# Monitoring and controlling VRVS Reflectors



**Catalin Cirstoiu**

**3/7/2003**

# Project goals

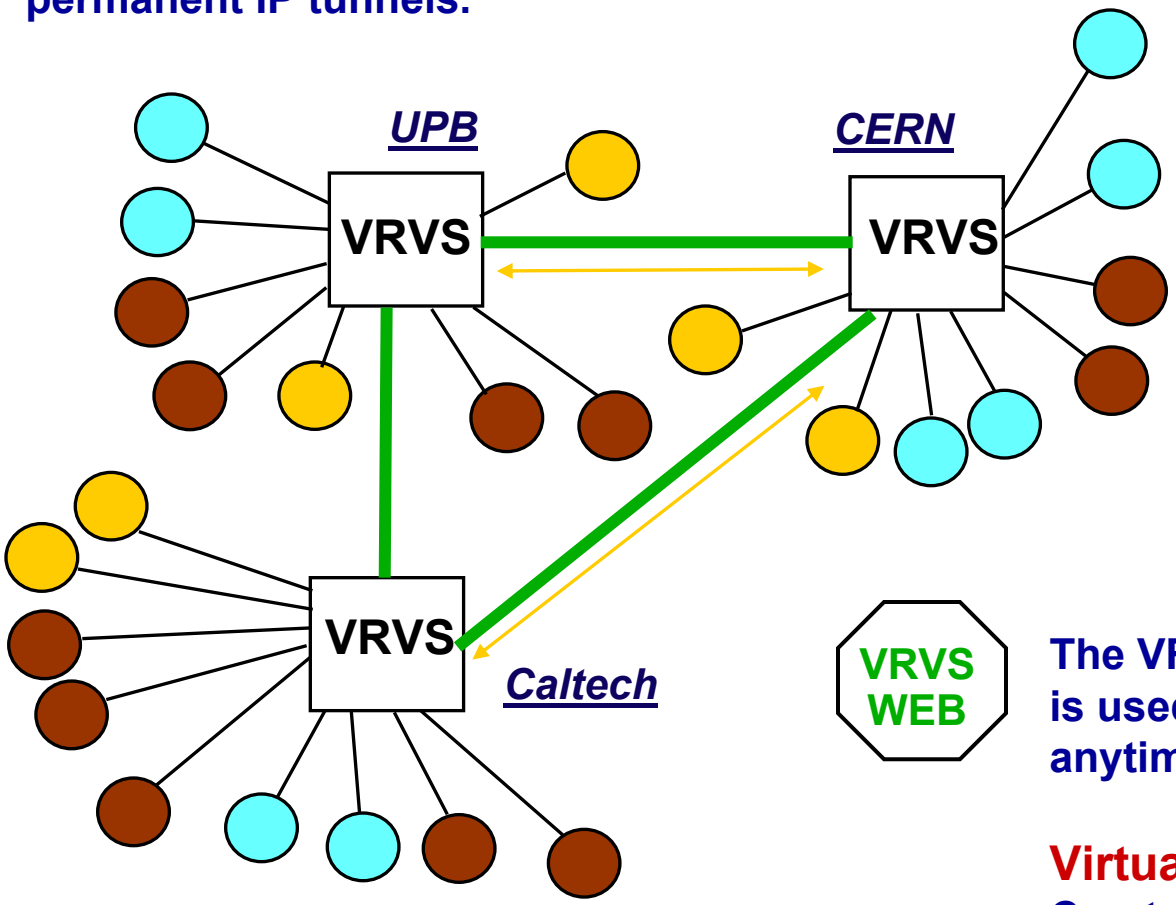
- Develop monitoring agents and alarm triggers for the VRVS system
- Develop a GUI monitoring client to observe in real-time the status of the whole system
- Develop a controlling client for the VRVS reflectors to optimize the network traffic of the multimedia packets

# What is VRVS?

- **Virtual Room Videoconferencing System**
  - a web oriented system for videoconferencing and collaborative work over IP networks
  - was initially (1997) built to provide a relatively low cost system for videoconferencing and remote collaboration over networks for the HENP community
  - now: 13750 machines, 7450 users, 61 countries
  - It provides:
    - Unified Video, Audio, Shared Virtual Spaces and Data Applications
    - Point to Point and Multi-point communication
    - Multiple Architecture Support: UNIX's, Linux, Win, Mac…
    - Network optimization (Latency, routing, reliability)
    - Should be highly scalable

# VRVS Architecture (Peer-to-Peer)

**Reflectors** are hosts that interconnect users by permanent IP tunnels.




The active IP tunnels must be selected so that there is no cycle formed.

**Tree**

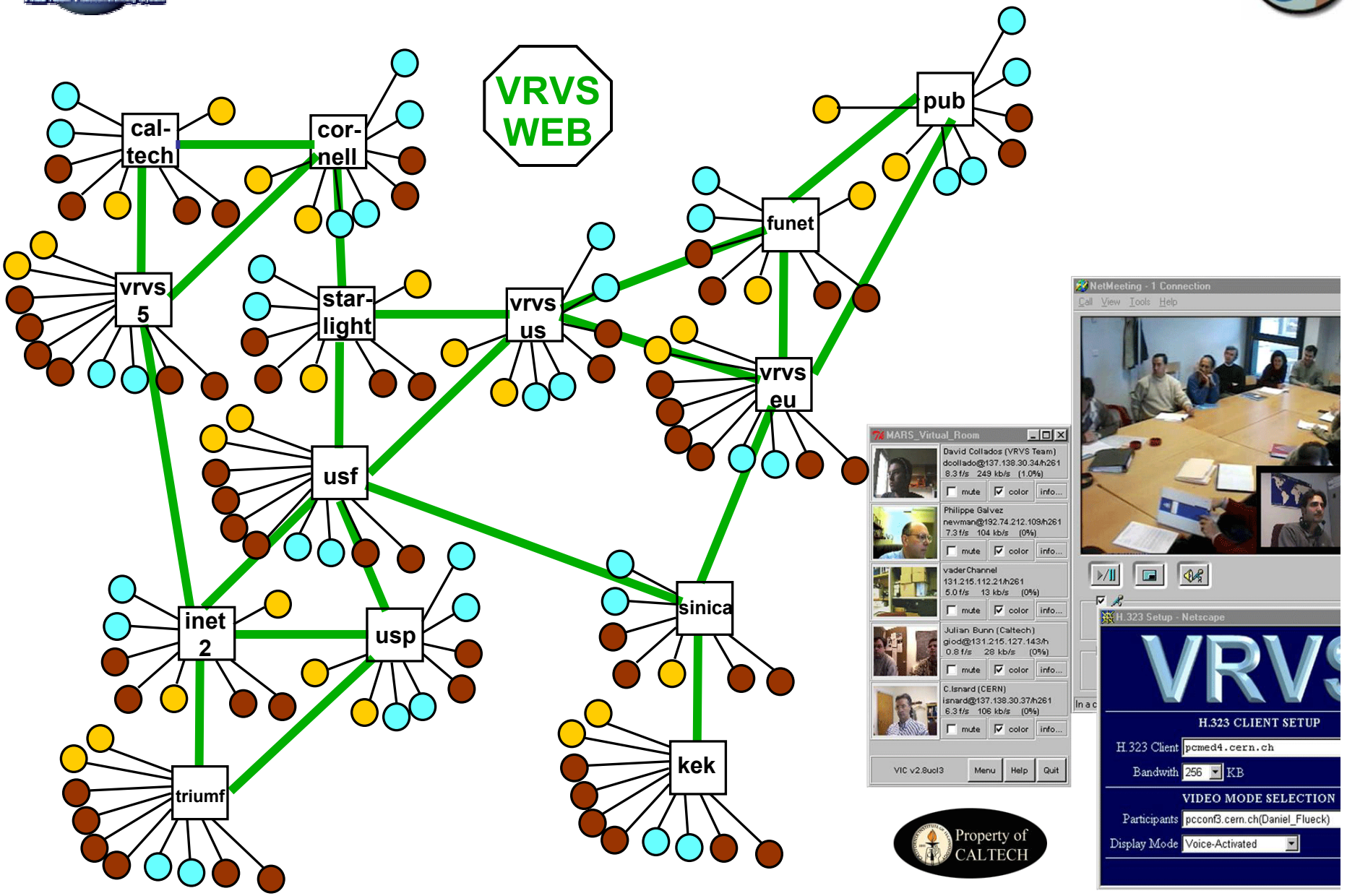
The selection is made according to the **assumed** network links performance.

VRVS WEB

The VRVS Integrated Web Interface is used to schedule and join or leave anytime a meeting.

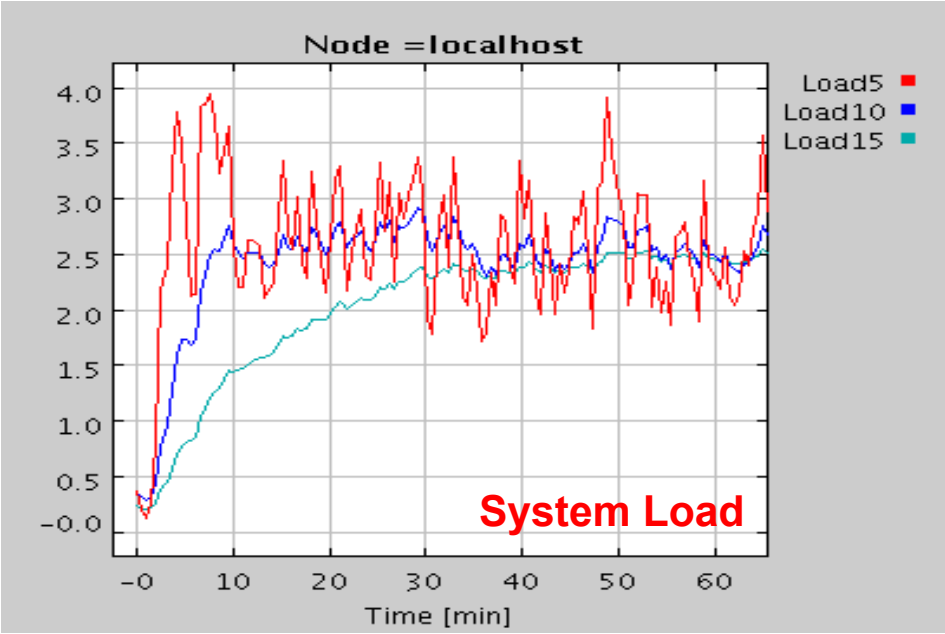
**Virtual Room** Concept:
Create a virtual space were people attending the same conference can exchange real-time information

# Should be highly scalable & reliable



# We need monitoring & controlling

- **Network failure**
  - Less bandwidth
  - Packet loss
  - Delay
  - Jitter
- **Machine failure**
  - Hardware breakdown
  - CPU overload
  - High memory usage

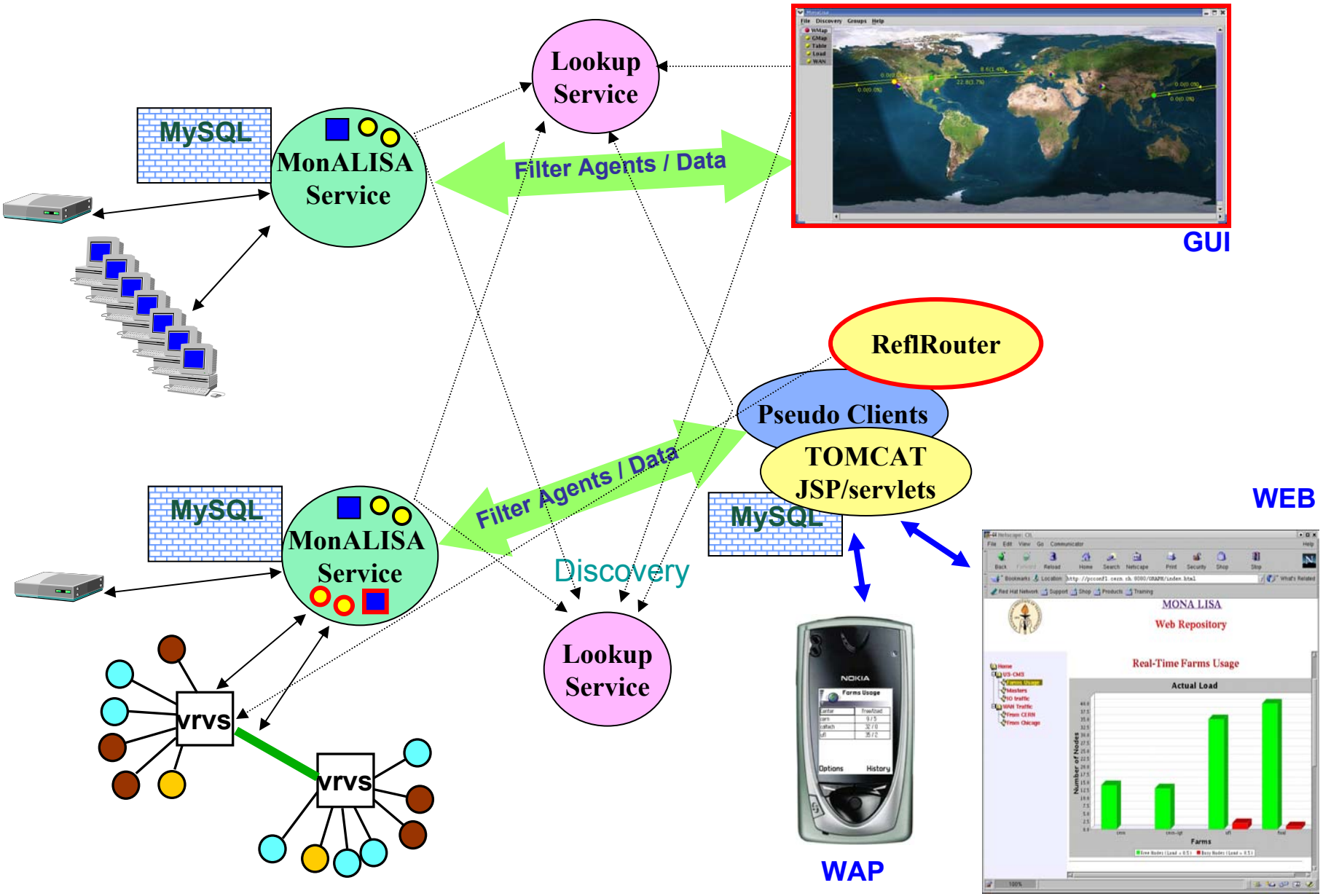



**If a central node (reflector) or a main link fails, the whole system is affected, not only the particular clients using that resource!**


# How? Using MonALISA

- **MONitoring Agents using a Large Integrated Services Architecture**
  - **a true dynamic service which provides the necessary functionality to be used by any other services that require such information (Jini, UDDI - WSDL / SOAP)**
  - **It provides in real time:**
    - **general network information**
    - **CPU / memory / disk information**
    - **historical data and extracted trend information**
    - **listener subscription / notification**
  - **Uses:**
    - **(mobile) agent filters and alarm triggers**
    - **algorithms for decision-support**

# Single service multiple clients



# Querying the Reflector

- **Monitoring modules**
  - SyncVrvsClientsT & SyncVrvsConnT
    - Connect to the reflector on a TCP port
    - Send the command
      - <pass_key>/<command_to_reflector>/<param>
    - Receive results & parse text
    - Build Results:
      - number of audio clients;
      - number of video clients;
      - number of virtual rooms.
      - peer links' quality;
      - lost packages
    - Return the vector of results

# Links quality evaluation

- **monABPing – customized ping module**
  - Quality factors important for videoconferencing
    - delay (RTT), packet loss, jitter
  - Compute RTT by sending UDP packets between monitored reflectors (configurable)
  - Compute jitter as a mediated variance of RTT for last period (configurable)
  - Compute link quality as
    - A + B * RTT + C * loss + D * jitter
    - allow 4 degrees of freedom for easily customization
  - The links to evaluate and parameters are reread each two minutes from a configurable URL

# Filter based data analysis

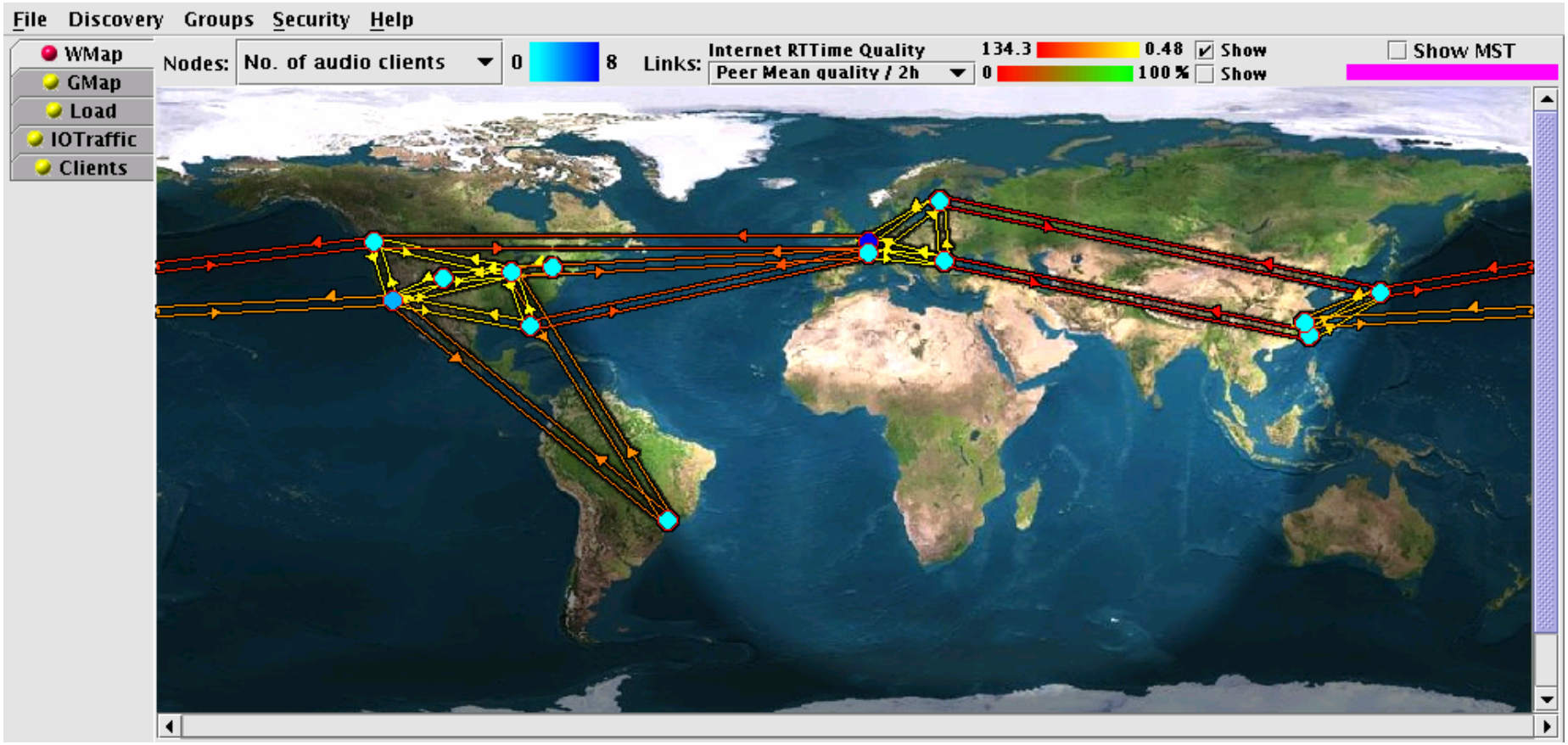
- **Filtering data – TriggerAgent**
  - Register for receiving the data coming from the modules
  - Compute time mediated values for peer links quality
    - Qual2h, Qual12h, Qual24h
  - Build a special type of result, “alarm” if there is no data from the reflectors for 1 minute
  - Inform all registered clients
- **Triggering Actions – vrvsRestartTrigger**
  - Listen for receiving the data from the modules
  - Take action if no interesting data comes …
    - for 1 minute → restart reflector
    - for 2 minutes → send e-mail/sms to a list of administrators

# VRVS controlling client

- **ReflRouter**
  - Receive data from the modules, keeping a timestamp for the results received for each reflector/connection
  - Compute the restrictions for the MST algorithm
    - List of active/inactive reflectors & available links for the MST
    - List of the links that **MUST** be (dis)connected
  - Compute the **Minimum Spanning Tree**
    - Uses the Barůvka's algorithm – O(m log n)
    - A momentum factor is used for links currently in MST to avoid fast switching between connections of similar quality
  - Generate the list of commands (connect/disconnect tunnel) that can/must be sent to each reflector

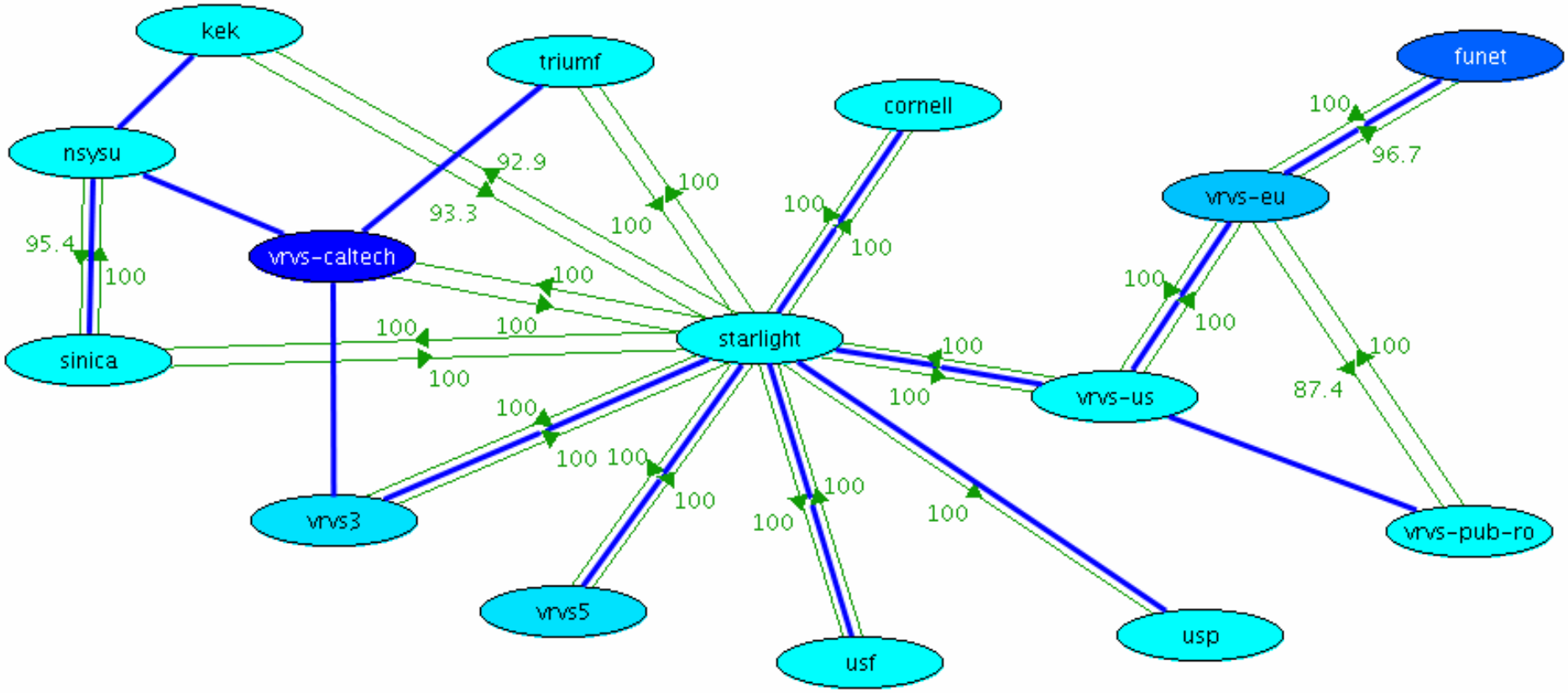
# GUI client

- **Display the current data obtained from the monitoring agents**



# GUI client

- **Display a real-time status of the active tunnels and the best currently available links between reflectors computed by the MST algorithm**



# Future work

- **MonaLisa's VRVS controlling client**
  - **Send the routing change commands to the reflectors when needed**
- **Distributed controlling client**
  - **Will employ a distributed version of the MST algorithm**

# Conclusions

- VRVS is a mature videoconferencing system with great usability, scalability, flexibility, efficiency and robustness.
- MonALISA provides a flexible and reliable monitoring service for higher level services in distributed systems.
- Using the unique capabilities of MonaLisa, filters and trigger agents can be dynamically deployed to any VRVS reflector to provide in real time the required monitoring information to clients and improve the scalability, efficiency and robustness of the entire distributed system.


# Questions?

- **MonaLisa**
  - **http://monalisa.cacr.caltech.edu**

- **VRVS**
  - **http://www.vrvs.org**

**Catalin Cirstoiu**
**catac@cs.pub.ro**