TLA+ Conference 2024

Practical insights from Datadog's use of TLA+ and simulations

Arun Parthiban & Sesh Nalla

DATADOG

Hi, I’m Arun

- Senior Engineer at Datadog for 3 years.
  - Task-platform- queues, schedulers, execution runtime
  - Started on a 3 person team and bootstrapped newer systems
  - System grew, multiple teams; work across all teams now.
- Previously, Staff Engineer at Samsung
  - Actor based system IOT cloud, hundreds of millions of devices connect globally

DATADOG

Sesh Nalla

- Senior Director at Datadog for 5 years
  - Leads high performance transaction systems
- Prior experience applying formal methods
  - Air traffic control systems
  - Brokerage Trading systems
- Couldn’t attend due to other work commitments

DATADOG

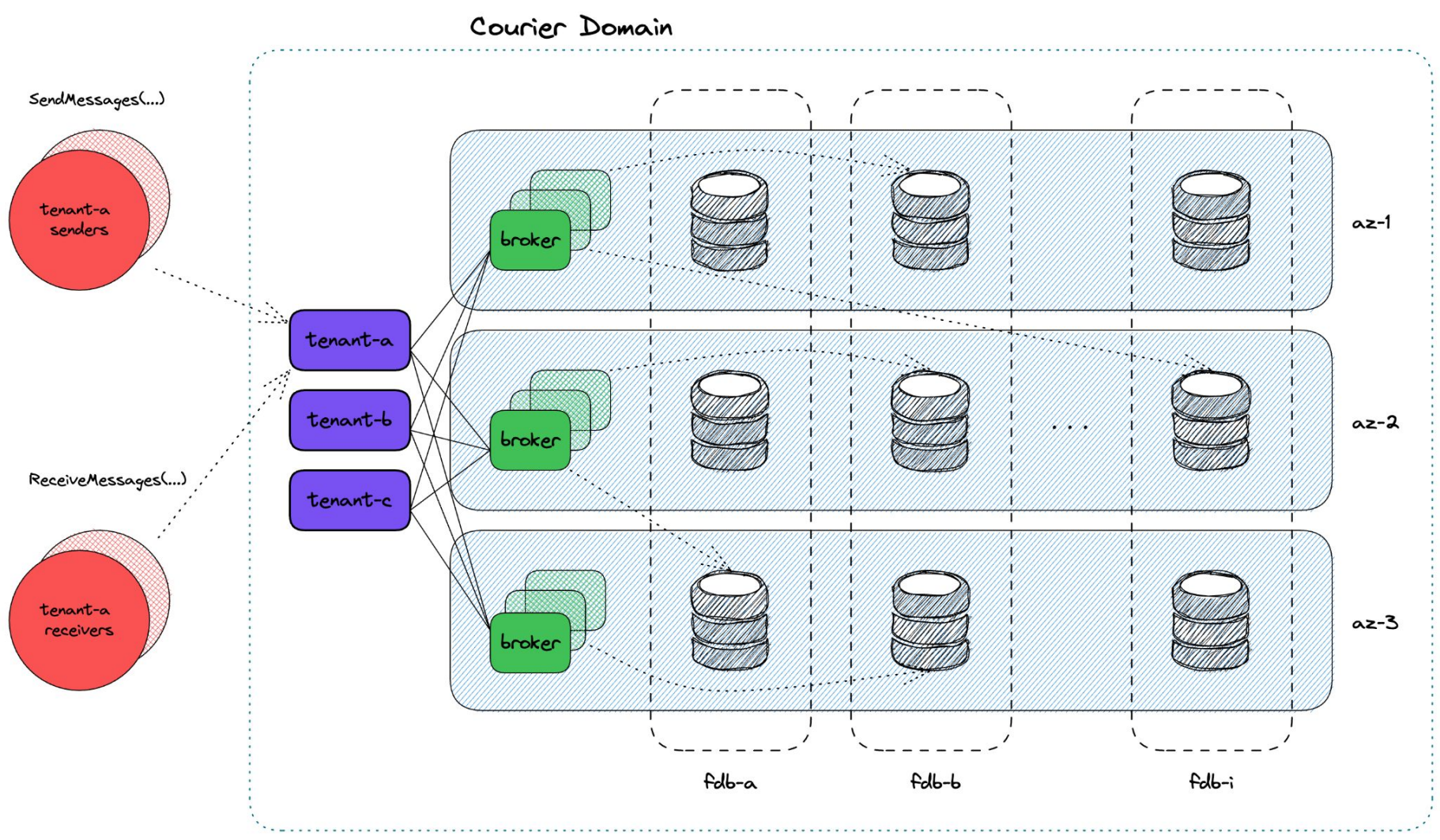
Evolution of queues at Datadog

- Redis based solution for 10+ years - Delancie
  - Single node throughput
  - Management overhead; sharding, upgrades etc.
  - Multi-tenant
- Additional requirements
  - Durability for new use cases
  - Millions of queues
  - Multi-cloud

Solution - Courier message queue

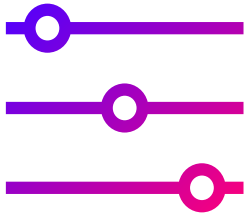
- Inspired by Apple's CloudKit queuing system(QuiCK)
  - "tens of **billions** of queues"
  - "QuiCK **scales linearly** with additional consumer resources, effectively avoids contention, provides **fairness** across"
- Similar to Delancie
  - Two layers of queueing
  - Leasing
- Built on FoundationDB
- APIs
  - SendMessage
  - ReceiveMessage
  - DeleteMessage

Solving for multi-tenancy



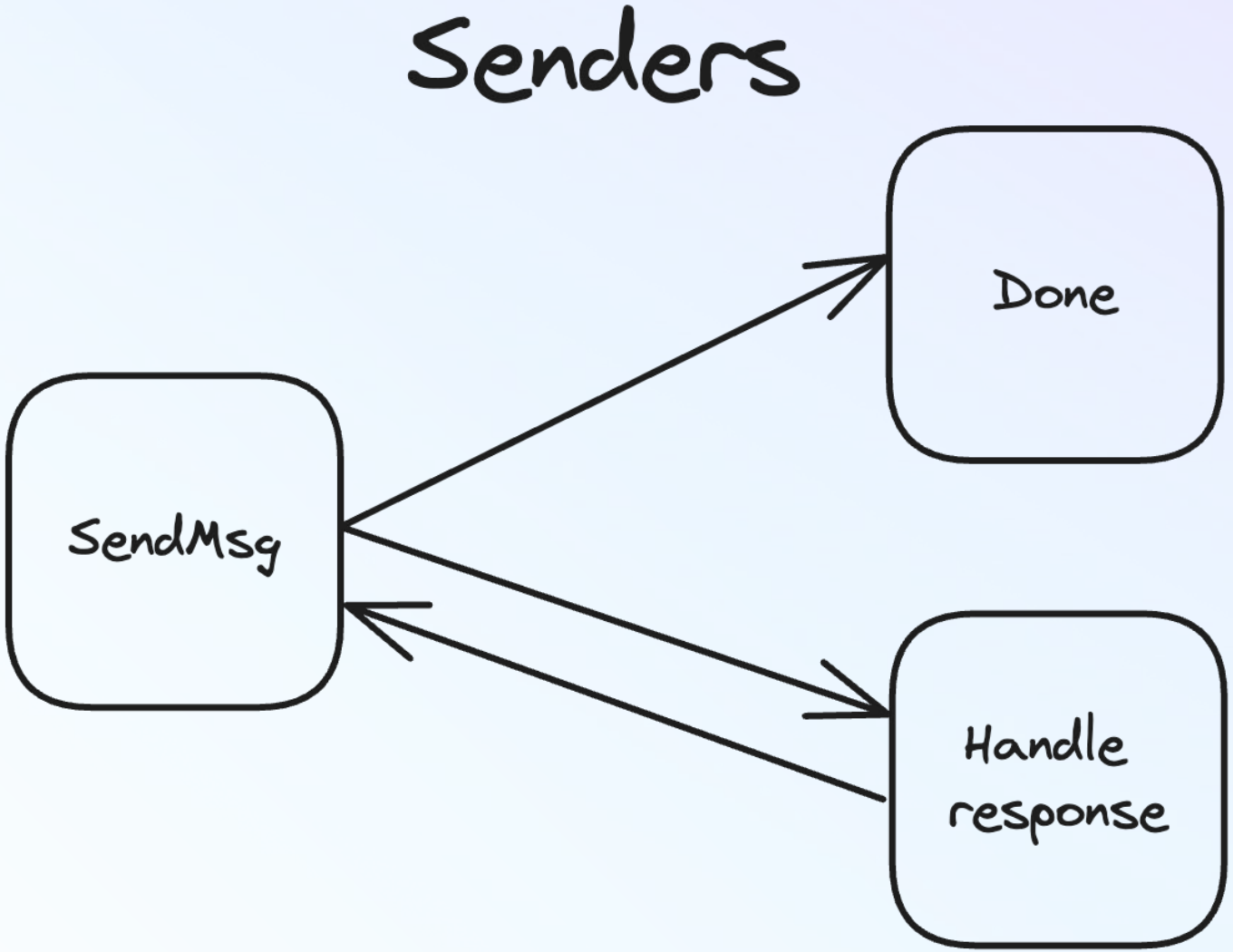
Can this system guarantee?

No lost messages

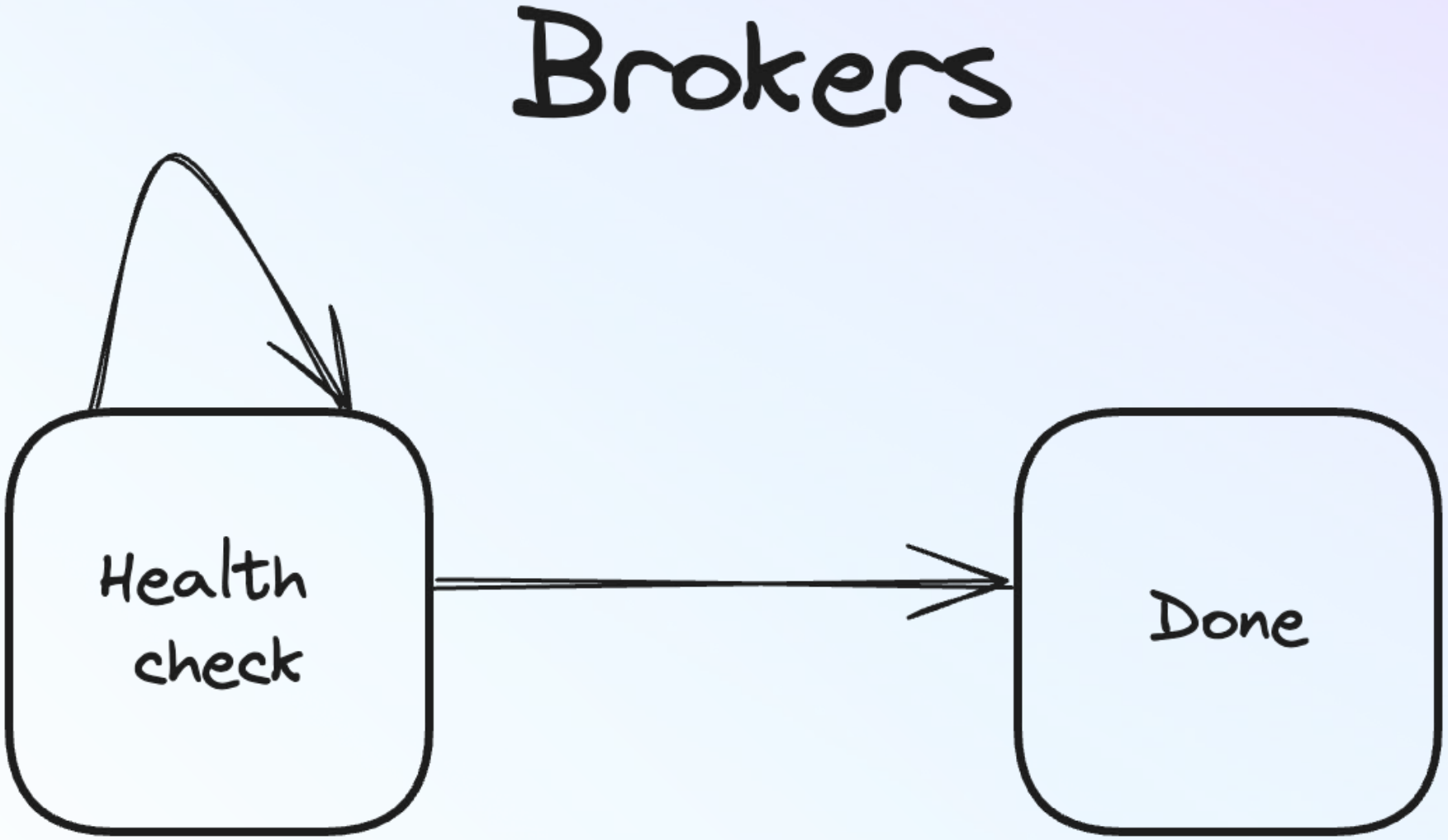


One active lease per message

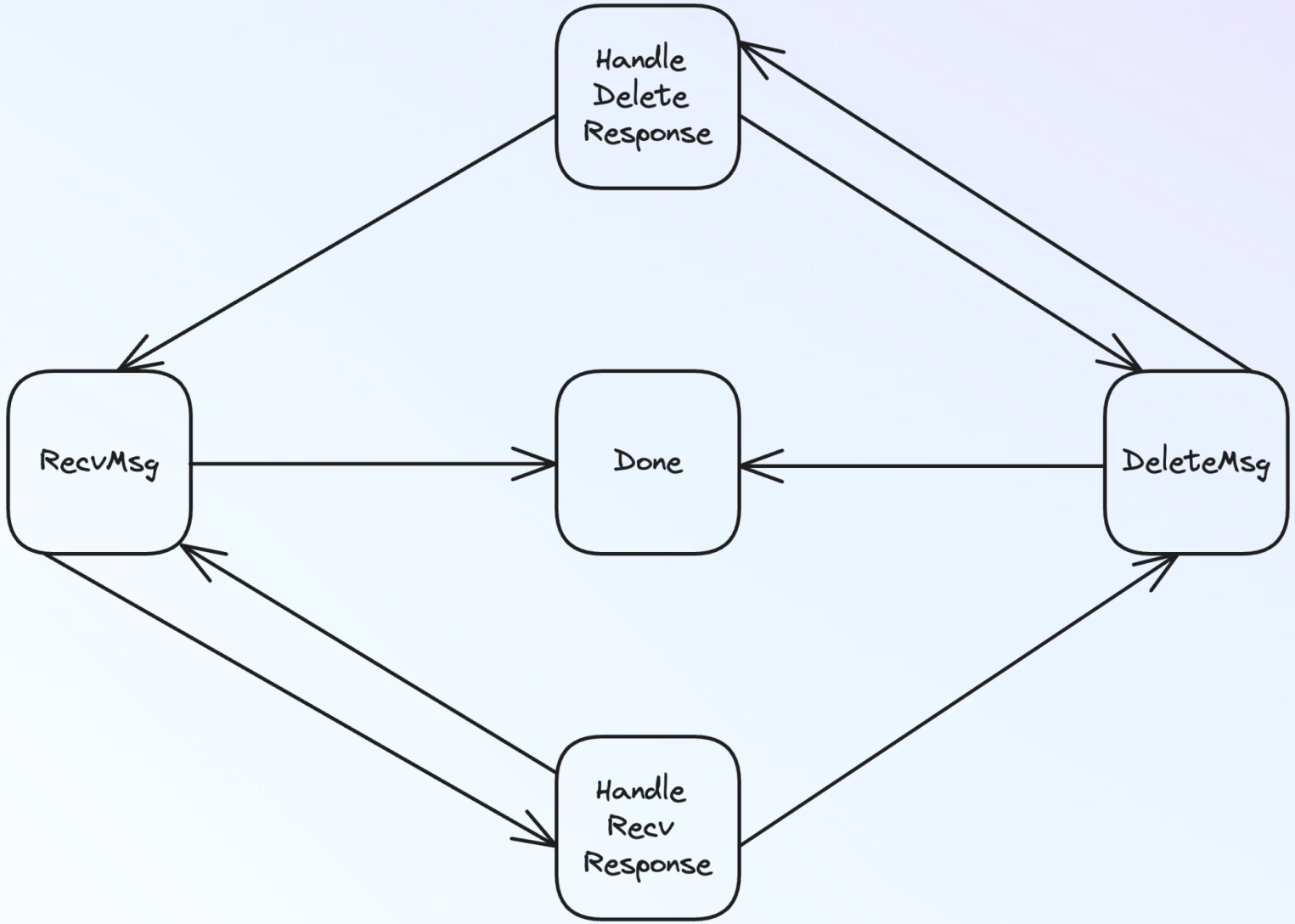
TLA+ Model: 3 processes



TLA+ Model: 3 processes

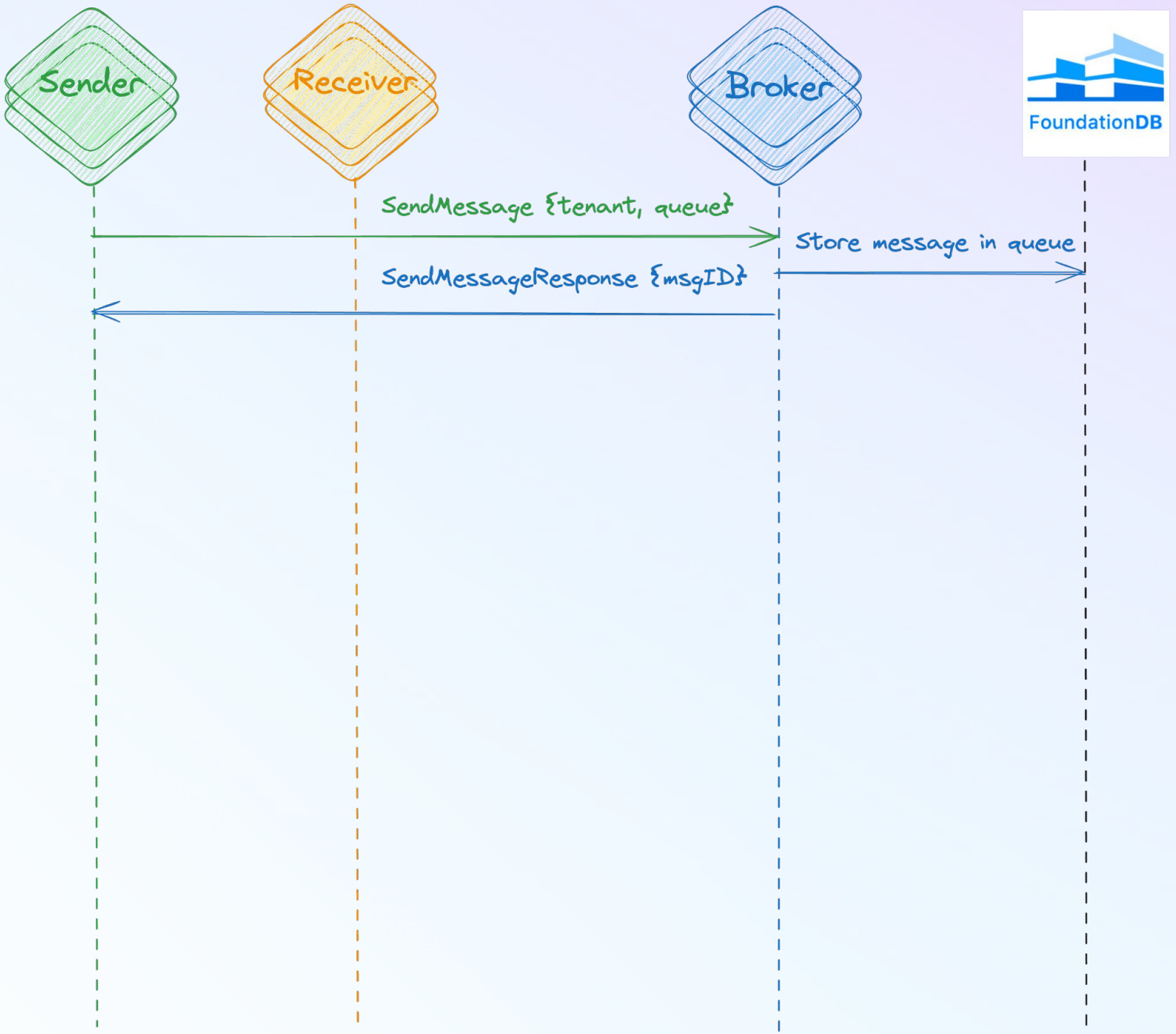


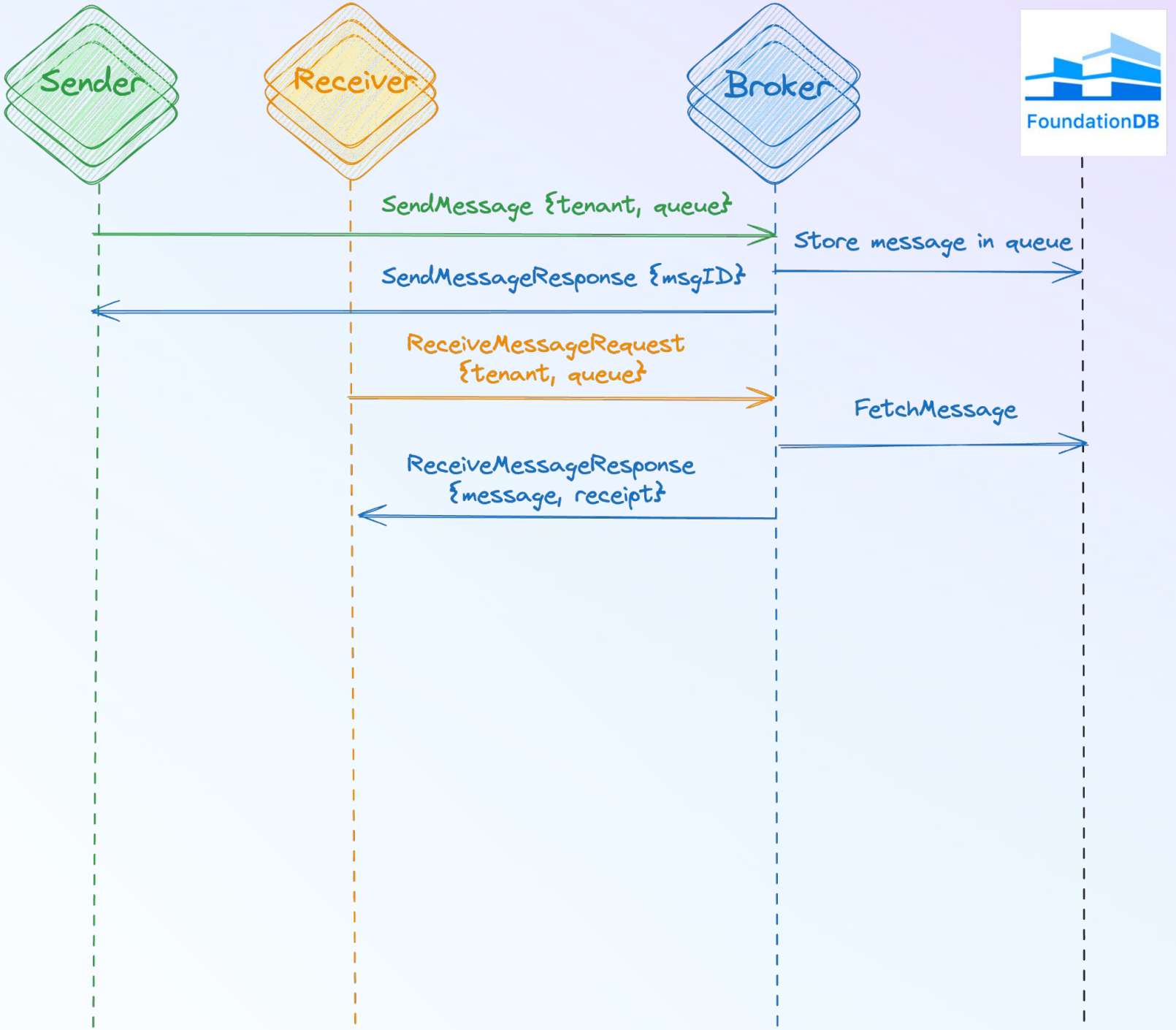
TLA+ Model: 3 processes

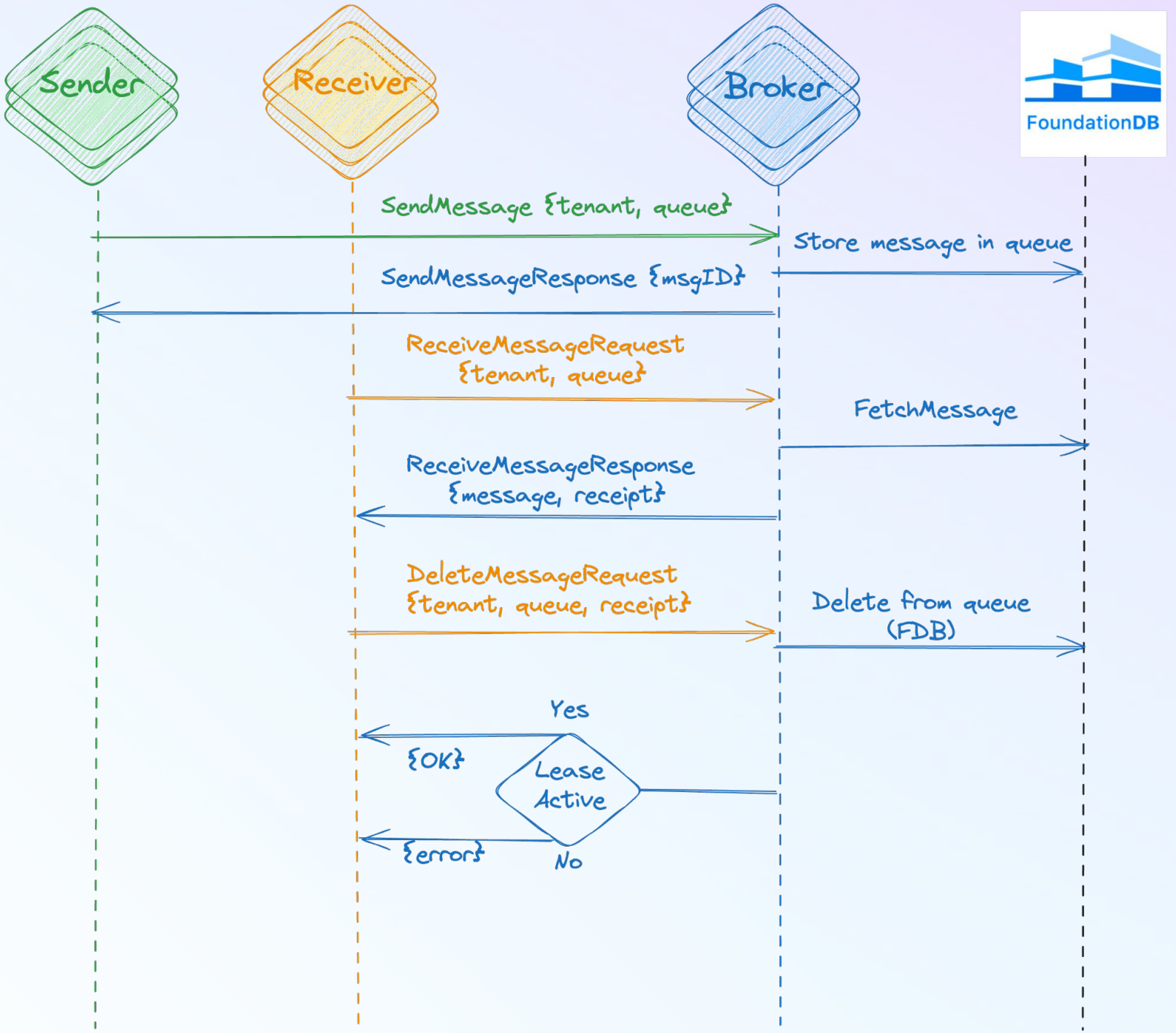


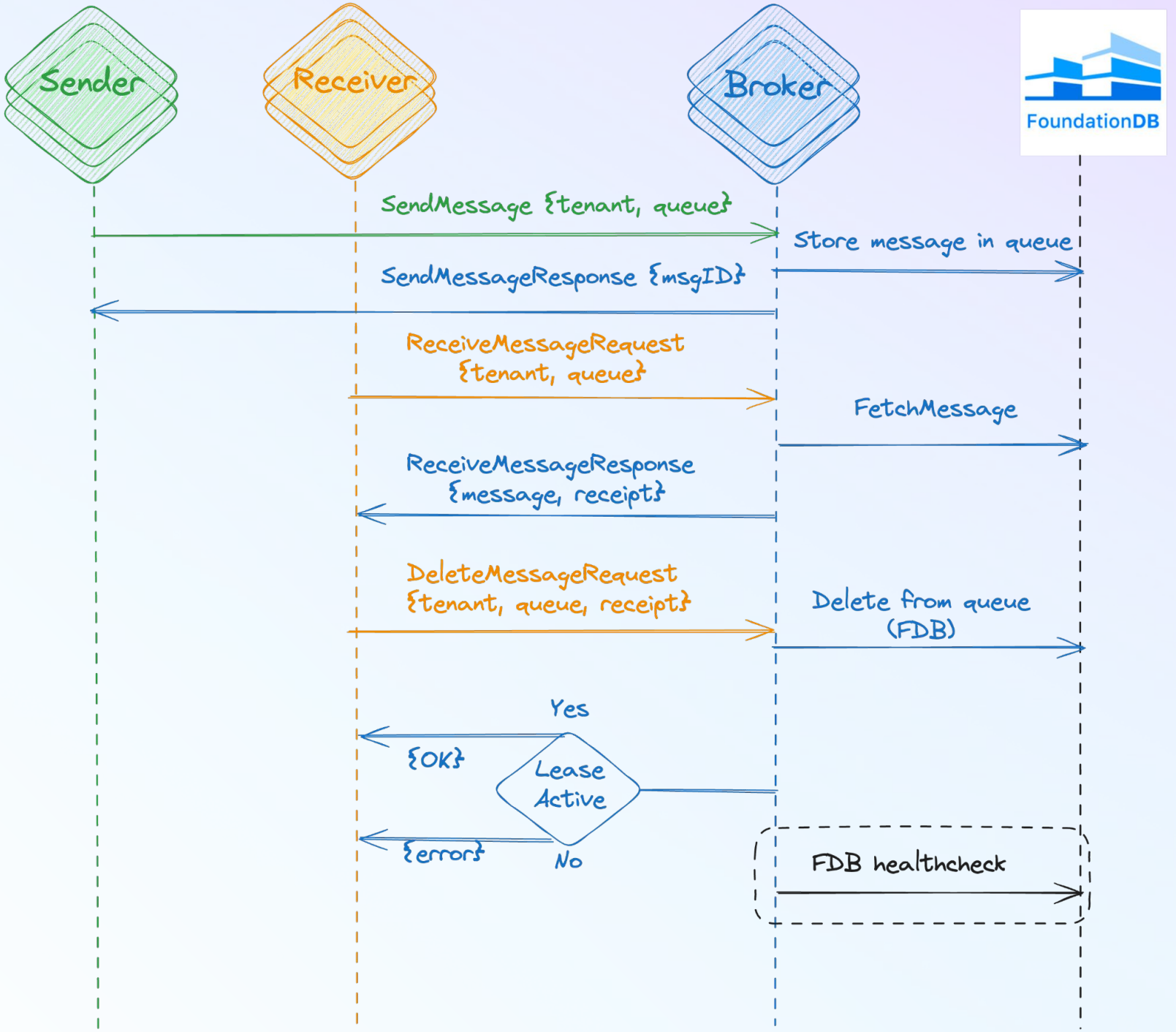
TLA+ Model: Variables

```
 8     VARIABLES
 9               \* FoundationDB clusters
10               clusters,
11               \* tracks stats around messages sent, received, deleted, etc.
12               stats,
13
14               \*  variables for coordinating between sender, broker and receiver
15               SendMsgOK,
16               SendMsgError,
17               ReceiveMsgOK,
18               ReceiveMsgError,
19               ReceiveMsgResult,
20               DeleteMsgOK,
21               DeleteMsgError,
```


Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}


Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}
ReceiveMessageRequest
{tenant, queue}
FetchMessage
ReceiveMessageResponse
{message, receipt}


Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}
ReceiveMessageRequest
{tenant, queue}
FetchMessage
ReceiveMessageResponse
{message, receipt}
DeleteMessageRequest
{tenant, queue, receipt}
Delete from queue
(FDB)
Yes
{OK}
Lease
Active
{error}
No


Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}
ReceiveMessageRequest
{tenant, queue}
FetchMessage
ReceiveMessageResponse
{message, receipt}
DeleteMessageRequest
{tenant, queue, receipt}
Delete from queue
(FDB)
Yes
{OK}
Lease
Active
{error}
No
FDB healthcheck

Model checker output

Status

Check again Full output

Checking courier.tla / courier.cfg

Success : Fingerprint collision probability: 6.1E-6

Start: 09:06:39 (Jun 24), end: 09:19:19 (Jun 24)

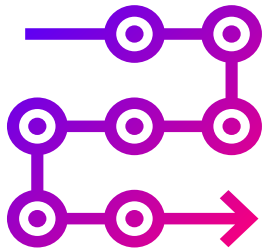
States

Time	Diameter	Found	Distinct	Queue
00:00:00	0	1	1	1
00:00:03	15	96 922	27 059	9 940
00:01:03	27	3 223 486	721 854	111 760
00:02:03	32	6 407 340	1 395 636	158 805
00:03:03	36	9 571 323	2 058 066	181 653
00:04:03	39	12 657 695	2 705 566	189 411
00:05:03	42	15 731 028	3 349 733	190 782
00:06:03	45	18 862 206	4 006 909	171 043
00:07:03	49	22 037 264	4 673 155	126 845
00:08:03	58	25 326 672	5 369 339	47 319
00:08:15	69	26 050 625	5 515 710	0
00:12:40	69	26 050 625	5 515 710	0

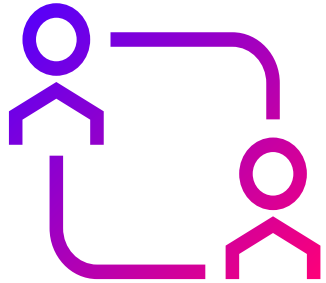
Coverage

Module	Action	Total	Distinct
courier	Init	1	1
courier	RestartBroker	5 515 715	41 970
courier	ClusterUnavailable	411 054	278 809
courier	ClusterAvailable	1 797 248	993 539
courier	ClusterHealthcheck	4 795 355	748 794
courier	SendMsg	2 059 007	46 402
courier	HandleSendMsgResponse	1 481 678	30 263
courier	ReceiveMsg	2 088 440	964 643
courier	HandleReceiveMsgResponse	3 819 990	1 257 642
courier	DeleteMsg	1 732 660	702 780
courier	HandleDeleteMsgResponse	2 334 900	450 867
courier	Terminating	14 612	0

Value of TLA+ model



Helped precision in parts of the implementation



Shared understanding & Language for team members

Arun Parthiban 10:19 AM

@mattbriancon I think this is a bug, not sure intentional or not: https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/cmd/broker/server.go#L81

server.go

```
            wl.Add(txn, sentAt.Add(msg.GetDelay().AsDuration()), msgId,
[]byte{})
```

DataDog/dd-source | Added by GitHub

5 replies Last reply 11 days ago

Arun Parthiban 3 days ago

I think there's a bug here: https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/cmd/broker/message.go#L62

message.go

```
    txn.Clear(ss)
```

DataDog/dd-source | Added by GitHub

+84 −130

+103 −166

Arun Parthiban 18 days ago

isn't this a concurrency bug in courier? https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/internal/router/connection.go#L59-L64

Arun Parthiban 7:12 AM

I think we can get rid of the snapshot reads in wl.randombefore

1 1 1

64 replies Last reply 11 days ago

Arun Parthiban 4:32 PM

actually, NVM its a bug. #1 doesn't do anything useful (edited)

1 reply 17 days ago

Arun Parthiban 10 days ago

@mattbriancon I want to make sure we both have the same understanding on ss.Remove. I think there's a bug

✅ 1

Arun Parthiban 11 days ago

I think I'm convinced it will conflict, but I may have found some edge case with RandomBefore https://github.com/DataDog/dd-source/pull/38349/files

Arun Parthiban 8 days ago

Arun Parthiban 8 days ago

https://github.com/DataDog/dd-source/pull/38936

#38936 Fix bug in TTL logic

Labels

team:Task Platform

DataDog/dd-source | Jun 23rd | Added by GitHub

Arun Parthiban 10:19 AM
@mattbriancon I think this is a bug, not sure intentional or not: https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/cmd/broker/server.go#L81

server.go

```
            wl.Add(txn, sentAt.Add(msg.GetDelay().AsDuration()), msgId,
[]byte{})
```

DataDog/dd-source | Added by GitHub

5 replies Last reply 11 days ago

Arun Parthiban 3 days ago
I think there's a bug here: https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/cmd/broker/message.go#L62

message.go

```
    txn.Clear(ss)
```

DataDog/dd-source | Added by GitHub

+84 −130

+103 −166

Arun Parthiban 18 days ago
isn't this a concurrency bug in courier?
https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/internal/router/connection.go#L59-L64

Arun Parthiban 7:12 AM
I think we can get rid of the snapshot reads in wl.randombefore
1 1 1

64 replies Last reply 11 days ago

Arun Parthiban 4:32 PM
actually, NVM its a bug. #1 doesn't do anything useful (edited)

1 reply 17 days ago

Arun Parthiban 10 days ago
@mattbriancon I want to make sure we both have the same understanding on ss.Remove. I think there's a bug
✅ 1

Arun Parthiban 11 days ago
I think I'm convinced it will conflict, but I may have found some edge case with RandomBefore https://github.com/DataDog/dd-source/pull/38349/files

Arun Parthiban 8 days ago

Arun Parthiban 8 days ago
https://github.com/DataDog/dd-source/pull/38936

#38936 Fix bug in TTL logic
Labels
team:Task Platform
DataDog/dd-source | Jun 23rd | Added by GitHub

Arun Parthiban 10:19 AM
@mattbriancon I think this is a bug, not sure intentional or not: https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/cmd/broker/server.go#L81

server.go

```
            wl.Add(txn, sentAt.Add(msg.GetDelay().AsDuration()), msgId,
[]byte{})
```

DataDog/dd-source | Added by GitHub

5 replies Last reply 11 days ago

Arun Parthiban 3 days ago
I think there's a bug here: https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/cmd/broker/message.go#L62

message.go

```
    txn.Clear(ss)
```

DataDog/dd-source | Added by GitHub

+84 −130

+103 −166

Arun Parthiban 18 days ago
isn't this a concurrency bug in courier?
https://github.com/DataDog/dd-source/blob/main/domains/task_platform/apps/courier/internal/router/connection.go#L59-L64

Arun Parthiban 7:12 AM
I think we can get rid of the snapshot reads in wl.randombefore
1 1 1

64 replies Last reply 11 days ago

Arun Parthiban 4:32 PM
actually, NVM its a bug. #1 doesn't do anything useful (edited)

1 reply 17 days ago

Arun Parthiban 10 days ago
@mattbriancon I want to make sure we both have the same understanding on ss.Remove. I think there's a bug
✅ 1

Arun Parthiban 11 days ago
I think I'm convinced it will conflict, but I may have found some edge case with RandomBefore https://github.com/DataDog/dd-source/pull/38349/files

Arun Parthiban 8 days ago

Arun Parthiban 8 days ago
https://github.com/DataDog/dd-source/pull/38936

#38936 Fix bug in TTL logic
Labels
team:Task Platform
DataDog/dd-source | Jun 23rd | Added by GitHub

How we started

- Pluscal to verify idempotency in Husky, Datadog's wide-columnar storage
  - Researched models from CosmosDB, CockroachDB
  - Modeled post-production. Pluscal syntax made this easier
  - Large state space
- Courier
  - Started with Pluscal, too many states, slow to check
  - Re-wrote in TLA+, more control over state transitions
- Used Pluscal for modeling production bug fixes in Chrono, Datadog's cron scheduler

Marc's Blog

Formal Methods Only Solve Half My Problems

What latency can customers expect, on average and in outlier cases? What will it cost us to run this service? How do those costs scale with different usage patterns, and dimensions of load (data size, throughput, transaction rates, etc)? What type of hardware do we need for this service, and how much? How sensitive is the design to network latency or packet loss? How do availability and durability scale with the number of replicas? How will the system behave under overload?

Formal Methods Only Solve Half My Problems - Marc's Blog

[2023-03-08 Incident: Infrastructure Connectivity Issue Affecting Multiple Regions | Datadog]

Lessons learned

- Graceful degradation
  - System should degrade linearly with compute loss
- Failure modes of quorum based systems
- How will Courier fare?

Simulations

Obtaining statistical properties by simulating specs with TLC

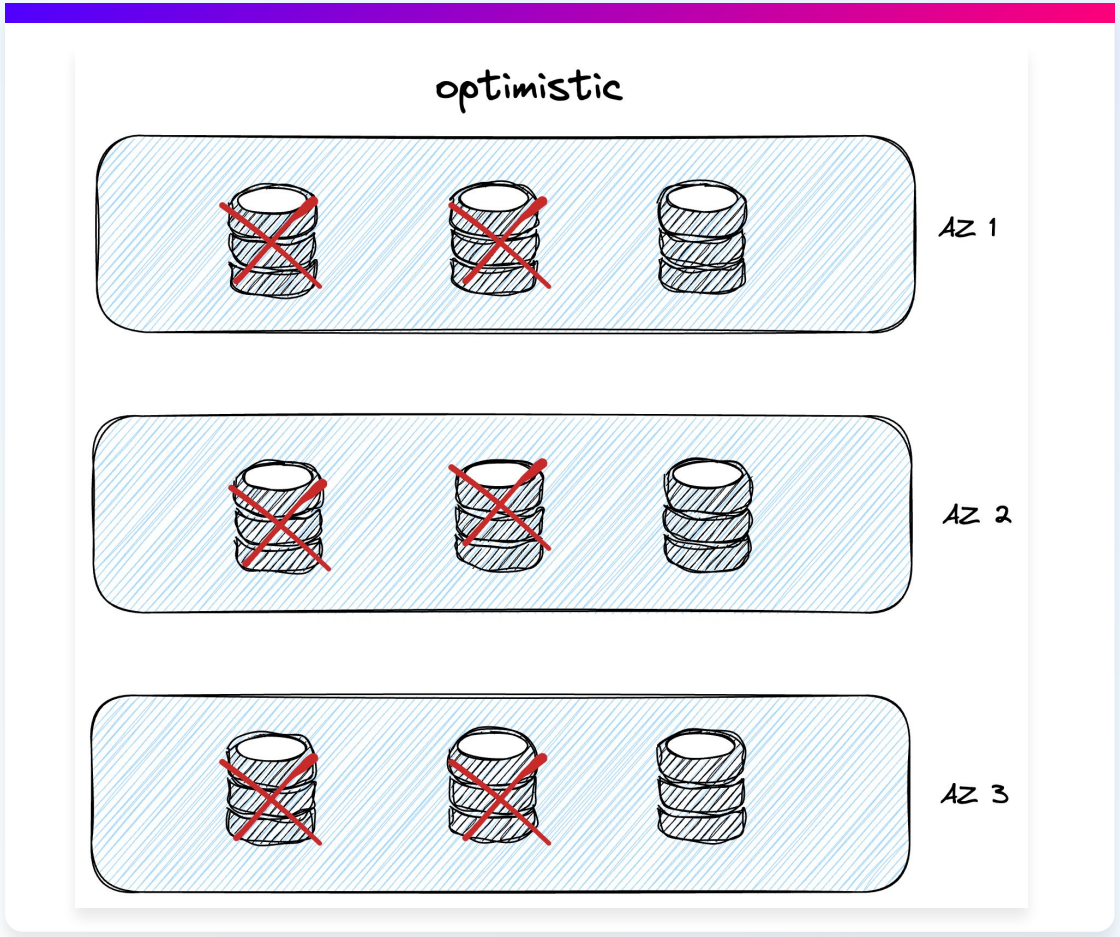
Jack Vanlightly & Markus A. Kuppe

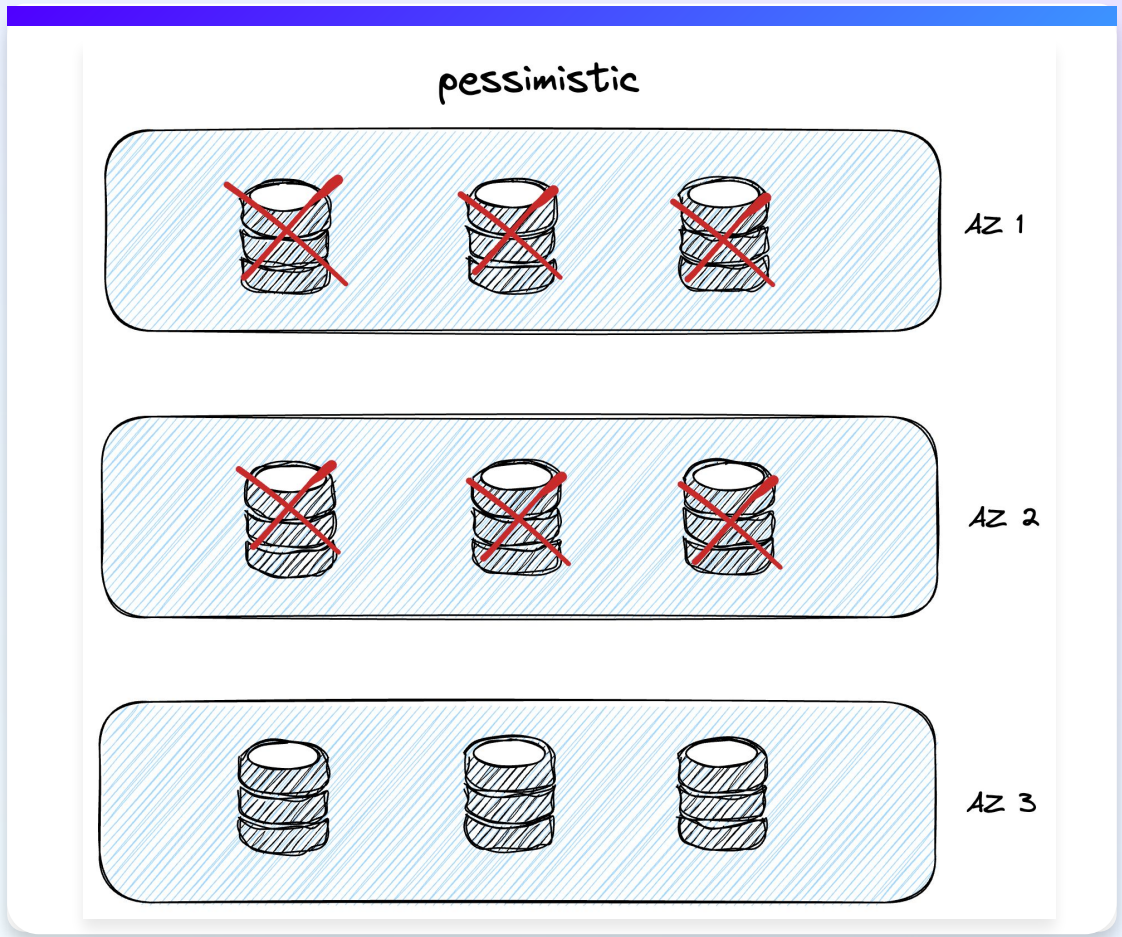
Simulating Courier

- SimPy: discrete event simulation library in Python
- Simulated senders, receivers, brokers, and FDB
- Measured throughput and availability against node loss

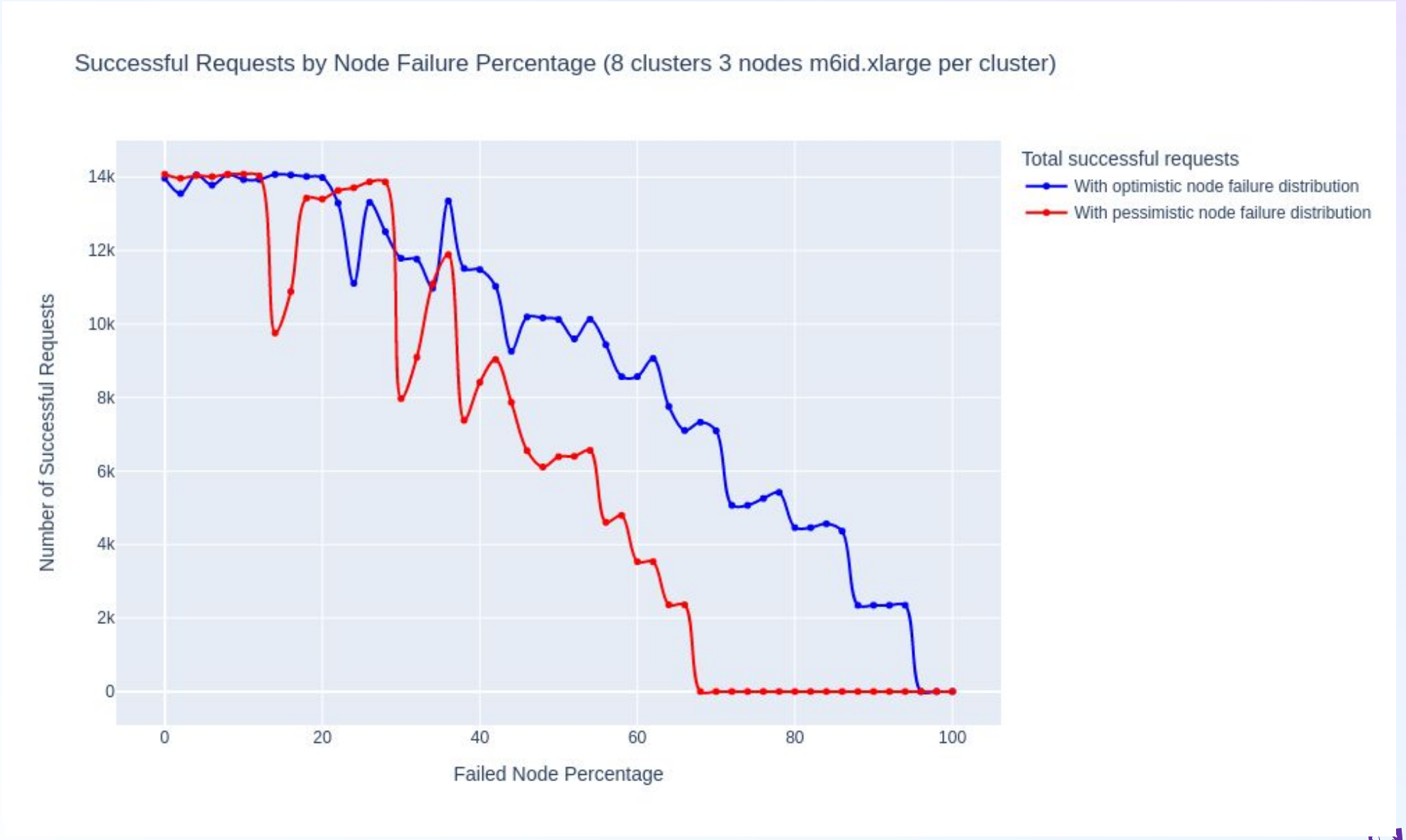
```
INSTANCES = {
    "m6id.xlarge": {"cpu": 4, "monthly_cost": 64, "memory": 16},
}

NUM_CLUSTERS = 8
NODES_PER_CLUSTER = 3
NUM_COURIER_PODS = 20
INSTANCE_TYPE = "m6id.xlarge"

# Numbers source: https://apple.github.io/foundationdb/performance.html
FDB_START_TRAN = {"min": 0.0003, "max": 0.001}
FDB_READ = {"min": 0.0001, "max": 0.001}
FDB_COMMIT = {"min": 0.0015, "max": 0.0025}
FDB_READ_LATENCY = {"min": FDB_READ["min"], "max": FDB_READ["max"]}
FDB_WRITE_LATENCY = {
    "min": FDB_READ["min"] + FDB_COMMIT["min"],
    "max": FDB_READ["max"] + FDB_COMMIT["max"]
}

FDB_OPERATIONS = {
    "enqueue": {"trans": 1, "reads": 1, "writes": 1},
    "dequeue": {"trans": 4, "reads": 2, "writes": 5},
    "complete": {"trans": 1, "reads": 1, "writes": 1},
}

FDB_CONCURRENT_OPS_PER_PROCESS = 15
NUM_TENANTS = 70
CLUSTERS_PER_TENANT = 4
```

Simulation scenarios

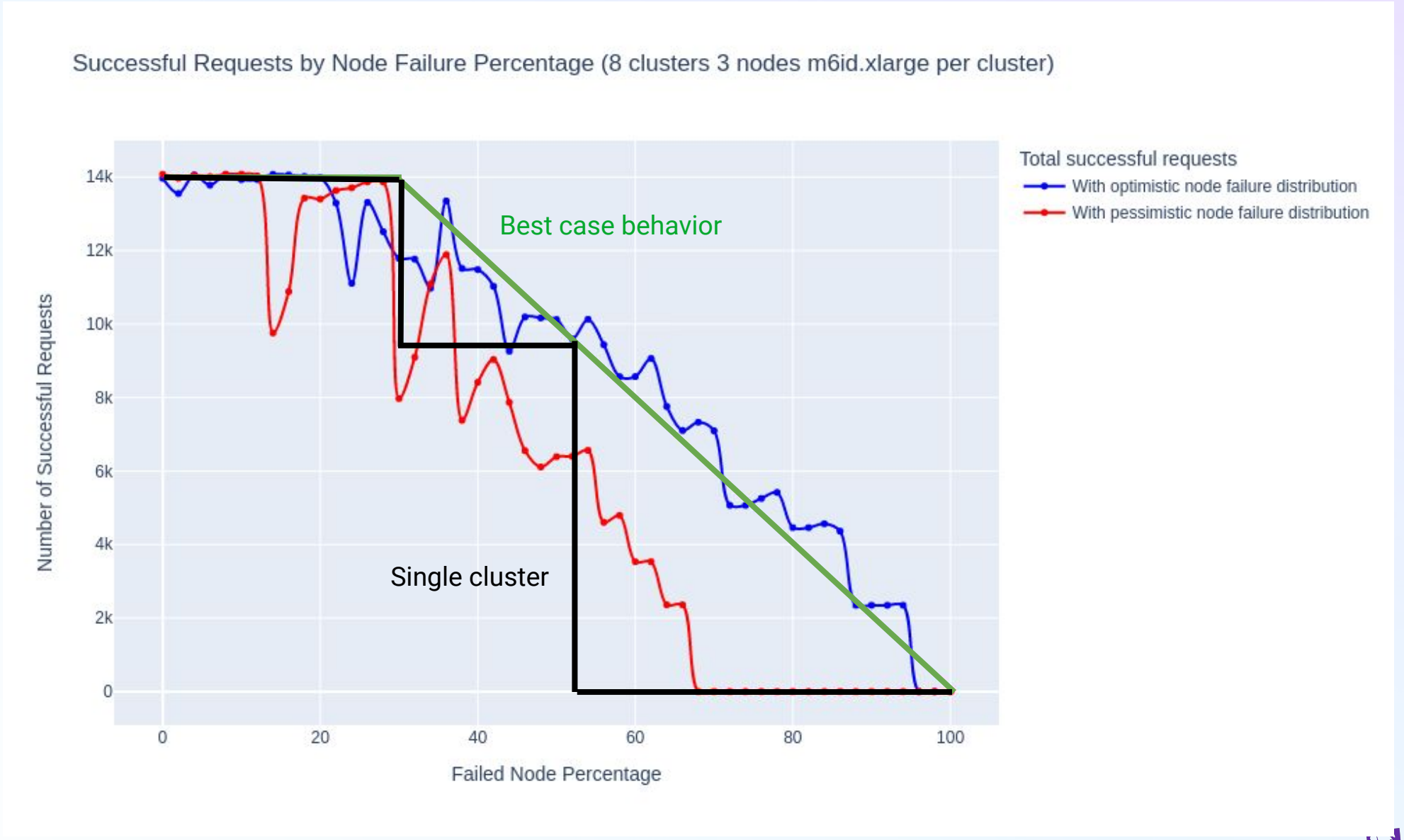




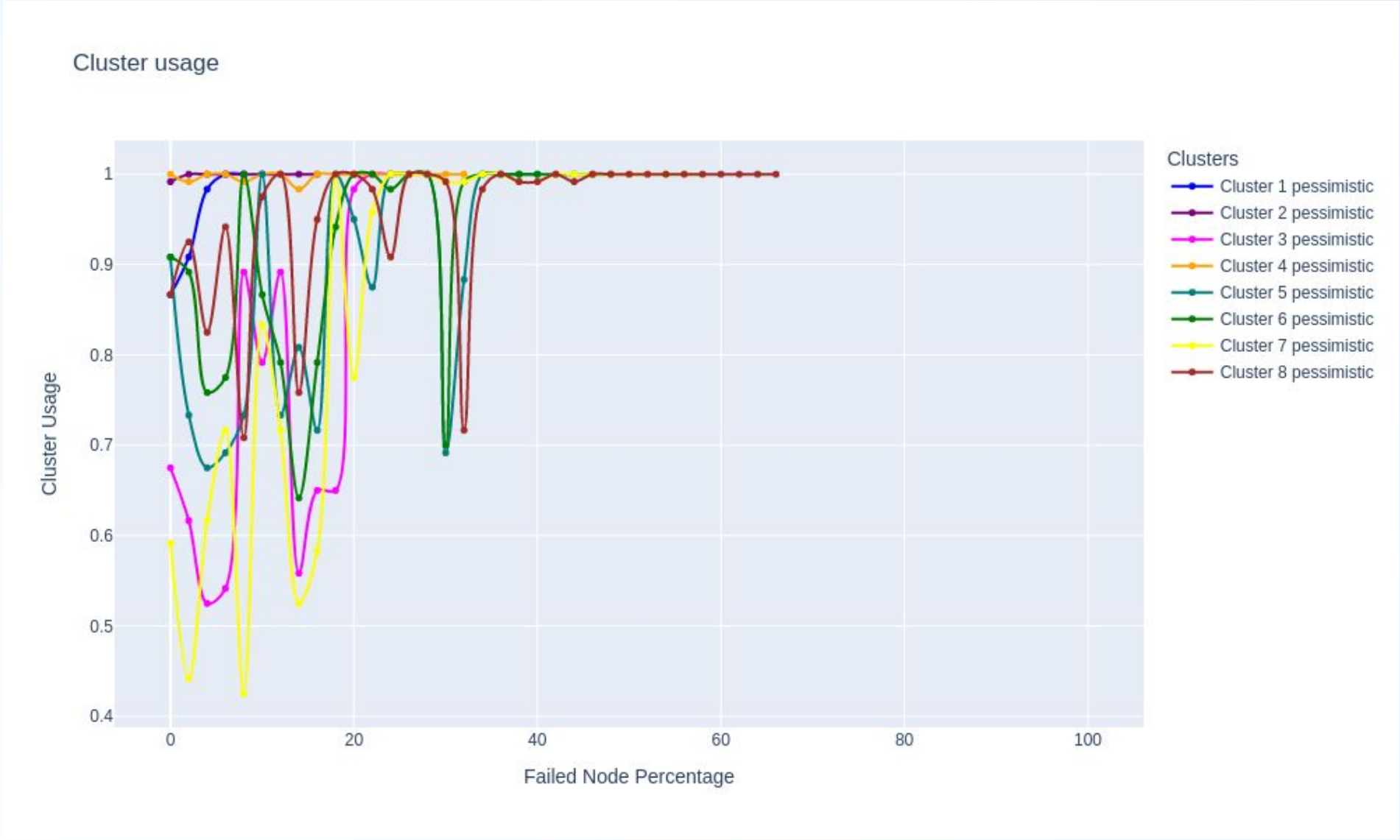
Simulation results



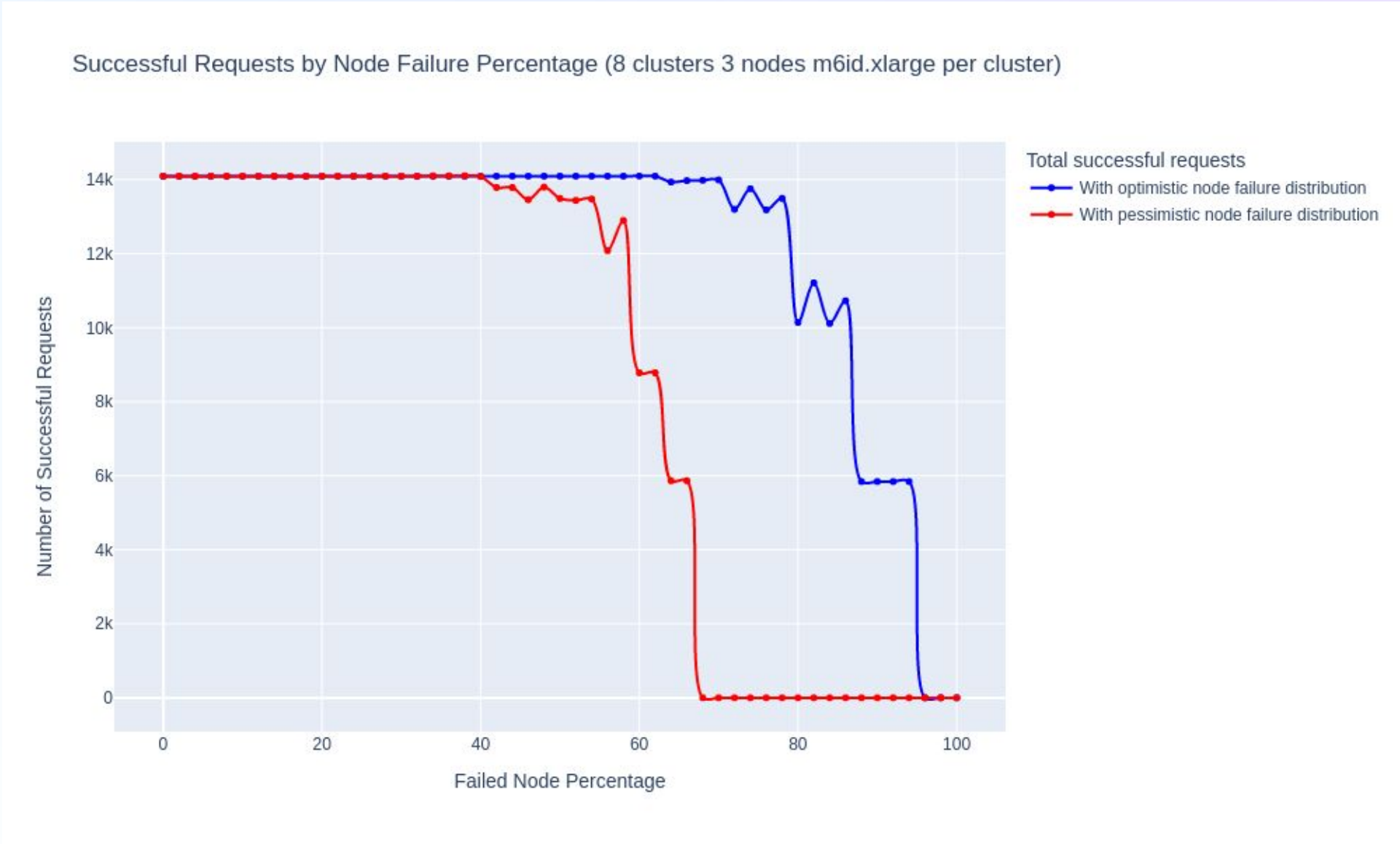
Simulation results



Debugging throughput oscillations

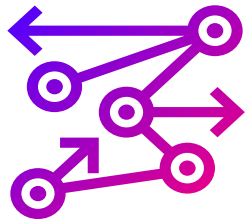


Simulations - overprovisioned

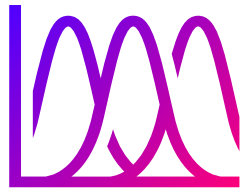


Chaos Testing - pessimistic

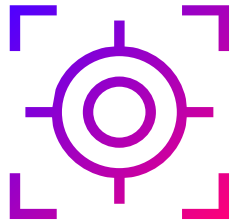
Value of simulations



Recreate complex failure modes



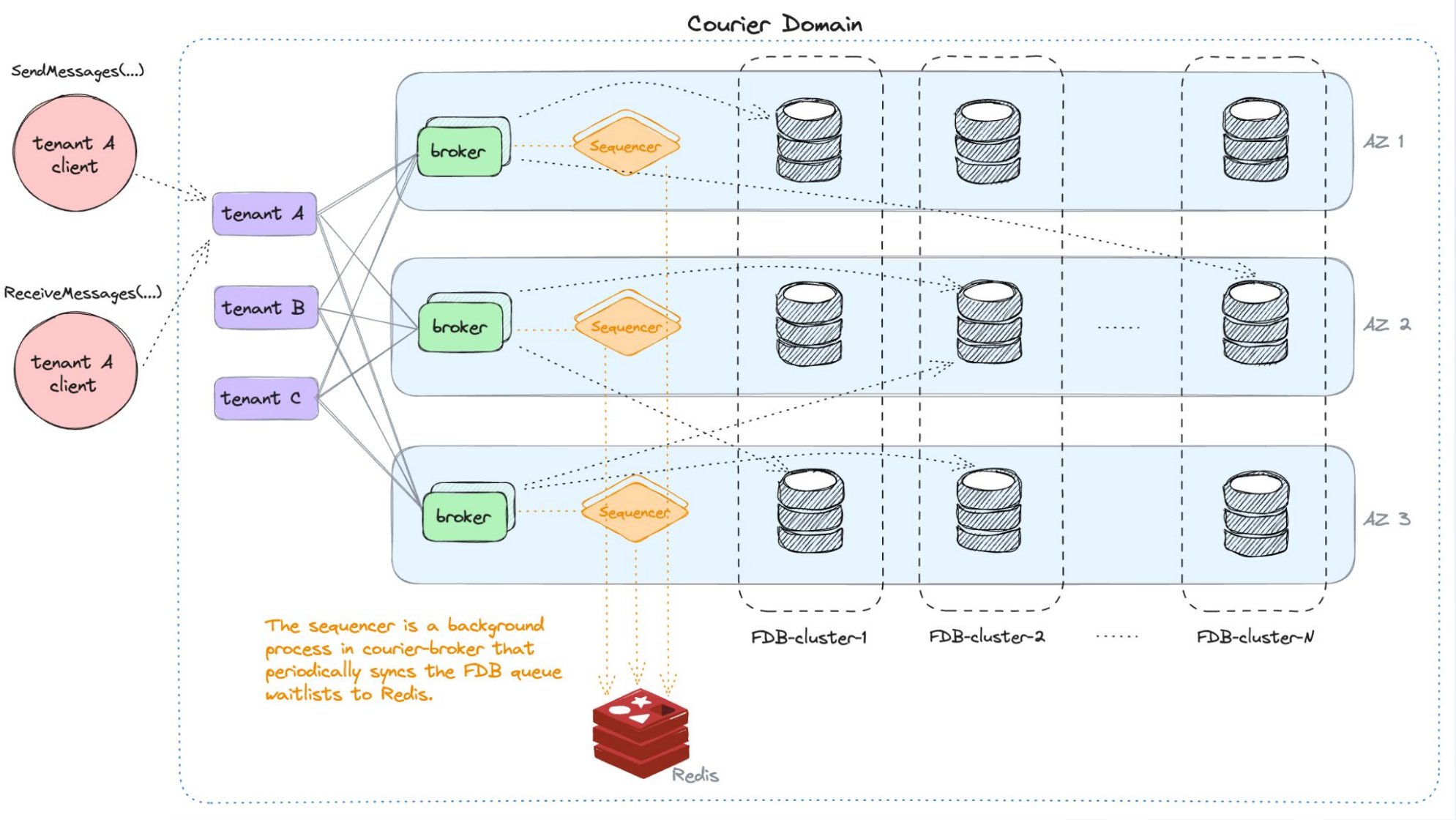
Range of impact

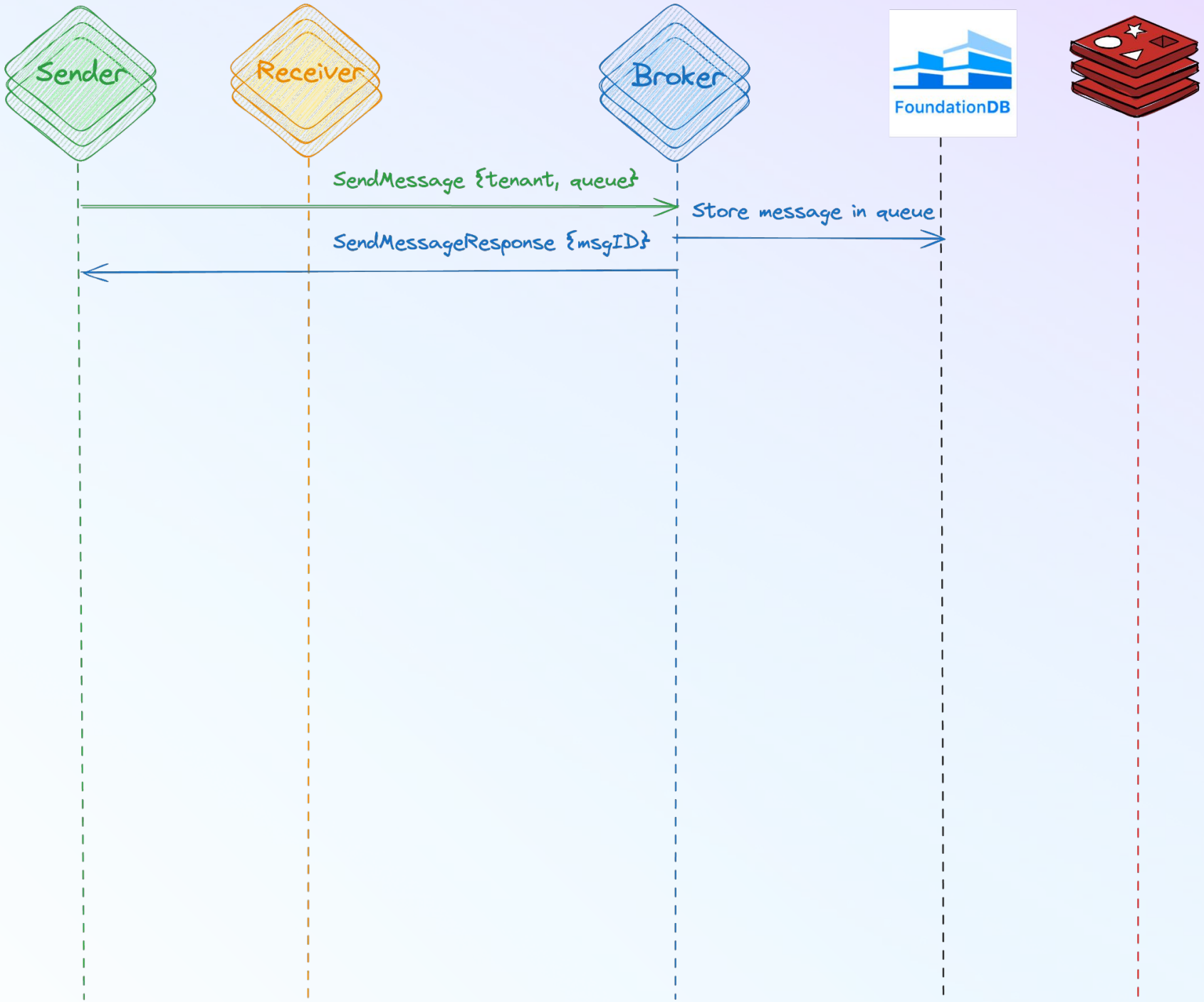


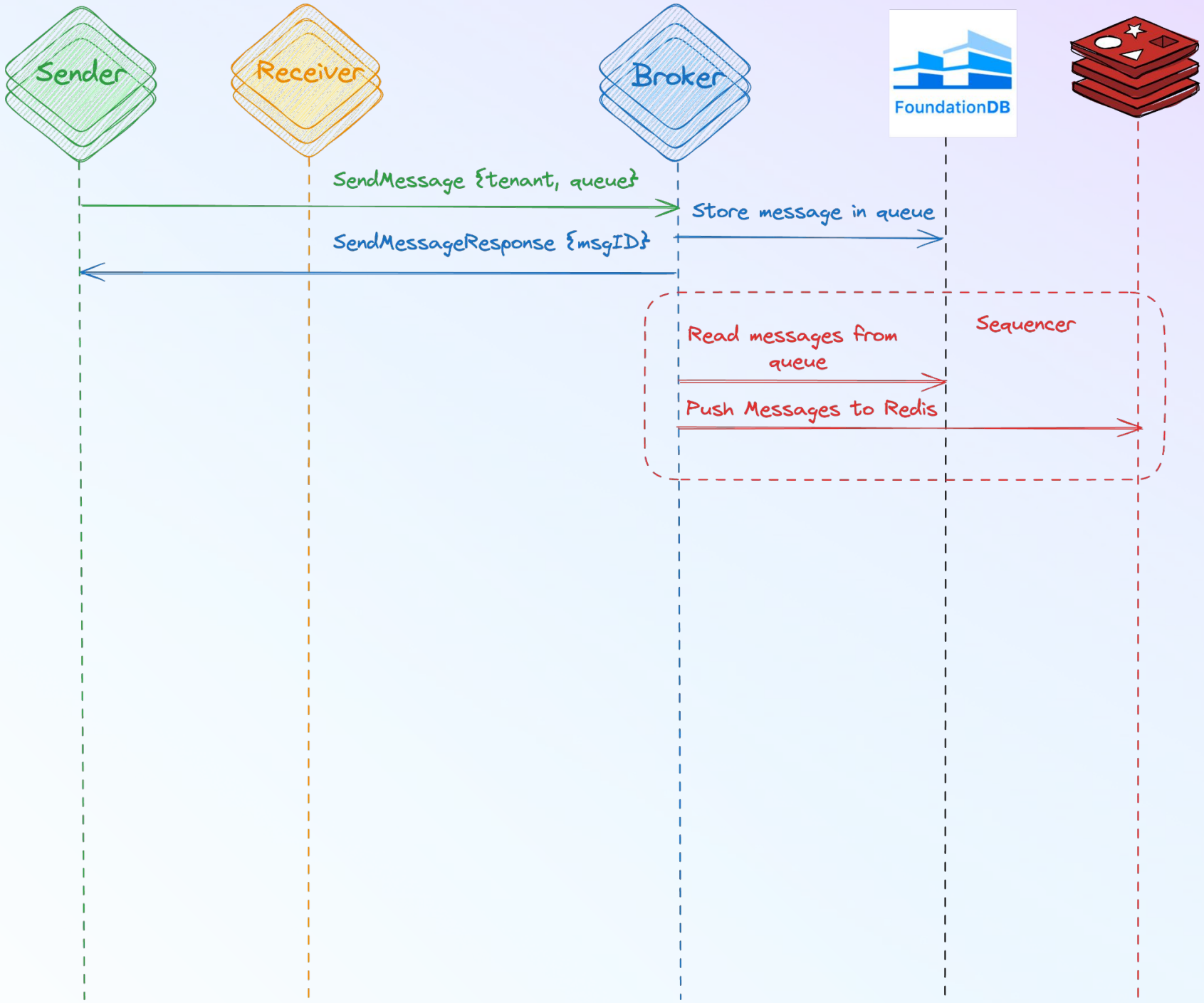
Targeted chaos testing

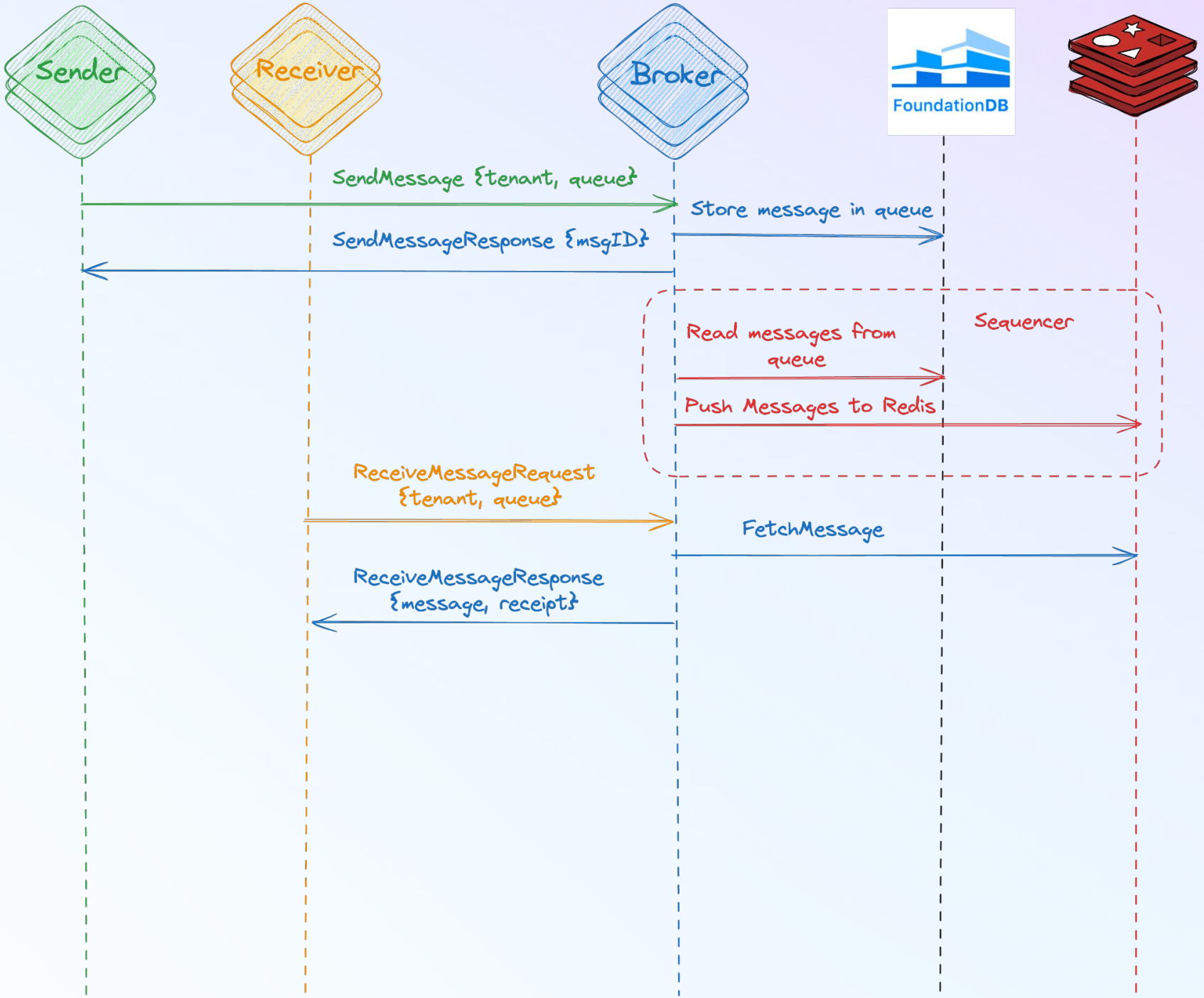
Design changes

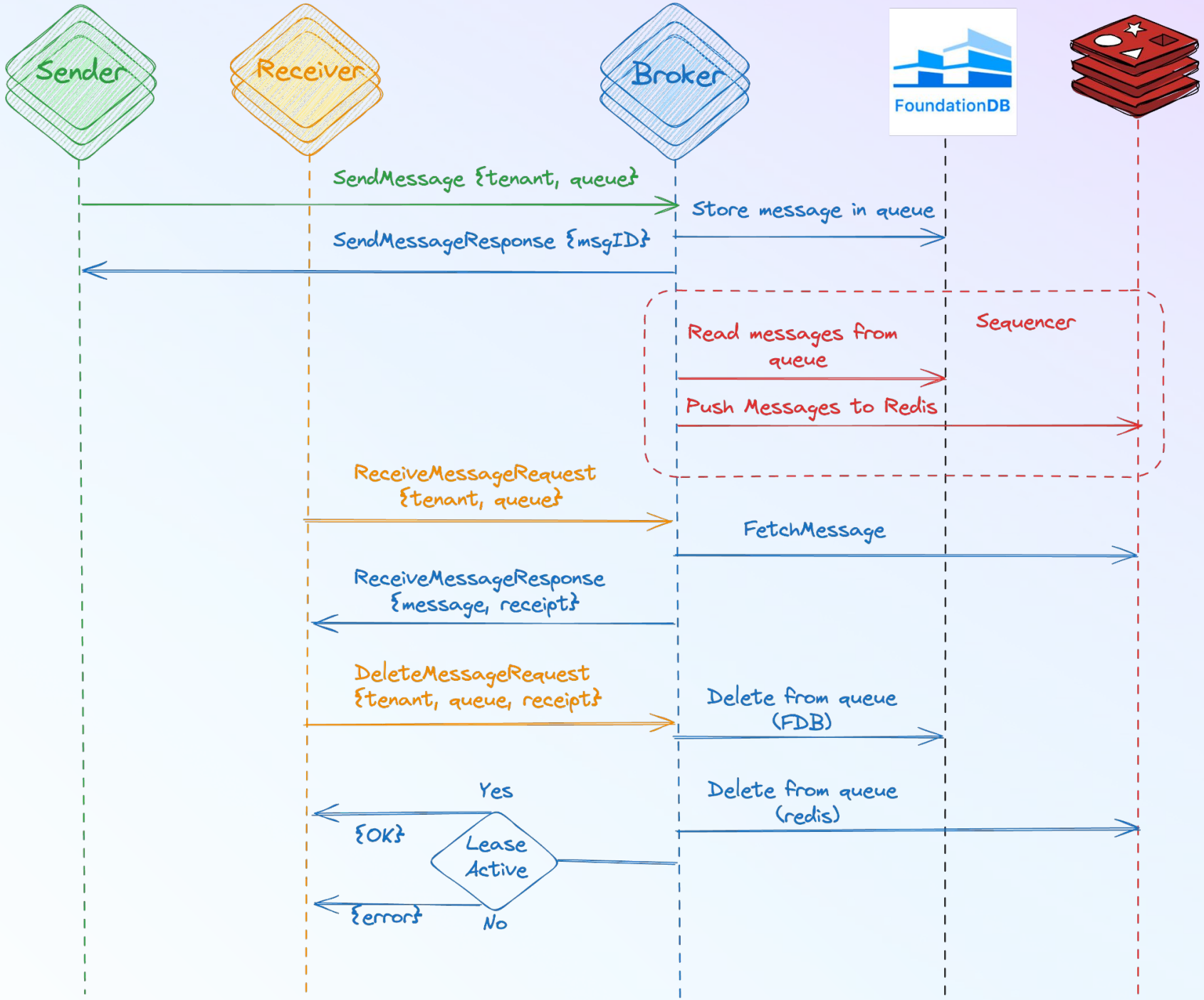
Design changes

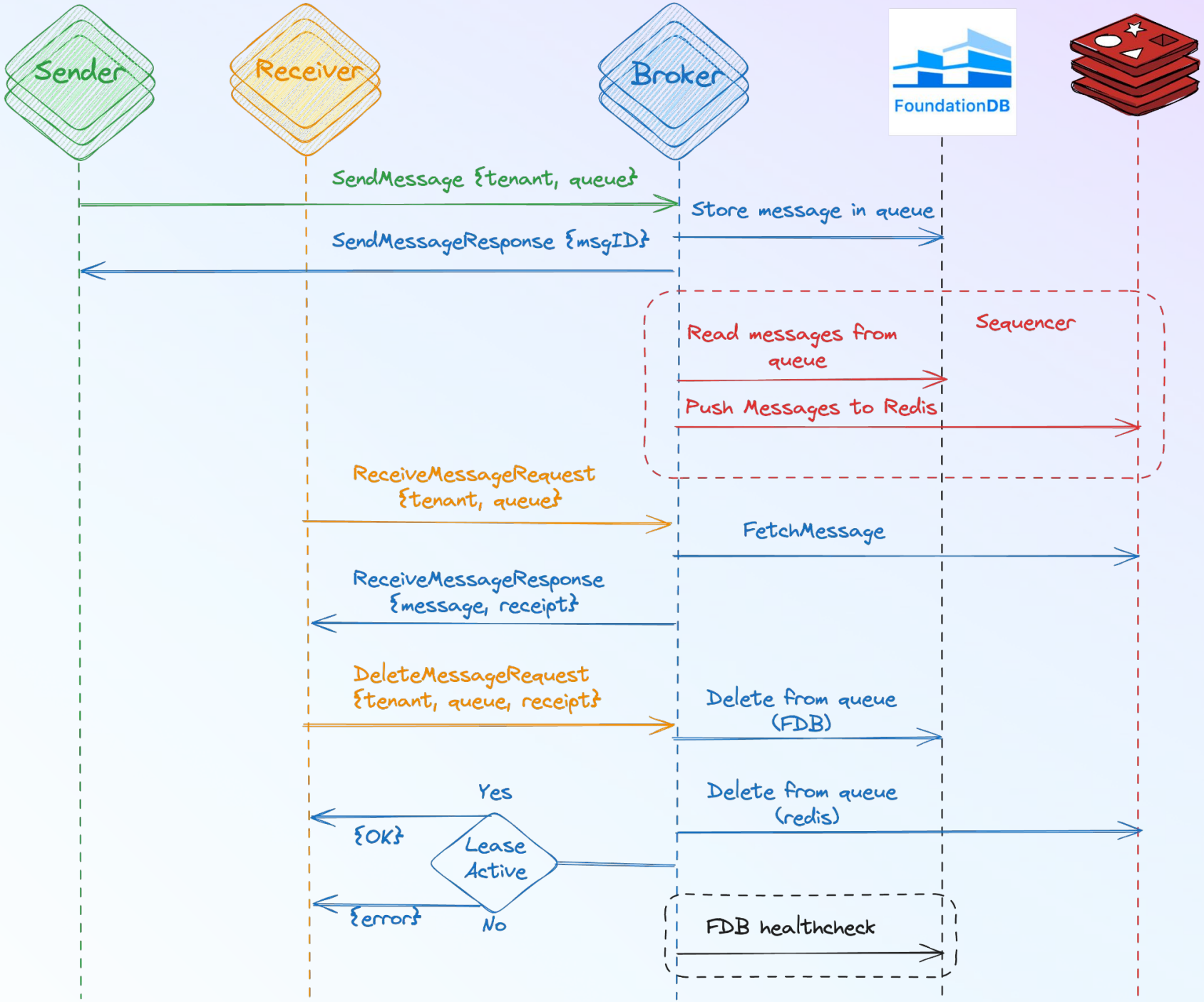



Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}


Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}
Sequencer
Read messages from queue
Push Messages to Redis


Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}
Sequencer
Read messages from queue
Push Messages to Redis
ReceiveMessageRequest {tenant, queue}
FetchMessage
ReceiveMessageResponse {message, receipt}

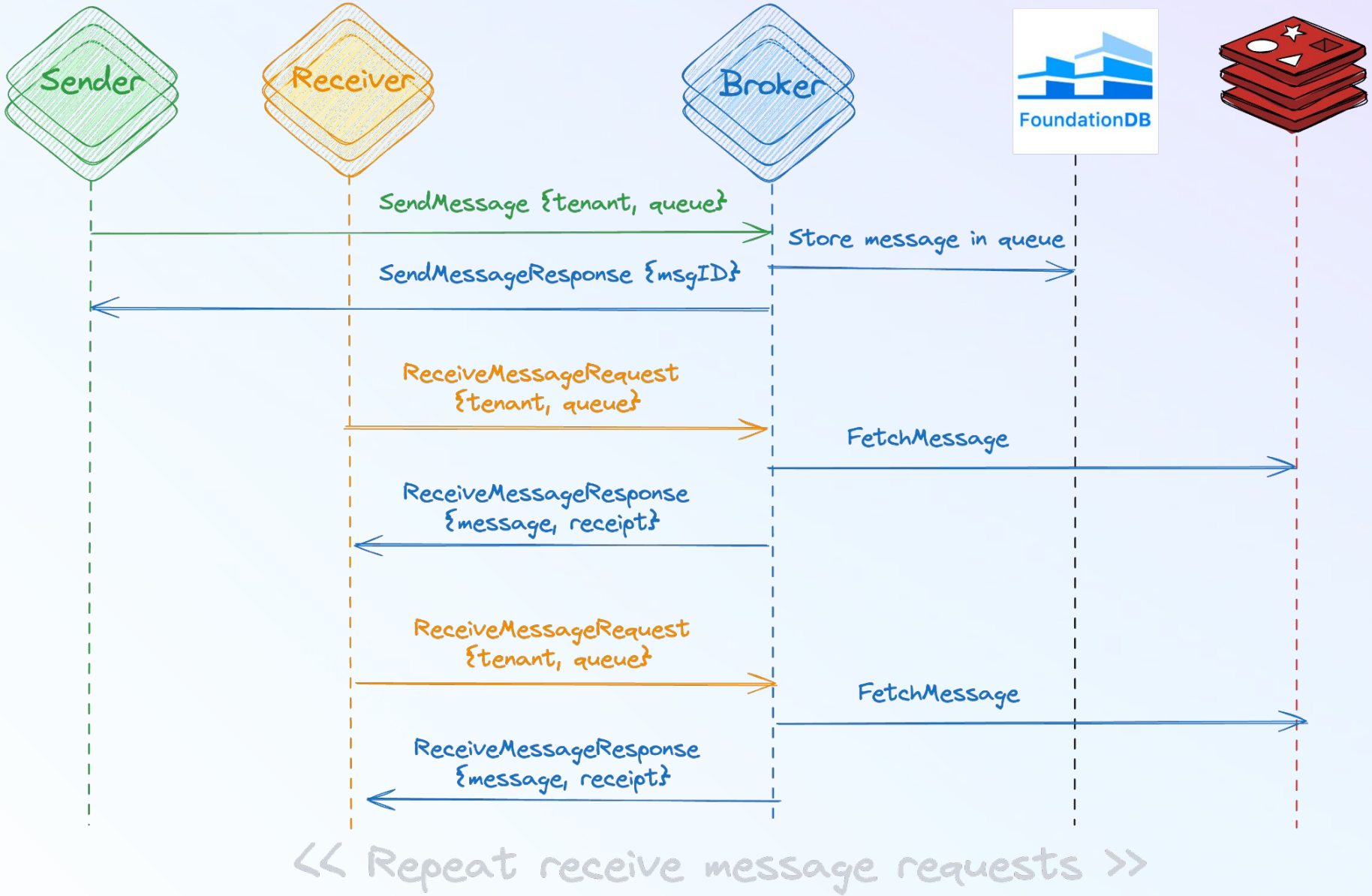

Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}
Sequencer
Read messages from queue
Push Messages to Redis
ReceiveMessageRequest {tenant, queue}
FetchMessage
ReceiveMessageResponse {message, receipt}
DeleteMessageRequest {tenant, queue, receipt}
Delete from queue (FDB)
Yes
Delete from queue (redis)
{OK}
Lease Active
{error}
No


Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}
Sequencer
Read messages from queue
Push Messages to Redis
ReceiveMessageRequest {tenant, queue}
FetchMessage
ReceiveMessageResponse {message, receipt}
DeleteMessageRequest {tenant, queue, receipt}
Delete from queue (FDB)
Yes
Delete from queue (redis)
{OK}
Lease Active
{error}
No
FDB healthcheck

Model Fails!

```
NoLostMsgs == <>[](Cardinality(Sender!Messages)
* Cardinality(Sender!ProcSet) = stats.deleted +
stats.deadLetterQueue)
```

Model Fails!


Sender
Receiver
Broker
FoundationDB
SendMessage {tenant, queue}
Store message in queue
SendMessageResponse {msgID}
ReceiveMessageRequest
{tenant, queue}
FetchMessage
ReceiveMessageResponse
{message, receipt}
ReceiveMessageRequest
{tenant, queue}
FetchMessage
ReceiveMessageResponse
{message, receipt}
<< Repeat receive message requests >>

Model Fails!

```
172     ReceiveMsg(self) == /\ pcReceiver[self] = "ReceiveMsg"
173                         /\ IF (Cardinality(senderMsgs) = 0 /\ QueueEmpty(clusters, receiverRequests[self].tenant, receiverRequests[self].queue) /\ Len(msgs[self]) = 0)
174                             \/ receiverRequests[self].attempts = 3 THEN
175                             /\ pcReceiver' = [pcReceiver EXCEPT ![self] = "Done"]
```

Design changes

- Intuitively, adding Redis on the Receive Messages path is an availability risk
- For our use-case this was something we could tolerate
- How do we ensure we did not introduce another failure mode?

Combining techniques

- Particularly valuable to combine modeling and simulations
- Modeling helped us verifying correctness of our system
- Simulations gave us estimates on how system behaves under load and failures
- Gave us confidence when we had design changes
- Enabled us to go from idea to production in 11 months

Deterministic simulators

- Met with Antithesis in 2022
- Incredibly powerful deterministic simulation platform
- At that time we were looking for something more “low level” and hosted on Datadog infrastructure

Joran Dirk Greef, Tigerbeetle

…Or to borrow from the world of auditing, our own inhouse DST serves as “internal audit function” with @AntithesisHQ as “external audit function”.

We simulate “from the inside of the binary out” (extremely protocol aware, e.g. checking page cache coherency with simulated disk).

Antithesis simulate “from the outside of the binary in” (the final compiled binary… so we’re testing Zig and LLVM here!).

https://twitter.com/jorandirkgreef/status/1765963724559429661

Deterministic simulators

- Go introduces non-determinism in many places
  - Goroutine scheduling, maps, selects etc.
  - Presented insights on testing distributed systems to Go language contributors
    - They face similar issues testing go schedulers itself
- Tried using Hermit (Meta)
  - Didn’t work with CGO; FDB client
  - Limitations on supported OS
  - May work for simple Go apps
- Can we make Golang itself deterministic?
  - Remains an area of exploration

Questions?

Sesh Nalla
sesh.nalla@datadoghq.com
linkedin.com/in/seshendranalla

Arun Parthiban
arun.parthiban@datadoghq.com
linkedin.com/in/arunparthiban

Thank you!