Finding The Right Abstraction for a Modular Router:
An Axiomatic Basis

S. Keshav and M. Karsten *
University of Waterloo
Waterloo, ON Canada

*Joint work with Sanjiva Prasad, IIT Delhi and Omer Beg, UW
Waterloo?

Where is that?
Local companies
Seagram
RIM/BlackBerry
MapleSoft
OpenText
ManuLife

School of CS
77 faculty
~2000 undergrads
~300 grads
Finding The Right Abstraction for a Modular Router: An Axiomatic Basis

S. Keshav and M. Karsten *
University of Waterloo
Waterloo, ON Canada

*Joint work with Sanjiva Prasad, IIT Delhi and Omer Beg, UW
Router or not?

Core router: Yes
Enterprise router: Yes
Load balancer: Yes
Firewall: Yes
Majordomo: Yes
Orkut member: Yes
Edge router: Yes
Ethernet switch: Yes
NAT box: Yes
Spam filter: Yes
IP layer: Yes
Simplest possible router

Abstract Switching Element (ASE)
Direct connection
Another direct connection
Combining ASEs
Theory
Reality

[Image of spaghetti with meat sauce]
Reality

- Many layers: Socket, Session, TCP, MPLS, IP, PPP, VLAN, ATM, MAC
- Middleboxes: Firewalls, NATs, shapers, load balancers, DNS redirectors...
- Cross-layered designs
Yet... it works!
SO IT WAS TRUE... THERE WAS A METHOD TO HIS MADNESS...
Axioms

-> = ‘leads to’

• Direct communication
  m@output port -> m@input port

• Simple switching
  if \( p \) is an ASE’s switching table with a translation \( p' \),
  \( pm@input \text{ port} \rightarrow p'm@output \text{ port} \)

• Transitivity
  if \( m@input \text{ port} \) of ASE A, and tables at A, B, C, ..., K are set up properly, \( m \) will reach output port of K
Concepts

• **Name**

  if pm@input port of ASE A, and tables at A, B, C, ..., K are set up properly, m will reach output port of K, and *p is its name*

• **Address**

  if two ASEs send a message with a given name to the same destination, the name is also an address

• **Name scope**

  set of ASEs where a name leads to the same destination

• **Routing**

  process of maintaining consistent forwarding within a particular naming scope
Data plane primitives

- push
- pop
- swap
- send
- receive
- copy
Control plane primitives

- Essentially manipulate the state table
- *update the state table*
- *get label from a control message header*
- *set label in a control message header*

- + a few other minor details
Universal forwarding loop

```c
process(ase prev, message msg) {
    bool setup = (ctl(msg) == SETUP
                  || prev in this->SETUP_ASE);
    name lin, lout;
    if (setup) lin = lout = getlabel(msg);
    name n = pop(msg);
    {<ase, name>} S = lookup(prev, n);
    if (!S && this->RESOLVE_ASE) {
        resolve(n); // wait for S update
        S = lookup(prev, n);
    }
    for each <ase, name> s_i in S {
        if (s_i.ase == this) { // local
            if (ctl(msg) == RLOOKUP) {
                respond(prev, msg, n, s_i.name);
            } else if (ctl(msg) == RUPDATE) {
                update(msg);
            } else {
                // other local control activity
            }
        } else { // forward
            message outmsg = copy(msg);
            push(outmsg, s_i.name);
            if (setup) {
                if (VC) lin = local_name(prev, n);
                update(s_i.ase, lin, prev, lout);
                setlabel(outmsg, lin);
            }
            send(s_i.ase, outmsg);
        }
    }
}
```
What can we do with this?

- Can build any forwarding system as a composition of specializations of the universal forwarding loop
- Can formally verify the correctness of any router using Hoare logic
Building a router

Config file

ASE specifications

Fwd.u

ASE₀.n

ASEₙ.n

Base Element

Click Element₀

Click Elementₙ

Fwd.click

Click exe

Forwarder