Sleepless in Seattle No Longer

Joshua Reich*, Michel Goraczko,
Aman Kansal, and Jitu Padhye

Columbia University*, Microsoft Research

A Short Story: Sleepless in Seattle

- A desktop machine
  - Workdays: often used, sometimes idle
  - Nights, holidays, weekends: often idle
    - **sometimes** accessed remotely by **user**
    - **more often** accessed by **IT** (patches, updates, scans)
- But **always powered on**

A Short Story: Sleepless in Seattle

- Why?
- B/c its user and the IT dept want
  - **continuous remote availability**
  - **seamless access**
    (no fiddling w/ manual tools to wake machine)

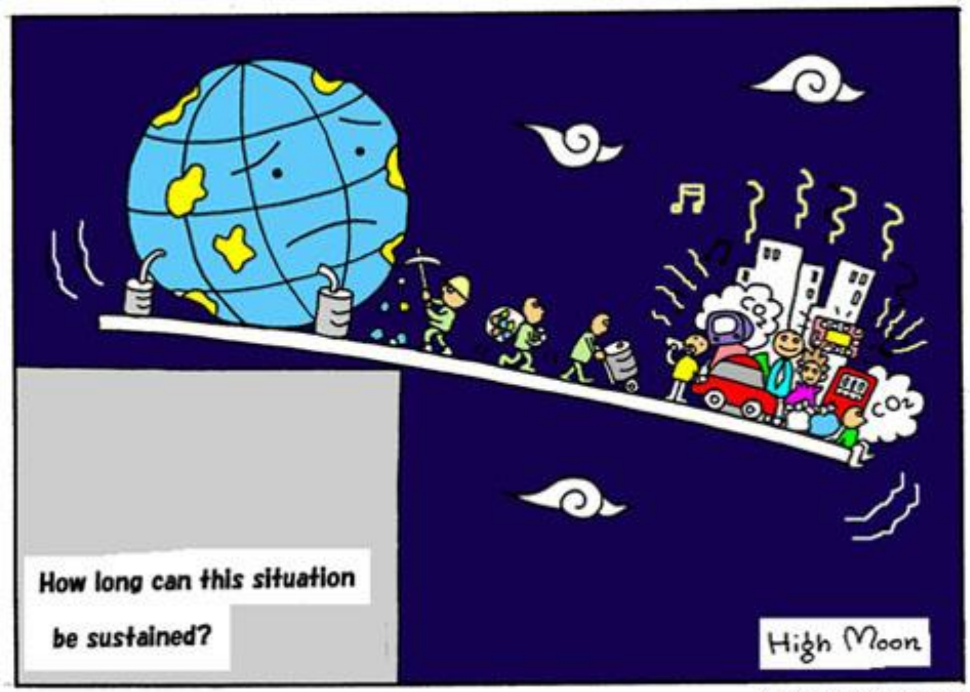
This Story is Typical

- **Enterprise machines rarely sleep**
  - **2/3rds** of office PCs are **left on** after hours*
  - Or is it **95%**? Power management **disabled****
  - 600+ desktops **always left on** (of total 700+)***
  - Almost all desktop at MSR left on after hours
  - [Your own stat or anecdote here]

*Robertson et. al.: After-hour power status of office equipment and energy usage of plug-load devices. LBNL report #53729
**Nordman, http://www.lbl.gov/today/2004/Aug/20-Fri/r8comm2.lo.pdf
***Agarwal et. al: Somniloquy, Augmenting network Interfaces to reduce PC energy usage (NSDI 2009)

Wasteful Resource Consumption

- **Not a story with a happy ending**



©Takatsuki Hiroshi

- Unless we **change** things
- This talk is about **making one such change**, focusing on **practicality** and **economic feasibility**

Outline

- Problem
- Sleep Proxy Architecture
- Deployment & Instrumentation
- Findings
- Related Work and Next Steps

Outline

- **Problem**
- Sleep Proxy Architecture
- Deployment & Instrumentation
- Findings
- Related Work and Next Steps

Back of Envelope Energy Waste

- If machine
  - Draws 100W when awake
  - Actually being used 50% of the time.
- Then **400-500 *kWh*** are **wasted** per year.
- For Microsoft this is something like **40 *GWh*.**
- Over the entire US, on the order of **20 *TWh***!*

*Wolfram Alpha, 112.6 million service industry workers, let's assume roughly 1/3rd have desktop machines for total of 40M enterprise desktops

Sleep Proxies **Can Help**

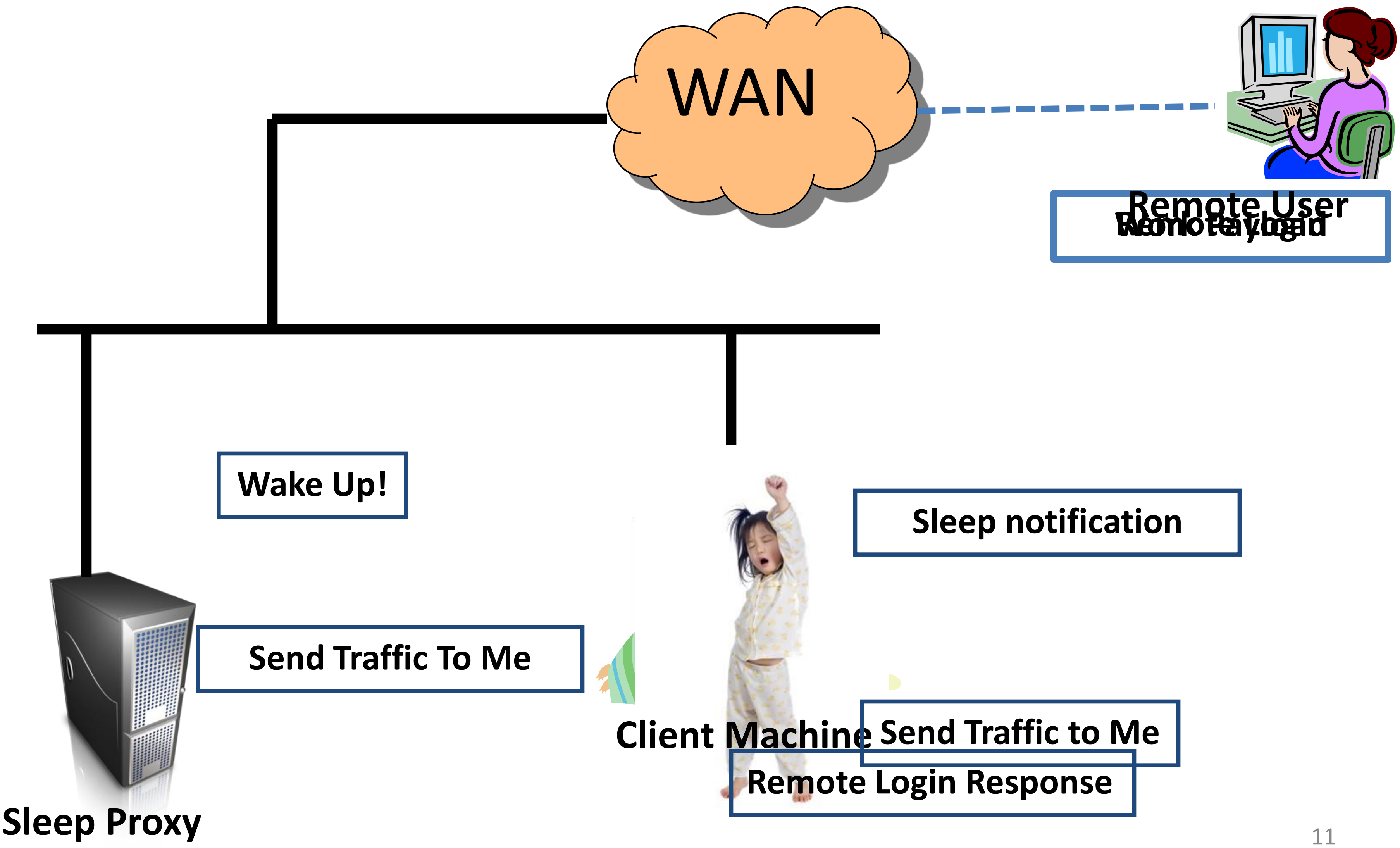
- A ***Sleep Proxy*** allows a machine to be
  - **network available**
  - while **physically asleep**

Reaction Policy

- When machine sleeps, sleep proxy takes over, examines traffic, following a ***Reaction Policy***
  - **Respond** (e.g., ARP)
  - **Wake** the sleep machine (e.g., remote login)
  - **Ignore** (e.g., ICMP)

- Reaction Policy choices determine
  - Amount of potential sleep **actually saved**
  - **Co$t and complexity** of sleep-proxying system

How a Network Sleep Proxy Works



WAN

Remote User

Remote Login

Work Payload

Wake Up!

Sleep notification

Send Traffic To Me

Client Machine

Send Traffic to Me

Remote Login Response

Sleep Proxy

Sleep Proxy Economics

The Type of **Green** Companie$ Really Care About

- Single machine savings: only **$60-$70** **per year** (though rising)
- Now multiply by 40M enterprise desktops
  => **$1-3 Billion*** yearly savings, just in USA.
- But for a single company – a couple of **100,000** to a couple of **million $'s** per year

*In line w/ Nordman report's $0.8 – 2.7 Billion estimated savings.

The Bottom Line

- **Savings**
  - Very substantial in **aggregate**
  - Relatively small for **individual** companies.

- **=>** Sleep-proxying systems need to be **cheap**
  - Low **hardware** cost
  - Good ***consolidation ratio***
    (#sleep proxies : #desktops)
  - Low **admin / setup** cost

Sleep-Proxying **Isn't a New Idea**

- First suggested **over a decade ago**
  - Christensen & Gulledge, 1998
- Taken up again **recently**
  - *Allman, et al.,* Hotnets, 2007
  - *Agarwal, et al.*, NSDI, 2009
  - *Nedevschi, et al.*, NSDI, 2009
- **Two other great papers** here at USENIX ATC
  - LiteGreen, *Das, et al.* **(Virtualization)**
  - SleepServer, *Agarwal, et al.,* **(Custom App Stubs)**

Our Contributions

- A design geared towards **cheap hardware**
  – **One** dedicated machine **per subnet** (or less)
  – Proxy can be run on a **low power** box
    - **Atom processor** machine? No prob.
    - Probably even wall-plug, Open/DDWRT style as well
- And **little work for IT**
  – **Simple, lightweight** client side install
  – **No** client-side configuration or hardware **changes**
  – **Little admin or setup** needed on proxy side

Our Contributions (cont.)

- First **operational enterprise** deployment
  - Likely where the biggest bang for the buck
  - Home users tending to low power devices anyway
  - Smaller # of desktops in academic-style networks
- Provide insight on what sleep-proxied enterprise **might actually look like**
  - Why machines **are woken**
  - Why they **stay awake**
  - Where our approach works well and falls short

Outline

- Problem
- **Sleep Proxy Architecture**
- Deployment & Instrumentation
- Findings
- Related Work and Next Steps

Sleep-Proxying System Design Goals

- Given normal workload,
  choose **architecture** and **reaction policy**
  - **No change** to network applications
  - **Minimal client-side**/network change, configuration
  - Sleep proxies that
    - Can be deployed on cheap, **low power hardware** (maybe even run on peers themselves)
    - Can **cover all clients** in a subnet
    - Close to **zero-configuration** /administration
- **Provide reasonable opportunity for sleep**

Our Sleep-Proxying Design Principle

First 90% savings w/ 10% of the **cost**

90 / 10

*Tom Cargill, Bell Labs. Popularized by Jon Bentley in Communications of the ACM, Programming Pearls, 1985

Our Sleep-Proxying Design Principle

Leave final 10% savings**, avoiding the other 90% of the cost**

10 / 90

*Tom Cargill, Bell Labs. Popularized by Jon Bentley in Communications of the ACM, Programming Pearls, 1985

Our Sleep-Proxying System Design

- **Client side service** (daemon)
  - Sends sleep notifications
  - Informs sleep proxy about all **LISTENING ports**
  - Almost **no resource consumption**
  - Uses **native OS sleep policies**
  - **User self-install** from standard MSI (two clicks)
  - No **client-side configuration work for IT**

Our Sleep-Proxying System Design

- **Sleep proxy reaction policy**
  - **Respond**: to **IP address resolution** traffic (e.g., ARP, Neighbor-Discovery)
  - **Wake:** client on **incoming TCP connection** attempts (recognized by presence of SYN flag)
  - **Ignore: all other traffic**

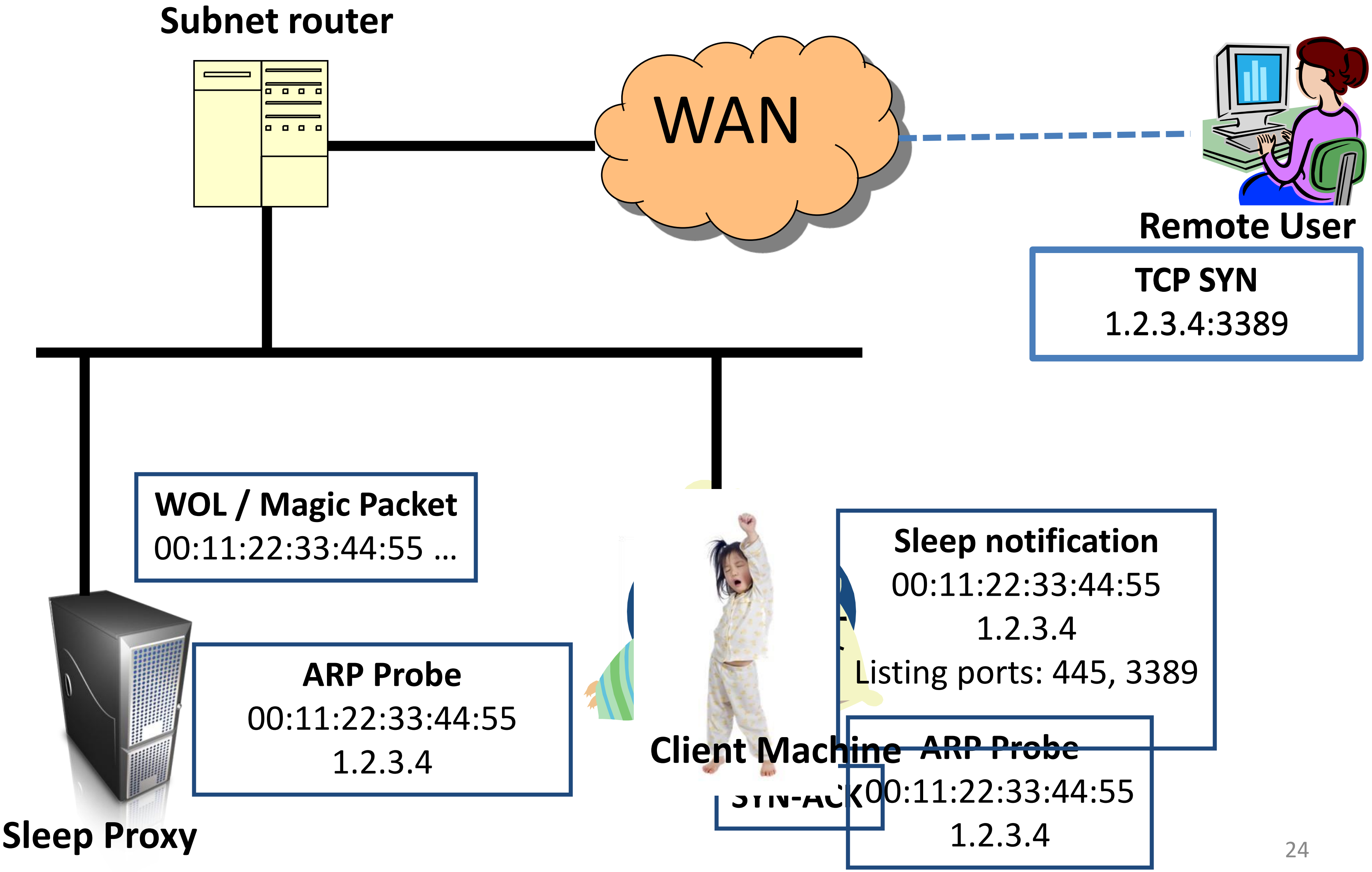
Design Benefits

- No need to define policies determining **for which applications clients should be woken**
- Great **consolidation ratios**
- Low **cost**, low **power**, potentially **peered**, proxies

Digital Engine Mini PC

- Practically no **IT management/config req'd.**

How Our Sleep Proxy Works



Subnet router

WAN

Remote User

TCP SYN
1.2.3.4:3389

WOL / Magic Packet
00:11:22:33:44:55 …

ARP Probe
00:11:22:33:44:55
1.2.3.4

Sleep Proxy

Client Machine

Sleep notification
00:11:22:33:44:55
1.2.3.4
Listing ports: 445, 3389

ARP Probe
00:11:22:33:44:55
1.2.3.4

SYN-ACK

Sample Wakeup Timeline

Remote User **RU** Client Machine **CM** Sleep Proxy **SP**

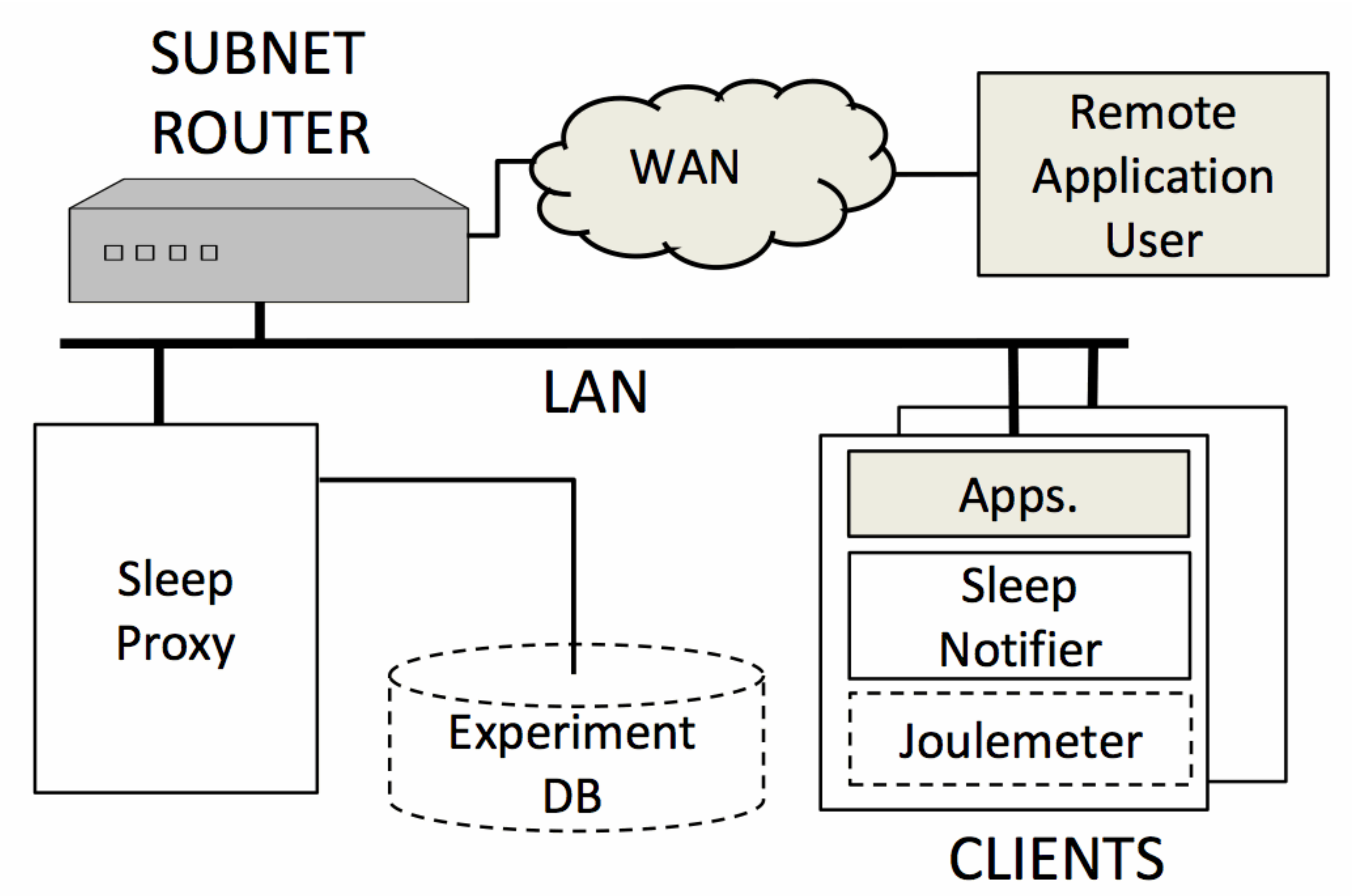
Step	Time	From → To	Packet Type	Note
1	0	RU->(CM) SP	SYN	
2	0.04	RU->CM	Magic packet	
3	3	RU->(CM) SP	SYN	Retransmit
4	5.6	CM->Bcast	ARP Probe	CM awake
5	9	RU->CM	SYN	Retransmit
6	9.01	CM->RU	SYN ACK	

Save by having sleep proxy replay most recent TCP SYN

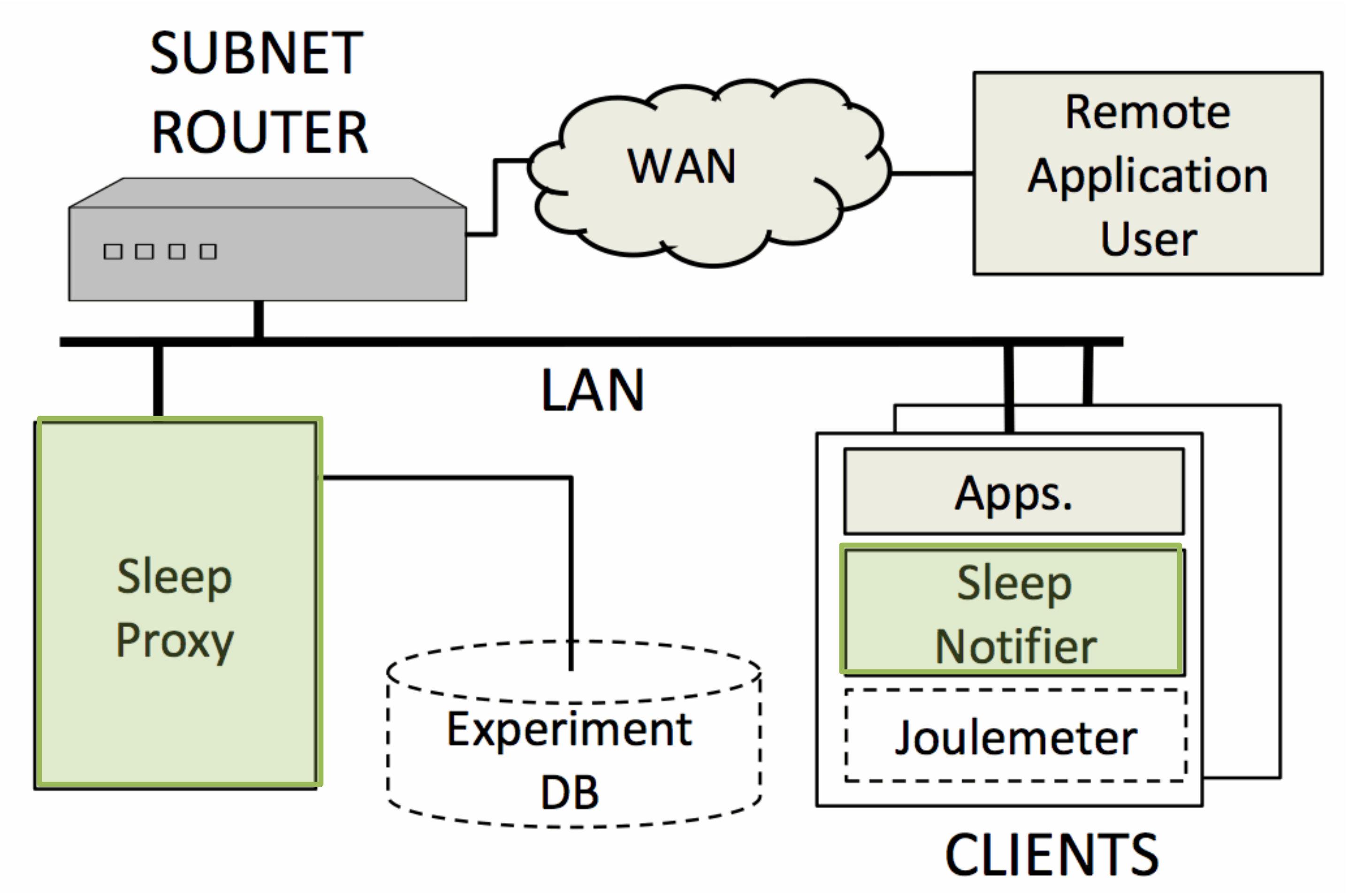
Outline

- Problem
- Sleep Proxy Architecture
- **Deployment & Instrumentation**
- Findings
- Related Work and Next Steps

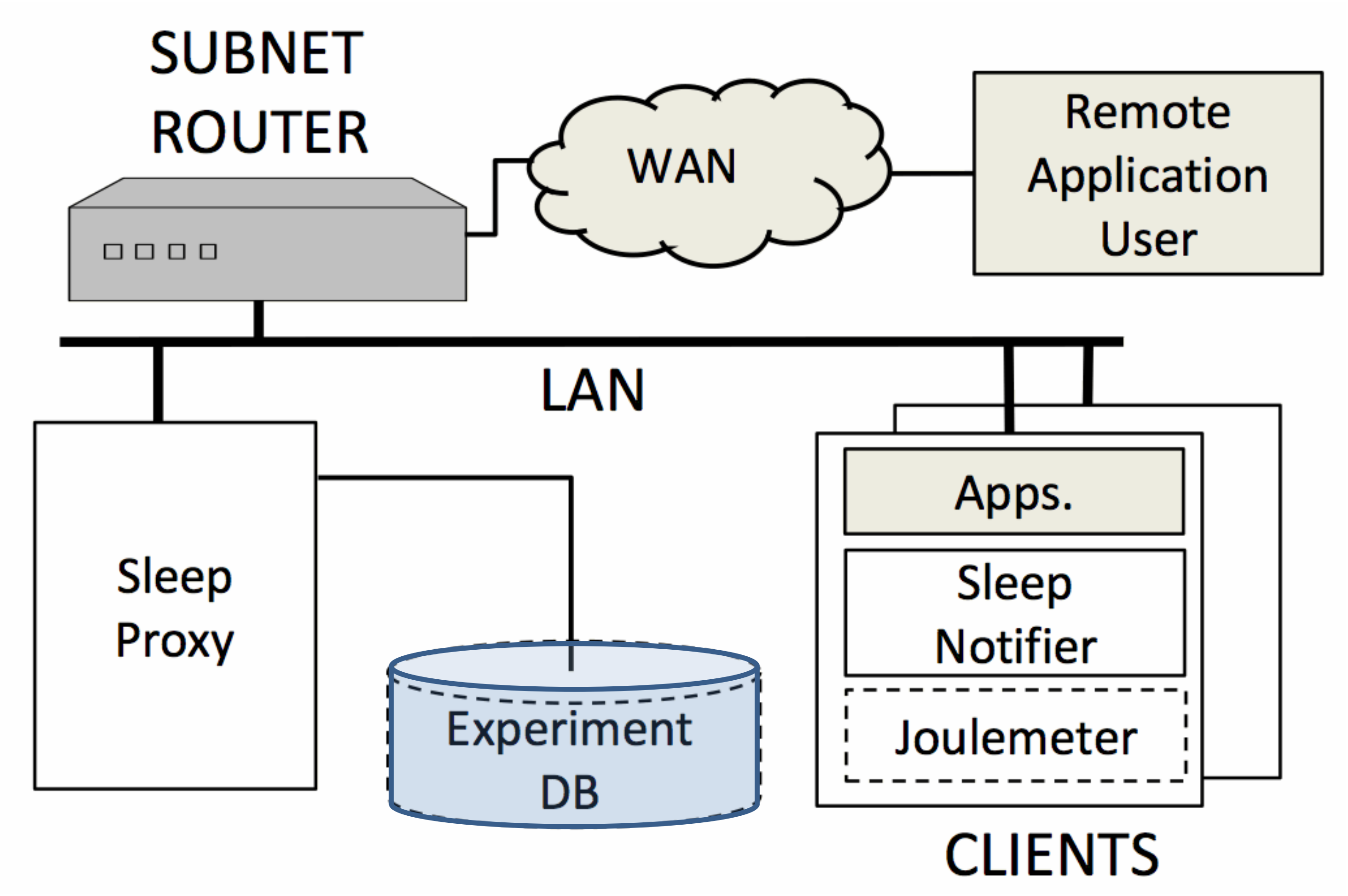
Deployment Architecture



Sleep-Proxying Subsystem

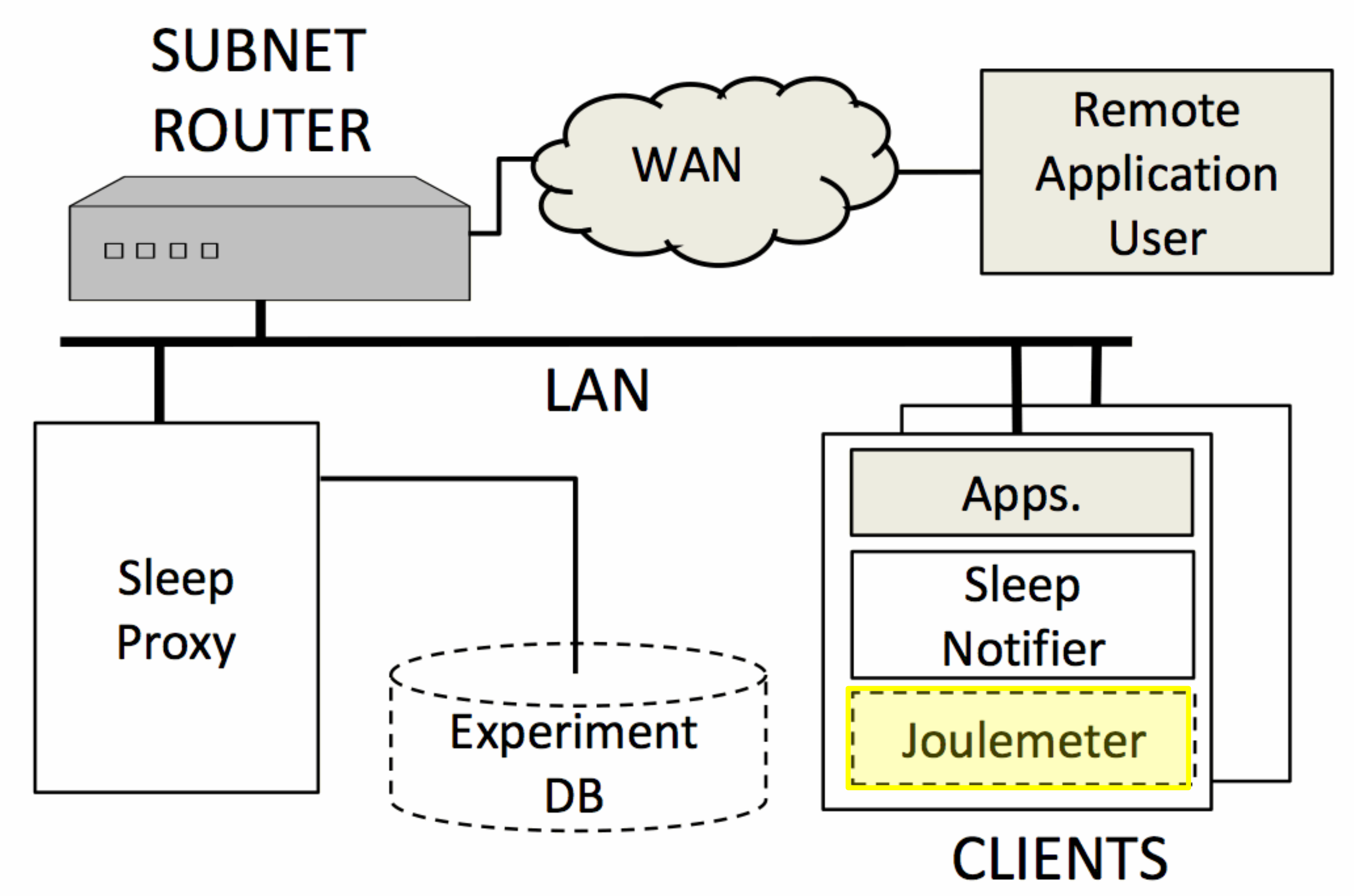


All Sleep Proxies Log Data to DB



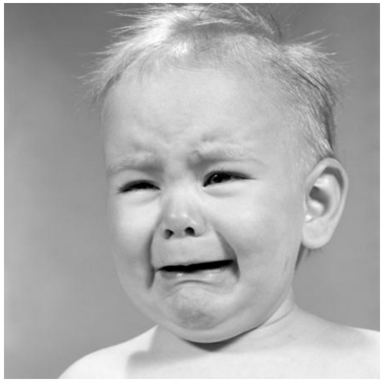
Joulemeter:

- Software-only power monitor
- **Assess Source of Sleep Problems**



Why Machines **Lose Sleep**

- ***Crying baby syndrome***:
  - – Sleeping machine (parent) woken often **by remote clients** (crying babies)



- Identify by measuring
  - – How quickly machines wake after sleeping
  - – What traffic is waking them up and from whom
  - – What processes run immediately after wakeup
  - – Who places **stay-awake requests** with OS*

*POWERCFG /REQUESTS

Why Machines **Lose Sleep**

- ***Application induced insomnia***
  - Machine won’t sleep b/c **app requests**
  - e.g., media server, virus scanner

- How does insomnia happen?
  - `WinAPI SetThreadExecutionState*`
    - `ES_CONTINUOUS`
    - `ES_SYSTEM_REQUIRED`
  - Have remote user hold file open on machine
- Identify by measuring
  - Who places **stay-awake requests** with OS

*http://msdn.microsoft.com/en-us/library/aa373208(VS.85).aspx

Deployment Stats

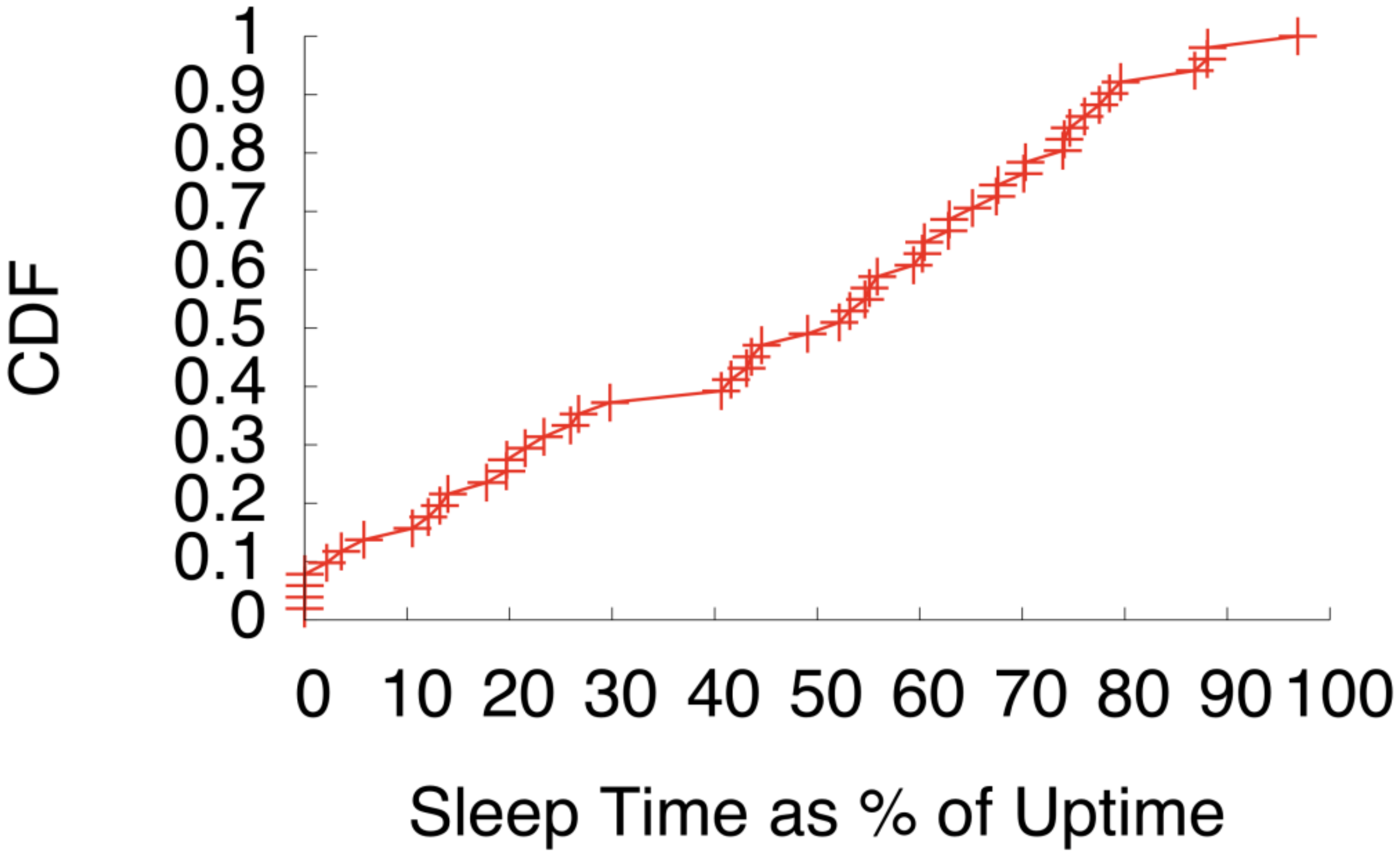
- Sleep Proxies on **6 subnets** in MSR Redmond
- Sleep Clients running on **50+ machines**
  - Installed **by users** (two clicks)
  - Most **primary** user workstations
  - **IT recommended**
- System in operation **almost one year**
- ~ **10 *MWh*** **saved**
  (not bad for a research prototype)

Outline

- Problem
- Sleep Proxy Architecture
- Deployment & Instrumentation
- **Findings**
- Related Work and Next Steps

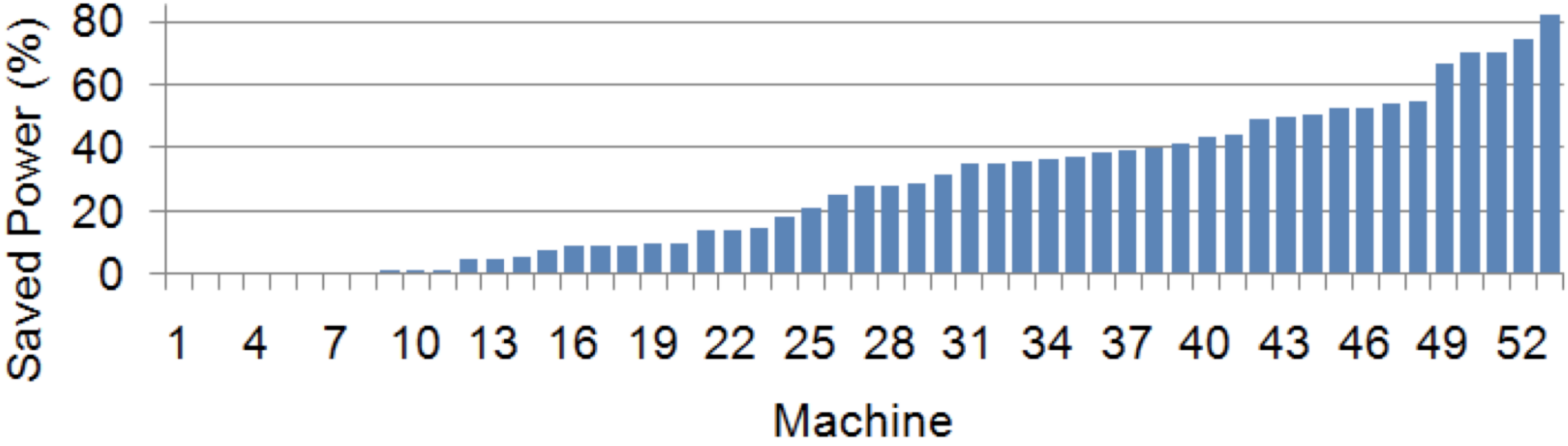
Sleep Savings

- Most machines **sleep most of the time**
- **~20%** machines sleep **very poorly**



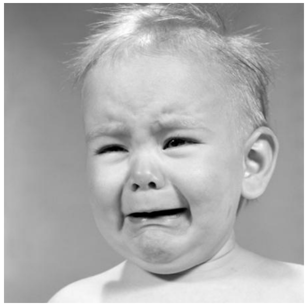
Energy Savings

- Substantial power savings for many machines
- Note: Saved Power is lower bound estimate.



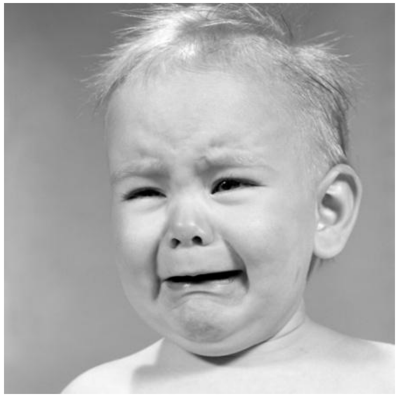
Why Machines **Lose Sleep**

- ***Crying baby syndrome***
  - Sleeping machine (parent) woken often by **remote clients** (crying babies)

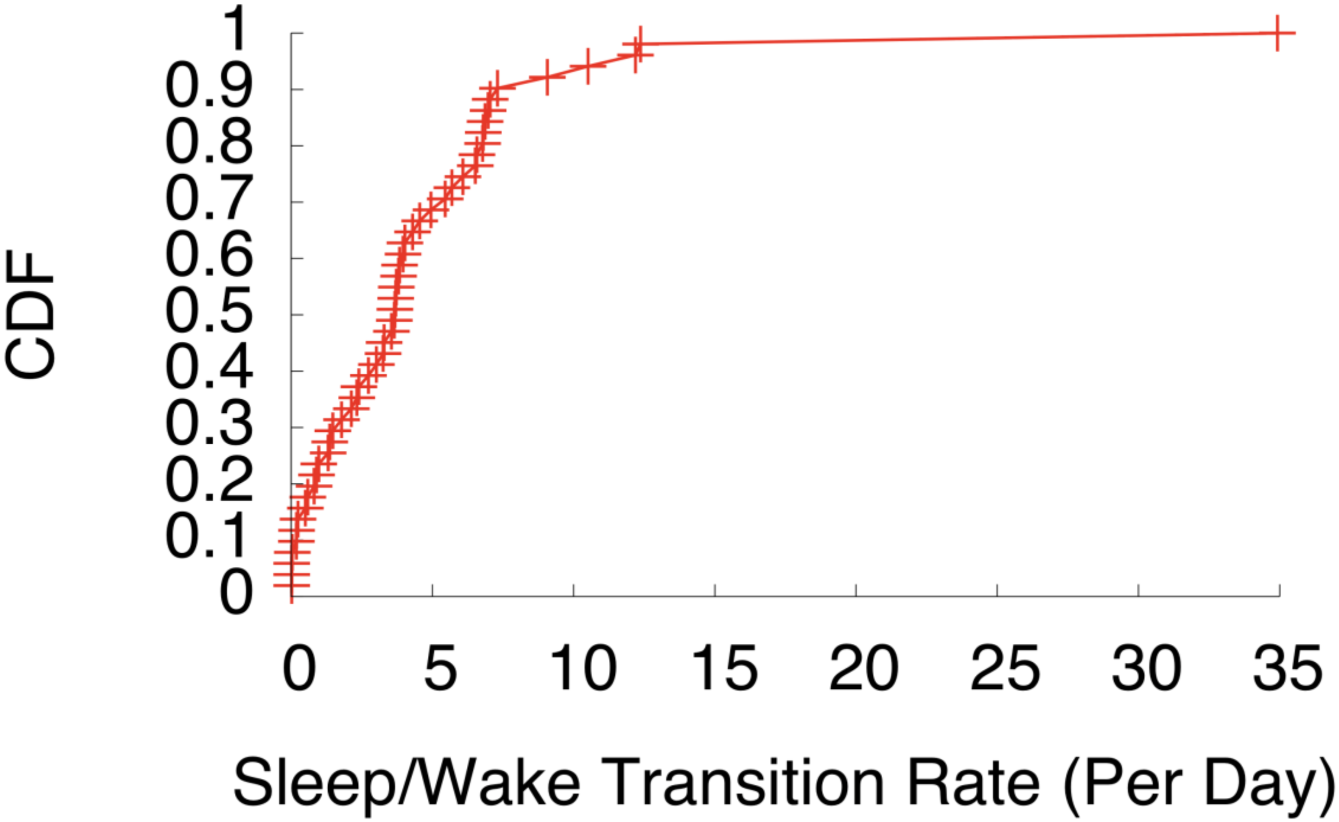


- ***Application induced insomnia***
  - Machine won’t sleep b/c **app requests**
  - e.g., media server, virus scanner

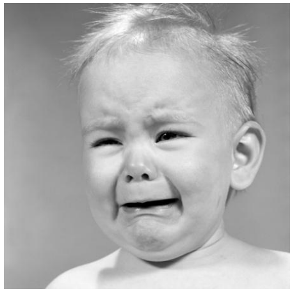
Impact of Crying Babies



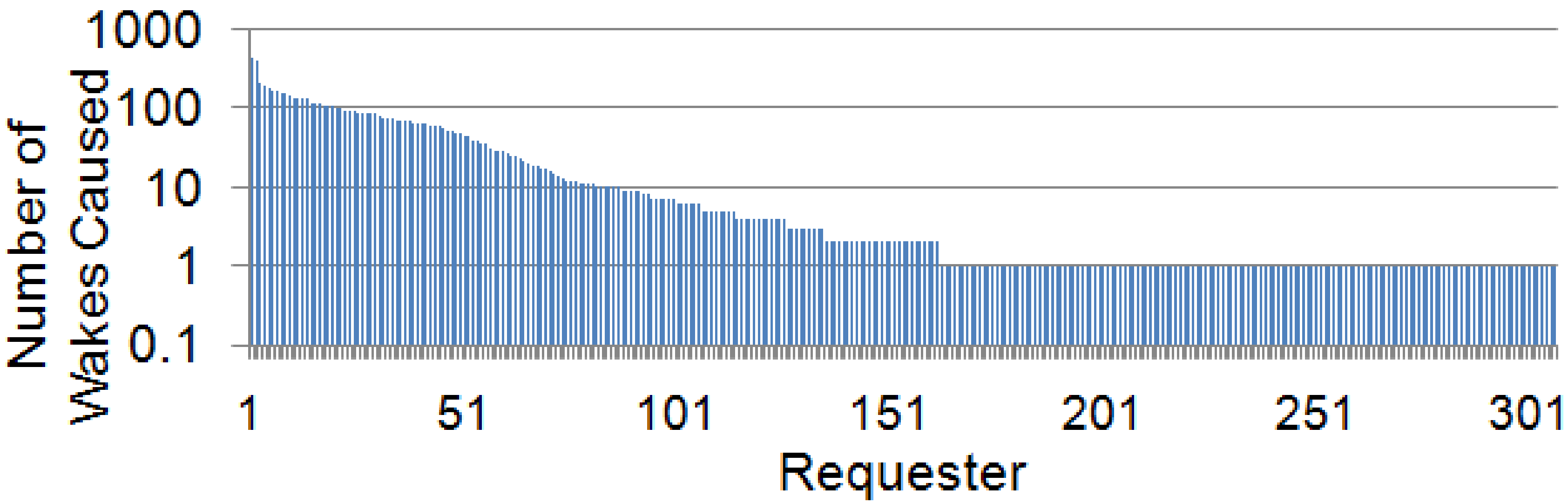
~10% of lost sleep



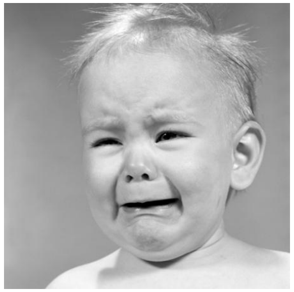
Who are the Crying Babies?



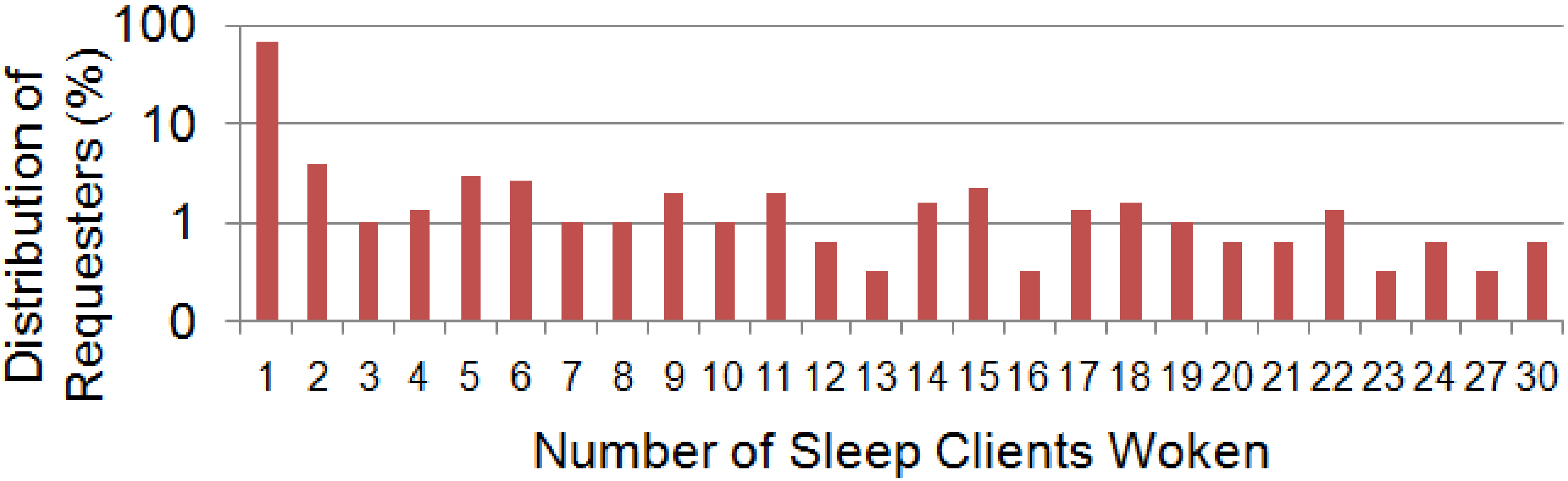
1. Small subset of remote machines (requesters) **that cause lots of wake events**



Who are the Crying Babies?



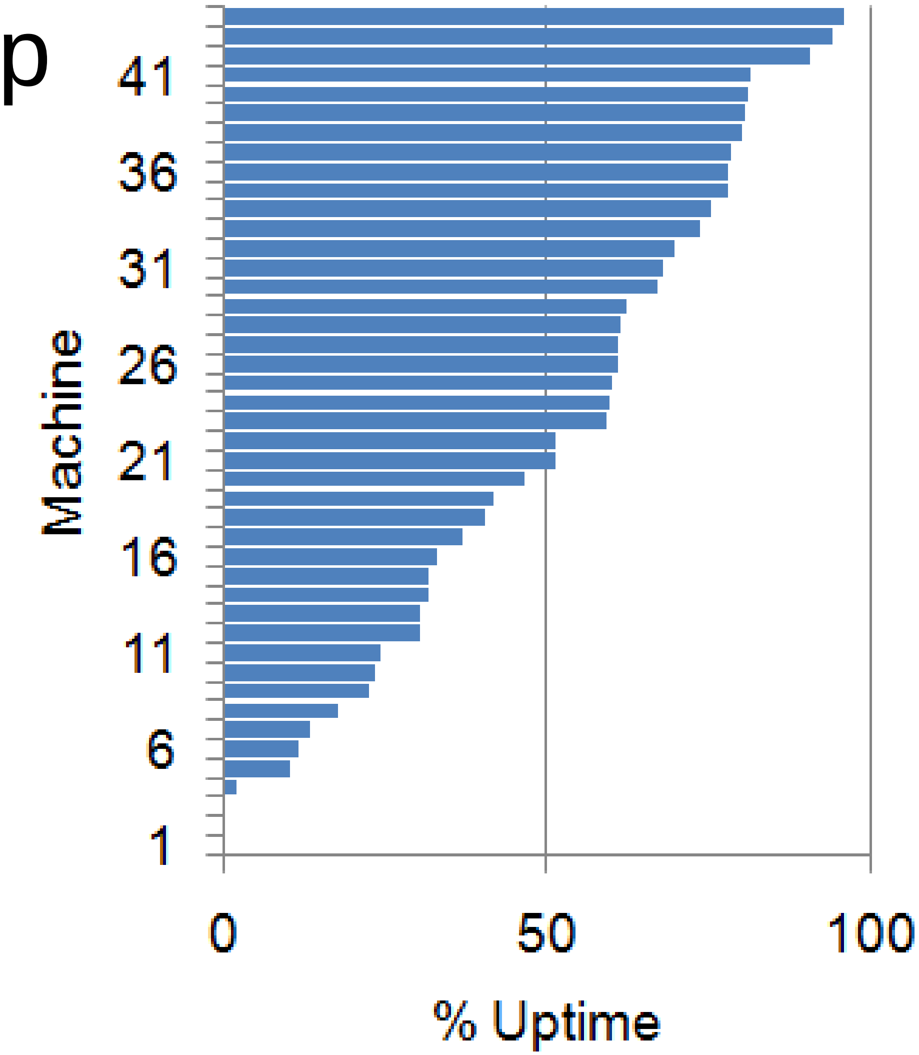
2. Small subset of remote machines (requesters) **that wake lots of sleeping clients**



Requestors mostly **IT servers** (e.g., virus scanners, patch server)

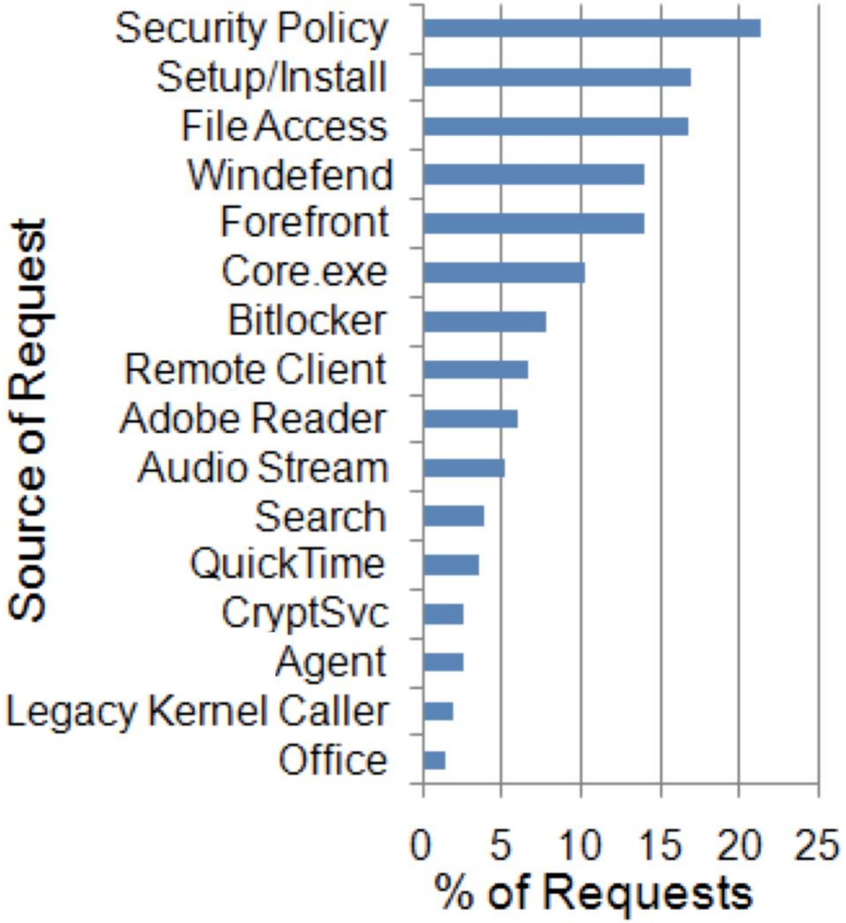
Impact of Insomnia

~90% of lost sleep



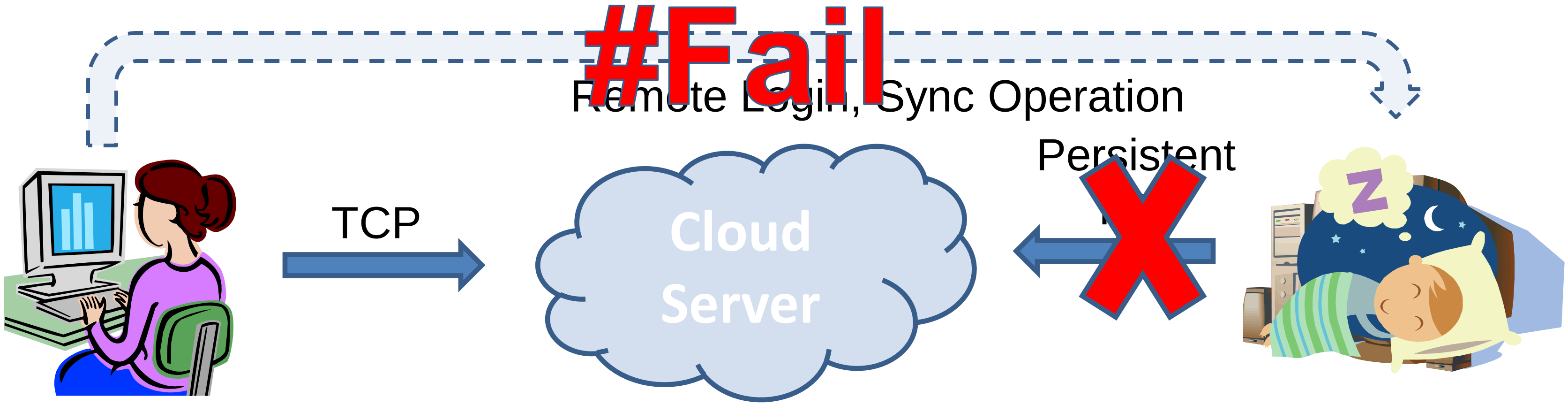
Who Causes Insomnia?

- 5 of top 7 are **IT apps**
- Several caused by
  - program bugs
  - legacy drivers
- Hard to improve via **reaction policy w/o big expen$e**
- Many amenable to **better coordination** of IT tasks



Persistent Cloud Applications

- Small minority used **LiveMesh, LiveSync**



- We refer to these as ***persistent*** cloud apps
  - Designed primarily to overcome NAT/firewall
- Requires more sophisticated reaction policy
- But, not used much in the enterprise

Findings Summary

- **Relatively simple reaction policy** can work well
  - filter by port
  - deal w/ tunneled packets, v4/v6, etc.
- **Insomnia** foremost cause of lost sleep
- **IT** main cause of both insomnia and crying baby
  - Unclear cost effective reaction policy that can help
  - But **intelligent scheduling of IT tasks may help greatly**
    - Wake once, do everything, then sleep soundly
- Greater complexity **can be useful**
  - Persistent cloud apps (**non-enterprise** systems)
  - BitTorrent, Skype, etc. (**non-enterprise** systems)
  - Additional sleep opportunities (**if economical**)

Outline

- Problem
- Sleep Proxy Architecture
- Deployment & Instrumentation
- Findings
- **Related Work and Next Steps**

Next Steps

- P2P Sleep-Proxying (in progress)
- Sleep-considerate IT app/server coordination
- Lightweight support for persistent cloud apps
- Change remote file access model

Us: Quick Overview

- **Reaction Policy:**
  - Wake on **incoming TCP connections**
- **Great** consolidation ratio
  - Unmodified **server** (1000’s)
  - Low power box (100’s, maybe 1000’s)
  - Peered proxy (100’s)
- Almost **no** client change
  - Daemon to send notification packets
  - Client **OS agnostic**
- Allows for **lots of sleep** in the enterprise

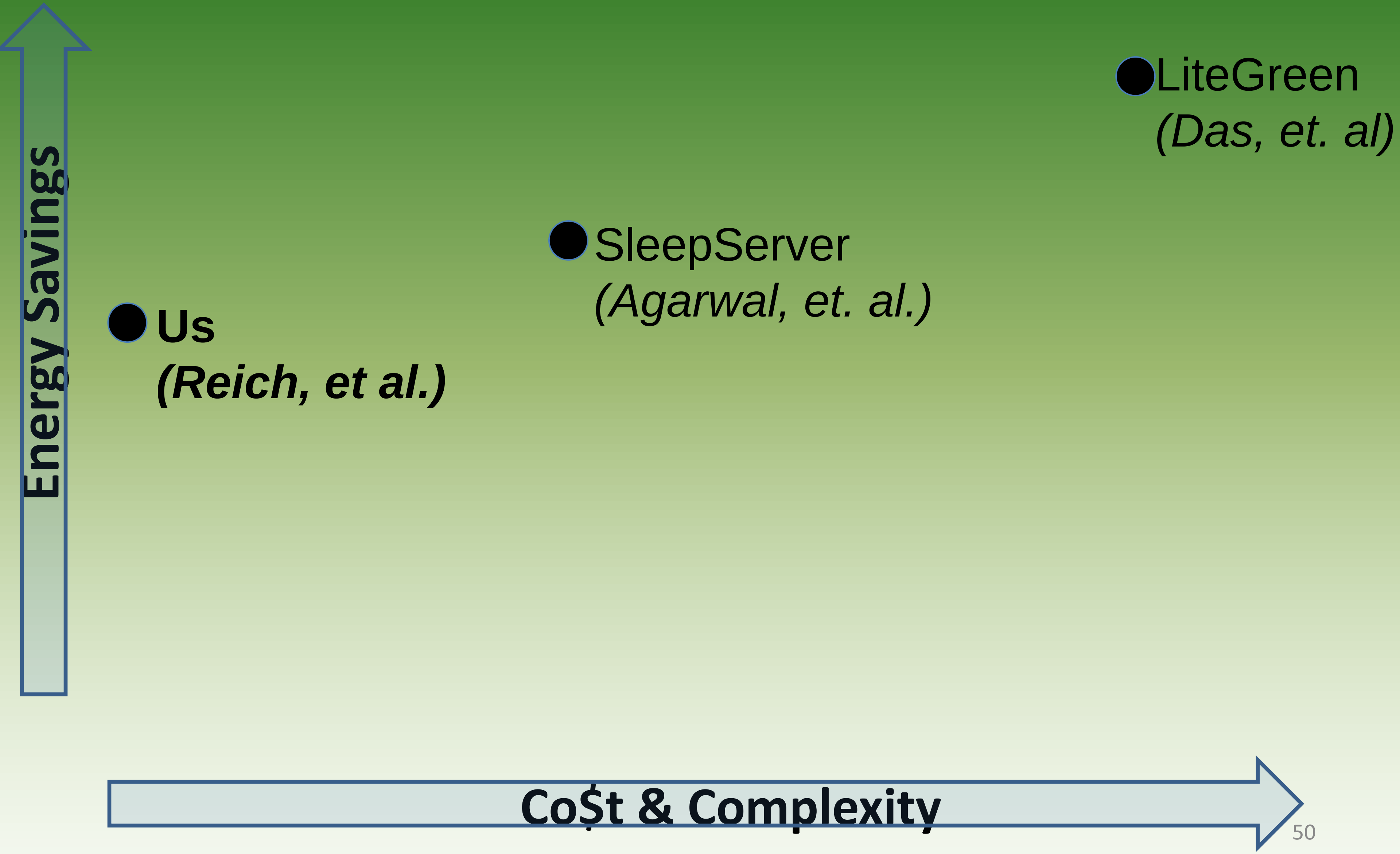
Comparison w/ **SleepServer**

- **Reaction Policy:**
  - Respond to **stubbed apps**
- **Good** consolidation ratio (100’s)
  - Unmodified **server**
- **Moderate** client change
  - Code, test, install **stub-aware apps**
  - Transfer **state / data**
  - **Credential** transfer
    (which can get complicated in enterprise)
- **Some additional sleep** in enterprise, potentially more in non-enterprise settings

Comparison w/ **LiteGreen**

- **Reaction Policy:**
  - Respond to **everything**
  - **Except** computational intense processes, local disk
- **Middling** consolidation ratio (10’s)
  - Powerful server + lots of RAM
- **Huge** client-side / network changes
  - **Virtualize OS**
  - **RDP even into local machine**
  - Move most locally stored data **onto SAN/NAS**
  - **Install Gigbit backbone** (if you don’t have already)
- A **good deal more** additional sleep opportunity (can deal w/ crying babies and even some IT apps)

Comparison w/ Other Work



LiteGreen
(Das, et. al)

SleepServer
(Agarwal, et. al.)

Us
(Reich, et al.)

Energy Savings

Co$t & Complexity

Questions & Answers

Isn’t This Just Your Network?

- Yes. We only have **empirical evidence** from our own deployment at Microsoft Research
- But we believe other nets **qualitatively similar**
  - **Functionally similiar**: security scans, patches, etc.
  - Related work (e.g., Nedevschi 2009)
  - **Anecdotes** from other researchers
- Of course, we are in the process of **verifying**
  - Let us know if you’d be interested in testing on your network!

Isn’t This Too Simple?

- No.
  Compared to other published approaches our is
  - **Less costly** to **deploy**
  - **Easier** to **maintain**
- We provide **cost effective power savings**
- The real question: why would you want to make things **more complicated than necessary**?

Why Not Built-In NIC Capabilities?

- Generality
  - – Old machines may not support patterns
  - – Complex network may require too many patterns
  - – Setting up pattern support may require
    - Fiddling w/ BIOS, other system settings
    - Non-uniform APIs
- Extensibility
  - – Wake on swipe, GPS coordinates
- Monitoring
- Can discard dedicated hardware w/ P2P anyway

Hasn’t This Already Been Done?

- (answer on next two slides)

What Isn’t Novel

- Suggesting a sleep proxy (1998)
- Comparing reaction policies (2009)

What is Novel

- **Build** on previous work
  - Adopt policy Nedevschi 2009 **predicted best**
  - **Improved on it** to support dynamic apps
- Focus on **economic feasibility**
- **Deploy on operational corporate network**
- **Learn lessons**
  - **Insomnia** is actually biggest problem
  - Economical solution **isn’t better reaction policies**