Power Management and Environmental Awareness

Anthony D. Joseph
CS 294-1

Lecture 12
November 15, 2000

Outline

- Scribe?
- Project checkpoints:
  - 15 minutes: Friday 9-11, Tuesday 9-11
  - See Helen to schedule times
- Douglis, et al, Storage Alternatives for Mobile Computers
- Stemm, Katz, Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices
- Czerwinski, et al, An Architecture for a Secure Service Discovery Service

Storage Alternatives for Mobile Computers [Douglis94]

- Motivation
  - Lower storage power usage important as storage devices represent up to 40% of power usage
  - Batteries are a very limited resource
- Analyze 3 technologies for power
  - Quantitative comparison of measurements
  - Analysis of techniques for improvements

Three Technologies

- Magnetic disk
  - Traditional
- Flash memory disk emulator
  - Block-based interface (looks like a disk)
- Flash disk file system
  - Byte-based interface (looks like NVRAM)

Magnetic Disk Considerations

- Good throughput for large writes
  - Especially for logging
- Very cheap
  - $1 – 5 MB then
  - $0.006 – 0.024 now
- But
  - They consume a lot of power
  - Disk "spin-up" has high latency, affecting performance

Flash Memory Considerations

- Pros
  - Noiseless, considerably lighter, faster reads, consume less power than disk drives
  - Much more rugged than disk drives
    - Can withstand > 1,000 G's of operating shock
    - Equivalent to dropping a flash card 8 feet to the floor
  - Throughput & latency good for reads
- Cons
  - Writes are very slow (must erase before writing)
  - Costly: $30 – 50 MB then
    - $1.87 – 2 MB now (versus 1.48 – 1.72 for DRAM!)
  - Limited endurance
    - "burn-out" after 100K writes then, 300K – 1M writes now

From Manufacturers' Statistics

- Disk spin-up takes 1 second
- Flash card erase takes 1.6 second
- Disk uses an order of magnitude more power
- Both Flash devices use same power

Lots of Traces

- Used several different machine's workloads
- Recorded
  - Filename, Read or Write
  - Offset, Size, Time
- No deletions
- Run through simulator

Raw Hardware Measurements

- Software benchmarks: R/W files
  - Sequential (throughput)
  - Random(latency)
  - Compressed & uncompressed
- Pre-erased/Non-pre-erased
- Spun up/spun down
- With and without DRAM Buffer

Measurement Results

- Disk provides best overall throughput
- Log writes are bad on disk due to Microsoft Fast File System "anomaly"
- Flash card is better than Flash disk for small files
  - Caused by byte access interface?
- Flash card worse than Flash disk for large files
  - Caused by a minimum amount of erasing that still has to be done ?

Simulator

- Constants
  - Assumed buffer cache and write-through policy
  - When NV storage is disk, assumes there is an SRAM intermediate level cache
- Computed
  - Power consumption in each mode
    - Reading, writing, sleeping, idle
  - Time to perform operations
  - Time to switch between modes

Five Experiments

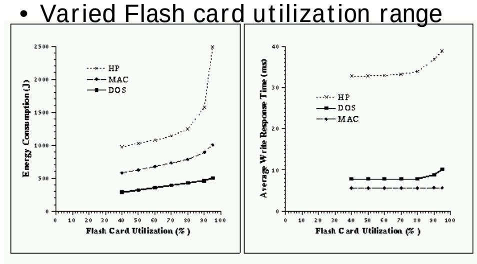
- Basic raw data comparison
- Storage utilization on flash card
- Decoupled erasures/writes on flash disk
- DRAM caching on flash
- NVRAM caching

#1: Basic Raw Data

- From datasheet: Flash card write worse than disk write
  - Concluded: perhaps SRAM would help
- From datasheet: Flash card writes faster than flash disk
- But, measured: Flash disk better!
  - Have to consider hardware/software characteristics

#2: Flash Card Storage Utilization

- Varied Flash card utilization range



(d) Energy consumption. (c) Response time.

Utilization 40 → 90%: Write response increases by 30%
Energy usage up by 150%!

#2: Storage Utilization on Flash Card

- Varied flash card utilization range
- As utilization went up…
  - More erasures per unit time (.9 to 1.9 mean)
- Why?
  - Must copy live data to free segment
- Why not in flash disk?
  - Flash disk uses 512byte chunks versus flash card's 1 or 2 64kbyte chunks

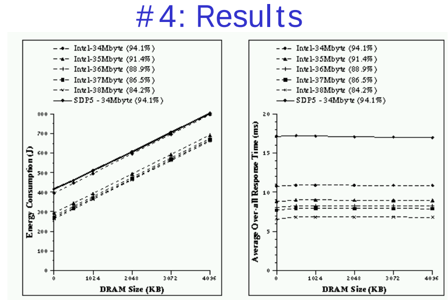
#3: Decoupled Erasure / Writes on Flash Disk

- Can do this in the future
  - Erase/write bandwidth: 150KB/s
  - If decoupled (writing pre-erased areas: 400KB/s)
- Next-generation (today)
  - Intelligent controllers can pre-erase
  - Also, *Write without Erase* and *Erase Sector* commands
- Results: Minimal impact on power!!
  - Decreases average write 56-61%!!
  - Concluded: Good idea!
    - Makes flash disk write as fast as flash card write
    - Flash disk better choice as utilization increases

#4: DRAM Caching for Flash Devices

- DRAM vs. Flash
  - DRAM has better performance
  - DRAM is volatile & consumes more power
- Tradeoff: adding more flash vs. DRAM
  - Would like to keep hot data in DRAM
    - But, otherwise use Flash to preserve power
  - Experiment varied:
    - Size of DRAM buffer cache
    - Flash disk utilization

#4: Results



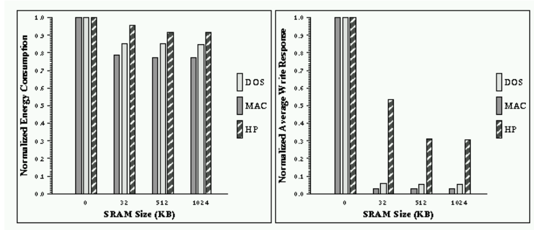
(a) Energy consumption as a function of DRAM size and flash size. (b) Response time as a function of DRAM size and flash size.

- No effect on response time
- Increases power usage, but no appreciable benefits

#5: NVRAM Caching

- Tradeoff:
  - SRAM uses a lot of energy
  - But, reduces spin-ups ➔ saves energy
- Examine impact of NVRAM on
  - Write performance, deferring spin-up
- Simulation: place NVRAM between disk and buffer cache

#5: Results



(a) Energy consumption. (b) Response time.

- Energy goes down slightly
- Response time goes down dramatically
- Conclusion: 32kB NVRAM cache is a good thing!

Conclusions

- Flash is an attractive alternative to magnetic disks
  - Lower power
  - Good read performance (3-6X faster)
  - "Acceptable write performance" (4X slower)
- Disk: power hog
  - Order of magnitude more than flash
- SRAM helps Flash
- Decouple erasures from writes
- Use small erasure segments

5-minute Break

Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices [Stemm, Katz]

- Motivation
  - Next generation devices
    - Seamless connectivity
    - Stringent power & size constraints
- But, network interface accounts for significant portion of power usage
  - Affects lifetime

Goal:Measure Network Interface Power Usage

- Power consumption data:
  - 2 PDAs & 4 network interfaces
- Definitions:
  - Idle state: device powered on, but not doing anything
  - Sleeping state:
    - Most circuits in the device are turned off
    - Can detect only simple events like an interrupt and takes a long time to come back up

Major Issue

- Surprise!
  - Network interface power ≥ PDA idle power
  - Moore's law only makes the problem worse!

➔ "Clear indication that power management of network interface is essential"

Goal: Gather Data for Network Interfaces

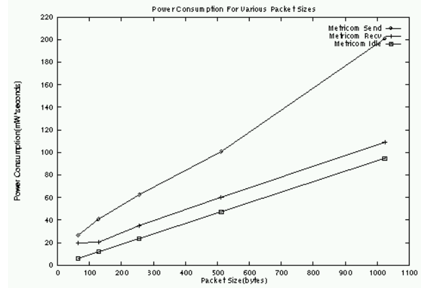


Figure 2. Energy consumption for different packet sizes for Metricom

Measurement Conclusions

- Receive state power consumptions is almost same as in Idle state
  - For both WaveLAN & Metricom
- Send state power consumption is almost the same as in Receive state for WaveLAN
  - But for Metricom, its considerably higher
- Hence, protocols should minimize packets sent from the mobile device
  - Good news: most mobile apps already do this

Part 2:Transport Layer Simulation

- Compared 4 protocols:
  - TCP Reno
    - ACK for every packet
  - TCP Reno with delayed ACKs
    - ACK every other
  - Reliable UDP
    - Fixed size window, ACK per w packets
  - Reliable UDP, unlimited window
    - ACK per transfer

Simulation

- Used ns
- Looked at
  - # of ACKs generated
  - # of packets they sent
  - Amount of time necessary to do transfer
    - Important for IDLE time power usage

SendRecv Energy Cost

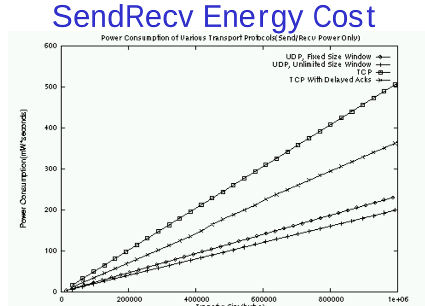


Figure 3. Energy for different transport protocols only including *SendRecv*

Measurement Conclusions

- Protocols with fewer ACKs are better
  - Sending takes more energy
  - UDP with unlimited window size is best
- But, other protocols also perform well enough if only SendRecv energy is considered
- BUT, energy equation really: SendRecv + idle!!

Accounting for IDLE power use

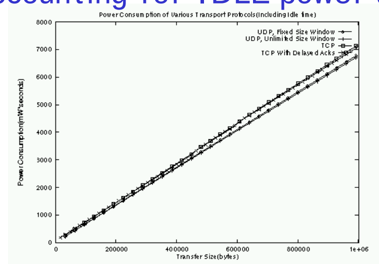


Figure 4. Energy for different transport protocols including *SendRecv* and *Idle*

- Now all the protocols about the same
- IDLE COST DOMINATES

Varying Error Rates

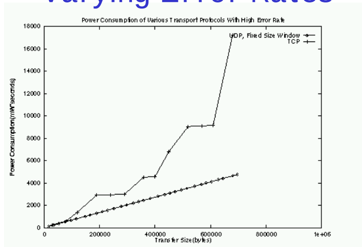


Figure 5. The effect of wireless losses on energy consumption

- TCP mistakes losses for congestion ➔ reduces transmission rate
- Transmission time goes up ➔ more idle time power usage!
- Lesson: protocols should be smarter

Lessons Learned So Far…

- Dominant energy cost is amount of transfer time because idle time uses power
- Wireless link error rate can increase energy costs significantly
  - But, depends on the protocols

Part 3 - Email Improvements

- They propose an e-mail reading program that wakes up periodically & checks for e-mail by contacting a mail host
  - The waiting period is the attention span (inverse of expected?)
- Evaluation of strategy's effectiveness
  - Traces of e-mail arrival times & sizes fed into a simulator
  - Trade-off between energy consumption and e-mail staleness with varying attention spans
- Trace Data
  - Arrival time, size of mail collected
  - Varied "attention spans"
    - Amount of time between waking to check for mail
- Measured
  - Average energy consumption
  - Average staleness (penalty for saving energy)

Email Simulations

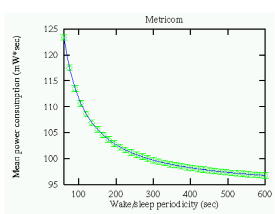


Figure 6. Energy vs. attention span for Metricom

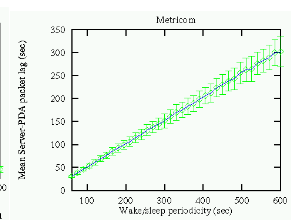


Figure 7. Staleness vs. Attention Span for Metricom

- A little staleness at beginning gives big savings
- Starts to flatten at 300s, marginal savings not as good
- 2 minutes staleness: 125->100 mW/s (20% savings)

Part 4: Web Browsing Improvements

- Collected trace data as above
  - Measuring times & sizes of web transfers
- Proposed scheme:
  - Device alternates between think time & sleep time
  - After completion of a transfer, network interface kept on for attention span interval
    - Note: This attention span has an intuitive meaning opposite to e-mail app's attention span

Simulations

- Trade-off between power-savings & user-perceived latency
- Trace data
  - HTTP traffic
  - “Think time” vs. “work time”
- Strategy:
  - Reduce power usage during think time
  - Turn off network interface when user has been in think phase past “attention span”
- Vary attention span in simulation

Web Simulations

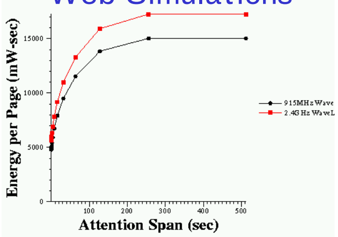


Figure 8. Energy per page as a function of Attention Span for Wavelan NIs

- Bigger attention span ➔ bigger power usage
- BUT, tension between attention span and response time
- Wrong! Even on Metricom (w/ high wakeup cost), can reduce 30% with almost no user-visible effect

Conclusions

- Total transfer time counts!
  - Protocols should not leave a mobile idle
- Application level strategies can help
  - Opportunities exist to reduce power with no user-visible effect

Service Discovery Service [Czerwinski99]

- Problem
  - Clients want to
    - Securely search for available services by attributes
    - Discover the interface to the service
    - Run services in the infrastructure
  - Clusters want to
    - Securely advertise their running services
- Solution
  - Implement a “Service Discovery Service” as a directory of this information
  - Clients and clusters need only bootstrap to an SDS server

Directory Services

- Thus, the SDS is itself a **directory service**
- We can compare/contrast with existing directory services:
  - Grapevine/Clearinghouse, DNS, SAP+SDP, Globe, SLP

Various Axes of Differentiation

- Query structure
  - Hierarchy reflected in query (DNS, globe)
  - Query grammar complexity (LDAP vs. DNS)
- Service data
  - Reference (globe) vs. content (SAP+SDP)
  - soft state (SAP) versus hard state (DNS)
- Hierarchy configuration
  - Manual/static vs. dynamic and adaptive
  - Response to a node failure

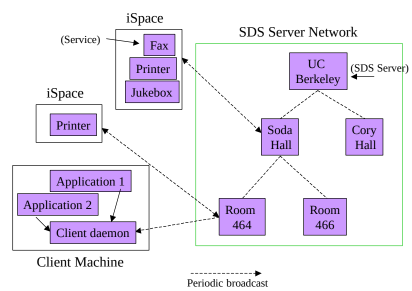
Directory Issues (cont'd)

- Update rate
  - Short cache timeouts needed for mobility
  - Lower rate via longer caching a key to scalability
- Push (advertisements) versus pull (queries)
  - DNS - pull only
  - SAP - push only (modulo caching)

Bootstrapping

- "Well-known" local name ("www.")
- List of unicast addresses (DNS)
- Listen for local broadcasts (Mobile-IP)
- Well-known global/local multicast address (SAP)
- Expanding ring search

Overview of Architecture



Design Points

- Server hierarchy based on administrative domains
  - Experimenting with self-adaptive partitioning & other aggregation strategies
  - Loose coupling to location
  - Utilize spatial locality to minimize server crossings
  - Children's domain a partition of parent's
  - Scales with client population & number of entries
  - Tolerant to node faults & dynamic changes to server pool
- Advertising & bootstrap through local broadcast

Design Points (cont'd)

- Service data
  - A set of extensible XML tags
    - Minimal predefined set (address specification, name, and interface)
    - Other tags are meta-data... i.e., location
- Queries
  - Under-specified service descriptions
  - Query types: LOCAL, UPWARD
  - Query results: BEST, LIST
    - Implies notion of ranking matches

SDS Server

- Acts as service aggregator
  - Stores/caches service information gained from listening to broadcasts
- Handles client's needs
  - Advertises existence through broadcasts
  - Fulfills queries using cache and other servers
- Maintains server hierarchy
  - Child provides heartbeat to parent
  - Negotiates partitioning with other servers

Ninja iSpace Environment

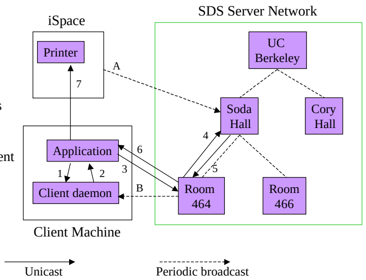
- Per-machine platform for running services
  - Executes and monitors a set of services
  - Restarts service for fault-tolerance
  - Able to spawn new services
- Provides minimal service advertisement functionality
  - Advertises each service through broadcasts
  - Creates service description using introspection plus defaults

Client-side Components

- Per-machine SDS client daemon
  - Listens to SDS server broadcasts
  - Caches server location info for client applications
  - Notifies SDS servers of overlap
- Client applications
  - Contacts daemon to learn server location
  - Directly contacts server to query
  - Future Work: could forward queries through daemon to allow client-side caching

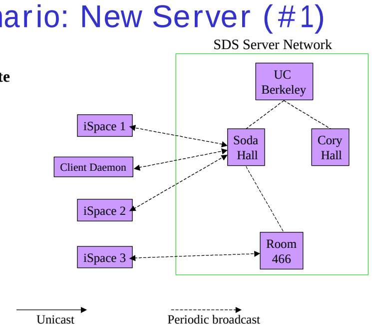
Scenario: Client Request

A. Service broadcast
B. Server broadcast
1. Local RMI to daemon to get server address
2. Return of server address
3. SDS_Query from client
4. Pass query to parent
5. Receive result from parent
6. Return result to client
7. Client uses service



Scenario: New Server (#1)

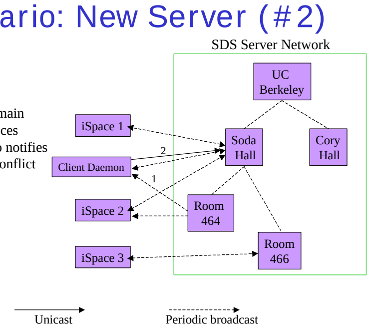
Initial steady-state



Scenario: New Server (#2)

New server joins

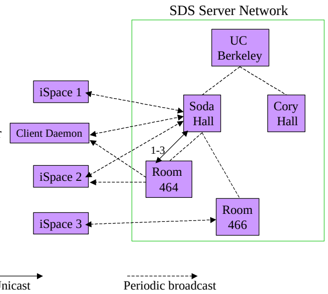
1. New server begins broadcasting to domain
2. Client daemon notices multiple servers, so notifies original server of conflict



Scenario: New Server (#3)

Domain negotiation

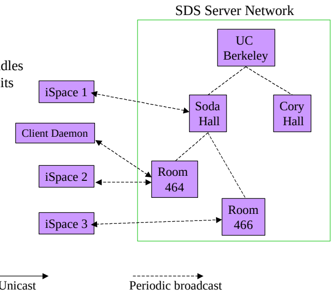
1. Affected SDS servers exchange coverage areas
2. One server determines partition, and sends information to other server
3. Soda Hall is set to be Room 464's parent



Scenario: New Server (#4)

Final steady-state

Room 464 server now handles the clients and services in its domain



Status of Implementation

- Uses Java RMI and UDP sockets
- First cut
  - Partition and hierarchy uses manually specified subnet addresses
  - Queries resolved by comparing XML tags based on string matching plus wildcards; need to add relational operators
  - No caching scheme
  - Servers do not negotiate domains

Discussion

- Caching / Partitioning scheme
  - When should a SDS server cache another server's information?
  - How should we handle cache invalidation?
  - Similar issue with client-side daemon cache
- Ranking
- Queries
  - Should clients be able to explore downwards?
- Management of user accessible SDS servers
- Security
  - How do we maintain/use private information?
  - How do servers authenticate one another?