Active Reliable Multicast on CANEs: A Case Study

Matt Sanders and Ellen Zegura **Georgia Tech.**

Ken Calvert **University of Kentucky**

Mark Keaton and Steve Zabele **TASC Inc.**

Bobby Bhattacharjee **University of Maryland**

this collaboration was supported by the DARPA ActiveNets program for the purpose of integrated project demonstrations

Two Projects:

CANEs

- Composable Active Network Elements
- GT and UKY
- CANEs Execution Environment (EE)
- Bowman Node Operating System (NodeOS)
- platform for active application composition

PANAMA

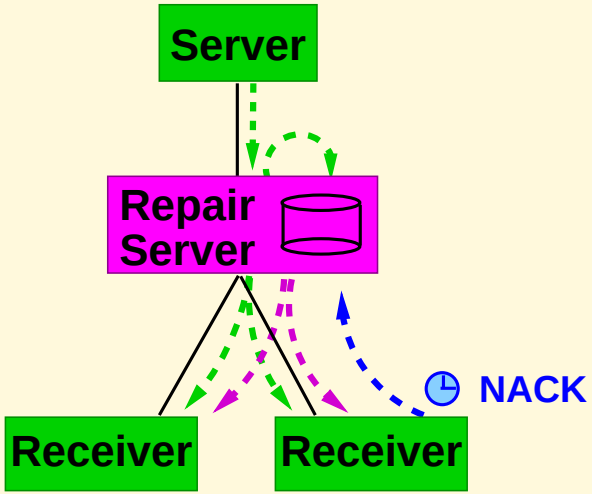
- Protocols for Active Networking with Adaptive Multicast Applications
- TASC and UMass
- Active Error Recovery protocol (AER)
- Nominee-based Congestion Avoidance protocol (NCA)
- active reliable multicast with congestion avoidance

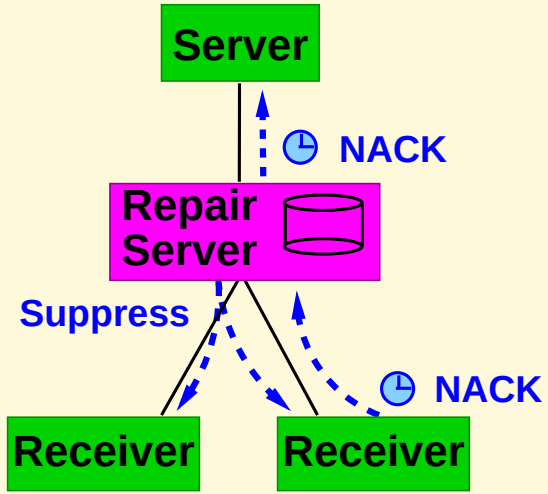
Outline

- AER/NCA protocol overview
- CANEs platform overview
- AER/NCA protocol decomposition
- operational experience
- four lessons learned
- conclusions

AER/NCA Protocol Overview

- end-to-end reliable multicast protocol
- can use repair servers between endpoints

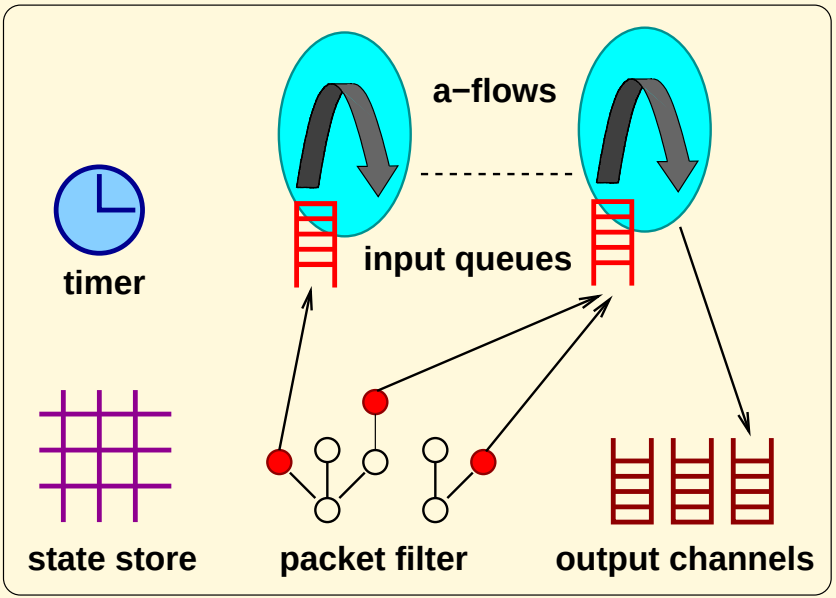




- supports semi and full reliability modes

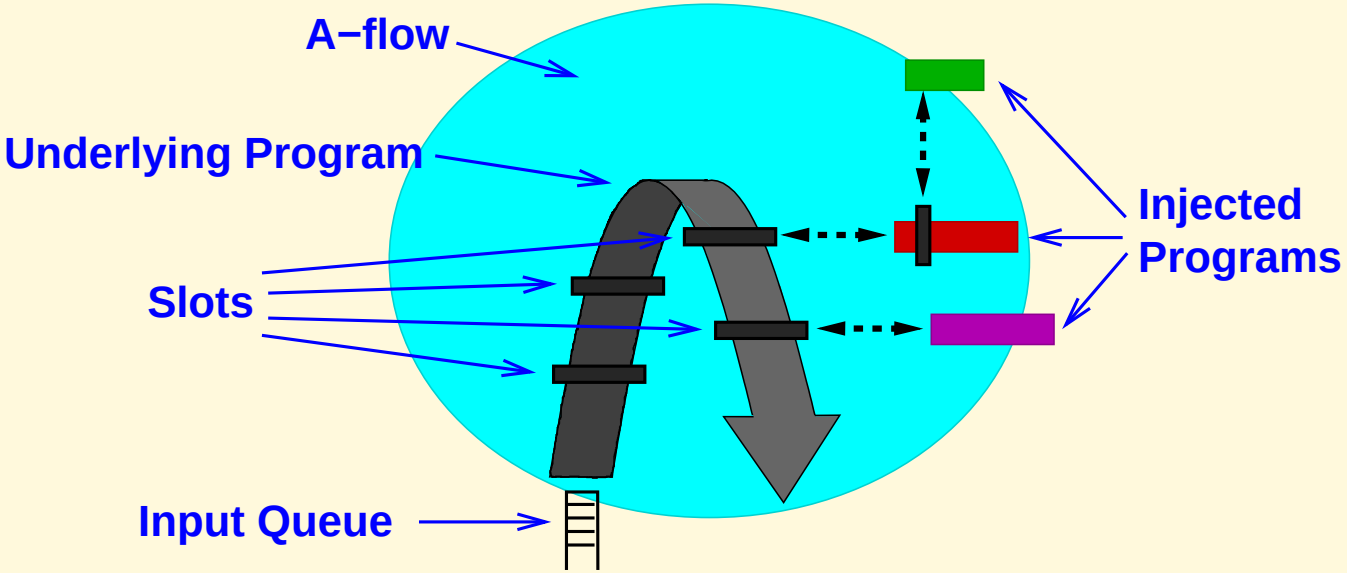
CANEs Platform Overview

- primary abstractions: a-flows, channels, state store
- also provides: timers, packet classifier, and extension



CANEs Environment Overview

- “reasonable” forwarding performance
- modular service construction framework



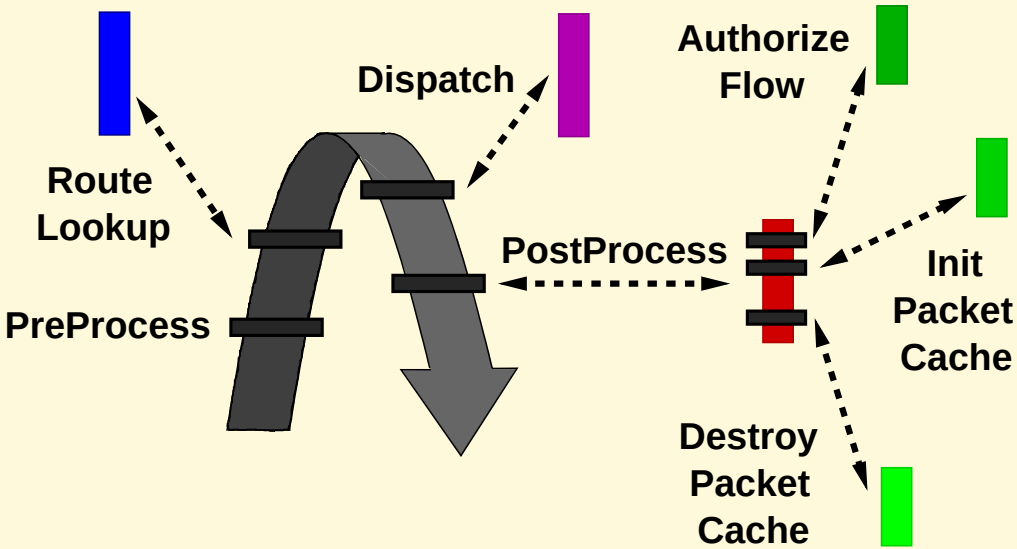
AER/NCA Protocol Decomposition

AER/NCA packet types:

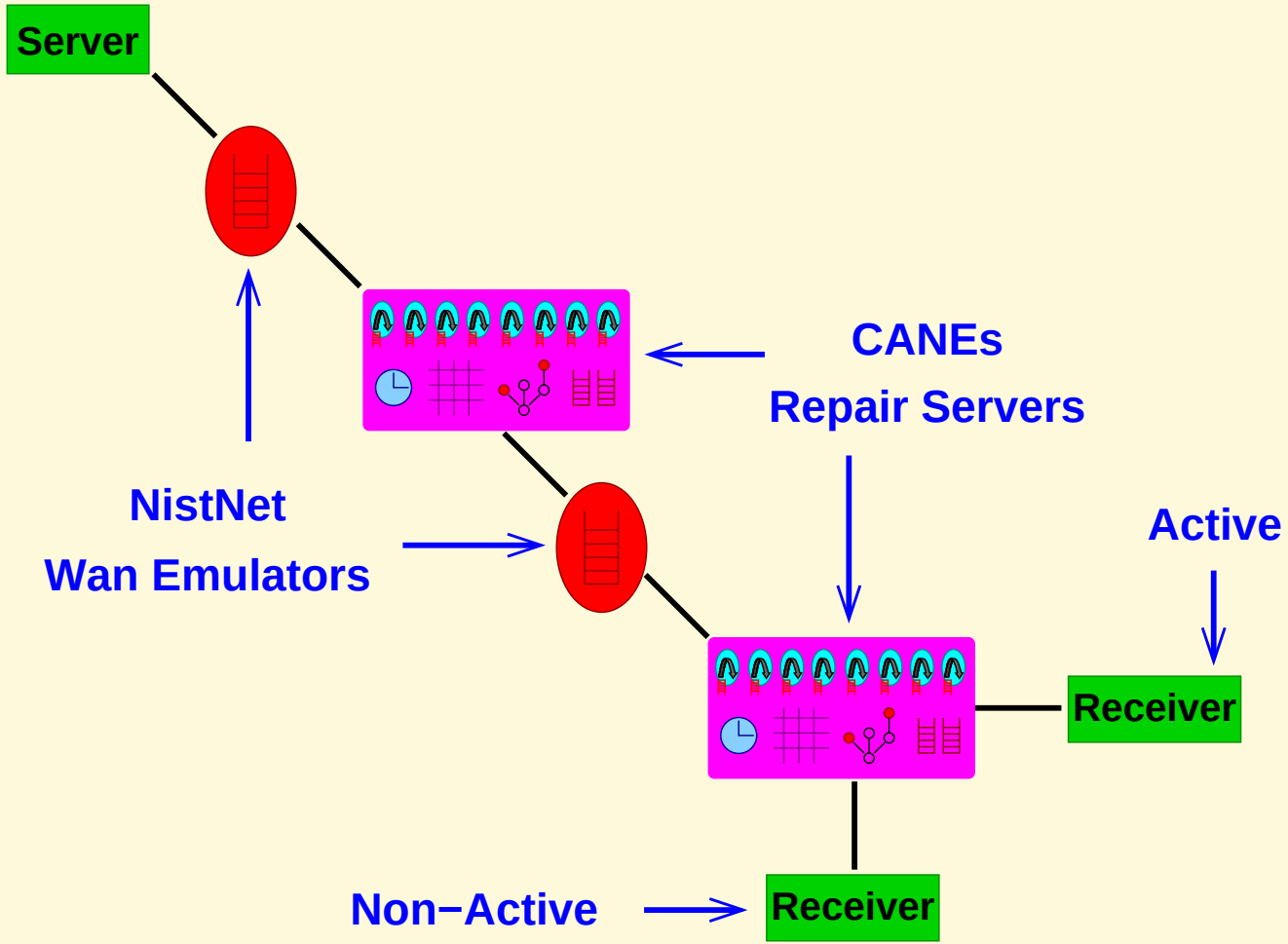
a-flow	Function	Direction
SPM	source path maintenance	down/multi
Data	data forward/cache	down/multi
UNACK	NACK transmission	up/uni
MNACK	NACK suppression	down/multi
RTT	round-trip-time	both/uni
CSM	nominee selection	up/uni
CCM	nominee feedback	up/uni
NPM	nominee path (not used)	up/uni

packet types were defined prior to the collaboration

SPM A-flow Decomposition



Operational Experience



Lesson 1: Timer Handlers

- timers are important for some class of active applications
  example: randomized NACK transmission

timers set

b/t	0.2	0.6	1.0	1.5
0.2	1240	1472	1689	1879
0.6	1240	1472	3332	1834
1.0	1308	1428	1659	1800
1.5	1246	1417	1679	1892

handlers run

b/t	0.2	0.6	1.0	1.5
0.2	141	260	377	464
0.6	134	249	1247	515
1.0	163	234	367	498
1.5	154	229	370	489

- CANEs timer facility modifications
- timer handler capabilities = packet handler capabilities
  example: in CANEs timer handlers should have slot context
- note: timer set and cancel ops are heavily used; therefore they should be very light weight calls
  example: out of order AER/NCA data packets

Lesson 2: Composition with A-flows

- need to preserve the notion of a principal while retaining the benefits of concurrency and low level packet classification
  example: each of the AER/NCA protocols is a separate application
  solution: an a-flow should be modified to be a collection of threads which are either packet arrival or timer event handlers
- decomposition comes at a cost both in contention for state and the work done to find state
  example: contention of NACK state by both the NACK and data a-flows
  solution: decompose with lock and hash granularity in mind

Lesson 3: Underlying Program Evolution

- allowing for a diversity of customization is challenging due to the trade-offs between efficiency and flexibility
  example: user control includes both packet contents and the fate of the packet
  solution: optimized for unicast and multicast without extra cost for either, while allowing each copy of the packet to be modified

Lesson 4: Non-Active Endpoints

- the use of previously non-active operating systems and applications is desirable, but their modification is costly
  example: video server and client nodes were non-active; applications were only active above the UDP layer
  solution: the CANEs platform supports raw Ethernet channels and application packet classification occurs after channel processing; only a route table change was needed on servers and clients

Conclusions

- application designers
  - know the capabilities and costs
  - consider parallelization and composition in protocol design
- execution environment **and/or** nodeOS designers
  - a timer facility is probably necessary
  - timer handlers are first class events
  - timer set and cancel operations need to be light weight
  - define a principal early
  - channel and classification specification should support the full range of non-active end-point OS and application interaction