Ensemble: A Tool for Building Highly Assured Networks

Professor Kenneth P. Birman

Cornell University

http://www.cs.cornell.edu/Info/Projects/Ensemble
http://www.browsebooks.com/Birman/index.html

Ensemble Project Goals

- Provide a powerful and flexible technology for “hardening” distributed applications by introducing security and reliability properties
- Make the technology available to DARPA investigators and the Internet community
- Apply Ensemble to help develop prototype of the ***Highly Assured Network***

Today

- Review recent past for the effort
  - Emphasis was Middleware
  - About 10-15 minutes total
- Then focus on 1997 goals and milestones
  - More attention to security opportunities, standards
  - Shift emphasis to lower levels of network
  - Ensemble “manages” protocol stacks, servers

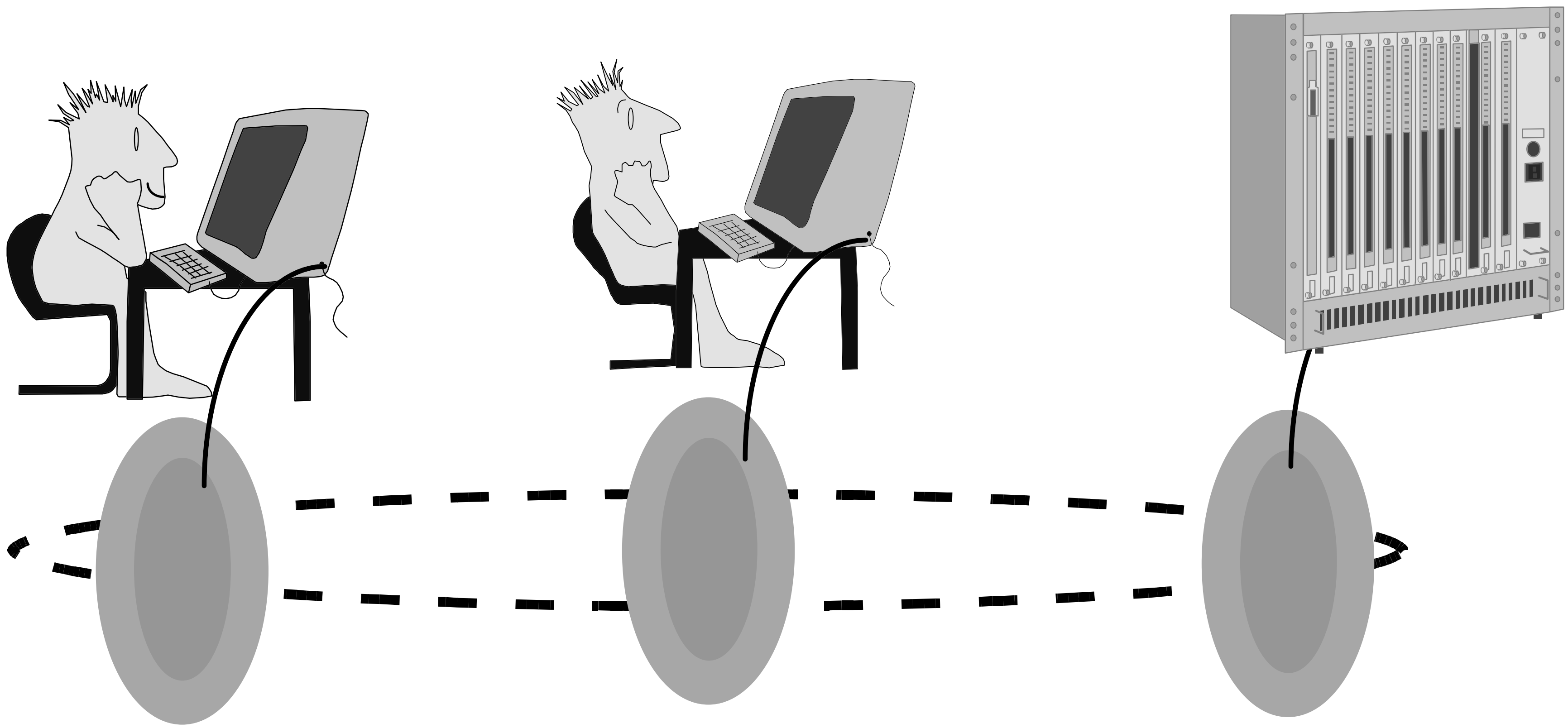
Why Ensemble?

- With the Isis Toolkit and the Horus system, we demonstrated that *virtually synchronous process groups* could be a powerful tool
- But Isis was inflexible, monolithic
- Ensemble is layered and can hide behind various interfaces (C, C++, Java, Tcl/Tk…)
- Ensemble is coded in ML, this facilitates automated code transformations

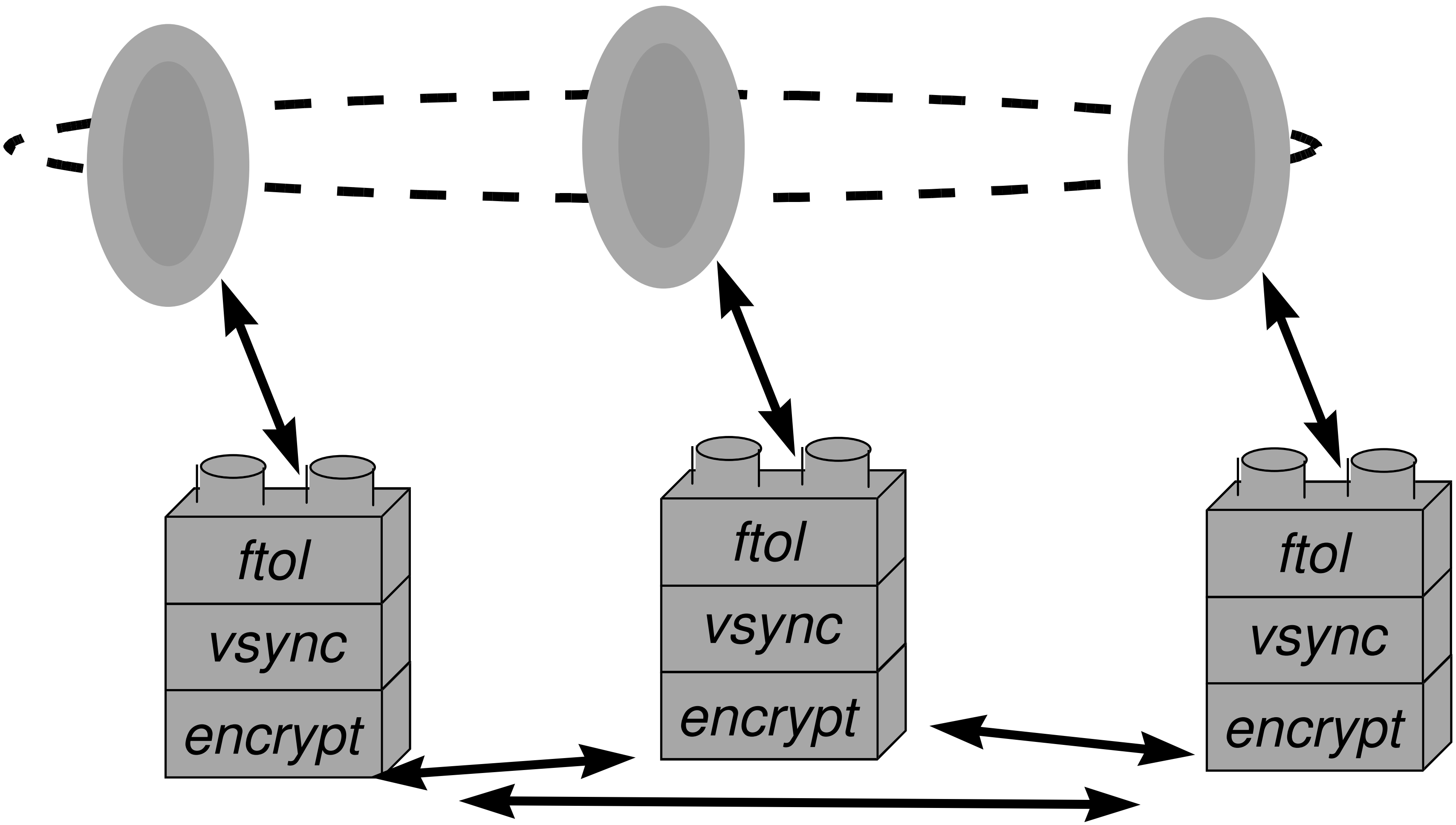
Key Idea in Ensemble: Process Groups

- Processes within network cooperate in groups
- Group tools support group communication (multicast), membership, failure reporting
- Embed beneath interfaces specialized to different uses
  - Cluster-style server management
  - WAN architecture of connected servers
  - Groups of PC clients for “groupware”, CSCW

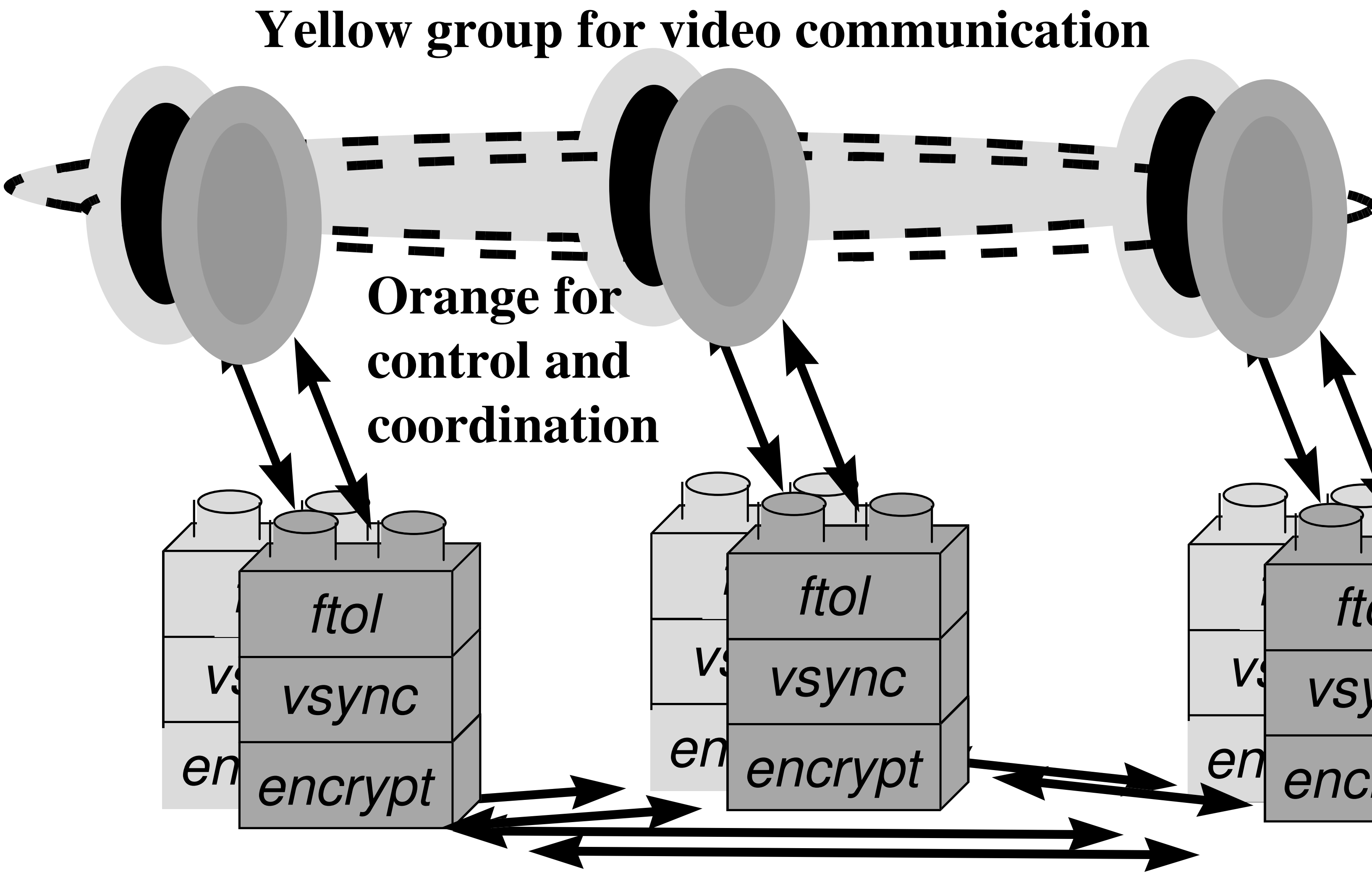
Group Members could be interactive processes or automated applications



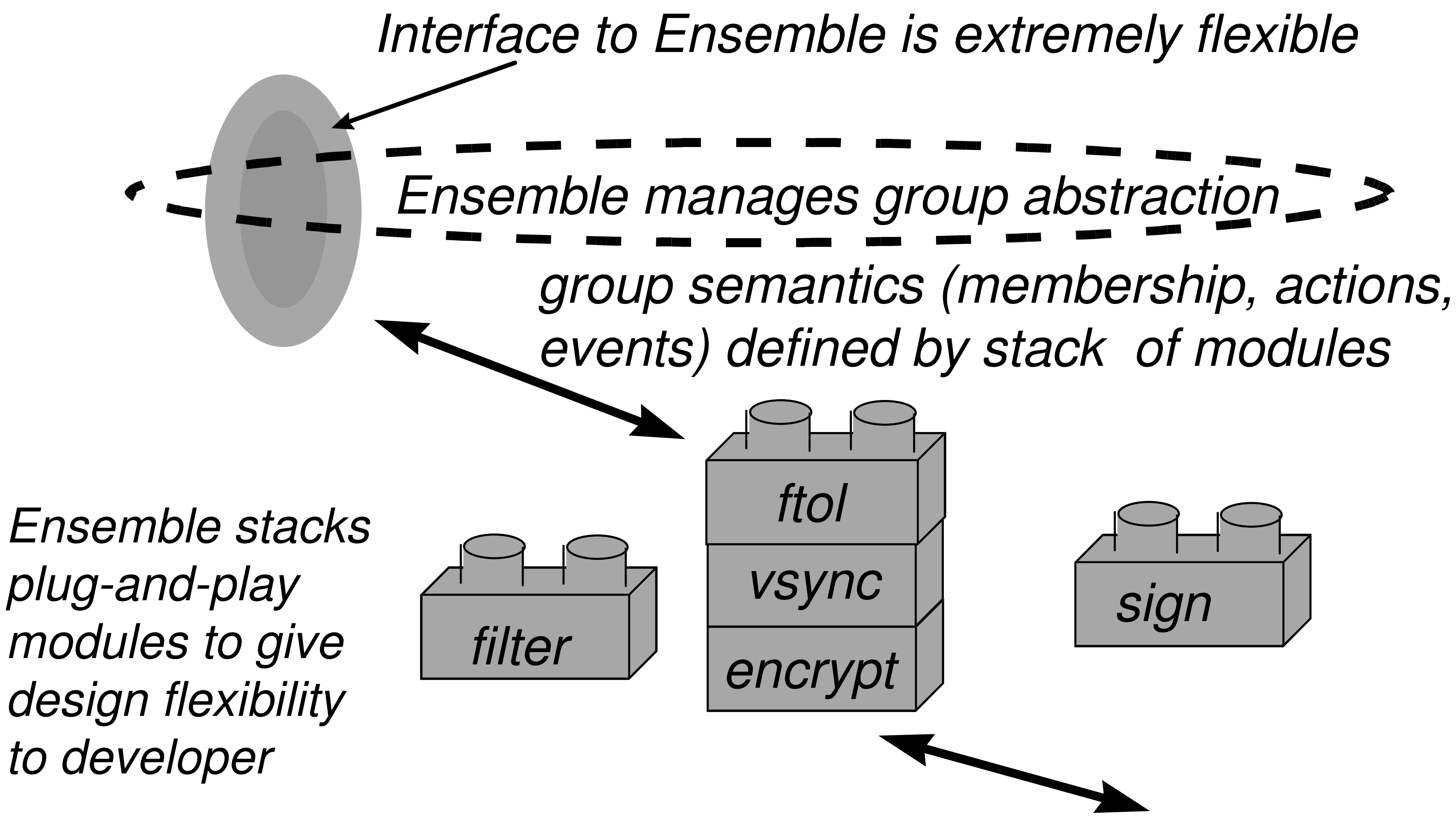
Processes Communicate Through Identical Multicast Protocol Stacks



Superimposed Groups in Application With Multiple Subsystems



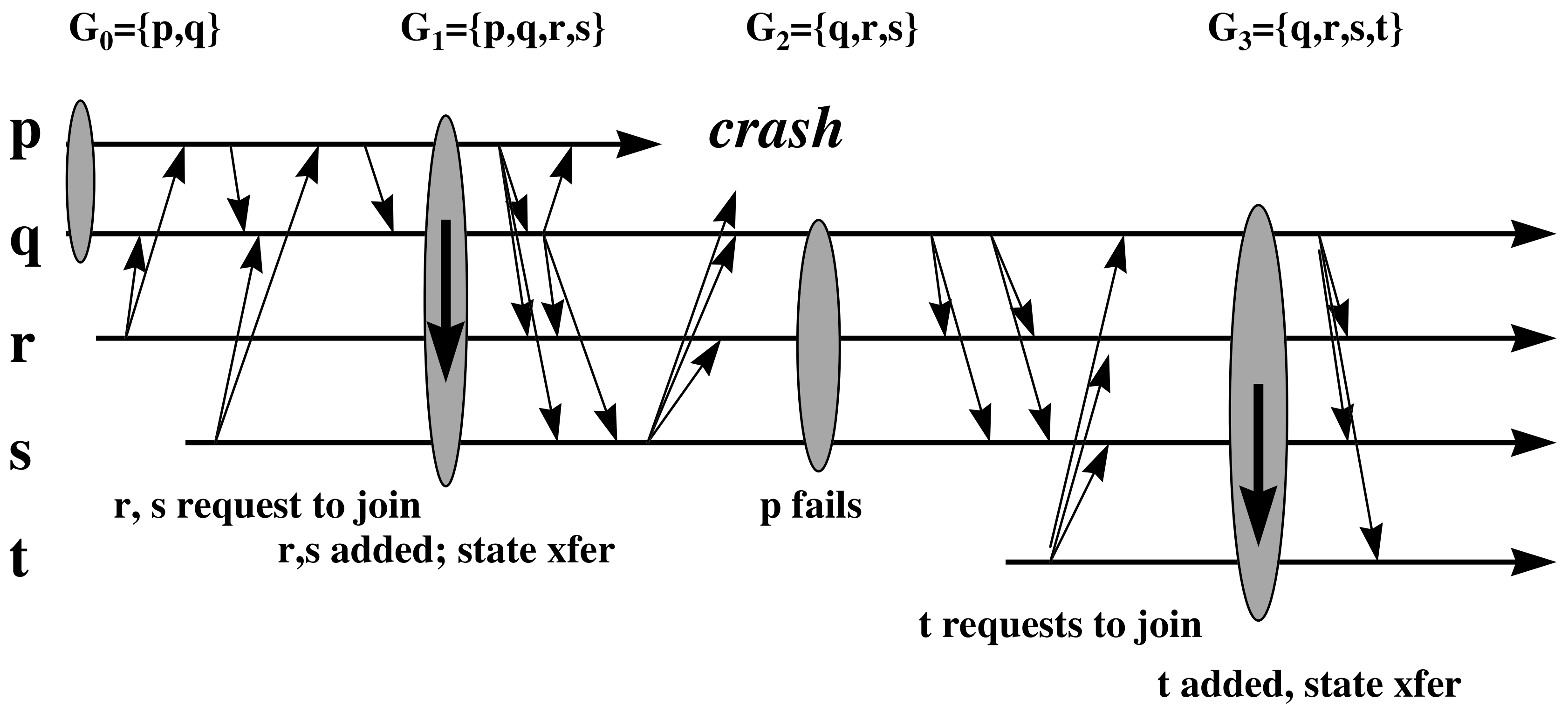
Layered Microprotocols in Ensemble



Why Process Groups?

- Used for replication, load-balancing, transparent fault-tolerance in servers
- Useful for secure multicast key management
- Can support flexible firewalls and filters
- Groups of clients in conference share media flows, agree on who is involved and what they are doing, manage security keys and QoS, etc...
- WAN groups for adaptive, partitionable systems

Virtual Synchrony Model



... to date, the *only* widely adopted model for consistency and fault-tolerance in highly available networked applications

Horus/Ensemble Performance

- A major focus for Van Renesse
- Over UNet: 85,000 to 100,000 small multicasts per second, saturates a 155Mbit ATM, end-to-end latencies as low as 65us.
- We obtain this high performance by “protocol compilation” of our stacks
- Ensemble is coded in ML which facilitates automated code transformations

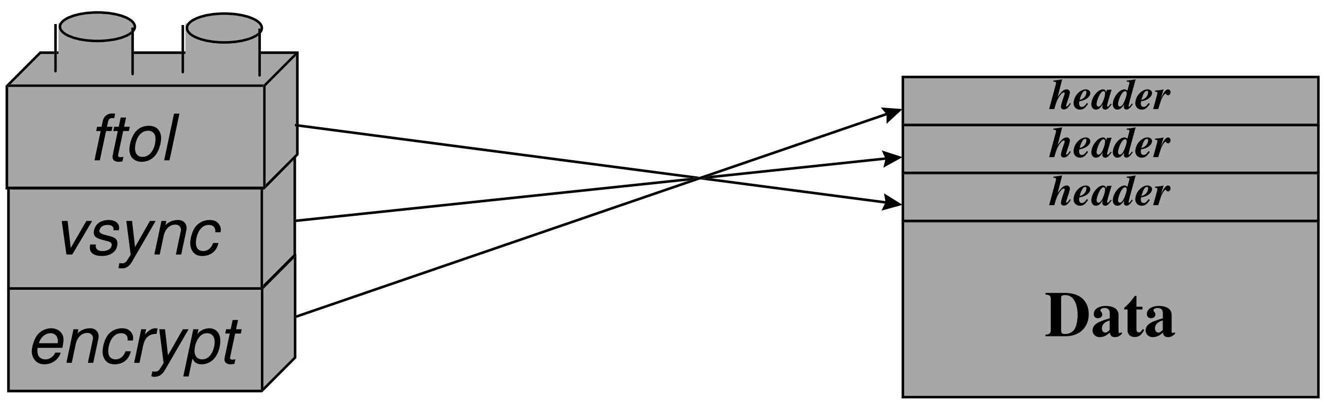
Getting those impressive numbers

- First had to work with a non-standard UNIX communication stack.
- Problem is that UNIX does so much copying that latency and throughput are always very poor.
- We used U-Net, a zero-copy communications stack from Thorsten Von Eicken's group. It runs on UNIX and NT

But U-Net Didn’t Help Very Much

- Layers have intrinsic costs:
  - Each tends to assume that it will run “by itself” hence each has its own header format. Even a single bit will need to be padded to 32 or 64 bits
  - Many layers only touch a small percentage of messages, yet each layer “sees” every message
  - Little opportunity for amortization of costs

Overhead



Van Renesse: Reorganizing Layers

- First create a notion of virtual headers
  - Layer says “I need 2 bits and an 8-bit counter”
  - Dynamically (at run time), Horus system “compiles” layers and builds shared message headers
  - Each layer accesses its fields through macros
  - Then separate into often changing, rarely changing, and static header information. Send the static stuff once, the rarely changing information only if it changes, the dynamic part on every message.

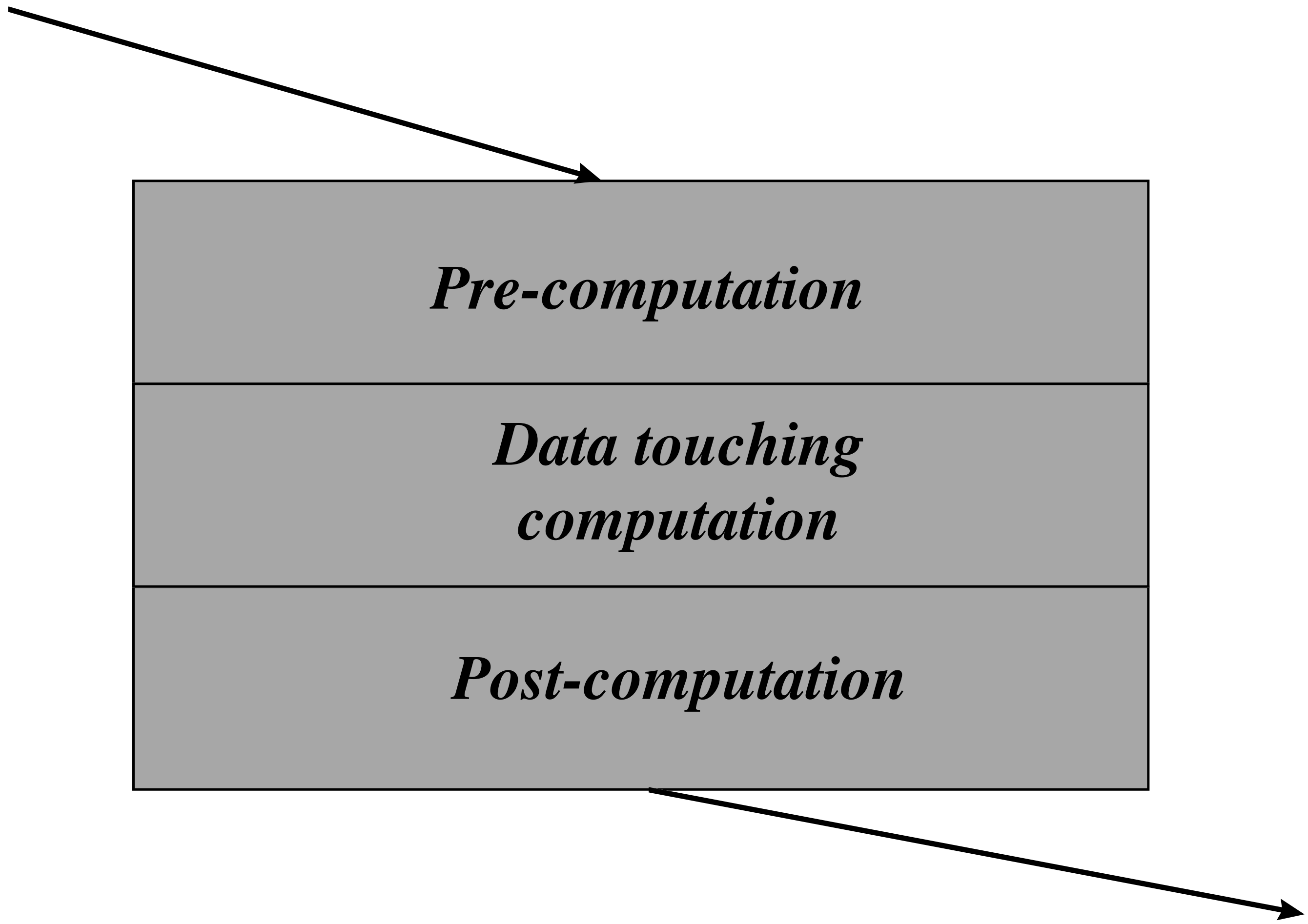
Impact of header optimizations?

- Average message in Horus used to carry one hundred bytes or more of header data
- Now see true size of header drop by 50% due to compaction opportunity
- Highly dynamic header: just a few bytes
- One bit to signal presence of “rarely changing” header information
-

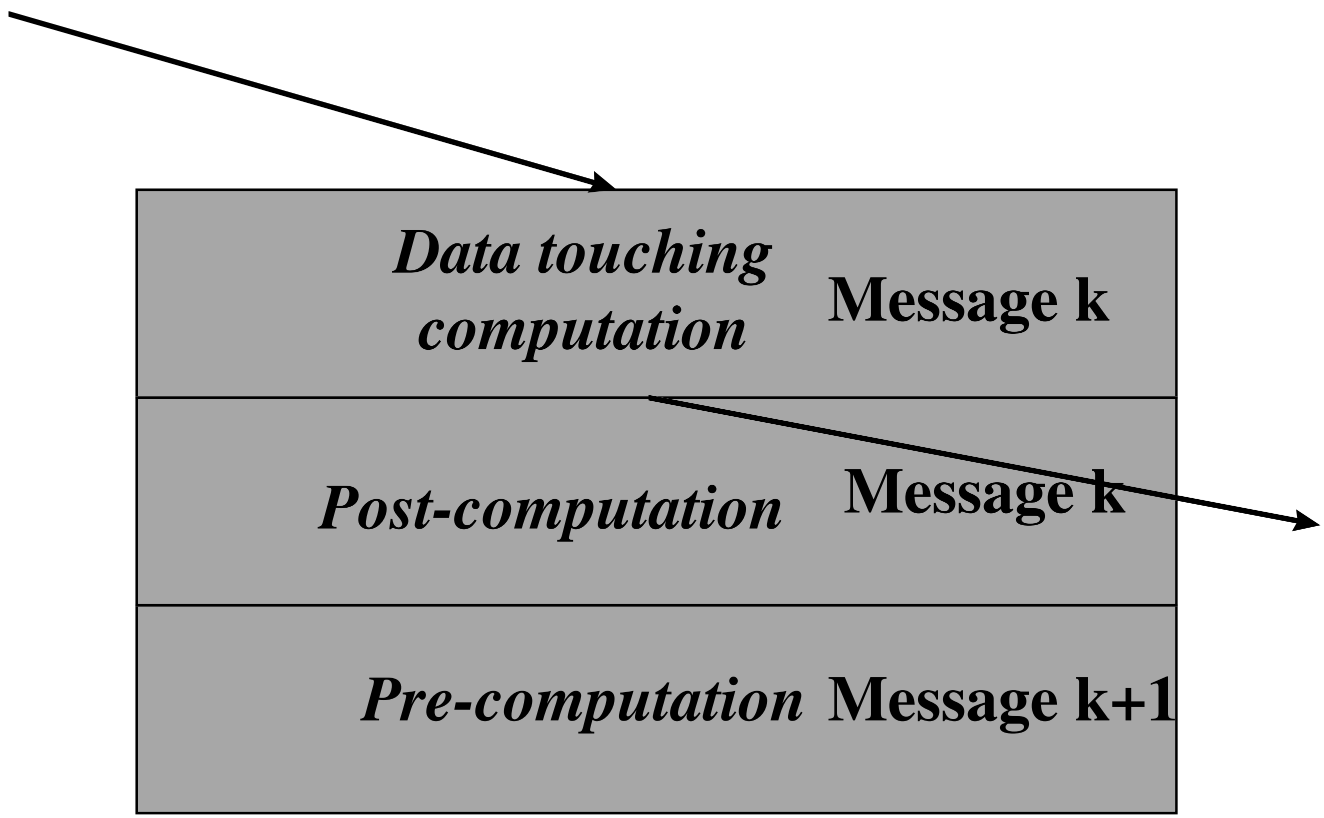
Next step: Code restructuring

- View original Horus layers as having 3 parts:
  - "Pre" computation (can do before seeing message)
  - Data touching computation (needs to see message)
  - "Post" computation (can delay until message sent)
- Move "pre" computing to after "post" and do both off critical path
- Effect is to slash latencies on critical path

Three stages to a layer



Restructured layer



Final step: Batch messages

- Look for places where lots of messages pass by
- Combine (if safe) into groups of messages blocked for efficient use of the network
- Effect is to amortize costs over many messages at a time

Final step: Batch messages

- Look for places where lots of messages pass by
- Combine (if safe) into groups of messages blocked for efficient use of the network
- Effect is to amortize costs over many messages at a time

… but a problem emerges: all of this makes Horus messy, much less modular

Ensemble: Move to ML

- Idea now is to offer a C/C++/Java interface but build stack itself in ML
- NuPrl can manipulate the ML stacks offline
- Hayden exploits this to obtain same performance as in Horus but with less complexity

Example: Partial Evaluation Idea

- Annotate the Ensemble stack components with indications of critical path:
  - Green messages always go left. Red messages right
  - For green messages, this loop only loops once
  - … etc
- Now NuPrl can partially evaluate a stack: once for “message is green”, once for “red”

Why are two stacks better than one?

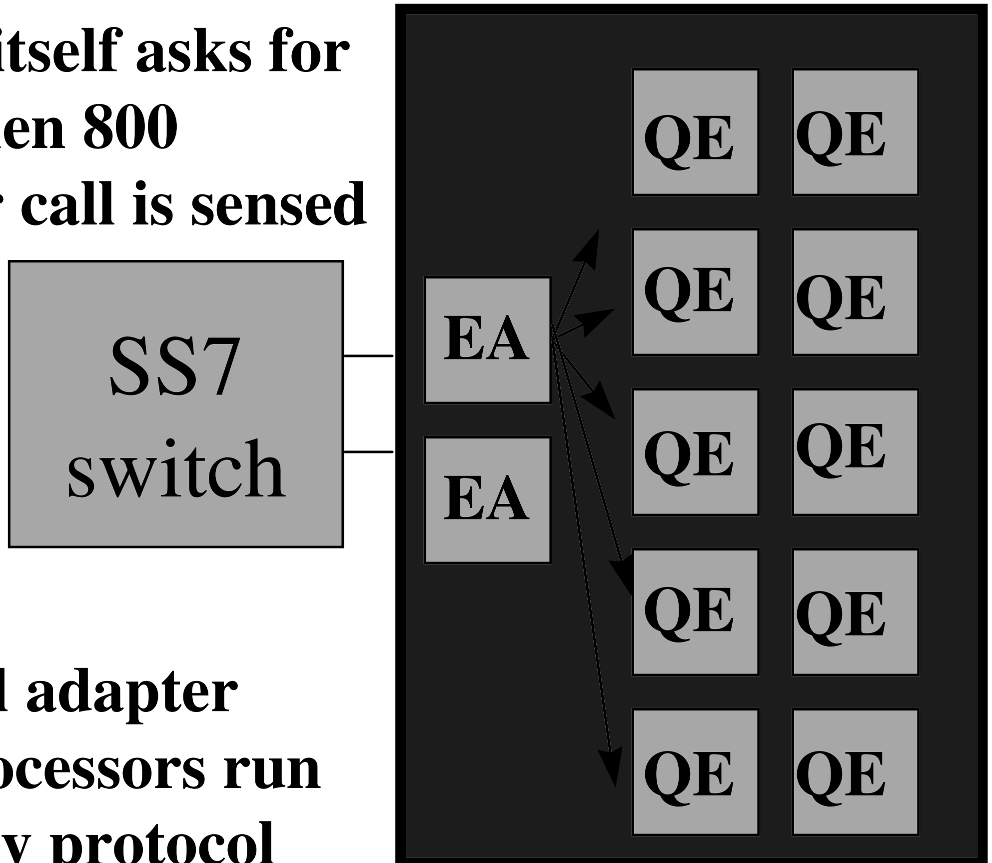
- Now have an if statement above two machine-generated stacks: If green … else (red) ….
- Each stack may be much compacted; critical path drastically shorter
- Also can do inline code expansion
- Result is a single highly optimized stack that is provably equivalent to original stack!
- Ensemble perf. is even better than Horus

Friedman: Performance isn't enough

- Is this blinding performance fast enough for a demanding real-time use?
- Finding: yes, if Ensemble is used "very carefully" and if other effort is employed, but no, if Ensemble is just slapped into place

IN coprocessor example

Switch itself asks for help when 800 number call is sensed



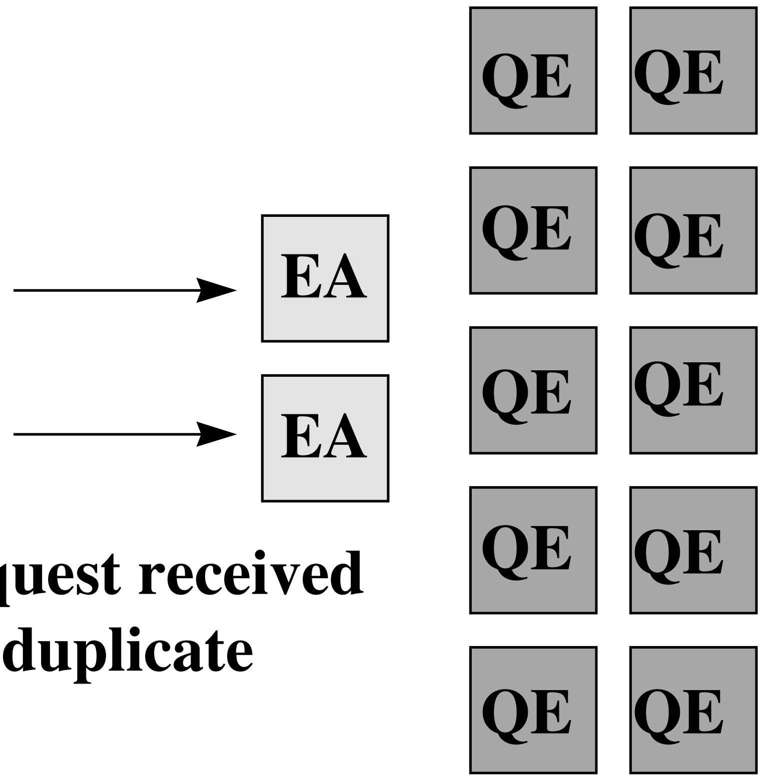
External adapter (EA) processors run the query protocol

Query Element (QE) processors do the 800-number lookup (in-memory database).

Goals: scaleable memory without loss of processing performance as number of nodes is increased

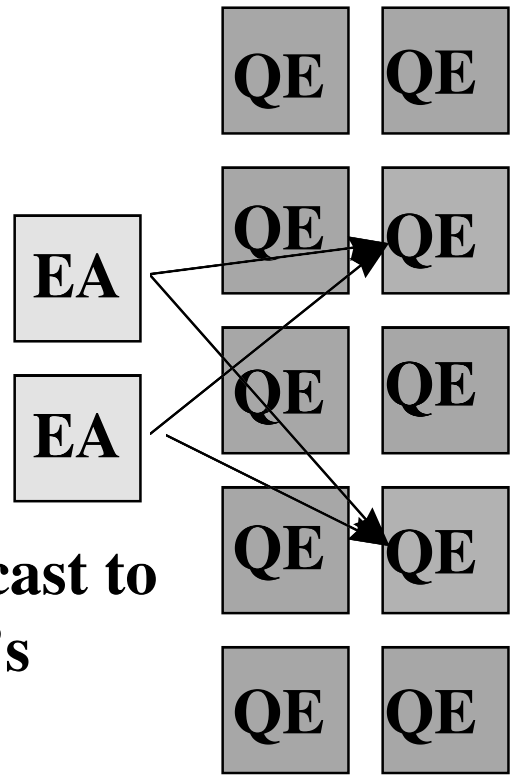
Primary backup scheme adapted (using small Horus process groups) to provide fault-tolerance with real-time guarantees

Traditional Realtime Approach



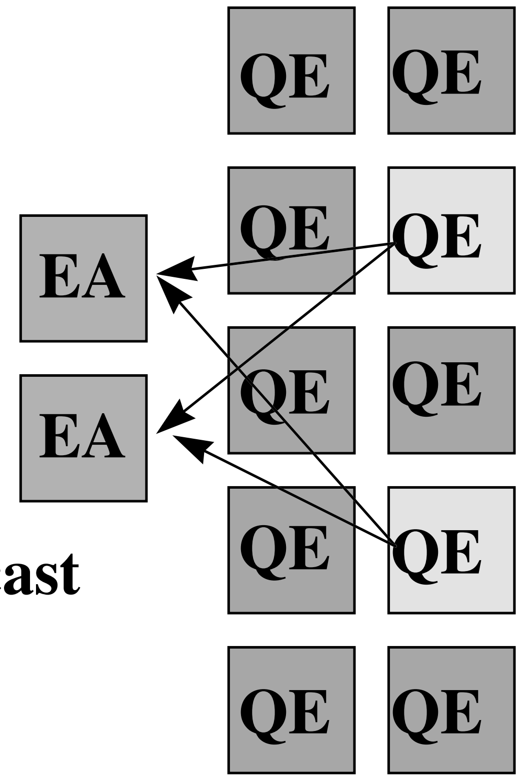
1. **Request received in duplicate**

Traditional Realtime Approach



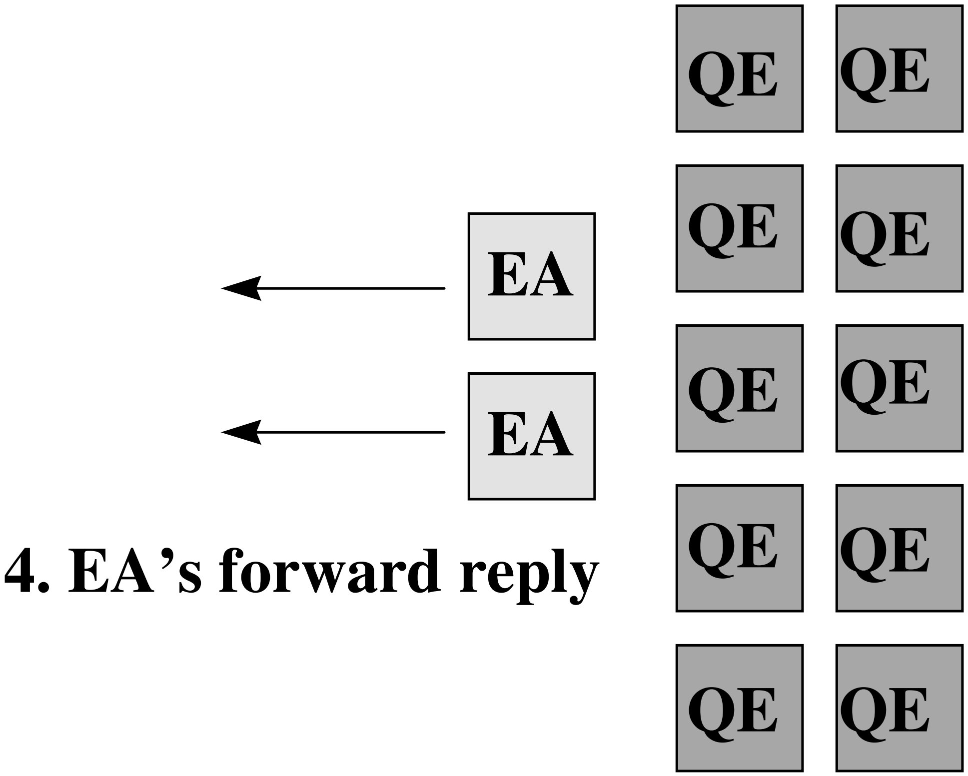
2. Request multicast to selected QE's

Traditional Realtime Approach



3. QE's multicast reply

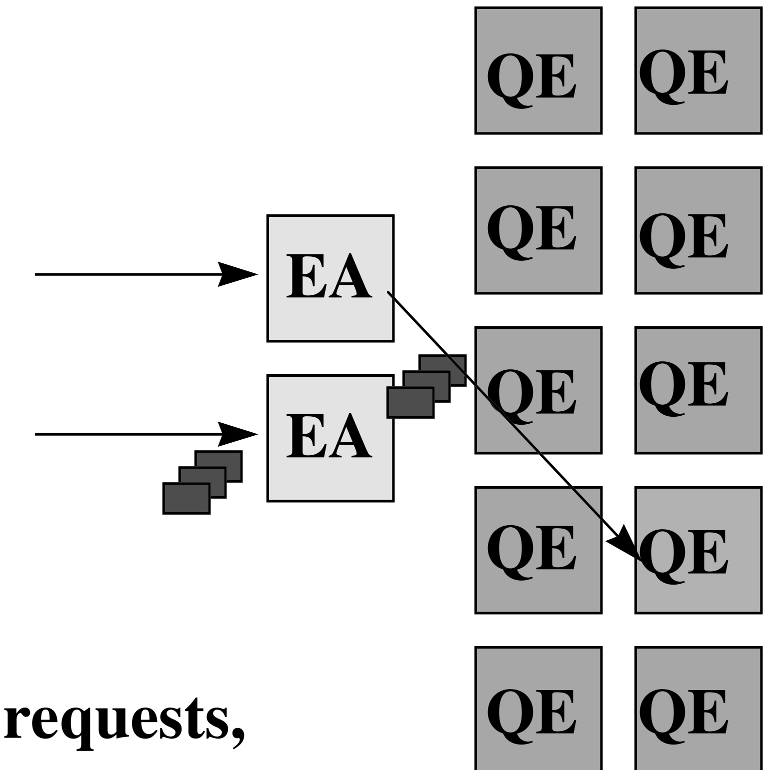
Traditional Realtime Approach



Criticism?

- Heavy overheads to obtain fault-tolerance
- No “batching” of requests
- Obvious match with group communication but overheads are prohibitive
- Likely performance? A few hundred requests per second, delays of 4-6 seconds to “fail-over” when a node is taken offline

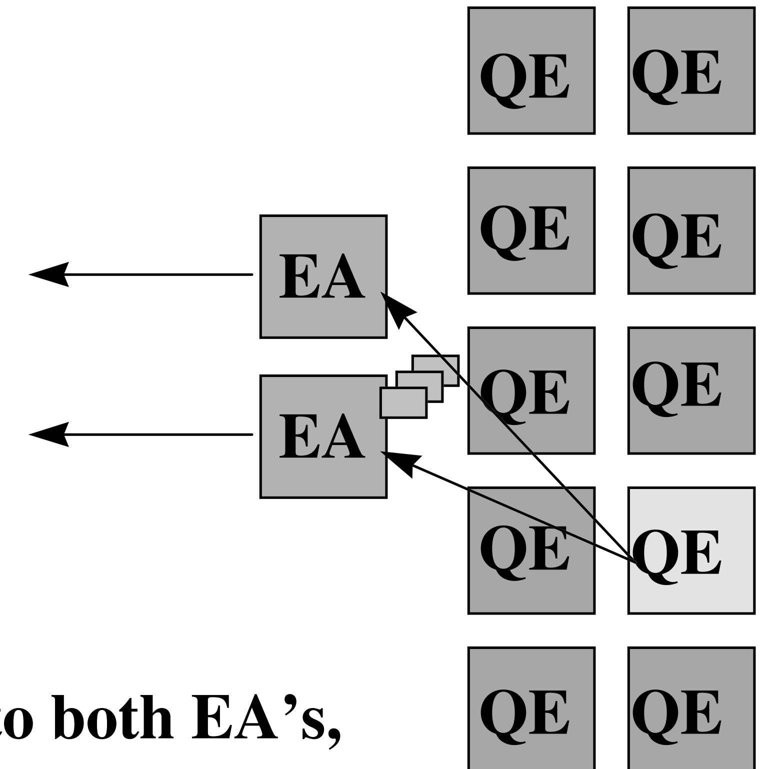
Friedman's Realtime Approach



EA's batch requests, primary sends a group at a time to single QE

Ensemble used to monitor status (live / faulty, load) of processing elements. EA's have this data.

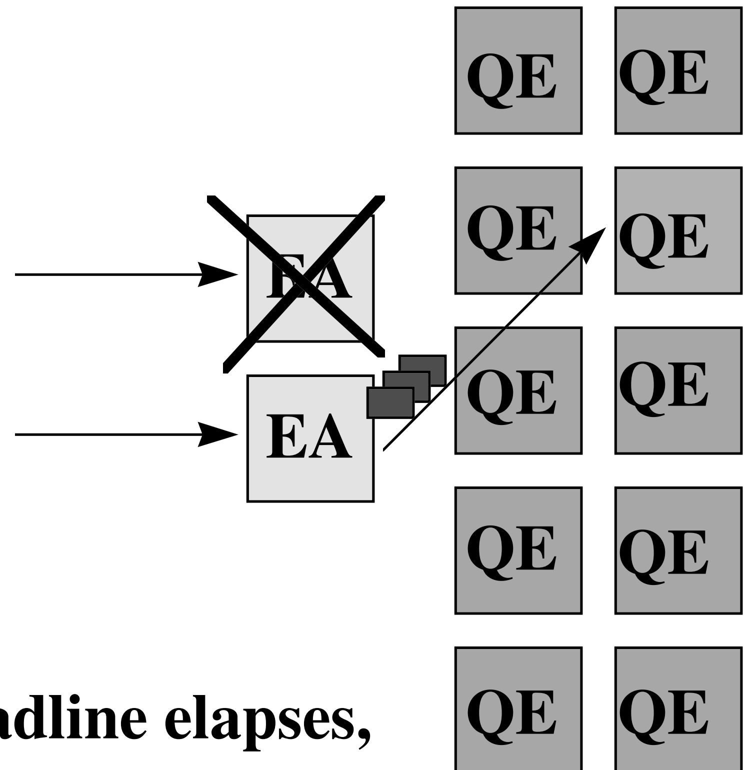
Friedman's Realtime Approach



QE replies to both EA's, they forward result

QE or EA could fail. Ensemble needs a few seconds to report this

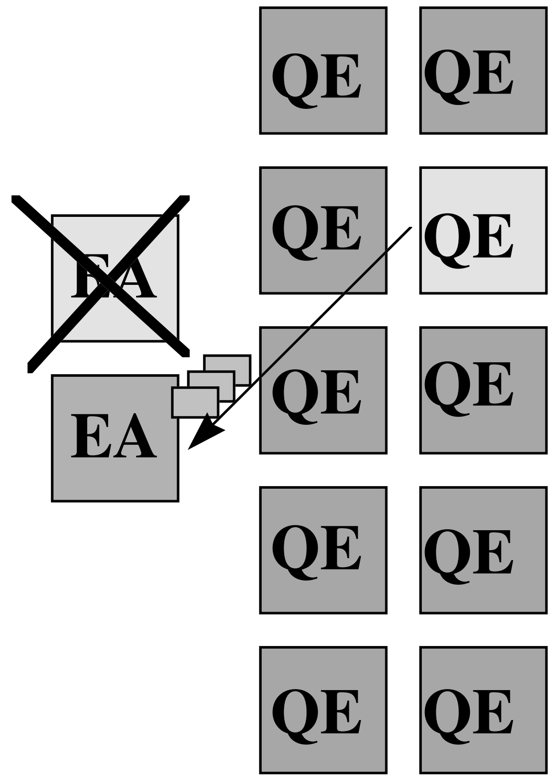
Friedman's Realtime Approach



If half of deadline elapses, backup EA retries with some other QE

Consistency of replicated data is key to correctness of this scheme

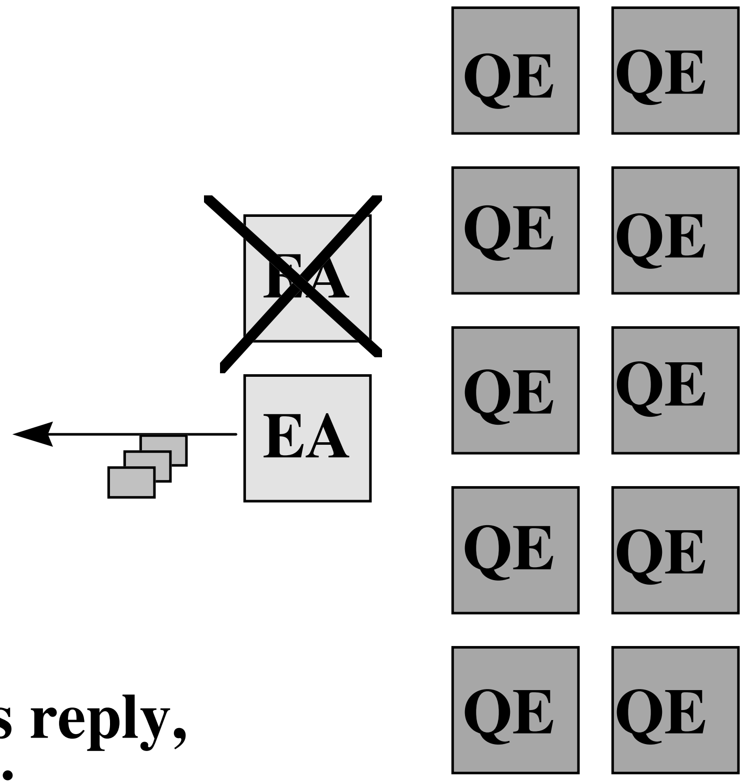
Friedman’s Realtime Approach



… QE replies

Consistency of replicated data is key to correctness of this scheme

Friedman's Realtime Approach



EA forwards reply, within deadline

Consistency of replicated data is key to correctness of this scheme

Friedman's Work

- Uses Horus/Ensemble to "manage" the cluster
- Designs special protocols based on Active Messages for batch-style handling of requests
- Demonstrates 20,000+ "calls" per second even during failures and restart of nodes, 98%+ responses within 100ms deadline
- Scaleable memory, computing and ability to upgrade components are big wins

Broader Project Goals for 1997

- Increased emphasis on integration with security standards and emerging world of Quality of Service guarantees
- More use of Ensemble to *manage protocol stacks external to our system*
- Explore adaptive behavior, especially for secure networks or secured subsystems
- Emphasis on four styles of computing system

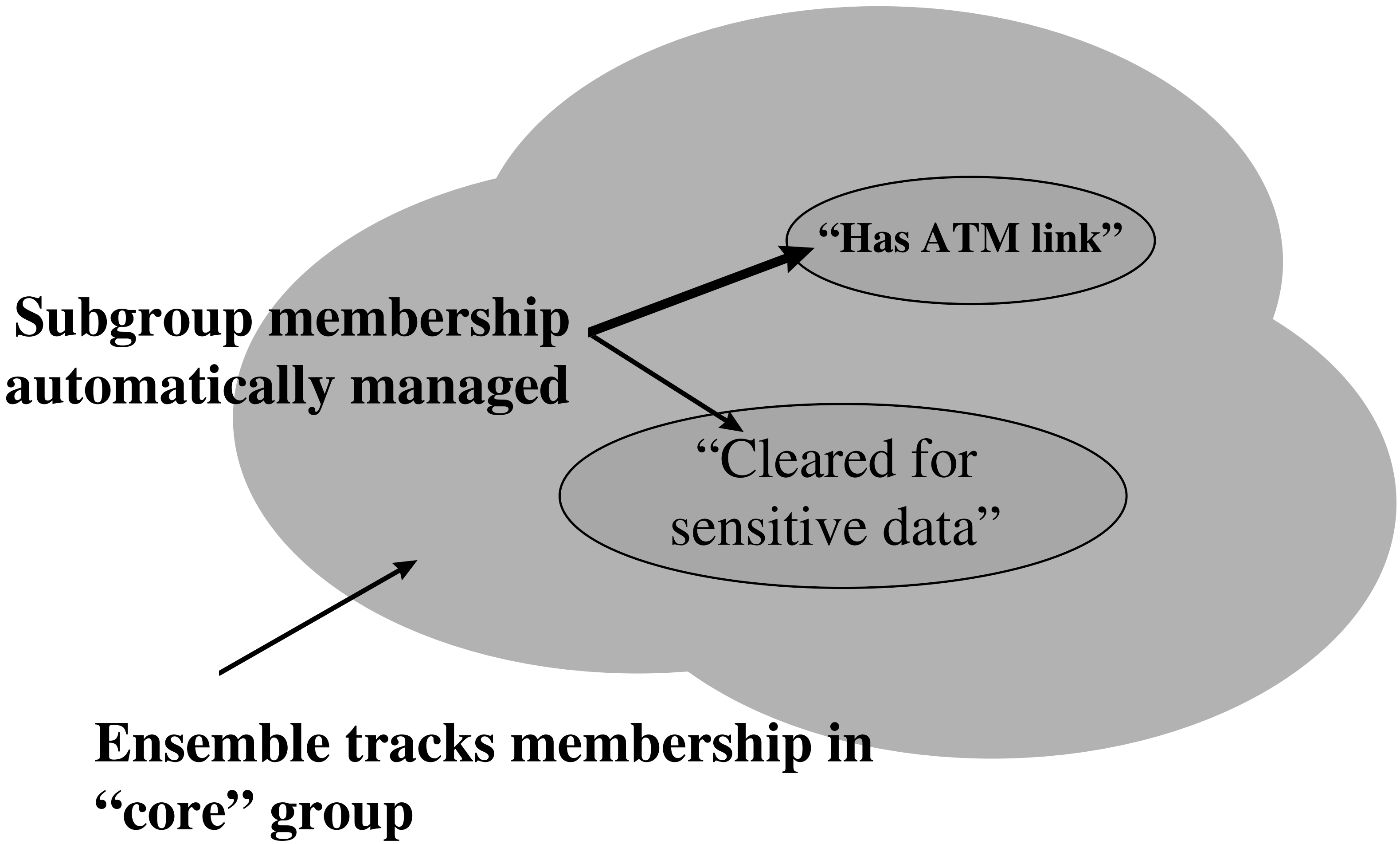
Secure Real-Time Cluster Servers

- This work extends Friedman's real-time server architecture to deal with IP fail-over
- Think of a TCP connection to a cluster server that remains up even if the compute node fails
- Our effort also deals with session key management so that security properties are preserved as fail-over takes place
- Goal: a "tool kit" in Ensemble distribution

Secure Adaptive Networks

- This work uses Ensemble to *manage* a subgroup of an Ensemble process group, or a set of "external" communication endpoints
- Goal is to demonstrate that we can exploit this to dynamically secure a network application that must adapt to changing conditions
- Can also download protocol stacks at runtime, a form of Active Network behavior

Secure Adaptive Networks



Secure Adaptive Networks

- Paper on initial work: on “Maestro”, a tool for management of subgroups of a group
- Initial version didn’t address security issues
- Now extending to integrate with our security layers, will automatically track subgroups and automatically handle

Probablistic Quality of Service

- Developing new protocols that scale better by relaxing reliability guarantees
- Easiest to understand these as having probablistic quality of service properties
- Our first solution of this sort is now working experimentally; seems extremely tolerant of transient misbehavior that seriously degrade performance in Isis and Horus/C

Four target computing environments

- Network layer itself: Ensemble to coordinate use of IPv6 or RSVP in multicast settings. We see as a prototype ***Highly Assured Network***
- Server clustering and fault-tolerance
- Wide-area file systems and server networks that tolerate partitioning failures
- User-level tools for building group conferencing and collaboration tools

Deliverables From Effort

- Ensemble is already available for UNIX platforms and port to NT is nearly complete
- Working with BBN to integrate with AquA for use in Quorum program (Gary Koob)
- R/T cluster tools and WAN partitioning tools available by mid summer
- Adaptive & probablistic tools by late this year

http://www.cs.cornell.edu/Info/Projects/HORUS/