Protocol Implementation

Digital Communications II

Michaelmas Term 2007
Based on Prof. Jon Crowcroft’s notes, and thus transitively on
S. Keshav’s “An Engineering Approach to Computer Networking”

Protocol implementation

- Depends on structure and environment
- Structure
  - Partitioning of functionality between user and kernel
  - Separation of layer processing (interface)
- Environment
  - Data copy cost
  - Interrupt overhead
  - Context switch time
  - Latency in accessing memory
  - Cache effects

Partitioning strategies

- How much to put in user space, and how much in kernel space?
  - Trade-off between
    - Software engineering
    - Customisability
    - Security
    - Performance
- Monolithic in kernel space
- Monolithic in user space
- Per-process in user space

Monolithic in kernel

```
User
  Kernel
  Session
  Transport
  Network
  Device Driver
  Device
  Application
  Application
```
Monolithic in user space

Per-process in user space

Interface strategies
- Single-context
- Tasks
- Upcalls

Single context
### Tasks

- Application
- Buffer Pool
- Device
- Transport
- Network
- Transport
- Datalink

### Upcalls

- Send EntryPoint
- ReceiveEntryPoint
- RegistrationEntryPoint

### Protocol implementation

- Upper Layer Send
- Upper Layer Receive
- Timeout
- Lower Layer Send
- Lower Layer Receive

### Some numbers

- 10 Kbps: 400 ms
- 100 Kbps: 40 ms
- 1 Mbps: 4 ms
- 100 Mbps: 40 µs
- User-to-kernel context switch: ~40 µs
- Copying the packet: ~25 µs
- Checksum in software: ~40 µs
- Scheduling delays: ~150 µs (depends on workload)
- Interrupt handling: ~10-50 µs (depends on the bus)
- Protocol processing: ~15 - 100 µs (depends on protocol complexity)
Rules of thumb

- Optimise common case
- Watch out for bottlenecks
- Fine tune inner loops
- Choose good data structures
- Beware of data touching
- Minimize number of packets sent
- Send largest packets possible
- Cache hints
- Use hardware
- Exploit application properties