Failure Detection

DISTRIBUTED SYSTEMS – CS425/ECE428 – SPRING 2021
Announcements

Exams
- Dates: Oct 4 7-8:30pm, Nov 1 7-8:30pm, Dec 10 7-10pm
- Process: online, open-book

Lecture back-channel
- Use the CampusWire #lecture room for questions during the lecture
- Sanchit will help answer questions this week

MP0
- Released yesterday, VM request due tomorrow
- Partner search thread on CampusWire

Office Hours
- Zoom office hours started, on calendar
- In-person office hours will start soon
Review of Last Lecture

Distributed systems: processes that fail independently and communicate by sending messages

What does fail means?
- Today: crash-stop (later, other models)

How are messages sent?
- Typically over sockets
- Network eventually delivers messages (no loss, corruption, duplicates, etc.)
  - See 438 and maybe CS461/ECE422 for how
- Messages arrive after a delay

Two models
- Synchronous: delay has a known bound
- Asynchronous: delay is unbounded
Failure Detection Protocols

Ping-Ack:
- **Monitor**: Send a ping every T seconds, wait for an ack
- **Monitored**: respond to each ping with an ack
- Monitor declares monitored process dead after a timeout

Heartbeat
- **Monitored**: Send a heartbeat every T seconds
- Monitor declares monitored process dead if no heartbeat after timeout
Correctness

Completeness: always detect a failure eventually
Accuracy: never detect a spurious failure

Synchronous systems:
  ◦ Ping-ack and Heartbeat complete and accurate with appropriate timeouts

Asynchronous systems:
  ◦ Ping-ack and Heartbeat complete but not accurate

Completeness and accuracy can be traded off
  ◦ 100% complete: always declare failure
  ◦ 100% accurate: never declare failure
  ◦ Cannot have both in asynchronous system

Asynchronous systems use probabilistic accuracy
  ◦ Rarely declare spurious failure
I. Ping-Ack Protocol

\[ p_i \] sends a ping every \( T \) time units

\[ p_j \] replies as soon as it receives a ping

When to consider process \( p_j \) failed?

- \( \Delta_1 \) after sending ping and no ack
- \( \Delta_1 = \) maximum round-trip time (why?)
II. Heartbeating Protocol

$p_j$ sends a heartbeat every $T$ time units

When to consider $p_j$ as failed?

- $T + \Delta_2$ after receiving last heartbeat
- $\Delta_2 = \text{maximum one-way delay}$
- $\Delta_2 = \text{minimum one-way delay}$
Metrics

**Accuracy:** probability that reported failure = actual failure

**Worst-case detection time:**
- Ping-ack: $T + \Delta_1 - \Delta'$
  - $\Delta'$ is the time it took to send the last ping
- Heartbeat: $T + \Delta_2 + \Delta''$
  - $\Delta''$ is the time it took to send the last heartbeat

**Bandwidth**
- Ping-ack: 2 messages every $T$ units
- Heartbeat: 1 message every $T$ units
Metrics for failure detection

Worst case failure detection time

- **Ping-ack**: $T + \Delta_1 - \Delta$ where $\Delta$ is time taken for the last ping from p to reach q before q crashed. $T$ is the time period for pings, and $\Delta_1$ is timeout value.

**Diagram**:
- $t$
- $t + T$
- $t + T + \Delta_1$
- $\text{ping } n$
- $\text{ack}$
- $\text{ping } n+1$

**Worst case failure detection time**:
$t + T + \Delta_1 - (t + \Delta) = T + \Delta_1 - \Delta$

**Q**: What is worst case value of $\Delta$ for a synchronous system?  
**A**: min network delay
Metrics for failure detection

Worst case failure detection time

- **Heartbeat:** $T + \Delta_2 + \Delta$ where $\Delta$ is time taken for last heartbeat from $q$ to reach $p$

  $T$ is the time period for heartbeats, and $T + \Delta_2$ is the timeout.

- **Worst case failure detection time:**
  
  $$(t + \Delta) + (T + \Delta_2) - t = T + \Delta_2 + \Delta$$

- **Q:** What is worst case value of $\Delta$ in a synchronous system?
  
  **A:** max network delay
Tradeoffs

Increasing $T$
- Decreases bandwidth
- Increases worst-case detection time

Increasing $\Delta_1/\Delta_2$
- Increases accuracy
- Increases worst-case detection time
Failure Detection in a Distributed System

That was for one process $p_j$ being detected and one process $p_i$ detecting failures

Let’s extend it to a larger distributed system

Difference from original failure detection is

- We want to detect failure of not merely one process ($p_j$), but all processes in system
Centralized Heartbeating

$p_j$, Heartbeat Seq. $l++$

$p_i$

Downside?
Ring Heartbeating

$p_j$, Heartbeat Seq. $l++$

Downside?
All-to-All Heartbeating

$p_j$, Heartbeat Seq. /++

$p_j$

$p_i$

... Advantage: Everyone is able to keep track of everyone

Downside?
Scalability

How do metrics change with number of nodes (n)?

- Accuracy
- Completeness
- Bandwidth
- Detection time

Ring vs. All-to-all

- Trades off bandwidth for completeness
Summary

Failure detectors are required in distributed systems to keep system running in spite of process crashes

Correctness: completeness & accuracy
- Absolute in synchronous systems
- Probabilistic (at least one) in asynchronous systems

Algorithms: Heartbeat and Ping

Distributed algorithms: Centralized, ring, all-to-all

Metrics: Bandwidth, Detection Time, Scale, Accuracy