Student presentations
December 2 and 4, 1987

A distributed `make` *with remote task execution*
Tracy Hoover, Ryutarou Ohbuchi, and Suresh Rajgopal

We have developed a program which processes a given `Makefile` and executes the tasks in the file remotely on a given set of remote machines. A naive scheduler allocates parcels of concurrent tasks to dispatchers (corresponding to remote machines) which then dispatch them to be executed on these machines. The dependency graph of tasks in the `make` is maintained, and onward processing from a node stops when a task fails to execute successfully.

Network utilities, part 1
Jim Chou

A network manager consisting of various utilities, minus a good user interface.

MS: A Moderated Session
Claire Durand, Wendy Kuchkuda, and Cathie Vishnevsky

MS provides a conferencing environment geared towards formal discussions with a designated leader. It is implemented with virtual circuits; we are using X Windows for our user-interface.

LNET: Message/Mail/File Print Spooling System
Steven M. Miller

The LNET Spooling System provides a distributed message, mail, file and print transport mechanism for LAN attached workstations. LNET has been implemented for PC's running PC-DOS (MS) and the NetBIOS communications interface for the token ring network. LNET consists of a server program and a collection of client/user applications and protocol handlers. Data transport is provided via a store and forward component of the server. All access to the LNET system is controlled by user verification and authorization by the server. Internetwork connectivity is enabled through gateway workstations appearing as privileged users to the system.

Rtools: Tools for Software Maintenance in a Distributed Computing Environment
Helen Harrison, Stephen Shaefer, and Terry Yoo

In an environment with large quantities of source code to maintain and multiple heterogeneous architectures, effective software maintenance requires more that the standard `makefile` system for creating and installing binaries. We have implemented an alternative which breaks the problem into 3 pieces. First a database contains all information necessary to compile and install binaries for each software subsystem. Second, using information from the database and related FreedomNet tools, a compilation is done to produce binaries for each architecture. Compiling binaries for other architectures is unknown to local makefiles. Third a distribution utility takes information from the database and produces an rdist script to install the binaries in the correct place on all machines.

Distributed system for executing data flow programs
Geof Alexander

Highlights of the system are the ability to:

- dynamically define the data flow language
- dynamically invoke subprograms (subgraphs)
- specify which "code" is distributed
- compile programs for fast execution
- specify which machines are available for executing data flow programs

Plexus: a multiplexed reliable communications package
Scott Southard and Robert B. Stam

Our project consists of designing and building a communications package to provide multiple reliable channels between programs running under UNIX and "tasks" running on a Macintosh, using an RS232 connection. The ultimate goal is to use Plexus as the basis of a multiwindowed multifunction terminal emulator on the Macintosh, that would allow multiple terminal sessions, and bidirectional file transfers, all running simultaneously.

Boatrace
James Chung and Azeemullah Khan

An XTrek-like game in which each contestant controls his/her own sailboat through an XWindow interface. The race course and the positions of all racers are displayed and updated in real time.

Remote Ikonas frame buffer and analog input device interfaces
John Airey and David Ellsworth

The software interfaces to the Ikonas graphics device and the analog input devices, sliders joysticks etc, that previously existed only on Grumpy have been made available on other machines. The Ikonas libraries use a TCP protocol while the analog input device library uses a datagram protocol.

A Distributed System To Serve Remote Collaborative Sessions
Sheng-Uei Guan, Joey Hughey, and Jih-Fang Wang

Participants who wish to collaborate synchronously from remote workstations are able to interact cooperatively through the use of a special communications server daemon. A chairman is responsible for the creation of the session and its initialization with relevant control information, which includes the list of participants, the agenda, and the degree of session confidentiality. The remaining participants must acquire access, then each is provided a tool window and a control window at their individual workstation. The possession of a token is used to regulate a participant's access to the tool window, which is displayed to all simultaneously. The control window can be used to request and release access to the tool window, to communicate either publically or privately to individual participants, and to display the agenda.