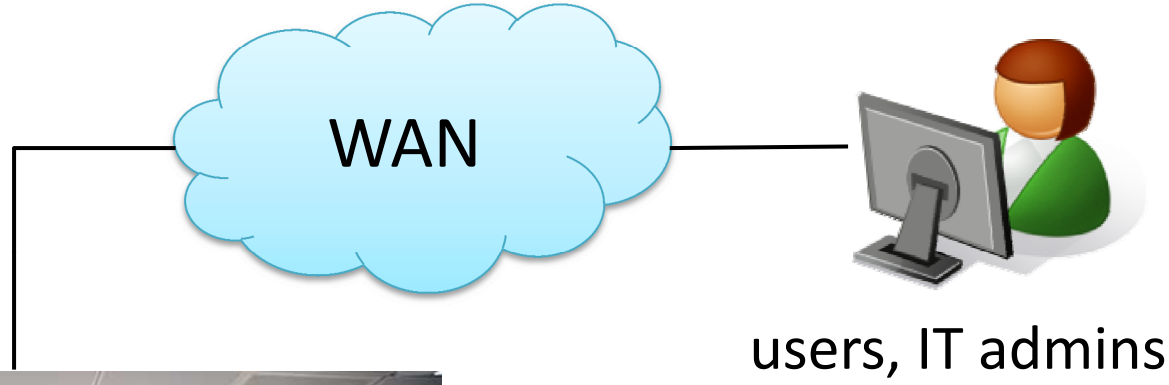
Don't Lose Sleep Over Availability: The GreenUp Decentralized Wakeup Service

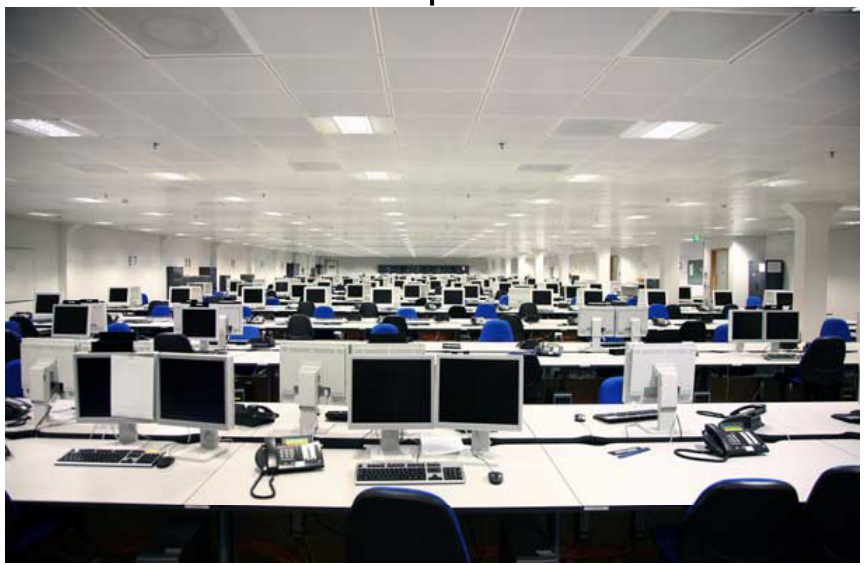
Siddhartha Sen, Princeton University

Jacob R. Lorch, Richard Hughes, Carlos G. J. Suarez,
Brian Zill, Weverton Cordeiro, and Jitendra Padhye

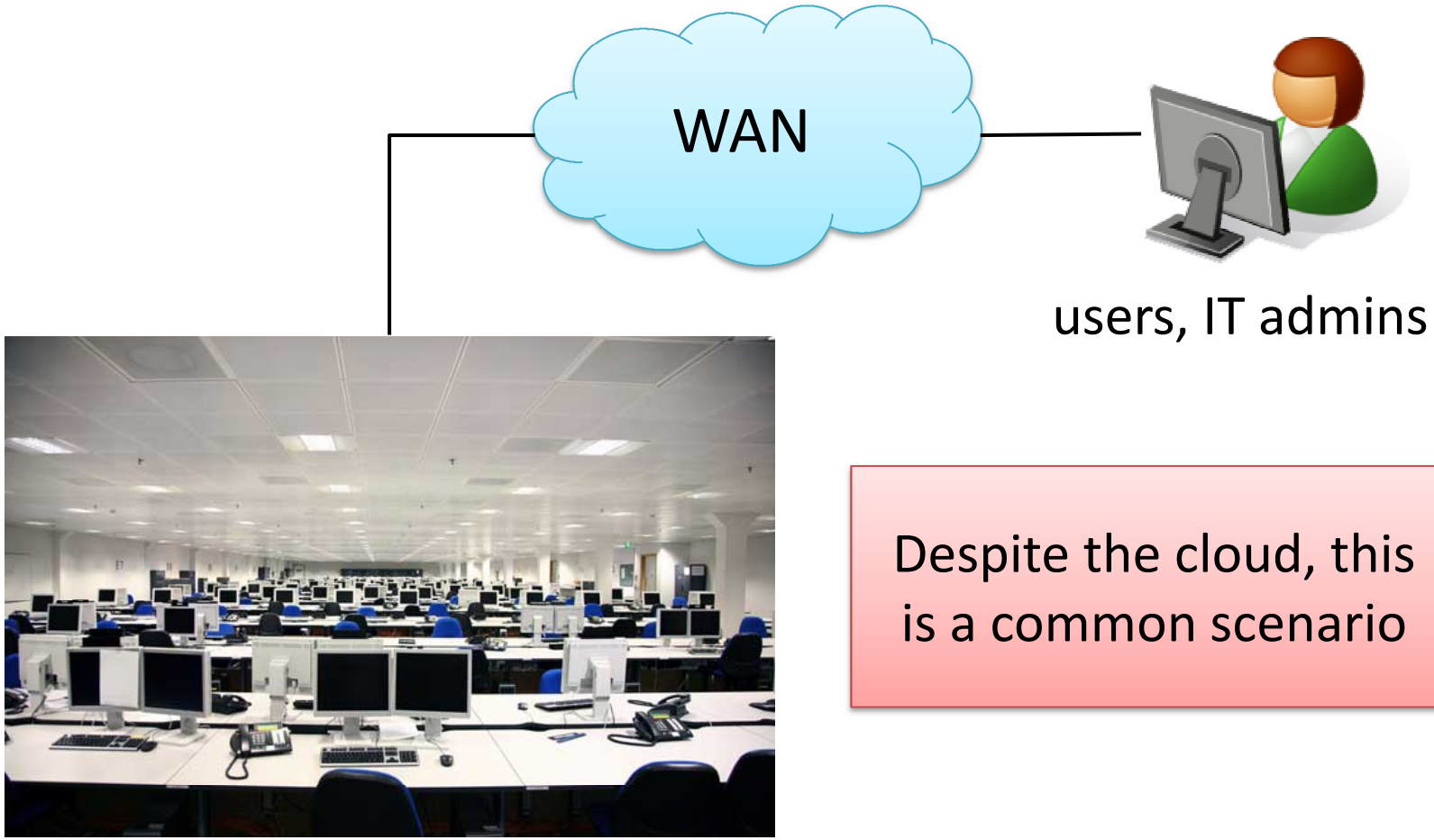
Enterprise networks

Enterprise networks



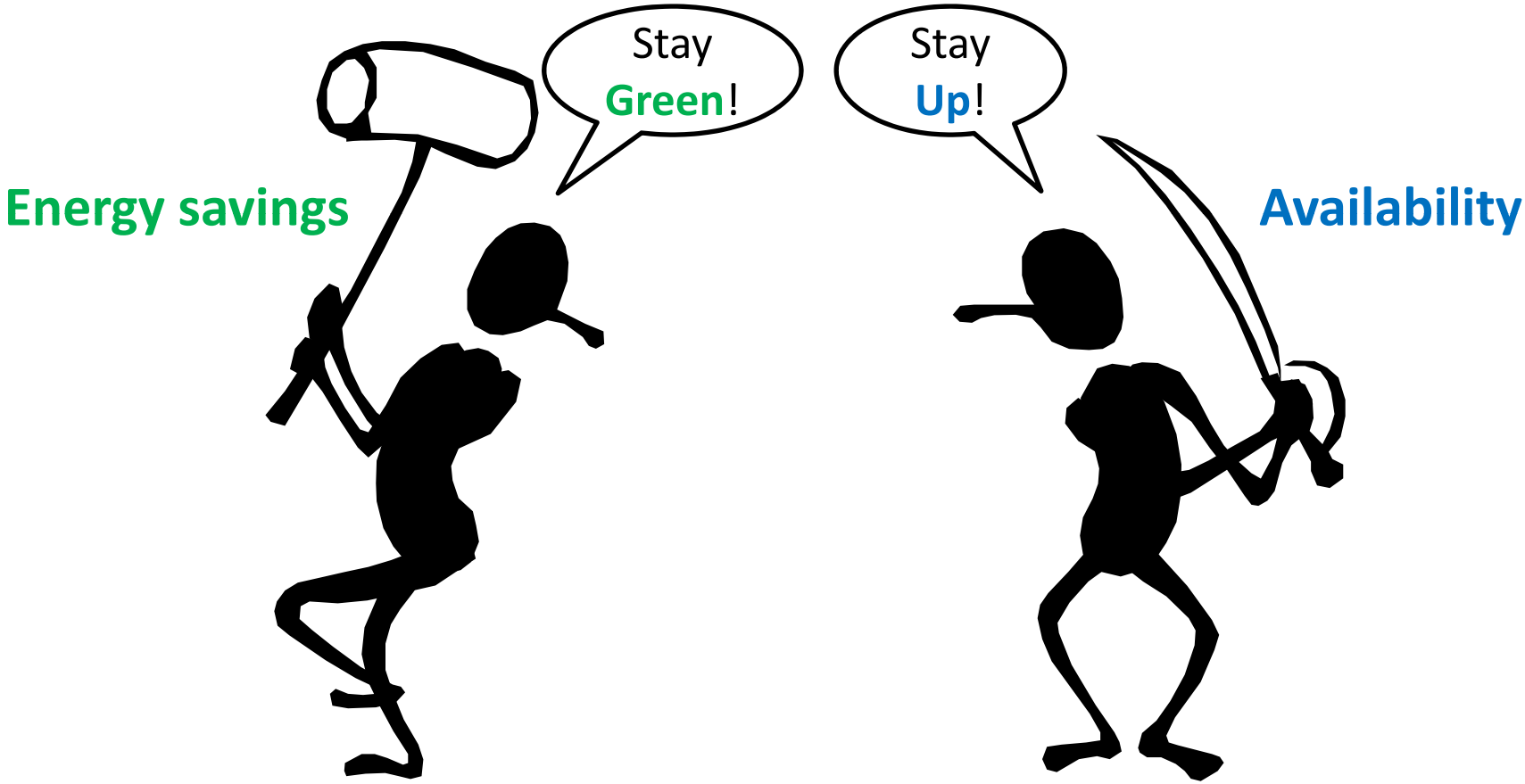


Enterprise networks

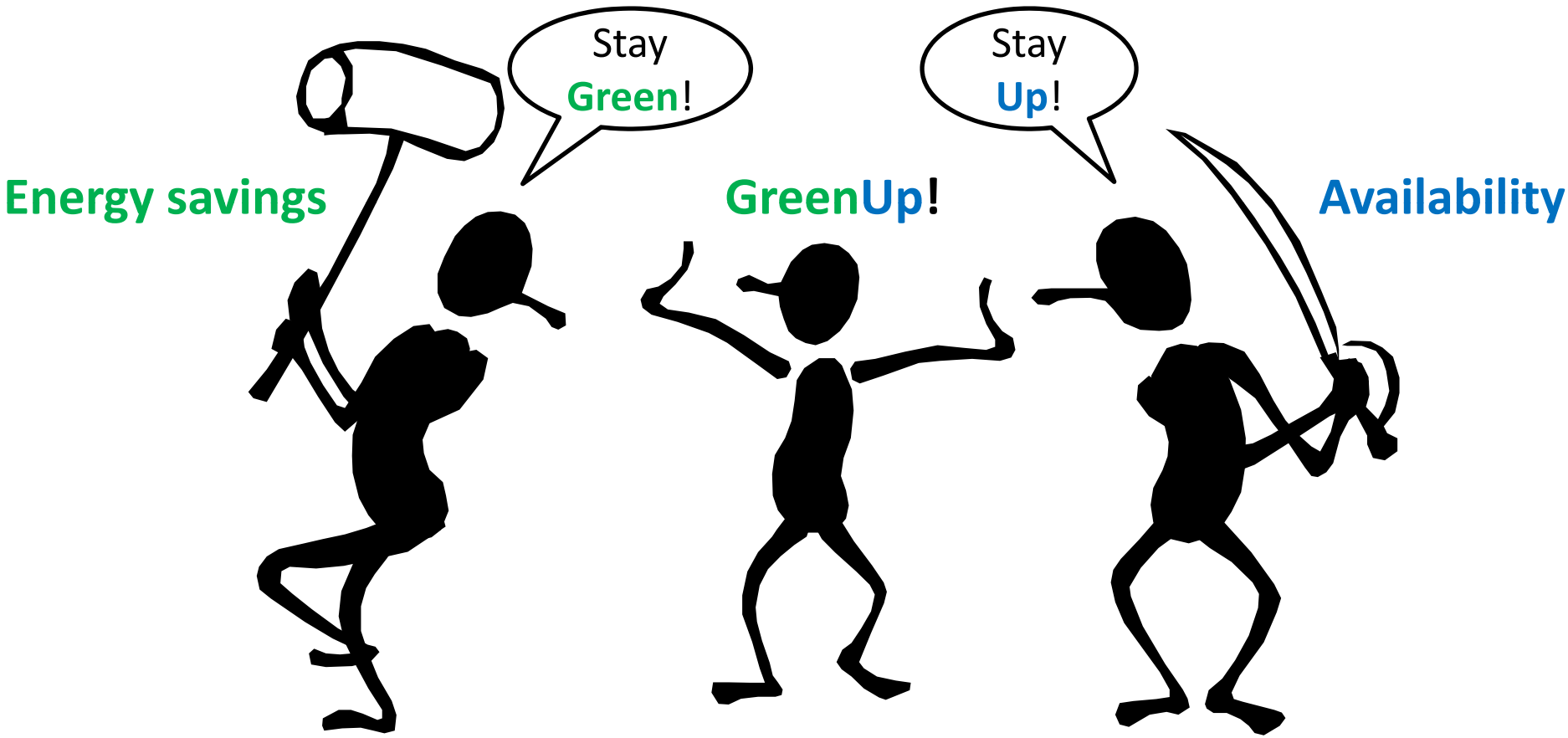


Despite the cloud, this is a common scenario

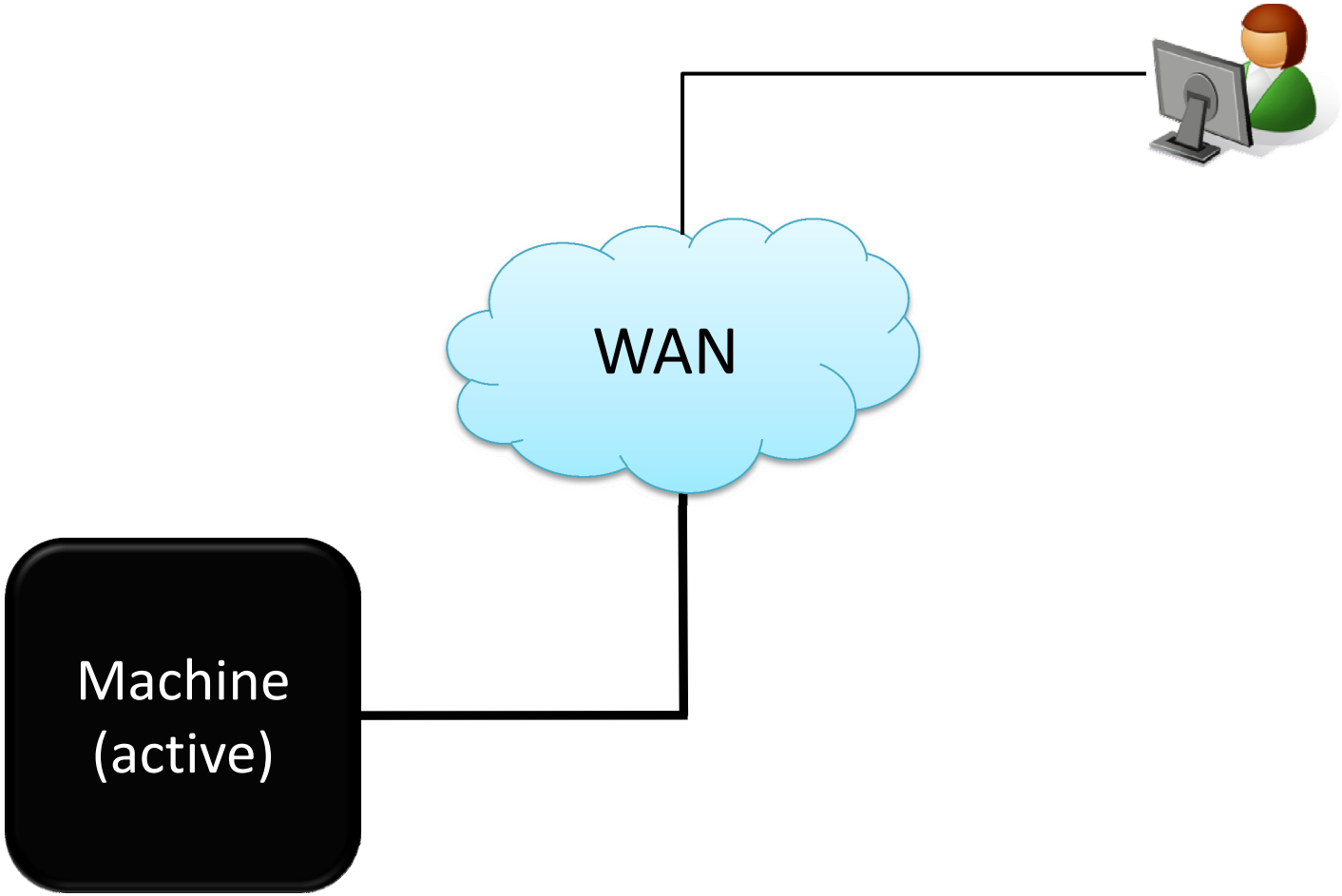
Enterprise networks



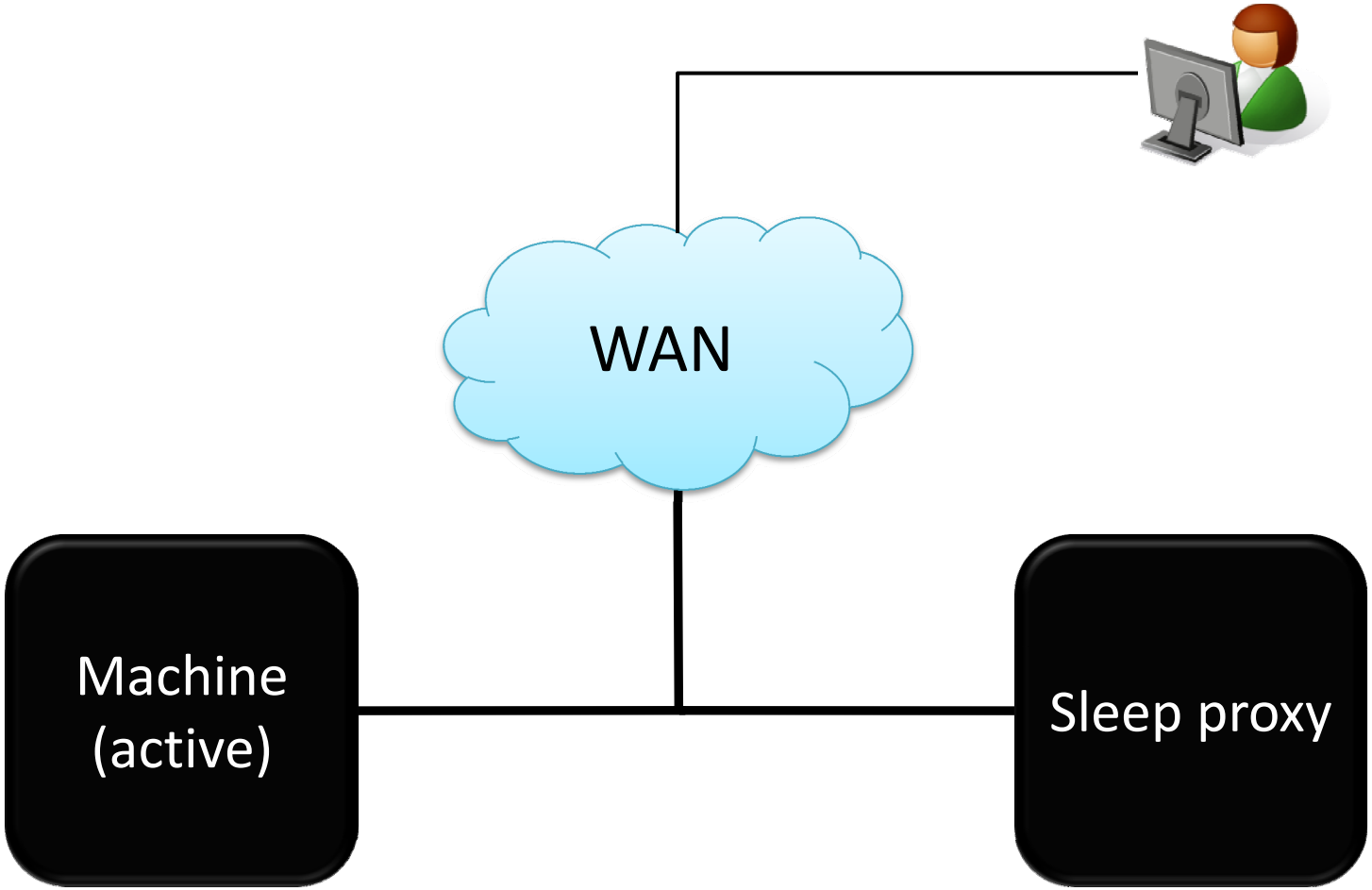
Enterprise networks



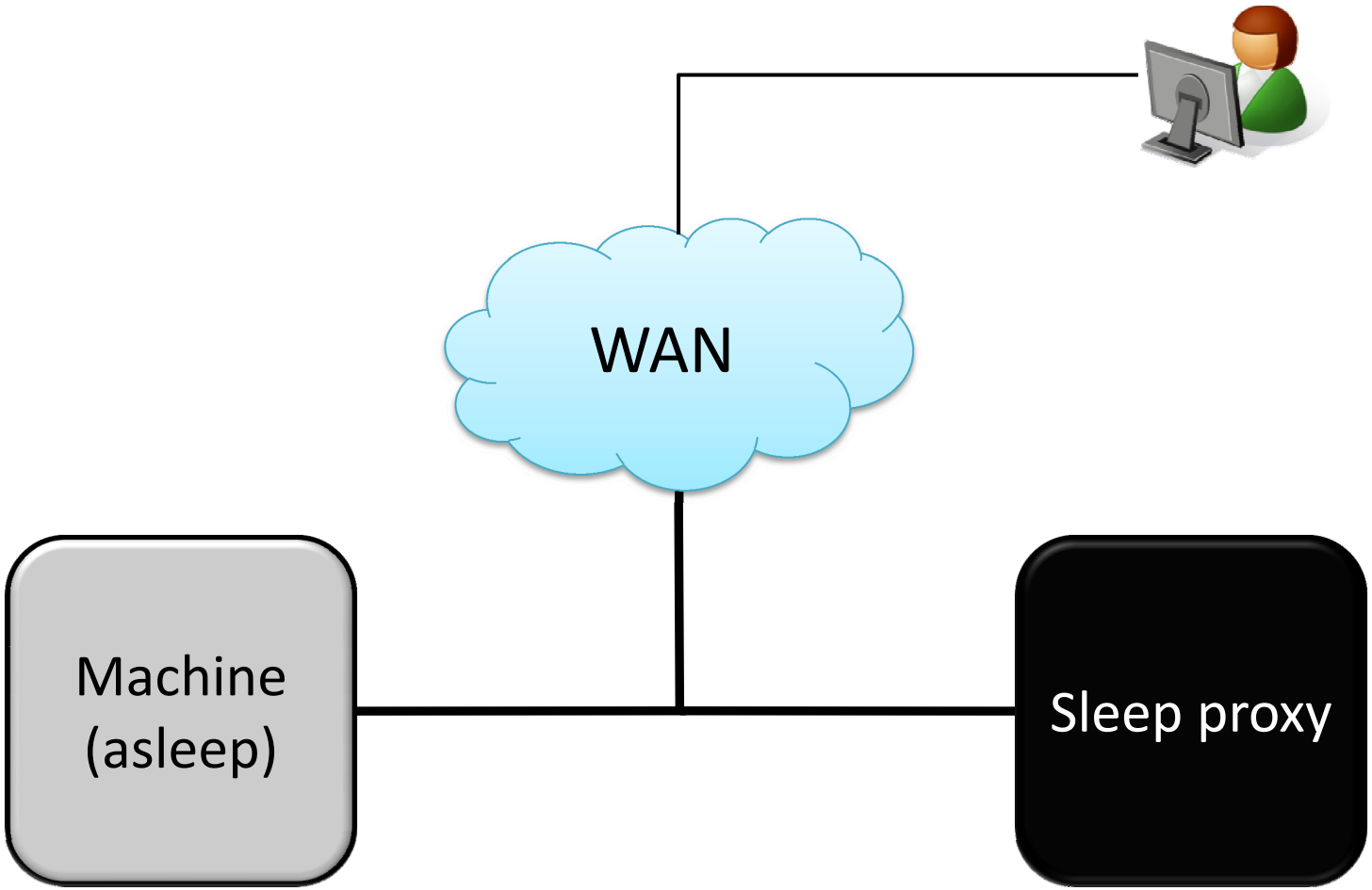
Sleep proxy



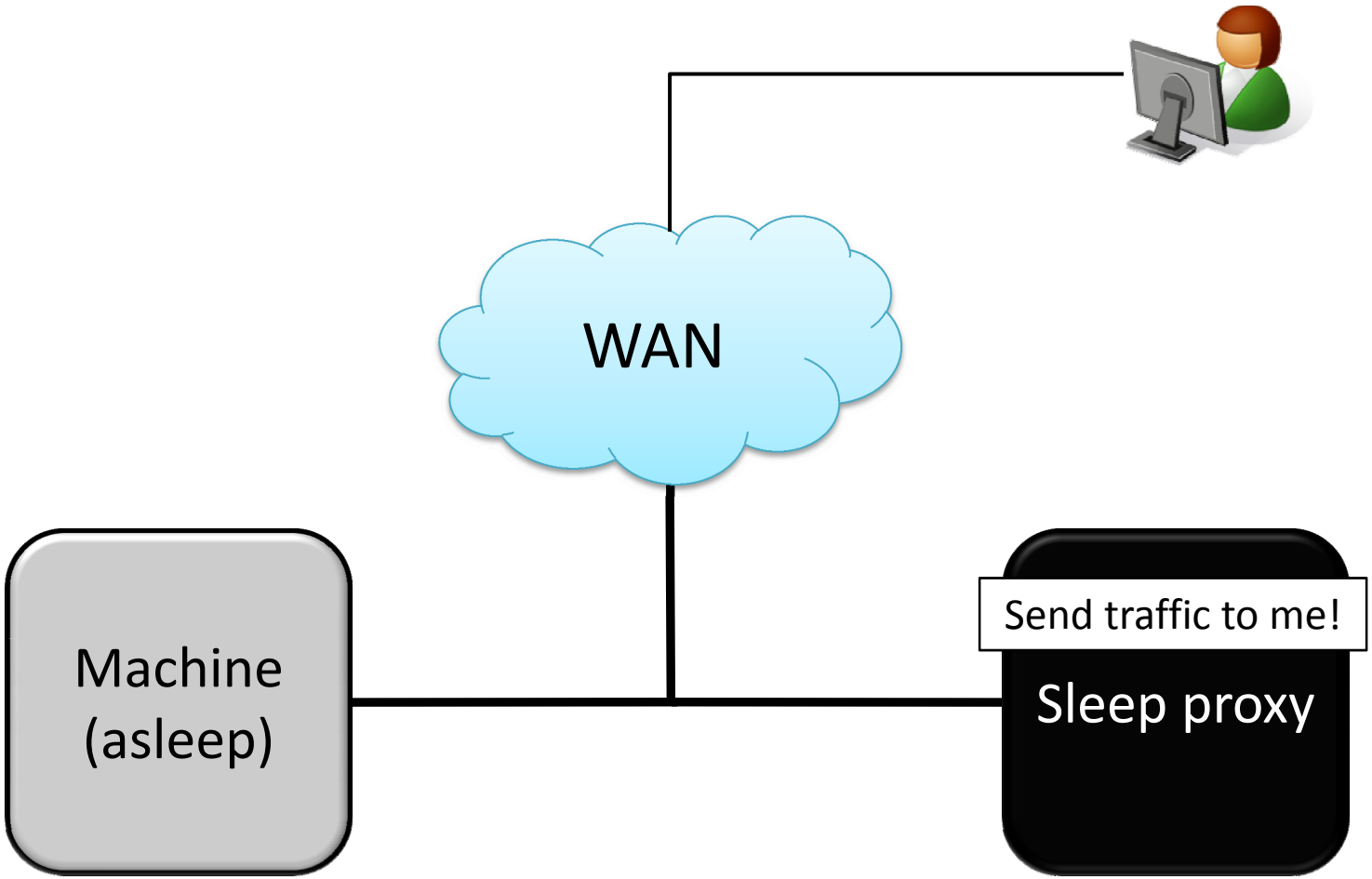
Sleep proxy



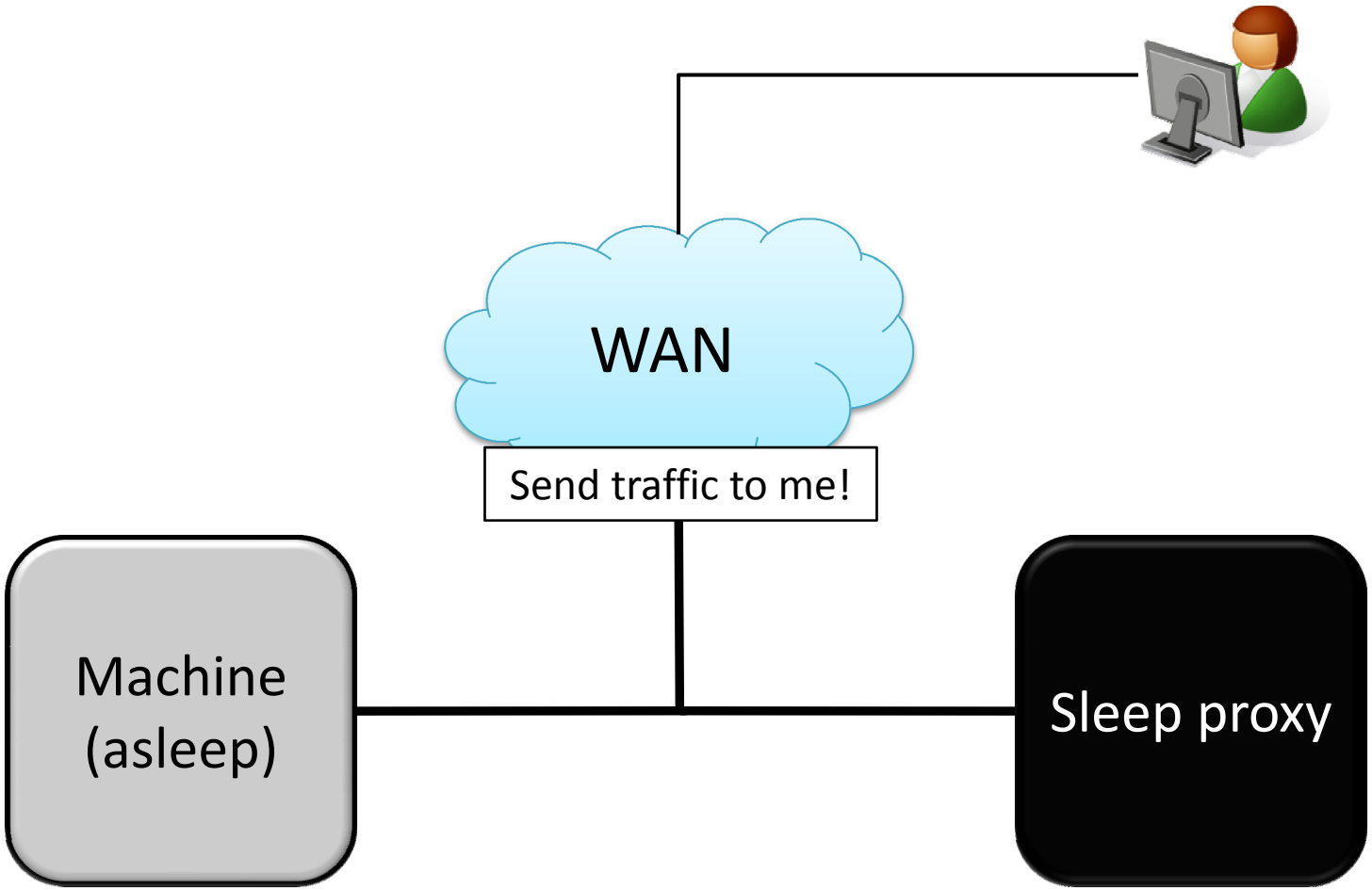
Sleep proxy



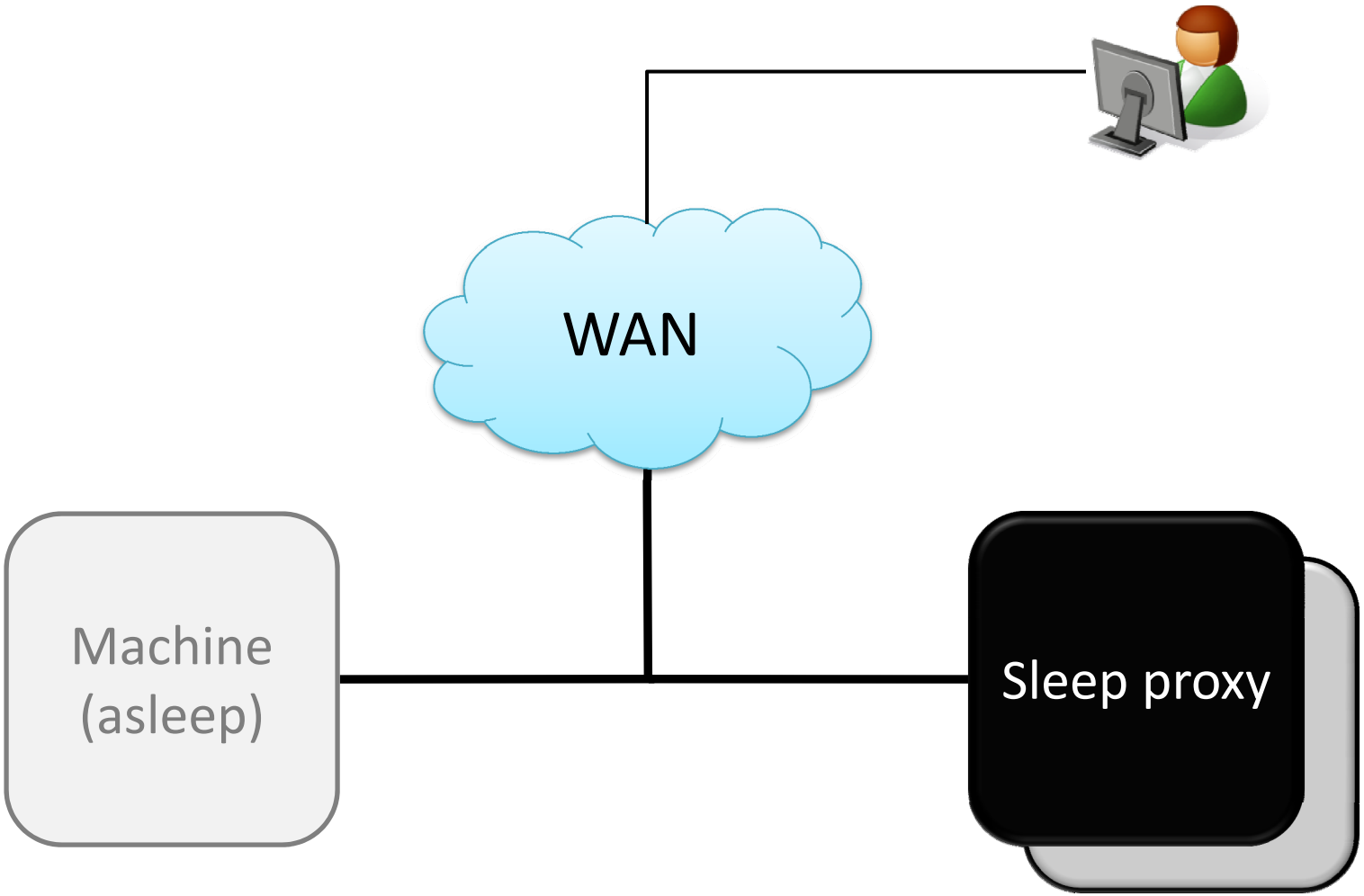
Sleep proxy



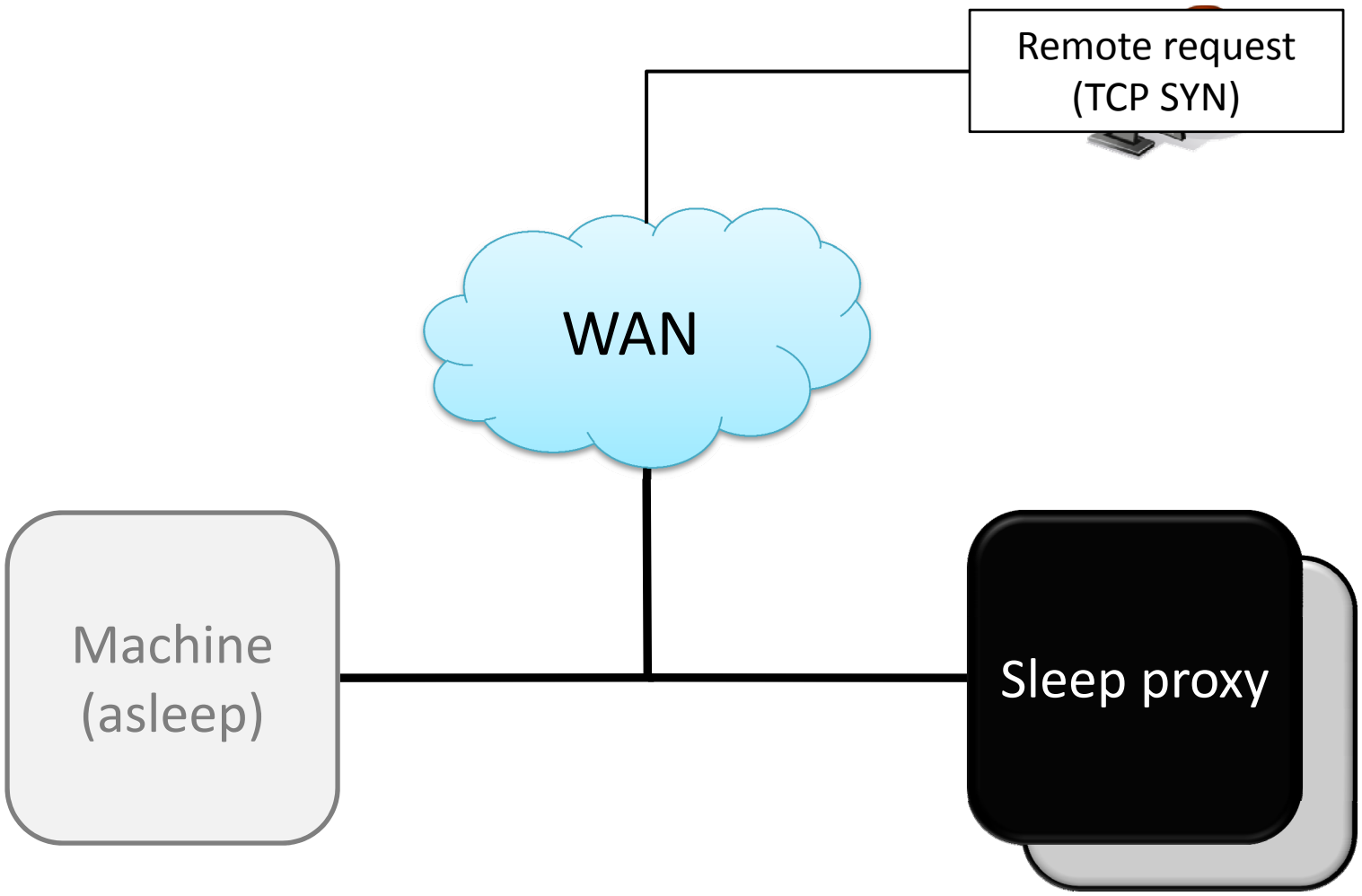
Sleep proxy



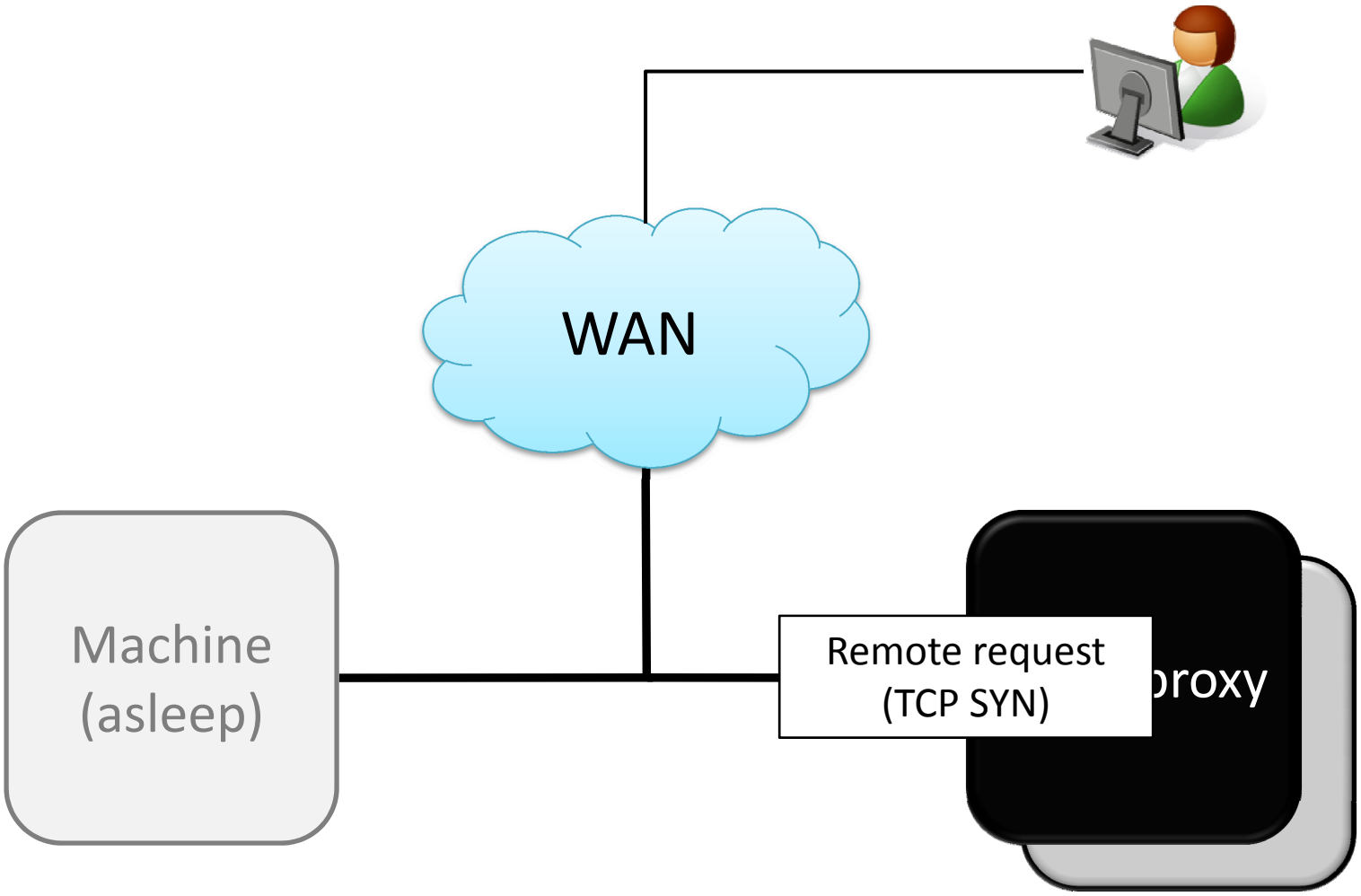
Sleep proxy



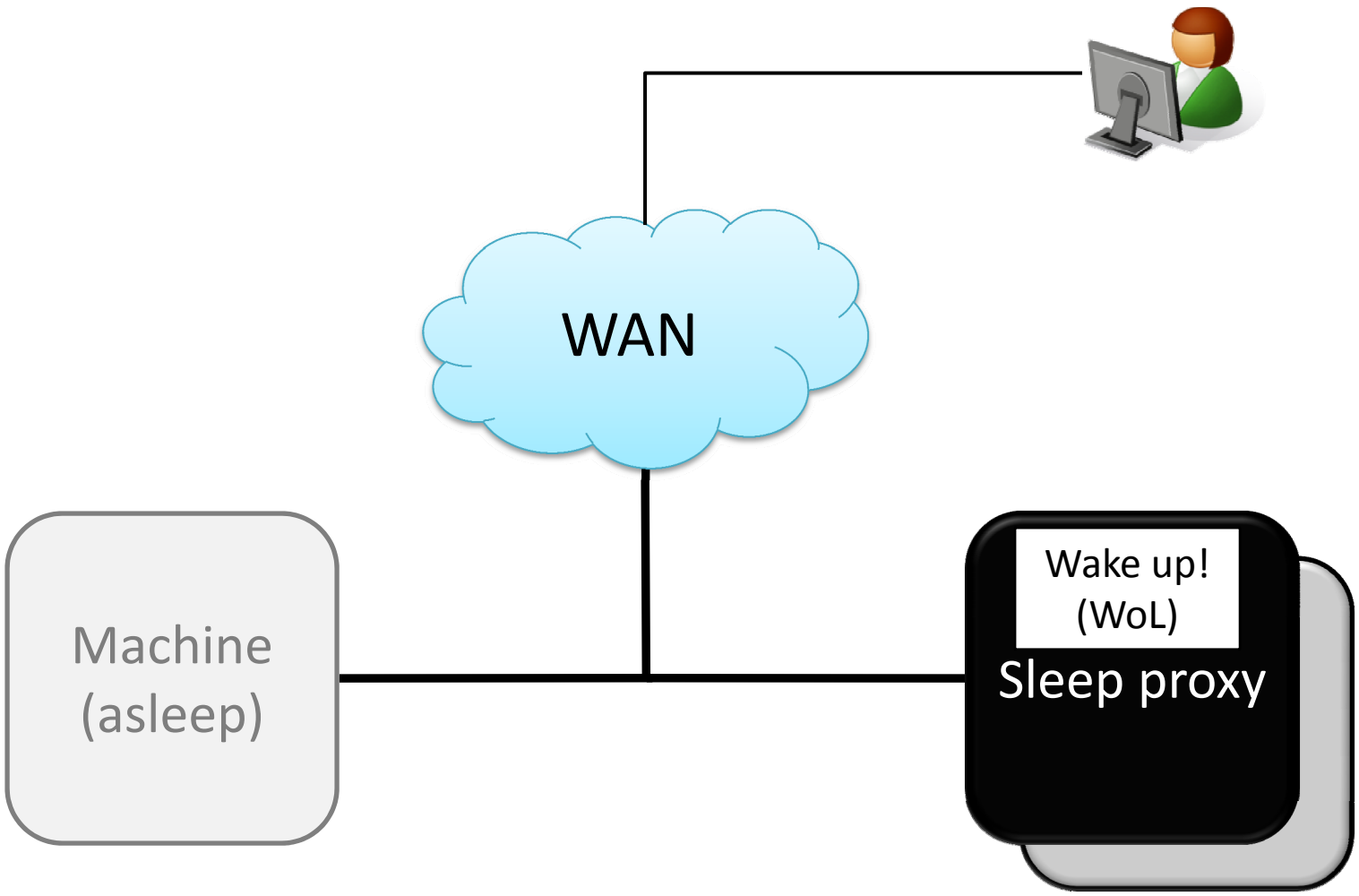
Sleep proxy



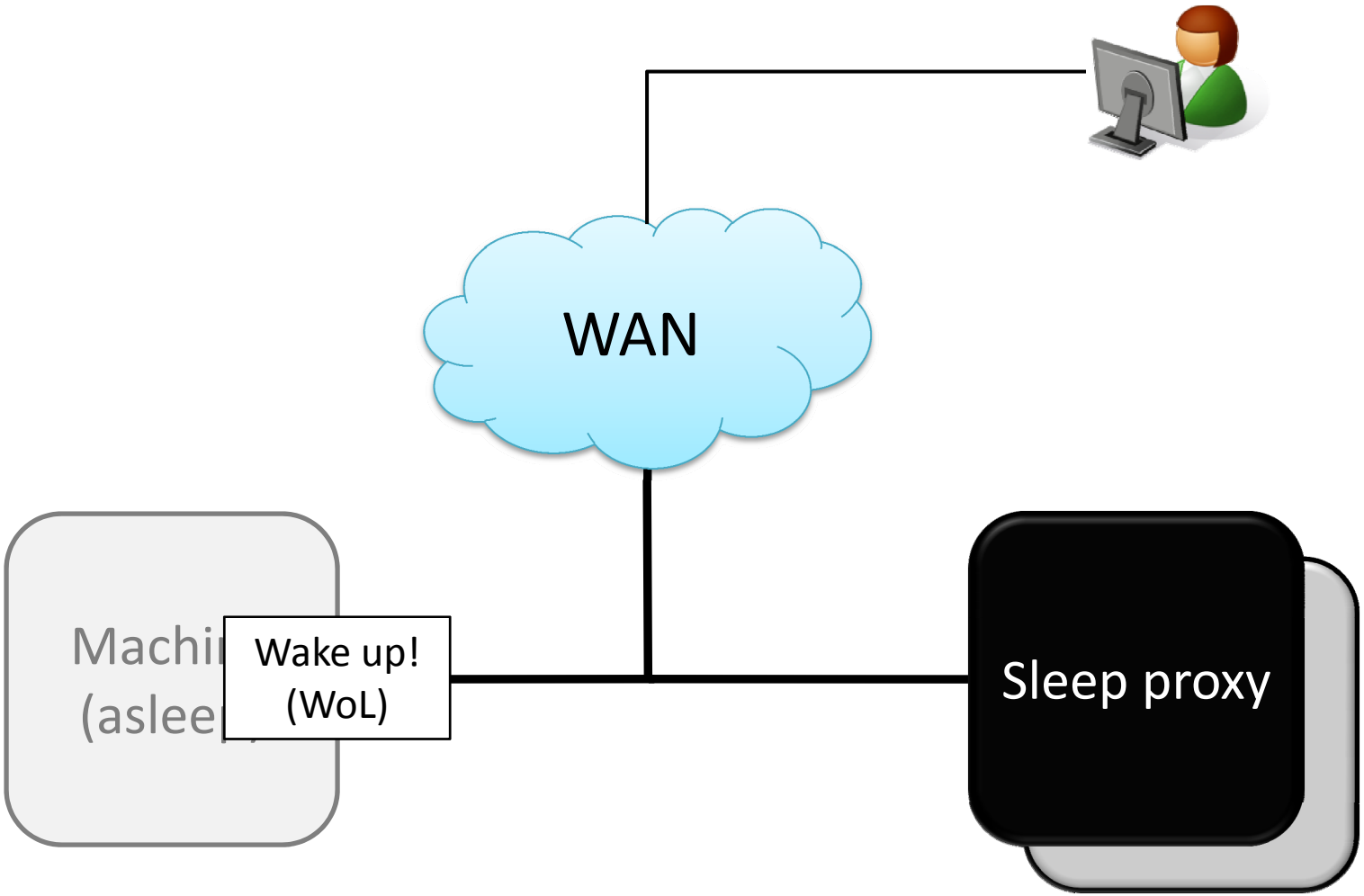
Sleep proxy



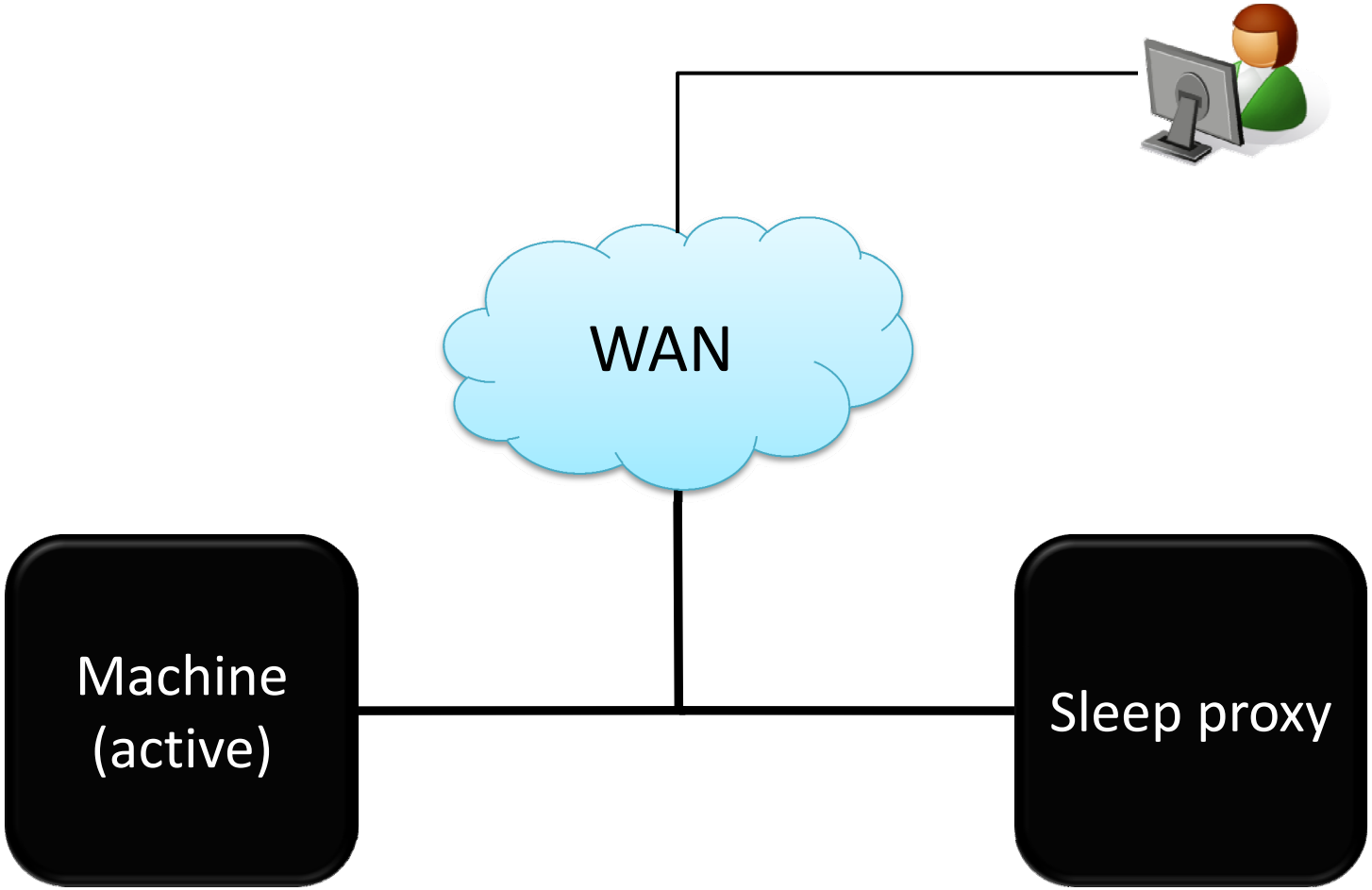
Sleep proxy



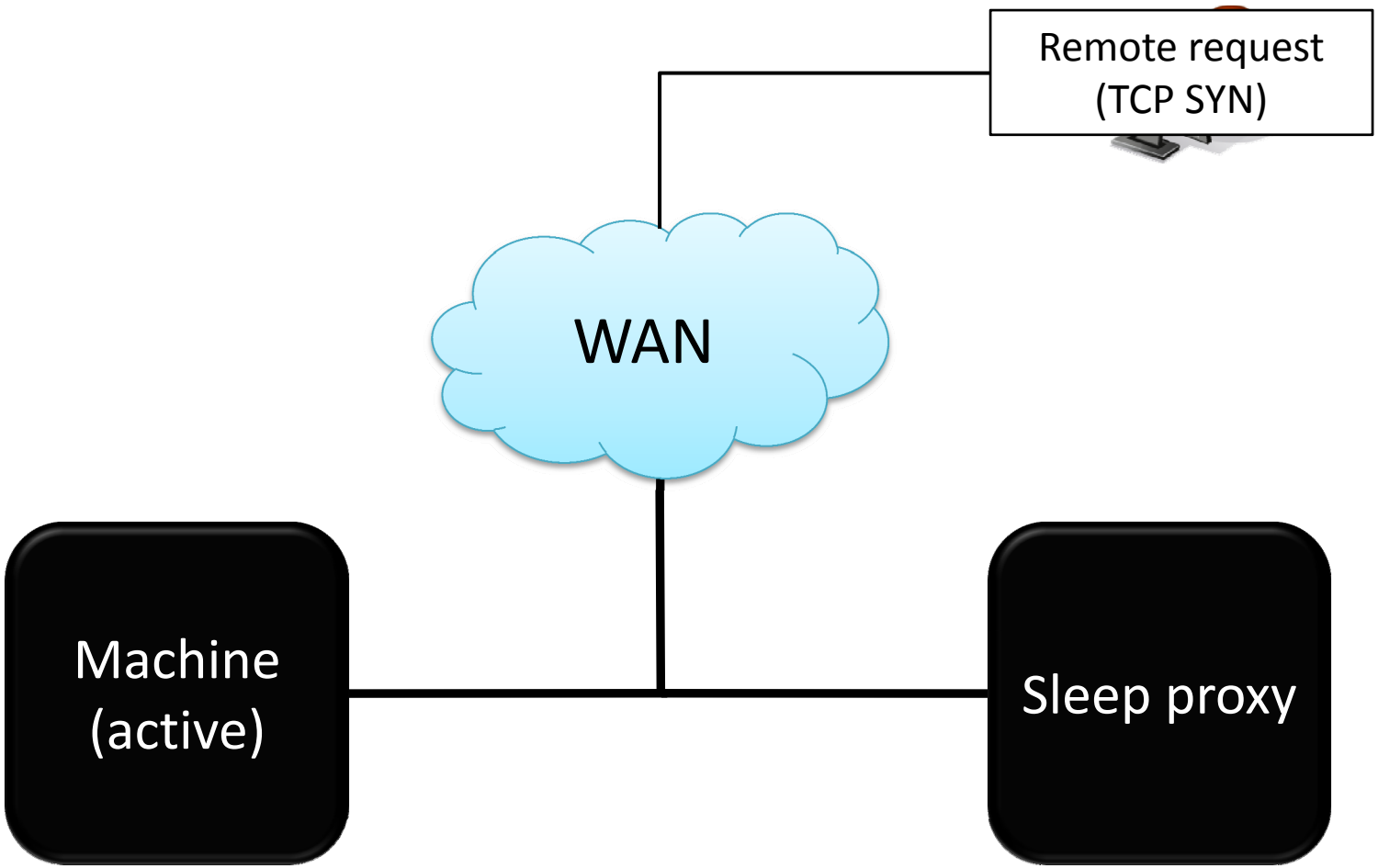
Sleep proxy



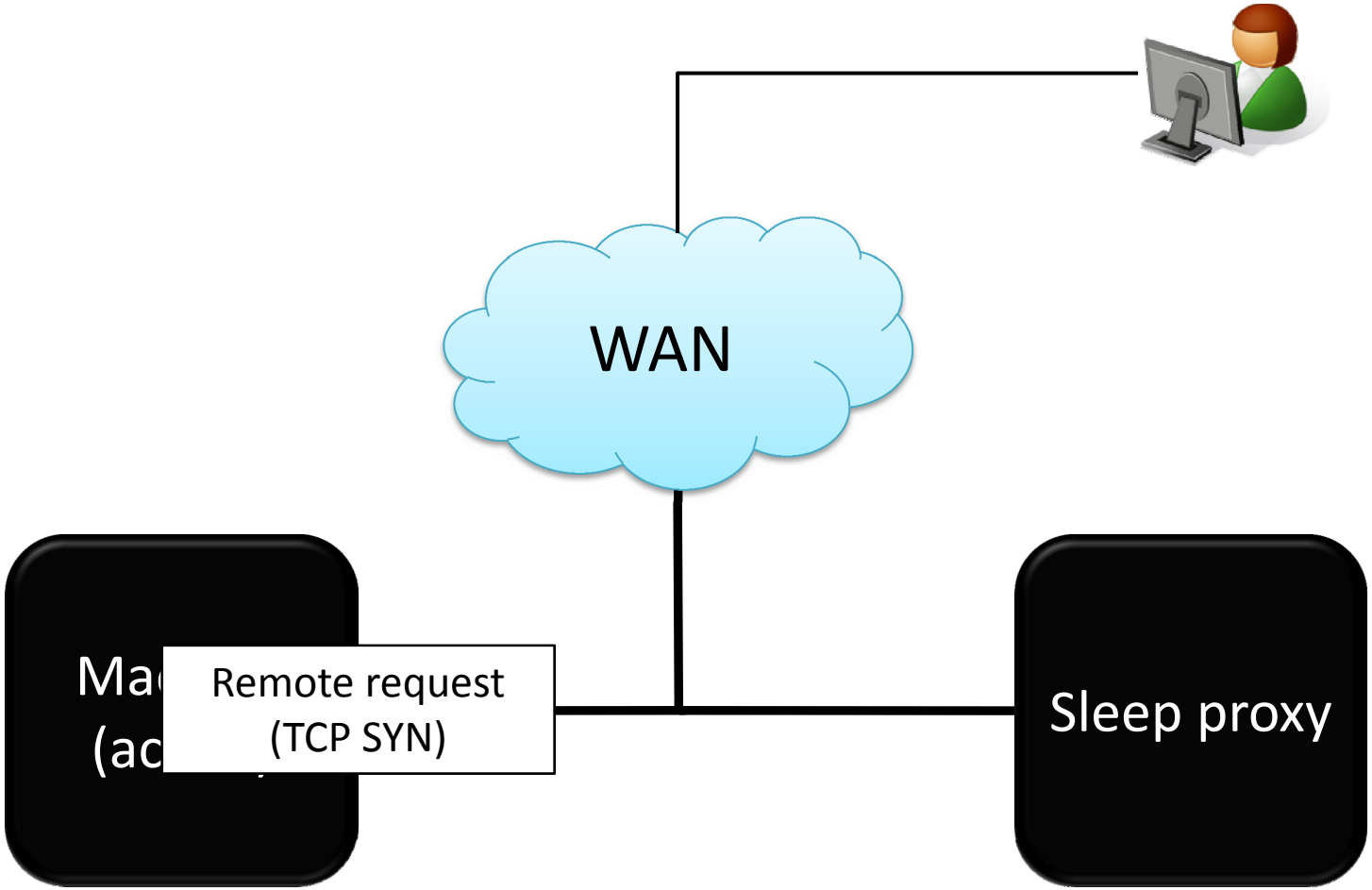
Sleep proxy



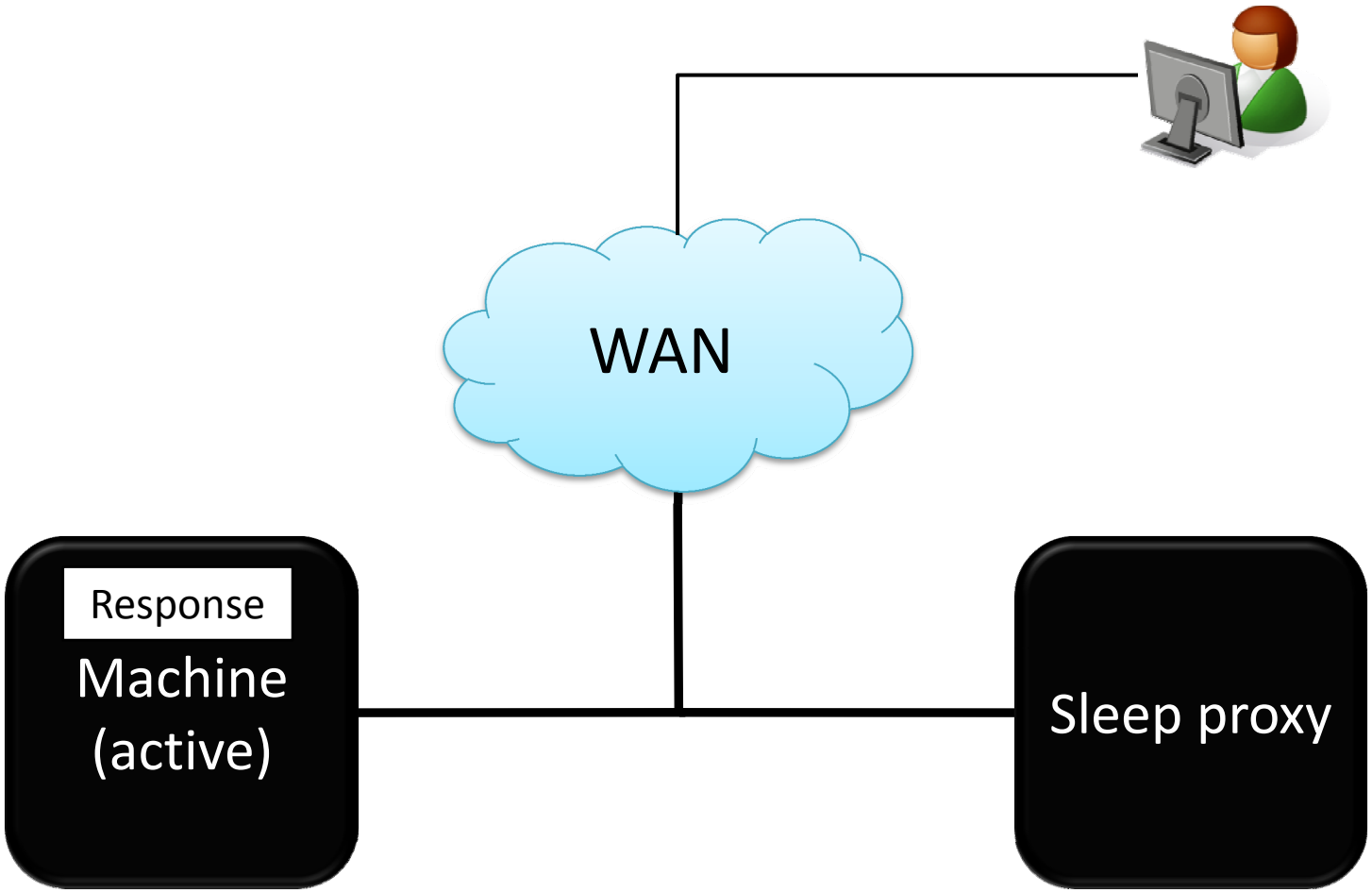
Sleep proxy



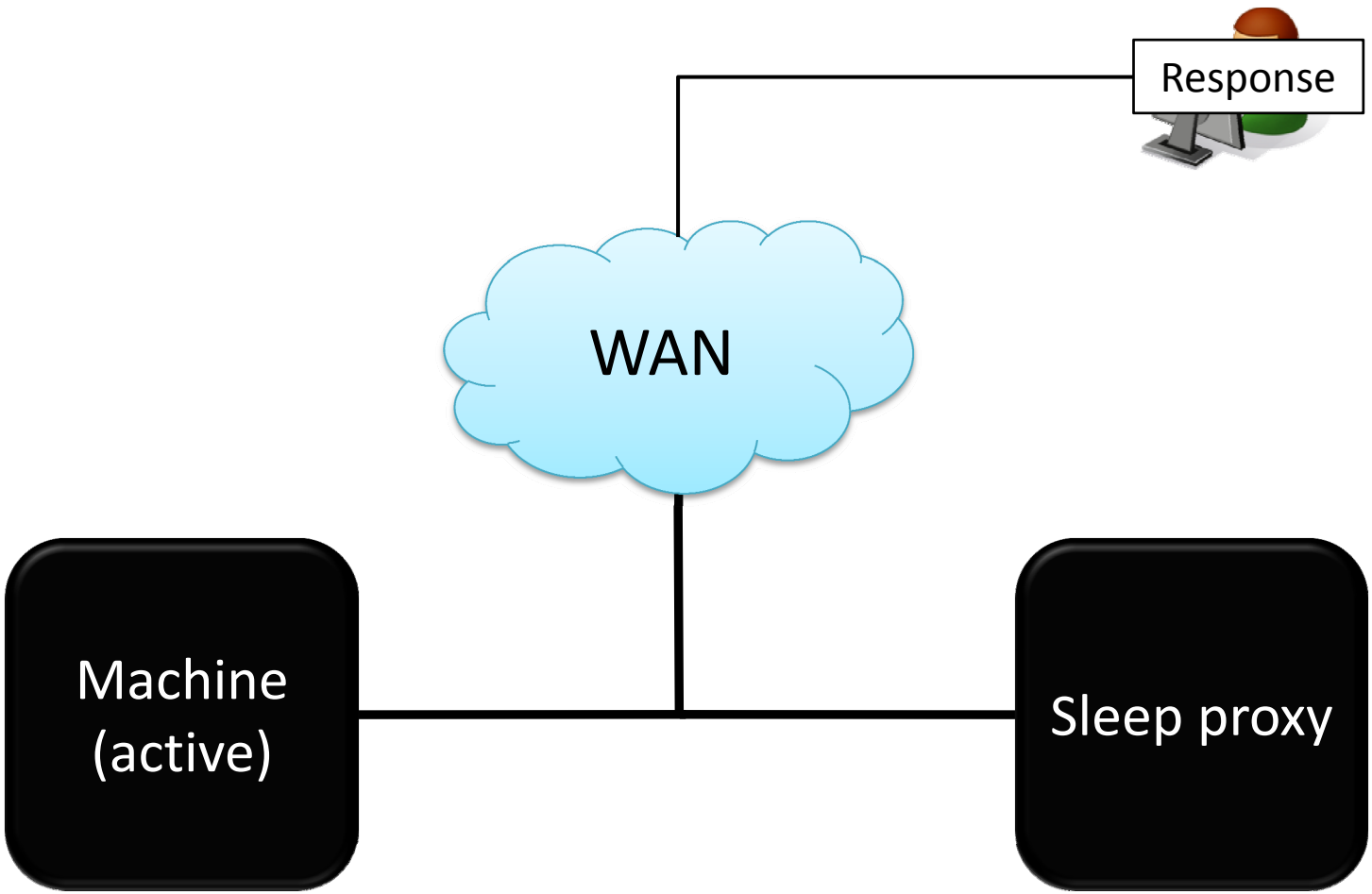
Sleep proxy



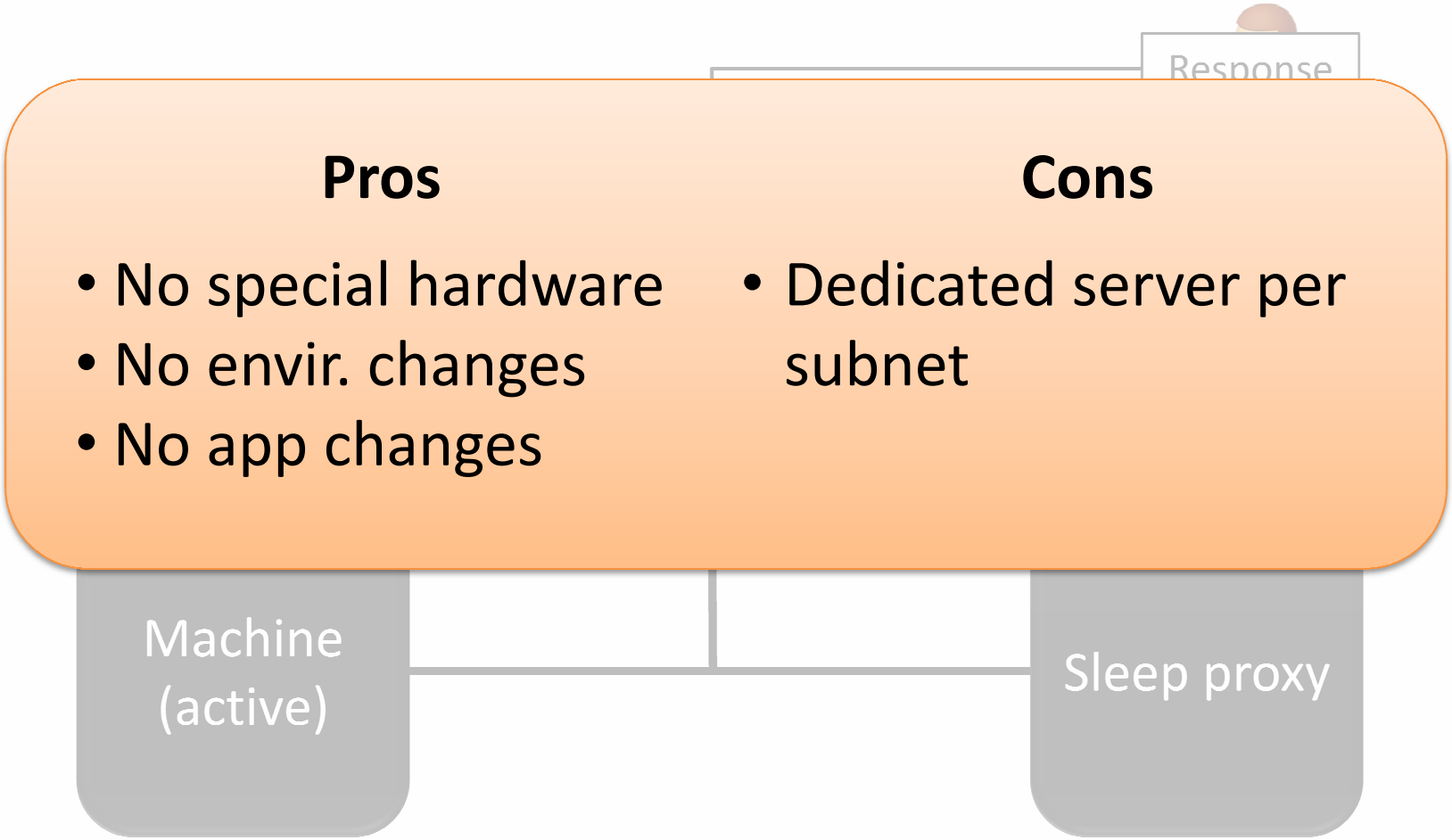
Sleep proxy



Sleep proxy



Sleep proxy


Response
Pros
• No special hardware
• No envir. changes
• No app changes
Cons
• Dedicated server per subnet
Machine (active)
Sleep proxy

Dedicated servers are a **problem**

- High deployment and management cost
- Single point of failure

Dedicated servers are a **problem**

- High deployment and management cost
- Single point of failure
- High availability becomes expensive!

GreenUp: A decentralized, minimal software-only sleep proxy

GreenUp: A decentralized, minimal software-only sleep proxy

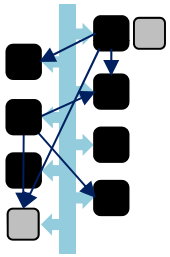
Any machine can act as a proxy (manager) for sleeping machines on the subnet

Outline

1. How does GreenUp work?
2. What can I learn from GreenUp?

Outline

1. How does GreenUp work?
2. What can I learn from GreenUp?



Distributed management

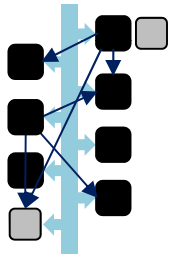
Machine	State
…	…
…	…
…	…

Subnet state coordination

Guardians

Outline

1. How does GreenUp work?
2. What can I learn from GreenUp?



Distributed management

Machine	State
…	…
…	…
…	…

Subnet state coordination

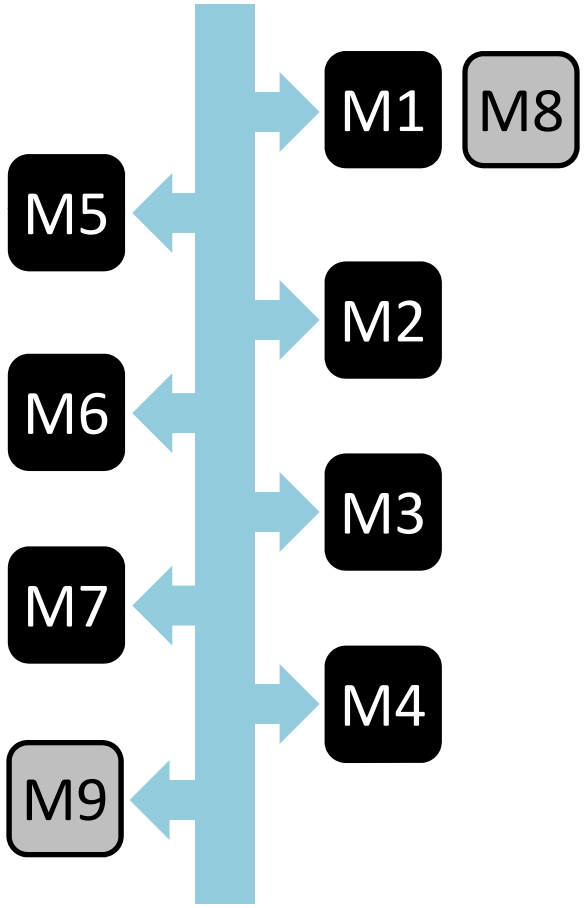
Guardians

3. How effective is GreenUp?
   - Evaluation on ~**100** user machines, currently deployed on ~**1,100** machines

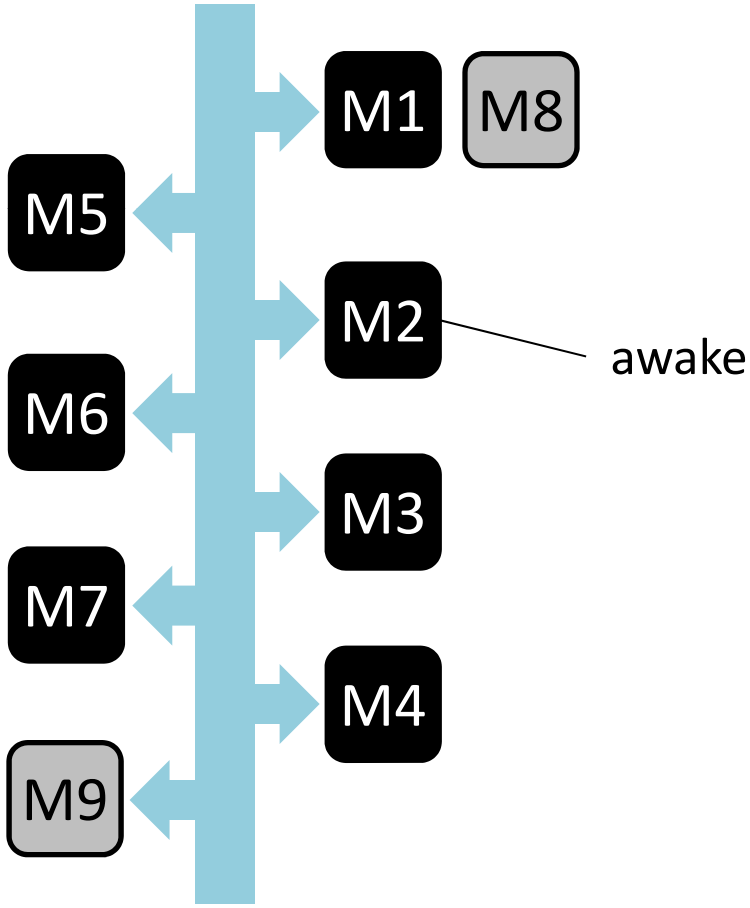
GreenUp’s environment

- Subnet domains
- Load-sensitive, unreliable machines
- Single administrative domain
- Availability most important

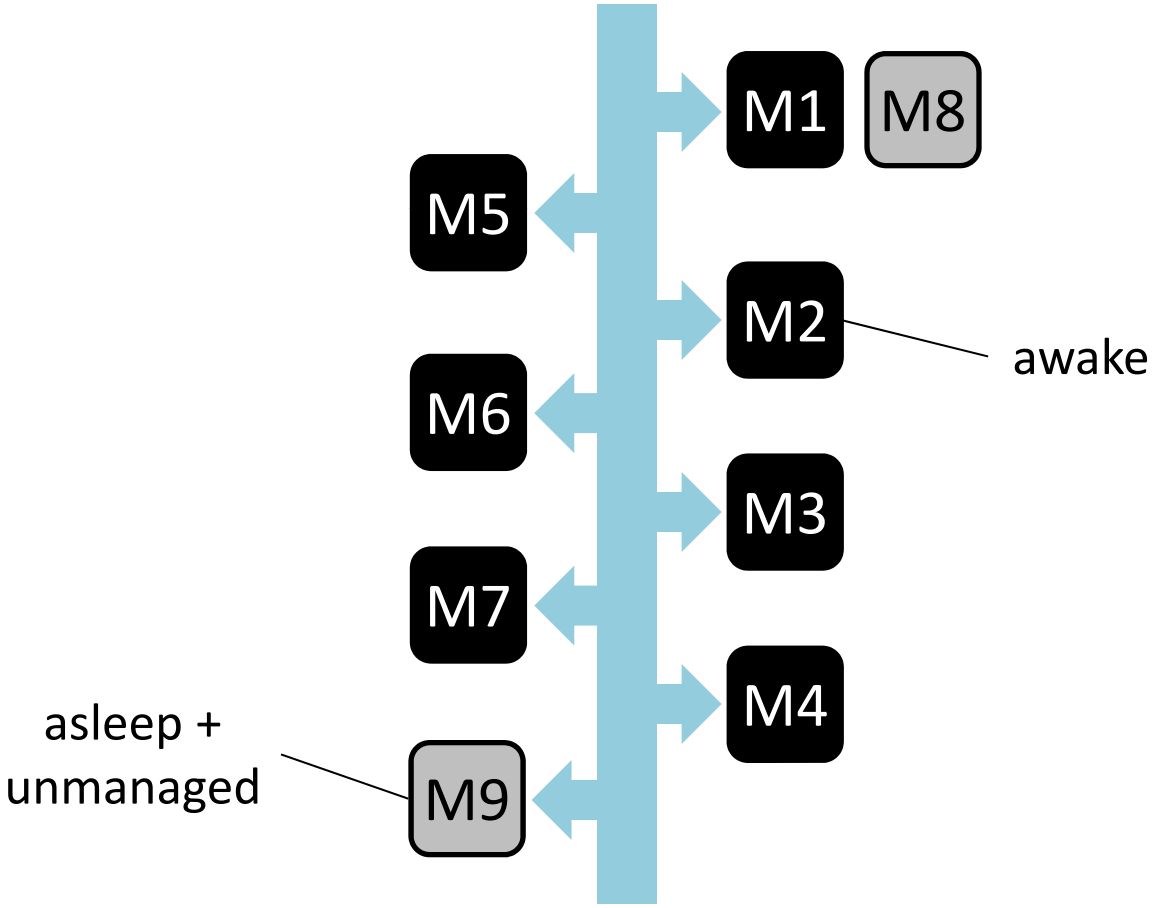
Running example (not to scale)



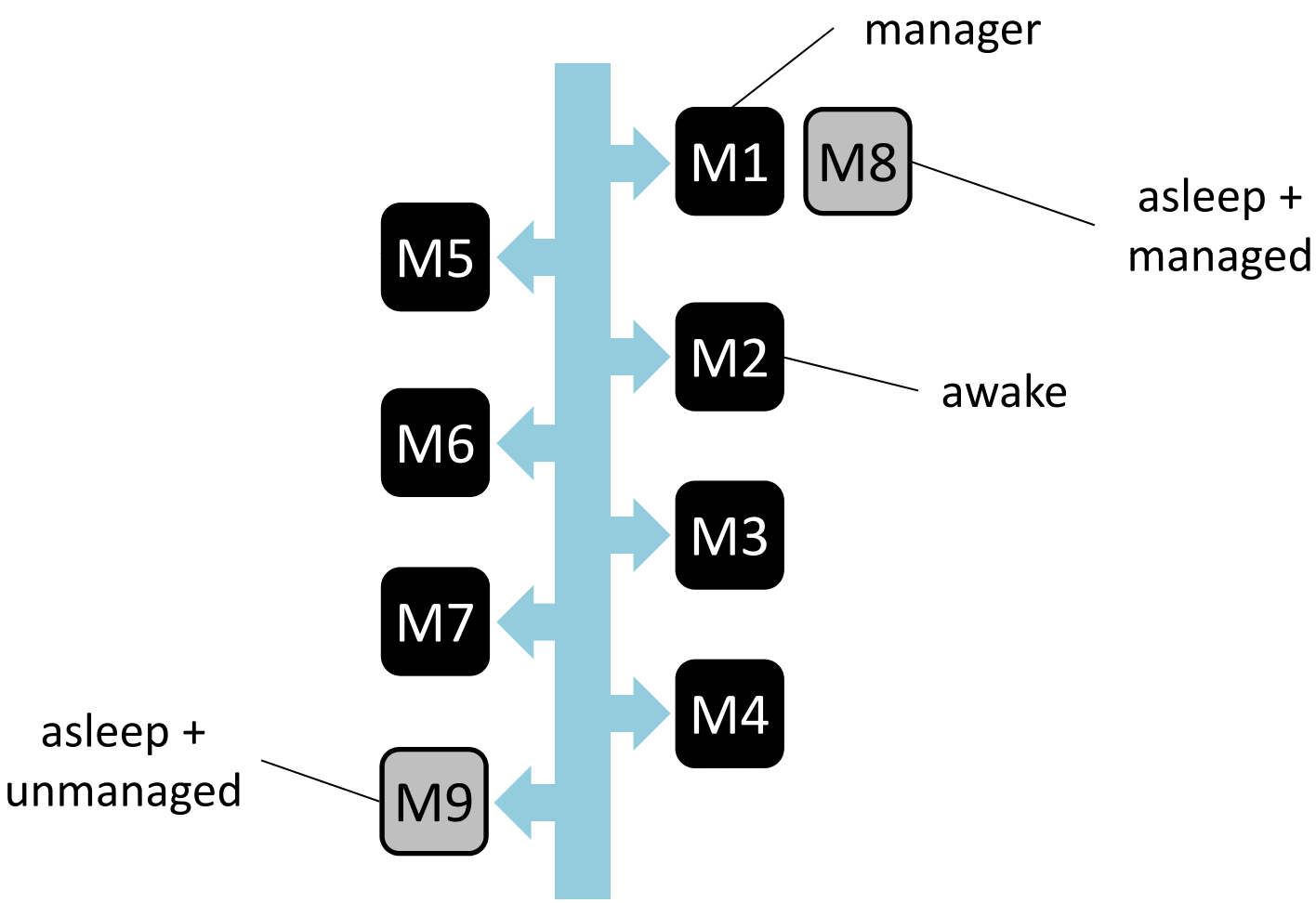
Running example (not to scale)



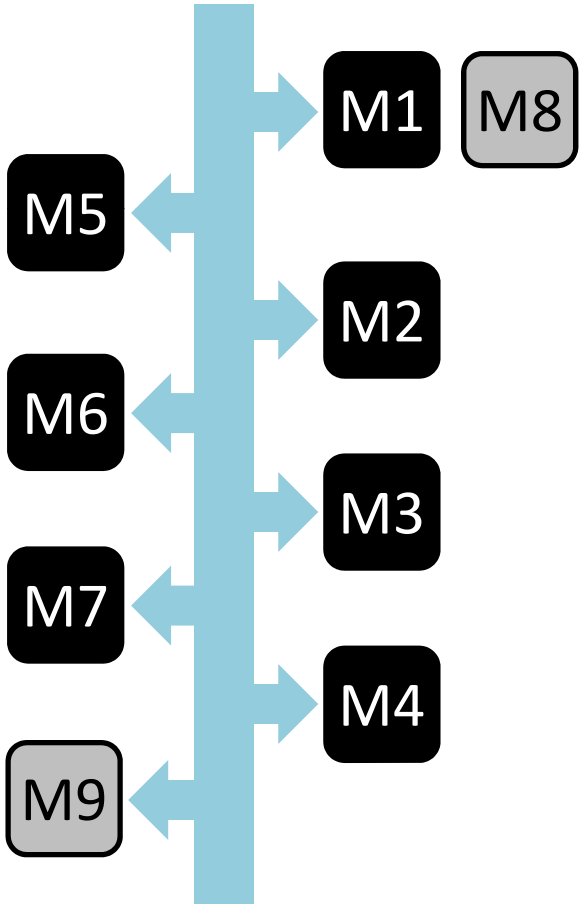
Running example (not to scale)



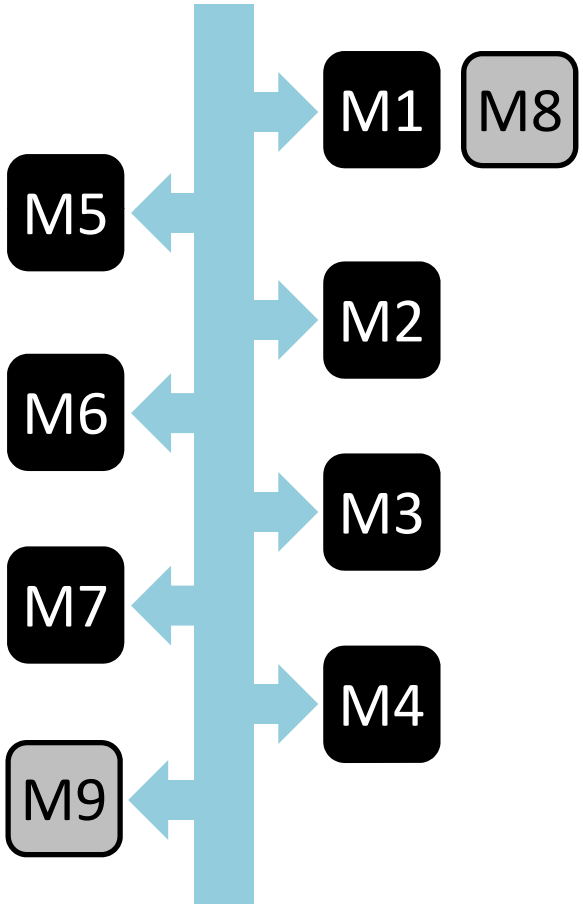
Running example (not to scale)



Distributed management: Who manages M9?

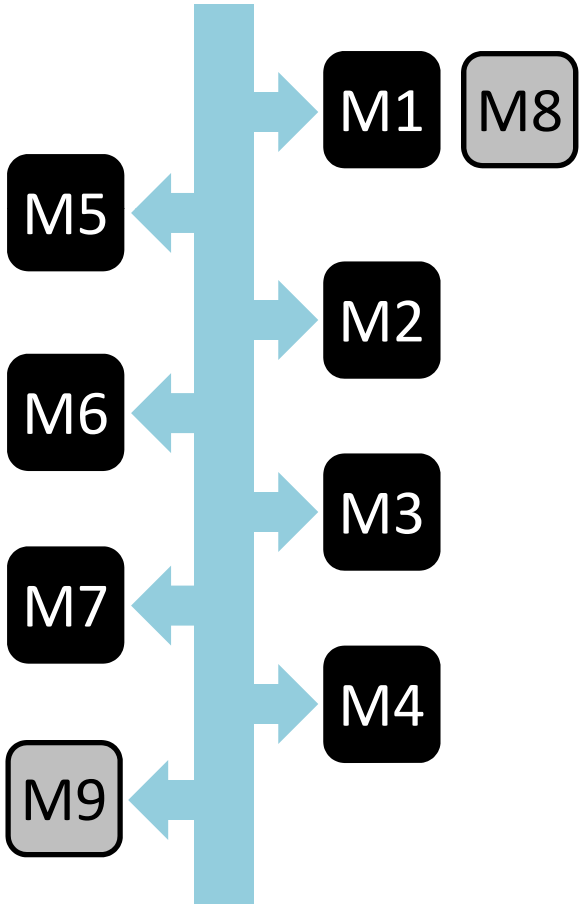


Distributed management: Who manages M9?



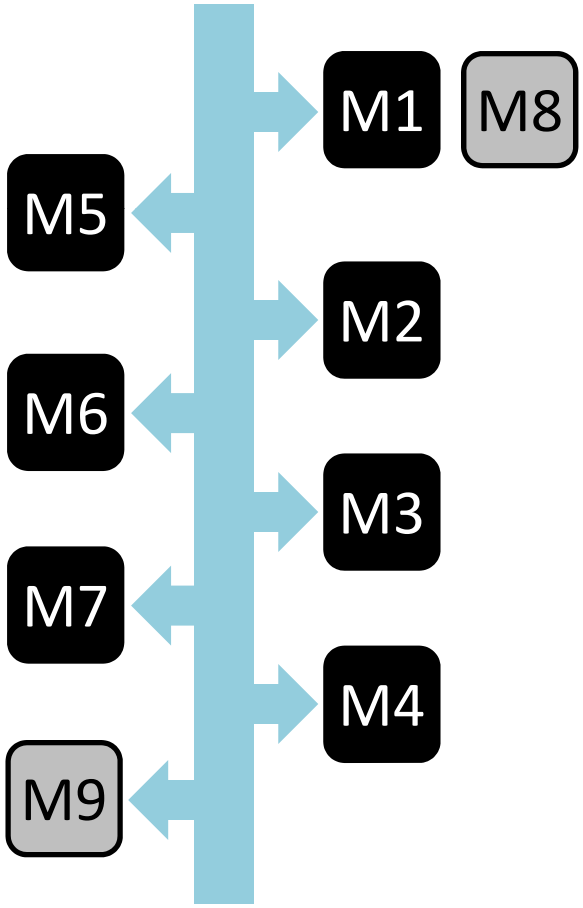
- Wait for notification?

Distributed management: Who manages M9?



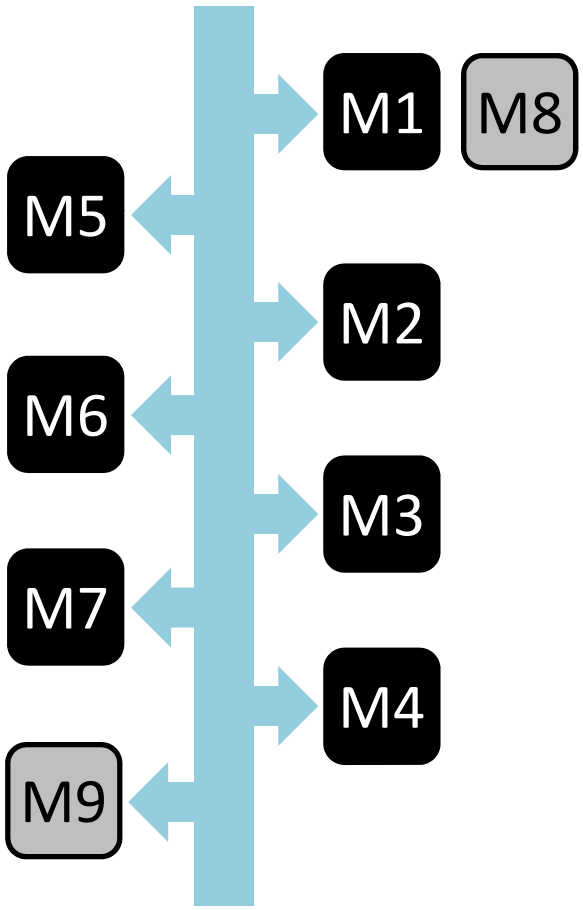
- Wait for notification?
  - No guarantees before sleep
  - M1 failure abandons M8

Distributed management: Who manages M9?



- Wait for notification?
  - No guarantees before sleep
  - M1 failure abandons M8
- Probe randomly, repeat since machines unreliable
- Load-sensitive machines, so distribute probing
  - Robust to manager issues

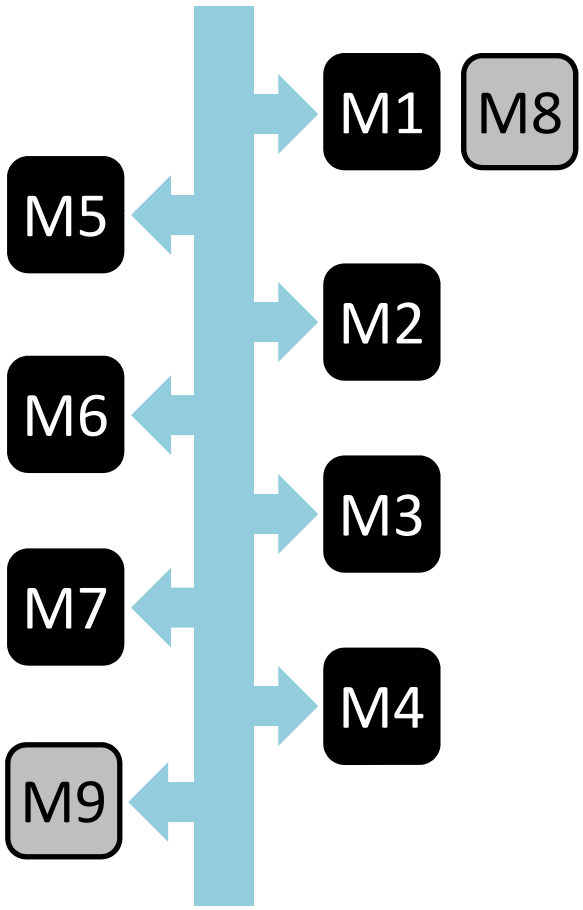
Distributed management: Who manages M9?



total # machines

$$\frac{n}{\#\text{ awake machines}}$$

Distributed management: Who manages M9?

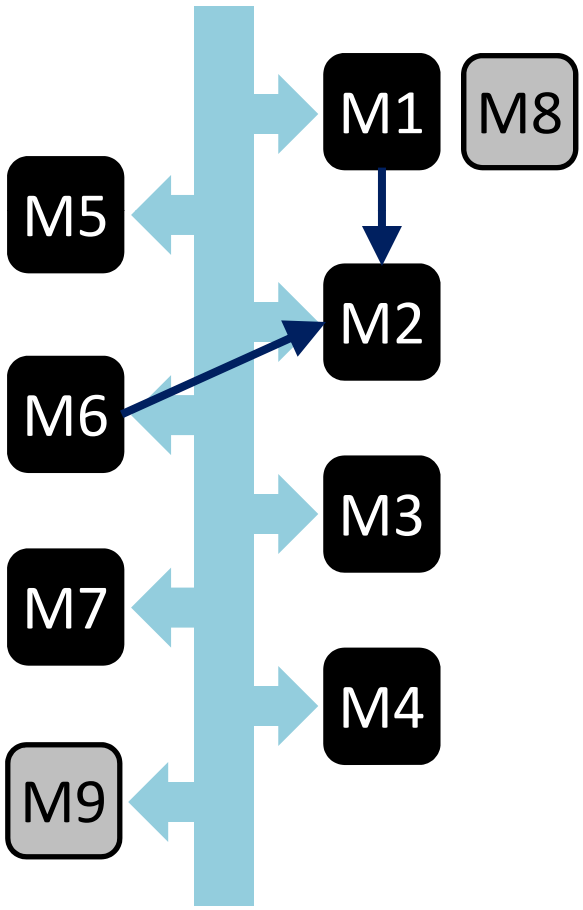


total # machines

managed by *i*

$$\frac{n - m_i}{\#\ \text{awake machines}}$$

Distributed management: Who manages M9?

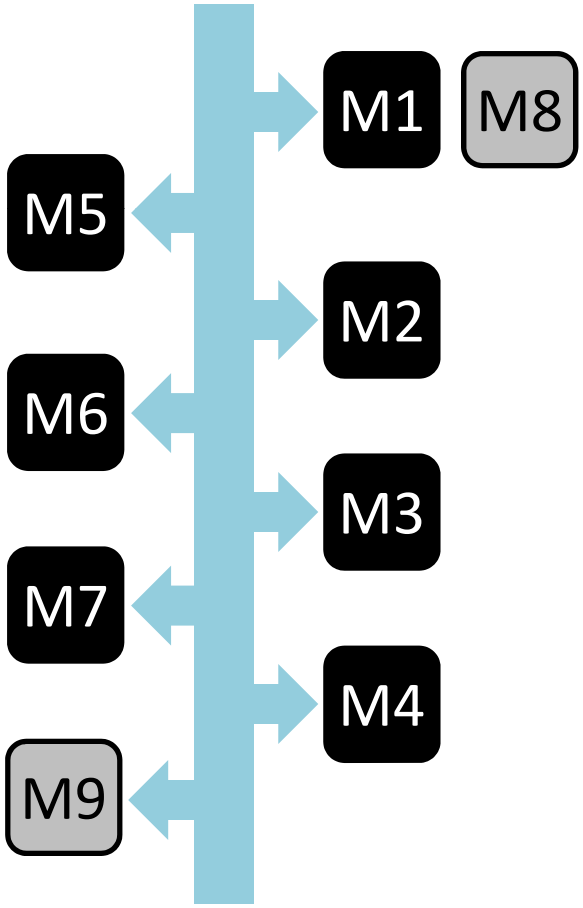


total # machines

managed by *i*

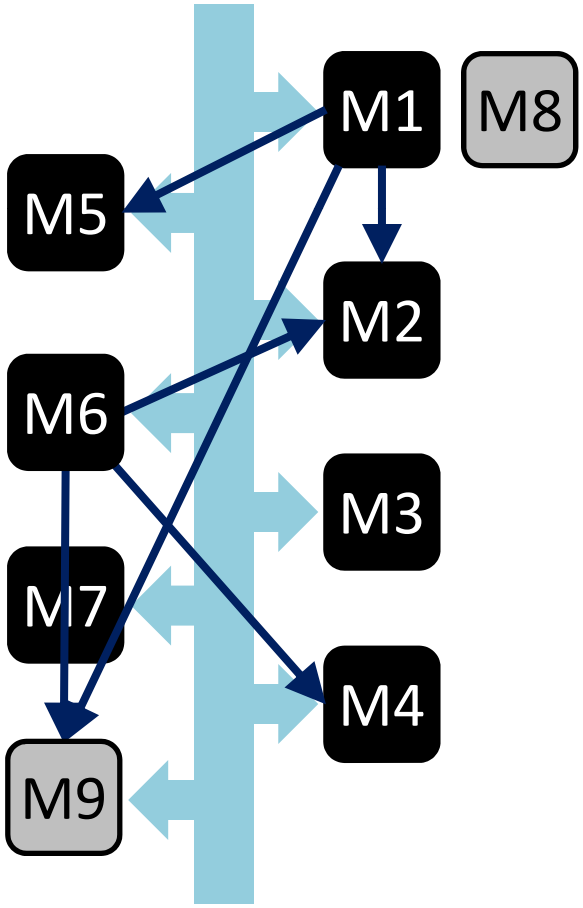
$$\frac{n - m_i}{\#\text{ awake machines}}$$

Distributed management: Who manages M9?



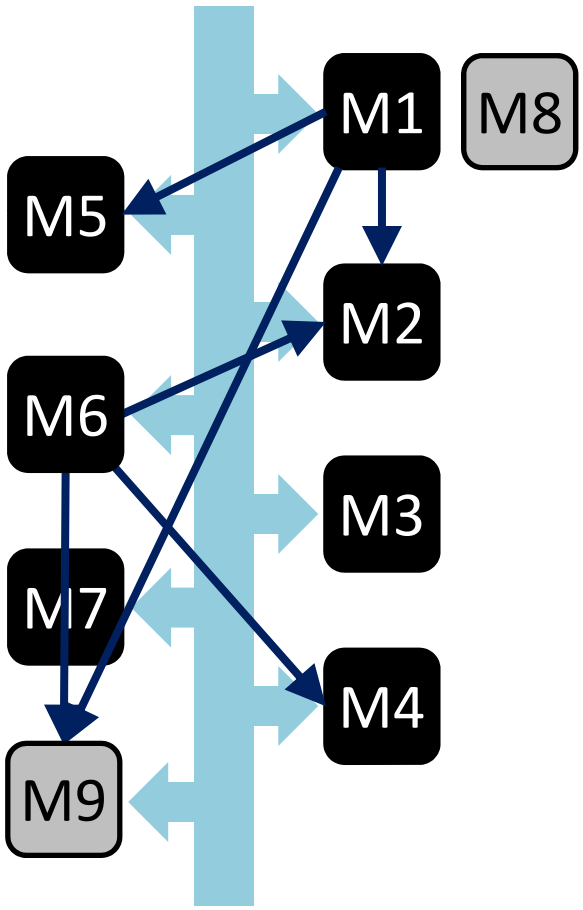
$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

Distributed management: Who manages M9?



$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

Distributed management: Who manages M9?

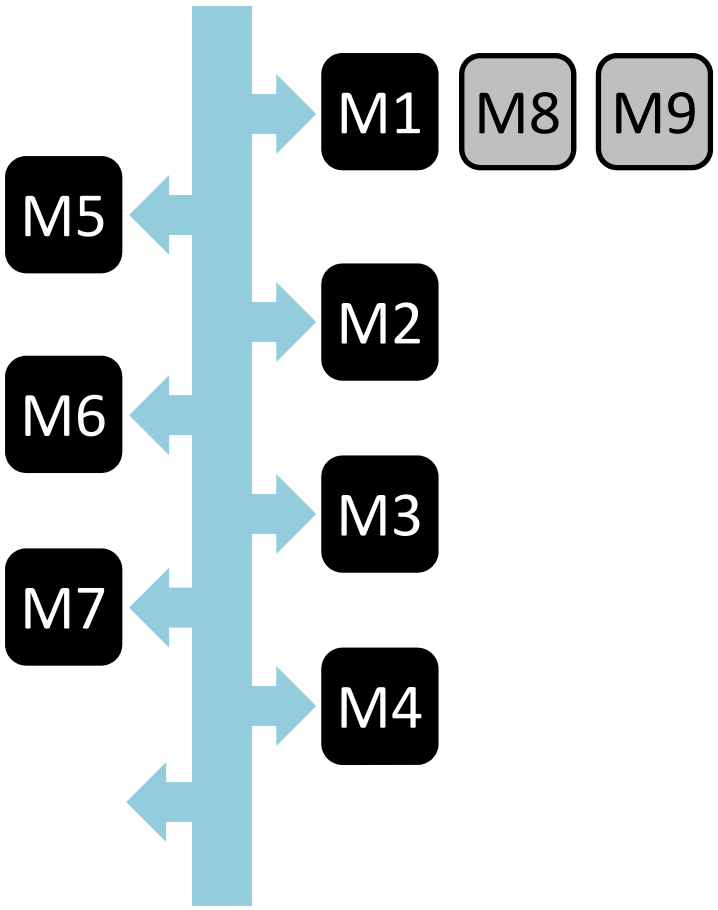


p = Pr(machine probed)

$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

- Coupon collector analysis

Distributed management: Who manages M9?

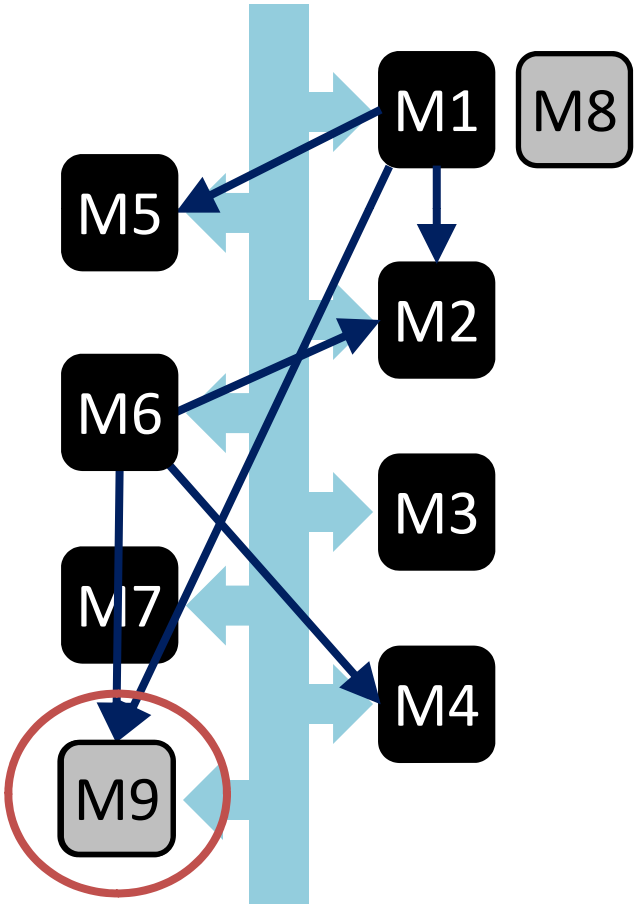


p = Pr(machine probed)

$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

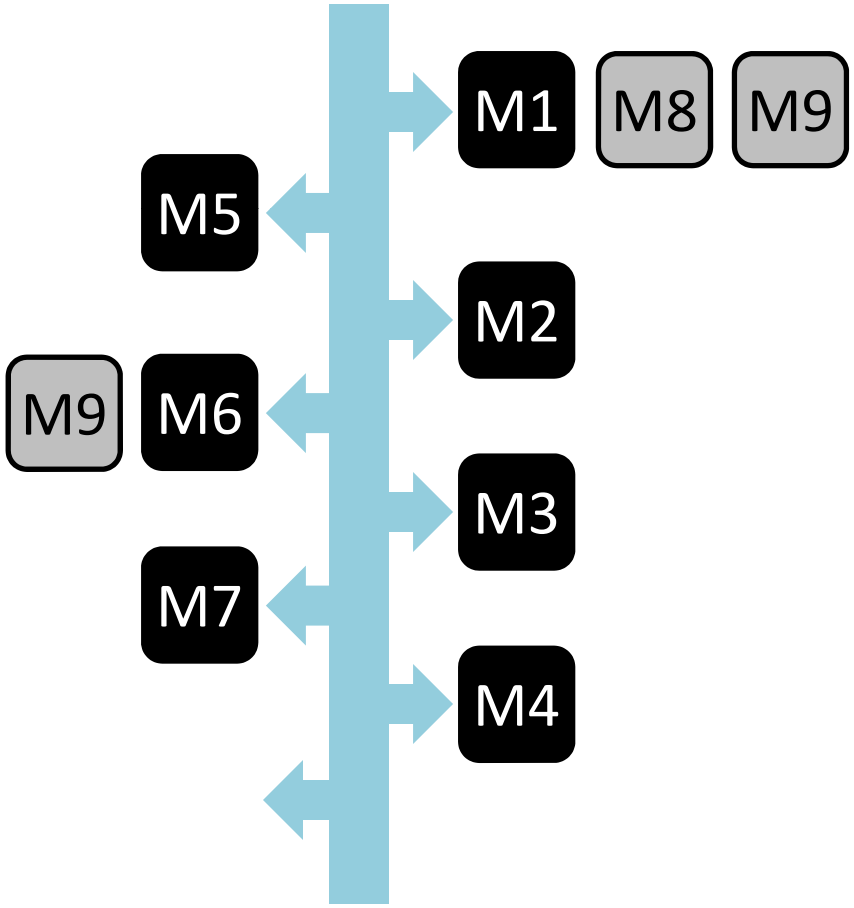
- Coupon collector analysis

Multiple managers



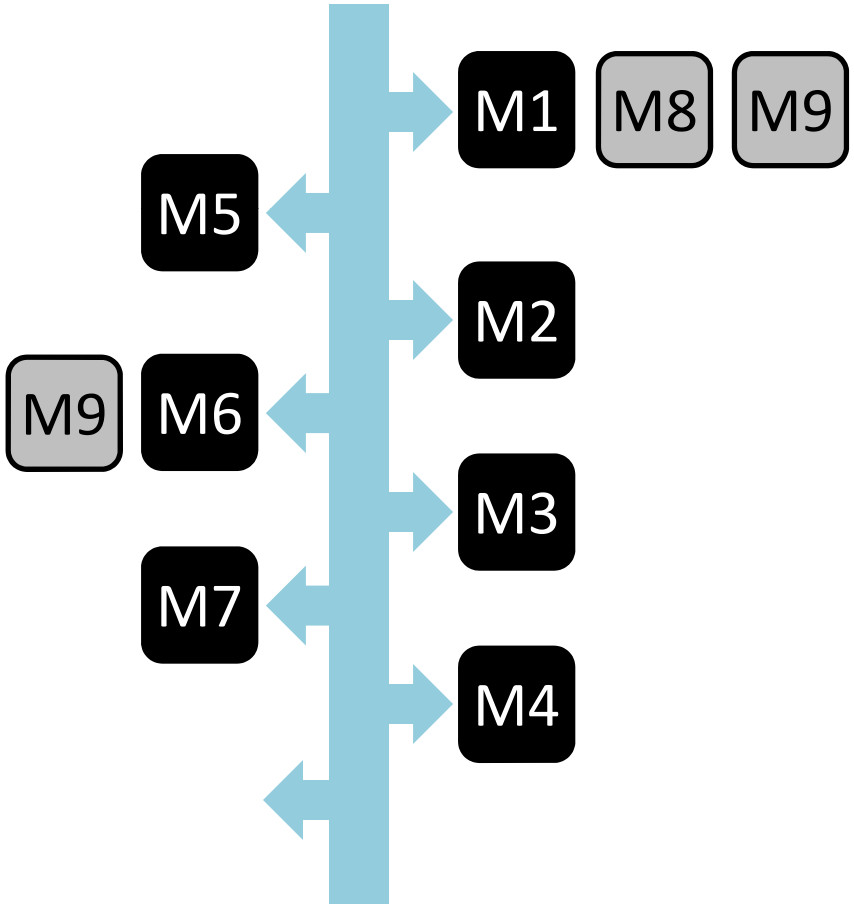
$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

Multiple managers



$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

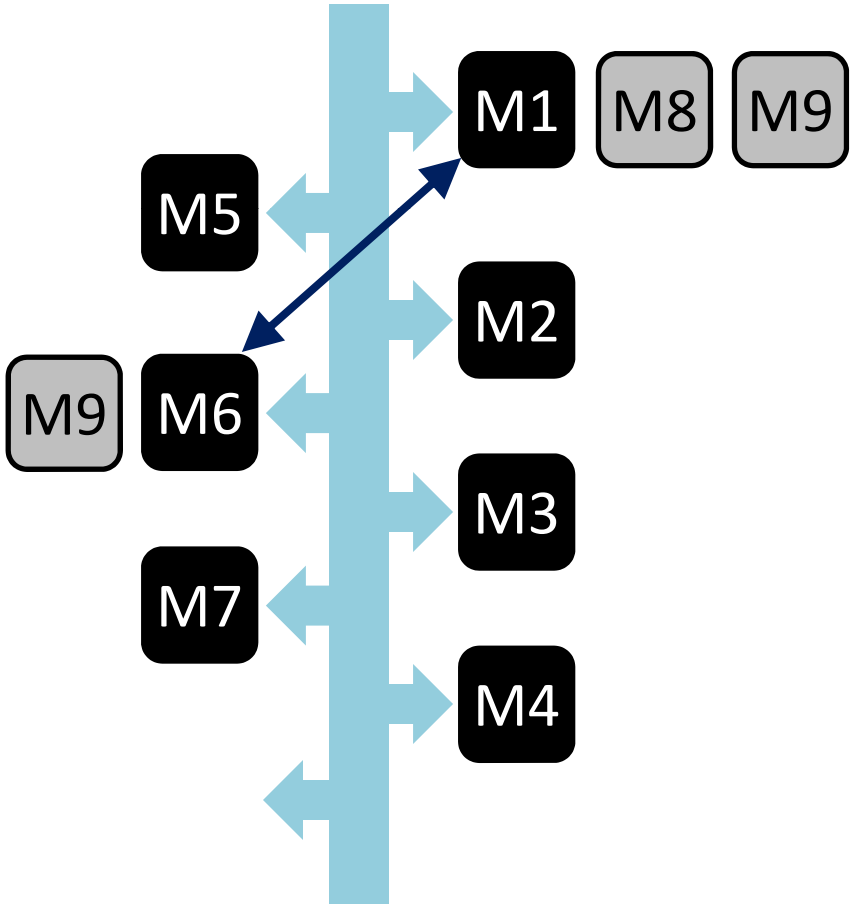
Multiple managers



$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

- Availability most important

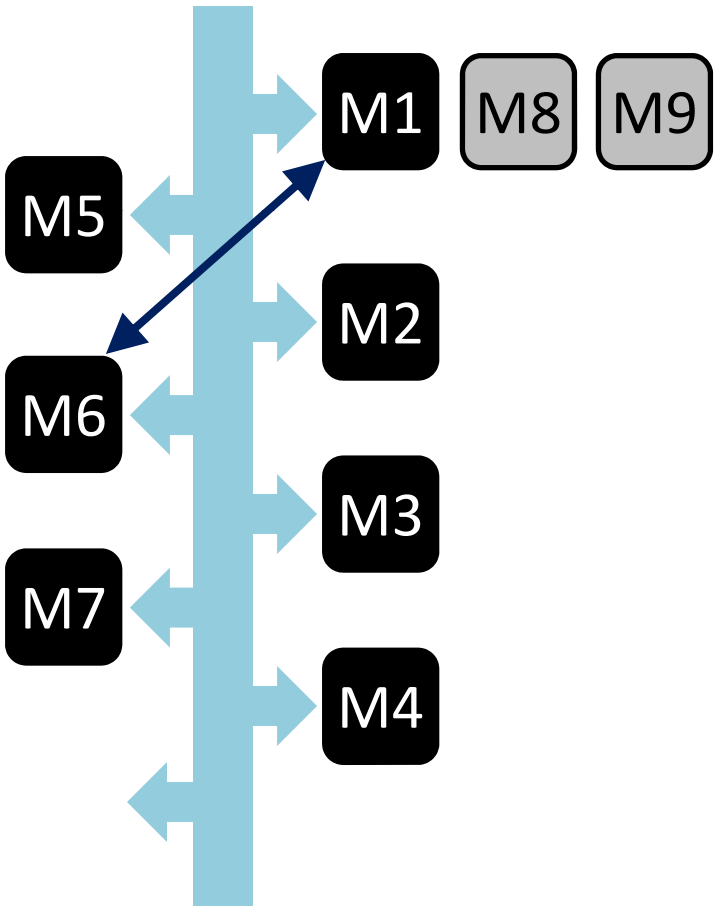
Multiple managers



$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

- Availability most important
- Simple resolution protocol

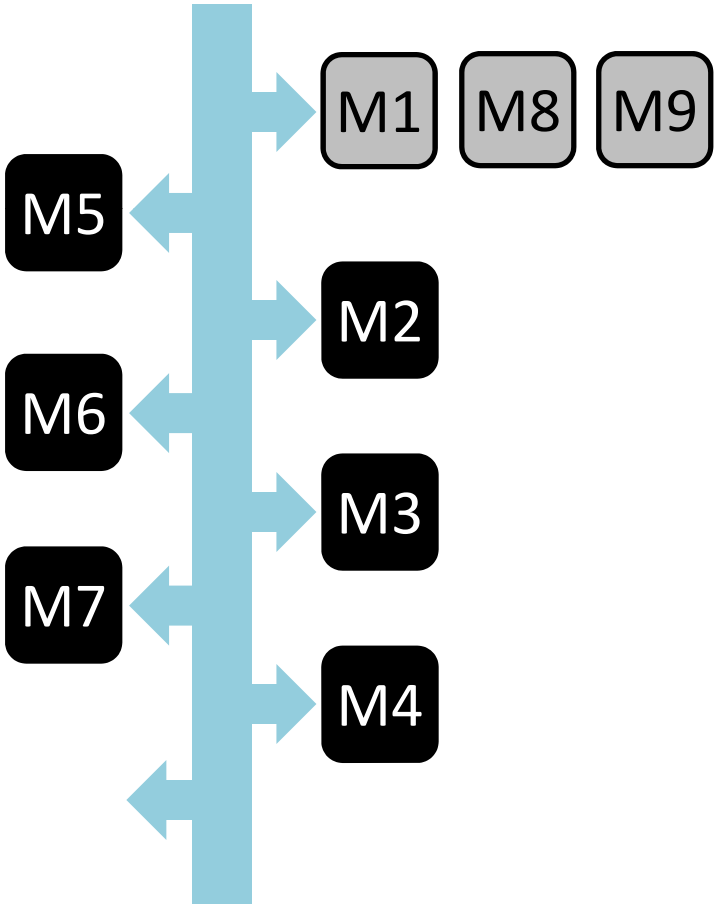
Multiple managers



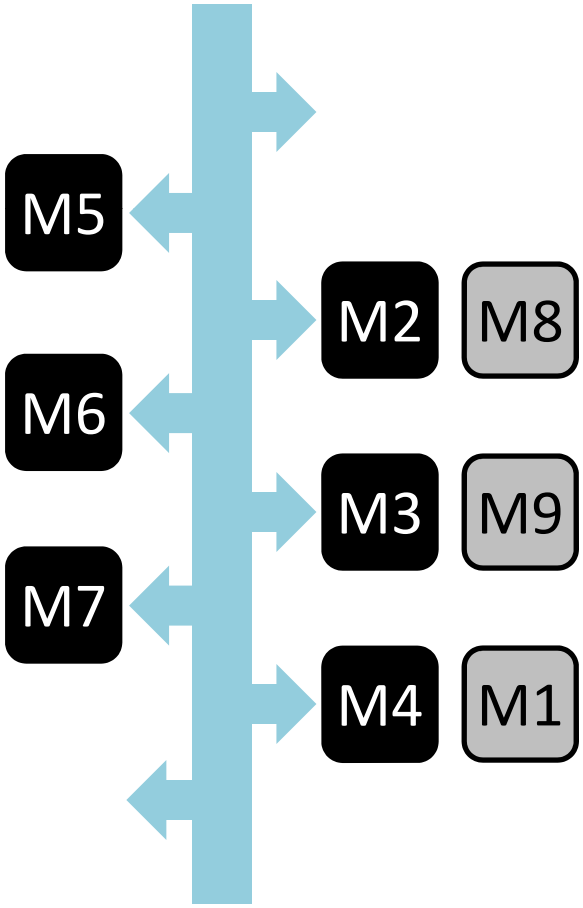
$$\frac{(n - m_i)\ln\left(\frac{1}{1-p}\right)}{\#\text{ awake machines}}$$

- Availability most important
- Simple resolution protocol

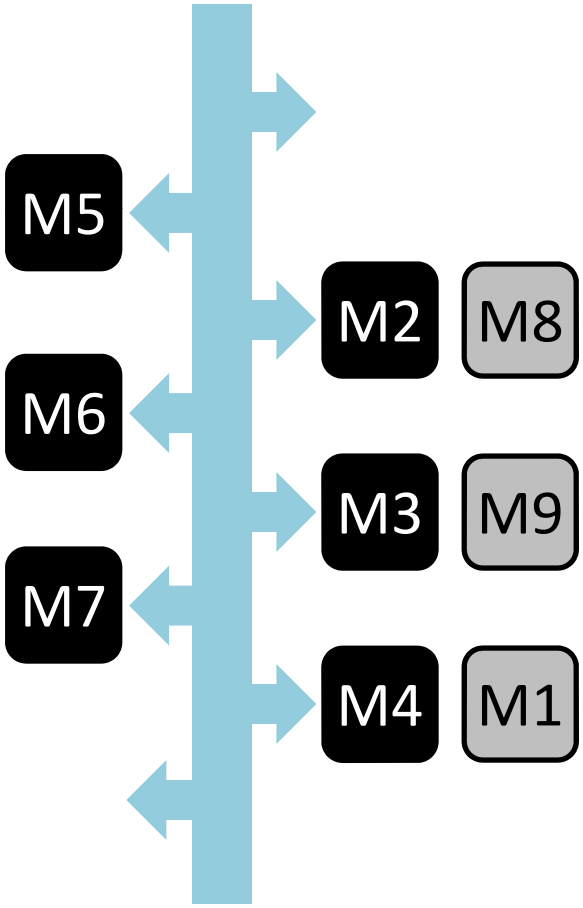
Load balance



Load balance



Load balance



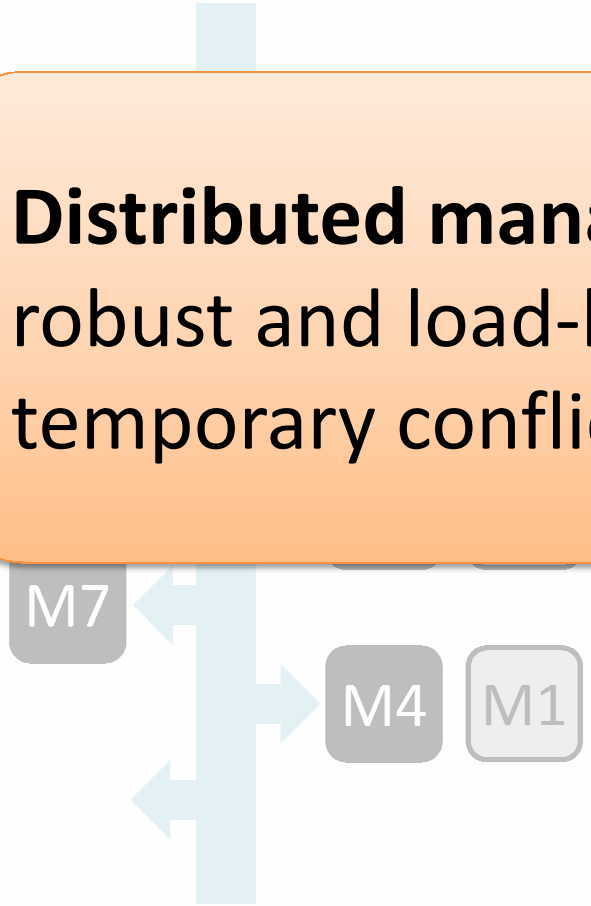
- Induction analysis: equivalent to balls-in-bins!

$$\frac{\ln(n/2)}{\ln\ln(n/2)}$$ after $n/2$ sleeps

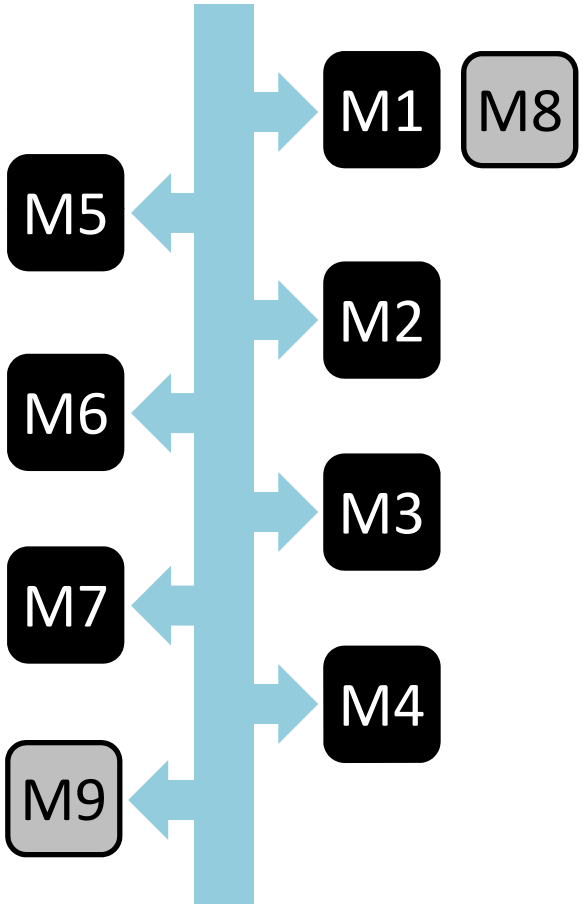
Load balance

- Induction analysis:

Distributed management elects leaders in a robust and load-balanced way, assuming temporary conflicts are tolerable.

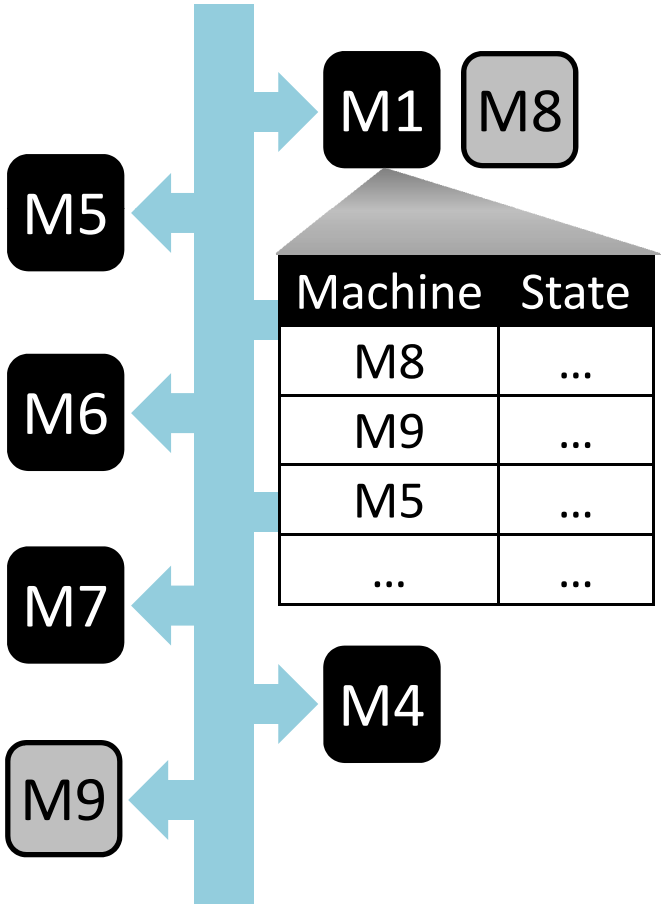


Subnet state coordination



- Distributed management relies on global state
  - Who to probe?
  - How to manage?

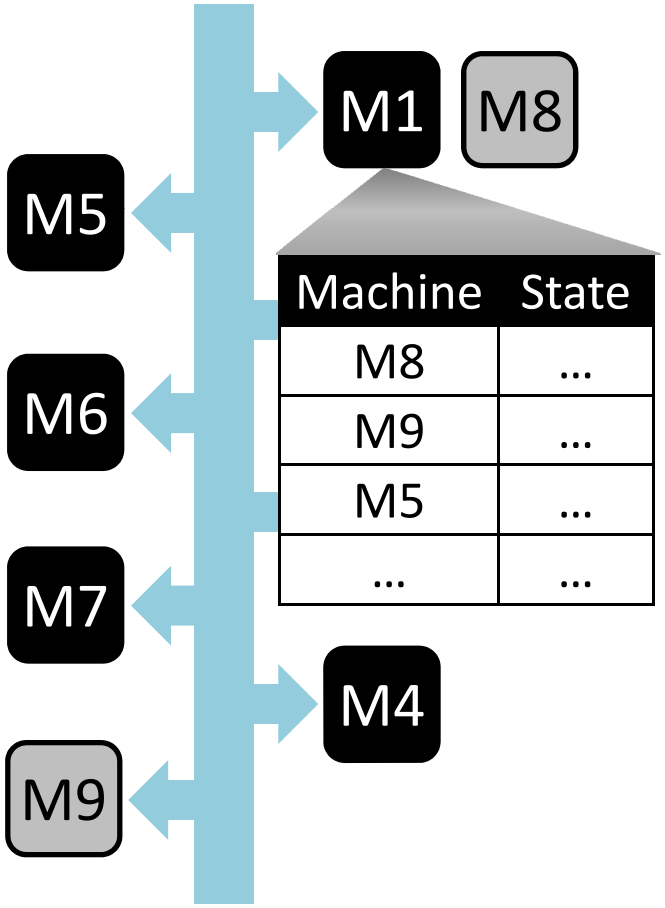
Subnet state coordination



- Distributed management relies on global state
  - Who to probe?
  - How to manage?

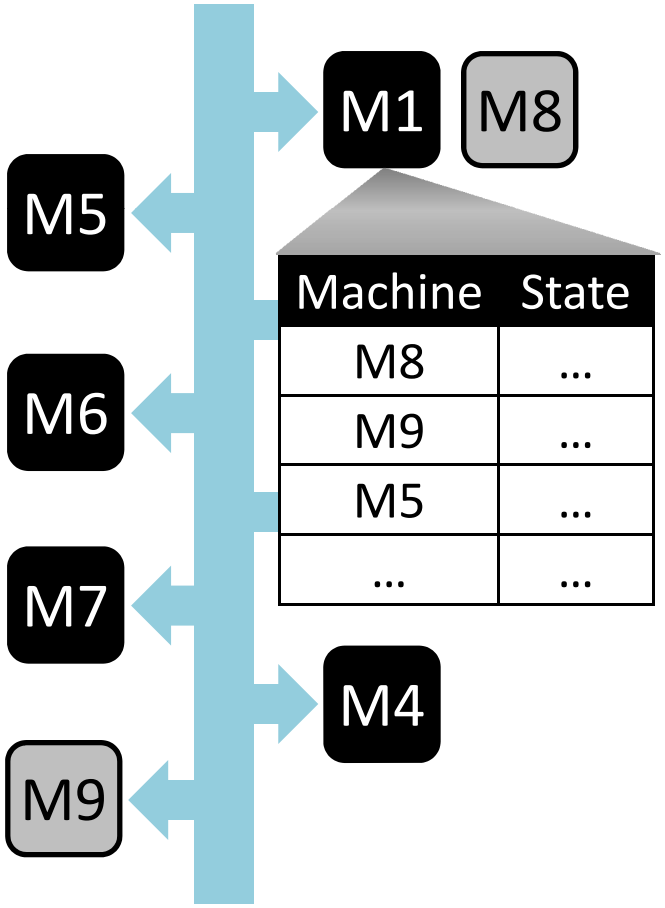
- IP address, MAC address
- TCP listen ports

Subnet state coordination



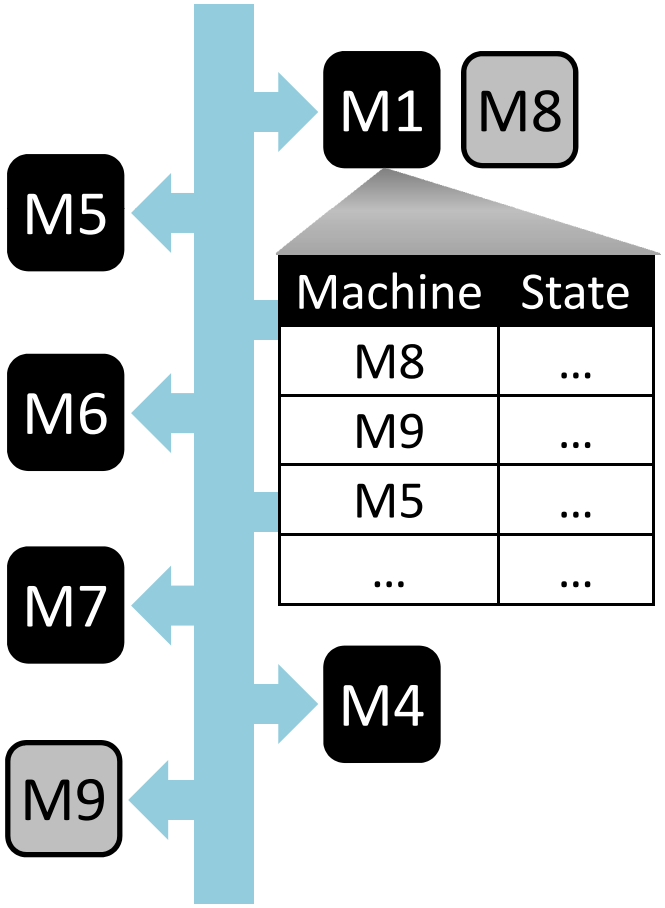
- Replicated state machine?
  - Unreliable machines, correlated behavior
  - Strong consistency overkill

Subnet state coordination



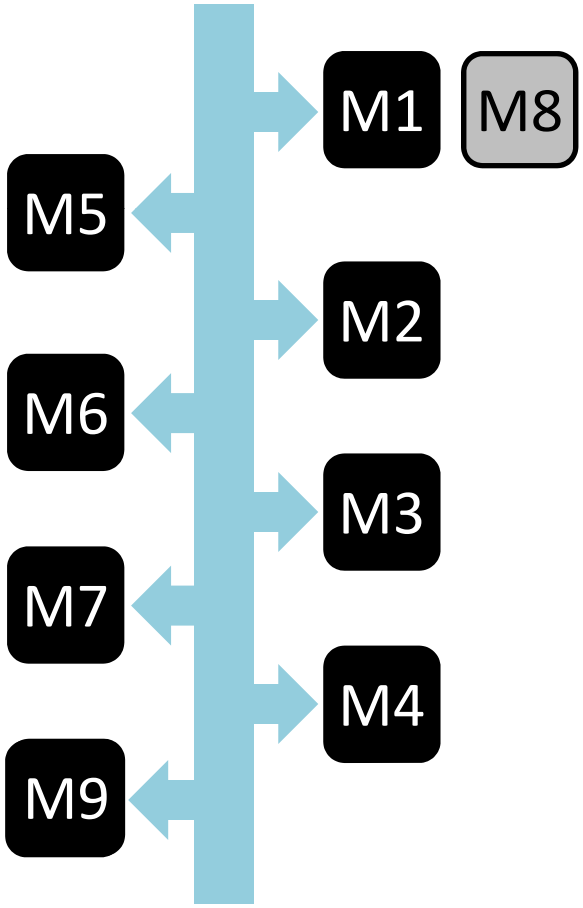
- Replicated state machine?
  - Unreliable machines, correlated behavior
  - Strong consistency overkill
- External database?
  - Lose instant deployability

Subnet state coordination



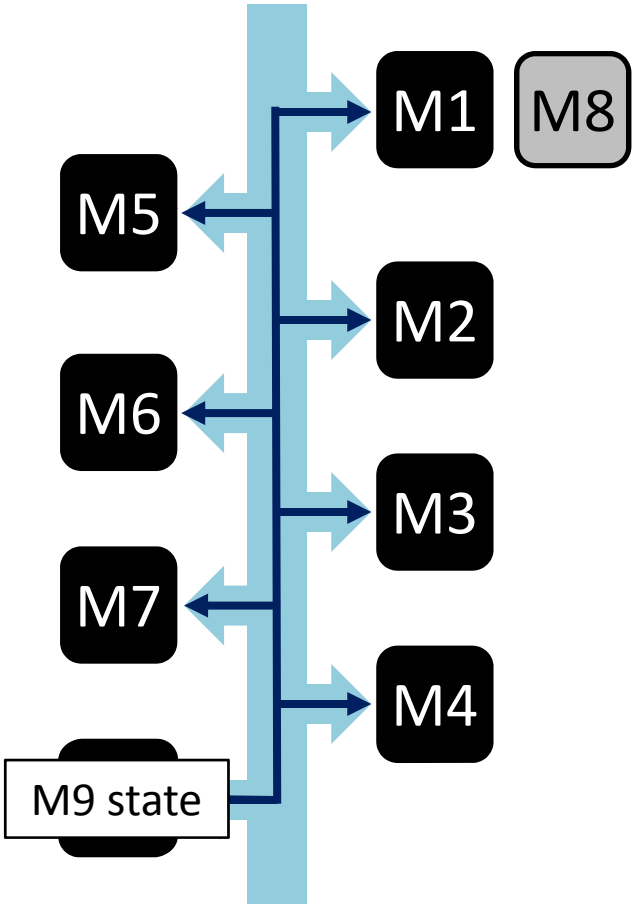
- Replicated state machine?
  - Unreliable machines, correlated behavior
  - Strong consistency overkill
- External database?
  - Lose instant deployability
- Exploit subnet and weaker consistency

Subnet state coordination



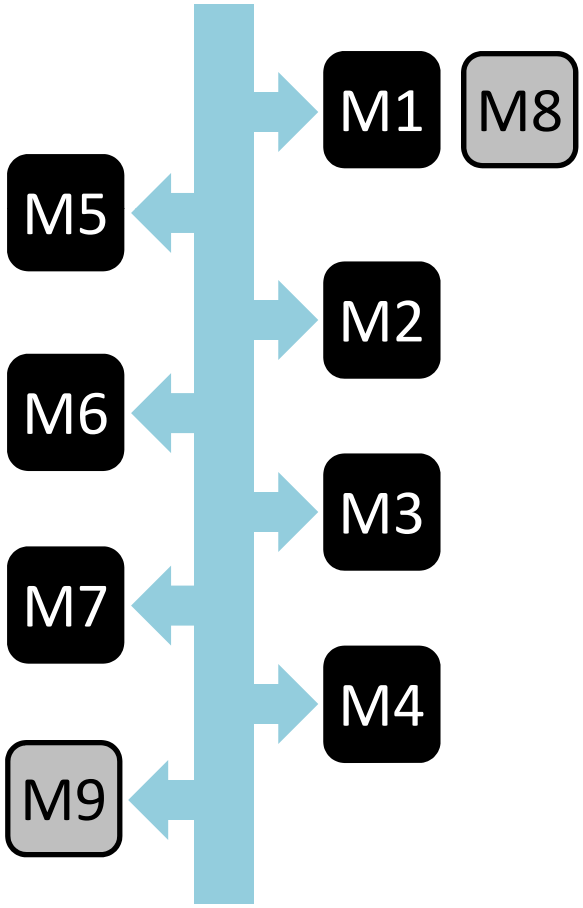
1. Periodic broadcast while awake

Subnet state coordination



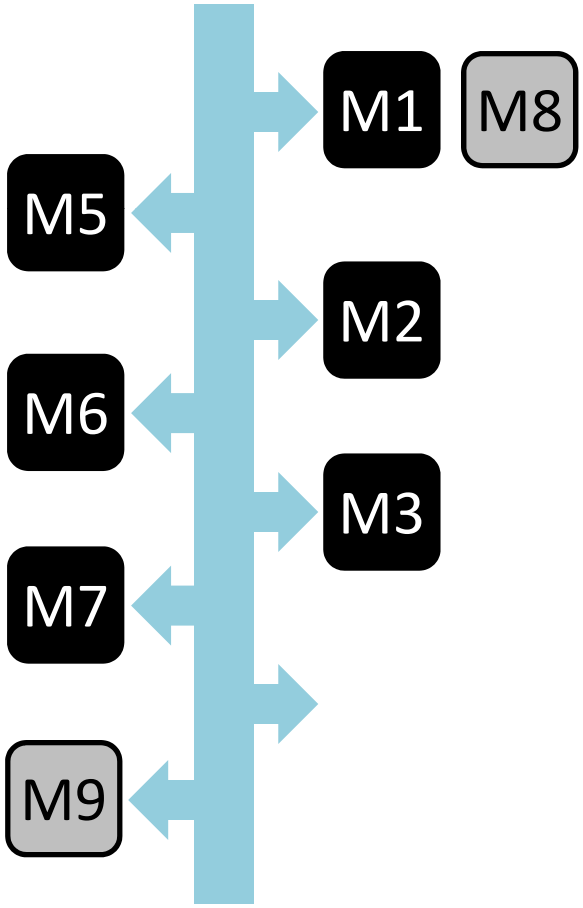
1. Periodic broadcast while awake

Subnet state coordination



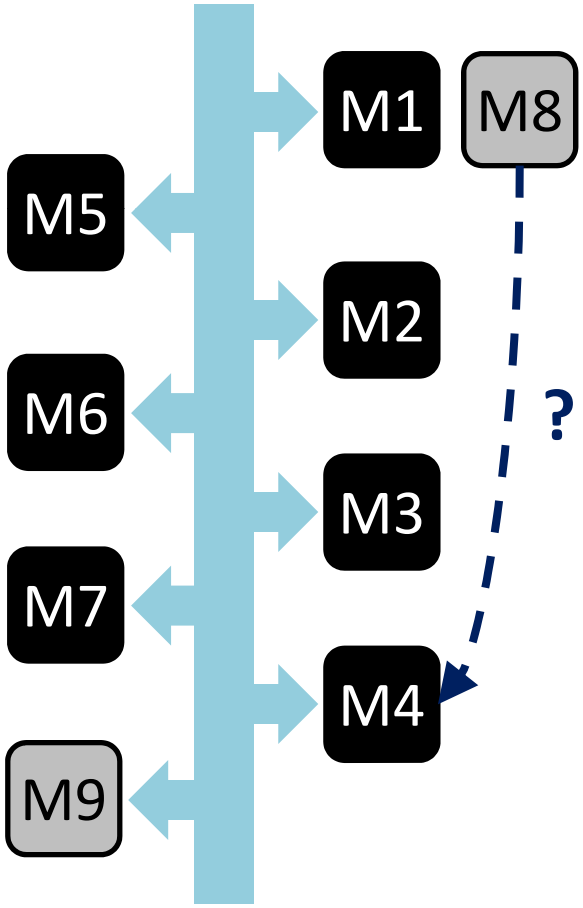
1. Periodic broadcast while awake

Subnet state coordination



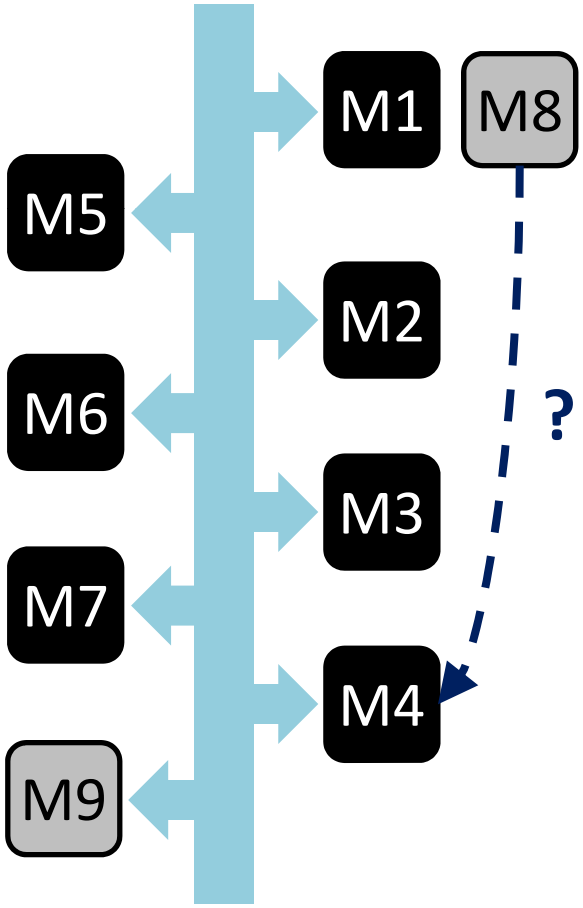
1. Periodic broadcast while awake

Subnet state coordination



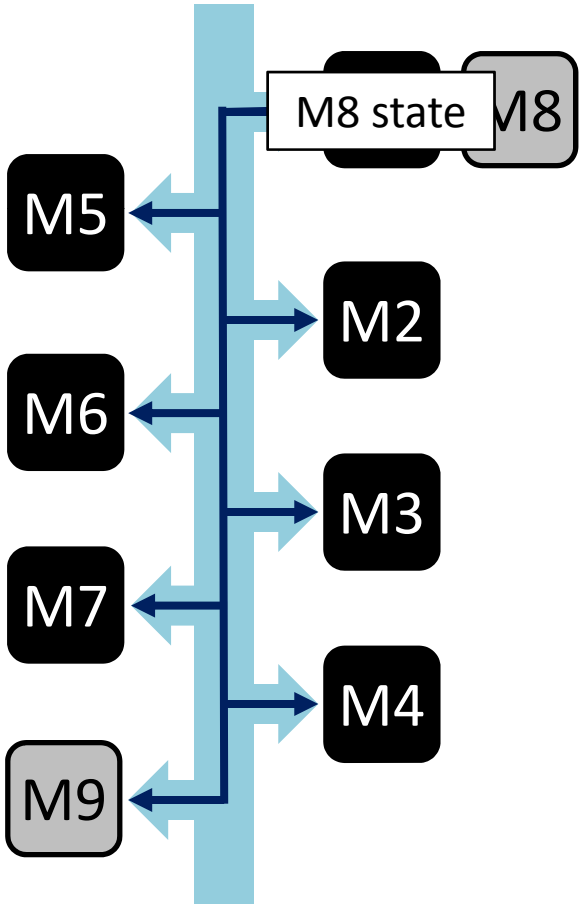
1. Periodic broadcast while awake

Subnet state coordination



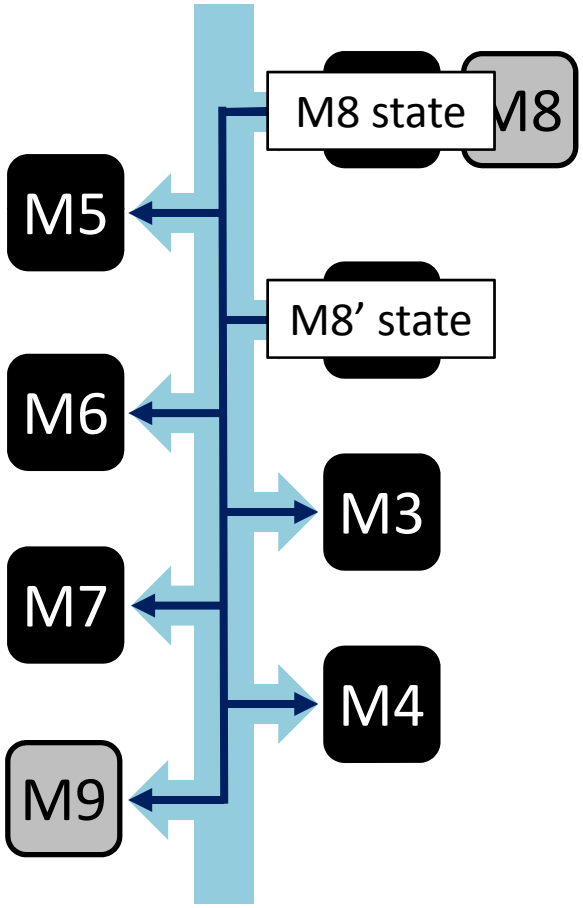
1. Periodic broadcast while awake
2. Rebroadcast by managers while asleep

Subnet state coordination



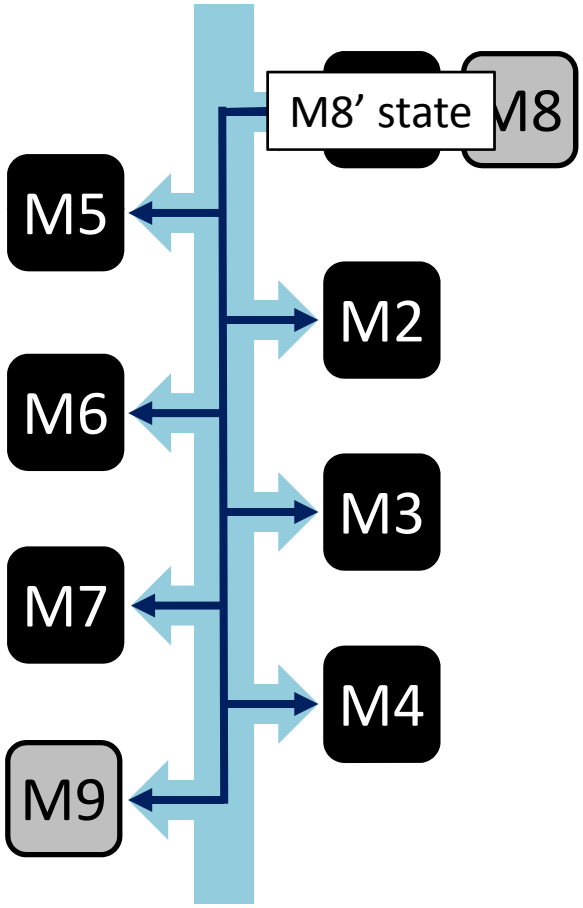
1. Periodic broadcast while awake
2. Rebroadcast by managers while asleep

Subnet state coordination



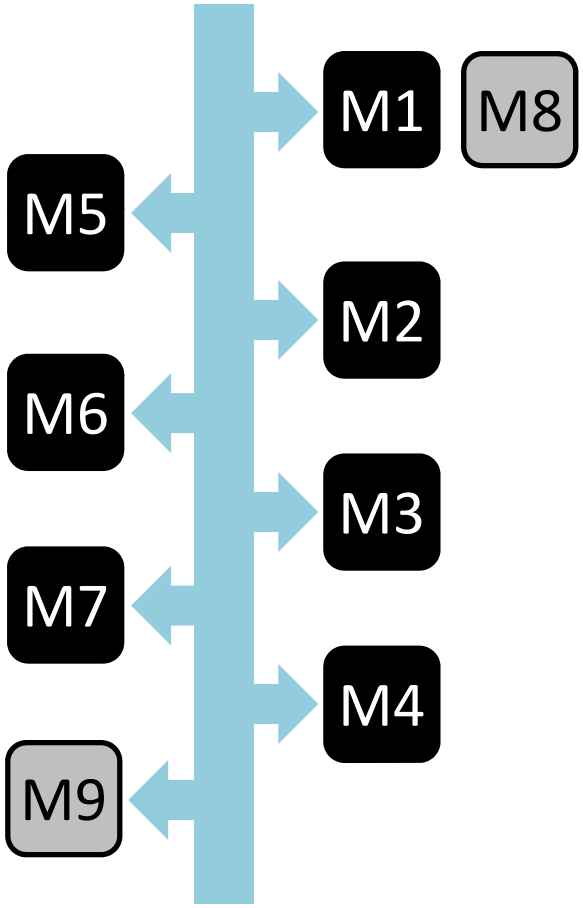
1. Periodic broadcast while awake
2. Rebroadcast by managers while asleep

Subnet state coordination



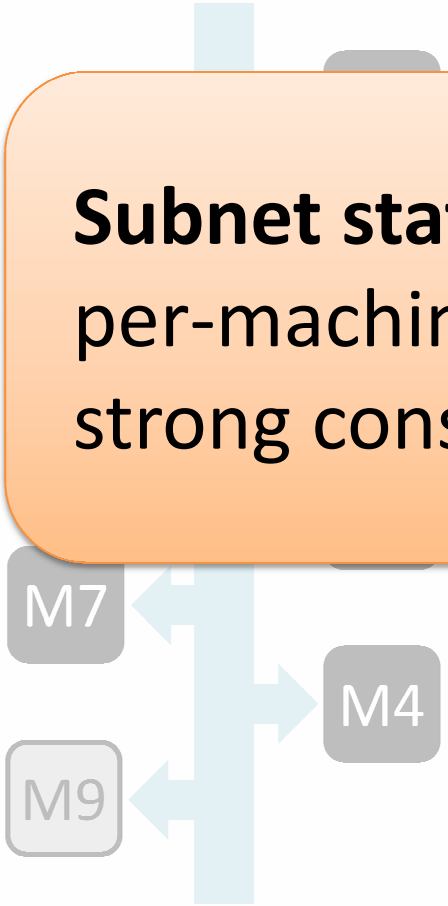
1. Periodic broadcast while awake
2. Rebroadcast by managers while asleep

Subnet state coordination



1. Periodic broadcast while awake
2. Rebroadcast by managers while asleep
3. Daily roll call to garbage-collect state

Subnet state coordination



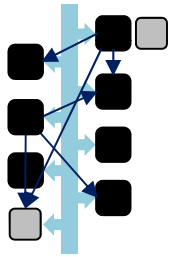
Subnet state coordination distributes per-machine state on a subnet when strong consistency is not required.

eep

3. Daily roll call to garbage-collect state

Outline

1. How does GreenUp work?
2. What can I learn from GreenUp?



Distributed management

Machine	State
…	…
…	…
…	…

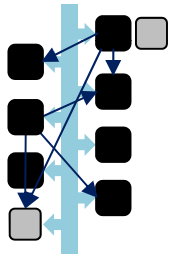
Subnet state coordination

Guardians

3. How effective is GreenUp?
   - Evaluation on ~**100** user machines, currently deployed on ~**1,100** machines

Outline

1. How does GreenUp work?
2. What can I learn from GreenUp?



Distributed management

- Protects against simultaneous sleep
- Caps the max load

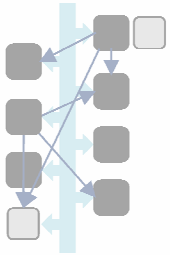
Guardians

coordination

3. How effective is GreenUp?
   - Evaluation on ~**100** user machines, currently deployed on ~**1,100** machines

Outline

1. How does GreenUp work?
2. What can I learn from GreenUp?



Distributed management

Machine	State
…	…
…	…
…	…

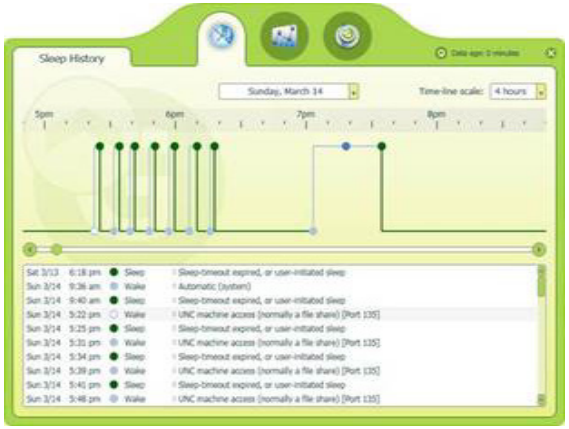
Subnet state coordination

Guardians

3. How effective is GreenUp?
   - Evaluation on ~**100** user machines, currently deployed on ~**1,100** machines

Deployment in Microsoft

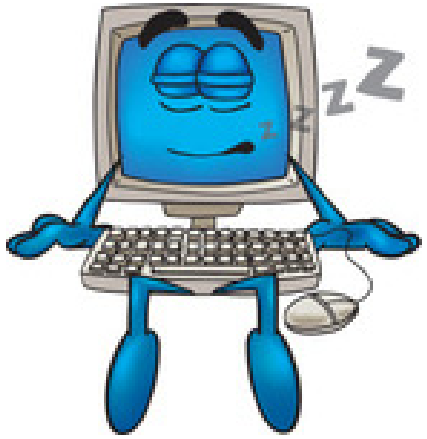
- C# code
  - Interfaces with packet sniffer/network driver
- Client GUI for users and easy deployment
- Pilot on ~1,100 machines



Evaluation

- Logs from 101 Windows 7 machines, Feb. – Sep. 2011
- Questions:
  - Does GreenUp consistently wake machines when accessed?
  - Does it do so in time to meet user patience?
  - Can GreenUp scale to large subnets?

GreenUp wakes machines reliably



GreenUp wakes machines reliably

- Connect to machines using Samba (TCP port 139)
- 11 different days (weekends, evenings):
  - 496 already awake, 278 woken, 5 unwakeable
  - Most failures due to WoL
- 99.4% success rate

GreenUp wakes machines reliably

- Connect to machines using Samba (TCP port 139)
- 11 different da
  (weekends, ev
  - 496 already a
    woken, 5 unwakeable
  - Most failures due to WoL
- 99.4% success rate

WoL is availability bottleneck!

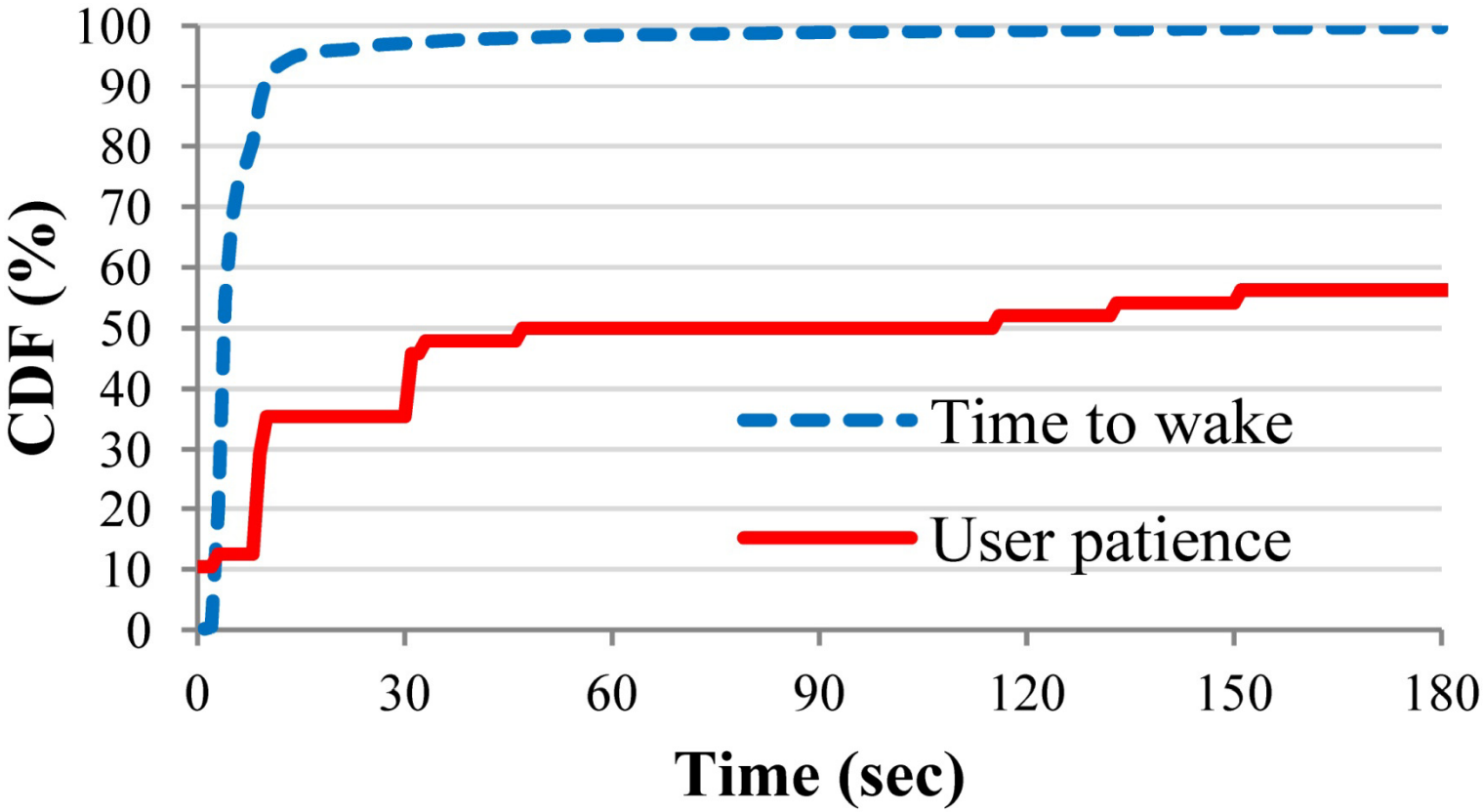
GreenUp wakes machines quickly

- GreenUp relies on *some* user patience
  - Wakeup delay
  - User retry logic

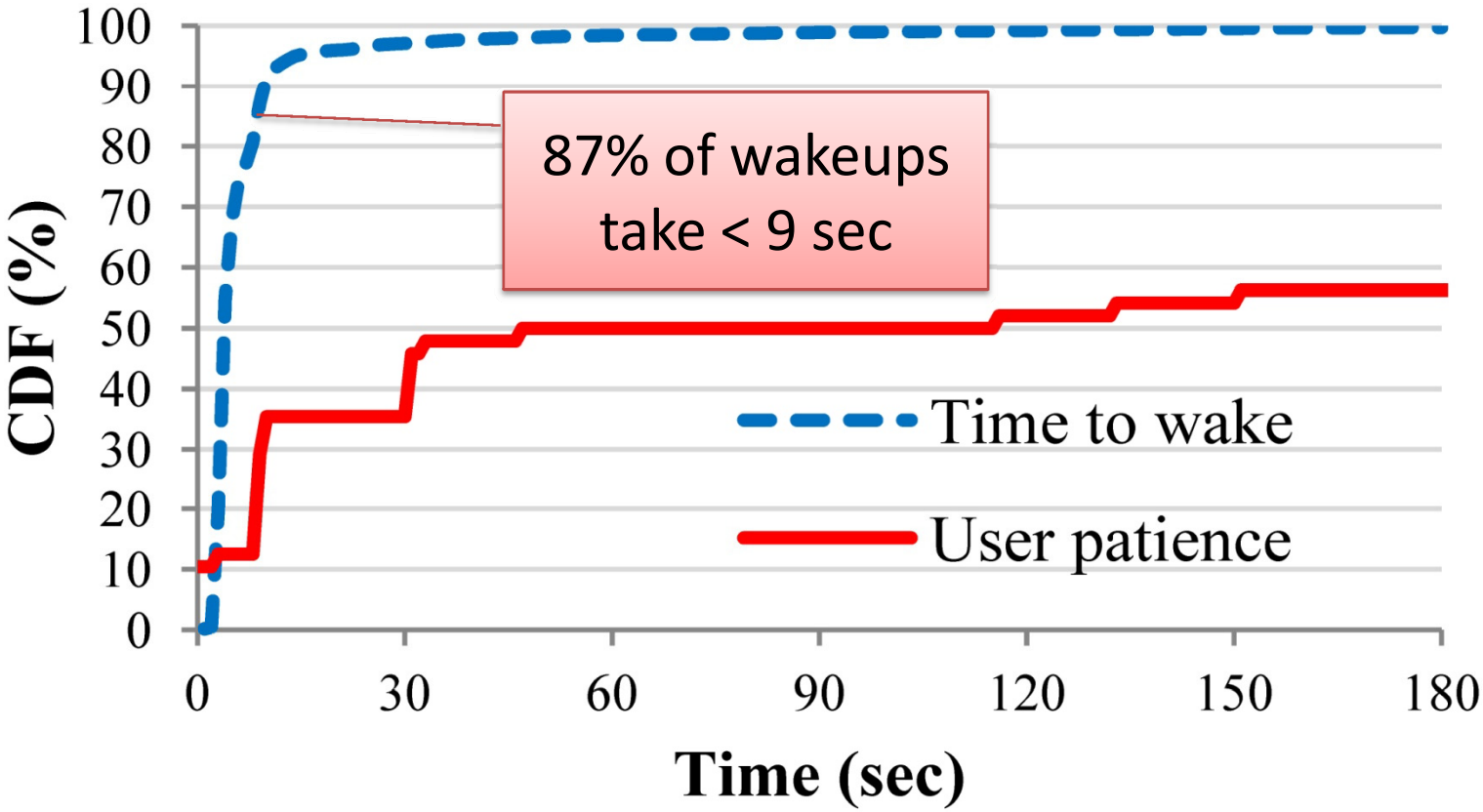
GreenUp wakes machines quickly

- GreenUp relies on *some* user patience
  - Wakeup delay
  - User retry logic
- Side-effect of WoL failure: manager logs how long user waits
  - 48 events

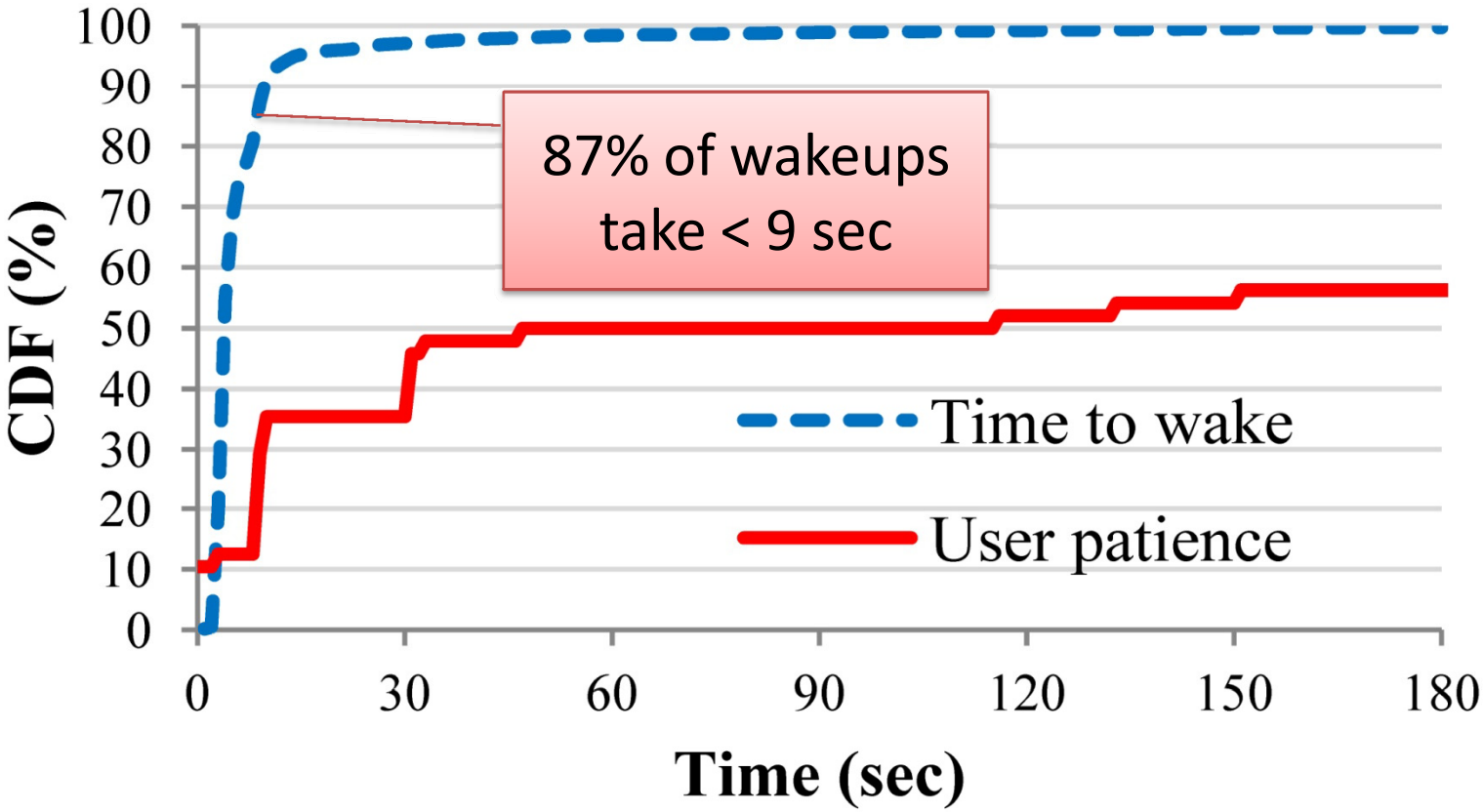
GreenUp wakes machines quickly



GreenUp wakes machines quickly

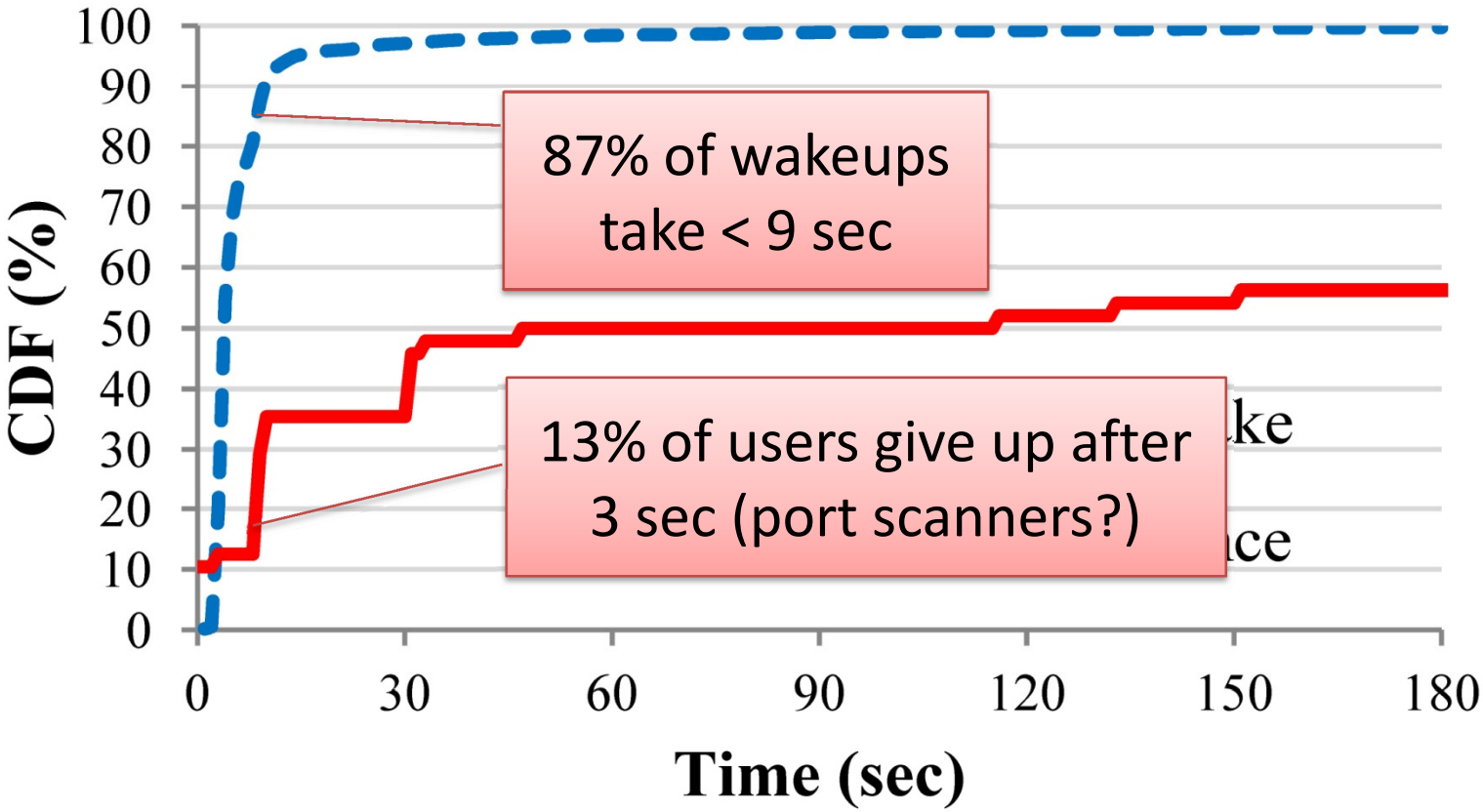


GreenUp wakes machines quickly



- Convolving: GreenUp wakes machines before user gives up 85% of the time

GreenUp wakes machines quickly

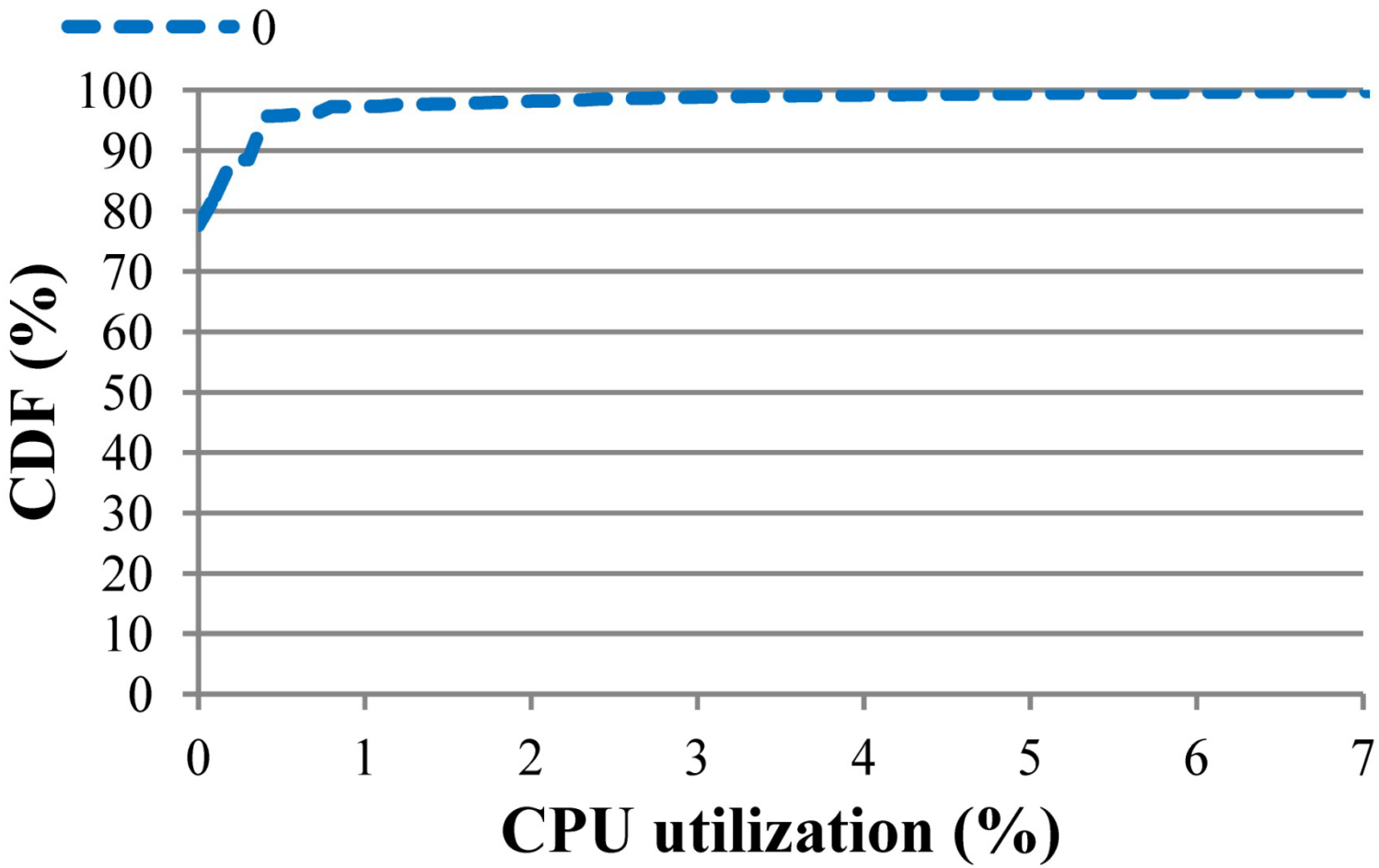


- Convolving: GreenUp wakes machines before user gives up 85% of the time

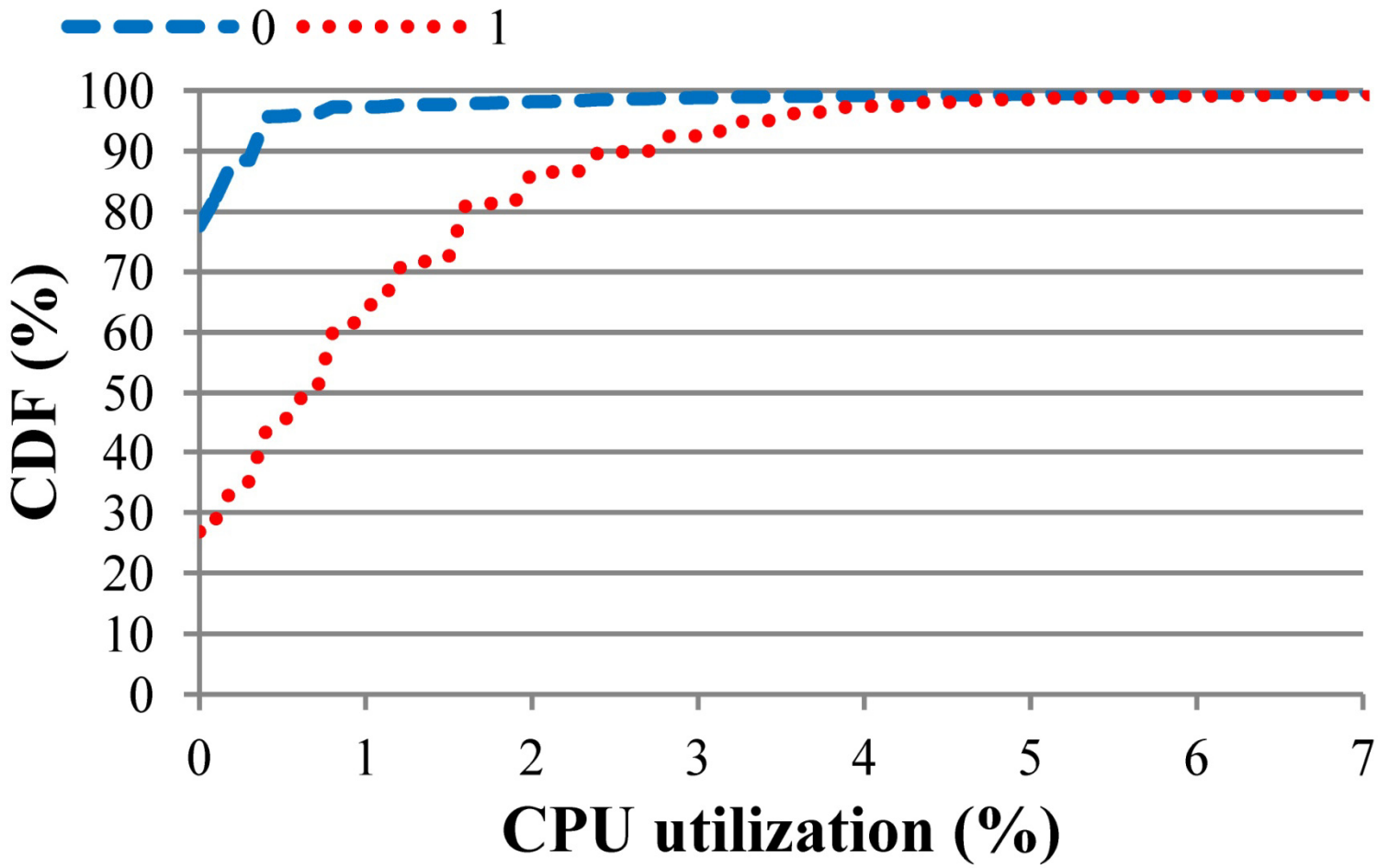
GreenUp scales to large subnets

- Sources of manager load
  - Intercept traffic for asleep machines
  - Broadcast state
  - Probe/respond to probes

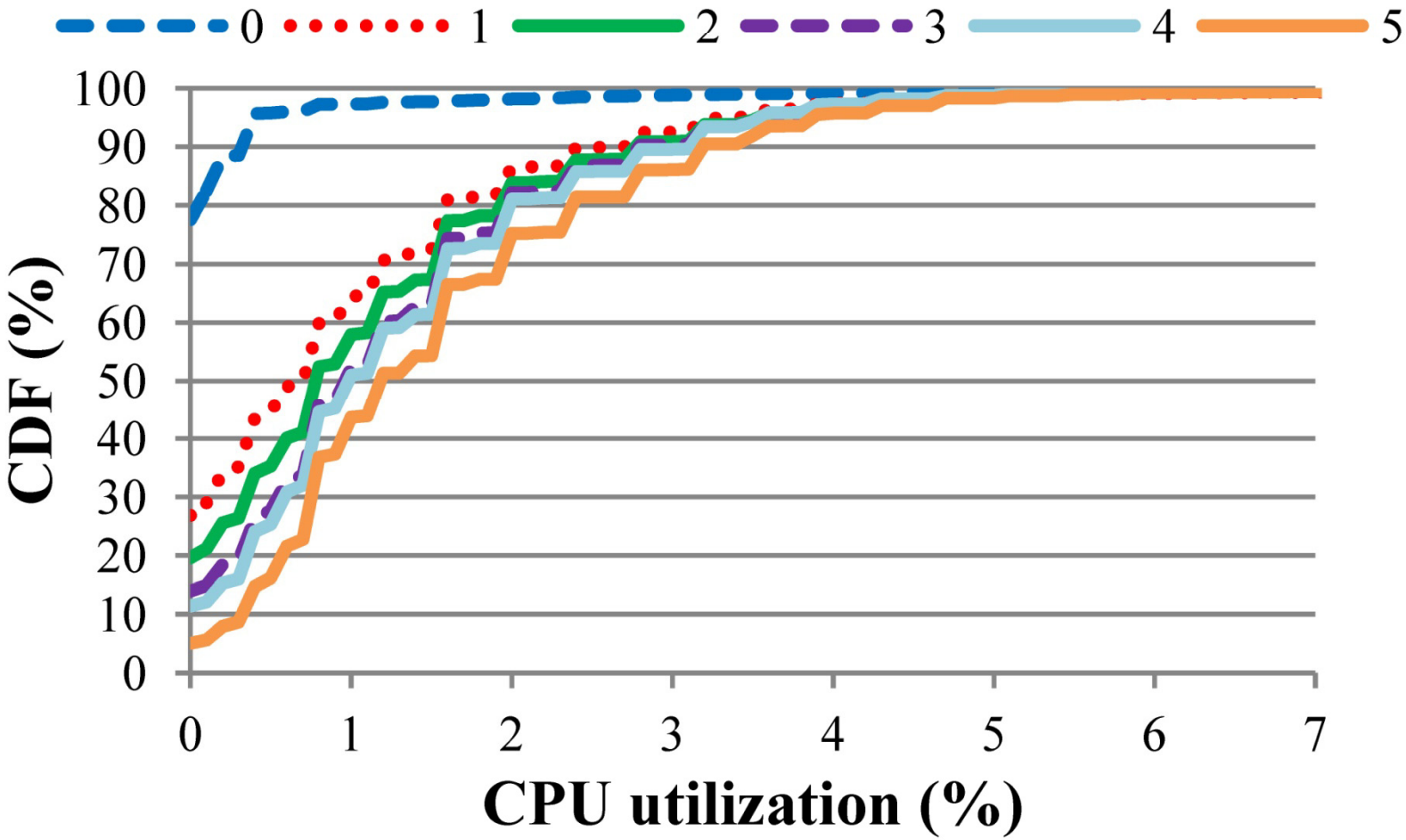
GreenUp scales to large subnets



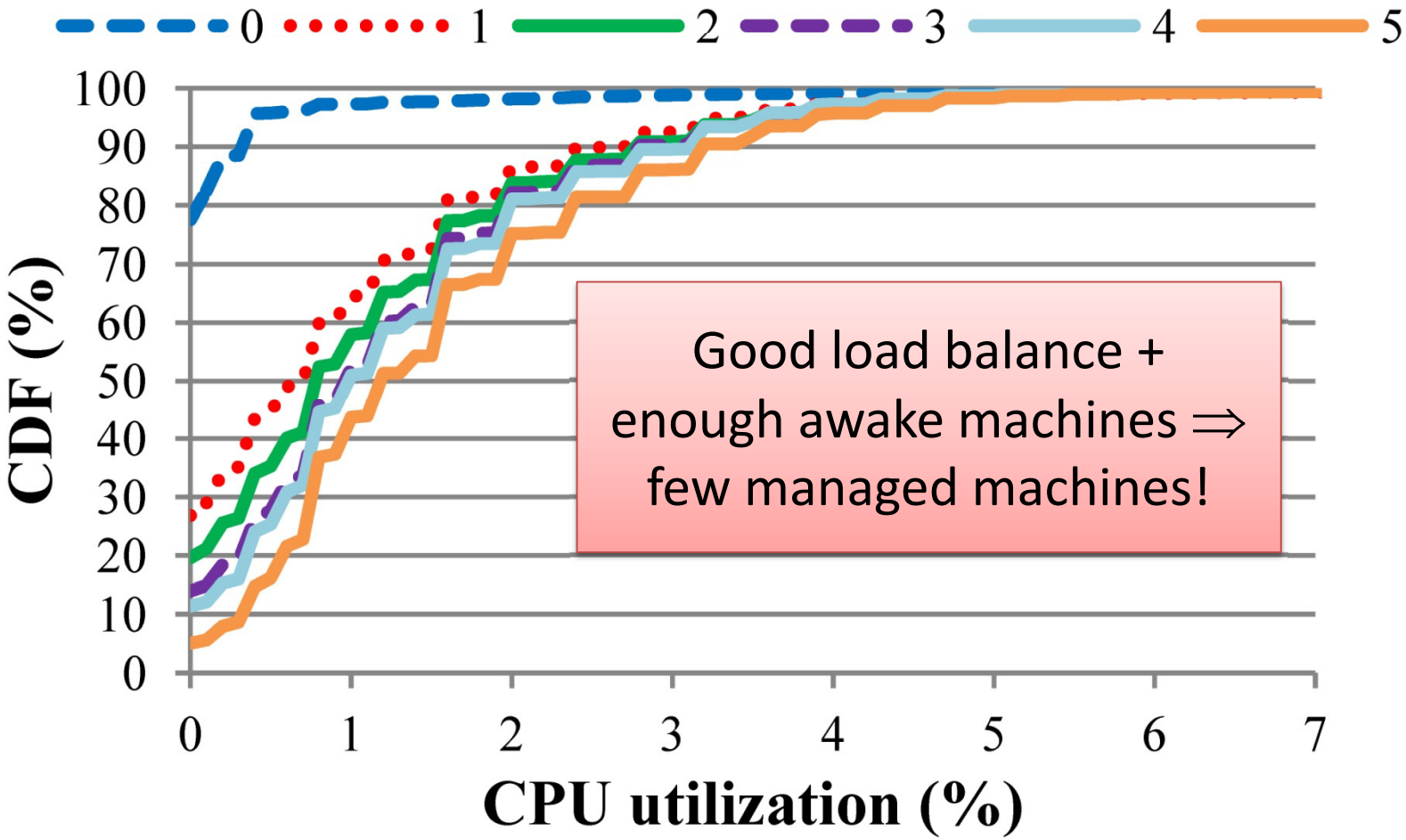
GreenUp scales to large subnets



GreenUp scales to large subnets



GreenUp scales to large subnets



GreenUp scales to large subnets

- Simulated probing load on 2.4-GHz, dual-core Windows 7 machine w/ 4GB memory and 1Gb/s NIC:

# of managed machines	CPU utilization
100	12%
200	21%
300	29%

GreenUp scales to large subnets

- Simulated probing load on 2.4-GHz, dual-core Windows 7 machine w/ 4GB memory and 1Gb/s NIC:

# of managed machines	CPU utilization
100	12%
200	21%
300	29%

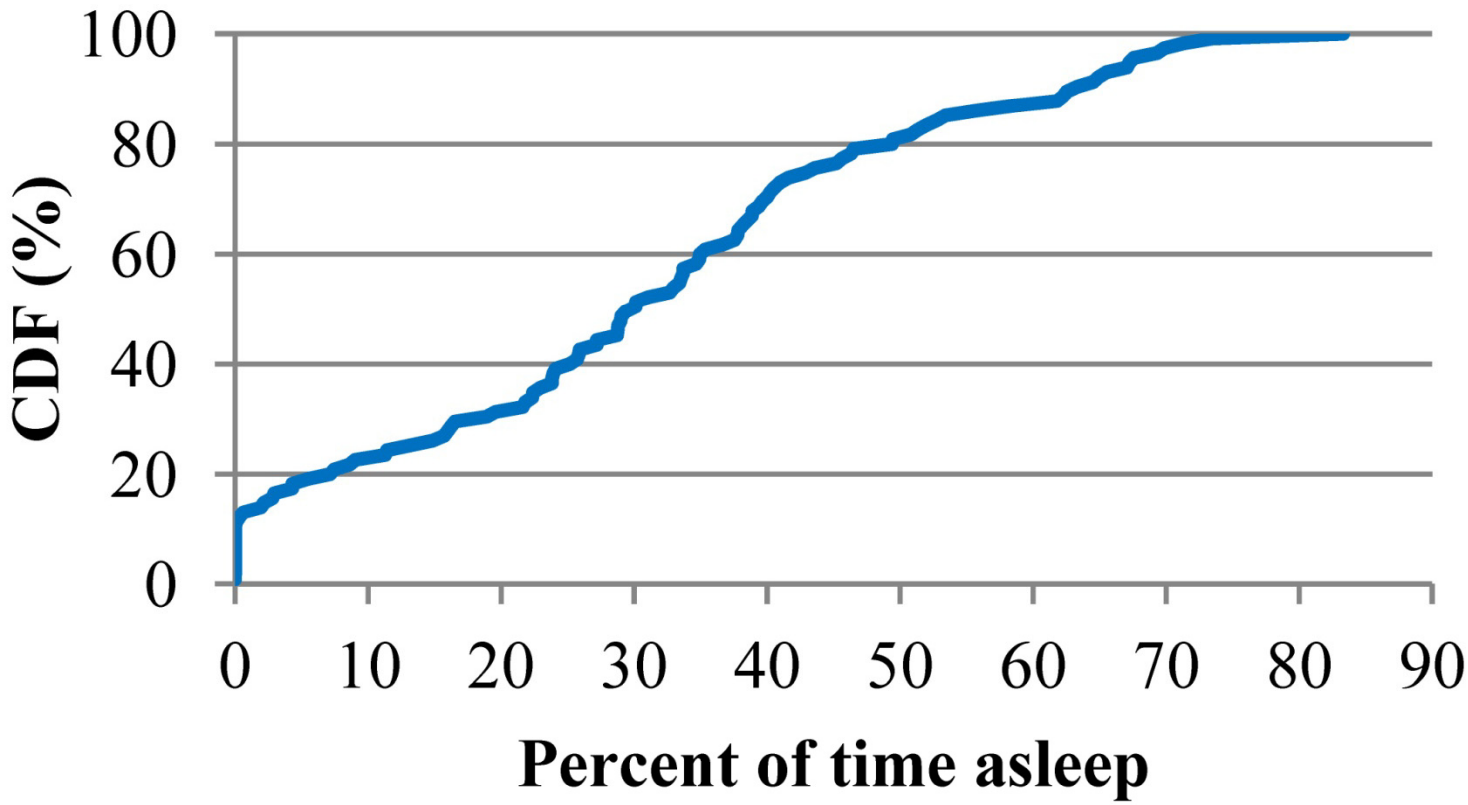
- Guardians ensure max load is 100

Does GreenUp save more energy?

- Energy savings depends on sleep time

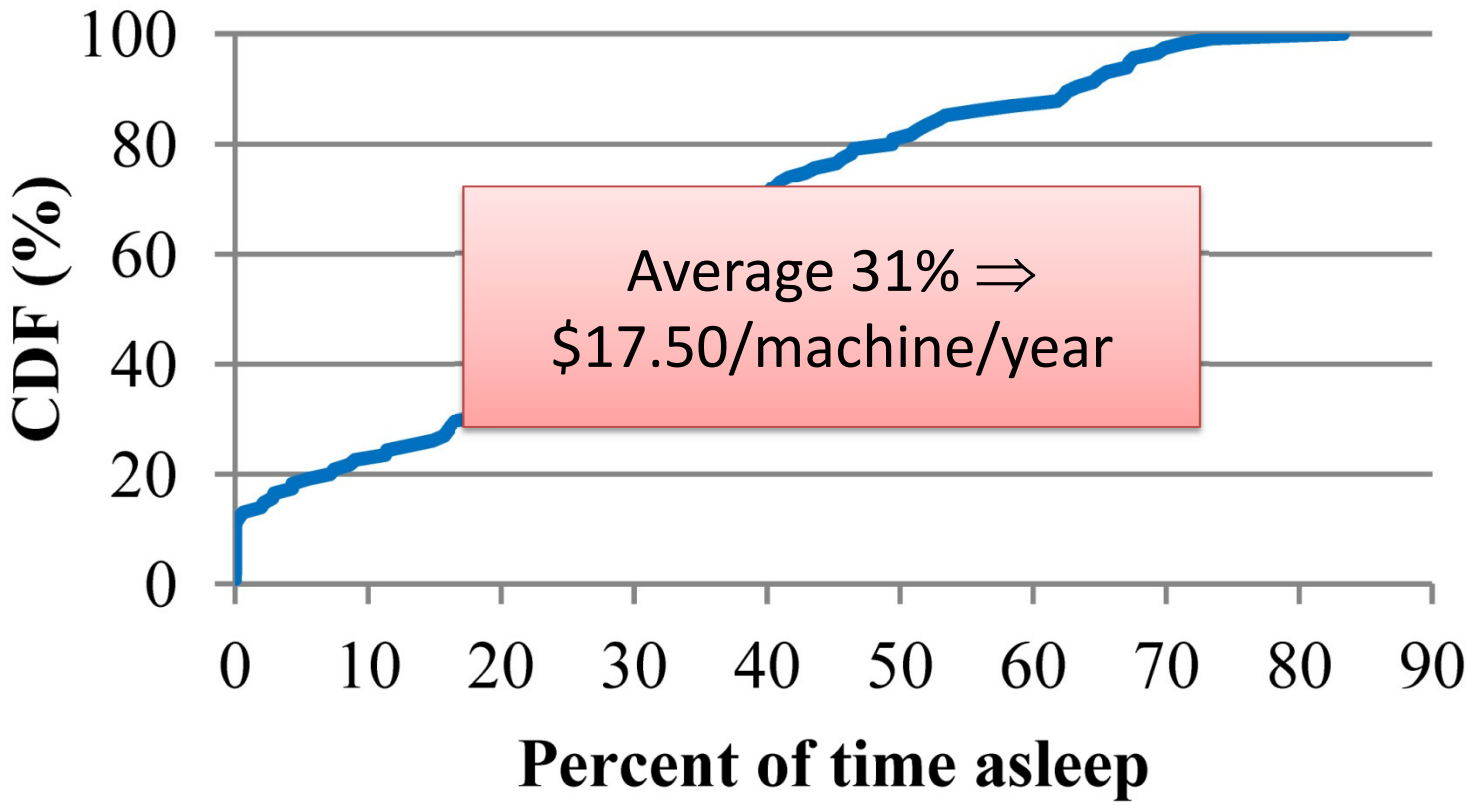
Does GreenUp save more energy?

- Energy savings depends on sleep time



Does GreenUp save more energy?

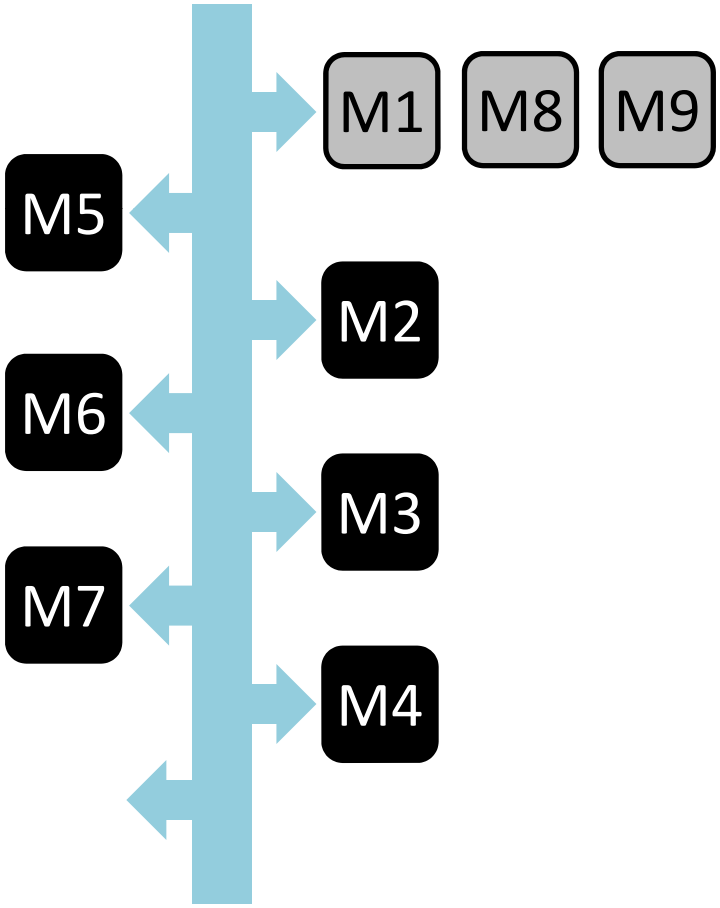
- Energy savings depends on sleep time



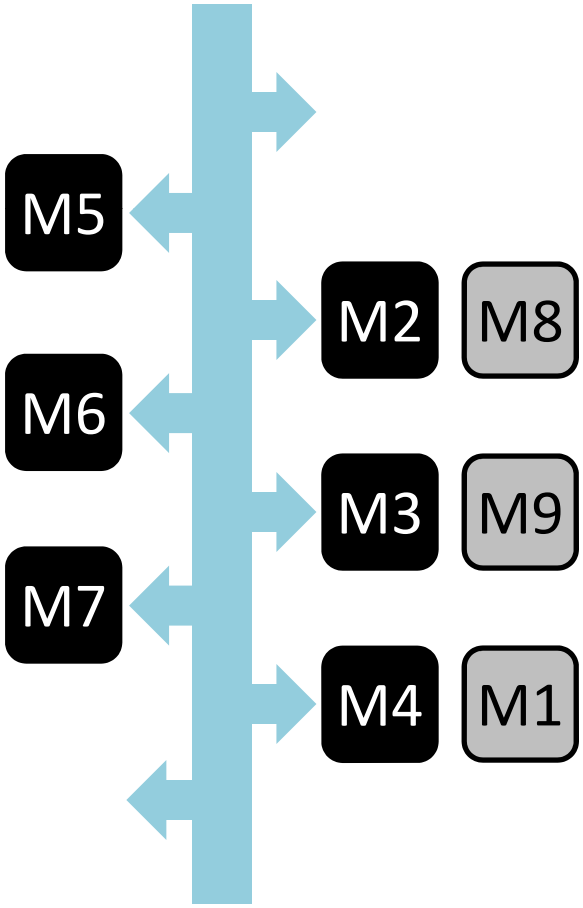
Does GreenUp save more energy?

- Energy savings depends on sleep time
- IT enforces sleep policy at Microsoft, so hard to tell

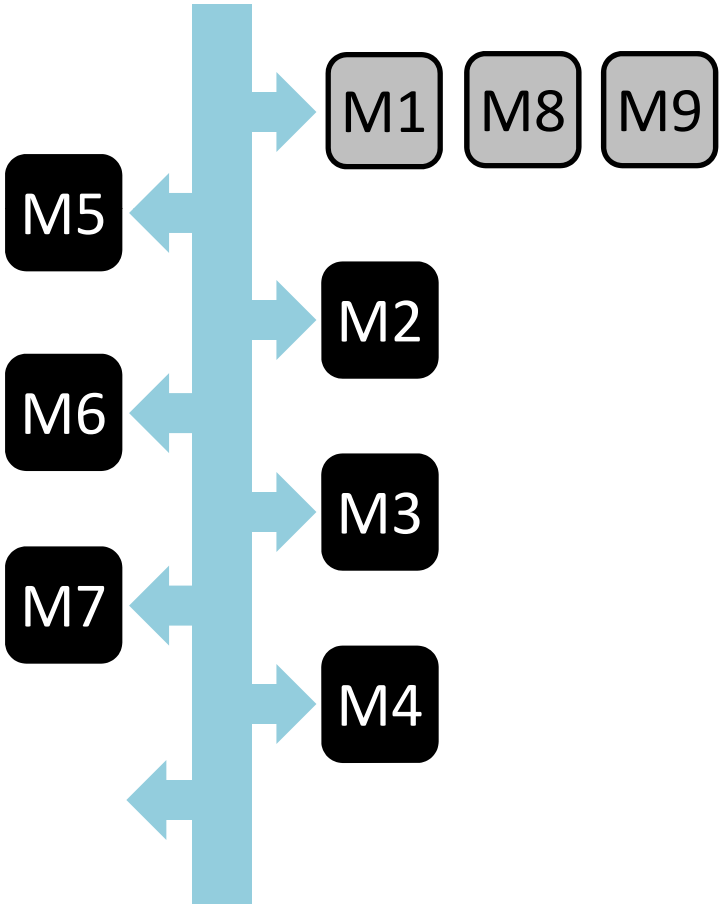
Extension: Higher availability via explicit load hand-off



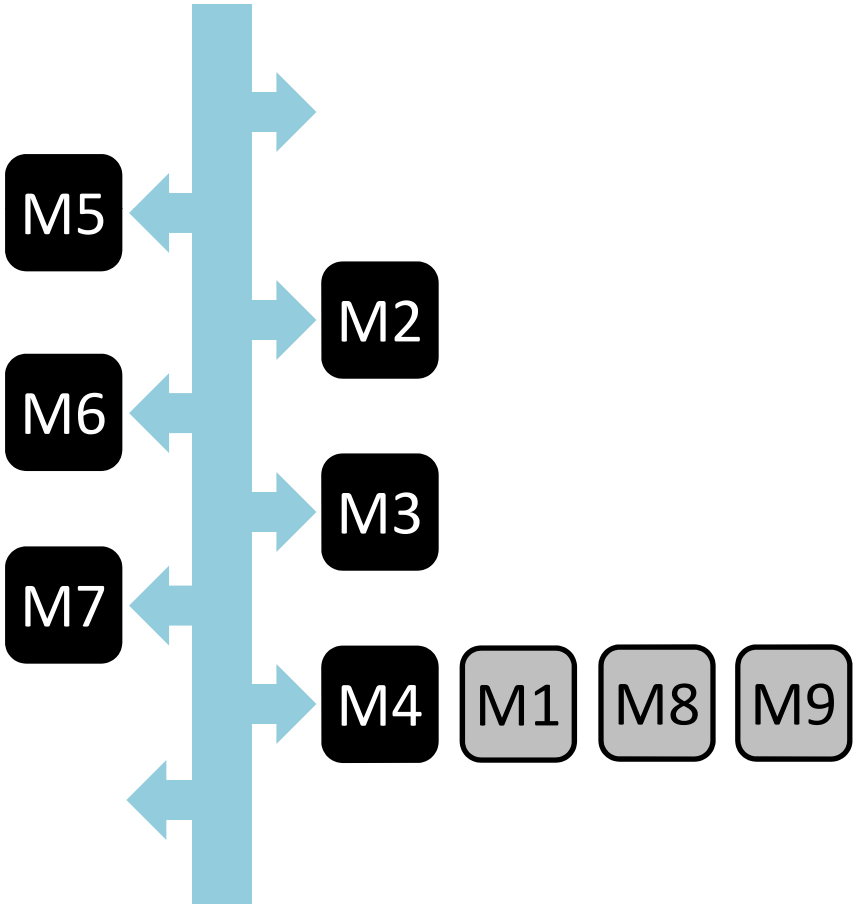
Extension: Higher availability via explicit load hand-off



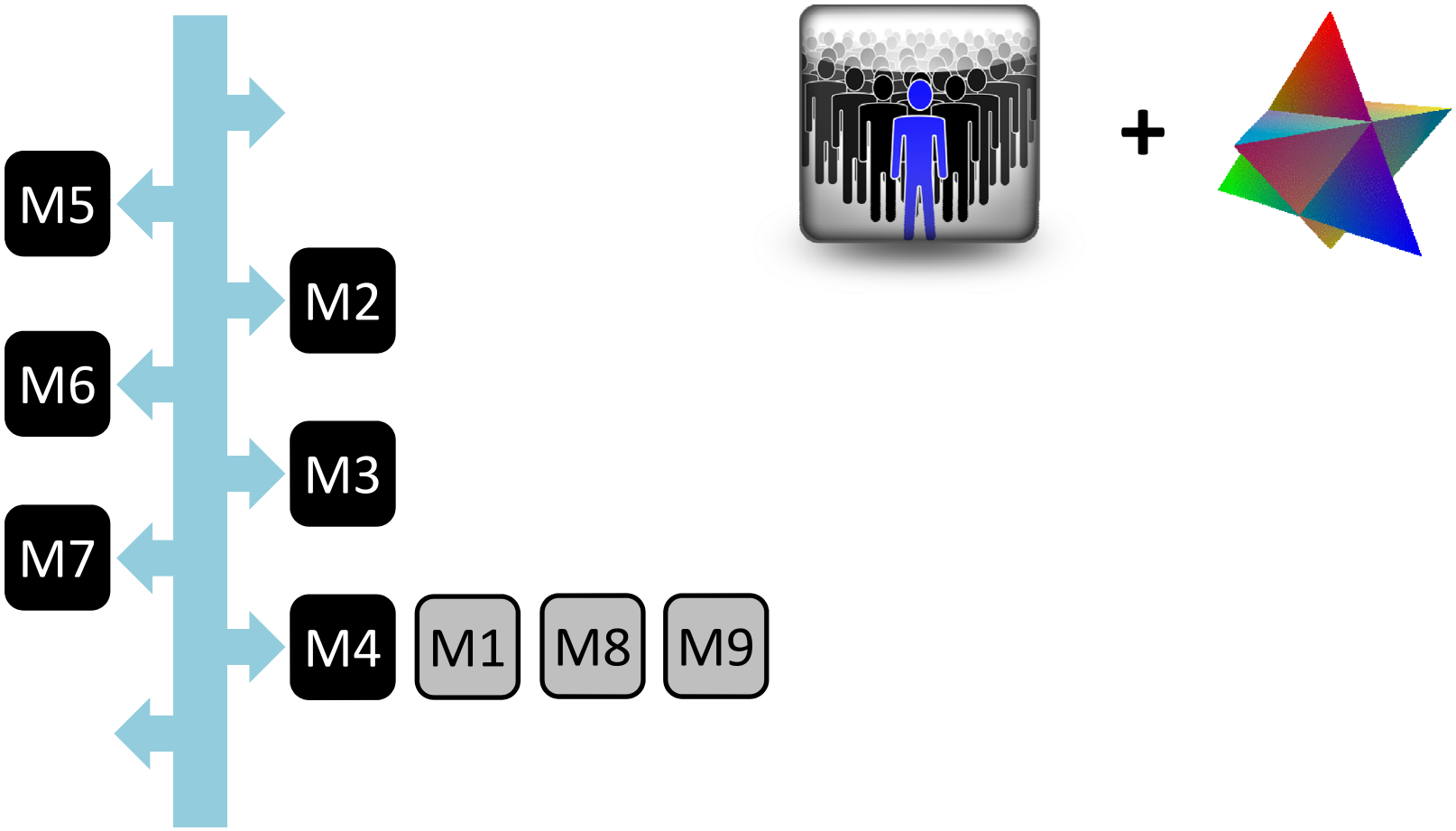
Extension: Higher availability via explicit load hand-off



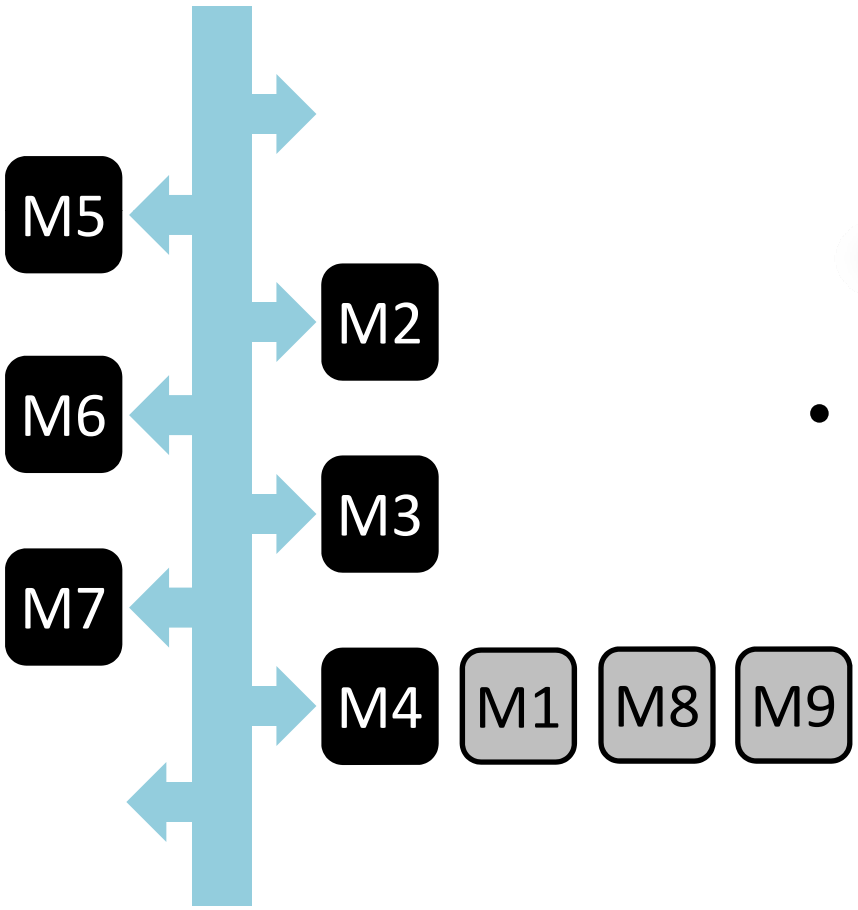
Extension: Higher availability via explicit load hand-off

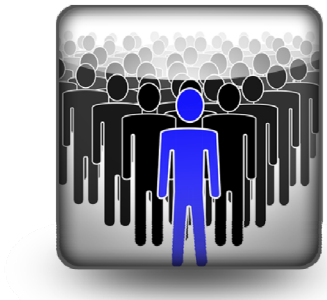


Extension: Higher availability via explicit load hand-off

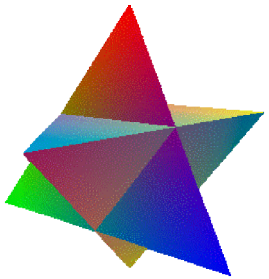


Extension: Higher availability via explicit load hand-off





+



- **Theorem**. Expected max load = $n/d \times H_d$.

awake machines

Harmonic numbers

Other solutions

- Sleep proxy idea: Christensen & Gulledge ’98
- Recently:

System	Technique
Somniloquy, NSDI ’09	augmented NICs
LiteGreen, ATC ’10 Jettison, EuroSys ’12	VM migration
SleepServer, ATC ’10	application stubs
Nedevschi *et al*., NSDI ’08 Reich *et al*., ATC ’10	dedicated servers

Other solutions

- Sleep proxy idea: Christens ge '98
- Recently:

Barriers to deployment

System	Technique
Somniloquy, NSDI '09	augmented NICs
LiteGreen, ATC '10 Jettison, EuroSys '12	VM migration
SleepServer, ATC '10	application stubs
Nedevschi *et al.*, NSDI '08 Reich *et al.*, ATC '10	dedicated servers

GreenUp

- Completely decentralized, software-only sleep proxy
- Useful distributed systems techniques
- High availability at low cost, even as machines sleep!

http://research.microsoft.com/en-us/projects/greenup/