ECE 568: Embedded Systems

Lecture #7: Power Aware Design
- Energy Reduction and Management

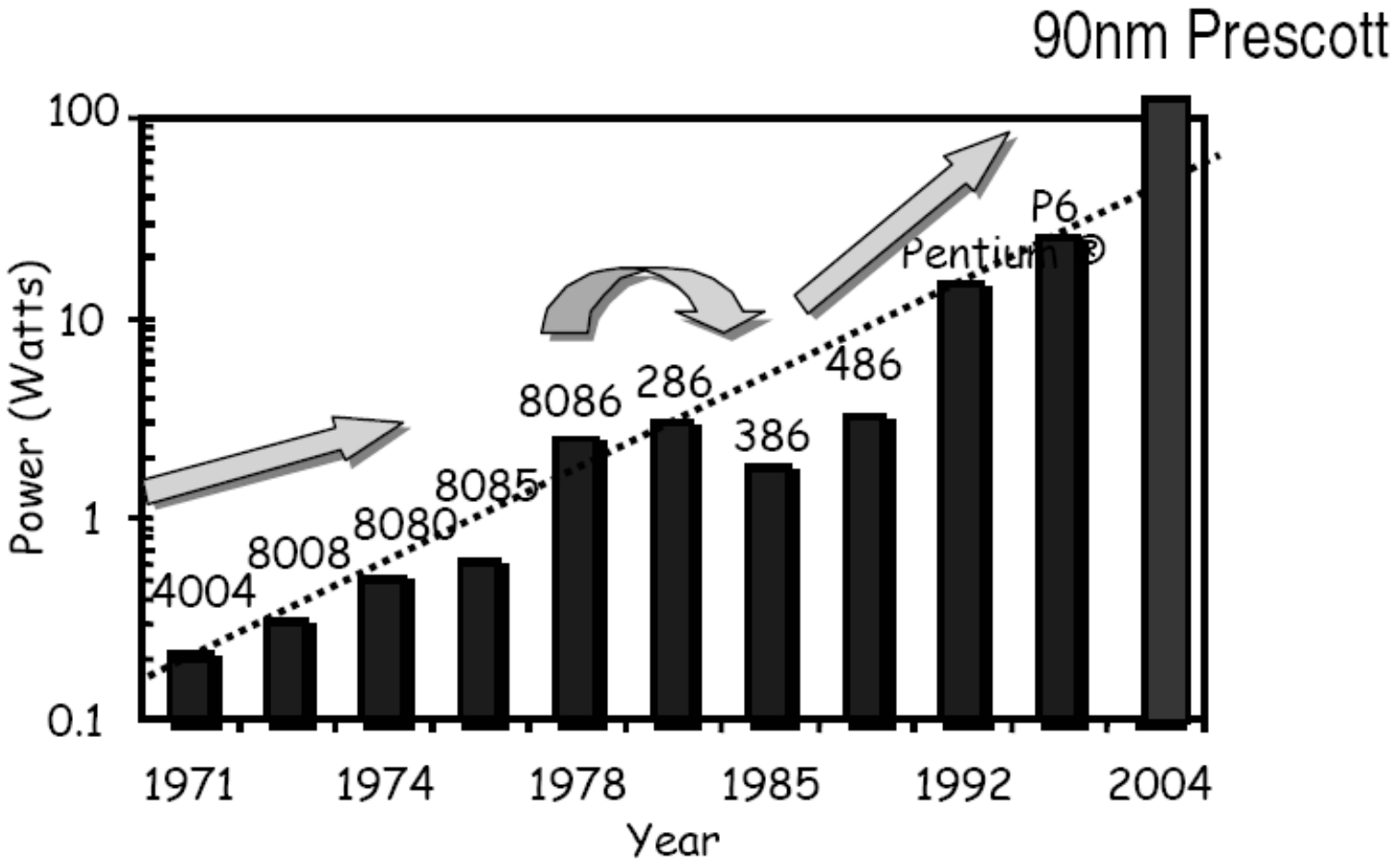
Vijay Raghunathan
vr@purdue.edu

Slides acknowledgement: Mani Srivastava, UCLA

Why bother about power?

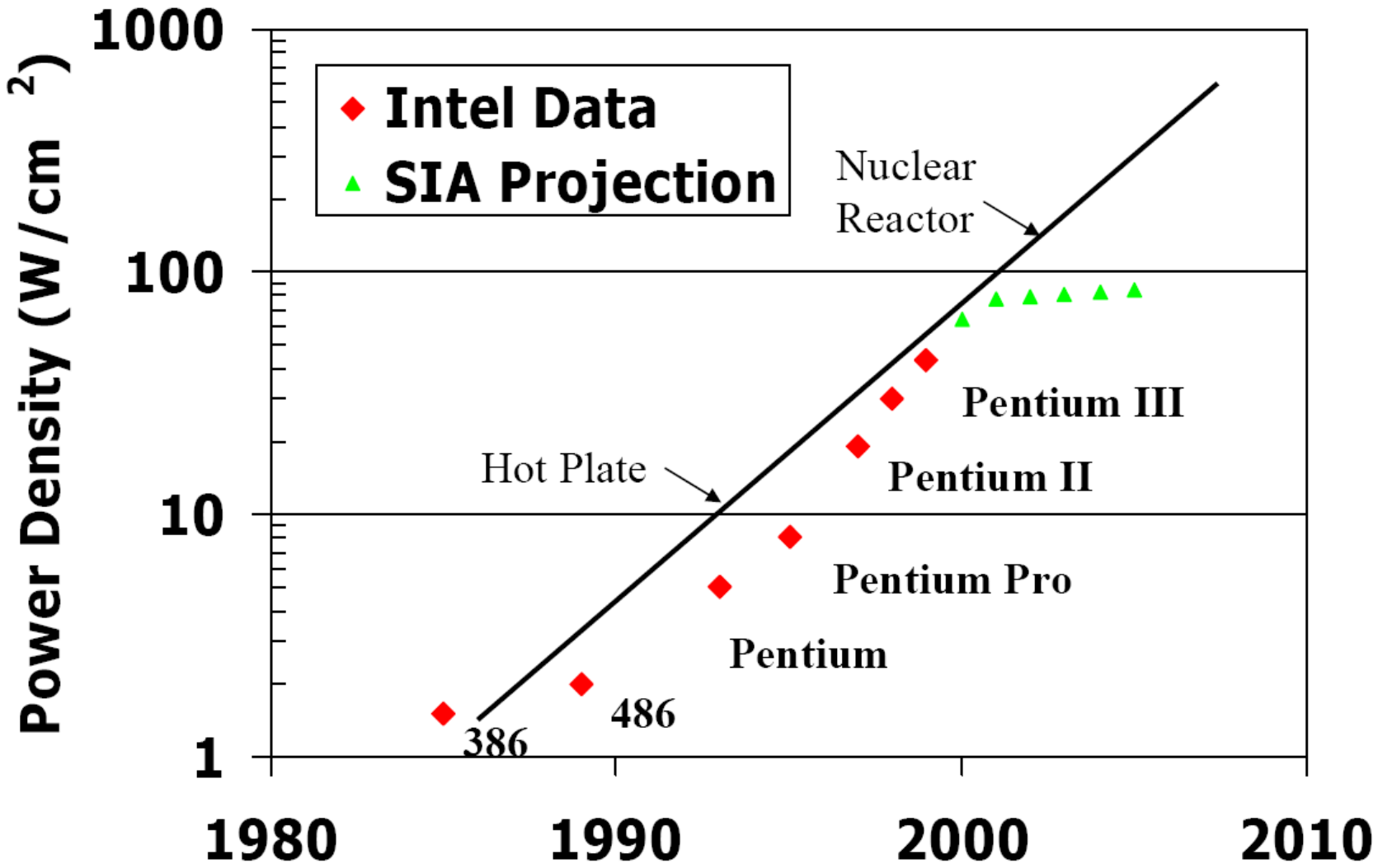
- Chip and system cooling costs
  - Chip packaging constraints
  - Machine room designs
- Power Delivery Costs
- Battery life (in portable systems)
- System reliability
- Environmental concerns

Processor power consumption trends

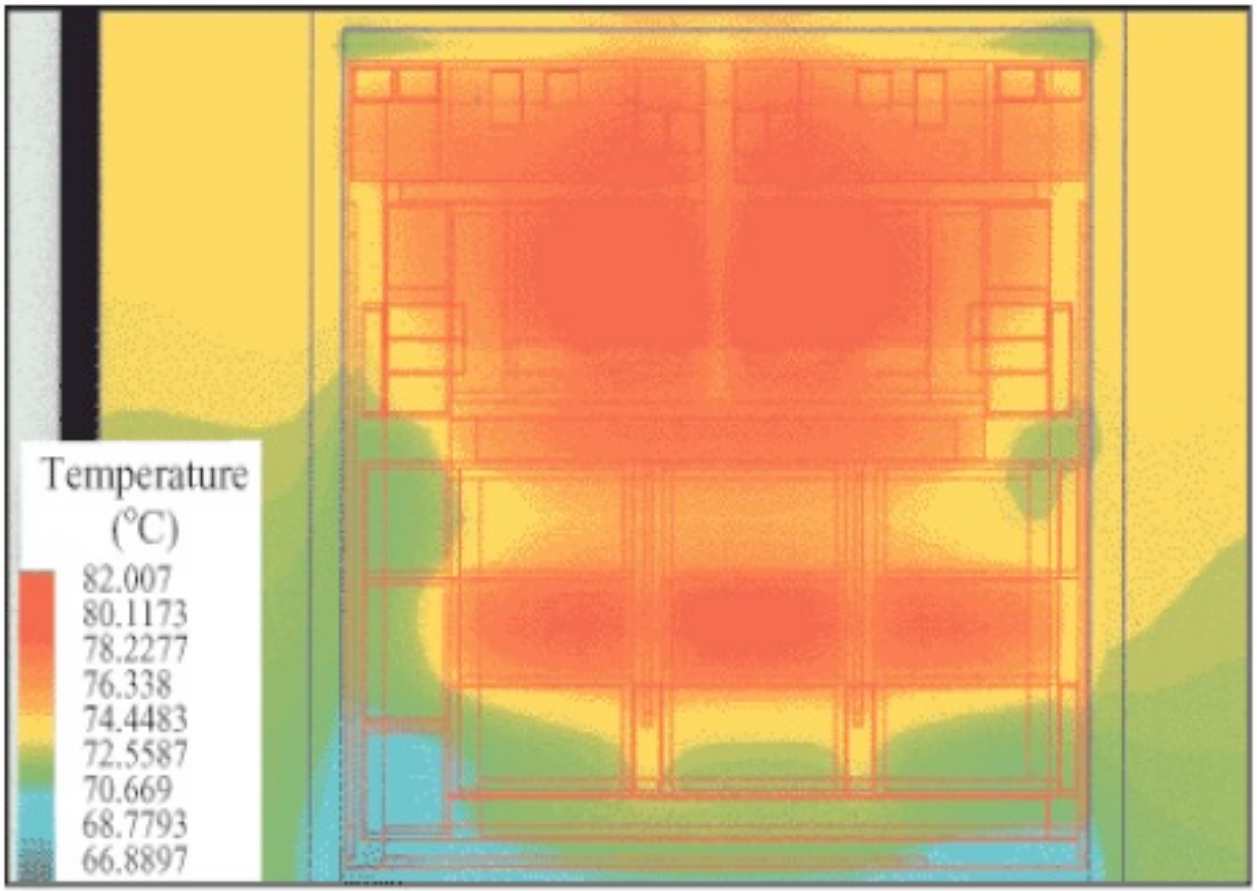


Source: V. De, Intel

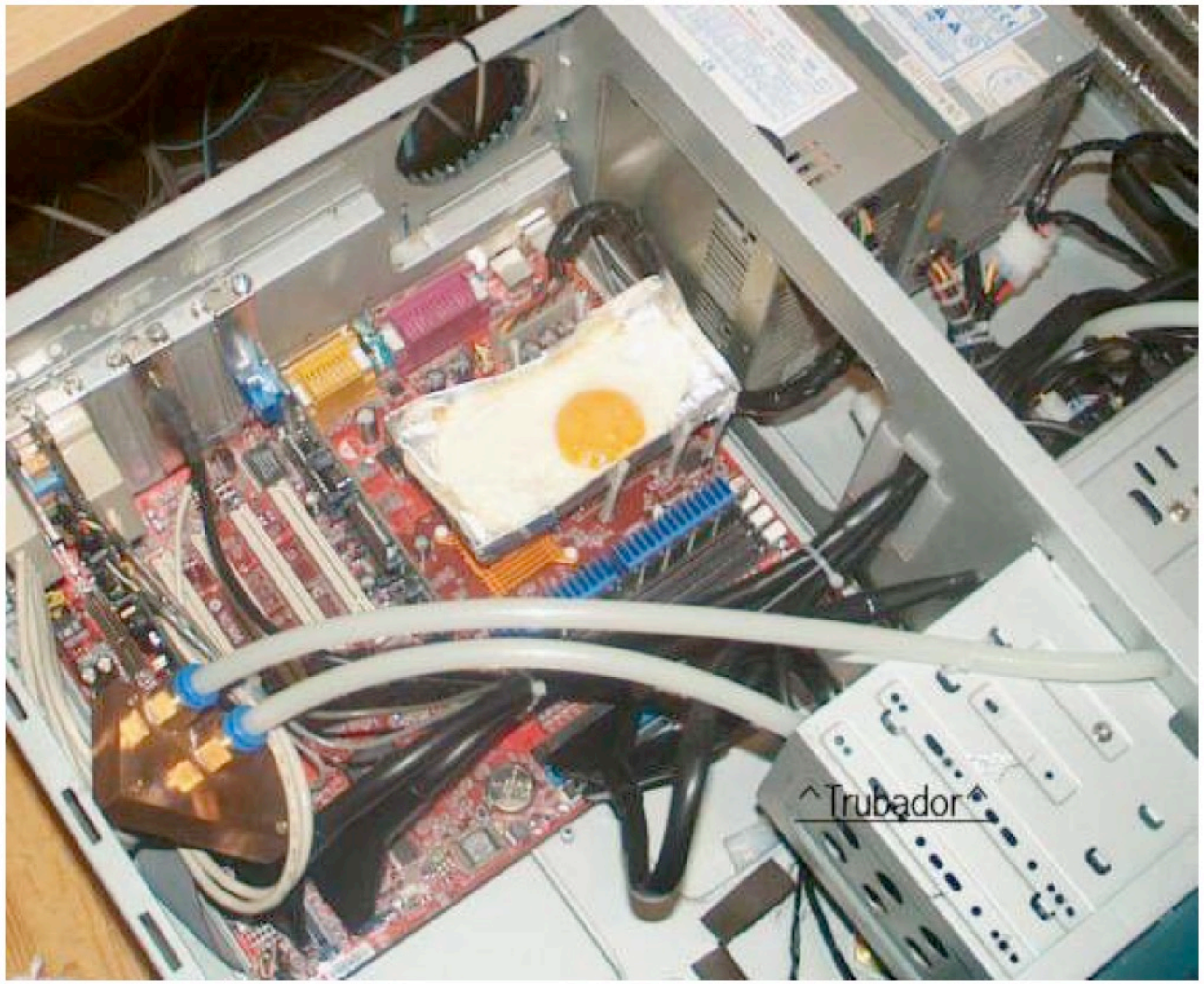
Power Density Trends



Leads to Temperature Problems!



Cooking using Computers



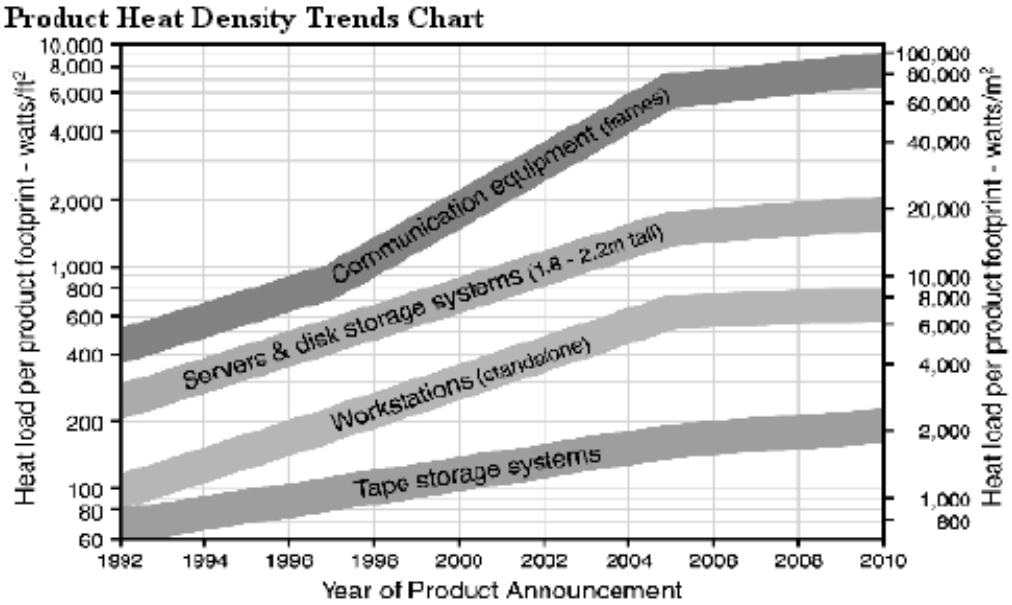
The power problem in large data centers

"Most data centers today cannot support the power and cooling requirements of a large number of new systems." Roger Schmidt, IBM

"What matters most to the computer designers at Google is not speed but power – low-power – because data centers can consume as much electricity as a city", Eric Schmidt, CEO, Google

- Most data centers have a capacity of 40 to 70 W/ft^2
  - They will need to support 500 W/ft^2 in the next few years
- Large servers
  - IBM z900 (single rack) – 9.2 kW
  - IBM p690 (single rack) – 12.5 kW
  - IBM p655 – 28 kW (1915 W/ft^2)

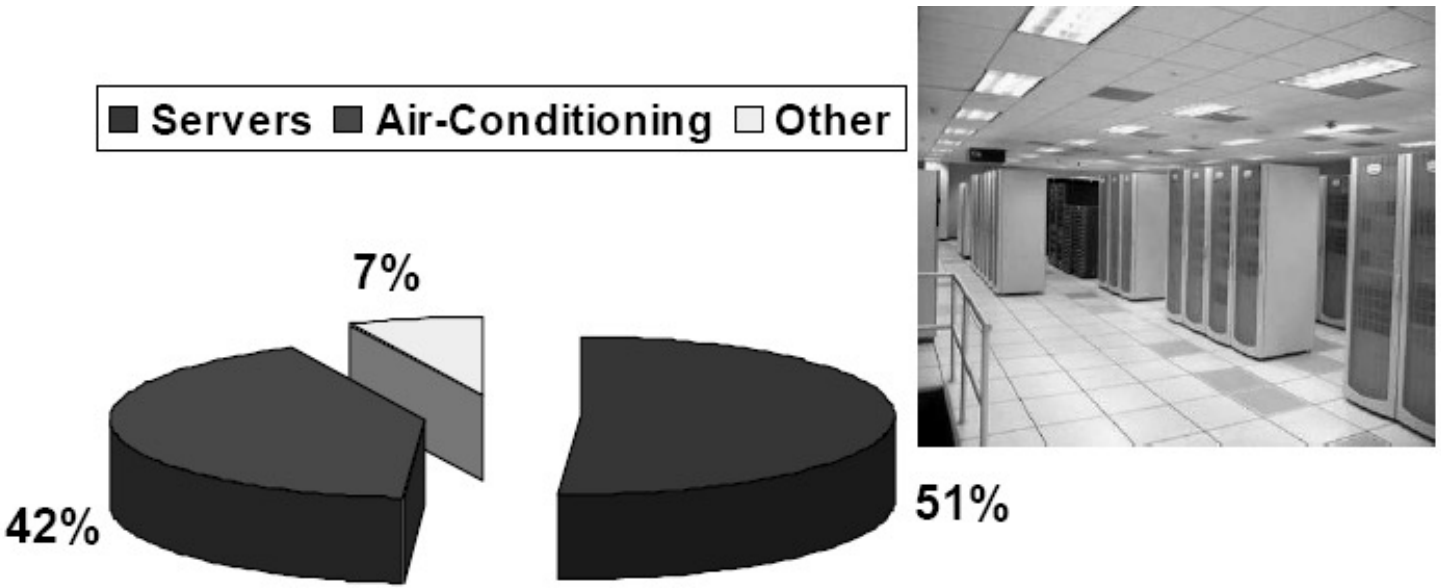
Heat Density Trends and Projections for IT Products



Sources:

1. "Intel's Huge Bet Turns Iffy", New York Times article, September 29, 2002
2. "Power, Heat, and Sledgehammer, Apr. 2002.

Data center cooling costs

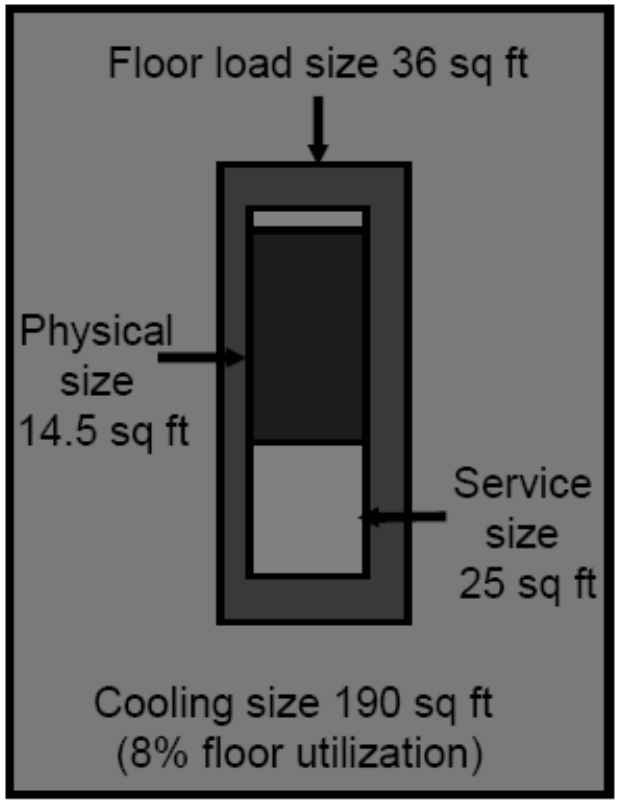


- Data center of a large financial institution in New York City
  - Power consumption ~ **4.8 MW**

Source: "Energy Benchmarking and Case Study – NY Data Center No. 2", Lawrence Berkeley National Lab, July 2003.

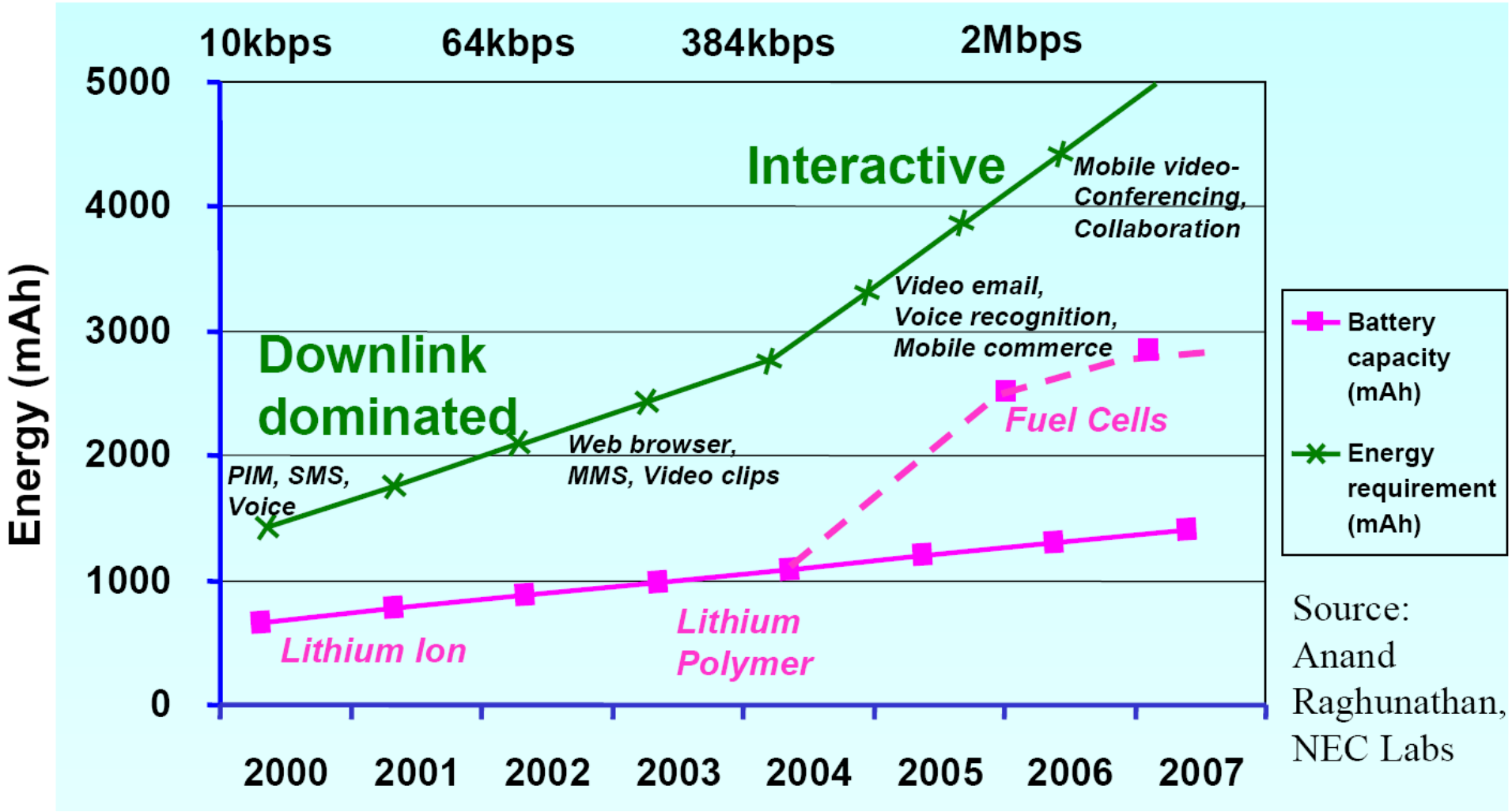
Machine room cooling implications

- If the trend continues a 200,000 ft^2 center could require 100 Megawatts of power
  - Add 60MW for mechanical room support
  - 160 MW is 16% of the output of a typical nuclear power plant
  - Yearly electricity costs would exceed $100M
  - Also cooling water resource and waste issues



Source: E. Kronstadt, IBM

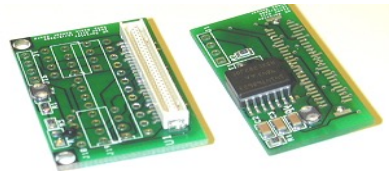
The Battery Gap in Portable Computing Systems



Source: Anand Raghunathan, NEC Labs

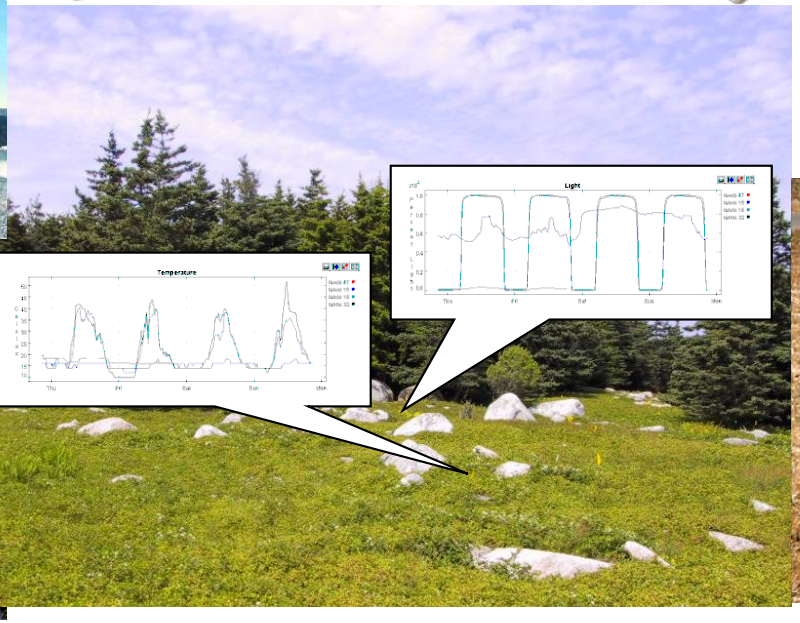
Emerging Application Domains

- Several months to years using 2AA batteries!
  - Using 2 x 2400mAH batteries, average power consumption should be less than 1mW to ensure 1 year lifetime

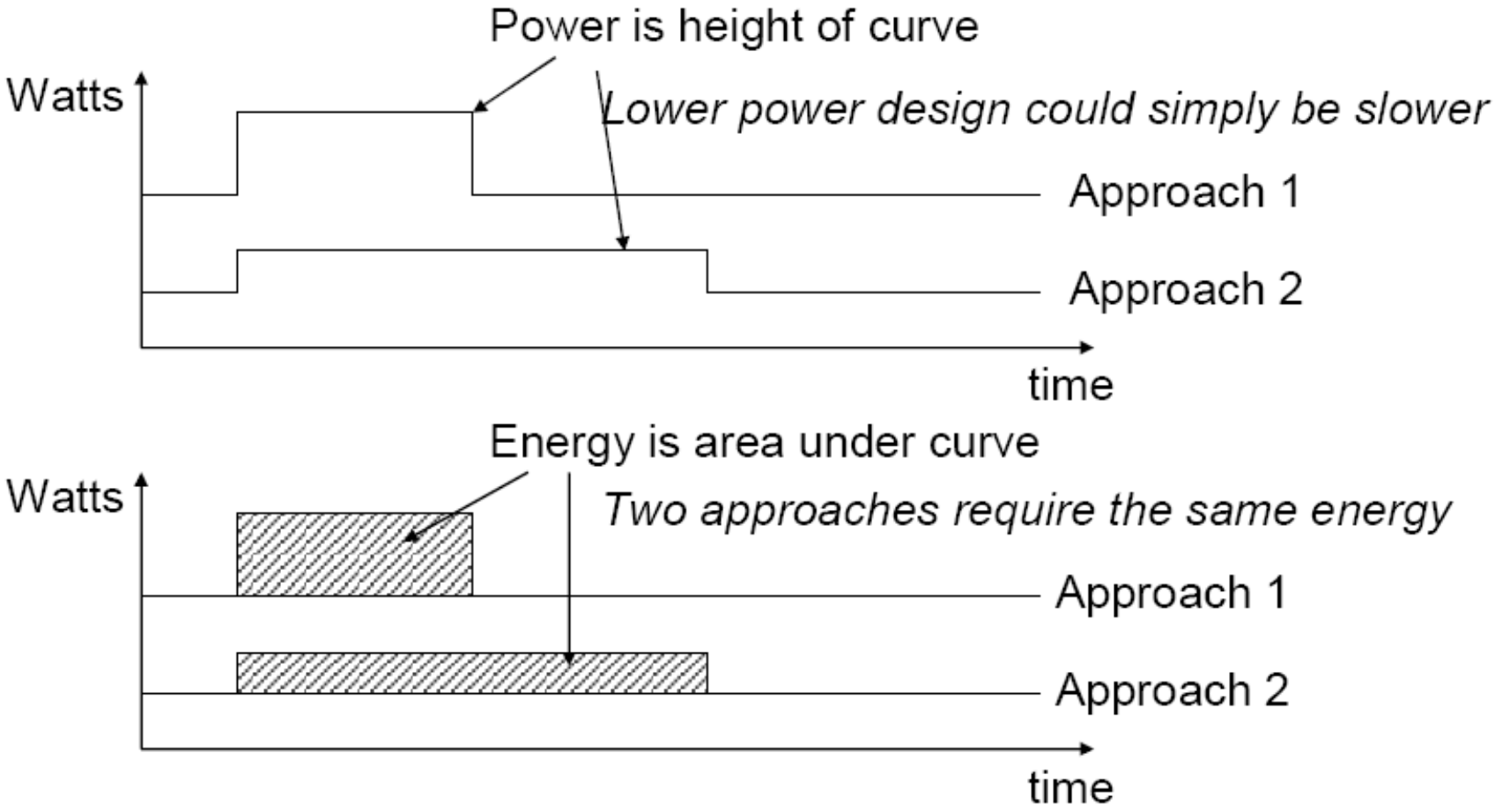









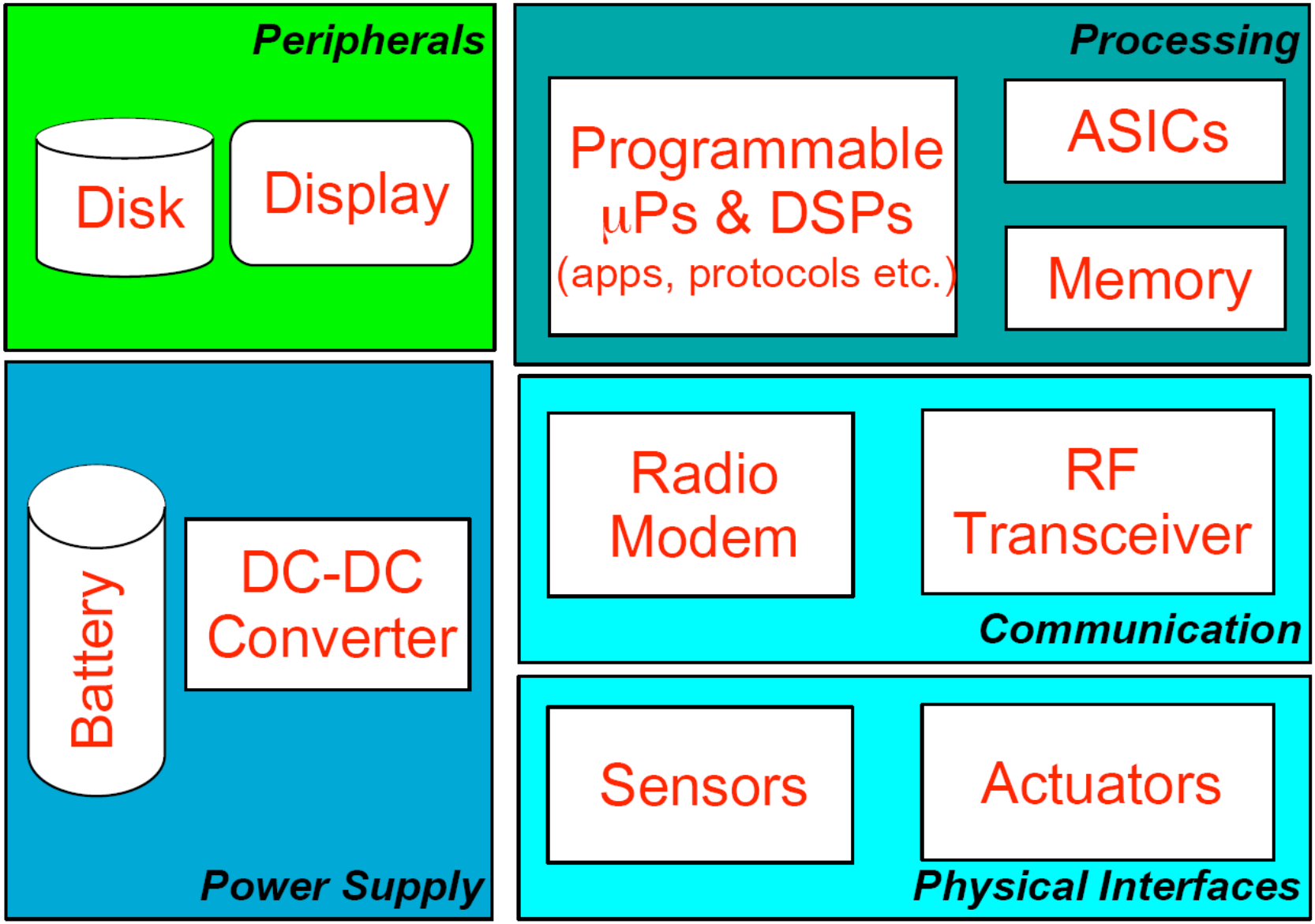
Power vs. Energy



Low Power vs. Energy Efficiency

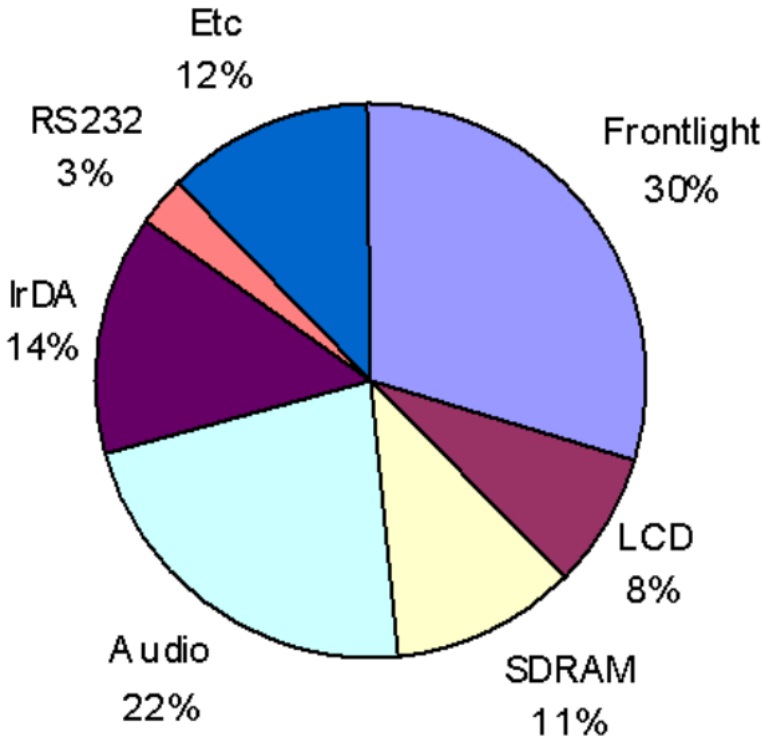
- Common embedded processors:
  - Atmel AVR, Intel 8051, StrongARM, XScale, ARM Thumb, SH Risc
- Large range of absolute power consumption, e.g.
  - 16.5 mW for ATMega128L @ 4MHz
  - 75 mW for ARM Thumb @ 40 MHz
- But, don't confuse low-power and energy-efficiency!
  - Example
    - 242 MIPS/W for ATMega128L @ 4MHz (4nJ/Instruction)
    - 480 MIPS/W for ARM Thumb @ 40 MHz (2.1 nJ/Instruction)
  - Other examples:
  - 0.2 nJ/Instruction for Cygnal C8051F300 @ 32KHz, 3.3V
  - 0.35 nJ/Instruction for IBM 405LP @ 152 MHz, 1.0V
  - 0.5 nJ/Instruction for Cygnal C8051F300 @ 25MHz, 3.3V
  - 0.8 nJ/Instruction for TMS320VC5510 @ 200 MHz, 1.5V
  - 1.1 nJ/Instruction for Xscale PXA250 @ 400 MHz, 1.3V
  - 1.3 nJ/Instruction for IBM 405LP @ 380 MHz, 1.8V
  - 1.9 nJ/Instruction for Xscale PXA250 @ 130 MHz, .85V (leakage!)
  - And, the above don't even factor in operand size differences!
- Moreover, need power management to actually exploit energy efficiency
  - Idle and sleep modes, variable voltage and frequency

Where Does the Power Go?



Power Breakdown of HP iPAQ

206MHz StrongArm SA-1110 processor
320x240 resolution color TFT LCD
Touch screen
32MB SDRAM / 16MB Flash memory
USB/RS-232/IrDA connection
Speaker/Microphone
Lithium Polymer battery
PCMCIA card expansion pack & CF card expansion pack



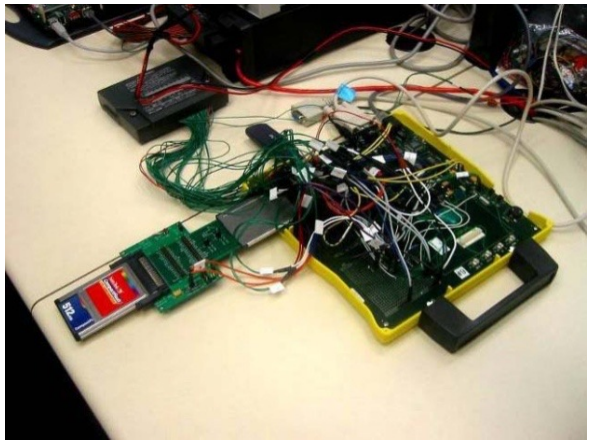
*** Note**

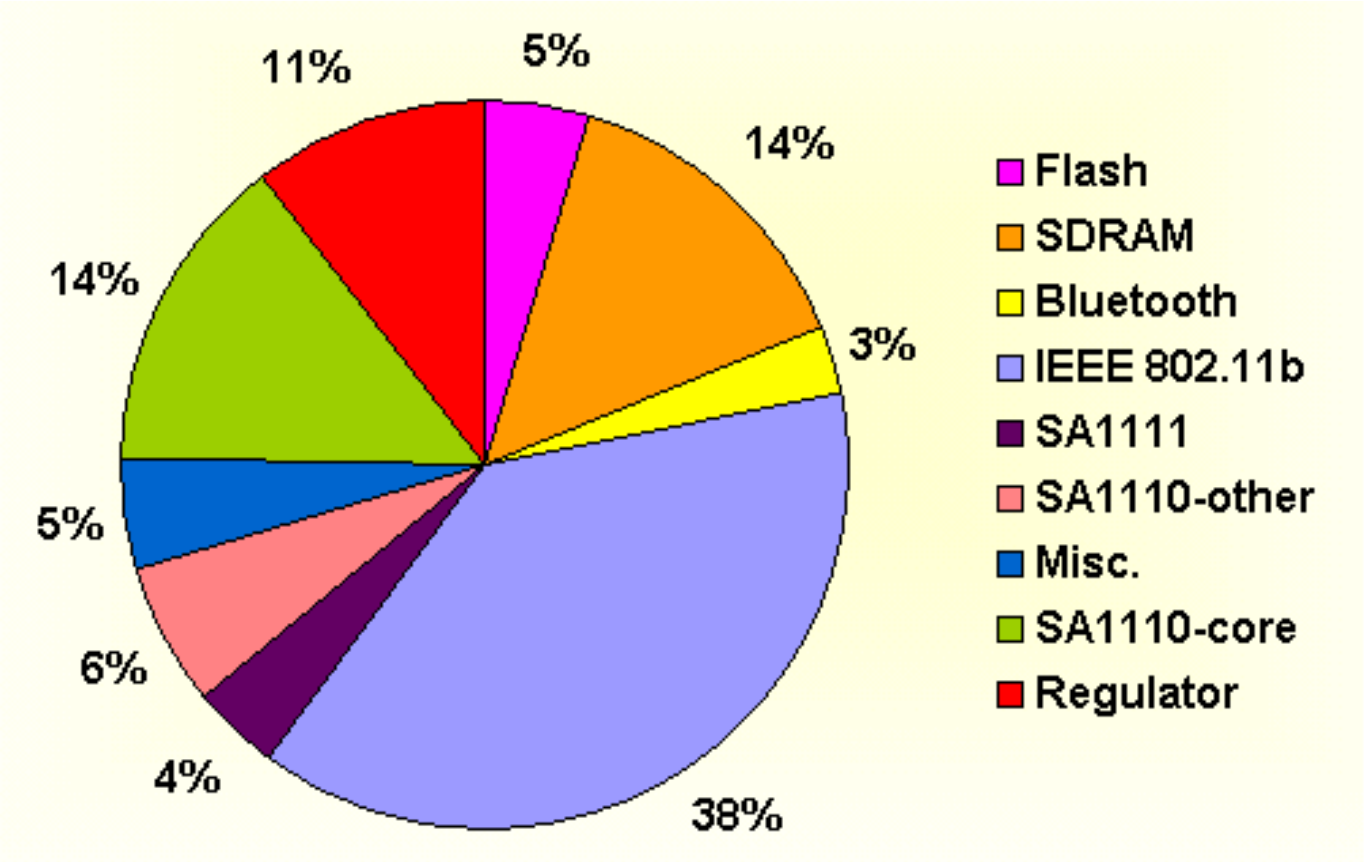
CPU is idle state of most of its time

Audio, IrDA, RS232 power is measured when each part is idling

Etc includes CPU, flash memory, touch screen and all other devices

Power Breakdown of the Intel Stargate Platform





Power Consumption in Post-PC Devices

- Wireless laptops, pocket computers, PDAs, wireless pads, wireless sensors, pagers, cell phones
- Energy and power usage of these devices is markedly different from normal computers
  - much wider dynamic range of power demand
  - share of memory, communication and signal processing subsystems become more important
    - disk storage and displays disappear or become simpler

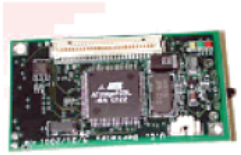
Major Power Hogs:

- Wireless modem
  - Rx is comparable, and may be more, than Tx
- DC-DC conversion
- Displays
  - Handheld flat panel display 6.4W
  - Helmet mounted display 4.9W
  - Integrated sight module display 2.6W
- Digital subsystem for protocol & applications
- Storage peripherals

Computation vs. Communication vs. Storage

Mote-class 802.15.4 Node

Transmit	2950 nJ/bit	Processor	4 nJ/op
Receive	2600 nJ/bit	~ 1400 ops/bit	



Microserver-class Node

Transmit	6600 nJ/bit	Processor	1.6 nJ/op
Receive	3300 nJ/bit	~ 6000 ops/bit	

Atmel AVR Flash (256b/page)

Write	470 nJ/bit to 120750 nJ/bit
Read	30 nJ/bit to 7600 nJ/bit

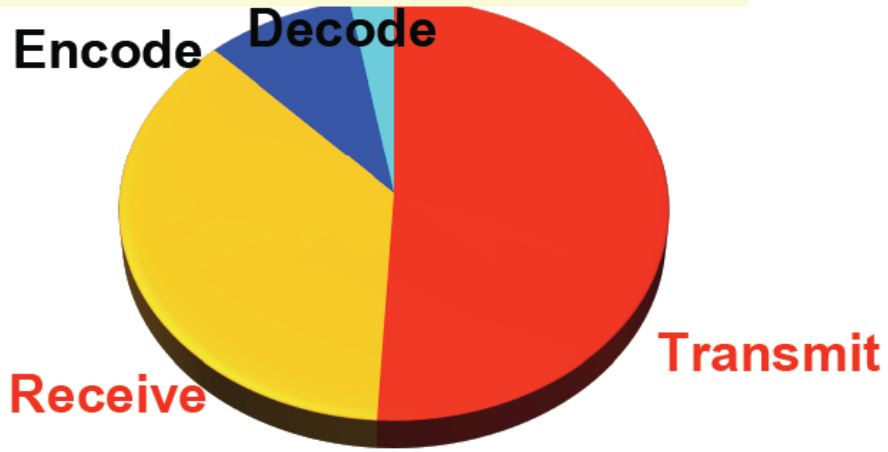
1GB NAND Flash Chip (512b/page)

Write	1 nJ/bit to 550 nJ/bit
Read	0.4 nJ/bit to 220 nJ/bit

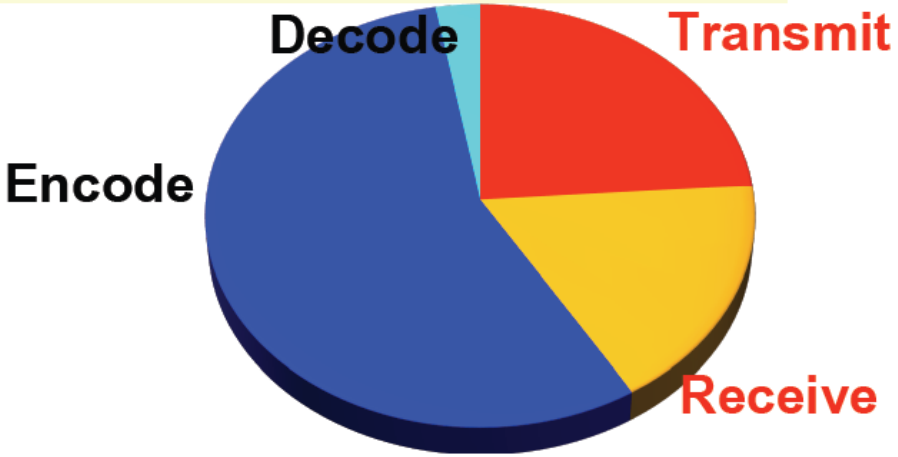
Energy/bit sent >> Energy/bit stored > Energy/op

Absolute Cost (Computation vs. Communication)

Energy breakdown for acoustic



Energy breakdown for image



Power Consumption in Digital Hardware

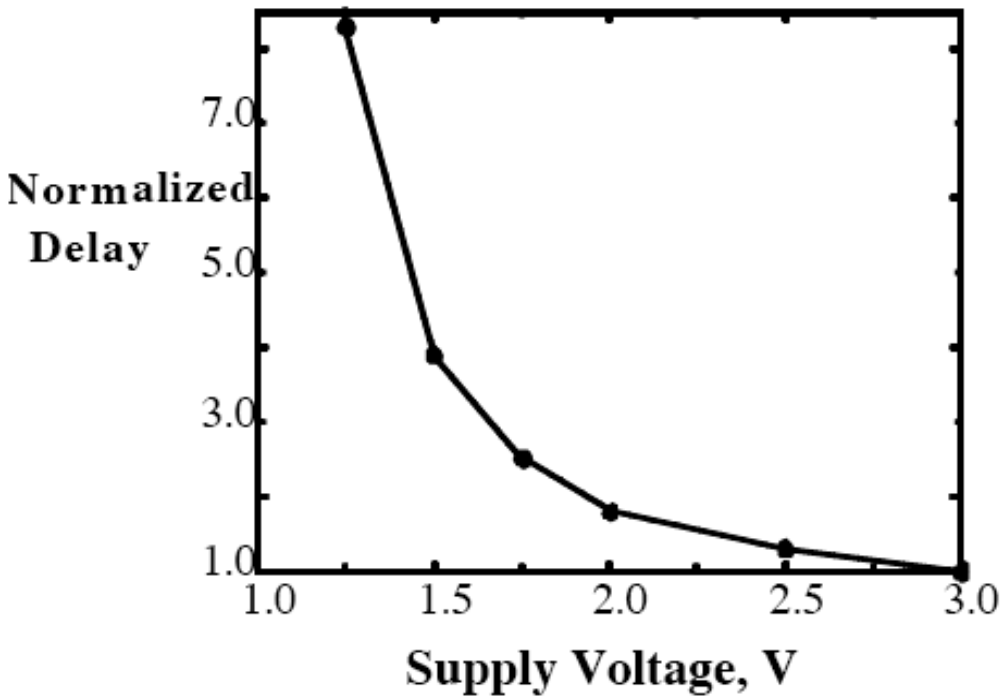
Recall: Power in Digital CMOS Hardware

- $\mathbf{P = A.C.V^2.f + A.I_{sw}.V.f + I_{leak}.V}$

- where
  - A = activity factor (probability of 0 → 1 transition)
  - C = total chip capacitance
  - V = total voltage swing, usually near the power supply voltage
  - f = clock frequency
  - I_{sw} = short circuit current when logic level changes
  - I_{leak} = leakage current in diodes and transistors

Why not simply lower V?

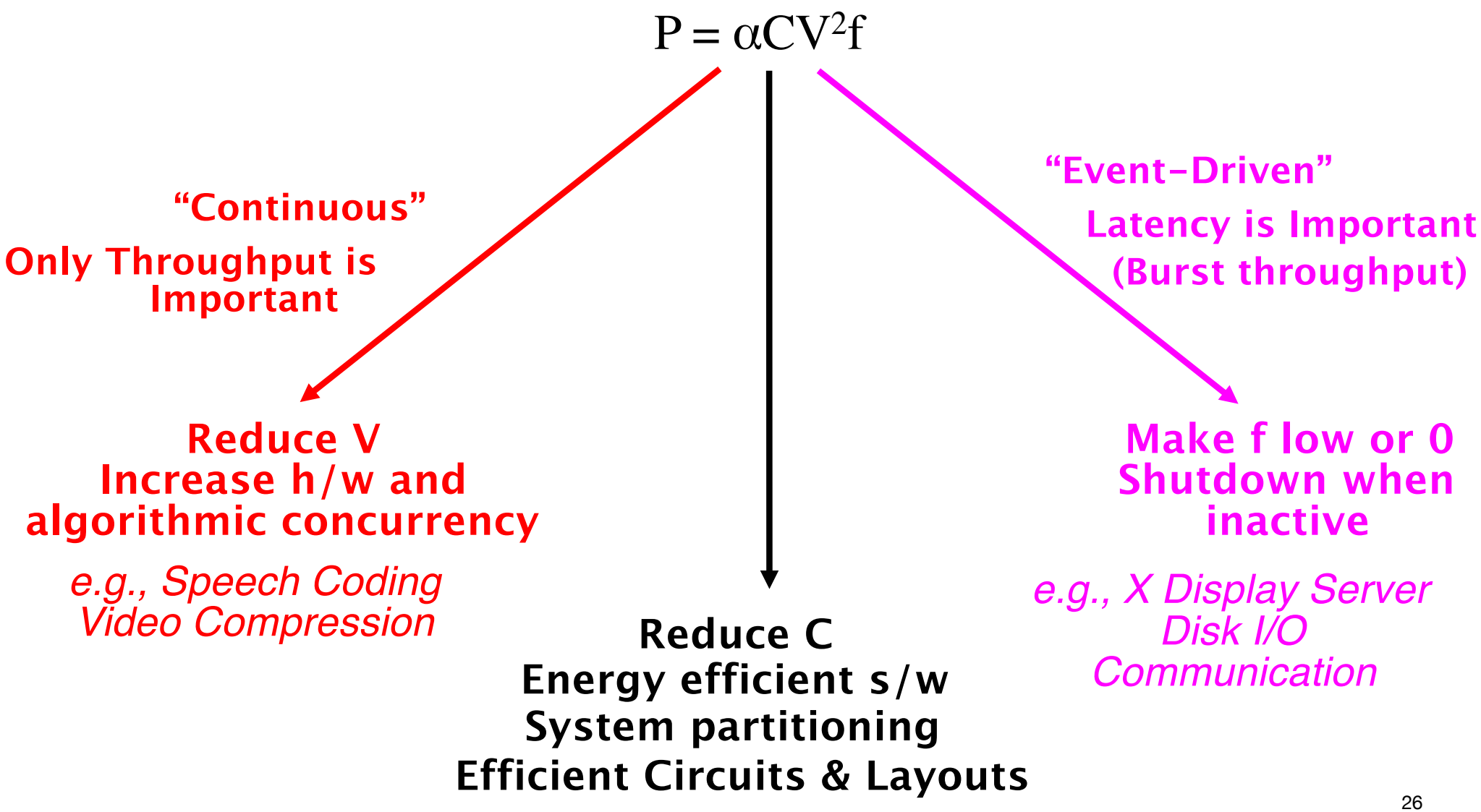
- Total P can be minimized by lower V
  - Lower V are a natural result of smaller feature sizes
- But, transistor speeds decrease dramatically as V is reduced to close to “threshold voltage”
  - performance goals may not be met
  - $t_d = CV / k(V-V_t)^{\alpha}$ where α is between 1-2
- Why not lower this “threshold voltage”?
  - makes noise margin and I_{leak} worse!
- Need to do smarter voltage scaling!



System Design for Low Power

- Need to explicitly design the system for power or energy efficiency
- IC technology still continue to help indirectly by increasing integration
  - more and faster transistors can enable low-power system architectures and design techniques
    - e.g. integration on a chip can reduce the significant I/O power consumption
- Energy efficient design of higher layers of the system also help
  - energy efficient protocols, power-aware apps. etc.
- Energy efficiency cuts across all system layers
  - entire network, not just the node
  - everything: circuit, logic, s/w, protocols, algorithms, UI, power supply...
    - complex global optimization problem
- Need to choose the right metric
  - e.g. individual node vs. network lifetime
- Trade-off between energy consumption & QoS
  - optimize energy metric while meeting QoS constraint
- Power-awareness, and not just low power
  - right energy at the right time and place

Approaches to Energy Efficiency

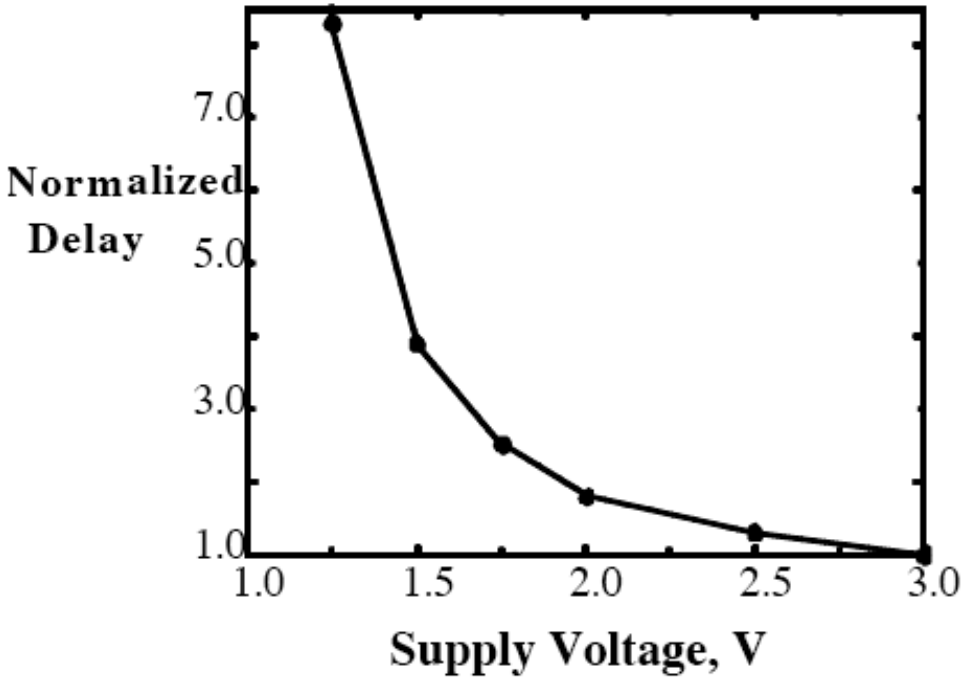


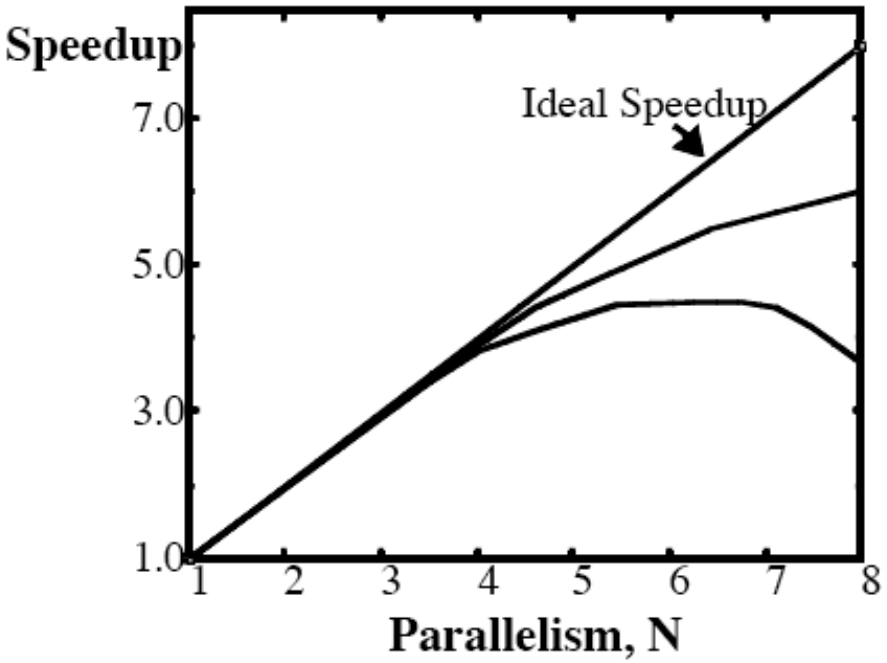
Reducing Supply Voltage: An Architectural Approach

- Operate at reduced voltage at lower speed
- Use architecture optimization to compensate for slower operation
  - e.g., concurrency, pipelining via compiler techniques
- Architecture bottlenecks limit voltage reduction
  - degradation of speed-up
  - interconnect overheads
- Similar idea for memory: slower and parallel

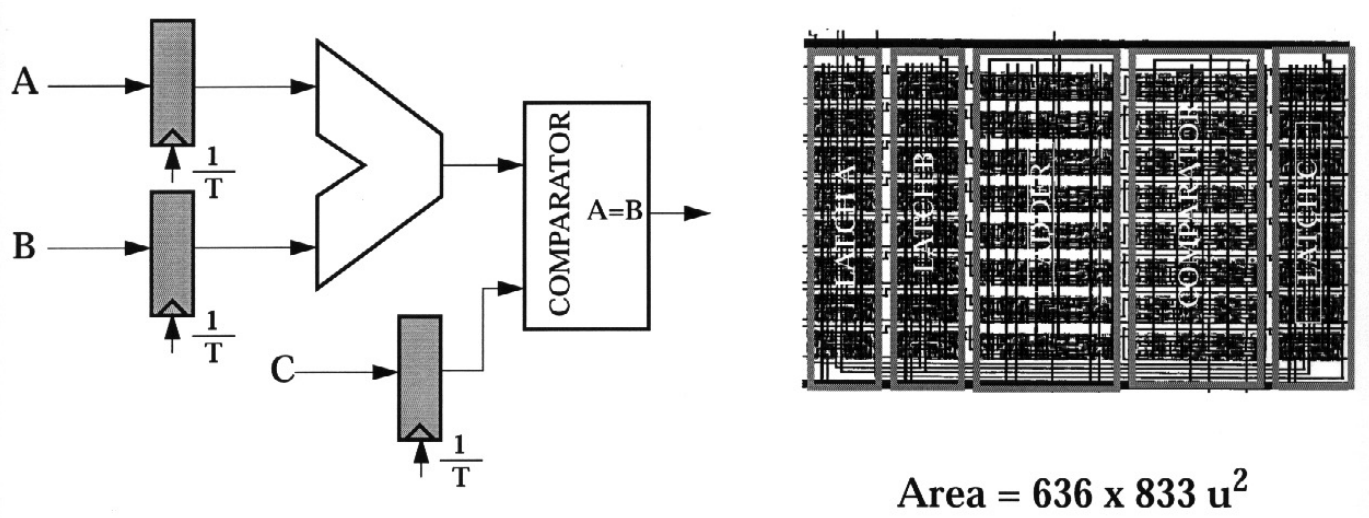
- **Trade-off AREA for lower POWER**

Example: Voltage-Parallelism Trade-off





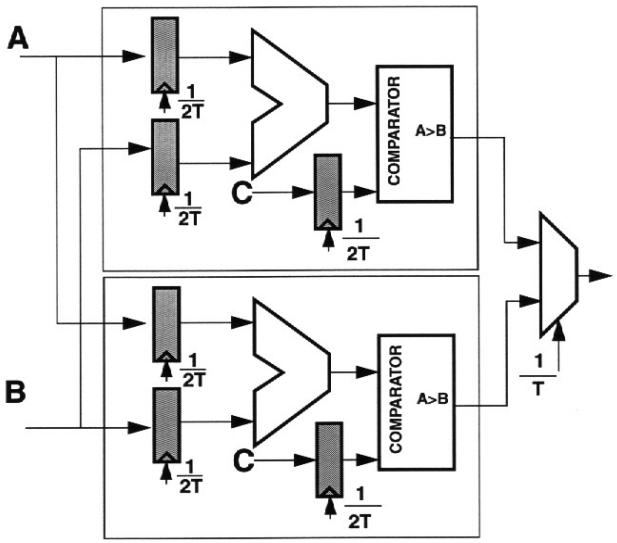
Example: Reference Datapath Design

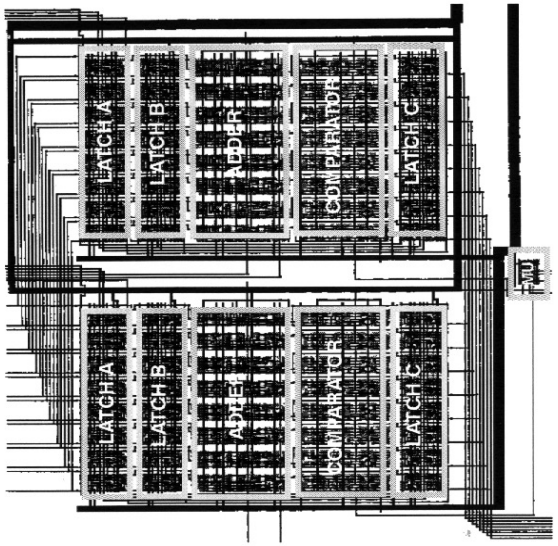


- Critical path delay: $T_{adder} + T_{comparator} = 25$ ns
- Frequency: $f_{ref} = 40$ MHz
- Total switched capacitance = C_{ref}
- $V_{dd} = V_{ref} = 5V$
- Power for reference datapath = $P_{ref} = C_{ref}V_{ref}^2f_{ref}$

from “Digital Integrated Circuits” by Rabaey

Parallel Datapath



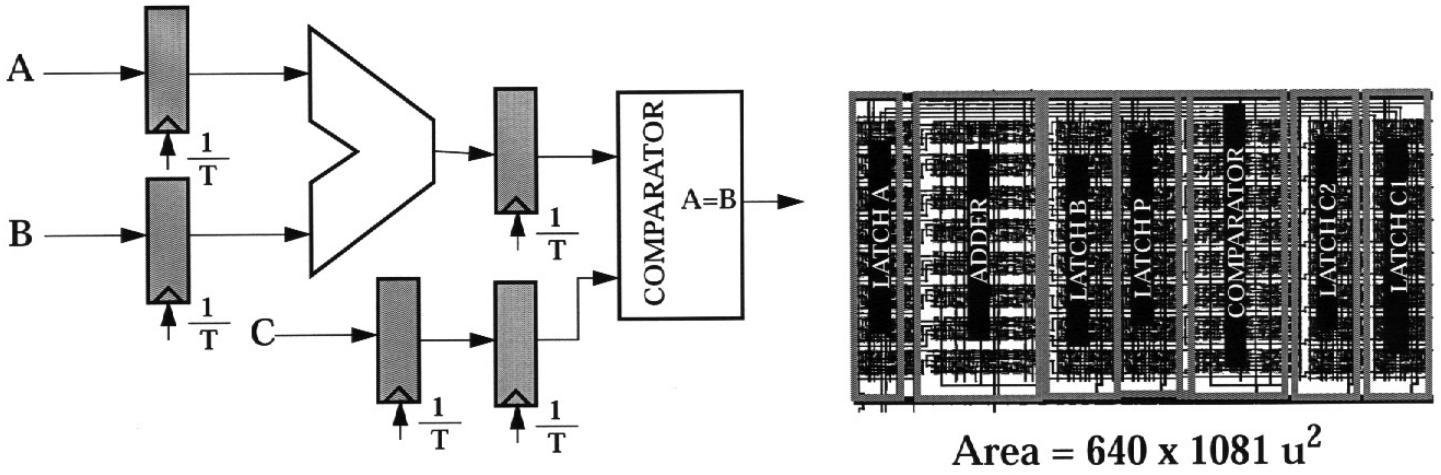


Area = 1476 x 1219 μ^2

- The clock rate can be reduced by x2 with the same throughput: $f_{par} = f_{ref}/2 = 20$ MHz
- Total switched capacitance = $C_{par} = 2.15C_{ref}$
- $V_{par} = V_{ref}/1.7$
- $P_{par} = (2.15C_{ref})(V_{ref}/1.7)^2(f_{ref}/2) = 0.36P_{ref}$

from “Digital Integrated Circuits” by Rabaey

Pipelined Datapath



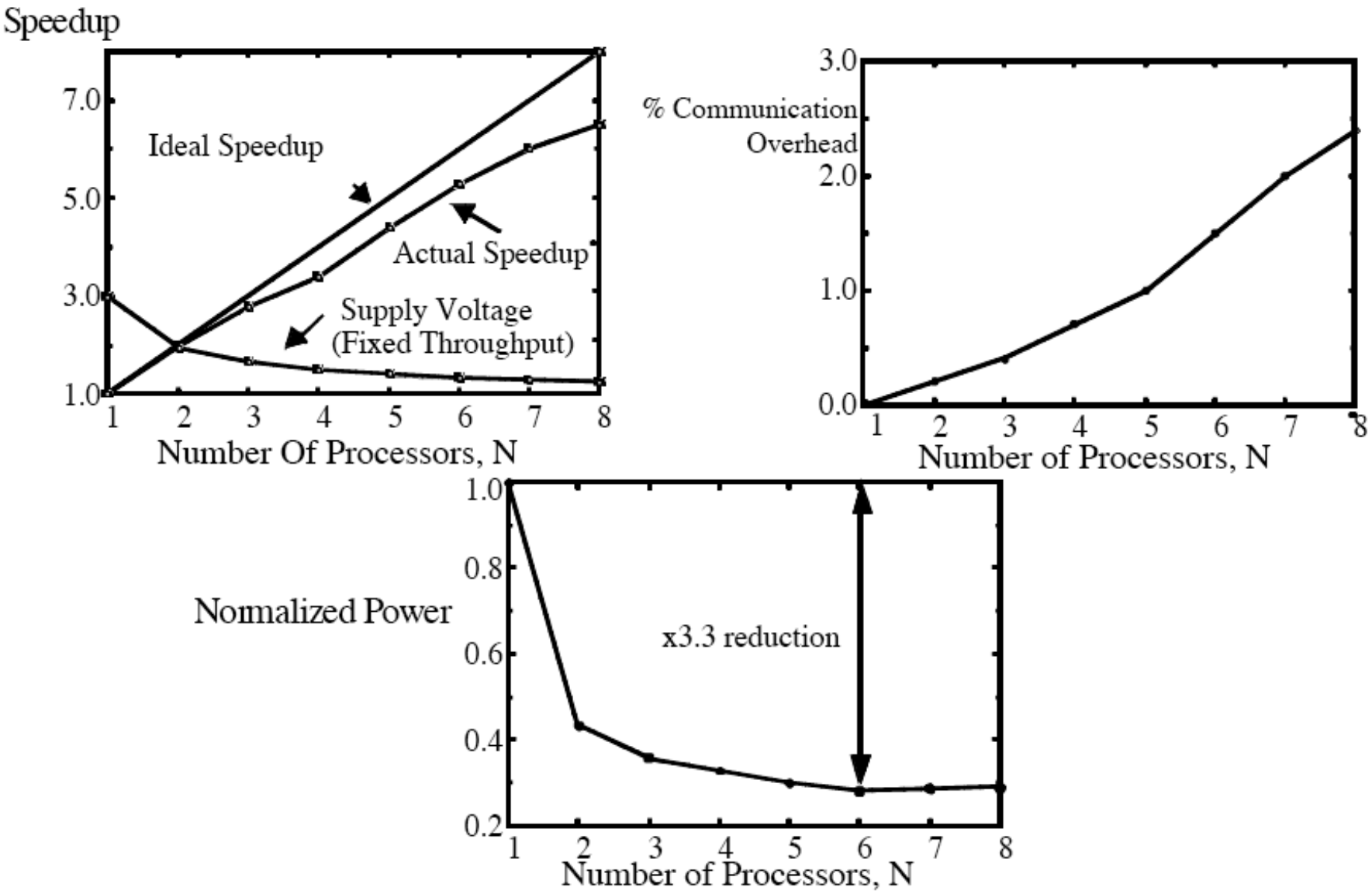
- $f_{pipe} = f_{ref}$
  $C_{pipe} = 1.1C_{ref}$
  $V_{pipe} = V_{ref}/1.7$
- Voltage can be dropped while maintaining the original throughput
- $P_{pipe} = C_{pipe}V_{pipe}^2f_{pipe} = (1.1C_{ref})(V_{ref}/1.7)^2f_{ref} = 0.37P_{ref}$

from “Digital Integrated Circuits” by Rabaey

Datapath Architecture-Power Trade-off Summary

Datapath Architecture	Voltage	Area	Power
Original	5V	1	1
Pipelined	2.9V	1.3	0.37
Parallel	2.9V	3.4	0.34
Pipeline-Parallel	2.0V	3.7	0.18

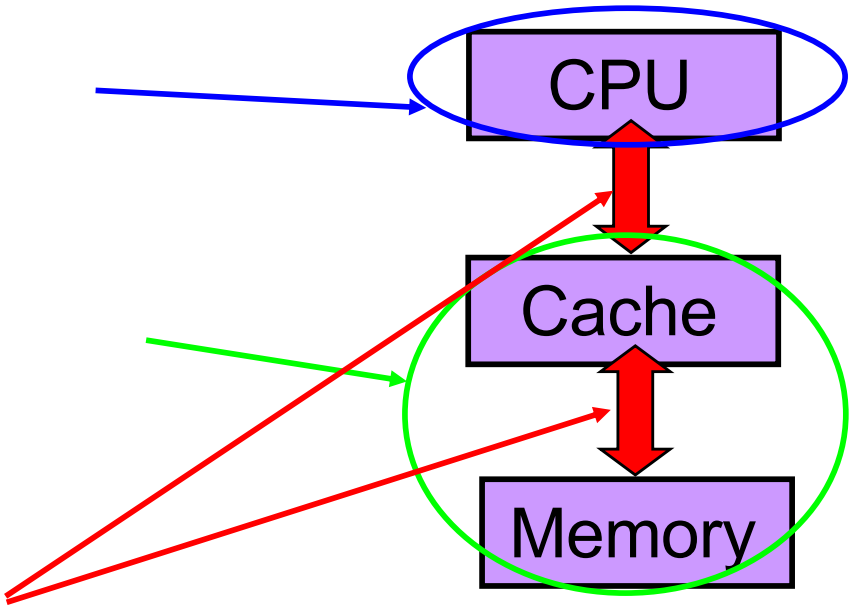
Example of Voltage Scaling



Other Techniques

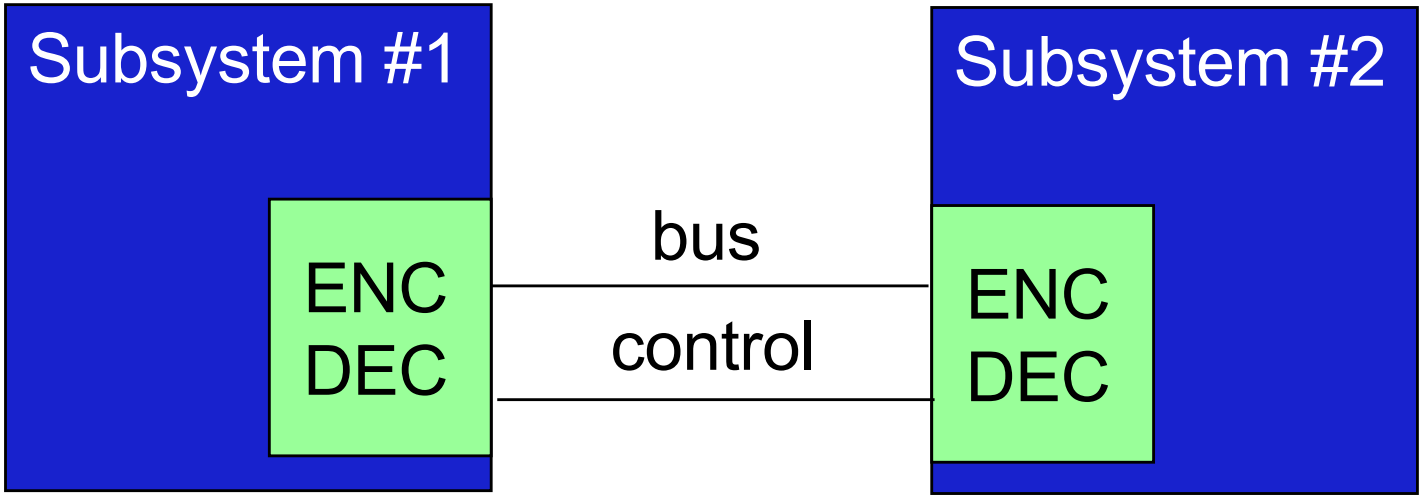
Low-power Software

- Software drives the underlying hardware
  - Significant energy savings can be obtained by clever software
- Strategies for low-power software
  - Code running on CPU
    - Usual compiler optimization
    - Instruction selection, ordering, and packing
    - Operand swapping
    - Processor-specific instructions
  - Code accessing memory
    - Data structures
    - Memory access optimization
    - Better use of registers
  - Data flowing over busses
    - I/O coding
  - Compiler controlled power management



Energy Efficient I//O Encoding

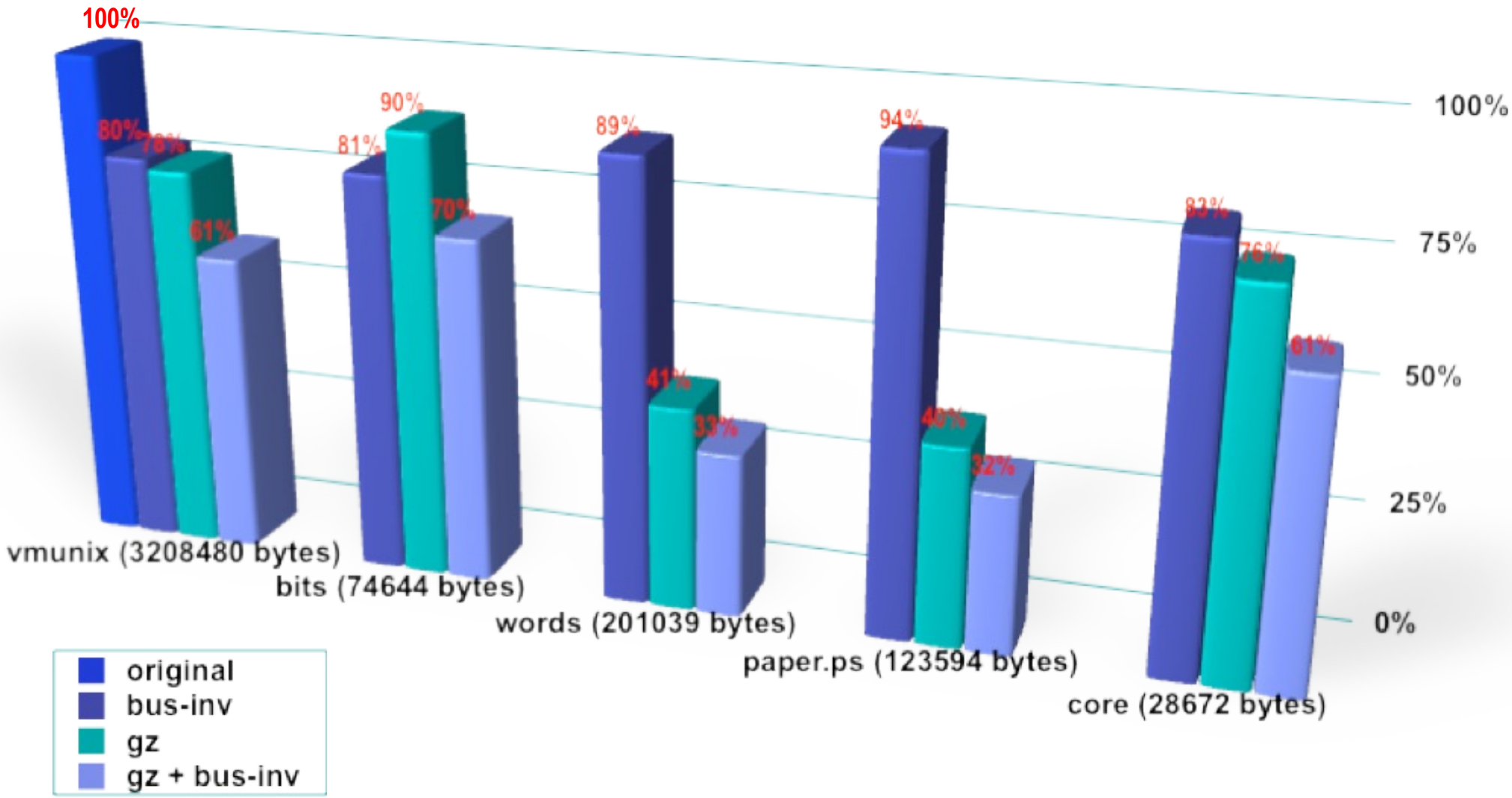
- C of system busses is >> C inside chips
  - large amount of power goes to I/O interfaces
    - 10-15% in uPs, 25-50% in FPGAs, 50-80% in logic
  - encoding bus data can reduce the power significantly
    - but need to handle encoding/decoding cost (power, latency)
  - exploit correlations in data being sent over the bus



Examples

- Compression to remove redundancy
- Gray code on address busses
  - addresses usually increment sequentially by 1
  - modified code that increments by 4 or 8 for word oriented CPUs
- T0 code for address busses
  - add redundant INC line
    - INC=0 : address is equal to the bus lines
    - INC=1 : Tx freezes the other bus lines, and Rx increments the previously transmitted address by a pre-agreed stride
- Bus-Invert (BI) Coding
  - transmit D or invert(W), whichever results in fewer transitions
    - an extra signal indicates polarity
  - Performance: max N/2 switches, optimal in average case for 1-bit redundancy codes
- Hybrid coding for shared address/data busses
  - INC and INV control, plus a SEL to distinguish between data vs. address mode
- Encode based on statistical analysis of bus traces
    - calculate spatio-temporal correlation (on-line or off-line)

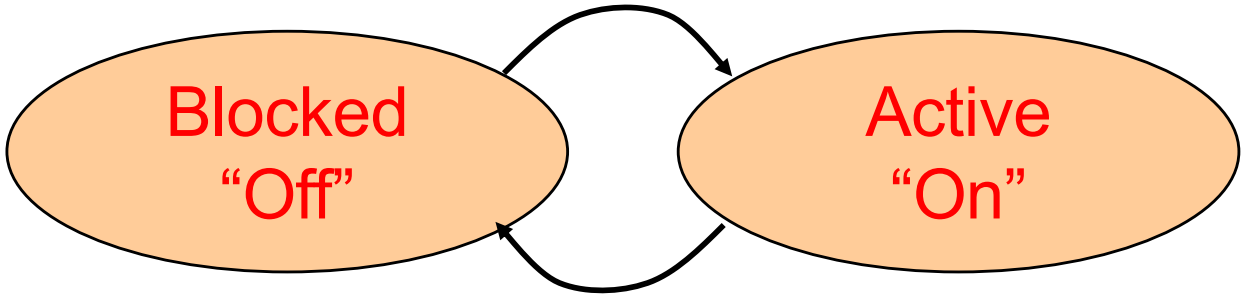
Example:
Normalized # of Transitions for Typical UNIX Files



from [Stan97]

Power Management via Shutdown

Shutdown for Energy Saving



T_{block} T_{active} **ideal improvement = 1 + T_{block}/T_{active}**

- Subsystems may have small duty factors
  - CPU, disk, wireless interface are often idle
  - But on-off pattern may not be deterministic

- Huge difference between "on" & "off" power
  - E.g., CPUs
    - TI MSP430: 3 mW (active) / 2 μW sleep
    - StrongARM: 400mW (active) / 50 mW (idle) / 0.16 mW (sleep)
  - E.g., 2.5" Hard Disk [Harris95]:
    - 1.35W (idle spin) / 0.4W (standby) / 0.2W (sleep) / 4.7W (start-up)

Example: Potential CPU Power Reduction in a Wireless X11 Display Terminal

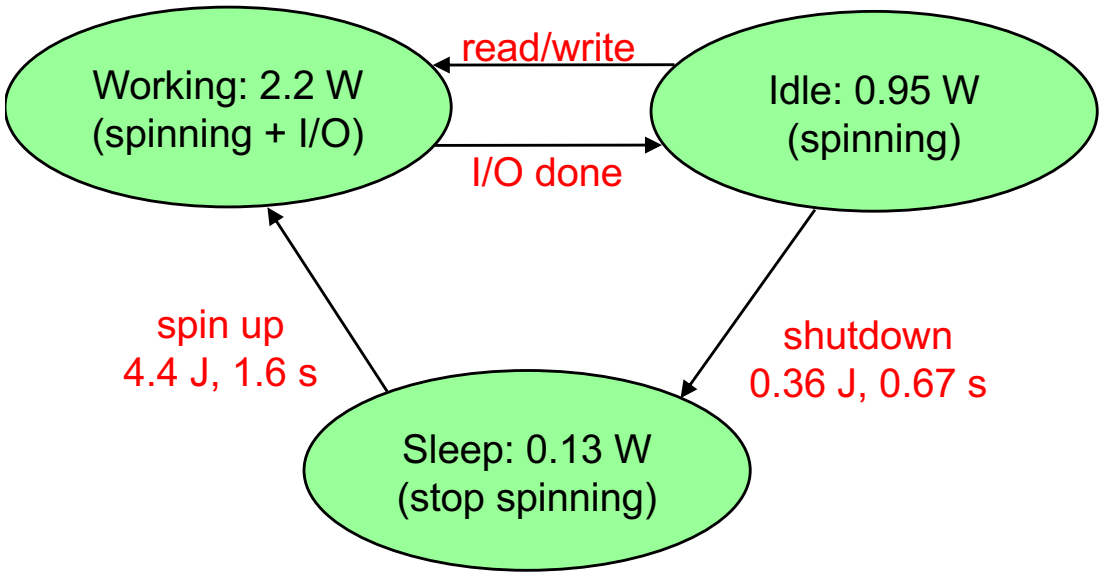
	Trace 1	Trace 2	Trace 3
Trace Length (sec)	**5182.48**	**26859.9**	**995.16**
Toff (sec)	**5047.47**	**26427.4**	**960.82**
Ton (sec)	**135.01**	**432.5**	**34.34**
Toff/(Toff+Ton)	**0.9739**	**0.9839**	**0.9655**
Max Energy Reduction	**x38.4**	**x62.1**	**x29.0**

- 96-98% time spent in the blocked state
- Average time in the blocked state is short (<< a second)

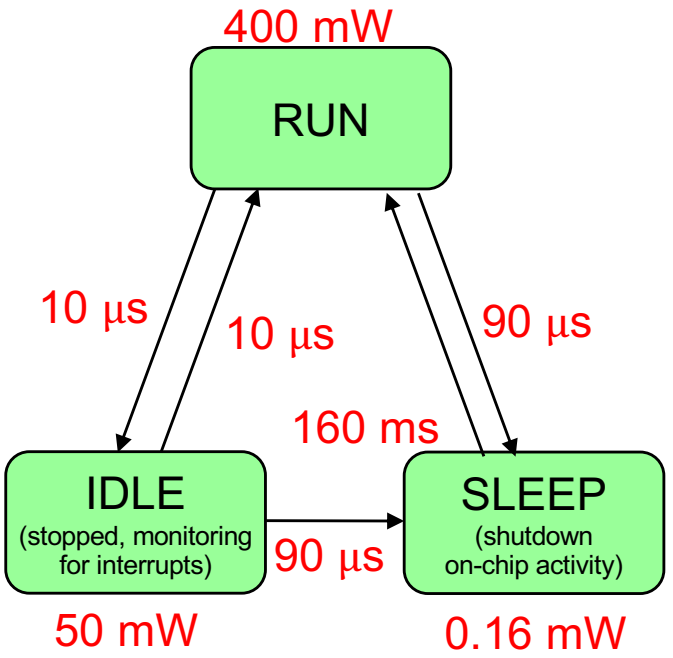
Examples

State	Power Consumption in Watts	Start-up Energy in Joules	Transition Time to Active
Sleep	0	4.75	5 S
Stand-by	.2	1.575	1.5 S
Idle	.9	.56	40 mS
Active	1.9	0	0

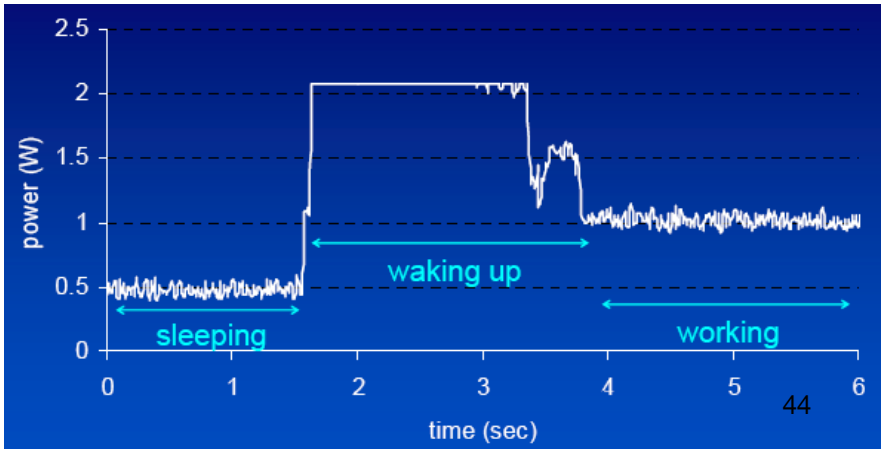
IBM mobile Hard Drive



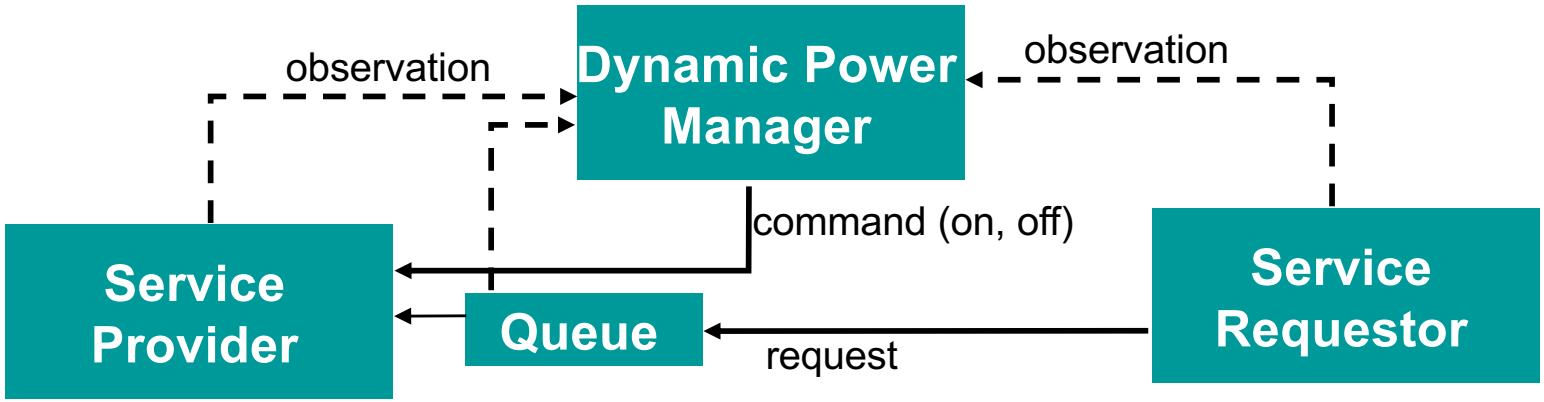
Fujitsu MHF 2043 AT Hard Drive



SA-1100 CPU

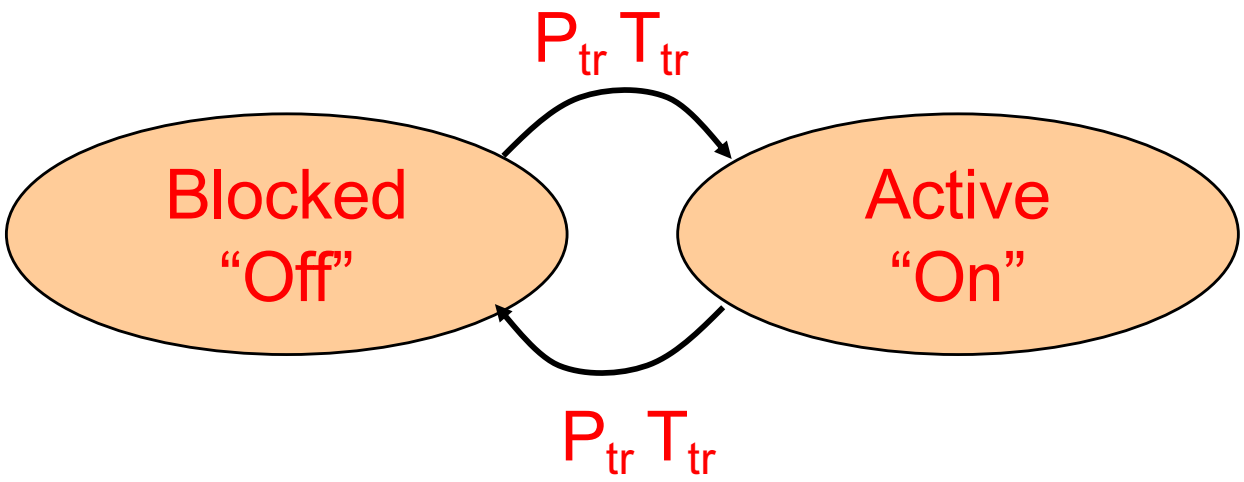


Generic Power-managed System



- An abstract & flexible interface between power-manageable components (chips, disk driver, display driver etc.) & the dynamic power manager
  - But need insight on how & when to power manage
    - power management policy
  - Essentially PM is a controller that needs to be synthesized
    - collect observations, issue commands
    - implemented in software or hardware, with negligible energy consumption
- Components (service providers) with several internal states
  - Corresponding to power and service levels
  - Can be abstracted as a power state machine
    - power and service annotation on states
    - power and delay annotation on edges
  - Self-managed vs. externally managed

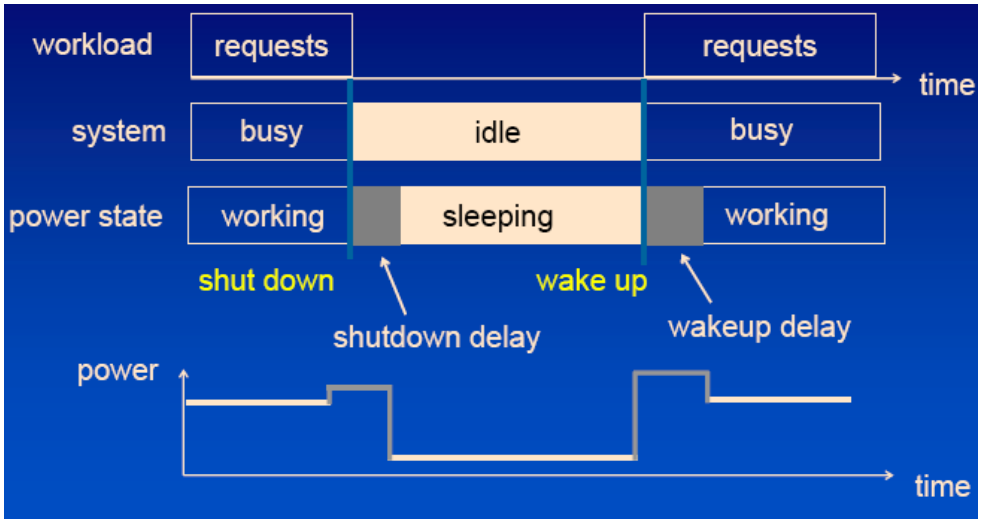
When is DPM useful?



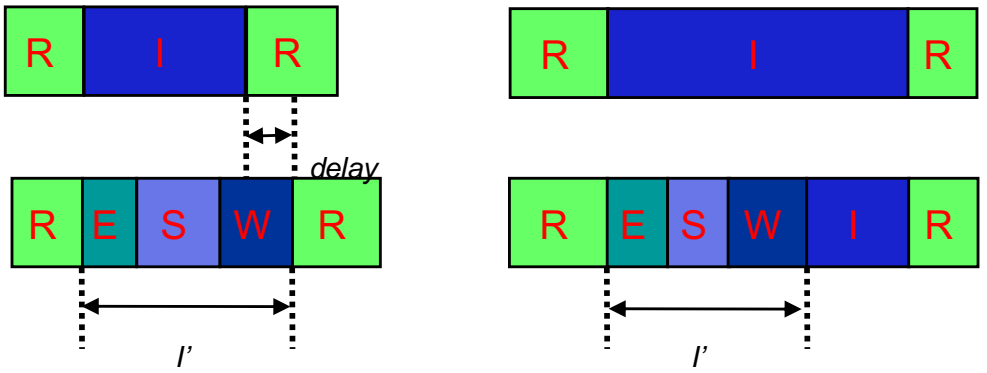
- If $T_{tr}=0$, $P_{tr}=0$ then DPM policy is trivial
  - Stop a component when it is not needed
- If, as is the reality, $T_{tr}\neq 0$, $P_{tr}\neq 0$
  - Shutdown only when idleness is going to be long enough to make it worthwhile
  - Complex decision if the time spent in state is not deterministic

Problems in Shutdown

- Cost of restarting: latency vs. power trade-off
  - increase in latency (response time)
    - e.g., time to save restore CPU state, spin up disk
  - increase in power consumption
    - e.g. higher start-up current in disks
- When to Shutdown
  - Optimal
  - Idle Time Threshold: shutdown if have been idle long enough
  - Predictive: shutdown if predicted idle time will be long enough
- When to Wakeup
  - Optimal
  - On-demand
  - Predictive: predict the arrival of next activity and pre-wakeup
- Cross-over or Breakeven point for shutdown to be effective



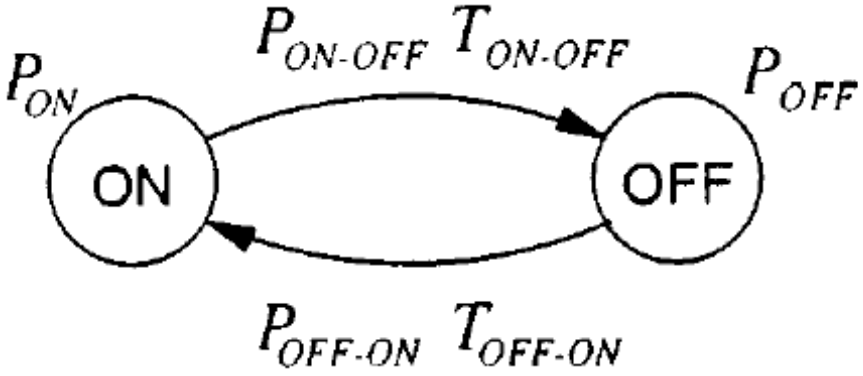
Workload and System Model



Pre-wakeup

Breakeven Point

- Breakeven point: minimum idle time that would make it worthwhile to shutdown
- DPM worthwhile when T_{BE} < Average T_{idle}



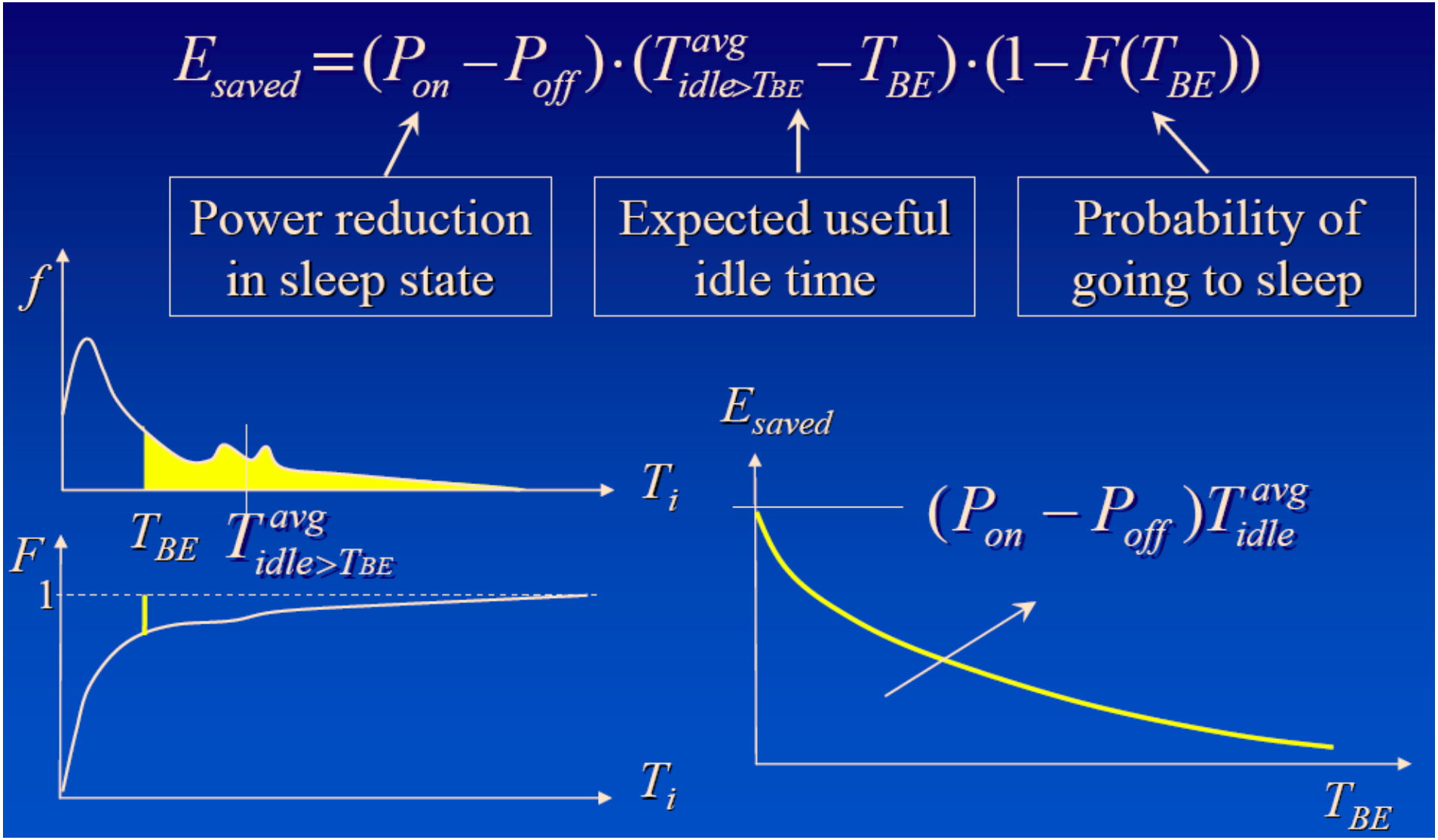
$$T_{TR} = T_{\mathrm{On,\,Off}} + T_{\mathrm{Off,\,On}}$$

$$P_{TR} = \frac{T_{\mathrm{On,\,Off}} P_{\mathrm{On,\,Off}} + T_{\mathrm{Off,\,On}} P_{\mathrm{Off,\,On}}}{T_{TR}}$$

$$T_{BE} = T_{TR} + T_{TR} \frac{P_{TR} - P_{\mathrm{On}}}{P_{\mathrm{On}} - P_{\mathrm{Off}}} \text{ if } P_{TR} > P_{\mathrm{On}}$$

$$T_{BE} = T_{TR} \text{ if } P_{TR} \leq P_{\mathrm{On}}.$$

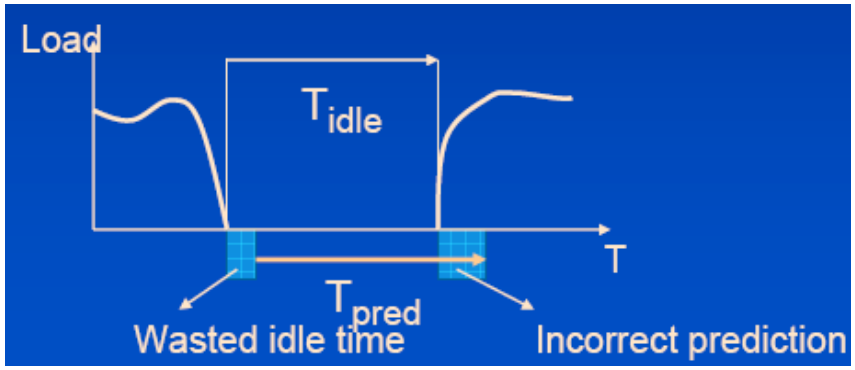
Effect of T_{BE} and $F(T_{idle})$ on Energy Savings



Ack: Simunic, UCSD, 2005

DPM Approaches: Predictive

- Approach:
  - Use a priori information or run-time observation to model workload behavior
  - Predict idle time and schedule shutdown and/or wakeup accordingly
    - Workload-specific predictor: deterministic vs. stationary stochastic vs. non-stationary stochastic
    - Prediction accuracy

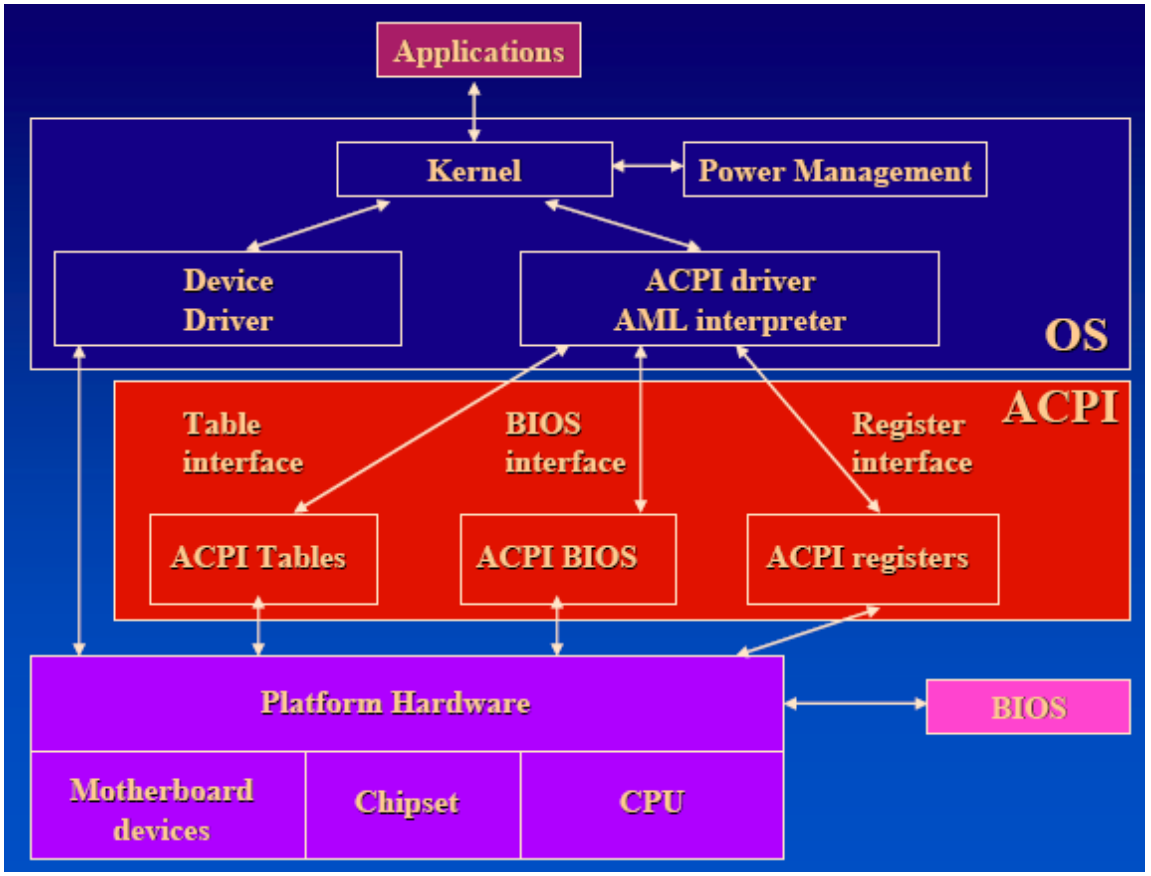


- Static techniques
  - E.g., fixed timeout $T_{threshold}$ with on-demand wakeup
    - Rationale: $P(T_{idle} > T_{threshold} + T_{BE} \mid T_{idle} > T_{threshold}) \approx 1$
    - Choice of $T_{threshold}$ crucial
    - $T_{threshold} = T_{BE}$ yields energy consumption not more than 2x worse than ideal oracle policy
    - Worst case when point activities are separated by $T_{idle} = 2T_{BE}$
    - Limitations: latency penalty after every shutdown, power wasted during $T_{threshold}$

- Adaptive techniques
  - E.g., maintain set of time out values to figure out how successful it would have been
  - E.g., weighted timeouts where weights based on performance relative to oracle policy
  - E.g., increase and decrease timeout based on its performance

Implementing DPM

- How?
  - Clock gating
  - Supply shutdown
  - Display shutdown
  - Motor shutdown
- Where?
  - User
  - Application
  - Compiler
  - OS
  - Firmware
  - Hardware

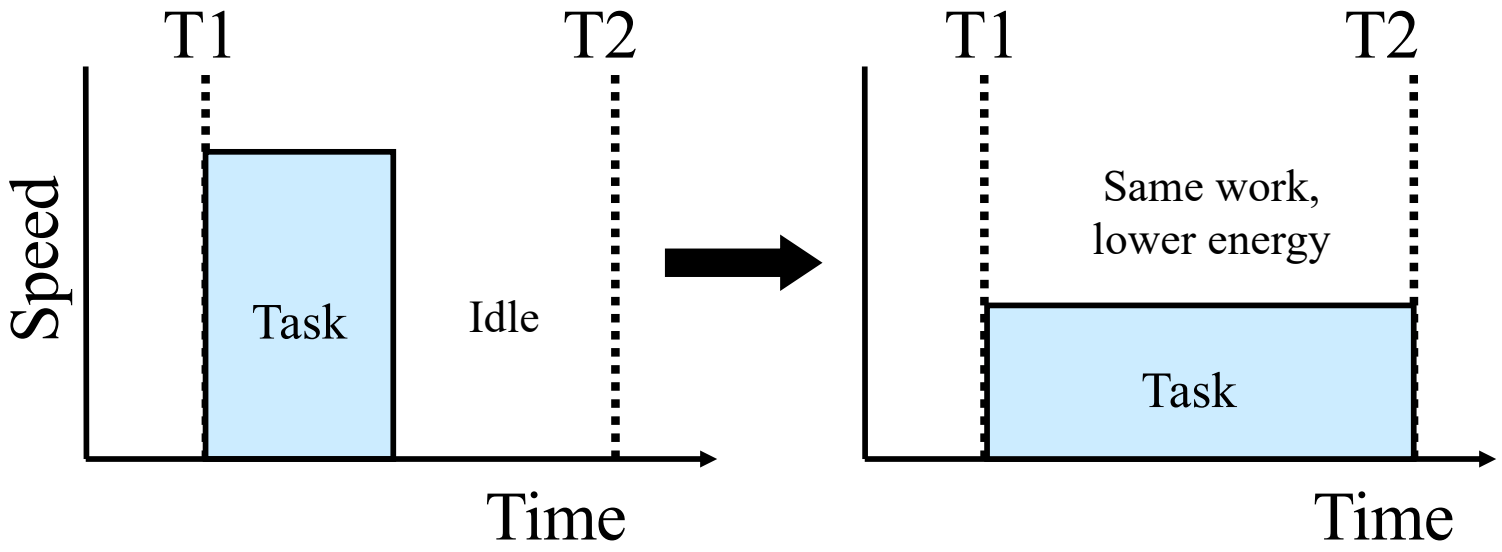


Advanced Configuration and Power Interface (ACPI)

Shutdown vs. Variable Voltage

Voltage Reduction is Better

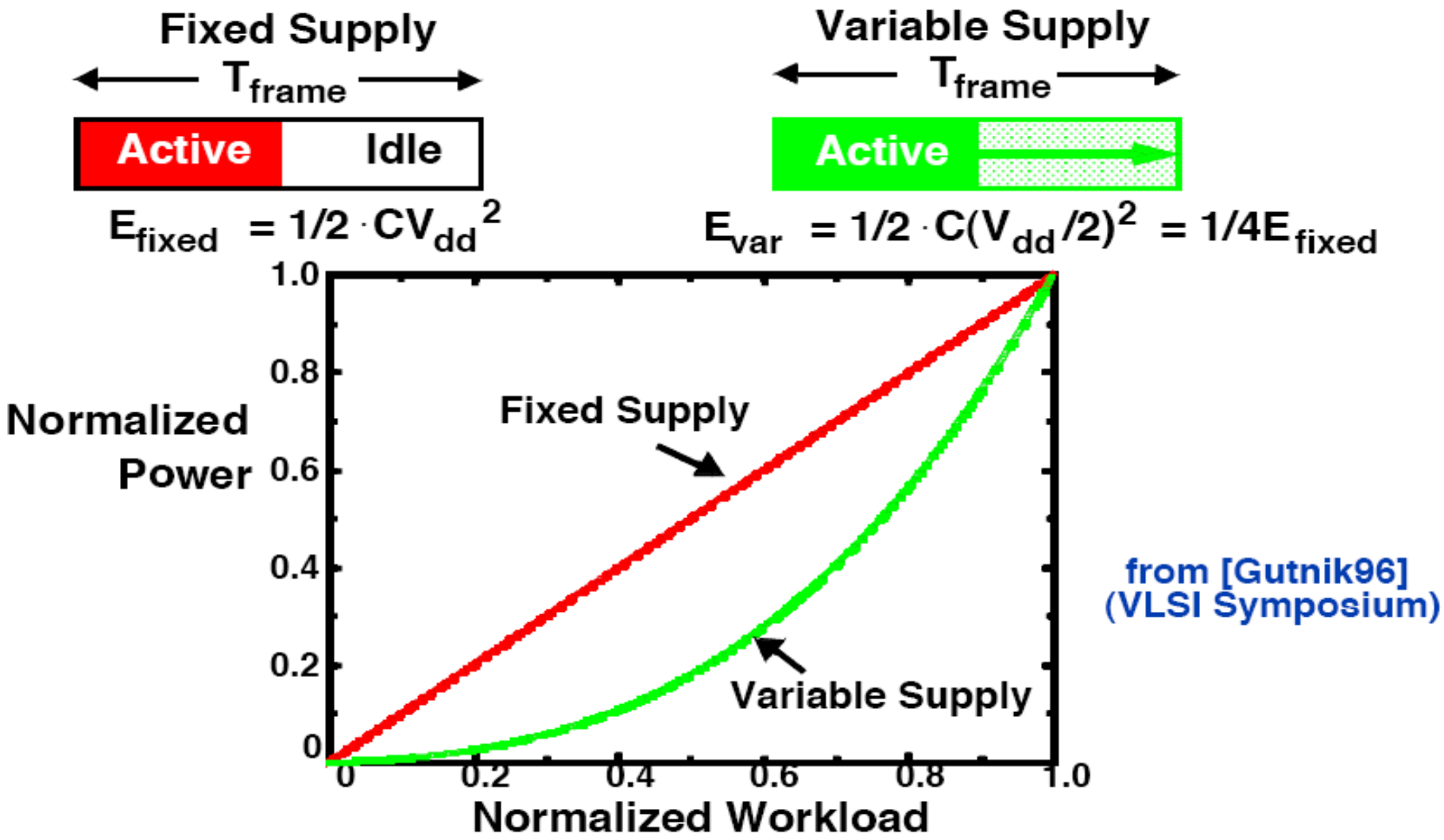
- Example: task with 100 ms deadline, requires 50 ms CPU time at full speed
  - normal system gives 50 ms computation, 50 ms idle/stopped time
  - half speed/voltage system gives 100 ms computation, 0 ms idle
  - same number of CPU cycles but 1/4 energy reduction



Problem with Voltage Reduction

- Voltage gets dictated by the tightest (critical) timing constraint
  - not a problem if latency not important
    - throughput can always be improved by pipelining, parallelism etc.
  - but, real systems have bursty throughput and latency critical tasks
- Solution: dynamically vary the voltage!

Varying the Supply Voltage



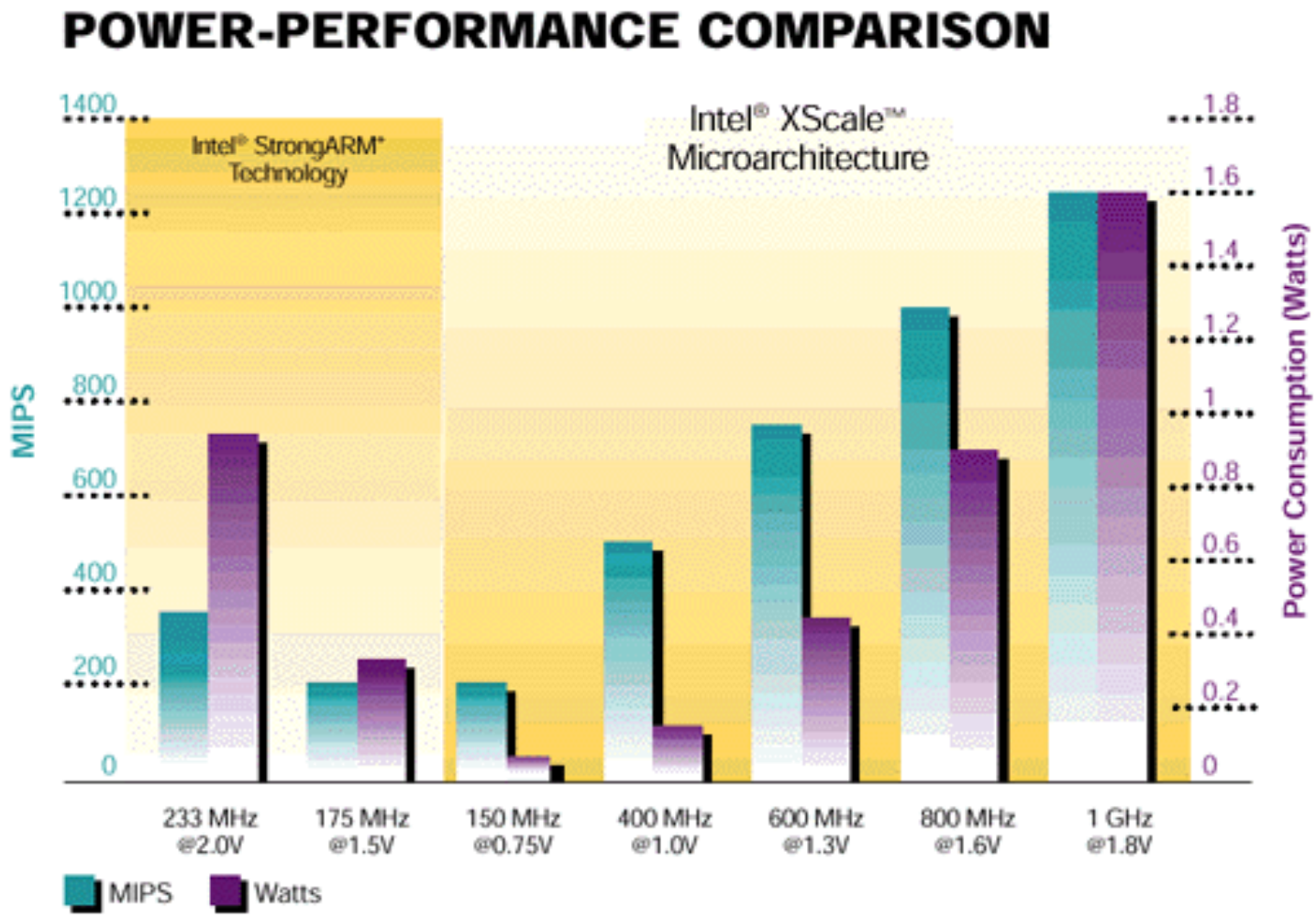
Dynamically Variable Voltage

- Use voltage to control the operating point on the power vs. speed curve
  - power and clock frequency are functions of voltage
- Technology exists
  - efficient variable voltage DC-DC regulators available commercially
  - most CMOS chips operate over a range of voltage
- Main problem is algorithmic:
  - one has to schedule the voltage variation as well!
    - via compiler or OS or hardware

Intel's Xscale (StrongARM-2)

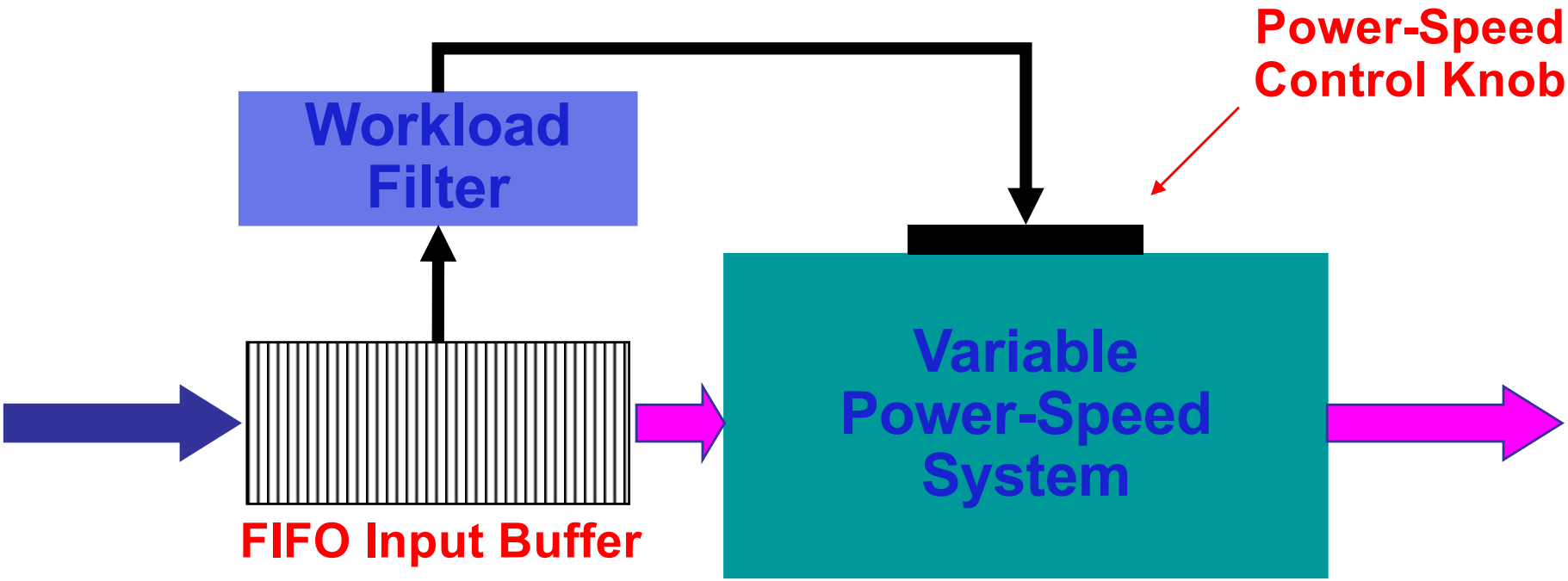
- Ultra-low power + high performance
  - via 7-stage pipeline ("Superpipeline")
- Compliant with ARM 5TE ISA
  - 16-bit Thumb instructions, additional DSP instructions
- Power-awareness features
  - Dynamic voltage and frequency scaling on the fly
    - energy per op dynamic range of ~6x in SA-2 vs. ~2x for SA-1
      - SA2: 1 mW (standby); 40-mW/185-MIPS @ 150-MHz/0.75-V ; 450-mW/750-MIPS @ 600-MHz/1.3-V; 900-mW/1000-MIPS @ 800-MHz/1.6-V
        (source: Intel's web site)
      - SA1: 33-mW @ 59-MHz/0.8-V to 360-mW @ 206-MHz/1.5V in 11 discrete steps
      - (source: estimated from Anantha Chandrakasan's publications)
    - 30 µS PLL-relock vs. 150 µS for SA-1
      - regulator transition time may be bottleneck?
  - Idle, sleep, and quick wakeup modes
    - 100 µW drowsy state
    - Functional block powered up only when needed
- MAC coprocessor for signal processing

XScale vs. StrongARM



How to Exploit Dynamic Voltage Scaling?

- Two observations:
  - bursty traffic or variable workload
  - workload averaging helps due to convex shape of power-speed curve
- Generic system architecture
  - many examples in hardware and software

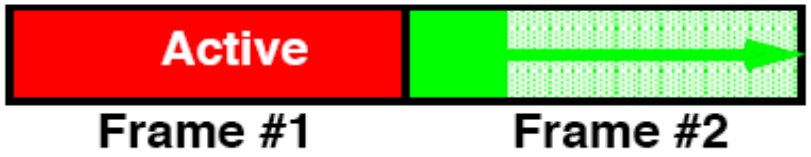


Dynamic Scheduling of System Voltage

- Observation:
  - most energy efficient way to execute N instructions in an interval T is to use a constant voltage & frequency
  - can't change V & f instantaneously: several μs to ms
    - dynamic system
- Ideal goal:
  - schedule voltage changes to minimize the energy consumed while meeting all the task deadlines
- Approaches:
  - Heuristics for general purpose interactive OSs
    - [Weiser94], [Govil95]
  - Optimal approaches under deadline constraints
    - [Yao95], [Hong98a], [Hong98b], [Hong98c]

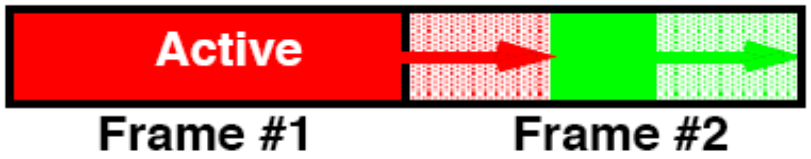
Workload Averaging Helps

Without Averaging:



$$E_{var} = CV_{dd}^2 + 1/4 \cdot C(V_{dd}/4)^2$$
$$= 65/64 \cdot CV_{dd}^2$$

With Averaging:



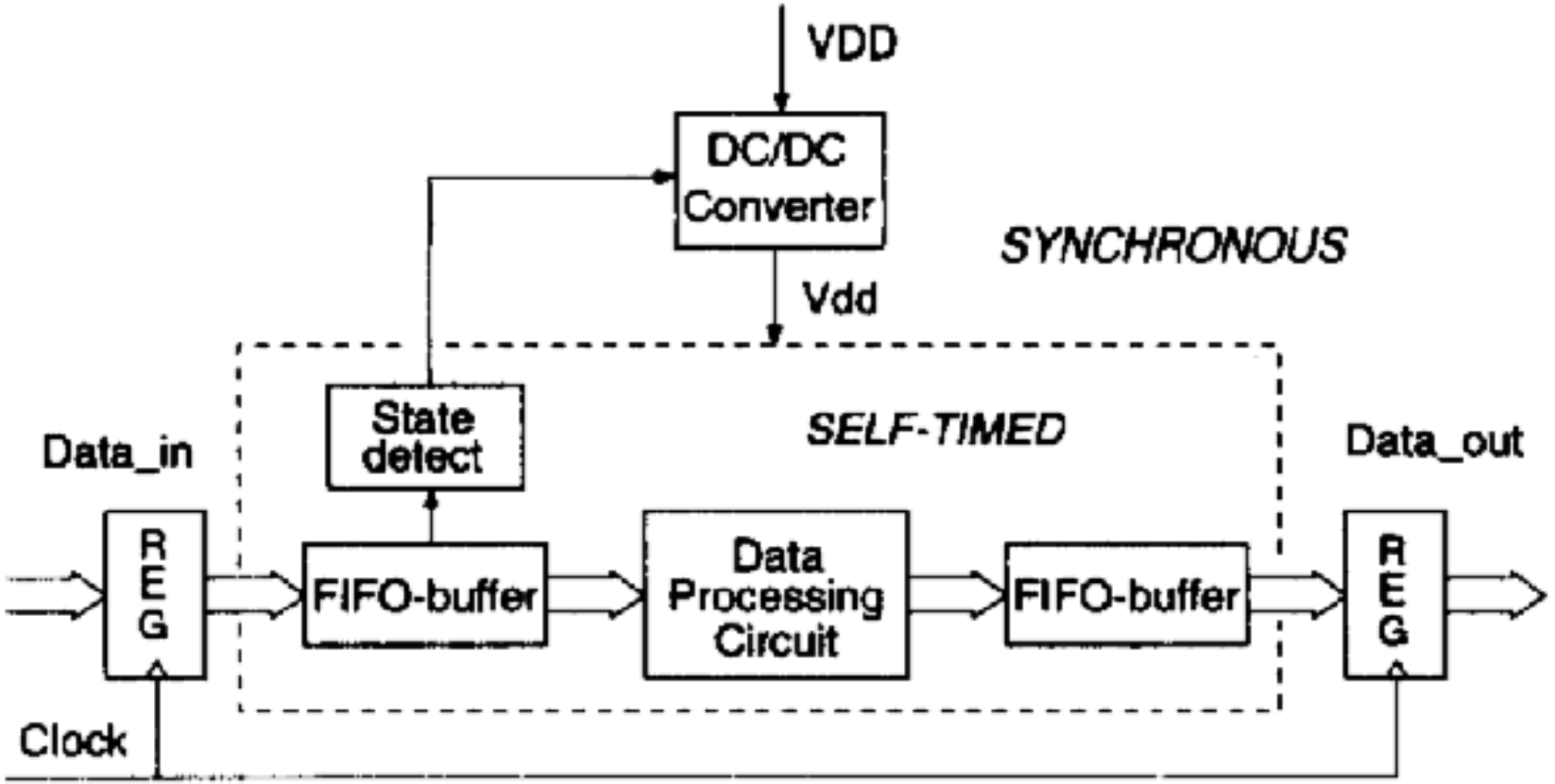
$$E_{avg} = 5/4 \cdot C(5/8 \cdot V_{dd})^2$$
$$= 125/256 \cdot CV_{dd}^2$$
$$\approx 0.5\ E_{var}$$

DVS in General-purpose OSs using Workload Averaging

- Approach #1: [Weiser94]
  - time divided into 10-50 ms intervals
  - f & V raised or lowered at the beginning of the interval based on CPU utilization during the previous interval
    - 50% savings for a processor in the range 3.3V-5V
    - 70% savings for a processor in the range 2.2V-5V

- Approach #2: [Govil95]
  - predicts CPU cycles needed in the next interval
  - sets f & V accordingly
  - many prediction strategies: some did well, others not

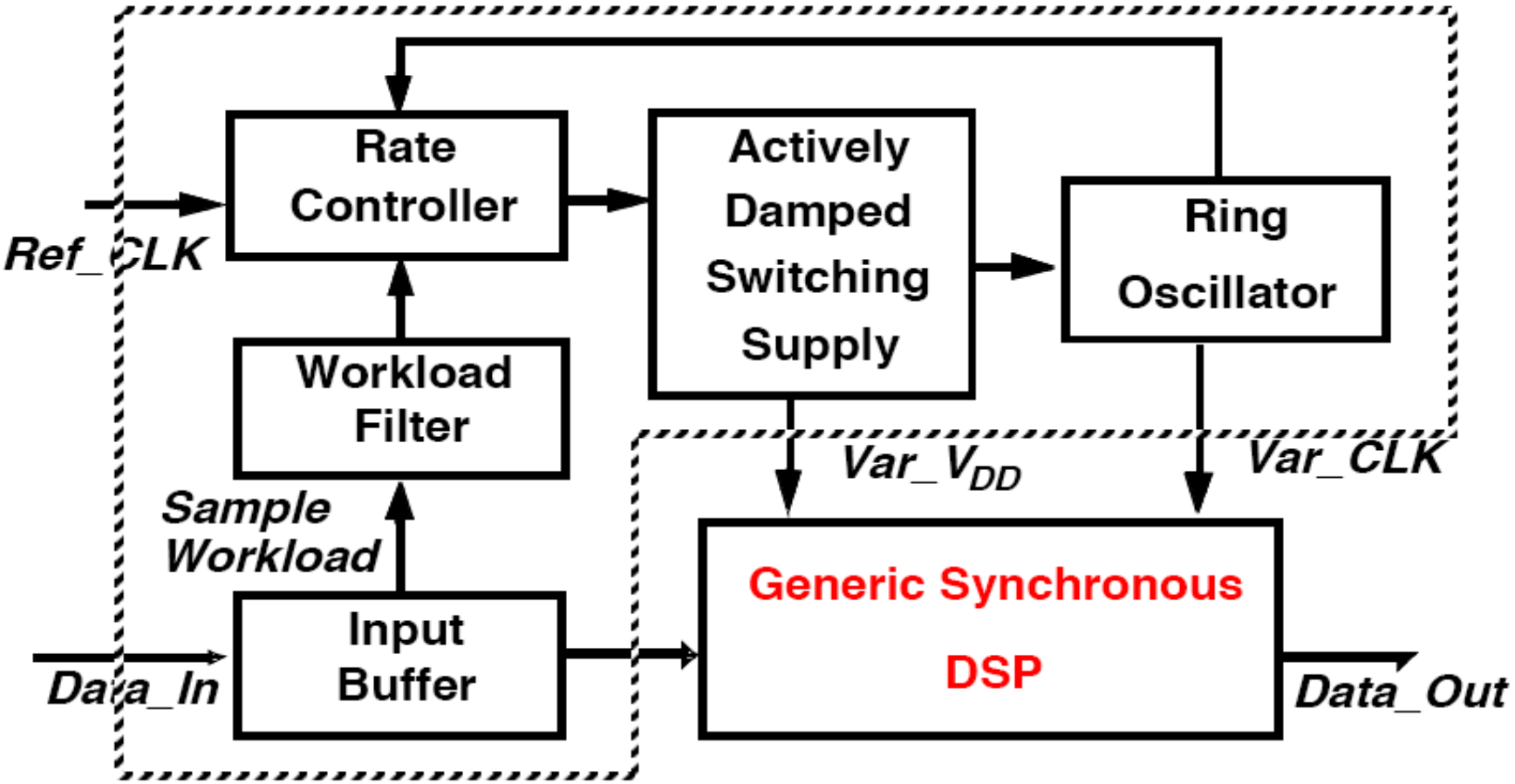
Adaptive Voltage Scaling in Asynchronous Systems



- Exploit data dependent computation time to vary the supply voltage

from [Nielsen94]

Adaptive Voltage Scaling in Synchronous Systems



from Gutnik & Chandrakasan (1996 VLSI Circuits Symposium)

Problem with Workload Averaging Approach

- Can handle average throughput constraints
  - e.g., DSP, general-purpose workstation
- But, can't ensure deadline constraints are met
- Deadlines are often critical
  - protocols, target tracking

- FIFO scheduling with workload averaging is not good for RTOSs!

Power Management Opportunities in RTOS Task Scheduling

- Low processor utilization factor (U)
  - E.g., CNC task set: U = 0.48
  - Result: Long intervals with processor being idle (power wastage)

- Instance to instance task exec. time varies, and is usually much lower than worst case exec. time
  - E.g., JPEG decoding: WCET / BCET = 9.7
  - Reasons: (a) Program control flow depends on inputs, (b) cache effects, and (c) floating point ops have variable latency
  - Result: Additional processor idle intervals (power wastage)

Choice of Power Management Strategy

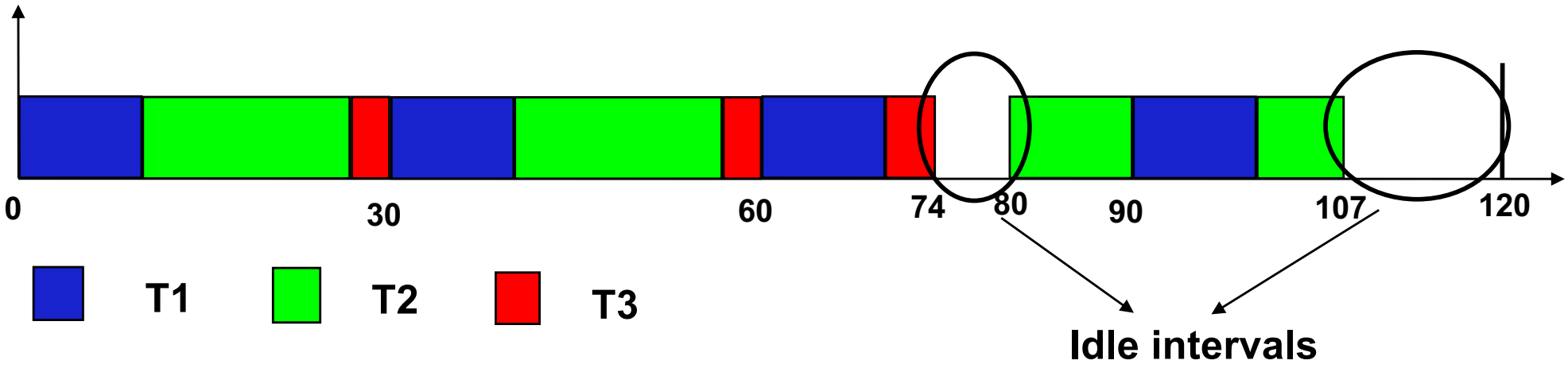
- Shutdown the processor whenever idle
  - Power savings ∝ Shutdown duration
    - Power savings increase linearly with shutdown time

- Slowdown the processor (i.e., reduce clock frequency) and scale the supply voltage (DVS) of the system to just meet instantaneous computational load
  - Power consumption ∝ (Supply voltage)^2
    - Quadratic savings (better than shutdown)
  - Shutdown becomes the secondary strategy, and is used to augment voltage scaling when further DVS is not possible

- Task scheduling problem also becomes a voltage scheduling one

Example

- Consider task set (period, WCET, deadline)
  - { T1(30, 10, 30), T2(40, 17, 40), T3(120, 10, 120)}
- CPU utilization = 10/30 + 17/40 + 10/120 = 84.17%



- Shutdown when idle →15.83% power savings
- Slowdown by 15.83% ?? → NO! Task T2 misses its deadline

How much static slowdown is possible while guarantying satisfaction of timing constraints?

Global Static Slowdown

- Key idea: Slowdown the CPU so that the task set is just schedulable, and any further CPU slowdown will make the taskset non-schedulable

- Option 1: Slowdown till $U > n*(2^{1/n} - 1)$
  - Problem: Very pessimistic. In above example,
  - $U = 0.8417 > 3*(2^{1/3} - 1)$ ➔ No slowdown possible!

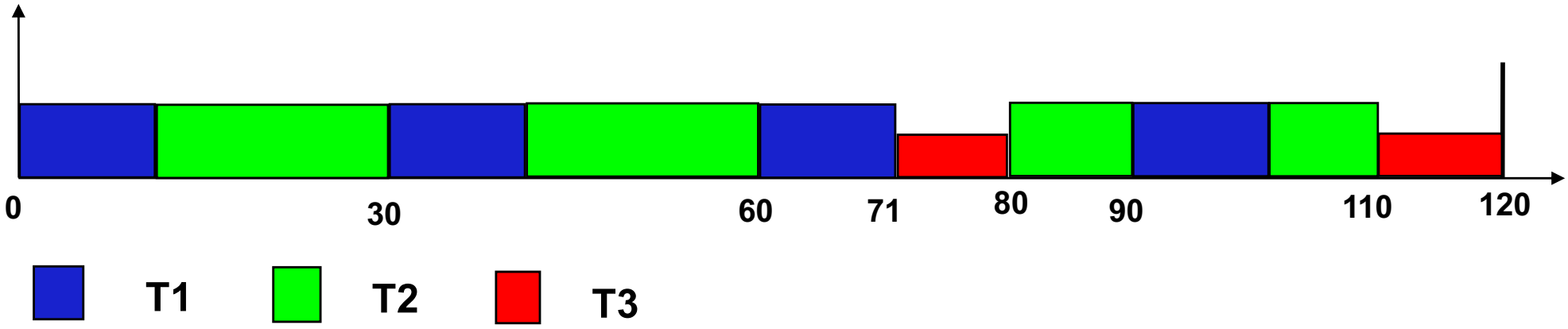
- Option 2: Perform RM analysis (e.g., response time)
  - Use schedulability test to find maximum slowdown S such that task set $\{(T_1, S*C_1), (T_2, S*C_2), \ldots (T_n, S*C_n)\}$ is schedulable
  - Shutdown whenever CPU is idle
  - For above example, S = 30/27 = 1.11 ➔ U = 93.43%
  - Power savings = 27.8%
  - Is this the best we can do? Not really……

Per-task Static Slowdown

- Consider previous example… during static slowdown, T_2 was the bottleneck that decided the global slowdown factor S = 1.11
- Can we slowdown T_2 beyond factor of 1.11?
  - Obviously not, because it is the critical task
- How about T_1?
  - T_1 has higher priority than T_2 ➔ it executes before T_2. Slowing down T_1 also delays T_2
  - Therefore, the answer is NO.
- How about T_3?
  - T_2 executes before T_3. Therefore, slowing down T_3 should not affect response time of T_2. YES!

Per-task Static Slowdown (contd.)

- Key idea: Using a slowdown factor of 1.11 for T_1 and T_2, slowdown T_3 further till it becomes just schedulable
  - Final slowdown factors: $T_1 = 1.11$, $T_2 = 1.11$, $T_3 = 1.9$
  - U = 1.00 (CPU completely utilized!)
  - Power savings: 31.42%
- This is the optimal static solution!



Static Slowdown Algorithm

```
Procedure COMPUTE_STATIC_SLOWDOWN_FACTORS
Inputs: Time_Periods[ ], WCETs[ ], Deadlines[ ]
Outputs: Static_Slowdown_Factors[ ]
{
SET S   //Tasks that can be further slowed down
SET S1
Current_Scaling_Factor = 1;
While (Scalable_tasks != φ) {
 f = Scale_WCETs(Periods[ ], WCETs[ ], Deadlines[ ]);
 S1 = Tasks that will miss deadlines
    with further scaling;
 Current_Scaling_Factor = Current_Scaling_Factor * f;
 For each task i in S
  Static_Slowdown_Factor[i] = Current_Scaling_Factor;
 S = All tasks with priority < Lowest priority
    of tasks in S1;
 }
}
```

Repeat until no more tasks are left to schedule

Compute set of critical tasks (*i.e.*, bottlenecks), and the slowdown factor

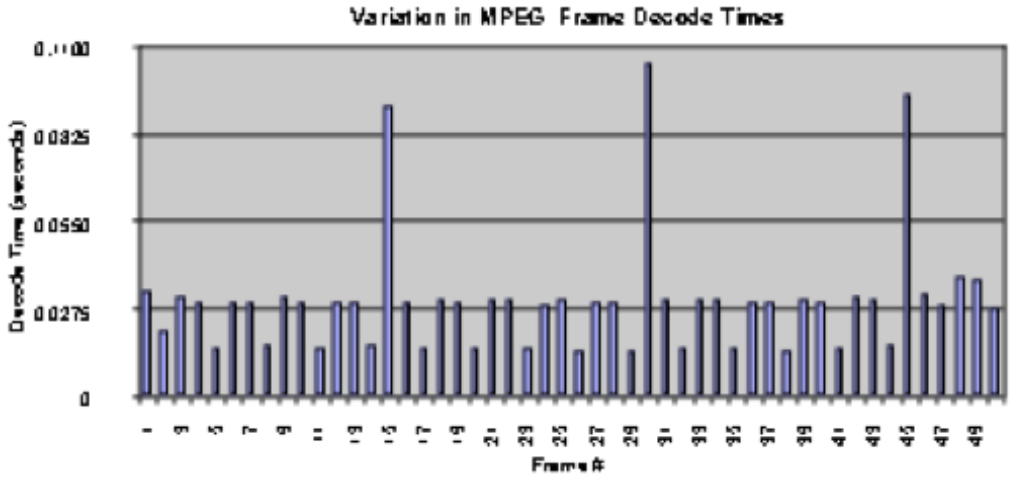
Set the slowdown factors so that task set is just schedule

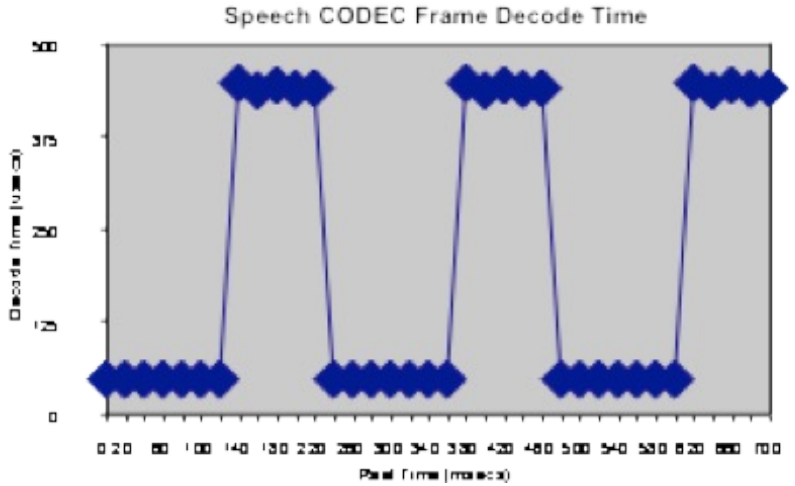
Remove all tasks with priority greater than the lowest priority critical task and continue

Achieving additional energy savings through dynamic slowdown & voltage scaling

Observation #1: Execution-time Variation

- Significant variation in execution time of real-time tasks
  - WCET:BCET often >> 1
  - e.g. on a test run, MPEG decoder time range [0.003s, 0.15s] with average = 0.035s
  - e.g. compressed speech playout task has different time for talkspurt vs. silence
- But, variation is not random due to correlation in underlying signal (speech, sensor etc.)





Observation #2: Applications Tolerant to Deadline Misses

- E.g. sensor networks
- Computation deadline misses lead to data loss
- Packet loss common in wireless links
  - e.g. a wireless link of 1E-4 BER means packet loss rate of 4% for small 50 byte packets
  - radio links in sensor networks often worse
- Significant probability of error in sensor signals
  - noisy sensor channels
- Applications designed to tolerate noisy/bad data by exploiting spatio-temporal redundancy
  - high transient losses acceptable if localized in time or space

- *If the communication is noisy, and applications are loss tolerant, is it worthwhile to strive for perfect noise-free computing?*

Exploiting Execution-time Variation and Tolerance to Deadlines

- Idea: predict execution time of task instance and dynamically scale voltage so as to minimize shutdown
- Execution time prediction
  - learn distribution of execution times (pdf)
  - Tasks with distinct modes can help the OS by providing hint after starting
    - E.g. MPEG decode can tell the OS after learning whether the frame is P, I, or F
- But, some deadlines are missed!
- Adaptive control loop to keep missed deadlines < limit
- Provides adaptive power-fidelity trade-off

Dynamic Slowdown Algorithm

```
Procedure COMPUTE_DYNAMIC_SLOWDOWN_FACTOR
Inputs: WCET, Deadline_Miss_History,
        Execution_History
Outputs: Dynamic_Slowdown_Factor
{
P = PREDICT_TIME(Execution_History);
α = ADAPTIVE_FACTOR(Deadline_Miss_History, α);
P = P * α;
Dynamic_Slow_Factor = WCET / P;
}
```

Set dynamic slowdown factor by stretching the adaptive predicted completion time to the worst-case completion time.

```
Procedure PREDICT_TIME
Input: Execution_History
Output: Predicted_time
{
If (Elapsed_time_for_task_instance = 0)
  Predicted_time = Σ_1^N (Runtime of kth task instance) / N ;
  //N is the number of past instances being monitored
Else
  Predicted_time = Expected value of execution time
       distribution from Elapsed_Time to WCET;
}
```

Predict the execution time of next instance based on past history of execution times

```
Procedure ADAPTIVE_FACTOR
Input: Deadline_Miss_History, Adaptive_Factor
Output: Adaptive_Factor
{
x = Number of deadline misses in past M instances;
If (x ≥ T1) Adaptive_Factor + = I;
If (x ≤ T2) Adaptive_Factor - = D;
}
```

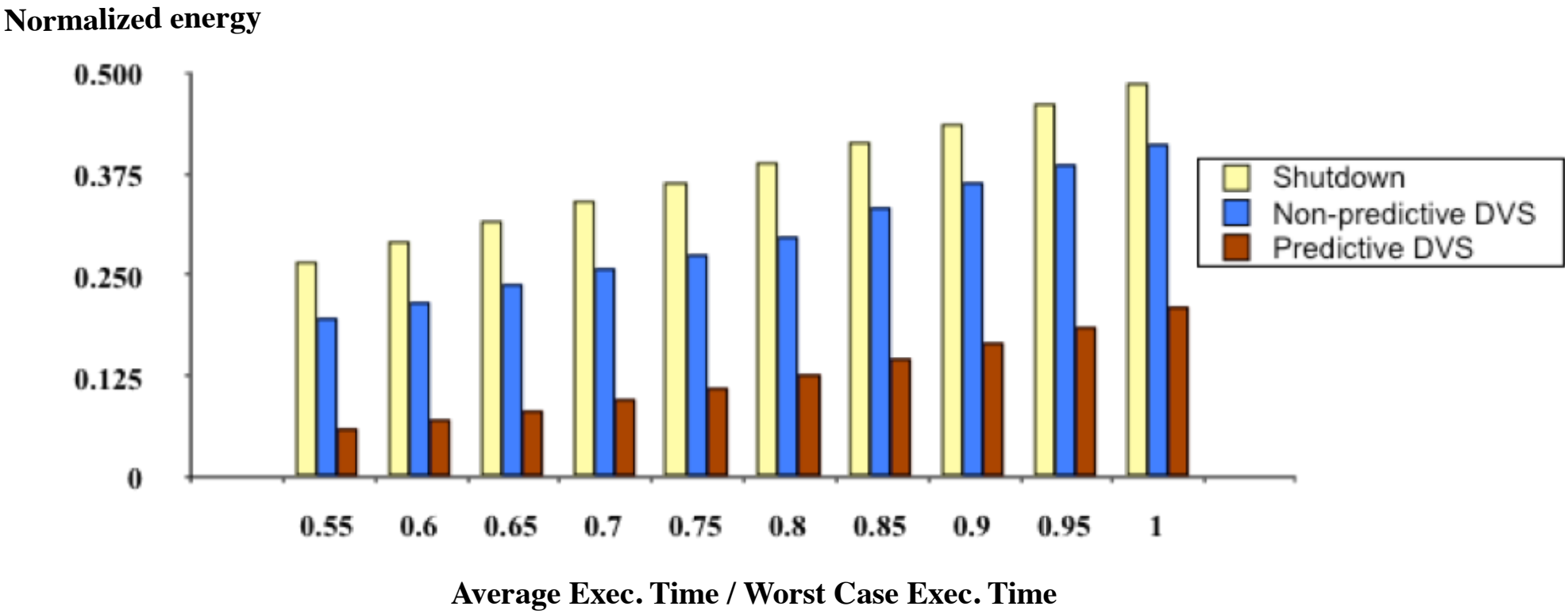
Adaptive control mechanism to vary the aggressiveness of the prediction scheme. More deadline misses cause prediction to turn conservative.

Dynamic Slowdown

- Consider the previous example: If first instances of T1 and T2 have an execution time of 0.5 times their respective WCETs
  - Power savings: 46.76%
- Further enhancement: Monitor if currently executing task is the only request in the system. If so, slowdown till it reaches arrival point of next request, or its deadline, whichever is earlier.

Scheme	Savings
No power management	0%
Shutdown based	15.83%
Global static slowdown	27.8%
Per-task static slowdown	31.42%
Dynamic slowdown	46.76%

Performance of Predictive DVS for Adaptive Power-Fidelity Tradeoff



- Result: up to 75% reduction in energy over worst case based voltage scheduling with negligible loss in fidelity (up to 4% deadline misses) on variety of multimedia and signal processing tasks

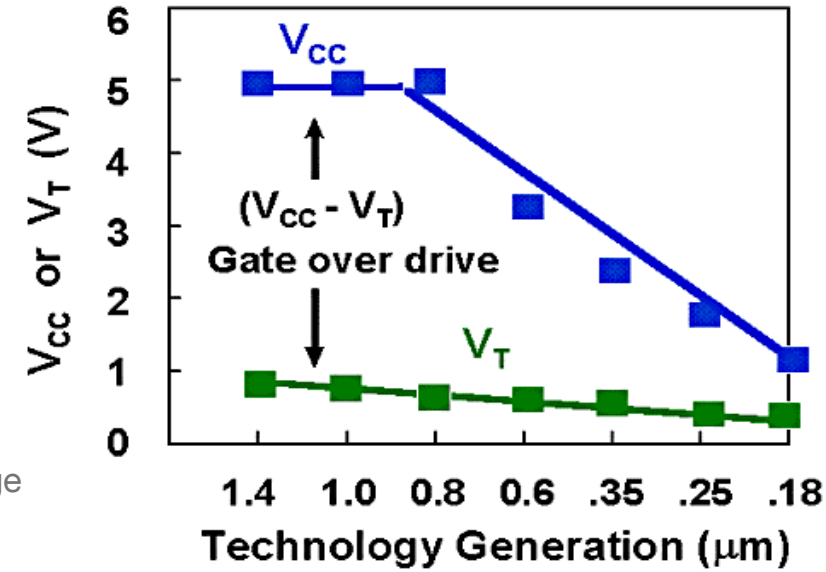
Extensions to EDF

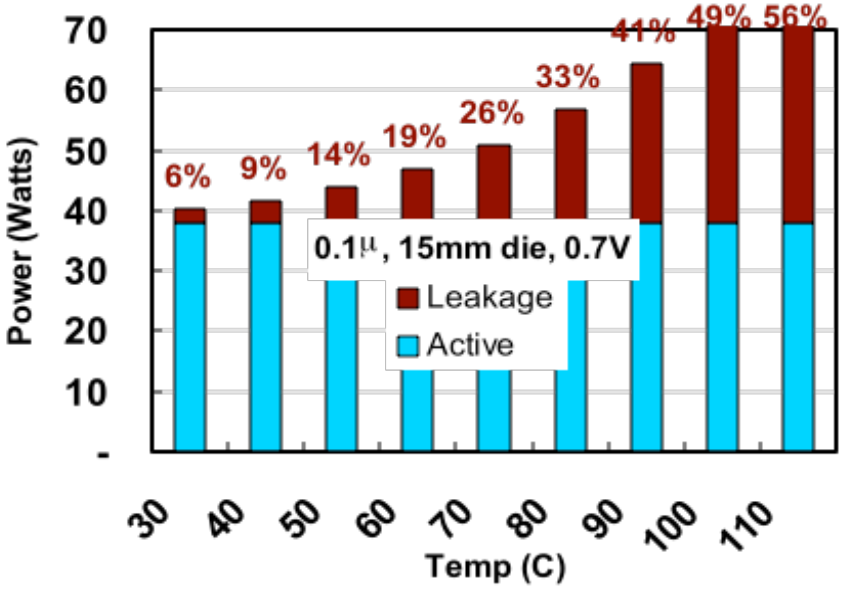
- Assuming $D_i = T_i$, the per-task static slowdown algorithm reduces to the global static slowdown algorithm.

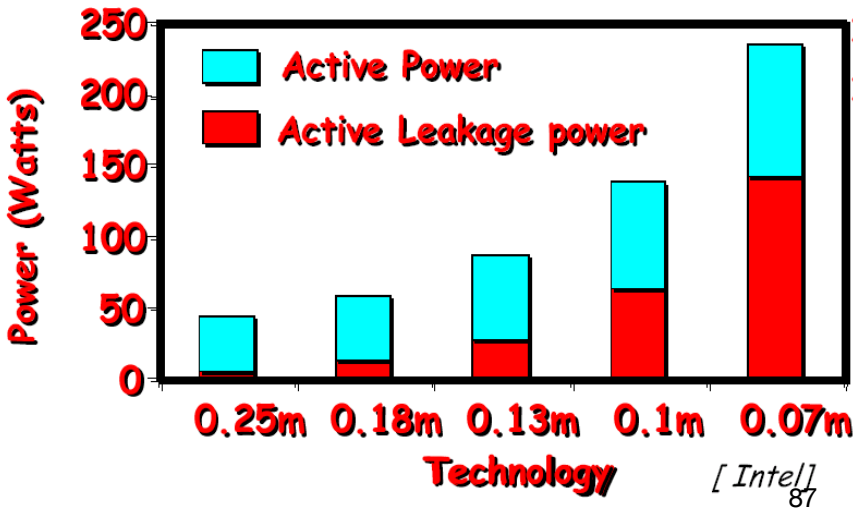
- Dynamic slowdown algorithm does not change

Leakage - the DVS Show-stopper

- Scaling improves
  - transistor density and functionality on a chip
  - speed and frequency of operation ➔ higher performance
- Scaling and power
  - active power ↑ as CV_{DD}^2f
    - scale V_{DD}
    - scale $V_{th} \Rightarrow I_{leak}$ ↑
  - standby or leakage power ↑ as $V_{DD}I_{leak}$
- P_{leak} catching up with P_{active} in nano-scale circuits
  - Subthreshold leakage
  - Gate leakage
  - Reverse-biased Junction Band-to-band-tunneling (BTBT) leakage
- Unfortunately, leakage energy ↑ if the speed ↓

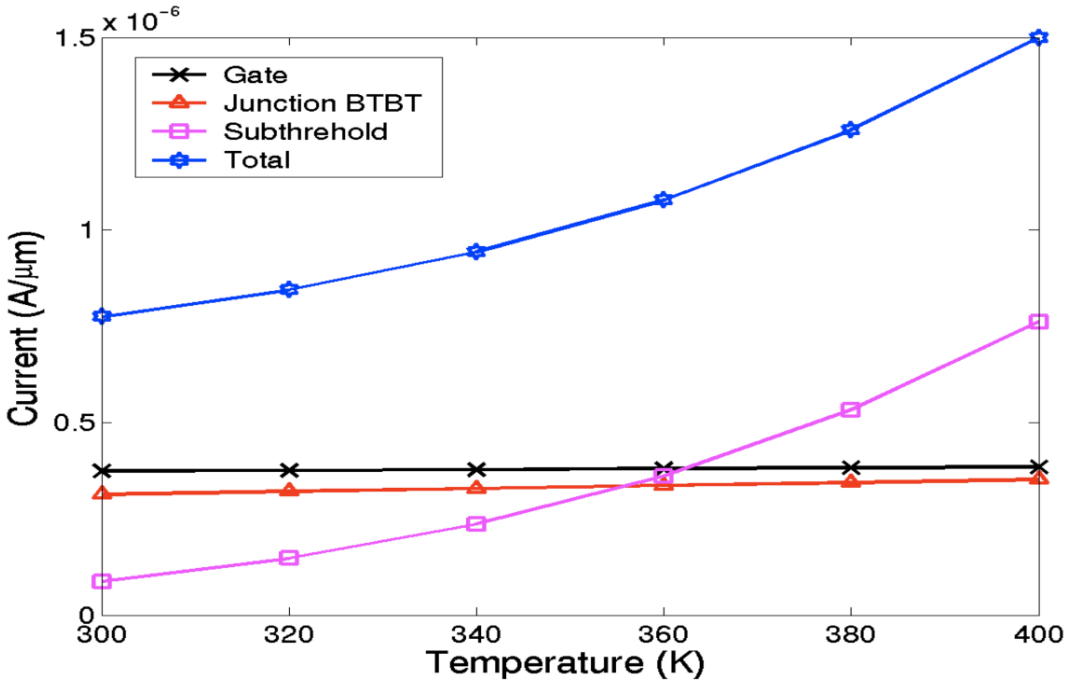






Leakage Behavior

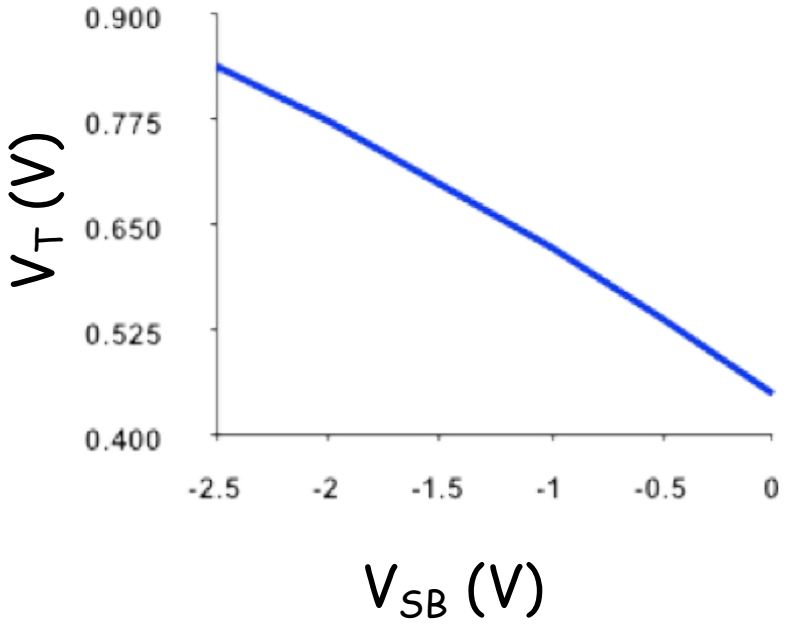
- Temperature dependence
  - Gate leakage dominates at low temperatures
  - Subthrehold leakage increases exponentially with temperature and dominates at higher temperature
- Subthreshold leakage increases exponentially with reduction in threshold voltage V_{th}
  - V_{th} can be controlled by body bias voltage V_{bs}
  - But V_{th} also effects speed
- Suggests two techniques
  - Dynamic V_{th} scaling via Adaptive Body Bias
  - Thermal management



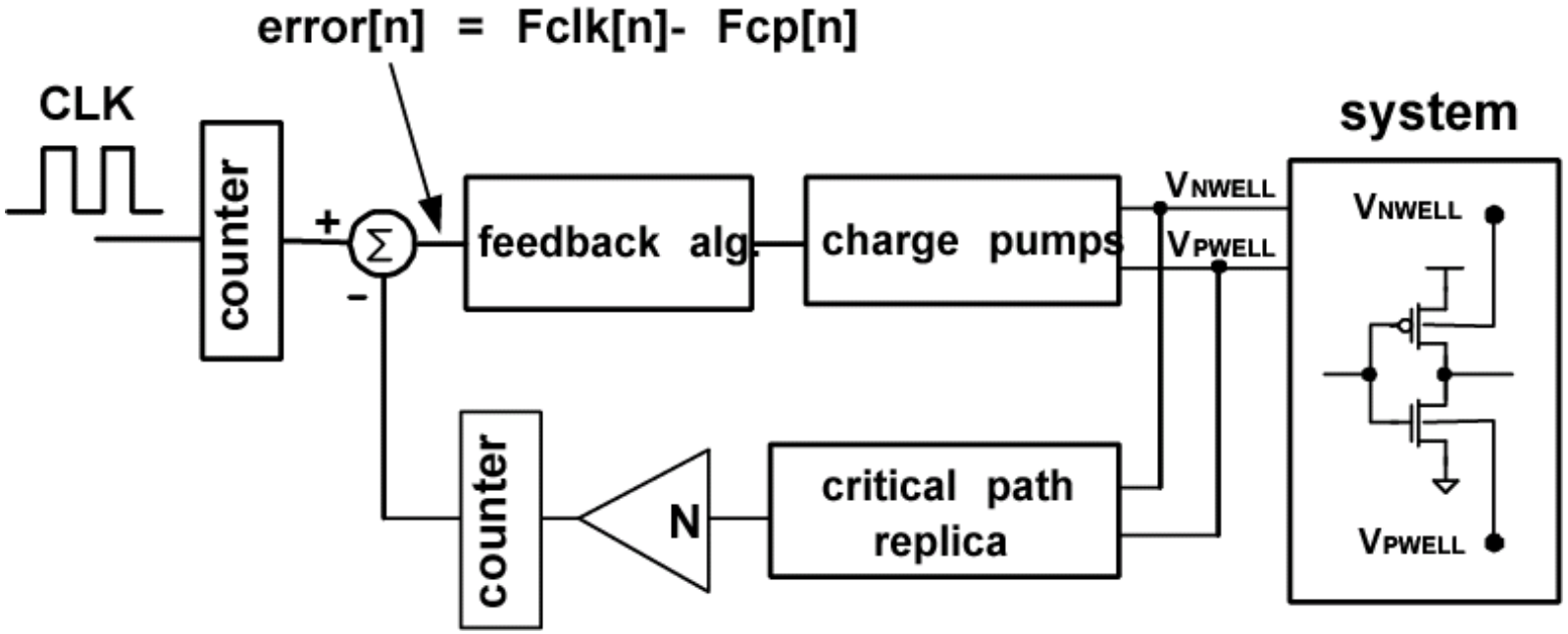
Variable V_T via Adaptive Body Bias

- $V_T = V_{T0} + \gamma(\sqrt{|-2\phi_F + V_{SB}|} - \sqrt{|-2\phi_F|})$

- For n-channel device, substrate is normally tied to ground ($V_{SB} = 0$)
- Adjusting substrate bias at run time is called adaptive body-biasing (ABB)
- Standby mode: Reverse bias
  - $V_{SB} < 0$ causes V_T to increase
  - Low leakage
- Active mode: Forward bias
  - $V_{SB} > 0$ causes V_T to decrease
  - Higher current drive



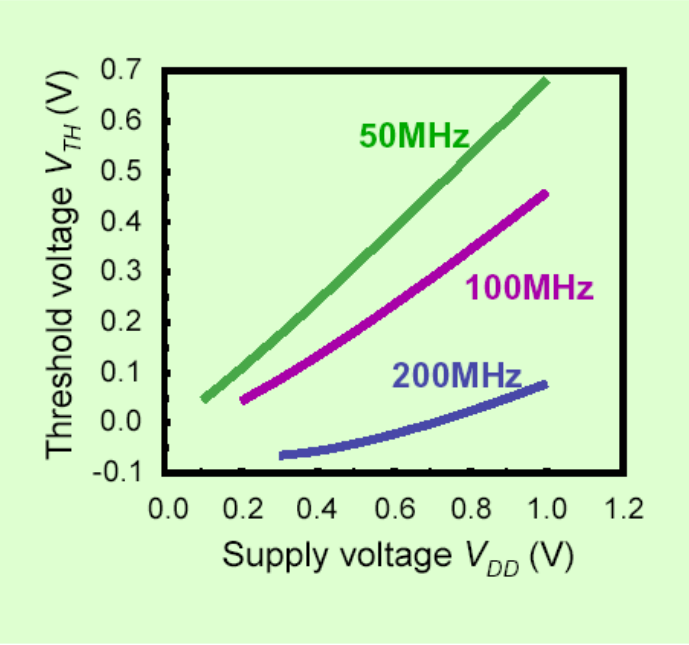
Controlling Leakage via Dynamic Threshold Voltage Scaling



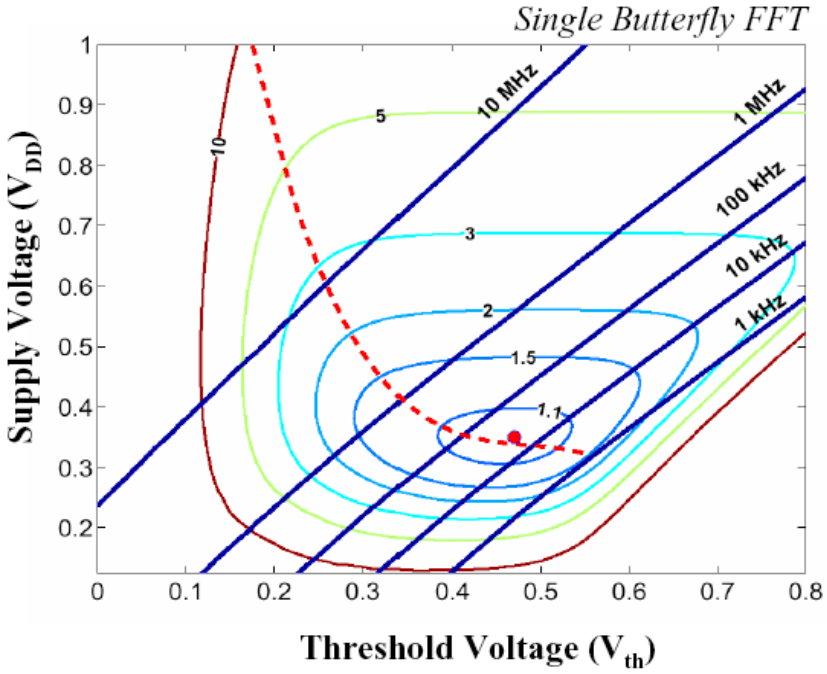
- Use critical path replica circuit to judge performance demand and adjust V_T accordingly
- High performance demand: Low V_T (no body bias)
- Low performance demand: Decrease clock frequency, use ABB to increase V_T

Joint Dynamic VDD and VT Scaling

Performance increases with V_{DD}, decreases with V_{TH}

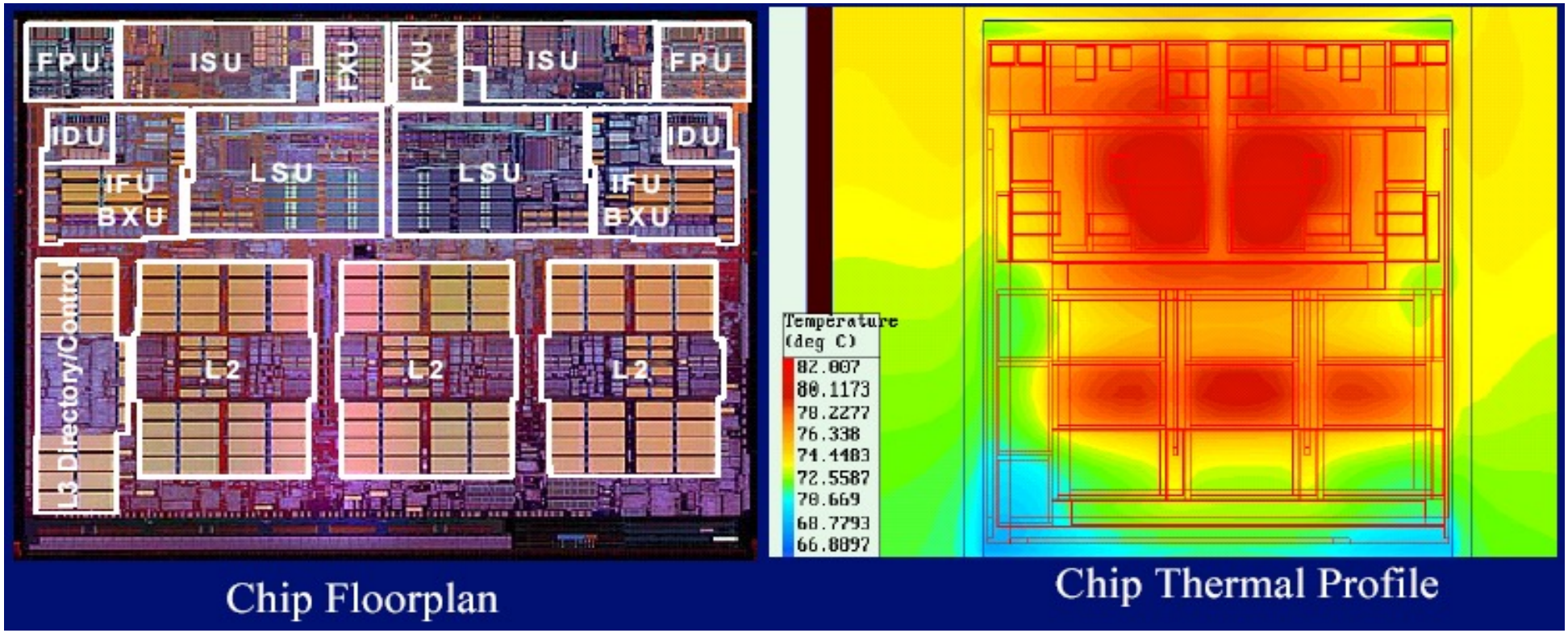


Optimal (V_{DD}, V_{TH}) exists for given frequency



Energy *vs.* Latency in Two Dimensions

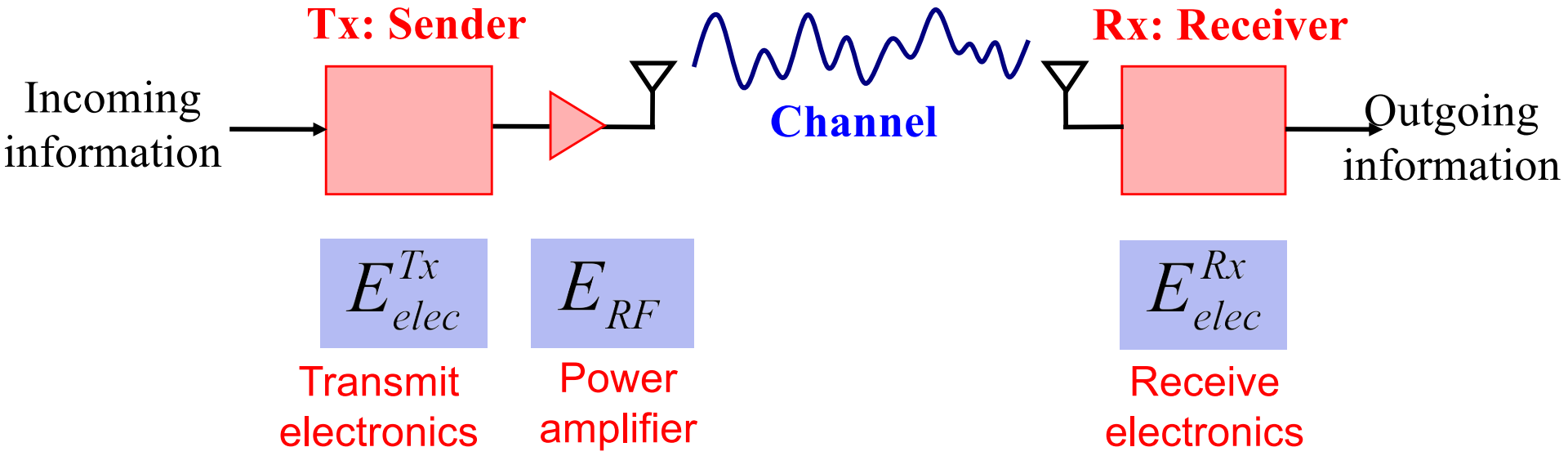
Dynamic Thermal Management



- Exponential dependence between leakage and T
- Dynamic thermal management
- Improves leakage, performance, and reliability

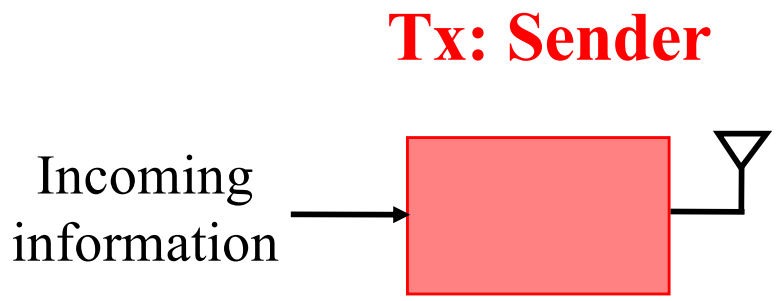
Dynamic Power Management of Radios

Energy in Radio: Recap….



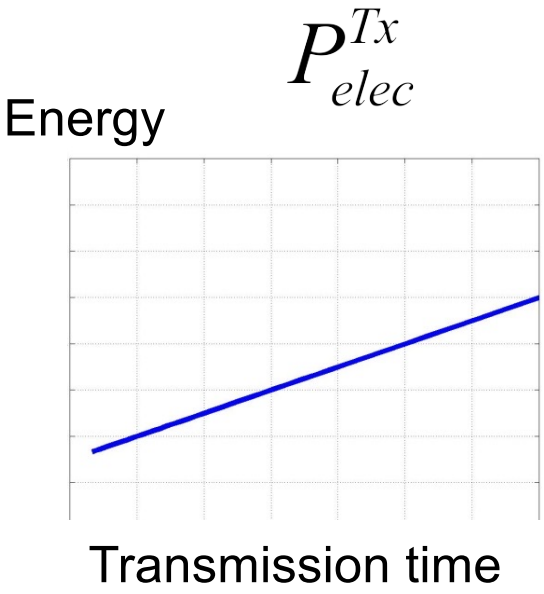
- Wireless communication subsystem consists of three components with substantially different characteristics
- Their relative importance depends on the transmission power of the radio

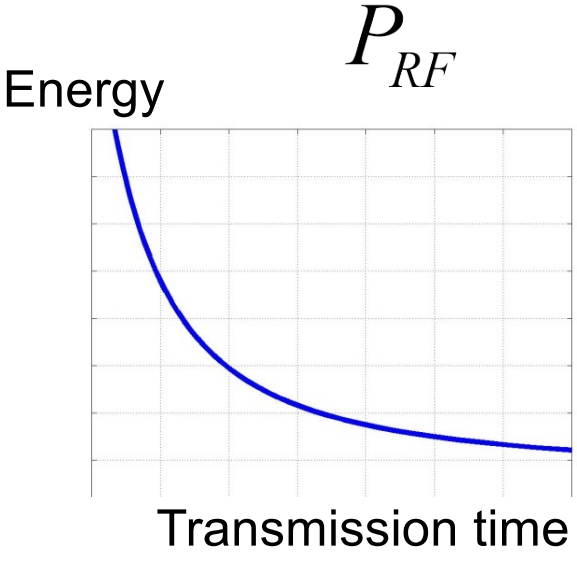
Energy Consumption of the Sender

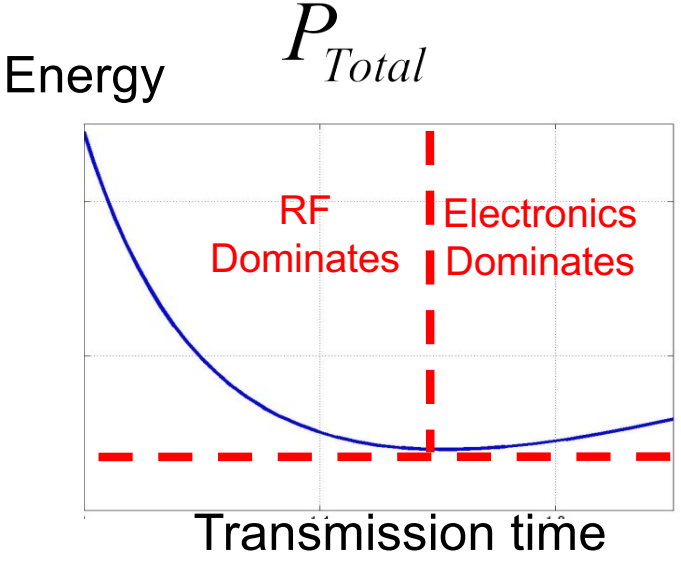


Parameter of interest:
energy consumption per bit

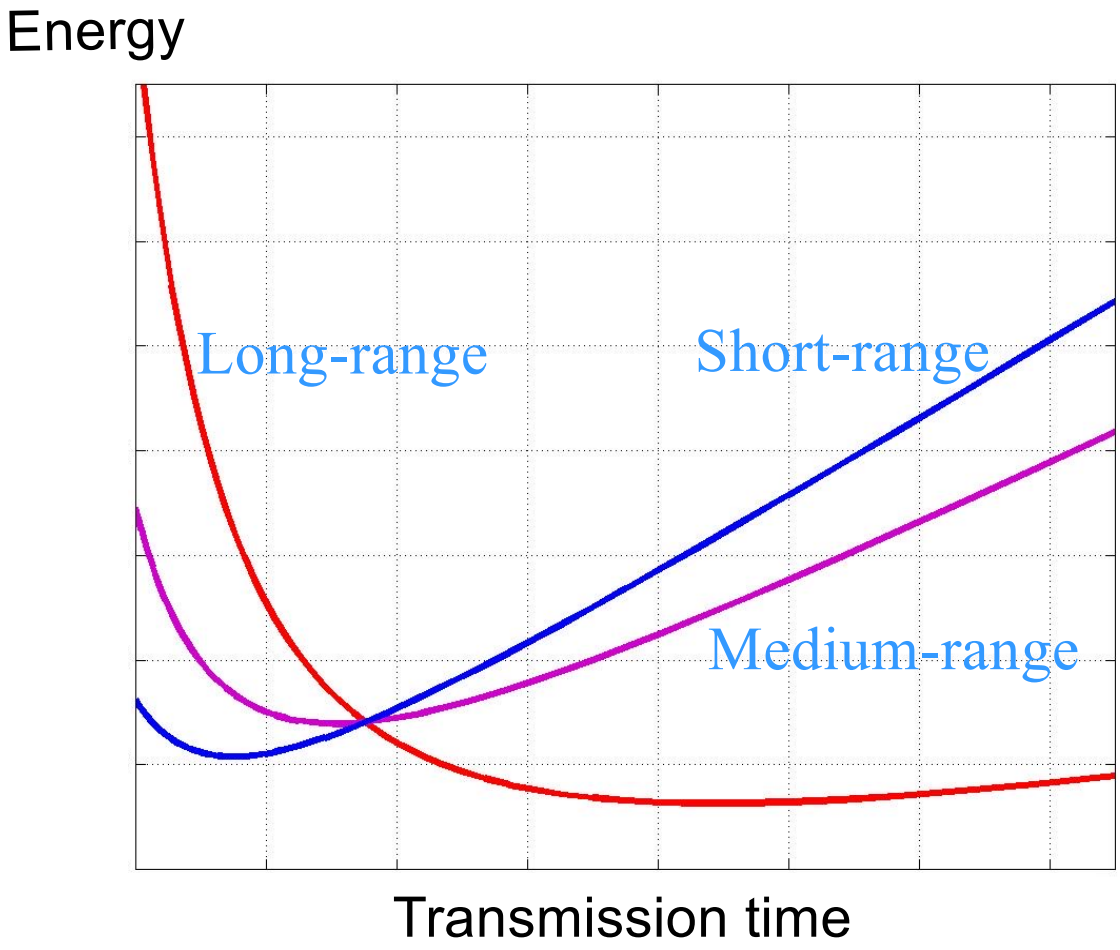
$$E_{bit} = \frac{P}{T_{bit}}$$





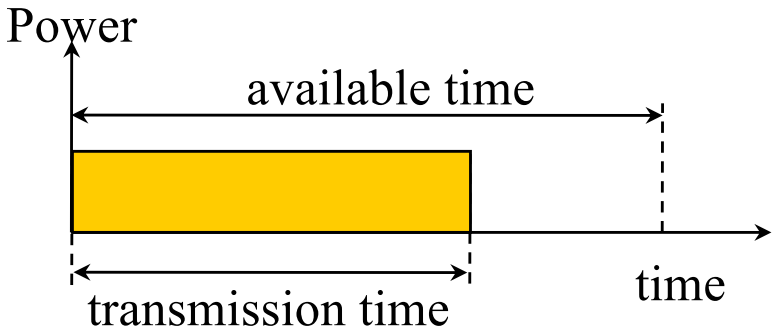


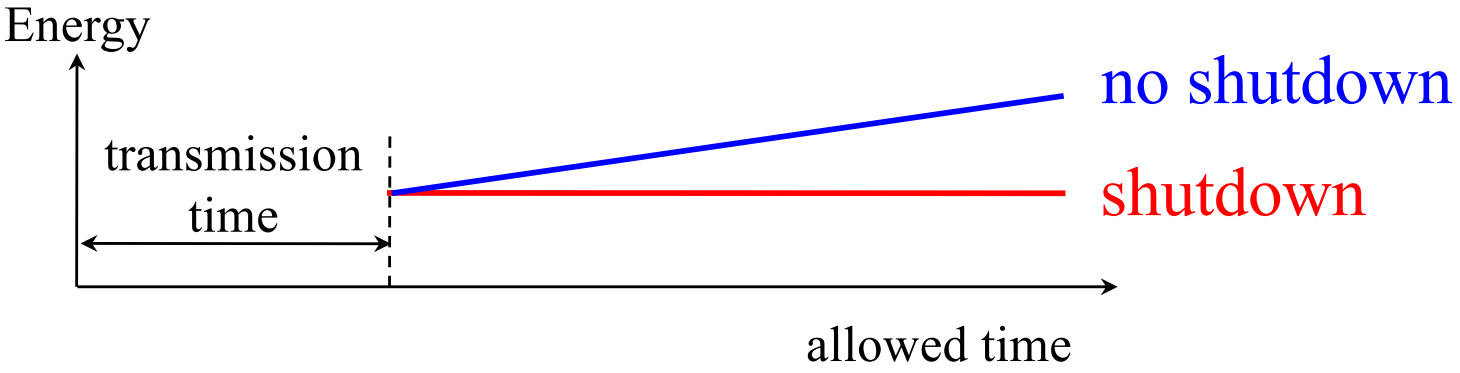
Effect of Transmission Range



Radio Energy Management #1: Shutdown

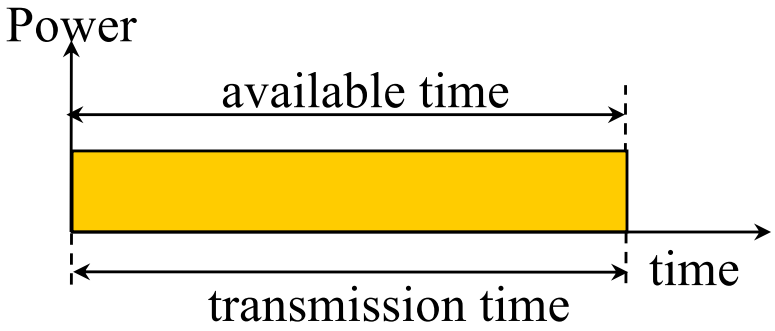
- Principle
  - Operate at a fixed speed and power level
  - Shut down the radio after the transmission
  - No superfluous energy consumption
- Gotcha
  - When and how to wake up?
  - Paging radio, wakeup protocol





Radio Energy Management #2: Scaling along the Performance-Energy Curve

- Principle
  - Vary radio 'control knobs' such as modulation and error coding
  - Trade off energy versus transmission time

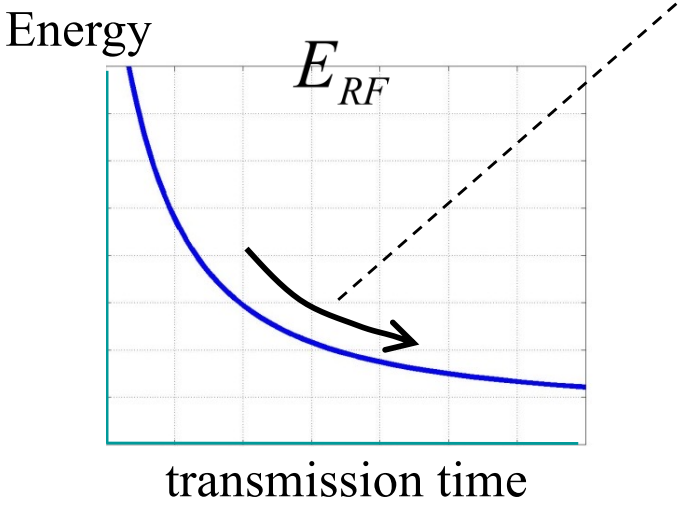


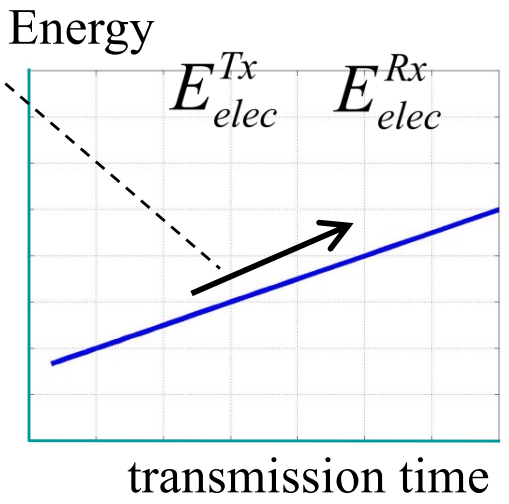
Modulation scaling

fewer bits per symbol

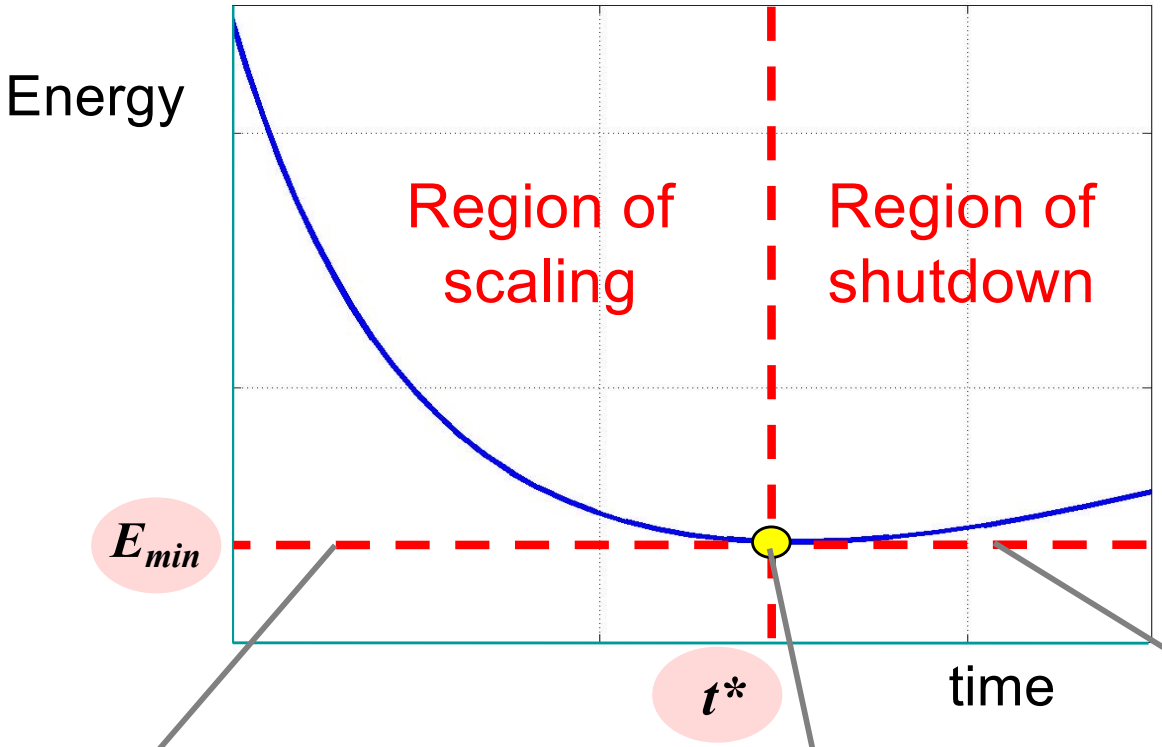
Code scaling

more heavily coded

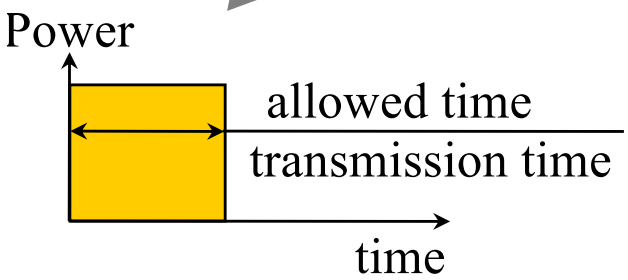


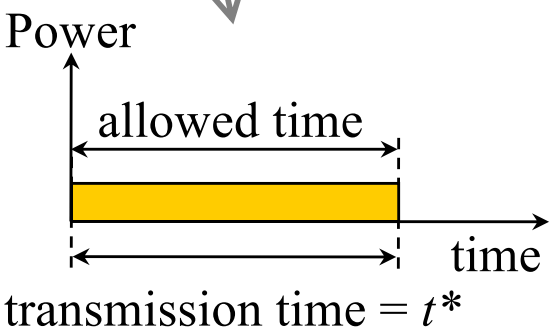


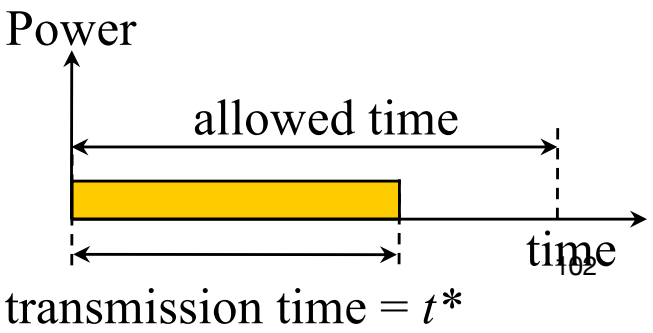
Scaling vs. Shutdown



- Use scaling while it reduces the energy
- If more time is allowed, scale down to the minimum energy point and subsequently use shutdown

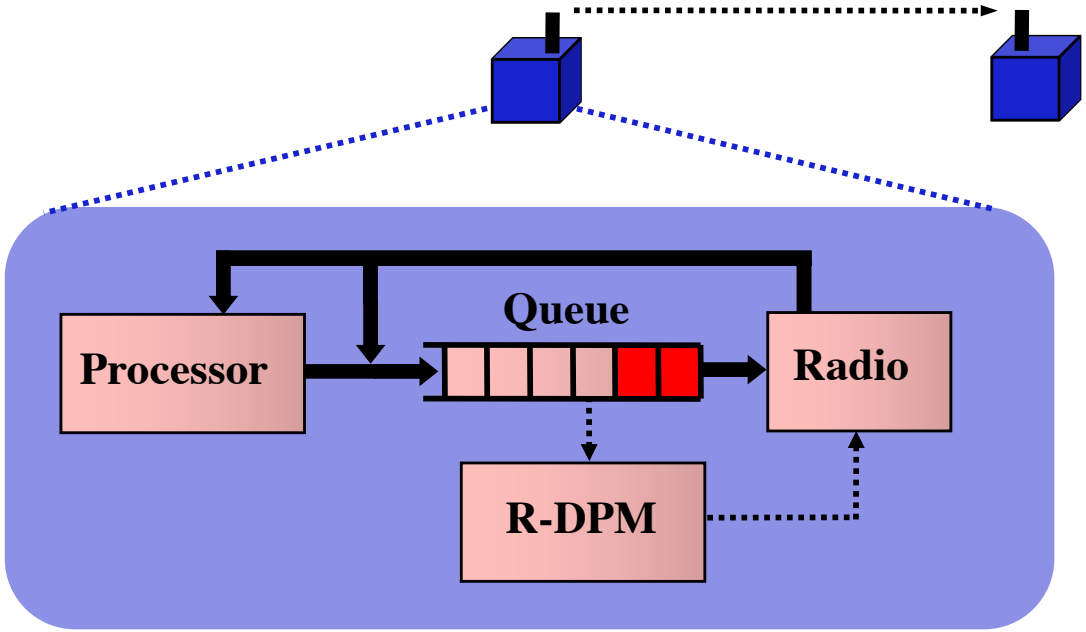


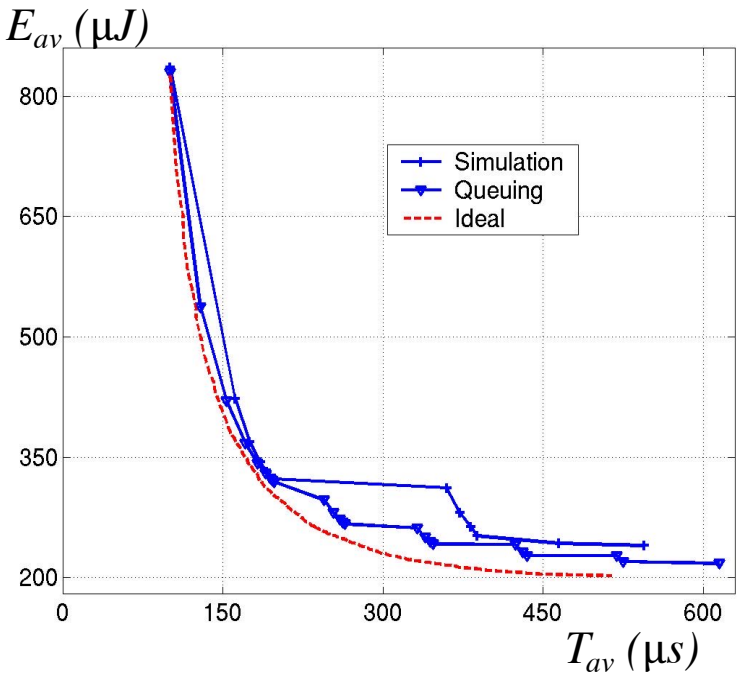




Exploiting DMS for Energy-aware Packet Scheduling

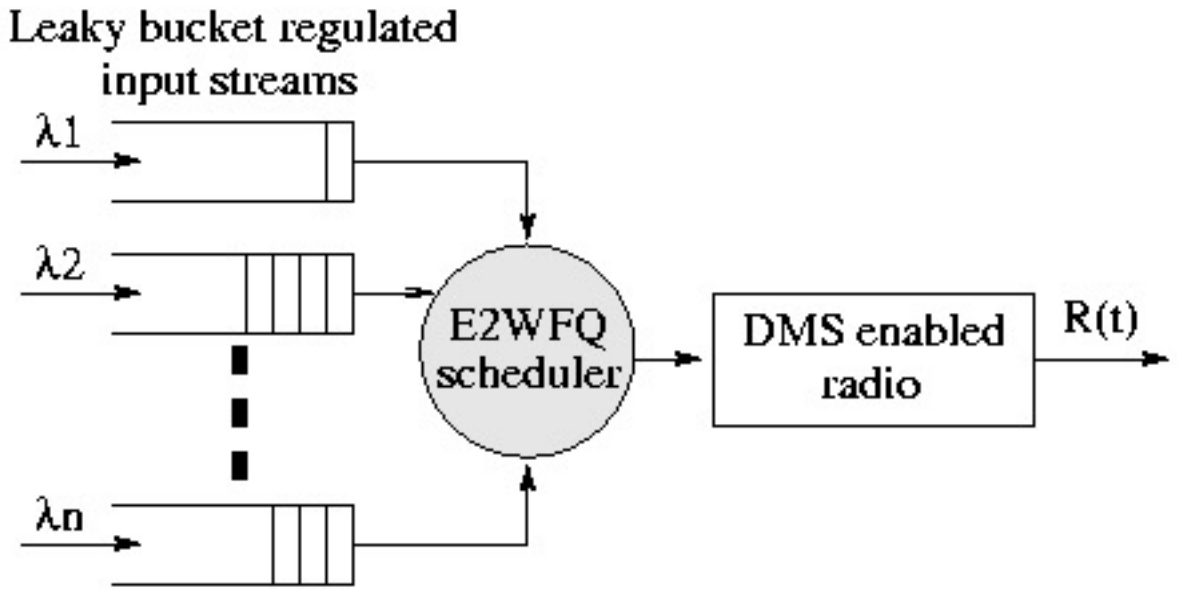
- Example: adapt modulation setting based on # of packets in the queue
- Different {queue, b} settings results in different point on the energy-delay curve





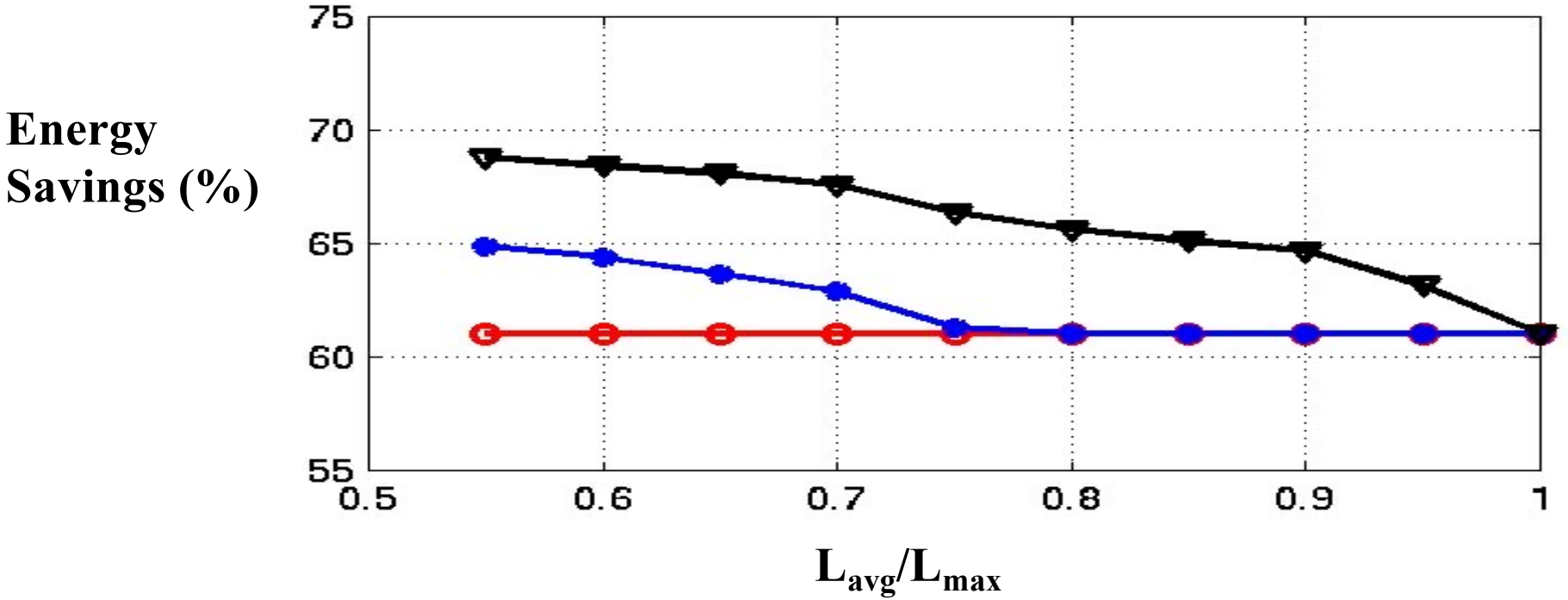
E^2WFQ : Combining DMS & WFQ [Raghunathan @ ISLPED-2002]

- Intuitively, what does fairness mean?
  - Each connection gets no more than what it wants (subject to a max.)
  - The excess, if any, is equally shared
- In E^2WFQ, excess is not distributed ➔ Used to save energy instead
- Energy saving opportunities
  - Average input rate of a stream is lower then guaranteed rate
  - Packet lengths may be variable (often shorter than worst case)
  - Low, time varying link utilization



Energy-aware Real-time Packet Scheduling

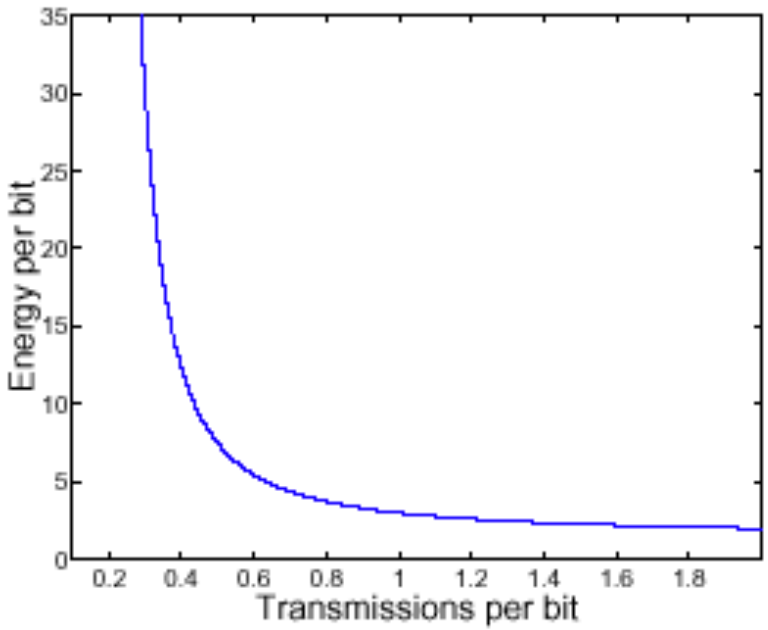
- Analogous to RTOS task scheduling
- Exploit variation in packet length to perform aggressive DMS



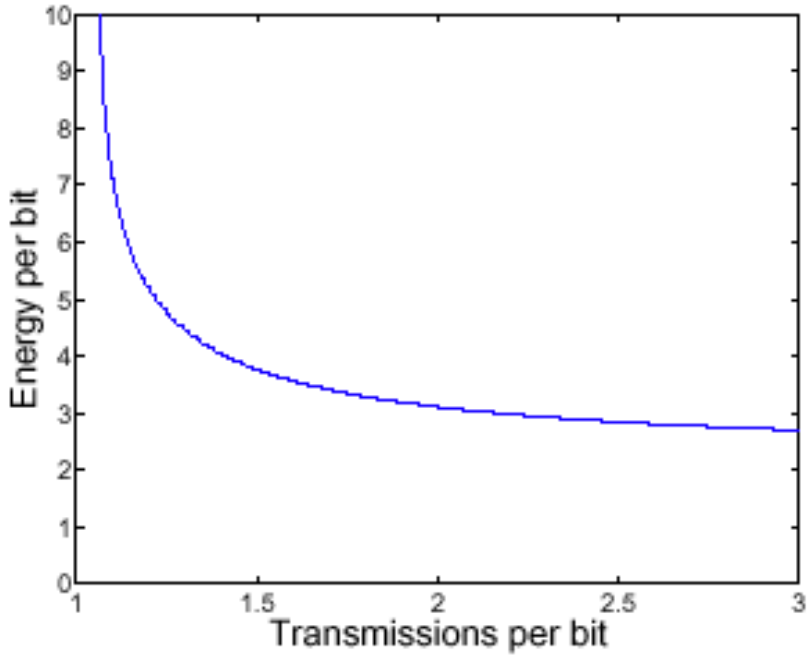
- Up to 69% reduction in transmission energy (source: work @ UCLA)

Another Energy vs. Delay Knob in Radios: Coding

Optimal coding



Antipodal signaling, block FEC

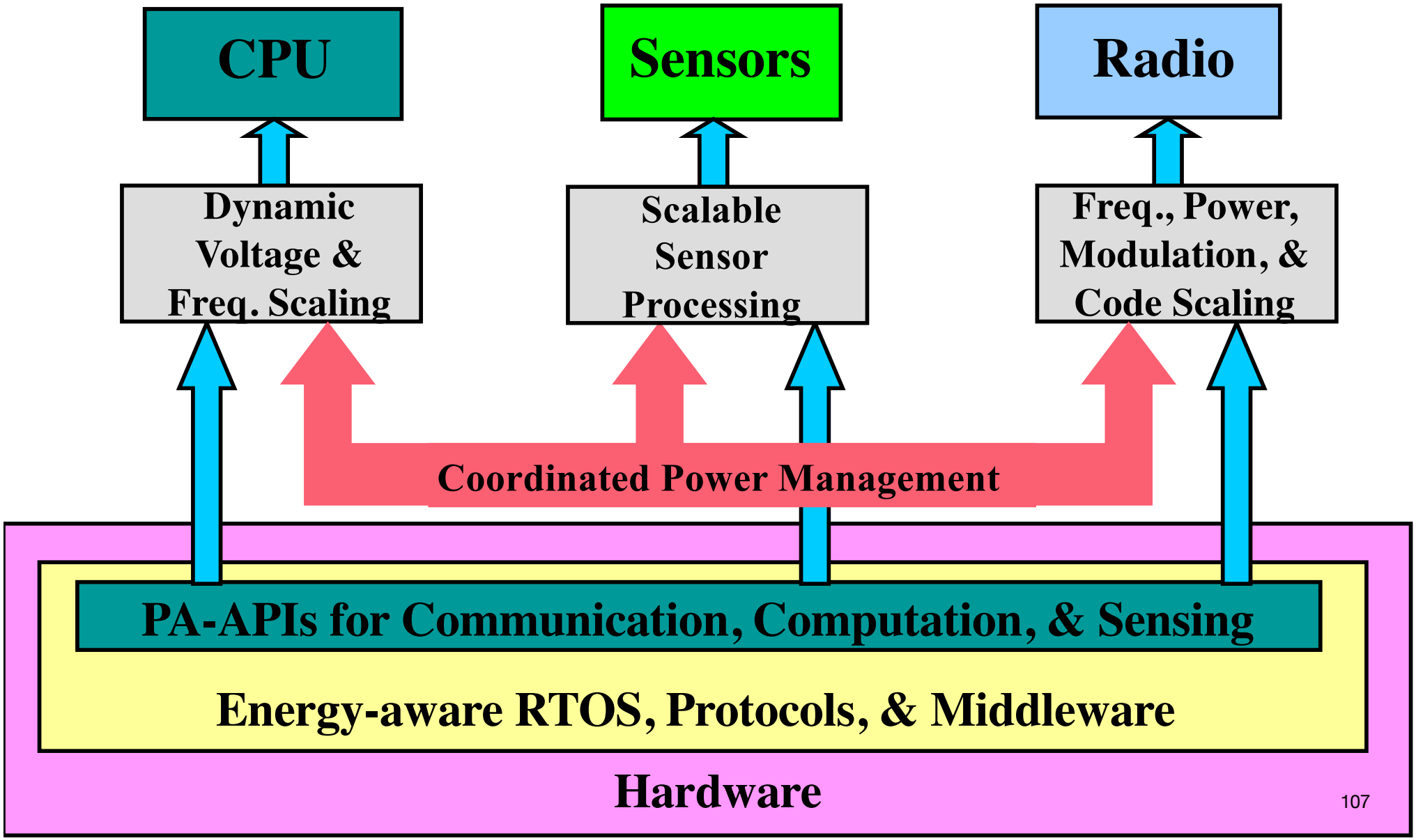


$$\mathcal{E}_b = sN(2^{\frac{2}{s}} - 1) \text{ Joule/bit}$$

s: transmissions/bit N: noise variance

Source: [Prabhakar01] @ Infocom 2001

Putting it All Together: Power-aware Sensor Node

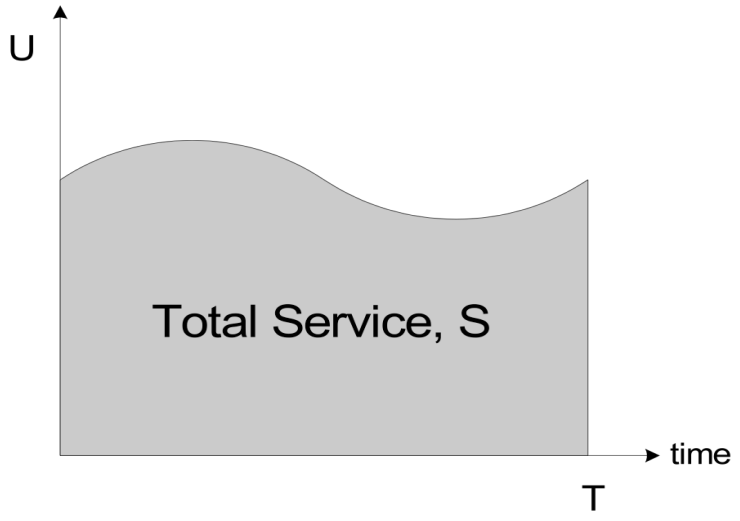


Battery and Harvesting Aware Power Management

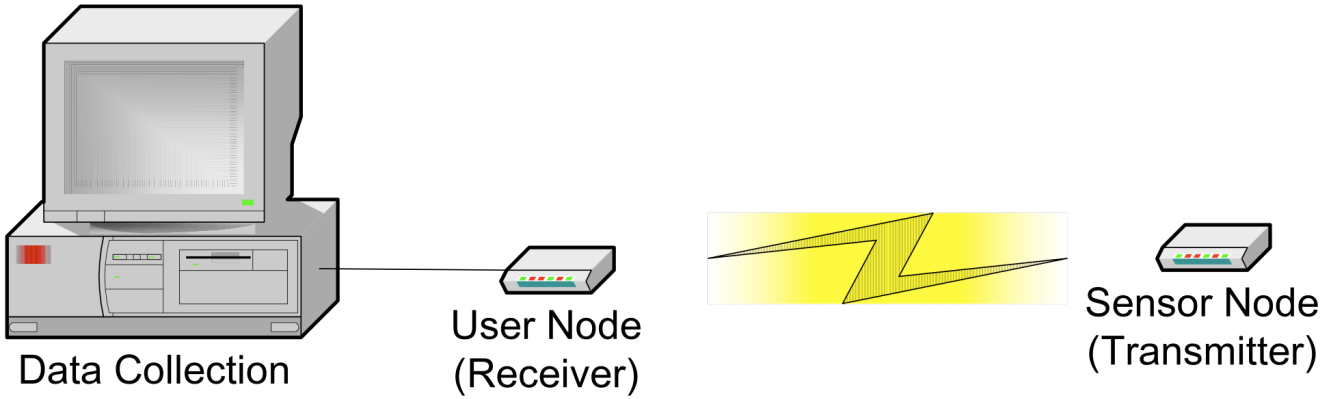
Battery-aware Operation

- Ideal battery: buckets of constant energy, constant voltage
- Real battery: rate capacity effect, relaxation effect, voltage drops etc.
- Due to battery properties, the ways at which the batteries are discharged have impact on the actual lifetime
- Problem: optimize lifetime for real batteries
  - Utility-based approach: find operation profile U(t) to maximize total service
    - Static Approach: Find constant k such that U(t)=k that maximize S
    - Dynamic Approach: U(t) varies with time

$$S_{APP} = \int_{t=0}^{T} U(t)dt$$



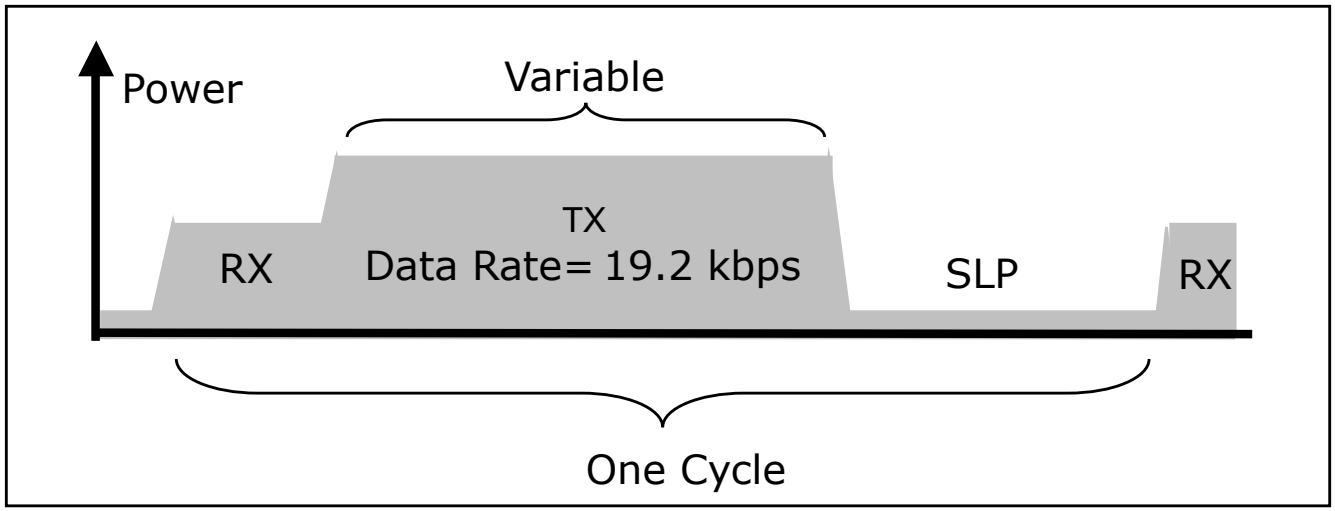
Example Scenario



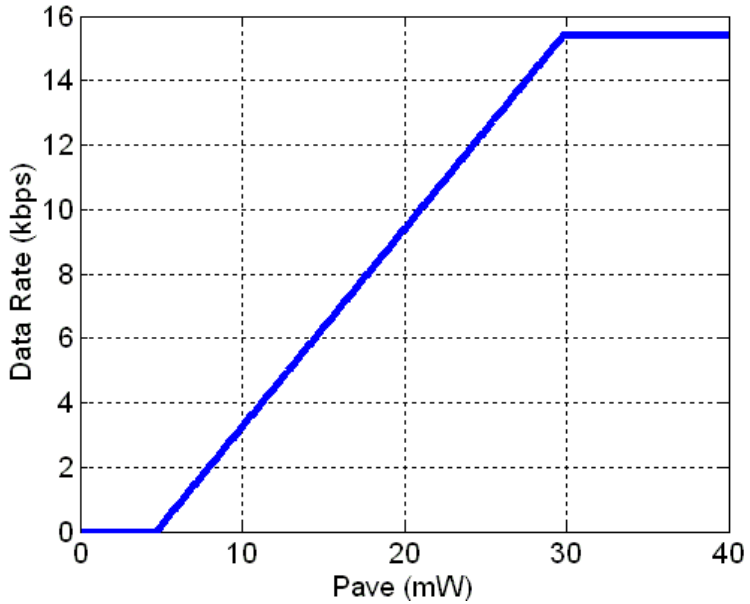
- **Goal:** Needs to transfer as much sensor data as possible with given battery capacity
- **Resource:** Communication
- **Protocol:** similar to 802.11 Ad-hoc Power Savings Mode

- **Simulation Setup:**
- Power Number: Based on Measurement of Medusa (Mote)
- Battery Simulator: Battery Model from John Newman's Group at UC Berkeley Chemistry Dept. exhibits both rate capacity effect and relaxation effect

Radio Throughput vs. Power

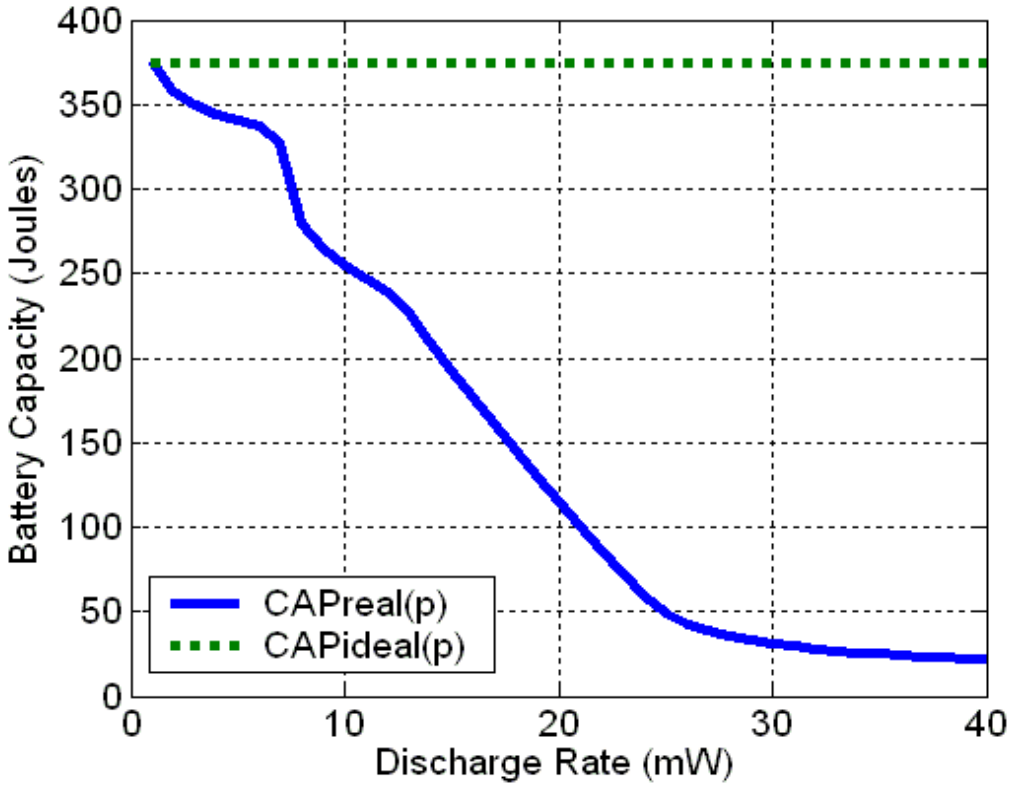


Radio BW R_{APP}:
Effective Data rate

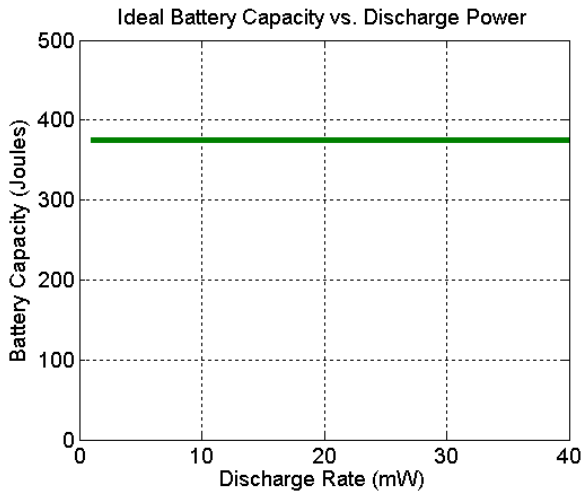


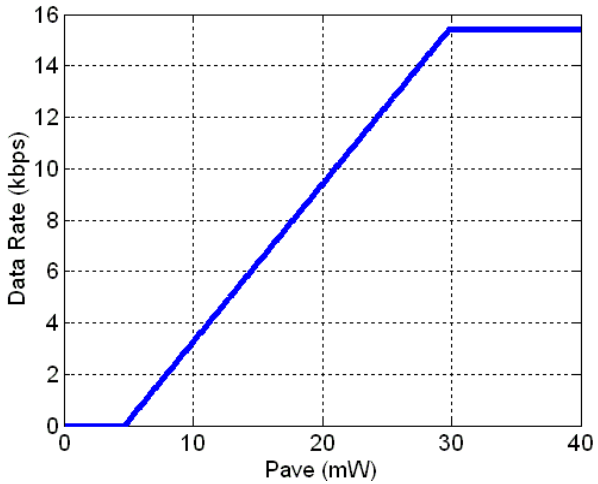
Battery Capacity Curve

- Battery model shows the Rate Capacity Effect

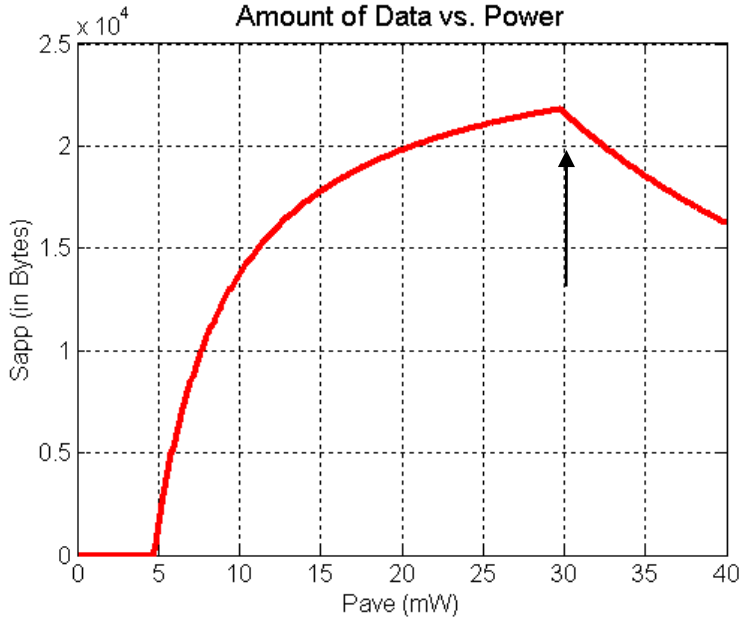


Non Battery Aware Approach

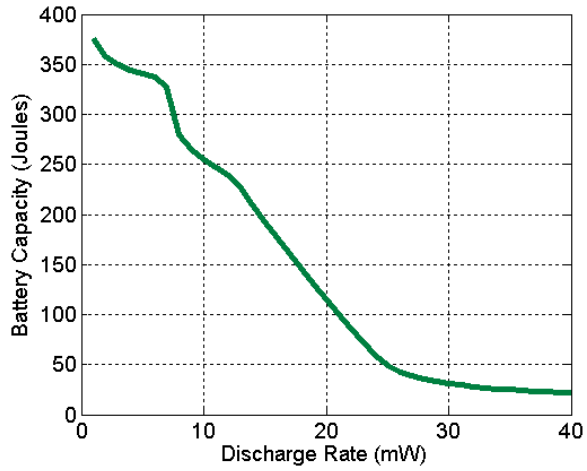


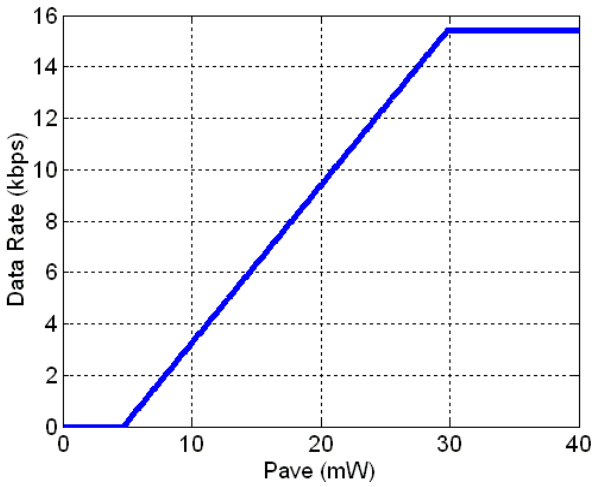


$T=CAP_{ideal}/P$



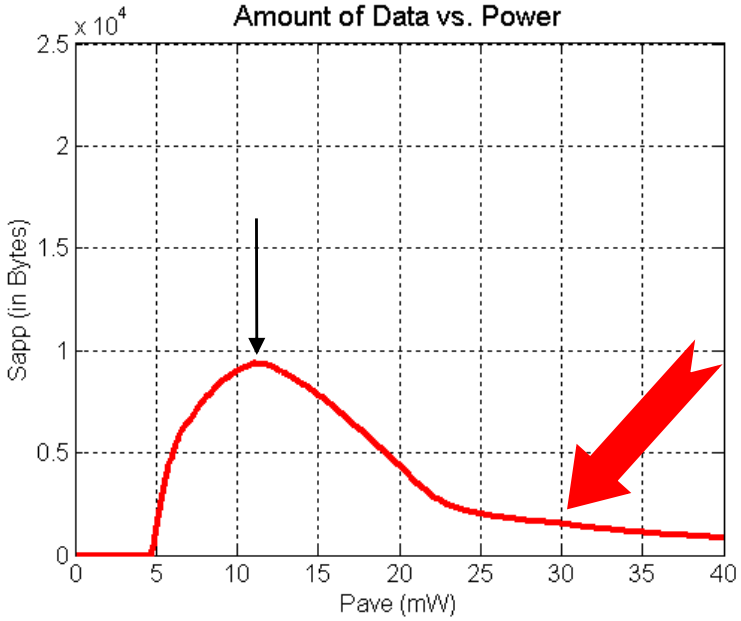
Static Battery State Aware Approach





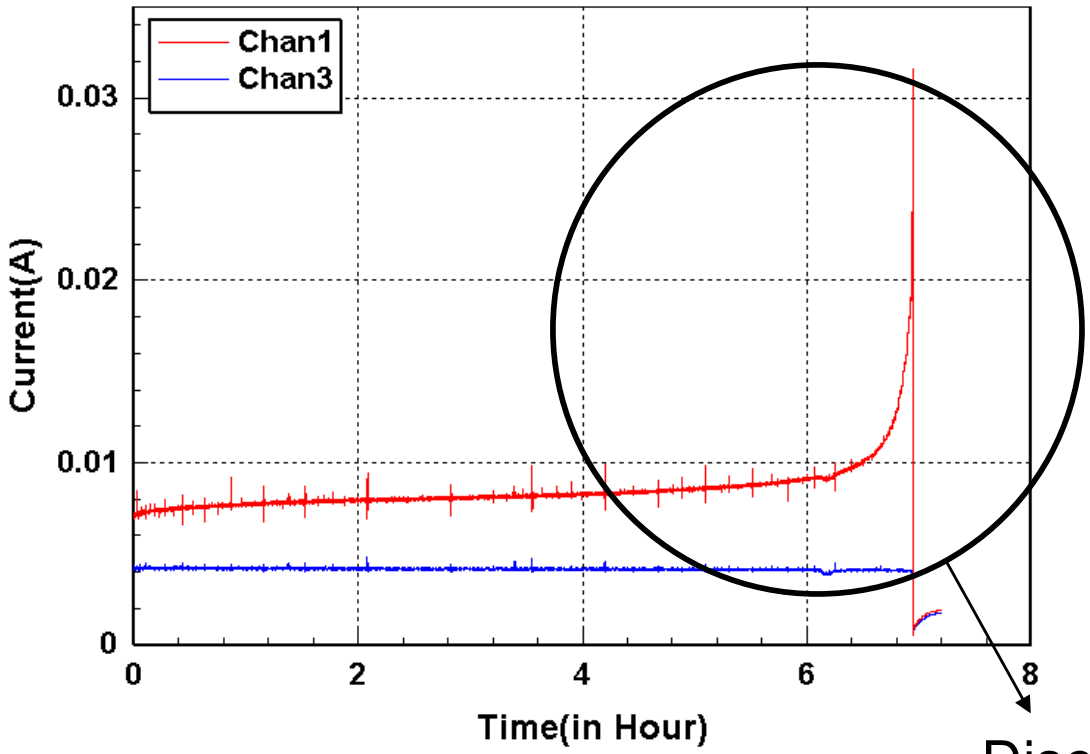
$T=CAP_{actual}(P)/P$

$S_{APP}=R_{APP}(P)\cdot T$



Actual Output when Non Battery Aware Approach is applied

Can we do better than Static Approach?



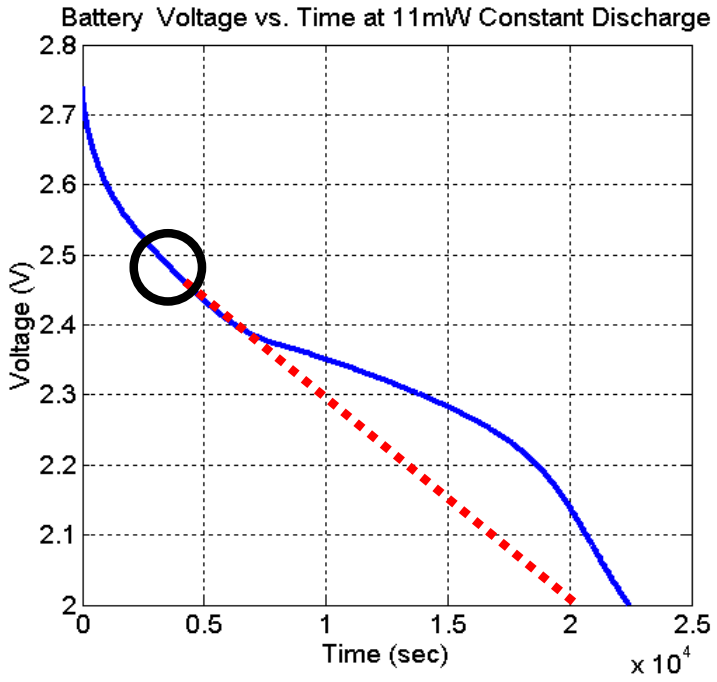
Discharge current toward the end of battery life increase rapidly due to DC/DC converter.

Dynamic Battery State Aware Approach Based on Voltage Slope

Make a linear projection based on voltage slope change in past intervals

$$S_{APP} = R_{APP}(p_i) \times T = R_{APP}(p_i) \times \frac{(V_{t_i} - V_{cutoff})}{\mu},$$

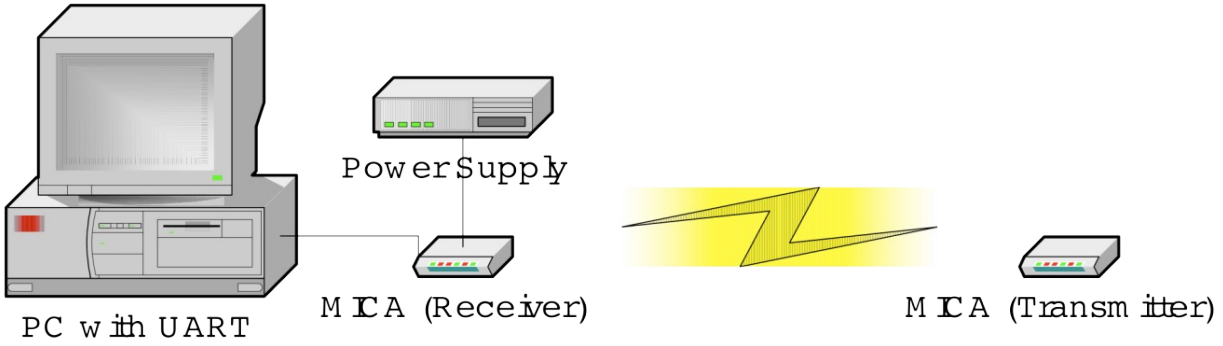
$$\text{where } \mu = \frac{V_{t_i} - V_{t_i - \Delta t}}{\Delta t}$$

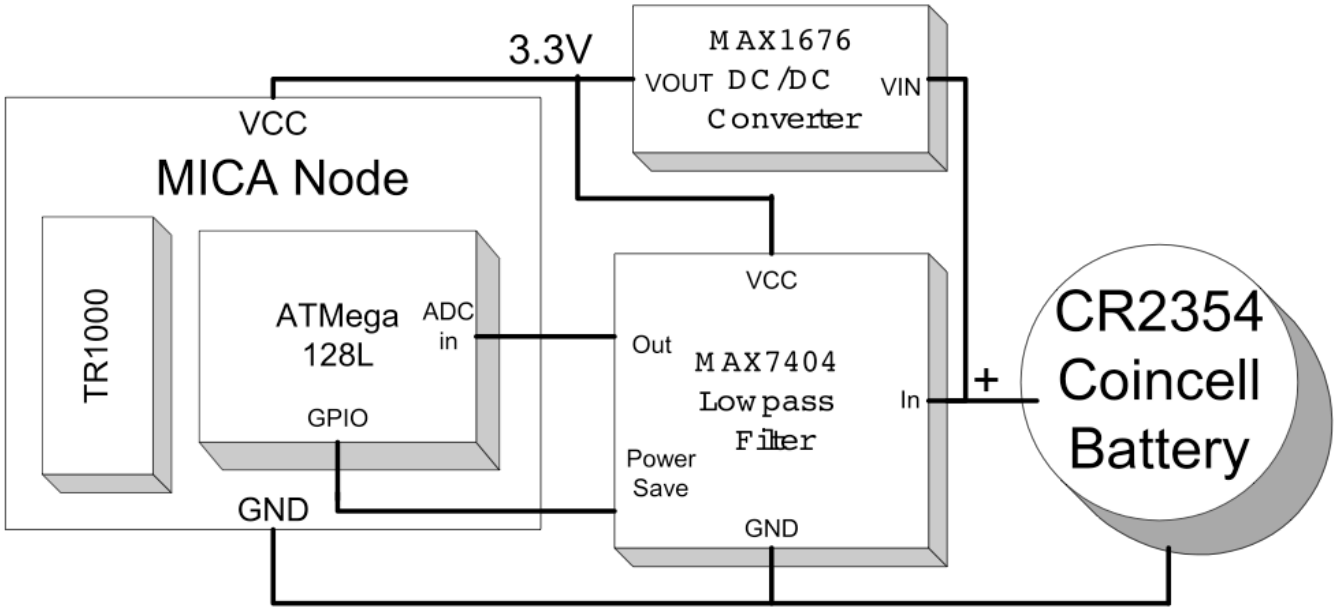


Dynamic Battery State Aware Approach- Based on Voltage Slope

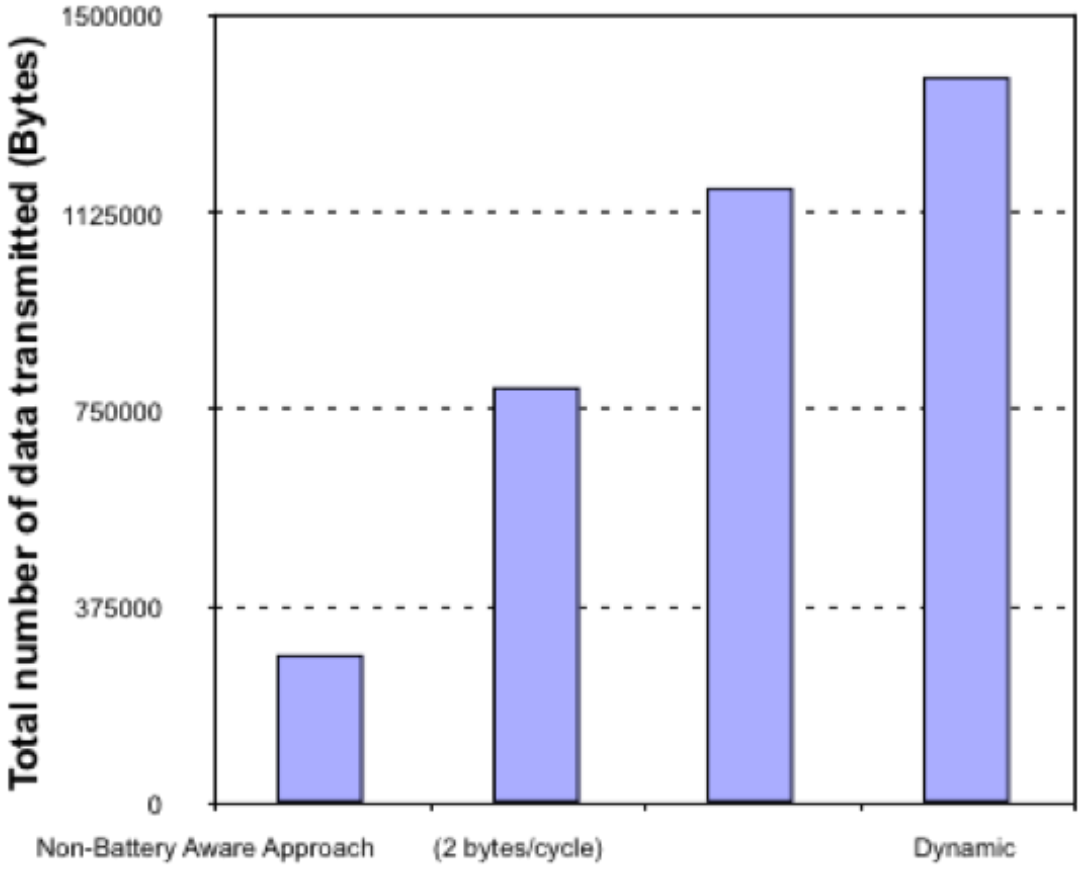
- 720% improvement over non-battery aware approach
- 20% improvement over the static approach
- Strength
  - Make use of Battery Voltage Information
  - Responds to battery state

Validation Set-up





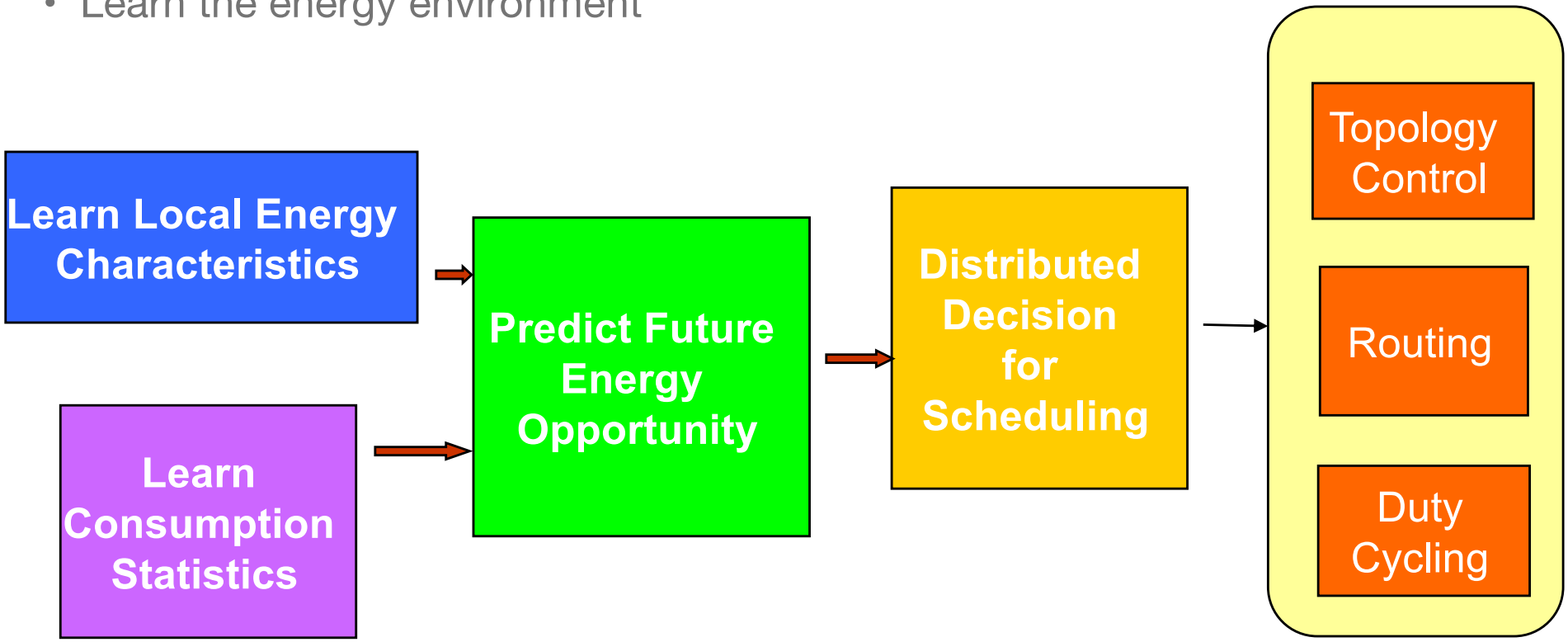
Result



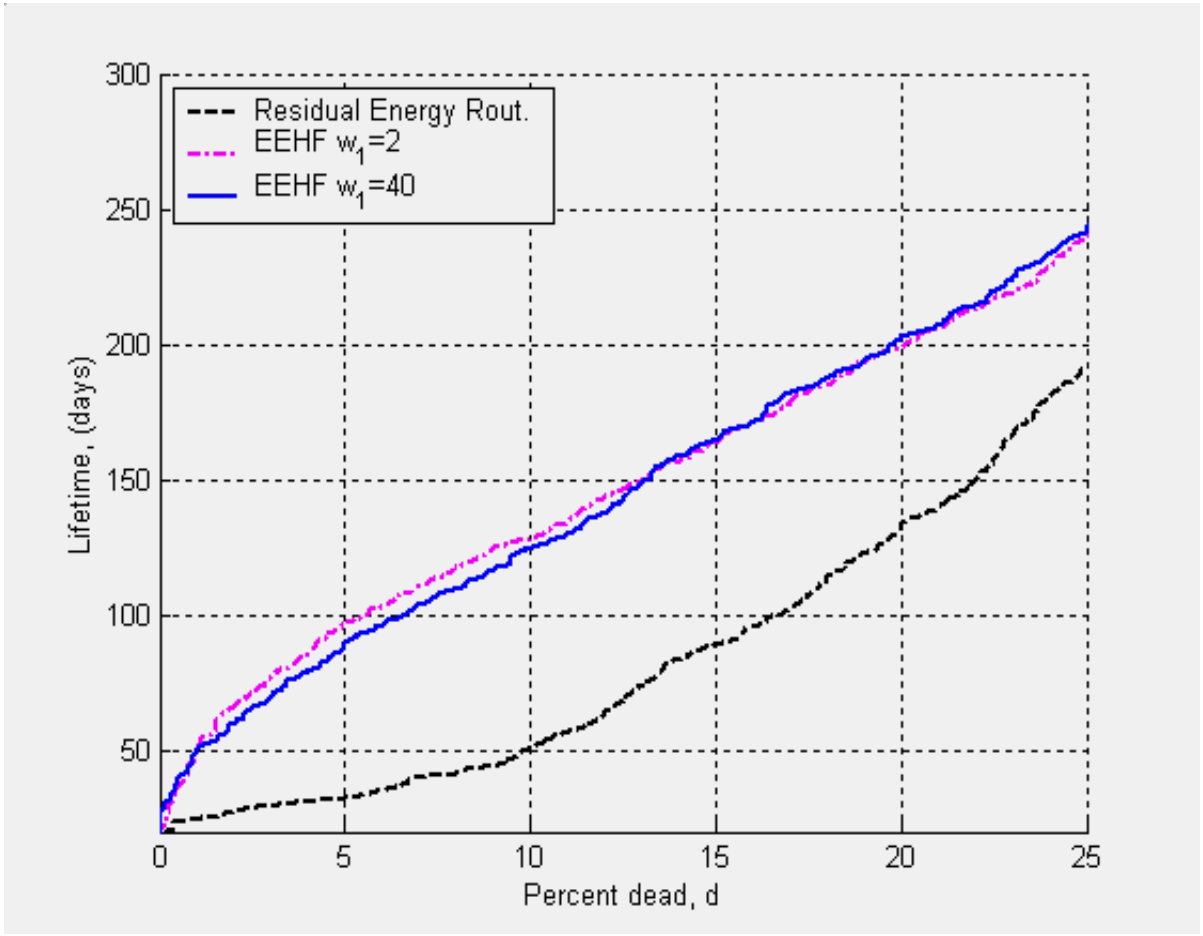
- 75% improvement over non-battery aware approach
- 18% improvement over the static approach (ranges from 12% to 25%)

Harvesting-aware Tasking

- Tasking aware of battery status & harvesting opportunities
  - Richer nodes take more load
  - Looking at the battery status is not enough
- Learn the energy environment



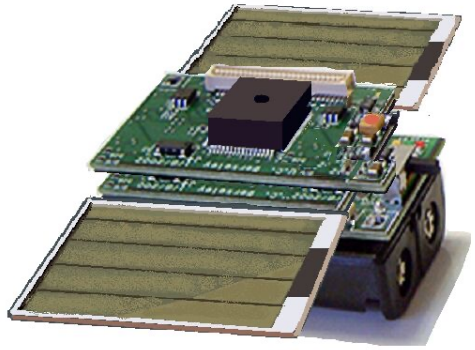
Example: Solar Harvesting Aware Routing



Simulation using light traces from James Reserve

morning

Afternoon



HelioMote Platform