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

WAN

users, IT admins
Enterprise networks

Despite the cloud, this is a common scenario
Enterprise networks

Energy savings

Stay Green!

Stay Up!

Availability
Enterprise networks

Energy savings

Stay Green!

GreenUp!

Stay Up!

Availability
Sleep proxy

Machine (active)

WAN
Sleep proxy

Machine (active)

WAN

Sleep proxy
Sleep proxy

Machine (asleep) — WAN — Sleep proxy
Sleep proxy

Machine (asleep)

WAN

Send traffic to me!

Sleep proxy
Sleep proxy

WAN

Send traffic to me!

Machine (asleep)

Sleep proxy
Sleep proxy

Machine (asleep)

WAN

Sleep proxy
Sleep proxy

Remote request (TCP SYN)

WAN

Machine (asleep)

Sleep proxy
Sleep proxy

Machine (asleep)

WAN

Remote request (TCP SYN)

Sleep proxy
Sleep proxy

- Machine (asleep)
- Sleep proxy
- Wake up! (WoL)

WAN
Sleep proxy

Machine (asleep)

Wake up! (WoL)

WAN

Sleep proxy
Sleep proxy

Machine (active)

WAN

Sleep proxy
Sleep proxy

Machine (active)

Remote request (TCP SYN)

Sleep proxy

WAN
Sleep proxy

Remote request (TCP SYN)

Machine (active)

WAN

Sleep proxy
Sleep proxy

- User
- WAN
- Response Machine (active)
- Sleep proxy
Sleep proxy

Machine (active)

WAN

Response

Sleep proxy
Sleep proxy

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?
Outline

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

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)

- M5
- M6
- M7
- M9 (asleep + unmanaged)
- M1
- M2
- M3
- M4
- M8 (awake)

M1 and M8 are awake.
Running example (not to scale)
Distributed management: Who manages M9?
Distributed management: Who manages M9?

- Wait for notification?

- Load-sensitive machines, so distribute probing robust to manager issues
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?

\[ \frac{\text{total # machines}}{n} = \# \text{awake machines} \]
Distributed management: Who manages M9?

\[
\frac{n - m_i}{\# \text{ awake machines}}
\]
Distributed management: Who manages M9?

\[
\frac{n - m_i}{\text{# awake machines}}
\]
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?

\[ p = \Pr(\text{machine probed}) \]

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

- Coupon collector analysis
Distributed management: Who manages M9?

\[ p = \Pr(\text{machine probed}) \]

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

- 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

- Availability most important
- Simple resolution protocol

\[
\frac{(n - m_i) \ln \left( \frac{1}{1-p} \right)}{\text{# awake machines}}
\]
Multiple managers

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

- Availability most important
- Simple resolution protocol
Load balance

M5
M6
M7
M1 M8 M9
M2
M3
M4
Load balance
• Induction analysis: equivalent to balls-in-bins!

\[
\frac{\ln(n/2)}{\ln \ln(n/2)} \quad \text{after } n/2 \text{ sleeps}
\]
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

<table>
<thead>
<tr>
<th>Machine</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>M8</td>
<td>...</td>
</tr>
<tr>
<td>M9</td>
<td>...</td>
</tr>
<tr>
<td>M5</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>M4</td>
<td></td>
</tr>
<tr>
<td>M7</td>
<td></td>
</tr>
<tr>
<td>M6</td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td></td>
</tr>
<tr>
<td>M8</td>
<td></td>
</tr>
</tbody>
</table>
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 distributes per-machine state on a subnet when strong consistency is not required.
Outline

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

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?

- Protects against simultaneous sleep
- Caps the max load

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?

3. How effective is GreenUp?
   - Evaluation on \( \sim 100 \) user machines, currently deployed on \( \sim 1,100 \) 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 days (weekends, evenings):
  – 496 already awake
  – 278 woken
  – 5 unwakeable
  – Most failures due to WoL

• 99.4% success rate
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

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

87% of wakeups take < 9 sec
GreenUp wakes machines quickly

- 87% of wakeups take < 9 sec
- 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.

**Graph:**
- 87% of wakeups take < 9 sec.
- 13% of users give up after 3 sec (port scanners?)

**Frequency Distribution (CDF):**
- Time (sec) vs. Frequency (%)
- CDF (%) range from 0 to 100
- Time (sec) range from 0 to 180
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

Good load balance + enough awake machines $\Rightarrow$ few managed machines!
GreenUp scales to large subnets

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

<table>
<thead>
<tr>
<th># of managed machines</th>
<th>CPU utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>12%</td>
</tr>
<tr>
<td>200</td>
<td>21%</td>
</tr>
<tr>
<td>300</td>
<td>29%</td>
</tr>
</tbody>
</table>
GreenUp scales to large subnets

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

<table>
<thead>
<tr>
<th># of managed machines</th>
<th>CPU utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>12%</td>
</tr>
<tr>
<td>200</td>
<td>21%</td>
</tr>
<tr>
<td>300</td>
<td>29%</td>
</tr>
</tbody>
</table>

• 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

Average 31% ⇒ $17.50/machine/year
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

M5 → M2 → M3 → M4 → M1, M8, M9
Extension: Higher availability via explicit load hand-off
Extension: Higher availability via explicit load hand-off

- **Theorem.** Expected max load $= \frac{n}{d} \times H_d$.  
  - # awake machines
  - Harmonic numbers
Other solutions

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

<table>
<thead>
<tr>
<th>System</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Somniloquy, NSDI ’09</td>
<td>augmented NICs</td>
</tr>
<tr>
<td>LiteGreen, ATC ’10</td>
<td>VM migration</td>
</tr>
<tr>
<td>Jettison, EuroSys ’12</td>
<td></td>
</tr>
<tr>
<td>SleepServer, ATC ’10</td>
<td>application stubs</td>
</tr>
<tr>
<td>Nedevschi <em>et al.</em>, NSDI ’08</td>
<td>dedicated servers</td>
</tr>
<tr>
<td>Reich <em>et al.</em>, ATC ’10</td>
<td></td>
</tr>
</tbody>
</table>
Other solutions

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

<table>
<thead>
<tr>
<th>System</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Somniloquy, NSDI ’09</td>
<td>augmented NICs</td>
</tr>
<tr>
<td>LiteGreen, ATC ’10</td>
<td>VM migration</td>
</tr>
<tr>
<td>Jettison, EuroSys ’12</td>
<td></td>
</tr>
<tr>
<td>SleepServer, ATC ’10</td>
<td>application stubs</td>
</tr>
<tr>
<td>Nedevschi et al., NSDI ’08</td>
<td>dedicated servers</td>
</tr>
<tr>
<td>Reich et al., ATC ’10</td>
<td></td>
</tr>
</tbody>
</table>
GreenUp

• Completely decentralized, software-only sleep proxy

• Useful distributed systems techniques

• High availability at low cost, even as machines sleep!