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

IP Router

IXP1200

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

- Programming environment
- State
- Control plane
- Data path

- Binding interface base
- Metabus
- Routelet state
- Control unit
- Spawning controller
- Composition controller
- Allocation controller
- Datapath controller
- Input port
- Forwarding engine
- Output port
routelet: virtual router

programming environment

state

collection plane

data path

binding interface base

metabus

control unit

spawning controller
composition controller
allocation controller
routelet: virtual router

- host processor
- memory units
- StrongARM Core
- microengines
- binding interface base
- metabus
- routelet state
- spawning controller
- composition controller
- allocation controller
- datapath controller
- control unit
- 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 \quad \rightarrow \quad \text{forwarding engines} \quad \rightarrow \quad output ports

virtual: router specification

network processor specification

physical: IXP1200 realization

components \quad \rightarrow \quad symbols

exit point \quad \rightarrow \quad entry point

global variable \quad \rightarrow \quad \text{input argument}

transport modules
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
  - components 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 #2

shared absolute registers (addresses: 13-15)

component #1

algorithm #1

component #2

algorithm #2

component #3

algorithm #3

pipeline registers (addresses: 0-8)

pipeline #1
dynamic binding system

network processor related specification

datapath constructor

IXP1200 related specification

binder

microengines

transport modules (.tmd files)

StrongARM Core

datapath admission control
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>
Genesis Developer Workbench
The NetBind Tool

• ~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/netbind
NetBind Release

NetBind

a binding tool for constructing data paths and virtual routers
in network processor-based routers

overview manual code papers genesis home

4 downloads of NetBind since December 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