Reducing network energy consumption via sleeping and rate-adaptation

Sylvia Ratnasamy
Intel Research

joint work with:
Sergiu Nedevschi, Lucian Popa (UC Berkeley),
Gianluca Iannaccone (Intel Research),
David Wetherall (Intel Research and U. Washington)

Motivation

- network energy consumption, a growing issue
  - network equipment increasingly power-hungry
  - rising energy costs
  - environmental concerns, …
- opportunity for conservation appears significant
  - longer-term network utilization is low (Sprint ~15%, Intel ~2%)
  - but networks are provisioned for peak load
  - and idle-time consumption remains high

Overview

Goal: save energy <u>without compromising performance</u>

- achieving this will depend on:
  - appropriate hardware-level support for power management
  - higher-layer algorithms that invoke this support wisely
- our study
  - model hardware-level support
  - design, evaluate higher-layer algorithms
  - explore how hardware support impacts savings/performance

Outline

- sleep and rate-adaptation in networks: rationale
- saving energy via sleeping
- saving energy via rate-adaptation
- sleeping *vs*. rate-adaptation
- conclusions

Rationale

- total energy consumption $E \sim p_{idle} T_{idle} + p_{active} T_{active}$
- power management in computers suggests two approaches to reduce E
  - sleep states: $p_{idle} \rightarrow p_{sleep}$ with $p_{sleep} << p_{idle}$
  - performance states: reduce speed → lower p_{active} , p_{idle} but higher T_{active}
    - ACPI, a common standard provides system/device sleep states for PCs
    - more on PC power management at: http://grouper.ieee.org/groups/802/3/eee_study/public/mar07/chalupsky_01_0307.pdf
- similarly, for networks, we assume
  - sleep states based on quickly powering off network interfaces when idle
  - performance states based on dynamically adapting the rate of a link/interface
  - network <u>as a whole</u> runs in either sleep or rate-scaling mode

Outline

- sleep and rate-adaptation in networks: rationale
- saving energy via sleeping
- saving energy via rate-adaptation
- sleeping *vs*. rate-adaptation
- conclusions

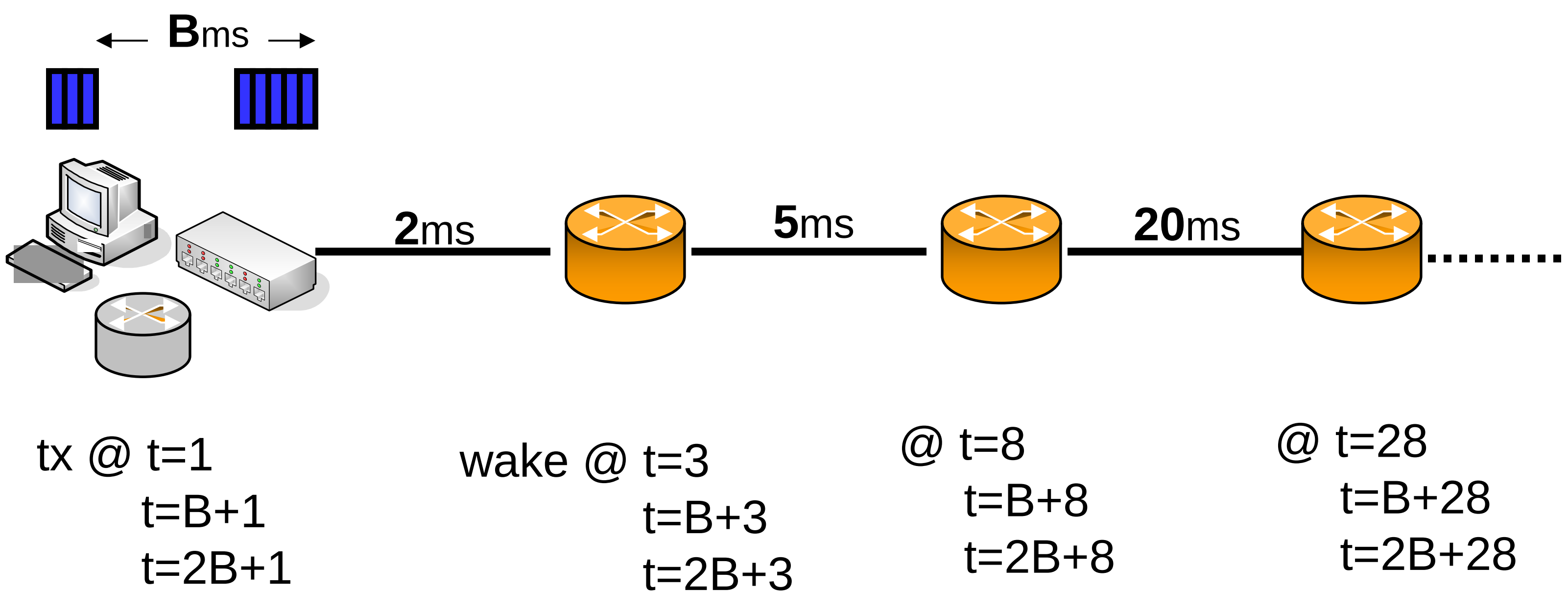
Network sleep states

Model

- sleep state with power draw p_{sleep}
- δ : transition period
- timer-driven sleep
- interfaces can sleep independently

Using sleep states

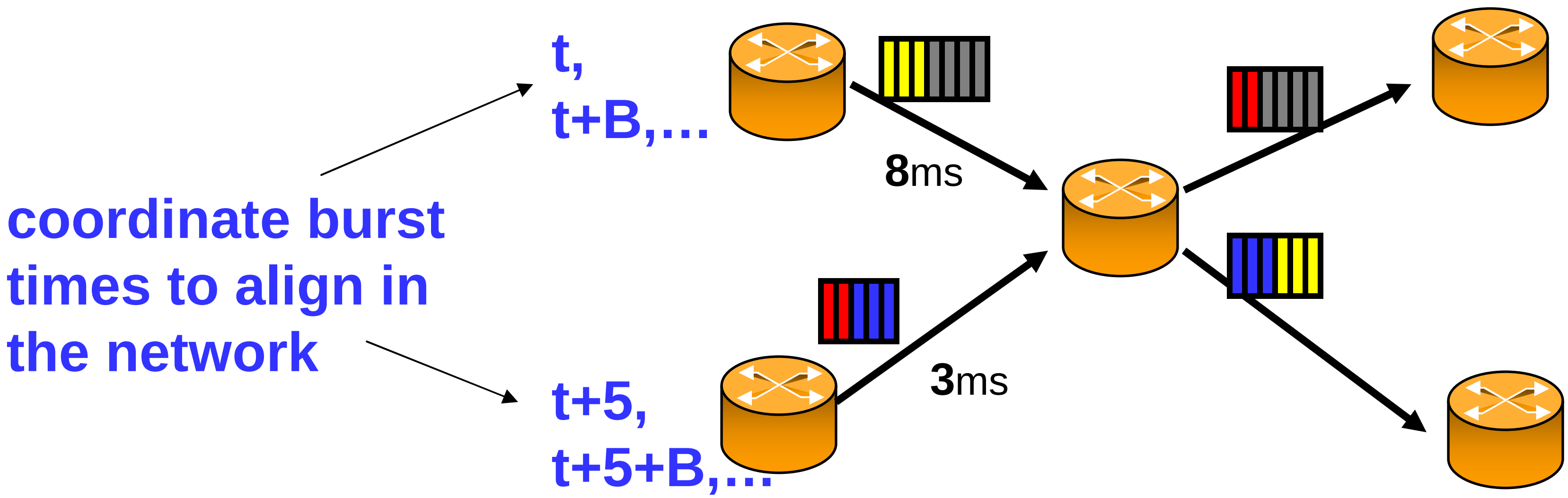
Basic idea: "buffer and burst" traffic shaping



1. source/edge transmits packets in a bunch every Bms
2. intermediate routers wake to process burst; sleep if next burst arrives < δ

Using sleep states

Basic idea: “buffer and burst” traffic shaping

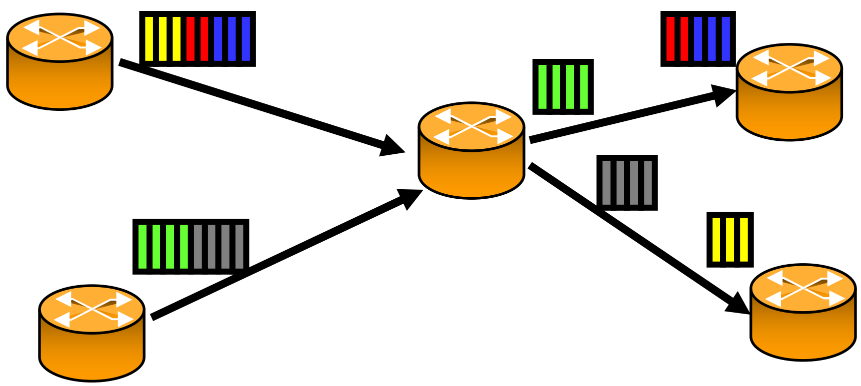


B&B for general network topologies: <u>coordinating ingresses</u>

Using sleep states

Basic idea: "buffer and burst" traffic shaping

or simple approximations might suffice
***e.g.*, bunch at ingress**



B&B for general network topologies: **ingresses burst independently**

Using sleep states

Basic idea: “buffer and burst” traffic shaping

- simple, general
- controlled tradeoff:
  - amortized transitions
  - added <u>end-to-end</u> delay ≤ B*ms*

Evaluated two coordination algorithms

- optimal_B&B (baseline)
  - optimally coordinated ingresses
- practical_B&B
  - ingresses buffer-and-burst independently

Evaluation methodology

- packet-level simulation (ns2)
- using real network topologies and traffic
  - Abilene backbone
  - Intel enterprise network

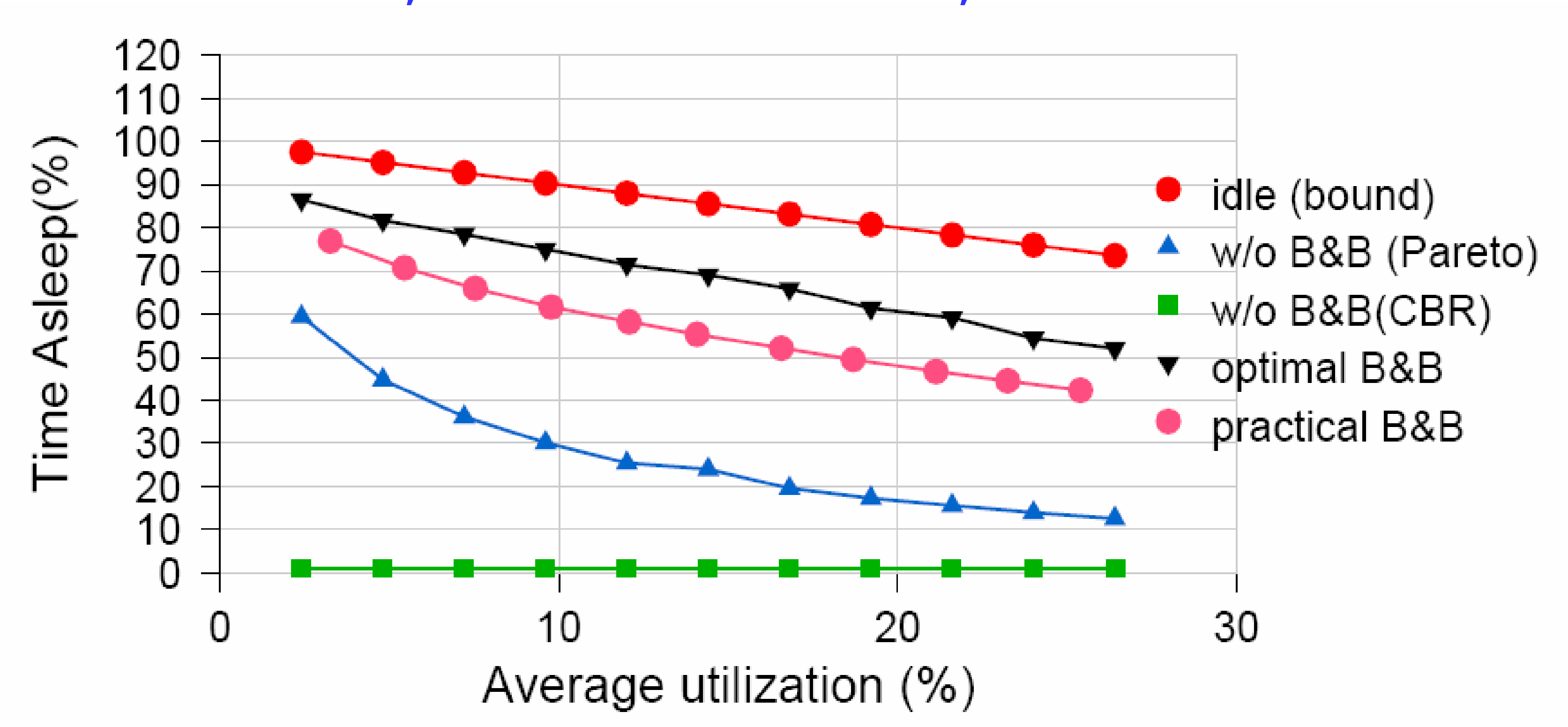
    (scale measured traffic to explore different utilizations)

- metrics
  - savings: % time asleep
    - will later translate these into energy savings
  - performance: 98 percentile delay
    - also looked at loss, avg. delay

- present key results here (more in a techreport)

Savings: % time asleep

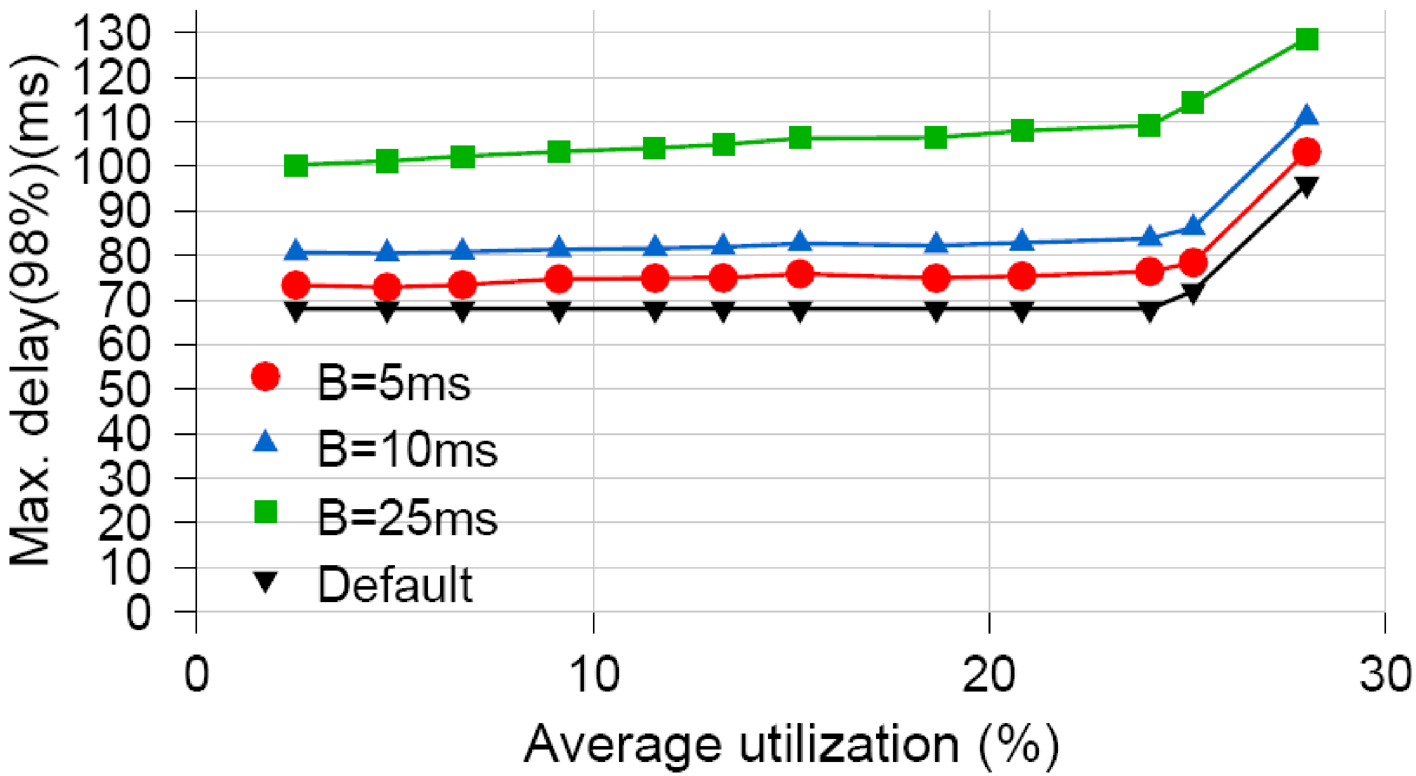
Abilene; transition time=1ms, B=10ms



1. simple traffic shaping is effective (but room for improvement)
2. poor sleeping w/o traffic shaping

Impact of sleeping on delay

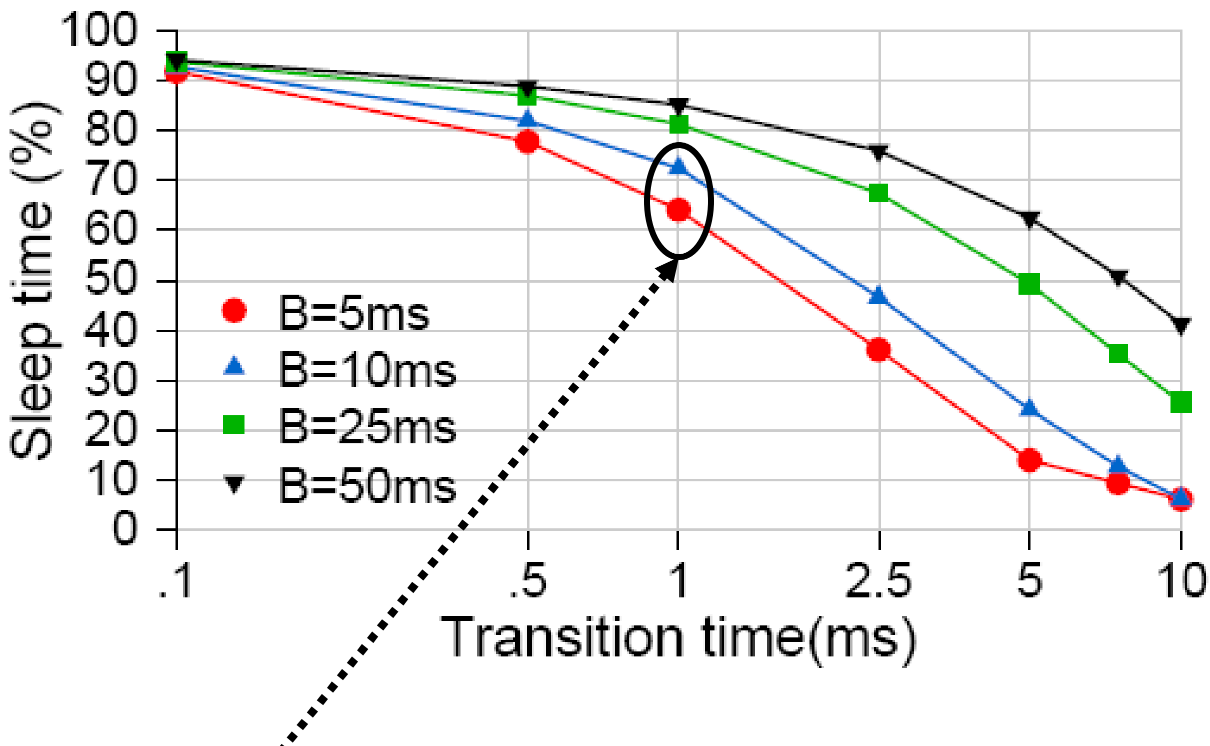
Abilene; transition time=1ms



added delay due to sleeping ~ bounded by Bms

Impact of transition time

Abilene; network utilization=5% (measured)



quick transitions (preferably < 1ms) needed

Outline

- sleep and rate-adaptation in networks: rationale
- saving energy via sleeping
- saving energy via rate-adaptation
- sleeping *vs*. rate-adaptation
- conclusions

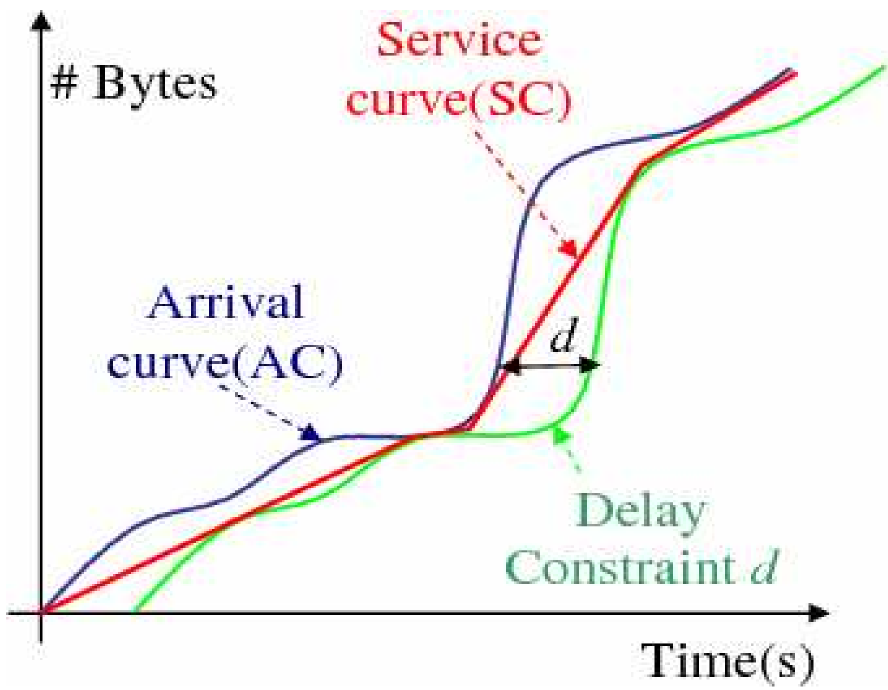
Network performance states

Model

- N performance states with rates $r_1, \ldots, r_n$
- δ : transition period
- interfaces can rate-adapt independently

Using performance states

Goal: increase/decrease rate iff doing so doesn't increase queuing delay by more than d ms



Using performance states

Goal: increase/decrease rate if doing so doesn't increase queuing delay by more than d ms

Algorithm:

r_f : estimated arrival rate as EWMA of past arrivals

q: current queue size

r_i : current service/link rate

rules:

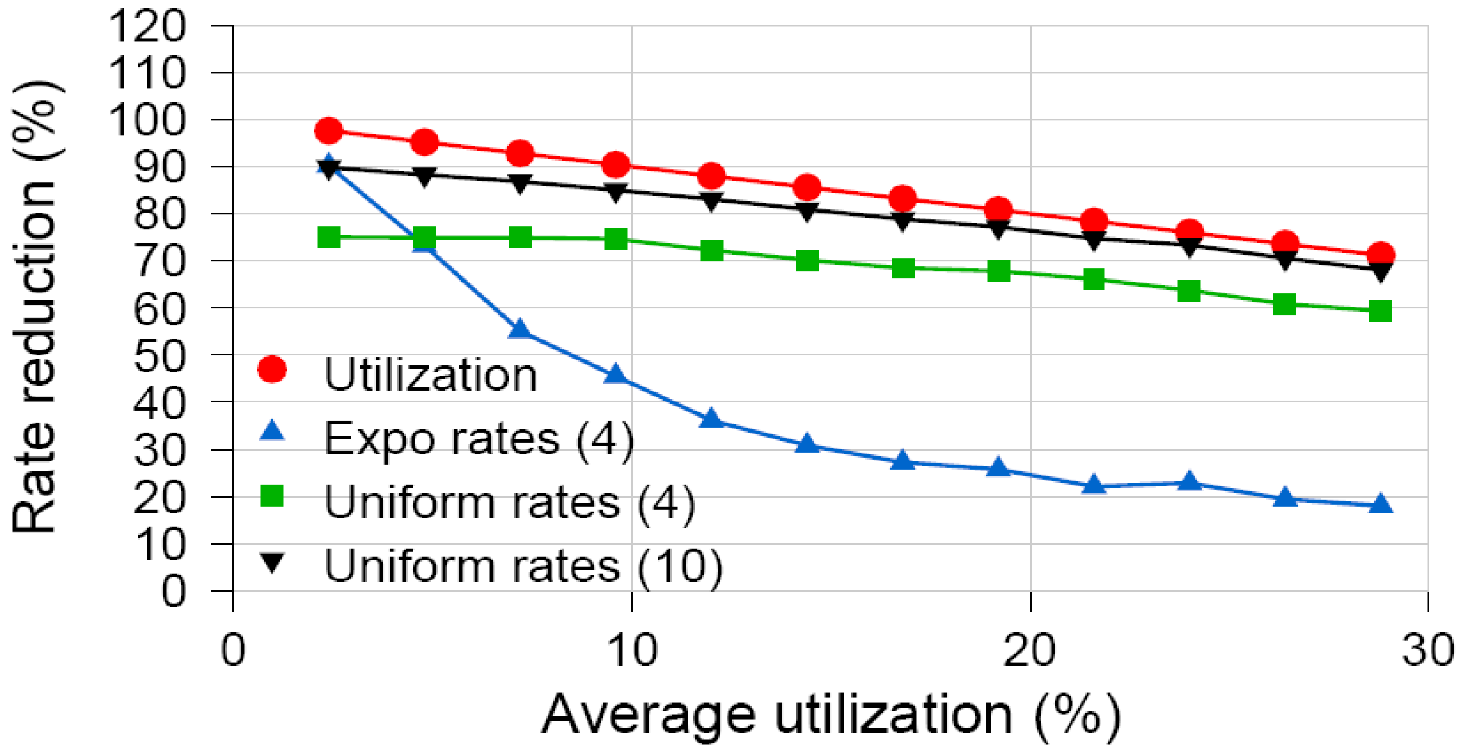
1. increase to r_{i+1} if $(q/r_i > d)$ **OR** $(\delta r_f + q)/r_{i+1} > (d - \delta)$
2. decrease to r_{i-1} if $(q = 0)$ **AND** $(r_f < r_{i-1})$
3. time of last rate change > k δ (k=4)

Evaluation methodology

- simulation environment: as before
- metrics
  - savings: % reduction in rate
    - will later translate these into energy savings
  - performance: 98 percentile delay

Savings: % reduction in rate

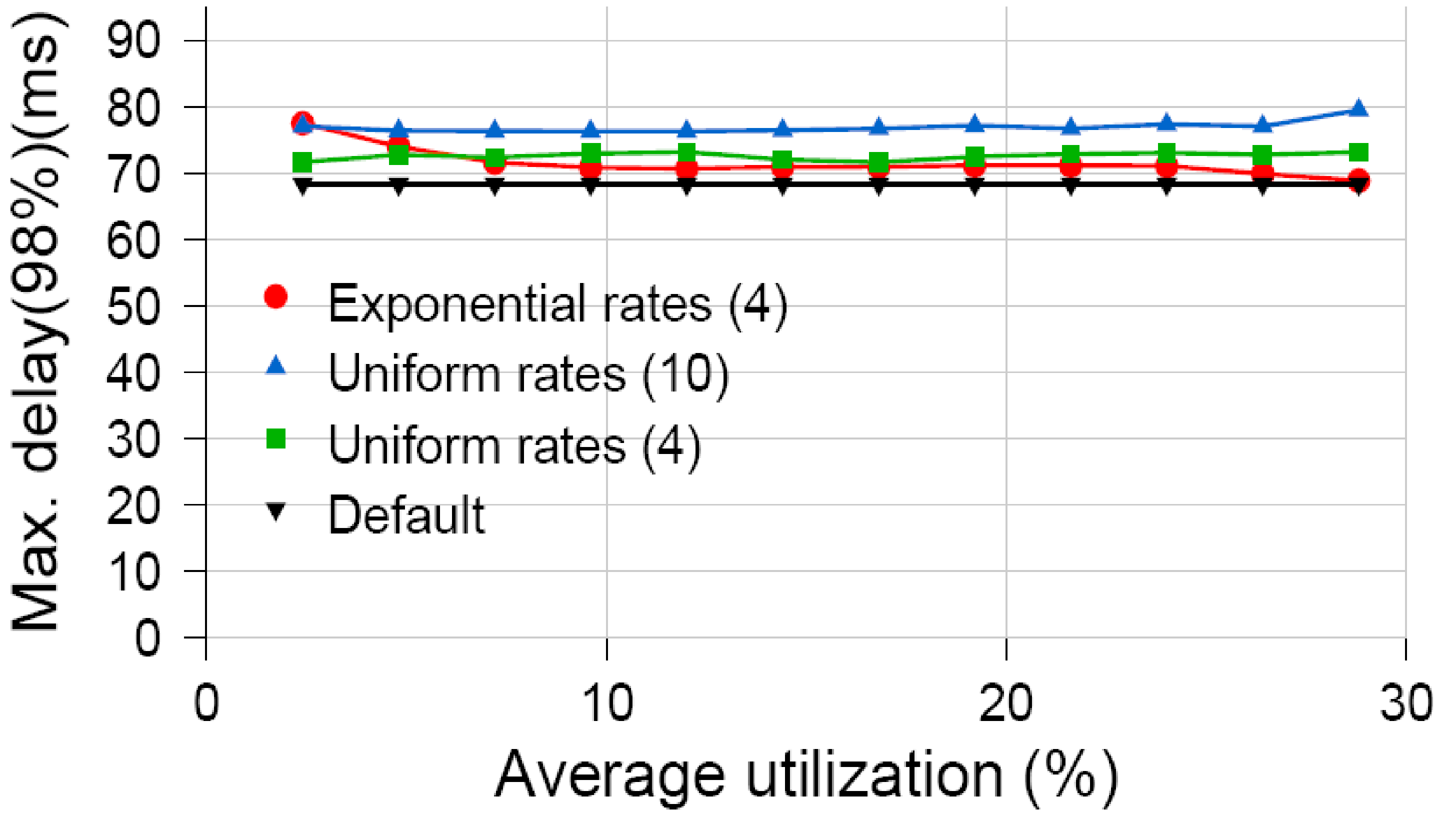
Abilene, transition time δ =1ms, *d*=3ms



1. rate scaling algorithm very effective
2. uniformly distributed rates better for higher utilizations

Impact on delay

Abilene, transition time δ =1ms, d=3ms



added delay < d x #$hops$

Outline

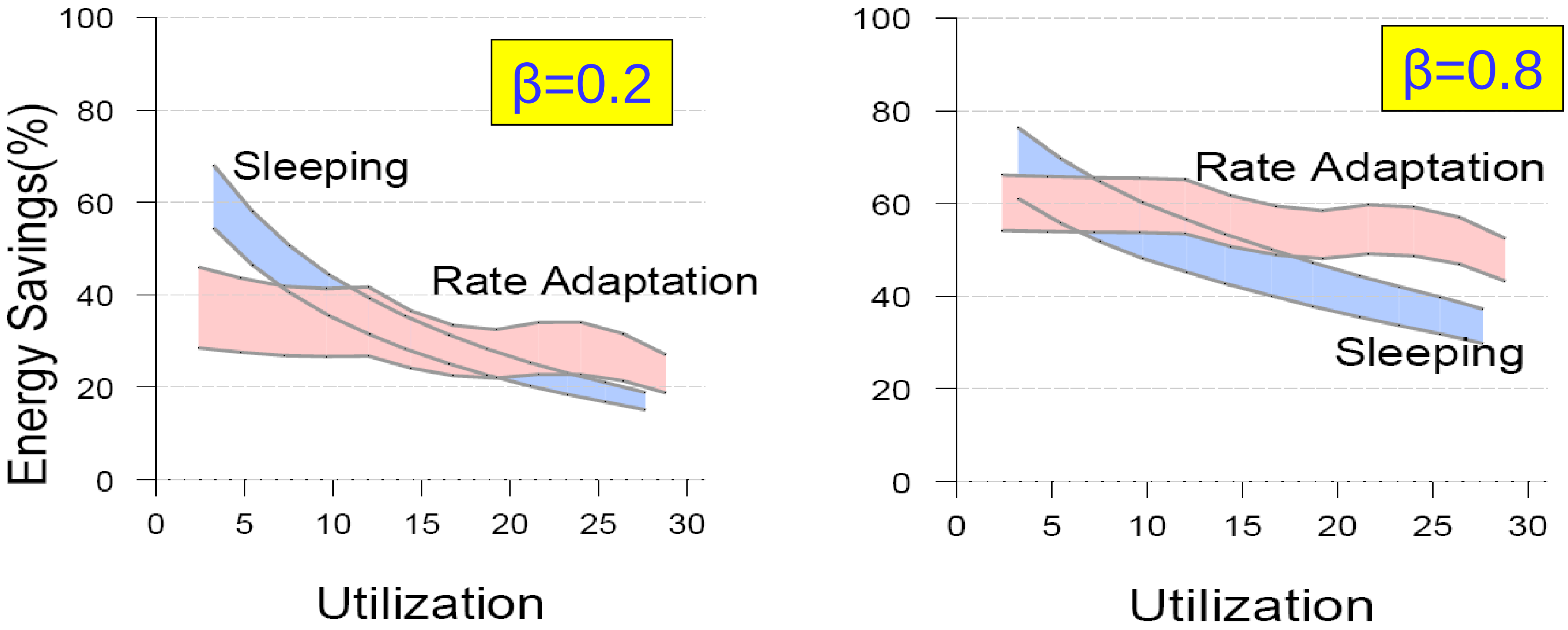
- sleep and rate-adaptation in networks: rationale
- saving energy via sleeping
- saving energy via rate-adaptation
- sleeping *vs*. rate-adaptation
- conclusions

Comparing energy savings

- total energy consumption ~ $p_{idle}\, T_{idle} + p_{active}\, T_{active}$
- energy savings depends on relative magnitudes of p_{active}, p_{idle}, p_{sleep}
- model
  - $p_{active} = c + fn(rate)$ [c : 10-30% of fn(max_rate)]
  - $p_{idle} = c + \beta\, fn(rate)$ [we consider $0.2 < \beta < 0.8$]
  - $p_{sleep} = \mu\, p_{idle}$ [we consider $0.0 < \mu < 0.2$]
- consider fn($rate$): linear and cubic (techreport)

Sleep *vs.* Rate-adaptation

varying relative magnitude p_{idle}, p_{active} (β) , fn(*r*): linear



1. sleeping better than rate-adaptation for lower utilizations
2. but "boundary" utilization depends greatly on power profile

Conclusions

<u>High level</u>

- simple schemes can offer significant savings w/ controlled impact on performance
- tradeoff depends greatly on power profile and network utilization
  - (would welcome data on equipment power consumption…)

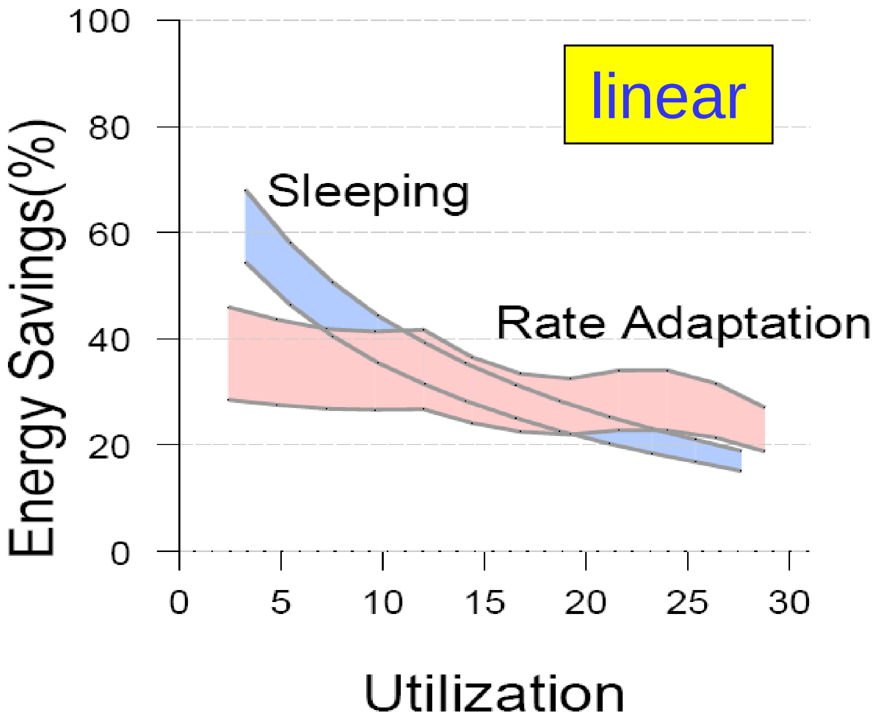
<u>Key observations</u>

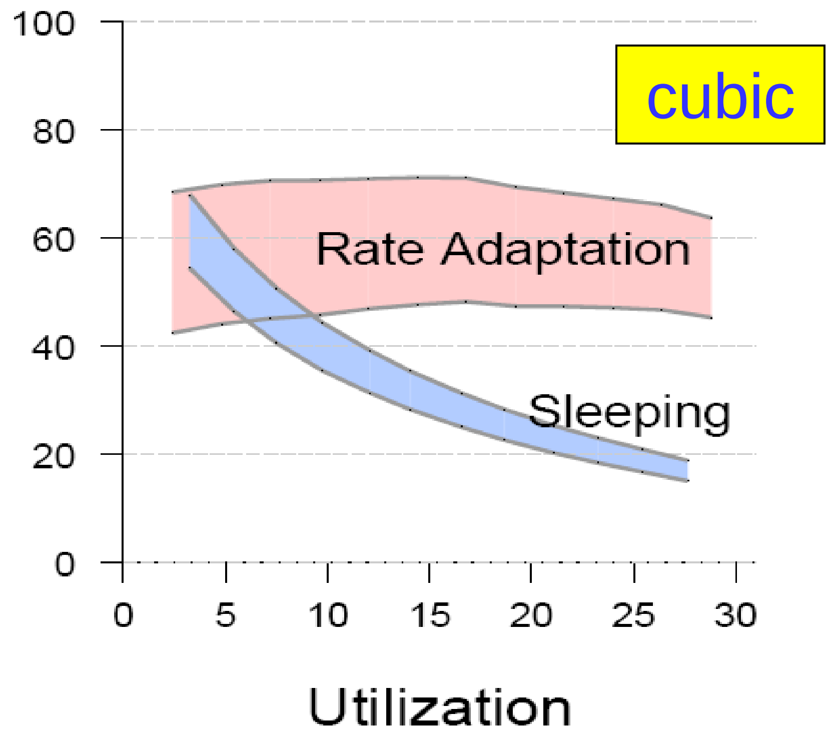
- sleeping is useful; better than rate-adaptation at (typical) low utilizations
- quick transitions (< 1ms) are critical to maintain performance
- distribution of rates beyond 10/100/1000 appears valuable

backup

Sleep *vs.* Rate-adaptation

varying fn(r): linear *vs.* cubic ; β=0.2





Based on measured power from Intel 82573L Gig Ethernet controller (ack: Robert Hays)