Security Analysis of Reliable Transport Layer Protocols for Wireless Sensor Networks

Levente Buttyán and László Csik
Laboratory of Cryptography and Systems Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics
Email: buttyan@crysys.hu

***Abstract*—End-to-end reliability of communications is an important requirement in many applications of wireless sensor networks. For this reason, a number of reliable transport protocols specifically designed for wireless sensor networks have been proposed in the literature. Besides providing end-to-end reliability, some of those protocols also address the problems of fairness and congestion control, and they are all optimized for low energy consumption. However, in this paper, we show that most of those protocols completely neglect security issues. As a consequence, they ensure reliable communications and low energy consumption only in a benign environment, but they fail in a hostile environment, where an adversary can forge or replay control packets of the protocol. More specifically, our analysis shows that control packet injection and replay can cause permanent loss of data packets, and thus, such misdeeds make the hitherto reliable protocol unreliable. In addition, even if the protocol can recover from such an attack, the recovery overhead caused by forged or replayed control packets can be large, which gives an opportunity for energy depletion attacks.**

***Keywords*-wireless sensor networks; transport protocols; security; reliability; dependability.**

I. Introduction

Wireless Sensor Networks (WSNs) consist of a large number of resource constrained sensor nodes and a few more powerful base stations. The sensors collect various types of data from the environment and send those data to the base stations using multi-hop wireless communications.

There are applications of WSNs where the sensors capture and transmit high-rate data (e.g., multimedia sensor networks [1]). In those applications, special mechanisms are needed to ensure end-to-end reliability and congestion control. Such mechanisms are usually implemented in the transport layer of the communication stack in form of a transport protocol. It is widely accepted that transport protocols used in wired networks (e.g., the well-known TCP) are not applicable in wireless sensor networks, because they perform poorly in a wireless environment and they are not optimized for energy consumption. For this reason, a number of transport protocols specifically designed for WSNs have been proposed in the literature (see [2] for a survey). The main design criteria that these protocols try to meet are end-to-end reliability, congestion control, and energy efficiency.

Interestingly, despite the fact that WSNs are often envisioned to operate in hostile environments, none of the proposed WSN transport protocols address security issues. As a consequence, the proposed protocols meet the above requirements only in a benign environment, but they fail in a hostile environment. In particular, in this paper, we show that most of the proposed WSN transport protocols fail to provide end-to-end reliability and are subject to increased energy consumption in the presence of an adversary that can replay or forge control packets of the protocol.

Note that replaying and injecting forged control packets into a WSN can be easily done due to the wireless nature of the medium, therefore, the assumption that WSN transport protocols may be subject to these kinds of attack are not far fetched. One could even consider a stronger adversary model that allows for the deletion of control and data packets, however, in such a model, it is theoretically impossible to ensure reliable communication [3]. Hence, we restrict our analysis to the weaker adversary model where only control packet replay and injection are allowed.

We must also note that there are many papers on security issues in WSNs (see e.g., [4] and the references therein), but to the best of our knowledge, the security of the transport layer in WSNs has been neglected so far. Indeed, most of the literature on WSN security deals with MAC layer and network layer security issues, and key management problems. In contrast to those works, we focus on the security issues at the transport layer in this paper.

The rest of the paper is organized as follows: In Section II, we give a high level summary of the various acknowledgement schemes used to provide reliability in communications. In Section III, we specify our attacker model and define metrics required to measure the impact of an attack. In Section IV, we describe specific WSN transport protocols and analyze them with respect to security. Finally, in Section V, we summarize the lessons that we learnt from the analysis, and we conclude our paper in Section VI.

II. Transport layer reliability mechanisms

Communications in WSNs usually take place between the sensor nodes and the base stations, and it is important to distinguish the direction of those communications. In case of *upstream* communication, the sender is a sensor node, and the receiver is a base station, while in case of *downstream* communication, these roles are reversed.

The goal of the sender is to reliably transmit to the receiver a full *message* that may consist of multiple *fragments*. If a fragment is lost, it must be retransmitted. This may be done in an *end-to-end* manner, where the source node itself repeats the lost fragment, or on a *hop-by-hop* basis, where intermediate nodes can cache and retransmit fragments if they are lost.

A reliable protocol can only detect fragment losses, if there is some kind of feedback in the system. Typically the following types of feedbacks are used:

- *Acknowledgement (ACK):* Acknowledgements can be:
  - *Explicit* – Upon receiving a fragment the node sends back a confirmation on it. An explicit ACK can confirm the reception of a *single* or *multiple* fragments.
  - *Implicit* – When a node overhears his neighbor forwarding a fragment sent by the node, it can assume that the delivery of the fragment to that neighbor was successful. This method can only confirm the delivery of a single fragment.
- *Negative Acknowledgement (NACK):* If a node somehow becomes aware of the fact that it did not receive a fragment, it can explicitly send a request for retransmission. A NACK can also refer to a single or multiple requested fragments. In multiple NACK schemes the notion of *loss window* refers to a range of lost fragments.
- *Selective Acknowledgement (SACK):* It is a combination of an explicit single or multiple ACK – used for the last fragments received in-order – and multiple ACKs for other fragments that were also received, but which are out of order.

Finally, we should mention the following two important theoretical problems related to NACK based schemes:

- *Lost last fragment problem:* Most NACK-based protocols use sequence numbers to detect fragment losses. If a node receives a fragment with a sequence number higher than expected, then it concludes that a fragment is lost. However, this method cannot detect if the last fragments of a stream are lost, since they will not going to be followed by a fragment with a higher sequence number. NACK based schemes must implement a specific solution for this problem.
- *Lost full message problem:* In wireless networks, it is possible, that an entire message is lost during transmission, as losses often occur in bursts, and messages in WSNs tend to consist of a few (often only one) fragments. Loss of an entire message cannot be directly detected by NACK based schemes, as the receiver never becomes aware of the existence of the message. This problem also requires special handling in NACK based schemes.

III. Attacker model

We assume that the attacker can eavesdrop the communications between any two nodes in the network, and he can forge and inject control packets anywhere in the network with a specified transmit power. However, we do not allow the attacker to delete (e.g., by jamming) control and data packets. We understand that such attacks are possible in wireless networks, but if we allowed them, then no transport layer protocol would be able to ensure the end-to-end reliability of the communications. In other words, deletion attacks must be addressed in another layer, typically below the transport layer.

Our attacker model is not affected by security mechanisms applied at the application layer, because we are not interested in the content of the data packets, and the attacker in our model only injects transport layer control packets. In contrast to this, security mechanisms implemented below the transport layer (e.g., link layer packet authentication) would be useful to prevent some of the attacks, but in fact, none of the proposed transport protocols assume any security mechanisms at lower layers, and therefore, we will not assume their presence either.

Attacks against WSN transport layer protocols come in two flavors: attacks against reliability and energy depleting attacks.

Reliability in the context of transport protocols refers to reliable data transfer. In particular, a reliable transport protocol must be able to guarantee that every packet loss is detected and that lost packets can be retransmitted until they reach their destination. Thus, an attack against reliability is considered to be successful, if either a packet loss remains undetected, or the attacker can permanently deny the delivery of a packet.

Energy depleting attacks are unique to sensor networks. In this case the goal of the attacker is to force the sensor nodes to perform energy intensive operations, in order to deplete their batteries, and thus, to decrease the lifetime of the network. In WSNs, the overall energy consumption of a sensor node is highly proportional to the number of the packets transmitted by the node. Therefore, an energy depleting attacker may try to coerce the sensor nodes to retransmit packets. In this case, we measure the successfulness of an attack by the ratio between the number of packets injected by the attacker in the network and the overall number of packets sent by the sensor nodes due to those injected packets.

IV. Analysis of the existing transport protocols

In this section, we analyze four WSN transport protocols in details with respect to their resistance to malicious attacks.

We have chosen these protocols because they are relatively well-known and frequently cited, and their descriptions are sufficiently detailed such that they permit the analysis. We note that several other protocols have similar problems (see Subsection IV-E).

A. *PSFQ*

PSFQ (Pump Slowly, Fetch Quickly) [5] is a general-purpose transport protocol, that provides downstream reliability with hop-by-hop recovery. The name of the protocol originates from the delivery method it uses. During the transmission, data fragments are transferred (pumped) with a relatively small speed, but if an error is detected, the protocol tries to quickly recover (fetch) the missing fragments from immediate neighbors.

1) Protocol overview: PSFQ uses a multiple NACK-based scheme to achieve reliability. It has three different working modes:

- Pump Operation – This mode is responsible for the normal data transfer. Each data fragment contains a message ID, a message length field, a sequence number and a TTL (Time To Live) value. When a node receives a fragment, it checks its local data cache and discards any duplicates. PSFQ buffers every new fragment, decreases their TTL value and schedules them to forward, if there are no gaps in the sequence numbers and TTL is not zero. Each scheduled fragment is delayed for a random period before it is forwarded. Within this random period, the node counts the number of times the same fragment is heard from neighboring nodes. If this counter reaches 4 before the scheduled rebroadcast, the transmission is canceled.
- Fetch Operation – A node goes into fetch mode once a gap is detected in the sequence numbers. PSFQ uses NACKs with three fields: message ID, message length, and loss windows. Each node attempts to obtain all lost fragments in a single fetch operation. To reduce collisions, neighbor nodes wait a random time before transmitting missing fragments. Other nodes that have the same missing fragment will cancel their scheduled retransmission if they hear a repair for the same fragment. NACKs are aggressively repeated for non-received fragments. However, NACK packets are only propagated once, and only after the number of repetition for the same NACK exceeds a predefined threshold. To tackle the lost last fragment problem, each node can enter fetch mode proactively and send a NACK for the next missing fragment.
- Report Operation – This working mode was designed to feedback data delivery status information, however it has marginal influence on our the security analysis.

2) Weaknesses of the protocol: One important problem with PSFQ is that it does not deal with the lost full message problem. Hence the protocol is not reliable, especially when the implementing WSN application uses relatively small messages consisting only of a few fragments.

Another general problem of the protocol is the inappropriate handling of TTL values. Due to the randomized transfer delays used by the forwarding method, it is possible that a copy of a fragment arrives to a node from the source earlier on a longer path than on the shortest one. This specific fragment will be scheduled for forwarding, even if it has a smaller TTL value than other fragments for the same message ID. If fragments arriving later, but with higher TTL value are discarded, then it is possible that the destination will not going to receive any copy of the discarded fragment. Also, an attacker can inject a fragment with a given message ID, message size, and sequence number but with a TTL value as low as 1. This fragment might prevent the proper propagation of the valid fragment, which can lead to permanent fragment losses. If duplicated fragments with higher TTL values are not discarded but also propagated, and NACKs can force retransmission of fragments with zero TTL, then this problem can be fixed, but it will not going to be very efficient.

Another problem stems from the fact that nodes cancel their scheduled transmission of a given fragment if they hear that fragment being transmitted by neighboring nodes 4 times. This means that an attacker can send enough spoofed copies of a fragment under different identities to force the cancelation of a transmission. Even if the intended receiver later detects the fragment loss and broadcasts a NACK for the fragment, the same method can be used by the attacker to force the cancelation of the retransmission by immediately injecting a false response to the NACK. Thus, in PSFQ, it is possible to permanently delete arbitrary messages from the system.

Energy depleting attacks are also possible against PSFQ. In general, an injected fake NACK forces the nodes to unnecessarily retransmit a fragment. For multiple NACK schemes one packet can provoke the retransmission of multiple fragments, which multiplies the impact of the attack. The problem with PSFQ, is that it does not limit the size of the loss window, so it can be as high as the size of the largest transmitted message. Similarly, an attacker can inject a fragment with a large sequence number, which generates a large loss window potentially in many nodes.

Another attack against the protocol would be to inject a false fragment with a new message ID, message length of 2, sequence number of 1, and a TTL value as high as possible. Since this is the first fragment of the message, it does not generate a gap in the sequence numbers, consequently every node will immediately propagate the fragment and due to the high TTL value, it will reach every node in the network. After receiving this fragment, every node will believe that only the last fragment is missing from the message, and so every node will proactively enter into fetch mode and aggressively send out NACKs for the second half of the message.

B. DTC

DTC (Distributed TCP Caching) [6] is a specifically modified TCP protocol for WSNs. It provides both up and downstream reliability with hop-by-hop recovery.

1) Protocol overview: This protocol implements a special SACK-based algorithm, where a SACK packet contains an ACK field that contains the sequence number of the last fragment that was received in-order, and a SACK field that lists the sequence numbers of additional fragments received out-of-order. It is important to note, that the SACK field also works as a multiple NACK, since it implicitly lists all missing fragments.

DTC assumes that each intermediate node between a source S and a destination D can store only a single fragment. Periodically, D sends a SACK packet to S. Along the path to S each intermediate node I examines the SACK packet. If it acknowledges a fragment that is stored by I, then I deletes that fragment from its cache. If the SACK negatively acknowledges a fragment that is stored by I, then I retransmits the missing fragment and inserts its sequence number into the SACK field. Finally, I forwards the SACK packet to the direction of S. If an intermediate node can retransmit all missing fragments listed in a SACK, it drops the SACK.

2) DTC weaknesses: As opposed to NACK-based protocols, ACK-based schemes can achieve full reliability without any further extension. Also, if an attacker injects an ACK into a system, it does not generate any additional traffic. However injected ACK packets can be very dangerous. In general, protocols that use ACKs assume, that an arbitrary fragment which was acknowledged explicitly or implicitly, can be deleted from the system as it arrived to its destination. Since an attacker can forge and insert fake ACKs for fragments that are actually lost, he can cause permanent fragment losses.

In DTC, this attack can be realized easily. A SACK lists multiple lost fragments, so an attacker can forge and inject another SACK that acknowledges all missing fragments. With this single packet, he can provoke multiple fragment losses.

Beside the previous vulnerability, energy depleting attacks are also feasible against DTC. This is so, because the SACK field also functions as a multiple NACK. Injecting a SACK with a large loss window generates a large traffic of retransmitted fragments.

In addition, the previous two attacks can be easily combined by injecting an inverse SACK packet to the system that requests the retransmission of every fragment that was actually received by D while acknowledges every lost fragment.

C. Garuda

Garuda [7] provides a scalable solution for sink to all sensors communication. It is a downstream reliability scheme using single NACKs, and a local recovery scheme realized by special CORE nodes.

1) Protocol overview: In this protocol, every 3rd node is a CORE node, serving as a local and designated loss recovery server. Nodes use implicit multiple NACKs to recollect missing fragments, where the NACK is the sequence number of the last message ID the node has received so far.

To protect against lost full messages, Garuda uses a special *Wait-for-First-Packet* (WFP) pulse, which is a finite series of short duration pulses repeated periodically. Sensor nodes upon reception of the pulses, also start pulsing. The sink after pulsing for a finite duration transmits the first fragment. If a node receives the first fragment, it stops pulsing the WFP and broadcasts the first fragment. Therefore, WFP serves as an implicit NACK for the first fragment, while termination of WFP pulsing is an ACK for it. As first messages can store the size of the data that is going to be transferred, reliable transfer of the first fragment can solve the lost last fragment problem.

2) Garuda weaknesses: The major problem with Garuda is the unconditional propagation of WFP pulses. If an attacker injects a WFP into the system, every node will immediately rebroadcast it until an undefined time. Even if nodes stop pulsing after some consecutive WFPs, the impact of this attack is proportional to the number of nodes in the network.

D. RBC

RBC (Reliable Bursty Convergecast) [8] implements a special window-less block acknowledgement scheme that can be used for hop-by-hop recovery.

1) Protocol overview: In RBC, intermediate nodes cache every fragment they receive. If a fragment is acknowledged, it is deleted from the cache, otherwise it is repeated n times. RBC implements a special cache queuing model capable of efficiently delivering out-of-order fragments, which is useful for bursty communication. The protocol uses multiple (block) ACKs.

2) RBC weaknesses: Unlike hybrid NACK-ACK schemes (such as DTC), in a protocol that uses solely ACKs, the receiver cannot request the retransmission of a fragment and the sender will never repeat an acknowledged fragment. Therefore a false ACK on a lost fragment guarantees fragment loss contrary to other schemes where recovery might be feasible, although it can have a big overhead. Moreover, as RBC supports block acknowledgements, it is possible to acknowledge every fragment stored by a node in one ACK. Upon reception of this packet, the node will completely empty its cache, which can lead to fragment losses with high probability.

E. Other protocols

There are numerous other WSN transport protocols [9], [10], [11], [12], [13], [14], [15], [16] that provide reliability. We analyzed all of them, however, due to space limitations, we cannot present that analysis in details in this paper. In any case, they use the same techniques that we have studied above in the context of the PSFQ, DTC, Garuda, and RBC protocols, and they all have similar weaknesses.

V. Lessons learned

Both ACK and NACK-based schemes are vulnerable to injected control packets. In general, ACK-based schemes are vulnerable to attacks against reliability, while NACK-based protocols are only vulnerable to energy depleting attacks. For both methods, the multiple version is significantly weaker. Moreover, if a protocol combines ACK and NACK packets – like SACK-based schemes – then it inherits the problems of both methods.

In practice, attacks against reliability are more important than energy depleting attacks, therefore NACK schemes may be preferred to ACK schemes. NACK schemes are also more suitable for multi-hop communication. However, pure NACK-based schemes have two inherited weaknesses, the lost last fragment and the lost full message problem.

It is relatively easy to solve the lost last fragment problem by informing the destination node about the number of fragments in the message at the beginning of the communications (e.g., in the first fragment). For the lost full message problem, we cannot identify a satisfactory solution for the time being. Garuda was the only protocol that tried to tackle the problem, however it led to a serious energy depleting attack. Perhaps this specific problem requires a dedicated ACK-based technique, while NACKs should be used everywhere else in the communication. It is important to note, that these problems only exist for event driven applications. For continuous communications, NACK-based schemes can be directly applied, as there are no first or last fragments.

Without adequate authentication – probably using cryptographic solutions – it seems impossible to fully protect a NACK-based protocol against energy depleting attacks. However, the impact of these attacks can be kept low with some precautions. If multiple NACKs are used, the loss window must be maximized and hop-by-hop or some kind of local retransmissions should be used instead of end-to-end recovery.

It is hard to estimate an impact threshold where these types of energy depleting attacks become dangerous. Ideally, the ratio between the number of injected packets and the number of packets generated due to the injected packets should be constant, however, this objective might be difficult to achieve, if possible at all. On the other hand, if this ratio is proportional to the size of the network, then the attack can definitely be considered to be dangerous.

VI. Conclusion

In this paper, we analyzed reliable transport protocols proposed for wireless sensor networks with respect to their resistance to malicious control packet replay and injection attacks. Our analysis shows that all of the proposed reliable WSN transport protocols have vulnerabilities. In particular, control packet replay and injection attacks can lead to unrecoverable data loss, jeopardizing the basic reliability requirement that these protocols are supposed to satisfy. In addition, even if a protocol can recover from packet loss, the recovery overhead caused by malicious control packets can be excessive, opening the doors for energy depleting attacks.

While these protocols were not designed to resist malicious attacks, and from this point of view, they cannot be blamed to fail in hostile environments, we must emphasize that we are not aware of *any* reliable WSN transport protocol that is designed with malicious attacks in mind. Although authentication in lower layers would help to prevent most of the attacks we described, it is likely to be an inefficient approach. The reason is that many WSN transport protocols require intermediate nodes on a path to cache data fragments until they can be sure that they have been delivered. Hence, control packets must be authenticated such that these intermediate nodes can verify them, meaning that one needs a broadcast authentication scheme. But those schemes are typically computationally expensive, and therefore, they should not be used at lower layers to protect each and every packet. Thus, the problem needs to be solved at the transport layer, by enhancing transport protocols with their own security mechanisms. Our future work is concerned with proposing such security mechanisms for WSN transport protocols.

Acknowledgment

The work described in this paper is based on results of the WSAN4CIP Project (www.wsan4cip.eu), which receives research funding from the European Community's 7th Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

The first author has been further supported by the Hungarian Academy of Sciences through the Bolyai János Research Fellowship.

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, pp. 921–960, 2007.

[2] C. Wang, K. Sohraby, B. Li, M. Daneshmand, and Y. Hu, "A survey of transport protocols for wireless sensor networks." *IEEE Network*, vol. 20, no. 3, pp. 34–40, 2006.

[3] L. Lamport, R. E. Shostak, and M. C. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[4] J. Lopez and J. Zhou, *Wireless Sensor Network Security.* IOS Press, 2008.

[5] C.-Y. Wan and A. T. Campbell, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," in *Proc. ACM WSNA 02*, Sept. 28 2002.

[6] A. Dunkels, J. Alonso, T. Voigt, and H. Ritter, "Distributed tcp caching for wireless sensor networks," in *Proc. 3rd Annual Mediterranean Ad Hoc Net. Wksp.*, June 2730 2004.

[7] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz, "A scalable approach for reliable downstream data delivery in wireless sensor networks," in *Proc. ACM MobiHoc 04*, May 2426 2004.

[8] H. Zhang, A. Arora, Y. ri Choi, and M. G. Gouda, "Reliable bursty convergecast in wireless sensor networks," *Comput. Commun.*, vol. 30, no. 13, pp. 2560–2576, 2007.

[9] F. Stann and J. Heidemann, "RMST: Reliable Data Transport in Sensor Networks," in *Proc. IEEE SNPA 03*, May 11. 2003.

[10] S. G. Y. G. Iyer and S. Venkatesan, "STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks," in *Proc. IEEE ICCCN 2005*, Oct. 1719 2005.

[11] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems.* New York, NY, USA: ACM, 2004, pp. 13–24.

[12] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. E. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: A reliable bulk transport protocol for multihop wireless network," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-169, Dec 2006. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-169.html

[13] J. Paek and R. Govindan, "RCRT: rate-controlled reliable transport for wireless sensor networks," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems.* New York, NY, USA: ACM, 2007, pp. 305–319.

[14] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference-aware fair rate control in wireless sensor networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 63–74, 2006.

[15] O. Gnawali, K.-Y. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler, "The tenet architecture for tiered sensor networks," in *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems.* New York, NY, USA: ACM, 2006, pp. 153–166.

[16] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz, "ESRT: event-to-sink reliable transport in wireless sensor networks." in *Proc. ACM Mobihoc 03.* ACM, June 13 2003, pp. 177–188.