CS 5204
Operating Systems
Lecture 8

Godmar Back

Announcements

- Project proposals due today by email
  - Look for reply email from me with word “approved” in it
- No paper evaluation required for Wednesday
  - Will cover E2E paper in lecture
- Out of town Oct 2-5:
  - No class on Oct 3 & 5;
  - Presentations will move back; I’ll update reading list with new tentative dates

Plan for Today

- More on fault tolerance in distributed systems
- Techniques for scalability
- Consistency models
- Importance of Openness & Flexibility

Fault Tolerance Terminology

- Distributed systems should tolerate partial failure
- *Faults* cause *errors* that lead to *failure*
- Aspects of dependability:
  - Availability
  - Reliability
  - Safety
    - Security
  - Maintainability
- Types of faults:
  - Transient vs. intermittent vs. permanent

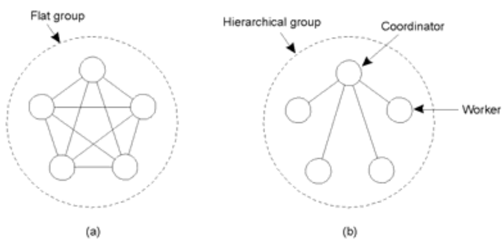
Failure Models

Type of failure	**Description**
Crash failure	A server halts, but is working correctly until it halts *Fail-Stop*: if this can be detected *Fail-Silent*: if not (and others may wrongly believe in crash)
Omission failure *Receive omission* *Send omission*	A server fails to respond to incoming requests A server fails to receive incoming messages A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure *Value failure* *State transition failure*	The server's response is incorrect The value of the response is wrong The server deviates from the correct flow of control
Arbitrary failure (*Byzantine* failures)	A server may produce arbitrary responses at arbitrary times

Types of Redundancy

- Redundancy masks failures
- Information Redundancy
  - Example: Forward Error Correction (FEC)
- Time Redundancy
  - Example: Transactions
- Physical Redundancy
  - Example: Triple Modular Redundancy
- Let’s look at redundancy through replication

Replication Protocols (I)



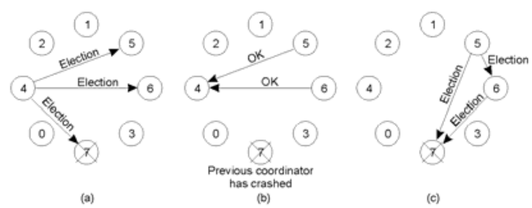
Types of Replication Protocols

- Primary-based protocols
  - hierarchical group
  - generally primary-backup protocols
  - options for primary:
    - may propagate notifications, operations, or state
- Replicated-write protocols
  - flat group using distributed consensus

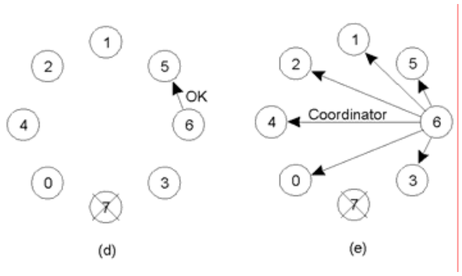
Election Algorithms

- For primary-based protocols, elections are necessary
- Need to select one "coordinator" process/node from several candidates
  - All processes should agree
- Use logical numbering of nodes
  - Coordinator is the one with highest number
- Two brief examples:
  - Bully algorithm
  - Ring algorithm

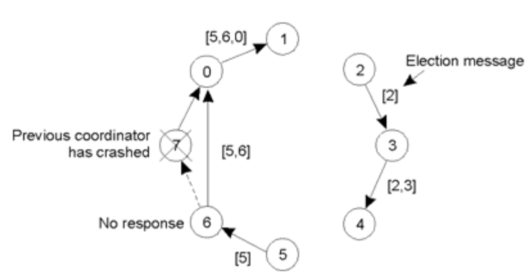
Bully Algorithm (1)



Bully Algorithm (2)



Ring Algorithm



K fault tolerance (in Repl. Write)

- A system is *k-fault tolerant* if it can tolerate faults in k components and still function
- Many results known; strongly dependent on assumptions
  - Whether message delivery is reliable
  - Whether there is a bound on msg delivery time
  - Whether processes can crash and in what way:
    - Fail-stop, silent-fail or Byzantine

Reliable Communication

- Critical: without it, no agreement is possible
- Aka "Two Army Problem"
  - Two armies must agree on a plan to attack, but don't have reliable messengers to inform each other
  - Assumes that armies are loyal (=processes don't crash)
- Hence all consensus is only as reliable as protocol used to ensure reliable communication
- Aside: in networking, causes corner case on TCP close

Byzantine Agreement in Asynchronous Systems

- *Asynchronous*:
  - Assumes reliable message transport, but with possibly unbounded delay
- Agreement is impossible with even one faulty process. Why?
  - Proof: Fischer/Lynch/Paterson 1985
  - Decision protocol must depend on single message:
    - Delayed: wait for it indefinitely
    - Or declare sender dead: may get wrong result

Alternatives

- Sacrifice safety, guarantee liveness
  - Might fail, but will always reply within bounds
- Guarantee liveness probabilistically
  - "Probabilistic Consensus"
  - See also "weak synchrony" assumption in Castro paper
  - Allow for protocols that may never terminate; but this would happen w/ probability zero
    - E.g., always terminates, but impossible to say when in the face of delay

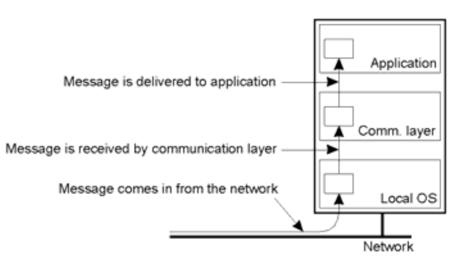
K-Fault Tolerance Conditions

- To provide K-fault tolerance for 1 client with replicated servers
  - Need k+1 replicas if at most k can fail silent
  - Need 2k+1 replicas if up to k byzantine failures and all respond (or fail stop)
  - Need 3k+1 replicas if up to k byzantine failures and replies can be delayed
    - Optimal: both necessary and sufficient

Virtual Synchrony

- Recall: for replication to make sense, all replicas must perform same set of operations
  - Generalization: "replicated state-machines"
- Virtual Synchrony:
  - Definition: A message is either delivered to all processes in a group or to none
- Keyword: *delivered (vs. received)*

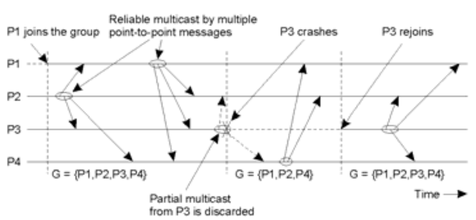
Message Receipt vs. Delivery



Virtual Synchrony (cont'd)

- Observation: ensuring virtual synchrony requires agreement on group membership
- Group membership can change:
  - Processes may leave/crash
  - Processes may join/restart
- Idea: associate message m with current Group View (set of receivers)

Virtual Synchrony & Views



Implementation using distributed protocol relying on reliable, in-order Point-to-point communication as in ISIS (Birman '91)

Atomic Multicast

= Virtual Synchrony + Total-order Delivery

Classification of virtual synchronous reliable multicasting:

Message Ordering	None	FIFO Ordering	Causal Ordering
Not total-ordered	Reliable Multicast	FIFO Multicast	Causal Multicast
Total-ordered	Atomic Multicast	FIFO Atomic Multicast	Causal Atomic Multicast

Summary Fault Tolerance

- Fault tolerance in replicated write systems requires:
  - Distributed consensus
  - Which assumes atomic multicast, which must solve two subproblems
    - Virtually synchronous: same set of msg
    - Totally ordered: same order