Making TCP/IP Viable for Wireless Sensor Networks

Adam Dunkels, Juan Alonso, Thiemo Voigt
Swedish Institute of Computer Science
{adam,alonso,thiemo}@sics.se

Abstract— **The TCP/IP protocol suite, which has proven itself highly successful in wired networks, is often claimed to be unsuited for wireless micro-sensor networks. In this work, we question this conventional wisdom and present a number of mechanisms that are intended to enable the use of TCP/IP for wireless sensor networks: spatial IP address assignment, shared context header compression, application overlay routing, and distributed TCP caching (DTC). Sensor networks based on TCP/IP have the advantage of being able to directly communicate with an infrastructure consisting either of a wired IP network or of IP-based wireless technology such as GPRS. We have implemented parts of our mechanisms both in a simulator environment and on actual sensor nodes. Our preliminary results are promising.**

I. Introduction

Many wireless sensor networks cannot be operated in isolation; the sensor network must be connected to an external network through which monitoring and controlling entities can reach the sensor network. The ubiquity of TCP/IP has made it the de-facto standard protocol suite for wired networking. By running TCP/IP in the sensor network it is possible to directly connect the sensor network with a wired network infrastructure, without proxies or middle-boxes [5]. It is often argued that the TCP/IP protocol stack is unsuited for sensor networks because of the specific requirements and the extreme communication conditions that sensor networks exhibit. We believe, however, that by using a number of optimization mechanisms, it is possible to achieve similar performance in terms of energy consumption and data throughput with TCP/IP as that obtained by using specialized communication protocols, while at the same time benefiting from the ease of interoperability and generality of TCP/IP.

We envision that data transport in a *TCP/IP sensor network* is done using the two main transport protocols in the TCP/IP stack: the best-effort UDP and the reliable byte-stream TCP. Sensor data and other information that do not require reliable transmission is sent using UDP, whereas TCP is used for administrative tasks that require reliability and compatibility with existing application protocols. Examples of such administrative tasks are configuration and monitoring of individual sensor nodes, and downloads of binary code or data aggregation descriptions to sensor nodes.

The contribution of this paper are our innovative solutions to the following problems with TCP/IP for sensor networks:

IP addressing architecture. In ordinary IP networks, IP addresses are assigned to each network interface that is connected to the network. Address assignment is done either using manual configuration or a dynamic mechanism such as DHCP. In a large scale sensor network, manual configuration is not feasible and dynamic methods are usually expensive in terms of communication. Instead, we propose a *spatial IP address assignment* scheme that provides semi-unique IP addresses to sensor nodes.

Header overhead. The protocols in the TCP/IP suite have a very large header overhead, particularly compared to specialized sensor network communication protocols. We believe that the shared context nature of sensor networks makes *header compression* work well as a way to reduce the TCP/IP header overhead.

Address centric routing. Routing in IP networks is based on the addresses of the hosts and networks. The application specific nature of sensor networks makes the use of data-centric routing mechanisms [6] preferable over address-centric mechanisms, however. We propose a specific form of an *application overlay network* to implement data-centric routing and data aggregation for TCP/IP sensor networks.

Limited nodes. Sensor nodes are typically limited in terms of memory and processing power. It is often assumed that the TCP/IP stack is too heavy-weight to be feasible for such small systems. In previous work [4], we have shown that this is not the case but that an implementation of the TCP/IP stack in fact can be run on 8-bit micro-controllers with only a few hundred bytes of RAM.

TCP performance and energy inefficiency. The reliable byte-stream protocol TCP has been shown to have serious performance problems in wireless networks [2]. Moreover, the end-to-end acknowledgment and retransmission scheme employed by TCP causes expensive retransmissions along every hop of the path between the sender and the receiver, if a packet is dropped. We have developed a distributed mechanism similar to TCP snoop [2] that we believe can be used to overcome both problems.

While we are not aware of any research on TCP/IP for wireless sensor networks, there is a plethora of work being done on TCP/IP for mobile ad-hoc networks (MANETs). There are, however, a number of differences between sensor networks and MANETs that affect the applicability of TCP/IP. 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 [7], but nodes in sensor networks are usually not as mobile as MANET nodes.

In Sections II through VI we describe our proposed solutions to the above problems and report on preliminary results. Finally, Section VII concludes the paper and presents the direction of our future work.

II. Spatial IP Address Assignment

For most sensor networks, the data generated by the sensor nodes needs to be associated with the spatial location where the data was sensed. It is therefore a reasonable assumption that the nodes in a sensor network have some way of determining their location, and methods for localization in sensor networks have been developed [14].

For TCP/IP sensor networks, we propose a *spatial IP address assignment* mechanism to solve the problem of address assignment. With spatial IP address assignment, each sensor node uses its spatial location to construct an IP address. Since we assume that the nodes are aware of their own spatial location, the address assignment requires neither a central server nor communication between the sensor nodes.

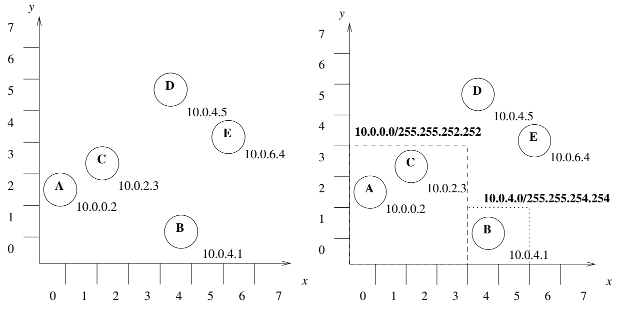

Fig. 1. Example spatial IP address assignment and two regional subnets.

Figure 1 shows an example network with spatially assigned IP addresses. In this particular network, each sensor has constructed its IP address by taking the (x, y) coordinates of the node as the two least significant octets in the IP address. We do not intend to specify the specific way that the addresses are constructed, but assume that it will vary between different kind of sensor networks.

Because location information is encoded in the IP addresses, we can define a *regional subnet* as a set of sensor nodes that share a prefix (Figure 1) and implement a straightforward *regional broadcast* mechanism, analogous to ordinary IP subnet broadcasts. This mechanism does not require a special mapping between logical and physical location as needed, e.g., in GeoCast [10].

The spatially assigned IP addresses are not guaranteed to be unique, since two or more adjacent sensor nodes may obtain the same location coordinates and thereby construct the same address. Nodes with duplicate addresses are in the proximity of each other, however, which helps to avoid routing problems; nodes with duplicate addresses are likely to share large parts of routing paths towards the nodes. Transport layer port number conflicts for sensors that are able to overhear each other's radio communication can be resolved by passive monitoring of the neighbors' communication.

III. Header Compression

Energy is often the most scarce resource in wireless sensor networks, and for many applications radio transmission is the most expensive activity [12]. The minimum size of a UDP/IP header is 28 bytes and a 4 bytes sensor data value sent using using UDP/IP has a 87.5% header overhead, which cause large amounts of energy to be spent in transmitting the header.

In sensor networks, all sensor nodes are assumed to cooperate towards a common goal, and therefore the nodes share a common context. For that reason, all nodes can agree on specific UDP/IP header field values for sensor data UDP datagrams. The headers can then be compressed using simple pattern-matching techniques. For example, since all nodes are part of the same IP subnet, there is no need to transmit full IP addresses in the headers of packets that are sourced from or are destined to nodes in the sensor network. Similarly, by utilizing only a small range of UDP ports for the sensor data datagrams, transmitting full 16-bit port numbers is not required for packets containing sensor data.

For TCP connections, standard header compression techniques [3], [9] can be used, but the specific requirements of the sensor network place additional challenges. For instance, while ordinary TCP header compression may be content with the connection end-points detecting and retransmitting incorrectly decompressed headers, a multi-hop wireless sensor network must perform in-network detection and retransmission in a more aggressive manner because of the energy consumption caused by end-to-end retransmissions. It should also be noted that others are working on multi-hop aware header compression techniques [11] that could be beneficial for TCP/IP sensor networks as well.

IV. Application Overlay Routing

The spatial IP addressing mechanism provides a way to send IP packets to nodes specified by their spatial location, but a pure IP packet routing scheme cannot readily support data aggregation or attribute based routing. Instead, we believe that application overlay networks may be a good way to implement such mechanisms. At first sight, an overlay network might seem too expensive for a wireless sensor network, because of the mapping required between the physical network and the overlay network. We argue, however, that by choosing an overlay network that fits well with the underlying physical nature of a sensor network, the mapping is not necessarily expensive.

We believe that UDP datagrams sent using link local IP broadcast [13] is a suitable mechanism for implementing an application overlay network on top of the physical sensor network structure. Link local broadcasts provide a direct mapping between the application overlay and the underlying wireless network topology. By tuning the header compression for the special case of link-local broadcasts, the header overhead of such packets does not need to be significantly larger than that of a broadcast packet directly sent using the physical network interface. Furthermore, link-local application layer broadcasts can also be used to implement both low-level mechanisms such as neighbor discovery and high-level protocols such as Directed Diffusion [8].

In addition to the compatibility aspects, an application layer overlay network also has the benefits of generality in that it can be run transparently over both sensor nodes and regular Internet hosts, without requiring proxies or protocol converters.

V. Tiny TCP/IP Implementation

It is often assumed that TCP/IP is too heavy-weight to be feasible to implement on a small system such as a sensor node. We have previously shown [4] that even a small system can run the full TCP/IP protocol stack, albeit with lower performance in terms of throughput. Our uIP TCP/IP implementation [4] occupies only a few kilobytes of code space and requires as little as a few hundreds bytes of memory, and we have successfully ported it to the Embedded Sensor Board (ESB) developed at FU Berlin [1]. The ESB is equipped with a number of sensors, an RF transceiver, and an MSP430 low-power 8-bit micro-controller with 2048 bytes of RAM and 60 kilobytes flash ROM.

VI. Distributed TCP Caching

The reliable byte-stream TCP was designed for wired networks where bit-errors are uncommon and where congestion is the predominant source of packet drops. Therefore, TCP always interprets packet drops as a sign of congestion and reduces its sending rate in response to a dropped packet. Packet drops in wireless networks are often due to bit-errors, which leads TCP to misinterpret the packet loss as congestion. TCP will then lower the sending rate, even though the network is not congested.

Furthermore, TCP uses end-to-end retransmissions, which in a multi-hop sensor network requires a retransmitted packet to be forwarded by every sensor node on the path from the sender to the receiver. As Wan et al. note, end-to-end recovery is not a good candidate for reliable transport protocols in sensor networks where error rates are in the range of 5% to 10% or even higher [15]. A scheme with local retransmissions is more appropriate since it is able to move the point of retransmission closer towards the final recipient of the packet.

To deal with these issues, we propose a scheme called *distributed TCP caching* (DTC) that uses segment caching and local retransmissions in cooperation with the link layer. Other mechanisms for improving TCP performance over wireless links, such as TCP snoop [2], focus on improving TCP *throughput*. In contrast, DTC is primarily intended to reduce the *energy consumption* required by TCP. DTC does not require any protocol changes neither at the sender nor at the receiver.

We assume that each sensor node is able to cache only a small number of TCP segments; specifically, we assume that nodes only have enough memory to cache a single segment.

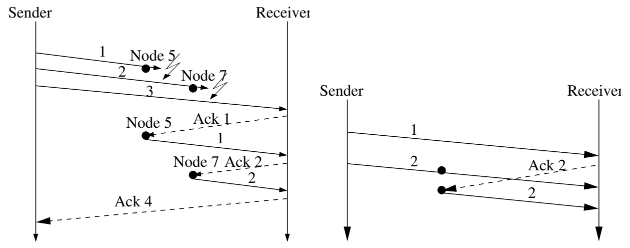

Fig. 2. Distributed TCP caching (left) and spurious retransmission (right)

The left part of Figure 2 shows a simplified example how we intend DTC to work. In this example, a TCP sender transmits three TCP segments. Segment `1` is cached by node 5 right before it is dropped in the network, and segment `2` is cached by node 7 before being dropped. When receiving segment `3`, the TCP receiver sends an acknowledgment (`ACK 1`). When receiving `ACK 1`, node 5, which has a cached copy of segment `1`, performs a local retransmission. Node 5 also refrains from forwarding the acknowledgment towards the TCP sender, so that the acknowledgment segment does not have to travel all the way through the network. When receiving the retransmitted segment `1`, the TCP receiver acknowledges this segment by transmitting `ACK 2`. On reception of `ACK 2`, Node 7 performs a local retransmission of segment `2`, which was previously cached. This way, the TCP receiver obtains the two dropped segments by local retransmissions from sensor nodes in the network, without requiring retransmissions from the TCP sender. When the acknowledgment `ACK 4` is forwarded towards the TCP sender, sensor nodes on the way can clear their caches and are thus ready to cache new TCP segments.

A. Segment Caching and Packet Loss Detection

DTC uses segment caching to achieve local retransmissions. Because of the memory limitations of the sensor nodes, it is vital to the performance of the mechanism to find an appropriate way for nodes to select which segments to cache. Initial analysis suggest that a desirable outcome of the selection algorithm is that segments are cached at nodes as close to the receiver as possible, and that nodes closer to the receiver cache segments with lower sequence numbers. To achieve this, each node caches the TCP segment with the highest sequence number seen, and takes extra care to cache segments that are likely to be dropped further along the path towards the receiver. We use feedback from a link layer that supports positive acknowledgments to infer packet drops on the next-hop. A TCP segment that is forwarded but for which no link layer acknowledgment is received may have been lost in transit, and the segment is *locked* in the cache indicating that it should not be overwritten by a TCP segment with a higher sequence number. A locked segment is cleared from the cache only when an acknowledgment that acknowledges the cached segment is received, or when the segment times out.

To avoid retransmissions from the original TCP sender, DTC needs to respond faster to packet drops than regular TCP. DTC uses ordinary TCP mechanisms to detect packet loss: time-outs and duplicate acknowledgments. Every node participating in DTC maintains a soft TCP state for connections that pass through the node. We assume symmetric and relatively stable routes, and therefore the nodes can estimate the delays between the node and the connection end-points. The delays experienced by the nodes are lower than those estimated by the TCP end-points, and the nodes are therefore able to use lower time-out values and perform retransmissions quicker than the connection end-points.

In TCP, duplicate acknowledgments signal either packet loss or packet reordering. A TCP receiver uses a threshold of three duplicate acknowledgments as a signal of packet loss, which may be too conservative for DTC. Since each DTC node inspects the TCP sequence numbers of forwarded TCP segments, the nodes may be able to compute a heuristic for the amount of packet reordering, and to lower the duplicate acknowledgment threshold if packet reordering is found to be uncommon in the network. Furthermore, care must be taken to avoid spurious retransmissions caused by misinterpreting acknowledgments for new data as acknowledgments that signal packet loss, as shown in the right part of Figure 2. The nodes can use estimated round-trip times to distinguish between an acknowledgment that detects a lost packet and one that acknowledges new data.

We are also considering using the TCP SACK option to detect packet loss and also as a signaling mechanism between DTC nodes.

B. Preliminary Results

We have performed simulations comparing standard TCP with DTC. Our results show vast improvements: For path lengths between 5 and 10 hops and packet loss rates between 5% and 15%, the number of retransmissions that the TCP sender has to perform decreases by a factor of four to eight. For example, with a packet loss rate of 10% for data packets (5% for acknowledgments and 2% for link level acknowledgments), a path length of 10 hops, and with 500 packets to be transmitted the number of required source retransmission decreases from 51 to 6 (averaged over 30 different runs).

In sensor networks, sensor data flows from sources to sinks, whereas control or management data flows from sinks to sources [15]. Therefore, nodes close to the sink usually are the first to run out of energy because sensor data sent towards the sink has to pass them. As shown by our initial simulation results in Figure 3, DTC is able to reduce the load at the nodes close to the sink/TCP sender.

We do not yet have any results from the TCP header compression coupled with DTC, but our UDP/IP header compressor is able to reduce UDP/IP headers for sensor data from 28 to three bytes.

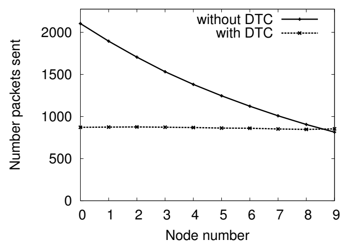

Fig. 3. DTC load reduction close to sender

VII. Conclusions and Future Work

In this paper we challenge the assumption that TCP/IP is unsuitable for sensor networks. Our main contributions are a spatial IP address assignment scheme and a mechanism for distributed segment caching called distributed TCP caching.

Future work will be targeted at further development and evaluation of the proposed mechanisms using both simulation and experiments with physical sensor networks. We are currently looking into the interactions between the link layer and header compression mechanisms that work together with DTC. For DTC, we will consider the energy consumption tradeoffs involved with link layers with different levels of reliability. We also intend to compare DTC with transport protocols specifically designed for sensor networks such as PSFQ [15]. Furthermore, we are currently implementing the DTC mechanism on actual sensor nodes in order to measure real-world performance and preliminary results show that the sensor nodes are capable of running both a full TCP/IP stack and the DTC mechanism.

References

[1] CST Group at FU Berlin. Scatterweb Embedded Sensor Board. Web page. Visited 2003-10-21. `http://www.scatterweb.com/`

[2] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP performance over wireless networks. In *MOBICOM'95*, 1995.

[3] M. Degermark, B. Nordgren, and S. Pink. IP header compression. RFC 2507, Internet Engineering Task Force, February 1999.

[4] A. Dunkels. Full TCP/IP for 8-bit architectures. In *MOBISYS'03*, San Francisco, California, May 2003.

[5] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, and J. Schiller. Connecting Wireless Sensornets with TCP/IP Networks. In *WWIC2004*, February 2004.

[6] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[7] Gavin Holland and Nitin H. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *MOBICOM '99*, August 1999.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[9] V. Jacobson. Compressing TCP/IP headers for low-speed serial links. RFC 1144, Internet Engineering Task Force, February 1990.

[10] Julio C. Navas and Tomasz Imielinski. GeoCast – geographic addressing and routing. In *MOBICOM'97*, pages 66–76, 1997.

[11] S. Mishra R. Sridharan, R. Sridhar. A robust header compression technique for wireless ad hoc networks. In *MobiHoc 2003*, 2003.

[12] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[13] J. Reynolds and J. Postel. Assigned numbers. RFC 1700, Internet Engineering Task Force, October 1994.

[14] A. Savvides, C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MOBICOM'01*, pages 166–179. ACM Press, 2001.

[15] C.Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol For Wireless Sensor Networks. In *WSNA'02*, 2002.