Administration of large distributed UNIX LANs with BONES

J. Schönwälder H. Langendörfer

Institute for Operating Systems and Computer Networks
TU Braunschweig
Bültenweg 74/75
D-3300 Braunschweig, Germany
E-mail: schoenw@ibr.cs.tu-bs.de
FAX: +49 531 391 5936

Abstract

Administration of large distributed UNIX LANs is a task that is not supported very well by the standard UNIX operating system. After a short review of some previous approaches we introduce a tool to maintain administrative information. We present an object-oriented data model that represents information not only tailored to replace standard UNIX configuration files.

1 Introduction

In the last years open environments based on the UNIX operating system have become more and more popular. The computing power of current computer technology permits the movement of applications from mainframes to decentralized workstation environments.

One problem with this evolution is that the operation and administration of decentralized environments becomes more and more complex. This is mainly due to the fact that the UNIX operating system was not designed to operate in large distributed environments. For example, user administration is normally done manually by updating some special text files. Administration tasks become even more complex if a system administrator tries to provide network transparency in a heterogeneous environment.

In this paper we present BONES, a tool for the administration of large UNIX LANs. BONES provides a central point of contact for administrative information based on an object-oriented data model. Configuration files are generated and distributed to the network nodes using the SMP protocol. After a short overview of related work, we present the architecture of BONES in section 3. Section 4 introduces the underlying data model and section 5 discusses the current implementation. We conclude in section 6 with a brief comparison of BONES with some related systems described in section 2.

2 Related Work

In the following paragraphs, we present a survey of some well known UNIX administration systems.

Network Information Services

Network Information Services (NIS) [17] from Sun Microsystems provides a distributed network lookup service that is primarily used to distribute the contents of UNIX administration files between a cluster of machines. Access to NIS maps is handled in the C library routines that normally access the standard unix administration files. Updates are made on a server storing the master maps and distributed to all slave server in a NIS domain. Members of a NIS domain pick the first responding server to look up NIS information. NIS is implemented on top of Sun Remote Procedure Calls (RPC).

There are some problems related to the NIS approach. First, you need a C library supporting NIS to make use of it. Integration of machines without NIS support is not possible. The second problem is the strict domain concept, e.g. there is no easy way to restrict an account to a subset of machines in a NIS domain. And since NIS does not use authenticated RPCs there is little security provided [6]. There are also no tools available to add or delete map entries without logging in to the NIS master machine.

Network Information Services Plus

In 1991 Sun Microsystems published a white paper about Network Information Service Plus (NIS+), the successor of NIS [7]. NIS+ is integrated in the Solaris environment and supports a hierarchical namespace. This allows decentralization of administrative information following an organizational hierarchy. There are also extensions to describe the schema of the data held in tables. Access to NIS+ information is authenticated. A fine grained authorization model can be used to grant read, modify, create and destroy privileges to tables, rows or columns.

While NIS+ solves some of the deficiencies of its predecessor, you still need a C library supporting NIS+ to make use of it. The data stored in NIS+ maps usually corresponds with classical UNIX administration files.

ACMAINT

ACMAINT [2] was developed to support account management on the Purdue University Computer Network. ACMAINT consists of three main parts: a database daemon handling the database, a set of front-end programs for users and system administrators, and transaction daemons which perform account management operations on the network hosts. The main features of ACMAINT are the minimal requirements on the machines controlled by ACMAINT. To integrate a new machine you only have to port the transaction daemon. The database itself is accessible over the network and implemented in ndbm format.

A major deficiency is that ACMAINT is limited to account management tasks and does not support the clustering of machines. The security of ACMAINT as described in [2] is low. Support of the Kerberos authentication service [8] was planned but not implemented in the publically available version.

Moira

Moira [15, 14, 9] was developed at MIT's Project Athena to provide a single point of contact for administrative information about the Athena environment. The heart of the Moira service management system is a relational database (RTI Ingres) storing all administrative information. On top of the database runs the Moira server which provides predefined transactions accessible through the GDB RPC protocol. Access control is handled using access control lists stored in the database and the Kerberos authentication service [8]. The Data Control Manager (DCM) distributes information to Hesiod name servers [4] which distribute administrative information to the client hosts of the Athena environment. Updates are done periodically and hence there might be a delay before changes (e.g. password changes) become effective.

While Moira has proven to handle the Athena environment, there are problems to use it in a different environment since Moira depends on a relational database and the Hesiod name service.

DME

The Open Software Foundation (OSF) is currently working on a Distributed Management Environment (DME) [1]. DME uses an object model to describe network devices and resources. An object server is responsible to store DME objects in persistent memory.

Management of the Distributed Computing Environment (DCE) is done using remote procedure calls and the network management protocols CMIP and SNMP that manipulate DME objects. Source code of the DME will be available to the members of the OSF at the end of 1993 [1].

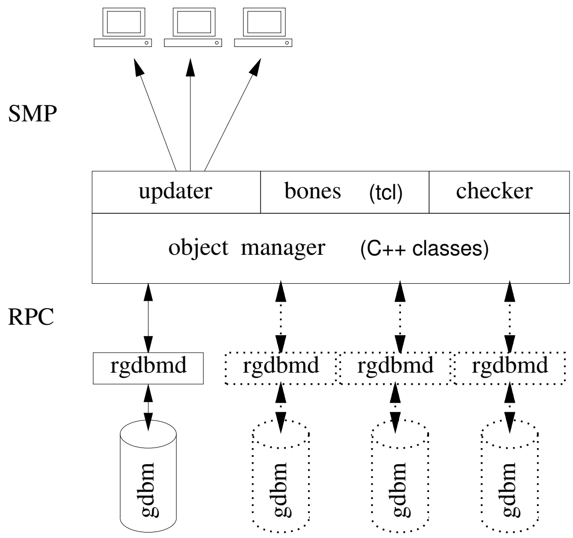

Figure 1: The architecture of BONES.

3 Architecture

The design of Bones was motivated by our analysis of existing systems. We have identified the following key issues that should a be addressed by administration environments:

- An administration tool should put few requirements on the hosts under its control.
- There is a strong need for a consistent data model that must be extensible to site specific needs.
- Changes to administrative information must be possible from every network node.
- An administration tool must take care of authenticated and encrypted communication.
- Data storage must be portable and independent from third party products like database systems.

The architecture of Bones is shown in figure 1. The main parts are the object manager, the programming interface, the updater and a consistency checker. The object manager handles objects stored in gdbm files. All access to the data files is done using SUN RPCs. This allows us to change the database from any host on the network. The object manager places read and write locks on object classes to allow concurrent access. Read and write access rights to object classes may be granted to user accounts.

The programming interface provided by the object manager is a library of C++ classes. Instances of these classes may be connected to objects stored in the gdbm files. A cache is used by the object manager to reduce the number of RPC calls to the gdbm server (rgdbmd). The C++ library is useful for writing applications that directly manipulate information stored by the object manager. We have written replacements for the UNIX programs that normally change administrative information (e.g. passwd or chsh). These programs immediately propagate the changes to the affected hosts.

There are two applications that periodically check the database. The program called checker tries to find inconsistencies in the database (e.g. accounts with the same user id). The updater looks for changes in the database that need to be transferred to client hosts. It maintains an update queue in order to keep track of hosts that are currently not reachable. Updates are distributed using the SMP protocol, an extension of the Simple Network Management Protocol (SNMP) providing authentication and data encryption [3, 5, 10].

Besides the C++ interface we have designed extensions to the tool command language (tcl) [11] that can be used to query and manipulate the information controlled by the object manager through an interactive shell. This script interface can be used to adapt Bones to site specific needs.

4 Data Model

BONES uses an object-oriented data model to represent information about users, projects, accounts, hardware, software and networks. The data model shown in Figures 2–5 contains more information than the UNIX administration file. We felt that is necessary to include information about projects, hardware configurations and installed software that is not needed by the UNIX system to get up and running since this information is quite often needed by the staff for the maintenance of large UNIX networks.

Before we discuss the details of our data model, some remarks on the notation. Each box in the following figures represents a class. The attributes of a class are listed below the class name. Key attributes are printed in bold face. Attributes referring to other objects have an arrow pointing to the referred class. Brackets at the start of an arrow indicate attributes that can refer to multiple objects. Inheritance is indicated by thick arrows representing the 'inherits from' relationship.

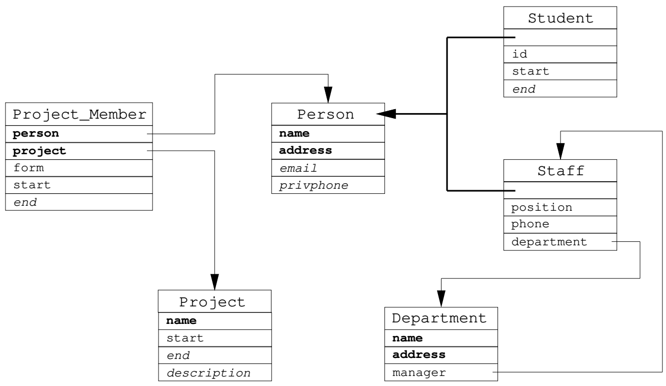

Figure 2: Part 'organization' of the data model.

Figure 2 shows the part of the data model describing persons, projects and departments. Class Person has two derived classes that provide additional information about students and the staff. Other derived classes could be used to represent additional user groups.

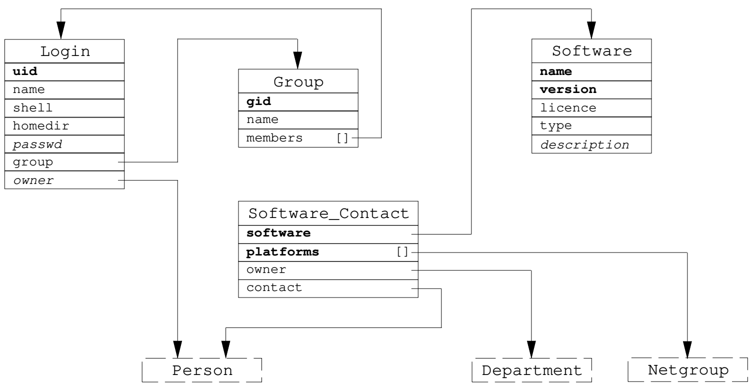

Figure 3: Part 'accounts and software' of the data model.

Accounts are represented by the classes Login and Group (figure 3). Note that a Login object refers to a Person object and does not contain any other user information. Figure 3 also contains two classes to store information about installed software and who maintains it.

Hardware is represented by the classes shown in figure 4. Class Device represents all boxes of a computing environment. We currently support the specializations Computer, Printer, Bridge and Disk. A Disk may have partitions with file systems. Another view on the hardware is provided by the class

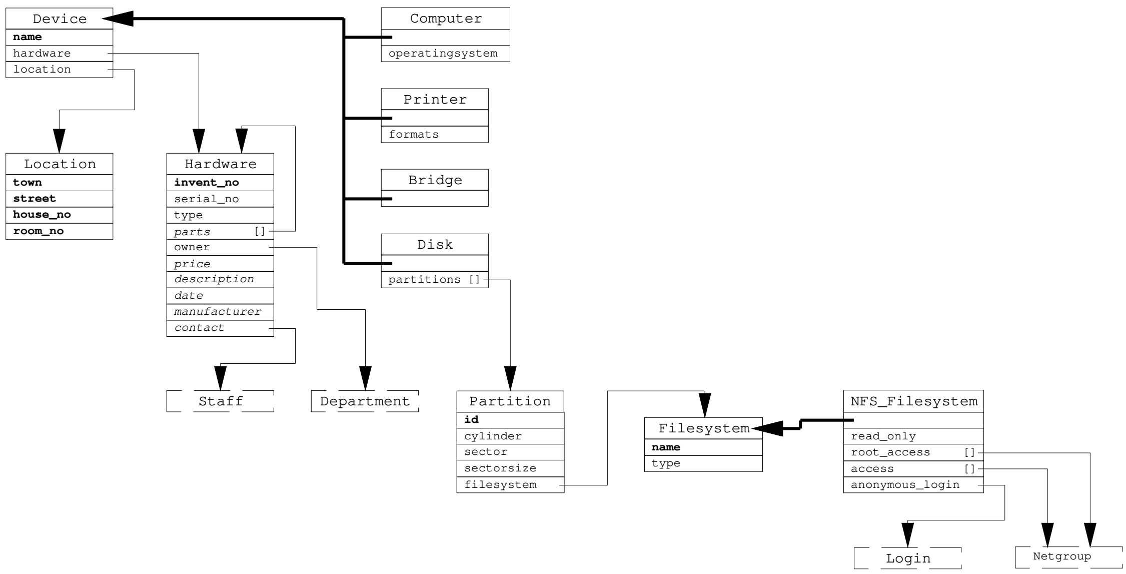

Figure 4: Part 'hardware' of the data model.

Hardware which describes a hierarchy of components that make up a Device. This information is often needed when a device is broken or when a system administrator contacts a hot-line.

Figure 5 shows the classes representing networks. We use three abstract classes for physical networks, logical networks and networks interfaces. Derived classes are used to model ethernets, IP networks and IP interfaces. Additional network types like Token Ring, FDDI or OSI networks can be integrated by deriving appropriate subclasses from the abstract classes.

Another important class shown in figure 5 is called Netgroup. A netgroup is a set of devices sharing either common accounts, IP host information, IP services or ethernet information. There are no restrictions on the composition of netgroups. The updater uses netgroups to determine what information must be transferred to which network node. Therefore it is possible to restrict accounts to a given set of hosts by defining an appropriate netgroup.

5 Implementation and future Improvements

Bones has been implemented in C++. The current version consists of an RPC server providing access to gdbm files, a consistency checker, an update daemon and the tcl-based frontend called bones. The updater uses the SMP implementation that has been done as part of the ISODE project [13].

The current implementation has the restriction that it can only act on a single object server. This will change in the future since independent departments are not willing to give away the control over their network configuration. Together with the support for multiple data servers we are working to enhance the security and access control of the RPC interface.

The update service is currently limited to account management since we use the UNIX Management Information Base (MIB) that comes with the ISODE SMP implementation. We are working on MIB extensions to allow the distribution of additional information.

We are also developing graphical user interfaces based on Bones tcl interface. We use the tk toolkit [12] for a simple database browser and we are integrating Bones into our graphical network editor called Ined [16].

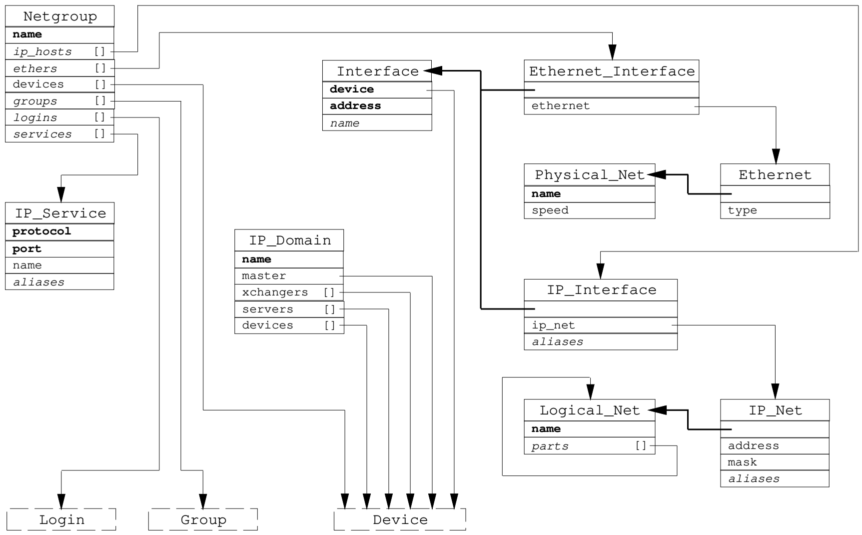

Figure 5: Part 'network' of the data model.

6 Conclusions

In this paper we have presented a tool for the administration of large UNIX LANs. The main features are a central database and an object-oriented data model representing a large set of administrative information in a uniform way. The SMP protocol is used to update network nodes without impacting the security of the network.

	NIS	NIS+	ACMAINT	MOIRA	BONES
Database	central	distributed	central	central	central
Data Model	unix	unix	relational	relational	object-oriented
Data Storage	dbm	dbm	ndbm	ingres	gdbm
Data Distribution	access	access	periodic	periodic	periodic
Authentication	—	SUN RPC	—	Kerberos	SMP

Table 1: Comparison of some administration tools

Table 1 summarizes the systems described in section 2 and compares them with Bones. The differences of these approaches are in the complexity of the database schema, the update protocol and frequency, the security mechanisms implemented and the user interfaces provided.

References

[1] M. Autrata, M. Santifaller, and H. Streppel. OSF's Distributed Management Environment. In *Proc. GUUG*, September 1992. (written in german).

[2] D.A. Curry, S.D. Kimery, K.C. DeLa Croix, and J.R. Schwab. ACMAINT: An Account Creation and Maintenance System for Distributed UNIX Systems. In *Proc. of Lisa IV*, October 1990.

[3] J. Davin, J. Galvin, and K. McCloghrie. SNMP Administrative Model. RFC 1351, MIT Laboratory for Computer Science, Trusted Information Systems, Hughes LAN Systems, July 1992.

[4] S.P. Dyer. The Hesiod Name Server. In *Proc. of USENIX Winter Conference*, pages 183–190, 1988.

[5] J. Galvin, K. McCloghrie, and J. Davin. SNMP Security Protocol. RFC 1352, Trusted Information Systems, Hughes LAN Systems, MIT Laboratory for Computer Science, July 1992.

[6] D.K. Hess, D.R. Safford, and U.W. Pooch. A Unix Network Protocol Security Study: Network Information Service. *Computer Communication Review*, 1992.

[7] Sun Microsystems Inc. Solaris ONC Network Information Service Plus (NIS+). Technical report, Sun Microsystems Inc., 1991.

[8] John T. Kohl and B. Clifford Neuman. The Kerberos Network Authentication Service V5. Internet-draft, MIT, September 1992.

[9] P. Levine, M.R. Gretzingen, M.A. Rosenstein, J.M. Diaz, B. Sommerfeld, and K. Raeburn. Moira, the Athena Service Management System. Technical Plan Section E.1, Massachusetts Institute of Technology, March 1990.

[10] K. McCloghrie, J. Davin, and J. Galvin. Definitions of Managed Objects for Administration of SNMP Parties. RFC 1353, Hughes LAN Systems, MIT Laboratory for Computer Science, Trusted Information Systems, July 1992.

[11] J.K. Ousterhout. TCL: An Embeddable Command Language. In *Proc. Winter USENIX Conference*, pages 133–146, 1990.

[12] J.K. Ousterhout. An X11 Toolkit Based on the TCL Language. In *Proc. Winter USENIX Conference*, pages 105–115, 1991.

[13] M.T. Rose, J.P. Onions, and C.J. Robbins. *The ISO Development Environment: User's Manual.* Performance Systems International, X-Tel Services Ltd, July 1991.

[14] M.A. Rosenstein. Moira User's Manual. Technical report, Massachusetts Institute of Technology, February 1991.

[15] M.A. Rosenstein, D.E. Geer, and P.J. Levine. The Athena Service Management System. In *Proc. of USENIX Winter Conference*, pages 203–212, February 1988.

[16] J. Schönwälder and H. Langendörfer. INED – An Application-Independent Network Editor. In *Proc. World Conference On Tools and Techniques for System Administration, Networking, and Security*, Arlington, (Virginia), April 1993.

[17] Sun Microsystems Inc. *System and Network Administration*, March 1990. Part Number 800-3805-10.