An Embedded Platform with Duty-Cycled Radio and Processing Subsystems for Wireless Sensor Networks

Zhong-Yi Jin
Dept. of Computer Science & Eng
UCSD
zhjin@cs.ucsd.edu

Curt Schurgers
Dept. of Electrical & Computer Eng
UCSD
curts@ece.ucsd.edu

Rajesh Gupta
Dept. of Computer Science & Eng
UCSD
gupta@cs.ucsd.edu

ABSTRACT
Wireless sensor nodes are increasingly being tasked with computation and communication intensive functions while still subject to constraints related to energy availability. On these embedded platforms, once all low power design techniques have been explored, duty-cycling the various subsystems remains the primary option to meet the energy and power constraints. This requires the ability to provide spurts of high MIPS and high bandwidth connections. However, due to the large overheads associated with duty-cycling the computation and communication subsystems, existing high performance sensor platforms are not efficient in supporting such an option. In this paper, we present the design and optimizations taken in a gateway node that provides access to a Wi-Fi radio in an on-demand basis. We discuss our strategies to reduce duty-cycling related costs by partitioning the system and by reducing the amount of time required to activate or deactivate the high-powered components. We compare the design choices and performance parameters with those made in the Intel *Stargate* platform to show the effectiveness of duty-cycling on our platform. We have built a working prototype, and the experimental results with two different power management schemes show significant reductions in latency and average power consumption compared to the *Stargate*. The WGN running our power-gating scheme performs about 6 times better in terms of average system power consumption than the *Stargate* running the suspend-system scheme for large working-periods where the active power dominates. For short working-periods where the transition (enable/disable) power becomes dominant, we perform up to 7 times better. The comparative performance of our system is even greater when the sleep power dominates.

Categories and Subject Descriptors
C3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems – *Real-time and embedded systems*.

General Terms
Design, Performance, Measurement, Experimentation.

Keywords
Dynamically managed sensor nodes, Power management in sensor nodes, Gateway nodes, Low power design.

1. INTRODUCTION
A wireless sensor network (WSN) consists of a collection of wireless sensor nodes which are small embedded devices with on-board sensors and wireless radios. Without wires, sensor nodes either rely on limited energy supply from batteries or harvested energy from intermittent sources like solar or wind. To ensure long lifetimes demanded by the application and deployment scenarios, the sensor nodes have to be very energy efficient. Popular sensor nodes such as the Berkeley Mote [15] address this issue using low power hardware as well as aggressive power management techniques. However, their design choices also make these nodes useful only to applications requiring limited processing power, short communication range and low network bandwidth.

A common WSN architectural solution to these constraints is to deploy within the sensor network a small number of high performance nodes equipped with high powered components like fast processors or high bandwidth radios. As high performance sensor nodes are usually placed in the same environment as regular sensor nodes, they also rely on limited energy sources. Therefore, energy-efficiency is critical for the high performance nodes to last as long as the rest of the sensor nodes.

There are two observations that we can explore to improve the energy-efficiency of high performance nodes. Firstly, as a general fact, using components with high peak power consumption doesn't necessarily imply high energy consumption. Studies have shown that when a sufficient amount of data need to be processed or transmitted, high performance processors or radios that consume more power than their sensor node counterparts may complete the same amount of work faster and therefore end up using less energy [10, 12, 17]. Secondly, in the specific case of sensor networks, those high powered components are not required to be active all the time as sensor networks usually do not generate large amounts of data until certain triggering events are detected. In other words, the node or its components needs to be active only a fraction of the time to achieve application goals. Therefore, duty-cycling based power management techniques such as selectively enabling or disabling components are important in reducing energy consumption for high performance nodes.

A platform needs to meet two requirements to support efficient duty-cycling. One is that it needs to consume very little (or no) power when there are no ongoing activities. While general purpose high performance nodes such as the *Stargate* [4] provide a good balance of performance and power consumption, they are not designed to support efficient power management via duty-cycling. For example, the lowest power consumption of a *Stargate* is 16.2mW in its inactive suspended state [11]. In contrast, a typical sensor node such as the Telos Mote uses only about 10μW in a similar state [15]. This high standby power consumption significantly limits the effectiveness of duty cycling, making the *Stargate* less energy efficient for very low duty cycle sensor network applications such as environmental monitoring. The other requirement is that a platform needs to be able to activate or deactivate various subsystems with very little overheads according to runtime demands. The *Stargate* is also not sufficient at this point, as it generally takes a long time for a *Stargate* to transit in and out of the suspend state, bringing significant energy and performance overheads to duty-cycling [11]. Nevertheless, *Stargates* and similar platforms are widely used in monitoring applications [2, 8, 20]. In these cases, special power sources such as large lead-acid batteries have to be added to the systems. These workarounds impose additional constraints to the deployments of sensor networks.

In this paper, we describe the design, implementation and evaluation of a wireless gateway node (WGN) that enables efficient power management through duty-cycling. As shown in Figure 1, gateway nodes are often intrinsic parts of sensor networks and are required to bridge data between sensor networks and servers in other networks. The low and bursty traffic load of sensor networks makes the WGN an ideal application of our low power design. Specifically, we focus on using Wi-Fi (802.11b) radios to interface sensor nodes with devices in other networks because they offer higher bandwidth and consume less energy per bit than most existing sensor node radios [17], making them a useful air interface for many sensor network applications. Using Wi-Fi radios also provides access to the ubiquitous Wi-Fi infrastructure networks, allowing them to be used as backbones to connect wireless sensor networks with remote servers or as overlays for sensor nodes that are distantly apart to interact with each other. Although our focus is on the gateway nodes, the basic design and techniques can also be applied to any other high performance nodes or used in the context of any multiple radio hierarchies [1].

Our contributions are as follows:

1. Design of a dual-processor high performance sensor node platform that supports efficient duty-cycling.
2. Experimental analysis of how to reduce duty-cycling related costs at hardware, operating system and network levels.

We have built a working prototype with commercial off-the-shelf components and evaluate the design experimentally using two power management schemes. Our results show significant reductions in latency and average power consumption compared to the *Stargate* and similar platforms. Given different design optimization criteria used for the two platforms, instead of a comprehensive comparison of the strengths and weaknesses, our results provide a measure of potential reduction in power consumption due to architectural redesign for duty-cycling.

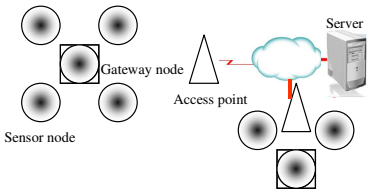

Figure 1. Wireless sensor networks with 802.11b based gateway nodes

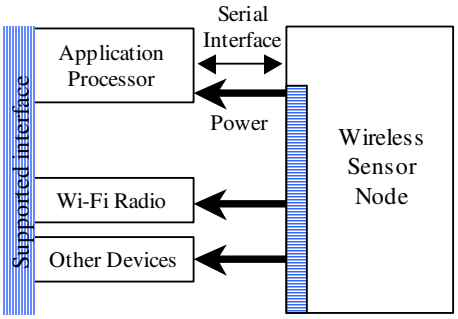

Figure 2. WGN Block Diagram

2. RELATED WORK

To energy efficient design, the importance of separating real-time monitoring functions that have to be optimized for low power from functions invoked with light duty-cycles is first unveiled in the development of the WINS nodes [16]. The WINS node enables continuous sensing over an extend period of time by partitioning the system into a low powered event-detection subsystem and a high powered processing subsystem which can be enabled or disabled on demand.

Our work is directly comparable to the emerging Micro-servers that are being explored to solve a large body of sensor network research problems ranging from image tracking [18] to network life extension [3]. Numerous efforts have been undertaken to reduce the average power consumption of these devices. Triage [3] extends the lifetimes of their Micro-servers by trading latency for power reduction. Its tiered architecture consists of a slightly modified *Stargate* computer and a MicaZ mote, which is used to power on the *Stargate* only when sufficient amount of data are being batched for processing. Due to the large latency in powering on/off the *Stargate*, their platform is not usable for our gateway application in terms of delay and power consumption. The LEAP platform [12] faces similar issues. The Cyclops [18] platform integrates a CPLD (Complex Programmable Logic Device) with a sensor node for their image sensing application. It is not suitable for our purposes as the CPLD is not ideal for general purpose applications. The PASTA platform [19] consists of a family of modules that can be stacked together. By embedding a power microcontroller clocked at 32kHz into each module, their architecture enables independent power control of modules and module subsystems. Their processor module (PXA module) uses an Intel PXA255 processor also but no latency numbers are reported for activating this module from power-off state.

3. DESIGN APPROACH

Figure 2 shows a high-level block diagram of our design. To minimize standby energy consumption, we exploit the low power operation of a sensor node processor and use it for subsystem scheduling and power management. A second, more powerful application processor is used to provide on demand processing capabilities to subcomponents such as the Wi-Fi radio, which are physically connected to the application processor. Power to each individual subcomponent is either controlled directly by the sensor node processor or indirectly by the application processor through the sensor node processor. We use a serial interface for inter-processor communication because it is supported in various forms by most of the existing sensor node processors.

Unlike the PASTA platform [19] where multiple microcontrollers are used to regulate power to the modules, we have the sensor node processor acting as the "master" device for power management. Our approach simplifies the design while also taking advantage of the fact that the sensor node processor is almost always on for sensing or networking purposes. This enhances the efficiency of running the power management itself.

To reduce the latency in activating the application processor from power-off mode, we found it is critical to minimize both hardware and software startup time. On the hardware side, commercial microcontrollers for embedded devices are usually designed to support fast startup. On the software side, it is important to minimize the amount of the code that needs to be loaded every time during boot up.

It generally takes less time to resume a program from low-power mode when the program is either suspended or running at a slower speed than to reload the entire program by powering the application processor back on from power-off mode. However, a certain amount of power is still consumed while the application processor is in low-power mode. While it is clear that having minimal standby power consumption would extend the operation time, some real time applications also have time constraints, as they need to complete certain tasks within a fixed amount of time. Instead of making the power and latency trade off at design time, the sensor node processor needs to be able to put the application processor into either low-power mode or power-off mode at runtime based on application requirements.

4. PLATFORM IMPLEMENTATION

The hardware architecture of our WGN is illustrated in Figure 3. As explained in the introduction, our main contributions are on the level of the architectural design approach (see also section 3). To illustrate these ideas and perform experiments, we have to make specific design choices for our test bed platform. Our approach, however, is not restricted to these specific choices alone.

We use the Ubicom IP2022 processor [21] as our application processor. With a maximum clock speed of 120MHz (approximately 120MIPS), this 8-bit processor is less powerful than the 400MHz (480MIPS, Dhrystone 2.1) PXA255 on the *Stargate*. However, it is significantly faster than the microcontrollers on most existing sensor nodes and provides sufficient processing capability to our gateway application. With its integrated flash memory and RAM, this processor doesn't need complex external hardware support and thus doesn't incur extra

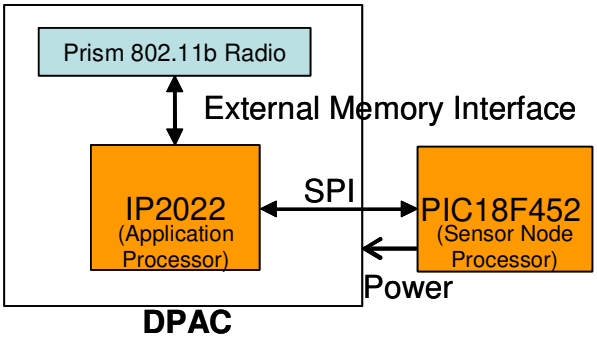

Figure 3. WGN Architecture

energy and latency overheads. We also find the IP2022 a convenient choice as it is used in the DPAC module [6] with a Wi-Fi radio.

Because of a clear separation of the master (sensor node processor) and the rest of the system on our platform, we can select any existing sensor nodes for our power management purposes. Our design only requires that the sensor node processor and the application processor can communicate and wake each other up as necessary. This is easier to implement than those that require external logics or chip-set supports as in PC platforms. Commonly used serial interfaces such as the UART, I2C or SPI are sufficient to meet these requirements. We select the SPI as it supports duplex data transfers at a rate of more than 11Mbps, sufficient to sustain the throughput of the Wi-Fi radios.

For evaluation purposes, we use for power management a home grown sensor node equipped with a PIC18F452 microcontroller that consumes about 8mW (3.3V) in full power mode (10MIPS). The hardware SPI on the PIC is still available. The power to the DPAC module is managed by the PIC through a MOSFET switch. Alternatively, we can use an I2C power switch to control additional components. We use Ubicom IPOS, a lightweight OS for the IP2022. The IPOS provides the very basic OS functions and gets compiled with the target application. Our entire gateway application is about 60 Kbytes.

5. POWER MANAGEMENT SCHEMES

We experiment with two different power management schemes for our gateway application to evaluate the platform as well as to understand the power versus latency tradeoffs. We refer to these two schemes broadly as *power-gating* and *power-saving* modes. In the former, the emphasis is on subsystem shutdown for both communication and processing, while the latter seeks to exploit various slowdown modes. While the choice between shutdown and slowdown has been a subject of various optimizations in the power management literature, the basic tradeoffs concern how aggressive a power management scheme is in reducing power versus the latency penalty it incurs. Our platforms support both *power-gating* and *power-saving* modes. In the following subsections, we describe the design choices behind the two schemes.

5.1 Power-Gating Scheme

Our *power-gating* scheme saves power by putting the system into *power-gating* mode according to online demands. While in the *power-gating* mode, the Wi-Fi radio and the application processor are powered off and no packets can be sent or received. A successful use of the *power-gating* scheme requires participating gateway nodes and devices in other networks to coordinate their active periods while minimizing the total amount of energy required for such synchronizations [5, 14, 22]. We measure the overheads of such protocols when used with our WGN. The overheads are quantified either as time in seconds or energy in Joules calculated by integrating power over time.

5.2 Power-Saving Scheme

Our *power-saving* scheme reduces power consumption by putting the radio in the 802.11b power saving mode [9] while keeping the application processor in various low power modes. Since most Wi-Fi radios natively support the 802.11b power saving mode, it is significantly faster to put the radio in and out of the power saving mode than to suspend and resume the entire radio in each duty cycle. Although the speedup is hardware dependent, it is reported in one case that it is almost 86 times faster to resume from power saving mode than from suspended mode [1]. One challenge in supporting such a scheme is to synchronize power saving states across the processors, the radio and the access point. In the 802.11b power saving mode, a radio wakes up periodically to check for any incoming packets buffered at the AP (Access Point) and the sleep duration is determined by the listen interval. A listen interval is established during the association process and is fixed until new association attempts are made. Since the re-association process involves multiple packet exchanges between the station and the AP, it is expensive to change the listen interval frequently. However, a fixed listen interval is not ideal for most sensor network applications as events occur randomly. Long listen intervals introduce large communication delays while short listen intervals waste energy if there are no events or data to send or receive. Thus, the choice of the listen interval is a matter of design tradeoff between energy savings and latency incurred. Instead of listening at a fixed interval, we use an event-driven approach to transition in and out of the power saving mode as explained in its implementation below.

Our strategy is based on the 802.11b standard [9] that after a station exits power saving mode, the AP should send all buffered packets to that station as if they just arrived. Therefore, if the listen interval is set to an arbitrary large value (up to 2^16 beacon intervals), one can eliminate its effects and dynamically control the packet receiving time by forcing the station out of the 802.11b power saving mode. Although a successful packet exchange is required between the station and the AP to enable or disable the 802.11b power saving mode, this can by done by simply changing the power management bit in the frame control field of any outgoing packets. A potential benefit of this strategy is that with some buffering, sending and receiving can be performed within the same active periods and therefore reduces the total amount of time the radio and the application processor need to be awake. We experiment with this approach by sending and receiving 1024 bytes of data in an UDP packet every 6 seconds for a period of 30 minutes using the same method described in the last paragraph of section 6.3 and observe no packet loss. Note that to avoid overrunning the buffer of the AP, a small packet should be sent to instruct the receiving device running our scheme to wakeup more frequently before transmitting a large amount data.

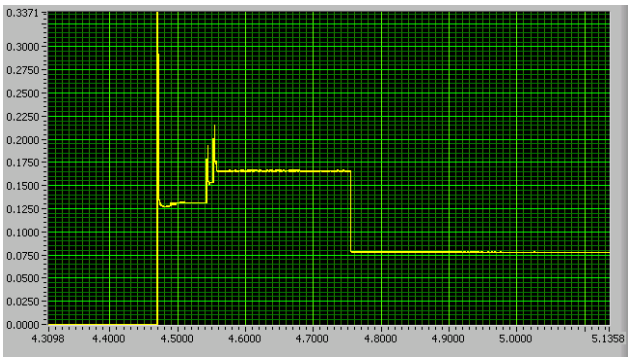

Figure 4. Switch from off (gated) mode to active mode

6. EXPERIMENTAL RESULTS AND ANALYSIS

To evaluate the performance of our design, we experiment with the power management schemes discussed in section 5.

6.1 Measurements and Chart Interpretation

Power consumption is measured as the product of voltage and current. Our WGN is directly powered by a DC power supply of 3.3V. To measure current, a resistor of 1 Ohm is connected in series with the DC power supply. The voltage drop across the resistor is sampled using a National Instrument DAQPad-6020E (12Bit, 100KS/S) and stored in Lab-View spreadsheet format. For comparison purposes, hardware components such as the Wi-Fi radio and the processors as well as software components like the OS, IP and UDP libraries are always being initialized to default states during initial power up. For simplicity, sensor data are randomly generated by the PIC processor rather than from real sensors or other sensor nodes.

6.2 Power-Gating Overheads

In this experiment, we measure how long it takes to activate the Wi-Fi radio and the application processor from power-off state and the overall power consumption of the system in active state. We start the experiment by powering up the whole system, including the radio and both processors. The activation process ends when the application processor is running at full speed, ready to execute real application tasks. The exact time of completion can be observed on the chart as a sharp voltage drop induced for evaluation purposes by shutting the application processor off. Since the 802.11b association-time varies significantly with different access points, authentication methods and network settings, it is excluded from measurements in this experiment.

As shown in Figure 4, the WGN is powered up at around 4.47 seconds. It takes about 280ms (4.47s- 4.75s) to switch from *power-gating* mode to active mode. The reason the system still consumes about 82mA of current after shutting off the application processor is that the Wi-Fi radio draws power even after being put into off state, probably due to hardware limitations. A similar case has been reported in [13] where a Netgear MA701 CF Wi-Fi card, which is also based on the Intersil chip, consumed 83mA (3.3V)

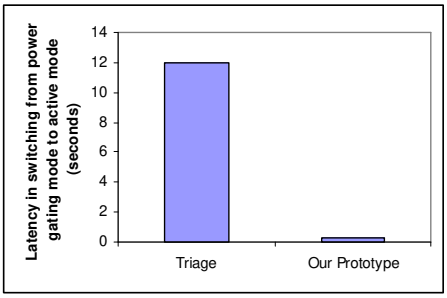

Figure 5. Compare power gating scheme latency with Triage

of power in off mode. For our *power-gating* scheme, we work around this by shutting off power to both the application processor and the Wi-Fi radio.

The delay of the application processor in switching from power-off mode to active mode severely affects the efficiency of duty-cycling. Large latency not only reduces the responsiveness of the system but also increases the average power consumption. In Figure 5, we compare the latency with the one from the Triage system [3] discussed in the related work section. The latency of the Triage system shown in Figure 5 is the amount of time to fully activate the *Stargate* from the power-off state. The startup latency is not determined by processor speed alone. RAM and MMU initialization time, flash-memory read and write speed and most importantly the operating system overhead all play major roles. Reducing such latency would require careful hardware and software design and engineering. It is our future work to reduce startup latencies on PXA255 class processors by techniques such as using smaller OS (eCOS [7]) and by dividing RAM into banks that can be power managed separately.

6.3 Power-Saving Overheads

In this experiment, we try to evaluate the performance of our *power-saving* scheme. The experiment setup is similar to the one in Figure 1 but consists of only one WGN, one access point (a part of the UCSD campus wireless network) and one server that is connected to the internet through wired Ethernet. We perform the evaluation by first passing data from the sensor node to the application processor and then have the application processor send the data to the server through the Wi-Fi radio.

Figure 6 shows the results of running the *power-saving* scheme code in Figure 8. The system is powered up at around 4.0s. The networking subcomponent (Wi-Fi radio and the application processor) enters into the sleep state at round 12s. At about 16s (4 + 4 + 8), the PIC interrupts and sends the first 64 bytes of data to the IP2022. The rise of power consumption at that point is the result of waking up the IP2022 and the radio from low power states. After waking up, the IP2022 instructs the radio to exit the 802.11b power saving mode and sends the 64 byte of data in a UDP packet to the server. If there are incoming packets buffered at the AP for this node, they will be delivered by the AP upon receiving this packet.

Certain fixed delays are introduced to simplify synchronization efforts. As shown in Figure 8, the PIC waits 1 second after the interrupt for the IP2022 to fully wakeup. The 1 second timer of the IP2022 can keep the networking subcomponent from entering

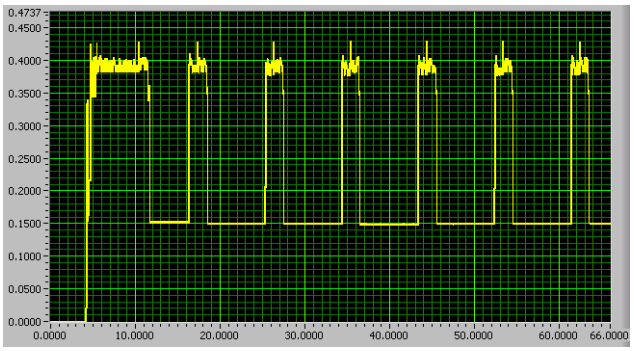

Figure 6. Power saving scheme

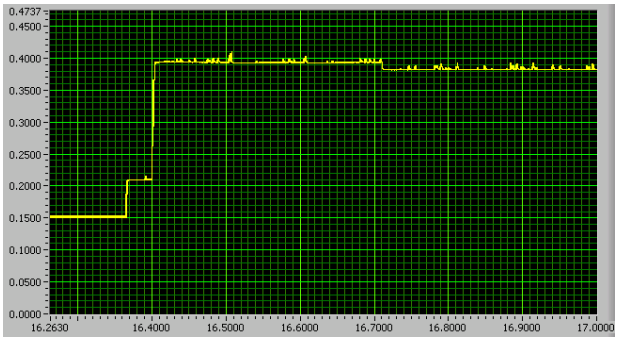

Figure 7. Zoom in view of Figure 6

```
// IP2022                          // PIC
InitEverything();                  Init();
EnableWiFi();                      PowerUpDPAC();
RunDHCPClient();                   //sleep 4s for DPAC
// send notification               //to initialize
SendUDP();                         Sleep(4);
while(TRUE) {                      while(TRUE) {
 PwrSaveWiFi();                     Sleep(8);
 PwrSaveProcessor();                InterruptSPI();
 WaitForInterrupt();                // Wait for IP2022
 ActivateProcessor();               // to wake up
 ActivateWiFi();                    Sleep(1)
 SetTimer(1);                       //send 64bytes via SPI
 while(TimerNotExpired()){          WriteDataSPI(64);
  //read data from SPI              }
  if(ReadDataSPI()){
     //send data via UDP
     SendUDPWiFi();
     ResetTimer(1);}
  if(ReadWiFi()){
     ResetTimer(1);}
 }}
```

Figure 8. Pseudo code of our power-saving scheme

into low power states when a continuous stream of data is being sent or received. Taking these delays into considerations, the time of the uprising edges shown in Figure 6 can be calculated as:

$$T_0 = 16;\ T_{(n+1)} = T_n + 1 + 8;$$

where $T_{(n)}$ is the time to send the nth packet. The amount of time to send 64 bytes of data in various formats from the PIC to the IP2022, from the IP2022 to the Wi-Fi radio and from the Wi-Fi radio to the access point is too small to be recognized under the displayed time scales. Figure 7, a zoomed in view of Figure 6

indicates that it takes only about 30ms (16.37s-16.4s) to wake up the IP2022 and fully power up the radio from the 802.11b power saving doze state.

Our last experiment serves to measure packet round trip delays when a continuous stream of data is being sent to our WGN running the *power-saving* scheme. To measure the delays, we implement on the application processor an echo server that listens on a predefined port and echoes incoming UDP packets back to the sender. Round trip time is calculated as the difference between the sending time, which is encoded as part of the probe packet being sent, and the receiving time recorded when the echoed probe is being received. As shown in Figure 9, after an initial delay of 6 seconds, the wait time is eliminated completely because both the radio and the application processor are kept alive for 1 second by each packet sent or received. As long as packets come in or out with an inter-packet delay of less than 1 second, the WGN will not return back to the *power-saving* mode.

6.4 Average System Power Consumption and Latency

Average system power consumption is calculated[1] based on the amount of energy consumed in one working-period, which is defined as the period from the beginning of one active period to the beginning of the next active period. For example, the period from 16s to 25s in Figure 6 is a working-period with a duration of 9 seconds. A power managed working-period can be further divided into four sub-periods: a period to enable the system, a period of doing the real work, a period to disable the system and a sleep period. A duty-cycle is calculated as the percentage of the time that a system does real work over an entire working-period. It does not include time spent in enabling or disabling the system. Note that we can maintain a fixed duty cycle by proportionally changing the working time and the duration of the working-period.

Table 1 lists the durations of these periods as well as the corresponding power consumptions in these periods for both the *Stargate* and our system running different power management schemes. We choose to compare our platform with the *Stargate* because it is one of a few gateway nodes commonly used in sensor networks. Other gateway nodes, such as those based on the Soekris board [8], share similar designs as the *Stargate*.

For our WGN, the enable-power of the *power-gating* scheme is less than that of the *power-saving* scheme because not all components are powered up at the same time. As shown in Figure 4, there is actually a long delay before the WGN reaches full power. The high sleep-power of our *power-saving* scheme is caused by the limitations of the Intersil chip as explained in section 6.2.

In the remainder of this section, we compare the performance of our power management schemes using the WGN with the performance of the following two commonly used schemes on the *Stargate*:

- Suspend-Wi-Fi Scheme: Suspend the Wi-Fi radio only.

[1] Average System Power = (Total energy consumed in one working-period) / (Duration of the working-period)

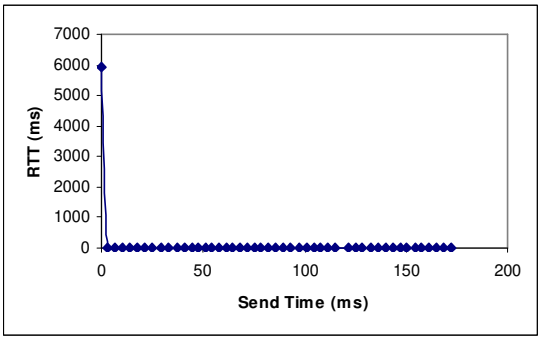

Figure 9. Round trip delay with a continuous stream of data

Table 1
Latencies and Power consumptions

	Stargate		Our WGN		
	Suspend Wi-Fi scheme	Suspend system scheme	Always on scheme	Power gating scheme	Power saving scheme
Enable Latency	0.485s	3.329s	–	0.28s	0.03s
Enable Power	0.751w	0.155w	–	0.545w	0.693w
Active Power	2.009w	2.009w	1.419w	1.419w	1.419w
Disable Latency	0.313s	0.757s	–	0s	0.003s
Disable Power	1.62w	1.11w	–	1.419w	1.32w
Sleep Power	0.751w	0.054w	–	5.13μw	0.495w

- Suspend-System Scheme: Suspend both the Wi-Fi radio and the *Stargate* computer.

The *Stargate* data are based on measurements from [11]. We combine the latencies that are reported separately for the Wi-Fi radio and the PXA255 and compute the average power consumption. Similar to our approach, the authors in [11] measure data without sensors attached. Accordingly, we use the "Processor Core Idle" data for the Suspend-Wi-Fi scheme and the "Proc./Radio Core Sleep" data for the Suspend-System scheme. The active power consumption in the active period is based on 50% TX and 50% RX. For our own schemes, the power consumed by the entire WGN is reported. The load on the 1Ohm resistor is included to simulate a real sensor. Our always-on scheme simply keeps the system in maximum power all the time and is used to serve as a baseline. The sleep-power of our *power-gating* scheme is the same as the sleep-power of the attached sensor node. Since it is too small to be measured accurately, we simply plug the standby power consumption of the Mica2 mote from [15] in the table.

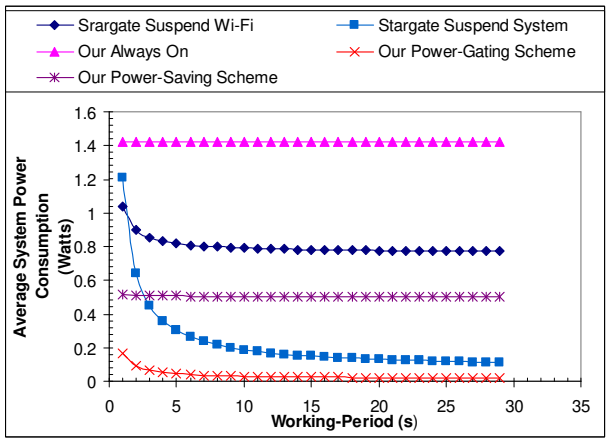

Figure 10. Average system power consumption under various working-periods and at a fixed 1% duty cycle

Figure 10 shows the average system power consumption of the five schemes running under various working-periods and at a fixed 1% duty cycle, which is very common for sensor networks. With a fixed duty cycle, the time spent doing real work increases in proportion to the duration of the working-period, and therefore the average amount of real work per unit time remains constant. Large duty-cycle latencies mean less sleep time. The WGN running our *power-gating* scheme performs about 6 times better than the *Stargate* running the suspend-system scheme for large working-periods where the active power dominates. For short working-periods where the transition (enable/disable) power becomes dominant, we perform up to 7 times better. Although not shown in the figure, when the sleep-power becomes dominant, the comparative performance of our system is even better. For instance, if we only duty cycle the nodes once a day for an active period of 1 second, our WGN will perform 778 times better than the *Stargate* in terms of average system power consumption. The *Stargate* has very high sleep-power because its PXA255 processor is in charge of power management and can not be powered off completely. As shown in Figure 5, even if we could power off the *Stargate* processor completely, the large latency to activate the processor from power-off mode would be prohibitive for many sensor network applications, not to mention the significant energy overhead associated with the delay. Our WGN running the *power-saving* scheme also outperforms the *Stargate* running the suspend-Wi-Fi scheme.

Although it is possible to reduce the active-power of the *Stargate* processor to be close to that of our WGN by dynamic voltage scaling (DVS), our WGN would still perform better because of lower duty-cycle latencies. This is a direct result of the new architecture we propose. To demonstrate its benefits, while normalizing away the specific hardware choices, we consider the *Stargate* schemes assuming the same hardware power numbers as our own schemes. The results are shown in Figure 11, which corresponds to Figure 10, but is normalized in terms of hardware numbers and illustrates the gains specifically due to our architecture.

Figure 12 shows the lifetimes of our WGN and the *Stargate* running the five schemes under various working-periods and at a fixed 1% duty cycle. They are computed based on a power supply

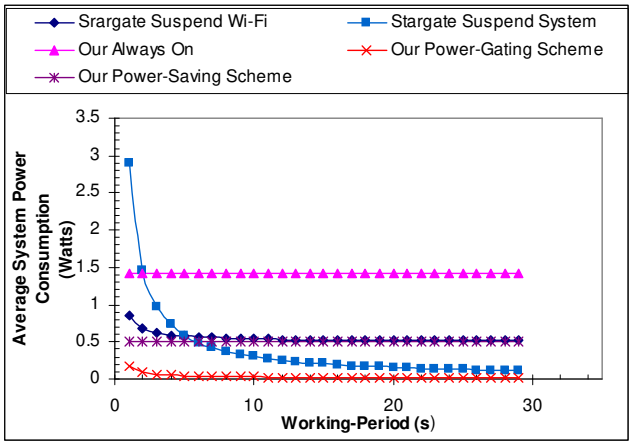

Figure 11. Average system power consumption based on normalized power under various working-periods and at a fixed 1% duty cycle

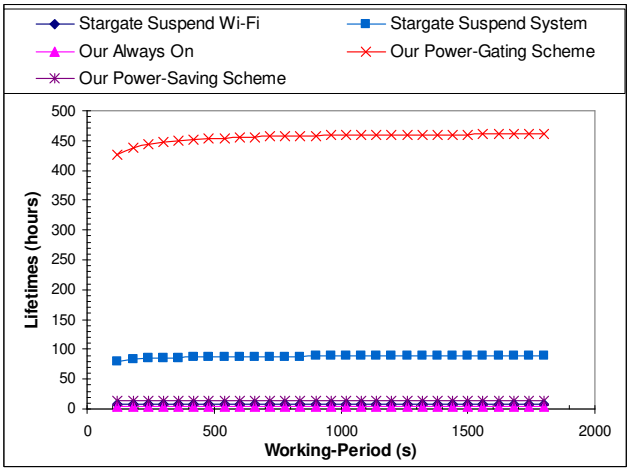

Figure 12. Lifetimes under various working-periods and at a fixed 1% duty cycle (2200 mAh at 3 volts)

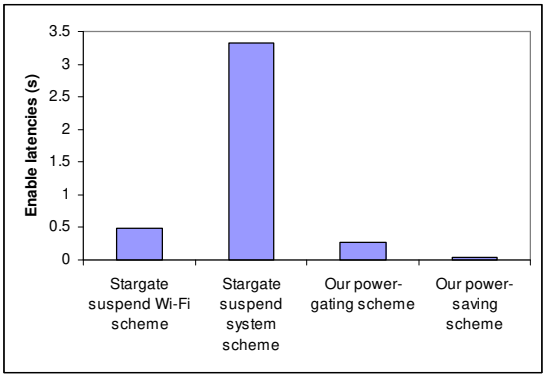

Figure 13. System response time

of 2200 mAh at 3 volts from a pair of AA batteries. The WGN could last longer with larger working-periods and smaller duty cycles because of the extremely low sleep-power.

Another aspect of the performance is the system response time which is critical for certain applications. Figure 13 shows the

system response time under different power management schemes. When running power-gating scheme, our WGN is about 12 times better than the *Stargate* running the suspend-system scheme. The WGN running the power-saving scheme is about 16 times better than the WGN running the suspend-Wi-Fi scheme.

7. CONCLUSIONS

In this paper, we present the design and optimizations of a low power gateway node. By introducing a dual-processor hardware architecture and a choice of appropriate duty-cycling of the processing and radio subsystems, we successfully reduce the standby power consumption while also providing support for spurts of high MIPS and high bandwidth connections. We are also able to reduce duty-cycling related costs by using a lightweight OS and through careful system integration. The result is a platform that supports efficient power management through duty-cycling. The WGN running our power-gating scheme performs about 6 times better in terms of average system power consumption than the *Stargate* running the suspend-system scheme for large working-periods where the active power dominates. For short working-periods where the transition (enable/disable) power becomes dominant, we perform up to 7 times better. The comparative performance of our system is even greater when the sleep power dominates.

8. REFERENCES

[1] Agarwal, Y., Schurgers, C. and Gupta, R., Dynamic Power Management using On Demand Paging for Networked Embedded Systems. in *Asia South Pacific Design Automation Conference (ASP-DAC'05)*, (China, 2005), 755-759.

[2] Arora, A., Ramnath, R., Ertin, E., Sinha, P., Bapat, S., Naik, V., Kulathumani, V., Zhang, H., Cao, H., Sridharan, M., Kumar, S., Seddon, N., Anderson, C., Herman, T., Trivedi, N., Zhang, C., Nesterenko, M., Shah, R., Kulkarni, S., Aramugam, M., Wang, L., Gouda, M., Choi, Y.-r., Culler, D., Dutta, P., Sharp, C., Tolle, G., Grimmer, M., Ferriera, B. and Parker, K. ExScal: Elements of an Extreme Scale Wireless Sensor Network *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05) - Volume 00*, IEEE Computer Society, 2005.

[3] Banerjee, N., Sorber, J., Corner, M.D., Rollins, S. and Ganesan, D. Triage: A Power-Aware Software Architecture for Tiered Microservers *Technical Report 05-22. University of Massachusetts-Amherst*, Amherst, MA, 2005.

[4] *Stargate*, http://www.xbow.com

[5] Dam, T.v. and Langendoen, K. An adaptive energy-efficient MAC protocol for wireless sensor networks *Proceedings of the 1st international conference on Embedded networked sensor systems*, ACM Press, Los Angeles, California, USA, 2003.

[6] *DPAC*, www.dpactech.com.

[7] *eCOS*, http://ecos.sourceware.org/.

[8] Hartung, C., Han, R., Seielstad, C. and Holbrook, S. FireWxNet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments *Proceedings of the 4th international conference on Mobile systems, applications and services*, ACM Press, Uppsala, Sweden, 2006.

[9] *IEEE 802.11b 1999*, http://grouper.ieee.org/groups/802/11.

[10] Jejurikar, R. and Gupta, R., Optimized slowdown in real-time task systems. in *the 16th Euromicro Conference on Real-Time Systems (ECRTS'04)*, (2004), 155-164.

[11] Margi, C.B., Petkov, V., Obraczka, K. and Manduchi, R., Characterizing Energy Consumption in a Visual Sensor Network Testbed. in *2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities* (2006).

[12] McIntire, D., Ho, K., Yip, B., Singh, A., Wu, W. and Kaiser, W.J., The low power energy aware processing (LEAP)embedded networked sensor system. in *the Fifth International Conference on Information Processing in Sensor Networks*, (Nashville, Tennessee, USA, 2006), ACM Press, 449-457.

[13] Pering, T., Raghunathan, V. and Want, R. Exploiting Radio Hierarchies for Power-Efficient Wireless Device Discovery and Connection Setup *Proceedings of the 18th International Conference on VLSI Design (VLSID'05)*, IEEE Computer Society, 2005.

[14] Polastre, J., Hill, J. and Culler, D. Versatile low power media access for wireless sensor networks *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ACM Press, Baltimore, MD, USA, 2004.

[15] Polastre, J., Szewczyk, R. and Culler, D., Telos: Enabling Ultra-Low Power Wireless Research. in *the Fourth International Conference on Information Processing in Sensor Networks*, (2005).

[16] Pottie, G.J. and Kaiser, W.J. Wireless integrated network sensors. *Commun. ACM*, *43* (5). 51-58.

[17] Raghunathan, V., Pering, T., Want, R., Nguyen, A. and Jensen, P., Experience with a low power wireless mobile computing platform. in *the 2004 international symposium on Low power electronics and design*, (Newport Beach, California, USA, 2004), ACM Press, 363-368.

[18] Rahimi, M., Baer, R., Iroezi, O.I., Garcia, J.C., Warrior, J., Estrin, D. and Srivastava, M. Cyclops: in situ image sensing and interpretation in wireless sensor networks *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ACM Press, San Diego, California, USA, 2005.

[19] Schott, B., Bajura, M., Czarnaski, J., Flidr, J., Tho, T. and Wang, L. A modular power-aware microsensor with >1000X dynamic power range *Proceedings of the 4th international symposium on Information processing in sensor networks*, IEEE Press, Los Angeles, California, 2005.

[20] Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D. and Hong, W. A macroscope in the redwoods *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ACM Press, San Diego, California, USA, 2005.

[21] *Ubicom IP2022*, http://www.ubicom.com.

[22] Ye, W., Heidemann, J. and Estrin, D., An Energy-Efficient MAC Protocol for Wireless Sensor Networks. in *Infocom '02,* , (New York, NY, 2002), 1567-1576.