

tinyLUNAR: One-Byte Multihop Communications Through Hybrid Routing in Wireless Sensor Networks

Evgeny Osipov^{*,**}

LTU Luleå University of Technology,
Department of Computer Science and Electrical Engineering,
Campus Porsön, S-971 87 Luleå, Sweden

Abstract. In this paper we consider a problem of implementing a hybrid routing protocol for wireless sensor networks, which natively supports data-centric, geographic-based and address-centric communication paradigms. We demonstrate the feasibility of such protocol by presenting tinyLUNAR, an adapted to the specifics of sensor networks reactive routing scheme originally developed for mobile wireless ad hoc networks. In addition to the support for several communications paradigms tinyLUNAR implements highly efficient multihop forwarding using only 1 B field that can be directly encoded in the standard IEEE 802.15.4 MAC header.

1 Introduction

Over the recent years wireless sensor networks (WSN) appeared as a unique networking environment with respect to routing amongst other aspects. Firstly, WSNs inherit the need for routing on geographic coordinates from its closest “relative” mobile ad hoc networks (MANETs) due to spatial distribution of nodes. Secondly, differently from MANETs and the Internet in general where communications are purely address-centric, the communications in sensor networks are heavily data-centric. With this type of communications a set of nodes satisfying certain attributes (e.g. particular readings of on-board sensors larger than a pre-defined threshold in a specific geographic region) should report the information mainly in connection-less manner. However, while data-centric communications dominate in sensor networks, there are numbers of applications that still require address-centric communications (e.g. sending an alarm message to a base station with a pre-defined unique ID). Finally, being built of severely energy and computing resources constrained devices WSNs place serious performance requirements routing protocols and data forwarding.

* The work described in this paper is based on results of the IST FP6 STREP UbiSec&Sens (www.ist-ubisecens.org).

** A significant part of this work has been performed in the Department of Wireless Networks in RWTH Aachen University while the author was working there.

The motivation behind this work is straightforward. During the last several years the area of WSN routing blossomed with a variety of protocols that separately support data-centric, geographic-based and address-centric communication paradigms. A good overview of the existing approaches is presented in [1]. One general conclusion, however, is that none of the existing protocols supports all three communication styles. This leads to a situation that in complex WSN applications such as large scale surveillance in homeland security scenarios where all types of communications are present at least three software instances for routing is needed. While not arguing against the development of specialized protocols, in this work we want to demonstrate the feasibility of designing an efficient hybrid protocol natively supporting multiple communication paradigms.

In this paper we present tinyLUNAR, an adapted Lightweight Underlay Ad hoc Routing protocol [18] to the specifics of WSN environment. The major contribution delivered by our protocol is implementation of the multihop forwarding using a one byte field that can be encoded directly in the IEEE 802.15.4 MAC header. Secondly, tinyLUNAR supports multiple data communication types in one package, exposes flexible interfaces to an application level programmer and delivers competitive performance in comparison to the existing protocols. The protocol is implemented under TinyOS v.2.x¹ and currently is the default routing scheme for the secure distributed data storage middleware tinyPEDS [3]. To the best of our knowledge currently there are no competitors to tinyLUNAR by its existing functionality and potential capabilities.

The paper is structured as follows. In Section 2 we present the considered network model overview routing principles in wireless sensor networks and formulate the design requirements for tinyLUNAR. We outline our solution and present the background material in Section 3. The details of tinyLUNAR operations follow in Section 4. Overviewing the related work in Section 5 we discuss future developments of the protocol in Section 6. We conclude the paper in Section 7.

2 Problem Statement: Design Objectives for tinyLUNAR

2.1 Networking Model of WSN

We design tinyLUNAR for a network formed by wireless sensor nodes arbitrarily deployed in a monitored area. We do not place any assumptions either on the scale of the network or on its topological structure. The network can be hierarchical, deploying certain cluster formation scheme, or flat. We consider a relatively static network with low or no node mobility. The low mobility is implicitly present in a form of changing the position of a cluster head node and eventual node failures.

Note that we neither place any specific assumptions on the structure of node's IDs. Moreover, we consider a general case where each sensor node maintains a set of identities, including a) MAC addresses; b) position information (geographic coordinates, relative position); c) functional roles (cluster head, actuator);

¹ TinyOS web portal. Online. Available: <http://www.tinyos.net>

d) description of on-board sensors (temperature, humidity); etc. We also do not consider availability of a centralized directory service for address/name resolution.

The target application of the sensor network requires both data-centric and address-centric communications. The network supports anycast, unicast, convergecast and multicast traffic flows.

2.2 Routing in Wireless Sensor Networks

There are two routing approaches applicable to wireless sensor networks. In centralized routing schemes [11,2] the forwarding map is computed at a base station based on the link and neighbor information gathered from the network. While one can achieve accurate computation of optimal paths in this case, the centralized schemes are known for poor scalability as they require periodic gathering of the topology information from all network nodes. On the other hand, in the distributed approaches [5, 13, 14] the routes are computed cooperatively by all network nodes exchanging the network state and control information. In this paper we concentrate on the *decentralized* routing approaches only.

The distributed routing approaches fall into two global categories: the *proactive* and *reactive* schemes. The *proactive* approach is inspired by the routing experience in the wireline Internet. The routing topology is created prior to data transmissions from mobile nodes. The routing information is then dynamically updated according to changes of the network topology. In contrast, the *reactive* routing approach assumes no existing routing state in the network prior to data transmission from the particular station. Upon arrival of a first data packet the node enters a route discovery phase in which it announces the request for the particular destination address to the network. In reactive routing the routing information is maintained in the network only for the period of activity of the particular session. The major representatives of proactive routing for WSN is DSDV [14]. For reactive routing these are adapted versions of DSR [8] and AODV [13].

A special subclass of reactive routing schemes constitute *self-routable* protocols. In this approaches the forwarding state is not maintained by intermediate relay nodes. Instead, the forwarding decision is taken individually for every packet. Examples of the self-routable schemes are geographic routing [10, 9] and flooding.

2.3 Design Objectives for tinyLUNAR

The protocols referenced in the previous section represent only a small share of approaches developed for sensor networks during the last decade. While each scheme targets a specific need in WSNs one conclusion is clear: Currently there are no solution supporting several communication types in one package. The major *design objective* for tinyLUNAR is a capability to support as many types of traffic flows as possible for both data-centric and address-centric communications. Of course, functional universality comes at a price of increased complexity. With tinyLUNAR we want to understand how complex a universal protocol can be. At the same time we are not arguing against the development of specialized routing protocols. They obviously are best suited in specially engineered and rather narrow-purpose

WSNs. However, we foresee that in large scale distributed networks with rich functionality a single implementation of routing supporting several connection types is more efficient than several separate single purpose schemes.

3 From LUNAR to tinyLUNAR: Solution Outline

The original Lightweight Underlay Adhoc Routing [18] is a reactive protocol that uses a simple mechanism of flooding and limited re-broadcasting (by default the range is limited to three hops) to establish a label switching *virtual circuit* in the forwarding plane. While keeping the core logic of its predecessor the operations of tinyLUNAR is different in the following way. Firstly, tinyLUNAR does not interpret data packets as connection initiation events. Instead, the protocol exposes well defined interfaces that allow upper layer programmers to configure the characteristics of the path. With these interfaces it is possible to parametrically specify a) various classes of destination identifiers; b) the type of the desired communications; and c) the propagation behavior for the route request messages.

Secondly, in the data forwarding plane we change the dimension and impose certain structure on the *selector*² field. The size of the selector field in tinyLUNAR is 1 byte. This potentially allows implementing the multihop forwarding using only the *type* field of the standard IEEE 802.15.4 MAC header and not consuming the payload space.

Finally, we scale the forced path re-establishment mechanisms accordingly to satisfy bandwidth and energy limitations of wireless sensor networks. While the periodic route rediscovery is still present in tinyLUNAR, the duration of the period is tailored to a specific application that uses the protocol in a particular sensor network. For example, if tinyLUNAR is used in a hierarchical network to build routes toward a cluster head then, naturally, the the period is synchronized with the cluster head re-election mechanism.

3.1 Packet Forwarding Via Label Switching

Label switching is technique for overcoming the inefficiency of traditional layer 3 hop-by-hop routing. In the Internet the label switching (or virtual circuit) model is used amongst others in MPLS [16].

A simplified example of multihop data forwarding using label switching is illustrated in Figure 1. Assume each application running at a particular node is assigned with an ID number which allows a deterministic internal multiplexing of data packets between different applications³. Suppose also our application needs a bidirectional unicast path for communication. In this setting the application with *AppID* = 120 at nodes with addresses *X* and *Z* communicate through a node with address *Y*. The figure shows the content of the forwarding table

² Traversing the network LUNAR's route request messages obtain a special forwarding label (called *selector*) in each router along a path to a destination.

³ Further on, we use the terms *application*, *component*, *interface* in the sense defined by TinyOS.

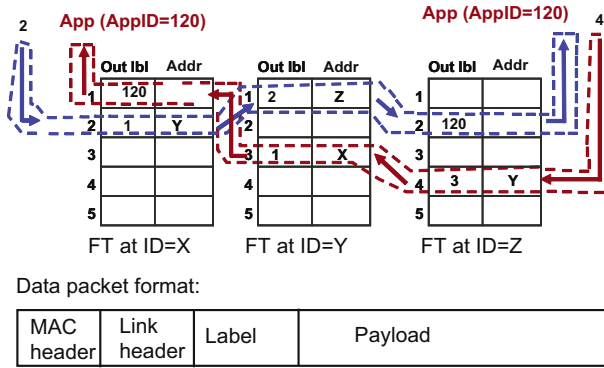


Fig. 1. Packet forwarding via label switching

(FT) in each node. The label switching paradigm consists of two phases: the path establishment and the data forwarding. Upon establishing a connection node *X* locally generates an incoming label and creates an entry in its FT. This number will be used by the next hop node to send packets to *X* on the *backward* path. In the created entry node *X* sets the outgoing label to the application ID 120 as the destination for the incoming packets. The empty address field in the first FT entry of *X* in our example indicates that the packet must be multiplexed locally to the application. Finally, node *X* signals the generated label to the next hop node in a path establishment message. During the forward propagation of this message node *Y* locally generates its incoming label (in our case 3). As an outgoing label it specifies the received incoming label from node *X* and also records the address of the node from which the request is received. In its turn node *Z* figures out that it is the destination node and performs the following actions. Firstly, it creates a backward label (in our case 4), which will be used by local application to send packets to node *X*. The outgoing label is set to the incoming label generated by node *Y*, the address of *Y* is recorded in the corresponding field. Secondly, the local application (*AppID* = 120) is signaled that for sending packets to node *X* it needs to indicate 4 as the entry point to the virtual circuit between *Z* and *X*. Finally, node *Z* initiates building of the forward path from *X* to itself by the same procedure described for node *X*. The path establishment message returns back to node *X* using the established backwards path from *Z* to *Y* and from *Y* to *X*. The result of the path establishment procedure are two numbers available for our communicating applications indicating the entry points of the established virtual circuits (2 from node *X* and 4 from node *Z*). The data forwarding that may commence immediately upon successful path establishment is straightforward. Each packet is assigned with an outgoing label as shown in the figure. In packets sent to the network it is the outgoing label recorded in FT for a specific incoming label. Normally, the incoming labels are directly mapped into FT indices inside the next hop node. Originally, this was meant to allow the label lookup and the label switching procedures to be placed directly within the switching fabric.

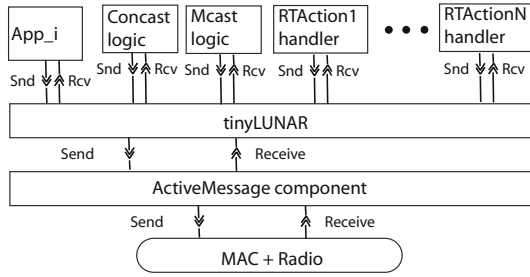


Fig. 2. Position of tinyLUNAR in the TinyOS software architecture

This would enable the label switching technique performing much faster than the traditional routing method where each packet is examined in CPU before a forwarding decision is made⁴.

4 tinyLUNAR: Protocol Description

Overall, the design of tinyLUNAR is partially influenced by the specifics of TinyOS software architecture. These design steps, however, do not affect the applicability of the protocol to a general class of embedded operating systems.

As is the case with its predecessor we position tinyLUNAR directly above the link layer⁵ as illustrated in Figure 2. It intercepts all broadcast and unicast packets addressed to a node and is responsible for their forwarding to the remote destination and local demultiplexing between the communicating components. All networking components communicate directly through tinyLUNAR. Note that by networking components we mean pieces of software with self-consistent functionality, e.g. aggregator node election, multicast/convergecast handlers, etc. Each local component executed in a node is assigned a *locally unique* ID. The networking components can be one of the two types depending on the type of communications : a) connection-less and b) connection-oriented (uni- or bidirectional).

4.1 Modified Selector Structure

The selector field appended to *each* outgoing data packet and *all* tinyLUNAR control routing messages is a *one byte* integer, its format is shown in Figure 3(a).

⁴ One general comment is needed on the uniqueness of the labels. Since incoming labels are assigned internally by the relay node the values are obviously only *locally unique*. The determinism in multihop forwarding between several neighbors is achieved by *per-link uniqueness* introduced by MAC and link layer addresses. In wireless sensor networks achieving the global uniqueness of MAC level IDs is challenging. However, for correct functionality of the label switching forwarding it is sufficient to have a two-hops MAC uniqueness. An example of a distributed address auto-configuration protocol which achieves the needed uniqueness level is [15].

⁵ In TinyOS the *ActiveMessage* component situated directly above the MAC layer can be regarded as the link layer component.

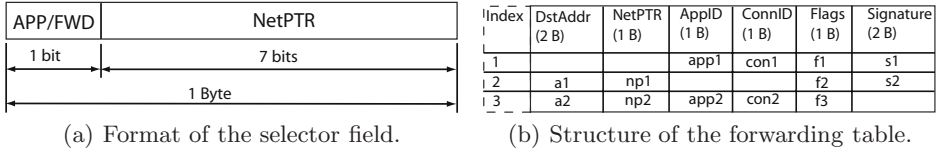


Fig. 3. Format of the selector field and the structure of forwarding table in tinyLUNAR

The first bit of the selector field splits the value space two parts: the application and the forwarding parts. When the *APP/FWD* bit is zero tinyLUNAR treats the following seven bits denoted as *NetPTR* as an ID of the *internal* application or service component where the packet shall be multiplexed. When the *APP/FWD* bit is set the *NetPTR* part is interpreted as the incoming label as explained in Section 3.1.

The rationale behind restricting the selector field to one byte is simple. We want to minimize the overhead due to multihop forwarding. In addition, we want to maximally re-use the standard fields of the IEEE 802.15.4 MAC header. Our intention is to encode the selector value in the *type* field of the later.

4.2 Generation of Incoming Labels

The format of the forwarding table is shown in Figure 3(b). The purpose of the *DstAddr*, *NetPTR*, *AppID* fields is essentially the same as in our example in Section 3.1. The *ConnID* field is used for connection oriented application components to differentiate between incoming (outgoing) connections. The field *flags* is used internally for the FT management purposes; *Signature* is used to detect duplicates of route request messages, its computation is described below. The procedure for local generation of *incoming* labels is as follows. As it is described in Section 3.1, the major objective for the label switching forwarding is to make the route lookup procedure fast. Recall also that many of the embedded operating systems including TinyOS lack the support for dynamic memory allocation. In tinyLUNAR we statically allocate the memory for the forwarding table⁶. The incoming label in our protocol is the index of the FT entry. In this way we completely avoid implementing a sophisticated searching algorithm. Upon reception of a packet with a selector belonging to the forwarding space we find the outgoing label and the MAC address of the next hop by directly indexing the forwarding table with the value extracted from the *NetPTR* part.

Recall that the selector field should be present in all control messages of tinyLUNAR. In order for the routing layer to direct these packets to the corresponding processing blocks we assign the selector values for route request (RREQ) and route reply (RREP) messages in the application space. The two values are the same for all RREQs and RREPs generated by any node.

⁶ The number of entries in the FT is an adjustable parameter of tinyLUNAR, it is bounded by 128 entries (maximal number of incoming labels that can be encoded in 7 bits of the selector).

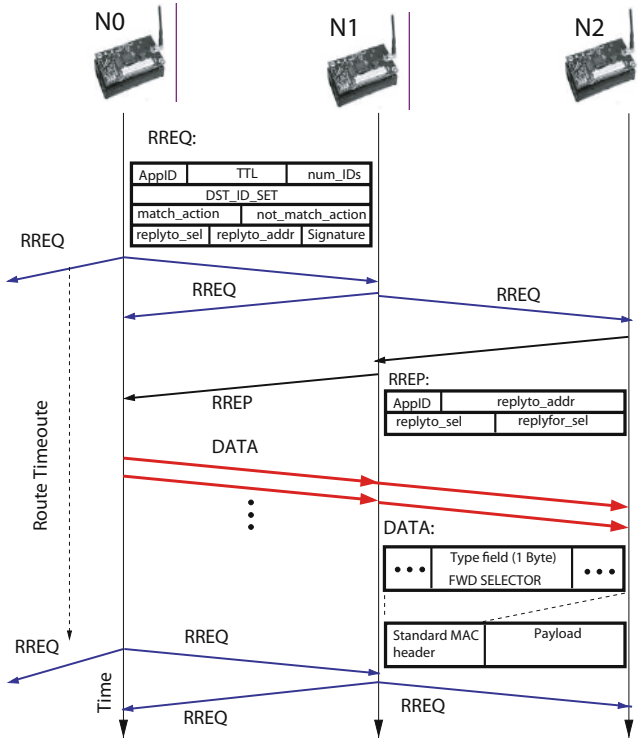


Fig. 4. Path establishment and data forwarding phases of tinyLUNAR

4.3 Path Establishment in tinyLUNAR

The path establishment and the data forwarding phases of tinyLUNAR are schematically shown on a time sequence graph in Figure 4.

TinyLUNAR is a reactive routing protocol where the route establishment is initiated upon a demand from the application. Normally, in *all* reactive routing schemes the indication of such demand is the first data packet arriving from an upper layer component. Recall, however, that our major design objective is to create a flexible routing protocol supporting different types of addressing and communications. We foresee that guessing the routing properties from the content of data packets is difficult. Instead, we decided to expose a set of intuitive interfaces to upper-layer programmers to allow them actively configuring the characteristics of desired routes.

An intuitive abstraction for the route request procedure (RREQ) from the programmer’s point of view is an “if” statement formed by the source node that travels through the network. The intermediate nodes check whether *identity condition* matches their own identity. If yes then the node performs a *matching action* otherwise a *not matching action* is invoked. The identity condition is a set of tuples (*ID_CLASS*, *VALUE*) that describes an individual node or a set

```

RREQinit();
  appendRREQcondition(ID_CLASS, VALUE);
  •
  •
  •
  appendRREQcondition(ID_CLASS, VALUE);
  addRREQaction(MATCH_ACTION, ACTION);
  addRREQaction(NOT_MATCH_ACTION, ACTION);
RREQfini();
    
```

Parameter	Meaning
IDENTITY CONDITIONS (256 ID classes are possible)	
ID_CLASS_GEOGRAPHIC	Geographic coordinates
ID_CLASS_ROLE	Functional role played by node (e.g. cluster head)
ID_CLASS_ADDRESS	Unique address (e.g. MAC)
MATCH ACTIONS (256 MATCH and NOT MATCH actions are possible)	
RT_ACTION_ROUTE_UNIDIR	Establish one way unicast route
RT_ACTION_ROUTE_BIDIR	Establish bidirectional unicast route
RT_ACTION_SUBSCRIBE	Establish a multicast tree
RT_ACTION_REPORT	Establish a convergecast tree (directed diffusion like)
NOT MATCH ACTIONS	
RT_ACTION_REBCAST	Re-broadcast to all. Simple flooding.
RT_ACTION_REBCAST_GEO	Re-broadcast geo based

Fig. 5. Interfaces exposed by tinyLUNAR to actively control the characteristics of the path and examples of ID classes, matching and not matching actions

of nodes. The *match action* allows a programmer to specify the behavior of the destination node(s). With the *not match action* the programmer may coordinate the propagation of the RREQ message through the network. Figure 5 illustrates the above concepts.

The following piece of the NesC code from a networking component shows an example of forming a RREQ message for a bidirectional path to a node which is a cluster head in a neighboring cluster, the RREQ should be propagated using the shortest geographic distance.

```

dstregion= getDSTregion();
if(call tinyLUNAR.RREQinit()== SUCCESS) {
//First condition should always be "OR
  call tinyLUNAR.appendRREQcondition(OR_CLAUSE, ID_CLASS_GEOGRAPHIC,
                                     CONDITION_TYPE_EQ, (void*)dstregion);
//Subsequent conditions could be any
  call tinyLUNAR.appendRREQcondition (AND_CLAUSE, ID_CLASS_ROLE,
                                     CONDITION_TYPE_EQ, (void*)ROLE_CLUSTER_HEAD);
  call tinyLUNAR.addRREQaction(MATCH_ACTION, RT_ACTION_ROUTE_BIDIR);
  call tinyLUNAR.addRREQaction(NOT_MATCH_ACTION, RT_ACTION_REBCAST_GEO);
  call tinyLUNAR.finiRREQpacket(APP_ID);
}
    
```

When being called by the *RREQfini()* interface, tinyLUNAR creates an entry for incoming route reply or data messages, forms and sends the RREQ

message. In the first vacant entry of the forwarding table field *AppID* is the ID of a component that initiates the route discovery. The index of the entry becomes an incoming label. The entry with index 1 in Figure 3(b) is an example of an entry created through the *RREQfini()* interface. The format of the RREQ message is shown in Figure 4. There *AppID* is the ID of the communicating component, *TTL* is the maximum hop count that RREQ is allowed to traverse, *numID* is the number of items in the following identity set (*DST_ID_SET*), *match_action* and *not_match_action* fields are set by using the *appendRREQcondition()* interface, *replyto_sel* is the locally generated incoming label and *replyto_addr* is the MAC address of the node. The *Signature* field is a CRC number computed over all fields of the RREQ message. The completed message is marked with the RREQ selector and is sent to the broadcast MAC address.

The following after that path establishment procedure with respect to setting up the label switching path follows the logic presented in Section 3.1. The entry with index 2 in Figure 3(b) is an example of an entry created at a relay node after processing and re-broadcasting the RREQ message and entry 3 is created at the destination node before sending the RREP message. Further we only highlight differences in the path establishment procedure of tinyLUNAR: the identity checking, the decision for RREQ propagation and the reaction of the destination nodes on the received RREQ messages. The identity checking procedure while being more complex than conventional matching of a single destination address is rather straightforward and follows a similar logic. With tinyLUNAR a node has a choice on the particular pattern to propagate the route request when its identity *does not match* the one in the RREQ message. In the simplest case it is a “blind” re-broadcast. However, the node may drop the not matched request if it does not satisfy the propagation parameters. For example if one of the target identities is a geographic coordinate and the *not_match_action* is geographic forwarding then the node will re-broadcast the request only if it is within a certain proximity metric to the target region.

As for the *matching action* the destination node will issue the route reply message only if the specified type of communication requires either joining a multicast group, sending acknowledgments (reliable routing) or a bi-directional unicast path. However when the *match_action* is REPORT the matching node may start sending the data immediately upon completed processing of the RREQ message.

In the forwarding plane tinyLUNAR provides a connection-less service. When upper layer components require connection-oriented communication this can be achieved by using port numbers in the protocol specific header. TinyLUNAR supports connection-oriented components by interfacing the *ConnID* field in the forwarding table to such components.

It remains to say that as is the case with LUNAR our protocol adopts the simplest route recovery procedure. It re-builds paths periodically from scratch. In tinyLUNAR the route expiration time is tailored to the specifics of relatively static sensor networks. We keep a route as active for several minutes, moreover the timer is shifted every time a data packet arrives on this route. In hierarchical

Table 1. Memory footprint of tinyLUNAR and tinyAODV

	Item	tinyLUNAR	tinyAODV
RAM	FIB (B)	56	133 (incl. cache)
	Supporting (B)	714	204
	Total (B)	770	337
ROM	Total (B)	1134	2760 (AODV_Core and AODV_fwd components)

networks we foresee that the route expiration timer is synchronized with a cluster head election protocol.

4.4 Implementation Details and Memory Footprint

We implemented tinyLUNAR in TinyOS v 2.x operating system. Note that currently TinyOS uses the type field of the IEEE 802.15.4 MAC header to indicate the communicating applications. In order to minimize changes to the core parts of the operating system we decided to encode the selector in the payload part of the packet for the proof-of-concept implementation.

The memory footprint of tinyLUNAR component and the reference numbers from the implementation of tinyAODV (the number of supported FT entries in both protocols is 7) are shown in Table 1. Note that tinyLUNAR consumes twice less ROM memory than its counterpart. With respect to the RAM consumption the total number for tinyLUNAR is higher, however the size of the FT table in memory is more than twice lower. The remaining RAM overhead in TinyLUNAR comes from the universality of the protocol. An additional gain in RAM can be achieved by further careful optimization of the implementation.

5 Related Work

The major representatives of routing protocols for wireless sensor networks are overviewed in Section 2. In this section we discuss the relation of tinyLUNAR to the existing approaches. The universality of the protocol through supporting both data-centric and address-centric communications uniquely positions our approach in the domain of WSN routing. The ports of almost all address-centric protocols from MANETs such as DSDV, AODV remain address-centric in WSNs. One exception from this rule could be a *possible* modification of DSR, where the forwarding plane (source routing) is separated from the routing plane (reactive route request). With relatively easy modifications the addressing concept presented here can be adapted to work with DSR as well. However, one clear advantage of tinyLUNAR is its ability to conduct multihop communications using only one byte overhead. Obviously, the source routing of DSR becomes quickly less efficient on paths larger than one hop.

As for the data-centric routing, the parametrized specification of the destination node(s) in tinyLUNAR allows to implement address-centric communications as a special case of data-centric ones. In principle, any existing data-centric

routing scheme can be adapted according the principles described in this paper. However, to the best of our knowledge we are unaware of such attempts. Furthermore, normally in the data-centric domain the routing issues are hidden behind a specific data-centric application. Typical examples of this are TinyDB [12] and Directed Diffusion [6], where the authors mainly focus on a systematic way of querying the information from the network and not on the routing issues.

6 Discussion and Future Developments

6.1 Addressing and Routing for WSN

While tinyLUNAR allows address-centric connections, a general question, however, is to which extend this communication paradigm is suitable for wireless sensor networks. In the original Internet architecture *addresses* indicate the location of the data, while *names* (e.g. URLs, email, etc.) are used to describe the communication parties [17]. In the context of the Internet, however, addresses were mainly introduced to hierarchically organize the network and to perform an efficient route lookup based on fixed-length bit sequences. Another property of addresses that justify their usage in the Internet is centralized control over their spatial distribution. The presence of names and addresses implied a two stage destination resolution process: Firstly the name is mapped to a destination address using a name resolution service and then the address is mapped to the forwarding path using a routing protocol. In a general class of *randomly deployed and large scale* wireless sensor networks, however, the control on global address distribution and the subsequent their centralized topological organization is rather infeasible task. In this case the Internet's purely address-based routing approach appears as redundant to WSNs. This also limits the usability of parts of the address-centric MANET routing protocols as an additional bandwidth and energy consuming directory service is required. TinyLUNAR on contrary follows an alternative communication model *routing by name*, appeared in the early time of the Internet [4] and recently revived and proved to be successful in the context of address self-configuration in MANETs [7].

6.2 Future Development

In general, the active message communication paradigm is a very powerful tool which increases the intelligence of the network. We conjecture that using this mechanism only for internal multiplexing between the communication components as it is currently implemented in TinyOS is suboptimal. The selectors used in tinyLUNAR play a twofold role. Firstly, they are used as forwarding labels; secondly, inside the end nodes they also indicate the application to which the packet must be multiplexed. Thus, having multiple functional roles tinyLUNAR selectors semantically better reflect the meaning of active message tags. In our future work we intend to move the tinyLUNAR functionality as close to the MAC layer as possible. In the case of TinyOS this would require modification of the *ActiveMessage* component.

The current implementation of tinyLUNAR includes a limited set of route request and reply actions. In particular, implementation of the RREQ propagation handler based on geographic coordinates, support for in-network processing and building the multicast and convergecast trees remain to be implemented. We leave the development of these issues for our future work. We also consider inserting some connection-oriented functionality in tinyLUNAR by encoding a limited number of ports in the selector field. By this we intend to further reduce the communication overhead for a selected class of connection-oriented applications.

7 Conclusions

In this paper we presented tinyLUNAR, a reactive routing protocol for wireless sensor networks. TinyLUNAR features the simplicity of its predecessor originally developed for mobile ad hoc networks. We showed that multihop forwarding in wireless sensor networks is feasible to implement using only one byte field of the IEEE 802.15.4 MAC header by adopting the label switching forwarding paradigm. The interfaces exposed by tinyLUNAR to upper-layer programmers allow flexible configuration of the protocol's behavior. One distinct feature which makes our protocol unique is its ability to build routes to parametrically specified destinations. With this property TinyLUNAR is capable to establish routes both for data-centric and address-centric communications.

References

1. Ács, G., Buttyán, L.: A taxonomy of routing protocols for wireless sensor networks. In: *Hiradastehnika*, December 2006 (2006), [Online]. Available: <http://www.hit.bme.hu/~buttyan/publications/AcsB06ht-en.pdf>
2. Deng, J., Han, R., Mishra, S.: INSENS: Intrusion-tolerant routing in wireless sensor networks. In: Department of Computer Science, University of Colorado, no. CU-CS-939-02, Boston, MA (2002)
3. Girao, J., Westhoff, D., Mykletun, E., Araki, T.: Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Elsevier Journal on Ad Hoc Networks* (2007)
4. Hauzeur, B.: A model for naming, addressing and routing. *ACM Transactions on Office Information Systems* (October 1986)
5. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. In: *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (2000)
6. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: a scalable and robust communication paradigm for sensor networks. In: *MOBICOM*, pp. 56–67 (2000), [Online]. Available: <http://doi.acm.org/10.1145/345910.345920>
7. Jelger, C., Tschudin, C.: Dynamic names and private address maps: complete self-configuration for manet. In: *ACM CoNEXT'06*, December 2006, ACM Press, New York (2006)
8. Johnson, D.: Routing in ad hoc networks of mobile hosts. In: *Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, U.S. (1994)

9. Karp, B., Kung, H.T.: Gpsr: Greedy perimeter stateless routing for wireless sensor networks. In: Proc. ACM MOBICOM, Boston, MA, August 2000 (2000)
10. Kuhn, F., Wattenhofer, R., Zollinger, A.: Worst-case optimal and average-case efficient geometric ad-hoc routing. In: Proc. 4th ACM International Conference on Mobile Computing and Networking, ACM press, New York (2003)
11. Li, Q., Aslam, J., Rus, D.: Hierarchical power-aware routing in sensor networks. In: Proc. DIMACS Workshop on Pervasive Networking (May 2001)
12. Maden, S.R., Franklin, M.J., Hellerstein, J.M.: TinyDB: An acquisitional query processing system for sensor networks. ACM Transactions on Data Base Systems (March 2005)
13. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental) (July 2003), [Online]. Available <http://www.ietf.org/rfc/rfc3561.txt>
14. Perkins, C.E., Bhagwat, P.: Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: SIGCOMM, pp. 234–244 (1994), [Online]. Available: <http://doi.acm.org/10.1145/190314.190336>
15. Ribeiro, C.: Robust sensor self-initialization: Whispering to avoid intruders. In: IEEE SECURWARE'07: International Conference on Emerging Security Information, Systems and Technologies, IEEE Computer Society Press, Los Alamitos (2007)
16. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), (January 2001), [Online]. Available <http://www.ietf.org/rfc/rfc3031.txt>
17. Shoch, J.: Inter-network naming, addressing, and routing. In: 17th IEEE Conference on Computer Communication Networks, IEEE Computer Society Press, Los Alamitos (1978)
18. Tschudin, C., Gold, R., Rensfelt, O., Wiblilng, O.: Lunar: a lightweight underlay network ad-hoc routing protocol and implementation. In: Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN'04) (2004)