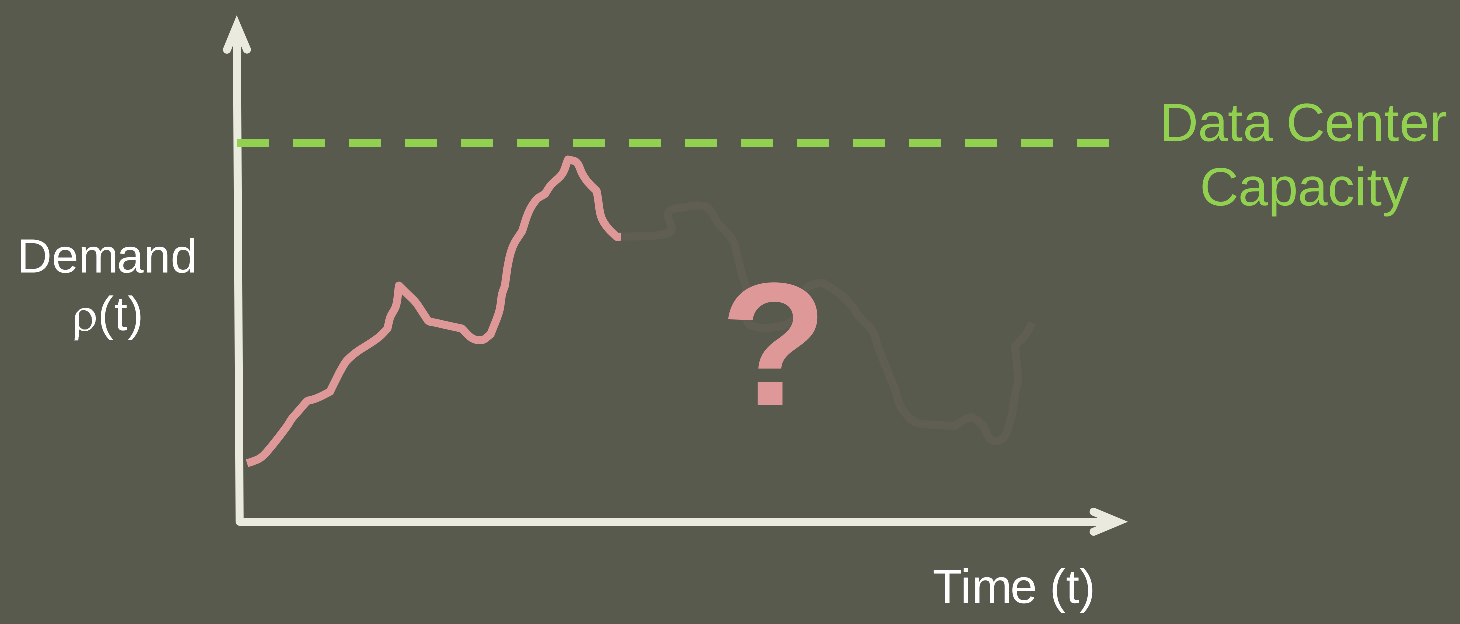
Energy-efficient Dynamic Capacity Provisioning in Server Farms

VARUN GUPTA
Carnegie Mellon University

Partly based on joint work with:

Anshul Gandhi (CMU)
Mor Harchol-Balter (CMU)
Mike Kozuch (Intel Research)

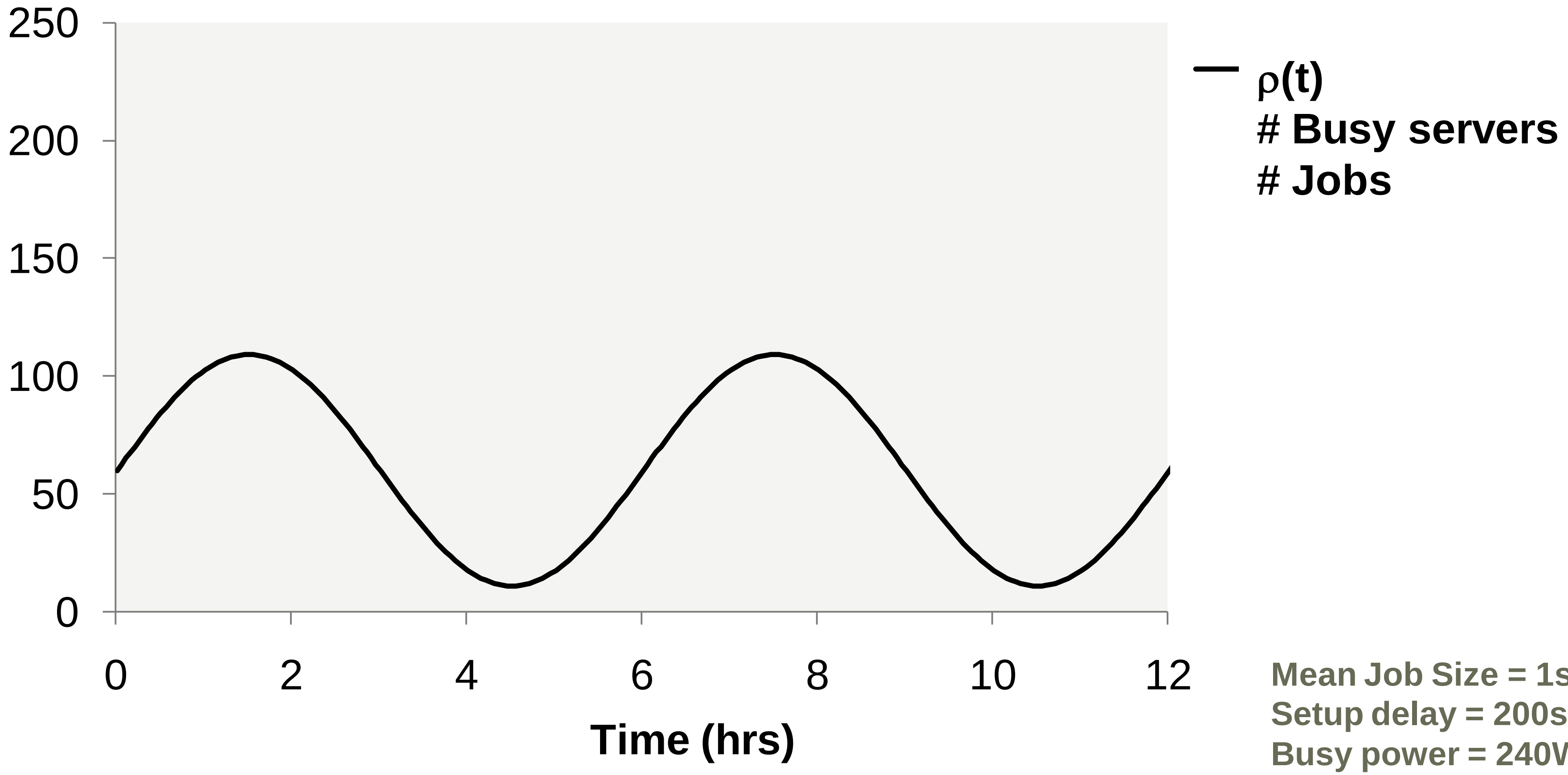
The "provisioning for peak" problem



PROBLEM: Want to turn servers OFF/ON to match ρ(t)…

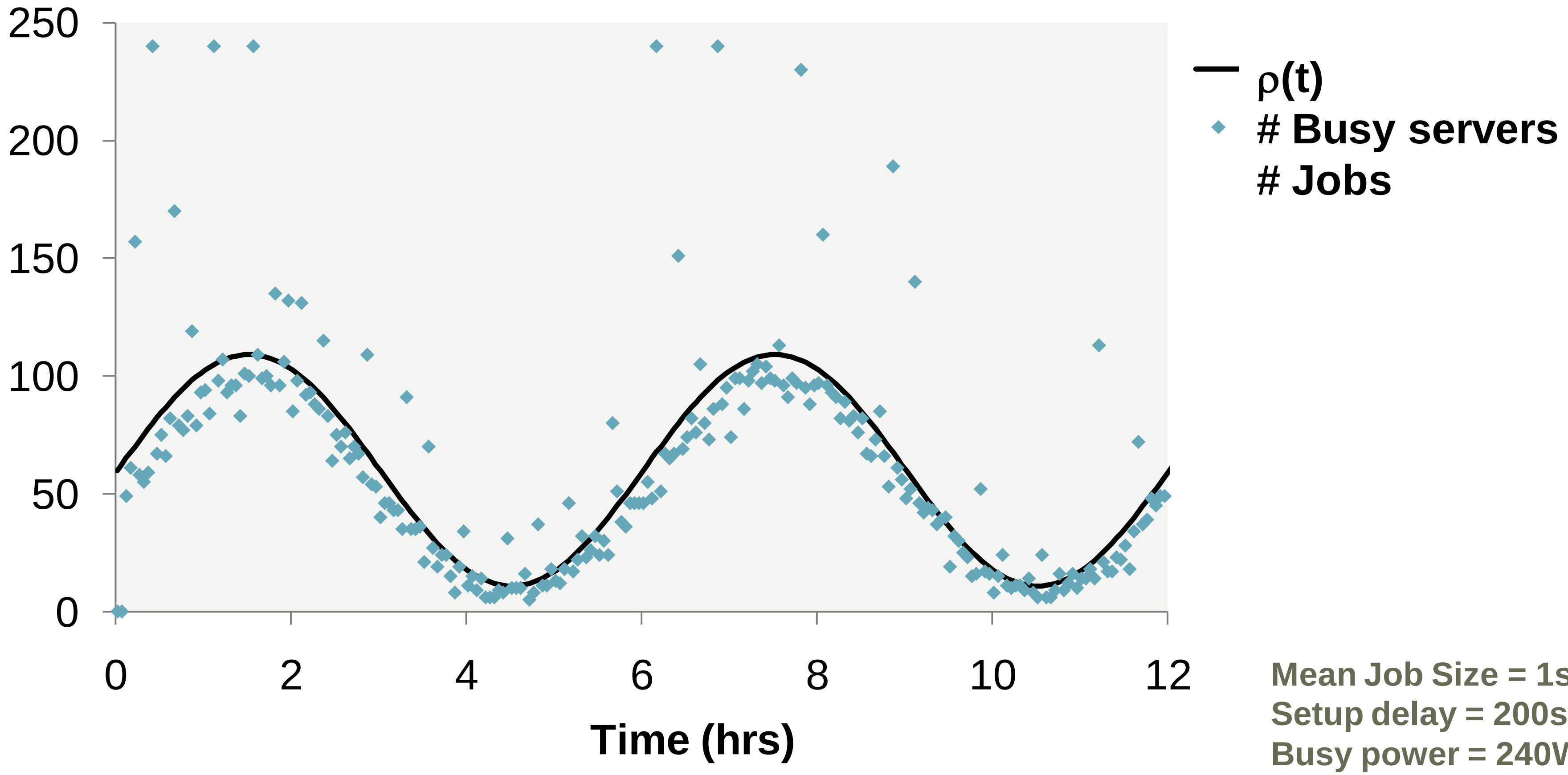
… and also minimize setup penalties!

First Attempt: ON/OFF policy



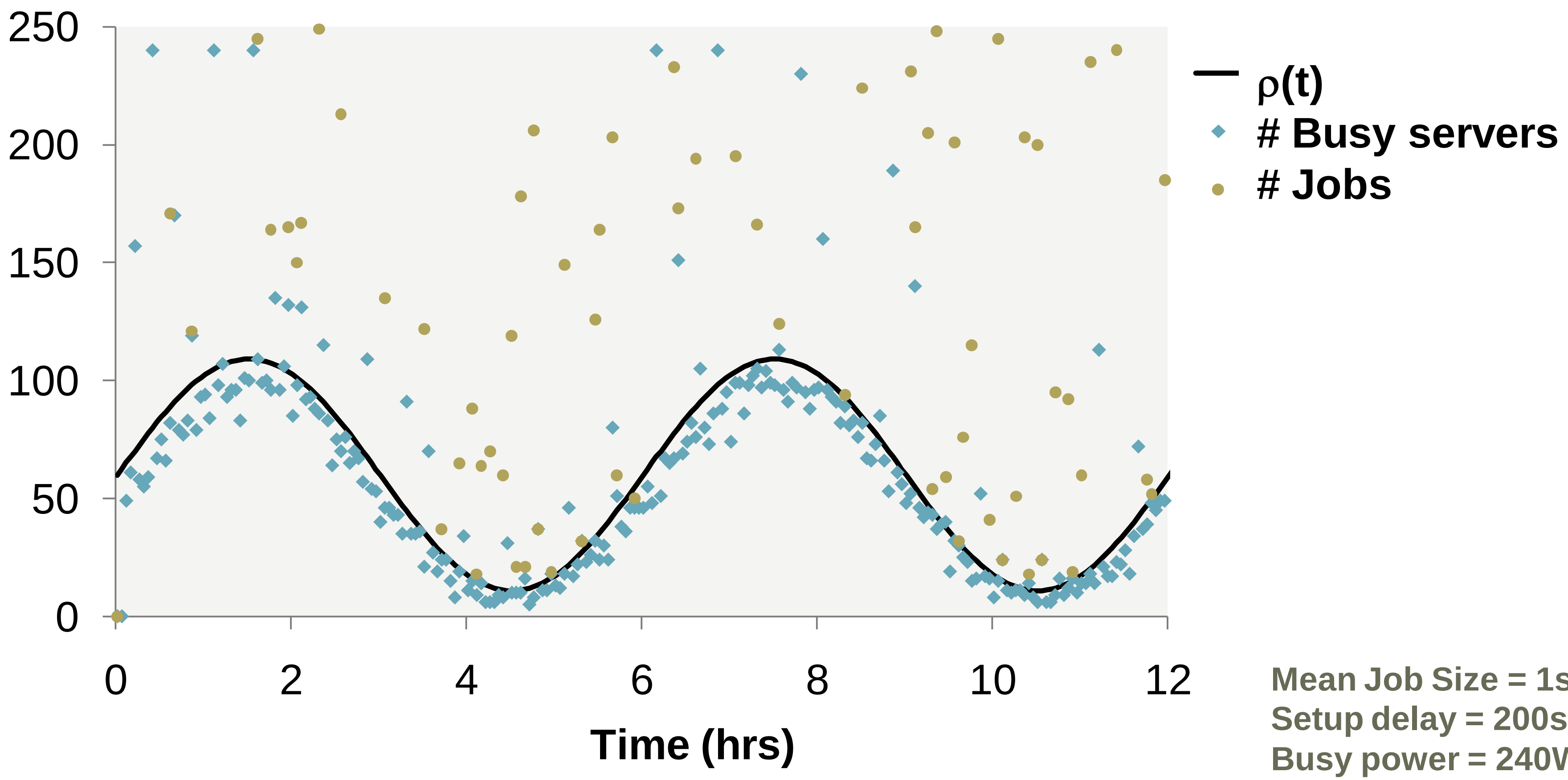
Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

First Attempt: ON/OFF policy



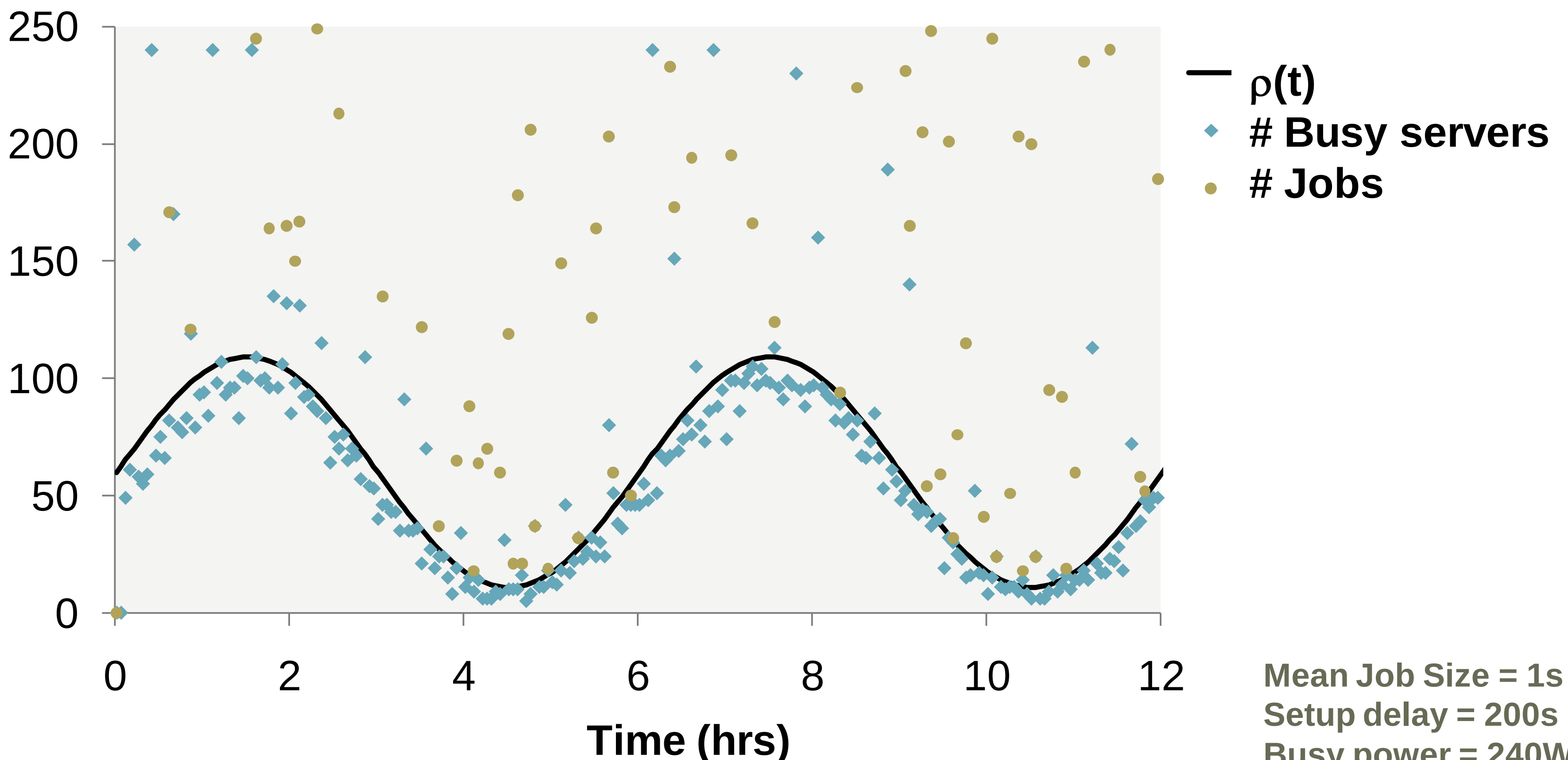
Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

First Attempt: ON/OFF policy



Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

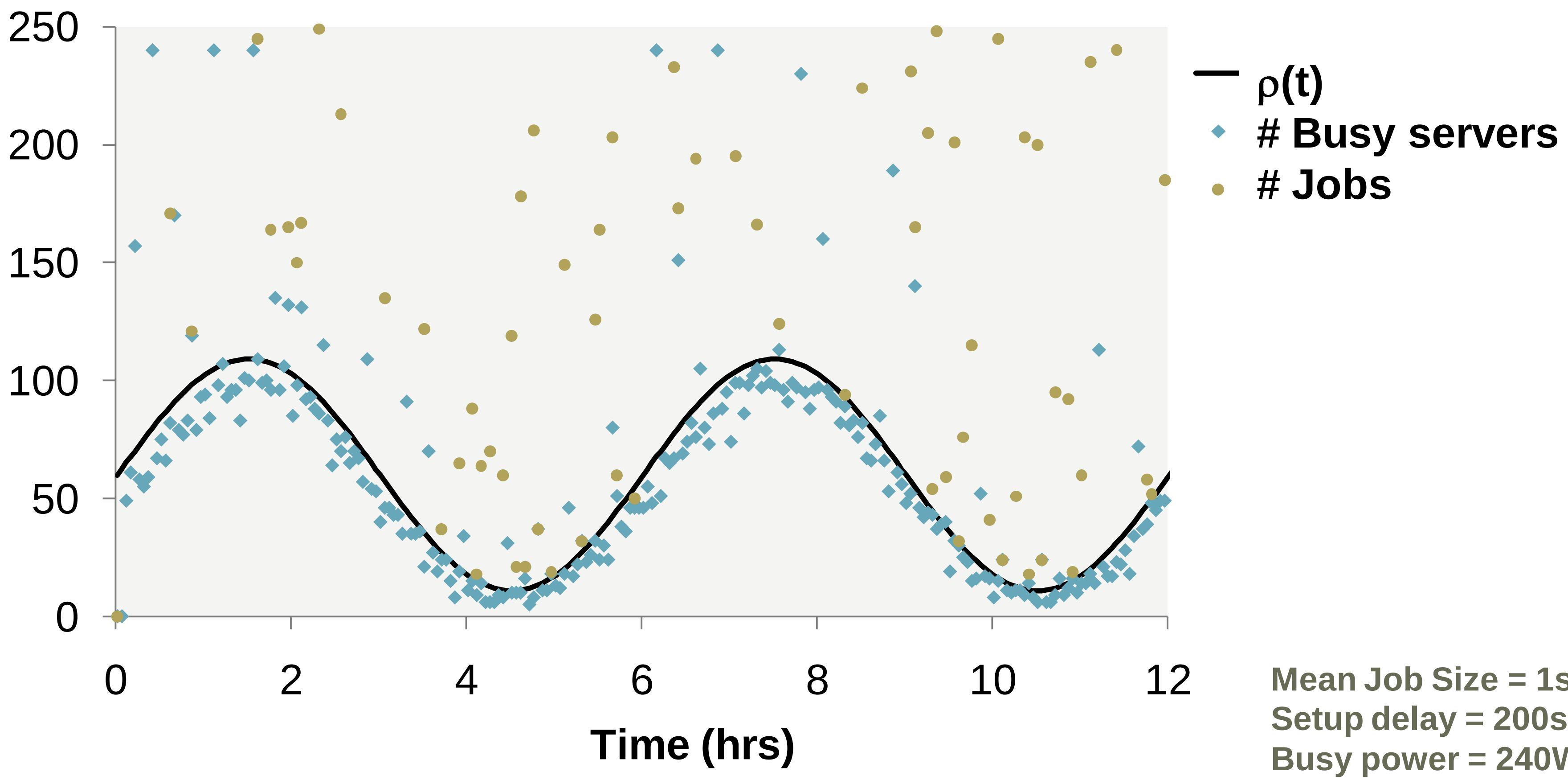
First Attempt: ON/OFF policy



Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

WITH SETUP: E[Power] = 51.9 kW E[Response time] = 13s
NO SETUP: E[Power] = 14.4 kW E[Response time] = 1s

First Attempt: ON/OFF policy



Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

Can we do better than ON/OFF?

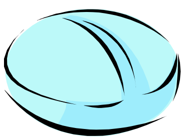
Add inertia while turning servers OFF

Our Prescription: DELAYEDOFF

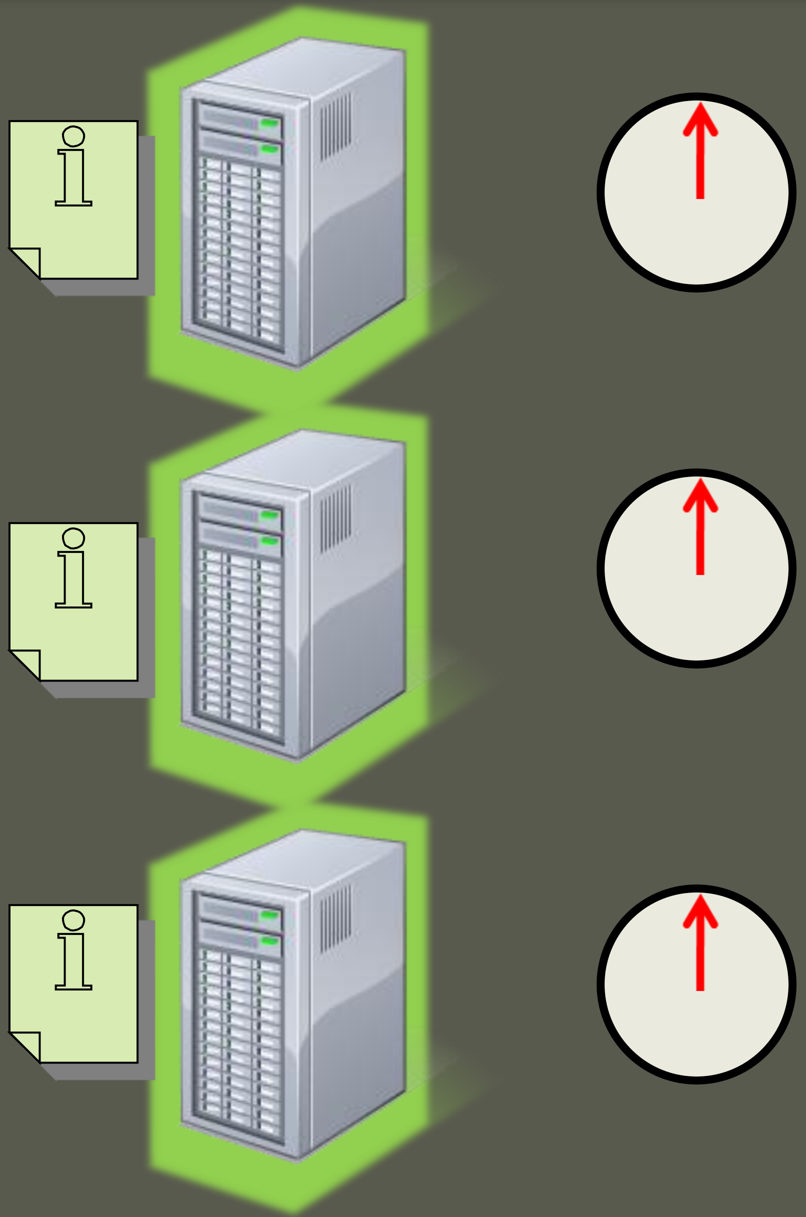
Turn OFF a server after idle for t_{wait}

t_{wait} independent of $\rho(t)$!

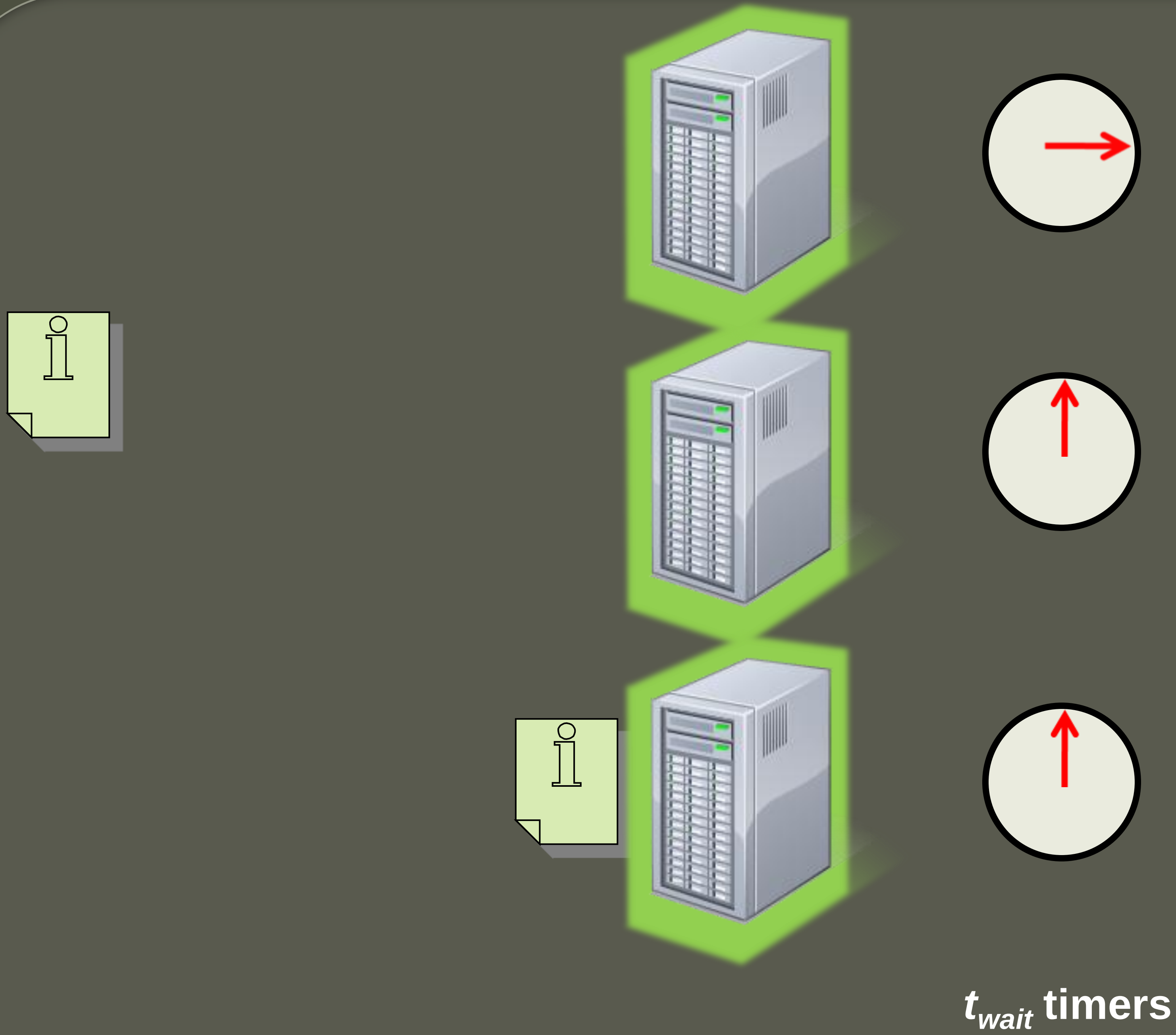
new arrival routed to the
most recently busy (MRB) server

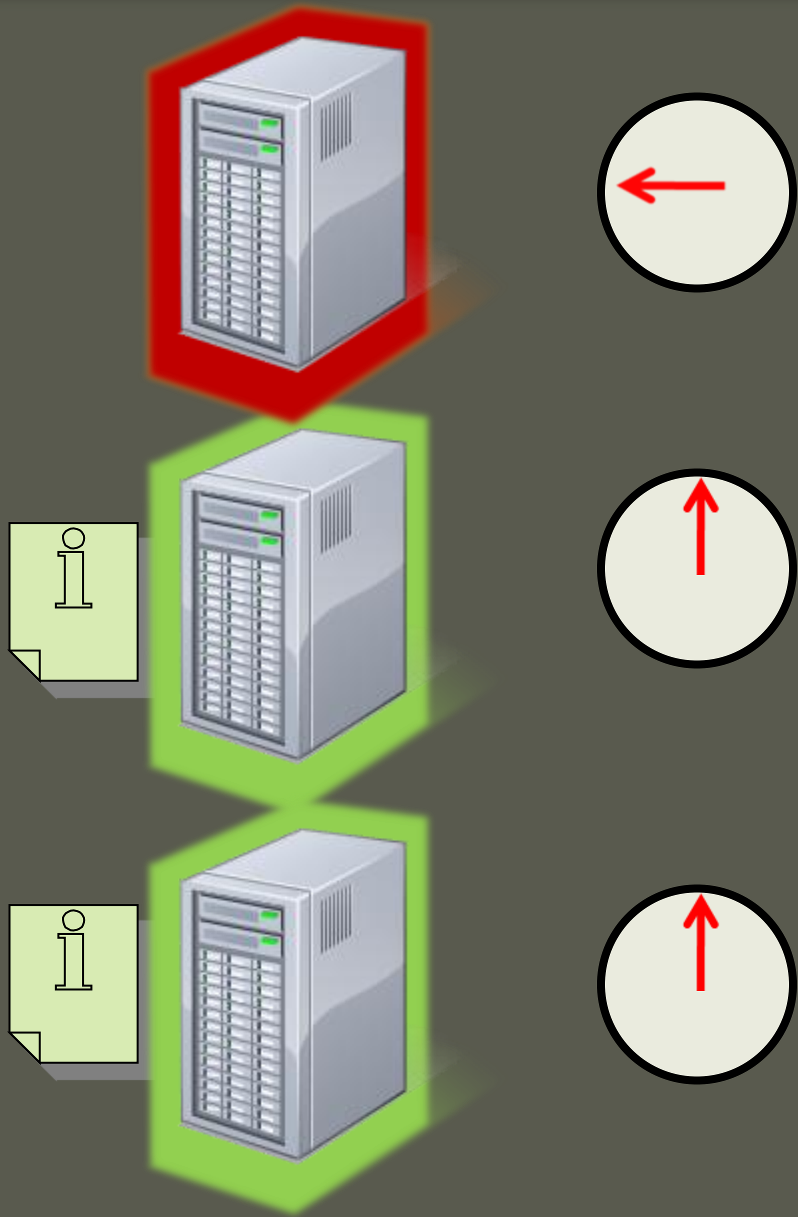


arriving job turns a server ON
if all servers busy



t_{wait} **timers**


t_{wait} timers

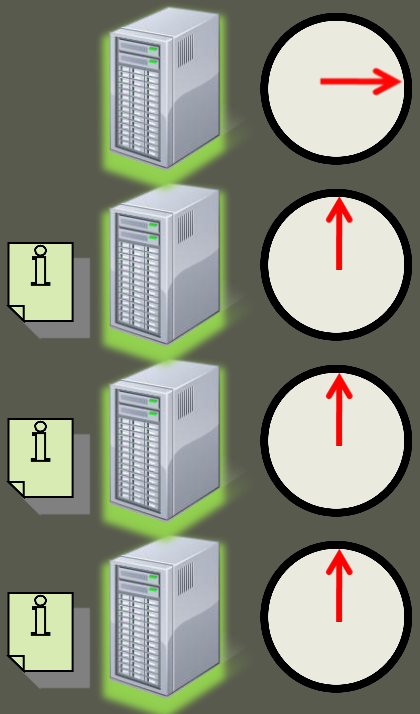


t_{wait} **timers**

Our Prescription: DELAYEDOFF

THEOREM: For Poisson arrivals, as the load $\rho \to \infty$, the number of ON servers is concentrated around $\rho + \sqrt{\rho \log \rho}$

--- Proof Intuition ---

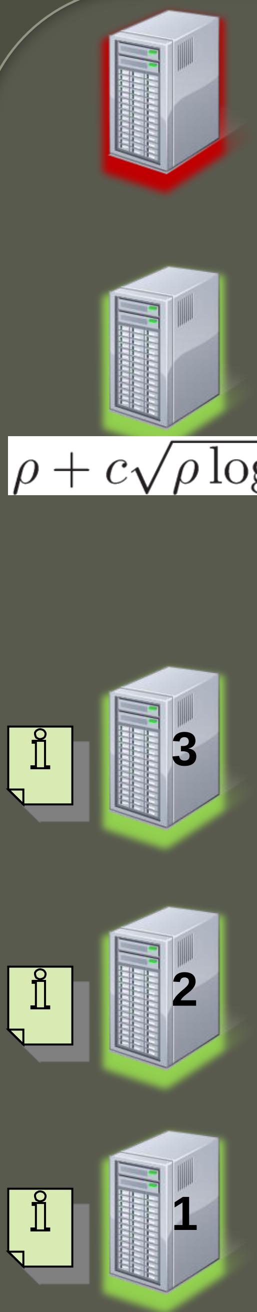


Step 1: An equivalent system view

k jobs run on the “first” k servers

Step 2: Analysis of idle periods of M/G/∞

Time from a k→(k-1) to the next (k-1)→k transition



Intuition for idle periods

1. # jobs $\approx$ Normal with mean and variance ρ

2. $Pr[\text{jobs} \geq \rho + c\sqrt{\rho \log \rho}] \quad \propto \quad \frac{1}{\sqrt{\rho}} e^{-\frac{c^2 \log \rho}{2}}$

$$\propto \quad \frac{1}{\rho^{\frac{c^2+1}{2}}}$$

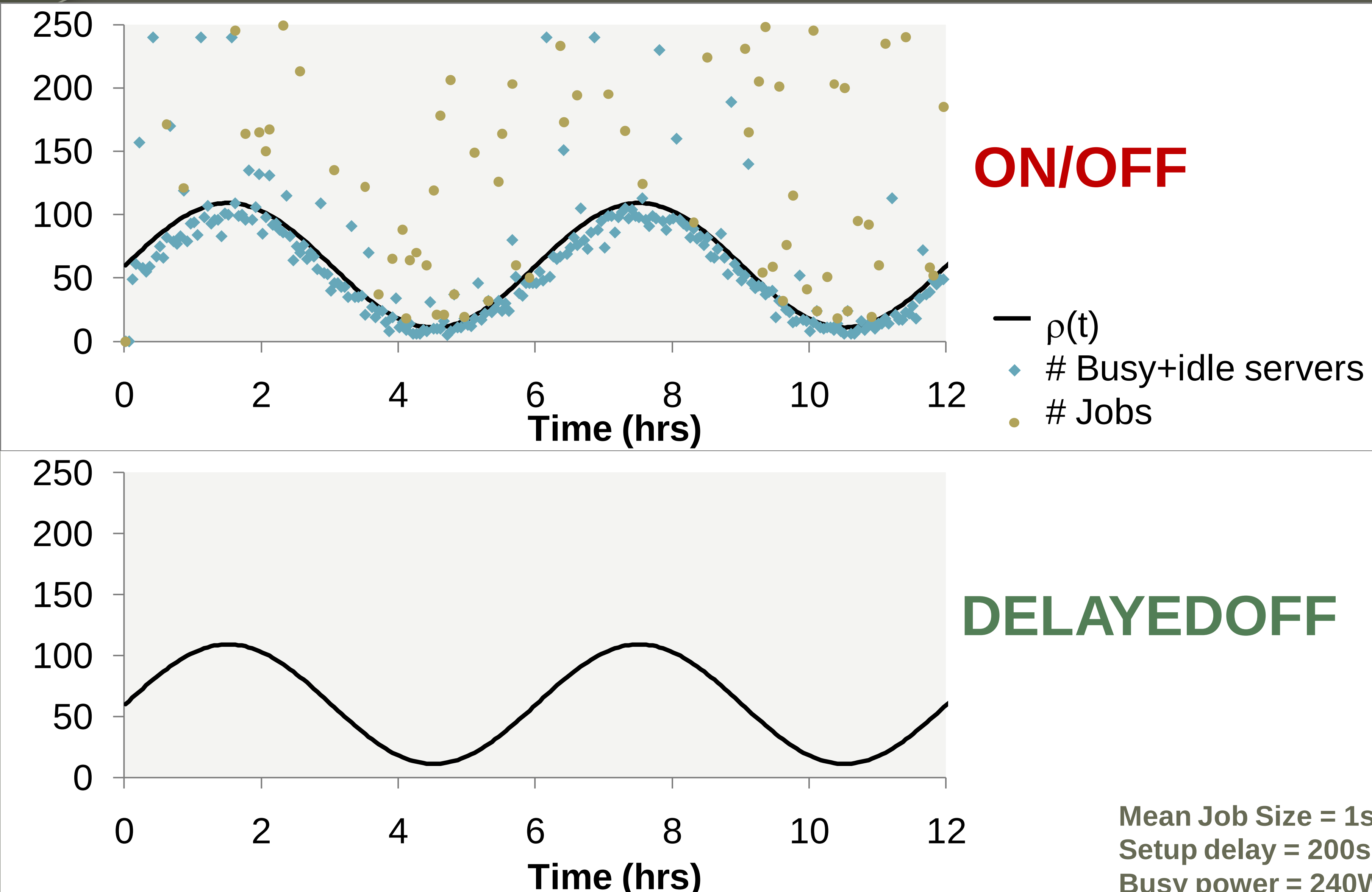
3. Events happen at rate ρ

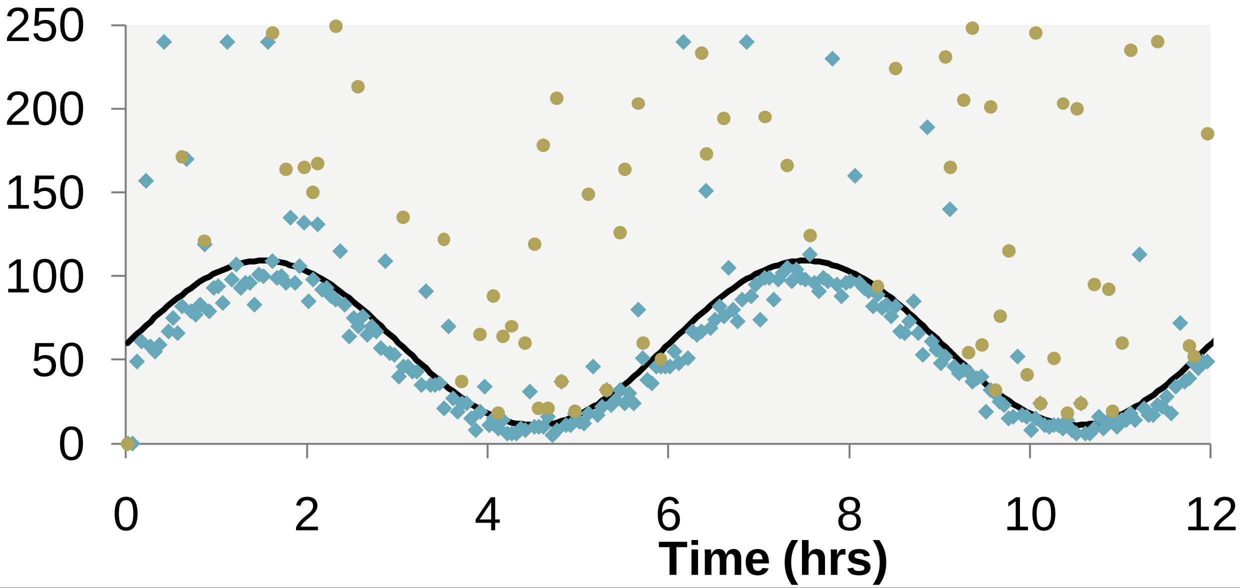
4. Mean idle period of $\rho + c\sqrt{\rho \log \rho}$ server $\propto \frac{\rho^{\frac{c^2+1}{2}}}{\rho}$

$$\propto \rho^{\frac{c^2-1}{2}}$$

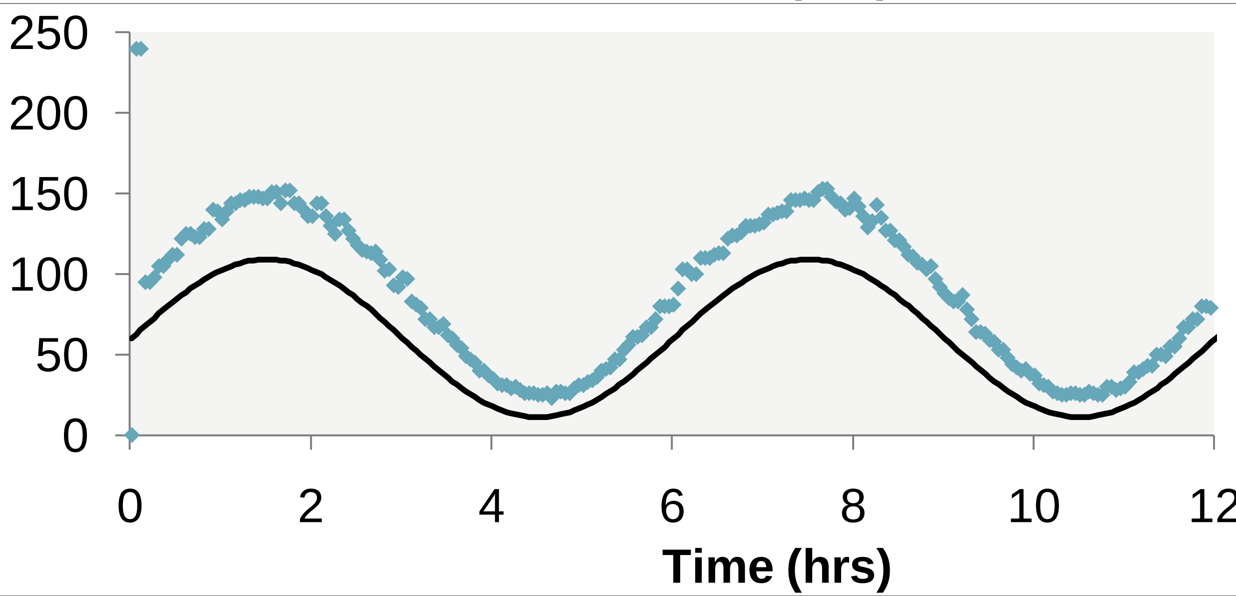
MRB $\Rightarrow \rho + (1 \pm \epsilon)\sqrt{\rho \log \rho}$ servers for any constant t_{wait} !

In practice, we choose t_{wait} to amortize setup cost


ON/OFF
ρ(t)
Busy+idle servers
Jobs
Time (hrs)
DELAYEDOFF
Time (hrs)
Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

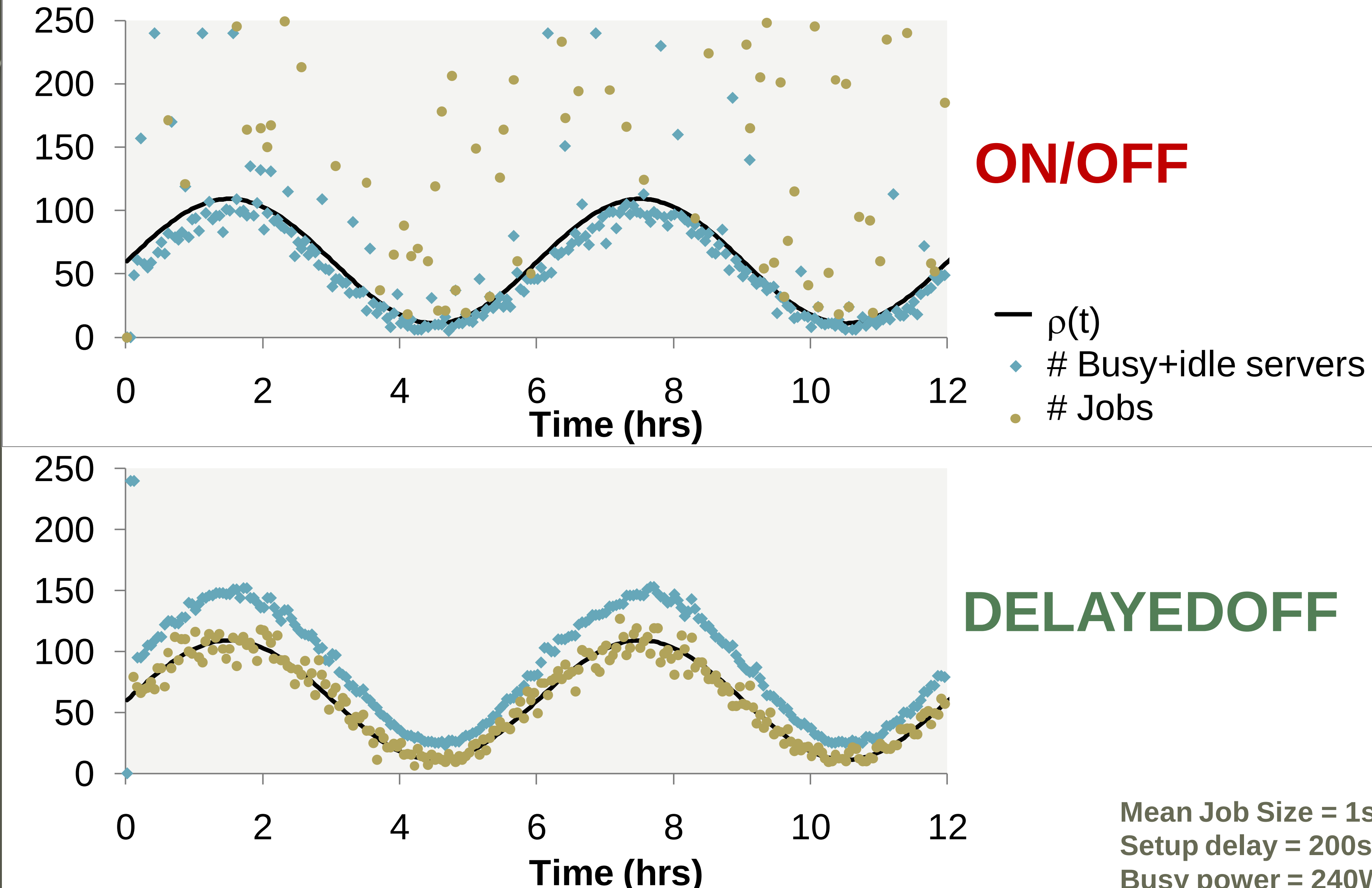


ON/OFF



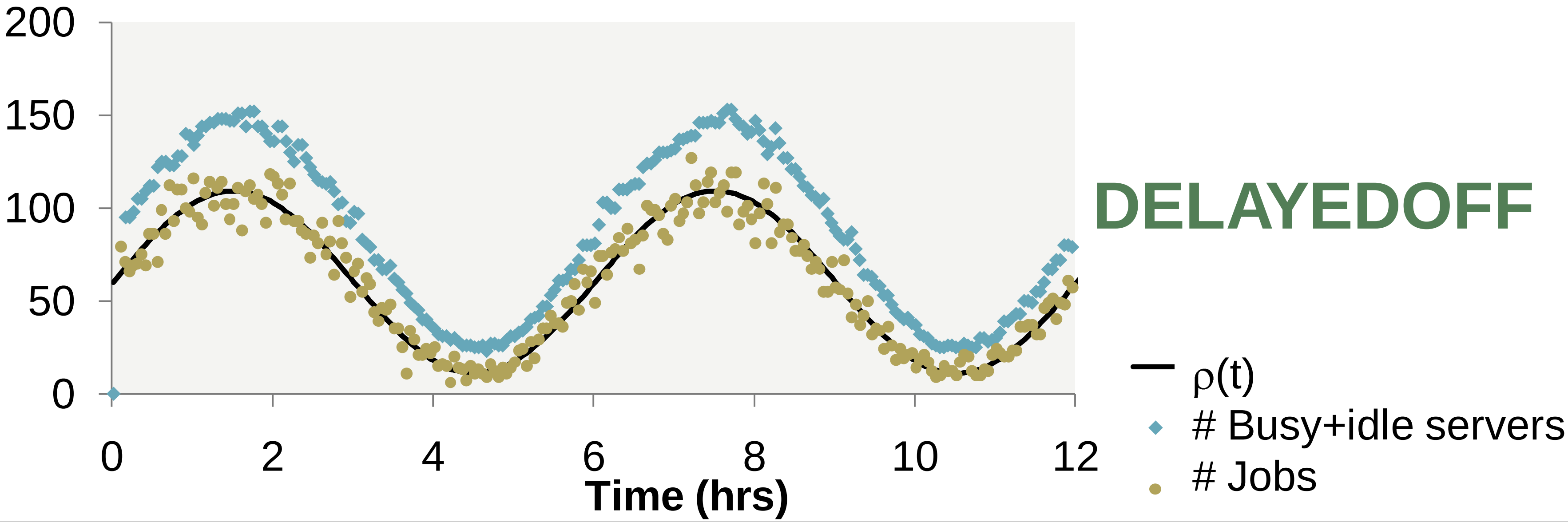
DELAYEDOFF

Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

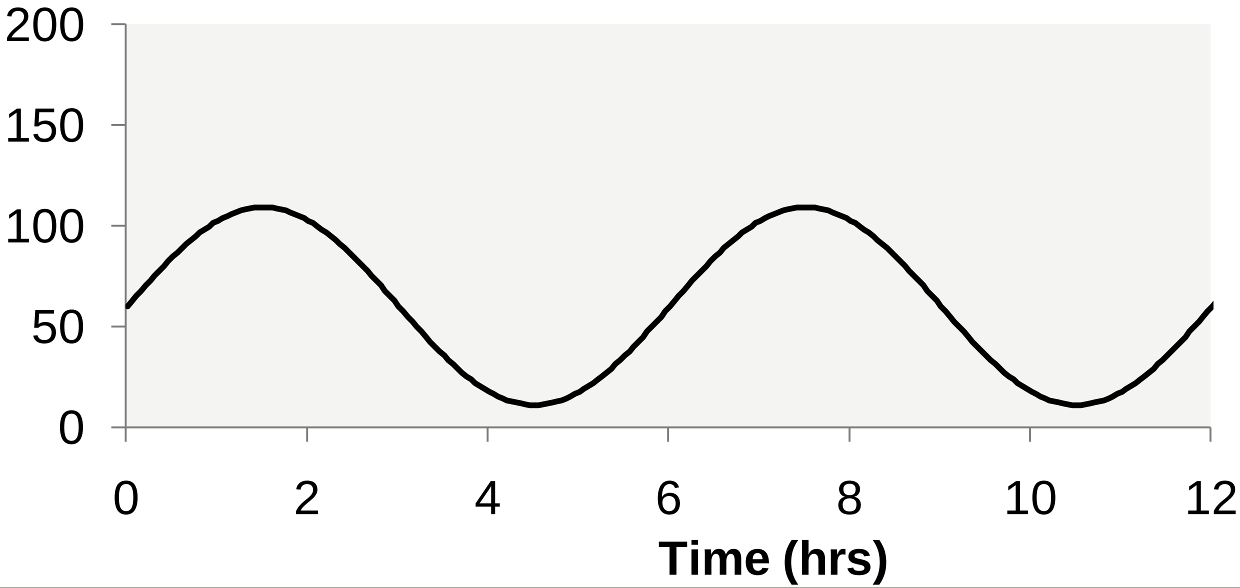


Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

DELAYEDOFF: E[Power] = 18.9 kW E[Response time] = 1.002s
NO SETUP: E[Power] = 14.4 kW E[Response time] = 1s



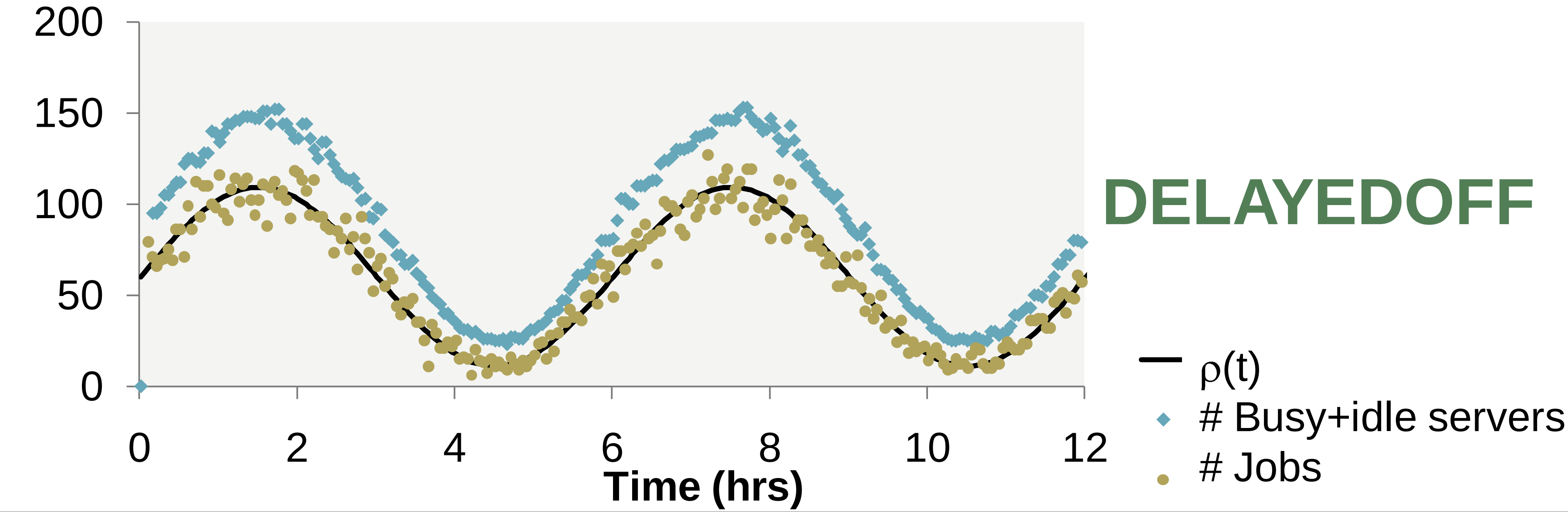
DELAYEDOFF

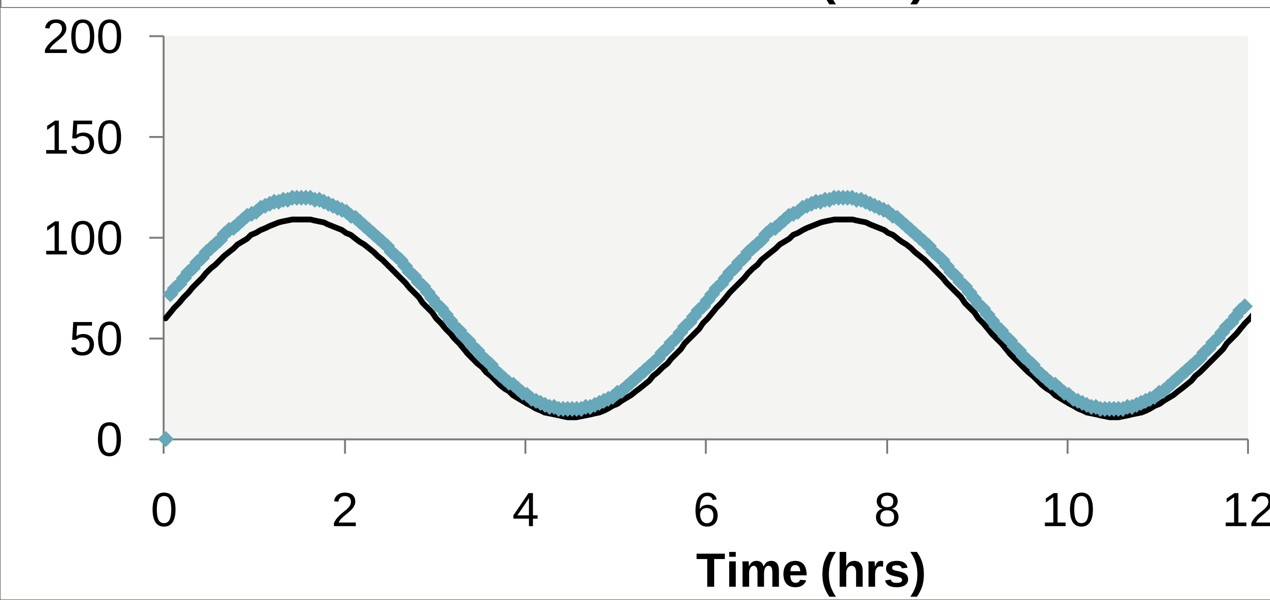


SQ. ROOT W/ LOOKAHEAD

(“optimal offline”)

Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

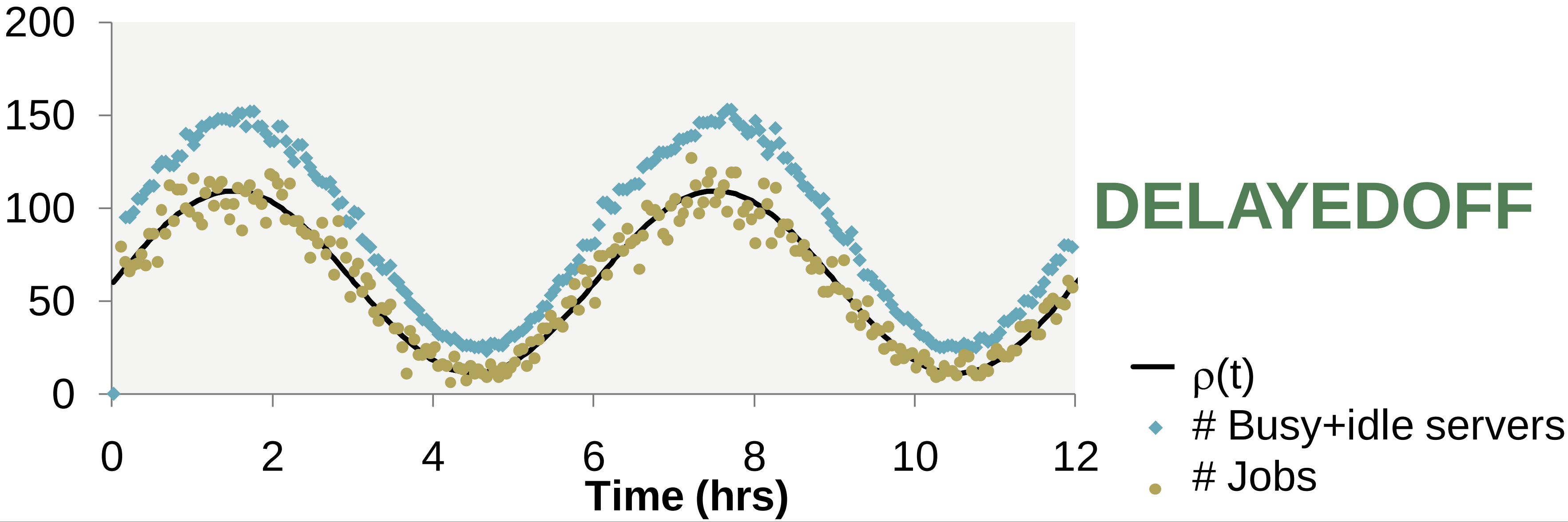


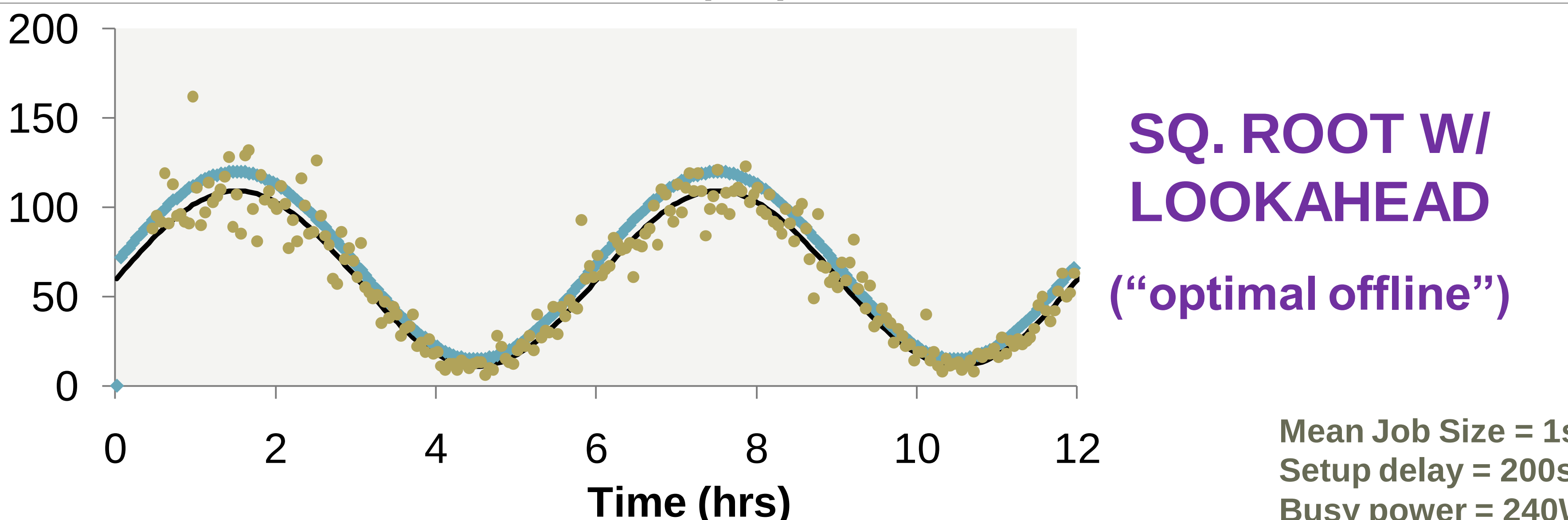


SQ. ROOT W/ LOOKAHEAD

("optimal offline")

Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W





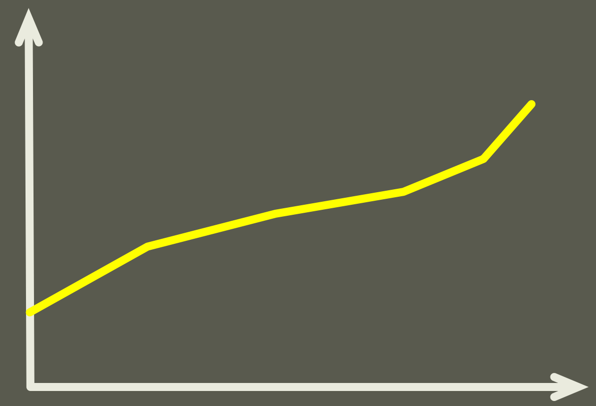
Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

DELAYEDOFF:	E[Power] = 18.9 kW	E[Response time] = 1.002s
√ w/LOOKAHEAD:	E[Power] = 15.8 kW	E[Response time] = 1.036s

DELAYEDOFF mods

1. Speed scaling algorithms
2. A simple proxy for MRB routing
3. Heterogeneous servers
4. Managing Virtual Machines in the Cloud
5. Wear-leveling/Performance tradeoffs

Effect of DELAYEDOFF on speed-scaling



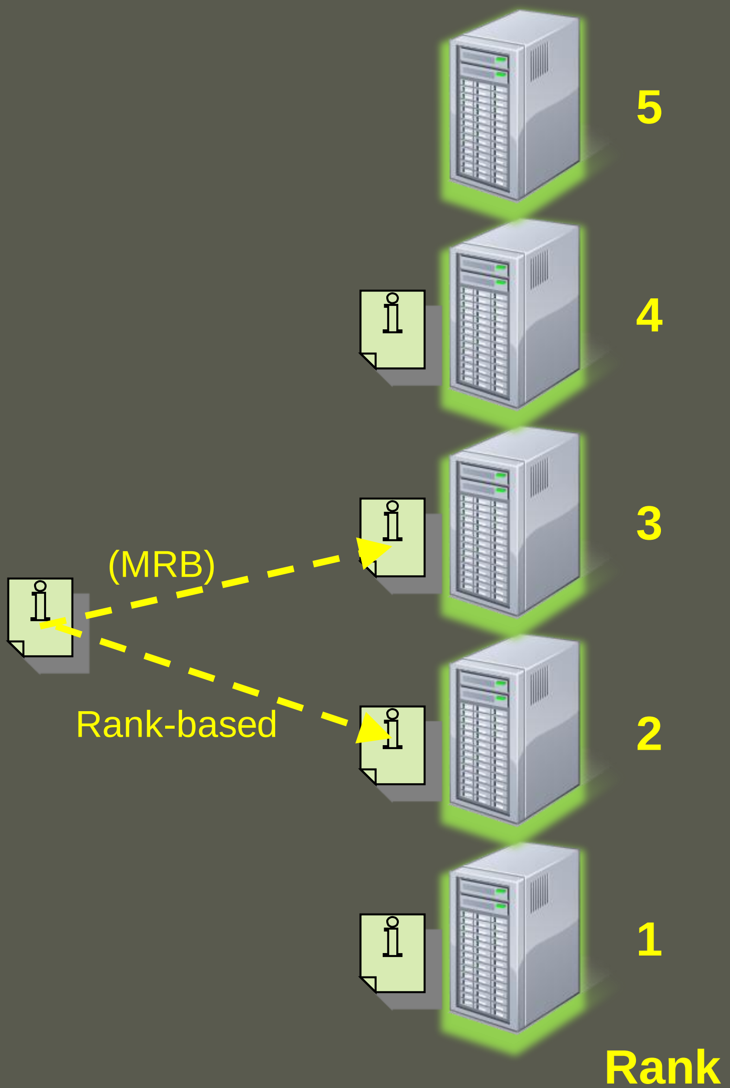
Q: Optimal speed to balance energy-performance?

A: your favorite metric = $\mathbf{F}(\mathbf{E}[\text{energy}/\text{job}], \mathbf{E}[\text{response time}])$

$$\mathbf{E}[\text{energy}/\text{job}] \xrightarrow{MRB} \frac{\mathrm{P(s)}}{\mathrm{s}} \qquad \mathbf{E}[\text{response time}] \xrightarrow{MRB} \frac{1}{\mathrm{s}}$$

$$\mathrm{s}^* = \operatorname{argmin}_{\mathrm{s}} \mathbf{F}\left(\frac{\mathrm{P(s)}}{\mathrm{s}}, \frac{1}{\mathrm{s}}\right)$$

A simple proxy for MRB routing

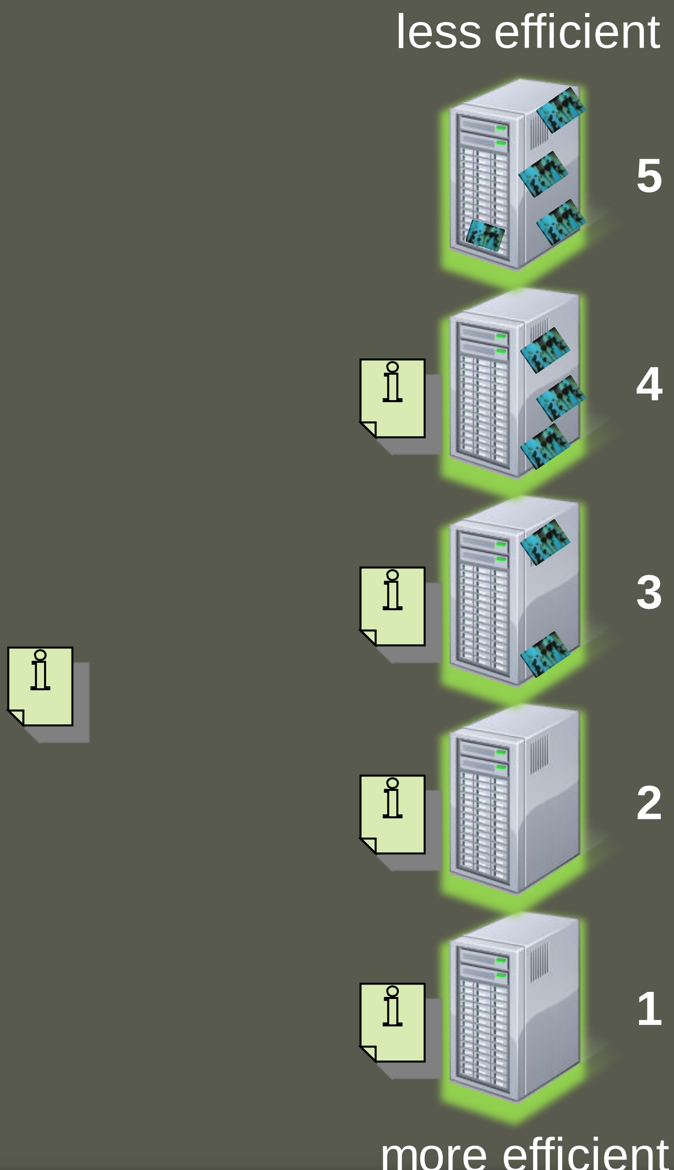


MRB requires a lot of state updates 

Proxy policy
- Assign static ranks to servers
- Route a new arrival to *highest ranked idle* server

Almost the same performance as MRB
+ easier to implement than MRB
+ easy to extend

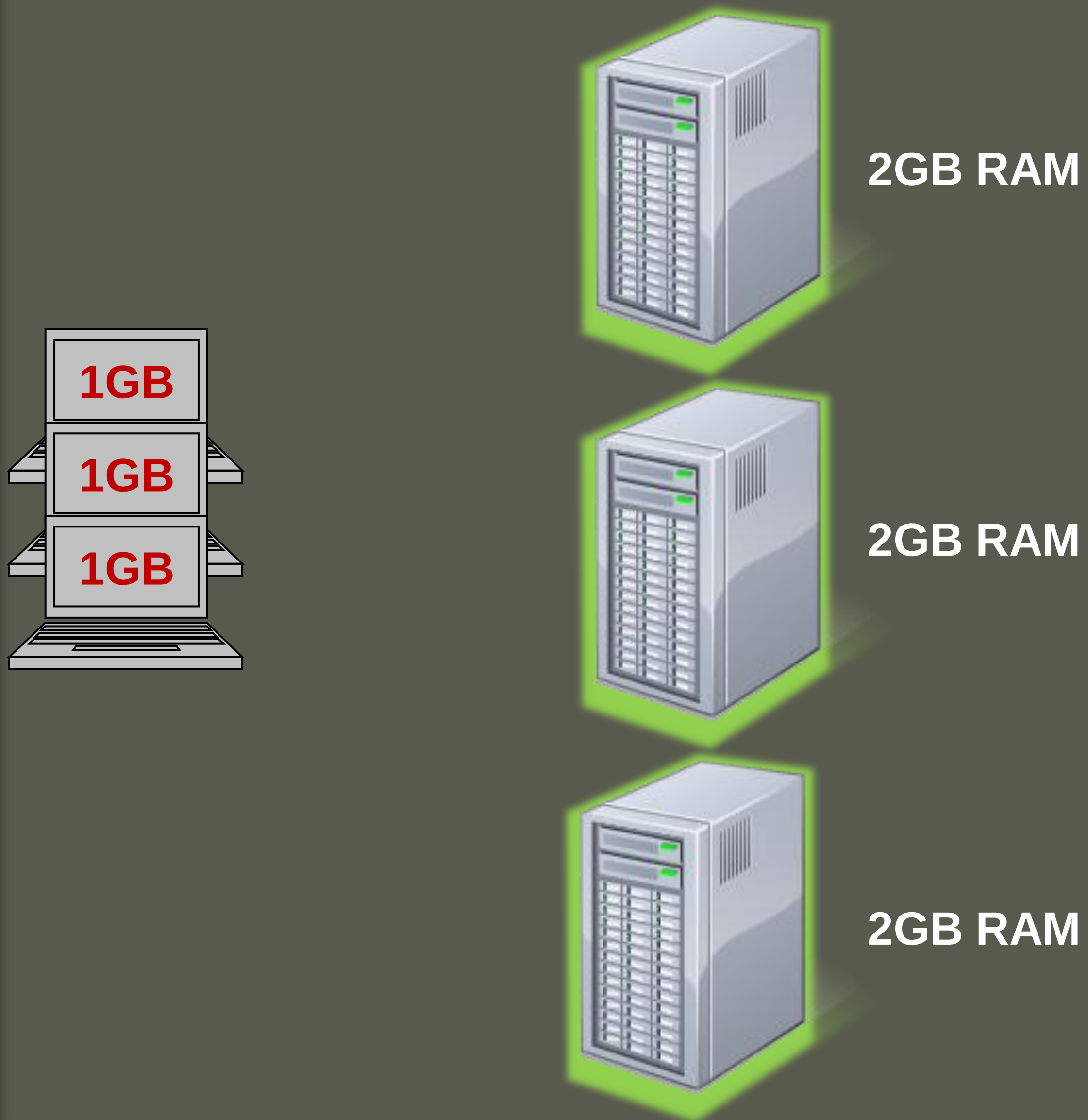
DELAYEDOFF for heterogeneous servers



Assign ranks to servers based on efficiency (1 = most efficient)

Route a new arrival to *highest ranked idle* server

Managing Virtual Machines in the Cloud



Managing Virtual Machines in the Cloud

Split physical servers into “virtual” servers

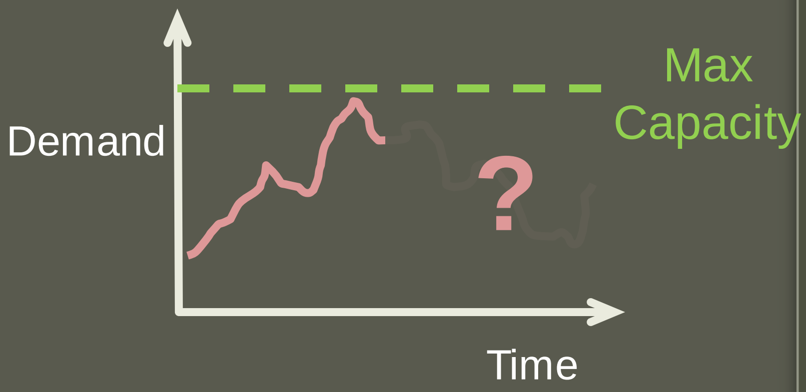
Assign static ranks to virtual servers

Route a new VM request to *highest ranked idle* virtual server

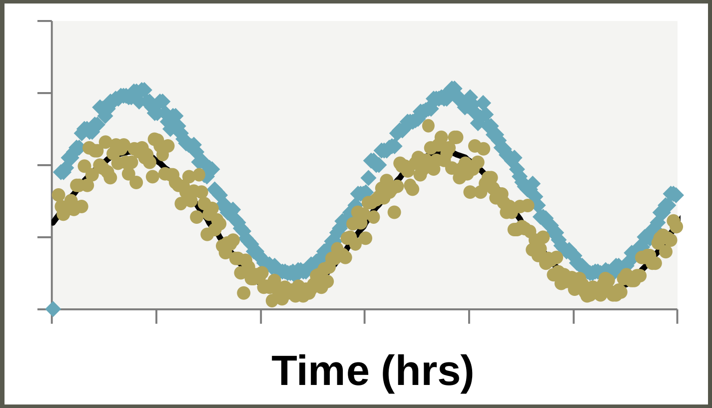
Turn the physical server OFF after each virtual server has idled for t_{wait}

Conclusions

- Problem: unpredictable demand and non-trivial setup costs



- DELAYEDOFF : A new traffic-oblivious capacity scaling scheme

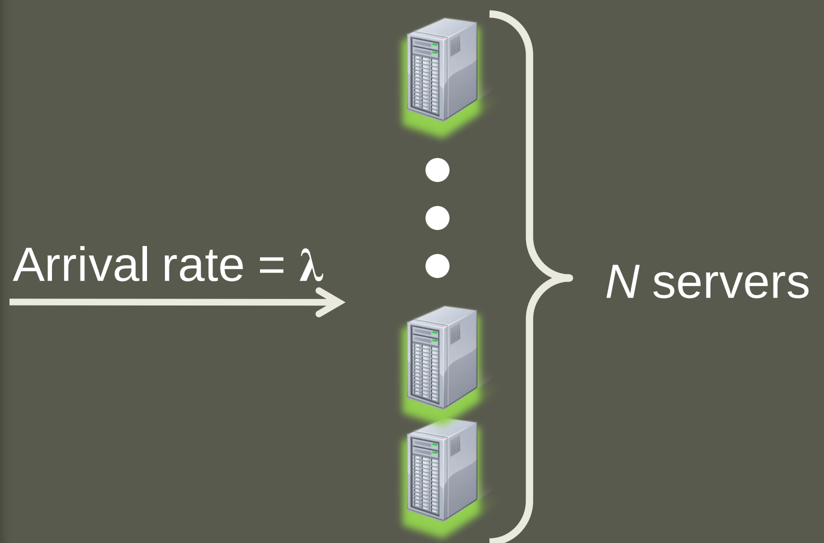


- Extensions to real-world scenarios

The Importance of Being MRB

THEOREM [MRB]: As the load $\rho \to \infty$, the number of ON servers is concentrated around $\rho + \sqrt{\rho \log \rho}$

THEOREM [Round-Robin]: As the load $\rho \to \infty$, for constant job sizes, the number of ON servers is $\rho \left(1 + \frac{t_{wait}}{\text{job size}}\right)$



avg. interarrival time