Extensible Routers

Jeff Chase
Duke University

Motivation

- We've looked at many different proposals for router extensions and changes.
- There are many others (multicast, anycast, IPv6)
- There are huge obstacles to deployment.
  - Nobody owns/controls the Internet
  - Everybody must agree to deploy
    - "You go first"
  - Incentives not in place
- Result: "ossification", frustration

ANTS: A Modest Proposal

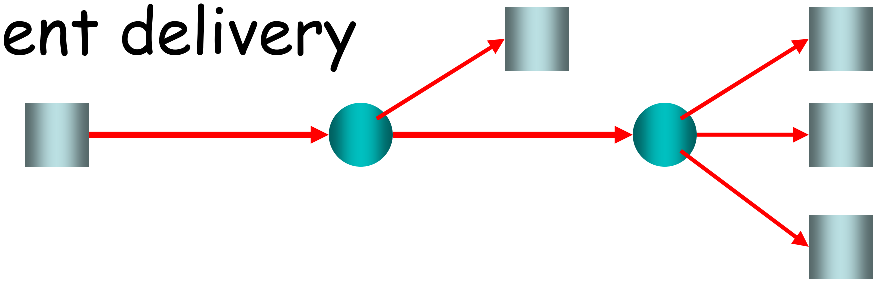
- "Active Networks" [Wetherall, Tennenhouse]
  - "Systematic means of upgrading protocol processing in the network".
  - "Decouple services from the infrastructure"
  - "Untrusted user can freely customize the network"
- Packets are capsules that (conceptually) carry code.
  - Code executes in the routers
  - Anybody can put code in their packets/capsules
- "Reconcile flexibility with performance and security"

What can we learn about research?

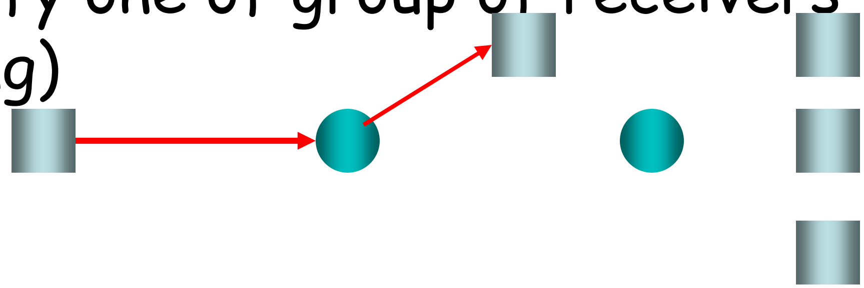
- Philosophical issues:
  - Fantasy ("vision") vs. reality
  - Dream "what if…"
  - Spin vs. science
  - Positive results vs. positive impact
- Massive public investment through DARPA
- Principals and principles moved on
  - E.g., Tennenhouse to Intel
- Focus now on more modest forms of extensibility
    - PlanetLab, network processors

Plethora of (*Proposed*) Useful Network Protocols

- Multicast
  - Specify group of receivers for a message for efficient delivery

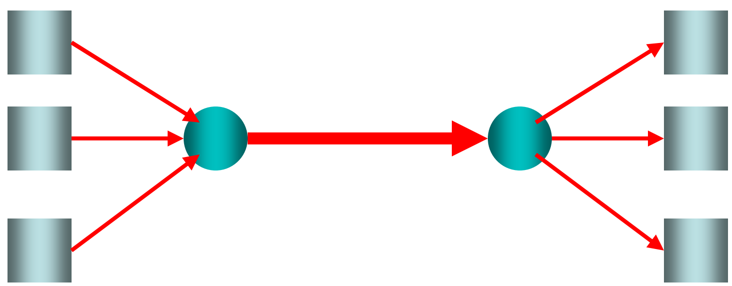


- Anycast
  - Specify one of group of receivers (load balancing, naming)



Plethora of (*Proposed*) Useful Network Protocols

- RSVP
  - Reserve network resources for shared delivery



- IPv6
  - More bits for IP addresses
  - Support for multicast, anycast, RSVP
  - What about newer protocols/variants?

Programmable Networks

- Insert computation into routers
- Associate with each packet (*capsule)* a program responsible for transmitting it to its endpoint
- The entire network adapts to achieve peak efficiency

Active Networking Issues

- Speed
  - Routing in hardware w/o software intervention
  - Running program in the router *will* increase latency
    - Even relative to a fixed software implementation
- Resource allocation
  - Programs in routers consuming unbounded resources
- Safety/Security
  - Restricting access to sensitive resources/program state
- Trust
  - *I'm* going to run *your* code in *my* router?

Caching Fast-Changing Data

- Service that provides rapidly changing information
  - Military information system, airline flight status, stock quotes
- Web Caching?
  - Today's proxy caches cannot cache dynamically generated data (well….)
  - Depends heavily on cache placement
  - Wrong granularity: pages as opposed to objects (My Yahoo)
- Active Networks can be customized to provide:
  - Application-specific cache coherence
  - Application-specific object granularity

AN Caching Protocol

- Quotes cached at Active Nodes on client-server path
- Subsequent requests intercepted to consult cache
- Caches automatically lie on the path between client/server
  - Do not redirect to caches in wrong direction
- Application specific cache coherence
  - Different clients have different requirements for "freshness"
- (Potential) Benefits:
  - Decrease client latency
  - Decrease the traffic at routers
  - Decrease server load

{razor,vahdat}@cs.duke.edu

Rethinking Performance

- Traditional networking metrics:
  - Bandwidth, latency on a packet level
- What really matters is end-to-end performance
  - Application throughput
  - Client-perceived latency
- Active Networks may slow routing down
  - But improve end-to-end application performance
  - Use application-specific notions of throughput/latency

Who Can Introduce New Services?

- Originally, goal was to allow anyone to introduce and test a new service
  - However, issues with wide-area resource allocation makes it important to verify the "correctness" of capsule code
  - Current model requires approval from central authority (such as IETF)
  - Makes deploying protocols slower than original vision, but still much faster than current Internet

Protection Issues

- Need to protect against
  - Node runtime corruption by service code
  - Corrupted/spoofed capsule code
  - Soft state cached at Active Nodes for one protocol manipulated by another service
- How does Active Networks provide protection for above?

Protection Issues

- Need to protect against
  - Node runtime corruption by service code
    - Java
  - Corrupted/spoofed capsule code
    - MD-5 signature
  - Soft state cached at Active Nodes for one protocol manipulated by another service
    - Restricted ANTS API
    - Guarded access to state among separate services
    - Hierarchical service model allows multiple service types to cooperate

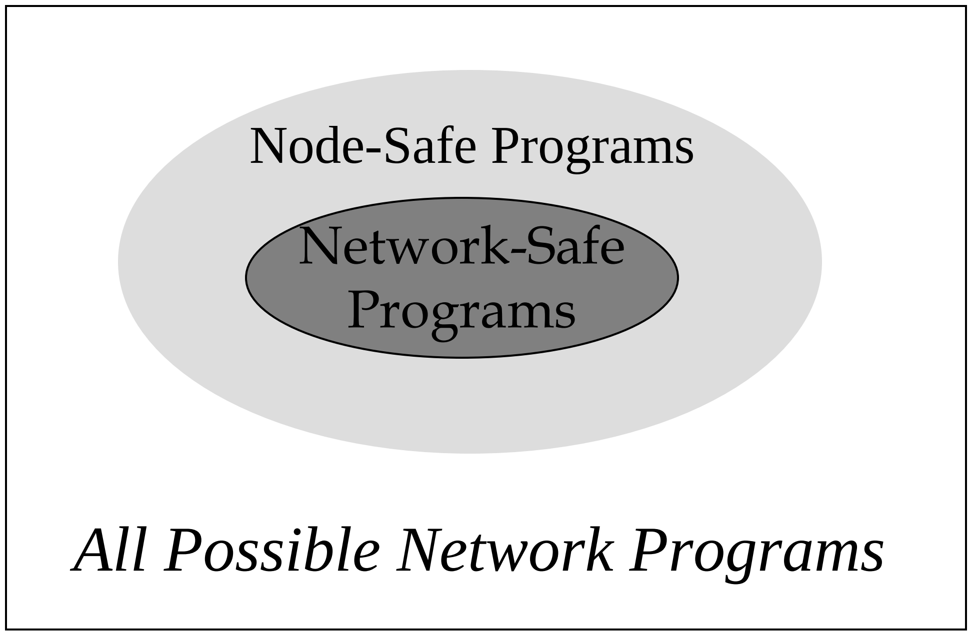
Resource Allocation Issues

- Difficulties with allocating resources in active nets:
  - Single capsule consumes too much resources at active node
  - Capsule and other capsules it creates consume unbounded resources across wide area
  - End application introduces large number of capsules
- How to address these problems?

Resource Allocation Issues

- Difficulties with allocating resources in active nets:
  - Single capsule consumes too much resources
    - Current Java technology allows per-capsule resource consumption limits
  - Capsule and other capsules it creates consume unbounded resources across wide area
    - Difficult problem
    - What resources does a capsule need?
    - Certification
  - App introduces large number of capsules
    - Not well-addressed in either Internet or AN
    - Users cooperate to provide fair access?

Security and Resource Allocation



- Multicast program that spawns two packets at each node

Active Networks Discussion

- Introduce programmability for
  - Rapid introduction of new protocols
  - Increased end-to-end performance
- Rethink network performance in terms of app performance
- Issues:
  - Speed, Resource allocation, Safety/Security
- Active Networks can make explicit "transparent" network caching, network address translation, etc.

Lessons

- Node APIs define the power of the capsule system.
- Capsules may be “glue” to specialized node APIs.
  - “specialized network-embedded resources”
- Soft state and code caching
- Protecting state from code vs. from users of code
- Sandboxing, code signing, code fingerprinting

More Philosophy

- What's the "killer app"?
- Do we need a "killer app"?
- Is any such "killer app" possible for extensibility?
- What kinds of extensions can ANTS support?
  - XCP?
  - Pushback?
  - Any resource control functions?
  - Services vs. "router properties"
- What can ANTS do that we cannot do in an overlay?
- Does ANTS help build better overlays?
- Is this OS research or networks research?

Click

- Software-based router
- Extensible
  - Introduce new elements with new functions
- Configurable
  - Connect elements in a graph
  - Packets take a path through the graph
  - Static checking for legal graph
    - Source all outputs, sink all inputs
    - Match push vs. pull for ports/connectors
    - Queues bridge between push and pull
- Real, fast, real fast

Click Lessons

- Graph model is elegant in its simplicity
- Abstract/decouple the composition of functions from the functions themselves (elements)
  - Functions are local, operate only on packets
    - E.g., queue policies and traffic engineering
  - Elements may have fan-in or fan-out > 1
- A library of predefined elements allows construction of an (almost) standards-compliant router.
- Similar approach has been proposed for Web services (SEDA SOSP 2001)

State in Click

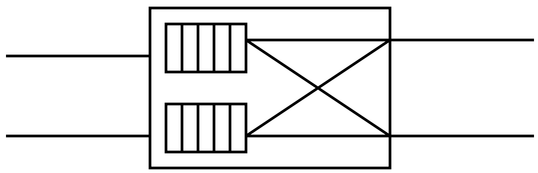
- May pass data downstream via annotations
- Flow-based router context
  - Identify flow path through the element graph
  - Why not an ANTS-like state store?
  - Any notion of "services"?
- Some instances of "inconvenient" global state.
- What about route selection (vs. forwarding)?

The Click Modular Router

{razor,vahdat}@cs.duke.edu

Motivation

- Routers responsible for forwarding arriving data to proper output port



Routing Table

Prefix	Output
xxx	0
yyy	1

- What policy must be expressed in routers?

{razor,vahdat}@cs.duke.edu

Motivation

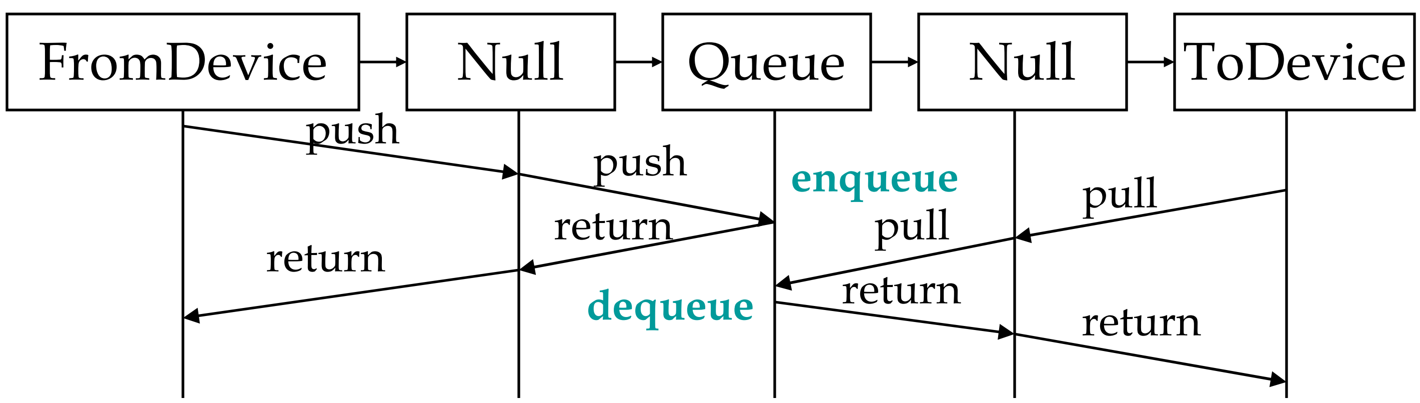
- Policy must be expressed in routers
  - Resource allocation/Quality of Service
  - Congestion control
  - Traffic Shaping
- Existing routers based around proprietary hardware/software extensions
- Commodity operating systems can be modified
  - Complex, a lot of work
  - Click is all about providing a framework for extending router functionality

Click Architecture

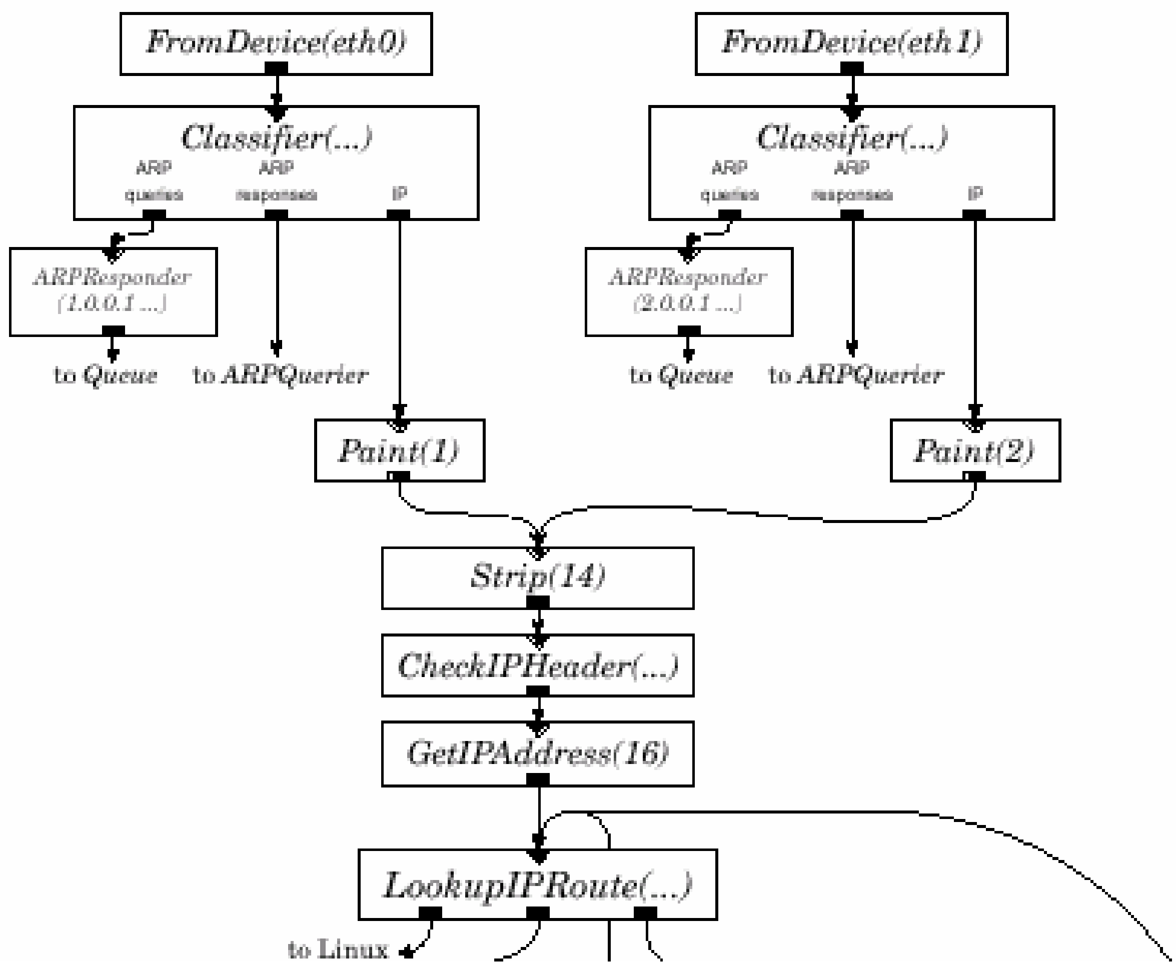
- Elements
  - Object-oriented class determines behavior
  - Queues, flow classifiers, input/output devices
- Input and output ports
  - Connect elements together
- Configuration strings
  - Specify initialization behavior of elements
- Implementation language allows users to specify behavior/configuration of Click Router

Push and Pull Processing

- Data moves through system through both push and pull
  - Packets move from input device through connectivity graph until they reach a queue through *push* operations
  - When output devices are ready to receive new packets, they *pull* packets
    - Pulls move backward through connectivity graph until they reach an element that can provide a packet (e.g., queue)

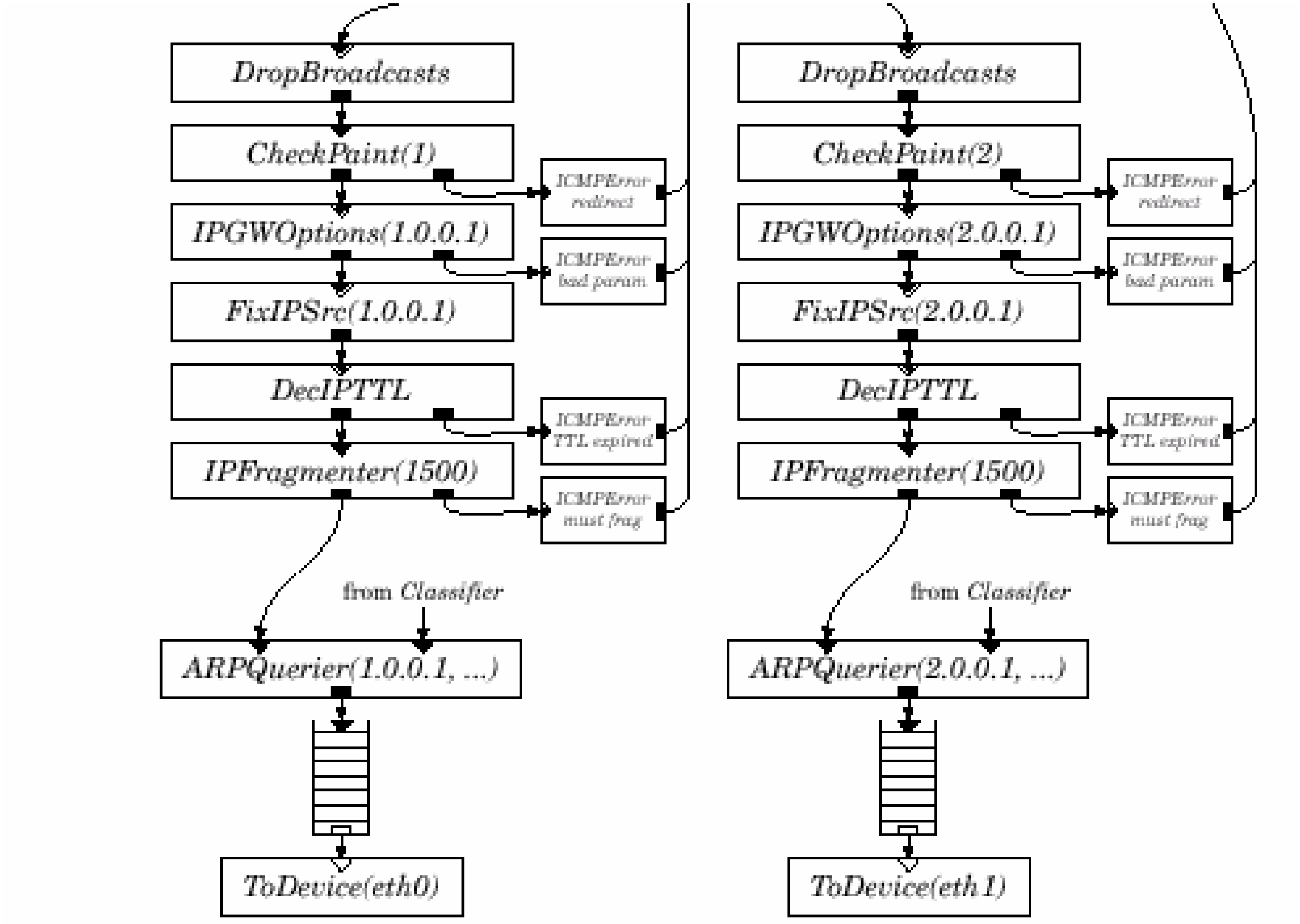


IP Routing



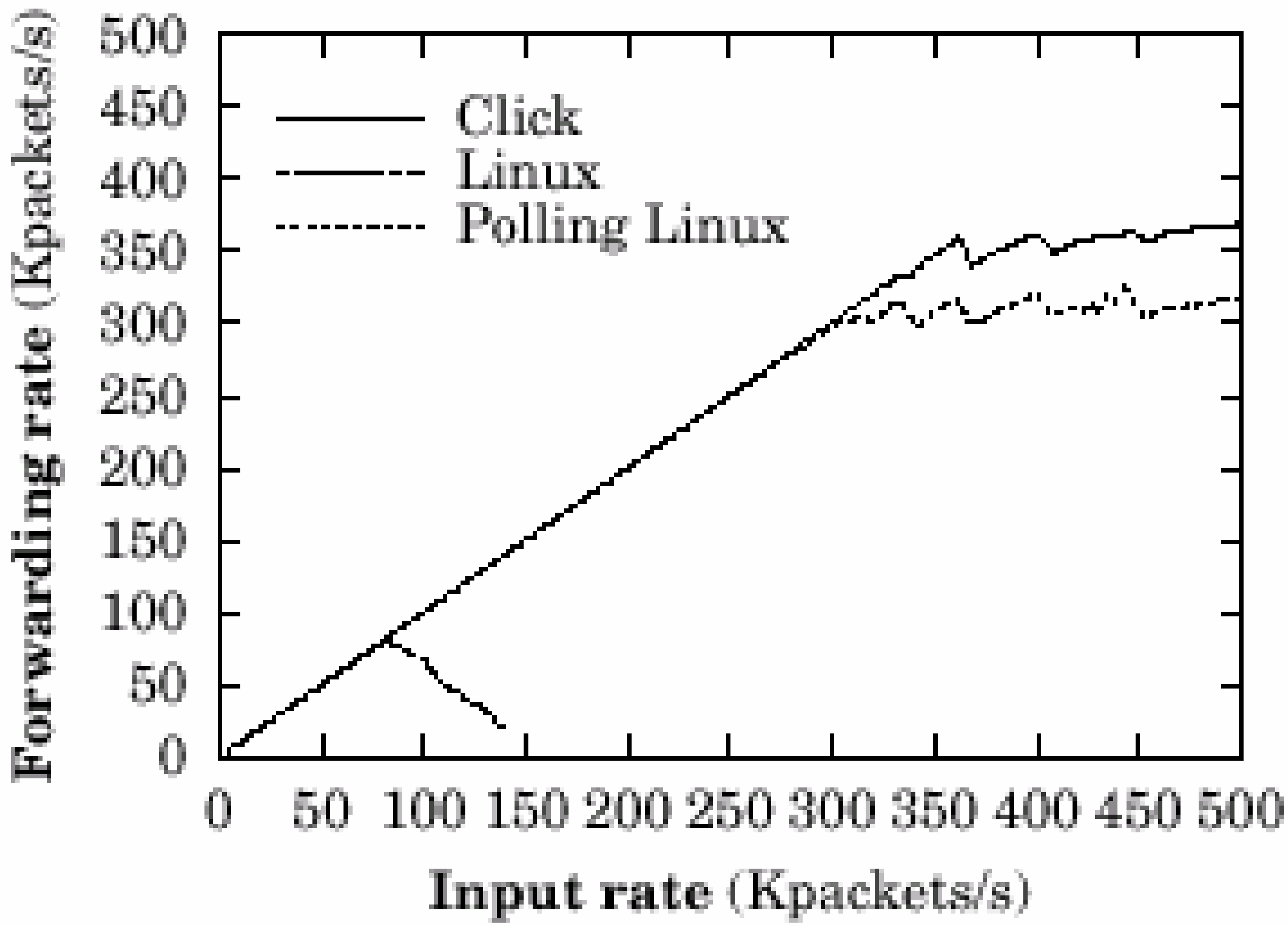
{razor,vahdat}@cs.duke.edu

IP Routing



{razor,vahdat}@cs.duke.edu

Performance



{razor,vahdat}@cs.duke.edu

Discussion

- *Extensibility* key to future systems/protocols
  - Lesson learned from deployment of operating systems, network protocols: do not make decisions that cannot be revisited
- Extensibility comes at what cost?
  - Performance
  - Safety
- Proper abstractions are critical

{razor,vahdat}@cs.duke.edu

Extensible Routers

- Public extensibility (ANTS) vs. ownercontrol (Click)
- Focus on cost of extensibility
- New mechanisms to push functions to NICs
- Control functions in general-purpose processors
- Not rocket science, but is there a market?