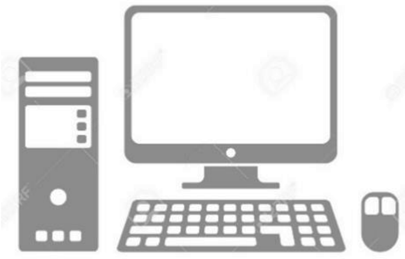
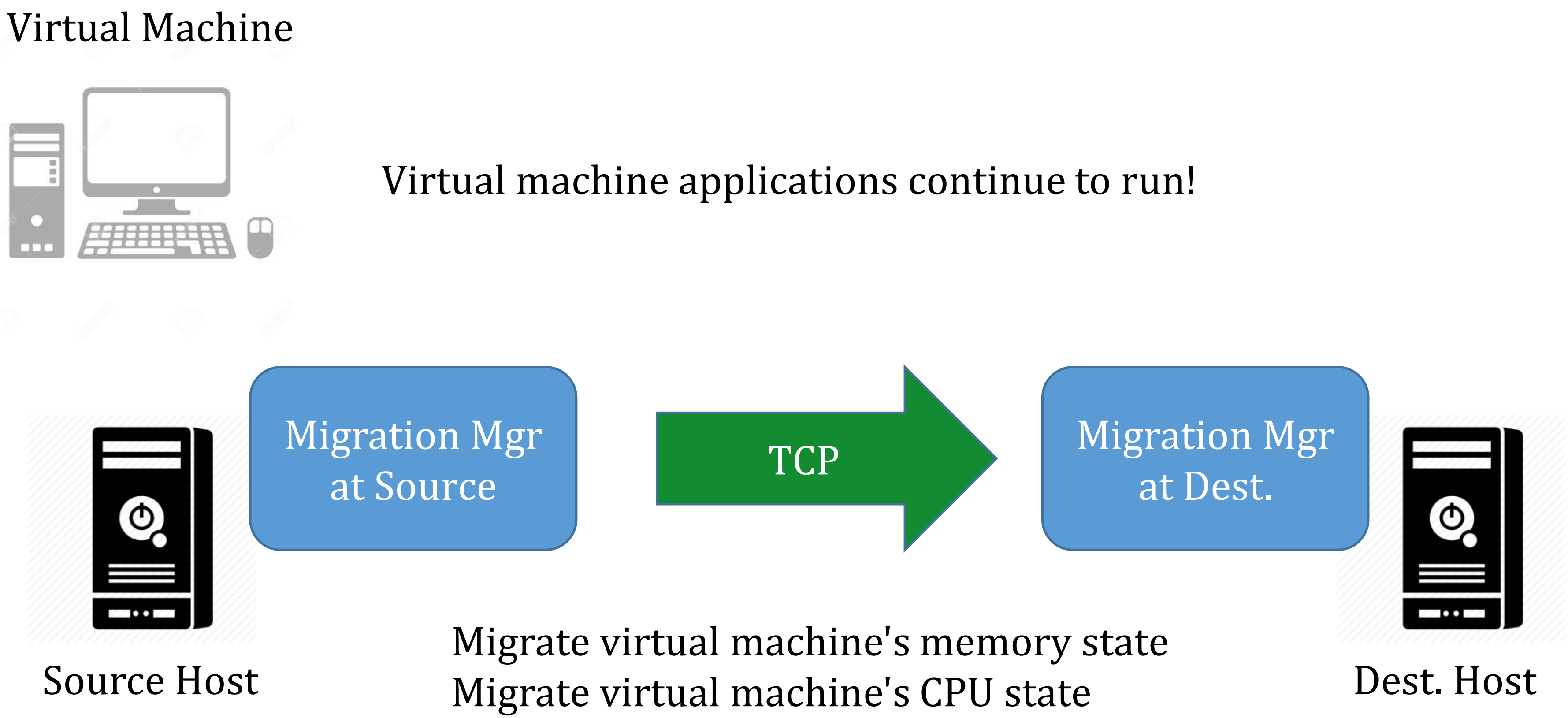
Live Migration
of
Virtual Machines



Live Migration of Virtual Machines, Christopher Clarke, Keir Fraser, et. al. NSDI 2005

Post-copy live migration of virtual machines, Hines, Deshpande, Gopalan, VEE 2009

What is live VM migration?



Virtual Machine

Virtual machine applications continue to run!

Migration Mgr at Source

TCP

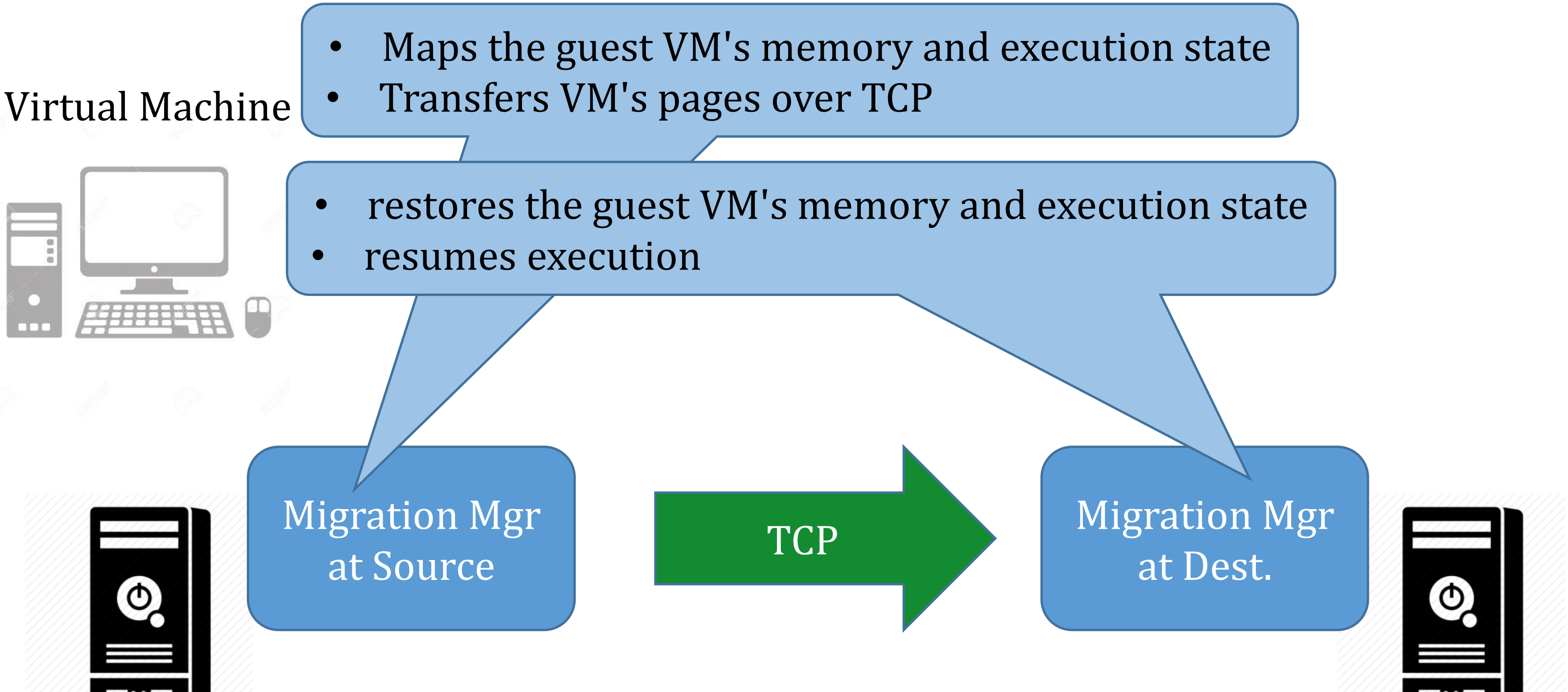
Migration Mgr at Dest.

Source Host

Dest. Host

Migrate virtual machine's memory state
Migrate virtual machine's CPU state
Optionally, migrate virtual disk state

What is live VM migration?



- Maps the guest VM's memory and execution state
- Transfers VM's pages over TCP

- restores the guest VM's memory and execution state
- resumes execution

<u>Examples</u>

xend for Xen

QEMU for KVM

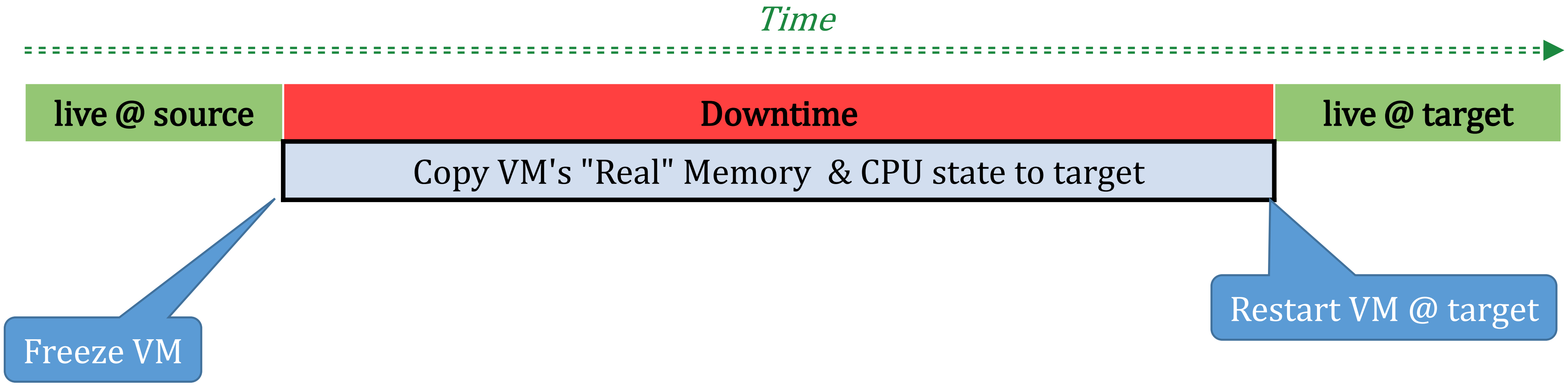
Why Live VM Migration?

- Why Migrate?
  - Load balancing – move VMs from highly loaded to lightly loaded severs
  - Server Maintenance – When servers need to be upgraded
  - Energy Savings – Move load off to shut down server and save energy
- Why Live? To avoid disruption of VM users
  - To save investment in long running jobs
  - To keep network connections alive
- Why VM? (Why not migrate processes?)
  - Process migration leaves residual dependencies at source host
    - system call redirection, shared memory, open files, IPC, etc.

Performance Goals in Live Migration

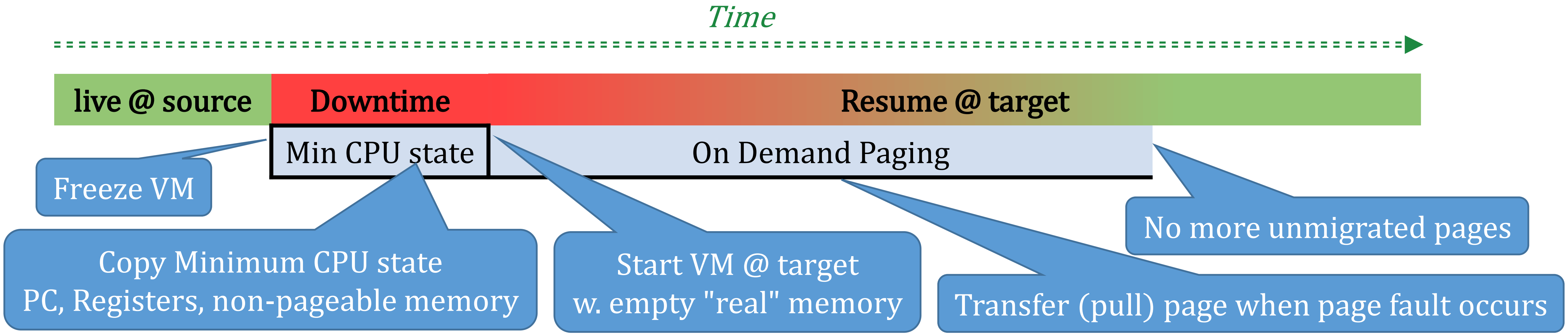
- Minimize downtime
  - Time from VM stop to VM restart
- Reduce total migration time
  - Time from migration start to migration stop
- Avoid interference with normal system activity
  - E.g. network bandwidth
- Minimize network activity
- Maximize Reliability
  - If migration fails, can the VM continue at source or target?

Stop-and-Copy Migration



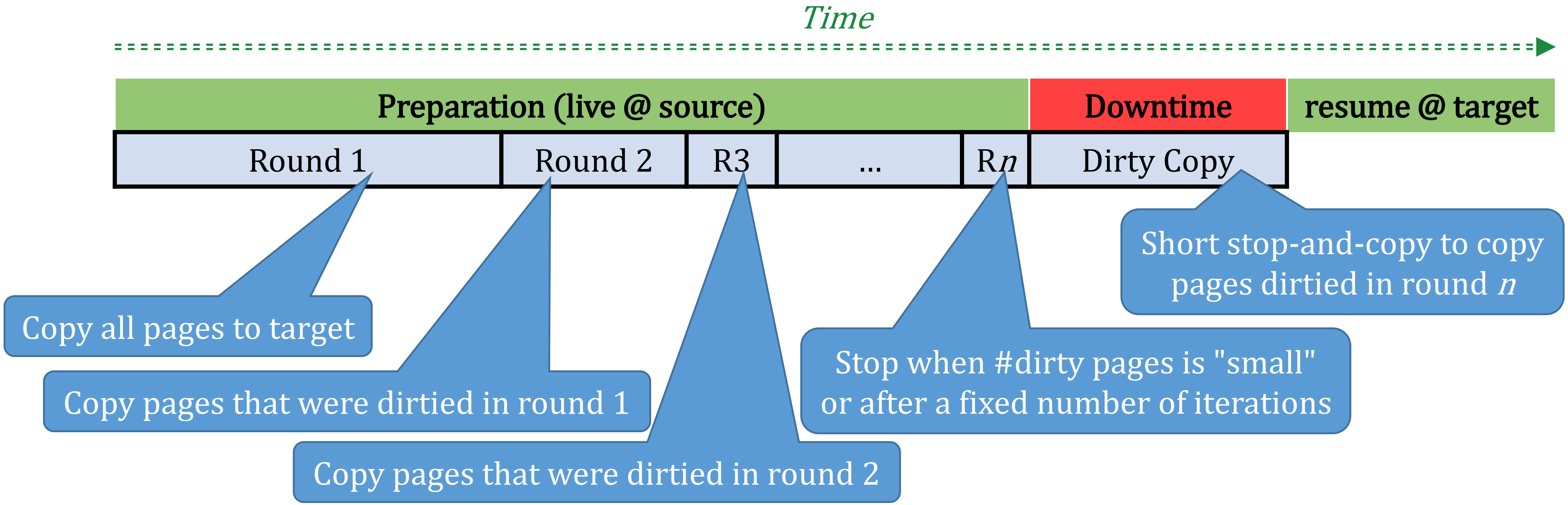
- Looong downtime!
- Relative short migration time = downtime
- Manage TCP bandwidth to trade network impact vs. downtime
- If migration fails, source is still correct, VM can continue

Pure Demand Paging Migration



- Very short "downtime"
- Slooow warm-up – page faults over network!
- Target migration manager must track pages –
  - Unused vs. used@source vs. used@target
- Very long, unpredictable migration time
- If migration fails both source and target are incorrect

Pre-copy Migration


Time
Preparation (live @ source)
Downtime
resume @ target
Round 1
Round 2
R3
...
R*n*
Dirty Copy
Copy all pages to target
Copy pages that were dirtied in round 1
Copy pages that were dirtied in round 2
Stop when #dirty pages is "small" or after a fixed number of iterations
Short stop-and-copy to copy pages dirtied in round *n*

- Very short downtime (close to pure demand paging)
- No slooow warm-up
- Requires extra network resources (Round 2-n are re-copying pages!)
- Long migration time – predictable?
- If migration fails, source is up to date, VM can be recovered

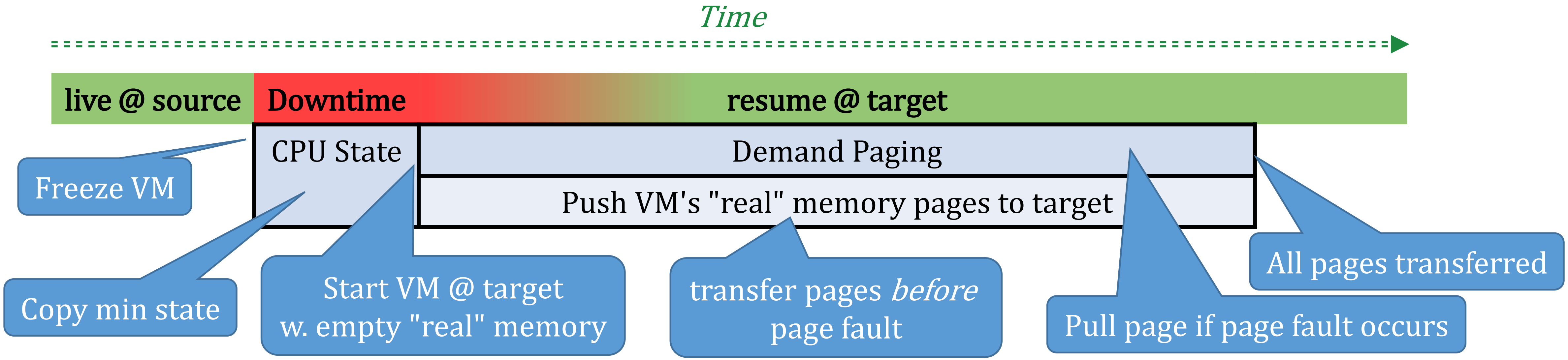
How do we track dirtied pages?

- Mark all VM's memory pages as Read Only after each iteration
- Trap write operations via hypervisor
  - Hypervisor dispatches writes to source migration manager
  - Source migration manager updates its "dirty" bits for pages, enables RW on the page, and re-dispatches the write
- At the end of an iteration, migration manager creates new "dirty" bits for the next iteration, and uses old "dirty" bits for copies
- Overhead: Trap each write instruction during migration
- Works well as long as writes are infrequent

Optimizations

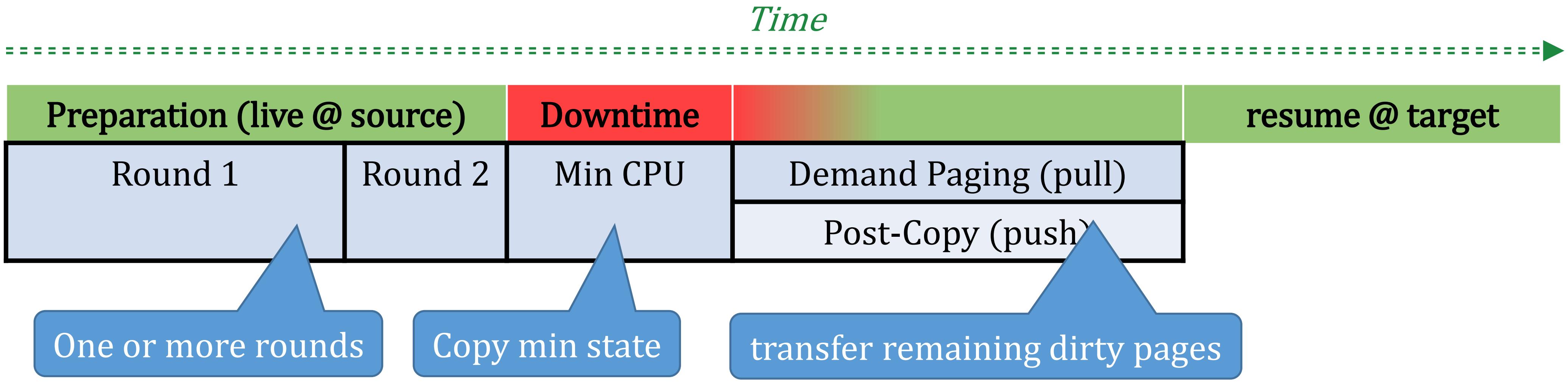
- Problem: Heavy TCP usage during migration impacts running services
  - Solution: Limit bandwidth used by migration (makes each iteration longer, therefore bigger)
- Problem: Page dirtied after iteration ends, but before page transferred
  - Solution: If the page is dirty in the NEXT round, don't transfer it THIS round
- Problem: Rogue processes don't stop dirtying memory
  - Solution: Identify and "stun" these rogue processes
- Problem: Unused pages in VM's real memory copied to target
  - Solution: Only transfer pages marked as "used" in the VM's page tables
  - If page gets re-used, page fault penalty at target

Post-copy Migration



- Very short "downtime" (close to pure demand paging)
- Avoid most slooow warm-up – most pages pushed BEFORE they are demand paged
  - Still pay cold start penalty at target
- Predictable (short) migration time
- No extra transmission required – each page transferred only once
- If migration fails, both source and target are in incorrect state

Hybrid Pre/Post-copy Migration



Combines both benefits and drawbacks of both pre and post migration

- Some extra page copying, but not as much
- Some cold start penalty, but not as much
- Some page faulting over network, but not much
- Improved reliability, but no post freeze recovery

Migrating Network Connections

Within a LAN

- The migrated VM carries its IP address, MAC address, and all protocol state, including open sockets
- Switches need to re-learn the new location of the VM's MAC address
- Send an unsolicited Address Resolution Protocol (ARP) reply from target… switches will relearn

Across a WAN

- Source and target subnets may have different IP addresses
- May have to close down and re-open connections
- Or tunnel using VPN or a similar mechanism

Migrating Disk Data

Many gigabytes of local disk image possible!

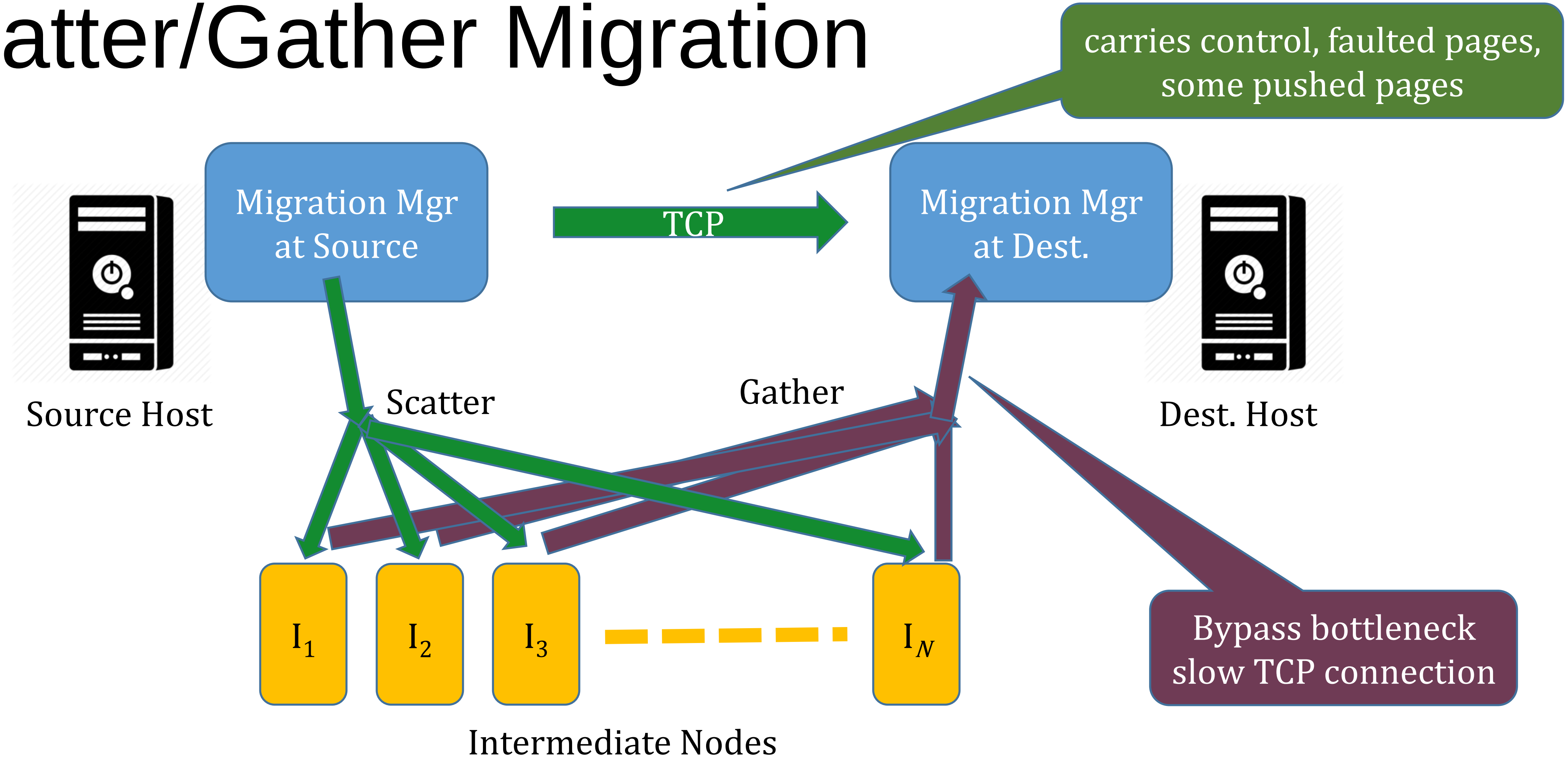
Within a LAN

- Assume the disks are available on the network, and accessible from the target
- NFS (Network File System), AFS (Andrew File System), NBD (Nework Block Device), ISCSI
  ….

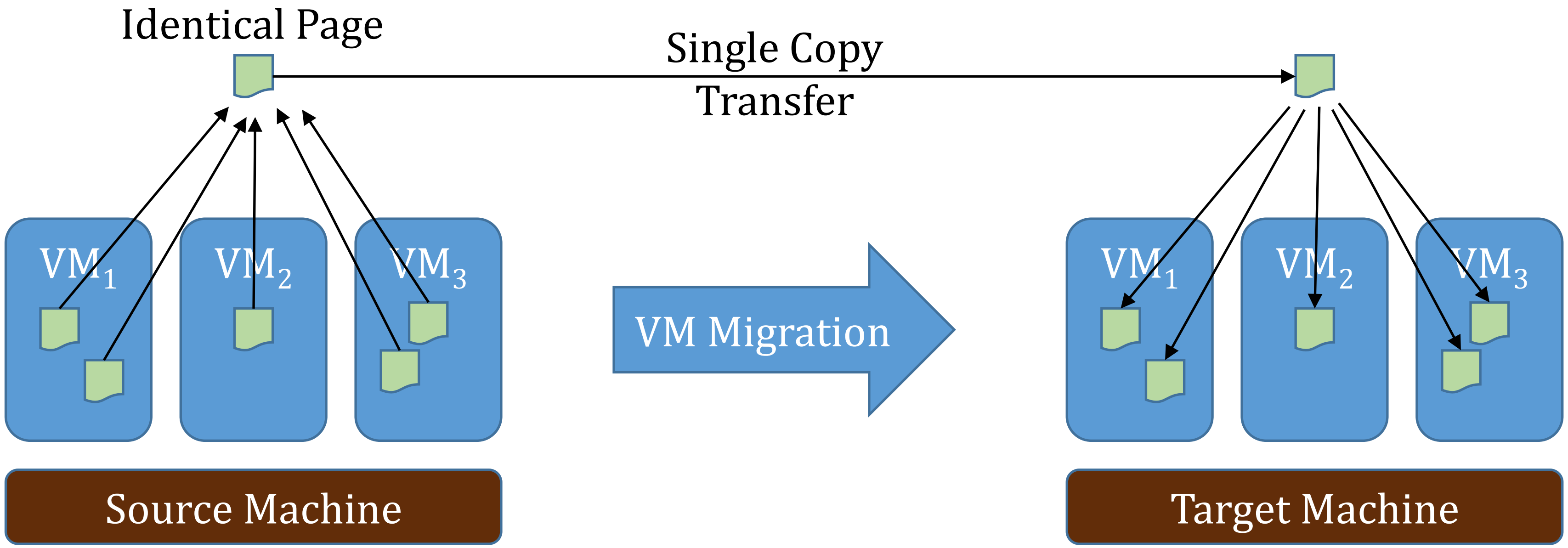
Across a WAN

- Disk image may need to be transferred
- Can be pre-copy or post-copy
- May need bandwidth saving optimization, such as compression and/or de-duplication

Scatter/Gather Migration



Multi-VM (Gang) Migration



De-Duplicate pages to reduce network traffic

- Most commonly shared memory pages (libraries)
- Identify multiple pages across VMs
  - byte-wise comparison expensive
  - checksum is cheaper
- Send single copy over network
- Re-distribute at target