

Energy Efficient Network Memory for Ubiquitous Devices

Joshua B. Fryman, Chad M. Huneycutt, Hsien-Hsin S. Lee
Kenneth M. Mackenzie[†], David E. Schimmel
Center for Experimental Research in Computer Systems (CERCS)
Georgia Institute of Technology
Atlanta, GA 30332-0280

{ fryman, chadh, leehs, kenmac, schimmel }@cercs.gatech.edu

[†]Current affiliation: Reservoir Labs, New York, NY.

Abstract

This article explores the energy and delay trade-offs that occur when some or all of the local storage is moved out of the embedded device, and into a remote network server. We demonstrate that using the network to access remote storage in lieu of local DRAM results in significant power savings. Mobile applications continually demand additional memory, with traditional designs increasing DRAM to address this problem. Modern devices also incorporate low-power network links to support connected ubiquitous environments. Engineers then attempt to minimize utilization of the network due to its perceived large power consumption. This perception is misleading. For 1KB application “pages,” network memory is more power efficient than one 2MB DRAM part when the mean time between page transfers exceeds 690ms. During each page transfer the application delay to the user is only 16ms.

1. INTRODUCTION

This article explores the energy and delay trade-offs that occur when some or all of the local storage is moved out of the embedded device, and into a remote network server. Contrary to designer intuition, we demonstrate that this can be more power efficient than local storage.

Embedded consumer systems continue to add more features while shrinking their physical device size. Current 2.5/3G cell phones incorporate 144kbps or better network links, offering customers not only phone services but also e-mail, web surfing, digital camera features, and video on demand. With feature expansion demanding additional storage and memory in all computing devices, densities of DRAM and Flash are increasing in an attempt to keep pace. This continuous storage expansion translates to a growing power dissipation, temperature, and battery drain.

To reduce energy effects and increase battery life, designers use the smallest parts and lowest part count possible. This has the added benefit of keeping manufacturing costs down. This effort to minimize available resources works against application feature expansion and device flexibility for dynamic upgrades.

In an attempt to address some of these problems, companies such as NTT Japan are investing time and research effort in solutions that allow for Mobile Computing – dynamically migrating application code between the remote device and other network connected systems [7].

One avenue for power savings has not been fully considered, however. Many embedded devices, and all mobile devices, have a network link (GSM, Bluetooth, Ethernet, etc.) into a larger distributed environment. After designers incorporate sufficient power to support a network link, they then attempt to minimize use of the link due to its excessive energy needs during activity. Products therefore incorporate all the needed local storage in the device, buffering as much as possible to avoid retransmission. This ignores the fact that the remote server has a much less restricted power budget, and can

be made arbitrarily fast to handle requests quickly.

For ubiquitous always-on devices like 3G cell phones, there is the potential to use the network link as a means for accessing applications remotely. This could reduce local storage space, thereby reducing energy demands on the mobile platform. This remote memory could lie in a remote server, or simply be cached within the network infrastructure.

Utilizing the network link to access remote memory can provide a more energy efficient solution than traditional local memory. Traditional designs assume that the additional cost of utilizing the network link for moving code and data will far outweigh any benefit of removing or reducing local storage. The common misconception assumes the network is in use constantly, and therefore is much more power consuming than local storage. This is not always the case, as we will demonstrate. The best low-power mobile DRAM available today is 100 times less expensive to access in terms of energy per bit than a very low-power Bluetooth network. But for these same parts, the sleep-mode current of the Bluetooth network module is 10 times less expensive than the DRAM part. Therefore, if sufficient time elapses between accesses, the network link is more power efficient than local DRAM.

2. DEVICE MODELS

To investigate the possible performance effect of using the network as a mechanism for accessing remote storage, different device models and characteristics must be considered. There are three fundamental models of embedded computing devices that we examine: Legacy, Pull, and Push. Each model is characterized by the type of network link and communications model incorporated. We assume that the applications exhibit sufficient locality such there are well-defined “working sets” that change infrequently [1, 10].

Each model is considered independently. While general comparisons can be made across models, each has different design-time characteristics which make direct comparison difficult. The underlying hardware design behind each model is the same, however, and is shown in Figure 1A. The basic operation of such a device is that the program and data values are copied from Flash to local DRAM for performance reasons. This copying requires sufficient DRAM for holding all or part of the Flash contents.

We propose that by utilizing the network link to access the equivalent contents of Flash from a remote server, a more energy efficient model can be constructed at a lower cost. This is achieved by reducing the Flash component to just a boot-block sized unit, and removing some part of DRAM from the local storage. The DRAM removed would normally contain the contents of Flash copied on boot-up or during application mode change. Instead, we propose that a space large enough to hold the worst-case working set be reserved in the local DRAM, along with space for all local data. This concept for reduction is shown in Figure 1B.

While we suggest removal of the DRAM chip(s) and the resizing

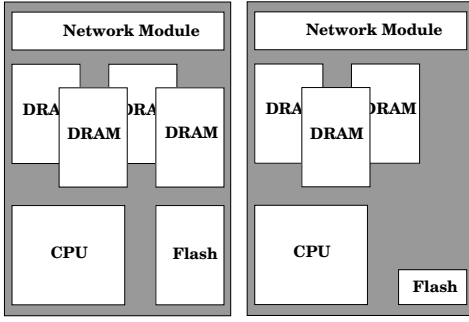


Figure 1: Basic 3G cell phone or other ubiquitous networked device. On the left is part (A) a typical mobile embedded device. Part (B) on the right shows the small reduction proposed in this work.

of the Flash storage, these actions are not strictly necessary. By carefully using V_{DD} gating, each DRAM and Flash unit could be disabled when not needed. This V_{DD} gating would result in power trade-offs similar to this work, but would not provide the increased flexibility for future application insertion and patching. Moreover, the total manufacturing cost of our design decreases, whereas V_{DD} gated units do not (and may even increase).

Next, we introduce each of the three embedded system models and the notation to analyze the energy and delay issues inherent in each. Full analysis of each model is in Section 3. Additional details on the equations are available in [4].

2.1 Legacy

The Legacy device was originally conceived and constructed without expectation for ever needing communication to other systems. We examine the issues of energy and delay in this model assuming a network link is added and local storage is reduced. The legacy application remains unchanged, but the code and data now come from network memory.

The original design expected a certain amount of normal energy consumption during computation and sleep or idle times. To see how adding a network link impacts this, we model the extra energy incurred by using the network link to fetch new code and data, as well as the energy consumed by the network link when in a sleep state. We assume the network link is only used for fetching new code and data, and that the legacy application itself is not attempting to communicate to other devices. We also model the extra time the CPU now spends waiting for network transactions to complete.

In order to “request” new code or data, a message must be generated and sent to the remote server. This transmission time (T_{Tx}) will consume energy as determined by the type of network link (E_{Tx}). Once the request is received at the remote server, there is some interval of time spent processing the request (T_{Srv}), during which there will be additional energy consumption on the local device monitoring the network (E_{Srv}). Once processed, the server will reply with the necessary information, which takes time to receive (T_{Rx}), consuming more energy (E_{Rx}). The “payload” of the transmission will consist of some number of bits, which consume power in proportion to the rate of the network communications. In comparison, local storage only incurs a very minor time to access (T_{DRAM}), with a correspondingly small energy use (E_{DRAM}). Whether transferred by network or from local storage, the CPU will be idle¹ during these transfers, consuming some amount of energy.

¹The CPU could be working on other tasks during this time, thus having a different energy signature. This is addressed later in this article.

Regardless of which method is used – network or local storage – after transferring the payload, time is spent in computation (T_{Busy}) before the next request is generated. During this time, the CPU will consume a different amount of energy while busy (E_{Busy} , and the backing store can be put into a powered-down or sleep mode. Thus, during the work period, the network link and local storage will consume their respective sleep power.

The total energy consumed by the network link (E_N) in the Legacy model is $E_N = E_{Tx} + E_{Srv} + E_{Rx} + E_{Busy}$, and the total energy in the local storage (E_L) is $E_L = E_{DRAM} + E_{Busy}$. In terms of energy, the network model is equivalent to the local storage model when $E_N = E_L$, but to consider the delay impact on application performance, we construct the energy-delay product:

$$E_N \cdot (T_{Tx} + T_{Srv} + T_{Rx} + T_{Busy}) = E_L \cdot (T_{DRAM} + T_{Busy}) \quad (1)$$

Solving this equation for T_{Busy} provides the energy-delay equilibrium point where using a network backing store is equivalent to using local DRAM. When T_{Busy} is greater than this equilibrium value, the network link is more energy-efficient from a total system perspective. That is, so long as the next application page fetched from the network occurs *on or after* computation for T_{Busy} seconds, the network memory model is more efficient.

2.2 Pull

Unlike the isolated Legacy model, the Pull model assumes a network link has been incorporated in the embedded device since creation. The characterization Pull comes from how the network is used: the local device, on its own initiative, pulls information from the network. External network devices can not arbitrarily send information to a device operating in a Pull mode.

Using the same notation as the Legacy model, there are only minor differences in the energy analysis. In the Pull model, the original design engineers already budgeted power for a network link. The link was expected to be in a power-down sleep state during normal operation except when the program requested remote activity. We only need to calculate the impact of new behavior (our additional traffic) over the original expected behavior (sleep state). Therefore, we consider the *difference* between the network link being in sleep state as opposed to actively sending and receiving messages.

The result is that the original assumption that all network link energy was a new burden is modified. We subtract the energy required for sleep-mode from the energy required to transmit and receive information. This change in the additional energy needed represents the new burden on the power source.

2.3 Push

Similar to the Pull model, the Push model also assumes a network link was built-in originally. In contrast with the Pull model, the network link is always on so that if not actively transmitting, it is in receive-listen mode. Thus external network services can immediately push information to the local device, such as e-mail notices, software patches, etc.

As the Pull model reduces the energy drain to store information in the network compared to the Legacy model, the Push model reduces the drain further. Since the device was designed assuming an always active receive mode network, the original design allotted sufficient power for this purpose. Therefore, we subtract the power term for normal receive-mode network links, rather than the smaller power term for a sleep-mode link as in the Pull model. That is, we only account for the additional energy of both sending extra messages out and idling the CPU during responses.

3. ANALYSIS

We now analyze in detail both the energy equilibrium point as well as the energy-delay product for each of the three modes discussed in Section 2. In order to have a quantitative analysis, we obtained technical data on current market products for both DRAM and Flash memory.

Using data sheets available from vendors including Elpida, Fujitsu, Intel, Micron, NEC, and Samsung, we selected low-power or “mobile” parts to represent typical market performance. We calculate the energy consumption in terms of pJ per bit by computing the *best-case* power consumption listed in the electrical characteristics of each product. This gives us a relative measure of how much energy is used in a *best-case* situation to read or write to the local storage device. During sleep mode, these devices consume very low current but still require some power for refresh functions. These calculations are shown for DRAM in Table 1.

Similarly, we calculate energy information from the data sheets published by several network link vendors. Unlike the DRAM, we determine the *worst-case* power per bit consumed, and the standby or sleep-mode power. In this situation, the transmit (Tx) and receive (Rx) modes are considered separately, as some links display different profiles by operating state. We restricted our search to monolithic, fully-integrated network modules to ensure valid power measurements. Using multiple chip solutions requires external components and glue logic which make power calculation difficult if not impossible. The components we considered and their power calculations are shown in Table 2.

For our analysis, we demonstrate a conservative extreme: *best-case* local storage vs. *worst-case* network links for remote storage. While neither of these models is generally realistic, they demonstrate the *extreme* bounds where network links are more effective than local storage. Thus in actual application, network links will be more efficient than we demonstrate here.

Next we define our memory, network, and platform CPU models and specify exact characteristics. Then we observe the basic trade-off between local storage and network storage of information. Finally, we examine whether these trends hold across alternate network choices and the implications for system designers.

3.1 Components

3.1.1 Best-Case Memory

To construct the best-case memory power model, we carefully choose to ignore certain effects in the CPU-to-memory interaction.

Since Flash is substantially slower than DRAM, the application is copied from Flash to DRAM for faster execution, and then Flash is placed in deep-sleep mode or V_{DD} gated off. Therefore, we *ignore* the contribution of Flash to the total energy. We also ignore the effects of initiating and waiting for memory access, and assume all accesses begin instantaneously at the maximum supported rate of the DRAM device.

Moreover, we define the transition from idle or sleep mode to active mode as instantaneous. We choose minimal V_{DD} and current consumption at all times, and ignore refresh operations. We also define that any code or data accessed is in the DRAM, and does not load from Flash.

This constitutes a *best-case* memory model. For analysis arguments, we use for the DRAM device the Fujitsu FCRAM model MB82D01171A, a 2MB part with the lowest power consumption of all devices measured in pJ/bit.

3.1.2 Worst-Case Network

For this analysis we restrict the additional traffic needed to support the network memory model to unalterable content such as pro-

grams, static global data, etc.

We further model the request for code or data to a remote server as fully encapsulated in a 64-byte packet. This could be reduced or expanded based on the network topology and error handling needs, but has sufficient storage space for a range requests. The response packet being a variable-payload version of the request packet will consist of 20 bytes for control information followed by the actual payload of variable size. These values are based on our implementations of such a client-server system [6].

We assume that for the total count of DRAM chips, at least one is for mirroring part or all of Flash. Based on the working set principle, only a small fraction of this space is actually needed at any given moment. Rather than store a large mirror image, only sufficient space for the worst-case working set should be reserved in local DRAM, with excess DRAM then removed. By using the network link to access applications, we could also shrink the Flash such that it contains only a boot image, and not all applications that could ever be run. This also reduces the burden of pushing massive code patches out to all systems in the network. Since steady-state mode changes occur relatively infrequently [1, 10], the need to load new code and data from the network will also occur infrequently.

The *worst-case* network model uses typical V_{DD} with worst-case current consumption in all cases. With slower transfer rates, higher current consumption, and a long duration of remote server processing T_{Srv} , the network appears unattractive for energy savings at first glance. We will now demonstrate that this is not the case.

The analysis that follows will implicitly use the concept of one computational task running on the mobile device. Since the CPU would normally be busy during the additional network transactions to receive new code (using the best-case zero-overhead local memory access), we model the CPU as completely idle during these periods. If multiple tasks were present, the CPU could simply switch to the next task and continue processing. This would not add the energy overhead of sitting idle and delaying all work, and thus is not the worst-case scenario for network impact.

Our analysis uses the idea that *one* DRAM chip is removed, although it could include reducing Flash as well as multiple DRAM chips. Our network link model is the CSR BC2-Ea, a fully integrated Bluetooth module. This module exhibits a starting time of $10\mu s$ and a settling time of $5\mu s$ in the internal ADC for gain control. A transition from Active to Sleep mode in the network module is sub-1ms. As we will demonstrate later, the server processing time for requests is set to 10ms. With such a large time for server processing, we ignore these second and third-order effects of the network module design.

3.1.3 Mobile CPU

Our CPU model for the mobile device is the DEC SA-110, a 0.5W processor during high computation and 0.02W during idle periods when operating at 160MHz [3]. Several interesting factors arise from using this particular processor as our representative model.

The SA-110 can transition between Idle and Active mode with effectively no delay. This is accomplished by the two separate clock domains within the SA-110 – in Idle mode, the internal bus clock and clock grid stop signaling. The actual steps to enable Idle mode are toggling a register, loading an uncacheable address, and waiting for an interrupt – a few instructions. The recovery is achieved after receiving an interrupt, and restoring the original register values. Since the transition between Active and Idle mode is nearly instantaneous, we do not model the time necessary for such transitions in the CPU.

3.2 Initial Impact

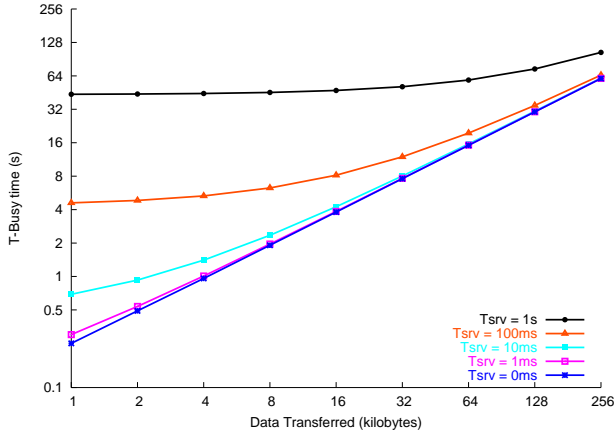


Figure 2: Each line represents the T_{Busy} equilibrium point for different remote server processing times T_{Srv} . The network transmission speed is the limiting factor during payload transfers, shown as the asymptote when $T_{Srv} = 0ms$.

Given that network transmission speeds lag substantially behind the bandwidth of local memories, the bounds on T_{Busy} will be dependent on the network speed. With increasing payload in transfers, the remote server processing time becomes less important than the overall network performance. Figure 2 illustrates the boundaries as T_{Srv} varies from zero to one second.

It is unreasonable to assume there will be zero processing overhead on the remote storage system. The incoming network request has to be received, interrupt handlers invoked, memory searched, etc. Using our already existing client-server system as a basis [6], an Intel PIII 800MHz system running RedHat Linux 8.0 is capable of processing and responding to requests in sub-10ms times. During this time the server is also running a full interactive X desktop with multiple applications running. Therefore, we use 10ms as an approximate remote server processing time. While the remote server could be optimized and made arbitrarily powerful, it will still be serving multiple targets and similar response times could be expected.

To compare the Legacy, Pull, and Push models using our established T_{Srv} of 10ms, we again plot the necessary T_{Busy} to reach energy-delay equilibrium compared to local storage accesses. Figure 3 demonstrates the trade-offs between the three models. Any value of T_{Busy} beyond the times shown in this figure are a “win” for using remote storage instead of local storage.

The Legacy model presents the worst energy-delay product result. We have added a network link to a design that did not expect it. For the network link to be more efficient than local DRAM, it requires a significant amount of time spent in computation, T_{Busy} .

The Pull model provides better energy-delay results than the Legacy model, as can be expected from subtracting the sleep mode power. The improvement turns out to be small compared to the energy costs associated with transferring the data as well as the remote server processing time T_{Srv} . The actual difference between the Legacy and Pull models is slightly less than 8%. This does indicate that adding a network link to a legacy system when using a pull-based communications model will have a small impact when compared to the energy consumed by local storage devices.

The Push model uses the least additional energy and thereby benefits most from using remote storage. Since the network link is already expected to be in a receive-mode state at all times, the only extra energy used to access remote storage is the energy of the transmit

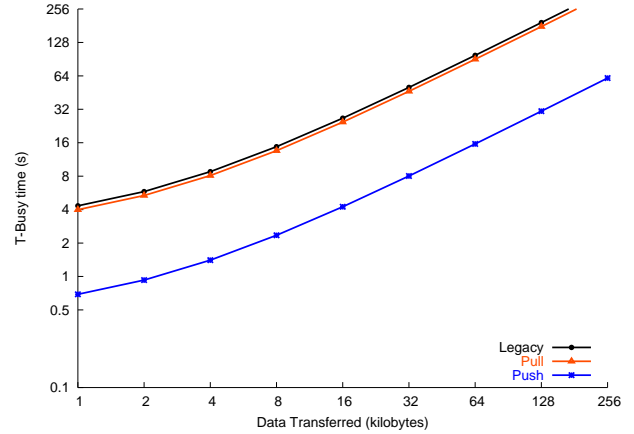


Figure 3: Comparing the energy-delay equilibrium characteristics of Legacy, Pull, and Push models when $T_{Srv} = 10ms$.

operations.

The energy-delay benefit for a 1KB “page” change with T_{Srv} of 10ms in the Legacy model requires 4.33s as a minimum change interval. With the Pull model, the required busy time falls to 3.99s. When considering the Push model, the time is reduced to 0.69s. In relative comparison, this same 1KB “page” of code loaded across the network with $T_{Srv} = 10ms$ will present a total application delay of 16ms to the user while accessing the network. This includes the sending, processing, and return of payload through the network.

A larger “page” size to transfer may be more realistic to consider, however. For a 16KB change with $T_{Srv} = 10ms$, the Legacy model requires 26.6s between transfers, and the Pull model requires 24.5s. The Push model reduces this time to a mere 4.2s. The delay the user experiences while the network transfer occurs is 185ms.

3.3 Portability

In order to compare these results based on a very low-power Bluetooth integrated module to other network types, we now consider two alternate network models. Neither of these alternatives come in complete monolithic solutions, but instead comprise two or three highly integrated chips with some minimal external glue logic. The estimates for the Cypress Wireless USB chipset and the Bermai Integrated 802.11a chipset include only the main chip components. Power consumption of the glue logic is not considered, and therefore these numbers are slightly smaller than they should be in a worst-case scenario. In particular the 802.11a chipset has particularly large currents in *any* mode of operation before considering the glue components.

Using the Push model as a baseline, we compare the CSR BC2-Ea solution to both the Cypress and Bermai solutions. Figure 4 displays the results of this comparison. The surprising result from this figure is that the very power-hungry 802.11a network is a much better selection than low-power Bluetooth or similar modules. The substantially higher data rate causes the limiting factor not to be the network link speed, but rather the remote server processing time.

This comparison is against a Push model, where sufficient power was built-in to support the network in constant-receive mode. A more illustrative example of the substantial power drain involved in 802.11 chipsets can be seen by comparing to the Pull model, shown in Figure 5. Note that the initial energy cost of the 802.11 network far exceeds other options, but that if the typical payload transferred in the network is ≥ 32 kilobytes then the 802.11 network is a better design choice.

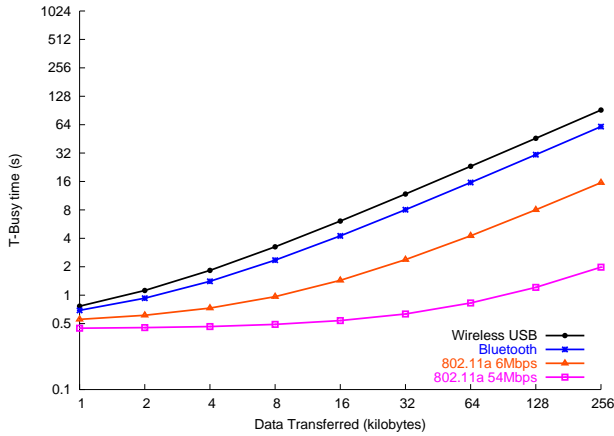


Figure 4: Comparing the Push model energy-delay equilibrium characteristics with Bluetooth, Wireless USB, and 802.11a network modules when $T_{Srv} = 10ms$.

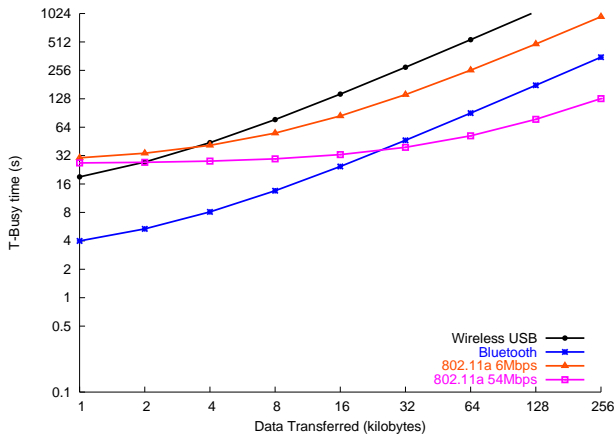


Figure 5: Comparing the Pull model energy-delay equilibrium characteristics with Bluetooth, Wireless USB, and 802.11a network modules when $T_{Srv} = 10ms$.

While these results do not specifically tie to any estimated average transfer size, they show interesting trends. Ultimately the typical payload size will be entirely dependent upon the applications and network support by commercial providers. This work shows that by performing careful analysis of what types of applications and data will be used in the network, and the characteristics of those applications, increasing local storage may be the wrong approach to longer battery life.

4. RELATED WORK

Utilization of the network for accessing backing store as a low-power mechanism is novel. Prior work concentrated on using remote memories for high-performance reasons, avoiding accesses to slow disks or to expand memory for working sets of code or data [12, 2]. Other work examining the network in power-limited devices has concentrated and minimizing the usage [5] and optimizing protocols.

A significant amount of effort is being spent to find ways of improving the overall energy efficiency of networks. WLANs can improve their efficiency by using ad-hoc relaying [9], while others look at tying battery level with ad-hoc routing methods to increase net-

work robustness as well as node run-times [8].

Using the availability of low-power short-range devices such as Bluetooth, researchers are building larger networks in an energy-efficient manner. These new systems compete with more traditional network options [11]. Such prototypes strengthen the viability of using limited embedded hardware for larger projects.

With each generation of network technology, data rates increase and power consumption decreases. Next-generation technology such as Ultra-Wideband is anticipated to be higher data rate and lower power than current Bluetooth devices, with similar if not better range. As network links approach the performance characteristics in bandwidth and power of local DRAM, the argument for moving to network-based storage becomes more compelling.

5. CONCLUSION

This article has presented a trade-off between local storage and remote network storage. Conventional wisdom has been to minimize the use of networks. This work is meant to cast doubt on such a broadly general rule, and to encourage reconsideration of how devices will be used in order to minimize power consumed.

We have shown that the underlying assumptions are misleading, and that using remote servers to store information can be more energy-efficient than using local storage. Accounting for processing times, protocol overheads, and using *best*-case local storage behavior versus *worst*-case network link behavior, we demonstrated that networks are more efficient when transfer times exceed a small threshold.

For a reasonable, average working set of 32KB transferred via the network, the link is more efficient than local DRAM if transfers occur no more frequently than every 4.24s, with a user-experience delay of only 0.185s, using a Push model for comparison. This assumes the network link replaces *one* local 2MB DRAM chip. Removal of larger or multiple components makes network usage even more efficient.

Today, network links use 100 times the energy of DRAM during accesses, but consume 10 times less energy during sleep. Network devices continue to approach low-power DRAM performance characteristics in terms of speed and power, making network memory increasingly attractive. The reduction in part count, and thus price, can aid in the marketing of disposable devices such as cell phones. This model where applications are downloaded on demand also provides a mechanism for pay-per-use services, such as custom games or video players. We are currently working on detailed simulations to study the impact of this network memory model on overall network congestion.

6. ACKNOWLEDGEMENTS

We would like to thank the reviewers for their useful comments in restructuring this article. This work was funded in part by the National Science Foundation under grants CCR-98-76180, CCR-01-21638, and EIA-99-72872.

7. REFERENCES

- [1] G.A. Abandah and E.S. Davidson. Configuration Independent Analysis for Characterizing Shared-Memory Applications. Technical report, University of Michigan, CSE-TR-357-98, 1998.
- [2] D.Pnevmatikatos and E.P. Markatos. On Using Network RAM as a Non-volatile Buffer. In *Cluster Computing*, pages 295–303, 1999.
- [3] James Montanaro et al. A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor. In *IEEE Journal of Solid-State Circuits*, volume 31(11), pages 1703–1714, November 1996.
- [4] J.B. Fryman, C.M. Huneycutt, H.-H.S. Lee, K.M. Mackenzie, and D.E. Schimmel. Energy Efficient Network Memory for Ubiquitous Devices. Technical report, Georgia Institute of Technology, GIT-CERCS-03-05, 2003.

- [5] P.J.M. Havinga and G.J.M. Smit. Energy-efficient wireless networking for multimedia applications. In *P. Havinga and G. Smit, Energy-efficient Wireless Networking for Multimedia Applications, in Wireless Communications and Mobile Computing, Wiley, 2000.*
- [6] C.M. Huneycutt, J.B. Fryman, and K.M. Mackenzie. Software Caching using Dynamic Binary Rewriting for Embedded Devices. In *International Conference on Parallel Processing, 2002.*
- [7] NTT Japan. BLUEBIRD Project. 2003.
<http://www.ntts.co.jp/java/bluegrid/en/>.
- [8] D. Kim, J.J. Garcia-Luna-Aceves, K. Obraczka, J. Cano, and P. Manzoni. Power-Aware Routing Based on The Energy Drain Rate for Mobile Ad Hoc Networks. In *Proceedings of the IEEE Intl Conference on Computer Communication and Networks, Oct 2002.*
- [9] M. Kubisch, S. Mengesha, D. Hollos, H. Karl, and A. Wolisz. Applying ad-hoc relaying to improve capacity, energy efficiency, and immission in infrastructure-based WLANs. Technical report, Technical University Berlin, July 2002.
- [10] R. Batchu, S. Levy, and M. Murdocca. A Study of Program Behavior to Establish Temporal Locality at the Function Level. Technical report, Rutgers University, DCS TR-475 2001.
- [11] S. Baatz, C. Bieschke, M. Frank, K. Kühl, P. Martini, and C. Scholz. Building Efficient Bluetooth Scatternet Topologies from 1-Factors. In *Proceedings of the IASTED Intl Conference on Wireless and Optical Communications, July 2002.*
- [12] S. Dwarkadas, N. Hardavellas, L. Kontothanassis, R. Nikhil, and R. Stets. Cashmere-VLM: Remote Memory Paging for Software Distributed Shared Memory. In *Proceedings of the Intl Parallel Processing Symposium and the Symposium on Parallel and Distributed Processing, pages 153–159, 1999.*

Vendor	Model	MB	width	MHz	V_{DD}	access mA	sleep mA	pJ/bit	pJ/bit/MB
Elpida	EDL1216AASA	16	16	133	2.3-2.7	80	1.5	86.5	5.4
Fujitsu	MB82D01171A-80	2	16	125	2.3-2.7	20	0.2	23.0	11.5
Micron	MT48V4M32-10	16	32	100	2.3-2.7	100	0.35	71.9	4.5
Micron	MT48V16M16-10	32	16	100	2.3-2.7	80	0.35	115.0	3.6
NEC	μ PD4664312	8	16	150	2.7-3.3	45	0.1	49.6	6.2
Samsung	K4S643233-75	8	32	100	2.3-2.7	85	5	61.1	7.6
Samsung	K4S283233-75	16	32	100	2.7-3.6	220	6	185.6	11.6
Samsung	K4S561633-1H	32	16	100	2.7-3.6	130	6	219.4	6.9

Table 1: Mobile DRAM characteristics: size, bit-width, speed, voltage, best-case access current, best-case sleep-mode, current use, pJ per bit in accessing, and a normalized pJ per bit per megabyte of memory. Refresh impact not included.

Vendor	Type	Range	Model	kbps	V_{DD}	TX mA	RX mA	sleep uA	TX μ J/bit	RX μ J/bit
AMI Semi	SpreadS	300m	ASTRX1	40	3.3	14	25.0	10.0	1.155	2.063
AMI Semi	Modem	n/a	A519HRT	1.2	5.0	0.6	0.6	n/a	2.500	2.500
CSR	Bluetooth	100m	BC2-Ea	1500	1.8	53	53.0	20.0	0.064	0.064
MuRata	Bluetooth	100m	LMBTB027	1000	1.8	60	58.0	30.0	0.108	0.104
NovaTel	Wireless	n/a	Expedite	38.4	3.3	175	130.0	5.0	15.039	11.172
OKI Semi	Bluetooth	100m	MK70	921.6	3.3	115	72.0	n/a	0.412	0.258
Option	GSM	n/a	GlobeTrotter	116	3.3	550	50.0	50.0	15.647	1.422
Radiometrix	UHF	30m	BiM-UHF	40	5.0	21	16.0	1.0	2.625	2.000
Siemens	Bluetooth	20m	SieMo S50037	1500	3.3	120	120.0	120.0	0.264	0.264
UTMC	Bus	n/a	UT63M1xx	1000	5.0	190	40.0	n/a	0.950	0.200
Vishay	IrDA	Varies	TFBS560x	1152	5.0	120	0.9	1.0	0.521	0.004
Wireless Futures	Bluetooth	100m	BlueWAVE 1	115.2	3.3	60.9	60.9	50.0	1.745	1.745
Cypress	W-USB	10m	CYWUSB6941,2	1000	3.3	120	135	20.0	0.396	0.446
Bermai	802.11a	50m	BER7000	54000	3.3	454	364	3030	0.028	0.022

Table 2: Link characteristics: speed, voltage, current draw in various states, and worst-case μ J per bit power consumption for TX and RX. The last two entries (Cypress and Bermai) are only approximations.