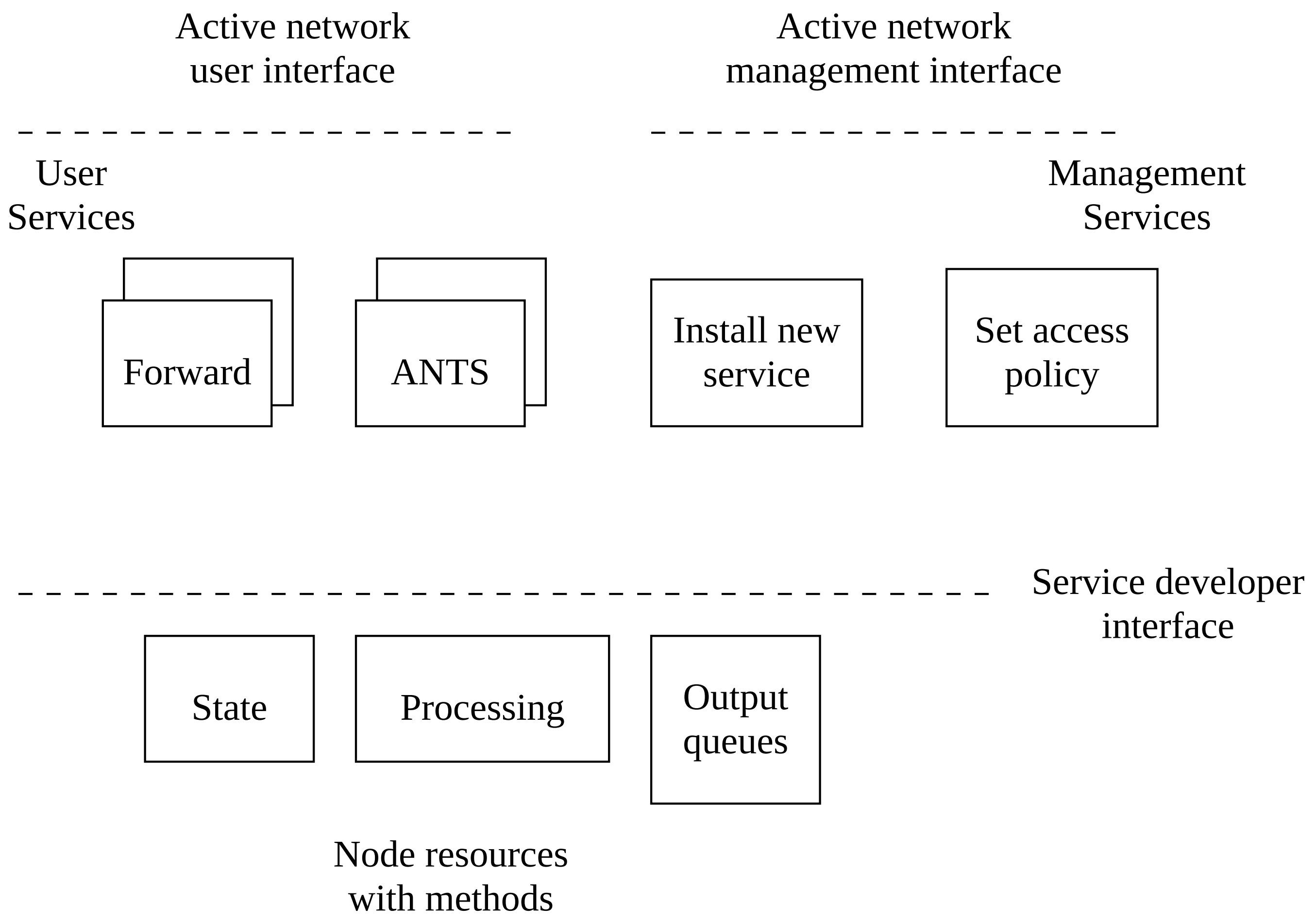
# CANEs Architecture

**Samrat Bhattacharjee  Ken Calvert  Ellen Zegura**

Networking and Telecommunications Group
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA.
`http://www.cc.gatech.edu/projects/canes`

# World View


Active network
user interface
Active network
management interface
User
Services
Management
Services
Forward
ANTS
Install new
service
Set access
policy
Service developer
interface
State
Processing
Output
queues
Node resources
with methods

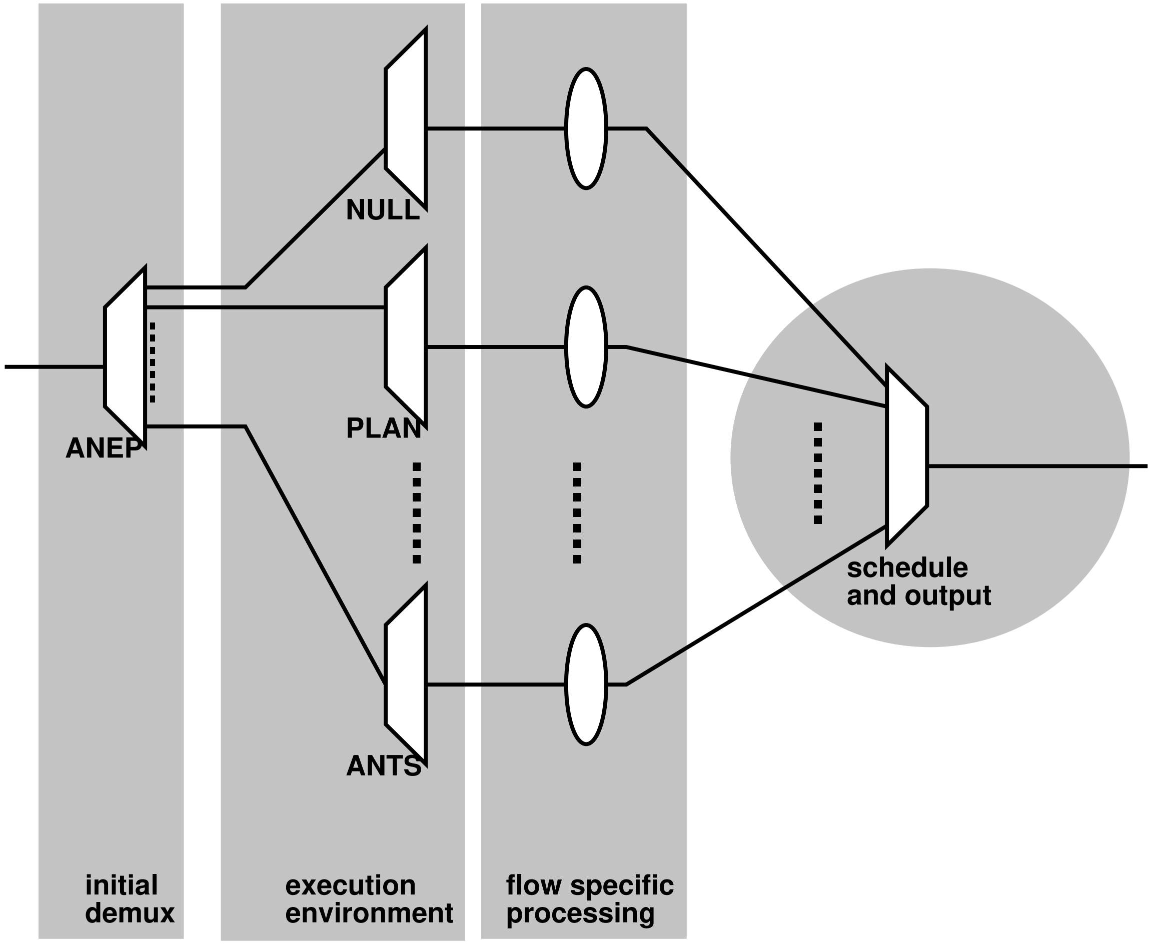
# Services

- Forwarding
  - Unicast/small multicast
  - Large multicast
  - Anycast
- Environment
  - ANTS
  - PLAN
- State maintenance
  - Modify state
  - Remove state

# Customization Methods

- Service selection
- (Simple) service parameterization
- Trigger-based service parameterization
- Installation of new services
- Installation of new environment

# Generic Packet Processing



## Requirements for Software Structure of AN node

- Be able to assimilate autonomous code objects
- Provide communication mechanism
- Allow composition of autonomous code
- Possible Solutions:
  - Traditional IPC — pipes, shared memory
  - Netscape Plug-in
  - Software Bus

## Possible Approach — Event based solution

- Code Objects implements *specific* functionality and..
- May **bind** to and **raise** generic events
- Exports **handlers** for events
- Thus, events will provide an anonymous publish-subscribe interface
- In general, code objects not aware of other objects installed/active
- Structure the active node as a synchronous reactive system

## Event Structure — Synchronous Events (Syncents)

- Generic Event structure
- Event handlers bind to named events
- Generic event Interface
- Example: Code object supplies handler :
  `bind (OUTPUT-QUEUE-FULL, congestion-control)`
  Node will raise event : `raise (OUTPUT-QUEUE-FULL, current-node-state)`
- Requires event registry — dynamic event list
- Arbitrary number of handlers for each event