CSCI 5105

Instructor: Abhishek Chandra

Today

- Reliable Communication
  - Reliable RPC
  - Reliable Group Communication

RPC: Failures

- Network failures:
  - Unable to locate servers
  - Lost requests/replies
- Server crashes
- Client crashes

RPC: Network failures

- Client unable to locate server:
  - Return error or raise exception
- Lost requests/replies:
  - Timeout mechanisms
  - Make operation idempotent
  - Use sequence numbers, mark retransmissions

RPC Failure Semantics: Server failure

- Server may crash during RPC
  - Did failure occur before or after operation?
- Operation semantics
  - Exactly once: desirable but impossible to achieve
  - At least once
  - At most once
  - No guarantee

5

RPC Failure Semantics: Client Failure

- Client crashes while server is computing
  - Server computation becomes orphan
- Possible actions
  - Extermination: log at client stub and explicitly kill orphans
  - Reincarnation: Divide time into epochs between failures and delete computations from old epochs
  - Expiration: give each RPC a fixed quantum T; explicitly request extensions

6

Reliable Group Communication

- Group of servers
  - Need to multicast messages reliably
- Questions:
  - What does "reliable" mean?
  - What happens when groups change?

7

Reliable Multicast

- Basic Multicast:
  - Sender multicasts message, may reach subset of receivers
- Goal: Message should reach all receivers
- Two cases:
  - When no process fails
  - When a process fails or joins a group

8

Basic Reliable Multicast

- Assume no process failures
- Receivers send an ACK for each message received successfully
- Sender keeps local copy of message
  - Retransmits if hasn't received all ACKs
- What is the problem?

9

Scalability in Reliable Multicast

- Feedback implosion:
  - Large number of receivers => sender becomes overloaded with ACKs
- Reducing number of ACKs
  - Use NACKs
  - Problem?

10

Scalable Feedback Control

- Avoid sending all ACKs/NACKs to sender
- Feedback suppression
  - Receiver multicasts NACK
  - If a receiver sees a NACK, suppresses its own NACK
- How to avoid synchronized NACKs?
  - Use random timers
- How to avoid feedback traffic to reliable hosts?
  - Separate multicast channel for feedback and retransmission
- How to scale up retransmissions?
  - Allow a receiver with message to transmit

11

Hierarchical Feedback Control

- Use tree structure for feedback/retransmissions
  - Receivers divided into small groups
  - Each group has a coordinator
  - Coordinators organized into a multicast tree
- Coordinator keeps track of missed messages and retransmissions for its group
  - Passes these along up the tree

12

Reliable Multicast with Process Failures

- What happens when processes can fail?
  - What if some of the processes received the message, but others did not?
  - What if the sender fails?
  - What if a new process joins the group?

13

Atomic Multicast

- Virtual synchrony: A message is delivered either to all processes of a group or none at all
- Total ordering: Messages are delivered in the same order

14

Virtual Synchrony: Basics

- Message receipt vs. delivery:
  - Receipt: message received at the communication layer
  - Delivery: message delivered to the application
- Group view:
  - Set of processes in the group when sender issued the message
  - Unique for each message m
  - Can differ for different messages
- View change: Change in group membership
  - Node leaving or joining a group

15

Virtual Synchrony

- What happens when a group view changes?
  - Message vc multicast to the group to announce this change
- Can think of two messages:
  - m and vc
  - Need to order these messages
- If sender fails, can either deliver the message to everyone or none at all

16

Message Ordering

- Total ordering: Messages are delivered in the same order to all processes
- What order is it w.r.t. message sending?
  - Unordered: Messages can be delivered in any order
  - FIFO: Messages from same process delivered in order
  - Causal: Causally-ordered messages delivered in order