ISBN: 978-972-8924-39-3 © 2007 IADIS

A DISTRIBUTED AGENT SYSTEM FOR DYNAMIC OPTICAL PATH PROVISIONING

R. Voicu*, I. Legrand*, H. Newman *
California Institute of Technology, Pasadena, CA, USA*

C. Dobre†, N. Tapus†
"Politehnica" University of Bucharest, Romania †

ABSTRACT

In this paper we present a prototype system developed to enable data intensive grid applications to efficiently use and coordinate optical network resources, to improve the performance and throughput of global-scale grid systems, such as those used in high energy physics and many other fields of science. We developed a multi agent system for secure lightpath provisioning based on dynamic discovery of the topology in distributed networks. In this approach, agents will act on behalf of services, managing access to resources, and collaborate with other services to generate on demand effective end to end optical connections.

KEYWORDS

on demand end to end optical path provisioning, monitoring optical switches, distributed agents

1. INTRODUCTION

The High Energy Physics experiments at CERN's Large Hadron Collider [1], expected to begin operations in 2007, will face unprecedented challenges due to the volume and complexity of the experimental data, and the need for collaboration among scientists located around the world. The massive datasets which will be acquired, processed, distributed and analyzed are expected to grow to the 100 Petabyte level and beyond by 2010.

We used the MonALISA (Monitoring Agents using a Large Integrated Service Architecture) framework [2] for developing the Optical Control Plane system implemented as a distributed set of collaborating agents. The MonALISA service system provides near real-time access to complete monitoring information and allows analyzing and processing this information in a dynamic network of agents, capable to build higher level services.

2. THE MONALISA FRAMEWORK

MonALISA (Monitoring Agents in A Large Integrated Services Architecture) provides a distributed service for monitoring, control and global optimization of complex systems. It is based on an ensemble of autonomous multi-threaded, agent-based subsystems which are able to collaborate and cooperate to perform a wide range of monitoring and decision tasks in large scale distributed applications, and to be discovered and used by other services or clients that require such information. It is a fully distributed system with no single point of failure. The scalability of the system derives from the use of a multi threaded engine to host a variety of loosely coupled self-describing dynamic services, the ability of each service to register itself and then to be discovered and used by any other services, or clients that require such information.

2.1 The Agent Communication System in the MonALISA Framework

The MonALISA architecture, presented in Figure 1, is based on four layers of global services. The network of JINI [3] - Lookup Discovery Services (LUS) provides dynamic registration and discovery for all other

services and agents. MonALISA services are able to discover each other in the distributed environment and to be discovered by the interested clients. The registration uses a lease mechanism. If a service fails to renew its lease, it is removed from the LUSs and a notification is sent to all the services or other applications that subscribed for such events. Remote event notification is used in this way to get a real overview of this dynamic system.

The second layer represents the network of MonALISA services that hosts many monitoring tasks through the use of a multithreaded execution engine and a variety of loosely coupled agents that analyze the collected information in real time. The collected information can be stored locally in databases. Agents are able to process information locally and to communicate with other services or agents to perform global optimization tasks. A service in the MonALISA framework is a component that interacts autonomously with other services either through dynamic proxies or via agents that use self-describing protocols. By using the network of lookup services, a distributed services registry, and the discovery and notification mechanisms, the services are able to access each other seamlessly. The use of dynamic remote event subscription allows a service to register an interest in a selected set of event types, even in the absence of a notification provider at registration time. The third layer of Proxy services, shown in the figure, provides an intelligent multiplexing of the information requested by the clients or other services and is used for reliable communication between agents. It can also be used as an Access Control Enforcement layer to provide secures access to the collected information.

Higher level services and clients access the collected information using the proxy layer of services. A load balancing mechanism is used to allocate these services dynamically to the best proxy service.

The ability of the agent-based architecture to self-organize, adapt rapidly to changing conditions, and be invested with increasing degrees of intelligence and autonomy, dramatically simplifies the operation and management of global-scale grids.

Inside the MonALISA framework the services are able to communicate between them through a set of proxy services. The system uses Jini for dynamic registration and discovery mechanism of all the services and agents. The communication between services is made through a set of proxy services which provides an intelligent way multiplexing the information requested by other services or clients. The proxy and lookup services are replicated to guarantee reliability through redundancy. Every proxy can handle more than 1000 messages per second and the framework uses more than one proxy to achieve reliable communication between agents.

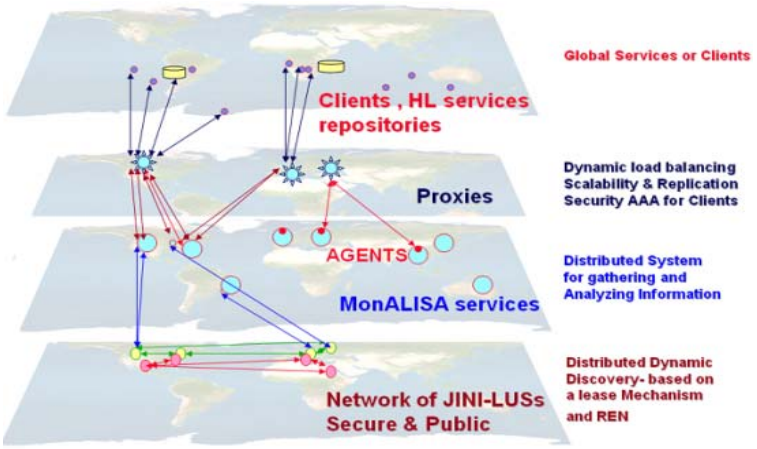

Figure 1. The MonALISA Architecture

Agents are hosted by the monitoring services and are able to publish and discover themselves based on a group name. Every agent must be a member of at least one group and is able to communicate only with the agents from the same group. The messages sent between agents may be unicast or broadcast messages inside the group. When an agent joins or leaves a group all the other agents inside the group are automatically notified.

2.2 Monitoring Modules

We developed two monitoring modules which run embedded in the MonALISA service and provide information about the optical power on ports and the state of the cross-connect links inside the switch in near real-time. The modules use Transaction Language 1 (TL1) [4] commands to retrieve monitoring information from the optical switch. Based on the monitoring information the agent is able to detect and to take informed decisions in case of eventual problems with the cross connections inside the switch or loss of light on the connections.

3. DYNAMIC LIGHT PATH PROVISIONING

The Optical Switch Agent is a software agent that is dynamically deployed and runs embedded in a MonALISA service. Its role is to monitor and control an optical switch, to publish and to continuously update its configuration. The configuration consists of the port map, which specifies the devices attached to the switch, state of the ports, optical cross-connects inside the switch and the necessary routing information. The agents use the MonALISA framework to discover each other, publish their configuration, and collaborate to create on demand and end to end optical path.

3.1 The Path Provisioning Algorithm and Implementation

The design and implementation of the algorithm followed the main requirement for this prototype system: being able to establish an end-to-end connection in the shortest possible time. In order to achieve this, every agent in the system has the exact view of the network and adapts very quickly to changes. The network topology, implemented as a network graph [5], has agents as vertices and optical links between switches as edges, every edge having a cost associated with it. The system is modeled using a directed graph [5] because is possible to have both full-duplex and simplex links between optical switches. Every agent in the graph computes a shortest path tree using a variant of Dijkstra's algorithm. The agents system uses a two-phase commit strategy for creating the optical cross-connects and a lease mechanism to guarantee the reliability in case of partial failures.

An agent that receives a lightpath request determines, based on the local tree that is already built, if the request can be fulfilled or not. If it is possible initiates transactions with both the local and remote ports involved in the path. Once the transaction is started, the agent assigns a unique ID for the path, sends the remote cross-connect commands and after that it tries to establish the cross-connects on the local switch. An independent thread is waiting for acknowledgements from the remote agents. Any remote agent which receives such a cross-connect request starts a local transaction only with the ports involved in the cross-connect. If it succeeds in creating the cross-connect, it commits the local transaction and it sends back an "acknowledged" message, otherwise the transaction is rolled-back and a "not acknowledged" message is sent. Based on the received messages the local agent takes the decision whether or not the transaction can be committed or it has to be rolled back. The algorithm described above is reliable and guarantees that the system remains in a consistent state even if a network problem occurs. The newly created lightpath has a lease assigned which must be renewed by all the involved agents and in this way it can provide a viable mechanism for the system to recover from partial failures.

To improve the performance and the response time all the functions executed by an agent are performed in asynchronous sessions using a pool of threads. A task can be a request for a lightpath from a client or a cross connect request coming from another agent, or a rerouting task triggered by a topology change. The only sequential part of the algorithm described above is in the "pre-commit" phase, and this involves only the ports that are supposed to be in the lightpath. Any request submitted during this phase, which do not involve these ports can be fulfilled in parallel.

Using this information, an agent is able to detect the loss-of-light on fiber, and take specific decisions if the port is part of a lightpath. The agent who detects the problem notifies the initiator, which is responsible to try to reroute the traffic through another path, if this is possible. When the initiator detects a change in topology that affects the lightpath, it forces the shortest path tree to be recalculated. Based on the new tree, the agent is able to take the decision if the light can be rerouted using other path, or it can tear down the entire path. This is very useful, because in case of successful rerouting, the problem will not disturb already established sockets, upper network layers, like TCP, not being able to detect the problem.

The routing algorithm used to establish an end to end lightpath is similar with link-state routing protocols. The work presented here uses an algorithm similar to link-state routing algorithms because they converge faster than distance-vector algorithms. In order to guarantee consistency and reliability of the entire system, a two-phase commit strategy and a lease mechanism were also developed.

3.2 Security and User's Authentication

The system is integrated in a secure and reliable way with the end user applications and provides simple shell like commands to create on demand light path between end systems. Every agent accepts light path requests through a secure connection based on X.509 certificates. The client certificates must be signed by a certification authority (CA) which is trusted by the optical switch agent which accepts the requests. It is also possible to import the client's public key in the trusted store of the agent. The requests may be provided from a command line interface or from the MonALISA GUI Client. An optical switch daemon (OSD) must be installed on the end host machine. It locates the optical switch agents from a specific group and connects to one of them, usually to the closest one, based on topology information. The shell commands use UNIX named pipes to communicate with the daemon which routes the requests to the optical switch agent.

3.3 The User's Interface

In the MonALISA framework the overall status of the dispersed systems being monitored is provided by either a GUI client or through specialized web portals. For the dedicated modules and agents used to monitor and control Optical Switches the GUI client of MonALISA provides a dedicated panel. This panel facilitates the interaction between users and the monitored Optical Switches. It offers to the end user a number of features such as complete perspective over the topology of the monitored optical networks or the possibility to monitor the state of the Optical Switches or the possibility to dynamically create new optical paths.

The main panel is presented in Figure 2. The principal aspect of this panel is that it displays in an intuitive way the current topology of the monitored Optical Switches and the links between. For the Optical Switches we use different colors to represent the state of their internal ports and the state of the links between the represented entities. In the panel, as seen in Figure 2, besides the Optical Switches a number of other devices (the blue ovals) can also be represented. These devices, equipped with optical network cards, are connected by optical links to the Optical Switches being monitored.

The Optical Switches and the links connecting them are also represented in the 3D Map panel (Figure 3). This panel locates the MonALISA monitoring services on a 3D view of the world geographical map. It also shows the monitoring WAN links, real-time traffic on them, the capacity of the links, the connectivity between sites, the optical switches controlled and other parameters like sites Load, CPU usage, IO parameters, etc. In this way the user is presented with an easy to use complete visualization tool which represents the global state of the entire monitored systems.

For the optical paths created from this panel (represented in green color in Figure 2) the user is presented with the details of which devices and ports one particular path is crossing (from end to end) together with the components involved in that path (the highlighted devices in Figure 2).

The panel can be also used to create new optical paths or delete existing ones. In order to gain access to these operations the user must present though the necessary credentials. The security layer is implemented using RMI over SSL.

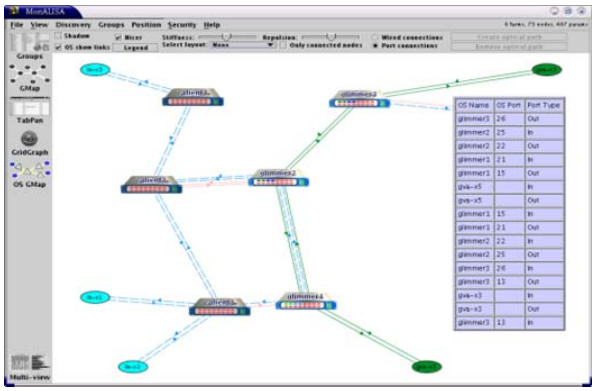

Figure 2. Network topology and current status in main panel

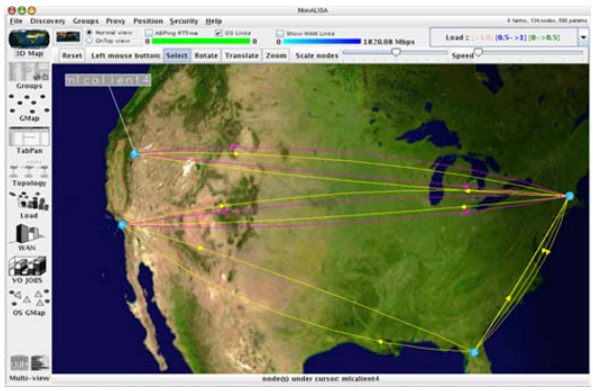

Figure 3. Same topology and status on the 3D Map panel

4. RESULTS

We developed dedicated modules for two types of optical switches, Glimmerglass [6] and Calient [7]. The system is currently used to create dynamically on demand optical connections between computers located at CERN (Geneva) and California Institute of Technology (CALTECH) located in Pasadena, CA, using the networking infrastructure of USLHCNet [8] and Internet2 [9].

USLHCNet provides two transatlantic 10 Gb/s optical links between CERN and Starlight (Chicago) and MANLAN (New York). On the Internet2 network, the pure optical connections are simulated using several VLANs to provide direct connections from Chicago and New York to CALTECH. The topology of the network infrastructure used is shown in Figure 4. The system is able to create dynamically an end to end lightpath in less than one second independent of the number of switches involved and their location. It monitors and supervises all the created connections and is able to automatically generate an alterative path in case of connectivity errors. The alternative path is set up rapidly enough to avoid a TCP timeout, and thus to allow the transfer to continue uninterrupted.

The optical fibers are simulated through two VLANs between the two optical switches. One VLAN is routed through New York and the other one through Chicago. The monitoring module is able to simulate a fiber cut. The optical agent will detect this as a real loss-of-light and will try to reroute the path. In the example above a disk to disk transfer is presented, using two 4-disk servers, one at Caltech and the other one at CERN in Geneva. During the transfer four fiber cuts were simulated, corresponding to the four drops in the Figure 4. In the example above a disk to disk transfer is presented, using two 4-disk servers, one at Caltech and the other one at CERN in Geneva. During the transfer four fiber cuts were simulated, corresponding to the four drops in the Figure 4. These fibers cuts simulations were done on the Geneva – Starlight and Geneva – Manlan links and the transfer was rerouted four times between these two links. The "fiber cut" and the reroute are done quick enough that the TCP does not sense the loss in connectivity and the transfer continues. The recovery time differs for various TCP stacks and the round trip time between end points.

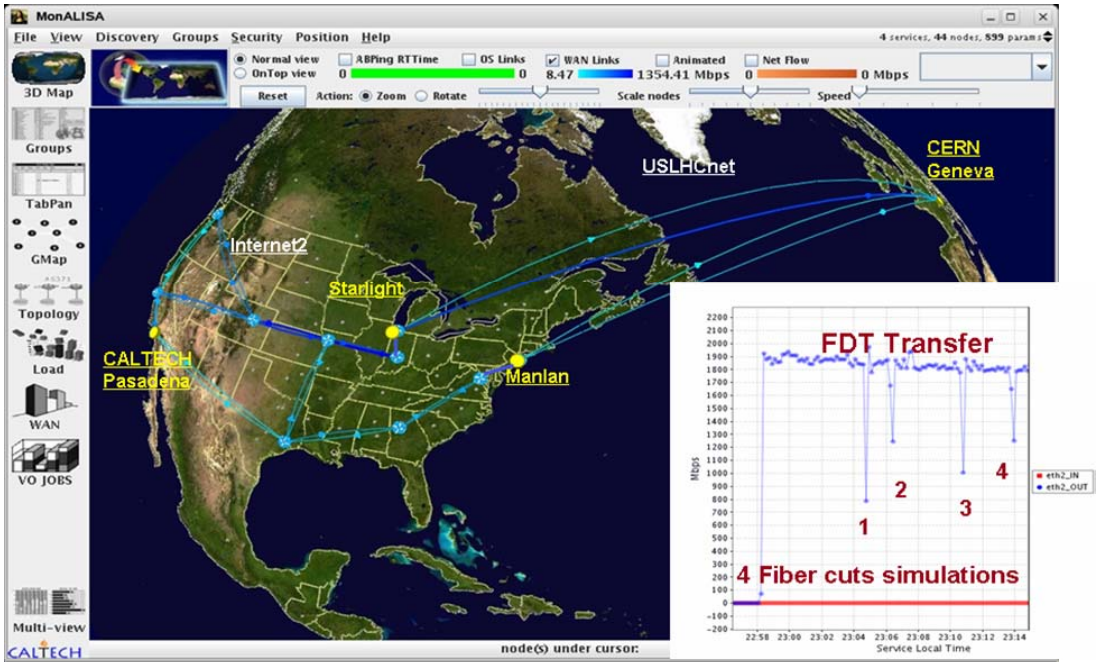

Figure 4. The network topology used for creating dynamically, on demand an end to end optical path. The total disk to disk throughput between a server at CERN and one at CALTECH. Four "fiber cuts" were simulated during the transfer. The throughput drops when a rerouting is done, but it recovers quickly.

5. CONCLUSION AND FUTURE WORK

The MonALISA agent framework was successfully used to develop an integrated Optical Control Plane system that controls and creates end-to-end optical paths on demand. It is a distributed system, without a single point of failure. The system automatically detects network errors and is capable to create an alternative path rapidly enough to avoid a TCP timeout, so that data transport continues uninterrupted. This new approach enables data intensive applications to dynamically use and coordinate network resources. The distributed agent system can be used to directly control network devices, or it can be interfaced with optical control planes protocols.

We are working to further develop this distributed agent system and to provide integrated network services capable to efficiently use and coordinate shared, hybrid networks and to improve the performance and throughput for data intensive grid applications. This includes services able to dynamically configure routers and to aggregate local traffic on dynamically created optical connections. We are also developing agents able to interoperate with standard protocols (MPLS [10], GMLPS [11]) and other network services (Dragon [12] and UCLP [13]). We are planning to integrate the current work in the future developments setting up dynamic VLANs using VCAT/LCAS technologies.

REFERENCES

[1] CERN Large Hadron Collider (CERN LHC) web page http://lhc.web.cern.ch/lhc/

[2] MonALISA web page http://monalisa.caltech.edu/

[3] Jini web page http://www.jini.org/

[4] Transaction language 1 – TL1 http://en.wikipedia.org/wiki/TL1

[5] Harary, F. Graph Theory. Reading, MA: Addison-Wesley, 1994

[6] Glimmerglass http://www.glimmerglass.com/

[7] Calient Networks http://www.calient.net/

[8] USLHCNet http://www.uslhcnet.org/

[9] Internet2 http://www.internet2.edu/

[10] Rosen, E., Viswanathan, A. and R. Callon, "Multiprotocol label switching Architecture", RFC 3031, January 2001

[11] E. Mannie, "Generalized Multi-Protocol Label Switching Architecture", draft-ietf-ccamp-gmpls-architecture-07.txt, November 2003.

[12] The Dragon Project web page: http://dragon.east.isi.edu/

[13] UCLP (User Controlled LightPath Provisioning) web page: http://phi.badlab.crc.ca/uclp/