Dynamic Power Management in Embedded Systems

Rajesh Gupta, UC San Diego

http://mesl.ucsd.edu

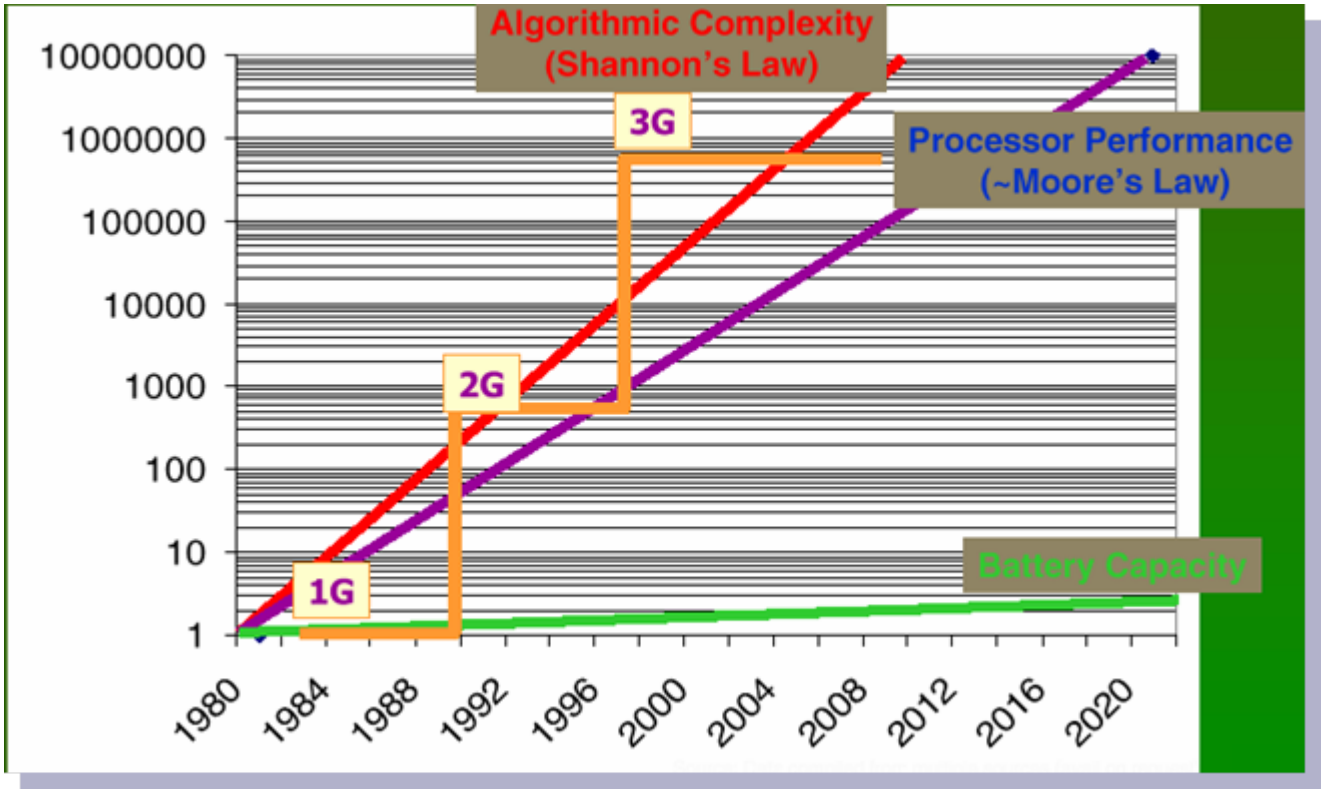
February 2007

Outline

- The case for energy and power efficiency
  - Where does power go?
  - What is new?
- What is dynamic power management?
  - DPM versus DSS
- Managing Power Across Multiple Layers
  - Application, System Software, Architecture, HW
- Summary

No Moore's Law For Batteries

- 2-3% growth per year in battery capacity



Source: J. Rabaey, UC Berkeley

Not that we would want it either…

	Power (Energy) Density	Source of Estimates
Batteries (Zinc-Air)	1050 -1560 mWh/cm^3 (1.4 V)	Published data from manufacturers
Batteries(Lithium ion)	300 mWh/cm^3 (3 - 4 V)	Published data from manufacturers
Solar (Outdoors)	15 mW/cm^2 - direct sun 0.15mW/cm^2 - cloudy day.	Published data and testing.
Solar (Indoor)	.006 mW/cm^2 - my desk 0.57 mW/cm^2 - 12 in. under a 60W bulb	Testing
Vibrations	0.001 - 0.1 mW/cm^3	Simulations and Testing
Acoustic Noise	3E-6 mW/cm^2 at 75 Db sound level 9.6E-4 mW/cm^2 at 100 Db sound level	Direct Calculations from Acoustic Theory
Passive Human Powered	1.8 mW (Shoe inserts >> 1 cm^2)	Published Study.
Thermal Conversion	0.0018 mW - 10 deg. C gradient	Published Study.
Nuclear Reaction	80 mW/cm^3 1E6 mWh/cm^3	Published Data.
Fuel Cells	300 - 500 mW/cm^3 ~4000 mWh/cm^3	Published Data.

➲ Must reduce power, energy consumption.

Low Power Has Been A Design Focus

- Speed power efficiency has indeed gone up
  - 10x / 2.5 years for μPs and DSPs in 1990s
    - between 100 mW/MIP to 1 mW/MIP since 1990
  - IC processes have provided 10x / 8 years since 1965
  - rest from power conscious IC design in recent years
- Another 20X is possible.

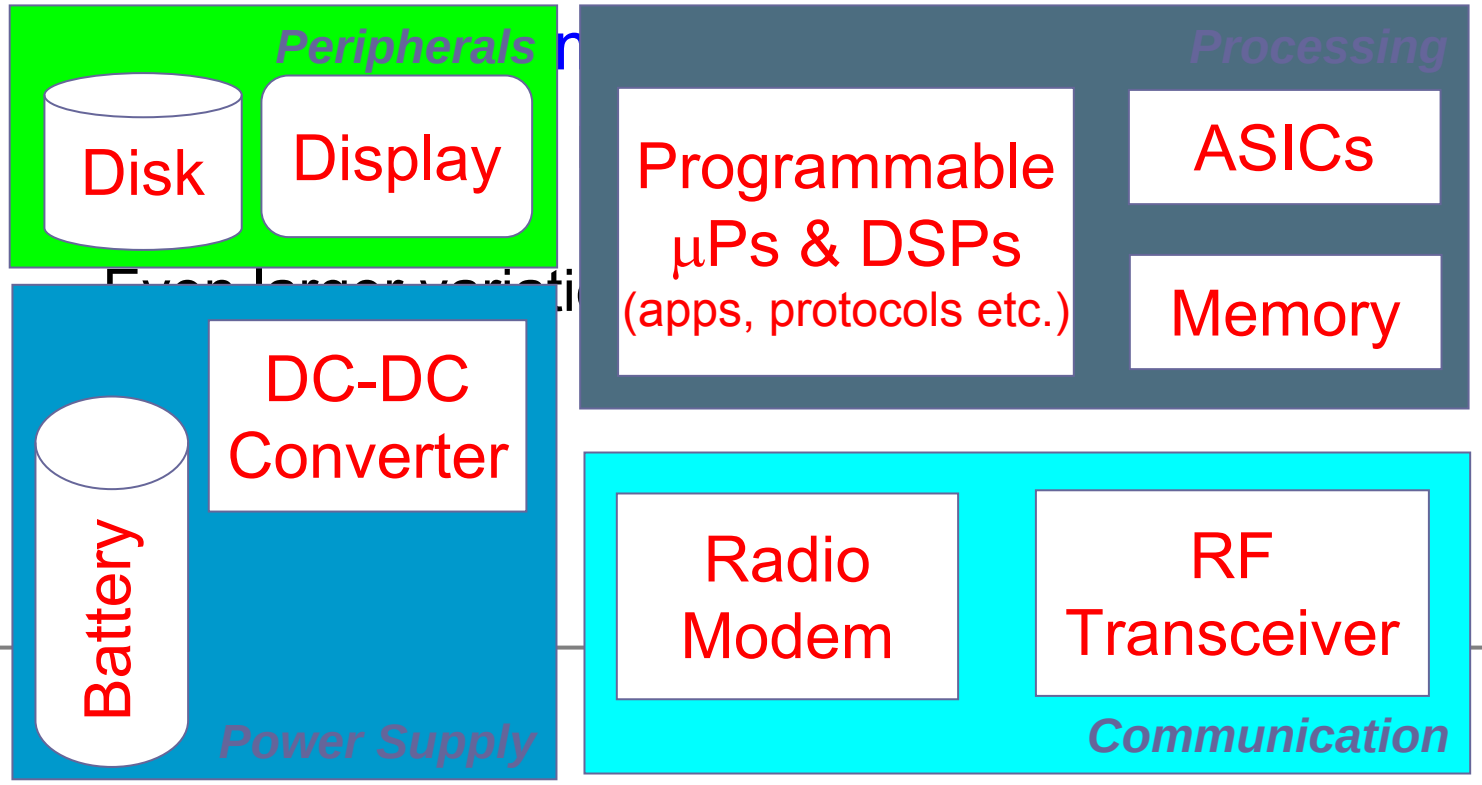
Processor	MHz	Year	SPECint-95	Watts
P54VRT (Mobile)	150	1996	4.6	3.8
P55VRT (Mobile MMX)	233	1997	7.1	3.9
PowerPC 603e	300	1997	7.4	3.5
PowerPC 604e	350	1997	14.6	8
PowerPC 740 (G3)	300	1998	12.2	3.4
PowerPC 750 (G3)	300	1998	14	3.4
Mobile Celeron	333	1999	13.1	8.6

Source: ISI/USC, DARPA PACC Program

Unfortunately, that is not enough

- There is a bottom line to energy efficiency…
  - Computation cost (2004 projected): 60 pJ/op
  - Minimum thermal energy for communications:
    - 20 nJ/bit @ 1.5 GHz for 100 m
    - 2 nJ/bit @ 1.5 GHz for 10 m
- … but not to the need for energy (or power).

The New Age Computing & Comm.

- New devices with markedly different usage of energy and power (than desktops and laptops)
  - 

"System Design" for Low Power

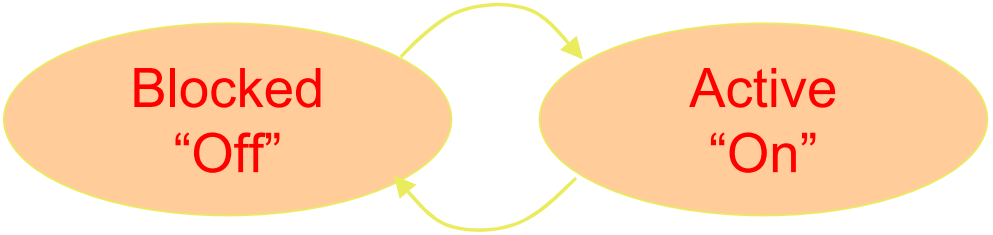
- Energy efficiency (has to) cut across all system layers
  - circuit, logic, software, protocols, algorithms, user interface, power supply...
  - Node versus network
- Trade-off between energy consumption & QoS
  - optimize energy metric while meeting "quality" constraint

➲ When all low power tricks have been done, "duty cycling" remains the only available variable to reduce energy consumption

- Must capture the "application intent".

Shutdown for Energy Saving

- Use low duty cycle of many subsystems:

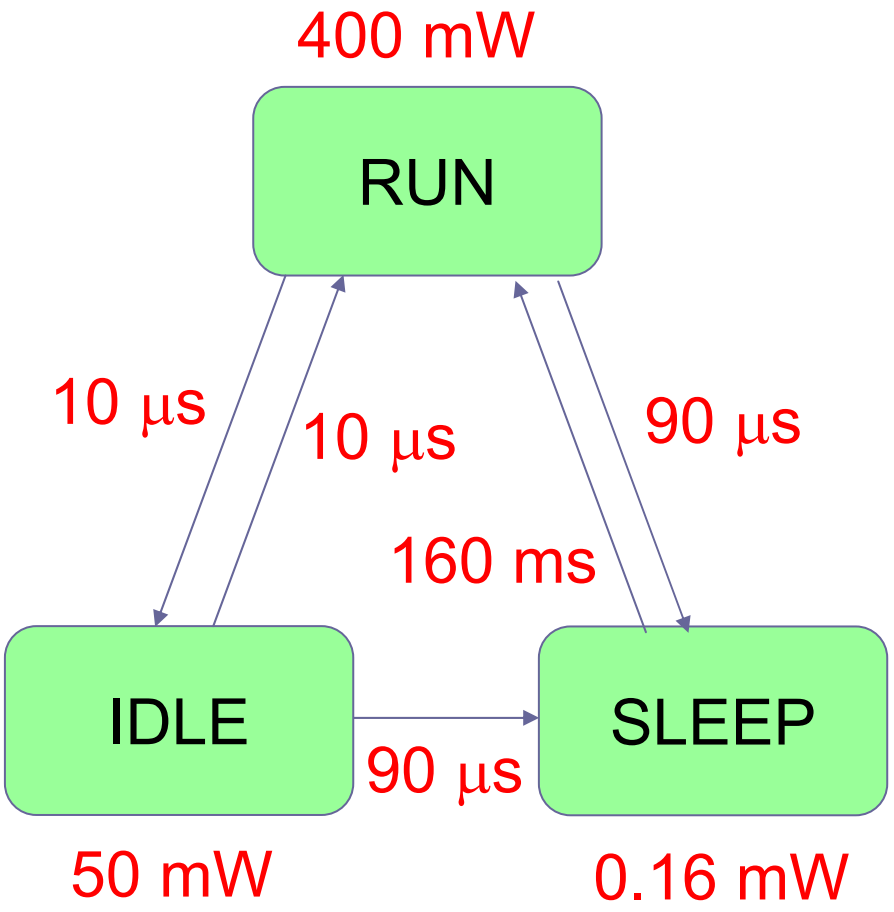


T_{block} T_{active} **ideal improvement = 1 + T_{block}/T_{active}**

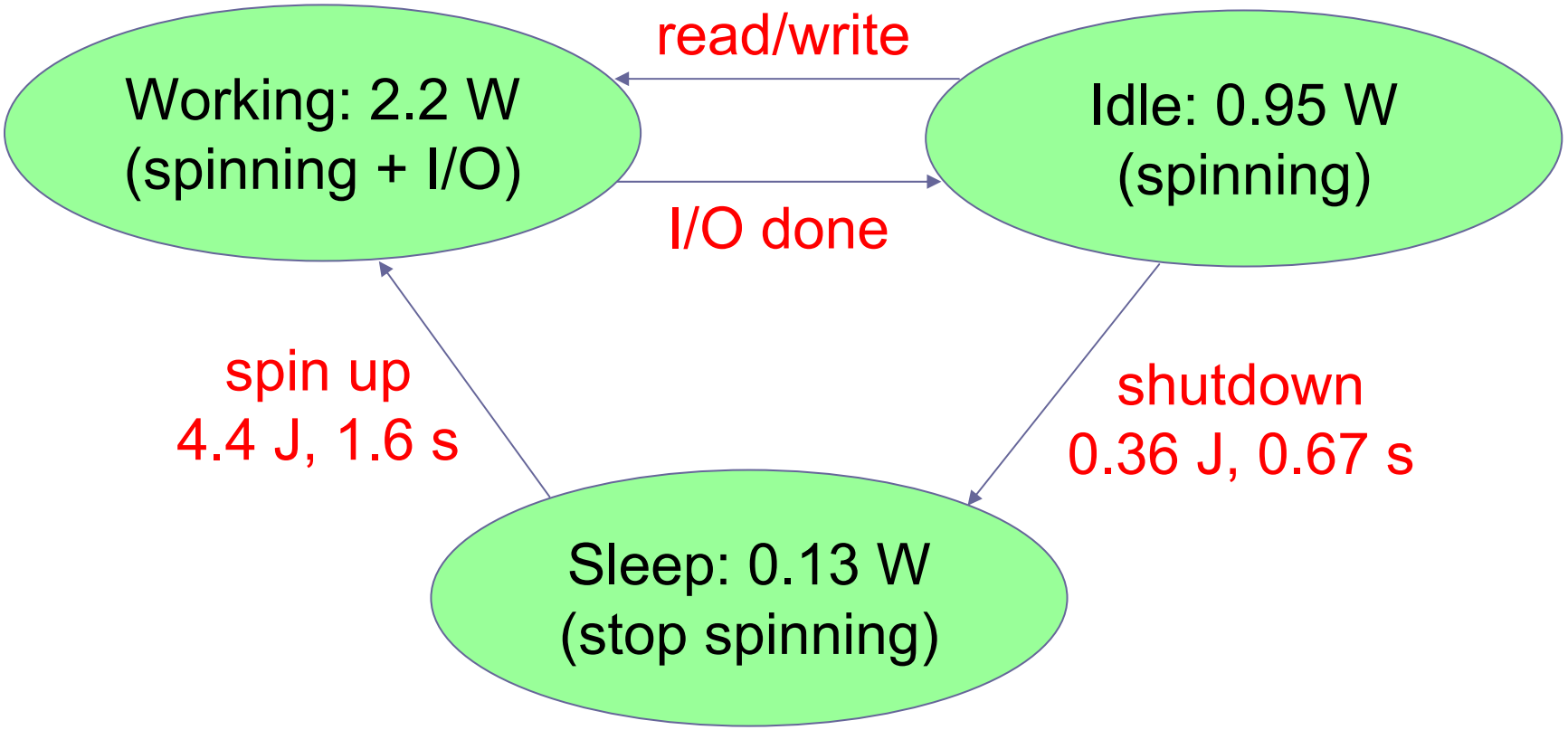
- Large difference between "on" & "off" power

Example: SA-1100 CPU

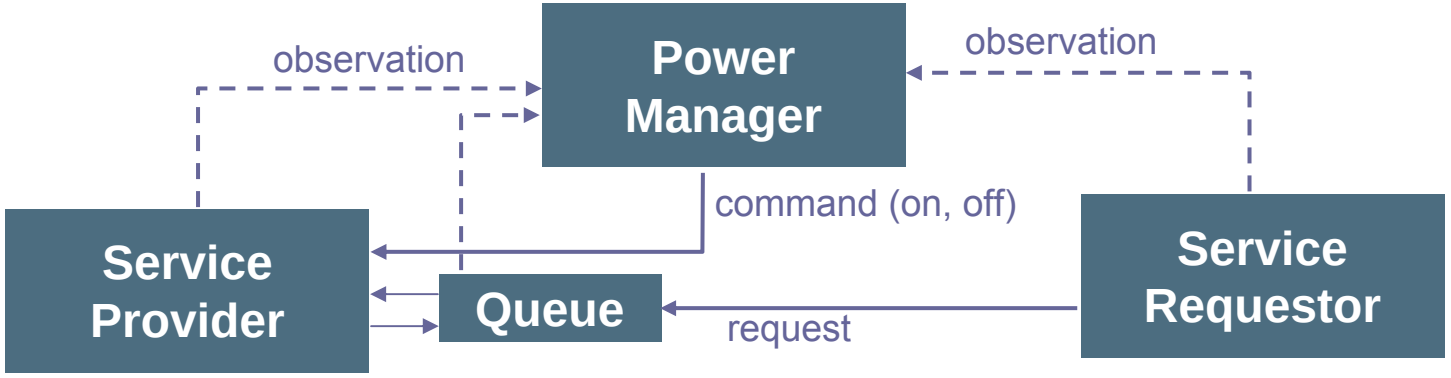
- RUN
- IDLE
  - CPU stopped when not in use
  - Monitoring for interrupts
- SLEEP
  - Shutdown on-chip activity



Example: Fujitsu MHF 2043 AT



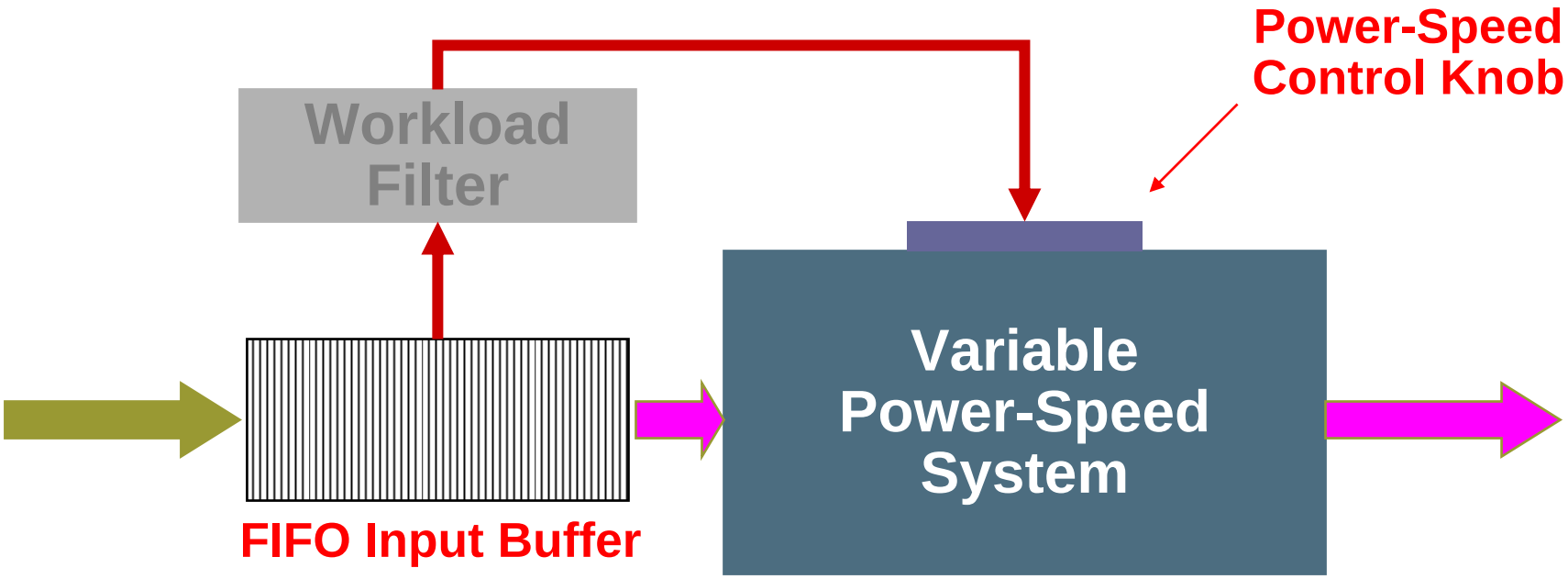
A Generic Power-managed System



- Lots of prior and ongoing activity in defining the interface between power-manageable components (chips, disk driver, display driver, radios, etc.) and the power manager (PM)
- Components (or service providers) have internal states
  - Abstracted as power state machine
  - power and service annotation on states; power and delay annotation on edges
- PM Policy: uses insight into how and when to invoke PM

E.G.: Dynamic Speed Scaling: V/f

- Generic system architecture
  - many examples in hardware and software



Controlling Power/Energy Consumption

- DPM
  - "shutdown" through choice of right system & device states
    - Multiple sleep states
- DSS
  - "slowdown" through choice of right system & device states
    - Multiple active states
- DPM + DSS
  - Choice between amount of slowdown and shutdown
  - However, the problem changes "qualitatively".

Slowdown Preferred Over Shutdown

- Example: 50ms task with 100ms deadline
  - 50ms computation, 50ms idle/stopped time
  - Half speed: 100ms computation, 0ms idle
    - ¼ energy than the full speed case *if* voltage scales
- Use voltage to control the operating point on the power vs. speed curve
  - i.e., power and clock frequency are functions of voltage
- Because of higher savings in slowdown often shutdown becomes a "secondary" strategy
  - First slow down and then look for ways to shutdown.

Slowdown loosing its effectiveness..

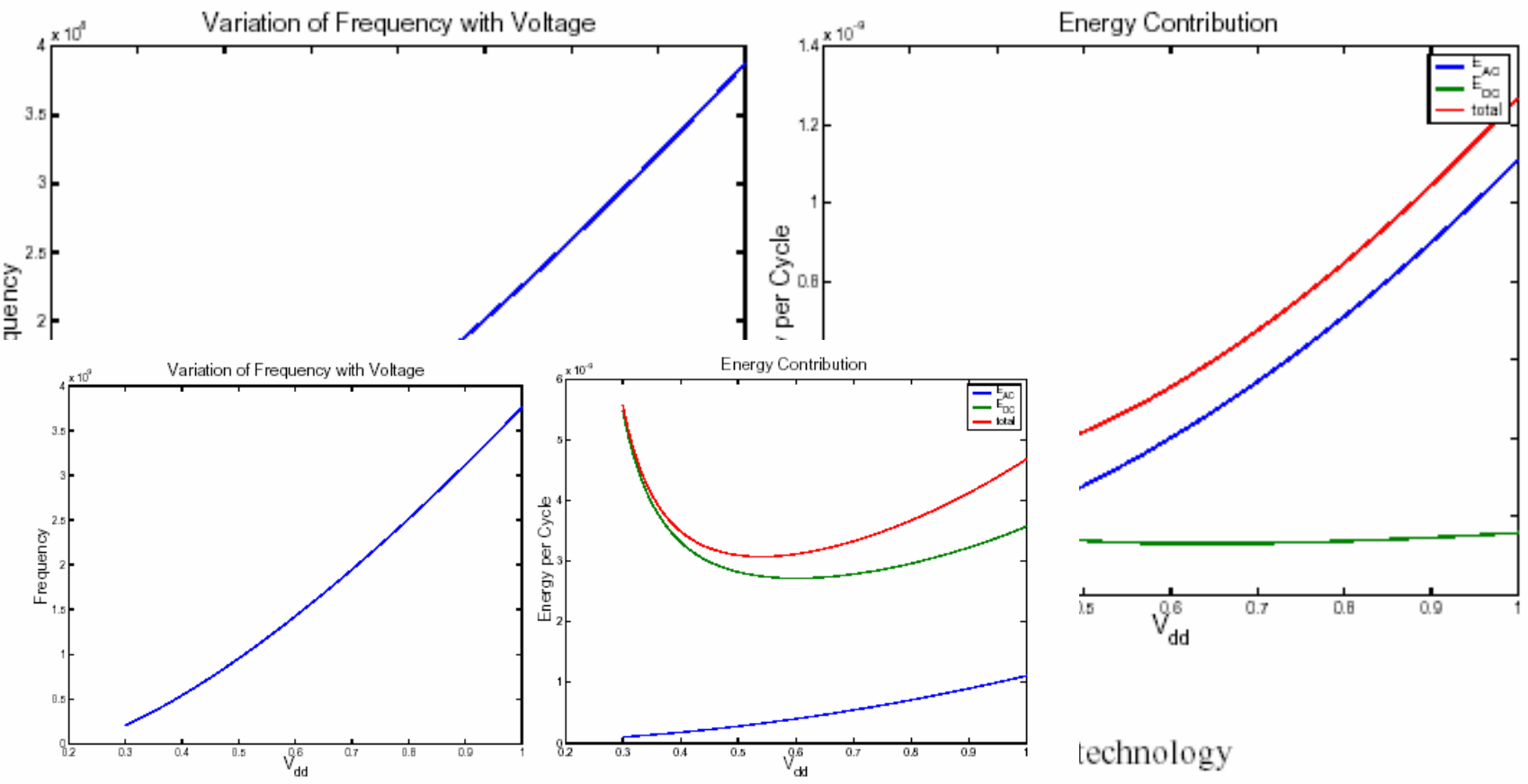
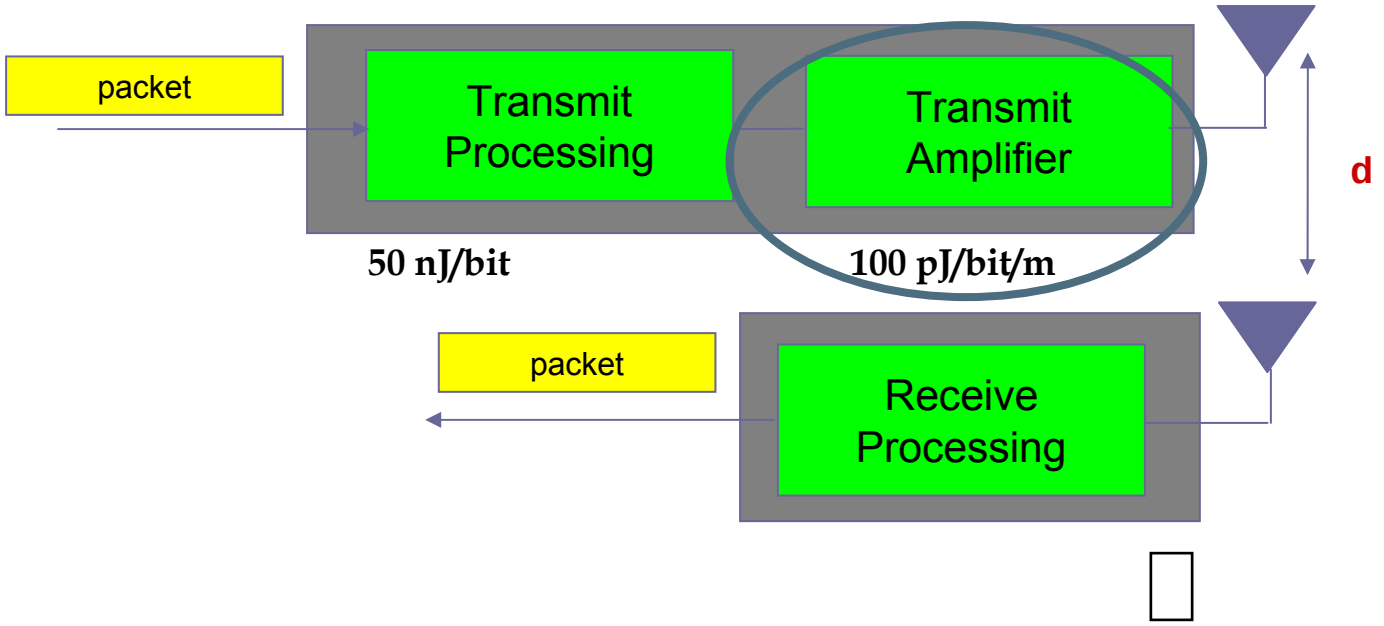


Figure 2. Cycle time and Energy variation for $0.07\mu m$ technology

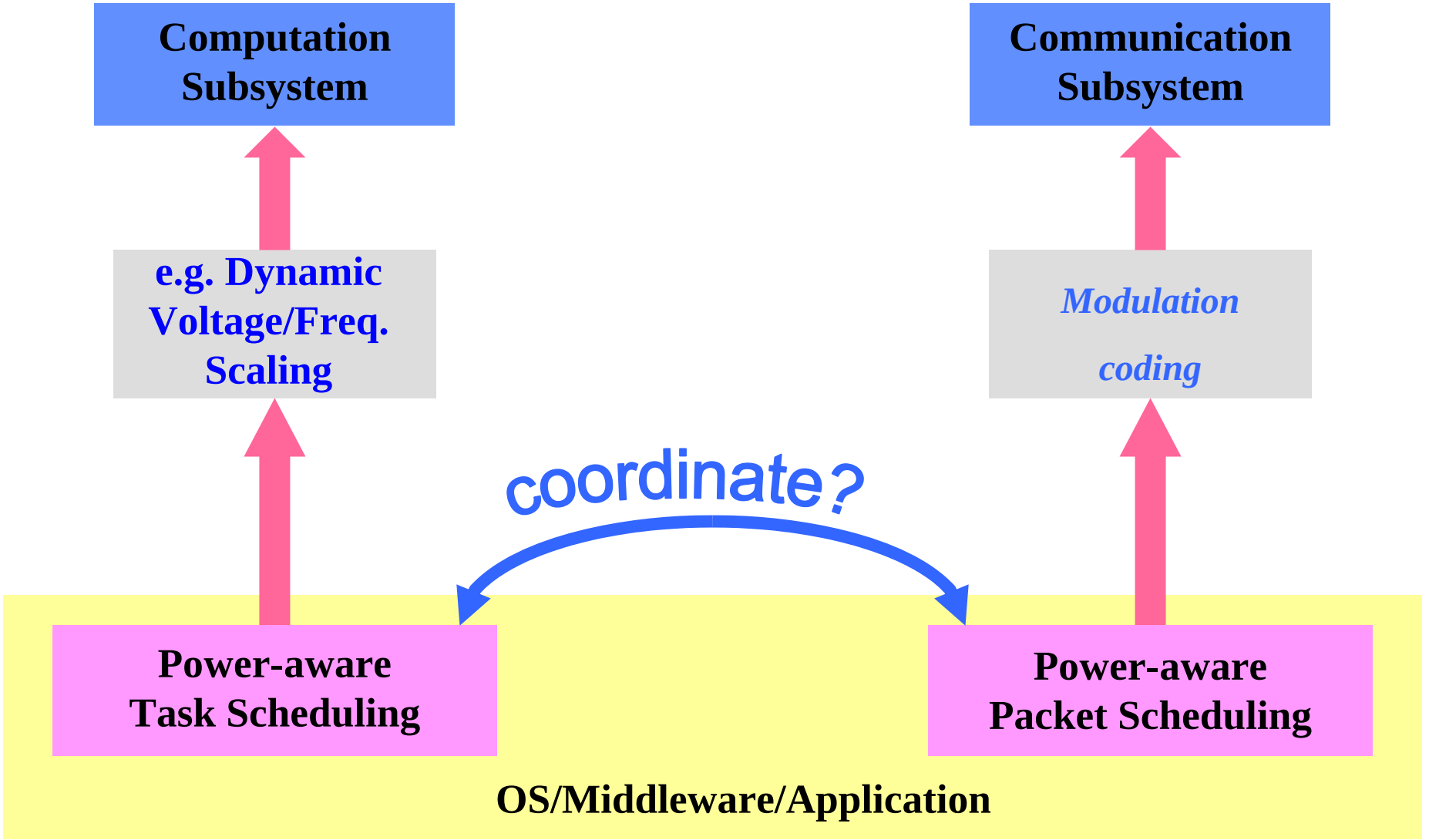
Radio's Need Duty-cycling Too



- The RF power dominates over the electronic power associated with transmit or receive processing, except for short distances (5-10 m).
- Voltage scaling and other low power logic techniques do not affect RF power consumption

Source: Heinzelman ISLPED'00; Wang ISLPED'01

Power Management in Communication Subsystems



Power Savings Mechanisms

Dynamic Power Management (DPM)

- When a device is idle, it can transition to low-power "sleep" states.
- Current trend is to design devices with multiple sleep states and provide device driver hooks to change these states under OS control.

Dynamic Voltage Scaling (DSS, or DVS)

- A device can be run at different speeds at different power levels
- Execution of jobs can be slowed down to save power as long as all jobs are completed by their deadline.

Application level "knobs"

- quality and performance measures, application tolerances

A. Dynamic Power Management

- When a device becomes idle, it can transition to lower power usage state.
- A fixed amount of additional time and energy are required to transition back to active state when a new request for service arrives.
- What is the best time threshold to transition to the sleep state?
  - Too soon: pay start-up cost too frequently.
  - Too late: spend too much time in the high-power state
- Generally, transition to sleep state when the cost of being in active state is at least the cost of 'waking up'.

Our Work In This Context

- We have developed quantitative bounds on the quality of DPM algorithms based on Competitive Analysis [TCAD 01]
  - provides a basis for DPM strategy comparison
- Developed DPM strategies for devices with both multiple active and multiple sleep states [TCAD 02]
- Design and analyze algorithms for systems that allow both DPM and DVS [SODA 03, TECS02]
- Important conclusions
  - **Not all power states are useful in a given DPM strategy**
  - **DPM generally useful for improving quality measures.**

Competitive Analysis

- Deterministic algorithm (ski rental)
  - Transition to sleep state when the cost of being in active state is at least the cost of 'waking up'.
    - Normalize cost of transitioning from sleep to active state to 1.
    - Power consumption rate of active state is α.
  - This algorithm is 2-competitive.
  - 2 is the best possible competitive ratio for any deterministic algorithm.
- Probabilistic algorithm
  - Idle period length generated by known distribution with density function p(t).
  - Choose threshold T to minimize cost:

$$\arg \min_{T} \int_{0}^{T} \alpha t \; p(t)\, dt \; + \int_{T}^{\infty} [\alpha T \; + 1]\, p(t)\, dt$$

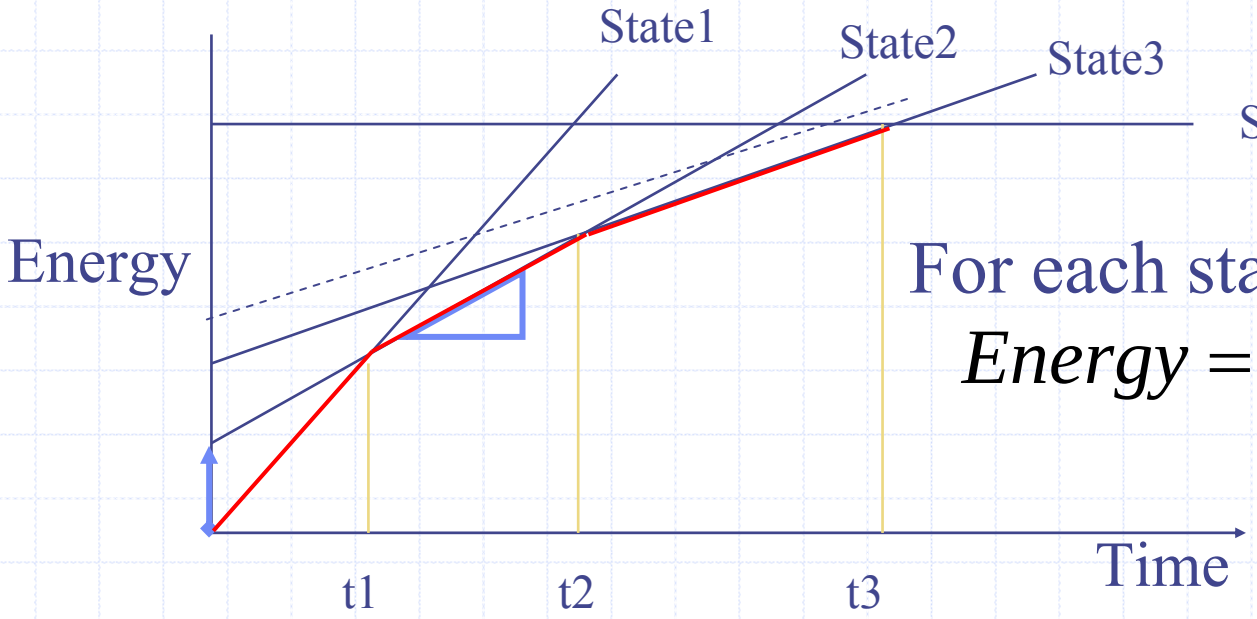
  - For any distribution p(t), the expected cost of the above algorithm is within e/(e-1) of the optimal cost. Furthermore, there is a distribution for which no algorithm can be better than e/(e-1) times optimal.

Multi-state DPM Case

- Let there be k+1 states
  - Let State k be the shut-down state and 0 be the active state
  - Let α_i be the energy dissipation rate at state i
  - Let β_i be the total energy dissipated to move back to State 0
  - States are ordered such that $\alpha_{i+1} \leq \alpha_i$
  - $\alpha_k = 0$ and $\beta_0 = 0$ (without loss of generality).
  - Power down energy cost can be incorporated in the power up cost for analysis (if additive).

- Now formulate an optimization problem to determine the state transition thresholds.

Lower Envelope Idea



For each state i, plot:

$$Energy = \alpha_i (Time) + \beta_i$$

- LEA can be deterministic or probabilistic

$$T_i = \arg\min_T \int_0^T [\alpha_{i-1} t + \beta_{i-1}]\, p(t)\, dt$$

$$+ \int_T^\infty [\alpha_{i-1} T + \alpha_i (t - T) + \beta_i]\, p(t)\, dt$$

  - PLEA is e/(e-1) competitive.

Model Checking

- Model the system as a two player game between DPM algorithm and a non-deterministic adversary
  - Adversary can generate the worst-case input for the algorithm.
- Use model checking to automatically verify whether a specific algorithm achieves a given competitive ratio.

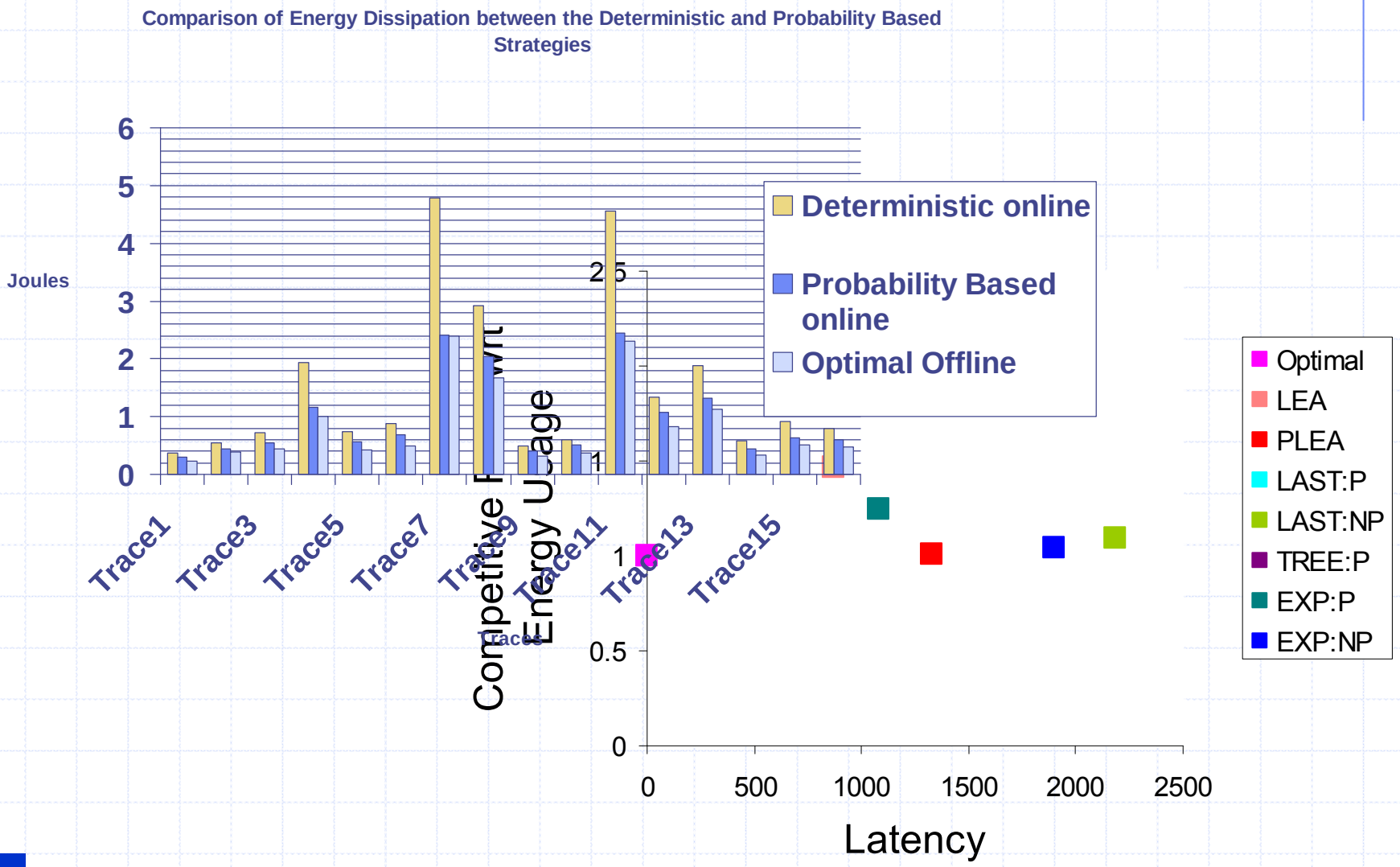
[Shukla, Gupta 01]

Experimental Study: IBM Mobile Hard Drive

State	Power Consumption	Start-up Energy (Joules)	Transition Time to Active
Sleep	0	4.75	5s
Stand-by	0.2	1.575	1.5s
Idle	0.9	0.56	40ms
Active/Busy	2.4	0	0s

Trace data with arrival times of disk accesses from Auspex file server archive.

IBM Mobile Hard Drive



B. Dynamic Voltage Scaling

- Device which can run at any speed s.
- Power consumed if running in state s is given by convex function P(s).
- Jobs arrive through time. Job j has:
  - Arrival time: a_j
  - Deadline: b_j
  - Work required: R_j
- Schedule S = (s, job)
  - s(t) is the speed of the device at time t.
  - job(t) is which job is executed at time t.

Dynamic Voltage Scaling

(Dynamic Voltage Scaling - No Sleep: DVS-NS)

- Schedule S is feasible for set of jobs J if for every j in J:

$$\int_{a_j}^{b_j} s(t)\delta(job(t), j)dt = R_j$$

- Cost of Schedule S is:

$$cost(S) = \int_{t_0}^{t_1} P(s(t))dt$$

DVS with Sleep State (DVS-S)

- Schedule S = (s, job, h):
  - h(t) = sleep or on
  - If h(t) = sleep, then s(t) = 0.
- Power is a function of speed and state:
  - P(s, state) = P(s) if state = on.
  - P(s, state) = 0 if state = sleep.
- P(0) = α is power required to keep device active with no tasks running.
- Let k be the number of times the device transitions from sleep state to the on state
- Cost of a schedule S is: $cost(S) = k + \int_{t_0}^{t_1} P(s(t), h(t))dt$

Critical Speed

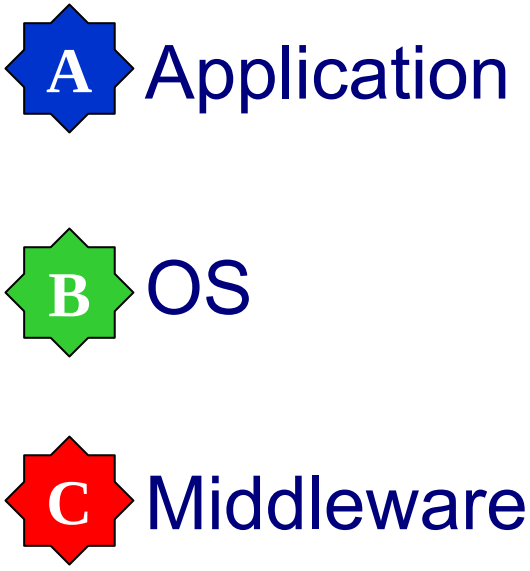
- If the cost to transition from sleep state to the on state were 0, the optimal speed for all jobs would be the s that minimizes $(R_j/s)\ P(s)$
  - This is the s that satisfies $P(s) = s\ P'(s)$.
  - Call this S_{crit}, the critical speed for α.
- If we compress the execution of a task by x,
  - we expend additional energy because we execute the job faster
  - we save α x.
  - S_{crit} is the point at which it is no longer beneficial to compress the execution of a task.
- Our approach
  - Decide on Active/Idle intervals (determined by critical speed)
  - Decide on Sleep/On intervals (determined by the cost of staying on)

C. Enable Application "Knobs"

- Knowing an application's intent one can do a lot of power saving tricks at all levels: architecture, compiler, OS, middleware
- Conversely, if the awareness for power/energy is seeped into all these levels, one can reduce power significantly
- Together they can create a new contract in the computing system!
- Since power is important in radios, things with radios move (or they monitor things that move), location awareness is even more phenomenal for power reduction.

Outline:

Bringing energy awareness in application, OS and Middleware

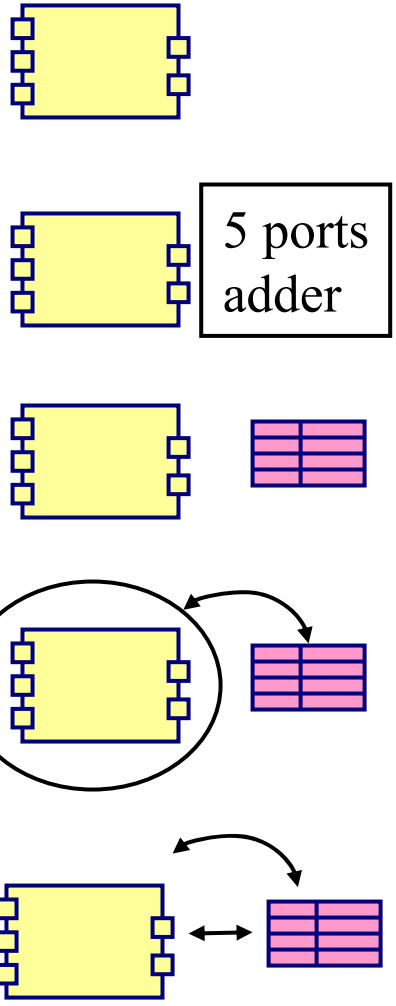


What does is mean to be ‘aware’?

- That the application and the services know about energy, power
  - File system, memory management, process scheduling
  - Make each of them energy aware
- How does one make software to be “aware”?
  - Use “reflectivity” in software to build adaptive software
  - Ability to reason about and act upon itself (OS, MW)

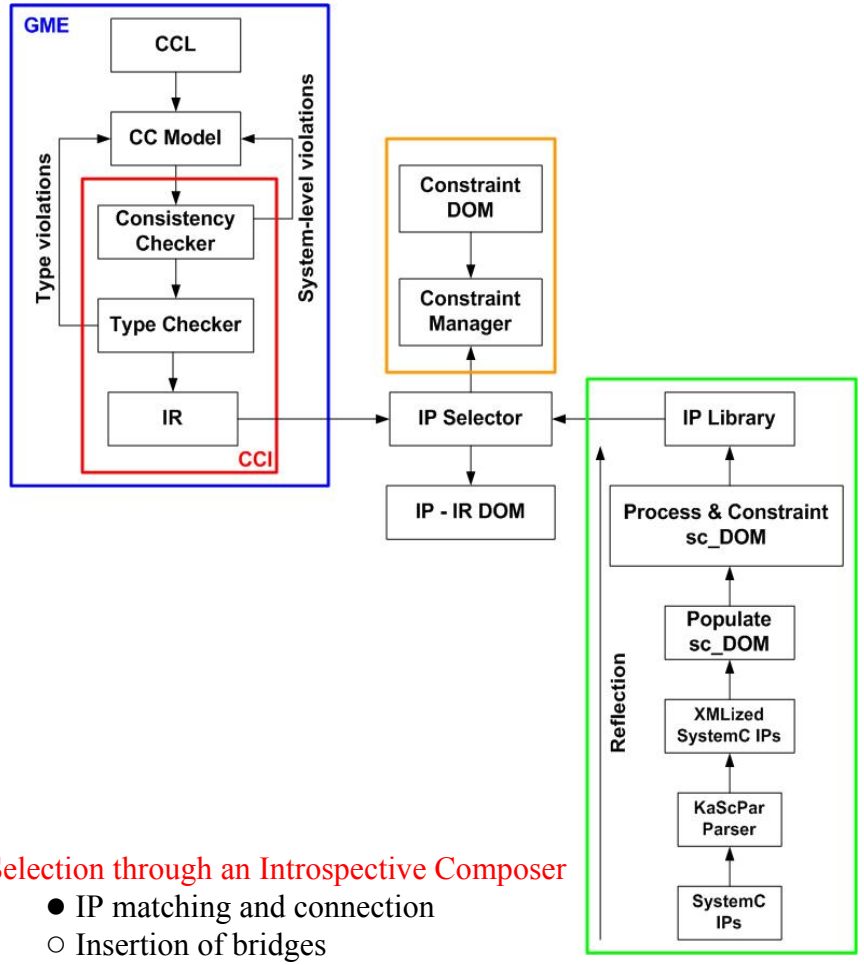
Reflection and Introspection: A HW Guy's Way of Looking At It

- Component:
  - A unit of re-use with an interface and an implementation
- Meta-information:
  - Information about the structure and characteristics of an object
- Reification:
  - A data structure to capture the meta-information about the structure and the properties of the program
- Reflection:
  - An architectural technique to allow a component to provide the meta- information to himself
- Introspection:
  - The capability to query and modify the reified structures by a component itself or by the environment



Building HW Components W/ Meta-data

1. Start with SystemC descriptions of IP blocks
   - Multi-level (RTL, TL) descriptions
2. Capture meta information of these IP into XML
   - Mostly structural information for now.
3. Generate library of 'XMLized' IP blocks
   - Schema to match datatype and protocol type information across IP blocks
   - Create DOM model and constraints for the library
4. Develop methods for IP selection, composition, verification, synthesis
   - Automated methods for IP instantiation, interface generation



IP Selection through an Introspective Composer

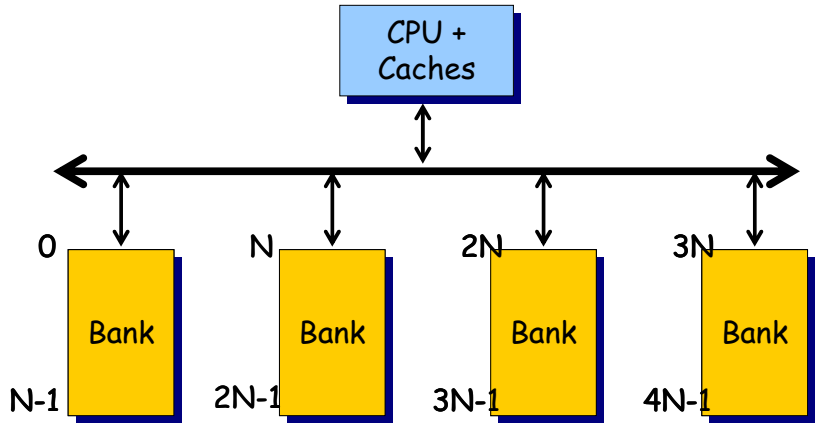
- IP matching and connection
  - Insertion of bridges
  - Validation of functionality
  - Create an executable specification

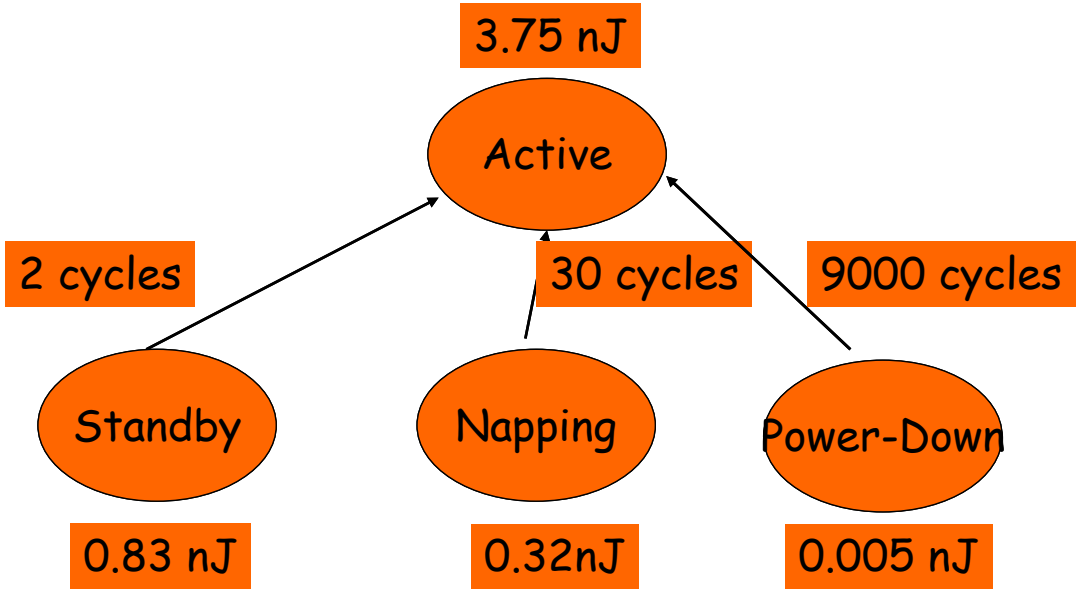
Applying Reflection: performance, energy

- Use Meta data to represent resource demands, dynamic behavior of the program carrying it.
  - Resources: Memory (R/W, Cache), Processor (IPC)
- Enables energy-performance tuning by exploring resource demand variations throughout programs' execution
  - Example: Profile of application over memory banks
  - Vary frequency of processor based on IPC demand

Example: Rambus DRAM (RDRAM)

- High bandwidth (>1.5 GB/sec)
- Each RDRAM module can be activated/deactivated independently
- Read/write can occur only in the active mode
- Three low-power operating modes:
  - Standby, Nap, Power-Down



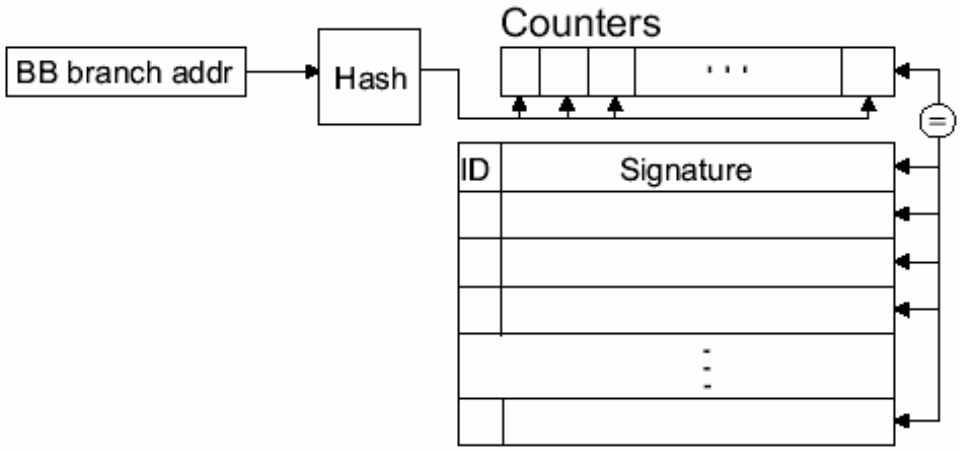


Approach

1. Characterize application offline
   - Divide an application into phases of execution
     - A group of program intervals executing similar code
   - Each phase has similar demand on resources
     - Similar code, similar resource demands (memory, IPC)
2. Annotate source code
   - Phase signatures
3. Enable OS (and hardware) to recognize signature
   - Smart hardware and/or online learning techniques
4. Dynamically tune the power manager
   - As application moves from one phase to another.

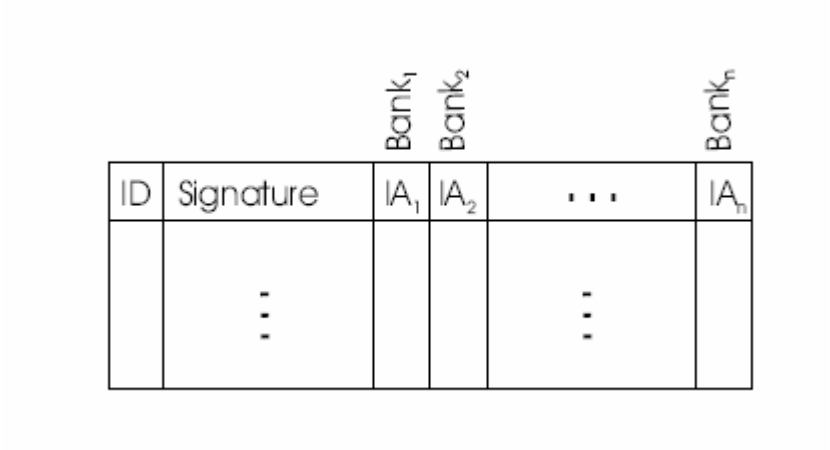
1 Understanding application behavior

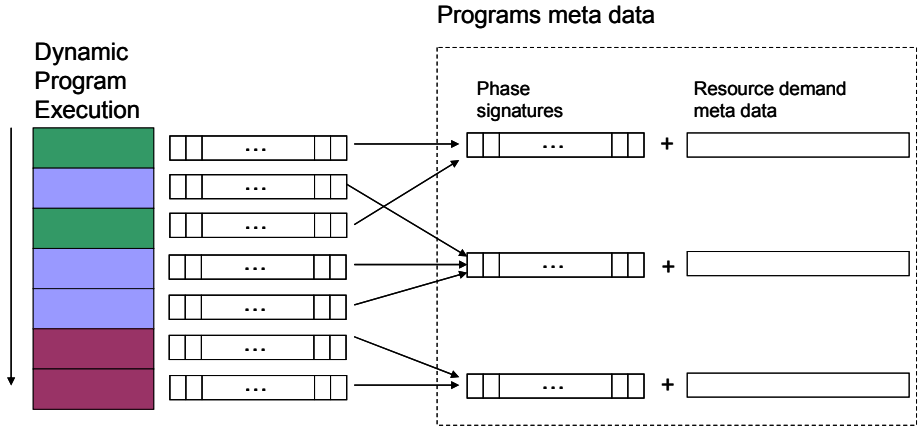
- Divide an application into phases of execution
  - A group of program intervals executing similar code
- Each phase has similar dema on resources
  - Similar code, similar resourc demands
- Demand for resources varies during the execution of application
  - As it moves from one phase to another.
- Phases identified using BBV or LBV
  - Keep track of loop branches



2 Offline analysis of application

- A data structure with the summary of the information of interest for each phase is attached to the program
  - Fixed location for the program metadata
  - OS support to access metadata.
- Also a signature of each phase is attached to the program.
  - No. of times the loops of the program are executed in the particular phase

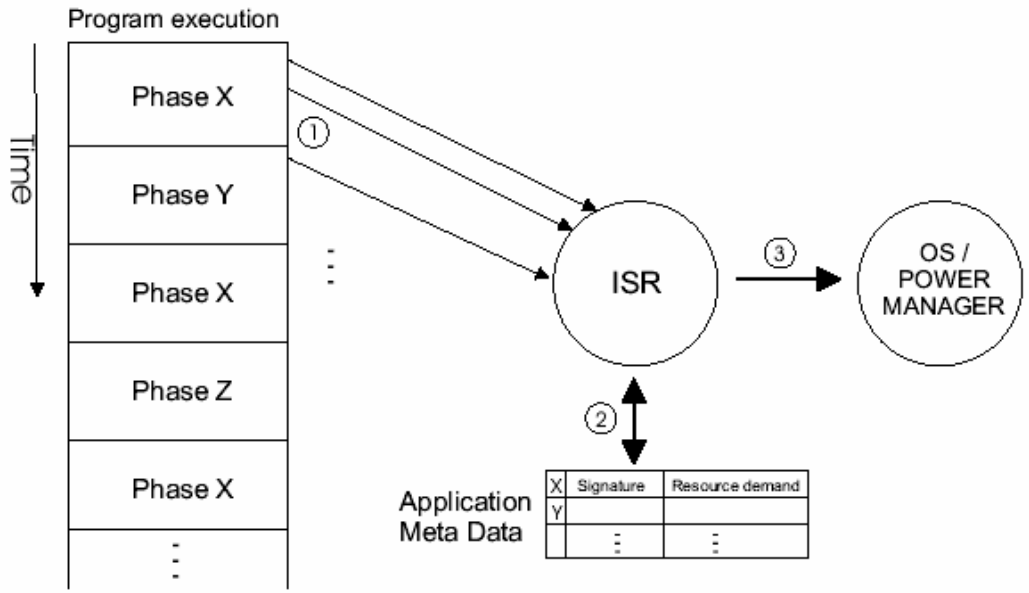




3 Runtime Analysis

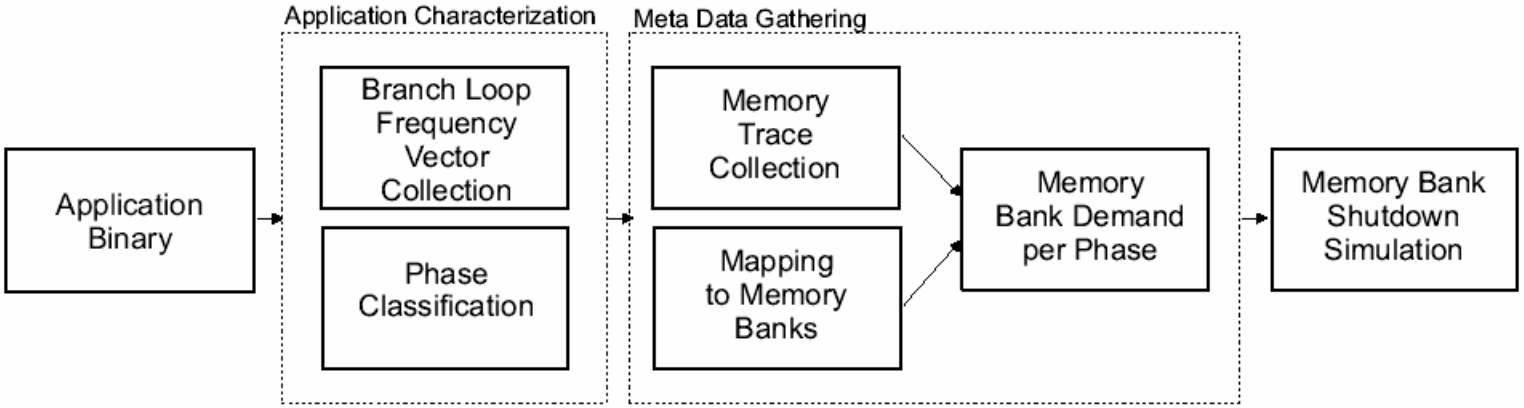
- Challenge:
  - Detect in which phase the program is running
- Learning, signature construction, partial matches
  - Match the signature created offline with a online signature using Manhattan distance.
    - If distance < threshold a match is found.
    - Threshold tells how similar two intervals of execution are.
  - The same technique used for splitting the program in phases offline is applied online but using partial signatures for matching with the offline computed signatures.

4 Matching signature at runtime



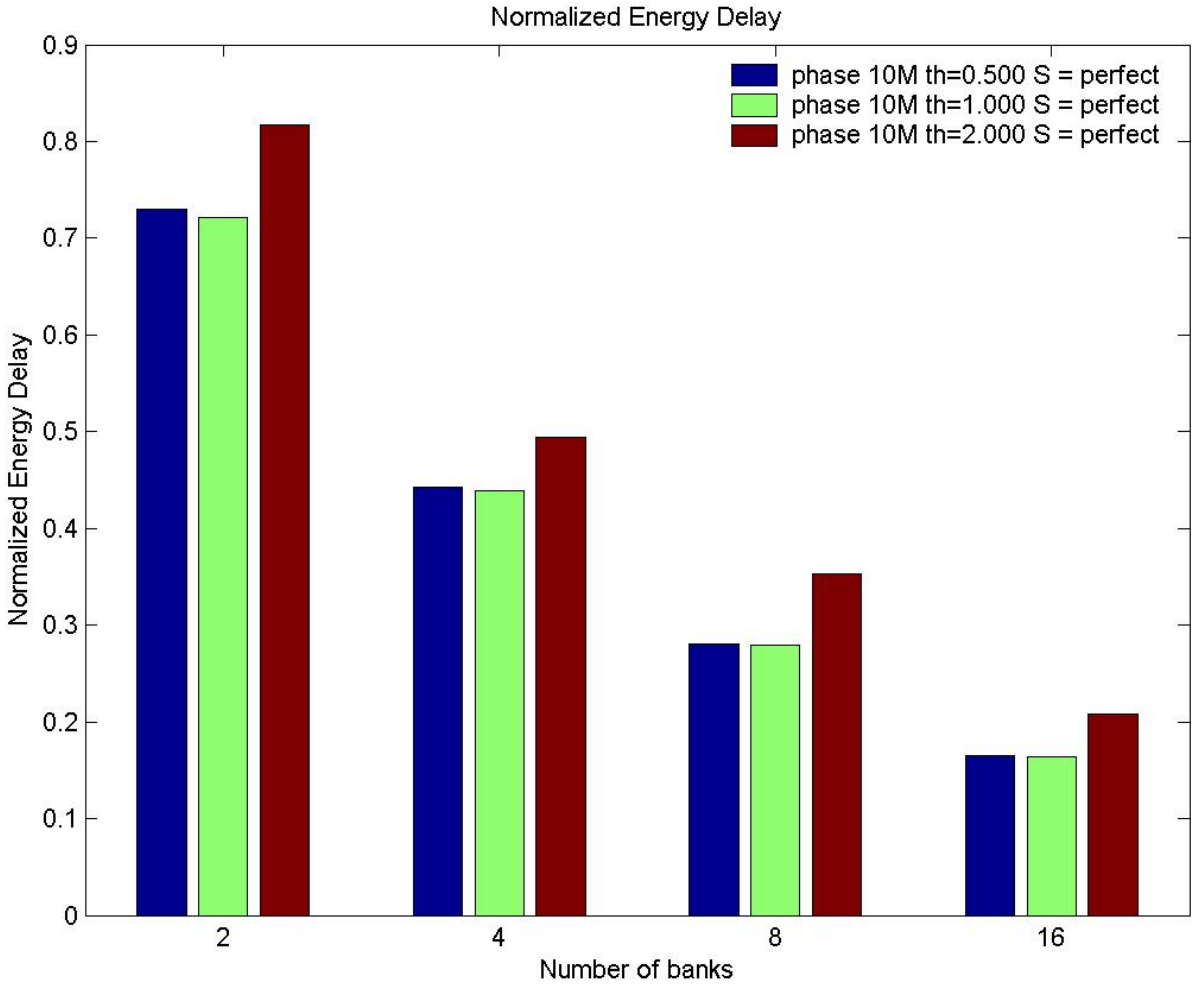
- Use performance counters:
  - Can be programmed to generate an interrupt on specified counts
- ISR provides matching with the meta data and mode changes
  - Every S*10,000 loop branches try a match
  - Phase matching can also be done in hardware
- Notify power manager to trigger proper action

Adaptation for Memory Behavior



- Number of engineering optimizations
  - Frequency of adaptation
  - Granularity of analysis (phase granularity)
  - Tradeoff against cost of adaptation.

Results – Normalized to NAP



Average among bzip, mpeg, ghostscript and ADPCM

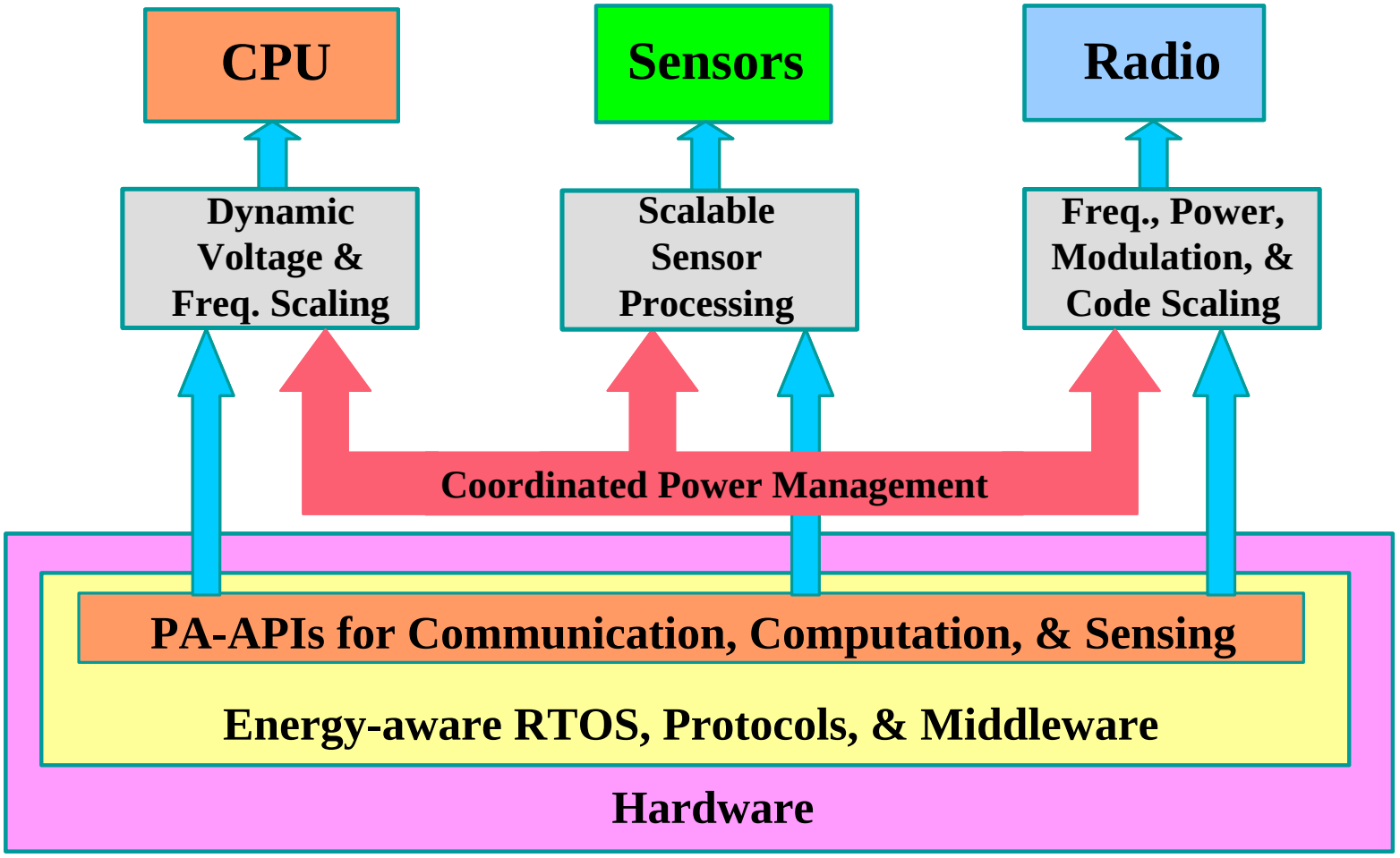
Results - overheads

- Approx. 350K instructions for every 10,000 loop branch instructions
- Number of instructions executed by the match algorithm at every 10,000 loop branches to match a partial signature (500 instructions per phase)

# of phases	# instructions	overhead
5	2,580	0.7%
10	4,500	1%
20	8,280	2%
30	12,060	3%

- Size overhead. 4 bytes per inter arrival estimate per bank / phase. 4 x 16 x 10 = 640 bytes assuming 16 banks and 10 phases.
- The signatures take1280 bytes for 10 phases. Total of 2KB of meta data

“Hardware” View



Going Forward: Multiple Radios Are Common

802.11x, BT, GSM

HP h6300: GSM/GPRS, BT, 802.11

Moto CN620: BT, 802.11, GSM

- These radios typically function as isolated air interfaces to isolated networks.

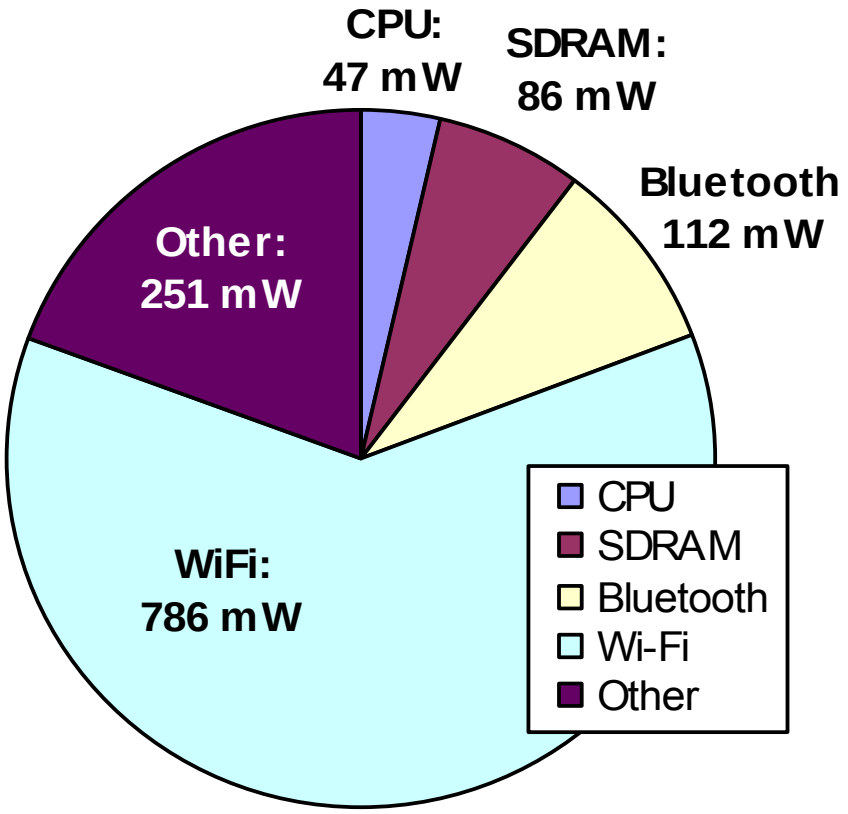
Collaborating Radios Can

- Improve Performance
  - Aggregate connectivity
- Improve Reliability
  - Radios as backup interfaces
- Improve Security
  - Multiple/Side-Channel Authentication
- Improve Efficiency (Spectral, Energy)
  - Dynamically match radios to traffic, range
  - Use radios to page another, duty cycle other radios

- Collaborating radios have a great potential for system-wide improvement
  - Energy, mobility management, capacity enhancement, channel failure recovery, networking, security, ….
  - We focus on energy.

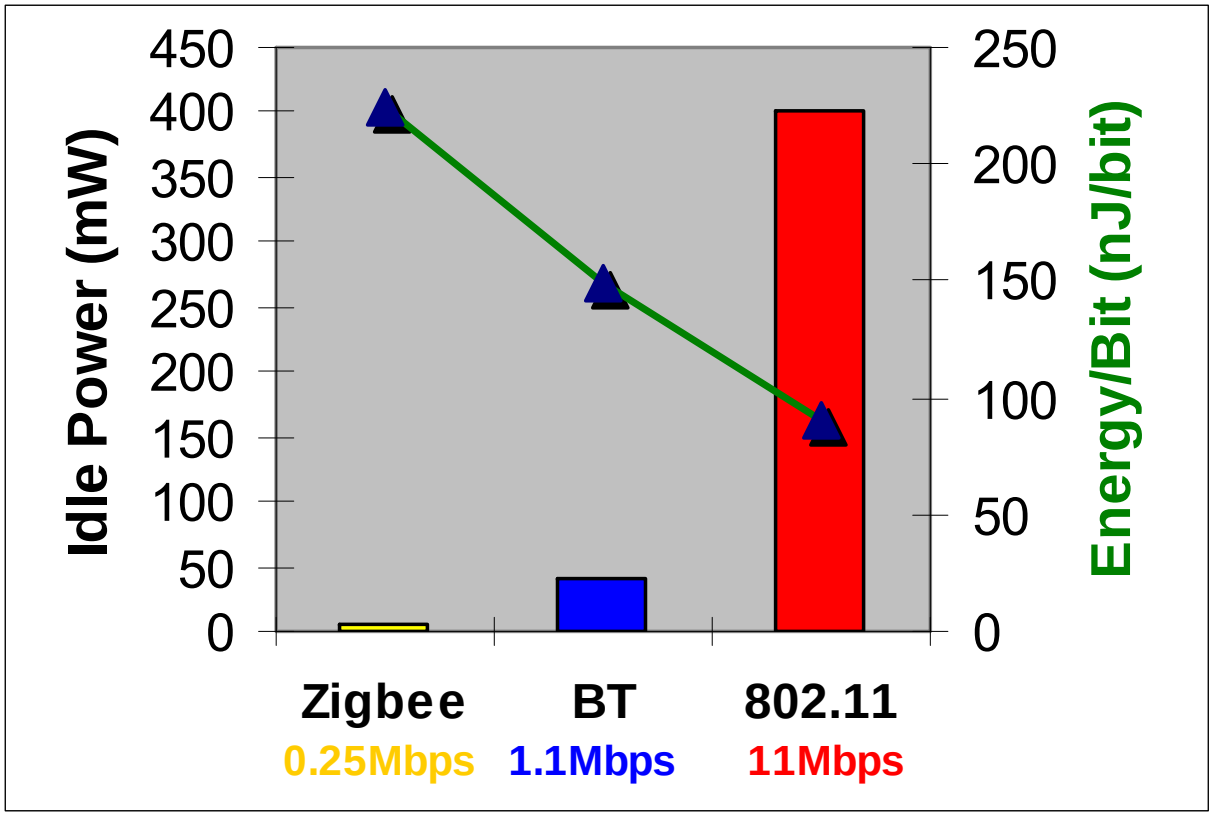
Typical power distribution

Depending on the usage model, the power consumption of emerging mobile devices can be easily dominated by the wireless interfaces!



Power breakdown for a *fully connected* mobile device in *idle* mode, with LCD screen and backlight turned off.

Common Radio Standards



Higher throughput radios have a **lower** energy/bit value

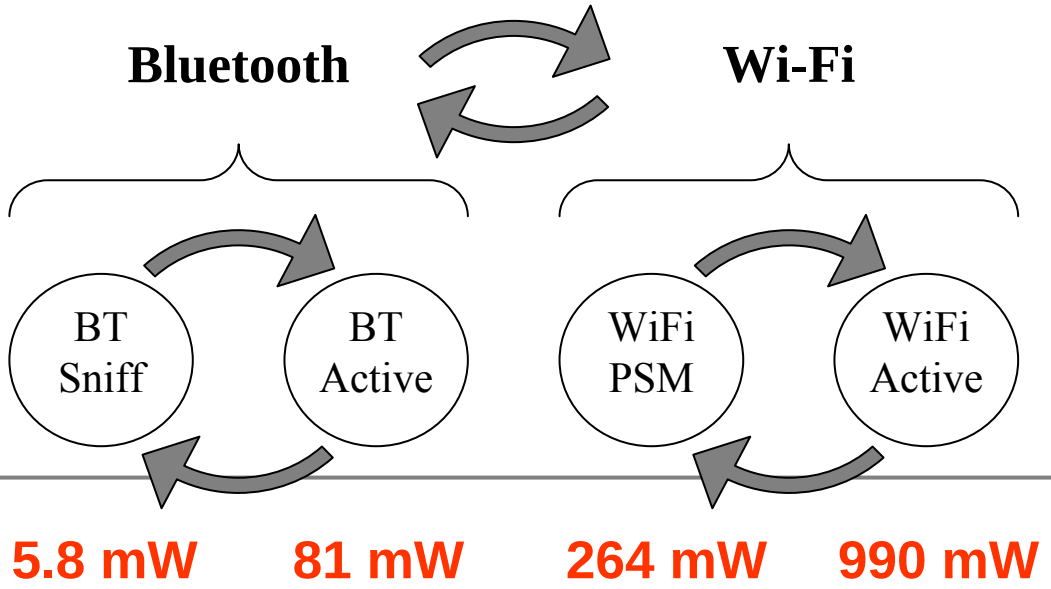
… have a **higher** idle power consumption

And they have different ranges.

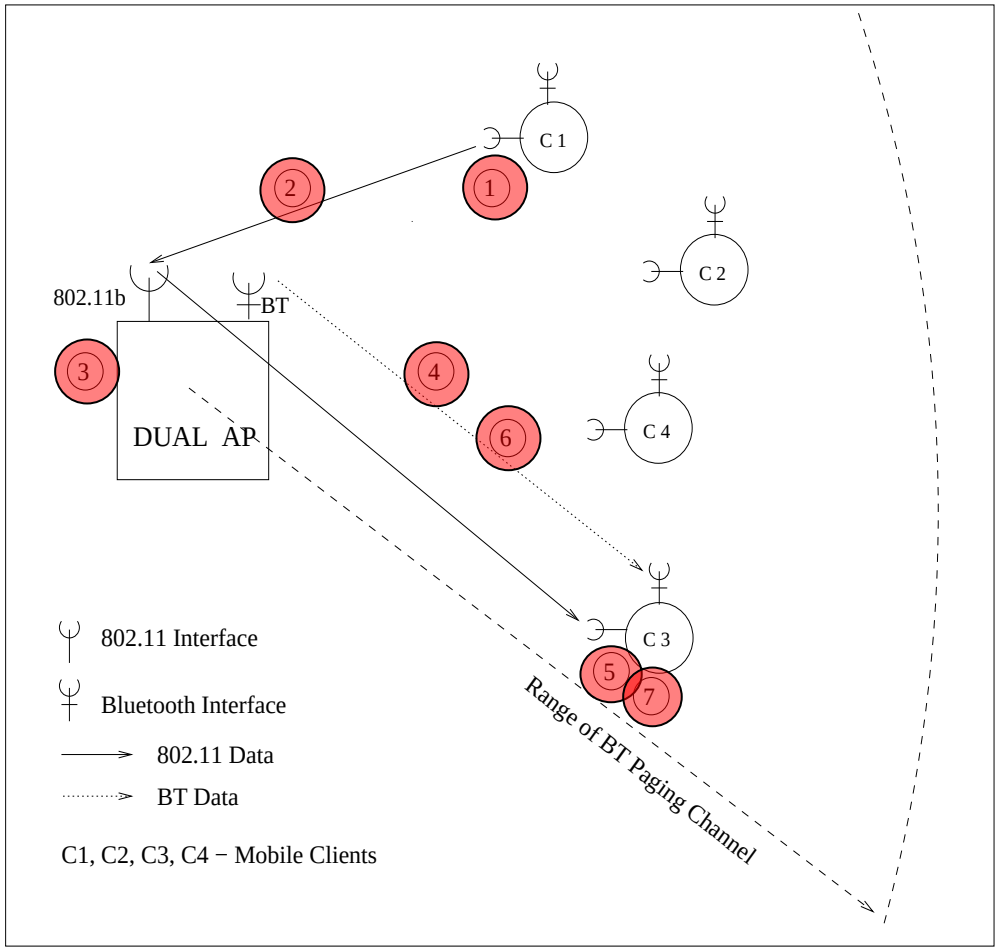
Consider: BT and WiFi

Objective: Always-on low-power operation with high peak bandwidth and overall energy efficiency

- Two possibilities:
  1. Use BT to page WiFi as needed
  2. Build a switching hierarchy for energy efficient operation
     - Effectively expand the power states available at the system level
     - Switching policies are key to a good implementation.



1. BT as a paging radio

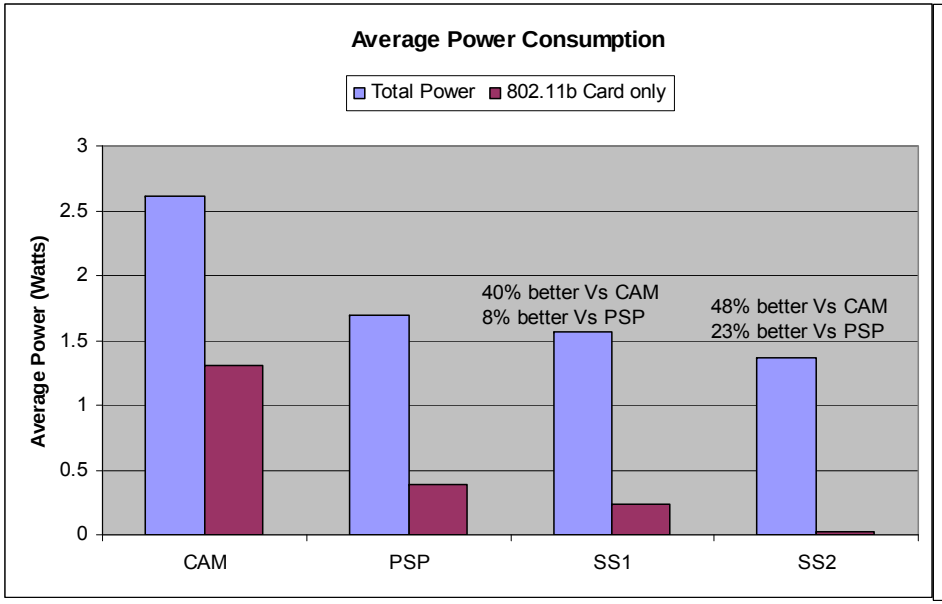


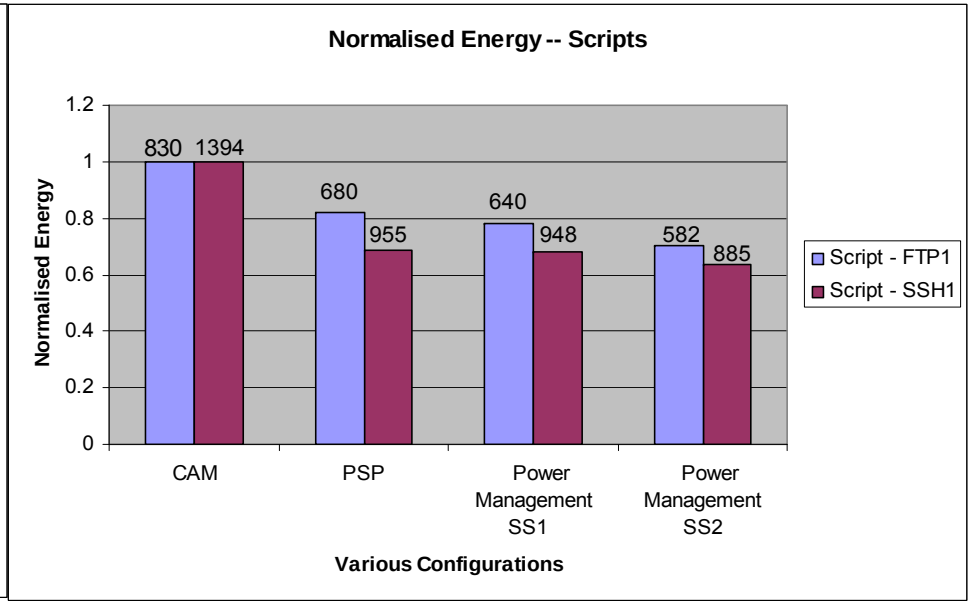
Scenario : An application on C1 wants to communicate with C3

1. C1 turns its 802.11 radio ON
2. C1 starts communication, sends data to AP through 802.11
3. AP matches C3's destination IP with its BT address
4. AP sends WAKE-UP page to C3 via it's BT interface, C3 turns on it's 802.11 radio on receiving the WAKE-UP page
5. When C1 finishes sending data it switches OFF its 802.11 radio
6. If all connections to and from C3 are closed, AP sends SLEEP page
7. On receiving SLEEP page C3 turns OFF its 802.11 radio

Simple paging (with range compensation)

- Implemented iPAQs (3870), familiar linux and CISCO PCM-350, built-in BT
- Measured power and latency on FTP and SSH sessions



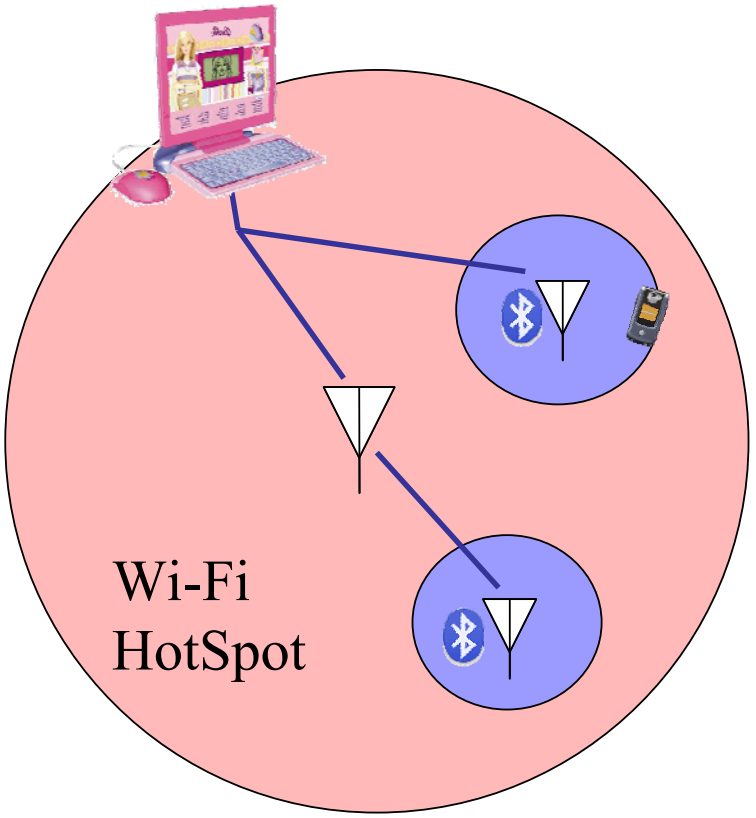


Power Savings for 802.11 card only vs PSP : 41% (SS1) to 95% (SS2)

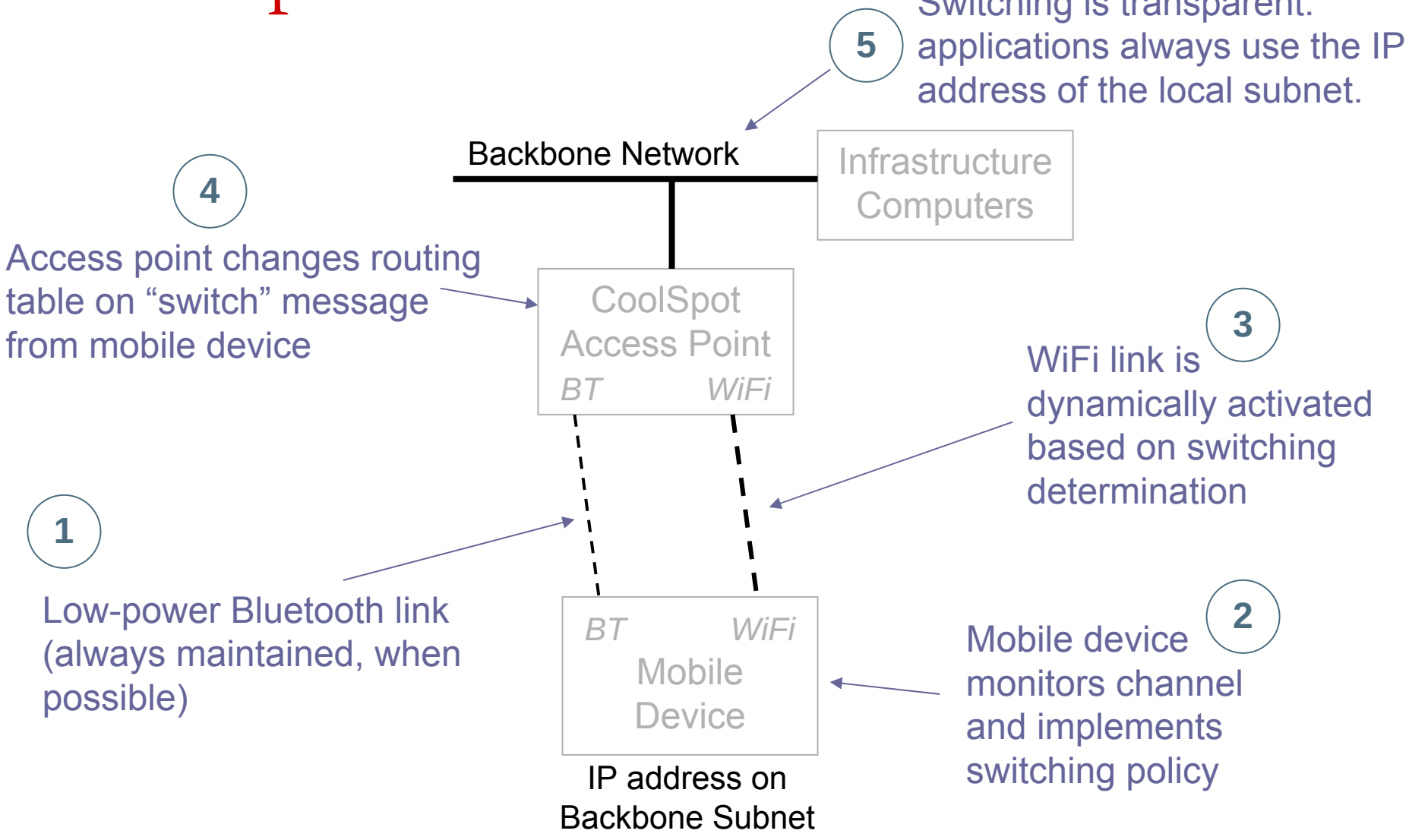
Throughput - Same as Awake Mode (CAM) , maximum throughput

Latency - Setup latency, amortized across session

2. CoolSpots: Radio Hierarchy



CoolSpots Network Architecture



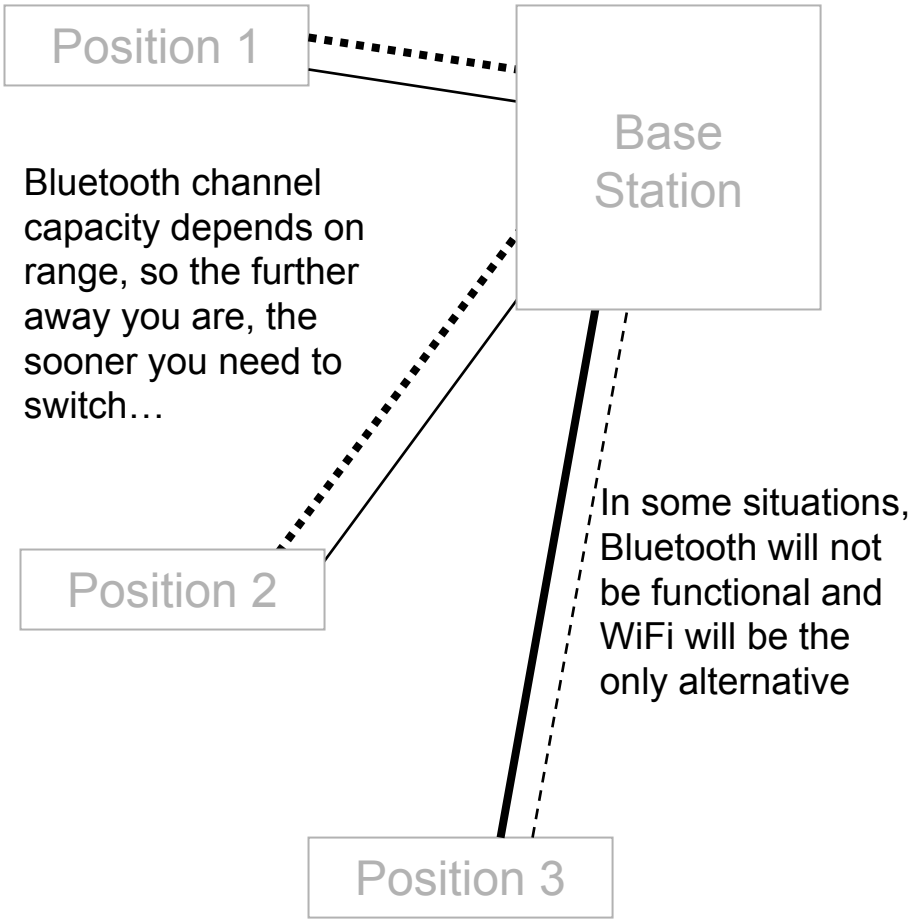
Switching Policies

- Three main components contribute to the behavior of a multi-radio system

- Position: Where you are
  - Need to address the difference in range between Bluetooth and WiFi
- Benchmarks: What you are doing
  - Application traffic patterns greatly affect underlying policies
- Policies: When to switch interfaces
  - A non-intrusive way to tell which interface to use

Where: Position

- Different radio ranges affect the switching decision
- However, optimal switching point will depend on exact operating conditions, not just range
- Experiments and (effective) policies will measure and take into account a variety of operating conditions



What: Benchmarks

Baseline: target underlying strengths of wireless technologies

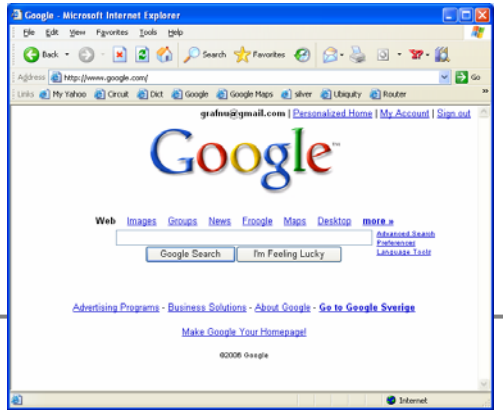
- Idle: connected, but no data transfer
- Transfer: bulk TCP data transfer

Video: range of streaming bit-rates varying video quality

- 128k, 250k, 384k datarates
- Streaming data, instant start

WWW: realistic combination of idle and data transfer conditions

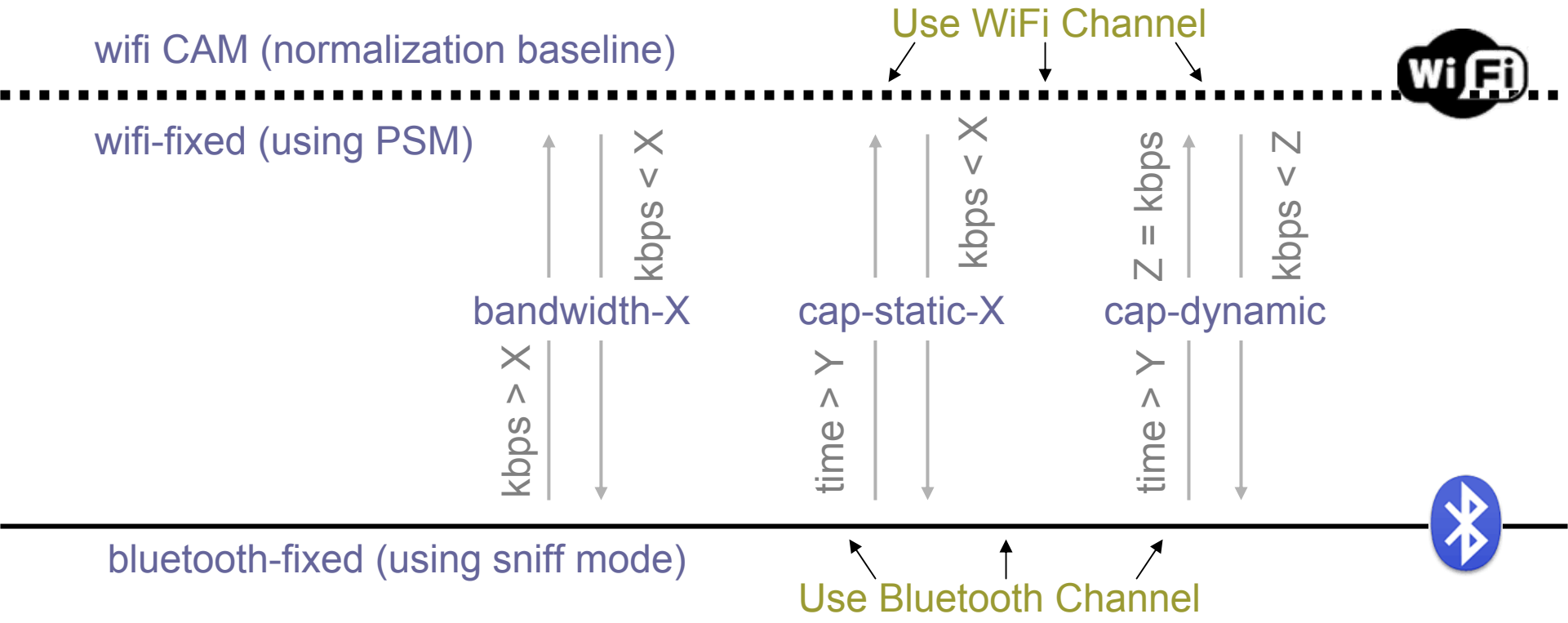
- Idle: “think time”
- Small transfer: basic web-pages
- Bulk transfer: documents or media



What: Benchmarks

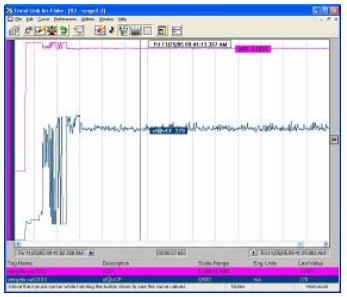
Benchmark	Time over WiFI	Data Transmitted	Average Bandwidth (Data Size / Time)	Data Pattern
idle	60s	0.0 MB	0 kbps	None
transfer-1	13s	6.6 MB	4482 kbps	Bulk transfer
transfer-2	27s	13.3 MB	4519 kbps	Bulk transfer
www-intel	176s	21.6 MB	1022 kbps	Intermittent data
www-gallery	150s	2.9 MB	158 kbps	Intermittent data
video150k	150s	3.1 MB	172 kbps	Real time streaming video
video250k	150s	7.3 MB	402 kbps	Real time streaming video
video384k	150s	8.5 MB	464 kbps	Real time streaming video

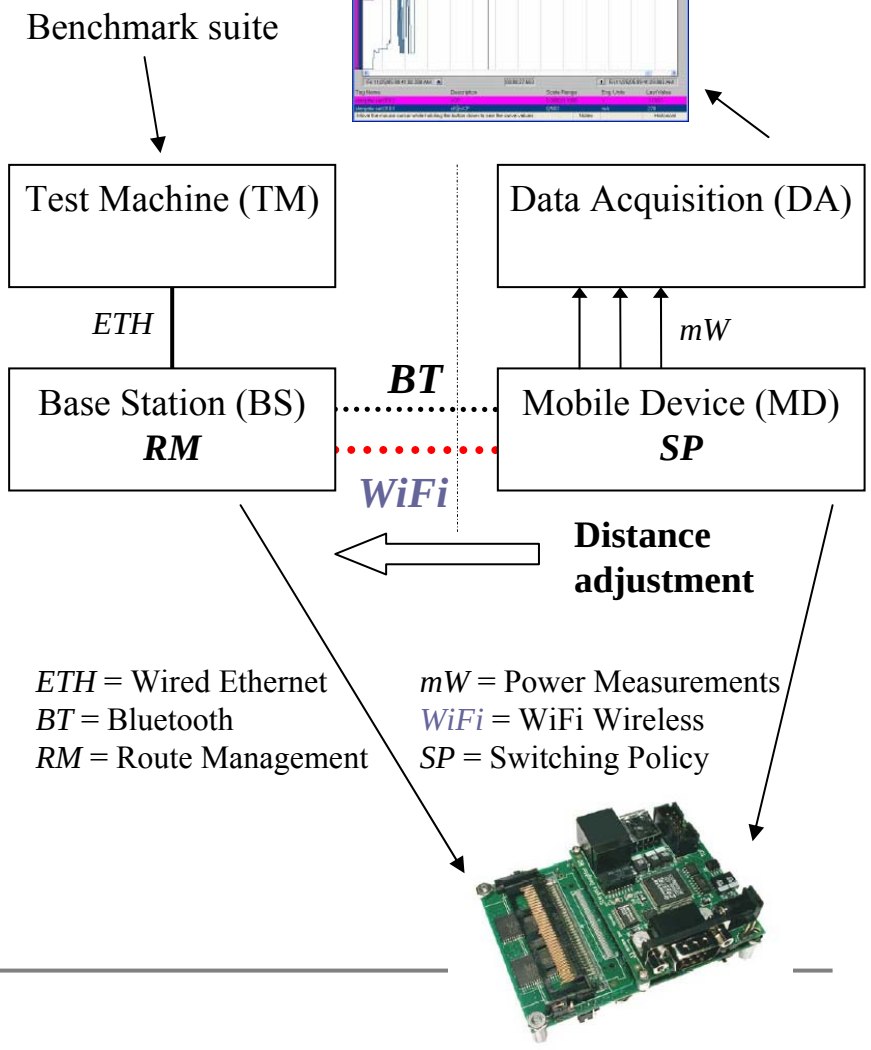
When: Policies



Experimental Setup

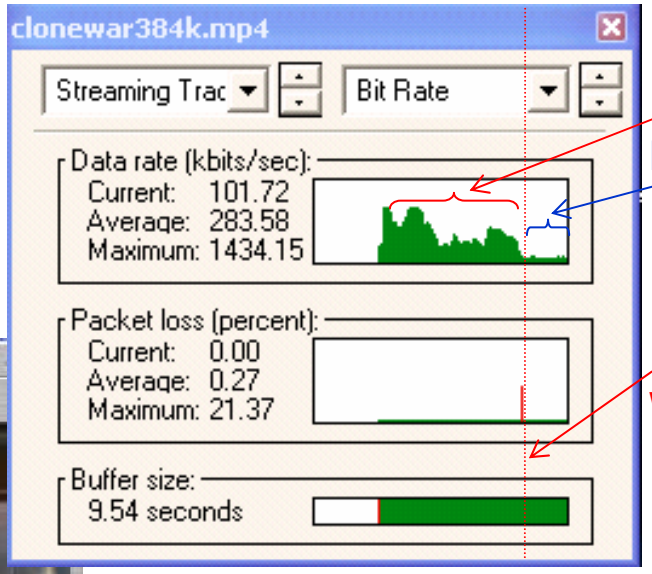
- Characterize power for WiFi & BT
  - Multiple Policies
  - Different locations
  - Suite of benchmark applications
- Stargate research platform
  - 400Mhz processor, 64MB RAM, Linux
  - Allows detailed power measurement
- Tested using “today’s” wireless:
  - WiFi is NetGear MA701 CF card
  - Bluetooth is a CSR BlueCore3 module
- Use the geometric mean to combine benchmarks into an aggregate result
- Moved devices around on a cart to vary channel characteristics





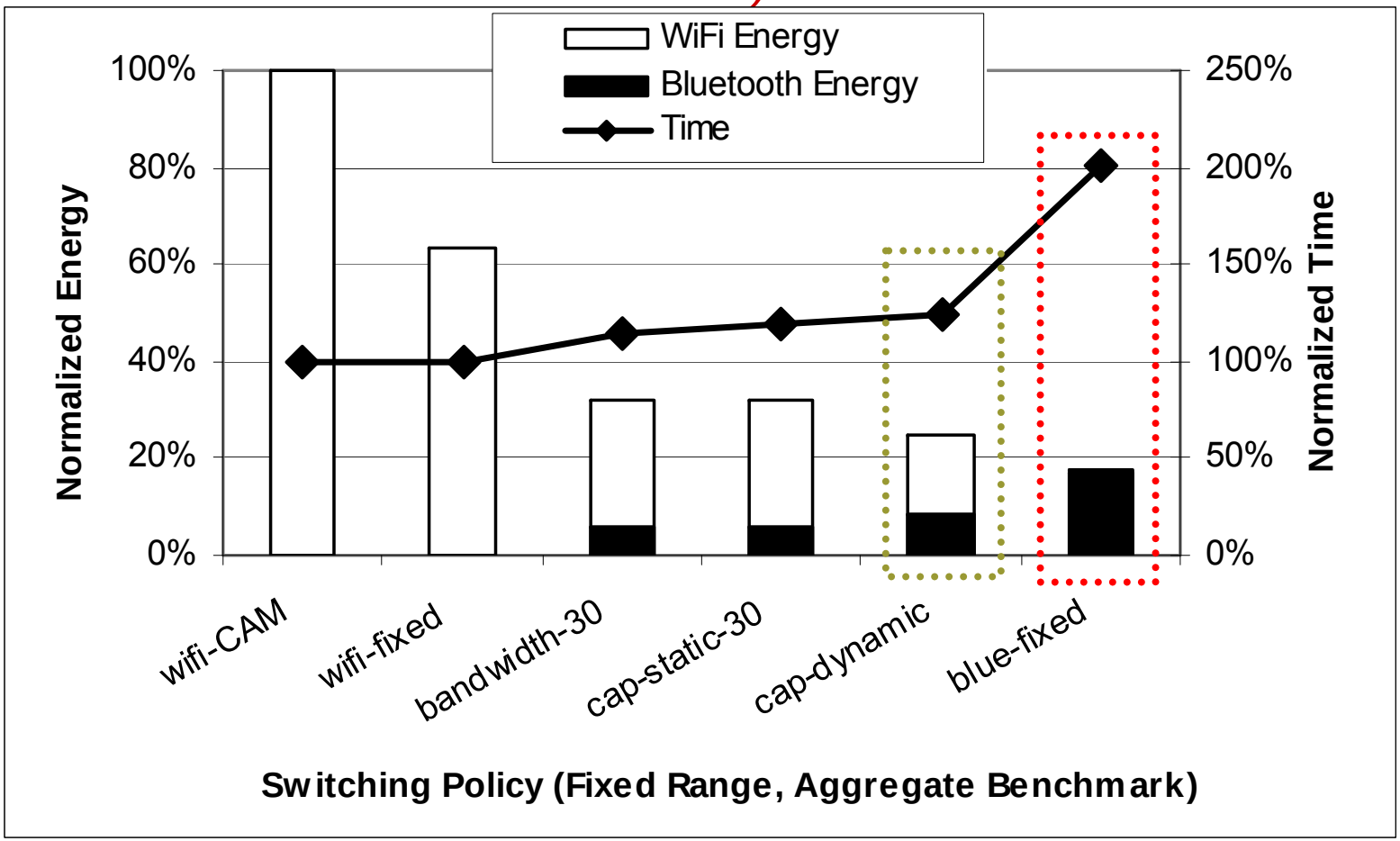
Switching Example: MPEG4 streaming

- Simple bandwidth policy
- Switch from WiFi to BT when application has buffered enough data



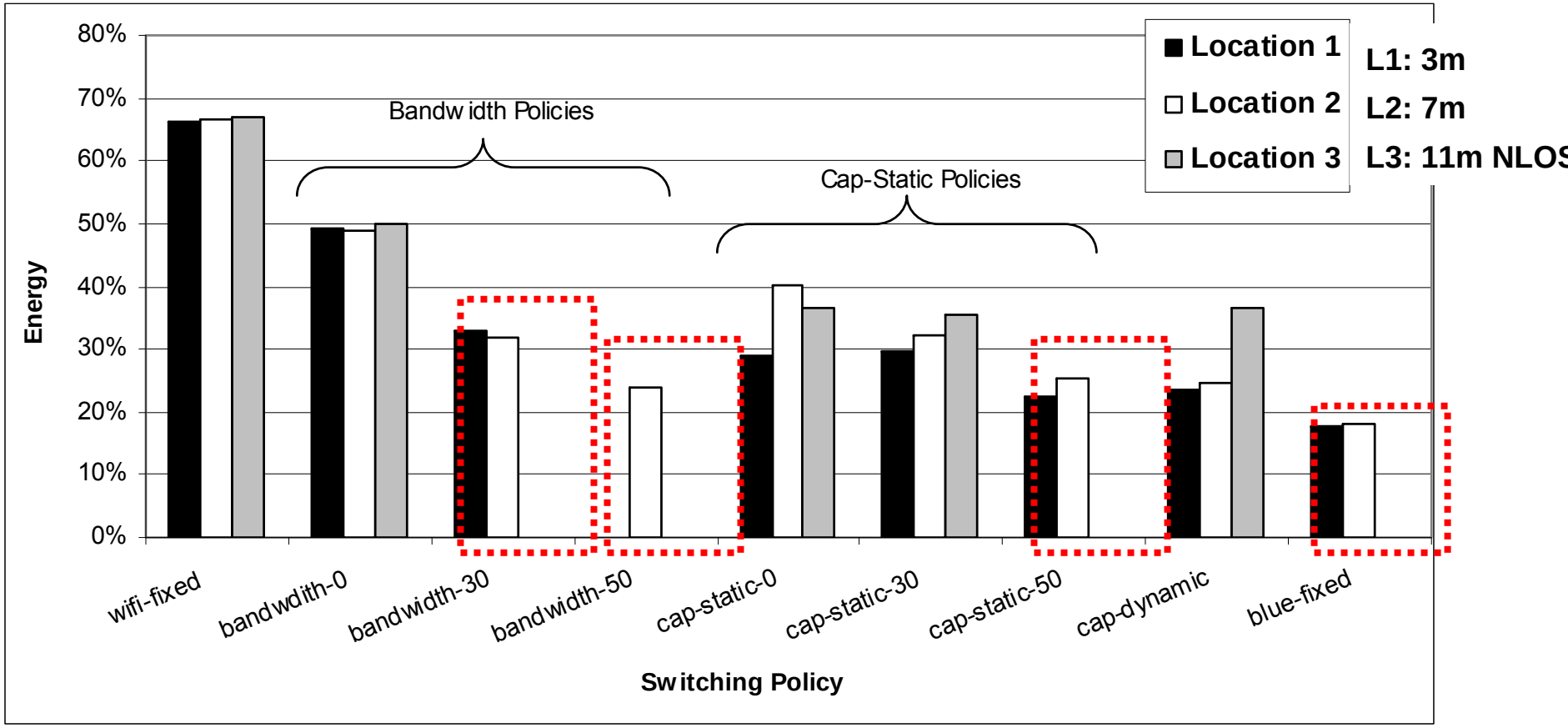
Switching is transparent to unmodified applications!

Results (Intermediate Location)



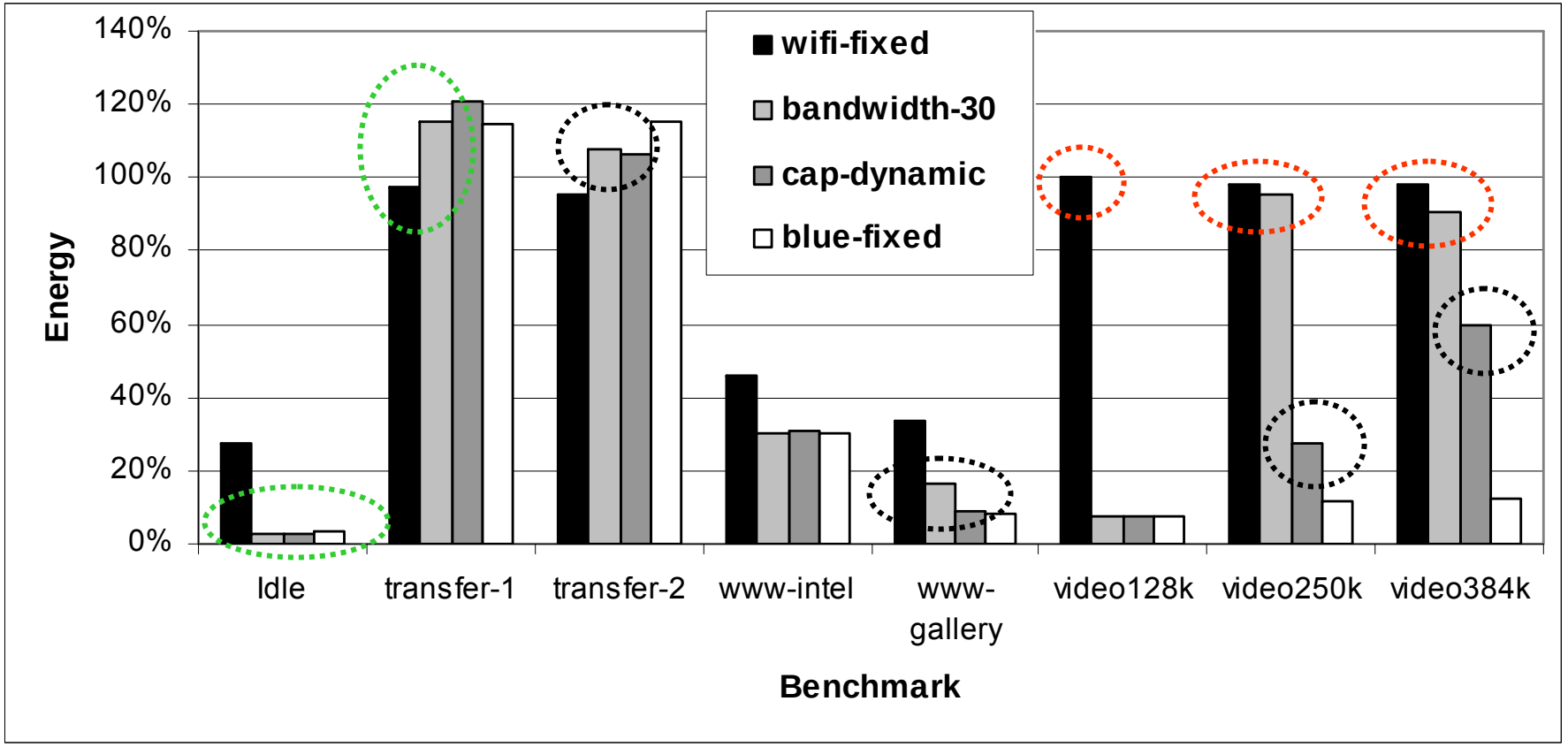
- ***blue-fixed*** **does well in terms of energy but at the cost of increased latency**
- ***cap-dynamic*** **does well in terms of both energy and increased latency**

Impact of Range/Distance



Missing data indicates failure of at least one application, and therefore an ineffective policy!

Results across various benchmarks



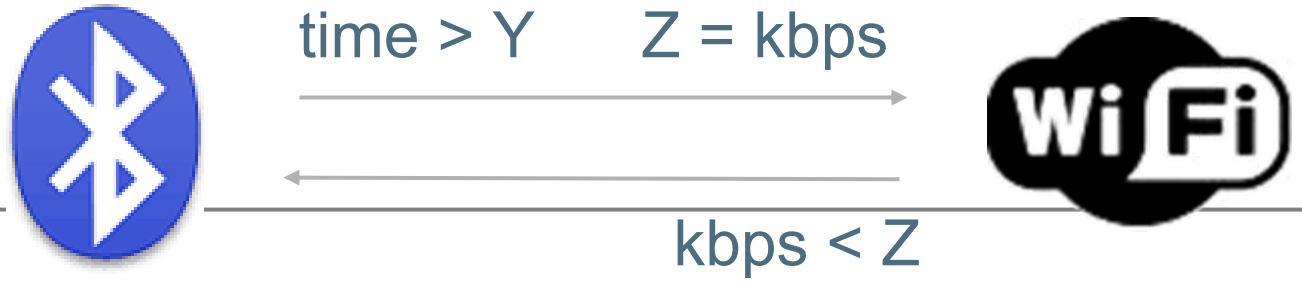
wifi-fixed **consumes lowest energy for data transfer,** ***any bluetooth policy*** **for idle**

- **Overall,** ***cap-dynamic*** **does well taking into account energy and latency**

Video benchmarks really highlight problems with ***wifi-fixed*** **and** ***bandwidth-x***

Cap-Dynamic Switching Policy

- Switch up based on measured channel capacity
  - (ping time > Y): 40ms-800ms, estimates channel conditions
- Remember last seen Bluetooth bandwidth
  - (Z=kbps)
- Switch down based on remembered bandwidth
  - (kbps < Z): limited mobility



Switching Policies – Summary

- "Wifi-Fixed" Policy (WiFi in Power Save Mode)
  - Works best for as-fast-as-you-can data transfer
  - Higher power consumption, especially idle power
- "Blue-Fixed" Policy
  - Very low idle power consumption
  - Increases total application latency, fails at longer ranges
- "Bandwidth" Policy
  - Static coded bandwidth thresholds, fails to adapt at longer ranges
  - Switches too soon (bandwidth-0) or switches too late (bandwidth-50)
- "Capacity-Static" Policy
  - Estimates channel capacity and uses that to switch up
  - Fails at longer ranges due to incorrect switch-down point
- "Capacity-Dynamic" Policy
  - Dynamic policy, remembers the last seem switch-up bandwidth
  - Performs well across all benchmarks and location configurations!

Closing Thoughts

- Algorithmic approaches to power management have come a long way
  - Realistic models of the system, and its environment
  - The challenge remains how a “good” PM is actually implemented
- Going forward, a comprehensive “awareness” of energy is needed from application to distributed hw/sw infrastructure
- Multiple radios open up many possibilities for system-level performance and reliability increases
  - CoolSpots shows ~50% reduction in energy consumption over current power management in WiFi across applications, ranges
- Many improvements possible that take into account
  - Application behavior, Radio link quality, Network queues instead of ping latency, other scenarios (multi-user environments, p2p configurations)
  - Network infrastructure instead of standalone CoolSpots APs