

# Assessment of DataBase Technologies – 2001 Vienna University of Technology's TUNETDB

Caroline Langer  
e9825633@student.tuwien.ac.at

Herbert Valerio Riedel  
e9725348@student.tuwien.ac.at

Reinhard Weiss  
e9826383@student.tuwien.ac.at

June 17, 2001\*

## **Abstract**

This paper analyzes the network administration database employed at the Vienna university of technology. While discussing the design flaws and showing workarounds, we'll come up with the need for a new and improved architecture.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>TU-Net Database—The misery of now</b>	<b>2</b>
2.1	Used software and hardware . . . . .	2
2.2	Data model . . . . .	2
2.3	Interaction with the database . . . . .	5
2.3.1	The email interface . . . . .	5
2.3.2	The web interface . . . . .	5
2.3.3	The command line interface . . . . .	7
2.4	Problems and flaws in design . . . . .	7
2.5	Suggestions for improvement and additional requirements . . . . .	9
<b>3</b>	<b>Searching for a solution</b>	<b>11</b>
3.1	Workarounds . . . . .	11
3.1.1	Frontends over existing email interface . . . . .	11
3.1.2	Frontends over existing web interface . . . . .	11
3.2	New approaches . . . . .	11
3.2.1	Relational SQL database . . . . .	11
3.2.2	Ganymede . . . . .	12
3.2.3	Directory Service . . . . .	13
3.3	Conclusion . . . . .	13
<b>4</b>	<b>Our directory service based model</b>	<b>14</b>
4.1	X.500 & LDAP Introduction . . . . .	14
4.2	Key features of Directory, i.e. LDAP based, Databases . . . . .	14
4.2.1	What's a Directory Service? . . . . .	14
4.2.2	How does LDAP work? . . . . .	15
4.2.3	Authentication . . . . .	15
4.3	Data Model . . . . .	16
4.4	Integration with existing infrastructure . . . . .	17
<b>5</b>	<b>Conclusion</b>	<b>18</b>

**List of Figures**

1	TU-Net Database – ER Diagram . . . . .	3
2	TU-Net Database – Workflow . . . . .	6
3	TU-Net Database – Searching for an object . . . . .	7
4	TU-Net Database – A selfexplaining Help-File . . . . .	9
5	Ganymede . . . . .	12
6	LDAP Tree for Data Model . . . . .	17

## 1 Introduction

To explain details later we first have to give a short overview about what is called the TUNET. It is a tool based on a programm called NETPLANE, that stands for NETwork PLANning Environment. It provides mechanisms for Network Documentation, Administration, and Planning.

### Network Documentation

- Single database
- Decentralised registration
- Data organised in layers

### Network Administration

- Status requests
- Process update-requests from users

### Network Planning

As this system is based on a database, it is possible to plan new networks, as soon as the objects are stored in the database even before they actually exist in physical.

We will give a short overview over the used data-modell, look at the different ways of using and accessing the TU-Net Database and try to find improvements and an alternative solution as the currently running system seems to be packed with some bugs.

Therefore we will first look for solutions based on the the currently runnig system such as another frontend over the existing email or web frontend.

As this does not seem to be a longterm solution, we then discuss different new approaches such as using GANYMEDE, redesigning and using SQL, and finally using LDAP. We will go into to details to explain our LDAP Model.

## 2 TU-Net Database—The misery of now

TU-Net Database is a central tool accessible through the Internet administered by the so called *Zentraler Informatik Dienst* developed to manage all computers connected to the Computer-Network of the Technical University of Vienna. It can be accessed by both, interested people and technical staff who have special rights like adding new objects to the Network or editing existing objects. Those requests will finally be processed by the ZID.

### 2.1 Used software and hardware

The system runs on a single host, `uhura.kom.tuwien.ac.at`, a dual SPARC 750 Mhz SMP machine.

- NETPLANE Software [6]
- Oracle DBS
- SunOS

### 2.2 Data model

Even extensive subnetworks can be registered. Basic elements are objects (with optional and non-optional attributes) and connections. It is possible to arrange the objects hierarchically (so called object-trees). The data structure used therefore consists of object and attributes, mapped to a relational database into 4 tables, namely

**OBJECT** objects with unique *ID* and some common attributes.

**ATTRIBUTE** additional attributes linked to tuples in OBJECT table.

**CONNECTION** table listing association between objects.

**PERSON** persons and organizations

A short explanation of the fields is to follow:

#### Fields that occur in all Objects

**REM:** any annotation.

**STAT:** state of the entry (e.g. Dirty, ReverseOnly, Hidden).

**ACL:** Access Control List, controls access to entries.

**LAST:** last modification date, is set automatically.

**PERS:** person who did the last modification, also set automatically.

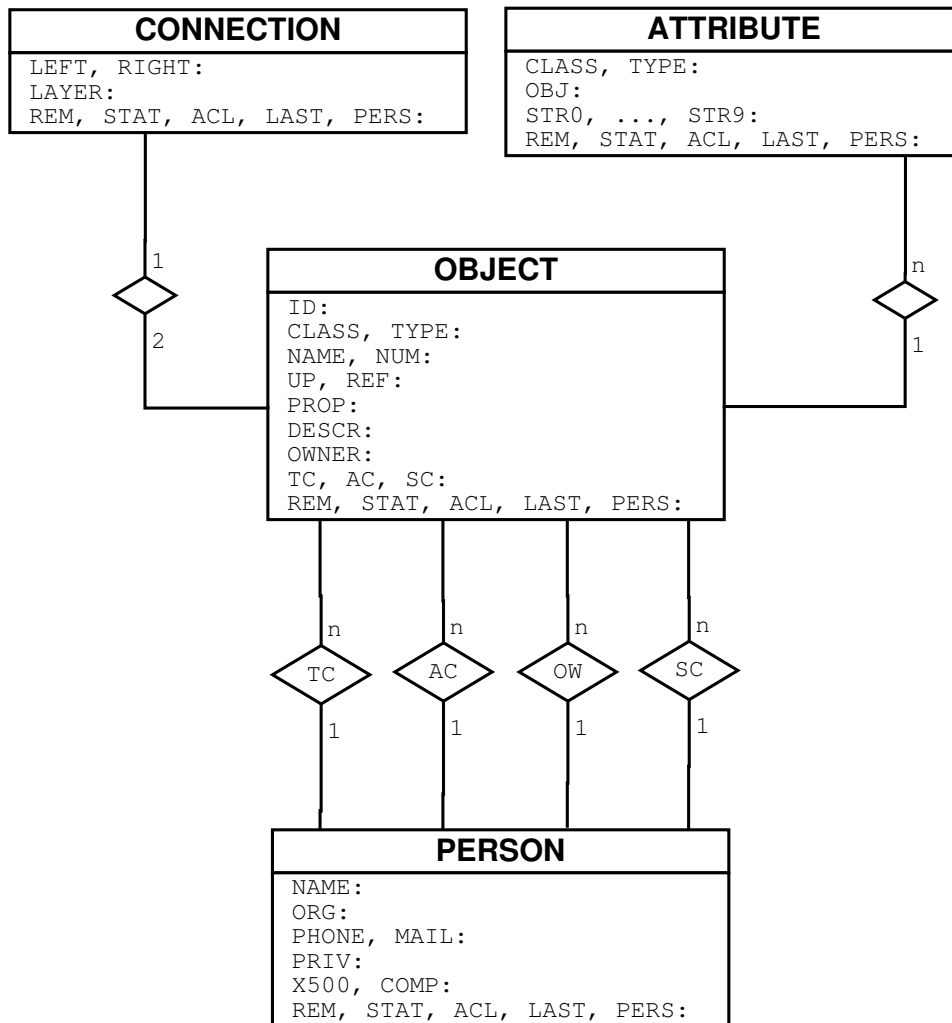


Figure 1: TU-Net Database – ER Diagram

**Object Table****ID:** unique.**CLASS, TYPE:** every object is classified through this pair, some examples are device/bridge, device/repeater, interface/10Base(Ethernet 10Mbit), cable/serial, etc.**NAME, NUM:** an object that does not have a father is called a Top-Level Object. Such objects must have a unique name, others either must have a unique name or a number that has to be unique within its sisters and brothers.**UP:** reference to the object's father, child is deleted when father is deleted.

**REF:** reference to some other object (some subnets to a main net).

**PROP:** some properties.

**DESCR:** detailed description of the object.

**OWNER:** proprietor of the object (e.g. number of the department).

**TC, AC, SC:** administrative, technical and service contact.

#### Attribute Table

**CLASS, TYPE:** every attribute is classified through this pair e.g. info/identification, info/protocol, ADDR/IP etc.

**OBJ:** reference to the object it is belonging to.

**STR0, . . . , STR9:** 10 information fields, utilization depends on. . .

**CLASS,TYPE**

#### Connection Table

**LEFT, RIGHT:** reference to the two endpoints of a connection.

**LAYER:** level of the connection.

#### Person Table

**NAME:** unique id.

**ORG:** description of the organisation (e.g. number of a department).

**PHONE, MAIL:** phone number and e-mail contact.

**X500, COMP:** X.500 Distinguished name, a list of of the number of departments this person is responsible for, these are also compared to the. . .

**OWNER** - field of the objects.

**PRIV:** state of privilege of the person's entry.

#### Efficiency of representation

It can be easily shown, that even common database queries will create high load due to the inadequate data model in use. For instance the simple query to display a 80-port switch with all its ports, which are represented as child objects, requires numerous joins. A quantitative calculation of the involved complexity, and thus the degree of *inscalability*, is left as an exercise to the reader.

## 2.3 Interaction with the database

The database is accessible either through a deprecated email interface or through a state of the art web interface. Also a command-line interface exists.

### 2.3.1 The email interface

The email way of causing changes to be done to the database involves writing an email, which must conform to a simple syntax, to the host-master email address. An example of the simple but hard to remember text based change request format is shown below.

```
To: hostmaster@noc.tuwien.ac.at
Subject: anmeldung NA
```

(n)

```
*na: NA,,Dummy.subd
*in: 1999-09-09
*de: Intel 286
*lo: r,HG0815
*ac: E605,Maier Max,456789
*ow: E605
*tc: E605,Mayer Karl,987654
*pt: IP

*if: realtek 10/100
*na: MAC,00-00-E8-8C-AF-1D
*na: IP,128.130.111.11, Dummy.subd.@
*cn: TP,TP-HG0815-09
```

When the email is delivered to the hostmaster's mailbox an automated reply with some ticket number is replied as an acknowledgment. The actual processing takes place at some undefined point of time.

### 2.3.2 The web interface

The web front-end<sup>1</sup> offers a more interactive way of changing the information contained within the database, by offering form fields to enter the data and search for network objects.

Users are organised in three different groups (Administrator, Technical Staff, Anonymus User), all with different permissions.

---

<sup>1</sup>which was originally designed for internal use only, but then modified to be more or less usable for public use

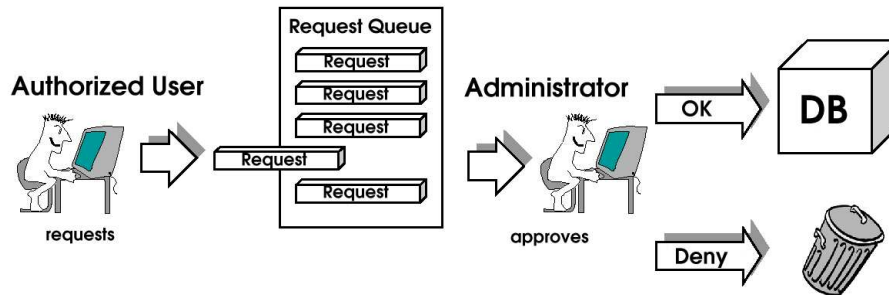


Figure 2: TU-Net Database – Workflow

**Anonymus User.** An anonymus user can only browse through the objects but has no rights to edit the TU-Net Database. Figure 3 on the following page shows the HTML-form for searching objects.

**Technical staff.** This people also can simply browse through the TU-Net Database, however their main task is to administrate their section's servers and workstations connected to the the TU-Computer-Network . The process of adding a new object or editing an existing object, like a server, to the TU-Computer-Network is divided into three steps:

1. Accessing the TU-Net Database. Herefore you must be entitled to enter the corresponding sites.
2. Add/ Edit an object. For adding new objects to the TU-Computer-Network in special you can work with so called *Templates* – that are objects already defined with special attributes. Creating new Templates is possible in the same way. For editing you must specify the object or browse for it. Normally each object connected to the TU-Computer-Network is marked with an unambiguous Identitycode.
3. Send the request. Changes are not reflected instantaneously, instead they are put into a request queue which can be viewed by the user. Sent requests could be undone in some very uncomfortable way by sending a new request. This queue should then be processed later by the hostmaster.

**ZID.** It seems that the same frontend it used to process the incoming requests. The hosdtmaster finally stores the data in the TU-Net Database or rejects it due to syntactic errors. Unfortunately this might take a few days.

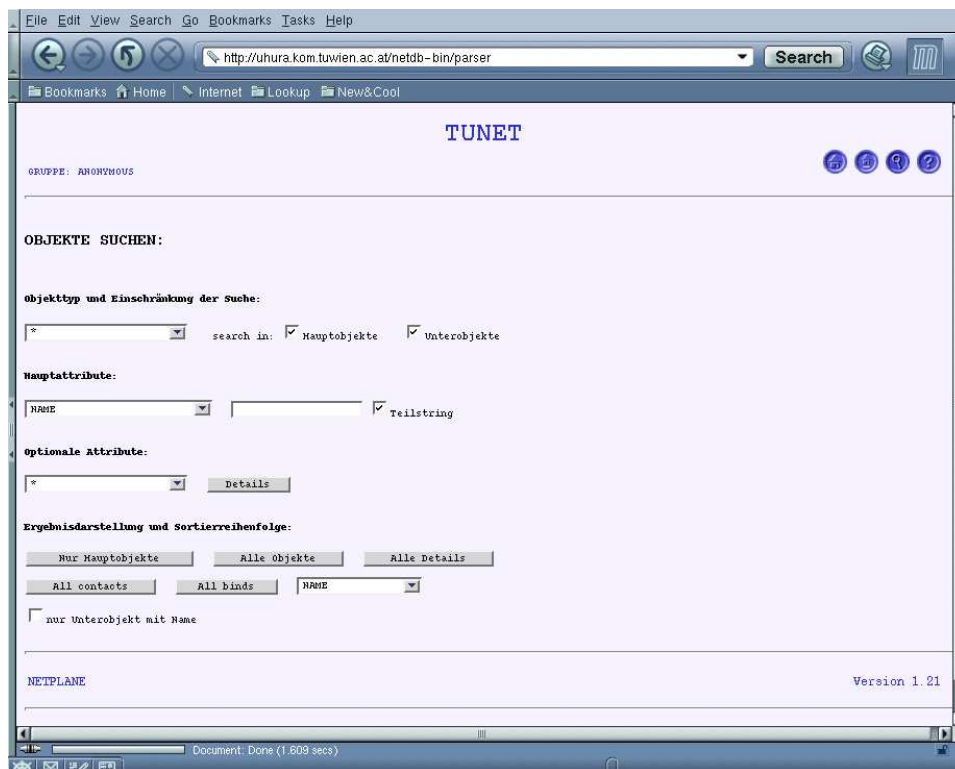


Figure 3: TU-Net Database – Searching for an object

## Templates

So called Templates are there to make the registration of new objects more easy. That are objects with special attributes that have been stored earlier and can be used now to register objects of same type or to create new structures.

### 2.3.3 The command line interface

According to [7], a command line interface exists, which allows to import from and export to a text based format. Sadly, this command line interface has no access control facilities, and as such can only operate with super user privileges. This interface is obviously used by administration internally only.

## 2.4 Problems and flaws in design

Some investigations and a survey taken amongst users has lead to the following incomplete list of criticized items in the actual implementation:

**No open interface.** The lack of an open unified interface causes severe interoperability problems, since there is no clean way to make use of the data contained in the database with custom components. This leads to inconsistencies due to independent data replication in individual databases.

**Inadequate web interface.** Due to the poor implementation of the web interface, which offers plenty examples of bad user interface design, the web interface can't be regarded as being user friendly.

**Invalid HTML code.** The web interface's HTML code is invalid, and thus causes problems with several web browsers.

**Error prone email interface syntax.** The email interface features a hard to remember syntax, which leads to errors, which are reported with unacceptable delays, due to the nature of the email delivery subsystem.

**Time wasting.** Both email and web interfaces cause wasting of time, due to high response times, especially for reporting invalid user entries.

**Lack of documentation.** Although provisions have been made for adding documentation, we have not been able to spot it, except for some little signs.

**Lack of security.** The exposure of scripting commands in the cgi forms is not very secure, since it allows to inject malicious commands directly into the scripting interpreter. The pitiful attempt to scramble the scripting commands is not regarded as a good security concept. Furthermore only weak authorization techniques are used.

**Inflexible.** Although the user is presented with many entry options—of which only a few are allowed—some less used but common settings<sup>2</sup> can't be set.

**Not scalable.** The used data model leads to a poorly scalable system.

**Proprietary.** Using a proprietary solution leads to dependencies with the software authors, which may become unavailable or lack the expertise to fix or implement needed features, i.e. proprietary solutions are not suitable as long term solutions. A free software solution would be a better choice, especially for a university, which can assign (pay-ed) development jobs to its own students, instead of wasting tax money with external entities.

---

<sup>2</sup>e.g. special DNS record types

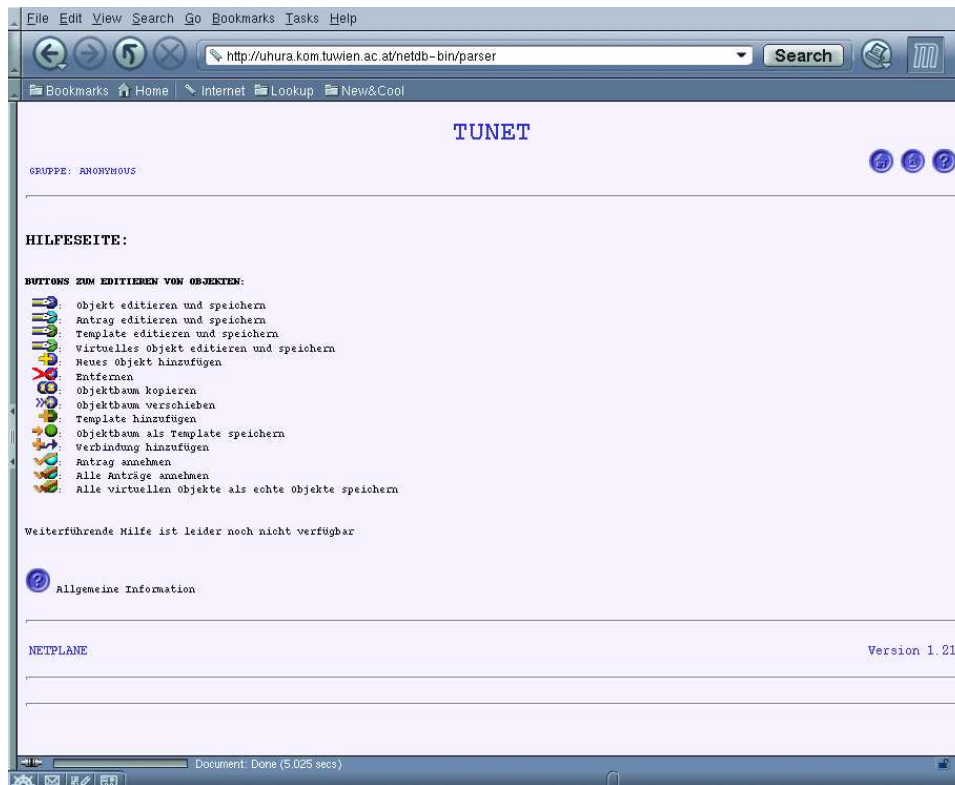


Figure 4: TU-Net Database – A selfexplaining Help-File

## 2.5 Suggestions for improvement and additional requirements

Some technicians would like to be more independent of the current system. They are willing to invest their time in developing an own interface. Some also suggested to program a new system. This could be realized by a payed practical for students. This small group (perhaps 2 people) should investigate the real needs for every user.

One of the current problems is that the system is not at all scalable. Every request has to be read and checked by a ZID-employee. We think that they should trust more the capabilities of the institutes. One counter-argument was that not every technician, especially in the non-computer-science institutes, is a computer expert. On the other hand side an self-explaining, well designed interface should correct mistakes and make the handling easier.

Staff cuts and reduction in volume and servicing will be possible when introducing a more effective system with open interfaces.

During our interviews some suggestions for improvement of the current system have arisen. The response time has been much too long for an effective working but has improved during our study when the hostmachine

was upgraded. Another point was that an user-friendly interface should be layed over the actual one, adapted to the needs of the different users.

The help page has been Under Construction for over two years and not been processed at all. Together with the not-at-all-self-explaining buttons which are by the way colour coded this has been the main reason of the technicians for not using the web-interface.

Security issues such as the possibility of deleting or creating an entry twice and more times or leakage of session handling or persitency issues should have been remodeled a long time before our survey.

## 3 Searching for a solution

Some possible approaches to overcome the shortcomings outlined in the previous chapter are described here.

### 3.1 Workarounds

Let's start with a discussion of workarounds for the existing implementation.

#### 3.1.1 Frontends over existing email interface

Since the format for emails is quite easy to produce by programs, it is possible to interact with the tunetdb in an automated fashion, which would allow for keeping the TU-Net Database in sync with almost no manual intervention required.

The lack of a proper machine parseable acknowledgment reply, makes this approach unreliable. Furthermore the lack of a way to query the database by email doesn't help either. Finally the administration seems to regard the use of the email interface as being deprecated.

#### 3.1.2 Frontends over existing web interface

This workaround would solve the problem of the email frontend workaround. By imitating the user accessing this web frontend<sup>3</sup>, a clean interface could be implemented, which wraps queries and change requests to scripting friendly language bindings.

The drawbacks of this proposal are a higher load on the web server and the high cost of programming and maintaining<sup>4</sup> such a wrapper.

### 3.2 New approaches

As shown before the use of workarounds does not constitute a viable solution, so it's becoming clear that a new architecture is required to try to solve the requirements.

#### 3.2.1 Relational SQL database

Not such a good idea, since there's no such thing as a protocol. It would require the use of CORBA/RMI/ODBC/XMLRPC which would be unnecessary overkill in our opinion.

---

<sup>3</sup>one could go a step further, and inject commands directly into the cgi script, by exploiting the bad design

<sup>4</sup>every change of the web interface would need to be reflected in the wrapper

### 3.2.2 Ganymede

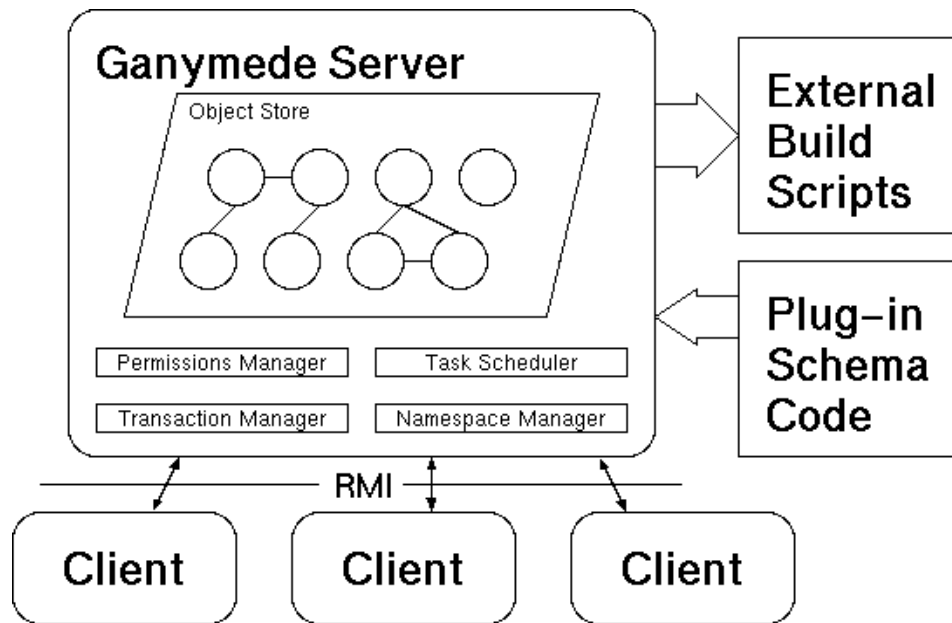


Figure 5: Ganymede

This application looks quite promising. Taken from the announcement of this package[10]:

Ganymede is a GPL'ed metadirectory system. Ganymede provides support for concurrent, team-based management of network directory services. It features a multithreaded database server with support for plug-in Java classes to customize the structure, management, and distribution of network directory data. Ganymede allows large groups of administrators to share administrative control over designated portions of a master network directory database, and provides transactional reliability and intelligent constraint management to keep network directories consistent. Ganymede keeps complete audit trails for all activity and can send email notification of relevant directory changes to every member of your admin team, keeping administrative teams coordinated and effective. Ganymede's sophisticated graphical user interface is designed to provide a high enough level of ease and safety of use to allow even relatively untrained users to make changes to your directory databases.

This approach seems to be the next best alternative after the directory service based one, presented later on.

### **3.2.3 Directory Service**

Due to the hierarchical structure of the data to be stored, the tree structure of directory based databases like LDAP is well suited for storage. Furthermore the LDAP protocol is well known and bindings exists for popular scripting languages like perl or python. Most requirements like scalability and redundancy are fulfilled.

### **3.3 Conclusion**

Of all possible solutions mentioned in this chapter, the last one looks most promising. Thus we'll go into detail into LDAP in the next chapter.

## 4 Our directory service based model

We are going to create a Lightweight Directory Access Protocol-Model ([1], [2]).

### 4.1 X.500 & LDAP Introduction

X.500 is an overall model for Directory Services in the OSI world. The model encompasses the overall namespace and the protocol for querying and updating it. The protocol is known as "DAP" (Directory Access Protocol). DAP runs over the OSI network protocol stack—that, combined with its very rich data model and operation set makes it quite "heavyweight". It is rather tough to implement a full-blown DAP client and have it "fit" on smaller computer systems. Thus, the folks at University of Michigan, with help from the ISODE Consortium, designed and developed. . .

LDAP, or "Lightweight Directory Access Protocol". LDAP is, like X.500, both an information model and a protocol for querying and manipulating it. LDAP's overall data and namespace model is essentially that of X.500. The major difference is that the LDAP protocol itself is designed to run directly over the TCP/IP stack, and it lacks some of the more esoteric DAP protocol functions.

A major part of X.500 is that it defines a global directory structure. It is essentially a directory web in much the same way that http & html are used to define & implement the global hypertext web. Anyone with an X.500 or LDAP client may peruse the global directory just as they can use a web browser to peruse the global Web. Additionally, with the help of web-to-X.500 gateways, you can use your favorite web browser to peruse both!

### 4.2 Key features of Directory, i.e. LDAP based, Databases

We now give an overview of some LDAP-features. You can find more about this topic in [9].

#### 4.2.1 What's a Directory Service?

A directory is like a database, but tends to contain more descriptive, attribute-based information. The information in a directory is generally read much more often than it is written. As a consequence, directories don't usually implement the complicated transaction or roll-back schemes that regular databases use for doing high-volume complex updates. Directory updates are typically simple all-or-nothing changes, if they are allowed at all.

Directories are tuned to give quick-response to high-volume lookup or search operations. They may have the ability to replicate information widely in order to increase availability and reliability, while reducing response time.

When directory information is replicated, temporary inconsistencies between the replicas may be OK, as long as they get in sync eventually.

There are many different ways to provide a directory service. Different methods allow different kinds of information to be stored in the directory, place different requirements on how that information can be referenced, queried and updated, how it is protected from unauthorized access, etc. Some directory services are local, providing service to a restricted context (e.g., the finger service on a single machine). Other services are global, providing service to a much broader context.

#### 4.2.2 How does LDAP work?

LDAP directory service is based on a client-server model. One or more LDAP servers contain the data making up the LDAP directory tree or LDAP backend database. An LDAP client connects to an LDAP server and asks it a question. The server responds with the answer, or with a pointer to where the client can get more information (typically, another LDAP server). No matter which LDAP server a client connects to, it sees the same view of the directory; a name presented to one LDAP server references the same entry it would at another LDAP server. This is an important feature of a global directory service, like LDAP.

#### 4.2.3 Authentication

To access the LDAP service, the LDAP client first must authenticate itself to the service. That is, it must tell the LDAP server who is going to be accessing the data so that the server can decide what the client is allowed to see and do. If the client authenticates successfully to the LDAP server, then when the server subsequently receives a request from the client, it will check whether the client is allowed to perform the request. This process is called access control.

In LDAP, authentication is supplied in the "bind" (login) operation. Ldapv3 supports three types of authentication: anonymous, simple and SASL authentication. A client that sends a LDAP request without doing a "bind" is treated as an anonymous client. Simple authentication consists of sending the LDAP server the fully qualified DN of the client (user) and the client's clear-text password. This mechanism has security problems because the password can be read from the network. To avoid exposing the password in this way, you can use the simple authentication mechanism within an encrypted channel (such as SSL), provided that this is supported by the LDAP server.

Finally, SASL is the Simple Authentication and Security Layer [3]. It specifies a challenge-response protocol in which data is exchanged between the client and the server for the purposes of authentication and establishment

of a security layer on which to carry out subsequent communication. By using SASL, LDAP can support any type of authentication agreed upon by the LDAP client and server.

Further on authenticating users to access information from your Directory Tree, your LDAP server can authenticate users from other services too (Sendmail, Login, Ftp, etc.). This is accomplished migrating specific user information to your LDAP server and using a mechanism called PAM (Pluggable Authentication Module).

### 4.3 Data Model

The proposed data model maps the physical topology to the ldap tree structure and takes advantage of the existing white pages directory service for references to contact entities. LDAP allows for hierarchie based access control, which would suffice for the required granularity, when putting each insitute into it's own tree.

The following is an example of how a host, namely isync.se-linux.ifs.tuwien.ac.at, could be represented in LDIF ([4]):

```
dn: cn=isync, dc=se-linux, dc=ifs, dc=tuwien, dc=ac, dc=at
cn: isync
objectclass: hostinfo
rem: AMD K6
loc: cn=he0405, ou=Raeume, o=Technische Universitaet Wien, c=at
ac: cn=Trixl Horst, ou=Mitarbeiter, o=Technische Universitaet Wien, c=at
tc: cn=Trixl Horst, ou=Mitarbeiter, o=Technische Universitaet Wien, c=at
ow: cn=IFS, ou=Institute, o=Technische Universitaet Wien, c=at

dn: cn=eth0, cn=isync, dc=se-linux, dc=ifs, dc=tuwien, dc=ac, dc=at
cn: eth0
objectclass: hostnetdev
rem: RealTek 10/100
mac: 00-e0-7d-01-c7-aa
ip: 128.130.204.18
cname: mail
cname: cvs
cname: cvs.ifs.tuwien.ac.at.
cname: db
cname: ldap
conn: cn=tp-02, cn=he0405, ou=Raeume, o=Technische Universitaet Wien, c=at
```

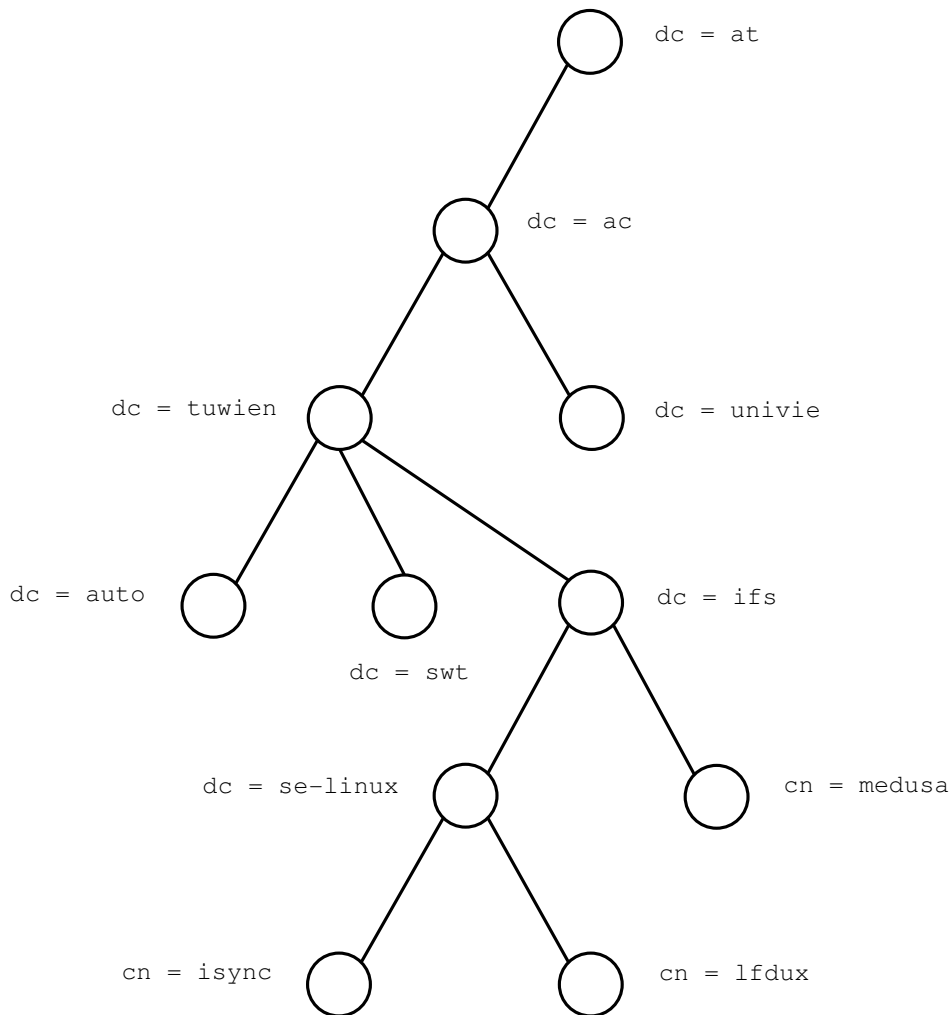


Figure 6: LDAP Tree for Data Model

#### 4.4 Integration with existing infrastructure

Since LDAP is already in use at the university of technology, e.g. for the white pages directory system, the integration would be seamless, and would in fact represent just an extension of the already used technology. Furthermore access control for file services and mail routing could be based natively on LDAP, since it's already common use to do so. The advantage over basing access control over DNS would be lower propagation times and a higher degree of security, since DNS is not regarded as being as reliable as LDAP is supposed to be.

## 5 Conclusion

We have shown up the obvious problems with the actual proprietary implementation in use and suggested different workarounds, of which all would have some kind of faults, so we analyzed different new approaches and described their advantages and finally we went into detail for the LDAP based solution, which seems to be perfectly suited for mapping the physical hierarchical world to the tree based representation of this directory service data model.

## References

- [1] W. Yeong, T. Howes and S. Kille, *Lightweight Directory Access Protocol*, RFC 1777, March 1995.
- [2] M. Wahl, S. Kille and T. Howes, *Lightweight Directory Access Protocol (v3)*, RFC 2251, December 1997.
- [3] J. Myers, *Simple Authentication and Security Layer (SASL)*, RFC 2222, October 1997.
- [4] G. Good, *The LDAP Data Interchange Format (LDIF)*, RFC 2849, June 2000.
- [5] *Directory Services Markup Language (DSML)*, <http://www.dsml.org/>, December 1999.
- [6] CoCo Software Engineering GmbH, *NETPLANE—The Network PLanning Environoment*, <http://www.coco.co.at/netplane/index.html>.
- [7] CoCo Software Engineering GmbH, *NETPLANE—The Network PLanning Environoment White Paper*, version 1.12, March 2000.
- [8] Martin G. Rathmayer, *TUNETDB Internal Memo*.
- [9] *Lightweight Directory Access Protocol HOWTO*, <http://www.linuxdoc.org/HOWTO/LDAP-HOWTO.html>, version 1.04, 28 February 2001.
- [10] *GANYMEDE*, <http://www.arlut.utexas.edu/gash2/>