MonALISA

MONitoring Agents using a Large Integrated Services Architecture

Monitoring and Control of Large Scale Distributed Systems

Enrico Fermi International School of Physics

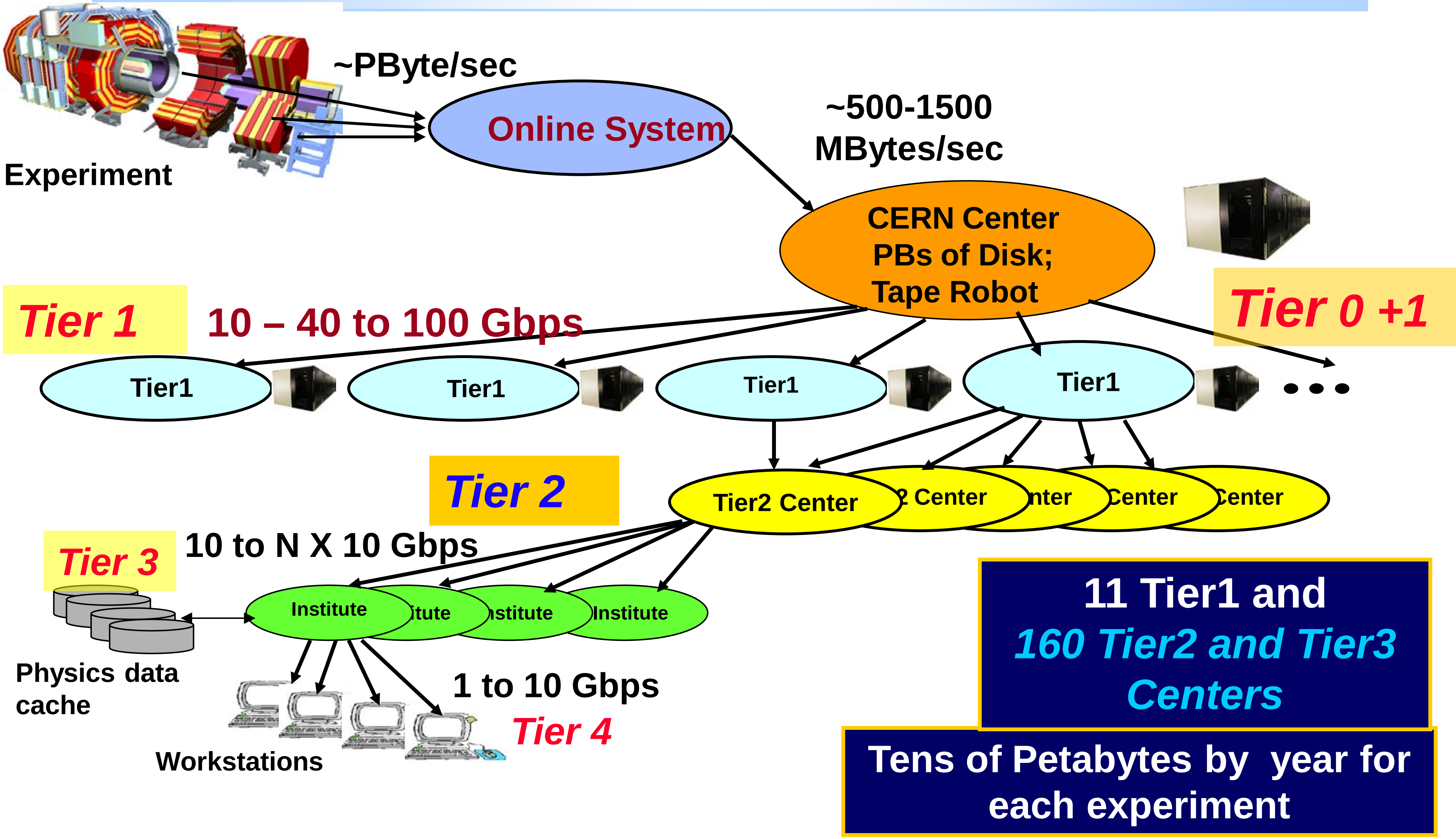
Grid and Cloud Computing

26 July 2014

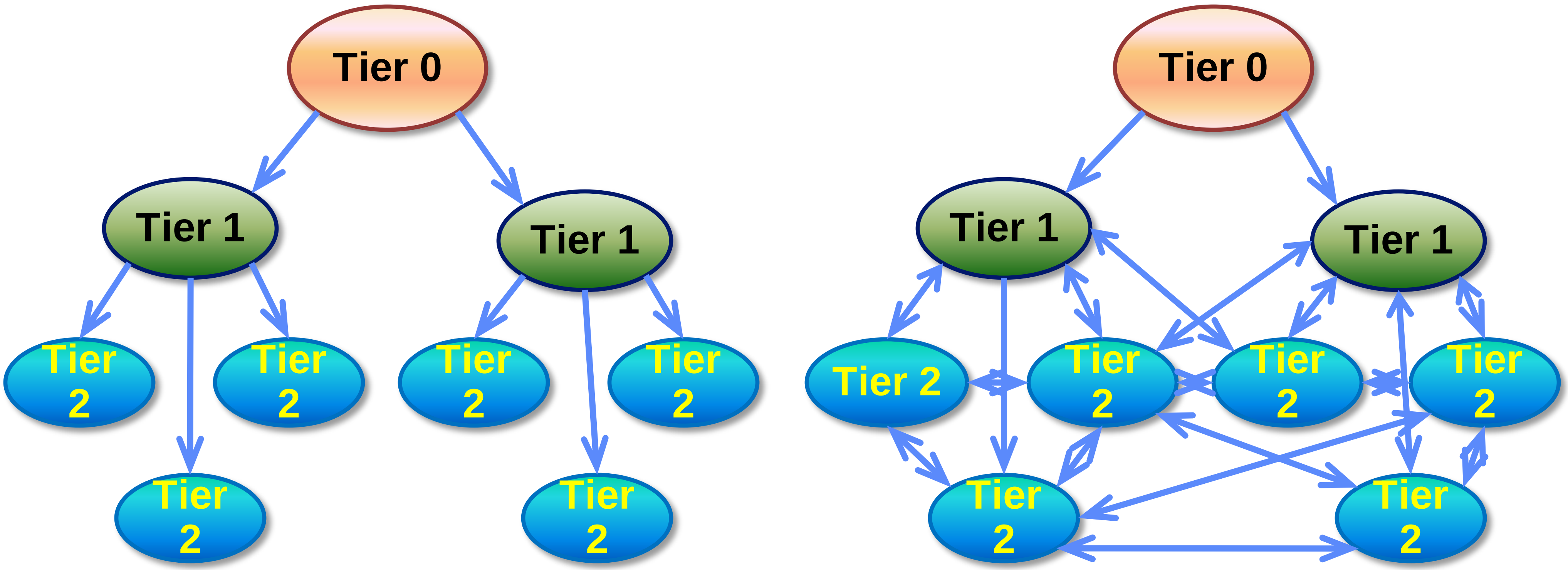
Iosif Legrand

Caltech / CERN

The MONARC - LHC Computing Model

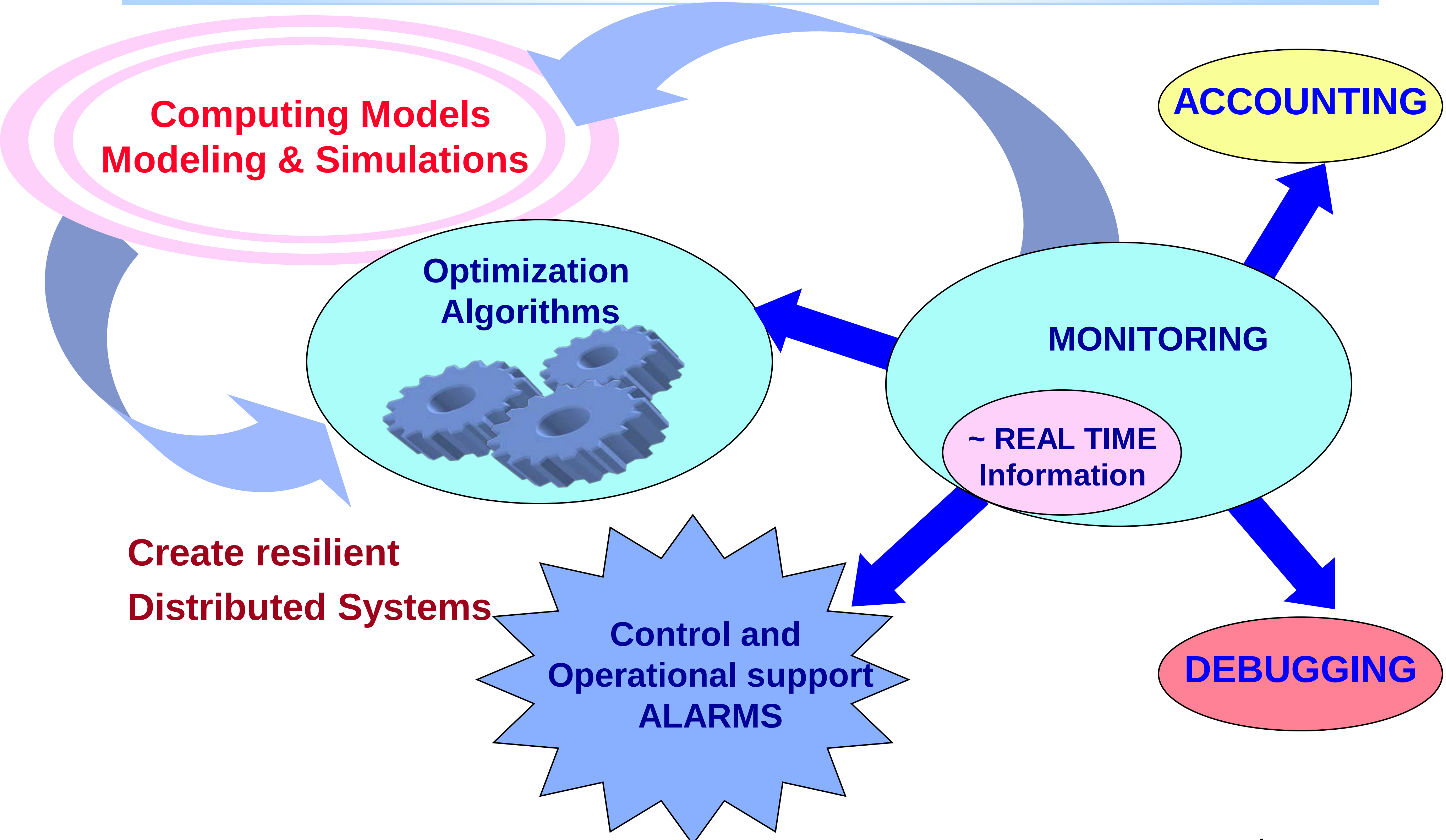


Moving toward a "Cloud" Model



A dynamic scheme to access data requires significantly more knowledge about the data distribution state of the servers and the network infrastructure . The information should be available to large number of jobs in near real-time

Monitoring Information is necessary for System Design, Control, Optimization, Debugging and Accounting

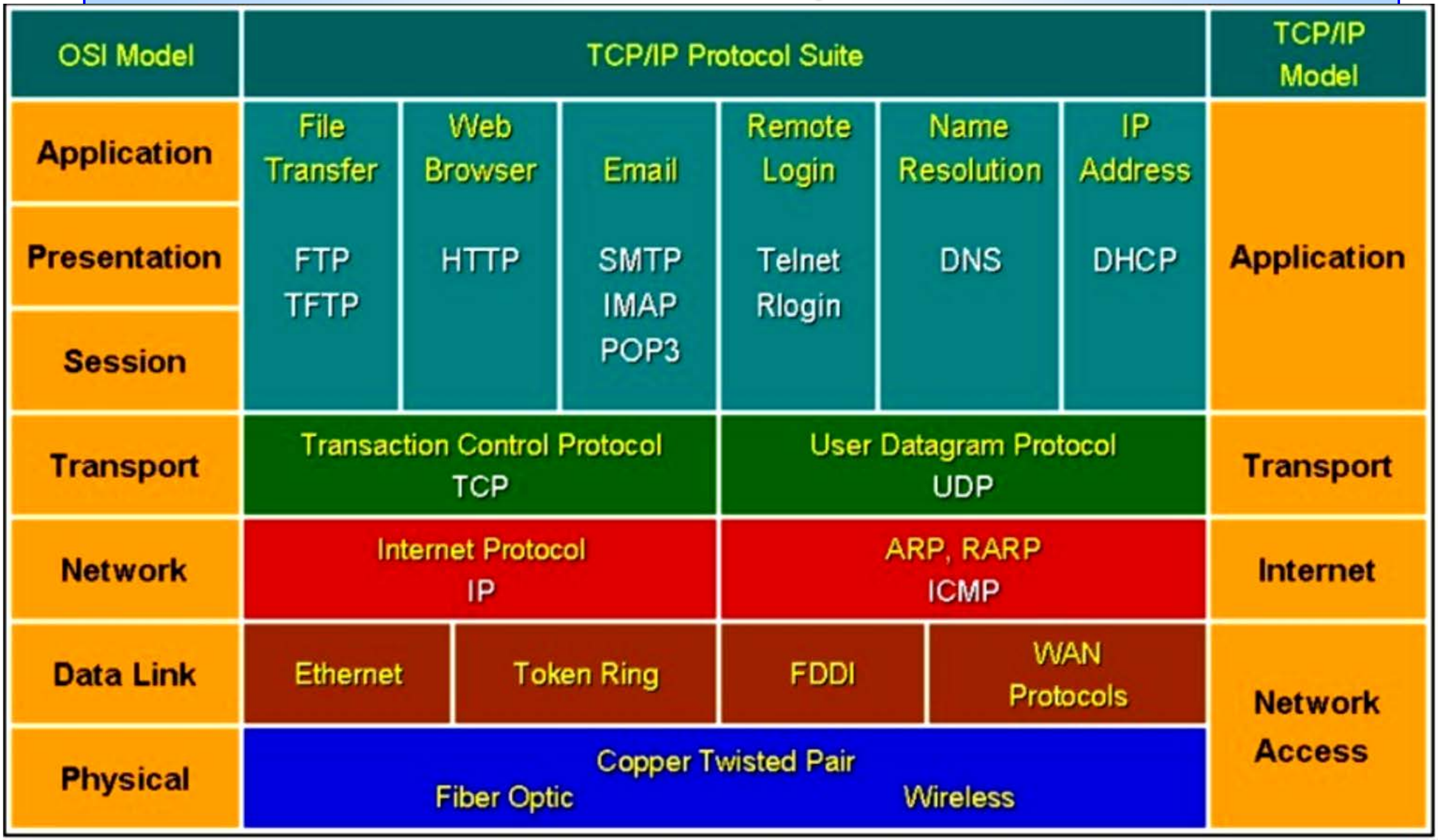


Monitoring Distributed Systems

An essential part of managing large scale, distributed data processing facilities, is a monitoring system that is able to monitor computing facilities, storage systems, networks and a very large number of applications running on these systems in near-real time.

The monitoring information gathered for all the subsystems is essential for design, modelling, debugging, accounting and the development of “higher level services”, that provide decision support and some degree of automated decisions and for maintaining and optimizing workflow in large scale distributed systems.

Network Layers
OSI and TCP/IP Layered Models



Transport Layer

Provide *logical communication* between application processes running on different hosts ***The transport layer is responsible for process-to-process delivery***

- **Datagram messaging service (UDP)** ***It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.***

- **Reliable, in-order delivery (TCP)**
  - **Connection set-up**
  - **Discarding of corrupted packets**
  - **Retransmission of lost packets**
  - **Flow control**
  - **Congestion control**

SNMP - Simple Network Management Protocol

Is a UDP-based network protocol

With SNMP you can monitor hardware devices and software applications. It is also possible to control and configure these devices.

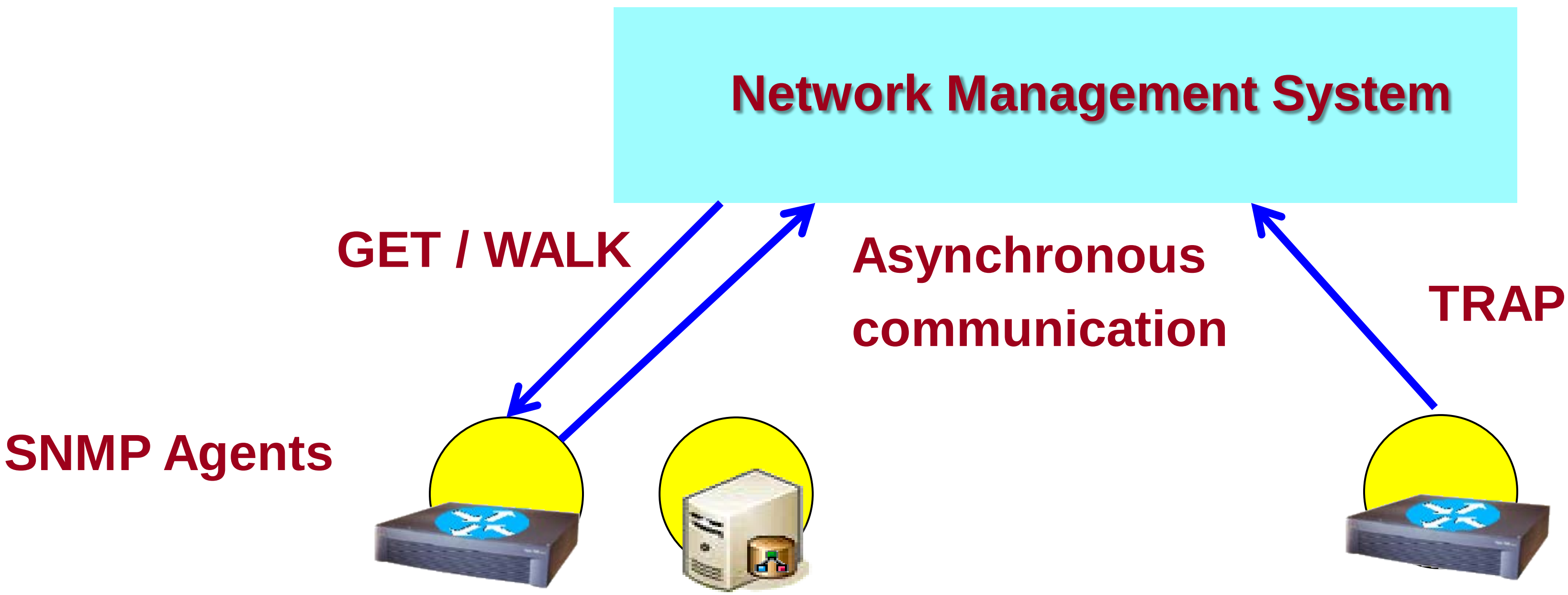
Common devices managed by SNMP include

- **Computer hosts**
- **Routers**
- **Switches**
- **IP telephones**
- **Printers**
- **Software application and services (Data Bases ,Web Servers, Batch Systems ..)**

Main SNMP components

Three key components:

- **Management Information Base (MIB)**
- **SNMP Agent**
- **Network Management System (NMS)**



SNMP Agent

An SNMP agent on a managed device exposes status information as variables. Various data can be made available about the device using SNMP for example:

- System name
- Free memory
- Processor usage
- Uptime
- Traffic on each interface

Can be easily extended to report new parameters

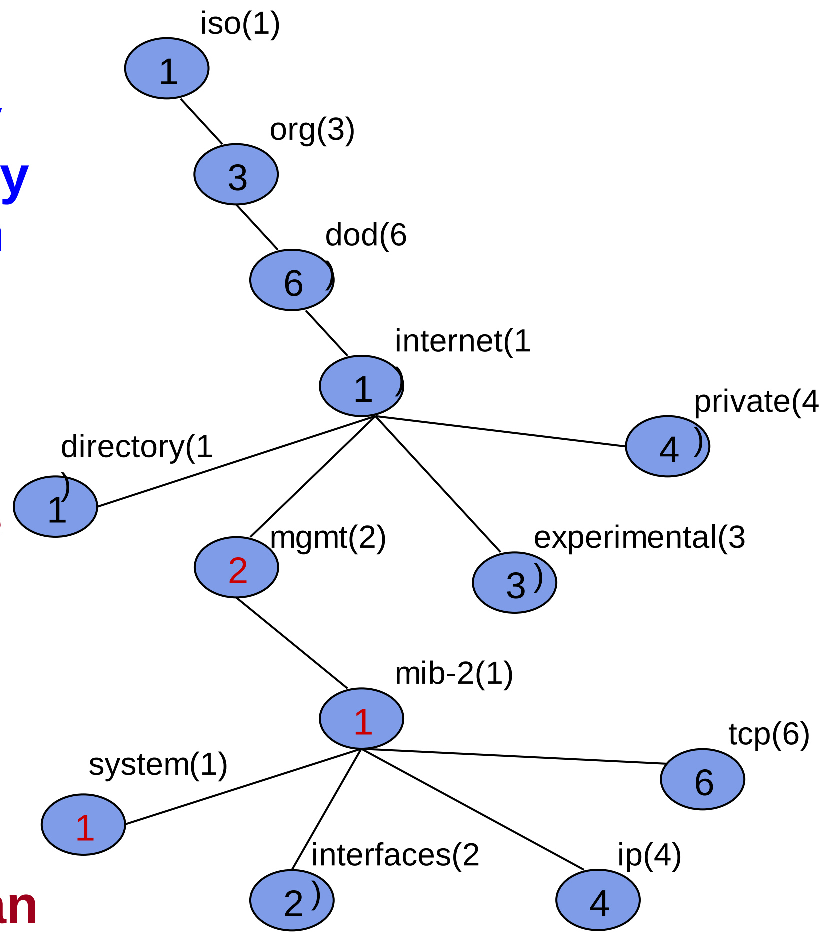
SNMP agent can also accept requests from clients to perform 'active' administration and to modify managed devices configuration

Agent status information and active administration commands are defined in "MIB" files

SNMP MIB - Management information base files

The very basic component of the structure used in case of SNMP is an object. Every information that can be queried through SNMP is looked in terms of an object. They organized in a hieratical structure and every object has an associated unique ID (known as OID).

A group of objects form a MIB.



MIB – File that defines variables that can be read or set on the managed device using SNMP

Defines information about the managed device that can be polled for using SNMP clients

Defines active management settings that can be set or changed to alter the device configuration using SNMP

Trap Mechanism

Trap subscriptions

-Asynchronous notification.

-SNMP agents can be programmed to send trap messages when a certain set of circumstances (or thresholds) arise.

- Managed devices agent will send notifications to listeners

- The Agent sends notifications to all subscribers

SNMP History

SNMP version 1

- **was published in 1988**
- **RFC 1157**

SNMP version 2 added additional functionality

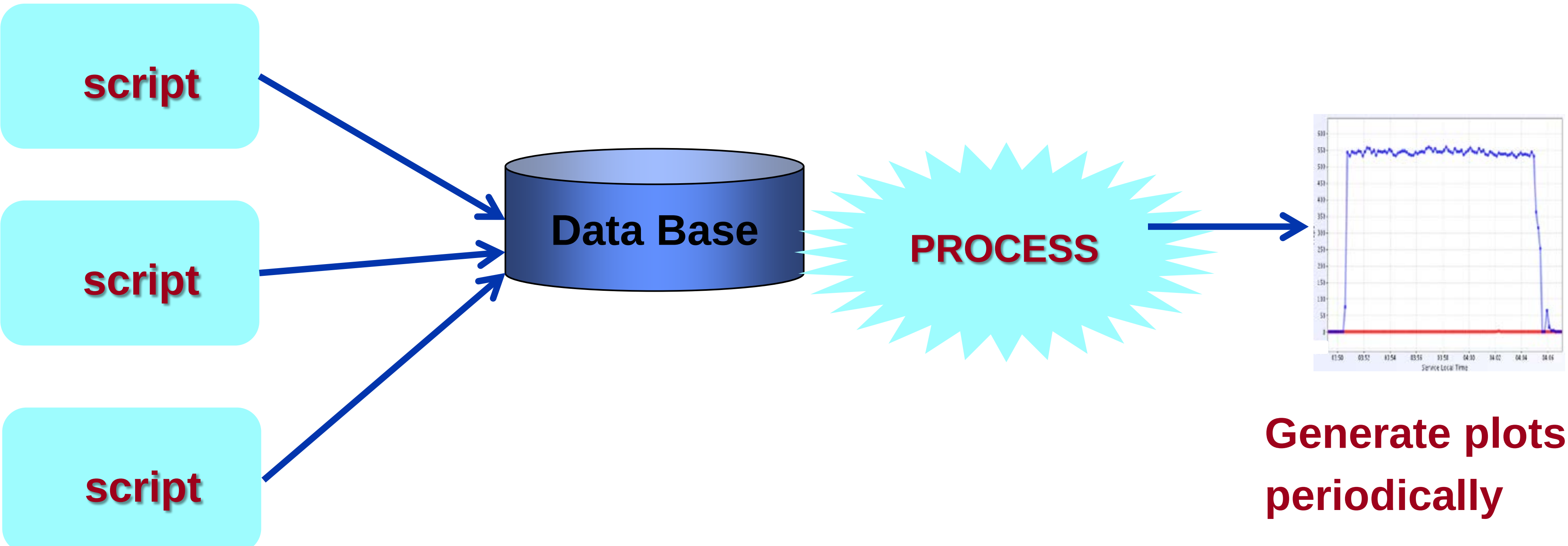
- **RFC 1441 (1993)**

SNMP v3 added security features

- **RFC 3410-3415 (1999)**

Wildly accepted and there are many implementations for most network devices, computers, printers, home electronic devices, and large scale software products.

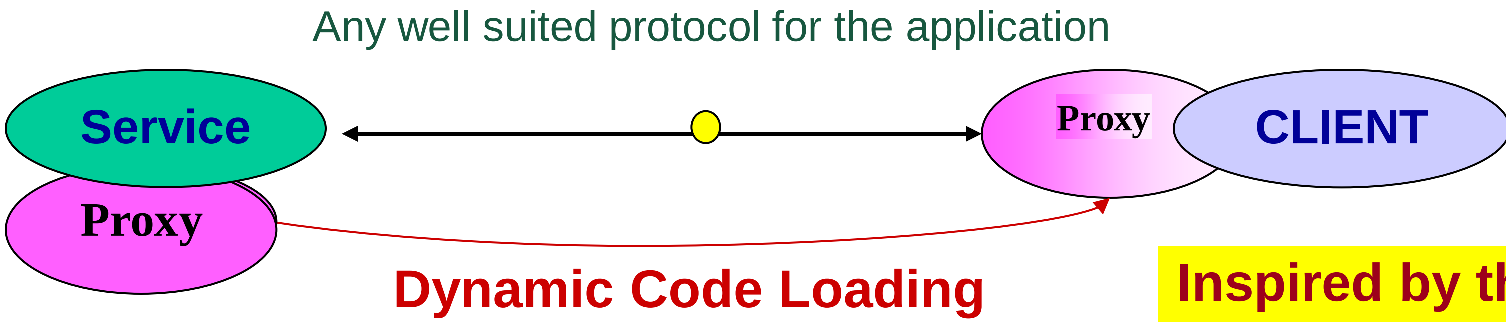
A Simple Monitoring System



But such an architecture

- Does NOT scale
- Can not be used in large scale geographically distributed systems
- Can not be used to control, manage and optimize complex flows

Philosophy: Mobile Code to build Distributed Services



Inspired by the JAVA-JINI Technology

Services can be used dynamically

- **Remote Services** — **Proxy == RMI Stub**
- **Mobile Agents** — **Proxy == Entire Service**
- **"Smart Proxies"** — **Proxy adjusts to the client**

Act as a true dynamic service and provide the necessary functionally to be used by any other services that require such information (Jini, interface to WSDL / SOAP)

- mechanism to dynamically discover all the "Service Units"
- remote event notification for changes in the any system
- lease mechanism for each registered un

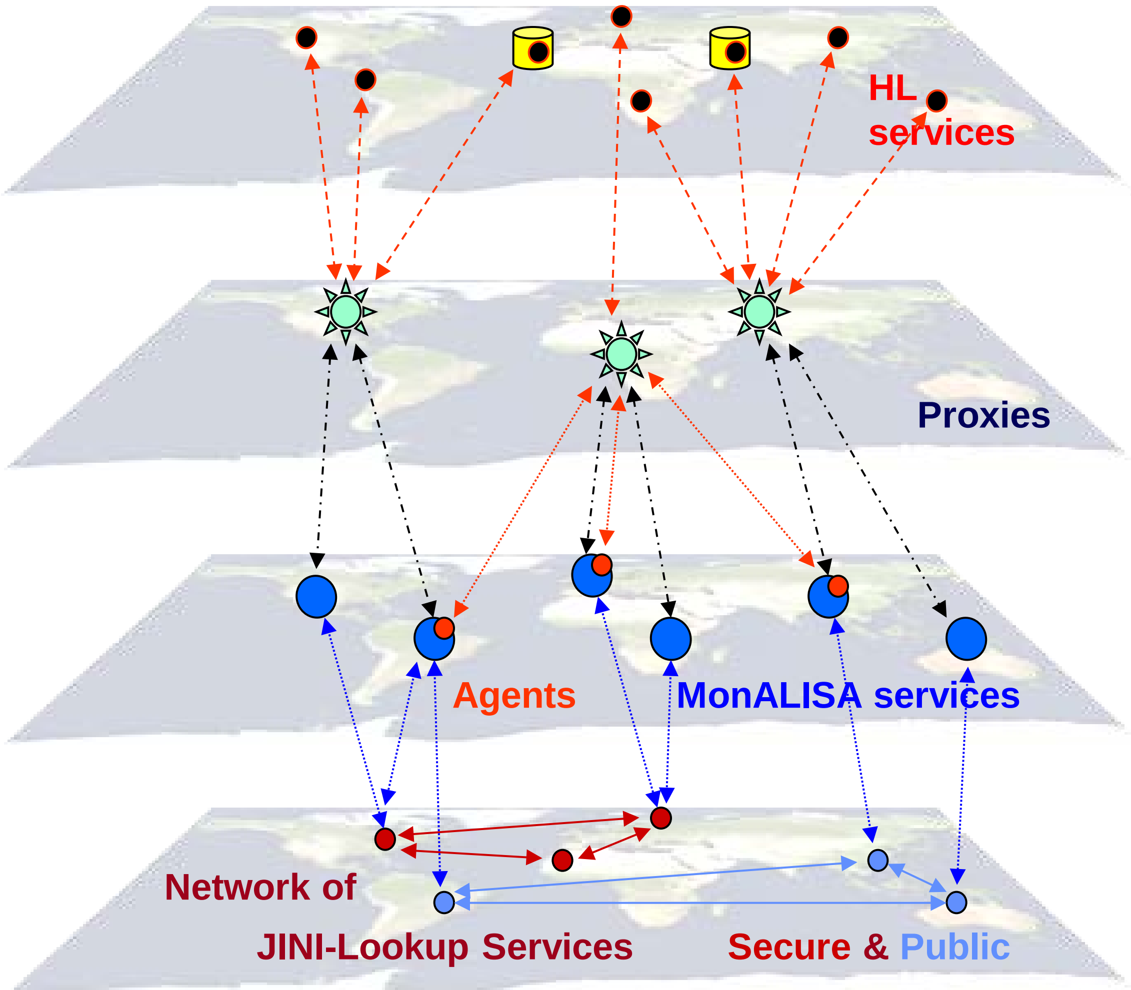
Mobile Code - AGENTS

- Mobile code supports a flexible form of distributed computation where the desired nonlocal computations need not be known in advance at the execution site

ADVATGAES

- Take decisions at the run time
- Flexibility
- Dynamic Deployment
- Easy to reconfigure

The MonALISA Architecture



Regional or Global High Level Services, Repositories & Clients

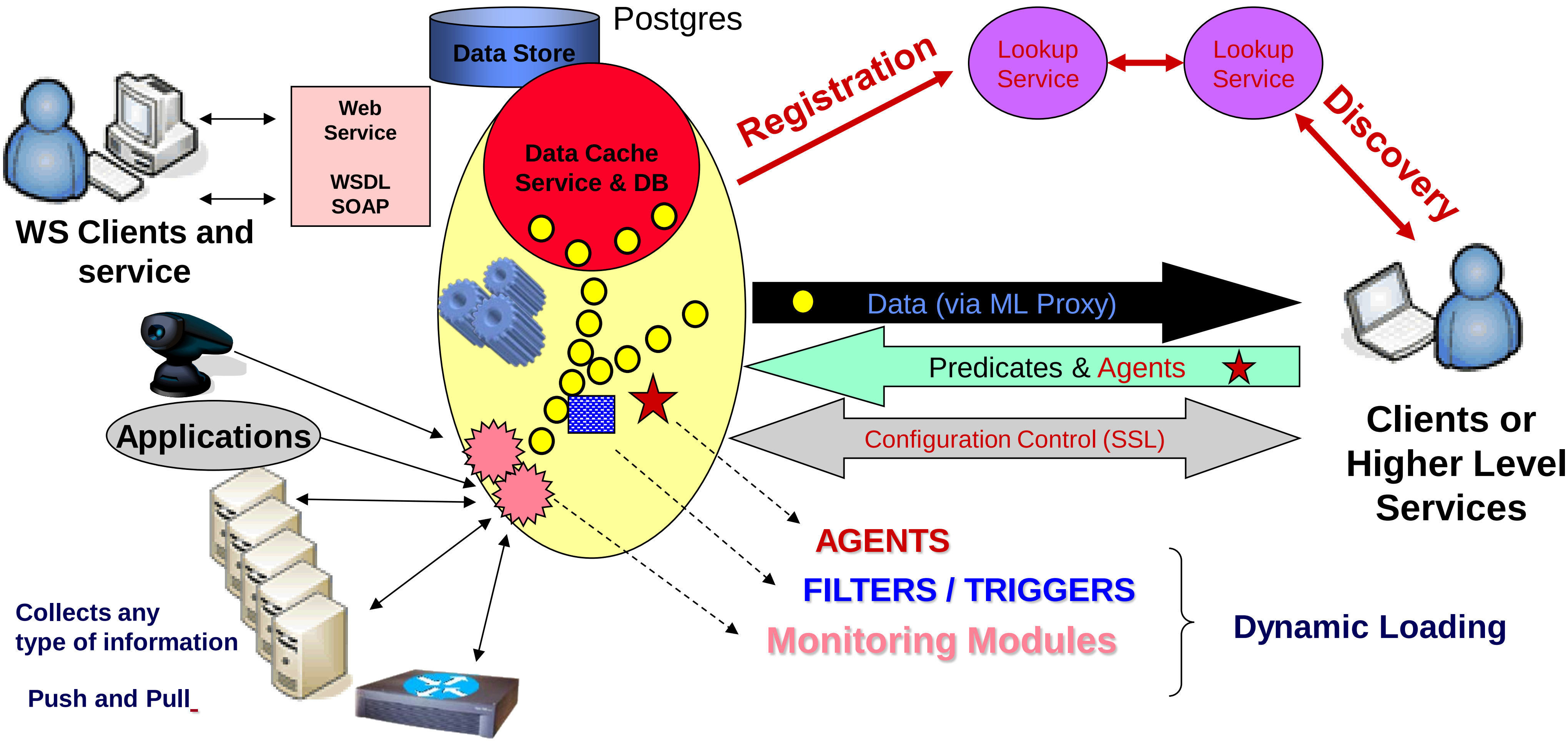
Secure and reliable communication
Dynamic load balancing
Scalability & Replication
AAA for Clients

Distributed System for gathering and analyzing information based on mobile agents:
Customized aggregation, Triggers, Actions

Distributed Dynamic Registration and Discovery-based on a lease mechanism and remote events

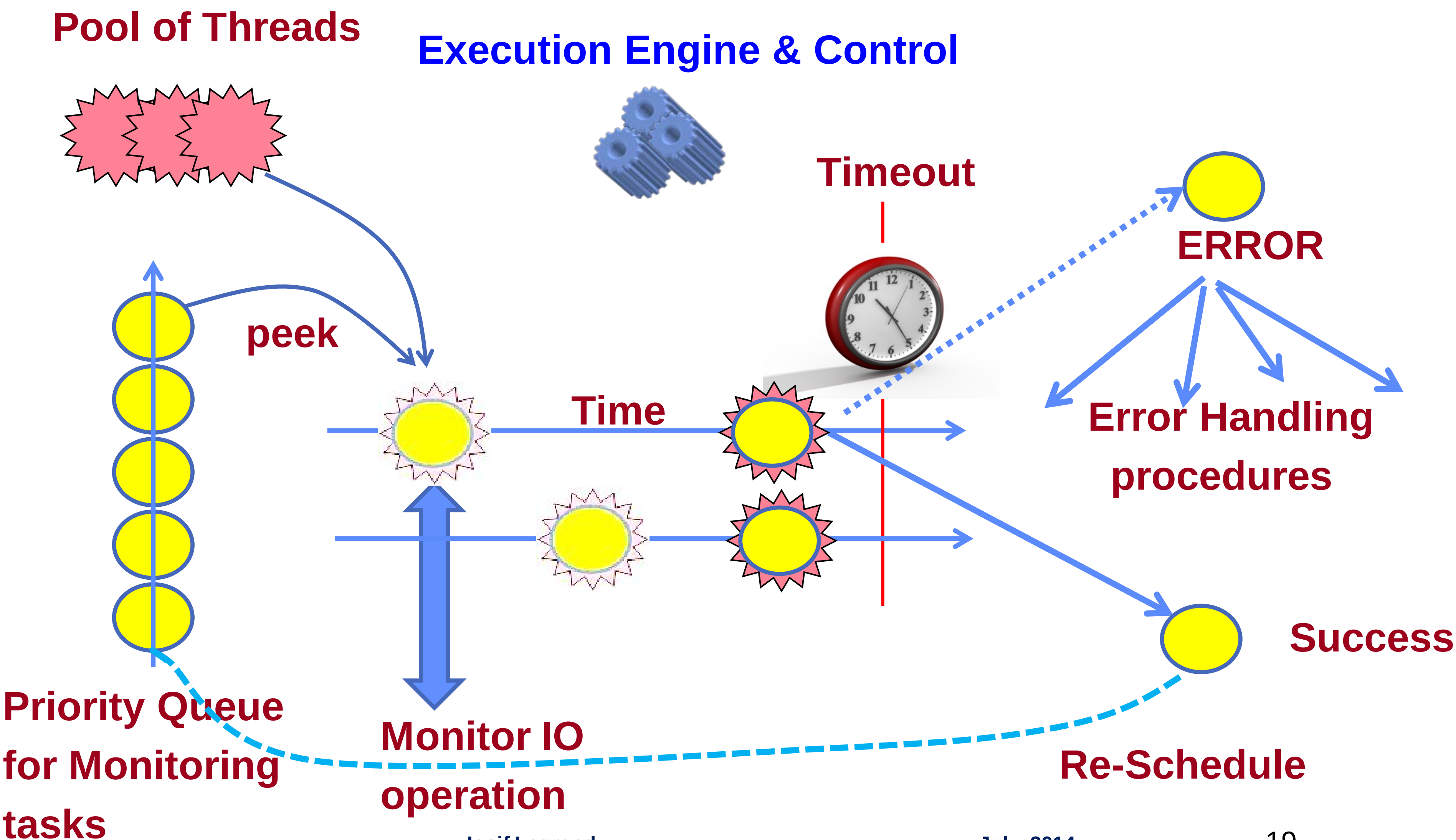
Fully Distributed System with no Single Point of Failure

MonALISA Service & Data Handling

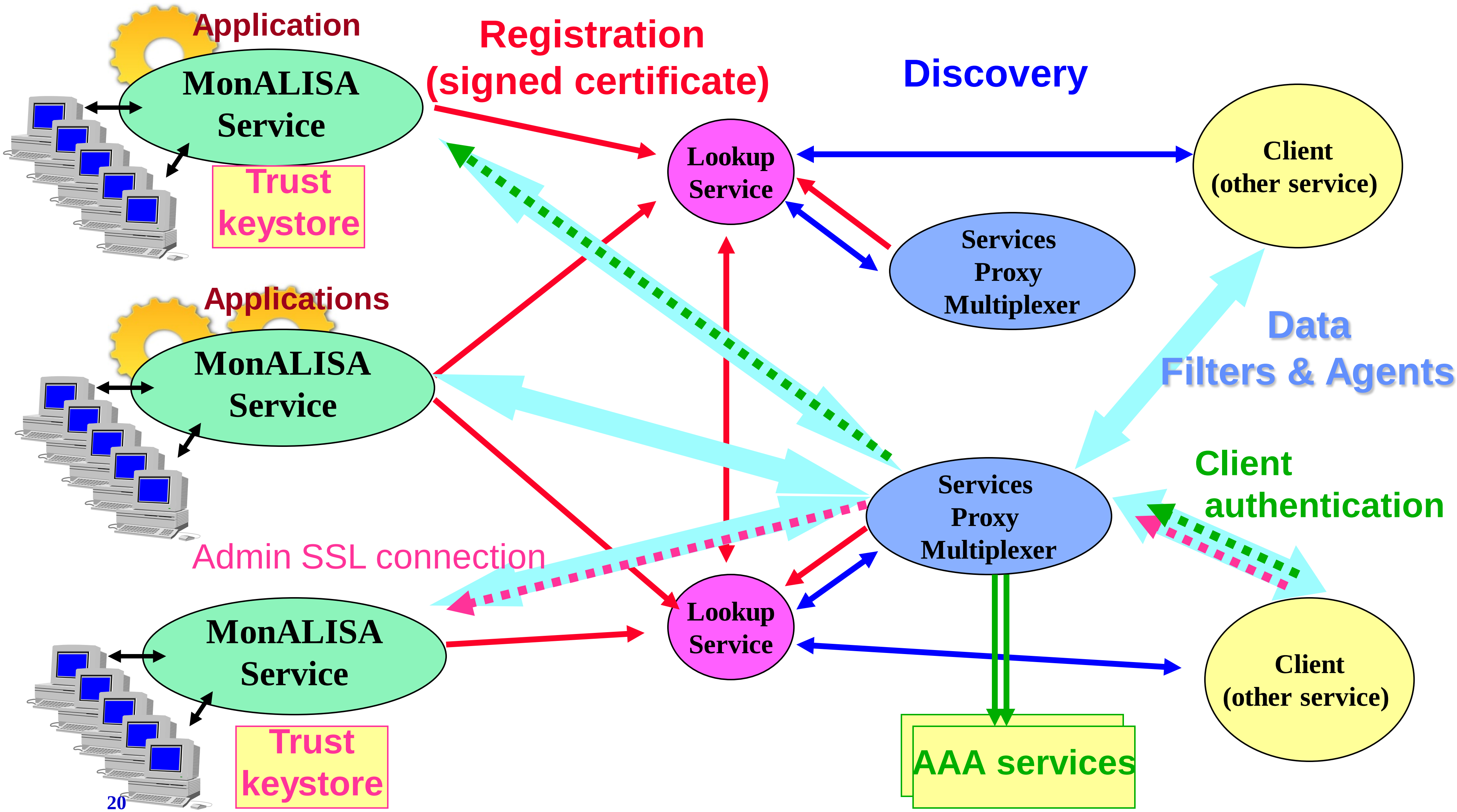


18

Multi-Thread Execution Engine



Registration / Discovery
Admin Access and AAA for Clients



MonALISA collects any type of monitoring information in distributed systems

The MonALISA package includes:

- Local host monitoring (CPU, memory, network traffic , processes and sockets in each state, LM sensors, APC UPSs), log files tailing
- SNMP generic & specific modules
- Condor, PBS, LSF and SGE (accounting & host monitoring), Ganglia
- Ping, tracepath, traceroute, pathload and other network-related measurements
- TL1, Network devices, Ciena, Optical switches
- Calling external applications/scripts that return as output the values
- XDR-formatted UDP messages (ApMon – user's defined information).

New modules can be easily added by implementing a simple Java interface.

Filters can be used to generate new aggregate data.

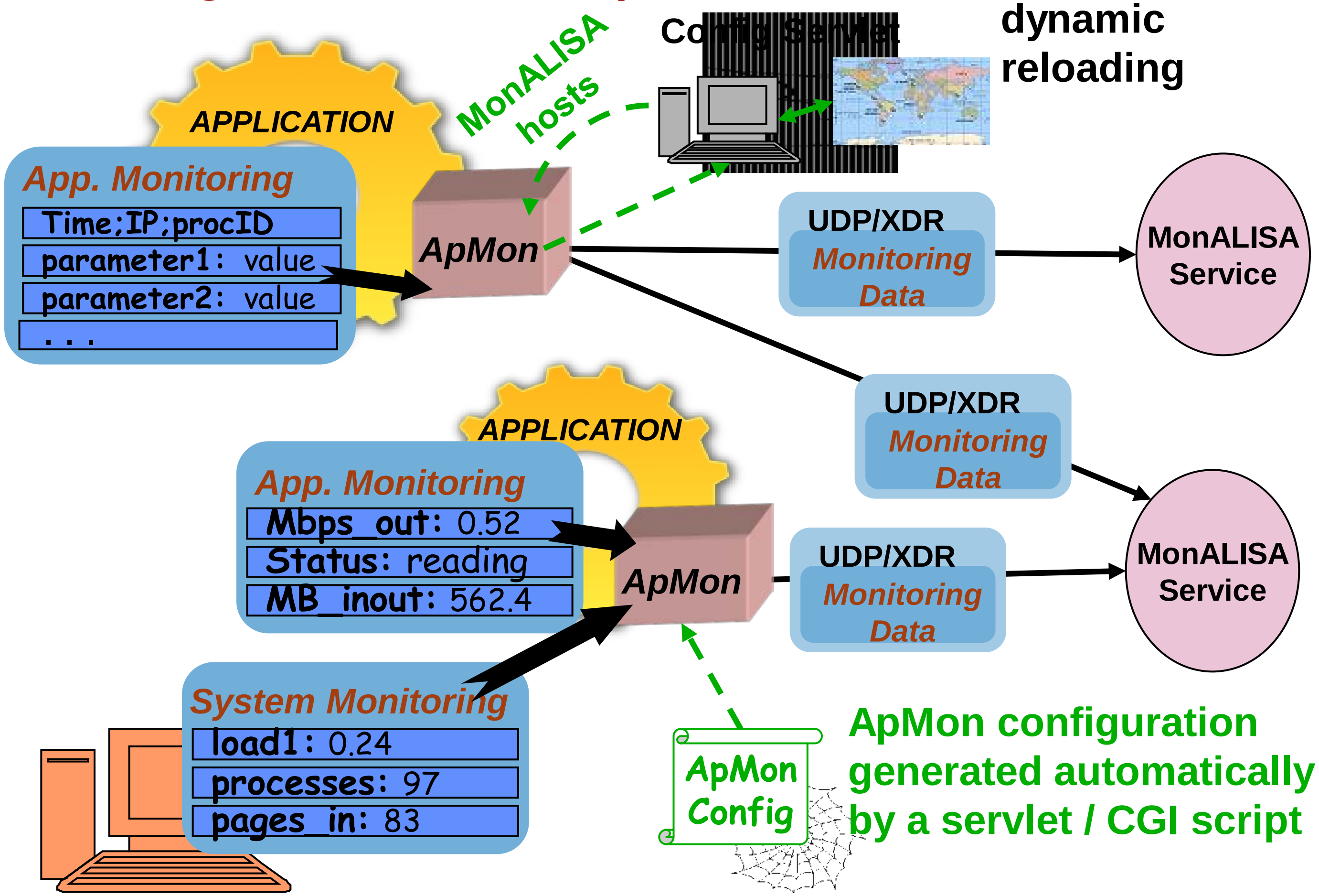
The Service can also react to the monitoring data it receives (actions & alarms).

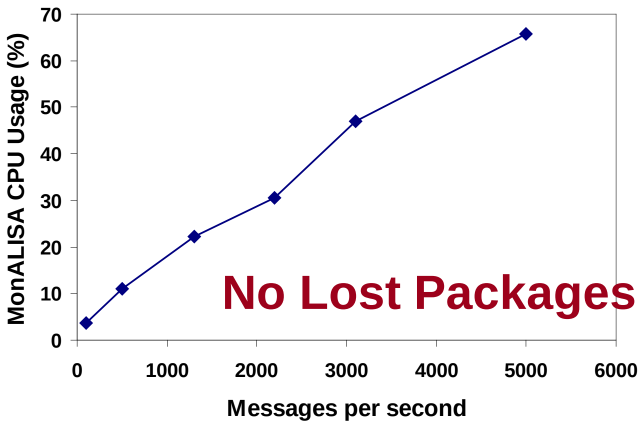
MonALISA can run code as distributed agents for global optimization

ApMon – Application Monitoring

UDP based Library of APIs (C, C++, Java, Perl. Python) that can be used to send any information defined by users or applications to MonALISA services

- ***Flexibility, dynamic configuration, high communication performance***
  - **Automated system monitoring**
  - **Accounting information**



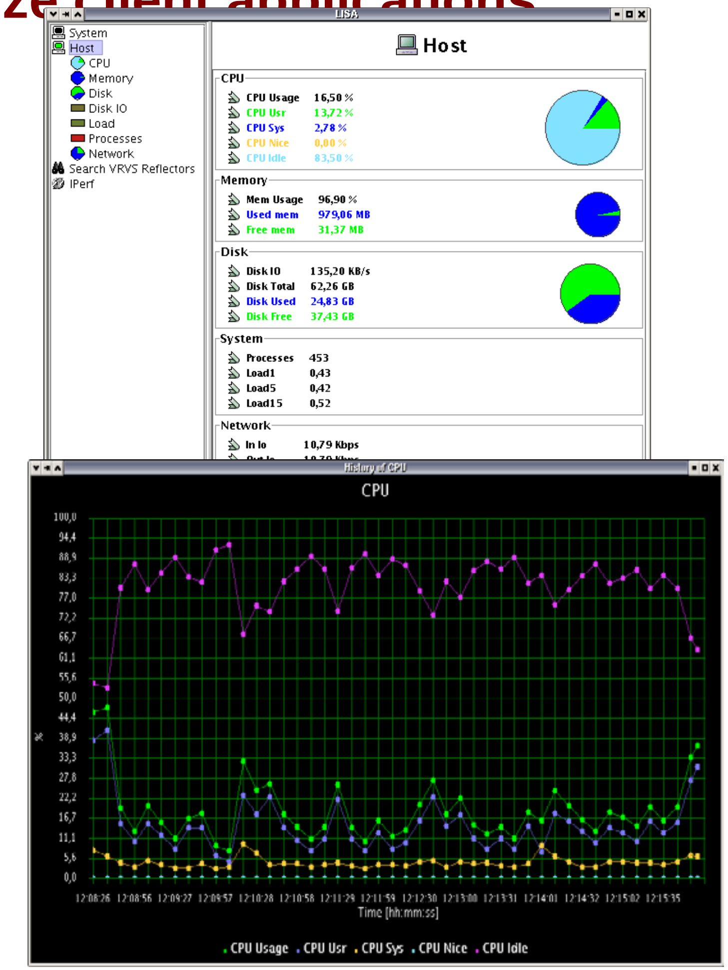


LISA- Localhost Information Service Agent

End To End Monitoring Tool

A lightweight Java Web Start application that provides complete monitoring of the end user systems, the network connectivity and can use the MonALISA framework to optimize client applications

- It is very easy to deploy and install by simply using any browser.
- It detects the system architecture, the operating system and selects dynamically the binary parts necessary on each system.
- It can be easily deployed on any system. It is now used on all versions of Windows, Linux, Mac.
- It provides complete system monitoring of the host computer:
- CPU, memory, IO, disk, …
- Hardware detection
- Main components, Audio, Video equipment,
- Drivers installed in the system
- A user friendly GUI to present all the monitoring information.

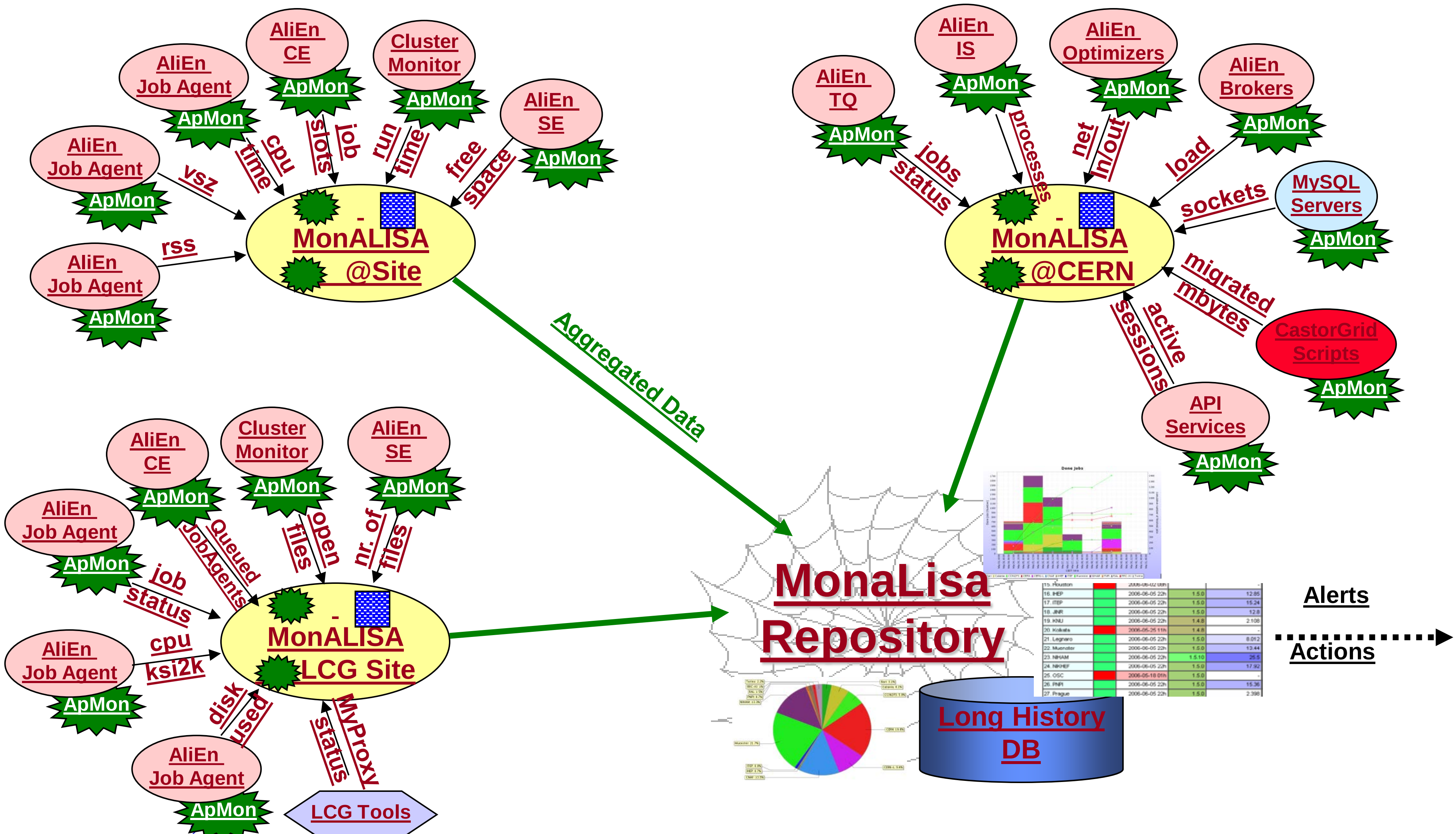


LISA- End User / Client Agent

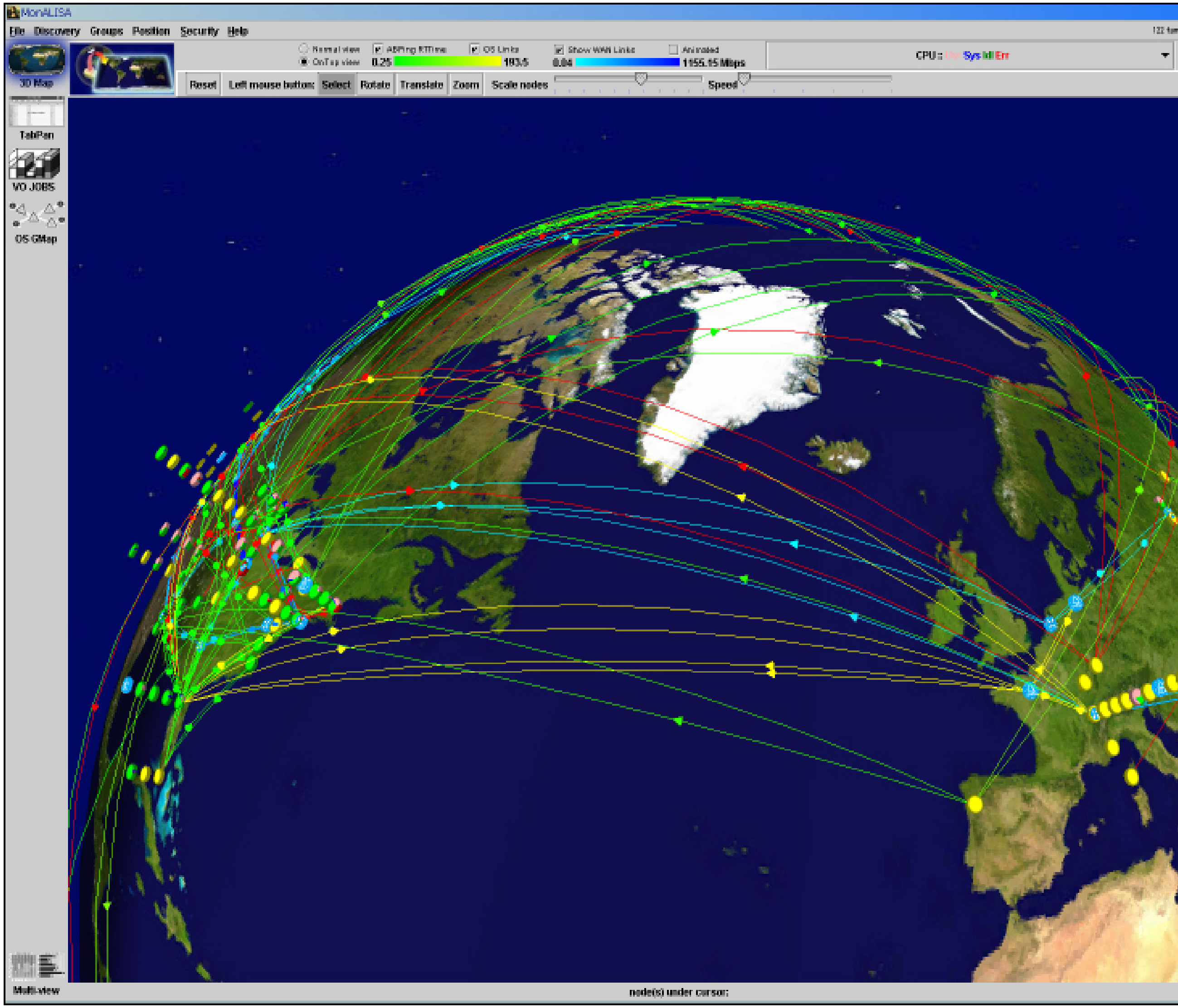
- Authorization
- Service discovery
- Local detection for hardware and software
- Complete monitoring of the system
- End to end performance measurements
- System configuration
- Act as an active listener the events related with the requests generated by its local applications.

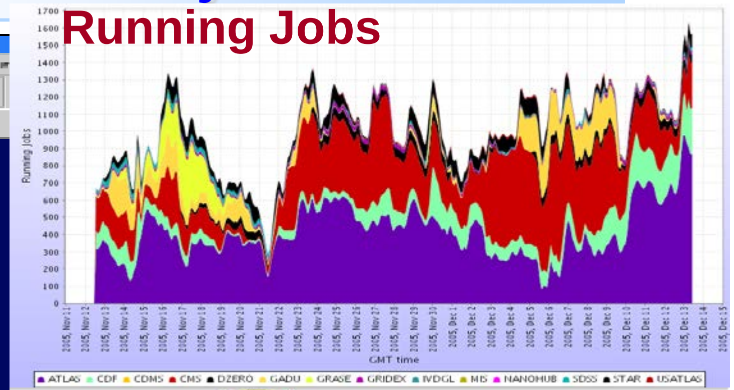
- **The End User Agent implements the secure APIs the applications may use to generate requests and to interact with the Network Service System.**
- **It will continuously receive time estimations to complete the request .**
- **When it is possible, will help to provide correct system & network configuration to the end systems.**

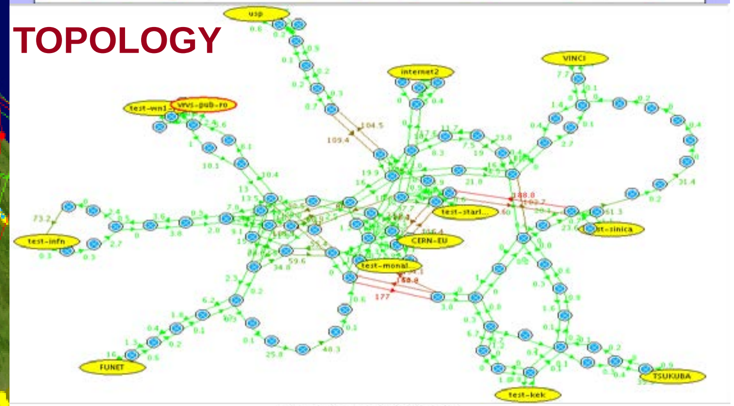
Monitoring architecture in ALICE

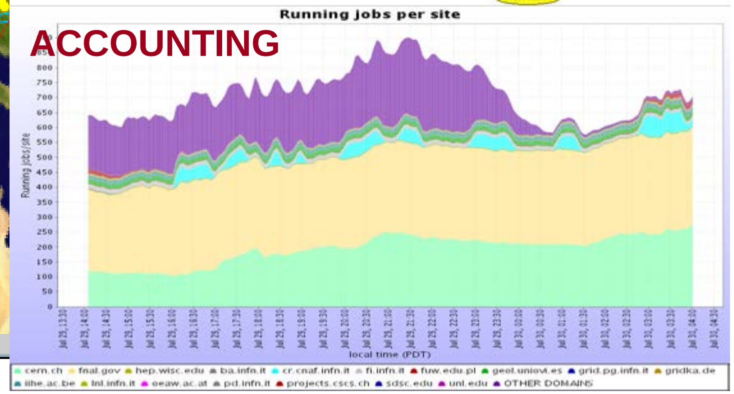


Monitoring Grid sites, Running Jobs, Network Traffic, and Connectivity

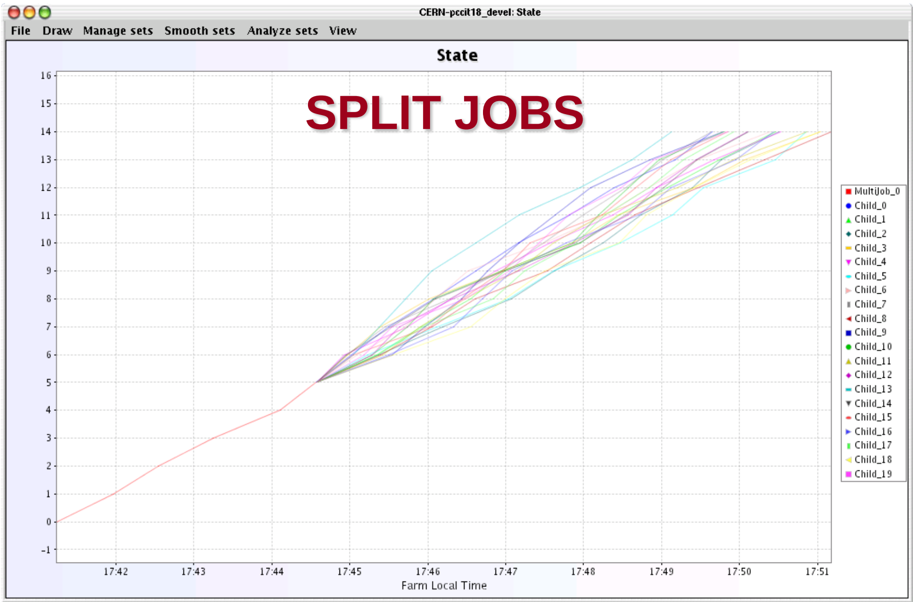


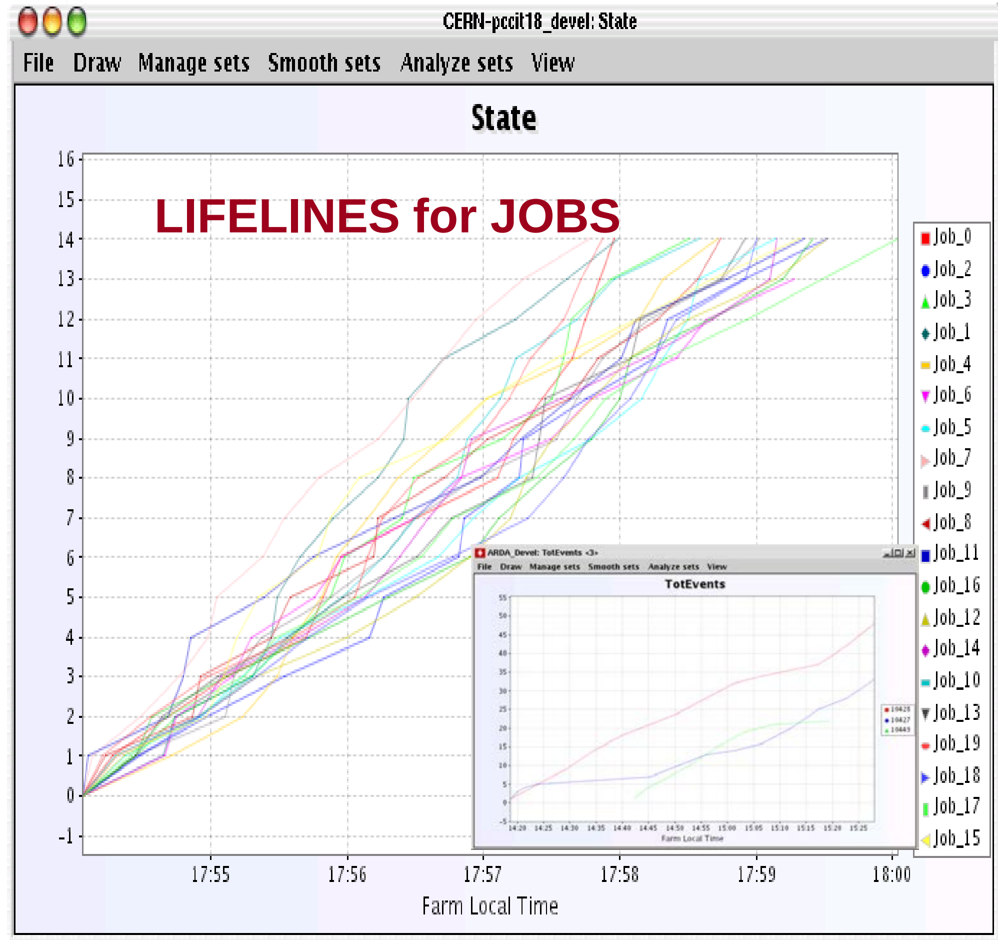




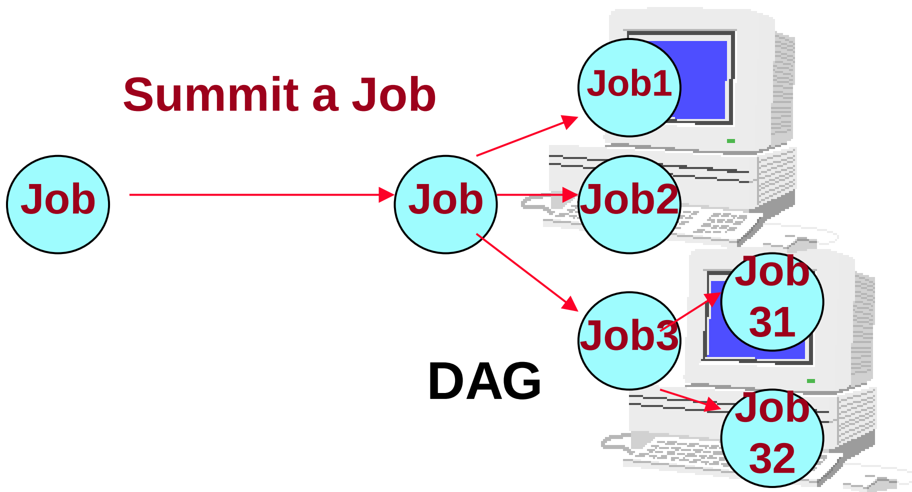


Monitoring the Execution of Jobs and the Time Evolution



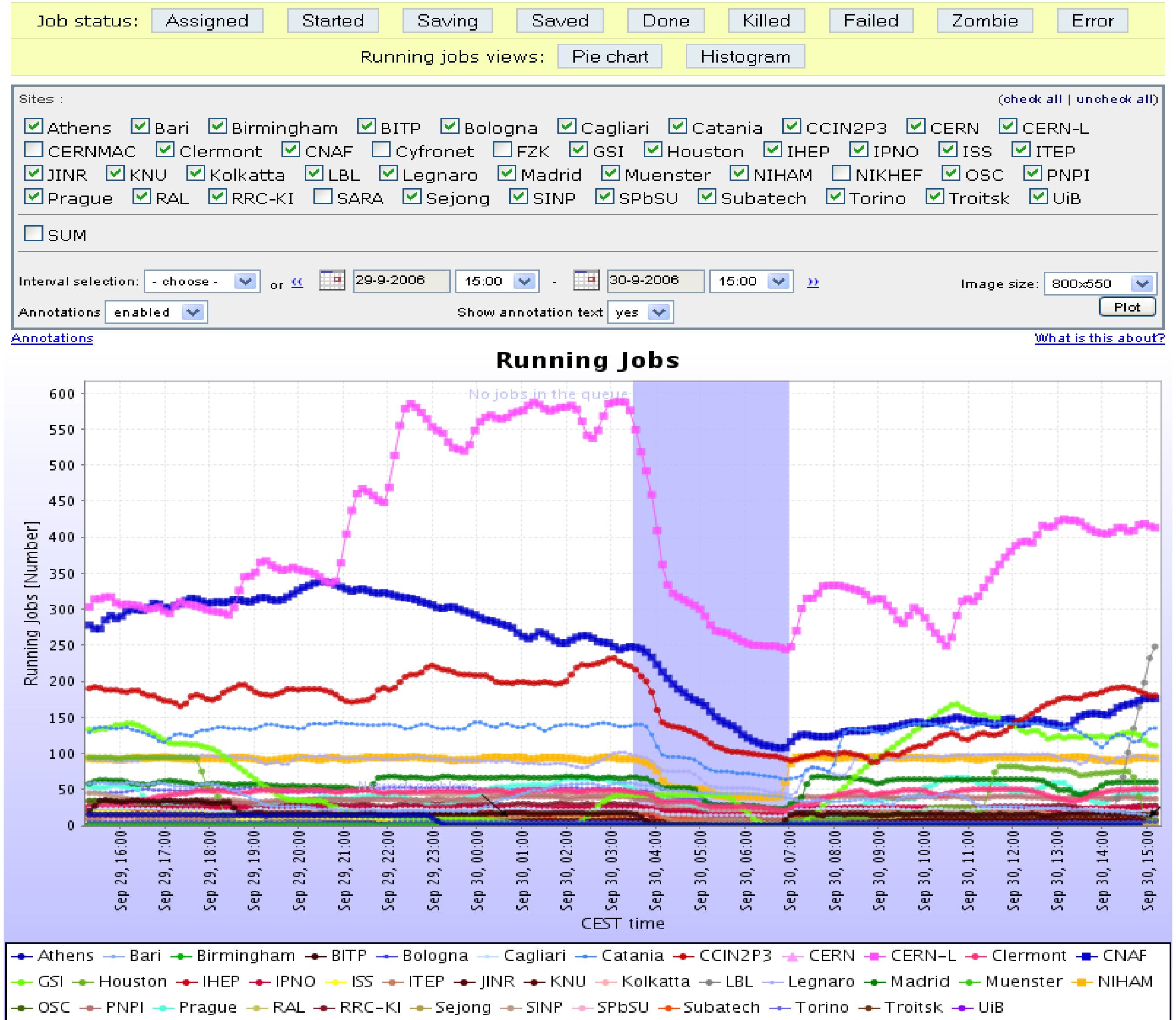


Summit a Job



DAG

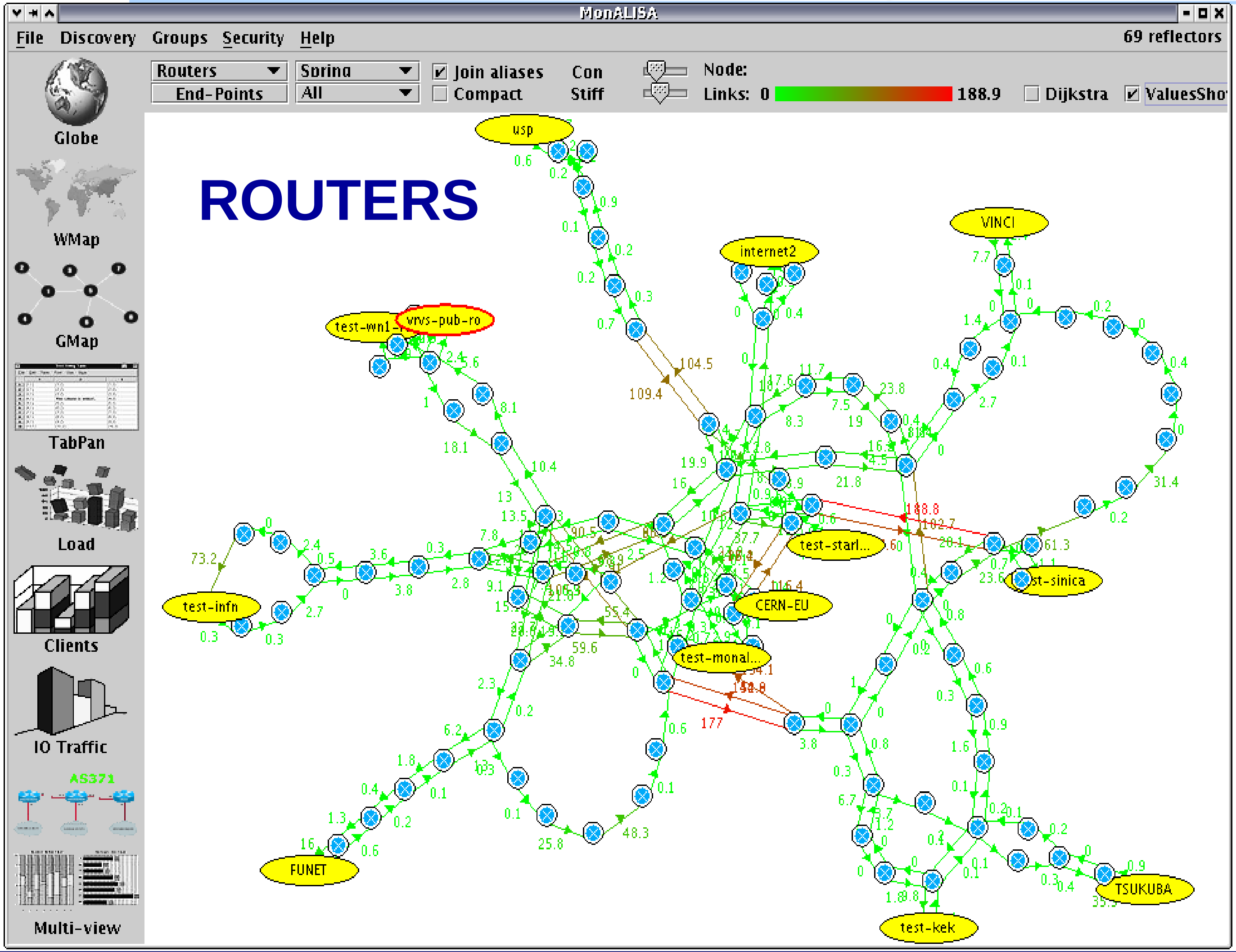
Monitoring in ALICE: jobs, resources, services



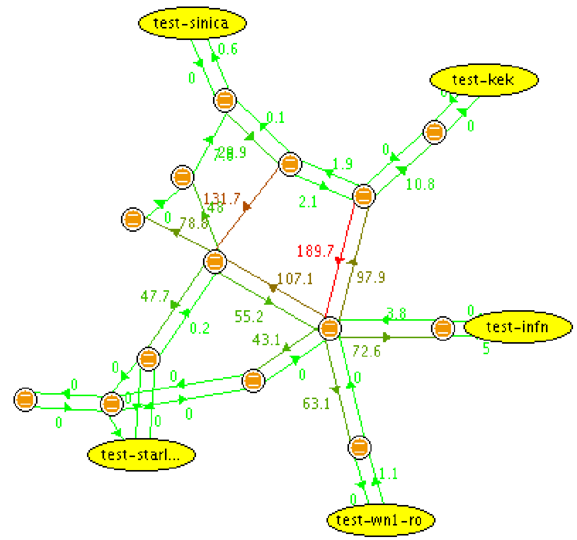
Running Jobs

Farm	Last value	Min	Avg	Max
Athens	0	0	4.746	15
Bari	13	13	37.05	60
Birmingham	0	0	0	0
BITP	16	0	16.02	35
Bologna	5	0	4.291	5
Cagliari	22	13	21.13	23
Catania	135	63	125.2	144
CCIN2P3	180	84	164.1	234
CERN	0	0	0.041	1
CERN-L	413	238	390.8	591
Clermont	50	20	42.56	50
CNAF	176	106	224.7	340
GSI	111	1	66.9	172
Houston	7	0	35.44	94
IHEP	0	0	0	0
IPNO	25	13	23.29	25
ISS	0	0	4.508	20
ITEP	18	3	16.25	20
JINR	7	0	4.365	8
KNU	0	0	0	0
Kolkatta	20	0	15.47	20
LBL	248	0	133	262
Legnaro	93	41	86.98	102
Madrid	60	21	57.56	68
Muenster	0	0	0	0
NIHAM	93	39	87.2	95
OSC	0	0	6.076	34
PNPI	37	9	33.99	40
Prague	48	0	37.88	67
RAL	0	0	0	0
RRC-KI	27	1	21.67	31
Sejong	2	1	6.375	10
SINP	27	12	30.19	45
SPbSU	3	1	2.995	4
Subatech	15	7	13.8	15
Torino	22	17	37.53	53
Troitsk	6	5	7.72	10
UiB	19	4	17.04	24
Total	**1898**		**1776**	

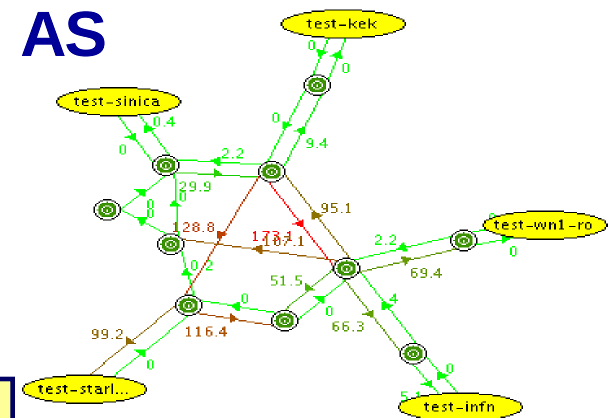
Monitoring Network Topology (L3), Latency, Routers



NETWORKS

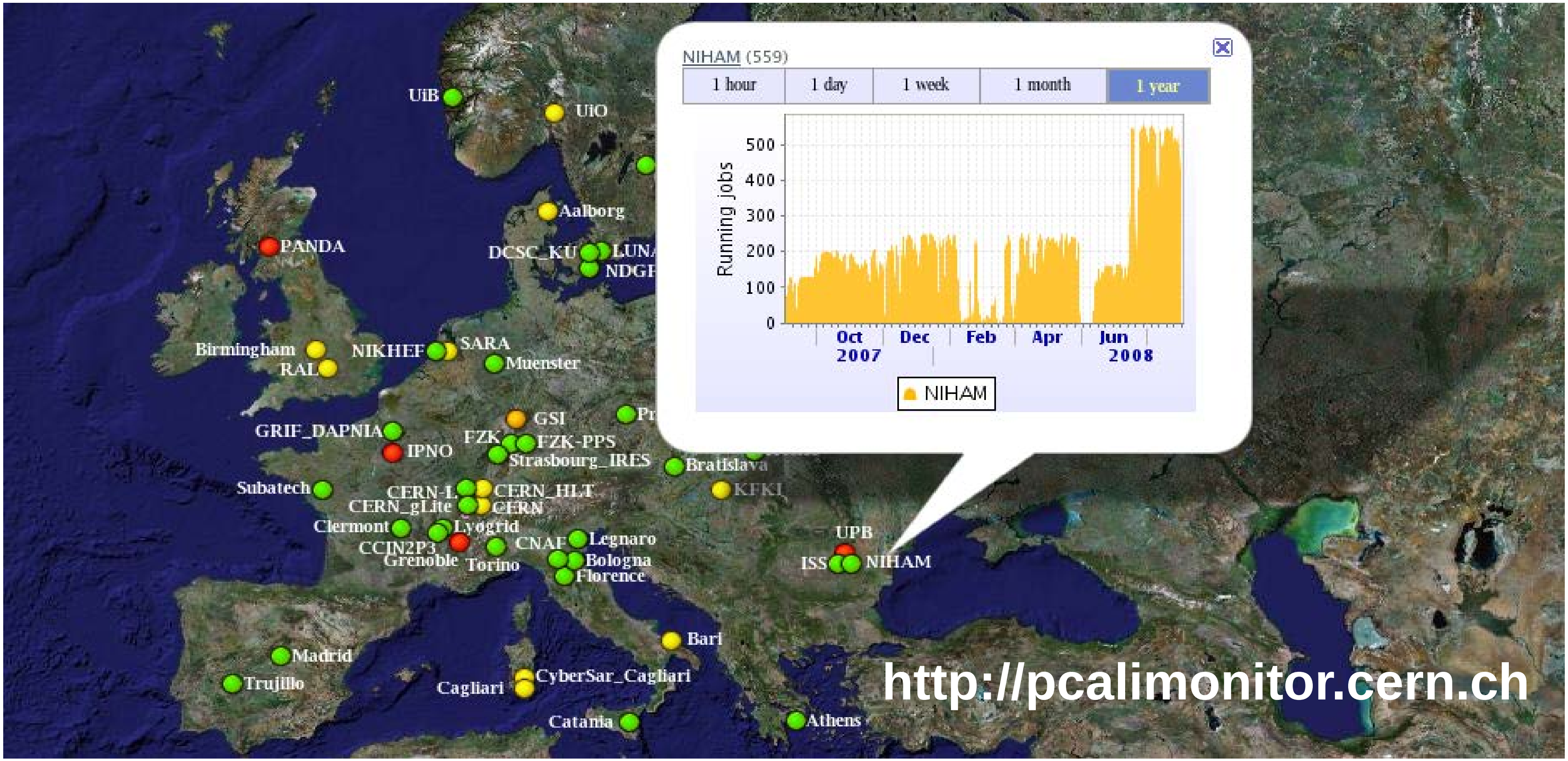


AS

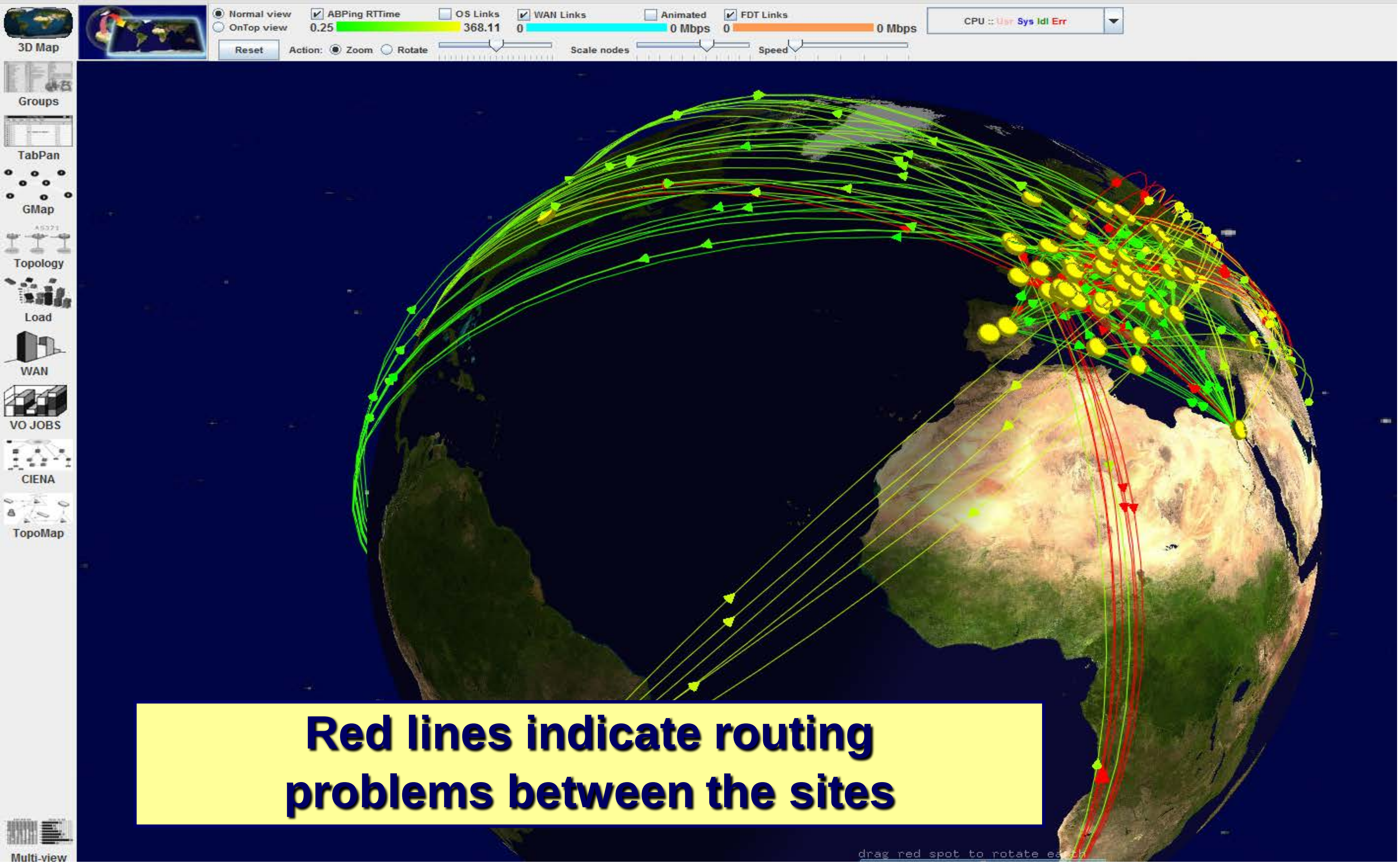


Real Time Topology Discovery & Display

ALICE : Global Views, Status & Jobs



Real Time Ping Measurements between ALICE Sites

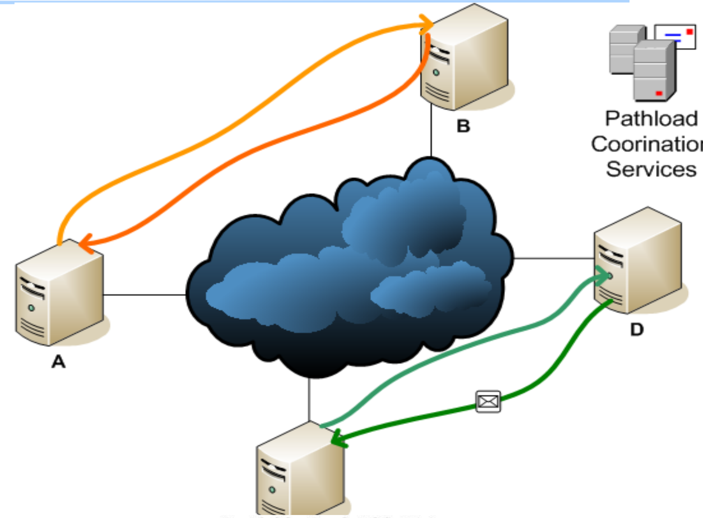


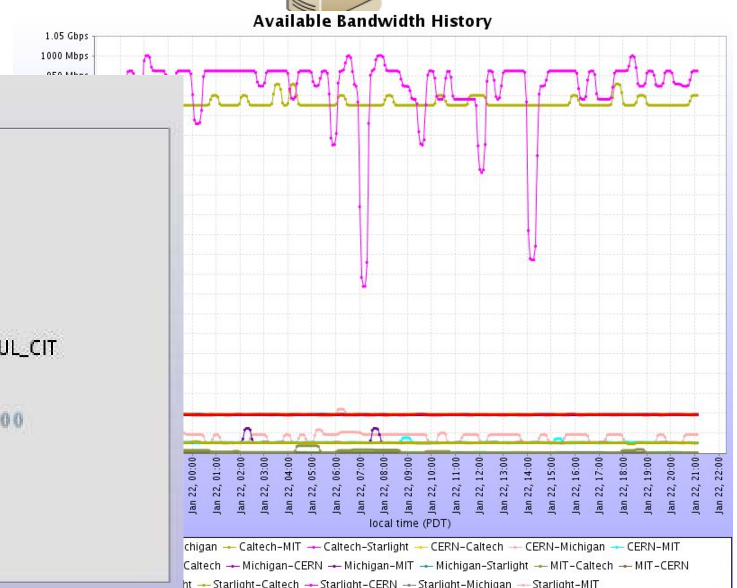
Red lines indicate routing problems between the sites

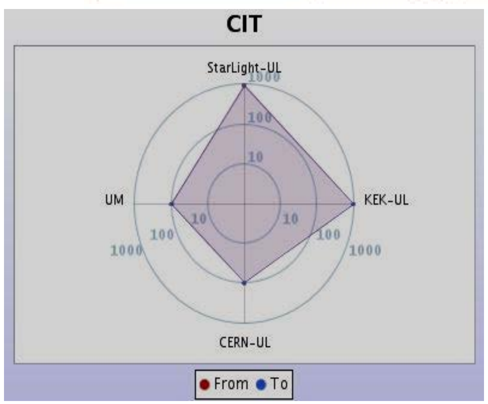
Available Bandwidth Measurements

Available bandwidth measurements using pathload

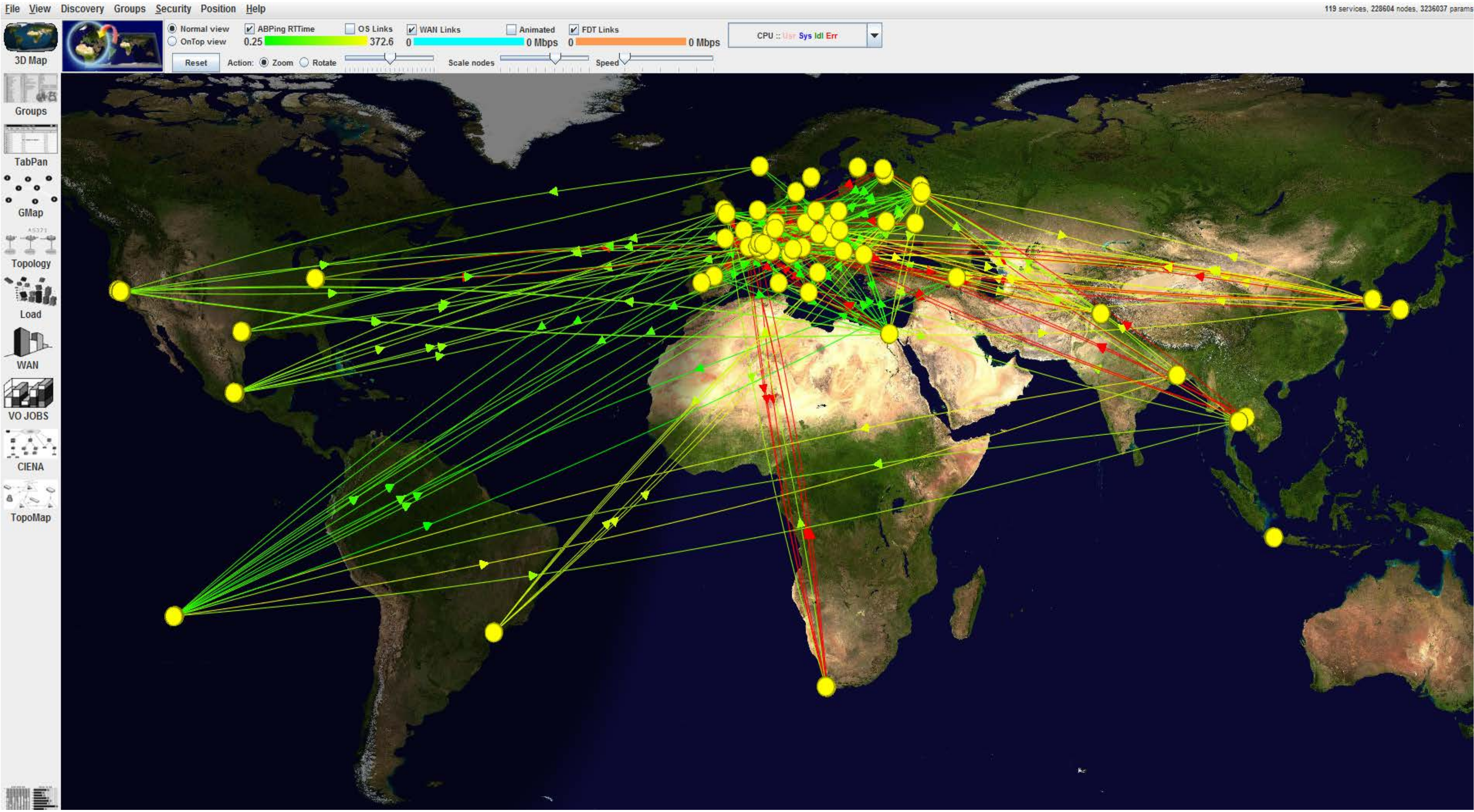
Available bandwidth between UL sites (average)					
Site (from->to)	CERN-UL	KEK-UL	SPRACE-UL	UL_CIT	UM
CERN-UL	-	794.4 Mbps	97.37 Mbps	97.73 Mbps	97.85 Mbps
KEK-UL	750 Mbps	-	96.32 Mbps	993.2 Mbps	96.34 Mbps
StarLight-UL	97.4 Mbps	97.53 Mbps	-	875 Mbps	97.5 Mbps
UL_CIT	97.5 Mbps	993.2 Mbps	876 Mbps	-	96.63 Mbps
UM	97.48 Mbps	96.84 Mb	[illegible]	[illegible]	[illegible]
Min	97.4 Mbps	97.53 Mb	[illegible]	[illegible]	[illegible]
Max	750 Mbps	993.2 Mb	[illegible]	[illegible]	[illegible]



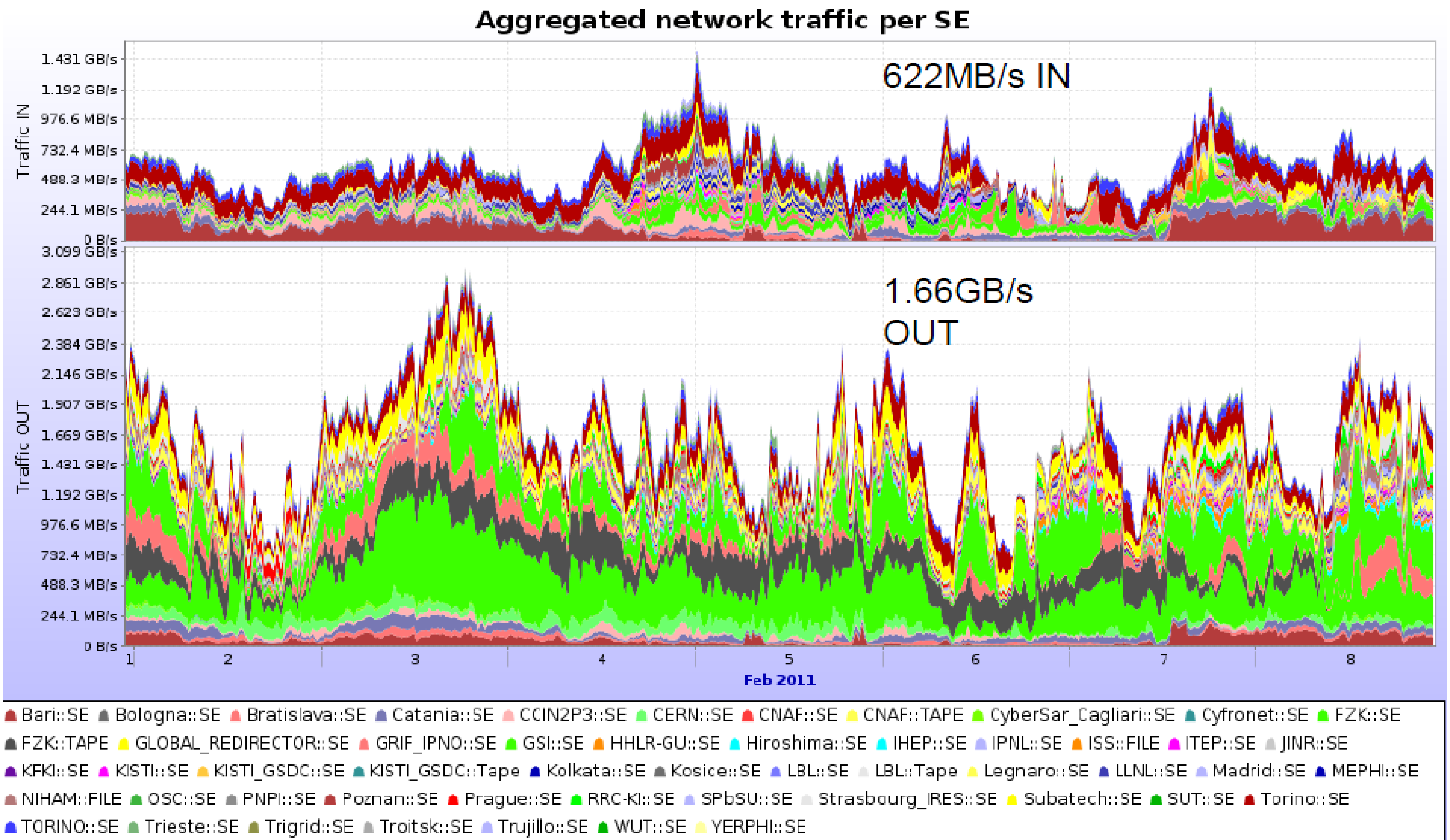




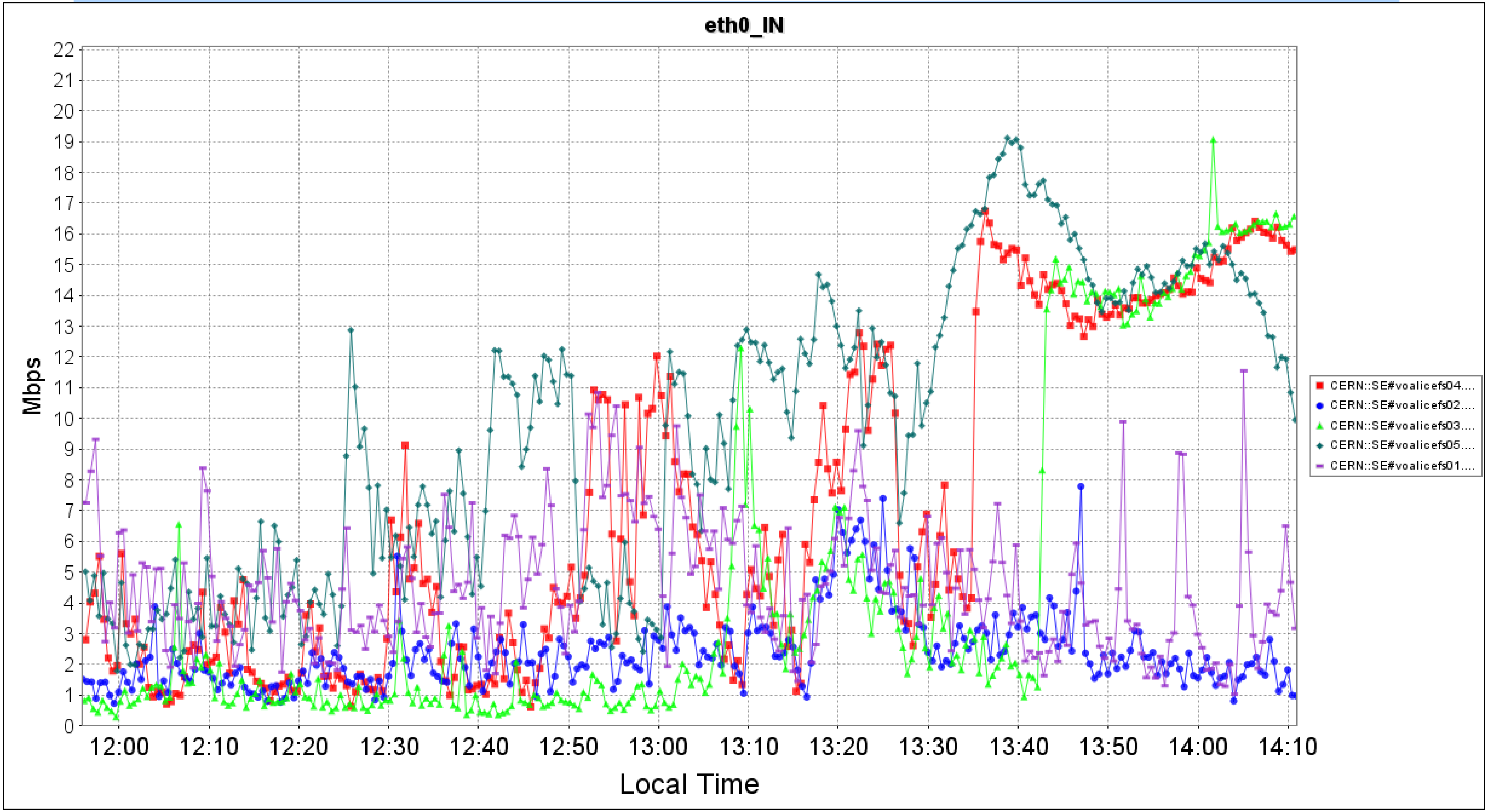
Connectivity map between ALIICE siites



Monitoring in ALICE: Xrootd servers

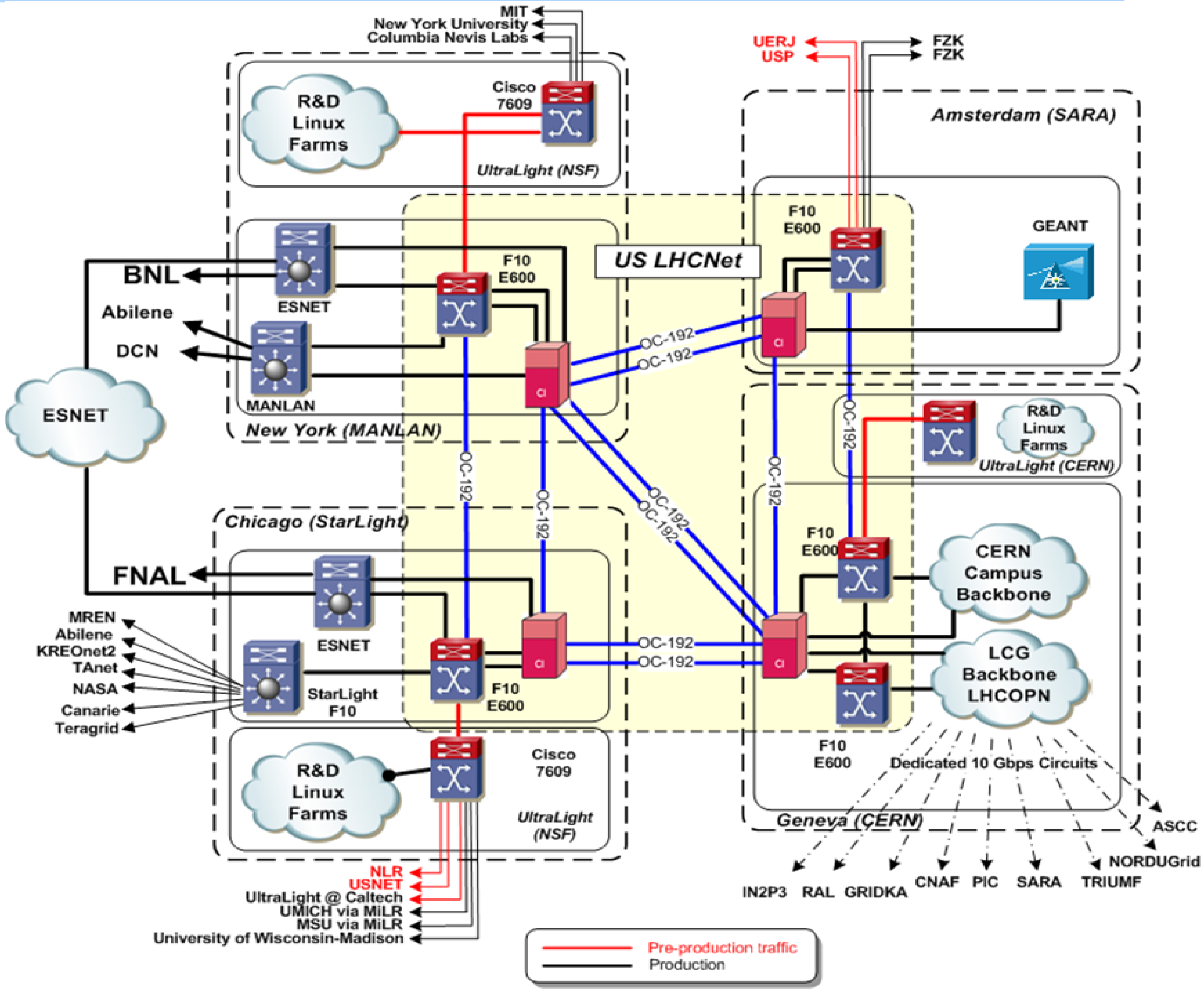


IO Traffic IN on the SE- CERN servers



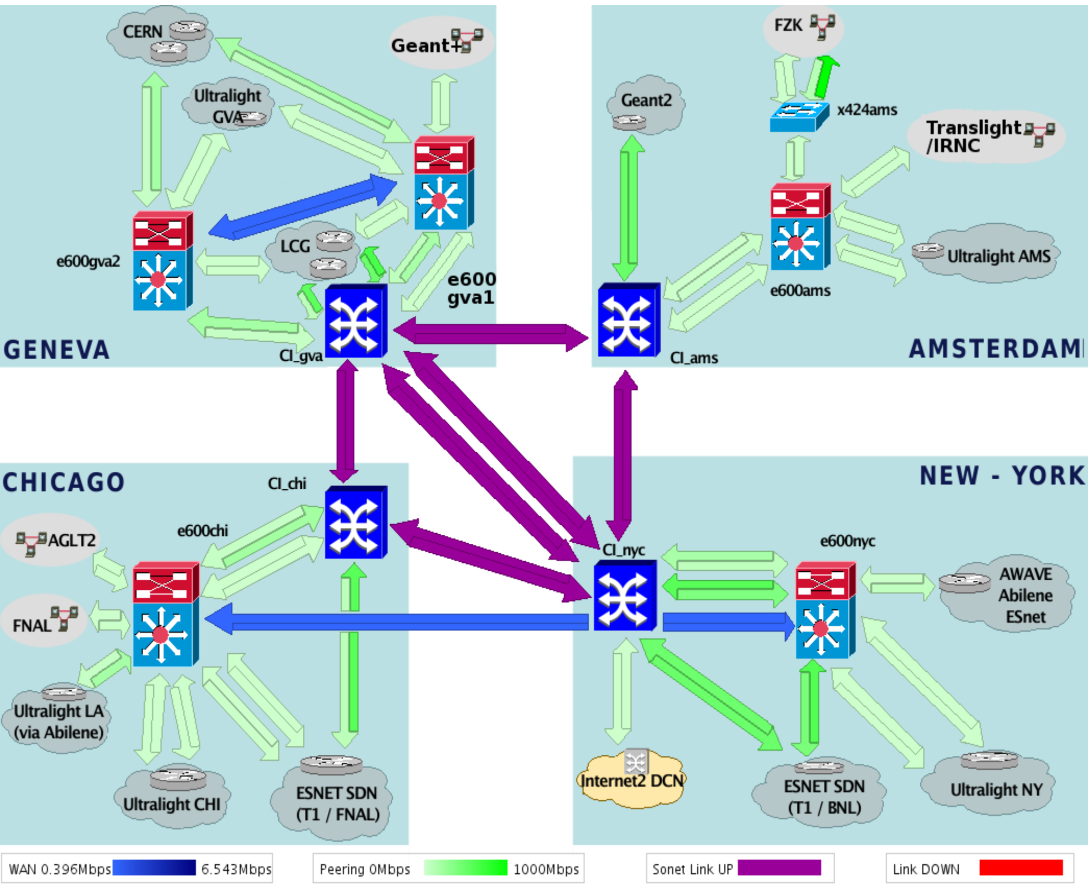
USLHCNet: High-speed trans-Atlantic network

- CERN to US
  - FNAL
  - BNL
- 6 x 10G links
- 4 PoPs
  - Geneva
  - Amsterdam
  - Chicago
  - New York
- The core is based on Ciena CD/CI (Layer 1.5)
- Virtual Circuits

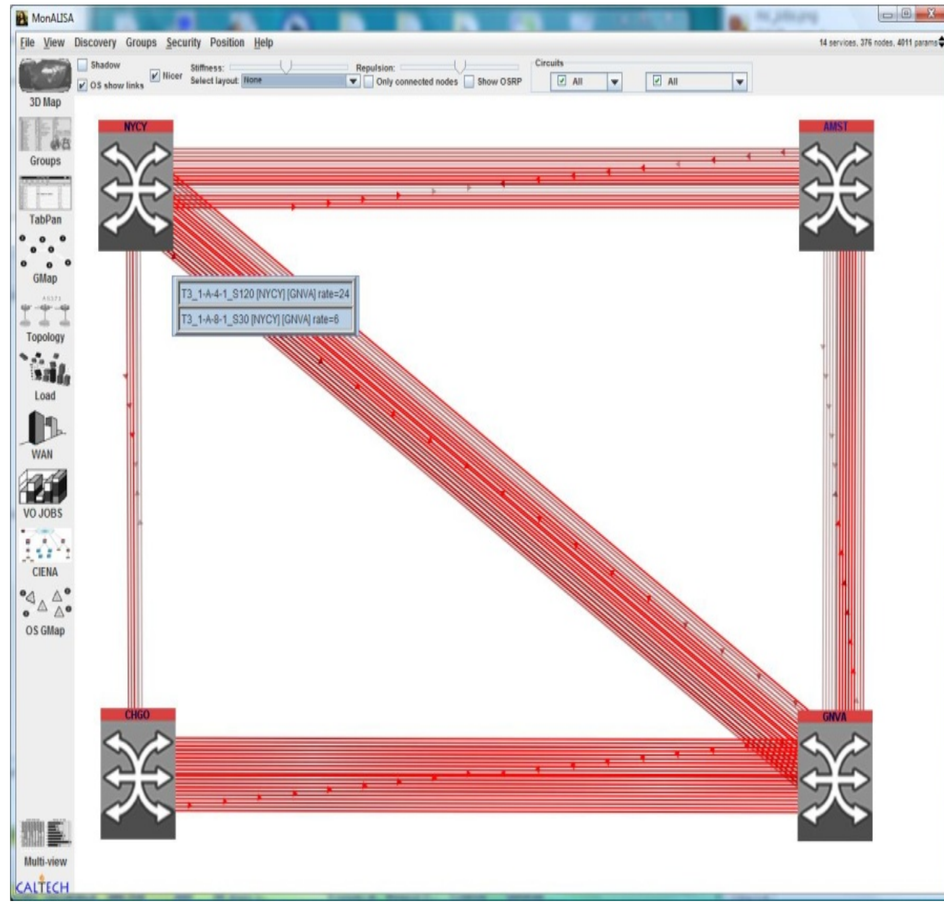


Monitoring USLHCNet Topology

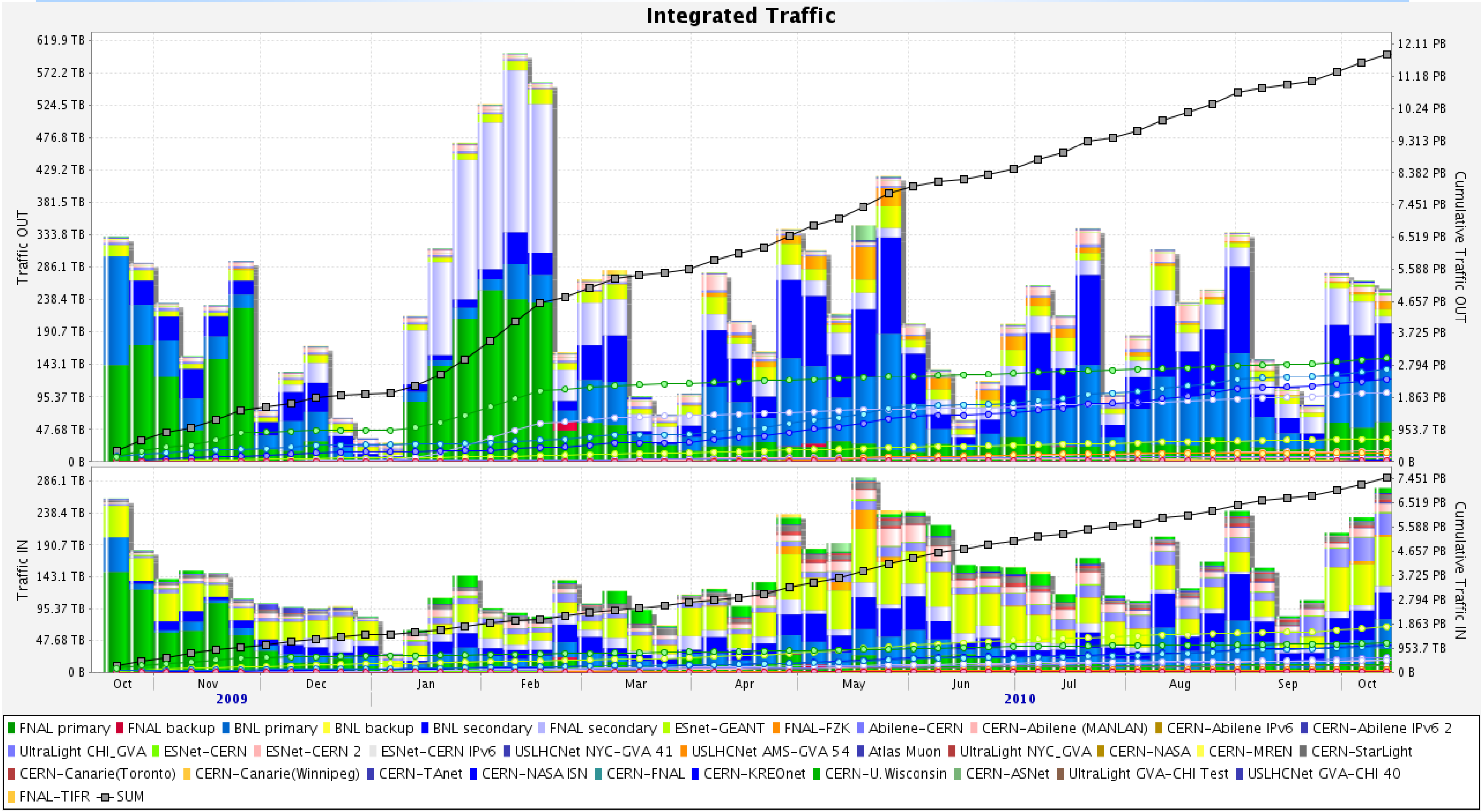
Topology & Status & Peering



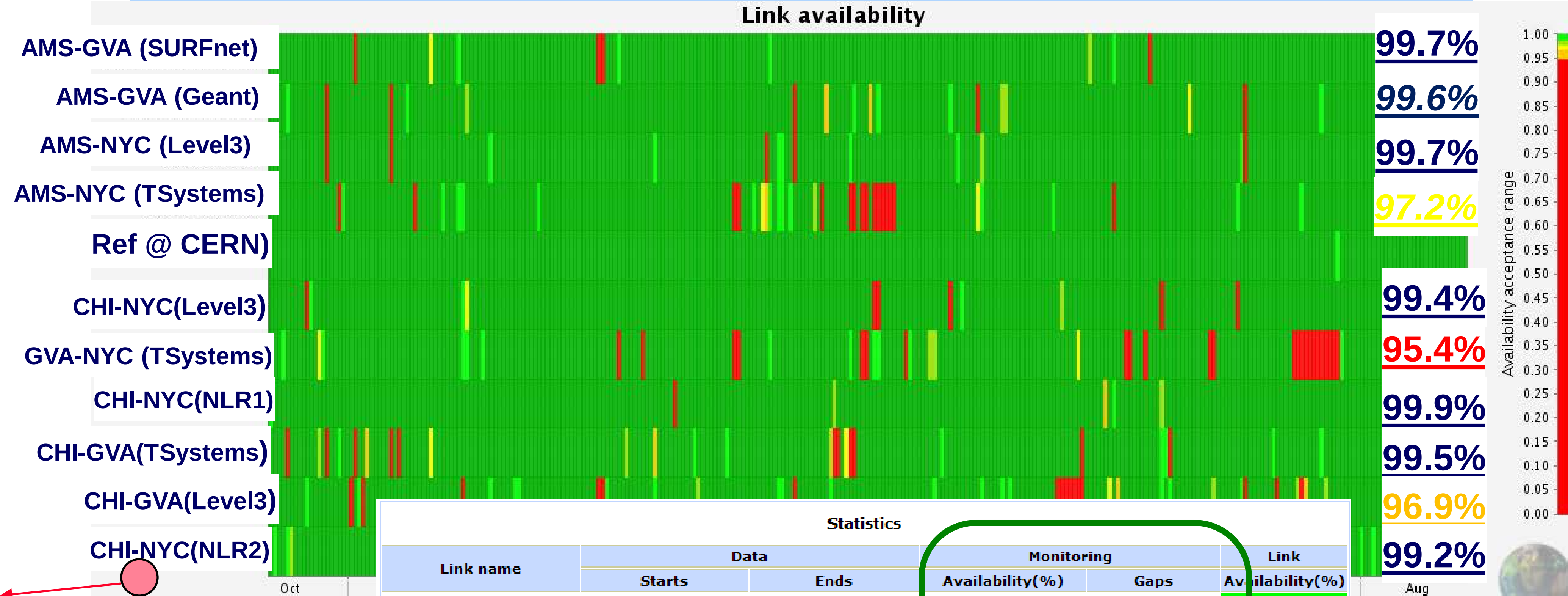
Real Time Topology for L2 Circuits



USLHCnet: Accounting for Integrated Traffic



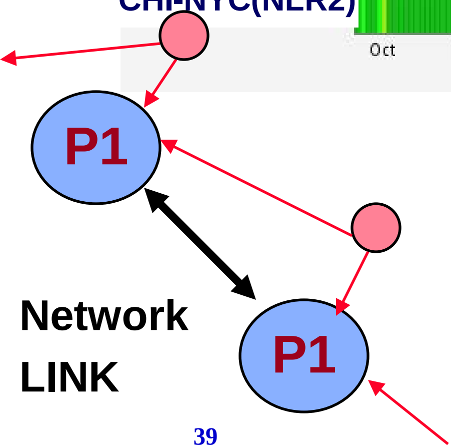
Monitoring Links Availability
Very Reliable Information



Statistics

Link name	Data		Monitoring		Link
	Starts	Ends	Availability(%)	Gaps	Availability(%)
AMS-GVA Ciena (SURFnet)	10 Oct 2009 15:30	29 Aug 2010 15:30	100%	-	99.70%
AMS-GVA F10 (Geant)	10 Oct 2009 15:30	29 Aug 2010 15:30	100%	-	99.67%
AMS-NY (Level3)	10 Oct 2009 15:30	29 Aug 2010 15:			99.72%
AMS-NY (TSystems)	10 Oct 2009 15:30	29 Aug 2010 15:			97.28%
GVA1-GVA2 (USLHCNet)	10 Oct 2009 15:30	29 Aug 2010 15:			99.100%
GVA-NY (Level3)	10 Oct 2009 15:31	29 Aug 2010 15:			99.45%
GVA-NY (TSystems)	10 Oct 2009 15:31	29 Aug 2010 15:			95.24%
CHI-NY (NLR 1)	10 Oct 2009 15:30	29 Aug 2010 15:30	100%	-	99.93%
CHI-GVA (TSystems)	10 Oct 2009 15:30	29 Aug 2010 15:30	100%	-	99.50%
CHI-GVA (Level3)	10 Oct 2009 15:30	29 Aug 2010 15:30	100%	-	96.99%
CHI-NY (NLR 2)	10 Oct 2009 15:31	29 Aug 2010 15:30	100%	-	99.27%

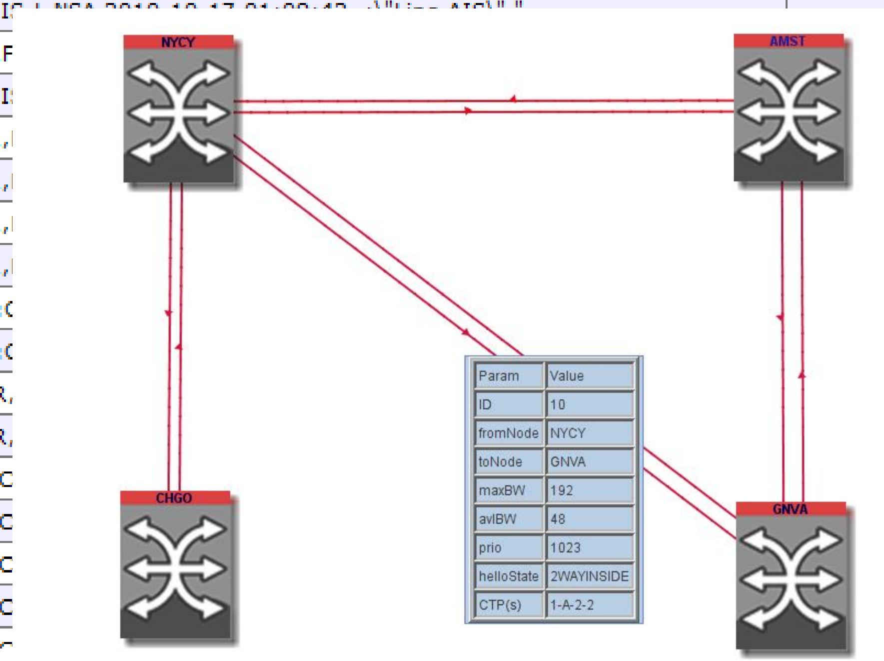
100% monitoring availability



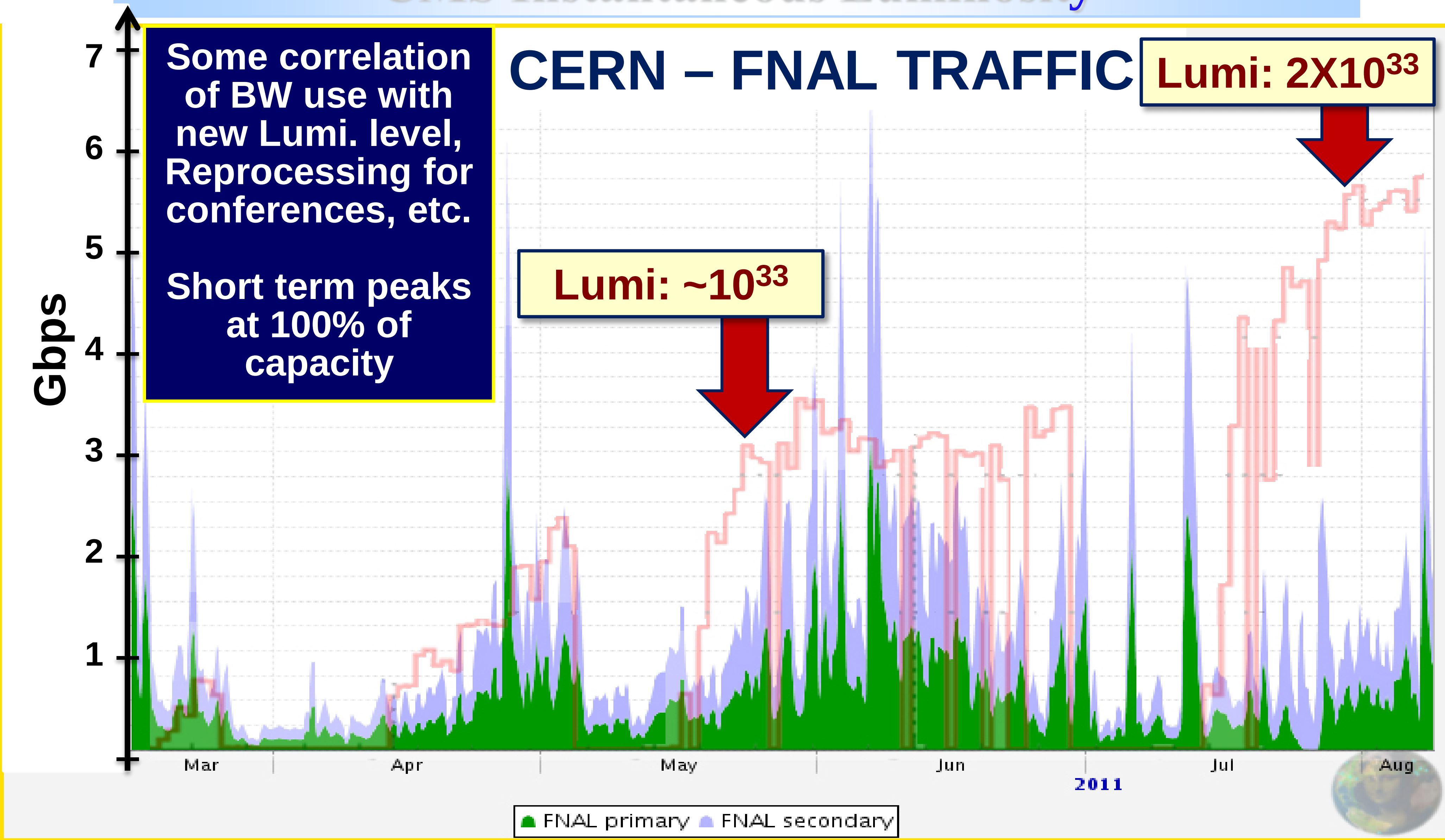
ALARMS and Automatic notifications for USLHCnet

CIENA Alarms for USLHCNet

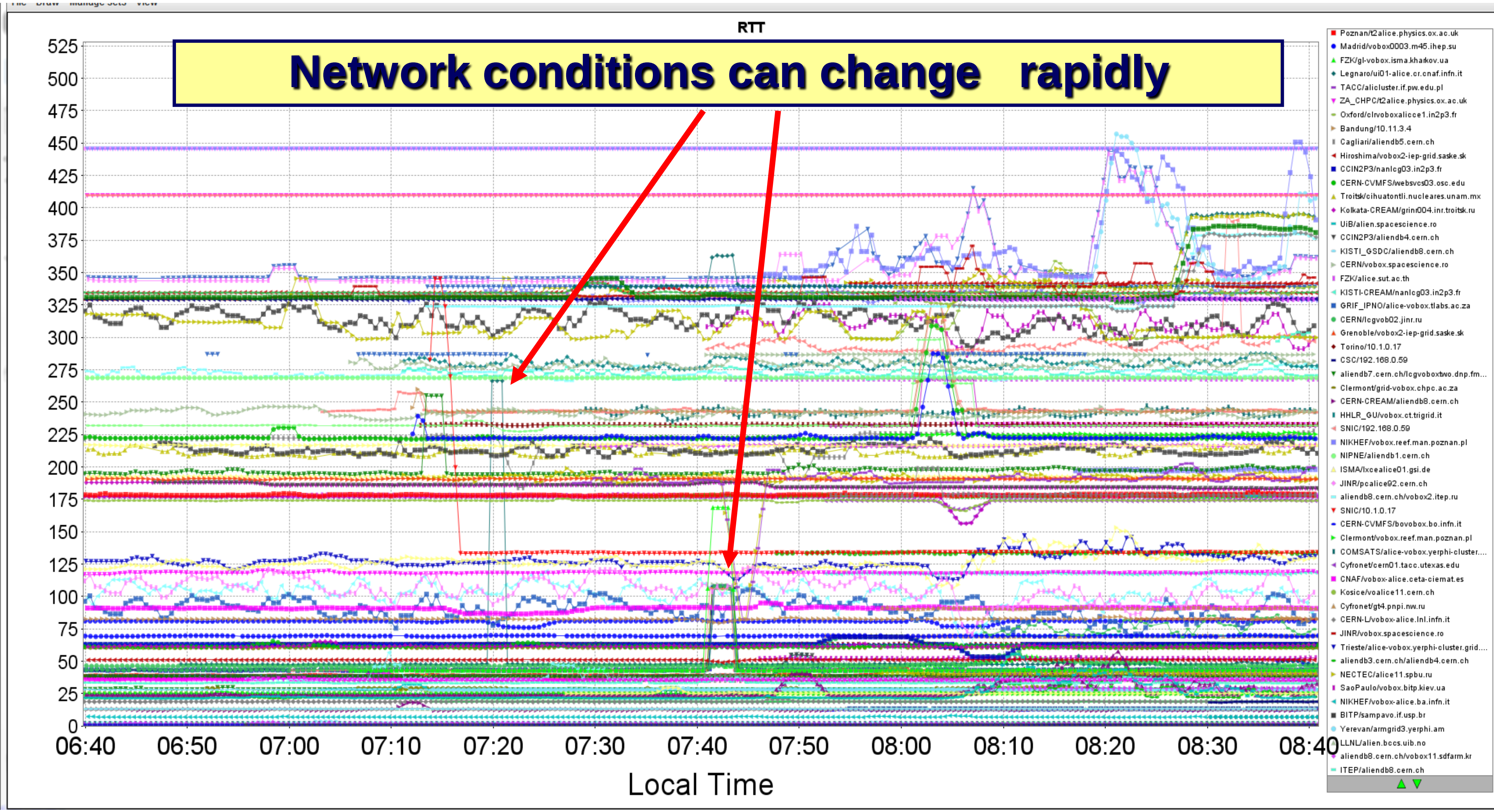
Date (GMT)	Site	Node IP	Alarm	Remarks
last week				Filter
18.10.2010 12:09	AMS_USLHCNET_CDS	192.65.197.40	"1-A-3-1-1,GIGE:CR,LOS,SA,2010-10-18,10:09:00,,:\"Loss of signal\","	
18.10.2010 12:06	AMS_USLHCNET_CDS	192.65.197.40	"1-A-3-1-1,GIGE:CR,LOS,SA,2010-10-18,10:05:33,,:\"Loss of signal\","	
17.10.2010 03:21	CHI_USLHCNET_CDS	192.65.196.107	"TimingInput_LINE_1,REF:MN,SYNCCLK,NSA,2010-10-17,01:20:56,,:\"Frequency offset ...	
17.10.2010 03:20	CHI_USLHCNET_CDS	192.65.196.107	"TimingInput_LINE_1,REF:MN,SYNCCLK,NSA,2010-10-17,01:20:56,,:\"Frequency offset ...	
17.10.2010 03:19	AMS_USLHCNET_CDS	192.65.197.40	"1-A-2-2,OC192:MN,RFI-L,NSA,2010-10-17,01:19:16,,:\"Line RFI\","	
17.10.2010 03:10	AMS_USLHCNET_CDS	192.65.197.40	"1-A-2-2,OC192:MN,AI	
17.10.2010 03:09	AMS_USLHCNET_CDS	192.65.197.40	"1-A-2-2,OC192:MN,RF	
17.10.2010 03:09	GVA_USLHCNET_CDS	192.65.196.172	"1-A-8-1,OC192:MN,AI	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-chi-S1-2,SNC:CR,	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-chi-S1-3,SNC:CR,	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-chi-S1-6,SNC:CR,	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-chi-S1-7,SNC:CR,	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-nyc-3513-9,SNC:	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-nyc-3524-6,SNC:	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-nyc-S1-1,SNC:CR,	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-nyc-S1-4,SNC:CR,	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-chi-3500-4,SNC:C	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-chi-3500-6,SNC:C	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-chi-3506-5,SNC:C	
17.10.2010 03:06	GVA_USLHCNET_CDS	192.65.196.172	"gva-chi-3506-6,SNC:C	



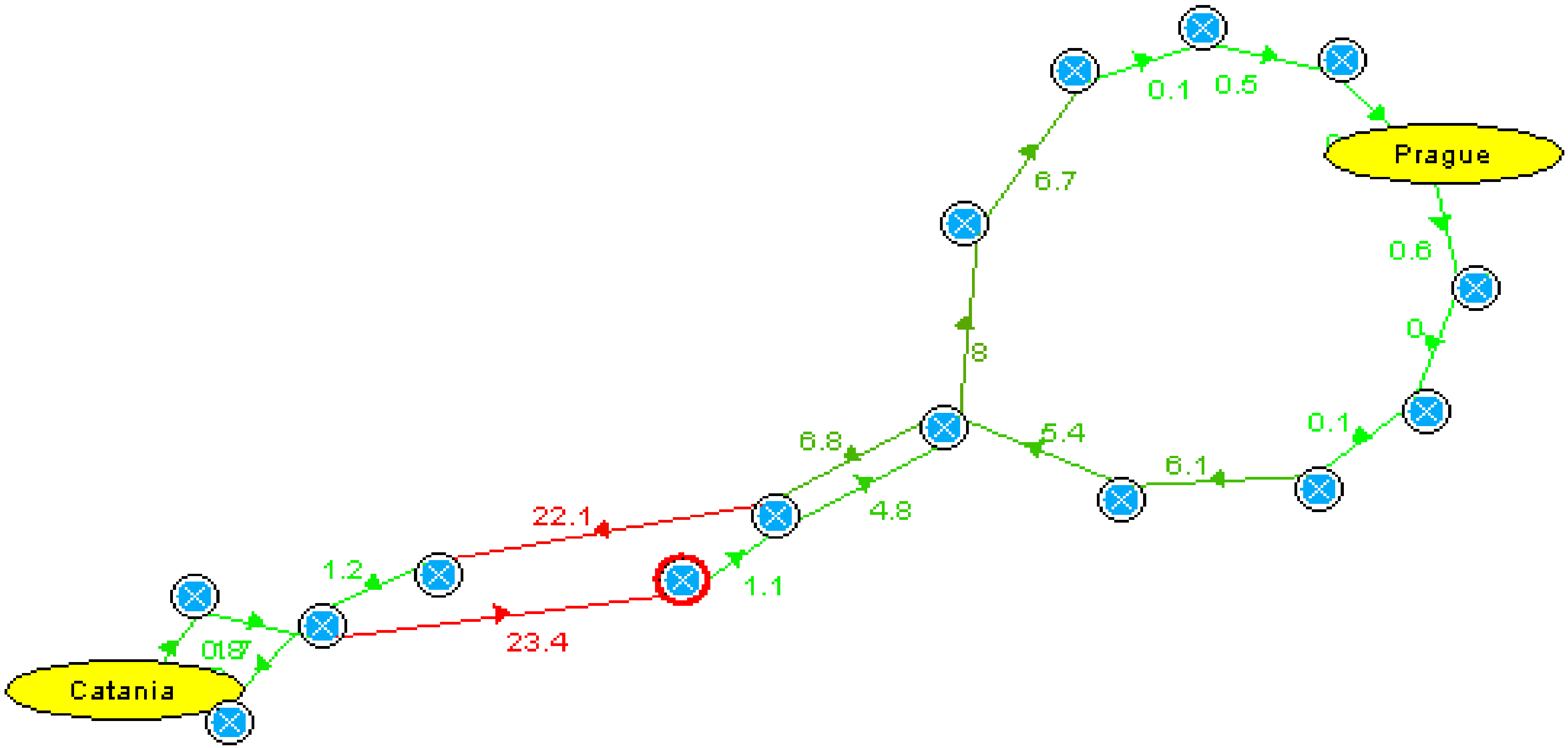
CERN-FNAL Traffic and CMS Instantaneous Luminosity



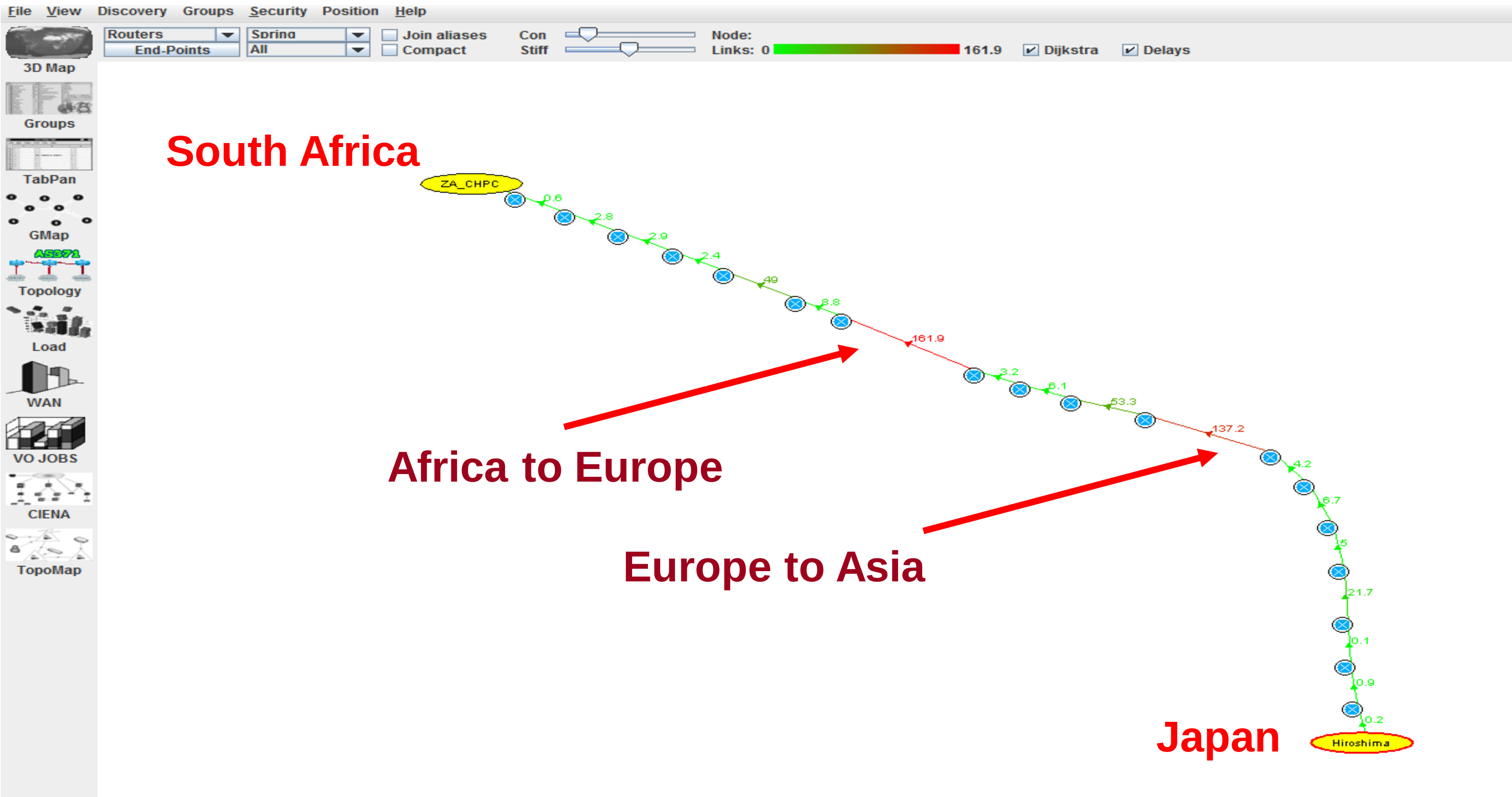
ALICE: RTT measurements from CERN to all sites



Asymmetric Routing



Path monitoring for each pair of sites



Data Transfer in WAN

MIT Open Courseware
Data Networks: Lecture Notes

LEC #	TOPICS	FILES
1	Data Networks	(PDF)
2	The Data Link Layer: Framing and Error Detection	(PDF)
3 & 4	The Data Link Layer: ARQ Protocols	(PDF)
5 & 6	Introduction to Queueing Theory	(PDF)
7	Burke's Theorem and Networks of Queues	(PDF)
8 & 9	M/G/1 Queues	(PDF)
10 & 11	Reservations Systems M/G/1 Queues with Priority	(PDF)
13 & 14	Packet Multiple Access: The Aloha Protocol	(PDF)
15 & 16	Local Area Networks	(PDF)
17 & 18	Fast Packet Switching	(PDF)
19	Broadcast Routing	(PDF)
20	Routing in Data Networks	(PDF)
21	Optimal Routing	(PDF)
22 & 23	Flow and Congestion Control	(PDF)
24 & 25	Higher Layer Protocols: TCP/IP and ATM	(PDF)

ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-263j-data-communication-networks-fall-2002/

Dynamic adjustment of window size

- TCP starts with CW = 1 packet and increases the window size slowly as ACK's are received
  - Slow start phase
  - Congestion avoidance phase
- Slow start phase
  - During slow start TCP increases the window by one packet for every ACK that is received
  - When CW = Threshold TCP goes to Congestion avoidance phase
  - Notice: during slow start CW doubles every round trip time

    Exponential increase!
- Congestion avoidance phase
  - During congestion avoidance TCP increases the window by one packet for every window of ACKs that it receives
  - Notice that during congestion avoidance CW increases by 1 every round trip time - Linear increase!
- TCP continues to increase CW until congestion occurs

How to efficiently move data in WAN
TCP Performance

What influences the TCP performance?

- Available bandwidth
- Packet Loss
- Out of order delivery
- Round-trip
- Congestion avoidance algorithm
- TCP setup and tuning
- Buffers in Switches and routers

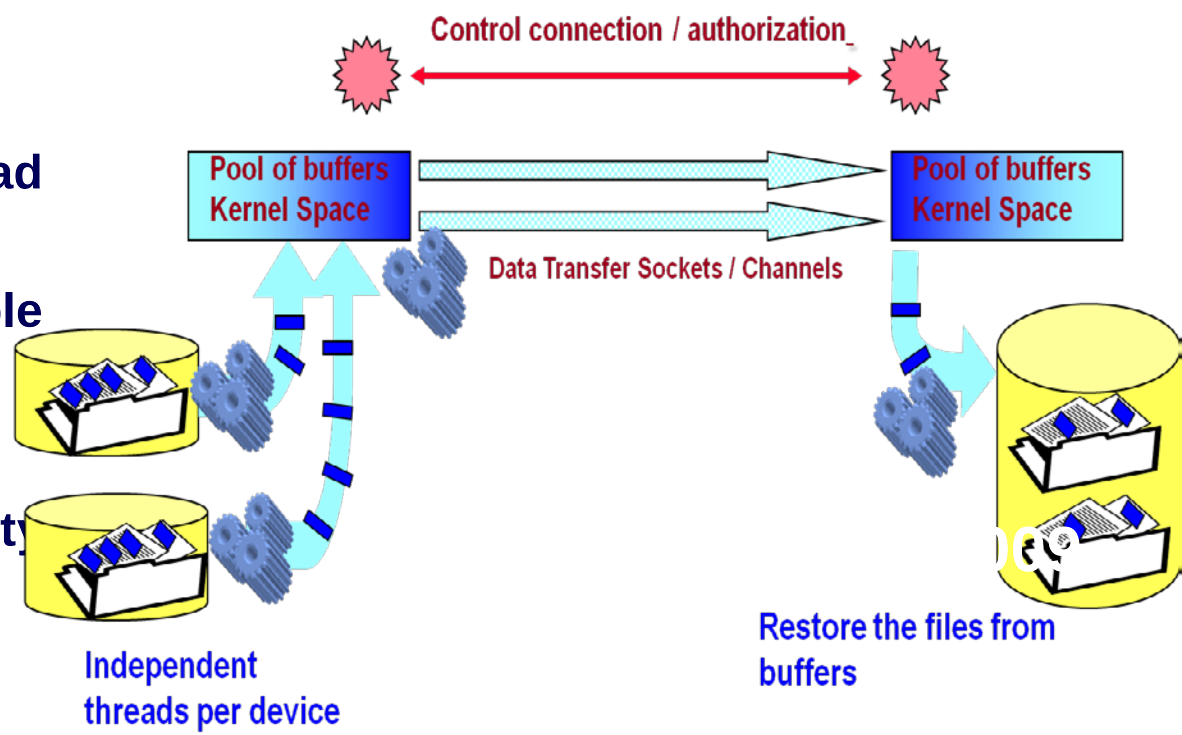
$$BW \sim \frac{\text{Segment Size}}{\text{RTT} * \text{SQRT (Loss Prob)}}$$

Long distance connections ➔ Parallelism

FDT – Fast Data Transfer

- **FDT is an application for efficient data transfers using parallel streams**
- **Easy to use. Written in java and runs on all major platforms.**
- **It is based on an asynchronous, multithreaded system which is using the NIO library and is able to:**

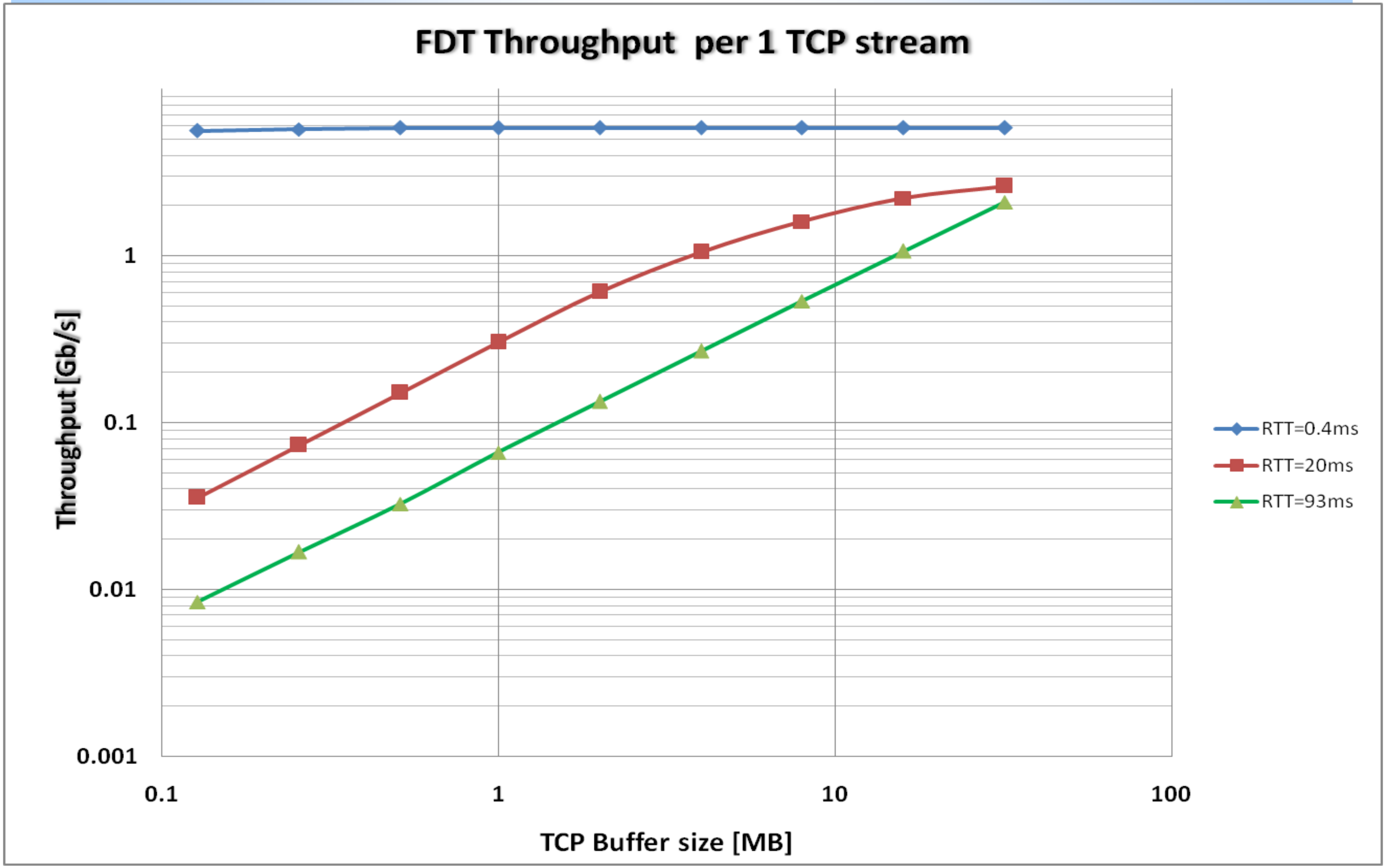
- **stream continuously a list of files**
- **use independent threads to read /write on each physical device**
- **transfer data in parallel on multiple TCP streams, when necessary**
- **resume a file transfer session**
- **allows to "plug-in" external security APIs (SSL, GSI–SSH)**
- **Controlled by the MonALISA**



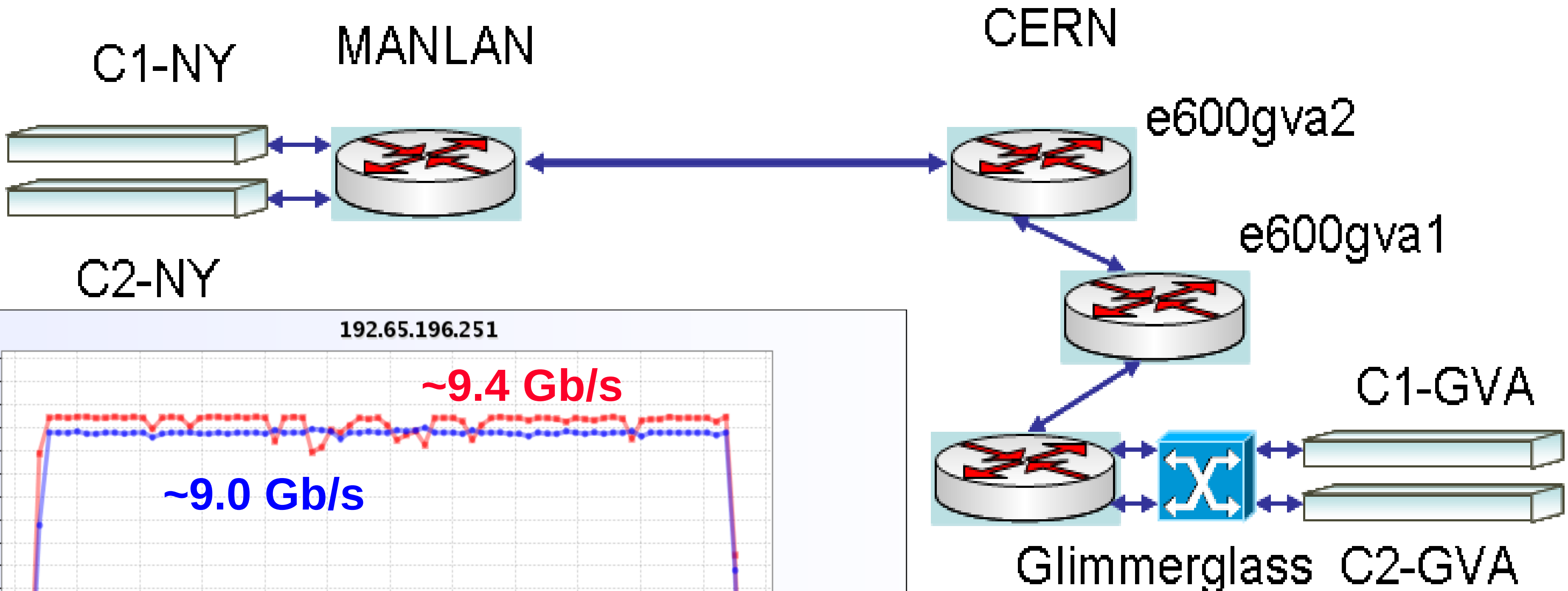
FDT features

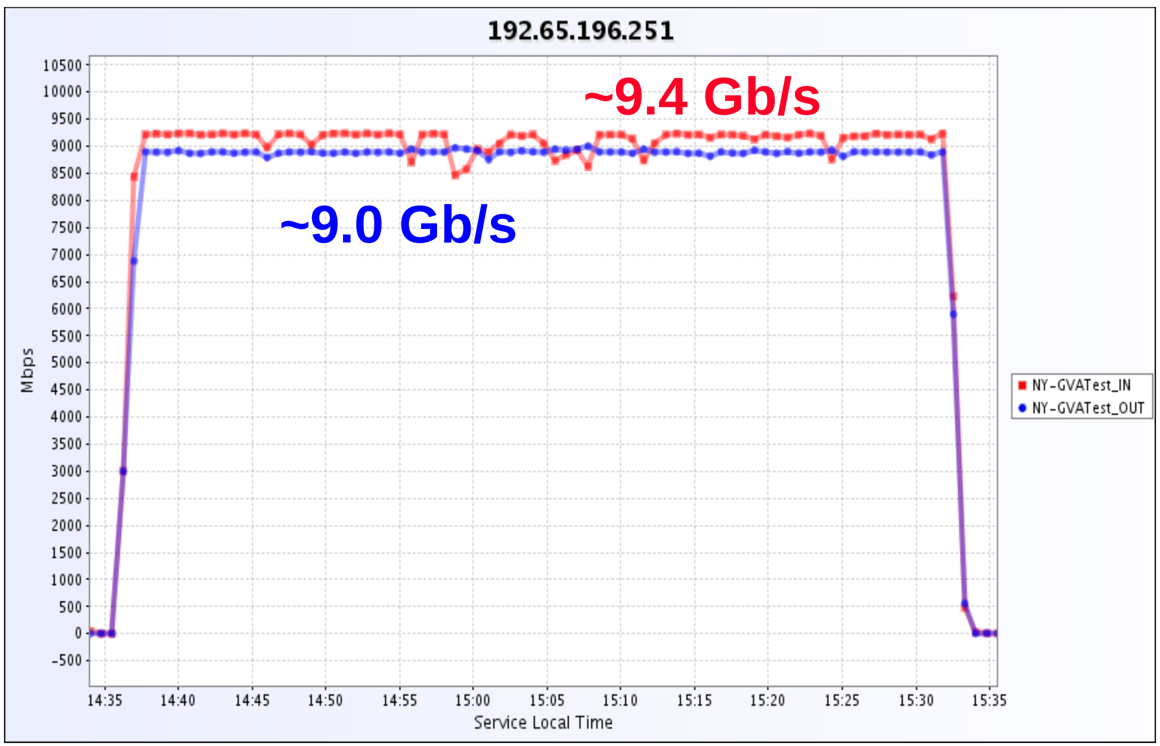
- The FDT architecture allows to "plug-in" external security APIs and to use them for client authentication and authorization. Supports several security schemes :
  - IP filtering
  - SSH
  - GSI-SSH
  - Globus-GSI
  - SSL
- User defined loadable modules for Pre and Post Processing to provide support for dedicated MS system, compression …
- FDT can be monitored and controlled dynamically by MonALISA System

FDT Throughput TESTS



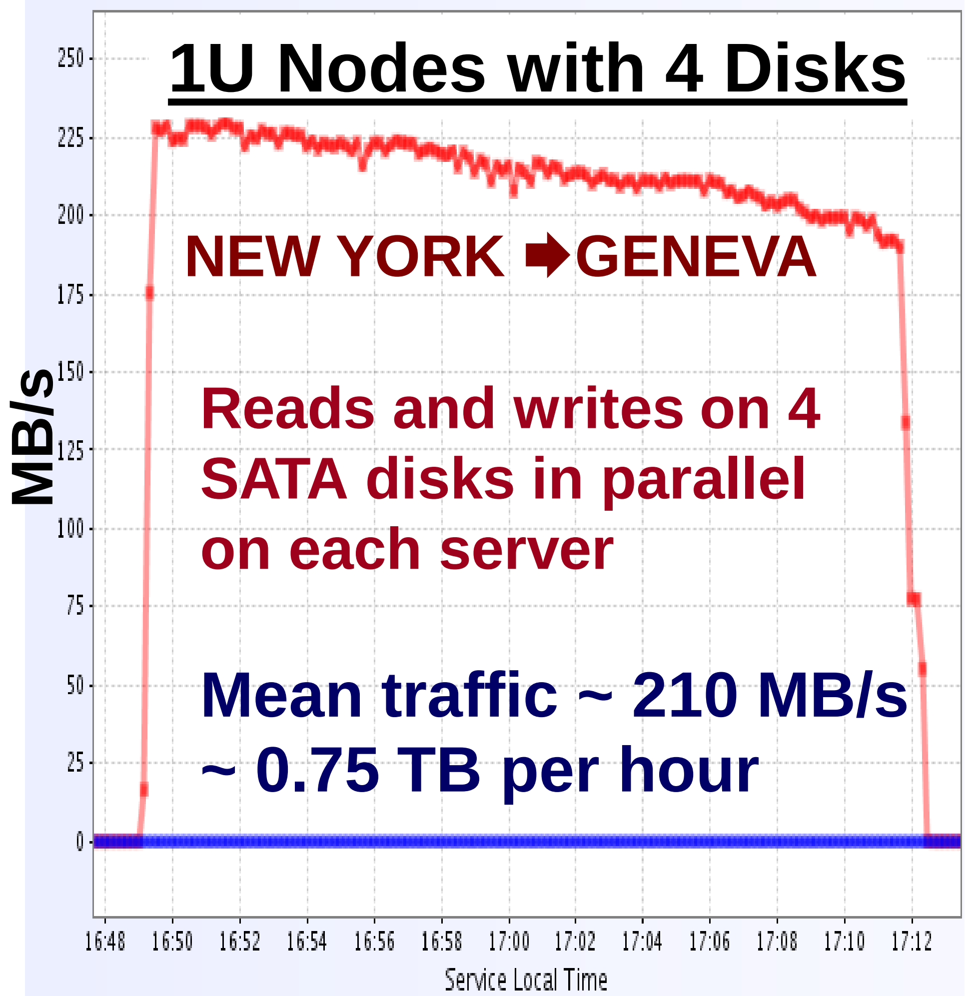
FDT – Memory to Memory Tests in WAN

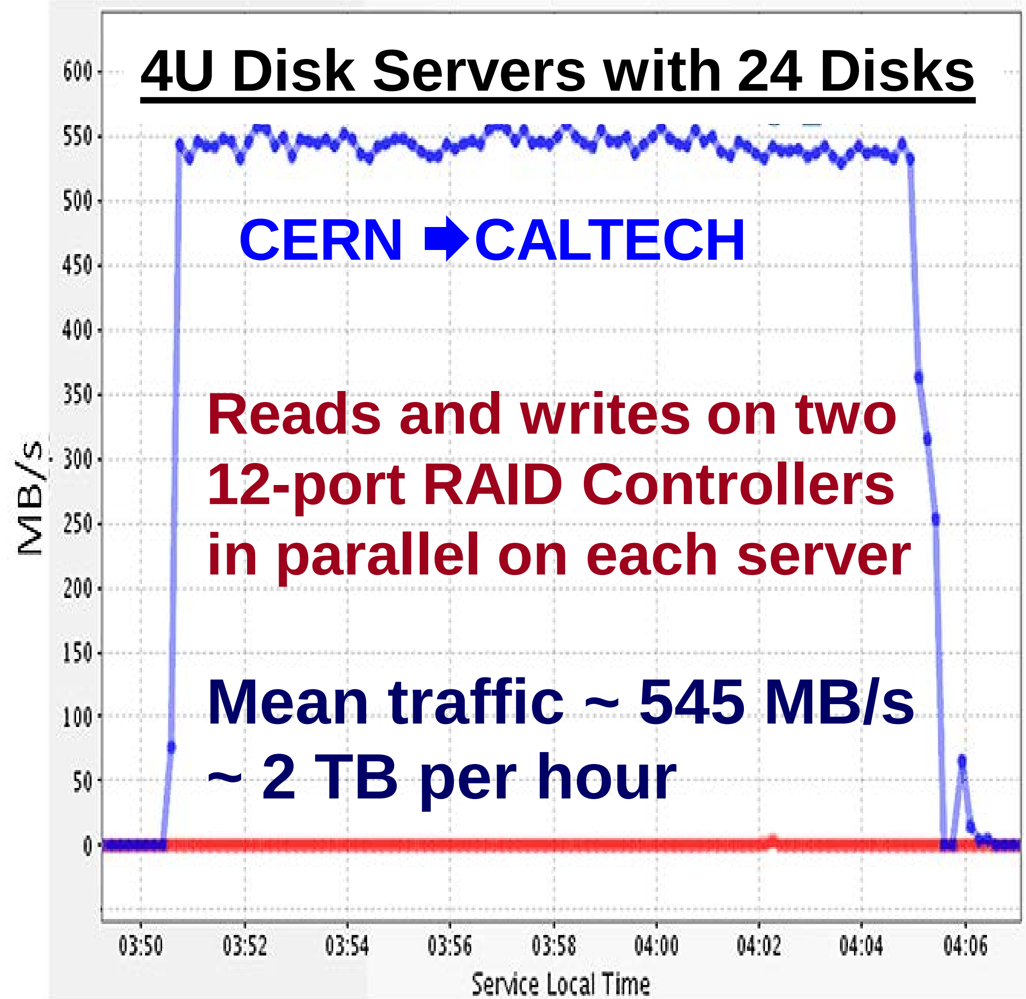




CPUs Dual Core Intel Xenon @ 3.00 GHz, 4 GB RAM, 4 x 320 GB SATA Disks Connected with 10Gb/s Myricom

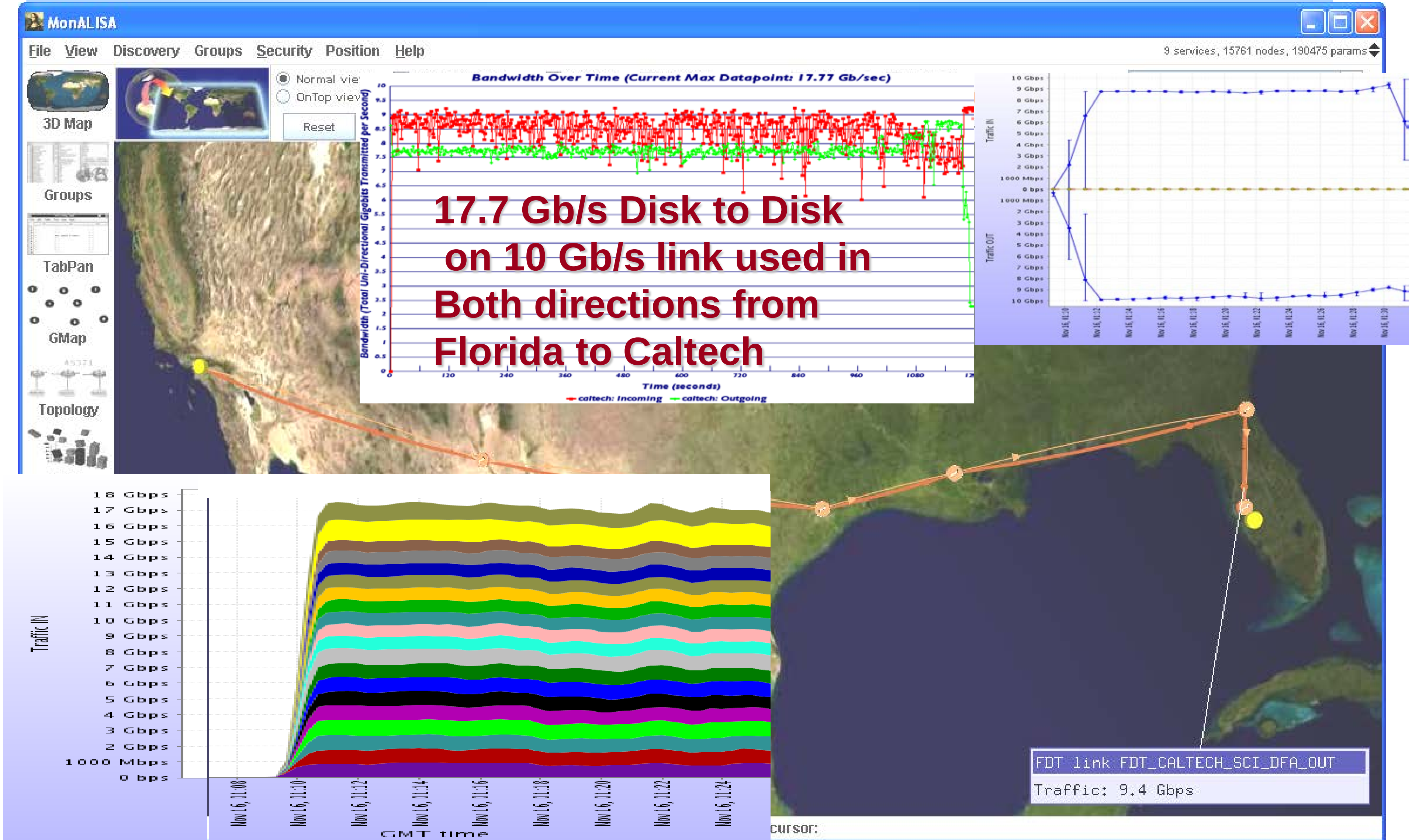
Disk -to- Disk transfers in WAN





- Lustre read/ write ~ 320 MB/s between Florida and Caltech
- Works with xrootd
- Interface to dCache using the dcap protocol

FDT & MonLISA Used at SC 2006



SC2008 – Bandwidth Challenge, Austin, TX

CIENA – Caltech Booths

FDT TRANSFERS

Ciena, OTU-4 standard link carrying a 100 Gbps payload (or 200 Gbps bidirectional) with forward error correction.

10x10Gbps multiplex

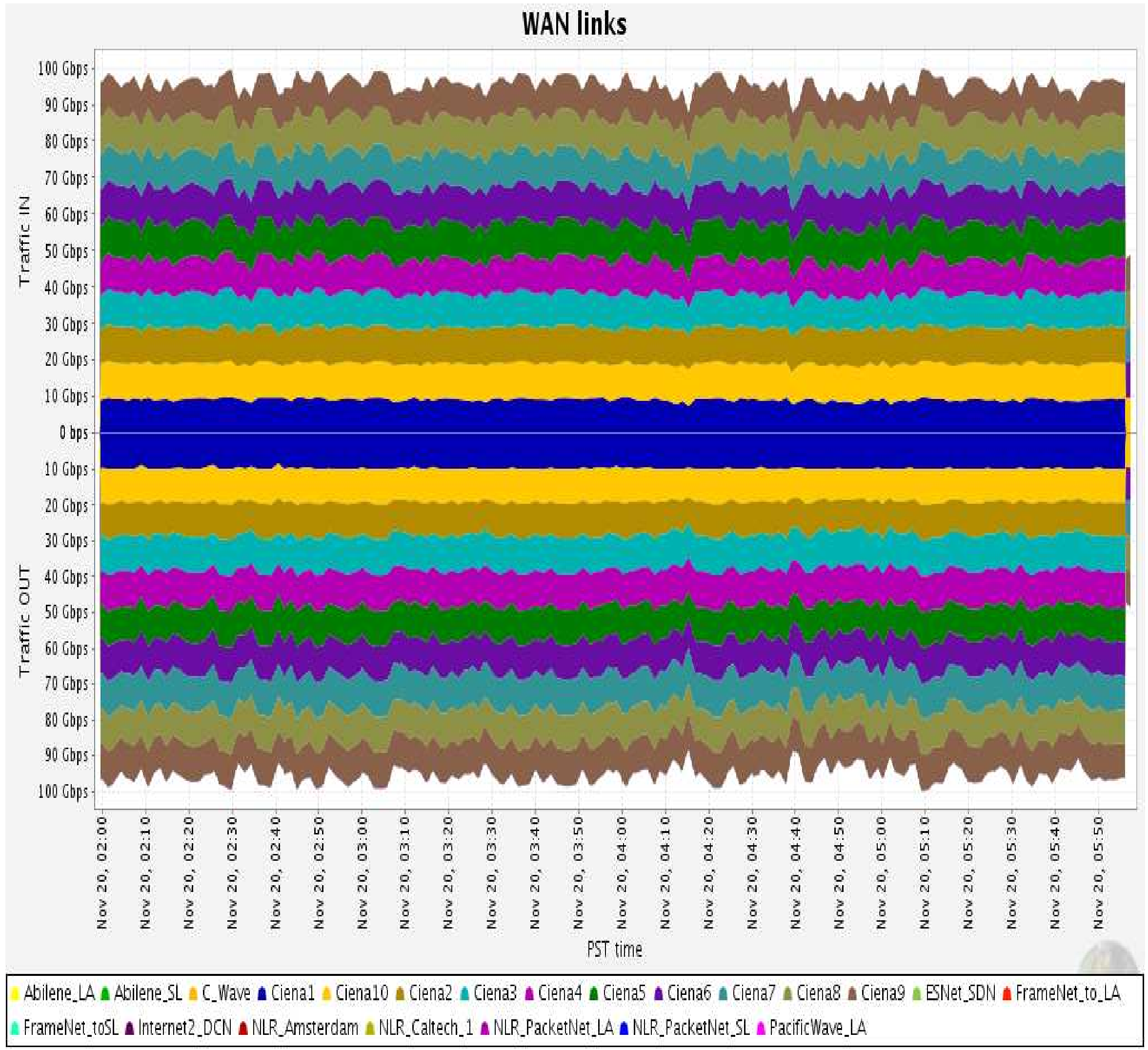
over an 80km fibre spool

Pick : 199.9 Gb/s

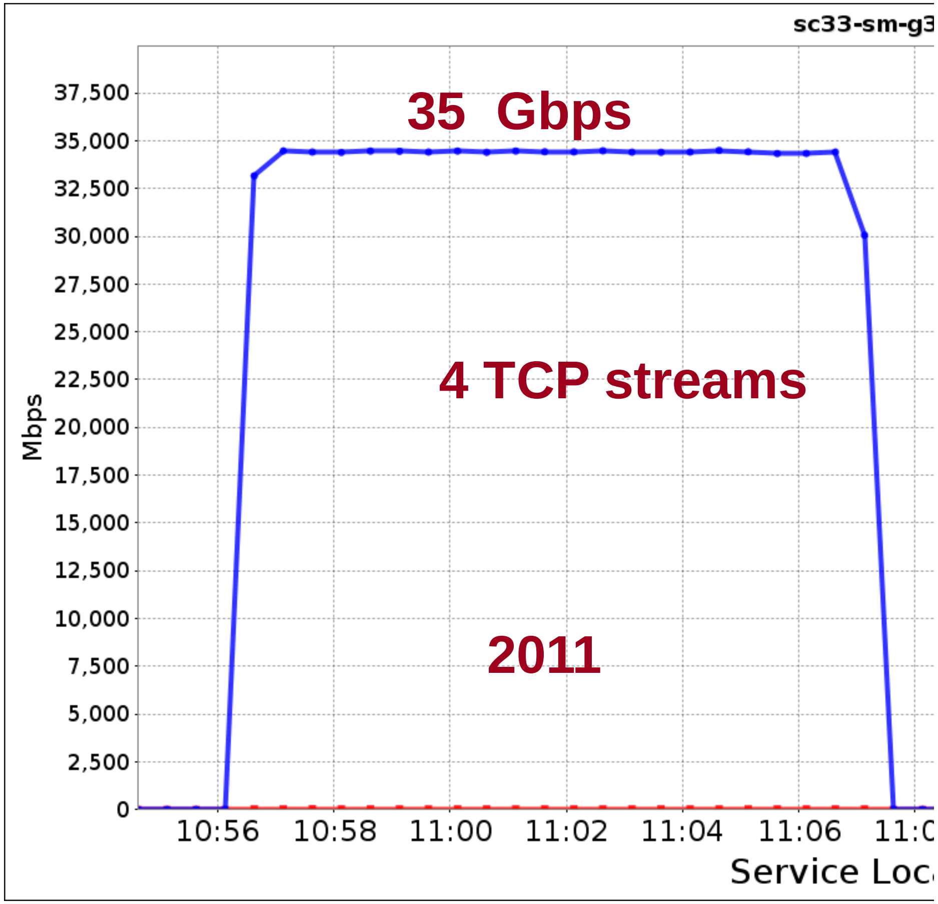
Mean 191 Gb/s

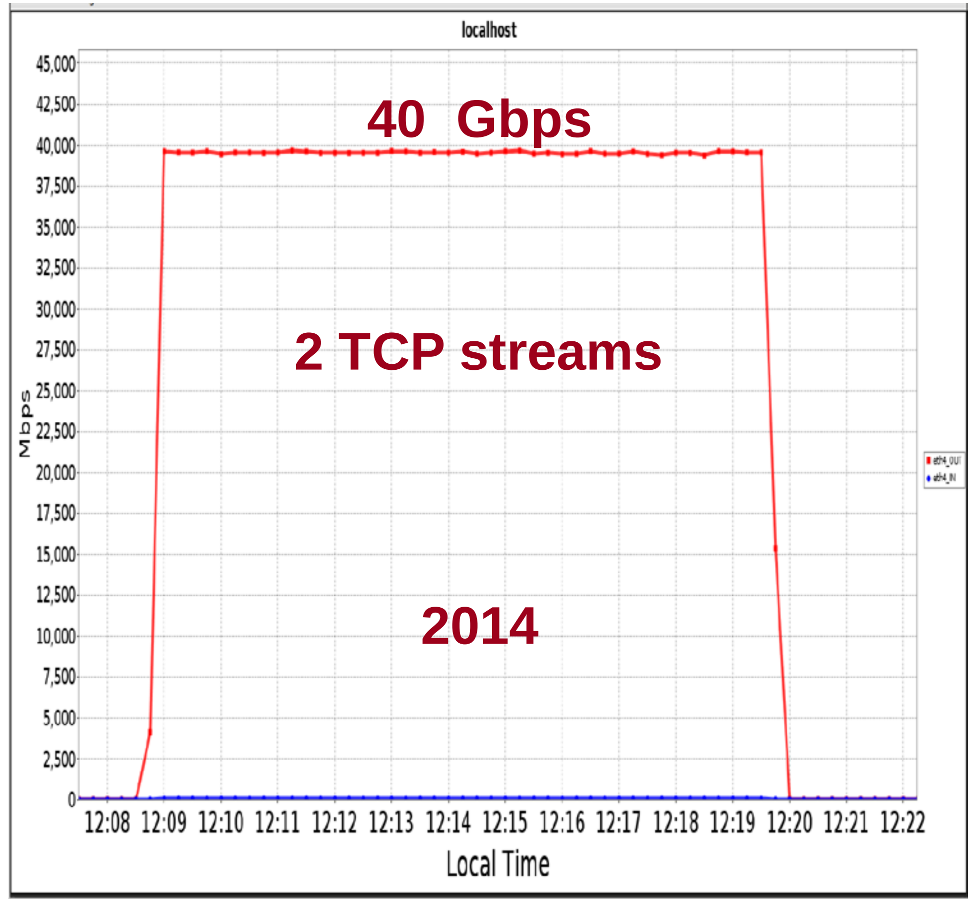
We transferred :

~ 1PB in ~12 hours



FDT on 40Gbps NICs (LAN)

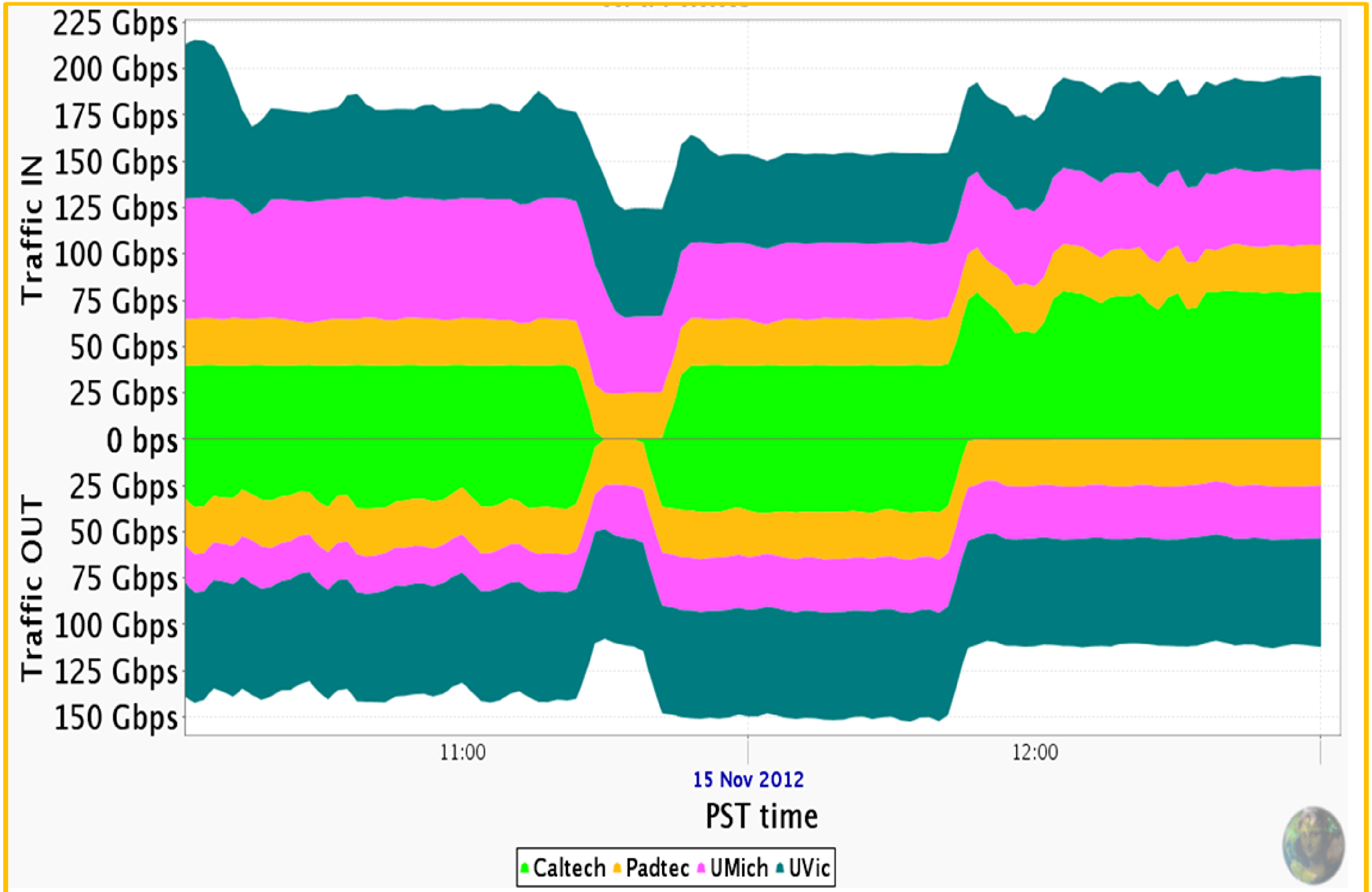




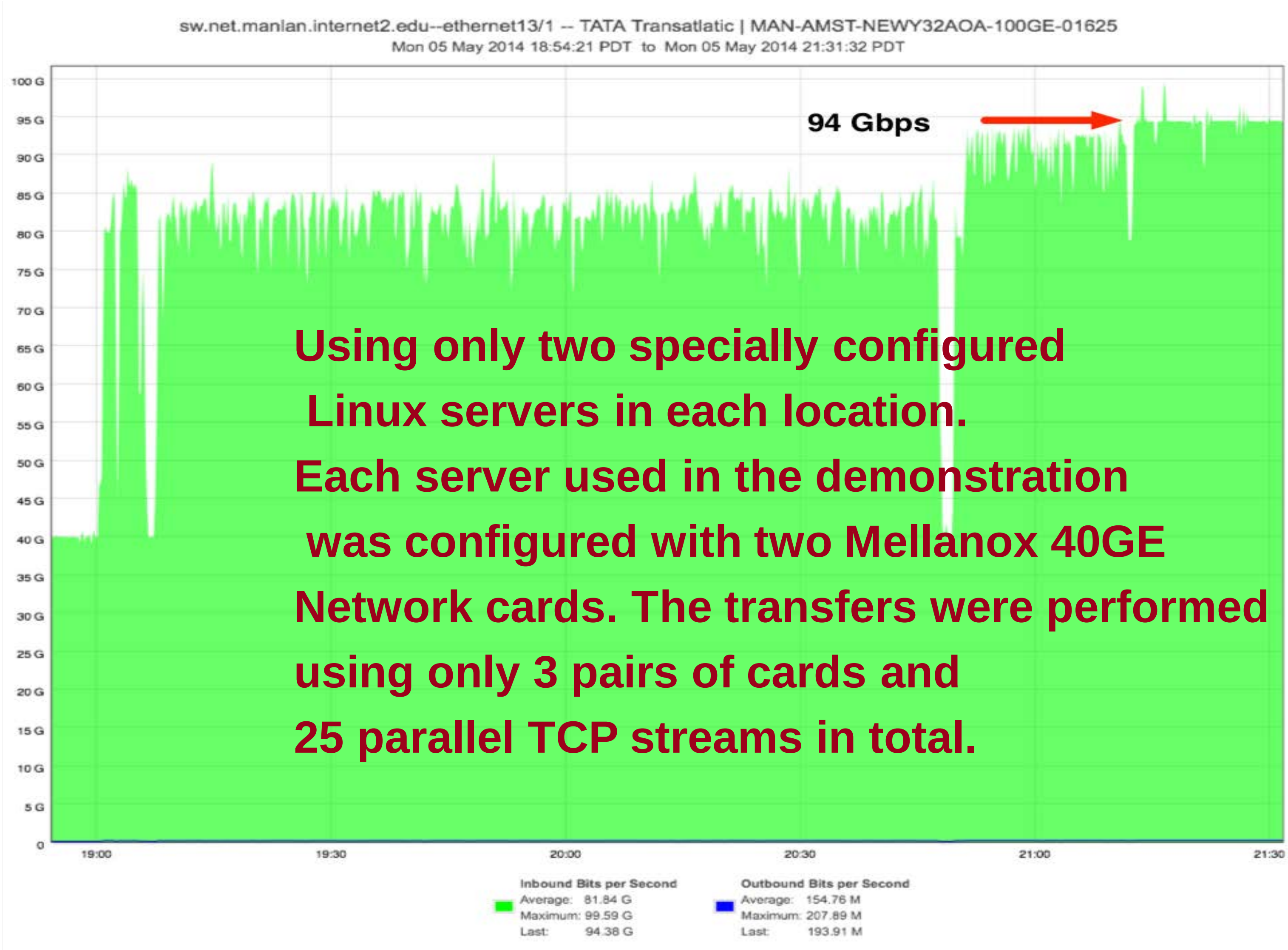
Exactly the same hardware … but better drivers and kernel

Improved CPU usage

FDT used SC2012 on 40Gbps NIC cards



FDT used on network tests between CERN to Ottawa (May 2014)



Using only two specially configured Linux servers in each location.

Each server used in the demonstration was configured with two Mellanox 40GE Network cards. The transfers were performed using only 3 pairs of cards and 25 parallel TCP streams in total.

FDT Status

The multi-threaded approach in FDT together with the zero copy and the java-NIO was a right architecture that works fine with 1, 10 and 40 NIC cards.

FDT is internally instrumented (ApMon) and can report a lot of Monitoring information into the MonALISA system.

It can also be controlled by MonALISA agents.

It is not used for data transfers in HEP Grids , but it is used a lot in Amazon cloud (~ several PB /year)

Tuning for high performance data transfers

Default TCP send and receive buffer size were initially 64KBytes

It should be ~8 – 32 MB. Auto-tuning of the buffer size were introduced recently and works fine

MTU –Maximum Transfer Unit Jumbo frames MTU 9000

Firewalls

IRQ pinning (also know as IRQ affinity)

Congestion control : cubic

Network mapping for all ALICE Sites

- **Continuous WAN measurements for 85x85 site matrix**
  - **MonALISA with FTD**
- **Complex topology – automatic analysis of network conditions, coupled with SE tests**
- **Resulting in**
  - **Per site list of ‘best set’ of Storage elements**
  - **Given to the client for data reading/writing**

Tuning high performance data transfers

The High Speed NICs and TCP congestion control mechanism produces bursty traffic as seen by the network devices.

The queue length is in general described by the integral of the difference between the arrival process and the departure process:

QL = $\int_t A(t) - D(t)$

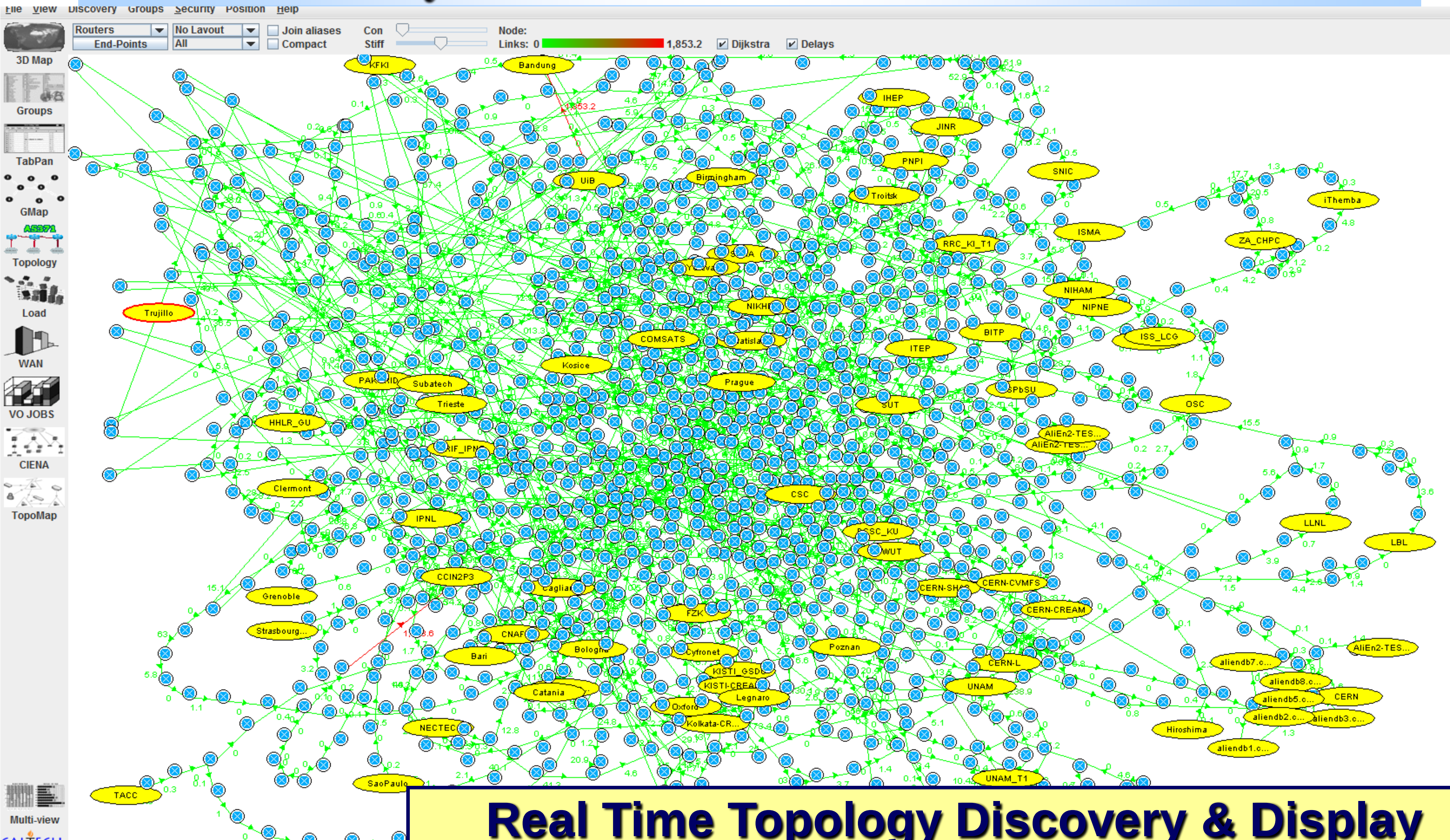
The arrival processes in in general a random process while the departure process can be a deterministic process with a random component.

The only case where the arrival rate is proportional with the queue length

$$\int_t A\,(t) \sim A(t)$$

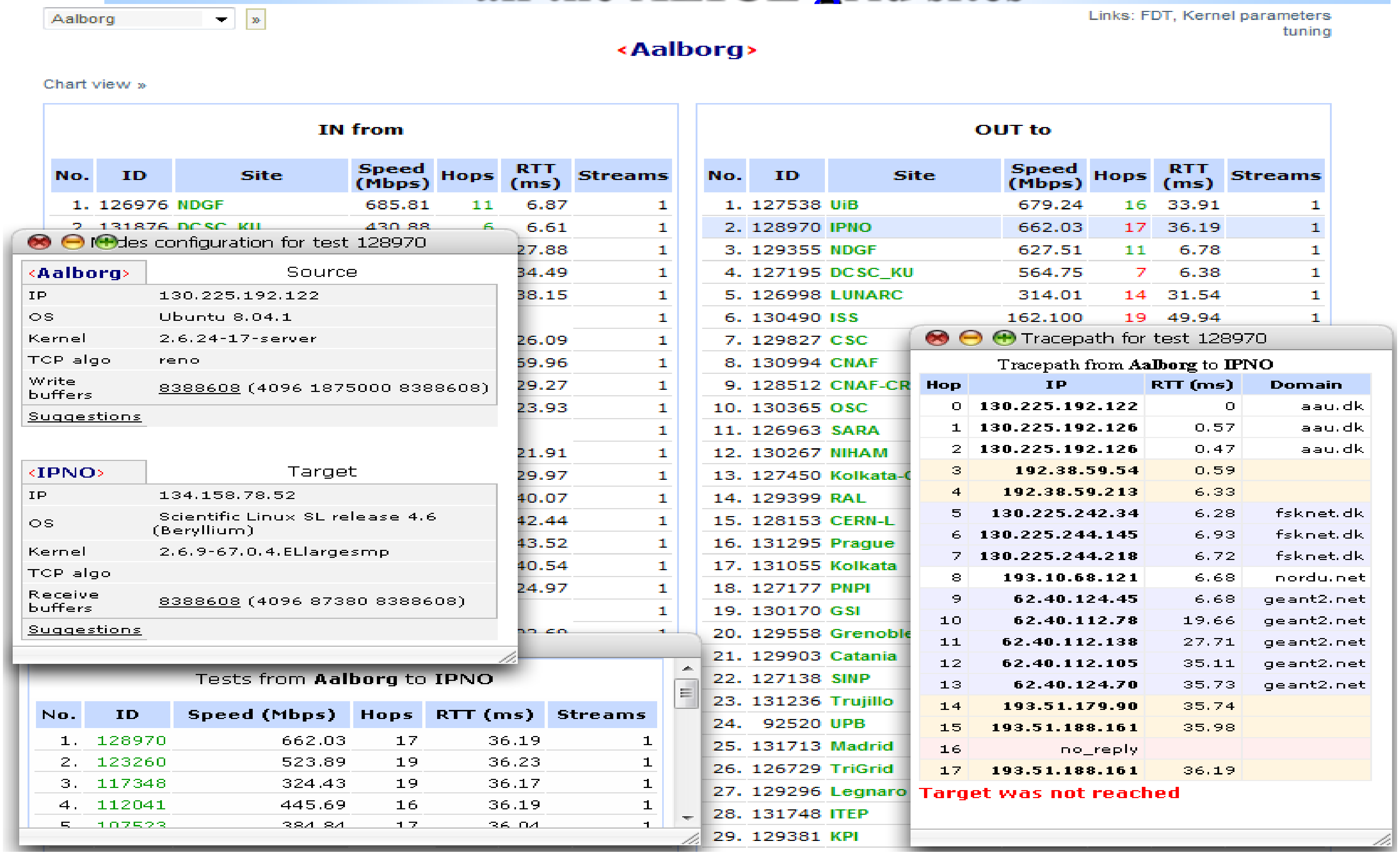
is for the Poisson process. High throughput IO is much better described by chain of bursty arrivals with strong correlations between data queues as well as between the sender and receiver process.

Monitoring Network Topology (L3), Latency and Routers for all ALICE sites

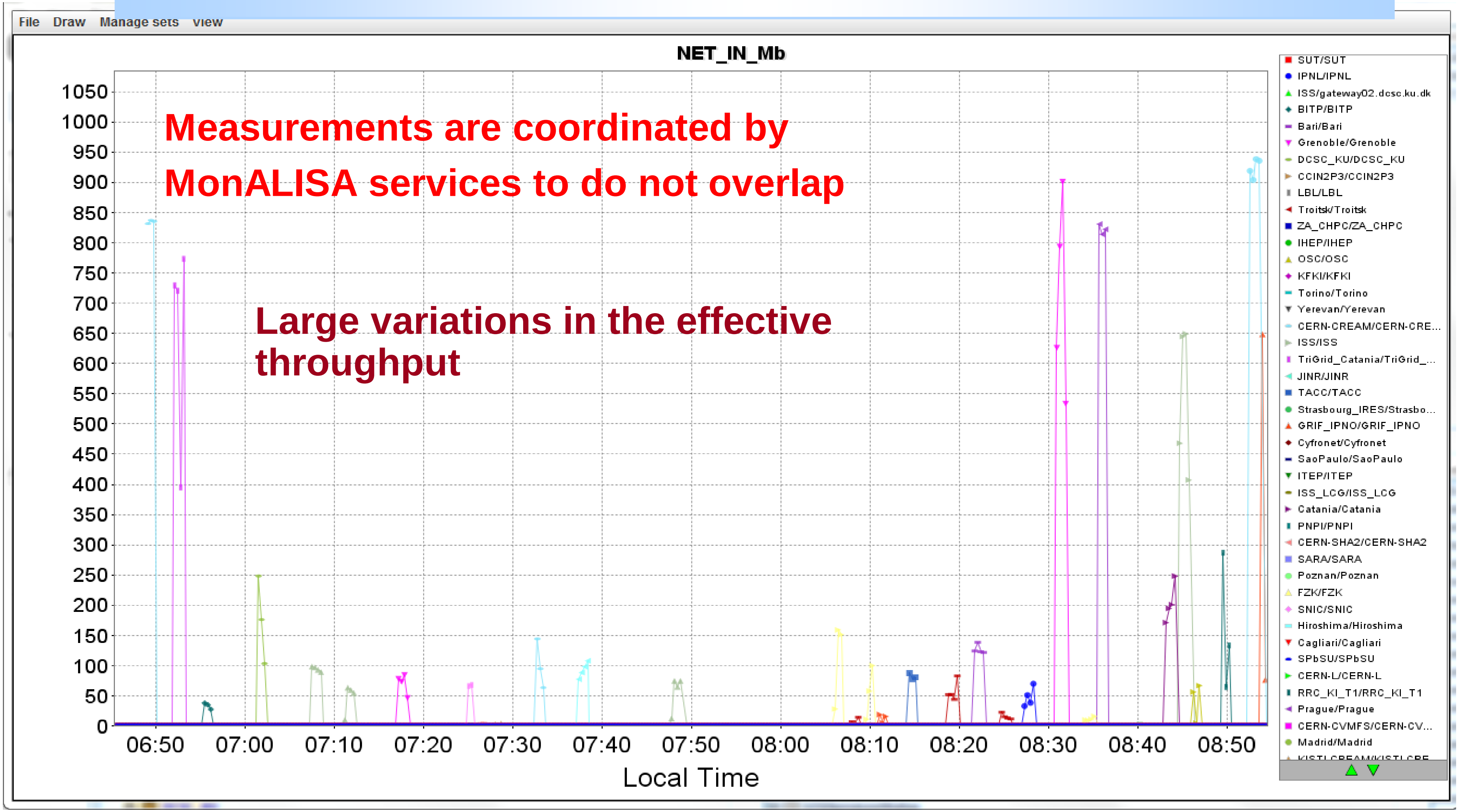


Real Time Topology Discovery & Display

Active Available Bandwidth measurements between all the ALICE grid sites



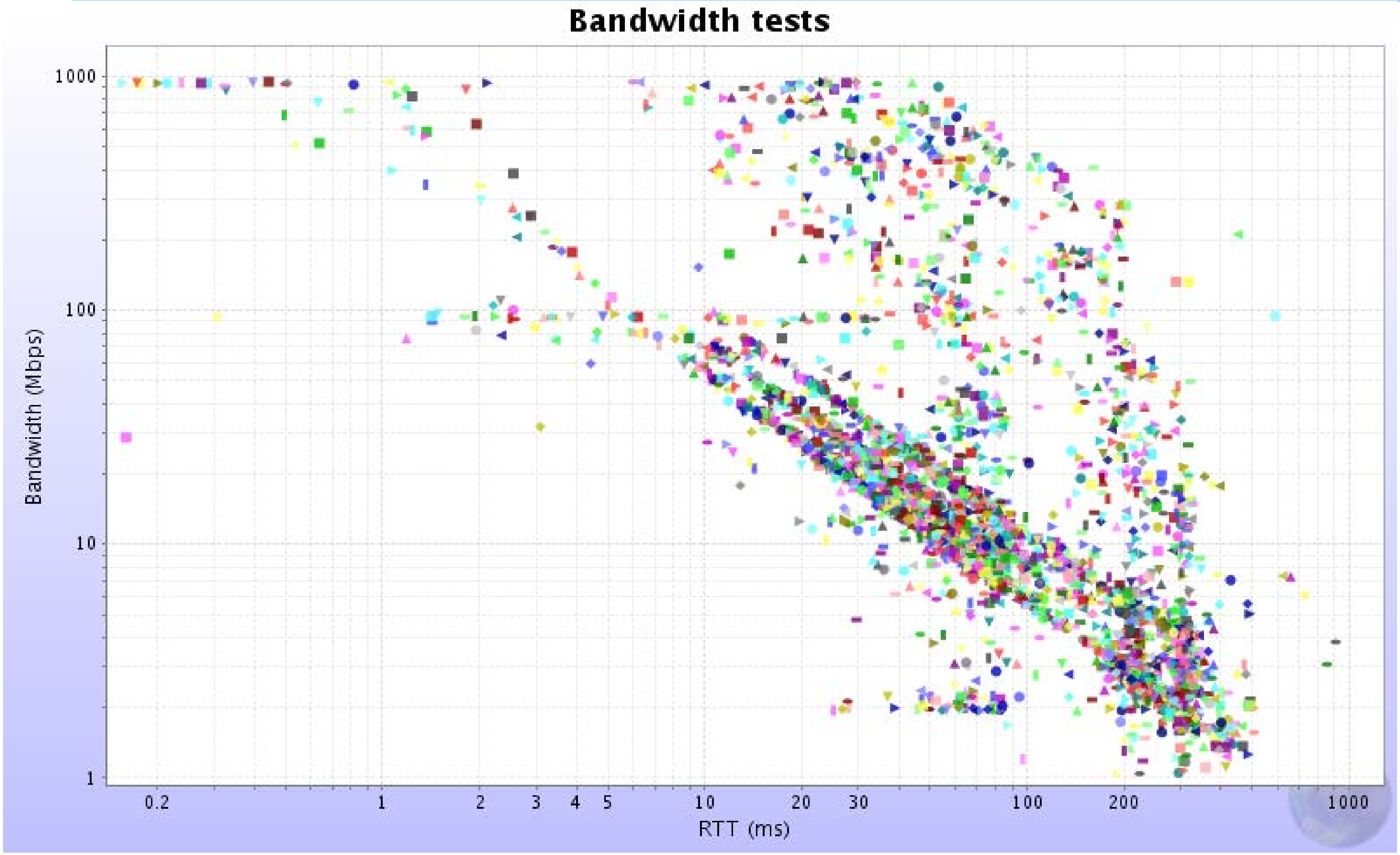
FDT Throughput measurements among sites



Measurements are coordinated by MonALISA services to do not overlap

Large variations in the effective throughput

Active Available Bandwidth measurements between all the ALICE grid sites



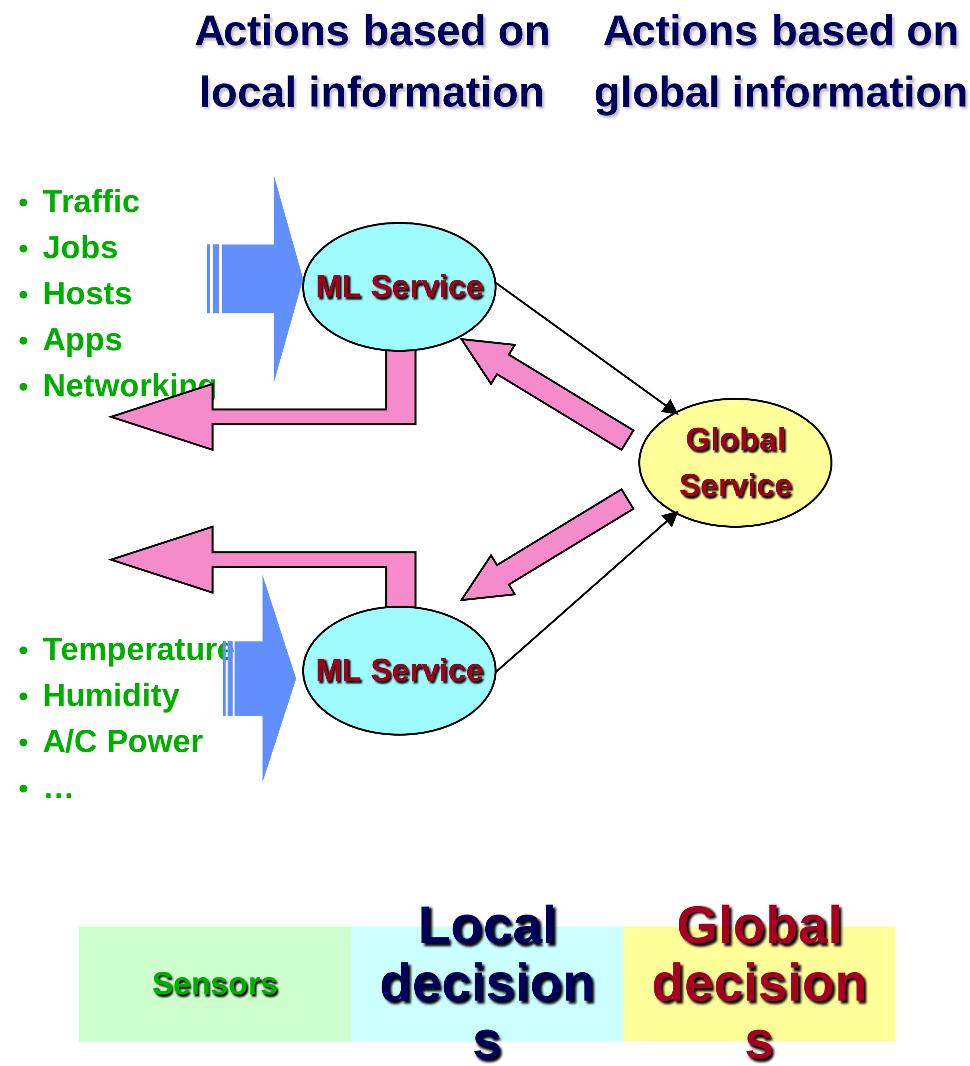
ALICE Network mapping

- The bandwidth tests, routing, kernel parameters are
  - Available to the site administrators for tuning of local network and host parameters
  - Negotiations with network providers
- However…. the situation is not ideal
  - Network tuning is a difficult task
  - Even well-intended operators sometimes have difficulty responding to inquiries (terminology barrier?)
  - New sites usually need ‘global’ help from network experts

Control in Distributed Systems

Operational Decisions and Actions

- Based on monitoring information, actions can be taken in
  - ML Service
  - ML Repository
- Actions can be triggered by
  - Values above/below given thresholds
  - Absence/presence of values
  - Correlation between multiple values
- Operational actions
  - Alerts
    - e-mail
    - Instant messaging
  - Supervision for Services
  - External commands
  - Event logging

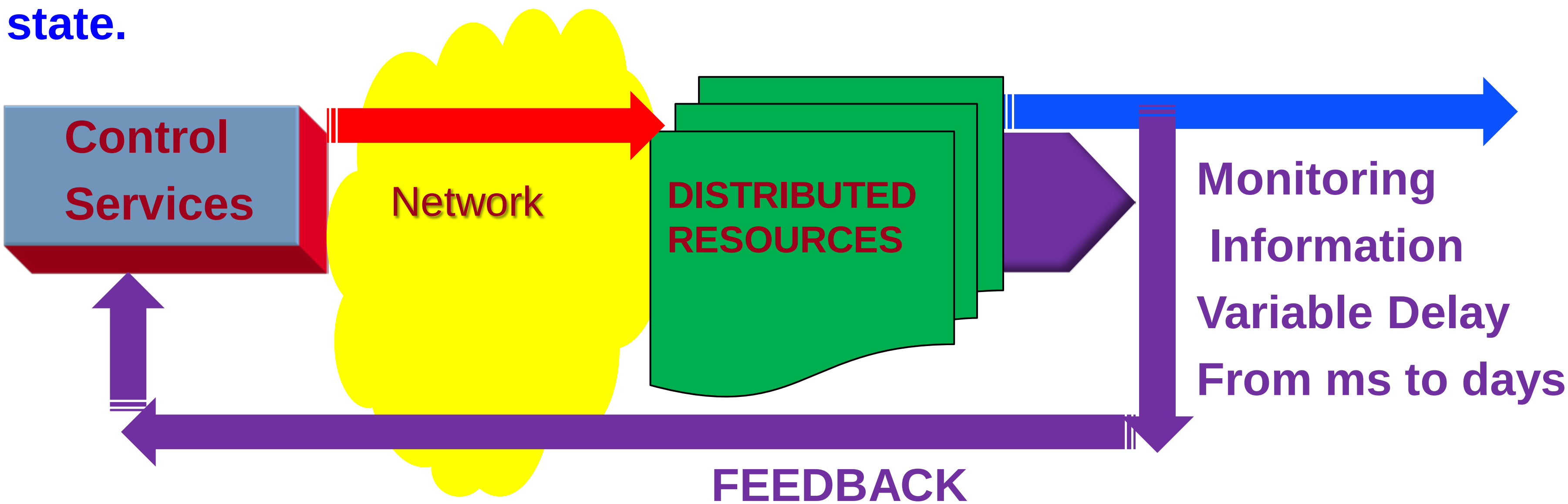


Control and Optimization

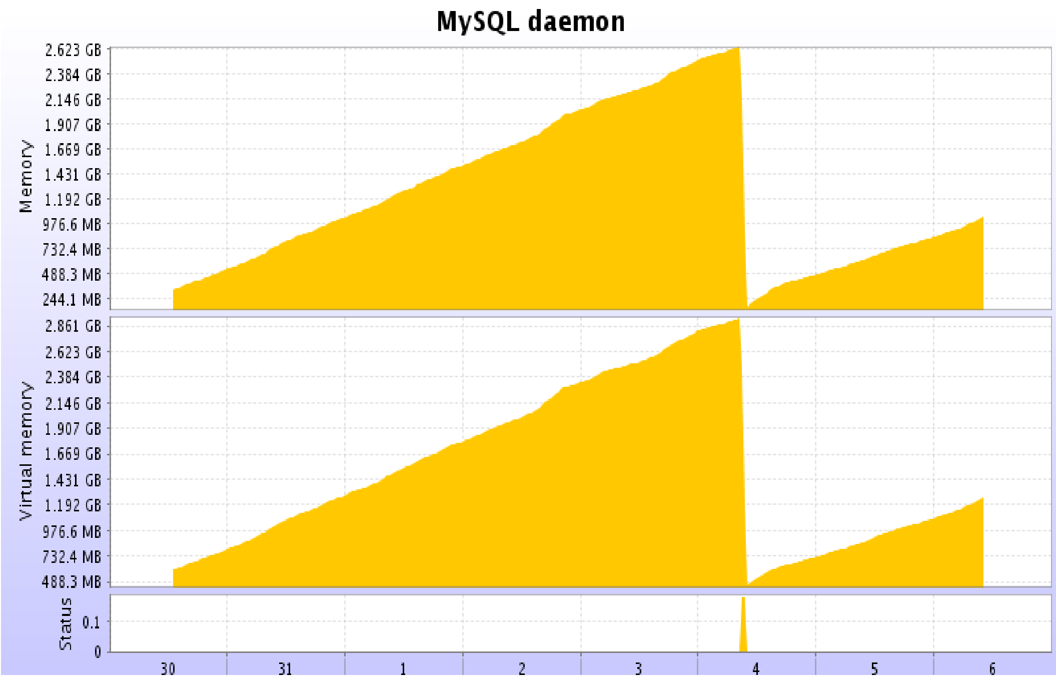
Time delays in receiving monitoring data for the control units :

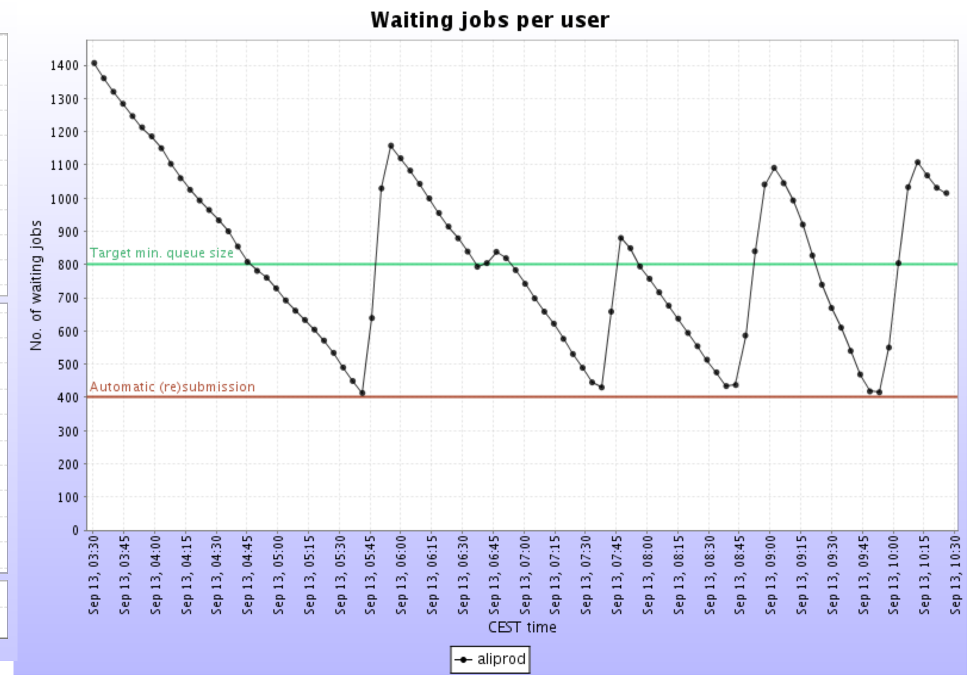
- give rise to phase lag
- degenerate system stability and performance

Maximize temporal determinism. In general, a time-lag in a feedback loop will result in overshoot and oscillation. These oscillation could fade out, continue or increase to bring the system into an unstable state.



ALICE: Automatic job submission Restarting Services



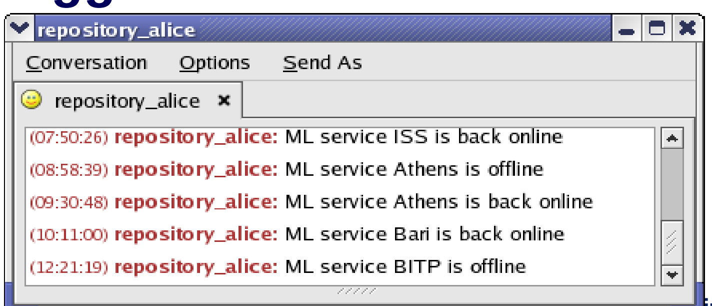


MySQL daemon is automatically Restarted when it runs out of memory

Trigger: threshold on VSZ memory usage

ALICE Production jobs queue is kept full by the automatic submission

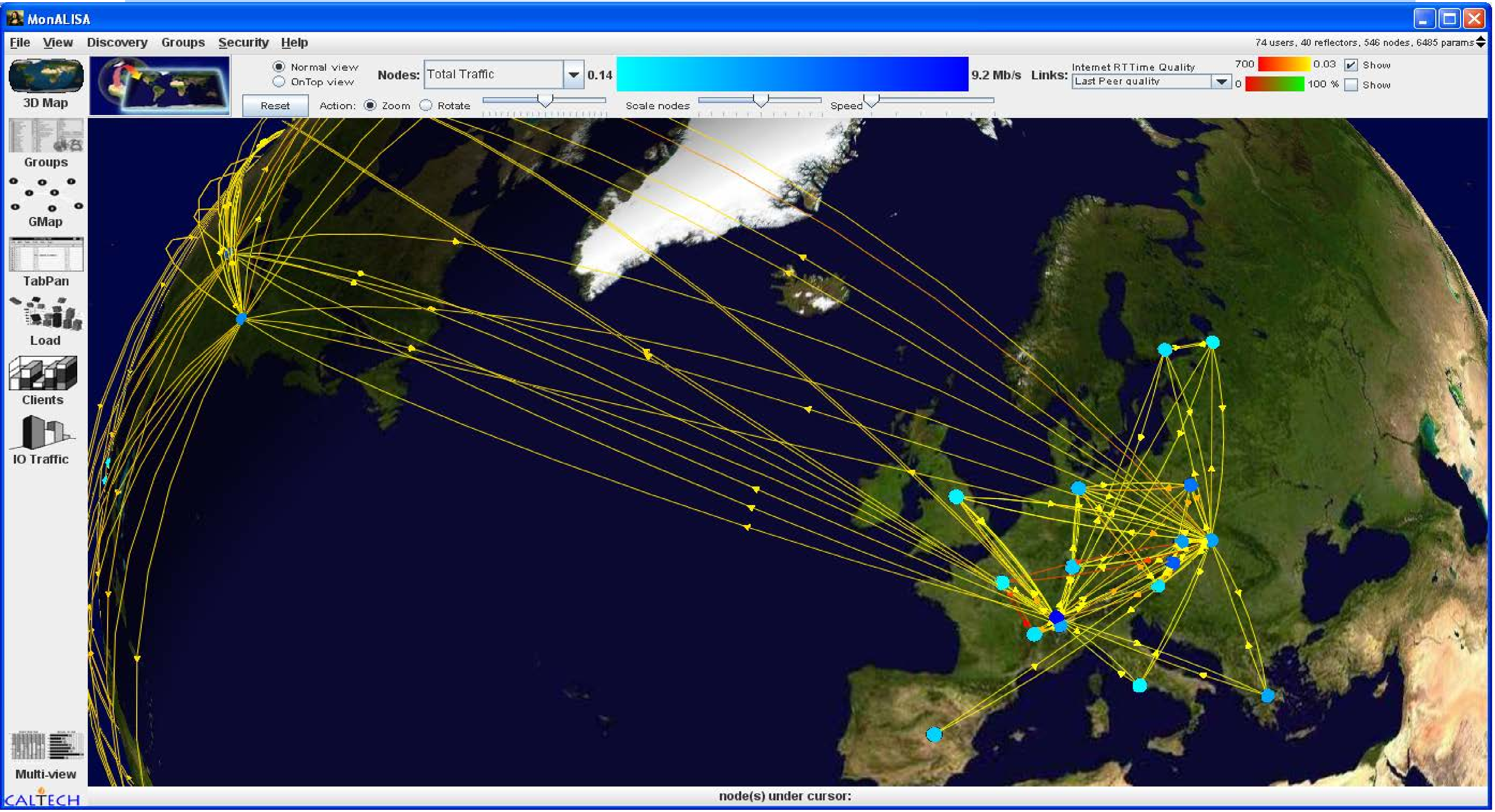
Trigger: threshold on the number of *aliprod* waiting jobs



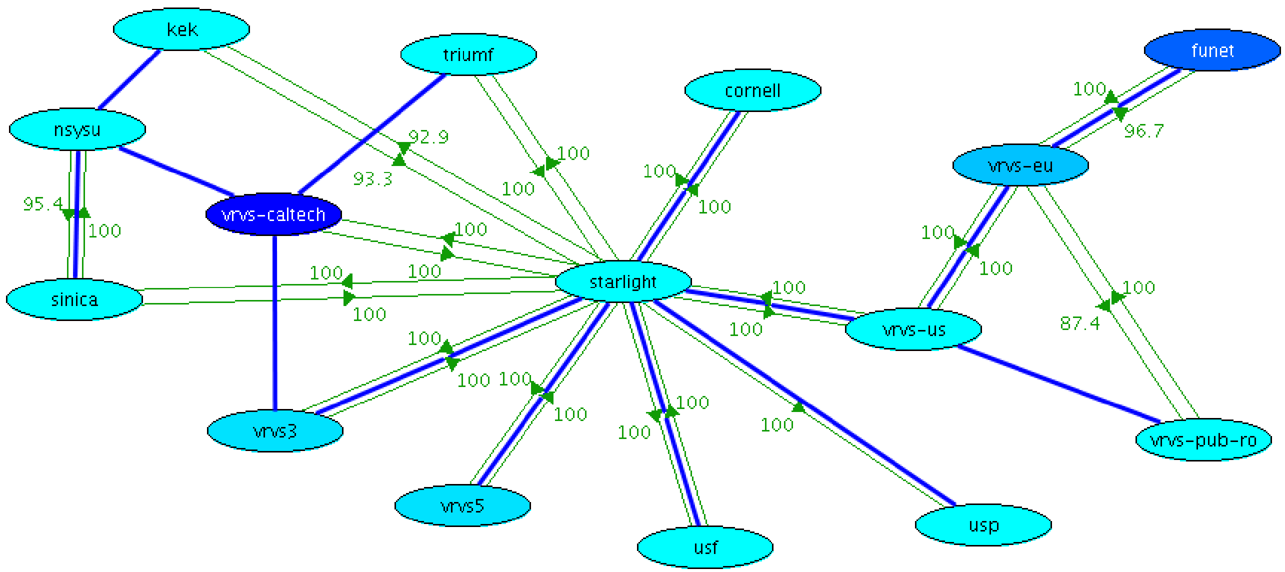
Administrators are kept up-to-date on the services' status

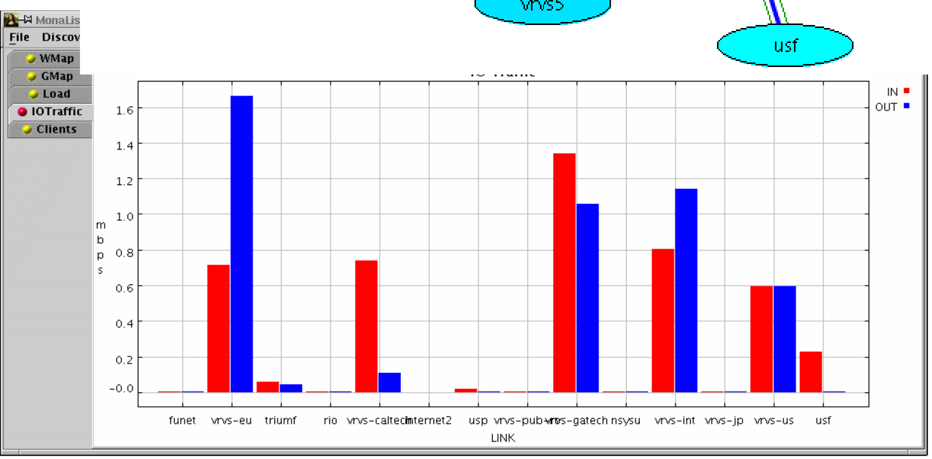
Trigger: presence/absence of monitored information

SeeVOGH (EVO) : Real-Time monitoring for Reflectors and the quality of all possible connections



EVO: Creating a Dynamic, Global, Minimum Spanning Tree to optimize the connectivity



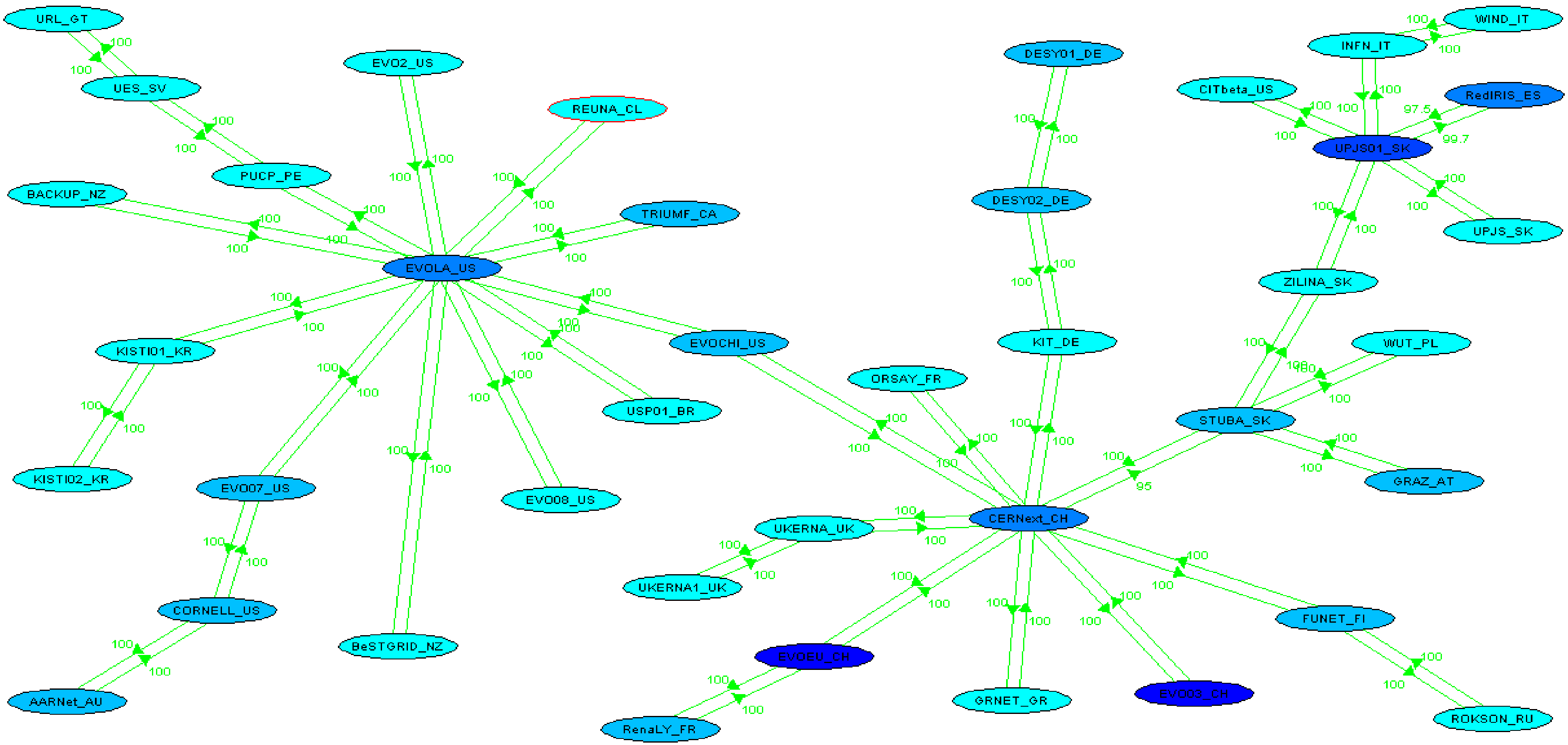


A weighted connected graph $G = (V,E)$ with n vertices and m edges. The quality of connectivity between any two reflectors is measured every second. Building in near real time a minimum-spanning tree *with addition constrains*

$$w(T) = \sum_{(v,u)\in T} w((v,u))$$

Resilient Overlay Network that optimize real-time communication

Dynamic MST to optimize the Connectivity for Reflectors

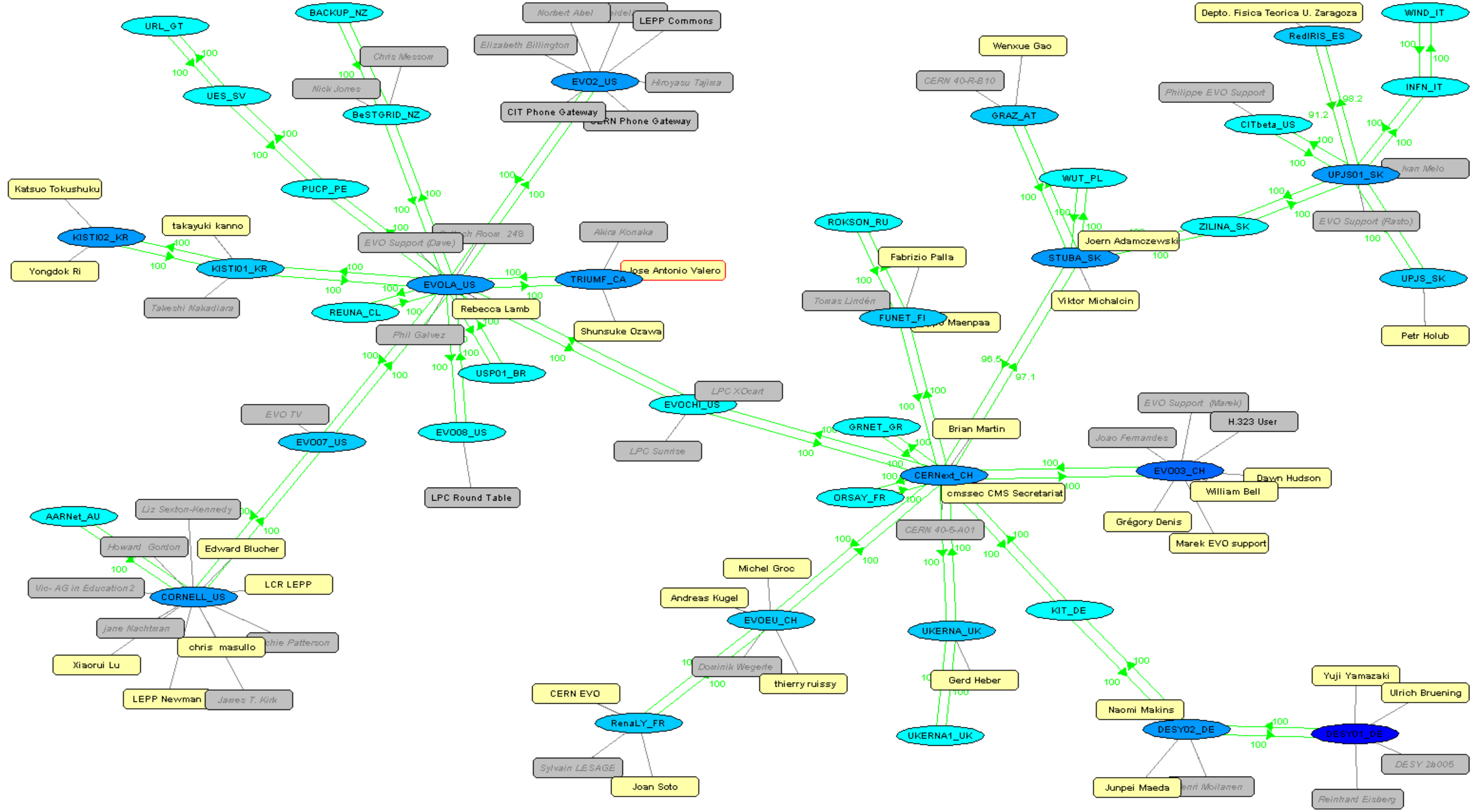


Frequent measurements of RTT, jitter, traffic and lost packages
The MST is recreated in ~ 1 S case on communication problems.

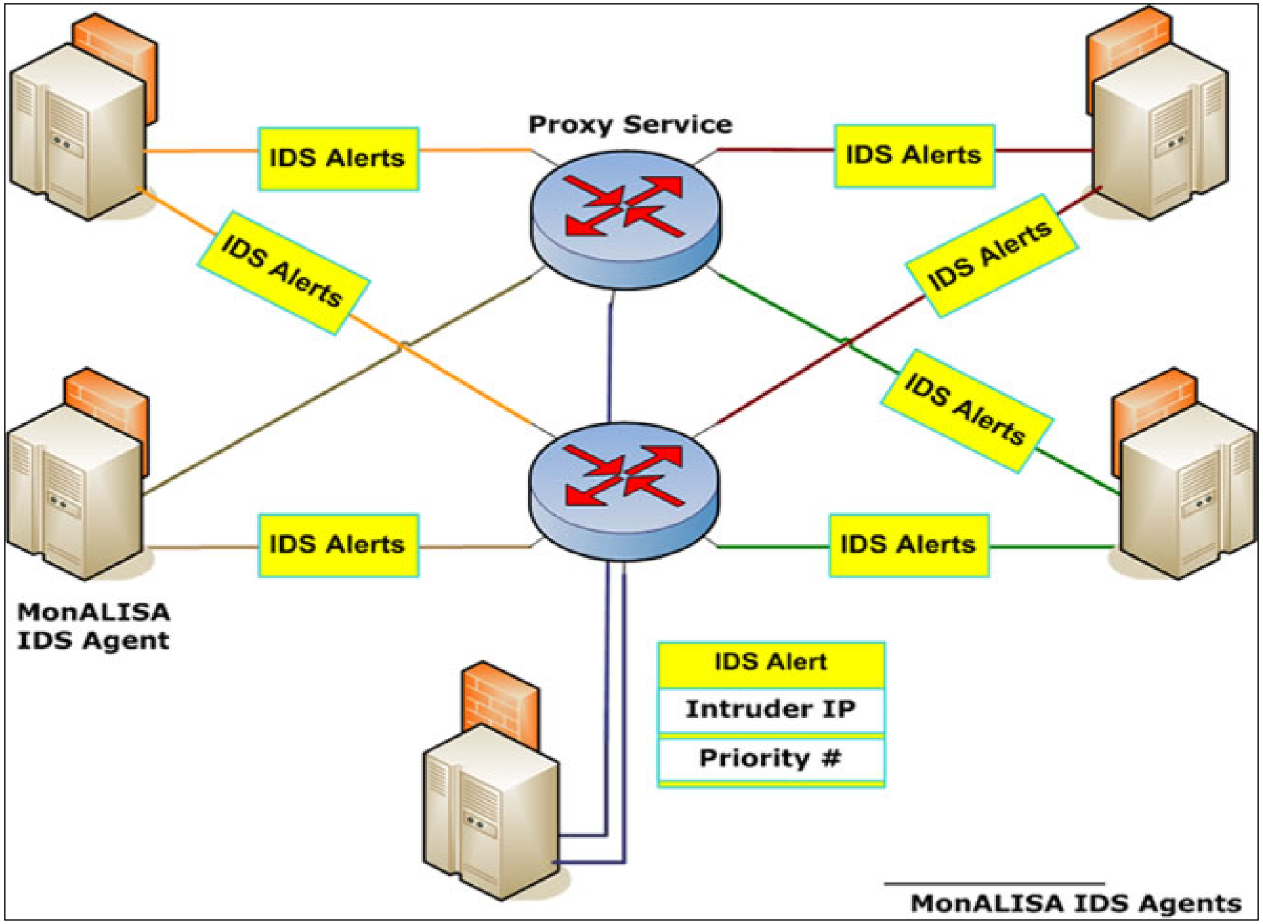
LISA- Provides an Efficient Integration for Distributed Systems and Applications

- It is using external services to identify the real IP of the end system, its network ID and AS
- Discovers MonALISA services and can select, based on service attributes, different applications and their parameters (location, AS, functionality, load …)
  - Based on information such as AS number or location, it determines a list with the best possible services.
  - Registers as a listener for other service attributes (eg. number of connected clients).
  - Continuously monitors the network connection with several selected services and provides the best one to be used from the client's perspective.
  - Measures network quality, detects faults and informs upper layer services to take appropriate decisions

SeeVOGH : Optimize how clients connect to the system for best performance and load balancing

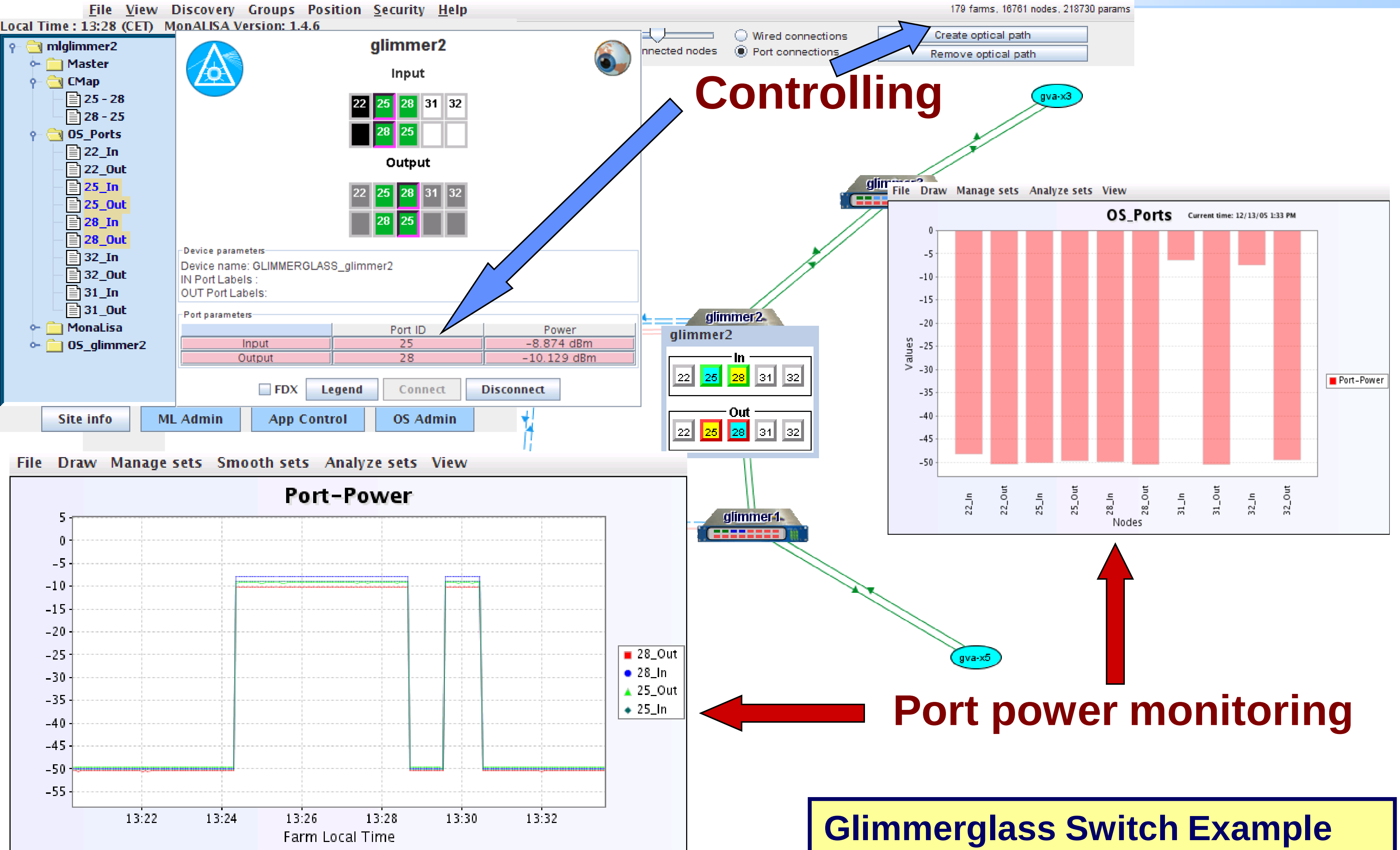


Distribute Information from protect communities from network attacks



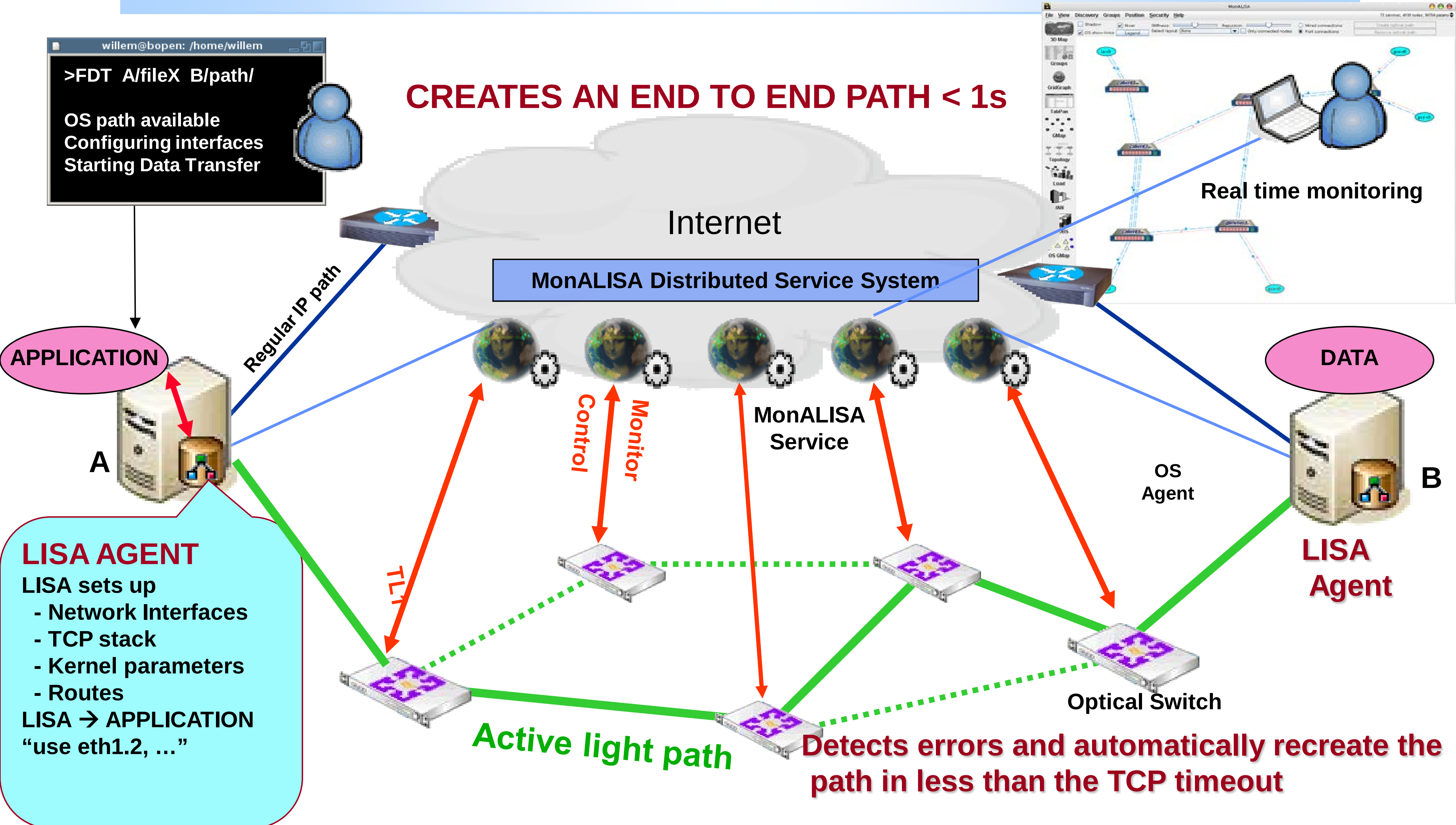
Propagate information to all peers and network devices to block the attackers .

Monitoring the Topology and Optical Power on Fibers for Optical Circuits

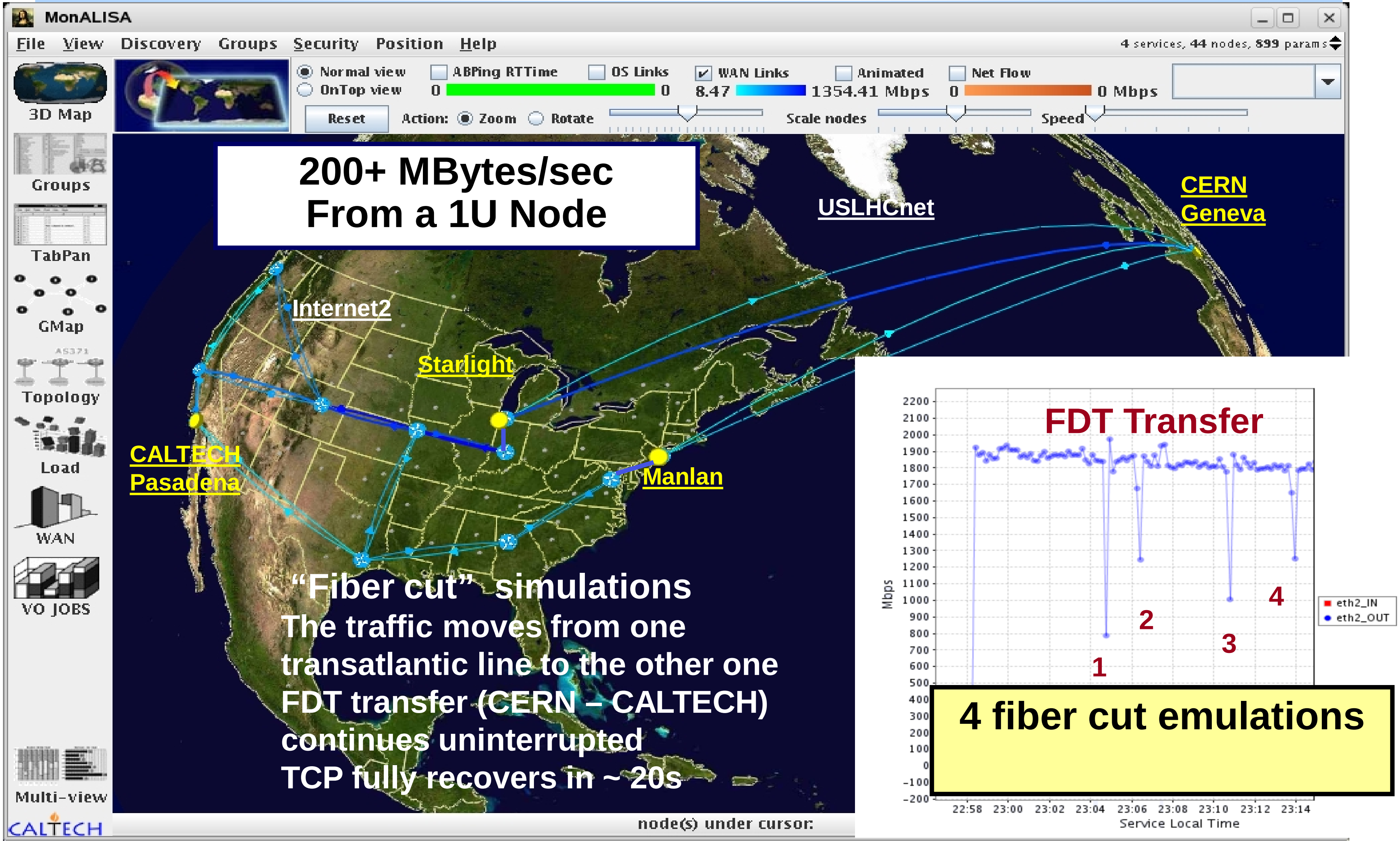


Glimmerglass Switch Example

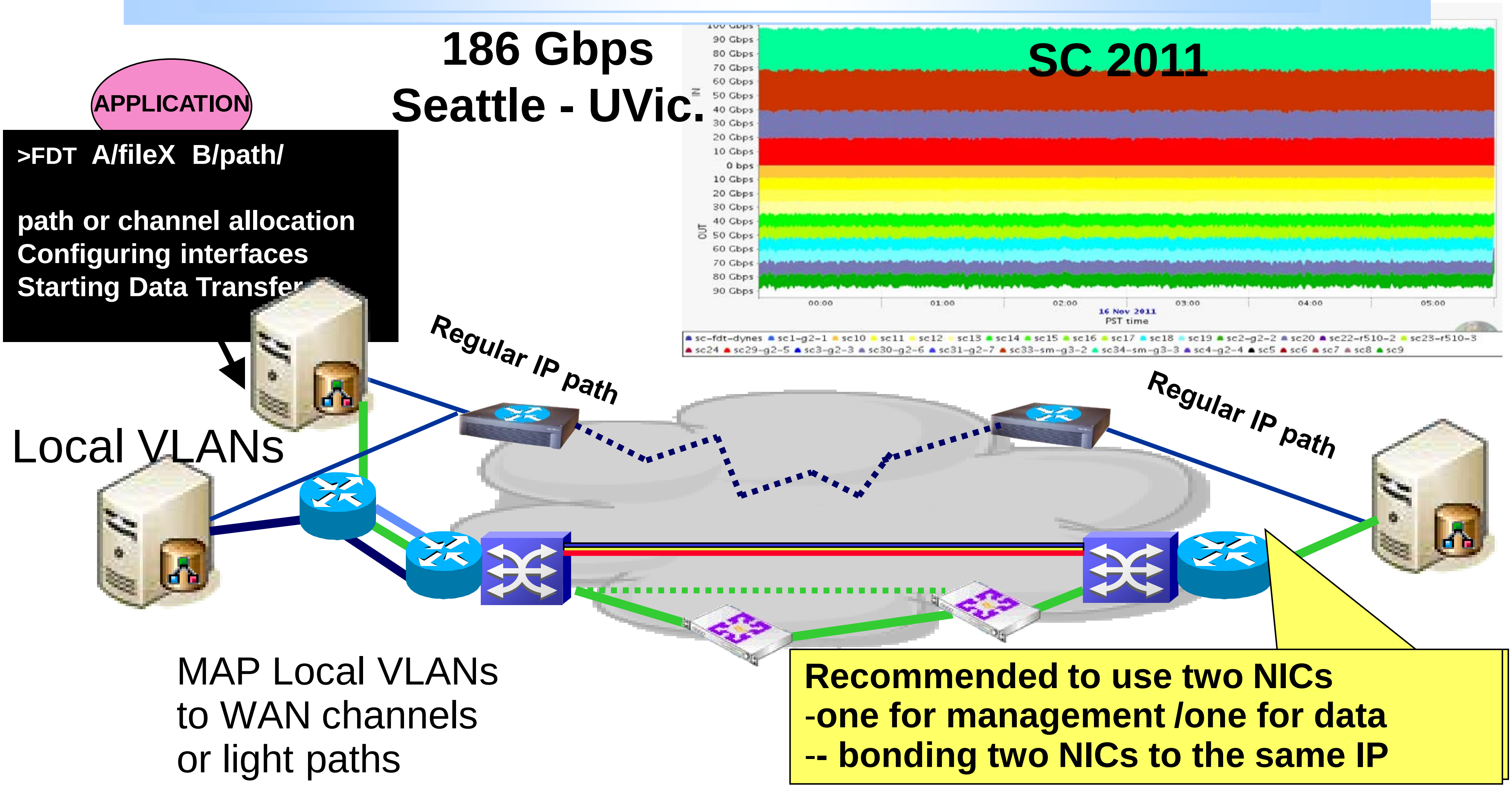
"On-Demand", End to End Optical Path Allocation



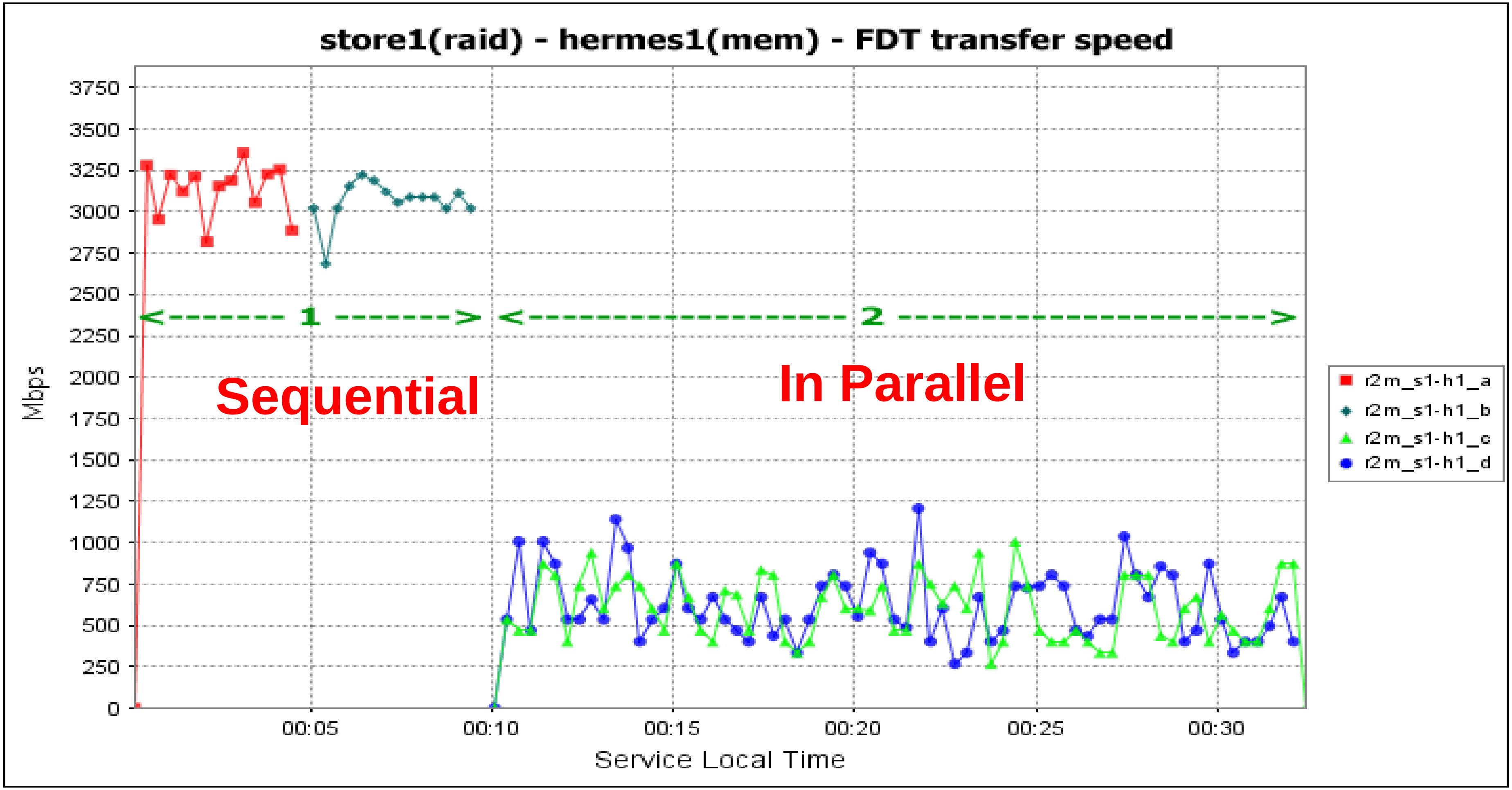
Controlling Optical Planes
Automatic Path Recovery



"On-Demand", Dynamic Circuits Channel and Path Allocation

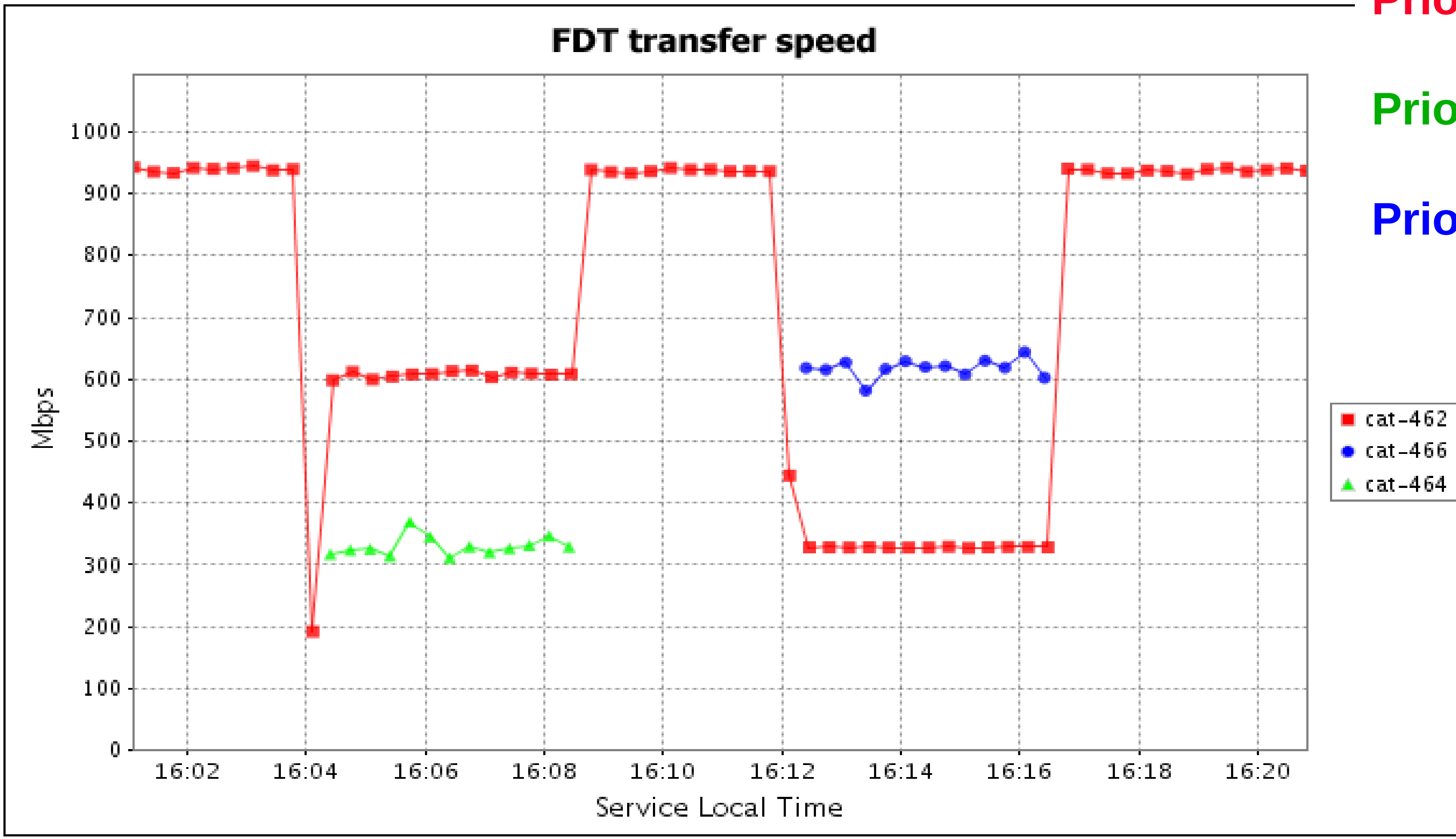


The Need for Planning and Scheduling for Large Data Transfers



2.5 X Faster to perform the two reading tasks sequentially

Dynamic priority for FDT Transfers on common segments



Priority 4

Priority 2

Priority 8

The Challenge in how to use very large amounts of monitoring information

- MonALISA collects ~ 6 millions persistent monitoring parameters and ~ 100 millions of volatile parameters per day

- We can easily access any of these parameters but in general when there problems, finding and understanding them is not so easy

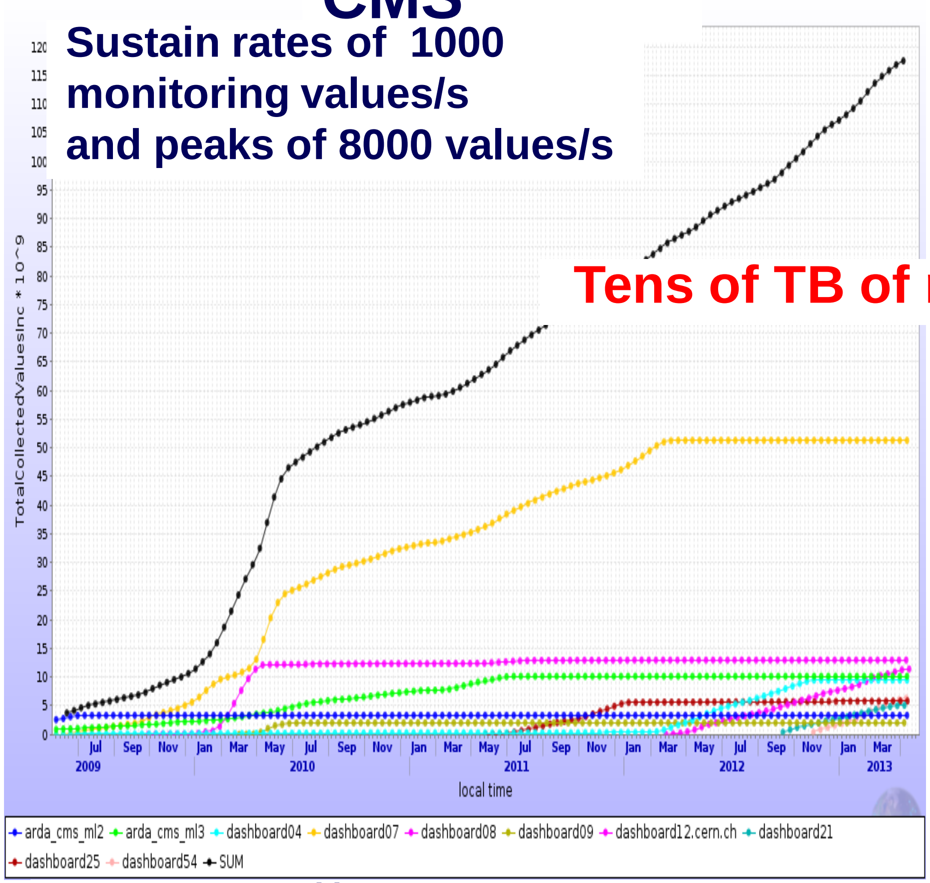
  - Are all of them correct ?
  - Do we have all the information we need ? In many cases application and services are not (correctly) instrumented .
  - Do we collect redundant information ?

- It is important to build correlation engines for most important data flows and in this way to help detect pathological problems for distributed systems

Collecting and Storing Very Large Amounts of Monitoring Information

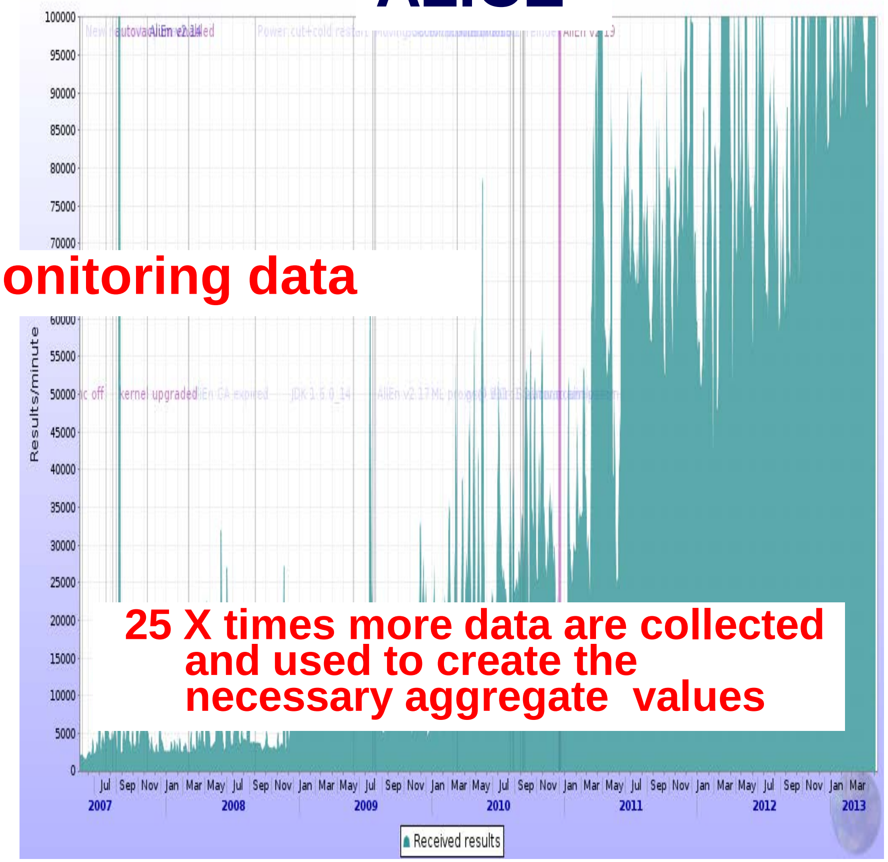
CMS

Sustain rates of 1000 monitoring values/s and peaks of 8000 values/s



Tens of TB of monitoring data

ALICE

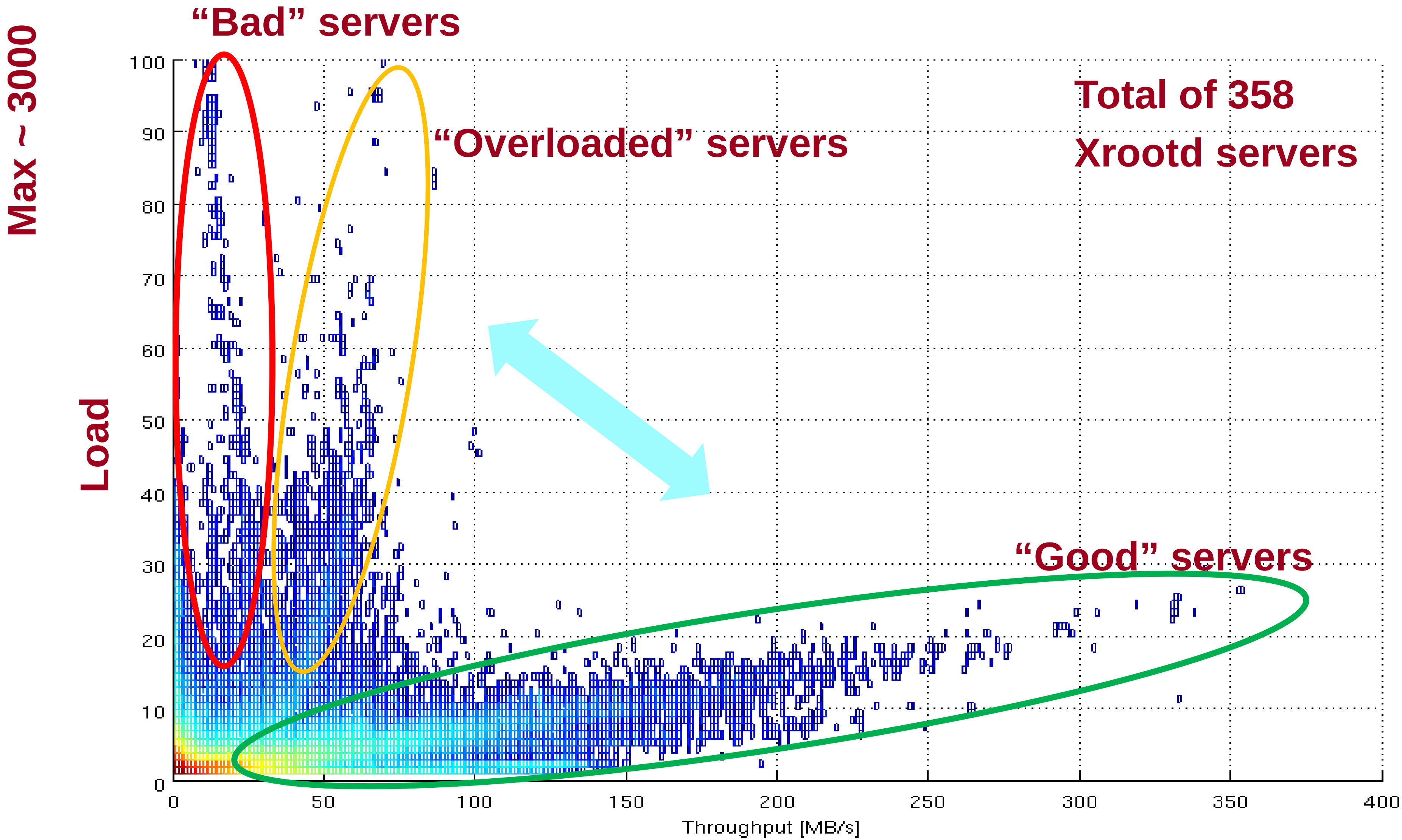


25 X times more data are collected and used to create the necessary aggregate values

~ 1.2 X 10 ^{11} Monitoring parameters Received in the CMS Dashboard

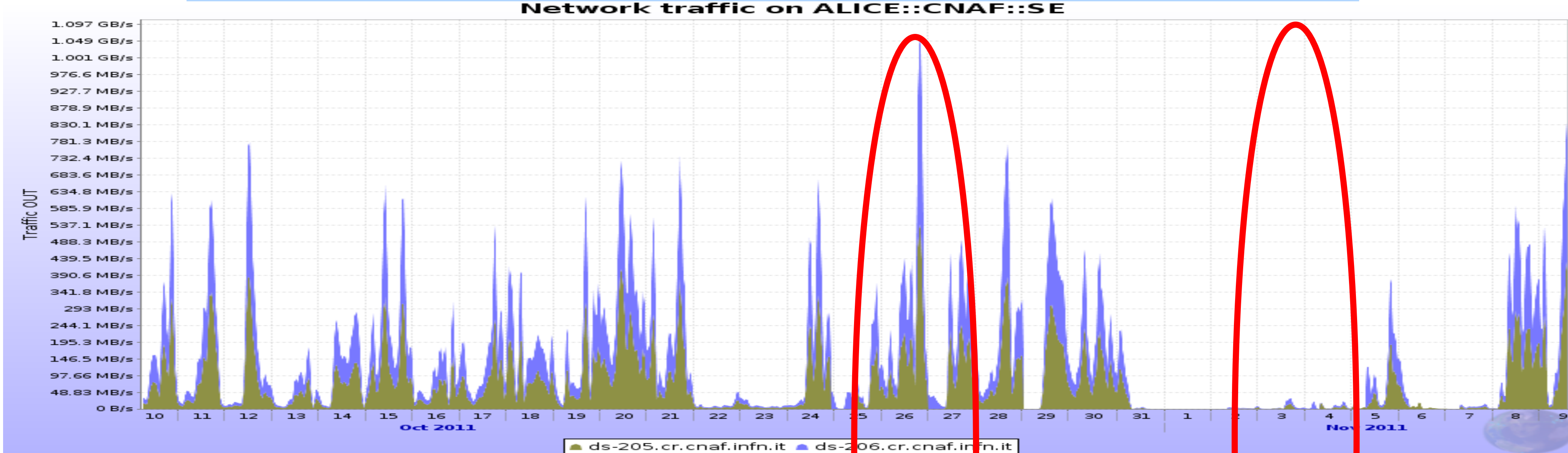
~1.3 X 10 ^{11} Monitoring parameters Received in the ALICE Repository

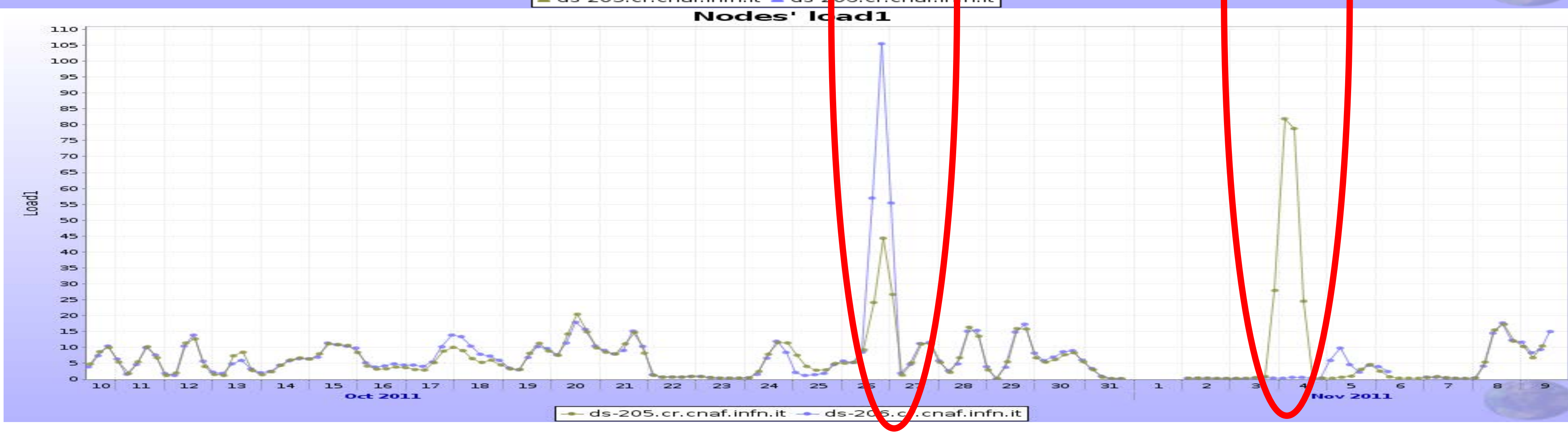
Load – Traffic distribution for all Xrood servers



86

Good Servers ... have problems





87

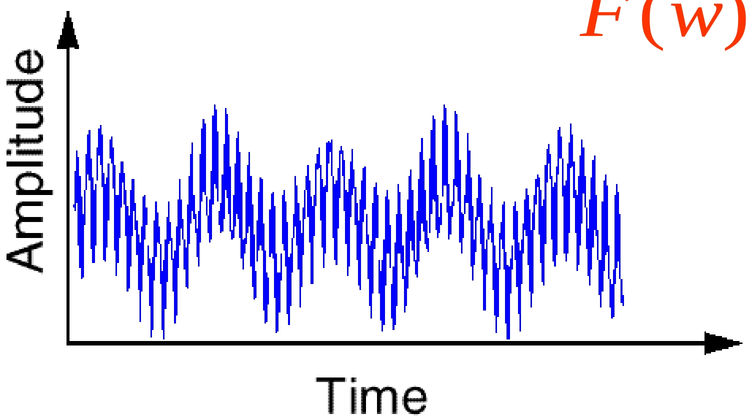
Trying to find a better way to store and analyze large sets of monitoring information

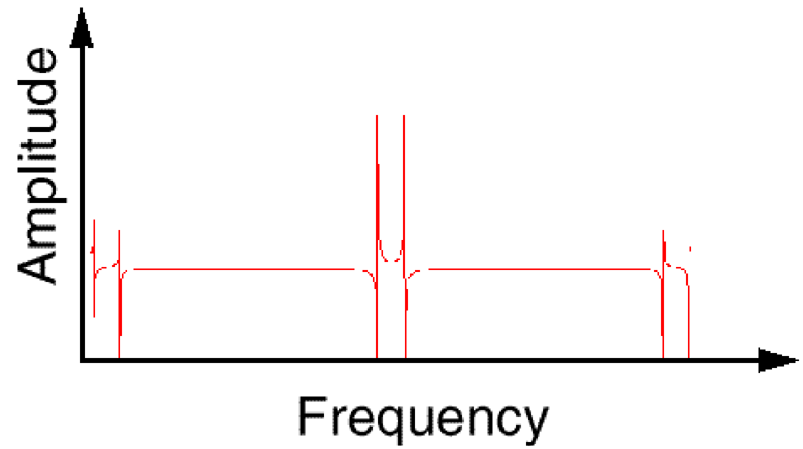
- Very large amount of monitoring information is currently collected.
- The users want more and more monitoring information, but is really difficult to analyze all the data we collect.
- Deleing older data or keeping only long term mediated values is not really a solution.
- Wavelets seems to provide an effective way to compress monitoring information and to analyze large, complex time series data.

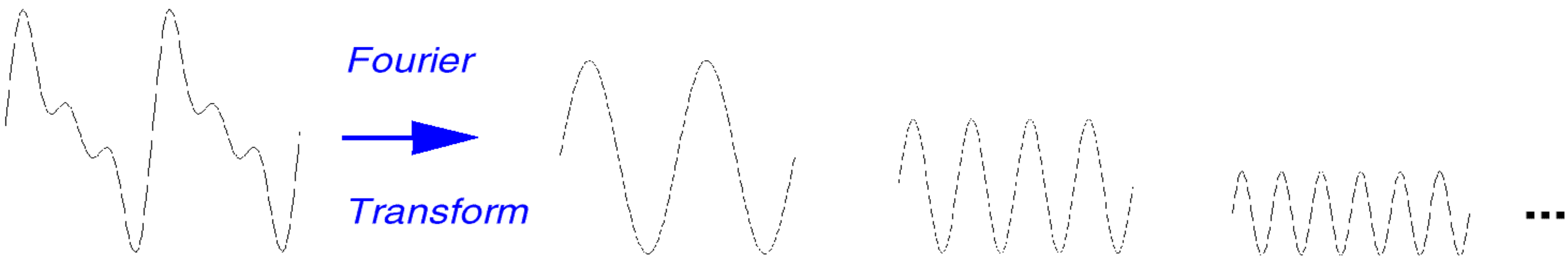
The Fourier Transform

Represents a signal into constituent sinusoids of different frequencies

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-iwt} dt$$







However it does not indicate when different frequencies occur

THE WAVELET TRANSFORM

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t) \bullet \psi^{*}\left(\frac{t-\tau}{s}\right) dt$$

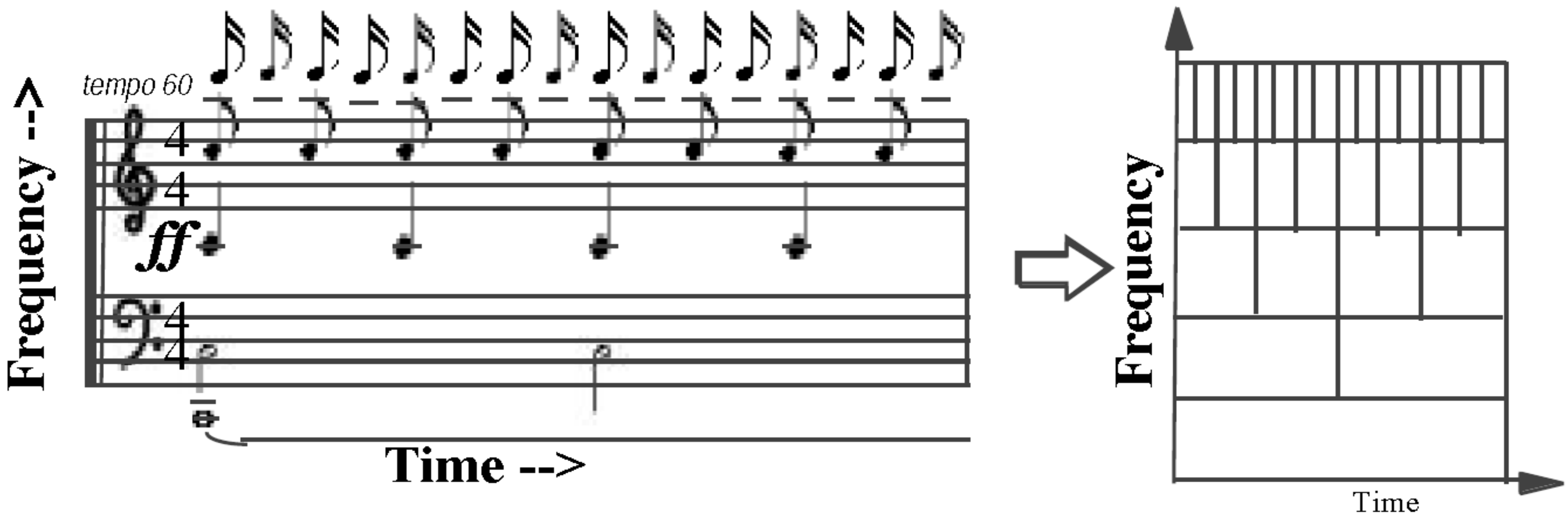
Translation

(The time location of the window)

Scale

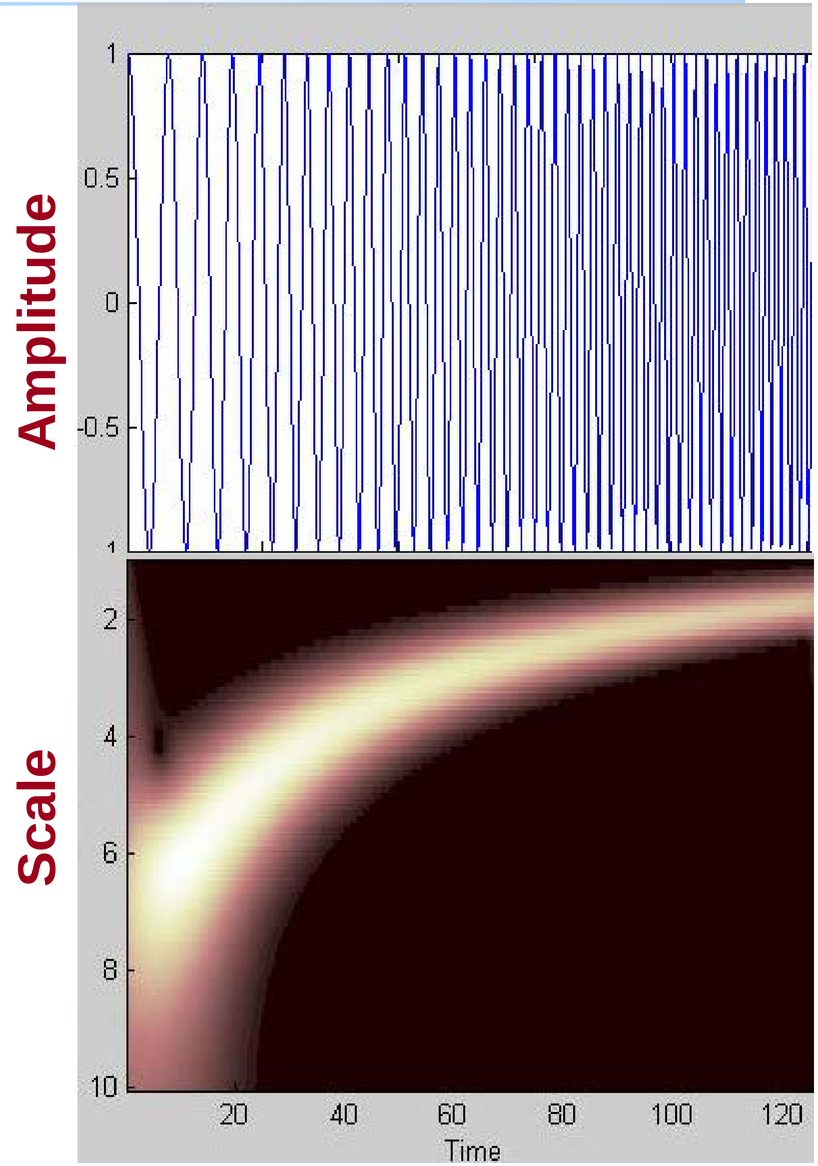
Mother Wavelet

Scale is generalized local frequency

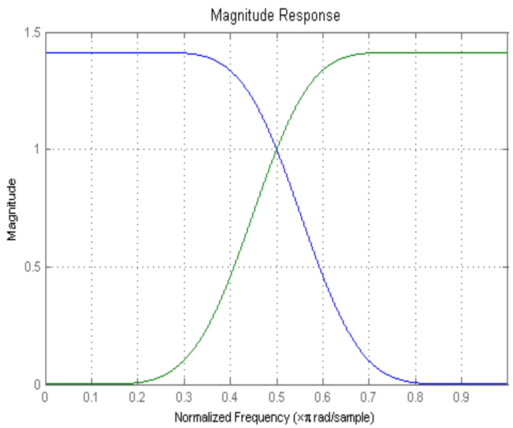


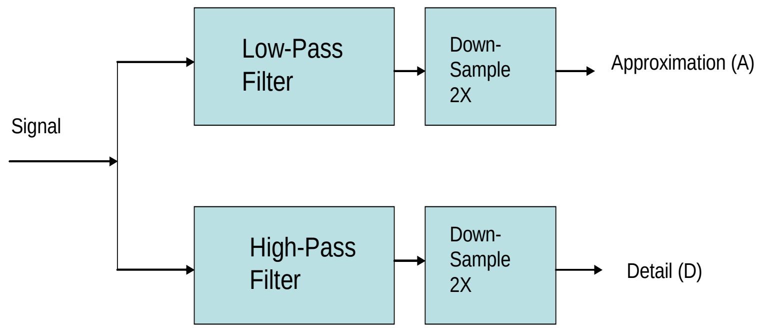
Wavelet Transform

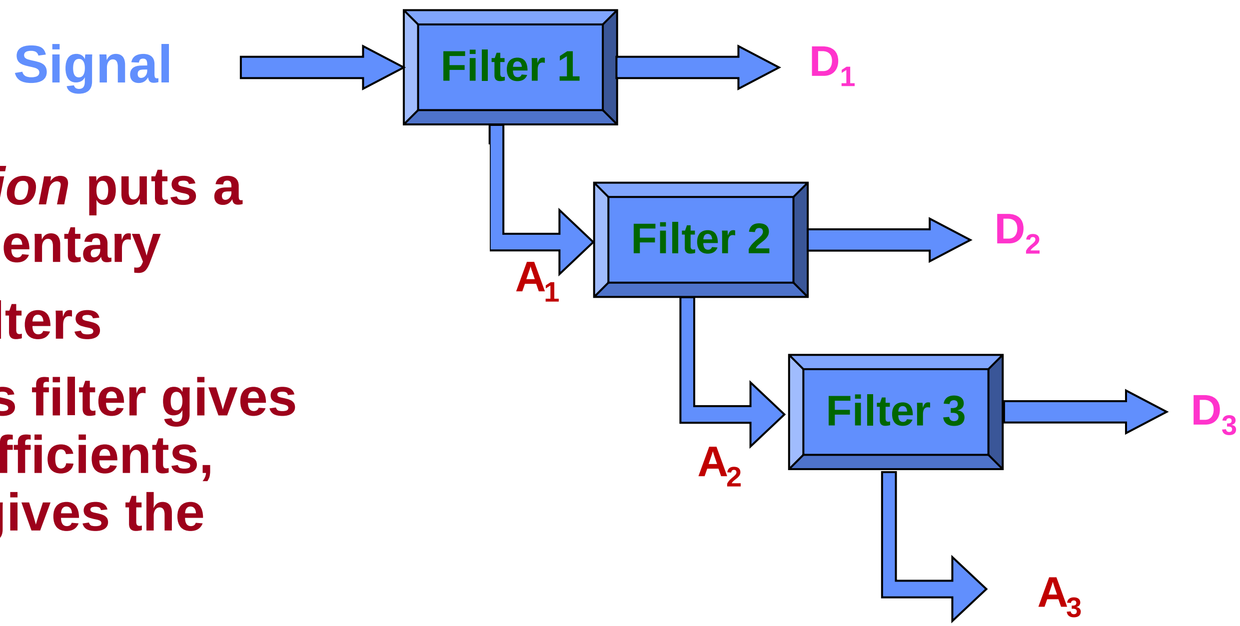
- **Provides the time-frequency representation**
- **Capable of providing the time and frequency information simultaneously**
- **WT was developed to overcome some resolution related problems of the STFT**
- **We pass the time-domain signal from various highpass and low pass filters, which filters out either high frequency or low frequency portions of the signal. This procedure is repeated, every time some portion of the signal corresponding to some frequencies being removed from the signal**



Discrete Wavelet Transform





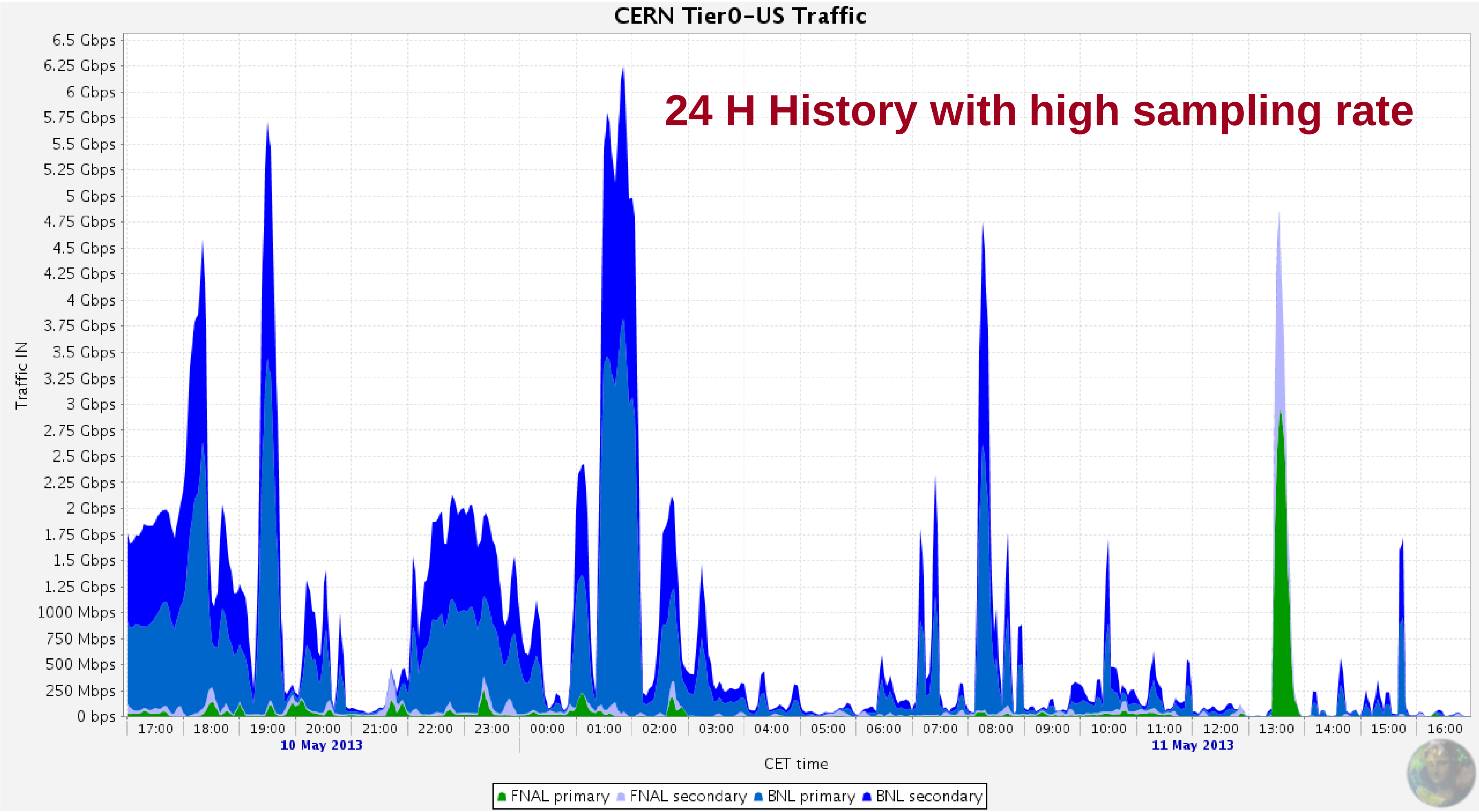


A *single level decomposition* puts a signal through 2 complementary

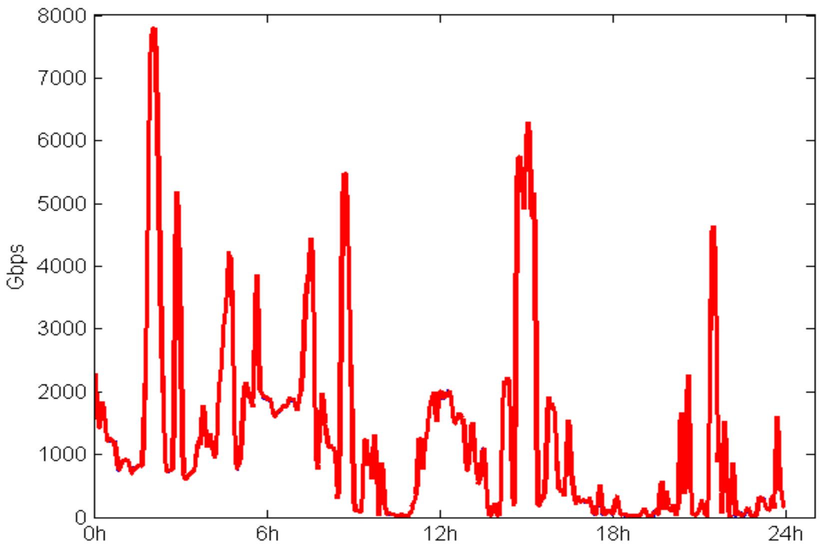
low-pass and high-pass filters

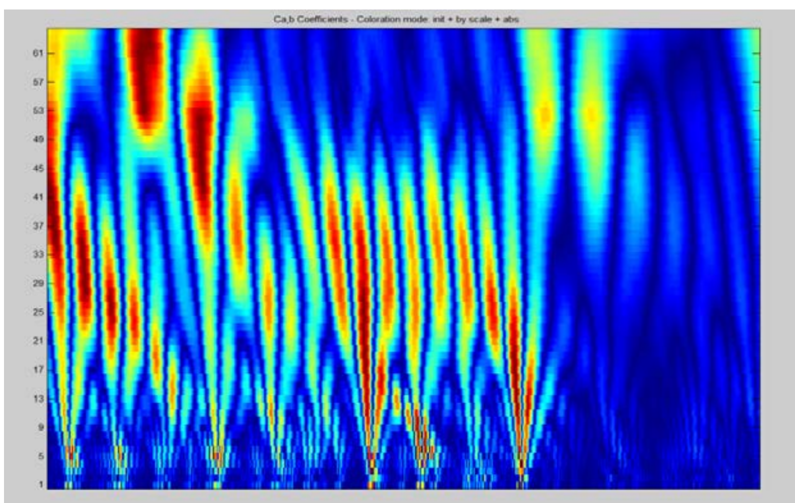
The output of the low-pass filter gives the approximation (A) coefficients, while the high pass filter gives the detail (D) coefficients

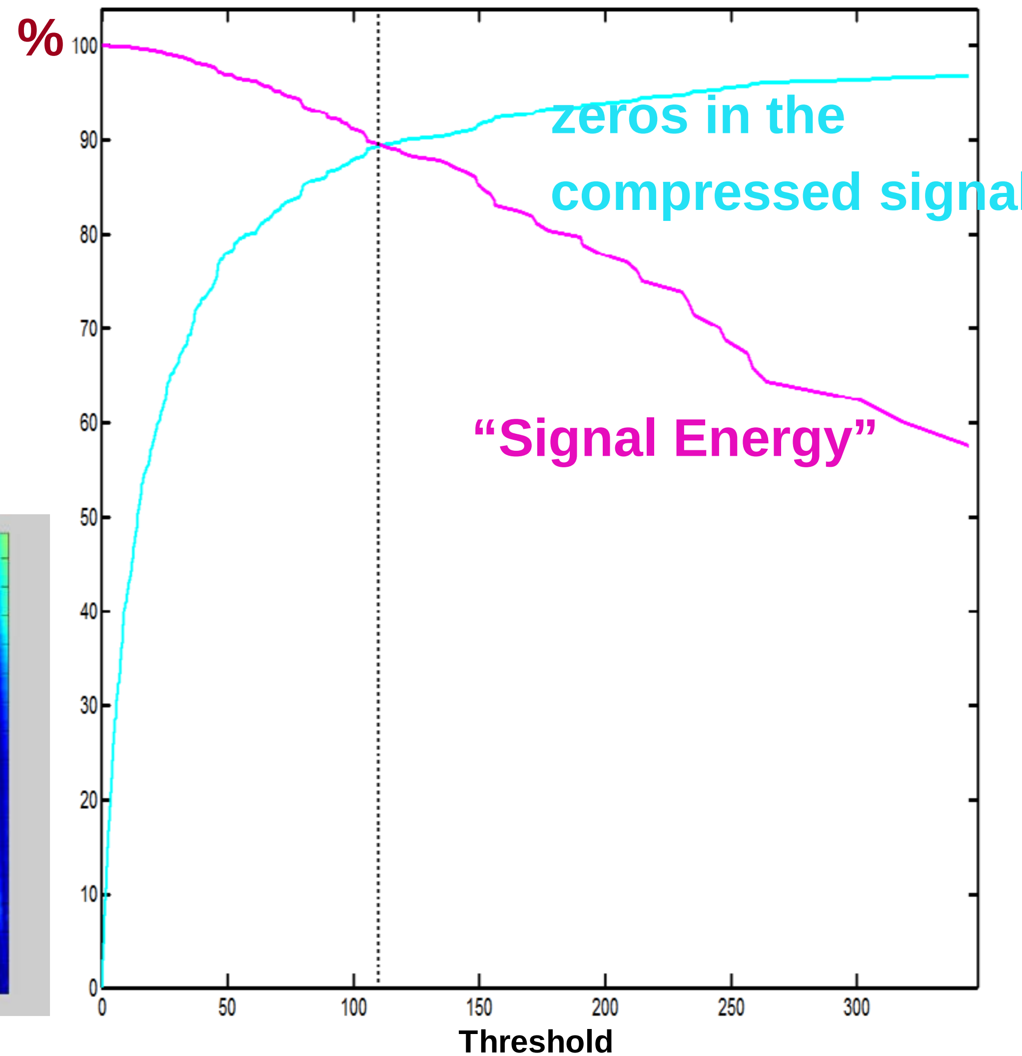
Network Traffic in USLHC net



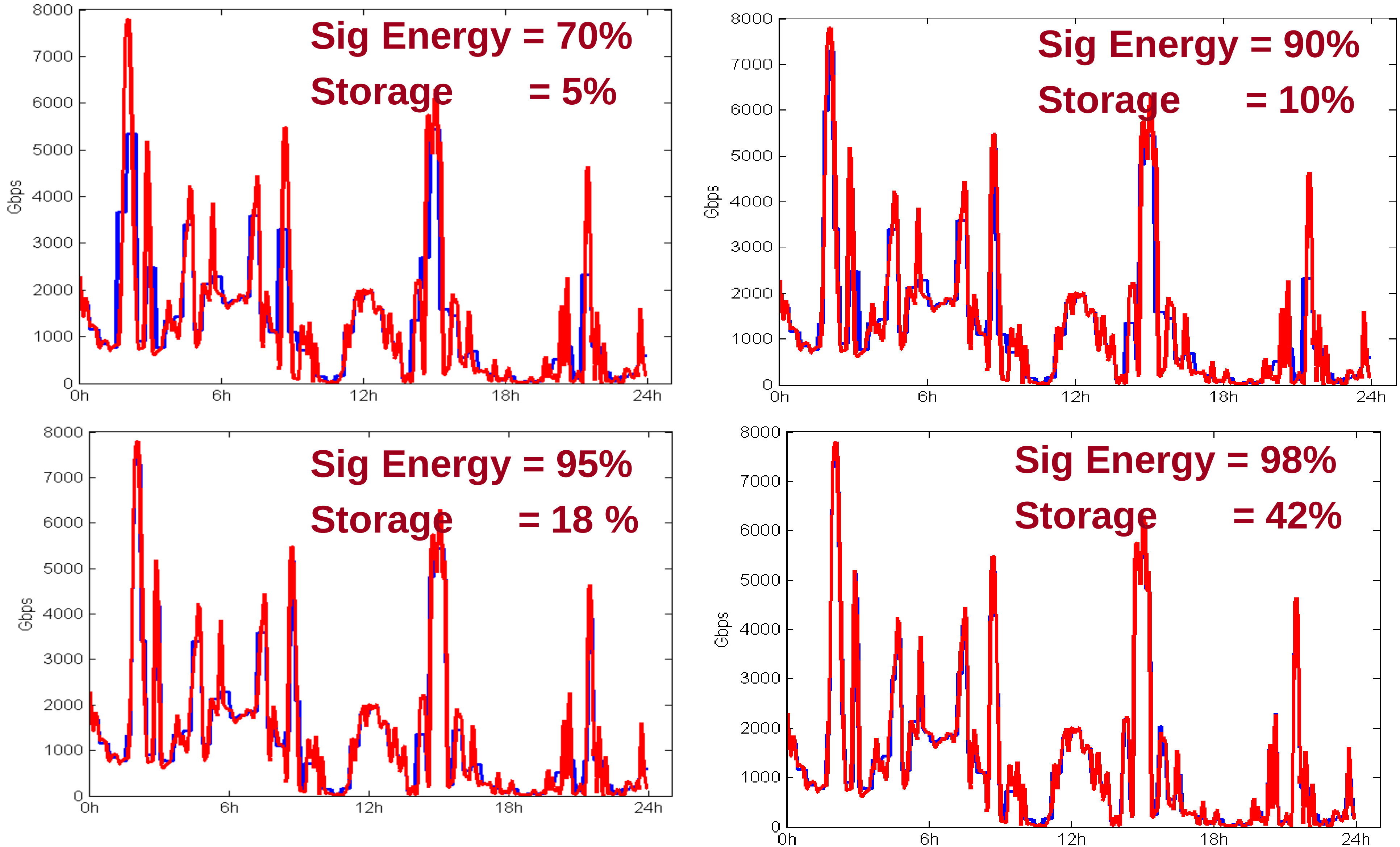
Compressing Time Series Data



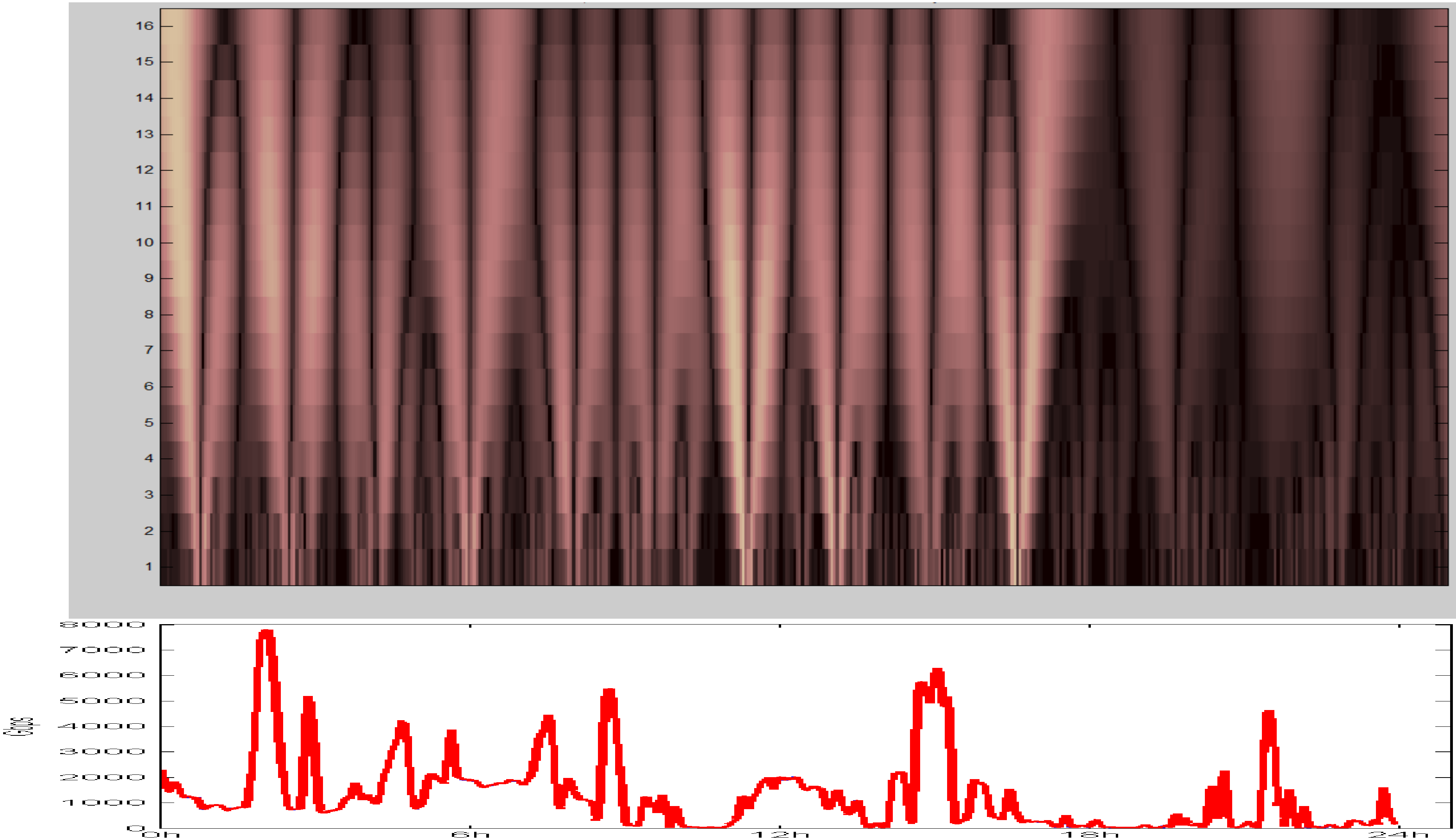




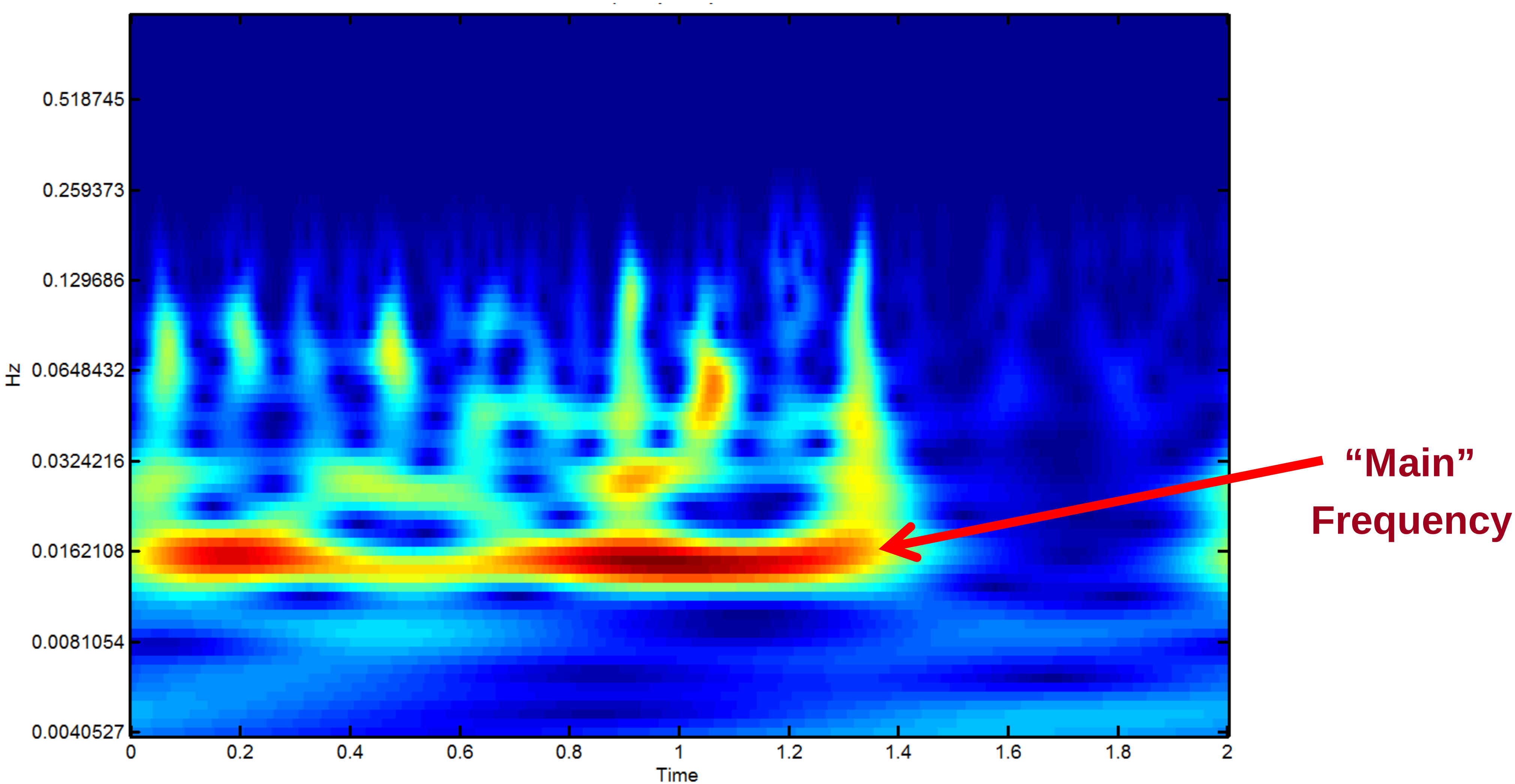
Example of Compression for high sampling network traffic data



Self Similarity Structure in the Transformed Space



Similar data transfers operations that overlap in the total traffic pattern



Detecting Clusters in Multiple Time Series Data Using Wavelets

- Use a multidimensional grid structure onto data space
- These multidimensional spatial data objects are represented in an n-dimensional feature space
- Apply wavelet transform on feature space to find the dense regions in the feature space
- Apply wavelet transform multiple times which result in clusters at different scales from fine to coarse

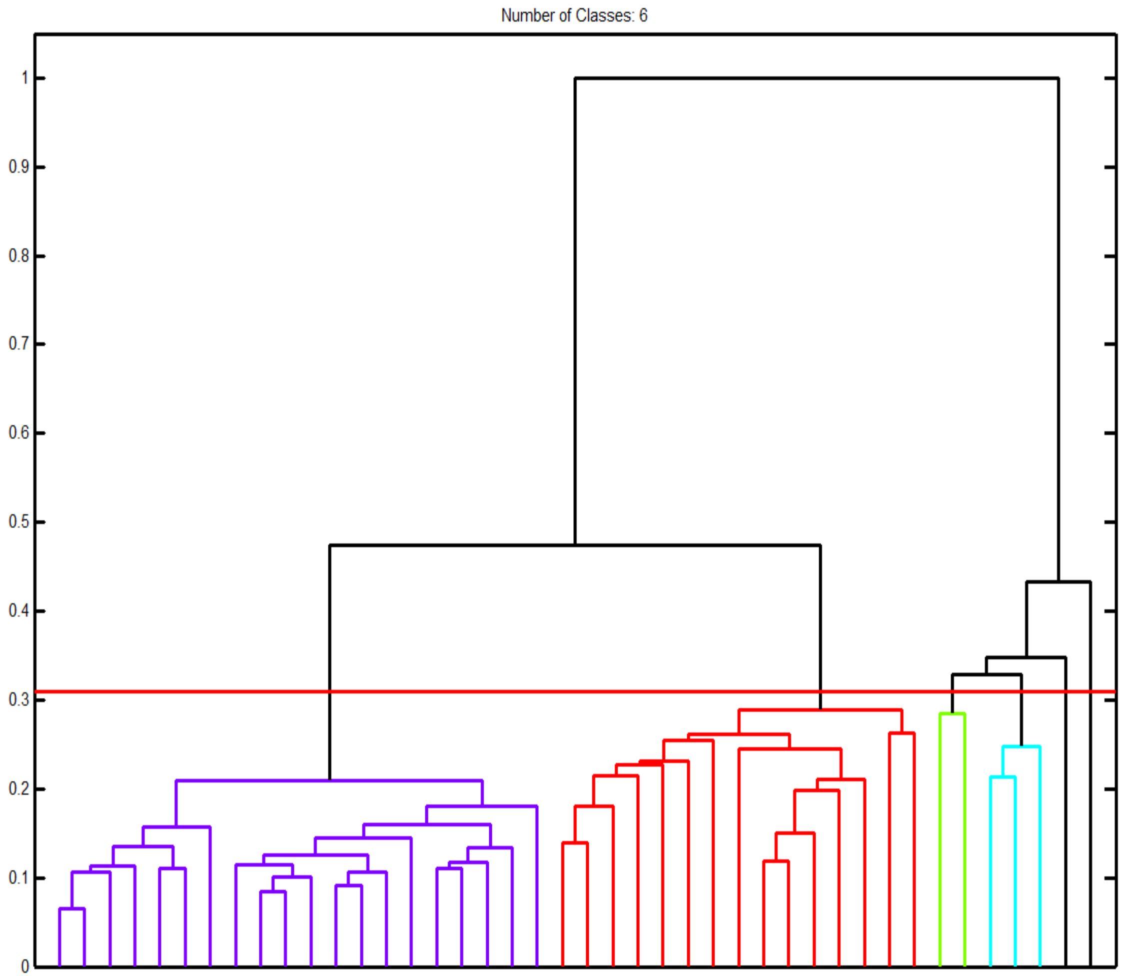
Two months history for the efficiency on all Alice grid sites

High resolution data

> 20 000 Values per time series

Not so easy to understand

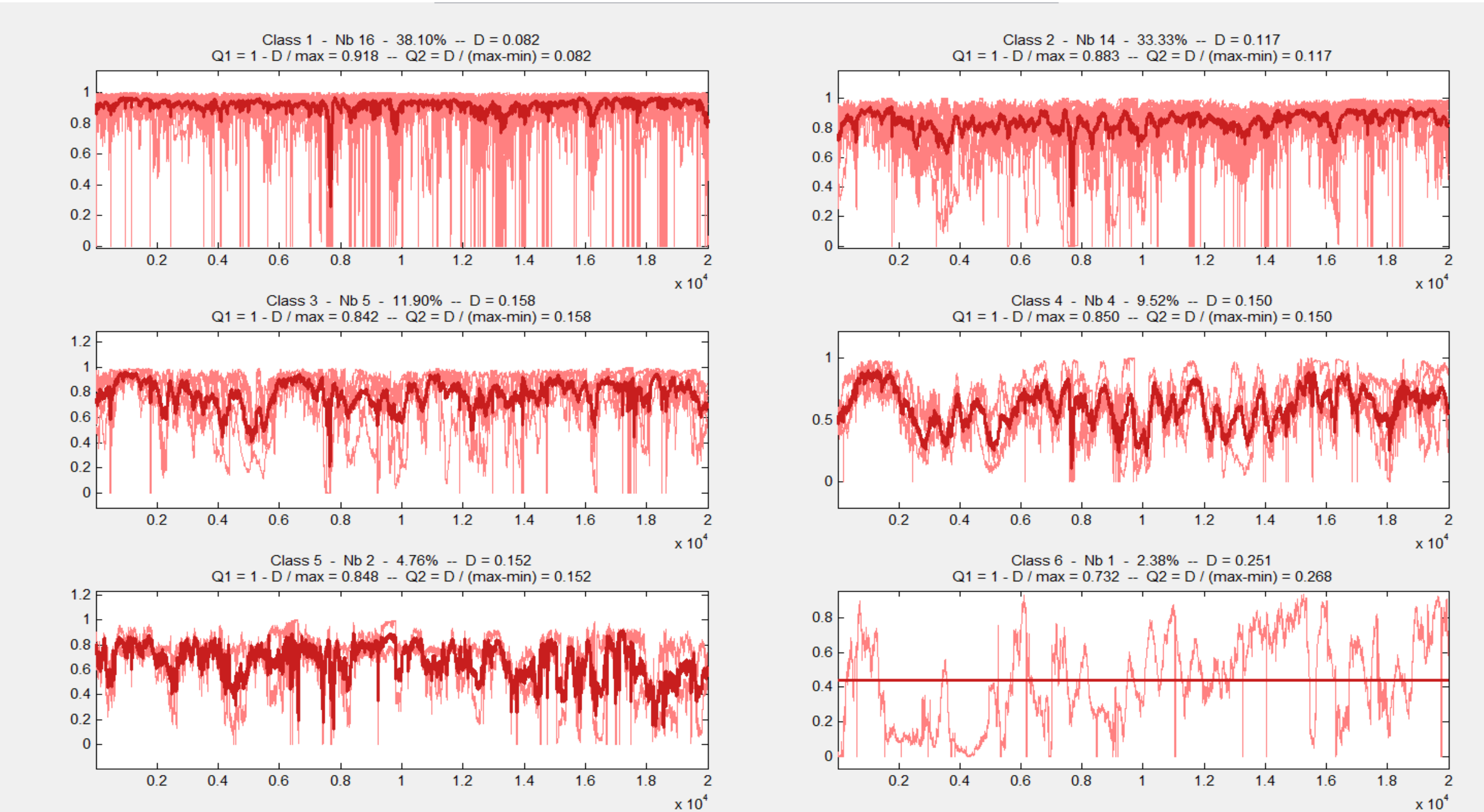
Acceding hierarchical clusters for the site efficiency plot



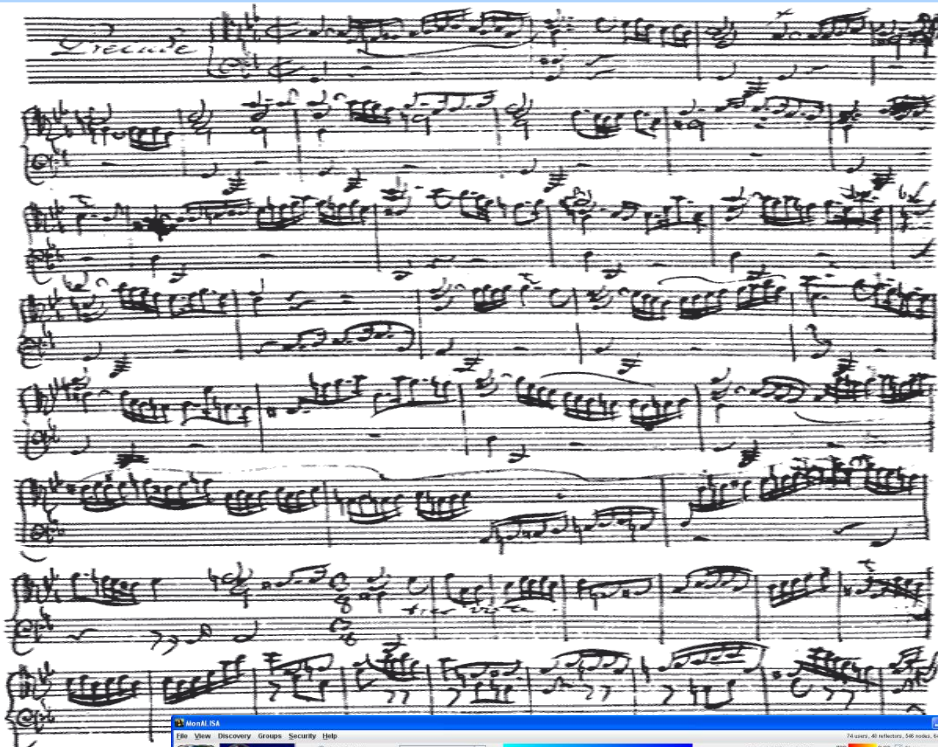
Major Advantages:

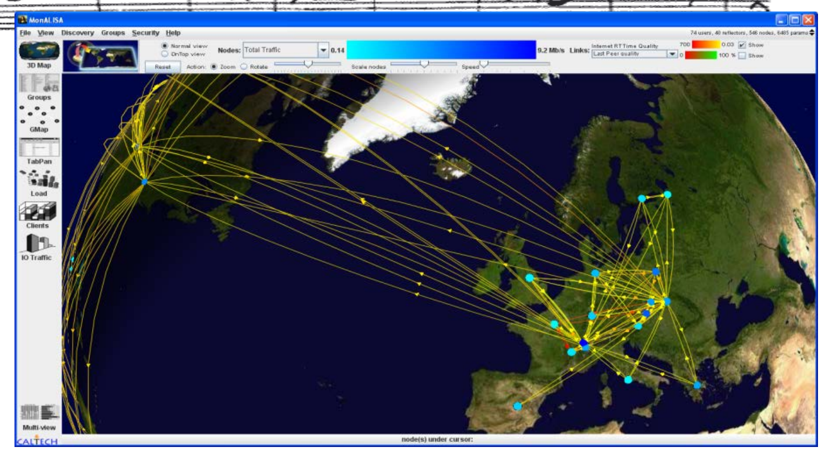
- Complexity O(N)
- Detect arbitrary shaped clusters at different scales
- Not sensitive to noise, not sensitive to input order

Results for cluster classification for sites



Monitoring: A listener to the “grid” orchestra ?

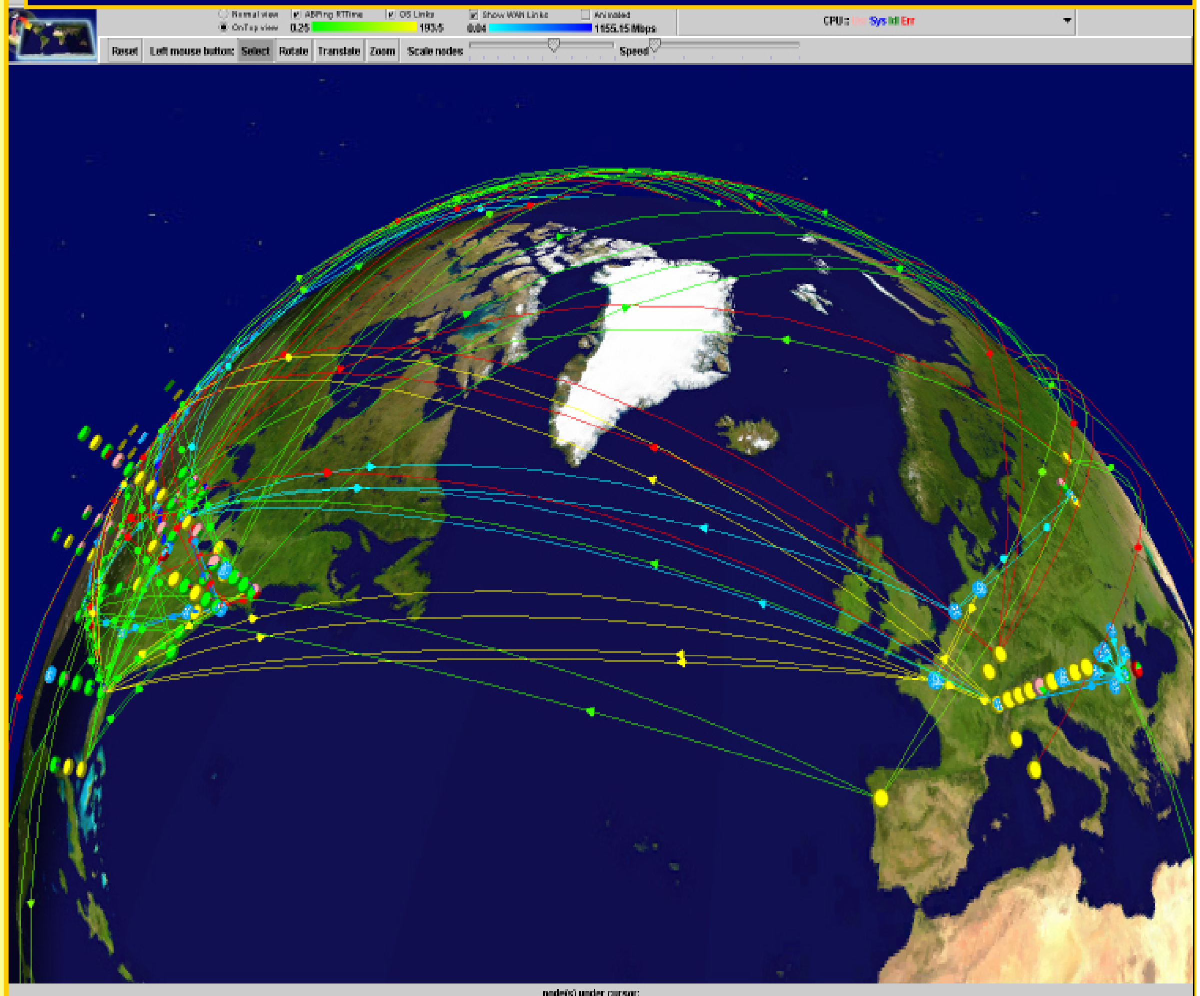




Can we analyze and use in near real time monitoring data in this form ?

MonALISA Today

Running 24 X 7 at ~370 Sites



- Monitoring
  - 60,000 computers
  - > 100 Links Major Nets
- Tens of Thousands of Grid jobs running concurrently
- > 14,000 end-to-end network path measurements
- Using Intelligent Agents
- Collecting > 6mil persistent parameters in real-time
- ~ 100 millions of volatile parameters per day
- Updating ~ 35,000 parameters per second
- Repository servers 10 mil. users request / year

The eight fallacies of distributed computing

It is fair to say that at the beginning of this project we underestimated some of the potential problems in developing large distributed systems in WAN, and indeed the "eight fallacies of distributed computing" are very important lessons:

1) **The network is reliable.**
2) **Latency is zero.**
3) **Bandwidth is infinite.**
4) **The network is secure.**
5) **Topology doesn't change.**
6) **There is one administrator.**
7) **Transport cost is zero.**
8) **The network is homogeneous.**

The MonALISA Experience

- Unified platform for all the monitoring information
- Service Oriented Architecture ; Dynamic Discovery
- Agent model for monitoring modules / filters and actions
- Functionality to dynamically subscribe for any type of information "on the fly"
- Use of a simple and efficient communication approach (problems with RMI, XML ….)
- Multithread approach is instrumental for the performance and reliability for the system
- Good Graphical views & representations to present information
- Simple and efficient approach for storing data