



# Low Overhead Interrupt Handling with SMT

---

CS 252

Greg Gibeling  
Andrew Schultz



# Problem

---

- Traditional OS interrupt handling adds latency (context switch + copies)
- High bandwidth devices overwhelm traditional I/O architecture
  - Ethernet speed increasing faster than processor (10 Gb/s Ethernet becoming common)
  - Require interrupt coalescing, TOEs and other techniques to handle



## Problem (cont)

---

- Motivation: Click
  - Click removes traditional interrupt handling code in favor of direct polling on DMA descriptors
- Can we do better?
  - Get low latency, low overhead demanded by Click with traditional interrupt paradigm



# Solution

---

- Re-evaluate I/O architecture
- Allow OS more control over resources it needs for low latency
- Use “hot” SMT thread to reduce context switch overhead
- Zero kernel/user copy of data



# How

---

- Dedicate SMT thread to device handling
- Cache pinning to give more control over what stays in cache
- Tighter integration with hardware to support user level networking (a la U-Net)
  - Allow device handling thread to initiate DMA directly to user buffers
- Evaluate with M5 or ML-RSIM (maybe compare Click under polling and new I/O)



# Cool Things

---

- Able to support high speed I/O applications like Click with commodity HW/SW (w/o rewriting int. handling)
- Primarily aimed at I/O but has implications for improving all exception handling
- Architectural support for application control of data unit size



# What's Related

---

- Hardware
  - EMP – True zero copy on NIC
  - SMT exception handling – Done for TLB misses
  - On-chip NICs – Tighter coupling between NIC and CPU
  - TOEs and accelerators – Shift processing to special devices
- Software (OS)
  - U-Net – ADU + fewer copies, more application control
  - Click – Change method from interrupt to polling
  - Interrupt coalescing – Add latency, reduce thrashing