CS 582: Distributed Systems

Failure Detectors (Cont'd)

Dr. Zafar Ayyub Qazi

Fall 2024

Acknowledgements: Slides use some material from Radhika Mittal (UIUC) and Indranil Gupta (UIUC)

Office hours

- Monday/Wednesday, 4:30-5:30pm

Previous Lecture: Failure Detection

You should be able to:

- ❑ Explain and analyze the different types of building blocks for detecting that a process has crashed
- ❑ Explain the important properties of failure detectors
- ❑ Explain if we can accurately detect process failures in asynchronous systems

Previous Lecture: Failure Detection

You should be able to:

- ❑ ~~Explain and analyze the different types of building blocks for detecting that a process has crashed~~
- ❑ ~~Explain the important properties of failure detectors~~
- ❑ ~~Explain if we can accurately detect process failures in asynchronous systems~~

Recap: Failure Detection

- Send ping-acks or heartbeats
- Report crash if no response until timeout
- Timeout can be precisely computed for synchronous systems and estimated for asynchronous
- Properties: Completeness, Accuracy, Speed, Scale

"In a distributed system, a failure detector must trade-off between <u>speed</u> and <u>accuracy</u>. Reacting too quickly may cause false alarms, but reacting too slowly may cause prolonged downtime."

— Leslie Lamport

By the end of class today

You should be able to:

- ❑ Explain, analyze, apply, evaluate different types of failure detectors
    - ❑ Centralized, Ring-based, All-All, Gossip

- ❑ Explain and analyze Phi Accrual failure detectors

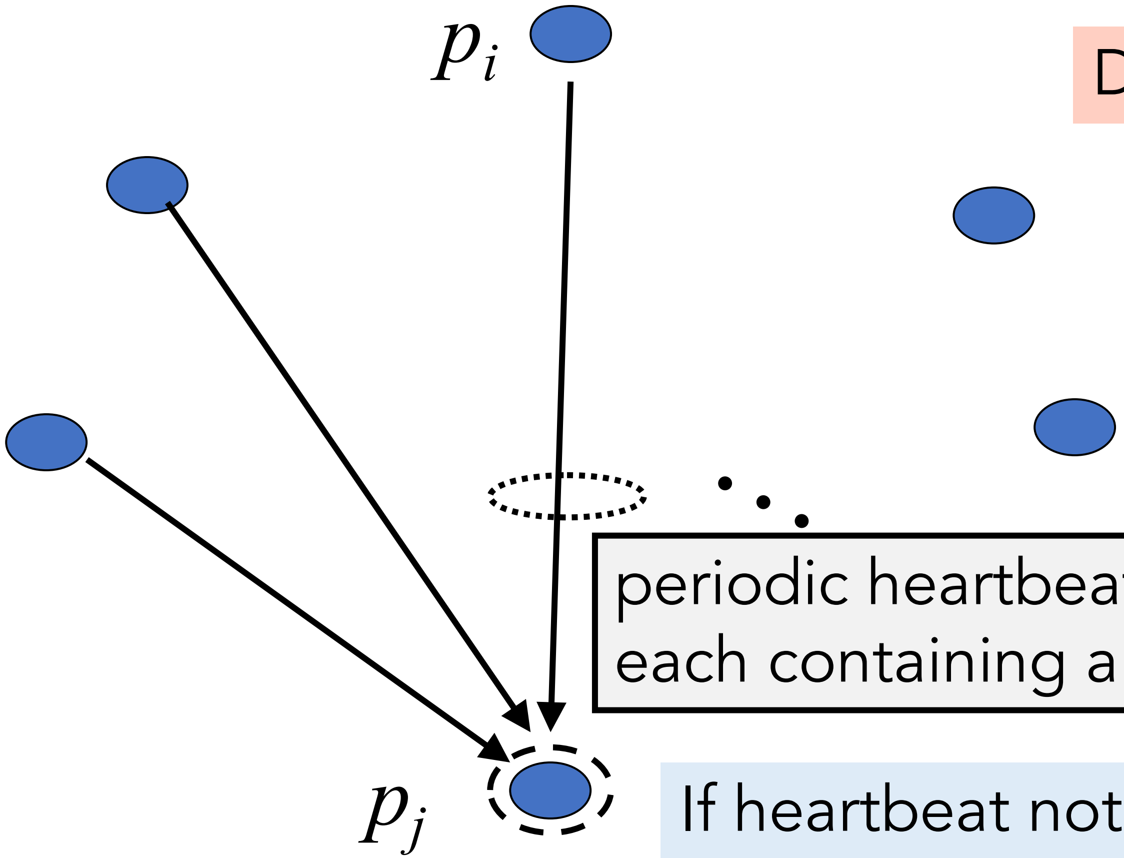
How to detect a process has crashed?

Extending heartbeats

- How do we extend to a system with multiple processes?
- How do distribute failure detection responsibilities?

Centralized Heartbeating

Centralized Heartbeating



Does this scheme ensure completeness?

Any other issues with this scheme?

periodic heartbeat messages **to a central process**, each containing a sequence number

If heartbeat not received from p_i within timeout, mark p_i as failed

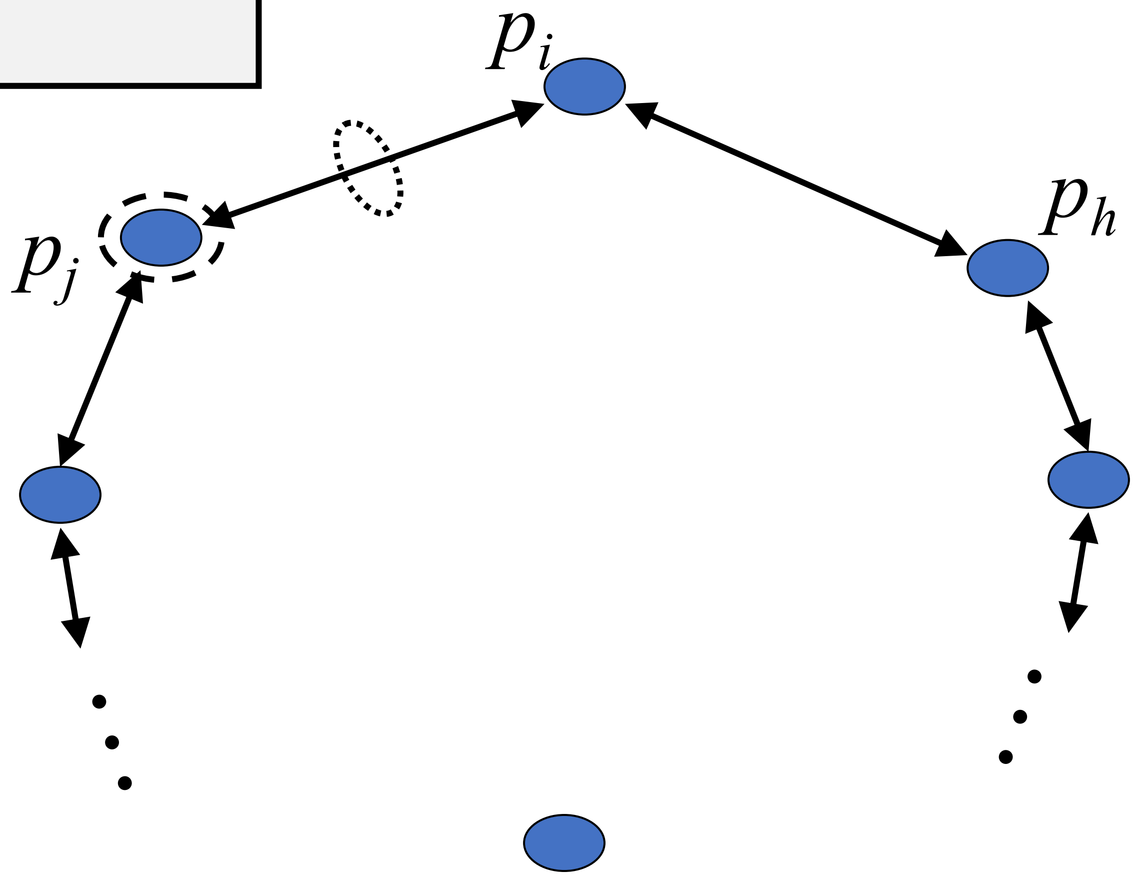
Centralized Heartbeating: Analysis

- This is a simple scheme to implement; only one process is responsible for failure detection
- For n-1 processes, other than P_J, it is complete
- However, when P_J fails, there is no guarantee about who detects that failure
- The other disadvantage is that if you have thousands of processes in your group, P_J might be highly overloaded with messages.

Ring Heartbeating

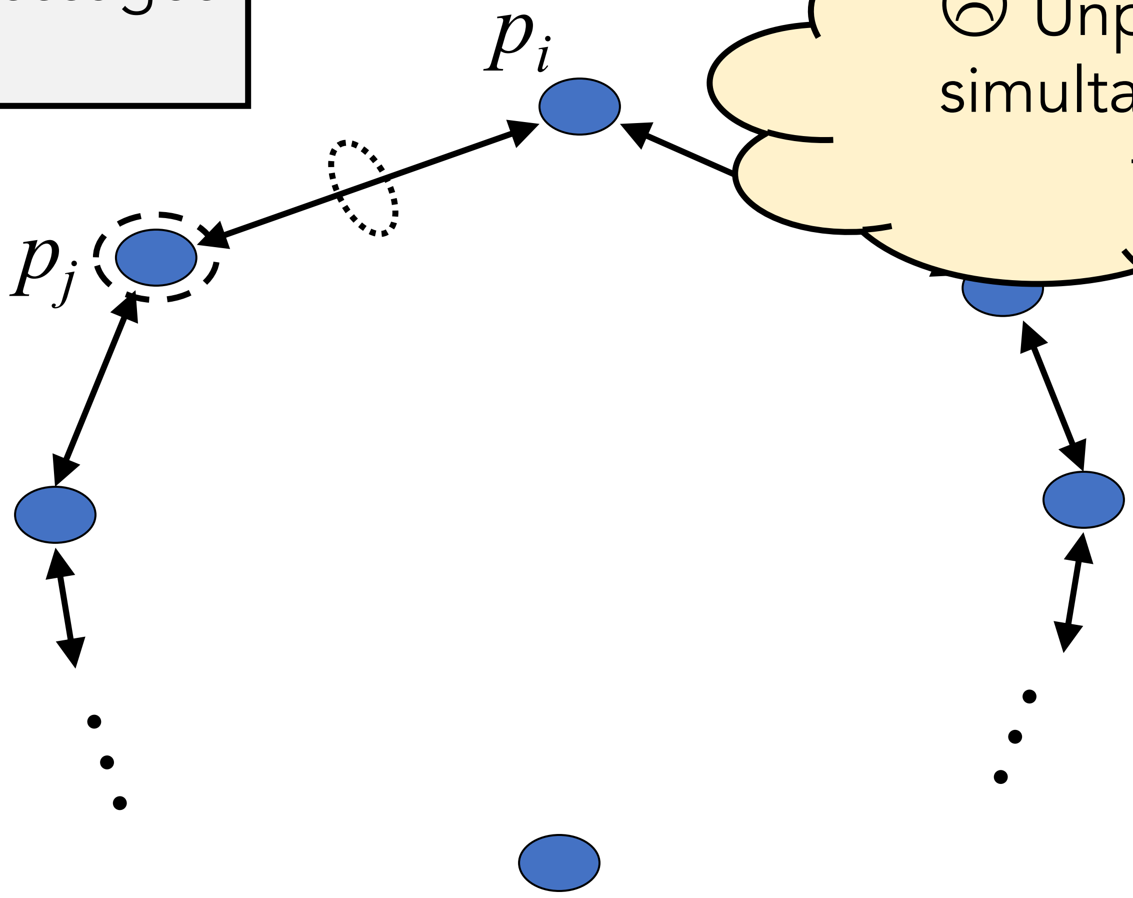
Periodic heartbeat messages <u>to neighbours only</u>

Does this scheme ensure completeness?



Ring Heartbeating

Periodic heartbeat messages <u>to neighbours only</u>



☹ Unpredictable on simultaneous multiple failures

Ring Heartbeating: Analysis

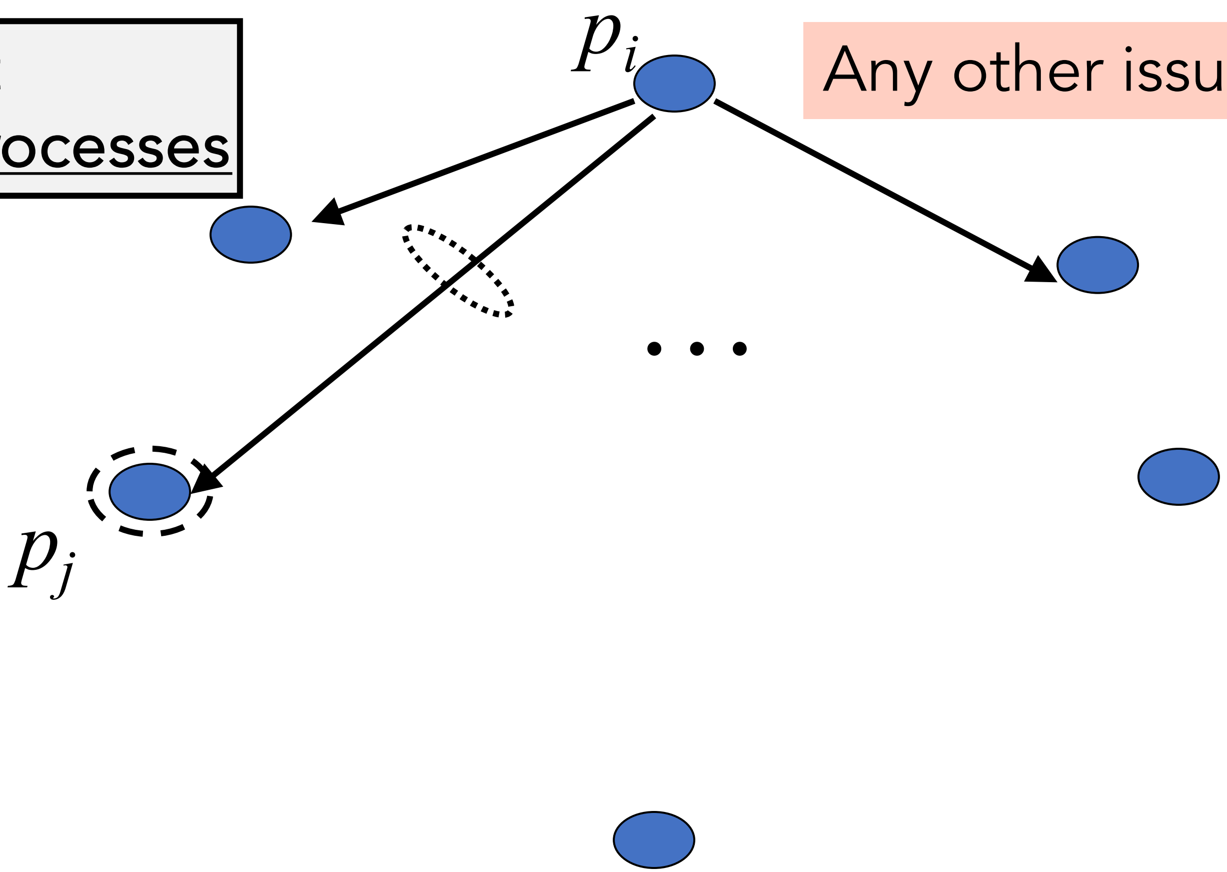
- The task of failure detection is distributed (in contrast to a centralized scheme)
    - A node is only responsible for detecting the failures of its neighbors
- If both neighbors of a node fail simultaneously, then the failure of the node can go undetected
- There will be an overhead of repairing the ring after failures happen

All-to-All Heartbeating

Does this scheme ensure completeness?

Periodic heartbeat messages to all processes

Any other issues with this scheme?



All-to-All Heartbeating analysis

- Ensures completeness (as long as one other non-faulty process is up)
- Although, it has an equal load per member. The load is high
- Another problem with all-to-all heartbeating is that if you have one process P_J that is slow and is receiving packets at longer delay than others, it might end up marking all the other or almost all the other processes as having failed, with higher probability. And so you might have a lower accuracy or a very high rate of false positives in all-to-all heartbeating.

Gossip

Gossip-based Failure Detectors

- Nodes periodically exchange information about the state of other nodes they know about

- This "gossip" spreads through the network, eventually reaching all nodes

- Used in many real distributed systems, in particular, large-scale, decentralized systems
  - E.g., Amazon DynamoDB, Meta's Cassandra, MongoDB, etc

Gossip: Node State

- Each node maintains a list of other nodes it knows about:
- For each known node, it keeps:
  - A heartbeat counter and/or timestamps
  - The last time it heard from this node
  - The status of the node (e.g., alive, suspected)

Gossip Protocol

- Periodically (e.g., every second), each node
  - Selects one or more random nodes
  - Sends it list of known nodes and their statuses to the selected node(s)
- Information merging:
  - When a node receives gossip:
    - It updates its own list with any newer information
    - For each node in the received list
      - If it's a new node, add it to the list
      - If the received heartbeat is higher, update the heartbeat and timestamp
      - If the received heartbeat is lower, keep the local (newer) information

Gossip: Failure Detection

- Each node periodically checks its list:
  - If a node's information hasn't been updated for a certain timeout: Mark it as "suspected”
  - If it remains unchanged for a longer timeout:
    - Mark it as "failed"

Gossip Analysis

- Advantages

- Disadvantages

Gossip Analysis

Advantages

- Scalability
- Robustness
- Network efficiency

Disadvantages

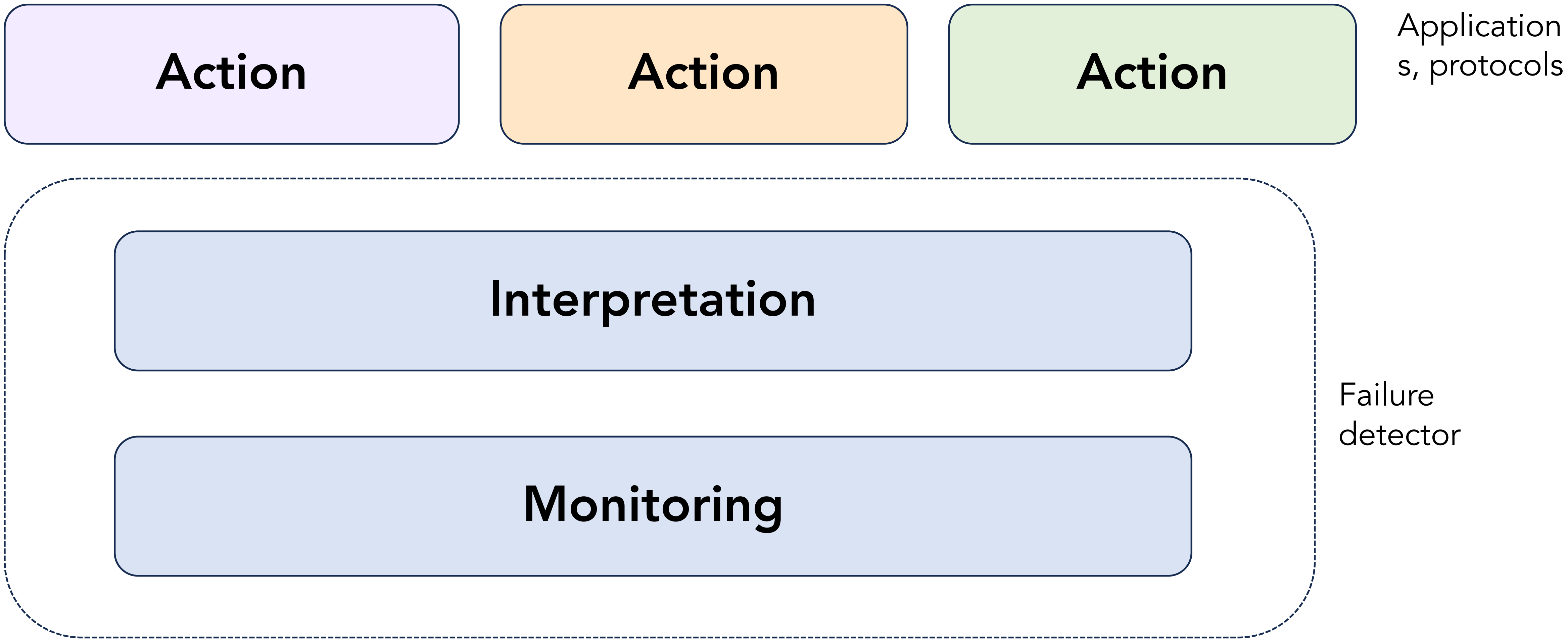
- Delayed detection
- Eventual consistency
- Network overhead

Questions

Let’s Zoom Out

How a failure detector will be used?

What is a failure detector doing?



Action

Action

Action

Applications, protocols

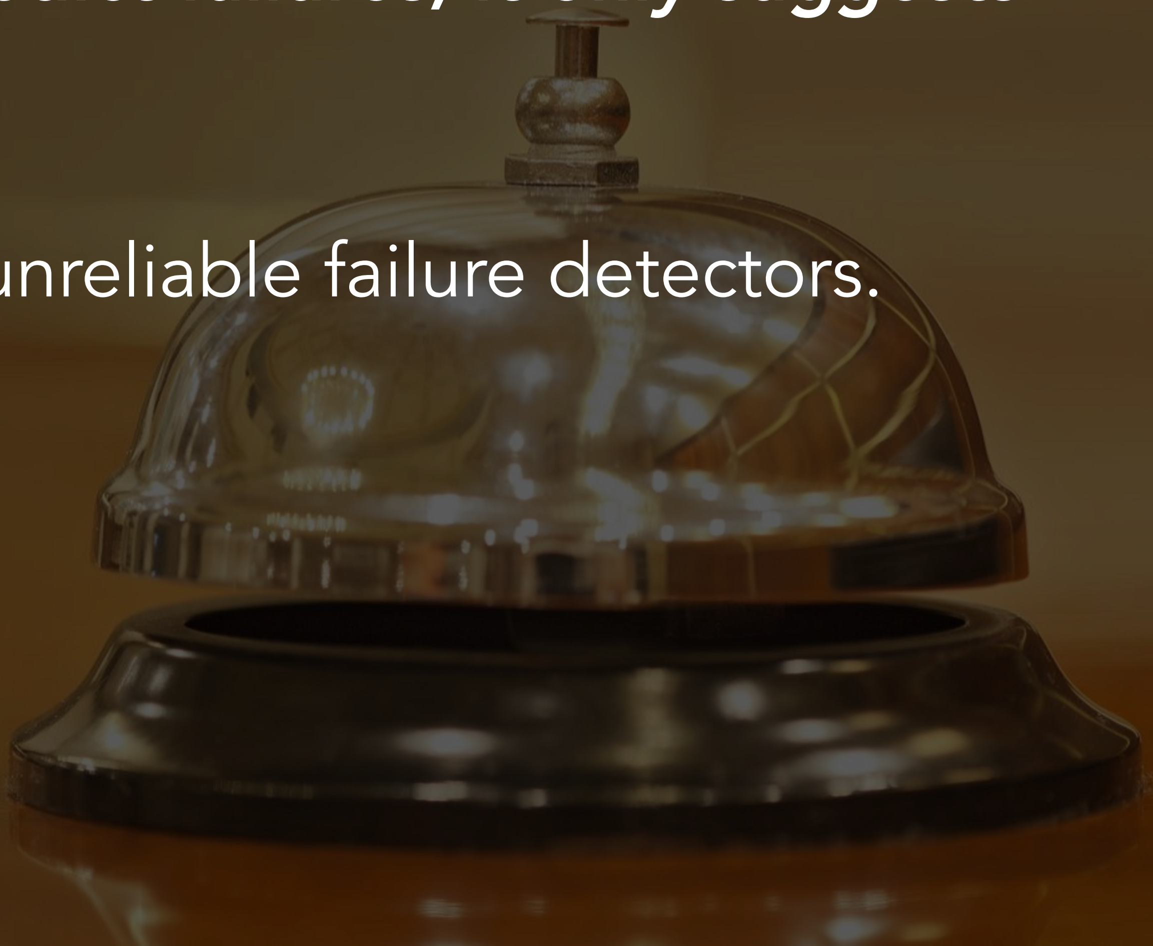
Interpretation

Monitoring

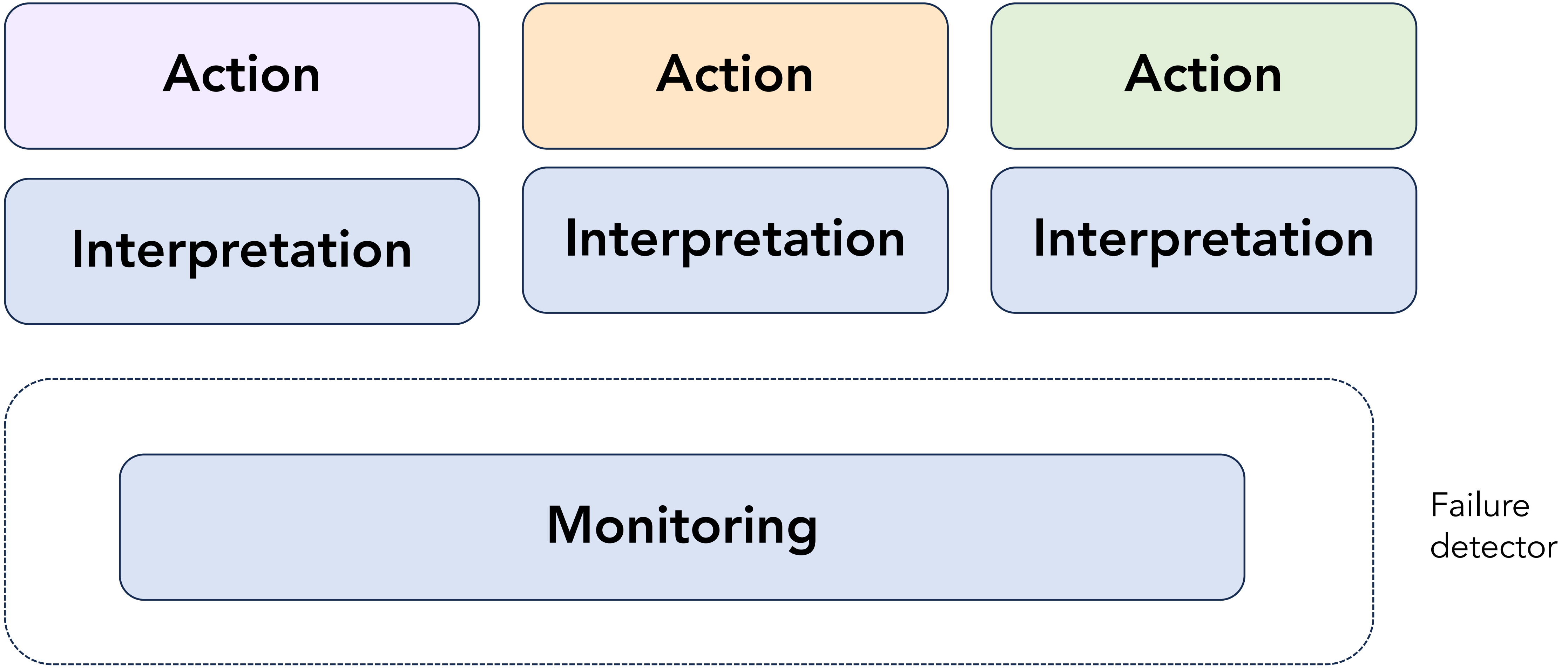
Failure detector

- ***A failure detector does not predict failures, it only suggests suspicions, …”***

-- Chandra and Toueg's work on unreliable failure detectors.



What is a failure detector doing?



Action

Action

Action

Interpretation

Interpretation

Interpretation

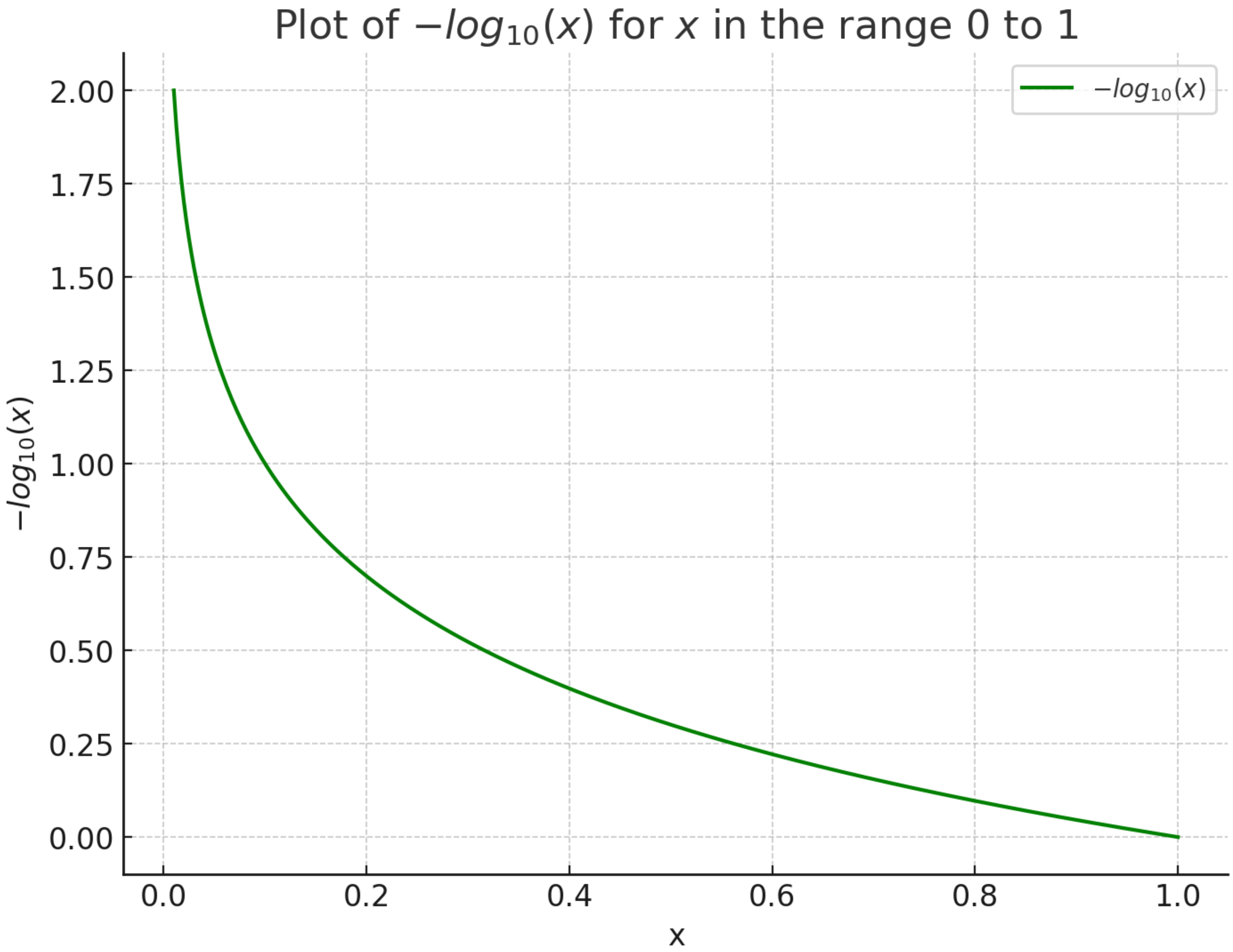
Monitoring

Failure detector

Phi Accrual Failure Detectors

- Instead of providing a binary up/down output, it provides a continuous suspicion level (φ)
- The (φ) represents the likelihood that a node has failed
- It uses the history of heartbeat inter-arrival times to estimate the probability of the next heartbeat's arrival

$$\varphi(t_{now}) \overset{\text{def}}{=} -\log_{10}(P_{later}(t_{now} - T_{last}))$$



Plot of $-log_{10}(x)$ for x in the range 0 to 1
$-log_{10}(x)$
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
x
$-log_{10}(x)$

Phi Accrual Failure Detectors

- Dynamic Thresholds:
  - Instead of fixed timeouts, it uses dynamic thresholds based on the calculated **φ** value.
  - Higher **φ** values indicate higher suspicion of failure

- Interpretation of **φ** Values
  - Low **φ** (e.g., < 1): High confidence the node is alive
  - High **φ** (e.g., > 5): High suspicion the node has failed
  - Intermediate values: Increasing uncertainty

Summary: Failure Detection

- Failure detection (detecting a crashed process):
  - Send periodic ping-acks or heartbeats
  - Report crash if no response until a timeout
  - Timeout can be precisely computed for synchronous systems and estimated for asynchronous
  - Metrics: *completeness, accuracy, speed, scale.*
  - Failure detection for a system with multiple processes:
    - Centralized, ring, all-to-all, gossip
      - Tradeoffs between completeness, bandwidth usage, speed
    - Phi Accrual Failure Detector

Next Lecture

Remote Procedure Calls