International Journal of Computer Science and Engineering (IJCSE)
ISSN 2278-9960
Vol. 2, Issue 3, July 2013, 69-80
© IASET

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

SURVEY ON SYSTEM ARCHITECTURE FOR WIRELESS SENSOR NETWORKS

RAHMAAN K[1], ANBUMANI P[2] & NARENDRAN M[3]

[1]Final Year, Tagore Institute of Engineering and Technology, Deviyakurichi, Tamil Nadu, India

[2,3]Assistant Professor, Tagore Institute of Engineering and Technology, Deviyakurichi, Tamil Nadu, India

ABSTRACT

In this paper we present and operating system and three generations of a hardware platform designed to address the needs of wireless sensor networks. In this operating system, called Tiny OS uses an event based execution model to provide support for fine-grained concurrency and incorporates a highly efficient component model; Tiny OS enables us to use a hardware architecture that has a single processor time shared between both application and protocol processing. It's show how a virtual partitioning of computational resources not only leads to efficient resource utilization but allows for a rich interface between application and protocol processing. In that rich interface, in turn, allows developers it's exploit application specific communication protocols that significantly improve system performance. Hardware platforms we develop are used to validate a generalized architecture that is technology independent. This general architecture contains a single central controller that performs both application and protocol-level processing. It's flexibility, this controller isdirectly connected to the RF transceiver. It's efficiency, the controller is supported by a collection of hardware acceleratorsthat provide basic communication primitives that can be flexibility composed into application specific protocols.

KEYWORDS: Wireless Sensor Networks, TinyOS, AM Communication Paradigm, Data Collection

INTRODUCTION

The emerging field of wireless sensor networks combines sensing, with computation, and communication into a single tiny device. It's through advanced mesh networking protocols, in this devices form a sea of connectivity that extends the reach of cyberspace out into the physical world. Water flows to fill every room of a submerged ship, mesh networking connectivity will seek out and exploit any possible communication path by hopping data from node to node in search of its destination. The capabilities of any single device are minimal the composition of hundreds of devices offers radical new technological possibilities. The power of wireless sensor networks lies to the ability to deploy large numbers of tiny nodes that assemble and configure themselves. The usage scenarios for these devices range from real-time tracking, it's monitoring of environmental conditions, ubiquitous computing environments, to situ monitoring of the health of structures or equipment. Often referred to as wireless sensor networks, they can also control actuators that extend control from cyberspace into the physical world. The most straightforward application of wireless sensor network technology to monitor remote environments for low frequency data trends. For example, a chemical plant could be easily monitored for leaks by hundreds of sensors that automatically form a wireless interconnection network and immediately report the detection of chemical leaks.

Unlike traditional wired systems, deployment costs would minimal. Instead of having to deploy thousands of feet of wire routed through protective conduit, installers simply have to place quarter-sized device, such as the one pictured in at each sensing point. The network could be incrementally extended by simply adding more devices – no rework or complex configuration. With in this devices presented in this paper , the system would be capable of monitoring for anomalies for several years on a single set o batteries. In addition to reducing the installation costs, wireless sensor networks have the ability to dynamically adapt to changing environments. Adaptation mechanisms can respond to changes in network topologies or can cause the network to shift between different modes of operation. Example, the same embedded network performing leak monitoring in a chemical factory might be reconfigured into a network designed to localize the source of a leak and track the diffusion of poisonous gases.

Figure 1: DOT – Wireless Sensor Network Device Deliberate to be the Estimated Size of Area

The network could then direct workers to the safest path for emergency emigration. Current wireless systems only scratch the surface of possibilities emerging from the amalgamation of low-power announcement, sensing, energy storing, and computation.

TinyOS has been designed to run on a generalized architecture where a single CPU is shared between application and protocol handling. We detail three generations of wireless nodes and a host of application deployments that have verified the capabilities of our general system architecture. The Mica platform has been produced in the largest quantities – over 5000 Mica nodes have been produced and distributed to over 250 companies and research establishments from around the world. The Mica platform includes a low power transceiver, power management subsystem extended storage and embedded microcontroller. The most advanced hardware platform we present is a single-chip CMOS device that integrates the processing speed , storage and communication capabilities to form a complete classification node. This single chip node – called Spec – measures just 2.5 mm x 2.5 mm, it's contains a micro controller, with transmitter, ADC, general resolve I/O ports, UART, memory and encryption engine. To the tiny chip only needs to be supported by a 32 KHz watch crystal, an off-chip inductor and a power supply a battery and a 4 MHz clock. The Spec node represents the approaching generation of wireless sensor nodes that will be manufactured for currencies and deployed in the millions.

WIRELESS SENSOR NETWORKS

The concept of wireless sensor networks is based on a simple equation:

Sensing + CPU + Radio = Thousands of potential applications

As soon as people understand the capabilities of a wireless sensor network, hundreds of applications spiral to mind. It seems like a straightforward combination of modern equipment. However actually combining sensors radios and CPU’s into an operative wireless sensor network requires a detailed understanding of the both capabilities and limitations of each of the fundamental hardware components, as well as a detailed understanding of modern schmoozing technologies and dispersed structures theory. Each individual node must be designed to deliver the set of primitives necessary to synthesize the interconnected web that will emerge as they are positioned, while meeting severe requirements of size, cost and power consumption. A core experiment is to map the overall system requirements down to individual device capabilities requirements and activities. To make the wireless sensor network vision authenticity architecture must be developed that produces the envisioned applications out of the underlying hardware capabilities.

To develop this system architecture we work since the high level application requirements down through the low-level hardware necessities. In this process we first attempt to understand the set of target submissions. To limit the number of applications that we must consider, we focus on a set of application classes that we believe are demonstrative of a large segment of the potential usage scenarios. We use this set of application classes to explore the system-level necessities that are placed on the overall architecture. From these system-level requirements we can then drill down into the separate node-level requirements. Moreover, we must provide a detailed contextual into the capabilities of modern hardware. After we present the raw hardware competences, we present a rudimentary wireless sensor node. The Rene node represents a first censored at system architecture, and is used for comparison against the system architectures obtainable in later chapters.

Sensor Network Application Classes

Three application classes we have selected:

- Environmental data collection
- Security monitoring
- Sensor node tracking.

Environmental Data Collection

At the network level, the environmental data collection application is categorized by having a large number of nodes continually sensing and transmitting data back to a set of base stations that store the data using traditional methods. These networks generally require very low data rates and tremendously long lifetimes. In typical usage scenario, the nodes will be evenly dispersed over an outdoor environment. This distance between adjacent nodes will be minimal yet the distance across the whole network will be significant. After placement, the nodes must first discover the topology of the network and estimate optimum routing strategies [10].

The routing strategy can then be used to direction data to a central collection points. In ecological monitoring applications, it is not indispensable that the nodes develop the optimal routing strategies on their own. Instead it may be possible to calculate the optimal routing topology outside of the network and then interconnect the necessary information to the nodes as required. This is possible because the physical topology of the network is moderately constant. While the time variant nature of RF announcement may cause connectivity between two nodes to be alternating, the overall topology of the network will be moderately stable. Environmental data collection applications typically use tree-based routing topologies where each routing tree is entrenched at high-capability nodes that sink data. Data is occasionally transmitted from child node to parent node up the tree-structure until it reaches the sink. By tree-based data collection each node is responsible for systematic the data of its entire offspring. Nodes with a large number of pro genies transmit significantly more data than leaf nodes. These nodes can rapidly become energy bottlenecks [11, 12]. Once the network is configured, each node occasionally samples its sensors and transmits its data up the routing tree and back to the base station. For many scenarios, the interval between these programs can be on the order of minutes.

Typical reporting periods are expected to be between 1 and 15 minutes; while it is possible for networks to have meaningfully higher reporting rates. The typical environment parameters being monitored, such as illness, light concentrations, and moisture, do not change quickly sufficient to require higher reporting rates. In addition to huge sample intervals, environmental monitoring applications do not have severe latency requirements. Data samples can be behind inside the network for moderate periods of time without significantly affecting application performance. In general the data is collected for future inquiry not for real-time operation. In order to meet lifetime supplies each communication event must be precisely scheduled. The senor nodes will remain latent a majority of the time; they will only awaken to transmit or receive data. If the precise schedule is not met the communication proceedings will fail.

Security Monitoring

Additionally, it is essential that it is confirmed that each node is still contemporary and functioning. If a node were to be disabled or fail it would signify a security violation that should be reported. For security monitoring solicitations the network must be configured so that nodes are responsible for authorizing the status of each other. One approach is to have each node be allocated to peer that will report if a node is not functioning. The optimal topology of a security monitoring network will look moderately different from that of a data collection network.

In a collection tree each node must convey the data of all of its decedents. Because of this it is finest to have a short wide tree. In contrast, with a security network the optimum configuration would be to have a linear topology that forms a Hamiltonian cycle of the network. The power consumption of each node is only comparative to the number of children it has. In a linear system, each node would need only one child. This would evenly distribute the energy depletion of the network. The accepted norm for security systems today is that each sensor should be checked nearly once per hour. Combined with the ability to evenly dispense the load of checking nodes, the energy cost of performing this check becomes negligible. A majority of the energy depletion in a security network is spent on meeting the strict latency requirements associated with the signalling the alarm when a security violation happens. Once detected a security violation must be connected to the base station immediately. The latency of the data announcement across the network to the base station has a critical impact on application performance. Users demand that alarm circumstances be reported within seconds of detection. This means that network nodes must be able to re-join quickly to requests from their neighbours to forward data.

In security networks reducing the latency of an alarm transmission is meaningfully more important than reducing the energy cost of the transmissions. This is because alarm events are predictable to be rare. In a fire security system alarms would nearly never be signalled. In the event that one does occur a significant amount of energy could be enthusiastic to the transmission. Reducing the transmission inexpression leads to higher energy consumption because routing nodes must monitor the radio channel more frequently. In security networks a massive majority of the energy will be spend on confirming the functionality of neighbouring nodes and in being organized to instantly forward alarm pronouncements. Actual data transmission will consume a small segment of the network energy.

Sensor Node Tracking

A third usage scenario commonly discussed for sensor networks is the chasing of a tagged object through a region of space monitored by a sensor network. There are many circumstances where one would like to track the location of valuable assets or personnel. Current uncatalogued control systems attempt to track objects by recording the last checkpoint that an object passed through. However with these systems it is not possible to regulate the current location of an object. For example UPS pathways every shipment by scanning it with a bar code whenever it passes through a routing center. The system breaks down when objects do not flow from barrier to barrier. In typical work environments it is unreasonable to expect objects to be continually passed through checkpoints.

With wireless sensor networks objects can be chased by simply tagging them with a small sensor node. The sensor node will be tracked as it moves complete a field of sensor nodes that are deployed in the environment at known locations. Instead of sensing ecological data these nodes will be deployed to sense the RF messages of the nodes attached various objects. The nodes can be used as active tags that broadcast the presence of device. A database can be used to record the location of chased objects relative to the set of nodes at known location. With this system it becomes possible to ask anywhere an object is currently, not just where it was last scanned [13]. Unlike sensing or security networks node tracking applications will repeatedly have topology changes as nodes move through the network. While the connectivity amongst the nodes at fixed locations will remain relatively stable, the connectivity to mobile nodes will be repeatedly changing.

Additionally the set of nodes being tracked will recurrently change as objects enter and leave the system. It is necessary that the network be able to efficiently detect the presence of new nodes that enter the network.

SYSTEM EVALUATION METRICS

One result is that many of these evaluation metrics are consistent. Often it may be necessary to decrease presentation in one metric such as sample rate in order to increase another such as lifetime. Taken composed, this set of metrics form a multidimensional space that can be used to designate the capabilities of a wireless sensor network. The capabilities of a platform are characterized by a volume in this multidimensional space that contains all of the valid effective points. In turn a specific application deployment is represented by a particular point. A system platform can successfully perform the submission if and only if the application requirements point lies inside the capability hyperspace.

Lifetime

Dangerous to any wireless sensor network deployment is the expected lifetime. The goal of both the ecological monitoring and security application scenarios is to have nodes placed out in the arena unattended for months or years. The main limiting factor for the lifetime of a sensor network is the energy supply. Each node must be designed to manage its local source of energy in order to maximize total network lifetime. In many placements it is not the average node lifetime that is significant but rather the minimum node lifetime. In the case of wireless safety systems, every node necessity last for multiple years. A single node failure would create susceptibility in the security systems.

Coverage

Tied to range is a linkage’s ability to scale to a large number of nodes. Scalability is a key component of the wireless sensor network value proposal. A user can deploy a small trial network at first and then can recurrently add sense points to collect more and different information. A user must be confident that the network knowledge being used is capable of scaling to meet his eventual need. Increasing the number of nodes in the system will influence either the lifetime or effective sample rate. More sensing points will cause more data to be transmitted which will increase the power ingesting of the network. This can be offset by specimen less often.

Cost and Ease of Deployment

A key improvement of wireless sensor networks is their ease of deployment. Environmentalists and construction workers installing networks cannot be expected to understand the underlying networking and communication mechanisms at work inside the wireless network. For system deployments to be effective the wireless sensor network must configure itself. It must be possible for nodes to be placed throughout the situation by an untrained person and have the system simply work. Ideally the system would mechanically configure itself for any possible physical node placement. However, real systems must place restraints on actual node placements – it is not possible to have nodes with immeasurable range. The wireless sensor network must be capable of providing response as to when these constraints are violated.

The network should be able to assess quality of the network positioning and indicate any potential problems. This translates to requiring that each device be capable of performing link discovery and influential link quality**.**

Response Time

The ability to have low response time conflicts with many of the procedures used to increase network lifetime. Network lifetime can be increased by having nodes only operate their wirelesses for brief periods of time. If a node only opportunities on its radio once per minute to transmit and receive data, it would be impossible to meet the application necessities for response time of a security system.

Response time can be better-quality by including nodes that are powered all the time. These nodes can listen for the alarm messages and forward them down a routing backbone when essential. This, however, reduces the comfort of deployment for the system.

SOFTWARE ARCHITECTURE FOR WIRELESS

TinyOS draws strongly from previous architectural work on lightweight filament support and efficient network interfaces. Included in the TinyOS system architecture is an Active Communications system. We believe that there is an important fit between the event based nature of network sensor applications and the event created primitives of the Active Messages communication model. In operational with wireless sensor networks two issues emerge strongly these devices are concurrency concentrated several different flows of data must be kept moving concurrently and the system must provide efficient modularity hardware specific and application specific mechanisms must snap together with little processing and storage overhead. We address these two problems in the context of current system sensor knowledge and the tiny micro threaded OS. Analysis of this solution provides valuable initial instructions for architectural innovation.

Tiny Micro Threading Operating System(TinyOS)

Surviving embedded device operating systems do not meet the size power and efficiency requirements of this regime. These requirements are amazingly similar to that of building efficient network interfaces which also must maintain a large number of concurrent flows and manipulate numerous outstanding events [19]. In network boundary cards, these issues have been tackled through corporal parallelism [11] and virtual machines [12]. We tackle it by building an extremely efficient multithreading engine. As in TAM [12] and CILK [13], TinyOS maintains a two-level scheduling structure so a small amount of dispensation associated with hardware events can be performed directly while long running tasks are interrupted. The execution model is similar to finite state mechanism models, but noticeably more programmable**.**

TinyOS is designed to scale with the current technology trends supporting both smaller, tightly integrated designs, as well as the border of software components into hardware. This is in contrast to old-fashioned notions of scalability that are centered on scaling up total power/resources/work for a given computing pattern. It is essential that software architecture plans for the eventual incorporation of sensors, processing and announcement. In order to qualify the vision of single-chip a low cost sensor node TinyOS combines an exceedingly efficient execution model component model and communication devices

TinyOS Component Model

In addition to using the highly effectual event-based execution, TinyOS also includes a specially designed component model directing highly effectual modularity and easy composition. An efficient component model is indispensable for embedded systems to increase reliability deprived of sacrificing performance. The component model allows an application designer to be able to easily combine independent components into an application specific configuration.

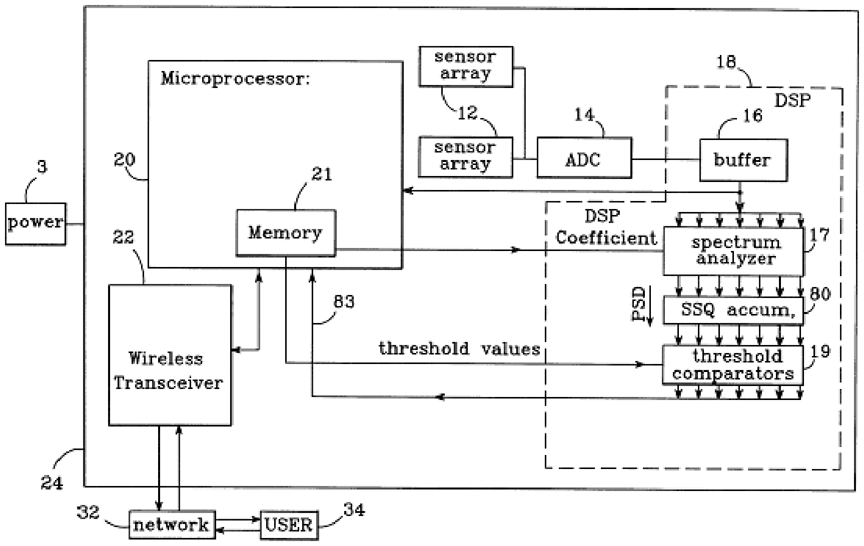

Figure 2: Component Diagram for Sensing Application

In TinyOS, each module is defined by the set of instructions and events that makes up its interface. In turn, a complete system requirement is a listing of the components to include plus a specification for the interconnection between components. The TinyOS constituent has four interrelated parts a set of command handlers a set of event handlers an encapsulated private data frame, and a bundle of simple tasks. Responsibilities commands and event handlers execute in the context of the frame and operate on its state. To simplify modularity each component also declares the commands it uses and the events it signals. These announcements are used to facilitate the composition process. As shown in Figure 2, composition creates a graph of components where high level components issue commands to lower level apparatuses and lower level apparatuses signal events to the higher level components. The lowest layer of apparatuses interacts directly with the essential hardware.

A specialized language NESC has been commercial to express the component graph and the command/event boundaries between components. In NESC multiple understanding and events can be grouped together into interfaces. Interfaces simplify the interconnection between apparatuses. In TinyOS, storage frames are statically allocated to allow the memory necessities of the complete application to be unwavering at compile time. The frame is a dedicated C Structure that is statically allocated and directly accessible only to the component. While TinyOS does not have memory fortification variables cannot be directly accessed from outside of a constituent. In addition to allowing the calculation of maximum memory requirements pre-allocation of frames prevents the overhead associated with active allocation and avoids pointer related errors. This savings establishes itself in many ways, including performance time savings because variable locations can be statically compiled into the program instead of accessing state via pointers.

In TinyOS, instructions are non-blocking requests made to lower level components. Characteristically a command will deposit request parameters into its local frame and provisionally post a task for later execution. It may also invoke lower instructions but it must not wait for long or indeterminate latency actions to take place. A appreciation must provide feedback to its caller by returning status indicating whether it was effective or not, e.g., buffer overrun. Event trainers are invoked to deal with hardware events either directly or indirectly. The lowest level components have supervisors that are connected directly to hardware interrupts which may be exterior interrupts timer events or counter events. An event handler can deposit data into its frame, post tasks, signal advanced level events or call lower level commands. A hardware event triggers a cascade of processing that goes upward through events and can bend downward through commands. In order to avoid cycles in the command/event chain instructions cannot signal events. Both commands and events are intended to perform a small fixed amount of work which happens within the context of their component's state.

COMPONENTS CONTAINED IN TINYOS

There are several system level components that are involved in TinyOS. Moreover, there are a collection of example presentations that demonstrate the usages of the TinyOS system.

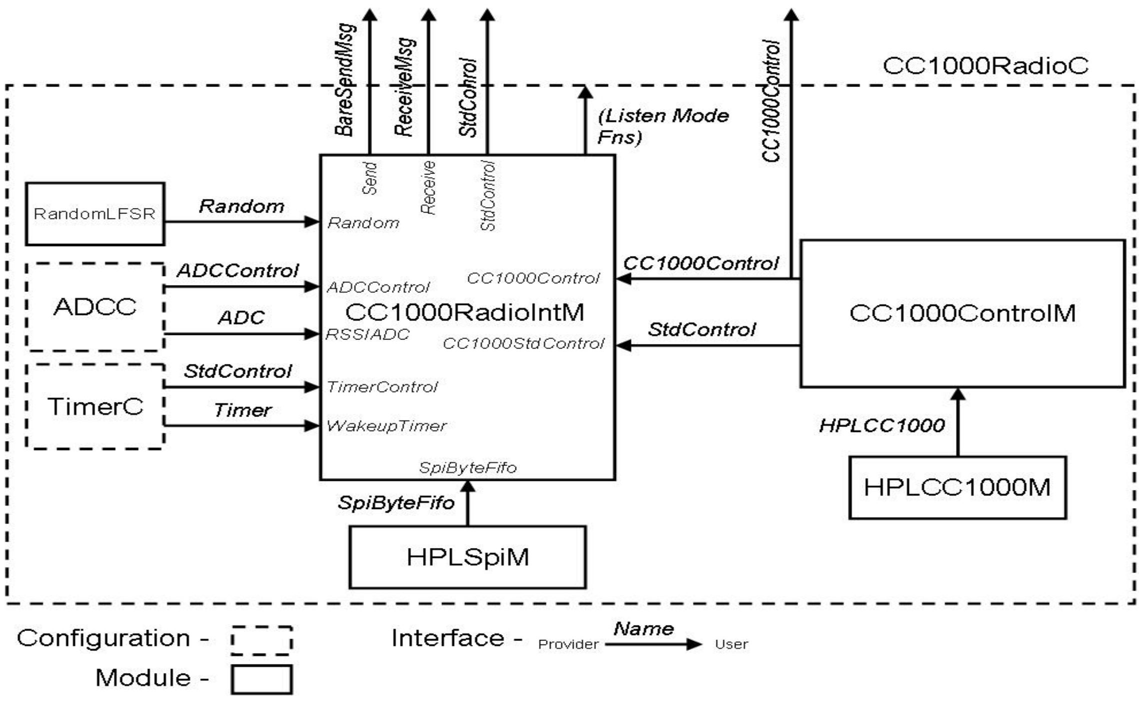

Figure 3: Components Contained in TinyOS

We have developed a generalized design for wireless sensor nodes that builds from the single controller design used in Rene. The architecture is based on the principle that shared pools of resources should be used when possible to exploit the benefits of dynamic allocation, that buffering needs be used to decouple the general purpose data path and the radio, and that protocol plasticity is essential. It addresses performance and efficiency issues by including special-purpose hardware accelerators for management the real-time, high-speed necessities of the radio. Accelerators deliver general building blocks, not comprehensive solutions. The core of the architecture is a central computational engine that is timeshared across submission and protocol processing. Only a single calculation engine is included because it allows the allocation of all processing resources to a single task when necessary. We show that this allows for an efficient use of dispensation capabilities. Ideally this single processor includes extra hardware provision for the fine-grained concurrency that it must provide. As this processor is intended to be allotted to multiple concurrent operations it must be designed so that framework switching is as efficient as possible. A traditional mechanism for decreasing context switch overhead is through the inclusion of register openings. Multiple register sets can be included in the CPU so that each context switch does not require the indexes to be written out to memory. Instead the operating system basically switches to a free register set. As is typical in microcontroller designs, the data path is connected to the rest of the system components through a shared interconnect.

Memory I/O ports analogue-to-digital converters, system timers and hardware accelerators are devoted to this interconnects. By utilizing a high-speed, low latency interconnect, data can be moved easily between the processor, memory and peripheral devices. In addition to allowing the CPU to interact with its peripheral devices, this communal interconnect also allows the separate peripheral devices to interact with each other. A peripheral placed on this bus has the ability to wrench data directly out of the memory subsystem or to push data into a UART marginal. This creates a highly plastic system where a data encoding peripheral can pull data directly from memory and push it into a data transmission accelerator, such as modulation of an RF announcement channel. In such a system, the CPU is simply composing the data transmission it does not have to directly handle the data. All devices on this shared interrelate operate through a shared memory interface.

Each device has control structures that are charted into a shared address space. This allows apparatuses that were not originally intended to function together to be combined in new and interesting ways. A data encoder intended to read from memory, transform data, and write to memory many not even know that it is actually pulling data from a radio receiver block's memory interface and assertive it into a UART's portion of the communication memory. In this architecture, the size of the communal address space dedicated to each operation can be set animatedly to meet application requirements. The true power of this system is in the special-purpose hardware accelerators that it qualifies. These accelerators provide efficient applications of low-level operations that are inefficient on a general-purpose data path. Each accelerator is designed to provide support for operations that are critical to sensor network announcement. By increasing the efficiency of these processes, the overall power ingesting of the system can be greatly reduced. It is important that these accelerators are communication primitives in its place of complete protocol implementations so that the system can support a wide range of announcement protocols instantaneously simply finished software reconfiguration.

The hardware accelerators also support operations that need to be achieved as fast as possible to optimize the radio power consumption. This includes support for start symbol discovery as well as the low-level bit inflection. The goal is to include the minimum hardware functionality that is essential to efficiently support the needs of applications and decouple communication rates from processing rates. While the hardware accelerators are designed to deliver primitives that can be used to construct communication protocols, these primitives are not unavoidably simple. For example, a hardware accelerator for encryption would be considered a primitive.

This component would take the data it remains given and encode or decrypt it as necessary. As a hardware accelerator it could be used for encoded communications, data verification, or to safeguard that data stored in an off-chip flash is kept secure. The hardware accelerator would be applied so that the core cryptographic alteration was unprotected to the CPU and other peripherals. This is in contrast to a system where an entire secure communication subsystem would be encapsulated without exposing any of the internal primitives. Once again this design choice allows for suppleness without surrendering efficiency.

One of the most exciting systems to be assembled on top of the TinyOS platform is one call Tiny DB. Tiny DB is designed to transform a wireless sensor network into spilling database. SQL queries are entered into a user interface and broadcasted onto the sensor network. The network executes the queries over inward sensor readings and returns the results. Just like in normal SQL, these queries can include combination operations to refine the data into averages, max, or min values. The TinyDB query optimizer mechanically distributes the accumulation operations into the network.

CONCLUSIONS

This paper has accessible a system architecture for wireless sensor nodes that is capable of addressing the strict requirements of wireless sensor networks. By exploiting a single shared controller that is amplified by a collection of specialized hardware accelerators, the architecture is able to support stretchy, application-specific communications protocols without surrendering efficiency. This architecture has been validated through the development of three hardware boards and a software operating system.

We have developed the TinyOS operating system which delivers the fine-grained concurrency apparatuses required to implement wireless sensor network protocols and applications. TinyOS influences an event based execution model to efficiently share a single processor across multiple autonomous functional operations. Additionally, TinyOS delivers a highly-efficient constituent model that has almost no runtime overhead yet allows application developers to partition submissions into easy-to-manage modules. This allows for the component modules to be verified self-sufficiently before composing them together into a complete application.

We have presented a comprehensive architecture that addresses key issues that arise when building a wireless sensor network device that must meet strict power consumption and size requirements. They include flexibility, fine-grained concurrency, precise harmonization and decoupling between RF and data path speed. We argue that these properties must be present in the system architecture in order to support wireless sensor network solicitations. The platform must be flexible enough to meet the wide range of application necessities that sensor networks are addressing. We have identified core application situations that range from environmental data collection to security networks to node tracking networks.

Each scenario has substantially different communication patters and protocols that must be supported by particular hardware architecture. To validate our general construction we first presented the Mica node. Composed from off-the- shelf apparatuses, it only approximated our general architecture. It included dedicated hardware accelerators that help decouple the RF and data path speed and increase the synchronization accuracy of communication protocols. In evaluation of this stage we demonstrated how these simple accelerators result in significant performance improvements without losing suppleness. The Mica platform has proven itself both in theory and finished deployment in long-term battery operated application situations.

We have made the leap beyond the competences of off-the-shelf hardware by fully realizing our general architecture in the form a single-chip CMOS device called Spec. Unimpeded by the capabilities of commercially available components we have integrated a suite of hardware accelerators and custom radio in order to realize order-of magnitude improvements on key evaluation metrics. These include critical metrics such as communication rates processing overhead for start symbol detection and timing accuracy.

Spec is representative of the upcoming of wireless sensor network devices. In addition to micro-benchmarks and theoretical analysis, we have also presented real-world application deployments. We have grounded our study by going as far as taking our nodes to the punishment to track military vehicle movement during a live-fire training exercise. We have also tracked scale size vehicles in mock-up situations, and deployed countless-other investigational networks. The Mica platform combined with TinyOS has been transported to over 250 administrations to be the foundation of a countrywide effort into wireless sensor network investigation and development.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their detailed comments.

REFERENCES

1. W.R.Heinzelman, A. Chandrakasan, and H.Balakrishnan,"Energy-Efficient communication protocol for wireless micro sensor networks", in proceedings of HICSS-33, pp.3005-3014, January 2000.

2. N. Labroche, N. Monmarch´e, and G. Venturini, “A new clustering algorithm based on the chemical recognition system of ants,” in Proceedings of ECAI 2002, pp. 345–349, July 2002.

3. A. E. Langham and P. W. Grant, “Using competing ant colonies to solve k-way partitioning problems with foraging and raiding strategies,” in Proceedings of the 5th European Conference on Artificial Life, September 1999.

4. I.Akyilidiz et al.,” Wireless sensor networks: a survey, “Computer Networks, Vol 38, pp.393-422, March 2002

5. Ying Liang Hang Li “An Energy-Efficient Clustering Algorithm for Wireless Sensor Sensor Network” . March 2006.

6. Junpei Kamimura, Naoki Wakamiya, Masayuki Murata,” Energy-Efficient Clustering Method for Data Gathering in Sensor Networks” January 2004.

7. Culler, D.E., J. Singh, and A. Gupta, Parallel Computer architecture a hardware/software approach. 1999.

8. Esser, R. and R. Knecht, Intel Paragon XP/S - architecture and software environment. 1993: Technical Report KFA-ZAM-IB-9305.

9. Culler, D.E., et al. Fine-grain parallelism with minimal hardware support: a compiler-controlled threaded abstract machine. 1991.

10. Blumofe, R., et al., Cilk: An Efficient Multithreaded Runtime System. Proceedings of the 5th Symposium on Principles and Practice of Parallel Programming, 1995.

11. Hu, J., I. Pyarali, and D. Schmidt, Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-speed Networks. Proceedings of the 2 nd Global Internet Conference, IEEE, 1997.

12. Von Eicken, T., et al. Active messages: a mechanism for integrated communication and computation. in 19th Annual International Symposium on Computer Architecture. 1992.

13. Gay, D., et al., The nesC Language: A Holistic Approach to Networked Embedded Systems. 2003: Programming Language Design and Implementation (PLDI).

14. Renesse, R.V., et al., A framework for protocol composition in horus. 1995: Proceedings of the ACM Symposium on Principles of Distributed Computing.

15. Agarwal, A., et al., The MIT alewife machine: A large-scale distributed-memory multi-processor. 1991: Proceedings of Workshop on Scalable Shared Memory Multiprocessors.

16. Montz, A.B., et al., Scout: A communications-oriented operating system. 1995:

17. Hutchinson, N.C. and L.L. Peterson, The x-kernel: An architecture for implementing network protocols. 1991: IEEE Transactions on Software Engineering. p. 17(1):64-76.

18. Want, R., et al., The Active Badge Location System. ORL, 24a Trumpington Street, Cambridge CB2 1QA, 1992.

19. Bahl, P. and V. Padmanabhan, RADAR: An in-building RF-based user location and tracking system. IEEE Infocom, 2000. 2: p. 775

20. Chandrakasan, A.P., S. Sheng, and R.W. Brodersen, Low Power CMOS Digital Design. 1992, University of California Berkeley.

21. Pering, T., T. Burd, and R. Brodersen, The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms. 1998.

22. Ye, W., J. Heidemenn, and D. Estrin, An Energy-Efficient MAC Protocol for Wireless Sensor Networks. 2001: Submitted for review, July 2001.

23. Stemm, M., et al. Reducing power consumption of network interfaces in handheld devices. in International Workshop on Mobile Multimedia Communications (MoMuc-3). 1996. Princeton, NJ.

24. Lamport, L., Time, clocks, and the ordering of events in a distributed system. Comm., 1978. ACM 21(7): p. 558-565.

25. Mills, D.L., Internet time synchronization: the Network Time Protocol. IEEE Trans. Communications, 1991. COM-39(10): p. 1482-1493.

26. Doherty, L., K.S.J. Pister, and L.E. Ghaoui. Convex position estimation in wireless sensor networks. in IEEE Infocom. 2001: IEEE Computer Society Press.

27. Borriello, J.H.a.G., Location Systems of Ubiuitous Computing. Computer, 2001.34(8): p. 57-66.

28. Priyantha, N.B., A. Chakraborty, and H. Balakrishnan. The Circket Location- Support System. in MobiCom 2000. 2000. Boston, Massachusetts.

29. Molnar, A., Personal Communication. 2004: To be submitted to IEEE International Solid-State Circuits Conference 2004.