Messaging and Group Communication

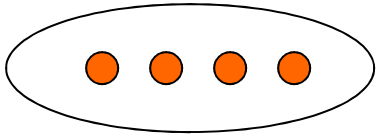
ICS 230 Distributed Systems

Group Communication

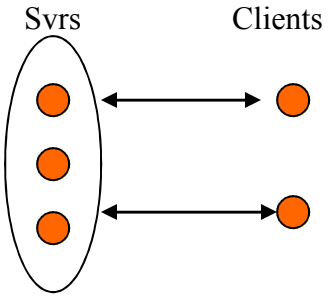
- Communication to a collection of processes – process group
- Group communication can be exploited to provide
  - Simultaneous execution of the same operation in a group of workstations
  - Software installation in multiple workstations
  - Consistent network table management
- Who needs group communication ?
  - Highly available servers
  - Conferencing
  - Cluster management
  - Distributed Logging….

What type of group communication ?

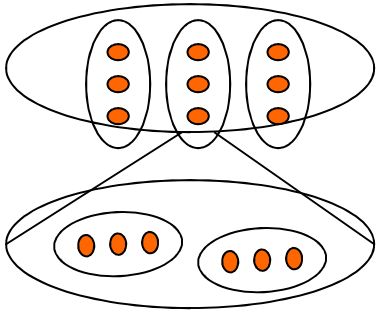
- Peer
  - All members are equal
  - All members send messages to the group
  - All members receive all the messages



- Client-Server
  - Common communication pattern
    - replicated servers
  - Client may or may not care which server answers

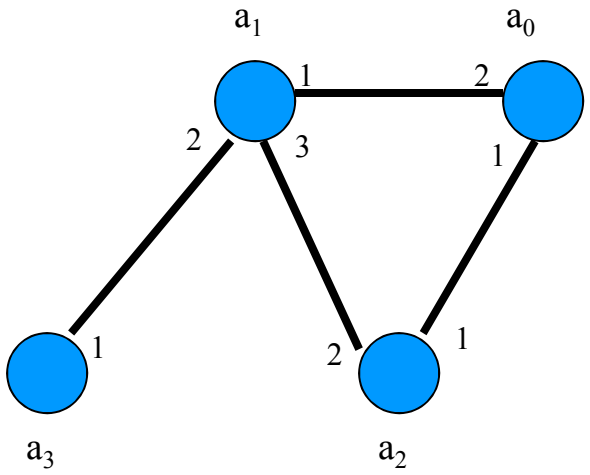


- Diffusion group
  - Servers sends to other servers and clients
- Hierarchical
  - Highly and easy scalable



Message Passing System

- A system consist of n objects a_0, …, a_{n-1}
- Each object a_i is modeled as a (possible infinite) state machine with state set Q_i
- The edges incident on a_i are labeled arbitrarily with integers 1 through r, where r is the degree of a_i
- Each state of a_i contains 2r special components, $outbuf_i[l]$, $inbuf_i[l]$, for every $1 \leq l \leq r$
- A configuration is a vector $C=(q_o,\ldots,q_{n-1})$, where q_i is the state of a_i



Message Passing System (II)

- A system is said to be asynchronous if there is no fixed upper bound on how long it takes a message to be delivered or how much time elapses between consecutive steps
- Point-to-point messages
  - $snd_i(m)$
  - $rcv_i(m,j)$
- Group communication
  - Broadcast
    - one-to-all relationship
  - Multicast
    - one-to-many relationship
    - A variation of broadcast where an object can target its messages to a specified subset of objects

Using Traditional Transport Protocols

- TCP/IP
  - Automatic flow control, reliable delivery, connection service, complexity
    - linear degradation in performance
- Unreliable broadcast/multicast
  - UDP, IP-multicast - assumes h/w support
  - message losses high(30%) during heavy load
    - Reliable IP-multicast very expensive

Group Communication Issues

- Ordering
- Delivery Guarantees
- Membership
- Failure

Ordering Service

- Unordered
- Single-Source FIFO (SSF)
  - For all messages m_1, m_2 and all objects a_i, a_j, if a_i sends m_1 before it sends m_2, then m_2 is not received at a_j before m_1 is
- Totally Ordered
  - For all messages m_1, m_2 and all objects a_i, a_j, if m_1 is received at a_i before m_2 is, the m_2 is not received at a_j before m_1 is
- Causally Ordered
  - For all messages m_1, m_2 and all objects a_i, a_j, if m_1 happens before m_2, then m_2 is not received at a_i before m_1 is

Delivery guarantees

- Agreed Delivery
  - guarantees total order of message delivery and allows a message to be delivered as soon as all of its predecessors in the total order have been delivered.
- Safe Delivery
  - requires in addition, that if a message is delivered by the GC to any of the processes in a configuration, this message has been received and will be delivered to each of the processes in the configuration unless it crashes.

Membership

- Messages addressed to the group are received by all group members
- If processes are added to a group or deleted from it (due to process crash, changes in the network or the user's preference), need to report the change to all active group members, while keeping consistency among them
- Every message is delivered in the context of a certain configuration, which is not always accurate. However, we may want to guarantee
  - Failure atomicity
  - Uniformity
  - Termination

Failure Model

- Failures types
  - Message omission and delay
    - Discover message omission and (usually) recovers lost messages
  - Processor crashes and recoveries
  - Network partitions and re-merges
- Assume that faults do not corrupt messages (or that message corruption can be detected)
- Most systems do not deal with Byzantine behavior
- Faults are detected using an unreliable fault detector, based on a timeout mechanism

Some GC Properties

- Atomic Multicast
  - Message is delivered to all processes or to none at all. May also require that messages are delivered in the same order to all processes.
- Failure Atomicity
  - Failures do not result in incomplete delivery of multicast messages or holes in the causal delivery order
- Uniformity
  - A view change reported to a member is reported to all other members
- Liveness
  - A machine that does not respond to messages sent to it is removed from the local view of the sender within a finite amount of time.

Virtual Synchrony

- Virtual Synchrony
  - Introduced in ISIS, orders group membership changes along with the regular messages
  - Ensures that failures do not result in incomplete delivery of multicast messages or holes in the causal delivery order(failure atomicity)
  - Ensures that, if two processes observe the same two consecutive membership changes, receive the same set of regular multicast messages between the two changes
    - A view change acts as a barrier across which no multicast can pass
  - Does not constrain the behavior of faulty or isolated processes

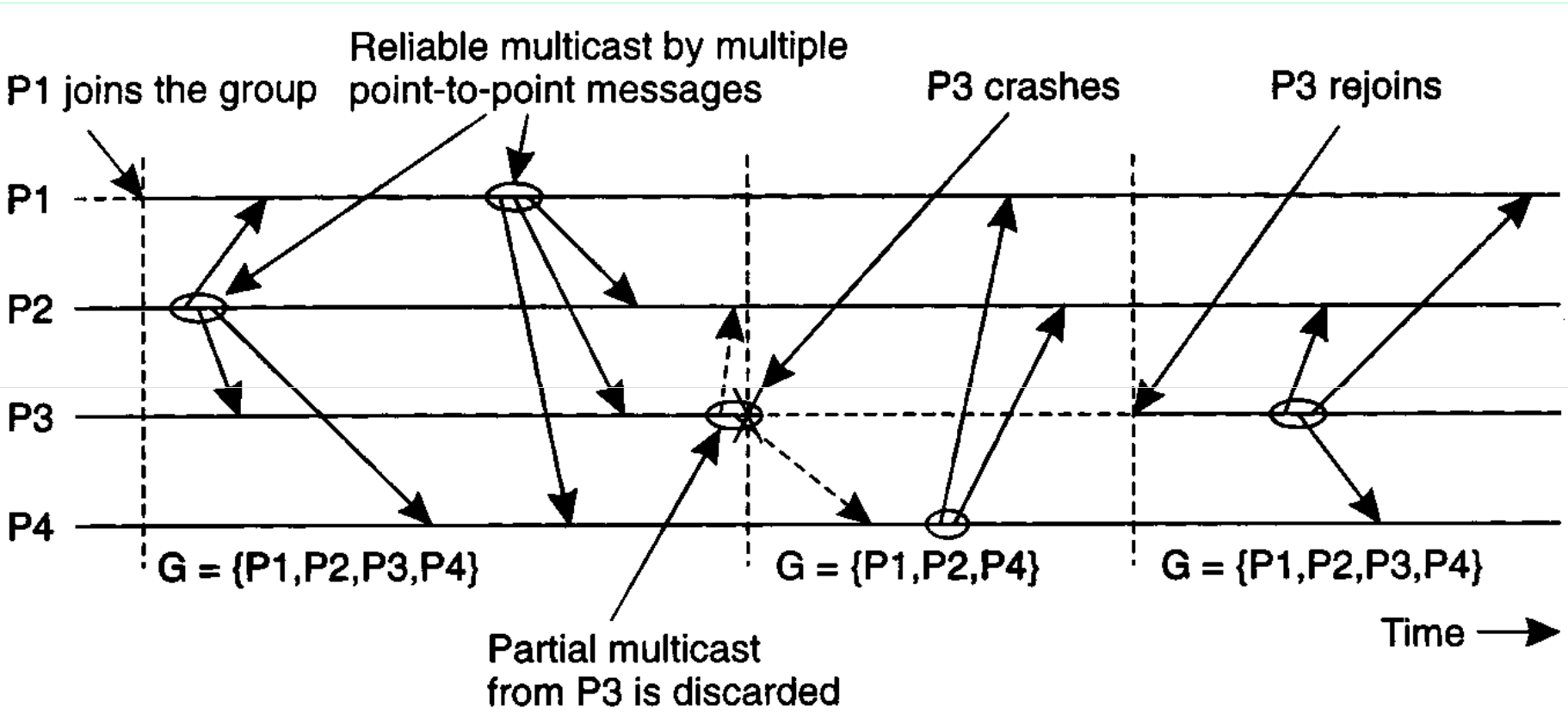


Figure 7-12. The principle of virtual synchronous multicast.

More Interesting GC Properties

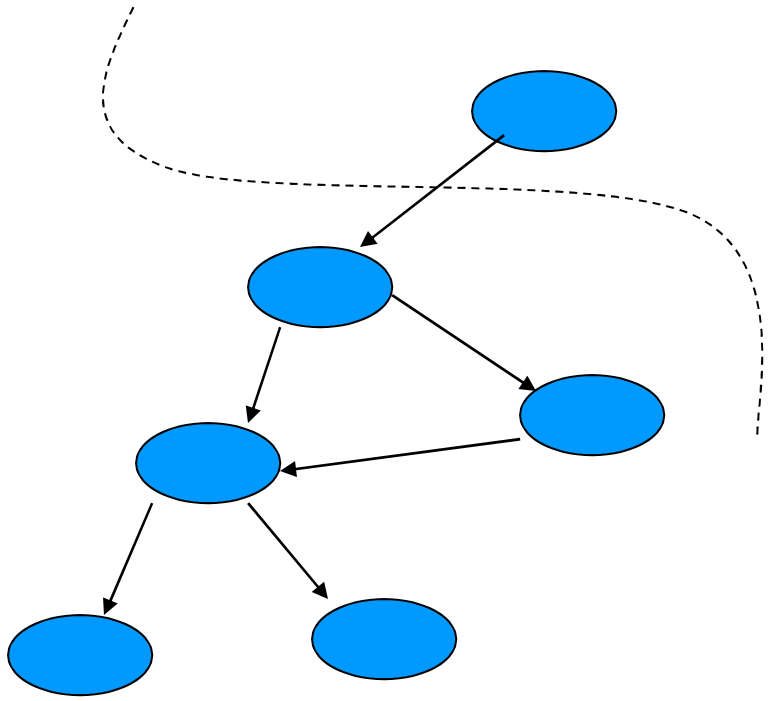
- There exists a mapping *k* from the set of messages appearing in all $rcv_i(m)$ for all i, to the set of messages appearing in $snd_i(m)$ for all i, such that each message *m* in a rcv() is mapped to a message with the same content appearing in an earlier snd() and:
- Integrity
  - *k* is well defined. *i.e.* every message received was previously sent.
- No Duplicates
  - *k* is one to one. *i.e.* no message is received more than once
- Liveness
  - *k* is onto. *i.e.* every message sent is received

Reliability Service

- A service is reliable (in presence of *f* faults) if exists a partition of the object indices into faulty and non-faulty such that there are at most *f* faulty objects and the mapping of *k* must satisfy:
  - Integrity
  - No Duplicates
    - no message is received more than once at any single object
  - Liveness
    - Non-faulty liveness
      - When restricted to non-faulty objects, *k* is onto. *i.e.* all messages broadcast by a non-faulty object are **eventually** received by all non-faulty objects
    - Faulty liveness
      - Every message sent by a faulty object is either received by **all** non-faulty objects or by **none** of them

Faults and Partitions

- When detecting a processor P from which we did not hear for a certain timeout, we issue a fault message
- When we get a fault message, we adopt it (and issue our copy)
- Problem: maybe P is only slow
- When a partition occurs, we can not always completely determine who received which messages (there is no solution to this problem)



Extended virtual synchrony

- Introduced in Totem
- Processes can fail and recover
- Network can partition and remerge
- Does not solve all the problems of recovery in fault-tolerant distributed system, but it avoid inconsistencies

Extended Virtual Synchrony(cont.)

- Virtual synchrony handles recovered processes as new processes
  - Can cause inconsistencies with network partitions
- Network partitions are real
  - Gateways, bridges, wireless communication

Extended Virtual Synchrony Model

- Network may partition into finite number of components
  - Two or more may merge to form a larger component
- Each membership with a unique identifier is a *configuration.*
  - Membership ensures that all processes in a configuration agree on the membership of that configuration

Regular and Transitional Configurations

- To achieve safe delivery with partitions and remerges, the EVS model defines:
  - Regular Configuration
    - New messages are broadcast and delivered
    - Sufficient for FIFO and causal communication modes
  - Transitional Configuration
    - No new messages are broadcast, only remaining messages from prior regular configuration are delivered.
  - Regular configuration may be followed and preceeded by several transitional configurations.

Configuration change

- Process in a regular or transitional configuration can deliver a configuration change message s.t.
  - Follows delivery of every message in the terminated configuration and precedes delivery of every message in the new configuration.
- Algorithm for determining transitional configuration
  - When a membership change is identified
    - Regular conf members (that are still connected) start exchanging information
    - If another membership change is spotted (e.g. failure cascade), this process is repeated all over again.
    - Upon reaching a decision (on members and messages) – process delivers transitional configuration message to members with agreed list of messages.
    - After delivery of all messages, new configuration is delivered.

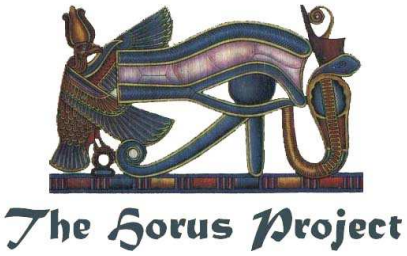
Totem

- Provides a Reliable totally ordered multicast service over LAN
- Intended for complex applications in which fault-tolerance and soft real-time performance are critical
  - High throughput and low predictable latency
  - Rapid detection of, and recovery from, faults
  - System wide total ordering of messages
  - Scalable via hierarchical group communication
  - Exploits hardware broadcast to achieve high-performance
- Provides 2 delivery services
  - Agreed
  - Safe
- Use timestamp to ensure total order and sequence numbers to ensure reliable delivery

ISIS

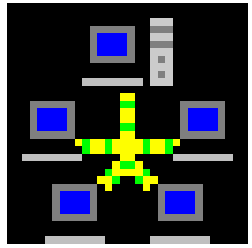
- Tightly coupled distributed system developed over loosely coupled processors
- Provides a toolkit mechanism for distributing programming, whereby a DS is built by interconnecting fairly conventional non-distributed programs, using tools drawn from the kit
- Define
  - how to create, join and leave a group
  - group membership
  - virtual synchrony
- Initially point-to-point (TCP/IP)
- Fail-stop failure model

Horus



- Aims to provide a very flexible environment to configure group of protocols specifically adapted to problems at hand
- Provides efficient support for virtual synchrony
- Replaces point-to-point communication with group communication as the fundamental abstraction, which is provided by stacking protocol modules that have a uniform (upcall, downcall) interface
- Not every sort of protocol blocks make sense
- HCPI
  - Stability of messages
  - membership
- Electra
  - CORBA-Compliant interface
  - method invocation transformed into multicast

Transis



- How different components of a partition network can operate autonomously and then merge operations when they become reconnected ?
- Are different protocols for fast-local and slower-cluster communication needed ?
- A large-scale multicast service designed with the following goals
  - Tackling network partitions and providing tools for recovery from them
  - Meeting needs of large networks through hierarchical communication
  - Exploiting fast-clustered communication using IP-Multicast
- Communication modes
  - FIFO
  - Causal
  - Agreed
  - Safe

Future Challenges

- Secure group communication architecture
- Formal specifications of group communication systems
- Support for CSCW and multimedia applications
- Dynamic Virtual Private Networks
- Next Generations
  - Spread
  - Ensemble
- Wireless networks ?
  - Group based Communication with incomplete spatial coverage
  - dynamic membership

Horus

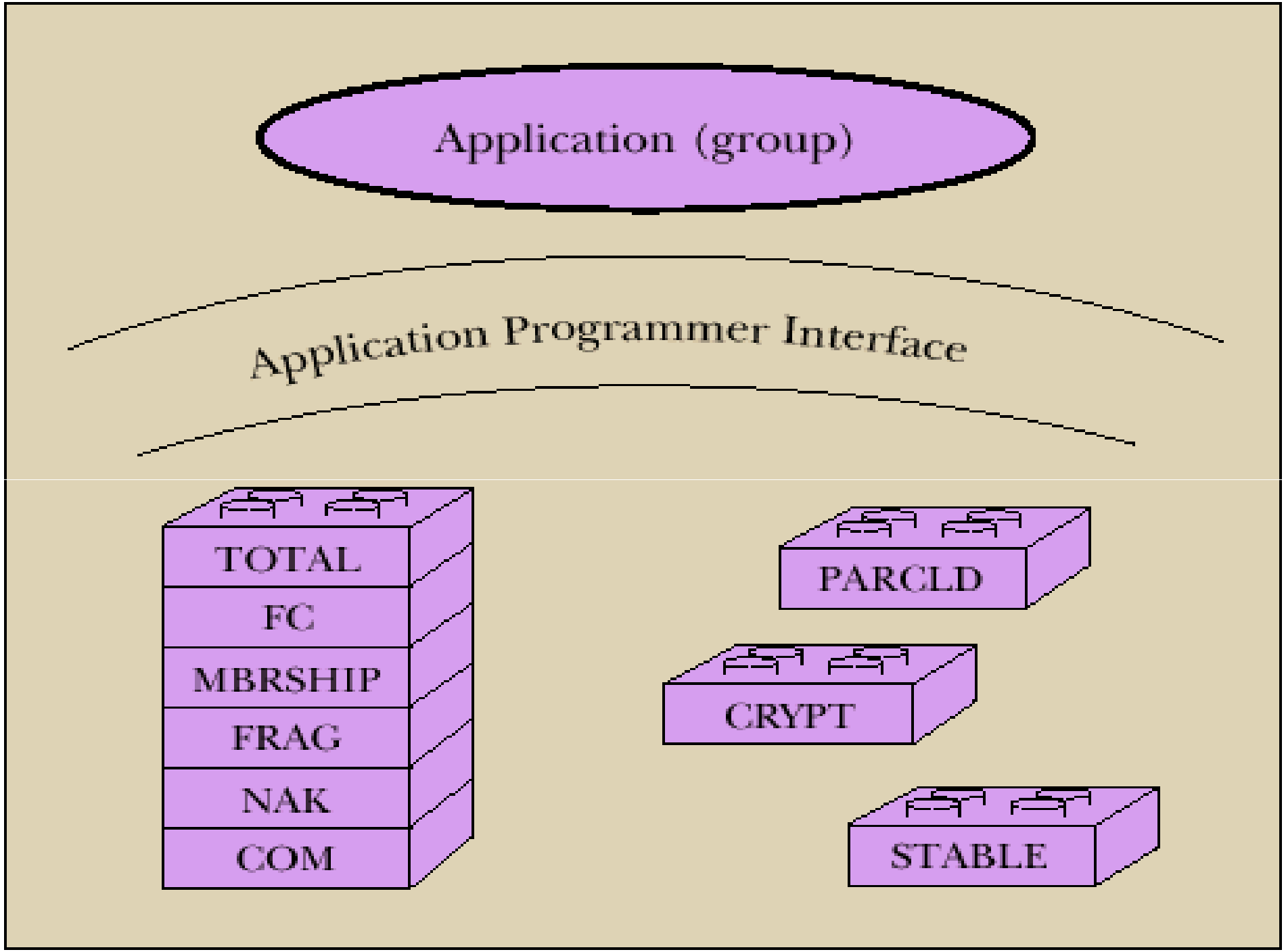
A Flexible Group Communication Subsystem

Horus: A Flexible Group Communication System

- Flexible group communication model to application developers.
  1. System interface
  2. Properties of Protocol Stack
  3. Configuration of Horus
     - Run in userspace
     - Run in OS kernel/microkernel

Architecture

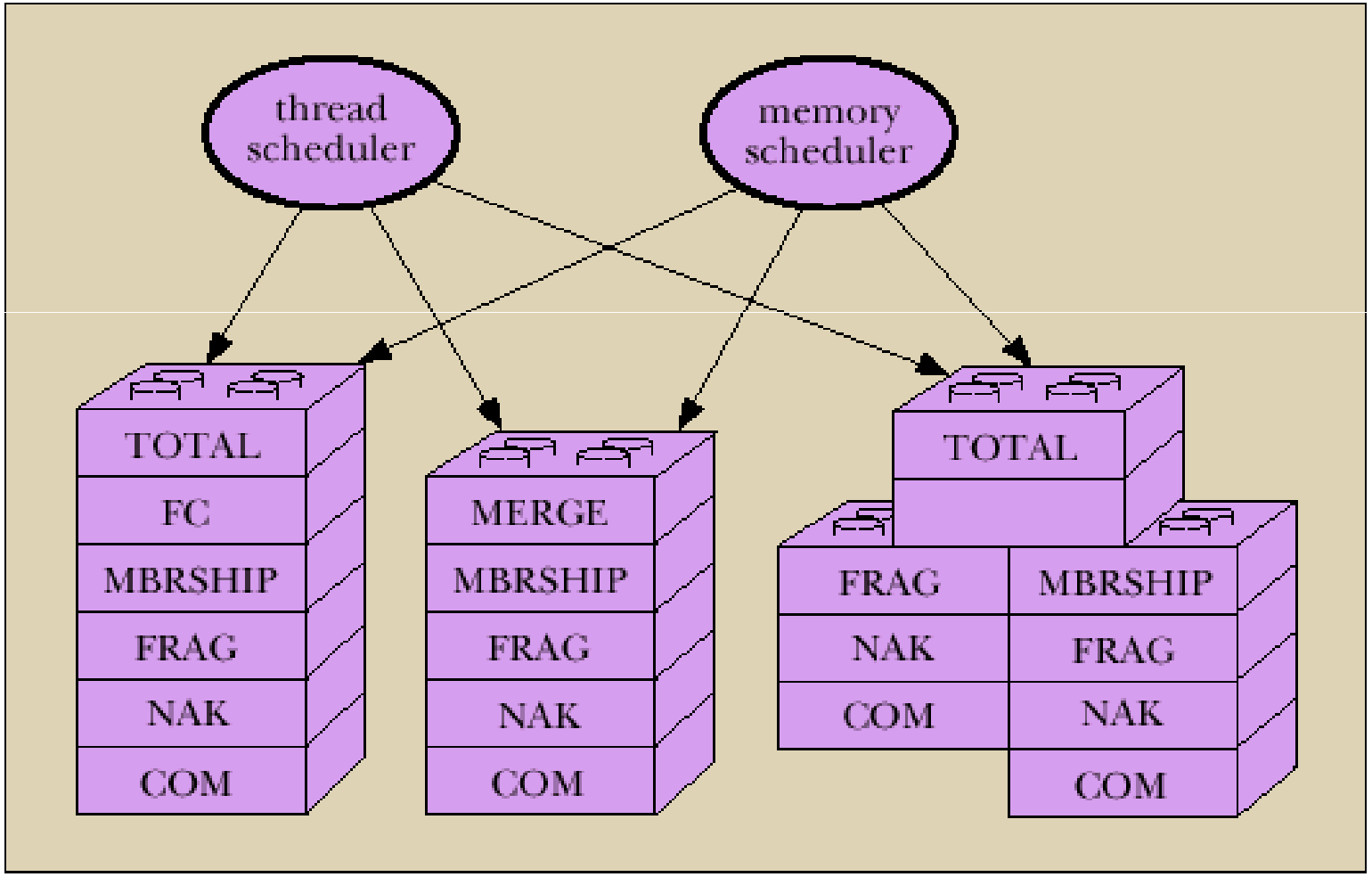
- Central protocol => Lego Blocks
- Each Lego block implements a communication feature.
- Standardized top and bottom interface (HCPI)
  - Allow blocks to communicate
  - A block has entry points for upcall/downcall
  - Upcall=receive mesg, Downcall=send mesg.
- Create new protocol by rearranging blocks.


Application (group)
Application Programmer Interface
TOTAL
FC
MBRSHIP
FRAG
NAK
COM
PARCLD
CRYPT
STABLE

Message_send

- Lookup the entry in topmost block and invokes the function.
- Function adds header
- Message_send is recursively sent down the stack
- Bottommost block invokes a driver to send message.

- Each stack shielded from each other.
- Have own threads and memory scheduler



Endpoints, Group, and Message Objects

- Endpoints
  - Models the communicating entity
  - Have address (used for membership), send and receive messages
- Group
  - Maintain local state on an endpoint.
  - Group address: to which message is sent
  - View: List of destination endpoint addr of accessible group members
- Message
  - Local storage structure
  - Interface includes operation pop/push headers
  - Passed by reference

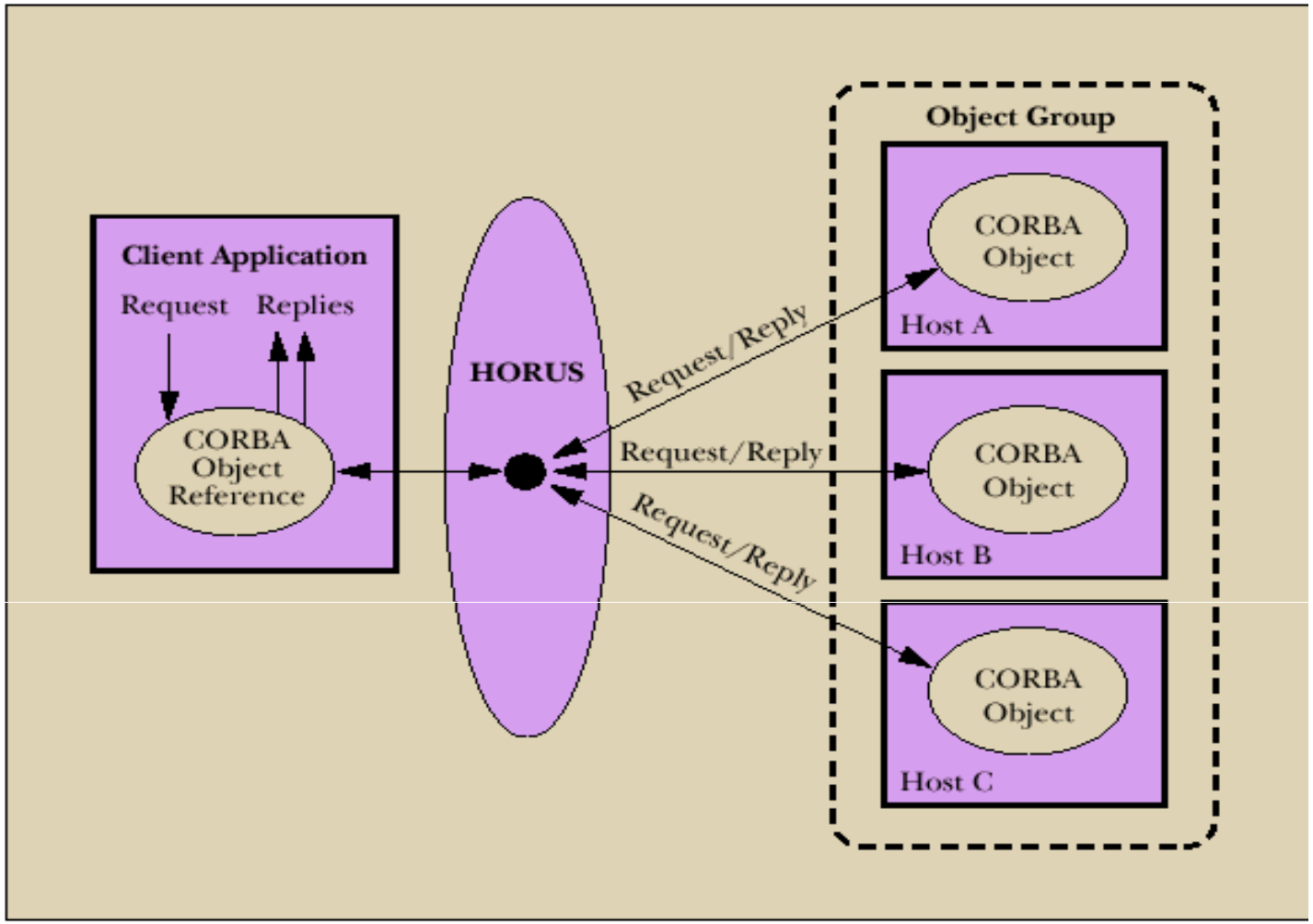


Figure 5. Object group communication in Electra

Transis

A Group Communication Subsystem

Transis : Group Communication System

- Network partitions and recovery tools.
  - Multiple disconnected components in the network operate autonomously.
  - Merge these components upon recovery.
- Hierachical communication structure.
- Fast cluster communication.

Systems that depend on primary component:

- Isis System: Designate 1 component as primary and shuts down non-primary.
  - Period before partition detected, non-primaries can continue to operate.
  - Operations are inconsistent with primary
- Trans/Total System and Amoeba:
  - Allow continued operations
  - Inconsistent Operations may occur in different parts of the system.
  - Don’t provide recovery mechanism

Group Service

- Work of the collection of group modules.
- Manager of group messages and group views
- A group module maintains
  - Local View: List of currently connected and operational participants
  - Hidden View: Like local view, indicated the view has failed but may have formed in another part of the system.

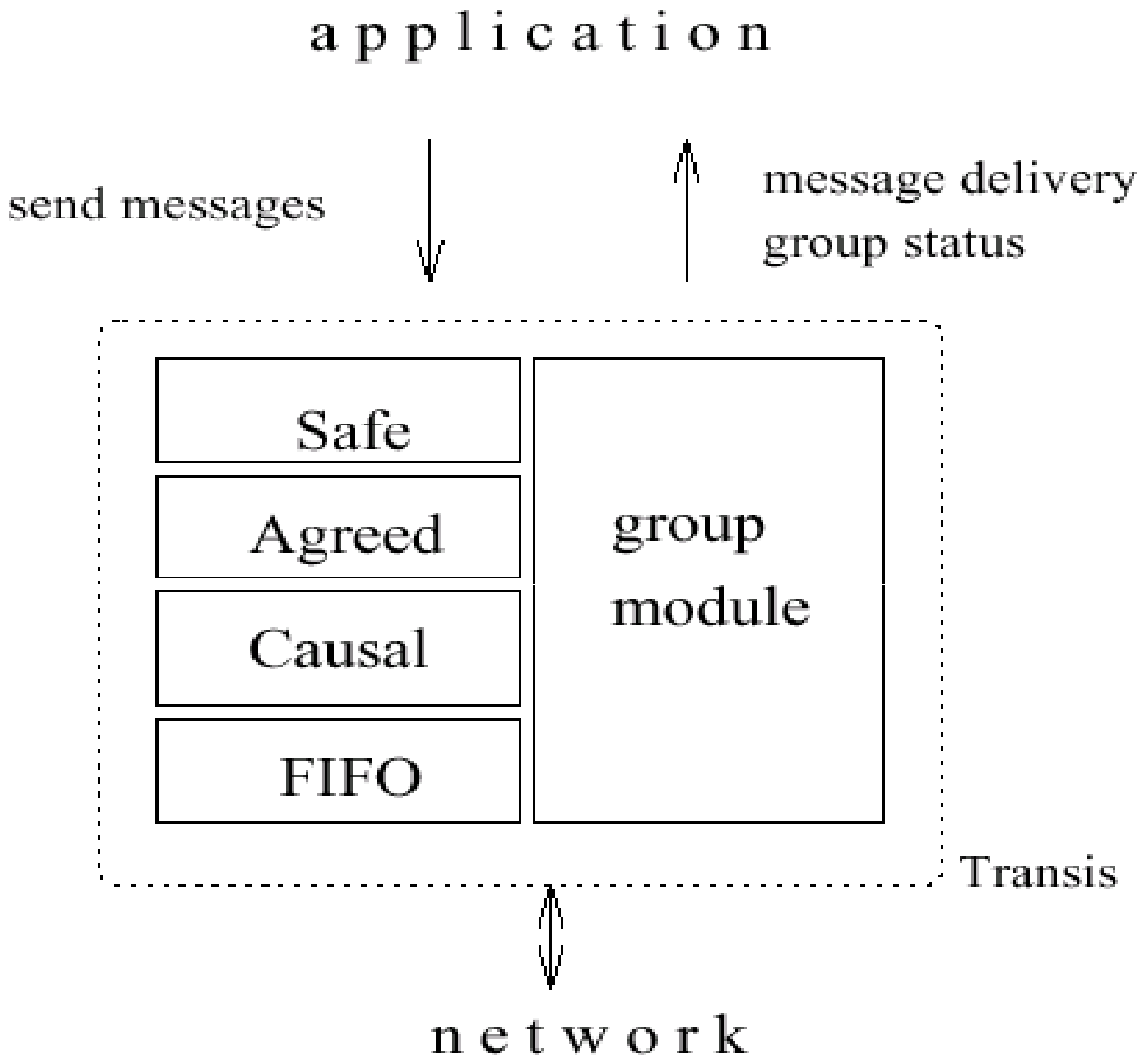


Figure 1: The System Model Structure

Network partition wishlist

1. At least one component of the network should be able to continue making updates.
2. Each machine should know about the update messages that reached all of the other machines before they were disconnected.
3. Upon recovery, only the missing messages should be exchanged to bring the machines back into a consistent state.

Transis supports partition

- Not all applications progress is dependent on a primary component.
- In Transis, local views can be merged efficiently.
  - Representative replays messages upon merging.
- Support recovering a primary component.
  - Non-primary can remain operational and wait to merge with primary
  - Non-primary can generate a new primary if it is lost.
    - Members can totally-order past view changes events. Recover possible loss.
    - Transis report Hidden-views.

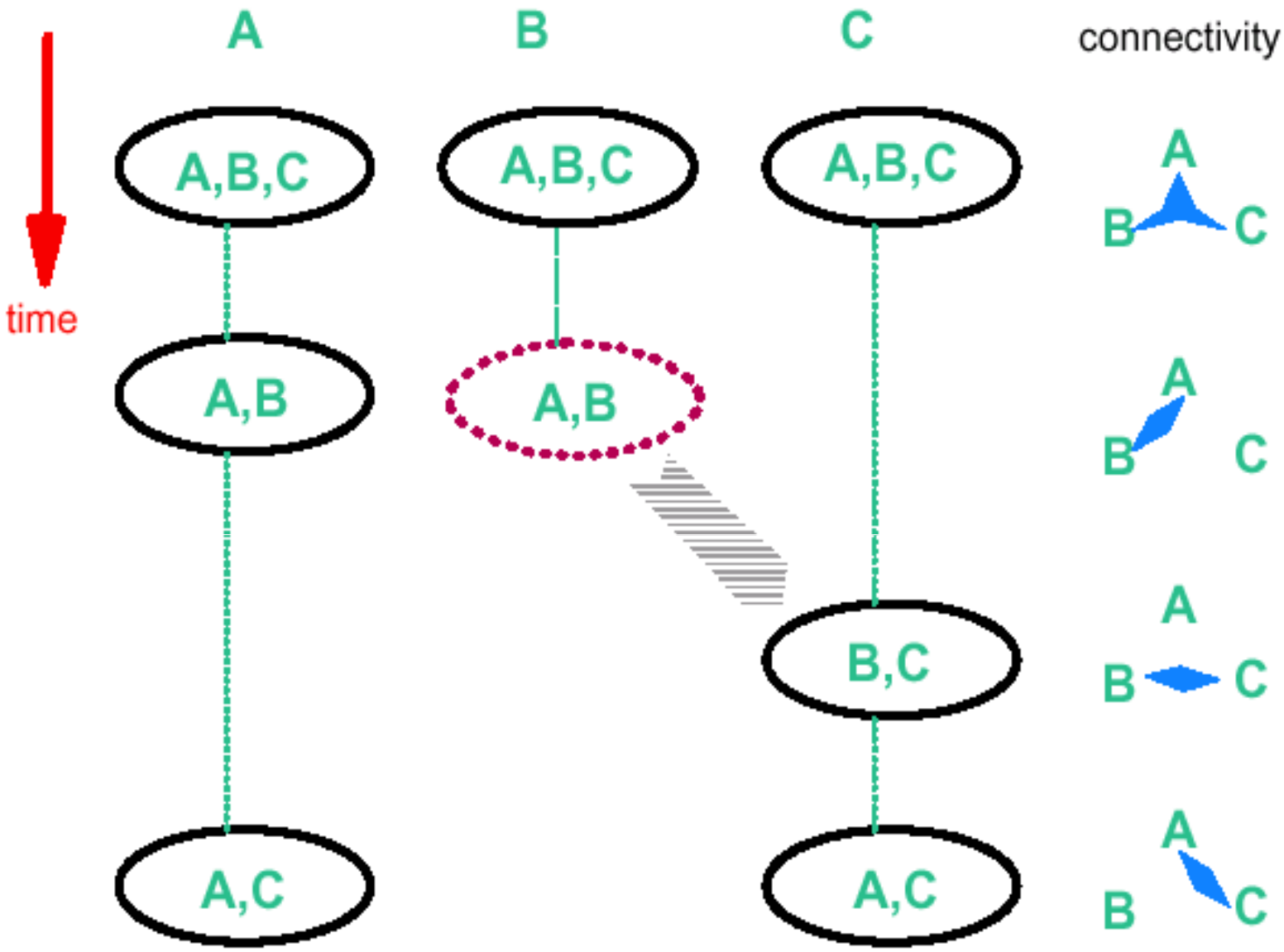


Figure 3: Breaking the symmetry between A and C

Hierarchical Broadcast

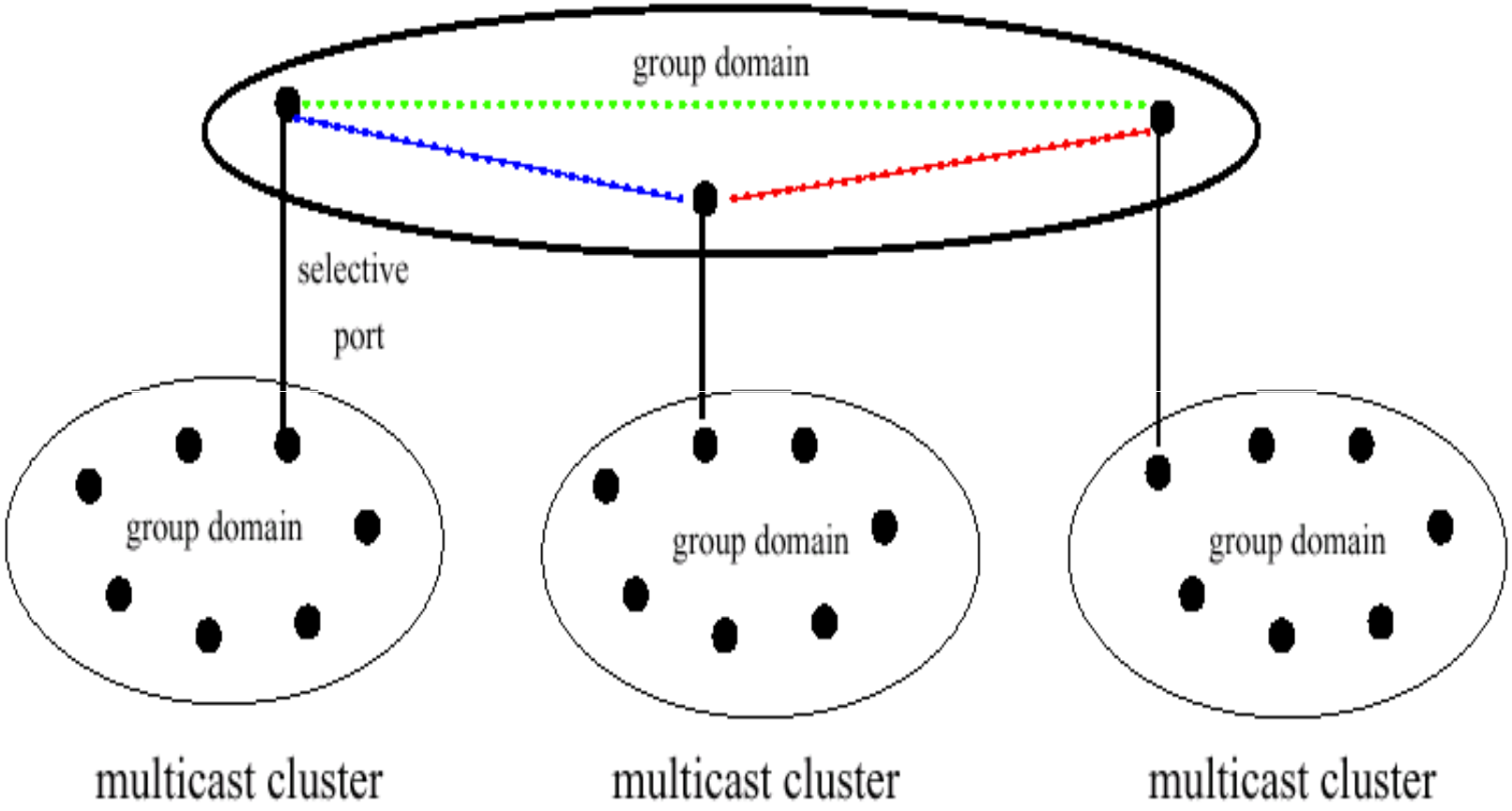


Figure 4: The Transis Communication Model

Reliable Multicast Engine

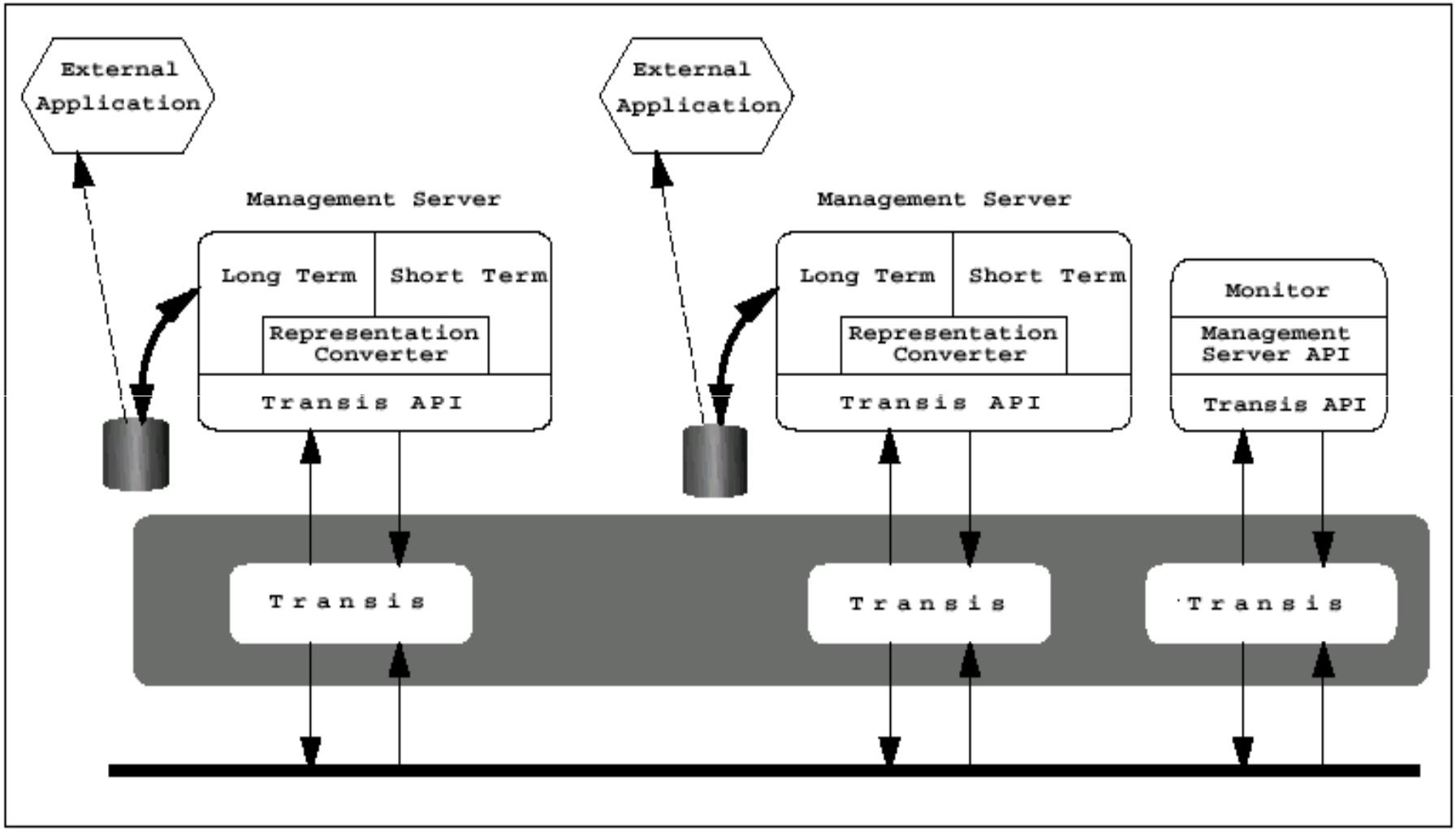
- In system that do not lose messages often
  - Use negative-ack
    - Messages not retransmitted
    - Positive ack are piggybacked into regular mesg
- Detection of lost messages detected ASAP

$$A.1\ ;\ \overset{A.1}{\hookrightarrow} B.1\ ;\ \overset{B.1}{\hookrightarrow} C.1\ ;\ \overset{C.1}{\hookrightarrow} D.1\ ;\ \ldots$$

- Under high network traffic, network and underlying protocol is driven to high loss rate.

Group Communication as an Infrastructure for Distributed System Management

- Table Management
  - User accounts, network tables
- Software Installation and Version Control
  - Speed up installation, minimize latency and network load during installation
- Simultaneous Execution
  - Invoke same commands on several machines


External Application
External Application
Management Server
Management Server
Long Term
Short Term
Representation Converter
Transis API
Long Term
Short Term
Representation Converter
Transis API
Monitor
Management Server API
Transis API
Transis
Transis
Transis

Management Server API

- Status: Return status of server and its host machines
- Chdir: Change the server's working directory
- Simex: Execute a command simultaneously
- Siminist: Install a software package
- Update-map: Update map while preserving consistency between replicas
- Query-map: Retrieve information from the map
- Exit: Terminate the management server process.

Simultaneous Execution

- Identical management command on many machines.
  - Activate a daemon, run a script
- Management Server maintains
  - Set M: most recent membership of the group reported by transis
  - Set NR: set of currently connected servers not yet reported the outcome of a command execution to the monitor

```
Initially:
   CONNECT to Transis;
   JOIN private group;
   JOIN group Cluster;
while(true) {
   m = RECEIVE();
   switch(m.type)
      case command from a user:
         NR = M;
         MULTICAST(command, Cluster);
         while(NR ≠ ∅)
            m = RECEIVE();
            switch(m.type)
               case view change message:
                  NR = NR \ (M \ m.M);
                  M = m.M;
               case result of execution from server:
                  NR = NR \ server;
         return the result;
      case view change message:
         M = m.M;
}
```

command can be one of the following:
Chdir, Status, Simex or Exit

(a) The Monitor

```
Initially:
   CONNECT to Transis;
   JOIN private group;
   JOIN group Cluster;
while(true) {
   m = RECEIVE();
   switch(m.type)
      case Chdir(dir) from the monitor M:
         contexts[M].working_dir = dir;
         send ACK to M's private group;
      case Status from the monitor M:
         get status of my machine;
         convert status to a system-independent form;
         send status to M's private group;
      case Simex(command) from the monitor M:
         convert command to a system-specific form;
         chdir(contexts[M].working dir);
         result = execute(command);
         convert result to a system-independent form;
         send result to M's private group;
      case Exit:
         terminate my process;
}
```

(b) The Management Server

Software Installation

- Transis disseminate files to group members.
  - Monitor multicasts a msg advertising
    - package P
    - set of installation requirements Rp
    - installation multicast group Gp
    - target list Tp.
  - Management server joins Gp if belongs to Rp and Tp.
  - Status of all Management server reported to Monitor
- Use technique in "Simultaneous Execution" to execute installation commands.

Table Management

- Consistent management of replicated network tables.
- Servers sharing replicas of tables form ***Service Group***
- 1 Primary Server
  - Enforces total order of update mesg
  - If network partition, one component (containing Primary) can perform updates

Questions…

- Could provide tolerance for malicious intrusion
  - Many mechanisms for enforcing security policy in distributed systems rely on trusted nodes
  - While no single node need to be fully trusted, the function performed by the group can be
- Problems
  - Network partitions and re-merges
  - Messages omissions and delays
  - Communication primitives available in distributed systems are too weak (*i.e.* there is no guarantee regarding ordering or reliability)
- How can we achieve group communication ?
  - Extending point-to-point networks

From Group Communication to Transactions…

- Adequate group communication can support a specific class of transactions in asynchronous distributed systems
- Transaction is a sequence of operations on objects (or on data) that satisfies
  - Atomicity
  - Permanence
  - Ordering
- Group for fault-tolerance
  - Share common state
  - Update of the common state requires
    - Delivery permanence (majority agreement)
    - All-or-none delivery (multicast to multiple groups)
    - Ordered delivery (serializability of multiples groups)
- Transactions-based on group communication primitives represents an important step toward extending the power and generality of GComm