The Apostle System Overview

Technical Report No. 88-001

Masayuki Ida[*] and Keisuke Tanaka[†]
Computer Science Research Lab.
Information Science Research Center

Aoyama Gakuin University
Address: 4-4-25 Shibuya, Shibuya-ku, Tokyo Japan 150

August 1988

[*] ida%aoyama.junet@relay.cs.net
[†] keisuke%cc.aoyama.junet@relay.cs.net

Abstract

The system *Apostle* is the kernel of the Trinity Initiative of Aoyama Gakuin University, aiming the integration of computer resources disparsed in three different campuses. The overview of the system *Apostle* is described. Its design and an implementation of us started from April 1988 by Computer Science Research lab. which was established at the same time under the Information Science Research Center.

The harmonic connection concept employed to design *Apostle* is the concept to build a distributed environment among widely interconnected computers or local area networks in one organization. The characteristics of the harmonic connection is to provide one virtual distributed OS with moderate comminication lines.

Local Area Network of each campus is connected with each other, and one loosely coupled distributed computing environment is to be provided. In this environment, traffics on the link for inter-campus connections are controlled to low. The dynamic linking feature of Sun-4 4.0 assists to reduce traffic amount. The major components of *Apostle* are the information server, the user interface, and the file cacher. The machines involeved are a super computer, several SUN-4s, terminal servers, interfaces for personal computers, a S3620 and others. *Apostle* also acts as a backbone for each LAN of several sections.

1 The Trinity Initiative – the needs behind the Apostle

1.1 The Environment

The Trinity Initiative of Aoyama Gakuin University (AGU) started in April 1988 along with the birth of Computer Science Research Laboratory (CSRL). The Trinity Initiative is a project inside the university to tie up the computing facilities which have dispersed among campuses.

Aoyama Gakuin has 118 years history and has divisions from kindergarten to university with graduate schools. The institution has as its objective a coherent education dedicated to the spirit of the Methodist Church of the Christian faith. AGU has six colleges and has three campuses, that is, Aoyama campus, Setagaya campus, and Atsugi campus. We have about 19,000 students. 10,000 students belong to Aoyama, 2,000 to Setagaya, and 7,000 to Atsugi.

Fig. 1 shows the physical locations of these three campuses. The Aoyama campus has the institution's headquater and is located in the central part of Tokyo. The Setagaya campus is for the college of Science and Engineering except for freshmen. The Atsugi campus was built in 1982. Freshmen students of all other colleges and schools other than that of Sci. and Engi. study for two years at Atsugi and move to Aoyama. All evening division and graduate division students except those of Sci. and Engi. study exclusively on the Aoyama campus.

The distance between Aoyama and Setagaya is 13 km (8 miles), Aoyama and Atsugi is 45km (30 miles), and Setagaya and Atsugi is 35km (22 miles).

1.2 The Characteristics of the Trinity Initiative and its needs

Information Science Research Center (ISRC) is in charge of the whole computing facilities of AGU and has branches for each campus. At the Setagaya campus, the super computer SX-1EA is scheduled to be there in October 1988. The Atsugi campus has hundreds of personal computers for teaching freshmen and introductory courses. The Aoyama campus has the gateways for several networks including JUNET, and several commercial network services. Computer Science Research Lab. (CSRL) under ISRC is located in the Aoyama campus.

Fig. 2 shows the connection of the computing facilities. ISRC has experiences with the DDX-P connection and then high speed digital line connection for 5 years for hierarchical main frame TSS system, and N-1 link to University of Tokyo. We have been coped with several different networking requests. Namely, personal computer connection, main frame TSS connection and UNIX-oriented network. These types of connection should have equal services for the three campuses of us. The Trinity Initiative is to connect these three different *persona* with a uniform or a single network.

Fig. 3 shows the model of the Trinity Initiative. Trinity is the integration of three campuses. And, Trinity is the integration of three levels of computing facilities, that is, super-computer-TSS, workstations and personal computers.

Trinity = the integration of Aoyama, Setagaya, and Atsugi campus

Trinity = the integration of supercomputer TSS system, workstations, personal computers

As for the needs of Apostle, we can count the following real problem to solve. We need an uniform environment among the three campuses, since lots of users, i.e. the professors, must migrate among the campuses mainly for teaching. There is a data that 30% of some 500 permanent faculty members (from research associate level to full professor level) have their roles on two or three campuses. User wants to assume he is always in the same environment and needs to access the same file system of him. Since almost all of them are not computer science related professors, they need a simple , same and easy-to-use environment. They need mailing and data retrieving facility at first. To cope with this migration of persons, the simplest approach is to carry portable WS or PC by themselves. But it is not realistic solution and is incomplete solution. The best way is to have a tight connection of three LANs with very high speed lines (and provide adequate workstations everywhere). But it's not feasible for our case, since the total traffic seems to be not so high to afford expensive connection. This is the reason why we need the harmonic connection.

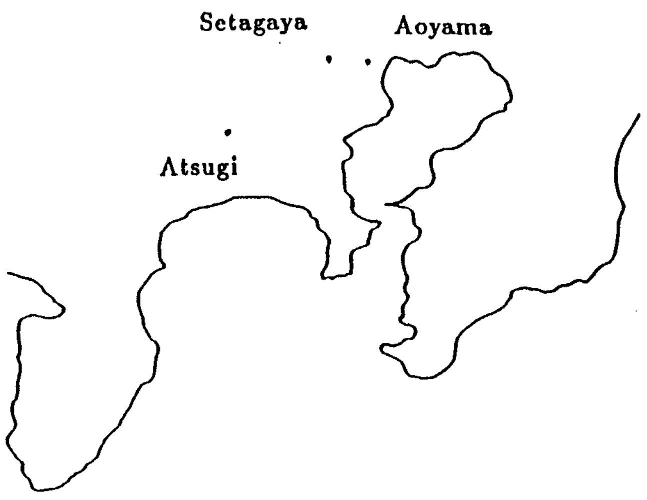

Figure 1: three campuses of Aoyama Gakuin University

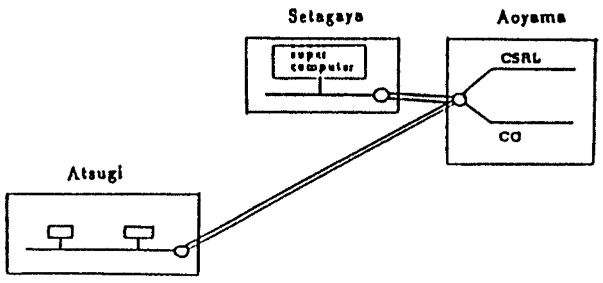

Figure 2: The connection of computers of three campuses

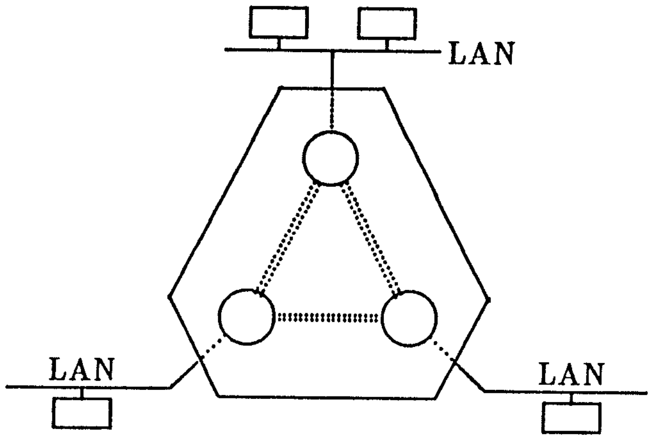

Figure 3: The Trinity model

2 The Harmonic Connection

– A loosely coupled distributed OS –

2.1 Distributed OS and Network Integration

In general, there are increasing needs to have distributed OS or network integration tool over the computers on one network. Distributed OS approach has the following characteristics:

1. computers/workstations on one network form one virtual machine/OS.

And as the results of (1),

2. resources are (or can be) shared among workstations.
3. it can have some load balancing mechanisms.

And for users,

4. it provides an uniform access to his environment. (he can log-in any machine on the network and can use the same environment with the same fashion.)
5. it can save/lessen the cost of maintenance/administration and the cost of training, since each machine has a same mechanisms.

These characteristics/merits are attained with the costs of
(a) traffic overhead on the network, which is usually high, and
(b) cpu/memory/disk resource consumption overhead on each machine.
We count Mach[Accetta86] as typical distributed OS. In our paper, we neglect the systems structures as in butterfly[BBN87] which are originally designed for parallel processing and are to be used as a whole. Mach is a distributed OS originally developed at CMU. It has a process migration mechanism. It has a portable design and implemented on several different machine architectures.

Meanwhile, network integration is mainly achieved with network file systems like NFS [SUN86a]. And with further assistance of YP[SUN86b], a system may act as one environment though each workstation is distinguishable. They are viewed as a connection facility which shares environments. To have an convenient system, NFS assumes an Ethernet.

2.2 A Definition of the Harmonic Connection

We are thinking of the connection of the machines whose physical locations are quite different but the organization to which they belong is the same and the machines are shareable.

We define the harmonic connection, our concept of integrated system of location independent computers, as a loosely coupled distributed OS.

The characteristics of the harmonic connection are;

1. It is for a distributed OS.
2. The whole component computers do not always reside in the same location.
3. A linking medium does not always be required a high bandwidth. Even if a medium is slow, the system should work as convenient as possible. Implementation should cope with variations of network media.

Table 1: A Comparison of several connection styles

	tightly coupled	loosely coupled	network integration	communication tool	mail/ message routing
Example	Mach	The harmonic connection	NFS/YP	rlogin, ftp, telnet	uucp
assumed media	dedicated bus, Ethernet	synchronous line, X.25	Ethernet	Ethernet	asynchronous line
traffic	very high	medium	high	low (occational)	low
connection	tight	medium	medium	loose	loose
act as a whole	Yes	Yes	No	No	No
assumed locations for each machine	the same cabinet, room,...	different site	the same Ethernet	the same Ethernet	different site
assumed organization	one	one	one	multiple	multiple

Table 1 shows the position of harmonic connection among several different connection scheme; tight coupling, network integration, communication, and message routing.

The harmonic connection is the principal idea in the *Apostle* system and is also a technology which is general enough to apply on a different organization. We intend to have a network for users whose accesses are locally dense. And the harmonic connection is suitable for the trinity initiative.

3 System Architecture

3.1 Design Considerations

We count two types of considerations. One is on the management/administration, and the other is on the extensibility.

A. Management/Administration issues

We point out the following items.

1. security consideration:
   Since the users of *Apostle* are basically casual and assumed to be novice, the system should be secure. Serious users are assumed to have their own machines which are connected to *Apostle* . For those, *Apostle* acts as a backbone of the total network, and is requested to be a secure communication channel. (each group like CSRL may have a loose security inside it.)
2. name management:
   Since there are lots of subdivisions in one organization, the name management facility should have the same hierarchical structure. It should be separated from the super-user of each component and divided.
3. compatibility with the current commercial OS:
   The system should share the fruit of the current technology.
4. request of a user to integrate his own machine which is out of concern for the Trinity:
   The system has no provision for this issue.
5. allocation of files:
   The system is upward compatible with UNIX file system.
6. auditor:
   Accesses and modifications of administrators should be auditorable.

B. Extensibility

We need the ability to freely upgrade the components without the destructive replacement of the whole.

3.2 A System Configuration

Our requirements are summarized as follows.

A) Uniform OS: for user education, and for staff management/training
B) Extensibility/Expandability: the ability to evolve/upgrade
C) Advancedness: International Grade and compatibility.
D) Moderate line to connect

The principles for implementations of the harmonic connection in the Trinity Initiative are:

1. use a moderate line to connect.
   The traffic inside the backbone should be minimized as possible. To do so, the process the user invoked by typing a command or by daemon process, should be executed on the machine he logged in and *not* on the machine he has a home environment. It is not feasible for us to have a tight connection to cope with this assumption.
2. provide user an uniform environment.
   User can access his home directory and the same environment automatically wherever he login.
3. compatibility with the commercial OS.
4. employ TCP/IP technology for LAN link to cope with different higher (application) level protocols employed by different vendors.

With these principles, we design the system of us with the following decisions:

1. Use SUN-4 4.0 and SUN Internetwork Router [SUN87] which is for a physical and data link layer under IP level over a synchronous line, for backbone connection. It means this uniform OS becomes SUN OS, whose 4.0 or later version is said to be compatible with AT&T UNIX System V R4.0 or later. Furthermore, 4.0 has dynamic linking feature which reduces the size of executable codes. It saves the traffic amount. Some data are in the later chapter.
2. Use two 64K NTT super digital lines to connect three campus gateway processors. (Actually, we are trying to integrate them to Multi Media Muxes of NEC, which AGU has been already using.)
3. As a whole, the three machines become one large machine from users point of view.

Fig. 4 shows the whole system. Three Sun-4s are the backbone of the system and are tied together with the harmonic connection concept. Consult Fig.1 and Fig.2 for the physical locations of them.

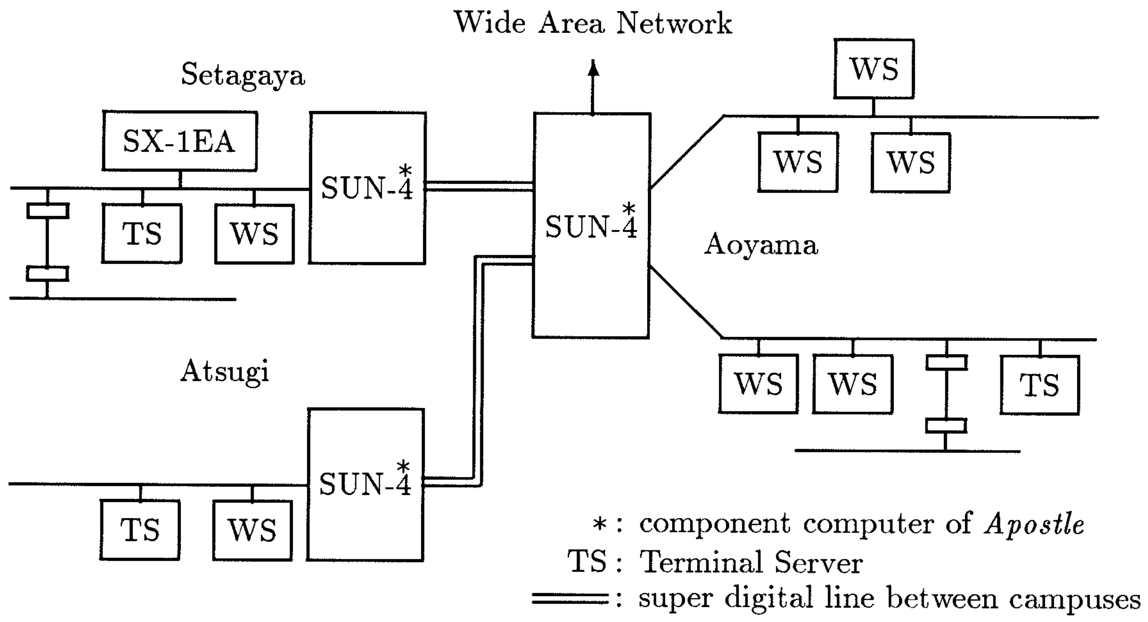

Figure 4: The system configuration

4 Design of *Apostle*

Apostle has three component computers. They are arranged to three campuses each by each as shown in Fig. 4.

4.1 Requirement Analysis

Apostle is analyzed to have the following characteristics:

1. Since all accesses are defined to be secure, everyone is prohibited to access private files of others. He may know who uses the system or what is his name in real life and so on. But he cannot know even the existence of private files of others. To implement this scheme, we adopt a restricted shell technique as a base. This restricted shell requires the division of the whole file system into two categories; public files and private files. Accesses to private files of others are inhibited by this shell.
2. Each user has one virtual environment. He is given his own environment independent from his physical location. To achieve this scheme, information about his working environment must be shared among components on the network.
3. The component computers in every campuses have the copies of the database, which means a set of information about users. There is no needs to cause a traffic across campuses to refer a record of the database.
4. Private files are cached when the owner of them accesses from different component.
5. A person is not allowed to log-in from different place at the same time. This assumption gives some merits to the implementation of *Apostle.* (File cacher described in 5.3 becomes simpler.)
6. The system will be implemented with faculty of Inter Process Communication (IPC). The functions for *Apostle* are arranged to some sub systems or processes. These processes have communication channel for IPC and their interfaces use Remote Procedure Call (RPC) mechanism of SUN.

4.2 Structure of *Apostle*

Apostle consists of three sub systems; an information server (IS), a user interface (UI) and a file cacher (FC).

The information server has three its *alter ego* on each component computer. Each ego of the information server is called a replica of IS. The user interface is a shell described in 4.1(1). Each component computer has its own file cacher.

The information server provides a mechanism to share information of individuals among component computers. A user interface provides a single virtual environment for each user. A file cacher provides the method to access a private file even if a user loges in different component. Fig. 5 shows the structure of *Apostle.* Each replica of IS is in each campus. UI interprets the command line given by a user. UI judges whether the user is permitted to execute the command and to access a file whose name is typed on a command line. For this judgment, UI communicates with a replica of IS on the component computer on which

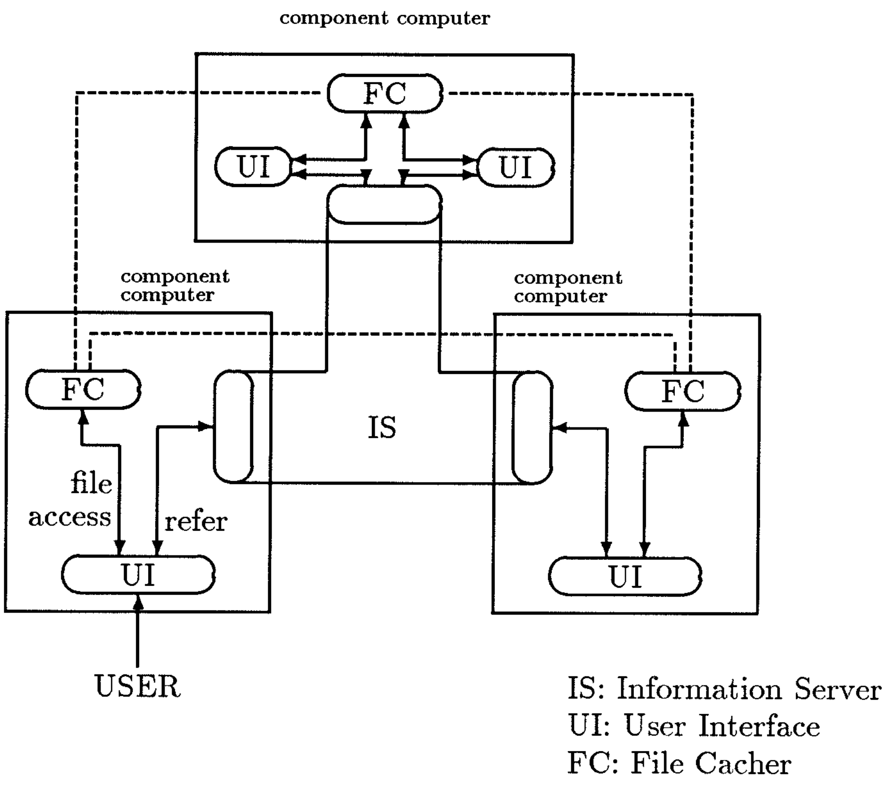

Figure 5: The structure of *Apostle*

UI itself works, and refers a record of the database. After this judgment, UI executes the command and accesses the file. If the file is a private file of the user and it is located on the different component, UI invokes a function of file cacher. FC transfers the copy of the specified private file into the component which the user loges in. If the copy of the private file was already transferred, FC omits the transfer and UI accesses it immediately. With the assumption of 4.1(5), this mechanism is not so difficult to implement. The detail is described in 5.3.

5 Implementation of *Apostle*

5.1 The Information Server (IS)

5.1.1 Structure of the Information Server

Each component computer of *Apostle* has a replica of IS. Each replica has a copy of the same database. The reason is as follows:

1. references to a record of the database are more frequent than change, and
2. the reference across the inter-campus link needs more cost than reference in one component do.

Each replica communicates with each other, and keeps the copy up-to-date.
A user interface or other applications communicate with a replica to refer/change a record of the database.

5.1.2 The Contents of the Database

IS provides the information about users. The database contains the following information:

- user name:
  This name is a symbol given as a human-readable string. Every users on *Apostle* are identified by this name. This name is effective to exchange messages with persons out of *Apostle*.
- user account code:
  This is another identifier for a user. A local operating system of each component uses this code. Because a person is identified by his user name, he does not have to know his account code. A system administrator may refer this code for his operations, such as user accounting.
- information about user working environment:
  This information tells where his private files are located, and where his mail box is. Currently, three types of slots are arranged for this information; 1) home directory for private files, 2) mail box to hold messages, and 3) group directory for private files shared by several persons. These slots contain a) the name of a component on which files are located, and b) the path name for a directory or a file.
- private information:
  This is a personal information of user; the name of a user in his real life, his address, and so on. *Apostle* itself does not need this information but some applications on *Apostle*, such as directory service, may refer/use this information.

5.1.3 Database Access Mechanisms

The scheme to refer to a record is very simple. To access the database, UI or some applications are only required to communicate with a replica of IS on the same component.
To keep the integrity of the database, the procedure to update an entry of the database needs special treatment. To avoid the inconsistency between copies of the database and to

avoid a heavy load for checking the inconsistency or repairing it, we design the procedure to update as follows:

1. A replica of IS accepts the request to change an entry of the database.
2. The replica refers a record of his own copy of the database, and checks whether the request is valid or not. If it is invalid, it will be rejected.
3. The replica verifies if all replicas are alive. When one of them is dead, the replica which accepts the request, wait until all replicas become alive. In this case, the request does not reflect immediately to the database.
4. When all of replicas are alive, the replica tells them the new information, and every entry in copies is changed.
5. If two requests are given at the same time and they conflict with each other, the request will be rejected.

Creation and deletion of an entry are achieved with the same manner as above.

Only an owner (or an administrator) is permitted to update a record of the database. A user identified by the user name in the record is defined to be the owner. No one is permitted to change other's records. But he can refer a record of others, if the owner permits.

5.1.4 Extensions to YP

IS is implemented as extensions of SUN's Yellow Pages (YP)[SUN86b]. SUN defined several maps for YP protocol to share database for network administration. We define new maps for IS of *Apostle*. These maps provide the translation of the name of user to other contents of the database for *Apostle* described in 5.1.2.

Since YP protocol itself has no needs to be modified, the YP facility offered by SUN is preserved. We modify *ypserv(8)* which is YP database server, and *ypbind(8)* which is YP subsystem to find YP database server.

For the procedure to update a record of the database, we implement a new YP subsystem, *ypupdate*. This new subsystem accepts the request to modify a record of the database, and modifies all of copies of the database on each component computer. After modification, it tells the YP database server, *ypserv*, that the database has been changed.

The relationship among these processes is shown in Fig. 6.

5.2 The User Interface (UI)

5.2.1 Classifications of Files

UI assumes the division of the whole file system into the following categories:

1. public files:
   Common commands and data files for these commands belong to this category. Also directories to create temporary files are in this category.
2. private files:
   This category includes all files located under the home directory of a user.

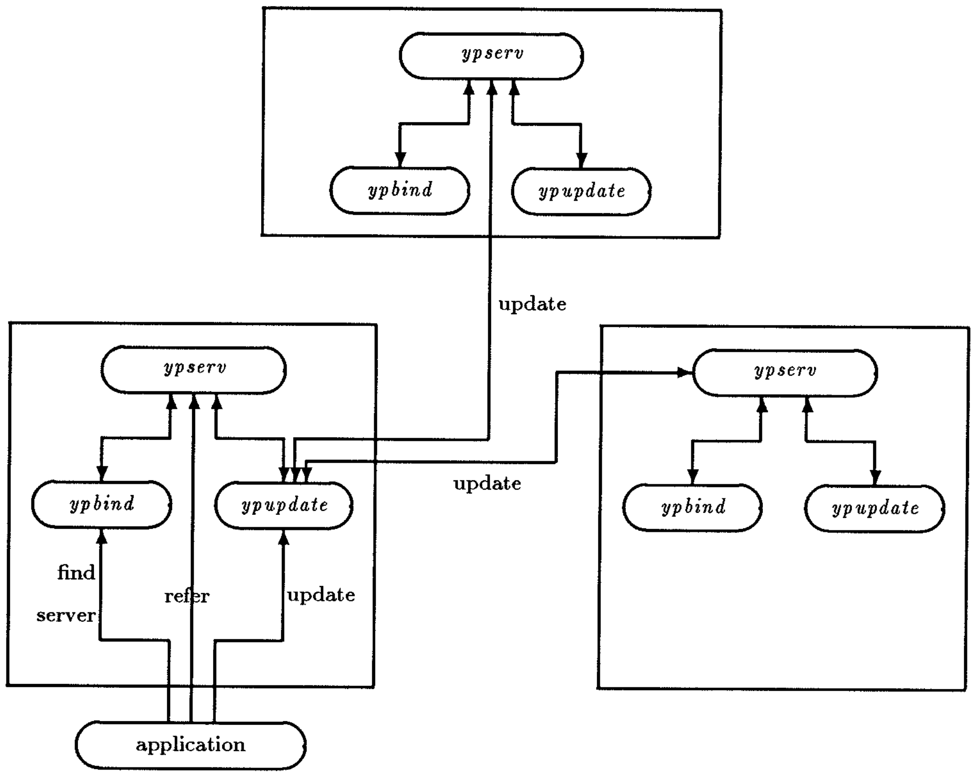

Figure 6: The relationship among subsystems of the information server

3. group files:
   These files are shared by several persons. Files for some project should be shared by members of the project. These files are in this category.
4. system files:
   These files depend on the individual operating system of each component computer of *Apostle.*

5.2.2 Allocation of Each File Class

Private/group files are located on one component computer on the network. When a user wants to refer his private files on a different component, copies of files are transferred from his home to a temporal directory on the component he loges in by the function of FC. Since these files are cached, the next reference to the same file does not need the traffic between components.

Each component has its own public files. *Apostle* itself does not provide the mechanism to keep the consistency among public files on each component.

System files are managed in each component. System files on a component are independent from those on other components. They are used for a local operating system on the component. Every users, except for an administrator of a component computer, is not permitted to access these files.

5.2.3 Access Rights

Four types of users are defined; 1) casual users, 2) administrators for one administrative domain, 3) system administrators, and 4) directors of *Apostle.*

Casual users use public files and private/group files to work. They can create temporary files under a public directory on a component computer he loges in. But they are not permitted to modify public files prepared by the system. The access to private/group files of others are prohibited.

Whole computing environment is divided into some administrative domain. Usual administration, such as the registration of new users, is done by an administrator of the administrative domain. This type of administrator is not permitted to modify files for *Apostle* or files which depend on the local operating system of a component computer.

System administrators manage files for a local operating system. They maintain system files to make a component computer or a network between components to work well.

Directors of *Apostle* control the whole system. They are permitted to access every files on components. Commands or subsystems for *Apostle* are maintained by these directors.

UI recognizes the type of a user, and judges the access right of the user. The classification of the types of users makes the access more secure than traditional UNIX, and tasks for maintenance of the system are arranged into three types of administrators.

5.2.4 Mechanisms

UI will be implemented through two steps.

The first step is the extension of shell program of UNIX. On this stage, all available commands are registered into the command table. This table tells the name of a command, the path name for the command, and its attribute. The attribute is a general description for the command.

By using this table, UI determines which file is needed by a command, or whether the command modifies the file. When a command line is given to UI by a user, firstly, UI checks the access right of the user for the file. When the user is permitted to access the file and the file is a private file, UI communicates with FC on the same component. FC creates the copy of the private file and tells UI the path name of the shadow file. Then, UI converts the name of file on the command line into the path name of the shadow file. Finally, UI executes the command by using the command line converted.

In this scheme, the shell program evaluates only a command line and the name of a file should be described explicitly on it. Therefore the access security is not enough, because the name of a file is not converted and judgment for an access right is not done inside processes.

The second step is the modification of system calls on which path name is given as a parameter, such as *open(2)*, *stat(2)*. UI checks an access right through the communication with IS, then converts the file name (if necessary), and executes the command. In the primitive of a system call, the access right is checked, and file cacher is called. Dynamic link editing for shared libraries supported by SunOS 4.0 or later is a candidate of the substrate mechanism.

In this scheme, many faculties of the previous version of UI will move into primitives of system calls. Every accesses become more secure, and file cache will be available even if the path name of a file is not declared explicitly.

5.3 The File Cacher (FC)

5.3.1 Key Techniques

When a user loges in a component computer different from the component on which his private files are located, the mechanism of FC is used for a quick response.

The access over inter-campus link makes the traffic too heavy. FC transfers the copy of a private file from his home directory to a temporary directory on the component he loges in. This transfer is done upon the first reference from UI, and the next reference, even it might be a destructive update, to the same file does not cause the traffic over inter-campus link. When the user loges out, these copies are removed or some of them are wrote back into the original files, if they are modified.

UI or some primitives for file handling call FC. Or each FC communicates with each other, if necessary.

5.3.2 Copies Created by FC

There are two types of copies created by FC. One is the contents of a file, and another is the attribute of a file. Usually, the size of the contents of a file is much lager than the size of its attribute. Therefore, if a command needs only the attribute of a file, the contents is not transferred. This mechanism includes the situation that the contents of a file is not transferred when there is only a directory reference.

5.3.3 File Locking

Even if a private file is cached and accessed to modify, there is no need to lock the original file, because of the assumption that a user is not permitted to log-in from different place at the same time. But group files should be locked to avoid the confliction on the modification. When a group file is transferred for modification, a modification-inhibit mark is flagged on the original files. UI sends a request to clear cache when the user loges out or he types a special command to clear cache. After FC receives these requests, it transfers the copy into the original file and clears the modification-inhibit mark. Otherwise, if the file is not modified, the file cacher omits the transfer.

6 Assessment for the Performance and the Dynamic Behavior

6.1 Summary of Assumed Environment

The environment of *Apostle* is considered as follows;
1) It is for the researchers not for student education.
2) Since most of the real computer scientists have their own machines which are connected to *Apostle*, the intended users of *Apostle* are the novices. For computer scientists, *Apostle* acts as a backbone for his message/file transfer.
3) The total number of permanent professors including research associates are some 500. The active users are estimated to 200.
4) The users seem to do the following job categories.

i) E-mail handling.

ii) TSS front-end for SX super computer using a special interface which enables the automatic conversion of UNIX command and host TSS command. (users are only required to remember UNIX commands.)

iii) Editing of the textual files including personal computer files, UNIX files, and host files. Code conversion, or media conversion are included.

iv) Reading and posting articles on JUNET news system.

v) Other operations including real-time message exchange, numeric operation, software development.

5) Usage is locally dense totally.

6.2 Assessment

We must estimate how much is the traffic among campuses. If the total amount is too large, response time for user is too bad, since the system characteristics has the principle that we hire 64kbps as a design speed for the intercampus connection. 64kbps is fast enough to have an occasional transfer, but is too slow to have an real time/interactive use.

The following assumptions are obtained as of now.
1) the actual performance of 64kbps sync line seems to be 48k or less. It is a 5kbyte/sec speed. Since we will use multi-medea-mux for multiplexed digital communication, we cannot make performance better.
2) the size of a program seems to be 50kbytes or so. Then, transferring it will need about 10 sec. It is not acceptable for interaction.
3) the size of average mail seems to be 4kbyte. it requires 1 sec to transfer.
4) the size of mail box seems to be 200kbyte. it requires 40 sec to transfer.
5) 80 percent of the command a user types is system given, not one prepared by himself.
6) but while, 80 percent of the files he refer is his private file, not system given ones.

There arises the following questions;
a) we should install special handling mechanism for mail reading/posting/saving?
b) we should need a kind of absenty at his home machine while he loges in at different campus. do they need co-work?
c) what is needed to transfer upon login? a kind of pre-fetch is needed?
d) in what extent we should prepare for his environment at a different machine?

We must answer for the above questions. Since we are in the early stage of implementation, answers will be described in [CSRL88b].

6.3 A Scenario Study

This section discusses the process during a conversation on *Apostle*. Figure 7 shows a sample conversation with *Apostle*. The user logged-in at Atsugi campus, while his home directory is at Aoyama campus.

After log-in, the whole environment of him is understood by the shell at Atsugi. And the current directory is set to his home at Aoyama namely, /home/aoyama/m-ida. When he typed 'cd' command, the directory is copied to temporary directory for cache at Atsugi. Files at the directory are not copied immediately to avoid the unneeded transfer. When he invoke or refer his file, the contents of it are transferred. After that, the copy at Atsugi will be used.

Details of the process is described in Fig. 7

6.4 Estimation of Dynamic-linking Effects

SunOS 4.0 has the dynamic-linking feature. It enables a dynamic library linking on execution time. As a result, object size needed is minimized. This fact has great influence to the dynamic behavior of *Apostle* . Fig. 8 shows the comparison of the sizes for /usr/ucb between 4.0 and 3.2 on SUN4. We choose this comparison as a case. Fig. 8 tells the size for each object of 4.0 is reduced to 50% of 3.2. Furthermore, if we neglect those which are not recompiled to port them from 3.2 to 4.0, such as e,edit,ex,rcp,vi, the reduction (or the effect of dynamic-linking) becomes larger. 4.0 version is about 40% of 3.2 version.

6.5 A Guideline for Applications

At first, the performance will deeply depend on the performance of FC. Since FC has great roles and FC has two levels of caching principles, application should care about the invocation or reference to the files. Unneeded refer may cause the performance worse.

Recommended managements are as follows;

1) Use MH as an recommended mail system. And *Apostle* provides a special treatment for inbox access. (Basically *Apostle* will not copy inbox into the component he logs-in.)

1-1) There is a problem about the mail box manipulation. If the user uses 'mail' command facility, the system automatically copies the whole message to /tmp directory. (though after copied, he can have quick access.) If he has lots of messages to read, say, 50

```
4.2 BSD UNIX (Apostle)

Atsugi login: m-ida                                 --- (1)
password:                                           --- (2)
Last login: Tue Jun 14 11:05:01 from backabon
atsugi% cd x3j13; pwd                               --- (3)
/home/aoyama/m-ida/x3j13
atsugi% ls                                          --- (4)
cl.mbox        clos.mbox      senddoc.1
cl-edit.mbox   packmail       x3j13.mbox
atsugi% packmail cl.mbox                            --- (5)
atsugi% ex /tmp/work.doc                            --- (6)
"work.doc" 213/4378
:r ~keisuke/important.doc                           --- (7)
access denied
"important.doc" No such file or directory
:!ls /usr/spool                                     --- (8)
access denied
/usr/spool not found
!
:q!
atsugi% logout                                      --- (9)
```

(1) Login process gets user name.
(2) Login process consults the database on Atsugi component and gets correct password of the user, communicating IS. Then, checks whether the password is valid and accepts. After that his top level directory contents are transferred.
(3) Upon typing the 'cd' command, FC transfers the contents of the current directory from his home host Aoyama to the local host Atsugi.
(4) FC refers locally the contents of the directory. No tramsmit of the file contents between hosts is occured.
(5) The contents of the file 'packmail' and 'cl.mbox' are transferred by FC. And then executed.
(6) Accesses the local file '/tmp/work.doc'. No network access is occured.
(7) UI inhibits his illegal access to files owned by others.
(8) UI inhibits his illegal access to system files.
(9) FC makes synchronization between cache and original files.

Figure 7: example of a conversation

or 100 or more, and if he has no will to read them, there arise a message copy everytime and causes unneeded transfer between different components. The system *Apostle* itself has no provision for the situation. The policy of this system is 'mail command system is not the standard mailing facility but use mh instead.'

1-2) When MH, he must use his home directory. His directory for inbox will affect a lot for access frequency. It might be too much traffic without any special provision.

2) There is no need to limit multiple login on the same component. A same-time-login from different site is prohibited.

3) There is no need to have a special consideration on reading/posting articles for news system. News articles are stored locally. Archives of each articles are in some extent stored locally. Main archives are on the Aoyama machine.

4) Classification of file semantics like objects, general files, messages (news, private mails, interactive messages) is under consideration.

5) *Apostle* is NOT a full distributed OS. How 'w' command walk? Can he see all the persons of three machine? At least we will provide more suitable command for such operation.

6) Mail address of each user is uniformly set to 'foo@cc.aoyama'. Mail system is altered to cope with this.

```
---------- SUN 4 /usr/ucb ------ compared at 1988.08.01 ------
   ====== 4.0 ============== 3.2 ==========            ====== 4.0 ============== 3.2 ==========
 122880  Mail        163840  Mail       0.750 |                          65536  pxref
   2504  biff         40960  biff       0.061 |     24576  quota         90112  quota        0.273
                        120  ccat             |     90112  rcp           90112  rcp          1.000
  11856  checknr      49152  checknr    0.241 |      3568  rdate         65536  rdate        0.054
                      40960  chsh             |     65536  rdist        131072  rdist        0.500
  16384  clear        24576  clear      0.667 |     32768  reset         57344  reset        0.571
   4488  colcrt       40960  colcrt     0.110 |     24576  rlogin        81920  rlogin       0.300
   3456  colrm        24576  colrm      0.141 |     16384  rsh           73728  rsh          0.222
                      40960  compact          |     16384  rup           81920  rup          0.200
  24576  compress     49152  compress   0.500 |      5632  ruptime       40960  ruptime      0.138
  15792  ctags        49152  ctags      0.321 |     16384  rusers        81920  rusers       0.200
 450560  dbx         425984  dbx        1.058 |      3632  rwho          40960  rwho         0.089
 196608  e           196608  e          1.000 |     24576  sccs          90112  sccs         0.273
 196608  edit        196608  edit       1.000 |     16384  script        40960  script       0.400
  32768  error        98304  error      0.333 |      3376  soelim        40960  soelim       0.082
 196608  ex          196608  ex         1.000 |      4160  strings       40960  strings      0.102
   4400  expand       40960  expand     0.107 |      5912  symorder      65536  symorder     0.090
                      57344  eyacc            |                          65536  syslog
  24576  finger       90112  finger     0.273 |     16384  tail          40960  tail         0.400
  16384  fmt          40960  fmt        0.400 |     40960  talk          98304  talk         0.417
   3416  fold         40960  fold       0.083 |      4616  tcopy
   3832  from         65536  from       0.058 |     32768  telnet        90112  telnet       0.364
   7088  fsplit                               |     24576  tftp          81920  tftp         0.300
  81920  ftp         114688  ftp        0.714 |     32768  tset          57344  tset         0.571
   3352  gcore        49152  gcore      0.068 |     24576  ul            49152  ul           0.500
  32768  gprof        57344  gprof      0.571 |                          40960  uncompact
   3424  groups       73728  groups     0.046 |     24576  uncompress    49152  uncompress   0.500
   3272  head         40960  head       0.080 |      2952  unexpand      40960  unexpand     0.072
   5608  last         73728  last       0.076 |      7800  unifdef       40960  unifdef      0.190
  16384  lastcomm     73728  lastcomm   0.222 |     24576  uptime        49152  uptime       0.500
   4848  leave        49152  leave      0.099 |      3616  users         40960  users        0.088
  16384  logger                               |     24576  vacation      73728  vacation     0.333
  32768  lpq          73728  lpq        0.444 |      2871  vgrind         2871  vgrind       1.000
  24576  lpr          90112  lpr        0.273 |    196608  vi           196608  vi           1.000
  24576  lprm         90112  lprm       0.273 |    196608  view         196608  view         1.000
  16384  lptest                               |     24576  vmstat        49152  vmstat       0.500
 122880  mail        163840  mail       0.750 |     24576  w             49152  w            0.500
  24576  man          49152  man        0.500 |      3328  wc            40960  wc           0.081
   6072  mkstr        40960  mkstr      0.148 |      3256  what          40960  what         0.079
  49152  more         57344  more       0.857 |     16384  whatis        40960  whatis       0.400
  32768  netstat      90112  netstat    0.364 |     16384  whereis       40960  whereis      0.400
  49152  page         57344  page       0.857 |       987  which           987  which        1.000
                      40960  pmerge           |      1504  whoami        65536  whoami       0.023
   1568  printenv     32768  printenv   0.048 |      3336  whois         65536  whois        0.051
                      65536  prmail           |      8544  xstr          49152  xstr         0.174
                      40960  pti              |     16384  yes           32768  yes          0.500
                     139264  pxp              |     24576  zcat          49152  zcat         0.500
                                                -----------------------------------------------
                                                  3001370              5861138               0.512
                                                    37992                74191               0.512

when ommiting eight cases which have the same size      1924360              4784128               0.402
    (e,edit,ex,rcp,vgrind,vi,view,which)                  27104                67382               0.402
```

Figure 8: Dynamic Linking Reduces the Size - Case for /usr/ucb

Table 2: Expected merits over several connection styles

	tightly coupled	loosely coupled	network integration	communication tool	mail/ message routing
assumed media	leased line	DDX-P, or super digital	leased line	leased line	dial-up line
initial cost	very high	medium	high	high	low
running cost/message	low	low/medium	low	high	high

7 Final Remarks

Since the project started in April 1988, we had not yet finished the whole design and implementation, and we cannot have a review and an evaluation.

Table 2 shows an estimation of the comparison of five types of connections concerning the cost assessment. As shown in the table, we expect the cost merit over tight connection though it can provide a resemble environment as tight connection.

The Trinity Initiative is a three year plan and the first version of *Apostle* is scheduled to finish at the end of March 1989. The details of *Apostle* will be published as [CSRL88b]. We will have a schedule to write papers about the result in the next year.

The harmonic connection itself is a quite general enough to apply other cases. So, we would like to find any other applicable cases.

Acknowledgment

The authors would like to express their gratitude to Prof. Hiroshi Oyachi (director, ISRC of AGU), Prof. Hisao Nishioka (president, AGU), and Prof. Kinjiro Ohki (chancellor, Aoyama Gakuin) and the office staffs of ISRC for their continuous encouragements. Also thanks are due to JUNET/WIDE research group members and Prof. Haruhisa Ishida of University of Tokyo for their technical communications.

References

[Accetta86] M.Accetta, R.Baron, W.Bolosky, D.Golub, R.Rashid, A.Tevanian, and M. Younge, *'Mach: A New Kernel Foundation for UNIX Development,'* in Proc. of USENIX 1986 summer conf. pp93-112, 1986.

[BBN87] BBN Advanced Computers Inc. *'Butterfly Product Overview,'* Oct.14 1987.

[CSRL88a] Masayuki Ida and Keisuke Tanaka, *'The Harmonic Connection Concept in the Trinity Initiative of Aoyama Gakuin University'* in proc. of JCCW'88 pp111-120 July. 1988.

[CSRL88b] Keisuke Tanaka and Masayuki Ida *'The* Apostle *System Reference Manual,'* CSRL Technical Report #88-002, Nov. 1988. (to be appeared)

[Lyon84] B. Lyon, *'Sun Remote Procedure Call Specification,'* Technical Report, 1984, Sun Microsystems, Inc.

[Postel81a] J. Postel, *'Transmission Control Protocol,'* RFC793, Sep. 1981.

[Postel81b] J. Postel, *'Internet Protocol,'* RFC791, Sep. 1981.

[SUN86a] Sun Microsystems Inc. *'Network File System Protocol Specification,'* Feb. 1986

[SUN86b] Sun Microsystems Inc. *'The Yellow Pages Protocol Specification,'* Feb. 1986

[SUN87] Sun Microsystems Inc. *'SunLink Internetwork Router System Administration Guide,'* Jul. 1987