Power Management for Networks to Reduce Energy Consumption

David Wetherall
Intel Research & University of Washington

Joint work with:

Sylvia Ratnasamy (Intel Research)
Sergiu Nedevschi (UC Berkeley)
Lucian Popa (UC Berkeley) and
Gianluca Iannaccone (Intel Research)

Ask me about …

1. Wireless privacy for ubiquitous computing
   - 802.11 links that conceal addresses/identifiers
   - Use of directional antennas
2. RFID sensor networks
   - Protocols, flexible tags/readers
3. Maintaining connectivity during routing transitions
   - Intra-domain techniques for IP and MPLS

Internet measurement, denial-of-service, …

Motivation

Network energy consumption a growing issue

- Equipment increasingly power-hungry (power density)
- Rising energy costs (significant fraction of TCO)
- Environmental concerns (e.g., EnergyStar), …

Opportunity for conservation appears significant

- Networks are provisioned for peak load
- But typical utilization is low (Sprint ~15%, Intel ~2%)
- And idle-time power consumption remains high

Most network energy consumption is wasted!
Want it to scale with utilization, not capacity

This Talk

Key Idea: Let networks (ISP and customer) sleep for brief periods or slow down when lightly loaded to save energy

- Stand to halve power for networks at 10-20% load
- With no extra loss, minimal added delay
- And simple algorithms with plausible hardware support

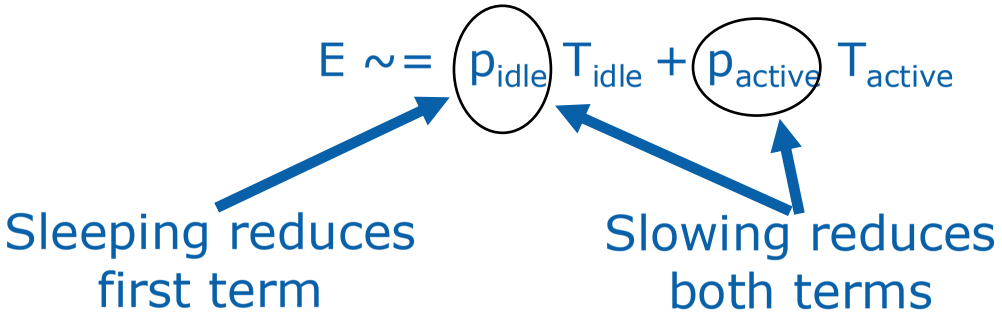
Analogy: PC use of sleep and performance states, e.g., ACPI

- Sleep states quickly power off components to different degrees
- Performance states run more slowly to save energy (DVS)
- OS uses these states to save energy from desktops to servers

We assume similar sleep/performance states for NICs, links, switches, and routers and consider how to best use them

Rationale

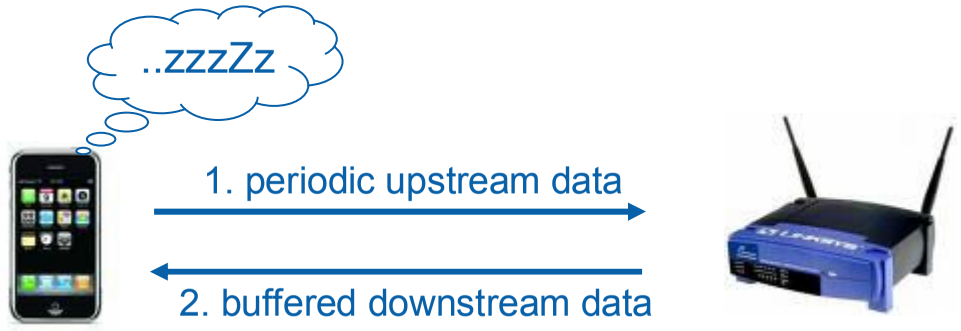
Total energy consumption:



Depend on platform support with good power profiles to save energy

Depend on careful use and rapid transitions to protect performance

Example: Sleeping for 802.11 clients



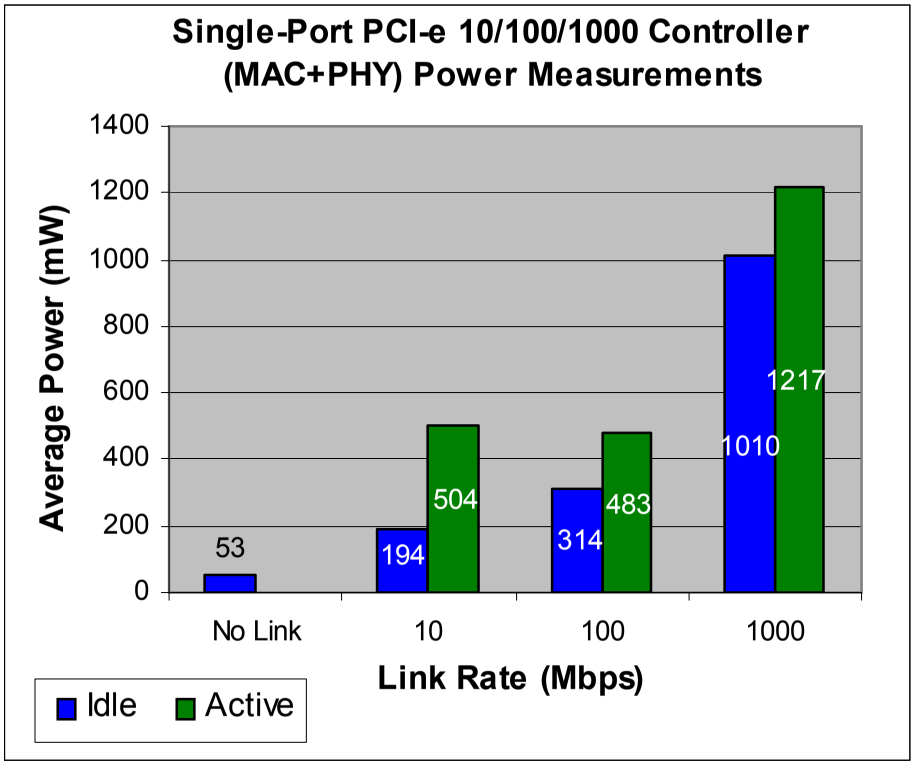
Mobile clients can sleep to extend lifetime; AP will buffer traffic

- 802.11 standard defines signaling protocols (power-save poll and automatic power save delivery) with rapid transitions (ms)

Energy-efficient Ethernet study group is standardizing analogous schemes for wired links; both ends can sleep

Energy Efficient Ethernet

Example: Multi-rate NICs



Slower rates use less power; holds for wireless too (Shannon)

- 802.11 rate transitions very rapid, wired currently slow (auto-negotiation)

Outline

1. Key questions and method
2. Sleeping
3. Rate adaptation (slowing down)
4. Comparison

For the details:

"Reducing Network Energy Consumption via Rate-adaptation and Sleeping," S. Nedevschi, et. al., NSDI 2008 (to appear)

1. Key questions and method

How much energy can be saved with what platform support?

Can we save significant energy without compromising performance?

Can we realize these savings with practical schemes?

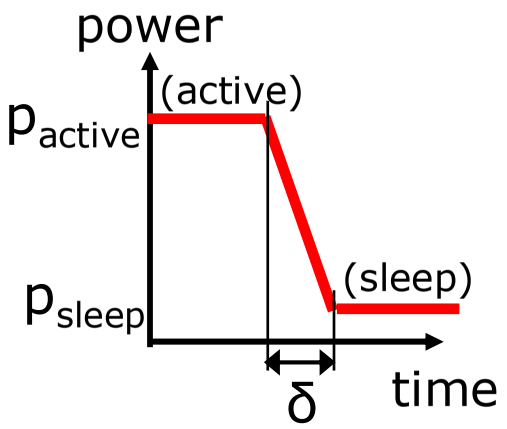
Methodology:

1. Assume basic sleep and rate models informed by practice
   - Includes empirical data from Intel NIC, Cisco GSR
2. Evaluate savings/performance with trace-driven simulations (ns)
   - Abilene and Intel topologies with macro-level traffic that we scale
3. Look for (unrealistic) bounds as well as practical schemes

2. Sleeping states

Model

- Single sleep state with power $p_{sleep} << p_{idle}$
- δ : transition period (ms)
- Timer or activity-driven wakeup
- Interfaces sleep independently

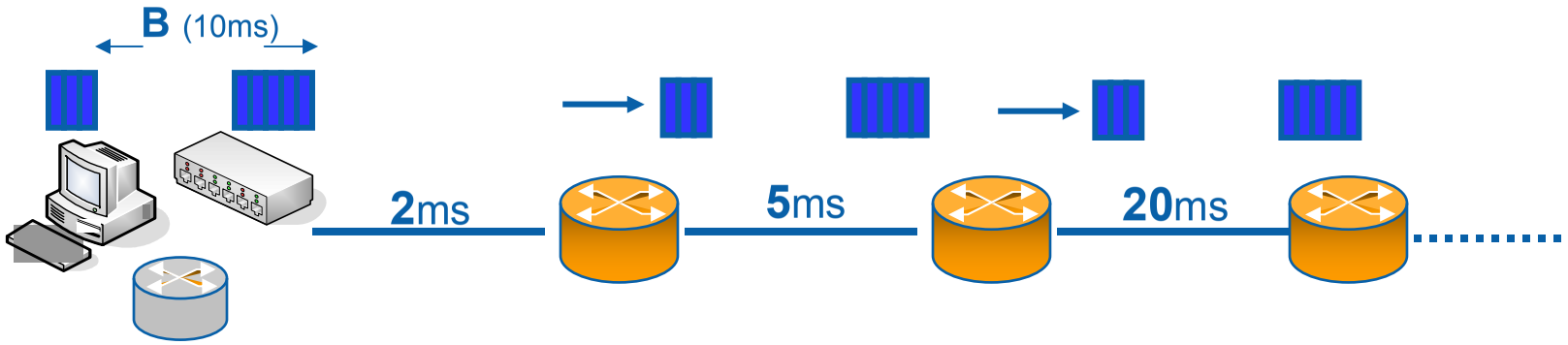


Metrics

- Energy savings in % time asleep (approx. savings if $p_{idle} \sim p_{active}$)
- Performance in loss and max delay

Making sleep gaps on links with buffer & burst (B&B)

Basic idea: use limited buffering at ingress to create predictable and useful sleep gaps (>2δ); do once, adds bounded delay



tx @ t=1
t=B+1
t=2B+1

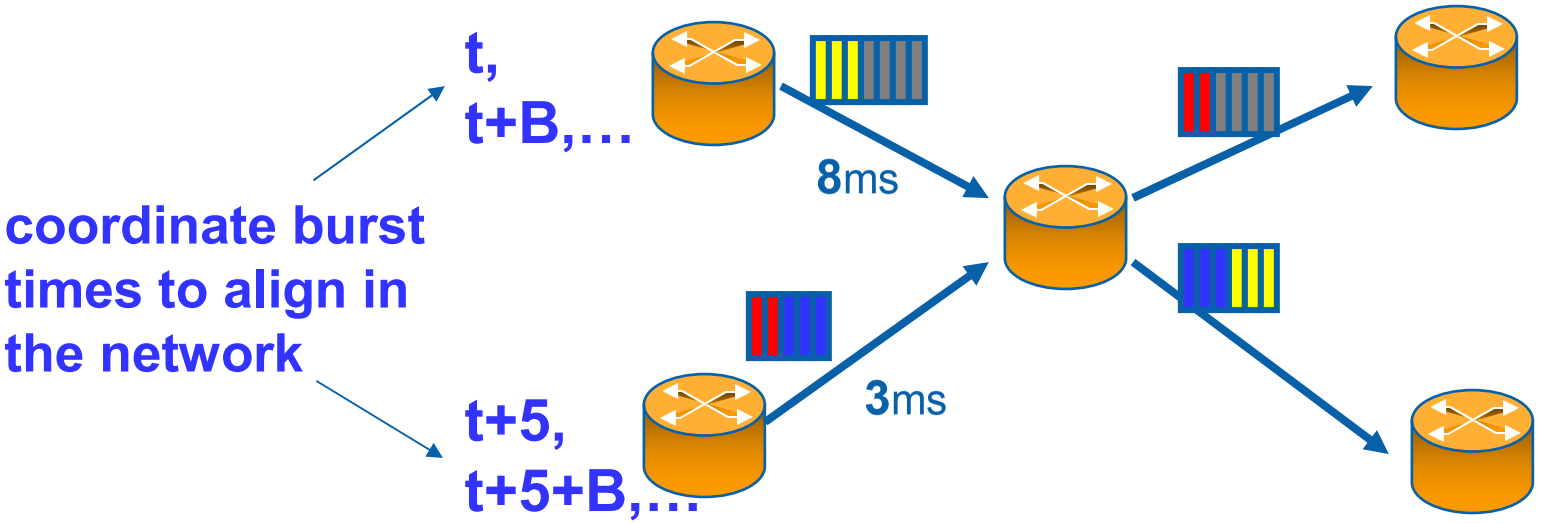
wake @ t=3
t=B+3
t=2B+3

@ t=8
t=B+8
t=2B+8

@ t=28
t=B+28
t=2B+28

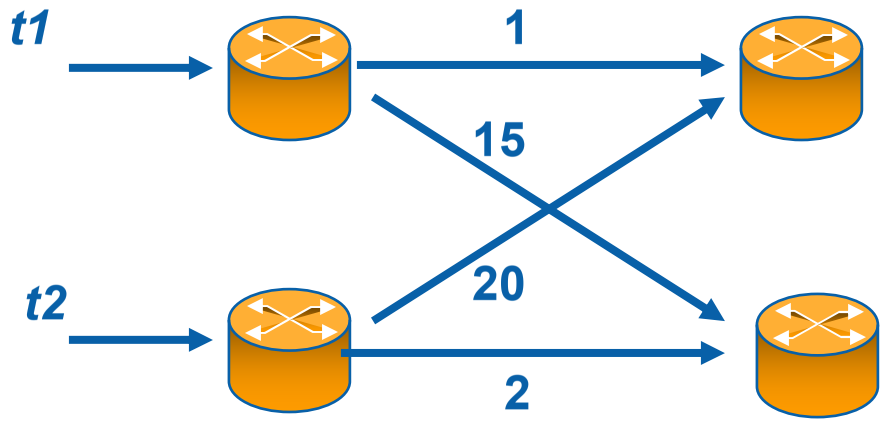
Want to preserve gaps across multiple ingresses

Basic idea: preserve bursts/gaps on links in networks by adjusting relative timing phase of different ingresses



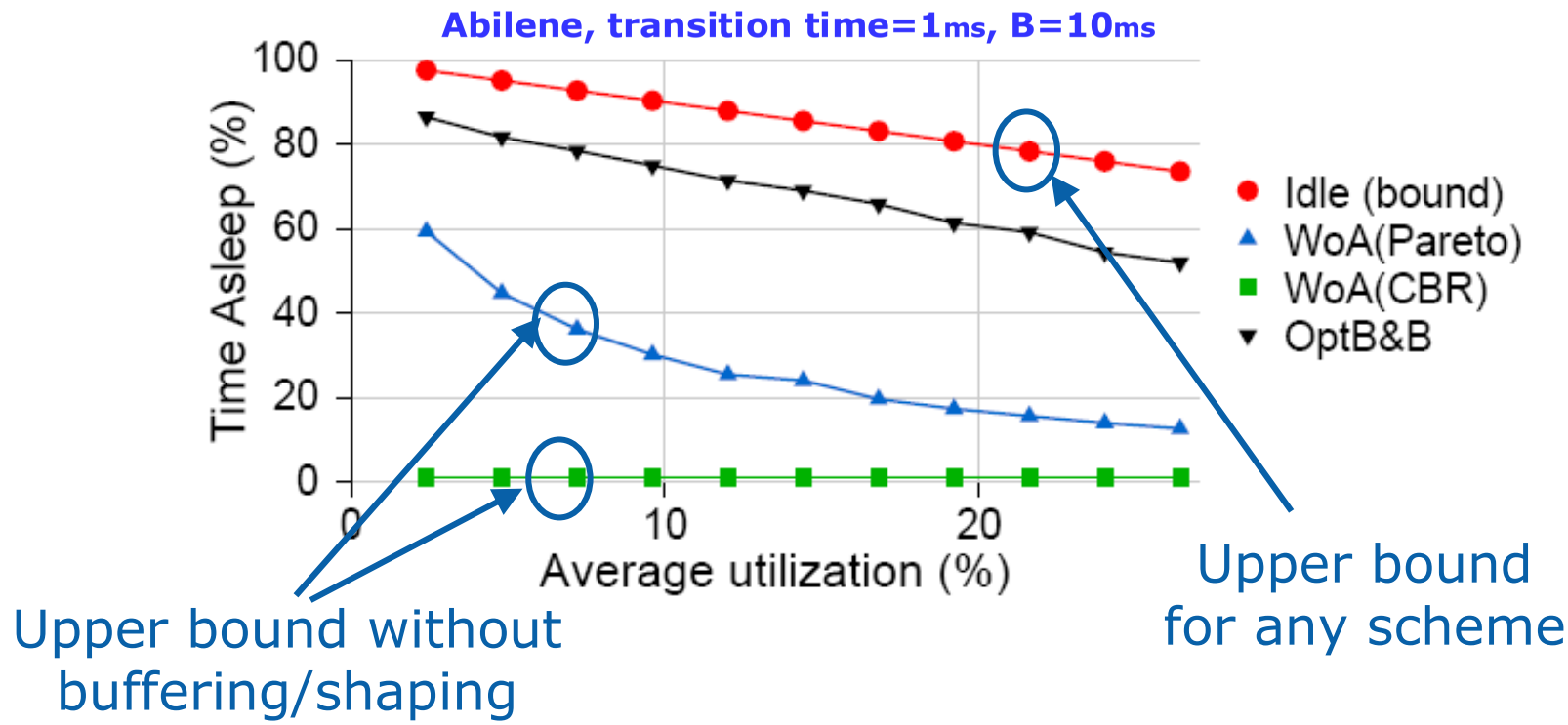
Best schedule can't preserve all gaps

"perfect" coordination not generally possible



Find best schedule via global search as an upper bound (optB&B) on sleeping benefits; don't add more buffering

Potential benefits of sleeping

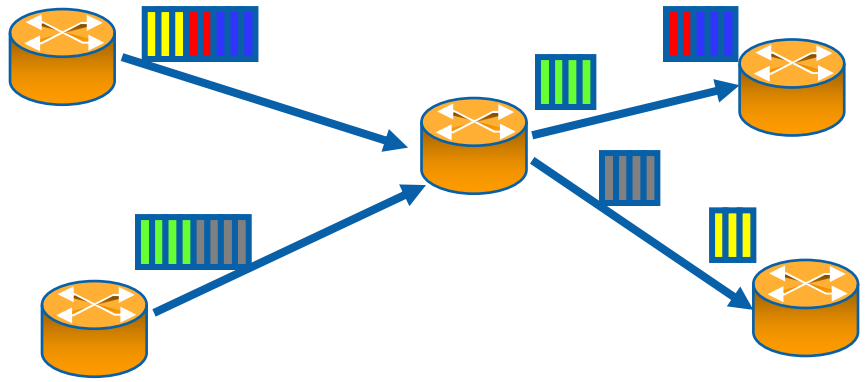


A little shaping can get most of the utilization gain

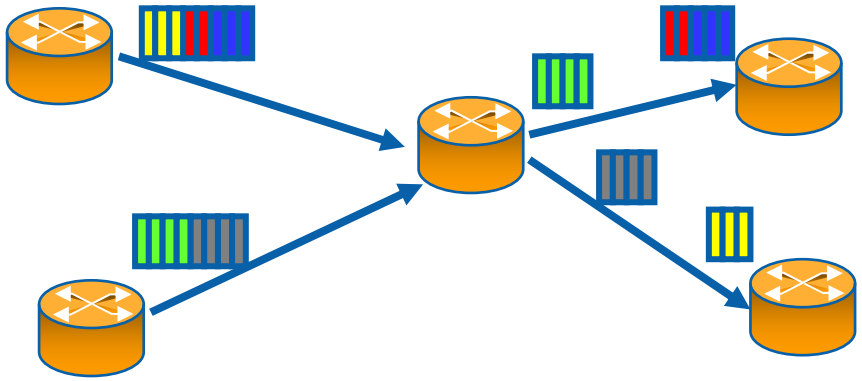
Approximating the best sleep schedule

Basic Idea: Ingresses independently organize their bunches by egress. Then, each router sees at most #egress bursts, typically much less.

heuristic is to bunch per egress at each ingress

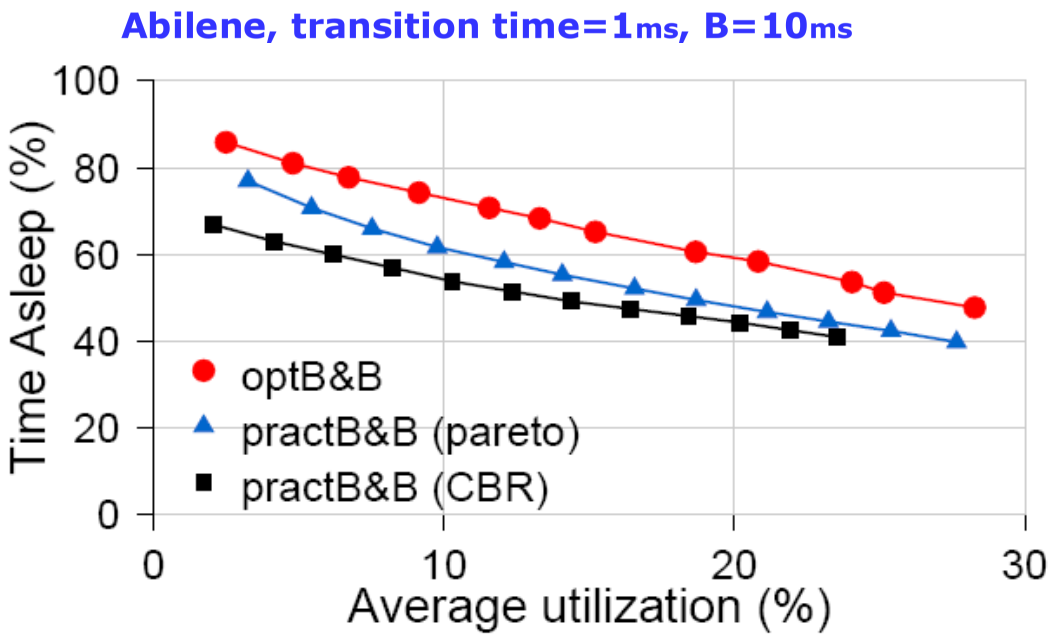


Practical sleeping algorithm (practB&B)



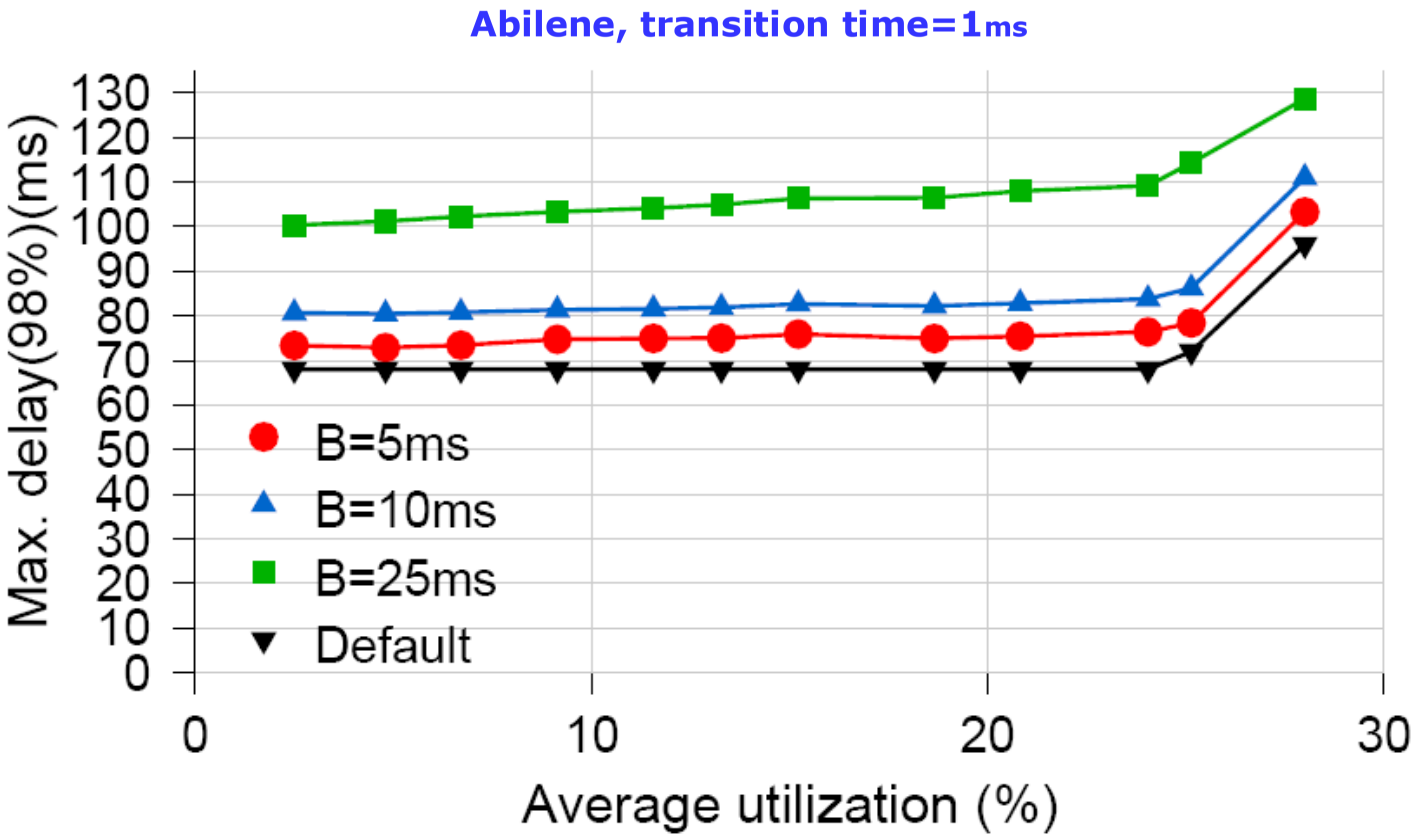
1. Ingress buffers and transmits packets in a bunch every Bms
2. Within bunch, packets are organized by egress
3. Router interfaces wake at known times to process bursts
4. Router interfaces sleep if start of next burst is >2δ ms away

Practical benefits of sleeping



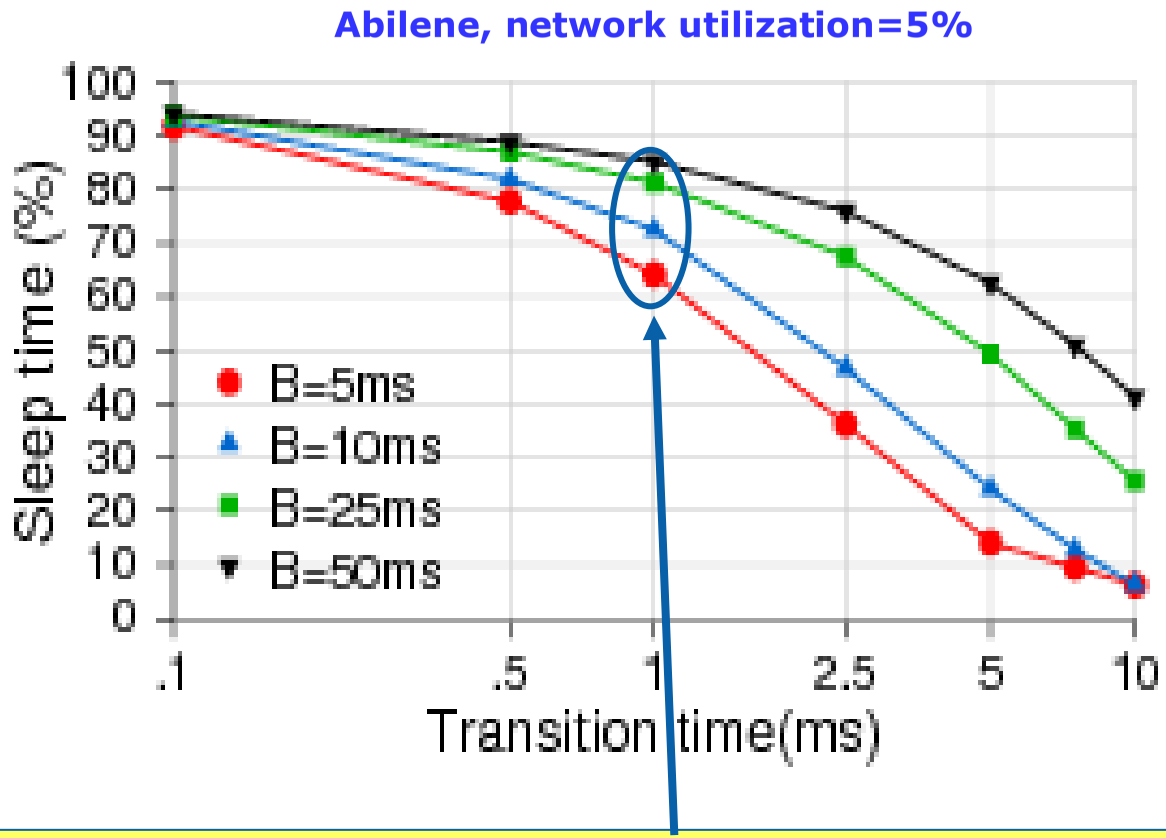
Practical algorithm realizes most of the benefit

Impact of sleeping on delay



No added loss; added delay ~ bounded by Bms

Impact of sleep transition time

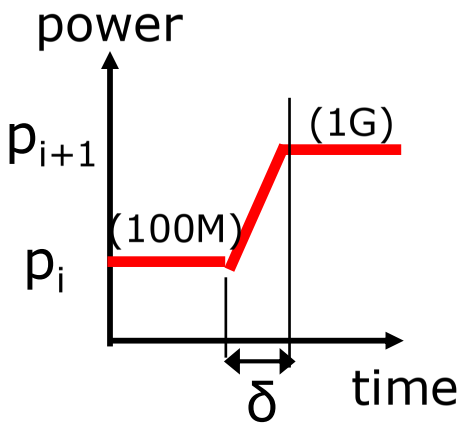


Quick transitions (preferably < 1ms) needed

3. Rate adaptation states

Model

- N performance states
- Rates r_1, …, r_n and $p_i < p_{i+1}$
- δ : transition period (ms)
- Interfaces can rate-adapt independently



Metrics

- Energy savings in reduction of average rate (approx. savings if power linear in rate and $p_{idle} \sim p_{active}$)
- Performance in loss and max delay

Practical rate adaptation (practRA)

Idea: Only lower rate if doing so will maintain minimal queuing delay (of at most d ms); aggressively increase rate to clear buildup

Algorithm:

r_f : estimated arrival rate as average (EWMA) of past arrivals

q: current queue size

d: target maximum queuing delay

r_i : current service/link rate

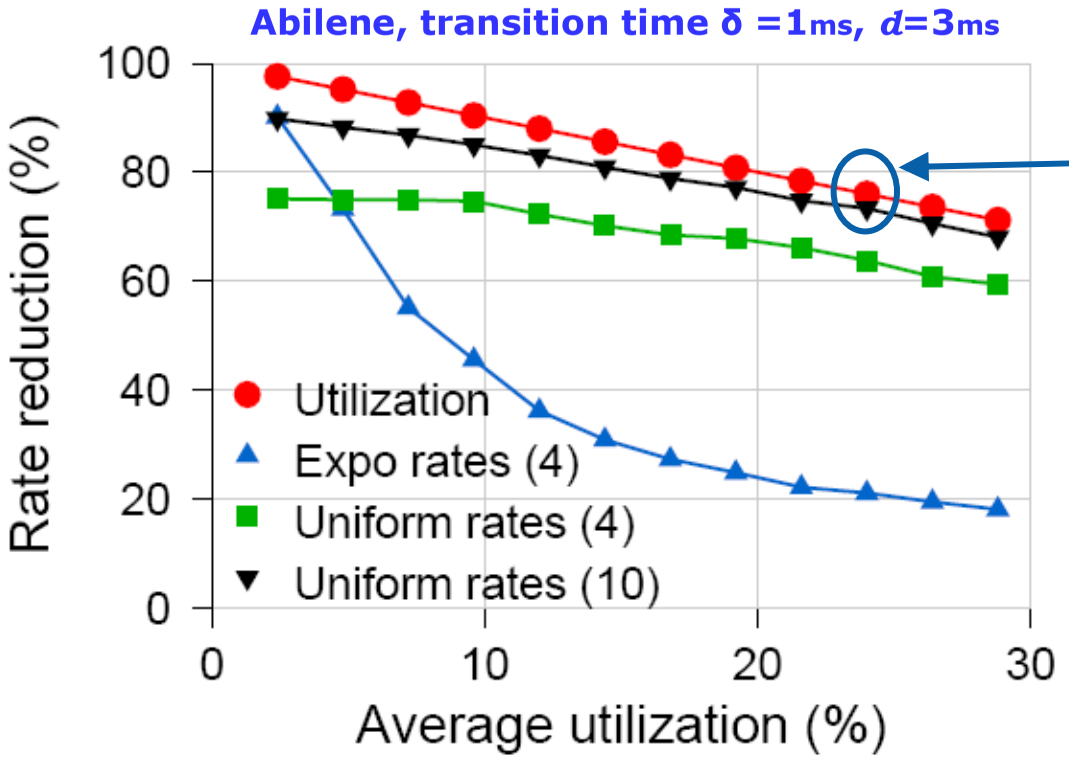
Rules:

1. increase to r_{i+1} iff $(q/r_i > d)$ **OR** $(\delta r_f + q)/r_{i+1} > (d - \delta)$
2. decrease to r_{i-1} iff $(q = 0)$ **AND** $(r_f < r_{i-1})$
3. time of last rate change > $k\,\delta$ $(k=4)$

Leave headroom for transition time

Avoid frequent changes

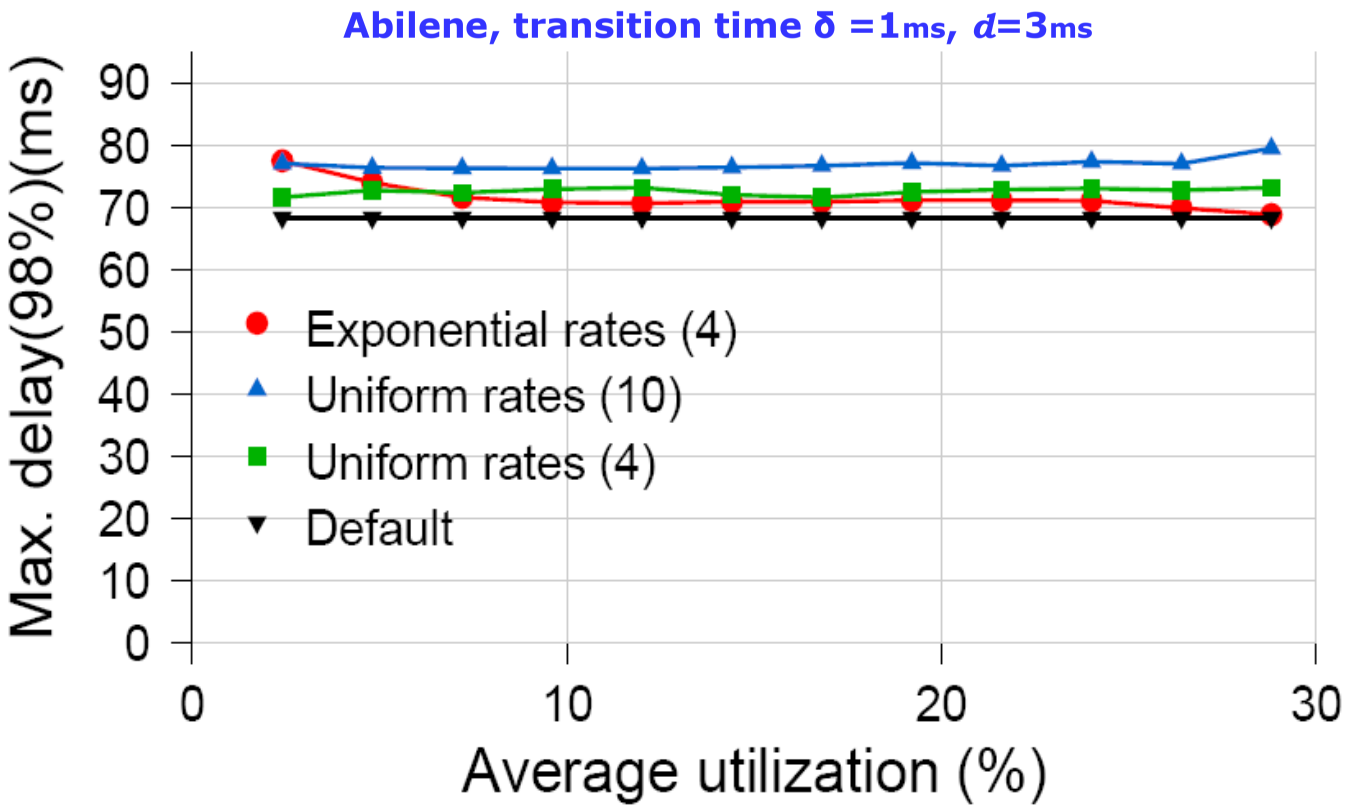
Benefits of rate adaptation



Upper bound for any scheme

Practical rate adaptation close with uniform rates

Impact of rate adaptation on delay



No added loss; added delay < d x $\#hops$

4. Comparing sleeping and rate adaptation

Need to convert metrics of sleep time and average rate reduction to overall energy savings.

1. Based on measured power profiles for real equipment
   - Intel NIC [hays_1107]
   - Cisco GSR router [Chabarek, Infocom08]
2. Models of future power profiles
   - Range of constants for C (static to dynamic power draw), beta (idle to active), and gamma (sleep to idle)
   - Frequency and DVS scaling for rate adaptation (power ~ f and f^3)

Measured equipment power profiles

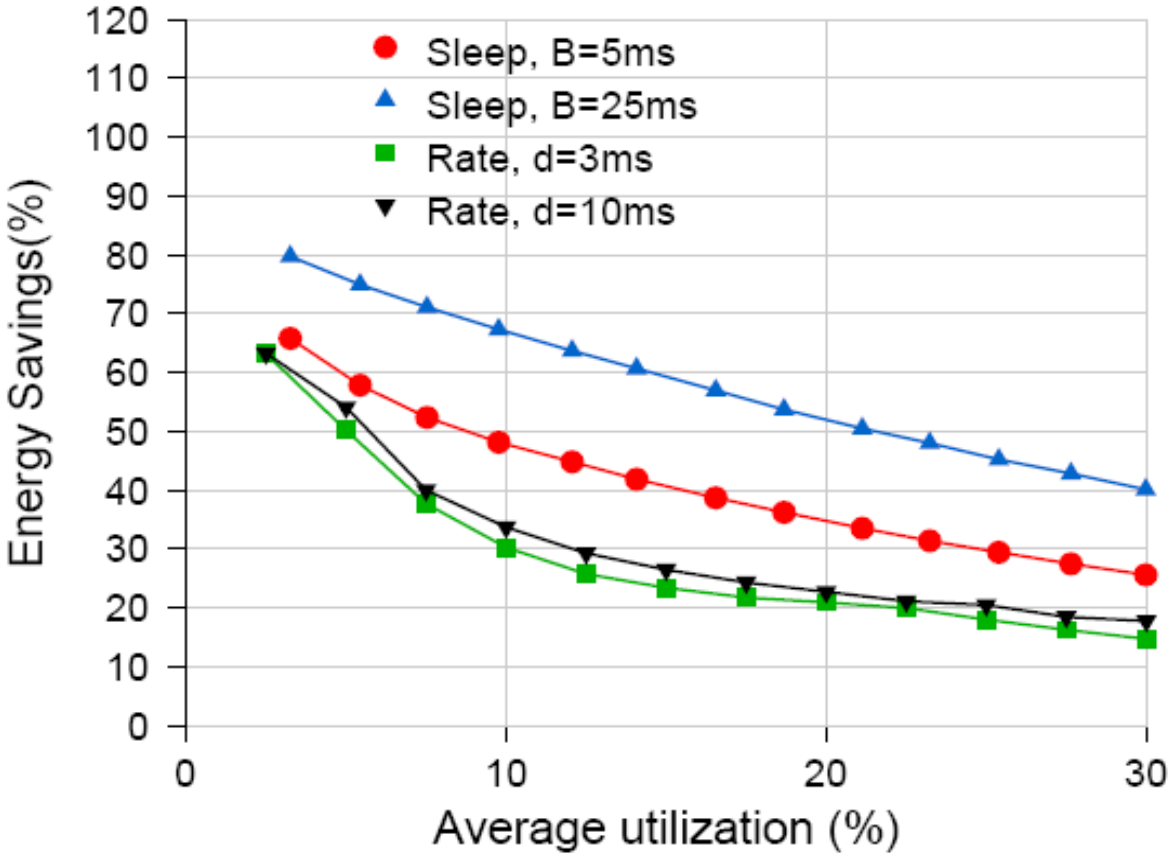
	Intel NIC (R=1Gbps)	Cisco GSR (R=10Gbps)
$p_{active}(R)$	1217 mW	200+ 80W/linecard
$p_{idle}(R)$	1010 mW	200+70W/LC
$p_{active}(R/10)$	483 mW	200+32W/LC
$p_{idle}(R/10)$	314 mW	200+22W/LC
$p_{active}(R/100)$	504 mW	200+33W/LC
$p_{idle}(R/100)$	194 mW	200+13W/LC
p_{sleep}	65 mW	200W

extrapolated

chassis only

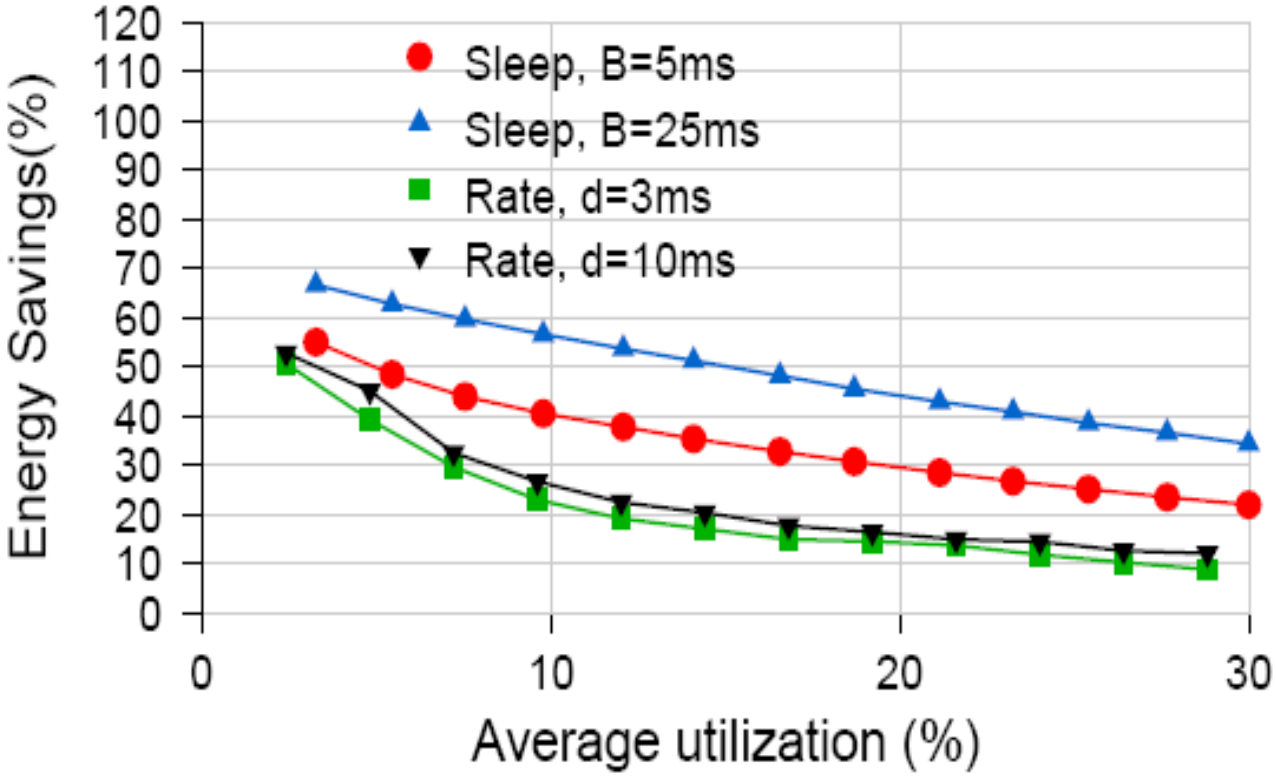
Sleeping and Rate-adaptation (Intel NIC)

Abilene backbone; transition time=1ms; rates=1G/100M/10Mbps



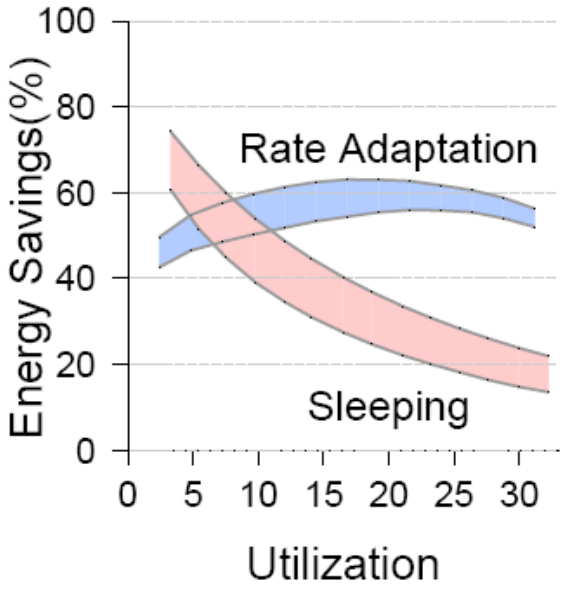
Sleeping and Rate-adaptation (Cisco GSR)

Abilene backbone; transition time=1ms; rates: 10G/1G/100Mbps



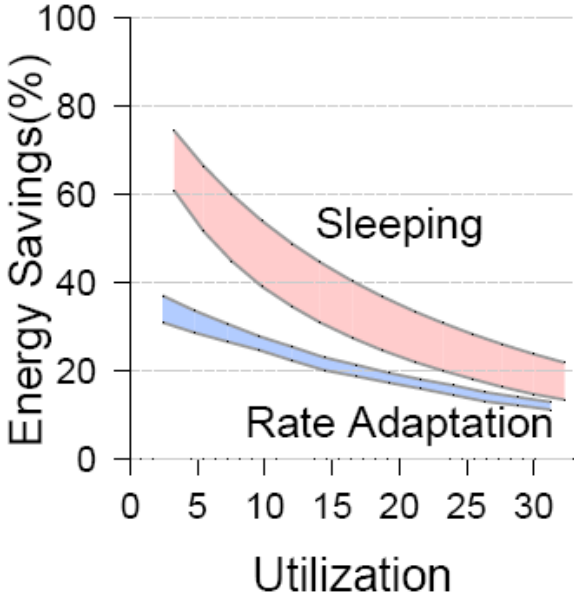
Sleeping and rate adaptation (models)

With DVS, 10 rates, δ=0.1 to 1ms



(a) $C = 0.1, \beta = 0.1$

Without DVS



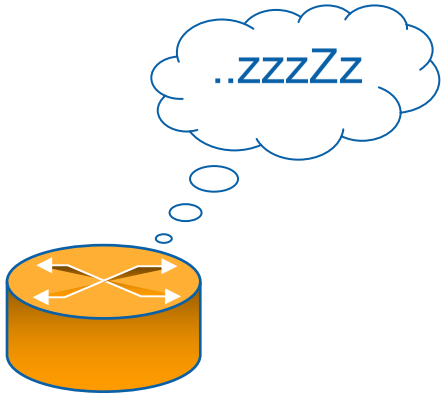
(a) $C = 0.1, \beta = 0.1$

Sleeping is simple and offers good savings for low load

Conclusion

Much potential value in sleep/performance states for network equipment:

- Stand to halve energy consumption for lightly loaded links (10-20%)
- Can do so with little impact on performance (small added delay)
- Can realize these gains with simple/practical schemes



Thank you. Questions?

David Wetherall
david.wetherall@intel.com
djw@cs.washington.edu

"Reducing Network Energy Consumption via Rate-adaptation and Sleeping," S. Nedevschi, et. al., NSDI 2008 (to appear)