An Axiomatic Basis for Communication

M. Karsten[1], S. Keshav[1], S. Prasad[2], M. Beg[1]
[1]David R. Cheriton School of Computer Science, University of Waterloo
[2]Department of Computer Science and Engineering, IIT Delhi

mkarsten@uwaterloo.ca

Introduction

ABC – rigorous yet intuitive way to think about (describe, understand, analyze, implement, etc.) communication networks

For example, did you know that

- NAT = ATM
- source routing is heavily used in the Internet

The Internet "Architecture"

- Original Internet Assumptions
  - static public IP address
  - 5-layer stack
  - no layer violations
  - forwarding based only on IP routing tables

In fact...

- All the original assumptions are violated
  - DHCP, NAT, Mobile IP → dynamic IP
  - many more layers: VLAN, P2P, MPLS,...
  - layering extensively violated: NAT, firewall, DNS redirection,...
  - forwarding based on VLAN ID, MPLS label, source IP,...

But...

- It still works
  - mostly
  - for most people
- Why?

Hypotheses

- changes preserve architectural invariants
  - 'axioms' of communication
- use axioms to intuitively understand networks
- ...as well as formally describe/analyze networks
  - e.g. deliverability of messages
- expressive meta-language to implement *any* packet forwarding scheme.

Divide and Conquer

- We are only studying connectivity (naming, addressing, routing, forwarding).
- Other areas, such as medium access, reliability, flow control, congestion control, and security, are ignored (for now).

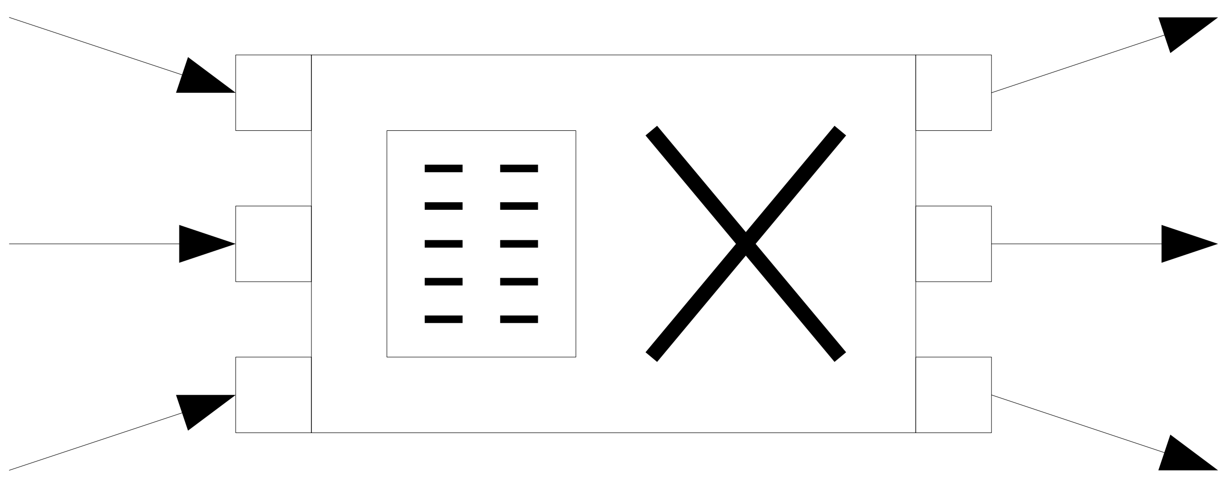
Outline

- Introduction
- **Axioms of Communication**
- Notes on Formalization
- Universal Forwarding Engine
- Conclusions

Notation / Definitions

Abstract Switching Element (ASE)

- switching table S_B: $\langle A,p \rangle \mapsto \{\langle C,p' \rangle\}$
- direct communication via ports: ${}^{A}B$, B^{C}
- message m at port x: m@x



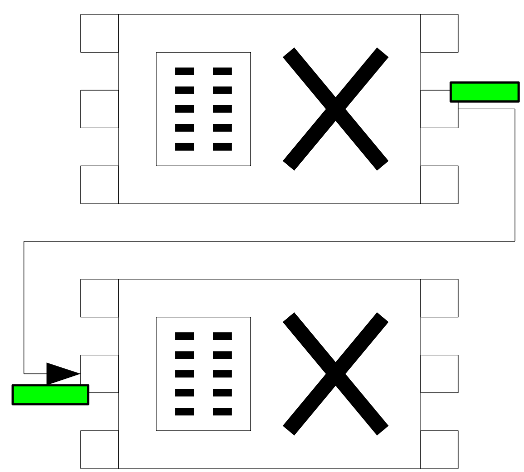
Axioms - Leads-To Relation

LT1 (Direct Communication)

$\forall\ A,B,m : \exists\ A^B, {}^{A}B \Leftrightarrow m@A^B \rightarrow m@{}^{A}B$

- e.g. link, radio

- but also: API

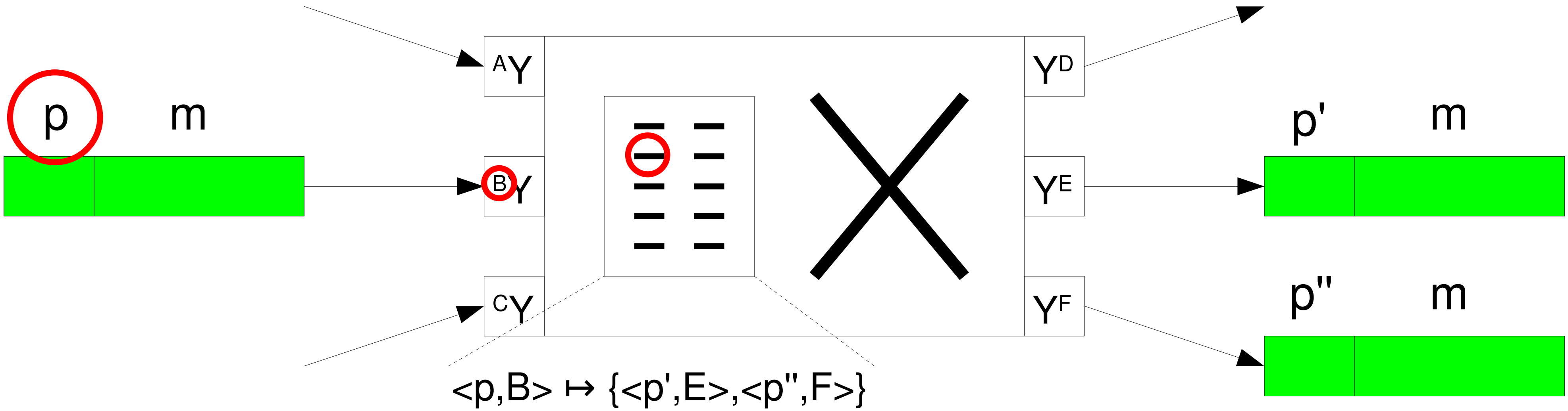


Axioms - Leads-To Relation

LT2 (Local Switching)

$\forall$ A,B,C,m,p,p' : $\exists$ ${}^{A}B, B^{C} \wedge$ <C,p'> $\in S_B[A,p]$

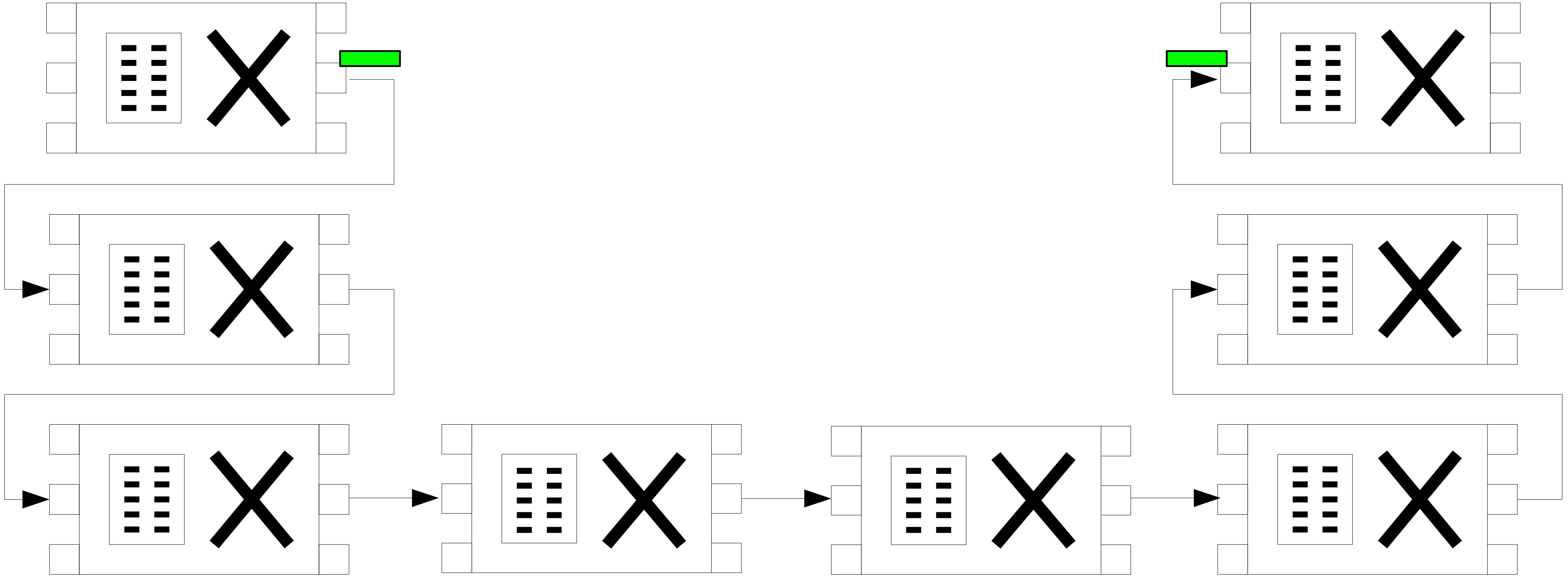
$\Rightarrow$ pm@${}^{A}B \rightarrow$ p'm@B^{C}



Axioms - Leads-To Relation

LT3 (Transitivity)

$\forall$ x,y,z,m,m',m'': m@x → m'@y ∧ m'@y → m''@z ⇒ m@x → m''@z



Axioms - Leads-To Relation

LT4 (Reflexivity)

$m@x \rightarrow m@x$

- simplification of proofs

Communication Concepts

Name

If ∃ ASEs A,B and prefix $p \neq \emptyset$, such that

$\forall m : pm@^{x}A \rightarrow p'm@^{y}B \rightarrow m@B^{z}$ and $p' \neq \emptyset$,

then p is a name for B at A.

p can be stack of ASE identifiers – source routing

Scope: ASEs where name *leads to* same ASE(s)

Name Space: set of names with same scope

Communication Concepts

Address

If ∃ ASEs A,B and prefix $p \neq \emptyset$, such that

$\forall m : pm@^{x}A \rightarrow$ **p**$m@^{y}B \rightarrow m@B^{z}$,

then p is an address for B at A.

...implies common scope along path

Routing: process to establish name space

Forwarding vs. Control

So far: data path only (local state in place)

- algebraic reasoning, e.g. equivalence of name
- formalization of “well-known” concepts

Need also: state setup and remote query

--> Control Patterns

Prefix - Details

Note: prefix p = stack of protocol headers

- need transformations before and after lookup
  - ASE-dependent operations
  - extract relevant fields from protocol header
    - e.g. destination address
  - write back p' into proper header fields

- source stack: logical stack of source fields
- destination stack: logical stack of dest fields

Control Pattern: Path Setup

Deliverability: dest stack ***q*** is name for dest ASE

Returnability: source stack ***r*** name for source ASE

Path Setup

- message qrm arrives from X
- determine r', forward as qr'm to Y
- add/update forwarding state: <Y,r'> ↦ <X,r>

Examples: Ethernet Bridge, NAT, virtual circuit

Outline

- Introduction
- Axioms of Communication
- **Notes on Formalization**
- Universal Forwarding Engine
- Conclusions

Formalization

- previously (HotNets'06): operational semantics
- now: powerful Hoare-style logic
- logic expressed as inference rules

$$\frac{\text{assumption(s)}}{\text{conclusion}}$$

- computation expressed as triples P{S}Q
  - pre-condition P
  - program statement {S}
  - post-condition Q

Outline

- Introduction
- Axioms of Communication
- Notes on Formalization
- **Universal Forwarding Engine**
- Conclusions

Forwarding Operations

Typical transformations from p to p'

- nop – forwarding
- push – encapsulation
- pop – decapsulation
- swap – label switching

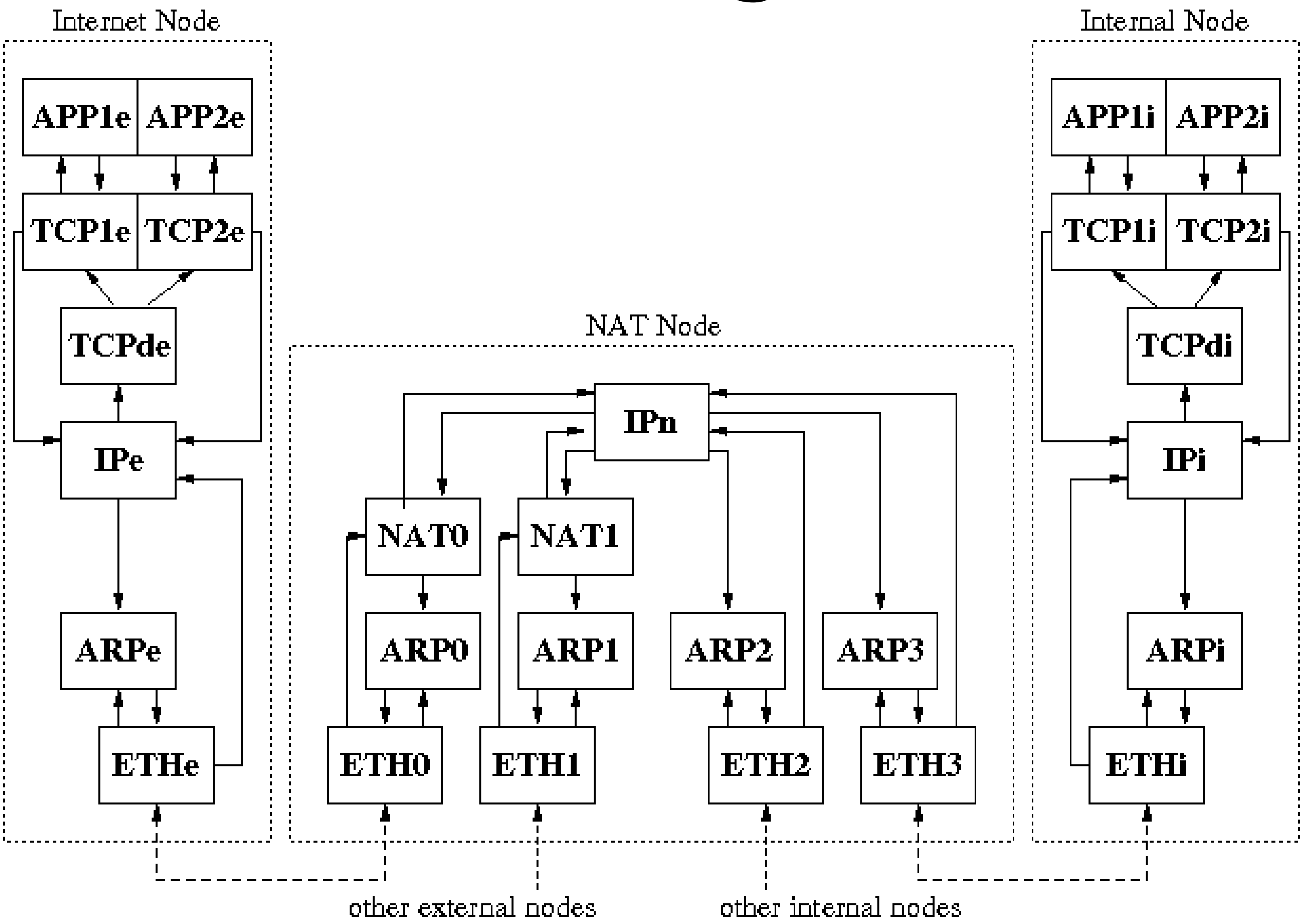
...leads to simple pseudo-code primitives

-

Universal Forwarding Loop

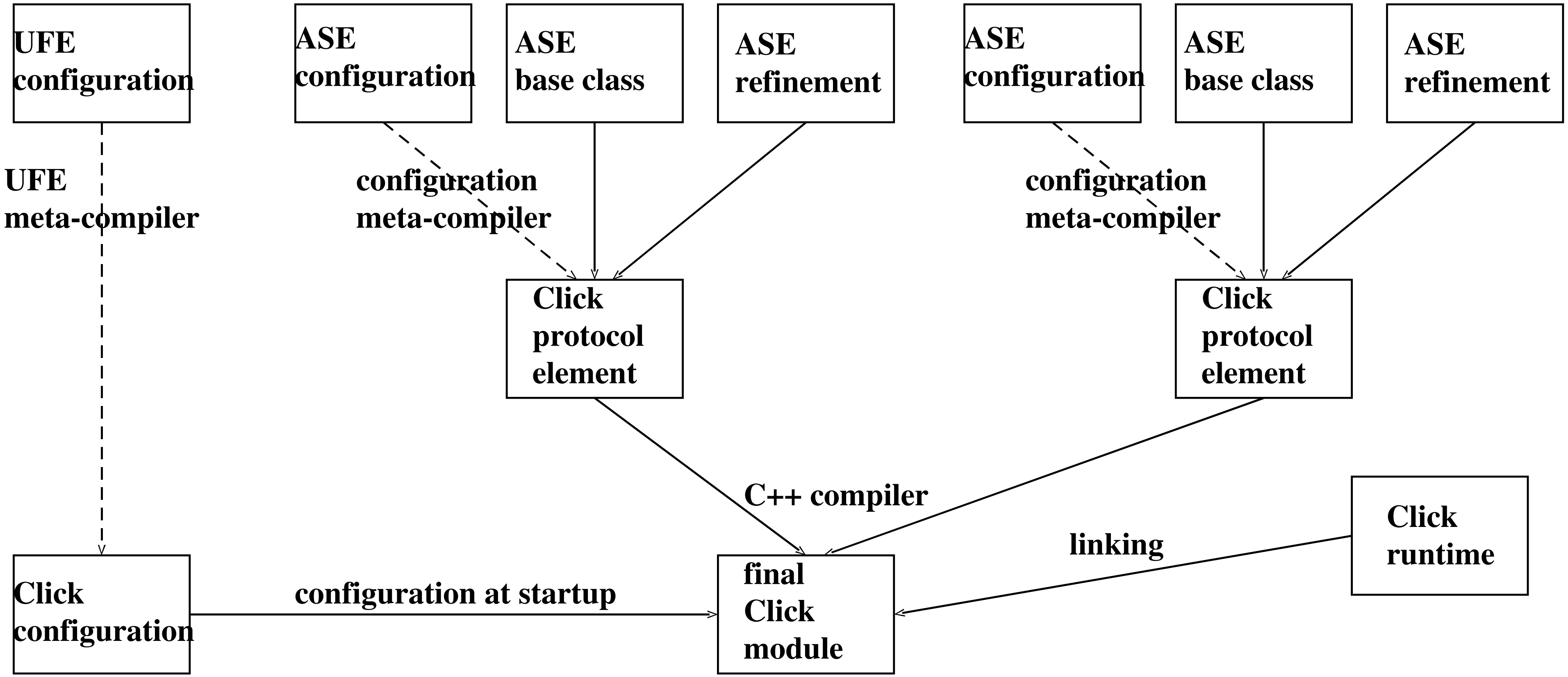
```
bool setup = (ctl(msg) == SETUP || prev in this->SETUP_ASE);
string lin, lout;
if (setup) lin = lout = getlabel(msg);
string n = pop(msg);
{<ase, string>} S = lookup(prev, n);
if (!S && this->RESOLVE_ASE) { resolve(n); S = lookup(prev, n); }
for each <ase, string> s_i in S {
    if (s_i.ase == this)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

Combining ASEs



Prototype

- based on Click router framework



Outline

- Introduction
- Axioms of Communication
- Notes on Formalization
- Universal Forwarding Engine
- **Conclusions**

Example Observations

- Path Setup: NAT ≈ MPLS ≈ ATM
  - outgoing source port ~ label
  - also: hierarchical mobility registration

- Consider forwarding objects in network (rather than “nodes”) --> stack of

  port numbers,
  IP protocol type, IP addresses,
  MAC protocol type, MAC addresses

  ≈ record route and source routing

Conclusions

- The Internet is complex, yet it works.
- We think it's because protocol designers implicitly follow some rules.
- We explicitly state the axioms --> clarity.
- Allows us (hopefully) to do formal analysis: correctness, deliverability, (performance, errors).
- Also allows us to construct a universal forwarding engine.