End User Agents: extending the "intelligence" to the edge in Distributed Service Systems

Internet2 Meeting
September 2005

Iosif Legrand
California Institute of Technology
MonALISA (Monitoring Agents using a Large, Integrated Services Architecture)
An Agent Based, Dynamic Service System able to Monitor, Control and Optimize Distributed Systems

LISA (Localhost Information Service Agent)
End User Agent, capable to effectively integrate user applications with Service Oriented Architectures.

Examples:
EVO system: a distributed videoconferencing system
Data transfers: creating on demand an optical path
MonALISA is A Dynamic, Distributed Service Architecture

- Real-time monitoring is an essential part of managing distributed systems. The monitoring information gathered is necessary for developing higher level services, and components that provide automated decisions, to help operate and optimize the workflow in complex systems.

- The MonALISA system is designed as an ensemble of autonomous multi-threaded, self-describing agent-based subsystems which are registered as dynamic services, and are able to collaborate and cooperate in performing a wide range of monitoring tasks. These agents can analyze and process the information, in a distributed way, to provide optimization decisions in large scale distributed applications.

- An agent-based architecture provides the ability to invest the system with increasing degrees of intelligence; to reduce complexity and make global systems manageable in real time.
The MonALISA Architecture Provides:

- Distributed **Registration and Discovery** for Services and Applications.
- Monitoring all aspects of complex systems:
  - System information for computer nodes and clusters
  - Network information: WAN and LAN
  - Monitoring the performance of Applications, Jobs or services
  - The End User Systems, its performance
- Can interact with any other services to provide in near real-time customized information based on monitoring data
- Secure, remote **administration** for services and applications
- Agents to supervise applications, to restart or reconfigure them, and to notify other services when certain conditions are detected.
- The MonALISA framework can be used to develop higher level decision services, implemented as a distributed network of communicating agents, to perform global optimization tasks.
- **Graphical User Interfaces** to visualize complex information
The MonALISA Discovery System & Services

- Fully Distributed System with no Single Point of Failure

- Clients, HL services, repositories

- Proxies

- AGENTS

- MonALISA services

- Network of JINI-LUSs Secure & Public

- Distributed Dynamic Discovery-based on a lease Mechanism and REN

- Global Services or Clients

- Dynamic load balancing

- Scalability & Replication

- Security AAA for Clients

- Distributed System for gathering and Analyzing Information.
MonALISA service & Data Handling

Client (other service) Web client

WEB Service
WSDL SOAP

Data Stores
Postgres MySQL

Data Cache Service & DB
Lookup Service

Registration

Discovery
Communications via the ML Proxy
data
Predicates & Agents

MonALISA Service

Applications

Configuration Control (SSL)

Client (other service) Java

User defined loadable Modules to write /sent data
Registration / Discovery
Admin Access and AAA for Clients

MonALISA Service

Registration (signed certificate)

Discovery

Client (other service)

MonALISA Service

Application

Filters & Agents

Client authentication

Data

Client (other service)

AAA services

Trust keystore

Admin SSL connection

Services Proxy Multiplexer

Lookup Service

MonALISA Service

Applications

Trust keystore

MonALISA Service
Communities using MonALISA

- OSG
- Grid3
- CMS
- ALICE
- VRVS System
- STAR
- D0
- ABILENE
- GLORIAD

It has been used for Demonstrations at:

- SC2003
- Telecom 2003
- WSIS 2003
- SC 2004
- I2 2005

More than 200 Sites running MonALISA and it monitors more than 12,000 nodes, more than 60 WAN links and Collects ~ 200,000 parameters /min

http://monalisa.caltech.edu
Monitoring OSG, GRID3, Running Jobs, I2 Network Traffic, and Topology

JOB Evolution

TOPOLOGY

ACCOUNTING
Monitoring I2 Network Traffic, Grid03 Farms and Jobs
Monitoring Network Topology
Latency, Routers

NETWORKS

ROUTERS
ApMon – Application Monitoring

Library of APIs (C, C++, Java, Perl, Python) that can be used to send any information to MonALISA services

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

Library of APIs (C, C++, Java, Perl, Python) that can be used to send any information to MonALISA services

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

- **Load1**: 0.24
- **Processes**: 97
- **Pages_in**: 83

- **Mbits_out**: 0.52
- **Status**: reading
- **MB_inout**: 562.4

- No Lost Packages

MonALISA hosts

Config Servlet

dynamic reloading

MonALISA Service

UDP/XDR Monitoring Data

ApMon configuration generated automatically by a servlet / CGI script
Monitoring the Execution of Jobs and the Time Evolution

SPLIT JOBS

LIFELINES for JOBS

Summit a Job

DAG

Job

Job

Job

Job

Job

Job
Monitoring ABILENE backbone Network

- Test for a Land Speed Record
- ~7 Gb/s in a single TCP stream from Geneva to Caltech
Monitoring VRVS Reflectors and Communication Topology
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 real-time measurements of the network performance.

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

minimum-spanning tree (MST)
A weighted connected graph $G = (V,E)$ with $n$ vertices and $m$ edges. The quality of connectivity between any two reflectors is measured every 2s. Building in near real time a minimum-spanning tree $T$

$$w(T) = \sum_{(v,u) \in T} w((v,u))$$
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
  - Provides embedded clients for IPERF (or other network monitoring tools, like Web 100)
  - A user friendly GUI to present all the monitoring information.
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 (e.g., 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.
Dynamic Discovery of Reflectors

Creates and maintains, in real-time, the optimal connectivity between reflectors (MST) based on periodic network measurements.

Detects and monitor the User configuration, its hardware, the connectivity and its performance.

Dynamically connects the client to the best reflector

Provides secure administration.

It is using alarm triggers to notify unexpected events.
MonALISA agents to create on demand on an optical path or tree

Discovery & Secure Connection

ML Demon

Time to create a path on demand <1s independent of the location and the number of connections

Optical Switch

Control and Monitor the switch

ML Agent MonALISA

ML proxy services used in Agent Communication

Runs a ML Demon

>ml_path IP1 IP4 "copy file IP4"

MonALISA agents to create on demand on an optical path or tree
Test Setup for Controlling Optical Switches

3 partitions on each switch
They are controlled by a MonALISA service

- Monitor and control switches using TL1
- Interoperability between the two systems
- End User access to service
Monitoring Optical Switches
Agents to Create on Demand an Optical Path
MonALISA is a framework capable to correlate information from different layers.

Near real time feedback between major layers is crucial for dynamic load balancing, adaptability and self-organization.