Remote Filesystem Event Notification and Processing for Distributed Systems

Authors – Kushal Thapa, Vinay Lokesh, and Dr. Stan McClellan

Presenter
Vinay Lokesh
Dept. of Computer Science
Texas State University
San Marcos, TX, USA
e-mail: v_v183@txstate.edu

April 2021

TEXAS STATE UNIVERSITY
The rising STAR of Texas

Vinay Lokesh
Graduate Research Assistant

Texas State University

v_v183@txstate.edu

+1 (404)-647-9907

Vinay Lokesh is currently serving as a Graduate Research Assistant at Texas State University. His current research involves developing remote filesystem notification on Linux environment and a python-based control interface for electric power system. He previously served as an Application Development Analyst at Accenture.

Vinay’s expertise spans Java Programming, Database and Software Development. He is currently a graduate student at Texas State University pursuing M.S in Computer Science. He also has an MCA from R.V College of Engineering.

Introduction

Why Remote Filesystem Monitoring?

System Security

Data Acquisition

Risk Analysis

Problem

Distributed systems may have difficulty in monitoring remote filesystem events in a loosely coupled and distributed architecture.

Solution

Enabling secure remote filesystem monitoring using existent operating system resident tools.

Background Work

Some of tools which work well for monitoring filesystem locally but lack the ability to monitor remote filesystems:

- inotify
- Direvent
- iWatch
- Kqueue
- FSEvents
- FileSystemWatcher
- Python Watchdog

Approach

1. **Building a simple and secure network architecture**.
   - Using Multiplexing
   - Using Reverse Port Forwarding

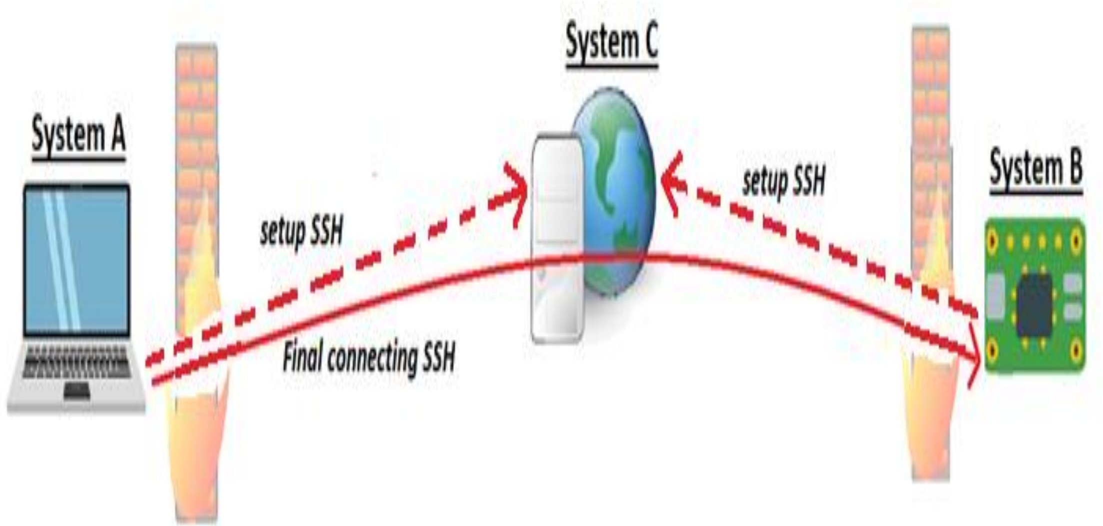

Figure 1. Representation of three-prong architecture. **Arch-1**

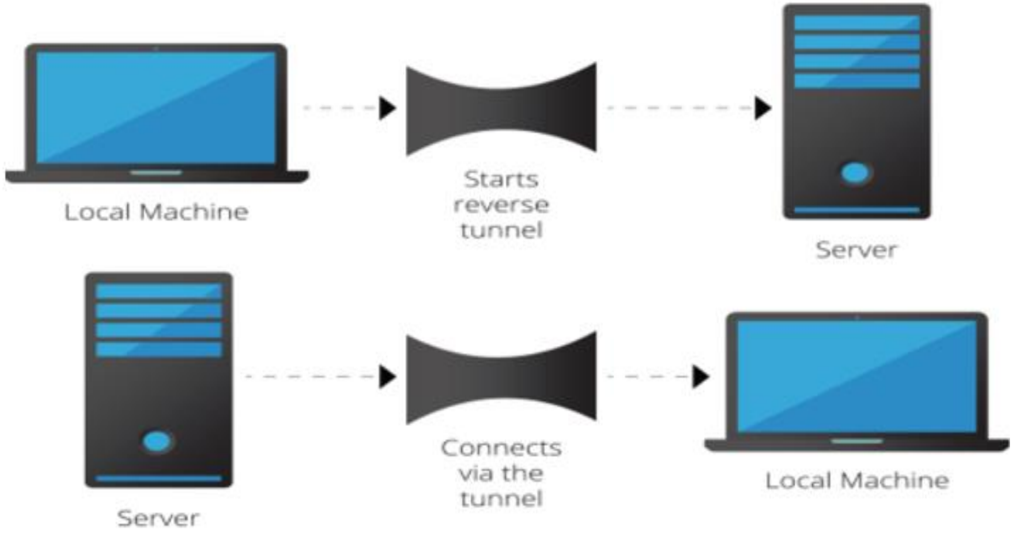

Figure 2. Working of SSH Reverse Port Forwarding

- System A is used to control remote devices behind a firewall, represented by System B.
- System A and System B are behind network firewalls, so a direct SSH connection cannot be made from A to B or vice versa. Thus, there is a need of system C, which is an open IP reachable server.
- Using port forwarding, an SSH tunnel can be made from A to B via C

Approach

Two other simpler network architectures – Arch-0 and Arch-2 were built using the components of our primary architecture (Arch-1) to compare the results.

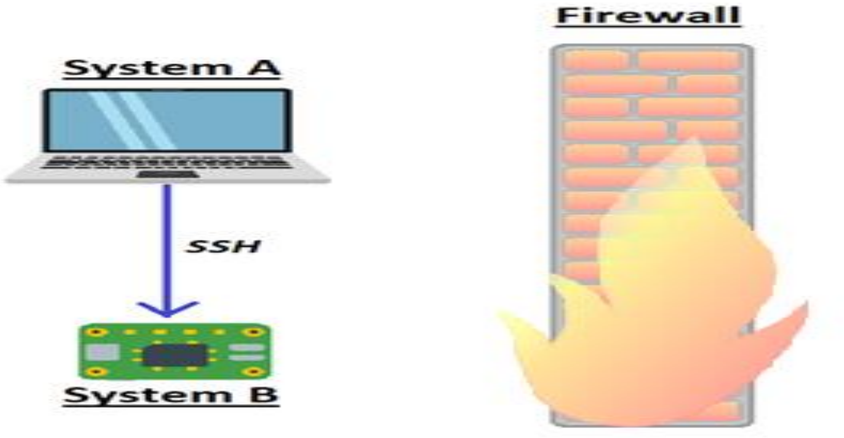

Figure 3. Representation of three-prong architecture. **Arch-0**

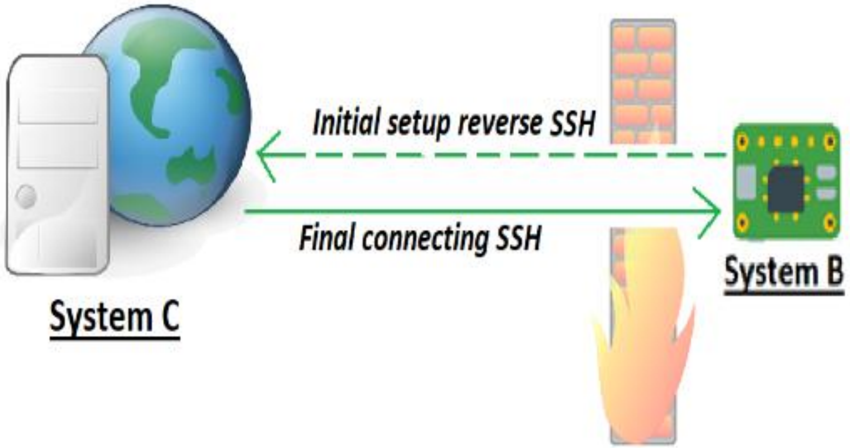

Figure 4. Representation of components, security zones and connections in a client-server architecture. **Arch-2**

Approach

- Arch-1, Arch-0, and Arch-2 is indicated by red, blue and green, respectively.
- Bold lines and thin lines are indicated by multiplexed and non-multiplexed connections, respectively.
- The dashed lines represent connections necessary for their corresponding architecture.

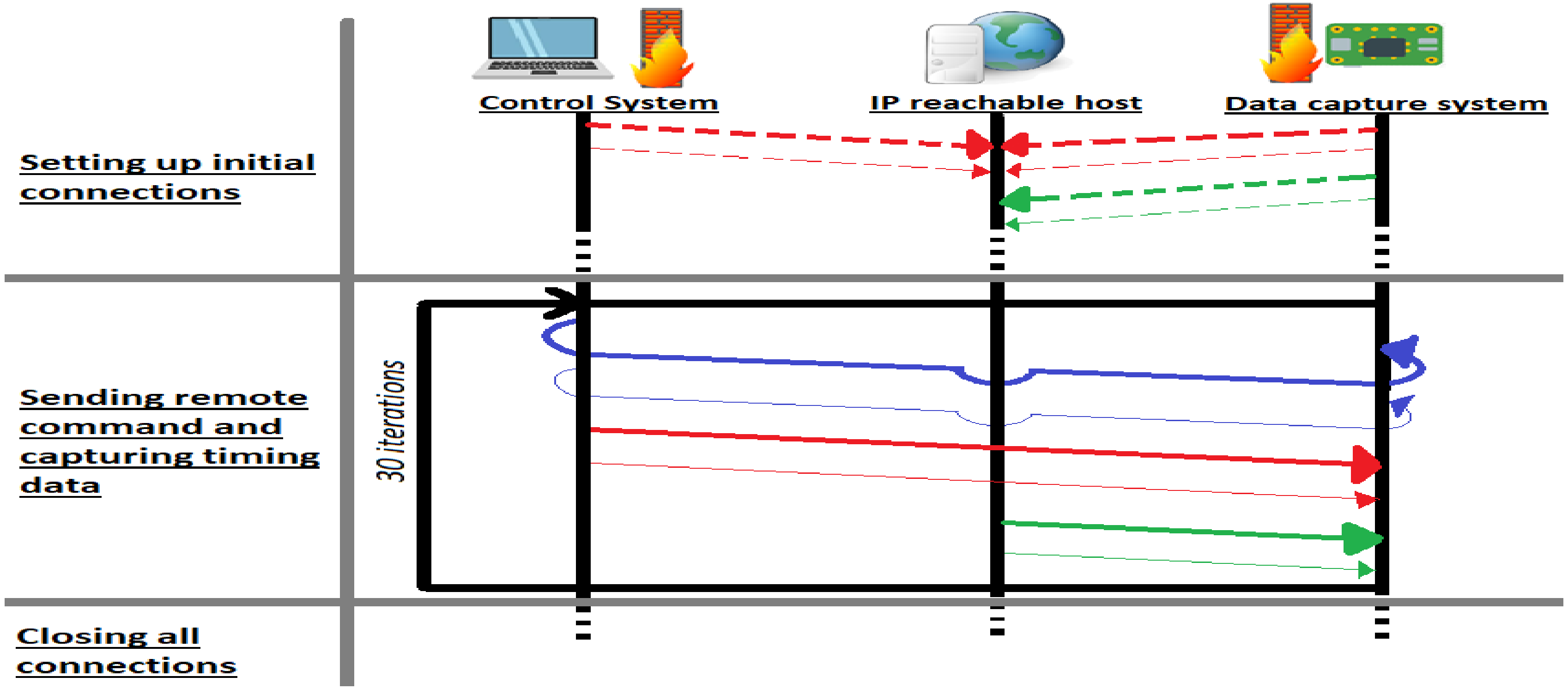

Figure 5. Vertical line diagram to represent three network architecture, Arch-0, Arch-1 and Arch 2.

Approach

2. Using the network architecture with Remote Filesystem monitoring

- Using SSHFS
- Using SSH

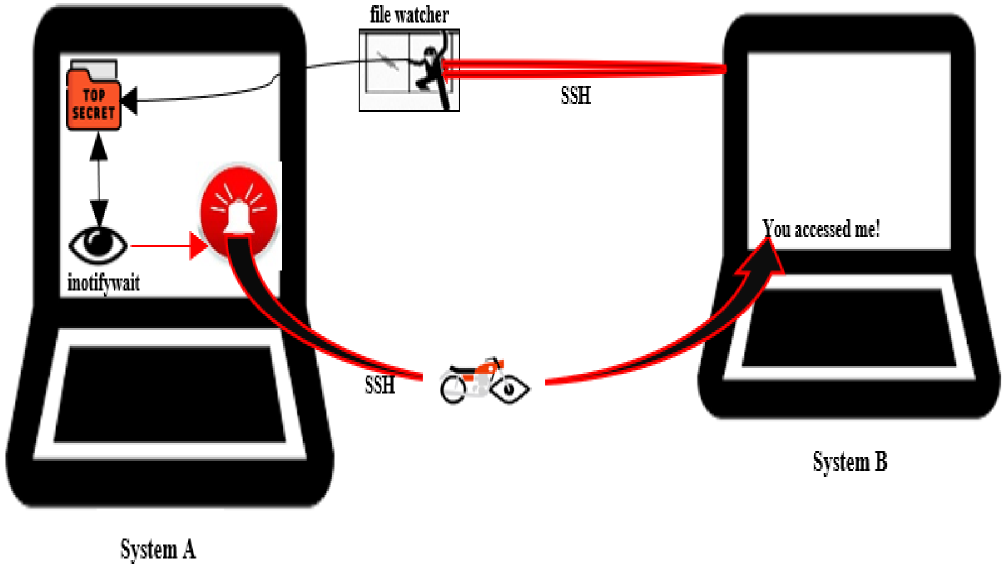

Figure 6. Remote Filesystem monitoring using SSH and inotify tools.

- The target file or directory in the local system is monitored using inotifywait.
- Using the event registration of the monitored target as a trigger, a command is sent to the other end of the channel using SSH.
- An inotifywait is issued on a secret directory in System A, so whenever a filesystem event occurs in that watched section of the filesystem, the event is transmitted to the remote monitoring configuration on System B along with a timestamp of the event.

Results

- Multiplexed SSH connections significantly reduce connection time.
- Arch-0 exhibits the fastest communication time in both multiplexed and non-multiplexed architecture.

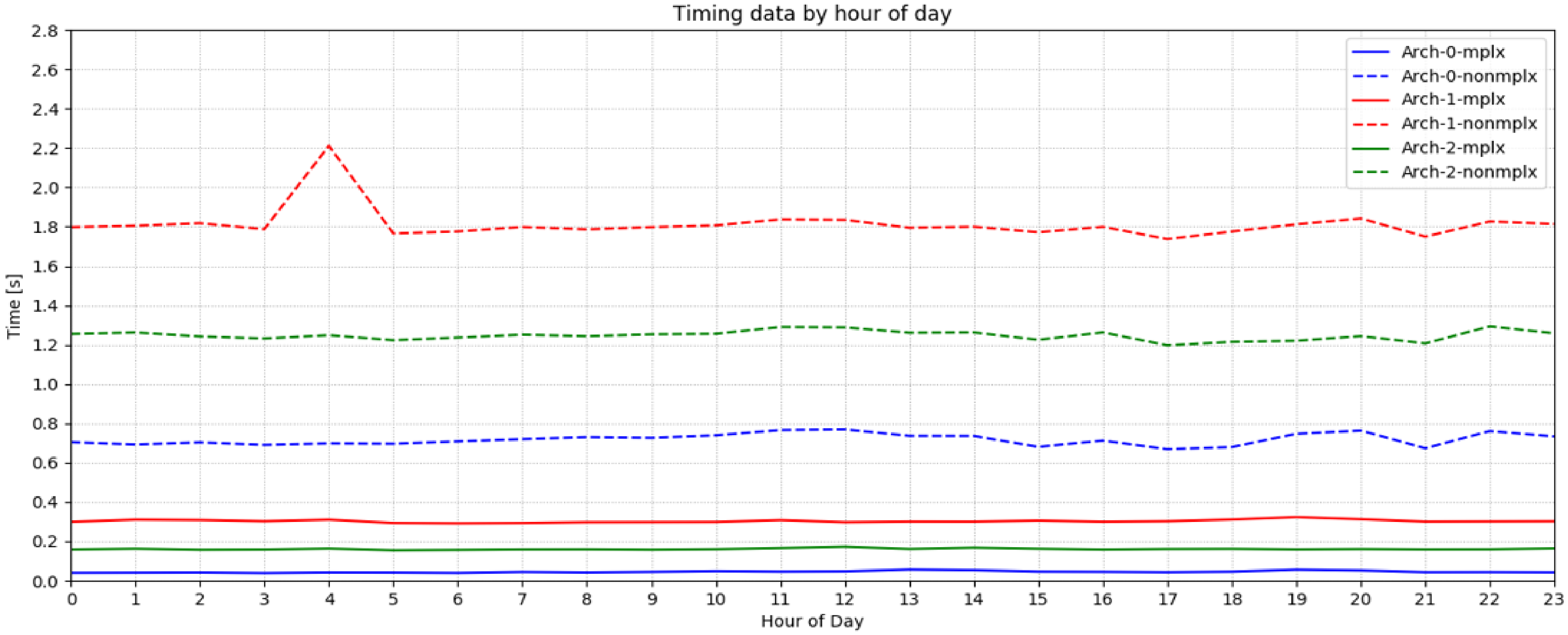

Figure 7. Timing data by hour of the day.

Results

- The multiplexed connection of Arch-1 recorded lower time than non-multiplexed version of Arch-0.
- The non-multiplexed connections exhibit substantial random latencies.

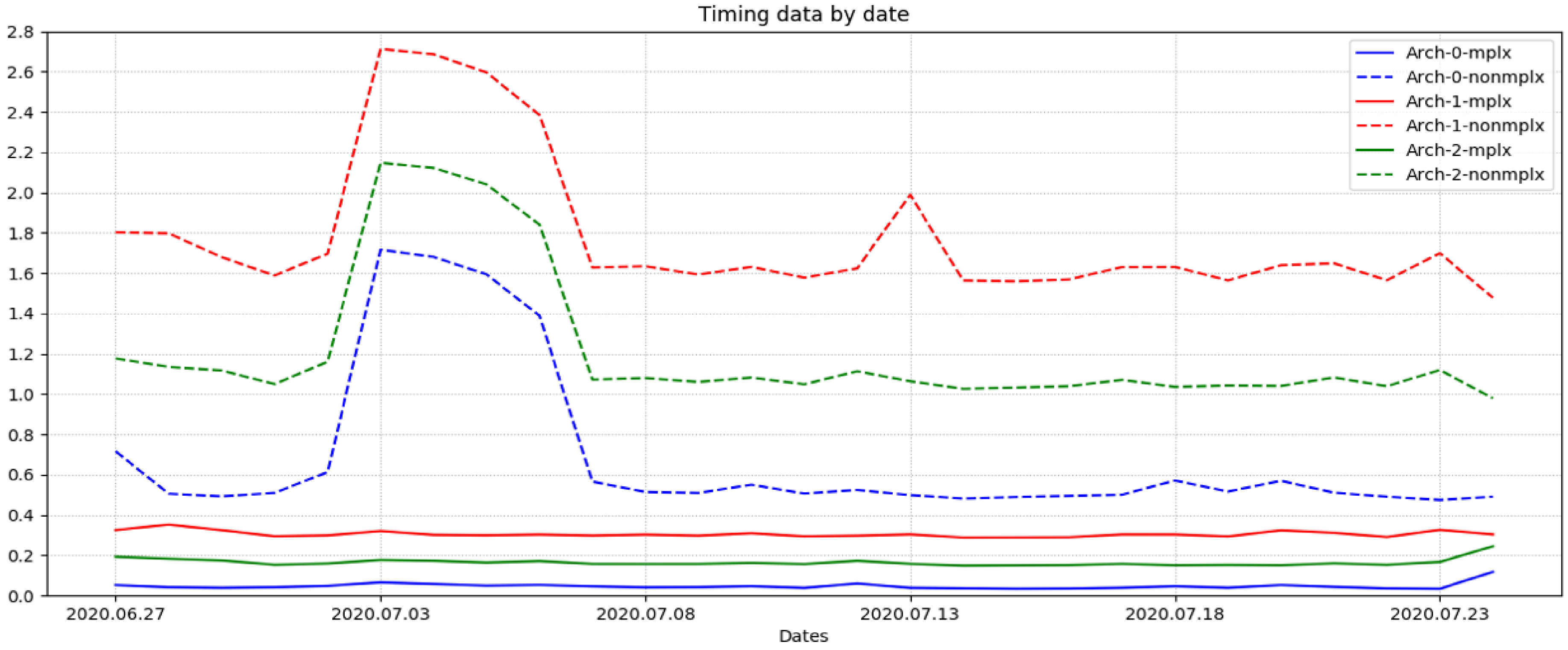

Figure 8. Timing data by date

Conclusion

- In a distributed and loosely coupled architectures, monitoring of filesystem events on remote systems, possibly behind firewalls, can have important application-layer benefits and utility.

- Simple and scalable technique using multiplexed SSH connections and inotify tools enables secure remote file system monitoring with minimum overhead.

- By recording timing of filesystem events on each of these network architectures, we note that multiplexed SSH connections are consistent, and much more efficient than other methods