On elections and wireless mesh networks
To communicate
Nodes must be in range
Decide they must, about when to wait
And when to complete the exchange
Allowing them to flesh
Out a tree from the mesh

CS 455: INTRODUCTION TO DISTRIBUTED SYSTEMS
[ELECTION ALGORITHMS]

Shrideep Pallickara
Computer Science
Colorado State University

April 23, 2019

Frequently asked questions from the previous class survey

- In the ring-based election:
  - Would there be a case where two processes have the same identifier?
  - What happens when one of the nodes fails?

- In the Bully Algorithm, when a process detects that a coordinator has failed, is that the supervisor of the election?
Topics covered in this lecture

- Election Algorithms
  - Bully algorithm [Garcia-Molina]
  - Elections in wireless environments [Vasudevan et al]
- Architectural Styles

THE BULLY ALGORITHM [GARCIA-MOLINA]
Election of a coordinator after the failure of p4

April 23, 2019
Professor: SHRIDEEP PALICKARA

Election of a coordinator after the failure of p4 and then p3

April 23, 2019
Professor: SHRIDEEP PALICKARA

Eventually …
Satisfying properties E1 and E2

- **E1 (safety)**
  - Impossible for two processes to decide that they are the coordinator
    - Process with the lower identifier will discover that the other exists and defer to it

- **E2 (liveness)**
  - Satisfied because of the assumption of reliable delivery
    - Processes either participate or crash

Safety ... not so soon

- Not guaranteed to meet safety condition if ...
  - Crashed processes are replaced by processes with the same identifier

- Process that replaces a crashed process (coordinator) may decide it has the highest ID
  - Just as another process (which detected the crash) is about to decide that it has highest ID

- Two processes may announce themselves as the coordinator **concurrently**
Safety ... not so soon

- No guarantees on message delivery order
  - Recipients reach different conclusions on which is the coordinator process

- E1 may also be broken if timeout values are inaccurate
  - If the process' failure detector is unreliable

A scenario where safety is violated due to inaccurate failure detection

- \( p_3 \) had not failed but was just **running slowly**

- \( p_2 \) sends its coordinator message, and \( p_3 \) does the same
  - \( p_2 \) receives this after it has sent its message
  - Sets \( elected_2 \) to \( p_3 \)

- \( p_1 \) receives \( p_2 \)'s message after \( p_3 \)'s
  - Sets \( elected_1 \) to \( p_2 \)
Performance of the algorithm

- **Best case**
  - 2nd highest identifier notices coordinator failure
    - Elects itself immediately and sends \((N-2)\) coordinator messages
    - Turnaround time is 1 message

- **Worst case requires** \(O(N^2)\) messages
  - Process with the lowest ID first detects failure
  - \((N-1)\) processes begin elections … each sending messages to processes with higher identifiers

---

**Elections in Wireless Environments**
Elections in wireless environments [Vasudevan’s algorithm]

- Solution can handle failing nodes and partitioning networks
- We will look at simplified approach
  - Ad hoc networks … but the nodes are not allowed to move physically

Wireless ad hoc network setting

- Each node can initiate election by sending election message to its immediate neighbors
- These are neighbors in its range
Forwarding of election messages and parent-child relationships

- When node receives an election message for first time
  - Designates the sender as *parent*
  - Sends out election message to all its neighbors except the parent

- When a node receives an election message from a node other than its parent
  - Merely acknowledge receipt of the message

When a node R has designated Q as its parent

- Forward election message to immediate neighbors (except Q)
- **Wait** for acknowledgements to come in before acknowledging election message from Q
But why wait?

- Neighbors that already have a parent will immediately respond to R
- If all neighbors have a parent?
  - R is a leaf node and will be able to report back to Q quickly
- Report information such as battery lifetime and other resource capacities
  - Allows Q to compare R’s capacities to that of other downstream nodes
  - Select best eligible node for leadership

But Q has sent an election message only because its parent P has

- When Q eventually acknowledges election message previously sent by P
  - It will pass most eligible node to P as well
- Source will know which node is best to be selected as a leader
  - Broadcast this information to all the other nodes
Election algorithm in a wireless network
Election algorithm in a wireless network

April 23, 2019
Professor: SHRIDEEP PALICKARA

Capacity

Election algorithm in a wireless network

April 23, 2019
Professor: SHRIDEEP PALICKARA
Election algorithm in a wireless network

April 23, 2019
Professor: SHRIDEEP PALLICKARA

Elected as Leader

CS455: Introduction to Distributed Systems [Spring 2019]
Dept. Of Computer Science, Colorado State University
Coping with situations when multiple elections are initiated

- Each source tags its *election* message with a unique identifier
- Nodes participate in elections with the highest identifier
  - Stopping participation in other elections
What we will look at

- Architectural styles for designing systems
  - Layered, objects, data, and event based
- Topologies
  - The role they play in systems design
- Implications:
  - Throughput, scaling, fault tolerance and resiliency, latencies
Components are the building blocks of distributed systems

- Modular units
- Well defined-interfaces
- Replaceable

- Connectors
  - Mediate communications and coordination between components

Architectural style of distributed systems are formulated in terms of components

- How they are connected to each other
- How they exchange data
- How they are configured into a system
Broad architectural styles

- Layered
- Object-based
- Data-centric
- Event-based

Layered architecture

- Components are organized in a layered fashion
- Component at layer $L_i$ can call components at layer $L_{i-1}$
- Widely adopted in the networking community
Requests go down the hierarchy; results flow upward

Object-based: Objects are components, connected via (remote) procedure calls
Data centered architectures

- Processes communicate through a **shared repository**
  - Shared distributed file system
  - Shared Web-based data services

Event-based architectures

- Communication is via events
- Processes are **loosely-coupled**
  - Don’t need to be aware of each other
  - Only specify what you need
- **Middleware** decides what goes where
  - Event routed to processes that are interested in them
Event-based architectures

- Component
- Event Delivery
- Event Bus
- Publish
- Component

Shared data spaces: Data-centric plus Event-based

- Processes are **time-decoupled**
  - No need to be active simultaneously
  - Consumers may be offline
SYSTEM ARCHITECTURES

Client Server architecture

- Server implements a service
- Client requests the service
  - Send request
  - Await server response

Request-reply semantics
Interaction between a client and a server

Client

Wait for result

Server

Provide Service

Time

Communications between the client and server

- Could be based on a connectionless, unreliable protocol
- But that means dealing with occasional transmission failures
  - Difficult!
Why dealing with occasional failures is difficult

- Is resending messages enough?
  - Client cannot detect whether
    - Original message was lost OR
    - The transmission of the reply failed
      - If request is resent, operation will be performed twice

Idempotent operations are those that can be repeated many times

- How much do I have in my checking account?
  - Idempotent

- Transfer $10,000 from my bank account
  - Not idempotent
Solution is to use reliable connection-oriented protocols

- Most Internet application protocols are based on TCP/IP
  - Client requests service after setting up connection
  - Server uses same connection to send a response

- Issues
  - Setting up and tearing down connection is costly
    - Even more so for small requests and responses

Demarcation of client-server roles is an issue

- Server for a distributed database
  - Forwards requests to file servers that manage the database table
  - The server itself acts as a client

- Suggested layers include
  - User-interface level
  - Processing level
  - Data level
An example of a 3-tier application

- User Interface
- Query Generator
- HTML Generator
- Ranking Algorithm
- Database

User interface level
Processing level
Data level

Timing diagram in such a setting

- USER INTERFACE
- APPLICATION SERVER
- DATABASE SERVER
- Wait for result
- Request operation
- Wait for data
- Return result
- Return Data
- Request Data
- Time
Client-server and variants

- **Vertical** distribution
- Tiers correspond to logical organization of applications
- Logically different components reside on different machines

The contents of this slide set are based on the following references
