CERN openlab Summer Student 2008

Networking overview

14 August 2008

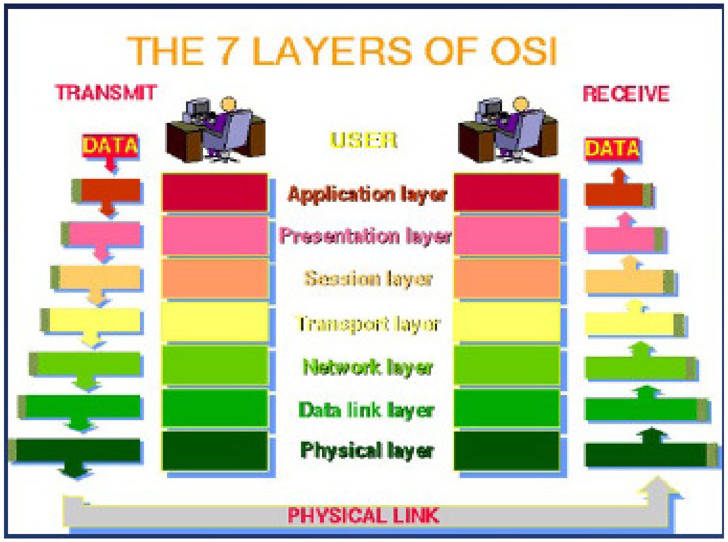
Ryszard Jurga

Agenda

- Introduction to OSI model
- More details about TCP
- Network performance
- Glance at CERN network
  - Campus network
  - LHC networking
- Network anomalies
  - CINBAD project

The OSI Model

- Open Systems Interconnection (OSI)
- Framework and protocols developed to allow different networks to communicate
- Each layer provides well-defined interface to the layer above
  - And each layer uses only the services of the layer below
- Each layer adds a header
  - some also a trailer



OSI Layers (1)

- Physical Layer
  - Concerned with transmission of bits and bytes
  - Standards for electrical, mechanical and signaling interfaces
    - What do bits and bytes look like "on the wire"
- Link Layer
  - Groups bits and bytes into frames and ensures correct delivery
  - Handles errors in physical layer
  - Adds bits (head/tail) + checksum (receiver verifies checksum)
  - Sublayers: LLC – Logical Link Control and MAC – Medium Access Control

OSI Layers (2)

- Network Layer ("Packet" layer)
  - Transmission and addressing of packets
  - Chooses the best path for the packet (routing)
    - Each packet gets routed independently to its destination
  - Connectionless
  - Unreliable, best effort service
  - Internet Protocol – IP
- Transport Layer
  - transparent transfer of data between end users
  - UDP, TCP

OSI Layers (3)

- Session Layer
  - Establishes, maintains and terminates sessions between end-user application processes across networks
- Presentation Layer
  - Translates application → network format
  - Can potentially include De-/Encryption, Compression...
- Application Layer
  - DNS, FTP, SMTP, NFS, …

TCP protocol

- designed in 70's
  - influenced by end-to-end argument
- ensures reliable service (network layer does not deal with lost messages)
- breaks message into segments (blocks), assigns a sequence number and sends them
- builds reliable network connection on top of IP (or other protocols)
  - detection of corrupted data, loss, duplicated and out of sequence packets
  - correction of errors

TCP details

- the receiver sends a TCP ACK packet to a sender in order to acknowledge receipt of a packet
  - Round Trip Time (RTT)
    - the minimum time for a TCP ACK to be received by the sender
    - e.g. Geneva-Taiwan RTT=~330ms

- TCP window
  - Amount of outstanding data a sender can send before it gets an ACK back from the receiver
  - Sender must keep all sent segments until acknowledged
  - optimal size = Bandwidth * RTT
    - e.g: 40MB for a 1Gb/s connection to Taiwan
  - recommended size = 2*optimal size

TCP flow control

- Technique that matches the transmission rate of sender to that of receiver and the network
  - to avoid flooding the network
  - to adjust tcp window

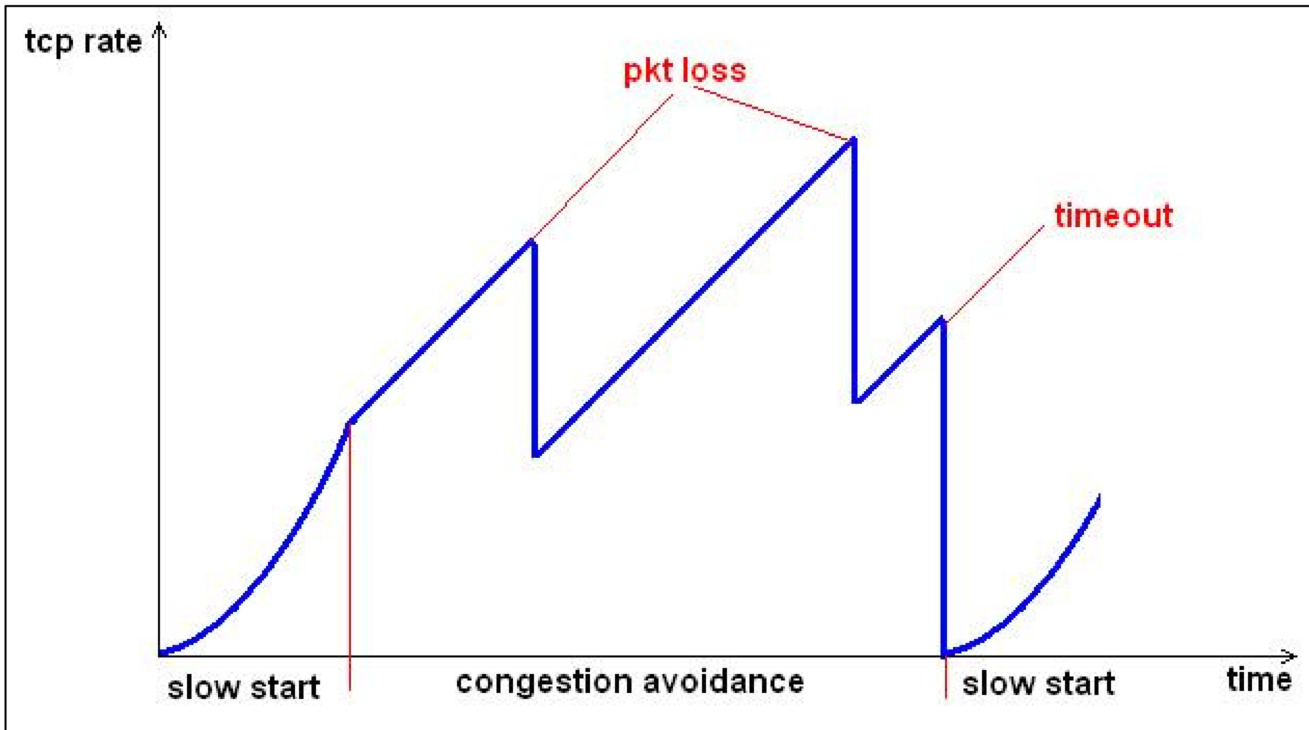
- Based on two mechanisms:
  - slow start
    - exponential increase in tcp window size
  - congestion avoidance
    - increase/decrease of tcp window based on different criterions (e.g. pkt loss, rtt, queuing delay)

TCP flow control

- Slow start
  - Initially tcp window is set to the MSS
  - on every TCP ACK a tcp window is increased by one MSS
    - data rate of sender doubles every RTT
  - the tcp window increases until:
    - the advertised tcp window size is reached
    - packet loss is detected on the network (back to congestion avoidance)
    - there is no traffic

TCP flow control – TCP Reno

- Congestion avoidance (TCP Reno)



is

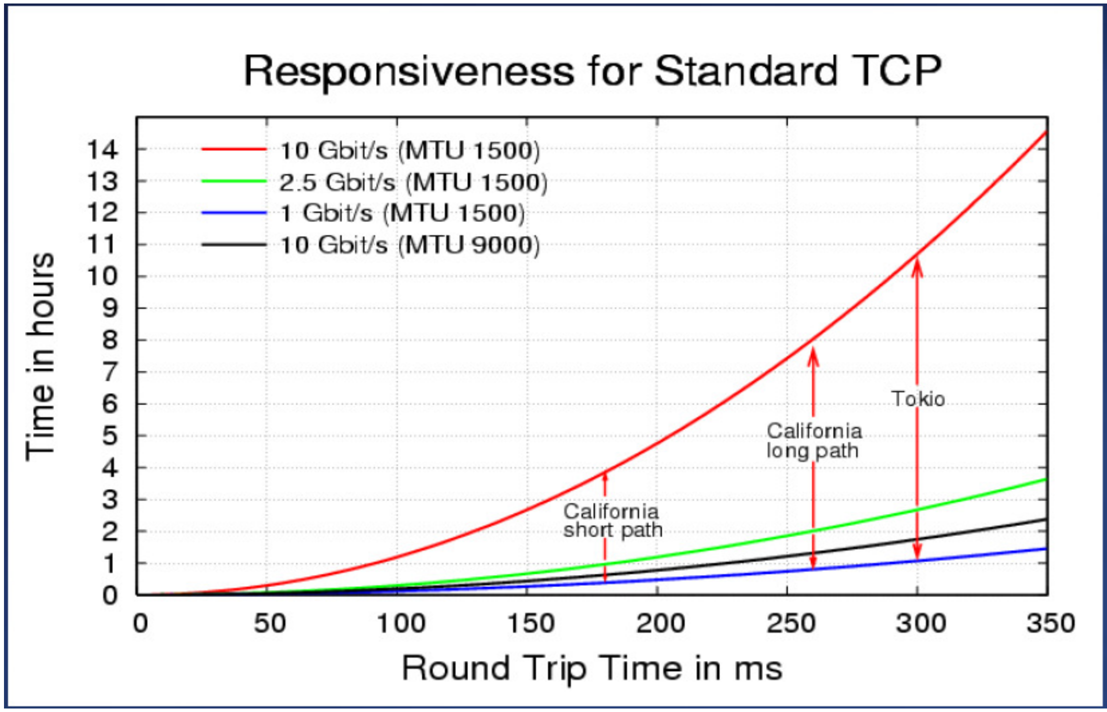
s

ase)

t to

TCP Reno - responsiveness

- Responsiveness ρ measures how quickly the connection goes back to full bandwidth after a packet loss



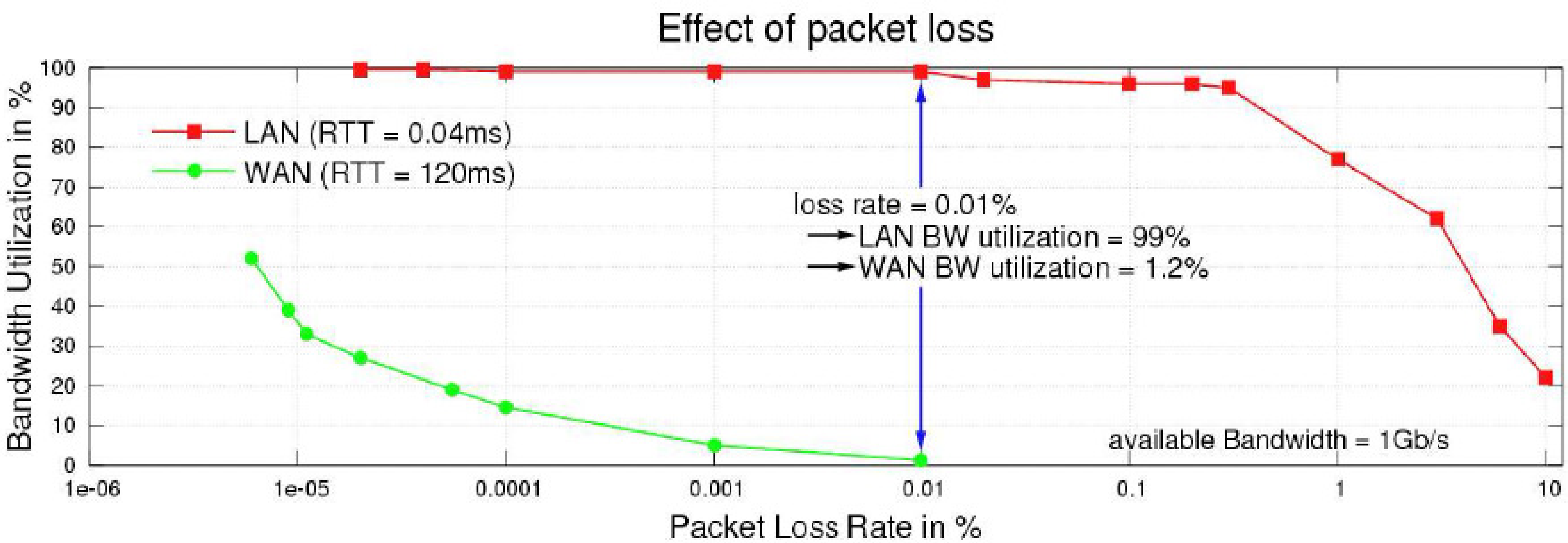
$$\rho = \frac{C * RTT^2}{2 * MSS}$$

C – Capacity of the link

RTT – Round Trip Time

MSS – Message size (MTU - 40Bytes)

TCP Reno - performance



TCP flow control variants

- try to optimize a given connection by using additional information about this particular connection
  - analyzing loss probability, RTT, queuing delay
- change the multiplicative parameters in the congestion avoidance protocol
- examples:
  - CUBIC-TCP, BIC-TCP, Hamilton TCP, TCP Vegas, TCP Westwood
- support for pluggable congestion control algorithms in Linux (>2.6.13)

TCP and Performance of Network Devices (1)

- Large traffic bursts can fill up buffers in the network device
  - Standard TCP (Reno) sends all data in the TCP buffer within a round trip time as fast as possible
    - FAST TCP distributes the traffic over RTT
  - Large tcp windows and many streams put a lot of pressure on the buffering
  - The larger these bursts, the higher are the risks that this buffer overflows and causes multiple segments to be dropped

TCP and Performance of Network Devices (2)

- Modern high-end routers are general-purpose computers atop a pool of packet-forwarding ASICs or specialized processors
  - For performance, any per-packet operation must happen in the ASICs
  - This is the so-called “fast path”
  - Special cases must be “process switched”
- TCAM vs DRAM
  - Fast, specialized memory vs large, general-purpose memory

TCP and hosts' CPU performance

- TCP/IP stack is usually implemented in OS
  - kernel context switches
  - multiple memory copies: to device driver buffer, OS buffer, user process memory
    - adds latency and consumes CPU

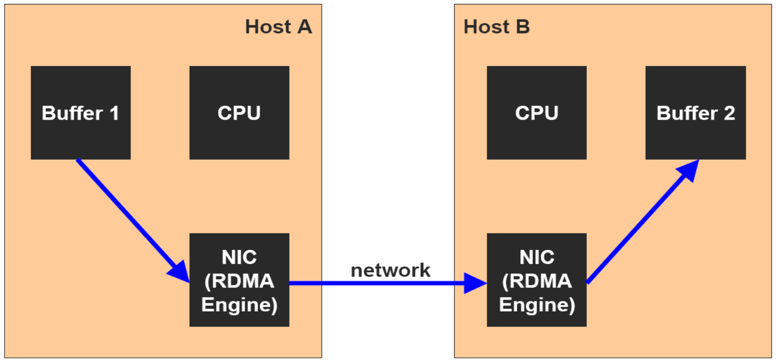
- Network bandwidth outstripped Moore's Law in recent years
  - e.g. 1995-2003: the Ethernet speed – 100x increase, 40x increase in transistor density

Closing the gap in the CPU performance

- TCP Offload Engine (TOE)
  - move TCP processing to NIC
  - does not reduce memory copies
  - increases NIC hardware complexity
    - limited resources: e.g. memory
  - requires more complex maintenance
    - e.g. applying patches against firmware
  - works fine with the Remote Direct Memory Access (RDMA)

RDMA

- "zero copy" mechanism
  - application/kernel buffers registered end exposed to remote peers via NIC driver
  - CPU bypassing
  - direct write/read to remote buffers
- designed for:
  - Infinibad
  - iWARP – RDMA over TCP/IP (e.g. Ethernet)

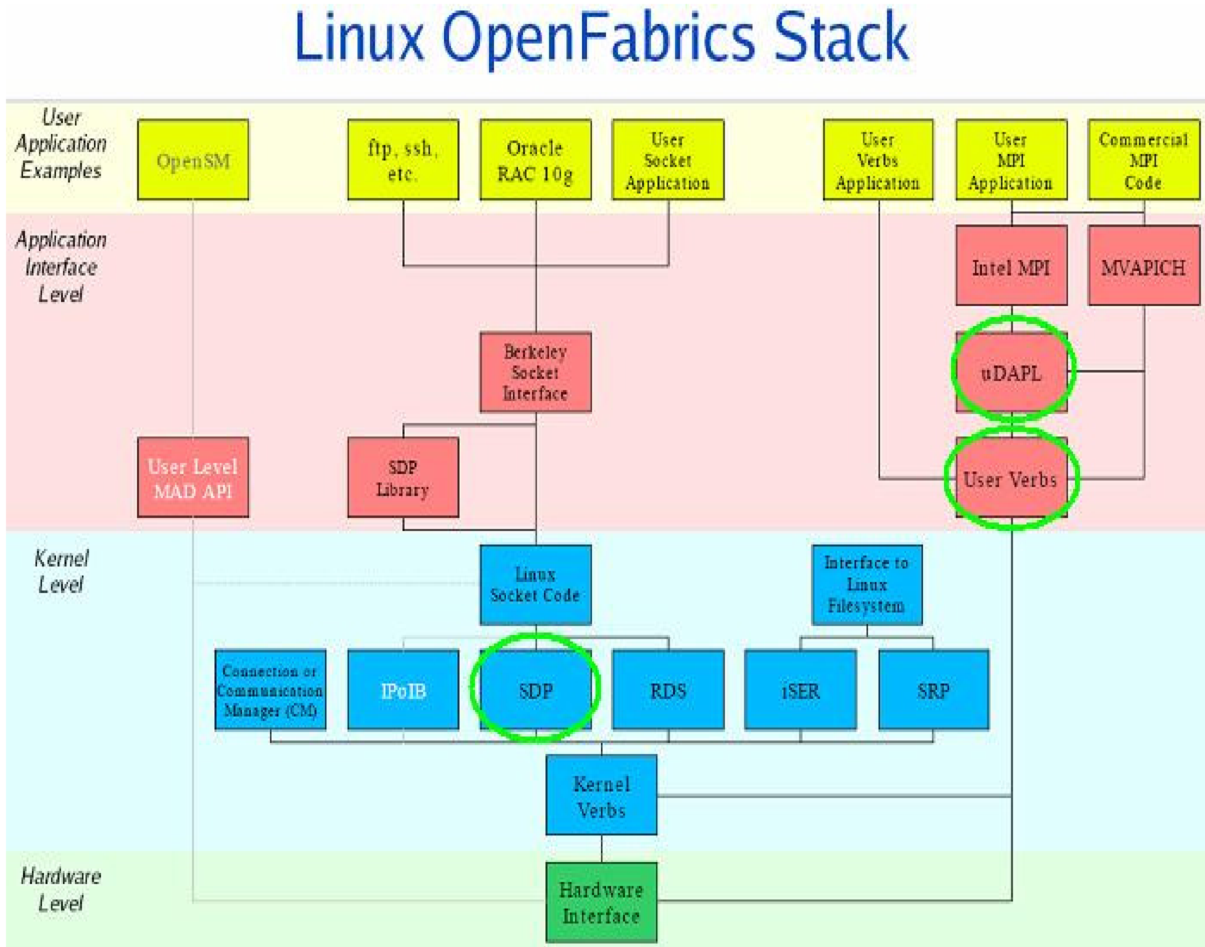


OpenFabrics stack (1)

- Provides a common API that allows applications to take advantage of the RDMA, low latency and high messaging rate capabilities
- Encompass both the InfiniBand and iWARP standards
- Incorporated in the Linux Kernel since 2.6.11

OpenFabrics stack (2)

- **Sockets Direct Protocol (SDP) and SDP Library**
  - compatible sockets interface with Berkeley Socket (provides AF_SDP in place of AF_INET address family)
  - LD_PRELOAD capable library
- **User verbs**
  - Direct access to hardware interface, used directly by user applications
- **uDAPL**
  - Interface between user applications and user verbs

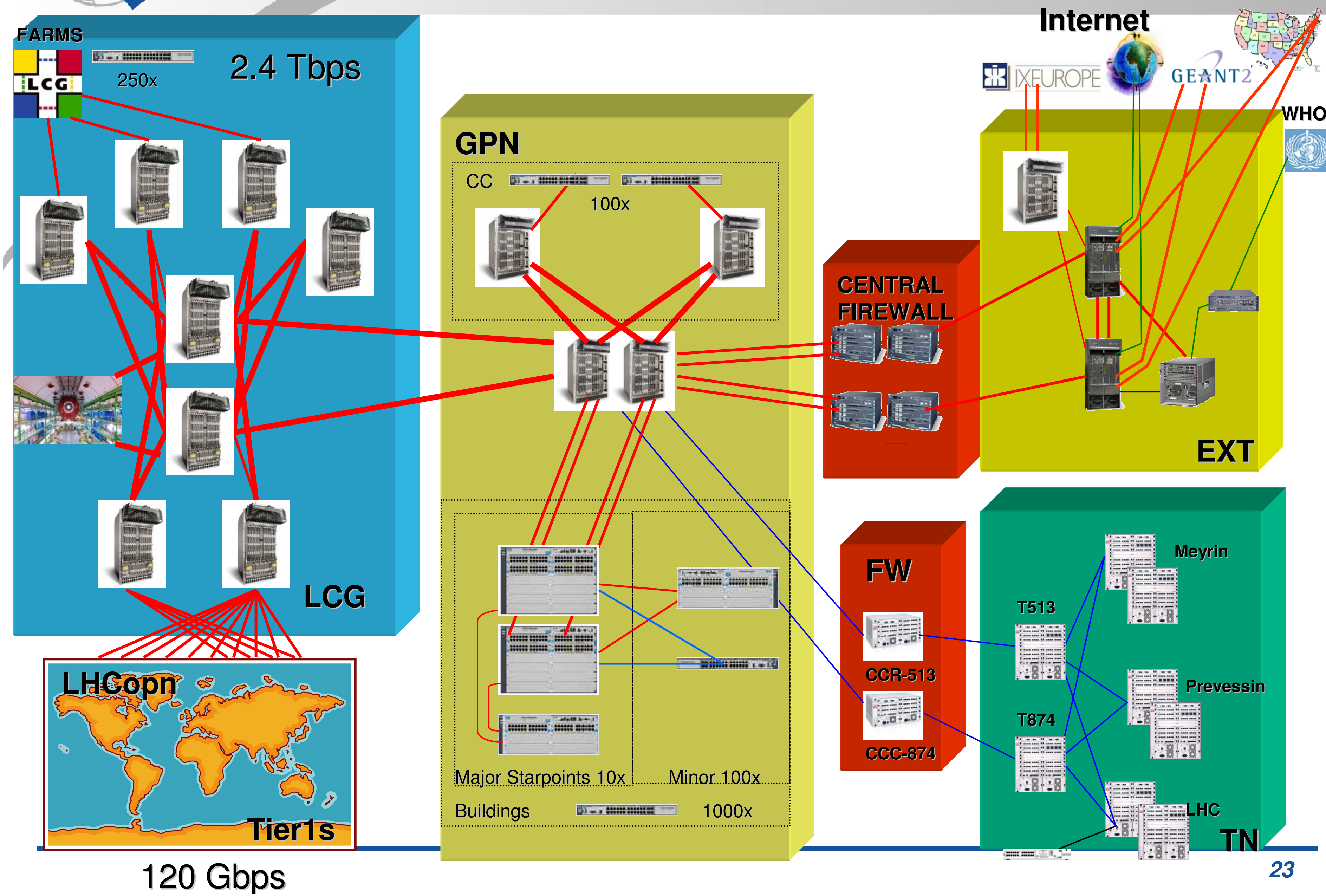


CERN campus network

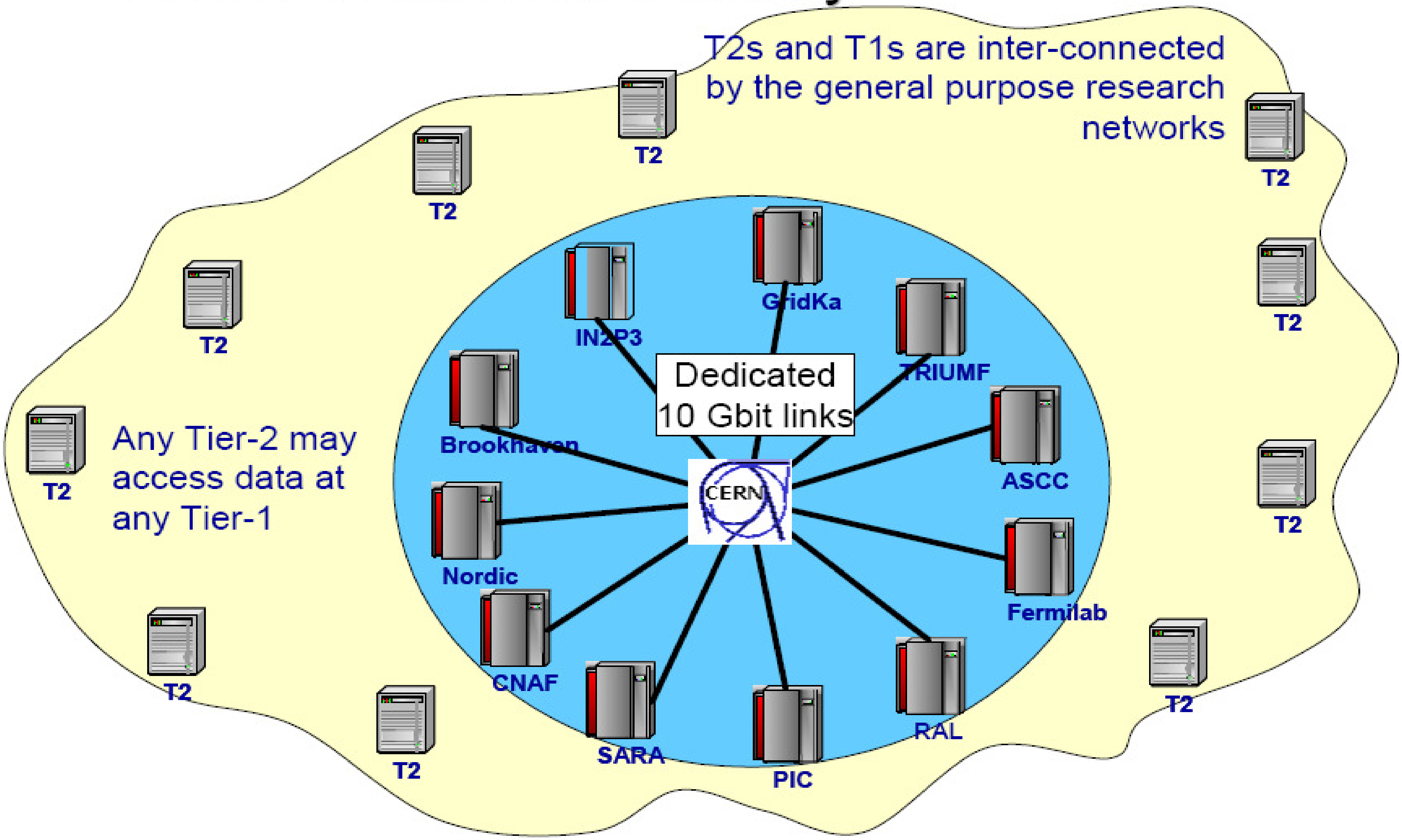
and

LHC optical network

Simplified overall CERN campus network topology



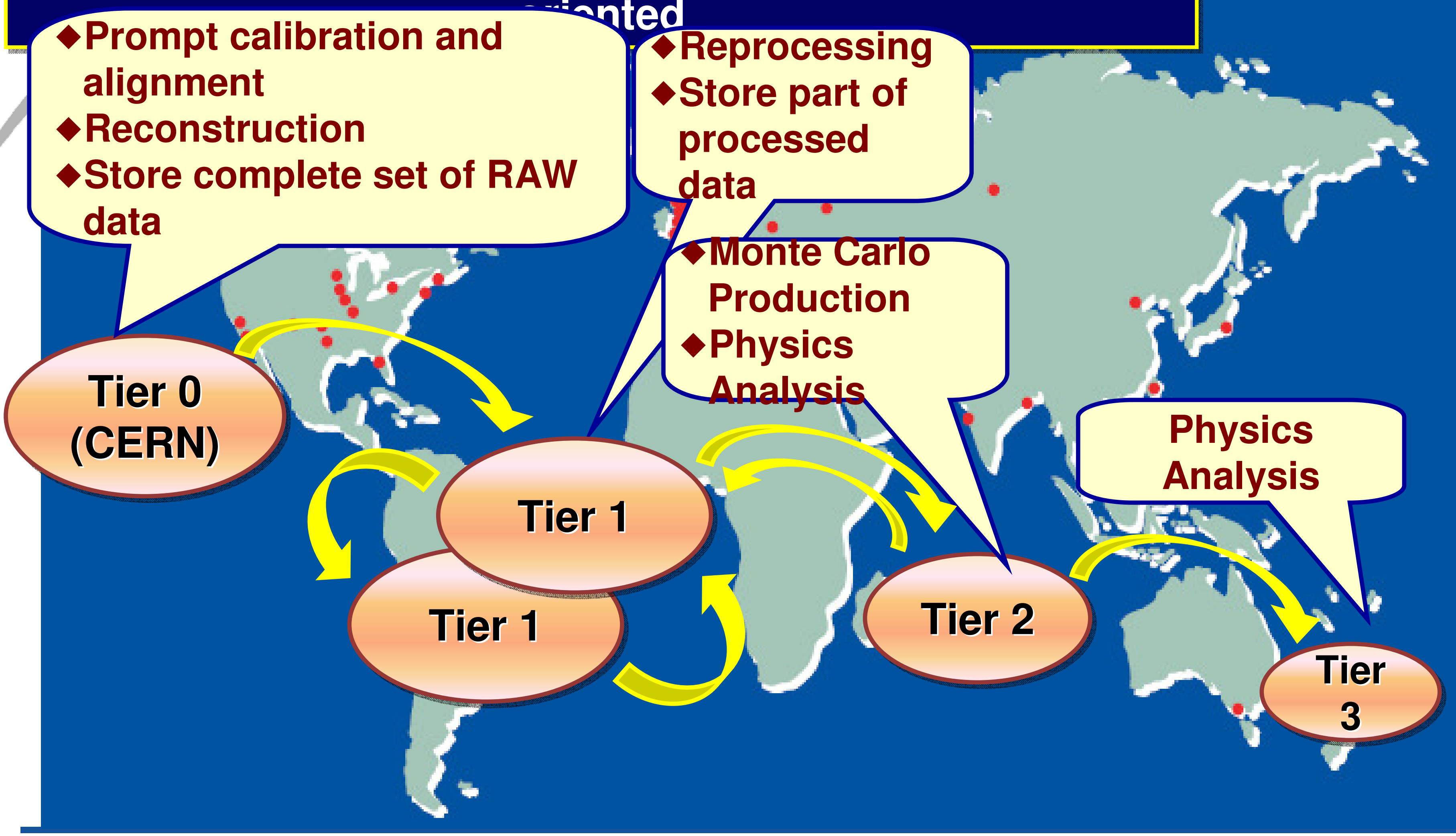
T0/T1/T2 Interconnectivity



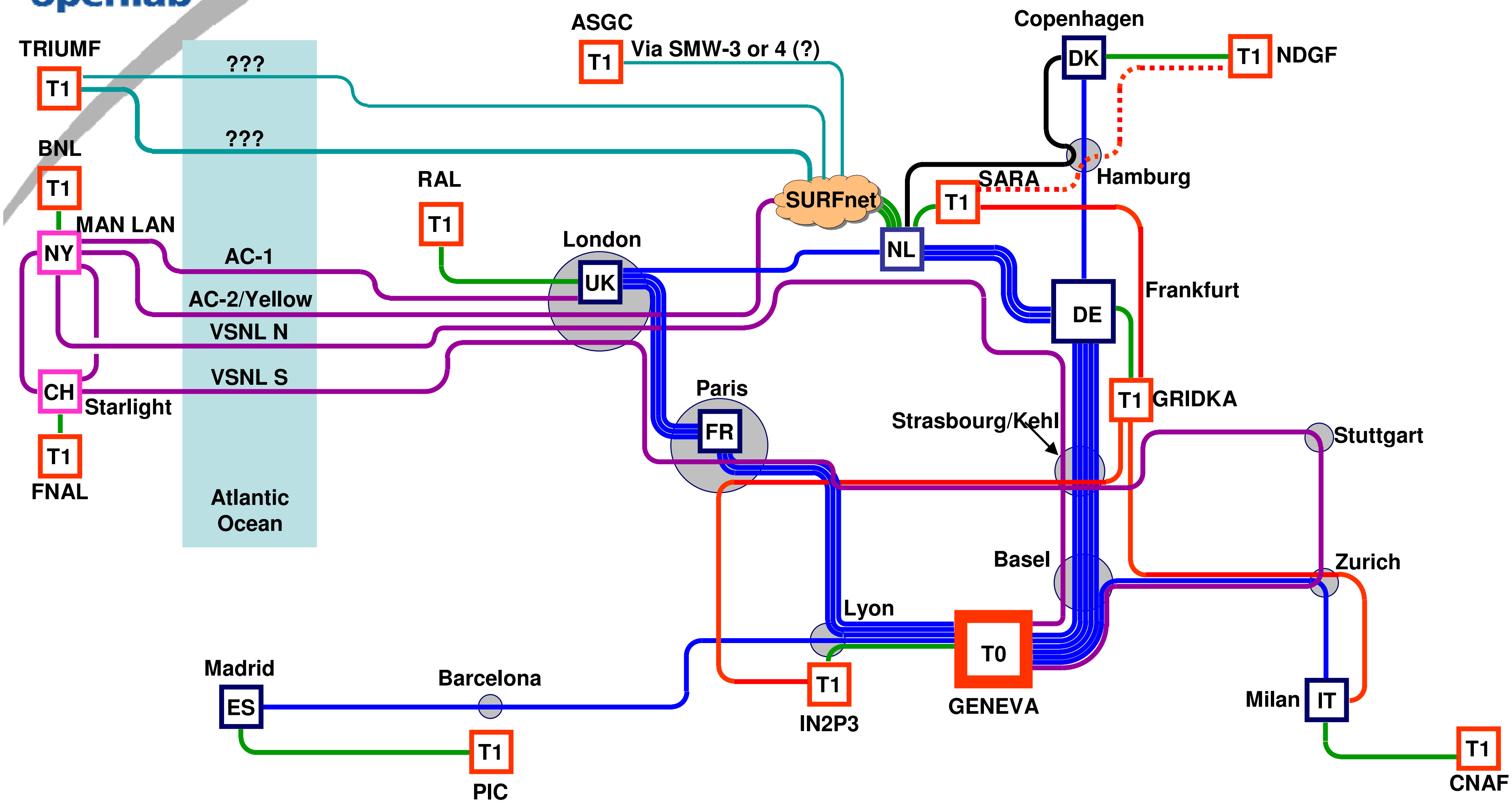
The Roles of Tier Centers

11 Tier1s, over 100 Tier2s

→ LHC Computing will be more dynamic & network-oriented



T0-T1 Lambda Routing



Michael Enrico (DANTE)

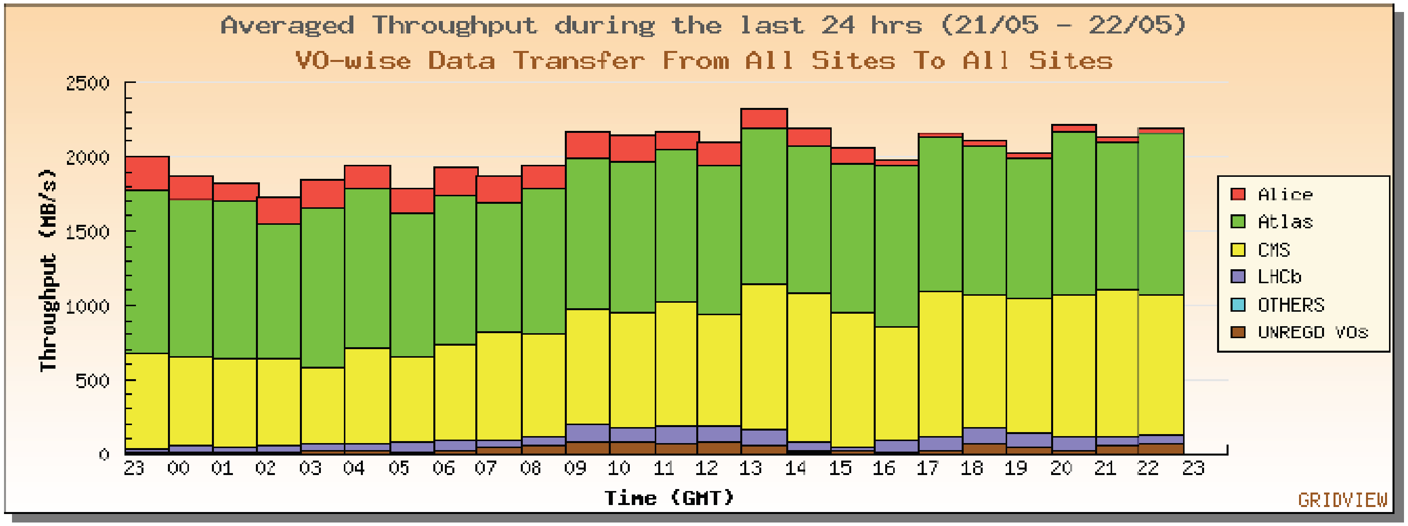
Service challenges

- are meant to enable CERN and the LHC experiments to test the transfer of the data coming from the experiments at CERN to the LCG Tier 1 sites around the world
  - from general connectivity, through achieving high throughput to reaching desired functionality and stability of the software stack

- Nominal rates per site - 150 – 200MB/s

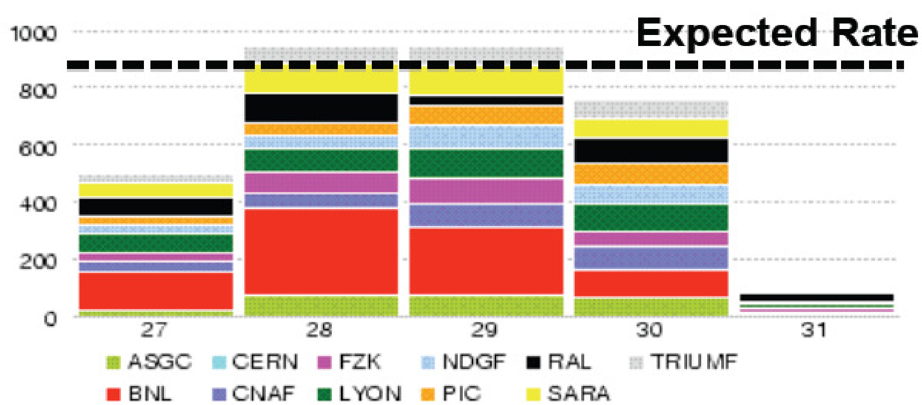
Service challenge – throughput latest results

T0-T1



T0-T1,T1-T1,T1-T2

Monte Carlo



Network Anomalies

Anomaly Definition (1)

- Anomalies are a fact in computer networks
- Anomaly definition is very domain specific:

Network faults	Malicious attacks	Viruses/worms
Misconfiguration	…	…

- Common denominator:
  - *“Anomaly is a deviation of the system from the normal (expected) behaviour (baseline)”*
  - *“Normal behaviour (baseline) is not stationary and is not always easy to define”*
  - *“Anomalies are not necessarily easy to detect”*

Anomaly Definition (2)

- Just a few examples of anomalies:
  - Unauthorised DHCP server (either malicious or accidental)
  - NAT (not allowed at CERN)
  - Port Scan
  - DDoS attack
  - Spreading worms/viruses
  - Exploits (attacker trying to exploit vulnerabilities)
  - Broadcast storms
  - Topology loops
- Examples of potential anomaly indicators:
  - TCP SYN packets without corresponding ACK
  - IP fan-out and fan-in (what about servers – i.e. DNS?)
  - Unusual packet sizes
  - Very asymmetric traffic to/from end system (what about servers?)
  - Unwanted protocols on a given subnet (packets '*that should not be there*')
  - Excessive value of a certain measure (i.e. TCP Resets)
  - ICMP packets

Anomaly Detection (1)

- Signature based detection methods:
  - Perform well against known problems

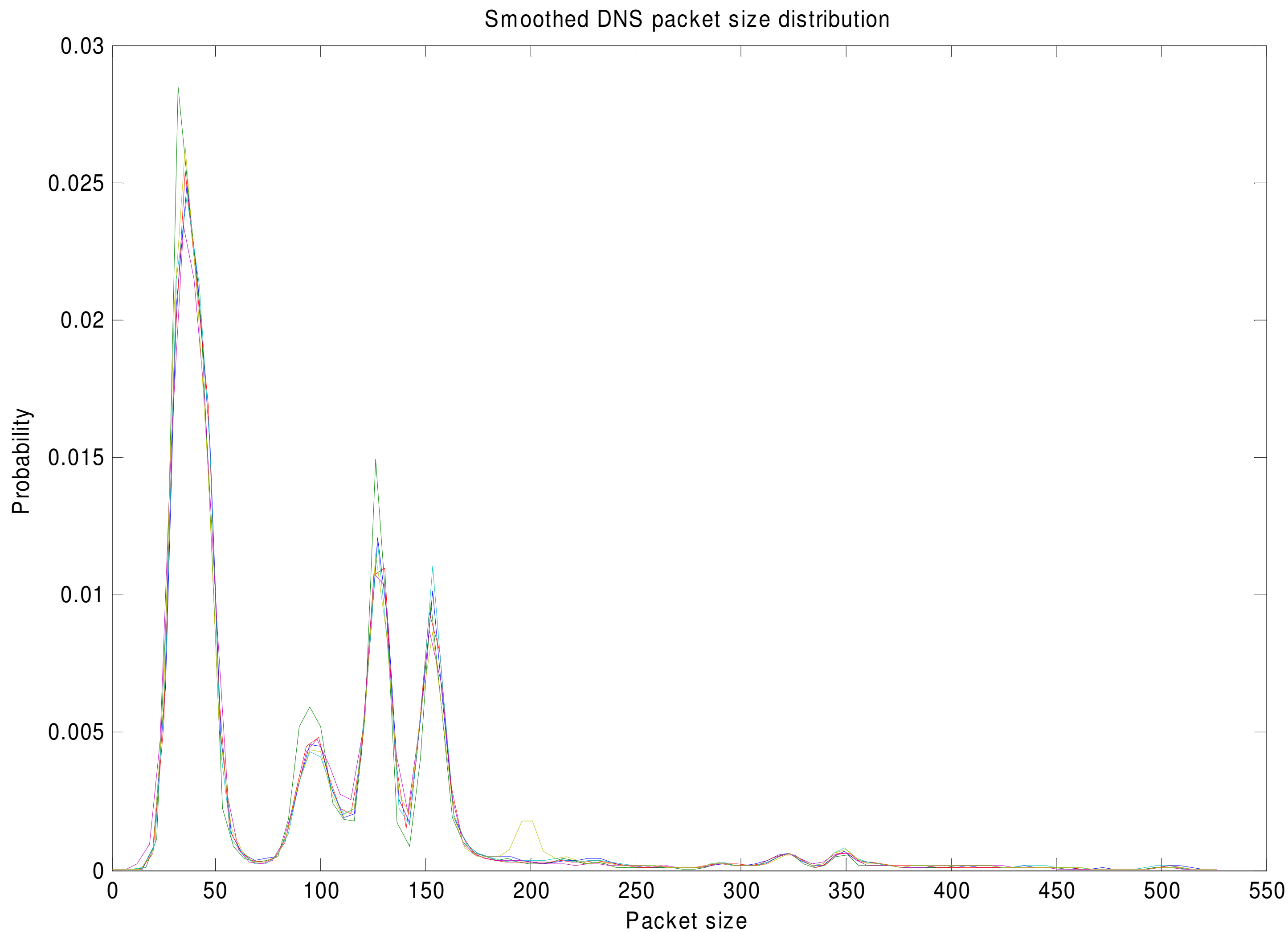
Example:
Martin Overton, “Anti-Malware Tools: Intrusion Detection Systems”, European Institute for Computer Anti-Virus Research (EICAR), 2005

```
00000760  E7 6F 8C 88 3A 79 B3 9D 9D 52 44 AD 62 61 3D 8F   çoI I:y³I IRD-ba=I
00000770  98 6D 4C 07 C2 00 E5 4C 48 F0 91 4E EB 87 89 77   ImLIÂ.åLHð´NëI Iw
00000780  7E E0 83 B1 94 94 CC E9 F5 97 97 53 95 5C 95 AF   ~àI±I IÌéõI ISI\I¯
00000790  C6 40 C5 CA AC 25 8E 47 F1 5D 0B 9F BB CB A6 67   Æ@ÅÊ¬%IGñ]I I»Ë¦g
000007A0  DB 44 E8 D2 48 3B 8F 76 CB 9E E1 53 FB FB 41 11   ÛDèÒH;IvËIáSûûAI
```

Signature found at W32.Netsky.p binary sample
Rules for Snort:

```
alert tcp $EXTERNAL_NET any -> any any (msg:"W32.NetSky.p@mm - SMB";content:"|4E EB 87 89
77 7E E0 83 B1 94 94 CC E9 F5 97 97 53 95 5C 95 AF C6 40 C5 CA AC 25 8E 47 F1 5D 0B|";
classtype:misc-activity;rev:1;)
```

Anomaly Detection (2)

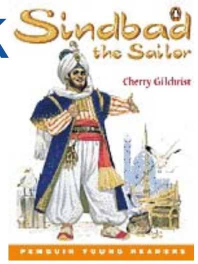


Anomaly Detection (3)

- Statistical detection methods – examples:
  - Threshold detection:
    - Count occurrences of the specific event over ΔT
    - If the value exceeds certain threshold -> fire an alarm
    - Simple and primitive method
  - Profile based:
    - Characterise the past behaviour of hosts (i.e. extract features, patterns, sequential patterns, association rules, classify into groups)
    - Detect a change in behaviour
    - Detect suspicious class of behaviour

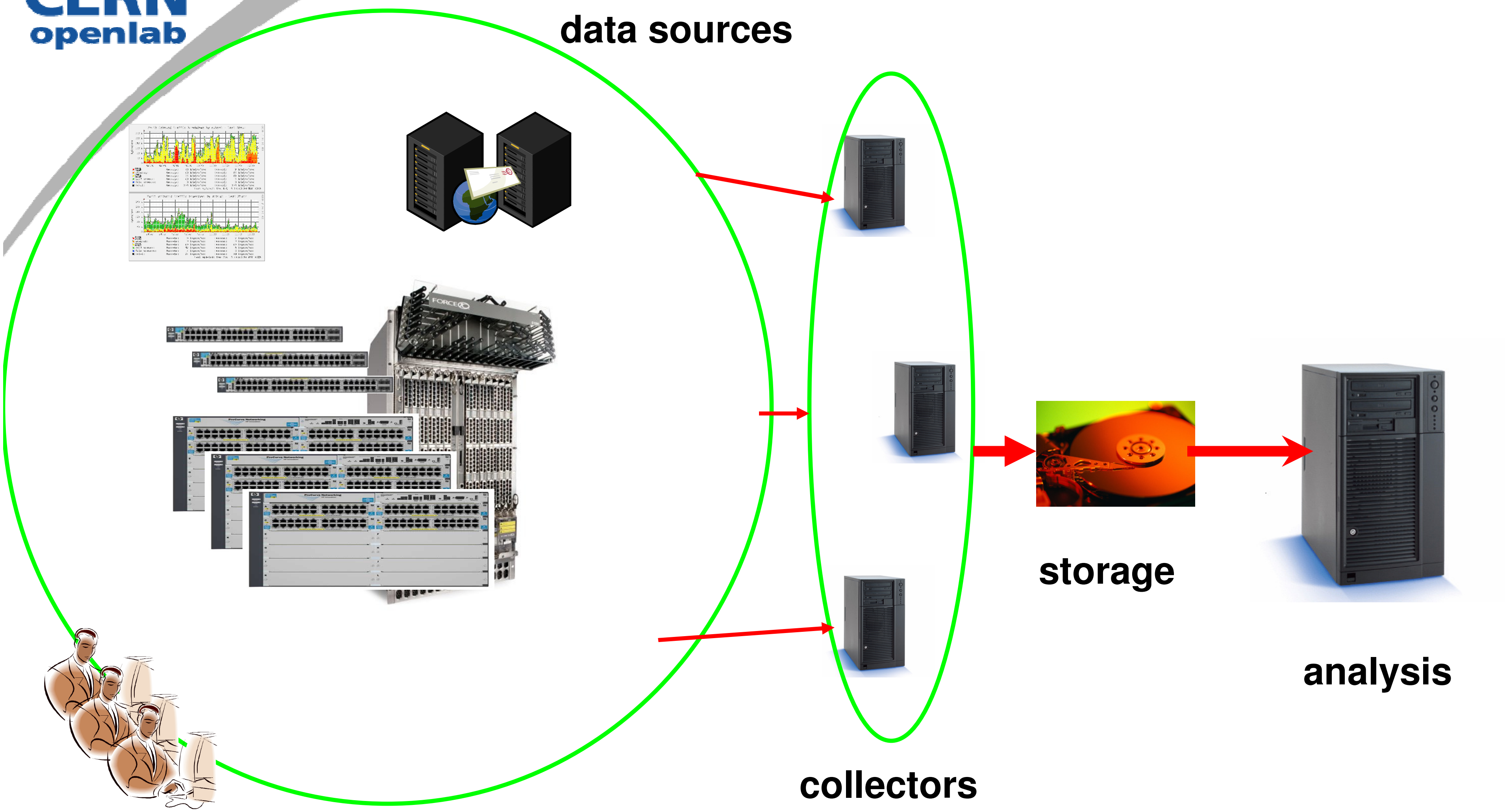
CINBAD Project definition

- **CINBAD: C**ern **I**nvestigation of **N**etwork **B**ehavior **A**nomaly **D**etection



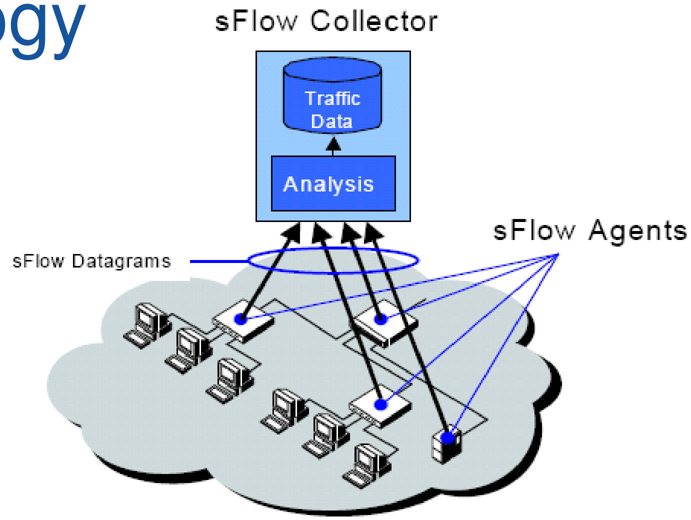
- The project goal is to understand the behaviour of large computer networks (10’000+ nodes) in High Performance Computing or large Campus installations to be able to:
  - Detect traffic anomalies in the system
  - Be able to perform trend analysis
  - Automatically take counter measures
  - Provide post-mortem analysis facilities

CINBAD Basic Principle



CINBAD data sources

- Network data sources
  - sFlow, Netflow, SNMP, RMON, probes, etc.
- Configuration data, topology
- Servers logs
  - DNS, DHCP, etc.
- Monitoring systems
  - alerts
- Human reports
  - network operator reports, user complains
- others



Acknowledgments

Artur Barczyk

Andreas Hirstius

Milosz Marian Hulboj

Martin Swany

Q&A