The following paper was originally published in the
Proceedings of the Twelfth Systems Administration Conference (LISA '98)
Boston, Massachusetts, December 6-11, 1998

An NFS Configuration Management System and its Underlying Object-Oriented Model

Fabio Q. B. da Silva, Juliana Silva da Cunha, Danielle M. Franklin,
Luciana S. Varejao, and Rosalie Belian
Federal University of Pernambuco

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

An NFS Configuration Management System and its Underlying Object-Oriented Model

Fabio Q. B. da Silva, Juliana Silva da Cunha, Danielle M. Franklin, Luciana S. Varejão, and Rosalie Belian – Federal University of Pernambuco

ABSTRACT

This paper describes an NFS configuration and management system for large and heterogeneous computer environments. It also shows how this system can be extended to address other services in the network. The solution is composed of a process that describes service configuration and management life-cycle, a modular architecture and an objected oriented model. The system supports multiple features, including: automatic host and service installation, service dependency inference and analysis, performance analysis, configuration optimization as well as service functioning monitoring and problem correction.

Introduction

The installation and configuration of hosts and services are two very common tasks in the administration of any computer network. Even small and homogeneous networks offer several different services that must be consistently configured on server and client computers. Furthermore, configuration management becomes a very complex problem when large and heterogeneous systems, together with their Internet connection, are taken into account. According to several authors [2,3], some of the reasons for this increase in complexity are [1]:

- Configuration of each service must be uniform over the network, requiring update on every host anytime a configuration change is required;
- In general, there are great differences in the actual format and location of the configuration files of a service for each platform. Therefore, to configure a service in a heterogeneous network amounts to configure the service in each platform separately;
- Each operating system provides its own set of non-standard configuration parameters for the common network services. In most cases, it is necessary to learn and use these non-standard features to achieve optimal performance of the service in each platform.
- Usually, large networks are managed by a team of system administrators. Configuration rules and parameters must be effectively communicated among team members to avoid inconsistencies that can arise due to personal preferences during the configuration process.
- It is difficult to detect and correct services misfunction before any damage to the users;
- It is also difficult to see dependencies between the services.

The configuration of hosts and services have dramatic influence on network performance, resilience and safety, and therefore are among the most critical tasks in system administration. For instance, it is widely known that a large percentage of security problems on networks connected to the Internet are due to bad Internet services (like HTTP and FTP) configuration.

Therefore, more systematic and structured processes of service configuration are necessary for the administration of today's networks and, in particular, those that are connected to the Internet. Furthermore, to achieve high levels of consistency and safety, any such process must be supported by automated tools which must possess three fundamental properties:

- efficient, robust and user friendly operation on large networks (hundreds of servers and thousands of clients);
- transparent support of heterogeneous platforms;
- consistent support of different services across every supported platform, and the possibility of co-relating them for analysis purposes.

This article presents a service and configuration management system that supports NFS configuration and monitoring on large and heterogeneous networks. The system is based on a formal model that describes hosts, network components and services in a generic and abstract way. This model allows the system to be extended to support several different services in a variety of platforms.

The next section will present the service configuration problem, the related work and how our solution extends the state of the art. Subsequently, we will show the process that describes an NFS service

configuration and management life-cycle. The next section will present the system architecture. Then we will demonstrate the generalization of our solution for the other network services. Finally, we will present our conclusions.

Service Configuration and Related Work

Typically, the configuration of a service is composed of several interconnected tasks: planning, consistency checking, deployment, and management. In each task, a number of critical issues must be addressed, including:

- capacity of servers to offer the required service;
- the placement of the server with respect to the network topology, to avoid network performance problems and bottlenecks;
- dependency relation among hosts, with respect to different services, to avoid many single points of failure on the network;
- configuration consistency on all servers and clients, for ease maintenance and crash recovery;
- documentation of the entire process to allow consistent and efficient administration.

These are only a few of the issues that must be addressed in the process of host and service configuration. Hardware vendor's specific tools, like the AdminTool from Sun Microsystems Inc. and Smit from IBM Inc., do not provide satisfactory solutions to deal with heterogeneous system.

According to Evard [20], "the general approach taken by the administrative community over this time period has been to develop a host cloning process and then to distribute updates directly to hosts from a central repository." In a large and heterogeneous network, this cloning process is not satisfactory because each machine has its own characteristics.

In most solutions, the central repository is a collection of ASCII files, like in LCFG [1], GeNUAdmin [2], Syslogd [4] and Fisk's system [5]. This leads to the problem of keeping the consistency and the integrity of the information. Some tools have changed this approach by using DBMSs, like UHA [14], Aurora [19], Finke's system [13].

The swatch [16] and pong [15] systems are designed to monitor the network and some services. However, neither addresses the complete life-cycle of service configuration and monitoring, and, therefore, does not support integrated service management.

None of the above cited approaches are based on a formal and generic conceptual model of the network and its services. Such a model is the basis of commercial tools like TME10([17] and Unicenter TNG([18], and are essential to support the inclusion of new services and to keep the overall integration of the system. These two market leaders offer extensive features to manage heterogeneous networks, but do not offer built in facilities for high level configuration management.

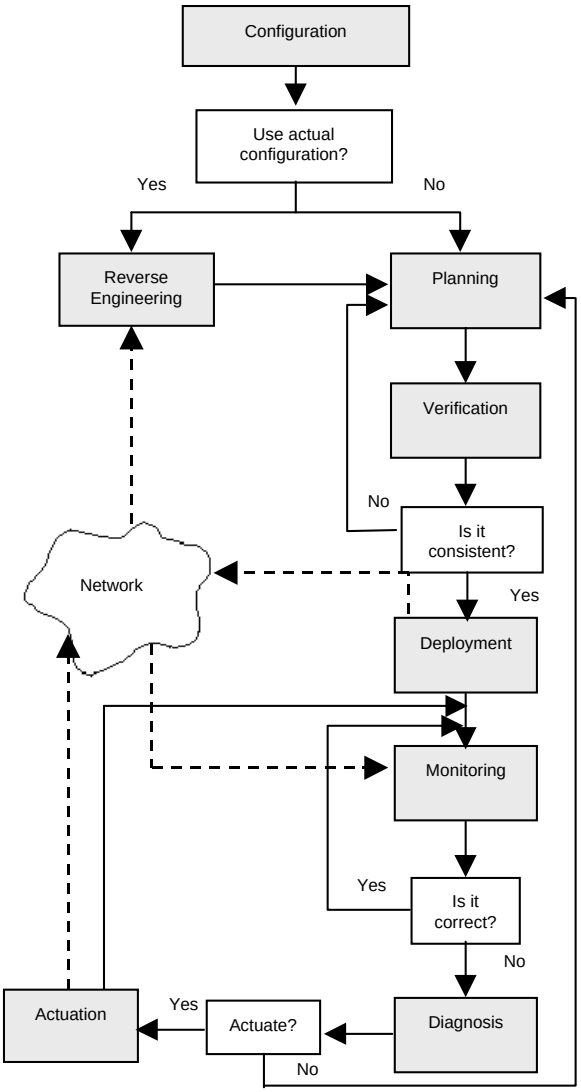

Figure 1: The Process.

In this paper we present a solution dedicated to address the problem of configuring and managing network services in heterogeneous environments. This system presents the following features:

- it allows service configuration to be carried out in an abstracted away from hardware and operating system details;
- service configuration is performed on a central database and is distributed automatically over the network. This enforces consistency and uniformity of the configurations on all hosts that use that service;
- since configuration distribution is performed automatically, the system scales up to deal with large number of hosts;
- it allows new services and platforms to be easily added;
- it supports reconfiguration, service dependency inference and analysis, performance analysis and configuration optimization in a single framework;

- it supports service monitoring, trouble shooting and problem correction.

An NFS Configuration and Management Process

The configuration and management of the NFS service follows a process that is common to most services, and is graphically depicted in Figure 1. Following this process, the administrator should perform a number of tasks in a coherent and consistent form:

- **Planning**: to define the filesystems exported by the servers and imported by the clients, as well as their access and security characteristics. This task should abstract away from platform specific features. In most cases, a service plan is constructed from the configuration information currently in use on the network. To extract this information, a Reverse Engineering stage is necessary. Therefore, the Planning stage in the process can start either with fresh configuration information or with information extracted from the network.
- **Consistency checking**: the planning of the service must be carefully checked for consistency. For instance, in large networks it is fairly easy for the system administrator to lose control over the import and export relationship among hosts. If this happens, servers may ending up exporting filesystems that are not used by any client and clients may try to import filesystems that are not exported by the server. Moreover, clients may try to mount a read-write filesystem that is exported as read-only, leading to an error.
- **Deployment**: the deployment of a service configuration is composed of three sub-phases:
  - **Generation of configuration files**: from a consistent service plan, the NFS export and import files should be generated. At this stage, platform specific features must be used to create files on the right format for each supported operating system and hardware platform.
  - **Configuration propagation**: the export and import files must then be distributed to the corresponding hosts.
  - **Configuration activation**: the necessary actions must be performed on each host to activate the new configuration. This involves rebooting the host.
- **Management**: he service must be monitored and controlled during execution, and problems must be identified and corrected. During the Management sub-process, current service behavior is compared to the planned behavior described in the configuration specification. Any deviation is detected and actions are fired to bring the service back to normal operation. If necessary, system administrator is notified for action.

The Architecture of the System

The NFS Configuration and Management System was implemented according to the architecture showed in Figure 2. This architecture was presented by Franklin in [11].

The architecture implements the process showed in Figure 1, which is based in the following stages: planning, deployment, monitoring, diagnosis and actuation.

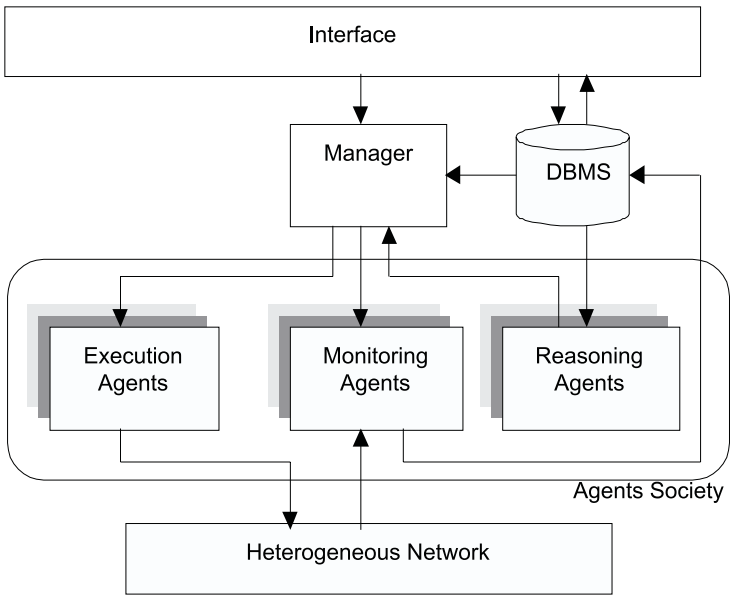

Figure 2: The Architecture.

The architecture has four components: the DBMS, the Agents Society, the Manager and the Interface.

The DBMS

The DBMS is the central element of the architecture and it is partitioned into two components:

- The Configuration Component: this is the implementation of the conceptual model described later on. It stores the NFS configuration definition and other information about its behavior in the network. The information resulting from the planning stage is stored in the DBMS using the Interface.
- The Monitoring Component: this has two functionalities: (1) it mirrors the configuration component, keeping monitoring information obtained from the network; (2) it supports the monitoring process, keeping monitoring parameters like, for instance, monitoring time intervals and agents status. The information is updated in the monitoring component by the monitoring agents, either as a result of the monitoring or the reverse engineering processes. This component is fully described in [21].

The configuration and monitoring components of the database have been implemented in ORACLE RDBMS.

Agents Society

The **Agents Society**, based on the intelligent agent paradigm described in [12], is composed of three types of agents, with complementary functionalities:

- The **Monitoring Agents** perceive the environment and collect information about the actual behavior of the NFS service on the network. There are two kinds of monitoring agents for NFS: one that searches configuration files and another that verifies the running daemons. These agents travel from the monitoring host to the target hosts, execute the scripts that collect information, receive the result from the scripts and deliver to the manager. The latter feeds the information to the monitoring database. For example, they collect information about which filesystems are imported and exported by the hosts. The monitoring agents are responsible for the Monitoring Stage in Figure 1.
- The **Reasoning Agents** compare the actual behavior of the service and the planned behavior stored in the DBMS, looking for inconsistencies. Based on predefined rules, these agents infer the actions that must be executed to correct possible problems. For example, they verify whether the filesystems imported by the client host are correctly exported by the corresponding server host. Currently, the agents provide the system administrator with a possible action, and he/she is responsible for its execution. Examples of some of the rules used by the reasoning agents are presented below:
  - If imported filesystems are not correctly exported by the corresponding servers, then either servers or clients must be reconfigured.
  - If the server's daemons are not running, then start them.
  - If there is a difference between the configuration and monitoring databases, then the system must be reconfigured in order to maintain the consistency between network and planning.
  - If a client can't mount a filesystem, then verify the server's status and inform the administrator.

  These informal rules are coded into the system as sets of formal rules that may activate one or more actions.
  The reasoning agents implement the diagnosis stage in Figure 1. The automatic execution of actions by the agents (without system administration intervention) and learning capabilities are under development using the IBM ABE [22] and the knowledge base construction language KIF [23].
- The **Execution Agents** perform actions on the network. That is, they change the configuration on the hosts in the network to eliminate a problem in the NFS functioning, according to what was defined by the Diagnosis Agent, or only to modify the actual configuration of the NFS service. For example, it corrects the inconsistency between the imported and exported files from clients and servers. This agent implements the deployment and actuation stages in Figure 1.

The agents have been implemented in Java, using the IBM Aglets Workbench API [24]. This API supports the construction of mobile multi-agents systems.

The Manager

The **Manager** controls the agents and interaction process between the architecture modules, activating, deactivating and creating the agents when necessary. When a set of agents is activated, it is not necessary to wait for all agents to finish their tasks to record information on the database. The manager records incomplete information and controls the time-out interval (defined by the system administrator) of all running agents. Incomplete information is dealt with by the reasoning agents. The manager has been implemented using the same technology as the agents.

Interface

The **Interface** allows the interaction between the system and the administrators. It provides the following functionalities: the DBMS front-end, visualization of all management processes, the NFS status in the network and visualization of the service planning.

The system interface has been developed using HTML embedded into ORACLE PL/SQL stored procedures. The use of HTML pages allows greater portability and easy of access through any web browser. Furthermore, the use of stored procedures facilitates the interaction with the database. However, it makes the interface dependent on the ORACLE database. A more portable solution using JAVA and ServLets to communicate with the DBMS is under construction.

The planning and verification stages are also implemented in PL/SQL and accessed directly through the interface. The remaining stages are implemented by the agents society, as described above.

Communication Protocols

A set of communication protocols are needed: between the agents of the same category, between agents of distinct categories, between the agents and

Figure 3: NFS configuration management systems interface.

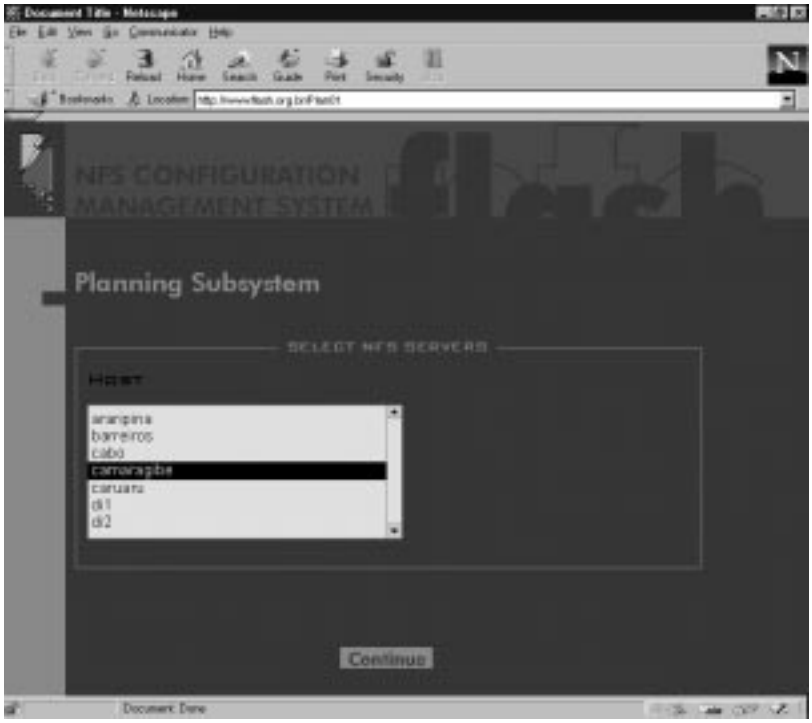

Figure 4: Planning subsystem.

the DBMS, between the agents and the network components. The latter implements the propagation and activation phases of the process described in Section III and is the subject of another paper [9].

The Use of the System

The navigational model follows the cycle defined in Figure 1. Each stage has been implemented as a separate subsystem, and are reached through the first page of the interface, as shown in Figure 3.

Planning and Validation Subsystems

This subsystem allows the definition of NFS clients and servers. Only hosts that can be servers appear in the list box of Figure 4 (e.g., those that have disks). Similarly for clients.

The following step is to configure the selected server and client hosts. As shown in Figure 5, once a host running Solaris 2.5.1 is selected, only the platform specific parameters for this version of the operating system appear in the configuration interface.

Once the NFS planning is finished, that is, all servers and clients are selected and configured, it is necessary to check and validate the configuration. This checking is perform by the validation subsystem, which is a PL/SQL procedure that either requests the system administrator to modify any inconsistencies found in the configuration or stores the configuration in the database if it is correct.

Deployment and Actuation Subsystems

To deploy or modify an NFS configuration, the manager verifies what needs to be updated (a deployment or modification may only affects a subset of the hosts) and schedules the process. The start of the process can be immediate or in a pre-schedule time defined by the administrator, as shown in Figure 6. At the deployment time, the agents and the configuration protocol are activated.

Monitoring Subsystem

In this subsystem, the manager queries the monitoring database looking for monitoring information. The system administrator tells the manager, through the interface, which agents need to be activated, which hosts to monitor and the time intervals of the monitoring processes, as shown in Figure 7.

At the defined time, the manager activates the monitoring agents, which travel to the target hosts and start the monitoring scripts, as discussed above. Once the monitoring process is finished the manager inserts the information in the monitoring database.

Diagnosis Subsystem

The diagnosis is also started through the manager, by a system administrator's request. However, it can also be automatically triggered every time new information from the monitoring process is stored in the database.

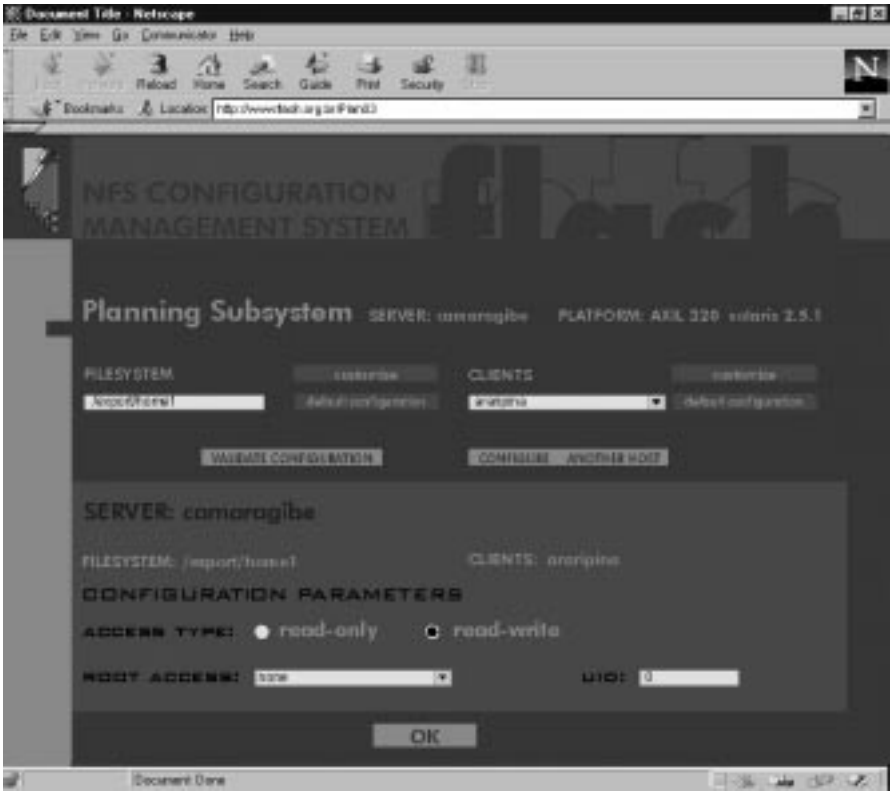

Figure 5: Planning subsystem – second interface.

The manager starts the diagnosis agents which will compare the configuration and monitoring database, looking for possible inconsistencies, and will notify the manager with two possible results:

- an OK sign, indicating that no inconsistencies have been found;
- the failure points and possible solutions and actions to bring the network back to a consistent state.

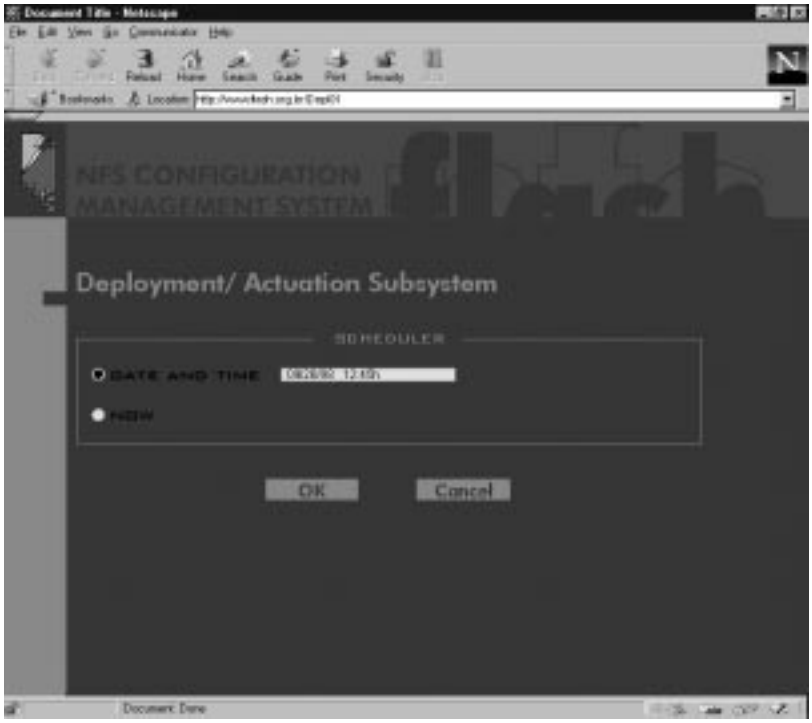

Figure 6: Deployment/actuation subsystem.

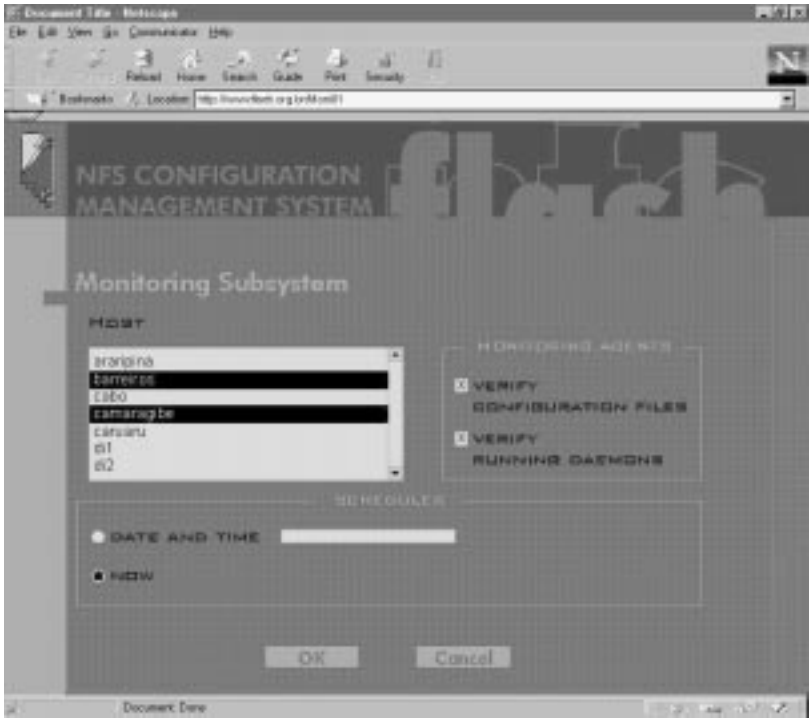

Figure 7: Monitoring subsystem.

The manager is responsible to format the result and send messages to the system administrator.

Reverse Engineering Subsystem

This subsystem is implemented by the monitoring agents already defined and by specific agents capable of collecting more detailed information about the network hosts. The reverse engineering is similar to the monitoring process, but in this case the collected information about the network will be stored in the configuration database. This process has to be authorized by the system administrator, since it changes the network configuration.

Generalization

The generalization depends on a conceptual model, specially designed to support all the necessary components and characteristics to configure and manage services in an heterogeneous network. This model, described in [10], uses object oriented paradigm that allows abstract descriptions of hosts, platform, services and configuration parameters, and dependencies among them. In this model, it is possible to construct a platform independent specification of a given service, which when combined with the specification of a given platform, produces the instantiation of the service for that platform. The inclusion of a new service is easily performed only by class instantiation.

The model showed in Figure 8 has seven classes, described as follows.

The class **Service** contains the description of the service (e.g., filesystem sharing services). This class is useful when there are several products that implements the same service. In this case, each implementation is modeled as a separate product and all versions belong to the same service.

The class **Product** contains product specific information, like, for instance, version and supported platforms (e.g., SunOS 4.1 version of NFS). This class has a self relationship permitting the dependency representation among services.

Each instance of the class **Parameter** represents a configuration parameter (e.g., filesystem access type). New parameters can be added through class instantiation. Each instance of the class **Product** refers to one or more instances of the class **Parameter**, therefore representing the information that is necessary to install product of a given service on the network hosts.

Each configuration parameter associated with a service has a set of restrictions. Different implementations of a service for distinct platforms have different set of restrictions. The restrictions are captured by the class **Parameter Restriction**, where each parameter connected to a service and a platform has well defined rules for its correct use during the configuration process (e.g., the filesystem access type parameter can only assume the values ***ro*** – *read-only* or ***rw*** – *read-write*).

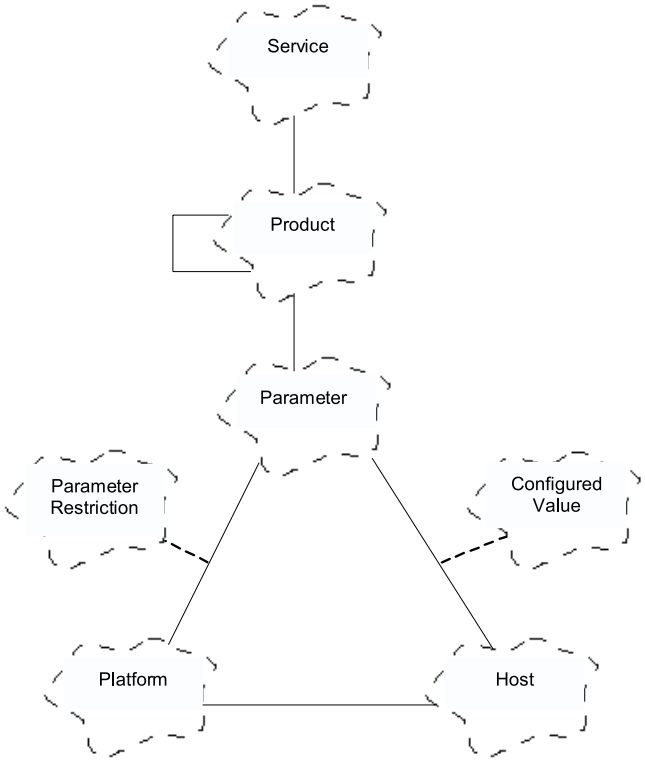

Figure 8: The conceptual model.

The class **Configured Value** holds the actual configuration values for each parameter of a given host (e.g., on host *host1*, the filesystem access type parameter has value ***ro*** to *filesystem1*). The class **Host** represents all hosts of a computer network and each instance of this class represents an actual host. (e.g., *host1*).

The class **Platform** captures the heterogeneous features of the hardware and operating system platforms on the networks (e.g., host IBM RS/6000 running AIX 4.1). A new platform can be easily added through class instantiation.

This model is independent of any implementation decision. So, it can be implemented in ASCII files or in any desired DBMS. In our case, we used ORACLE DBMS.

This object-oriented conceptual model specifies the database that holds network configuration information, it represents the DBMS component of the architecture showed on Figure 2. From this database, the configuration information is distributed to every host on the network using a distribution protocol. This protocol is described in [9].

Some features offered by this model are listed as follows:

- network service configuration planning;
- support to include new services and related network components;
- support of heterogeneous network;
- consistent and integrated view of the network;
- creation, alteration and visualization of host configuration even if it is down;
- visualization of the service topology, that is, the relations between servers and clients of the same service, and also the visualization and analysis of the services dependency graph, showing the relations among services;
- support to the monitoring and diagnosis phases to avoid failures, redundancies, inefficiencies and inconsistencies.

Conclusions

The service configuration and management system presented in this article has several important and desirable features:

- Consistency and uniformity of the configurations: achieved through the verification of configuration information stored in the database, according to rules also coded in the system.
- Automatic host and service reconfiguration: enforced by the configuration stored in the database that stays available even when the host is down.
- Scalability to deal with large networks: accomplished by the possibility of configuration replication and automatic propagation of configuration information to an arbitrary number of target hosts.
- Easy inclusion of new services and platforms: supported by the capability of defining meta-level configuration information in the database, as described in the previous section.

The solution presented in this article, uses several technologies from the design and implementation of a complex system, capable of support the planning, configuration and pro-active management of services in heterogeneous networks. The NFS prototype was used to validate the process, architecture and conceptual model.

The possibility of supporting other services, coherently integrated in the framework, is a clear advantage of this solution when compared to managing the services isolated and by hand. Currently, HTTP, NIS and DNS services are being included in the system. This will allow not only the configuration management of these systems separately, but also the correlation of information among all supported services.

Acknowledgements

The FLASH project is co-funded by the Brazilian Government agency CNPq, through the ProTeM-CC Program (Phase III) and by the Center for Advanced Studies and Systems at Recife (CESAR).

Availability

The prototype implementation, together with a configuration distribution system [9], has been used to assist the administration of laboratories of the Department of Informatics, UFPE. However, the system will only be available for distribution after the inclusion of the above mentioned services.

Author Information

Fabio Q. B. da Silva is an Associated Professor of the Department of Informatics at the Federal University of Pernambuco, Brazil, where he coordinates the FLASH Project. He holds a Ph.D. in Computer Science from the University of Edinburg, Scotland. He is also the Finance Director of the Center for Advanced Studies and Systems at Recife (CESAR), a not-for-profit organization dedicated to promote Industry/University interaction (http://www.cesar.org.br). Reach him electronically at fabio@di.ufpe.br.

Juliana Silva da Cunha is a Phd Student of the Department of Informatics at the Federal University of Pernambuco, Brazil and a member of The FLASH Team. She holds a master degree in Computer Science at Federal University of Ceará, Brazil. You can reach Juliana electronically at jsc@di.ufpe.br.

Danielle M. Franklin is currently a member of The FLASH Team and a system management consultant at CESAR. She holds a master degree in Computer Science. You can reach Danielle at dmf@di.ufpe.br.

Luciana S. Varejao is currently a master student in Federal University of Pernambuco's Infomatics Department and a member of The FLASH Team. She holds a bachelors degree in Electronic Engineering. You can reach Luciana at lsv@di.ufpe.br.

Rosalie Belian is a member of The FLASH Team. She holds a Master Degree in Computer Science. You can reach Rosalie electronically at rbb@di.ufpe.br.

References

[1] Paul Anderson, ''Towards a High-Level Machine Configuration System,'' *USENIX Systems Administration (LISA VIII) Conference Proceedings*, 1994.

[2] Magnus Harlander, ''Heterogeneous Unix Environment: GeNUAdmin,'' *USENIX Systems Administration (LISA VIII) Conference Proceedings*, 1994.

[3] John Rouillard and Richard Martim, ''Config: A Mechanism for Installing and Tracking System Configurations,'' *USENIX Systems Administration (LISA VIII) Conference Proceedings*, 1994.

[4] Rex Walters, ''Tracking Hardware Configuration in a Heterogeneous Network with syslogd,'' *USENIX Systems Administration (LISA IX) Conference Proceedings*, 1995.

[5] Michael Fisk, ''Automating the Administration of Heterogeneous LANs,'' *USENIX Systems Administration (LISA X) Conference Proceedings*, 1996.

[6] Grady Booch, *Object-Oriented Analysis and Design With Applications, Second Edition*, Addison-Wesley, 1994.

[7] FLASH Project, <http://www.di.ufpe.br/~flash>

[8] Hal Stern, *Managing NIS and NFS*, O'Reilly & Associates Inc., 1991.

[9] Glêdson E. da Silveira and Fabio Q. B. da Silva, ''A Configuration Distributed System for Heterogeneous Networks,'' *USENIX Systems Administration (LISA XII) Conference*, 1998.

[10] Juliana Silva da Cunha, Glêdson E. da Silveira, Fabio Q. B. da Silva, J. Neuman de Souza, ''An Object-Oriented Service Configuration Management System,'' *International Conference on Telecommunication (ICT-98)*, Chalkidiki, Greece, June 1998.

[11] Danielle Franklin. *I-DREAM: an Intranet baseD REsource and Application Monitoring system*. Master Degree Thesis, Federal University of Pernambuco, 1997.

[12] Michel Wooldridge, Nicholas R. Jennings, ''Intelligent Agents: Theory and Practice,'' *Knowledge Engineering Review*, Cambridge University Press, 1995.

[13] Jon Finke, ''Automation of Site Configuration Management,'' *USENIX Systems Administration (LISA XI) Conference Proceedings*, 1997.

[14] Gregory Thomas, James Schroeder, Merilee Orcutt, Desiree Johnson, Jeffrey Simmelink, John Moore, ''UNIX Host Administration in a Heterogeneous Distributed Computing Environment,'' *USENIX Systems Administration (LISA X) Conference Proceedings*, 1996.

[15] Helen Harrison, Mike Mitchell, Michael Shaddock, ''Pong: A flexible network services monitoring systems,'' *USENIX Systems Administration (LISA VIII) Conference Proceedings*, 1994.

[16] Stephen Hansen, E. Todd Atkins, ''Automated system monitoring and notification with swatch,'' *USENIX Systems Administration (LISA VII) Conference Proceedings*, 1993.

[17] Tivoli Systems Inc., http://www.tivoli.com .

[18] Computer Associates Inc., http://www.cai.com .

[19] Xev Gittler, W. Moore, J. Rambhaskar, ''Morgan Stanley's Aurora System: Design a Next Generation Global Production Unix Environment,'' *USENIX Systems Administration (LISA IX) Conference Proceedings*, 1995.

[20] Rémy Evard, ''An Analysis of UNIX System Configuration,'' USENIX Systems Administration (LISA XI) Conference Proceedings, 1997.

[21] Luciana S. Varejao. *Sistema de Monitoraçao para Redes Heterogêneas (A Monitoring System for Heterogeneous Networks)*. Master Degree Thesis (under development), Federal University of Pernambuco, 1998.

[22] Agent Building Environment, http://www.networking.ibm.com/iag/iagsoft.htm .

[23] Knowledge Interchange Format (KIF) http://logic.stanford.edu/kif/ .

[24] IBM Aglets Workbench Homepage http://www.trl.ibm.co.jp/aglets/ .