ON DISTRIBUTED SYSTEM SUPERVISION - A MODERN APPROACH: GENESYS

Jean-Eric Bohdanowicz[1], Laszlo Kovacs[2], Balazs Pataki[2], Andrey Sadovykh[3] and Stefan Wesner[4]
[1]EADS SPACE Transportation, 66, route de Verneuil, BP 3002, 78133 Les Mureaux, France; [2]MTA SZTAKI, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, Hungary; [3]LIP6, Laboratoire d'Informatique, Université Paris 6, 8, rue du Capitaine Scott, 75015 Paris, France; [4]HLRS - Stuttgart University, High Performance Computing Centre Stuttgart, Allmandring 30, 70550 Stuttgart, Germany.

Abstract: This article presents limitations of current network management standards in the context of comprehensive distributed system supervision. As a proposed solution, the article describes the GeneSyS project achievements, a modern approach allowing straightforward integration of all monitoring/control means, as well as providing basic intelligence capabilities. These issues are illustrated on several industrial examples.

Key words: SNMP, GeneSyS, supervision, distributed management, intelligent agent, Web-Services.

1. INTRODUCTION

For more than 30 years, the information systems have moved forward from a single computer to distributed computer societies.

These new information systems involve different heterogeneous components working together and include groupware, collaborative engineering, distributed simulation and distributed computation resources management (GRID) systems

The maintenance activities of such systems include:

- **Application management** including deployment, set-up, start, stop, hold/resume, configuration management (for instance, for redundancy management purpose) and resource management;

- **Operating systems management** comprising resource usage monitoring and control;
- **Time synchronisation**;
- **Network management** including parameterisation and performances (e.g. dynamic control of bandwidth allocation) **and monitoring**;
- **Security management**, like authentication/authorisation control;
- **Archiving** and etc.

The difficulty of these tasks depends on the network deployed, the components spread, the number of components, the availability of compatible management tools and the infrastructure. Working with thousands of parameters simultaneously becomes uneasy without intelligent solutions categorising, synthesising and filtering them, as well as situation pattern recognition and prediction mechanisms.

Historically, the principles of network supervision are older than those governing the frameworks and mainly based upon the SNMP protocol (and its extensions). There exist many commercial frameworks, like Unicenter TNG, HP OpenView, etc, using SNMP not only for network management, but also for application management. These frameworks inherit SNMP advantages: performance, maturity for network management, multiple compatible devices; as well as well known disadvantages: lack of security (UDP based), lack of complex data types support that makes it difficult to build intelligent solutions.

There exists a JMX (Java Management Extension) specification that solves earlier mentioned issues for Java platform systems. Besides, there exist frameworks (IBM Tivoli, etc.) using middleware standards, like the OMG CORBA.

However, several common constraints can be identified for available supervision technologies and frameworks:

- **Interoperability issues:** Components written on different languages using different toolkits, which are supposed to use the same architecture specification, may not be capable to co-operate on a full scale.
- **Components portability:** Often components are built to work only under their native operating systems like MS-Windows or UNIX. They are very sensible to transport mechanism and to low level communication protocols, in general.
- **Development/deployment complexity:** Many commercial applications have proprietary APIs that makes it difficult to create new agents and to plug them to the existing supervision systems.
- **Non-flexible architecture:** When agent and visualisation tools are realised in the same component, upgrades of the console impact agent functionality and vice versa.

- **Dedication to a particular monitoring layer, lack of comprehensive solutions:** For instance, there exist various application layer tools to supervise Oracle database. It would be very useful to get simultaneously the system information and network statistics to better control the system.
- **Lack of intelligence support:** Dealing with thousands of relevant parameters simultaneously is a laborious task
- **Lack of integration capability:** Often application management can't be supplemented with existing network management solutions due to the lack of integration capability.

The next section introduces the GeneSyS project intended to overcome these limitations.

2. GENESYS

GeneSyS (Generic System Supervision) is a European Union project (IST-2001-34162) co-funded by the Commission of the European Communities (5th Framework). EADS SPACE Transportation (France) is the project Co-ordinator, with University of Stuttgart (Germany), MTA SZTAKI (Hungary), NAVUS GmbH and D-3-Group GmbH (both of Germany) as participants. GeneSyS started in March 2002 and is due to be completed in October 2004[1]. The project is aimed at developing a new, open, generic and modular middleware for distributed systems supervision. Besides, the consortium intends to make GeneSyS an open standard in the distributed system supervision domain.

2.1 Proposed Solution

The protocol based supervision architectures (ICMP, SNMP) have the most remarkable interoperability characteristics due to the fact that, their message format is strictly fixed and they do not impose any limitations on a component implementation, requiring only the protocol support. This makes their usage independent from operating systems and programming languages.

Their force is also their weakness. The strict message format makes it difficult and often impossible to operate with a custom data required for modern supervision systems. The network management protocols are inseparable from their transport protocols.

Meantime, Web technologies provide with flexible means to build custom, XML based protocols and portable transport mechanisms independent from network protocols (Web Services).

Our proposal is to combine Web technologies to build a supervision middleware, which shares advantages of protocol based architectures: operating system and programming language independency; and provides flexible and customisable messaging protocol, as well as network portability.

2.2 Web Technologies as a Platform for a Supervision Framework

With the advancement of Web technologies, more and more works appeared to introduce these technologies in the world of supervision (DMTF WBEM, OASIS WSDM, etc.). GeneSyS was one of the firsts to bring the Web Services to this domain.

As a result of our research, an agent based approach was implemented which separates the monitoring/controlling and visualisation of monitoring data. Web Services technologies were chosen as the base for GeneSyS messaging protocol.

Basing the supervision infrastructure on agents seems logical, because the monitoring of IT entities requires properties that are available with software agents. A software agent is a program that is authorised to act for another program or human (see[2]). Agents possess the characteristics of delegacy, competency and amenability that are the exact properties needed for a monitoring software component.

Delegacy for software agents centres on persistence. Delegacy provides the base for an agent, which makes it an autonomous software component. By taking decisions and acting on their environment independently, software agents reduce human workload by interacting with their end-clients when it is time to deliver results. In case of GeneSyS, the agents reside either on the computer hosting the monitored entity or on a computer that is able to communicate with the monitored entity.

Competency within a software environment requires knowledge of the specific communication protocols of the domain (SQL, HTTP, API calls). A monitoring agent competency is to have knowledge about the monitored entity to be able to collect runtime information from it or to control it with commands.

Amenability in intelligent software agents can include self-monitoring of achievement toward client goals combined with continuous, online learning to improve performance. GeneSyS makes no restriction on its agents or on their intelligence or autonomous operations, but provides the ability to include it as found necessary by agent writers and also provides some middleware components (like monitoring data repository) that can be used to implement amenability.

Openness and standards based solution is one of the key requirements of GeneSyS especially in the light of the Consortium's intention to turn GeneSyS itself into an industry standard. After a number of iterations, we had two candidates for the realisation of the communication protocol:

- InterAgent Communication Model (ICM - FIPA based) (cf.[3])
- Web Services technologies (see[4])

The ICM framework has not been designed for monitoring or supervision needs but is a general communication framework for inter-agent communication. The Web Services framework standardised by the W3C is a generic framework for the interaction of Services over the Internet and is designed to exploit as much as possible existing protocol frameworks such as SOAP and HTTP. The Web Services framework is in contrast to ICM more a hierarchical or client-sever communication model.

Web Services has a major problem with respect to performance. The use of an XML based protocol cannot be as efficient as a binary protocol due to the text processing, which is highly performance consuming. Additionally, the most common transport protocol used for SOAP messages, the Hypertext Transfer Protocol (HTTP), is not very efficient as it lacks stateful connections. However we are convinced that these problems can be solved as Web Services potentially can use different protocols. The feature of alternative protocol bindings is already used for example in the .NET framework using Remoting, which uses different (proprietary) protocols. As this problem is not solely part of GeneSyS but the whole community including the major software vendors that are committed to Web Services will face this problem, the assumption that this limitation will disappear seems reasonable.

After a detailed comparison of these two technologies, we selected Web Services, including the SOAP XML based communication protocol as a base for GeneSyS. Going on the Web Services path, we have a strong industry backing with tools available for many languages. With this decision, we also defined the first instance of a Web Services based supervision system that has recently been followed by other companies and standards organisations (OASIS WSDM, DataPower Technology[5])

On top of SOAP and Web Services, a new layer of the GeneSyS protocol has been established called the GeneSyS Messaging Protocol (GMP). GeneSyS Messaging Protocol is a lightweight messaging protocol for exchanging structured supervision information in a decentralised, distributed environment. It is an XML protocol based on XML 1.0, XML Schema and XML Namespaces. GMP is intended to be used in the Web Services Architecture, thus, SOAP is considered as a default underlying protocol. However, other protocol bindings can be equally applied. Using XML to represent monitoring data was a natural choice. XML is a widely accepted industry standard that supports structured representation of complex data types, structures (enumerations, arrays, lists, hash maps, choices, and sequences) and it can be easily processed by both humans and computers. With the wide acceptance of XML, integration with supervised application and 3d party monitoring solutions can be smoothly achieved, since XML toolkits are available for every platform.

2.3 Basic Components and Communication Model

This section provides implementation details, illustrating common supervision framework architecture.

Fig.1 depicts the basic GeneSyS functionality.

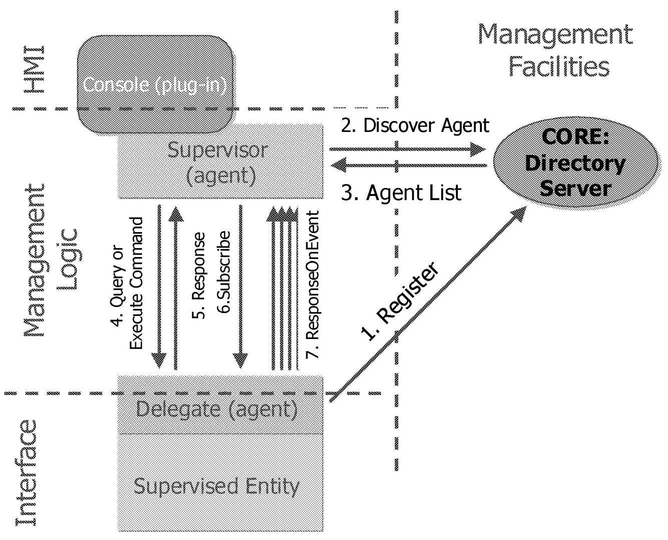

Figure 1. GeneSyS Communication Model

As showed above, supervision process involves several generic components. The Delegate implements an interface to the Supervised Entity (Operating System, Network, Applications, etc.), retrieves and evaluates monitoring information and generates monitoring events. The Supervisor is a remote controller entity that communicates with one or more Delegates. It may encapsulate management automation functionality (intelligence), recognising state patterns and making recovery actions. The Console is connected to one or many Supervisors to visualise the monitoring information in a synthetic way, and to allow for efficient controlling of Supervised Entity. The Core implements Directory Server, a location storage being updated dynamically.

The agents are registered in the Core with the purpose to be discoverable by other agents. Hereafter, the "agent" is a generic term comprising the Supervisor and the Delegate.

Both "pull" and "push" interaction models are available. The pull model is realised by the Query/Response mechanism, while the Event Subscribe mechanism secures the push model. All interactions between agents are provided for by the SOAP-RPC. The flexibility of XML standard is used to encode communication messages (GeneSyS Messaging Protocol) supporting complex data structures and custom data types.

2.4 Integration Capability

As a result, transport mechanism (SOAP-RPC) and the GeneSyS architecture itself are very flexible. That allows smooth integration with existing management frameworks.

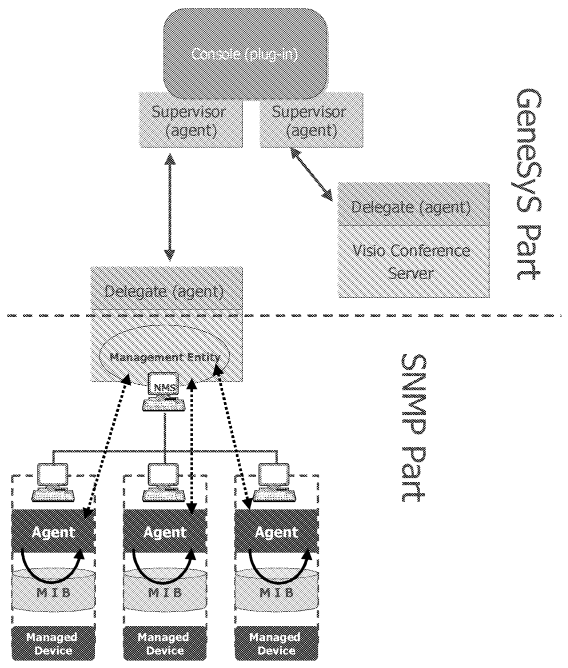

Figure 2. GeneSyS - SNMP Collaboration

Fig. 2 gives an example of collaboration between GeneSyS and SNMP. An SNMP Management Entity, a Network Management System, is plugged to a GeneSyS delegate, which makes network management information available at the administrator console. Thus, this information can be processed together with information of other agents (operating system, middleware, applications, like Visio Conference Server on the Fig. 2) in order to synthesise all the metrics in a single global view.

Hence, this SNMP bridge concept gives opportunity to benefit from both network and application level management. In that way, it is used in some of GeneSyS network agents mentioned lately in the following applicability sections.

2.5 Intelligence

An inherent property of software agents is autonomy, that is, the ability to work without the intervention of other programs or humans. Autonomous work requires some level of intelligence so that the agent can react on changes in its environment or can make decision based on its internal logic driven by rules or other means. Intelligence in agents is also required because in a complex environment with some 10 or 100 monitored entities, an administrator could be easily flooded with low level warnings like "memory is running low" or "maximum number of users almost reached". Instead, the administrator first needs a general, summarised view about the health of the systems and then can look at the details as necessary.

The GeneSyS framework provides API hooks for adding intelligence to agents as well as components for supporting the implementation of intelligence. Intelligence can be accomplished in several ways, which are only outlined here, as the actual implementation of this feature is not a main goal of GeneSyS:

- Specific Implementation: the "intelligence" to react on the system status can be done as part of the program code of the agent.
- Parameter based Generic Solution. The rules can be configured through parameters. A basic example is a "Threshold Miss Agent" where the parameters would be min and max values.
- Rule Based Systems. In complex settings, the usage of rule based systems could be an option where the rules can be expressed in an external file e.g. based on JESS.
- Workflow based systems. Another option could be to use workflow languages such as BPEL4WS to define workflows that act depending on events receive.

GeneSyS provides a data Repository that is connected to the middleware bus via the same API as any other agents, which means its functionality is available to all other agents connected to a given CORE. The Repository provides a generic XML data storage facility. Agents can store monitoring or control messages in the Repository, which can later be queried. With the use of the Repository, an agent can base its decisions on archived data, for example, by analysing past messages for detecting trends in the operation of the monitored entity. More over, the Repository is also capable for storing control messages – or a list of control messages – which can be "replayed" any number of times at any time it is necessary.

The Agent Dependency Framework (ADF) is another aid for adding intelligence to monitoring. ADF allows defining dependencies of monitored entities. To be more precise, not directly the dependencies of monitored entities but the dependencies of the agents monitoring the entity can be described. Each delegate agent can describe in its component description (which is stored in the Core) what agents it depends on. The dependency forms a directed graph that should never cause a circular reference. Once the dependency of each delegate is described, the dependency graph can be queried from the Core. Based on the dependency graph, a special supervisor console view can be created that draws a tree view of the dependent entities and gives a quick overview of the health of the system with green, yellow and red light depicting a healthy, questionable or erroneous state of the dependant systems. This way of visualising the monitored system with all its dependent components provides a way for tracking root cause of problems. For example, an administrator seeing a red light in the top of the dependency hierarchy can expand the tree until he finds the subsystem that generates the red light and which has been "propagated" up in the dependency tree. In the same way, an autonomous intelligent agent can walk this tree and find the root cause of the problem and can work only with that subsystem that was the source of the problem.

3. APPLICABILITY RESULTS

This section illustrates flexibility, integration capability and basic intelligence features with several real-life examples of the GeneSyS framework in use. The scenario was intended to prove the viability of the GeneSyS concept. Common system and network agents were developed to reflect system administrator needs. Custom application agents were used to monitor the system functional status (application load, resources used by applications, etc), user activities (documentation in use, on-line meetings, access violation, etc).

The main goal of these scenarios was to prove the capability of Web Services based distributed system to work in a heterogeneous environment. It includes support of different operating systems (Windows, Linux), programming languages and toolkits (C/C++/gSOAP, Java/Axis, .Net). Besides, while developing custom application agents (Oracle, EDB, GTI6-DSE, Mbone, Tomcat), the integration capability was ensured.

3.1 Distributed Training Scenario

This scenario was brought by EADS SPACE Transportation, the European aerospace industry leader. It concerns HLA-based simulations. HLA (see[6]) is a DoD standard for real-time interactive simulations. This standard is widely used in military, aerospace and automotive industries. The Distributed Training Scenario involves 4 real-time simulators playing different roles in joint training sessions of astronauts and ground controllers in order to prepare them in advance for contingency situation during the ATV to International Space Station (ISS) approach manoeuvre. The trainee teams are located in different places all over the world (Toulouse, Houston, Moscow), which imposes performance constraints on a supervision solution.

The flexible GeneSyS information allowed customising of System and Network agents and development of scenario specific Middleware and Application agents (RTI middleware, DIS-RVM application).

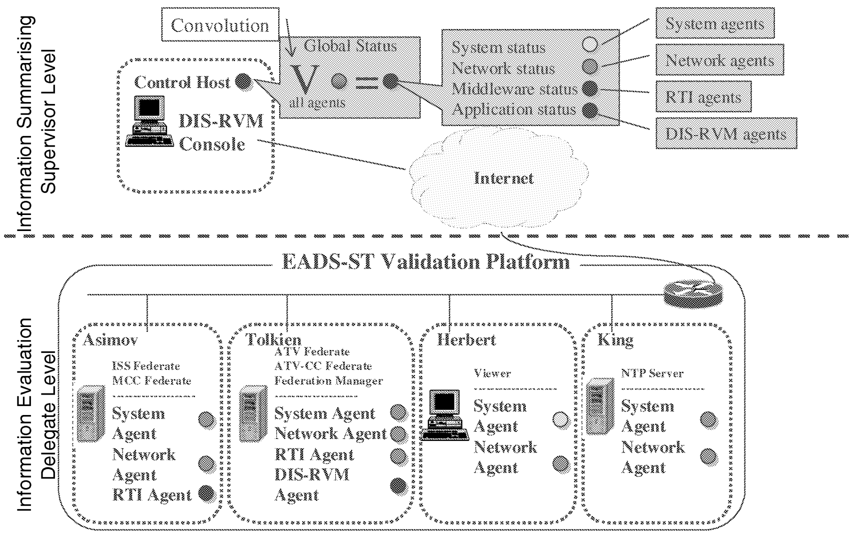

Figure 3. Intelligence in Distributed Training Supervision

Figure 7 depicts the deployment schema and gives intelligence implementation hints.

The "synthetic view" and "agent dependencies framework" approaches were used to provide administrators with a run-time system operation status summary and to allow a fast problem location.

Thus an administrator could browse down the agents to find a problem origin and then maintain the system.

3.2 Web Servers Scenario

The Web Servers Monitoring validation scenario aims at using GeneSyS for monitoring and controlling web servers and web based on-line services.

A Web Server is typically more than just an HTTP daemon: it may invoke external programs and those programs may use other programs for their execution, and so on. A typical Web Server can include, for example, an Apache server with a PHP interpreter and a MySQL database used by a number of PHP application. The Web Server is considered "healthy" only if all of these components are in good condition. Because these components may be dependent of each other it is not enough to have separate agents for all entities but these agents must be connected in a way to reflect the dependencies of the monitored entities.

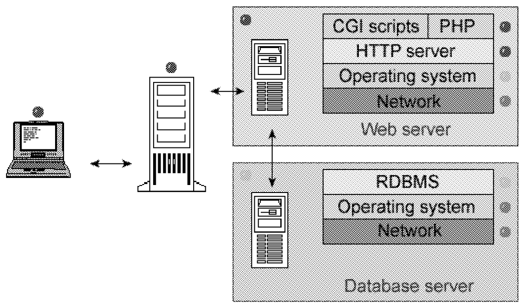

Figure 4. Web Application - A Common Deployment

Going on with the previous example, a Web Server could be considered healthy if the Apache daemon is up and running, the PHP applications it hosts respond in an acceptable time interval and the MySQL server has enough space for new records. If any of these conditions are not met the system should notify the administrator. More over, the unresponsiveness of Apache may be the result of a number of other dependent subsystems, like the operating or network system. So the "monitoring entity" could be divided into some more elements, namely the Apache server itself, the underlying operating system and the network connecting the server machine to the outer world. In this case even if Apache is found to be alive the operating system agent may report that the CPU load is too high and this could cause in a short time the Apache server being unable to respond to requests.

The Web Servers Monitoring scenario extensively uses the Agent Dependency Framework of GeneSyS, which provides the ability to describe the dependencies of system components and use this dependency graph to detect and find root cause of an erroneous system state.

4. CONCLUSION

This article presents an innovative supervision middleware intended to supplement classical distributed management approaches based on SNMP. The proposed framework has great integration capability illustrated on different real-life applicability examples. That permits using it in conjunction with existing network management infrastructure.

In comparison with other solutions, among other advantages, the authors would like to emphasise that the GeneSyS architecture is open to be extended with custom agents for all kind of applications.

The validation showed that, besides some ergonomics and performance issues, the solution is ready for the large community of the Internet users. That is why, generic components for system and network monitoring, as well as, visualisation tools, service components and development toolkits were released under open source policy and can be found at the GeneSyS SourceForge repository (see[7])

REFERENCES

1. GeneSyS project official web-site: http://genesys.sztaki.hu
2. Wallace Croft, David, "Intelligent Software Agents: Definitions and Applications", 1997, http://www.alumni.caltech.edu/~croft/research/agent/definition
3. The Inter-Agent Communication Model (ICM), Fujitsu Laboratories of America, Inc., http://www.nar.fujitsulabs.com/icm/about.html
4. Web Service Activity of W3C, http://www.w3.org/2002/ws/
5. DataPower Offering Web Services-Based Network Device Management, http://www.ebizq.net/news/2534.html
6. HLA, Institute of Electrical and Electronic Engineers - IEEE 1516.1, IEEE 1516.2, IEEE 1516.3
7. GeneSyS project SourceForge file repository, http://www.sourceforge.net/projects/genesys-mw