Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.892 Fall 1999 **Web Caching** December 1, 1999

Overview. Active networks. Active services.

1 Active networks

- Traditional networks route packets looking only at destination (and maybe source fields).
- Problem: rate of deployment of new protocols and apps is too slow.
- In active networks, nodes (routers) receive packets, *perform computation on them based on their internal state and control information carried in packet,* and forward zero or more packets to end points depending on the result of the computation.
- Hence, users and apps can control the behavior of the routers on their path and can end up with network services richer than those easily provided by the simple IP service model. At least, that's the promise!
- Caveat: the term "active networks" means a different things to different people. When we say "active networks," we refer to the above definition, with the computation occurring at the network (IP) layer of the protocol stack. It is this architectural decision of several active networks work that has caused the most debate and and argument in the community.
- But why bother? Isn't plain old IP good enough and all else happens end-to-end anyway? As it turns out, not! Consider these (potential) applications that do in fact customize data flow inside the network.
  - Firewalls: System administrators inject specialized policy programs for packet filtering.
  - Web proxies and caches: To make dynamically generated web pages cachable, servers could ship the functions that generate the HTML pages from the server to the web cache, who could generate the HTML page on demand. Also, could dynamic load balancing of web caches, moving them around the network to find the best location.
  - Transcoding services.
  - Nomadic routers: These are modules who adapt to the means by which a client connects to his/her environment. For example, do caching when disconnected, encryption when away from home, etc.
  - Transport Gateways: Snoop agents
  - Reliable multicast: Do Ack/Nack supression and retransmissions inside the network instead of at the endpoints.
  - Online auctions: Imagine an online auction, with a server receiving bids from a large number of clients. To reduce load on servers, servers could place filters in the network that look for low bids and automatically notify the client that the bid is too low.

  - Sensor data mixing and fusion: Geographically dispersed sensors are spread throughout the network. Active nodes, under instruction from clients, aggregate a subset of this sensor data together and present the formatted data to the client.

- Active networks aims to make applications like the above easy to develop and *deploy*. (As an aside, the original active networks papers claimed that active networks *enable* radical new applications. I disagree. And in fact, the paper in the readings corrects this misconception and states that active networks enable network services to be conveniently introduced, but anything that can be done with it can also be done without it.)

- Approaches:

  1. Single-node programmability. An example of this is programmable switches that separate code loading from execution. E.g., router plugins, switchware, active bridges.
  2. Control-path programmability.
  3. Capsules: Code sent along with packet.

- Research questions:

  - Execution environments: what sort of environment? How many on a router? I.e., service model definition.
  - Safety and security: programmable data raises serious security concerns.
  - Resource sharing: need to perform OS functions.
  - Performance: too much code will kill network performance.
  - Applications: what can we do easily that we can't do today?
  - Evaluation: how to demonstrate or prove that active networks are a good idea.

1.1 Case study: MIT ANTS system

- In conventional networks, all routers perform the same computation; in active network, routers have the same runtime system.

- Tradeoffs between functionality, performance and security.

- Three key components of system

  1. Capsules
  2. Active nodes execute capsules of protocol and maintain protocol state. Nodes provide capsule execution API and safety using OS and language techniques.
  3. Code distribution mechanism to ensure that capsule processing routines are automatically and dynamically transferred to nodes as needed.

- Capsules that share the same processing code are said to be a protocol. They may share state in the network. Capsule type (ID) is MD5 of code.

- What functions does the capsule forwarding (node) API need to export?

  1. Environment access (querying node location, routing tables etc.)

2. Capsule manipulation to access header and payload.
3. Control operations: create other capsules, forward them, suppress current capsule etc.
4. Caching: soft-state cach of app-defined objects.

- Safety provided by mobile code technology (e.g., Java).
- Node OS monitors capsule resource consumption.
- If node doesn't have capsule code, retrieve from previous hop on path so far. (Reminiscent of RSVP in that previous hop needs to be kept track of.)

1.2 Some comments

- Active nodes are a good idea, there are numerous applications and a coherent architecture is desirable.
- Key question: is all this necessary at the forwarding level of the network? A danger of messing extensively with the basic IP unicast forwarding services is that one runs into the real dangler of building a network on a fragile foundation rather than on a simple one with a well-defined service model.
- How to do things across many flows in the capsule model? (Hard, if not impossible.)
- In the ANTS design/implementation, long-lived services hard to build.
- Shouldn't we be exploring application-level alternatives for rapid deployment of new applications?

2 Active services

- Contribution of paper: defines a framework for active service tailored to real-time media gateways (video and audio). Prototype system called AS1.
- Think of this as one way of doing "application-level active networking" for a specific domain.
- Service agents *("servents"* perform operations like transcoding and multicast-to-unicast conversion in a *cluster environment.*
- Components of architecture:

  1. Service environment: defines programming model and execution environment.
  2. Service location: How to locate AS1 cluster.
  3. Service management: Resource allocation in cluster.
  4. Service control: How can client control servent?
  5. Service attachment: How to attach to cluster if client doesn't have multicast?
  6. Service composition: (Not explored in this work).

- Service environment: Environment at application layer and uses MASH infrastructure. Claim is that a good environment is domain-specific as are the APIs. Safety not addressed (can leverage work on type-safe languages etc.)

- Service location: How to locate cluster? One possibility is to use mechanism like DHCP (or even DNS). Other possibility is to listen on well-known multicast address for advertisements. Q: what are the advantages/disadvantages of this approach?

- Service management: This is the bulk of the paper.

  - Servent launching (ASCP): Uses announce/listen protocol model and no centralized manager. Periodically announce set of $(key, value)$ pairs on multicast group. Receivers either update, refresh or age entries out of table. Error recovery subsumed as part of normal operation. Host managers, one per cluster machine, use this to launch servents. Use multicast damping to prevent servent floods.
  - Processor sharing: Key idea is the use of a *birth-death* process to grow and wean number of cluster machines. If $n < N$, randomly fork copy on an idle machine with probability $p_l = min(1, N/n - 1)$. And if $n > N$, kill with probability $1 - N/n$. Property that no HMs exist can be made exponentially small in the target number. (Here, N is the desired target number of machines and n is the number of actual machines in current operation.)

- Service control: Application-specific control. In the MeGa architecture, uses SCUBA (Scalable ConsensUs-based Bandwidth Allocation) to control servents.

- Service attachment: uses "soft-state gateways" to bridge multicast-to-unicast regions.

- Paper describes deployment experience on Berkeley NOW.

2.1 Comments

- Paper describes a nice framework for active services in the context of media gateways and real-time MBone applications.

- If most active networks work is very general, this is very specific to a restricted class of applications (but a very real class).

- Use of soft-state announce-listen protocols is a nice departure from "standard" centralized managers. But I have some concerns about rapid fluctuations in servent or HM population sizes.

3 Summary

These two papers describe two different ways of building more flexible networks. One of them changes the way IP forwarding is done; the other works at the application-layer. One is general, the other domain-specific.