

## Active Networks

Presented by Yun Mao  
(slides in part from Jen Rexford, Ion Stoica, and Jonathan Smith)

## Nice Quotation from the Tennenhouse Paper

There is presently a disconnect between what users consider to be "inside" the network and the practitioner's perspective, which is somewhat restricted.

For example, web browsers allow users to interact with what they perceive to be "the network" without distinguishing among the many routers, domain name servers, and web servers that conspire to provide the service.

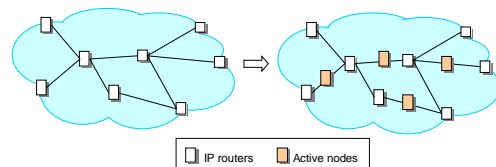
It may be time for practitioners to reevaluate their abstractions and start thinking about the network at a higher level.

## Passive Networks

- Dumb store-and-forward network
  - Smart end hosts implement key functions
  - Simple routers store and forward packets
  - Limited network processing (e.g., routing, forwarding, buffering, and packet scheduling)
- Packet header used in a simple way
  - Common, standardized format
  - Causes one of a small set of operations to occur
  - Packet forwarded or dropped based on those rules
  - Network (largely) ignores higher-layer headers

## Active Network Architecture

- Add active nodes to infrastructure



## Active Networks

- Packet == data + code
  - Smart hosts, as before
  - Active nodes that can execute code on the data
    - Store -> translate -> forward
  - Active packets that carry code to active nodes
- Postscript analogy
  - Contains both your data, and the program the printer runs to print your data
- Definition: Active networks allow an individual user, or groups of users, to inject customized programs into the nodes of the network.

## Motivation for Active Networks

- High-level goal
  - Leverage computation in the network
- User pull
  - Automatically adaptive streaming
  - Data aggregation to reduce data volumes
  - Computation closer to users to reduce latency
- Industry push
  - Ad-hoc collection of middleboxes emerging
  - Replace with generic, multi-purpose active nodes
  - Otherwise, proliferation of active components will happen anyway, without any common framework

### Motivation for Active Networks (Continued)

- Big mismatch in rates of innovation
  - Applications change quickly (e.g., Web, P2P, IM)
  - The network changes slowly
- Deploying new network technology is hard
  - Delay for standardization (at the IETF)
  - Additional delays for vendors to implement and service providers to deploy the new technology
- Better to decouple services from hardware
  - Minimize the amount of global agreement
  - Load new services on demand

### AN is expected?

- Extensibility
  - Lots of efforts from extensible OS (SPIN, exokernel, etc)
  - How about extensible networks? It's kind of a distributed OS after all.
- Programming Paradigm
  - Assembly (imperative)-> C (modular) -> Datalog / SQL (declarative), OCaml, Java (functional / type safe)
  - Network Programming: hand coded -> Click -> P2, PLAN, ANTS

### Active Network, Now, Really?

- Shhh. Don't tell people you are working on AN!
  - "This paper proposes a new active network architecture.." = automatic rejection?
- "Bad" word of mouth
  - Security?
  - Efficiency?
  - Killer applications?

### A Motivating Example



### Motivating Examples

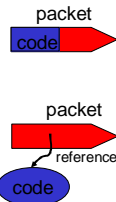
- Customized packet-drop policy
  - User watching video stream (MPEG)
  - Congestion leads to bandwidth limits
  - Drop selectively the B frames
  - Requires application-specific intelligence
- Other examples
  - TCP-SYN filtering
  - Web caching
  - Reliable multicast (or any multicast)
  - Support for mobility

### Two Models of Active Networks (ANs)

- Active networks are active in two ways
  - Switches run code on data flowing through them
  - Individuals can inject programs into the network
- Programmable switches: discrete ANs
  - Separation of program loading and execution
  - E.g. program loading only by network operator
  - Packet is demultiplexed to the right program
- Capsules: integrated ANs
  - Every packet is a program, and carries its code
  - Perhaps in a restricted programming language
  - E.g. ANTS, PLAN

### Where Is the Code?

- Packets carry the code
  - Maximum flexibility
  - High overhead
- Packets carry reference to the code
  - Reference is based on the code fingerprint: MD5 (128 bits)
  - Advantages:
    - Efficient: MD5 is quick to compute
    - Prevents code spoofing: verify without trust



### Three Parts to an Active Network

- Execution environment
  - Virtual machine with access to node resources
  - General, Turing-complete vs. restricted models
- Active applications
  - Provide an end-to-end, customized service
  - Load code on to the routers to program the VM
- Node operating system
  - Support multiple execution environments at once
  - Provide safety between execution environments

### Example: Capsules

- Capsule = code + data
  - Extension of IP packet format
- Type that identifies the code that should handle the capsule
  - E.g., may indicate a Java class
- Code runs in transient execution environment
  - Destroyed when the capsule evaluation ends
- Active storage
  - Capsules can leave information behind in a node's non-transient storage for subsequent capsules
- External methods cached on the node

### Concerns: Security, Safety, and Performance

- Protection
  - Can my service damage yours?
  - Need to run code in a sandbox
- Resource management
  - Can my service consume arbitrary resources?
  - Need careful control over resource allocation
- Performance
  - Can my program complete quickly enough to avoid introducing excessive latency?
  - Need to limit the complexity of the programs
  - ... or run them only on lower-speed links

### Security and Resource Mgmt.

- Untrusted users → need to isolate their actions
- Protection: make sure that one program does not corrupt other program
  - Node level protection
  - Network level protection

### Node Level Protection

- Relatively easy to solve
  - Allocate resources among users and control their usage
    - Fair Queueing, per-flow buffer allocation
  - Use light weight mechanisms: sand-box, safe-type languages, Proof Carrying Code (PCC):
    - PCC can also provide timeliness guarantees e.g., can demonstrate that an operation cannot take more time/space than a predefined constant
- Note: fundamental trade-off between protection and flexibility
  - Example: if a node uses FQ to provide bandwidth protection, it will constrain the delays experienced by a user

### Network Level Protection

- More difficult to achieve
- Challenge: enforce global behavior of a program only with local checks and control
- Main problem: programs very flexible. Active nodes can:
  - Affect routing behavior (e.g., mobile IP)
  - Generate new packets (e.g. multicast)

### Efficiency and Performance

- Running programs on packets
  - Questionable on higher-speed links
  - E.g., where you have just a few nsec per packet
- Feasible at the edge (e.g., 100 Mbps, 1 Gbps)
  - Firewall, NAT, shaper, proxy, intrusion detection
- Feasible for control plane in the core
  - Running routing protocols
- Computer architecture advances help
  - Faster conventional processors
  - Network processors and FPGAs
  - Multi-processor cores

### Performance Case study: ANTS

- ANTS implemented in Java
- In common case little overhead:
  - Extra steps over IP (classification, safe eval) run very fast
- Enough cycles to run simple programs
  - e.g. 1GHz, 1Gbps, 1000b packets, 100% → 1000 cycles; 10% → 10000 cycles
- PLAN, P4 are notably faster

### Summary: Enabling Technologies

- Component-based software engineering
  - Building blocks for composing software
- Code mobility (e.g., OCaml, Java)
  - Though previously between end hosts, not network nodes
  - Innovation in safe and efficient code mobility
  - Performance vs marketing
- Field-programmable gate arrays (FPGAs)
  - Enabling higher speed of packet processing
- Research in programming languages
  - And PL folks' interest in networking

### Stepping Back

- Was active network a success or failure?
  - General idea of computation and services inside the network?
  - Need for a principled approach to middleboxes, and a blurring of router vs. general network node?
  - Specific mechanism of packets carrying code?
- How much flexibility is flexible enough?
  - What granularity: packets vs. flows
  - When is code loaded: on demand vs. in advance
  - Who programs: user vs. network operator

### Active Networks vs. Overlay Networks

- Key difference:
  - Active nodes operate at the network layer; overlay nodes operate at the application layer
- Active Networks advantages:
  - Efficiency: no need to tunnel packets; no need to process packets at layers other than the network layer
- Overlay Network advantages:
  - Easier to deploy: no need to integrate overlay nodes in the network infrastructure
    - Active nodes have to collaborate (be trusted) by the other routers in the same AS (they need to exchange routing info)

## Conclusions

- Active networks
  - A revolutionary paradigm
  - Explores a significant region of the networking architecture design space
- But is the network layer the right level to deploy it?
  - Maybe, but only if all (congested) routers are active (sensor networks?)...
  - Otherwise, overlays might be good enough...