

Shadow-Bitcoin: Scalable Simulation via Direct Execution of Multi-threaded Applications

*Workshop on Cyber Security
Experimentation and Test*

August 10th, 2015



Andrew Miller, University of Maryland
amiller@cs.umd.edu

Rob Jansen, U.S. Naval Research Laboratory
rob.g.jansen@nrl.navy.mil

- [video removed for space reasons]

Goals of this Work

- **Directly execute Bitcoin** inside the Shadow network simulator
- Run a local and **private Bitcoin network**
- Explore **performance attacks on Bitcoin** using our simulation framework

Why should anyone care?

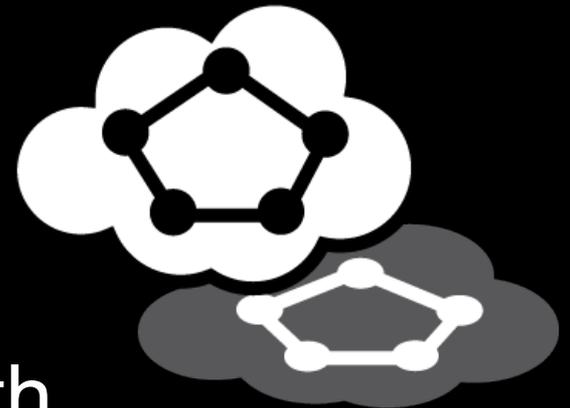
- **Expedite** research and development
- Evaluate software mods or attacks **without harming** real users
- Understand **holistic effects** before deployment
- Our techniques allow simulation support for many **new applications** and domains

Thread 1

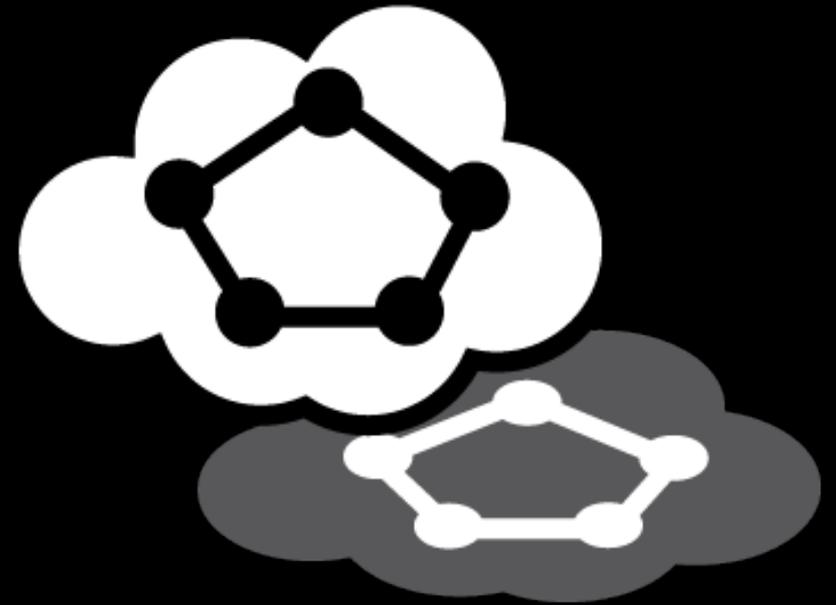
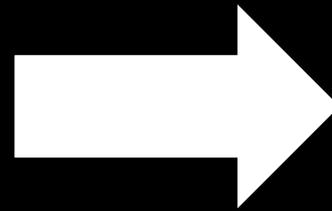
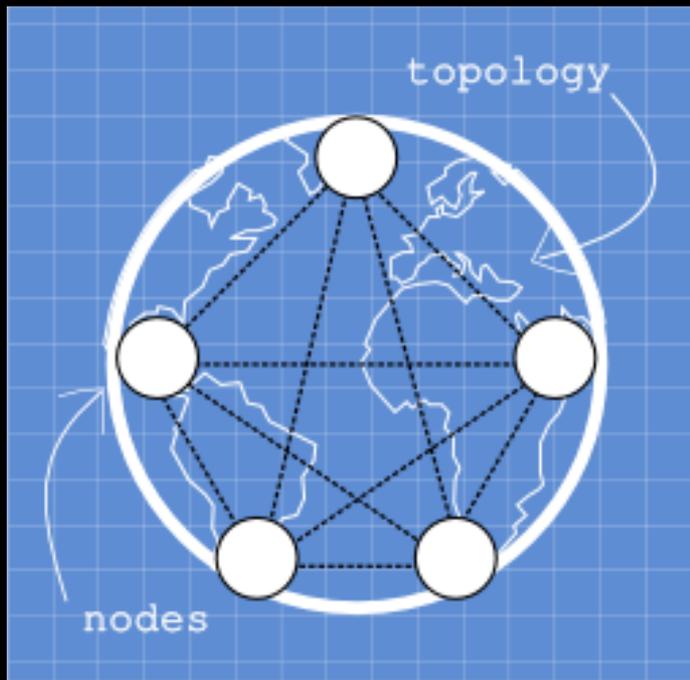
SHADOW BACKGROUND

What is Shadow?

- Parallel discrete-event network simulator
- Emulates POSIX C API on Linux, **directly executes** apps as plug-ins
- Simulates time, network, CPU
- Models routing, latency, bandwidth



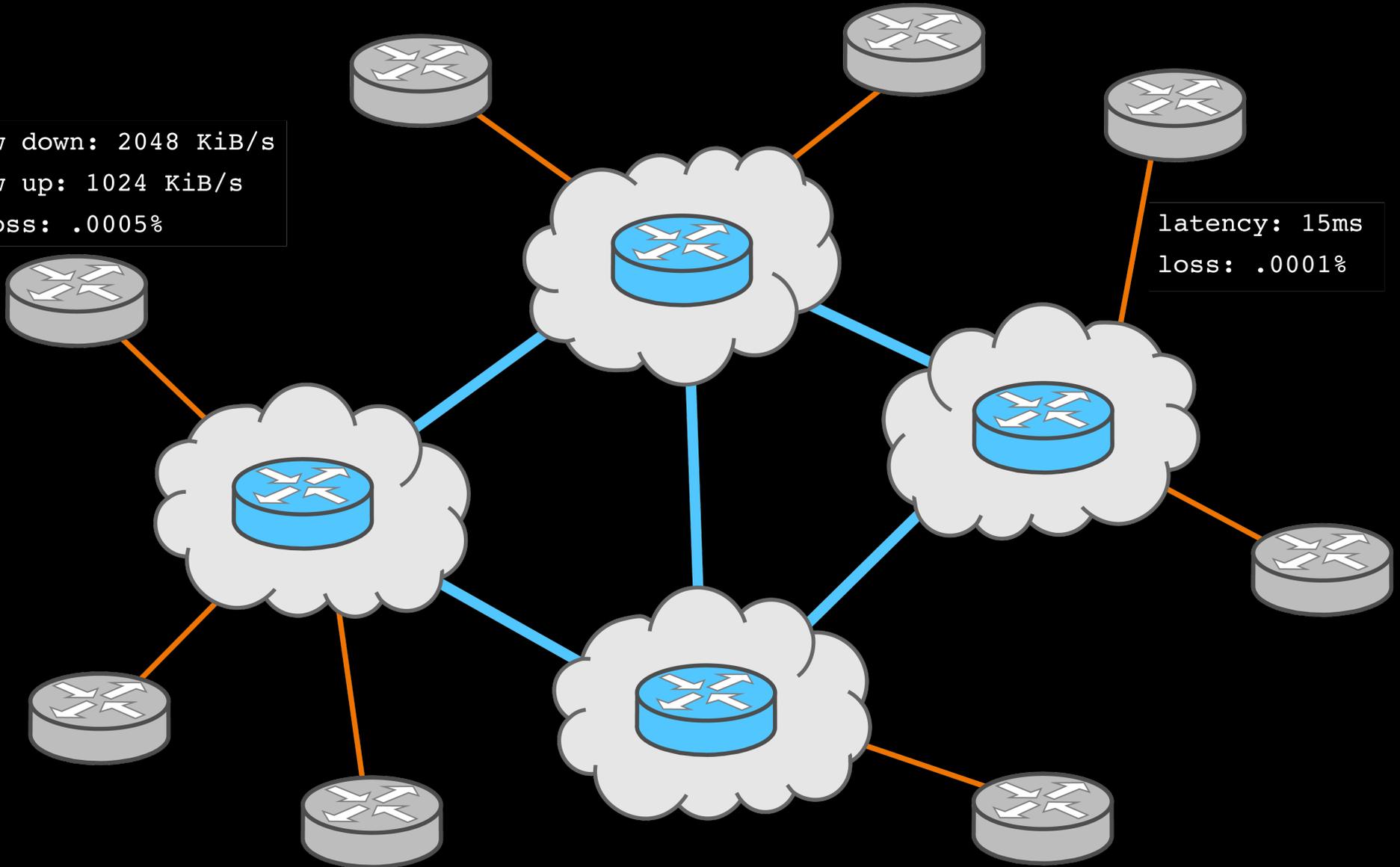
Bootstrapping Shadow



Virtual Network Configuration

bw down: 2048 KiB/s
bw up: 1024 KiB/s
loss: .0005%

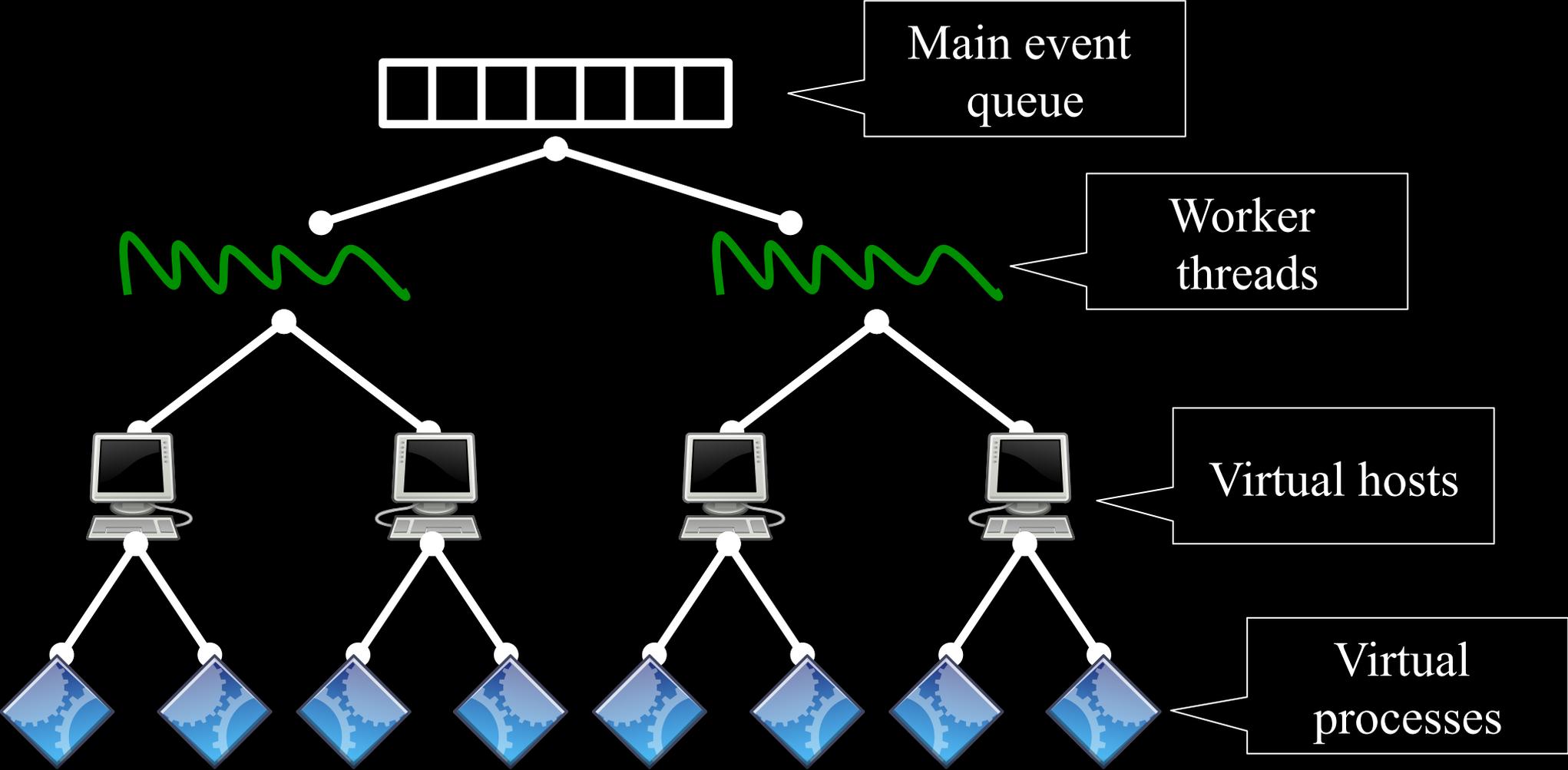
latency: 15ms
loss: .0001%



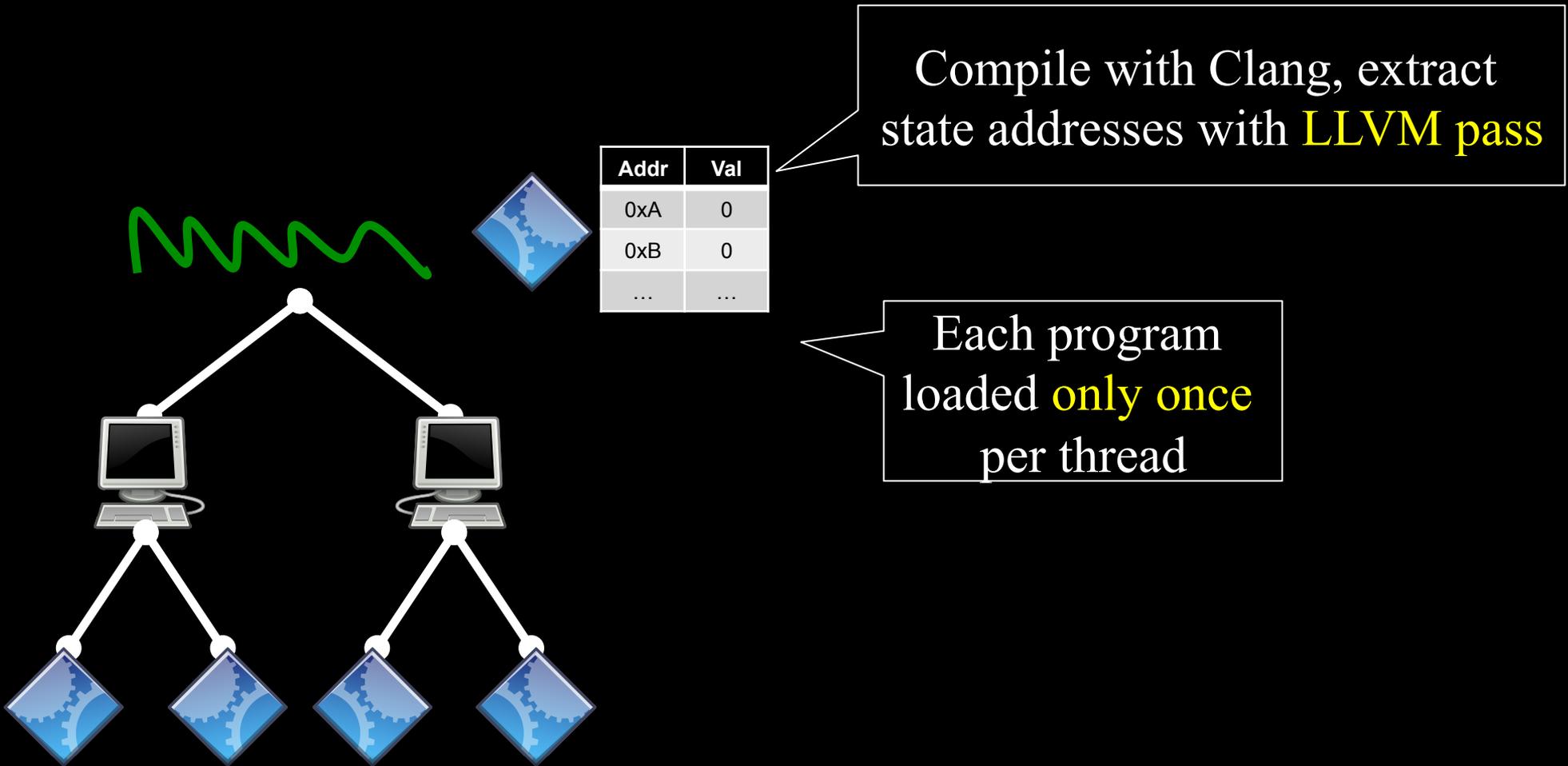
Virtual Host Configuration



Simulation Engine



Simulation Engine



Function Interposition

LD_PRELOAD=/home/rob/libpreload.so

libpreload (*socket, write, ...*)

Shadow
Engine

App
Plug-in

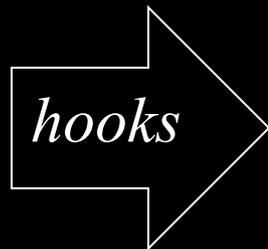
App
Libraries
(libc, ...)

Function Interposition

LD_PRELOAD=/home/rob/libpreload.so

libpreload (*socket, write, ...*)

Shadow
Engine



App
Plug-in

App
Libraries
(libc, ...)

Function Interposition

LD_PRELOAD=/home/rob/libpreload.so

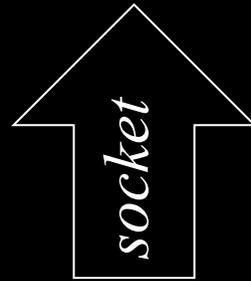
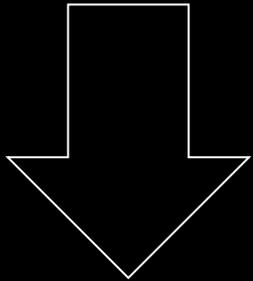
libpreload (*socket, write, ...*)



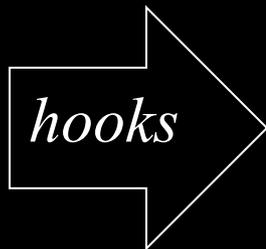
Function Interposition

LD_PRELOAD=/home/rob/libpreload.so

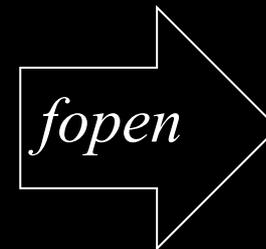
libpreload (*socket*, *write*, ...)



Shadow
Engine



App
Plug-in

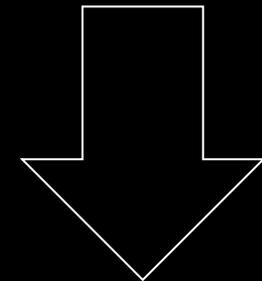
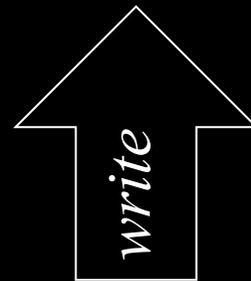


App
Libraries
(libc, ...)

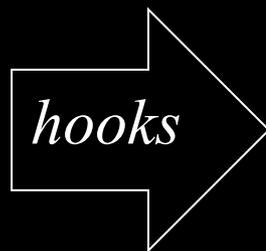
Function Interposition

LD_PRELOAD=/home/rob/libpreload.so

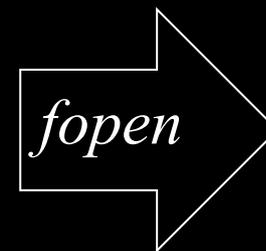
libpreload (*socket, write, ...*)



Shadow
Engine



App
Plug-in



App
Libraries
(libc, ...)

Function Interposition

LD_PRELOAD=/home/rob/libpreload.so

libpreload (*socket, write, ...*)

Single call stack,
must return

write

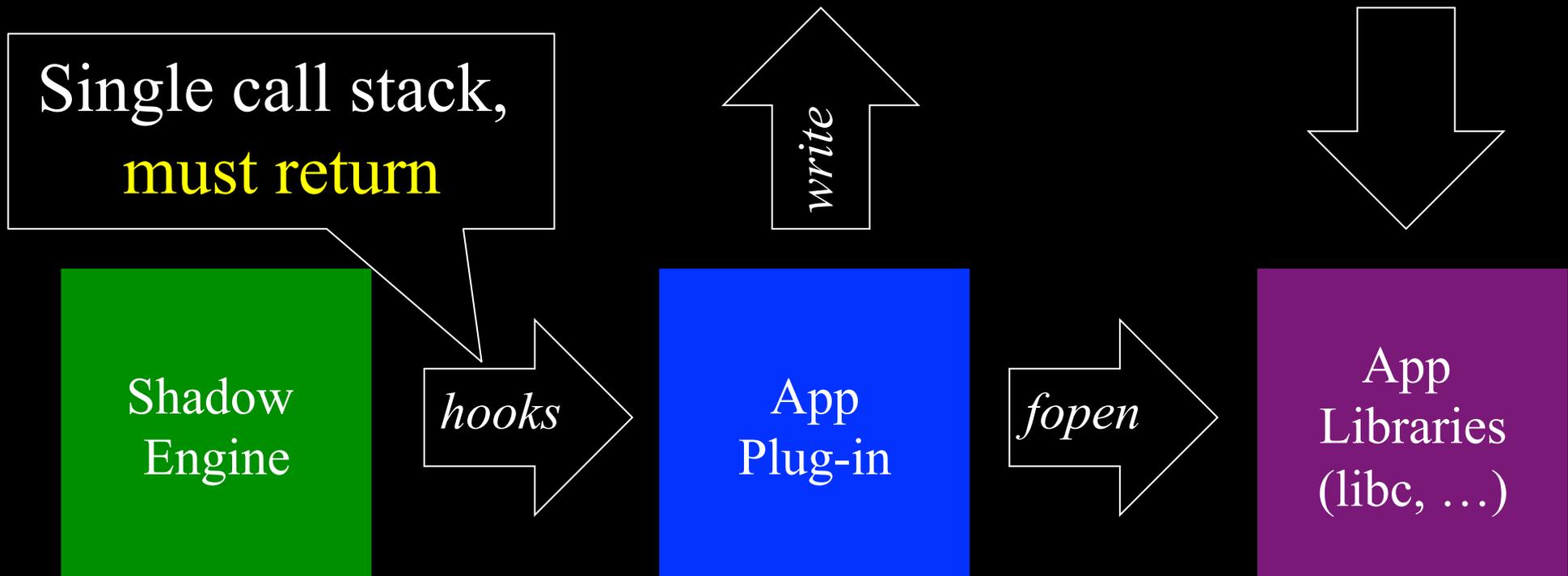
Shadow
Engine

hooks

App
Plug-in

fopen

App
Libraries
(libc, ...)



Shadow limitations

- App **shall not block**
 - Call any blocking library function (sleep)
 - Use blocking descriptors (read/write, send/rcv)
 - Wait for events (select, poll)
 - Busy wait (infinite loop)

Shadow limitations

- App **shall not block**
 - Call any blocking library function (sleep)
 - Use blocking descriptors (read/write, send/recv)
 - Wait for events (select, poll)
 - Busy wait (infinite loop)
- App **shall not spawn**
 - Multiple threads (pthreads)
 - Multiple processes (fork, exec)

Shadow limitations

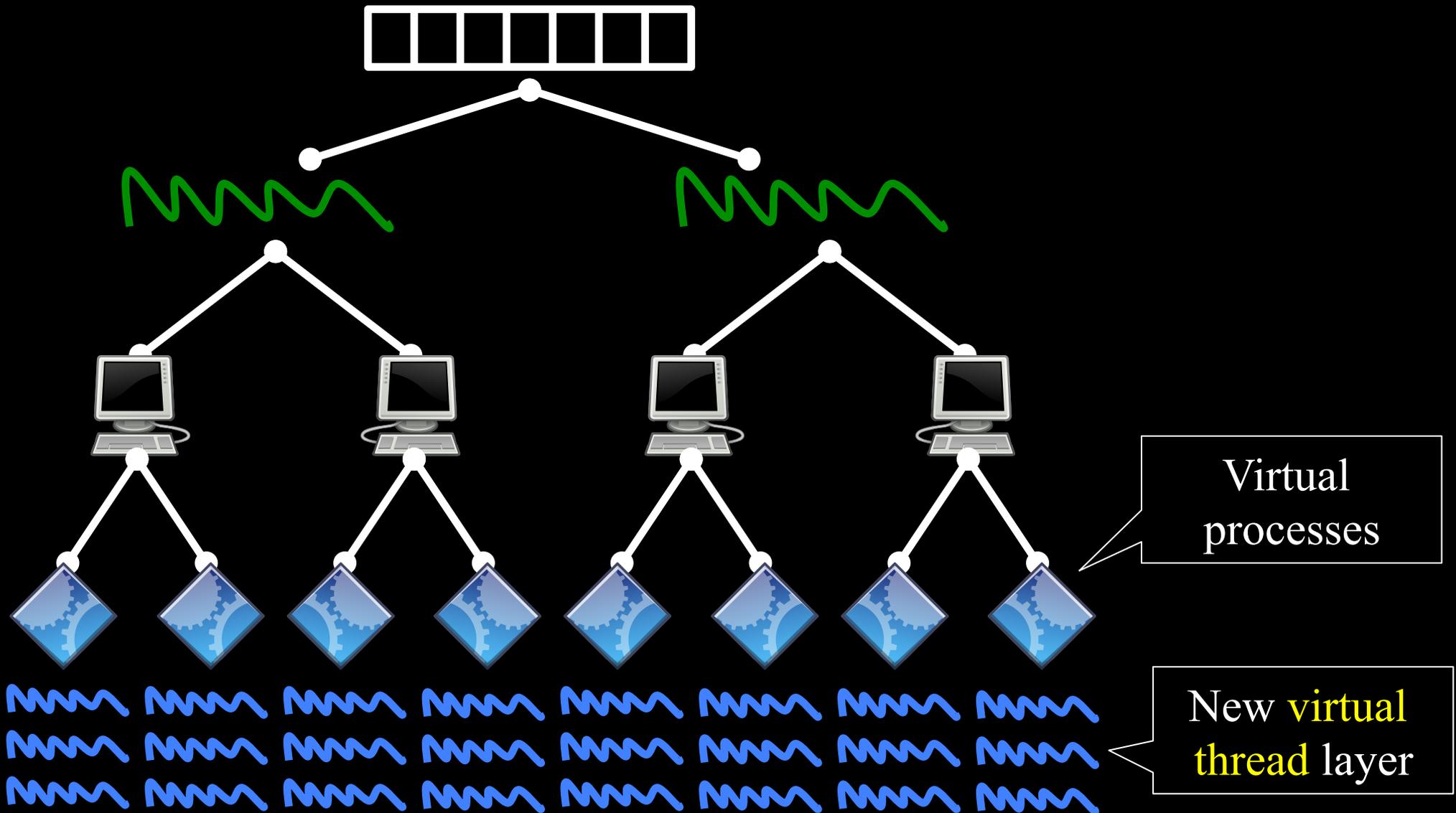
- App **shall not block**
 - Call any blocking library function (sleep)
 - Use blocking descriptors (read/write, send/recv)
 - Wait for events (select, poll)
 - Busy wait (infinite loop)
- App **shall not spawn**
 - Multiple threads (pthreads)
 - Multiple processes (fork, exec)

Problems!
Bitcoin blocks
and spawns
threads! ☹️

Thread 2

RUNNING BITCOIN IN SHADOW

Architectural Update



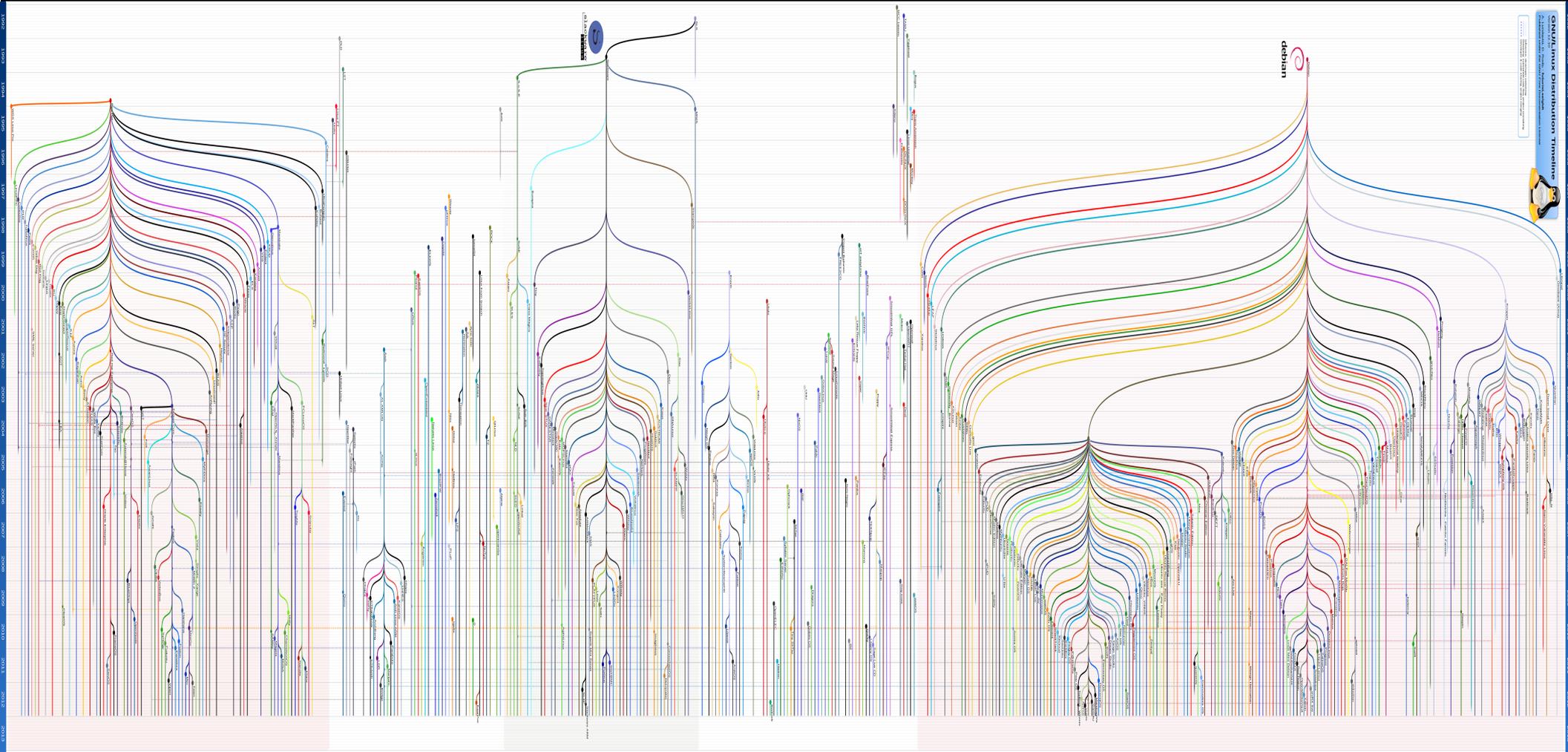
Non-blocking Virtual Threads

- GNU **portable threads** (pth) to the rescue
 - User-land **cooperative** threading (non-preemptive)
 - Single OS thread, multiple portable threads, **supports pthread** API
 - Supports many blocking IO functions: uses **make/set/get/swapcontext()** magic to **jump program stacks**

Limitations of GNU pth

- Not reentrant or thread-safe
- Relies on `select()` to poll events when all portable threads would block (max 1024 fds)

If you don't like it, fork it



Reentrant Portable Threads (rpth)

- Not reentrant or thread-safe
- Relies on `select()` to poll events when all portable threads would block (max 1024 fds)

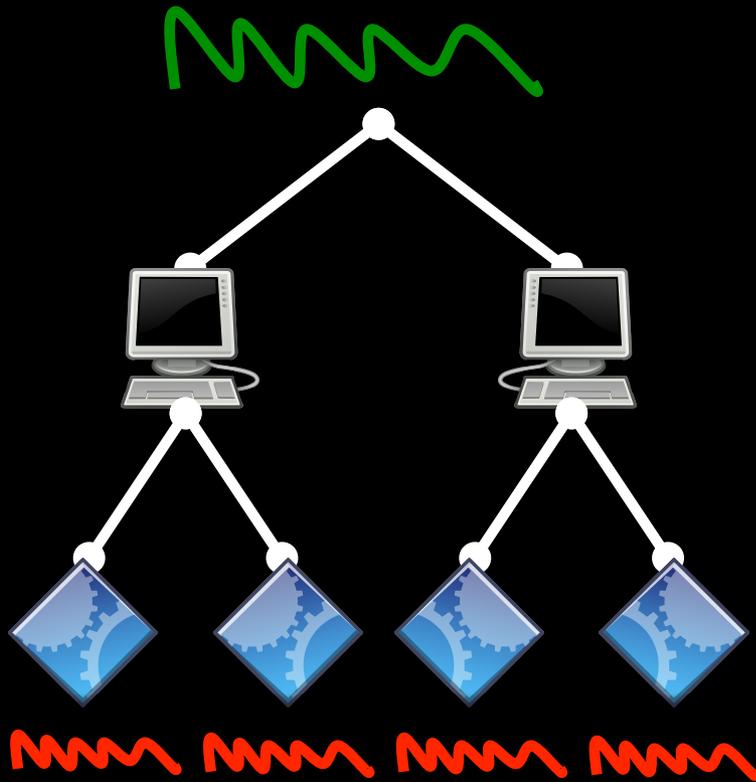
Reentrant Portable Threads (rpth)

- ~~Not reentrant or thread safe~~
 - Replace global state with **user-supplied states**
 - **Thread-local storage** for current state pointer
- Relies on select() to poll events when all portable threads would block (max 1024 fds)

Reentrant Portable Threads (rpth)

- ~~Not reentrant or thread safe~~
 - Replace global state with **user-supplied states**
 - **Thread-local storage** for current state pointer
- ~~Relies on select() to poll events when all portable threads would block (max 1024 fds)~~
 - Replace **blocking select** with **asynchronous epoll**
 - Add API support for epoll and timers

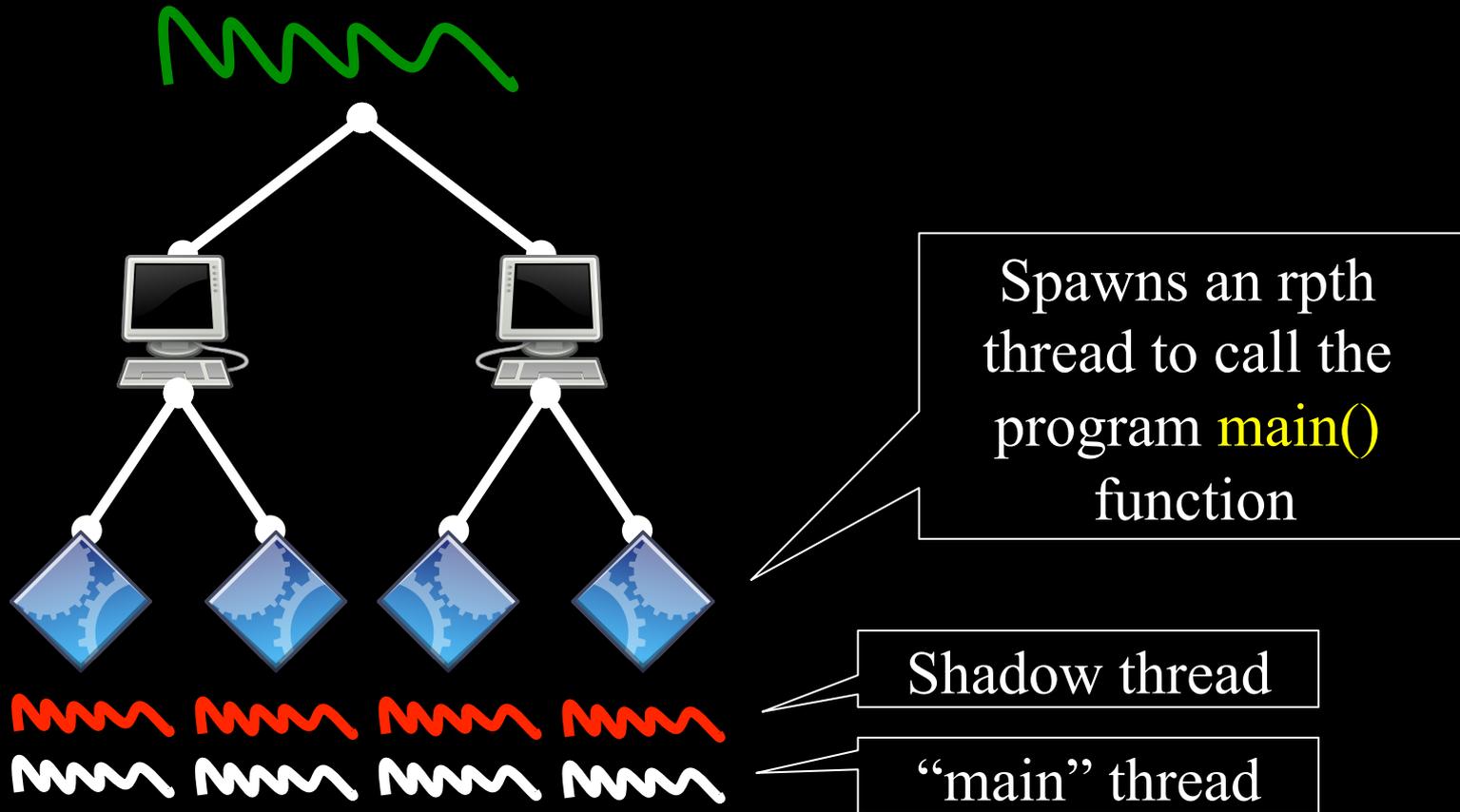
Integrating rpth with Shadow



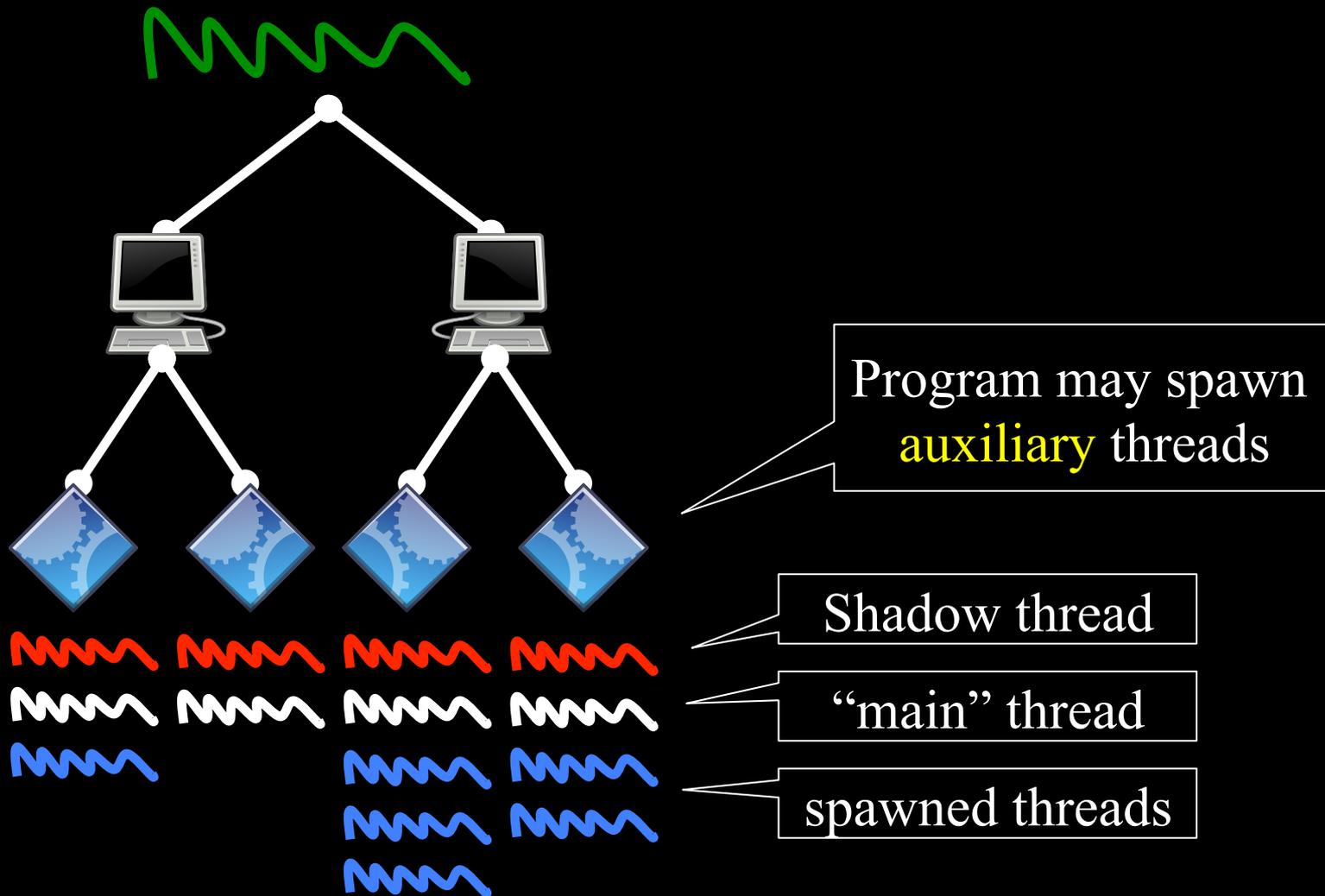
Each virtual process has a private **rpth instance**

Shadow thread

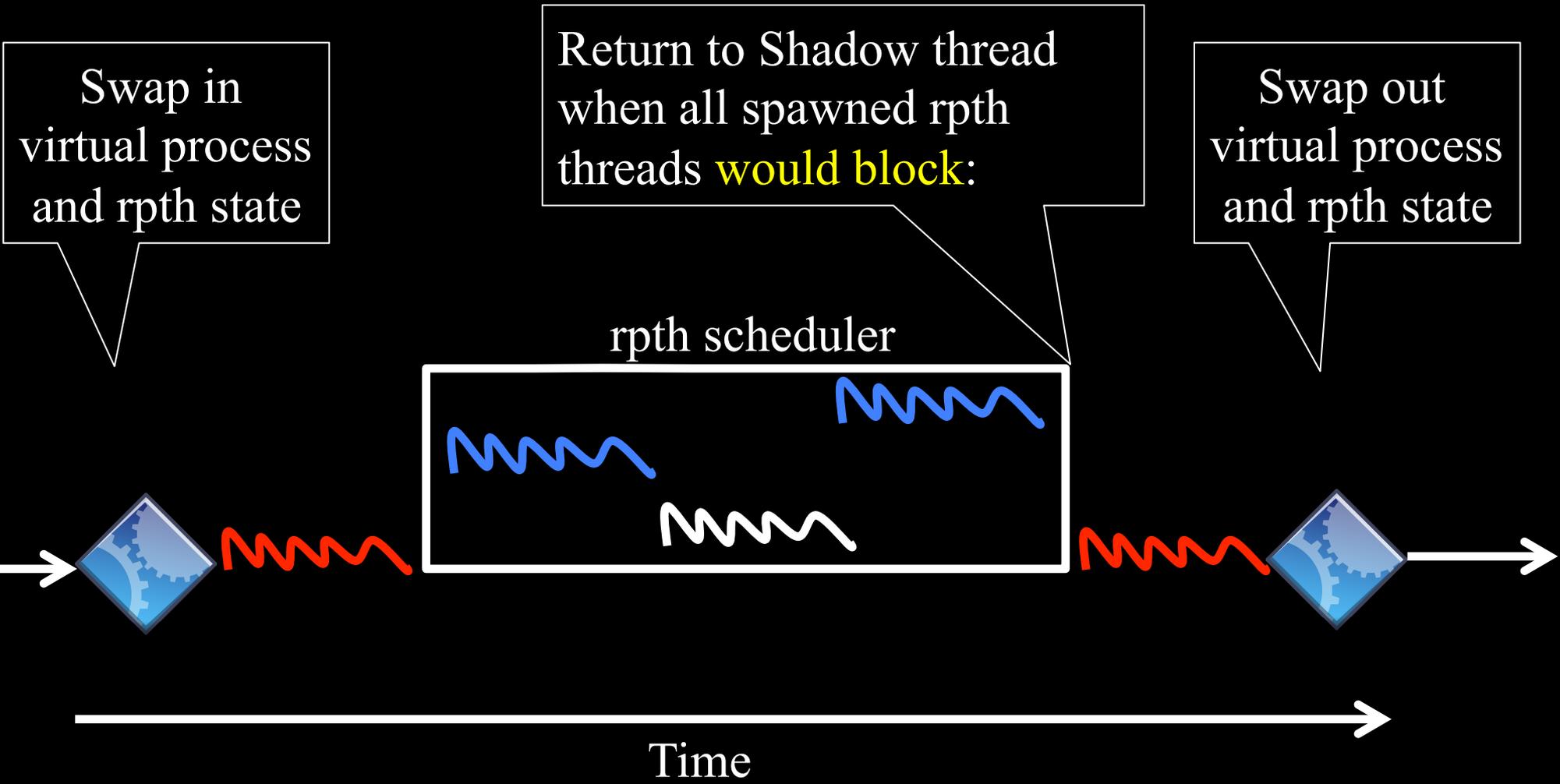
Integrating rpth with Shadow



Integrating rpth with Shadow



Execution Flow with rpth



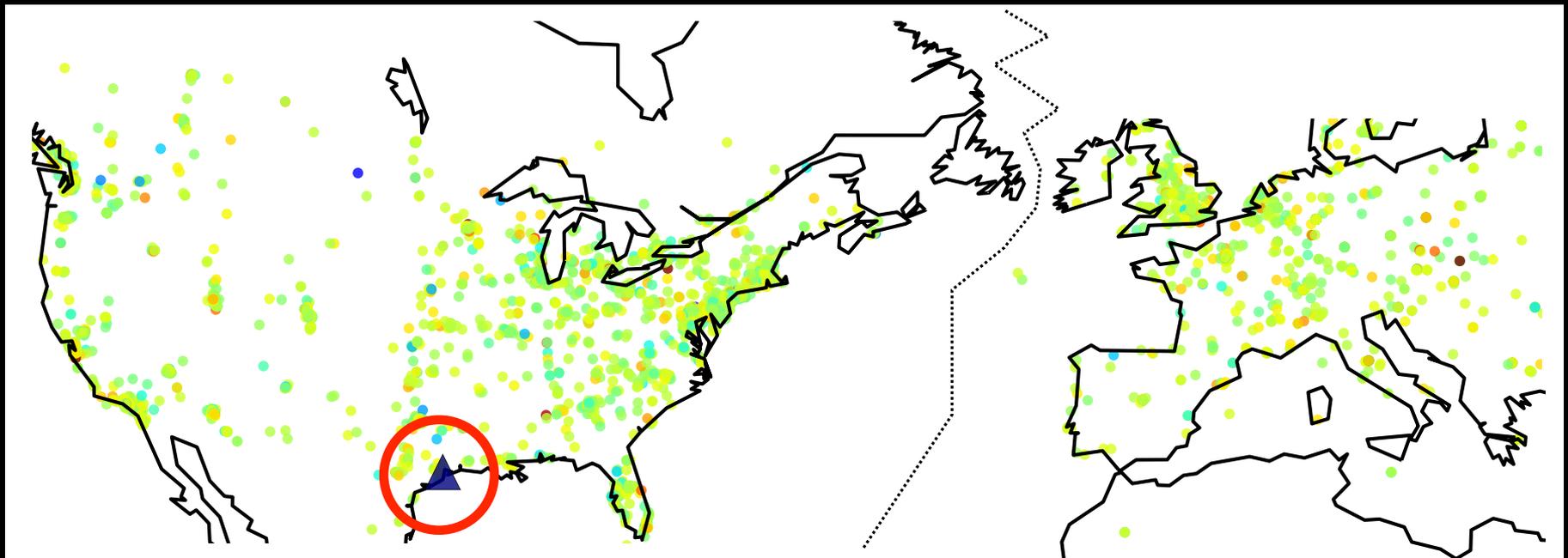
Creating a Private Bitcoin Network

- Crawled Bitcoin with CoinScope to **learn topology** – 6081 nodes (40% US, 40% EU)
- **Geo-locate nodes** based on IP address
- **Bootstrap blockchain** – Bitcoin block and index files are COW – enables aliasing of these large state files
- Inject new transactions to each node to **simulate spending**

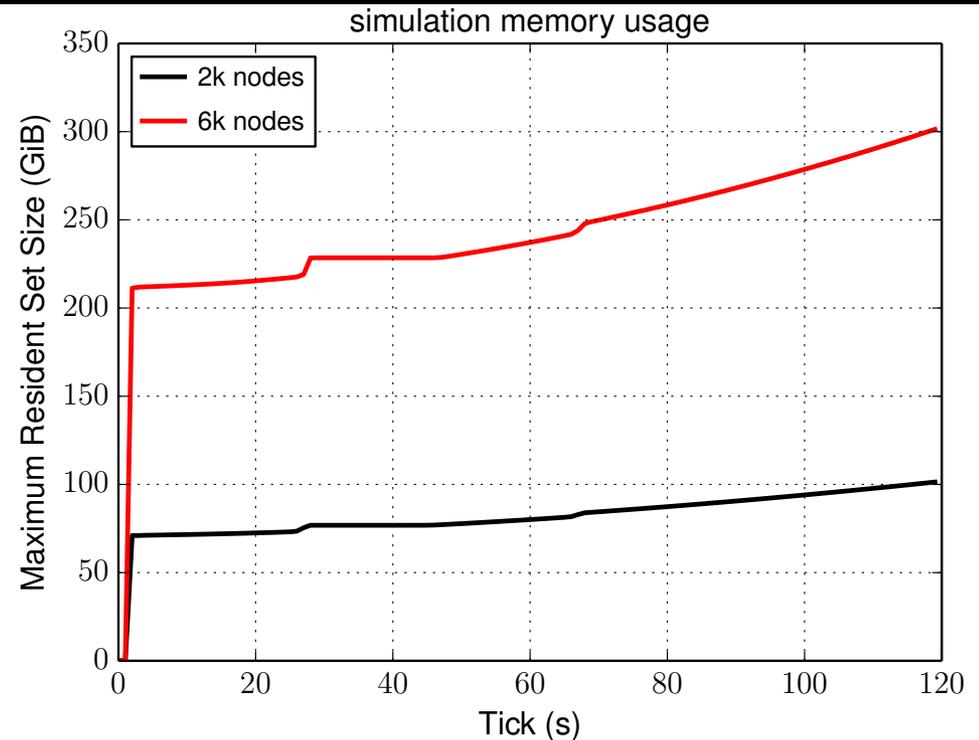
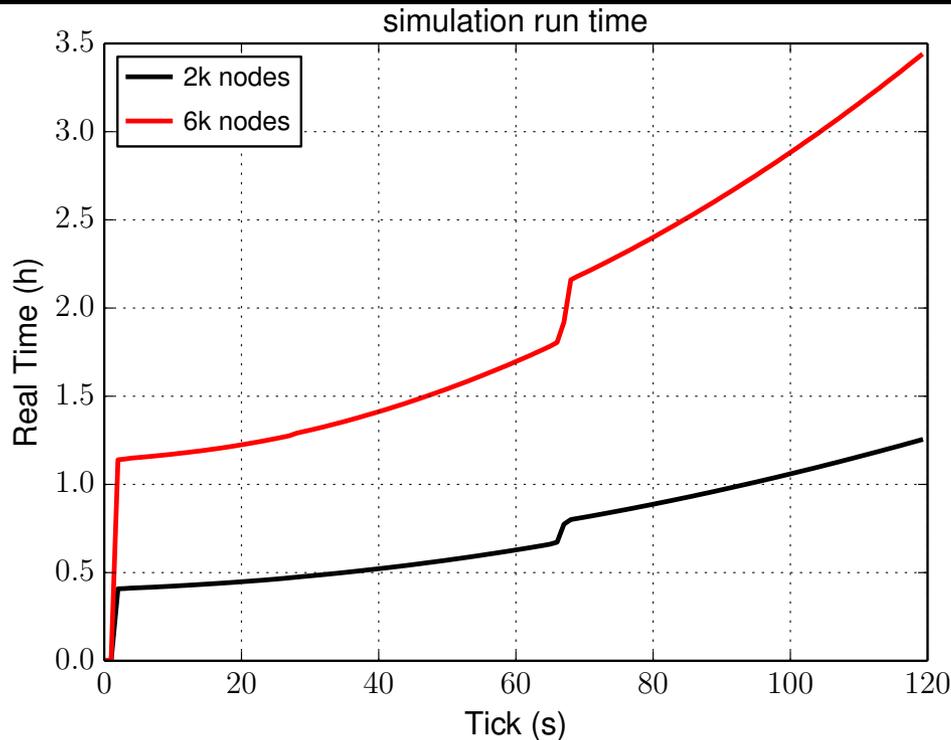
Transaction Propagation

Faster

Slower



Simulation Resource Usage



For each node:
~2.1 seconds to run 120 ticks
(~57x speedup)

For each node:
~51.2 MiB consumed

Thread 3

ATTACKING BITCOIN

Transaction Handling

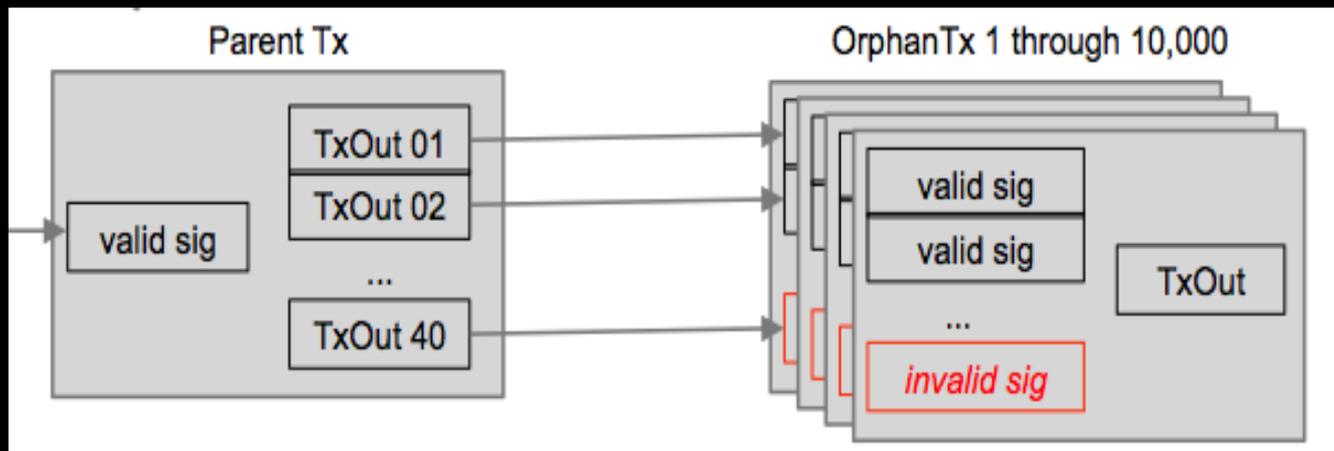
- Transactions form a **directed** graph
 - Tx with parent gets handled immediately
 - Validate Tx, verify up to 40 sigs
 - Senders of invalid Txs are marked as bad, and eventually disconnected

Transaction Handling

- Transactions form a **directed** graph
 - Tx with parent gets handled immediately
 - Validate Tx, verify up to 40 sigs
 - Senders of invalid Tx's are marked as bad, and eventually disconnected
- What if Tx has no parent?
 - Tx w/o parent gets queued as **orphan**
 - Once queued, sender of orphan is forgotten
 - When new Tx arrives, all linked orphans are validated (40 sig verifications each)

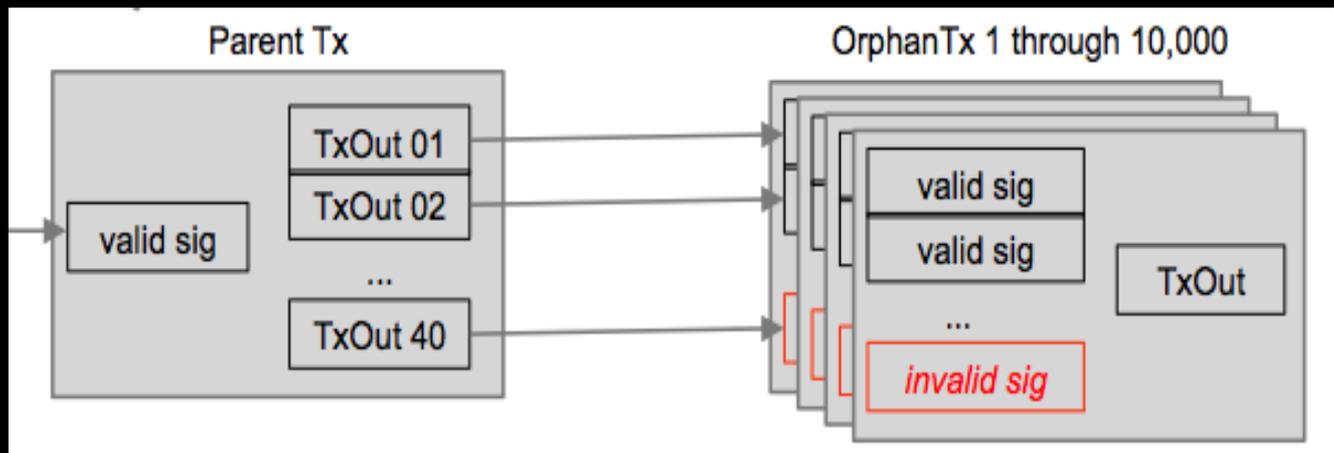
Dos Attack

- Goal: **Freeze a victim node**
 - Fill up orphans queue with **invalid** Tx's
 - Send valid parents with outputs linked to orphans
 - Node checks **all orphans**



Dos Attack

- Goal: **Freeze a victim node**
 - Fill up orphans queue with **invalid** Tx's
 - Send valid parents with outputs linked to orphans
 - Node checks **all orphans**



40 sigs/orphan,
10k orphans max,
0.6ms per sig

Freeze for 4+ mins,
Peers will abort,
No one to blame

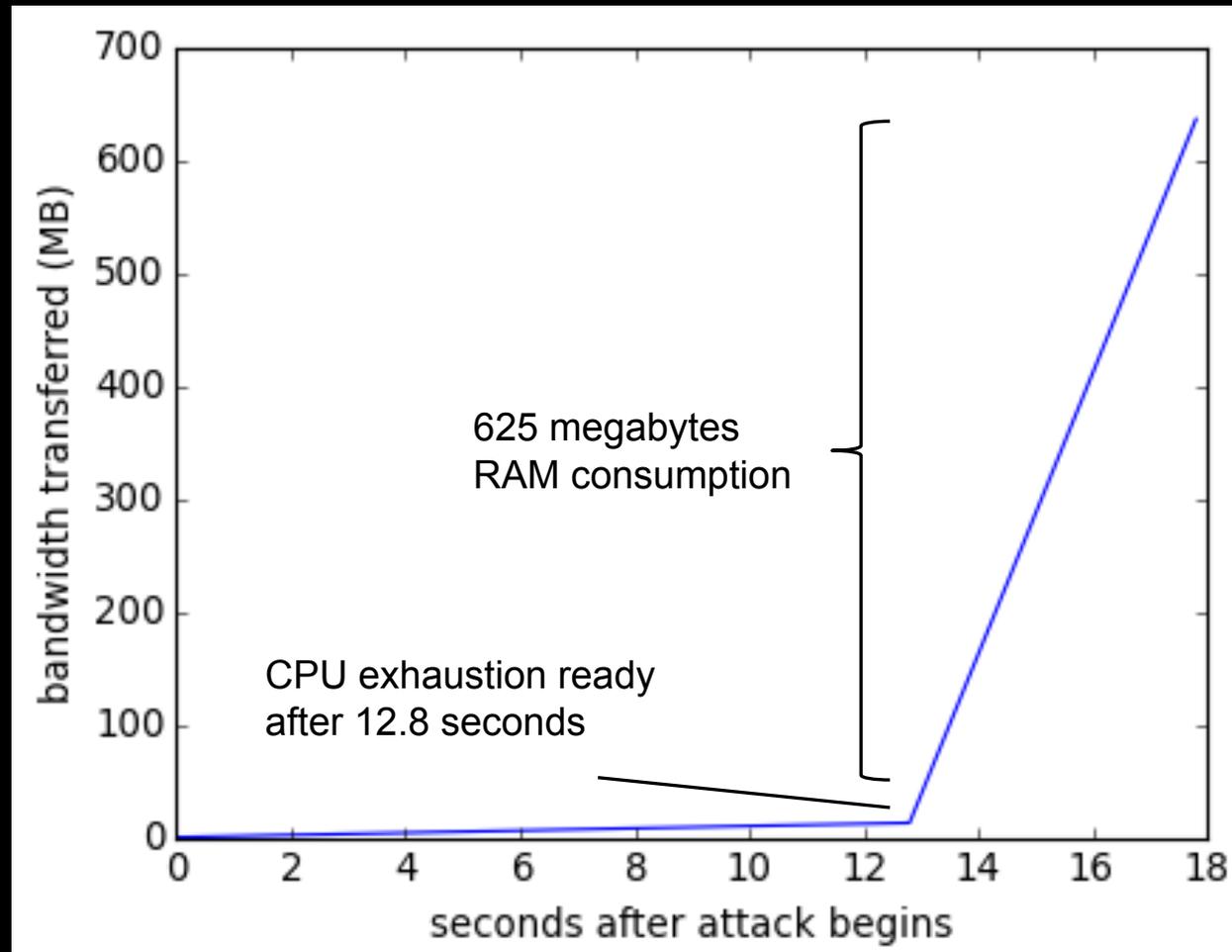
RAM Consumption

- While **MessageHandler** thread is frozen, **SocketHandler** thread buffers peer data
- Disconnect peer if $|\text{recvBuf}| > 5 \text{ MiB}$

RAM Consumption

- While **MessageHandler** thread is frozen, **SocketHandler** thread buffers peer data
- Disconnect peer if $|\text{recvBuf}| > 5 \text{ MiB}$
- Attack
 - Establish 100+ connections to victim
 - While victim is frozen, fill `recvBuf` to max
 - Can crash node if $< 500 \text{ MiB}$ available

Attack Time and Cost Profile



Fix Applied to Bitcoin

Fixed in commit 0608780



bitcoin / bitcoin

Stricter handling of orphan transactions

Prevent denial-of-service attacks by banning peers that send us invalid orphan transactions and only storing orphan transactions given to us by a peer while the peer is connected.

 master (#4885)  v0.11.0rc3 ... v0.10.0



gavinandresen authored on Aug 28, 2014



Showing **2 changed files** with **65 additions** and **17 deletions**.

Summary/Conclusion

- Enhanced Shadow to support applications that **block** and use **multiple threads**
- Wrote new **Bitcoin plug-in** for Shadow
- Created **Bitcoin network** for simulation
- Found and fixed **orphans attack** using new simulator architecture

shadow.github.io github.com/shadow	robgjansen.com, @robgjansen rob.g.jansen@nrl.navy.mil
---------------------------------------	--

think like an adversary

