Routelets and Network Processors

Michael Kounavis
Stephen Chou
Vassilis Stachtos
Andrew Campbell
Columbia University
network architecture development cycle

creation of components

profiling

architecting & management

spawning
first phase: virtual networks on-demand
first phase: virtual networks on-demand
routelet: virtual router

- Programming environment
- State
- Control plane
- Data path

control unit

routelet state

binding interface base

metabus

spawning controller
composition controller
allocation controller
datapath controller

forwarding engine

input port

output port
routelet: virtual router

programming environment

state

control plane

data path

binding interface base

metabus

spawning controller

composition controller

allocation controller

control unit
routelet: virtual router

- host processor
- memory units
- StrongARM Core
- microengines

binding interface base
metabus
routelet state
control unit
spawning controller
composition controller
allocation controller
datapath controller
input port
forwarding engine
output port
our approach

- data path
  - programming data paths
  - data path admission control
  - dynamic binding of data paths
- control plane
  - profiling routelets
  - managing routelet state
- programming environment
programming the data path

- **input ports** → forwarding engines → **output ports**

  - **components**: entry point → transport modules → exit point
  - **symbols**: global variable → transport modules → input argument

  - **virtual**: router specification
  - **network processor specification**
  - **physical**: IXP1200 realization
profiling datapaths

- virtual router related:
  - a graph connecting input ports, forwarding engines, and output ports
  - ports and engines are modular

- network processor related:
  - components are grouped into pipelines.
  - pipeline components are executed by the same threads sequentially
  - component are augmented with symbols

- IXP1200 related:
  - symbols are associated with IXP1200 properties (e.g., register space, microstore addresses etc)
datapath admission control

• datapath
  - specified as a set of pipelines; admitted if:
    • sufficient bandwidth
    • sufficient microstore space
    • sufficient contexts available
    • sufficient number of absolute registers

• pipeline placement:
  - exhaustive search has m! complexity
  - simple and works OK
register assignment

- component registers are used as
  - pipeline registers (context relative GPRs)
  - input argument registers (context relative GPRs)
  - global variables (absolute GPRs)

- allocations
  - static for context relative registers
  - dynamic for absolute registers
register usage

argument registers (addresses: 9-12)

pipeline #1 pipeline registers (addresses: 0-8)

component #1

algorithm #1

pipeline #2

shared absolute registers (addresses: 13-15)

component #2

algorithm #2

component #3

algorithm #3

pipeline #1 pipeline registers

pipeline #2
dynamic binding system

network processor related specification

datapath constructor

IXP1200 related specification

binder

datapath admission control

StrongARM Core

transport modules (.tmd files)

microengines
microassembler extensions

- preprocessor
- assembler
- uof2tmd
- .ucp file
- .uof file
- .tmd file

-.uof
- import variables
- export functions

-.tmd
- input arguments
- global variables
- entry points
- exit points

header
**dynamic binding**

- **transport modules:**
  - are placed into a microstore

- **immed instructions:**
  - are filled with input argument values

- **absolute registers:**
  - are used as global variables
  - assigned during the admission control process
  - their addresses are introduced into the code dynamically

- **branch instructions:**
  - are added to link exit points with entry points
virtual network demultiplexing

- **single-threaded:**
  - the VN demultiplexor branches to the beginning of the routelet code

- **multi-threaded:**
  - the VN demultiplexor places the packet to a queue. The routelet picks up the packet from the queue
managing routelet state

routelet state

SRAM
- routing tables, interface information, SDRAM allocations

SDRAM
- forwarding tables, packet buffers, routelet composition information

Scratchpad
- flags, forwarding MIB, performance stats
testbed
experiments

8 slow port configuration: 3 VNs supported

4 slow port configuration: 7 VNs supported

<table>
<thead>
<tr>
<th>module</th>
<th>VN</th>
</tr>
</thead>
<tbody>
<tr>
<td>vn demux</td>
<td>-</td>
</tr>
<tr>
<td>capacity arbitrator</td>
<td>-</td>
</tr>
<tr>
<td>verifiers</td>
<td>IPv4</td>
</tr>
<tr>
<td>modifier</td>
<td>IPv4</td>
</tr>
<tr>
<td>lookup</td>
<td>IPv4</td>
</tr>
<tr>
<td>queue</td>
<td>IPv4 &amp; Cellular IP</td>
</tr>
<tr>
<td>lookup</td>
<td>Cellular IP</td>
</tr>
</tbody>
</table>
IPv4 throughput

![Graph showing IXP1200 vs BSD IPv4 Throughput](image)

- **Throughput (Mbps)**
- **Packet size (bytes)**

Legend:
- Pink line: payload+header
- Yellow line: user-space BSD
Architecting Cellular IP

- hard handoff
- semisoft handoff (1 buffer)
- semisoft handoff (8 buffer)
Genesis Developer Workbench
IXopath Release

- ~15,000 lines
  - IXpath libraries,
  - IX libraries for Arm-Linux, microcode module, application and tool.

- methodology:
  - Dynamic binding of microcode components

- Release webpage:
  http://www.comet.columbia.edu/genesis/ixpath
IXpath v1.0

dynamic binding support for the IXP1200 network processor

overview manual code genesis home

0 downloads of ixpath since September 2001

genesis@comet.columbia.edu
results (summary)

• **Genesis kernel v1.0 developed**
  - datapaths implemented on IXP1200
    - www.comet.columbia.edu/genesis/ixpath
  - **Genesis Developer Workbench front-end**
  - **Genesis testbed consisting of 3 IXP1200 evaluation platforms**

• **Publications**
  - **Genesis Kernel Design** (JSAC March '01),
  - **Programmable Routing Protocols** (OPENARCH, April '01)
  - **Dynamic Binding** (technical report, under submission)
thanks for listening
Genesis Kernel Release

- **XML script enhancements**
  - support for virtual links
  - dynamic binding of components
  - multiple data paths

- **IXP 1200 platform specific support:**
  - allocation controller:
    - IXP resource reservation (memory, threads, ports)
  - composition controller:
    - components from TMD transport modules,
    - instantiation of C++ objects from shared libraries
  - Routelet state:
    - distributed over different memory types