

CS 252 Project Proposal

Spring 2005

Greg Gibeling
Andrew Schultz

SMT Low Overhead Interrupt Handling and Zero Copy Networking with Click

It is widely known that improvements in network bandwidth have been increasing even faster than microprocessor performance governed by Moore's Law. Given this trend, there is an increasing gap between the raw bandwidth available from the network and the ability for the processor to keep up with a highly utilized network. The traditional way of handling this gap has been to use outboard processing elements like TCP offload engines (TOEs) to reduce the amount of time spent by the processor in handling packet. However, not only does this hardware require substantial support from the operating system network stack, but it also does not scale well as raw packet I/O increases. Recent papers have proposed a tighter coupling of the I/O system and the processor, placing the NIC directly on the die with direct access to the cache.

Even with tighter integration between the NIC and the processor, packet processing still has a fair amount of overhead. This overhead comes primarily from two sources: context switching to handle device interrupts, and copying data from kernel to user space buffers. With extremely high bandwidth networking devices, the cost of taking interrupts to handle incoming and outgoing data becomes prohibitive, requiring techniques such as interrupt coalescing to prevent the interrupt rate from hurting system performance. This problem is particularly acute when low latency packet processing is required. For example, the Click modular router goes as far as ripping out the interrupt handling code and converts packet handling into a polling operation.

To address this problem, we propose using dedicated SMT threads for interrupt handling. The key idea is to dedicate both a thread, and some resources in the cache, to allow very low overhead interrupt handling. Additionally, we would like to explore using SMT threads and a more tightly coupled NIC to implement zero copy packet processing, in a similar vein as U-Net. Finally, we would like to determine the benefit of partitioning the cache to allow a thread to "pin" some space for its state, allowing a processor to specify particular state which it does not want evicted. We plan to evaluate the performance of our packet processing by comparing the performance of Click with the polling optimizations versus traditional interrupt handling with our enhancements.