Recovering from intrusions in distributed systems with Dare

Taesoo Kim

Ramesh Chandra, Nickolai Zeldovich

MIT CSAIL

Attackers routinely compromise distributed systems

Kernel.org Claims Cyber Attack

Written by Erica Thinesen view bio

Since starting a freelance writing business in 2006, Erica J. Thinesen has been working with small and medium-sized businesses to create...

01 September, 2011 Like 2 **kernel** **security** **linux kernel**

A few crucial servers at Kernel.org were attacked by hackers, the company recently wrote on its website.

Kernel.org is the most important distribution facility for a wide range of Linux-based sc itself. According to upda the site, the intrusion wa August 28.

sourceforge Find Open Source Software Browse Blog Support

SourceForge.net > Blog

Sourceforge Attack: Full Report

Posted on Saturday, January 29th, 2011 by **admin**

Category: **General**, **Site Status**

As we've previously announced, SourceForge.net has been the target of a directed attack. We have completed the first round of analysis, and have a much more solid picture of what happened, the extent of the impact, our plan to reduce future risk of attack. We're still working hard on fixing things, but we wanted to share what we know with the community.

We discovered the attack on Wednesday, and have been working hard to get things back in order since then. While several boxes were compromised we believe we caught things before the attack escalated beyond its first stages.

Recovery is manual and time-consuming

- Example: SourceForge.net attack
  - A hosting site for open source projects (>300K)

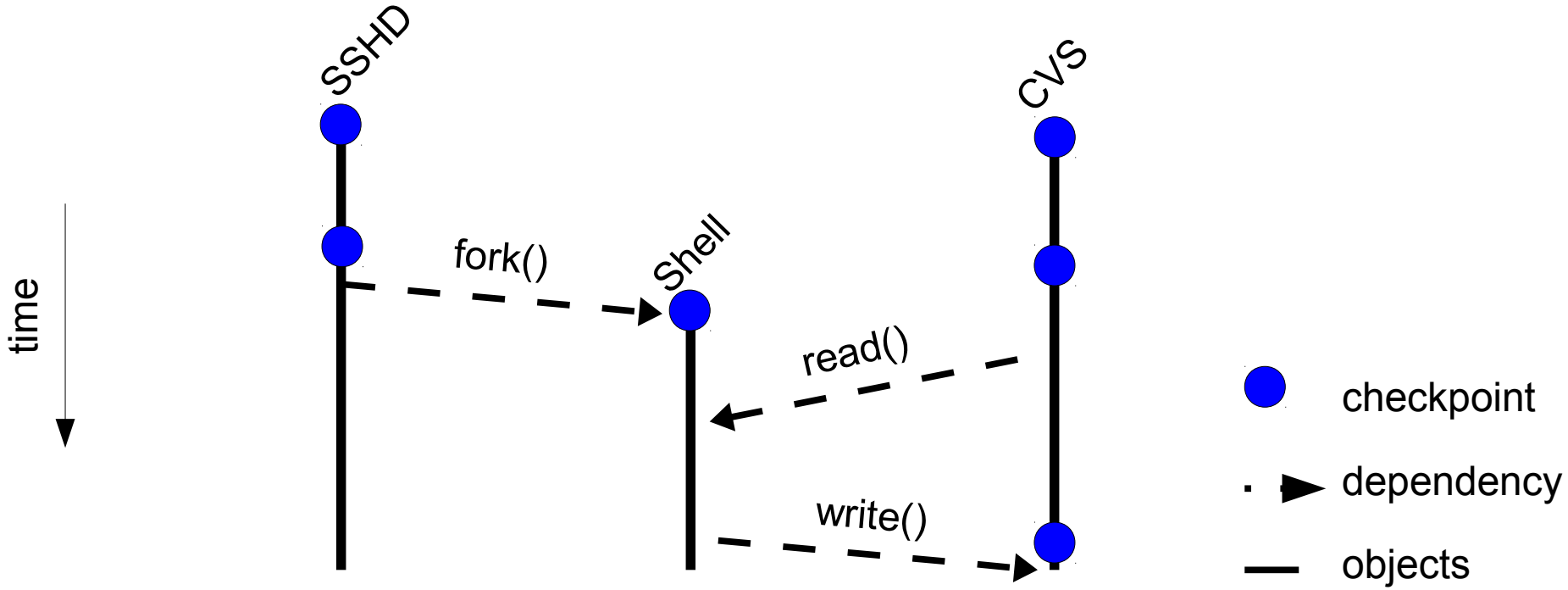
Jan 26, 2011	An operator detected a targeted attack Shutdown CVS, SSH and WebVC services
Jan 28, 2011	Reset passwords of 2 million users
Jan 29, 2011	Validate data such as commits and releases Restore services after fixing the bug

Retro: automatic recovery in a single machine

- Normal execution:
  - Record information about the system execution
  - Build a dependency graph of a system

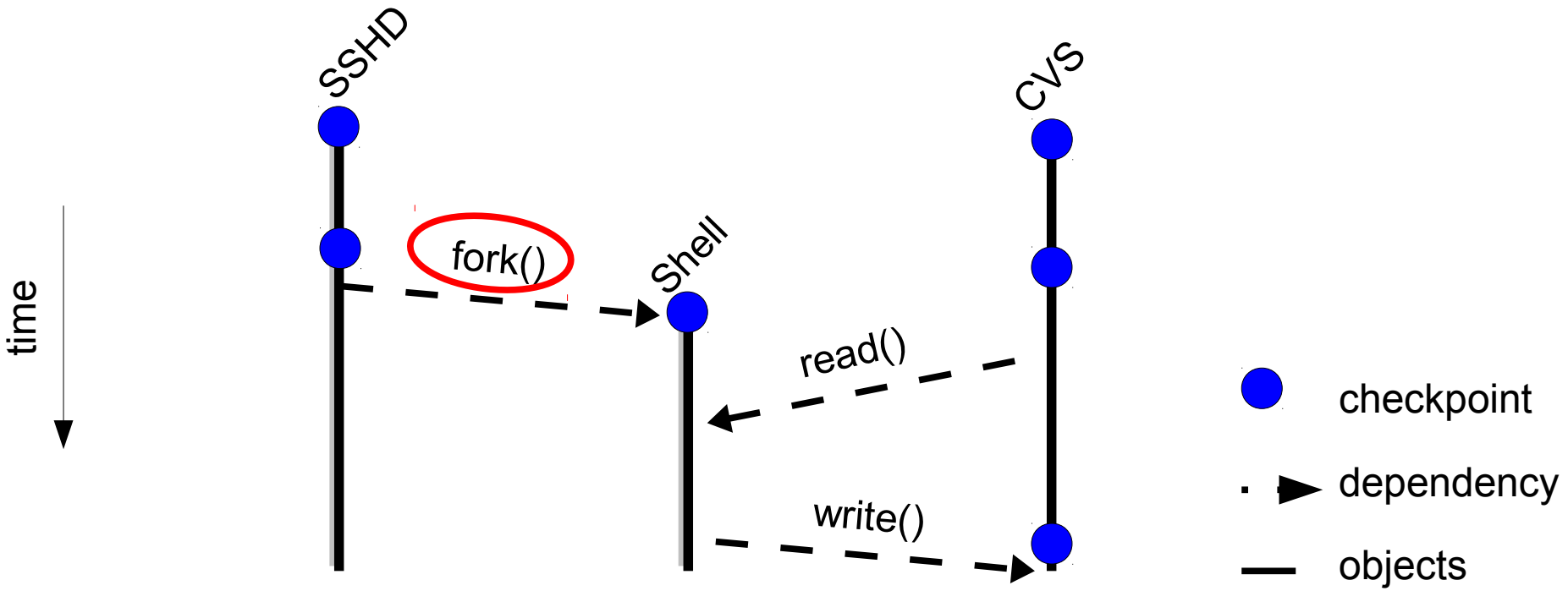
Review: Action History Graph (AHG)

- Objects: **data** (e.g., file) and **actor** (e.g., process)
  - **Checkpoint**: snapshot of state at a particular time
  - **Action**: unit of execution
- Each action has **dependencies** from/to objects



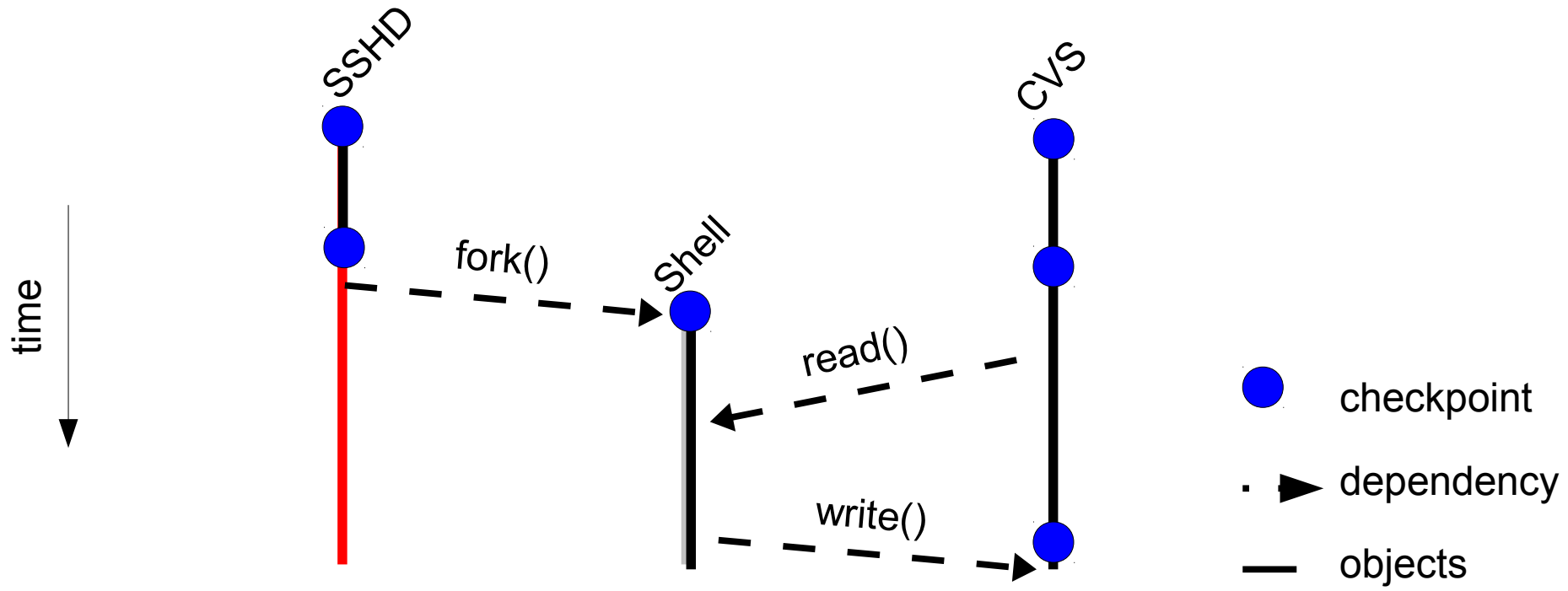
Review: repair with selective re-execution

- Need to specify the attack action (e.g., fork)



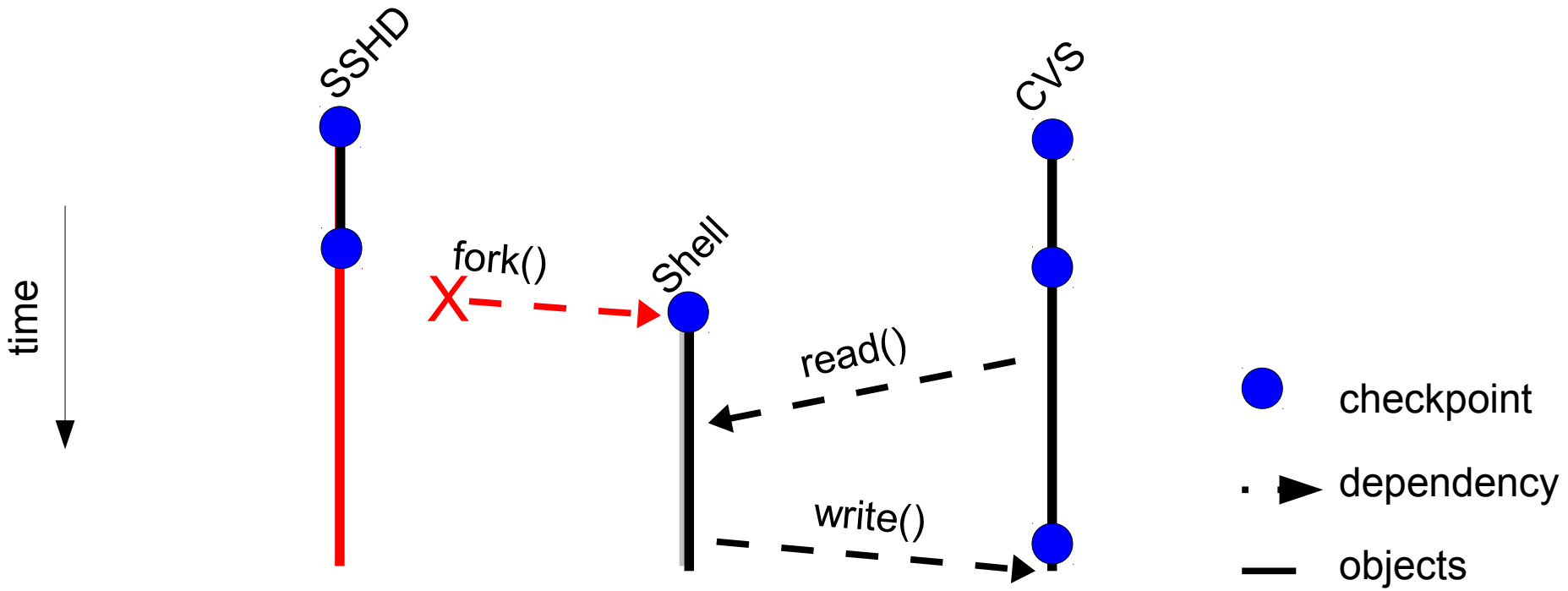
Review: repair with selective re-execution

- Need to specify the attack action (e.g., fork)
- **Rollback** objects affected by the attack



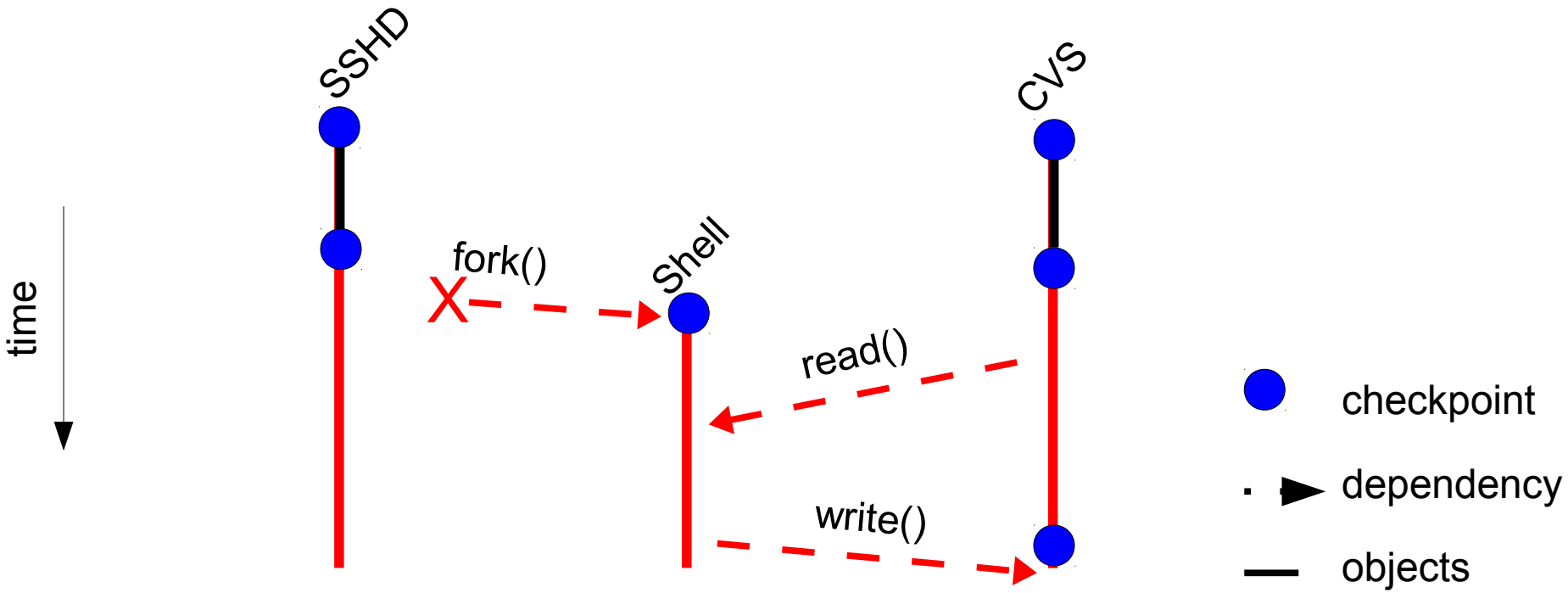
Review: repair with selective re-execution

- Need to specify the attack action (e.g., fork)
- **Rollback** objects affected by the attack



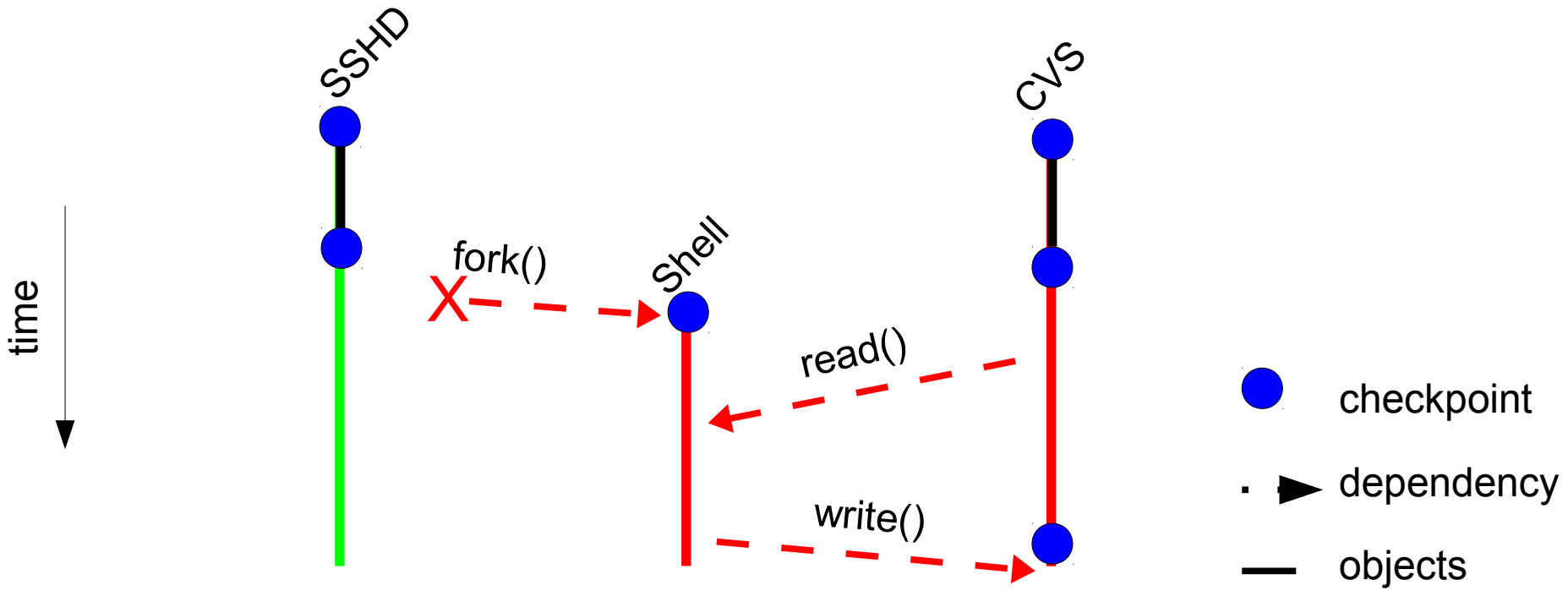
Review: repair with selective re-execution

- Need to specify the attack action (e.g., fork)
- **Rollback** objects affected by the attack

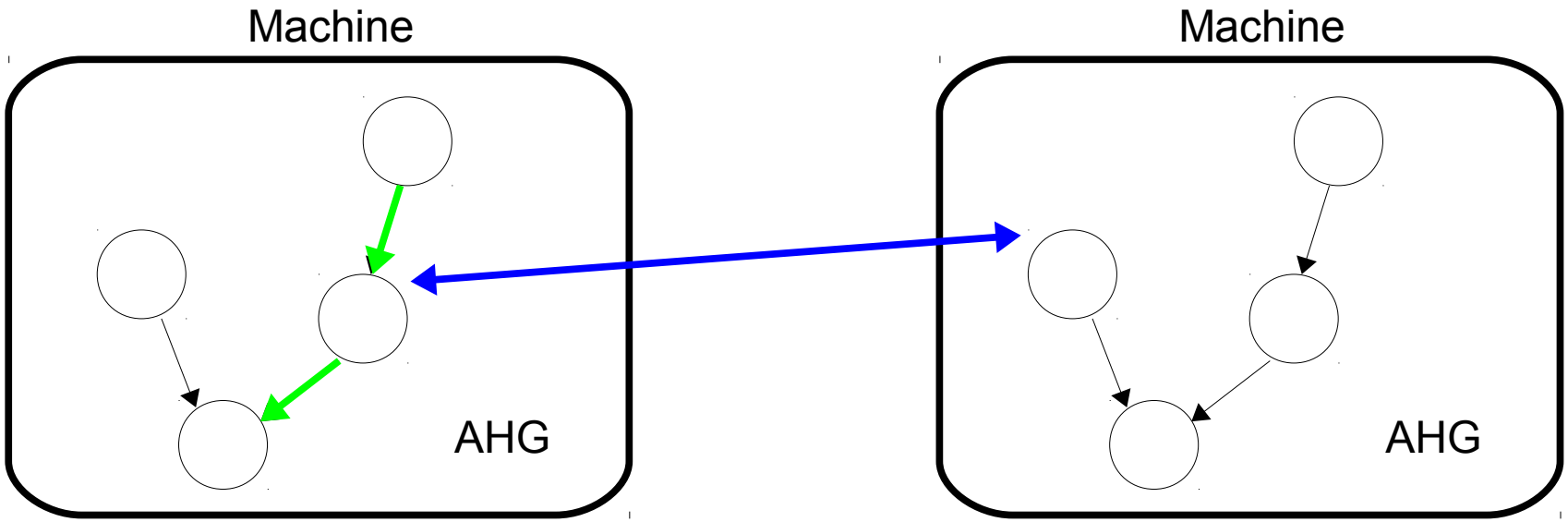


Review: repair with selective re-execution

- Need to specify the attack action (e.g., fork)
- **Rollback** objects affected by the attack
- **Re-execute** the rest of the actions



Challenges

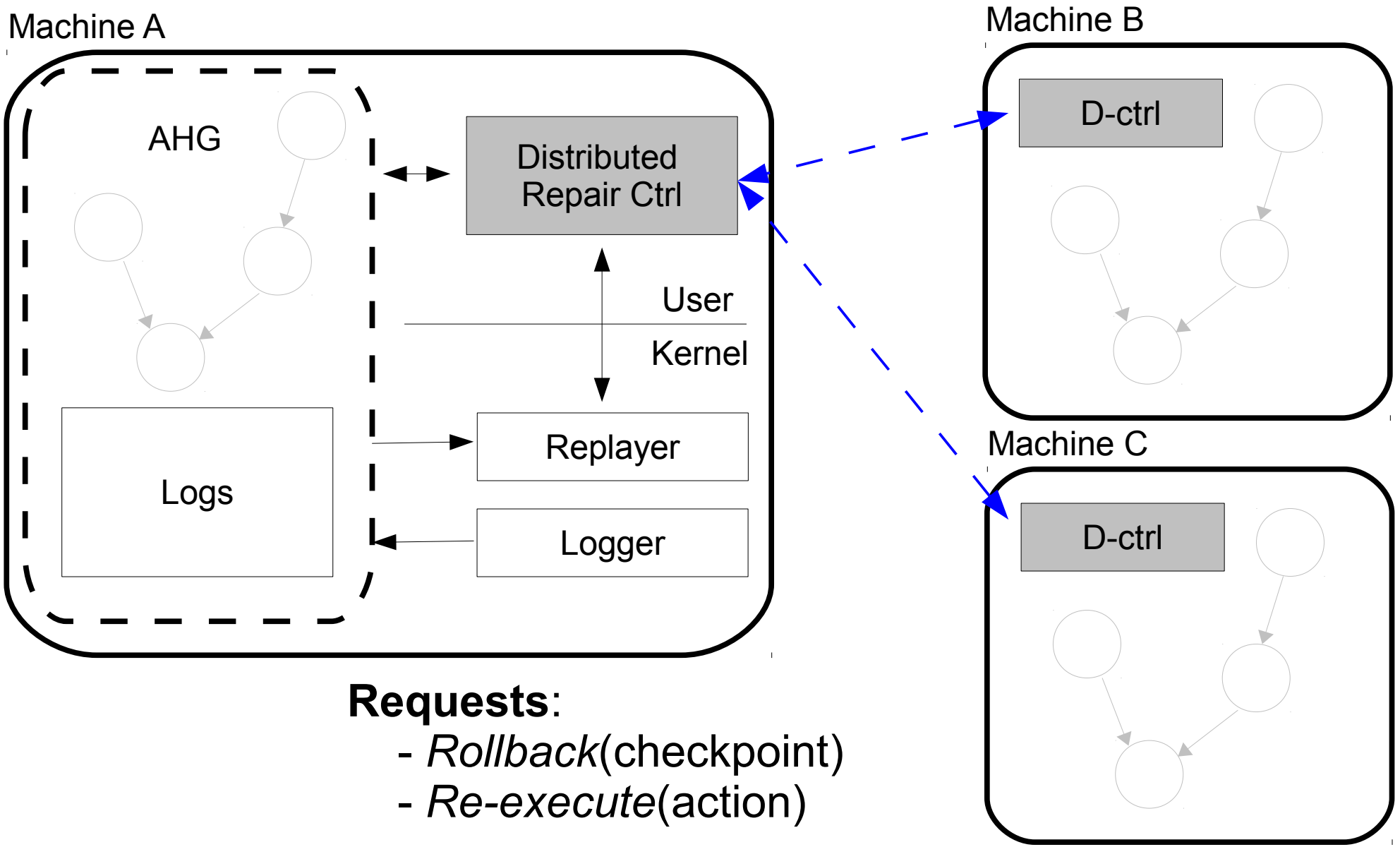


1. How to record dependencies across machines?

2. How to replay network connections?

3. How to minimize re-exec. of long-lived process?

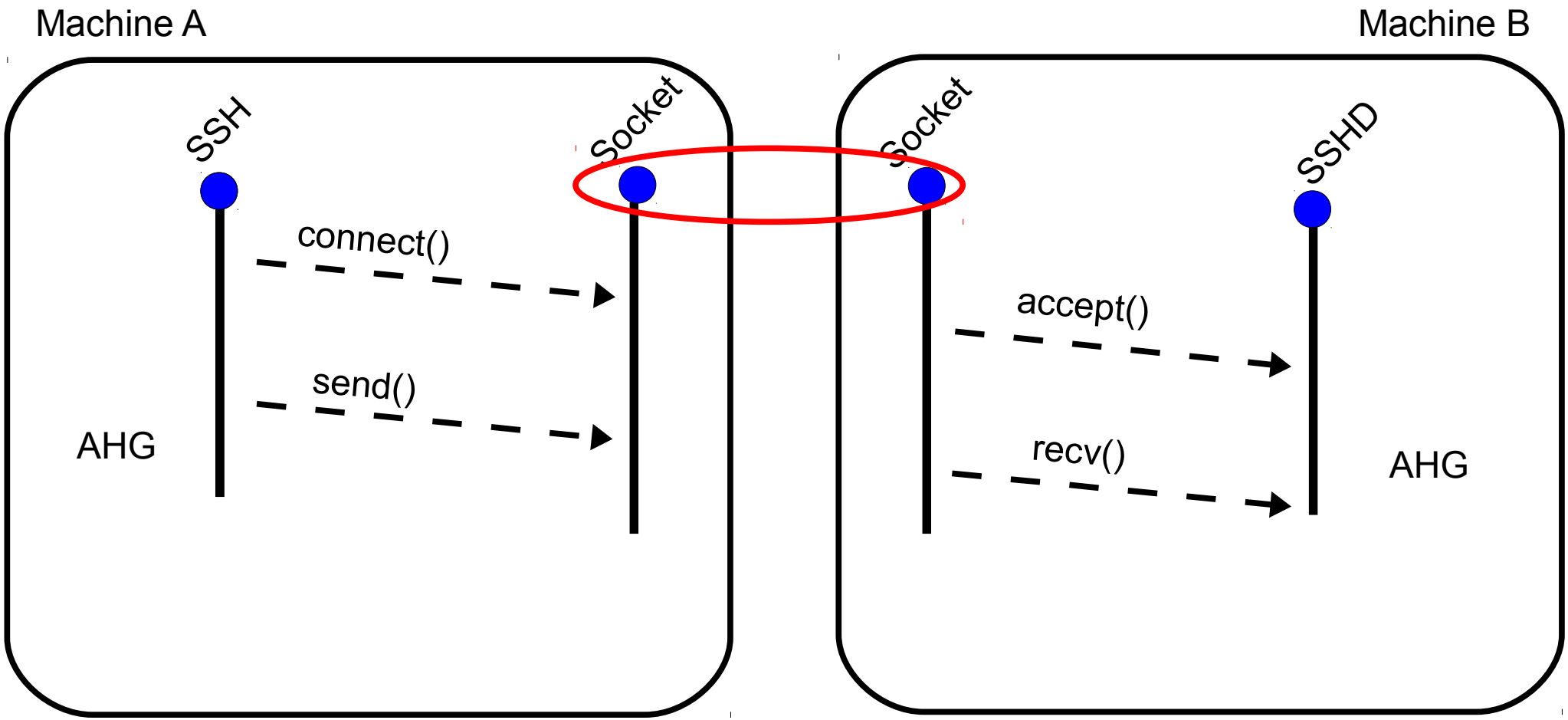
Overview of DARE's design



Requests:

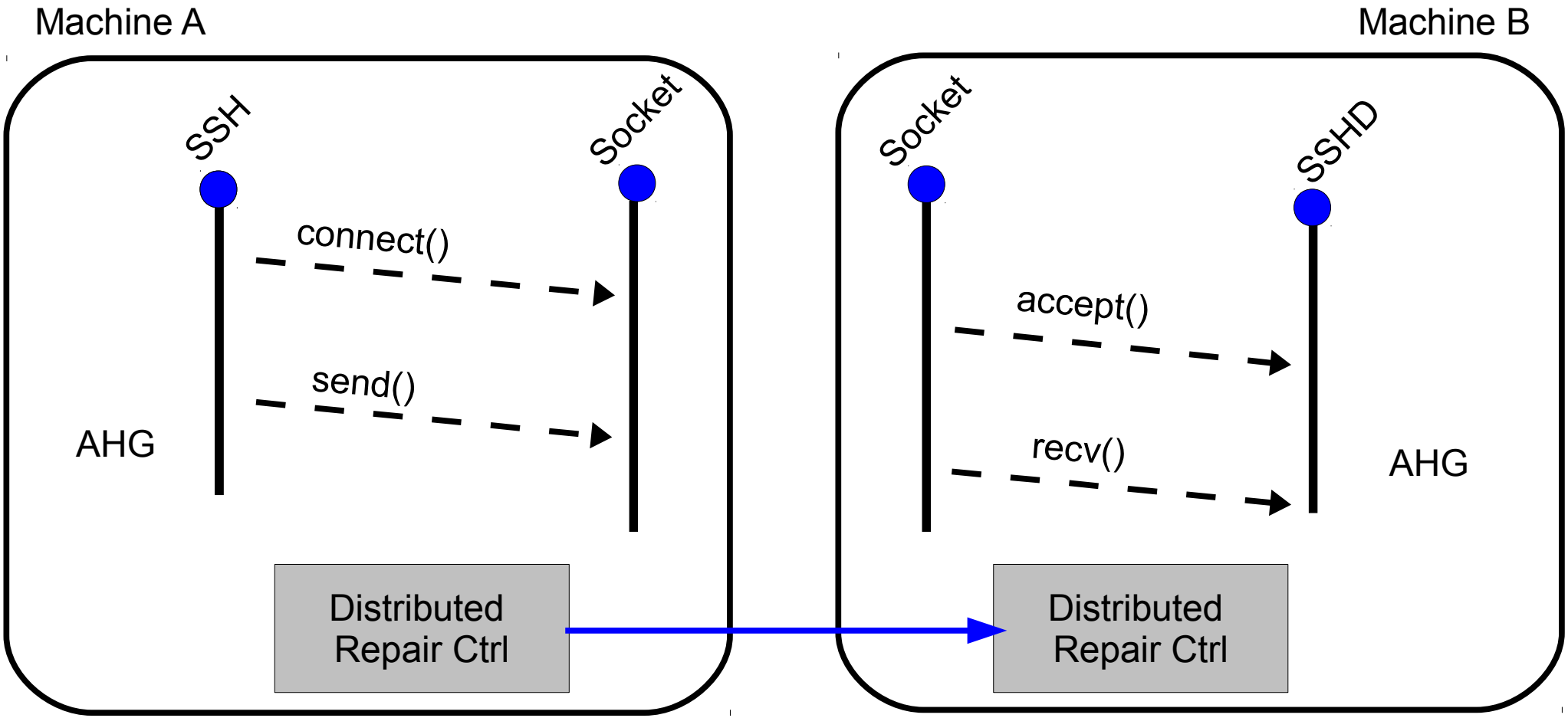
- *Rollback*(checkpoint)
- *Re-execute*(action)

Recording dependencies across multiple machines



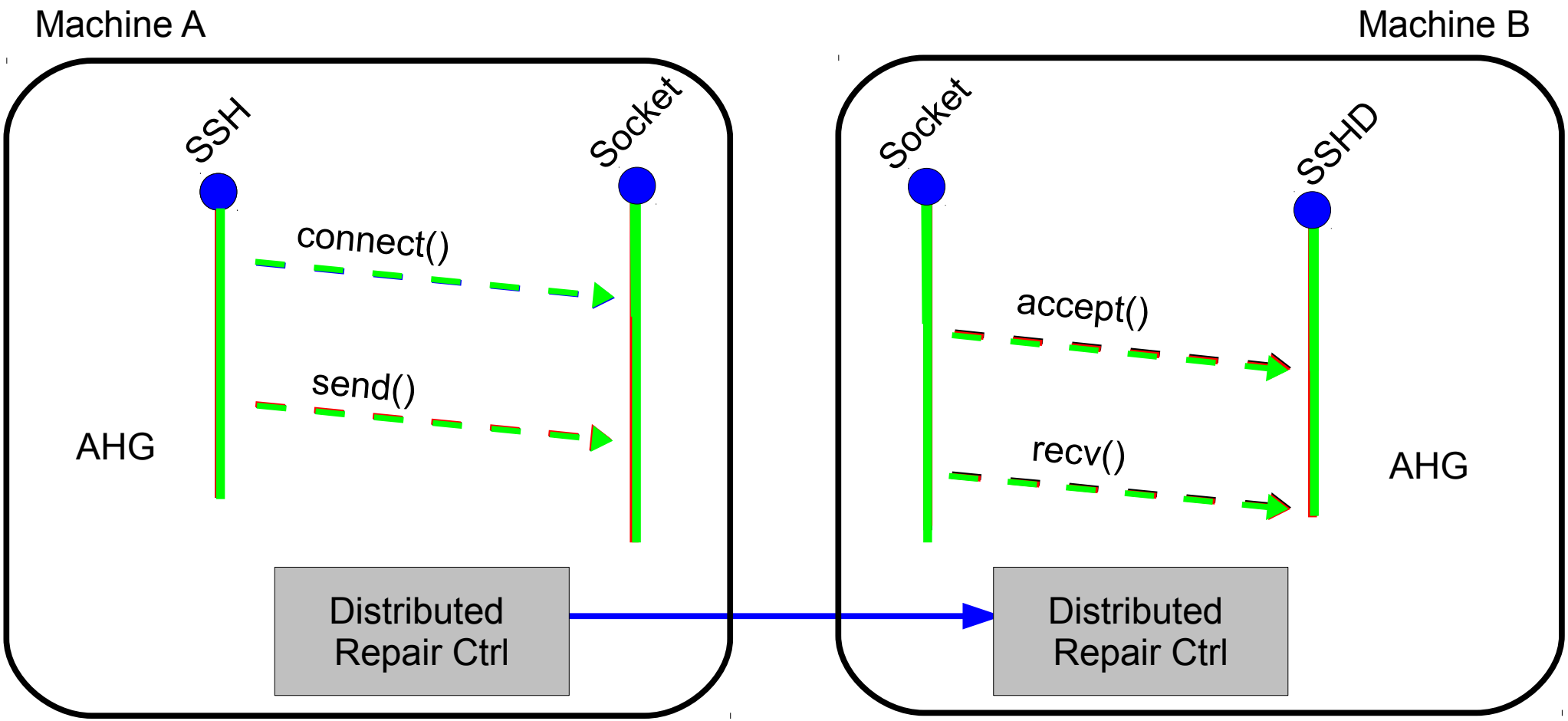
What if same IP and port used multiple times?

Approach: assign unique id to sockets



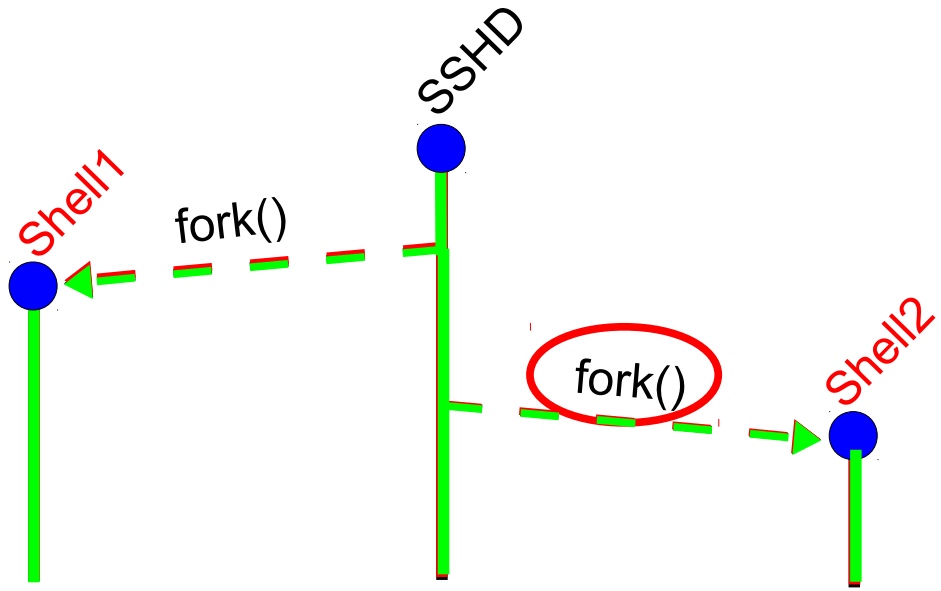
Send socket's ***unique id*** to the receiver

Repair network connections



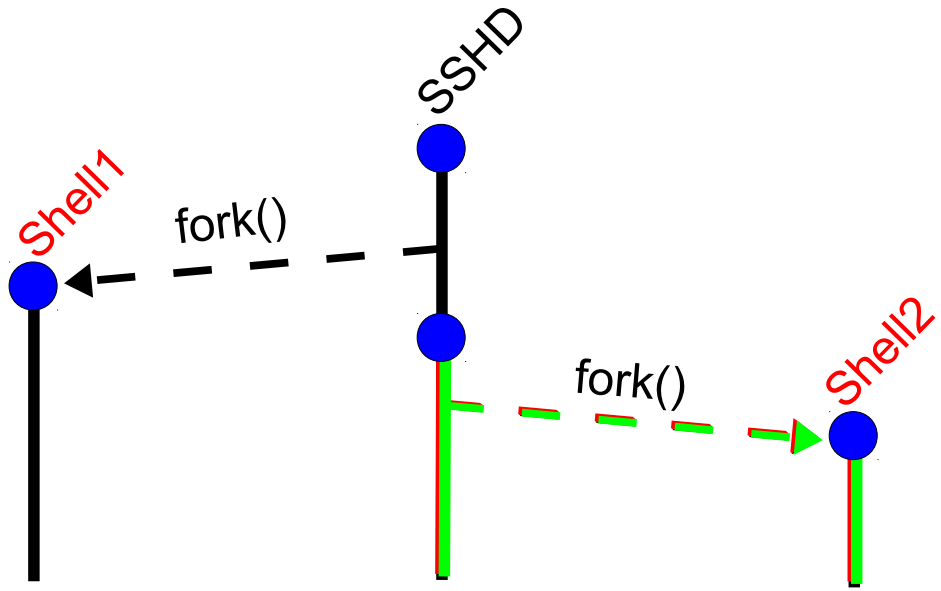
Send ***rollback(id)*** request to the receiver

Repair long-lived processes



- Repairing shell2 requires **re-execution** of shell1

Repair long-lived processes



- **Strawman**: process checkpoint
- **Problem**: poor performance
  - DMTCP (e.g., 0.6s w/ 4 MB log)
  - Linux-CR

Approach: mark *quiescent* state

- Long-lived processes (e.g., daemon)
  - Designed to be stateless

- Introduce *mark_quiescent()* syscall
  - Application needs modification to use the syscall
  - Re-running application rolls back state

Implementation

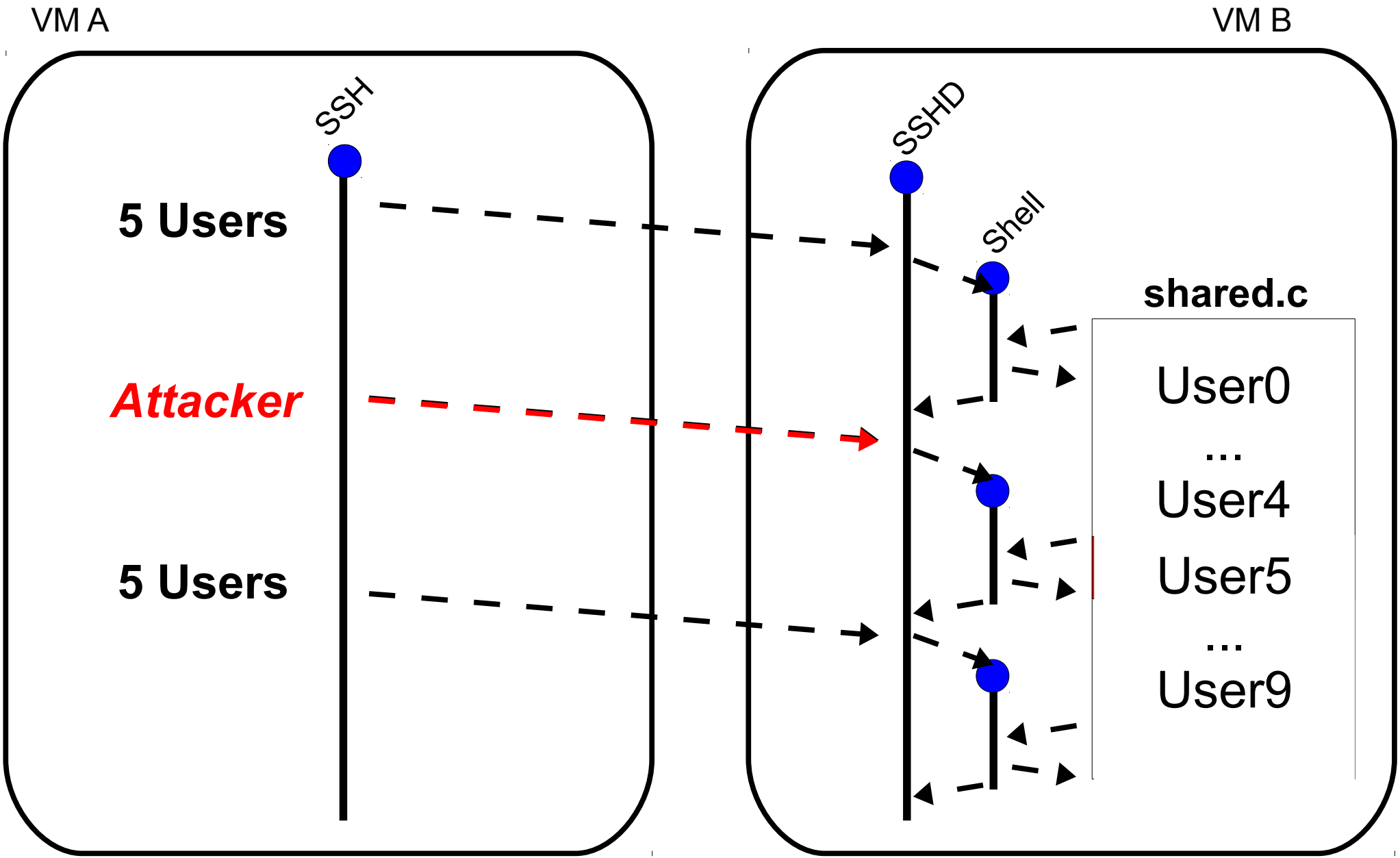
- Early prototype of DARE on Linux
  - Extend Retro's logger / repair controller
  - Add *mark_quiescent()* syscall
  - GUI Tools

Component	Lines of code
Logging kernel module	3,300 lines of C
AHG GUI Tool	2,000 lines of Python
Repair controller, managers	5,300 lines of Python
System library managers	800 lines of C

Evaluation

- Does it recover from a **synthetic** attack?
  - SSH attack with multiple users involved
- Does it **effectively minimize** re-execution?
  - *mark_quiescent()* works efficiently?

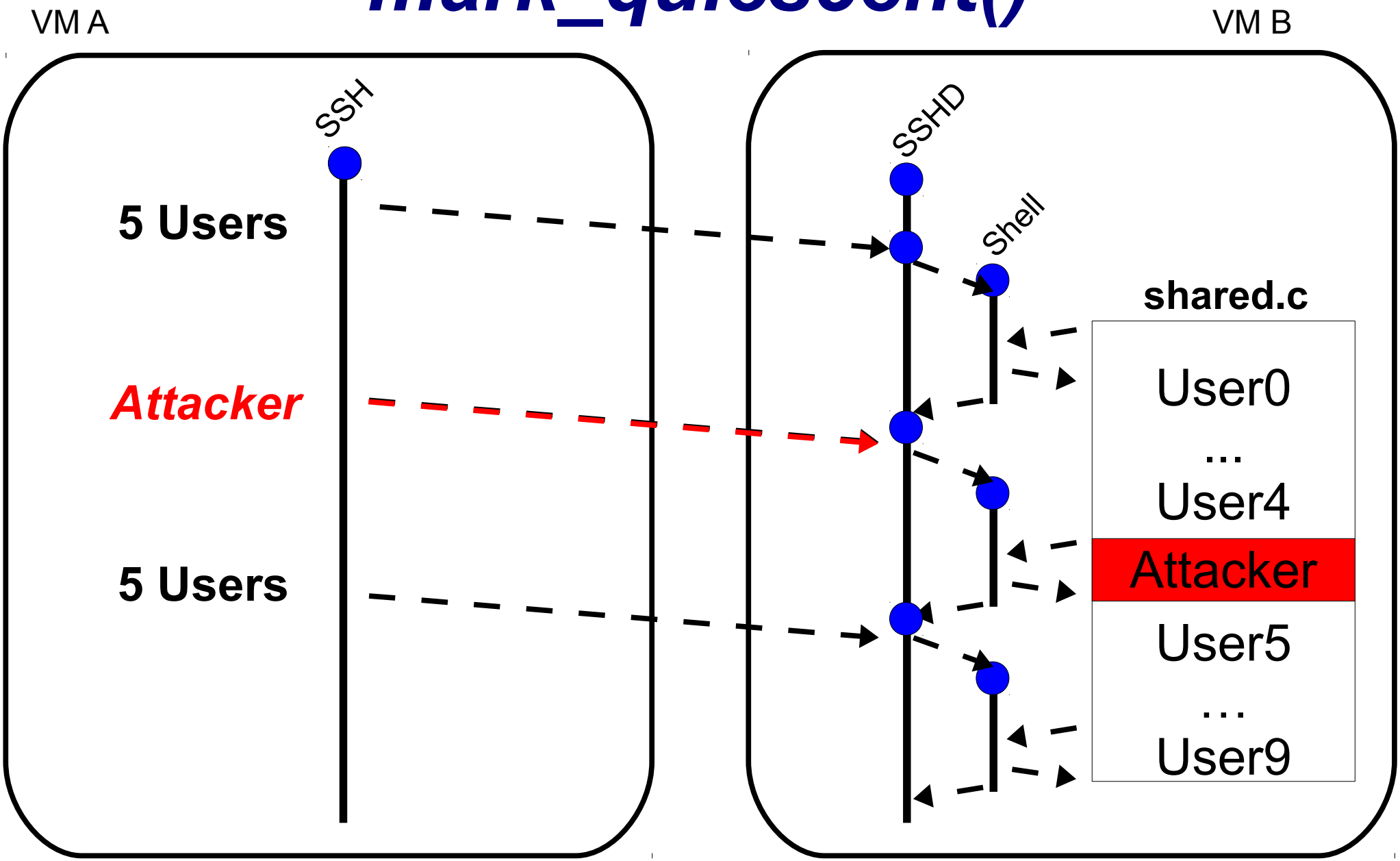
Experiment setup



Experiment results

- DARE **recovers a synthetic attack**
  - **8,953** objects in AHG (two VMs)
  - Restore the attack and rerun **10 legitimate users**

Experiment setup: using *mark_quiescent()*



Experiment results

- DARE **effectively minimizes** re-execution
  - Modify SSHD to use *mark_quiescent*()
  - Restore the attack and rerun **5 legitimate users**
  - Repair time: **3.7 s** → **0.44 s**

Open problems

- M**issing** dependencies
  - What if password or SSH key are stolen?
- Repair **across trust** domains
  - Who is allowed to undo an action?
  - How to trust undo requests?

Related work

- Record-and-reexecute:
  - **Retro**: initial design of repair controller, OS-level
  - **Warp**: retroactive patching, repairing web app
- Restoring network connections:
  - **DMTCP**: checkpoint and restore distributed processes
  - **Set/getsockopt**: TCP repair mode on Linux 3.5
- Detecting attacks in distributed systems
  - **Vigilante**: containment of internet worms
  - **Heat-ray**: preventing identity snowball attacks

Conclusion

- Efficient **recovery** mechanism in **distributed** systems using **selective re-execution**

- Three new techniques:
  - Record dependencies across multiple machines
  - Repair network connections
  - Repair long-lived processes