CMPSCI677 Operating Systems Spring 2015

Lecture 18: April 1

Dr. Prashant Shenoy *Scribe: Xiaolan Wang*

18 Fault Tolerance

18.1 Agreement in Faulty Systems

Problem: How should the processes agree on the result of a computation. For example, there are multiple replicas with faulty information, figure out the right output based on the input. The computation can be arbitrary, e.g., database query.
K-fault tolerance: system can survive k faults and yet function.
`Process fail silently:` Crash fault is an example of process fail silently. As long as there is output, the output is guaranteed correct. Thus, for crash fault, there should be at least $(k + 1)$ redundancy to tolerant such k faults.
Byzantine Failure: process run even if sick. The key property of byzantine failure is the process will produce erroneous, random, or malicious replies $\Rightarrow$ *high degree replications*.

18.2 Byzantine Faults

Simplified Scenario (unreliable channel only): Assume sending 1-bit message.
Two army problem: two armies waiting to attach: each army coordinate with a messenger, who can be captured by hostile army or transfer wrong information. In order to reach agreement, two armies will continuously send messages and waiting for acknowledgement.
Property: two perfect process can never reach agreement in presence of unreliable channel.
Byzantine General Problem: Relax the faulty type with good network.
Question: can N generals reach agreement with a perfect channel?
Recursive Algorithm by Lamport:
Round 1: broadcast information of itself.
Round 2: broadcast information received from previous round.
Example: assume there are four general wish to reach agreement with one general sending faulty message. Using Lamport algorithm, the other three generals will found who is lying.
Reaching agreement: assume there are k faults, we need $3k + 1$ processes to identify the faulty processes.
Example: assume there are three general wish to reach agreement with one general sending faulty message. Using Lamport algorithm, the other two generals will found that someone is lying, but cannot identify the faulty general. This is because there are not enough number of replications.
Detect faulty: assume there are k faults, we need $2k + 1$ processes to detect k faults.

18.3 Reliable One-One Communication

RPC semantics in the presence of failure with possibilities: Client unable to locate server; Lost request messages; Server crashes after receiving request; Lost reply messages; Client crashes after sending request.

18.4 Reliable One-Many Communication

To maintain reliable multicast, when lost message, retransmit is required. With ACK-based schemes, sender becomes the bottleneck since it has to deal with all the ACK from others. In order to resolve the problem, use NACK-based (negative ACK) schemes where sender only deal with a small number of negative messages and retransmit messages for those.

18.5 Atomic Multicast

Definition: atomic multicast guarantees that either all processes receive the message or none of them receive the message.
Atomic multicast apply for replicated databases: for example, a transaction of deposit 100 dollars, atomic multicast provides guarantee that either all databases are successfully updated or none of them is updated.
Problem: how to handle process crashes?
Require processes to discard received message and roll back.
Solution: using group view.
Each message is uniquely associated with a group of processes: view of the process group when message was sent; all processes in the group should have the same view and agree on it.
Example: implementing virtual synchrony (Table 18.5)

Multicast	Basic Message Ordering	Total-ordered Delivery?
Reliable multicast	None	No
FIFO multicast	FIFO-ordered delivery	No
Causal multicast	Causal-ordered delivery	No
Atomic multicast	None	Yes
FIFO atomic multicast	FIFO-ordered delivery	Yes
Causal atomic multicast	Causal-ordered delivery	Yes

18.6 Distributed Commit

Atomic multicast is an example of a more general problem, where all processes in a group perform an operation or not at all. Examples of such kind of problem includes reliable multicast with delivery of message as the operation and distributed transaction with commit transaction as the operation.
Approaches include: two phase commit (2PC) and three phase commit.

18.6.1 Two phase commit

Coordinator process coordinates the operation (can use any leader selection algorithm to find coordinator). There are two phases.
Phase 1: voting phase: process vote on whether to commit; Phase 2 (decision phase): actually commit or abort.