CS 268: Active Networks

Ion Stoica
May 1, 2006

(* Based on David Wheterall presentation from SOSP '99)

Motivations

- Changes in the network happen very slowly
- Why?
  - Network services are end-to-end
    - At the limit, a service has to be supported by all routers along the path
    - Chicken-and-egg problem: if there aren't enough routers supporting the service, end-hosts won't benefit
  - Internet network is a shared infrastructure
    - Need to achieve consensus (IETF)

Motivations

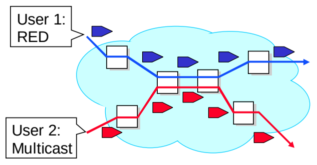
- Proposed changes that haven't happened yet on a large scale:
  - Support for congestion control (RED '93)
  - IP security (IPSEC '93)
  - More addresses (IPv6 '91)
  - Multicast (IP multicast '90)

Goals

- Make it easy to deploy new functionalities in the network → accelerate the pace of innovation
- Allow users to customize their services

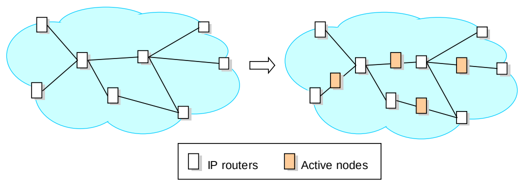
Solution

- Active networks (D. Tannenhouse and D. Wetherall '96):
  - Routers can download and execute remote code
  - At extreme, allow each user to control its packets



An Active Node Toolkit: ANTS

- Add active nodes to infrastructure

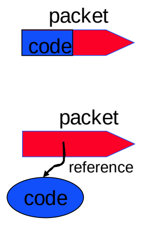


Active Nodes

- Provide environment for running service code
  - Soft-storage, routing, packet manipulation
- Ensure safety
  - Protect state at node; enforce packet invariants
- Manage local resources
  - Bound code runtimes and other resource consumptions

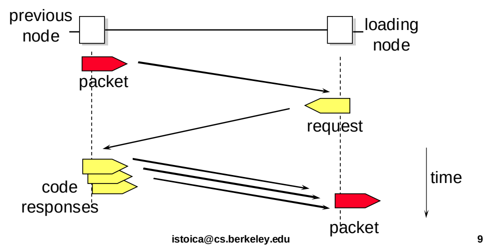
Where Is the Code?

- 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



Code Distribution

- End-systems pre-load code
- Active nodes load code on demand and then cache it



Lesson Learned

- Applications
- Performance
- Security and resource management

Applications

- Well-suited to implement protocol variations
- But not to enforce global policies and resource control (e.g., fire-walls and QoS)
  - Need a central authority to implement these functionalities
- Application examples: auctions, reliable multicast, mobility,…

Performances

- 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

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 predifined 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)

Examples

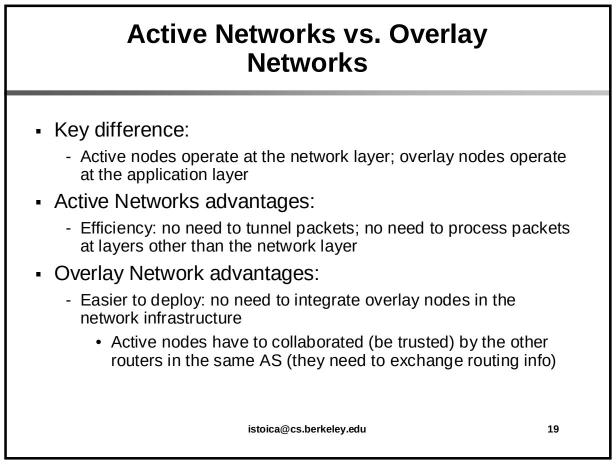
- Loops as a result of routing changes
- Resource wastage as a result of misbehaving multicast programs
  - Multicast height k, a node can generate up to m copies → total number of packets can be $O(m^k)$!
- Local solutions not enough
  - TTL too weak; unaware about topology
  - Fair Queueing offers only local protection

Solution

- Program certification by a central authority
- Limitations:
  - Slows innovation, but still better than what we have today
  - Dealing with a misbehaving node still remains difficult

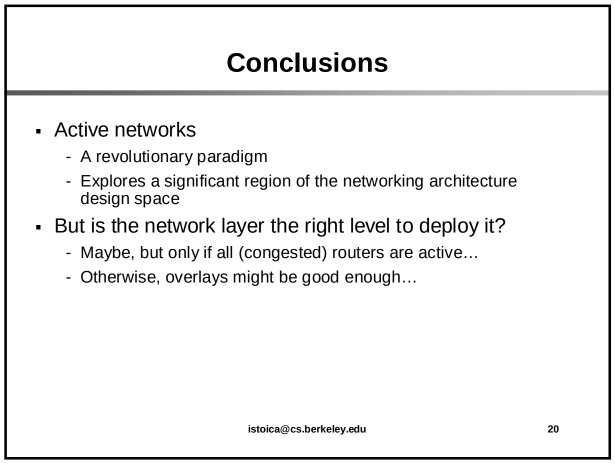
Restricting Active Networks

- Allow only administrators, or privileged users to inject code
  - Router plugins, active bridge
- Restrict affecting only the control plane → increase network manageability
  - SmartPackets
  - Netscript



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 collaborated (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…
  - Otherwise, overlays might be good enough…