Novel Systems

Active Networks
Denali

Extensible Networks

• Observations
  – Creating/disseminating standards hard
    • Prototyping/research
    • Incremental deployment
  – Computation may be cheap compared to communication
• In late 90s, many projects in this area
  – Active Networks
    • ANTS
    • Active Names
    • Software Routers
IP Network

- Two types of entities: Network and End Host
- Single component designed to be a building block
  - A few standard higher level services
  - Service model fixed over “long” period
- Has been good to us, but
  - Want services not easily built on this block
    - Multicast, intelligent dropping of video frames, etc
  - Want a common base wider than this block
    - CDN’s, content transformers, etc.
  - Don’t want to be stuck with a new fixed model for next 20 years

Extensible systems and Deployment Architectures

- Extensible Systems
  - Allows a system to be modified/extended at runtime
    - SPIN
    - Active Networks
- Deployment Architectures
  - Allows new functionality and new systems to be deployed easily
    - Microkernels
    - Denali
- Upgradeable systems
  - Software routers
Outline

Overview
Extensible Systems
Deployment Architectures
Conclusions

Extensible systems

• Specific service model
  – Extensions flesh out the model
  – Dividing line between architecture and service often blurry
• SPIN
  – Can push extensions into kernel code
• Ants
  – Can push extensions into packet handling code

Have rarely seen in actual deployment
Extensibility: ANTS Behavior

• Upgrade system in-place
  – ANTS inserts logic into the middle of packet handling
  – Active Names composes extensions into a pipeline
  – In either case, moves interior upgrade decision to the ends

• ANTS execution model
  – Receive a packet
  – Retrieve code based on a pointer within the packet
  – Executes code with packet as argument
    • Read/write node soft state and look at packet data
    • Create any desired outgoing packets
    • Queue outgoing packets on desired queues

Do people REALLY want rapid and incremental extensibility?

• Creation
  – Makes it hard to ensure system correctness
    • Adding a new protocol with soft state demands may put me over my state limit, causing my multicast session which otherwise worked well to now fail frequently

• Deployment
  – ISP’s and backbones don’t want even all IETF services
    • They may not make sense within their business model

• Predictability
  – Hard for individuals to customize setup if basic service behavior is unknown
    • How do we filter traffic if packet addresses are service specific
Extensibility: Correctness

- Correctness is hard in extensible systems
  - May run in configurations not previously tested
    * Extensions must execute in all versions of VM’s in CISCO, Juniper, Foundry, Linux, and Windows routers
    * Extensions must tolerate variable loss of “soft” state
    * Multicast packet may rely on properties of unicast routing that don’t apply in certain networks
    * In Active Names, extensions must work when ordered arbitrarily with previously unknown extensions

- Broken protocols must be fixed at end host rather than in the router: is this an improvement or a step back?
  * Much of current protocols already in the end host

- Positively, decreases need for interaction between separate implementations of protocol components

Extensibility: Security

- Challenges
  - Reasonable resource consumption bounds challenging
    * How much CPU is ok?
    * How many packets?
    * How much storage?
    * How do we deal with changes in these over time?
  - Can we deal with multi-node security issues?
    * Network bandwidth, loop-free routing
  - Can we reason about observed behavior caused by previously unknown code?

- Advantages
  - Domain specific code, allowing tighter behavior restrictions
How do we achieve security?

• Set resource limits
  – Number or identity of hops for a packet
  – Number of packets created per node
  – Processing time bound
  – **Require services to function if a default forwarding pattern is used**
• Provide security classes with different combinations of restrictions
• Recognize specific bad behavior
  – Disallow repeated receipt of same packet
• **Require trusted review**
  – IETF signs all code

What has this done to ease change?

• Deploying services is now hard because it’s
  – Challenging to get the capability into the hardware
  – Challenging to convince people to turn the capability on
    • Worry about technical problems of upgrade
    • Worry about business model of new services
• “Traditional” Active Networks addresses this as a technical issue.
  – The second challenge seems to be more social than technical
Outline

Overview
Extensible Systems
**Deployment Architectures**
Conclusions

**Deployment Architectures**

- The real action in the Internet has been the addition of infrastructure services
  - Let’s concentrate on improving this experience
- Provide platform on which to deploy a system
  - May or may not have a fixed purpose in mind
  - Provide commonly useful helper services
A picture of the Internet

A REAL picture of the Internet
Provide an abstraction other than “host” for services

• For all the talk of upgrading the network, augmenting the network is often easier
• Provide base utilities we services often use
  – Network communication
  – Load balancing
  – Image migration

Example system: Denali

• Support deployment of service infrastructure
  – Overlay/application level service deployment has been effective (multicast, CDN).
    • BUT: large cost (time and money) to roll out a network
    • SO: Let users roll out services onto standardized nodes
  – Server deployment currently scales down poorly
    • Effective web hosting model of co-located servers doesn’t work well for small traffic sites
    • Let users share nodes
Security

- Problem
  - Large poorly understood chunks of software with lots of resources
  - Actively encourages widespread deployment of code with minimal overhead
- Solutions
  - Use access control/monitoring/retribution
  - Use service modeling to limit behavior
    - Web server doesn’t contact random clients
    - Email server only opens port 25
  - Give up on service deployment and provide a service with interesting knobs

Deployment

- A single provider can roll out it out
  - Rollout at any scale
    - 1 machine or 1000 machines are useful
    - Supplement or replace existing infrastructure
- Some services (such as CDN’s) MAY require placement in hard to reach locations
  - Easy to tell ISP “install this box and it’ll cache content”
  - Harder to tell ISP “install this box that will support whatever our users want to run on it”
- Is standardization on a new platform important?
  - Much success without it
  - New OS designs have always been hard to move from research to industry
What has happened?

• Active networks has not taken off in production
  – Hard tradeoff between utility/security
  – Incremental deployment hard
  – Forces significant change in how we write, use, design, and manage networks
  – Most non-research gains seem obtainable using software routers and deployment policy
    • Decreases security and correctness issues substantially

• Concentration on endpoint tricks
  – Successful? for wide range of services
  – Deployed usefully by individual parties
  – Basic model of the Internet is augmented, not changed

Research Approaches

• These two lines of research seem to represent two basically different approaches:
  – X isn’t currently done for various reasons. Can we produce a system that changes things enough to let us do X
  – Y is done all the time in slightly different ways. Can we produce a system that looks at Y and lets us do it more easily, powerfully, etc

• Is there any take-home message about the strengths/importance of these approaches?
Conclusions

• Basic problem is interesting
  – Importance of incremental upgradeability is unclear
• Should consider stakeholder interests
  – A flexible system may be harder for owners to control
  – Reducing the number or diversity of directly participating stakeholders helps immensely
• Incremental deployment is challenging even for systems designed to ease it in the future
  – Safest to choose systems that deploy usefully at many scales