Where Next -Solaris

- **** Integrate our work with TAO ****

- **Continue testing to characterize the RT performance of Solaris**

- **Complete work on a prototype rate monotonic scheduling class**

- **Add connection based output queuing in the ATM driver**

- **Extend this work to support multiprocessor environments**

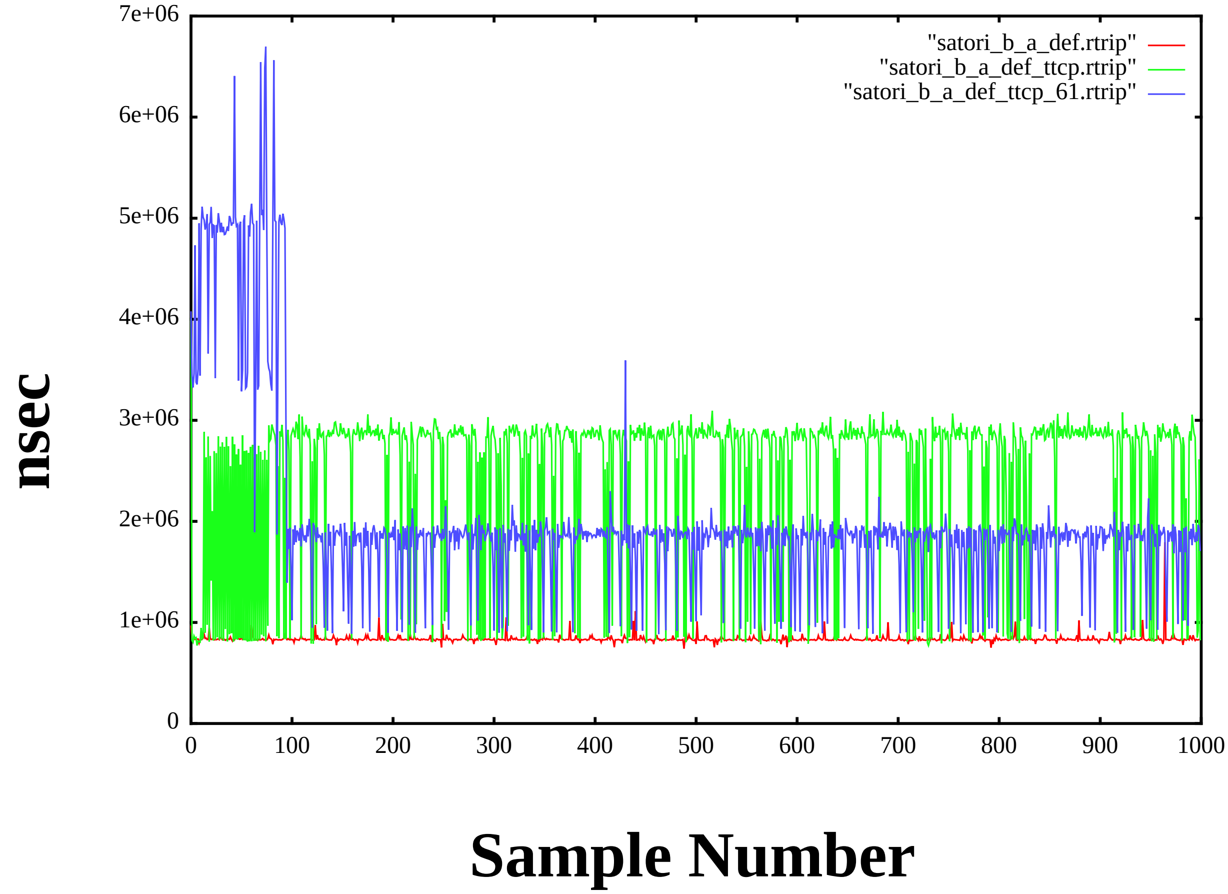
- **Evaluate Solaris I/O subsystem enhancements, quantify performance and consider extension.**

- **Evaluate alternative frameworks and operating systems**

Where Next -NetBSD

**** Integrate our work with TAO ****

Solaris - Remote IPC



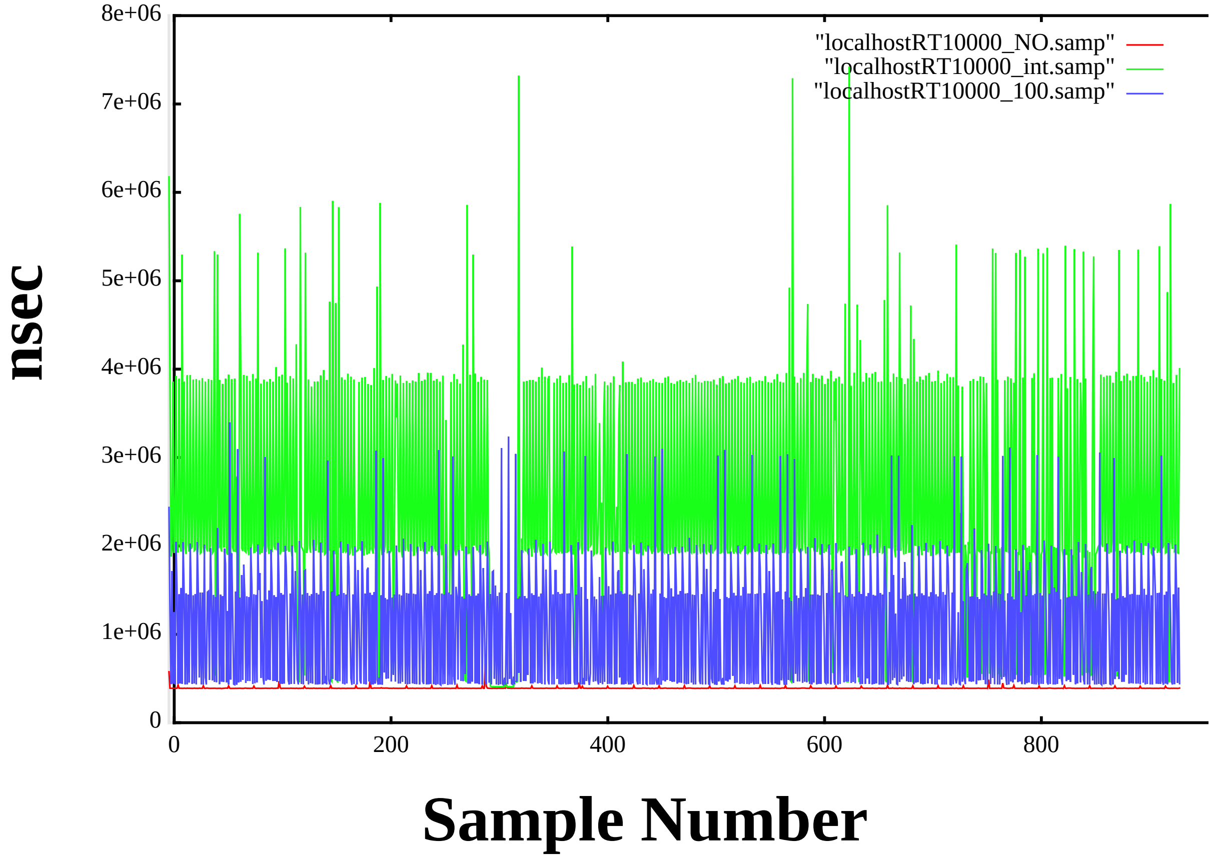
Round trip latency between a 300MHz Ultra2 and a 170MHz SPARC5, processes are in the TS scheduling class.

a) no competing network traffic

b) competing traffic processed with interrupt priority

c) competing traffic processed with SYS priority of 61

Solaris - Local IPC



Local IPC Round trip latency for two processes on a 170 MHz SPARC5 with real-time priorities of 125

a) no competing network traffic

b) competing traffic processed with interrupt priority

c) competing traffic processed with RT priority of 100

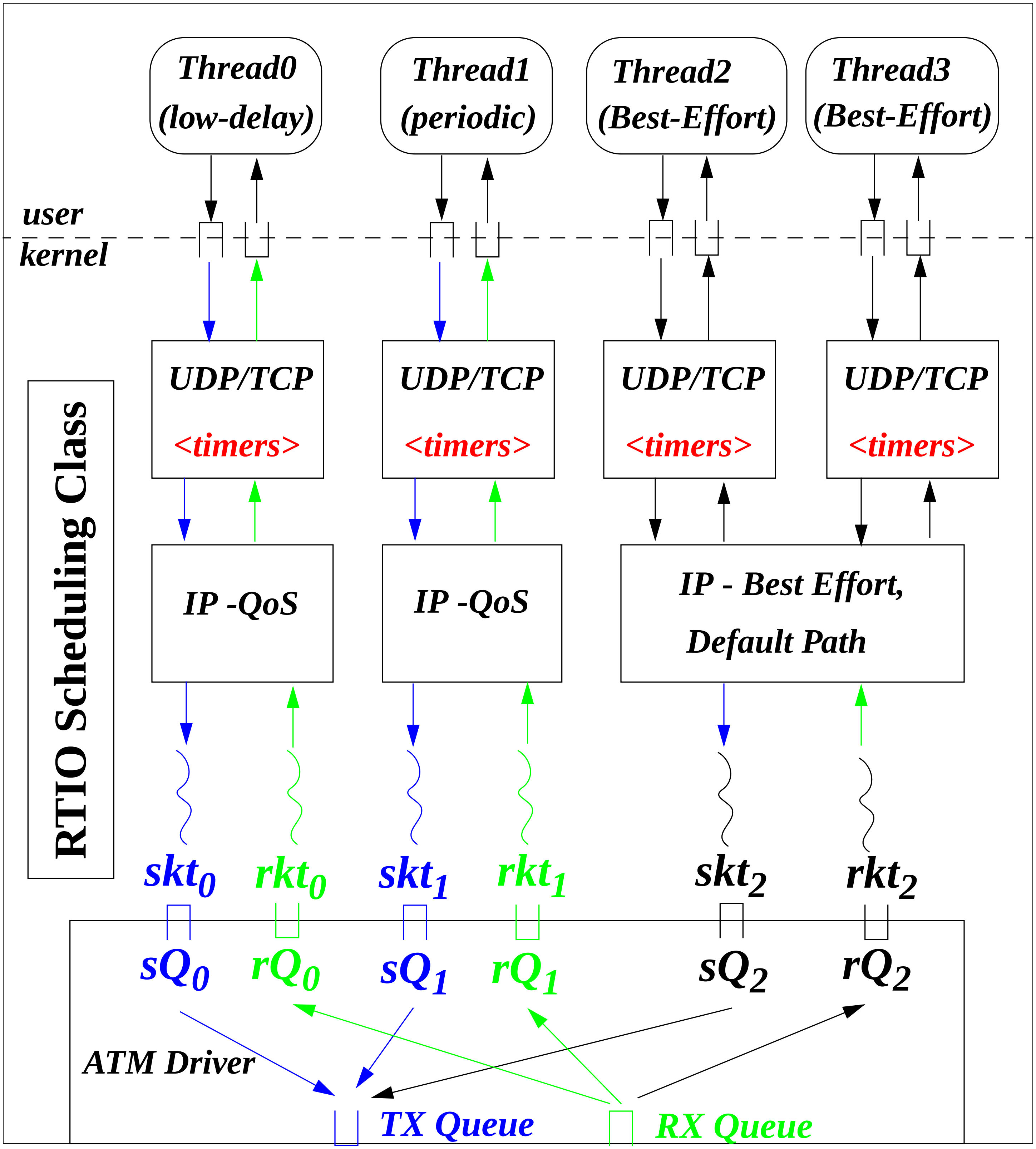
Status of Solaris Work

Rate Monotonic scheduling class scheduled to be completed by June 1, 1998. This includes admission control.

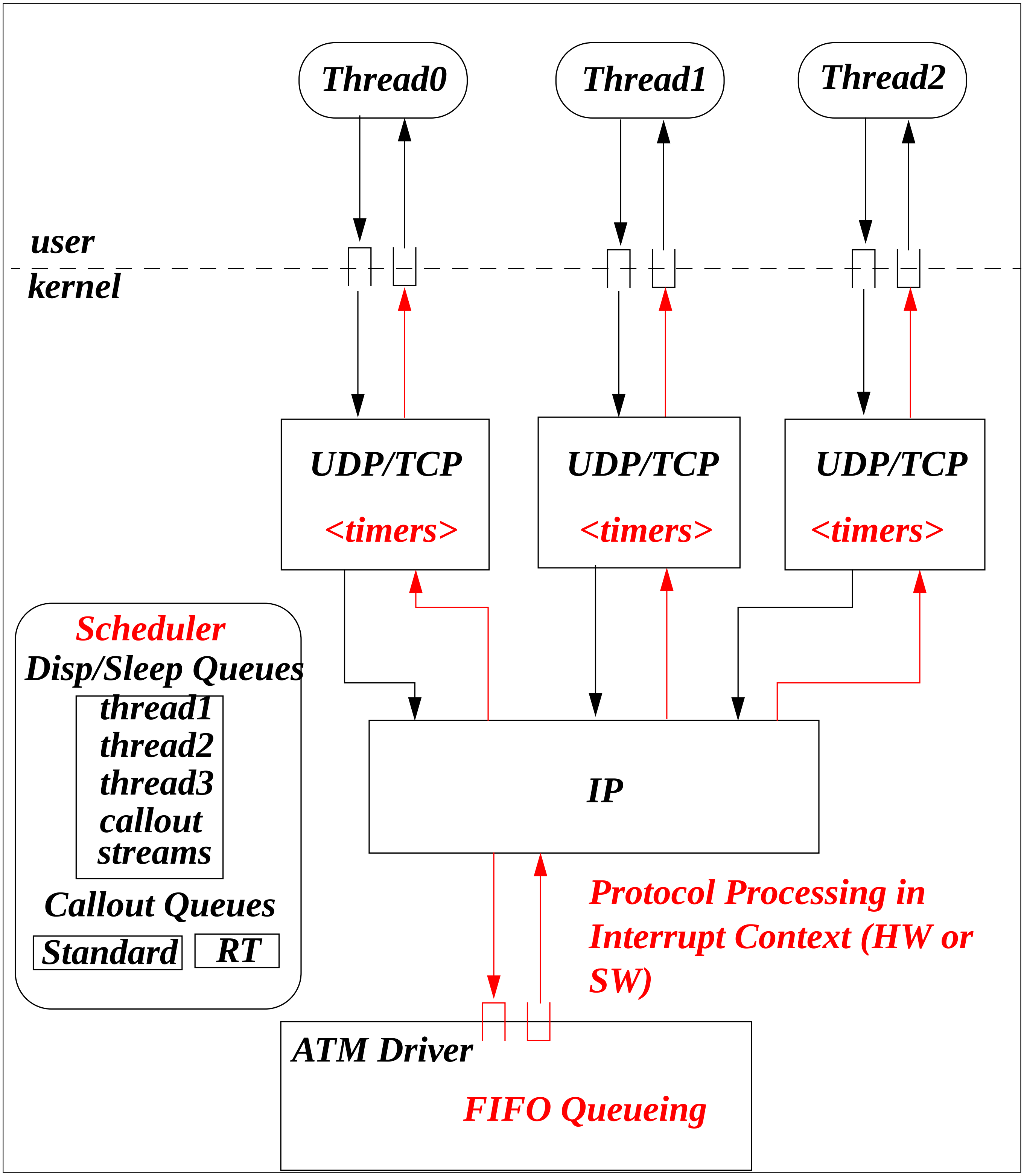
STREAMS subsystem modifications

- multiple kernel threads for protocol processing - prototype complete.
- demultiplex connections in ATM driver using VCs - prototype complete
- dedicated STREAMS - prototype complete
- periodic protocol processing using the real-time callout queue
- connection based buffering on the send side.

The Solution



The Problem - Network IO



Sources of Non-Deterministic Behavior in Solaris

Priority inversion:

- Protocol processing with Interrupt priorities
- FIFO queuing
- Hidden scheduling
- Resource locks

Lack of Periodic Real-Time CPU scheduling

Lack of QoS specification interface, admission control and enforcement

What About Solaris

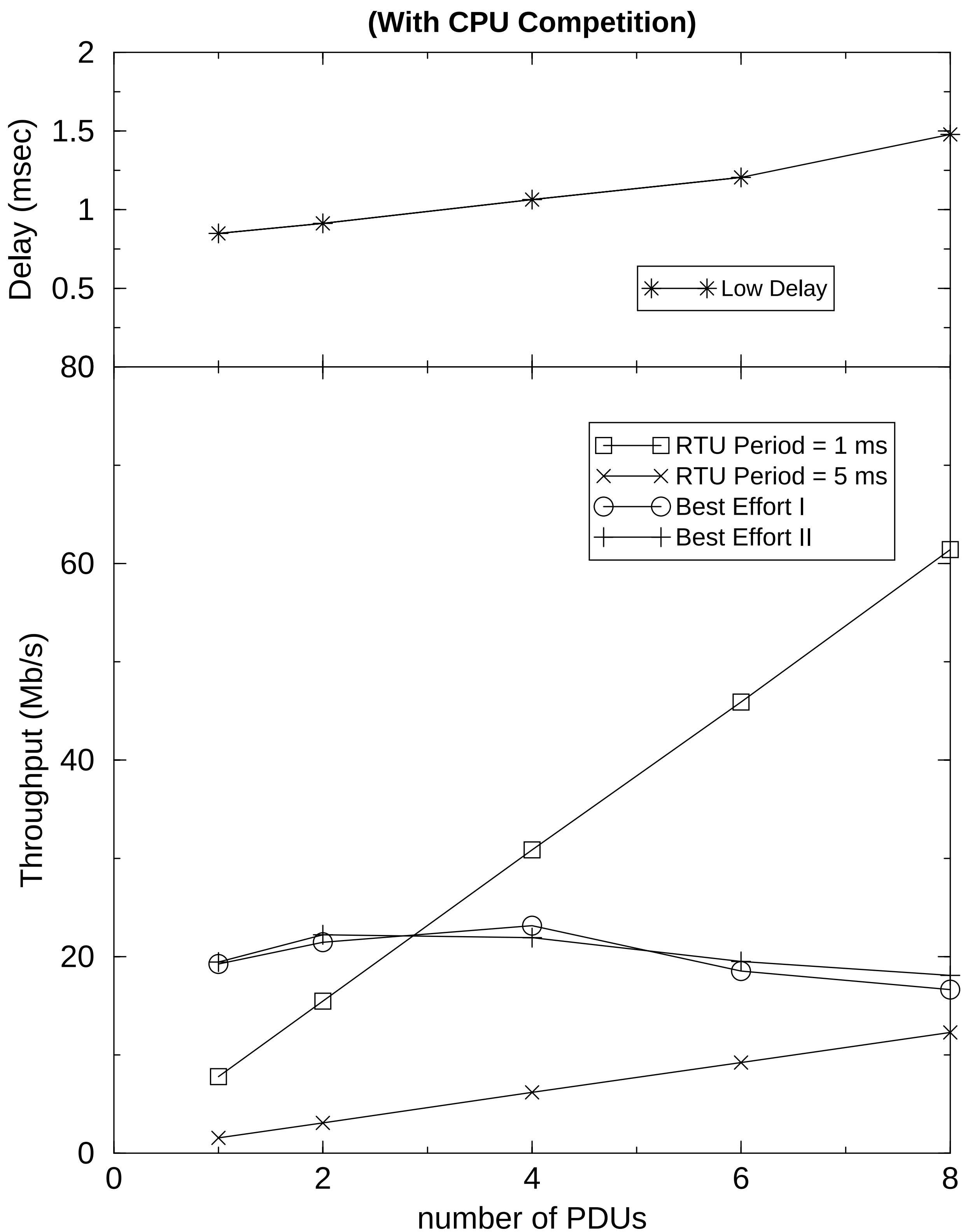
Our initial goal was to port much of our work and knowledge from NetBSD to Solaris

Solaris has proven to be much more difficult than expected:

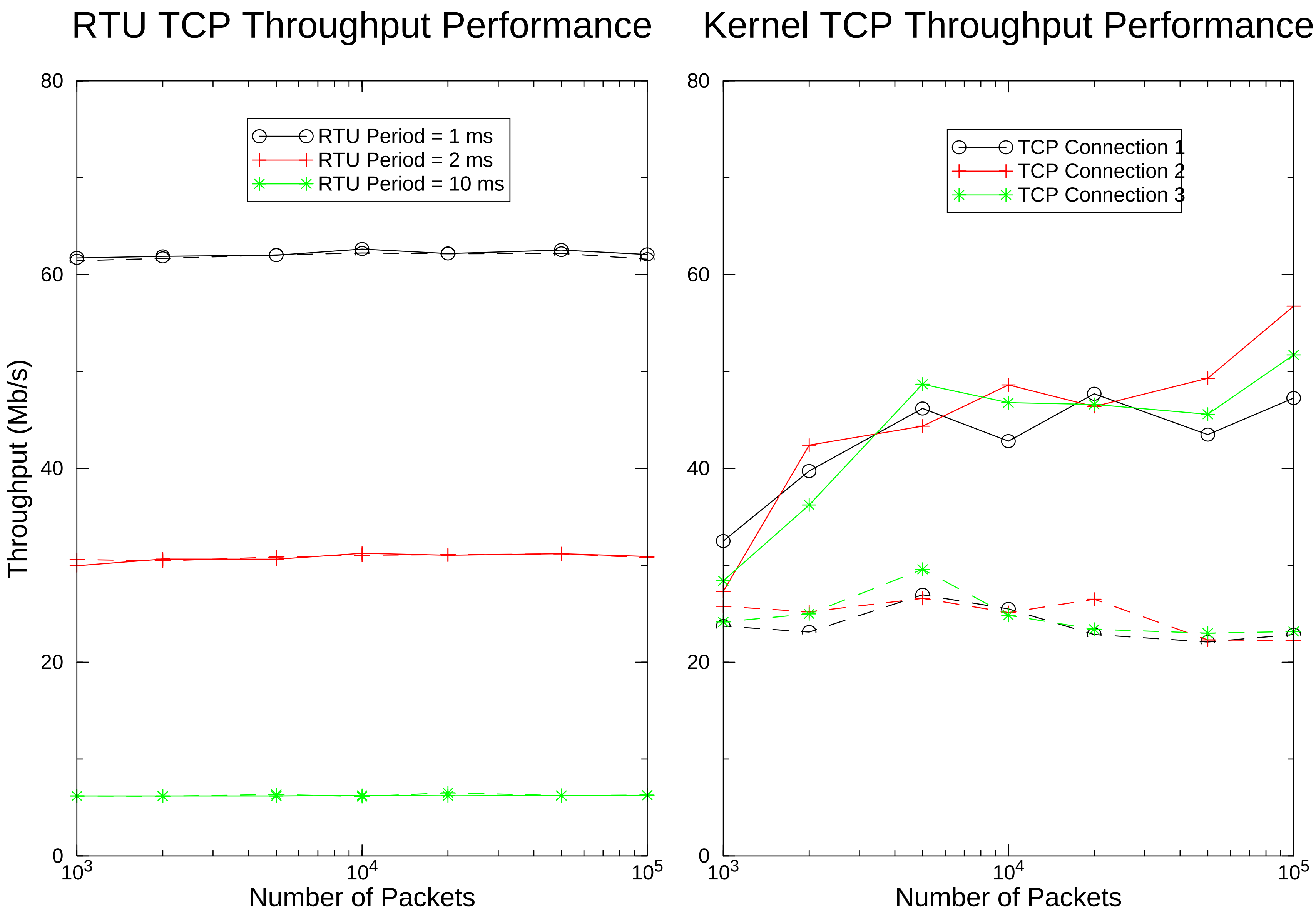
- Complexity - IP alone is 18,000 plus lines of code
- Synchronization objects and system threads complicate scheduling class implementation
- STREAMS implementation has been substantially modified from the SVR4 version
- Existence of a fixed priority Real-Time scheduling class and Real-Time callout queue ameliorate some of the problems encountered in NetBSD.

NetBSD with QoS

Low Delay, Guaranteed Bandwidth and Best Effort



NetBSD Experimental Results



Changes to NetBSD

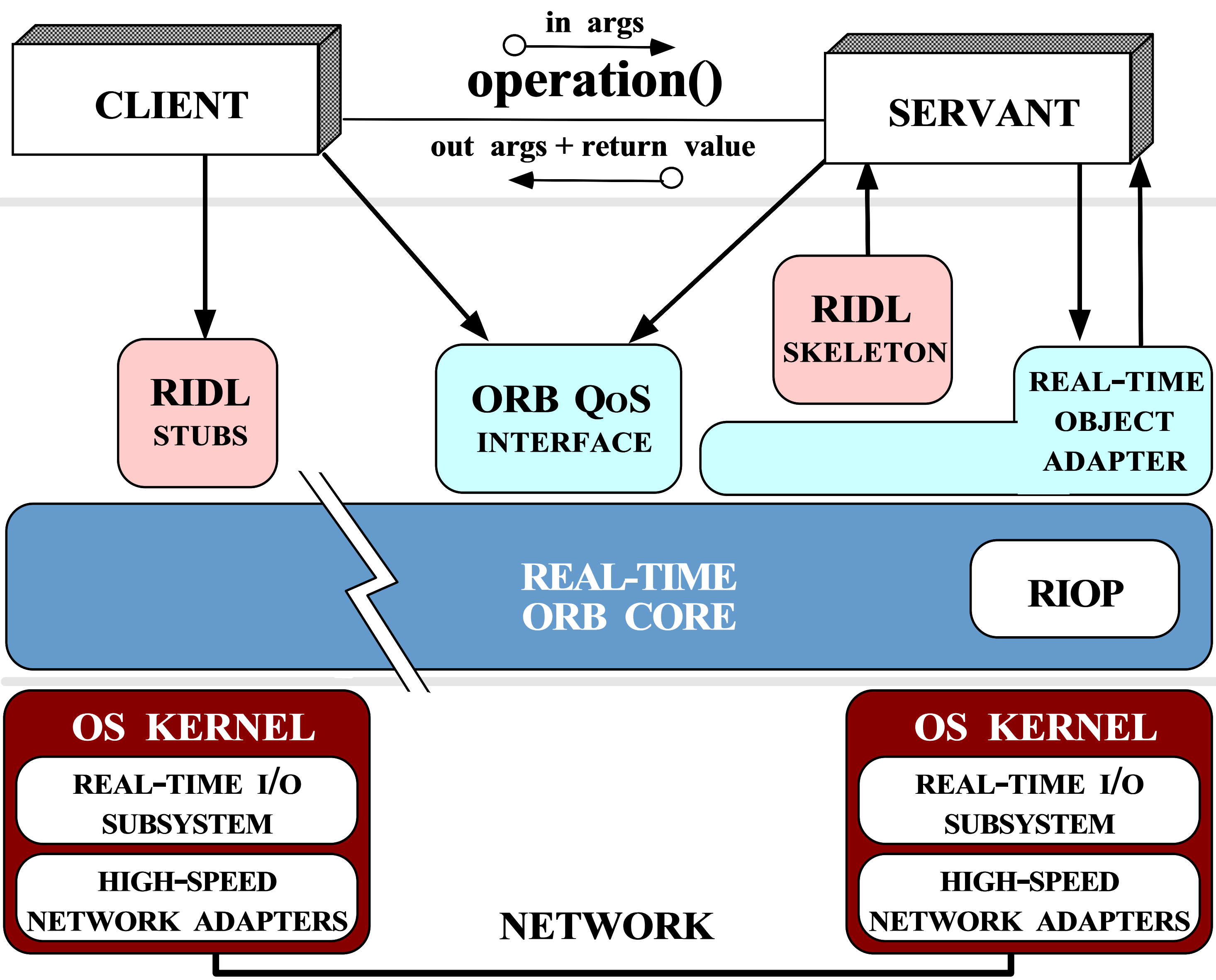
- **Real-Time Upcalls (RTU) -** Rate Monotonic, delayed preemption scheduling class. See http://www.arl.wustl.edu/arl/projects/ito/ton.ps
- **Universal Continuous Media I/O (UCMIO) -** zero-copy buffer semantics. See http://www.ccrc.wustl.edu/pub/chuck/stmt.html
- **Shared file IO and Network IO buffers -** Shared Network and Filesystem buffers. See http://www.arl.wustl.edu/arl/refpapers/milind/mars.html
- **QoS specification API and admission control**

I/O Subsystem Goals

Design, specification and prototype implementation of a high performance I/O subsystem that can provide Gbps bandwidth, bounded latency and QoS guarantees to applications.

- Support for multiple operating systems: Currently NetBSD and Solaris.
- Periodic and aperiodic data I/O
- Vertically integrated protocol stack
- Specification driven processing of I/O events
- Real-Time OS Scheduling and admission control

The Big Picture



Overview

- **Motivation**
- **I/O Subsystem Goals**
- **Modifications to NetBSD**
- **Modifications to Solaris**
- **Future work**

High Performance I/O with QoS

Fred Kuhns
Guru Parulkar
Rajeev Bector
Peter Memishian
Yunxi Shi
Dakang Wu

Applied Research Laboratory
Washington University
{fredk,guru,rajeev,meem,sher-lia,dw1}@arl.wustl.edu

****Sponsored by SPRINT and DARPA***