

SERVAL: an Internet software VLAN switch developed in Erlang



Erlang User Conference 2004

Juan José Sánchez Penas (LFCIA, Igalia)

Alejandro García Castro (Igalia)

Fco. Javier Morán Rúa (Igalia)

Igalia (www.igalia.com)

- ◆ Free Software company created in **Corunha** (north west of Spain) in 2001. Core team of 20 Software Engineers.
- ◆ Committed to the development of free software.
- ◆ Fisterra Open Source ERP: www.fisterra.org (GTK, C, GNOME and Corba technologies), with GPL license.
- ◆ Closely related to University of Corunha (www.udc.es):
 - ◆ Founders and members of Igalia are software engineers from UDC.
 - ◆ Erlang lectures since 1999.
 - ◆ **VoDKA** project (Video on Demand Server) was developed there by LFCIA group.
 - ◆ **Armistice** project was developed there by LFCIA group, Inditex and Alfa21.

Project motivation and goals

- ◆ Main goal: *easy* creation of VLANs between computers, no matter their location or the connection they use to access the network (Internet).
- ◆ Useful for:
 - ◆ Virtual corporate LANs (e.g. easy to manage remote VLANs configuration).
 - ◆ Sharing resources (e.g. file sharing, SMB, Rendezvous).
 - ◆ On-line games (e.g. using machine and software discovery protocols).
- ◆ R&D Project **SERVAl**: development of a system for emulating VLANs -- **Free Software with GPL license.**

Available solutions (I): VPNs

- ◆ Virtual connections between remote LANs communicating over possible untrusted networks
- ◆ Disadvantages:
 - ◆ Local area protocols cannot be used (cannot transmit non-routed traffic between the networks).
 - ◆ A router is needed.
- ◆ An example of this software is **FreesWan**, which implements **IPSec** (an standard protocol for encrypting IP traffic between two networks connected to IPSec gateways)
- ◆ **openvpn** combined with GNU/Linux **bridging**

Available solutions (II): VLANs

- ◆ Connection of LANs that are physically separated, enabling non-routed traffic between networks.
- ◆ Implemented using the IEEE 802.1Q standard.
- ◆ Layer two traffic with VLAN information to define Virtual LANs using sets of ports of different switches.
- ◆ Disadvantages:
 - ◆ Control on the physical switches required.
 - ◆ All the intermediate machines should support VLANs.
 - ◆ Dangerous reconfiguration and a trained team needed.

SERVVAL architecture requirements

- ◆ Client/server architecture.
- ◆ Linux and Windows client integration.
- ◆ Low latency.
- ◆ Server scalability.
- ◆ Server fault tolerance.
- ◆ Security: authorization, authentication, encryption.
- ◆ Heterogeneous protocol encapsulation.

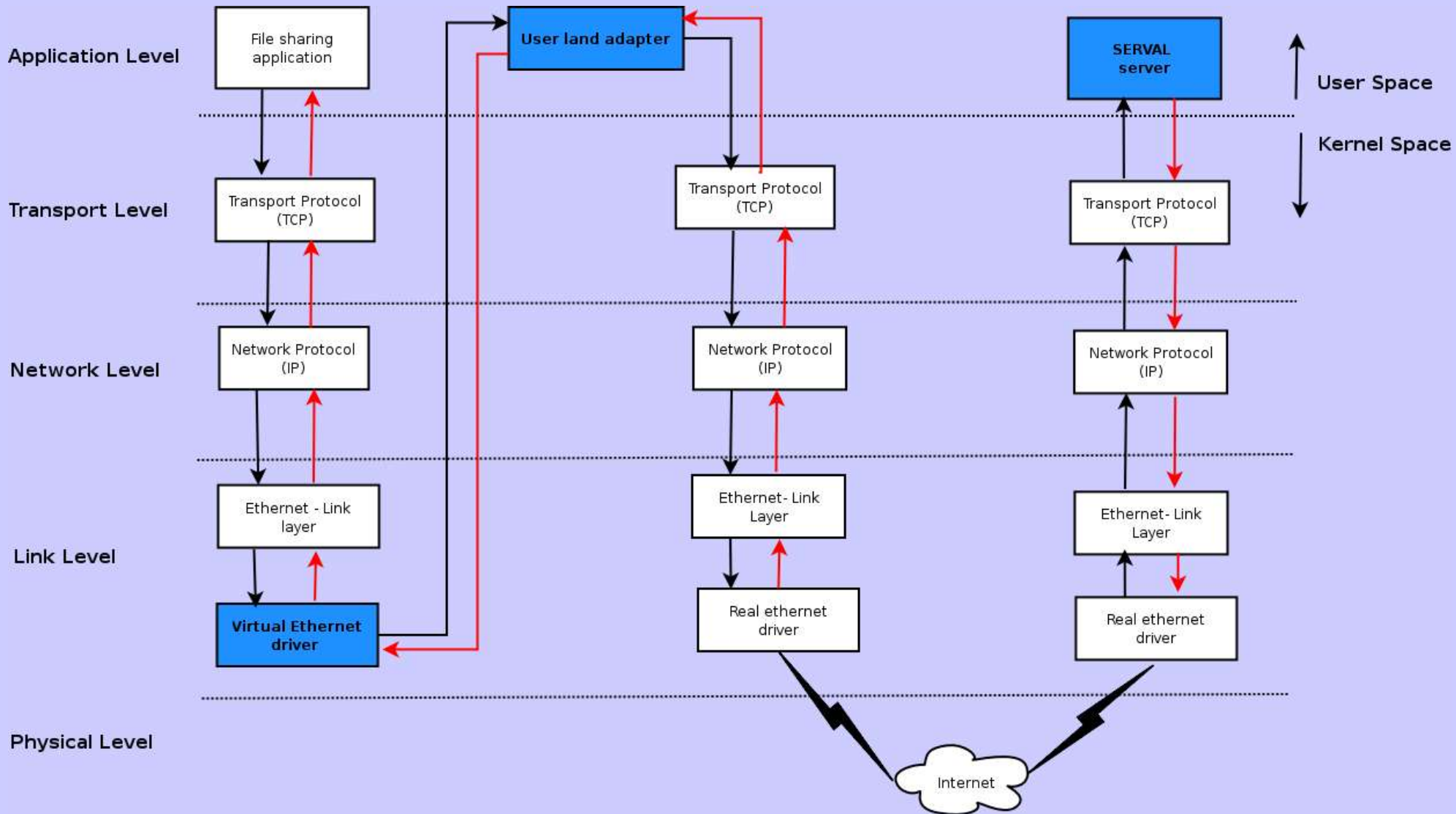
Architecture (I): the server side

- ◆ A system that can emulate VLANs behaviour using a scalable, distributed, TCP/IP **server** that acts as a software switch.
 - ◆ Program running as a daemon in an Internet host, listening for connections.
 - ◆ Able to group users in sets that emulate VLANs. Mediator between clients.
 - ◆ In order to send a message to other client: the message is sent to the server, which checks first if the receiver is in the same set.
- ◆ Implemented in Erlang/OTP

Architecture (II): the client side

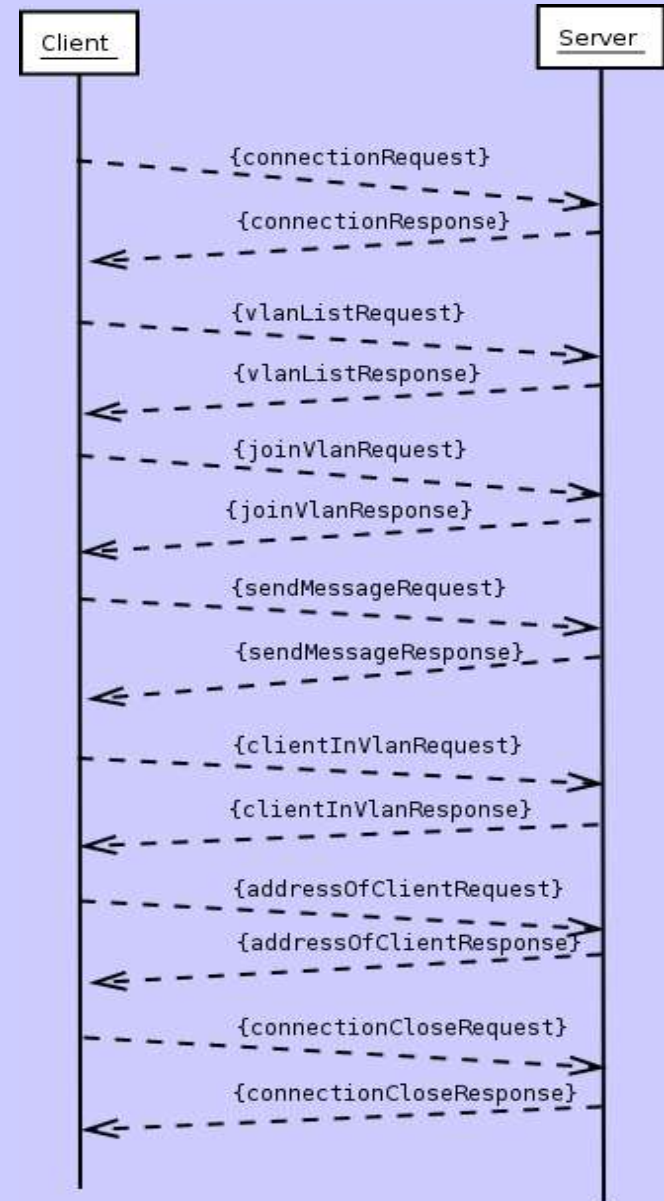
- ◆ **Clients** run programs that simulate network cards connected to the software switch. Two parts:
 - ◆ **Virtual Ethernet Driver:** Ethernet driver which implements a virtual network card.
 - ◆ **User-land adapter:** receives Ethernet frames from the Virtual Ethernet Driver and maps them to SERVAL messages. It has also an interface for receiving requests from the configuration module (e.g. enter or leaving VLANs).
- ◆ Implemented in Erlang/OTP and C

Architecture (III): client/server

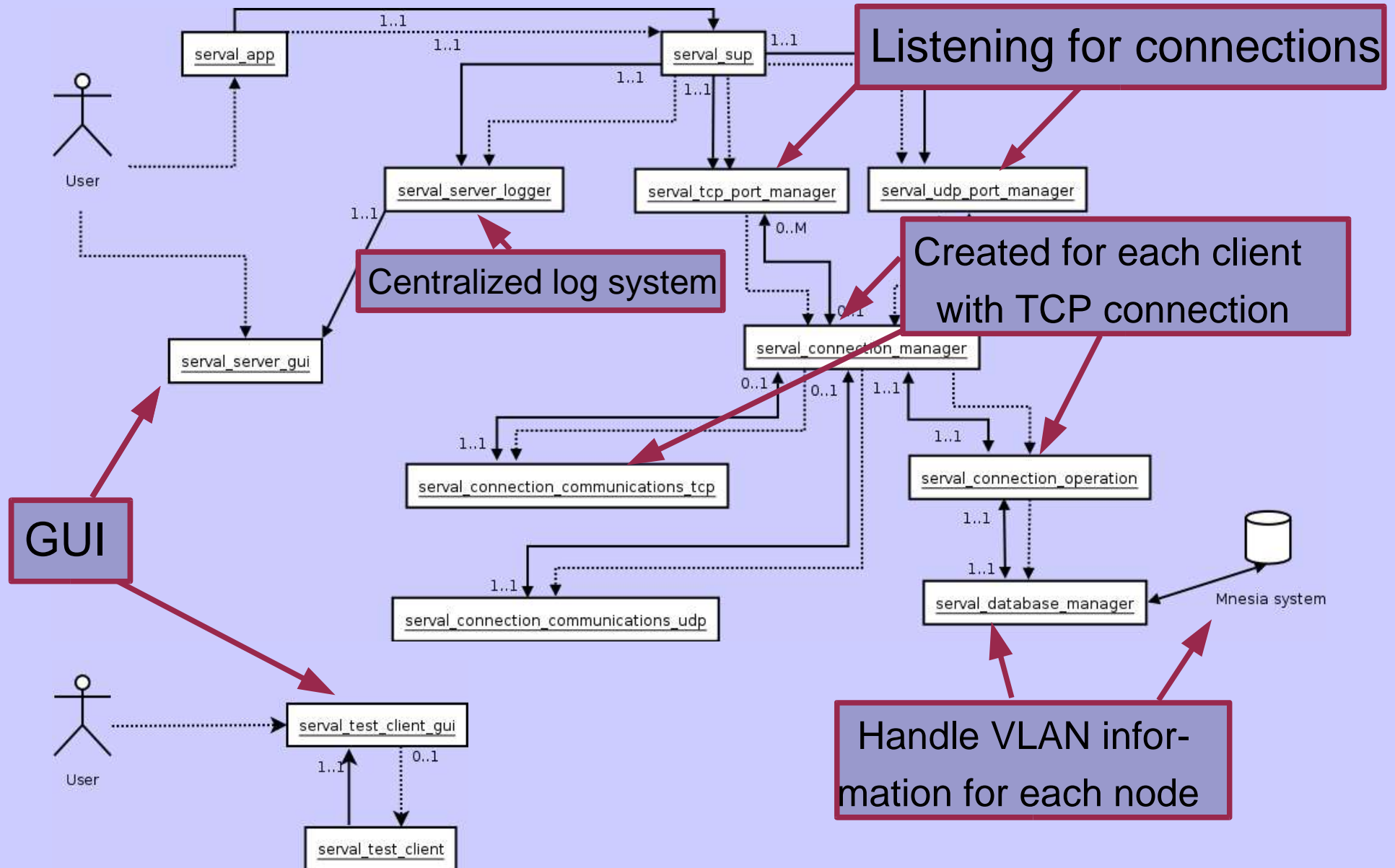


SERVAL internal protocol

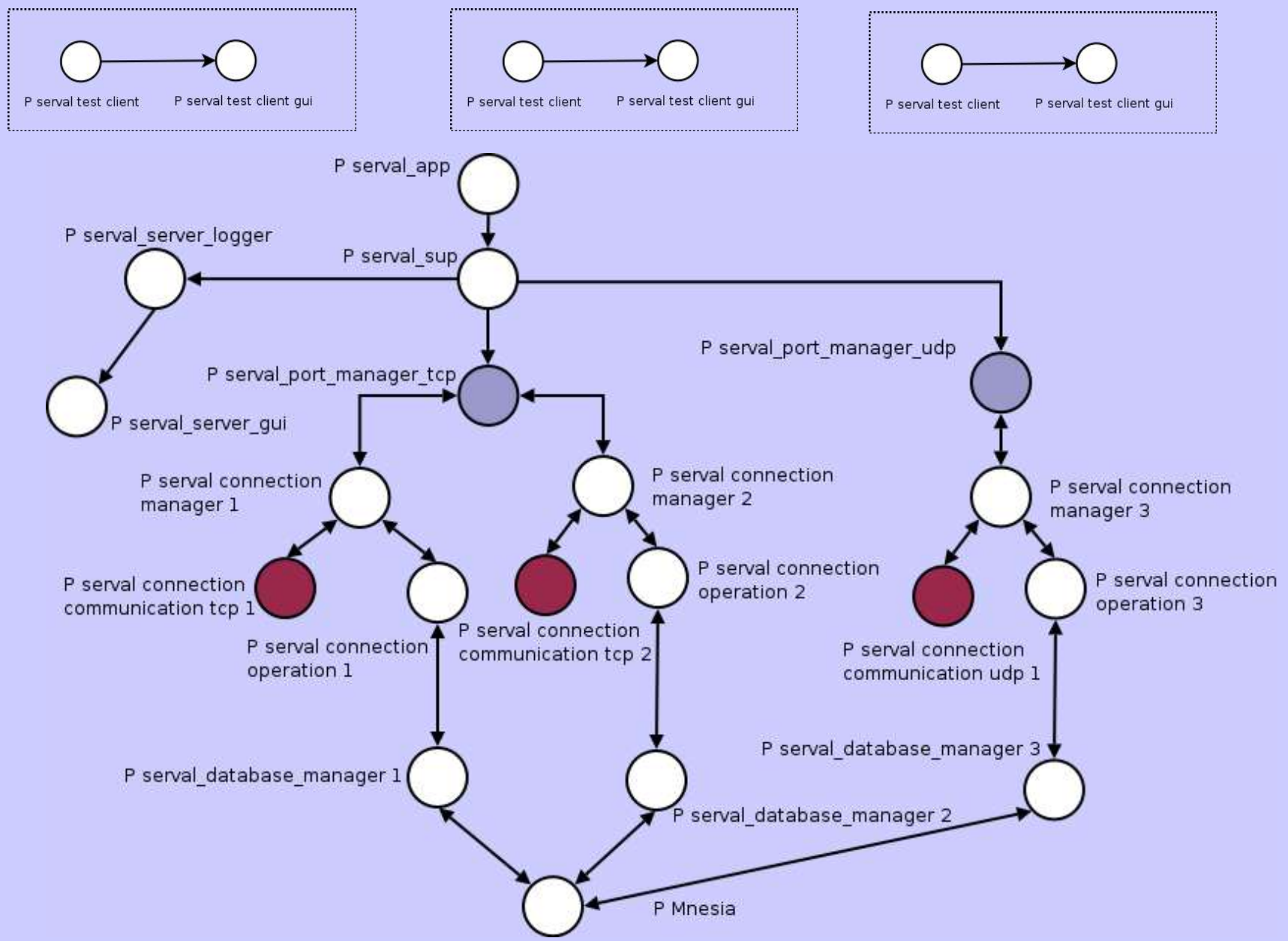
- ◆ ASN.1:
 - ◆ Definition, transmission and encapsulation.
 - ◆ Independent from language.
 - ◆ Supported by Erlang/OTP.
- ◆ Two kinds of messages:
 - ◆ Data sending
 - ◆ Users and VLANs management



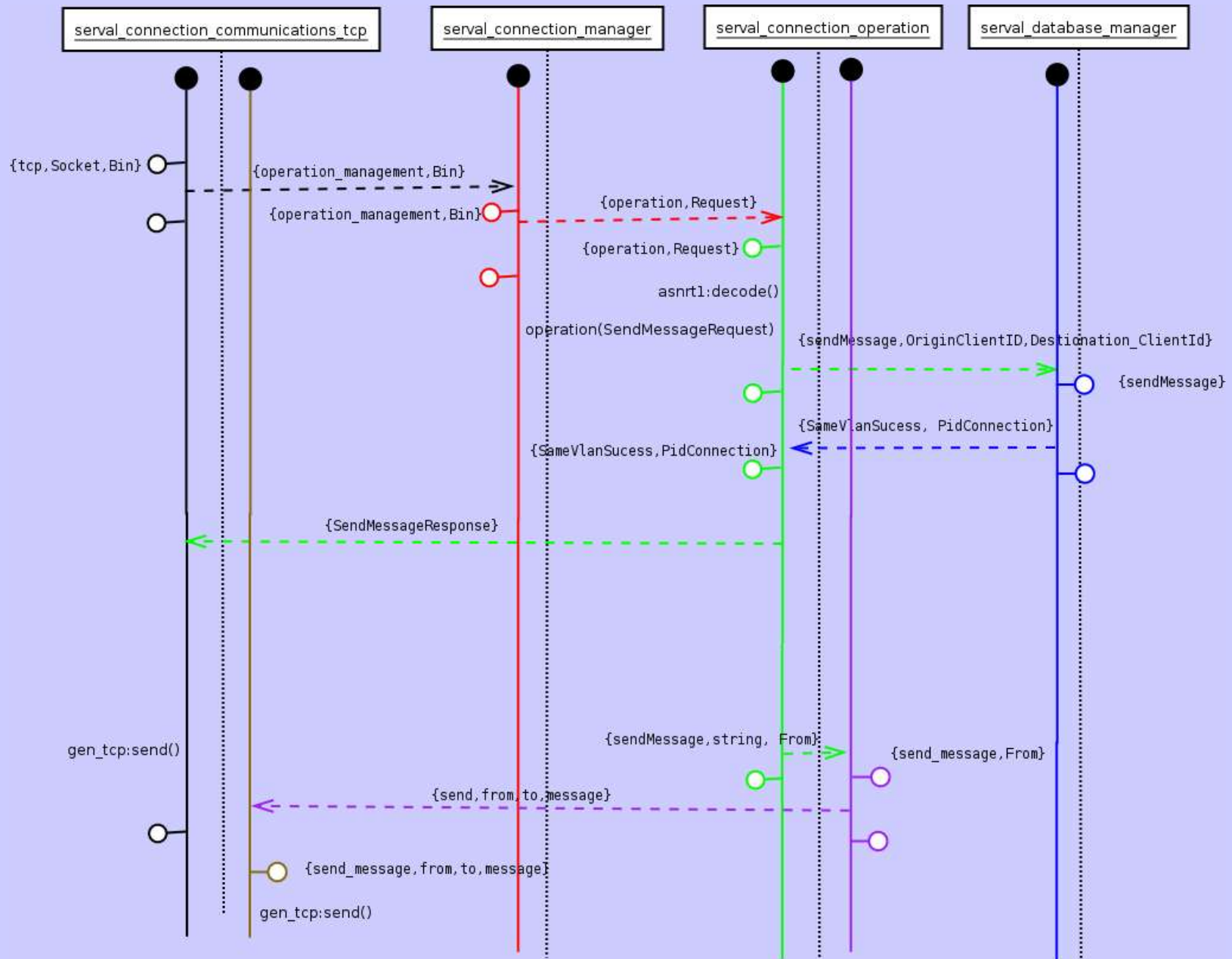
Process classes: server node and client



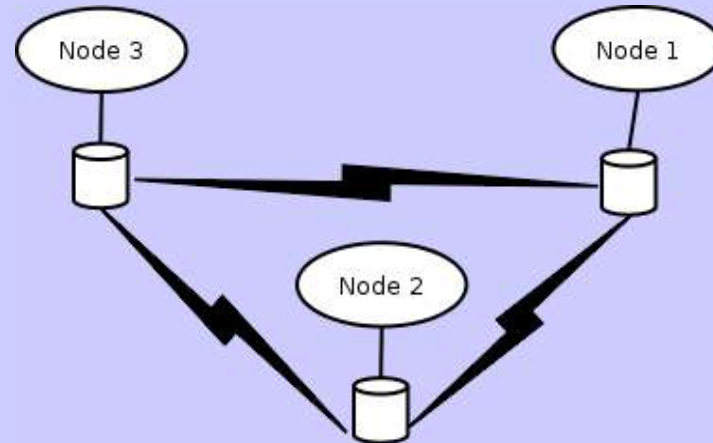
Processes: server node and clients



Message exchange between two clients in the same VLAN



Distributed SERVVAL server: Mnesia



- ◆ To build an scalable, concurrent and fault tolerant system.
- ◆ Clients can connect to any of the servers. They receive a list with the answer to the connection message. Connection mechanism in the client.
- ◆ Clients see clients in the same VLAN connected to any node.
- ◆ Data is shared using Mnesia.

Client screenshot: management

Host: localhost Port: 4567 ID: client2

Vlans List:

- vlan1
- vlan2

Clients for vlan: vlan1

- client2
- client1

Client address:

Connected

- bacteria
- bacteria
- bacteria

Servers List:

- bacteria
- bacteria
- bacteria

Joined Vlans List:

- vlan2
- vlan1

Message:

New Vlan:

```

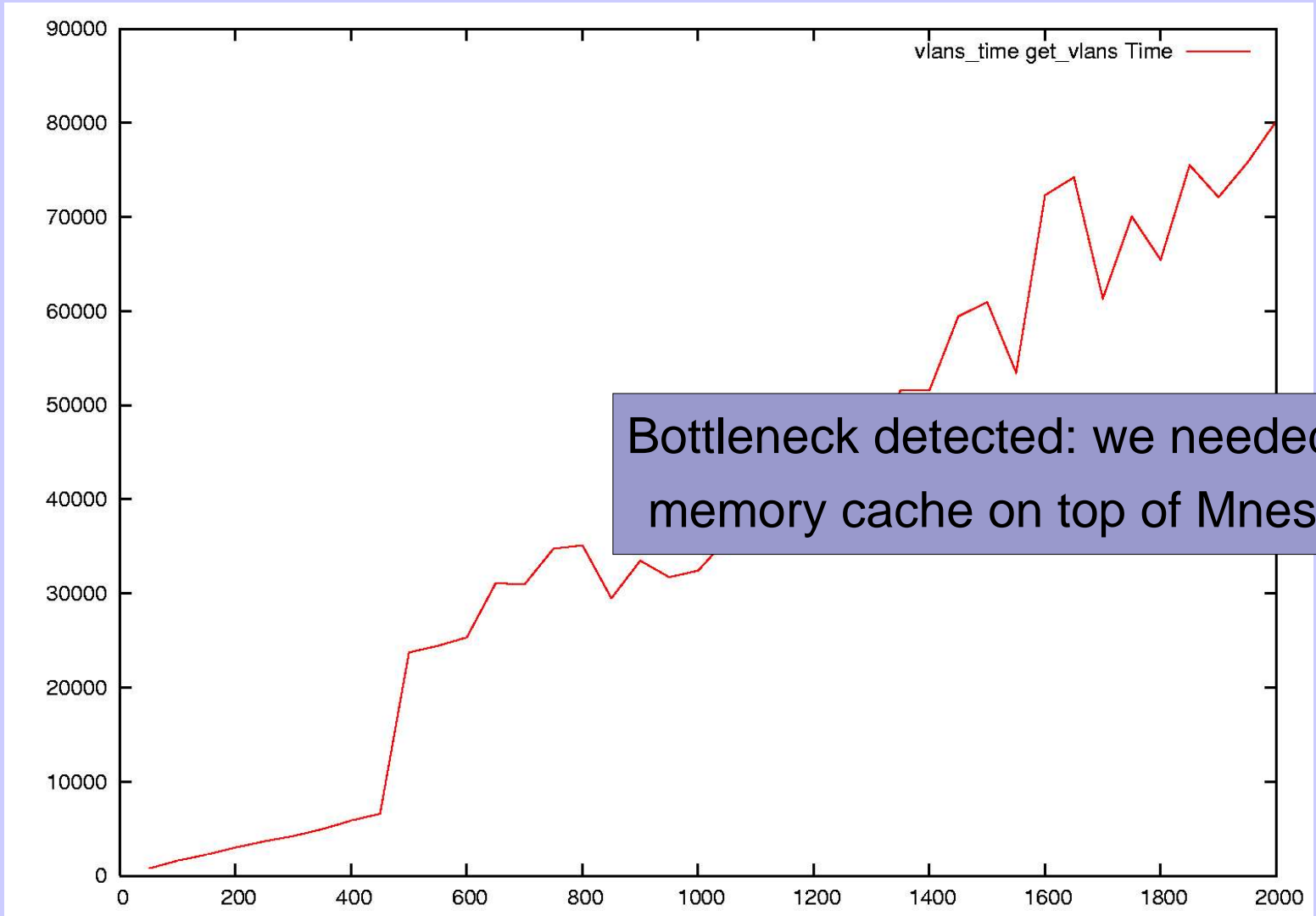
Info: Connected to localhost:4567\n
Info: Sending Identification \n
Testing the send message {'ServalMessage',\n                1,\n                {connectionRequest,\n                {'Connectio
Testing the send message {'ServalMessage',\n                2,\n                {vlansListRequest,{'VlansListRequest','*'}},\n
Message received in client {'ServalMessage',\n                1,\n                {connectionResponse,\n                {'Con
Message received in client {'ServalMessage',\n                2,\n                {vlansListResponse,\n                {'Vlans
Testing the send message {'ServalMessage',\n                3,\n                {joinVlanRequest,{'JoinVlanRequest','vlan2'}},\n
    
```

Performance analysis (I)

- ◆ Important values are:
 - ◆ Operation latency: *the time it takes an operation to be completed, since it is requested to the system until it is finished*. Especially important the latency in the data messages.
 - ◆ Throughput: number of operations that can be completed in a period of time.
- ◆ Method for testing performance:
 - ◆ `serval_gnuplot` module, which interacts with gnuplot.
 - ◆ Tests are performed both in client and server.

Performance analysis (II): an example

Number of VLANS

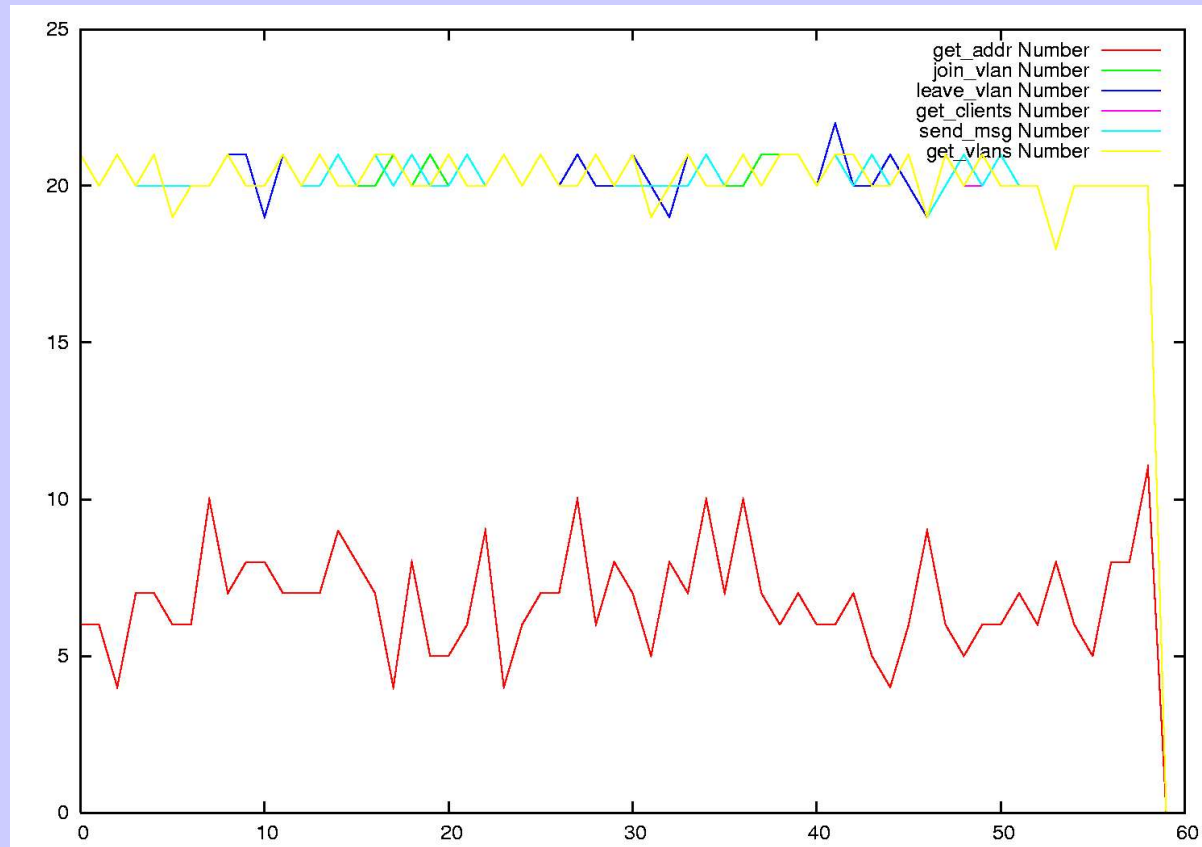


get_vlans message latency (milliseconds)

Performance analysis (III): an example

- ◆ Bottleneck detected with **adressOfClientRequest**, which was sent to the server too many times.
- ◆ Solved placing a client address cache.

Number of messages



Fault tolerance

- ◆ Failure inside a node (**process crash**):
 - ◆ Supervision tree has several policies depending on the type of process.
- ◆ **Node crash:**
 - ◆ Reconnection engine implemented in the clients.
 - ◆ Clients have a list of available servers.
 - ◆ Server nodes check if new connected clients come from crashed nodes.
- ◆ **Client crash:**
 - ◆ Need to distinguish between a client crash and a message lost.
 - ◆ Keep alive protocol and reactions in the server after crash.
 - ◆ Sometimes messages are lost. Performance is more important.

Use of Erlang/OTP inside SERVVAL

- ◆ Intensive use of Erlang behaviours.
- ◆ ASN.1 compiler for the internal protocol.
- ◆ Mnesia as distributed database.
- ◆ edoc for documenting the modules API.
- ◆ Erlang compiler with tinderbox for continuous integration.
- ◆ Integration with C for reading device files.
- ◆ ... more and more features ...

Most of the team members were using Erlang for the first time in an industrial project

*Ten thousand lines of Erlang code, lots of design and implementation documentation,
after 3 months of XP: basic functionality of the system*

Present and future work

- ◆ Congestion control protocol (client and server).
- ◆ Access control list.
- ◆ Complete support for the Virtual Ethernet Driver both in GNU/Linux (using TAP) and Windows.
- ◆ Secure communications (SSL support both in client and server).
- ◆ Improve support for TCP and UDP.
- ◆ Intelligent traffic filtering mechanisms for message intensive LAN protocols

Conclusions

- ◆ Prototype for a VLAN software switch with Erlang/OTP.
- ◆ Using Erlang/OTP had several advantages over considered alternatives.
- ◆ Available under GPL license: bazaar model of development.
- ◆ Solution with advantages over the available tools and technologies.
- ◆ Allows VLANs to be set all over the Internet. **Regional telecommunications company interested.**
- ◆ Still lots of things to be done. Promising results.

SERVAL: an Internet software VLAN switch developed in Erlang

Juan José Sánchez Penas (juanjo@lfcia.org, juanjo@igalia.com)

Alejandro García Castro (acastro@igalia.com)

Fco. Javier Morán Rúa (jmoran@igalia.com)

Project web: <http://serval.igalia.com>

(Includes design and edoc documentation and public CVS)

Project mailing list: serval-devel@sourceforge.net

Any kind of collaboration is welcome!