TRIDENT: In-field Connectivity Assessment for Wireless Sensor Networks

Timofei Istomin[1], Ramona Marfievici[1], Amy L. Murphy[2], Gian Pietro Picco[1]
[1] University of Trento, Italy — firstname.lastname@unitn.it
[2] Bruno Kessler Foundation, Trento, Italy — murphy@fbk.eu

ABSTRACT

Real-world deployments of wireless sensor networks (WSNs) are notoriously difficult to get right, partly due to the fact that their low-power wireless communication is greatly affected by the characteristics of the target environment. An early *in-field* connectivity assessment of the latter provides fundamental input to the WSN development. Unfortunately, state-of-the-art tools require a secondary networking infrastructure (e.g., USB cables) for gathering experimental data—a luxury one can rarely afford in-field.

We present TRIDENT, a tool expressly designed to support in-field connectivity assessment by relying only on the WSN nodes, without additional infrastructure. The tool is also designed to be *easy to use*, enabling domain experts (instead of WSN experts) to directly configure connectivity assessment towards their specific needs, without requiring coding. The tool is provided in two variants, targeting TMote Sky motes running TinyOS, and Waspmotes running the standard ZigBee stack, covering popular platforms in research and industry, respectively.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement techniques; C.2.1 [**Network Architecture and Design**]: Wireless communication

Keywords

802.15.4, Link quality, Connectivity assessment

1. INTRODUCTION

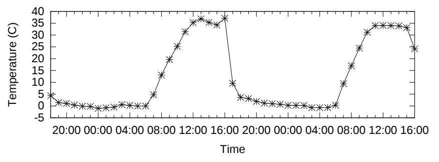
Wireless sensor networks (WSNs) are finding their way into real-world applications, whose successful deployments are increasingly reported. This accrued experience has also evidenced how the task of deploying a WSN entails a number of challenges. One of the most important concerns the low-power wireless communication exploited by WSNs. This type of wireless communication has peculiar characteristics, that have been studied by many researchers; a summary is provided by a recent empirical study [11].

Motivation. Communication in the 2.4 GHz band of IEEE 802.15.4, commonly employed by WSN devices, is also affected by factors strongly dependent on the environment where the WSN is deployed, and hard to capture in the controlled setups typically used in the studies above. These factors include the shape of the deployment site [10], variations in temperature [3] and humidity [14] and their combination in outdoor [8, 15] or industrial [4] scenarios.

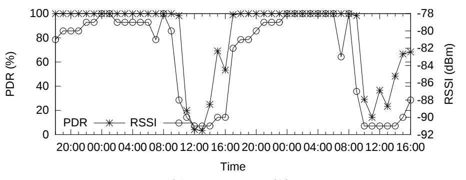
Figure 1 provides a concrete idea of the extent to which communication can be affected by these factors, albeit using a small-scale setup. A single link between two nodes communicating at 0 dBm and placed 40 m apart in an outdoor open field is observed over a 2-day period. Figure 1(a) shows the average temperature inside the plastic boxes holding the motes; Figure 1(b) shows that, for TMote Sky devices, this temperature is negatively correlated with the Received Signal Strength Indicator ($RSSI$) and Packet Delivery Rate (PDR) of the link. A 40°C temperature increase in the box (caused by a mere 11°C increase outside of it) causes a -13 dBm decrease in $RSSI$. This is enough to turn a perfect link ($PDR = 100\%$) into a dead one ($PDR = 0\%$).

Assessing quantitatively the characteristics of low-power wireless links *in-field*, i.e., in the target environment where a WSN must be deployed, is key for several, intertwined goals:

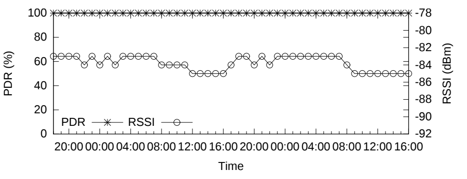
- *supporting the WSN deployment*, by determining where to place the motes to ensure communication among them;
- *informing the selection (or design) of protocols*, to ensure they are well-suited to the target environment;
- *deriving models*, to push the envelope of what can be predicted (or simulated) beforehand.



(a) Temperature.



(b) TMote Sky.



(c) Waspmote.

Figure 1: Effect of temperature on $RSSI$ and PDR.

Unfortunately, the tools supporting connectivity assessment (e.g., SWAT [13], SCALE [7], RadiaLE [2]) were conceived to study the properties of low-power wireless links in controlled environments, e.g., an office testbed. Therefore, they require a secondary, wired networking infrastructure (e.g., USB cables) for gathering experimental data—a luxury one can rarely afford in real-world settings.

Contribution. This paper presents TRIDENT [1], a tool expressly designed to simplify the chore of in-field connectivity assessment. TRIDENT relies only on the WSN nodes whose connectivity needs to be ascertained, without any external infrastructure.

TRIDENT is useful to WSN researchers and practitioners, who may use it towards any of the three aforementioned goals. However, the tool is designed to be easy to use also for domain experts who, being the "users", have precise knowledge of where the WSN nodes should be deployed from the application standpoint, but typically do not have a very deep knowledge about the inner working of the WSN, and definitely do not take part in programming it. The connectivity experiments can be configured easily without requiring any coding, and the data collected with a straightforward procedure. Therefore, TRIDENT empowers domain experts with the ability to assess the connectivity of WSN nodes deployed in-field, without the need for a supporting WSN expert. The goals and requirements we set for TRIDENT are described in Section 2.

TRIDENT covers the entire workflow concerned with connectivity assessment experiments. After the WSN nodes are flashed with the TRIDENT software, appropriate user interfaces enable the operator to (re)configure the experiment settings, discover nodes, and download the results. The latter are stored in a database, simplifying the storage and analysis of the (typically large volumes of) data gathered. The main operation of TRIDENT is described in Section 3, along with the various components of the tool.

We originally developed TRIDENT for the popular research-oriented platform constituted by TMote Sky motes running TinyOS. However, connectivity assessment is relevant also to industry-oriented platforms, for which we chose Waspmote devices running the standard ZigBee stack as a representative. Interestingly, supporting the latter platform is not simply a matter of porting the code from the former; the fact that the ZigBee stack is "closed", unlike the TinyOS one, forced us to find ways to reliably measure the main metric of PDR, which cannot be derived directly otherwise.

Although both platforms are based on radio chips compliant with IEEE 802.15.4, their behavior is very different, as shown again in Figure 1. The same setup and observations we described earlier were also performed for a link between a pair of Waspmote nodes, whose behavior is shown in Figure 1(c). The two experiments were performed in exactly the same conditions, by placing the two pairs of motes side-by-side on different channels. The chart shows that, for Waspmote, temperature is also negatively correlated with $RSSI$, but to a much lesser extent w.r.t. TMote Sky, allowing the PDR to remain at 100%. These sharp differences motivated different implementations of TRIDENT, discussed in Section 4.

Finally, Section 5 ends the paper with brief concluding remarks and an outlook on future work on TRIDENT.

2. REQUIREMENTS

In this section we describe the requirements we set for ourselves in designing TRIDENT, grouped according to their nature.

2.1 Type of Data Collected

Metrics. We want to support in-field collection of several metrics typically used to perform connectivity assessment.

The key metric provided is the *Packet Delivery Rate* (PDR), i.e., the ratio of packets received over those sent. PDR provides a direct assessment of the ability of a link to reliably communicate packets.

A number of metrics are extracted directly from the *radio chip*: Received Signal Strength ($RSSI$), Link Quality Indicator (LQI), and $RSSI$ noise floor, sampled after sending or receiving a packet. These metrics provide insights about physical layer parameters influencing PDR, and their correlation with it. Moreover, noise floor is useful to indirectly determine the presence of interference.

Finally, TRIDENT supports the acquisition, upon packet sending or receiving, of *environmental parameters* (e.g., temperature and humidity) from on-board sensors. These are useful to determine how the environment affects communication, as highlighted by the works mentioned earlier and concretely shown in Figure 1.

Aggregated vs. per-packet samples. The reason for which connectivity assessment is performed determines the nature of the data gathered. If a long-term observation is necessary, the amount of data recorded can rapidly become prohibitive for a resource-constrained platform like TMote Sky. Therefore, the tool should support the ability to store only an aggregate of the metrics collected, computed over a well-defined sequence of packet exchanges.

On the other hand, recording the individual metrics (e.g., $RSSI$ and noise floor) associated with *each* packet provides a fine-grained detail that is necessary in some cases, e.g., when the goal is to ascertain the time-variant characteristics of links with the resolution necessary to inform the design of network protocols.

The choice between aggregated or per-packet samples should therefore be left to the user, balancing the goals of connectivity assessment against the storage limitations of the platform at hand.

Single packets vs. bursts. Another related dimension is the way the channel is observed. Connectivity assessment is often performed by sending *probe* packets with an inter-message interval (IMI) relatively high (e.g., seconds), representative of several WSN applications. Nevertheless, some applications (e.g., recording data from accelerometers [6, 16]) require the transmission of bursts of packets. Moreover, a well-known property of the wireless channel is that the transmission of packets sent with a small-enough IMI [12] is more reliable. The tool should allow the user to choose whether to use single packets or bursts of packets as probes.

2.2 Type of Experiments Supported

Interactive vs. batch. Connectivity assessment may be needed for reasons yielding different requirements, as described earlier.

If the ultimate goal is to support a WSN deployment by helping determine a node placement enabling communication, this is often achieved by performing tests of short duration (e.g., a few minutes). These are useful to quickly evidence which nodes experience low PDR values; this information is used by the operator to relocate nodes and re-assess connectivity with another short test. To effectively support this process, the tool must provide a way to quickly (e.g., graphically) represent the PDR associated with WSN links.

On the other hand, connectivity assessment is often performed also to *characterize* the target environment. This requires long-term observations (e.g., days); the continuous presence of an operator would be impractical. The tool must provide the option to run *automatically* a battery of tests, including different settings, defined by the operator but executed without her involvement.

In our experience, the two modes of operation are often used in conjunction. Indeed, before starting a long-term batch experiment, a short-term interactive one is performed, to make sure that all nodes are functioning properly, and that the baseline connectivity is appropriate to the purpose of the experiment.

Mobile nodes. Mobile WSN applications, e.g., involving nodes placed on humans, animals, robots, or vehicles, are becoming increasingly popular. Therefore, the tool should support experiments where some of the nodes are mobile, to assess the connectivity between these and the fixed nodes. An interesting possibility, partially explored in our previous work [5], is to use mobile nodes as a way to perform a preliminary exploration of the deployment area. As the mobile node moves across the field and exchanges messages with fixed nodes, it samples the connectivity of a high number of locations, cumbersome to explore individually only with fixed nodes.

2.3 Support to Operators

In-field, wireless interaction with nodes. In-field operators must interact with the nodes for various purposes. The primary reason is to retrieve the results of experiments, stored on the nodes taking part in them. Another key operation is the re-tasking of the nodes with a new set of experiments. The operator may also need to interact with the nodes for the sake of monitoring the correct execution of the experiment, e.g., to retrieve statistics about the experiments performed or the battery level. Other useful operations are the ability to put selected nodes (or the entire WSN) in stand-by when they are not involved in an experiment, and wake them up later on.

In principle, some of these operations can be performed by directly accessing the node; for instance, data can be downloaded via USB. However, while this operation is trivial in a lab, it becomes cumbersome when nodes are deployed in-field in a harsh environments, e.g., outdoor in winter, or in places that are not easily accessible. Therefore, all of the interactions with the nodes should be performed over-the-air, by leveraging the wireless channel.

Data storage and processing. Connectivity assessment experiments may generate a huge quantity of data. Handling these as individual files becomes rapidly impractical. Further, the raw data gathered often needs to be processed in an automated way to simplify interpretation. The tool should therefore integrate a database for storing experimental data, enabling structured access and querying, and the definition of stored procedures providing a layer of abstraction in data manipulation and interpretation.

2.4 Non-functional Requirements

No infrastructure. As we already argued in the introduction, this non-functional requirement is a defining one. In-field experiments cannot afford the luxury of a secondary network; the experiment execution must rely only on the WSN nodes under test.

No coding required. Our desire to support domain experts implies that using the tool should *not* require writing even a single line of code. The configuration of experiments should occur entirely via the user interface; at most, domain experts must learn how to flash motes with a pre-canned binary before going in-field.

Ease of use. The logistics of in-field experiments makes them effort-demanding and time-consuming. The situation should not be exacerbated by a complex or cumbersome interaction with the tool. A graphical user interface, providing intuitive support for all the phases of the experiments, is therefore an obvious requirement.

Flexibility. The experiment settings, including number of nodes, their nature and role in the experiment, power and channel settings, number of messages, inter-message interval, number of test repetitions, etc., should be designed in a way that allows users to combine them freely, to explore different portions of the parameter space.

Decoupling from hardware platform. The tool should work on standard nodes without modifications to hardware. Nevertheless, as there are many WSN platforms available, supporting a new one in the tool should be simplified by its software architecture, by confining platform-specific details in well-identified components.

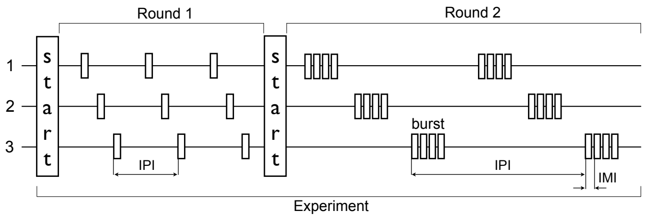


Figure 2: Sample TRIDENT experiment showing two rounds, staggered transmissions using single-packet and burst probes, and per-round synchronization.

3. DESIGN

Next we describe the design of TRIDENT. We begin with a description of the execution of connectivity assessment experiments, then provide an overview of the TRIDENT toolset.

3.1 Experiment Execution

Definitions. An entire TRIDENT experimental campaign is defined as a sequence of *rounds*, as shown in Figure 2. Each round has its own configuration parameters, detailed next, including whether it uses single-packet probes or burst probes, i.e., multiple packets transmitted in rapid sequence. The time between the beginning of two consecutive probes from the same node is the *inter-probe interval (IPI)*. For burst probes, we also define the *inter-message interval (IMI)* as the time between two messages belonging to the same burst. Both IPI and IMI are configurable on a per-round basis.

Probing the links. Connectivity is assessed by probing the communication link with packet transmissions and evaluating the received packets and their properties. Therefore, TRIDENT experiments must define precisely *when* each node transmits and listens.

All nodes behave the same: transmit a probe, pause for the IPI duration, repeat this process a configurable number of times. In between transmissions, nodes can be configured to listen for packets from other nodes. TRIDENT does not duty-cycle the radio during experiments, ensuring that no packets are lost due to the radio state. Moreover, to avoid collisions among simultaneously transmitted packets, which can confound the link evaluation, TRIDENT ensures that only one sender transmits at any given time.

This is achieved by having nodes begin their probe sequence in a staggered way based on their node identifiers. Specifically, the transmit time for the i-th probe of node n is defined as

$$t_{n,i} = t_{start} + nT_{IPI} + iNT_{IPI}$$

where t_{start} is the start time of the round, T_{IPI} is the value of the IPI, n is the node identifier, and N is the overall number of nodes. n and N are setup statically during the experiment design phase.

Staggering transmissions by assigning each nodes its transmit slot, requires the nodes to be time-synchronized. In TRIDENT this is achieved at the beginning of each round, as shown in Figure 2. This synchronization allows the system to compensate for clock drift during a long running experimental campaign.

Master node. Time synchronization is initiated by a special node, called the *master*. The latter acts in general as a coordinator towards the rest of the WSN nodes, as well as the "access point" enabling the operator to change the configuration of experiments. The master has the same binary code of the other nodes; its special role is determined by its identifier, $n = 0$.

The parameters describing the round configuration are also disseminated by the master node during synchronization at the beginning of each round. The option to change parameters in each round allows the interleaving of rounds with different power levels, or interleaving single-packet and bursty rounds, as shown in Figure 2.

Table 1: Per-round configuration parameters and their example values from [8].

Parameter	Example	Parameter	Example
# of nodes (N)	16	# of probes per sender	115
list of senders	$0 \ldots 15$	single-packet probe	yes
list of listeners	$0 \ldots 15$	inter-probe interval	16 s
transmit power	$27(-1$ dBm)	inter-message interval	N/A
probing channel	18	aggregate logging	yes

This choice has multiple consequences. First, only the master node is aware of the experimental campaign, and therefore is the only one affected by changes to the latter. As the master can receive an entire experimental campaign configuration over-the-air, physical access to nodes is not required to change or initiate a campaign. Second, prior to starting a long-running campaign, the operators can interactively run a number of small experiments, each time uploading the round description to the master, instructing the master to initiate the round, then collecting the results. After analysis, another short experiment can be carried out, or the long-running experiment can be initiated. This is all possible because the nodes are experiment-agnostic and the master can be controlled over-the-air.

Per-round configuration parameters. Table 1 shows the per-round configuration parameters available in TRIDENT, communicated by the master before starting a round. We already mentioned some of them, e.g., the overall number of nodes, the role (sender or listener), the type of probe, the values of IPI and IMI. Additional parameters include the radio channel, transmission power, overall number of probes transmitted per node, and number of packets per burst. The logging method must also be defined, choosing between storing information about every packet, or only the average over the entire round, based on the needs of the experiment and the available storage. Finally, rounds and/or entire experiments can be repeated a configurable number of times, to increase statistical relevance.

As for metrics, not shown in the table, PDR, $RSSI$, and (if available) LQI are collected by default. If the platform allows, sender and receiver can acquire per-packet environmental conditions such as noise floor, temperature, and humidity. These values are recorded according to the per-round logging policy.

Interacting with the nodes under experiment. Changing the per-round configuration parameters is not the only option to interact in-field with nodes. Table 2 describes the commands available to the operator. As already mentioned, the operator can upload the description of an entire campaign configuration on the master, which then uses it to orchestrate the various rounds with the appropriate per-round configuration. However, the operator can also start and stop the execution of the experiment; these commands are propagated network-wide, with a mechanism similar to the one used to mark the start of a single round, described in Section 4.

Table 2: In-field commands.

Command	Target	Description
UPLOAD	master	load a campaign configuration
START	network	start the execution of an experiment
STOP	network	stop the execution of an experiment
POLL	1-hop	query battery level
SLEEP	1-hop	place nodes in low-power listening
WAKE-UP	1-hop	remove nodes from low-power listening
DOWNLOAD(n)	node	download logs from selected node
ERASE(n)	node	erase flash of selected node
SNIFF	operator	toggle packet sniffing

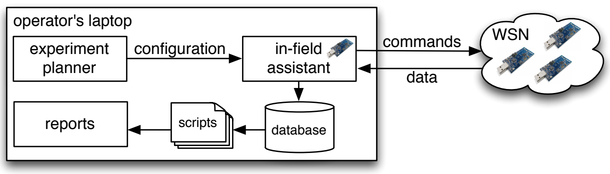


Figure 3: The TRIDENT toolset.

The master node can instruct all nodes to automatically enter a sleep state upon the end of an entire experimental campaign, allowing them to save their remaining battery power. In TinyOS, nodes in this state use the default low-power listening (LPL) MAC with a long sleep interval, currently set to 1 s. By duty-cycling the radio, nodes save battery power, but can be woken up later, e.g., to initiate over-the-air data download or to start a new experimental campaign. Alternately, the operator can also put to (or wake-up from) sleep a subset of the nodes, and query for their battery level.

Other commands enable the operator to download the experiment logs from a selected node, and to erase its flash memory after successful data transfer is verified. Finally, passive packet sniffing can be activated on the node managed by the operator, enabling the latter to ascertain whether all nodes properly started the experiment.

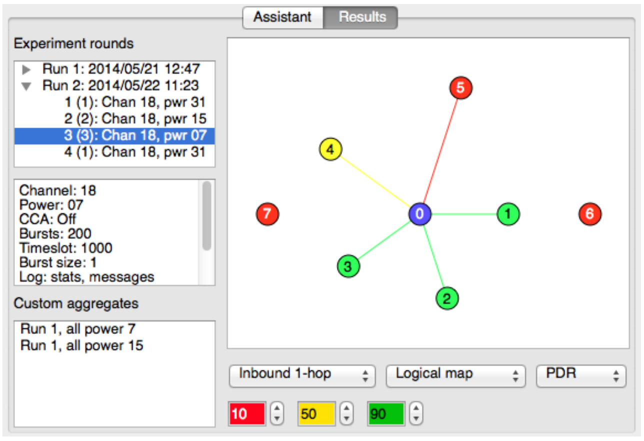
3.2 Toolset Overview

Figure 3 depicts the main structure of TRIDENT. WSN nodes, the subject of the experiment, are programmed with a *platform-specific mote-level runtime* that is experiment-agnostic; its behavior is established by the operator without requiring any coding.

This configuration is performed through the GUI of a dedicated component, the *experiment planner*, which resides on the operator's laptop. The experiment planner essentially enables the operator to quickly and easily define the various details of the experiment, by properly setting the values for the parameters in Table 1. This step does not need to be performed in-field, as the planner enables only the definition and storage of experiments.

The actual upload of experiments to the master, and from there to the rest of the WSN, is instead supported by the *in-field assistant*, which also enables the execution of the other commands in Table 2. Communication with the WSN is enabled by a mote acting as a gateway, connected to the USB port of the operator's laptop.

The in-field assistant provides also a simple visualization of the connectivity map built from available collected traces. An example is shown in Figure 4, visualizing the quality of the inbound links for node 0 according to a intuitive green/yellow/red color-coding, whose semantics in terms of PDR is configurable. This feature



Run 1: 2014/05/21 12:47
Run 2: 2014/05/22 11:23
1 (1): Chan 18, pwr 31
2 (2): Chan 18, pwr 15
3 (3): Chan 18, pwr 07
4 (1): Chan 18, pwr 31

Channel: 18
Power: 07
CCA: Off
Bursts: 200
Timeslot: 1000
Burst size: 1
Log: stats, messages

Run 1, all power 7
Run 1, all power 15

🟥 10 🟨 50 🟩 90

Figure 4: The "results view" of the TRIDENT in-field assistant, showing one-hop inbound connectivity and PDR for node 0.

Table 3: Platform support in TRIDENT.

hardware	TMote Sky	Waspmote + XBee
software	TinyOS	Arduino
PHY layer	802.15.4 (2.4 GHz)	802.15.4 (2.4 GHz)
radio chip	CC2420	EM250
TX power	$-25 \ldots 0$ dBm	$-8 \ldots 2$ dBm
metrics	RSSI, LQI, noise	RSSI
burst probes	yes	no
storage	flash chip, 1 MB	microSD, 2 GB
aggregate logging	per round, on motes	on operator's laptop
sensors	temperature, humidity	—

is particularly useful when TRIDENT is used for a short-term assessment, as it quickly informs the operator about areas with connectivity problems, whose nodes can then be re-arranged. A similar visualization is provided also for mobile nodes; once the in-field assistant is fed with a sequence of locations, it can "replay" the maps, showing to the operator how connectivity evolved due to mobility.

Finally, the *database* and the associated plotting *scripts* simplify the storage of the collected data and its offline analysis. The database contains generic and customizable stored procedures for data manipulation. The set of pre-canned scripts allows the user to quickly plot trends derived from the data collected, e.g., currently including network-wide PDR, cumulative distribution functions of the links w.r.t. their PDR, spatial and temporal variations of the metrics, correlation of PDR with $RSSI$ and LQI.

4. PLATFORM-SPECIFIC DETAILS

TRIDENT currently supports two hardware platforms: TMote Sky and Waspmote. The former directly integrates the CC2420 radio chip, while the latter relies on an extension module for radio communications, in our case the XBee S2 integrating the ZigBee-compliant EM250 system-on-chip. Both transceivers implement the 2.4 GHz IEEE 802.15.4 physical layer.

The software architecture of TRIDENT confines the differences mostly in the platform-specific runtime installed on motes, although a few changes are needed also to parts of the in-field assistant handling communication with the WSN and parsing the logs for visualization. We refer to these platform-specific portions of the software as the *backend*, and summarize the differences in Table 3. The Waspmote variant provides less features than the TMote Sky counterpart, as TinyOS allows low-level access to the radio chip while Waspmote provides only the high-level interface of the ZigBee application support sublayer (APS). These trade-offs are discussed in the rest of the section, along with other implementation details.

Available metrics. The two platforms provide different metrics. TinyOS records $RSSI$ and LQI for each received message, while the XBee radio module reports only $RSSI$. Moreover, unlike ZigBee, the low-level API available to TinyOS applications allows requesting $RSSI$ when the channel is idle to measure the noise floor.

The temperature and humidity sensor of TMote Sky provides important information for our studies of the environmental effects on connectivity. In principle, the same holds for Waspmote, given the wide range of sensors available for this platform. However, we have not yet implemented support for them in TRIDENT, although this does not pose any particular technical problem.

Experiment execution. On both platforms the experiment configuration is installed in the non-volatile storage of the master node. The TMote Sky backend supports uploading the configuration wirelessly or via USB, and stores it in a dedicated flash partition, while the Waspmote backend relies on a configuration file on the SD card.

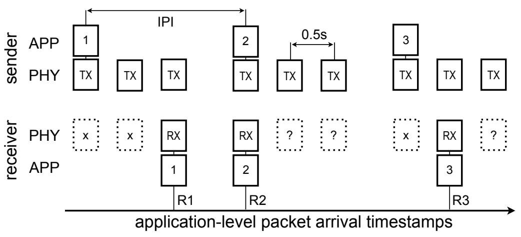


Figure 5: ZigBee transmits each broadcast packet three times.

As described in Section 3, the network is time-synchronized at the beginning of each round to avoid collisions among probes. The backends implement different techniques. In the case of TMote Sky, dissemination relies on a TDMA scheme where each node has its own time slots to repropagate commands, in a way similar to the mechanism outlined for probe transmission in Section 3. The common time reference needed for both the TDMA dissemination phase and to calculate t_{start} is established with TinyOS packet-level time synchronization service [9], yielding sub-millisecond precision.

As ZigBee does not provide such a synchronization service, we rely on the standard multi-hop broadcast feature, basically a network flooding. However, based on the ZigBee Pro feature set [17], broadcast packets are *always* sent 3 times in a row, increasing reliability at the expense of energy consumption. These broadcast packets are separated by a 500 ms interval plus a random delay between 0 and 40 ms. Therefore, in the worst case where the start synchronization message is received only upon third attempt, the time synchronization error goes slightly above 1 s per hop.

To secure a collision-free transmission schedule the IMI should be set long enough to compensate for these synchronization errors and also for the time drift of the nodes. On TMote Sky the synchronization error is negligible; the typical time drift of 100 ppm results in two nodes drifting apart by 36 ms in half an hour. Therefore the use of 250 ms time slots during 30-minute rounds can be considered safe, counting also the time needed for internal processing of the received packets. Instead, on Waspmote the second-per-hop error should be compensated; we achieve this by using 3 s time slots.

Moreover, transmission of probes as one-hop broadcast requires an additional second, again due to the triple transmission performed by the ZigBee network layer. Therefore, it is impossible to send bursts of packets with Waspmote; on TMote Sky, bursts are instead supported with a configurable IMI, set to 20 ms by default.

Another implication of ZigBee compliance is that nodes must join the wireless PAN (personal area network) before sending application data. A multi-hop ZigBee network is built around its coordinator with the standard join procedure, including channel scanning and handshaking, one hop at a time. This process requires up to minutes, depending on the network diameter. In case the channel is changed in between rounds, this affects the minimum interval between them, as the network topology must be rebuilt from scratch.

Determining the link-level PDR. In principle, the value of PDR can be obtained straightforwardly, since the number of transmitted and received packets is known for each link. However, Wasp-

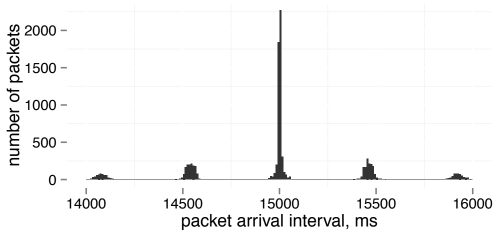


Figure 6: Distribution of packet arrival time.

mote introduce additional complexity due to the triple transmission of broadcast packets mandated by the ZigBee network layer. As shown in Figure 5, since duplicates are automatically filtered at the receiver, the link-level PDR cannot be determined by simply counting the delivered packets at the application layer. Consider two hypothetical experiment runs, one where each probe sent is always received upon first attempt, and one where it is received always upon the third attempt. The link-level PDR would be a meager 33% in the second case, yet the application-level PDR (the only directly measurable) would be 100% in both cases, providing a false representation of the quality of the wireless link.

Nevertheless, we can infer the number of failed attempts by looking at the packet arrival time, based on the fact that the three broadcast transmissions in ZigBee are spaced relatively far apart (500 ms). It is therefore possible to determine, upon receiving a broadcast packet, whether this was the first, second, or third transmission. We confirmed this experimentally: Figure 6 shows the distribution of the packet arrival interval (modulo the nominal IPI) measured at the application level for an intermediate-quality link. For a perfect link, all packets would be received with the same IPI, 15 s in this case; however, this is not the case when packets are lost. Consider an application-level packet i, received on first attempt. If the previous packet, $i - 1$, was received only on second or third attempt, the IPI between the two packets is smaller than 15 s. A similar reasoning holds in case the next packet $i + 1$ is not received upon first attempt, yielding an IPI greater than 15 s. Clearly, the histograms to the left and right of the central one in Figure 6 can be generated also by intermediate combinations, e.g., if i is received upon second attempt and $i + 1$ upon third, the IPI will be around 14.5 s.

Packet loss can be inferred by comparing the application-level packet arrival timestamps (e.g., R_1 and R_2 in Figure 5), provided there is at least one packet in the received sequence known to have arrived upon first attempt. This can be stated certainly when at least one pair of packets (not necessarily consecutive) has either the minimum or maximum possible recorded arrival interval (modulo the IPI), i.e., placed leftmost or rightmost on the histogram of Figure 6. Indeed, this means that the one of the packets was delivered on first attempt and the other on last, as in the case of R_1 and R_2.

In principle, it may happen that no such IPI is recorded during the whole test. In practice, this is unlikely to happen for intermediate-quality links. However, one can still infer the characteristics of the link based on the application-level PDR. If the latter is very good, one can assume that the majority of the packets were received on the first attempt and base the analysis on this fact. For very bad links, it may be impossible to measure the actual PDR precisely when there are just few packets received from the whole sequence.

Storing the experiment data. Due to the storage limitations of TMote Sky, per-message logging can replaced by storing per-round averages of the recorded values, computed on the nodes themselves. Waspmote does not have strict storage capacity limitation; full logs are always stored and the log analysis performed on the operator's laptop. However, we plan to implement on-board log processing also on Waspmote, to reduce the downloading time of large logs.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented TRIDENT, a tool for in-field connectivity assessment. Unlike similar tools in the literature, TRIDENT does not require any communication infrastructure besides the WSN nodes. Moreover, it is designed to be easy to use for domain experts, which can perform their connectivity experiments (e.g., to determine a correct placement of WSN nodes) without coding. TRIDENT supports several configuration parameters and metrics, enabling the investigation of many aspects of low-power communication. Moreover, the operator's interface provides a rich set of commands that simplify in-field management of experiments.

We continuously modify TRIDENT based on the lessons we learn from in-field experience in our deployments and experimental campaigns. Plans for the near future include support for remote control of experiments and in-network collection of their results, therefore significantly reducing the need for operators to be present in-field.

Acknowledgments. The authors wish to thank Matteo Chini and Matteo Ceriotti for their work on early versions of TRIDENT. The research we described is partly funded by the European Institute of Innovation and Technology (EIT) ICT Labs, under activity n. 12149 "From WSN Testbeds to CPS Testbeds" and n. 14280 "I^{3}C—Intelligent Integrated critical Infrastructures for smarter future Cities" .

6. REFERENCES

[1] http://wirelesstrident.sourceforge.net.

[2] N. Baccour, M. Ben Jamaa, D. do Rosario, A. Koubaa, H. Youssef, M. Alves, and L. B. Becker. A testbed for the evaluation of link quality estimators in wireless sensor networks. In *Proc. of the 8th Int. Conf. on Computer Systems and Applications (AICCSA)*, 2010.

[3] K. Bannister, G. Giorgetti, and E. K. S. Gupta. Wireless sensor networking for "hot" applications: Effects of temperature on signal strength, data collection and localization. In *Proc. of the 5th Workshop on Embedded Networked Sensors (HotEmNets)*, 2008.

[4] C. A. Boano, J. Brown, N. Tsiftes, U. Roedig, and T. Voigt. The impact of temperature on outdoor industrial sensornet applications. *IEEE Trans. on Industrial Informatics*, 6(3), 2010.

[5] M. Ceriotti, M. Chini, A. L. Murphy, G. P. Picco, F. Cagnacci, and B. Tolhurst. Motes in the jungle: lessons learned from a short-term wsn deployment in the ecuador cloud forest. In *Proc. of the 4th Int. Workshop on Real-World Wireless Sensor Networks Applications (REALWSN)*, 2010.

[6] M. Ceriotti, L. Mottola, G. Picco, A. Murphy, S. Guna, M. Corrà, M. Pozzi, D. Zonta, and P. Zanon. Monitoring Heritage Buildings with Wireless Sensor Networks: The Torre Aquila Deployment. In *Proc. of the 8th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2009.

[7] A. Cerpa, N. Busek, and D. Estrin. SCALE: A tool for simple connectivity assessment in lossy environments. Technical report, UCLA, 2003.

[8] R. Marfievici, A. L. Murphy, G. P. Picco, F. Cagnacci, and F. Ossi. How Environmental Factors Impact Outdoor Wireless Sensor Networks: A Case Study. In *Proc. of the 10th Int. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, 2013.

[9] M. Maroti and J. Sallai. Packet-level time synchronization (TEP 133). http://www.tinyos.net/tinyos-2.1.0/doc/html/tep133.html, 2008.

[10] L. Mottola, G. P. Picco, M. Ceriotti, S. Guna, and A. L. Murphy. Not all wireless sensor networks are created equal: A comparative study on tunnels. *ACM Trans. on Sensor Networks*, 7(2), 2010.

[11] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low-power wireless. *ACM Trans. on Sensor Networks*, 6(2), 2010.

[12] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. The β-factor: Measuring Wireless Link Burstiness. In *Proc. of the 6th Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, 2008.

[13] K. Srinivasan, M. A. Kazandjieva, M. Jain, E. Kim, and P. Levis. SWAT: enabling wireless network measurements. In *Proc. of the 6th Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, 2008.

[14] J. Thelen and D. Goense. Radio wave propagation in potato fields. In *Proc. of the 1st Workshop on Wireless Network Measurements*, 2005.

[15] H. Wennerstrom, F. Hermans, O. Rensfelt, C. Rohner, and L. Norden. A long-term study of correlations between meteorological conditions and 802.15.4 link performance. In *Proc. of 10th Int. Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2013.

[16] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of the 7th Symp. on Operating Systems Design and Implementation (OSDI)*, 2006.

[17] Zigbee Alliance. ZigBee-2007 PICS and stack profiles, 2008.