HashiCorp Consul

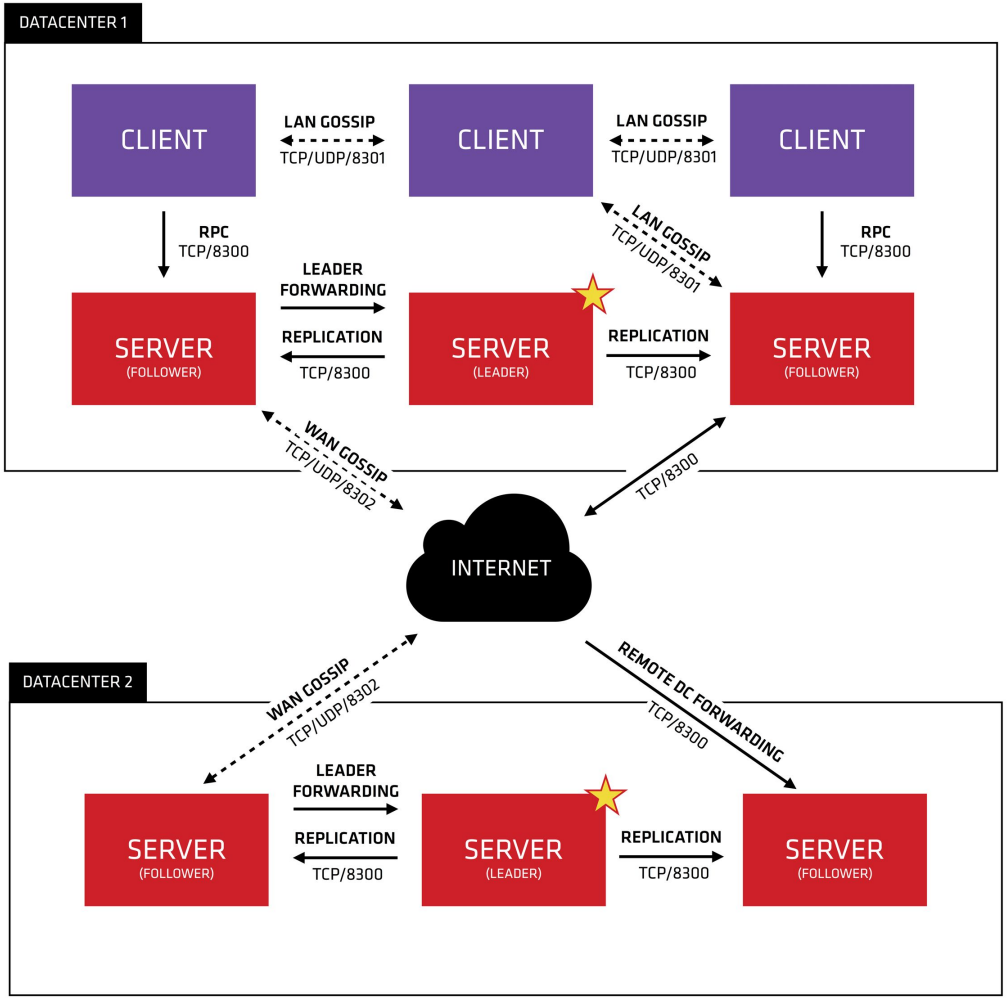
Service Discovery and Configuration Made Easy

What is Consul?

- Service Discovery over DNS and HTTP with built-in load balancing
- Health Checking on node and service level
- Replicated Key Value Store
- Multi Datacenter Federation

Architecture

- agent on all nodes
- consensus server nodes
- lan gossip inside dc
- lan gossip requires full dc mesh
- wan gossip between dc
- wan gossip requires full server mesh
- 3 or 5 server nodes per dc
- up to 100k client nodes per dc
- no data replication between dc



HashiCorp

Service

- Service configuration via file or API call

```
{
  "service": {
    "name": "redis",
    "tags": ["primary"],
    "port": 8000,
    "checks": [
      {
        "script": "/usr/local/bin/check_redis.py",
        "interval": "10s"
      }
    ]
  }
}
```

Health Check - Script

```
{
  "check": {
    "id": "mem-util",
    "name": "Memory utilization",
    "script": "/usr/local/bin/check_mem.py",
    "interval": "10s",
    "timeout": "1s"
  }
}
```

Health Check - HTTP

```
{
  "check": {
    "id": "api",
    "name": "HTTP API on port 5000",
    "http": "https://localhost:5000/health",
    "tls_skip_verify": false,
    "method": "POST",
    "header": {"x-foo":["bar", "baz"]},
    "interval": "10s",
    "timeout": "1s"
  }
}
```

Health Check - TCP

```
{
  "check": {
    "id": "ssh",
    "name": "SSH TCP on port 22",
    "tcp": "localhost:22",
    "interval": "10s",
    "timeout": "1s"
  }
}
```

Health Check - TTL - Dead Man's Switch

```
{
  "check": {
    "id": "web-app",
    "name": "Web App Status",
    "notes": "Web app does a curl internally every 10 seconds",
    "ttl": "30s"
  }
}
```

Health Check - Docker

```
{
  "check": {
    "id": "mem-util",
    "name": "Memory utilization",
    "docker_container_id": "f972c95ebf0e",
    "shell": "/bin/bash",
    "script": "/usr/local/bin/check_mem.py",
    "interval": "10s"
  }
}
```

Replicated Key Value Store

- consul kv put hello world
- consul kv get hello
- consul lock service/app myapp
- curl http://localhost:8500/v1/kv/hello


```
[
  {
    "LockIndex": 0,
    "Key": "hello",
    "Flags": 0,
    "Value": "d29ybGQ=",
    "CreateIndex": 89,
    "ModifyIndex": 108
  }
]
```

Network Coordinates

- network tomography system to compute network coordinates for nodes in the cluster
- useful to find nearest service or failing over to services in closest datacenter

```
$ consul rtt consul-1 consul-2
Estimated consul-1 <-> consul-2 rtt: 0.435 ms (using LAN coordinates)
$ consul rtt consul-1 consul-3
Estimated consul-1 <-> consul-3 rtt: 0.629 ms (using LAN coordinates)
```

Security

- Gossip can be secured using a shared key with AES-128 (GCM)
- RPC supports TLS with optional client authentication

Demo

Consensus Protocol - Raft

- Consensus is the process of agreeing on one result among a group of participants
- Consensus: Bridging Theory and Practice (2014)
- Raft: In search of an Understandable Consensus Algorithm (2014)
- The Secret Lives of Data

Raft - Components

- Leader Election
  - Follower
  - Candidate
  - Leader
- Replicated Log
  - Append only log
  - Can be compacted by snapshotting the state machine to save disk space
- State Machine
  - Finite state machine representing the current state, i.e. current values of the key value store
  - Consul: https://github.com/hashicorp/go-memdb

Raft - Follower

- Follower only append entries sent by the leader to their local log
- Follower don’t accept client write requests
- If a follower doesn’t receive a heartbeat after a timeout from the leader it becomes a candidate

Raft - Candidate

- A candidate assumes that the leader is failed and tries to become the new leader
- A candidate sends a vote request to all known raft members including its last log entry
- Everyone raft member who receives a vote request accepts the vote if its own log is shorter/equal otherwise rejects it
- If a quorum of nodes (member / 2 +1) accepts the vote the candidate transitions to become the leader of the next term

Raft - Leader

- There is always a single leader per term
- The leader accepts clients requests, i.e. read/write a value
- The leader replicates the events to all followers
- A follower acks which events it has written to its local log
- If a quorum (members / 2 + 1) has written the event it will be committed and is applied to the state machine
- The state of the state machine is visible to the client, i.e. read value x
- The leader distributes which events are committed to the follower
- The leader sends heartbeats to all followers even if no new events where written

Raft - Quorum - Members / 2 + 1

- The raft cluster is accessible as long as quorum of members is reachable
- Every state change requires the quorum
- Guarantees that no committed event is ever lost
- Larger quorum allows a higher failure tolerance but increases the latency as more replication is required
- Rule of thumb: odd number of members either 3, 5 or 7
- In consul only server nodes are part of raft

Consul Consistency Modes

- default - leader leasing
  - leader assumes it is leader for a given time frame and answers read queries
  - can lead to short period of stale reads during partitioning
- consistent
  - leader verifies that it is still leader before answering read queries
  - guarantees always consistent reads with extra latency for cluster round trip
- stale
  - every node can answer reads
  - stale reads possible but normally in a small time frame (replication latency)

Gossip Protocol

- SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol (2002)
- Making Gossip More Robust with Lifeguard (2017)
- Lifeguard : SWIM-ing with Situational Awareness (2017)
- HashiCorp Serf - https://www.serf.io
- Serf Convergence Simulator

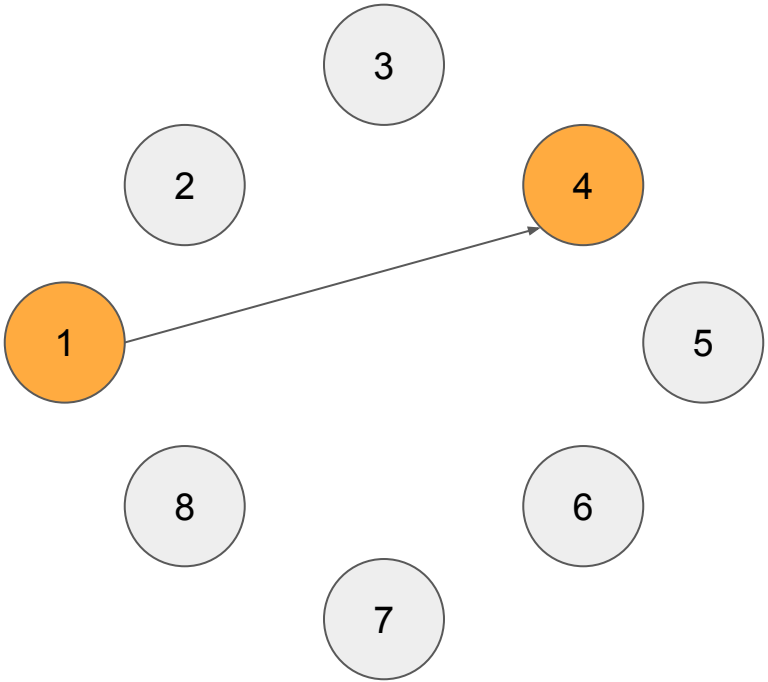
Gossip - Components

- Cluster Membership
- Failure Detection
- Custom Event Propagation

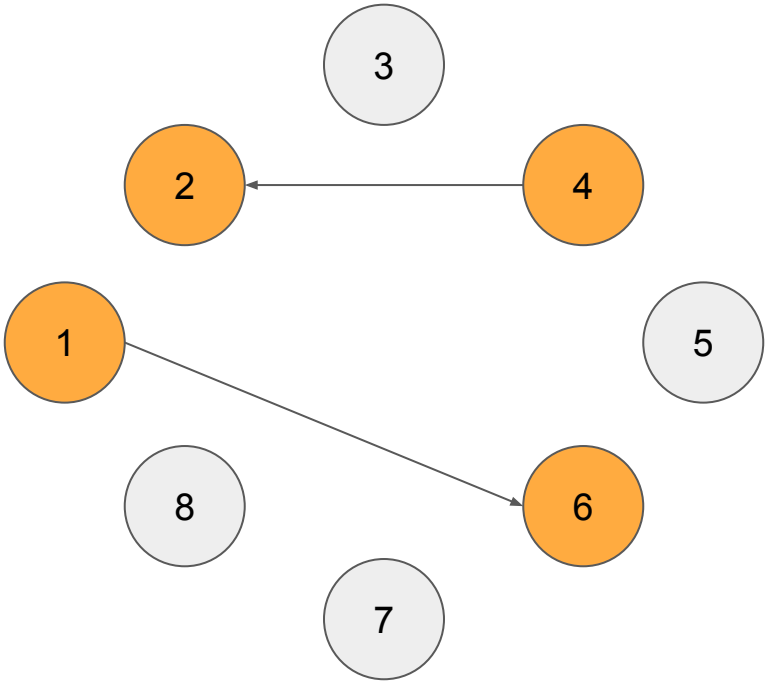
Without Gossip

- Cluster Membership
  - full static configuration of cluster members
  - cluster changes requires change on all members configuration
- Failure Detection
  - Heartbeat to central server
    - single point of failure
    - high density of network traffic to single node
  - Heartbeat to whole cluster
    - expensive broadcast
    - with increasing cluster size number of messages explodes
    - n * (n - 1) heartbeats required

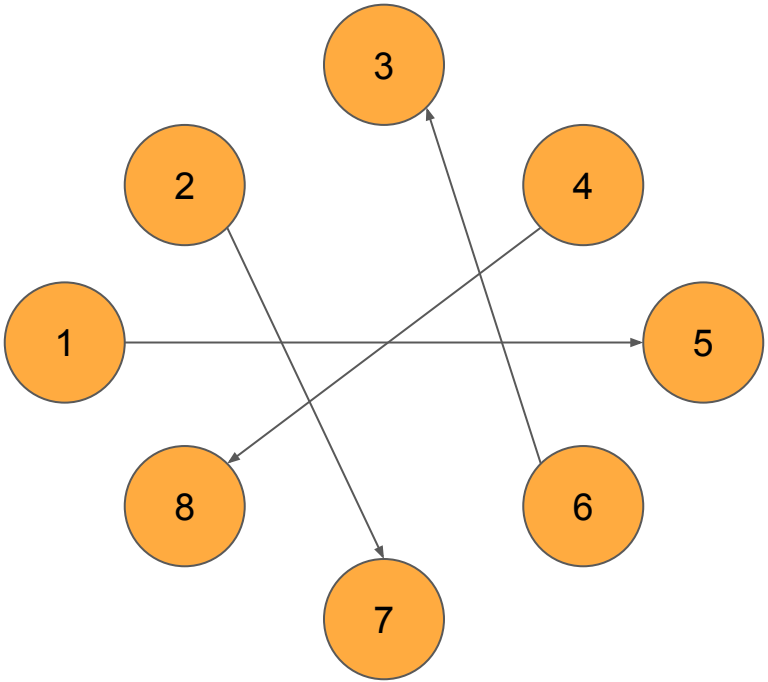
Gossip Pattern - 1. Iteration



Gossip Pattern - 2. Iteration



Gossip Pattern - 3. Iteration



Gossip - Cluster Membership

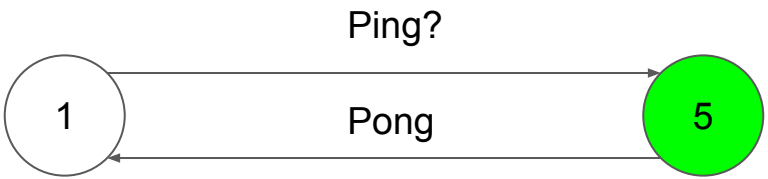
- p2p cluster
- every member has own view of the state of the cluster
- state changes are propagated to through the cluster
- received valid gossip messages are apply to own state and propagated to other nodes
- simple bootstrapping
  - reduce configuration by only require single contact point
  - on join sync with contact point current state
  - gossip to cluster own availability

Gossip - Failure Detection

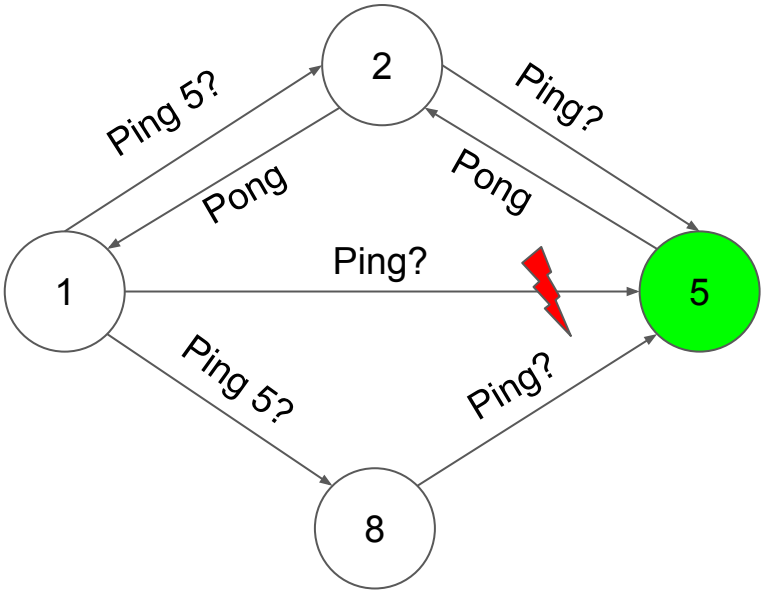
Two main concerns to optimize

1. Failure detection time should be small
2. False positives should be minimized, i.e. marking a node as failed although it is still alive

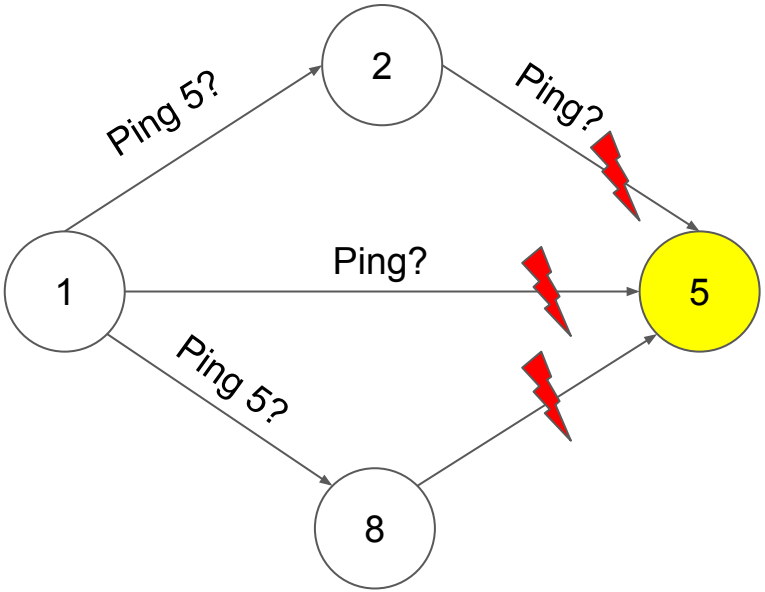
Gossip - Failure Detection - Alive



Gossip - Failure Detection - Alive (indirect)



Gossip - Failure Detection - Suspect



Gossip - Failure Detection

- If a node suspects another node to be failed it will gossip a suspicion event
- If the suspected node receives a suspicion event about itself it will refute this by gossiping a alive event
- If a suspected node did not refuted itself after a timeout it will be marked as failed and a failed event is gossiped
- Keep failed nodes in members list to check them occasionally
- If failed node is not revived after timeout remove it completely from members list

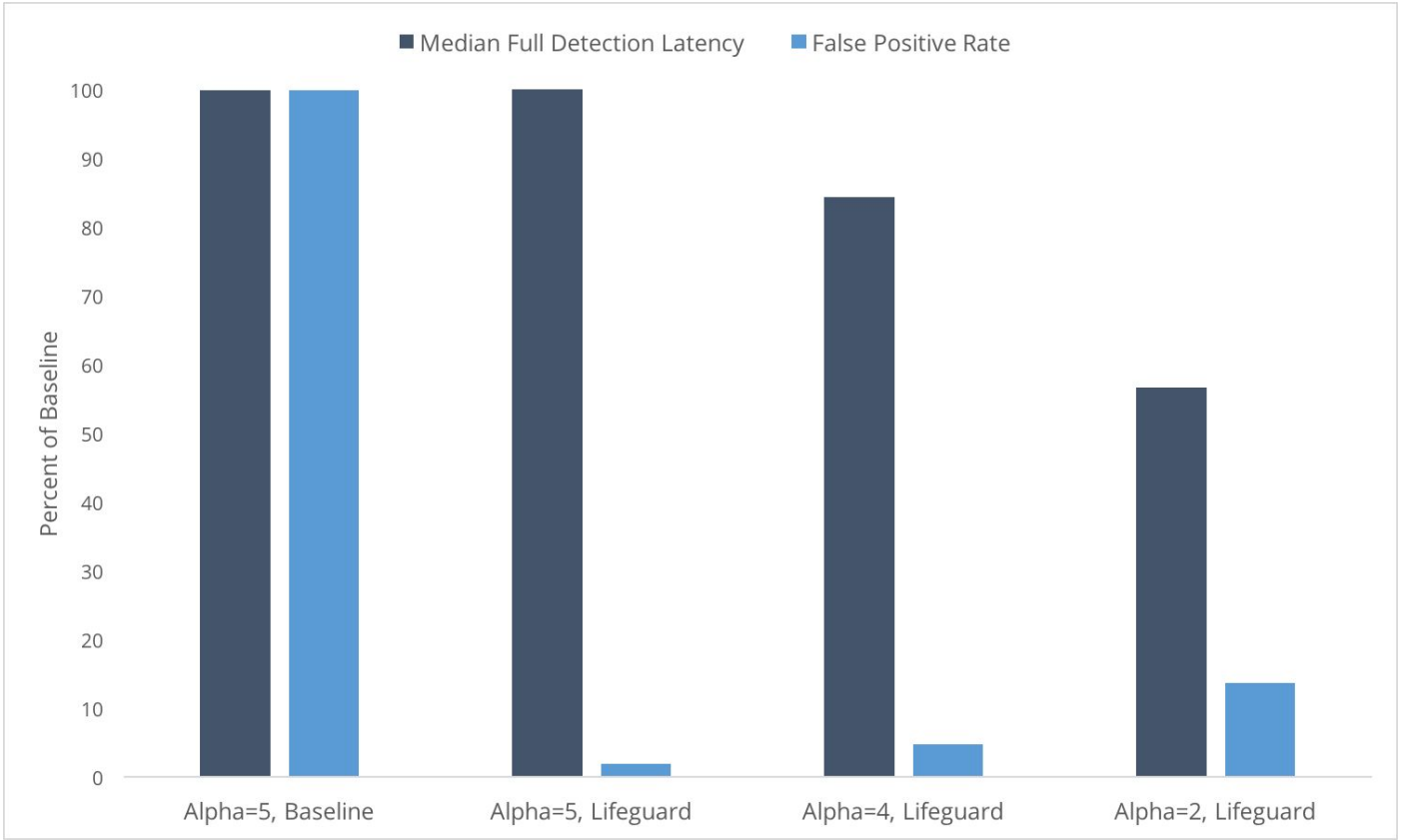
Gossip - Propagate Events

- Gossip can also be used to propagate custom events inside cluster
- In consul for example that a service is available or that a local service health check failed and the service is unavailable
- If the Consul server (Raft) leader node receives gossip messages it will update the service catalogue

Lifeguard : SWIM-ing with Situational Awareness

- Self-Awareness
  - allows a member to determine if it is currently degraded and to minimize the impact on the rest of the cluster by increasing the timeouts before suspecting other nodes
- Dogpile
  - replaces the fixed time frame to refute a failure with a dynamic one and logarithmically reduces as independent members confirm a member is failed
- Buddy System
  - notify a member that it is suspected of failure directly, which allows a member to refute a failure right away

Lifeguard : SWIM-ing with Situational Awareness



Consul and CI 2.0

- https://hq2.camunda.com/consul/#/hq/services
- Required by docker swarm for discovery and leader election
- Jenkins jobs use nginx.service.consul and nexus.service.consul for service discovery of the file server and nexus
- Initial setup around end of 2015, never touched after, just worked
- DNS forward to consul only for .consul domain by Bind DNS server, replaced by DNS on NAS in last weeks