

Network Monitoring in Practice

Luca Deri <deri@ntop.org>

1. Introduction

Monitoring Requirements [1/4]

- Guarantee the availability of the function on the net.
 - Service maintenance (availability, response time) need to face with technological changes and big quota increase.
 - Security of the services through the control of security components.
 - (Human) Mistake prevention and bottleneck identification/recovery.
 - Automatic or semiautomatic reaction on operation anomalies:
 - Real-time configuration modification in case of error.
 - Activation of redundant components in case of error.

Monitoring Requirements [2/4]

- Dynamic reactions to changes on the network and environment:
 - Changes regarding applications, users, components, services or fees.
 - Dynamic adaptation of the available transmission bandwidth according to requests originated by the management system.

Monitoring Requirements [3/4]

- Network Control:
 - Collection and (compressed) representation of relevant network information.
 - Definition and maintenance of a database of network configurations.
 - When applicable, centralisation of the control over peripherals and implemented functions (central management console).
 - Integration of management procedures on heterogeneous environments

Monitoring Requirements [4/4]

- Improvement of system/network administrators work conditions :
 - Improvement and standardisation of the available tools.
 - Identify and implement gradual automation of management functions.
 - Good integration of tools into the existing operational sequences.
- Progress through standardisation :
 - Transition of existing, often proprietary, solutions in a standardised environment.

Various Actors, Various Metrics

- End-Users vs. (Internet) Service Provider
 - Remote user (Dial-up or xDSL) vs. AOL
 - Internet User (no services provided)
 - Mostly P2P, Email, WWW traffic
 - ntop.org vs. Telecom Italia
 - Provided Services (e.g. DNS, Mail, WWW)
 - Connected to a Regional ISP (no worldwide branches)
 - Interoute/Level3 vs. Telecom Italia
 - Need to buy bandwidth for national customers
 - Need to sign SLA with customers influenced by the SLA signed with the global carrier.

End-Users Requirements

- Monitoring of Application performance:
 - Why this web page takes so long to load?
 - Why does the multicast video isn't smooth?
- Check that the expected SLA can be provided by the available network infrastructure
 - Do I have enough bandwidth and network resources for my needs and expectations?
- Is poor performance "normal" or there's an ongoing attack or suspicious activity?
 - Is there a virus that takes over most of the available resources?
 - Is anybody downloading large files at high priority (i.e. bandwidth monopolization)?

Service Provider Requirements

- Monitor SLA (Service Level Agreements) and current network activities.
- Enforce committed SLA and monitor their violation (if any).
- Detection of network problems and faults.
- Redesign the network and its services based on the user feedback and monitoring outcome.
- Produce forecasts for planning future network usage hence implement extensions before it's too late (digging out ground for laying cables/fibers takes a lot of time).

Problem Statement

- End-users and ISPs speak a different language
 - End-users understand network services
 - Outlook can't open my mailbox.
 - Mozilla isn't able to connect to Google.
 - ISPs talk about networks
 - BGP announces contain wrong data.
 - The main Internet connection is 90% full.
 - We need to sign a peering contract with AS XYZ for cheaper bandwidth.

Traffic Analysis Applications: Some Requirements [1/2]

- What: Volume and rate measurements by application host and conversation.
Why: Identify growth and abnormal occurrences in the network.
- What: Customizable grouping of traffic by logic groups (e.g. company, class of users), geography (e.g. region), subnet.
Why: Associate traffic with business entities and trend growth per grouping (aggregate data isn't very meaningful here: we need to drill-down the analysis at user level).

Traffic Analysis Applications: Some Requirements [2/2]

- What: Customizable filters and exceptions based on network traffic.
Why: Filters can be associated with alarm notifications in the event of abnormal occurrences on the network.
- What: Customizable time-periods to support workday reporting.
Why: Analyzing data based on the calendar helps identifying problems (e.g. the DHCP is running out of addresses every Monday morning between 9-10 AM, but the problem disappears for the rest of the week.)

Further Measurement Issues [1/2]

- Network appliances have very limited measurement capabilities (a router must switch packets first!).
 - Limited to few selected protocols
 - Aggregated measurements (e.g. per interface)
 - Only a few selected boxes can be used for network measurements (e.g. a router is too loaded for new tasks, this L2 switch isn't SNMP manageable)
 - High-speed networks introduce new problems: measurement tools can't cope with high speeds.

Further Measurement Issues [2/2]

- Need to constantly develop new services and applications (e.g. mobile video on 3G phones).
- Most of the services have not been designed to be monitored.
- Most of the internet traffic is consumed by applications (P2P) that are designed to make them difficult to detect and account.
- Modern internet services are:
 - Mobile hence not tight to a location and IP address
 - Encrypted and based on dynamic TCP/UDP ports (no fingerprinting, i.e. 1:1 port to service mapping)

Monitoring Capabilities in Network Equipment

- End-systems (e.g. Windows PC)
 - Completely under user control.
 - Simple instrumentation (just install new apps)
- Standard Network boxes (e.g. ADSL Router)
 - Access limited to network operators
 - Poor set of measurement capabilities
 - Only aggregated data (e.g. per interface)
- Custom Boxes (Measurement Gears)
 - Instrumentable for collecting specific data.
 - Issues in physical deployment so that they can analyze the traffic where

Problem Statement

- Users demand services measurements.
- Network boxes provide simple, aggregated network measurements.
- You cannot always install the measurement box wherever you want (cabling problems, privacy issues).
- New protocols appear every month, measurement protocols are very static and slow to evolve.

Benchmarking Terminology

- Traffic metrics are often not standardized contrary to everyday life metrics (kg, liter etc.).
- Vendors measurements are often performed in slightly different ways making the results not easy to compare.
- RFC 1242 “Benchmarking Terminology for Network Interconnection Devices” defines some common metrics used in traffic measurement.

RFC 1242: Some Definitions

- Throughput
- Latency
- Frame Loss Rate
- Datalink Frame Size
- Back-to-back
- Etc.

Very general RFC, not very “precise/formal”.

RFC 2285: Further Definitions

- “Benchmarking Terminology for LAN Switching Devices”
- It extends the 1242 RFC by adding new definitions that will be used in other RFCs, including:
 - Traffic burst
 - Network load/overload
 - Forwarding rate
 - Errored frames
 - Broadcasts

RFC 2432: Multicast Terminology

- “Terminology for IP Multicast Benchmarking”
- Very peculiar RFC targeting multicast traffic measurement.
- Some metrics:
 - Forwarding and Throughput (e.g. Aggregated Multicast Throughput)
 - Overhead (e.g. Group Join/Leave Delay)
 - Capacity (e.g. Multicast Group Capacity)

RFC 1944: Benchmarking Methodology for Network Interconnect Devices

- It defines how to perform network traffic measurements:
 - Testbed architecture (where to place the system under test)
 - Packet sizes used for measurements
 - IP address to assigned to SUT (System Under Test)
 - IP protocols used for testing (e.g. UDP vs TCP)
 - Use of traffic bursts during measurements (burst vs. constant traffic)
- In a nutshell it defines the test environment to be used for network traffic testing.

Other Benchmarking Methodology (BM) RFCs

- 2285: BM for LAN Switching Devices
- 2544: BM for for Network Interconnect Devices
- 2647/3511: BM for Firewall Performance
- 2761/3116: BM for ATM Benchmarking
- 2889: BM for LAN Switching Devices
- 3918: BM for IP Multicast

RFC 2544: Benchmarking Methodology for Network Interconnect Devices

It defines and specifies how to:

- Verify and evaluate the test results
- Measure common metrics defined in RFC 1242 such as:
 - Throughput
 - Latency
 - Frame Loss
- Handle “test modifiers” such as
 - Broadcast traffic (how can this traffic affect results)
 - Trial duration (how long the test should last)

Common Measurement Metrics

- Performance measurement
 - Availability
 - Response time
 - Accuracy
 - Throughput
 - Utilization
 - Latency and Jitter

Measurement Metrics: Availability [1/2]

- Availability can be expressed as the percentage of time that a network system, component or application is available for a user.
- It is based on the reliability of the individual component of a network.

Measurement Metrics: Availability [2/2]

$$\% \text{ Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

MTBF = mean time between failures

MTTR = mean time to repair following a failure.

Measurement Metrics: Response Time

- Response time is the time it takes a system to react to a given input.
 - Example: In an interactive transaction, it may be defined as the time between the last keystroke by user and the beginning of the resulting display by the computer.
- Short response time is desirable.
- Necessary for interactive applications (e.g. telnet/ssh) not very important for batch applications (e.g. file transfer).

Measurement Metrics: Throughput

- Metric for measuring the quantity of data that can be sent over a link in a specified amount of time.
- Often it is used for giving an estimation of an available link bandwidth.
- Note that bandwidth and throughput are very different metrics.
- Throughput is an application-oriented measure
- Examples:
 - The number of transactions of a given type for a certain period of time
 - The number of customer sessions for a given application during a certain period of time

Measurement Metrics: Utilization

- Utilization is a more fine-grained measure than throughput. It refers to determining the percentage of time that a resource is in use over a given period of time.
- Often very little utilization means that something isn't working as expected (e.g. little traffic because the file server crashed).

Measurement Metrics: Latency and Jitter

- Latency: amount of time it takes a packet from source to destination. It is very important for interactive applications (e.g. online games).
- Jitter: variance of intra-packet delay on a mono-directional link. It is very important for multimedia applications (e.g. internet telephony or video broadcast)
- Both are expressed in ms (milliseconds).

Measurement Metrics: Bandwidth

- Measurement Interval (T_c) - The time interval or “bandwidth interval” used to control traffic bursts.
- Burst Committed (B_c) - The maximum number of bits that the network agrees to transfer during any T_c .
- CIR (Committed Information Rate - The rate at which a network agrees to transfer information under normal conditions, averaged over a minimum increment of time. CIR, measured in bits per second, is one of the key negotiated tariff metrics. $CIR = B_c / T_c$.
- Burst Excess (B_e) - The number of bits to attempt to transmit after reaching the B_c value.
- Maximum Data Rate ($MaxR$) - Calculated value measured in bits per second. $MaxR = ((B_c + B_e)/B_c) * CIR = (B_c + B_e)/T_c$

Per-Link Measurements

- Metrics available on a link
 - # packets, # bytes, # packets discarded on a specific interface over the last minute
 - # flows, # of packets per flow
- It does not provide global network statistics.
- Useful to ISPs for traffic measurements.
- Examples:
 - SNMP MIBs
 - RTFM (Real-Time Flow Measurement)
 - Cisco NetFlow

End-to-End Measurements

- Network performance \neq Application performance
 - Wire-time vs. web-server performance
- Most of network measurements are by nature end-to-end.
- Per path statistics
 - Are paths symmetric? Usually they are not. (Routing issue?)
 - How does the network behave with long/short probe packets?
- It is necessary for deducting per-link performance measurements.

Monitoring Approaches [1/2]

- Active Measurement
 - To inject network traffic and study how the network reacts to the traffic (e.g. ping).
- Passive Measurement
 - To monitor network traffic for the purpose of measurement (e.g. use the TCP three way handshake to measure network round-trip time).

Monitoring Approaches [2/2]

- Active measurements are often end-to-end, whereas passive measurements are limited to the link where the traffic is captured.
- There is no good and bad. Both approaches are good, depending on the case:
 - Passive monitoring on a switched network can be an issue.
 - Injecting traffic on a satellite link is often doable only by the satellite provider.
- Usually the best is to combine both approaches and compare results.

Inline vs. Offline Measurement

- **Inline Measurements**
Measurement methods based on a protocol that flows over the same network where measurements are taken (e.g. SNMP).
- **Offline Measurements**
Measurement methods that use different networks for reading network measurements (e.g. to read traffic counters from CLI using a serial port or a management network/VLAN).

2. SNMP Monitoring

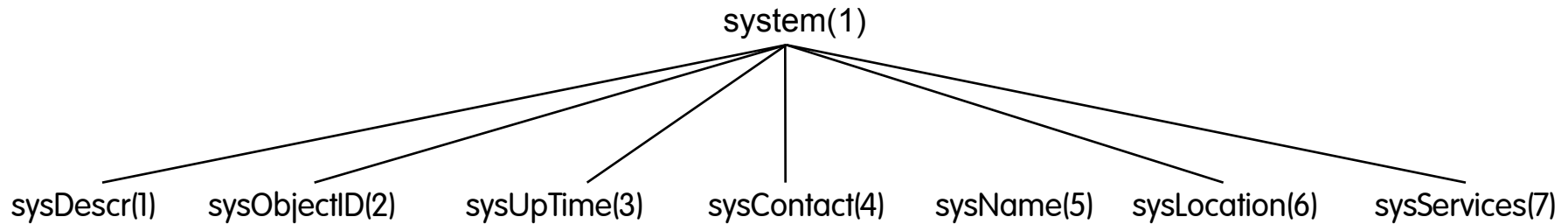
SNMP MIB II: Introduction

- MIB-II (RFC 1213) defines object types for the Internet Protocols IP, ICMP, UDP, TCP, SNMP (and other definitions not relevant here). Basically it models the management of the TCP/IP protocol stack.
- Altogether 170 object types.
- Some MIB definitions turned out to be too simple and minimal (Routing table, Interface table).
- Some MIB definitions presuppose a 4-Byte address format, hence these tables must be redefined for IP version 6 (IPv6).

SNMP MIB II: Goals

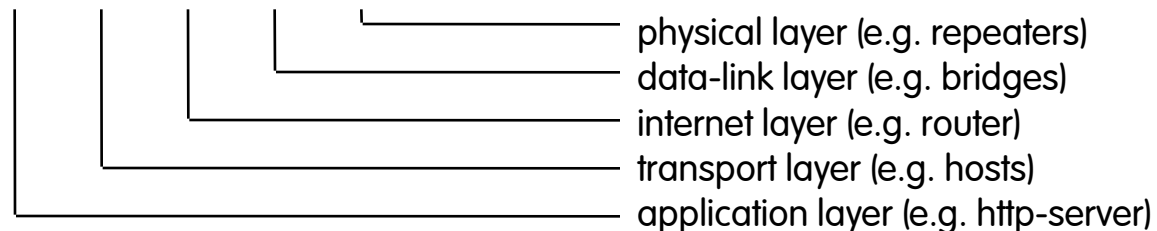
- Goals of the MIB-II definition:
 - Define basic error and configuration management for Internet protocols.
 - Very few and weak control objects.
 - Avoidance of redundant information in the MIB.
 - MIB implementation should not interfere with the normal network activities.
 - No implementation-dependent object types.

"system" Group [1/2]



- `sysUpTime.0` is a very important variable as it is used for determining service discontinuities:
 - If $\text{sysUpTime.0}_{t_1} > \text{sysUpTime.0}_{t_2}$ where $t_2 > t_1$ then the agent has been reinitialised and management application rely on previous values.

- `sysServices.0` roughly reports the services supplied by the system:



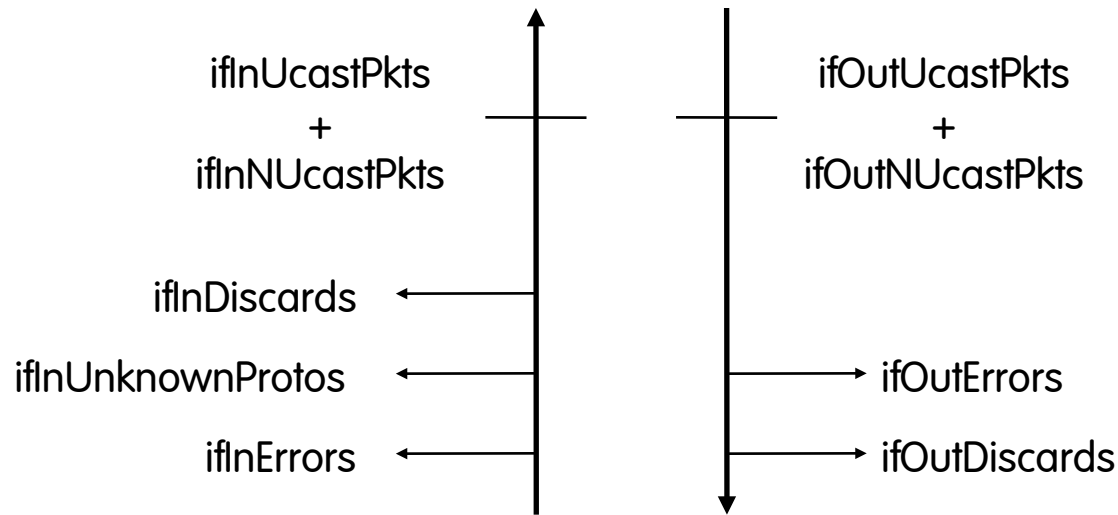
"system" Group [2/2]

- sysObjectld.0 has the format enterprises.<manufacturer>.<id>+ and it is used to identify manufacturer and model. For instance enterprises.9.1.208 identifies a Cisco (.9) 2600 router (.1.208).
- sysDescr.0 provides a precise description of the device (e.g. "Cisco Internetwork Operating System Software IOS (tm) C2600 Software (C2600-I-M), Version 12.2(23), RELEASE SOFTWARE (fc2) Copyright (c) 1986-2004 by cisco Systems, Inc.")
- In a nutshell the system group is important for:
 - Device mapping (via sysObjectld.0, sysDescr.0, and sysLocation.0)
 - Counter wrapping check (sysUpTime.0)
 - Reporting problems about the device to the administrator (sysContact.0)

"interface" Group Variables

- `ifAdminStatus`: the current administrative state of the interface. Values: `up(1)`, `down(2)`, `testing(3)`. A value different from `up` means that the interface is not physically present on the system or that it's present but unavailable to the operating system (e.g. the driver has not been loaded).
Caveat: SNMP MIB index holes
- `ifOperStatus`: the current operational state of the interface. Values: `up(1)`, `down(2)`, `testing(3)`. It is similar to `ifconfig <device> up/down`.
- `ifOutQLen`: the length of the output packet queue (in packets). It is useful for knowing more about transmission speeds and throughput (buffer full means that the receiver is not as fast as the sender).
- `ifLastChange` contains the value of `sysUpTime` at the time the interface entered its current operational state. Useful for detecting when an interface changed state (e.g. cable

Case Diagram for the "interface" Group



- Case diagrams illustrate dependencies between Variables:
 - the number of packets delivered by a network interface to the next higher protocol layer: $\text{ifInUcastPkts} + \text{ifInNUcastPkts}$.
 - the number of packets received by the network: $(\text{ifInUcastPkts} + \text{ifInNUcastPkts}) + \text{ifInDiscards} + \text{ifInUnknownProtos} + \text{ifInErrors}$

Using the "interface" Group [1/2]

- It is the base of SNMP-based monitoring.
- Many tools periodically poll values from interfaces (mostly ifInOctets and ifOutOctets).
- Values are aggregated and not divided per protocol, destination, AS. This is a major limitation if fine grained monitoring is required. The reason is that SNMP counters are basically the kernel counters 'exposed' via SNMP.
- Interface errors can be used for detecting communication problems, especially on WAN links.

Using the "interface" Group [2/2]

- Packet size statistics are not reported however simple Octets/Packets statistics can be computed.
- Many manufacturers (e.g. Cisco, Juniper) report information about both physical and logical interfaces (also known as sub-interfaces). Others (e.g. Extreme) have the entry in the table but counters are always zero.
- Using the interface counters it is possible to produce reports about:
 - VLAN (Virtual LAN)
 - PVC (Private Virtual Circuit) on Frame Relay Links

Using the "arp" Group

- Useful for accessing the arp (Address Resolution Protocol) table of a remote device.
- It can be used for identifying arp-poisoning attacks or misconfigured hosts (e.g. duplicated IP addresses).
- Example:

```
RFC1213-MIB::atIfIndex.4.1.172.22.6.168 = INTEGER: 4
RFC1213-MIB::atIfIndex.4.1.172.22.7.255 = INTEGER: 4
RFC1213-MIB::atPhysAddress.4.1.172.22.6.168 = Hex-STRING: 00 40 F4 67 49 08
RFC1213-MIB::atPhysAddress.4.1.172.22.7.255 = Hex-STRING: FF FF FF FF FF FF
RFC1213-MIB::atNetAddress.4.1.172.22.6.168 = Network Address: AC:16:06:A8
RFC1213-MIB::atNetAddress.4.1.172.22.7.255 = Network Address: AC:16:07:FF
```

Bridge MIB (RFC 1493)

- Useful for controlling the status of L2/L3 switches. Do not make the common mistake to believe that it is used only on bridges.
- It somehow complementary to the MIB II as it provides information the hosts connected to the switch ports.
- Common uses of the bridge MIB:
 - To know the MAC address of a host connected to the port X/unit Y of the switch `dot1dTpFdbTable.dot1dTpFdbAddress` (NOTE: the MIB II has the MAC address of the switch port).
 - The MAC/port association is the base for detecting the physical location of a host. In fact as switch ports are usually connected to wall sockets, this is a good method for know who's where (user -> computer -> switch port -> room/desk)
 - It keeps track of the "previous" MAC address (and the time) connected to a port so it is possible to track users as they move from a room to another.
 - It can be used for detecting ports with associated multiple MAC addresses (trunk) hence to detect users with multiple MACs (e.g. a user runs VMware on his PC, or a user has been infected by a virus/worm) or ports with a switch connected to it that the network

Get Port Mac Address

```
# snmpwalk -c public@1 14.32.6.17 dot1dTpFdbAddress
```

```
.1.3.6.1.2.1.17.4.3.1.1.0.208.211.106.71.251 = Hex-STRING: 00 D0 D3 6A 47
```

```
# snmpwalk -c public@6 14.32.6.17 dot1dTpFdbAddress
```

```
.1.3.6.1.2.1.17.4.3.1.1.0.2.185.144.76.102 = Hex-STRING: 00 02 B9 90 4C 66
```

```
.1.3.6.1.2.1.17.4.3.1.1.0.2.253.106.170.243 = Hex-STRING: 00 02 FD 6A AA F3
```

```
.1.3.6.1.2.1.17.4.3.1.1.0.16.13.56.16.0 = Hex-STRING: 00 10 0D 38 10 00
```

```
.1.3.6.1.2.1.17.4.3.1.1.0.96.84.144.248.0 = Hex-STRING: 00 60 54 90 F8 00
```

```
.1.3.6.1.2.1.17.4.3.1.1.0.208.2.214.120.10 = Hex-STRING: 00 D0 02 D6 78 0A
```

```
.1.3.6.1.2.1.17.4.3.1.1.0.208.211.54.162.60 = Hex-STRING: 00 D0 D3 36 A2 3C
```

```
.1.3.6.1.2.1.17.4.3.1.1.0.224.30.159.10.210 = Hex-STRING: 00 E0 1E 9F 0A D2
```

Note:

- the <community>@<id> means that MAC is searched on VLAN X

Get MAC Address Port [1/2]

```
# snmpwalk -c public@1 14.32.6.17 dot1dTpFdbPort
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.208.211.106.71.251 = INTEGER: 113
```

```
# snmpwalk -c public@6 14.32.6.17 dot1dTpFdbPort
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.2.185.144.76.102 = INTEGER: 113
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.2.253.106.170.243 = INTEGER: 113 <- this is not ifIndex
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.6.83.198.64.173 = INTEGER: 113
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.16.13.56.16.0 = INTEGER: 113
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.96.84.144.248.0 = INTEGER: 113
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.208.2.214.120.10 = INTEGER: 113
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.208.211.54.162.60 = INTEGER: 113
```

```
.1.3.6.1.2.1.17.4.3.1.2.0.224.30.159.10.210 = INTEGER: 65
```

```
nms-server2:/home/ccarring> snmpwalk -c public 14.32.6.17 dot1dBasePortIfIndex
```

```
.1.3.6.1.2.1.17.1.4.1.2.68 = INTEGER: 12
```

```
.1.3.6.1.2.1.17.1.4.1.2.69 = INTEGER: 13
```

```
.....
```

Get MAC Address Port [2/2]

```
# snmpwalk -On -c public 14.32.6.17 ifName
.1.3.6.1.2.1.31.1.1.1.1.1 = STRING: sc0
.1.3.6.1.2.1.31.1.1.1.1.2 = STRING: sl0
.1.3.6.1.2.1.31.1.1.1.1.3 = STRING: me1
.1.3.6.1.2.1.31.1.1.1.1.4 = STRING: VLAN-1
.1.3.6.1.2.1.31.1.1.1.1.5 = STRING: VLAN-1002
.1.3.6.1.2.1.31.1.1.1.1.6 = STRING: VLAN-1004
.1.3.6.1.2.1.31.1.1.1.1.7 = STRING: VLAN-1005
.1.3.6.1.2.1.31.1.1.1.1.8 = STRING: VLAN-1003
.1.3.6.1.2.1.31.1.1.1.1.9 = STRING: 2/1
...
.1.3.6.1.2.1.31.1.1.1.1.55 = STRING: 2/47
.1.3.6.1.2.1.31.1.1.1.1.56 = STRING: 2/48
.1.3.6.1.2.1.31.1.1.1.1.57 = STRING: 2/49 (Slot 2, port 49)
.1.3.6.1.2.1.31.1.1.1.1.58 = STRING: 2/50
```

Side note:

SNMP vs. CLI Counters [1/4]

- It a common belief among the network administrator community that SNMP and CLI counters are basically a different view of same thing.
- Many administrators do like CLI counters more, as:
 - Are formatted for direct human consumption
 - 0 packets input, 0 packets output
 - Many implementations provide command to clear/reset counter
 - clear interface ethernet 3
- Note: the definition of what a given counter counts is dependent on vendor documentation

Side note:

SNMP vs. CLI Counters [2/4]

```
c4500#sh int e1
Ethernet1 is up, line protocol is down
Last clearing of "show interface" counters never
Output queue 0/40, 0 drops; input queue 0/75, 0 drops
0 packets input, 0 bytes, 0 no buffer
    Received 0 broadcasts, 0 runts, 0 giants
    0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
    0 input packets with dribble condition detected
187352 packets output, 11347294 bytes, 0 underruns
187352 output errors, 0 collisions, 3 interface resets
```

- Notes:
 - CLI counters remain the basic way of life in element management.
 - Counters format/appearance change vendor to vendor (often even within the same manufacturer, e.g. Cisco IOS vs. CatOS vs. PIX).
 - Note: IOS, CatOS, and PIX are respectively the router, switch and firewall OS used by Cisco appliances.

Side note:

SNMP vs. CLI Counters [3/4]

- SNMP counters instead:
 - Allow you to compare apples to apples
 - Counters have standard definitions
 - as defined by IETF, IEEE, some vendors...
 - regardless of network element type or vendor
 - and globally unique, hard to pronounce names
 - 1.3.6.1.2.1.17.2.4 dot1dStpTopChanges
 - Have a well specified size
 - 32 or 64 bits wide (64 bit available in SNMP v2c or v3).
 - Counters do not necessarily start at zero
 - Vendor implementation friendly.
 - Are not designed for directing human consumption

Side note:

SNMP vs. CLI Counters [4/4]

`dot1dTpPortInFrames` OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The **number of frames** that have been received by this port from its segment. Note that a frame received on the interface corresponding to this port is **only counted by this object if and only if it is for a protocol being processed by the local bridging function, including bridge management frames.**"

REFERENCE

"IEEE 802.1D-1990: Section 6.6.1.1.3"

- Note: good counters are generally derived from underlying protocol specification.

How To Calculate Bandwidth Utilization (%) with SNMP

Bandwidth Utilization
$$\frac{(\Delta \text{ifInOctets} + \Delta \text{ifOutOctets}) \times 8 \times 100}{(\Delta \text{time}) \times \text{ifSpeed}}$$

Input Utilization
$$(\Delta \text{ifInOctets}) \times 8 \times 100 / ((\Delta \text{time}) \times \text{ifSpeed})$$

Output Utilization
$$(\Delta \text{ifOutOctets}) \times 8 \times 100 / ((\Delta \text{time}) \times \text{ifSpeed})$$

See: http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a008009496e.shtml

What else can you do with SNMP?

- Detect and clear hung TCP connections.
- Manipulate ARP entries.

- Get environmental temperature.
- Check CPU utilization.
- Monitor redundant/uninterruptible power supplies.
- Find P2P users (NAT table).
- Layout network topology (e.g. via CDP)

See: http://www.cisco.com/en/US/tech/tk648/tk362/tk605/tsd_technology_support_sub-protocol_home.html

3. Remote Monitoring

Networks are Changing... [1/2]

Force 1: the Internet

- network security will become even more critical in the future.
- the enterprise network will become a public network.

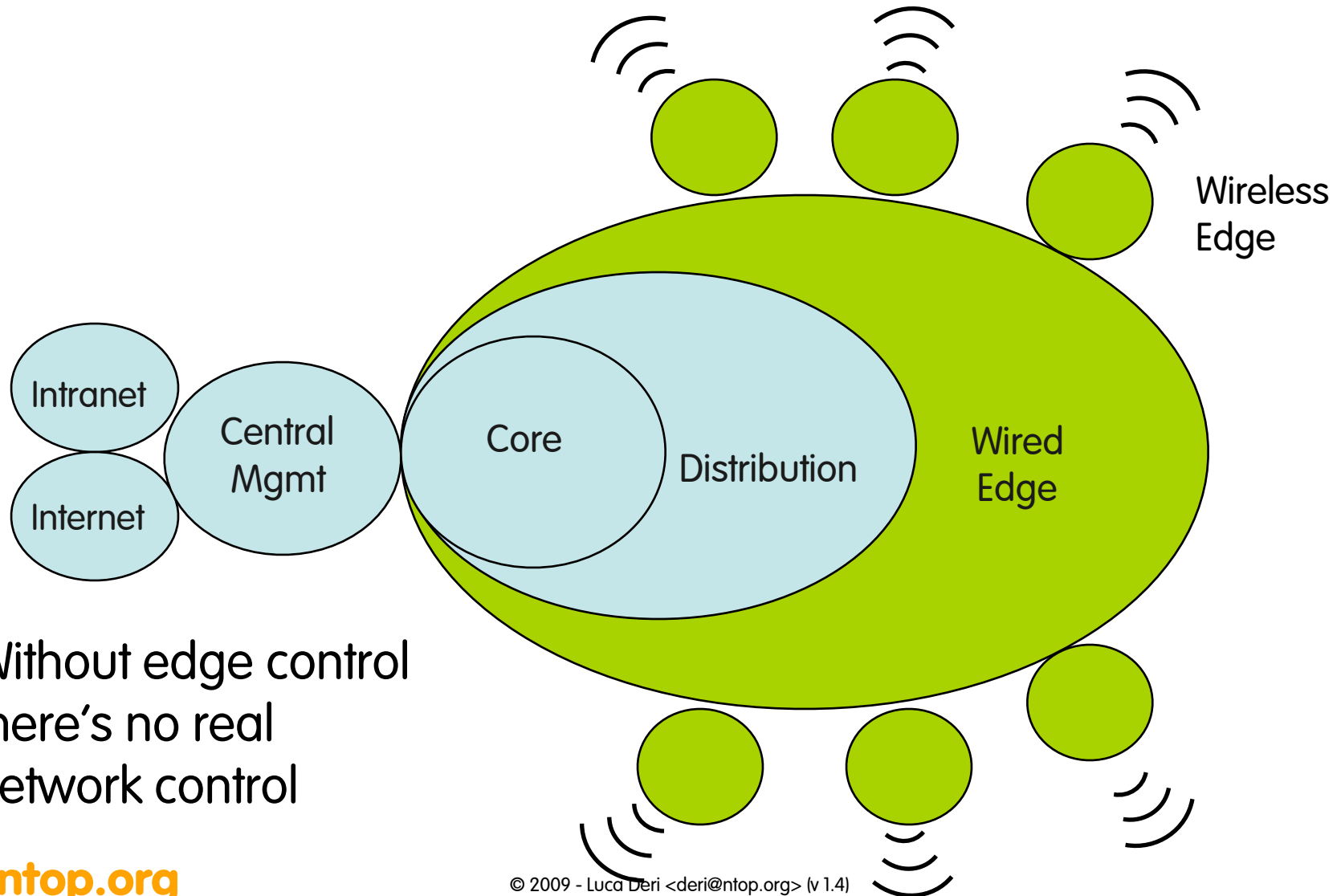
Force 2: Mobility

- supporting mobility across wired and wireless will be a key element of the network future.
- the network will become an anytime, anywhere resource.

Force 3: Dynamic Communications

- supporting a broad range of applications will be a key element in the future: convergence.
- the data network will become the only network.

Networks are Changing... [2/2]



Without edge control
there's no real
network control

Towards Remote Monitoring [1/4]

- Modern networks are distributed across various buildings, managed by different people with different skills (security, traffic engineer, DB administrator).
- It is necessary to collect traffic statistics on each network trunk, send them to a (limited number of) collector, in order to produce an aggregate network view.
- Some distributed analysis capabilities are necessary because a centralized network is not fault tolerant and scalable.

Towards Remote Monitoring [2/4]

- Deploying remote traffic analyzers (e.g. pcap-based probes) are not always feasible because:
 - Server manufacturers do not always permit generic, untested software (e.g. the licence enforces that on a Oracle server you install only Oracle-certified apps) to be installed.
 - Modern servers often have several network interfaces (1Gb main+failover for data and 100 Mbit for server access), so a multi-interface probe is required.
 - Monitoring a 1 GE using a network tap requires 2 x GE (one RX for each direction of the original GE).

Towards Remote Monitoring [3/4]

- Solution: use traffic analysis capabilities provided by network appliances.
- Drawbacks:
 - Not all the appliances provide traffic analysis capabilities (e.g. most ADSL routers do not)
 - Even if supported, not always such capabilities can be enabled (strong impact on CPU and memory).
 - Basic monitoring capabilities provided by the default OS are rather limited so a custom card is necessary.
 - Custom cards for traffic analysis are not so cheap.

Towards Remote Monitoring [4/4]

- Price of some commercial monitoring cards (monitoring software sold separately):

Product	Price (Card Only)
Cisco MSFC-2	46'000 \$
Juniper PM-PIC	30'000 \$

RMON: Remote Monitoring using SNMP

- Present in most mid-high end network appliances: often these are poor/limited implementations.
- Some vendors sell stand-alone probes: preferred case as
 - they are full implementations of the protocol.
 - They do not add additional load on the router.
- Not all the implementations (in particular those embedded in router/switches) support the whole standard but only selected SNMP groups.
- Together with Cisco NetFlow is the industrial, “trusted” monitoring standard.

What can RMON do?

- Collect data and periodically report it to a more central management station, which potentially reduces traffic on WAN links and polling overhead on the management station.
- Report on what hosts are attached to the LAN, how much they talk, and to whom.
- "See" all LAN traffic, full LAN utilization, and not just the traffic to or through the router.
- Filter and capture packets (so you don't have to visit a remote LAN and attach a LAN Analyzer) : it is basically a remote sniffer that can capture real-time traffic (until the integrated memory buffer is full).
- Automatically collect data, compare to thresholds, and send traps to your management station -- which offloads much of the work that might bog down the management station.

RMON vs. SNMP [1/2]

- The SNMP protocol is used to control and configure a probe. Usually GUI managers mask the complexity of SNMP-based configuration.
- Statistics and saved traffic are retrieved using SNMP by management applications to record statistics on a network and, possibly selected portions of the network traffic.

RMON vs. SNMP [2/2]

SNMP and RMON differ in the way they gather traffic statistics:

- SNMP is a periodic poll-request process: it requires a query of the SNMP device to get network statistics (the network status is kept by the manager).
- RMON, on the other hand, reduces the stress of the manager by gathering and storing the statistics in counters or buckets for retrieval by a management station.

RMON Monitoring Groups [1/3]

Groups	Function	Elements
Statistics	Contains statistics measured by the probe for each monitored interface on this device.	Packets dropped, packets sent, bytes sent (octets), broadcast packets, multicast packets.
History	Records periodic statistical samples from a network and stores them for later retrieval.	Sample period, number of samples, items sampled.
Alarm	Periodically takes statistical samples from variables in the probe and compares them with previously configured thresholds. If the monitored variable crosses a threshold, an event is generated.	Alarm type, interval, starting threshold, stop threshold.

RMON Monitoring Groups [2/3]

Groups	Function	Elements
Host	Contains statistics associated with each host discovered on the network.	Host address, packets, and bytes received and transmitted, as well as broadcast, multicast, and error packets.
HostTopN	Prepares tables that describe the hosts that top a list ordered by one of their base statistics over an interval specified by the management station. Thus, these statistics are rate-based.	Statistics, host(s), sample start and stop periods, rate base, duration.
Matrix	Stores statistics for conversations between sets of two addresses. As the device detects a new conversation, it creates a new entry in its table.	Bit-filter type (mask or not mask), filter expression (bit level), conditional expression (and, or not) to other filters.

RMON Monitoring Groups [3/3]

Groups	Function	Elements
Filters	Enables packets to be matched by a filter equation. These matched packets form a data stream that might be captured or that might generate events. Bit-filter type (mask or not mask), filter expression (bit level), conditional expression (and, or not) to other filters	Bit-filter type (mask or not mask), filter expression (bit level), conditional expression (and, or not) to other filters
Packet Capture	Enables packets to be captured after they flow through a channel.	Size of buffer for captured packets, full status (alarm), number of captured packets.
Events	Controls the generation and notification of events from this device.	Event type, description, last time event sent

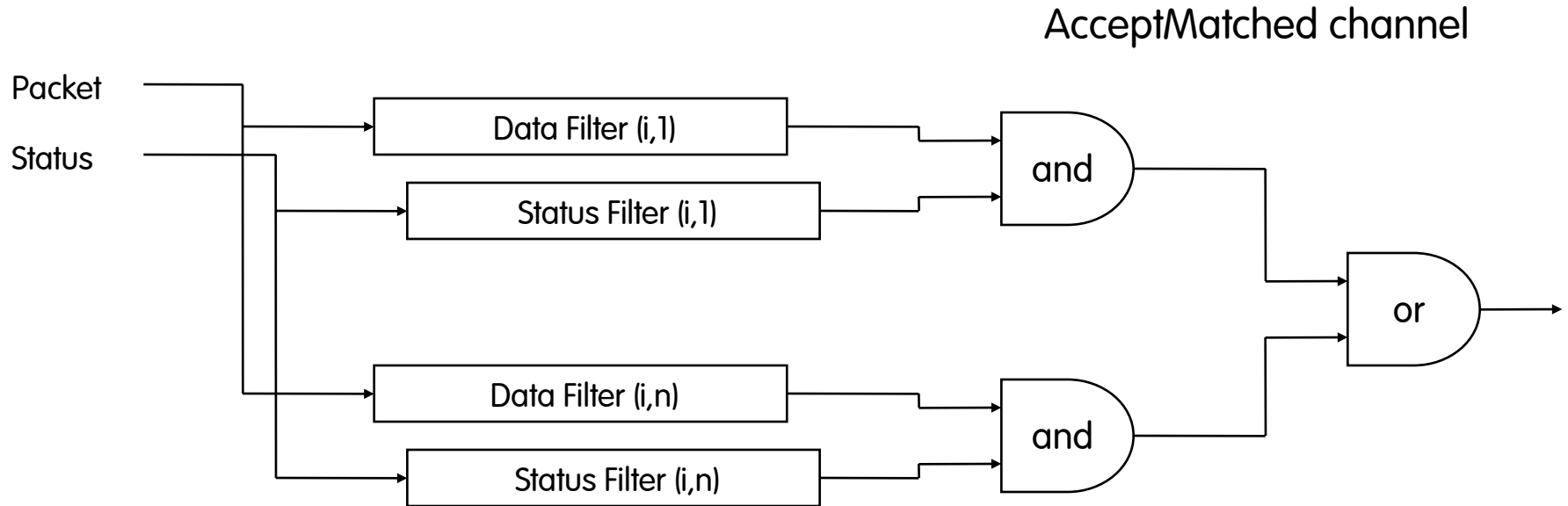
RMON Ethernet Statistics

- Packets: A unit of data formatted for transmission on a network.
- Multicast Packet: communication between a single sender and multiple receivers on a network.
- Broadcast Packet: a packet that is transmitted to all hosts on an Ethernet.
- Drop Events: An overrun at a port. The port logic could not receive the traffic at full line rate and had to drop some packets.
- Fragments: A piece of a packet. Sometimes a communications packet being sent over a network has to be temporarily broken into fragments; the packet should be reassembled when it reaches its destination.
- Jabbers: Packets received that were longer than 1518 octets and also contained alignment errors.
- Oversize Packets: Packets received that were longer than 1518 octets and were otherwise well formed.

RMON-1 Filters and Channels

- Received packets can be selected by means of filters. Filters are simple value/mask expressions (like with IP address, network and mask).
- A RMON channel is defined by a set of filter pairs: data [the packet] and status [packet info (e.g. length)] filters.
- A packet is accepted by a channel when,
 - if at least one pair of filters hold (acceptMatched channel)
 - or if at least one filter of all filters pairs does not pass the test (acceptFailed channel).

RMON Filter/Channel Example



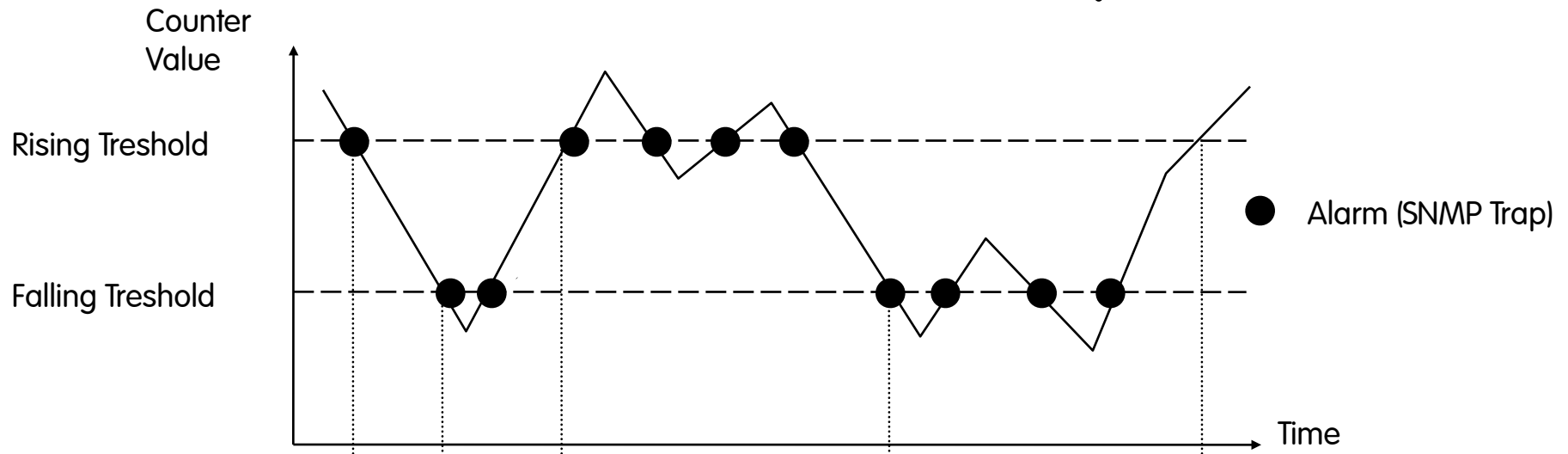
See <http://www.csu.edu.au/special/auugwww96/proceedings/wang/wang.html>

Network Utilization with RMON

- Most RMON managers use RMON counters to compute network utilization.
- Network Utilization can be calculated for all the ports of a given switch at regular intervals. This information can be gathered over the course of a day and be used to generate a network utilization profile of a switch or hub.

$$\% \text{ Network Utilization} = \frac{100 \times ((\# \text{ packets} \times 160) + (\# \text{ octets} \times 8))}{\text{port speed} \times \text{time (secs)}}$$

RMON Alarm Group



- In case of exceeding a upper limit value, an event is produced each time the threshold is exceeded or when a value that used to be above a threshold returns inside the specified range. Similar considerations can be applied to lower threshold value.
- Thresholds can either be on to the measured value (absolute) or on the difference of the current value to the last measured value (delta value).

Case Study: Counter Sampling Interval [1/2]

Example 1: (10 seconds sampling interval, threshold value 20, 10 seconds test interval)

Time:	0	10	20
Value:	0	19	32
Delta:		19	13
Actual Threshold To Check:		19	13

Example 2: (5 seconds sampling interval, threshold value 20, 10 seconds test interval)

Time:	0	5	10	15	20
Value:	0	10	19	30	32
Delta:		10	9	11	2

Case Study: Counter Sampling Interval [2/2]

- MIB instance value sampling must be done twice per sampling interval, otherwise exceeded thresholds may be undetected for overlapping intervals.
- Fast polling has some drawbacks:
 - Much more data is collected.
 - Increased load on SNMP agents.
 - More data changes are detected (this can lead to false positives).
- Slow polling has some drawbacks too:
 - Some alarms can be missed (inaccuracy)

RMON-Like Home-grown Network Probes [1/4]

- Every router/switch (ranging from Cisco boxes to Linux-based router) has the ability to define ACL (Access Control Lists) for preventing selected traffic to flow.
- ACLs with an 'accept' policy can very well be used to account traffic.
- Drawbacks:
 - ACLs are limited to IP whereas RMON is not (e.g. IPX, NetBEUI)
 - On many systems ACLs have impact on CPU.
 - The number or total/per-port ACLs is limited.
 - Often ACLs are limited to packet header (no payload).

RMON-Like Home-grown Network Probes [2/4]

ACL definition examples:

– Cisco

```
access-list 102 permit icmp any any
```

– Juniper

```
filter HTTPcounter {  
  from {  
    destination-address {  
      10.10.20/24;  
      10.40.30/25;  
      11.11/8;  
    }  
    destination-port [http https];  
  }  
  then {  
    count Count-Http;  
    accept  
  }  
}
```


RMON-Like Home-grown Network Probes [3/4]

- Linux (iptables)

```
[root@mail deri]# /sbin/iptables -xnvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts      bytes target     prot opt in     out     source            destination
 236675 169960206 RH-Firewall-1-INPUT all  --  *      *       0.0.0.0/0         0.0.0.0/0
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts      bytes target     prot opt in     out     source            destination
      0          0 RH-Firewall-1-INPUT all  --  *      *       0.0.0.0/0         0.0.0.0/0
Chain OUTPUT (policy ACCEPT 262868 packets, 233122676 bytes)
  pkts      bytes target     prot opt in     out     source            destination
Chain RH-Firewall-1-INPUT (2 references)
  pkts      bytes target     prot opt in     out     source            destination
 68169 81214627 ACCEPT     all  --  lo     *       0.0.0.0/0         0.0.0.0/0
   677   53751 ACCEPT     icmp --  *      *       0.0.0.0/0         0.0.0.0/0
      0          0 ACCEPT     esp  --  *      *       0.0.0.0/0         0.0.0.0/0
      0          0 ACCEPT     ah   --  *      *       0.0.0.0/0         0.0.0.0/0
```

RMON-Like Home-grown Network Probes [4/4]

- Counters are usually accessed from SNMP in addition to CLI (Command Line Interface).
- Proprietary MIBs allow values to be read from remote.
- Cisco has recently introduced a new technology named "Static NetFlow" that allows routers to emit flows for each defined ACL.
- Extreme Network's "ClearFlow" is also a similar technology. In addition it also has the ability to send alarms by setting thresholds on counter values.

NBAR: RMON-Like Traffic Stats [1/6]

- Cisco NBAR (Network Based Application Recognition) is a traffic classification engine with QoS support (i.e. you can shape traffic based on traffic stats).
- Real-time traffic pattern analysis (with payload analysis) and protocol discovery.
- NBAR Example: stop KaZaA traffic and give priority to the video conference traffic.

NBAR: RMON-Like Traffic Stats [2/6]

- Capable of classifying applications that have:
 - Statically assigned TCP and UDP port numbers.
 - Non-TCP and non-UDP IP protocols.
 - Dynamically assigned TCP and UDP port numbers during connection establishment.
 - Classification based on deep packet inspection: NBAR can look deeper into the packet to identify applications.
 - HTTP traffic by URL, host name or MIME type using regular expressions (*, ?, []), Citrix ICA traffic, RTP Payload type classification.
 - Currently supports 88 protocols/applications.
- NBAR statistics can be read using SNMP (Cisco NBAR Protocol

NBAR: RMON-Like Traffic Stats [3/6]

Caveats:

- Proprietary technology: available only on Cisco boxes with a recent IOS version.
- Strong router CPU overhead (more than NetFlow).
- It does not recognize all protocols.
- Difficult to configure in particular if associated with QoS/Bandwidth Management.

NBAR: RMON-Like Traffic Stats [4/6]

```
Router# conf t
Router(config)# ip cef
Router(config)# int eth0/0
Router(config-if)# ip nbar protocol-discovery
Router(config-if)# exit
Router(config)# int se0/0
Router(config-if)# ip nbar protocol-discovery
```

```
Router# show ip nbar protocol discovery int eth0/0 top 3
```

```
FastEthernet0/0
```

Protocol	Input	Output
	Packet Count	Packet Count
	Byte Count	Byte Count
	5 minute bit rate (bps)	5 minute bit rate (bps)
-----	-----	-----
ftp	64175242	45153848
	89351513113	2484576000
	1073000	28000
http	58194017	32519125
	82356099996	1958417833

NBAR: RMON-Like Traffic Stats [5/6]

```
Router# show policy-map int eth0/0
```

```
Ethernet0/0
```

```
Service-policy input: dscp_mark
```

```
Class-map: stream (match-any)
```

```
130521 packets, 97066868 bytes
```

```
5 minute offered rate 0 bps, drop rate 0 bps
```

```
Match: protocol rtp
```

```
0 packets, 0 bytes
```

```
5 minute rate 0 bps
```

```
Match: protocol rtspplayer
```

```
117857 packets, 79344153 bytes
```

```
5 minute rate 0 bps
```

```
Match: protocol netshow
```

```
12664 packets, 17722715 bytes
```

```
5 minute rate 0 bps
```

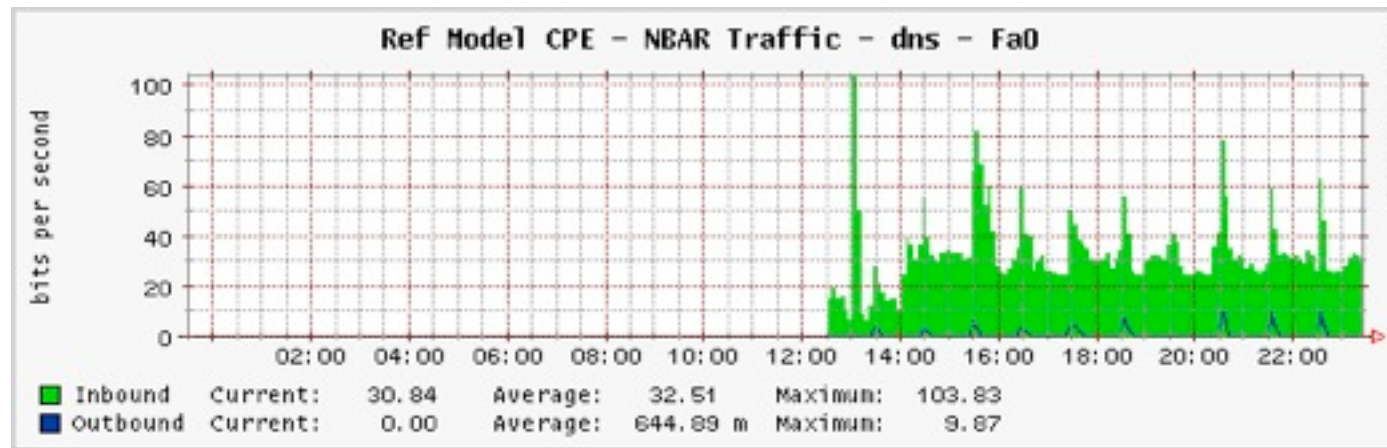
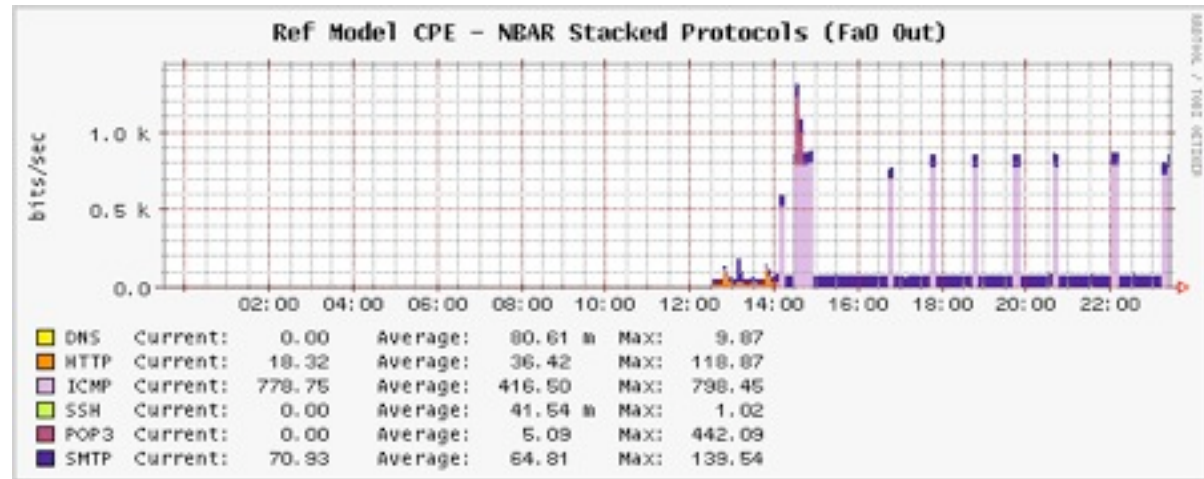
```
Match: ip dscp ef
```

```
0 packets, 0 bytes
```

```
5 minute rate 0 bps
```

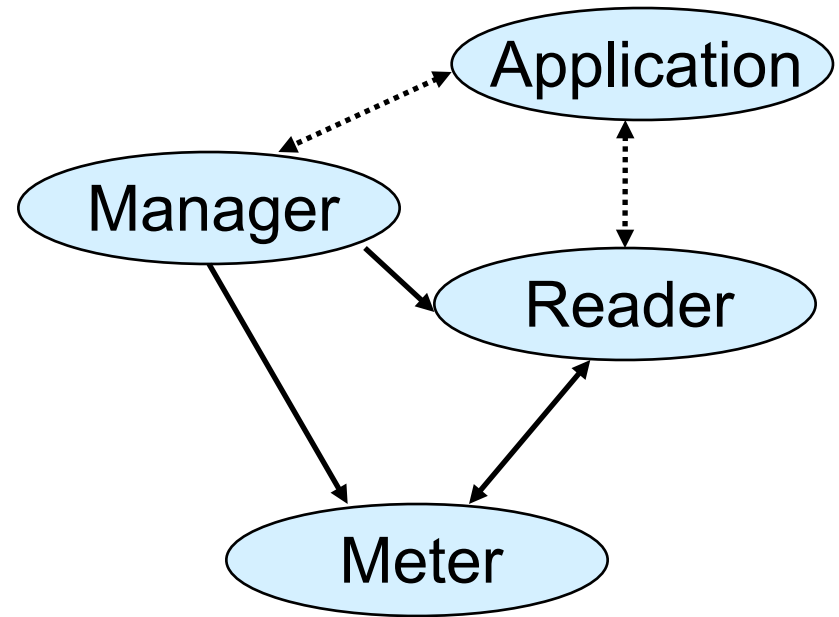
```
QoS Set
```

NBAR: RMON-Like Traffic Stats [6/6]



Real-Time Flow Measurement (RTFM)

- Very flexible and powerful meter
 - programmable rule sets
 - can serve several readers
 - programmable overload behavior
- Reader polls meter
- Realization by SNMP Meter MIB
- Free software implementation NeTraMet
- No acceptance at manufacturers
- Complicated to use (too powerful)
- Specified by RFCs 2720 - 2724



4. Flow Monitoring

SNMP vs. Network Flows [1/2]

- SNMP is based on the manager agent paradigm
 - The agent monitors the network and informs the manager (via traps) when something important happened (i.e. and interface changed state).
 - The manager keeps the whole system status by periodically reading (polling) variables (e.g. via SNMP Get) from the agent.
 - SNMP variables can be used for both element/device/system management (e.g. info about disk space and partitions) and traffic monitoring.

SNMP vs. Network Flows [2/2]

- Network flows are emitted by a probe towards one or more collectors according to traffic conditions.
 - Flows contain information about the analysed traffic (i.e. they do not contain device/probe information such as the MIB II variables).
 - Emitted flows have a well defined format (e.g. Cisco NetFlow v5) and often use UDP as transport (no specialized protocol like SNMP).
 - No concept of 'alarm' flows nor ability for the probe to perform actions based on flows: all the intelligence is in the collector.
 - Probe instrumentation is performed offline.
 - Probes are activated where the network traffic flows (e.g. inside routers and switches).

So What Do You Expect To Measure with Flows? [1/2]

- Where your campus exchanges traffic with by IP address, IP Prefix, or ASN.
- What type and how much traffic (SMTP, WEB, File Sharing, etc).
- What services running on campus.
- Department level traffic summaries.
- Track network based viruses back to hosts.

So What Do You Expect To Measure with Flows? [2/2]

- Track DoS attacks to the source(s), i.e. the 100 servers flooding XXX.com domain.
- Find busy hosts on campus (top host).
- How many destinations each campus host exchanges traffic with.
- Campus host counts by service, i.e. how many active web servers.

What You Can't Measure with Flows?

- Non-IP traffic (e.g. NetBIOS, AppleTalk).
- L2 information (e.g. interface up/down state changes).
- Filtered traffic (e.g. firewall policy counters).
- Per-link statistics (e.g. link usage, congestion, delay, packet loss).
- Application statistics (e.g. transaction latency, # positive/negative replies, protocol errors).

Network Flows: What Are They?

- “A flow is a set of packets with a set of common packet properties” (e.g. common IP address/port).
- A flow is (queued to be) emitted only when expired.
- Creation and expiration policy
 - What conditions start and stop a flow?
 - Maximum flow duration timeout regardless of the connection status (e.g. a TCP connection ends when both peers agreed on FIN/RST).
 - Emit a flow when there’s no flow traffic for a specified amount of time.

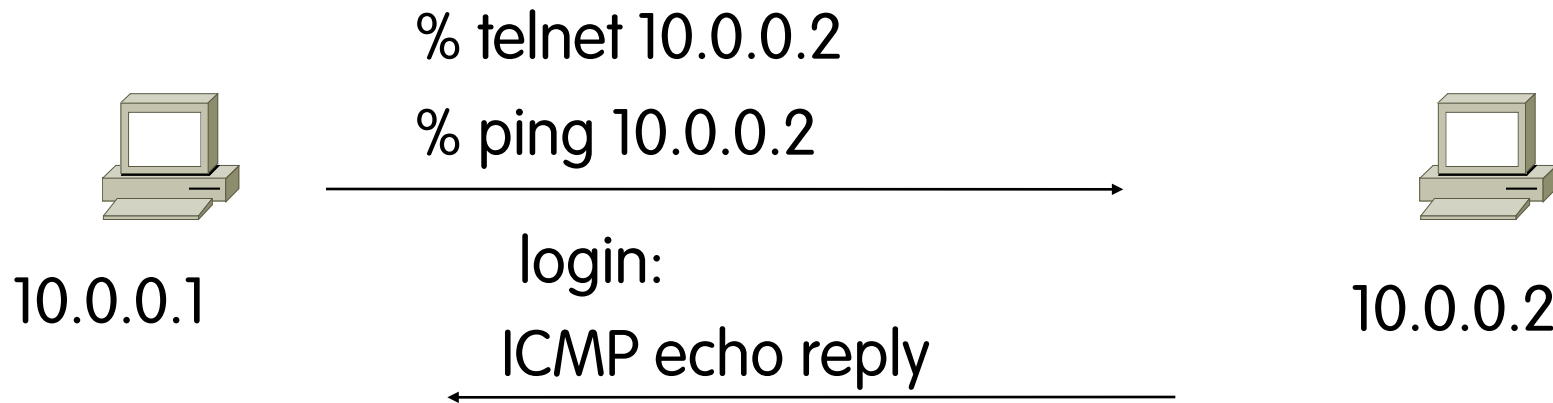
Network Flows Content

- Flow contain:
 - Peers: flow source and destination.
 - Counters: packets, bytes, time.
 - Routing information: AS, network mask, interfaces.
- Flows can be unidirectional (default) or bidirectional (v9/IPFIX only).
- Two, opposite, unidirectional flows are equivalent to one bidirectional flow.
- Bidirectional flows can contain other information such as round trip time, TCP behavior.

Network Flows Issues

- Overhead vs. Accuracy
 - More measurement results in more collected data.
 - More flow aggregation, less granularity.
 - Overhead (e.g. CPU load) on routers, switches, end-hosts.
- Security vs. Data Sharing
 - Emitted flows must reach collectors on protected paths (e.g. using a different network/VLAN).
 - User privacy must be respected.
 - Traffic measurements must be kept protected in order not to disclosure important network information to third parties.

Unidirectional Flow with Source/ Destination IP Key

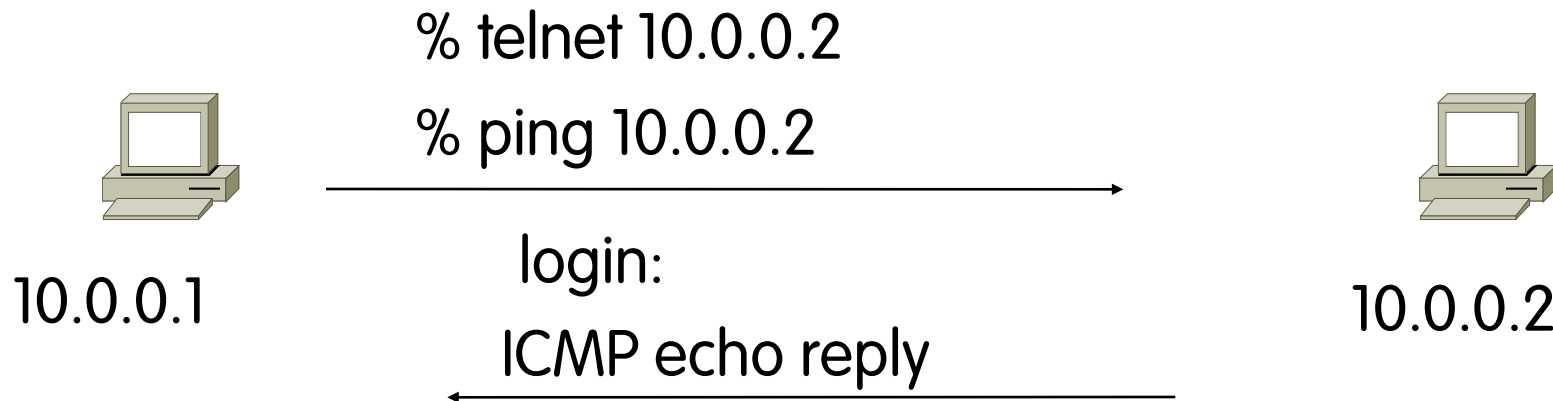


Active Flows

Flow	Source IP	Destination IP
------	-----------	----------------

1	10.0.0.1	10.0.0.2
2	10.0.0.2	10.0.0.1

Unidirectional Flow with IP, Port, Protocol Key

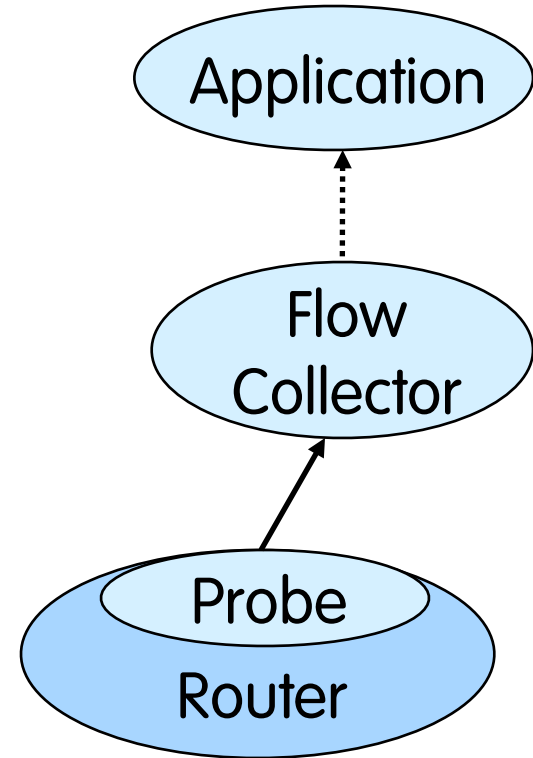


Active Flows

Flow	Source IP	Destination IP	Proto	srcPort	dstPort
1	10.0.0.1	10.0.0.2	TCP	32000	23
2	10.0.0.2	10.0.0.1	TCP	23	32000
3	10.0.0.1	10.0.0.2	ICMP	0	0
4	10.0.0.2	10.0.0.1	ICMP	0	0

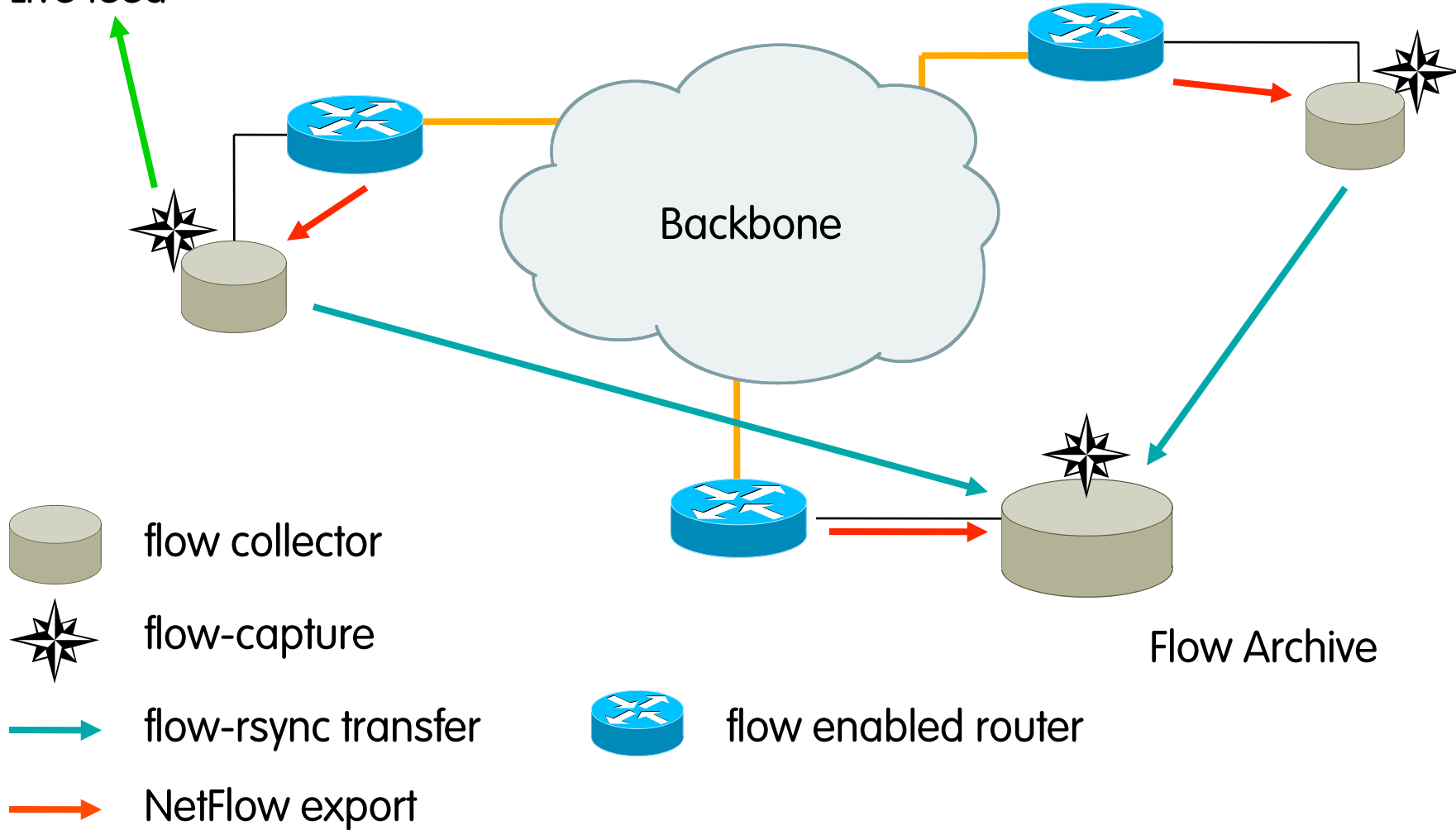
NetFlow Architecture

- Flows are exported (push) by the probe when expired, contrary to SNMP where the manager polls the agent periodically.
- The flow transport protocol is NetFlow (no SNMP).
- Probe/collector configuration protocol is not specified by the NetFlow protocol.
- The NetFlow collector has the job of assembling and understanding the exported flows and combining or aggregating them to produce the valuable reports used for traffic and security analysis.

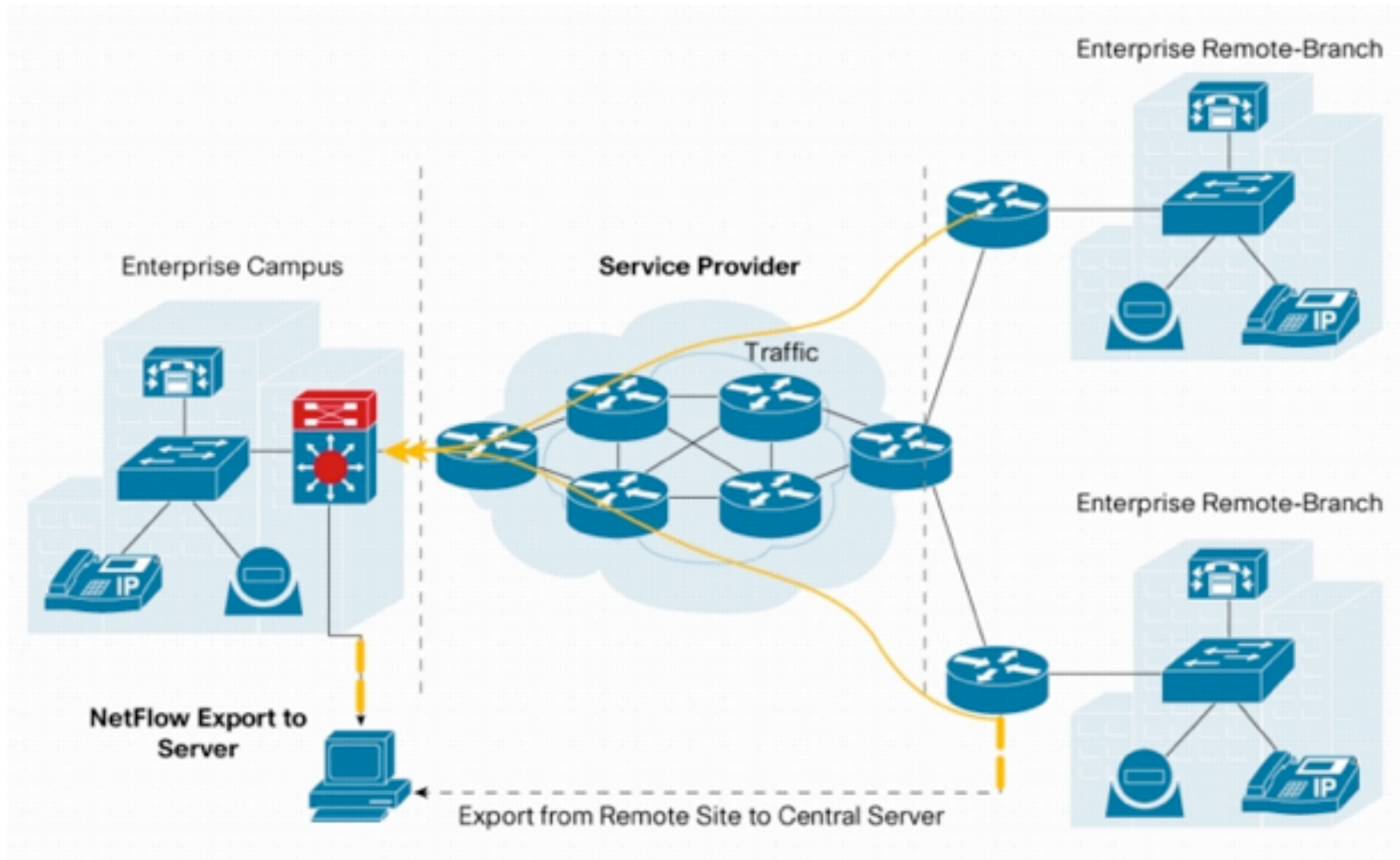


A Collection Architecture [1/2]

Live feed



A Collection Architecture [2/2]



Collection Space Constraints

- Space required depends on traffic
- Some average figures:
 - 67.320 octets/flow, 92 packets/flow
 - Busy router: 397 GB of traffic/day,
548,000,000 packets/day == 5.900.000
flows/day
 - At 60 bytes/flow, this is 350 MB of logs/day
 - With level 6 compression we get 4.3:1
 - Which works out to 82 MB/day for this router

Cisco NetFlow Basics

- Unidirectional flows (up to v8), bidirectional on v9.
- Several versions v 1,5,6,7,8,9. The most common is v5, the latest version is v9.
- Traffic analysis only on inbound (i.e. the traffic that enters the router) IP-only traffic (not on all platforms).
- IPv4 unicast and multicast: all NetFlow versions. IPv6 is supported only by v9.
- Open protocol defined by Cisco and supported on IOS and CatIOS platforms (no NetFlow support on PIX firewalls) as well as on on-Cisco platforms (e.g.

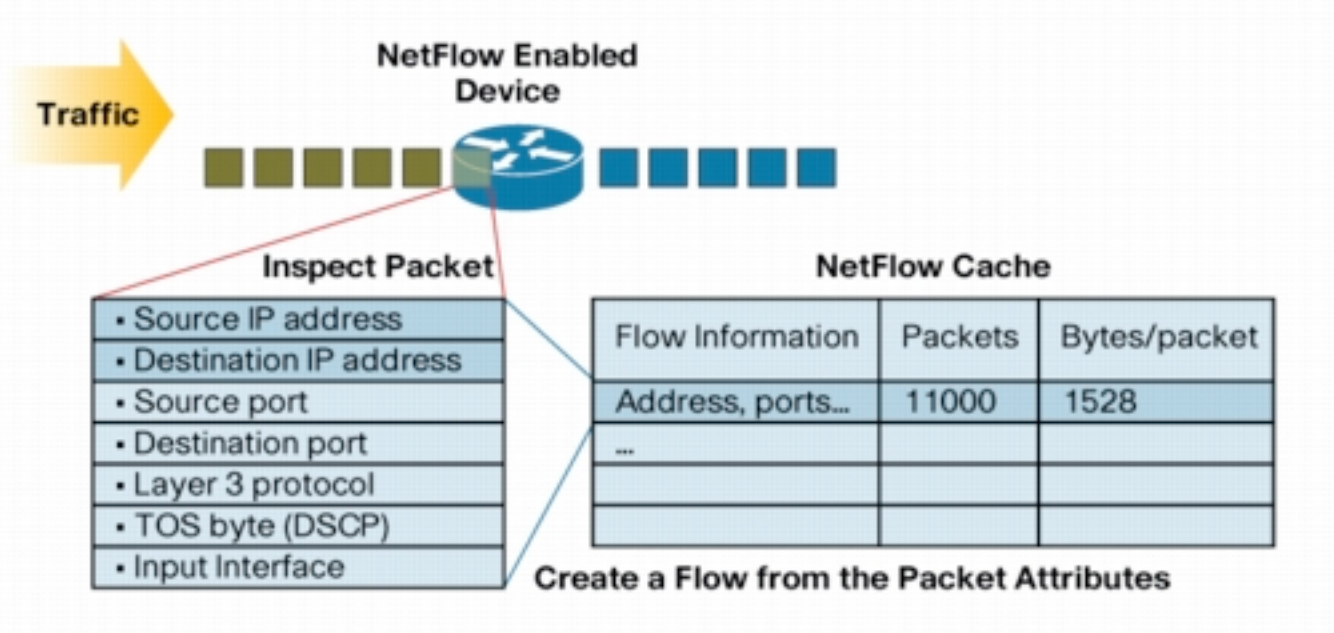
Cisco NetFlow Versions

- Each version has its own packet format
 - v1, 5,6,7,8 have a fixed/closed, specified format.
 - v9 format is dynamic and open to extensions.
- Sequence Numbers:
 - v1 does not have sequence numbers (no way to detect lost flows).
 - v5,6,7,8 have flow sequence numbers (i.e. keep track of the number of emitted flows).
 - v9 has packet (not flow) sequence number (i.e. easy to know the number of lost packets but not of lost flows).
- The “version” defines what type of data is in the flow.
- Some versions (e.g. v7) specific to Catalyst platform.

NetFlow: Flow Birth and Death [1/5]

- Each packet that is forwarded within a router or L3 switch is examined for a set of IP packet attributes.
- All packets with the same source/destination IP address, source/destination ports, protocol interface are grouped into a flow and then packets and bytes tallied.
- Active flows are stored in memory in the so-called NetFlow cache.

NetFlow: Flow Birth and Death [2/5]



NetFlow: Flow Birth and Death [3/5]

Flows are terminated when one of these conditions are met:

- The network communication has ended (e.g. a packet contains the TCP FIN flag).
- The flow lasted too long (default 30 min).
- The flow has been not active (i.e. no new packets have been received) for too long (default 15 sec).
- The flow cache was full and the cache manager had to purge data.

Note that the flow cache has a limited size, hence it's often not possible to accommodate all flows.

NetFlow: Flow Birth and Death [4/5]

1. Flow Cache—The First Unique Packet Creates a Flow

SrcIfl	SrcIPadd	DstIfl	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	162	/24	5	163	/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	161	/24	180	10	/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. Flow Aging Timers

- Inactive Flow (15 sec is default)
- Long Flow (30 min (1800 sec) is default)
- Flow ends by RST or FIN TCP Flag

SrcIfl	SrcIPadd	DstIfl	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

3. Flows Packaged in Export Packet

Non-Aggregated Flows—Export Version 5 or 9

4. Transport Flows to Reporting Server



NetFlow: Flow Birth and Death [5/5]

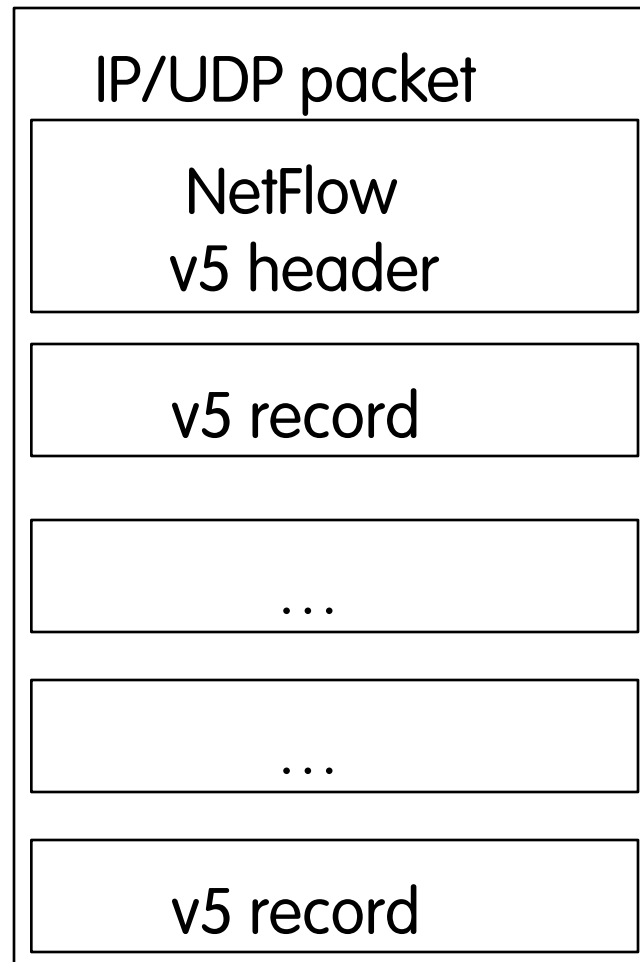
- The NetFlow cache is constantly filling with flows and software in the router or switch is searching the cache for flows that have terminated or expired and these flows are exported to the NetFlow collector server.
- The consequence is that network flows can be split in several netflow flows that, if necessary, are reassembled by the flow collector.

For more info see Cisco White Paper: "Introduction to

NetFlow Packet Format

- Common header among export versions.
- Version specific data field where N records of data type are exported.
- N is determined by the size of the flow definition (e.g. N=30 for v5). Packet size is kept under ~1480 bytes. No fragmentation on Ethernet.

Cisco NetFlow v5 [1/3]



Cisco NetFlow v5 [2/3]

```
struct netflow5_record {  
    struct flow_ver5_hdr flowHeader;  
    struct flow_ver5_rec flowRecord[30];  
} NetFlow5Record;
```

```
struct flow_ver5_hdr {  
    u_int16_t version;           /* Current version=5*/  
    u_int16_t count;            /* The number of records in PDU. */  
    u_int32_t sysUptime;        /* Current time in msec since router booted */  
    u_int32_t unix_secs;       /* Current seconds since 0000 UTC 1970 */  
    u_int32_t unix_nsecs;      /* Residual nanoseconds since 0000 UTC 1970 */  
    u_int32_t flow_sequence;    /* Sequence number of total flows seen */  
    u_int8_t engine_type;       /* Type of flow switching engine (RP,VIP,etc.)*/  
    u_int8_t engine_id;        /* Slot number of the flow switching engine */  
};
```

Cisco NetFlow v5 [3/3]

```
struct flow_ver5_rec {
    u_int32_t srcaddr;           /* Source IP Address */
    u_int32_t dstaddr;          /* Destination IP Address */
    u_int32_t nexthop;          /* Next hop router's IP Address */
    u_int16_t input;            /* Input interface index */
    u_int16_t output;           /* Output interface index */
    u_int32_t dPkts;            /* Packets sent */
    u_int32_t dOctets;          /* Octets sent */
    u_int32_t First;            /* SysUptime at start of flow */
    u_int32_t Last;             /* and of last packet of the flow */
    u_int16_t srcport;           /* TCP/UDP source port number (e.g, FTP, Telnet, etc.,or equivalent) */
    u_int16_t dstport;          /* TCP/UDP destination port number (e.g, FTP, Telnet, etc.,or equivalent) */
    u_int8_t pad1;              /* pad to word boundary */
    u_int8_t tcp_flags;         /* Cumulative OR of tcp flags */
    u_int8_t prot;              /* IP protocol, e.g., 6=TCP, 17=UDP, etc... */
    u_int8_t tos;               /* IP Type-of-Service */
    u_int16_t src_as;           /* source peer/origin Autonomous System */
    u_int16_t dst_as;           /* dst peer/origin Autonomous System */
    u_int8_t src_mask;          /* source route's mask bits */
    u_int8_t dst_mask;          /* destination route's mask bits */
    u_int16_t pad2;            /* pad to word boundary */
};
```

NetFlow v5 Flow Example [1/2]

Cisco NetFlow

Version: 5

Count: 30

SysUptime: 1518422100

Timestamp: May 7, 1993 08:49:48.995294598

CurrentSecs: 736757388

CurrentNSecs: 995294598

FlowSequence: 9751

EngineType: 0

EngineId: 0

SampleRate: 0

pdu 1/30

NetFlow v5 Flow Example [2/2]

pdu 1/30

SrcAddr: 10.16.237.114 (10.16.237.114)

DstAddr: 213.92.16.87 (213.92.16.87)

NextHop: 10.158.100.1 (10.158.100.1)

InputInt: 4

OutputInt: 1

Packets: 5

Octets: 627

StartTime: 1518415.920000000 seconds

EndTime: 1518416.352000000 seconds

SrcPort: 3919

DstPort: 80

padding

TCP Flags: 0x1b

Protocol: 6

IP ToS: 0x00

SrcAS: 0

DstAS: 0

SrcMask: 16 (prefix: 10.16.0.0/16)

DstMask: 0 (prefix: 0.0.0.0/0)

Why Do We Need NetFlow v9?

- Fixed formats (v1-v8) for export are:
 - Easy to implement.
 - Consume little bandwidth.
 - Easy to decipher at the collector.
 - Not flexible (many proprietary hacks such as using TCP/UDP ports for transporting ICMP type/code).
 - Not extensible (no way to extend the flow unless a new version is defined).
 - Some features are missing: L2, VLAN, IPv6, MPLS

NetFlow v9 Principles [1/2]

- Open protocol defined by Cisco (i.e. it's not proprietary) defined in RFC 3954.
- Flow Template + flow record
 - Template composed of type and length.
 - Flow record composed of template ID and value.
 - Templates are sent periodically and they are a prerequisite for decoding flow records.
 - Flow records contain the 'flow meat'.
- Options templates + option records contain probe configuration (e.g. sampling rate, interface packet

NetFlow v9 Principles [2/2]

- Push model probe -> collector (as with past versions).
- Send the templates regularly: each X flows, each X seconds.
- Independent of the underlying protocol, ready for any reliable protocol.
- Can send both template and flow record in one export.
- Can interleave different flow records in one export packet.

Some v9 Tags [1/4]

[1] %IN_BYTES	Incoming flow bytes
[2] %IN_PKTS	Incoming flow packets
[3] %FLOWS	Number of flows
[4] %PROTOCOL	IP protocol byte
[5] %SRC_TOS	Type of service byte
[6] %TCP_FLAGS	Cumulative of all flow TCP flags
[7] %L4_SRC_PORT	IPv4 source port
[8] %IPV4_SRC_ADDR	IPv4 source address
[9] %SRC_MASK	Source subnet mask (/<bits>)
[10] %INPUT_SNMP	Input interface SNMP idx
[11] %L4_DST_PORT	IPv4 destination port
[12] %IPV4_DST_ADDR	IPv4 destination address
[13] %DST_MASK	Dest subnet mask (/<bits>)

Some v9 Tags [2/4]

[16] %SRC_AS	Source BGP AS
[17] %DST_AS	Destination BGP AS
[21] %LAST_SWITCHED	SysUptime (msec) of the last flow pkt
[22] %FIRST_SWITCHED	SysUptime (msec) of the first flow pkt
[23] %OUT_BYTES	Outgoing flow bytes
[24] %OUT_PKTS	Outgoing flow packets
[27] %IPV6_SRC_ADDR	IPv6 source address
[28] %IPV6_DST_ADDR	IPv6 destination address
[29] %IPV6_SRC_MASK	IPv6 source mask
[30] %IPV6_DST_MASK	IPv6 destination mask
[32] %ICMP_TYPE	ICMP Type * 256 + ICMP code
[34] %SAMPLING_INTERVAL	Sampling rate

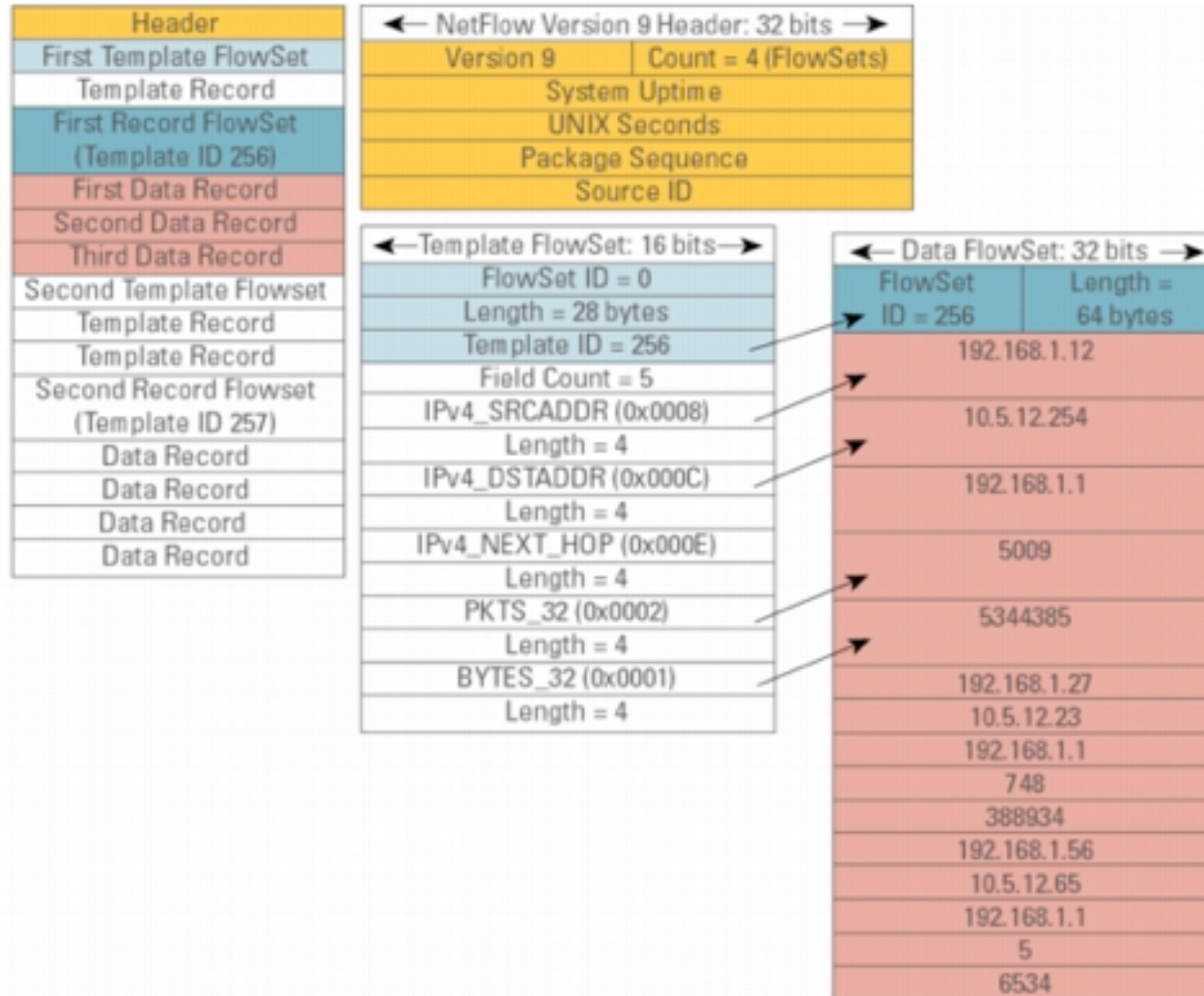
Some v9 Tags [3/4]

[37] %FLOW_INACTIVE_TIMEOUT	Inactivity timeout of flow cache entries
[38] %ENGINE_TYPE	Flow switching engine
[39] %ENGINE_ID	Id of the flow switching engine
[40] %TOTAL_BYTES_EXP	Total bytes exported
[41] %TOTAL_PKTS_EXP	Total flow packets exported
[42] %TOTAL_FLOWS_EXP	Total number of exported flows
[56] %IN_SRC_MAC	Source MAC Address
[57] %OUT_DST_MAC	Destination MAC Address
[58] %SRC_VLAN	Source VLAN
[59] %DST_VLAN	Destination VLAN
[60] %IP_PROTOCOL_VERSION	[4=IPv4][6=IPv6]

Some v9 Tags [4/4]

[70] %MPLS_LABEL_1	MPLS label at position 1
[71] %MPLS_LABEL_2	MPLS label at position 2
[72] %MPLS_LABEL_3	MPLS label at position 3
[73] %MPLS_LABEL_4	MPLS label at position 4
[74] %MPLS_LABEL_5	MPLS label at position 5
[75] %MPLS_LABEL_6	MPLS label at position 6
[76] %MPLS_LABEL_7	MPLS label at position 7
[77] %MPLS_LABEL_8	MPLS label at position 8
[78] %MPLS_LABEL_9	MPLS label at position 9
[79] %MPLS_LABEL_10	MPLS label at position 10
[80] %IN_DST_MAC	Source MAC Address
[81] %OUT_SRC_MAC	Destination MAC Address

v9 Flow Format



NetFlow v9 Flow Example [1/2]

Cisco NetFlow

```
Version: 9
Count: 4
SysUptime: 1132427188
Timestamp: Aug 18, 2000 23:49:25.000012271
    CurrentSecs: 966635365
FlowSequence: 12271
SourceId: 0
FlowSet 1/4
```

FlowSet 1/4

```
    Template FlowSet: 0
    FlowSet Length: 164
    Template Id: 257
    Field Count: 18
    Field (1/18)
        Type: LAST_SWITCHED (21)
        Length: 4
```

NetFlow v9 Flow Example [2/2]

Cisco NetFlow

Version: 9

Count: 1

SysUptime: 1133350352

Timestamp: Aug 19, 2000 00:04:48.000012307

CurrentSecs: 966636288

FlowSequence: 12307

SourceId: 0

FlowSet 1/1

Data FlowSet (Template Id): 257

FlowSet Length: 52

pdu 1

EndTime: 1133334.000000000 seconds

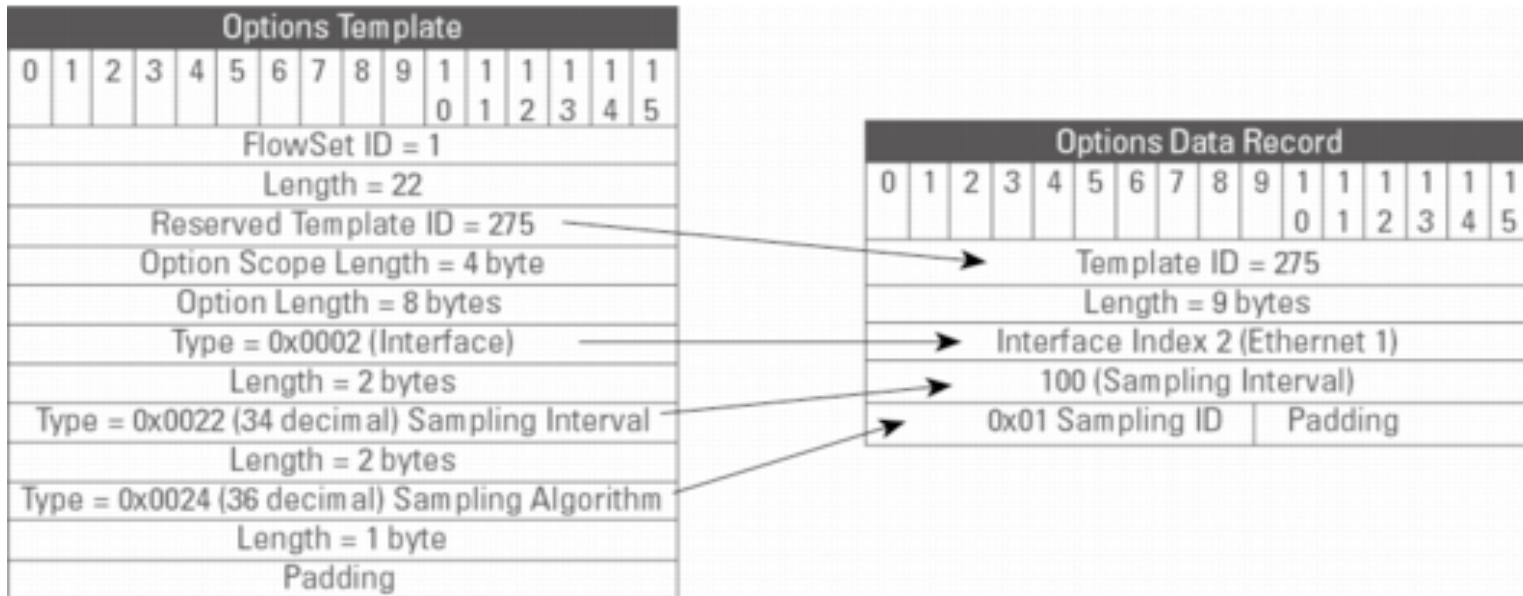
StartTime: 1133334.000000000 seconds

Octets: 84

Packets: 1

InputInt: 15

V9 Options Template



NetFlow v5 vs. v9

	v5	v9
Flow Format	Fixed	User Defined
Extensible	No	Yes (Define new FlowSet Fields)
Flow Type	Unidirectional	Bidirectional
Flow Size	48 Bytes (fixed)	It depends on the format
IPv6 Aware	No	IP v4/v6
MPLS/VLAN	No	Yes

Cisco IOS Configuration [1/2]

- Configured on each input interface.
- Define the version.
- Define the IP address of the collector (where to send the flows).
- Optionally enable aggregation tables.
- Optionally configure flow timeout and main (v5) flow table size.
- Optionally configure sample rate.

Cisco IOS Configuration [2/2]

```
interface FastEthernet0/0/0
  ip address 10.0.0.1 255.255.255.0
  no ip directed-broadcast
  ip route-cache flow
```

```
interface ATM1/0/0
  no ip address
  no ip directed-broadcast
  ip route-cache flow
```

```
interface Loopback0
  ip address 10.10.10.10 255.255.255.255
  no ip directed-broadcast
```

```
ip flow-export version 5 origin-as
ip flow-export destination 10.0.0.10 5004
ip flow-export source loopback 0
```

```
ip flow-aggregation cache prefix
  export destination 10.0.0.10 5555
  enabled
```

Cisco IOS Reporting [1/5]

```
krc4#sh ip flow export
Flow export is enabled
  Exporting flows to 10.0.0.10 (5004)
  Exporting using source IP address 10.10.10.10
  Version 5 flow records, origin-as
  Cache for prefix aggregation:
    Exporting flows to 10.0.0.10 (5555)
    Exporting using source IP address 10.10.10.10
3176848179 flows exported in 105898459 udp datagrams
0 flows failed due to lack of export packet
45 export packets were sent up to process level
0 export packets were punted to the RP
5 export packets were dropped due to no fib
31 export packets were dropped due to adjacency issues
0 export packets were dropped due to fragmentation failures
0 export packets were dropped due to encapsulation fixup failures
0 export packets were dropped enqueueing for the RP
0 export packets were dropped due to IPC rate limiting
0 export packets were dropped due to output drops
```

Cisco IOS Reporting [2/5]

```
krc4#sho ip ca fl
```

```
IP packet size distribution (106519M total packets):
```

```
  1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
    .002 .405 .076 .017 .011 .010 .007 .005 .004 .005 .004 .004 .003 .002 .002

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
    .002 .006 .024 .032 .368 .000 .000 .000 .000 .000 .000
```

```
IP Flow Switching Cache, 4456704 bytes
```

```
 36418 active, 29118 inactive, 3141073565 added
```

```
3132256745 age 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	2951815	0.6	61	216	42.2	26.6	21.4
TCP-FTP	24128311	5.6	71	748	402.3	15.0	26.3
TCP-FTPD	2865416	0.6	916	843	611.6	34.7	19.8
TCP-WWW	467748914	108.9	15	566	1675.8	4.9	21.6
TCP-SMTP	46697428	10.8	14	370	159.6	4.0	20.1
TCP-X	521071	0.1	203	608	24.7	24.5	24.2
TCP-BGP	2835505	0.6	5	94	3.3	16.2	20.7

Cisco IOS Reporting [3/5]

```
krc4#sho ip ca fl
```

TCP-other	1620253066	377.2	47	631	18001.6	27.3	23.4
UDP-DNS	125622144	29.2	2	78	82.5	4.6	24.7
UDP-NTP	67332976	15.6	1	76	22.0	2.7	23.4
UDP-TFTP	37173	0.0	2	76	0.0	4.1	24.6
UDP-Frag	68421	0.0	474	900	7.5	111.7	21.6
UDP-other	493337764	114.8	17	479	1990.3	3.8	20.2
ICMP	243659509	56.7	3	166	179.7	3.3	23.3
IGMP	18601	0.0	96	35	0.4	941.4	8.1
IPINIP	12246	0.0	69	52	0.1	548.4	15.2
GRE	125763	0.0	235	156	6.9	50.3	21.1
IP-other	75976755	17.6	2	78	45.4	3.9	22.8
Total:	3176854246	739.6	33	619	24797.4	16.2	22.6

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
AT5/0/0.4	206.21.162.150	AT1/0/0.1	141.219.73.45	06	0E4B	A029	507
AT4/0/0.10	132.235.174.9	AT1/0/0.1	137.99.166.126	06	04BE	074C	3
AT4/0/0.12	131.123.59.33	AT1/0/0.1	137.229.58.168	06	04BE	09BB	646
AT1/0/0.1	137.99.166.126	AT4/0/0.10	132.235.174.9	06	074C	04BE	3

Cisco IOS Reporting [4/5]

```
Router(config)#ip flow-top-talkers
Router(config-flow-top-talkers)#top 10
```

```
R3#show ip flow top-talkers
```

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
Et1/0	172.16.10.2	Et0/0	172.16.1.84	06	0087	0087	2100
Et1/0	172.16.10.2	Et0/0	172.16.1.85	06	0089	0089	1892
Et1/0	172.16.10.2	Et0/0	172.16.1.86	06	0185	0185	1762
Et1/0	172.16.10.2	Et0/0	172.16.1.86	06	00B3	00B3	2
Et1/0	172.16.10.2	Et0/0	172.16.1.84	06	0050	0050	1
Et1/0	172.16.10.2	Et0/0	172.16.1.85	06	0050	0050	

```
17 of 10 top talkers shown. 7 flows processed.
```

Cisco IOS Reporting [5/5]

```
R3#show ip flow top 10 aggregate destination-address
```

```
There are 3 top talkers:
```

```
IPV4 DST-ADDR bytes pkts flows
```

```
=====
172.16.1.86 160 4 2
172.16.1.85 160 4 2
172.16.1.84 160 4 2
```

```
R3#show ip flow top 10 aggregate destination-address sorted-by bytes match
```

```
source-port min 0 max 1000
```

```
There are 3 top talkers:
```

```
IPV4 DST-ADDR bytes pkts flows
```

```
=====
172.16.1.84 80 2 2
172.16.1.85 80 2 2
172.16.1.86 80 2 26 of 6 flows matched.
```


JunOS Configuration [1/3]

- Sample packets with firewall filter and forward to routing engine.
- Sampling rate is limited to 7000pps (addressed with future PIC).
- Fine for traffic engineering, but restrictive for DoS and intrusion detection.
- Juniper calls NetFlow cflowd (popular collector provided by CAIDA).

JunOS Configuration [2/3]

Firewall filter

```
firewall {
  filter all {
    term all {
      then {
        sample;
        accept;
      }
    }
  }
}
```

Enable sampling / flows

```
forwarding-options {
  sampling {
    input {
      family inet {
        rate 100;
      }
    }
    output {
      cflowd 10.0.0.16 {
        port 2055;
        version 5;
      }
    }
  }
}
```

JunOS Configuration [3/3]

Apply firewall filter to each interface.

```
interfaces {
  ge-0/3/0 {
    unit 0 {
      family inet {
        filter {
          input all;
          output all;
        }
        address 192.148.244.1/24;
      }
    }
  }
}
```

PC-Based NetFlow Probes

- There are some PC-based probes.
- Most of them are based on the pcap library.
- nProbe (www.ntop.org/nProbe.html)
 - Open Source (GPL2)
 - Fastest probe on the market
 - Support of v5/v9, IPFIX.
 - Flexible export format.
 - IPv4/v6 support, flexible template (not even supported by Cisco).
 - Available for both Unix and Windows

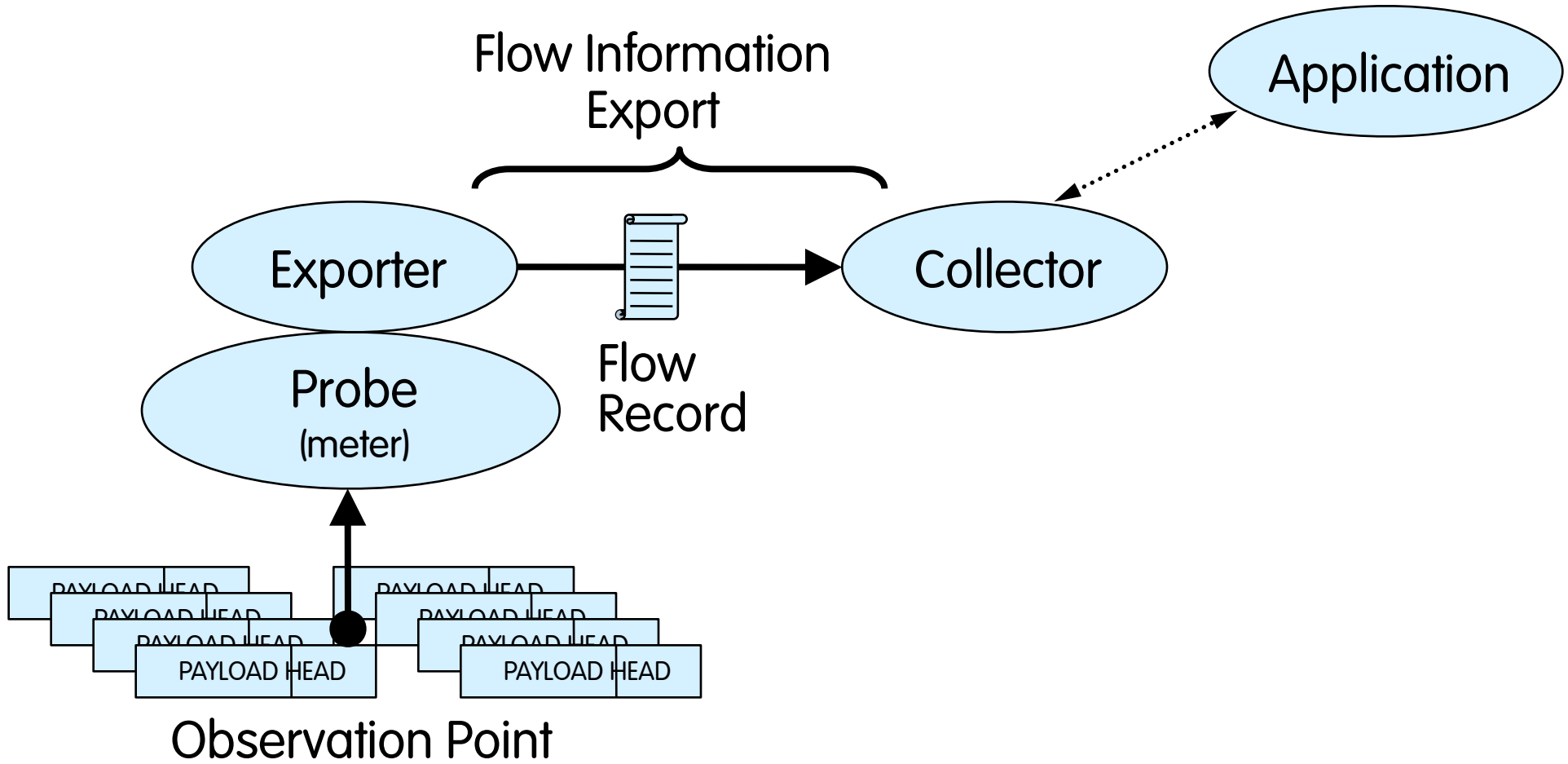
IPFIX Scope and General Requirements

- Goal: Find or develop a basic common IP Traffic Flow measurement technology to be available on (almost) all future routers.
- Fulfilling requirements of many applications.
- Low hardware/software costs.
- Simple and scalable.
- Metering to be integrated in general purpose IP routers and other devices (probes, middle boxes).
- Data processing to be integrated into various applications.
- Interoperability by openness or standardization.

IPFIX in a Nutshell

- Strongly based on NetFlow v9.
- Ability to define new flow fields using a standard format (OID).
- Transport based on SCTP (Stream Control Transport Protocol), optional UDP/TCP support.
- Current status: draft protocol specification.
- Bottom Line: IPFIX = NetFlow v9 over SCTP.

IPFIX Architecture



Flow Aggregation [1/5]

Raw flows are useful but sometimes it's necessary to answer to various questions:

- How much of our traffic is web, news, email, quake?
- How much traffic to/from departments?
- How much traffic to other departments, provider X, Google, etc.?
- Amount of traffic through interface X?

Flow Aggregation [2/5]

Main Active Flow Table

Flow	Source IP	Destination IP	Proto	srcPort	dstPort
1	10.0.0.1	10.0.0.2	TCP	32000	23
2	10.0.0.2	10.0.0.1	TCP	23	32000
3	10.0.0.1	10.0.0.2	ICMP	0	0
4	10.0.0.2	10.0.0.1	ICMP	0	0

Source/Destination IP Aggregation

Flow	Source IP	Destination IP
1	10.0.0.1	10.0.0.2
2	10.0.0.2	10.0.0.1

Flow Aggregation [3/5]

The same flow can be aggregated several times using different criteria. For instance from raw flows it's possible to generate:

- List of protocols
- Conversation matrix (who's talking to who)
- Top TCP/UDP ports

Aggregation flow early can save time/memory with respect to late aggregation (e.g. the conversation matrix is much easier to implement aggregating data on the probe instead of using raw/unaggregated flows).

Flow Aggregation [4/5]

- Flows can be aggregated using “external” criteria and not just based on raw flow fields.
- Usually these external criteria are applied on “key” (not “value”) fields such as port, IP address, protocol etc. and are used to group values together.
- Criteria are added (don’t replace) to existing fields.
- Example: port-map, protocol-map, ip-address

	IP src	IP dst	Proto	Src port	Dst port
Before	10.0.0.1	10.0.0.2	UDP	32000	53
	10.0.0.2	10.0.0.1	TCP	34354	80

	IP src	IP dst	Proto	Src port	Dst port	Src port map	Dst port map
After	10.0.0.1	10.0.0.2	TCP	32000	53	udp_other	domain
	10.0.0.2	10.0.0.1	TCP	34354	80	tcp_other	http

Flow Aggregation [5/5]

- Flows can be aggregated according to:
 - TCP/UDP Port, ToS (Type of Service), Protocol (e.g. ICMP, UDP), AS (Autonomous System)
 - Source/Destination IP Address
 - Subnet, time of the day.
- Aggregation can be performed by the probe, the collector or both.
- Probe aggregation is very effective in terms of resource usage and network flow traffic.
- Collector aggregation is more powerful (e.g. aggregate flows produced by different probes) but rather costly (receive all the aggregate).

Flow Filtering

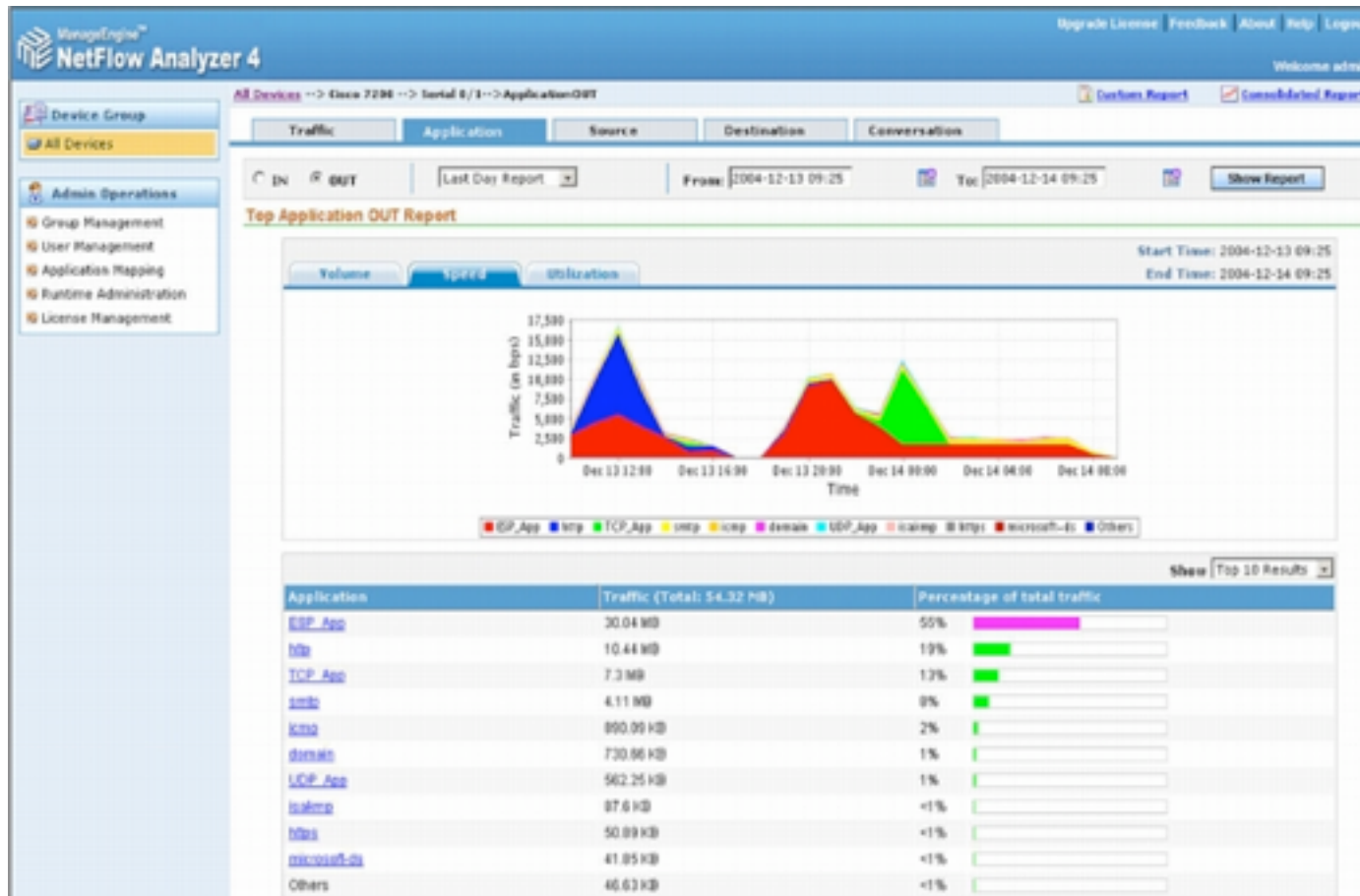
Filtering flows means: discard flows based on some criteria such as

- Flow duration (discard flows that lasted less than X seconds)
- Flow src/dest (ignore flows containing broadcast addresses)
- Flow ports (ignore flows originated by port X)

Note that:

- Filtering != aggregation: they do two different jobs
- Filtering and aggregation can coexist
- Filtering is usually applied before aggregating flows and not after.

NetFlow Traffic Report Example



Flows and Security

NetFlow/IPFIX can be used for security and not just for traffic accounting:

- Portscan/portmap detection
- Detect activities on suspicious ports
- Identify sources of spam, unauthorized servers (e.g. file servers)

Flows and Security: Portmap Scan

Start	SrcIPAddress	SrcP	DstIPAddress	DstP	P	Pkts
10:53:42.50	165.132.86.201	9781	128.146.0.76	111	6	1
10:53:42.54	165.132.86.201	9874	128.146.0.7	111	6	1
10:53:42.54	165.132.86.201	9982	128.146.0.80	111	6	1
10:53:42.54	165.132.86.201	9652	128.146.0.74	111	6	1
10:53:42.54	165.132.86.201	9726	128.146.0.75	111	6	1
10:53:42.54	165.132.86.201	9855	128.146.0.77	111	6	1
10:53:42.58	165.132.86.201	10107	128.146.0.82	111	6	1

Short timeframe, same IP source, different IP targets same port (111=RPC port).

Flows and Security: Backdoor Search

Start	SrcIPAddress	SrcP	DstIPAddress	DstP	P	Pkts
19:08:40	165.132.86.201	8401	128.146.172.232	1524	6	19
19:08:40	165.132.86.201	8422	128.146.172.230	1524	6	16
19:08:40	165.132.86.201	8486	128.146.172.234	1524	6	19
19:08:40	165.132.86.201	8529	128.146.172.236	1524	6	10
19:08:41	165.132.86.201	8614	128.146.172.237	1524	6	16
19:08:41	165.132.86.201	8657	128.146.172.238	1524	6	22

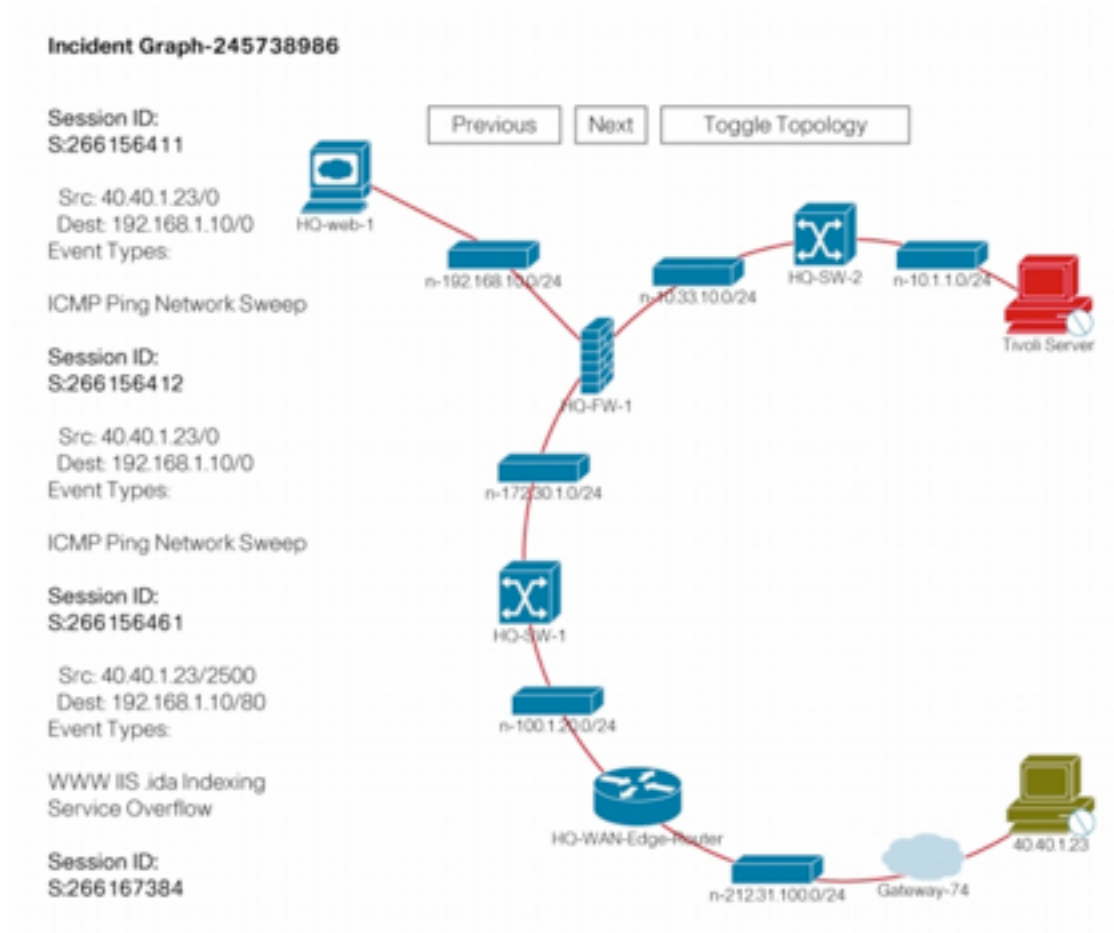
Same as portmap scan, targeting port 1524 (trinoo backdoor port [<http://www.auditmypc.com/port/tcp-port-1524.asp>]).

Flows and Security: Intrusion Detection

Simple flow-based IDS system:

- Flows with excessive octet or packet count (floods).
- IP sources contacting more than N destinations – host scanning.
- IP sources contacting more than M destination ports on a single host (for ports 0-1023) – port scanning.

Incident Report and NetFlow



sFlow

Driving Forces Towards sFlow

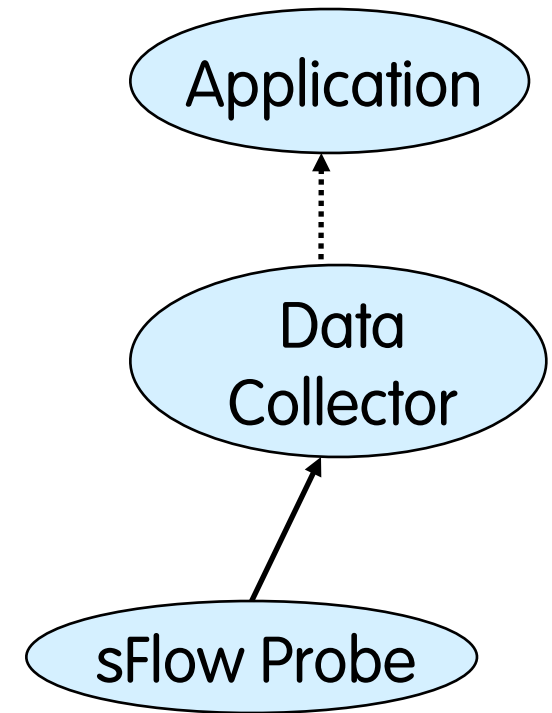
- Cost
 - Monitoring switched networks required multiple expensive probes.
 - Embedded monitoring solutions required extra hardware &/or software.
 - Traffic analysis solutions were costly.
 - Administrative costs of managing additional equipment.
- Impact to network performance
 - Switching performance impacted by measuring traffic flows.
 - Excessive network bandwidth used to export flow data.
- Poor scalability of monitoring system
 - Cannot keep up with Gigabit speeds.
 - Cannot build a network-wide traffic view for large, heavily used networks.

sFlow Principles

- Don't pretend to be as fast as the monitored network: you will lose data anyway.
- Even if you can monitor everything you'll run into trouble handling all the generated flows.
- Analyze 1 packet each X packets (sampling).
- The more packets you analyze the more precise are your traffic reports.
- If the network is too fast for you, sample more!

sFlow Architecture

- The probe samples traffic.
- Sampled packets are sent (in sFlow format) to the collector.
- Periodically the probe sends the collector interface statistics (SNMP MIB-II counters) inside sFlow packets. The packets are used to “scale” traffic.



sFlow Specification [1/3]

- Specified in RFC 3176 (Informational RFC) proposed by InMon Inc.
- It defines:
 - sFlow packets format (UDP, no SNMP).
 - A SNMP MIB per accessing sFlow collected data (<http://support.ipmonitor.com/mibs/SFLOW-MIB/tree.aspx>).
- The sFlow architecture is similar to NetFlow: the probe sends sFlow packets to the collector.

sFlow Specification [2/3]

- The sFlow probe is basically a sniffer that captures 1 out of X packets (default ratio is 1:400).
- Such packets is sent to the collector coded in sFlow format.
- Periodically the probes sends other sFlow packets that contain network interface statistics (e.g. interface traffic counters) used to scale collected data.

sFlow Specification [3/3]

- Using statistical formula it is possible to produce very precise traffic reports.
- % Sampling Error $\leq 196 * \sqrt{1 / \text{number of samples}}$ [<http://www.sflow.org/packetSamplingBasics/>]
- sFlow is scalable (you just need to increase the sampling ration) even on 10 Gb networks or more.
- ntop.org is part of the sFlow.org consortium.



sFlow Packet [1/2]

```
struct sample_datagram_v5 {
    address agent_address      /* IP address of sampling agent,
                               sFlowAgentAddress. */
    unsigned int sub_agent_id; /* Used to distinguishing between datagram
                               streams from separate agent sub entities
                               within an device. */
    unsigned int sequence_number; /* Incremented with each sample datagram
                                   generated by a sub-agent within an
                                   agent. */
    unsigned int uptime;        /* Current time (in milliseconds since device
                               last booted). Should be set as close to
                               datagram transmission time as possible.
                               Note: While a sub-agents should try and
                               track the global sysUptime value
                               a receiver of sFlow packets must
                               not assume that values are
```

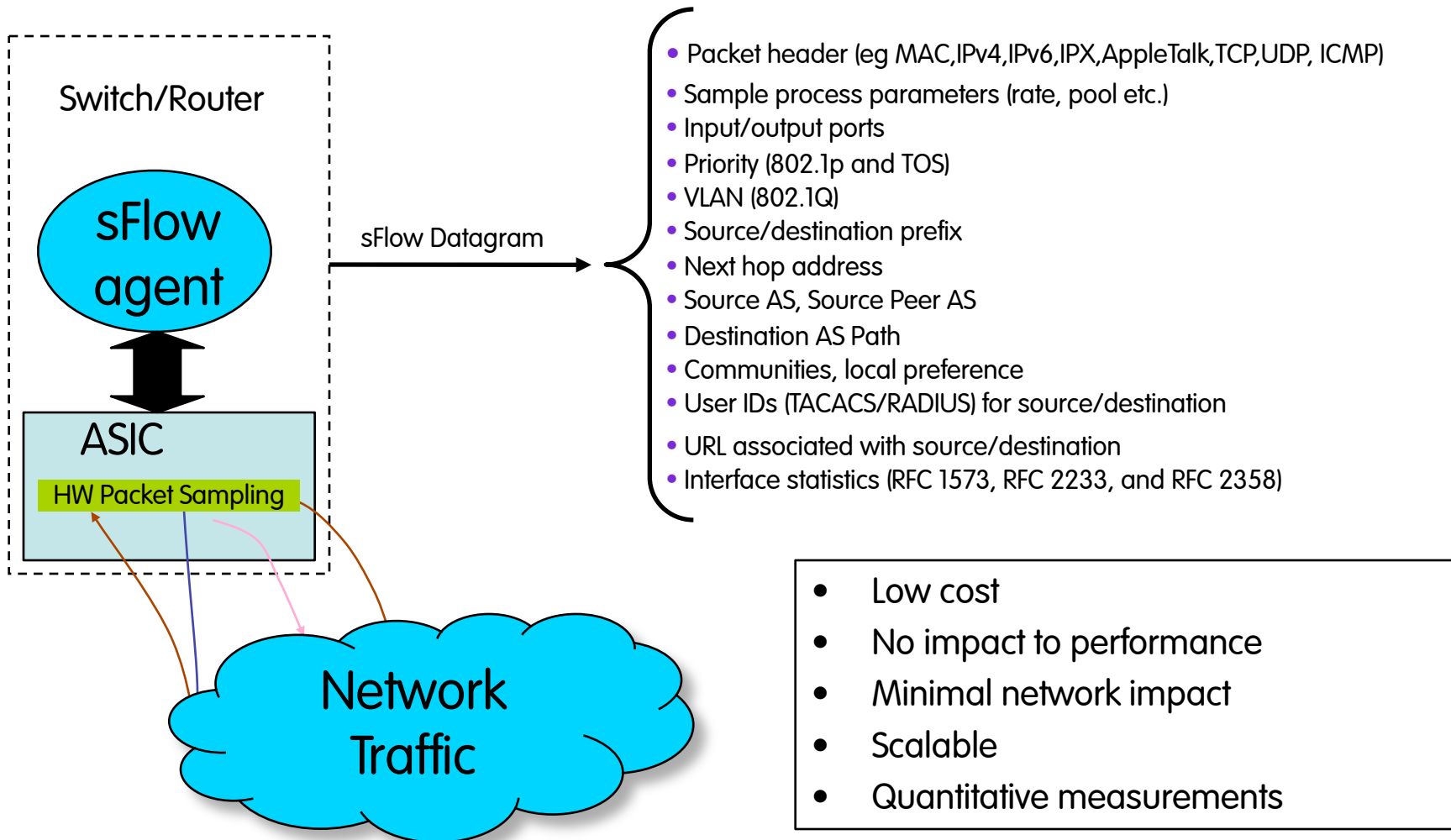
sFlow Packet [2/2]

```
struct flow_sample {
    unsigned int sequence_number; /* Incremented with each flow sample
                                   generated by this source_id.
                                   Note: If the agent resets the
                                   sample_pool then it must
                                   also reset the sequence_number.*/

    sflow_data_source source_id; /* sFlowDataSource */
    unsigned int sampling_rate; /* sFlowPacketSamplingRate */
    unsigned int sample_pool; /* Total number of packets that could have
                               been sampled (i.e. packets skipped by
                               sampling process + total number of
                               samples) */

    unsigned int drops; /* Number of times that the sFlow agent
                        detected that a packet marked to be
                        sampled was dropped due to
                        lack of resources. The drops counter
                        reports the total number of drops
                        detected since the agent was last reset. */
};
```

sFlow Summary



Integrated Network Monitoring

Traffic Analysis & Accounting Solutions

sFlow enabled switches



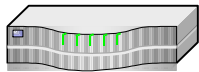
Core network switches

RMON enabled switches



L2/L3 Switches

NetFlow enabled routers



sFlow

RMON

NetFlow



- Network-wide, continuous surveillance
 - 20K+ ports from a single point
- Timely data and alerts
 - Real-time top talkers
 - Site-wide thresholds and alarms
- Consolidated network-wide historical usage data

sFlow vs. NetFlow

	sFlow	NetFlow
Native Environment	Switched	Routed
Operational Speed	Multigigabit	1 Gbit or less
Sampling	Always	Sometimes
Monitoring	Statistical	Accurate (without packet loss)

Radius [RFC 2139, 1997]

Radius is acronym for Remote Authentication Dial In User Service (RADIUS) specified in the following RFC:

- Authentication Protocol

Rigney, C., Rubens, A., Simpson, W, and Willens, S.; Remote Authentication Dial In User Service (RADIUS), RFC 2138, January 1997.

- Data Accounting

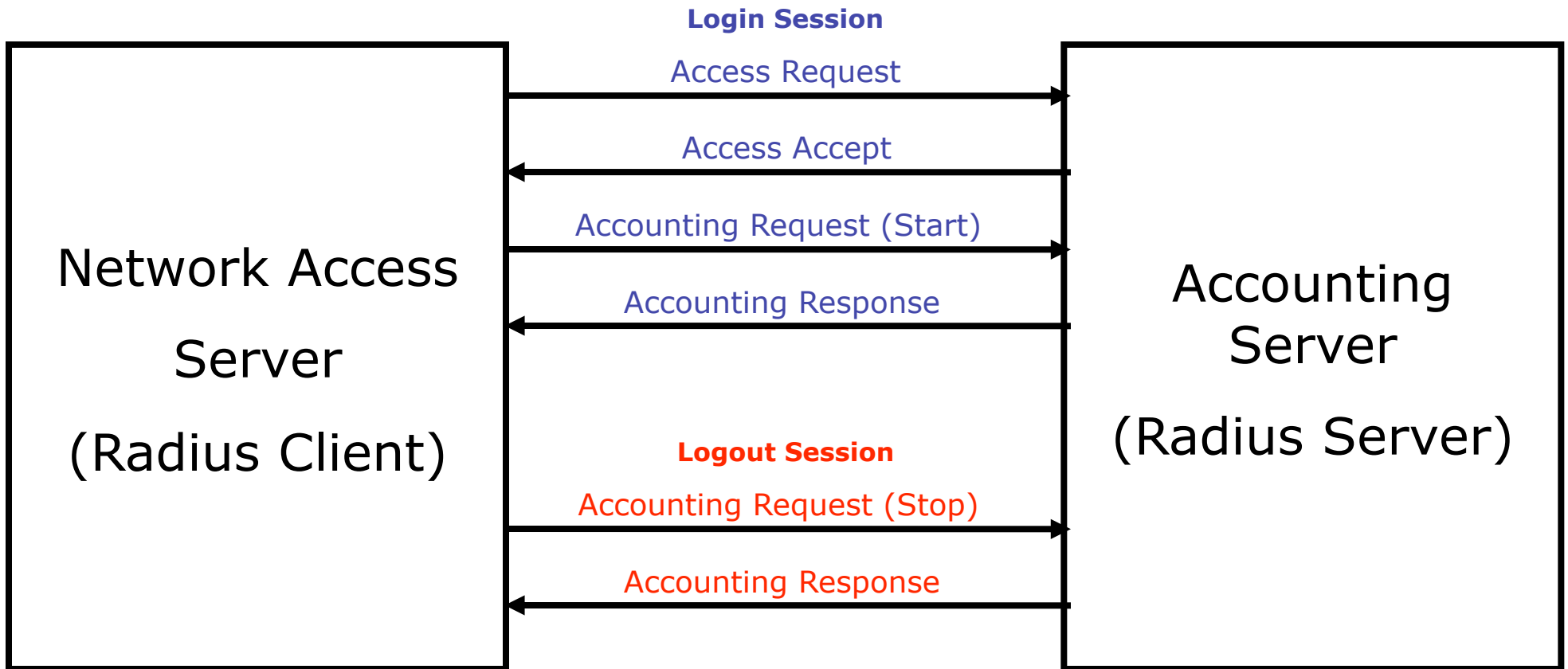
Rigney, C.; RADIUS Accounting, RFC 2139, January 1997.

Radius

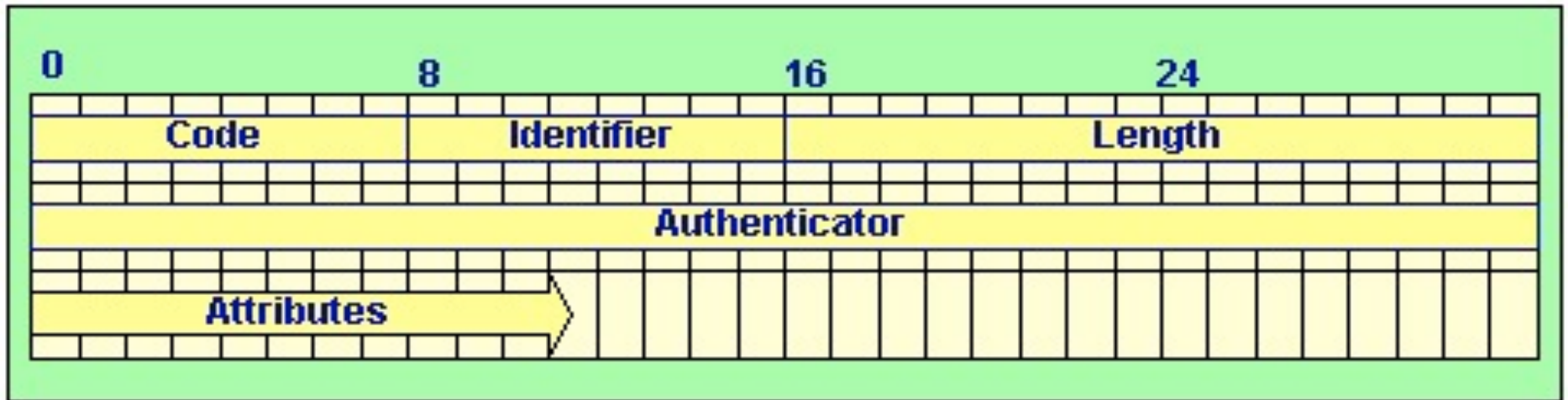
Radius is important because:

- It is the most used protocol for implementing authentication on network devices.
- Used for billing activities on wired lines (e.g. ADSL, Modem).
- Allows accounting for connection duration or data volumes.
- Supported by all the network devices (Low-end excluded).

The Radius Protocol



Radius Protocol: Messages



- *Code*: Byte containing command/reply RADIUS.
- *Identifier*: Byte identifies command/reply RADIUS.
- *Length*: Packet length.
- *Authenticator*: Value used for authenticating RADIUS server reply.
- *Attributes*: Attributes of command/reply.

Radius Protocol: Primitives

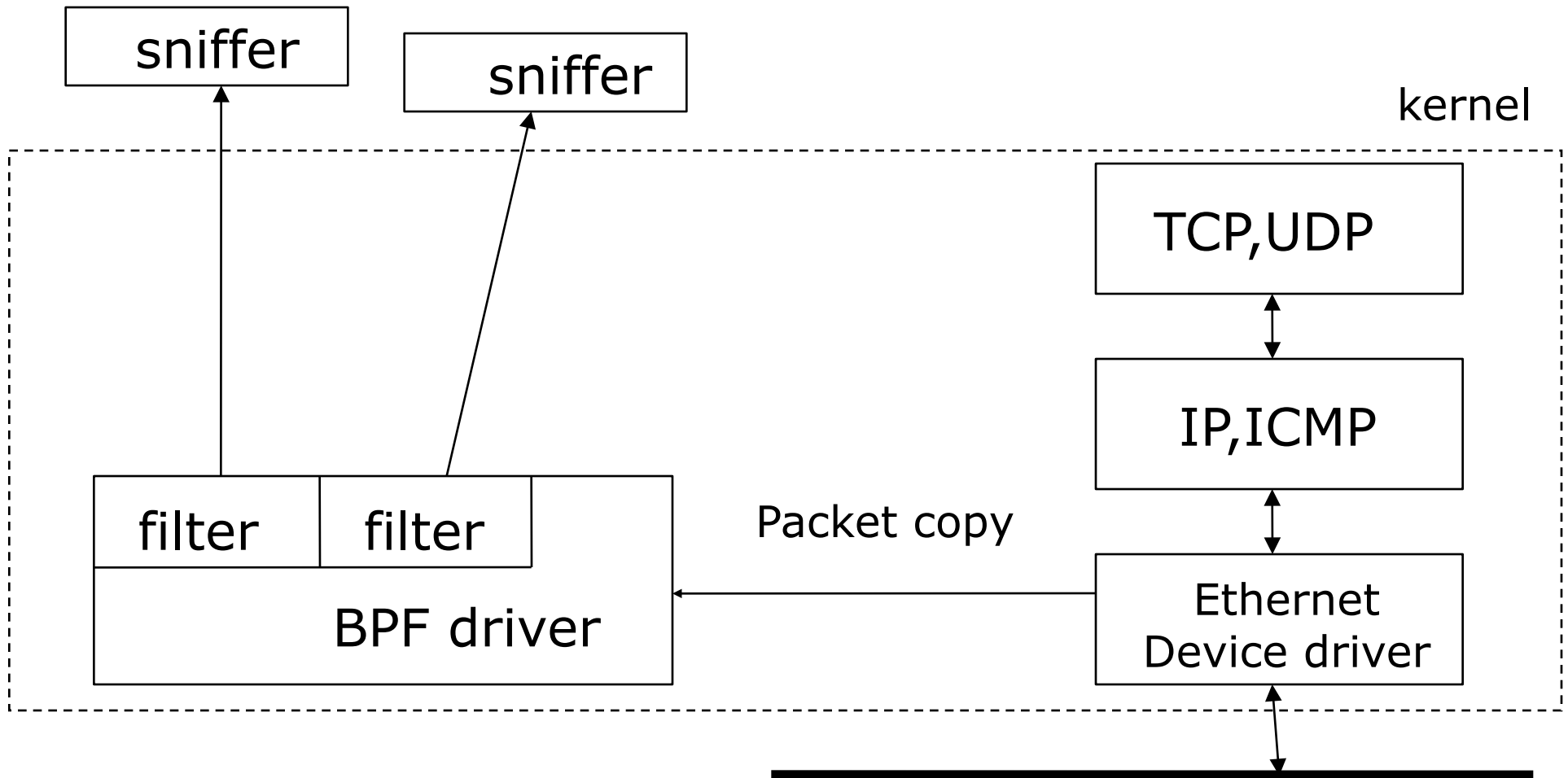
access-request, (client->server):

- Request to access to network services (e.g. user authentication).
- Possible reply:
 - access-accept, (server->client).
 - access-reject, (server->client).
 - access-challenge, (server->client): used for CHAP authentication.

accounting request, (client->server)

- Request to write accounting data on the accounting server.
- Replies:
 - accounting response, (server->client)

Packet Capture: libpcap



Libpcap: Usage Example [1/2]

```
pcapPtr = pcap_open_live(deviceName,  
    maxCaptureLen, setPromiscuousMode,  
    pktDelay, errorBuffer);  
  
while (pcap_dispatch(pcapPtr, 1,  
    processPacket, NULL) != -1);  
  
void processPacket(u_char *_deviceId,  
    const struct pcap_pkthdr *h,  
    const u_char *p) {  
    ...  
}
```

See also: <http://jnetpcap.sourceforge.net/>

Libpcap: Usage Example [2/2]

```
int main(int argc, char* argv[]) {
    /* open a network interface */
    descr = pcap_open_live(dev, BUFSIZ, 0, 1, errbuf);

    /* install a filter */
    pcap_compile(descr, &fp, "dst port 80", 0, netp);
    pcap_setfilter(descr, &fp);

    while (1) {
        /* Grab packets forever */
        packet = pcap_next(descr, &hdr);
        /* print its length */
        printf("Grabbed packet of length %d\n", hdr.len);
    }
}
```

Common Problems with Pkt Capture

- Security issues
 - All the network traffic is captured and not just the one destined to the sniffing host
 - If there is a switched network it is captured only a part of traffic (ARP poisoning)
 - Usability limited to who have root capabilities
NOTE: this append also with ICMP command (e.g. ping) and therefore they are setted with the setuid.
- Performance
 - Sniffer implies also the cpu load because all the captured packets must be analyzed by the program and not just those directed to the host

Traffic Mirror: possible solutions

Hardware:

- Hub (Copper Ethernet, Token Ring)
- Optical Splitter (Optical Fibers)
- Tap (Copper/Fiber)

Software:

- Switch Port Mirror (1:1, 1:N)
- Switch VLAN Mirror (N:1)
- Switch Traffic Filter/Mirroring (Juniper)

Network Taps



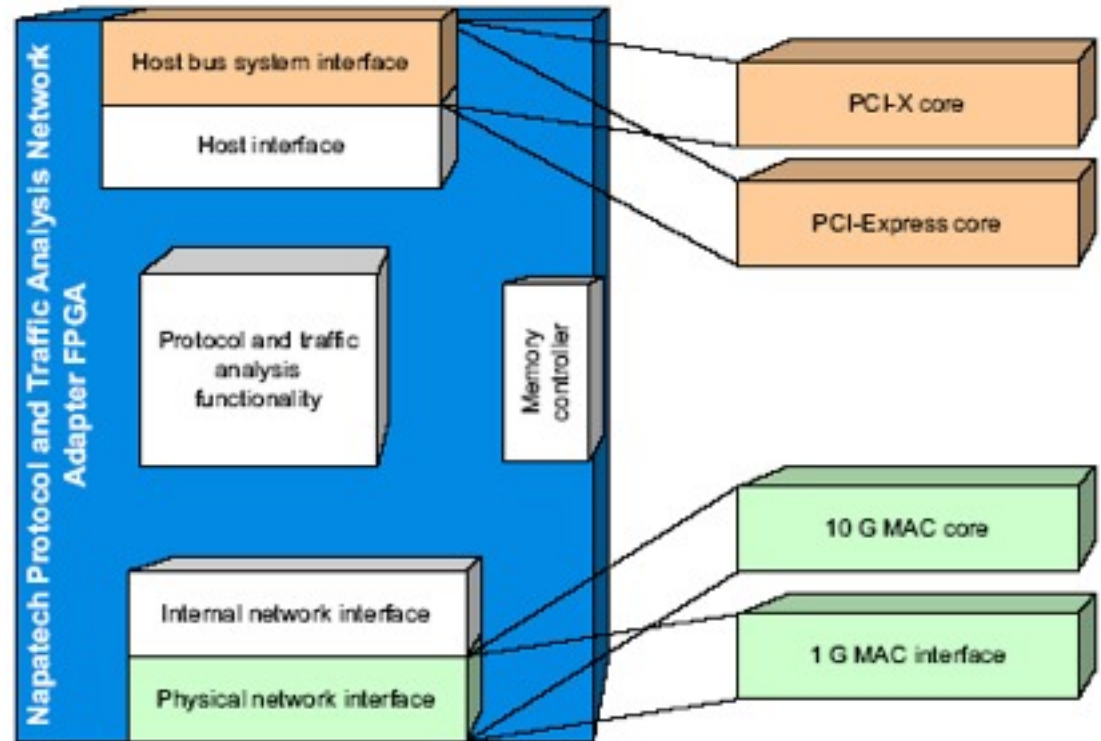
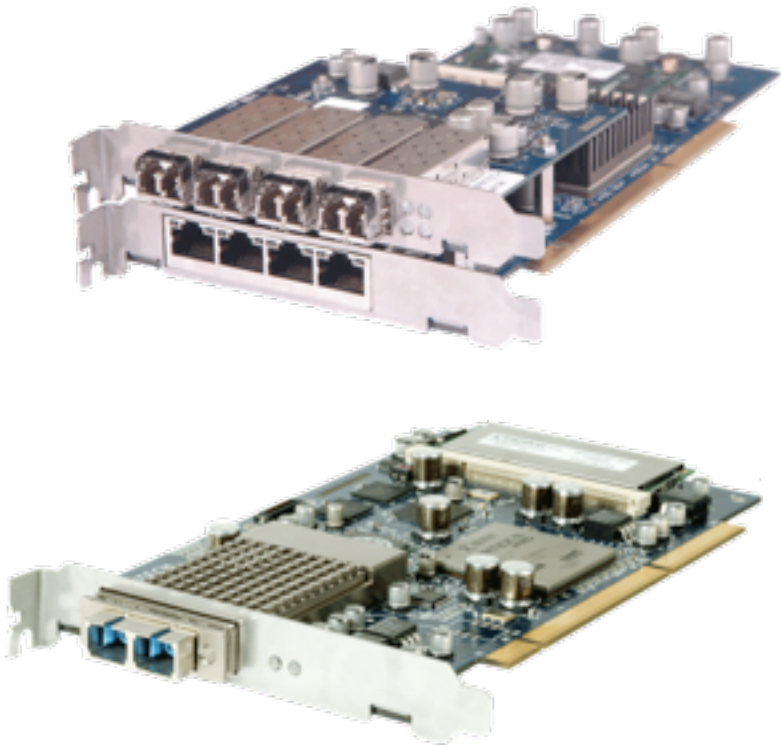
Packet Capture: Solutions

1. Use of NICs that feature an NPU (Network Process Unit). Every modern NIC has a limited NPU (multicast and ethernet).
2. Use a programmable card (e.g. Napatech)
3. Execution of traffic accounting/management code directly on the NIC (e.g. Intel IPX Family)
4. High-speed access (via mmap()) to packets directly on the NIC via the PCI bus (eg. Endace DAG Card)

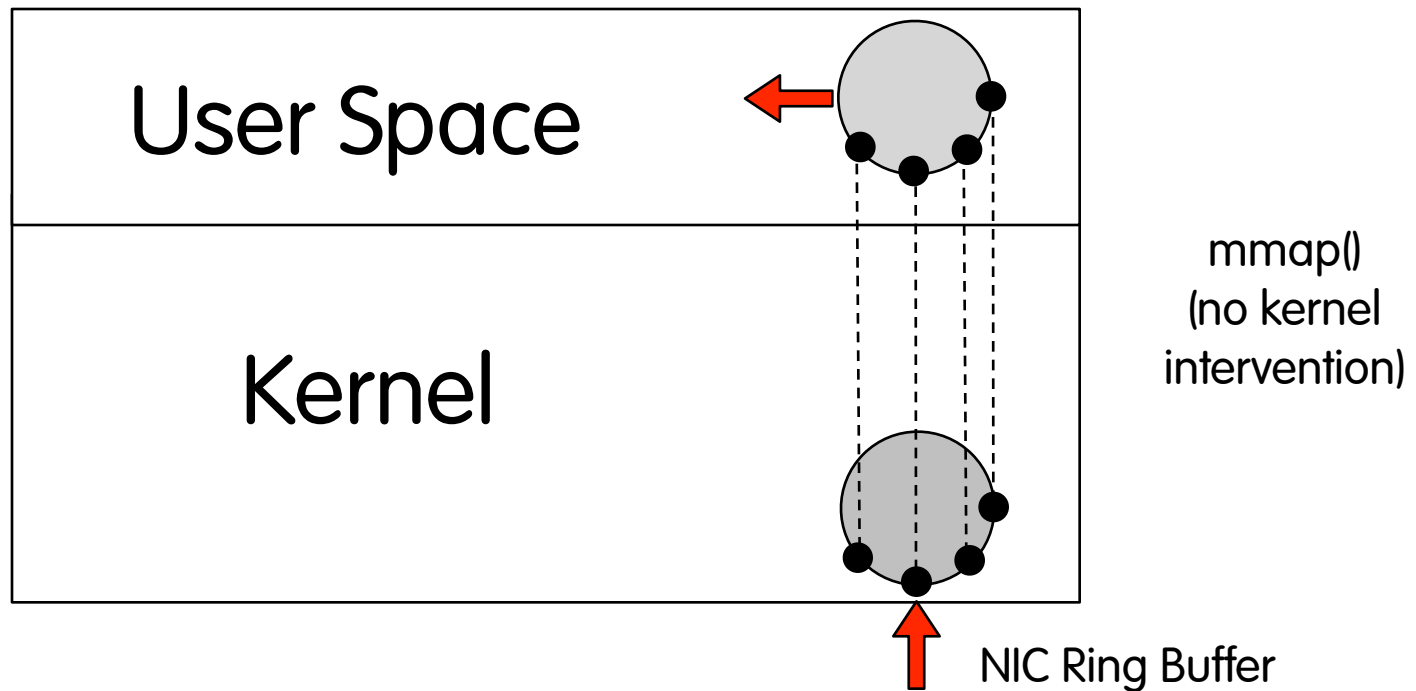
Packet Capture: DAG Card



Packet Capture: Napatech Cards



Packet Capture: PF_RING



Data Collection: RRD

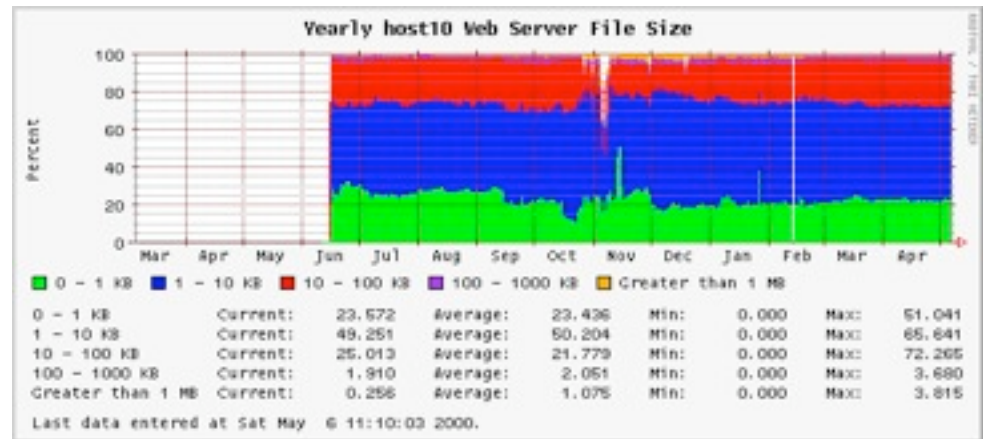
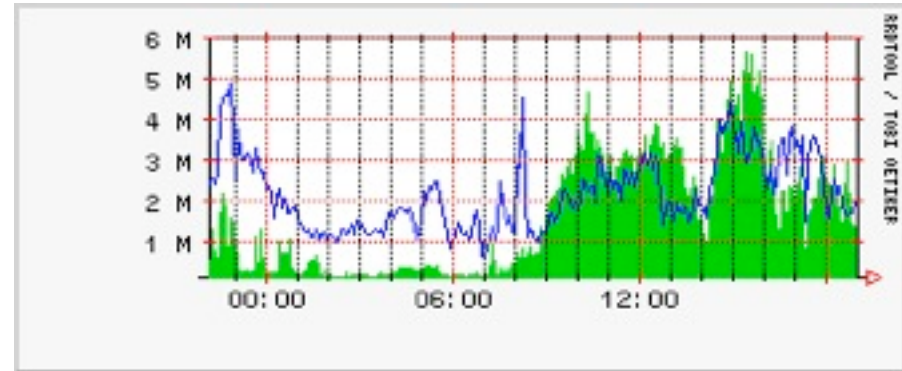
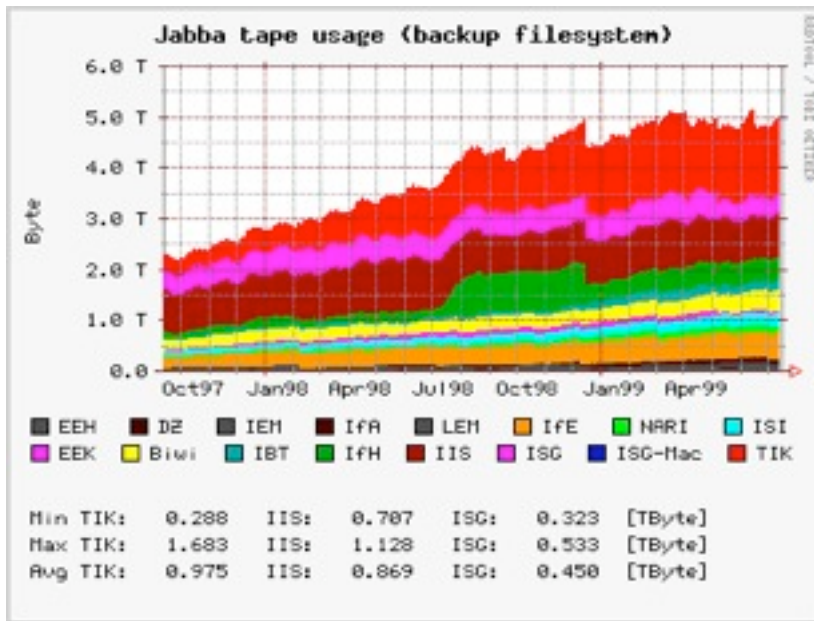
- RRD [<http://www.rrdtool.org/>]
 - Round Robin Database: Tool used to store and display data on which is based MRTG on.
 - Data are stored in “compress” format and they don’t grow with time (automatic data aggregation) and always equal file size.
 - Perl/C interface to access to data and produce graphs.

RRD Perl Example

```
$rrd = "$dataDir/$agent-$ifIndex.rrd";
if(! -e $rrd) {
    RRDs::create ($rrd, "--start", $now-1, "--step", 20,
        "DS:bytesIn:COUNTER:120:0:10000000",
        "DS:bytesOut:COUNTER:120:0:10000000",
        "RRA:AVERAGE:0.5:3:288");
    $ERROR = RRDs::error;
    die "$0: unable to create `'$rrd': $ERROR\n" if $ERROR;
}

RRDs::update $rrd, "$now:$ifInOctets:$ifOutOctets";
if ($ERROR = RRDs::error) {
    die "$0: unable to update `'$rrd': $ERROR\n";
}
```


Data Collection: RRD Graphs



5. Traffic Measurement: Some Case Studies

Path Characterization: Patchar

- Pathchar [<ftp://ftp.ee.lbl.gov/pathchar/>]
 - Send various packets (with different sizes) to all routers of a certain route that needs to be analyzed
 - It measures minimum response time and for each hop:
 - Hop delay
 - Bandwidth
 - Queueing
 - Studying the RTT with respect to packet size , it measures the available bandwidth
 - Drawbacks: extensive test duration, complex calculations due to the large number of probing packets that need to be used for precise measurements.
- Additional tools: pchar, pipechar.

Network Throughput: Iperf

- Iperf [<http://dast.nlanr.net/Projects/Iperf/>]
 - Client/server Architecture: the same binary is started in two different ways.
 - The client application sends to server TCP/UDP packets. It can specify the port, test duration, TCP window, test volume size.
 - Statistics: bandwidth, packet delay/loss, jitter.
 - Disadvantages:
 - The server application must be installed on the destination host.
 - The tool can't be installed/used on routers.

What Traffic Reports Do We Need? [1/2]

- Top N talkers (those who transmit most traffic).
- Top N conversations (the host pairs that transmit most traffic between each other).
- Top N Applications (e.g. SAP is using 70% of the available bandwidth).
- Data volume per entity basis (link, location, region, class of users).

What Traffic Reports Do We Need? [2/2]

- Data volume and rates per AS (e.g. do we need to sign a new peering contract ?).
- QoS marking per application or entity basis (e.g. does BGP reports us that we're sending the traffic on the optimal path ?).
- Reports about traffic we don't expect to see on the network (e.g. why host X is sending IPX packets although we speak pure IP ?).

Integrated Monitoring: Cacti

Cacti [<http://www.raxnet.net/products/cacti>] is an open-source tool able to:

- Collect data by means of SNMP and other non-SNMP methods.
- Configuration performed via web and stored in a MySQL database.
- Traffic statistics saved in RRD
- Extensibility by means of scripts and XML

Cacti: Host Configuration

console graphs

Console -> Create New Graphs Logged in as admin (Logout)

Create
New Graphs
Management
Graph Management
Graph Trees
Data Sources
Polling Hosts
Data Input Methods
Data Queries
Templates
Graph Templates
Host Templates
Data Templates
Import/Export
Import Templates
Export Templates
Configuration
Utilities
Cacti Settings
Utilities
User Management
Logout User

Localhost (127.0.0.1)

Create new graphs for the following host: [*Edit this Host](#)
[*Create New Host](#)

Localhost (127.0.0.1)

Host Template [Local Linux Machine]

Graph Template Name


Create: Linux - Memory Usage
Create: Unix - Load Average
Create: Unix - Logged In Users
Create: Unix - Processes

Data Query [SNMP - Get Mounted Partitions]

Index	Description	Storage Allocation Units	<input type="checkbox"/>
1	/	4096 Bytes	<input type="checkbox"/>
2	/boot	1024 Bytes	<input type="checkbox"/>
101	Real Memory	1024 Bytes	<input type="checkbox"/>
102	Swap Space	1024 Bytes	<input type="checkbox"/>
103	Memory Buffers	256 Bytes	<input type="checkbox"/>

Data Query [SNMP - Interface Statistics]

Index	Description	Type	Speed	Hardware Address	IP Address	<input type="checkbox"/>
3	eth1	ethernetCsmacd(6)	100000000	00:00:04:23:78:6E:E5		
2	eth0	ethernetCsmacd(6)	0	00:00:04:23:78:6E:E4		
1	lo	softwareLoopback(24)	10000000		127.0.0.1	<input type="checkbox"/>
					192.168.0.1	<input type="checkbox"/>



Cacti: Data Sources

Data Sources [host: No Host] **Add**

Select a host:

<< Previous Showing Rows 1 to 9 of 9 [1] **Next >>**

Name	Data Input Method	Active	Template Name	<input type="checkbox"/>
Localhost - Free Space - /dev/hda2	Get Script Data (Indexed)	Yes	Unix - Hard Drive Space	<input type="checkbox"/>
Localhost - Free Space - /dev/hda4	Get Script Data (Indexed)	Yes	Unix - Hard Drive Space	<input type="checkbox"/>
Localhost - Load Average	Unix - Get Load Average	Yes	Unix - Load Average	<input type="checkbox"/>
Localhost - Logged in Users	Unix - Get Logged In Users	Yes	Unix - Logged in Users	<input type="checkbox"/>
Localhost - Memory - Free	Linux - Get Memory Usage	Yes	Linux - Memory - Free	<input type="checkbox"/>
Localhost - Memory - Free Swap	Linux - Get Memory Usage	Yes	Linux - Memory - Free Swap	<input type="checkbox"/>
Localhost - Processes	Unix - Get System Processes	Yes	Unix - Processes	<input type="checkbox"/>
Localhost - Traffic - eth0	Get SNMP Data (Indexed)	Yes	Interface - Traffic	<input type="checkbox"/>
Localhost - Traffic - eth1	Get SNMP Data (Indexed)	Yes	Interface - Traffic	<input type="checkbox"/>

<< Previous Showing Rows 1 to 9 of 9 [1] **Next >>**

Cacti: Graph Templates

Graph Template Items [edit: Interface - Traffic (bits/sec)]						Add
Graph Item	Data Source	Graph Item Type	CF Type	Item Color		
Item # 1	(traffic_in): Inbound	AREA	AVERAGE	00CF00	⬇ ⬆ ✖	
Item # 2	(traffic_in): Current:	GPRINT	LAST		⬇ ⬆ ✖	
Item # 3	(traffic_in): Average:	GPRINT	AVERAGE		⬇ ⬆ ✖	
Item # 4	(traffic_in): Maximum: <HR>	GPRINT	MAX		⬇ ⬆ ✖	
Item # 5	(traffic_out): Outbound	LINE1	AVERAGE	002A97	⬇ ⬆ ✖	
Item # 6	(traffic_out): Current:	GPRINT	LAST		⬇ ⬆ ✖	
Item # 7	(traffic_out): Average:	GPRINT	AVERAGE		⬇ ⬆ ✖	
Item # 8	(traffic_out): Maximum:	GPRINT	MAX		⬇ ⬆ ✖	

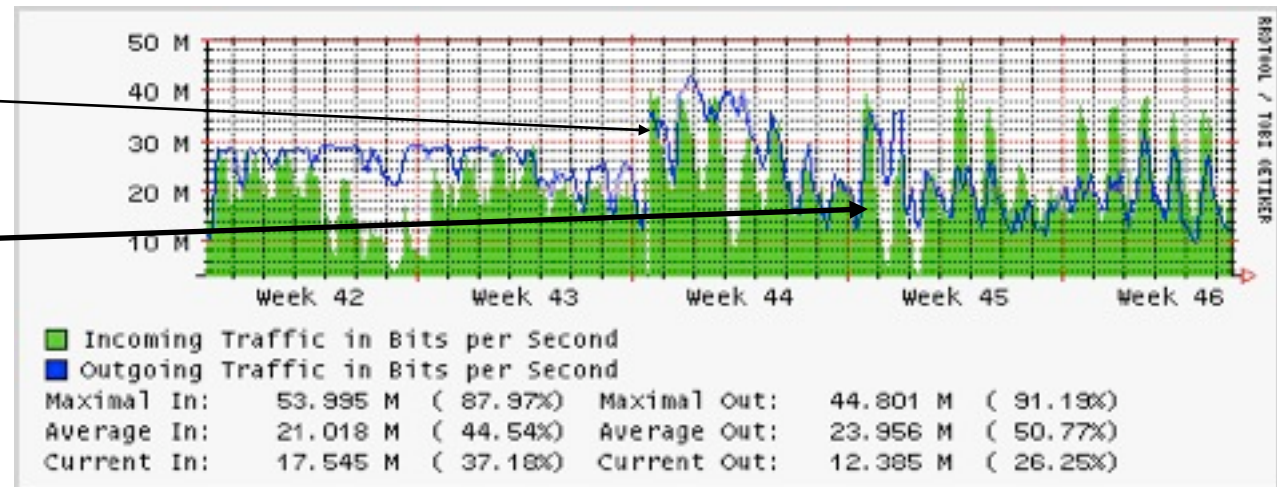
Graph Item Inputs		Add
Name		
Inbound Data Source		✖
Outbound Data Source		✖

Case Study: Bandwidth Management [1/3]

- Lack of bandwidth issues are not tackled purchasing additional bandwidth but managing the existing bandwidth.
 - Lesson learnt: the more bandwidth you have, the more you will use.

Upgrade: 34 -> 45 Mbit

Introduced CoS on P2P Traffic



Case Study: Bandwidth Management [2/3]

- Solution: Monitor and Find the Answer Yourself for Your Network (there's no general solution).
 - Analyze how the available bandwidth is used (e.g. why protocol X is used ?).
 - Traffic and Flow matrix: who's talking to who and what data are they exchanging ?

Case Study: Bandwidth Management [3/3]

Lessons Learnt from Practice:

- Poor performance can be due to use of backup-links because primary ones are unavailable (do you monitor failovers -via SNMP traps- such as STP, port status ?)
- Is your routing suboptimal or very dynamic? (SNMP provides you several MIBS for this purpose).
- Are you shaping too much? CoS (Class of Services) are good but don't misuse them! (Why don't you monitor the amount of traffic that is cut by your policers ?)

Case Study: Where is a Host?

- Association between IP address and name

```
deri@tar:~$ nslookup 131.114.21.22
```

```
Server: localhost
```

```
Address: 127.0.0.1
```

```
Name: jake.unipi.it
```

```
Address: 131.114.21.22
```

- Association between host and owner
 - `namenslookup -type=SOA`
 - WAIS [Wide Area Information System] <http://www.ai.mit.edu/extra/the-net/wais.html>
 - WHOIS [RFC-812]

Whois Example

Domain: unipi.it
Status: ACTIVE
Created: 1996-01-29 00:00:00
Last Update: 2008-02-14 00:02:47
Expire Date: 2009-01-29

Registrant

Name: Universita' degli Studi di Pisa
ContactID: UNIV302-ITNIC
Address: Centro SERRA
Pisa
56100
PI
IT
Created: 2007-03-01 10:42:01
Last Update: 2008-01-19 09:46:08

Registrar

Organization: Consortium GARR
Name: GARR-MNT

Where in the World is host X ?

- RFC 1876: A Means for Expressing Location Information in the Domain Name System
- <http://www.caida.org/tools/utilities/netgeo/>
- <http://www.maxmind.com/>
- <http://www.geobytes.com/>

Case Study: OS Fingerprinting

- Active
Send probe packets in order to guess the host OS (<http://nmap.org/>)
- Passive
Look at 3-way handshake and compare it with a database of known signatures in order to guess the host OS (<http://ettercap.sf.net/>).

OS Fingerprinting: Ettercap

WWWW:MSS:TTL:WS:S:N:D:T:F:LEN:OS

WWWW: 4 digit hex field indicating the TCP Window Size

MSS : 4 digit hex field indicating the TCP Option Maximum Segment Size
if omitted in the packet or unknown it is "_MSS"

TTL : 2 digit hex field indicating the IP Time To Live

WS : 2 digit hex field indicating the TCP Option Window Scale
if omitted in the packet or unknown it is "WS"

S : 1 digit field indicating if the TCP Option SACK permitted is true

N : 1 digit field indicating if the TCP Options contain a NOP

D : 1 digit field indicating if the IP Don't Fragment flag is set

T : 1 digit field indicating if the TCP Timestamp is present

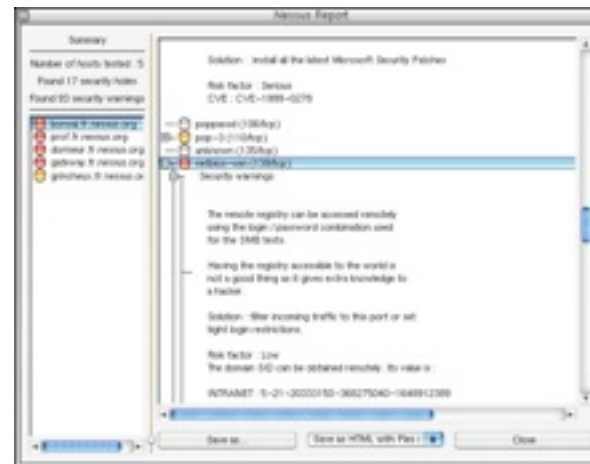
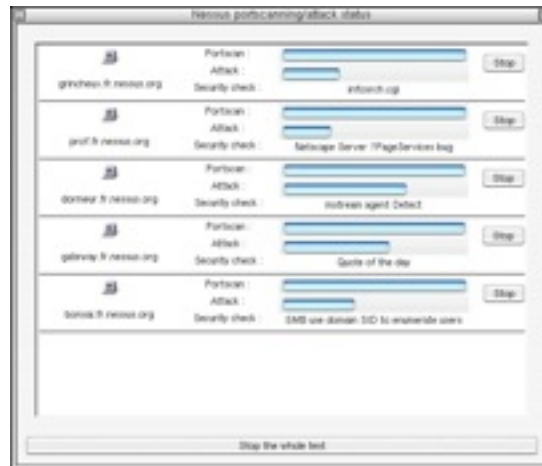
F : 1 digit ascii field indicating the flag of the packet

S = SYN

A = SYN + ACK

Case Study: Security Scanner

- Nessus [\[http://www.nessus.org/\]](http://www.nessus.org/)
- Saint [\[http://www.saintcorporation.com/\]](http://www.saintcorporation.com/)



Case Study: Network Security

Security is a process not a product (BS7799).

- Are you able to detect network anomalies ?
- Are you sure you know what to monitor ? Most of issues are produced by traffic we never expect to see in your network: monitor everything, filter things you expect to see, look at the rest and explain this happened.
- Do you have an automatic fault recovery ? Supposing you detect the problem (e.g. SNMP trap) is your system reacting automatically or waiting for you to come back from holidays ?

Case Study: P2P Detection

- P2P is hard to detect with standard methods:
 - It cannot be detected using fingerprints (e.g. port-2-protocol association).
- However it can be detected...
 - It can be detected in terms of deviation for a standard behaviour (e.g. a workstation cannot open more than X connections/minute nor keep more than Y connections open).
 - Analysis of initial payload bytes in order to detect the protocol.
 - High percentage of unsuccessful TCP connection establishment.
 - Packets/Bytes ratio above the average (P2P sources send many packets, mostly for talking with peers).
 - Identification of client-to-client (> 1024) communications with no FTP command channel open.

Case Study: SPAM Detection

- Large, open networks (e.g. universities, ISPs) are the best places for sending spam (unsolicited email).
- How to identify SPAM sources:
 - Problem similar to P2P but simpler (SMTP-only, 1 connection = 1 email) .
 - Select the set of top N SMTP senders.
 - Remove from the set all the known SMTP servers.
 - Studies shown that in average an host does not send more that 8-10 emails/minute.
 - Very simple problem to tackle using flow-based protocols such as NetFlow.

Case Study: Virus/Trojan Detection

- Problem similar to SPAM detection but more complex as the protocol/ports used are not fixed.
- Attacks do not have a precise target: they somehow behave as network scanners.
- Detection:
 - If the problem is known (e.g. traffic on UDP port 135) focus only on these selected traffic patterns.
 - Keep an eye on ICMP messages (e.g. port/destination unreachable) as they are the best way to detect network scanners.

6. Final Remarks

So What Can we Basically Expect from Network Monitoring ?

- Ability to automatically detect those issues that are permanently monitored (e.g. no traffic on the backbone link: network down ?).
- Receive alarms about potential (e.g. CPU utilization is too high) and real (e.g. disk is full) problems.
- Automatic notification and restore for known problems with known solutions (e.g. main link down, the backup link is used).
- Report to humans for all those problems that need attention and that cannot be restored (e.g. host X is unreachable).

Monitoring Caveats

- If a monitoring application needs human assistance for a problem that could be solved automatically, then the monitoring applications is not completely useful.
- Alarm (100% sure that there is something wrong) != Warning (maybe this is an issue): don't pretend to be precise/ catastrophic if this is not the case.
- Alarms are useless if there's nobody who looks at them.
- Too many (false) alarms = no alarm: humans tend to ignore facts if some of them are proven to be false.