Design Rationale of a Cross-layer, Trustful Space-Time Protocol for Wireless Sensor Networks

Davi Resner and Antônio Augusto Fröhlich
Software/Hardware Integration Lab
Federal University of Santa Catarina
PO Box 476, 88040-900 - Florianópolis, SC, Brazil
{davir,guto}@lisha.ufsc.br

***Abstract*—In this paper, we introduce a cross-layer, application-oriented communication protocol for Wireless Sensor Networks (WSN). TSTP – Trustful Space-Time Protocol – integrates most services recurrently needed by WSN applications: Medium Access Control (MAC), spatial localization, geographic routing, time synchronization and security, and is tailored for geographical monitoring applications. By integrating shared data from multiple services into a single network layer, TSTP is able to eliminate replication of information across services, and achieve a very small overhead in terms of control messages. For instance, spatial localization data is shared by the MAC and routing scheme, the location estimator, and the application itself. Application-orientation allows synergistic co-operation of services and allows TSTP to deliver functionality efficiently while eliminating the need for additional, heterogeneous software layers that usually come with an integration cost.**

***Keywords*-Wireless Sensor Networks; Application-oriented; Protocol; Trustful; Cross-Layer; Geographic; Space-Time**

I. Introduction

Wireless Sensor Networks have been the focus of intense research for well over a decade by now. Several physical layers have been proposed, along with a myriad of medium access and routing protocols. Such protocols have been made energy-aware; aggregation and fusion strategies have been employed; basic infrastructures have been enriched with location, timing, and security protocols; operating systems have been designed to support higher-level abstractions, along with large-scale management systems designed to handle the produced data properly. We are currently seeing these networks being connected to the Internet of Things (IoT).

Extensive research is also being carried out on cross-layer optimizations for wireless communication [9] and WSN [15] protocols. These works focus on taking into account related information given by one layer to make decisions at a different layer in the communication stack, and have been proven to be greatly effective. Such efforts have been often carried out in such a way as to preserve the interface of the original individual protocols and maintain the modularity of traditional layered architectures. In our opinion, however, this approach misses a great opportunity to design a truly application-oriented protocol for WSN, in which services are intimately combined to optimize resources at the same time as they address real and specific application needs.

In this paper, we describe the design of *Trustful Space-Time Protocol* (TSTP), an application-oriented, cross-layer protocol for WSN. Instead of focusing on keeping the original protocol interfaces in a modular, layered architecture with shared data, TSTP focuses on efficiently delivering functionality recurrently needed by WSN applications: trusted, timed, geo-referenced, SI-compliant data that is resource-efficiently delivered to a sink. TSTP delivers this functionality directly to the application in the form of a complete communication solution, which allows the design of optimized, synergistic co-operation of protocols while eliminating the need for additional, heterogeneous software layers that come with an integration cost and often result in replication of data.

By including data from the spatial and temporal localization services in every message, TSTP devices are able to localize themselves mostly by observing network traffic. A fully-reactive geographic routing and MAC scheme is used, which also takes advantage of this data. Finally, the security mechanisms exploit temporal synchronization as well, resulting in a small total overhead in terms of control messages. A reduction from 11 messages to 6 is achieved in the node bootstrapping process in comparison with a Standalone design (Section IX).

II. Related Work

TCP/IP is at the foundation of the internet today. Ordinary TCP implementations have been tuned for decades to traditional networks, made up of wired links and stationary hosts. TCP now performs well on such networks, but it has been shown that traditional implementations perform poorly on wireless networks [9]. Although there are proposals for TCP/IP-based WSN [23], many authors believe and have shown that more efficient solutions can be devised.

Cross-layer designs have been shown to be very efficient in optimizing wireless networks [20]. Any design that breaks in some way the black-box characteristic of the classic TCP/IP stack can be considered cross-layer [9]. In practice, cross-layer designs usually work by taking information from one or more layers of a typical layered stack to optimize a set of parameters or make a decision in another layer or set of layers.

For instance, CEDA [4] is a network-layer protocol for WSN which takes into account the energy level of neighboring nodes as a weight factor when routing data, avoiding nodes with less remaining energy [15]. The authors show that only

Header Format								
Bits: 3	**1**	**2**	**2**	**8**	***3*sb + tb***	***3*sb + tb***	**0 or 32**	9 - 54 octets
Message Type	Time Request	Spatial Scale	Temporal Scale	Location Confidence	Last Hop x,y,z,t	Origin x,y,z,t	Location Deviation	

Figure 1: TSTP message header format.

5% of the nodes have their energy depleted when all the nodes are dead using the AODV protocol [15]. TSTP makes extensive use of this kind of information sharing strategy.

From the myriad of cross-layer proposals, a minority involves the application layer. In [13], the authors propose an application-driven cross-layer optimization for video streaming over wireless networks, involving the application, data link and physical layers. The main goal of the strategy is to maximize an application-specific metric, the peak signal-to-noise ratio (PSNR), which closely represents user-perceived video quality. To achieve that, a cross-layer optimizer observes transmission data and packet error rate, data packet size and channel coherence time. Based on these parameters and a model predicting the behavior of the physical layer, the optimizer selects values of: video source rate (application layer), time slot allocation (data link layer), and modulation scheme (physical layer) which maximizes the average PSNR among all users. TSTP relates in the sense that its main focus is on fulfilling identified application needs.

There are also works that try to identify WSN needs, but they usually focus on aspects such as MAC, routing, QoS, mobility, and sometimes security [5], [15], [9], and are careful in preserving the modularity of the traditional layered model. On the other hand, TSTP focuses on geo-monitoring WSN applications and aims at identifying and intimately integrating services recurrently needed in this context: spatial localization, geographic routing, time synchronization, security and a domain-convenient API.

III. Application Interface

Communication in TSTP occurs between sensors and a specific node called *sink*. Sensors are nodes that can measure and report one or more types of data about the environment (e.g. temperature, luminosity) with a certain precision and maximum frequency. A sink is a node that is interested in such information. From the point of view of a sink, given that the information acquired about a desired space-time region is trustful and was measured with a certain precision, it does not matter which particular sensor measured it. TSTP is designed so that sinks announce what information they are interested in, and sensors deliver it if able.

Figures 1 and 2 show the main TSTP messages with which sinks request and sensors send information. A sink node announces interest in a physical quantity using an *Interest* message. This message specifies the space-time region in which the interest is valid (a sphere in space and a time interval), the periodicity of responses, SI unit, minimum required precision, and the response mode, which can assume two values:

0 Every able sensor in the region should respond;
1 Only one responding sensor is enough.

Interest Message						
Bits:	***4*sb + 2*tb***	***tb***	**40**	**7**	**1**	22 - 104 octets
Header	Region x,y,z,r,t,Δ_t	Period	Data S.I unit	Precision	Response Mode	

Data/Report Message				
Bits:	**40**	**variable**	**128**	30 - 75 octets + Data
Header	Data S.I unit	Data	MAC	

Figure 2: TSTP Interest, Data and Report messages.

Code	Unit	Scale	Size (bits)	Maximum Value
00	cm	50	8	127.5 m
01	cm	1	16	655.3 m
10	cm	25	16	16382.5 m
11	cm	1	32	42949.6 km

Table I: Spatial Scale Codes

When receiving an Interest message, a sensor checks if it is inside the desired region and able to measure the requested quantity with the required precision. If so, the Interest is saved and the sensor automatically responds with a *Data* message every *period* of time until the Interest expires or it leaves the interested area.

The Data message's payload consists simply of a measurement and unit. It can be made trustful by a Message Authentication Code (MAC) and encryption with the key derivation strategy exposed in Section VII.

Messages are routed geographically to the destination, as explained in Section V, so the sink is oblivious of which specific sensor shall receive the Interest message or respond with data.

Tables I and II show the meaning of the values of the scaling bits. These bits define the size of each spatial (denoted *sb*) and temporal (*tb*) values in the corresponding message, as well as a multiplier to be applied. For instance, with time scale value 01, time fields shall have 16 bits each and represent multiples of $500ms$, so a value of 42 would represent $21s$. Such scaling makes these fields adaptable to different application scenarios, which may vary from, for instance, monitoring a small room to a large forest.

The message type is indicated by the *Message Type* bits, according to Table III. The *Bootstrap* messages, as well as the other unmentioned fields, will be explained in the rest of the paper.

The units used in Interest and Data messages are SI base

Code	Unit	Scale	Size (bits)	Maximum Value
00	s	1	8	4.25 minutes
01	ms	500	16	9.1 hours
10	ms	1	32	49 days
11	ms	500	32	69 years

Table II: Temporal Scale Codes

Code	Message Type	Code	Message Type
000	Interest	100	Bootstrap 0
001	Data	101	Bootstrap 1
010	Report	110	Bootstrap 2
011	*Reserved*	111	Bootstrap 3

Table III: Message Types

or SI derived units, and their representation is inspired by the IEEE1451.0 standard [11]. Radian and Steradian, together with the seven SI base units, form the nine base units of the standard. Derived units are formed by the product of base units raised to a power (e.g. pressure is N/m^2), therefore, storing the exponents of each base unit is enough to represent any derived unit. For example, the unit of acceleration is m/s^2, and we can represent it as $rad^0\ sr^0\ m^1\ kg^0\ s^{-2}\ A^0\ K^0\ mol^0\ cd^0$ or as the sequence $0, 0, 1, 0, -2, 0, 0, 0, 0$.

In TSTP, each exponent is represented by 4 bits. Each value is multiplied by two to achieve a resolution of $1/2$. After that, 8 is added to the exponent (two's complement) to represent them as signed quantities. Therefore, exponents can assume values from -8 to +7. For example, the exponent $1/2$ would have 9 as its decimal representation since $2\times1/2+8 = 9$. Table IV presents additional examples. The *enum* field is useful in particular cases to indicate a ratio of the same unit (dimensionless); for example, when measuring Strain (unit m/m), *enum* is set to 1. When the output of a transducer represents positions of switches (e.g. "on", "off", "open", "closed"), it cannot be derived from base units and enumeration 4 is the appropriate value. As there are nine base units plus one enumeration field, 40 bits are necessary to represent any SI unit in TSTP. This is the only difference between the representation adopted in the IEEE1451.0 standard, which uses 80 bits.

Force(N)	enum	rad	sr	m	kg	s	A	K	mol	cd
Exponent	0	0	0	1	1	-2	0	0	0	0
Decimal		8	8	10	10	4	8	8	8	8
Strain(m/m)	enum	rad	sr	m	kg	s	A	K	mol	cd
Exponent	1	0	0	1	0	0	0	0	0	0
Decimal		8	8	10	8	8	8	8	8	8
Switches	enum	rad	sr	m	kg	s	A	K	mol	cd
Exponent	4	0	0	0	0	0	0	0	0	0
Decimal		8	8	8	8	8	8	8	8	8

Table IV: Examples of representation of units in TSTP

IV. Position Estimation

In mobile wireless networks, there are distributed algorithms allowing nodes to estimate their location via multilateration given that nodes can estimate their distances to one another and there are anchor nodes that know their own position [18]. There are physical layers which provide information that allows this distance estimation, for example: the Received Signal Strength Indication (RSSI) provided by an IEEE 802.15.4 implementation [17], and the Time Difference of Arrival (TDOA) provided by a UWB transceiver [16]. The location of anchor nodes can be determined equipping a subset of the nodes with GPS receivers, or simply pre-set if a node is not mobile. This makes anchor nodes more expensive and/or difficult to deploy, thus it is generally desirable to reduce their number.

The Heuristic Environmental Consideration Over Positioning System (HECOPS) [18] is one such algorithm. HECOPS enriches multilateration by establishing a ranking system to determine the reliability of each estimated position and using heuristics to mitigate the effects of measurement errors (which can be high, especially in low-cost nodes). Such enrichments reduce the number of necessary anchor nodes, by allowing non-anchor nodes to act as anchors when their confidence is high enough.

In the original implementation of HECOPS, nodes maintain a table of neighbor data containing their alleged coordinates, confidence and RSSI distance measurements. Every node periodically broadcasts its table, and the nodes receiving it can update theirs. With enough data from neighbors, a node can estimate its position and confidence via multilateration.

To cope with the irregular nature of RF signals and RSSI measurements, HECOPS defines a *deviation* heuristic value, which is detected when two anchor nodes estimate their distance to each other via RSSI and then compare it to their actual distance (since their coordinates are known). When a deviation is detected, it is heuristically assumed to affect every node in the same direction that the deviation was found, and nodes can take that into account when estimating their positions. The region affected by a deviation is defined as a circle around one of the anchor nodes, with radius equal to half the distance between the two anchors (Figure 3).

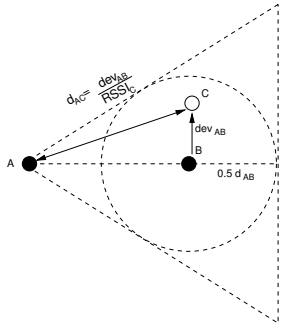

Figure 3: HECOPS deviation heuristic. A and B are anchor nodes that detect a deviation between them, and C is assumed to also be affected by it.

A. Design Rationale

Given that spatial coordinates are embedded in every TSTP message (*Last Hop* fields) and assuming distances can be estimated from information provided by the physical layer (such as RSSI), any node in a sufficiently active region can build the position estimation table passively, only by observing passing messages. *Location Confidence* and *Location Deviation* (as defined by HECOPS [18]) fields are included as well so that nodes can take deviation heuristics into account, as well as distinguish between anchors, confident and unconfident peers. This design brings the advantages of HECOPS mentioned above while introducing zero overhead in terms of new messages.

V. MAC and Routing

In Preamble-Sampling MAC (PS-MAC) protocols [6], sensor nodes spend most of their time in sleep mode and wake up for a short duration called clear-channel-assessment (CCA)

every checking interval (CI) to check whether there is an ongoing transmission on the channel. To avoid deafness, each data packet is preceded by a long preamble, assuring that receivers wake up in time to detect the transmission. PS-MACs enable receiver nodes to keep their radio off most of the time, saving considerable amounts of energy. X-MAC [3] and RB-MAC [1] are examples of PS-MAC protocols.

In Receiver-Based MAC (RB-MAC) [1], senders transmit data without defining a particular node as receiver. The final destination is a pre-defined sink. Preambles consist of micro-frames that contain useful information such as countdown to data transmission, sender distance to sink and payload sequence number. All neighboring nodes within communication range of the sender sense the channel every CI interval, obtain a micro-frame and extract the information; then, only eligible receivers (nodes closer to the sink) go back to sleep and wake up to receive the data at the time indicated by the countdown. Nodes that receive the data without error are relay candidates, and start a back-off timer based on its own distance to the sink and possibly other factors (e.g. remaining energy), which when elapsed will trigger a CCA. The node with the shortest back-off timer will sense no channel activity and proceed to transmit the preamble for CI units of time. Other relay candidates will detect the winner's preamble containing the same sequence number, drop the data and go back to sleep since the packet is already being forwarded. Figure 4 illustrates this process. As a consequence of the way relay candidates are determined, packets are geographically routed to the final destination in a greedy way.

RB-MAC is significantly more resilient to lossy links when compared to sender-based MAC protocols (in which sender nodes keep the addresses of perceived neighbors and define a receiver) [21]. As the number of network nodes increases, RB-MAC requires fewer retransmissions, consequently reducing latency and energy consumption [21].

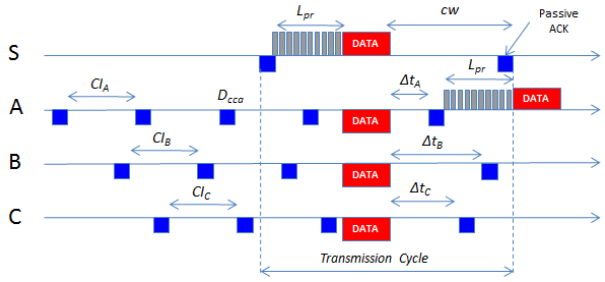

Figure 4: Timeline of RB-MAC.

A. Design Rationale

Because of its efficiency and the design consideration that most messages are sent from a sensor to a single sink, RB-MAC is used as TSTP's MAC and geographic routing mechanism. The preamble is composed of several micro-frames containing useful information (Figure 5):

All Listen: when this bit is set, all nodes that receive the microframe should wake up to receive the corresponding message, regardless of distance. Nodes set this bit when sending messages that are not destined to the sink (e.g. Interest messages).

Count: the number of microframes left until the corresponding message.

Last Hop distance: the distance from the node currently transmitting this message to the destination.

ID: kept the same for the entire lifetime of a message, used by forwarder nodes to determine whether another node is already forwarding the same message. It is a random number, except if the corresponding message was generated as a result of an Interest message with *Response Mode* 1 (Section III). In this case, the ID is calculated as

$$ID = (H(I) + \lfloor (t - t_1)/P \rfloor) \bmod 2^8$$

where $H(I)$ is a network-known hash function of the entire Interest message, t is the current time, t_1 is the region start time of the Interest message, and P is the period indicated in the Interest message. This mechanism has the effect that messages generated from the same "one-responding-sensor" Interest region will have the same ID and, therefore, will be treated as the same message.

Microframe			
Bits: 1	**7**	**32**	**8**
All Listen	Count	Last Hop distance	ID

6 octets

Figure 5: TSTP preamble's Microframe format.

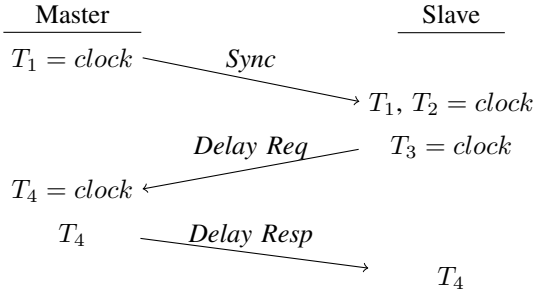

Figure 6: One-step PTP overview. *clock* is the current value of the local clock. Time flows from top to bottom.

VI. Time Synchronization

A variety of different algorithms and strategies for time synchronization exist (e.g. NTP, RBS, FTSP). In particular, we consider the Precision Time Protocol (PTP) [16], which is defined by the IEEE 1588 standard and designed to keep nodes in a Local Area Network synchronized with high precision. In the one-step approach, time is synchronized by the exchange of three messages, as shown in Figure 6. In each message, the Master node sends to the Slave its local timestamp (T_1 and T_4), so the Slave can calculate the network delays in both directions (β and γ) and its clock offset (ϕ) relative to the Master according to Equation 1.

$$\phi = \frac{\beta - \gamma}{2} \begin{cases} \beta = T_2 - T_1 \\ \gamma = T_4 - T_3 \end{cases} \quad (1)$$

PTP can keep a PAN synchronized with sub-millisecond precision [16].

A. Design Rationale

Given that timestamps are embedded in every TSTP message, a variation of PTP enables sensors to synchronize their clocks by taking advantage of passing messages of any type and introducing only one extra message on the network. In TSTP, there are no PTP roles: any node with enough confidence in its clock can play the Master part (and naturally, any node can be a Slave).

Any received message with the *PTP Request* bit (Figure 1) not set can take the role of a PTP *Sync* message; let M_1 be that message. When a sensor needs to recalibrate its clock, upon retransmitting M_1 it will do so with the *PTP Request* and *All Listen* bits set, making it a *Delay Req* message. Upon receiving a message with this bit set, the original sender of M_1 (and only the original sender) transmits back a special Data message M_2 with distance to destination set to 0 in the preamble. Any other node receiving the retransmitted M_1 clears the *PTP Request* bit and routes it as usual. Every node will ignore M_2 (because distance to destination is 0), except the node that requested it, which will extract the message's timestamp to recalibrate its clock, making M_2 a *Delay Resp* message. Figure 7 illustrates this process.

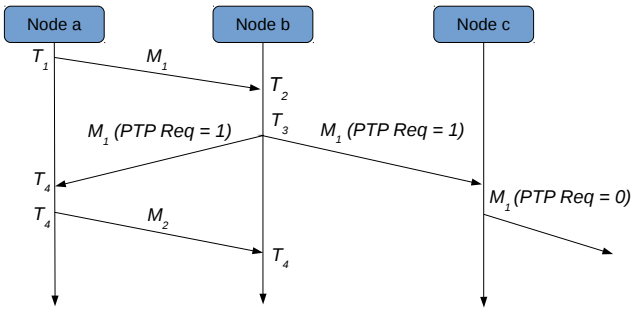

Figure 7: TSTP's time synchronization example.

VII. Security

WSN devices communicate through wireless technology, allowing any radio interface configured at the same frequency band to monitor or participate in communications – which is very convenient for attackers. In order to avoid attacks, a secure infrastructure must provide the principles of *confidentiality*, *authenticity* and *integrity* [22]. The use of Elliptic Curve Cryptography is popular for establishing shared keys because of its good processing/security trade-off, making it suitable for resource-constrained devices. There are many efficient implementations [14] [12] and proposals [10] [8] for security schemes in WSN and IoT, but they usually require either that a third-party act as a Certificate Agent using a secure, out-of-band channel or that sensitive cryptographic information (e.g. a pre-set secret) is pre-loaded in the sensor node.

EPOS' Trust Strategy [7], [19], outlined in Figure 8, contrasts by minimizing the pre-deployment effort, utilizing unique sensor IDs, synchronized clocks and time of deployment as naturally shared information between the sensor and the sink. It also takes advantage of hardware-accelerated AES engines (present in many devices due to being part of the IEEE802.15.4 standard) for encryption, key derivation, and One-Time Passwords (OTP) generation using the Poly1305-AES [2] algorithm.

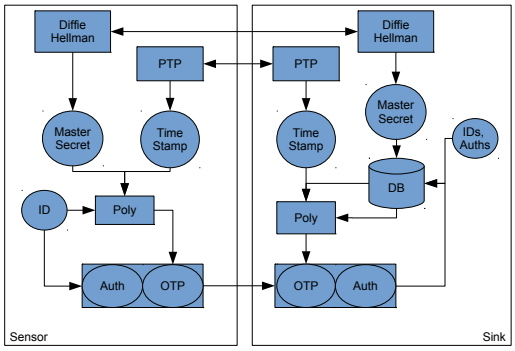

Figure 8: EPOS' key bootstrapping overview.

The protocol assumes synchronized clocks (Section VI). Sensors are assumed to hold a unique identifier (ID) known only by them and the sink, and $Auth$ is calculated independently by both as a one-way hash function of the ID. Each party also holds an Elliptic Curve Diffie-Hellman (ECDH) public-private key pair.

The first step for mutual authentication and key establishment between a sensor and the sink is a regular ECDH agreement, which will result in a shared Master Secret K_{ms}. Afterward, the sensor node calculates a One-Time Password using the Poly1305-AES algorithm, according to Equation 2, where T is the current truncated timestamp. The calculated OTP is then sent along with the $Auth$ code to the sink.

$$OTP = Poly_{1305}(K_{ms}, ID, T) \tag{2}$$

For authentication, the sink fetches on its database the corresponding ID for the received $Auth$ and reproduces the OTP calculation for every pending K_{ms}, until a match is found – in which case the matching ID and K_{ms} are tied together, and the sink has evidence that K_{ms} was shared with the only legitimate holder of that particular ID. The sink proceeds by sending back the *Auth OK* code, which is the ID encrypted under a fresh OTP, so the sensor also has evidence that K_{ms} was shared with the only other legitimate node that knows its ID: the sink. From this step on, secure messages are signed with a MAC and encrypted with AES using a fresh OTP as key, which assures data confidentiality, authenticity, integrity and temporality.

A more in-depth presentation and discussion about this protocol can be found in [19].

A. Design Rationale

EPOS' Trust Strategy was conceived with very similar context and design constraints as TSTP, and so it is incorporated

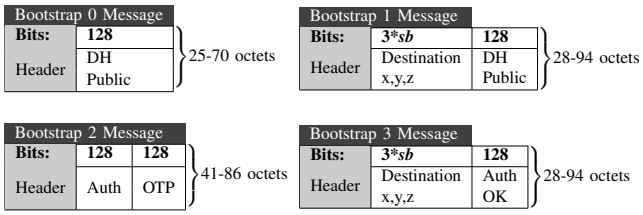

Figure 9: TSTP Security Bootstrap messages.

without change. Figure 9 shows the message formats used for performing the required two-way handshake, in order.

VIII. Bootstrapping

A new sensor joining the network must go through four Bootstrap stages, in order:

Position Determination using the HECOPS protocol (Section IV). A sensor must not send messages until its position has been estimated;

Clock Synchronization: Once position has been determined, the sensor can start routing packets on the network. Each routed packet is sent with the *PTP Request* bit set on the header until the sensor successfully synchronizes its clock (Section VI) with a pre-determined precision;

Security Bootstrapping with EPOS' Trust Strategy (Section VII) enables the sensor and the sink to authenticate mutually and establish a shared key. The bootstrapping sensor deduces the coordinates of the sink from passing TSTP messages it observes;

Report: Finally, the sensor sends a confidential Report message for each quantity it can measure, with a value equal to the smallest quantity it can sense (i.e. its precision) in the payload, as well as the unit it can sense in the Report message header.

IX. Analysis

In previous works, we analyzed the performance of each of the presented blocks [21] [16] [18] [19]. In this section, we bring an analytical discussion on the total latency and number of generated messages caused by the *integration* of all these blocks in the context of TSTP. For comparison, we also analyze these metrics in a design we call Standalone, in which each block is implemented *individually* on top of RB-MAC.

We analyze the case of a new sensor s being deployed in an active region of the network, and we estimate how long it takes for it to finish bootstrapping (Section VIII). We make the following assumptions:

- Radio ranges are uniform, symmetrical and normalized (1 unit area = 1 radio range);
- Sensor nodes are randomly placed in the network area according to a Poisson process, with density ρ nodes per unit area [24];
- Sensor s and all its neighbors will receive a message every $\lambda > CI$ units of time (e.g. both are in the route of a same Interest region with Period λ);
- A sensor contends to forward a message along with all its neighbors. Each relay candidate can win the contention with equal probability.

A. Position Determination

A sensor must receive at least one message from three different confident neighbors to perform trilateration and estimate its position. The expected value of the Poisson process, ρ, gives the expected number of sensors per unit area. The expected number of neighbors $E[N]$ of a sensor is then the expected number of sensors in the radio coverage area ($\pi \times 1$), minus the sensor itself:

$$E[N] = \rho\pi - 1 \tag{3}$$

Let $P[M = m|N = n]$ be the probability that, given that the sensor has n neighbors, the third different HECOPS coordinate will arrive exactly at the m^{th} message. For this event to happen, the first coordinate must arrive with the first message, and might come from any of the n neighbors. The second different coordinate will come from any of $n-1$ other neighbors and must arrive any time in the next $m-1$ messages. Before it does, only the first neighbor might send messages, and after it does, only the first and second neighbors might send messages until the last message, which must come from one of the other $n-2$ neighbors. Therefore:

$$P[M = m|N = n] = \frac{n(n-1)(n-2)\left(\sum_{i=0}^{m-3} 2^i\right)}{n^m} \tag{4}$$

for $n > 2$.

Using this result, the expected waiting time until a sensor receives a message from exactly 3 different neighbors, given that it has n neighbors, is given by:

$$E[H \mid N = n] = \lambda \sum_{m=3}^{\infty} (m \times P[M = m \mid N = n]) \tag{5}$$

The original, standalone HECOPS would take time 3λ to complete, assuming that neighbors alternate when sending location information, and do so with period $\lambda > CI$. It would generate 3 one-hop messages while TSTP generates none in this step. These messages would be generated continuously for maintenance of the position estimation tables.

B. Time Synchronization

Now, sensor s can start forwarding messages and needs to synchronize its clock using the *PTP Request* bit. The sensor needs to win the contention of a message to broadcast a PTP Request. From our assumptions, this will happen with probability $(n+1)^{-1}$ for every message, n being the number of neighbors of s. The probability of s winning contention exactly at the m^{th} message is:

$$P[C = m \mid N = n] = \left(\frac{n}{n+1}\right)^{m-1} \frac{1}{n+1} \tag{6}$$

The expected time until s wins contention is thus:

$$E[C \mid N = n] = \lambda \sum_{m=1}^{\infty} (m \times P[C = m \mid N = n]) \quad (7)$$

After transmitting a PTP Request, which takes CI units of time, the sensor receives a response, which also takes CI. As in the Standalone implementation, the original PTP requires an exchange of 3 new messages with the sink, which would take time $3L(D = D)$ (Section IX-C). This process would happen periodically for every node to maintain the clocks synchronized. TSTP generates only one single-hop message per iteration.

C. Security Bootstrapping and Message Latency

For the Security Bootstrapping step, the same analysis holds for TSTP and a Standalone implementation. The sensor and the sink exchange four messages for key establishment and authentication, and the Report step sends at least one message, yielding a total of 5 messages. Thus, we now estimate the expected time a message takes to arrive at its destination. We can reuse results from GeRaF's analysis [24] if we observe that RB-MAC is similar to the routing strategy presented in that work, with the main differences being that sensors have synchronized duty cycles and can be considered to be always awake, while GeRaF considers that sensors sleep and wake up randomly. The expected number of hops $E[n]$, assuming that the best possible relay (the one closest to the destination) is always selected, is approximated in [24] as a function of the distance to destination D:

$$\frac{D-1}{E[\zeta(D)]} + 1 \leq E[n] \quad (8)$$

where

$$E[\zeta(D)] = 1 - \int_0^1 e^{-MA(D-a,D)/\pi} da \quad (9)$$

is the average advancement towards the destination, $M = d\rho\pi$ is the average number of available relay nodes in the coverage area, and d is the probability that a node will be awake. As noted, in RB-MAC $d = 1$, because nodes have synchronized duty cycles. $A(r, D)$, given in [24], is the area of the intersection between two circles with centers at distance D from each other and radii 1 and r.

It is easy to include a factor p, representing the probability that a message is not corrupted during transmission between two consecutive hops, when calculating the total expected time a message takes to travel a distance D. The probability of retransmission at each hop for RB-MAC is [21]:

$$P_{Ret} = \frac{1}{1-(1-p)^n} \quad (10)$$

being n the number of relay candidates. In our case, $n = \frac{\rho\pi}{2}$. Assuming that at most one retransmission occurs at each hop, average message latency is given as the total expected time taken for a message to travel a distance d:

$$L(D = d) = \left(\frac{d-1}{E[\zeta(d)]} + 1\right) \times (1 + P_{Ret}) \times CI \quad (11)$$

and, finally, the total expected time taken for a sensor deployed at distance D from the sink to be fully functional (synchronized in time and space, authenticated, holding a shared key with the sink and reported) is given as:

$$E[H \mid N = \rho\pi - 1] + E[C \mid N = \rho\pi - 1] + 2CI + 5L(D = D) \quad (12)$$

generating a total of 5 node-sink messages and 1 one-hop message. With standalone implementations of each block, 3 one-hop messages and 8 node-sink messages would be generated for complete bootstrapping, with a total time of:

$$3\lambda + 3L(D = D) + 5L(D = D) \quad (13)$$

Figure 10 shows the time taken by each step of the bootstrapping process as a function of node density for a fixed distance. It can be seen that, for densities up to around 10 nodes per radio range, the integration of the building blocks in TSTP can result in up to 31% shorter time until successful bootstrapping, when compared with standalone implementations of each block. When the density is too high, PTP starts to dominate the total TSTP time because of the assumption that a sensor competes with equal probability against every neighbor to forward a received message. Thus, in this simplified model, the more neighbors a node has, the harder it is to win the contention and issue a PTP Request.

Figure 11 shows how node density and distance to the sink affect RB-MAC's latency. Also, it compares the total time between the Standalone and TSTP designs for the same variation of parameters. Here, it can be seen that for small distances and densities, the Standalone design can be better than TSTP in terms of latency. However, TSTP quickly outperforms the Standalone model when the values of distance and density get higher.

Figure 12 outlines the effects of channel quality on node-sink latency.

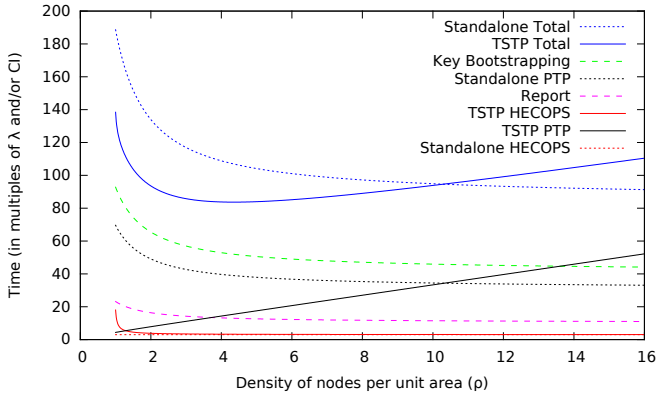

Figure 10: Time taken by each step of bootstrapping as a function of node density. D = 5. Time is a multiple of λ for HECOPS and PTP, and a multiple of CI for Report and Key Bootstrapping. The sums (Total) and representation in the same axis are possible because λ and CI are both set to 1.

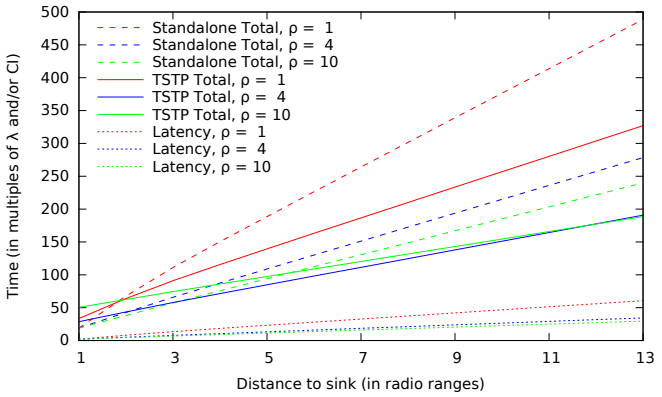

Figure 11: Single message latency from node to sink and total bootstrapping time as a function of distance to the sink, varying node density. Time is a multiple of CI for Latency. The sums (Total) and representation in the same axis are possible because λ and CI are both set to 1.

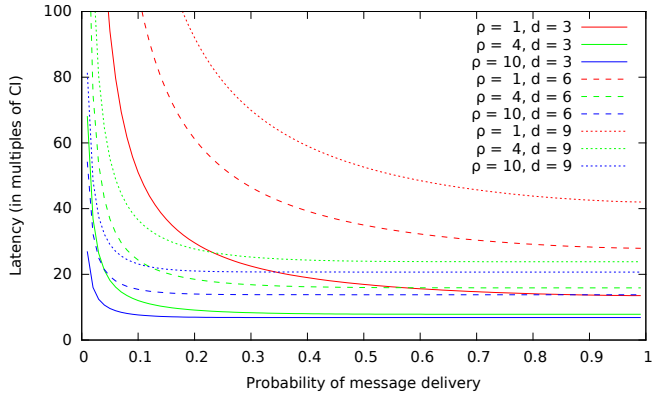

Figure 12: Single message latency from node to sink as a function of probability of successful message delivery, varying node density and distance to the sink.

X. Conclusion

In this paper, an application-oriented communication protocol for WSN was presented, which integrates several blocks in a cross-layer approach in order to efficiently deliver functionality often needed by WSN applications: MAC, spatial localization, geographic routing, time synchronization, security, and an application interface concerned with measuring and communicating physical quantities. An analytical model was developed to predict the total latency caused by two designs: TSTP and Standalone – in which the same blocks are present without integration. It was argued that TSTP can achieve 31% faster sensor bootstrapping time and reduce the number of generated messages from 11 to 6 in the bootstrapping process alone.

References

[1] M.R. Akhavan, T. Watteyne, and A.H. Aghvami. Enhancing the performance of RPL using a Receiver-Based MAC protocol in lossy WSNs. In *IEEE ICT*, pages 191–194, May 2011.

[2] Daniel J. Bernstein. The poly1305-aes message authentication code. In *Proceedings of Fast Software Encryption*, pages 32–49, 2005.

[3] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *ACM SenSys*, 2006.

[4] Jae Young Choi, Hyung Seok Kim, Iljoo Baek, and Wook Hyun Kwon. Cell based energy density aware routing: a new protocol for improving the lifetime of wireless sensor networks. *Computer Communications*, 28(11):1293 – 1302, 2005.

[5] Adam Dunkels, Fredrik Österlind, and Zhitao He. An adaptive communication architecture for wireless sensor networks. In *5th ACM SenSys*, pages 335–349, New York, NY, USA, 2007.

[6] A. El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *IEEE ICC*, pages 3418–3423, 2002.

[7] Antônio Augusto Fröhlich, Alexandre Massayuki Okazaki, Rodrigo Vieira Steiner, Peterson Oliveira, and Jean Everson Martina. A cross-layer approach to trustfulness in the internet of things. In *9th SEUS*, 2013.

[8] Antônio Augusto Fröhlich, Rodrigo Steiner, and Leonardo Maccari Rufino. A Trustful Infrastructure for the Internet of Things based on EPOSMote. In *9th IEEE DASC*, pages 63–68, 2011.

[9] Bo Fu, Yang Xiao, Hongmei Deng, and Hui Zeng. A survey of cross-layer designs in wireless networks. *Communications Surveys Tutorials, IEEE*, 16(1):110–126, First 2014.

[10] Qiang Huang, Johnas Cukier, Hisashi Kobayashi, Bede Liu, and Jinyun Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. In *2nd ACM WSNA*.

[11] IEEE Instrumentation and Measurement Society. 1451.0 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats. online, 2007.

[12] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *2nd SenSys*, pages 162–175. ACM, 2004.

[13] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer. Application-driven cross-layer optimization for video streaming over wireless networks. *Communications Magazine, IEEE*, 44(1):122–130, Jan 2006.

[14] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: A secure sensor network communication architecture. In *6th IPSN*, pages 479 –488, april 2007.

[15] Lucas D.P. Mendes and Joel J.P.C. Rodrigues. A survey on cross-layer solutions for wireless sensor networks. *Journal of Network and Computer Applications*, 34(2):523 – 534, 2011. Efficient and Robust Security and Services of Wireless Mesh Networks.

[16] Peterson Oliveira, Alexandre Massayuki Okazaki, , and Antônio Augusto Fröhlich. Sincronização de Tempo a nível de SO utilizando o protocolo IEEE1588. In *Brazilian Symposium on Computing System Engineering*, 2012.

[17] Rafael Pereira Pires, Lucas Francisco Wanner, and Antônio Augusto Fröhlich. An Efficient Calibration Method for RSSI-based Location Algorithms. In *6th IEEE INDIN*, pages 1183–1188, 2008.

[18] Ricardo Reghelin and Antônio Augusto Fröhlich. A Decentralized Location System for Sensor Networks Using Cooperative Calibration and Heuristics. In *9th ACM/IEEE MSWIM*, pages 139–146, 2006.

[19] Davi Resner and Antônio Augusto Fröhlich. Key establishment and trustful communication for the internet of things. In *4th SENSORNETS*, February 2015. To be published.

[20] V. Srivastava and M. Motani. Cross-layer design: a survey and the road ahead. *Communications Magazine, IEEE*, 43(12):112–119, Dec 2005.

[21] Rodrigo Vieira Steiner, Mohammad Reza Akhavan, Antônio A. Fröhlich, , and A. Hamid Aghvami. Performance evaluation of receiver based mac using configurable framework in wsns. In *IEEE WCNC*, pages 884–889, 2013.

[22] Hui Suo, Jiafu Wan, Caifeng Zou, and Jianqi Liu. Security in the internet of things: A review. In *ICCSEE*, volume 3, pages 648–651, March 2012.

[23] Dogan Yazar and Adam Dunkels. Efficient application integration in ip-based sensor networks. In *1st ACM BuildSys*, pages 43–48, New York, NY, USA, 2009. ACM.

[24] M. Zorzi and R.R. Rao. Geographic random forwarding (geraf) for ad hoc and sensor networks: multihop performance. *Mobile Computing, IEEE Transactions on*, 2(4):337–348, Oct 2003.