Reproducing Slow-leader and Slow-follower: One-Size-Fits-None (NSDI’25)

Slow Follower & Slow Leader — Simple Step-by-Step

1) Requirements & Install (once)

```
# Docker + compose plugin (Ubuntu)
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg lsb-release
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" \
  | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin

# Optional helpers
sudo apt-get install -y tmux jq coreutils
```

2) Bring up the cluster

```
# From folder with your docker-compose for Cassandra/etcd
docker compose up -d

# Check containers are running
docker ps
```

3) Slow Follower Experiment

In this fault, one follower node is made slow via tc netem (delay or loss) while the leader is healthy.
We use run_netdelay.sh (mid-run injection) or run_netfault.sh (full-run injection).

(a) Baseline

```
./run_netfault.sh loss "" "" 60 baseline
```

No fault; records baseline throughput.

(b) Delay — Full Run

bash

CopyEdit

```
./run_netfault.sh delay "cassandra_b" "100ms" 60 delay100_full
```

(c) Delay — Mid Run

bash
CopyEdit

```
./run_netdelay.sh "cassandra_b" "100ms" 60 delay100_mid 30
```

(d) Loss — Full Run

bash
CopyEdit

```
./run_netfault.sh loss "cassandra_b" "20%" 60 loss20_full
```

(e) Loss — Mid Run

bash
CopyEdit

```
./run_netdelay.sh "cassandra_b" "20%" 60 loss20_mid 30
```

4) Slow Leader Experiment

Here, the leader node is slowed to test how leadership bottlenecks affect throughput.
The target leader container must be identified first.

(a) Find current leader

bash
CopyEdit

```
docker exec etcd0 etcdctl endpoint status -w table \
--endpoints="http://etcd0:2379,http://etcd1:2379,http://etcd2:2379"
```

Look for Leader column; note the leader’s container name.

(b) Delay — Full Run

bash
CopyEdit

```
./run_netfault.sh delay "<leader_container>" "100ms" 60 leader_delay100_full
```

(c) Delay — Mid Run

bash
CopyEdit

```
./run_netdelay.sh "<leader_container>" "100ms" 60 leader_delay100_mid 30
```

(d) Loss — Full Run

bash
CopyEdit

```
./run_netfault.sh loss "<leader_container>" "20%" 60 leader_loss20_full
```

(e) Loss — Mid Run

bash
CopyEdit

```
./run_netdelay.sh "<leader_container>" "20%" 60 leader_loss20_mid 30
```

5) Sweep runs (optional)

You can automate multiple runs for a range of values.

Loss sweep example:

bash
CopyEdit

```
./sweep_loss.sh 60
```

This runs baseline + loss {1%,5%,…,70%} against cassandra_b & cassandra_c.

6) Outputs per run

- Cassandra stress log → outputs/<timestamp>_<label>/<label>_cassandra-stress.log
- metadata.txt → fault parameters + run timestamp.
- You can use tail_all.sh to quickly preview last 15 lines of all logs.

bash
CopyEdit

```
./tail_all.sh
```

7) Reset / clear injection

bash
CopyEdit

```
docker exec <container> tc qdisc del dev eth0 root 2>/dev/null || true
```

8) Copy results to local machine

powershell
CopyEdit

```
scp -i "C:\path\to\key.pem" -r ^
  cc@<HOST>:~/outputs ^
  C:\path\to\local\outputs
```

9) Collect your result folders

Decide where your outputs/<timestamp>_<label>/ directories live (on the remote or copied locally).
Examples:

- Linux: ~/cassandra-demo/outputs
- Windows (after scp): C:\Newer-fault\outputs

In the plotting code below, you will point BASE_DIR to this folder.

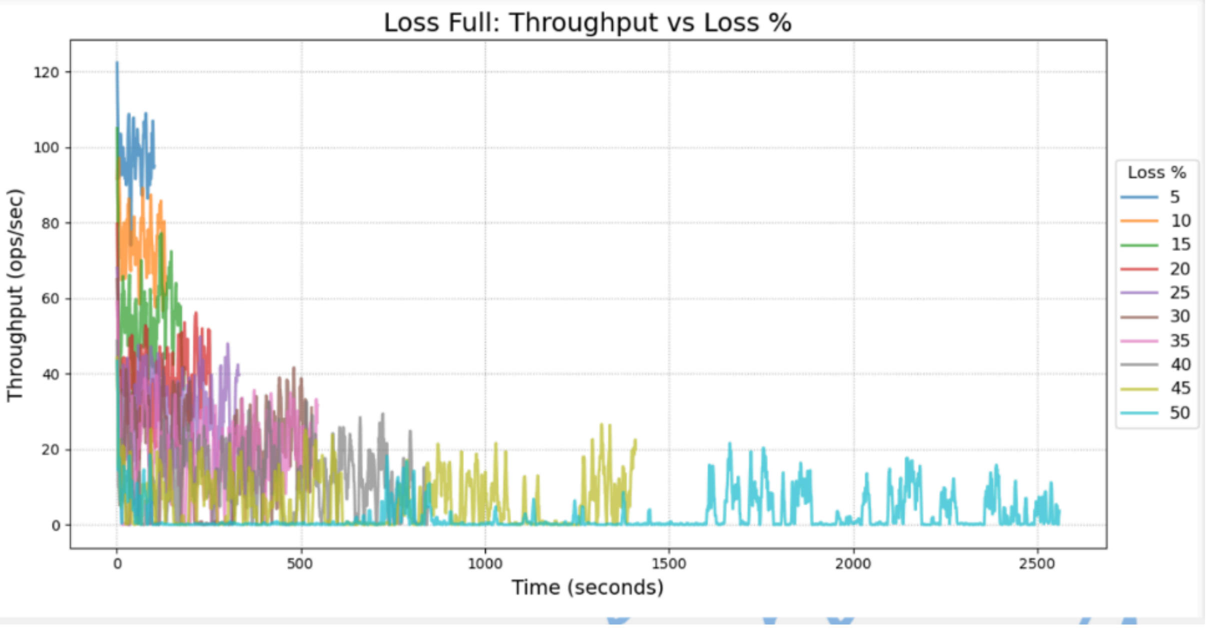
10) Parse & plot throughput from cassandra-stress logs

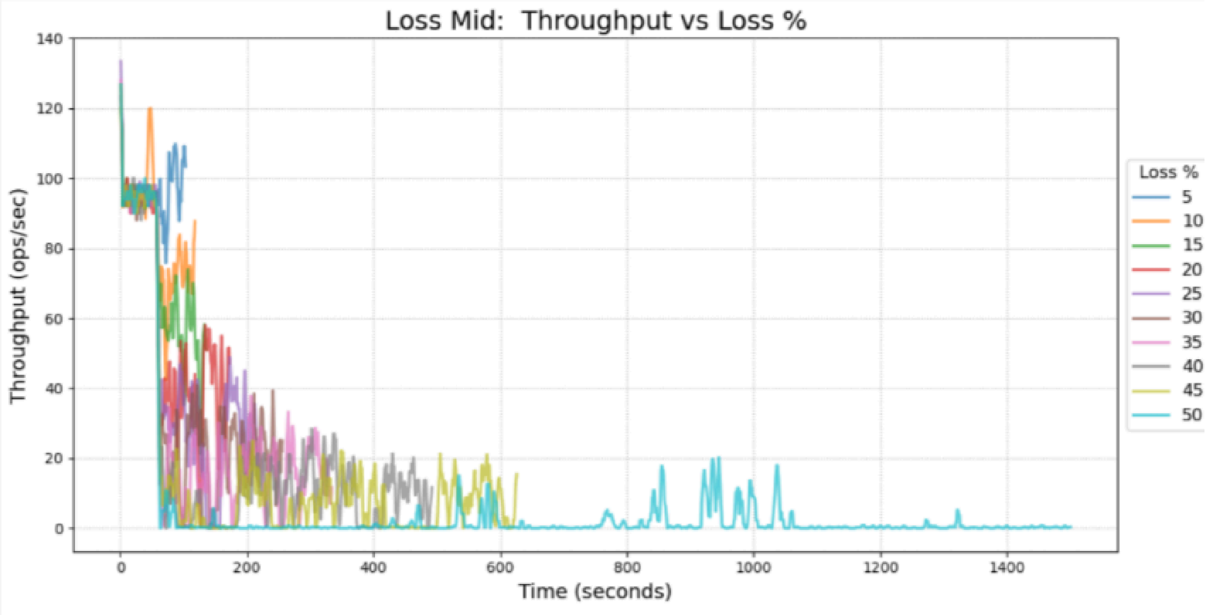
Use this Python script to extract ops/s over time from each run’s log and produce line charts. It assumes each run has:

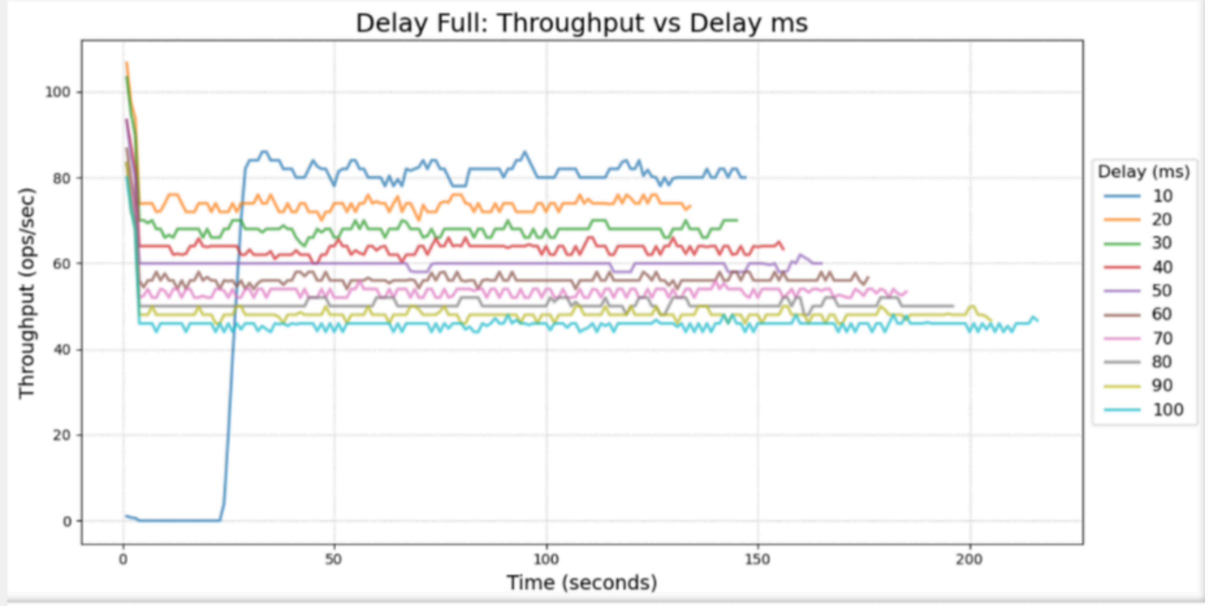
php-template
CopyEdit
outputs/<stamp>_<label>/<label>_cassandra-stress.log

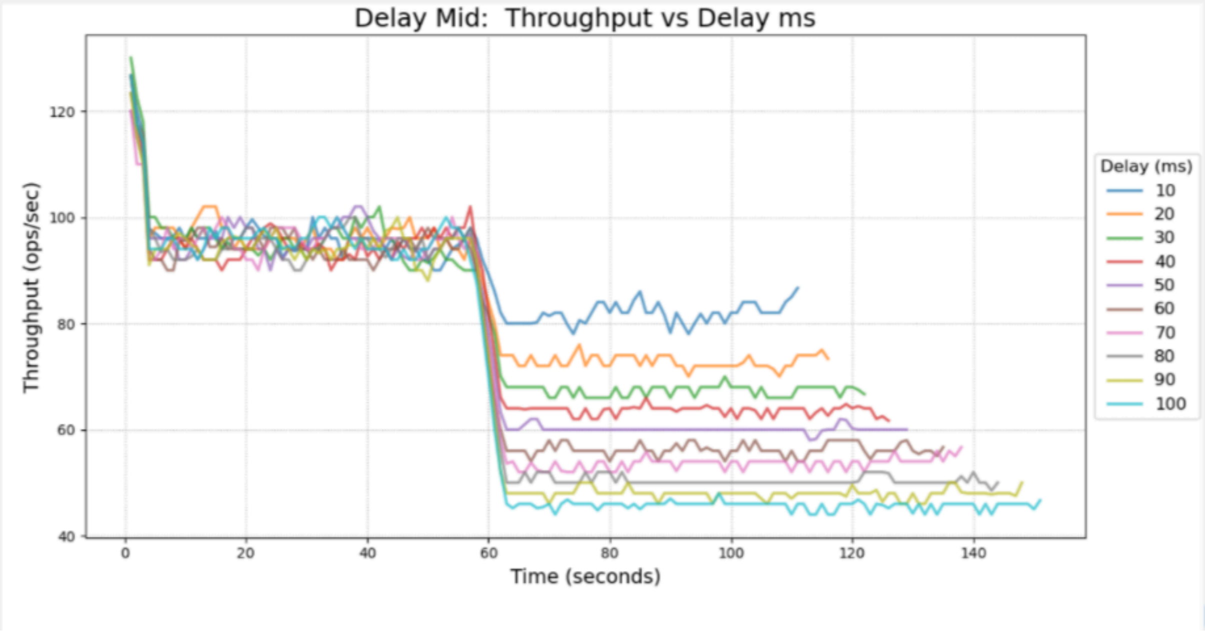
> The regex targets typical cassandra-stress periodic lines reporting throughput. If your image prints a slightly different format, adjust the regex noted in the comments.

```
# save as plot_throughput.py and run: python3 plot_throughput.py

# Requirements: Python 3.10+, pandas, matplotlib

import os, re, sys

import pandas as pd

import matplotlib.pyplot as plt

# 1) POINT THIS TO YOUR outputs/ DIRECTORY

#    Example Linux: BASE_DIR = "/home/cc/cassandra-demo/outputs"

#    Example Windows: BASE_DIR = r"C:\Newer-fault\outputs"

BASE_DIR = "/home/cc/cassandra-demo/outputs"

# 2) A helper to parse a single stress log and return DataFrame(second, ops)

#    Adjust the regex if your logs differ. We look for "...op/s" numbers.

LINE_RX = re.compile(r'(?i)\b([0-9]+(?:\.[0-9]+)?)\s*op/s\b')

def parse_stress_log(path):

    ops = []

    sec = 0

    with open(path, 'r', encoding='utf-8', errors='ignore') as f:

        for line in f:

            m = LINE_RX.search(line)

            if m:

                try:
                    val = float(m.group(1))
                except:
                    continue
                ops.append((sec, val))
                sec += 1
    if not ops:
        return pd.DataFrame(columns=["second","ops"])
    df = pd.DataFrame(ops, columns=["second","ops"])
    return df
# 3) Collect all runs
runs = []   # list of (label, df)
for d in sorted(os.listdir(BASE_DIR)):
    run_dir = os.path.join(BASE_DIR, d)
    if not os.path.isdir(run_dir):
        continue
    # label is the suffix after last underscore
    label = d.split("_", maxsplit=1)[-1] if "_" in d else d
    logfile = os.path.join(run_dir, f"{label}_cassandra-stress.log")
    if not os.path.isfile(logfile):
        continue
    df = parse_stress_log(logfile)
    if not df.empty:
        # light smoothing for readability (moving average 5)
        df["ops_smooth"] = df["ops"].rolling(window=5, min_periods=1, center=True).mean()
        runs.append((label, df))
if not runs:
    print("No parsable runs found under:", BASE_DIR)
    sys.exit(0)
# 4) Plot each run individually to PNG for slides
PLOTS_DIR = os.path.join(BASE_DIR, "_plots")
os.makedirs(PLOTS_DIR, exist_ok=True)
for label, df in runs:
    plt.figure(figsize=(10,4))
    plt.plot(df["second"], df["ops_smooth"], label=label)
    plt.title(f"Throughput vs Time — {label}")
    plt.xlabel("Time (s)")
    plt.ylabel("Ops/s")
    plt.grid(linestyle=":")
    plt.legend()
    out = os.path.join(PLOTS_DIR, f"{label}_throughput.png")
    plt.tight_layout()
    plt.savefig(out, dpi=150)
    plt.close()
    print("Saved:", out)
# 5) (Optional) Quick overlay helpers by family (Full/Mid, Loss/Delay)
def overlay(pattern, title, legend_title):
    import fnmatch
    subset = [(lbl, df) for lbl, df in runs if fnmatch.fnmatch(lbl, pattern)]
    if not subset:
        print("No matches for pattern:", pattern); return
    plt.figure(figsize=(12,5))
    for lbl, df in subset:
        plt.plot(df["second"], df["ops_smooth"], label=lbl)
    plt.title(title); plt.xlabel("Time (s)"); plt.ylabel("Ops/s")
    plt.grid(linestyle=":")
    plt.legend(title=legend_title, loc="center left", bbox_to_anchor=(1,0.5))
    out = os.path.join(PLOTS_DIR, f"{title.replace(' ','_')}.png")
    plt.tight_layout(); plt.savefig(out, dpi=150); plt.close()
    print("Saved:", out)
# Examples (tweak to your labels):
overlay("*loss*full", "Loss Full — Throughput Overlays", "Runs")
overlay("*loss*mid",  "Loss Mid — Throughput Overlays",  "Runs")
overlay("*delay*full","Delay Full — Throughput Overlays","Runs")
overlay("*delay*mid", "Delay Mid — Throughput Overlays", "Runs")
print("\nAll plots stored under:", PLOTS_DIR)
```



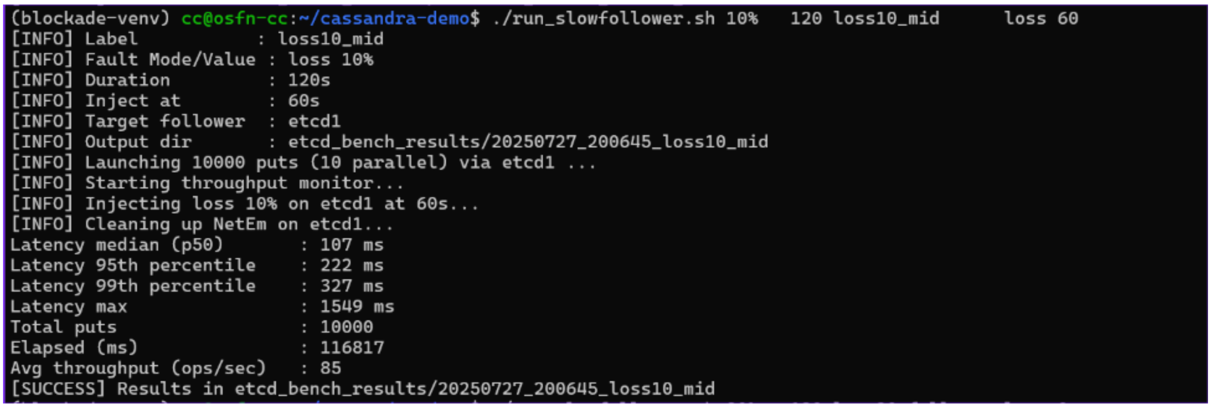

Loss Mid: Throughput vs Loss %
Throughput (ops/sec)
Time (seconds)
0
20
40
60
80
100
120
140
0
200
400
600
800
1000
1200
1400
Loss %
5
10
15
20
25
30
35
40
45
50


Delay Full: Throughput vs Delay ms
Throughput (ops/sec)
Time (seconds)
0
20
40
60
80
100
0
50
100
150
200
Delay (ms)
10
20
30
40
50
60
70
80
90
100



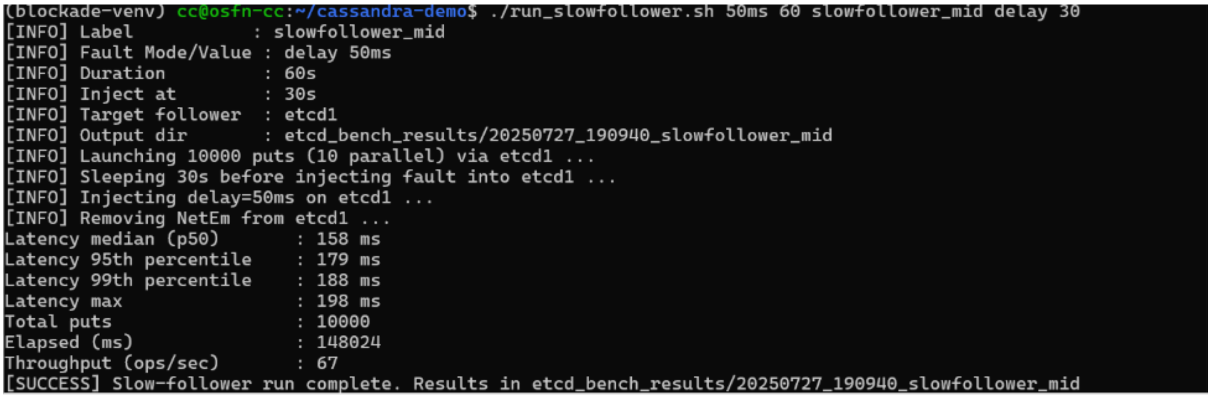
10) Run Example

./run_slowfollower.sh 10% 120 loss10_mid loss 60



```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ ./run_slowfollower.sh 10%   120 loss10_mid      loss 60
[INFO] Label           : loss10_mid
[INFO] Fault Mode/Value : loss 10%
[INFO] Duration         : 120s
[INFO] Inject at        : 60s
[INFO] Target follower  : etcd1
[INFO] Output dir       : etcd_bench_results/20250727_200645_loss10_mid
[INFO] Launching 10000 puts (10 parallel) via etcd1 ...
[INFO] Starting throughput monitor...
[INFO] Injecting loss 10% on etcd1 at 60s...
[INFO] Cleaning up NetEm on etcd1...
Latency median (p50)        : 107 ms
Latency 95th percentile     : 222 ms
Latency 99th percentile     : 327 ms
Latency max                 : 1549 ms
Total puts                  : 10000
Elapsed (ms)                : 116817
Avg throughput (ops/sec)    : 85
[SUCCESS] Results in etcd_bench_results/20250727_200645_loss10_mid
```

./run_slowfollower.sh 50ms 60 slowfollower_mid delay 30



```
(blockade-venv) cc@osfn-cc:~/cassandra-demo$ ./run_slowfollower.sh 50ms 60 slowfollower_mid delay 30
[INFO] Label           : slowfollower_mid
[INFO] Fault Mode/Value : delay 50ms
[INFO] Duration         : 60s
[INFO] Inject at        : 30s
[INFO] Target follower  : etcd1
[INFO] Output dir       : etcd_bench_results/20250727_190940_slowfollower_mid
[INFO] Launching 10000 puts (10 parallel) via etcd1 ...
[INFO] Sleeping 30s before injecting fault into etcd1 ...
[INFO] Injecting delay=50ms on etcd1 ...
[INFO] Removing NetEm from etcd1 ...
Latency median (p50)       : 158 ms
Latency 95th percentile    : 179 ms
Latency 99th percentile    : 188 ms
Latency max                : 198 ms
Total puts                 : 10000
Elapsed (ms)               : 148024
Throughput (ops/sec)       : 67
[SUCCESS] Slow-follower run complete. Results in etcd_bench_results/20250727_190940_slowfollower_mid
```