Fault-Tolerant Broadcasts and Related Problems

Chapter 5

Distributed Computing Systems
Chapter 5 : Contents

• Introduction
• Models of Distributed Computation
  – Failure Models
• Broadcast Specifications
  – Reliable, FIFO, Causal, Atomic, Uniform
• Broadcast Algorithms
• Consensus
• Complexity Analysis
Introduction

• Consensus: a common decision
• Reliable broadcast messages
  – Agreement on the set of messages they deliver
  – Agreement on the order of message deliveries
  – Tolerating possible (benign) failures
System Models

• Models of distributed computation
  – message-passing vs. shared-memory

• Message passing models
  – process / communication attributes
  – Synchronous
    • message delivery is bound by $\delta$
    • Clock drift bound by $\rho$

\[
(1 + \rho)^{-1} \leq \frac{C_p(t) - C_p(t')}{t - t'} \leq (1 + \rho)
\]
System Models

• Clocks between two processes
  – Approximately synchronized
    • If there exists \( \varepsilon \) for Every \( t, (p,q) \) such that
      \[
      \left| C_p(t) - C_q(t') \right| \leq \varepsilon
      \]
  – Perfectly synchronized
    • \( \varepsilon = 0 \)
    • theoretically useful
System Models

• Message send / receive / delivery
Failure Models

• Process Failures
  – Crash (fail-stop)
    • A faulty process stops and does nothing
  – Send omission
    • Crash or omits to send message
  – Receive omission
    • Crash or omits to receive message
  – general omission
Failure Models
(Process Failures)

– Arbitrary (Byzantine failure)
  • A faulty process can exhibit any behavior
  • “Malicious”

– Arbitrary with message authentication
  • Arbitrary but message sender is unforgeable
Failure Models
(Process Failures)

– Timing Failures:
  • only in synchronous systems
  • one or more ways of
    – general omission
    – clock drift exceeds the specified bound
    – performance failure (runs slower than specified)
  • More severe than general omission, not than arbitrary failure

• **Benign** failures: no more severe than timing failures
Failure Models

• Communication Failures
  – Crash
    • A link stops transporting messages
  – Omission
    • Intermittent omissions
  – Arbitrary
  – Timing failures
    • A faulty link transports messages faster or slower than specified
System Models

• Network topology
  – point-to-point
  – broadcast networks
  – unable to tolerate network partition
    • must have sufficient connectivity

• Determinism vs. Randomization
  – State transitions determined uniquely
Broadcast Specifications

: Introduction

• Reliable Broadcast
  – Validity :
    • all messages broadcast by correct processes are delivered eventually
  – Agreement :
    • all the current processes agree on the set of messages they deliver
  – Integrity :
    • no spurious messages are ever delivered
  – no specification on the order
Broadcast Specifications

: Introduction

- **FIFO Broadcast**
  - Reliable + msgs from the same sender are delivered in FIFO order

- **Causal Broadcast**
  - FIFO + delivered in causal order (partial)

- **Atomic Broadcast**
  - Every processes deliver all msgs in the same order
  - FIFO Atomic / Causal Atomic
Reliable Broadcast

- Reliable bcast properties
  - **Validity**
    - If a correct process broadcasts a msg $m$, then all correct processes eventually deliver $m$
  - **Agreement**
    - If a correct process delivers a msg $m$, then all correct processes eventually deliver $m$
  - **Integrity**
    - For any message $m$, every correct process delivers $m$ at most once, and only if $m$ was previously broadcasted by $sender(m)$
FIFO Broadcast

- FIFO Bcast := Reliable + FIFO
  - FIFO Order:
    - If a process broadcasts a message $m$ before it broadcasts a message $m'$, then no correct process delivers $m'$ unless it has previously delivered $m$
Causal Broadcast

• Preserve causal dependency between messages
  – ‘happens before’ relation

• Causal Broadcast := Reliable + Causal
  – Causal Order
  • If the broadcast of a message $m$ causally precedes the broadcast of a message $m'$, then no correct process delivers $m'$ unless it has previously delivered $m$
Atomic Broadcast

• Atomic := Reliable + Total order
  – Total Order
    • If correct processes $p$ and $q$ both deliver messages $m$ and $m'$, then $p$ delivers $m$ before $m'$ if and only if $q$ delivers $m$ before $m'$
  – All messages delivered in the same order
    • What kind of “order”?  
      – FIFO Atomic  
      – Causal Atomic
Timed Broadcasts

• \(\Delta\)-Timeliness
  – Delivered within a bounded time \(\Delta\) (latency)
  – (Real-Time) \(\Delta\)-Timeliness
    • There is a known constant \(\Delta\) such that if the broadcast of \(m\) is initiated at real-time \(t\), no correct process delivers \(m\) after real-time \(t+\Delta\)
  – (Local-Time) \(\Delta\)-Timeliness
    • There is a known constant \(\Delta\) such that no correct process \(p\) delivers a message \(m\) after local time \(ts(m)+\Delta\) on \(p\)’s clock
Uniform Broadcasts

- Previous properties did not mention faulty processes
- "Uniform" regardless of whether they’re correct or faulty
  - Uniform agreement / integrity / timeliness
- Uniform FIFO / Uniform Causal / Uniform Total Order
Properties of Broadcast

- Inconsistency and Contamination
  - Refuse to deliver inconsistent broadcast messages

- Amplification of failures
  - Amplification of failures in lower level primitives
Relationship among Broadcast Primitives

- Reliable Broadcast
- Atomic Broadcast
- FIFO Broadcast
- FIFO Atomic Broadcast
- Causal Broadcast
- Causal Atomic Broadcast

Order Relations:
- Total Order
- FIFO Order
- Causal Order
Broadcast Algorithms

• send(m), receive(m) primitives
  – validity : If $p$ sends $m$ to $q$, and both $p$ and $q$ and the link between them are correct, then $q$ eventually receives $m$
  – Uniform Integrity : For any message $m$, $q$ receives $m$ at most once from $p$, and only if $p$ previously sent $m$ to $q$
Reliable Broadcast

- **Broadcast(R, m)**
  
tag $m$ with $sender(m)$ and $seq\#(m)$
  
send($m$) to all neighbors including $p$

- **Deliver(R, m)**
  
upon receive($m$) do
  
if $p$ has not previously executed deliver(R,m) then
  
  if $sender(m) \neq p$ then send($m$) to all neighbors deliver(R,m)

- **Uniform** if only receive omission occurs
Timed Reliable Broadcast

• Assumptions
  – At most $f$ processes can fail
  – Every two correct processes are connected via a path length $< d$
  – Message delay $< \delta$

• Real-Time Timeliness
  – Previous RB algorithm is RT with
    $\Delta = (f + d)\delta$
Timed Reliable Broadcast

• Local-Time Timeliness
  – Assume
    • Clocks synchronized within $\varepsilon$
    • Clock drift $< \rho$
  – Modified RB
    tag msg with local clock & hop-count (k) upon receive(m)
    discard m if $t' - ts(m) > k(\varepsilon + \delta(1+\rho))$
  – This is LT-Timed RB with
    • $\Delta = (f + d)\delta (1+\rho) + (f+1)\varepsilon$
FIFO / Causal Broadcast

• FIFO Bcast
  – Maintain next[q] for each sender
  – Check seq# before delivery

• Causal Bcast
  – Augment msg with previous C-delivered msgs since previous C-bcast
  – Check causality before delivery
Atomic Broadcast

• There are no deterministic Atomic Bcast algorithms for asynchronous systems
  \( \Theta \) No deterministic consensus algorithm

• Timed Atomic Bcast
  – Modify Timed R-B to agree on delivery order
    • schedule deliver\((m)\) at ts\((m) + \Delta\)
Terminating Reliable Broadcast

• a priori knowledge of next broadcast
  – SF(Sender Fail) message to notify failure
  – Termination
    • every correct process eventually delivers some message
  – Integrity
    • Every process delivers at most one message, and if it delivers \( m \neq \text{SF} \) then the sender must have Broadcast(m)
Consensus

• \textit{propose}(v) \& \textit{decide}(v)

• Properties
  – termination: every correct processes eventually decide
  – validity: if a process \textit{propose} v, then all correct processes decide v
  – agreement: If a correct process \textit{decides} v, then all correct processes decide v
Relations among Problems

• Problem reduction
  – Terminating RB == Consensus on synchronous systems
  – Terminating RB => Consensus, not the reverse, holds on asynchronous systems

• Consensus using TRB
  – propose : use TRB to broadcast \( v \)
  – decide : decide first value in \( v \)

• TRB using Consensus
  – broadcast : send msg with \( ts(m) = t \)
  – delivery : consensus on delivery whether \( ts(m) = t + \delta \)
Complexity Results

• Fault-Tolerance
  – Lower bound $n$ to tolerate $f$ failures?
  – Arbitrary failures: $n > 3f$
    • There is no algorithm that tolerates arbitrary failures if $n \leq 3f$
  – Arbitrary with authentication: $n > 2f$
  – Benign failures: $n > f$
Complexity Results

• Time Complexity
  – a round := send + receive + state change
  – $f + 1$ rounds on worst case
  – $f + d$ rounds on link failures (max. $d$ links)

• Message Complexity
  – Arbitrary failures : $O(nxf)$ on worst case
  – General omissions : $O(n+f^2)$
  – Crash failures : $O(n+f \log f)$