1983

Transparent Integrated Local and Distributed Environment (TILDE) Project Overview

Douglas E. Comer
Purdue University, comer@cs.purdue.edu

Report Number:
83-466

Transparent Integrated Local and Distributed Environment

(T I L D E)

Project Overview

Douglas Comer

Computer Science Department
Purdue University
West Lafayette, IN 47907

Tilde Report
CSD-TR-466

This project is supported in part by grants from the National Science Foundation (MCS-8219178), SUN Microsystems Incorporated, and Digital Equipment Corporation.

1. Introduction

TILDE is a multi-year project exploring distributed computing based on an architecture in which the primary *computing engine* consists of a cluster of heterogeneous machines loosely coupled with high-speed local area networks (see Figure 1).

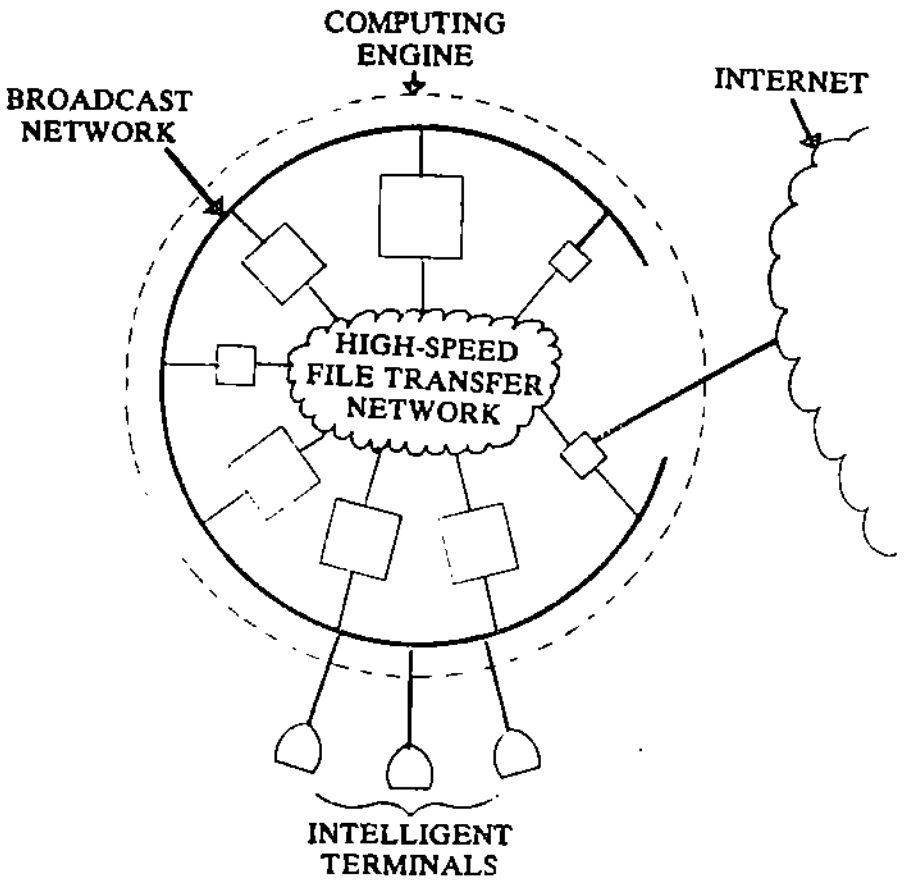

Figure 1. A multi-machine computing engine.

At the center of the computing engine is a high-speed file transfer network that transfers single blocks or entire files among machines. Because the computing engine places high demand on the file transfer network, it must behave well under high load (we plan to use token-passing ring technology).

Another local area network provides an interface between the Computing Engine and users. Because users need to search for services among all machines, the interface network must provide broadcast or, preferably, multicast service (we plan to use CSMA-CD technology for the interface network).

Multi-machine computing engines are themselves connected with an internet. Thus, at least one of the machines in a given computing engine serves as a gateway, connecting the computing engine to the internet. Although remote computing engines can communicate, we assume that transmission delays across the Internet may be significantly longer than delays within the computing engine.

Each individual machine in a computing engine provides a set of *services* such as electronic mail, general purpose command interpretation, hardcopy printing, or high-speed vector processing. Heavily used services are duplicated on several machines to improve performance. Less heavily used services, or services that require special-purpose hardware may be available on only one machine in each computing engine, or may only be available on remote computing engines.

Conceptually, a user has access to all possible services simultaneously, whether they are supplied by the local computing engine or a remote one. The user interface that provides such access consists of an *intelligent terminal* that communicates with the computing engine over the interface network. More powerful than conventional terminals but less powerful than independent "workstations", intelligent terminals provide two important mechanisms. First, they request services from individual machines of the computing engine on behalf of the user. Second, they provide limited local computing power for special-purpose interactive tasks (e.g., front-ends of editors).

Multi-machine computing engines differ from conventional architectures in significant ways. They have more power than systems of personal workstations because most processing is performed on arbitrarily powerful processors in the computing engine. They have greater flexibility than conventional time-sharing systems because new services and more computational power can be added to the system by adding individual machines to the engine. Most importantly, it is possible to choose hardware appropriate to the computation. For example, CPU-intensive services such as "number crunching" can be performed on large mainframe machines, while services like electronic mail can be provided by inexpensive minicomputers.

Project Goals

The goal of the TILDE project is to explore general-purpose computing systems that run on the multi-machine computing engine. The scope includes systems architecture as well as principles of distributed systems. We focus on location-transparent interfaces, distributed file systems, and high-level services. More specifically, the project explores computing systems in which the user interface hides details of the underlying architecture, making the multi-machine computing engine appear to be a single, large time-sharing system.

Tilde Prototype

The immediate project objective is to produce an experimental prototype Tilde system. The prototype will serve as the basis for further experiments with system design, and as the foundation on which new high-level services are built. The model we have in mind is a UNIX-like system with the syntactic parts of the interactive shell (command interpreter) resident in the intelligent terminal, and the command set is augmented with new commands that implement high-level services. We envision a prototype system that will:

– provide roughly the same mix of general-purpose computing as a single-machine running the UNIX timesharing system

– be easily expandable. In particular, it will be possible to increase the power of the system by adding more individual machines to the computing engine; it will be possible to increase the variety of services provided by introducing new user-level software without rebinding the kernel.

– have rougly the same performance characteristics as a single-machine UNIX system (i.e., the user will perceive no gross difference in performance).

– provide each user access to all services.

– provide access to new high-level services.

– integrate communication/access among local and remote computing engines.

Research Plan

The research is experimental. It consists of designing, building, measuring, and improving. We will design and build a multi-machine computing engine prototype by Fall of 1984 and refine it by Spring of 1985. The prototype will include a 3-machine computing engine and at least 6 intelligent terminal interface units.

The intellectual challenge lies in finding a system architecture that is powerful, flexible, and efficient. Solving the problem requires more than engineering skills because the design involves research with naming, synchronization, and communications mechanisms. Solving the problem involves more than mathematics because system performance is important. The main result of our work will be improved understanding of the issues underlying multi-machine systems and computational services.

2. Technical Issues

Naming

Computing environments provide access to objects such as processes, users, devices, files, and services. In distributed environments, the question of naming is central to system design because names can only be exchanged freely among mechanisms that dereference them the same way. For example, if each machine in the computing engine maintains its own file name space, a given file name may not refer to the same file when interpreted on two different machines.

A naming mechanism is *transparent* if all names are known globally (i.e., if a given name refers to exactly the same object independent of the context in which it is interpreted). Name transparency is desirable because it allows users to exchange objects like programs that reference data by name. Ultimately, name transparency can only be achieved if names are unique. Such uniqueness requires either a central authority to designate all names, or an agreement by which individual "sites" (e.g., a computing engine) can assign names. Consulting a central authority is too inefficient, so the question becomes how to assign transparent names in a distributed environment.

The major complication with distributed name assignment is that most schemes introduce location dependencies. For example, some distributed systems prepend machine names onto file names, making names take the form *machine:dfile*. We reject this approach because it implies that references to a file must change whenever the file moves from one machine to another (see "file system design" below).

Another naming consideration concerns name length. Users prefer to *shorten* names, using abbreviations in place of longer names whenever the correct name can be deduced from the context in which the name is resolved. For example, file names in UNIX are specified either as full path names or shortened names (which are interpreted with respect to the "current" directory). Allowing users to substitute short names for longer ones makes naming convenient, but increases the probability that names conflict.

In general, TILDE provides transparency only for those objects which users routinely manipulate. Primarily, this means that in a given computing engine, TILDE interprets file names and user login names consistently. We expect, however, that other identifiers may not be known globally. For example, process identifiers may remain local to the machine that executes the process. Keeping some identifiers local helps make the system more efficient. One of the important issues is whether such partial transparency provides sufficient flexibility.

File System Design

Having transparent file names is especially important because users reference data and programs by naming the files in which they reside. The file system can be partitioned into two parts, one that deals with the binding of file names to files, and another that provides access to a file. The first part, usually called the directory, implements the naming scheme and the binding of names to objects. The second part provides access to a file object once its location is known.

Because users interact with more than one machine in the computing engine at a given time, the file system must provide efficient access to all files. On one hand, centralized file systems are less efficient than distributed systems because the central file server forms a bottleneck in the system. Oh the other hand, consistency and synchronization are difficult in distributed systems. We are interested in a compromise between centralized file systems and completely distributed ones. One compromise considers file systems which keep data "close" to the most frequent point of access, and migrates files slowly as the focus of access changes. In such an environment, files reside on storage devices attached to individual machines in the computing engine. Although files reside on particular machines, any machine in the computing engine can access any file because each machine also provides block-level access to its files through the file transfer network. The goal of a migrating file system is to make most file access as efficient as a conventional single-processor computing system while providing as much flexibility as a completely distributed file system.

Intelligent Terminal Interface

Intelligent terminals serve as the user interface to a computing engine. We seek to use the intelligent terminal for two purposes. First, it will handle the syntactic details of keyboard input and pointer tracking with a single, uniform mechanism. Thus, the input to all services can use a single style. Second, it will allow programmers to write specialized front-end software and download it into the terminal to

improve response. Thus, it will be possible to have a mail system that responds to keystrokes rapidly, even though a machine in the computing engine performs much of the mail processing.

3. High-Level Services

A major part of Tilde concentrates on designing high-level services that the computing engine provides. Two such projects are already underway.

Electronic Mail

Electronic mail is currently based on independent memos. However, subscribers frequently use mail systems to transport files and to conduct conferences among groups of individuals. Research on mail will explore a higher-level mail environment based on conversations rather than memos. The notion of "conversation" encompasses traditional memo systems, bulletin-boards, and conferencing systems. our conversation-based mail will allow users to "attach" files to a message in such a way that the recipient can save the attached files without reading them. It will also keep the history of an exchange, sort incoming mail so that all messages pertaining to a given conversation are kept together, and stamp an expiration date on messages so messages like "system going down at 10AM" will disappear automatically at the appropriate time.

Event Service

Tilde will provide a service that handles scheduled events. Examples of such events include: asking for a reminder message at a given time, scheduling computation to be performed at a given time (or on a periodic basis), keeping an appointment calendar, and providing a time-stamp for distributed algorithms. The idea is to collect all services that deal with time together and provide a single, uniform interface for accessing them. The event service must be reliable (in the sense that system crashes will not cause loss of events), and efficient (in the sense that scheduling an event is fast).

4. Personnel

Below is a summary of Tilde personnel along with their network addresses:

Project Coordinator

Douglas Comer (dec@purdue) directs the Tilde project, coordinating the research efforts. In addition, Comer manages the high-level mail, and event services subprojects. Comer also manages the distributed naming research.

Other Principal Investigators:

John T. Korb (jtk@purdue) is in charge of the intelligent terminal subproject.

Walter F. Tichy (wft@purdue) is in charge of the migrating file system research.

Faculty Associates

Thomas P. Murtagh (tpm@purdue) works with Comer on distributed naming.

Supported Students

Larry Peterson (llp@purdue) works on the high-level mail service.

Christopher Kent (cak@purdue) works on network protocols and system architecture

Zuwang Ruan (zxr@purdue) works on migrating file systems.