An Infrastructure for Network Development. Proof of Concept: *Fast UDP*

Edgar A. León
University of New Mexico

Michal Ostrowski
IBM T. J. Watson Research Center

USENIX'05 · HPCS PERCS · Scalable Systems Lab · IBM · The University of New Mexico

Motivation

- HPC applications constrained by computational resources
- Host network bandwidth scales poorly with respect to processor, bus and link bandwidths
- As network speeds increase, incoming network data may overwhelm host processor
- Applications may starve under high network loads
- Host overhead due to communication processing degrades application performance

Goal

Build an infrastructure to:

- Study NIC/OS interaction
  - OS bypass
  - Cache Injection
  - Matching on the NIC
  - Protocol Offloading
  - Interrupt direction and filtering
- Develop and evaluate next-generation Smart Network Interface Controllers

Network Infrastructure

Build infrastructure in *Mambo* architecture simulator

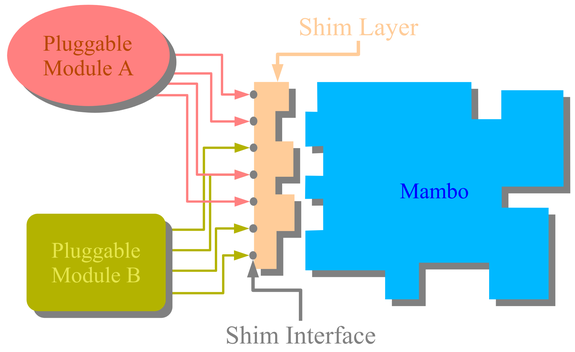
- Problem

  Mambo is not open source
- Objective

  Allow the creation of *pluggable modules*
  - Can be dynamically loaded in Mambo
  - Run as Mambo components
  - Do not need access to Mambo source code

The Shim Layer

- Allows module header space to be independent of internal Mambo headers
- Provides a mambo-independent interface to library modules
- Export functions, not data structures
- Data Structures encapsulated by *Shim Handle*
  - Handle is opaque to Libraries
- Pluggable Modules
  - Dynamically loaded using dlopen
  - Mambo entry points explicitly defined by the *Shim Interface*



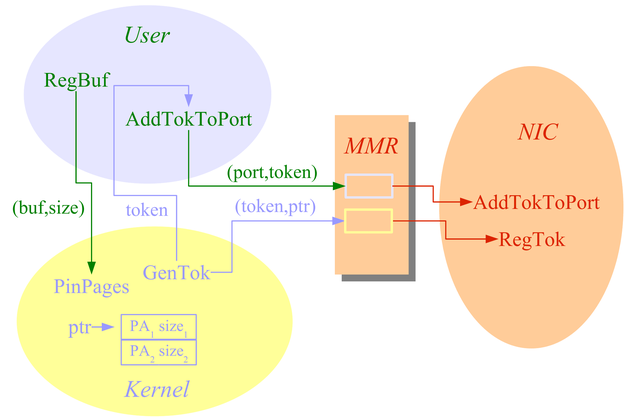
NIC API / Shim Interface

- `mem_write`
- `mem_read`
- `memmap_define`
- `memmap_delete`
- `set_memmap_io_funcs`
- `schedule_job`
- `delay_cycles`
- `raise_interrupt`

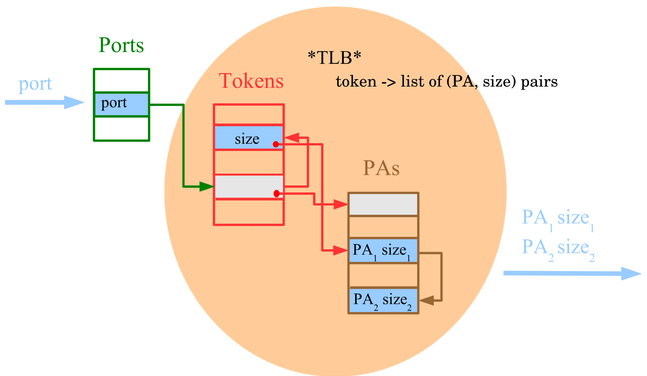
Fast UDP

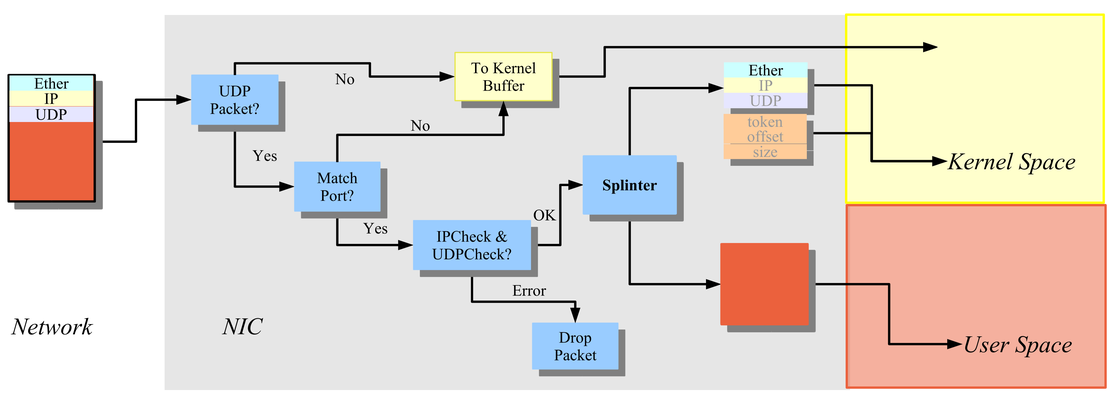
- *Splinter* data from control information
  - Application's data bypasses the Kernel
  - Make data available to applications fast
  - Reduce host overhead due to communication
- *Matching* on the NIC
  - NIC has enough information to perform data placement directly
- NIC *Offload*
  - Splinter, Message Matching, Data Placement, UDP/IP checksum semantics

User / Kernel / NIC Interfaces



NIC Data Structures and Matching





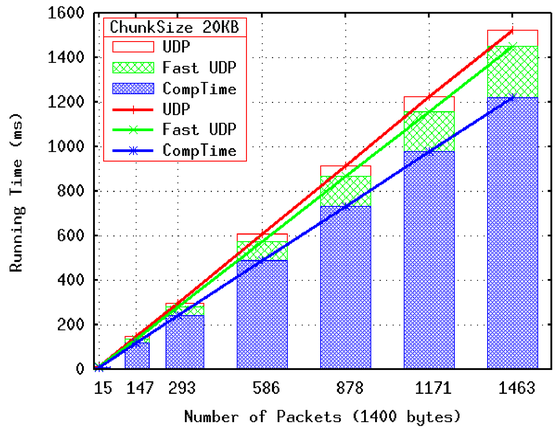
Test Application

```
timestamp

  while(1)
  {
     recvfrom(sock, buff+offset, ...);
     if (offset >= i++ * chunk_size)
       sort_chunk(buff, i);
     if (i == num_chunks)
       break;
  }

timestamp
```

Results



Conclusions and Future Work

- Developed an infrastructure to investigate communication mechanisms that:
  - Improve host scalability
  - Reduce host overhead
  - Improve overall application performance
- Proof of concept: Fast UDP
  - 5% improvement on an 80% computation-bound application
- Extend Shim Interface to allow cache injection
  - Simulated NIC injects data directly into an L2/L3 data cache
  - Investigate the scenarios when this optimization provides positive and negative impact on applications