RTU - Threads for Protocol Processing

Weihai Yu
Oct. 23, 1998
WUSTL
weihai@arl.wustl.edu
**BSD protocol processing priorities**

A packet is handled by procs of different priorities:
1. Hardware interrupt
2. Software interrupt
3. Kernel mode process
4. User mode process
RTU model of protocol processing (1)

• User and kernel mode “threads”.
• Total scheduling:
  – kernel mode
  – user mode
  – software interrupts
QoS guarantees for continuous media

Requirements

- Guaranteed share of CPU.
  (process for time sharing not acceptable)
- Limited priority inversion.
  (concurrency control on shared data causes trouble)
- Reduced context switching cost.
- Reduced mode switching (system call) cost.

Processing nature

- Periodic.
- Small iteration units (per packet).
- Independence between iterations.
RTU model of protocol processing (2)

- Periodic threads.
- Rate monotonic with delayed preemption.
- “Atomic” iteration units.

```
active
  ↓
in_queue
  ↓
complete
```

only at iteration boundaries
Process and RTU priorities

- low latency RTU
- periodic RTU (RMDP)
- best-effort RTU
- kernel mode process
- user mode process
Implementation in NetBSD

- System call
- Callout
- Interrupt
- User process
- User process
- Kernel
- Dispatch
- Process policy
- RTU policy
Implementation in NetBSD (cont’d)

Dispatching at
1. sleep()
2. userret()
3. iteration
boundary

network adapter
Implementation in NetBSD (cont’d)

- Kernel RTU invocations are simply procedure calls.
- User RTU invocations are like signal delivery.
  - No RTU context switching.
  - A small memory area in each process shared by kernel and different RTU iterations
Re-implementing TCP

```
Wrapped at
1. tcp_output()
2. ipintr()
Triggered at
1. tcp_userreq()
2. network adaptor
3. connect()
4. accept()
```
Problems

• Reasonable priority assignment?
• Very disciplined programmer, must guarantee:
  – short iteration, non-blocking.
  – explicit check of yield condition.
• Sound API, estimate CUP usage and period.
• The hosting process must be runnable, and switched anyway.
• Over-simplified protocol processing model
  – ack, retransmission, ...
  – socket layer: buffer shortage, synch on buffer, waiting for incoming events ...
• Periodic system calls (moving data between application and socket layer buffers).
• Does BSD also assume “atomic” processing?
Known bugs

- Scheduling points
- Handler wrapping
- Handler triggering
Future directions

• Fix design and implementation problems or come up with a new model?
• Multiprocessors.