Composable Active Network Elements: Lessons Learned

Ellen Zegura, Georgia Tech
Ken Calvert, U. of Kentucky

ANETS PI Meeting, 25 May 2000

www.cc.gatech.edu/projects/canes/

The Cast

- Bobby Bhattacharjee (GT, now UMd)
- Ken Calvert (UK)
- Youngsu Chae (GT)
- David Haynes (GT, now Motorola)
- Richard Liston (GT)
- Shashidhar Merugu (GT)
- Matt Sanders (GT)
- Billy Mullins (UK)
- Srinivasan Venkatramen (UK)
- Ellen Zegura (GT)

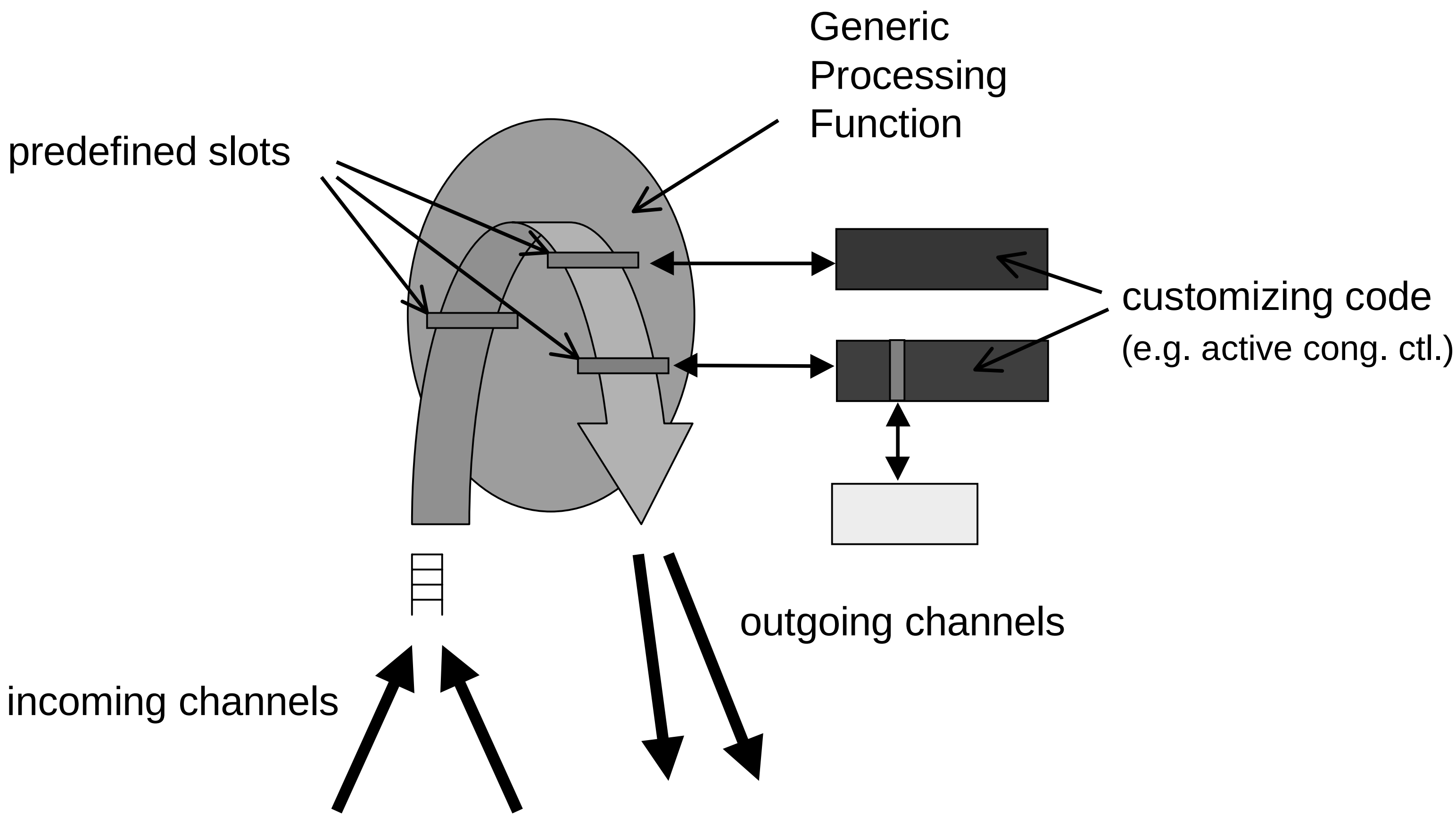
History

- Fall 1995: Ellen heard Dave Sincoskie give talk about active networking
- Ellen said “This looks cool.”
- Ken said “But what is it good for?”
- CANEs began…. (w/Bobby Bhattacharjee)
  - active applications (e.g., congestion control)
  - platform offering middle ground between flexibility and performance

CANEs Project Goals

- Focus on benefit of bringing together application information and network information
  - not…rapid deployment of new protocols
- Offer constrained programmability and modularity via “primitive elements” + composition paradigm
  - not…programming language based, with virtual machine at every node
- Fast forwarding path for vanilla traffic
- Explore compositional formal reasoning

CANEs EE Model



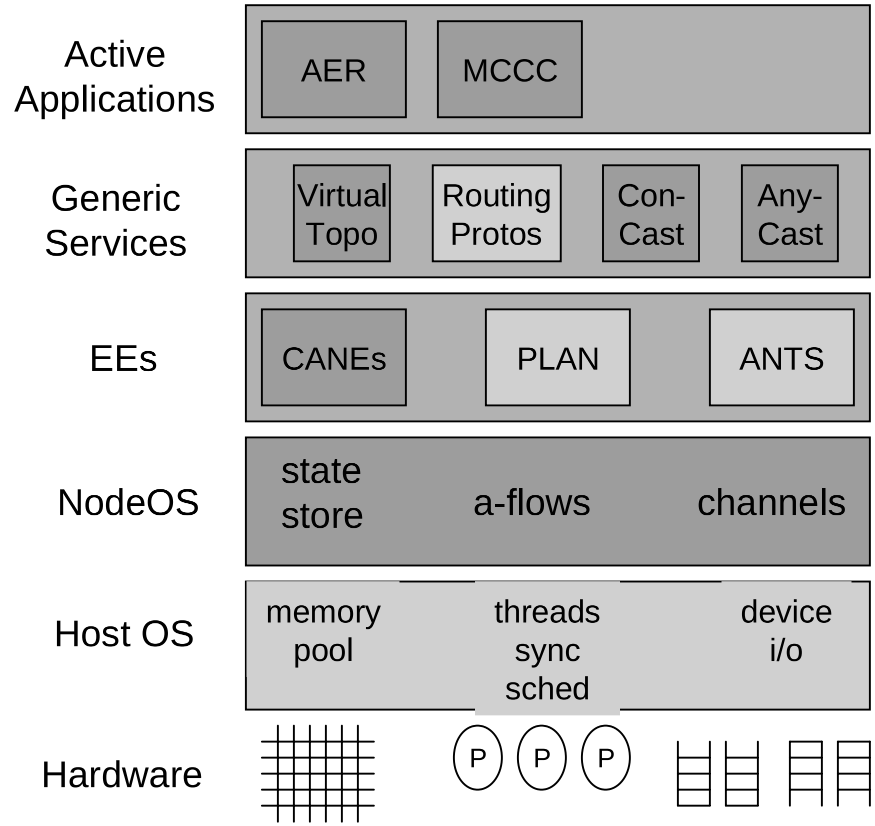
CANEs User Interface

- User specifies underlying program and set of injected programs per packet type, conveyed by signaling
- Underlying program
  - skeleton/default packet processing (e.g.,generic fwding)
  - contains slots that identify locations in code
  - slot is raised when location is reached in control flow
- Injected programs
  - code to customize skeleton (e.g., select routing table)
  - one or more injected programs per slot
  - programs in a slot execute concurrently when slot is raised

Bowman+CANEs

Extensible implementation of the node architecture

- EE: CANEs
- Node OS: Bowman
  - a-flows
  - channels
  - state store
- Bowman extensions
- Miscellaneous other components
  - packet classifier
  - topology construction



Project Accomplishments I

- Platform:
  - CANEs EE (released Nov 1999)
  - Bowman NodeOS (released Nov 1999) [Infocom’00]
- Applications:
  - 1st active application(?): Application-specific congestion control [GT-96-02, HPN’97]
  - Network-aware caching [Infocom’98]
  - Programmable network query and synthesis to support topology-sensitive applications [OpenArch’00]
  - Reliable multicast (w/TASC and UMass)

Project Accomplishments II

- Active network simulator
- Documents:
  - Node Architecture (Calvert)
  - Composable Services (Zegura)
- Team 4 involvement

Active Congestion Control

Observations:

- Application knows how to adapt to congestion
  - Which packets to drop, according to data and history
- Network nodes know when to adapt
  - Which nodes are congested, and when

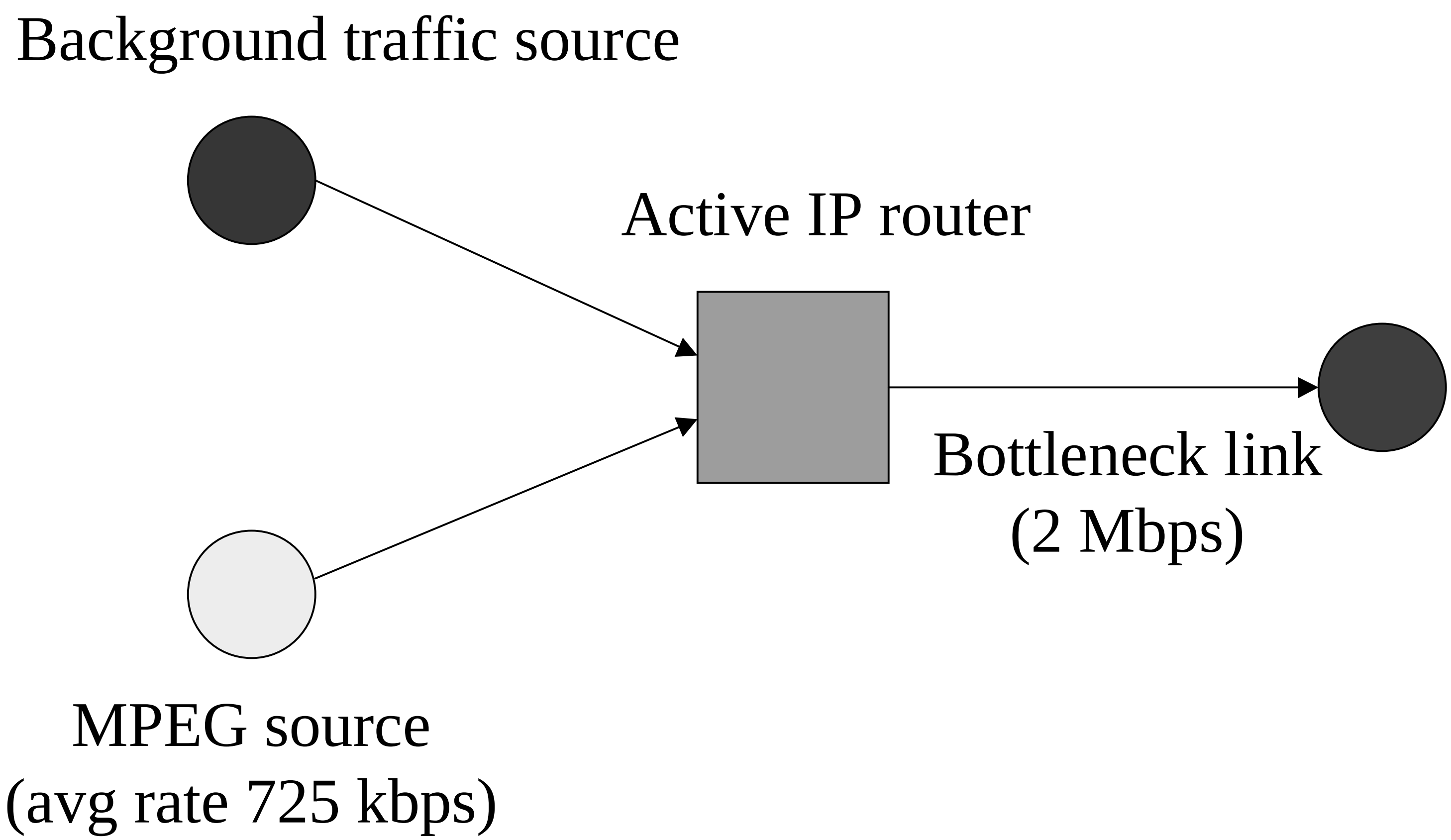
⇒ Bring these bits of knowledge together!

  - Application provides “advice” regarding discard
  - Node notifies end-system of congestion

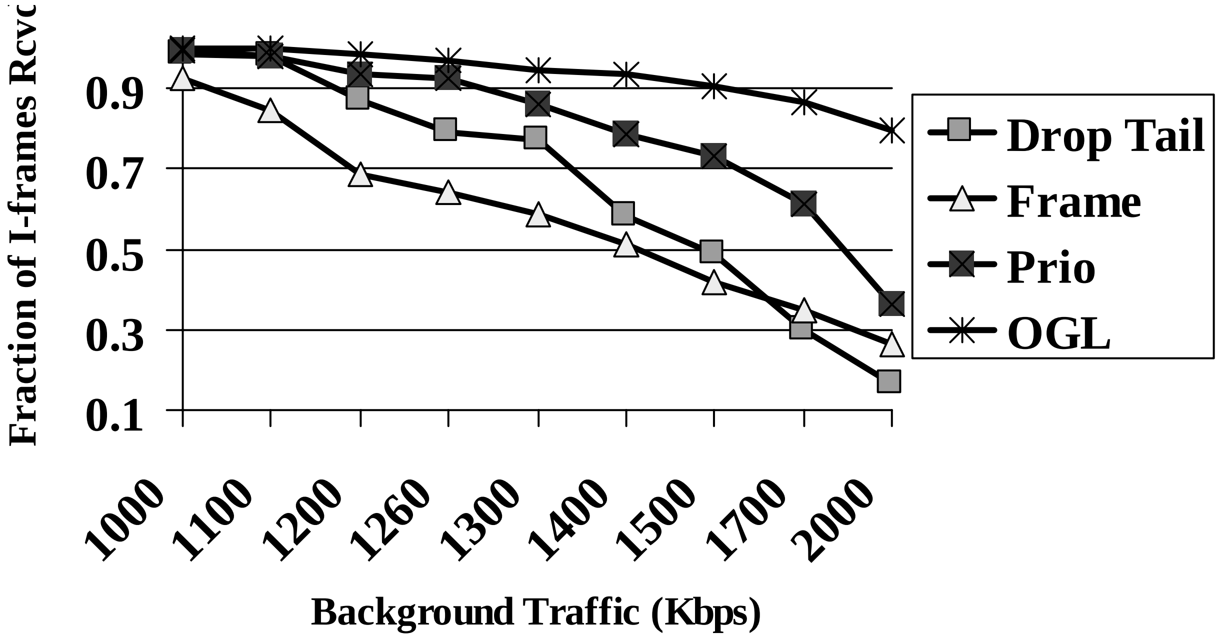
Intelligent Discard for MPEG

- Principle: P, B frames depend on I frames
- Discard approaches:
  - Discard application-layer units (e.g. Frames, GOPs)
  - Static priorities (e.g., I frame higher than P, B)
  - Drop P, B if corresponding I already dropped
  - Evict P, B from queue to make room for I
- Experimental method: active IP option
- Evaluation metrics:
  - Application-layer quality (e.G., SNR, I-frames received)
  - Network impact (e.g., Received bytes discarded)

Experiment Configuration

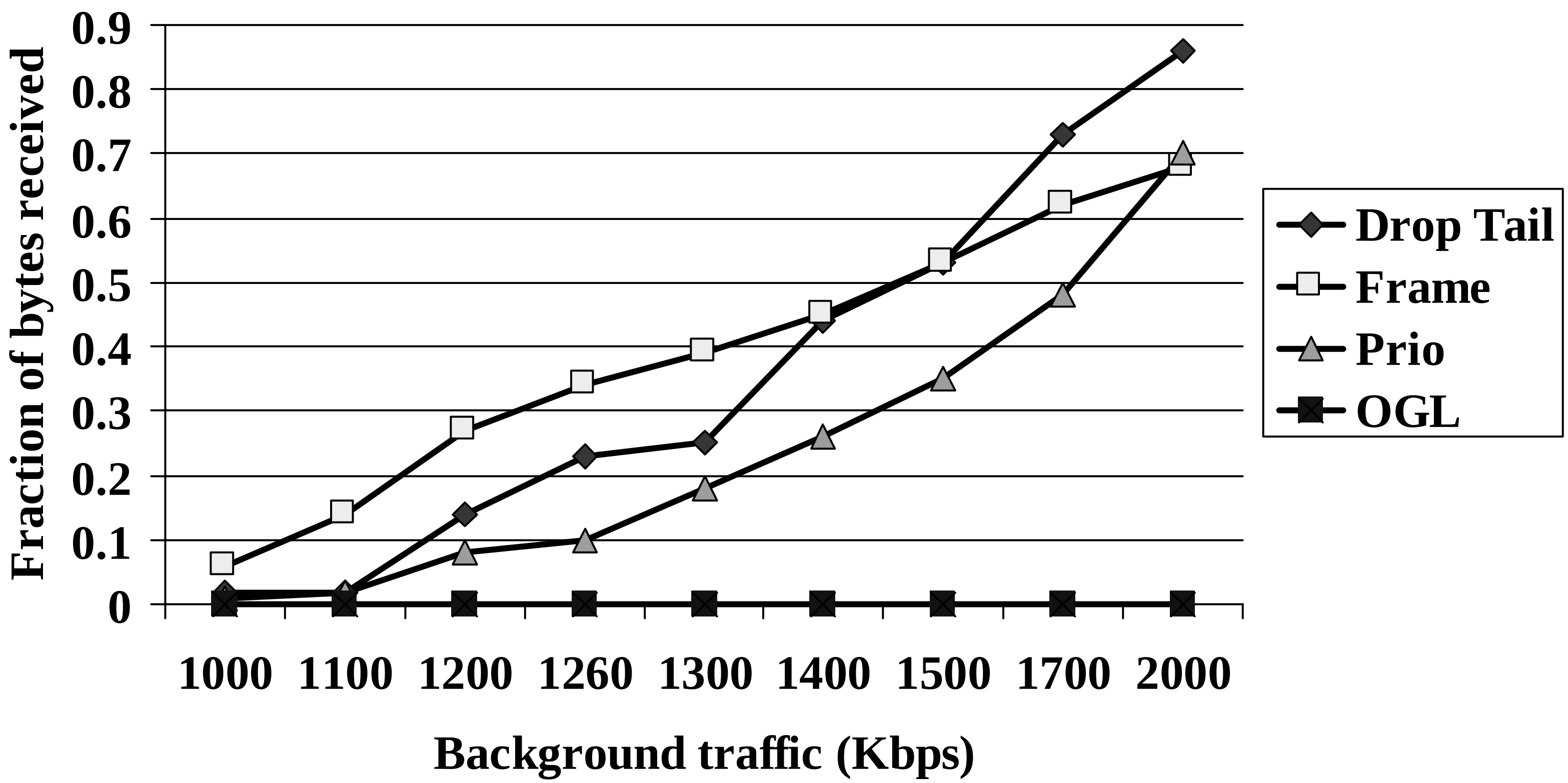


Result: I-frames Received

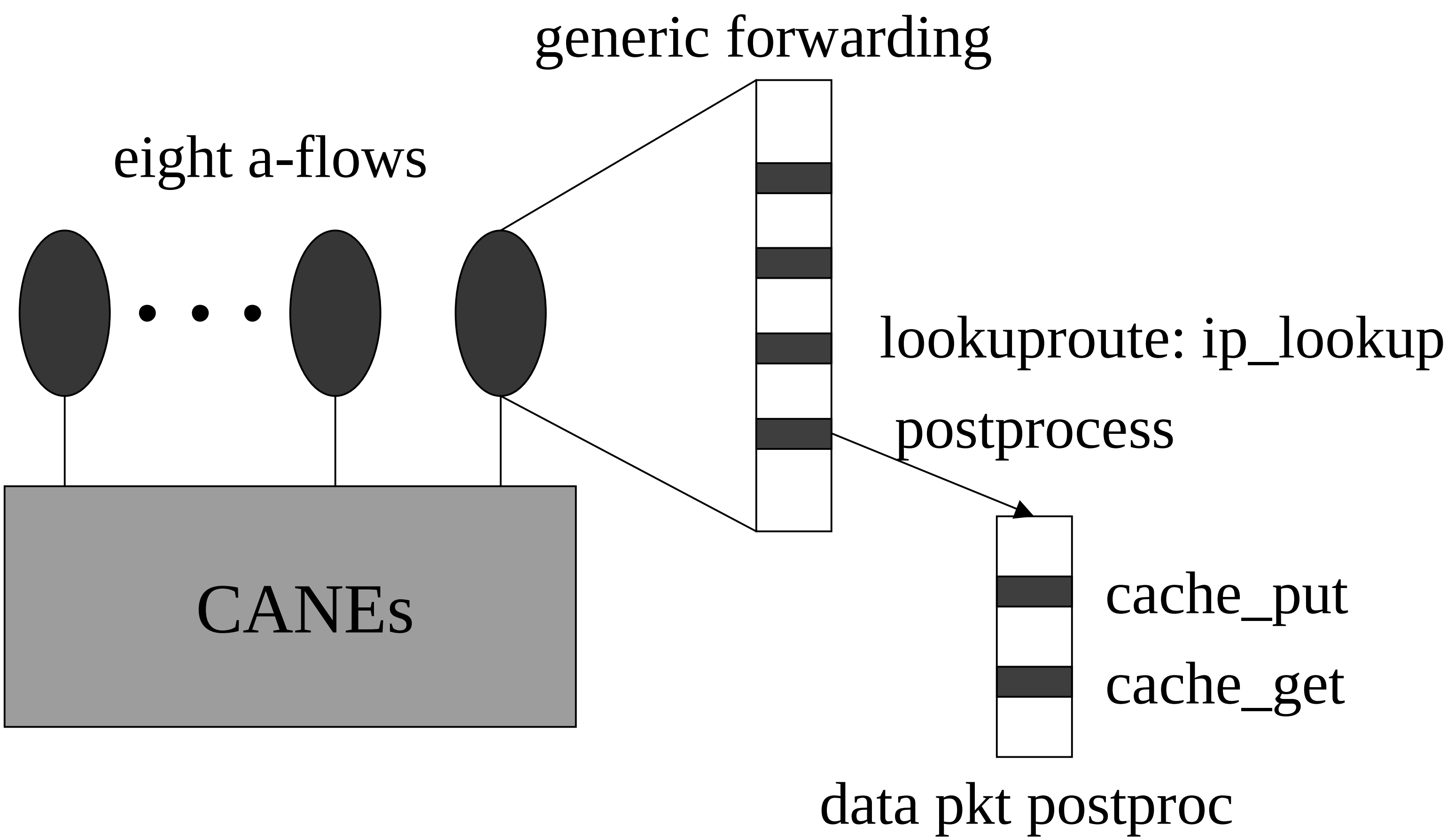


One active router, bottleneck 2Mbps, MPEG source averages 725 Kbps

Result: Data Discarded at Receiver



Highlight: Reliable Multicast (I)



Highlight: Reliable Mcast (II)

- Eight a-flows, one per packet type
- One underlying program, 21 total injected programs including four user-defined
- Lots of timer-driven activity, led to change in timer support
- Relatively easy interoperability with non-active video endsystem application

Project Introspection

Things Done and Not Done

- Things we didn't plan to do, but did:
  - Build a NodeOS (or part of one)
  - Define languages (topology specification, filter specification, signaling)
  - Participate heavily in demonstration team
- Things we planned to do, but didn't:
  - Implement other applications/services
  - Create wide-area CANEs testbed/ABONE

Lessons Learned

- Programmatic
  - Difficulty of parallel development of layers
  - Value of (forced!) integration with other projects
  - Value of full time staff (to echo JMS)
  - Challenge of distributed collaboration (and kids!)
- Technical
  - Language design is unavoidable
  - Importance of timer-driven processing
  - Importance of naming (topologies, reusable configurations of underlying+injected programs)

Mistakes?

- Choosing C over Java
- Insufficient resources to go from prototype to version usable by larger community (and do other things)

CANEs: Status

- Porting CANEs to Utah-flavored NodeOS
  - EE developers toolkit
  - CANES+Bowman → CANEs´+EEtoolkit+energy
- Incorporating Seraphim for Fall demos
- Implementing ActiveCast services