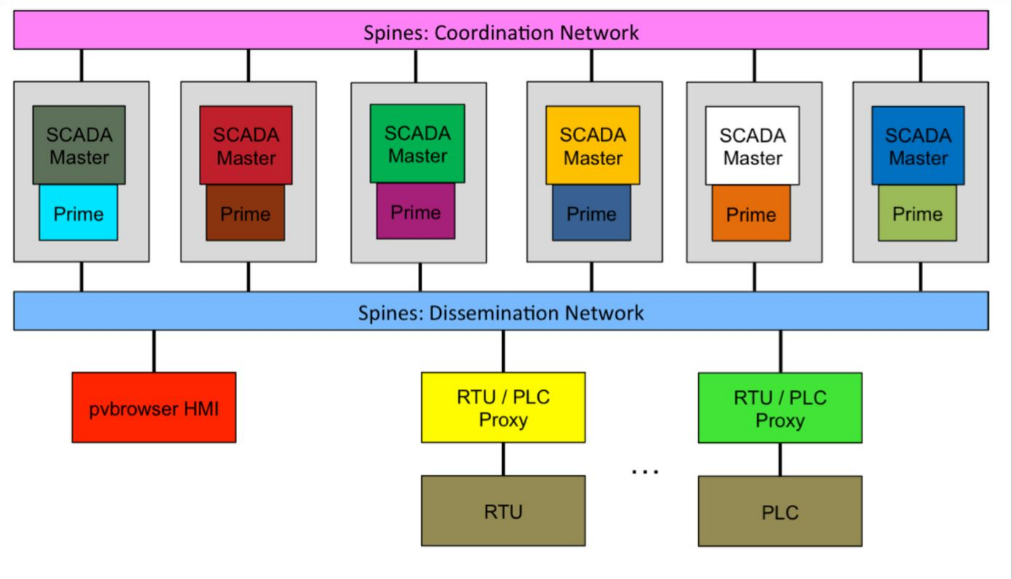
Towards a Resilient US Power Grid

Valentina Alzate, Ben Cillie, Caroline Reynolds, Megan Rosen, Daniel Weber

The goal of our project is to find errors in Spire's protocol that can be exploited by an attacker to cause a fatal slowdown or a total system failure.

The Spire System

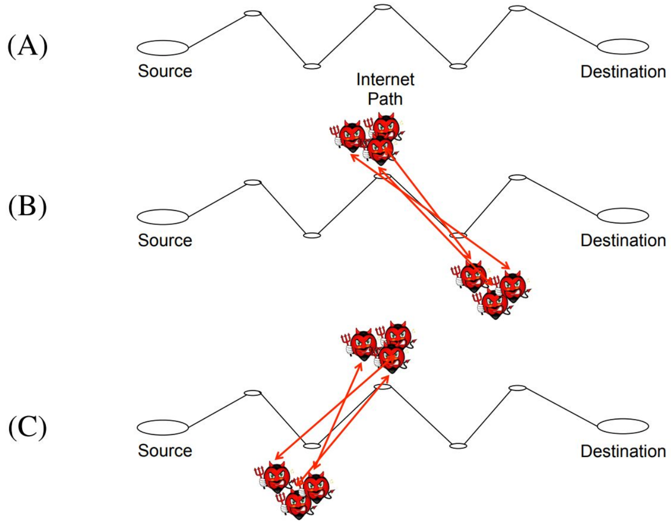
The goal of Spire is to create an intrusion-tolerant, reliable system to operate the power grid that is exposed to the open internet.



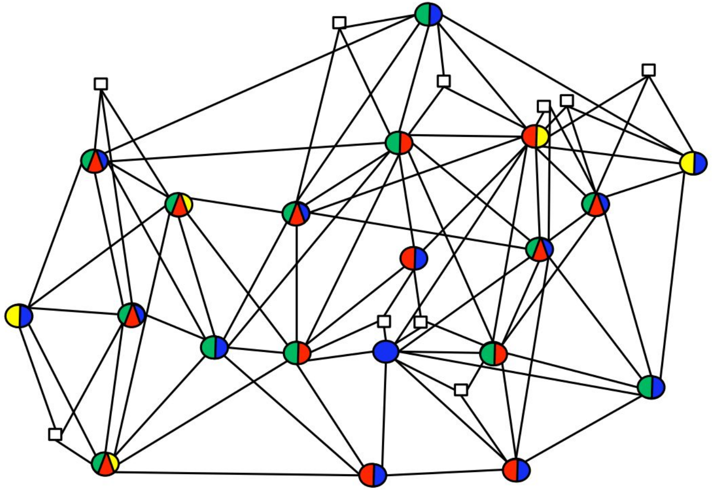
Spines

An Intrusion Tolerant Network

Conventional Infrastructure



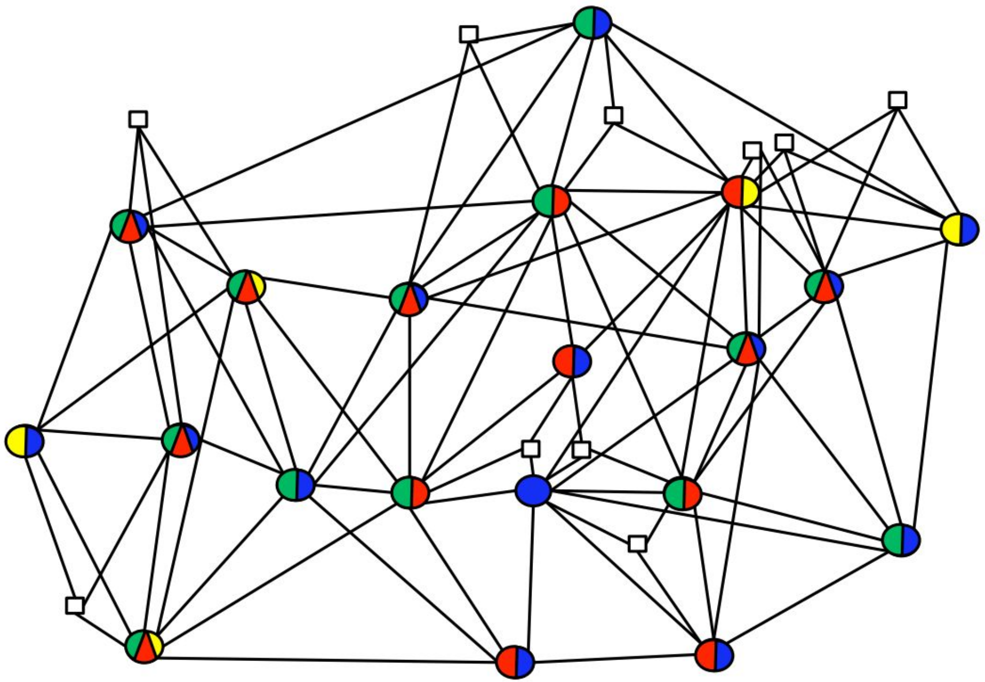
- Overlay network built on top of existing IP infrastructure
  - Multi-homing



Spines

An Intrusion Tolerant Network

- Intrusion Tolerance
  - Fairness Principle
  - Flooding



Prime

How to Create a Reliable System?

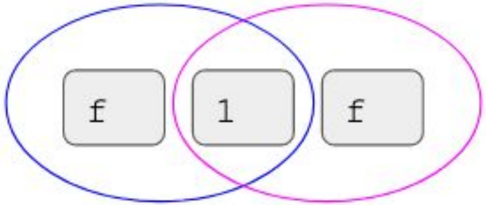
- Problems to Solve:
    - What happens if our server goes down?
    - What happens if our server is compromised by an attacker?

The Answer: REDUNDANCY

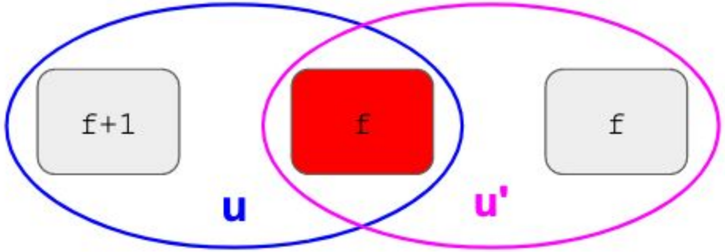
Prime

How many replicas do we need?

- Fail Stop Failure
    - A replica becomes completely unresponsive
- Handling Fail Stop Failure: $N \geq 2f + 1$



- Byzantine Failure
    - A replica responds in any unexpected way
    - Harder to account for in a system
- Handling Byzantine Failure: $N \geq 3f + 1$



Prime

Consensus Algorithms

- We seek 3 things:
  - 1) Termination
  - 2) Integrity
  - 3) Agreement
- Prime guarantees that we achieve these properties in a timely manner.
  - Older protocols did not enforce a timeliness condition

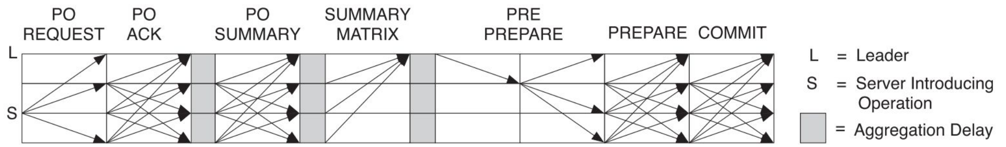


Fig. 3. Operation of Prime with a malicious leader that performs well enough to avoid being replaced ($f = 1$).

Prime: Deep Dive

3 Things:

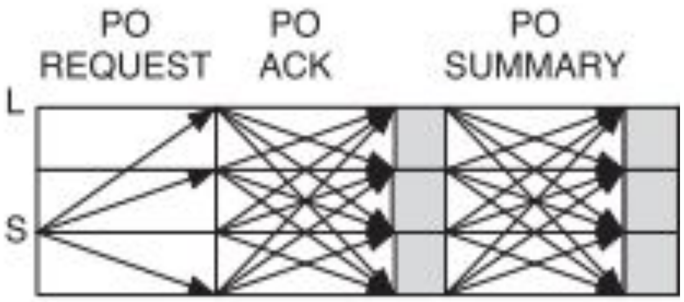
Integrity

Agreement

Termination

Prime Protocol: Pre-Ordering

- Pre-Order Requests: Servers send their client updates to all other servers with a unique sequence number.
- Acknowledgement: Servers acknowledge that they have received a pre-order requests.
- Summary: Servers send summaries of their believed current state of the system.



Prime Protocol: Suspect Leader

- Timeliness of Agreement
- Leader leads the ordering process
- Slow leader = slow execution
- Turnaround Time
  - RTT PING
- A leader is replaced if it is significantly slower than the average replica.

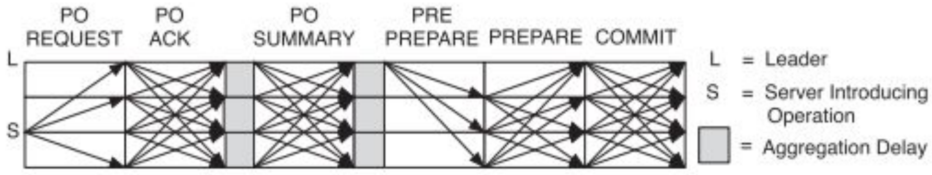
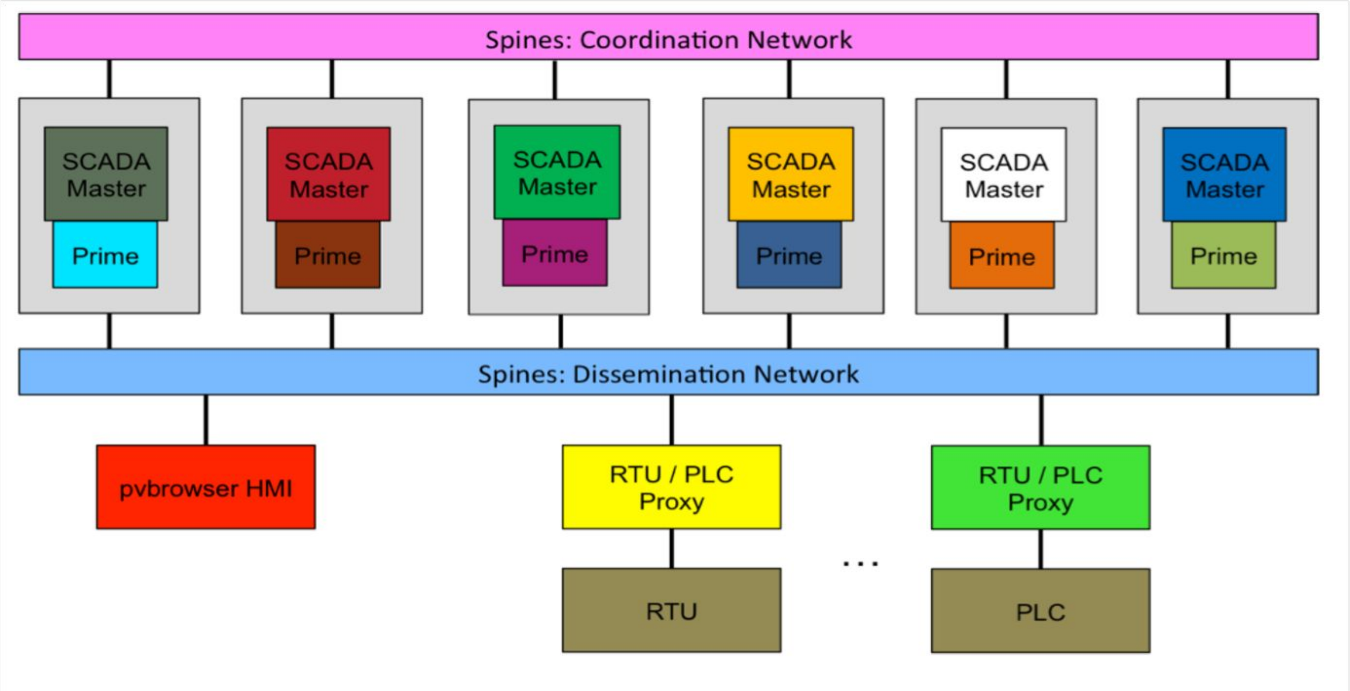


Fig. 2. Fault-free operation of Prime ($f = 1$).

Our Test Bed Environment



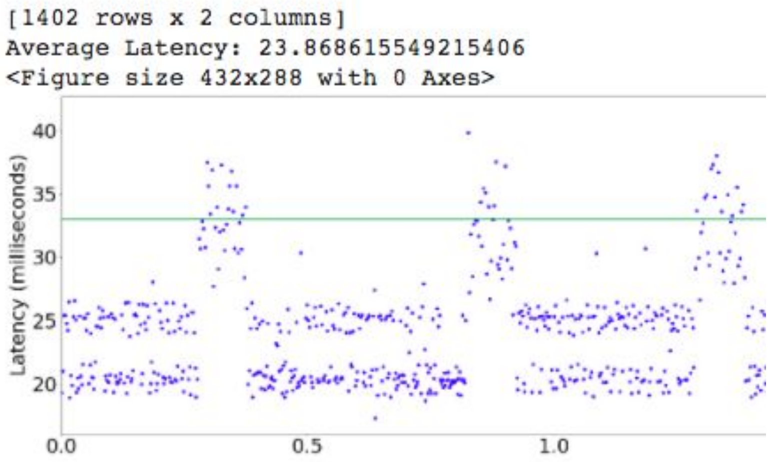
Planning Our Attacks

- Attack Types
  - Internal vs. External
  - Failstop vs. Byzantine
- Combine strategies!
- Measuring Results
  - Latency
  - Resource levels
  - Number of leader changes

RTT Ping DoS Attack

Our Motivation

- Replay packet spam attack showed regular latency spikes
- Isolate and spam that message



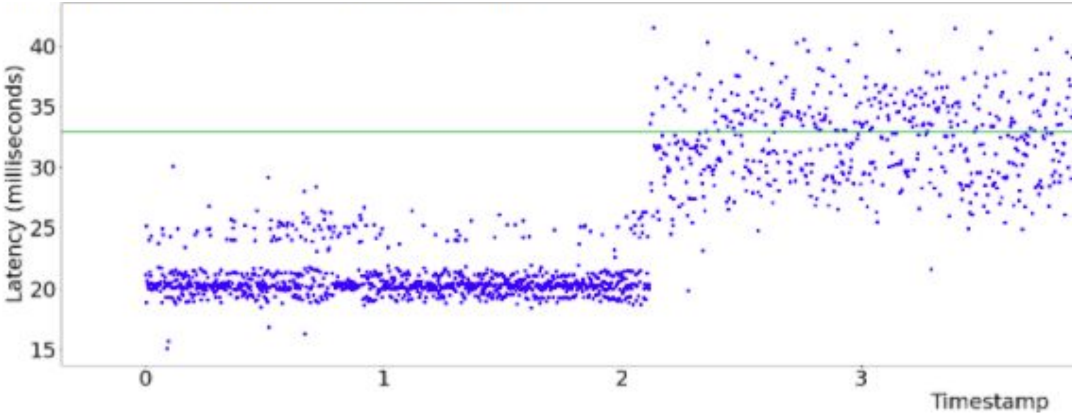
Time (seconds) vs. Latency (ms) for Prime Replay DoS Attack using Prime Client

The Approach

- The culprit: RTT_PING packet type
- Wait until faulty replica generated a RTT_PING packet
- Save packet, send packet to every server repeatedly

Results

- Regularly raised latency above target
- Attack limited by Spines network timeliness protocol



Time (sec) vs. Latency (ms) during RTT_PING Attack using Prime Client

Average Latency: 31.8 ms
10th Lowest (During Attack): 23.9 ms
10th Highest (During Attack): 40.7 ms

Follow The Leader Attack

Prime Suspect Leader Protocol

- The suspect leader sub protocol is incorporated into the prime system to mitigate leader attacks.
- Allows replicas to measure turnaround time of the leader.
  - If leader_tat > accepted_tat, then that leader is suspicious
- Non leaders can reach a consensus to remove a leader.

Our Approach

- Target each current leader with excessive messages using a compromised replica
- Cause a delayed round trip time which will force the leader to be changed
- Cause each leader to be changed to the next leader quickly

Causing a Single Leader Change

- Modifications to Faulty Prime from a RTT Ping DOS attack to targeting a single leader

```
while (1) {
        UTIL_Broadcast(mess);
}
```

→

```
for  (i = 0; i < 1000000; i++) {
        UTIL_Send_To_Server(mess, 1);
}
```

Choosing the Messages

- Most efficient is sending RTT_Ping
  - Why? Leader replies to rtt ping
- We send other messages to non leader replicas
  - we broadcast all messages, other than ping (ie act normally for any other message we handle)

Targeting any Leader to Cause Repeated Changes

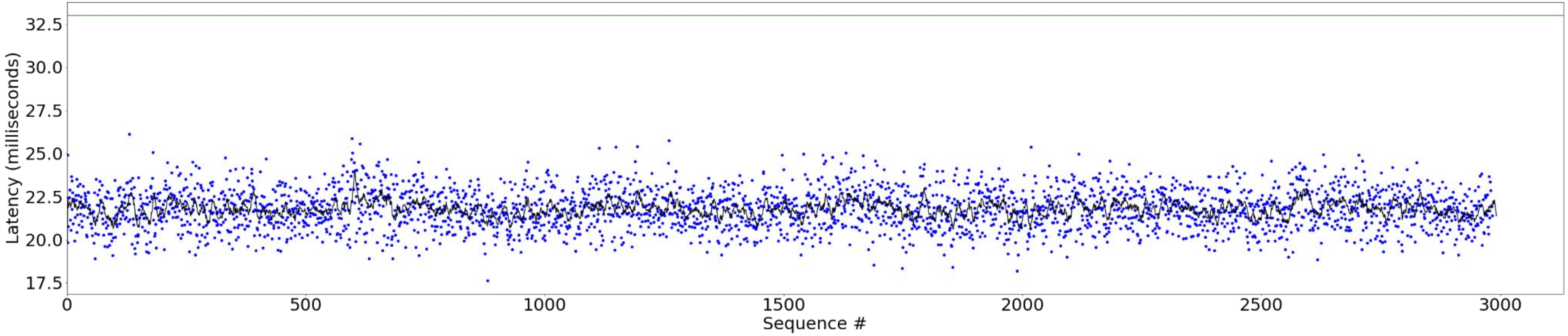
- Target the current leader using the current view
  - (View - 1)Mod6 + 1
- Ping the leader repeatedly while broadcasting all other messages (normal behavior)
- Successful at targeting the current leader while the current view is up to date

Too Many Pongs

- Every Ping will result in a Pong
  - Too many pongs to process
  - View is not updated efficiently, can’t keep track of current leader
- Filter out all message types other than New Leader Proof, New Leader, and Ping Messages when in normal state
  - Pings are used to spam
  - New Leader messages update the view

SPIRE System Baseline

Average Latency (ms): 21.78
Latencies above 33 ms: 0.00%
Latencies above 100 ms: 0.00%

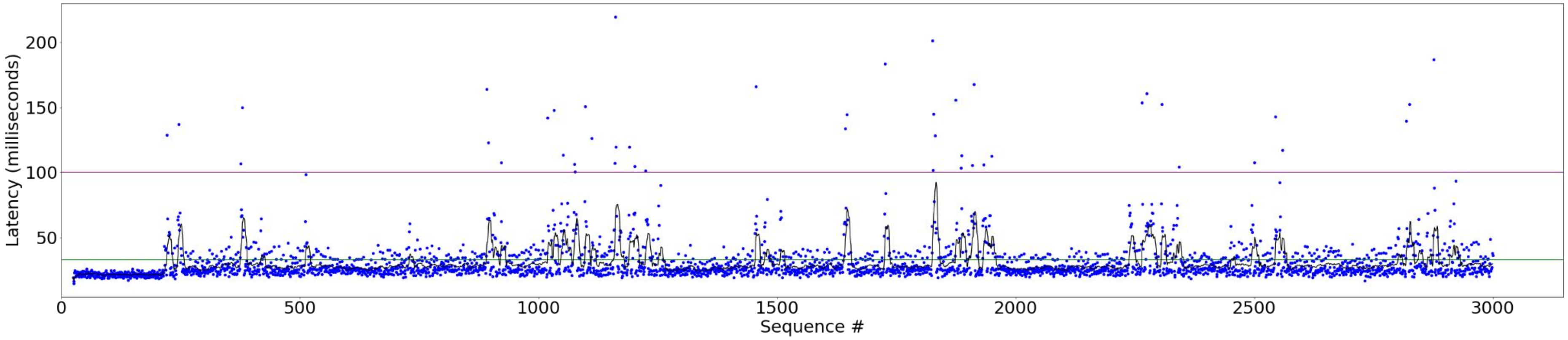


Demo Time!

Follow the Leader - 100% current

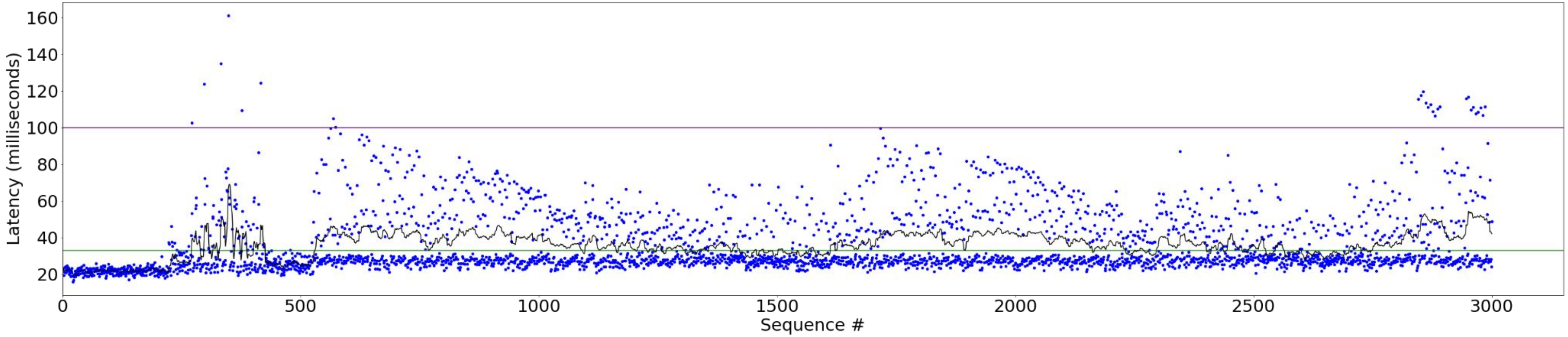
- This is the attack we just demonstrated!

Average Latency (ms):	31.97
Latencies above 33 ms:	25.63%
Latencies above 100 ms:	1.50%



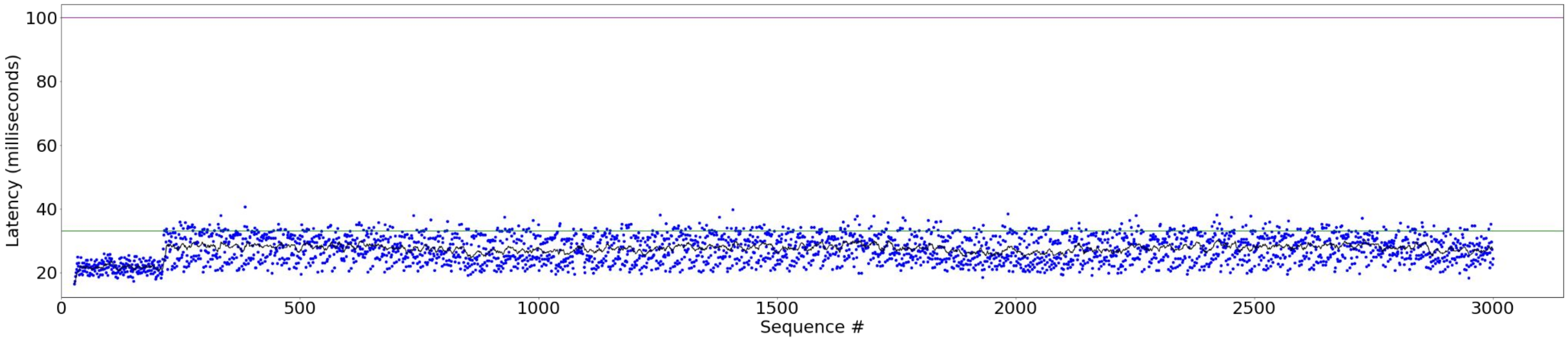
Follow the Leader + Scada1 in Proactive Recovery

Average Latency (ms):	37.62
Latencies above 33 ms:	33.06%
Latencies above 100 ms:	0.90%



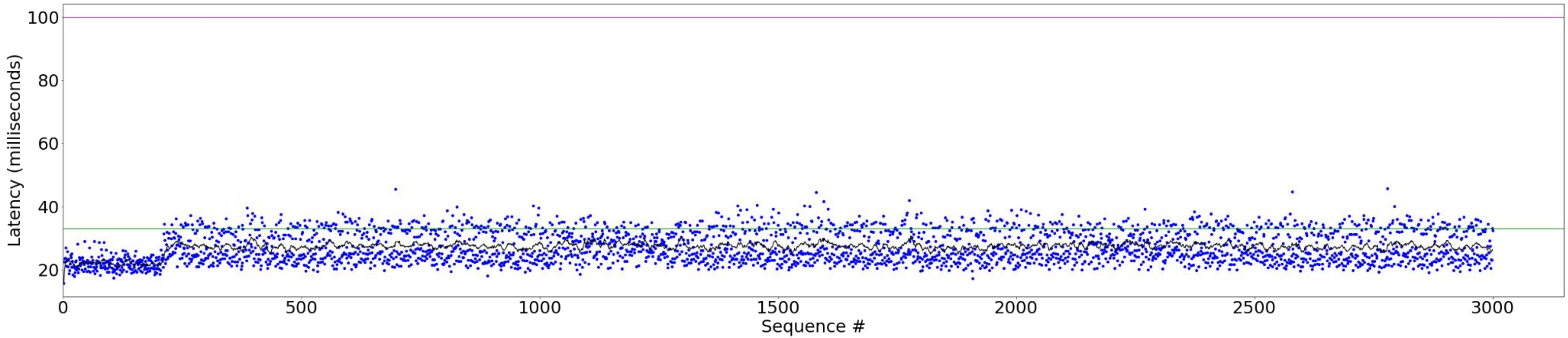
Follow the Leader - 50% current, 50% next

Average Latency (ms):	27.70
Latencies above 33 ms:	10.37%
Latencies above 100 ms:	0.00%



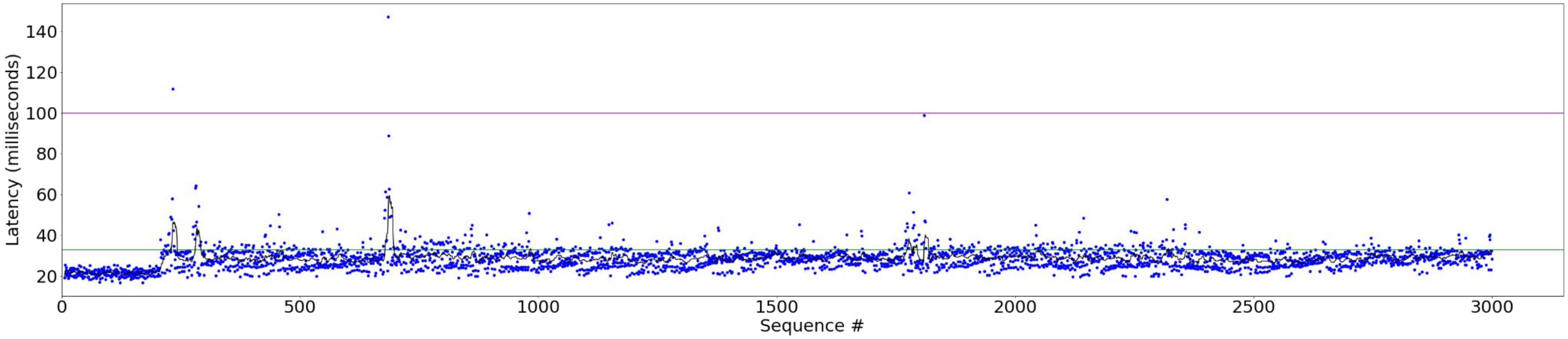
Follow the Leader - 75% current, 25% next

Average Latency (ms):	27.49
Latencies above 33 ms:	16.80%
Latencies above 100 ms:	0.00%



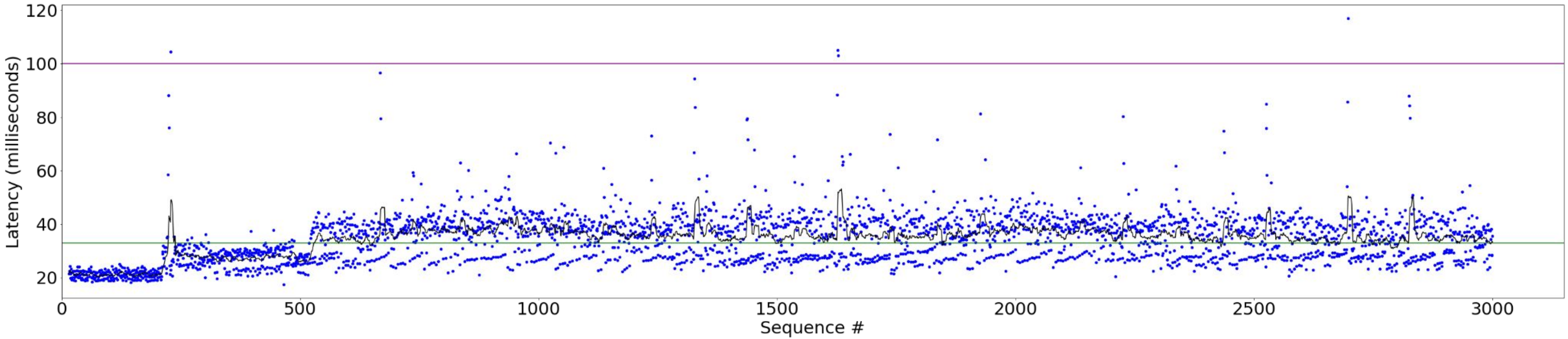
Follow the Leader - 90% current, 10% next

Average Latency (ms):	29.07
Latencies above 33 ms:	12.60%
Latencies above 100 ms:	0.07%



Follow the Leader - 90%/10% + Scada1 in Proactive Recovery

Average Latency (ms):	36.44
Latencies above 33 ms:	54.87%
Latencies above 100 ms:	0.13%

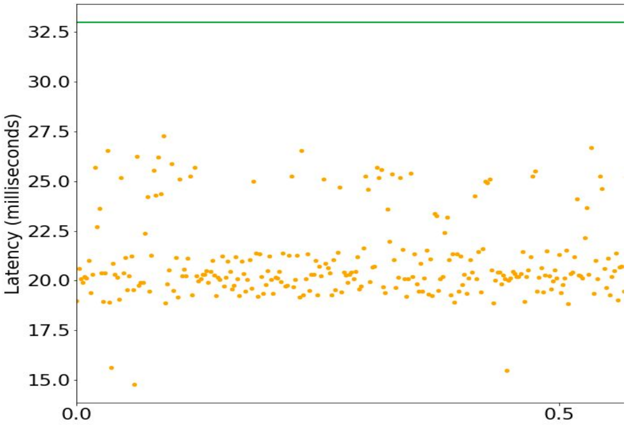


Questions?

Pre-Order Memory Consumption Attack

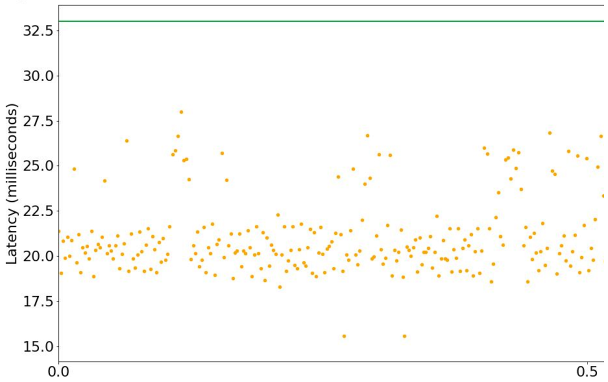
Previous Sequence Number Attacks

Improper Sequence Number Update



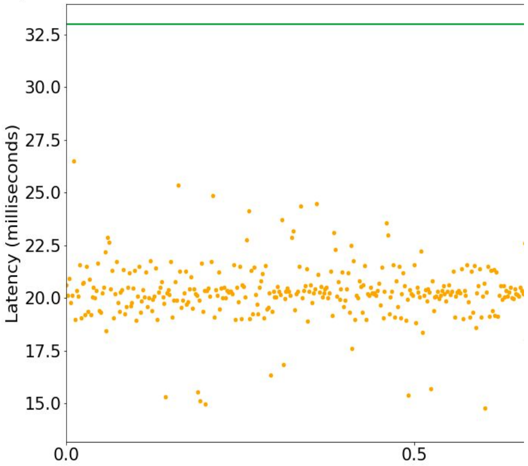
Average Latency: 21.1036558908

No Sequence Number Update



Average Latency: 20.8574664536

Spam 10,000 / message



Average Latency: 20.24648035190615

The BACKGROUND

- The key is **INTEGRITY.**
- Every replica must save update information until it is executed
- All updates must be executed in order
- A replica can only flush old updates once they have been executed

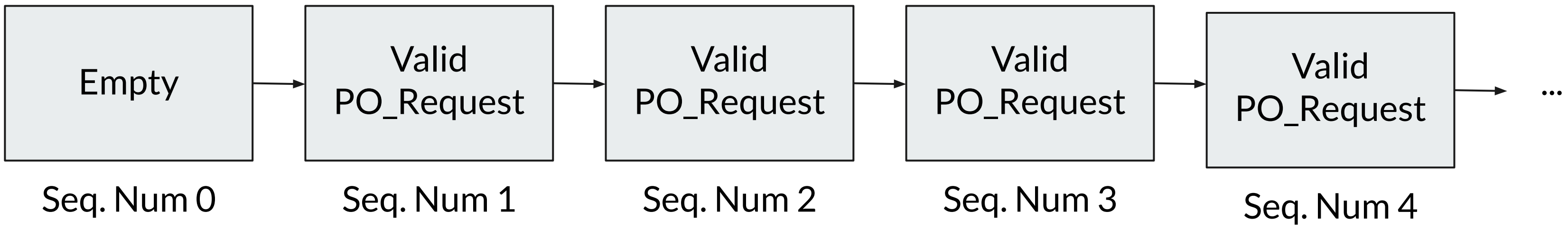
```
a = 0
a = 5
a += 1
```

VS.

```
a = 0
a += 1
a = 5
```

The Attack

- Skip a sequence number, lengthen data structure to eat up RAM
- Generate valid PO_Requests and send to all replicas
- Assure we always have a client update to order



Demo Time!

Problems We Faced

- Assure list of updates does not grow infinitely and consume memory
- We store our own PO_Requests, would also eat our memory
- Work around catch up protocol
- Implementation Bugs

Results

- With Spam, **16GB of RAM** is consumed in **under 15 minutes**
- Spam and no-spam variants
  - Spam variant works quickly, can be detected
  - No-spam variant works more slowly, goes undetected by IDS
- Non-spam attack variant goes undetected by NIDS
- Once RAM limit is reached, replicas become increasingly unresponsive
- Implementation bugs

Questions?

Future Steps

- PO Request Attack
    - Increase Reliability
    - Test with Intrusion Detection System
- Follow the Leader Attack
    - Control Leader while in Proactive Recovery

Mitigation

- Memory Attack: Bound the memory that one server can consume on another server
  - Bounded queue of updates
- Follow the Leader Attack:
  - Rate Limiting

Thank You!

Yair Amir
Sahiti Bommareddy
Daniel Qian
Jerry Chen
And, the rest of the SFRC class

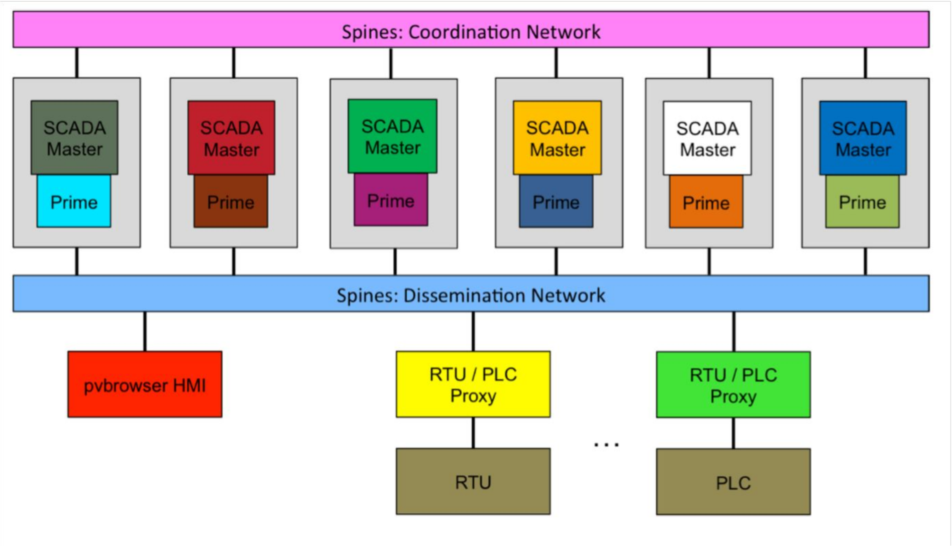
Questions?

(Conclusion) … So were we successful?

TL;DR - The Spire System

- Spines creates an intrusion-tolerant reliable network that isn't vulnerable to conventional network attacks (DOS, MITM, BGP Hijacking)

- Prime ensures that our distributed system maintains correctness while executing commands in a timely manner.



Follow the Leader - 50% current, 50% next (a closer look)