Mach-US:
UNIX On Generic OS Object Servers

J. Mark Stevenson

Daniel P. Julin

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

jms@cs.cmu.edu
dpj@isis.stratus.com

Mach-US: The CMU Mach Multi-server OS

Mach-US is an object-oriented multi-server OS

Implemented in C++ upon the Mach3.0 microkernel

Symmetric, has no central OS server

Runs unmodified UNIX BSD4.3 binaries

Mach-US: Research Rationale

Multiple system servers on a microkernel

Application Programmer Interface (API) neutral
OS services

Remote Method Invocation (RMI) for
OS services/objects

Client side OS computation

UNIX API re-implementation

Outline

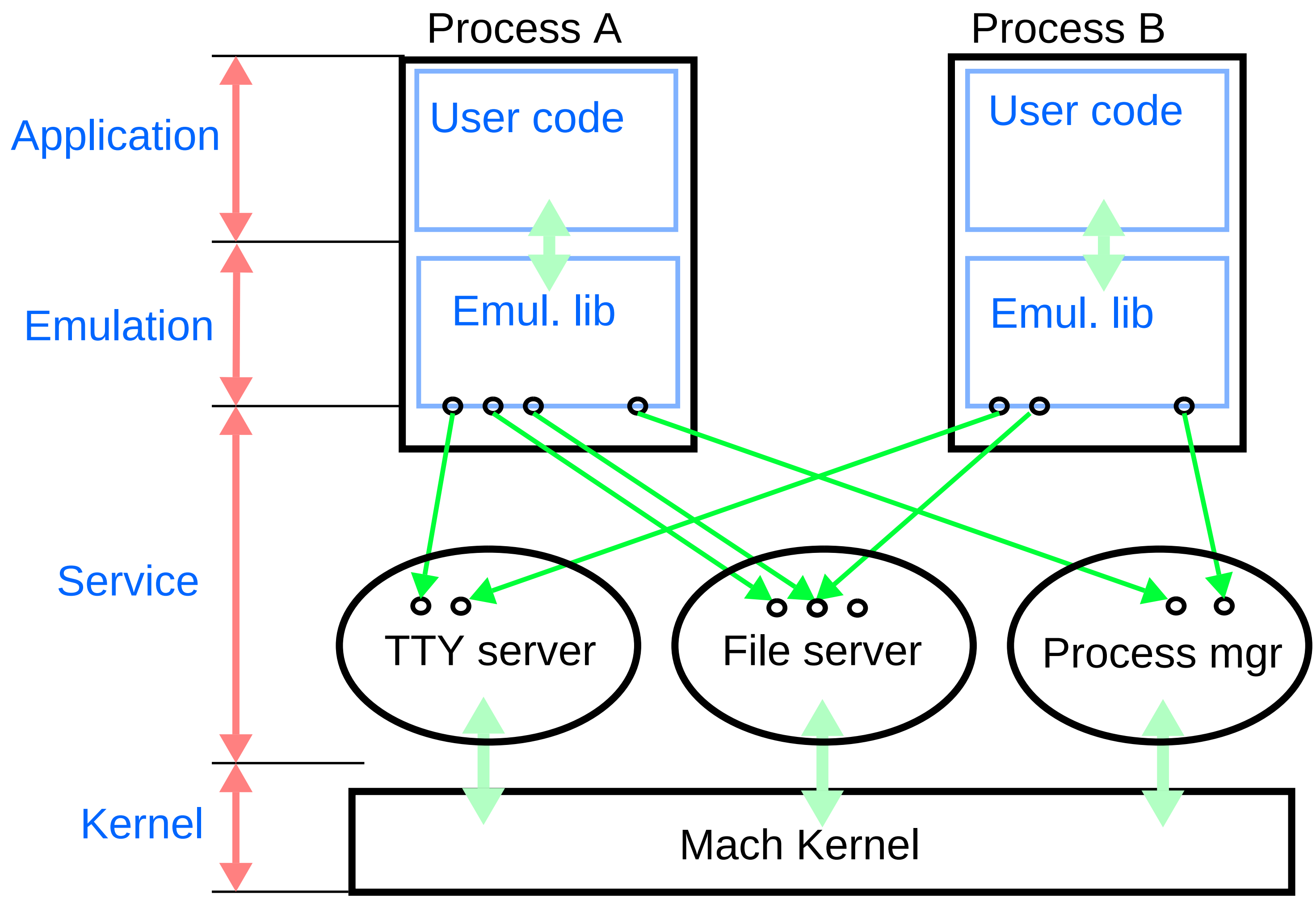
Mach-US Introduction

Status and Performance

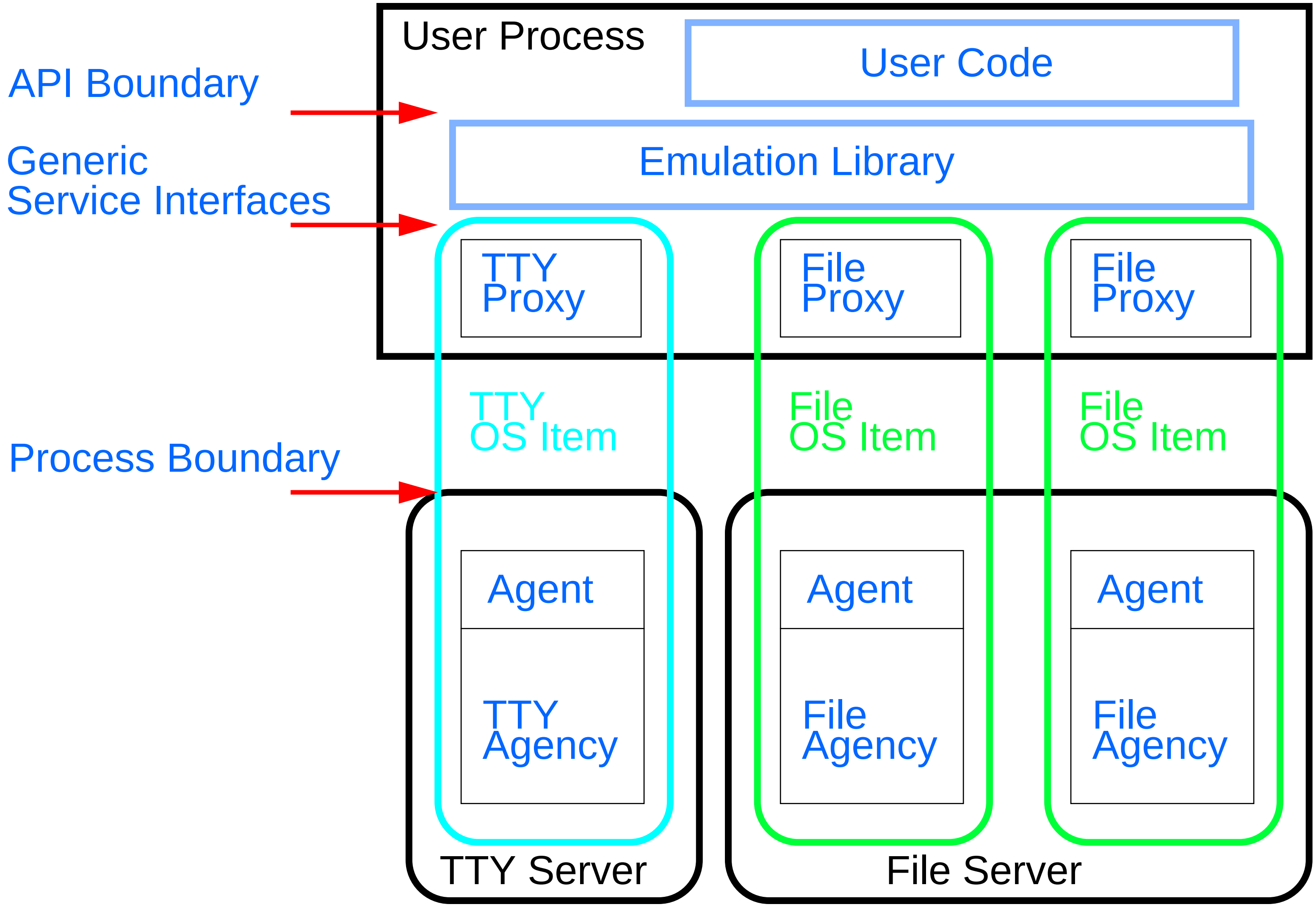
OS Feature Analysis

Summary

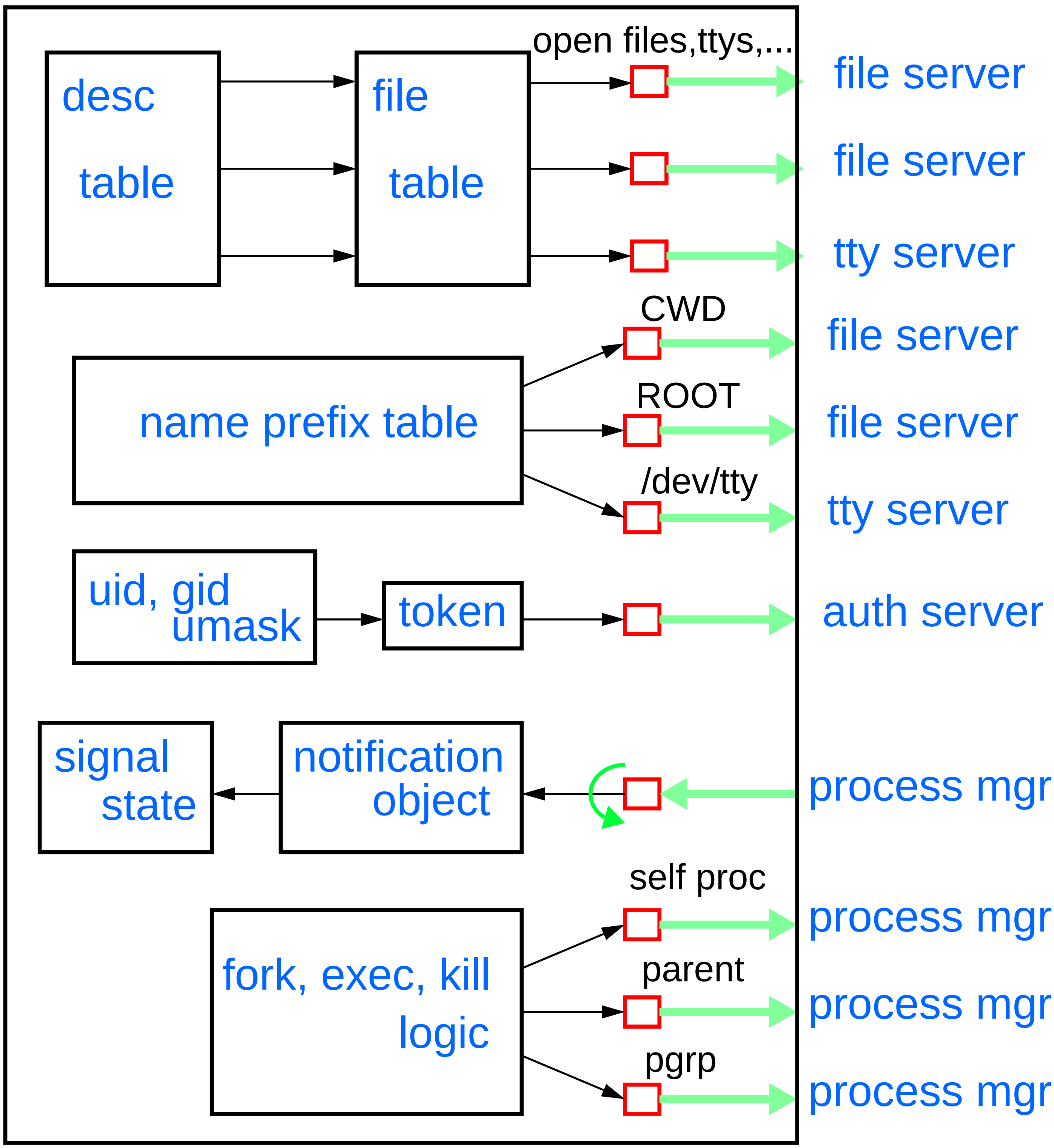
Mach-US: Architecture



OS Items & System Boundaries



Mach-US: UNIX Emulation Library



Service Interface: Description

Service Interface Properties:

- Defined in C++ as "abstract classes"
- N interfaces per item, N items per interface
- Each is a communication paradigm

Service Interfaces Supplied:

- **Naming** access directories of OS-items
- **Access Control** limit OS-item usage
- **I/O** accesses byte and record data flow
- **Net** OSI-XTI transport-layer semantics
- **Async Notification** events into client
- **Process Management** for process info

Service Interface: Design

Guidelines:

- Separate semantics from API
- User data maintained in emulation library
- EmulLib supplies inter-server coordination
- Client side computation of OS work
- Enable client side caching
- Fast and simple

Requirements:

- No system-crashing data on client side
- Only read-permitted data on client side
- Must support UNIX API

Client-to-Server Model

Proxy features:

- Data transfer to proxy is server defined
- Invoke proxy for local or remote code
- Created on "out" OS-item return and fork

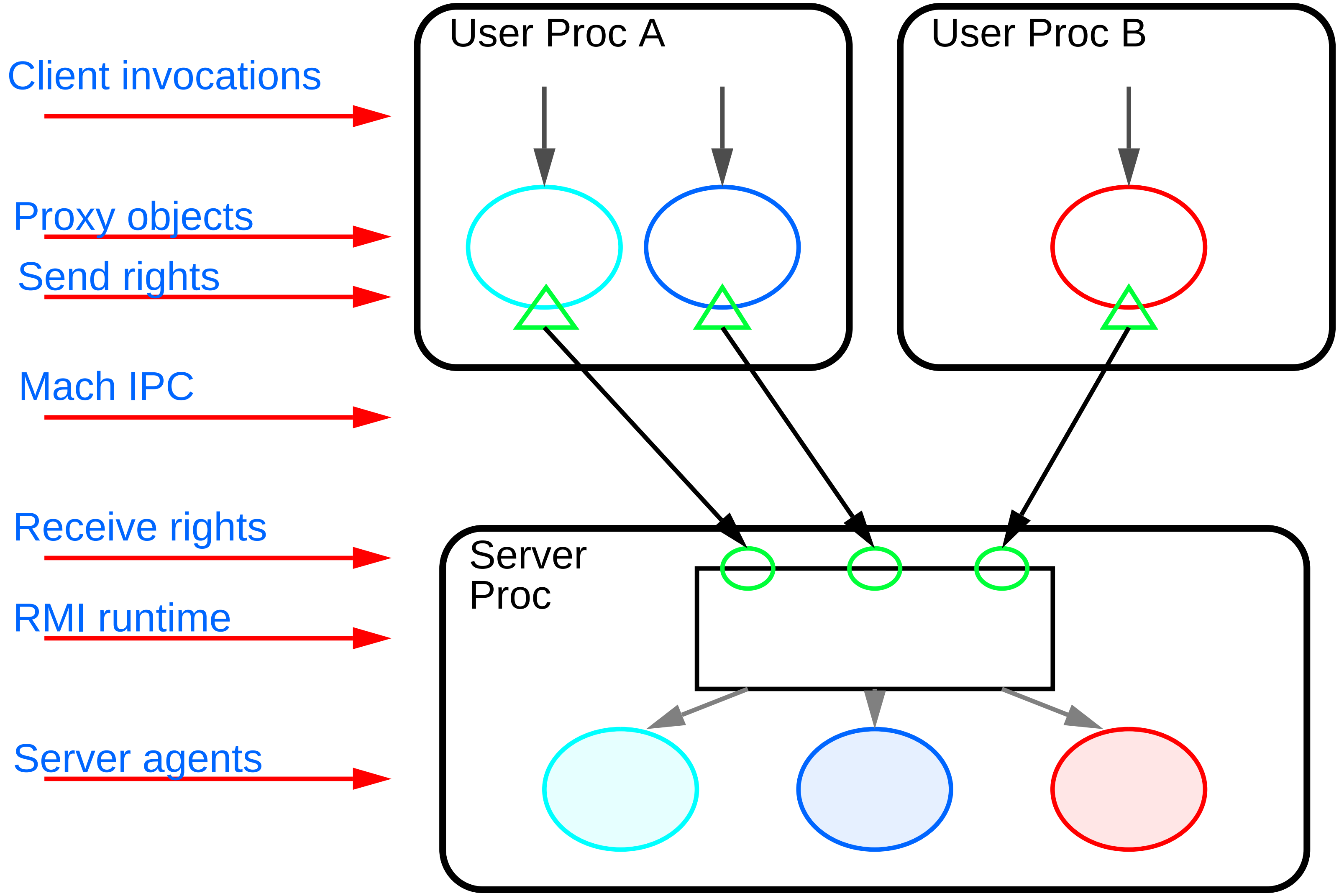
Remote method invocation (RMI) mechanism:

- Invocations securely passed to Agent
- Methods can be **atomically** interrupted
- IPC no-more-senders for robust OS-item closes

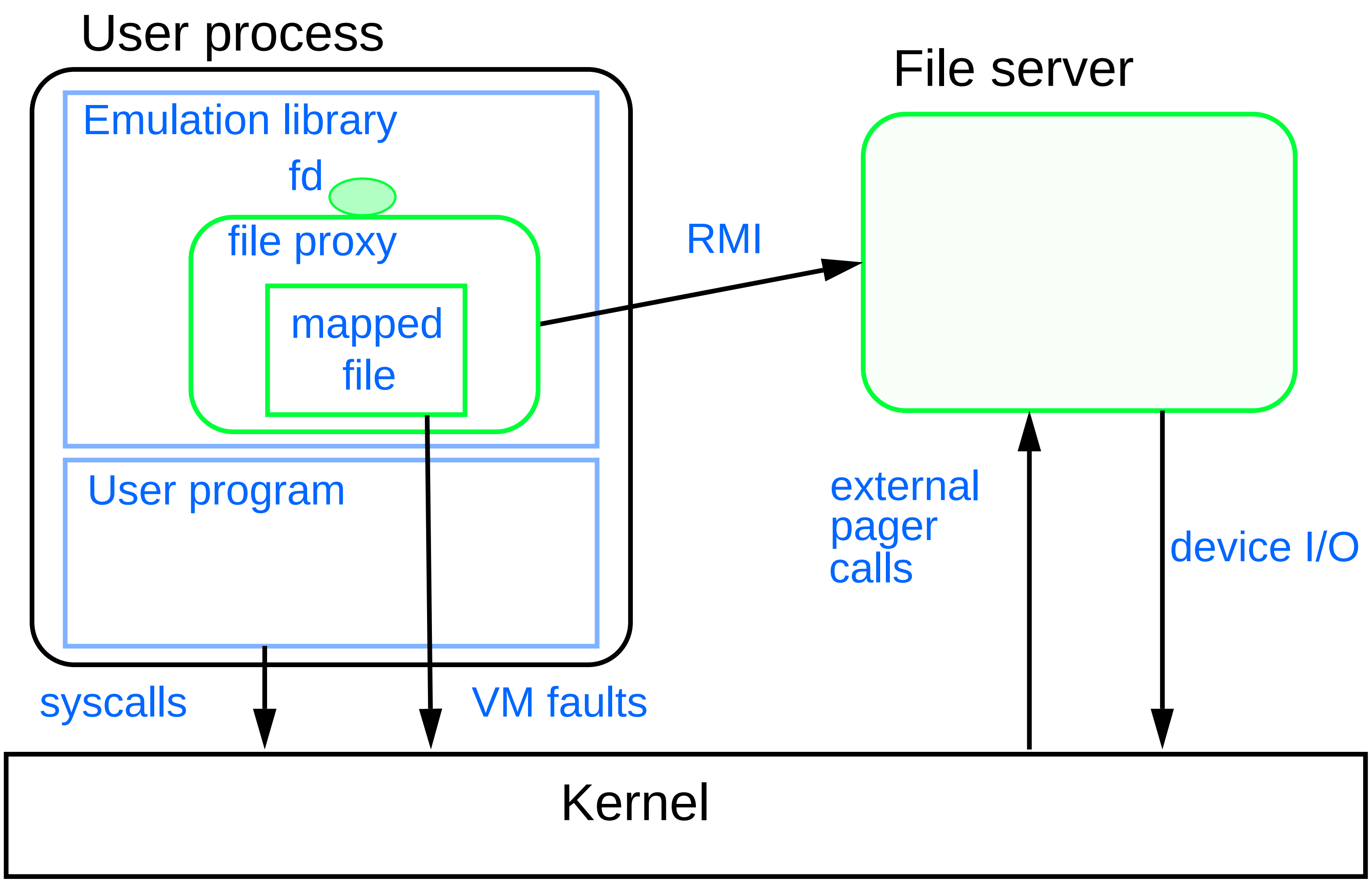
Agent features:

- Access mediated using credentials, access control lists, and access mode
- Created on "out" OS-item return (open)

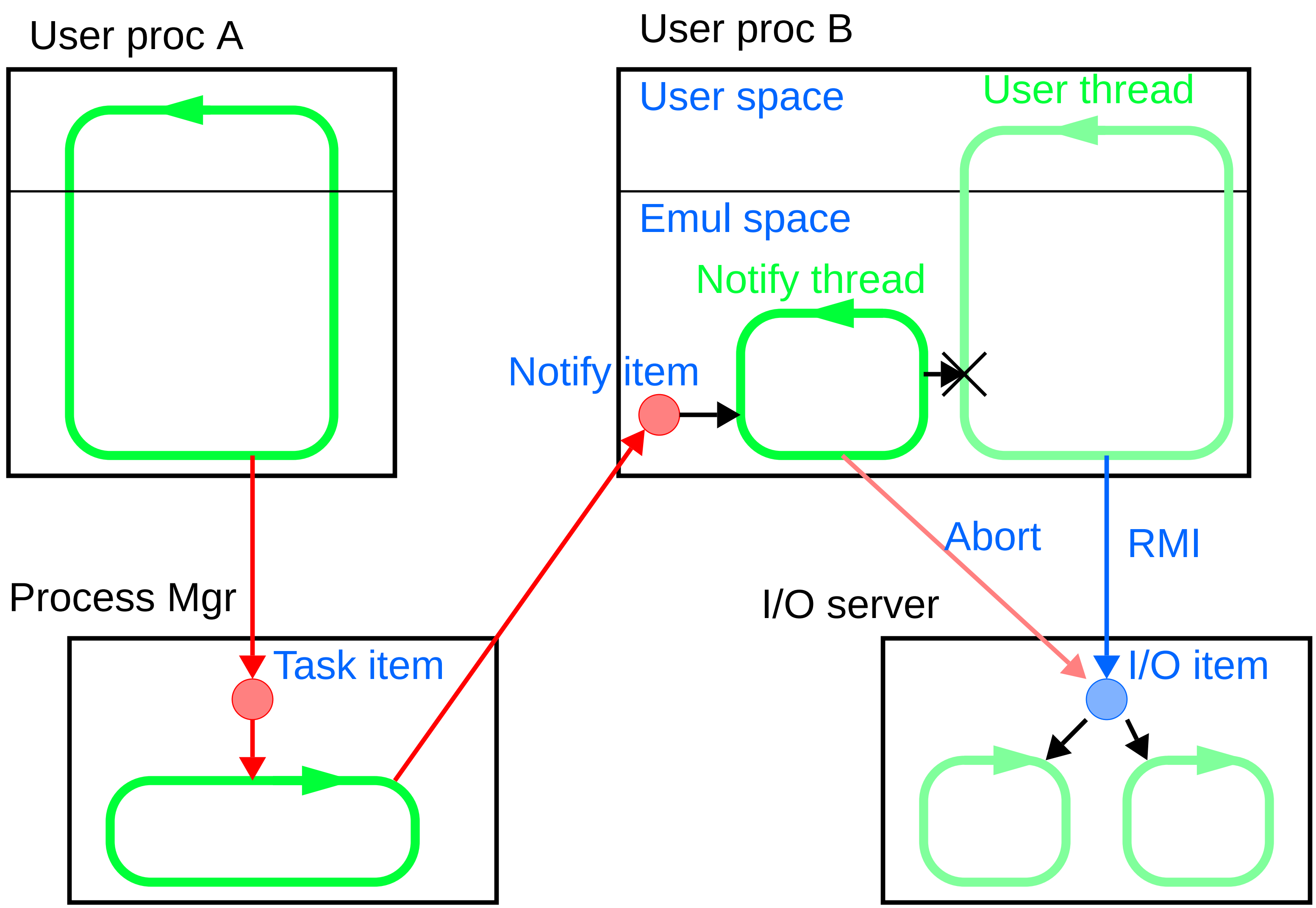
Remote Method Invocation System



Example: simple UNIX file operations



Signal Delivery



Developer Support Library

Interface support examples:

- Record/Byte streams,
- Directories, links
- Temporary OS items

General support examples:

- Vnodes
- Pagers, mapped files, shared memory
- Sync/Async interrupt handling

Support results:

- Faster prototyping
- Easier integration of foreign code

Outline

Mach-US Introduction

Status and Performance

OS Feature Analysis

Summary

Status: Functionality

Runs UNIX binaries:

- Daily UNIX utilities: ls, emacs, find, ...
- Compilation: gcc, make, cpp, ...
- Net Access: telnet(d), ftp(d), inetd

Development environment for Mach-US test software

Self building

Status: Reliability

Positive:

- Users say: "It feels like a regular UNIX"
- No fear of the dark: stays up indefinitely
- Handles heavy compilation stress

Negative:

- Uncommon untried code paths still exist
- Small differences from BSD: error values, esoteric semantics

Mach-US: Performance

Test Name	Mach-UX	Mach-US	Slowdown
Compile	7.3 sec	8.9 sec	22%
Null FTP Get/Put	260 KB/sec	240 KB/sec	7.7%
Parallel Compile(1) Parallel Compile(10)	39 sec 146 sec	49 sec 139 sec	26% -4.8%

Outline

Mach-US Introduction

Status and Performance

OS Feature Analysis

Summary

Analysis: Multiple Servers

Configuration flexibility enhanced

Strong subsystem separation:

- Just one inter-server call needed
  (keyboard signals)
- Failure isolation ensured

Simplified development/test cycle

Easier authoring and replacement of OS functionality

Analysis: API-Neutral Services

API neutrality

No final proof, only one emulation library today

OS interface redesign: very powerful

- Common OS-item interface semantics
- Separation of user data from secure data
- Service call reduction:
  No service calls for: dup, lseek, sigsetmask,...

Analysis: Remote Method Invocation

Object oriented technology for OS implementation

Helpful, as with any large project

Proxies: enabled caching and communication flexibility

E.g., cached names, shared memory

Remote method invocation: valuable for complexity hiding

- Call site transparency
- Access control
- Abort/Event support

Analysis: Intelligent API Emulation Libs

Intelligent OS client libraries:

- Client side computation wins for multiprocessor systems
- Compile test invoked 9290 syscalls which needed only 2430 server calls
- Flexibility enables OS interface research

API re-implementation:

- Don’t Do It:
  - API conformance consumes developer time
- Do It:
  - Needed to run existing applications
  - Syscall Interface vs. Service Interface separation is valuable

Summary: Mach-US Description

Mach-US Properties:

- Object-Oriented multi-server OS
- Uses Mach3.0 microkernel
- Runs unmodified UNIX BSD4.3 binaries
- Symmetric, has no central OS server

Mach-US Features:

- Client-Server communication flexibility
- Lowered user to OS communication need
- User space OS processing
- Active method interruption
- Extensive developer support library

Summary: Mach-US Demonstrates

Multiple system servers on a microkernel is viable

- Easier authoring, debugging, and replacement

Decoupling API, OS-service, and process boundaries gives important flexibility for research in OS interface design and user-to-OS communication mechanisms

OS interface redesign can drastically cut OS calls

Intelligent client-side OS libraries can do significant operating system work

API re-implementation is painful but necessary

Object technology works for OS development

Mach-US: Availability

Mach-US is available today

WEB: http://www.cs.cmu.edu:8001/afs/cs/project/
/mach/public/www/projects/mach_us.html

FTP: Host: mach.cs.cmu.edu,
subdirs: doc/mach_us, src/mach_us

SUP:

- Current CMU-Mach Licensed Projects:
  jms@cs.cmu.edu
- Other UNIX Licensed Projects:
  lli@cs.cmu.edu

EMAIL: jms@cs.cmu.edu