Operating Systems

Anita K. Jones
Editor

Grapevine: An Exercise in Distributed Computing

Andrew D. Birrell, Roy Levin,
Roger M. Needham, and Michael D. Schroeder

Xerox Palo Alto Research Center

Grapevine is a multicomputer system on the Xerox research internet. It provides facilities for the delivery of digital messages such as computer mail; for naming people, machines, and services; for authenticating people and machines; and for locating services on the internet. This paper has two goals: to describe the system itself and to serve as a case study of a real application of distributed computing. Part I describes the set of services provided by Grapevine and how its data and function are divided among computers on the internet. Part II presents in more detail selected aspects of Grapevine that illustrate novel facilities or implementation techniques, or that provide insight into the structure of a distributed system. Part III summarizes the current state of the system and the lessons learned from it so far.

CR Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications, distributed databases*; C.4 [**Performance of Systems**]—*reliability, availability and serviceability*; D.4.7 [**Operating Systems**]: Organization and Design—*distributed systems*; H.2.4 [**Database Management**]: Systems—*distributed systems*; H.2.7 [**Database Management**]: Database Administration; H.4.3 [**Information Systems Applications**]: Communications Applications—*electronic mail*

General Terms: Design, Experimentation, Reliability

Part I. Description of Grapevine

1. Introduction

Grapevine is a system that provides message delivery, resource location, authentication, and access control ser-

Authors' Present Addresses: Andrew D. Birrell, Roy Levin, and Michael D. Schroeder, Xerox Palo Alto Research Center, Computer Science Laboratory, 3333 Coyote Hill Road, Palo Alto, CA 94304; Roger M. Needham, University of Cambridge Computer Laboratory, Corn Exchange Street, Cambridge, CB2 3QG, United Kingdom.

vices in a computer internet. The implementation of Grapevine is distributed and replicated. By *distributed* we mean that some of the services provided by Grapevine involve the use of multiple computers communicating through an internet; by *replicated* we mean that some of the services are provided equally well by any of several distinct computers. The primary use of Grapevine is delivering computer mail, but Grapevine is used in many other ways as well. The Grapevine project was motivated by our desire to do research into the structure of distributed systems and to provide our community with better computer mail service.

Plans for the system were presented in an earlier paper [5]. This paper describes the completed system. The mechanisms discussed below are in service supporting more than 1500 users. Designing and building Grapevine took about three years by a team that averaged two to three persons.

1.1 Environment for Grapevine

Figure 1 illustrates the kind of computing environment in which Grapevine was constructed and operates. A large internet of this style exists within the Xerox Corporation research and development community. This internet extends from coast-to-coast in the U.S.A. to Canada, and to England. It contains over 1500 computers on more than 50 local networks.

Most computing is done in personal *workstation* computers [12]; typically each workstation has a modest amount of local disk storage. These workstations may be used at different times for different tasks, although generally each is used only by a single individual. The internet connecting these workstations is a collection of Ethernet local networks [6], gateways, and long distance links (typically telephone lines at data rates of 9.6 to 56 Kbps). Also connected to the internet are *server* computers that provide shared services to the community, such as file storage or printing.

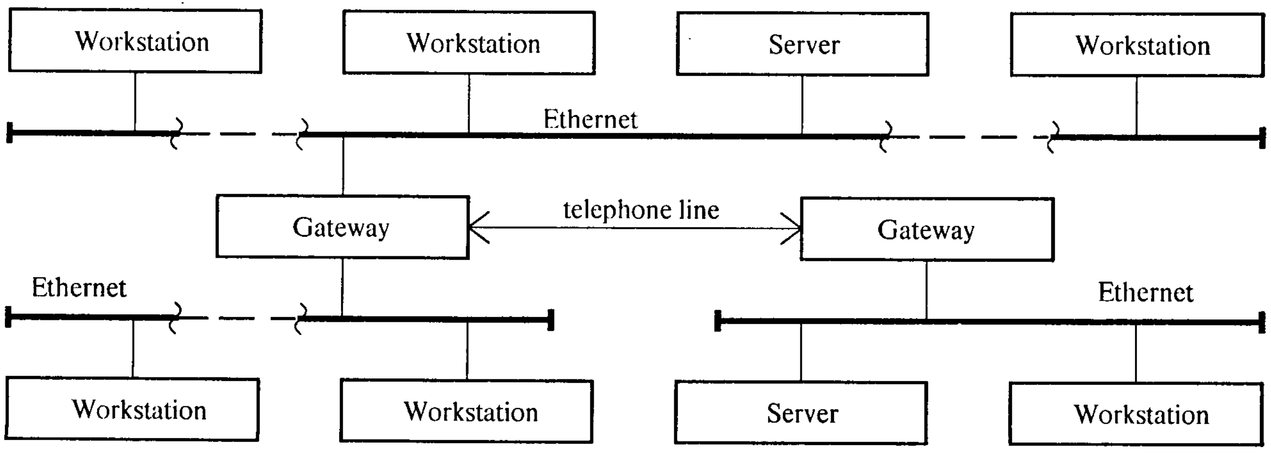
Protocols already exist for communicating between computers attached to the internet [11]. These protocols provide a uniform means for addressing any computer attached to any local network in order to send individual packets or to establish and use byte streams. The individual packets are typically small (up to 532 bytes), and are sent unreliably (though with high probability of success) with no acknowledgment. The byte stream protocols provide reliable, acknowledged, transmission of unlimited amounts of data [1].

1.2 Services and Clients

Our primary consideration when designing and implementing Grapevine was its use as the delivery mechanism for a large, dispersed computer mail system. A computer mail system allows a group of human users to exchange messages of digital text. The sender prepares a message using some sort of text editing facility and names a set of recipients. He then presents the message to a delivery mechanism. The delivery mechanism moves the message from the sender to an internal buffer for each recipient, where it is stored along with other messages for that recipient until he wants to receive them. We call the buffer for a recipient's messages an *inbox*. When ready, the recipient can read and process the messages in his inbox with an appropriate text display program. The recipient names supplied by the sender may identify *distribution lists*: named sets of recipients, each of whom is to receive the message. We feel that computer mail is both an important application of distributed computing and a good test bed for ideas about how to structure distributed systems.

Buffered delivery of a digital message from a sender to one or more recipients is a mechanism that is useful in many contexts: it may be thought of as a general communication protocol, with the distinctive property that the recipient of the data need not be available at the time the sender wishes to transmit the data. Grapevine separates this message delivery function from message creation and interpretation, and makes the delivery function available for a wider range of uses. Grapevine does not interpret the contents of the messages it transports. Interpretation is up to the various message manipulation programs that are software *clients* of Grapevine. A client

Fig. 1. An Example of a Small Internet.



program implementing a computer mail user interface will interpret messages as interpersonal, textual memos. Other clients might interpret messages as print files, digital audio, software, capabilities, or data base updates.

Grapevine also offers *authentication*, *access control*, and *resource location* services to clients. For example, a document preparation system might use Grapevine's resource location service to find a suitable printing server attached to the internet (and then the message delivery service to transfer a document there for printing) or a file server might use Grapevine's authentication and access control services to decide if a read request for a particular file should be honored.

Grapevine's clients run on various workstations and server computers attached to the internet. Grapevine itself is implemented as programs running on server computers dedicated to Grapevine. A client accesses the services provided by Grapevine through the mediation of a software package running on the client's computer. The Grapevine computers cooperate to provide services that are distributed and replicated.

2. Design Goals

We view distributed implementation of Grapevine both as a design goal and as the implementation technique that best meets the other design goals. A primary motivation for the Grapevine project was implementing a useful distributed system in order to understand some system structures that met a real set of requirements. Once we chose message delivery as the functional domain for the project, the following specific design goals played a significant role in determining system structure.

Grapevine makes its services available to many different clients. Thus, it should make no assumptions about message content. Also, the integrity of these services should not in any way depend on correctness of the clients. Though the use of an unsatisfactory client program will affect the service given to its user, it should not affect the service given to others. These two goals help determine the distribution of function between Grapevine and its clients.

Two goals relate to Grapevine's reliability properties. First, a user or client implementor should feel confident that if a message is accepted for delivery then it will either be made available to its intended recipients or returned with an indication of what went wrong. The delivery mechanism should meet this goal in the face of user errors (such as invalid names), client errors (such as protocol violations), server problems (such as disk space congestion or hardware failures), or communication difficulties (such as internet link severance or gateway crashes). Second, failure of a single Grapevine server computer should not mean the unavailability of the Grapevine services to any client.

The typical interval from sending a message to its arrival in a recipient's inbox should be a few minutes at most. The typical interactive delay perceived by a client program when delivering or receiving a message should be a few seconds at most. Since small additions to delivery times are not likely to be noticed by users, it is permissible to improve interactive behavior at the expense of delivery time.

Grapevine should allow decentralized administration. The users of a widespread internet naturally belong to different organizations. Such activities as admission of users, control of the names by which they are known, and their inclusion in distribution lists should not require an unnatural degree of cooperation and shared conventions among administrations. An administrator should be able to implement his decisions by interacting directly with Grapevine rather than by sending requests to a central agency.

Grapevine should work well in a large size range of user communities. Administrators should be able to implement decentralized decisions to adjust storage and computing resources in convenient increments when the shape, size, or load patterns of the internet change.

Grapevine should provide authentication of senders and recipients, message delivery secure from eavesdropping or content alteration, and control on use and modification of its data bases.

3. Overview

3.1 Registration Data Base

Grapevine maintains a *registration data base* that maps names to information about the users, machines, services, distribution lists, and access control lists that those names signify. This data base is used in controlling the message delivery service; is accessed directly for the resource location, access control, and authentication services; and is used to configure Grapevine itself. Grapevine also makes the values in the data base available to clients to apply their own semantics.

There are two types of entries in the registration data base: *individual* and *group*. We call the name of an entry in the registration data base an *RName*.

A group entry contains a set of RNames of other data base entries, as well as additional information that will be discussed later. Groups are a way of naming collections of RNames. The groups form a naming network with no structural constraints. Groups are used primarily as distribution lists: specifying a group RName as a recipient for a message causes that message to be sent to all RNames in that group, and in contained groups. Groups also are used to represent access control lists and collections of like resources.

An individual entry contains an *authenticator* (a password), a list of *inbox sites*, and a *connect site*, as well as additional information that will be discussed later. The inbox site list indicates, in order of preference, the Grapevine computers where the individual's messages may be buffered. The way these multiple inboxes are used is discussed in Sec. 4.2. The connect site is an internet address for making a connection to the individual. Thus, an individual entry specifies ways of authenticating the identity of and communicating with—by message delivery or internet connection—the named entity. Individuals are used to represent human users and servers, in particular the servers that implement Grapevine. Usually the connect site is used only for individuals that represent servers. Specifying an individual RName (either a human or a server) as a recipient of a message causes the message to be forwarded to and buffered in an inbox for that RName.

3.2 Functions

Following is a list of the functions that Grapevine makes available to its clients. Responses to error conditions are omitted from this description. The first three functions constitute Grapevine's *delivery service.*

Accept message:

[sender, password, recipients, message-body] → ok

The client presents a message body from the sender for delivery to the recipients. The sender must be RName of an individual and the password must authenticate that individual (see below). The recipients are individual and group RNames. The individuals correspond directly to message recipients while the groups name distribution lists. After Grapevine acknowledges acceptance of the message the client can go about its other business. Grapevine then expands any groups specified as recipients to produce the complete set of individuals that are to receive the message and delivers the message to an inbox for each.

Message polling:

[individual] → {empty, nonempty}

Message polling is used to determine whether an individual's inboxes contain messages that can be retrieved. We chose not to authenticate this function so it would respond faster and load the Grapevine computers less.

Retrieve messages:

[name, password] → sequence of messages → ok

The client presents an individual's name and password. If the password authenticates the individual then Grapevine returns all messages from the corresponding inboxes. When the client indicates "ok," Grapevine erases these messages from those inboxes.

Grapevine's authentication, access control, and resource location services are implemented by the remaining functions. These are called the *registration service*, because they are all based on the registration data base.

Authenticate:

[individual, password] → {authentic, bogus}

The authentication function allows any client to determine the authenticity of an individual. An individual/password combination is authentic if the password matches the one in the individual's registration data base entry.[1]

Membership:

[name, group] → {in, out}

Grapevine returns an indication of whether the name is included in the group. Usually the client is interpreting the group as an access control list. There are two forms of the membership function. One indicates direct membership in the named group; the other indicates membership in its closure.

Resource location:

[group] → members
[individual] → connect site
[individual] → ordered list of inbox sites

The first resource location function returns a group's membership set. If the group is interpreted as a distribution list, this function yields the individual recipients of a message sent to the distribution list; if the group is interpreted as the name of some service, this function yields the names of the servers that offer the service. For a group representing a service, combining the first function with the second enables a client to discover the internet addresses of machines offering the service, as described in Sec. 5. The third function is used for message delivery and retrieval as described in Sec. 4.

Registration data base update and inquiry:

There are various functions for adding and deleting names in the registration data base, and for inspecting and changing the associated values.

3.3 Registries

We use a partitioned naming scheme for RNames. The partitions serve as the basis for dividing the administrative responsibility, and for distributing the data base among the Grapevine computers. We structure the name space of RNames as a two-level hierarchy. An RName is a character string of the form *F.R* where *R* is a *registry* name and *F* is a name within that registry. Registries can correspond to organizational, geographic, or other arbitrary partitions that exist within the user community. A two-level hierarchy is appropriate for the size and organizational complexity of our user community, but a larger community or one with more organizational diversity would cause us to use a three-level scheme. Using more levels would not be a fundamental change to Grapevine.

[1] This password-based authentication scheme is intrinsically weak. Passwords are transmitted over the internet as clear-text and clients of the authentication service see individuals' passwords. It also does not provide two-way authentication: clients cannot authenticate servers. The Grapevine design includes proper encryption-based authentication and security facilities that use Needham and Schroeder's protocols [9] and the Federal Data Encryption Standard [8]. These better facilities, however, are not implemented yet.

3.4 Distribution of Function

As indicated earlier, Grapevine is implemented by code that runs in dedicated Grapevine computers, and by code that runs in clients' computers. The code running in a Grapevine computer is partitioned into two parts, called the *registration server* and the *message server*. Although one registration server and one message server cohabit each Grapevine computer, they should be thought of as separate entities. (Message servers and registration servers communicate with one another purely by internet protocols.) Several Grapevine computers are scattered around the internet, their placement being dictated by load and topology. Their registration servers work together to implement the registration service. Their message servers work together to implement the delivery service. As we will see in Secs. 4 and 5, message and registration services are each clients of the other.

The registration data base is distributed and replicated. Distribution is at the grain of a registry; that is, each registration server contains either entries for all RNames in a registry or no entries for that registry. Typically no registration server contains all registries. Also, each registry is replicated in several different registration servers. Each registration server supports, by publicly available internet protocols, the registration functions described above for names in the registries that it contains. Any server that contains the data for a registry can accept a change to that registry. That server takes the responsibility for propagating the change to the other relevant servers.

Any message server is willing to accept any message for delivery, thus providing a replicated mail submission service. Each message server will accept message polling and retrieval requests for inboxes on that server. An individual may have inboxes on several message servers, thus replicating the delivery path for the individual.

If an increase in Grapevine's capacity is required to meet expanding load, then another Grapevine computer can be added easily without disrupting the operation of existing servers or clients. If usage patterns change, then the distribution of function among the Grapevine computers can be changed for a particular individual, or for an entire registry. As we shall see later this redistribution is facilitated by using the registration data base to describe the configuration of Grapevine itself.

The code that runs in clients' machines is called the *GrapevineUser package*. There are several versions of the GrapevineUser package: one for each language or operating environment. Their function and characteristics are sufficiently similar, however, that they may be thought of as a single package. This package has two roles: it implements the internet protocols for communicating with particular Grapevine servers; and it performs the resource location required to choose which server to contact for a particular function, given the data distribution and server availability situation of the moment. GrapevineUser thus makes the multiple Grapevine servers look like a single service. A client using the GrapevineUser package never has to mention the name or internet address of a particular Grapevine server. The GrapevineUser package is not trusted by the rest of Grapevine. Although an incorrect package could affect the services provided to any client that uses it, it cannot affect the use of Grapevine by other clients. The implementation of Grapevine, however, includes engineering decisions based on the known behavior of the GrapevineUser package, on the assumption that most clients will use it or equivalent packages.

3.5 Examples of How Grapevine Works

With Fig. 2 we consider examples of how Grapevine works. If a user named *P.Q* were using workstation 1 to send a message to *X.Y.*, then events would proceed as follows. After the user had prepared the message using a suitable client program, the client program would call the delivery function of the GrapevineUser package on workstation 1. GrapevineUser would contact some registration server such as *A* and use the Grapevine resource location functions to locate any message server such as *B*; it would then submit the message to *B*. For each recipient, *B* would use the resource location facilities, and suitable registration servers (such as *A*) to determine that recipient's best inbox site. For the recipient *X.Y*, this might be message server *C*, in which case *B* would forward the message to *C*. *C* would buffer this message locally in the inbox for *X.Y*. If the message had more recipients, the message server *B* might consult other registration servers and forward the message to multiple message servers. If some of the recipients were distribution lists, *B* would use the registration servers to obtain the members of the appropriate groups.

When *X.Y* wishes to use workstation 2 to read his mail, his client program calls the retrieval function of the GrapevineUser package in workstation 2. GrapevineUser uses some registration server (such as *D*) that contains the *Y* registry to locate inbox sites for *X.Y*, then connects to each of these inbox sites to retrieve his messages. Before allowing this retrieval, *C* uses a registration server to authenticate *X.Y*.

If *X.Y* wanted to access a file on the file server *E* through some file transfer program (FTP) the file server might authenticate his identity and check access control lists by communicating with some registration server (such as *A*).

3.6 Choice of Functions

The particular facilities provided by Grapevine were chosen because they are required to support computer mail. The functions were generalized and separated so other applications also could make use of them. If they want to, the designers of other systems are invited to use the Grapevine facilities. Two important benefits occur, however, if Grapevine becomes the *only* mechanism for authentication and for grouping individuals by organization, interest, and function. First, if Grapevine per-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.