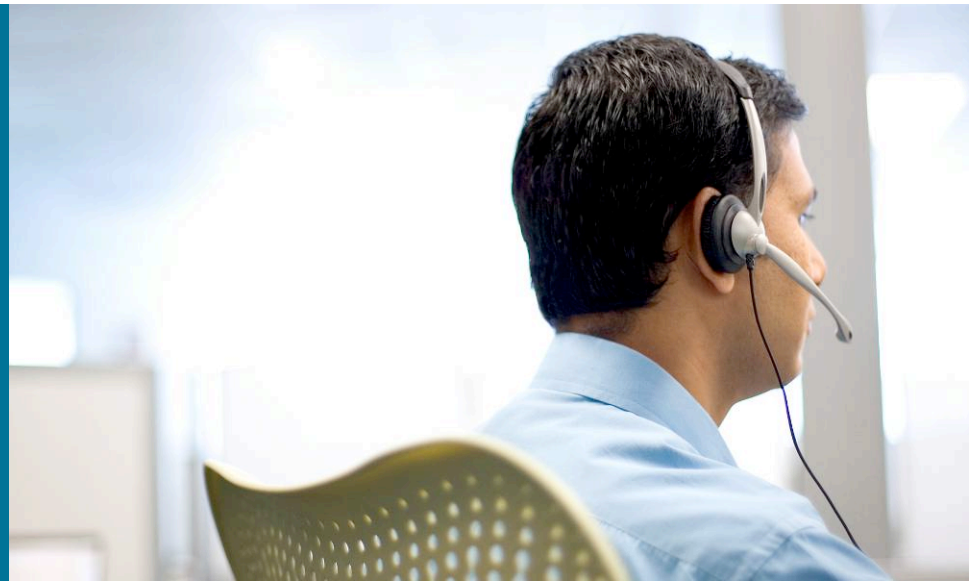
Infrastructure Security for Large Networks



Roland Dobbins <rdobbins@cisco.com>

Technology Solutions Engineering (TSE)

Part I
Listening to the Network

Introduction

The Importance of Detection and Classification

- In order to operate and ensure availability of the network, we must have the ability to detect undesirable network traffic and to classify it appropriately
- We cannot contain/mitigate what we cannot detect

  All the mitigation technology in the world isn't helpful if we've no visibility into threats to network availability

- Detection and classification must be part of the network architecture and operational security practice

  Otherwise, we're left scrambling to figure out what's happening—or even if anything is happening at all—instead of how what we're going to do about it

- In order to detect the abnormal, and possibly malicious, we have to know what's normal—we must establish a baseline of network activity, traffic patterns, etc.
- Classification is key—it provides the context for further action

Topics for Discussion

- In this session, we'll be talking about ways to 'listen to the network' by making use of specific features and tools
- Our aim is to explore some of the many ways to gain visibility into network behavior, integrating it into our operational security framework
- This is not a session on Intrusion Detection Systems (IDS)—there are plenty of IDS-specific tutorials and documentation available elsewhere. Instead, we'll be delving into other technologies and techniques
- We are at the intersection of 'traditional' information security and network operational security (opsec)—it is important to distinguish between policy/compliance, penetration attempts, and opsec
- This session is focused on opsec

Six-Phased Methodology for Incident Handling

1. Preparation
2. Detection/Identification
3. Classification
4. Traceback
5. Reaction
6. Post Mortem

This Session Covers Multiple Steps

- Preparation
- Detection/Identification
- Classification
- Traceback
- Reaction
- Post Mortem

Agenda

- Introduction to Detection/Classification Concepts
  - Principles of Detection
  - Telemetry—A Conceptual Overview
- Cisco IOS® CLI
  - Show process cpu
  - Show interface
  - Show ip interface
- Telemetry
  - NetFlow
  - DNS
  - BGP

Principles of Detection

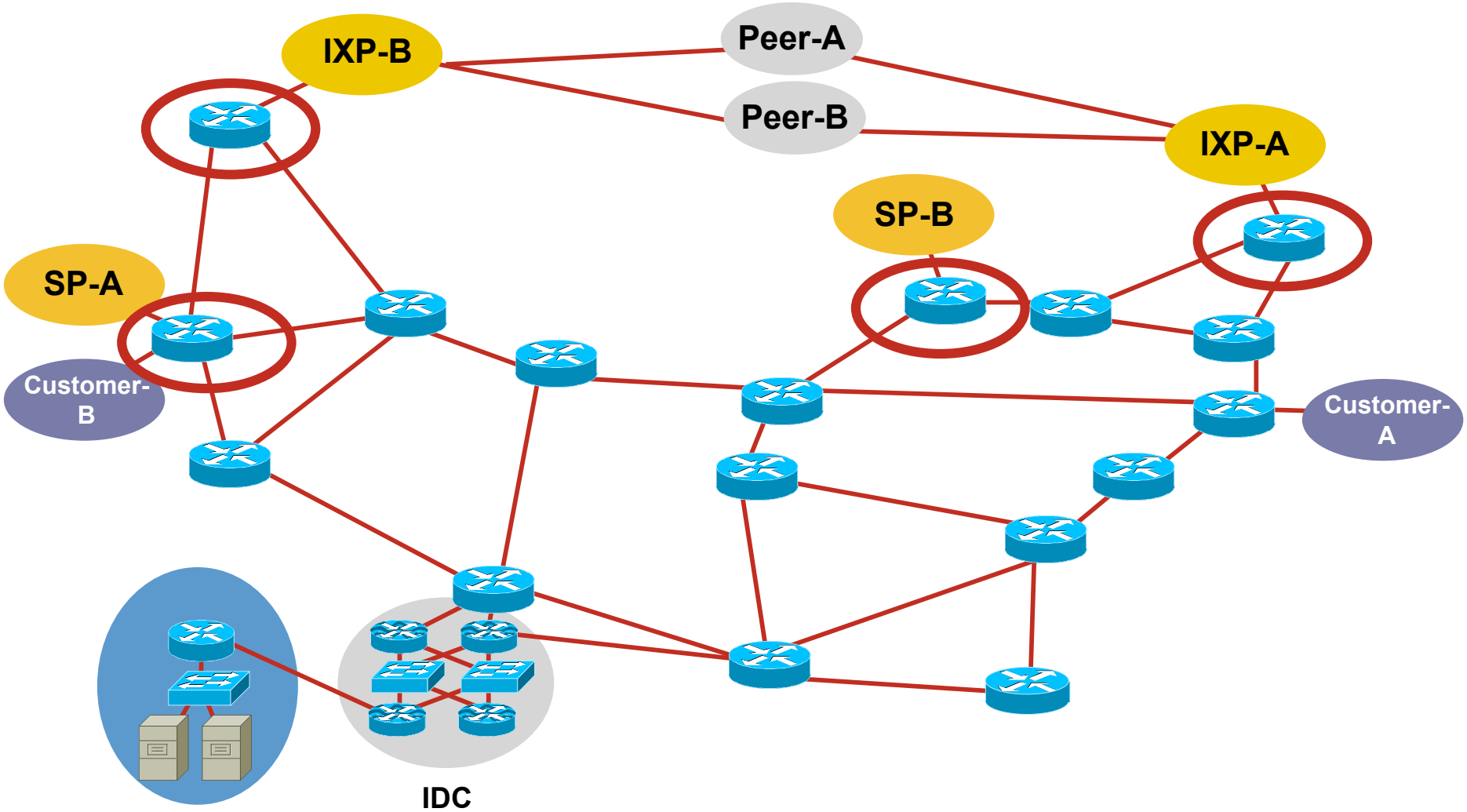
Principles of Detection

- When discussing detection, there are general principles which apply across the entire network topology
- There are also specific concepts and requirements which apply to various portions of the network
- We can characterize these groupings of concepts and requirements into principles which apply to each portion of the topology

Principles of Detection—Peering Edge

- At the peering edge, we're concerned with traffic ingressing and egressing our network
- SPs typically have more than one peer, as well as more than one peering edge router—this means that traffic may (and almost certainly is) routed asymmetrically
- Because of this asymmetry, it's important that the various methods and systems employed to detect and classify network traffic allow visibility across the peering edge
- This means we don't have to simulate the topology in our heads

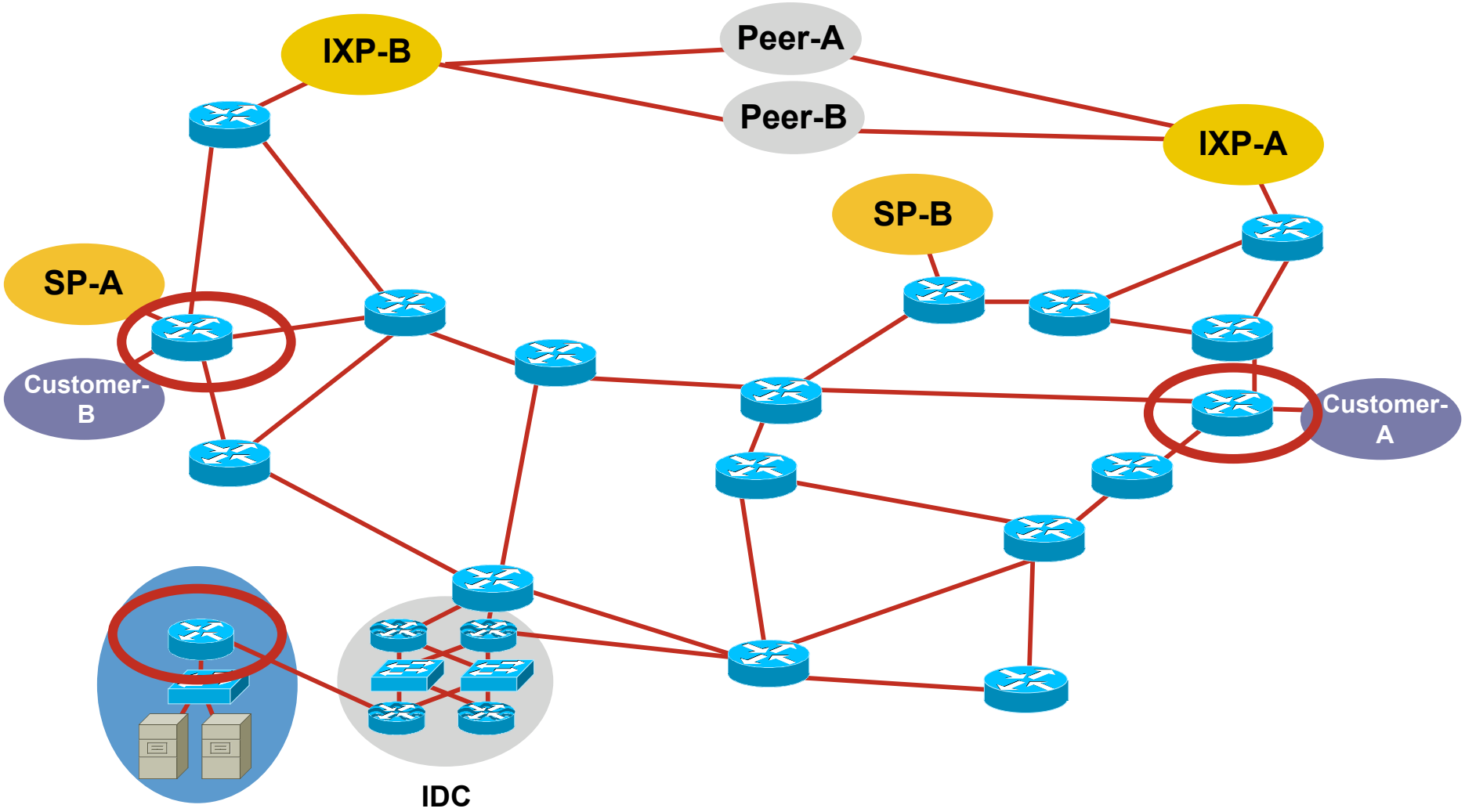
Principles of Detection—Peering Edge



Principles of Detection—Customer Edge

- While we care about traffic exiting our network towards the customer, the focus at the customer edge is on ingress
- Why? Because if interesting or undesirable traffic is heading downstream, we will detect it ingressing somewhere else
- From the SP standpoint, DoS, worms, from customer networks are problematic—they must be detected and mitigated as close to the ingress point as possible
- It is also important to protect the SP peering links and other SPs (and their customers) from attack

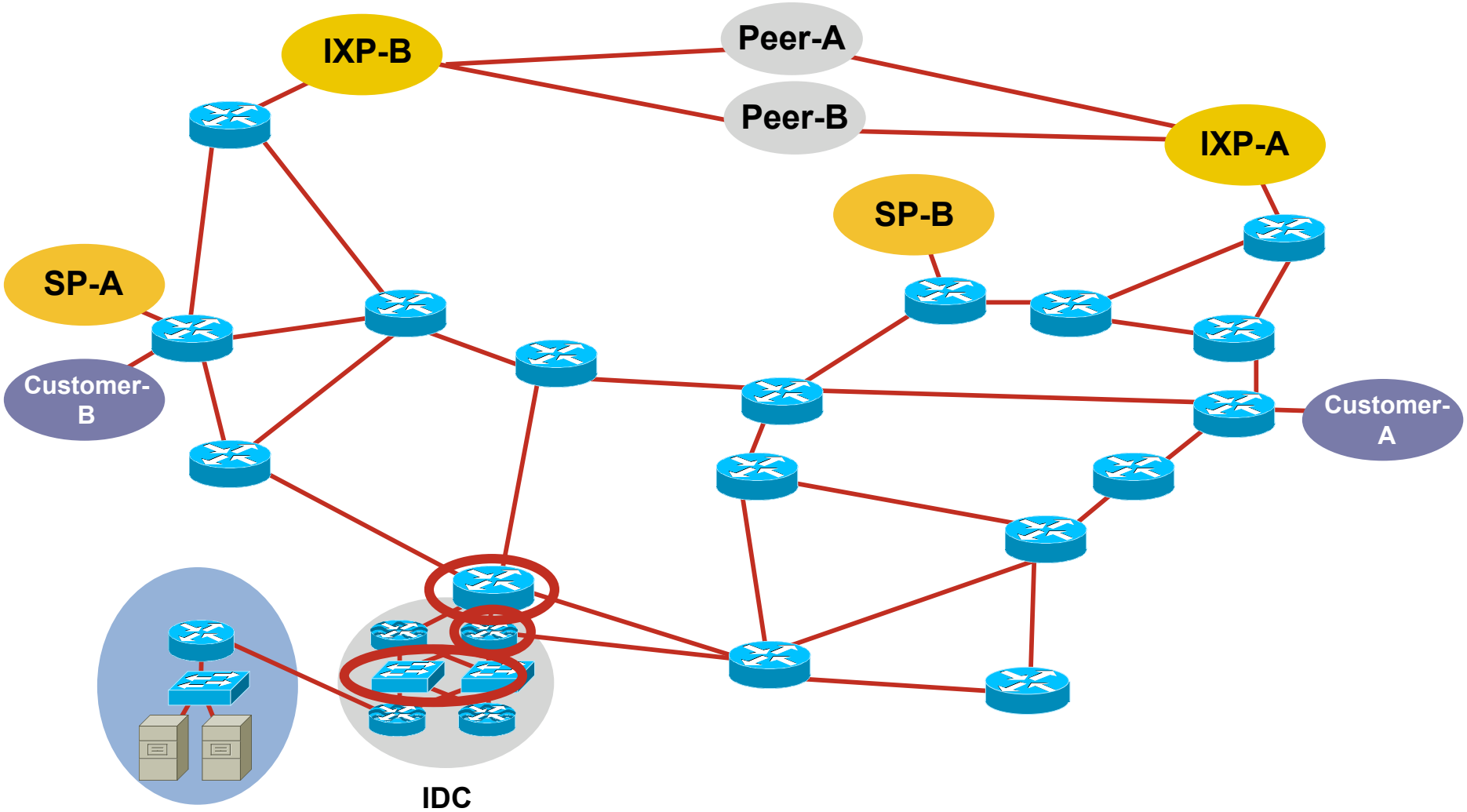
Principles of Detection—Customer Edge



Principles of Detection—IDC

- The Internet Data Center (IDC) is a vital part of the SP infrastructure—shared services such as DNS, status Web pages, etc., are often housed within the IDC
- Hosted and co-located customers also depend upon the IDC for connectivity to the world
- Because it is an edge of the network, we generally see topological symmetry at the interconnection between the IDC and the rest of the network—this gives us good bidirectional visibility
- We must have the capability to examine traffic between hosts within the IDC as well as traffic to/from the Internet

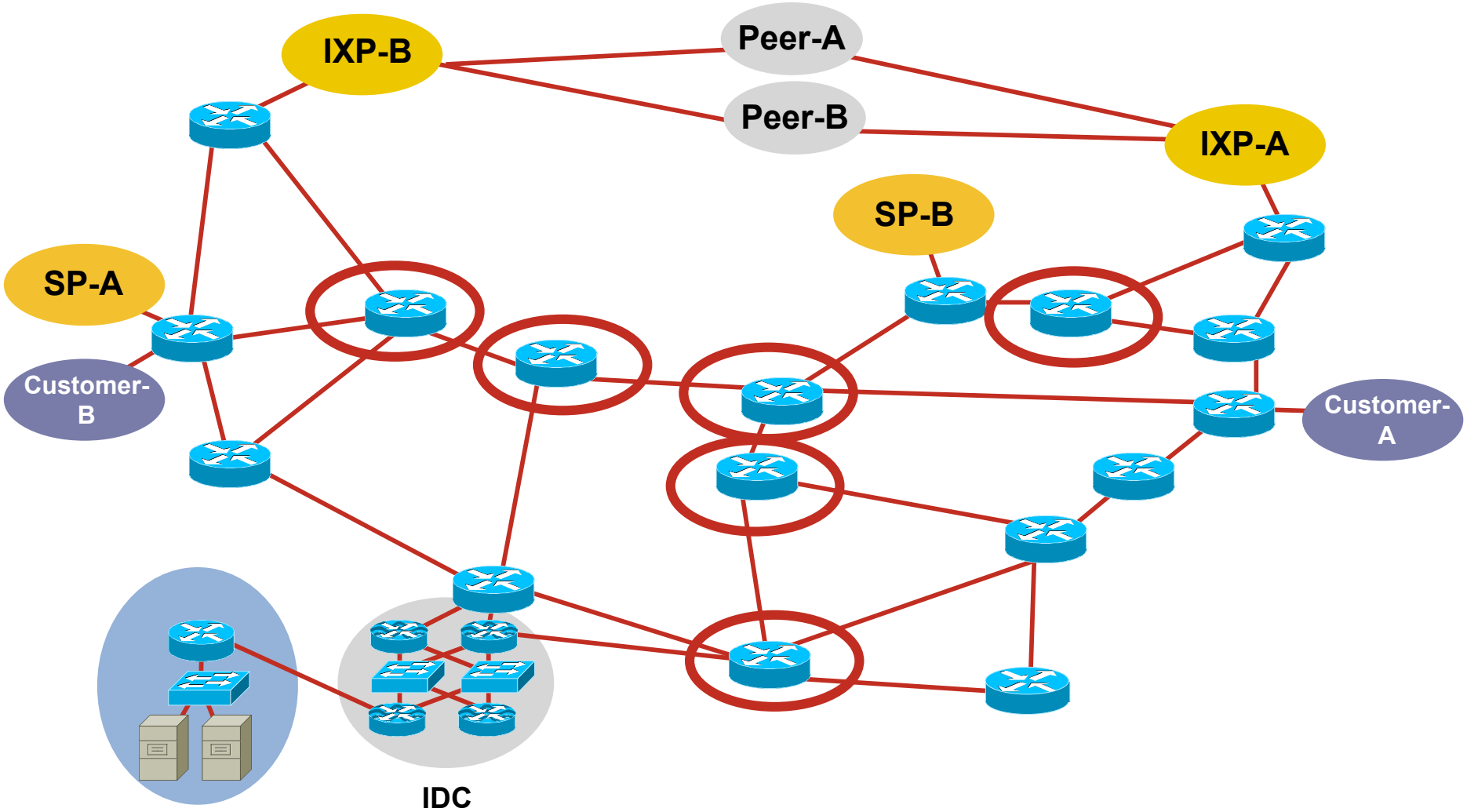
Principles of Detection—IDC



Principles of Detection—Core

- In the core, router health—CPU load, memory, etc.—is paramount. Problems in the core = problems elsewhere
- While we do not generally perform first-order detection and classification within the core itself, it is often necessary to trace traffic through the core to its origin(s) and/or destination(s)
- In many instances, we must examine the same traffic at various points in the topology in order to gain a clear understanding of a given event
- It is important that detection and classification technologies utilized within the core have minimal impact on performance

Principles of Detection—Core



How Do We Know if Something Is Happening?

“The Internet Is Down!”

- An all-too-common way of detecting problems—“The Internet is down!”
- This can be caused by a power outage, user error, misconfiguration of CPE, backhoe attack—or by DDoS/worms/etc.
- From the customer’s point of view, an outbound DoS launched from compromised systems can be just as devastating as an inbound attack from ‘the Internet’
- End-users and/or local admins often lack the instrumentation and personnel to accurately detect and classify network-related security events
- Because the SP’s business is the network itself, it is incumbent upon the SP to ensure these capabilities are present on the SP side of the last-mile hop
- Enterprises should be concerned about their side of the last-mile hop, remote WAN offices, etc.

Telemetry—A Conceptual Overview

What Is Meant by ‘Telemetry’?

Te·lem·e·try—*n.* The science and technology of automatic measurement and transmission of data by wire, radio, or other means from remote sources, as from space vehicles, to receiving stations for recording and analysis.

Source *The American Heritage® Dictionary of the English Language, Fourth Edition*

Network Telemetry

- Network telemetry offers extensive and useful detection capabilities
- This telemetry is often coupled with dedicated analysis systems to collect, trend, and correlate observed activity
- There are several forms of telemetry available from routers, switches, and other network devices
- There are a number of open source and commercial tools available which greatly enhance the utility of network telemetry
- Getting started with network telemetry is both inexpensive and relatively easy

Network Telemetry — Time Synchronization

- When dealing with network telemetry, it is important that dates and times are both accurate and synchronized
- Enabling Network Time Protocol (NTP) is the common method of time synchronization — it is supported by routers, switches, firewalls, hosts, and other network-attached devices
- Without time synchronization, it's very difficult to correlate different sources of telemetry
- More information on NTP can be found at

  http://www.ntp.org

Network Telemetry — OOB Management

- In-Band access to network infrastructure, hosts, etc., works very well — until there's a problem on the network
- In order to maximize reachability of and control over the network even during disruptive events, it is necessary to build an isolated Out-of-Band (OOB) management network
- Many devices such as routers and switches have serial console ports; others have Ethernet management interfaces
- Transmitting network telemetry over the OOB network minimizes the chance for disruption of the very information which gives us network visibility

Network Telemetry — Antispoofing

- There are many mechanisms available in modern network infrastructure devices to disallow spoofed traffic from transiting the network - Unicast Reverse Path Forward (uRPF), DHCP Snooping with IP Source Guard, Cable IP Source Verify, etc.
- Spoofed traffic is by definition invalid traffic - there is no reason to allow spoofed traffic to ingress and transit your network. Disallowing spoofed traffic is a basic step in improving network resiliency
- By eliminating spoofed traffic, we remove clutter from the ‘data horizon’ generated by analyzing network telemetry
- This greatly reduces the traceback problem - with antispooing measures in place, we know that purported source IPs originating from network edges under our control are valid, and we eliminate bogon-sourced traffic from the peering edge

The Cisco IOS CLI

Cisco IOS CLI—`sh process cpu (sh proc c)`

- A basic indication of a potential issue is high CPU on a router
- The `sh proc c` command gives information about CPU utilization in 5-second, 1-minute, and 5-minute bins
- High values for the IP Input process is a good indicator that traffic ingressing/egressing the router is contributing meaningfully to CPU load
- The amount of process-driven traffic vs. interrupt-driven traffic is also important

Example—sh proc c Plus Exclude (e)

```
7600>show proc c | e 0.00%__0.00%__0.00%
CPU utilization for five seconds: 38%/26%; one minute: 40%; five minutes: 43%
 PID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min TTY Process
   5   192962596  13452649      14343  0.00%  0.52%  0.44%   0 Check heaps
  15   4227662201540855414        274  0.65%  0.50%  0.49%   0 ARP Input
  26   2629012683680473726         71  0.24%  0.29%  0.36%   0 Net Background
  50     9564564  11374799        840  0.08%  0.07%  0.08%   0 Compute load avg
  51    15291660    947844      16133  0.00%  0.03%  0.00%   0 Per-minute Jobs
  58    15336356  92241638        166  0.08%  0.02%  0.00%   0 esw_vlan_stat_pr
  67    10760516 506893631         21  0.00%  0.01%  0.00%   0 Spanning Tree
  68  31804659682556402094       1244  7.02%  7.04%  7.75%   0 IP Input
  69    25488912  65260648        390  0.00%  0.02%  0.00%   0 CDP Protocol
  73    16425564  11367610       1444  0.08%  0.02%  0.00%   0 QOS Stats Export
  81    12460616   1020497      12210  0.00%  0.02%  0.00%   0 Adj Manager
  82   442430400  87286325       5068  0.65%  0.73%  0.74%   0 CEF process
  83    68812944  11509863       5978  0.00%  0.09%  0.11%   0 IPC LC Message H
  95    54354632  98373054        552  0.16%  0.12%  0.13%   0 DHCPD Receive
  96    61891604  58317134       1061  1.47%  0.00%  4.43%   0 Feature Manager
```

Example—sh proc c

```
7600>sh proc c | e 0.00
CPU utilization for five seconds: 41%/26%; one minute: 46%; five minutes: 44%
 PID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min TTY Process
  15   4227657321540854233        274  0.40%  0.39%  0.47%   0 ARP Input
  26   2629008963680468704         71  0.08%  0.36%  0.39%   0 Net Background
  50      9564512  11374786        840  0.08%  0.07%  0.08%   0 Compute load avg
  68  31804578042556183430       1244  9.65%  8.49%  7.75%   0 IP Input
  69     25488888  65260576        390  0.32%  0.05%  0.01%   0 CDP Protocol
  82    442429604  87286223       5068  0.73%  0.73%  0.74%   0 CEF process
  83     68812848  11509849       5978  0.32%  0.13%  0.12%   0 IPC LC Message H
  95     54354508  98372867        552  0.16%  0.12%  0.13%   0 DHCPD Receive
  96     61891268  58317034       1051  2.94%  2.05%  5.40%   0 Feature Manager
 171     22376816 154769997        144  0.16%  0.05%  0.05%   0 IGMP Input
 175          624        92       6782  0.57%  0.49%  0.16%   1 SSH Process
```

Example—sh fm sum

```
7600>sh fm sum
Current global ACL merge algorithm: ODM
ODM optimizations enabled
Interface: GigabitEthernet1/1 is up
 ACL merge algorithm used:
  inbound direction: ODM
  outbound direction: ODM
 TCAM screening for features is ACTIVE outbound
Interface: GigabitEthernet1/2 is up
 ACL merge algorithm used:
  inbound direction: ODM
  outbound direction: ODM
 TCAM screening for features is INACTIVE outbound
```

Cisco IOS CLI—`sh proc c` (Cont.)

- There are processes which are platform-specific —i.e., `Feature Manager` is found on the 6500/7600 only, while `IPC CBus` is 7500-specific
- Aliasing the more complex sh proc c commands to a single-letter alias as part of the standard config is extremely useful when the box is under high load and it's hard to type on the console:

```
Router(config)#alias exec p show proc c | e 0.00%__0.00%__0.00%
```

- Understanding your platform(s), and what's normal—including periodically-run processes (`BGP Scanner`, for example)—is key
- On the 12000, one must either `attach` to a linecard or perform an `execute` command specifying a linecard in order to see its CPU load; on the 7500, one uses the `if-con` command to session to a VIP

Cisco IOS CLI—`show interface (sh int)`

- `sh int` displays interface-level statistics, including throughput (pps) and bandwidth (bps)
- Typically, routers are set to use a 5-minute decaying average for interface statistics by default—changing this to 1 minute gives more granular statistics
- Looking for high input/output rates over a period of a minute or so can be very helpful
- Clearing the counters is often necessary to see what's happening now—be sure you aren't discarding useful info before you do it

Example—sh int gi3/13

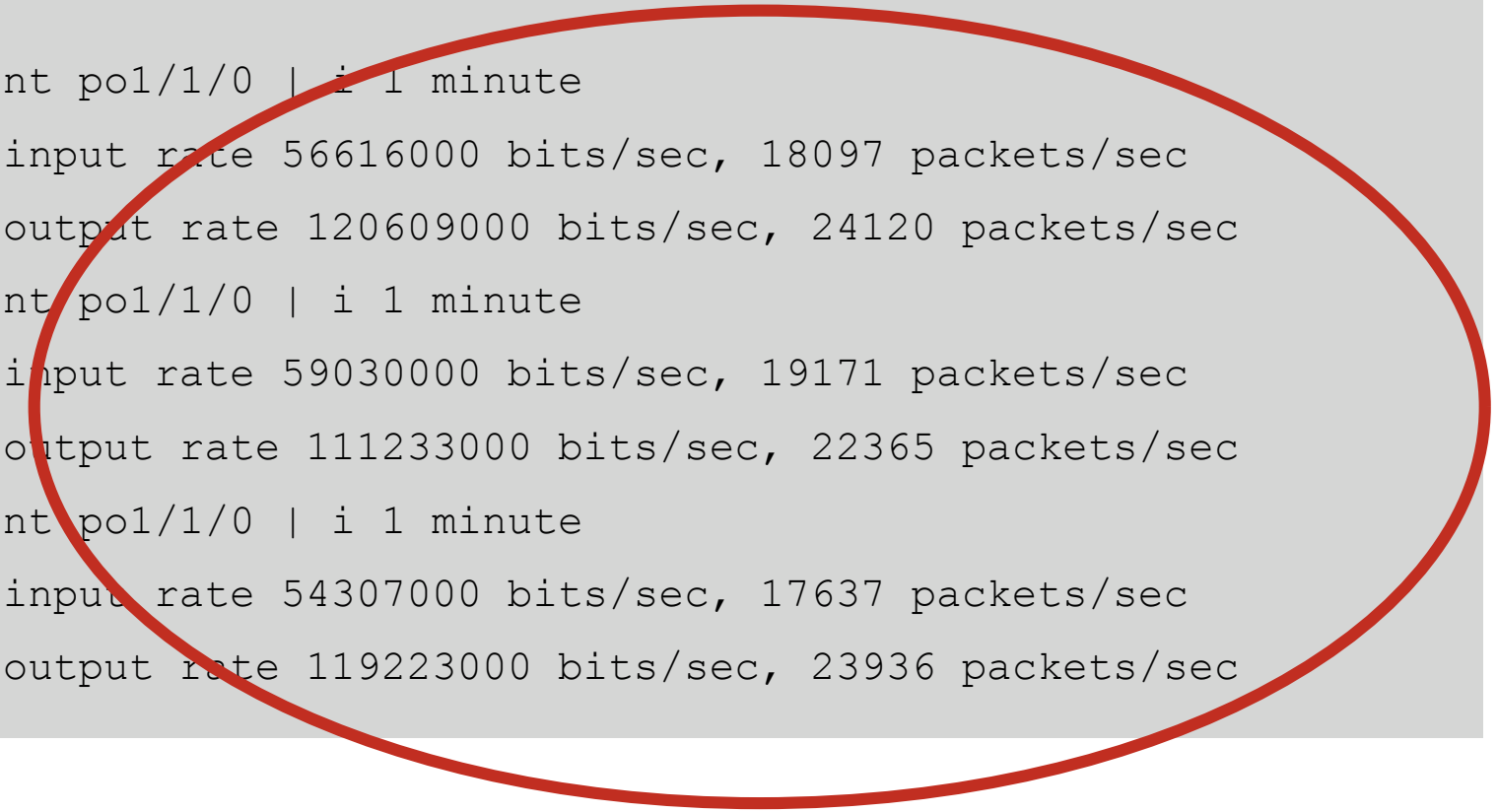
```
GigabitEthernet3/13 is up, line protocol is up (connected)
 Hardware is C6k 1000Mb 802.3, address is 00d0.0136.0001 (bia
  00d0.0136.0001)
 Description: IP TELEPHONY
 Internet address is 10.98.202.130/26
 MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
    reliability 255/255, txload 1/255, rxload 1/255
 Encapsulation ARPA, loopback not set
 Keepalive set (10 sec)
 Full-duplex mode, link type is autonegotiation, media type is SX
 output flow-control is unsupported, input flow-control is unsupported,
  1000Mb/s
 Clock mode is auto
 input flow-control is off, output flow-control is off
 ARP type: ARPA, ARP Timeout 04:00:00
 Last input 00:00:00, output 00:00:00, output hang never
 Last clearing of "show interface" counters 1y39w
```

Example—sh int gi3/13 (Cont.)

```
Input queue: 0/75/15005/235 (size/max/drops/flushes); Total output drops: 1
 Queueing strategy: fifo
 Output queue :0/40 (size/max)
 5 minute input rate 4751000 bits/sec, 3006 packets/sec
 5 minute output rate 4499000 bits/sec, 2755 packets/sec
 L2 Switched: ucast: 19841909032 pkt, 3347755205145 bytes - mcast: 96885779 pkt, 5131184435
   bytes
 L3 in Switched: ucast: 27282638229 pkt, 5095662463006 bytes - mcast: 94 pkt, 5191 bytes
   mcast
 L3 out Switched: ucast: 43107617667 pkt, 7275264441541 bytes
    47118207406 packets input, 9306459456266 bytes, 0 no buffer
    Received 83653389 broadcasts, 0 runts, 0 giants, 0 throttles
    0 input errors, 0 CRC, 0 frame, 649 overrun, 0 ignored
    0 input packets with dribble condition detected
    43210876182 packets output, 8089398934796 bytes, 0 underruns
    0 output errors, 0 collisions, 0 interface resets
    0 babbles, 0 late collision, 0 deferred
    0 lost carrier, 0 no carrier
    0 output buffer failures, 0 output buffers swapped out
```

Example—sh int Plus include (i)

```
12000>sh int po1/1/0 | i 1 minute
 1 minute input rate 56616000 bits/sec, 18097 packets/sec
 1 minute output rate 120609000 bits/sec, 24120 packets/sec
12000>sh int po1/1/0 | i 1 minute
 1 minute input rate 59030000 bits/sec, 19171 packets/sec
 1 minute output rate 111233000 bits/sec, 22365 packets/sec
12000>sh int po1/1/0 | i 1 minute
 1 minute input rate 54307000 bits/sec, 17637 packets/sec
 1 minute output rate 119223000 bits/sec, 23936 packets/sec
```

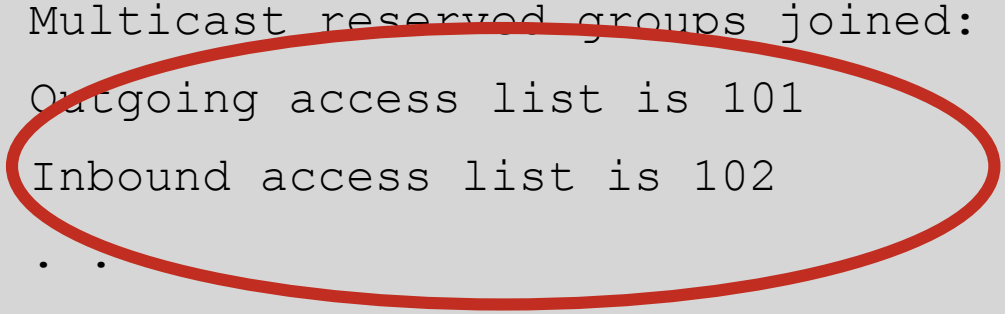


Cisco IOS CLI—`sh ip int`

- `sh ip int` gives information about features configured on an interface
- It's useful to get the number or name of an ACL in order to check ACL counter hits (6500/7600 only shows ACL counters on Sup720 w/PFC3BXL)
- uRPF drop information is also available via `sh ip int`, shows information about spoofed and/or Remotely-Triggered Black Hole (RTBH)-dropped packets

Example—sh ip int

```
7500>sh ip int po10/0/0
POS10/0/0 is up, line protocol is up
 Internet address is 172.19.20.242/30
 Broadcast address is 255.255.255.255
 Address determined by non-volatile memory
 MTU is 4470 bytes
 Helper address is not set
 Directed broadcast forwarding is disabled
 Multicast reserved groups joined: 224.0.0.10
 Outgoing access list is 101
 Inbound access list is 102
 . .
```



Example—`sh ip int` (Cont.)

```
12000>sh ip int po1/1/0 | i veri
 IP verify source reachable-via ANY
  794407 verification drops
  1874428129 suppressed verification drops
12000>sh ip int po1/1/0 | i veri
 IP verify source reachable-via ANY
  794408 verification drops
  1874444463 suppressed verification drops
```

Cisco IOS CLI—`sh ip traffic`

- `sh ip traffic` provides a lot of useful global statistics, including per-protocol counts for ICMP, TCP, UDP, and multicast traffic
- Very useful for troubleshooting in general, as well as for spotting oddities
- Also shows global uRPF drop statistics

Example—sh ip traffic

```
12000>sh ip traff | i RPF
     0 no route, 124780722 unicast RPF, 0 forced drop
12000>sh ip traff | i RPF
     0 no route, 124816525 unicast RPF, 0 forced drop
12000>sh ip traff | i RPF
     0 no route, 127777619 unicast RPF, 0 forced drop
12000>sh ip traff | i RPF
     0 no route, 135875095 unicast RPF, 0 forced drop
12000>sh ip traff | i RPF
     0 no route, 150883277 unicast RPF, 0 forced drop
```

NetFlow

NetFlow Records and Key Fields

- NetFlow maintains per-'conversation' flow data in Flow Records in a cache on a NetFlow-enabled device, and optionally exports that flow data to a collection/analysis system
- It is a form of network telemetry which describes traffic conversations headed to/passing through a router

  Key Fields

  Key field values define a Flow Record

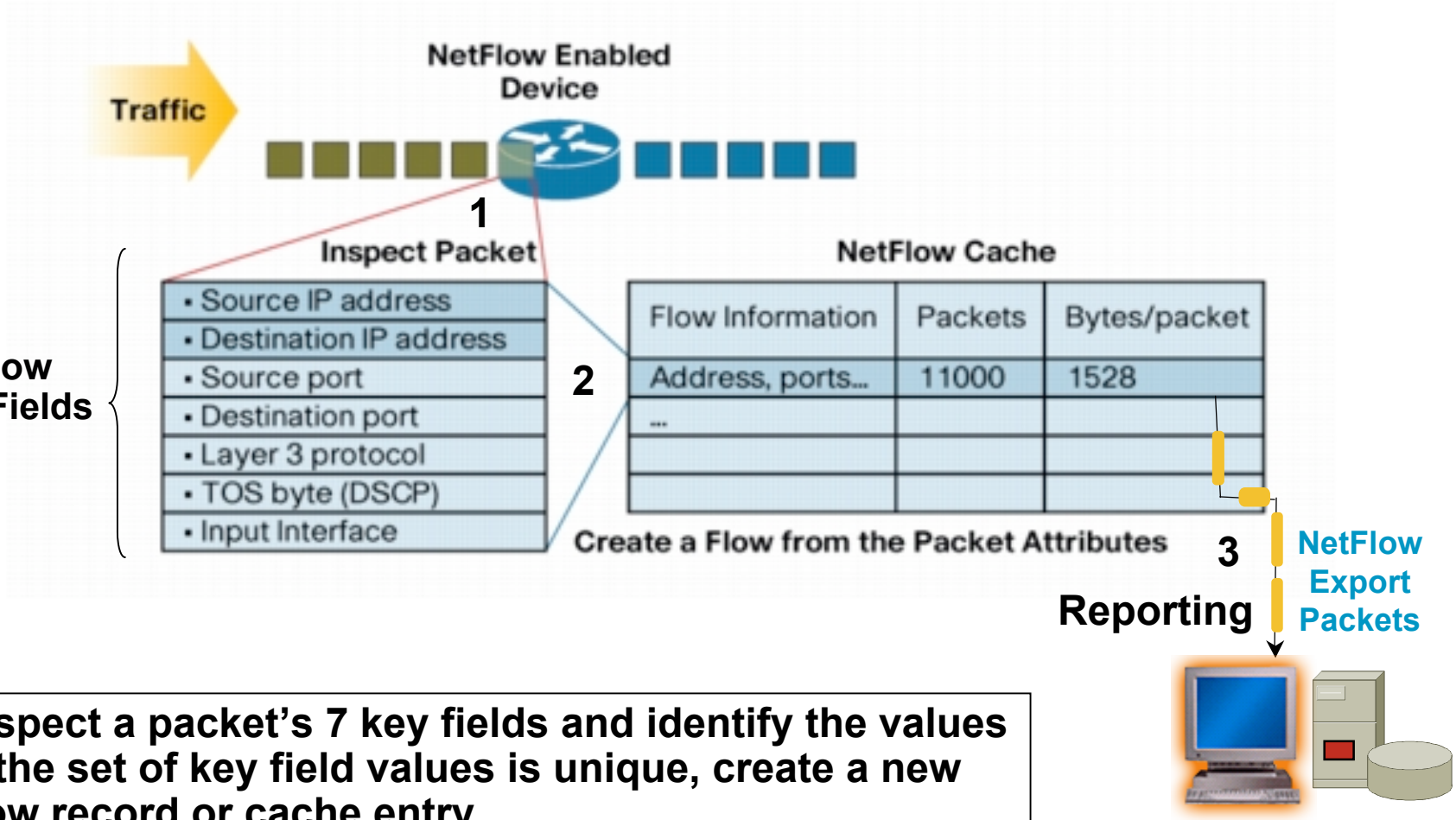
  An attribute in the packet used to create a Flow Record

  If the set of key field values is unique, a new flow is created

NetFlow Non-Key Fields and Statistics

- Non-key fields are not used to define a flow and are exported along with the flow and provide additional information
- NetFlow non-key fields
  - Source and destination AS's
  - Source and destination IP prefix masks
  - IP address of next-hop router
  - TCP flags
  - Output interface
- NetFlow features provide per flow statistics
  - Number of packets and bytes in flow
  - Time-stamps for first and last packets in flow

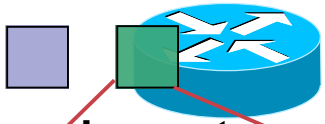
What Constitutes a Flow?



1. Inspect a packet's 7 key fields and identify the values
2. If the set of key field values is unique, create a new flow record or cache entry
3. When the flow terminates, export the flow to the collection/analysis system

NetFlow Key Fields Creating Flow Records

Example 1



Inspect Packet

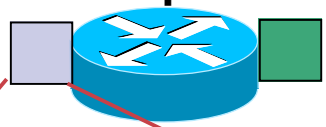
Key Fields	Packet 1
Source IP	1.1.1.1
Destination IP	2.2.2.2
Source port	23
Destination port	22078
Layer 3 Protocol	TCP - 6
TOS Byte	0
Input Interface	Ethernet 0

Create Flow Record in the Cache

Source IP	Dest. IP	Dest. I/F	Protocol	TOS	…	Pkts
1.1.1.1	2.2.2.2	E1	6	0	…	1100 0

1. **Inspect packet for key field values**
2. **Compare set of values to NetFlow cache**
3. **If the set of values are unique create a flow in cache**
4. **Inspect the next packet**

Example 2



Inspect Packet

Key Fields	Packet 2
Source IP	3.3.3.3
Destination IP	2.2.2.2
Source port	23
Destination port	22078
Layer 3 Protocol	TCP - 6
TOS Byte	0
Input Interface	Ethernet 0

Add new flow to the NetFlow Cache

Source IP	Dest. IP	Dest. I/F	Protocol	TOS	…	Pkts
3.3.3.3	2.2.2.2	E1	6	0	…	1100 0
1.1.1.1	2.2.2.2	E1	6	0	…	1100 0

NetFlow Cache on a Router

1. Create and update flows in NetFlow cache

Key fields in yellow
Non-key fields white

SrcIf	SrcIPadd	DstIf	DstIPadd	Protocol	TOS	Flgs	Pkts	Src Port	Src Msk	Src AS	Dst Port	Dst Msk	Dst AS	NextHop	Bytes/ Pkt	Active	Idle
Fa1/0	173.100.21.2	Fa0/0	10.0.227.12	11	80	10	11000	00A2	/24	5	00A2	/24	15	10.0.23.2	1528	1745	4
Fa1/0	173.100.3.2	Fa0/0	10.0.227.12	6	40	0	2491	15	/26	196	15	/24	15	10.0.23.2	740	41.5	1
Fa1/0	173.100.20.2	Fa0/0	10.0.227.12	11	80	10	10000	00A1	/24	180	00A1	/24	15	10.0.23.2	1428	1145.5	3
Fa1/0	173.100.6.2	Fa0/0	10.0.227.12	6	40	0	2210	19	/30	180	19	/24	15	10.0.23.2	1040	24.5	14

2. Expiration

- Inactive timer expired (15 sec is default)
- Active timer expired (30 min (1800 sec) is default)

SrcIf	SrcIPadd	DstIf	DstIPadd	Protocol	TOS	Flgs	Pkts	Src Port	Src Msk	Src AS	Dst Port	Dst Msk	Dst AS	NextHop	Bytes/ Pkt	Active	Idle
Fa1/0	173.100.21.2	Fa0/0	10.0.227.12	11	80	10	11000	00A2	/24	5	00A2	/24	15	10.0.23.2	1528	1800	4

4. Export version

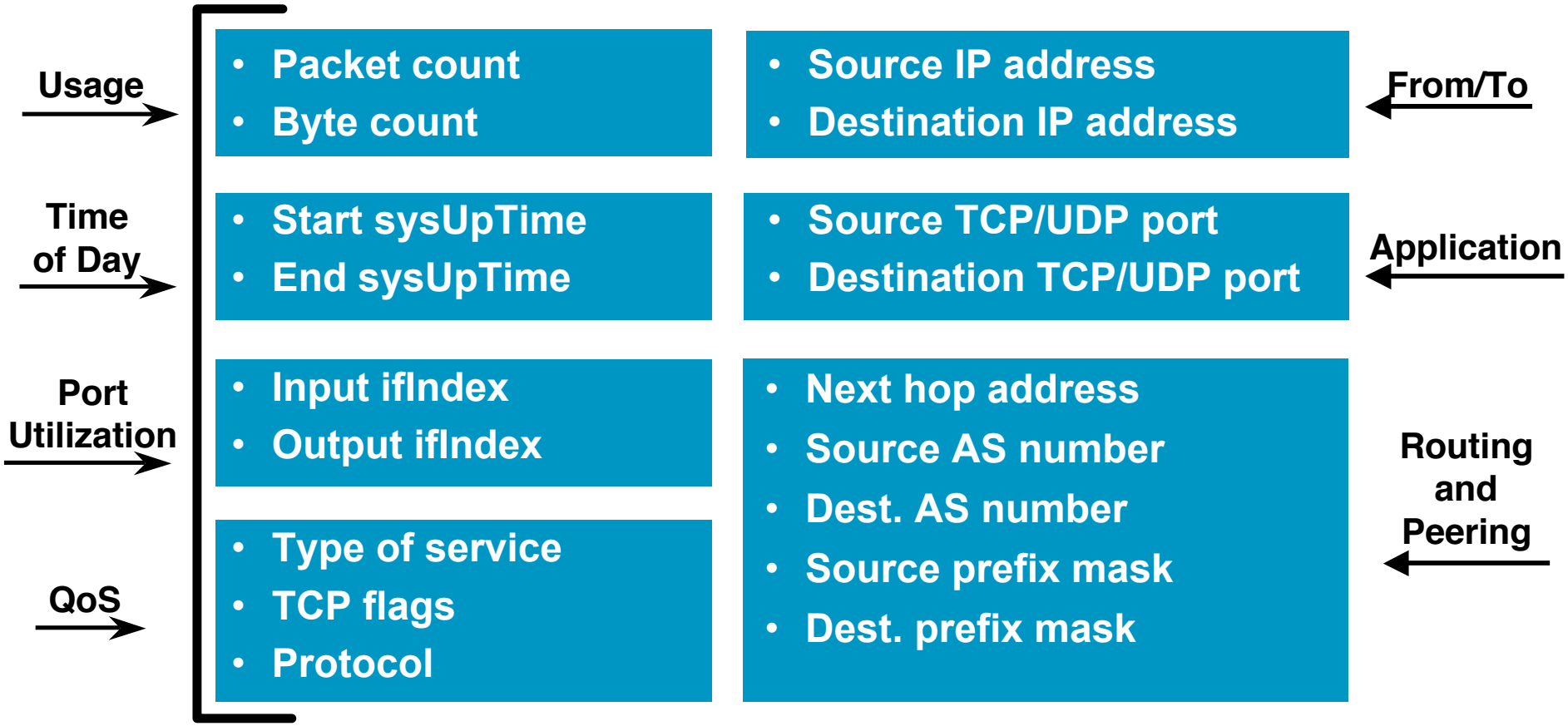
Non-Aggregated Flows—Export Version 5 or 9

5. Transport protocol

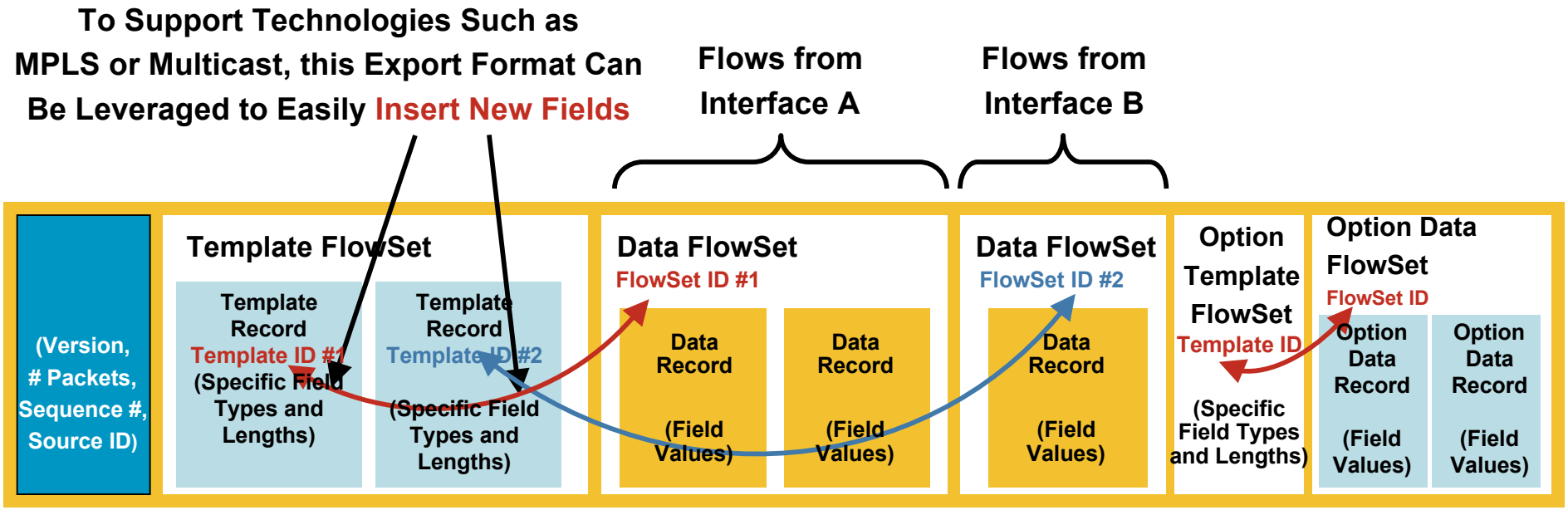
30 Flows per 1500 byte export packet

Export Packet

NetFlow Version 5—Flow Format



NetFlow v9 Export Packet Format



- Matching ID #s is the way to associate template to the data records
- The Header follows the same format as prior NetFlow versions so collectors will be backward compatible
- Each data record represents one flow
- If exported flows have the same fields then they can be contained in the same template Record E.G. Unicast traffic can be combined with multicast records
- If exported flows have different fields then they can't be contained in the same template record e.g. BGP next-hop can't be combined with MPLS Aware NetFlow records

Key Concept — NetFlow Scalability

- Packet capture is like a wiretap
- NetFlow is like a phone bill
- This level of granularity allows NetFlow to scale for very large amounts of traffic

  We can learn a lot from studying the phone bill

  Who's talking to whom, over what protocols and ports, for how long, at what speed, for what duration, etc.

  NetFlow is a form of telemetry pushed from the routers/switches — each one can be a sensor

NetFlow Deployment Considerations

- NetFlow should typically be enabled on all router interfaces where possible, it is useful for on-box troubleshooting via CLI as well as for export to analysis systems
- Ingress and egress NetFlow are now supported. Analysis systems typically must be configured to understand which is in use, for purposes of directionality
- 1:1 NetFlow is useful for troubleshooting, forensics, traffic analysis, and behavioral/relational anomaly-detection
- Sampled NetFlow is useful for traffic analysis and behavioral/relational anomaly-detection. Sampling is typically used in high-volume traffic situations where 1:1 NetFlow Data Export (NDE) is impractical
- Subinterface telemetry is supported using `ip flow ingress` and `ip flow egress` commands (supersede ip route cache flow)

NetFlow—v5 Fields for 6500/7600

Bytes	Content	Description	Flow masks: · X=Populated · A=Additional field (see the "Populating Additional NDE Fields" section)					
			Source	Destination	Destination Source	Destination Source Interface	Full	Full Interface
0-3	srcaddr	Source IP address	X	0	X	X	X	X
4-7	dstaddr	Destination IP address	0	X	X	X	X	X
8-11	nexthop	Next hop router's IP address	0	A[1]	A	A	A	A
12-13	input	Ingress interface SNMP ifIndex	0	0	0	X	0	X
14-15	output	Egress interface SNMP ifIndex	0	A 1	A	A	A	A
16-19	dPkts	Packets in the flow	X	X	X	X	X	X
20-23	dOctets	Octets (bytes) in the flow	X	X	X	X	X	X
24-27	first	SysUptime at start of the flow (milliseconds)	X	X	X	X	X	X
28-31	last	SysUptime at the time the last packet of the flow was received (milliseconds)	X	X	X	X	X	X
32-33	srcport	Layer 4 source port number or equivalent	0	0	0	0	X[2]	X 2
34-35	dstport	Layer 4 destination port number or equivalent	0	0	0	0	X	X
36	pad1	Unused (zero) byte	0	0	0	0	0	0
37	tcp_flags	Cumulative OR of TCP flags[3]	0	0	0	0	0	0
38	prot	Layer 4 protocol (for example, 6=TCP, 17=UDP)	0	0	0	0	X	X
39	tos	IP type-of-service byte	X[4]	X 4	X 4	X 4	X 4	X 4
40-41	src_as	Autonomous system number of the source, either origin or peer	X	0	X	X	X	X
42-43	dst_as	Autonomous system number of the destination, either origin or peer	0	X	X	X	X	X
44-45	src_mask	Source address prefix mask bits	X	0	X	X	X	X
46-47	dst_mask	Destination address prefix mask bits	0	X	X	X	X	X
48	pad2	Pad 2	0	0	0	0	0	0

[1] With the destination flow mask, the "Next hop router's IP address" field and the "Output interface's SNMP ifIndex" field might not contain information that is accurate for all flows.
[2] In PFC3BXL or PFC3B mode, for ICMP traffic, contains the ICMP code and type values.
[3] Always zero for hardware-switched flows.
[4] Populated in PFC3BXL or PFC3B mode.

Source: Cisco Systems, Inc.

NetFlow—Enabling on the Router

```
Router(config)#int Fa6/3.0
Router(config-subif)#ip flow ingress
Router(config-subif)#ex
Router(config)# ip flow-export version 9
Router(config)# ip flow-export source loopback0
Router(config)# ip flow-export destination 10.42.42.1 9991
Router(config)# ip flow-export template refresh-rate 15
Router(config)# ip flow-export template timeout-rate 90
Router(config)# ip flow-export template options export-stats
Router(config)# ip flow-export template options refresh-rate 25
Router(config)# ip flow-export template options timeout-rate 120
Router(config)# ip flow-cache timeout active 1
```

NetFlow Data Export (NDE)—Verifying on the Router

```
Router# show ip flow export

Flow export is enabled
Exporting flows to 10.42.42.1 (9991) 10.0.101.254 (9991)
 Exporting using source IP address 10.0.101.203
  Version 9 flow records

Export Stats for 10.42.42.1 (9991)
      3807 flows exported in 190 udp datagrams
      0 flows failed due to lack of export packet
      190 export packets were sent up to process level
      0 export packets were dropped due to no fib
      0 export packets were dropped due to adjacency issues
      0 export packets were dropped enqueuing for the RP
      0 export packets were dropped due to IPC rate limiting
```

Cisco 7200 NetFlow CLI Example

```
7200>sh ip cache flow
IP packet size distribution (14952M total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
   .001 .325 .096 .198 .029 .014 .010 .010 .012 .003 .003 .005 .003 .003 .002

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .004 .005 .009 .043 .217 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 4456704 bytes
  65527 active, 9 inactive, 2364260060 added
  4143670560 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow     /Flow
TCP-Telnet      1398292      0.3        14   156      4.6       6.0      17.2
TCP-FTP        99569986     23.1         1    41     24.2       0.0       4.8
TCP-FTPD         185530      0.0         1    66      0.0       1.5      17.4
TCP-WWW       440235639    102.5         8   483    919.5       2.9      10.1
TCP-SMTP       18951357      4.4        21   629     94.1       6.4      20.0
TCP-X             11340      0.0         1    48      0.0       0.2      40.8
TCP-BGP            4018      0.0         2    51      0.0       7.5      12.5
TCP-NNTP        2701390      0.6       104   846     65.5      10.6      16.9
TCP-Frag          38932      0.0        11   407      0.1       1.9      17.2
TCP-other     403434143     93.9         7   444    688.2       6.9      18.6
```

Cisco 7200 NetFlow CLI Example (Cont.)

TCP-other	403434143	93.9	7	444	688.2	6.9	18.6
UDP-DNS	65590214	15.2	1	114	24.0	1.6	17.7
UDP-NTP	2415600	0.5	1	76	0.6	0.5	18.6
UDP-TFTP	70011	0.0	5	77	0.0	32.2	17.8
UDP-Frag	1017582	0.2	85	88	20.1	14.4	17.9
UDP-other	462375834	107.6	11	392	1189.0	5.3	23.5
ICMP	856323251	199.3	1	89	217.4	0.3	37.7
IGMP	98	0.0	4275	444	0.0	487.8	15.4
IPINIP	46	0.0	11229	412	0.1	1039.7	6.8
GRE	104643	0.0	10	86	0.2	47.9	15.8
IP-other	9766627	2.2	102	318	232.5	85.6	19.7
Total:	2364194533	550.4	6	411	3481.2	3.3	24.3

Cisco 7200 NetFlow CLI Example (Cont.)

```
SrcIf         SrcIPaddress    DstIf         DstIPaddress    Pr SrcP DstP  Pkts
Fa0/1         10.66.74.46     Fa0/0         219.103.129.162 01 0000 0800     1
Fa0/1         10.66.115.182   Fa0/0         194.22.114.198  01 0000 0800     1
Fa2/1         10.66.74.46     Fa0/0         61.79.227.123   01 0000 0800     1
Fa0/1         10.66.74.46     Fa0/0         211.167.105.242 01 0000 0800     1
Fa0/0         129.42.184.35   Null          64.104.193.198  06 2891 0019     3
Fa2/1         10.66.115.182   Fa0/0         202.20.138.184  01 0000 0800     1
Fa2/1         10.66.115.182   Fa0/0         63.76.237.255   01 0000 0800    10K
Fa2/1         10.66.74.46     Fa0/0         61.205.214.45   01 0000 0800     1
Fa2/1         10.66.115.182   Fa0/0         220.114.157.1   01 0000 0800     1
Fa0/0         64.104.252.196  Fa2/1         64.104.200.210  11 0000 0000     1
Fa0/1         64.104.192.130  Fa0/0         217.136.19.103  11 2710 2710  3603
```

Cisco Catalyst 6500/7600 Series Routers—Cisco IOS NetFlow CLI Example

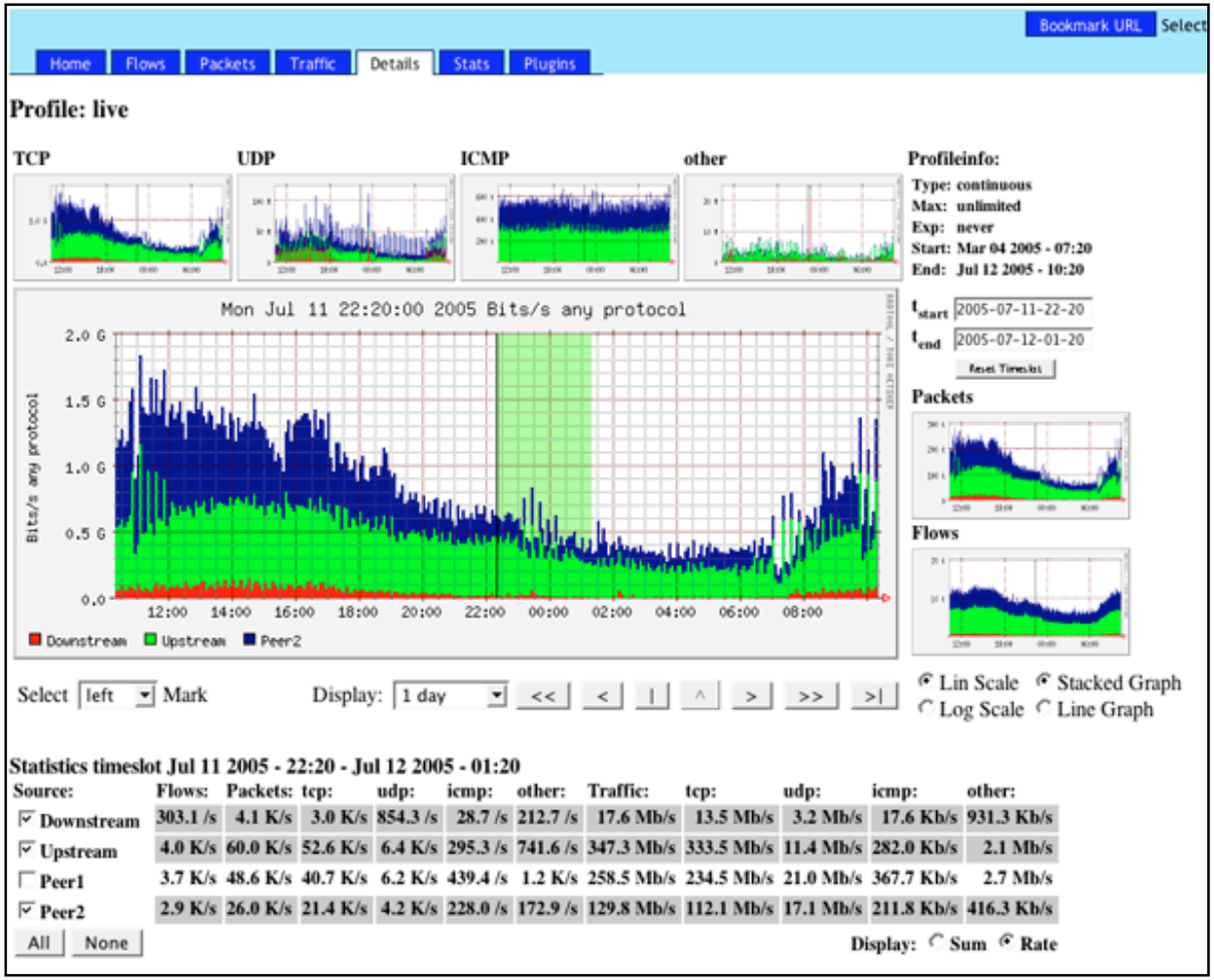
```
6500>sh mls NetFlow ip detail
Displaying NetFlow entries in Supervisor Earl
DstIP           SrcIP           Prot:SrcPort:DstPort  Src i/f:AdjPtr
-----------------------------------------------------------------
Pkts          Bytes       Age   LastSeen  Attributes
-------------------------------------------------

    QoS     Police Count Threshold    Leak     Drop Bucket  Use-Tbl Use-Enable
----------+------------+---------+----------+----+-------+-------+----------+
172.87.19.217   171.70.15[illegible].90   tcp :10112  :www      102[illegible]: 0
3            144         10        00:07:11   L3 - Dynamic
  0x0          0               0        0       NO   48        NO       NO
171.101.24.123  171.69.89.39    tcp :1303   :139      400 : 0
0            0           39    00:06:42   L3 - Dynamic
  0x0          0               0        0       NO   48        NO       NO
202.56.200.22   172.19.61.10  icmp:0      :0        1028: 0
26           2028        383   00:07:05   L3 - Dynamic
  0x0          0               0        0       NO   78        NO       NO
```

Cisco Catalyst 6500/7600 Series Switches: CatOS (Hybrid) NetFlow CLI Example

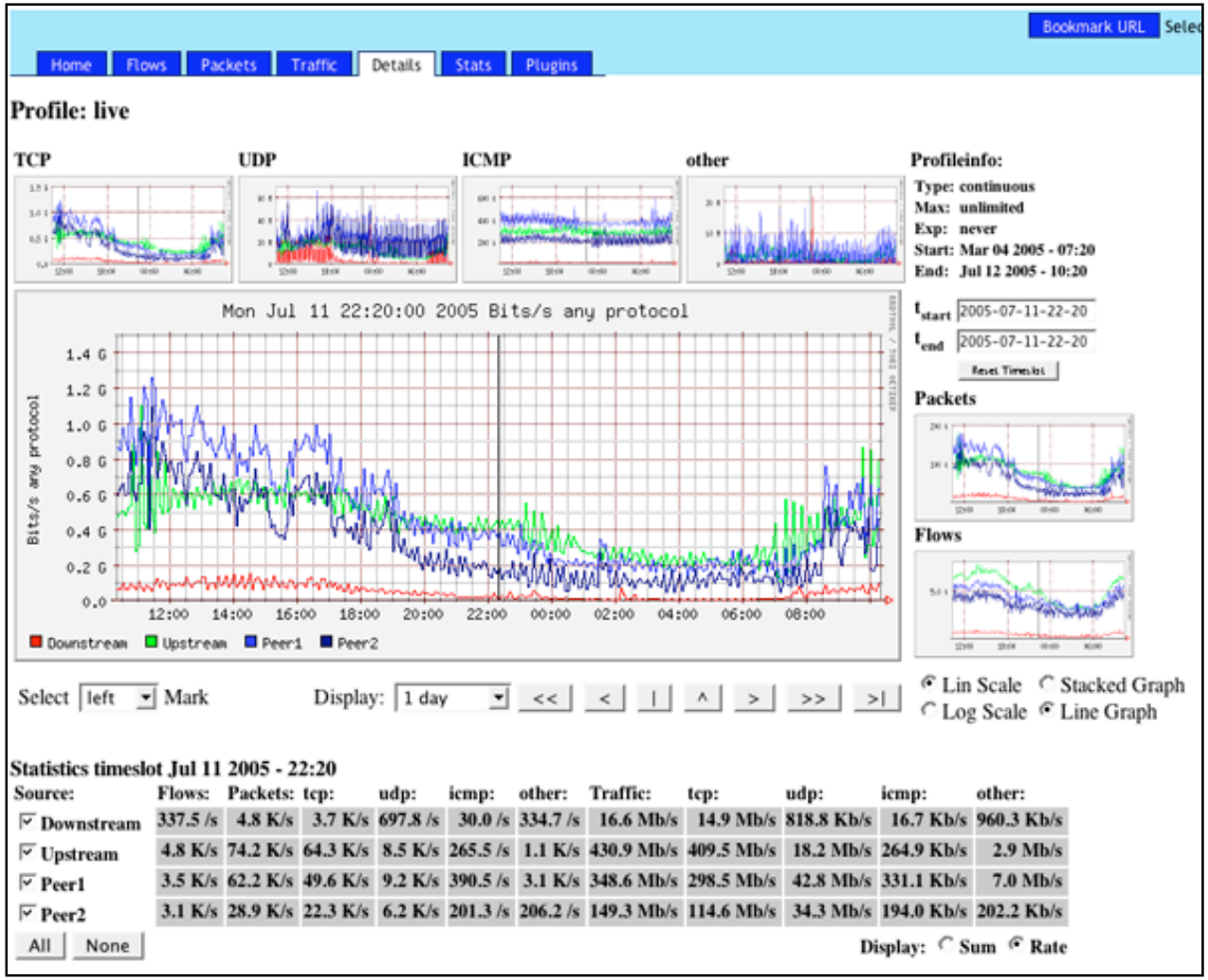
```
6500CatOS> (enable) sh mls stat entry ip dest 172.19.61.10
                                  Last    Used
Destination IP   Source IP       Prot  DstPrt SrcPrt Stat-Pkts  Stat-Bytes
---------------- --------------- ----- ------ ------ ---------- ---------------
 172.19.61.10  66.133.186.82   TCP   WWW    4881   5          659
 172.19.61.10  10.86.110.78    TCP   WWW    39398  5          493
 172.19.61.10  24.130.143.154  TCP   443    4602   7          852
 172.19.61.10  172.30.91.146   TCP   WWW    2439   5          669
 172.19.61.10  144.198.52.166  TCP   WWW    1852   5          592
 172.19.61.10  65.167.120.4    TCP   WWW    27421  5          697
 172.19.61.10  194.152.95.238  TCP   WWW    44695  5          381
 172.19.61.10  167.206.112.86  TCP   WWW    57965  48         2090
 172.19.61.10  66.122.131.11   ICMP  0      0      10         840
 172.19.61.10  69.15.18.170    TCP   WWW    33206  5          678
 172.19.61.10  156.153.255.126 TCP   WWW    55628  5          476
 172.19.61.10  62.223.250.195  ICMP  0      0      29         2262
 172.19.61.10  193.251.55.241  TCP   WWW    3718   5          606
 172.19.61.10  65.167.120.4    TCP   WWW    25497  6          715
 172.19.61.10  24.214.103.12   TCP   WWW    3444   5          681
 172.19.61.10  161.109.128.102 TCP   WWW    1246   5          682
```

NetFlow—nfdump and nfsen



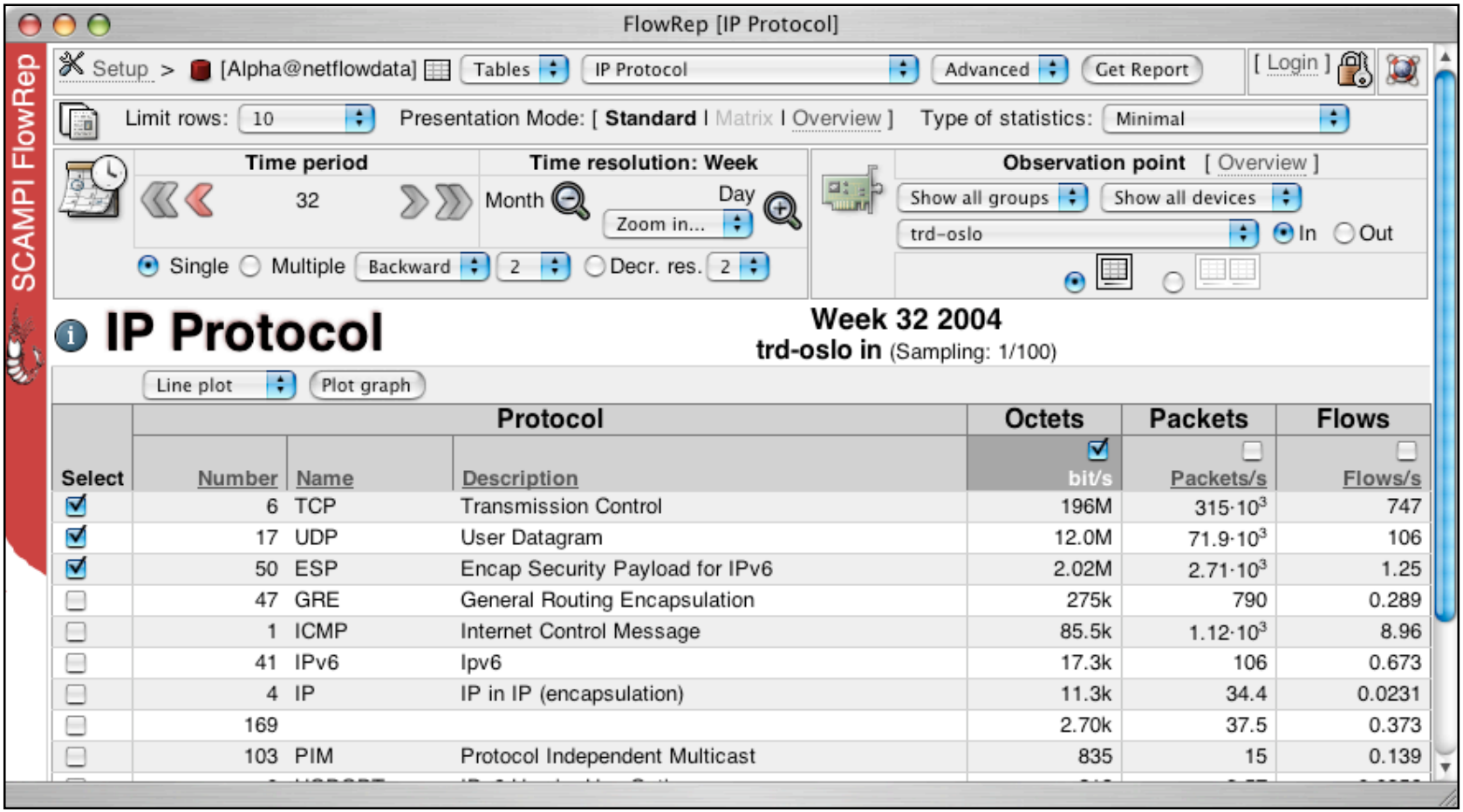
Source: http://nfsen.sourceforge.net

NetFlow—nfdump and nfsen (Cont.)



Source: http://nfsen.sourceforge.net

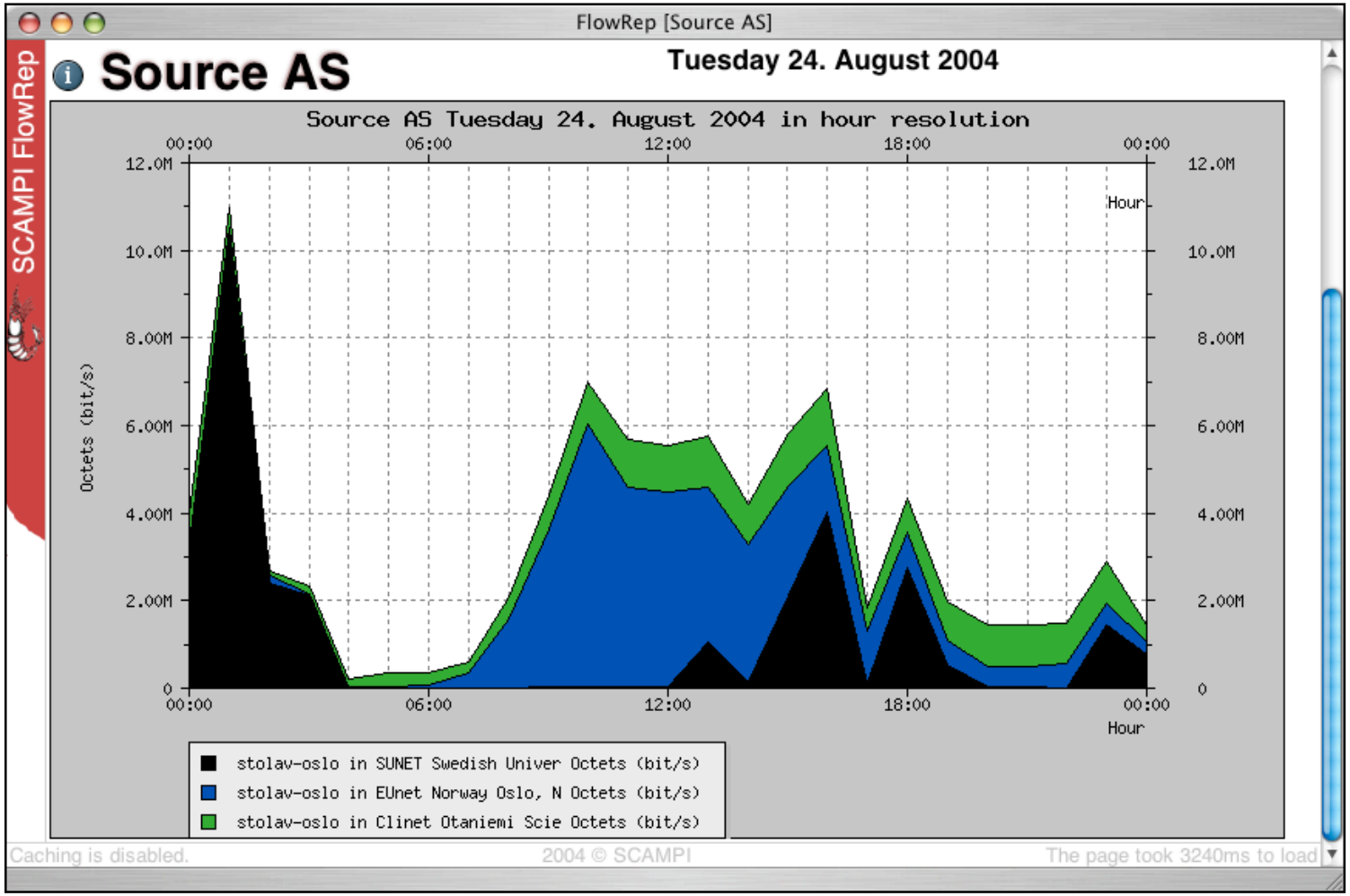
NetFlow—Stager



Select	Protocol Number	Name	Description	Octets bit/s	Packets Packets/s	Flows Flows/s
☑	6	TCP	Transmission Control	196M	$315 \cdot 10^3$	747
☑	17	UDP	User Datagram	12.0M	$71.9 \cdot 10^3$	106
☑	50	ESP	Encap Security Payload for IPv6	2.02M	$2.71 \cdot 10^3$	1.25
☐	47	GRE	General Routing Encapsulation	275k	790	0.289
☐	1	ICMP	Internet Control Message	85.5k	$1.12 \cdot 10^3$	8.96
☐	41	IPv6	Ipv6	17.3k	106	0.673
☐	4	IP	IP in IP (encapsulation)	11.3k	34.4	0.0231
☐	169			2.70k	37.5	0.373
☐	103	PIM	Protocol Independent Multicast	835	15	0.139

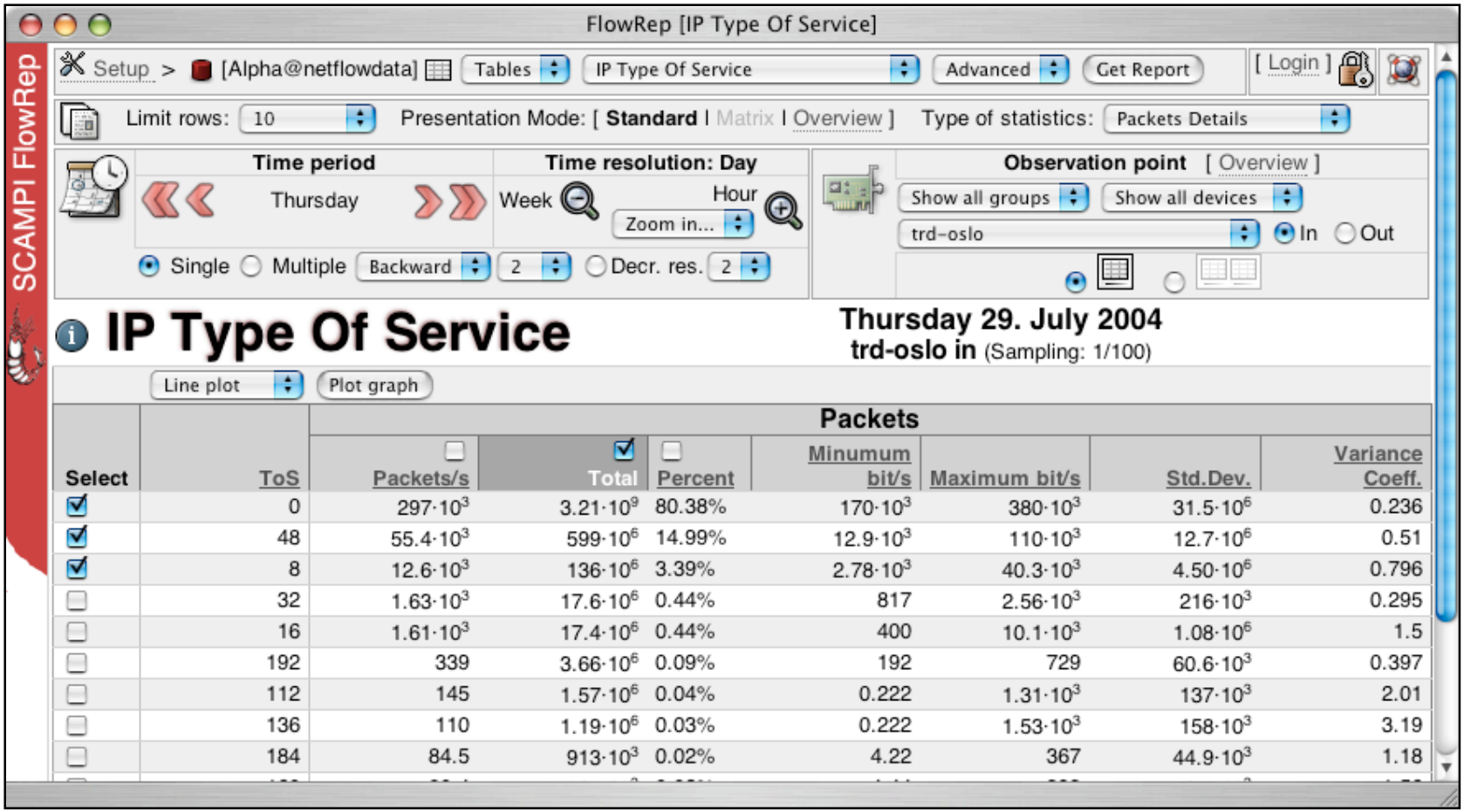
Source: UNINETT

NetFlow—Stager (Cont.)



Source: UNINETT

NetFlow—Stager (Cont.)



Select	ToS	Packets						
		Packets/s	Total	Percent	Minumum bit/s	Maximum bit/s	Std.Dev.	Variance Coeff.
☑	0	$297 \cdot 10^3$	$3.21 \cdot 10^9$	80.38%	$170 \cdot 10^3$	$380 \cdot 10^3$	$31.5 \cdot 10^6$	0.236
☑	48	$55.4 \cdot 10^3$	$599 \cdot 10^6$	14.99%	$12.9 \cdot 10^3$	$110 \cdot 10^3$	$12.7 \cdot 10^6$	0.51
☑	8	$12.6 \cdot 10^3$	$136 \cdot 10^6$	3.39%	$2.78 \cdot 10^3$	$40.3 \cdot 10^3$	$4.50 \cdot 10^6$	0.796
☐	32	$1.63 \cdot 10^3$	$17.6 \cdot 10^6$	0.44%	817	$2.56 \cdot 10^3$	$216 \cdot 10^3$	0.295
☐	16	$1.61 \cdot 10^3$	$17.4 \cdot 10^6$	0.44%	400	$10.1 \cdot 10^3$	$1.08 \cdot 10^6$	1.5
☐	192	339	$3.66 \cdot 10^6$	0.09%	192	729	$60.6 \cdot 10^3$	0.397
☐	112	145	$1.57 \cdot 10^6$	0.04%	0.222	$1.31 \cdot 10^3$	$137 \cdot 10^3$	2.01
☐	136	110	$1.19 \cdot 10^6$	0.03%	0.222	$1.53 \cdot 10^3$	$158 \cdot 10^3$	3.19
☐	184	84.5	$913 \cdot 10^3$	0.02%	4.22	367	$44.9 \cdot 10^3$	1.18

Source: UNINETT

Key Concepts—Anomaly Detection

- No signatures—instead, uses observed behavior as the baseline
- No false positives—everything reported is actually happening (setting thresholds for tuning severity)
- Can detect ‘minute-0’ attacks
- Can highlight behaviors which are not indicative of attack traffic, but are still of interest
- Can be used as an indicator to focus more closely using packet-capture, IDS, etc.
- Not limited to large-scale events—user-defined thresholds determine severity/alert scaling

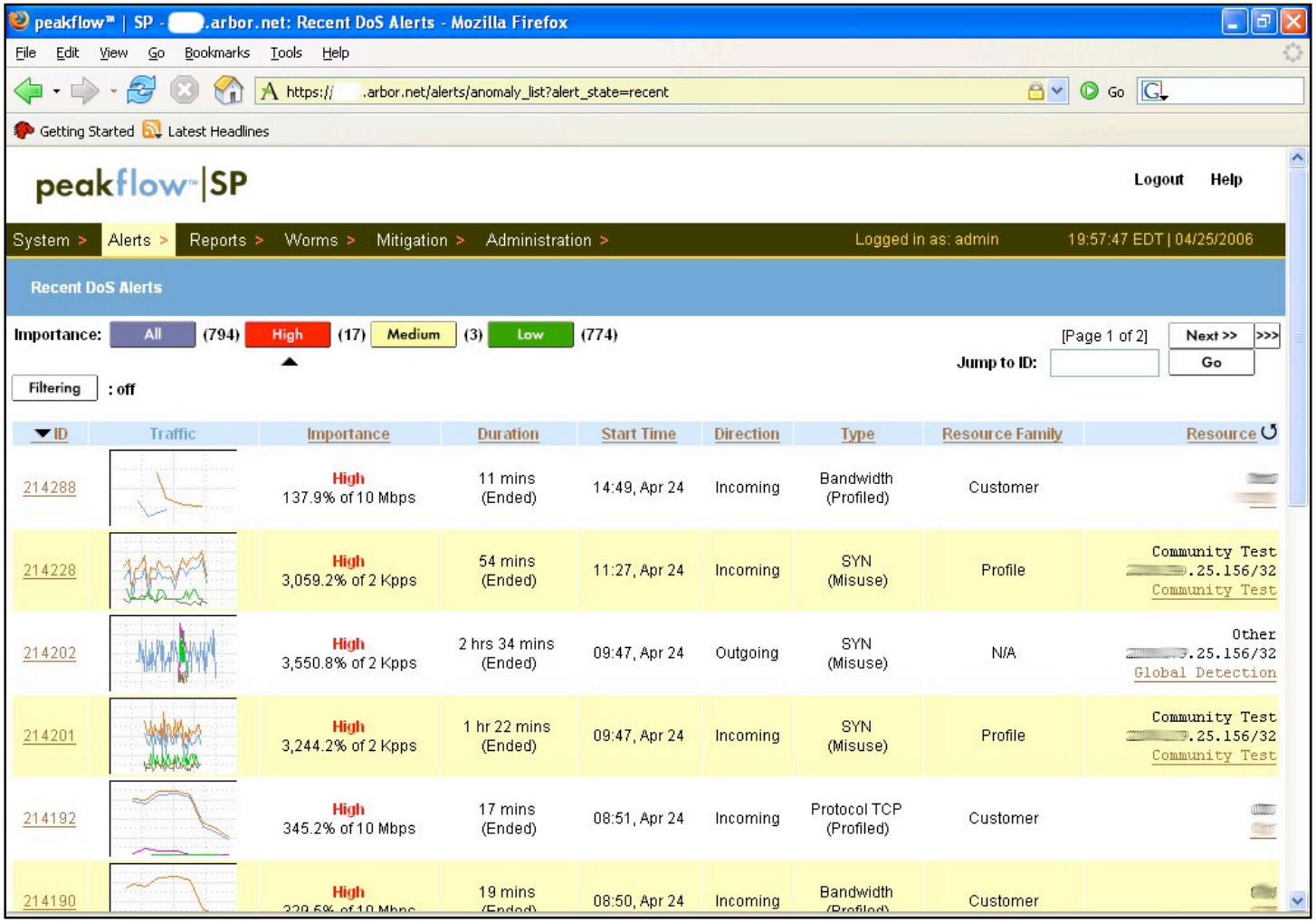
NetFlow and Anomaly-Detection

- Prior to implementing an anomaly-detection system, traffic analysis should be performed in order to gain an understanding of general traffic rates and patterns
- Interfaces are generally classified on the anomaly-detection system (backbone, peering, distribution, etc.)
- IP ranges for netblocks inside the network are input into the system for purposes of determining directionality and sources/destinations within the network
- Learning is generally performed over a significant interval, including both peaks and valleys of network activity

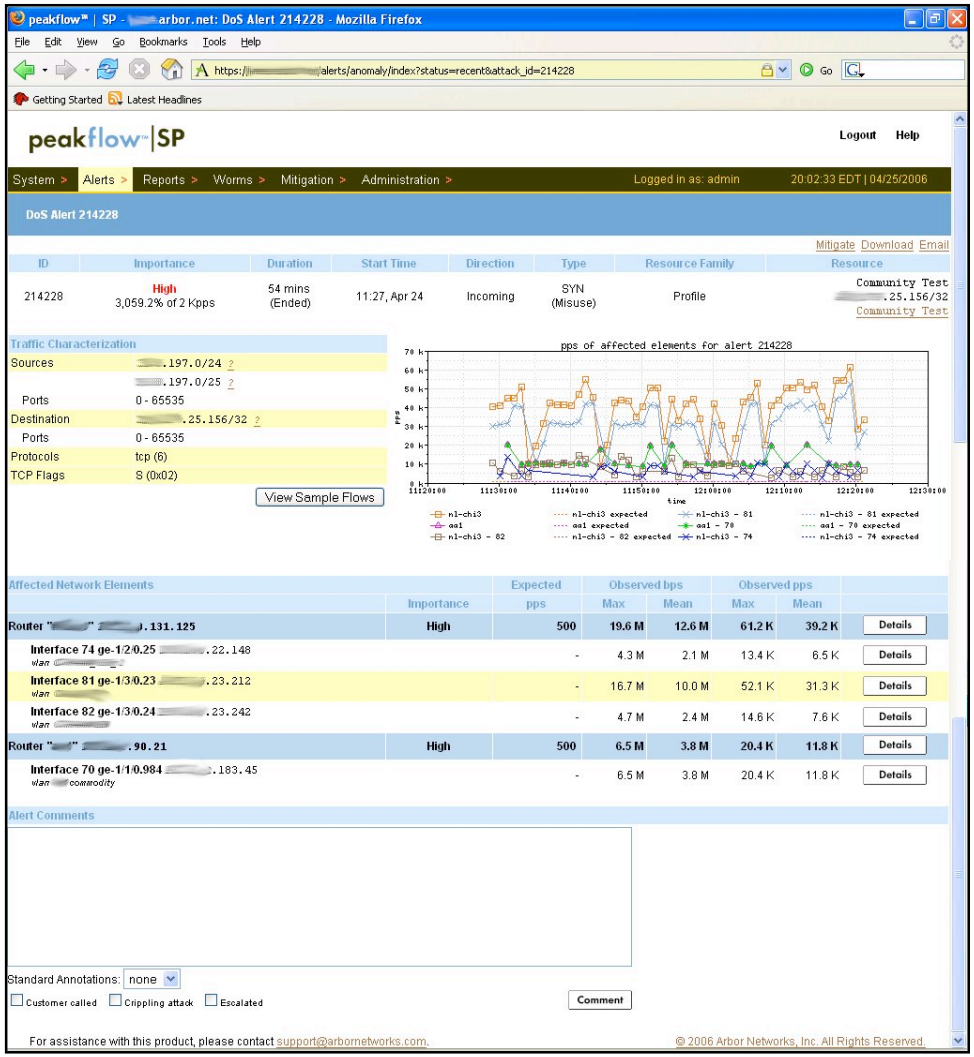
NetFlow and Anomaly-Detection (Cont.)

- Thresholds, alarm suppression intervals, event threshold suppression intervals, behavioral rules and other parameters are adjusted prior to 'going live'
- If traffic rates/patterns change dramatically, thresholds may require adjustment over time
- Arbor Peakflow SP DoS (SPs, public-facing enterprise networks) and Cisco CS-MARS (enterprise) perform statistical anomaly-detection; Arbor Peakflow/X performs relational/behavioral anomaly-detection (large enterprise)

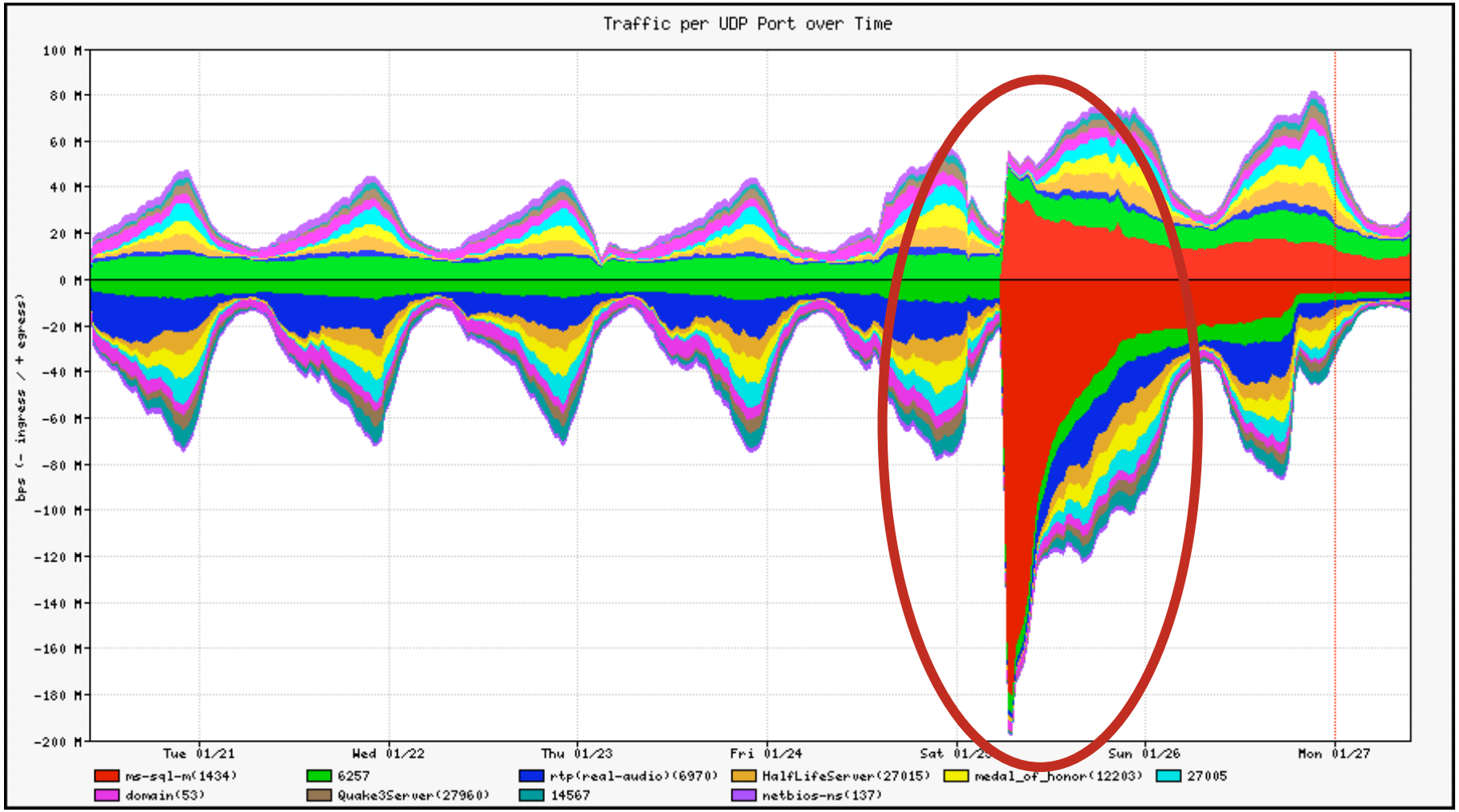
Example—Arbor Peakflow SP DoS Module



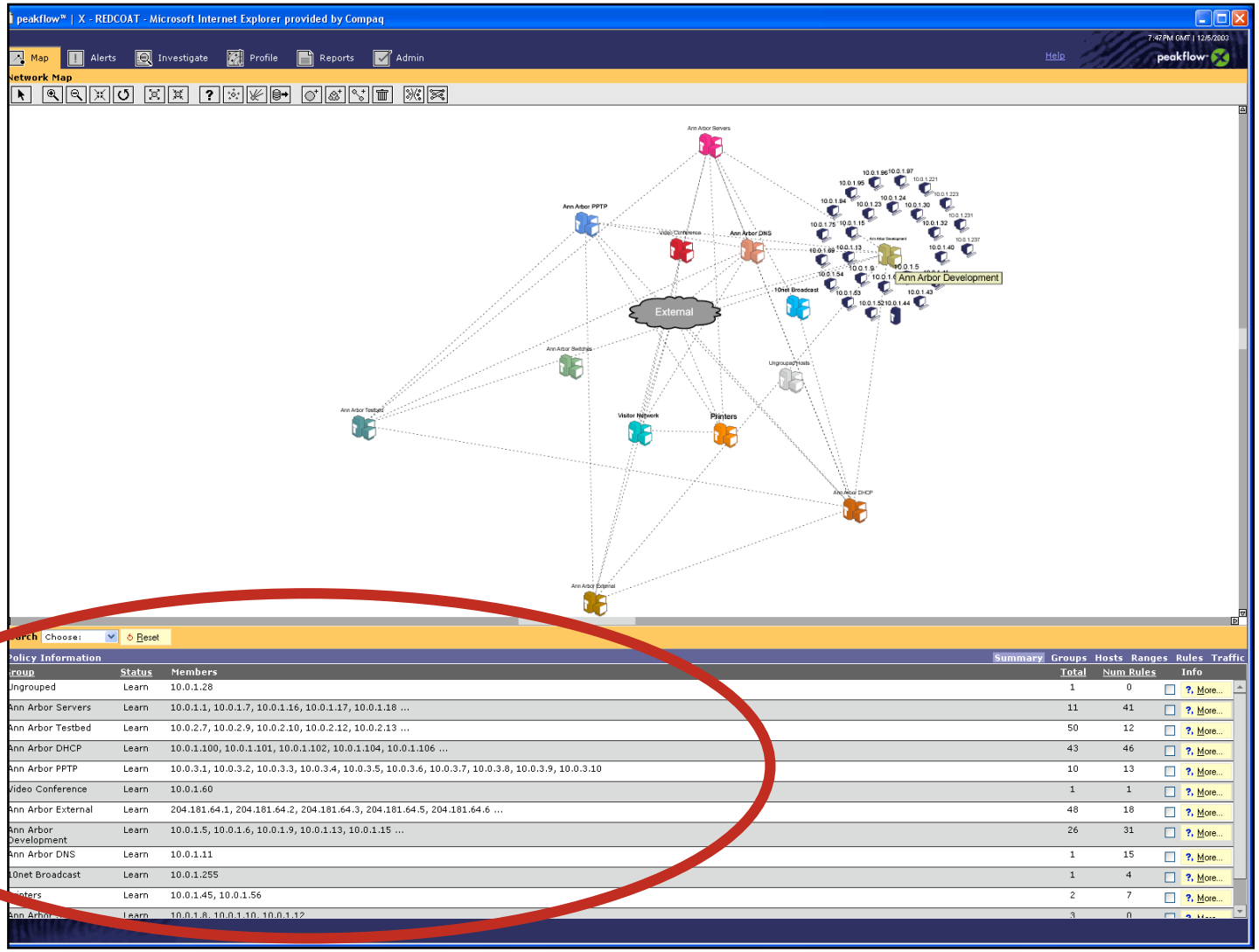
Example—Arbor Peakflow SP DoS Module



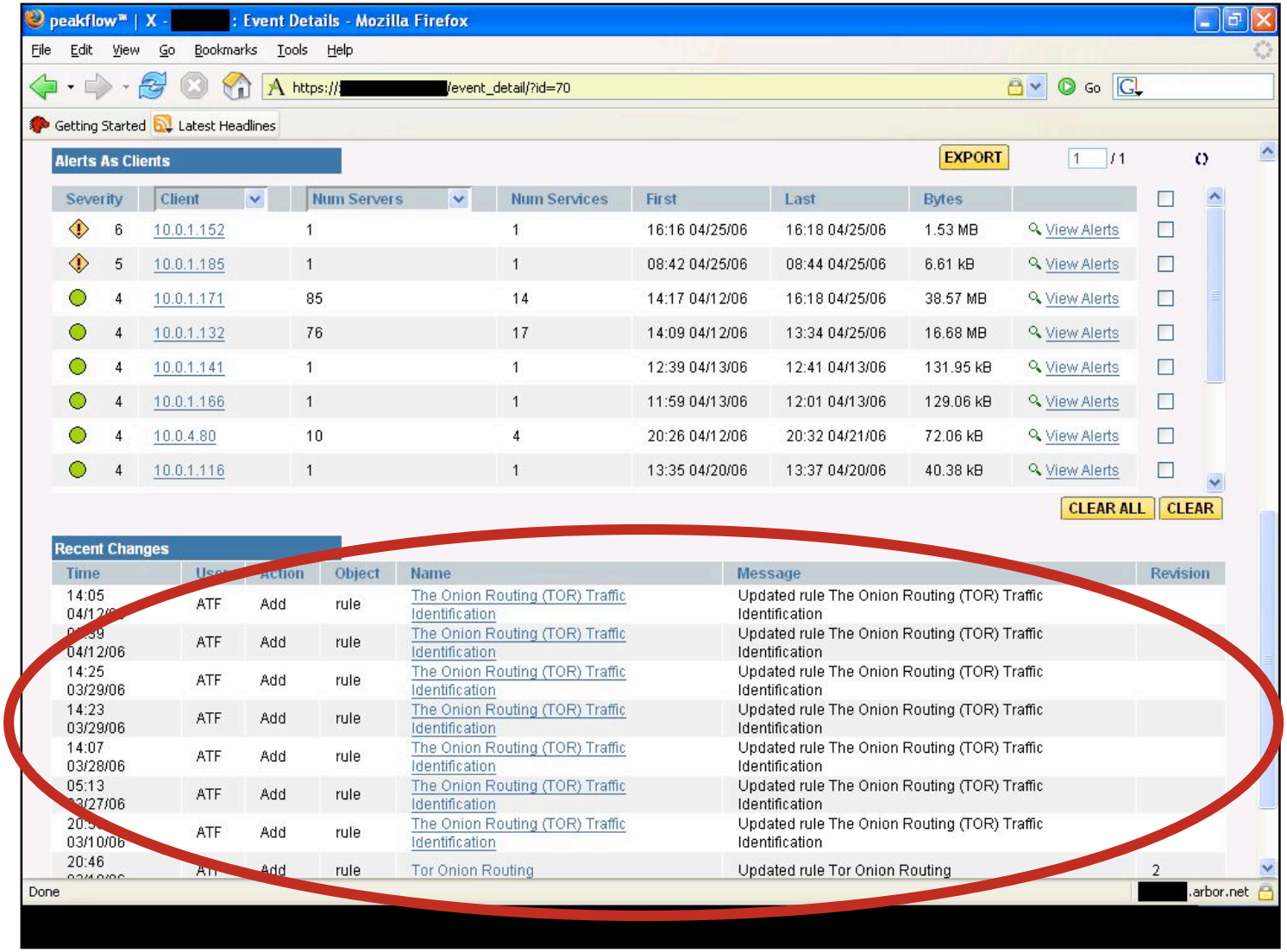
Example—SQL Slammer



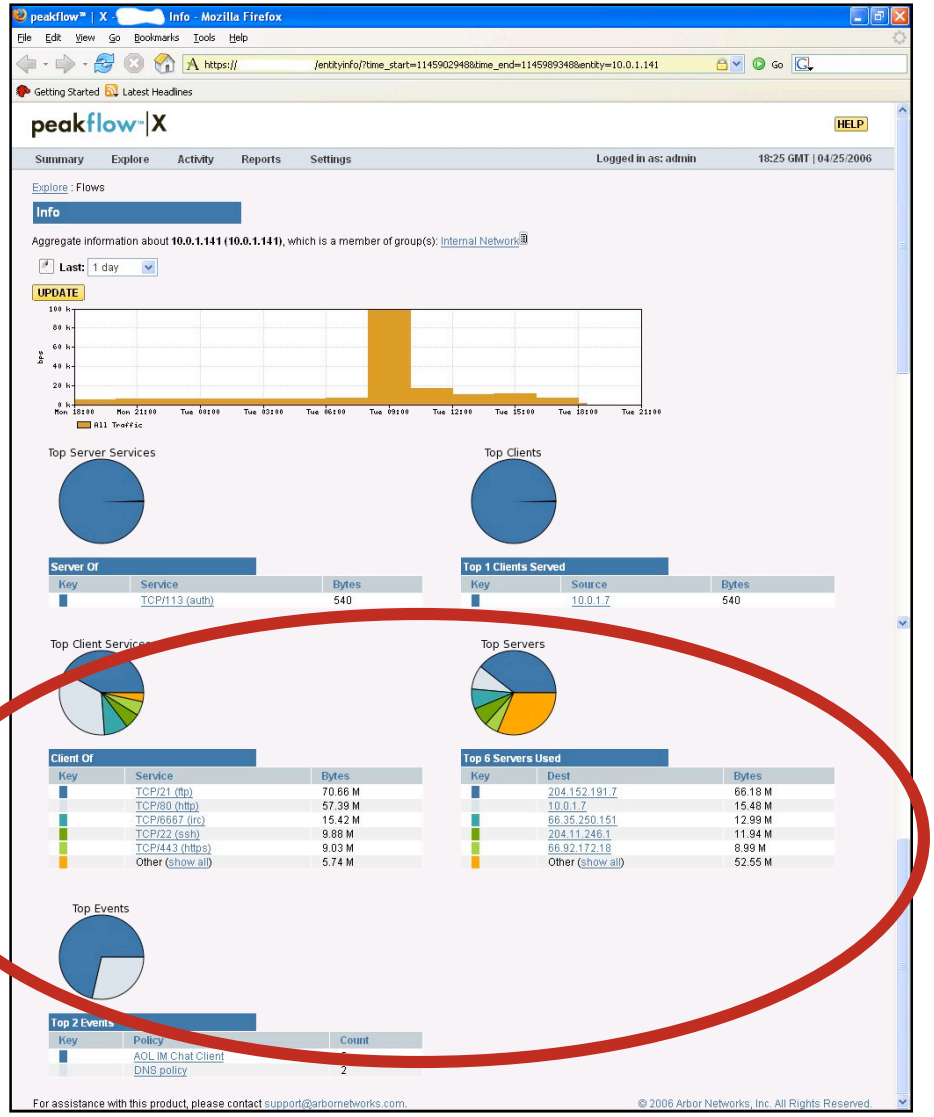
Example—Arbor Peakflow/X



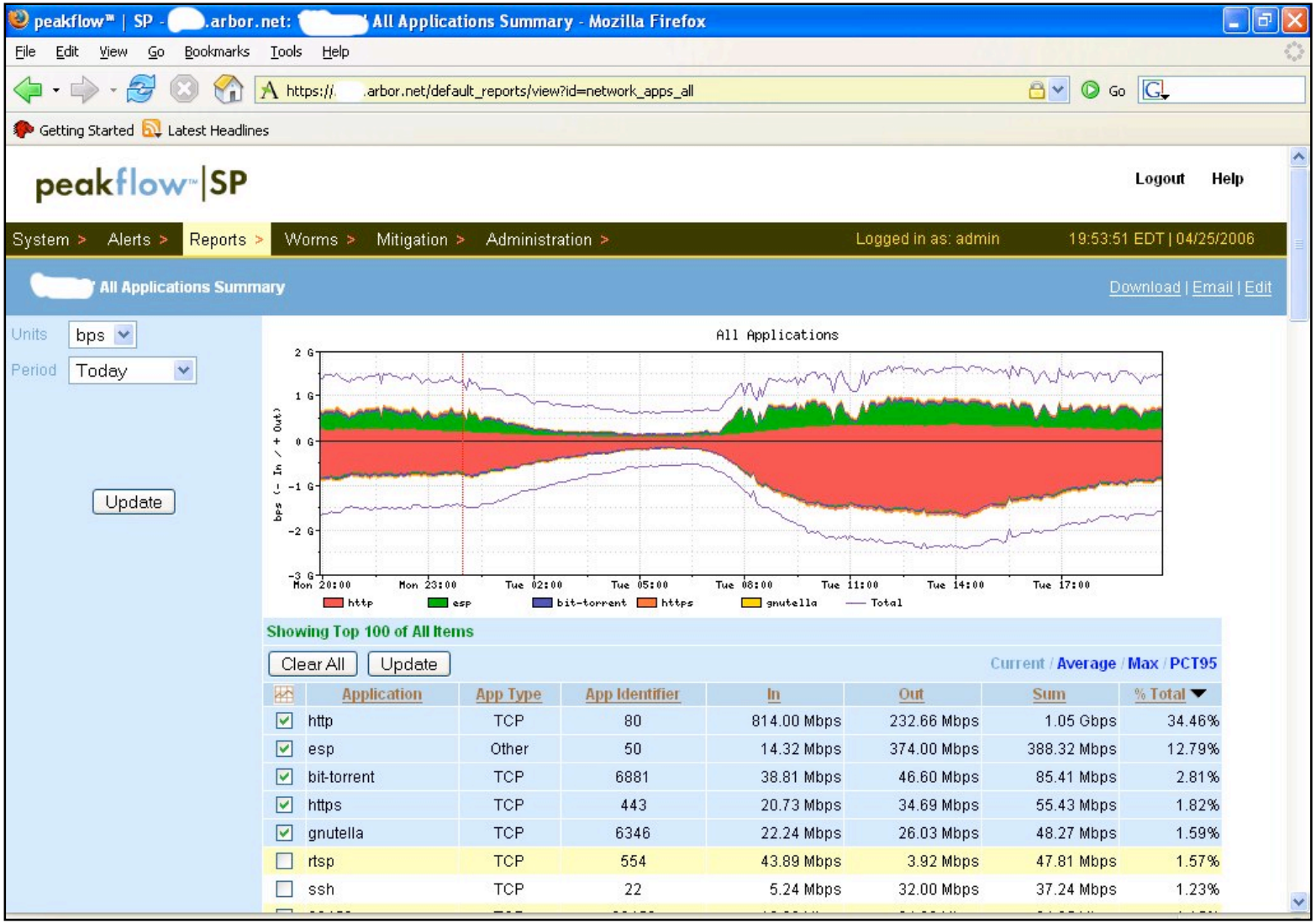
Example—Arbor Peakflow/X (Cont.)



Example—Arbor Peakflow/X (Cont.)



NetFlow—Application Distribution



Application	App Type	App Identifier	In	Out	Sum	% Total
http	TCP	80	814.00 Mbps	232.66 Mbps	1.05 Gbps	34.46%
esp	Other	50	14.32 Mbps	374.00 Mbps	388.32 Mbps	12.79%
bit-torrent	TCP	6881	38.81 Mbps	46.60 Mbps	85.41 Mbps	2.81%
https	TCP	443	20.73 Mbps	34.69 Mbps	55.43 Mbps	1.82%
gnutella	TCP	6346	22.24 Mbps	26.03 Mbps	48.27 Mbps	1.59%
rtsp	TCP	554	43.89 Mbps	3.92 Mbps	47.81 Mbps	1.57%
ssh	TCP	22	5.24 Mbps	32.00 Mbps	37.24 Mbps	1.23%

NetFlow—More Information

- Cisco NetFlow home

  http://www.cisco.com/warp/public/732/Tech/np/NetFlow/

- Linux NetFlow reports HOWTO

  http://www.linuxgeek.org/NetFlow-howto.php

- Arbor Networks Peakflow SP and Peakflow/X

  http://www.arbornetworks.com

- nfdump and nfsen

  http://nfdump.sourceforge.net

  http://nfsen.sourceforge.net

- Stager

  http://software.uninett.no/stager/

DNS

Utilizing DNS Telemetry for Detection

- The Domain Name System (DNS) is a ‘background’ service we often don’t think about, but in actuality use many, many times each day
- Many types of application use name-based lookups—Web browsers, email servers, Web servers—and malware such as trojans and bots running on compromised hosts
- By examining DNS logs and statistics, we can detect activity which should be further investigated
- Correlating DNS-related info with other forms of telemetry (NetFlow, packet capture, application logs, etc.), we can often infer the causes and effects of unusual network activity

Example—dnstop query types

```
0 new queries, 38 total queries          Wed Jun 1 17:35:51 2005

Query Type      count       %
---------- --------- ------
A?                  9   23.7
NS?                 1    2.6
SOA?                1    2.6
PTR?               15   39.5
MX?                10   26.3
TXT?                2    5.3
```

Example—dnstop sources output

```
0 new queries, 38 total queries Wed Jun 1 17:35:51 2005

Sources          count    %
---------------- -------- -----
172.19.61.44        19    50.0
172.19.60.20         9    25.0
172.19.61.33         9    25.0
```

Example—dnstop source, record output

```
0 new queries, 38 total queries                    Wed Jun  1 17:35:51 2005

Source           3LD                                   count      %
---------------- --------------------                --------- ------
172.19.61.44     cnn.com                                  10   26.3
172.19.61.33     107.128.in-addr.arpa                      5   13.2
172.19.61.44     19.172.in-addr.arpa                       5   13.2
172.19.60.28     cisco.com                                 5   13.2
172.19.61.33     71.171.in-addr.arpa                       2    5.3
172.19.61.44     24.172.in-addr.arpa                       2    5.3
172.19.61.44     www.slacker.com                           2    5.3
172.19.61.44     google.com                                2    5.3
172.19.60.28     speakeasy.net                             1    2.6
172.19.61.44     168.192.in-addr.arpa                      1    2.6
172.19.61.33     .                                         1    2.6
172.19.60.28     www.cnn.com                               1    2.6
172.19.61.44     telecomplete.co.uk                        1    2.6
```

Example—dnstop destination output

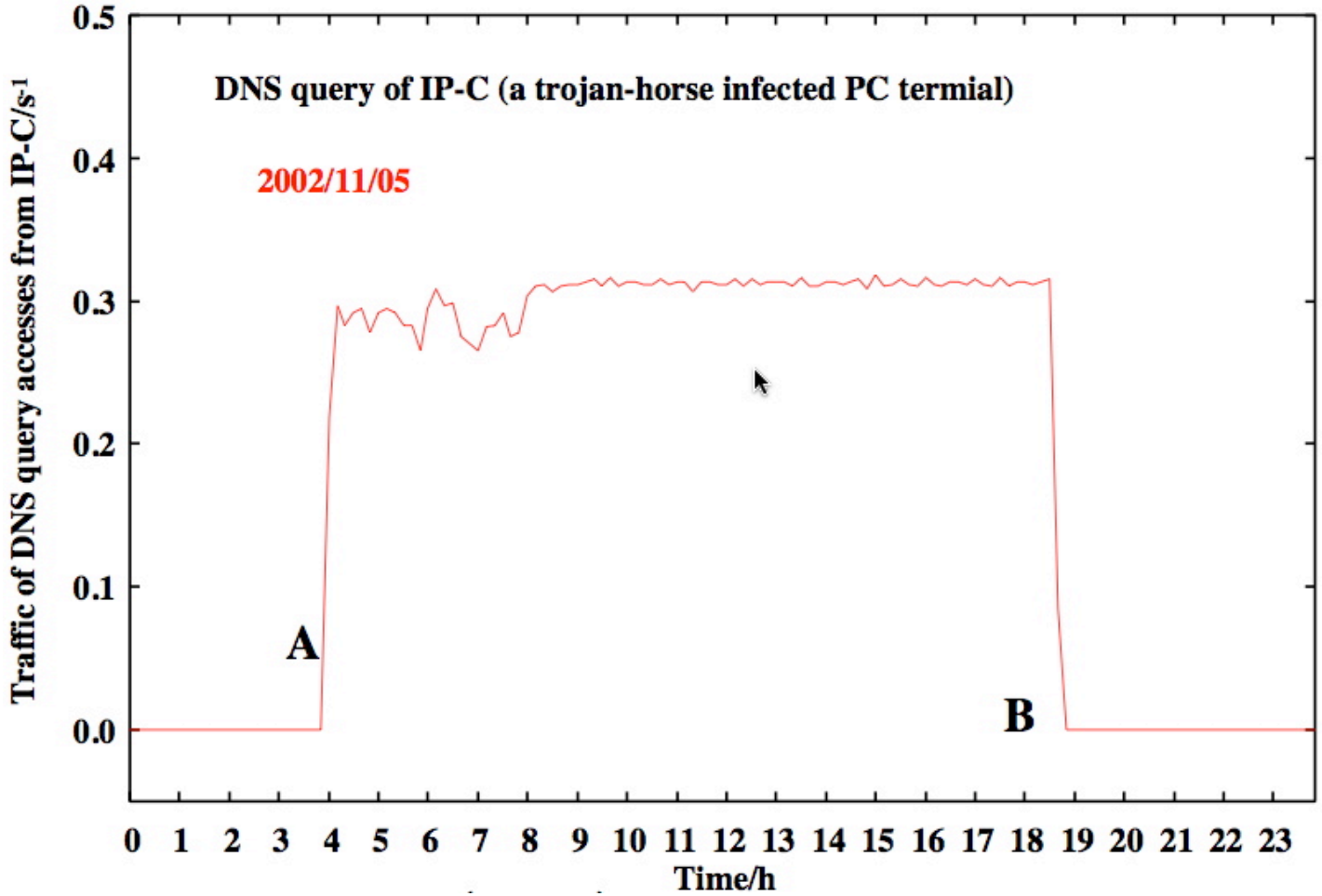
```
0 new queries, 38 total queries Wed Jun 1 17:35:51 2005


Destinations      count   %
---------------- --------- ------
172.19.226.120       29   77.7
10.158.254.13         5   15.0
172.19.220.131        4    9.3
```

DNS—Correlation

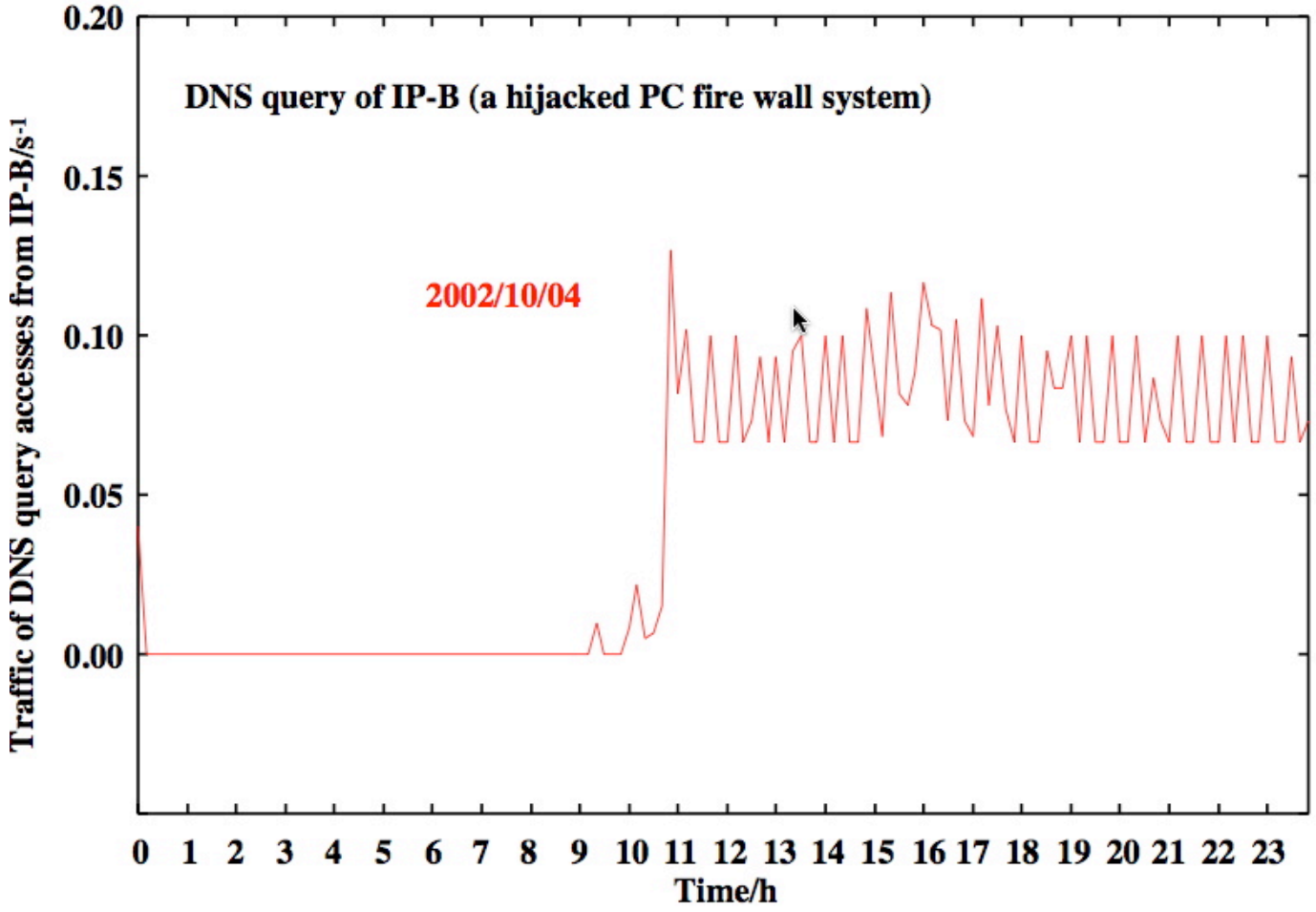
- By correlating DNS queries with other forms of telemetry and graphing, we can spot trends and infer root causes
- Kumamoto University in Tokyo have published several insightful papers on this subject—the tools they used were server logs (BIND, sendmail, QPopper), IDS, and grep
- In other words, you can try this at home!

Example—DNS Queries, Compromised PC



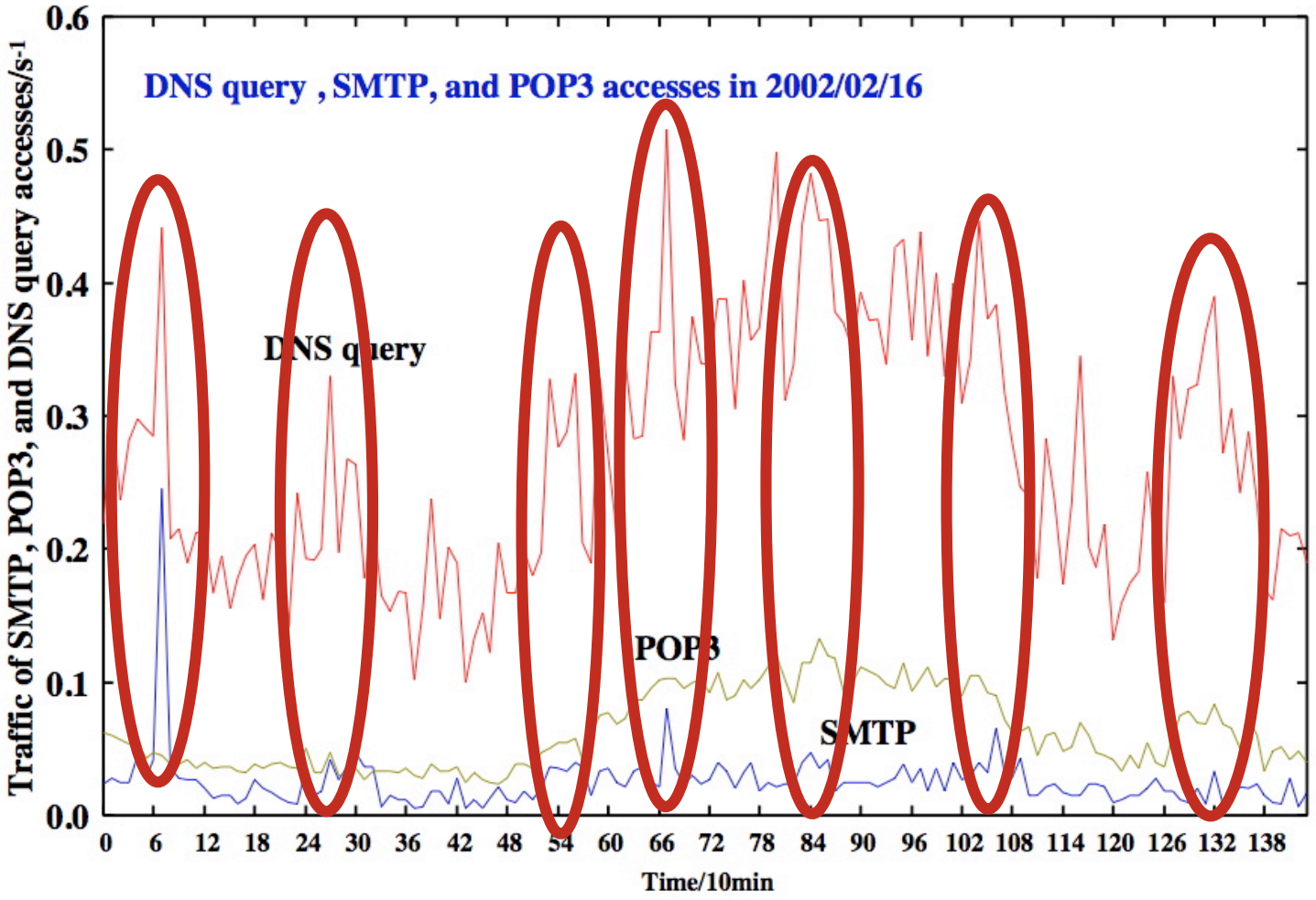
Source: Kumamoto University, Tokyo

Example—DNS Queries, Compromised PC-Based Firewall



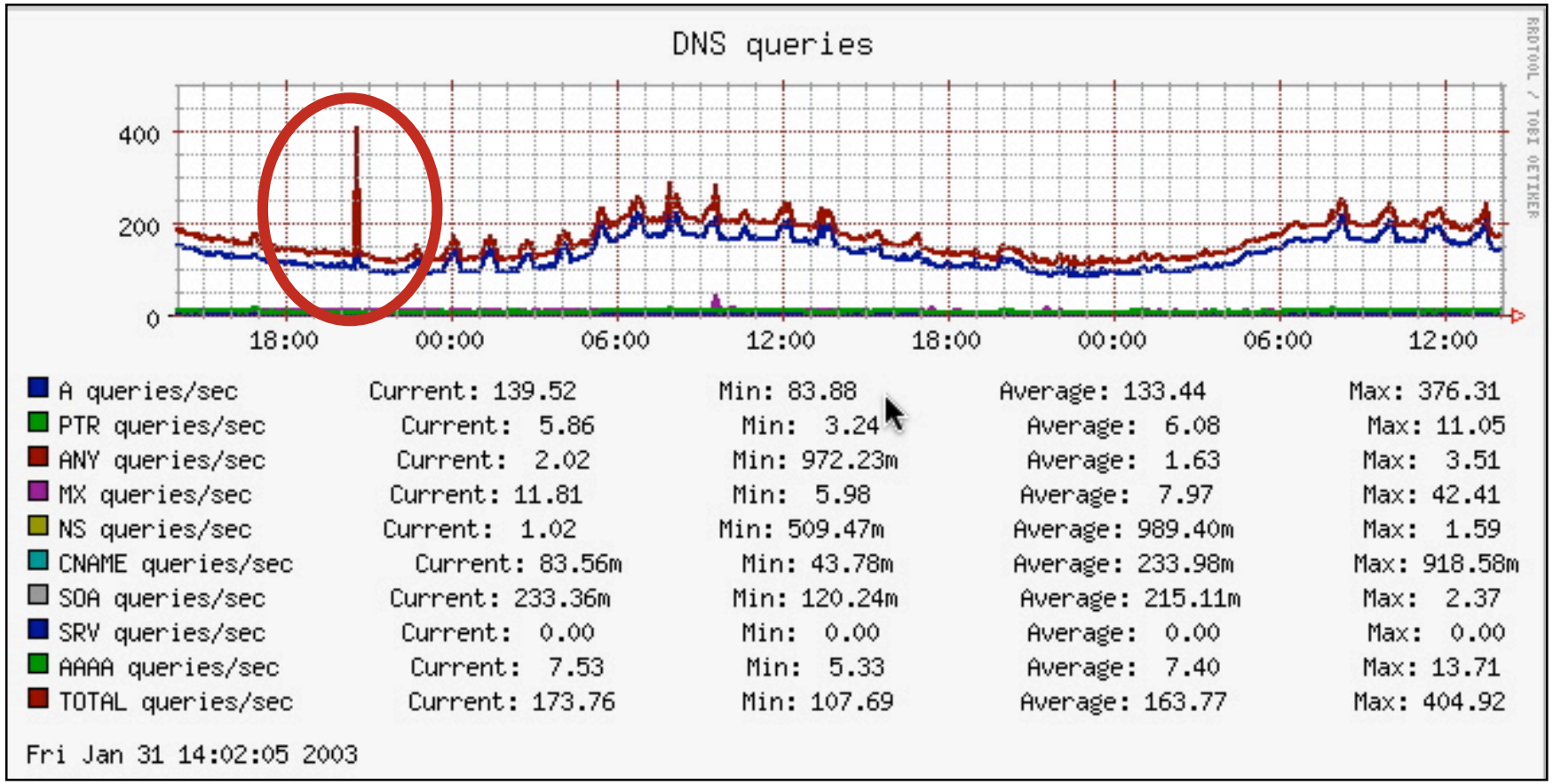
Source: Kumamoto University, Tokyo

Example—Correlating DNS and SMTP



Source: Kumamoto University, Tokyo

RRDTool Graph of DNS Queries/Sec



Source: http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/

DNS—More Information

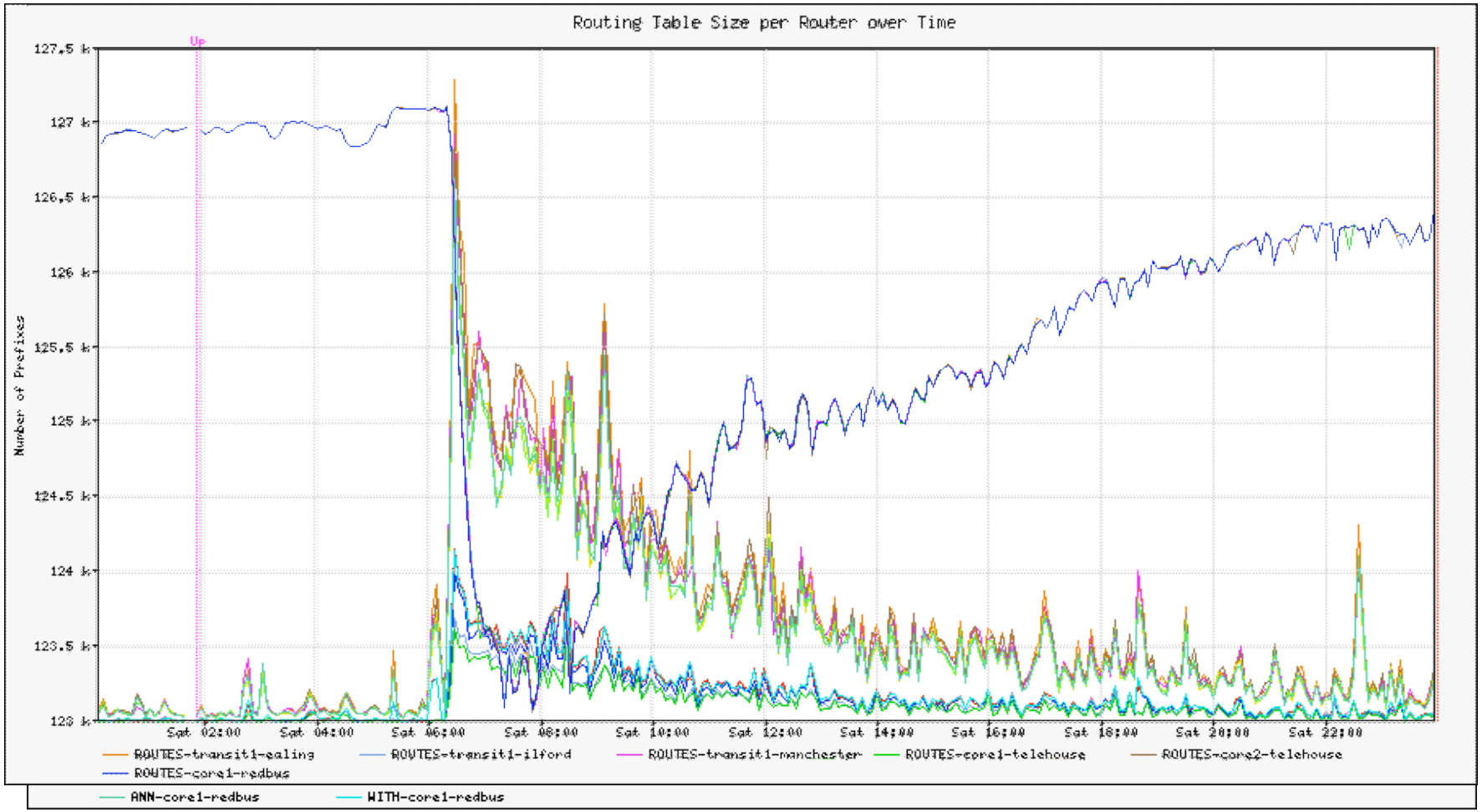
- dnstop home—
  http://dns.measurement-factory.com/tools/dnstop/
- dnslogger home—
  http://www.enyo.de/fw/software/dnslogger/
- Kunamoto University Papers on DNS-based Detection
  http://www.cc.kumamoto-u.ac.jp/~musashi/musashicsec27.pdf
  http://www.cc.kumamoto-u.ac.jp/~musashi/dsm32-12.pdf
  http://www.cc.kumamoto-u.ac.jp/~musashi/info2006-mcfsit.pdf
- Dan Kaminsky on DNS as a Covert Channel
  http://www.doxpara.com/dns_bh
- DNS as an IDS
  http://staff.science.uva.nl/~delaat/snb-2005-2006/p12/report.pdf
- Detecting Mass-Mailing Worms via DNS
  http://www.sigcomm.org/sigcomm2005/paper-IshToy.pdf

BGP

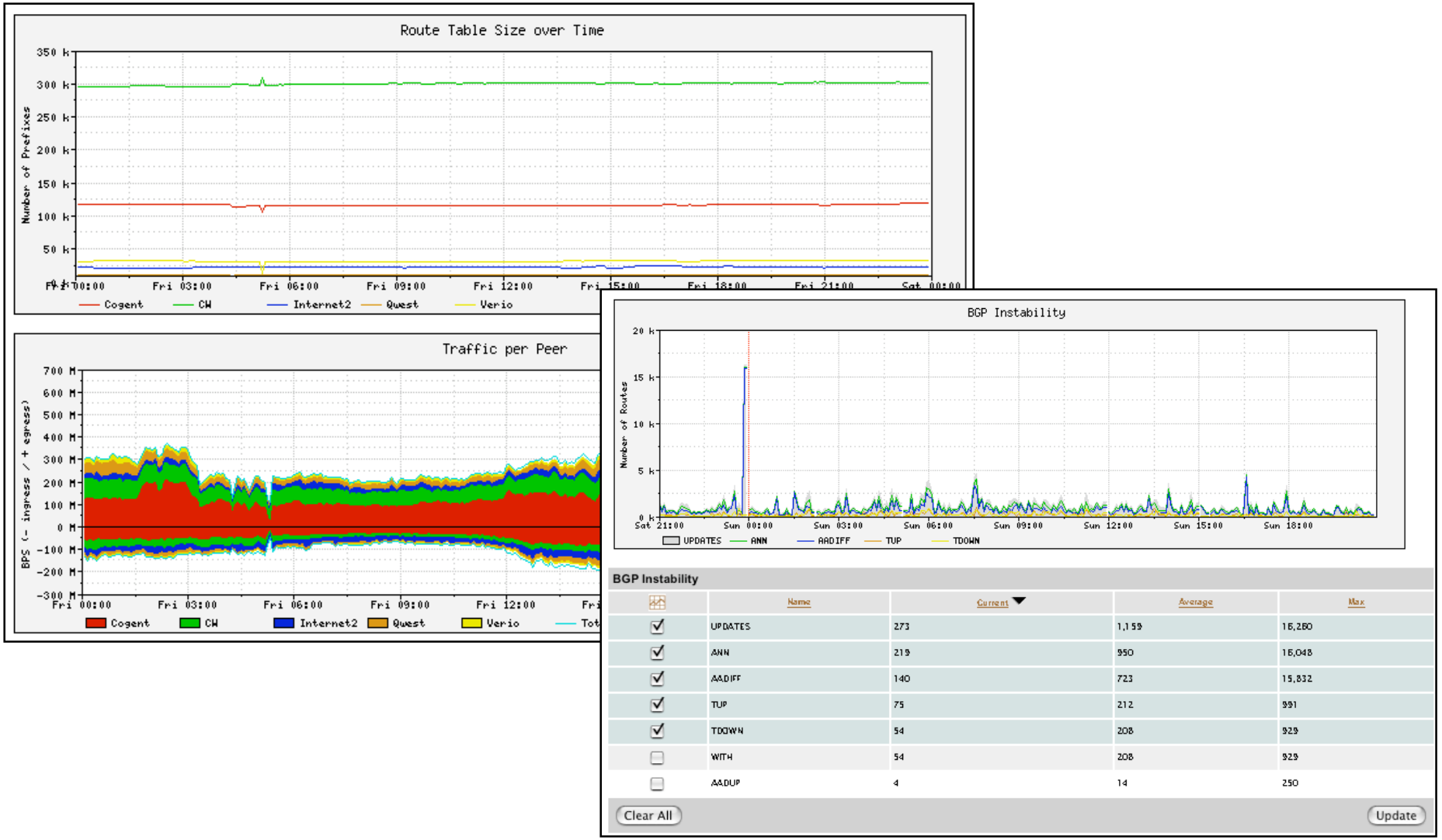
BGP

- Large-scale network security events such as worms, DDoS attacks, etc., often produce side-effects visible in the global routing table
- Correlating BGP information with other forms of telemetry (NetFlow, SNMP, RMON, etc.) can be effective in determining the true impact of incidents
- Zebra (http://www.zebra.org) and Quagga (http://www.quagga.net) are two open source BGP daemons which can log BGP updates for further analysis
- Arbor Peakflow SP Traffic provides BGP visualization, trending, NetFlow traffic correlation, additional functionality (http://www.arbornetworks.com/products_sp.php)
- RIBs/updates available from http://archive.routeviews.org/, http://www.ripe.net/ris/index.html, http://www.renesys.com (commercial, useful monitoring tools/services for your ASN)

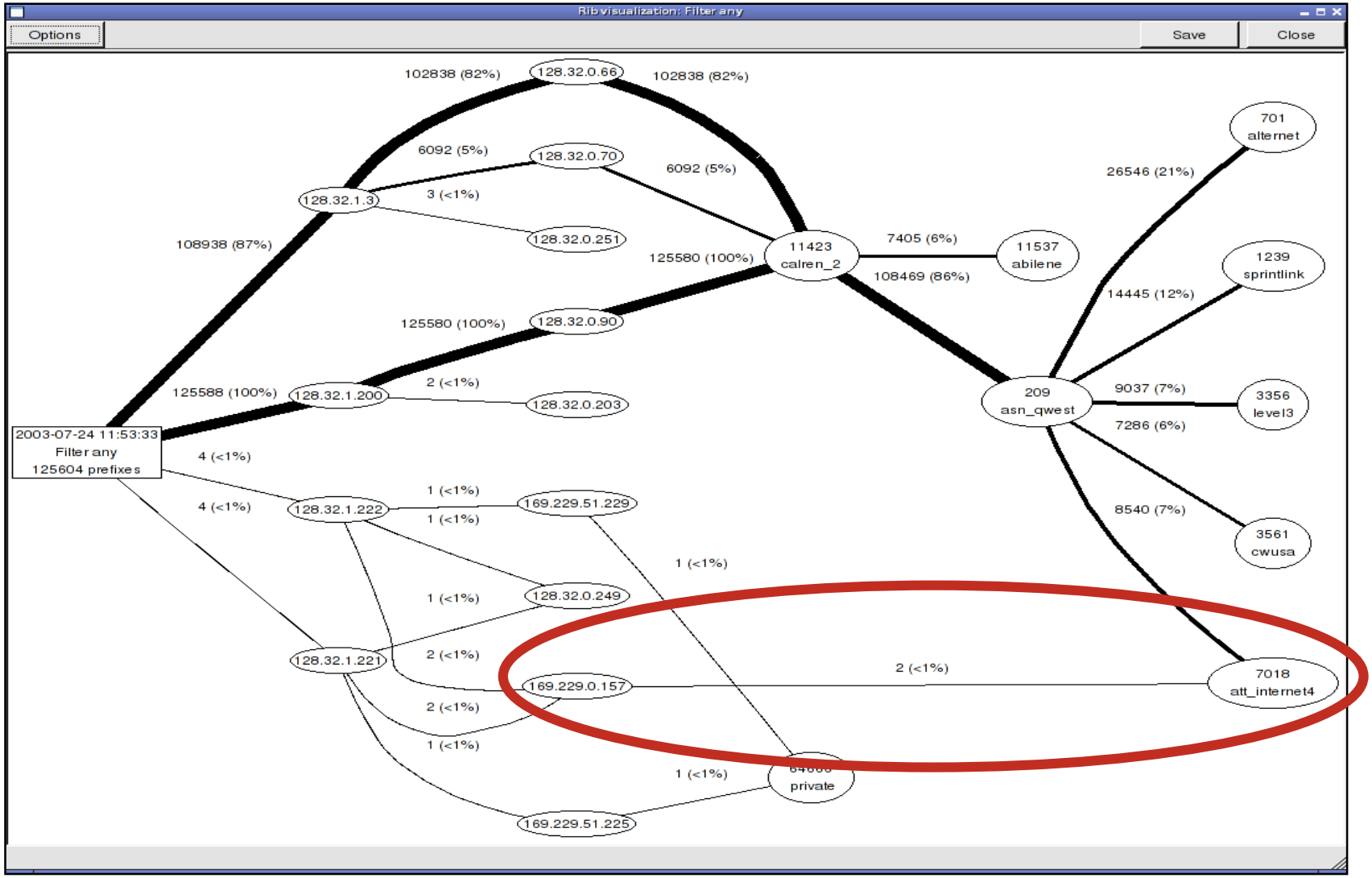
BGP Example—SQL Slammer



Arbor Example—Correlating BGP and NetFlow



Packet Design RouteExplorer—Detecting BGP Backdoor Routes

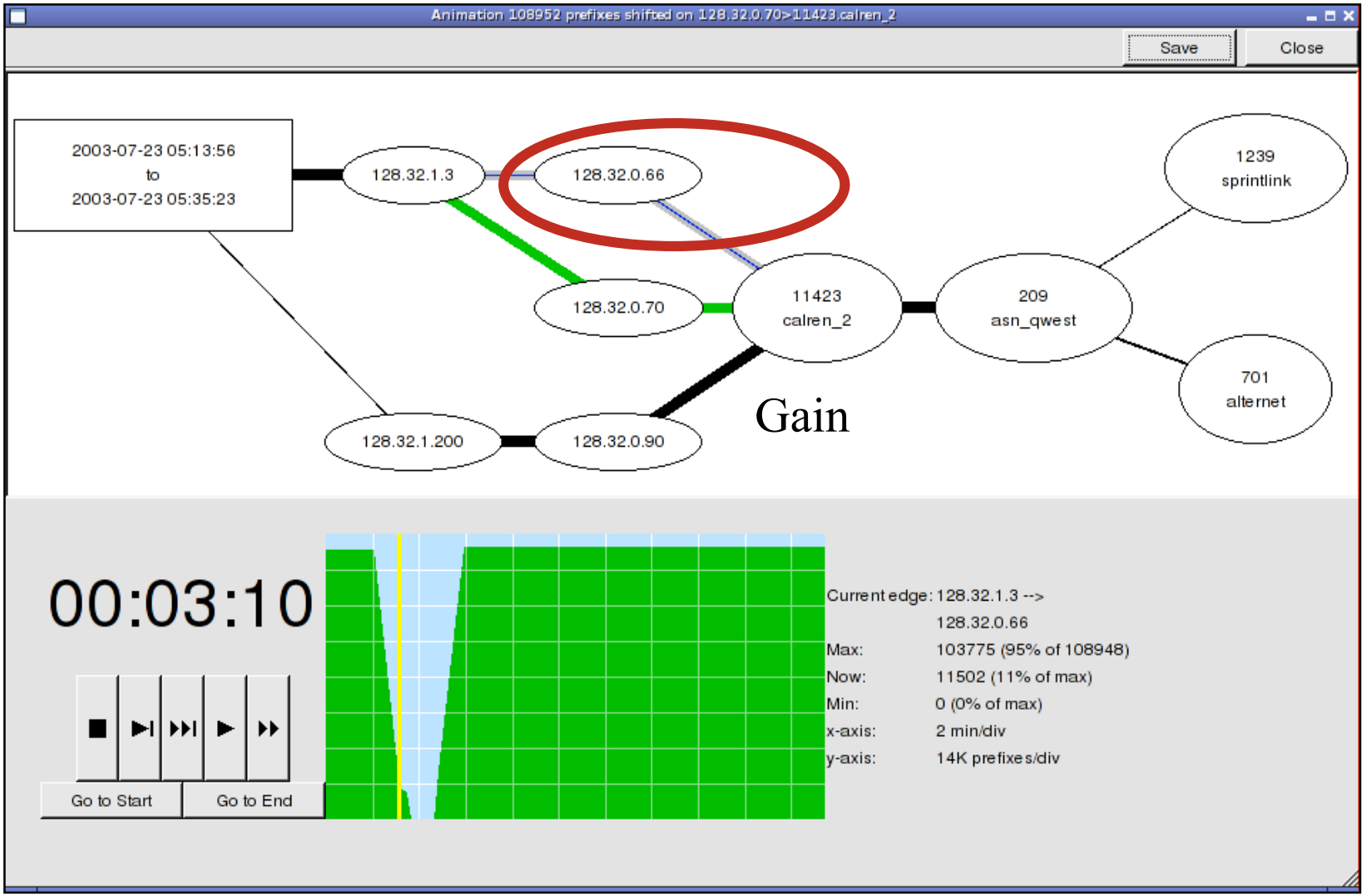


BGP Example—Two Backdoor Routes

- Hard to notice in `sh ip bgp` output
- This incident was due to a department having a ‘special arrangement’ to reach a certain AT&T customer
- Backdoor peering can have severe impact

  What if AT&T started sending full routes?

Packet Design—BGP Peer Reset



BGP—More Information

- Slammer/BGP analysis—

  http://www.nge.isi.edu/~masseyd/pubs/massey_iwdc03.pdf

- Team CYMRU BGP Tools—

  http://www.cymru.com/BGP/index.html

- Packet Design Route Explorer—

  http://www.packetdesign.com/products/rex.htm

- Team CYMRU BGP Tools—

  http://www.cymru.com/BGP/index.html

Summary

How to Get Started

- Which of these methods are you using now? If you have syslog, is it being stored in a searchable database
- If you're starting from scratch, begin with some Cisco IOS CLI shortcuts—it's the most basic and ubiqitous form of telemetry available. Study and understand differences between your various platforms
- Next is syslog—not just from routers, firewalls, and switches, but from hosts, applications (think DNS, TACACS), etc.
- Then—or perhaps in parallel—get started with NetFlow. It is the single most operationally-useful source of telemetry available, and it's built into the routers. Learn how to use NetFlow via the CLI, and then start exporting it, graphing it, and investigate anomaly-detection
- SNMP (and perhaps RMON) require a significant investment in resources, but the payoff is worth the effort. Again, not just routers/switches/firewalls—hosts, applications, etc., have MIBS and traps. Use them
- Finally, make use of advanced methods such as DNS mining. You'll learn a heck of a lot about user/server/application interaction, and can use it to help detect spamming, botted hosts, etc.

Summary

- Detection and classification are vital to ensuring the security and availability of the network and the services which traverse it
- There are many different forms of network telemetry, from counters on the CLI to NetFlow to routing protocols
- There are many open source, Cisco, and Cisco Partner tools available to help us generate and analyze network telemetry in an operationally-significant manner
- It isn't expensive to get started—free tools, free *NIX and surplus hardware abound
- Scripting knowledge helps, but there are plenty of tools which are useful out-of-the-box
- Yes, you can (and should) try this at home!

Additional References

- Product security:

  Cisco's product vulnerabilities; a page that every SE must know [http://www.cisco.com/warp/public/707/advisory.html]

  Security reference information: Various white papers on DoS attacks and how to defeat them [http://www.cisco.com/warp/public/707/ref.html]

- ISP essentials:

  Technical tips for ISPs every ISP should know [ftp://ftp-eng.cisco.com/cons/isp/]

- Technical tips:

  Troubleshooting high CPU utilization on Cisco routers [http://www.cisco.com/warp/public/63/highcpu.html]

  The "show processes" command [http://www.cisco.com/warp/public/63/showproc_cpu.html]

  NetFlow performance white paper [http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/ntfo_wp.htm]

- Mailing lists:

  Cust-security-announce: All customers should be on this list

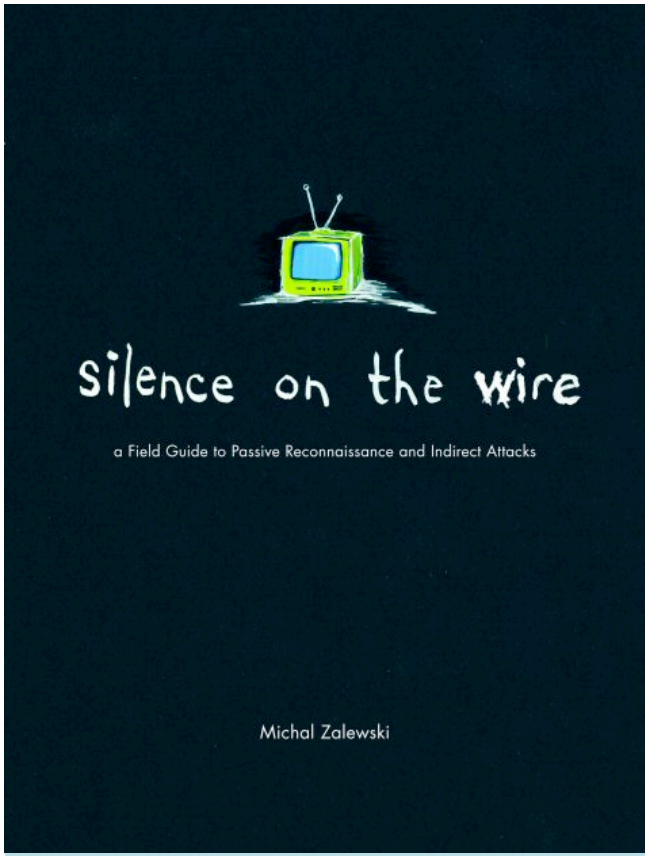
  Cust-security-discuss: For informal discussions

Recommended Reading

Silence on the Wire

by Michael Zalewski

ISBN: 1593270461

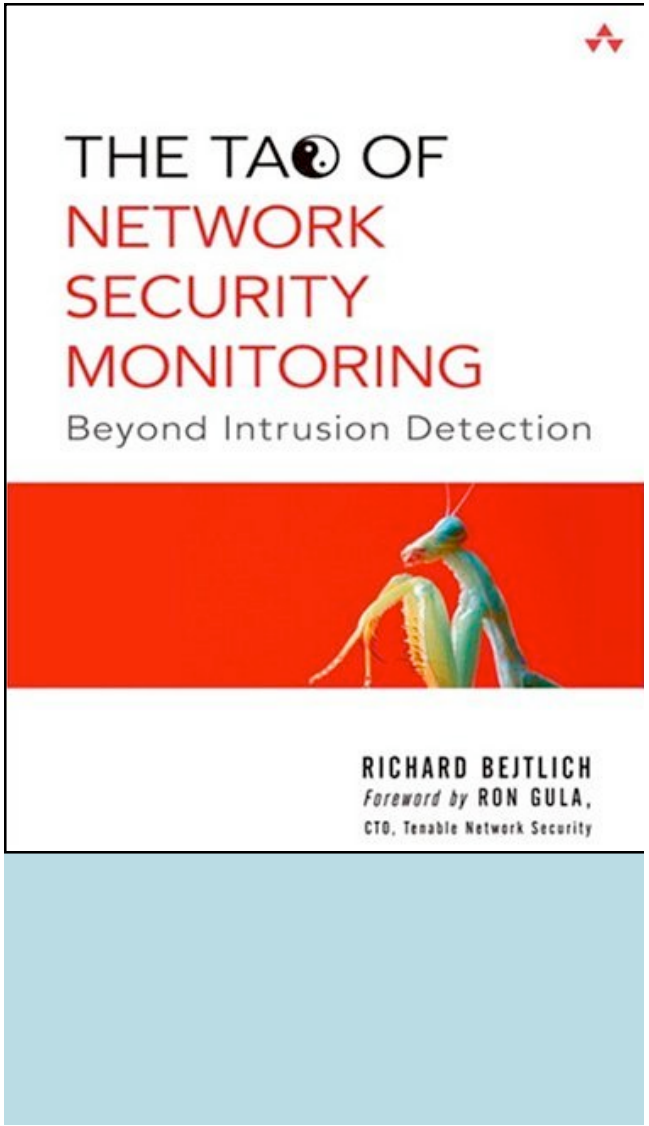


Recommended Reading

The Tao of Network Security Monitoring

by Richard Beitlich

ISBN: 0321246772

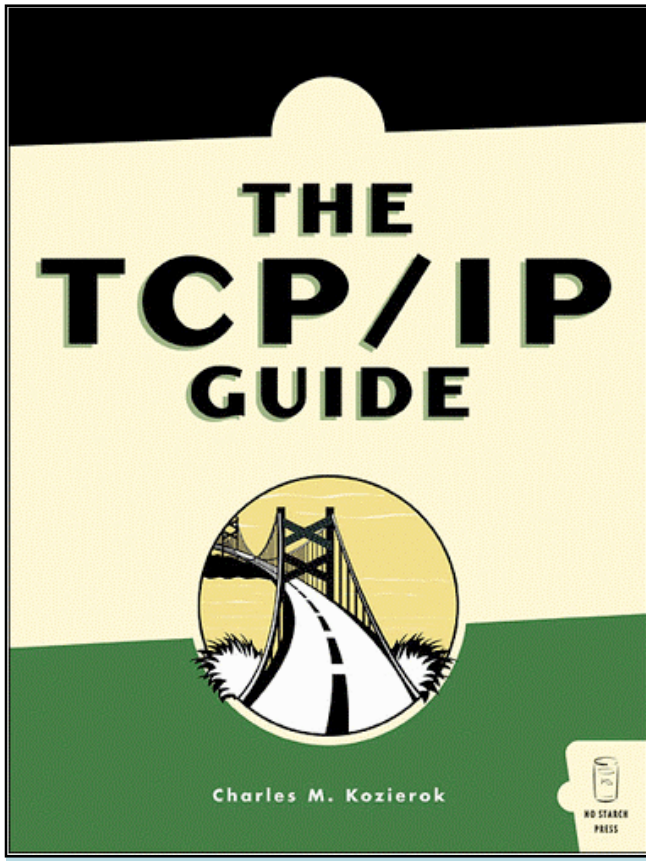


Recommended Reading

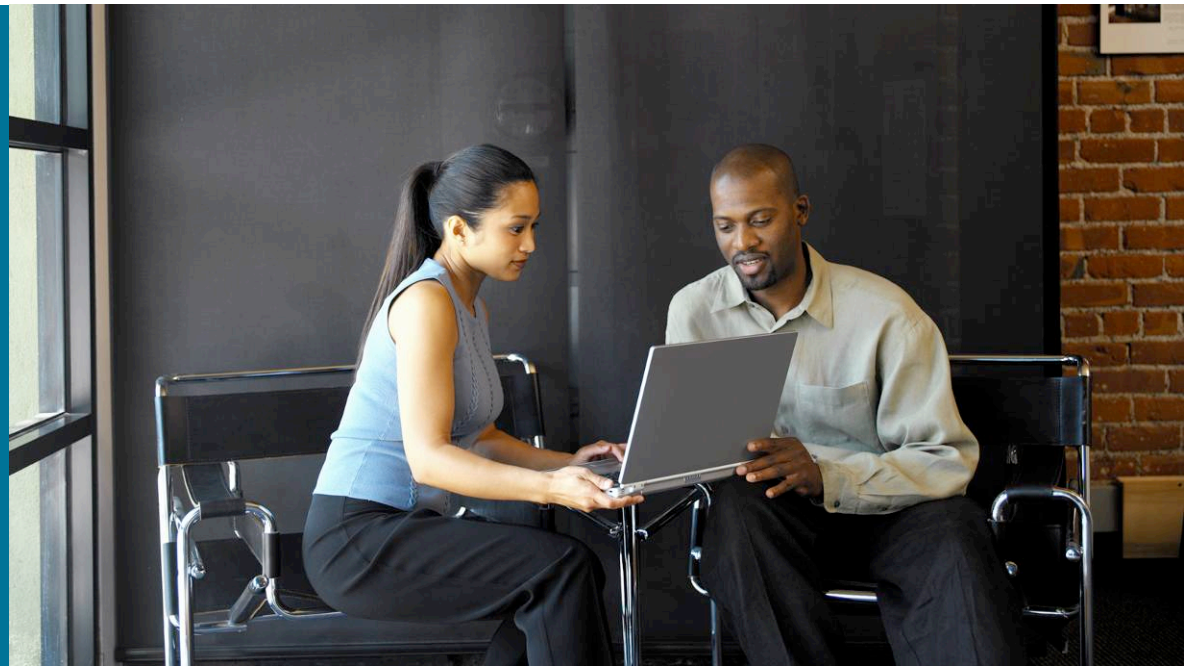
The TCP/IP Guide

by Charles M. Kozierok

ISBN: 159327047X



Q and A

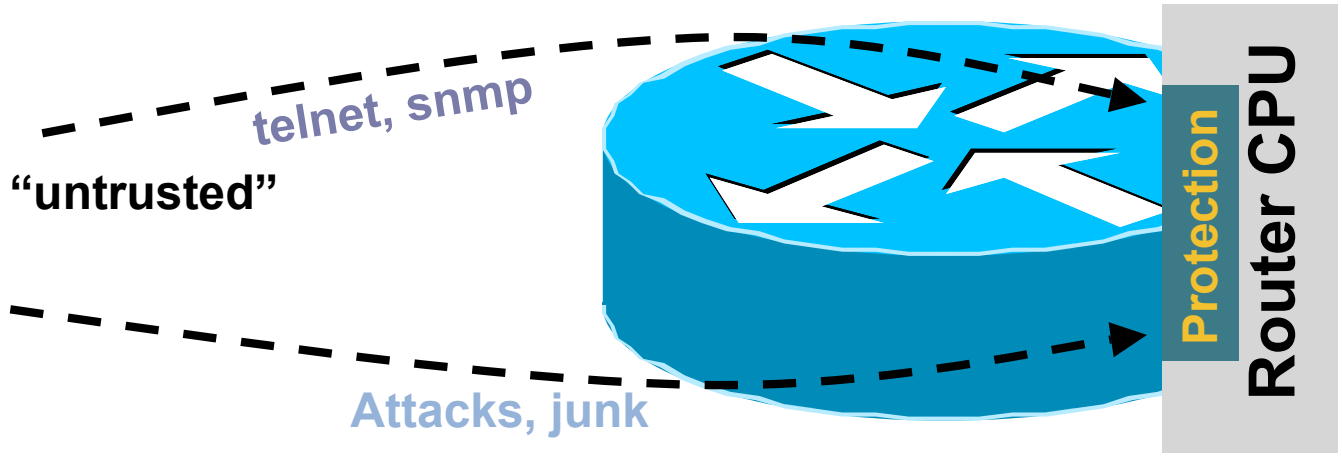


Part II
Protecting the Infrastructure

Agenda

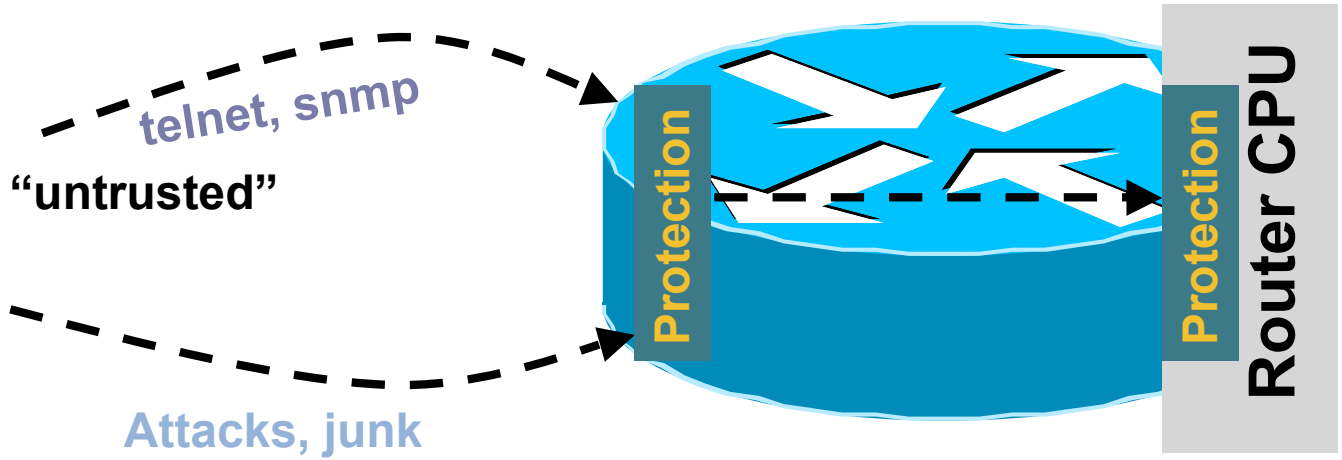
- Infrastructure Protection Overview
- Understanding Routers and Planes
- Infrastructure Protection from the Inside Out
  - Router Hardening: Traditional Methods
  - Router Hardening: Protecting the CPU
  - Network Hardening

Router Hardening: Traditional Methods



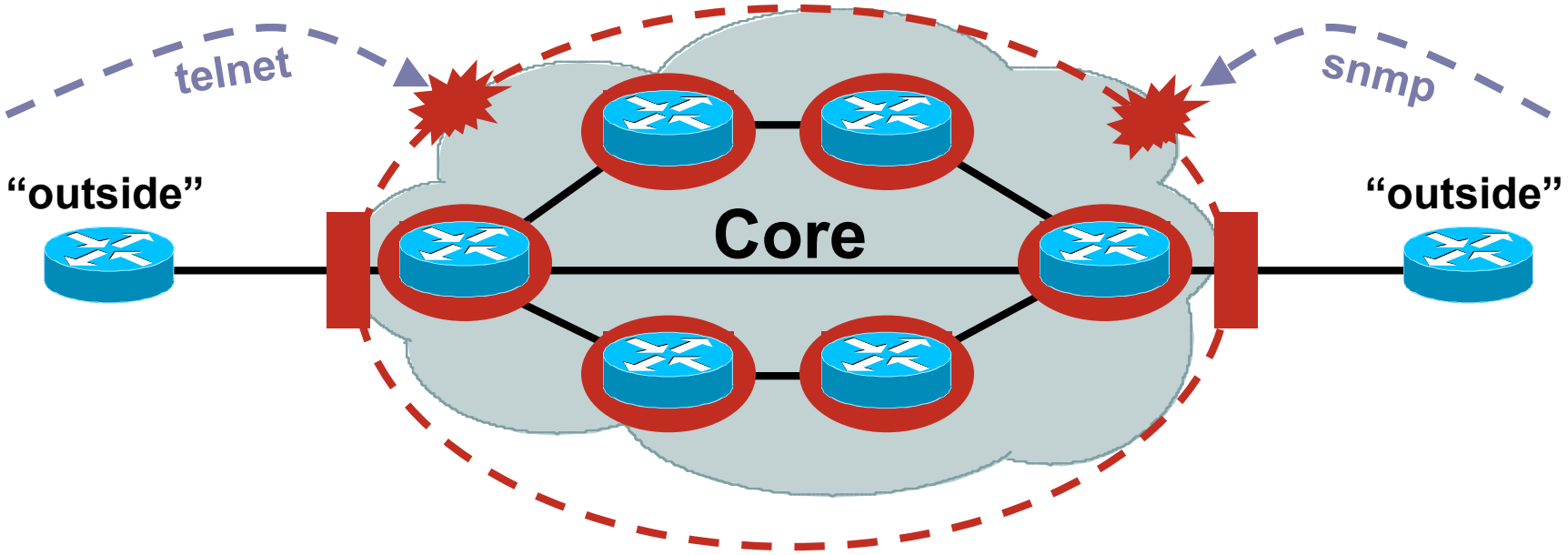
- We will look at best practices on securing the CPU

Router Hardening: Protecting the CPU



- We will look at best practices on preventing unwanted traffic from reaching the CPU

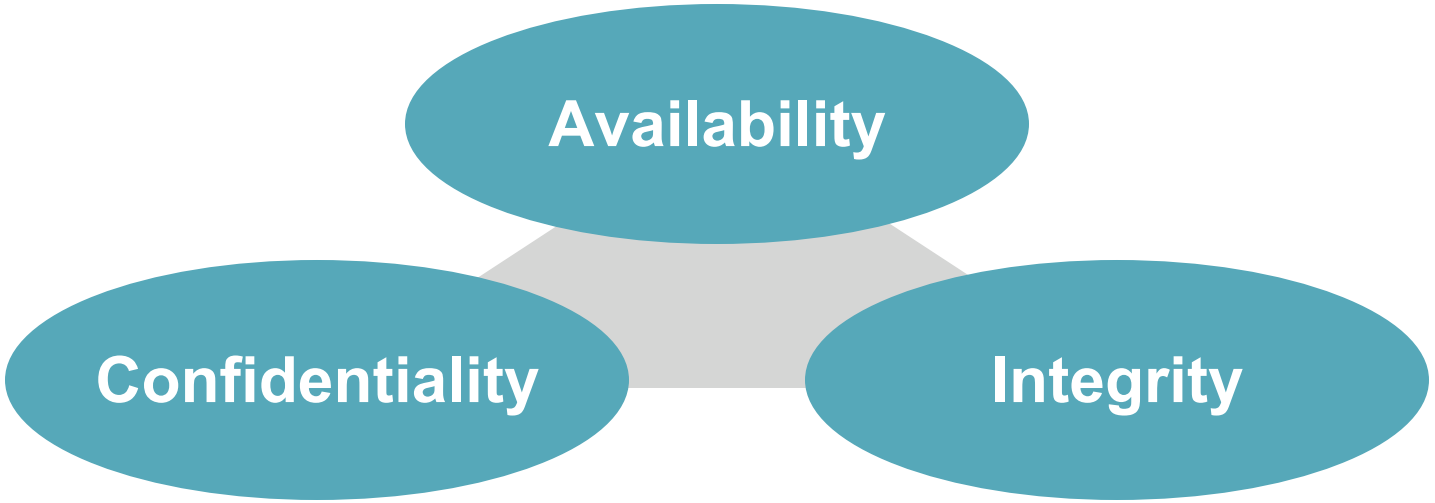
Network Hardening



- We will look at best practices on preventing unwanted traffic from reaching the core routers

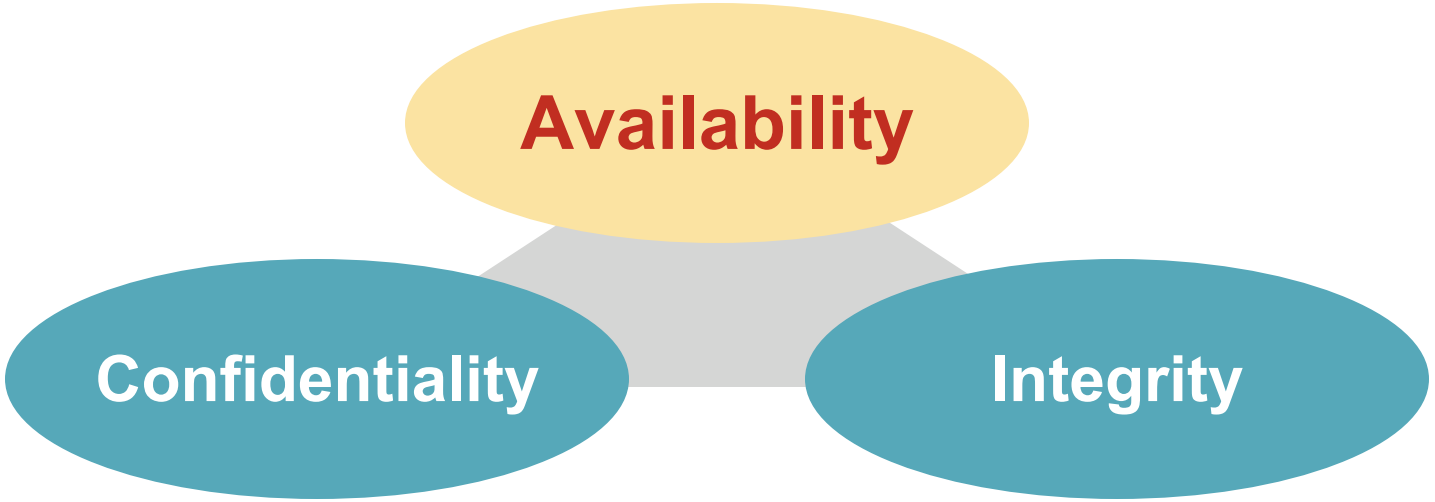
Infrastructure Protection Overview

Three Security Characteristics



- The goal of security is to maintain these three characteristics

Three Security Characteristics



- Primary goal of infrastructure security and this session is maintaining availability

Network Availability: Protect the Infrastructure

- Security is the heart of internetworking's future; we have moved from an Internet of implicit trust to an Internet of pervasive distrust
- No packet can be trusted; all packets must earn that trust through a network device's ability to inspect and enforce policy
- Protecting the infrastructure is the most fundamental security requirement
- Infrastructure protection should be included in all high availability designs
- A secure infrastructure forms the foundation for continuous service delivery

Understand the Threats

- Internal
  - Inadvertent human error (fat finger attack)
  - Malicious insider
- External
  - Worms
  - Packet floods
  - PSIRT vulnerability
  - Intrusion
  - Route hijacking
  - Service attacks (DNS, voice, etc.)

Understand the Threats

- Internal
  - Inadvertent human error (fat finger attack)
  - Malicious insider
- External
  - Worms
  - Packet floods
  - PSIRT vulnerability
  - Intrusion
  - Route hijacking
  - Service attacks (DNS, voice, etc.)

Taking a Measured Approach

The techniques we will be discussing are extremely useful, but must be applied in an architecturally sound, situationally appropriate, and operationally feasible manner

- Don’t try to do all of this at once—pick a technique with which you are comfortable and which you think will benefit you the most
- Pilot your chosen technique in a controlled manner, in a designated portion of your network
- Take the lessons learned from the pilot and work them into your general deployment plan and operational guidelines
- It is not uncommon to take 9–12 months to deploy

Understanding Routers and Planes

Routers and Planes

- A network device typically handles traffic in several different forwarding planes
- There are nuances to the definition of these planes

  IETF RFC3654 defines two planes: control and forwarding

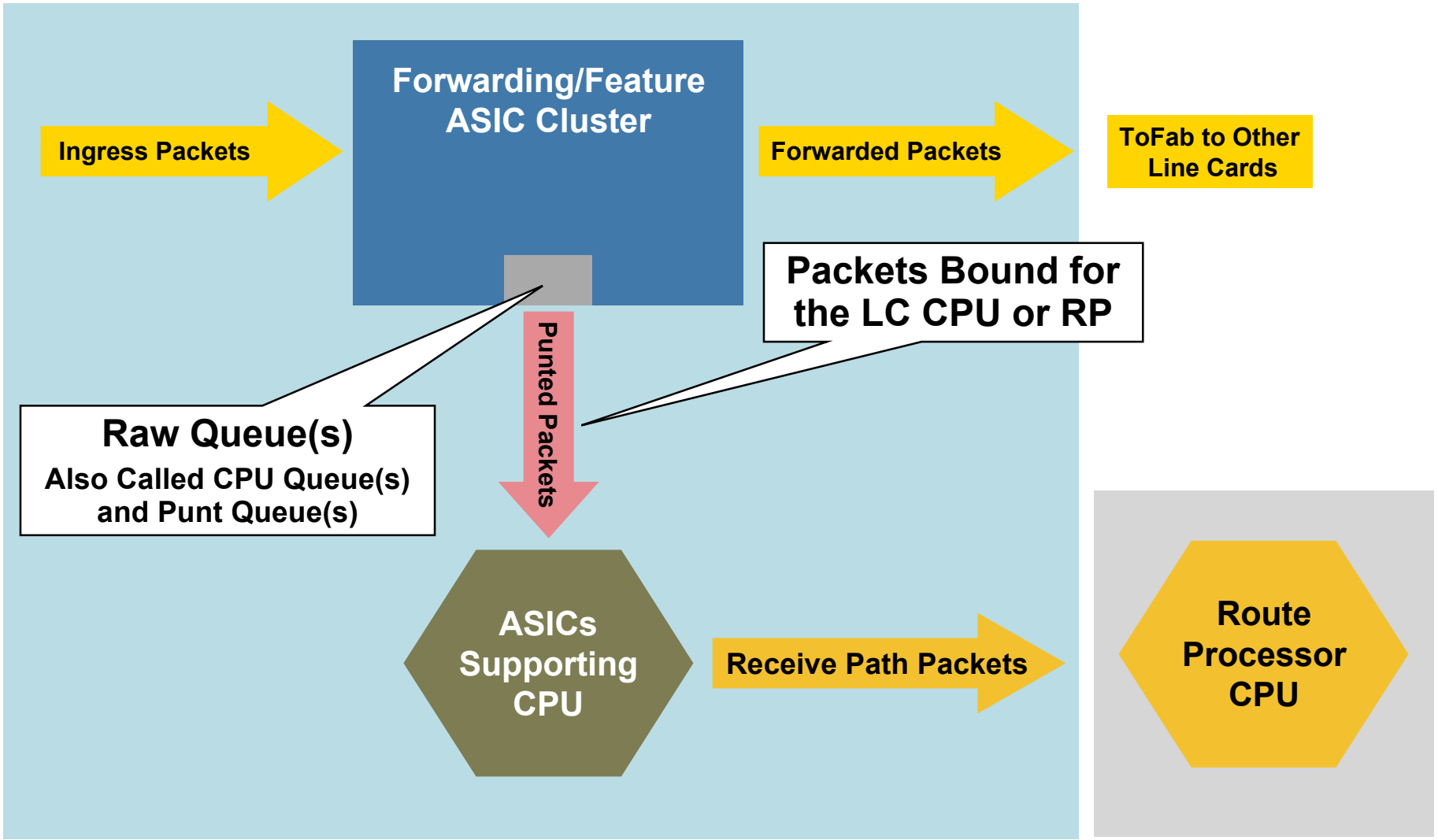
  ITU X805 defines three planes: control, management, and end-user

  Cisco defines three planes: control, management, and data

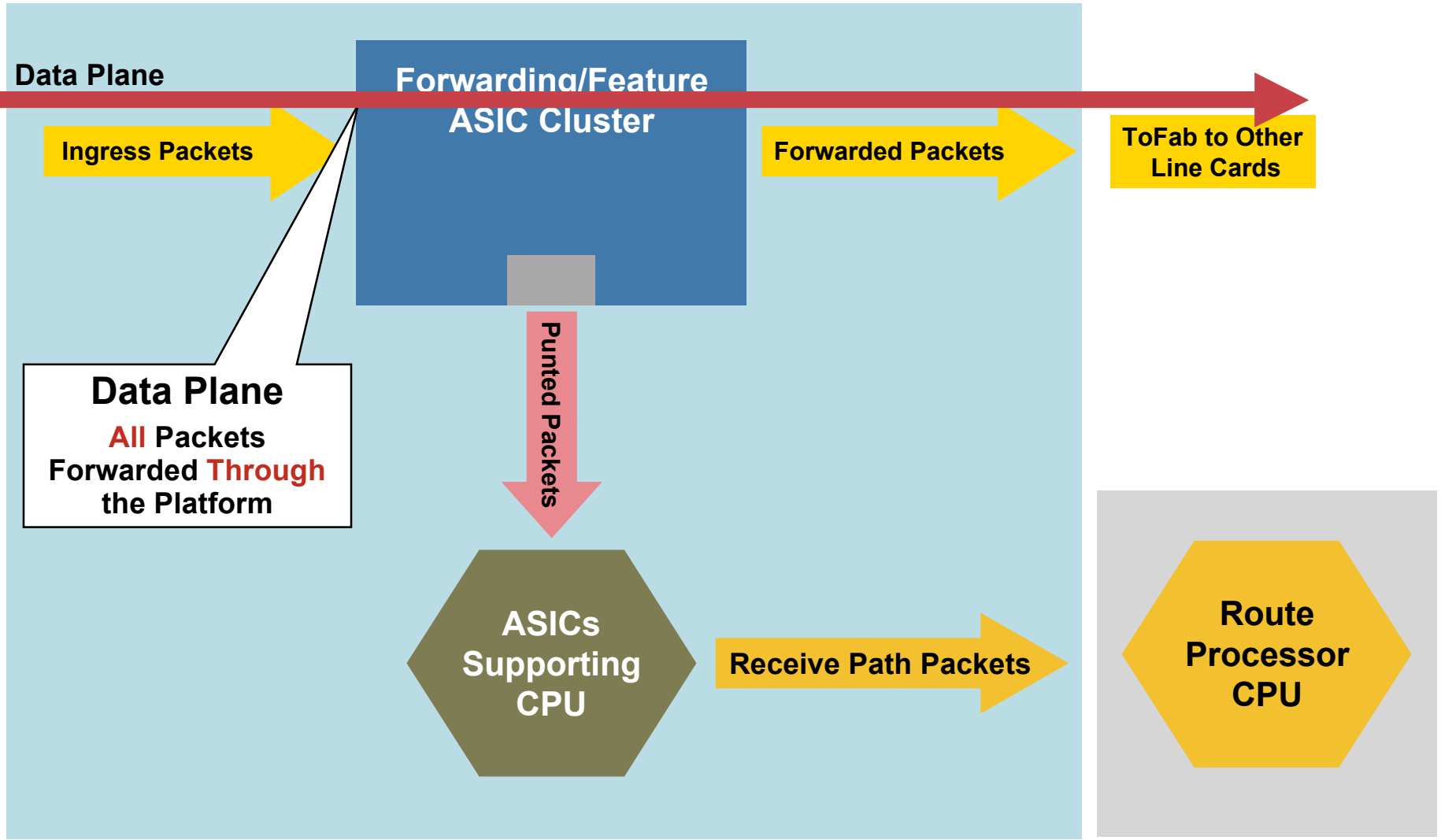
Routers and Planes

- Traffic to the control and management plane is always destined to the device and is handled at process level ultimately:
  - In hardware switched platforms, control/management plane traffic is sent to the RP/MFSC and then sent to the process level for processing
  - In software switched platforms, it is sent directly to the process level for processing
- Traffic in the data plane is always destined through the device and is:
  - Implemented in hardware on high-end platforms
  - CEF switched (in the interrupt) in software-switched platforms
- Some data plane traffic may also reach the control plane
  - Packets that are not routable reach to control plane so that ICMP unreachable messages can be generated
  - Packets that have IP options set are also handled by the processor

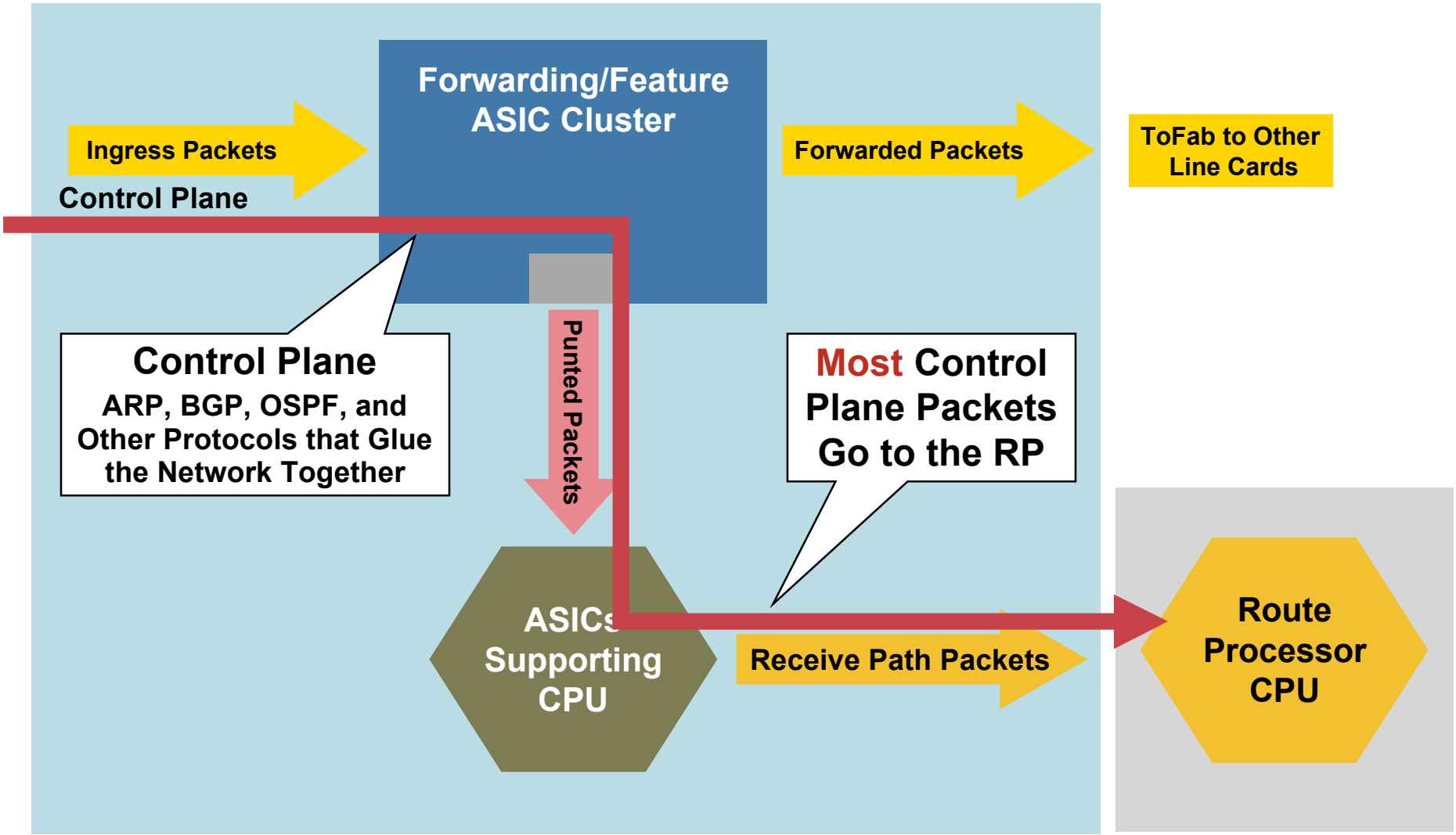
ASIC Based Platform—Main Components



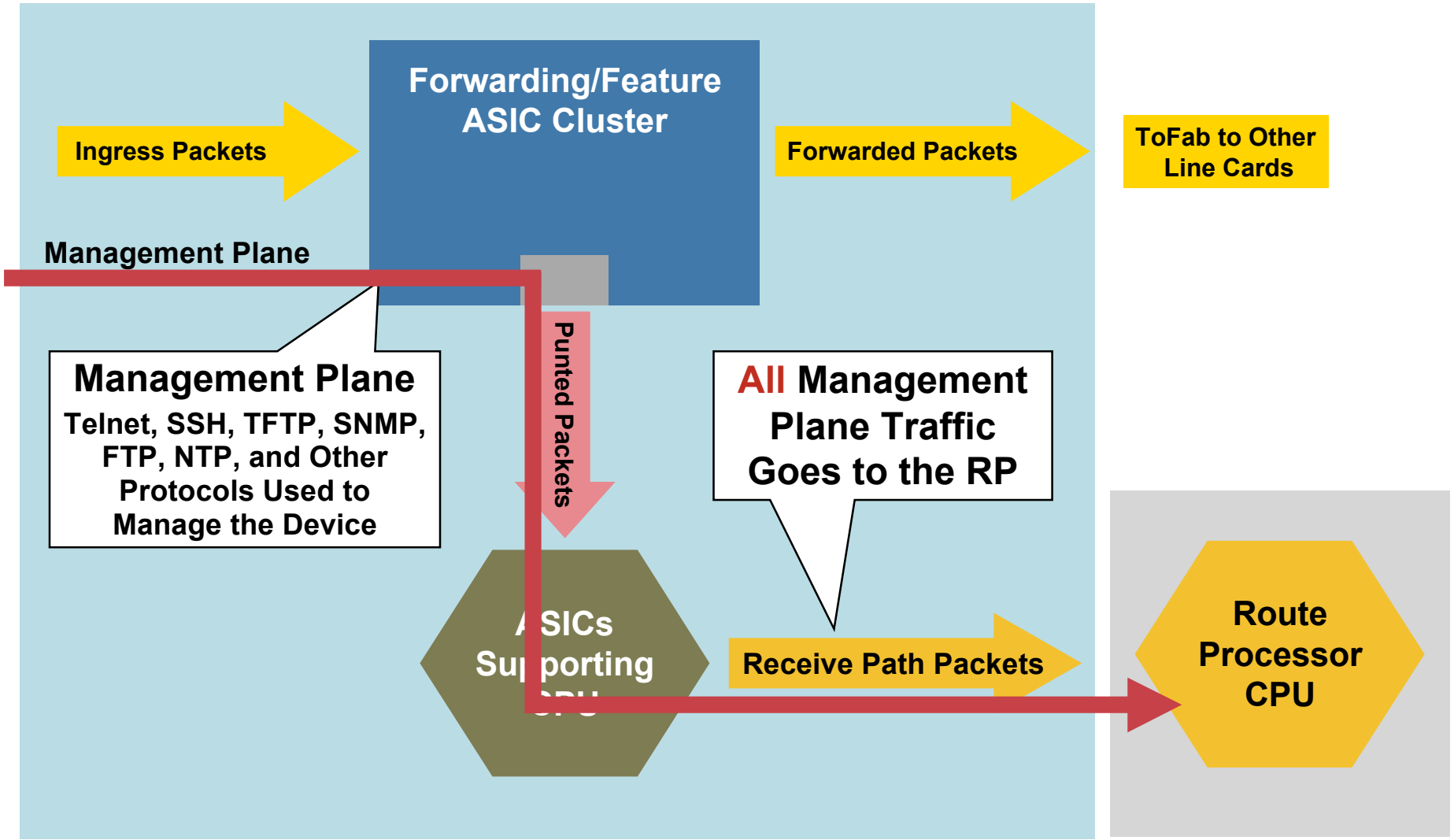
Data Plane



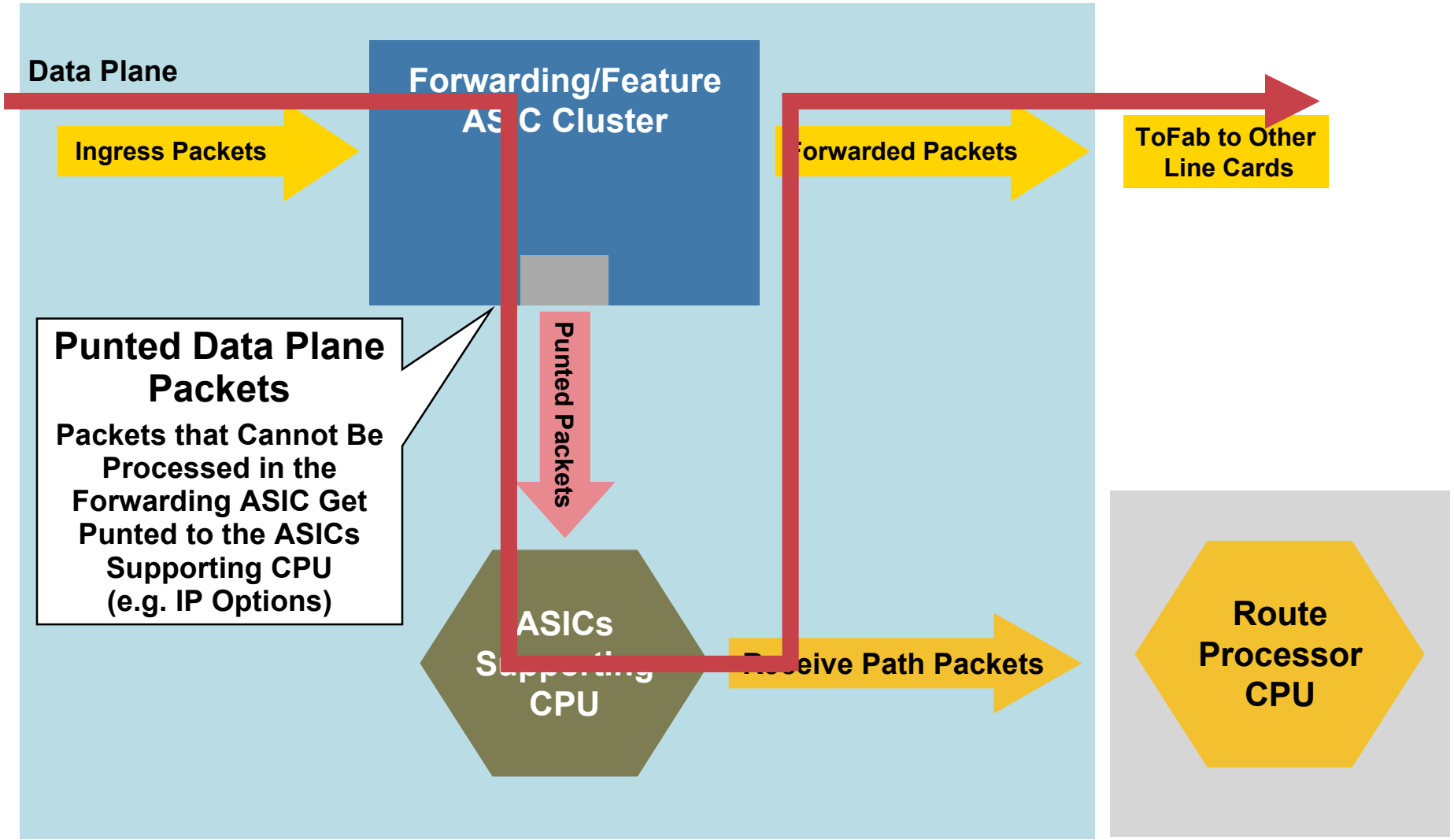
Control Plane



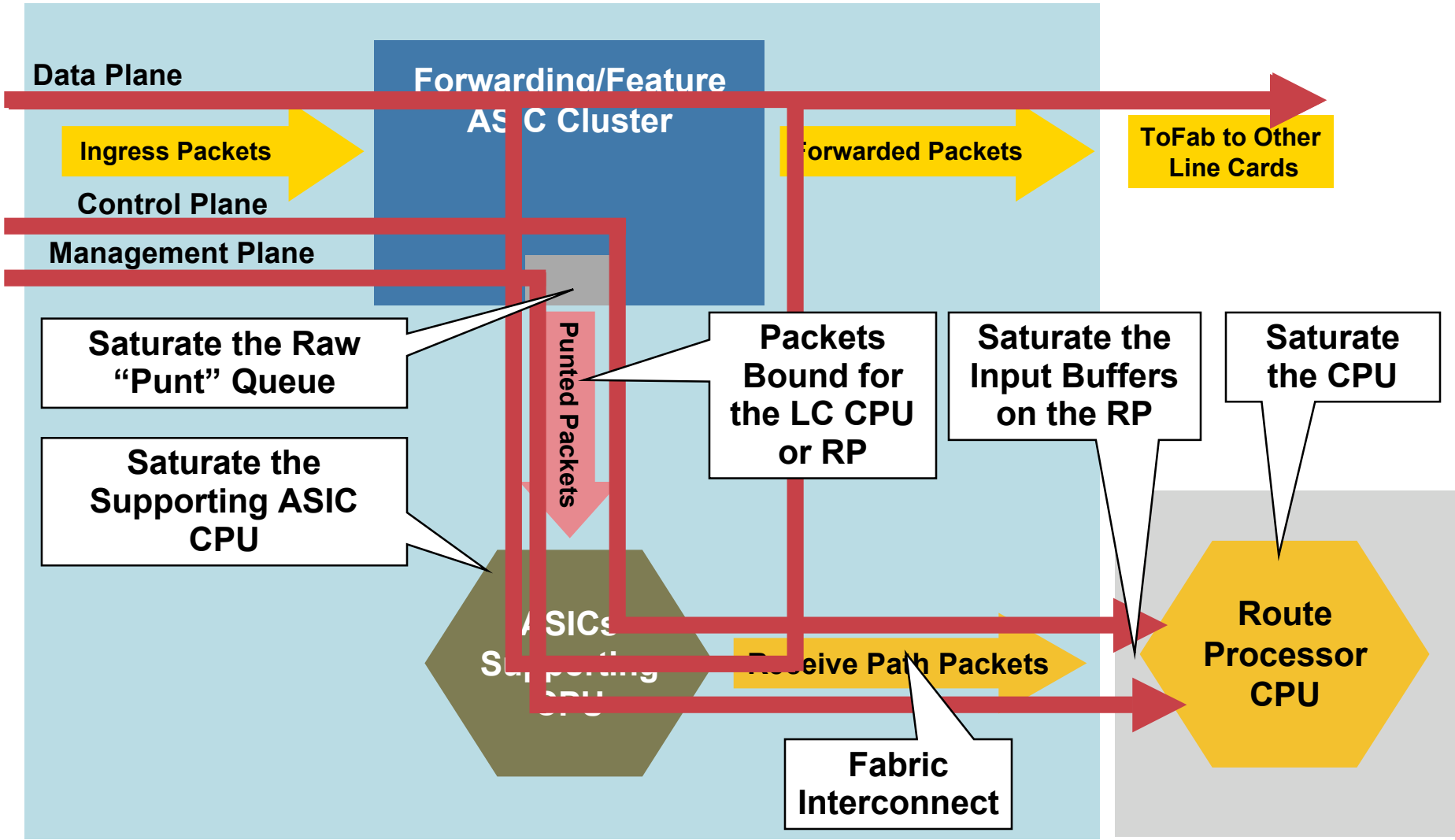
Management Plane



Feature Punt



Punt-Path Attack Vectors



Router Hardening: Traditional Methods

Router Security Best Practices

- Many organizations publish guides to best practices around router security
- In addition to CCO resources, these include:

  http://www.first.org/resources/guides/

  http://www.sans.org/resources/policies/

  http://www.ietf.org/html.charters/opsec-charter.html
- These guides do a good job of documenting best practices, especially in what we are referring to as traditional methods for router hardening
- Therefore, we will just quickly review a sample of the key points and features

Router Hardening: Traditional Methods

- Disable any unused protocols
  - no service tcp-small-servers
  - no cdp run
- VTY ACLs
- SNMP Community ACL
- SNMP views
- Disable SNMP RW
  - Use SNMPv3 for RW if needed
- Prevent dead TCP sessions from utilizing all VTY lines
  - service tcp-keepalives-in
- Edge QoS enforcement
- Use secret password
  - "service password encryption" is reversible and is only meant to prevent shoulder-surfing
- Run AAA
  - Don't forget Authorization and Accounting
- Disable extraneous interface features
  - no ip directed-broadcast
  - no ip proxy-arp
  - no ip redirects

Router Hardening: Traditional Methods

- Source address validation (RFC2827/BCP38, RFC3704/BCP84)

```
ip verify unicast source reachable-via {any|rx}
cable source-verify [dhcp]
ip verify source [port-security]
```

- Disable source-routing

```
no ip source-route
```

- Prefix-list filtering on eBGP peers
- BGP dampening
- MD5 on BGP and igp

- Hardware-dependent issues
  - –Control ICMP unreachable generation

```
ip icmp rate-limit unreachable
ip icmp rate-limit unreachable DF
interface null0
 no ip unreachables
```

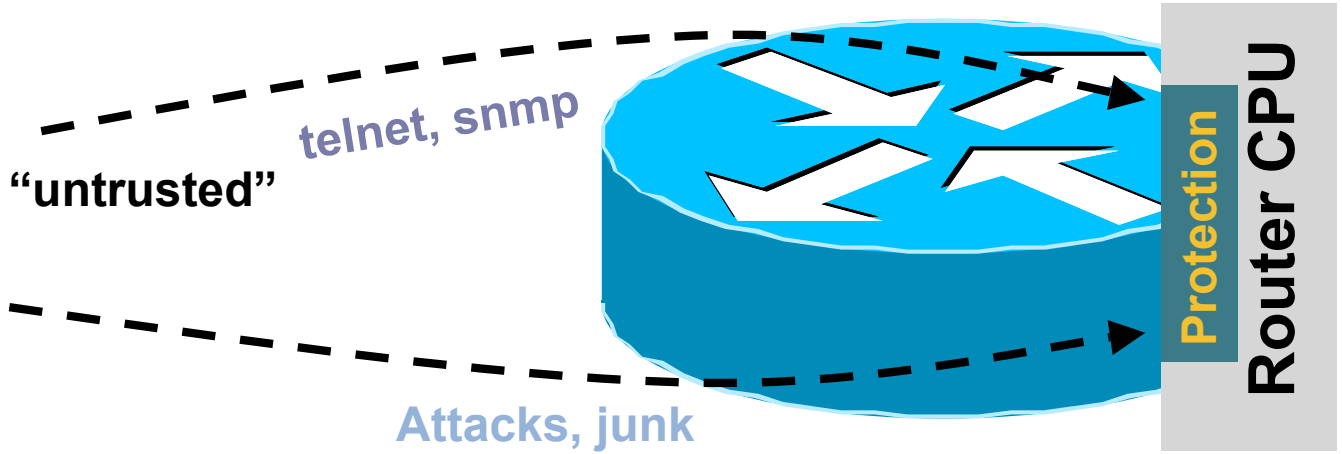
  - –Ensure CPU cycles for management

```
scheduler allocate
```

  - –Selective Packet Discard (SPD)

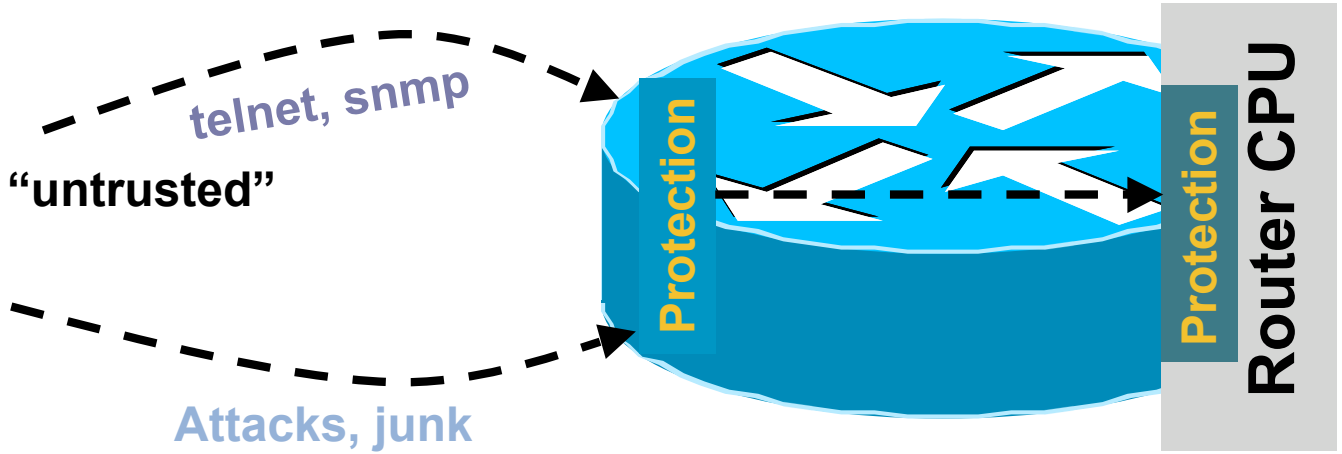
Router Hardening: Protecting the CPU

The Old World: Router Hardening



- Policy enforced at process level (VTY ACL, SNMP ACL, etc.)

The New World: Router Hardening



- Central policy enforcement, prior to process level
- Granular protection schemes
- On high-end platforms, hardware implementations

Router Hardening: Protecting the CPU Receive Access-Lists

Receive ACL Command

- Introduced in:

  12000: 12.0(21)S2/12.0(22)S

  7500: 12.0(24)S

  10720: 12.0(31)S

  Router(config)# ip receive access-list [number]

- Standard, extended, or compiled ACL
- As with other ACL types, show access-list provide ACE hit counts
- Log keyword can be used for more detail

Receive ACLs (rACLs)

- Receive ACLs filter traffic destined to the RP via receive adjacencies (control and management plane only)
- rACLs explicitly permit or deny traffic destined to the RP
- rACLs do not affect the data plane
- Traffic is filtering on the ingress line card (LC), prior to route processor (RP) processing
- rACLs enforce security policy by filtering who/what can access the router

Receive Adjacencies

- CEF entries for traffic destined to router, not through it
  - Real interface(s)
  - Loopback interface(s)

```
c12008#sh ip cef
Prefix                  Next Hop Interface
0.0.0.0/32              receive
10.0.10.1/32 receive
10.1.1.0/24             10.0.3.1          Serial6/0
10.0.3.0/30             attached           Serial6/0
10.0.3.0/32             receive
10.0.3.2/32             receive
10.0.3.3/32             receive
224.0.0.0/24            receive
255.255.255.255/32 receive
```

- Packets with next hop receive are sent to the RP for processing

12000 rACL Processing

- LC CPU always handles rACL processing
- Under attack, LC CPU utilization increases
- Impact depends on LC engine type

  E0/E1/E2: high CPU can impact routing and L2 traffic

  E2 w/ throttle ucode: high CPU → activates throttling, only precedence 6/7 traffic forwarded to RP

  E3: one of three queues dedicated for prec. 6/7 traffic, another for L2 keepalives

  E4/E4+: Eight queues, prec. 6/7 and L2 keepalives in dedicated queues

  E5: one of three queues dedicated for prec. 6/7 traffic, another for L2 keepalives

- rACL always improves resiliency to attack

rACL: Iterative Deployment

- Step 1: Identify protocols/ports used in the network with a classification ACL.
  - Permit any any for various protocols/ports
  - Get an understanding of what protocols communicate with the router
  - Permit any any log at the end can be used to identify any missed protocols
  - This should be slowly to ensure no protocols are missed
- Step 2: Review identified packets, begin to filter access to the GRP/PRP
  - Using list developed in step 1, permit only those protocols
  - Deny any any at the end → basic protection

rACL: Iterative Deployment

- Step 3: Restrict a macro range of source addresses

  Only permit your CIDR block in the source field

  eBGP peers are the exception: they may fall outside CIDR block

- Step 4: Narrow the rACL

  Increasingly limit the source addresses to known sources: management stations, NTP peers, AAA server, etc.

rACL: Iterative Deployment

- Step 5: Limit the destination addresses on the rACL

  Filter what interfaces are accessible to specific protocols

  Does the protocol access loopbacks only? Real interfaces?

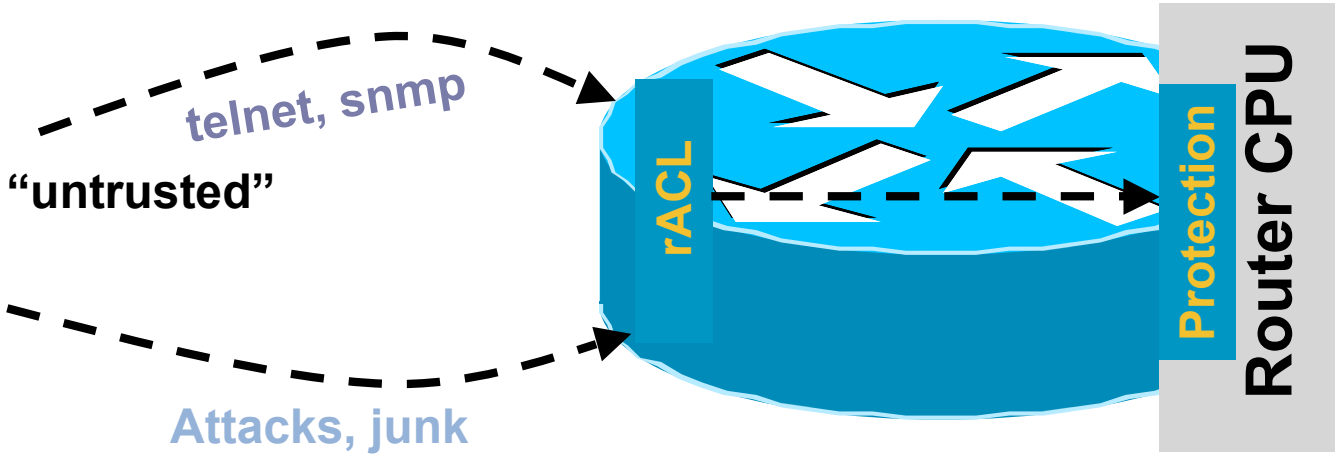
- Rinse, repeat

  Remember, start slow and open

  Gradually improve security over time

  If you try and start very secure, you are increasing your chance of dropping legitimate traffic

rACLs: Summary



- Contain the attack: compartmentalize
  - Protect the RP
- Widely deployed and highly effective
  - If you have platforms that support rACLs, start planning a deployment
  - rACL deployments can easily be migrated to control plane policing (next topic)

Router Hardening: Protecting the CPU Control Plane Policing (CoPP)

Control Plane Policing (CoPP) Command

- Introduced in:

  12000: 12.0(29)S (aggregate mode)

  12000: 12.0(30)S (distributed mode)

  6500/7600: 12.2(18)SXD1

  10720: 12.0(32)S

  Most other platforms: 12.2(18)S/12.3(4)T

```
Router(config)# control-plane [slot slot-number]
Router(config-cp)# service-policy input control-plane-policy
```

- Uses the Modular QoS CLI (MQC) syntax for QoS policy definition
- Dedicated control-plane “interface” for applying QoS policies—single point of application
- Platform specifics details: centralized vs. distributed vs. hardware
- Unlike rACL, CoPP handles data plane punts as well as control/management plane traffic

Deploying CoPP

- One option: mimic rACL behavior
  - CoPP is a superset of rACL
  - Apply rACL to a single class in CoPP
  - Same limitations as with rACL: permit/deny only
- Recommendation: develop multiple classes of control plane traffic
  - Apply appropriate rate to each
  - “Appropriate” will vary based on network, risk tolerance, risk assessment
  - Be careful what you rate-limit
- Flexible class definition allows extension of model
  - Fragments, TOS, ARP

Configuring CoPP

Four Required Steps:

1. Define ACLs

   Classify traffic

2. Define class-maps

   Setup class of traffic

3. Define policy-map

   Assign QOS policy action to class of traffic (police, drop)

4. Apply CoPP policy to control plane “interface”

Step 1: Define ACLs

Group IP Traffic Types into Different Classes

- Known Undesirable—traffic that is deemed “bad” or “malicious” to be denied access to the RP
- Critical—traffic crucial to the operation of the network
- Important—traffic necessary for day-to-day operations
- Normal—traffic expected but not essential for network operations
- Reactive Undesirable—traffic that is deemed “bad” or “malicious” to be denied access to the RP
- Catch-All—all other IP traffic destined to the RP that has not been identified
- Default—all remaining non-IP traffic destined to the RP that has not been identified

Step 1: Define ACLs

The Router IP Address for Control/Management Traffic Is 10.1.1.1

- Known Undesirable—ACL 120
- Critical—ACL 121
- Important—ACL 122
- Normal—ACL 123
- Reactive Undesirable—ACL 124
- Catch All—ACL 125
- Default—no ACL required

```
! KNOWN UNDESIRABLE – Traffic that should never touch the RP
access-list 120 permit tcp any any fragments
access-list 120 permit udp any any fragments
access-list 120 permit icmp any any fragments
access-list 120 permit ip any any fragments
access-list 120 permit udp any any eq 1434
```

Step 1: Define ACLs

The Router IP Address for Control/Management Traffic Is 10.1.1.1

- Known Undesirable—ACL 120
- Critical—ACL 121
- Important—ACL 122
- Normal—ACL 123
- Reactive Undesirable—ACL 124
- Catch All—ACL 125
- Default—no ACL required

```
! CRITICAL -- Defined as routing protocols
access-list 121 permit tcp host 10.1.1.2 eq bgp host 10.1.1.1 gt 1024
access-list 121 permit tcp host 10.1.1.2 gt 1024 host 10.1.1.1 eq bgp
access-list 121 permit tcp host 10.1.1.3 eq bgp host 10.1.1.1 gt 1024
access-list 121 permit tcp host 10.1.1.3 gt 1024 host 10.1.1.1 eq bgp
access-list 121 permit ospf any host 224.0.0.5
access-list 121 permit ospf any host 224.0.0.6
access-list 121 permit ospf any any
```

Step 1: Define ACLs

The Router IP Address for Control/Management Traffic Is 10.1.1.1

- Known Undesirable—ACL 120
- Critical—ACL 121
- Important—ACL 122
- Normal—ACL 123
- Reactive Undesirable—ACL 124
- Catch All—ACL 125
- Default—no ACL required

```
! IMPORTANT -- Defined as traffic required to manage the router
access-list 122 permit tcp 10.2.1.0 0.0.0.255 eq 22 host 10.1.1.1 established
access-list 122 permit tcp 10.2.1.0 0.0.0.255 host 10.1.1.1 eq 22
access-list 122 permit tcp 10.2.1.0 0.0.0.255 host 10.1.1.1 eq telnet
access-list 122 permit udp host 10.2.2.1 eq tftp host 10.1.1.1
access-list 122 permit udp host 10.2.2.2 host 10.1.1.1 eq snmp
access-list 122 permit udp host 10.2.2.3 host 10.1.1.1 eq ntp
```

Step 1: Define ACLs

The Router IP Address for Control/Management Traffic Is 10.1.1.1

- Known Undesirable—ACL 120
- Critical—ACL 121
- Important—ACL 122
- Normal—ACL 123

Reactive Undesirable—ACL 124

Catch All—ACL 125

Default—no ACL required

```
! NORMAL -- Defined as other traffic destined to the router to track and limit
access-list 123 permit icmp any any ttl-exceeded
access-list 123 permit icmp any any port-unreachable
access-list 123 permit icmp any any echo-reply
access-list 123 permit icmp any any echo
access-list 123 permit icmp any any packet-too-big
```

Step 1: Define ACLs

The Router IP Address for Control/Management Traffic Is 10.1.1.1

- Known Undesirable—ACL 120
- Critical—ACL 121
- Important—ACL 122
- Normal—ACL 123

Reactive Undesirable—ACL 124

Catch All—ACL 125

Default—no ACL required

```
! REACTIVE UNDESIRABLE – Traffic that should never touch the RP
access-list 124 permit tcp any eq 22 host 10.1.1.1 established
access-list 124 permit tcp any  host 10.1.1.1 eq 22
access-list 124 permit tcp any  host 10.1.1.1 eq telnet
access-list 124 permit udp any eq tftp host 10.1.1.1
access-list 124 permit udp any  host 10.1.1.1 eq snmp
access-list 124 permit udp any host 10.1.1.1 eq ntp
```

Step 1: Define ACLs

The Router IP Address for Control/Management Traffic Is 10.1.1.1

- Known Undesirable—ACL 120
- Critical—ACL 121
- Important—ACL 122
- Normal—ACL 123

Reactive Undesirable—ACL 124

Catch All—ACL 125

Default—no ACL required

```
! CATCH ALL -- Defined as other IP traffic destined to the router
access-list 125 permit ip any any
```

Step 2: Define Class-Maps

- Create class-maps to complete the traffic-classification process

  Use the access-lists defined on the previous slides to specify which IP packets belong in which classes

- Class-maps permit multiple match criteria, and nested class-maps

  match-any requires that packets meet only one “match” criteria to be considered “in the class”

  match-all requires that packets meet all of the “match” criteria to be considered “in the class”

- A “match-all” classification scheme with a simple, single-match criteria will satisfy initial deployments
- Traffic destined to the “undesirable” class should follow a “match-any” classification scheme

Step 2: Define Class-Maps

```
! Define a class for each “type” of traffic and associate the
! appropriate ACL
class-map match-all CoPP-known-undesirable
  match access-group 120
class-map match-all CoPP-critical
  match access-group 121
class-map match-all CoPP-important
  match access-group 122
class-map match-any CoPP-normal
  match access-group 123
class-map match-any CoPP-reactive-undesirable
  match access-group 124
class-map match-any CoPP-catch-all
  match access-group 125
```

Step 3: Define Policy-Map

- Class-maps defined in Step 2 need to be “enforced” by using a policy-map to specify appropriate service policies for each traffic class
- For example:

  For undesirable traffic types, all actions are unconditionally “drop” regardless of rate

  For critical, important, and normal traffic types, all actions are “transmit” to start out

  For catch-all traffic, rate-limit the amount of traffic permitted above a certain bps

  Note: all traffic that fails to meet the matching criteria belongs to the default traffic class, which is user configurable, but cannot be deleted

Step 3: Define Policy-Map

```
! Example “Baseline” service policy for each traffic classification
policy-map CoPP
 class CoPP-known-undesirable
    police 8000 1000 4470 conform-action drop  exceed-action drop
  class CoPP-critical
    police 5000000 2500 4470 conform-action transmit  exceed-action transmit
class CoPP-important
    police 1000000 1000 4470 conform-action transmit  exceed-action transmit
class CoPP-normal
    police 1000000 1000 4470 conform-action transmit  exceed-action drop
class CoPP-reactive-undesirable
    police 8000 1000 4470 conform-action drop  exceed-action drop
class CoPP-Catch-All
    police 1000000 1000 4470 conform-action transmit  exceed-action drop
class class-default
    police 8000 1000 4470 conform-action transmit exceed-action transmit
```

Step 4: Apply Policy to “Interface”

- Apply the policy-map created in Step 3 to the “control plane”
- The new global configuration CLI “control-plane” command is used to enter “control-plane configuration mode”
- Once in control-plane configuration mode, attach the service policy to the control plane in the “input” direction

  Input—applies the specified service policy to packets that are entering the control plane

Step 4: Apply Policy to “Interface”

- Centralized

```
Router(config)# control-plane
Router(config-cp)# service-policy [input | output] <policy-map-name>
```

- Distributed

```
Router(config)#control-plane slot <n>
Router(config-cp)#service-policy input <policy-map-name>
```

```
! Example
! This applies the policy-map to the Control Plane
control-plane
  service-policy input CoPP
```

Monitoring CoPP

- "show access-list" displays hit counts on a per ACL entry (ACE) basis
  - The presence of hits indicates flows for that data type to the control plane as expected
  - Large numbers of packets or an unusually rapid rate increase in packets processed may be suspicious and should be investigated
  - Lack of packets may also indicate unusual behavior or that a rule may need to be rewritten
- "show policy-map control-plane" is invaluable for reviewing and tuning site-specific policies and troubleshooting CoPP
  - Displays dynamic information about number of packets (and bytes) conforming or exceeding each policy definition
  - Useful for ensuring that appropriate traffic types and rates are reaching the route processor
- Use SNMP queries to automate the process of reviewing service-policy transmit and drop rates
  - The Cisco QoS MIB (CISCO-CLASS-BASED-QOS-MIB) provides the primary mechanisms for MQC-based policy monitoring via SNMP

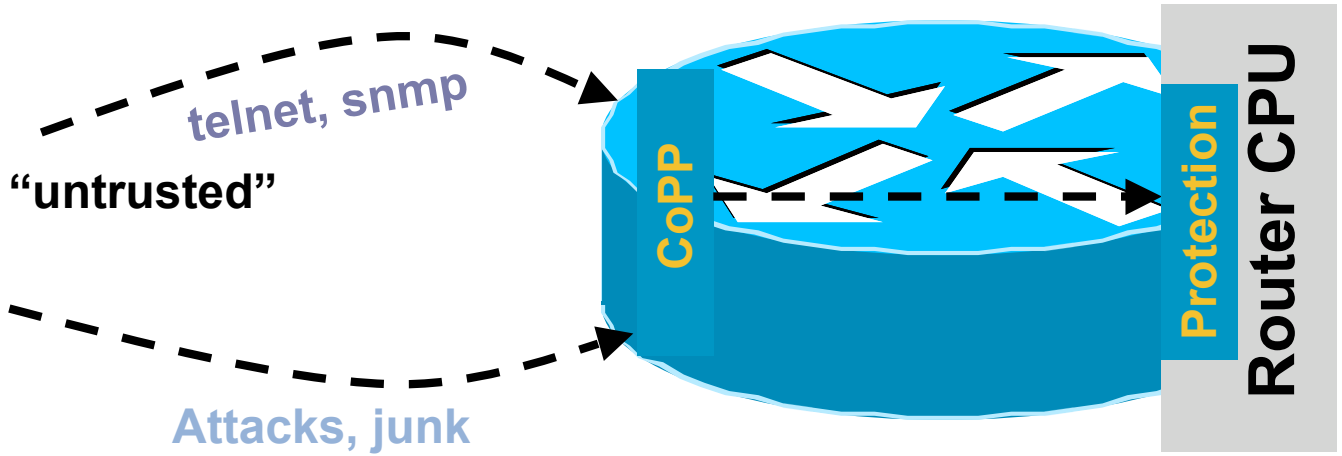
Show Policy-Map Command

```
Router#show policy-map control-plane input
 Control Plane
  Service-policy input: CoPP
    Class-map: Critical (match-all)
      16 packets, 2138 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 121
      police:
          cir 5000000 bps, bc 2500 bytes
        conformed 16 packets, 2138 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          transmit
        conformed 0 bps, exceed 0 bps
…
    Class-map: class-default (match-any)
      250 packets, 84250 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      police:
          cir 8000 bps, bc 1000 bytes
        conformed 41 packets, 5232 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          transmit
        conformed 0 bps, exceed 0 bps
Router#
```

CoPP and SNMP

```
! Using SNMP...
[Linux]$ snmpwalk -m all 10.82.69.157 cisco .1.3.6.1.4.1.9.9.166.1.15.1.1.2
enterprises.cisco.ciscoMgmt.ciscoCBQosMIB.ciscoCBQosMIBObjects.cbQosClassMapStat
    s.cbQosCMStatsTable.cbQosCMStatsEntry.cbQosCMPrePolicyPkt.1035.1037 =
    Counter32: 3924
[Linux]$ snmpwalk -m all 10.82.69.157 cisco .1.3.6.1.4.1.9.9.166.1.15.1.1.5
enterprises.cisco.ciscoMgmt.ciscoCBQosMIB.ciscoCBQosMIBObjects.cbQosClassMapStat
    s.cbQosCMStatsTable.cbQosCMStatsEntry.cbQosCMPrePolicyByte.1035.1037 =
    Counter32: 344523
[Linux]$
! Via CLI...
Router#sh policy-map control-plane input
 Control Plane
  Service-policy input: CoPP
    Class-map: class-default (match-any)
      3924 packets, 344523 bytes
      5 minute offered rate 1000 bps, drop rate 0 bps
      Match: any
      police:
          cir 8000 bps, bc 1000 bytes
        conformed 3875 packets, 336178 bytes; actions:
          transmit
        exceeded 49 packets, 8345 bytes; actions:
          transmit
        conformed 1000 bps, exceed 0 bps
```

Control Plane Policing



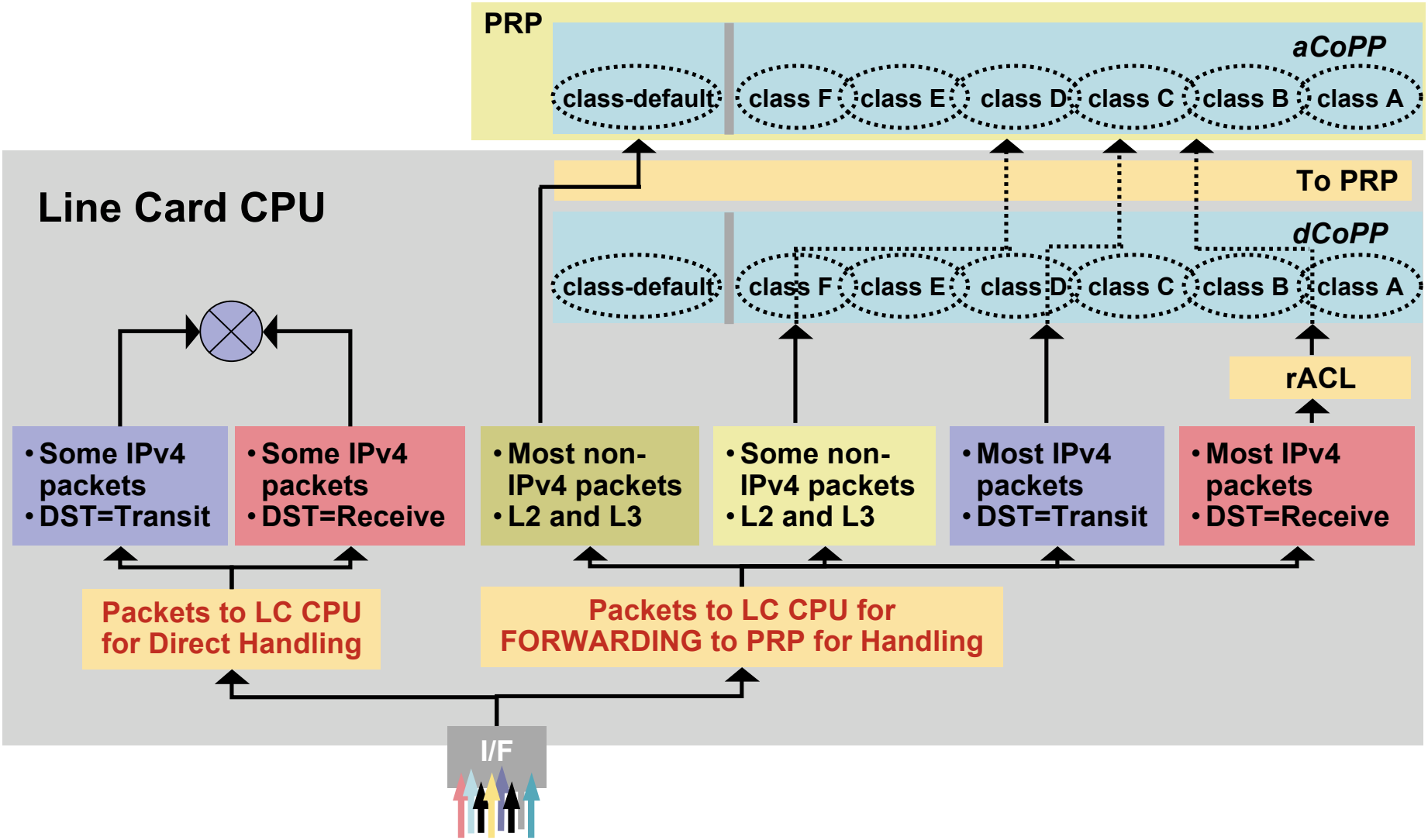
- Superset of rACL: start planning your migrations
- Provides a cross-platform methodology for protecting the control plane
  - Consistent "show" command and MIB support
- Granular: permit, deny and rate-limit
- Platform specifics details: centralized vs. distributed vs. hardware

Router Hardening: Protecting the CPU Hardware-Specific Issues

Hardware Specific Issues

- There are specific restrictions and caveats associated with CoPP imposed by hardware
- We will briefly discuss the 12000 and 6500/7600
- See the appendix section for some more details
- This highlights the fact that you must fully understand how your router handles packets and this is a whole topic itself

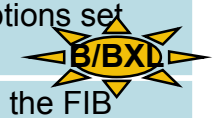
12000 Hardware Specific Issues



6500/7600 Hardware-Based Rate Limiters

Unicast Rate Limiters	
CEF Receive	Traffic destined to the router
CEF Glean	ARP packets
CEF No Route	Packets with not route in the FIB
IP Errors	Packets with IP checksum or length errors
ICMP Redirect	Packets that require ICMP redirects
ICMP No Route	ICMP unreachables for unroutable packets
ICMP ACL Drop	ICMP unreachables for admin deny packets
RPF Failure	Packets that fail uRPF check
L3 Security	CBAC, Auth-Proxy, and IPSec traffic
ACL Input	NAT, TCP Int, Reflexive ACLs, Log on ACLs
ACL Output	NAT, TCP Int, Reflexive ACLs, Log on ACLs
VACL Logging	CLI notification of VACL denied packets
IP Options	Unicast traffic with IP Options set
Capture	Used with optimized ACL logging

Multicast Rate Limiters	
Multicast FIB-Miss	Packets with no mroute in the FIB
IGMP	IGMP packets
Partial Shortcut	Partial shortcut entries
Directly Connected	Local multicast on connected interface
IP Options	Multicast traffic with IP Options set
V6 Directly Connect	Packets with no mroute in the FIB
V6*, G M Bridge	IGMP packets
V6*, G Bridge	Partial shortcut entries
V6 S, G Bridge	Partial shortcut entries
V6 Route Control	Partial shortcut entries
V6 Default Route	Multicast traffic with IP Options set
V6 Second Drop	Multicast traffic with IP Options set



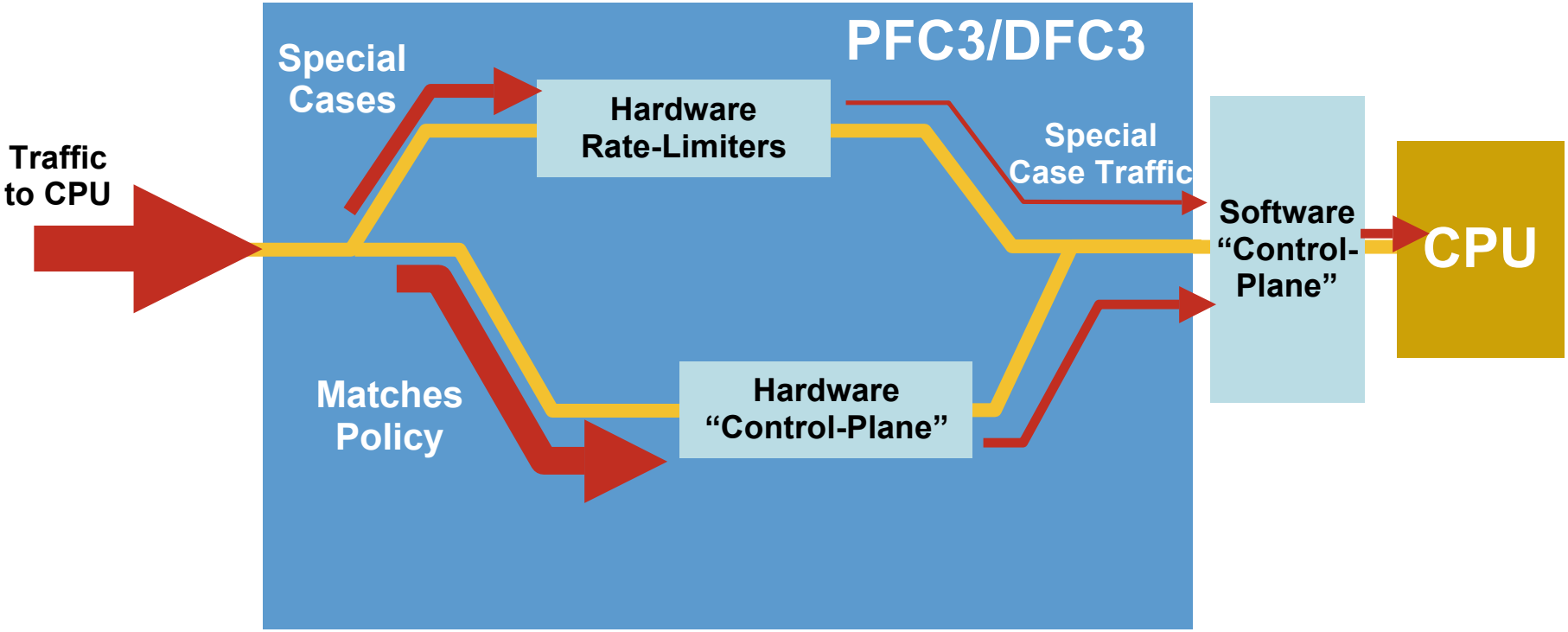
Shared Across the 10 Hardware Revocation Lists

Layer 2 Rate Limiters	
L2PT	L2PT encapsulation/decapsulation
PDU	Layer 2 PDUs

General Rate Limiters	
MTU Failure	Packets requiring fragmentation
TTL Failure	Packets with TTL<=1

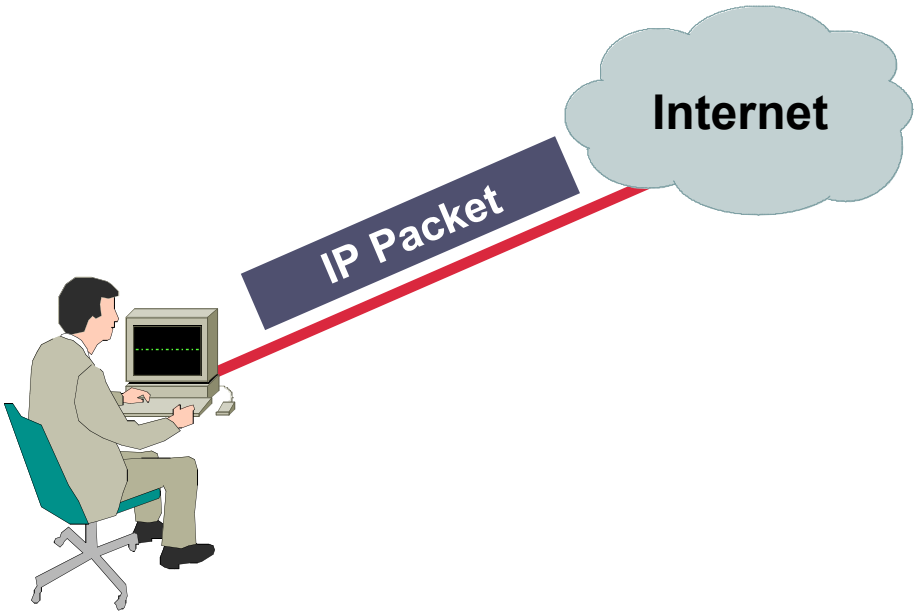
Control Plane Policing
6500/7600 with CPU Rate Limiter

Special-Case Rate Limiters Override Hardware Control Plane Policing!



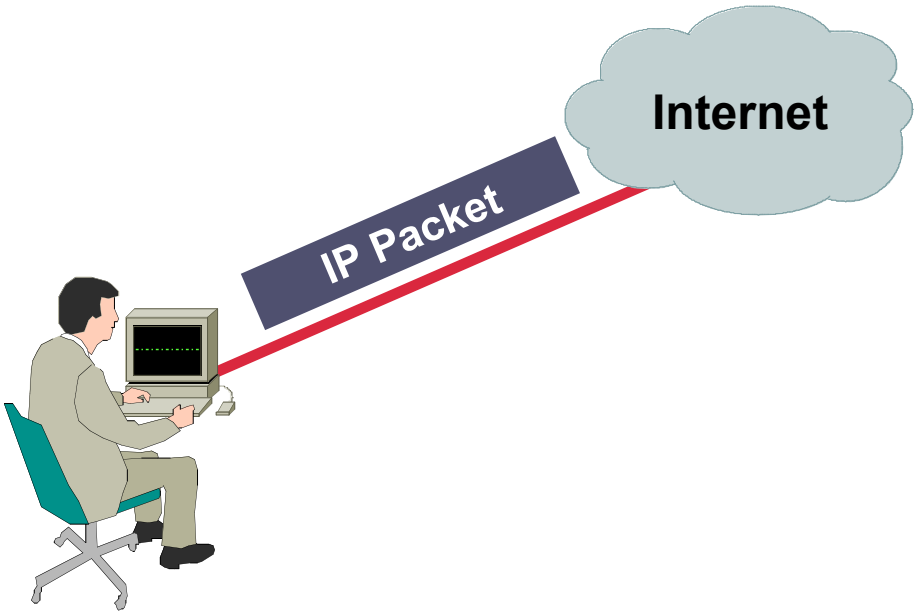
Network Hardening

It Is All About the Packet



- Once a packet gets into the Internet, someone, somewhere has to do one of two things:
  - Deliver the packet
  - Drop the packet

It Is All About the Packet



- In the context of an attack, the question is who and where will that drop that packet

Network Hardening

- In the context of denial of service attacks, if the packet makes it to the router, it's already too late
  - CoPP and rACL help dramatically, but they do not solve the problem
  - The unwanted packets must be dropped on ingress into your network
- Three methods:
  - Infrastructure ACL
  - Core Hiding
  - RFC2547 (MPLS) VPN

Network Hardening: Infrastructure ACL (iACL)

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

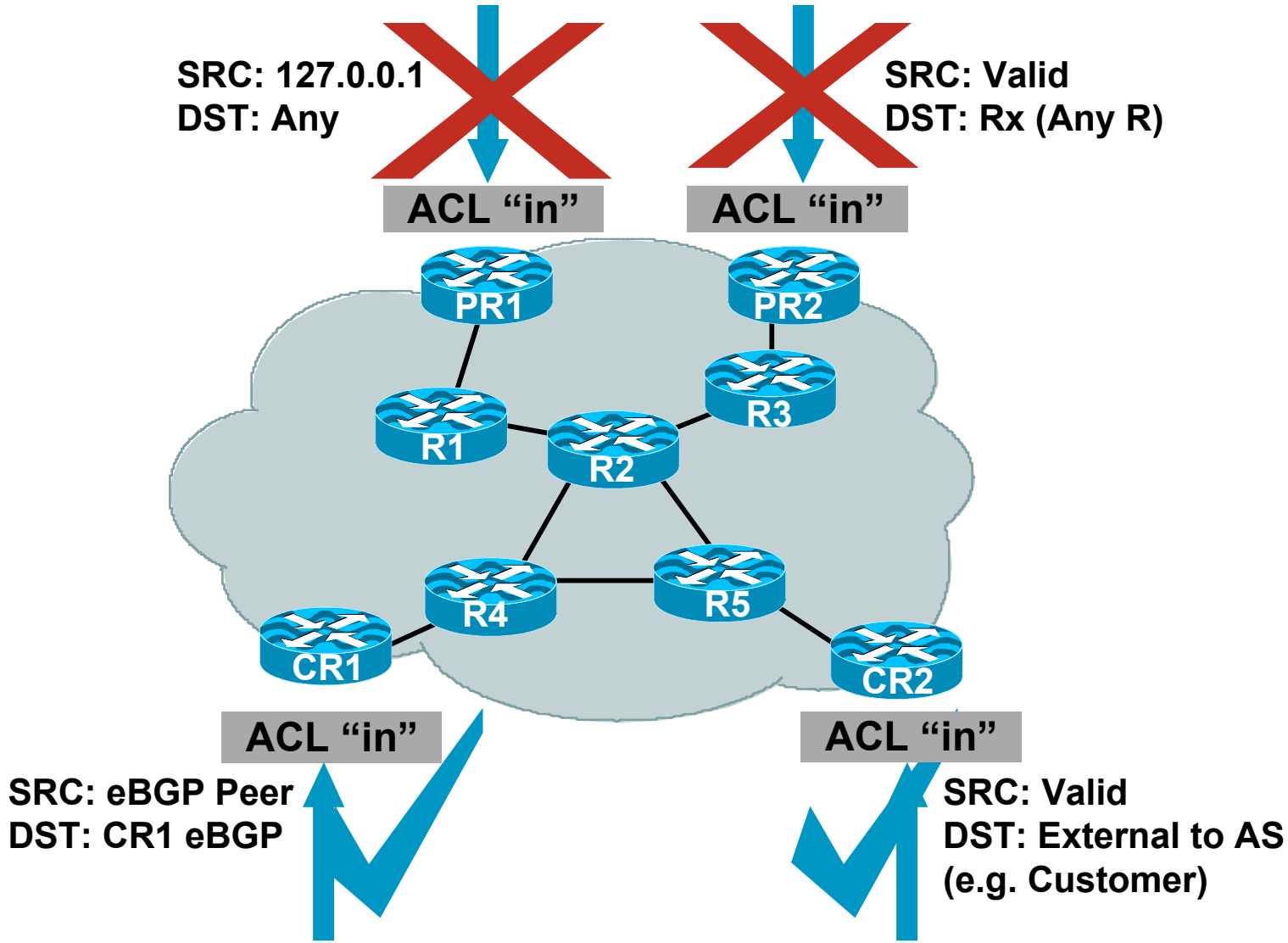
Infrastructure ACLs

- 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



Other iACL Possibilities

Edge QoS Enforcement

- Control what traffic is “important” in your network
- Philosophical debate for some
- 6/7 is easy

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 address blocks
- Deny all others to infrastructure address 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 to individual hosts if possible

Example: Infrastructure ACL

```
! Deny our internal space as a source of external packets
access-list 101 deny ip our_CIDR_block any
! Deny src addresses of 0.0.0.0 and 127/8
access-list 101 deny ip host 0.0.0.0 any
access-list 101 deny ip 127.0.0.0 0.255.255.255 any
! Deny RFC1918 space from entering AS
access-list 101 deny ip 10.0.0.0 0.255.255.255 any
access-list 101 deny ip 172.16.0.0 0.0.15.255 any
access-list 101 deny ip 192.168.0.0 0.0.255.255 any
!Permit eBGP from outside out network
access-list 101 permit tcp host peerA host peerB eq 179
access-list 101 permit tcp host peerA eq 179 host peerB
! Deny all other access to infrastructure
access-list 101 deny ip any core_CIDR_block
! Permit all data plane traffic
access-list 101 permit ip any any
```

Infrastructure ACL Summary

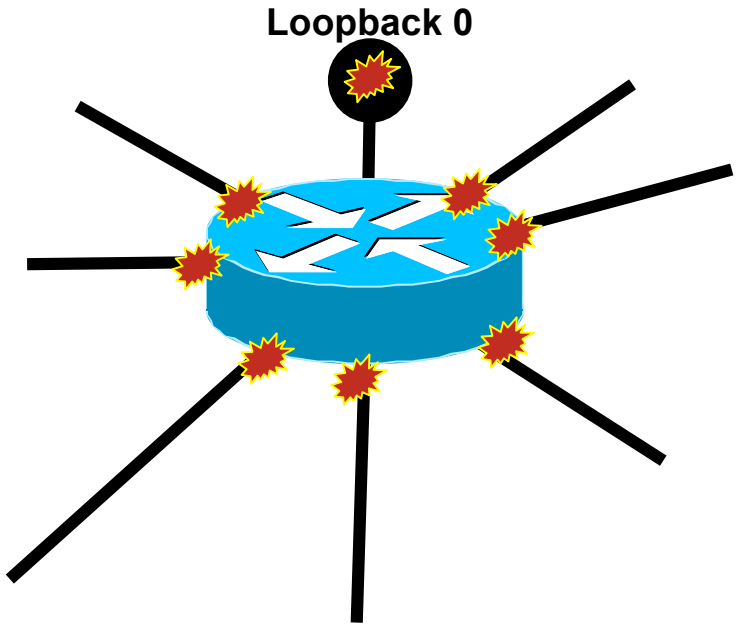
- Infrastructure ACLs are very effective at protecting the network if properly and universally deployed
- Infrastructure ACLs also have a few weaknesses
  - –Hardware restrictions associated with deploying ACLs or the ACEs required in iACL may prevent deployment
  - –Operational overhead in maintaining and deploying iACL
    - •Collisions with customer ACLs difficult to manage

Network Hardening: Core Hiding

Today's Principle—Reachability

- Today, the common practice is to insure there is reachability to all the links going into a device on the backbone

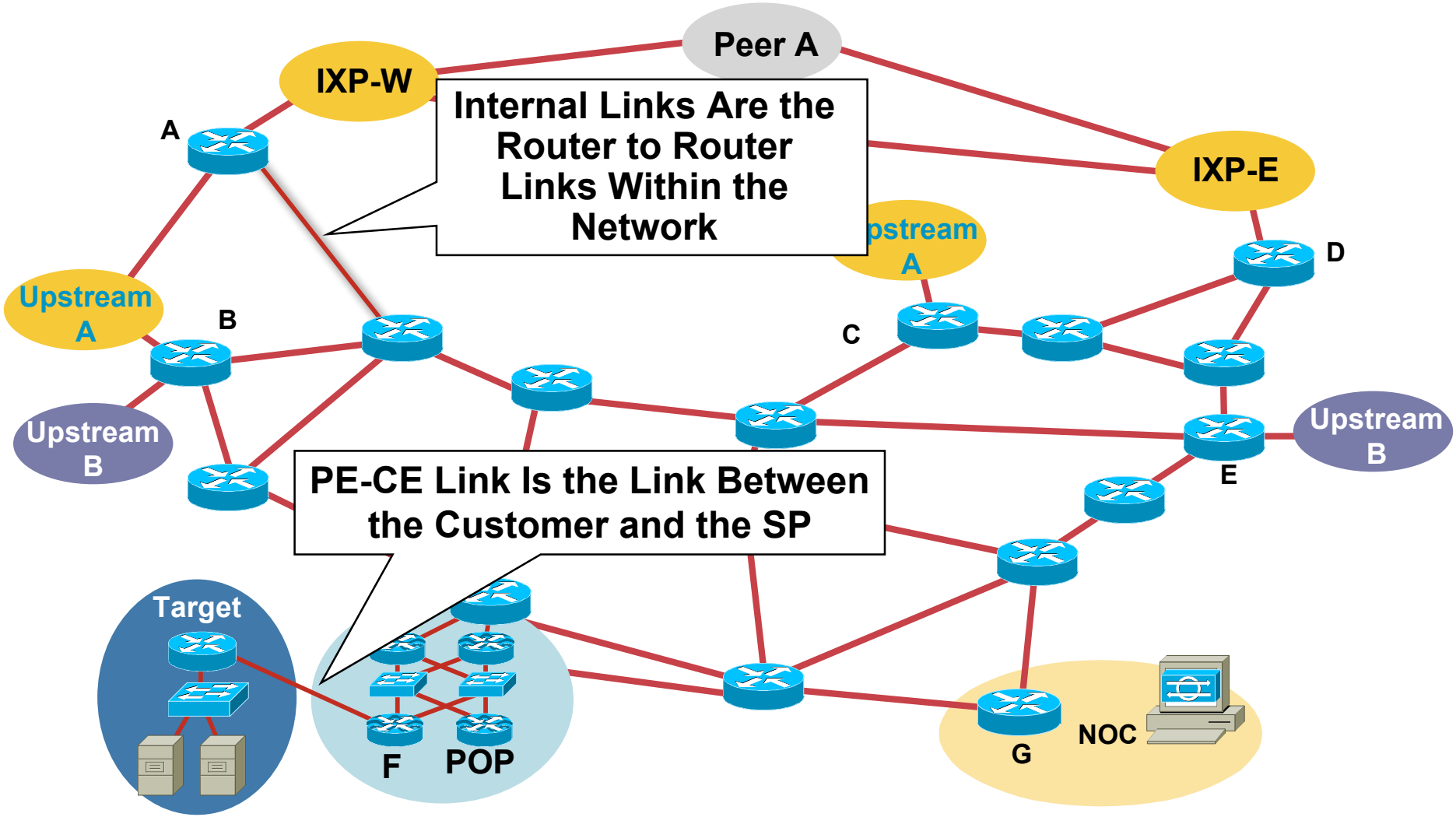
  Links are usually reachable from the entire Internet, pingable, and targetable



Network Hardening: Core Hiding

- Three different types of addresses reside on routers:
  - Network to customer external links (PE-CE Link)
  - Router to router internal links
  - Router loopbacks
- Use routing tricks to deny reachability to these IP addresses
  - What if the best route on the ingress router to these IP addresses was Null0?

Link Types



Limit Reachability to PE-CE IP Addresses

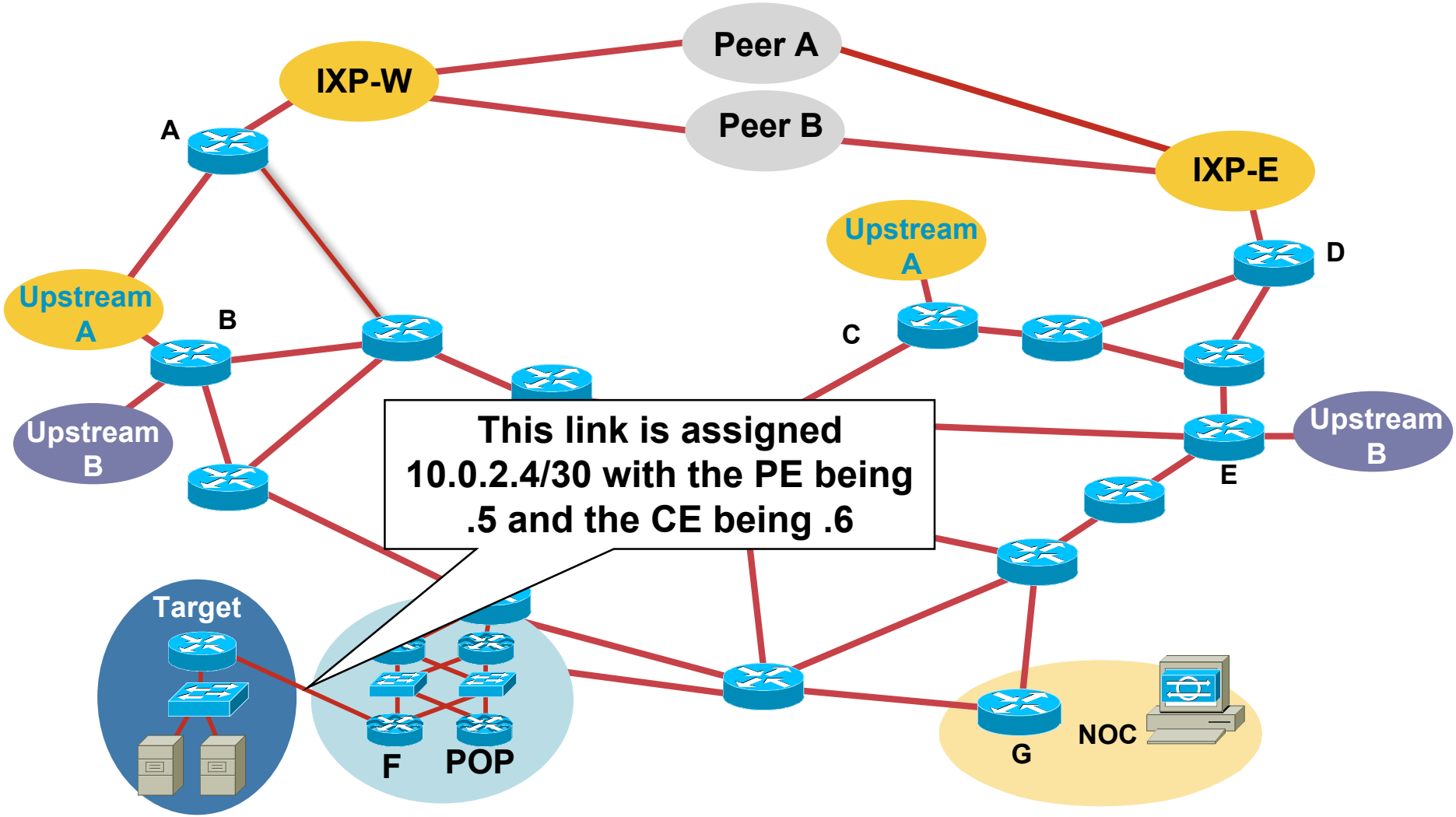
- Most router attacks target IPs obtained from a traceroute
- Remove the ability to reach customer's PE-CE link networks from the Internet

  No one needs to reach the PE-CE link besides the adjacent routers

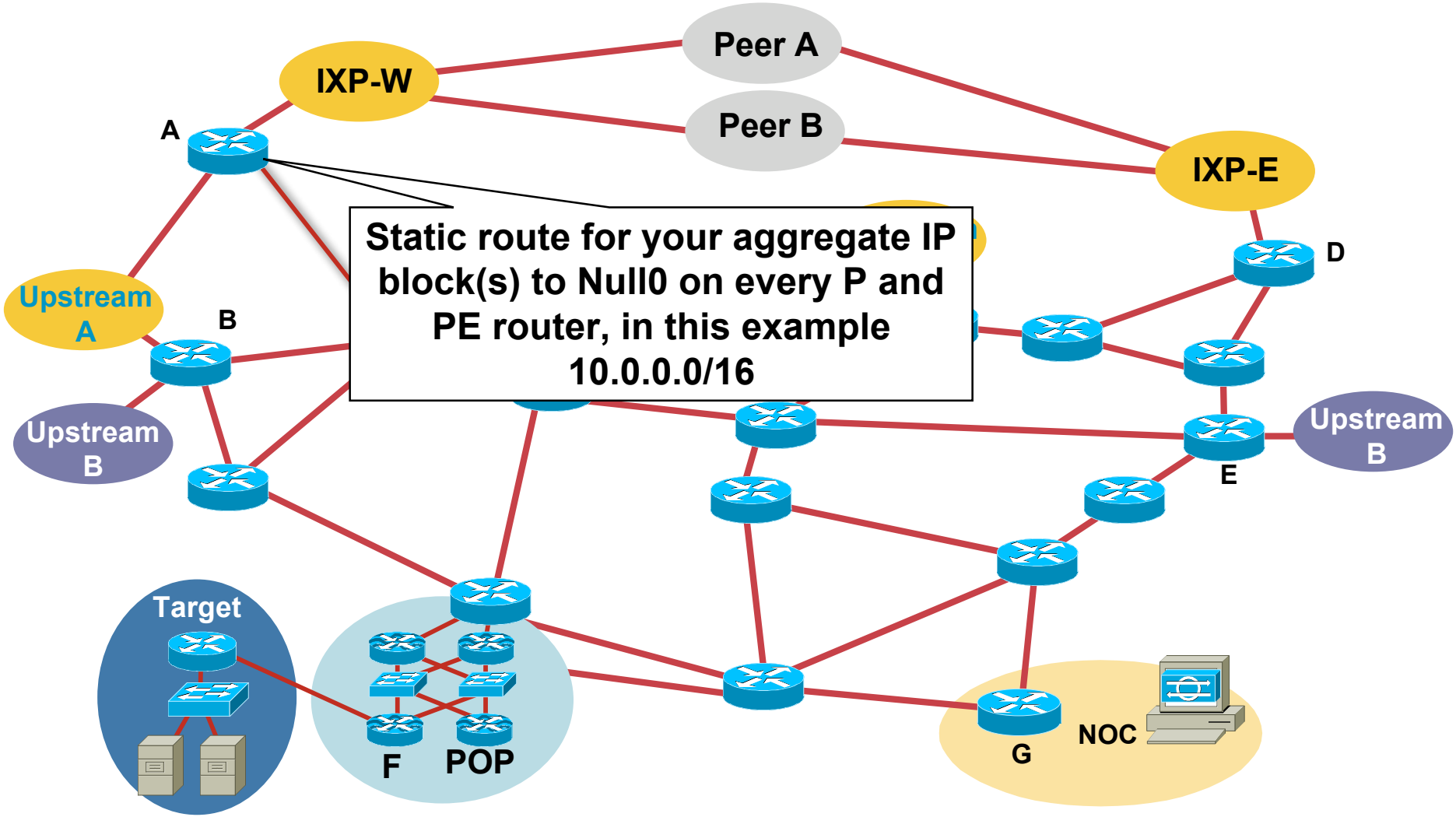
  Exceptions include:

  NAT, VoIP, IPSec, GRE, etc.

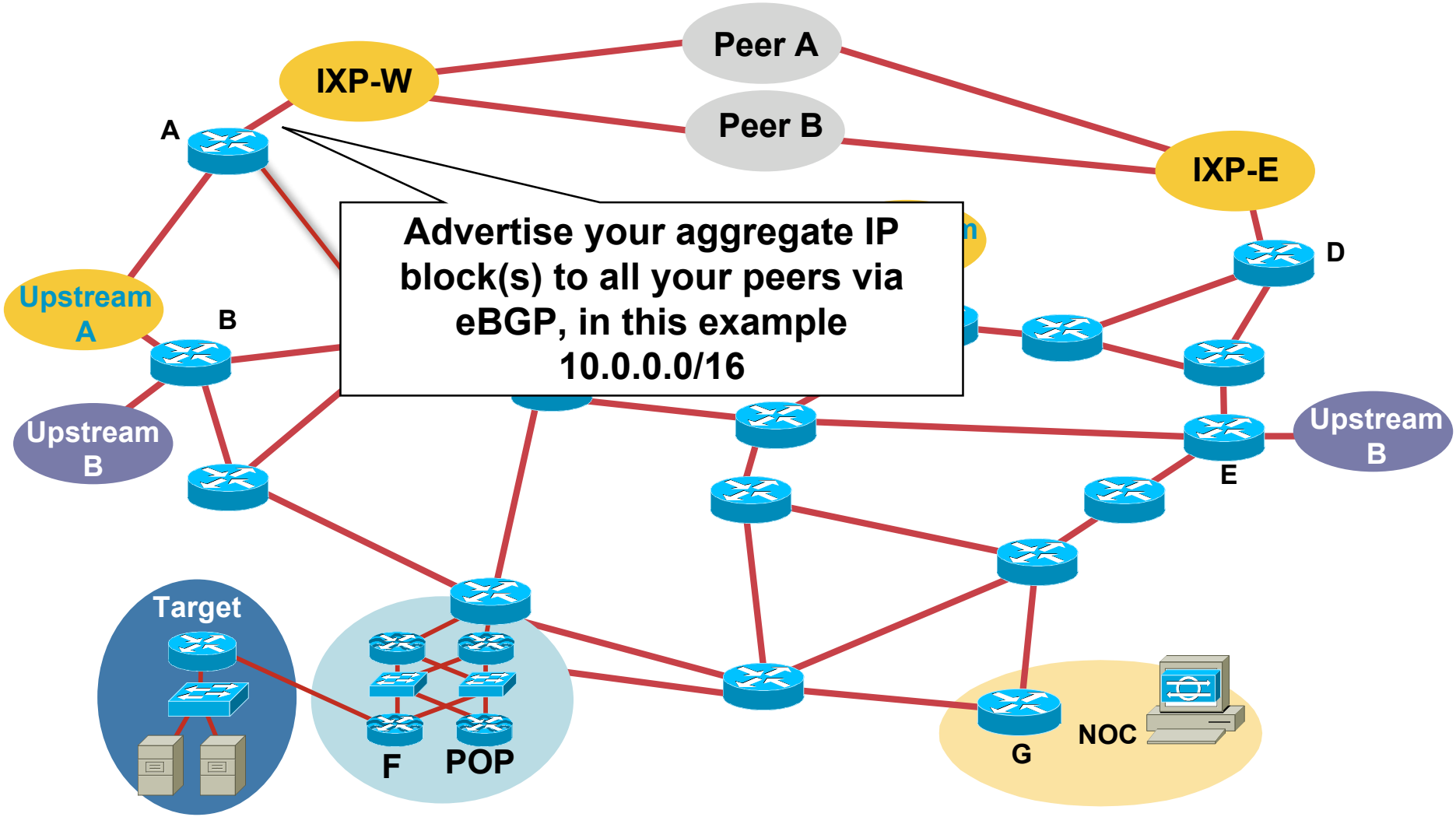
Core Hiding: PE-CE Links



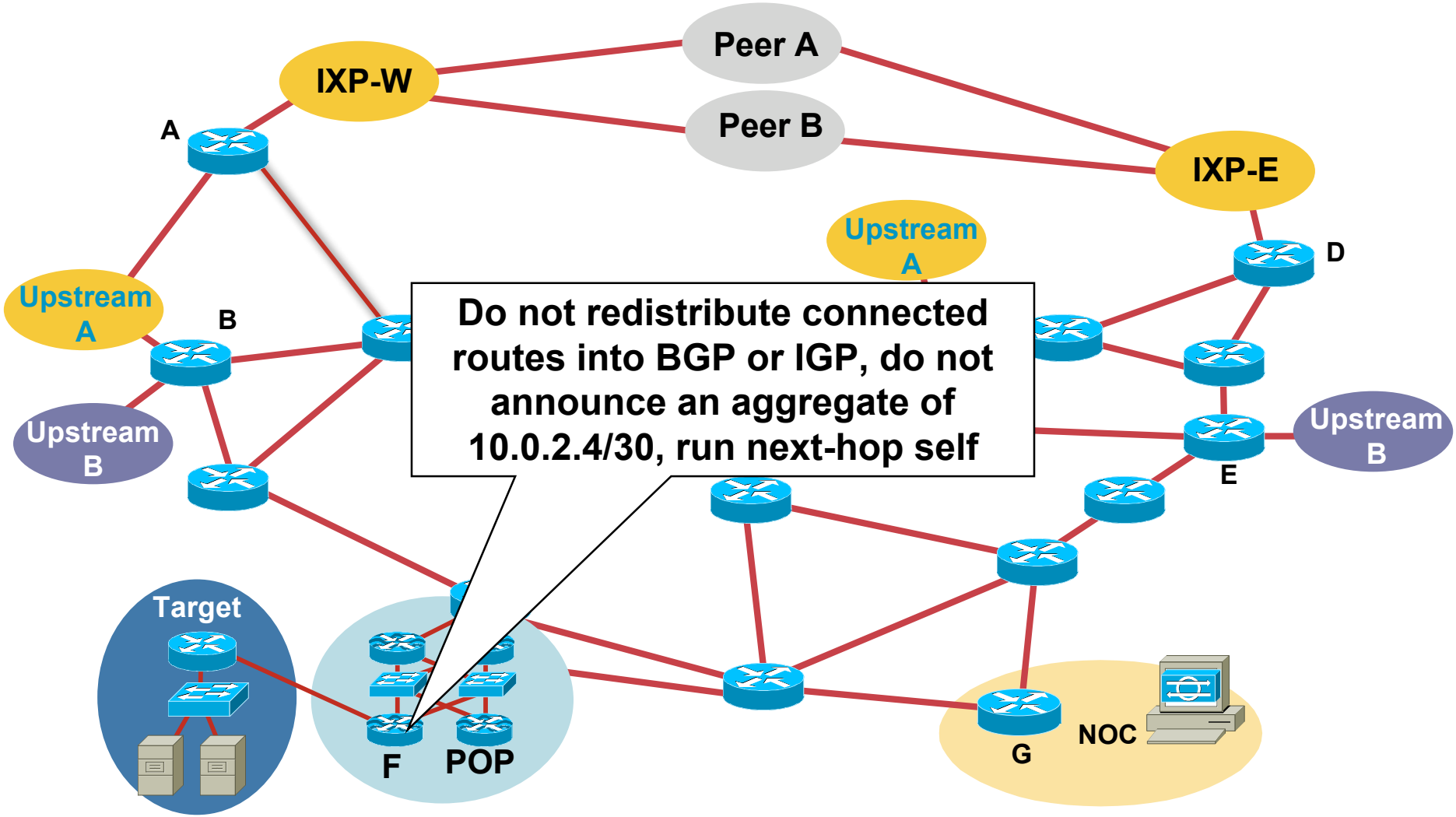
Core Hiding: PE-CE Links



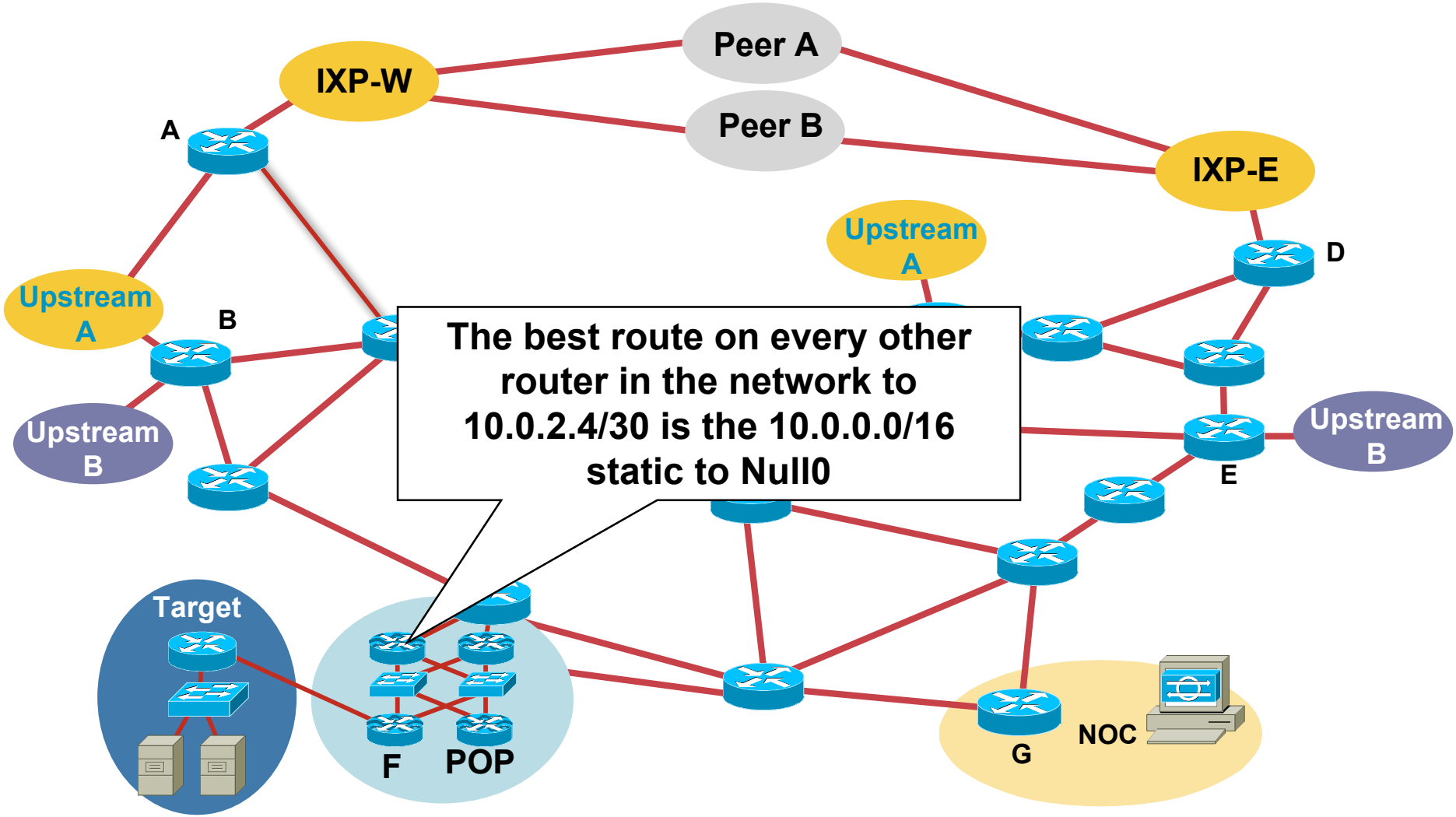
Core Hiding: PE-CE Links



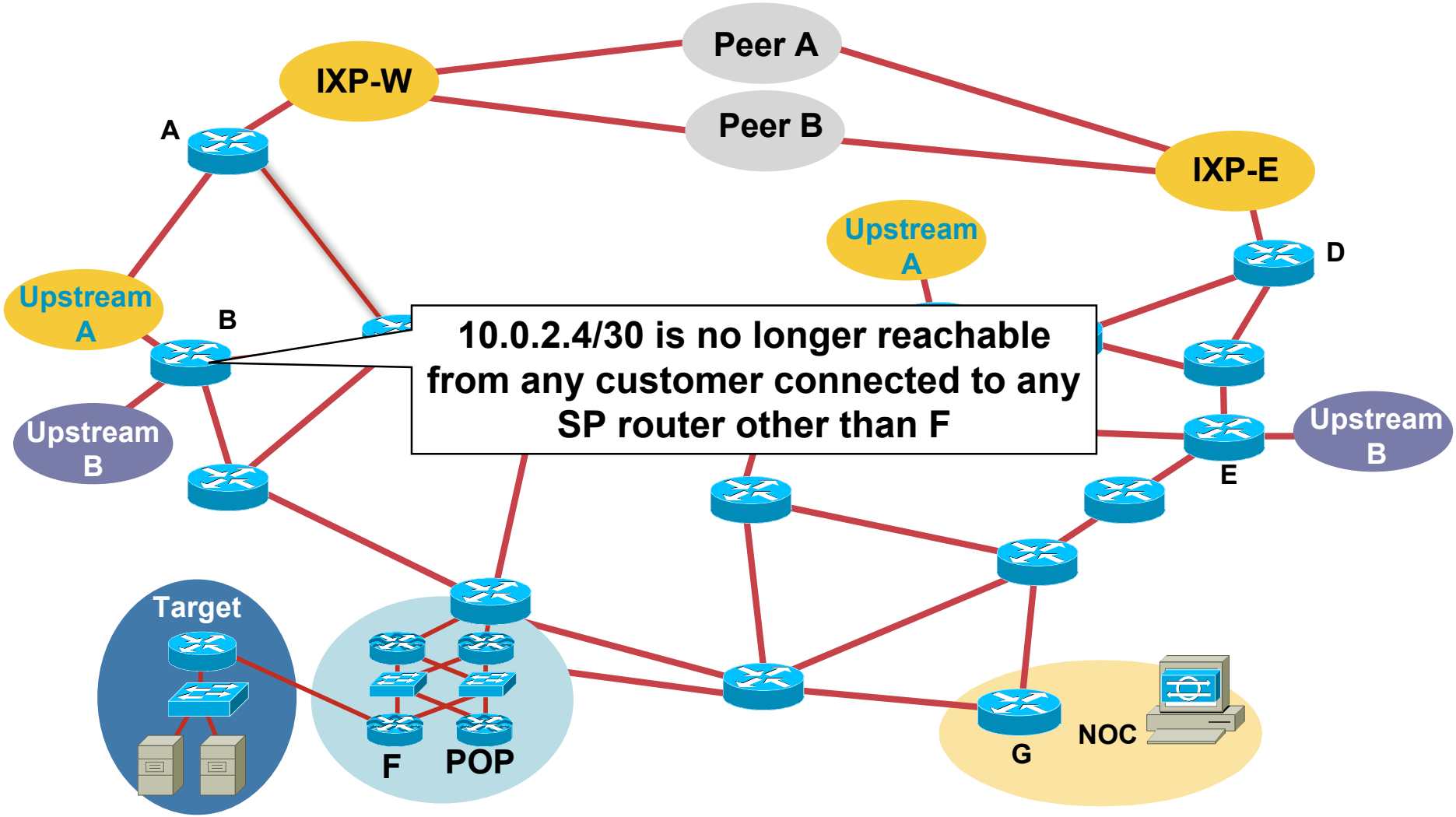
Core Hiding: PE-CE Links



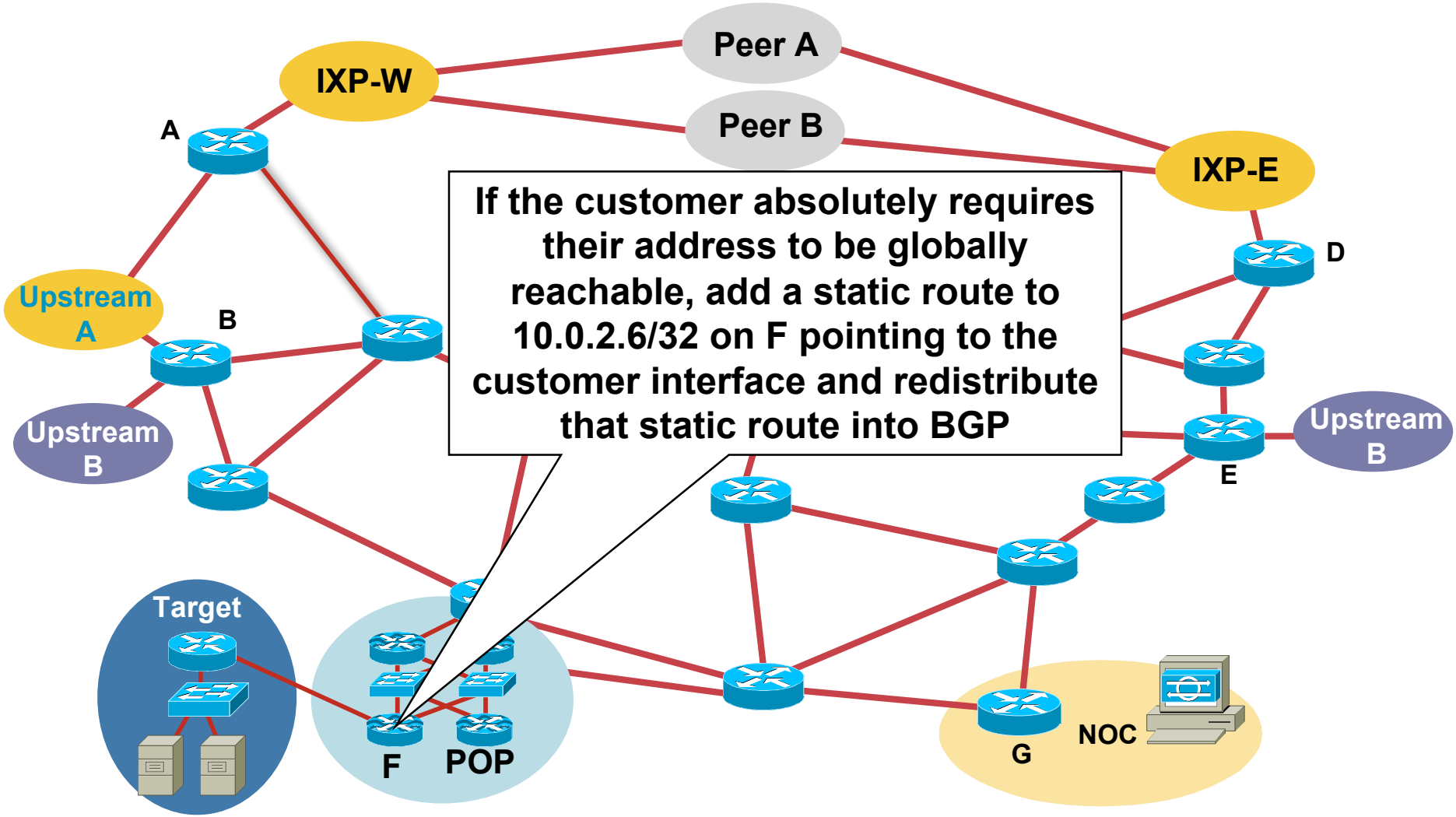
Core Hiding: PE-CE Links



Core Hiding: PE-CE Links



Core Hiding: PE-CE Links



Core Hiding: PE-CE Links

Caveats

- Only works if you carry all customer PE-CE links as /30 or /31 in iBGP
  - –Plus you can eliminate 10,000s of routes from your table
- If you aggregate and carry PE-CE links as a per-PE aggregate, must consider alternatives
  - –Stop announcing aggregate and carry /32 exceptions, the downside is you have to carry more routes
- Corner case of an attack ingressing and egressing on the same SP router
  - –Does not add 100% security
  - –Remember—it is all about adding hurdles

Core Hiding: PE-CE Links

Implications

- Traceroute through the router not impacted
- Any packets destined to the (non-directly connected) infrastructure breaks
  - Can change the way you troubleshoot
    - PING
    - Traceroute

Limit Reachability to Internal Link IP Addresses

- Most router attacks target IPs obtained from a traceroute
- Remove the ability to reach router to router networks from the Internet

  No one needs to reach the these networks besides the adjacent routers

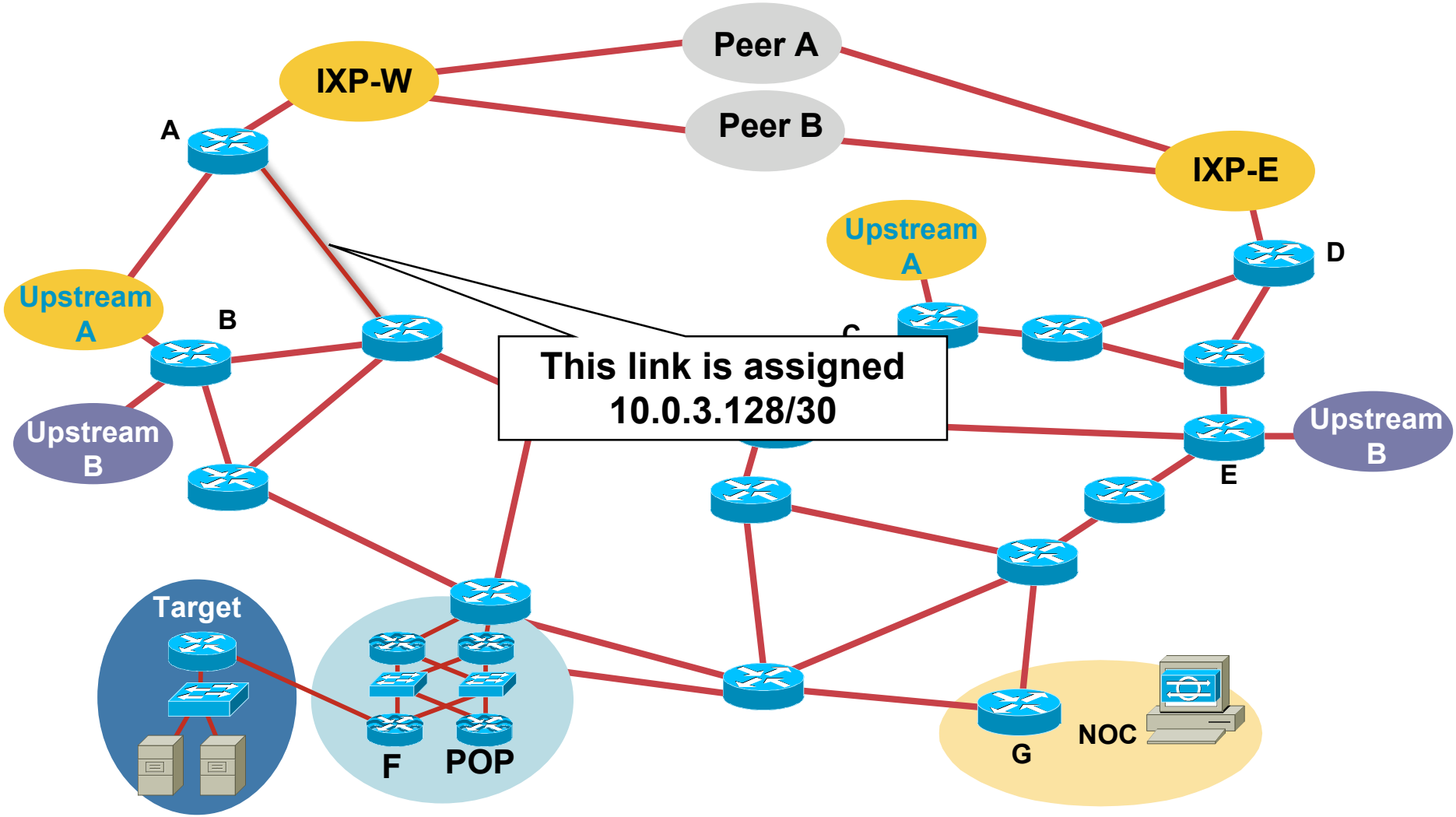
Routing as a Security Technique

- Utilize IS-IS fast convergence technique
- Command was intended to decrease convergence time by removing the IP network of the link running IS-IS from the IS-IS database

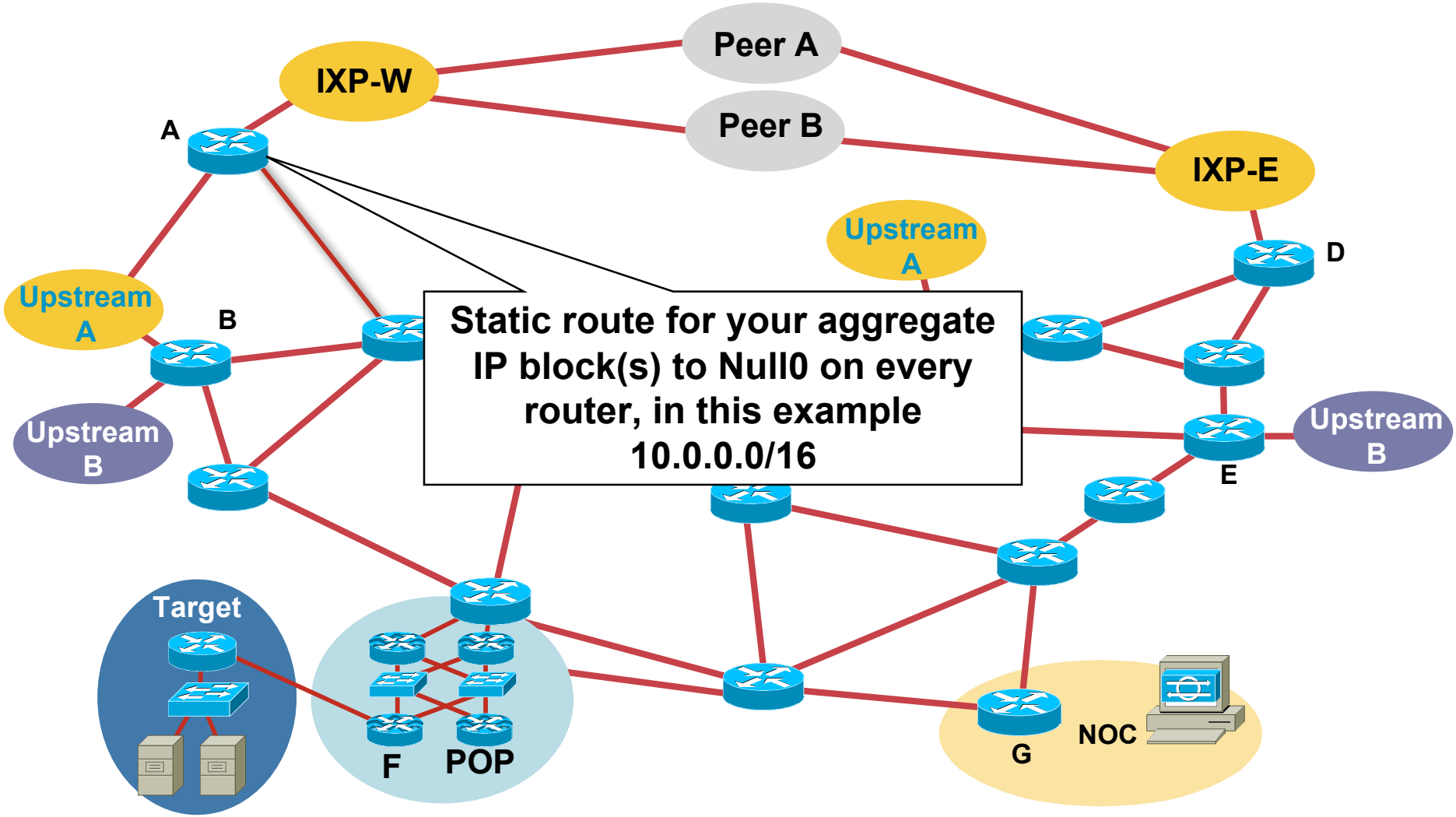
  Use the interface command “no isis advertise prefix” to remove that specific IP address from the IS-IS database

  Use the global IS-IS configuration command “advertise passive-only” to remove all interface IP addresses from the IS-IS database
- Can be overloaded and used as a security technique

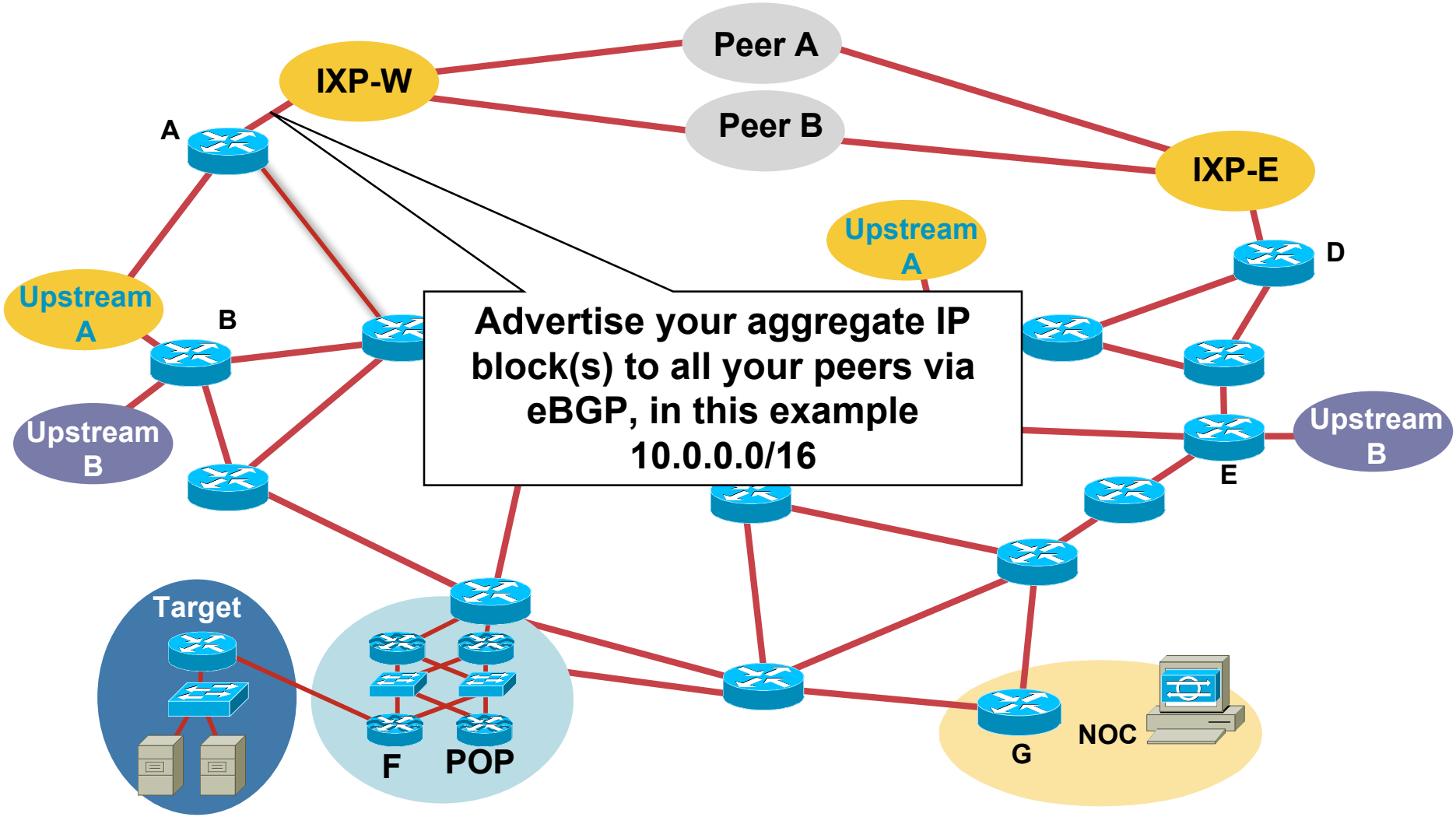
Core Hiding: Internal Links



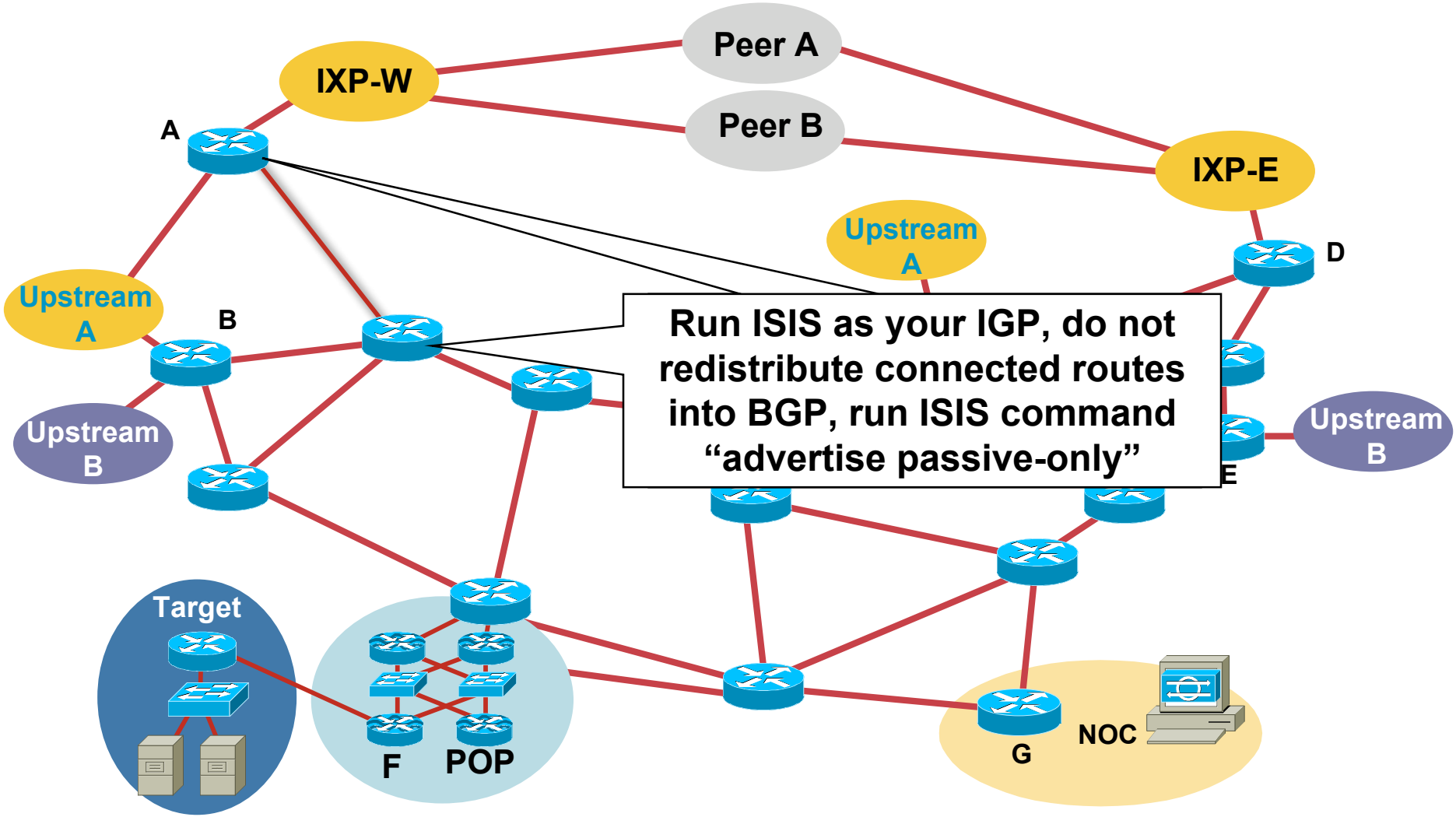
Core Hiding: Internal Links



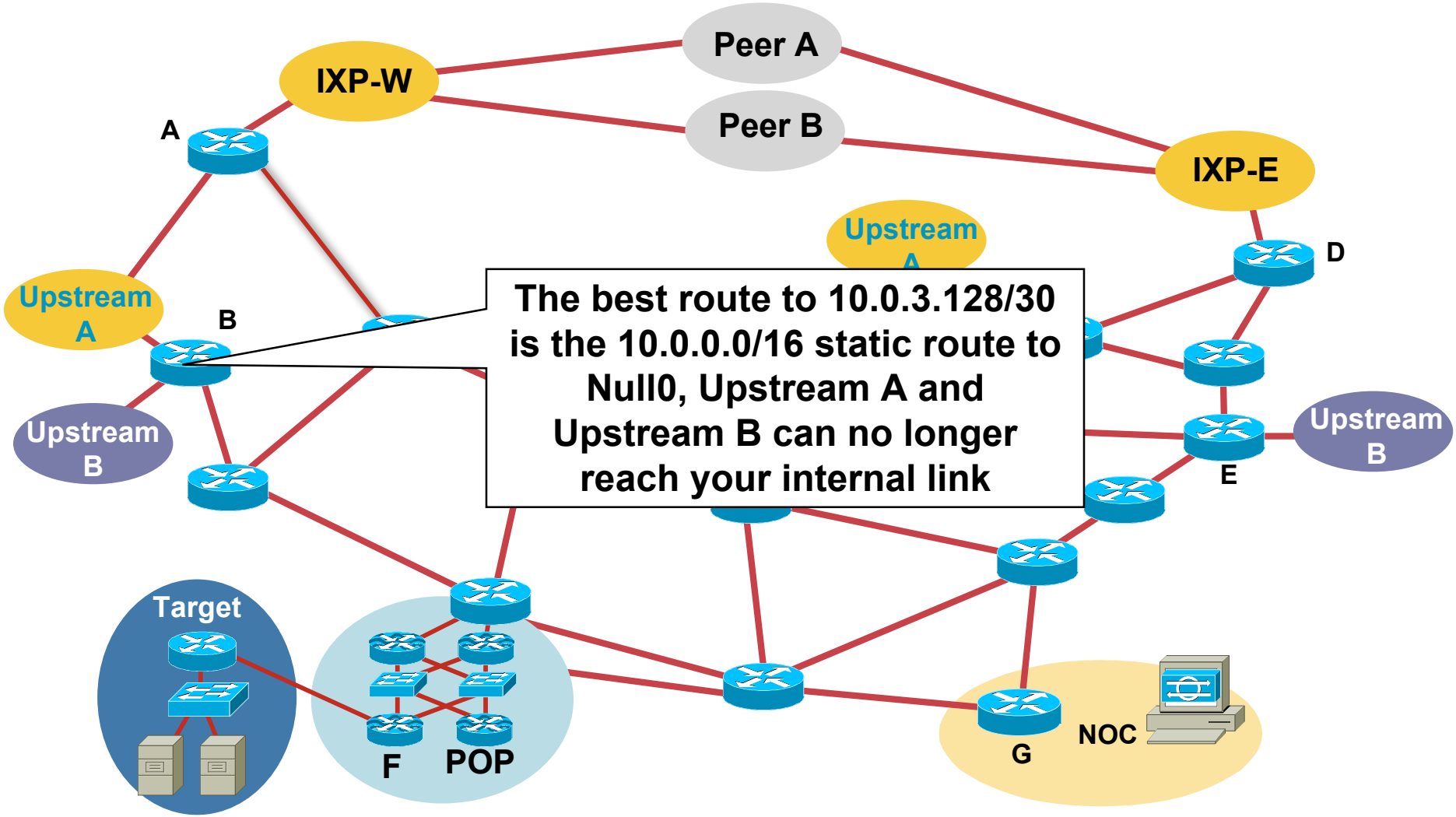
Core Hiding: Internal Links



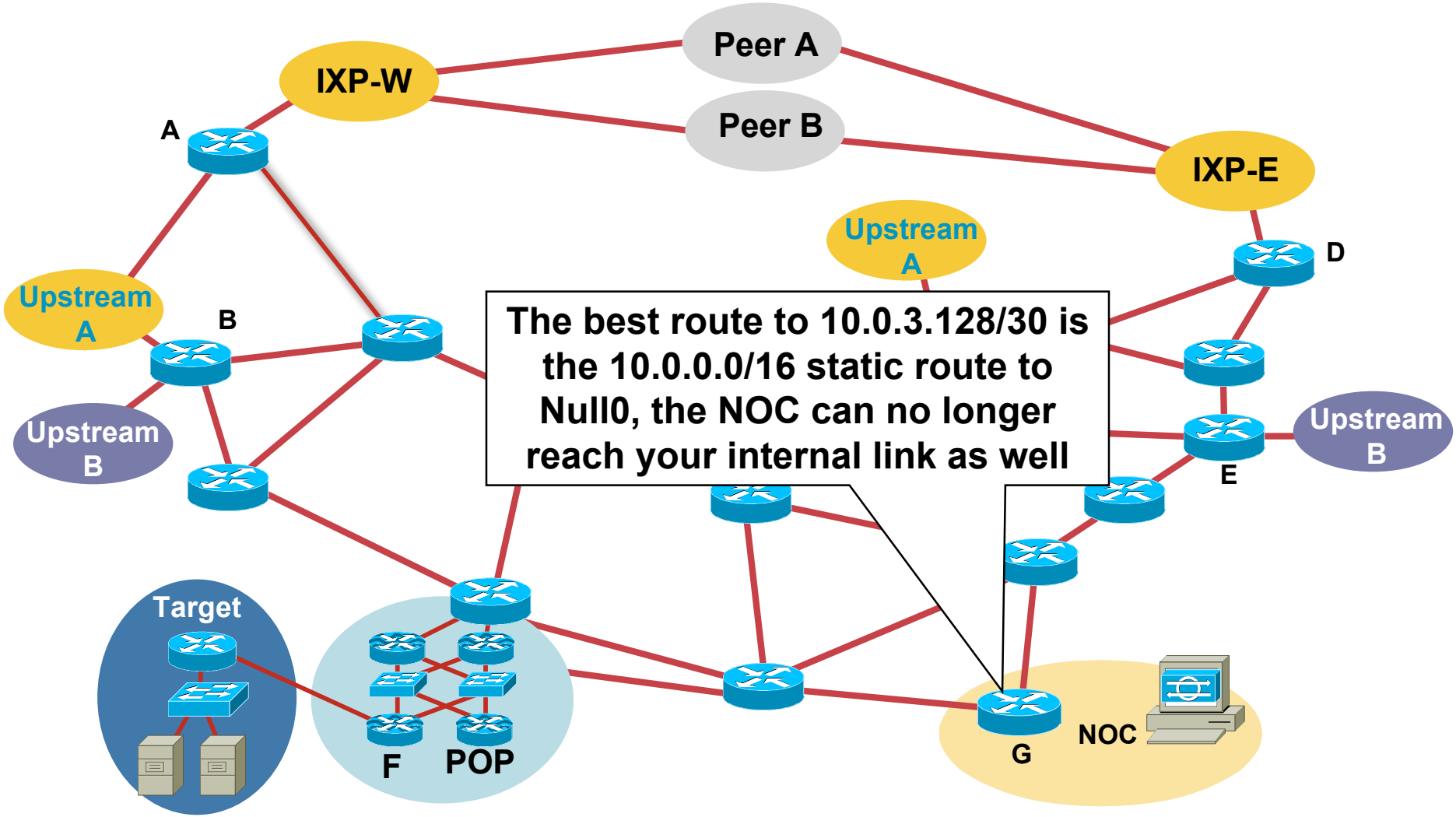
Core Hiding: Internal Links



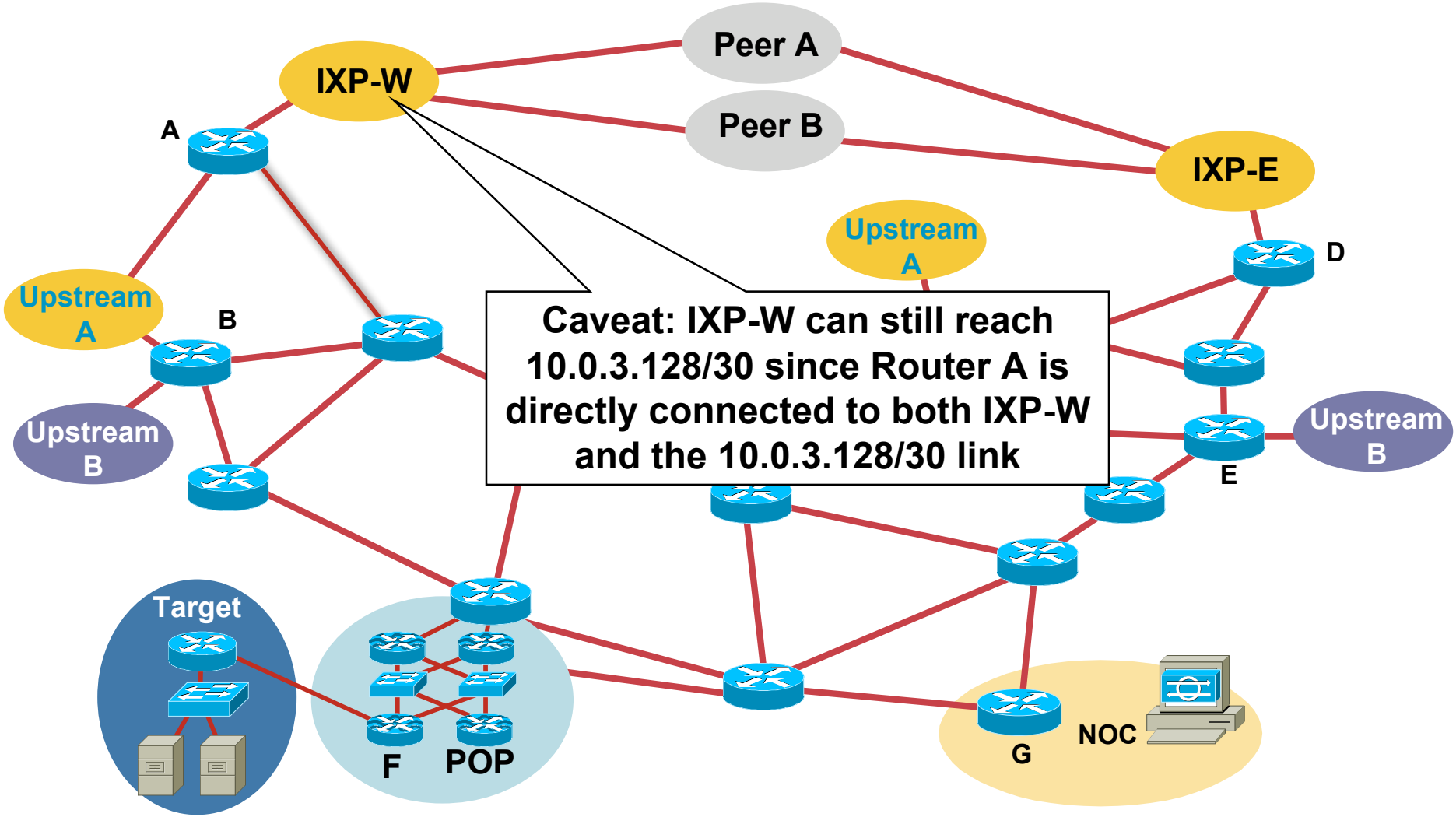
Core Hiding: Internal Links



Core Hiding: Internal Links



Core Hiding: Internal Links



Core Hiding: Internal Links

Caveats:

- Requires use of IS-IS as an igp
- Corner case of an attack ingressing on a router that the internal link is directly connected to

  Does not add 100% security

  Remember—it is all about adding hurdles
- Can dramatically change the way we troubleshoot

  Traceroute and PING to your routers no longer works

  Traceroute and PING through your routers is not impacted

Core Hiding: Loopbacks

- Loopbacks are still vulnerable
- Cannot use IS-IS routing trick as BGP depends on loopbacks for next-hops, they must be carried in the IGP (IS-IS)
- Only solution is to number loopbacks out of RFC1918 space or withdraw entire loopback aggregate from Internet

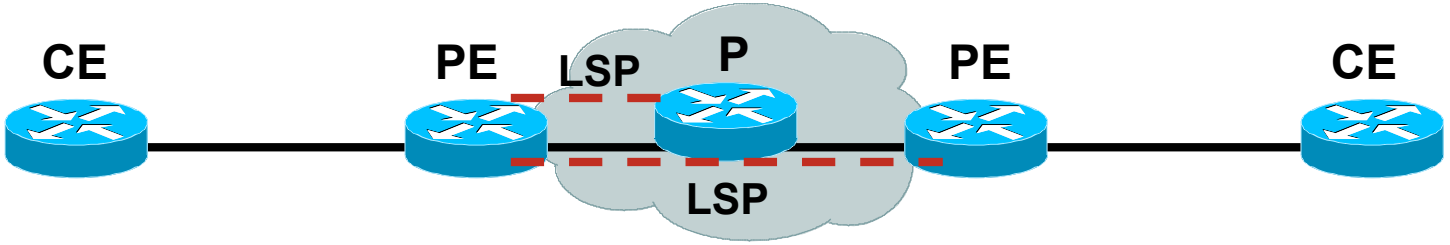
Core Hiding Summary

- Core hiding is an operationally inexpensive method of securing the core
- Core hiding has several weaknesses
  - –Various corner cases where it fails
  - –Requires a specific network design that may not be feasible

Network Hardening: RFC2547 (MPLS) VPN

Network Hardening: RFC2547 (MPLS) VPN

- Place all customers, including the Internet, within distinct VRFs on an IP/MPLS network
- The core becomes opaque and unreachable
- Only the network itself and the NOC should exist in the global table



- An MPLS core by itself is not enough since there are still LSPs to the infrastructure
- PEs must still be secured by another mechanism

Additional References

- iACL Deployment Guide

  http://www.cisco.com/warp/public/707/iacl.html
- rACL Deployment Guide

  http://www.cisco.com/warp/public/707/racl.html
- CoPP Deployment Guide

  http://www.cisco.com/en/US/products/sw/iosswrel/ps1838/products_white_paper09186a00
  80211f39.shtml
- Cisco Network Foundation Protection (NFP)

  http://www.cisco.com/warp/public/732/Tech/security/infrastructure/
- SP Security Archive

  ftp://ftp-eng.cisco.com/cons/isp/security/
- NANOG

  http://www.nanog.org/previous.html

  http://www.nanog.org/ispsecurity.html

Q and A