CSCI 5673
Distributed Systems

Lecture Set Six

TOTEM:
A FAULT-TOLERANT MULTICAST GROUP COMMUNICATION SYSTEM

Lecture Notes by
Shivakant Mishra
Computer Science, CU Boulder

Last update: February 21, 2017

Introduction

- Totem provides *reliable totally-ordered multicasting* of messages over LANs
- Intended for complex applications with critical requirements for
  - *fault tolerance*
  - *real-time performance*
- Exploits *hardware broadcast* of most LANs

Introduction

- The Totem single-ring protocol
  - Supports *high-performance fault-tolerance distributed systems* that continue to operate despite network partitioning and remerging, and processors fail and restart.
  - Provides *totally ordered message delivery* with low overhead, high throughput and low latency using a logical token-passing ring.
  - Provides rapid detection of network partitioning and processor failure together with reconfiguration and membership services.

Totem Services

- Built as a hierarchy of protocols:

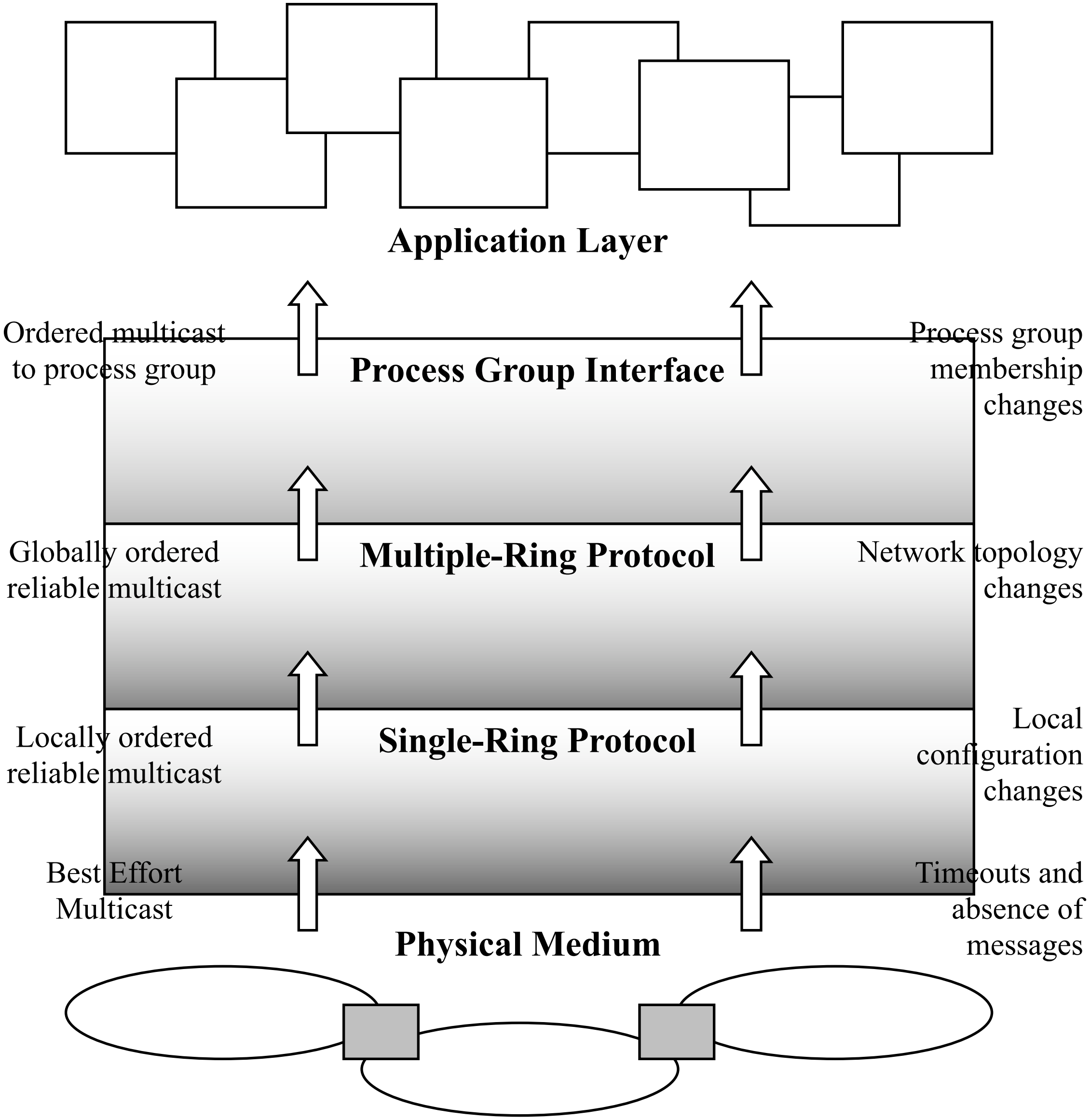
Application layer

Process group interface

Multiple-ring protocol

Single-ring protocol

Physical medium


Application Layer
Ordered multicast to process group
Process Group Interface
Process group membership changes
Globally ordered reliable multicast
Multiple-Ring Protocol
Network topology changes
Locally ordered reliable multicast
Single-Ring Protocol
Local configuration changes
Best Effort Multicast
Physical Medium
Timeouts and absence of messages

Totem: High Level Description

- Configuration and configuration change
- Message originate and delivered in a configuration

Model

- Distributed system is built on a broadcast domain consisting of finite number of processors that communicated by broadcast messages.
- *Originate*: the first broadcast message generated by the application
- To achieve reliability, a message is retransmitted. A processor receives all of its own broadcast messages.
- Broadcast domain can be partitioned in to components.
- Each processor has its own identifier and stable storage. If the processor fails and restarts, its id does not change and its states (all/partial) may have been retained in the stable storage

Model

- Processors are logically arranged into a *ring:*
  - Each ring has a *representative,* and *identifier* that consists of ring sequence number and the identifier of the representative
  - Ring refers to the infrastructure of the Totem
- *Configuration* is the view provided to the application
  - *Membership* of a configuration is a set of processor identifiers.
  - *Regular* configuration has the same membership and identifier as its corresponding ring.
  - *Transitional* configuration consists of processors that are members of a new ring coming directly from the old ring.
- *Token* controls access to the ring; only processor that has possession of the token can broadcast a message.

Membership Services

- Uniqueness of Configuration
- Consensus
- Termination
- Configuration Change Consistency

Reliable Ordered Delivery Services

- Reliable Delivery for Configuration C
  - each message has unique identifier
  - if processor *p* delivers message *m*, *p* delivers *m* once only. If *p* delivers two different messages, the *p* delivers 1 of those messages *strictly before* it delivers the other.
  - if p originates message m, then p will deliver m or will fail before delivering a Configuration Change message to install a new regular configuration
  - if p is a member of regular configuration C, and no configuration change occurs, then p will deliver in C all the messages originated in C
  - if p delivers message m originated in C, then p is a member of C
  - if p and q are both members of configurations C1 and C2 then p and q deliver the same set of messages in C1 before delivering a Configuration Change message that terminates C1 and starts C2.

Reliable Ordered Delivery Services

- Delivery in Causal Delivery for Configuration C
  - delivery order should respect Lamport causality within a configuration
- Delivery in Agreed Order
  - guarantees that processors deliver messages in a consistent total order. When a processor delivers a message, it has delivered all preceding messages in the same total order
- Delivery in Safe Order
  - When processor delivers a message, it has determined that every processor in the current configuration has received the message and will deliver that message unless that processor fails.

Virtual Synchrony

- all processors agree on group membership
- ensures that configuration change occurs at the same point in the message delivery history for all operational processors.
- processors that are members of two successive configurations must deliver the *same* set of messages in the first configuration
- failures do not result in incomplete delivery of messages
- if the system partitions, only processors in the primary component continue to operate

Extended Virtual Synchrony

- Totem extends the virtual synchrony model to systems:
  - all components of a partitioned system continue to operate and subsequently remerge
  - failed processor can be repaired and can rejoin the system with stable storage intact.
- Two processors can deliver different set of messages, but they must not deliver messages in inconsistent order. i.e. p delivers m1 before m2, q must not deliver m2 before m1.
- Delivery in agreed order or safe order as requested by the originator of the message
  - if processor p deliver message m as safe in configuration C, then every processor in C has received m and delivered m before it installs a new configuration, unless that configuration fails.
  - This is achieved by installing a transitional configuration to deliver any remaining messages from the prior configuration.

Atomic Multicast

- Guarantee that a message is delivered either to all the processes or none at all
- Requires that all messages are delivered in the same order to all the processes.

Total Ordering Protocol

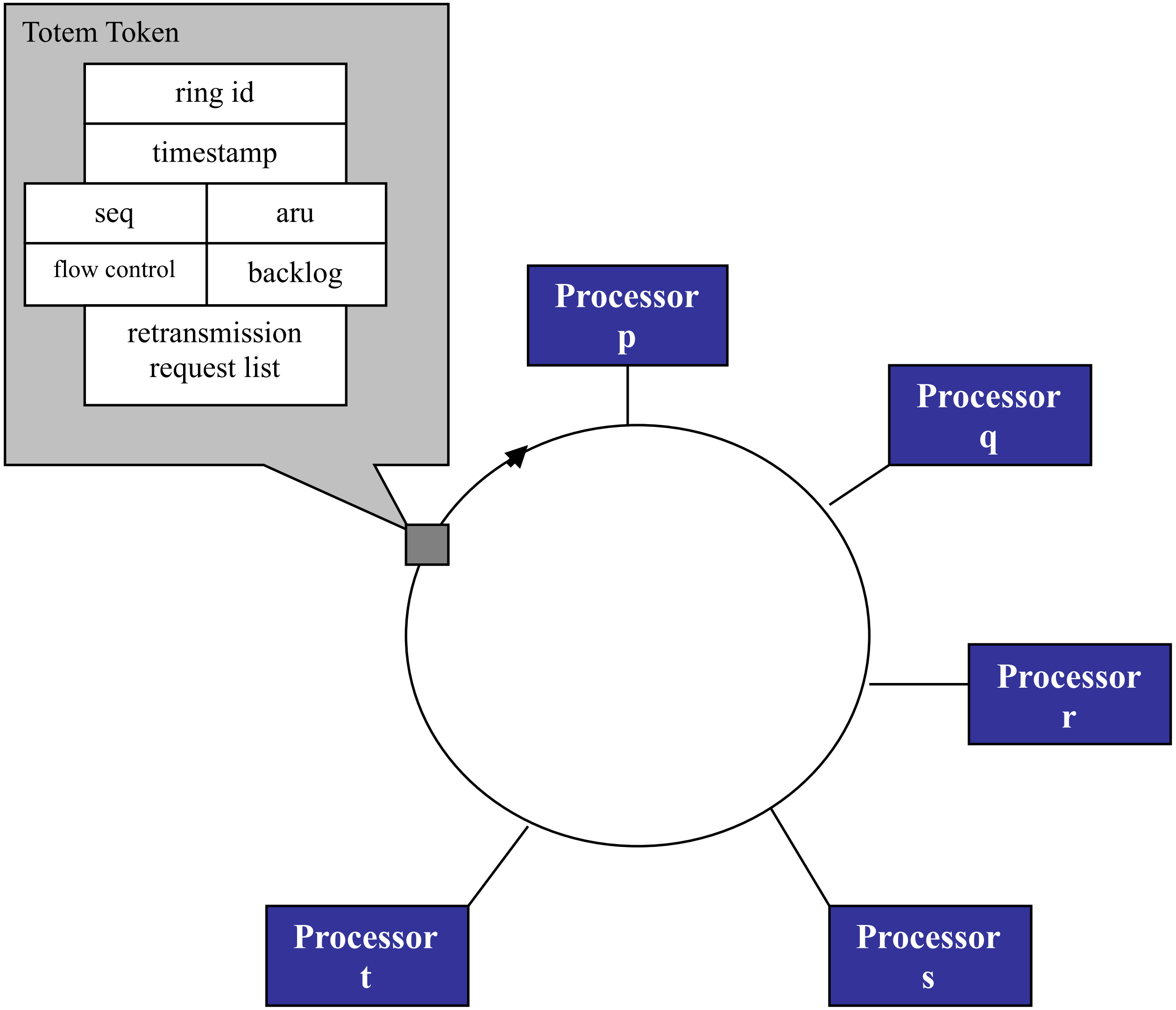
- When processor gets a hold of the token, it can broadcast one or more messages
- Each message header contains a sequence number derived from a field of the token. Hence, there is a single sequence of message sequence numbers for all the processors in the ring.
- Delivery of messages in sequence number order is agreed delivery
- The token has an additional field aru (all-receive-up-to) which is used in safe delivery to determine when all the processors in the ring have received a message.

Total Ordering Protocol – cont’

- The total ordering protocol is unable to continue when the token is lost.
- A Token Retransmission Timeout is used to detect such loss. On a timeout, the processor retransmits the token to the next processor.
- There is mechanism to detect redundant tokens.
- The Membership Protocol handles the loss of all copies of the token

Message Losses

- Loss of token: Positive Ack mechanism
- Other processes are able to detect gaps in the messages they receive and may request the retransmission of certain messages (Negative Ack mechanism)
- The retransmission request list for messages is also within the token.


Totem Token
ring id
timestamp
seq
aru
flow control
backlog
retransmission request list
Processor p
Processor q
Processor r
Processor s
Processor t

Membership Protocol

- resolves processor failure, network partitioning, and loss of all copies of the token.
- detects such failures and reconstructs a new ring on which the total ordering protocol can resume operation
- ensures consensus
- generates a new token and recovers messages that had not been received by some of the processors when the failure occurred.

Membership Protocol

- The protocol attempts to form as large a membership as possible through *consensus* and *termination*:
  - **Consensus** ensures every member in a configuration agrees on the membership of that configuration.
  - **Termination** ensures “every processor installs some configuration with an agreed membership within a bounded time unless it fails within that time”.
- This is possible through Totem’s use of an unreliable failure detector, which must exclude some slow processes, as they are indistinguishable from failed ones.
- With a change in membership detected, the membership protocol constructs a new ring and reaches a new consensus.
- Two *Configuration Change* messages are then sent out to ensure an accurate transition from old to new configuration is achieved.

Membership Protocol

- Join Message
  - contains set of identifiers of the processors that the sender is considering for membership in a new ring – proc_set
  - contains a set of identifier of failed processors – fail_set
  - processor broadcast Join message to achieve consensus
- Configuration Change Message
  - processor delivers this message directly to the application
  - describes a change from an old configuration to a transitional configuration, or from a transitional configuration to a new configuration

Membership Protocol

- Commit Token
  - sent out by the representative of the ring to circulate in the new ring
  - has a *member_list* field containing the membership of the new ring
  - each processor in the member_list has a *highest_delivered* field which indicates the largest sequence number of a message that the process has delivered on the old ring.
  - first rotation of the Commit Token is used to collect information needed to determine correct handling of messages from old ring.

Operations of the Membership Protocol - handling failure

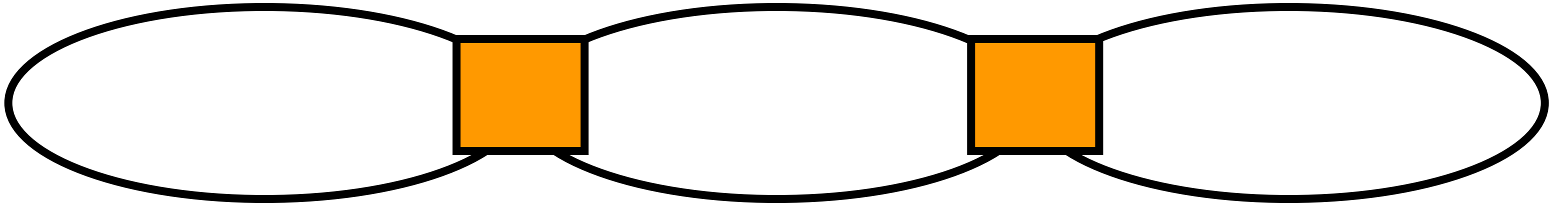
- Mechanism to detect failures is the Token Loss timeout
- When an Token timeout expires, or a Join message is received, the processor invokes the protocol for the formation of a new ring.
- Processor collects info about operational processors and failed processors and broadcast that info in the proc_set and fail_set fileds of a Join message.
- Upon receive the Join Message, processor updates the its my_proc_set and m_fail_set. *If these are changed*, it broadcasts a Join message with the updated sets.
- Consensus is met when my_proc_set and my_fail_set are equal to proc_set and fail_set of all Join messages from every processor.

Operations of the Membership Protocol

- The representative of the ring is the processor that has the lowest identifier. The representative then broadcasts a Commit Token to collect needed information to determine correct handling of messages from the old ring
- When the Commit Token are done circulating twice, a new ring is formed but not yet installed.
- The recovery protocol is then executed to retransmit messages from their old ring that must be exchanged to maintain agreed and safe delivery.
- In one atomic action, each processor delivers the exchanged messages to the application along with the Configuration Change message. The new ring is installed. Message delivery is then resumed.

Additional Issues

- Multiple Ring Protocol



- Partition Merging