SWIM: *S*calable *W*eakly-consistent *I*nfection-style Process Group *M*embership Protocol

Das, A., **Gupta, I**. and Motivala, A., 2002, June. Swim: Scalable weakly-consistent infection-style process group membership protocol. In *Proceedings International Conference on Dependable Systems and Networks* (pp. 303-312). IEEE.

https://courses.engr.illinois.edu/cs425/fa2019/L6.FA19.pptx

Control Planes and Microservices

Control planes are the information systems for your distributed collection of microservices.

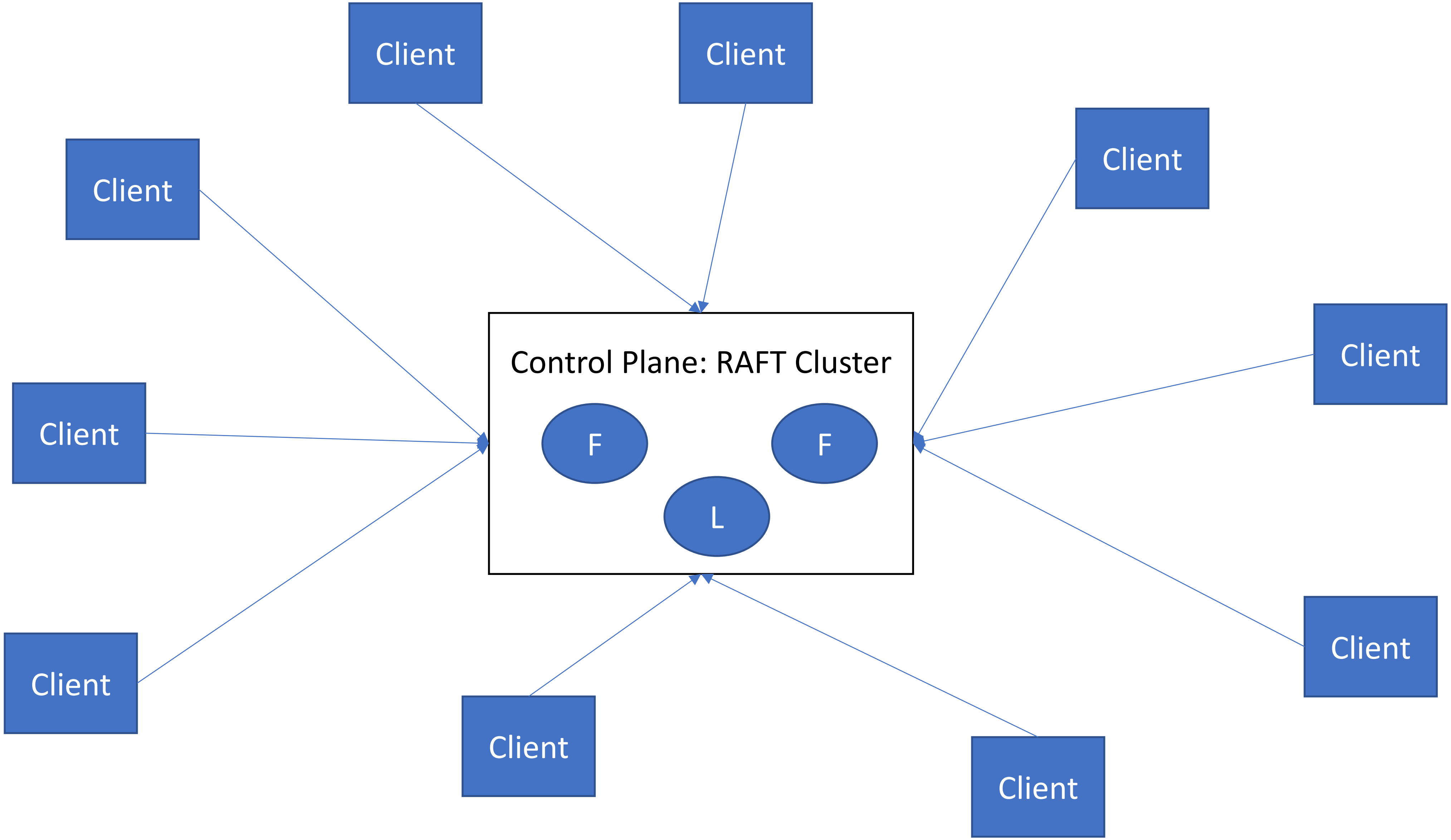
They help microservices find each other

They track system state

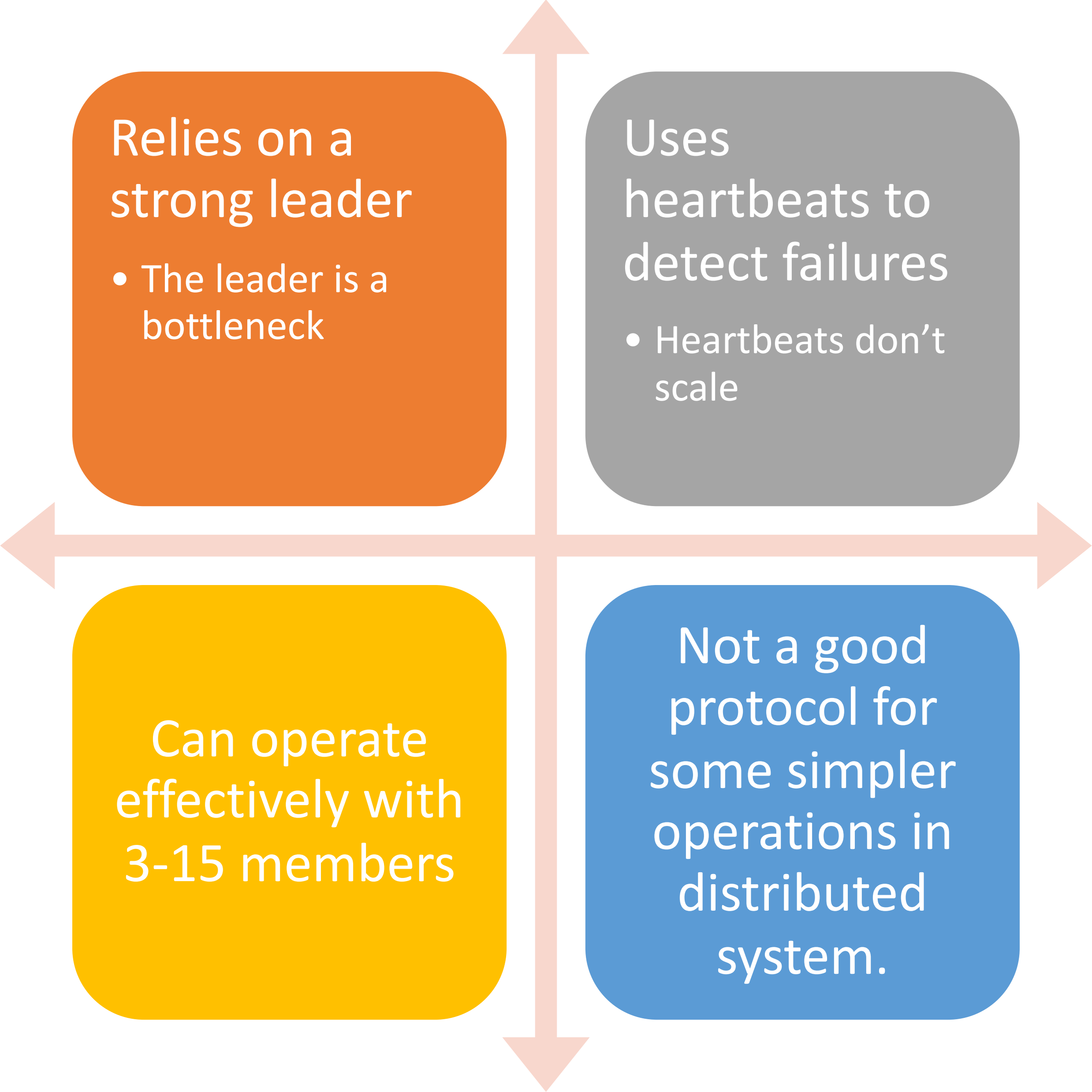
These are the bases for higher level operations

RAFT Recap

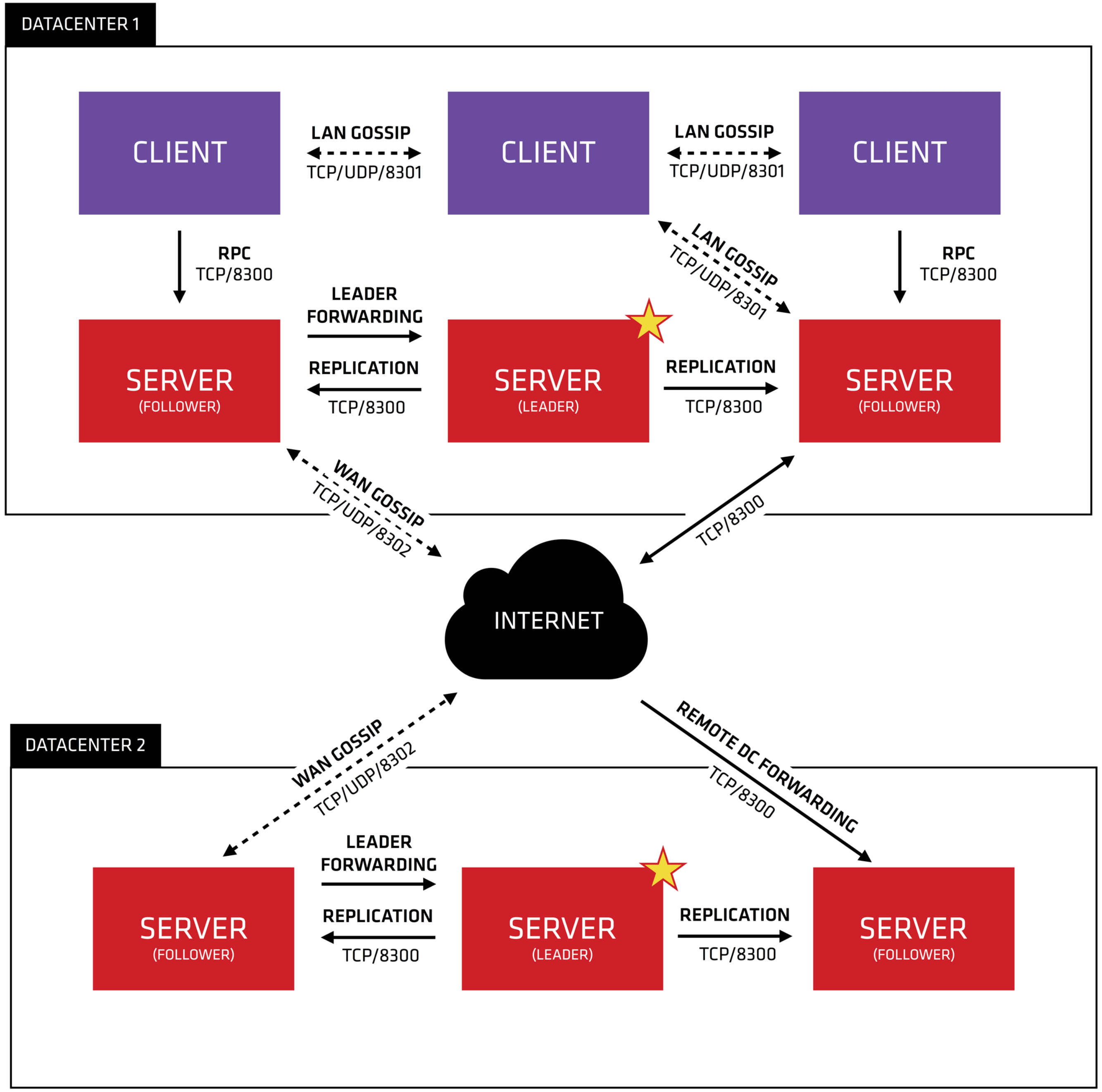
- RAFT and similar systems are used to manage ordered logs.
- Logs capture the state of a distributed system and so must be handled carefully
- RAFT is fault-tolerant and strongly consistent
- Consensus model for managing distributed state logs


Client
Client
Client
Client
Control Plane: RAFT Cluster
F
F
L
Client
Client
Client
Client
Client
Client
Client

RAFT Scaling Limitations for Clients



Consul, RAFT, and SWIM/Gossip



https://www.consul.io/docs/internals/architecture.html

Scaling to Data Center Size and Beyond

- There are simpler services needed by distributed systems than logs.
- But these need to scale to 1000s, 10,000s, 100,000s or more processes.
- Even better, can you design a system that will scale indefinitely?
- Example: how do you detect server failures if you are running a modern cloud center?

Two Key Operations for Distributed Systems

Membership: Each process knows all the other members

Faulty members need to be removed from each member's list

Fault Detection: Failed processes are detected by the other members and communicated throughout the system.

SWIM Insight Compared to Prior Systems

- Membership changes and fault detection are separate processes
- Monitoring needs to go on all the time, but membership changes because of faults occur at a longer time scale.
- Therefore, SWIM has two basic operations: Fault Detection and Dissemination

Problems with All-to-All Heartbeats

- Each member of a cluster sends and receives small heartbeat messages with all other members
- If M_x doesn't receive a heartbeat from M_y within a timeout, it marks M_y as faulty
- This works for smaller systems
- But it grows quadratically (like N^2) with the size of the cluster

Properties of Failure Detection Protocols

- Strong completeness: crash failures are detected by all non-faulty members
- Speed of detection: the time interval between a failure and its first detection
- Accuracy: the rate of false positives
- Network Message Load required

You Can’t Have It All

- Failure detection in an asynchronous network cannot be simultaneously 100% accurate and strongly complete
- So, you need to make a choice
- **Strong completeness** is the usual choice over 100% accuracy
- So, we need a way to minimize false positives
  - That is, incorrectly marking a process as failed when it has not

SWIM Failure Detection, Step 1

SWIM cluster members only monitor a subset of the other members.

You have two parameters: T_p (protocol time) and K (# of members to monitor)

Every T_p seconds, each member sends out a **PING** to K other members of the cluster.

If M_x receive the ACK from M_y within a timeout period, all is well.

No need to update memberships.

Swim Failure Detection, Step 2

- If M_x doesn't hear from M_y before the timeout, it asks for help
- M_x sends a PING-REQ(M_y) message to K other members of the cluster
- The other members PING(M_y) and return their results to M_x
- If M_y responds to any of these pings, and if M_x gets this message back from a member of K, all is well.
- Again, no unnecessary membership updates

SWIM Failure Detection, Step 3

If M_x cannot confirm that M_y is alive even after Step 2, then it needs to tell the rest of M that it has detected a failure.

This is the Dissemination part of the protocol

Dissemination and the Gossip Protocol

- Let Q_f be the number of non-faulty members of the cluster with N members
- The likelihood that a member is pinged by some other member in the protocol time step is **1 – exp (-Q_f)** in the limit of large N.
  - In other words, very likely unless the system is very faulty
- The time required for detecting a failure is **T_p / (1-exp(-Q_f))**
  - In other words, pick T based on a good guess for your average Q_f
- A more complicated expression helps you pick K to put a bound on false positives
  - Depends on Q_f and the probability of UDP packet delivery

SWIM: Strong Completeness Satisfied

- A faulty member will eventually be PINGed, detected as faulty, and removed.
- Message load per member of the cluster is constant
- Compare all-to-all, where members must respond to **~N^2** pings for an N-member system
- None of the properties of SWIM’s failure detection protocol depend on the cluster size.
  - That scales!

SWIM Protocol Primitives Summary

- PING: sent by one member to another randomly chosen member
- PING-REQ (M_y): sent by a member to a subset of other members if it thinks M_y is faulty
  - The other members will now PING M_y
- ACK: the response to a PING

Dissemination

Telling the rest of the cluster about membership changes

Disseminating Failures

This part of the protocol needs to propagate efficiently to the entire system.

We'll assume failures are relatively rare compared to the protocol time T for PINGs

Infection-Style Dissemination

- M_x includes information about failed members in all of its communications with other members
  - Hey, M_z, PING, and by the way, M_y has failed
  - Hey, group, PING-REQ(M_a), and, by the way, M_y has failed
  - Hey, M_z, I’m alive, here’s your ACK, and by the way, M_y has failed.
- Anyone receiving this message from M_x will remove M_y from its group list

Dissemination Performance Summary

- (Analysis deleted)
- Let LAMBDA be a parameter.
- After **t=LAMBDA*log(N)** rounds of the protocol, only **N^[(-2)(LAMDA -1)]** members have not heard about the failure
- Note that this goes quickly to zero, even for large N, thanks to the log()

Reducing False Positives with Suspicions

A healthy cluster member may fail the PING test because of temporary network issues or its own load.

- Note 1: this is likely in VMs since real hardware is shared by many VMs
- Note 2: sidecars can help with load issues

SWIM uses a “Suspicion” subprotocol to reduce these.

Propagating Suspicion

- If M_y fails M_x’s PING and PING-REQ tests, M_x marks it as “suspicious” rather than “failed”.
- M_x propagates the message “M_x suspects M_y” to the rest of the cluster
- M_z marks M_y as suspicious when it receives the message
- Suspicious M_y is still treated as alive by the cluster

Suspicion, Continued

- If M_z later successfully pings M_y, it moves it back to its “alive” list
- M_z then propagates the message “M_z knows M_y is alive” to the rest of the cluster
- Other processes remove M_y from their suspicious list when they receive this message.
- M_y can also disseminate this message

Suspicion, Continued

- Suspected entries are marked as failed if they haven't responded within a time out.
- If M_x expires M_y, it will propagate the message
- Since M_y can be suspected and unsuspected several times, its "suspected" status is associated with an "incarnation number", monotonically increasing index

SWIM Scaling Properties

- Imposes a constant message load per group member regardless of the cluster size
- At least one working member detects failures within a constant time.
- Provides a deterministic bound on the time it takes for any non-faulty process to learn about a faulty process
- Latency for learning about failures increases logarithmically (slowly) with cluster size (good thing)
- Reduces false positives (PING-REQ and Suspicion mechanisms) without performance penalties

Final Thoughts

- In practice, the Suspicion mechanism can still lead to too many fault positives under certain circumstances
  - Slow networks
  - Overloaded processes can be slow to respond
  - Denial of service attacks can make servers unresponsive
- SWIM may also not work well if there are too many faulty members
- Having healthy processes frequently getting kicked on and rejoining the system can hurt performance
- Consul uses a modified SWIM with extensions called Lifeguard to address this.

Dadgar, Armon, James Phillips, and Jon Currey. "Lifeguard: Local health awareness for more accurate failure detection." In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 22-25. IEEE, 2018.