Implementing Spawning Networks

Michael E. Kounavis
COMET Group,
Columbia University
Motivation

- deployment of network architectures:
  - ad-hoc in nature and costly
  - no systematic exploration of the “network design space”

- virtual networking
  - isolation and privacy
  - quality of service support
Enabling Technology: The Genesis Kernel

network architecture #1

life cycle services

binding interface base

metabus #1

transport environment

network architecture #2

metabus #2

child virtual network kernel

parent virtual network kernel

programming environment

spawning
Implementation

• approach:
  • we have ‘virtualized’ the networking code from the 4.4BSD kernel

• features:
  • ‘generic’ link layer interfaces
  • dynamically selected ARP
  • modular routelet components

• routelets:
  • IPv4, Cellular IP, Mobiware
IPv4 Routelet

- **ARP**
- **virtual ethernet**
- **virtual network demultiplexor**
- **virtual ethernet**
Programming Environment

- transport
- QOS control
- routing
- management

virtual network architecture

- VirtualRouteletState
- VirtualSpawningController
- VirtualCompositionController
- VirtualDatapathController
- VirtualAllocationController

binding interface base

metabus

life cycle services

- Spawner
- Architect
- Profiler
- Maestro
Implementation

- **approach:**
  - we have modified an existing ‘off-the-shelf’ middleware technology (OmniORB)

- **features:**
  - **metabus:**
    » dedicated middleware support on a per virtual network basis
  - **orplet core:**
    » a virtual Object Request Broker core component
  - **meta-servers:**
    » dynamically deployed naming services
Metabus

metaservers

ORBllet core

Inter-ORB protocol engine

IPC

routelet
Profiling Process (I)

- approach:
  - we separate the ‘binding rules’ characterizing a programmable network architecture from the ‘binding data’

- binding rules:
  - routelet composition rules
  - network object deployment rules

- binding data:
  - network topology
  - system parameters
Profiling Process (II)

• deriving the analytical form:
  • topology description is converted from the address space of the ‘parent’ to the address space of the ‘child’ network
  • binding rules are associated with binding data
Example: Cellular IP

```xml
<?xml version="1.0"?>
<compact_form>
<binding_rules>
<architecture>"cip"</architecture>
<node_types>
<type>
<name>"base_station"</name>
<data>
<parameter>"number_of_leaves"</parameter>
<parameter>"root_address"</parameter>
<parameter_array>
<length>"number_of_leaves" + 1</length>
<parameter>
<name>"vn_demuxors"</name>
<type>VN_DEMUX</type>
</parameter>
</parameter_array>
<parameter_array>
<length>"number_of_leaves" + 1</length>
<parameter>
<name>"arbitrators"</name>
<type>ARBITRATOR</type>
</parameter>
</parameter_array>
</data>
</type>
</node_types>
<routelets>
<routelet>
<name>"cip_routelet"</name>
...
Spawning Process

- **spawning services:**
  - spawner
  - constructors
  - component storage

- **network creation steps:**
  - datapath creation
  - programming environment creation
  - signaling plane deployment
Status of the Work

- we have been able to spawn simple and well-understood network architectures
  - Standard IPv4
  - Mobiware
  - Cellular IP
- some issues are still not resolved
  - architecting
  - VPN resource management
  - scalability
Spawning Testbed
On-going Work

- porting the code to IXP1200
- signaling engines
- investigating optimality:
  - cost of a signaling system?
  - domain partitioning?
  - separation between timescales?
People

- Andrew T. Campbell
- Stephen Chou
- Michael Kounavis
- Vassilios Stachtos
- John Vicente
thanks for listening!