CompSci 131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- **Elections**

- **Reading assignment:**
  - **Today: 6.3**
  - **Next time: 6.4**

  - **Complete the assignment *before* next class**

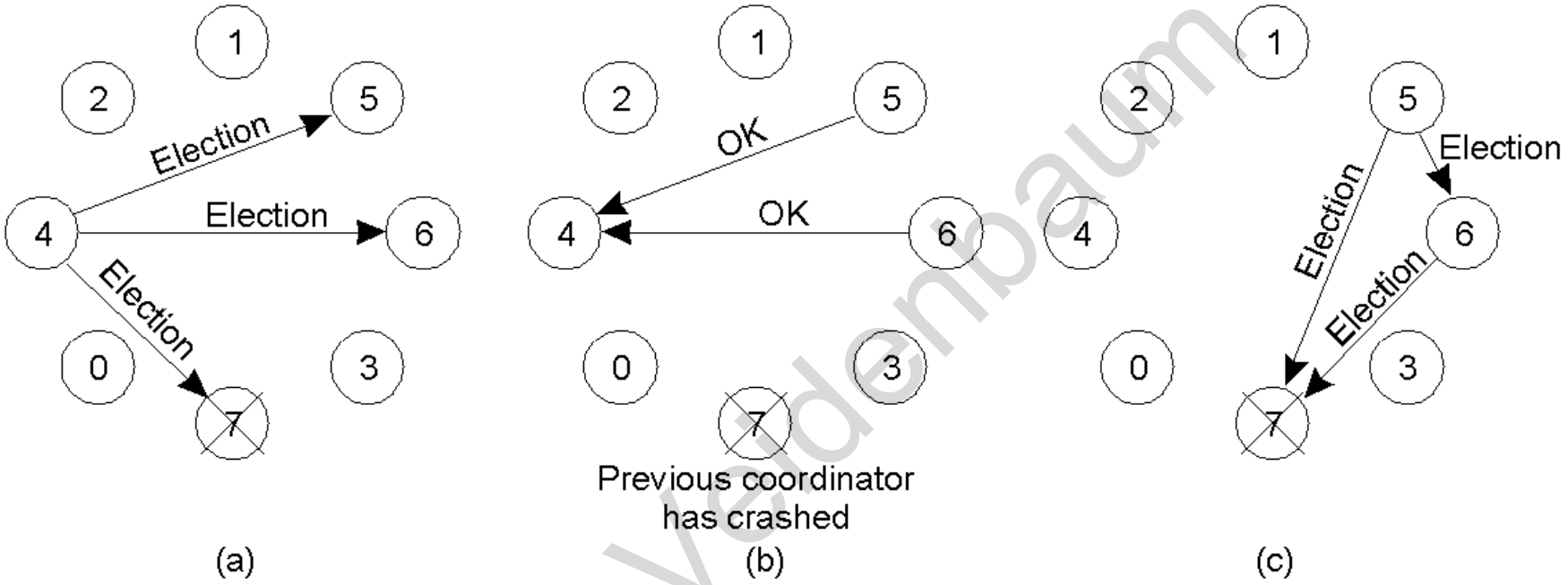
Last Lecture Covered

- **Mutual exclusion algorithms**

Election Algorithms

- Elect a coordinating process
  - Does not matter which
    - » but has to be agreed on by all
  - Processes come and go, *making this hard*
- Need some way of distinguishing processes
  - Assign each a unique number
  - Usually elect one with a highest number
    - » Assumes all numbers are known to all processes
      - Do not know which one is up or down!
- A process initiates an election when
  - It first joins in
  - When it detects the coordinator is down

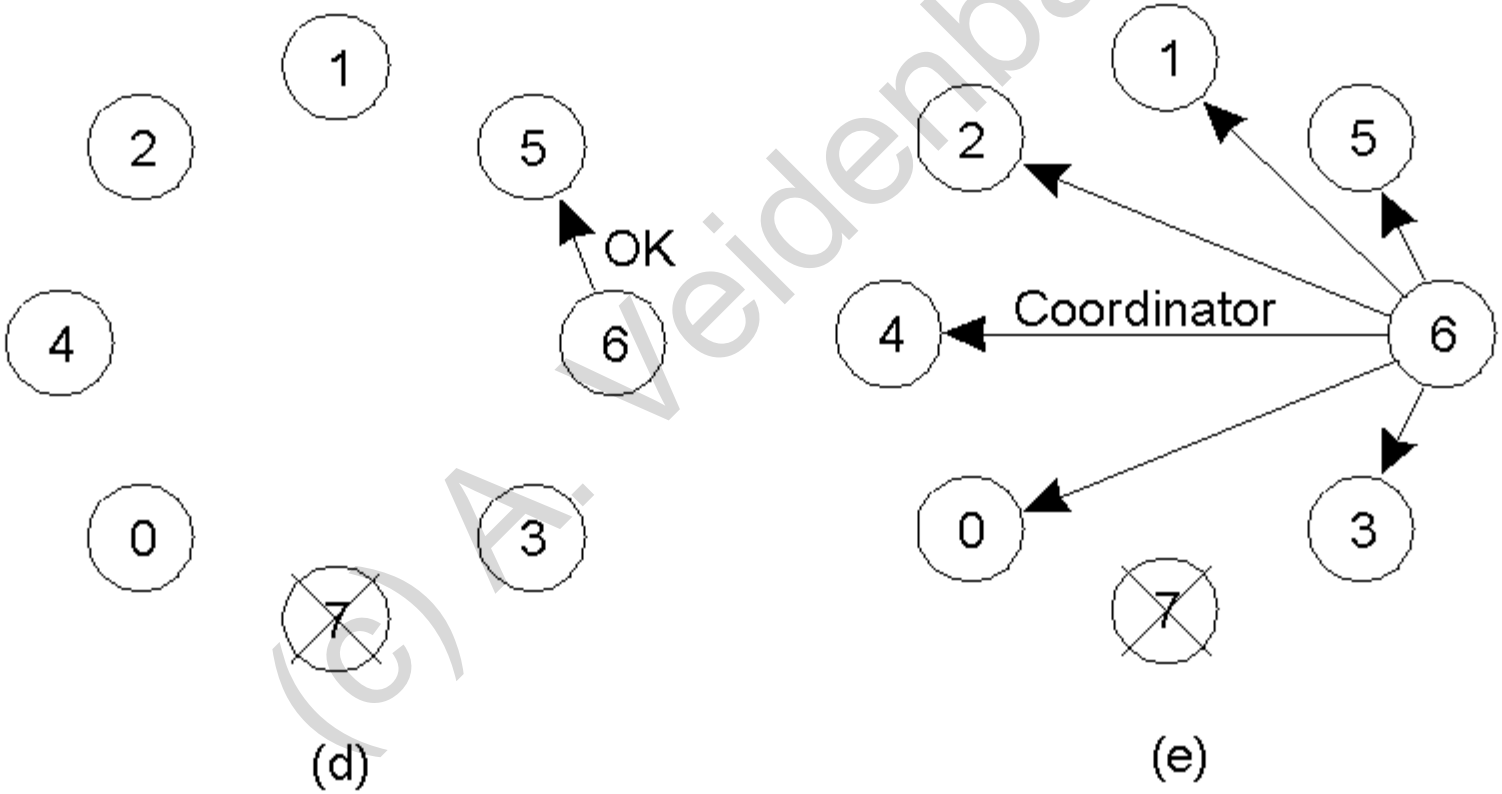
The Bully Algorithm



- **Run the election when a process comes up or detects the coordinator down**
  - **# 7 in this example**
- **Sends election message to higher #'d processes**
  - **becomes coordinator if no one responds**
- **Process 5 and 6 respond, so 4 stops**
- **Now 5 and 6 each hold an election (multiple elections at the same time)**
  - **see messages**

Final State

- **Process 6 tells 5 to stop**
- **Process 6 wins and tells everyone**



A Ring Algorithm

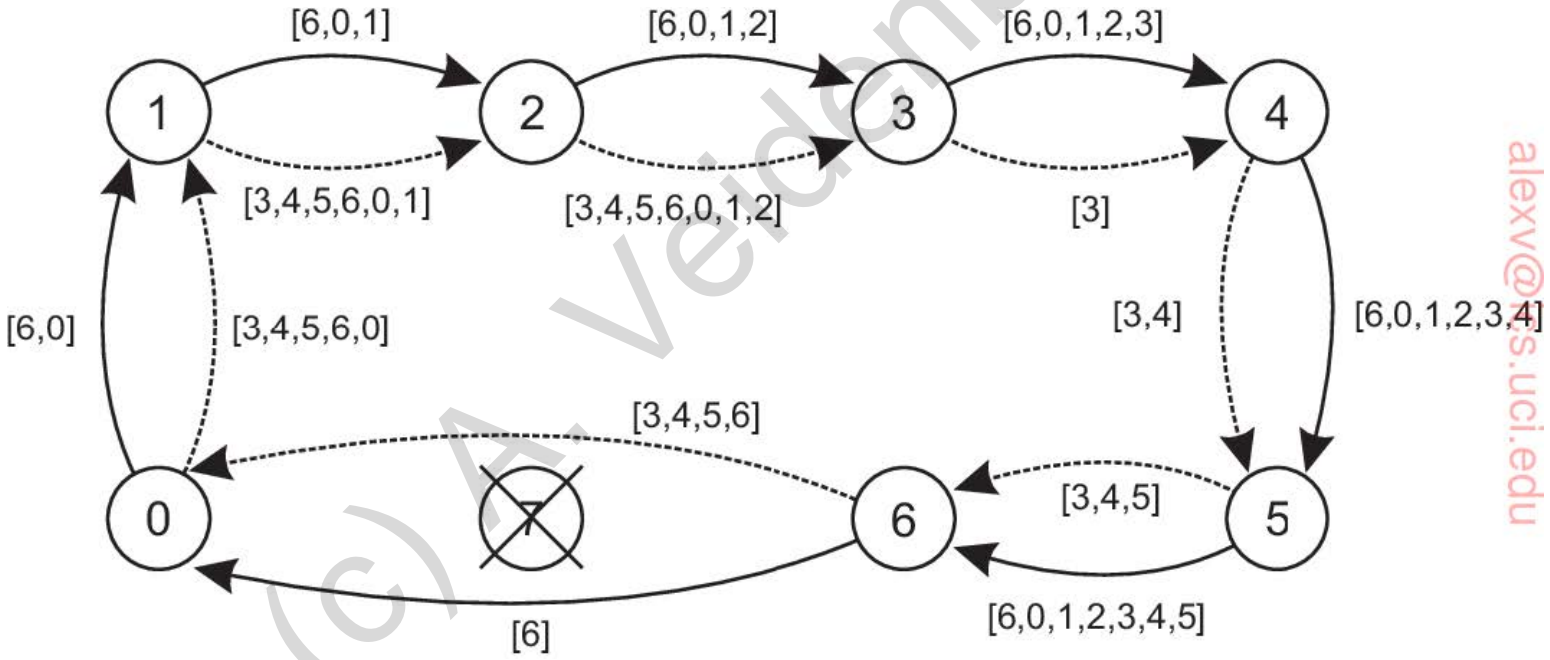
- The ring is logical, imposes order on message traversal
  - Let's assume unidirectional traffic flow
- A process knows its neighbor in the ring
  - If neighbor is down, it resends to the next guy
    - » In fact, it knows all current node IDs
- But there is no token in the usual sense
  - Instead, an Election message is sent
    - » Contains initiator ID
- The rest is straight-forward in a basic case

A Ring Algorithm (2)

- Each process adds its ID to the message and forwards
- The initiator gets its message back and
  1. Removes the election message
  2. sends a new message (Coordinator) to everyone
     - » Or changes the message type, but not content
- Coord. Message circulates and everyone updates
  - The coordinator is the highest numbered process
  - It also informs everyone of newly added or down nodes
- Works well IF there *is only one message*

A Ring Algorithm (3)

- **What if there are two election message? Or More?**
- **Consider the ring below, 7 is the coordinator**
- **6 *and* 3 detect coordinator is down**



- **The example has two elections going on at once**
  - **Both will circulate completely**
  - **Both get the same members**
  - **Both will elect the same coordinator**
  - **Both send Coordinator message**
    - » **WITH THE SAME NUMBERS**

Elections in wireless environments

- **Assumptions:**
  - **Ad-hoc network, nodes come and go**
  - **Messages may be lost (we will not discuss this)**
  - **May use another condition to elect a coordinator**
    - » **Battery capacity remaining, storage available, etc**
      - **Or just highest node id again!**

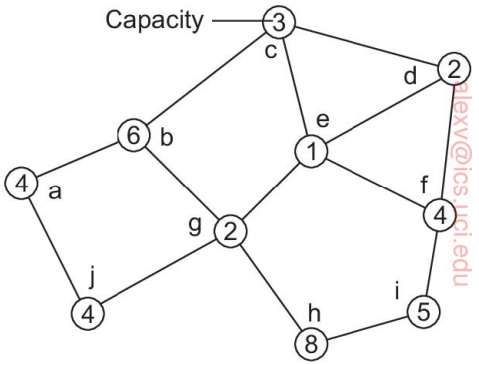
- **A node initiates an election by sending a message to its "immediate" neighbors**
  - **waits for their ACKs before completing the election**

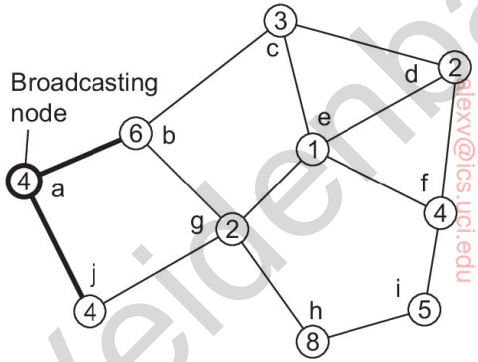
Elections in wireless environments (2)

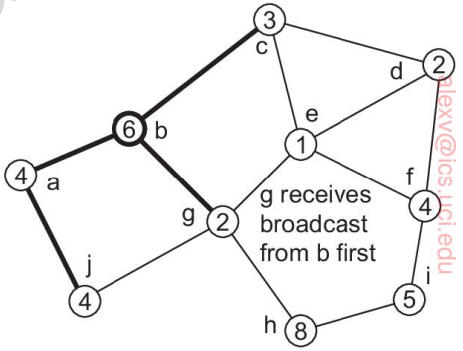
- **A node receiving a message for the first time declares the sender *its parent***
  - **Does not ack the parent immediately**
  - **All other nodes are "non-parents"**
- **A node ACKs a message from non-parent neighbor immediately**
- **Leaf nodes report their <id,capacity> back up**
- **The most eligible node is passed to the parent in ACK**
  - **Eligible = highest remaining battery power or other resource**

Example

- **Node capacity is the number in the circle**
- **Node** a **starts the election**







Example (2)

- **a sent to b,j**
- **b sent to g, c**