Technical Report

UCAM-CL-TR-168
ISSN 1476-2986

Number 168

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Distributed computing with a processor bank

J.M. Bacon, I.M. Leslie, R.M. Needham

April 1989

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

© 1989 J.M. Bacon, I.M. Leslie, R.M. Needham

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Distributed Computing with a Processor Bank

J M Bacon, I M Leslie and R M Needham

University of Cambridge Computer Laboratory,
New Museums Site, Pembroke Street, Cambridge CB2 3QG, UK

The Cambridge Distributed Computing System (CDCS) was designed some ten years ago and was in everyday use at the Computer Laboratory until December 1988. An overview of the basic design of CDCS is given, an outline of its evolution and a description of the distributed systems research projects that were based on it. Experience has shown that a design based on a processor bank leads to a flexible and extensible distributed system.

1. The Cambridge Distributed Computing System

CDCS [Needham82] was designed some ten years ago and was in everyday use, as the main research environment at the Computer Laboratory, over many years. It is based on the Cambridge Ring (CR) local area network and employs the "pool of processors" approach to distributed computing. CDCS provides a number of common services such as file storage and printing, which may be invoked from the heterogeneous systems running in processor bank machines. It also provides a naming and authorisation infrastructure and management of the processor bank.

Typical usage of CDCS is for a user, via a terminal server, to ask the Resource Manager for a processor from the processor bank. The user specifies a software system to be loaded into the acquired processor, is authenticated to this system and to CDCS and then runs applications on this single machine. Although the user is free to acquire more than one machine, CDCS originally provided virtually no support for users to spread a task across a number of machines. Processor bank machines may be loaded with public operating systems, which may be used to run public utilities, or private research systems.

1.1. Evolution of CDCS

CDCS was initially implemented on a single Cambridge Ring and was extended to operate over three bridged Cambridge Rings on two sites. Project Universe [Leslie 84] further extended the basic system design to CR-based systems connected by satellite over a wide area.

The program development environment provided through the processor bank and file server, which for practical reasons were small research systems, was augmented by two Vax Unix^{TM} systems, with local discs, and a number of Ethernet-based MicroVax2s. Sun and Xerox distributed systems also use the Ethernet [figure1]. The recent development of the Cambridge Fast Ring (CFR) gives potential for systems research based on an order of magnitude increase in LAN speed.

CDCS was originally programmed in an ad-hoc way and in 1982 the Mayflower project was set up to develope a language and environment for programming distributed services and applications. A concurrent programming language Concurrent CLU (CCLU) including a language level remote procedure call (RPC) facility was produced and used for development of a number of services, tools and applications.

2. CDCS Design Overview

2.1 LAN Medium and Protocol Hierarchy

The Cambridge Ring is a slotted, 10Mbps ring with anti-hogging and indication of success or failure of delivery at the lowest (minipacket) level [Wilkes 79]. The recent Cambridge Fast Ring, designed to operate at 100Mbps follows this tradition but with a larger packet size [Hopper 88]. A protocol hierarchy was designed for CDCS with a Basic Block protocol above the CR and a choice of a Single Shot (request-response or transaction) Protocol and a Byte Stream Protocol at the next level. Above this, specialised application protocols were developed for sevice bootstrapping, file service and garbage collection, above SSP and for terminal connections above BSP. CDCS was an early example of the use of lightweight protocols in a reliable, high speed, high bandwidth LAN environment.

2.2 Software System Structure

A processor bank may contain heterogeneous hardware which may be loaded with heterogeneous software systems. Each system may have its own implementation of naming, protection, reliability, etc. An aim of CDCS was to allow such systems to share common services such as printing and disc storage and common utilities such as electronic mail.

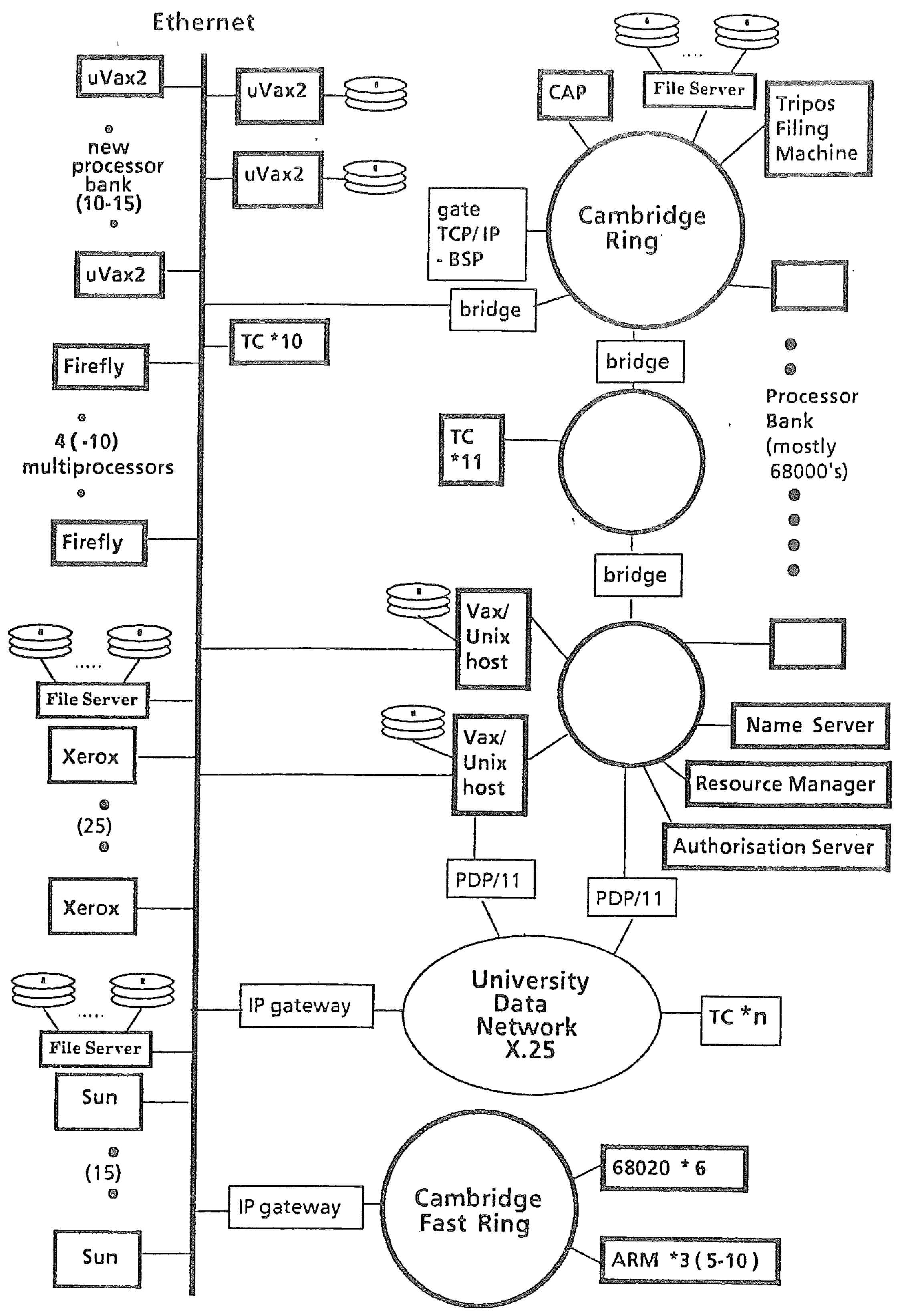

FIGURE 1. Part of the Distributed Computing Research Environment (to December 88). The Cambridge (old) Ring world is now being dismantled.

The basic model of CDCS software is *client-server,* for invocation of system-provided services. Initially, a simple transaction protocol was used for service invocation (SSP), comparable with other Distributed Systems developed at the time [Shrivastava 82, Mullender 86]. The Mayflower project went on to provide language level service invocation, CCLU with RPC, with compile time type checking and run time consistency checking of RPC arguments, which may include user-defined types.

2.3 Naming of Services

The system infrastructure supports the naming of services in the form of flat text names which are mapped to network address (ring station) and port number. As with other early systems [Tanenbaum 85] there is no provision for a set of servers at different network addresses cooperating to offer a service. A name server at a well known address, a fixed station number for any CR in a system, provides the mapping. In principle, name server lookup is part of every interaction to avoid embedding network addresses in programs. In practice, the result of a name server lookup is cached and used as a hint for subsequent invocations of a service. The name server approach was used successfully in Project Universe [Leslie 84] for a number of CR based LANs linked by satellite.

The name server is small and holds name to address mappings of system services which change very infrequently. Its use was not extended to support the more dynamic requirements of binding RPC calls associated with for example, distributed services being developed, utilities such as mail or distributed compilers, or distributed applications.

2.4 Naming and Protection of Objects in the Cambridge File Server (CFS)

The file storage service (CFS) is the major CDCS service concerned with object management. This "universal file server" [Birrell 80, Dion 80,81] is designed to be used by any number of different clients' file directory servers, each with its own text naming conventions and access control policies. It does however support existence control and concurrency control. Its major client in practice was the Tripos Operating System [Richards 79], designed as a single user system and appropriate for loading into processor bank machines. The CAP file system was also replaced by CFS.

CFS thus provides a low level storage service. It supports the primitive types *byte* and *id* and the single type constructor *sequence.* A file is a sequence of bytes and an *index* is a sequence of id's. Every file and every index has an *id.* Indexes may be used by clients to mirror their directory structures. The CFS interface is such that creation of an object is combined with storing its *id* in an index. The CFS existence control policy, that an object exists while it is reachable from the root, is supported by a garbage collection mechanism which CFS initiates to run asynchronously in a processor bank machine.

CFS was designed to support filing systems as its clients. It may also be used directly by services or utilities and an evaluation of its design for this style of use is given in the references cited in section 3 .

Mandatory concurrency control, typically multiple reader single writer locks on whole files, is often argued to be inappropriate at the lowest level of distributed systems, since clients will often be multiple instances of the same program, able to synchronise their access to shared objects at a higher level. A more flexible approach is to provide a separate lock primitive and possibly to support finer grained concurrent write sharing [Burrows 88].

The Tripos directory service was initially provided as part of the Tripos system loaded into processor bank machines. Such systems cannot be trusted to carry out access control. The Tripos Filing Machine [Richardson 83,84] was developed to function both as a trusted directory server for Tripos systems and as a caching machine to improve the performance of the file service.

2.5 Authorisation for Service Use

Each processor bank operating system carries out its own user authentication but must register its current user with a CDCS authorisation server, the Active Object Table manager (AOT). AOT issues a session key, with a random component, which, together with information on the category of the user, functions as a capability for service use, in that a server may check with AOT that a request comes from an authenticated user. Users loading private software systems into processor bank machines must also be authorised to use public services by registering with AOT.

2.6 Reliability

Each service may take its own independent approach to reliability, for example, CFS provides atomic transactions on special files, typically used for client file system metadata.

The infrastructure was designed for rapid rebooting through the boot server. The network interfaces and node software provide facilities for remote control and debugging.

A dead man's handle technique is used to monitor allocated processor bank systems.

The CCLU RPC system is a communications facility and is not concerned with application level issues such as orphan extermination and server restart. The default RPC semantics are *at most once* but an option may be specified which offers *exactly once* semantics in the absence of server crashes and prolonged network failure.

3. CCLU and the Mayflower Project

The Mayflower project was set up in 1982 to provide an environment for developing and running distributed applications and services. It comprises a language, CCLU, a communications protocol, RPC, integrated into the language system, and an operating system, the Mayflower supervisor. Details are given in [Hamilton 84] and [Bacon 87].

3.1 Sequential CLU

CLU [Liskov81] was selected and extended by the Mayflower group. CLU is object orientated in style and provides procedures for procedural abstraction, iterators for control abstraction and clusters for data abstraction. It is strongly typed and supports user defined abstract types. It has separate compilation facilities and the compiler generates and checks interface specifications. Parameterised clusters go some way towards providing the facilities usually associated with a polymorphic typing system.

3.2 Concurrency Features in CCLU

CLU was extended with a fork primitive to support the dynamic creation of lightweight processes sharing an address space, allowing the construction of high performance services with internal concurrency (multi-threaded servers). Semaphores and monitors were added for process synchronisation. It was found, in a systems environment, that classical monitors unduly restrict concurrency in large systems [Cooper 85] and a critical region construct, with programmer specified locking, was added. This allowed the critical code to be kept to a minimum while modular structure was maintained by the use of clusters.

3.3 CCLU RPC

CCLU RPC is a type-checked, type-safe, language-level construct incorporating dynamic binding under program control. Arbitrarily complex objects of practically any type in the CLU language, including user-defined abstract types, can be passed in arguments to RPCs. The philosophy of Mayflower RPC is not to hide from the programmer that certain processing is remote and the peculiar semantics and overhead of remote operations are made explicit. The language was therefore modified to add new syntax for the definition and call of remote procedures, rather than using the method of preprocessing and stub generation often associated with transparent RPC [Birrell 84]. The programmer may use the default communications semantics of *at most once* or may select reliable *exactly once* semantics in the absence of node crashes and prolonged network failure. CCLU RPC operates through bridges and across a ring-ethernet gateway.

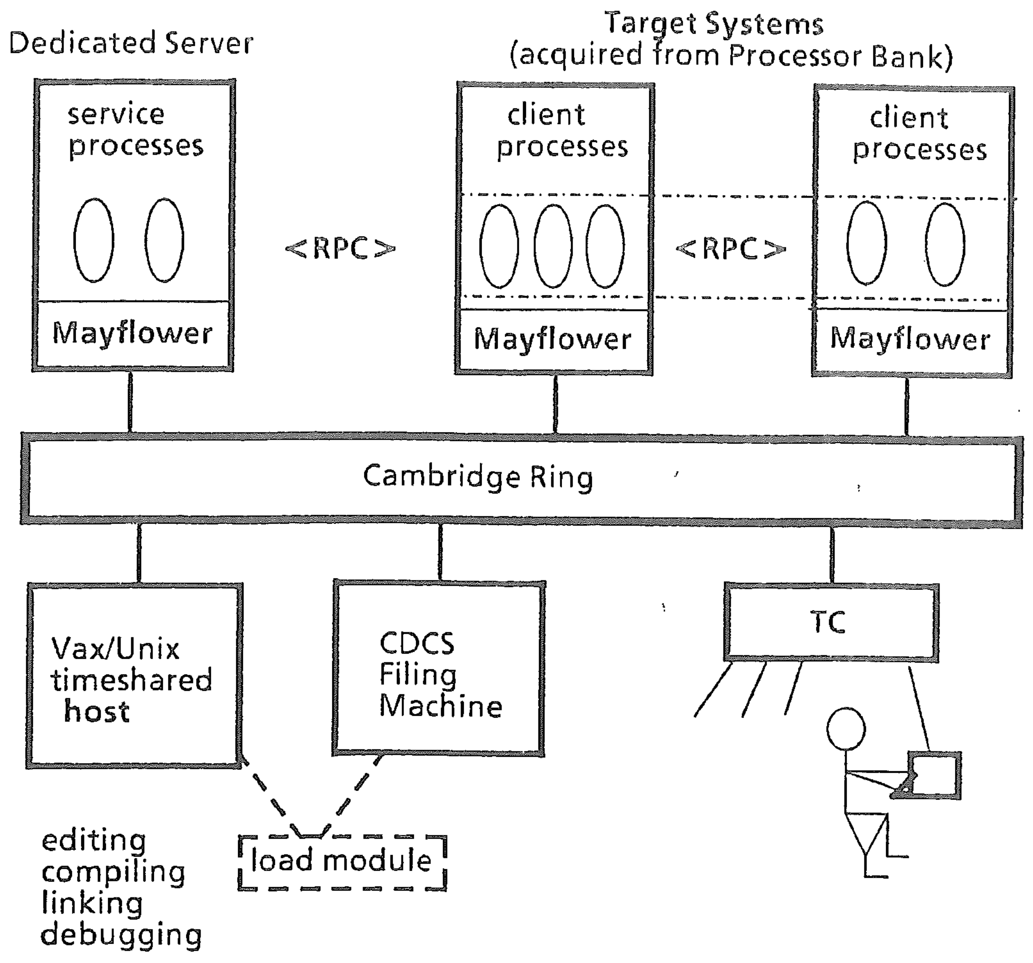

FIGURE 2. Mayflower on CDCS

3.4 The Mayflower "Lightweight" Kernel

The Mayflower kernel was designed for implementing high performance services for distributed systems. It supports lightweight processes running in a shared address space (a Mayflower domain) hence supporting what are often called multi-threaded servers. Resources are allocated to a domain and are shared by all processes therein. Multiple domains per node may be used but inter-domain communication is by (expensive) RPC.

System development is carried out in CCLU on (Micro)Vaxes under Unix and target code runs on processor bank MC68000s under the Mayflower kernel and, for preliminary testing, on Vaxes under Unix. CDCS infrastructure, services and applications can be written in CCLU to run under Mayflower and have RPC interfaces [figure 2]. The CDCS authorisation server AOT was reimplemented and a new processor bank manager was designed [Craft 85].

3.5 Multiple Transport Protocols

CDCS had started out with heterogeneity as a central design aim. The initial phase of Mayflower produced a single language subsystem within CDCS; CCLU over the Mayflower supervisor on 68000's. Recent work has extended the RPC system to allow interworking between CCLU programs running on ring based 68000's over Mayflower and on Ethernet based Microvaxes over UNIX. RPC runs over UDP (User Datagram Protocol) and IP (Internet Protocol) on the Ethernet and over the Basic Block protocol on the Cambridge Ring.

The transport protocol required is selected at RPC bind time and the network address now includes network type as well as a network specific address. RPC gateways can be written in CCLU and other networks are easy to add.

3.6 Multiple Data Formats

Some interworking was also seen to be desirable between programs written in CCLU running over Mayflower and programs running on Xerox or Sun workstations written to use Xerox Courier RPC and Sun RPC respectively. It was found that a subset of the types supported by CCLU RPC, the built in immutable types, are used in Sun XDR (eXternal Data Representation standard) and the Courier data standard. CLU RPC was therefore extended to allow selection of any one of these data representations.

CCLU clients can access existing Xerox and Sun services, UNIX and Xerox XDE (Xerox Development Environment) and Interlisp clients can access CCLU servers and limited support is provided for new multi language distributed applications.

4. Distributed Systems Research on CDCS

4.1 Processor Bank Management and Dynamic Software Configuration

In the original CDCS design each available system configuration was held as a single, fully-linked load module in the file server and was loaded into processor bank machines under the control of the Resource Manager (RM). This is reasonable for the small number of system services originally available but is undesirable as a basis for research into extensible, heterogeneous systems with dynamically loaded utilities and applications. Craft's RM [Craft 83,85] proposes multi-level resources which may consist of several software layers on a variety of hardware and which are configured dynamically when requested. One or more processor bank machines with associated software may be requested, or a session on a shared system such as a Unix system. Automatic preloading of commonly used systems avoids delay to users.

Allocated resources are monitored by an Aliveness Server using a dead-man's handle mechanism and interested parties are notified of any events in which they have registered an interest by an Event Notification Server.

4.2 A Debugger for Distributed Concurrent Programs [Cooper 87, 88]

A debugger supporting source level debugging of distributed CCLU programs in the target environment under real conditions of use was implemented in CCLU. Research issues include the policies and mechanisms associated with breakpointing processes of a distributed computation, determining the operating system and run-time library support desirable for implementing this kind of debugger, and defining a more appropriate format than flat files in which to store debugging information produced by compilers and linkers.

A small debug agent is included at every CCLU node (within the Mayflower kernel or within the run-time system for CCLU on the Unix emulation of Mayflower) and a small amount of statistical data, for example on the most recent RPC calls, is gathered. The relevant agents are activated from a debugger machine acquired from the processor bank when the program is to be debugged. The policy on breakpoints is that all processes of a computation should be halted "instantly" when any breakpoint occurs.

Special emphasis is given to debugging system services, both before and after installation, and to the fact that a program being debugged will be using shared services. A new service or a new version of an existing service may be installed on a processor bank machine under the control of the debugger. It may be tested, at first with an artificially generated test load, and may subsequently be released to real clients. It is argued that services should be written to compute the elapsed time of their clients excluding the time halted at breakpoints.

4.3 A Distributed Mail System [Brooks 88]

The processor bank model is particularly suited to providing network services which have very uneven resource requirements such as mail. The system load from mail has infrequent peaks so that timesharing systems degrade when mail is being processed and dedicated mail servers remain idle for substantial times. Users' private workstations cannot be trusted to run mail since the software could be modified to allow confidential mail to be read into the workstation or mail to be altered. The approach used in this research is to have a small dedicated server as a permanent mail daemon augmented by processors from the pool at times of high load. A mechanism for dynamically acquiring machines is required for this type of application and, ideally, for releasing some if they are requested by higher priority tasks. A processor bank machine, unlike a workstation, can be loaded with trusted, unmodifiable software which provides the mail program interface.

The distributed mail service uses the CFS interface directly and this has provided useful experience for the design of future network storage services. A read only capability is required for mail objects, for example.

4.4 A Distributed Compilation System [Wei 89]

A distributed compilation system was developed for CCLU in which compilation of separate modules is assigned to individual processors under the direction of a controller server of which there is one per active user. A library server maintains the interface specifications of library clusters and of compilation units which are written by the compiler instances when a unit is compiled. These specifications are read by the compiler instances when external calls are being type checked. The library server must also ensure that the concurrent compilations carried out by several users do not conflict, for example, when the specification for a module is changed as it is being used in the compilation of other modules.

As well as contributing to research in distributed compilation this project provides an example of the requirement for a minimum configuration of three machines but the possibility of dynamically acquiring other free machines for both compilation servers and library servers, and releasing them to higher priority tasks when requested.

4.5 A Distributed File Directory Service [Seaborne 88]

The provision of file naming and directory facilities for heterogeneous operating systems connected by a LAN was investigated. A replicated and distributed directory service was developed in prototype form in CCLU under Mayflower on processor bank machines. The directory service is designed to function as a universal directory service and is located above a low level file storage service such as CFS.

The design is based on a generalised directory structure and a generalised pathname comprising a list of components. New directories may be created and attached to the naming graph. A directory entry is a string, naming an inferior file or directory object together with an access control list and a unique identifier for the object. The directory service returns to the client the identifier of the file or directory named by the full pathname specified in the request.

The service is implemented as a set of servers and a primary site cache management policy is used for maintaining consistency of the directories cached by the service instances.

File servers hold the contents of files and the transfer of the contents only involves the clients and the file servers.

4.6 Project Universe [Leslie 84]

In the Universities Extended Ring Satellite Experiments (Universe) project, a number of CR based LANs running CDCS software were connected by a 1Mbit/s satellite broadcast channel [figure 3].

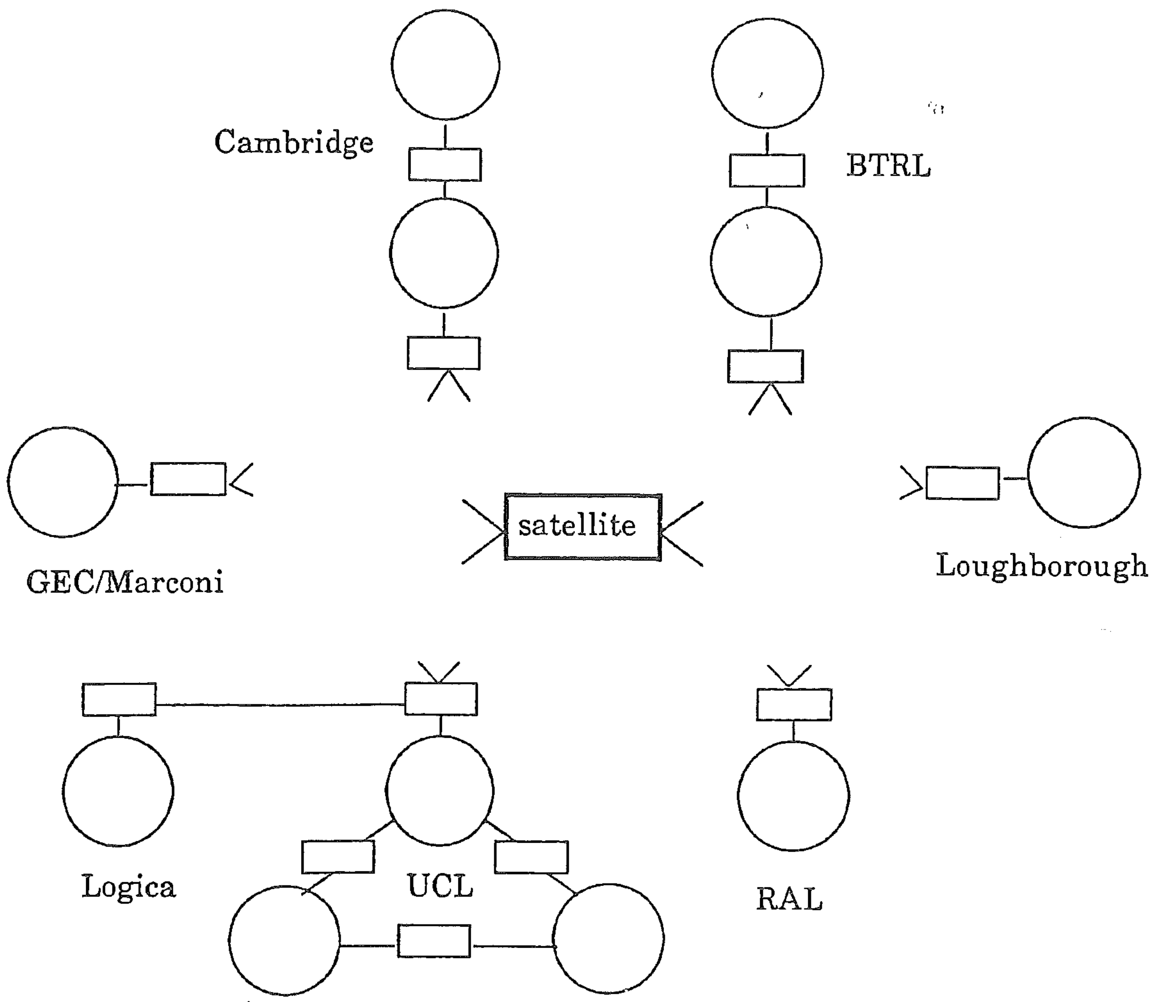

FIGURE 3. The Universe Network

To avoid delay in gateways and therefore to be able to regard the whole network as an efficient distributed system, a single network rather than an internet was constructed. Experiments included remote bootstrapping of trusted system software; use of the Cambridge file servers as a remote storage service for multiple instances of Tripos Filing Machines; and transfer of voice and video.

4.7 A Reliable, Distributed Telephone Exchange [Want 88]

The ISLAND project at the laboratory studied the provision of integrated services on a local area network. Transporting data, voice and video in the same network has the practical advantage of minimising wiring and also allows applications to integrate these media. The intended network medium for this work was the CFR but prototype systems were developed on CDCS.

The control and management of a network telephone is in the domain of distributed computing in such a system. The voice connections between telephones are virtual circuits. Control and data information can be freely mixed with voice at a network interface. The new problems that result are the management issues relating to the distributed control of the real time media. Digital ring phones providing a number of novel services were built and controlled from personal workstations and a reliable, distributed PABX was implemented in CCLU over Mayflower on processor bank machines.

4.8 A Voice Storage and Editing Service [Calnan 88]

CCLU was also used for work under the ISLAND project concerning the storage and editing of encoded voice. A voice storage service was designed, comprising a protocol and an interface to CFS, which has sufficient performance to support real time working for telephone conversations. A voice editor was also designed, implemented and tested. Encoded voice is divided into phrases, delimited by "silences" (background noise) and each phrase is stored as a file. Integration of text and voice is also supported.

The fact that a new, specialised storage service had to be used for real time performance rather than existing file services is relevant to the design of future network storage services.

Following experience in ISLAND and the UNISON project [Tennenhouse 87], the Laboratory is about to carry out further research in multimedia systems, including video as well as voice, based on the CFR.

5. Current Work, Summary and Conclusions

A major advantage of the processor bank approach is that new systems may be made available to users as technology evolves without any change in the underlying system. Also, the model of independently managed subsystems sharing common services is widely applicable. Some ten years of experience with CDCS have shown that the two major functions provided above the basic communications infrastructure, processor bank management and support for service invocation by heterogeneous, independently managed subsystems, form a good basis for distributed system design.

Although single user, diskless systems were originally envisaged in the processor bank, a range of configurations may be accommodated. Special purpose hardware may be included, several systems may be acquired to run a parallel application and a range of operating systems may be made available to users. A new processor bank would now include the DEC Firefly shared memory multiprocessors, currently used as research systems. A system model under investigation proposes a high quality graphics terminal per user, rather than a conventional terminal or workstation, supported by processing resources available across the network [Dixon 88].

CDCS had started out with heterogeneity as a central design aim. The Mayflower project produced a single language subsystem within CDCS; CCLU over the Mayflower supervisor on 68000's. Subsequent work allowed a number of transport protocols and data formats to be selected. Current distributed systems research is making use of CCLU but a new lightweight kernel, capable of running on VAX architectures, including multiprocessors, as well as M68000's is being developed.

After some years of experience with CCLU RPC we feel that non-transparent syntax best reflects the realities of an environment comprising distributed programs. Finer control over timeout and retry strategies than those provided may be desirable. A fully type checked high level language facility is highly desirable. Extension of CCLU to experiment with object mechanisms such as inheritance and dynamic linking is in progress [Hailes 88].

A number of research projects have used CCLU on CDCS and have illustrated that the processor bank approach to distributed computing, supported by system and language level mechanisms, provides a good basis for distributed computing. The processor bank provided a testbed for research into distributed implementations of services and into the possibility of acquiring extra machines dynamically in response to load on a particular service.

A new processor bank comprising Ethernet-based Microvaxes is now used. Various software systems are offered including Ultrix, VMS and Amoeba and a new lightweight kernel is being developed. Distributed Systems simulation experiments are currently being programmed in CCLU on the new processor bank [Dickman 88]. Other research projects which follow directly from the experience gained in CDCS include the design of network storage services [Thomson 87, Wilson 87], naming services [Ma 88] and monitoring services [Lam 88]. For current large scale systems each service must be designed as a set of servers and again, processor bank machines are being used to prototype distributed service implementations.

Acknowledgements

This paper presents work of the Systems Group at the Computer Laboratory over many years. Some individuals' work is referenced above and many others have contributed.

References

[Bacon 87] Bacon J M and Hamilton K G, "Distributed Computing with RPC: The Cambridge Approach", Proc IFIPS conference on Distributed Processing, eds. Barton M et al. 355-369, North Holland 1988

[Birrell 80] Birrell A D and Needham R M, "A Universal File Server" IEEE Trans SE, SE-6 (5), 450-453, Sept 80

[Birrell84] Birrell A D and Nelson B J, "Implementing Remote Procedure Call" ACM Transactions on Computer Systems 2(1), 39-59, Feb 84

[Brooks 88] Brooks P M, "Distribution of Functions in Computer Networks" University of Cambridge submitted PhD thesis, 1988

[Burrows 88] Burrows M, "Efficient Data Sharing" University of Cambridge PhD thesis and TR 153, 1988

[Calnan 88] Calnan R, " The Integration of Voice within a Digital Network" University of Cambridge submitted PhD thesis, 1989

[Cheriton84] Cheriton D R, "The V Kernel: A Software Base for Distributed Systems" IEEE Software, 1(2), April 84

[Cooper 85] Cooper R C B and Hamilton K G "Preserving Abstraction in Concurrent Programming" IEEE Trans SE, SE14(2), 258-262, Feb 88, and University of Cambridge Computer Laboratory TR 76, August 1985

[Cooper 87] Cooper R C B, "Pilgrim: A Debugger for Distributed Systems" Proc IEEE 7th ICDCS, Berlin 1987

[Craft 83] Craft D H, "Resource Management in a Decentralised System" ACM SOSP9, Operating Systems Review 17(5), 11-19, Oct 1983

[Craft 85] Craft D H, "Resource Management in a Decentralised System" PhD thesis, University of Cambridge 1985, TR 73

[Dickman 88] Dickman P J, Thesis proposal 1988

[Dion 80] Dion J, "The Cambridge File Server" ACM Operating Systems Review, 14(4), 26-35, Oct 80

[Dion 81] Dion J, "Reliable Storage in a Local network" University of Cambridge, PhD thesis, 1981

[Dixon 88] Dixon J, Thesis proposal 1988

[Hailes 88] Hailes S M V, Thesis proposal 1988

[Hamilton 84] Hamilton K G , "A Remote Procedure Call System" University of Cambridge PhD thesis, TR 70, 1984

[Hopper 88] Hopper A and Needham R M "The Cambridge Fast Ring Networking System". IEEE Trans on Computers Sept 88

[Lam 88] Lam K Y, Thesis proposal 1988

[Leslie 84] Leslie I M et al, "The Architecture of the Universe Network", Proc ACM Sigcomm 84, CCR 14(2), 2-9, June 84

[Liskov 81] Liskov B et al , "CLU Reference Manual" Springer Verlag, LNCS 114, 1981

[Ma 88] Ma C, Thesis proposal 1988

[Mullender86] Mullender S J and Tanenbaum A S, "The Design of a Capability Based Distributed Operating System" Computer Journal, 29(4), 289-300, March 86

[Needham 82] Needham R M and Herbert A H , "The Cambridge Distributed Computing System" Addison Wesley 1982.

[Richards 79] Richards M et al, "Tripos - A Portable, Real -time Operating System" Software, Practice and Experience, 9, 513-526, 1979

[Richardson 83] Richardson M F and Needham R M, "The Tripos Filing Machine" ACM Operating Systems review, 17(5), 120-128, 1983

[Richardson 84] Richardson M F, "Filing System Services for Distributed Computer Systems", University of Cambridge PhD thesis, 1984

[Seaborne 87] Seaborne A F, "Filing in a Heterogeneous Network" University of Cambridge PhD thesis , 1988

[Shrivastava 82] Shrivastava S and Panzieri F, "The Design of a Remote Procedure Call Mechanism", IEEE Trans Computers, 31(7), July 82

[Tanenbaum 85] Tanenbaum A S and van-Renesse R, "Distributed Operating Systems" ACM Computing Surveys 17(4), 419-470, Dec 85

[Tennenhouse 87]Tennenhouse D L et al, "Exploiting Wideband ISDN's: The Unison Exchange", Proc IEEE Infocom 87, 1018 - 1026, 1987

[Thomson 87] Thomson S, Thesis proposal 1987

[Want 88] Want R, " Reliable Management of Voice in a Distributed System" University of Cambridge PhD thesis, and TR 141, 1988

[Wei 87] Wei M, "Distributed Compilation" University of Cambridge PhD thesis in preparation, 1988

[Wilkes 79] Wilkes M V and Wheeler D J, "The Cambridge Digital Communication Ring" Local Area Communications Network Symposium (sponsors MITRE corp. & NBS), Boston, Mass. May 1979.

[Wilson 87] Wilson T D, Thesis proposal 1987