Adopting Internet Protocols to Sensor Internetworking

Murad Kamalov
Helsinki University of Technology
mkamalov@cc.hut.fi

Abstract

Recent research in the area of the Wireless Sensor Networks (WSN) has shown that protocols most commonly used today in the Internet can be implemented within energy and memory constraints of the WSNs. This paper surveys recent research performed in the area of making IP protocols viable for WSNs. Lightweight implementations of IP and IPv6 protocols, uIP and uIPv6 will be covered, as well as viability of running TCP in WSNs will be discussed. Adoption of the Internet protocols for the WSN results in a need for adoption of security mechanisms used in the Internet for WSNs. Thus, we will also cover some symmetric and asymmetric cryptography based, security solutions for WSNs, existing today.

KEYWORDS: Wireless Sensor Networks, Internet Protocols, Security, Symmetric and Assymetric cryptography

1 Introduction

Wireless Sensor Networks (WSN) and the Internet of Things, Internet where every important device (sensors, domestic appliances, etc.) is connected to the Interenet, have became increasingly hot topics in the last couple of years. Many new protocols and security solutions were proposed for WSNs and until recently it was widely assumed that protocols used in todays Internet could not be used in WSNs, because of their high memory, CPU and communication requirements. Nevertheless recent research and actual implementations have shown that it is possible to implement a TCP/IP protocol stack on constrained devices used in WSNs [4, 6, 13, 5, 8, 3].

There are three ways to connect the WSN to TCP/IP based networks. First is a proxy based approach, where WSN internally uses its own non-TCP/IP protocol and connection with TCP/IP network is performed through proxy, which relays data between WSN and the TCP/IP network. Second approach is using overlay networks, where the overlay network is build on top of both TCP/IP network and the WSN network. Applications in this case are built on top of the overlay layer. For example, Dunkels et al. proposed using Delay Tolerant Networks (DTN) bundle protocol [5], as an overlay network. Finally, the third approach is implementing TCP/IP protocol stack directly in the WSN. In this case, every sensor node has its own IP address and can directly communicate to any host in the Internet. In this paper we will consider third approach, implementing TCP/IP in WSNs.

Implementing the TCP/IP stack in WSNs has many advantages. Firstly, it makes different WSNs compatible with each other, allowing them to communicate easily. To date the trend in WSNs has been to use different, incompatible solutions for different problems. Secondly, TCP/IP solves problem of connecting WSNs with the Internet and setting up secure end-to-end connections with endpoints in the Internet, which in turn enables the fully functional Internet of Things. Direct connection of the WSN to the Internet is certainly an advantage, because gateways, which translate protocols used internally in WSN to TCP/IP, used in the Internet, increase complexity of the systems and break end-to-end security. And thirdly, TCP/IP allows to reuse many existing and proven solutions used in the todays Internet for the WSNs. This is especially important in the area of security, where time-proven solutions are usually preferred.

Further in this paper we will survey uIP [3], lightweight implementation of TCP/IP stack for WSNs and its IPv6 equivalent, uIPv6 [6]. Those implementations have broken the common assumption that Internet protocols are initially unsuitable for WSNs. Plain usage of TCP/IP stack in WSNs does not imply that all security solutions used today in the Internet can be used in WSNs. They also have to be adopted and optimized, in order to make them usable in the WSNs. Thus, we will also cover some security aspects of WSNs, related to todays Internet. Specifically, symmetric and asymmetric cryptography will be compared. Asymmetric cryptography was analogically to TCP/IP, considered too heavy for WSNs. But thanks to recent research in Elliptic Curve Cryptography (ECC) [14], its usage also becomes viable in WSNs [11], making adaptation of existing Internet security protocols for WSNs possible.

The rest of this paper is organized as follows. In Section 2 of this paper, we will survey uIP and uIPv6. In Section 3, we will concentrate on security, usage of both symmetric and asymmetric approaches in WSNs, will be described. In subsection 3.3 quick analysis of viability of TLS security protocol in WSNs will be presented. In Section 4, we will analyze, how well does security protocols, used currently in the Internet satisfy needs of WSNs. Section 5 concludes this paper.

2 TCP/IP in Wireless Sensor Networks

TCP/IP is considered quite heavyweight protocol, especially for memory and CPU constrained devices. There were many attempts made to implement TCP/IP stack for WSNs, but majority of them ether allowed establishing very limited

number of connections or were initially intended only for one particular application (e.g web server) and were not generic, as TCP/IP stack should be. Adam Dunkels first time described a fully functional implementation of TCP/IP for WSNs in 2003 [3]. There are two TCP/IP implementations described in by Dunkels: uIP (micro IP) and lwIP (lightweight IP). lwIP is initially intended for more powerful devices and implements majority of TCP/IP features. In contrast, uIP is more lightweight implementation, hence it omits some of the less important and most resource consuming features of the TCP/IP. Only the features unimportant for interoperability with full TCP/IP implementation are omitted from uIP.

Recently the IPv6 enabled implementation of micro IP, called uIPv6 [6], was introduced. Both uIP and uIPv6 are open-sourced and are part of the Contiki operating system [2]. Main goal of the Contiki operationg system is to provide IP connectivity for battery powered, memory and CPU constrained devices.

2.1 Common problems of TCP/IP in WSNs

Independent from the particular implementation all of the TCP/IP implementations for WSNs face following four problems:

1. Addressing and routing in TCP/IP is host-centric (not data-centric)
2. Header overhead in TCP/IP
3. TCP performance over wireless links
4. TCP end-to-end retransmissions [5]

Solutions for some of those problems were proposed, by Dunkels et al. [4]. Some of those solutions are quite interesting. Next, we will discuss some of them in greater detail.

Addressing and routing in TCP/IP is host-centric. The solution for host-centric routing vs. data-centric routing, is to employ techniques of spatial IP address assignment and application overlay routing. In spatial IP address assignment, each node calculates its own IP address based on its position. This technique works quite well in more or less static networks, but will fail in case if node is mobile in WSN. Another part of the solution, application overlay routing, is needed in order to enable data aggregation and attribute based routing, specific to data-centric networks. Although overlay networks can often be quite inefficient, Dunkels et al. [5] argue, that overlay network can be implemented efficiently, if its design corresponds to the underlying physical nature of the sensor network.

TCP end-to-end retransmission. As a solution for the problem of TCP end-to-end retransmission, Dunkels et al. [5] propose mechanism for distributed caching of the TCP segments, which assumes symmetric and relatively stable routes. Proposed technique enables TCP to perform retransmissions from nodes located closer to the segment destination, thus reducing energy consumption (caused by the TCP retransmissions) of the nodes closer to the sink node. Technique seems to be feasible, especially taking into account its backwards compatibility with existing TCP implementations. Its only minus is assumption of symmetric and stable routes, which in case of WSNs might not always be easy to fulfill. In addition, transmission of the data over the same routes over and over again, can cause certain nodes in the WSN to drain battery faster.

TCP performance over the wireless links. This, is not a problem of WSNs alone, but problem of any TCP implementation deployed in the wireless environment. The reason is that TCP was originally designed for wired infrastructures, where main cause of packet loss is congestion. Hence, normal TCP implementation interprets packet loss, as a network congestion and as a result reduces sending rate of IP packets. In contrast, in wireless networks, the main cause of packet loss are bit errors, which results in lowering the sending rate of IP packets, in wireless networks, in situations when network is not actually congested.

2.2 uIP

As mentioned above, uIP is the first fully interoperable implementation of TCP/IP for WSNs. When compiled for 8-bit system, uIP code takes approximately 5KB and amount of RAM used by the implementation can be configured to be as low as 200 bytes. The interesting property of uIP implementation is tight coupling of TCP and IP layers, which is done in order to minimize code size and memory usage of the stack. Another property of uIP, that makes it different from the traditional implementations of TCP/IP, is usage of the event-driven (asynchronous) paradigm, in contrast to stop-and-wait semantics used in POSTFIX Socket and Windows WinSock APIs. Event-driven semantics makes uIP considerably more lightweight, even on the low-end systems because application gains control over the data, as soon as TCP/IP stack receives the packet. In addition, event-driven approach does not cause overhead of the task management and context switching, which is an issue in stop-and-wait semantics.

Nowadays uIP provides two APIs to the application programmer. First is event-based API, where application is invoked, by the stack, at the moment when application data has been received. Second is Protosockets API, which is POSTFIX-like API based on lightweight protothreads. Protosockets API allows to implement applications in classical, sequential way without resorting to event-driven programming. Nevertheless event-driven API is considered lower level, than Protosockets API, hence it uses less memory.

The basic principle of the uIP operation is based on a single threaded main loop depicted in Figure 1. Loop checks repeatedly for new packets and timeouts, processes them, calls the application callbacks, outputs packets to the network device, if needed, and returs back to the main loop. Timeouts, which main loop checks every iteration, are used to trigger the timer based events, such as TCP retransmissions, delayed acknowledgements and round-trip time estimations.

uIP uses single table to hold state of the connections and single global buffer, which is large enough, to hold one packet at a time. It means that uIP can process only one

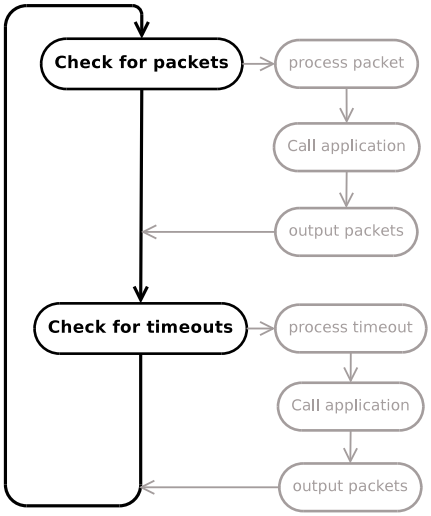

Figure 1: uIP main Loop [3]

packet at a time. The same global buffer is used for all the TCP connections of a sensor node and for both, incoming and outgoing packets. In case of incoming packets, when packet arrives it is placed to the global buffer, after that it is processed and if it contains any application data, the data is passed to the corresponding application. If application needs the data, for later use, it should copy it to the separate buffer. In case of outgoing packet, application should give a pointer to the data being sent, to a stack. Stack generates a header for the data and writes it to the global buffer, then network device reads the data and header from the global buffer and sends it out. Important thing about outgoing packets is that TCP/IP stack does not handle retransmissions internally, but calls application in case if retransmission is needed. This happens, because, in order to save the memory and CPU cycles, uIP TCP/IP stack does not queue every transmitted packet, but assumes that in case if retransmission is needed it will notify the application and application will reproduce the data and repeat the sending.

One of the important TCP features that is not implemented in the uIP is sliding TCP window. Sliding TCP window allows to send multiple TCP segments simultaneously, without waiting for acknowledgments of the previous segments. The reason why authors of uIP decided to omit this functionality from implementation is that it requires many 32-bit operations, which on 8-bit processor will create a lot of computational overhead and will significantly increase the amount of code and RAM usage. In addition to sliding TCP window, TCP congestion control is not implemented in uIP, because there is no need for that without sliding TCP window. Stack already sends only one TCP segment at a time, so sending rate cannot be further decreased. The absence of a sliding TCP window functionality significantly decreases TCP throughput, especially taking into account that majority of TCP receivers implement delayed segment acknowledgment algorithm in order to decrease amount of pure acknowledgments being sent. Typically acknowledgment delay is between 200-500ms. Assuming that average acknowledgment timeout is 200ms, maximum throughput of uIP will be 4166 bytes/s. With delayed acknowledgment algorithm disabled, maximum throughput will be around 25000 bytes/s [3]. Authors of uIP consider throughput provided by uIP appropriate because usually small sensor nodes do not generate so much data, that provided throughput could become a problem. Otherwise sliding TCP window does not affect uIP TCP/IP stacks interoperability with other implementations in any way.

2.3 uIPv6

The availability of the uIPv6, IPv6 enabled TCP/IP stack implementation for WSNs, was announced in second half of the 2008. Code size of uIPv6 is 11.5 kilobytes and it requires around 2 kilobytes of RAM. The main motivator behind creating the IPv6 ready implementation of TCP/IP is support for the Internet of Things, where every important device can be connected to the Internet and can have its own IP address. Of course, it requires large IP address space and auto configuration capabilities, which IPv6 can provide. Architecturally implementation of uIPv6 is very similar to the implementation of uIP. Both, UDP and TCP modules, are tightly coupled with implementation of IP module and the similar (to uIP) event-driven interface is provided to the application developer. Also, global buffer based memory management of uIPv6 is very similar to the memory management of uIP.

3 Security

Security is one of the most critical issues, which should be taken into consideration while implementing TCP/IP stacks for the WSNs and connecting sensor networks directly to the Internet. Although, security challenges in the WSNs have been largely covered by the previous work, there was no evaluation made yet, what kind of security challenges will be experienced in the WSNs, with large deployments of TCP/IP stacks in WSNs.

One of the main challenges of designing security protocols for WSNs, is coping with specifics of WSNs. The main difference of the WSNs from wired networks, is battery powered nodes operating in wireless environment, which means that the main requirement for sensor nodes is to be energy efficient. Hence, in order to ensure energy efficiency, sensor nodes use data aggregation techniques, in order to reduce size of the messages being forwarded. Unfortunately, data aggregation also means that use of end-to-end security protocols is a challenge. With end-to-end security, intermediary nodes will be unable to read the content of the forwarded messages and aggregate them. This is the main reason why link layer security protocols are more popular in WSNs, at the moment, than end-to-end security solutions. In case of link layer security, each node can decrypt the data and use the data aggregation techniques to reduce the size of the forwarded messages.

In addition to data aggregation, DoS attacks have much more serious consequences in WSNs, when compared to ordinary IP networks. DoS attacks are really dangerous for WSNs, because they drain batteries of the sensor nodes and take the WSN out of work really fast (replacing the batteries might not be an option for remote an physically inaccessible deployments). In case of the end-to-end security, malicious

DoS request will travel through the network without being authenticated, between source and destination, thus draining batteries of the nodes on the route. Link layer security mechanisms can perform request authentication immediately after request has entered the network, thus blocking malicious requests from spreading into the network. Unfortunately, majority of the most popular security protocols used in the IP networks nowadays, such as TLS, SSH and IPsec are with end-to-end architecture, which means that they might not be viable in all of the WSN deployments.

In addition to problems discussed above, IP network security solutions might not be reasonable choice for deployment in WSNs, because of the computational resources they require. So in order to deploy existing Internet security solutions in the WSNs they should be optimized.

3.1 Symmetric

The situation with cryptography in WSNs is very similar to TCP/IP stack versus specific WSN protocols. Symmetric cryptography was considered as an only option for WSNs for a long time, mostly because of its computation speed. Asymmetric cryptography, in contrast, was considered too resource consuming for WSNs. Nevertheless there are multiple problems with symmetric cryptography based key distribution mechanisms in WSNs. The major problem is that symmetric cryptography based key distribution mechanisms are too communicationally intensive. And another problem is use of centralized servers to distribute the keys (like Kerberos). Which means that in order to operate correctly wireless nodes should always have connectivity to the centralized key distribution server, which is often impossible due to conditions in which WSNs operate.

There are multiple symmetric cryptography based security solutions proposed for use in WSNs. One of the most popular is TinySec [9], security suite developed in Berkley and used by TinyOS. TinySec has three major security goals: provide access control, message integrity and message confidentiality. To provide access control and message integrity TinySec uses MACs (Message Authentication Code) based on CBC (Cipher-Block Chaining) mode block cipher. And in order to provide confidentiality services TinySec uses Skipjack block cipher in CBC mode with carefully chosen IV (Initialization Vector) structure. Overall TinySec does not use any new symmetric cryptography methods, but relies on old and time-proven cryptography methods, which prove to be lightweight enought for WSNs. Nevertheless, TinySec does not address the most important problem of symmetric cryptography, key distribution mechanisms. Basically any key distribution mechanism can be used with TinySec, but authors describe three very trivial distribution protocols: single network-wide key, per-link keys between neighbor nodes and group keys (similar to single network-wide key, but within limited groups of nodes). Those “key exchange protocols” are really basic and in order to provide better security more advanced key distribution protocols should be employed. But as was already mentioned, more advanced symmetric key distribution protocols cause communicational overhead which is not acceptable by the majority of WSN deployments.

3.2 Asymmetric

Recent developments in ECC (Elliptic Curve Cryptography) have made usage of asymmetric cryptography in WSNs viable. ECC is asymmetric encryption method, based on algebraic structure of elliptic curves over finite fields. Its main advantage, is its ability to provide the same level of security, as RSA, with significantly smaller keys and faster computation time. For example, 160-bit key in ECC provides equivalent security to 1024-bit RSA. There were multiple papers published evaluating energy-efficiency and performance of ECC in WSNs [15, 14, 1]. So far, research in this area concludes that ECC is, on average, much more energy and time efficient when compared to RSA.

The main advantage of asymmetric cryptography over symmetric one, is that key distribution mechanisms, based on asymmetric cryptography, are much more suitable for WSNs and also less communication intensive when compared to symmetric key distribution mechanisms. For example, distribution of public/private keys and certificates for nodes can be performed off band, by preloading them to the nodes before node deployment. When asymmetric keys are deployed to the wireless nodes, authenticated ECDH (Elliptic Curve Diffie-Hellman) can be used, in order to set up symmetric session keys, which will be used for actual bulk data encryption. Analogically to wired networks, both asymmetric and symmetric methods are usually used together (hybrid encryption) in WSNs. In hybrid encryption, asymmetric mechanisms are used to establish symmetric keys and authenticate parties, and symmetric mechanisms are used for bulk data encryption (confidentiality) and integrity. For example, asymmetric cryptography can be used in conjunction with TinySec, providing public-key based key distribution mechanism for TinySec.

3.2.1 Neighbor endorsement mechanism

Wang et al. proposed method for remote access control [15]. The interesting thing about their approach is that it allows wireless nodes to authenticate to remote (multiple hops away) nodes using end-to-end authentication mechanism and simultaneously prevent DoS attacks by using neighbor endorsement mechanism. This means that if node A wants to communicate to node B, which is 10 hops away from node A, it should be endorsed by N neighbor nodes, before it can send the remote request. The technique is based on Shamirs secret sharing, where secret is divided into shares and shares are pre-distributed among the sensor nodes, at any moment N shares of the secret are needed in order to discover the whole secret. The technique works as follows, node A which wants to communicate to node B, authenticates himself to N neighbor nodes using his asymmetric keys, then he acquires secret shares from those neighbors, sums them and gets the secret (S). Now secret S is used, as pairwise key, to communicate with remote node B, which also calculates S, based on his own secret share and access list of node A (describes access permissions). Thus, endorsement mechanism proposed by Wang et al. also allows to setup pairwise key between communicating entities, without using any additional key exchange protocol.

Currently, use of described endorsement mechanism, in

combination with existing end-to-end security protocols for IP networks, does not seem to be viable (at least without breaking interoperability), but future research in neighbor endorsement mechanisms in WSNs, could provide protection against DoS attacks in WSNs and simultaneously allow usage of existing end-to-end security protocols in IP based WSNs. Data aggregation in this case still would be impossible (because of end-to-end security), but protection against DoS could be sufficient, for some of the WSN deployments.

3.2.2 Packet Level Authentication and Host Identity Protocol

Another interesting, asymmetric cryptography based, hop-by-hop protocol which is currently in research stage, is Packet Level Authentication (PLA) [10]. PLA is being designed as an Internet protocol, but authors also encourage its usage in sensor networks. The basic idea of PLA is to protect network itself, by authenticating packets on every hop from source to destination. Thus DoS attacks, for example, can be stopped long before malicious packet will reach its destination. Every packet sent to the network should be signed by senders private key and should include senders certificate signed by a trusted third party, so that each node on a path from source to destination can validate identity of a packets sender and integrity of a packet data.

Disadvantage of current implementation of PLA, is its dependency on centralized entity (trusted third party), which can cause big delays in revoking certificates of compromised nodes in WSNs. A lot of effort in PLA research is being put on improving performance of asynchronous cryptographic algorithms used nowadays. Also, the goal of a project, from the beginning, was to design security system similar to a modern money, where each node could validate authenticity of a packet locally, without resorting to centralized authority (similar to money, everybody is able to validate). Thus eventually research in PLA can lead to some interesting security solutions for WSNs and the Internet in general.

Good property of PLA is that it can be used in conjunction with higher level security solutions, such as HIP (Host Identity Protocol) [12, 7], without breaking interoperability. HIP is end-to-end protocol, which separates identifier (public key) of a node from its locator (IP address), thus enabling authentication, mobility and multihoming for nodes in a network. In nowadays Internet IP address is used as both, identifier and locator of a node. Architecturally HIP is a new layer between Network and Transport layers of OSI model. Thus applications on top of Transport layer use Host Identity (HI) instead of IP address and HI is resolved to actual IP address of the receiver internally by a potocol stack. HIP might be too heavy for some of the deployments of the sensor networks. Firstly, Diffie-Helman based handshake (two round trips), used in HIP to authenticate communicating parties, can be too communicationally intensive for some WSNs. Secondly, puzzle used in handshake, can be computationally too complicated for sensor nodes to solve. And finally FQDN (Fully Qualified Domain Name) to HI and HI to IP address resolution, using DNS or similar system, can also be communicationally too intensive for some of the WSN deployments. Nevertheless real tests on physical sensor nodes should be performed, in order to identify computational/communicational requirements, of Future Internet protocols, such as HIP, and their suitability for actual WSN deployments.

3.3 TLS in Wireless Sensor Networks

TLS (Transport Layer Security) is the most popular security protocols in the Internet nowadays. It provides end-to-end security, between communicating parties, on the transport layer. Certainly TLS is too heavy for internal use by WSNs (usage of TLS in mobile phones confirms that), but it can be used for communication with hosts in the Internet, in order to send some security critical data once in a while. As TLS is very well supported in the Internet, it might be the only option for WSN node, in order to set up a secure session with a host in the Internet.

The major challenge in using TLS in the WSNs, is preserving its interoperability with existing TLS implementations in the Internet and simultaneously making it more lightweight. In order to make TLS more lightweight, new cipher suites (based on ECC cryptography) should be developed. Currently the two most popular TLS cipher suites in the Internet are TLS_RSA_WITH_3DES_EDE_CBC_SHA (TLS 1.1) and TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (TLS 1.0), which are too heavy for WSNs. Good thing about TLS is that it is independent from the cipher suite in use, so new cipher suites can be developed for TLS.

The main property that could be optimized in TLS protocol, is the number of cipher suites supported by client and server and sent in Client Hello and Server Hello messages. In WSNs, support for only one cipher suite could be enough, so cipher suites list could contain only one entry. Also Wander et al. claim [14], that ECC based public key certificates can be reduced in size to 86 bytes. The only problem with reducing certificate sizes is that it could break interoperability with other TLS implementations.

Overall TLS protocol is quite difficult to optimize for WSNs, because interoperability with other TLS implementations can be easily broken. TLS can be used in WSNs, in rare cases when security critical data needs to be sent to hosts in the Internet, otherwise for internal use in WSNs, it is more feasible to use more lightweight secure session establishment protocols designed specifically for WSNs.

4 Analysis

Although implementing lightweight and functional TCP/IP stack on WSN nodes is possible, using security protocols designed for the IP networks is not always feasible for the following reasons.

Firstly, security protocols used in the todays Internet are not architecturally suitable for WSNs. WSNs are usually data-centric networks, which means that nodes on the route from A to B manipulate the data being routed in the packets, usually by performing data aggregation, decreasing amount of duplicated data being forwarded. Data-centrism in WSNs, means that end-to-end security is not suitable for WSNs and as majority of security protocols in todays Internet are based

on end-to-end architecture, it makes them unsuitable for all of the WSN deployments. For this reason link-layer security solutions are popular in WSNs, where security is provided from hop-to-hop, so every hop can decrypt the data and manipulate it. Link-layer security is also not ideal solution, as it has problem of compromised nodes. Compromised nodes are in control of malicious users, which can get access to the data being routed by the nodes.

Secondly, security protocols used today in the Internet were initially designed for nodes with high computational capabilities and permanent power supply. Protocols, such as TLS, are not efficient in mobile devices, thus their performance in WSN nodes will be even worse. Nevertheless, if nodes have TCP/IP connectivity and they are able to communicate with hosts in the global Internet, usage of modern security solutions deployed in the Internet might be useful for them. One of the user cases, when WSN node could communicate with the host in the Internet, using for example TLS protocol, is when WSN node has security critical data which has to be sent to the machine in the Internet. In this case standard security solutions may be the only option for the WSN node. It is hard to estimate how much will new cryptographic algorithms, such as ECC, improve the performance of Internets security protocols in WSNs. In order to estimate that, actual implementations of those protocols should be made and tested in sensor networks.

Thirdly, end-to-end security protocols are vulnerable to DoS attacks and because of the fact, that DoS attacks in WSNs can quickly take WSN out of operation, there should be some mechanism for preventing DoS attacks in almost every WSN. Unfortunately majority of security protocols used in the Internet cannot provide protection against DoS.

It is clear that WSNs connected to the Internet should be able to communicate with hosts in the Internet over secure channels. Taking into account, that it is very difficult to make radical changes in the Internet, such as introducing new protocols, the only viable option for sensor networks is to adopt already existing security solutions. But the big question is how to make them efficient in the Wireless Sensor Networks, without breaking the interoperability with existing implementations.

5 Conclusions

We have surveyed the state of the art research in adopting Internet Protocols for Wireless Internetworking. Such TCP/IP stack implementations for WSNs, as uIP and uIPv6, have shown that IP technologies are viable in Wireless Sensor Networks and that it is possible to connect WSNs to the global Internet, without usage of proxies and overlay networks.

Security is extremely important in the Internet nowadays and to fully connect WSNs to the Internet TCP/IP is not enough, security should also be there. The next stage for research in sensor internetworking is to adopt Internets security solutions for WSNs. In this paper we have reviewed, some of the recent research in the area of cryptography, related to WSNs, and have shown that implementing Internets security protocols in the WSNs is maybe event more difficult task, than implementing TCP/IP in WSNs. But eventually, in order for WSNs to be fully functional in the Internet, some support for Internets security protocols should be on every sensor node connected to the Internet.

References

[1] F. Amin, A. H. Jahangir, and H. Rasifard. Analysis of Public-Key Cryptography for Wireless Sensor Networks Security. In *Proceedings of World Academy of Science, Engineering and Technology*, 2008.

[2] A. Dunkels. Home Page of Contiki OS. `http://www.sics.se/contiki/`. Referenced at 14.04.2009.

[3] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 85–98, San Francisco, California, USA, May 2003. ACM.

[4] A. Dunkels, T. Voigt, and J. Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004)*, Berlin, Germany, Jan. 2004.

[5] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, and J. Schiller. Connecting Wireless Sensornets with TCP/IP Networks. In *Proceedings of the Second International Conference on Wired/Wireless Internet Communications (WWIC2004)*, Frankfurt (Oder), Germany, Feb. 2004.

[6] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008)*, Raleigh, North Carolina, USA, Nov. 2008.

[7] A. Gurtov. *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. John Wiley & Sons, June 2008.

[8] J. W. Hui and D. E. Culler. IP is dead, long live IP for wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 15–28, Raleigh, North Carolina, USA, 2008. ACM.

[9] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, Baltimore, MD, USA, 2004. ACM.

[10] D. Lagutin. Redesigning Internet - The Packet Level Authentication architecture. Master's thesis, Helsinki University of Technology, 2008.

[11] A. Liu and P. Ning. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 245–256. IEEE Computer Society, 2008.

[12] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. RFC 5120, Host Identity Protocol. `http://tools.ietf.org/html/rfc5201`, April 2008.

[13] G. Mulligan, C. O'Flynn, M. Durvy, J. Abeillé, P. Wetterwald, B. Leverett, E. Gnoske, M. Vidales, N. Tsiftes, N. Finne, and A. Dunkels. Seamless Sensor Network IP Connectivity. In *Proceedings of the 6th European Conference on Wireless Sensor Networks, EWSN 2009*, Cork, Ireland, Feb. 2009.

[14] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz. Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pages 324–328. IEEE Computer Society, 2005.

[15] H. Wang, B. Sheng, C. C. Tan, and Q. Li. Comparing Symmetric-key and Public-key Based Security Schemes in Sensor Networks: A Case Study of User Access Control. In *Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems*, volume 00, pages 11–18. IEEE Computer Society, 2008.