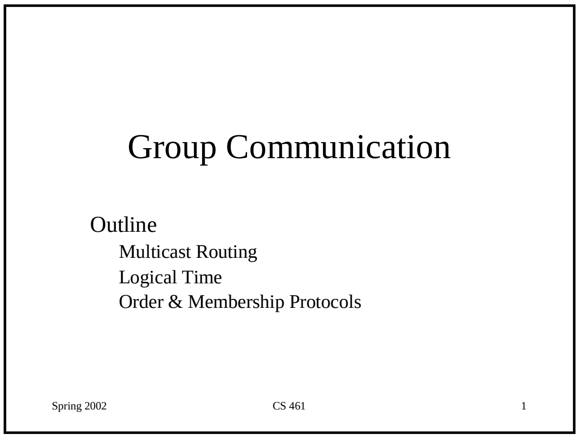
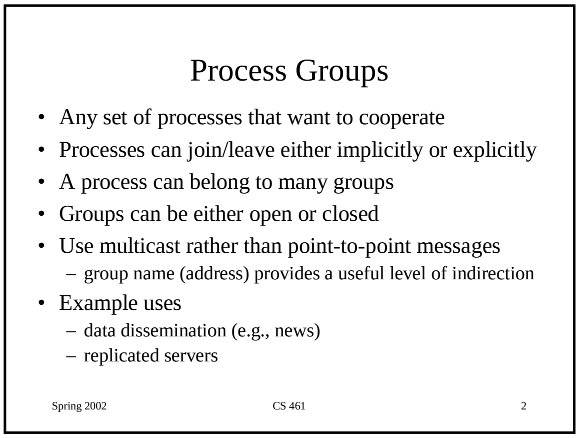


Group Communication

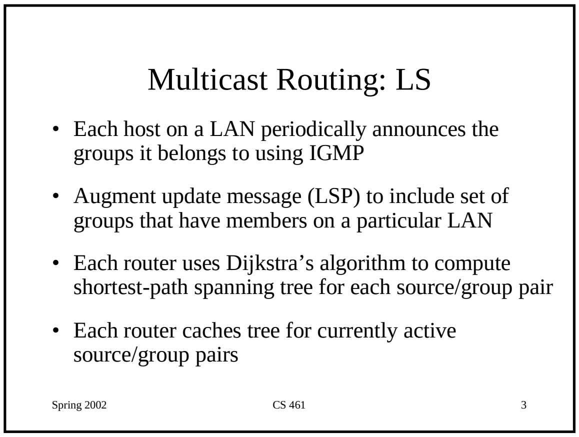
Outline

- Multicast Routing
- Logical Time
- Order & Membership Protocols



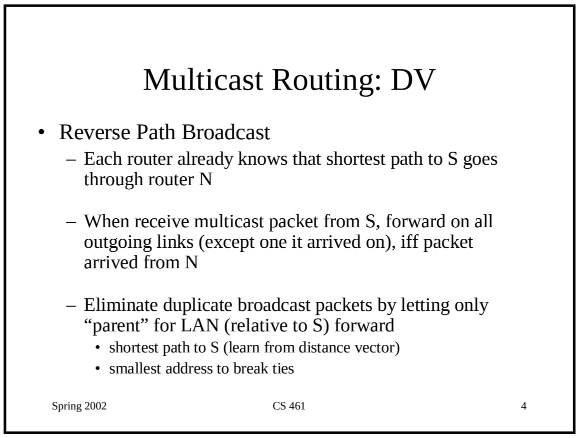
Process Groups

- Any set of processes that want to cooperate
- Processes can join/leave either implicitly or explicitly
- A process can belong to many groups
- Groups can be either open or closed
- Use multicast rather than point-to-point messages
  - group name (address) provides a useful level of indirection
- Example uses
  - data dissemination (e.g., news)
  - replicated servers



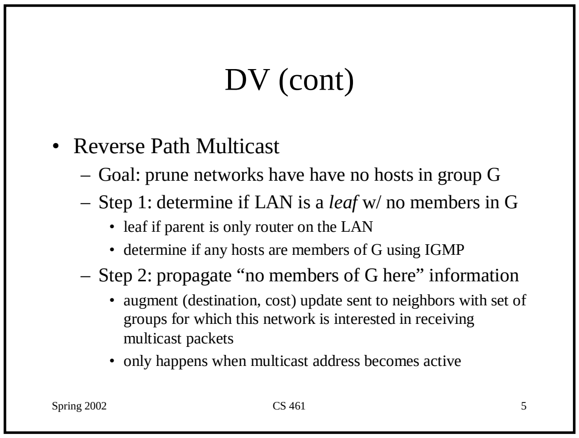
Multicast Routing: LS

- Each host on a LAN periodically announces the groups it belongs to using IGMP
- Augment update message (LSP) to include set of groups that have members on a particular LAN
- Each router uses Dijkstra’s algorithm to compute shortest-path spanning tree for each source/group pair
- Each router caches tree for currently active source/group pairs



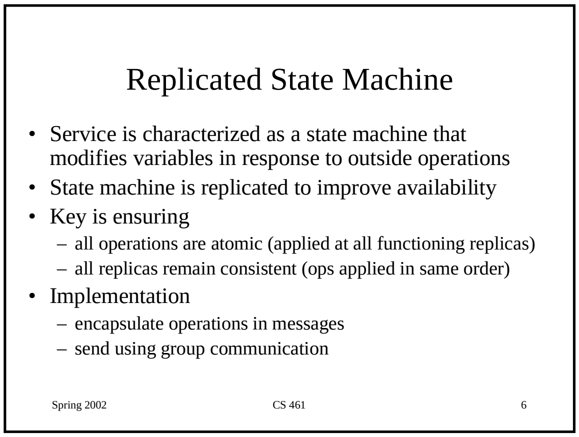
Multicast Routing: DV

- Reverse Path Broadcast
  - Each router already knows that shortest path to S goes through router N
  - When receive multicast packet from S, forward on all outgoing links (except one it arrived on), iff packet arrived from N
  - Eliminate duplicate broadcast packets by letting only “parent” for LAN (relative to S) forward
    - shortest path to S (learn from distance vector)
    - smallest address to break ties



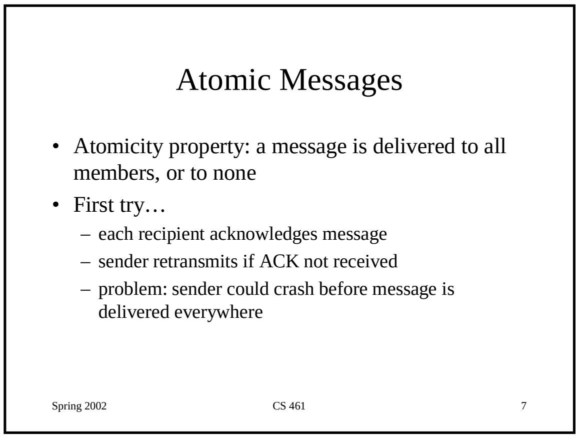
DV (cont)

- Reverse Path Multicast
  - Goal: prune networks have have no hosts in group G
  - Step 1: determine if LAN is a *leaf* w/ no members in G
    - leaf if parent is only router on the LAN
    - determine if any hosts are members of G using IGMP
  - Step 2: propagate “no members of G here” information
    - augment (destination, cost) update sent to neighbors with set of groups for which this network is interested in receiving multicast packets
    - only happens when multicast address becomes active



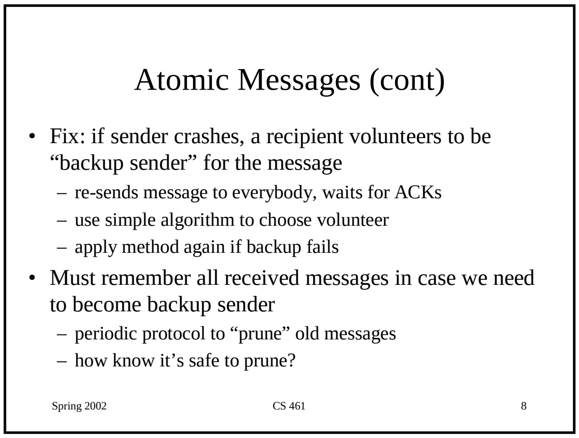
Replicated State Machine

- Service is characterized as a state machine that modifies variables in response to outside operations
- State machine is replicated to improve availability
- Key is ensuring
  - all operations are atomic (applied at all functioning replicas)
  - all replicas remain consistent (ops applied in same order)
- Implementation
  - encapsulate operations in messages
  - send using group communication



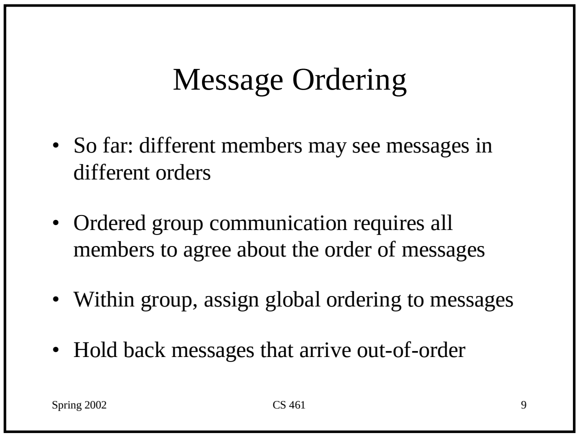
Atomic Messages

- Atomicity property: a message is delivered to all members, or to none
- First try…
  - each recipient acknowledges message
  - sender retransmits if ACK not received
  - problem: sender could crash before message is delivered everywhere



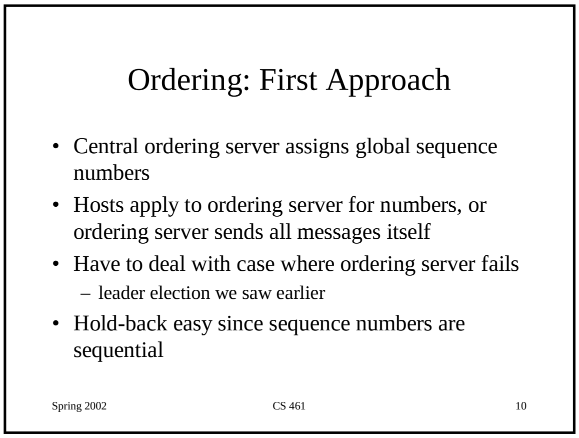
Atomic Messages (cont)

- Fix: if sender crashes, a recipient volunteers to be "backup sender" for the message
  - re-sends message to everybody, waits for ACKs
  - use simple algorithm to choose volunteer
  - apply method again if backup fails
- Must remember all received messages in case we need to become backup sender
  - periodic protocol to "prune" old messages
  - how know it's safe to prune?



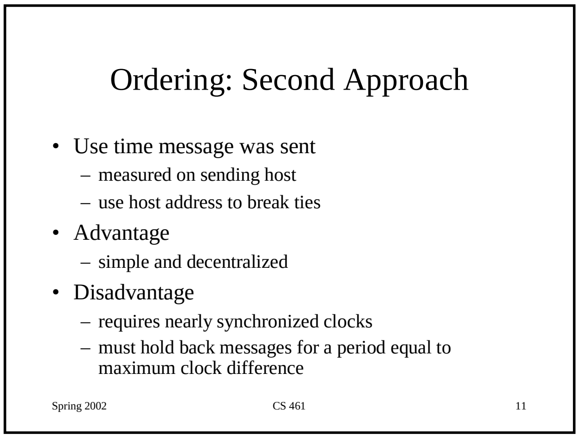
Message Ordering

- So far: different members may see messages in different orders
- Ordered group communication requires all members to agree about the order of messages
- Within group, assign global ordering to messages
- Hold back messages that arrive out-of-order



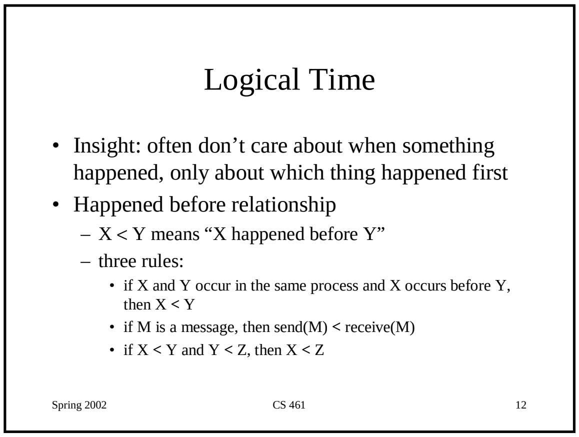
Ordering: First Approach

- Central ordering server assigns global sequence numbers
- Hosts apply to ordering server for numbers, or ordering server sends all messages itself
- Have to deal with case where ordering server fails
  - leader election we saw earlier
- Hold-back easy since sequence numbers are sequential



Ordering: Second Approach

- Use time message was sent
  - measured on sending host
  - use host address to break ties
- Advantage
  - simple and decentralized
- Disadvantage
  - requires nearly synchronized clocks
  - must hold back messages for a period equal to maximum clock difference

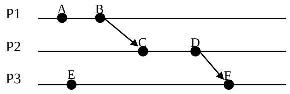


Logical Time

- Insight: often don't care about when something happened, only about which thing happened first
- Happened before relationship
  - X < Y means "X happened before Y"
  - three rules:
    - if X and Y occur in the same process and X occurs before Y, then X < Y
    - if M is a message, then send(M) < receive(M)
    - if X < Y and Y < Z, then X < Z

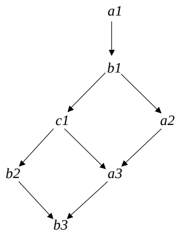
Logical Time (cont)

- Given two events X and Y, either
  - X < Y, or
  - Y < X, or
  - neither (X and Y are concurrent)
- < relation defines a partial order
- Example



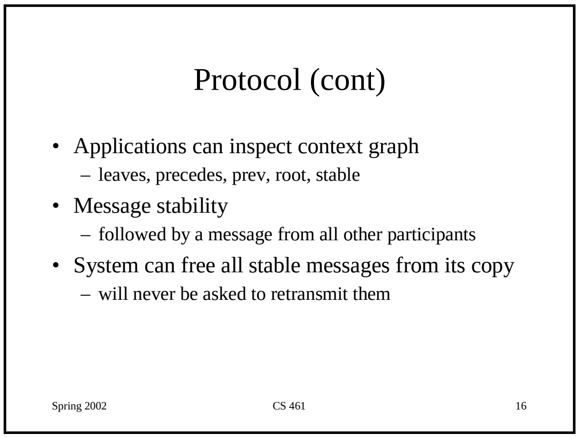
Message Context

- A process sends a message *in the context of* all the messages it has received.
- Group communication represented with a *context graph.*
- Example: 3 senders, denoted *a, b,* and *c*



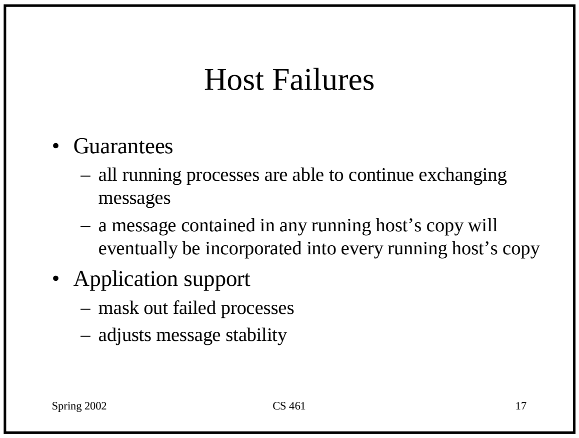
Protocol

- Each server maintains a copy of the context graph
  - union of all copies equals “global graph”
- Send: mid + mid of all predecessor messages
  - leaves of sender’s copy of context graph
  - bounded by number of participants (why?)
- Receive: add to local copy and deliver to application
  - hold back if not all predecessors are present
  - ask sender to retransmit missing messages (why?)
  - pass up to application in “context” order



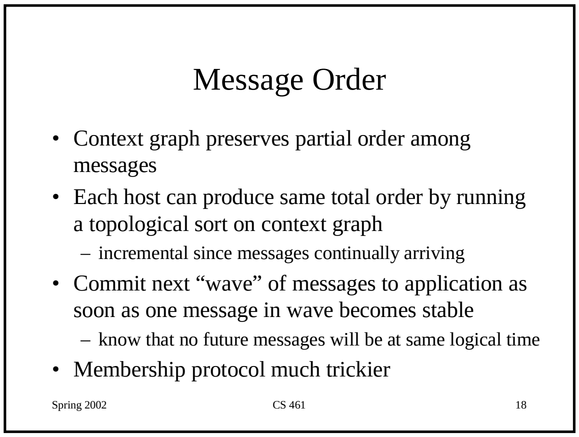
Protocol (cont)

- Applications can inspect context graph
  - leaves, precedes, prev, root, stable
- Message stability
  - followed by a message from all other participants
- System can free all stable messages from its copy
  - will never be asked to retransmit them



Host Failures

- Guarantees
  - all running processes are able to continue exchanging messages
  - a message contained in any running host’s copy will eventually be incorporated into every running host’s copy
- Application support
  - mask out failed processes
  - adjusts message stability



Message Order

- Context graph preserves partial order among messages
- Each host can produce same total order by running a topological sort on context graph
  - incremental since messages continually arriving
- Commit next “wave” of messages to application as soon as one message in wave becomes stable
  - know that no future messages will be at same logical time
- Membership protocol much trickier