Active Networks
“Towards an Active Network Architecture,”
D. Tennenhouse, D. Wetherall,
CCR 1996
Winner of ACM SIGCOMM 2007 “Test-of-Time Award”
Internet in 1996: Routers are passive -- just move bits around
Bits are either dropped or delivered unaltered
Routers is a close platform. Only vendors can modify functionality at routers.
Hard to deploy new services
Example:

IPv6
IP Multicast
RED
Internet evolves slowly compared to PC and Web
Web and PC florishes because anyone can easily deploy new application and services (they’re programmable!)
Two more examples:

Facebook
Second Life
Idea: Let’s make the Internet programmable
Users can insert code into the network and run computations on packets
Users can insert code into the network and run computations on packets

1. Cisco etc.
2. Authorized Vendors
3. End users
Users can **insert** code into the network and run computations on packets

1. install program onto router
2. packet carries program
Users can insert code into the network and run computations on packets

1. program/function name
2. scripts
3. binaries
Users can insert code into the network and run computations on packets.

1. special, “active nodes”
2. any routers
Users can insert code into the network and run computations on packets

1. network (eg: routing)
2. transport (eg: packet filtering)
3. application (eg: compression)
Examples of Services in Network
Authorized application vendors can program firewall to let their packets through
Users can adapt video to fit their bandwidth/screen-size
downsize the video for me please
Shrink as close to the source as possible
Web caches can generate dynamic web pages
rotate these ads for me please
Wireless base station can retransmit packets
Sender -> BS -> Receiver

3 Dup ACKs
Sender -- BS -- Receiver

Retransmit
Approaches to Active Network
Two Approaches to Active Network

1. Discrete
2. Integrated
Discrete Approach
Packets are send normally, but header identifies additional function to operate on the packet (possible changing it)
Integrated Approach
Packets carry code with them, code gets executed from node to node
Capsules = “Packets that carry code (and maybe data)”
Examples
ack() { print "ok"}

ping(src, dest) {
    if this is dest
        eval(src, ack())
    else
        eval(dest, ping(src, dest))
}
```cpp
ack(x) { print x}
traceroute(src, dest, x) {
    if this is dest
        eval(src, ack(x))
    else
        next = getNextHop()
        eval(next, traceroute(src, dest, x+1))
}
```
Execution Environment for Capsules
Issue: need to restrict the capability of capsules

(e.g. my capsules shouldn’t delete your capsules, or change the routing tables of other capsules)
Issue: need to limit the resources used by capsules

(e.g. a capsule that goes into infinite loop should not hang the router, or should not replicate itself infinitely)
Issue: capsules need to be executed on a variety of platforms
We know how to do this using virtual machines and sandboxes (e.g. java applets)
Router provides basic API to access routing tables, links information (e.g. getNextHop())
Capsules may leave states behind in the executing environment.
Example: in information fusion applications
Example: roll call -- find out how many multicast receivers are there
var total = 0, count = #children

\[call() \{\]
  if no children eval(parent, reply(1))
  for each child c
    eval(c, call())
\[

\[reply(x) \{\]
  total += x; count --
  if (count is 0) eval(parent, reply(total))
\[
}
Will need to support garbage collection of states and execution environment
Active Network and E2E Arguments
E2E Argument

“

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing the questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement)

”
Does Active Network violate E2E argument?
E2E is more about which layer to implement a function, not which node
Previously,
app/transport layer = end hosts
network layer = routers
Distinction is not as clear with active networks
How to choose end-point?

The end-point is a trustworthy entity.
In Active Network, we should trust our own code, regardless of where it is executed.
Implementation and Performance
ANTS: Active Network Implementation from MIT
Java based Implementation
Code are hashed using one-way function (MD5)
Capsule include a 128bit hash that identifies which code to run
Security Implications:

can’t change the code (hash will be inconsistent)

can’t guess the hash without knowing the code
Code needs to be signed and certified by a trusted authority, then posted online for others to use
Code can be cached each node. If a code needed by a capsule is not available, ask from the node upstream (where the capsule came from)
To bootstrap the process, the code is install in the “local” active node (e.g. NUS gateway)
Code size is limited to 16KB to avoid distributing large amount of code
Can process up to 1.5Mbps (T1 link)
100Mbps possible with in-kernel, native implementation (but less protection)
Historical Perspective
Initiated a flurry of research activities and debates between 96-00
Main Criticisms

“Killer App”? Performance + Security?
An example of research that involves:

OS, PL, Networking, Security, DS
Still relevant today:

\(^{\text{(somewhat?)}}\)

network no longer just forward packets
NAT
WAN accelerator
SIP gateway
NAT
WAN accelerator
SIP gateway
Also relevant in wireless sensor networks for deploying new services onto sensor fields
PlanetLab is a global research network that supports the development of new network services. Since the beginning of 2003, more than 1,000 researchers at top academic institutions and industrial research labs have used PlanetLab to develop new technologies for distributed storage, network mapping, peer-to-peer systems, distributed hash tables, and query processing.

PlanetLab currently consists of 825 nodes at 406 sites.