Experimental Operating System Lab on a Dime

Dennis Brylow
Department of Mathematics,
Statistics and Computer Science

In theory, simulating an Operating System is the same as the real thing. In practice, it is not.

Setting up specialized Operating System Laboratory hardware usually requires more time, money, and space than most schools can afford.

Marquette's Systems Lab has concentrated specifically on developing a scalable, duplicable design for an Experimental Operating System Laboratory environment with minimal cost and very low space requirements.

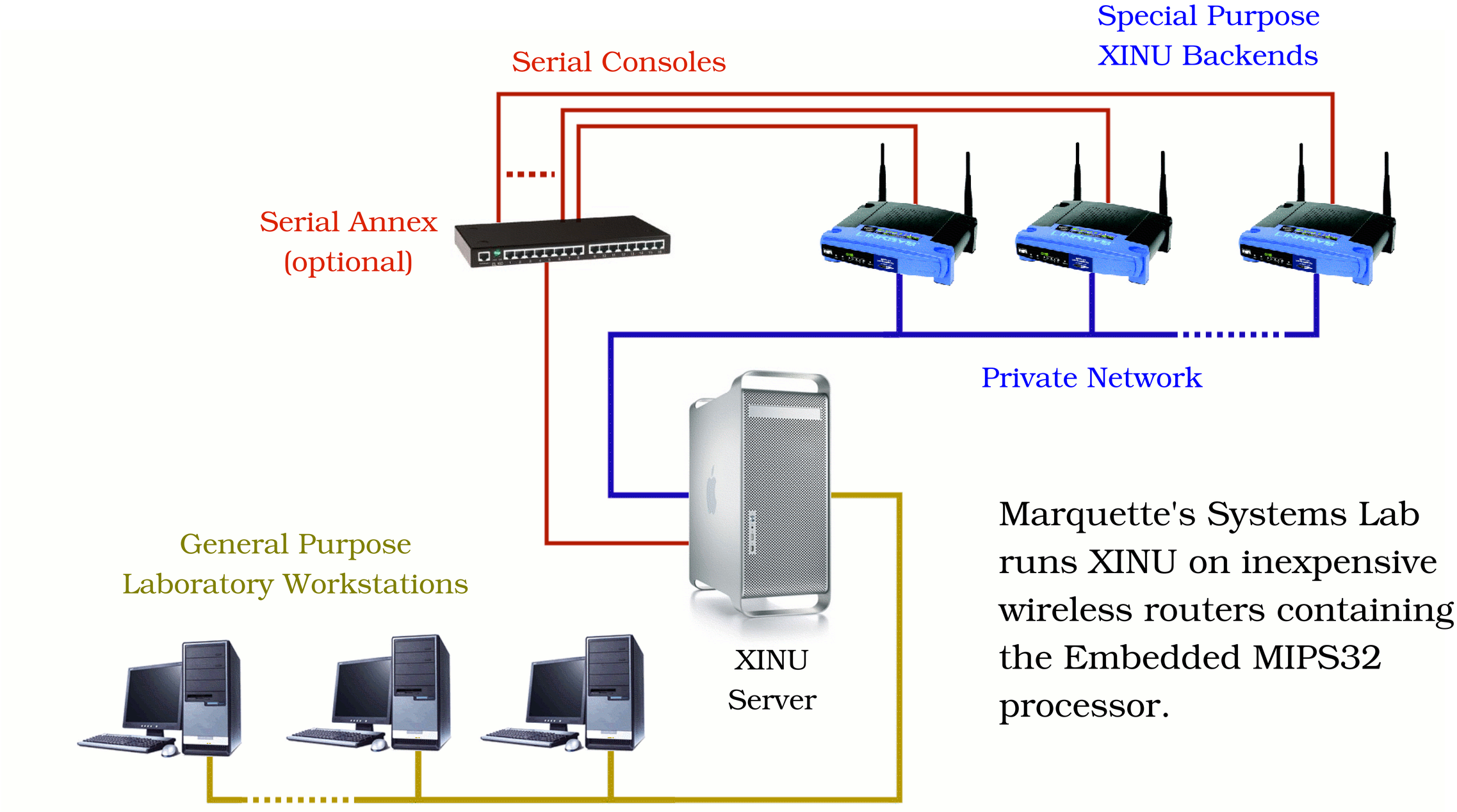
Our Experimental Operating Systems Laboratory serves as a target platform in undergraduate courses on Hardware Systems (COSC 065), Operating Systems (COSC 125 and COEN 183), Embedded Operating Systems (COSC 195) and Compiler Construction (COSC 170).

Coming Soon: Additional modules for coursework in Embedded and Realtime Systems, Networking protocol stacks, Wireless Networking, and Internetworking.

http://www.mscs.mu.edu/~brylow/xinu/

XINU in the 21st Century

Purdue University's XINU Operating System has been successfully deployed in classrooms, research labs, and commercial products for more than 20 years. Marquette University presents a new reimplementation of XINU, targeted to modern RISC architectures, written in ANSI-compliant C, and particularly well-suited to embedded platforms with scarce resources.



Marquette's Systems Lab runs XINU on inexpensive wireless routers containing the Embedded MIPS32 processor.

XINU Backends

Backend targets upload student kernel over private network on boot, run O/S directly.

No simulations or emulation; real hardware.

MIPS targets: We use Linksys WRT54GL wireless routers (~$60) with serial port modifications (~$10) running an embedded MIPS32 200MHz processor, 4 MB flash, 16 MB RAM, two UARTs, wired and wireless network interfaces.

PowerPC targets: We use Apple G3 desktops (recycled) with 512 MB RAM, linear framebuffer, PCI bus, NIC, HD. Apple G4 MiniMac also supported.

CISC targets: Classic XINU runs on Intel x86, Sun 3/Motorola 68K, Sparc, and VAX, among others.

XINU Server

General purpose server with multiple network interfaces manages a private network for the XINU backends, using standard network protocols like DHCP and TFTP.

Backend serial consoles can connect directly to server's serial ports, or, in larger installations, to a serial annex or concentrator that allows many more serial ports.

Daemon allows users on frontend workstations to remotely access backend serial consoles, or upload fresh kernels. Optional rebooting hardware allows clients to remotely reset crashed backends.

Server tools freely available for modern UNIX platforms, including Fedora Linux and Solaris.

XINU Frontends

General purpose computer laboratory workstations can compile the XINU kernel, using the standard GNU C compiler and UNIX toolchain. GCC cross-compilers are readily available when the frontend architecture does not match backend architecture.

Backend consoles can be connected directly to frontend serial ports, or frontends can communicate with server daemon that manages collections of backend serial consoles.

With fully remote console access, kernel upload and powercycling, any machine on the network is a potential frontend, and need not be physically near the XINU server and laboratory hardware. Students can work on their operating system projects from their dorm room computers.

Typical COSC 125 Operating Systems Sequence:

Weeks	1-3	Introduction to C language and development environment. (We use GNU toolchain on Linux.)
Week	4	Output device driver (UART or linear framebuffer.)
Week	5	Process Control (Process Control Blocks, Context Switch in appropriate assembly language.)
Week	7	Priority Scheduling, Process Termination.
Week	8	Preemption (Interrupts) and Synchronization (Counting Semaphores with Wait Queues.)
Week	9	Sleep/Wakeup (Delta Queues.)
Week	10	Memory Management (Dynamic Memory Allocation, Deallocation, Free Lists, Free Space Compaction.)
Week	12	Asynchronous Device Driver (UART or Network.)
Week	14	Advanced Topic Basic Command-Line Shell, or Window System Pseudo-Device Management, or Simple Network File System.

Advanced Assignments for later courses:

Network Packet Analyzer, Network Router, Virtual Memory with Network Backing Store, Networks of Sensors, Realtime Scheduling, IP Telephony.

Available now:

MIPS XINU with preemptive multitasking, priority scheduling, synchronization primitives, dynamic memory management, serial port and system clock drivers; Daemon and client for remote serial console connection, kernel upload, and powercycling; schematics and/or parts lists for WRT54GL serial port modifications, optional serial port annex, and powercycling hardware.

Ongoing Work:

Wired and Wireless network interface drivers for the WRT54GL; Integrated TCP/IP stack; Remote single-step debugging facilities from front-end console; PowerPC G3 serial console driver and automatic powercycling. Textbook on Embedded Systems Programming in progress.

Future Work:

Support for compilation and console connection from non-UNIX frontend machines; Support for exotic hardware peripherals, such as sensors, EJTAG hardware, or IP telephony.

This work partially supported by the Wehr Foundation and by Cisco Systems.