Global Journal of Computer Science and Technology
Network, Web & Security
Volume 12 Issue 16 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Designing and Implimentation of Spatial IP Address Assignment Scheme for a Wireless Network

By Fazal Wahab Khattak

ICMS, Hayatabad, Peshawar, Pakistan

Abstract - Wireless sensor networks are composed of large numbers up to thousands of tiny radio- equipped sensors. Every sensor has a small microprocessor with enough power to allow the sensors to autonomously form networks through which sensor information is gathered. Wireless sensor networks makes it possible to monitor places like nuclear disaster areas or volcano craters without requiring humans to be immediately present. Many wireless sensor network applications cannot be performed in isolation; the sensor network must somehow be connected to monitoring and controlling entities. This research paper investigates a novel approach for connecting sensor networks to existing networks: by using the TCP/IP protocol suite in the sensor network, the sensors can be directly connected to an outside network without the need for special proxy servers or protocol converters.

Bringing TCP/IP to wireless sensor networks is a challenging task, however. First, because of their limited physical size and low cost, sensors are severely constrained in terms of memory and processing power. Traditionally, these constraints have been considered too limiting for a sensor to be able to use the TCP/IP protocols. In this research paper, I show that even tiny sensors can communicate using TCP/IP. Second, the harsh communication conditions make TCP/IP perform poorly in terms of both throughput and energy efficiency.

With this research paper, I suggest a number of optimizations that are intended to increase the performance of TCP/IP for sensor networks. The results of the work presented in this research paper have a significant impact on the embedded TCP/IP networking community. The software evolves as part of the research paper has become widely known in the community. The software is mentioned in books on embedded systems and networking, is used in academic courses on embedded systems, is the focus of articles in professional magazines, is incorporated in embedded operating systems, and is used in a large number of embedded devices

GJCST-E Classification : C.2.1

Strictly as per the compliance and regulations of:

Designing and Implimentation of Spatial IP Address Assignment Scheme for a Wireless Network

Fazal Wahab Khattak

Abstract - Wireless sensor networks are composed of large numbers up to thousands of tiny radio- equipped sensors. Every sensor has a small microprocessor with enough power to allow the sensors to autonomously form networks through which sensor information is gathered. Wireless sensor networks makes it possible to monitor places like nuclear disaster areas or volcano craters without requiring humans to be immediately present. Many wireless sensor network applications cannot be performed in isolation; the sensor network must somehow be connected to monitoring and controlling entities. This research paper investigates a novel approach for connecting sensor networks to existing networks: by using the TCP/IP protocol suite in the sensor network, the sensors can be directly connected to an outside network without the need for special proxy servers or protocol converters.

Bringing TCP/IP to wireless sensor networks is a challenging task, however. First, because of their limited physical size and low cost, sensors are severely constrained in terms of memory and processing power. Traditionally, these constraints have been considered too limiting for a sensor to be able to use the TCP/IP protocols. In this research paper, I show that even tiny sensors can communicate using TCP/IP. Second, the harsh communication conditions make TCP/IP perform poorly in terms of both throughput and energy efficiency.

With this research paper, I suggest a number of optimizations that are intended to increase the performance of TCP/IP for sensor networks. The results of the work presented in this research paper have a significant impact on the embedded TCP/IP networking community. The software evolves as part of the research paper has become widely known in the community. The software is mentioned in books on embedded systems and networking, is used in academic courses on embedded systems, is the focus of articles in professional magazines, is incorporated in embedded operating systems, and is used in a large number of embedded devices.

I. Introduction

Wireless sensor networks consist of large numbers of sensors equipped with a small microprocessor, a radio transceiver, and an energy source, typically a battery. The sensors nodes autonomously form network through which sensor readings are transported. Applications of wireless sensor networks can be found in such diverse areas as wild-life habitat monitoring [5], forest fire detection [6], alarm systems [3], medicine [8], and monitoring of volcanic eruptions [12].

In order to make large scale networks feasible, the sensor nodes are required to be physically small and inexpensive. These requirements severely constraints the available resources on each sensor node in terms of memory size, communication bandwidth, computation speed, and energy. Many wireless sensor network applications do not work well in isolation; the sensor network must somehow be connected to monitoring and controlling entities.

Since communication within the sensor network is done using short range radios, a straightforward approach to connecting the sensors with the controlling entities is to deploy the controlling entities physically close to the sensor network. In many cases however, placing those entities close to the sensors, and hence to the phenomenon being observed, is not practical.

Instead, by connecting the sensor network and the controlling entities to a common network infrastructure the sensors and the controlling entities can communicate without being physically close to each other. Because of the success of the Internet, the TCP/IP protocols have become the de-facto standard protocol stack for large scale networking. However, conventional wisdom states that TCP/IP is inherently unsuitable for communication.

Author : ICMS, Hayatabad, Peshawar, Pakistan.
E-mail : pakis007@gmail.com

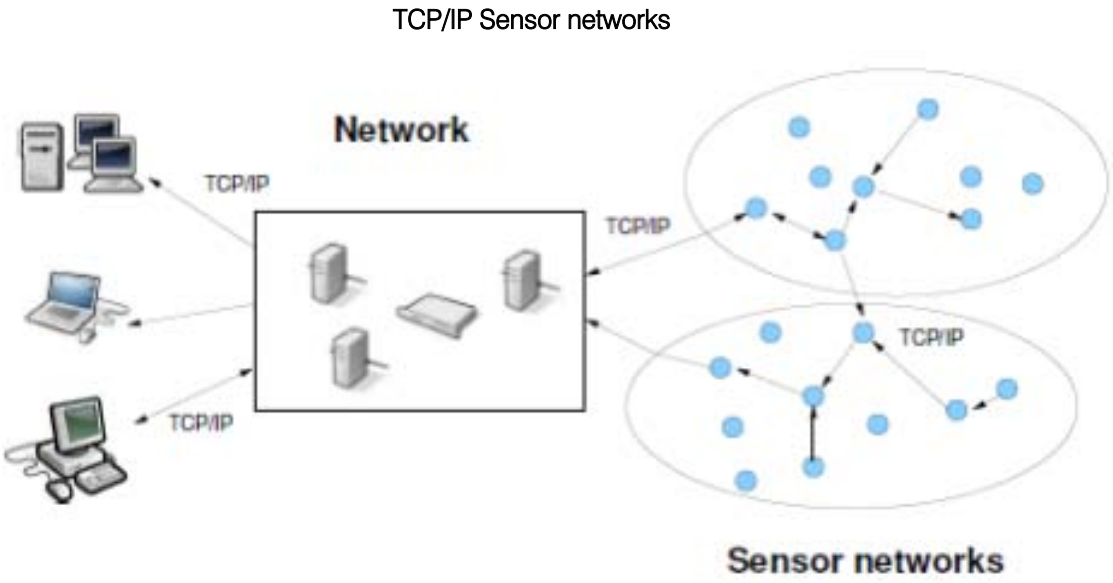

Figure 1.1 : Using TCP/IP both outside of and inside the wireless sensor network

Using TCP/IP both outside of and inside the wireless sensor network station within wireless sensor networks, because of the extreme communication conditions in sensor networks. Hence, a large number of protocols specifically tailored for sensor networks have been developed. While it is unquestionably true that the TCP/IP protocols were not designed to run in the kind of environments where sensor networks are envisioned, the claim that TCP/IP is inherently unsuitable for wireless sensor networks has not been verified.

The purpose of this licentiate research paper is to lay the groundwork for exploring the use of TCP/IP for wireless sensor networks. Using TCP/IP for sensor networks allows connecting the sensor networks directly to IP network infrastructures, as shown in Figure 1.1. In a set of four papers I present the software for an experimental platform, describe the problem area, and propose a set of mechanisms that are intended to allow TCP/IP to be efficiently used in wireless sensor networks.

The software platform consists of a lightweight implementation of the TCP/IP protocol stack and an equally lightweight and flexible operating system. Both the operating system and the protocol implementation are specifically designed to run on resource constrained sensor nodes.

a) Method

In order to explain and motivate the work in this research paper, I use two perspectives: the engineering perspective and the research perspective.

Engineering is about founding solutions to complex problems, within a given set of limitations. Research is about developing understanding. In experimental computer science [20], this understanding commonly is developed by producing artifacts and solving complex problems doing engineering and drawing conclusions from the solutions.

A single solution may not be possible to generalize, but taken together, a number of solutions can be said to span a solution space to a particular problem. Exploring, characterizing, and analyzing this solution space develops understanding for the character of the problem. This classification of engineering and research is based on definitions from Brooks' [15] and Phillips and Pugh [6].

The research in this research paper has mostly been exploratory. The problem area was not defined in advance, but has been developed as part of the research paper work. The exploratory method starts with finding an interesting question to answer. The question usually involves an interesting problem to solve.

The problem is then solved in a set of different ways, using either different tools or methods or variations of the same method. Based on observations of the solutions, or of the process leading to the solutions, an initial answer to the question can be formulated. From the answer and the solutions to the problems, it might be possible to generalize the question into a hypothesis.

This hypothesis can then be tested using experimentation in order to validate or invalidate it. The process of testing the hypothesis typically leads to a number of questions that need to be answered.

Thus the research process is iterative in that a research question leads to a hypothesis, which leads to further questions. In this research paper, the initial question was if the TCP/IP protocol stack could be implemented so that it would fit in a severely memory constrained system. After twice solving the problem of implementing TCP/IP with limited resources, the question could be answered: the TCP/IP protocol stack

can be implemented using very small amounts of memory.

This observation lead to the generalized question if TCP/IP could be useful is wireless sensor networks. This generalization was made because of the similarities of parts of the problem domains sensor network nodes have severely limited memory resources. as well as intuition developed when answering the initial question. The event-driven nature of sensor networks seemed to fit the event-driven design of the small TCP/IP implementations.

Furthermore, it appeared that many of the problems with TCP/IP in sensor networks could be solved with relatively straight-forward mechanisms. These observations lead to the hypothesis that TCP/IP could be a viable alternative for wireless sensor networks. This research paper takes the first steps towards validating or invalidating this hypothesis.

b) Research Issues

This research paper takes the first steps towards the use of the TCP/IP protocol suite in wireless sensor networks. This section summarizes the research issues that are indentured and treated in this research paper. Many of these issues are of the engineering kind: a problem that needs a solution that is not only correct, but also is able to work within the available limitations. These issues are the primary focus of papers A and B. Papers A and B solve the specific problems of implementing TCP/IP on a limited device and on designing an operating system for sensor nodes that allows rapid prototyping and experimentation.

Papers C and D focus on the research challenges involved in TCP/IP for wireless sensor networks. The formulation of these challenges are based on the software artifacts developed in paper A. Paper B presents a software framework designed to support future experimentation.

i. *TCP/IP on a Limited Device*

The TCP/IP protocol suite, which forms the basis of the Internet, is often perceived to be .heavy-weight. in that an implementation of the protocols requires large amounts of resources in terms of memory and processing power. This perception can be corroborated by measuring the memory requirements of popular TCP/IP implementations, such as the one in the Linux kernel [13] or in the BSD operating system [16]. The TCP/IP implementations in these systems require many hundreds of kilobytes of random access memory (RAM).For most embedded systems; cost typically is a limiting factor.

This constrains the available resources such as RAM and processor capabilities. Consequently, many embedded systems do not have more than a few kilobytes of RAM. Within the constraints of such a small embedded system, it is impossible to run the TCP/IP implementations from Linux or BSD Within this research paper, I investigate the solution space to the problem of running TCP/IP within constrained memory limits.

By developing a very small TCP/IP Implementation that is able to run even on a system with very small amounts of memory, I demonstrate that the solution space of the problem is larger than previously shown. While this is not an exhaustive investigation of the solution and, hence, cost are not technical limitations, but functions of business models. It is therefore out of scope of this research paper to discuss these matters in any detail. For simplicity, I assume that cost is proportional to memory size and processor resources, but at the same time note that this is a gross oversimplification.

c) Research Issues

Space, it does show that the solution space is large enough to accommodate even small embedded devices.

i. *Operating Systems for Wireless Sensor Networks*

The resource limitations and application characteristics of wireless sensor networks place specific requirements on the operating systems running on the sensor nodes. The applications are typically event-based: the application performs most of its work in response to external events. Resources are typically severely limited: memory is on the order of a few kilobytes, processing speed on the order of a few MHz, and limited energy from a battery or some other non-renewable energy source. Early research into operating systems for sensor networks [4] identified the requirements and proposed a system, called TinyOS, that solved many of the problems.

The TinyOS designers did, however, decide to leave out a set of features commonly found in larger operating systems, such as multithreading and run-time module loading. In this research paper, I argue that multithreading and run-time loading of modules are desirable features of an operating system for sensor network nodes. I have implemented an operating system that includes said features and runs within the resource limitations of a sensor node, and thereby show that these features are feasible for sensor node operating systems.

ii. *Connecting Sensor Networks and IP Networks*

A number of practical problems manifest themselves when doing a real-world deployment of a wireless sensor network. One of these is how to get data into and out of the sensor network, which may be deployed in a remote location. One way to solve this problem is to connect the sensor networks to an existing network infrastructure as an access network to the sensor network.

Today most network infrastructures, including the global Internet, use the Internet Protocol (IP) [17] as its base technology. It is therefore interesting to

investigate how wireless sensor networks can be connected to IP network infrastructures. From the engineering perspective, the problem of connecting a sensor network with an IP network can easily be solved. In many cases, it is possible to simply place a PC inside or on the border of the sensor network and connect the PC both to the IP network and to the sensor network.

The PC then acts as the gateway between the sensor network and the IP network. There are also many other possibilities, such as using a special-purpose device that connects the two Networks [17] or using satellite access to a special base station connected to the sensor network [9]. From the research perspective, however, the problem still has opportunities for investigation.

Paper C is a first step towards characterizing the solution space. It presents three different types of solutions to the problem: proxy architectures overlay networking and direct connection by using TCP/IP in the sensor network. This research paper focuses on the last solution: connecting sensor networks and IP networks by using the TCP/IP protocols inside the sensor network as in the outside IP network.

iii. *TCP/IP for Wireless Sensor Networks*

From the research perspective, investigating the use of TCP/IP in wireless sensor networks is of importance because the intersection of the TCP/IP protocol suite, the dominating communication protocol suite today, and wireless sensor networks, a new area in computer networking research, has not been previously studied. In general, the purpose of research is to provide understanding of problems and to gain new knowledge.

Within this particular problem, we can develop new understanding of the interaction between wireless sensor networks and wired network infrastructures by identifying, solving, and studying the problems with TCP/IP in sensor networks. From the engineering perspective, however, using the TCP/IP protocol suite inside the wireless sensor network may not be the best approach to solving the problem of connecting wireless sensor networks to IP networks, for some arbitrary definition of best.

There may be many other solutions to the problem that perform better both in a quantitative sense, e.g. that provide higher throughput or better energy efficiency, and in a qualitative sense, e.g. that provide a better security architecture. Prior to this research paper, however, no research has to the best of my knowledge been carried out to support claims in either way. There are a number of problems with TCP/IP for wireless sensor networks. An enumeration of the problems, which are identified in paper D, follows.

d) IP Addressing Architecture

In ordinary IP networks, each network interface attached to a network is given its own unique IP address. The addresses are assigned either statically by human configuration, or dynamically using mechanisms such as DHCP [8]. This does not fit well with the sensor network paradigm. For sensor networks, the addresses of the individual sensors are not interesting as such. Rather, the data generated by the sensors is the main interest. It is therefore advantageous to be able to loosen the requirement that each sensor has a unique address.

e) Address Centric Routing

Packet routing in IP networks is *address centric*, i.e., based on the addresses of the hosts and networks. The application specific nature of sensor networks makes *data centric* routing mechanisms [14] preferable. Data centric routing uses node attributes and the data contained in the packets to route packets towards a destination. Additionally, data centric mechanisms are naturally adapted to in-network data fusion [12].

f) Header Overhead

The protocols in the TCP/IP suite have a high overhead in terms of protocol header size, particularly for small packets. For small data packets, the header overhead is over 95%. Since energy conservation is of prime importance in sensor networks, transmission of unnecessary or redundant packet header fields should be avoided.

g) TCP Performance and Energy Efficiency

The reliable byte-stream protocol TCP has serious performance problems in wireless networks, both in terms of throughput [7] and in terms of energy efficiency. To be able to use TCP as a reliable transport protocol in wireless sensor networks, methods must be developed to increase the performance of TCP in the specific setting of sensor networks. The end-to-end acknowledgment and retransmission scheme employed by TCP is not energy efficient enough to be useful in wireless sensor networks. A single dropped packet requires an expensive retransmission from the original source. Because sensor networks often are designed to be multi-hop, a single retransmission will incur transmission and reception costs at every hop through which the retransmitted packet will travel.

h) Limited Nodes

Sensor nodes are typically limited in terms of memory size and processing power. Any algorithm developed for sensor networks must therefore take these limitations into consideration.

II. Contributions and Results

The main scientific contributions of this research paper are:

- The design and implementation of the uIP and the lwIP TCP/IP stacks that demonstrate that TCP/IP can be implemented on systems with very limited

memory resources, without sacrificing interoperability or compliance.

- The formulation of initial solutions to the problems with TCP/IP for sensor networks, which point towards the feasibility of using TCP/IP for wireless sensor networks. This opens up opportunities for new research.
- The design and implementation of the Contiki operating system that has a number of features currently not found in other operating systems for the same class of hardware platforms. These features enable rapid experimentation for further research into the area of this research paper.

The work presented in this research paper has had a visible impact on networking for embedded systems and, to a lesser degree, on sensor networks. Less than a year after paper D was published, the 6lowpan IETF workgroup [16] was established. The focus of the workgroup is on standardizing transmission of IP packets over IEEE 802.15.4 [4], a sensor networking radio technology.

The workgroup charter explicitly cites paper D and the uIP stack presented in paper A. The work in this research paper is mentioned in books on embedded systems and networking [5, 6] and cited in numerous academic papers (e.g. [3, 11, 14, 9, 13, 4, 12, 15, 16, 13, 5, 6, 7]). Articles in professional magazines have been written by using the uIP software for wireless sensor networks [8].

The software has been used in academic projects [5, 2], in courses e.g. at University of California, Los Angeles (UCLA) [74] and Stanford University [19], as well as in laboratory exercises [18, 79]. Finally, the software is being used in embedded operating systems [1, 11], and in a large number of embedded products (e.g. [12, 20, 2, 12, 13, 16, 17, 18, 15]).

III. Related Work

This chapter presents related work. The discussion is divided into four sections: small TCP/IP implementations, operating systems for sensor networks, connecting IP networks with sensor networks, and TCP/IP for sensor networks.

a) Small TCP/IP Implementations

There are several small TCP/IP implementations that fit the limitations of small embedded systems. Many of those implementations does, however, refrain from implementing certain protocol mechanisms in order to reduce the complexity of the implementation.

The resulting implementation may therefore not be fully compatible with other TCP/IP implementations. Hence, communication may not be possible many small TCP/IP implementations are tailored for a specific application, such as running a web server. This makes it possible to significantly reduce the implementation complexity, but does not provide a general communications mechanism that can be used for other applications. The PICmicro stack [10] is an example of such a TCP/IP implementation.

Unlike such implementations, the uIP and lwIP implementations are not designed for a specific application. Other implementations rely on the assumption that the small embedded device always will be communicating with a full-scale TCP/IP implementation running on a PC or work-station class device. Under this assumption it is possible to remove certain mechanisms that are required for full compatibility. Specifically, support for IP fragment reassembly and for TCP segment size variation are two mechanisms that often are left out.

Examples of such implementations are Texas Instrument's MSP430 TCP/IP stack [12] and the TinyTCP code [19]. Neither the uIP or the lwIP stack are designed under this assumption.

In addition to the TCP/IP implementation for small embedded systems, there is a large class of TCP/IP implementations for embedded systems with less constraining limitations. Typically, such implementations are based on the TCP/IP implementation from the BSD operating system [6].

These implementations do not suffer from the same problems as the tailored implementations.

Such implementations does, however, in general require too large amount of resources to be feasible for small embedded systems. Typically, such implementations are orders of magnitude larger than the uIP implementation.

b) Operating Systems for Sensor Networks

TinyOS [14] is probably the earliest operating system that directly targets the specific applications and limitations of sensor devices. TinyOS is built around a lightweight event scheduler where all program execution is performed in tasks that run to completion. TinyOS uses a special description language for composing a system of smaller components [13] which are statically linked with the kernel to a complete image of the system. After linking, modifying the system is not possible [15].

The Contiki system is also designed around a lightweight event-scheduler, but is designed to allow loading, unloading, and replacing modules at run-time. In order to provide run-time reprogramming for TinyOS, Levis and Culler have developed Mat´e [15], a virtual machine for TinyOS devices. Code for the virtual machine can be downloaded into the system at run-time.

The virtual machine is specifically designed for the needs of typical sensor network applications. Similarly, the Magnet OS [9] system uses a virtual Java machine to distribute applications across the sensor network. The advantages of using a virtual machine

instead of native machine code is that the virtual machine code can be made smaller, thus reducing the energy consumption of transporting the code over the network. One of the drawbacks is the increased energy spent in interpreting the code. For long running programs the energy saved during the transport of the binary code is instead spent in the overhead of executing the code.

Contiki does not suffer from the executional overhead as modules loaded into Contiki are compiled to native machine code.

c) Connecting IP Networks with Sensor Networks

SensorWare [13] provides an abstract scripting language for programming sensors, but their target platforms are not as resource constrained as ours. Similarly, the EmStar environment [18] is designed for less resource constrained systems. Reijers and Langendoen [19] use a patch language to modify parts of the binary image of a running system.

This works well for networks where all nodes run the exact same binary code but soon gets complicated if sensors run slightly different programs or different versions of the same software. The Mantis system [2] uses a traditional preemptive multi-threaded model of operation. Mantis enables reprogramming of both the entire operating system and parts of the program memory by downloading a program image onto EEPROM, from where it can be burned into fiash ROM.

Due to the multithreaded semantics, every Mantis program must have stack space allocated from the system heap, and locking mechanisms must be used to achieve mutual exclusion of shared variables. In Contiki, only such programs that explicitly require multi-threading need to allocate an extra stack.

At the time of publication of paper C, there was very little work done in the area of connecting wireless sensor networks and IP networks. Recently, however, a number of papers on the subject has been published. Ho and fall [4] have presented an application of Delay Tolerant Networking (DTN) mechanisms to sensor networks.

Their work is similar to that presented in paper C, but is more focused on the specifics of the DTN architecture. The overlay architecture presented by Dai and Han [12] unifies the Internet and sensor networks by providing a sensor network overlay layer on top of the Internet. While this work is similar in scope to the work in this research paper, it explores a slightly different path: this research paper explores the interconnectivity in a lower layer of the protocol stack.

The FLexible Interconnection Protocol (FLIP) [18] provides interconnectivity between IP networks and sensor networks, but relies on protocol converters at the border of the sensor network. This research paper investigates an architecture where no explicit protocol converters are required. Finally, the Plutarch architecture [2] changes the communication architecture of the Internet in a way that is able to accommodate natural inclusion of sensor networks in the new communication architecture.

This work is orthogonal to the work in this research paper. The intention with this research paper is to investigate how sensor networks can be connected with today's IP network infrastructures.

d) TCP/IP for Wireless Sensor Networks

While I am not aware of any previous work on TCP/IP for wireless sensor networks, the area of mobile ad-hoc networks (MANETs) is the area which is most closely related to the area of TCP/IP for wireless sensor networks. MANETs typically use the TCP/IP protocol suite for communication both within the MANETs and with outside networks.

There are, however, a number of differences between sensor networks and MANETs that affect the applicability of TCP/IP. First, MANET nodes typically has significantly more resources in terms of memory and processing power than sensor network nodes. Furthermore, MANET nodes are operated by human users, whereas sensor networks are intended to be autonomous.

The user-centricity of MANETs makes throughput the primary performance metric, while the per-node throughput in sensor networks is inherently low because of the limited capabilities of the nodes. Instead, energy consumption is the primary concern in sensor networks. Finally, TCP throughput is reduced by mobility [16], but nodes in sensor networks are usually not as mobile as MANET nodes.

While the specific area of TCP/IP for wireless sensor networks has not been previously explored, there are a number of adjacent areas that are relevant to this licentiate research paper. The following sections presents the related work in those areas.

i. *Reliable Sensor Network Transport Protocols*

Reliable data transmission in sensor networks has attained very little research attention, mostly because many sensor network applications do not require reliable data transmission. Nevertheless, a few protocols for reliable data transport have been developed.

Those protocols target both the problem of reliable transmission of sensor data from sensors to a sink node, and the problem of reliable transmission of data from a central sink node to a sensor. Potential uses of reliable data transmission is transport of important sensor data from one or more sensors to a sink node, transmission of sensor node configuration from a central server to one or more sensors, program downloads to sensor nodes, and other administrative tasks.

Most protocols for reliable transport in sensor networks are designed specifically for sensor networks and therefore cannot be readily used for e.g.

downloading data from an external IP network, without protocol converters or proxy servers. Reliable Multi-Segment Transport (RMST) [18] provides a reliable transport protocol for bounded messages on top of the Distributed Diffusion routing paradigm [4]. RMST uses either hop-by-hop reliability through negative acknowledgments and local retransmissions, or end-to-end reliability by using positive acknowledgments and end-to-end retransmissions. The authors provide simulation results and conclude that reliable transport for sensor networks is best implemented on the MAC layer. The results provided rely on the fact that the Directed Diffusion routing substrate is able to find relatively good paths through the network, however.

Pump Slowly Fetch Quickly (PSFQ) [3] is a reliable transport protocol that focuses on one-to-many communication situations and uses hop-by-hop reliability. In PSFQ, data is slowly pumped towards the receivers, one fragment at a time. If a nodes along the path towards the receiver notices that a data fragment has been lost, it issues a *fetch* request to the closest node on the backward path.

The number of fetch requests for a single fragment is bounded and fetch requests are issued only within the time frame between two data fragments are pumped. Event-to-Sink Reliable Transport (ESRT) [13] is a transport protocol that provides a semi-reliable transport in only one direction. Data that is sent from sensors to a sink is given a certain amount of reliability. The sink node, which is assumed to have more computational resources than the sensors, computes a suitable reporting frequency for the nodes.

ii. *Header Compression*

Header compression is a technique that reduces packet header overhead by refraining from transmitting header fields that do not change between consecutive packets. The header compressor and the decompressor share the state of streams that pass over them. This shared state is called the header compression *context*.

The compression works by not transmitting full headers, but only the delta values for such header fields that change in a predictable way. Early variants of header compression for TCP were developed for low speed serial links [15] and are able to compress most headers down to only 10% of their original size.

Early header compression schemes did not work well over lossy links since they could not recover from the loss of a header update. A missed header update will cause subsequent header updates to be incorrect because of the context mismatch between the compressor and the decompressor. The early methods did not try to detect incorrectly decompressed headers. Rather, these methods trusted recipients to drop packets with erroneous headers and relied on retransmissions from the sender to repair the context mismatch.

Degermark et al. [16, 17] have presented a method for compressing headers for both TCP/IP and for a set of real-time data protocols. The method is robust in the sense that it is able to recover from a context mismatch by using feedback from the header decompressor. The feedback information is piggybacked on control packets such as acknowledgments that travel on the reversepath. Furthermore, authors introduces the TWICE algorithm.

The algorithm is able to adapt to a single lost header delta value by applying the received delta value twice. Incorrectly decompressed headers are identified by computing the checksum of the decompressed packet. If the checksum is found to be incorrect by the decompressor, a full header is requested from the compressor, thus synchronizing the header compression context. Sridharan et al. [17] have presented Routing-Assisted Header Compression (RAHC), a header compression scheme that is particularly well-suited for multi-hop networks.

Unlike other header compression schemes, the RAHC algorithm works end-to-end across a number of routing hops. The algorithm utilizes information from the underlying routing protocol in order to detect route changes and multiple paths.

iii. *TCP over Wireless Media*

TCP [18] was designed for wired networks where congestion is the predominant source of packet drops. TCP reduces its sending rate detecting packet loss in order to avoid overloading the network. This behavior has shown to be problematic when running TCP over wireless links that have potentially high bit error rates. Packet loss due to bit errors will be interpreted by TCP as a sign of congestion and TCP will reduce its sending rate. TCP connections running over wireless links may therefore see very large reductions in throughput.

A number of mechanisms for solving these problems have been studied. Wireless TCP enhancements can be divided into three types [6]: *splitconnection*, *end-to-end*, and *link-layer*. The split-connection approach, as exempli fied by Indirect TCP [5] and M-TCP [16], splits each TCP connections into two parts: one over the wired network and one over the wireless link. Connections are terminated at a base station to allow a specially tuned protocol to be used between the base station and the wireless host. TCP snoop [7] is a link-layer approach that is designed to work in a scenario where the last hop is over a wireless medium.

TCP snoop uses a program called the *snoop agent* that is running on the base station before the last hop. The snoop agent intercepts TCP segments and caches them. If it detects a failed transmission, it will

immediately retransmit the lost segment .When duplicate acknowledgements to be sent towards the original sender of the segment. A-TCP [5] is primarily designed for wireless ad-hoc networks and is an example of the end-to-end approach. A-TCP inserts a conceptual layer in between IP and TCP that deals with packet losses because of transmission errors and unstable routes. Unlike the other approaches, A-TCP requires modifications to the end-host.

iv. *Addressing in Sensor Networks*

Addressing in sensor networks is different from addressing in other computer networks in that the sensors do not necessarily need to have individual addresses [12]. Instead, many sensor network applications benefit from seeing the *data* sensed by the network the primary addressing object [15]. This allows routing to be *data-centric* rather than the traditional address-centric.

One of the earliest data-centric routing protocols is Directed Diffusion [14] which propagates an information interest through the network. When a sensor obtains information for which an interest has been registered, it transmits the information back towards the source of the information interest. A different approach is taken by TinyDB [18] where the sensor network is viewed as a distributed data base.

The data base is queried with an SQL-like language. Query strings are processed by a base station, and compressed and optimized queries are disseminated through the sensor network. Results are distributed back through the routing tree that was formed when the query was propagated. This is an addressing scheme where the data is explicitly addressed and where individual nodes are not possible to address directly.

IV. Conclusions and Future Work

This licentiate research paper takes the first steps towards the use of the TCP/IP protocol suite in wireless sensor networks. It builds the framework in which the use of TCP/IP can be further investigated, identifies the problems with TCP/IP for sensor networks, and formulates initial solutions to the problems. The contribution of this work is that it for the first time brings TCP/IP, the dominant protocol stack, together with wireless sensor networks.

The results of the work presented in this research paper have had a significant impact on the embedded TCP/IP networking community. The software developed as part of the research paper has become widely known in the community.

The software is used in academic research projects, academic courses, as well as a large number of embedded devices. I will continue this work with experimental studies of the use of TCP/IP in wireless sensor networks.

Further investigation must be made before the hypothesis that TCP/IP is a viable protocol suite for wireless sensor networks can be validated or invalidated.

We have already made simulation studies of the Distributed TCP Caching mechanism [13] and are designing a MAC layer that will support DTC. We intend to evaluate the energy efficiency of TCP/IP for sensor networks by using the method described by Ritter et al. [17].

While this method has been developed to experimentally evaluate a model of life-time bounds [4], it also is useful for comparing the energy efficiency of communication protocols. I will also continue to investigate software construction for memory stressed systems, based on the bindings in papers A and B.

This work consists of developing mechanisms and methods for implementing computer programs for resource limited embedded systems and sensor nodes. I am currently working on a lightweight mechanism called proto threads that provides sequential flow of control for event-driven systems.

References Références Referencias

1. eCos Embedded Configurable Operating System. Web page. URL: ttp://sources.redhat.com/ecos/
2. H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker J. Deng, and R. Han. Mantis: system support for multimodal networks of in-situ sensors. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 50.59, 2003.
3. P. Agrawal, T.S. Teck, and A.L. Ananda. A lightweight protocol for wireless sensor networks. March 2003.
4. J. Alonso, A. Dunkels, and T. Voigt. Bounds on the energy consumption of routings in wireless sensor networks. In *Proceedings of the 2ndWiOpt, Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Cambridge, UK, March 2004.
5. A. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, May 1995.
6. H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Trans. Netw.*, 5(6):756.769, 1997.
7. H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP performance over wireless networks. In *Proceedings of the first ACM Conference on Mobile Communications and Networking*, Berkeley, California, November 1995.
8. D. Barnett and A. J. Massa. Inside the uIP Stack. *Dr. Dobb's Journal*, February 2005.

9. R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. W. D. Kim, B. Zhou, and E. Sirer. On the need for system-level support for ad hoc and sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(2):1.5, 2002.
10. J. Bentham. *TCP/IP Lean: Web servers for embedded systems*. CMP Books, October 2000.
11. S. Beyer, K. Mayes, and B. Warboys. Application-compliant networking on embedded systems. In *Proceedings of the 5th IEEE Internation Workshop on Networked Appliances*, pages 53.58, October 2002.
12. C. Borrelli. TCP/IP on Virtex-II Pro Devices Using lwIP. XAPP 663, Xilinx Inc., August 2003.
13. A. Boulis, C. Han, and M. B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS `03)*, May 2003.
14. M. Britton, V. Shum, L. Sacks, and H. Haddadi. A biologically-inspired approach to designing wireless sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks*, Istanbul, Turkey, 2005.
15. F. P. Brooks Jr. The computer scientist as a toolsmith II. *Communications of the ACM*, 39(3):61.68, March 1996.
16. K. Brown and S. Singh. M-TCP: TCP for mobile cellular networks. *ACM Computer Communications Review*, 27(5):19.43, October 1997.
17. P. Buonadonna, D. Gay, J. Hellerstein W. Hong, and S. Madden. TASK: Sensor Network in a Box. In *Proceedings of the Second European Workshop on Sensor Networks*, 2005.
18. B. F. Cockburn. CMPE 401 - Computer Interfacing. Web page. URL: http://www.ece.ualberta.ca/. cmpe401/
19. G. H. Cooper. TinyTCP. Web page. 2002-10-14. URL: http://www.csonline.net/bpaddock/tinytcp/
20. Nu Horizons Electronics Corp. TCP/IP Development Kit. Web page. URL: http://www.nuhorizons.com/

This page is intentionally left blank