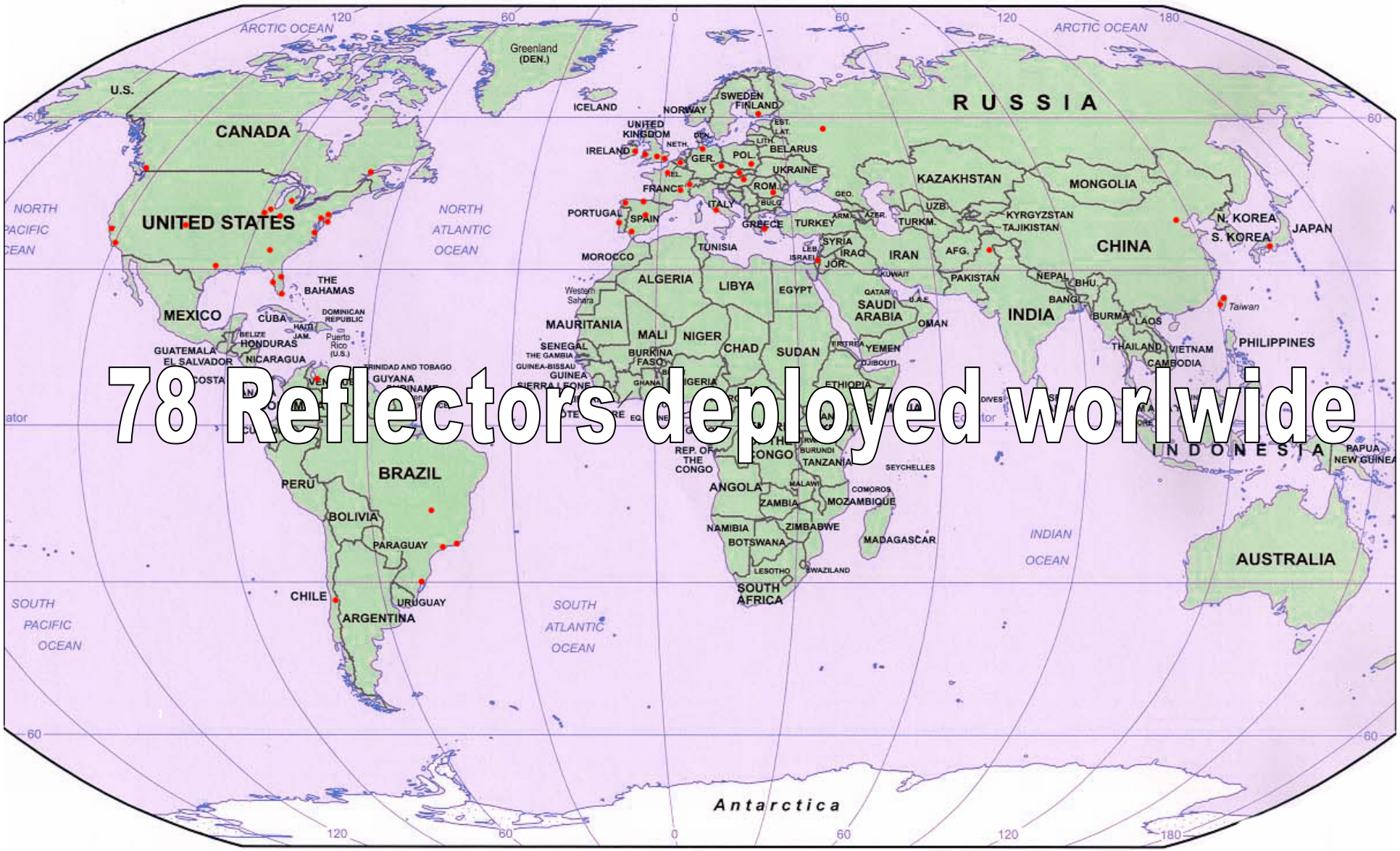
VRVS Research Roadmap (Caltech)

VRVS Deployment and Usage

VRVS Reflectors Deployment



VRVS Reflectors Deployment

78 reflectors Deployment World wide in 27 Different Countries

USA	27
Spain	5
Brazil	5
Switzerland	5
UK	3
France	3
Slovakia	3
Canada	2
Taiwan	2
Greece	2
Portugal	2
Israel	2
Japan	2
Pakistan	2

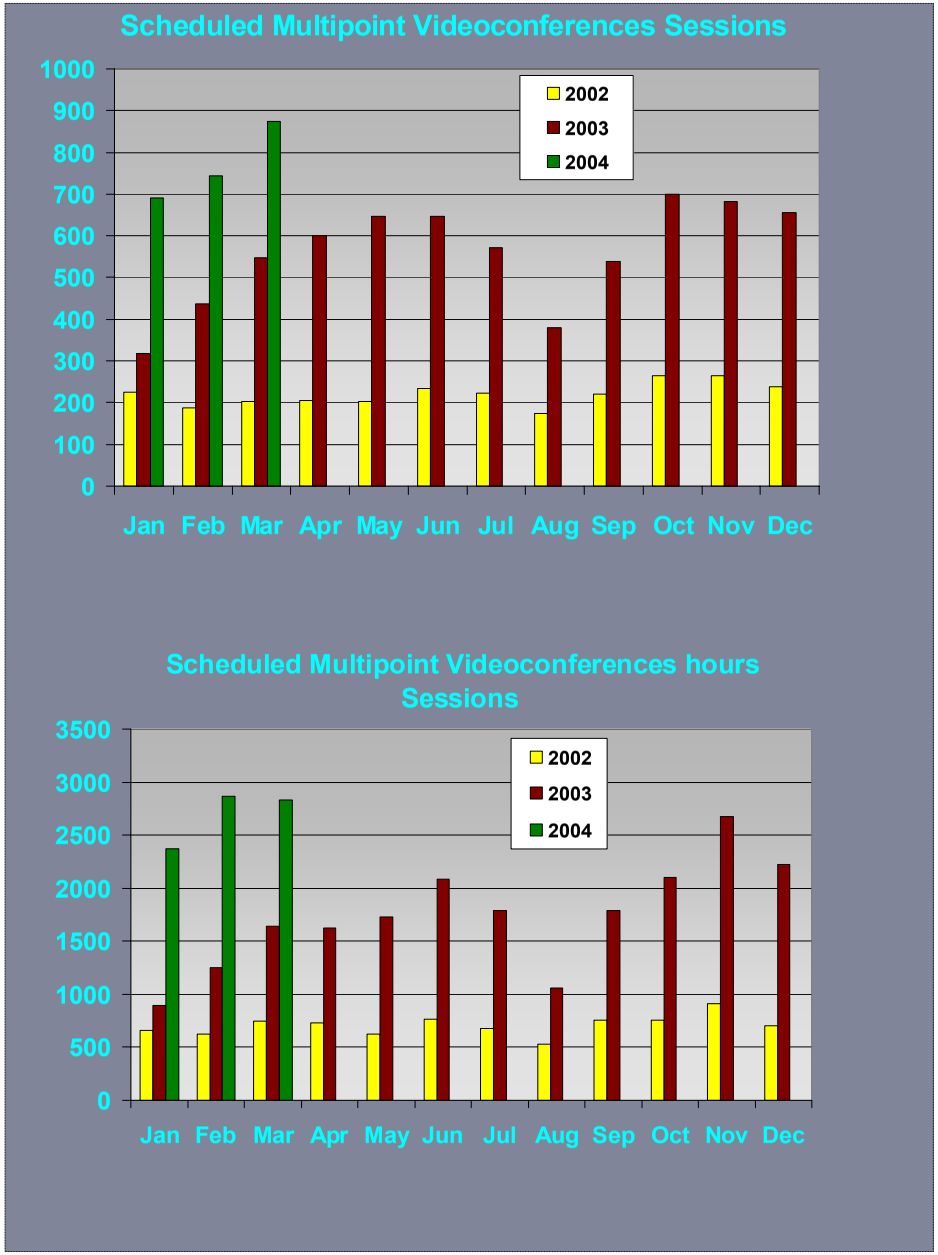
Italy	1
Germany	1
Chile	1
Poland	1
Venezuela	1
Hungary	1
China	1
Ireland	1
Russia	1
Czech Republic	1
Belgium	1
Romania	1
Australia	1
Finland	1

VRVS registered users and current usage

9461 different Users
Registered
from 104 Countries

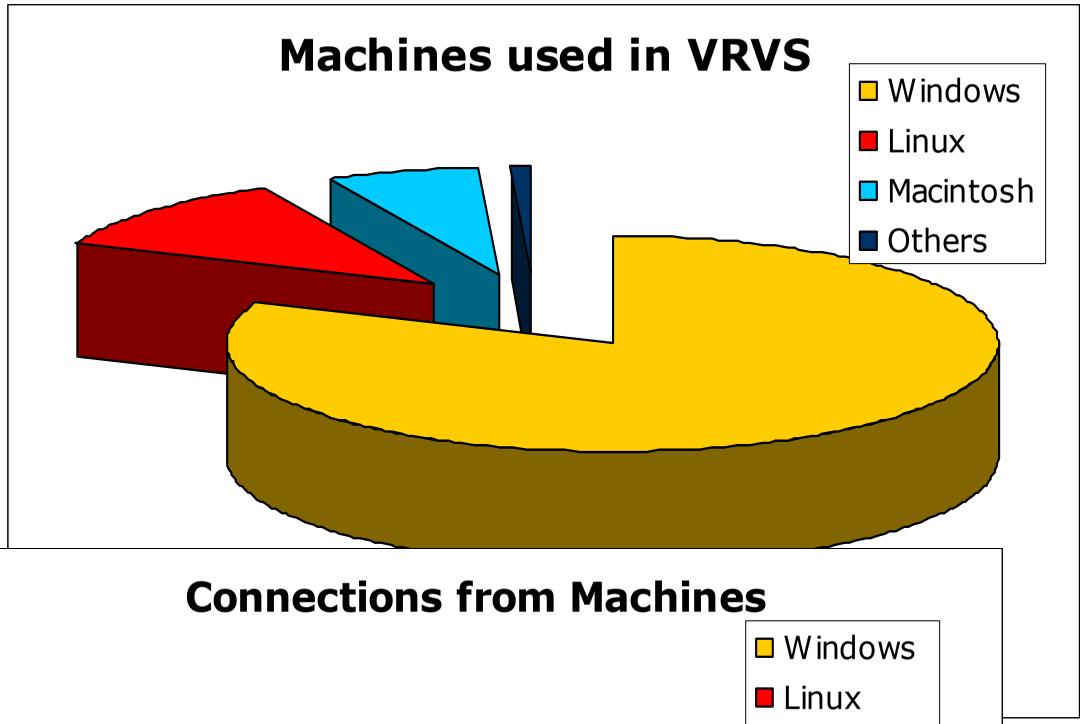
USA	2239
Spain	1304
Italy	638
Switzerland	576
France	571
Brazil	479
Germany	476
UK	398
Japan	168
Canada	167

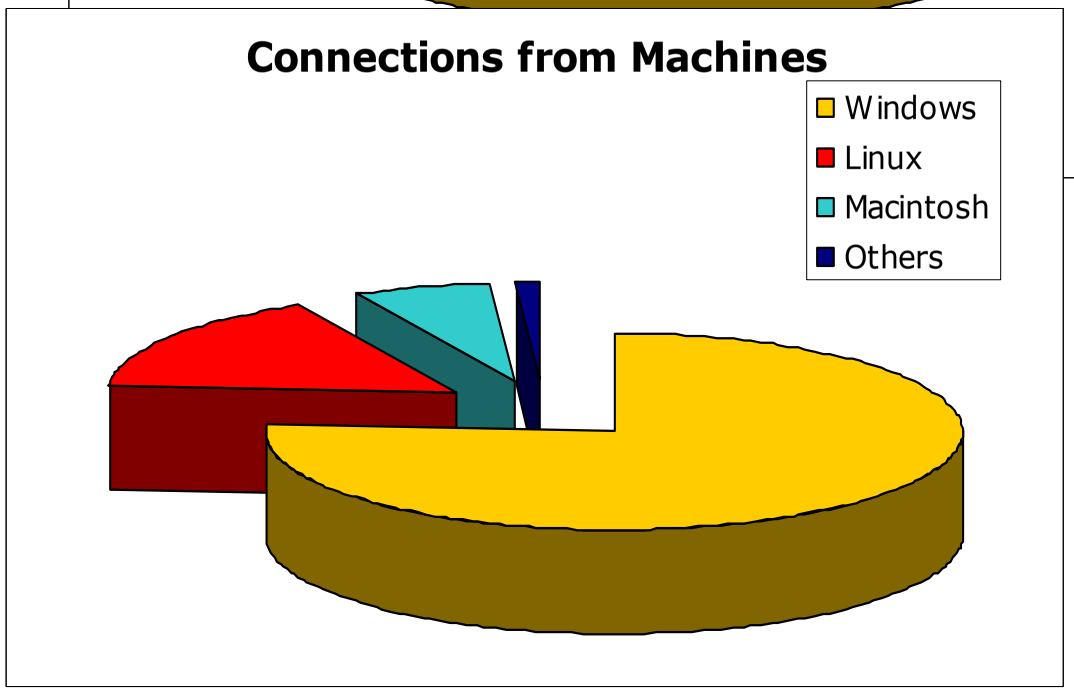
Taiwan, Greece, Argentina, Russia, Slovakia, etc…



Machines and OS

- VRVS support different
- Operating Systems
- according to the need
- and the demand of the
- final users:





- 1st : Windows
- 2nd: Linux
- 3rd: Macintosh
- 4th: Other UNIX

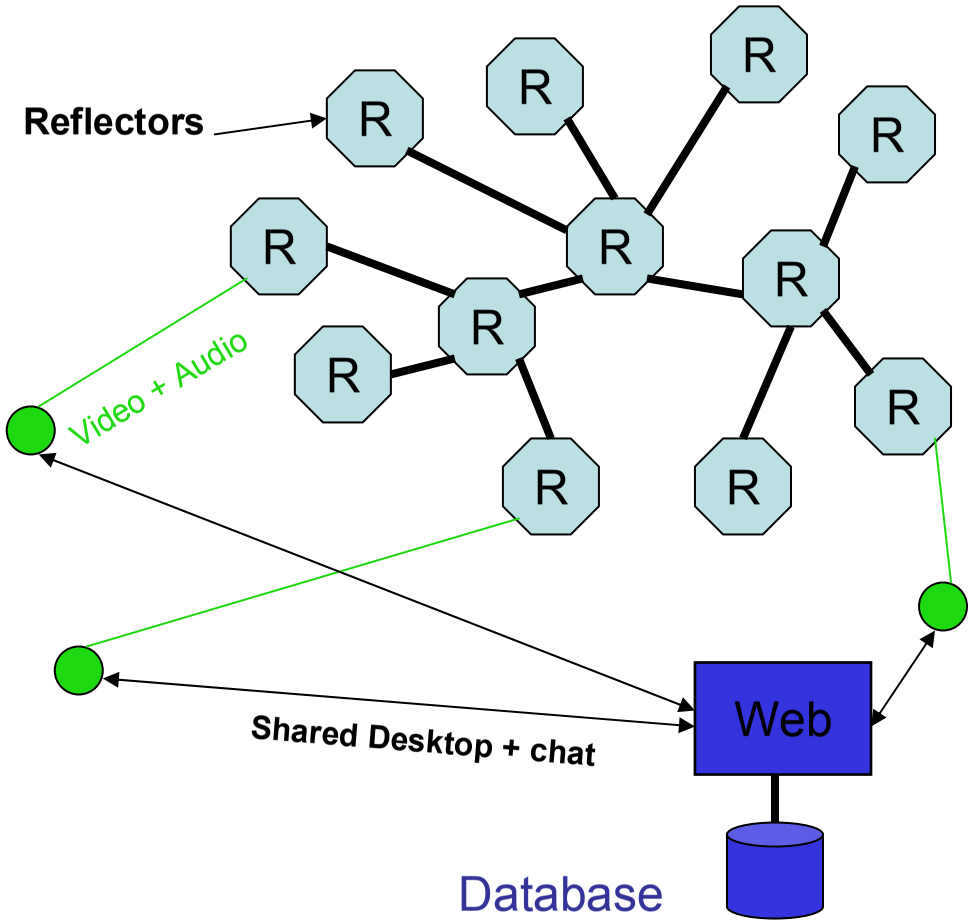
Call Details Record (CDR)

	Number of VRVS Meetings	**Number of Participants**	**Total number of Minutes of video/audio connection**
NOV 2003	692	2951	144 Days, 17h, 14mn *(3473 hours, 14mn)*
DEC 2003	656	2734	129 Days, 18h, 57mn *(3114 hours, 57mn)*
JAN 2004	687	2980	189 Days, 4h, 23mn *(4540 hours, 23mn)*
FEB 2004	742	2797	175 Days, 8h, 48mn *(4208 hours, 49mn)*
MAR 2004	873	3461	195 Days, 23 h, 17mn *(4703 hours, 17mn)*

VRVS R&D

Caltech

VRVS v3.0 (Feb 03), v3.1 (May 03)



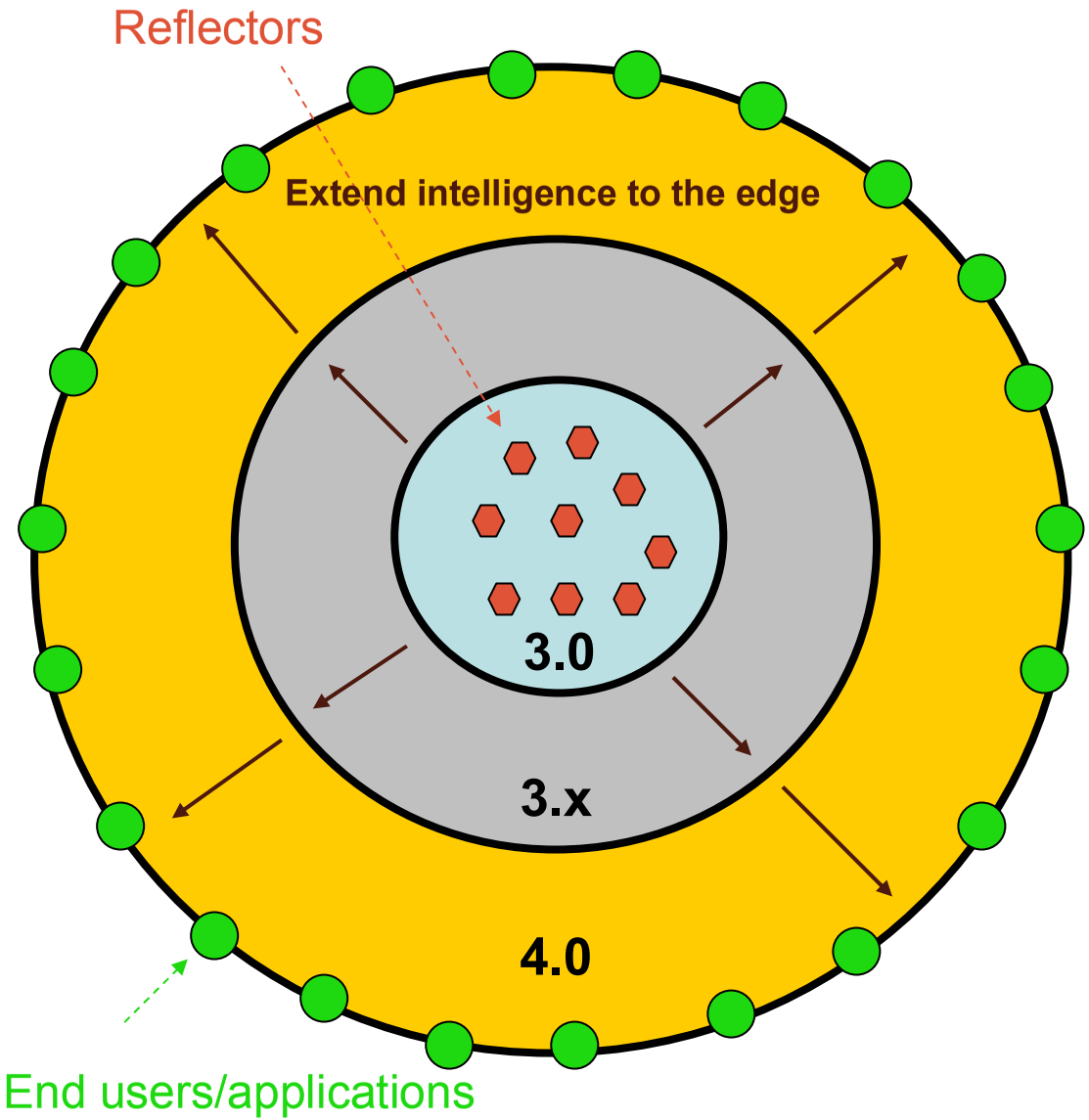
Principal characteristics

- Centralized database to manage users/meetings
- Tunneling between reflectors using one port independently of number of parallel conferences or users connected
- Shared desktop and chat connection via the web server
- VRVS Reflectors are manually configured and started by an operator
- The VRVS network infrastructure configuration is static

Principal limitations

- Any network break between reflectors needs manual rerouting configuration
- Centralized Web server and database is a potential point of failure
- Chat and shared desktop network traffic via Web server
- No control/monitoring between reflectors and between reflectors and end users.

VRVS Main Technical Trend Evolution



V3.(0,1):

VRVS core infrastructure is statically and manually configured and operated

V3.(2,x):

VRVS core infrastructure is automatically configured and monitored. The core software is self dependant and can take self decision to improve performance/quality without manual intervention

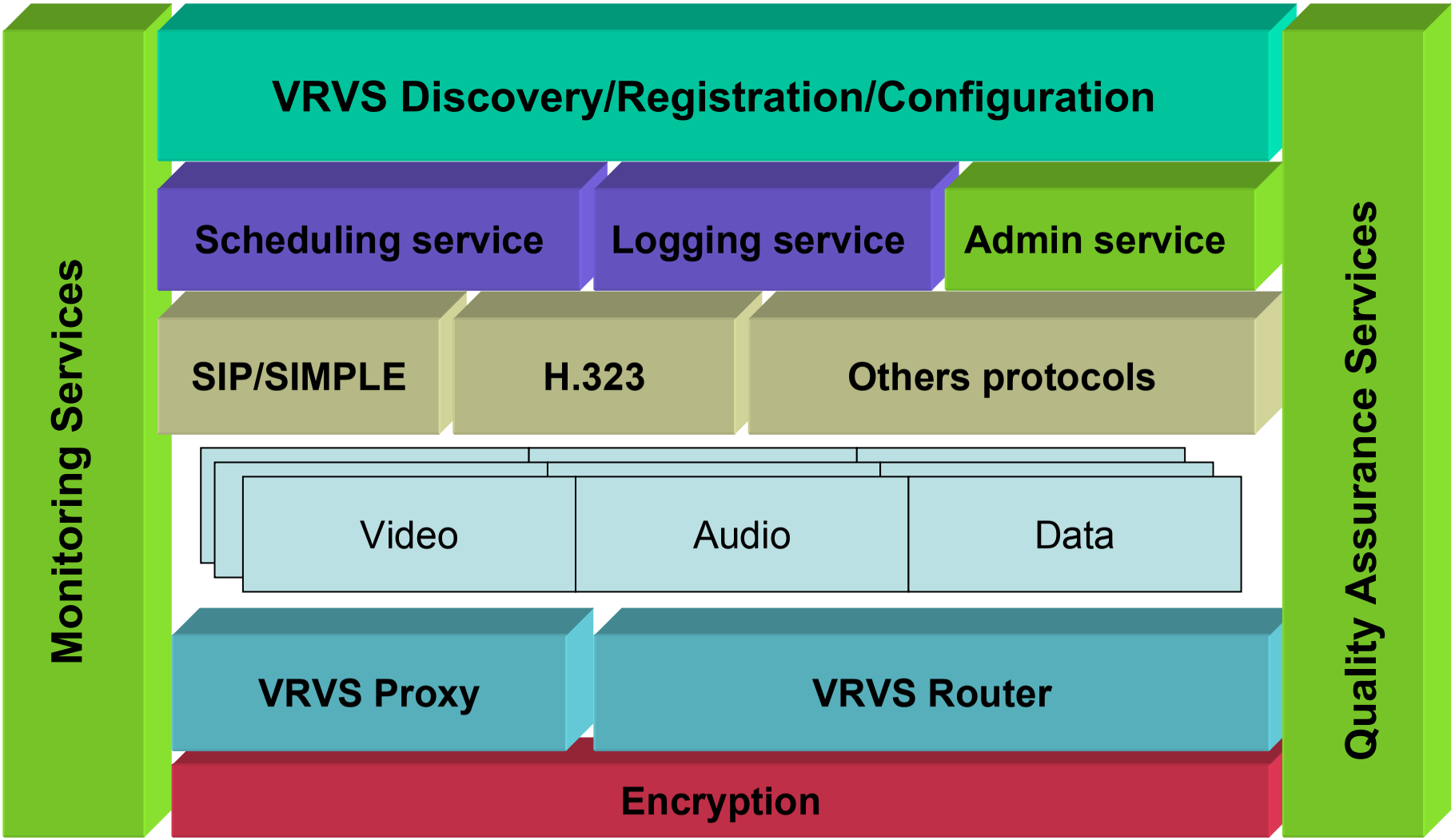
V4.0 and beyond:

- **This is a Globally Distributed Self Managed End2End Real-time Infrastructure. It provides the best quality/performance possible**
- **Extend the core intelligence to the edge.**
- **Have a full End2End control and monitoring**
- **The self managed infrastructure has a full knowledge of all the critical/sensitive parameters *(all network layers, hardware and software at the end nodes, resources allocated and available,..)* in order to take adequate decisions *(alarms, automatic rerouting of traffic, disconnection, remove/add services,..)***
- **Administrator is fully aware of the operational status via constant feedback (via UI, email, phone,..) from the self managed core software**

VRVS Reflector functionalities

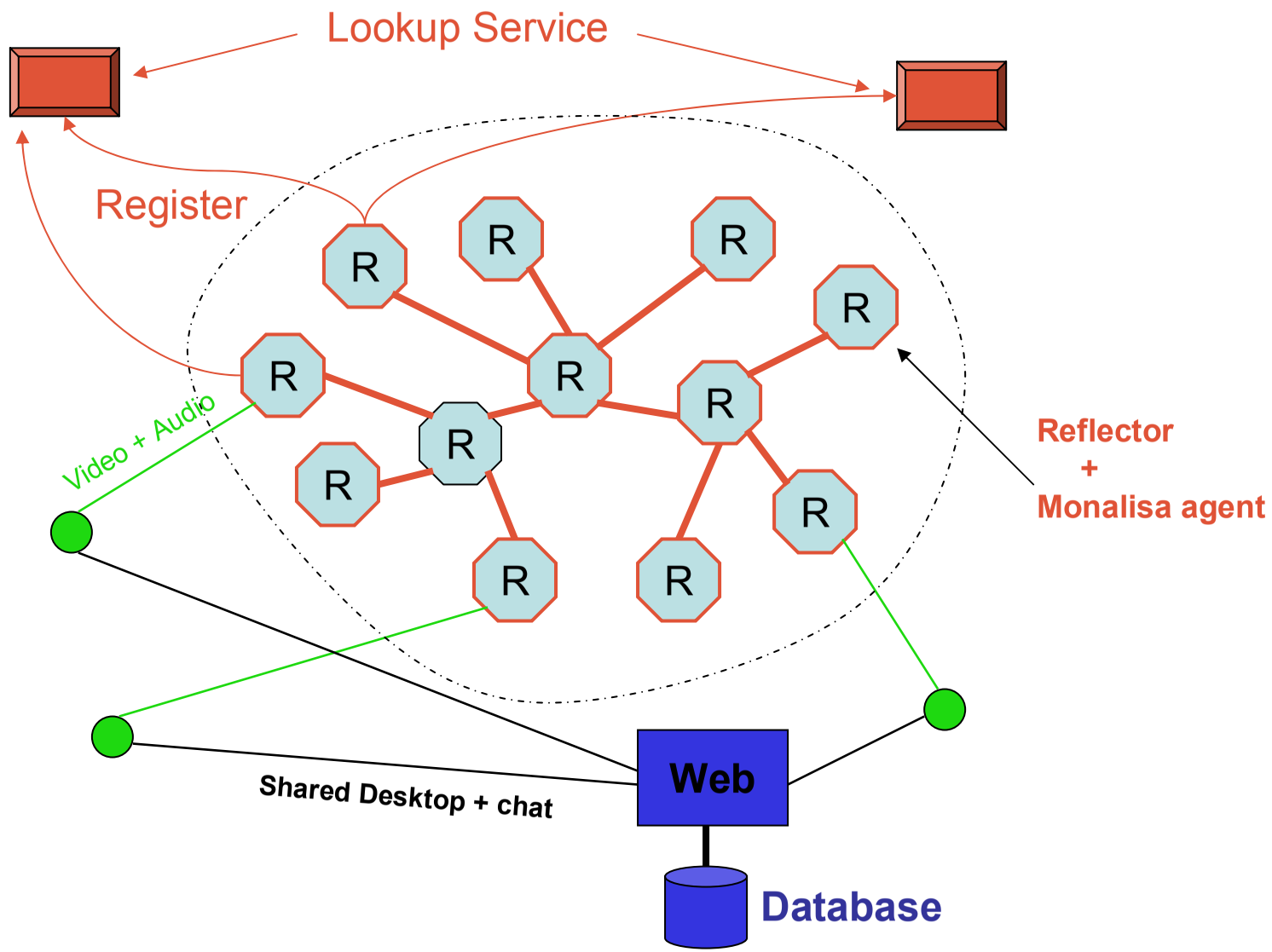
- Dynamic registration to high level directory services
- Automatic re-activation of components and services
- Automatic and secure code update
- Continuous monitoring of network quality (packet loss, jitter, latency) between its peers and its possible peers
- Automatic rerouting to obtain the best performance/quality
- Automatic Alarm notifications when monitored parameters (system or network) go beyond a preset threshold
- Dynamically provides services (video, audio, data,..) that matches the current resources/capabilities to the end users/applications
- Provides access to real-time and historical data

VRVS Reflector Architecture



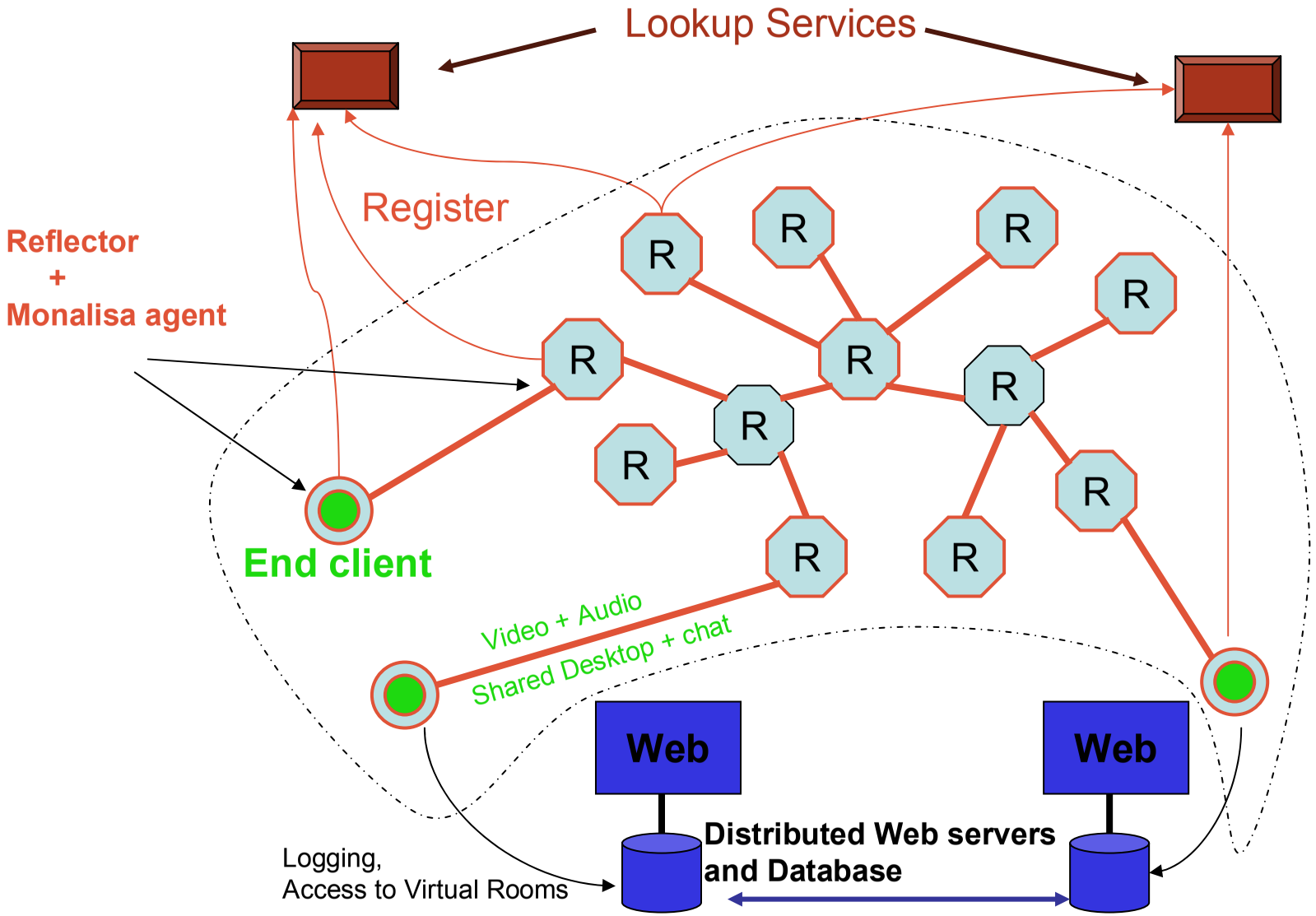
VRVS v3.2 (Dec 03), v3.x: *VRVS Services evolution*

Globally Distributed Self Managed Real-time Infrastructure



VRVS v4.0 (Oct 04) *VRVS Services evolution:*

Globally Distributed Self Managed End2End Real-time Infrastructure



End users/applications functionalities (1/2)

- Dynamic registration to high level directory services

- Automatic detection of the system parameters (CPU, Memory,..), hardware components (Audio card, video card, …), services capabilities (video, audio, …), network environment and capabilities (wireless environment, DSL, available bandwidth, …)

- *Automatic re-activation of components and services*

- Automatic code update

- Continuous monitoring of network quality (packet loss, jitter, latency) to the attached reflector and possible others reflectors

- Automatic rerouting to obtain the best performance/quality *(The communication between an end node could be rerouted transparently and automatically to an other reflector for performance optimization)*

End users/applications functionalities (2/2)

- Automatic Alarm notifications when monitored parameters (system or network) go beyond a preset threshold.

As example: if Desktop CPU is too high, the system will automatically try to perform the following:

  - ✓ *reduce services (video/audio/data/..) running in the machine and inform user of the change*
  - ✓ *or if no improvement, inform the user of the problem and from where it comes from (if possible) and then propose a solution (ultimately reset the system)*
  - ✓ *Keep inform the general system administrator*

- Dynamically gets services (video, audio, data,..) that matches the current resources/capabilities to end users/applications

- Provides access to real-time and historical data

Development tasks list (1/3)

• **Task 1:** Implement an automatic alarm notifications mechanism. An alarm is triggered when one of several monitored parameters (system or network) go beyond a preset threshold

• **Task 2:** Add new parameters to be accessed/monitored by the VRVS/Monalisa. Ex: hosts connected, participants information, ongoing meetings, type of clients, ..

• **Task 3:** Modify the VRVS/Monalisa GUI to display/access all the added/new information. *Ex: Select an ongoing meeting and display via the GUI only the reflectors involved in the meeting, the quality of the connections, and the hosts/participants connected*

• **Task 4:** Add the possibility to have administrative actions via the VRVS/Monalisa GUI. *Ex: Remove/add host, disable/enable audio/video clients, change VRVS network topology,…*

• **Task 5:** Create a VRVS java based reflector proxy to run on the desktop client.

• **Task 6:** Create a Monalisa java based agent to run on the desktop and to be attached to the rest of the VRVS topology

Development tasks list (2/3)

• **Task 7:** Develop a java-based Hardware/system detection software to run on the desktop client

• **Task 8:** Modify the current VRVS Java applets and Window associated to be a stand alone java based application to run on the desktop client and launch via the web browser. The GUI needs to be re-think

• **Task 9:** Integrate the Java based software components developed in Tasks 5, 6, 7 and 8 into a main Java based software client, to become the ***VRVS Client***. It will allow the users/applications to connect to the system and to be part of the already self-managed distributed infrastructure.

• **Task 10:** Study, design and implement a distributed database architecture

• **Task 11:** Study, design and implement a load balancing and distributed web servers architecture

• **Task 12:** Integrate SIP protocol within the reflectors which will then become a SIP proxy. It will use the efficient and very scalable VRVS infrastructure for global and scalable SIP services.

•

Development tasks list (3/3)

- **Task 13:** Provide to the research community advanced Mbone applications (vic, rat) with new features, enhanced functionalities, etc..

- **Task 14:** Study possibility to port the entire reflector software to Java

- **Task 15:** Design and implement encryption functionality (1) Between the reflectors, (2) Between the end client and the reflector

- **Task 16:** Study integration of Global Authentication mechanism (PAPI, Shibboleth, Globus Toolkit,..) and perform initial implementation

- **Task 17:** Design and Port the H.323 VRVS Agent to Java (http://www.alphaworks.ibm.com/tech/j323engine)

- **Task 18:** Build a HDTV codec client to be compatible with the VRVS technology. Create the first multipoint HDTV videoconference

- **Task 19:** Build an advanced "collaborative environments"