Fault Injection on Prime

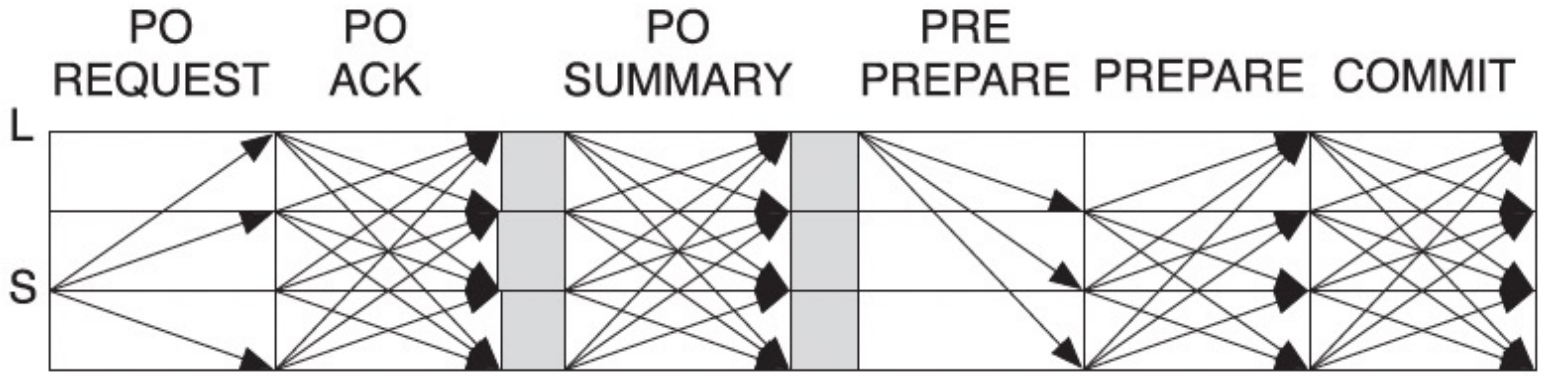
Presented by: Aren Alyahya Supervised by Dr. Amy Babay

BFT System

- Replication system that guarantees a correct result even with the existence of some system comprises
- A small inconsistency in the implementation can lead to serious problems which conflicts with the purpose of these systems

Prime

- Prime is a BFT protocol that grantees performance requirement ‘Bounded-delay’.



Inspiration

- ByzzFuzz: It is a tool that is randomly injects faults in a BFT protocol to detect violation in the system.
- Injection types:
  - Small-scope message mutations: change part of a protocol message.
  - Dropping massages
- Bugs types:
  - Correctness Violation
  - Liveness violation

Goal

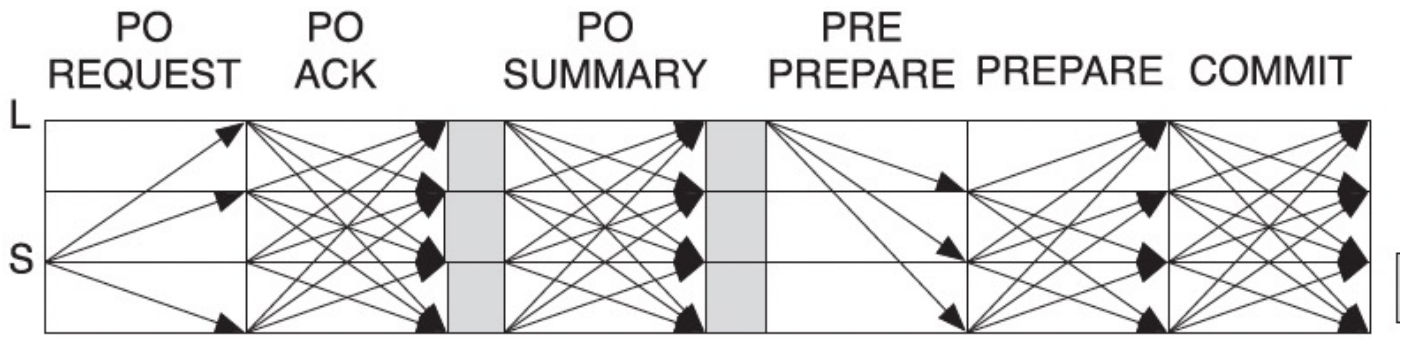
- Find safety or liveness violations
- Understand the causes of these violations
- Compare the effectiveness of several injection methods.

Fault Injection Types

- Small mutation injection:
  - Sequence numbers decrement.
  - Random sequence numbers.
  - Random sequence numbers within a range

Injection Placements

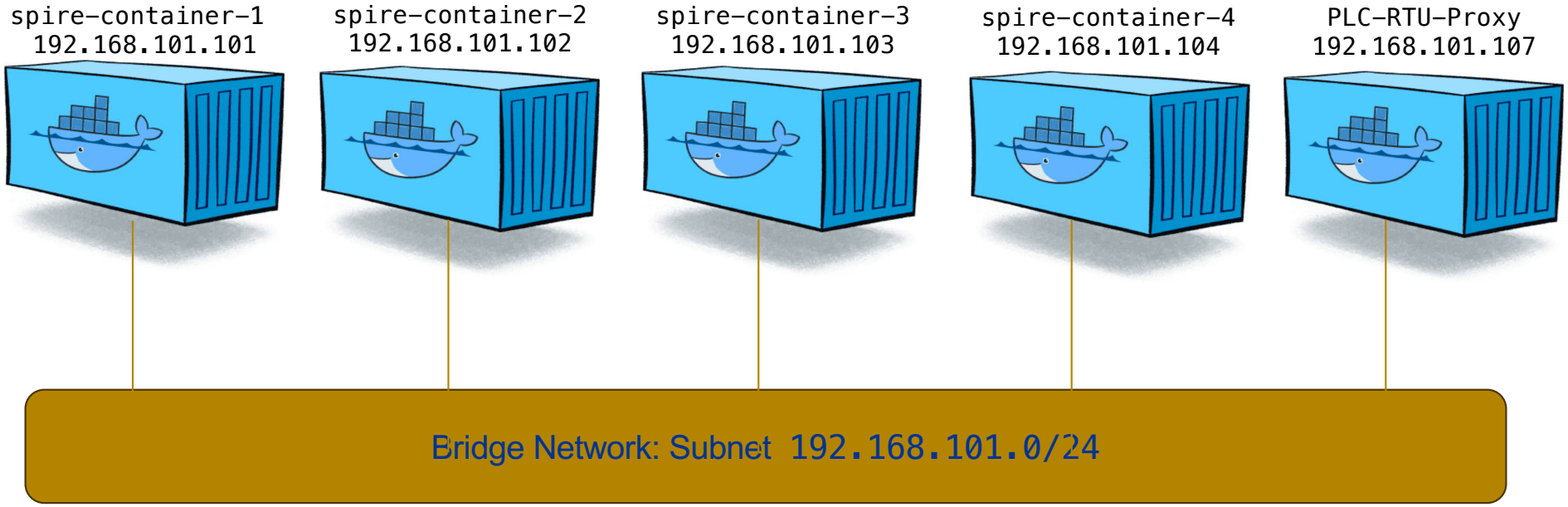
- Test 1: PRE_ORDER_Construct_PO_Ack
- Test 2: PRE_ORDER_Construct_PO_Request
- Test 3: PRE_ORDER_Construct_Update
- Test 4: ORDER_Construct_Pre_Prepare
- Test 5: ORDER_Construct_Prepare
- Test 6: VIEW_Construct_Replay
- Test 7: VIEW_Construct_Replay_Prepare



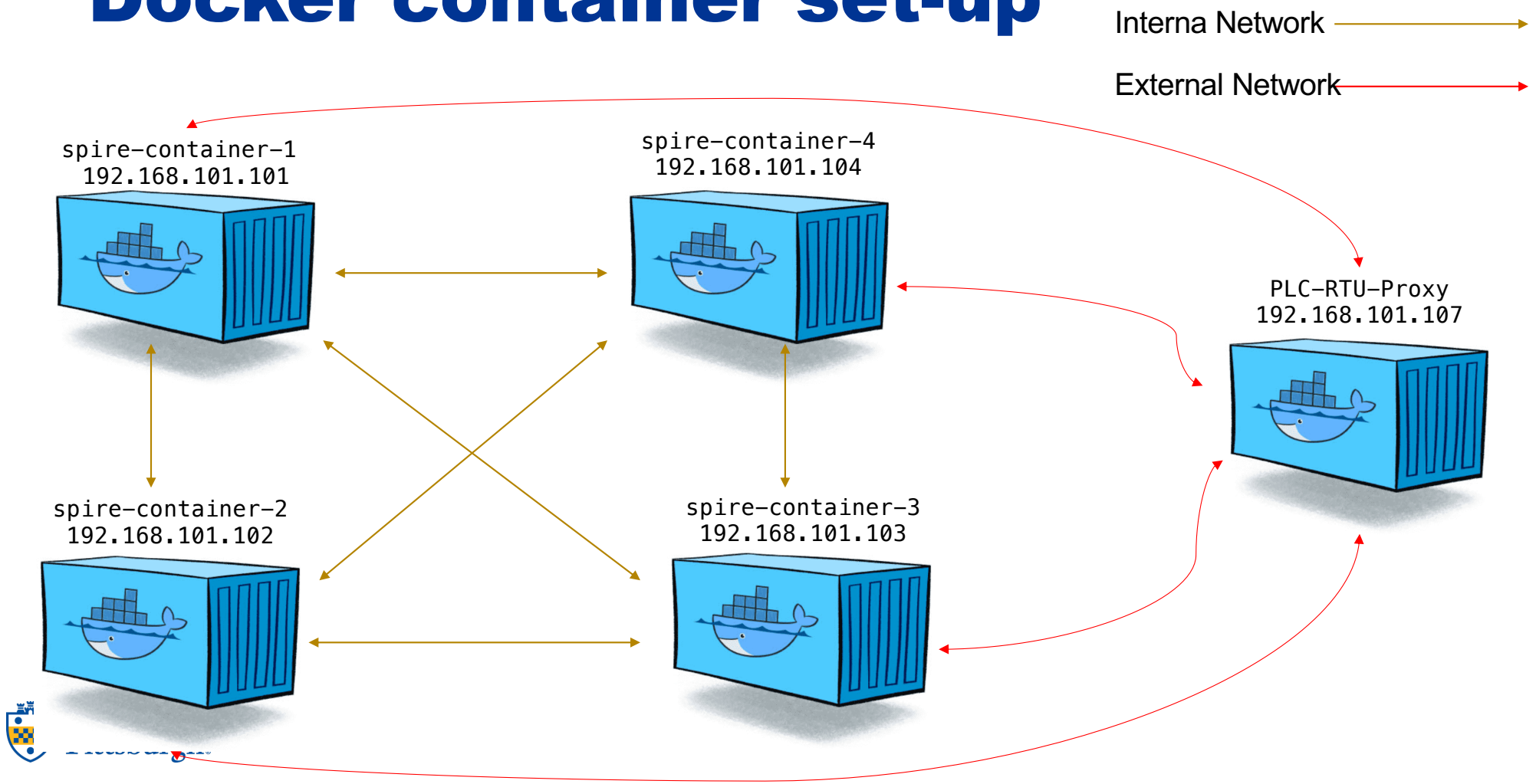
Working Environment

University of Pittsburgh

Docker container set-up



Docker container set-up

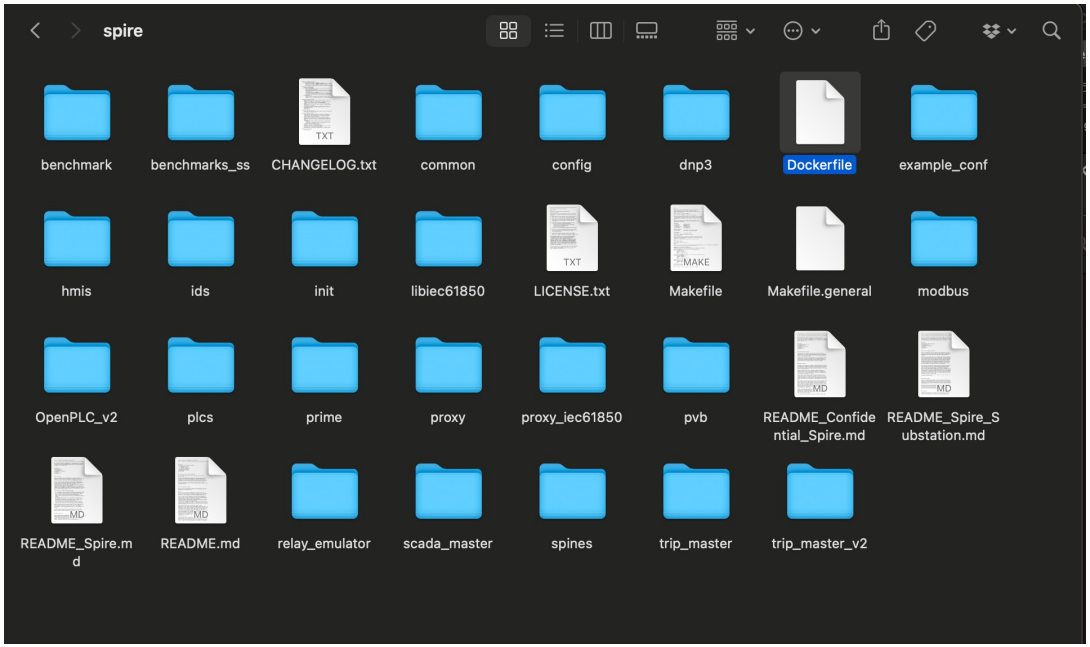


DockerFile

```
FROM almalinux/9-base
RUN yum clean all && yum update -y
RUN yum install openssl-devel -y
RUN yum install flex byacc -y
RUN yum install -y epel-release
RUN yum install -y qt5-qtbase-devel
RUN yum install -y qt5-qtwebkit-devel
RUN yum install -y qt5-qtsvg
RUN yum install -y qt5-qtsvg-devel
RUN yum install -y qt5-qttools-devel
RUN yum install -y qt5
RUN yum install -y qt5-qtmultimedia-devel
#RUN yum install -y qt5-*
RUN dnf --enablerepo=crb install -y qt5-devel
RUN yum install -y qt5* --skip-broken
RUN yum install cmake -y
RUN yum install -y autoconf
RUN yum install -y automake
RUN yum install -y bison

COPY . /app
RUN cd example_conf; ./install_conf.sh conf_4

RUN printf 'Y\n1\n' | make libs
RUN make
RUN cd spines/daemon; bash gen_keys.sh
RUN cd prime/bin; ./gen_keys; ./gen_tpm_keys.sh
RUN cd scada_master; ./gen_keys
```



University of Pittsburgh

Docker Commands

- Create the Image

```
docker build -t arenalyahya/spire:1 .
```

- Create the bridge network

```
docker network create --subnet 192.168.101.0/24 my-net
```

- Create 4 containers

```
docker run -itd --cap-add=NET_ADMIN --network=my-net --ip= 192.168.101.101 --name my-container1  arenalyahya/spire:1
docker run -itd --cap-add=NET_ADMIN --network=my-net --ip= 192.168.101.102 --name my-container2  arenalyahya/spire:1
docker run -itd --cap-add=NET_ADMIN --network=my-net --ip= 192.168.101.103 --name my-container3  arenalyahya/spire:1
docker run -itd --cap-add=NET_ADMIN --network=my-net --ip= 192.168.101.104 --name my-container4  arenalyahya/spire:1
```

Followed Approach

- An injection is done by one machine (ex. Replica 1).
- Run the Prime setup.
- Run the benchmark:
  - A total of 50 updates sent by the client every 1 second

```
cd benchmark; ./benchmark 1 192.168.101.107:8120  1000000 50
```

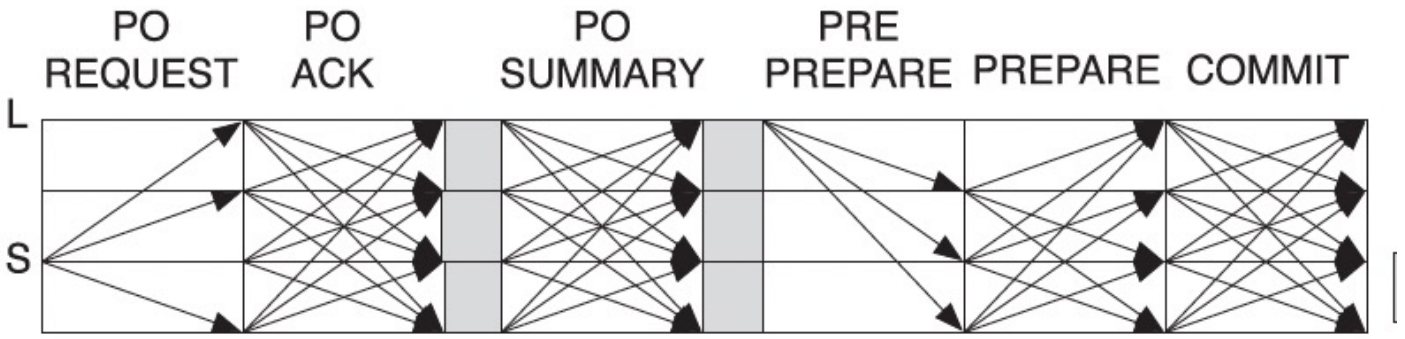
- Save the updates from all machines in files
- Process the files

Fault Injection Types

- Small mutation injection:
  - Decrement sequence numbers.
  - Random sequence numbers.
  - Random sequence numbers within a range

Injection Placements

- Test 1: PRE_ORDER_Construct_PO_Ack
- Test 2: PRE_ORDER_Construct_PO_Request
- Test 3: PRE_ORDER_Construct_Update
- Test 4: ORDER_Construct_Pre_Prepare
- Test 5: ORDER_Construct_Prepare
- Test 6: VIEW_Construct_Replay
- Test 7: VIEW_Construct_Replay_Prepare



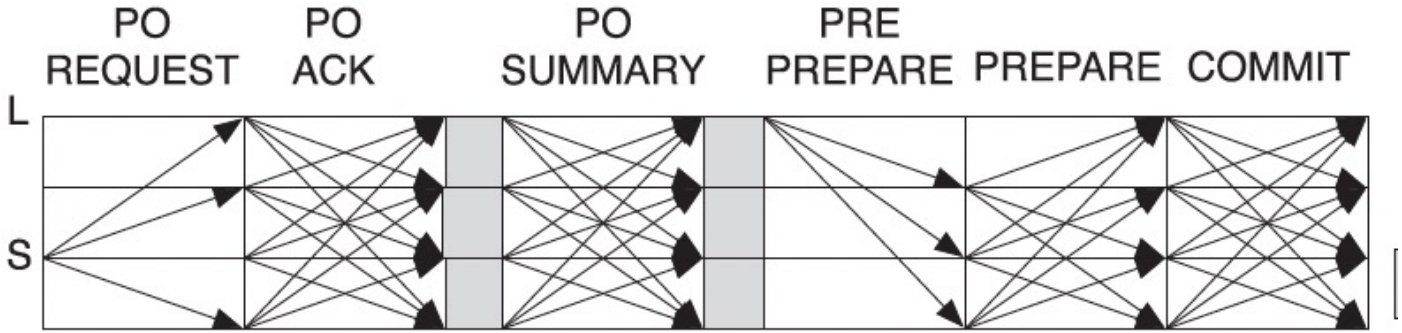
Processing the files

- Compare between two files in terms of matching:
  - Server Id
  - Sequence number
  - The content of the message

M1	M2
12/09/24 21:29:51.630289 Executing an update: server_id: 2 (incarnation: 1733779791 seq_num: 1) contents: 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000200000000000000066000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000 12/09/24 21:29:51.669061 Executing an update: server_id: 1 (incarnation: 1733779725 seq_num: 0) contents: 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000100000000000000065000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000	12/09/24 21:29:51.618670 Executing an update: server_id: 2 (incarnation: 1733779791 seq_num: 1) contents: 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000200000000000000066000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000 12/09/24 21:29:51.662643 Executing an update: server_id: 2 (incarnation: 1733779791 seq_num: 0) contents: 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000200000000000000065000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000000

University of Pittsburgh

Test 1:
PRE_ORDER_Construct_PO_Ack



University of Pittsburgh

Test 1:
PRE_ORDER_Construct_PO_Ack

`if (ack_part->seq.seq_num % 10 == 0) {`		

`ack_part->seq.seq_num--;`
`}`
	`srand(time(NULL));`
	`ack_part->seq.seq_num = rand()%51;`
	`ack_part->seq.incarnation = rand();`
	`}`
		`srand(time(NULL));`
		`ack_part->seq.seq_num = rand()%51;`
		`ack_part->seq.incarnation = rand();`
		`}`

Sequence # Decrement	Random Sequence #	Sequence # within a range

University of Pittsburgh

Test 1:
PRE_ORDER_Construct_PO_Ack

`if (ack_part->seq.seq_num % 10 == 0) {`		

`ack_part->seq.seq_num--;`
`}`
	`srand(time(NULL));`
	`ack_part->seq.seq_num = rand()%51;`
	`ack_part->seq.incarnation = rand();`
	`}`
		`srand(time(NULL));`
		`ack_part->seq.seq_num = rand()%51;`
		`ack_part->seq.incarnation = rand();`
		`}`

Sequence # Decrement	Random Sequence #	Sequence # within a range
Total updates: 121	Total updates: 53	Total updates: 50
20 differences	12 differences	11 differences
Process continued	Machine 1 and 2 stopped, machine 3 and 4 hung	

University of Pittsburgh

Response to Sequence # Decrement

The name of the injected function	M1	M2	M3	M4
Test 1: PRE_ORDER_Construct_PO_Ack	No	No	No	No
Test 2: PRE_ORDER_Construct_PO_Request	No	No	No	No
Test 3: PRE_ORDER_Construct_Update	No	No	No	No
Test 4: ORDER_Construct_Pre_Prepare	No	No	No	No
Test 5: ORDER_Construct_Prepare	No	No	No	No
Test 6: VIEW_Construct_Replay	No	No	No	No
Test 7: VIEW_Construct_Replay_Prepare	No	No	No	No

Response to Sequence # random change

The name of the injected function	M1	M2	M3	M4
Test 1: PRE_ORDER_Construct_PO_Ack	Stop	Stop	hung	hung
Test 2: PRE_ORDER_Construct_PO_Request	No	No	No	No
Test 3: PRE_ORDER_Construct_Update	No	No	No	No
Test 4: ORDER_Construct_Pre_Prepare	Stop	Stop	hung	hung
Test 5: ORDER_Construct_Prepare	No	No	No	No
Test 6: VIEW_Construct_Replay	No	No	No	No
Test 7: VIEW_Construct_Replay_Prepare	No	No	No	No

Response to Sequence # random change within a range

The name of the injected function	M1	M2	M3	M4
Test 1: PRE_ORDER_Construct_PO_Ack	Stop	Stop	hung	hung
Test 2: PRE_ORDER_Construct_PO_Request	No	No	No	No
Test 3: PRE_ORDER_Construct_Update	No	No	No	No
Test 4: ORDER_Construct_Pre_Prepare	Stop	hung	hung	hung
Test 5: ORDER_Construct_Prepare	No	No	No	No
Test 6: VIEW_Construct_Replay	No	No	No	No
Test 7: VIEW_Construct_Replay_Prepare	No	No	No	No

Challenges faced

- Sequence number:
  - The sequence number consists two number (incarnation + seq). Does that I need to inject a mutation in both or one.
  - Identifying the client sequence number from Prime sequence number .
- View change:
  - Injecting views Is not trivial
  - Whan the view changes

Thank You!!

University of Pittsburgh