Adding a Flow-Oriented Paradigm to Commodity Operating Systems

Cristian Soviani, Stephen A. Edwards, and Angelos Keromytis

Department of Computer Science, Columbia University in the City of New York

{soviani,sedwards,angelos}@cs.columbia.edu
The Status Quo: Memory as Buffer

Web Server

OS Kernel

Main memory

DATA

Hard Disk
MPEG encoder
Crypto Accelerator
Network Interface

Memory–I/O bus (e.g., PCI)

Network
I/O Becoming Faster than CPUs

CPU and Memory speeds

(Source: Intel)
Our Idea: Flows Controlled by OS

Like a modern router’s control & data plane
Layers

Typical

- User
- Kernel
- HW

e.g., `sendfile()`

Ours
Abstract Server Model

Sources
- Hard Drive
- Video in
- Network

Transformers
- Crypto
- MPEG encode
- MPEG decode

Sinks
- Network
- Hard Drive
- Audio Out
Signaling API

Need some sort of mechanism for creating, controlling, and tearing down flows, e.g.,

```c
flow = flow_open();
flow_source(flow, "/usr/http/secretfile.html");
flow_xformer(flow, "/dev/crypto");
flow_xformer(flow, "/dev/http");
flow_sink(flow, "/dev/inet/192.168.1.3");
flow_start(flow);
flow_stop(flow);
```
Exception Handling

What happens if something goes wrong?

- OS may try to re-start device
- Redirect to a different device
- Error passed to application
- Switch to all-software flow
- Terminate
What if two processes want access to /dev/crypto?

- Performance requirements
- Available resources
- Priorities

Flow-level scheduling costly, but infrequent

Detailed (e.g., bus access) scheduling more frequent
Programmable Peripherals

RadiSys ENP-2611
Network Card w/ IXP2400

Altera Stratix PCI

FPGAs: flow components or wrappers around legacy peripherals (DMA absorbers)
Proof-of-Concept System

- **SRAM 16kB**
- **CPU0**
- **LRU arbiter**
- **NIC CPU**
  - **SRAM 8kB**
  - **8kB dual ported SRAM**
  - **private circuitry**
- **CRYPTO CPU**
  - **SRAM 8kB**
  - **8kB dual ported SRAM**
  - **private circuitry**
- **HDC CPU**
  - **SRAM 8kB**
  - **8kB dual ported SRAM**
  - **private circuitry**
- **SNIC CPU**
  - **SRAM 8kB**
  - **8kB dual ported SRAM**
  - **private circuitry**

**General purpose peripherals**
- **Interrupt controller**
- **USER I/O**
- **JTAG UART**
- **VGA controller**

**Main system memory**
- **off-chip 2MB ZBT SRAM**

**NIC device**
- **CRYPTO device**
- **HDC device**
- **SNIC device**
## Experimental Results

Time to send 1 Million packets through the pipeline

<table>
<thead>
<tr>
<th>Packet size</th>
<th>Memory-centric</th>
<th>Flow-centric*</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 bytes</td>
<td>15.5s</td>
<td>13.6</td>
<td>13%</td>
</tr>
<tr>
<td>1024 bytes</td>
<td>114</td>
<td>59.2</td>
<td>49%</td>
</tr>
</tbody>
</table>

*Plus main processor mostly idle for these packets
Conclusions

New flow-centric architecture for operating systems
Have the OS manage inter-peripheral flows under application control
Requires programmable peripherals: many already extant
Proof-of-concept showed nearly a $2 \times$ speedup
Minimal performance impact from additional computations (e.g., security)