Grilo and Heidrich *EURASIP Journal on Wireless Communications and Networking* 2013, **2013**:139
http://jwcn.eurasipjournals.com/content/2013/1/139

EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Routing metrics for cache-based reliable transport in wireless sensor networks

António M Grilo[1*] and Mike Heidrich[2]

Abstract

The Internet of Things (IoT) will bring the pervasive networking of objects, integrating different technologies that will interconnect nodes with heterogeneous capabilities and resources. Although radio-frequency identification will likely play a major role in the IoT, it is not the only one. Wireless sensor networks (WSN) will likely become the paradigm for communication of more powerful nodes in the IoT. The energy and bandwidth constraints of WSNs have motivated the development of new reliable transport protocols in which intermediate nodes are able to cache packets and to retransmit them to the destination in the course of the end-to-end packet recovery process (e.g., pump slowly fetch quickly, reliable multi-segment transport, distributed TCP caching, distributed transport for sensor networks (DTSN)). These protocols use memory resources at the intermediate nodes to achieve lower energy consumption, higher goodput, and lower delay. In a heterogeneous IoT environment, nodes are likely to differ greatly in terms of memory capacity, ranging from almost memoryless tags to more powerful sensors/actuators. Consequently, network nodes will present different eligibilities to support the reliable transport functions, which must be taken into account when setting up routes. The ability to select paths formed by cache-rich nodes becomes essential to maximize network performance and increase energy efficiency. This paper proposes new routing metrics related with the availability of the transport-layer cache at intermediate nodes. A cross-layer protocol architecture is also proposed to support those metrics as well as to integrate legacy metrics such as hop distance and link quality. Simulation results based on the DTSN transport protocol demonstrate that the proposed metrics and cross-layer architecture are essential to leverage the transport layer error recovery mechanism, resulting in increased energy efficiency.

Introduction

The Internet of Things (IoT) will bring the pervasive networking of objects, integrating different technologies that will interconnect nodes with heterogeneous capabilities and resources. Although radio-frequency identification will likely play a major role in the IoT, it is not the only one. Wireless sensor networks (WSN) will likely become the paradigm for communication of more powerful nodes in the IoT [1].

WSNs enable flexible and cost-effective monitoring and control in a broad variety of application areas including - but not limited to - healthcare, assisted living, building and industrial automation, habitat monitoring, security/surveillance, and military applications. The advantages offered by the WSN stem from its infrastructureless, autonomous, and self-organizing principles, coupled with the reduced cost of individual nodes. This reduced cost means that each WSN node may be quite limited in terms of energy, processing, memory, and/or communication resources. In WSNs comprising battery-equipped nodes, energy efficiency is considered the main performance factor since it dictates the maximum operational lifetime of the network. Since the radio transmissions are usually more energy demanding than other WSN node functions, power saving is usually complemented with low-power multi-hop communications. However, low-power communications also lead to lower link reliability. The end-to-end error rate increases with the number of hops, usually requiring error recovery mechanisms to keep it within acceptable bounds. Finding the right balance between energy efficiency and end-to-end reliability is one of the main tasks of the transport layer.

Reliable transport protocols have traditionally been designed to perform end-to-end error control transparently to the intermediate nodes (e.g., transmission control protocol (TCP)). However, the resource constraints in

*Correspondence: antonio.grilo@inov.pt
[1] IST/INESC-ID/INOV, Rua Alves Redol, N° 9, Lisbon 1000-029, Portugal
Full list of author information is available at the end of the article

wireless sensor networks require a different paradigm. Transport protocols such as pump slowly fetch quickly (PSFQ) [2], reliable multi-segment transport (RMST) [3], distributed TCP caching (DTC) [4], and distributed transport for sensor networks (DTSN) [5,6] employ intermediate caching in order to decrease the cost of end-to-end delivery. In these protocols, intermediate nodes are allowed to store and retransmit packets that are missing at the destination, trading-off memory for energy, goodput, and delay. A recent study has analytically characterized the significant improvements that this approach can achieve [7]. These schemes require intermediate nodes to have enough memory to be able to cache effectively. Even in infrastructure Internet, the advantages of intermediate caching were already recognized as a means to improve the performance of TCP [8].

In the heterogeneous IoT, nodes are likely to differ greatly in terms of memory capacity, ranging from almost memoryless tags to more powerful sensors/actuators. Consequently, network nodes will present different eligibilities to support the reliable transport functions, which must be taken into account when setting up routes. The ability to select paths formed by cache-rich nodes becomes essential to maximize network performance and increase energy efficiency.

Obviously, this only applies to cache-based reliable transport sessions. Other traffics can be more efficiently transmitted by following routes built with different metrics. Such dependence between the transport requirements and routing naturally brings the need for a cross-layer architecture that closely couples those functions. The concept of a cross-layer architecture in WSNs is not new. While protocol layer independence facilitates software development and reutilization, it is widely accepted that a cross-layer architecture can significantly increase the energy efficiency and quality of service (QoS) of the WSN.

This paper proposes new routing metrics related with the availability of transport-layer cache at intermediate nodes. A cross-layer protocol architecture is also proposed to support those metrics as well as to integrate legacy metrics such as hop distance and link quality. In this architecture, a 'soft-state' Session Manager is the entity that effectively assigns the routes as well as the respective cache resources to individual transport sessions[a], based on hints provided by the network layer. Simulation results using the DTSN transport protocol demonstrate that the proposed metrics and cross-layer architecture are essential to leverage the transport layer error recovery mechanism, resulting in increased energy efficiency.

As far as the authors know, this is the first time that a cache-based routing metric to support reliable transport in WSNs is proposed.

The rest of this paper is organized as follows. The next section presents the problem statement. The relevant related work on WSN transport protocols with intermediate node caching is introduced next. The proposed cache-related metrics are then presented, followed by the proposed cross-layer architecture, which is in turn followed by the simulation results. The main conclusions of the paper are then presented.

Problem statement

The objective of the mechanisms proposed in this paper is to maximize the energy efficiency by minimizing the cost of guaranteed delivery from sensor nodes to sink nodes in a WSN, defining *cost of transmission* as the expected transmission count (ETX) [9], which is the expected number of frame transmissions required to successfully transmit an application data packet from the source to the destination. The ETX is a suitable performance metric since it is simultaneously related with goodput, delay, and energy consumption. The estimates for the latter can be obtained based on the knowledge of independent variables such as the physical data rate, mean frame size, and (in the case of energy consumption) the energy consumed per transmitted/received bit.

It is assumed that the target WSN application is characterized by unicast data transmission between sensor and/or aggregator nodes and one or more sink nodes (and/or actuator). It is also assumed that the application requires 100% guaranteed delivery while being delay-tolerant. Such requirements can be found, for example, in WSN scenarios in which data are recorded at the sensor or aggregator nodes during some time period, which are being transferred at the end of that period or retrieved by a *mule*[b]. Data gathering finishes when all the stored data blocks have been transmitted. Another possible scenario is the transmission of a snapshot obtained by an imaging sensor (assuming that full reliability is required). In both cases, the stored data block should be transmitted as reliably and energy efficiently as possible in order to maximize the lifetime of the WSN.

Such application characteristics are considered in [7], where the authors evaluate the impact that different cache partitioning policies[c] may have on overall transmission cost while assuming that underlying routing decisions are independently taken based on some criteria (e.g., hop count, link quality, energy availability). The approach followed in this paper differs from the one above since it reverses the degrees of freedom, leading to the following problem statement: *Assuming that at each node the cache is uniformly divided among all the sessions that cross it, which route should be assigned to each of the sessions that are currently active in the WSN in order to minimize the transmission cost?*

We are aware of the limitations of this approach, since a complete solution should be able to integrate non-uniform cache partitioning policies (e.g., based on hop distance and link quality as in [7]) with flexible routing decisions, considering both as degrees of freedom of the optimization solution space. Since the problem becomes significantly more complex, this is left for future work. The present paper should be regarded as an intermediate step to achieve that goal.

It should be pointed out that the problem addressed in this paper bears some similarity with the load balancing problem. In order to maximize overall cache utilization (CU) in the network, sessions should be routed through cache-rich paths, eventually using multipath routing. However, the caching mechanism is implemented at the transport layer, whereas load balancing techniques are usually implemented at the network layer, leading to different cross-layer implications. Although the integration between load balancing and cache balancing is an interesting research topic, it is out of scope of this paper.

There are currently several WSN transport protocols that could be used to support the kind of application scenario described in the beginning of this section. Given the past experience of the authors with DTSN and the ready availability of implementations, it was selected as the transport protocol for this work. Nevertheless, the proposed cache-based routing metrics and cross-layer architecture are almost totally independent from the details of the DTSN transport protocol and can be easily adapted to RMST, DTC, or even to the TCP caching extensions proposed in [8].

Related work

Since the proposed routing metrics require the close coupling between the transport and network layers, this section shall focus on relevant WSN protocols at those layers. Due to the current lack of research on cross-layer architectures integrating transport-related routing metrics, the most relevant transport and routing protocols shall be separately analyzed.

Transport protocols

Starting with the reliable transport protocols, this section shall only cover those that employ intermediate caching to minimize end-to-end retransmissions. The interested reader is referred to [10] for a more extensive survey on WSN transport protocols.

RMST [3] offers two simple services: data segmentation/reassembly and guaranteed delivery using a negative acknowledgement (NACK)-based automatic repeat-request (ARQ) mechanism. RMST can operate end-to-end or in a store-and-forward mode where intermediate nodes recover all the fragments of a block before relaying them forward. Since in this case the data blocks are reconstructed at each hop, RMST requires significant memory resources at individual nodes.

PSFQ [2] is a protocol primarily designed for downstream multicast dynamic code update, though it can also be configured for unicast communication. Regarding intermediate caching, data are reconstructed at each hop, just like in RMST (see above). While this makes sense for dynamic code update (i.e., each node must get all the executable code fragments), it can be very limiting for other applications since it poses significant requirements on node storage capabilities.

DTC [4] enhances TCP in order to make it more efficient in WSNs. Its guaranteed delivery service is similar to that of DTSN. DTC improves the transmission efficiency by compressing the headers and by using cache at selected intermediate nodes. DTC is fully compatible with TCP, leaving the endpoints of communication unchanged - it only requires changes in the logic of intermediate nodes. Although the paper only considers caching a single segment, this significantly improves the efficiency of end-to-end delivery, minimizing the energy spent with retransmissions. The caching mechanism may also be easily extended to use more than one segment per node. The authors also propose to use the TCP selective acknowledgment (SACK) option in order to optimize the use of the cache. In this proposal, the SACK block is used to carry information about segments in cache, allowing nodes farther from the destination to free their cache entries in case a node that is closer already has the segment in cache. Some improvements have been proposed to the original DTC protocol. In [11], a multipath scheme between sender and receiver allows the TCP session to be rerouted when intermediate nodes fail. In [12], the focus is rather on the medium access control (MAC)-transport cross-layer optimization, assuming a carrier sense multiple access with collision avoidance (CSMA/CA) MAC with ARQ. While DTC always tries to cache more recent segments, the new scheme allows to maintain an older segment already in cache with 50% probability. The paper also proposes an alternative method to estimate the round-trip time.

Wisden [13] is a sensor-to-sink wireless structural data acquisition system that incorporates data synchronization and data compression algorithms besides reliable transport. The latter can be performed hop-by-hop, in which case intermediate nodes keep lists of missing packets and cache recently forwarded packets. Missing packets are requested from the previous node using a NACK mechanism. Since the amount of cache at each node is limited, there may not be enough space to store all the packets in cache. Hence, an end-to-end recovery scheme is used to ensure that all packets are delivered. The sink node keeps track of missing packets and requests them using the same NACK scheme used for hop-by-hop. It is assumed that

the sink node has no memory constraints, which is able to store all out-of-order packets until the gaps are filled.

Rate-controlled reliable transport (RCRT) [14], developed later by the same research lab, is another reliable sensor-to-sink transport protocol, which includes congestion control and explicit rate adaptation functions. Although RCRT defines a purely end-to-end NACK-based retransmission mechanism, [15] proposes a hop-by-hop variant, where each node along the path keeps a packet cache and a list of missing packets per flow. Whenever missing packets are detected, the respective sequence numbers are included in a NACK packet, which is sent to the previous node. In case an intermediate node receives a NACK and has any of the requested packets in cache, it retransmits them towards the sink and forwards the updated NACK towards the source. Cached packets are deleted once they are confirmed by a positive ACK.

Reliable transport with memory consideration (RTMC) [16] integrates hop-by-hop error recovery with flow control to prevent cache overflow at intermediate nodes. A connection-oriented approach is followed since cache is explicitly reserved per flow upon reception of a special control packet in the beginning of the session, which is also explicitly freed at the end of the session. Each node processes data packets coming from both the previous node and the next node. Packets from the previous node are stored in cache and transmitted to the next node as soon as possible. Packets relayed by the next node are used as implicit ACKs to free space in the local cache as well as to know the amount of cache that is currently free at the next node. A node will only relay data packets when there is free room at the next node. Packets are explicitly requested from the previous node when detected as missing or when the cache is empty. No end-to-end reliability scheme is considered.

Actor-actor reliable transport protocol (A^2RT) [17] seeks to provide realtime and reliable data delivery between partitioned resource-rich actor nodes whose connectivity is bridged by resource-constrained sensor nodes. Caching of packets at actor nodes improves the performance, compared with end-to-end reliability. Cross-layer interaction with the routing protocol allows a transport wrapper entity to divide the end-to-end path into multiple segments between partitioned actor nodes, minimizing the number of sensor nodes that are used to bridge two consecutive actor nodes. In each segment, a reliable transport protocol is used to guarantee actor-to-actor delivery until the destination is reached. An end-to-end wrapper session is maintained between the source and the final destination in order to assure end-to-end delivery.

In [7], the authors propose a Markov model to evaluate the impact of caching in end-to-end WSN reliable transport. Several cache-partitioning policies are tested under different network conditions.

TCP segment caching at intermediate network nodes in infrastructure Internet Protocol (IP) networks was recently proposed in an Internet Research Task Force draft [8]. Although this TCP extension is not being proposed in the context of the IoT, this proposal will likely influence future developments in the context of the IPv6 over Low-Power WPAN (6LoWPAN) working group (http://datatracker.ietf.org/wg/6lowpan/).

The DTSN protocol is more directly related to the present paper and thus shall be described more extensively.

The basic DTSN specification [5,6] was thought for critical data transfer requiring end-to-end full reliability in the fashion of TCP. However, for the sake of improving energy efficiency in WSNs, DTSN employs selective repeat ARQ (SR-ARQ) with NACKs. Positive ACK packets are also used to prevent the situations where the complete message or its last packet is lost (which cannot be detected solely based on NACKs). Both NACKs and ACKs are to be sent by the receiver only upon request by the sender (explicit acknowledgment request (EAR)), which can be piggy-backed in data packets.

In DTSN, a session is a source/destination relationship univocally identified by the tuple <*source address, destination address, application identifier, session number*>, designated as the session identifier. Within a session, packets are sequentially numbered. The acknowledgment Window (AW) is defined as the number of packets that the source transmits before generating an EAR. The output buffer at the sender works as a sliding window, which can span more than one AW. Its size depends on the specific scenario, namely on the memory constraints of individual nodes.

In order to minimize end-to-end retransmissions, the intermediate nodes are able to cache a number of packets. Upon interception of a NACK, in case they find any of the requested packets in cache, they retransmit those packets to the receiver. After deleting the respective sequence numbers from the NACK, the intermediate node allows the modified NACK packet to resume its walk towards the source.

Any cached packets whose reception by the destination is confirmed by the destination are purged from the cache, releasing memory space to cache ensuing packets.

All the above approaches, including DTSN, assume that the routes are defined beforehand by the routing layer, with no participation of the transport layer. In contrast with these approaches, our paper focuses on routing metrics that lead to a more efficient operation of the transport protocol, hence implying cross-layer interactions between those layers.

Table 1 summarizes the main characteristics of the caching-based transport protocols mentioned above.

Table 1 Main characteristics of caching-based transport protocols

Protocol	Direction	Xcast	ARQ	Packet recovery	Symmetric reverse route
RMST	Any	Ucast	Selective repeat w/ NACK	Optional, store-and-forward	Desirable
PSFQ	Sink-to-sensor	Bcast	Selective repeat w/ NACK	Store-and-forward	N/A
DTC	Any	Ucast	TCP SACK	Probabilistic caching + E2E	Desirable
Wisden	Sensor-to-sink	Ucast	Selective repeat w/ NACK	Probabilistic caching + E2E	Desirable
RCRT w/ cache [15]	Sensor-to-sink	Ucast	Selective repeat w/ NACK + ACK	Probabilistic caching + E2E	Desirable
RTMC	Any	Ucast	Selective repeat w/ NACK + implicit ACK	Store-and-forward	Mandatory
A^2RT	Actor-to-actor	Ucast	Depends on selected transport	Store-and-forward + E2E	Desirable
DTSN	Any	Ucast	Selective repeat w/ NACK + ACK	Probabilistic caching + E2E	Desirable

Routing protocols

The specific requirements of WSNs brought the need for new routing protocols that are more energy efficient than those employed in mobile ad-hoc networks (MANETs). Although data-centric routing protocols such as directed diffusion [18] can be adequate in several WSN scenarios, legacy node-centric routing still has extensive applicability in WSNs. In fact, some routing protocols employed in WSNs are simply WSN-optimized versions of MANET routing protocols (e.g., a modification of destination-sequenced distance vector (DSDV) [19] was used in the WSNs described in [20,21]).

The IPv6 routing protocol for low power and lossy networks (RPL) [22] is a promising routing protocol in the context of the IoT, which was recently specified by the IETF Routing Over Low Power and Lossy (ROLL) Networks Working Group. RPL is essentially node-centric and allows the establishment of and supports the formation of directed acyclic graphs, also based on a distance-vector paradigm. It separates packet processing and forwarding from the routing optimization objective. RPL is flexible enough to support different combinations of routing metrics [23], allowing the definition of optimization objectives, such as minimizing energy, minimizing latency, and maximizing throughput. However, no metrics were included to deal with memory or cache availability (only a generic node overload bit flag was defined).

In this paper, a variant of DSDV is employed at the network layer in order to simplify the implementation, although the RPL protocol would be the most likely candidate to integrate the proposed cross-layer architecture. The proposed cache-related routing metrics are integrated in this DSDV implementation.

Transport cache management and cache-based route metrics

The cache module of the transport layer is of the utmost importance to increase the efficiency of the retransmission-based packet recovery mechanism [7]. At each intermediate node, the total cache size C_n is divided among the different sessions that cross the node. Let $\omega_i^n \geq 0$ be a weight related to the fraction of cache at node n that is assigned to session i. The actual fraction of cache for session i is given by the normalized weight $\rho_i^n = \frac{\omega_i^n}{\sum_j \omega_j^n}$. The number of packet slots used by session i or cache utilization of session i (CU_i) is then given by the following expression:

$$\mathrm{CU}_i = \lfloor \rho_i^n \times C_n \rfloor = \lfloor \frac{\omega_i^n}{\sum_j \omega_j^n} \times C_n \rfloor. \quad (1)$$

The weights ω_i^n can be assigned based on the several factors, such as priority of the session, or they can be the same for all sessions. Possible cache partitioning policies are analyzed extensively in [7]. There, it is also assumed that the cache of each session follows a strict first-in-first-out policy for simplicity. In fact, old packets may be already cached ahead in the path, and so it is preferable to cache more recent packets.

The cache mechanism can only be used effectively if the nodes that form the route have enough memory resources to support it. In a heterogeneous IoT environment, one

is expected to find a huge disparity between the memory capabilities of individual nodes, ranging from virtually memoryless tags to more powerful nodes such as smartphones and nodes embedded in appliances and other equipments. As such, the choice of the routes used by transport sessions must be carefully chosen as to include as much as possible nodes that coincide on both of the following desirable characteristics:

- Significant memory capacity assigned to transport cache functions
- Few transport sessions sharing the cache, which means that a larger portion of cache can be assigned to each transport session

Intermediate node caching is only a means to achieve the main goal, which is to route data traffic through reliable energy-efficient paths. The better the quality of the links that constitute the route, the lesser the impact of the caching mechanism on overall performance. Considering that the quality of links is approximately even between different routes, caching may become important as a criterium to break ties. The routing metrics proposed in this paper consist of the following (the lengths used in the current implementation are shown between chevrons):

- *Hop distance to the destination (H) <1 octet>*. This is the classical hop distance metric. The value of this field is incremented at each hop.
- *Accumulated link quality (AccLQ) <4 octets>*. Although intermediate node caching seeks to overcome the effects of high link error rates, it is demonstrated in [7] that it is usually better to choose paths formed by good quality links, if these are available. This metric provides an estimate of the overall quality of the links in a path. It consists of the product between the link quality estimates of intermediate links (considering both directions of each link) from the source to the destination. If node m is located $i-1$ hops away from the destination and $\mathrm{LQ}_{x,y}$ represents the link quality estimate from node x to node y, then at node n located i hops away from the destination, this field is updated as follows:

$$\mathrm{AccLQ}_i = \mathrm{LQ}_{n,m} \times \mathrm{LQ}_{m,n} \times \mathrm{AccLQ}_{i-1}. \quad (2)$$

  This assumes that the link quality estimates are calculated and locally broadcasted according to a procedure such as the one described in the section 'MAC layer'.
- *Minimum link quality (MinLQ) <4 octets>*. Optional metric representing the minimum between the link quality estimates (also considering both link directions) between the present node and the destination. It allows the definition of minimum standards for the constructed paths.
- *Total weight (TW) <4 octets>*. Average sum of cache weights, $\sum_j \omega_j^n$, at the intermediate nodes along the path to the destination, where j represents the local flow index at the node. At node n located i hops away from the destination, this field is updated as follows:

$$\mathrm{TW}_i = \lceil [(i-1) * \mathrm{TW}_{i-1} + \sum_j \omega_j^n] / i \rceil. \quad (3)$$

- *Average cache size (CS) <4 octets>*. This field carries the average C_n calculated from the current node to the destination. At node n located i hops away from the destination, this field is updated as follows:

$$\mathrm{CS}_i = \lceil [(i-1) * \mathrm{CS}_{i-1} + C_n] / i \rceil. \quad (4)$$

The link quality product allows an estimate of path reliability, which indirectly incorporates the hop distance. The minimum link quality allows the identification of paths with link quality bottlenecks as well as to place minimum bounds on the acceptable link quality.

The last two metrics were not proposed before in the literature. They allow each node to calculate the average CU_i along the path to the destination.

Lexical metric composition [24] was used, with the routing metrics being evaluated in the following order: AccLQ, H, CU_i.

Algorithm 1 compares two routes based on the respective metrics. When two routes present similar AccLQ, the best performance is provided by the shortest route and/or the one with greater CU_i. A threshold LQTOLERANCE ($0 < \mathrm{LQTOLERANCE} < 1.0$) is used to establish the minimum AccLQ difference required to consider one route better than the other. This makes the AccLQ metric not

Algorithm 1 *CompareRoutes*($m1, m2$): Function that compares two routes based on the respective routing metrics

Require: *RoutingMetrics* $m1$, *RoutingMetrics* $m2$

```
int result = 0;
if m2.AccLQ > m1.AccLQ × (1.0 + LQTOLERANCE) then
    result = 1;
else if m1.AccLQ × (1.0 − LQTOLERANCE) < m2.AccLQ < m1.AccLQ × (1.0 + LQTOLERANCE) then
    if m2.H < m1.H then
        result = 1;
    else if m2.H = m1.H then
        if m2.CU > m1.CU then
            result = 1;
        end if
    end if
end if
```

strictly monotonic, which can lead to loops [d]. Loops are avoided by placing the H metric in second place. Finally, CU_i distinguishes between routes with similar AccLQ and equal H.

The following section presents a cross-layer protocol architecture that uses these metrics to select the best routes for sessions that require an end-to-end reliable transport.

Cross-layer architecture for transport cache optimization

The fact that the caching mechanism operates at the transport layer on a per-session basis leads to inefficient routing decisions when these are independently made by the network layer. Since the caching mechanism requires NACK and/or ACK packets to travel through the reverse path of data, if these routes are not symmetric or are changed too frequently, the transport protocol will end up operating end-to-end most of the time. On the other hand, the scalability may be a problem if the routing protocol is used to find routes on a per-session basis instead of on a per-source-destination-pair basis. The protocol architecture proposed in this paper provides a compromise between the previous solutions, dividing the burden of routing decisions among the transport and network layers (see Figure 1).

This protocol architecture comprises three main components, which run at each WSN node:

1. *MAC layer*. Besides providing per-link reliability based on some form of ARQ, the proposed architecture assumes that the MAC layer keeps track of the quality of the links between a given node and its neighbors. This information is provided to the Session Manager and is used by the route selection algorithm.
2. *Network layer*. This component is responsible for the end-to-end forwarding of packets, though it does not guarantee packet delivery. The routing protocol tries to find the best route towards destination nodes, taking into account the routing metrics provided by the Session Manager. In the proposed architecture, the paths found by the routing protocol may be simply used as hints by the Session Manager. The forwarding algorithm receives the next hop information from the transport layer together with the respective packet. In case the transport layer does not provide the identification of the next hop, the forwarding algorithm requests a default route from the routing module.
3. *Transport layer*. This component is responsible for guaranteeing end-to-end data delivery, resorting to the retransmission of lost packets whenever required. The cache module is an optional component, which allows intermediate nodes to retransmit missing data packets, avoiding costly end-to-end retransmissions from the source. Each sent, received, or intercepted packet is provided to the Session Manager to be analyzed. On the other hand, when sending or forwarding a packet, the transport layer obtains the address of the next hop from the Session Manager, passing this information to the network layer.
4. *Session Manager*. This component is responsible for making per-session routing decisions, guaranteeing that per-session routes are stable during a lower-bounded time interval. The symmetry between data and NACK routes is also guaranteed[e]. Active sessions together with the respective cache assignments are set up or dropped by the Packet Analyzer based on a soft-state scheme. Routing metrics are kept for each session, and the data on local cache availability are provided to the routing protocol. The Session Manager is the cross-layer component of the architecture, integrating functions that traditionally belong to the transport layer and the network layer.

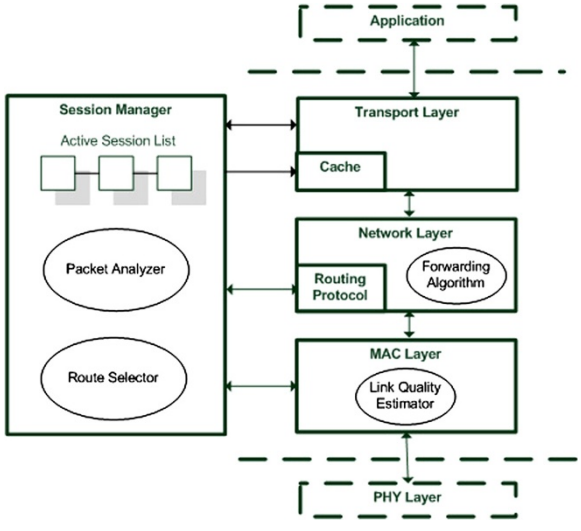

Figure 1 Proposed cross-layer architecture.

Each of these components is described in more detail in the following subsections.

MAC layer

The MAC layer is responsible for providing per-link reliability as well as for providing link quality estimates to the Session Manager for route evaluation and selection.

It is considered that the MAC layer includes an ARQ mechanism, which attempts to transmit each data frame until a MAC layer ACK is received or the maximum number of retries is exceeded. The resultant packet error rate

(PER) will be lower than the frame error rate (FER) measured at the physical layer: for a given FER value, a rough estimate of the PER is $\text{PER} = 1 - \text{FER}^{rl}$, where *rl* is the MAC retry limit[f].

Several link quality estimation schemes are proposed in the literature. In [25], short-term FER estimates are derived from measurements of the signal-to-interference-plus-noise ratio (SINR). In [26], an exponential moving average of the SINR is taken as the link quality estimate. Another possibility is to use the ARQ mechanism to estimate the FER, calculating an exponential moving average over the number of retries required to successfully transmit a data frame. In our paper, we abstract the link quality estimation procedure, assuming that the MAC layer at each node is able to provide acceptably accurate FER estimates of the links from its neighbors to itself, for example, by means of an exponential moving average over individual FER values. The FER could be calculated as a function of the SINR measured from received or overheard packets (e.g., periodic routing updates), or it could be based on a retry sequence number piggybacked in the MAC header. In this paper, the link quality is defined as $1 - \text{PER}$, with the PER estimate being derived fom the FER estimate (see above). Received link quality estimates are broadcasted to the neighbors, piggybacked in routing control packets in order to minimize the overhead. In this way, all nodes know the estimated link quality in both communication directions.

Network layer

In the proposed architecture, the network layer is still the routing entity by default. However, when the Session Manager is active, the routes found by the network layer are used only as hints, with the final routing decision being taken by the Session Manager itself. It is also the Session Manager that defines the routing metrics, provides the respective information element to the network layer, and implements route evaluation. Consequently, the Session Manager can change the routing policy transparently to the network layer. When the Session Manager is not present or active, or when the transport sessions do not present special QoS requirements (e.g., guaranteed delivery), the network layer falls back to its default set of routing metrics (e.g., hop distance).

Preliminary simulations performed by the authors have shown that reactive routing protocols such as AODV and DSR, which resort to flooding of routing requests, have very poor performance under packet error rates above 5%, mainly due to the low reliability MAC layer broadcast (this issue is well known from the literature [27]). This led the authors to choose a proactive routing protocol for architecture evaluation. As already said, in the current implementation, a variant of DSDV [19] is used as the base protocol, but other proactive routing protocols, such as RPL [22], could be also used. The main adaptations introduced in DSDV were the following:

1. Only potential destinations of data (e.g., sink nodes) are allowed to generate routing advertisements. This modification aims to reduce the size of the routing tables that are periodically exchanged between neighbor nodes.
2. An optional route metric information element can be added as an extension to the DSDV header and transported transparently by the routing protocol.
3. The evaluation of routes is performed by the Session Manager. Otherwise, DSDV resorts to its default hop distance metric.
4. The destination nodes update the sequence number of their routing updates in an *epoch* basis, where an *epoch* can span several routing update periods. In this way, the reliability of broadcast routing updates is increased, assuming that the topology is usually stable within epochs. The risk of loops due to a sudden change of topology is restricted to the limited time window of an epoch.
5. Upon request from the transport protocol, all its packets are intercepted and passed to it, even if the current node is not the final destination. This functionality is essential for the implementation of the transport cache mechanism.
6. Link quality estimates obtained from the MAC are piggybacked in advertisement messages.

These modifications make DSDV more energy efficient and reliable in WSNs featuring high packet error rates and provide an elegant and transparent interfacing between the network layer and the Session Manager.

Transport layer

This section only presents the higher level functionality that is directly connected to the cross-layer architecture. The protocol-specific aspects are already described in the 'Related work' section for the case of DTSN, while the general aspects of cache partitioning are already described in the 'Transport cache management and cache-based route metrics' section. In the proposed architecture, the cache partitioning policy is defined by the Session Manager, which parameterizes the cache module, assigning the weights ω_i^n.

Since the performance of the transport protocol increases with the cache utilization, the latter must be maximized while respecting the cache partitioning policy. One way to achieve this objective is to allow each session to use the cache belonging to other sessions, but which is currently not being used. Algorithm 2 decides if a given packet *p* will be cached and, in some cases, on which packet to replace.

Algorithm 2 Packet caching procedure

Require: *Packet p*, *Cache C*

```
1:  SessionCache s, x;
2:  Packet paux;
3:  if C.maxSize > 0 then
4:    if C.inUse = C.maxSize then
5:      s = x ⊂ C | (x.inUse − x.maxSize) ≥ 0 ∧ ∀y⊂C(x.inUse − x.maxSize) ≥ (y.inUse − y.maxSize)
6:      if ¬Exists(s) then
7:        s = PerSessionCache(p);
8:      end if
9:      DropOldestPacket(s);
10:   end if
11:   if C.inUse < C.maxSize then
12:     CachePacket(C, p);
13:   end if
14: end if
```

Each session is assigned a fraction of packet slots from the overall cache *C*. The capacity of the cache is given by *C.maxSize* in number of packets, and the number of packet slots currently in use is given by *C.inUse*. Likewise, the cache size assigned to a session *s* is given by *s.maxSize*, while the number of cache slots in use by the session is given by *s.inUse*. In case there are still packet slots left in the overall cache, the intercepted packet is always cached independent of the number of slots already in use by the respective session (lines 11 and 12). When the cache is already full, i.e., $C.inUse = C.maxSize$, a packet is dropped from one of the sessions whose $s.inUse - s.maxSize$ difference is positive and the largest. If no session is found under these conditions, a packet is dropped from the session to which *p* belongs (line 9). Packet *p* is then cached in the released slot (lines 11 and 12).

When sending a locally generated packet or forwarding an intercepted packet, the transport layer always requests the next hop address to the Session Manager.

The proposed scheme requires special extensions of the transport layer header in order to carry the route feedback information in NACK and ACK packets as well as to establish the reverse path. These extensions are transparent to the transport protocol and shall be treated in detail in the next section.

Session manager

As already mentioned, both the routing and transport headers carry extensions in order to support the Session Manager's route selection mechanism. These extensions are transparent to the transport and routing protocols and are only interpreted by the Session Manager.

In the forward direction, transport packets (e.g., DATA and EAR packets in DTSN) carry a small extension consisting of the address of the previous node as they travel from the source to the destination. This field is, of course, updated at every hop and allows the establishment of the reverse route which is exactly symmetric to the forward path (see below) independently of the routing protocol.

The other header extension consists of a routing metrics structure (see the 'Transport cache management and cache-based route metrics' section) carried by routing updates as well as transport feedback packets (e.g., NACK and ACK in DTSN) as they are propagated away from the destination. Since these metrics are updated hop-by-hop, they allow the respective Session Managers to update the session information (see below) and eventually decide to move the session to a new route.

Route evaluation and selection are performed by the Session Manager based on Algorithm 1. The Session Manager keeps an active session list based on the analysis of data and control packets performed by the Packet Analyzer. Each element of the active session list includes the following fields:

- *Session ID*. This field identifies the session univocally, and its precise format depends on the transport and network protocols being used. In the case of IP protocol family, it consists of the source and destination IP addresses and port numbers.
- *Node Type*. This field records the local node's role in the transmission of the session: *source*, *destination*, or *intermediate node*.
- *Next Hop*. This is the network layer address of the next hop node in the data forwarding path.
- *Previous Hop*. This is the network layer address of the previous hop node in the data forwarding path, which corresponds to the next hop node in the ACK/NACK path.
- *Route Valid*. This field indicates whether the Next Hop and Previous Hop fields are both currently valid. If the Session Manager has not found a valid bidirectional route for a session, the packets that belong to that session shall ultimately follow the default route provided by the routing protocol.
- *Route Expiry Time*. This field stores the next time the route for this session will be marked invalid and reset. Until then, the current route is used unless it becomes broken.
- *Forward Expiry Time*. As already mentioned, the management of the sessions is performed in a soft-state way. This field keeps the same instant at which the session will be deleted in case no packet is received in the forward direction, which can be due to the termination of the application or due to the

route being broken between the source node and this node. It is reset each time a packet is received in the forward direction.

- *Reverse Expiry Time.* This field performs a similar role to the previous one, but for packets flowing in the reverse direction (e.g., ACK or NACK). It can be used to detect that the route is broken between this node and the destination of the session. It is reset each time a packet is received in the reverse direction.
- *Routing Metrics.* This field is exactly the same as the Routing Metrics header extension (see above) and is calculated over the segment between the present node and the destination of the session. It is updated whenever an ACK or NACK packet belonging to the respective session is received/intercepted.

The core of the Session Manager is the Packet Analyzer. This module analyzes each transport layer packet in order to identify new active sessions as well as to check and/or refresh the expiry times of existent sessions and session routes. For the sake of processing efficiency, the Packet Analyzer is only invoked when the transport layer presents a received/intercepted packet for processing. In case of an intercepted packet, the latter is allowed to proceed to the intended recipient after the Packet Analyzer routine terminates, following the path (Previous Hop or Next Hop fields) specified in the session record.

The routing protocol is only used to establish the forward route, i.e., from sender to destination. The reverse route consists simply of a chain of Previous Hop addresses on the forward route, assuring the required symmetry.

Algorithm 3 is executed by the Packet Analyzer, assuming that DTSN is used at the transport layer.

In the beginning, all session records whose Forward Expiry Time or Backward Expiry Time have expired are terminated. Session record termination implies that the associated cache resources are released for use by the other active sessions. This is followed by the packet processing code, which is different depending on the type of DTSN packet. For intercepted NACK and ACK packets, the Session Manager only refreshes the session's Route Expiry Time and updates the routing metrics of the session record and of the packet (lines 4 to 9). For DISC packets, the session is terminated (lines 10 to 13).

For EAR and DATA packets, a new session record is created in case it still does not exist (lines 15 to 21). The previous hop is updated in order to account for any routing changes (line 22), and the address of the current node replaces the received previous hop address in the packet (line 23). Then, in case the route is invalid or has just expired, a new route is selected (lines 24 to 37). In case the old route is valid but has just expired, it is only updated if the routing metrics are worse than those of the routing protocol's current best route (lines 27 to 35). In order to fairly compare the cache utilization, the metrics of the route provided by the routing protocol must be previously changed to take into account the weight ω_i^n of the session as if it was already using that route (line 26). In case the

Algorithm 3 Pseudocode of the Packet Analyzer

```
Require: Packet p, RoutingProtocol routing
 1: Route newRoute;
 2: SessionRecord r;
 3: RemoveExpiredSessions();
 4: if p ∈ {ACK, NACK} then
 5:    if r = GetSessionRecord(p) then
 6:       RefreshSession(r);
 7:       r.routingMetrics = p.routingMetrics;
 8:       UpdateRoutingMetrics(p.routingMetrics);
 9:    end if
10: else if p ∈ {DISC} then
11:    if r = GetSessionRecord(p) then
12:       Remove(r);
13:    end if
14: else if p ∈ {EAR, DATA} then
15:    if r = GetSessionRecord(p) then
16:       RefreshSession(r);
17:    else
18:       r.sessionID = p.sessionID;
19:       r.routeValid = FALSE;
20:       AddSession(r);
21:    end if
22:    r.previousHop = p.previousHop;
23:    p.previousHop = GetLocalAddress();
24:    if (r.nodeType ≠ DESTINATION)∧
        ((r.routeValid = FALSE)∨
         RouteExpired(r.routeExpiryTime)) then
25:       newRoute
           = routing.GetRoute(r.sessionID.destination);
26:       newRoute.routingMetrics.sumWeights =
           newRoute.routingMetrics.sumWeights+
           r.weight;
27:       if (newRoute.valid = TRUE) then
28:          if (r.routeValid = FALSE)∨
              (CompareRoutes(r.routingMetrics,
               newRoute.routingMetrics) > 0) then
29:             r.routeValid = TRUE;
30:             r.routeExpiryTime = Now()+
                 ROUTE_TIMEOUT;
31:             r.nextHop = newRoute.nextHop;
32:             r.routingMetrics =
                 newRoute.routingMetrics;
33:             routing.UpdateRoutingMetrics(newRoute);
34:          end if
35:       end if
36:    end if
37: end if
```

new route is adopted for the session, the routing protocol is notified so that it will immediately take into account that an additional session is now using the cache (line 33). The routing protocol is expected to change the routing metrics accordingly. This procedure will result in balanced cache utilization.

It should be noted that the Route Expiry Time establishes a tradeoff between route optimality and route stability. While the former is important to maximize the error recovery mechanisms of the transport protocol, the latter is essential to leverage the potential benefits of the caching mechanism. Consequently, it is expected that the transport layer performance be maximum for some value of Route Expiry Time. The optimization of the Route Expiry Time is out of the scope of this paper.

It should also be noted that while the context associated with the transport sessions has non-negligible memory footprint, this depends on the maximum number of simultaneous flows that are allowed to cross the WSN node. The latter can be controlled by means of simple admission control schemes (these mechanisms are out of scope of this paper). Even if the storage capacity of some nodes is not enough to support the Session Manager, they can still coexist and interact with the more powerful nodes at the network layer, though this leads to the graceful degradation of the network efficiency.

Simulation results

Evaluation of the proposed route metrics and cross-layer architecture was conducted with network simulation using an implementation of the DTSN protocol for the ns-3 [28] platform. Since ns-3 still does not support the IEEE 802.15.4 standard, the wireless technology used in the simulations is 802.11g at 6 Mbps, with disabled Request-To-Send/Clear-To-Send (RTS/CTS) and a number of retries equal to 4, in order to make it as similar as possible to the IEEE 802.15.4 standard.

In the presented simulations, fixed FER values are used at the PHY layer. In the simulations, the FER values were set based on the PER measurements reported in [29] for IEEE 802.15.4, which point to maxima of around 0.035 for indoor scenarios and 0.1 for outdoor scenarios. With IEEE 802.11b/g interference, the worst-case PER varies approximately between 0.20 (for 20-byte packets) and 0.80 (for 127-byte packets), even for very short communication distances (i.e., 5 m). All these experiments have only considered single-hop point-to-point communication. It is expected that in dense multihop WSNs, the PER may become as high as 0.2 or even 0.3 due to co-channel interference.

The main protocol stack parameters used in the simulations are listed in Table 2. The application packet size of 1,024 octets is huge when compared with typical packet sizes employed in WSN applications. In our simulations, this packet size tries to compensate the high physical data rate employed in the simulations, resulting in a transmission delay within the same order of magnitude as that obtained with IEEE 802.15.4 using smaller packets.

Table 2 Protocol stack parameters used in the simulations

Protocol	Parameter	Value
PHY	Data rate (Mbps)	6
	Error model	Fixed FER
	Tx. range (m)	~ 160
MAC	Max. Tx. attempts	4
	RTS/CTS	No
DSDV	Update interval (s)	50
	Epoch length	∞
DTSN	Max. Tx. attempts	∞
	AW size (packet)	40
	Number of AWs per Tx. window	2
Session Manager	Route lifetime	300
	Forward expiry time (s)	1,500
	Reverse expiry time (s)	1,500
Application	Packet size (byte)	1,024
	Traffic pattern	CBR
	Per-session packet inter-arrival time (s)	1.0

The network topology consists of a 10×10 grid with 110 m of vertical and horizontal inter-node distance. The simulations are organized in three main scenarios (see Table 3). The first scenario consists of a test with uniform link quality, where the cache size of each node is configured in a way that allows testing of the effect of each cross-layer architecture component, as well as the impact of the cache utilization metric. The second scenario considers a fixed cache size and links of different qualities so that the impact of each route metric is assessed, with emphasis on the AccLQ metric. The third scenario is similar to the second scenario, but link quality and cache sizes are chosen randomly within established value ranges. Additionally, tests are performed with both two sessions and four sessions.

Scenario 1: customized cache sizes

The topology used in this scenario is depicted in Figure 2. The source node (node 40) is represented by a grey square, and the destination (node 49) is represented by a grey triangle. Empty circles represent nodes with $C_n = 0$, while black circles represent nodes with $C_n > 0$. Two sessions are activated at the source node, having the same destination. The simulations stop once 10,000 packets are correctly received by the destination of each session. As can be seen, there are two cache-rich routes. The cross-layer architecture should be able to balance the cache utilization by sending each session through a different

Table 3 Scenario characteristics

Scenario	Constants	Independent variables	Dependent variables	Objective
1	PER	Cache metrics on/off	ETX	Test impact of cache metrics
	Cache-rich nodes	Session Manager on/off		and Session Manager
		Cache-rich C_n		
2	C_n	Route metrics	ETX	Test impact of
	Good and bad links	PER on good links		link quality
	PER on bad links			
3	C_n	Route metrics	ETX	Test overall impact of
		PER		route metrics
		C_n		
		Number of flows		

route. In order to evaluate the impact of each component of the cross-layer architecture, simulations were run with four different protocol stack configurations where DTSN caching is enabled, identified by the code listed in Table 4. A fifth configuration (designated end-to-end (E2E)) consists of disabling the DTSN caching mechanism, together with $r = 0$ and $t = 0$. It should be noted that the configuration $r = 0$ and $t = 0$ does not disable the transport caching mechanism: it simply sets the routing metric as the hop distance and disables route management by the Session Manager. Two PER values are considered: 0.2 and 0.1. Since these values are uniform in each simulation setting, the AccLQ metric becomes irrelevant in practice.

The ETX for PER $= 0.2$ is depicted in Figure 3, as a function of cache size in the black nodes. As expected, the worst results belong to the 'E2E' configuration. Partial configurations with $(r = 1, t = 0)$ or $(r = 0, t = 1)$ present worse performance than $(r = 1, t = 1)$. The $(r = 1, t = 0)$ configuration is unable to balance the cache utilization since the routing protocol selects only one route for each destination. The consequence is that both sessions are routed through the same route. On the other hand, in the $(r = 0, t = 1)$ configuration, while the Session Manager tries to balance the cache utilization, its route selection is restricted to the options provided by the routing protocol, which in this case employs the hop distance as the routing metric, with no regard for cache utilization. Only the $(r = 1, t = 1)$ configuration is able to select the best routes based on cache utilization, as well as to perform a balanced assignment of routes to the active sessions, even if the latter has the same destination.

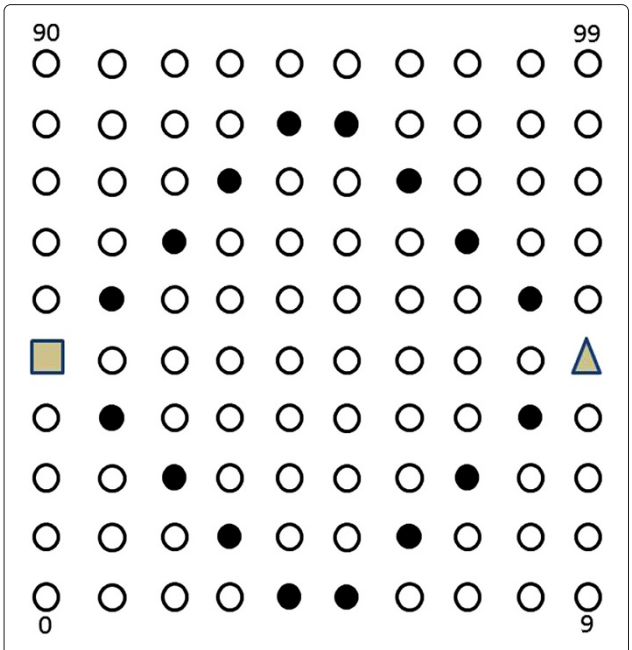

Figure 2 Network topology for scenario 1.

The ETX for PER $= 0.1$ is depicted in Figure 4. As expected, the overall cost is lower, but caching and routing strategies still have a very significant performance impact. The difference between $(r = 1, t = 0)$ and $(r = 0, t = 1)$ is now negligible. As the PER becomes lower, the differences between the curves shrink until cache has no noticeable impact.

Scenario 2: customized link quality

The topology used in this scenario is similar to the one depicted in Figure 2, but now the cache size is uniform ($C_n = 40$), and the links between the black circles feature PER $= 1 \times 10^{-7}$ (fixed). Routing functions are active both at the network layer and at the Session Manager, which corresponds to the $(r = 1, t = 1)$ configuration of the previous scenario. The simulations try to evaluate the impact of the AccLQ metric as the PER of the other links

Table 4 Protocol stack configuration codes

	0	1
Routing metric (r)	H	$H + \text{CU}$
Session Manager active (t)	No	Yes

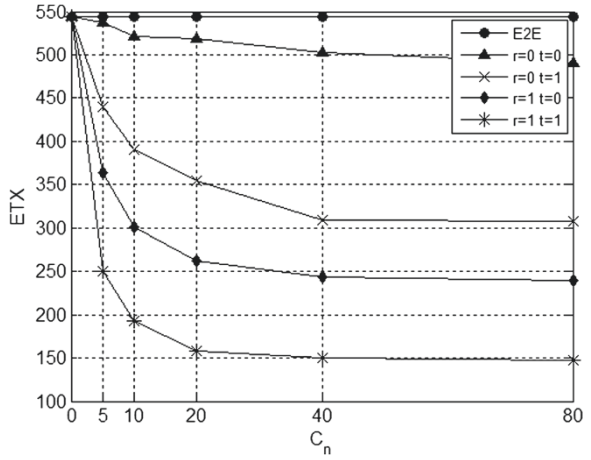

Figure 3 Scenario 1: ETX as a function of cache size for PER = 0.2.

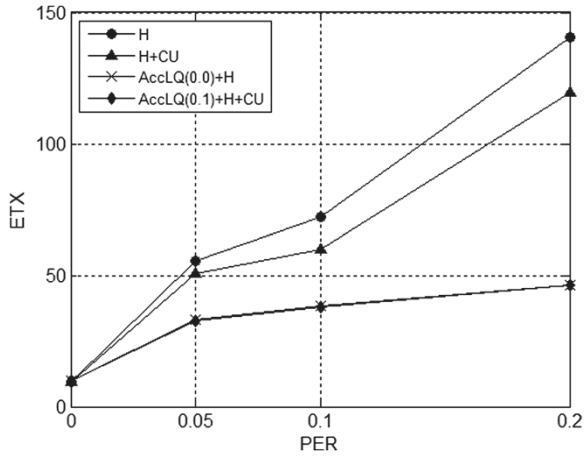

Figure 5 Scenario 2: ETX as a function of the PER.

increases while being uniform in each simulation setting. The results are depicted in Figure 5. Each curve corresponds to the use of different routing metrics, which are evaluated in the order they appear from left to right, as per Algorithm 1. AccLQ(0.1) stands for the AccLQ metric with LQTOLERANCE = 10%.

The H metric alone provides the worst performance. The combination between H and CU reduces the ETX by making better use of caching. Although the cache size is uniform, CU is able to provide cache balancing for the two sessions, which results on more cache being assigned to each. However, link quality has the most significant impact. CU is advantageous here only because the routing protocol is unable to find the best paths from an AccLQ point of view. As can be seen, when the AccLQ metric is used, it greatly reduces the ETX. It is interesting to see that in this scenario, CU had no impact at all when combined with AccLQ. The explanation is now obvious. The AccLQ metric is able to find the paths formed by the links with PER $= 1 \times 10^{-7}$. With such a low PER value, the number of retransmissions is very low, and thus, the impact of caching is negligible, matching the results in [7].

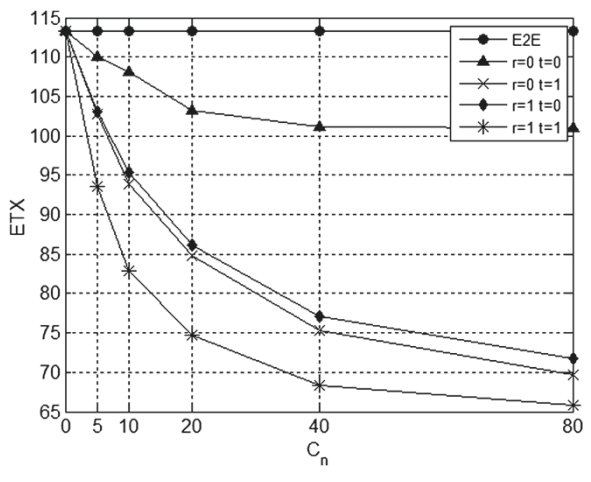

Figure 4 Scenario 1: ETX as a function of cache size for PER = 0.1.

Scenario 3: random link quality and cache size

This scenario presents characteristics that are more akin to what is expected from an IoT environment. The topology used in this scenario is again the 10×10 grid. Simulations are done for two sessions (keeping the same position for the sender and receiver nodes) as well as four sessions. The traffic model is the same as for the previous simulations, i.e., constant bitrate (CBR) with a packet inter-arrival time of 1 s. However, the PER is now randomly chosen using a uniform probability distribution within the interval $[\text{PER}_{\text{min}}, \text{PER}_{\text{max}}]$, where PER_{max} is a PER upper limit fixed at 0.2 and PER_{min} is a lower PER limit. The latter constitutes the independent variable in the simulations. The larger the interval $[\text{PER}_{\text{min}}, \text{PER}_{\text{max}}]$, the higher the link quality differentiation. On the other hand, the smaller the interval $[\text{PER}_{\text{min}}, \text{PER}_{\text{max}}]$, the easier it is to find more routes featuring a similar AccLQ. Regarding the cache size, it is now considered that 20% of the nodes are more powerful nodes with $C_n = 60$, while the remaining 80% of the nodes have no caching capability at all ($C_n = 0$). The powerful nodes are randomly selected.

Figure 6 depicts the ETX as a function of PER_{min} for two sessions, with different combinations of routing metrics. As expected, for large $[\text{PER}_{\text{min}}, \text{PER}_{\text{max}}]$, the AccLQ metric is more important than the CU. However, when the number of routes with similar AccLQ is larger (small $[\text{PER}_{\text{min}}, \text{PER}_{\text{max}}]$), CU is important to find the most energy-efficient routes. This can be easily concluded from the $H + \text{CU}$ curve, which presents one of the highest ETX for low PER_{min} while presenting one of the lowest ETX for high PER_{max}. The variation of the AccLQ tolerance

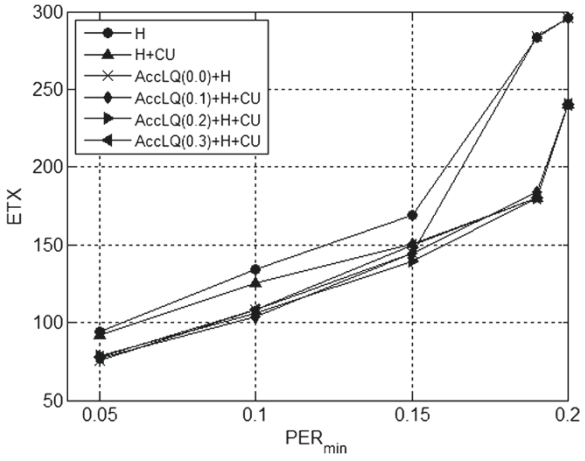

Figure 6 Scenario 3: ETX as a function of PER_{min}, for PER_{max}= 0.2 for two sessions.

had practically no effect in the simulated scenarios, with a value of 0.1 being enough to achieve good AccLQ-similar route differentiation.

It should be noted that the $[PER_{min}, PER_{max}]$ is just a simple and artificial way to control the number of alternative routes with similar AccLQ. In fact, the relationship between the $[PER_{min}, PER_{max}]$ and the number of found AccLQ-similar paths depends on the node density: in dense scenarios - where the number of alternative routes is higher - CU is expected to have more impact than in sparse scenarios.

Figure 7 depicts the ETX as a function of PER_{min} for four sessions. At this time, the sender-receiver pairs are the following: (70, 29), (79, 20), (2, 97), and (7, 92). The results are very similar to the situation with two sessions, with the

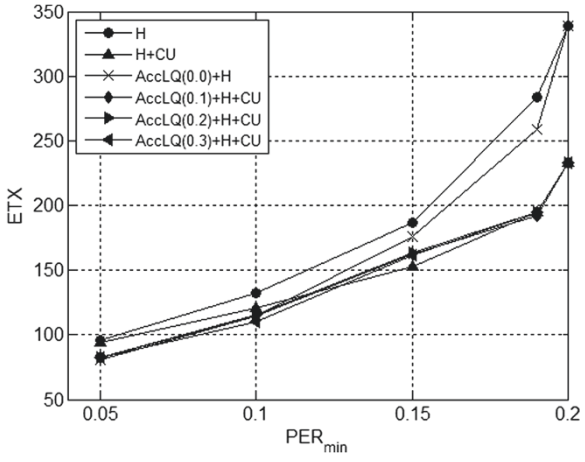

Figure 7 Scenario 3: ETX as a function of PER_{min}, for PER_{max}= 0.2 for four sessions.

ETX being a little bit higher for $PER_{min} = 0.2$. This probably indicates the existence of bottleneck nodes where the cache must be shared by more than two sessions. Despite this bottleneck effect, the CU metric is again an important means to lessen the effects of high PER values, being able to find cache-richer routes.

Conclusions

This paper has presented a new set of metrics meant to establish efficient routes on behalf of sessions transmitted over cache-assisted reliable transport protocols. Three routing metrics and their combination were considered for the evaluation of routes: link quality, hop distance, and cache utilization. The cache utilization metric is a novel metric specially useful in IoT scenarios, where a huge disparity between the memory resources of network nodes is likely to occur.

An associated cross-layer architecture for delay-tolerant WSNs was also proposed. In this architecture, while the routes are still determined by the network layer, they are assigned to the individual transport layer sessions. Central to the cross-layer architecture is an entity designated as the Session Manager. The latter is responsible for managing the assignment of routes to individual transport sessions and also for the interpretation and evaluation of routing metrics. These routing metrics are completely transparent to the routing protocol, increasing the modularity of the architecture and making it especially suitable for implementations based on RPL, whose specification offers a wide set of possible routing metrics, which is easy to extend. The assignment of routes to individual transport sessions has an additional advantage, which is to allow a balanced distribution of resources among different transport sessions, even when these flow between the same communication endpoints. In this paper, the transport layer cache is the main resource whose usage must be leveraged and balanced.

Simulation results have shown that the optimization of energy efficiency can only be achieved when combining all the three metrics in the route evaluation procedure, allowing the system to adapt to a larger diversity of scenarios. In this way, when the alternative paths differ significantly in terms of the quality of the links, the link quality metric becomes more relevant for route selection. On the other hand, when the alternative paths are similar in terms of link quality, the cache metrics become more relevant.

Future work by the authors shall address the integration of cache partitioning policies in the proposed architecture, as well as adaptations to suit delay-sensitive services requiring reliable delivery. Another promising topic concerns the adaptation of the transport caching concept to data-centric convergecast scenarios. In this case, benchmarking and optimization results could be obtained,

for example, by integrating and extending the analytical frameworks presented in [7,30].

Endnotes

[a]The word *session* is used instead of *flow* since, by definition, one network layer *flow* has a well-defined source and destination, whereas a transport layer *session* may combine a data flow and a control flow (e.g., acknowledgment packets) in the opposite direction.

[b]A mule is a special mobile WSN node (e.g., robot) that is sent to retrieve the data stored on a remote WSN and then transports the data to the central system where it shall be processed.

[c]A cache partitioning policy defines the rules according to which the cache of one node is assigned to the sessions that simultaneously cross that node.

[d]A loop path always presents higher H than any of its subpaths. Similarly, a loop path always presents a lower AccLQ than any of its subpaths, being also a strictly monotonic metric. However, the introduction of LQTOLERANCE compromises strict monotonicity, allowing a loop path to be considered equivalent to some of its subpaths in terms of AccLQ if the difference of AccLQ is within the LQTOLERANCE interval.

[e]In future work, the authors plan to evaluate how this requirement can be made more flexible.

[f]This estimate assumes a binary symmetric channel.

Competing interest
Both authors declare that they have no competing interests.

Acknowledgements
This work was supported in part by national funding from FCT - Fundação para a Ciência e a Tecnologia (INESC-ID multiannual funding) through the PIDDAC program funds under project PEst-OE/EEI/LA0021/2013 and by the MPSat project (FCT Ref. PTDC/EEA-TEL/099074/2008). The authors wish to thank the wireless team at the Fraunhofer ESK for the interesting and fruitful discussions and brainstorming, in particular Günter Hildebrandt and Neda Petreska. António Grilo is also grateful to Prof. Dr. Matthias Kranz from the Technische Universität München for having hosted him during the first stages of this work.

Author details
[1]IST/INESC-ID/INOV, Rua Alves Redol, N° 9, Lisbon 1000-029, Portugal.
[2]Fraunhofer Institute for Communication Systems, Munich 80686, Germany.

Received: 26 July 2012 Accepted: 13 May 2013
Published: 28 May 2013

References

1. L Atzori, A Iera, G Morabito, The Internet of Things: a survey. Comput. Netw. J. Elsevier **54**(15), 2787–2805 (2010)
2. C Wan, A Campbell, L Krishnamurthy, in *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*. PSFQ: a reliable transport protocol for wireless sensor networks (ACM, New York, 2002), pp. 1–11
3. F Stann, J Heidemann, in *Proceeding of the 1st IEEE International Workshop on Sensor Net Protocols and Applications*. RMST: reliable data transport in sensor networks. (Anchorage, Alaska, May 2003 (IEEE)), pp. 102–112
4. A Dunkels, J Alonso, T Voigt, H Ritter, in *Proceedings of the 3rd Annual Mediterranean Ad-Hoc Networks Workshop*. Distributed TCP caching for wireless sensor networks. Bodrum, Turkey, June 2004
5. B Marchi, A Grilo, M Nunes, in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'07)*. DTSN: distributed transport for sensor networks (Aveiro, Portugal, July 2007 (IEEE)), pp. 165–172
6. F Rocha, A Grilo, P Pereira, M Nunes, A Casaca, in *Proceedings of Wireless Systems and Mobility in Next Generation Internet: 4th International Workshop of the EuroNGI/EuroFGI Network of Excellence: 16-18 January 2008, Barcelona, Spain*. Performance evaluation of DTSN in wireless sensor networks (Springer, New York), pp. 1–9
7. N Tiglao, A Grilo, An analytical model for transport layer caching in wireless sensor networks. Perform. Eval. J. Elsevier **69**(5), 227–245 (2012)
8. P Sarolahti, J Ott, C Perkins, TCP segment caching. draft-sarolahti-irtf-catcp-00 (2012). [IRTF draft]. http://datatracker.ietf.org/doc/draft-sarolahti-irtf-catcp/. Accessed 22 May 2013
9. D de Couto, D Aguayo, J Bicket, R Morris, in *Proceedings of the 9th Annual ACM International Conference on Mobile Computing and Networking (MOBICOM'2003): September 2003 San Diego, CA, USA*. High-throughput path metric for multi-hop wireless routing (ACM, New York, 2003), pp. 134–146
10. C Wang, K Sohraby, B Li, M Daneshmand, Y Hu, A survey of transport protocols for wireless sensor networks. IEEE Netw. **20**(3) (2006)
11. Y Liu, H Huang, K Xu, in *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*. Multi-path-based distributed TCP caching for wireless sensor networks, (Washington DC, USA, July 2007 (ACIS, IEEE)), pp. 134–146
12. A Ayadi, P Maille, D Ros, in *Proceedings of the 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2010)*. Improving distributed TCP caching for wireless sensor networks, (Juan Les Pins, France, 23-25 June 2010 (IFIP, IEEE)), pp. 1–6
13. N Xu, S Rangwala, K Chintalapudi, D Ganesan, A Broad, R Govindan, D Estrin, in *Proceedings of the Second International Conference on Embedded Networked Sensor Systems (SenSys 2004): November 2004, Baltimore, MD, USA*. A wireless sensor network for structural monitoring (ACM Press, New York 2004), pp. 13–24
14. E Paek, R Govindan, in *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (Sensys 2007)*. RCRT: rate-controlled reliable transport for wireless sensor networks, Sydney, Australia, 23–25, June 2010 (ACM Press), pp. 305–319
15. M Yaghmaee, D Adjeroh, in *Proceedings of the 4th International Symposium on Telecommunication (IST2008)*. A reliable transport protocol for wireless sensor networks, (Tehran, Iran, August 2008 (IEEE)), pp. 440–445
16. H Zhou, X Guan, C Wu, in *Proceedings of the IEEE International Conference on Communications (ICC 2008)*. Reliable transport with memory consideration in wireless sensor networks, (Beijing, China, May 2008 (IEEE)), pp. 2819–2824
17. N Handigol, K Selvaradjou, C Murthy, in *Proceedings of the International Conference on High Performance Computing (HiPC 2010)*. A reliable data transport protocol for partitioned actors in wireless sensor and actor networks, (Goa, India, December 2010 (IEEE)), pp. 1–8
18. C Intanagonwiwat, R Govindan, D Estrin, J Heidemann, F Silva, Directed diffusion for wireless sensor networking. IEEE/ACM Trans. Netw. **11**, 2–16 (2003)
19. C Perkins, P Bhagwat, in *Proceedings of the ACM SIGCOMM'94 1994, London, UK*. Highly destination-sequenced dynamic distance vector routing (DSDV) for mobile computers (ACM New York, 1994), pp. 234–244
20. A Grilo, K Piotrowski, P Langendoerfer, A Casaca, in *Proceedings of the 8th International Conference on Ad-Hoc Networks and Wireless (ADHOCNOW-2009): September 2009, Murcia, Spain. Lecture Notes in Computer Science, vol. 5793*. A wireless sensor network architecture for homeland security application (Springer, 2009)
21. A Grilo, A Casaca, M Nunes, C Fortunato, in *Proceedings of the International Conference CIP 2010: September 2010*. Wireless sensor networks for the protection of an electrical energy distribution infrastructure (Springer, Berlin/Heidelberg, 2010), pp. 373–383
22. T Winter, P Thubert, A Brandt, J Hui, R Kelsey, P Levis, K Pister, R Struik, JP Vasseur, R Alexander, RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550, IETF, (2012). http://tools.ietf.org/html/rfc6550. Accessed 22 May 2013
23. J Vasseur, M Kim (Ed.), K Pister, N Dejean, D Barthel, Routing metrics used for path calculation in low-power and lossy networks. RFC 6551, IETF (2012). http://tools.ietf.org/html/rfc6551. Accessed 22 May 2013

24. T Zahariadis, P Trakadas, Design guidelines for routing metrics composition in LLN. draft-zahariadis-roll-metrics-composition-03 (2012). [IETF draft]. http://tools.ietf.org/html/draft-zahariadis-roll-metrics-composition-03. Accessed 22 May 2013
25. S Simoens, D Bartolome, in *Proceedings of the IEEE Vehicular Technology Conference 2001 (VTC 2001) Spring, Volume 2*. Optimum performance of link adaptation in HIPERLAN/2 networks, (Rhodes, Greece, May 2001 (IEEE)), pp. 1129–1133
26. HM Tsai, N Wisitpongphan, O Tonguz, in *Proceedings of the 1st International Symposium on Wireless Pervasive Computing (ISWPC'2006)*. Link-quality aware ad hoc on-demand distance vector routing protocol, (Phuket, Thailand, January 2006 (Springer)), pp. 1–6
27. S Cho, J Sin, B Mun, in *Proceedings of the IEEE Global Telecommunications Conference 2003 (GLOBECOM'03), volume 3*. Mechanism to improve the reliability of the broadcasting for multisources, (San Francisco, CA, USA, December 2003 (IEEE)), pp. 1263–1268
28. The Network Simulator 3 – ns-3. [http://www.nsnam.org/]. Accessed 22 May 2013
29. M Petrova, J Riihijärvi, P Mähönen, S Labella, in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2006): April 2006, Las Vegas, NV, USA*. Performance study of IEEE 802.15.4 using measurements and simulations, (IEEE New York 2006), pp. 487–492
30. V Pilloni, L Atzori, Deployment of distributed applications in wireless sensor networks. Sensors, MDPI. **11**, 7395–7419 (2011)

doi:10.1186/1687-1499-2013-139
Cite this article as: Grilo and Heidrich: **Routing metrics for cache-based reliable transport in wireless sensor networks.** *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:139.