.18.252
Down User reset

*Feb 20 10:10:09.191: %TCP-6-BADAUTH: No MD5 digest from
10.0.18.252(179) to 10.0.18.251(38215) (RST) *Feb 20
10:10:11.187: %TCP-6-BADAUTH: No MD5 digest from
10.0.18.252(179) to 10.0.18.251(38215)

*Feb 20 10:11:47.115: %BGP-5-ADJCHANGE: neighbor 10.0.18.252 Up

show router bgp neighbor 10.0.18.252
 <lots of text trimmed>
SRTT: 216 ms, RTTO: 957 ms, RTV: 741 ms, KRTT: 0 ms
minRTT: 172 ms, maxRTT: 592 ms, ACK hold: 200 ms
Flags: active open, nagle, md5
IP Precedence value : 6
```

Lab Exercise 3

- Configure an **E**BGP session between your PE-CE routers
  - Use your group AS+1 and your AS+2 for the CE ASN
    - E.g. AS11 on CE1, AS12 on CE2, AS21, AS22.
  - Use authentication
    - Decide on a key between yourselves
  - EBGP uses the interface addresses, not loopback!
- Is BGP exchanging routes?
- Is everything else still working?
- If it's all working – **COFFEE BREAK!**

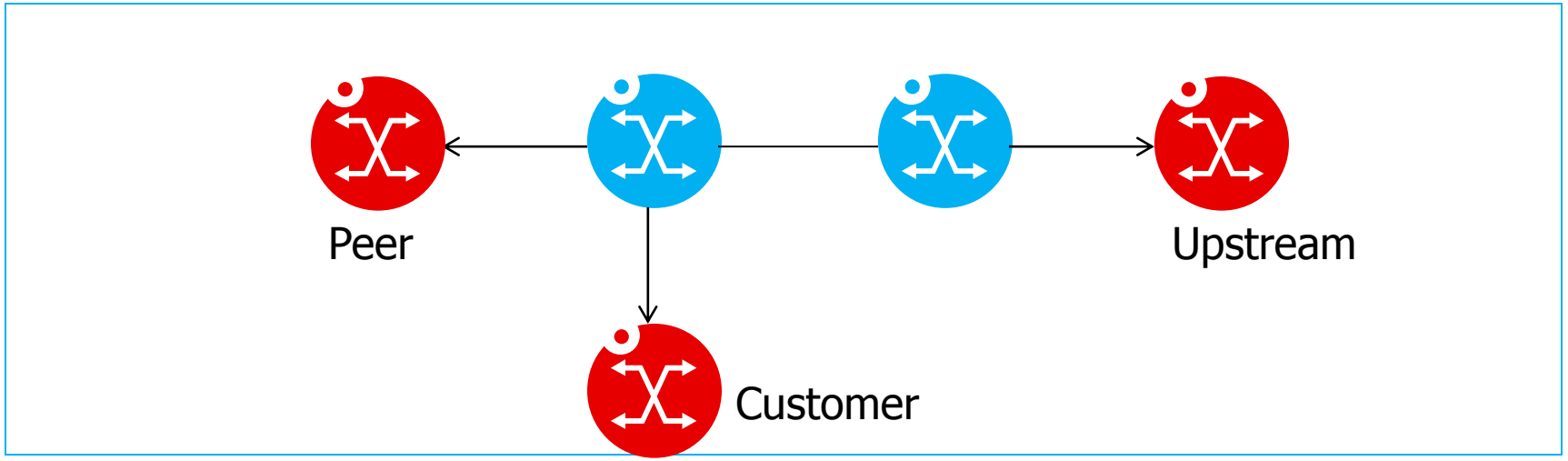
Lab Exercise 3.1

- What did you see?
  - PE advertises routes to CE
    - Your /24 prefixes in which the loopbacks are present
  - CE is advertising no routes to the PE
- Inject your loopback route into BGP
  - network 192.168.XX.1 mask 255.255.255.255
- Do you now see it on the PE via BGP?

Lab Exercise 3.2

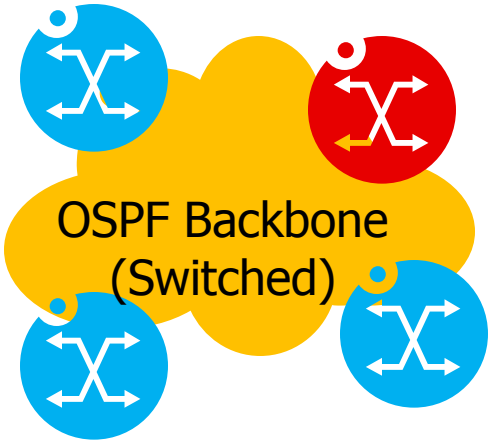
- Configure a default route on the PE-CE BGP session
  - neighbor X.X.X.X default-originate
- Delete the static-routes for the CE loopbacks on the PE routers
  - no ip route 192.168.XX.1 255.255.255.255
- Is everything working? (CE to CE pings)

Passive Interface Default



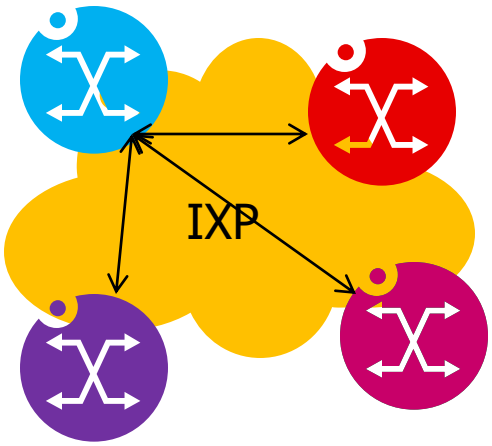
- Without passive interface default, **all** IP interfaces on a router will participate in the IGP instance
- Setting passive interface default and then explicitly configuring IGP-speaking interfaces is a better idea
- Other vendors require explicit interface configuration as normal approach

The case for protocol authentication



- In the simple network shown above, three blue routers are participating in OSPF without authentication. All routers are on a single L2 network (e.g. a single switch, or common VLAN)
- The malicious red router can join the OSPF instance and receive, and inject, routes

The case for protocol authentication



- At an IXP it is common to configure bi-lateral BGP sessions between participants
- If a participant leaves an exchange, often their IP address will be reused by a new joining member
- Old BGP sessions may be hijacked by the new member!

Common routing protocol misconfigurations

- IGP:
  - Interfaces enabled towards customers and peers
    - Always explicitly enable IGP only on internal interfaces!
    - Use protocol authentication
    - Audit this regularly – use config management tools
  - L2 domains (e.g. VLAN) has non-trusted devices present in it
    - Enable protocol authentication
    - Minimize use of multipoint L2 topologies between routers

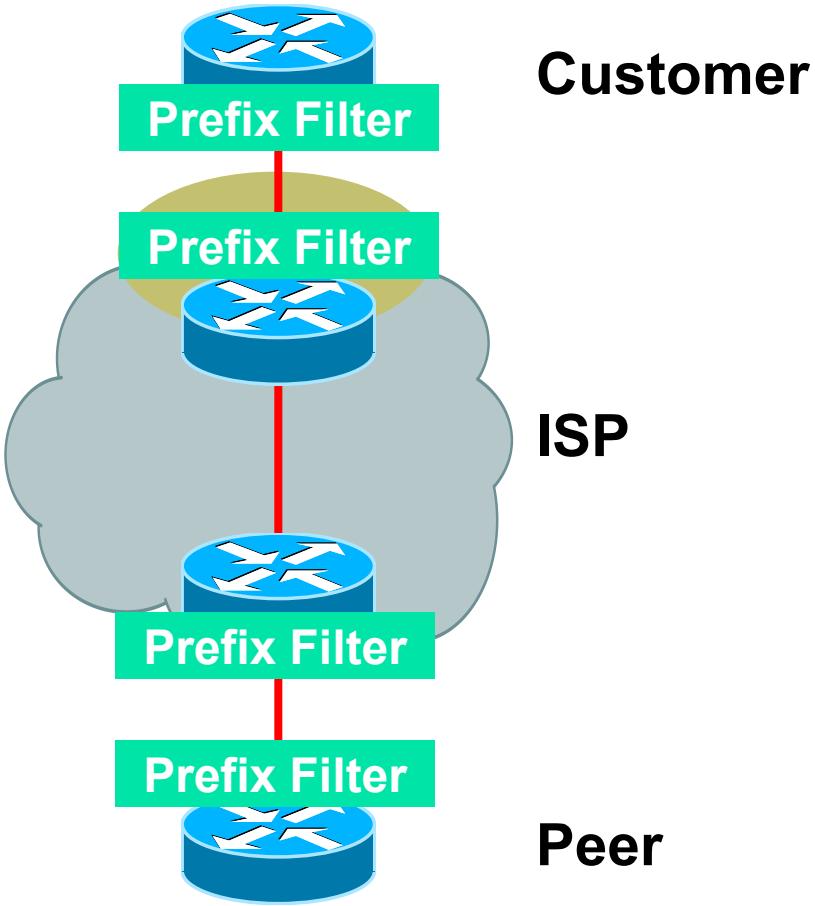
Common routing protocol misconfigurations

- BGP:
  - IXP participant changes, or addresses are hijacked
    - By default, BGP will try to establish outgoing BGP sessions
    - The peer can glean the ASN configuration from these packets
    - And configure their router
      - Your side is already configured!
    - Use authentication **or** regularly audit and remove idle/inactive peering sessions
  - MD5 keys can be error prone to exchange (and are often exchanged via insecure means)
    - Is the effort worth the benefit?
    - Other mechanisms may help
  - MD5 for internal (IBGP) sessions still makes sense

Prefix Filters

Apply Prefix Filters to All eBGP Neighbors to Prevent Injection of False Routing Information

- To/from customers
- To/from peers
- To/from upstreams



Route Filtering

- Consider filtering routes on input to a protocol
  - You can do this on both RIP and BGP
  - The architecture of OSPF and IS-IS does not allow input filters to be placed on them
- Possible inbound filters include:
  - Reject any of your own routes advertised to you
  - Reject any external RFC 1918 routes advertised to you
  - Reject any external routes from reserved or unallocated address space
  - Reject any instance of the default route sent to you

BGP thoughts

- Consider what routes you should get from peer
  - Accept those
  - Filter all as default
- Customer (non transit) BGP routes should be received with
  - Customer AS Number, AS-Path length = 1,
  - Customer address range only
  - What degree of sub-netting will you allow
    - What is useful?

Prefix-List for a BGP Prefix List

```
ip prefix-list rfc1918 seq 5 deny    0.0.0.0/8 le 32
ip prefix-list rfc1918 seq 10 deny   10.0.0.0/8 le 32
ip prefix-list rfc1918 seq 15 deny   127.0.0.0/8 le 32
ip prefix-list rfc1918 seq 20 deny   169.254.0.0/16 le 32
ip prefix-list rfc1918 seq 25 deny   172.16.0.0/12 le 32
ip prefix-list rfc1918 seq 30 deny   192.0.2.0/24 le 32
ip prefix-list rfc1918 seq 35 deny   192.168.0.0/16 le 32
ip prefix-list rfc1918 seq 40 deny   224.0.0.0/3 le 32
ip prefix-list rfc1918 seq 45 permit 0.0.0.0/0 le 32
```

BGP with Prefix-List Route Filtering

```
router bgp myASN
       neighbor CE.AD.RE.SS prefix-list rfc1918 in
       neighbor CE.AD.RE.SS prefix-list rfc1918 out
```

Lab 4

- Implement the prefix-list on your E-BGP (PE-CE) session to deny bogon addresses

- What happens?

Lab 4.1

- Change the address on loopback1 on the CE routers to:
- 200.0.XX.1/32 where XX is the router-ID
- Announce this new network

- Is connectivity back?

Lab 4.2

- Configure next-hop-self on your IBGP (PE-PE) sessions

```
configure t
  router bgp <ASN>
     neighbor X.X.X.X next-hop-self
```

Agenda: Additional Routing Security

- Routing Protocol Authentication
- Securing RIP
- Securing OSPF
- Securing IS-IS
- Securing BGP
- ➔ Additional Routing Security

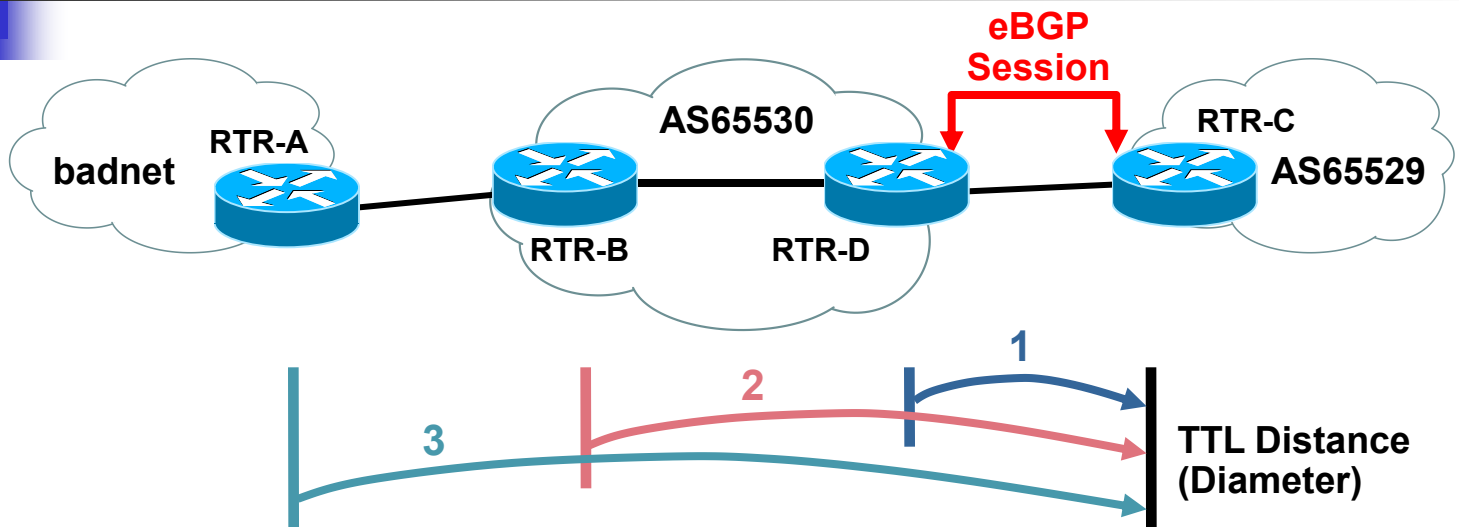
Other BGP thoughts

- Clean routes
  - Will you accept MEDS?
  - Leave communities alone
    - If you honor extended community format all should be OK
    - AS:nn
  - Is default allowed?
  - Filter martians and bogons

BGP Support for TTL Security Check

- AKA BGP TTL Security Hack (BTSH/GTSH)
- Protects eBGP sessions from CPU attacks using forged IP packets
- Prevents attempts to hijack eBGP session by attacker not part of either BGP network or that is not between the eBGP peers
- Configure minimum Time To Live (TTL) for incoming IP packets from a specific eBGP peer
  - BGP session established and maintained only if TTL in IP packet header is equal to or greater than configured TTL value. Initial TTL set to 255
  - If value is less than configured value packet is silently discarded and no ICMP message generated
- Not supported for iBGP and occurs after MD5 check if enabled
- Available in 12.0(27)S, 12.3(7)T, and 12.2(25)S
  - http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123newft/123t/123t_7/gt_btsh.htm

BGP TTL Security Check: How Does It Work?



Example on RTR-C:

```
router bgp 65529
neighbor 10.1.1.1 ttl-security hops 1
! expected TTL value in the IP packet header is 254
```

- Spoofed IP packets may have correct IP source and destination addresses (and TCP source and destination ports); however, unless these packets originate on a network segment that is between the eBGP peers, the TTL values will be less than the "minimum" configured in the BGP TTL security check

Not really security related…but…

- Extensive use of policies to provide customer services
  - E.g. Provider provisioned Local Pref
  - Examples at www.sprint.net
    - Under BGP policies

Martian Addresses

- One way to filter is to add prefixes to your martian address list
  - Address prefixes for which the routers ignore all associated routing information
- Martians are not installed into the routing table
- In JUNOS and Alcatel-Lucent software, the default martian addresses are:
  - 0.0.0.0/8 orlonger
  - 127.0.0.0/8 orlonger
  - 128.0.0.0/16 orlonger ← Deprecated!
  - 191.255.0.0/16 orlonger ← Deprecated!
  - 192.0.0.0/24 orlonger
  - 223.255.255.0/24 orlonger ← Deprecated!
  - 240.0.0.0/4 orlonger

Bogons

- Addresses that are unallocated
- MUST MUST MUST MUST MUST MUST
  - Keep them up to date!
- http://www.team-cymru.org/Services/Bogons/
- Broken bogon lists cause significant pain for everyone!

Additional BGP Security

- Consider enabling route damping
  - Limits the number of times an unstable (or compromised) peer can destabilize your own routing process
- Consider establishing a limit to the number of prefixes you will accept from a peer
  - max-prefix

Source MAC Address Filtering

In shared peering environments over broadcast capable media, consider using source MAC address filtering

```
[edit interfaces fe-0/0/3]
lab@R1# show
fastether-options {
    source-filtering;
    source-address-filter {
        00:e0:18:01:18:4c;
    }
}
unit 0 {
    family inet {
        address 10.2.100.1/24;
    }
}
```

BGP Attack Vectors

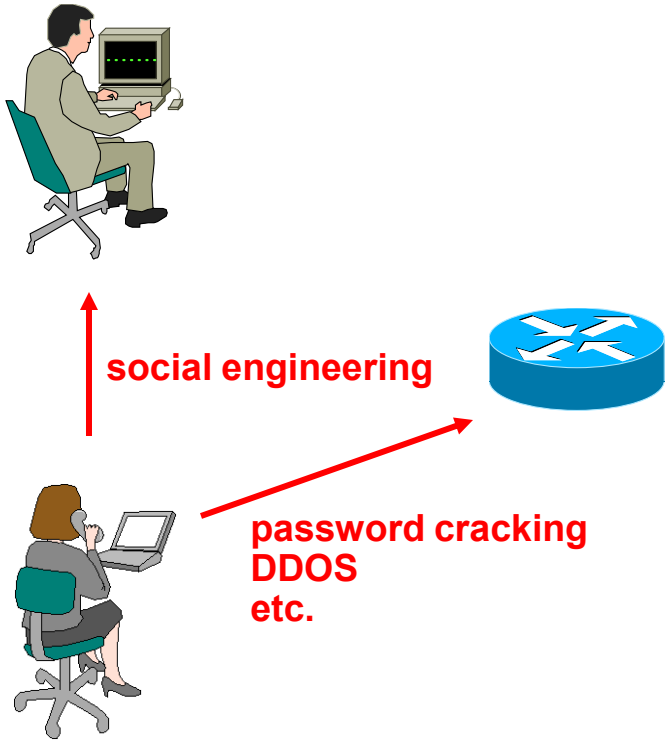
- Understanding BGP Attack Vectors will help you plan and prioritize the techniques deployed to build greater resistance into the system.
- The following documents will help you gain perspective on the realistic Risk Assessment:
  - NANOG 25 - BGP Security Update
    - http://www.nanog.org/mtg-0206/barry.html
  - NANOG 28 - BGP Vulnerability Testing: Separating Fact from FUD
    - http://www.nanog.org/mtg-0306/franz.html
- Look for the updates links to get the latest risk assessments.
  - http://www.cisco.com/security_services/ciag/initiatives/research/projectsummary.html

Whacking the BGP Session

- Four Macro Ways you can Whack the BGP Session:
  - Saturate the Receive Path Queues: BGP times out
  - Saturate the link: link protocols time out
  - Drop the TCP session
  - Drop the IGP causing a recursive loop up failure

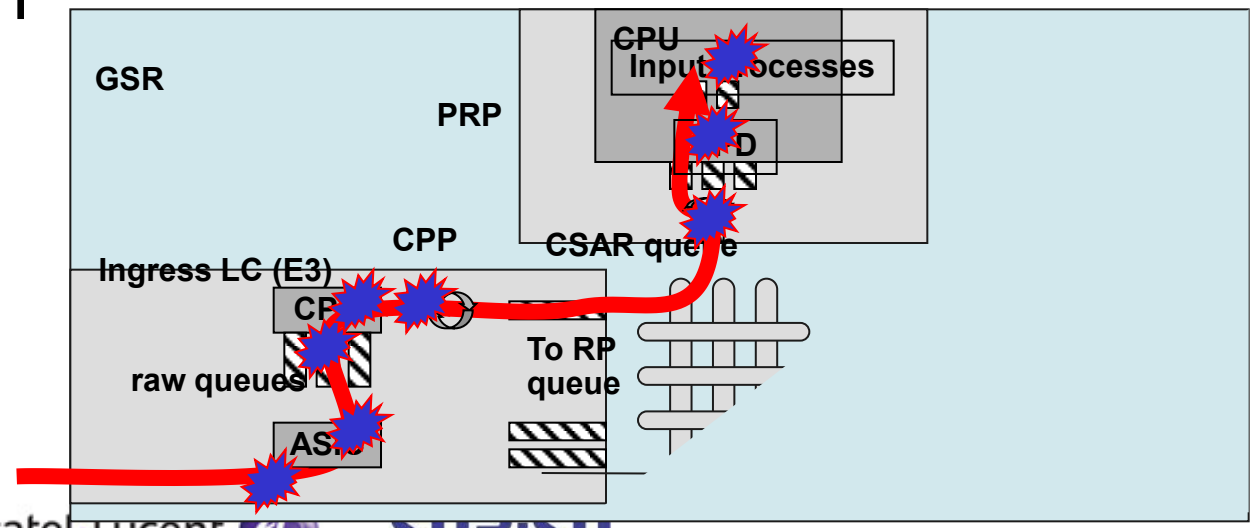
Attacking Routing Devices

- All the normal host attack methods apply to routers
  - Social engineering
  - Password cracking
  - Denial of service
  - etc.
- What an attacker needs:
  - Access to the router
  - (or)
  - Access to the network



Saturate the Receive Path Queues

- Routers usually have various receive path queues that are hit as the packet heads for the TCP Stack.
- Saturation Attacks fill these queues – knocking out valid packets from the queues.
- Consequence: BGP Times out – Dropping the BGP Session



Saturate the Link

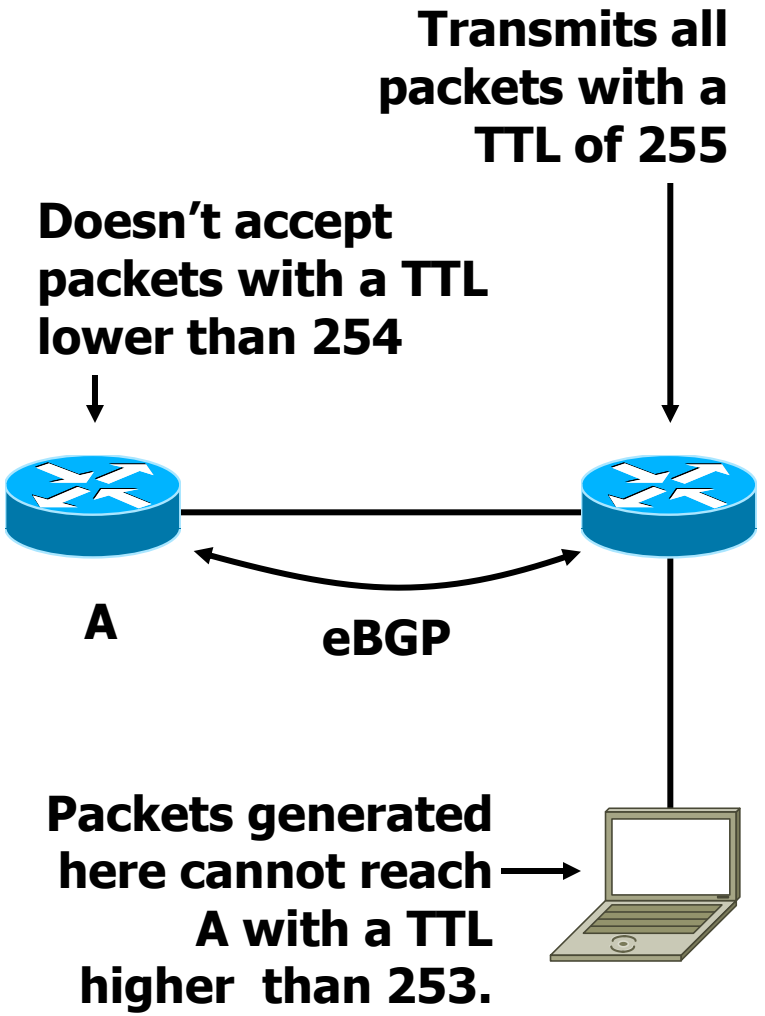
- DOS Attacks Saturating the link will knock out valid control plane packets.
- Link packet over POS, ATM, or Ethernet will drop out – which drop out the link – which drop out the FIB's next hop – which knocks out the BGP Entries
- This is a very effective brute force attack.

Drop the TCP Session

- Dropping the TCP Session was thought to require a breath of packets.
- TCP Session can be dropped with a RST or a SYN (per RFC).
- Successful L4 Spoof is required
  - Match source address
  - Match source port
  - Match destination address (obvious)
  - Match destination port
  - Match Sequence Number (now just get inside the window)

Generalized TTL Security Mechanism

- GTSH is a hack which protects the BGP peers from multihop attacks.
- Routers are configured to transmit their packets with a TTL of 255, and to reject all packets with a TTL lower than 254 or 253.
- A device which isn't connected between the routers cannot generate packets which will be accepted by either one of them.



Peer Authentication

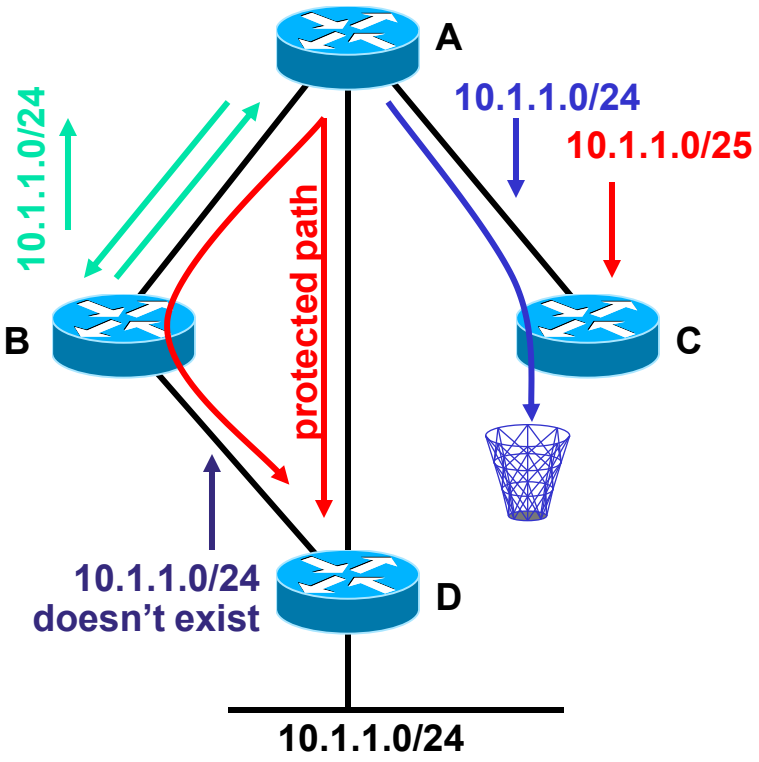
- MD5 Peer authentication can protect against:
  - Malformed packets tearing down a peering session
  - Unauthorized devices transmitting routing information
- MD5 Peer authentication cannot protect against:
  - Reset routing protocol sessions due to denial of service attacks
  - Incorrect routing information being injected by a valid device which has been compromised

Drop the IGP

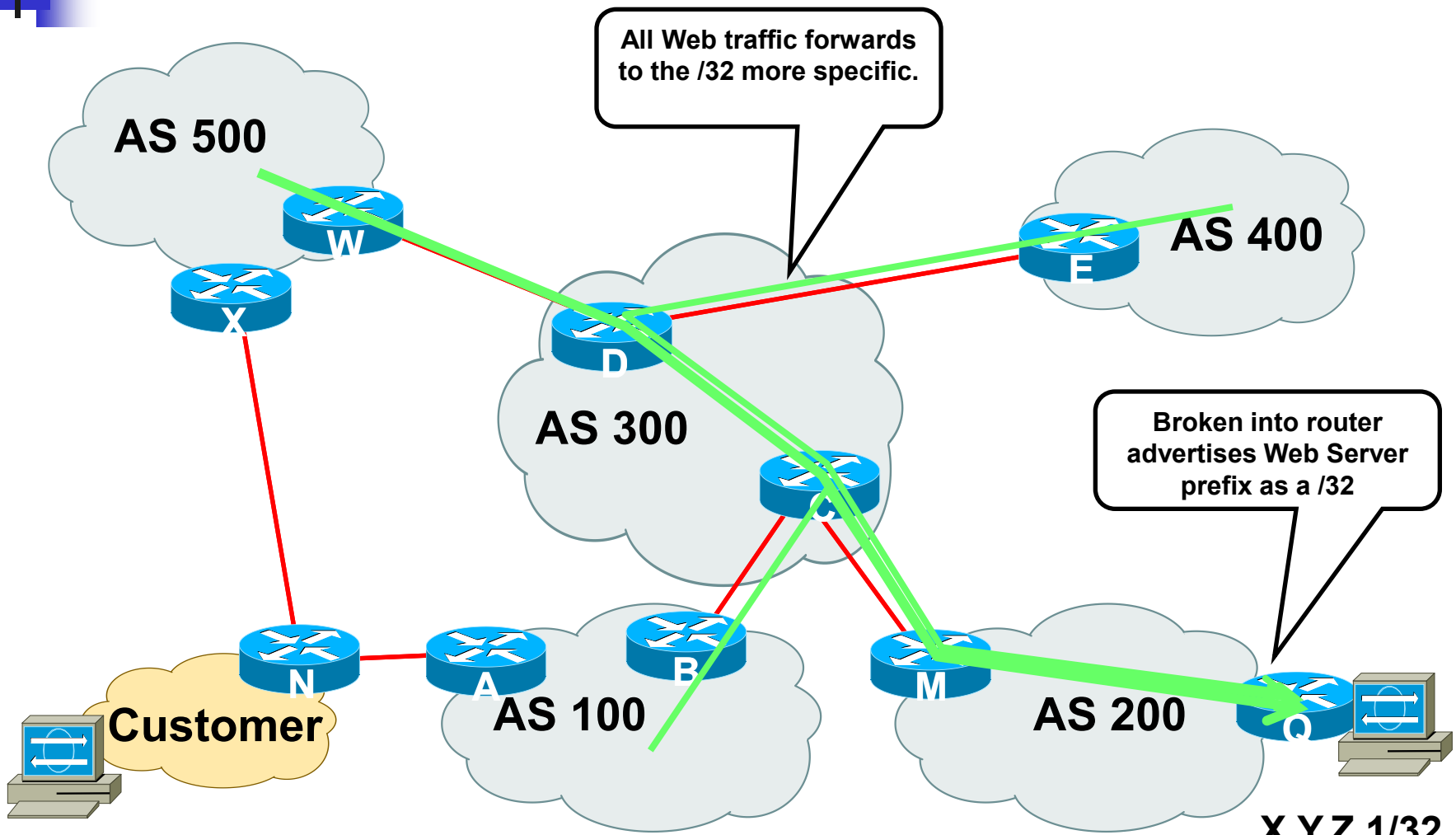
- Miscreant Success Principle - If you cannot take out the target, move the attack to a coupled dependency of the target.
- BGP's coupled dependency is the IGP it requires for recursive look-up.
- EIGRP and OSPF are both open to external attacks.

Attacking Routing Data

- How could you attack routing data?
- Modification
  - Direct traffic along an unprotected path
  - Direct traffic into a black hole
  - Create a routing loop
- Overclaiming
  - Injecting nonexistant destinations
  - A longer prefix!
- Underclaiming
  - Removing destinations



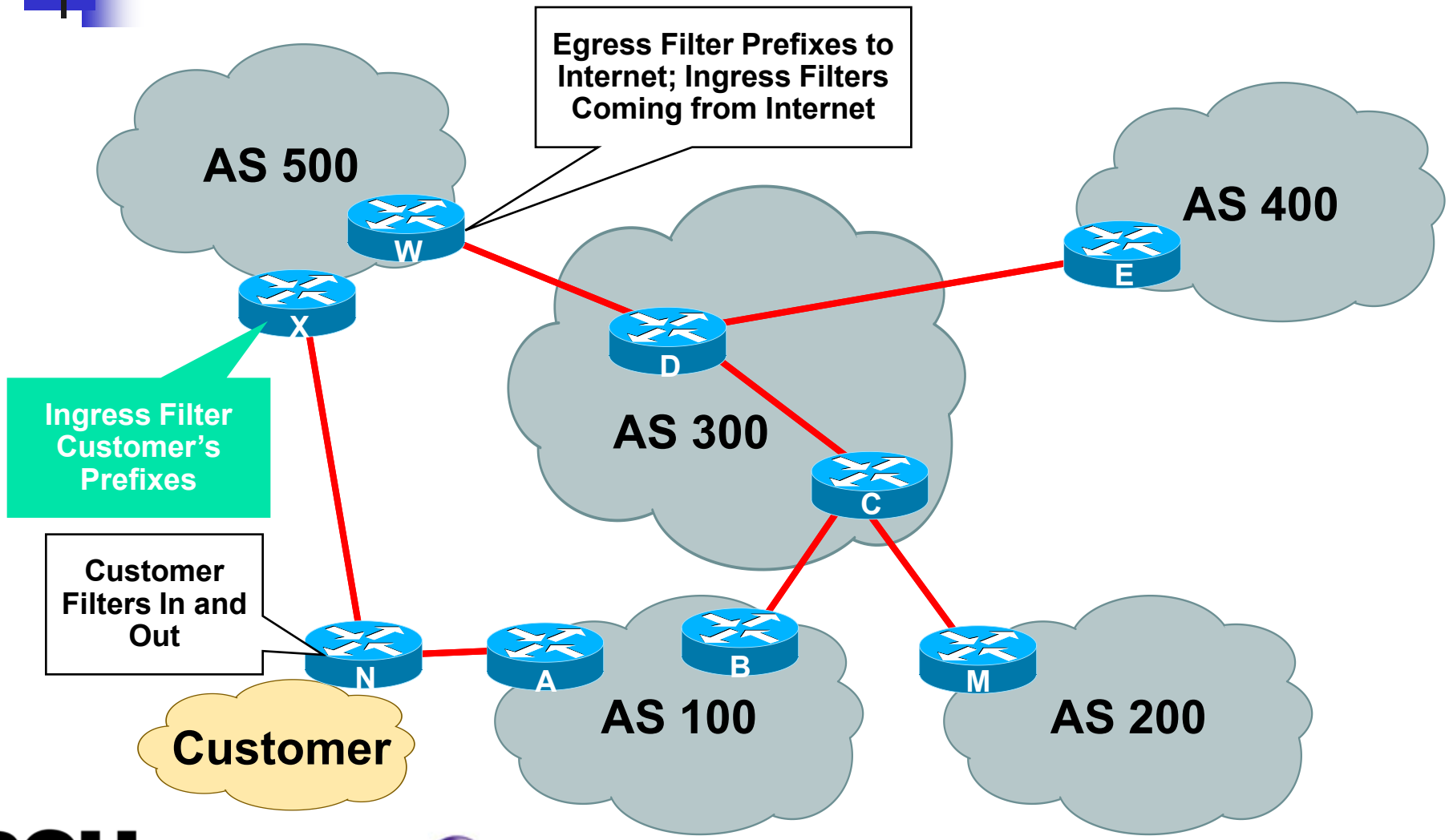
What is a prefix hijack?



Malicious Route Injection
What can ISPs Do?

- Customer Ingress Prefix Filtering!
- ISPs should only accept customer prefixes which have been assigned or allocated to their downstream customers.
- For example
  - Downstream customer has 220.50.0.0/20 block.
  - Customer should only announce this to peers.
  - Upstream peers should only accept this prefix.

Where to Prefix Filter?



Bogons and Special Use Addresses

- IANA has reserved several blocks of IPv4 that have yet to be allocated to a RIR:
  - http://www.iana.org/assignments/ipv4-address-space
- These blocks of IPv4 addresses should never be advertised into the global internet route table
- Filters should be applied on the AS border for all inbound and outbound advertisements
- Special Use Addresses (SUA) are reserved for special use :-)
  - Defined in RFC3330
  - Examples: 127.0.0.1, 192.0.2.0/24

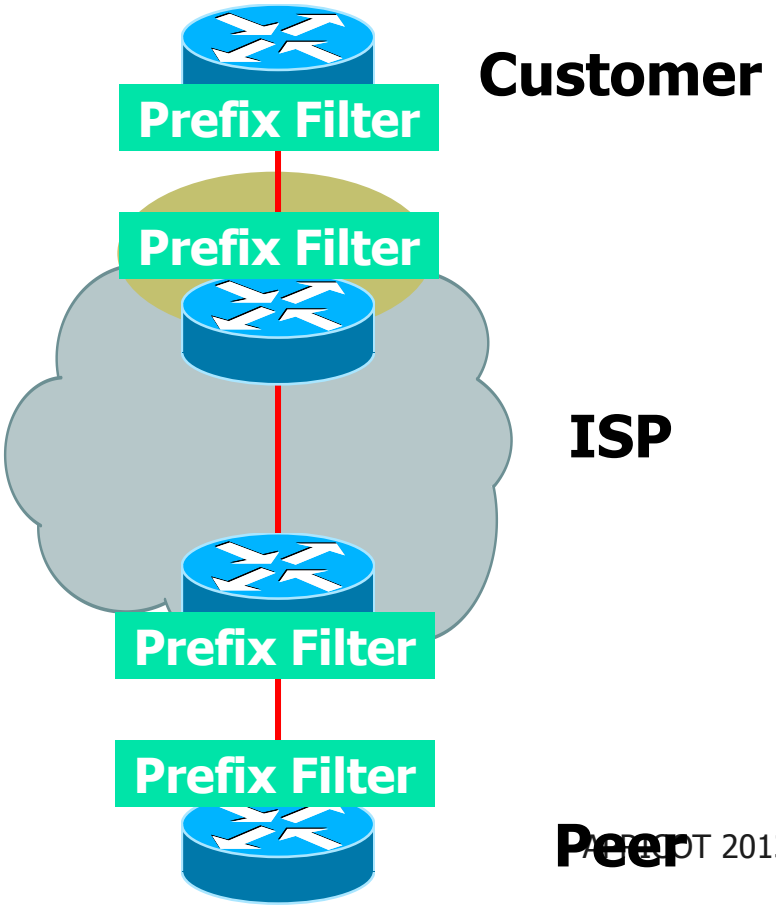
Bogons and Special Use Addresses

- IANA is fully allocated! IPv4 is exhausted!
  - http://www.iana.org/assignments/ipv4-address-space
  - http://tools.ietf.org/html/draft-vegoda-no-more-unallocated-slash8s-01
- Filters should be applied on the AS border for all inbound and outbound advertisements
- Special Use Addresses (SUA) are reserved for special use :-)
  - Defined in RFC5735 (updates RFC3330)
  - Examples: 127.0.0.1, 192.0.2.0/24

Prefix Filters: Application

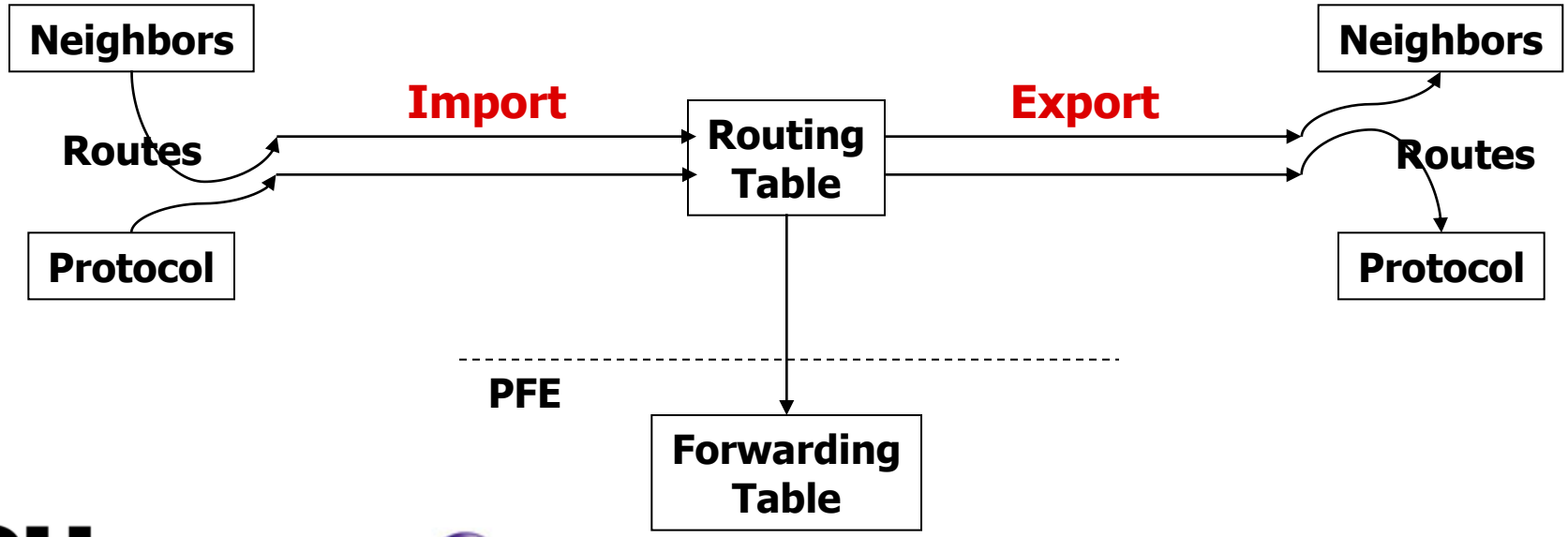
Apply Prefix Filters to All eBGP Neighbors

- To/from customers
- To/from peers
- To/from upstreams



Import and Export Policies

- Perform policy filtering with respect to the software routing table



Match Conditions

- Cisco IOS uses **route-map** for policies
- Policies typically contain some form of match criterion
- Possibilities include:
  - Neighbor address
  - Protocol (source of information)
    - BGP, direct, DVMRP, IS-IS, local, MPLS, OSPF, PIM, RIP, static, aggregate
  - Routing protocol information
    - OSPF area ID
    - IS-IS level number
    - BGP attributes
  - Regular expression-based matches for AS path and communities

Match Actions

- The action associated with a given entry is performed for matching routes:
  - Terminating actions
    - Accept
    - Deny
  - Flow control actions
    - Continue to another
  - Modify attributes actions
    - Metric
    - Local Preference
    - Next-hop address
    - Community

Default Policies

- Each platform behaves differently
  - JUNOS, SR-OS, IOS-XR all have conservative export policies that will not announce anything by default
  - IOS is much more 'chatty'
- IOS will export IGP learned routes by default into IGPs
  - We manage this with passive interface default, and only enabling IGP on interfaces where we want the networks announced (including loopback)
- BGP
  - Import all routes learned from BGP neighbors
  - Export all active routes learned from BGP neighbors to all BGP neighbors
    - EBGP-learned routes are exported to all BGP peers
    - IBGP-learned routes are exported to all EBGP peers (assumes logical IBGP full mesh)

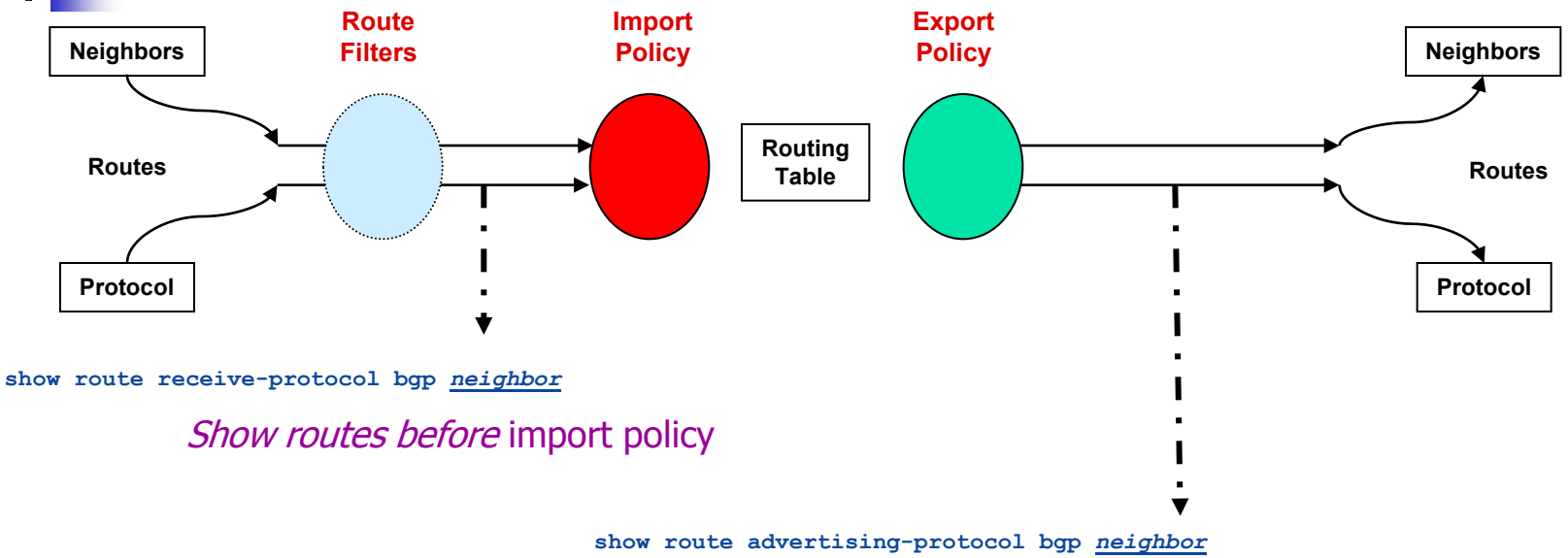
Apply Routing Policy to BGP

- BGP has three filtering points per direction:
  - Global
  - Groups of neighbors
  - Individual neighbors
- Only the *most* specific policies are applied to a particular peer
  - Neighbor policy overrides group and global policies
  - Group policy overrides global policy

BGP Policy Application Example

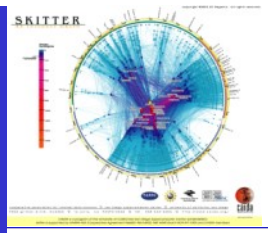
```
configure terminal
  route-map std-peer-in deny 1
     match prefix-list MYPREFIXES
  route-map std-peer-in permit 2
     set local-preference 100
  route-map std-peer-out permit 1
     set community 65999:65
  router bgp ASN
     neighbor x.x.x.x route-map std-peer-in in
     neighbor x.x.x.x route-map std-peer-out out
```

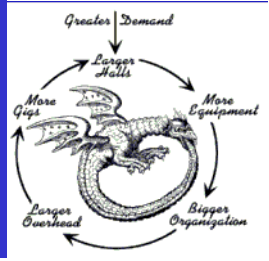
Monitoring Policy Operation



- The `show route receive-protocol` and `show route advertising-protocol` commands:

Remote Trigger Black Hole [RTBH]

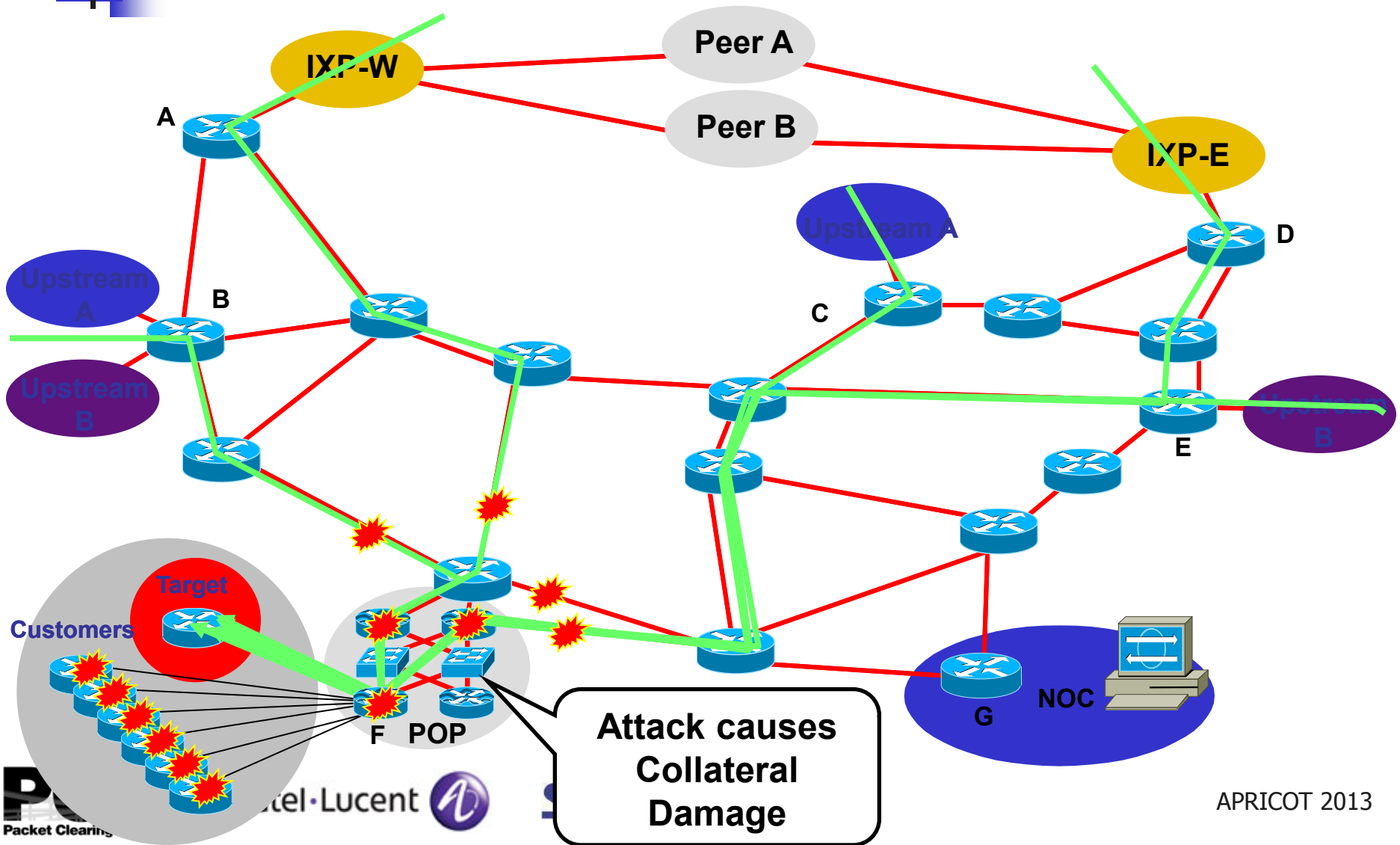




RTBH Filtering

- We use BGP to trigger a network wide response to a range of attack flows.
- A simple static route and BGP will allow an ISP to trigger network wide black holes as fast as iBGP can update the network.
- This provides ISPs a tool that can be used to respond to security related events or used for DOS/DDOS Backscatter Tracebacks.

Customer is DOSed – Before – Collateral Damage



Shunning with uRPF and BGP

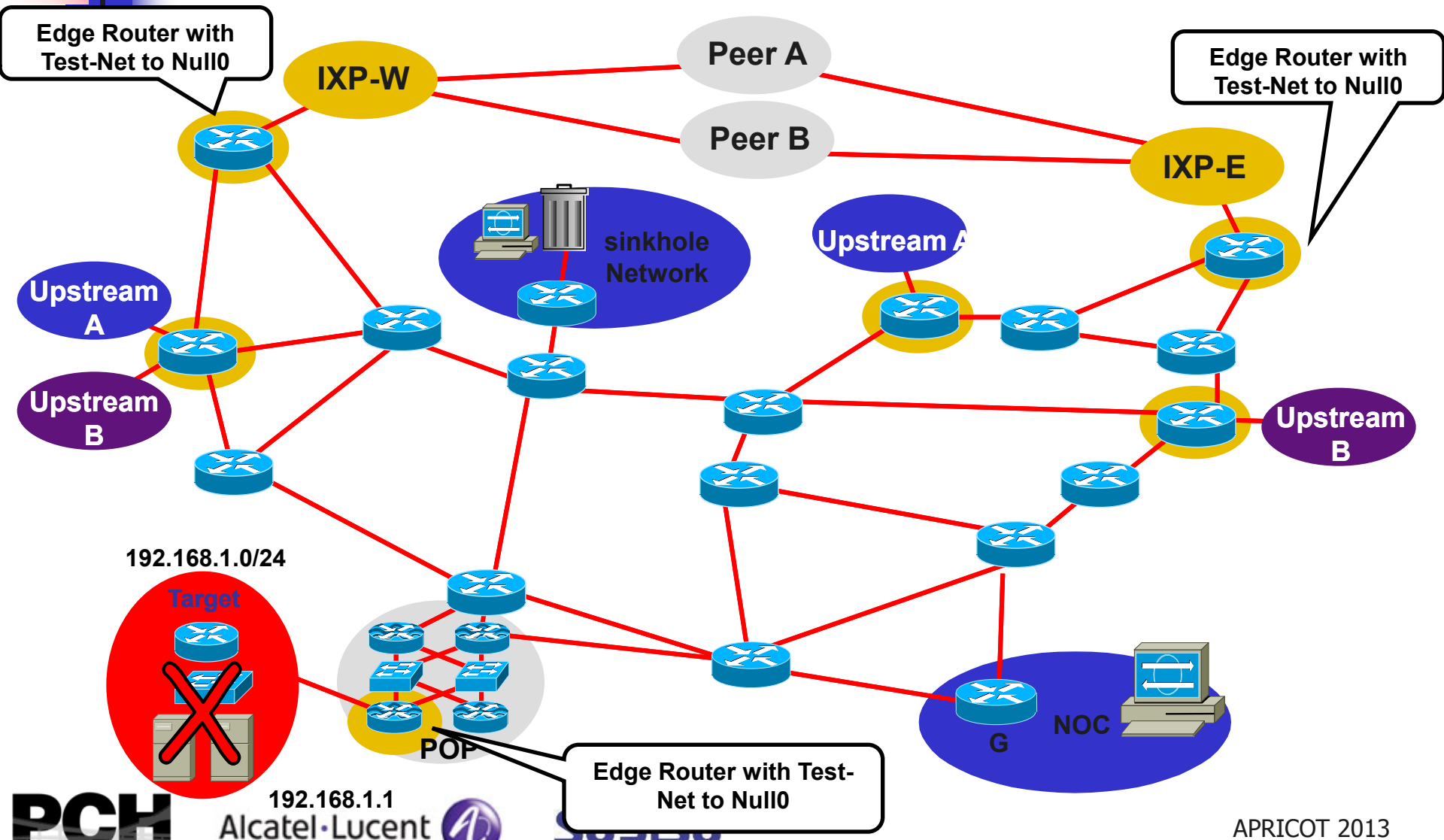
- BGP cannot send "next-hop null0"… but:
- BGP can send "next-hop 192.0.2.1"
- And on each border router:
  - ip route 192.0.2.1 null0
- Router receives iBGP routing update:
  - "Route x.x.x.x next-hop 192.0.2.1" (comm: local-AS)
  - And it has an ip route 192.0.2.1 null0
  - Thus: x.x.x.x → null0

Step 1- Prepare all the Routers w/ Trigger

- Select a small block that will not be used for anything other than black hole filtering. Test Net (192.0.2.0/24) is optimal since it should not be on the Net and is not really used.
- Put a static route with Test Net – 192.0.2.0/24 to Null 0 on every edge router on the network.

```
ip route 192.0.2.1 255.255.255.255 Null0 255
ip route 192.0.2.2 255.255.255.255 Null0 199
ip route 192.0.2.3 255.255.255.255 Null0 50
```

Step 1- Prepare all the Routers w/ Trigger



Step 2 – Prepare the Trigger Router

- The trigger router is the device that will inject the iBGP announcement into the ISP's Network.
- iBGP neighbor - it can (and probably should) be a route-reflector client, no need for full mesh
- Can be a separate router or Arbor's Peakflow DoS anomaly-detection system - if a separate router, something small like a 2600 will do (it doesn't need to receive BGP routes, only send them; use a prefix-list to ensure that it doesn't end up redistributing routes)
- Can be a production router (some ISPs do this, but not the recommended approach)
- Can be a workstation with Zebra/Quagga (interface with Perl scripts and other tools).

Trigger Router's Config

Redistribute Static with a route-map

```
router bgp 109
redistribute static route-map static-to-bgp
.
!
route-map static-to-bgp permit 10
match tag 66
set ip next-hop 192.0.2.1
set local-preference 50
set community no-export
set origin igp
!
Route-map static-to-bgp permit 20
```

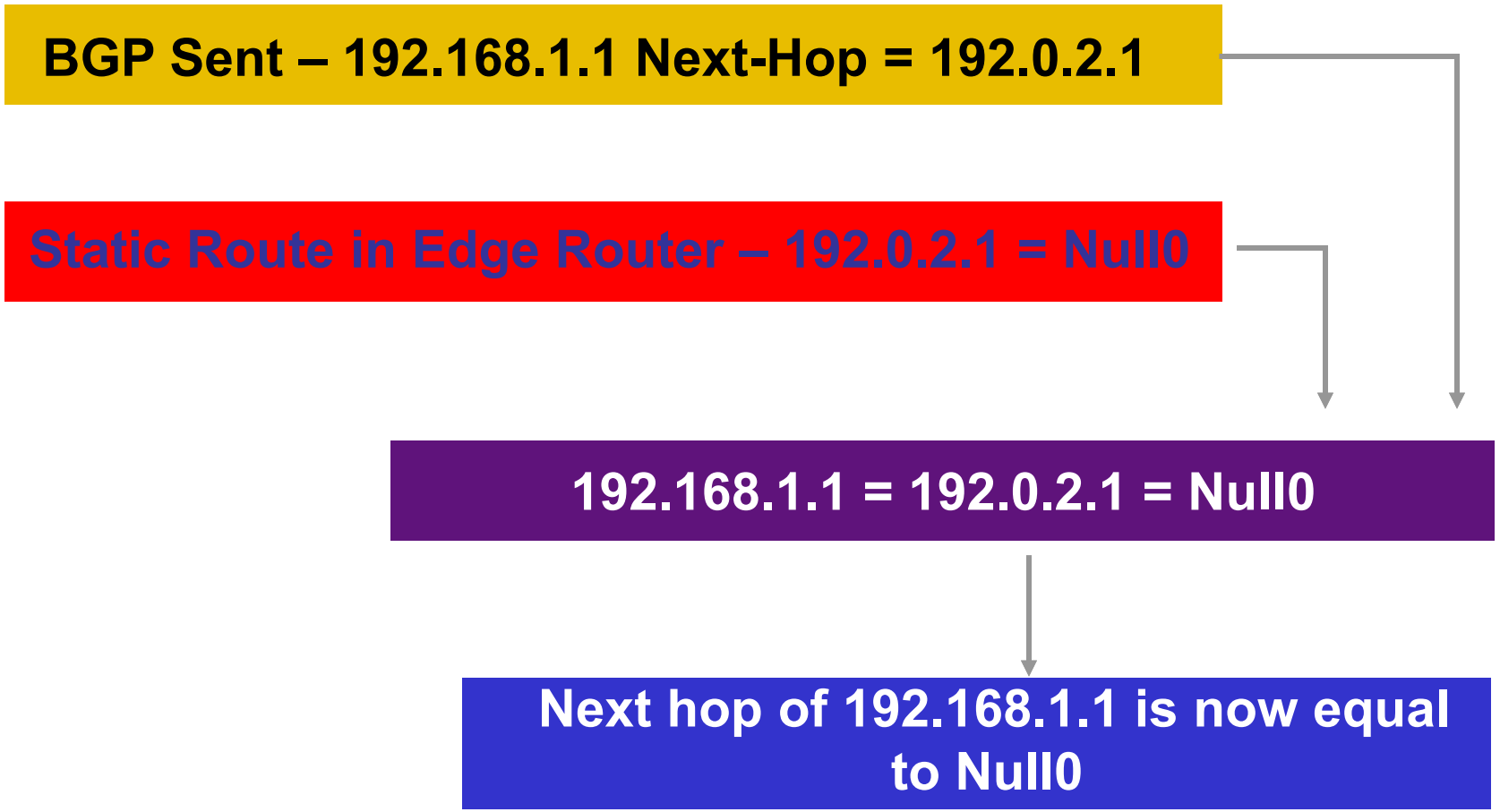
Match Static Route Tag

Set Next-Hop to the Trigger

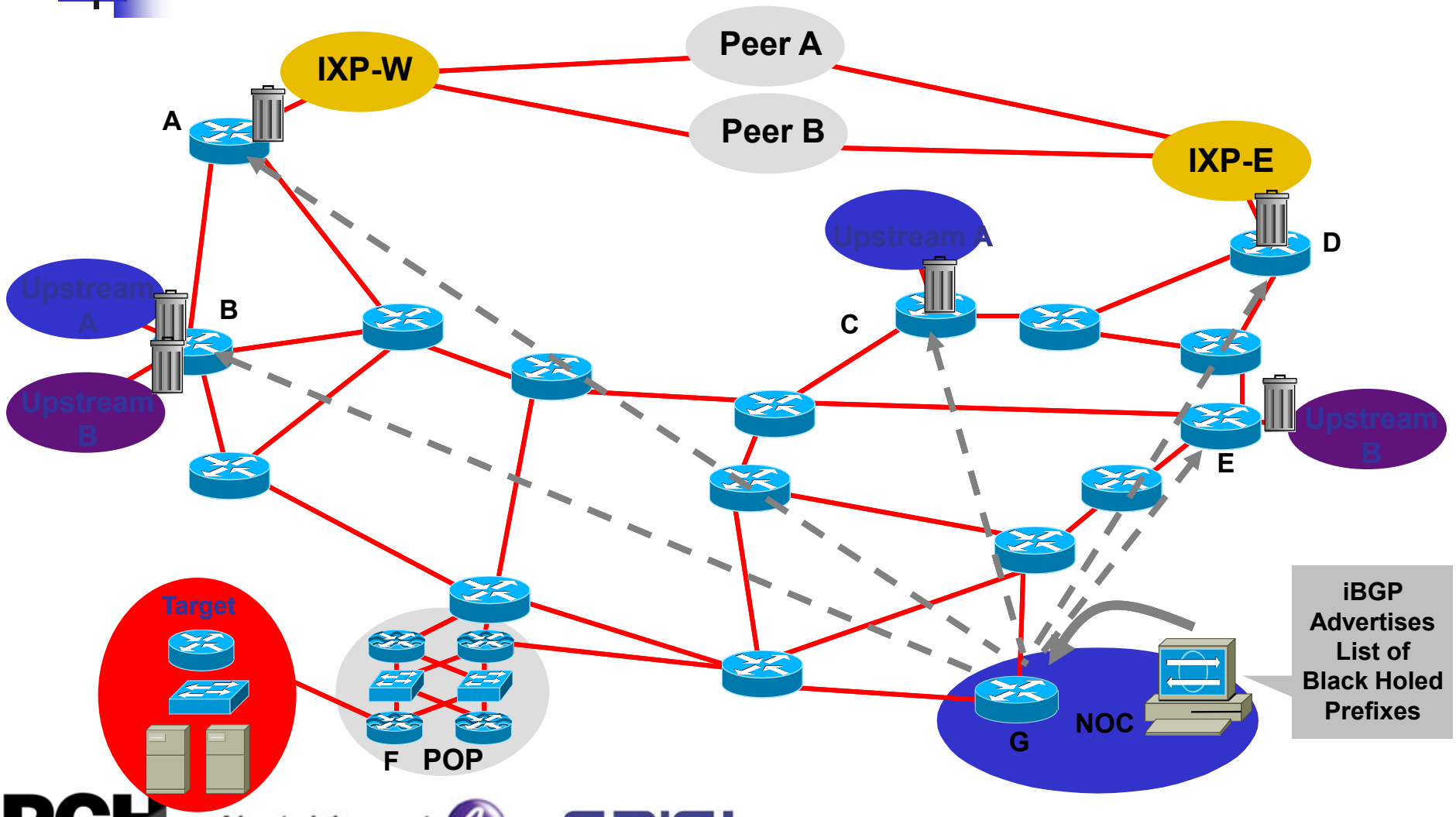
Step 3 – Activate the Black Hole

- ISP adds a static route of the destination address they wish to black hole to the advertising router. The static is added with the "tag 66" to keep it separate from other statics on the router.
  - ip route 192.168.1.1 255.255.255.255 Null0 Tag 66
  - BGP Advertisement goes out to all BGP-speaking routers which peer with the trigger.
- Routers hear the announcement, glue it to the existing static on the route, and changes the next-hop for the BGP advertised route to Null0 – triggering black hole routing.

Activate the Black Hole



Activate the Black Hole



Lab 5

- On **all** routers configure:

```
config t
ip bgp-community new-format
```

- On PE1 configure:
  - ip route 192.0.2.1 255.255.255.255 null0
- On PE1 configure:

```
config t
ip community-list 66 permit 666:666
route-map pe-rtbh permit 1
  match community 66
  set ip next-hop 192.0.2.1
route-map pe-rtbh permit 2
    ^Z
router bgp <ASN>
    neighbor <CE-ADDRESS> route-map pe-rtbh in
```

Lab 5

- On CE1 configure loopback**2**
  - Use ip address 100.0.XX.1/32
- On CE1 configure route-map:

```
config t
  route-map rtbh permit 1
  set community 666:666
```

- On CE1 configure:

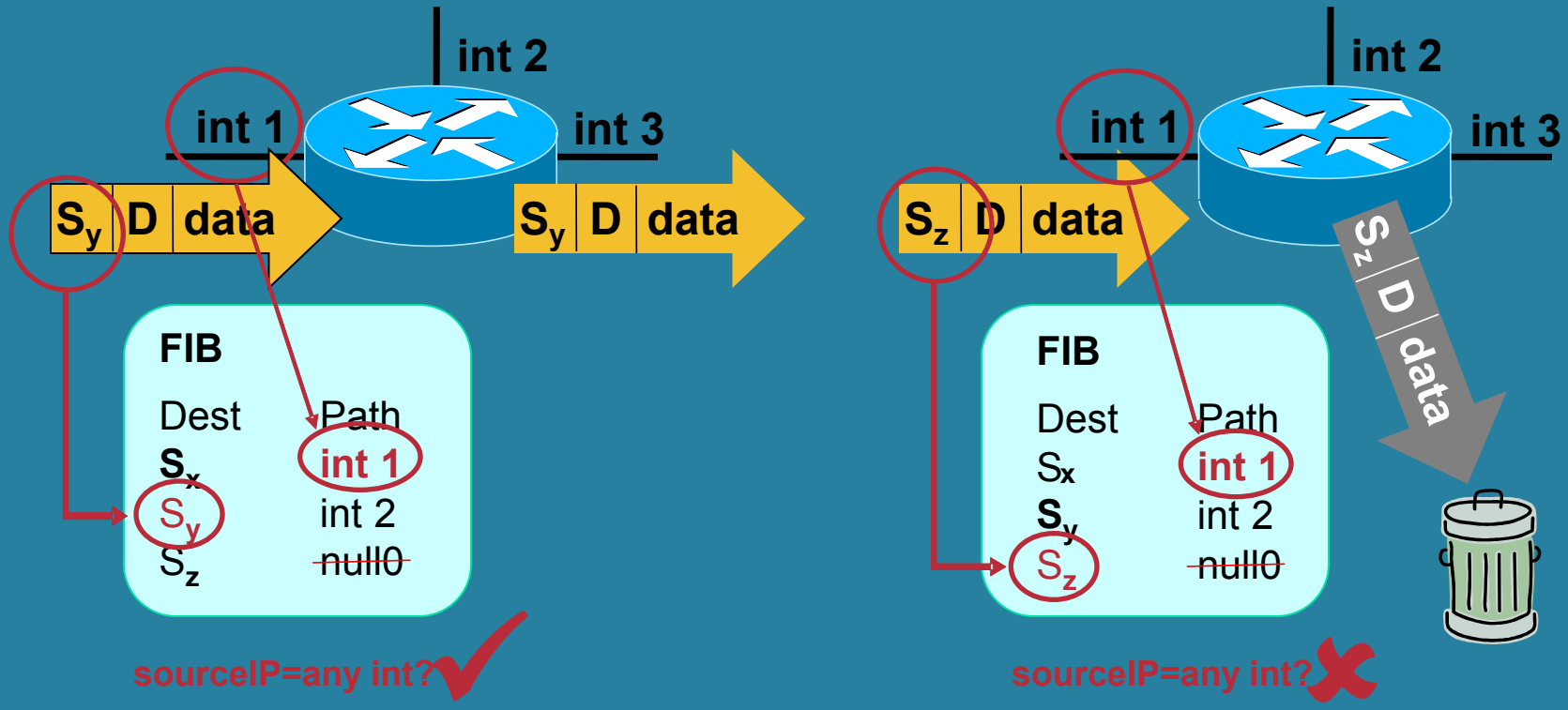
```
router bgp <ASN>
  neighbor <PE-ADDRESS> send-community
  network 100.0.XX.1 mask 255.255.255.255 route-map rtbh
```

Flipping it Around: Triggered Source Drops

- Dropping on destination is very important
  - Dropping on source if often what we really need
- Reacting using source address provides some interesting options:
  - Stop the attack without blackholing real services
  - Filter command and control servers
  - Filter (contain) infected end stations
- Must be rapid and scaleable
  - Leverage pervasive BGP again

uRPF – Loose Mode

router(config-if)# ip verify unicast source reachable-via any



IP verify unicast source reachable – via any

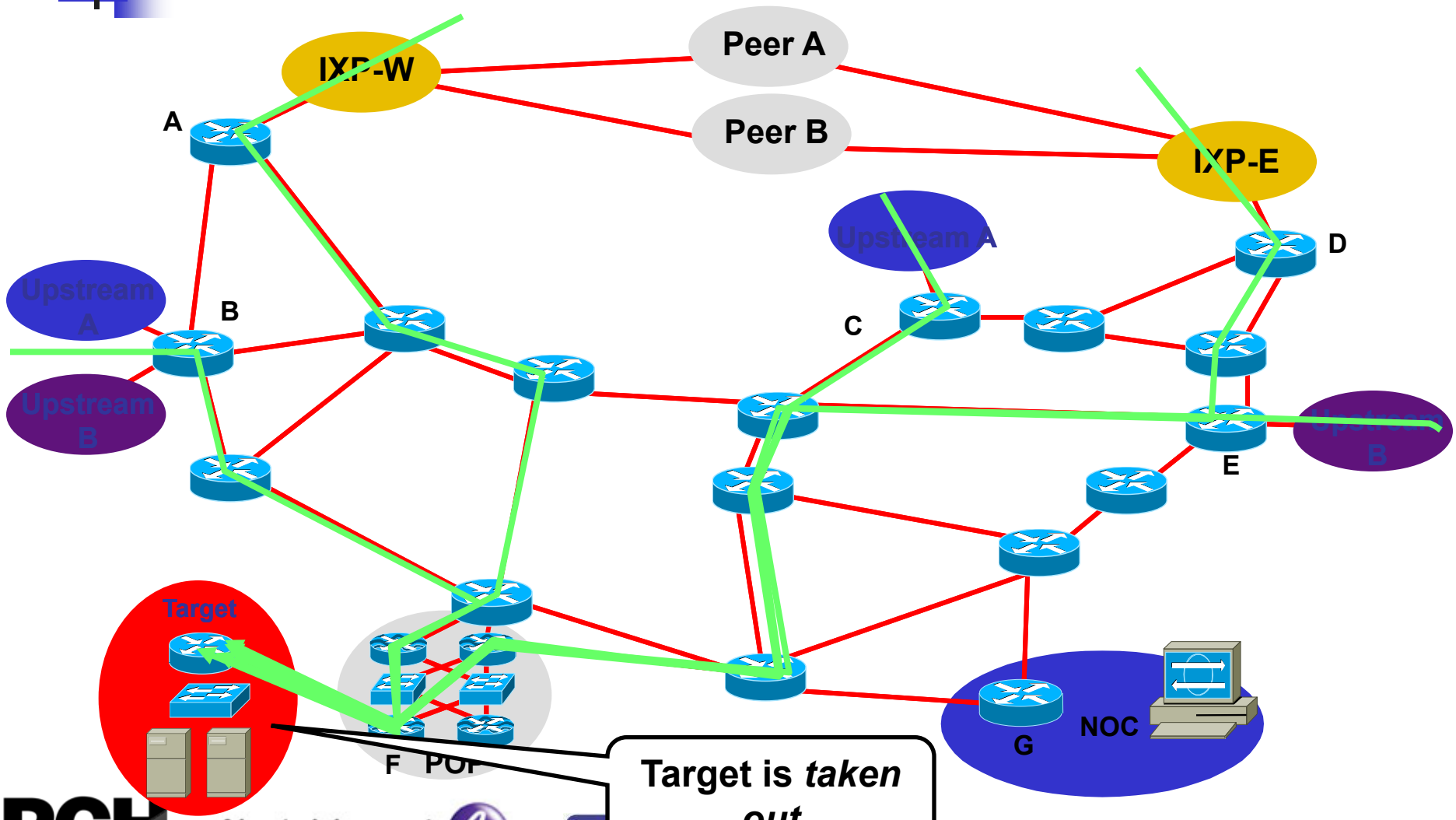
Source Based RTBH Filtering

- Uses the same architecture as destination based filtering + unicast RPF
  - Edge routers must have static in place
  - They also require unicast RPF
  - BGP trigger sets next hop -- in this case the "victim" is the source we want to drop

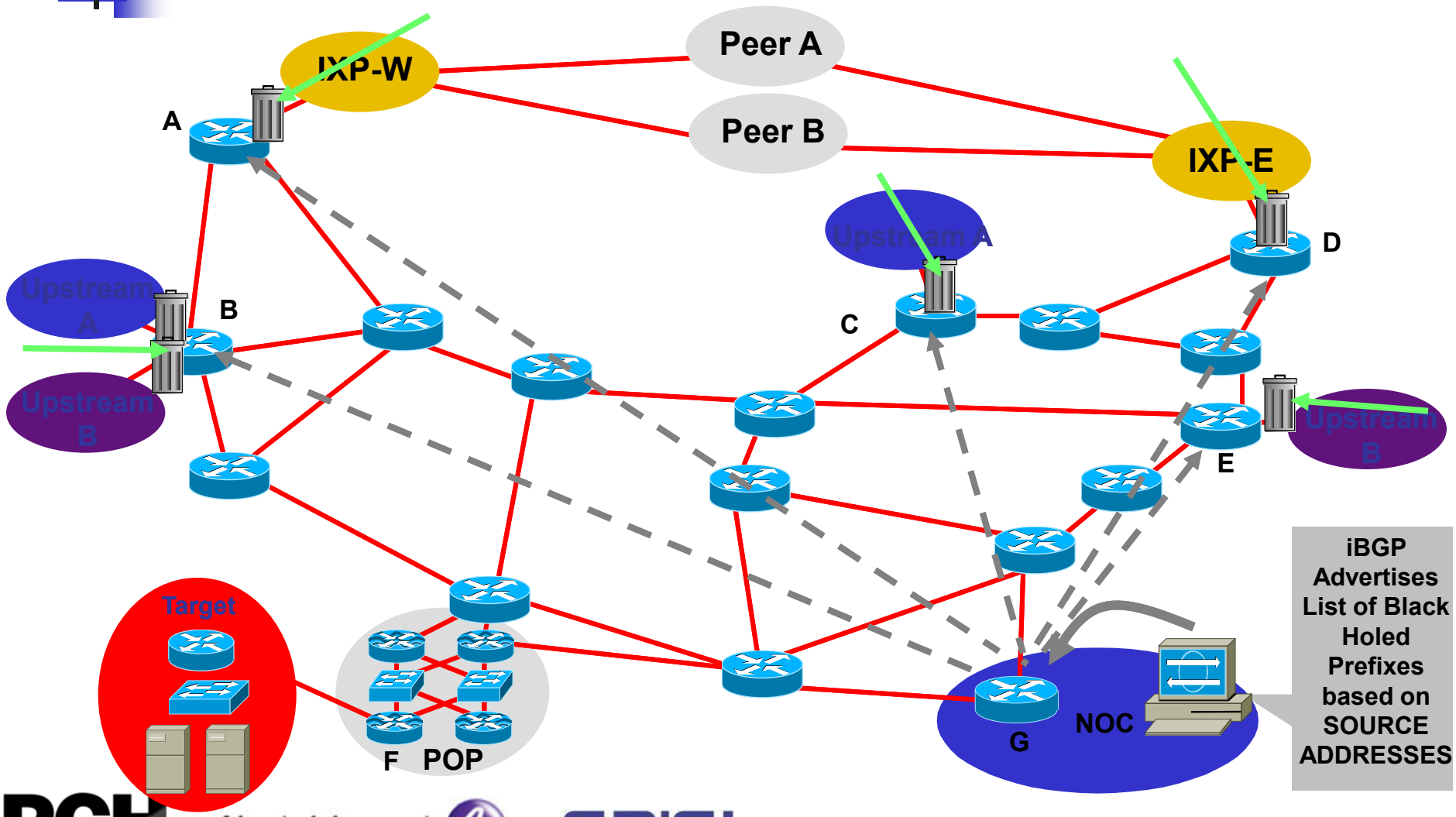
Source Based RTBH Filtering

- What do we have?
  - Black Hole Filtering – If the destination address equals Null 0 we drop the packet.
  - Remote Triggered – Trigger a prefix to equal Null 0 on routers across the Network at iBGP speeds.
  - uRPF Loose Check – If the source address equals Null 0, we drop the packet.
- Put them together and we have a tool to trigger drop for any packet coming into the network whose source or destination equals Null 0!

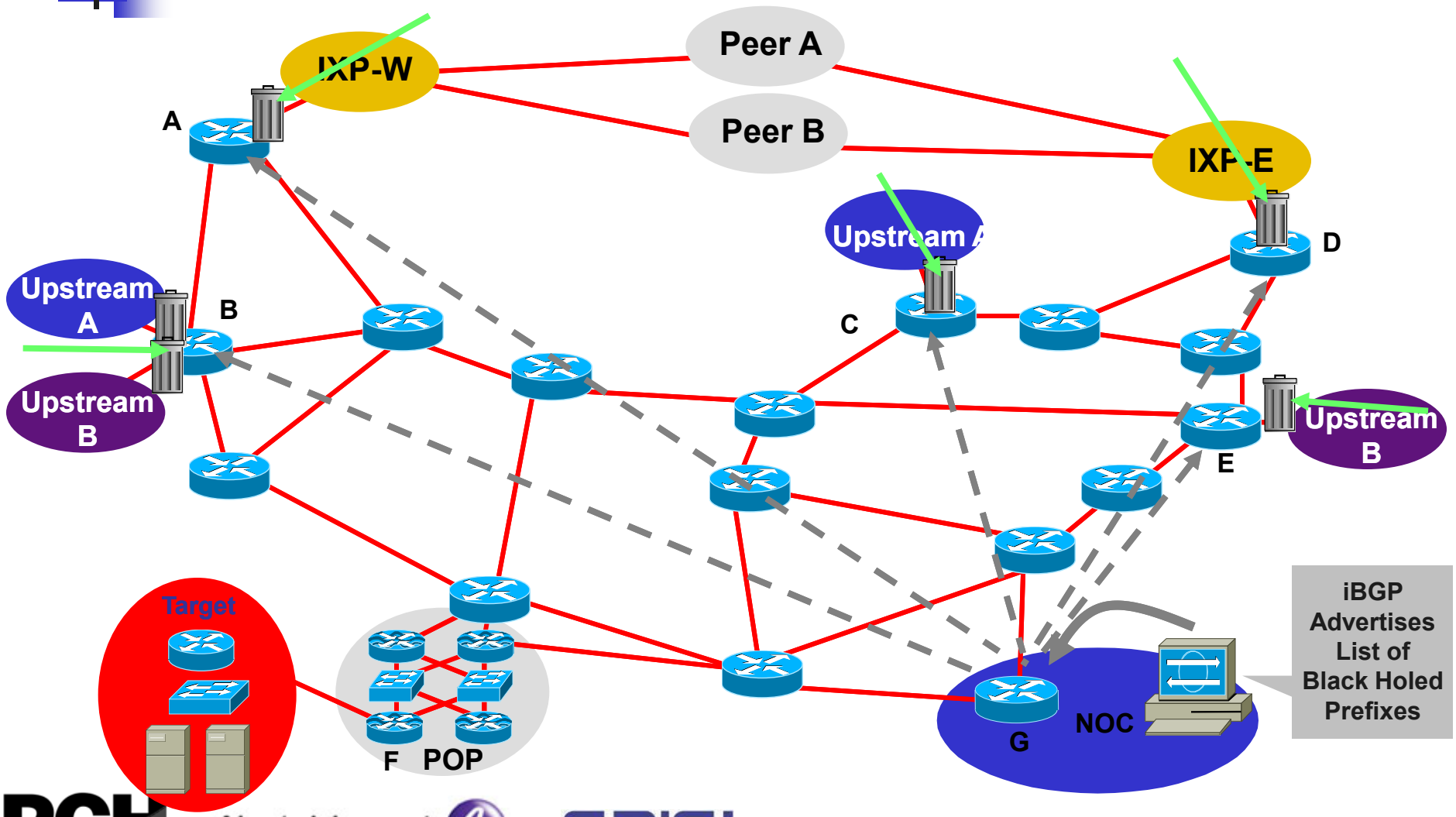
Customer is DOSed - Before



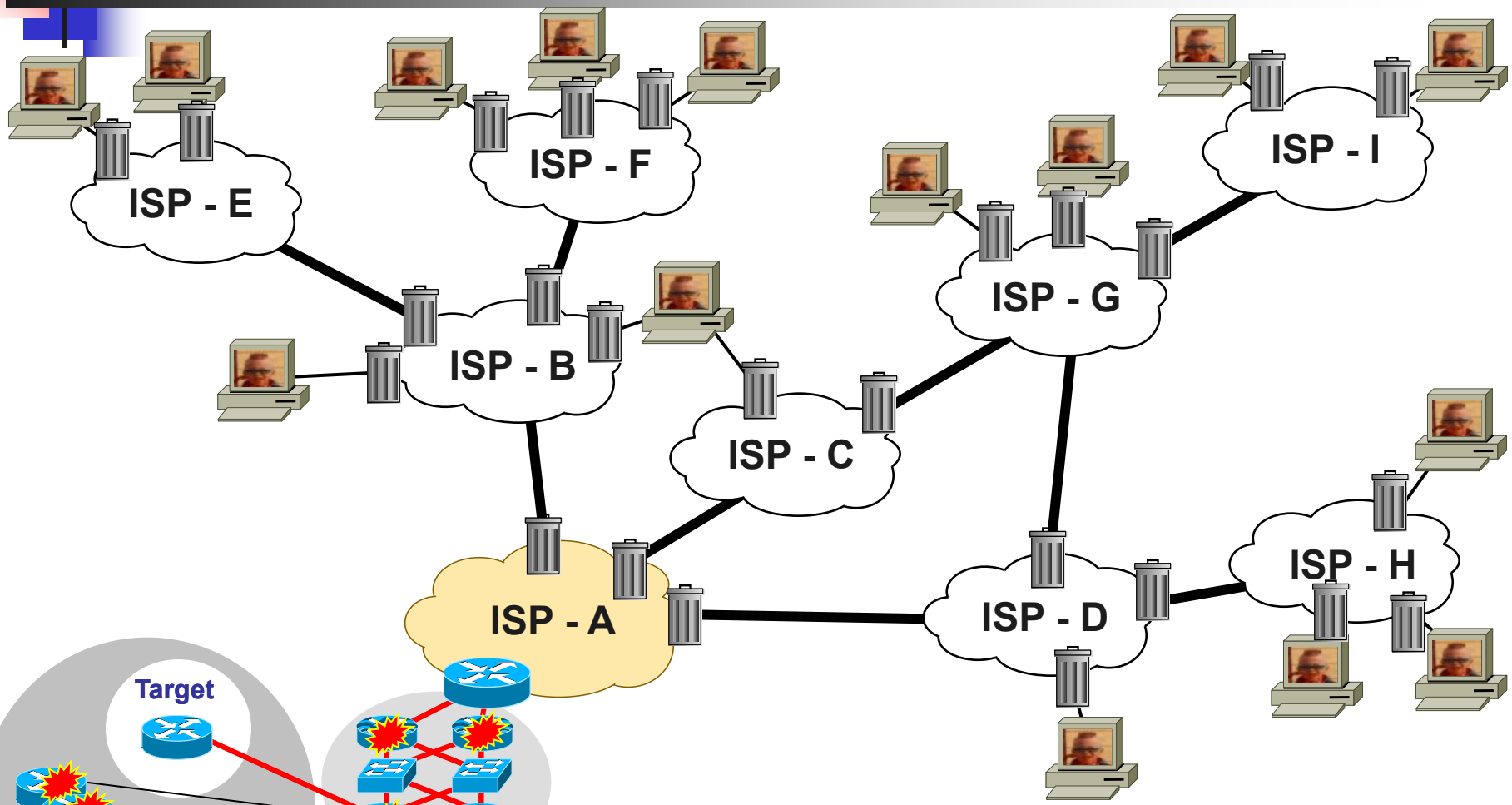
Customer is DOSed – After



Customer is DOSed – After – Packet Drops Pushed to the Edge



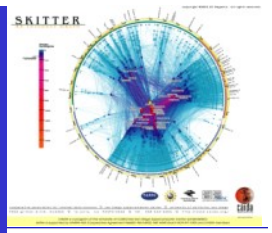
Inter-Provider Mitigation

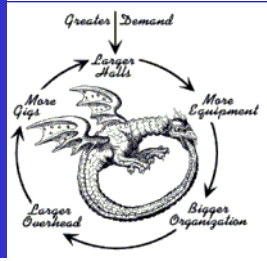


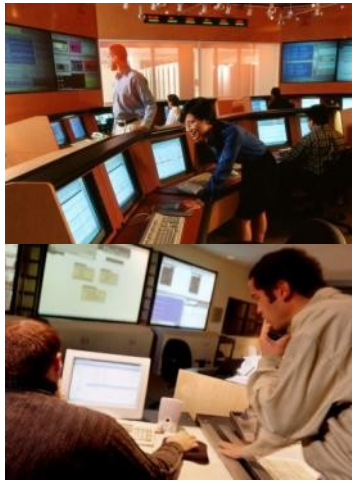
What can you do to help?

- Remote Triggered Black Hole Filtering is the most common ISP DOS/DDOS mitigation tool.
- Prepare your network:
  - ftp://ftp-eng.cisco.com/cons/isp/essentials/ (has whitepaper)
  - ftp://ftp-eng.cisco.com/cons/isp/security/ (has PDF Presentations)
  - NANOG Tutorial:
    - http://www.nanog.org/mtg-0110/greene.html (has public VOD with UUNET)

Sink Holes


SKITTER

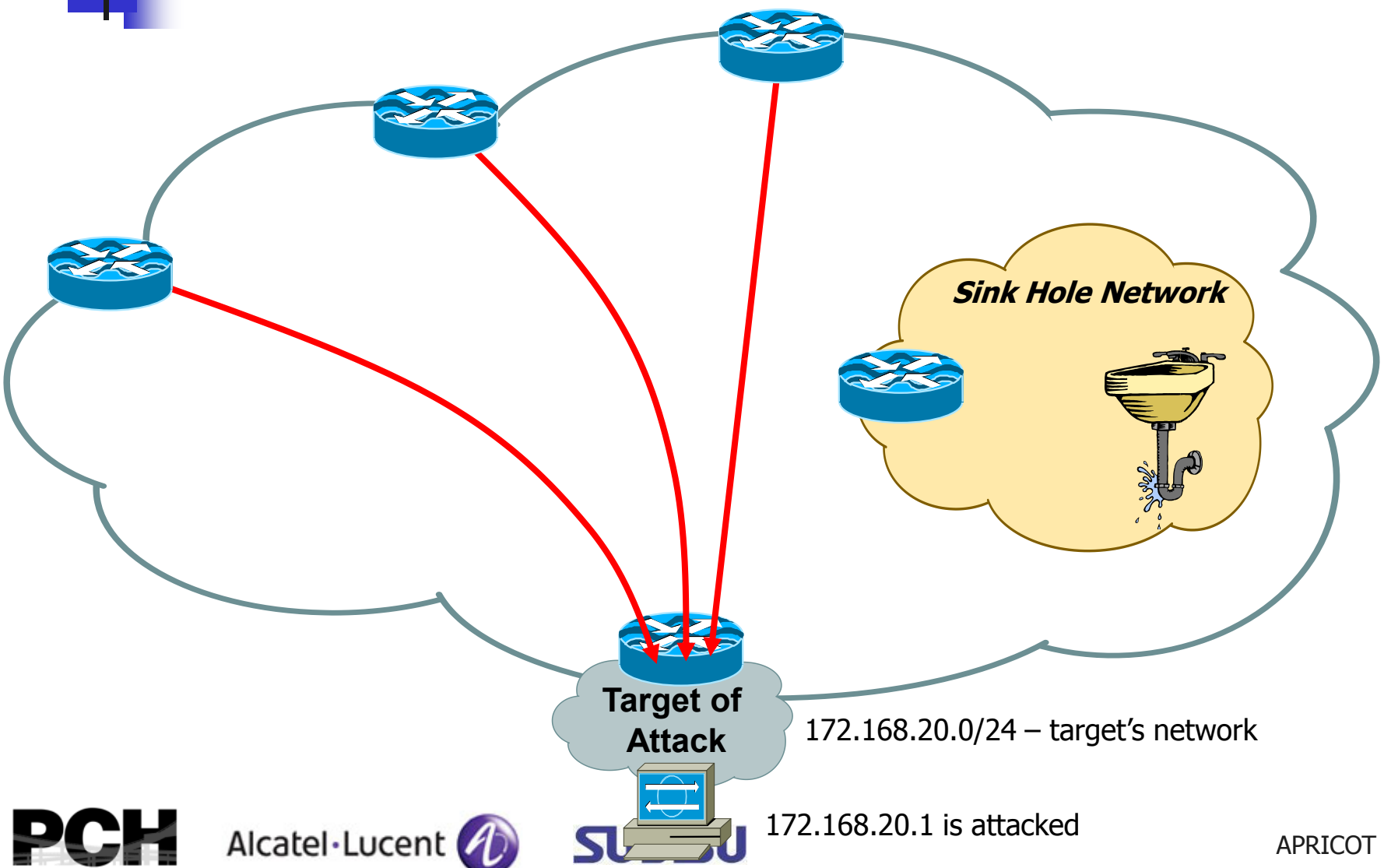

Greater Demand
Larger Halls
More Equipment
Bigger Organization
Larger Overhead
More Gigs



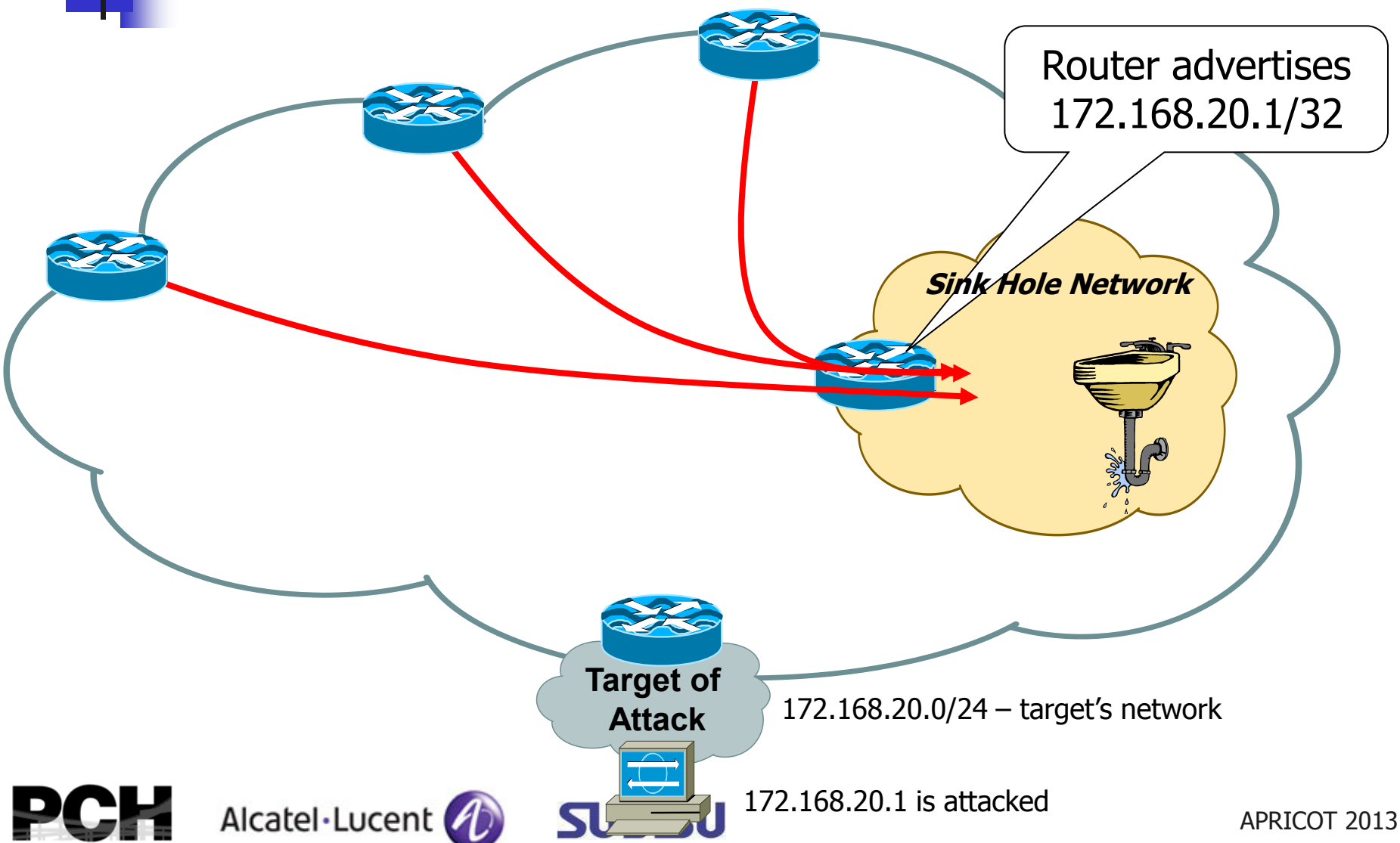
Sink Hole Routers/Networks

- Sink Holes are a *Swiss Army Knife* security tool.
  - BGP speaking Router or Workstation that built to *suck in* attacks.
  - Used to redirect attacks away from the customer – working the attack on a router built to withstand the attack.
  - Used to monitor *attack noise, scans,* and other activity (via the advertisement of default)
  - http://www.nanog.org/mtg-0306/sink.html

Sink Hole Routers/Networks

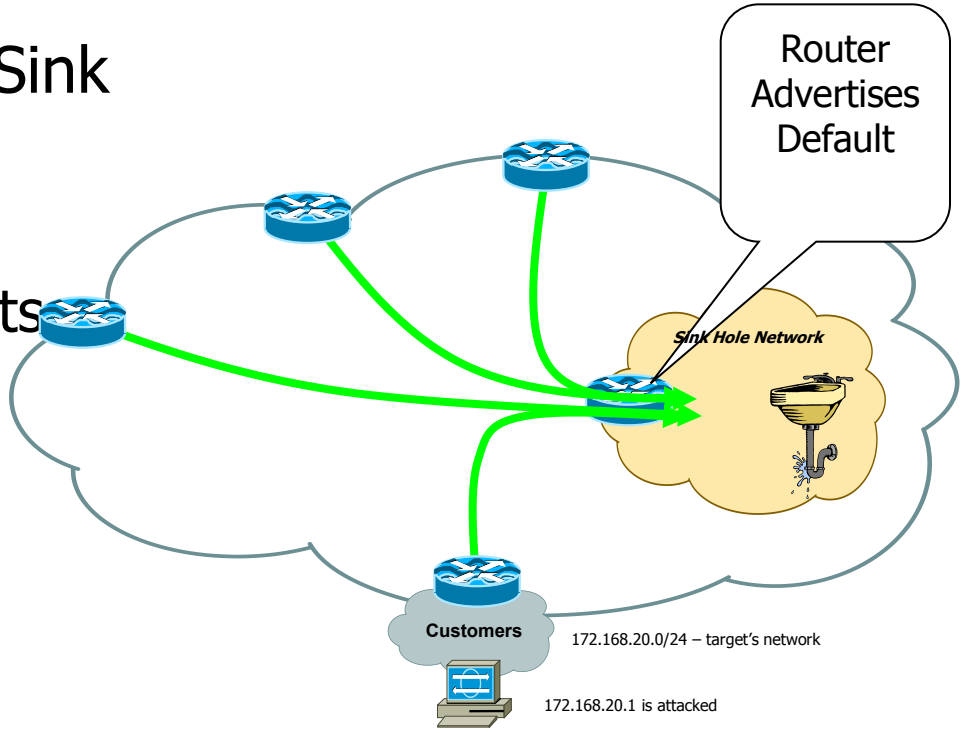


Sink Hole Routers/Networks

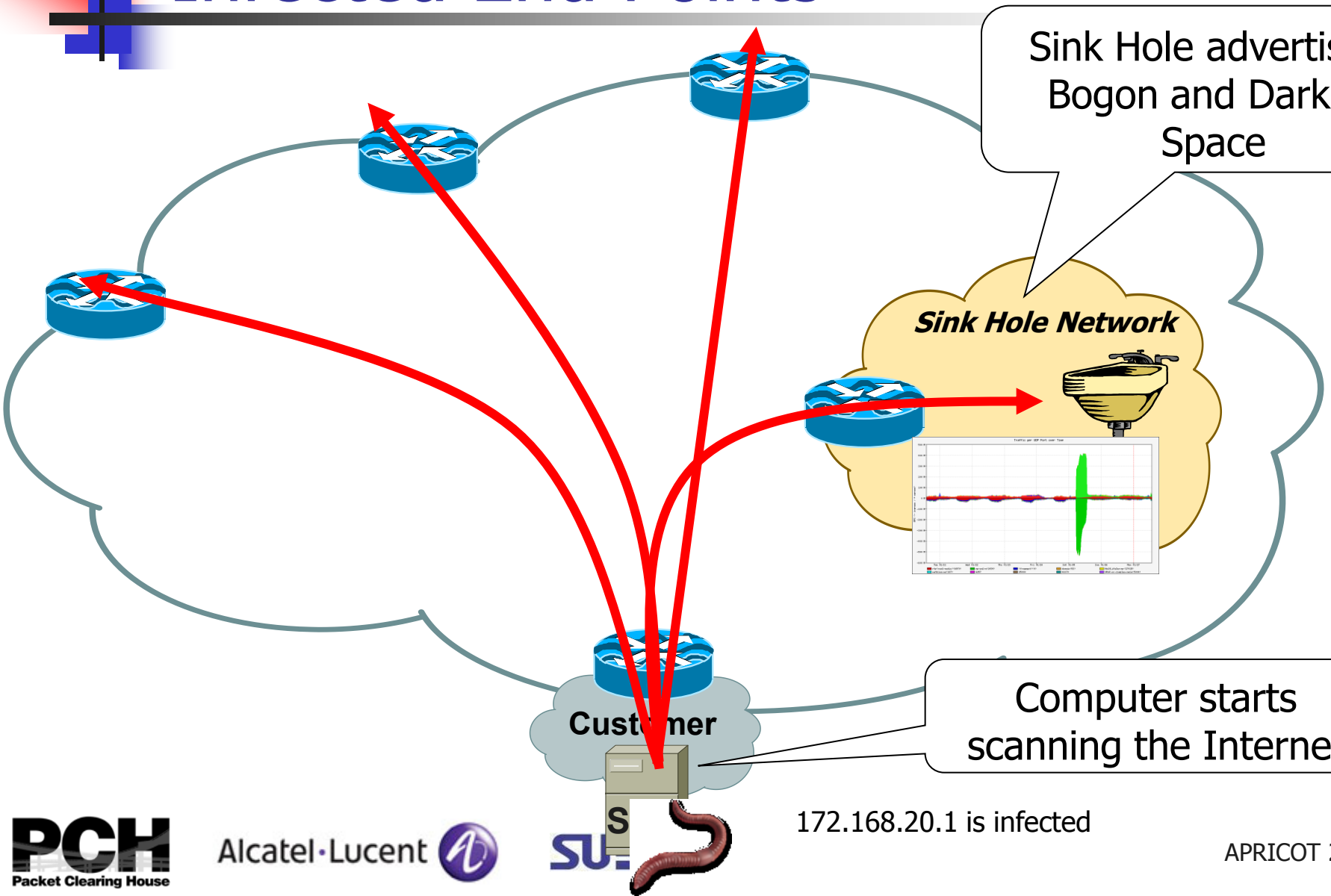


Sink Hole Routers/Networks

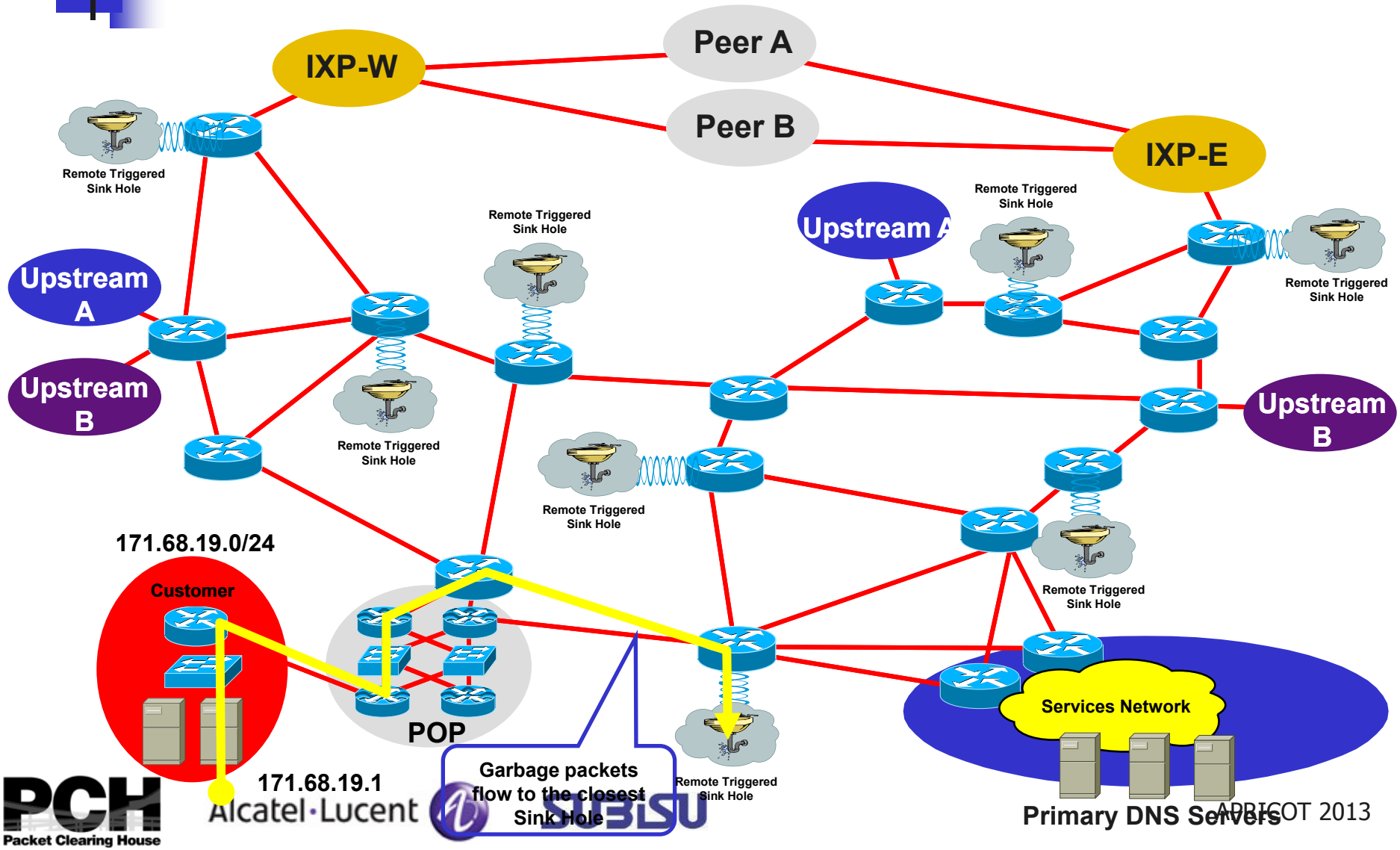
- Advertising Default from the Sink Hole will pull down all sort of *junk* traffic.
  - Customer Traffic when circuits flap.
  - Network Scans
  - Failed Attacks
  - Code Red/NIMDA
  - Backscatter
- Can place tracking tools and IDA in the Sink Hole network to monitor the noise.



Infected End Points



Anycast Sink Holes

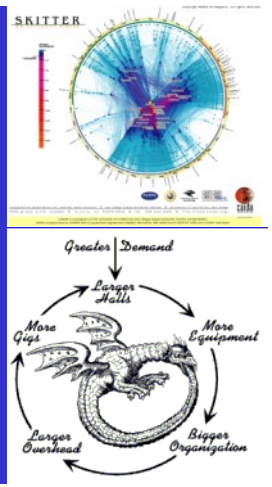


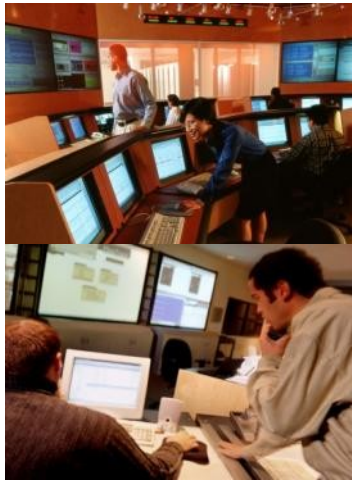
Filters, a.k.a.

access-lists

ip-filters

firewall filters





Filters

- Access-lists (a.k.a. filters, ip-filters, firewall filters) allow us to specify criteria to match upon and an action for packet forwarding decisions
- The concept of ACLs can be used in several approaches:
  - Classic ACLs, or transit ACLs
    - Permit traffic to **transit** the router
  - Infrastructure ACLs, or I-ACLs
    - Permit traffic **to** the router (and infrastructure)

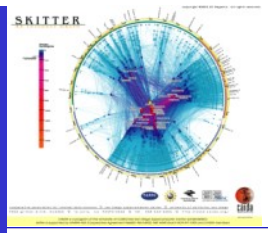
Packet filter applications

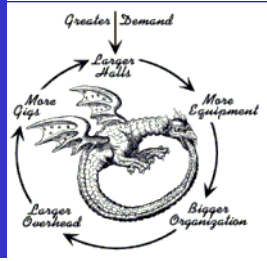
- Protect control plane (Routing Engine, Supervisor, RSP, Control Plane)
- Limit services
- Drop packets that don’t belong
  - Spoofed packets
  - uRPF perhaps better application

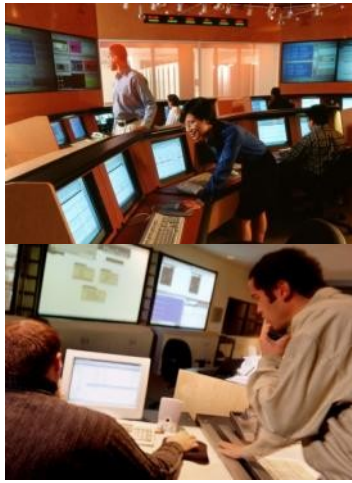
Access-list concept (IOS)

- `access-list <1-199> <permit|deny> <protocol> <source-address|any|host> <wildcard-bits> <dest-address|any|host> <dscp|fragments|log|log-input|option|prec|time-range|tos>`
- access-list 100 permit ip host 1.1.1.1 any log
- access-list 100 deny ip any host 2.2.2.2
- access-list 100 deny ip any any fragments

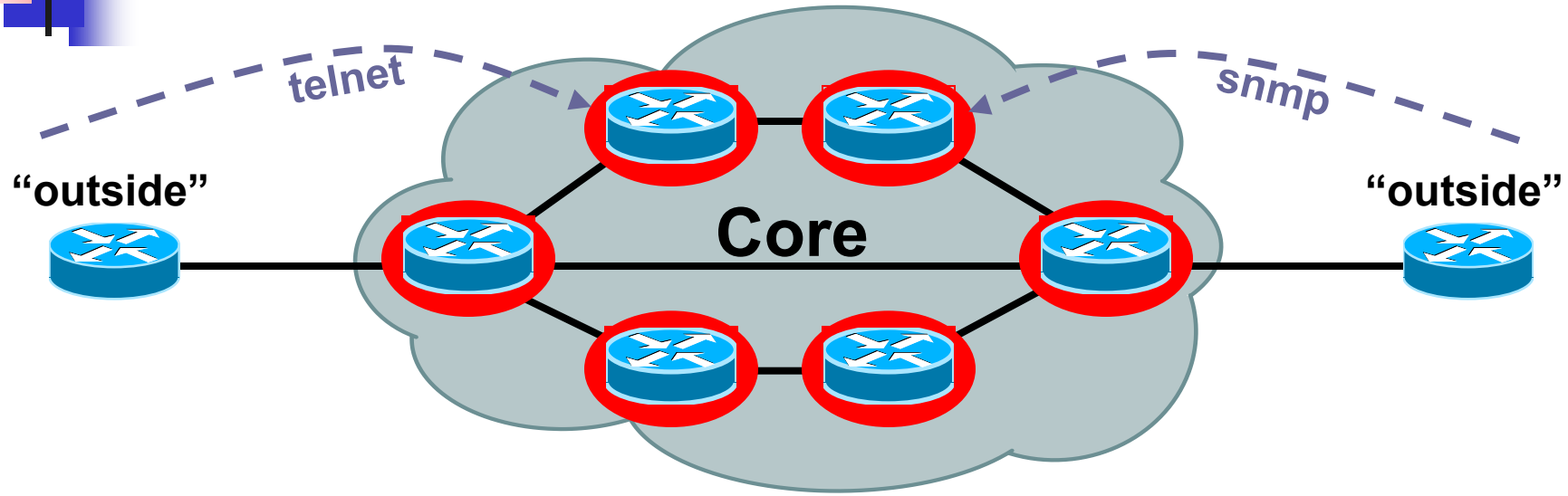
Edge Protection





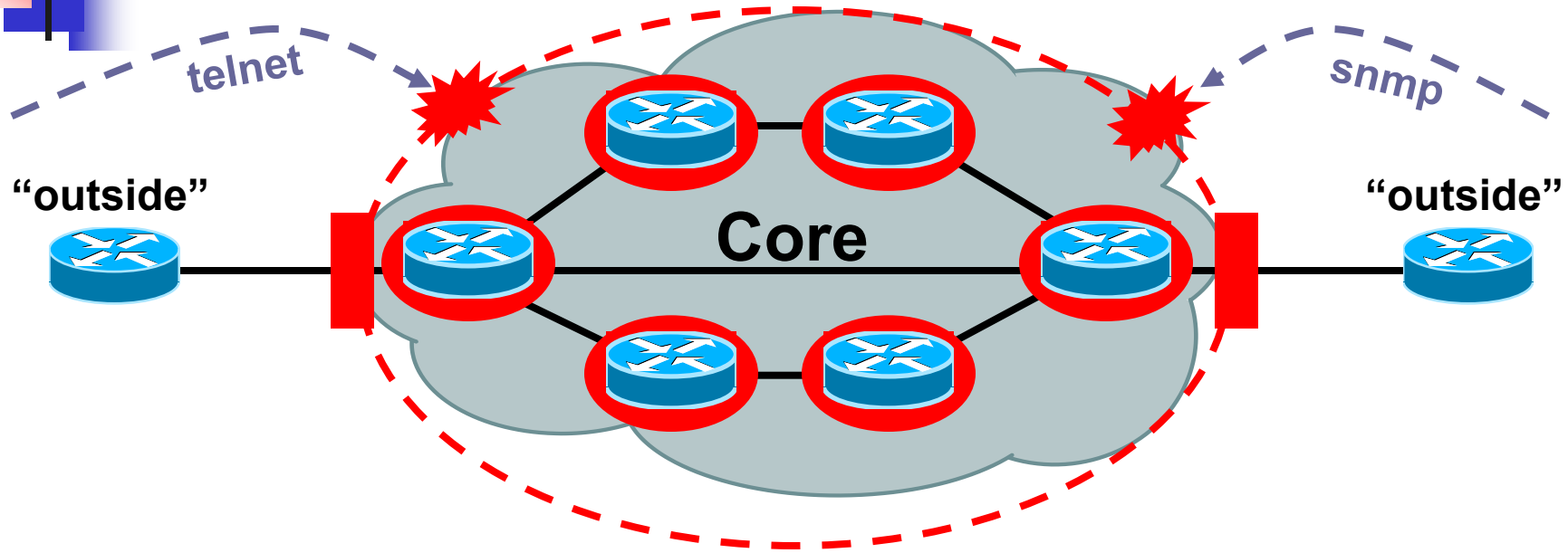


The Old World: Network Edge



- Core routers individually secured
- Every router accessible from outside

The New World: Network Edge



- Core routers individually secured PLUS
- Infrastructure protection
- Routers generally NOT accessible from outside

Infrastructure ACLs

- Basic premise: filter traffic destined TO your core routers
  - Do your core routers really need to process all kinds of garbage?
- Develop list of required protocols that are sourced from outside your AS and access core routers
  - Example: eBGP peering, GRE, IPSec, etc.
  - Use classification ACL as required
- Identify core address block(s)
  - This is the protected address space
  - Summarization is critical → simpler and shorter ACLs

Infrastructure ACLs

- Infrastructure ACL will permit only required protocols and deny ALL others to infrastructure space
- ACL should also provide anti-spoof filtering
  - Deny your space from external sources
  - Deny RFC1918 space
  - Deny multicast sources addresses (224/4)
  - RFC3330 defines special use IPv4 addressing

A Digression: IP Fragments and Security

- Fragmented Packets can cause problems...
  - Fragmented packets can be used as an attack vector to bypass ACLs
  - Fragments can increase the effectiveness of some attacks by making the recipient consume more resources (CPU and memory) due to fragmentation reassembly

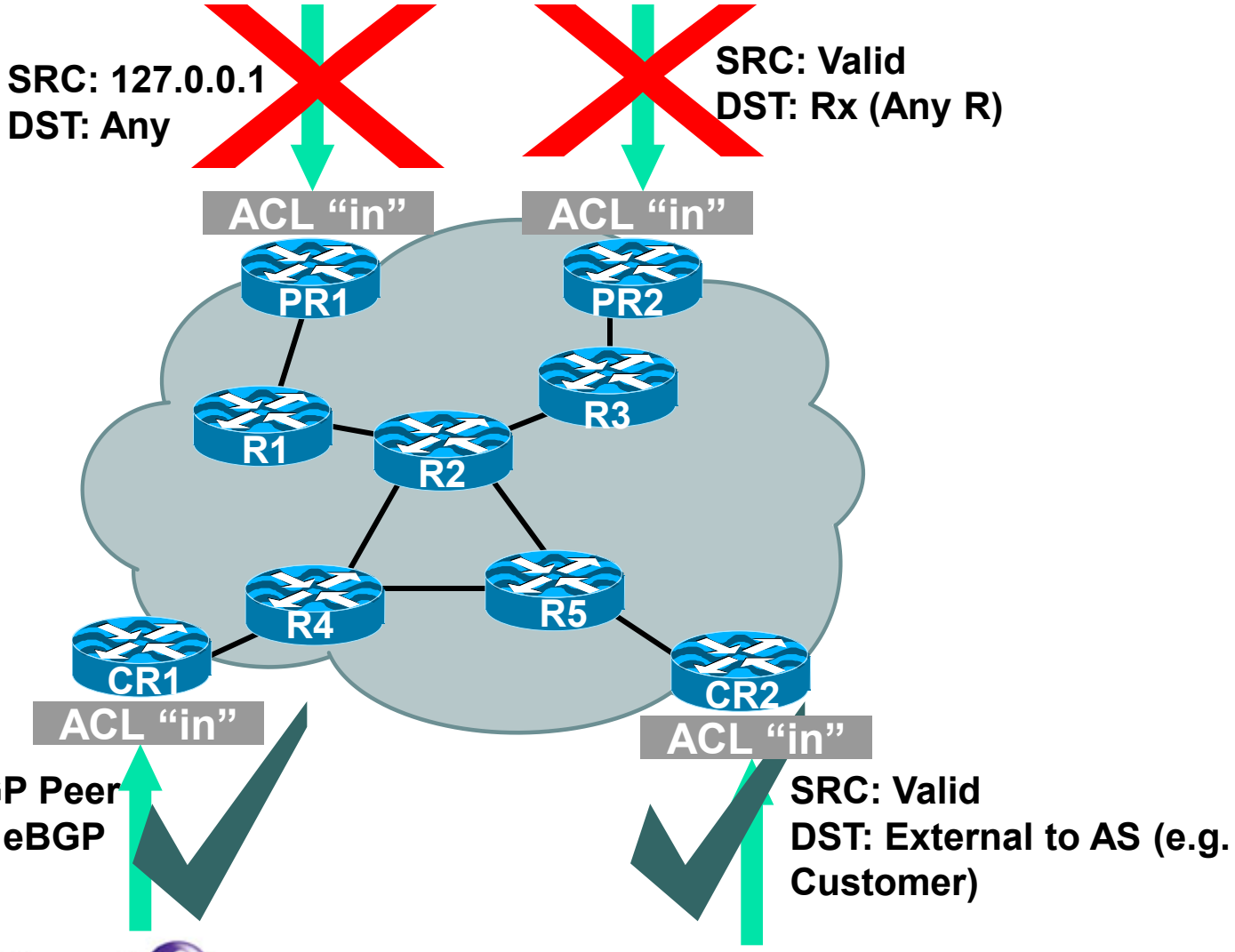
A Digression: IP Fragments and Security

- By default (without the fragments keyword)…
  - Initial fragments and non-fragmented packets
    - L3 ACLs— access control entry (ACE) action executed (permit/deny) available L3 information
    - L4 ACLs—ACE action executed (permit/deny) available L4 information
  - Non-initial fragment packets (assuming L3 match)
    - L3 ACLs—ACE action executed (permit/deny) available L3 information
    - L4 ACLs—ACE action executed (permit/deny) based on L3 info (there is no L4 info in the fragment) and protocol only
- The ACL fragments keyword enables specialized handling behavior
  - Initial fragments and non-fragmented packets
    - L3 and L4 ACLs—the packet does not match the entry since the fragment keyword is used. The packet then "falls through" to the next line(s)
  - Non-initial fragment packets (assuming L3 match)
    - With L3 and L4 ACLs—with an L3 match (and protocol matches the IP protocol), the action of the ACE is executed (permit/deny)

Infrastructure ACLs

- Infrastructure ACL must permit transit traffic
  - Traffic passing through routers must be allowed via permit IP any any
- ACL is applied inbound on ingress interfaces
- Fragments destined to the core can be filtered via fragments keyword

Infrastructure ACL in Action



IP Options

- Provide control functions that may be required in some situations but unnecessary for most common IP communications
- IP Options not switched in hardware
- Complete list and description of IP Options in RFC 791
- Drop and ignore reduce load on the route processor (RP)
- Caution: some protocols/application require options to function:
  - For example: strict/loose source routing, resource reservation protocols (RSVP) and others
- ip access-list extended drop-ip-option
  - deny ip any any option any-options
  - permit ip any any

IP Options

- ip options drop
- ip options ignore—router ignores options
  - Best practice when router doesn't need to process options
  - "ignore" not available on all routing platforms
  - Available in 12.0(22)S, 12.3(4)T and 12.2(25)S http://www.cisco.com/en/US/products/sw/iosswrel/ps1829/products_feature_guide09186a00801d4a94.html

Iterative Deployment

- Typically a very limited subset of protocols needs access to infrastructure equipment
- Even fewer are sourced from outside your AS
- Identify required protocols via classification ACL
- Deploy and test your ACLs

Step 1: Classification

- Traffic destined to the core must be classified
- NetFlow can be used to classify traffic
  - Need to export and review
- Classification ACL can be used to identify required protocols
  - Series of permit statements that provide insight into required protocols
  - Initially, many protocols can be permitted, only required ones permitted in next step
  - Log keyword can be used for additional detail; hits to ACL entry with log will increase CPU utilization: impact varies by platform
- Regardless of method, unexpected results should be carefully analyzed → do not permit protocols that you can't explain!

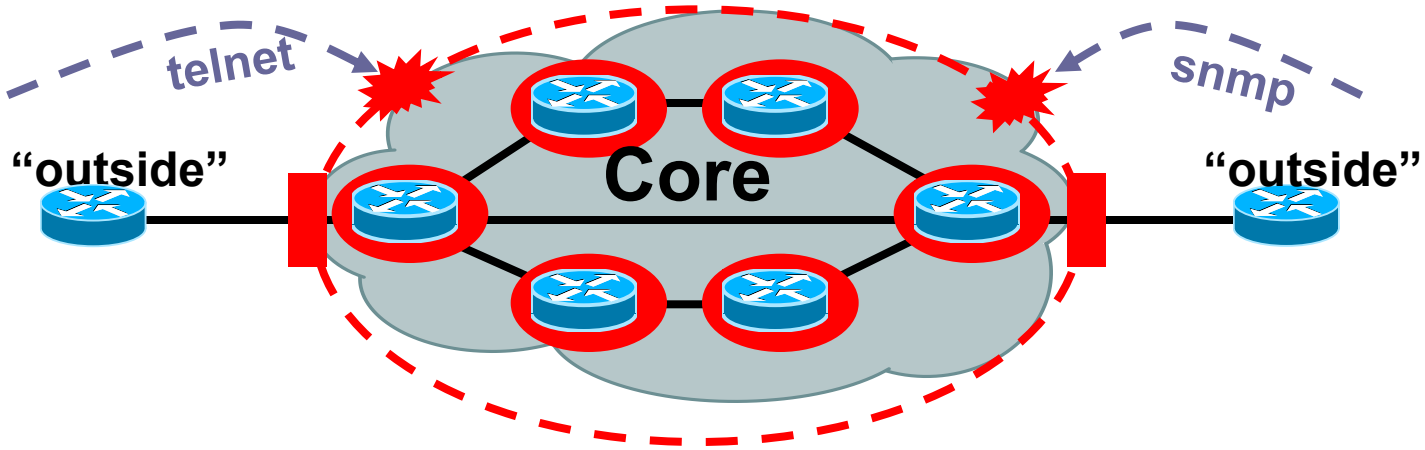
Step 2: Begin to Filter

- Permit protocols identified in step 1 to infrastructure only address blocks
- Deny all other to addresses blocks
  - Watch access control entry (ACE) counters
  - Log keyword can help identify protocols that have been denied but are needed
- Last line: permit ip any any ← permit transit traffic
- The ACL now provides basic protection and can be used to ensure that the correct suite of protocols has been permitted

Steps 3 and 4: Restrict Source Addresses

- Step 3:
  - ACL is providing basic protection
  - Required protocols permitted, all other denied
  - Identify source addresses and permit only those sources for requires protocols
  - e.g., external BGP peers, tunnel end points
- Step 4:
  - Increase security: deploy destination address filters if possible

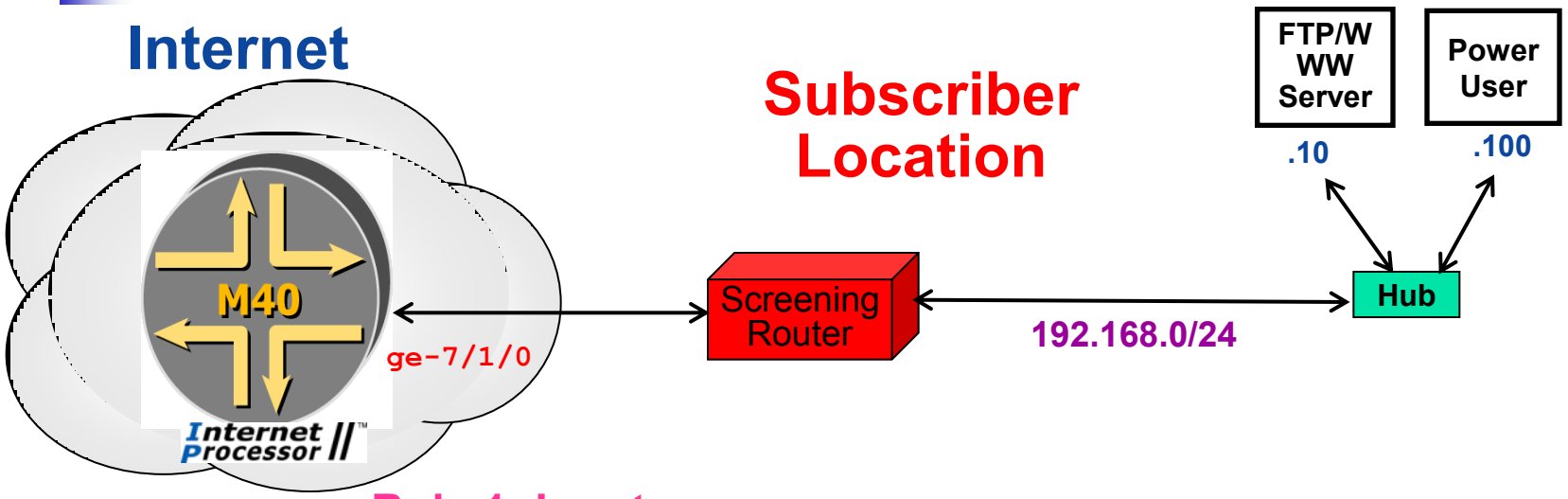
Infrastructure ACLs



- Edge "shield" in place
- Not perfect, but a very effective first round of defense
  - Can you apply iACLs everywhere?
  - What about packets that you cannot filter with iACLs?
  - Hardware limitations
- Next step: secure the control/management planes per box

Using packet filters

Spoof Prevention



Rule 1: Input
From SA = 192.168.0
Then Accept
From SA = other
Then Reject

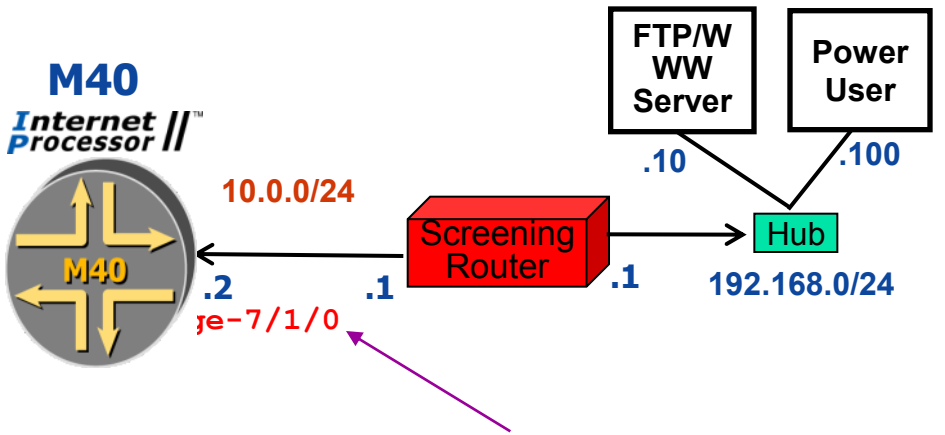
Rule 1 prevents the origination of spoofed packets from this site

Rule 2: Output
From SA = 192.168.0
Then Reject
From SA = other
Then Accept

Rule 2 blocks spoofed packets from entering this site

Inbound Spoof Prevention

```
[edit firewall family inet]
lab@router# show
filter no-spoofs-in {
    term allow-valid {
        from {
            source-address {
                192.168.0.0/24;
                10.0.0.0/24;
            }
        }
        then accept;
    }
    term reject {
        then {
            count bad-source-address;
            log;
            discard;
        }
    }
}
```

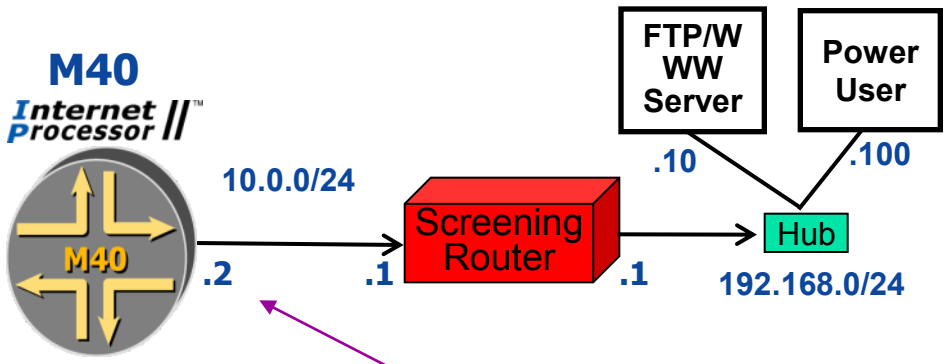



```
[edit interfaces ge-7/1/0]
lab@router# show
unit 0 {
    family inet {
        filter {
            input no-spoofs-in;
        }
        address 10.0.0.2/24;
    }
}
```

Preventing Fragmentation Exploits

```
[edit firewall]
lab@San_Jose-3# show
family inet {
    filter no-frags {
        term 1 {
            from {
                is-fragment;
                protocol [ icmp udp ];
            }
            then {
                count no-frags;
                log;
                discard;
            }
        }
        term 2 {
            then accept;
        }
    }
}
```

Permits diagnostic pings while blocking fragmented ICMP/UDP traffic
(For example, Teardrop, Boink, POD)

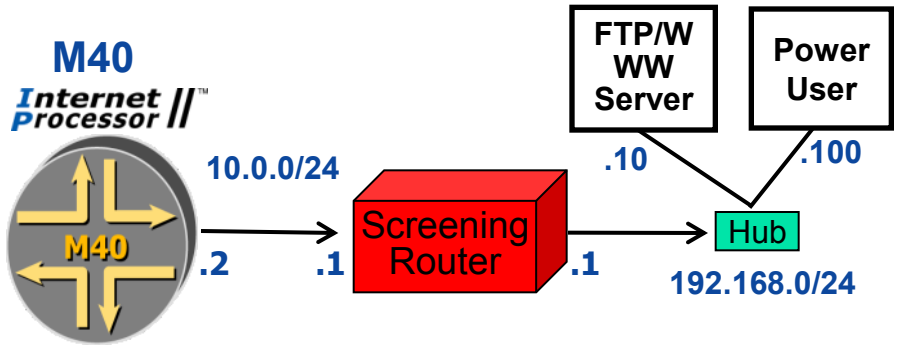


Filter applied in output direction of subscriber interface

```
[edit interfaces ge-7/1/0]
lab@router# show
unit 0 {
    family inet {
        filter {
            output no-frags;
        }
        address 10.0.0.2/24;
    }
}
```

Securing the FTP/WWW Server

```
[edit firewall family inet filter ftp-www-only]
lab@San_Jose-3# show
term allow-ftp-www {
    from {
        destination-address {
            192.168.0.10/32;
        }
        protocol tcp;
        destination-port [ ftp ftp-data http ];
    }
    then accept;
}
term reject-other {
    from {
        destination-address {
            192.168.0.10/32;
        }
    }
    then {
        count unauthorized-service-requests;
        log;
        discard;
    }
}
term accept-all {
    then accept;
}
```



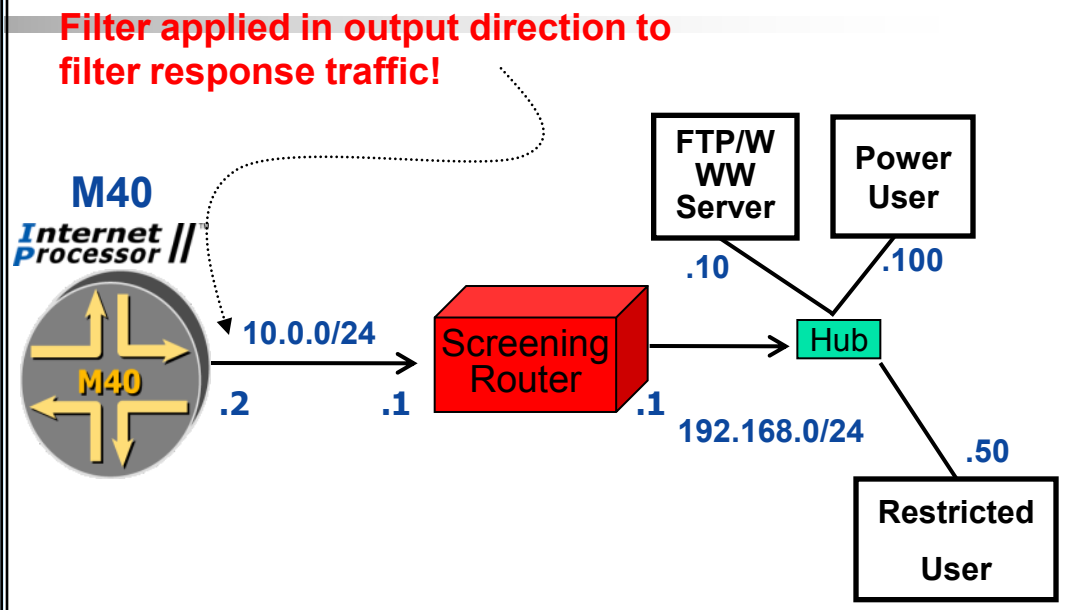

```
interfaces ge-7-1/0 {
    unit 0 {
        family inet {
            filter {
                output ftp-www-only;
            }
        }
    }
}
```

Filter applied in output direction of subscriber interface

Remember the implicit *deny all* for unmatched traffic!

Outgoing Service Restriction

```
[edit firewall family inet]
lab@San_Jose-3# show filter user-control
term normal-user-allow {
    from {
        destination-address {
            0.0.0.0/0;
            192.168.0.100/32 except;
        }
        protocol tcp;
        source-port http;
        tcp-established;
    }
    then accept;
}
term track-unauthorized {
    from {
        destination-address {
            0.0.0.0/0;
            192.168.0.100/32 except;
        }
    }
    then {
        count unauthorized;
        discard;
    }
}
term power-user {
    then accept;
}
```



Note the use of `except`, which exempts the power user from a particular terms in this filter

Rate Policing

- Instead of allowing or dropping packets that meet match conditions, you can use a filter to identify traffic that is to be policed (rate-limited)
  - You can apply a policer directly to an interface to rate-limit all traffic associated with that protocol family
- Traffic that matches the filter is then policed according to an average bandwidth and a burst size
  - Can specify bandwidth as a percentage of interface speed
- When traffic exceeds the policing parameters, it can:
  - Be discarded
  - Have its loss-priority (PLP) bit set
  - Be associated with a forwarding class (output queue)

Rate Policing Example

```
[edit firewall]
lab@router# show
policer p1 {
    if-exceeding {
        bandwidth-limit 400k;
        burst-size-limit 100k;
    }
    then discard;
}
family inet {
    filter limit-ftp {
        term ftp {
            from {
                source-address {
                    1.2.3.0/24;
                }
                protocol tcp;
                destination-port [ ftp ftp-data ];
            }
            then {
                policer p1;
                count count-ftp;
            }
        }
    }
}
```

- **Example:**
  - **`bandwidth-limit`**
    - **In bits per second**
    - **30,520 bps to 4.29 Gbps**
  - **`burst-size-limit`**
    - **In bytes per second**
    - **Min should = 10 times MTU (low speed) or bandwidth times 3–5 milliseconds (high speed)**
    - **Max = 16.7 Mb**

Lab

- Protecting the management plane of our routers is very important
- We need to restrict access to telnet/ssh to only permitted hosts

```
access-list 10 permit host <HOST>
access-list 10 permit 192.0.2.0 0.0.0.255
line vty 0 4
  access-class 10
```

Lab

- Protecting the management plane of our routers is very important
- We need to restrict access to telnet/ssh to only permitted hosts

```
access-list 10 permit host <HOST>
access-list 10 permit 192.0.2.0 0.0.0.255
line vty 0 4
  access-class 10
```

- **What hosts should you permit?**

Lab

- **What hosts should you permit?**
  - Your bastion/jump hosts
  - Your NOC/engineering subnets
- In the lab, permit your loopback addresses **of the whole pod**:

```
access-list 10 permit host 10.0.X.X
access-list 10 permit host 200.0.XX.1
line vty 0 4
  access-class 10 in
```

- Does telnet between routers still work?
- When does it work and not work?
  - `telnet <LOOPBACK-IP> /source-interface loopback0`

Infrastructure ACL

- Protecting the control plane of our routers is very important
- We need to restrict access to only the protocols we are using in the lab
- On the interface between PE1 and PE2, create a classification ACL:

```
ip access-list 100 permit ip any any log
interface Eth X/X
   ip access-group 100 in
```

- Monitor the log and identify protocols
- You can do this on CE1/CE2 as well, apply to the interface facing PE1/PE2

Infrastructure ACL

- Using the log entries, can you construct an Infrastructure access-list ?
- What protocols should be permitted in this ACL?

Infrastructure ACL

- Using the log entries, can you construct an Infrastructure access-list ?
- What protocols should be permitted in this ACL?
  - OSPF
  - BGP
  - telnet/ssh
- What should the final entry be?

Infrastructure ACL

- Using the log entries, can you construct an Infrastructure access-list ?
- What protocols should be permitted in this ACL?
  - OSPF
  - BGP
  - telnet/ssh
- What should the final entry be?
  - Permit any any!

Infrastructure ACL

```
access-list 101 permit ospf any any
access-list 101 permit tcp any any eq
  179
access-list 101 permit tcp any any eq 23
access-list 101 permit tcp any any eq 22
access-list 101 permit ip any any
```

Infrastructure ACL - strict

```
access-list 102 permit ospf host <PE1> host 224.0.0.5
access-list 102 permit ospf host <PE1> host 224.0.0.6
access-list 102 permit ospf host <PE1> host <PE2>
access-list 102 deny ospf any any
access-list 102 permit tcp host <PE1> host <PE2> eq 179
access-list 102 deny tcp any any eq 179
access-list 102 permit tcp host <BASTION> any eq 23
access-list 102 deny tcp any any eq 23
access-list 102 permit tcp host <BASTION> any eq 22
access-list 102 deny tcp any any eq 22
access-list 102 permit ip any any
```