Efficient Data Aggregation with CCNx in Wireless Sensor Networks

Torsten Teubler, Mohamed Ahmed M. Hail, and Horst Hellbrück

Lübeck University of Applied Sciences, Electrical Engineering and Computer Science,
Mönkhofer Weg 239, 23562 Lübeck, Germany
{teubler,hail,hellbrueck}@fh-luebeck.de
http://www.cosa.fh-luebeck.de

Abstract. CCNx is the reference implementation for a content centric networking (CCN) protocol developed by the Palo Alto Research Center CCNx group. It serves also as reference for our CCN-WSN, a CCNx implementation for wireless sensor networks (WSN). Efficient data aggregation with CCN-WSN is a challenge. In order to collect data from source in the network data sinks have to poll data sources with interests and exclude fields in interests are necessary bloating the interest messages. We solve the problem by introducing three building blocks in CCN-WSN: unicast faces for packet filtering and "link" abstraction, a forwarding service for creating network overlay structures used by applications and an intra-node protocol providing an API for applications to interact with the forwarding service. For evaluation purpose we implement an application using a forwarding service implementing a tree topology to collect data in the WSN.

Keywords: Content Centric Networking, Aggregation, Wireless Sensor Networks, Internet Integration, Future Internet.

1 Introduction

Wireless Sensor Networks (WSN) and Content Centric Networking (CCN) are two emerging technologies. WSN consist of spatially distributed often battery-driven autonomous sensors to monitor physical or environmental conditions. They gather real world data in applications like habitat or industrial monitoring, or agriculture. Data is processed in the network and aggregated information is provided to a gateway or a data sink within the network. Several WSN approaches use specialized routing and processing protocols which are different from today's protocols in the Internet. Recent approaches foresee even the use of IPv6 stacks [8] on sensor nodes.

The Internet Protocol IP addresses nodes via hierarchical addresses based on location. However, users are interested in content and information instead of the location of data. To overcome this issue content centric networking (CCN) approaches have been developed in the recent past. CCN addresses data itself instead of the location where data is stored.

T. Bauschert (Ed.): EUNICE 2013, LNCS 8115, pp. 209–220, 2013.

Users of WSN are also interested in content instead of the nodes that retrieve this content. Consequently, a content centric approach fits well in the WSN domain.

One emerging and advanced implementation in content centric networking is the CCNx protocol [4]. We have recently implemented CCN-WSN [12] a CCNx derivate which covers a subset of CCNx functionality.

In this paper we add the following contributions:

- We suggest unicast faces for wireless broadcast networks that enable packet filtering and provide a "link" abstraction.
- We propose a forwarding service for creation of overlay structures, that offers efficient forwarding.
- We develop an intra-node protocol as an API for applications to interact with forwarding services.

Section 2 discusses related work. Section 3 introduces fundamentals of CCNx. A problem statement for different request types is given in Section 4. Section 5 describes the concept followed by implementation details in Section 6. An example in Section 7 implements a forwarding service and an application. Section 8 concludes this work and discusses further topics of research.

2 Related Work

In this section we discuss related work in the area of routing, data exchange, and data aggregation in content centric networking (CCN) and wireless ad-hoc networks.

The goal of research of routing in CCN is efficient data delivery to requesters based on content names. The goal of [5] and [9] is to reduce the overhead of interest routing in CCN by adding different information to the interest message like frequency of node usage. Interest routing needs extra mechanisms to route interests to certain nodes (unicast), otherwise interests are flooded. Our approach uses a forwarding service and unicast faces to route the content efficiently.

[11] argue that Named Data Networking (NDN) forwarding is complicated and challenging. The main concept of this approach is to reach large scale flows for efficient forwarding in NDN. The authors do not consider data traffic and network load for the total network using this approach.

Neighborhood-Aware Interest Forwarding (NAIF) [10] achieves bandwidth reduction of interest flooding by forwarding only fractions of interest packets at eligible relays. A similar approach is introduced in [6] and [7]. The authors discuss the forwarding of Interest messages based on the hop distance from the previous forwarder to the destination. [10], [6] and [7] virtually forward interests, whereas our approach forwards content objects.

The authors in [2] propose an approach to support content search and content retrieve as a native process of the network. The authors indicate that only end user has the possibility to aggregate data in the network. In contrast to our approach, intermediate nodes can aggregate data. Additionally, this approach is

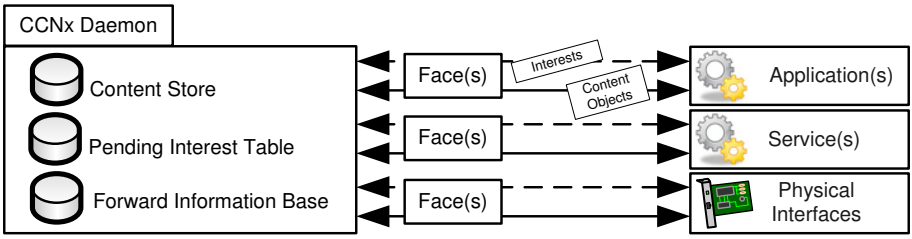


Fig. 1. CCNx Node Model

designed for PCs running Linux OS. Our approach is implemented on resource constraint devices in WSN like sensor nodes.

To the best of our knowledge, we are the first to propose a concept for data aggregation in WSN using CCNx implementation. Our approach is compatible to CCNx and therefore interoperable with existing CCNx implementations.

3 CCNx Overview

This section introduces to CCNx and highlights some features we use in this work. More detailed information is available on CCNx technical documentation web page [1] and [12].

Figure 1 illustrates the CCNx node model with the daemon and its main data structures *Content Store* (CS), *Pending Interest Table* (PIT), and *Forward Information Base* (FIB) on the left hand side. Applications, services, and physical interfaces (e.g. network interface cards) communicate via the CCNx daemon using *faces*. Faces in CCNx are a generalization of an interface comparable to TCP/IP sockets. In CCN-WSN the radio is a broadcast face that allows forwarding of interests and content within the network.

Message types in CCNx are *interests* and *content objects*. Interests request content objects by name.

CCNx follows a hierarchical naming scheme using the slash ("/") as name component delimiter. It is recommended that the first component is a registered unique DNS name to avoid name clashes among different content providing authorities. CCNx content names contain ASCII characters and bytes coded as two hexadecimal digits with a preceding percent character ("%"). A valid CCNx content name for example is: `/fh-luebeck.de/forward/%C1.de.fhl.tree~1~17~0.`

Interests are sent by applications or services. At interest reception, the daemon looks up the CS if it contains a content object matching the interest name. An interest name matches a content name, if the interest name equals the prefix of the content name and the prefix is the longest prefix among all compared content names. If no content object matches in the CS, the PIT is searched. If the PIT contains a matching interest, the incoming face of the recent interest is added to the list in this PIT entry. This allows forwarding incoming content

objects matching this interest to that face as well. If no entry in the PIT matches, the FIB is searched for a matching face to forward the interest. If no matching face is found the interest is discarded. Otherwise the interest is forwarded to this face and an entry is created in the PIT. One important feature of interests is that they can have an optional scope to control the interest dissemination. Scope 0 means that an interest forwarded only to the daemon, scope 1 forwards an interest to the applications/services, and scope 2 forwards an interest to the neighboring nodes. Interests have a lifetime which is set by the originator.

Content objects are sent only as a response to an interest. They are stored in a CS and arrive through faces generated by applications or services. On reception of a content object the CS is searched and if a matching content object is found the new content object is discarded to avoid unnecessary copies of duplicated content. If no entry in the CS matches, the PIT is searched. If a matching interest is found in the PIT the content object is forwarded to the list of faces of that interest and stored in the CS for the lifetime of the content object. As the interest is satisfied the corresponding entry is removed from the PIT.

4 Problem Statement

This section discusses the problem of data aggregation with standard CCNx and CCN-WSN forwarding mechanisms for two request types. We distinguish between *any* and *all* requests.

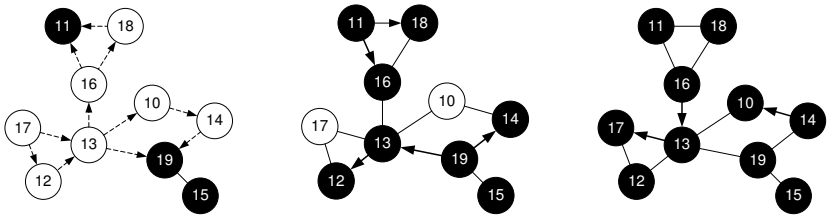
Any Requests. An *any request* is the standard request type in CCNx. A node requesting some content sends an interest and receives exactly one matching content object from any node providing this content.

This situation is depicted in Figure 2. Node 17 is interested in some content. Nodes holding content objects matching the interests name are depicted in black. In the first step node 11, 15 and 19 have the available content. Node 17 sends an interest through the network along the communication links (dashed arrows in Figure 2(a)). However, node 19 does not forward the interest because it is able satisfy it.

The nodes forward the content objects back to the face the interest arrived. E.g. content object from node 19 reaches node 13 (Figure 2(b)) and is forwarded to node 17 (Figure 2(c)). Content object providers add additional information e.g. node id as name suffix which allows the originator of the request to learn about the content provider.

As node 13 already stored and forwarded the content object, the content object from node 11 is either stored but not forwarded because there is no matching interest or discarded if it is a duplicate. This behavior of CCNx is intentional by design and reasonable if the content of the matching content objects is identical.

However, if we consider that sensor nodes observe e.g. temperatures and an application sends an interest for temperature this application retrieve a temperature value from any node. In standard CCNx behavior, one node “wins”. For



(a) Interest flooding on node 17

(b) Content forwarding on node 11 and 19

(c) Content from node 19 reaches node 17

Fig. 2. Any Request: Nodes with matching Content are marked in black

many applications any requests are not satisfying e.g. when an average temperature is requested. For this case we need another type of request.

All Request. In contrast to the previously described *any request* the result of an *all request* is data provided from all nodes that originate this data under a specific name. Examples of all requests may include a list of all measurements or aggregated values like *average*, *min* or *max* values.

Standard CCNx supports only any requests but it is possible to build all requests with CCNx *exclude* mechanism. Excludes are content names added as field to an interest to avoid these content names from matching this interest. To make content objects unique the CCNx daemon adds a digest (SHA-256 digest of the encoded content object) as last component of the content objects name when provided by an application. In our CCN-WSN application the daemon adds the node identifier as a name component. By this mechanism every content name is unique.

Invoking an *all request* using excludes works as follows: an *any request* is sent and a content object is received. Using the name of the received content object in an exclude in a further any request, a second content object is delivered different from the first one. All content objects can be fetched from the network with that name prefix by iterative sending of any requests using excludes.

This approach works well but has some obvious drawbacks. For every content object an interest has to be sent. Even if some content objects got stored in nearby nodes as a result of the first interest—like the content object from node 11 stored in node 13 in Figure 2(c)—this is a significant overhead. Furthermore, not all content objects come closer to the requester by iteratively sending interests, see content object from node 15 Figure 2(c)

Excludes occupy significant space in the message. This approach does not scale with the number of nodes in the network because messages have a limited length determined by the maximum transfer unit (MTU) of the physical layer.

These issues limit the use of CCNx with wireless resource constrained (especial energy constrained) devices. Sending takes a lot of energy and the MTU is very limited (e.g. 127 Byte with IEEE 802.15.4) in WSN.

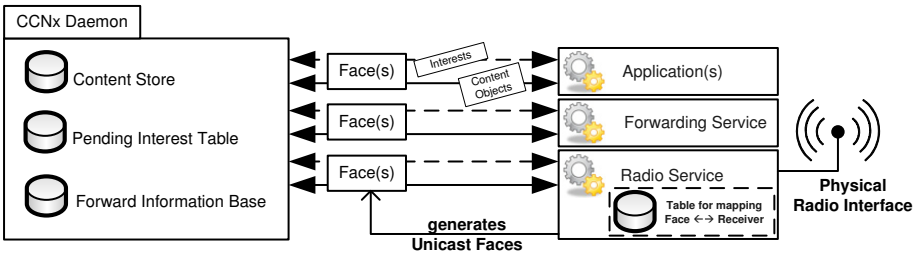


Fig. 3. Conceptual overview of our node model

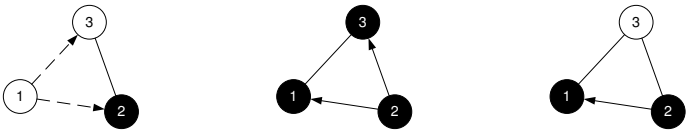
5 Concept

Our concept adds three main components to the CCNx-Model: *forwarding service*, *unicast faces*, *intra node protocol* as illustrated in Figure 3. This section introduces these components and their functionality.

Forwarding Service. The forwarding service builds an overlay network on top of the wireless broadcast network. This forwarding service provides efficient forwarding mechanisms for applications on top of existing CCNx forwarding mechanisms like the forwarding information base of the daemon. To save resources the overlay is constructed on demand of the application like a reactive routing protocol in ad-hoc networks. Furthermore, the forwarding service hands data to applications to allow aggregation of the data. Aggregated data from the application will be forwarded to the next node in the network by the forwarding service.

Unicast Faces. For WSN devices explicit faces based on TCP or UDP sockets across a network are not available like in the CCNx reference implementation. In wireless broadcast networks the radio is a broadcast face sending all interests and content objects to the neighborhood. Figure 4 illustrates this behavior for a single interest and content message exchange. In this scenario links between all nodes exist. In Figure 4(a) an interest with scope 2 is sent from node 1 to the neighbors node 2 and 3 (see dashed line) matching the content object stored on node 2. With the radio as broadcast face like illustrated in Figure 4(b) the content object is stored at node 3 as well because the content object matches the previously received interest from node 1. Node 3 will forward the content again to node 1.

With unicast faces in Figure 4(c) the content object is only stored and cached on node 1. Node 3 rejects the message because the destination address of the message of the content object is node 1. WSN benefit from reduced number of messages and less processing. Unicast faces are a basis for our forwarding service as inter node communication is now well under control and predictable.



(a) Interest propagation with Scope 2

(b) Using no unicast faces

(c) Using unicast faces

Fig. 4. Need of unicast faces with wireless ad-hoc links

Intra Node Protocol. The intra node protocol allows an inter process communication between services and applications using CCNx command marker technique. Interests with command marker (so called commands) of an intra node protocol have scopes smaller or equal to one to avoid interest propagation beyond the node and use a special name component *intra_node*.

Summarizing the concept, we foresee support for several applications e.g. one for each sensor and our approach is fully compatible with CCNx using only standard CCNx features. Therefore, it can be applied to all existing CCNx implementations. Implementation details are presented in the next section.

6 Implementation

In this section we describe the implementation of the unicast faces and the intra node protocol of our approach using CCNx. An example implementation of a forwarding service is given in Section 7.

Unicast Faces. Unicast faces are implemented as a part of the radio service depicted in Figure 3. The requirement is that the radio layer uses some node addressing scheme and radio packets include destination and source addresses. The service establishes a new unicast face automatically if a broadcast interest (destination is the broadcast address) from a neighboring node is received. The radio service maintains a mapping between this new face and the neighbor address. Unicast faces have a limited lifetime. When links break the resources on the node are released.

Intra Node Protocol. Communication between applications and services is performed via faces and the daemon. For our approach we suggest an intra-node protocol for communication with the forwarding service. Figure 5 shows a message exchange between *forwarding service*, *CCNx daemon*, and *aggregation application(s)* in a path time diagram. The intra-node communication is identified by the name component *intra_node*. In the example presented here the aggregation application communicates with the forwarding service. Dashed arrows depict interests, solid arrows depict content objects. Interests with scope 0 are sent to the daemon only, whereas interests with scope 1 are forwarded to applications or services on the local node. The first interest is sent with scope 1 and the second interest with scope 0 just to the daemon.

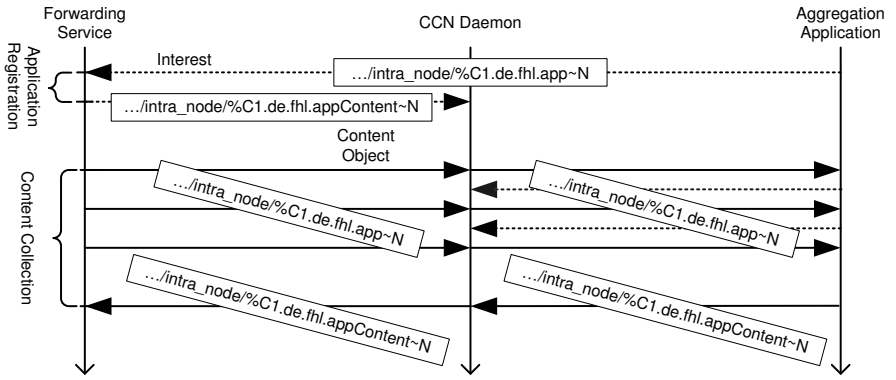


Fig. 5. Path Time Diagram of the Intra-node Protocol

At node startup the application registers at the forwarding service. This interest (`.../intra_node/%C1.de.fhl.app~N`) is implemented as a command denoted by the command marker `%C1` with one parameter `N` (delimited by a tilde character ("~")) identifying the application. Application identifiers need to be globally unique and the same for all applications in the network. When a forwarding service receives this interest it sends an interest (`.../intra_node/%C1.de.fhl.appContent~N`) also implemented as command with one argument (application identifier) and with scope 0 to receive content from the application. With the application identifier the forwarding service distinguishes between multiple applications running on a node. The lifetime of the two initial interests is as long as the predicted application lifetime.

If a content object is received from a neighboring node the forwarding service changes the name of this content object to same name of the interest sent from the application during the application registration to match even this interest. When the application receives a content object, it sends an interest with scope 0 to be able to receive further content objects from the forwarding service. This is necessary in the case where data from several nodes has to be collected for aggregation.

The application receives content objects from the forwarding service and processes the data. When the application finishes processing the data, it sends a content object matching the name of the interest sent during application registration by the forwarding service. An example for interaction of intra and inter node communication is provided in Section 7.

When the forwarding service receives a content object from the application it will then forward the content object with a new name on the network. As the interests from the application registration are satisfied the application registration has to be performed again.

7 Example of Data Aggregation

This section describes an implementation of a forwarding service as a proof of concept. The forwarding service named *tree forwarding* in this example constructs a directed tree. Along this tree data is forwarded by the service to the sink while it is collected and aggregated by the application. The application we implemented is a simple WSN node discovery which concatenates received lists (maybe empty) of node id's and adds its own node id.

Our example implementation has three phases. In phase zero the application registers as already described in Section 6. In phase one the tree is constructed. In phase two data is forwarded along the tree, collected and aggregated on the nodes. We describe phase one and two in detail in the following.

Phase One: Tree Construction. Figure 6 illustrates the steps for creating a tree graph according to [3]. Rectangles in Figure 6 aside the nodes show received interests. These interests are additionally stored in the tree forwarding service to make forwarding decisions later. In Figure 6(a) a *tree construction interest* (`.../%C1.de.fhl.tree~1~17~0`) is sent from node 17 because it is the sink in our example. The name follows the CCNx command message format with command name *tree* from namespace *de.fhl*. The first parameter is the application id ("1" in this example), the second parameter is the sender id and the third is a hop count. All tree construction interests are sent with scope 2 to reach the neighboring nodes only.

When the forwarding service receives a tree construction interest with a hop count smaller than its own interest it sends a new tree construction interest by adding its node id as second parameter and incrementing the hop count (Figure 6(b)). Newly received interests are highlighted in bold face in Figure 6. For simplicity we show only the parameters instead of the whole interests beginning from Figure 6(b). As the network in our example has three hops from the sink tree construction takes four steps. Step three is illustrated in Figure 6(c). Figure 6(d) shows the final result of the tree construction process before phase two starts.

Phase Two: Forwarding. The end of phase one is detected if no more interests are received by the nodes for a certain time.

In phase two, the forwarding service of a *leaf node* starts to send data. Leaf nodes can be identified as the forwarding service has not received any tree construction interest from neighbors with a higher hop count.

The tree forwarding service sends a content object without data via the intra-node protocol (see Section 6) to the application. The application itself sends a content object back to the forwarding service containing the current data (node id) using the intra node protocol.

Figure 7(b) shows an example where node 14 forwards data. At node 14, the forwarding service sends a content object named `.../intra_node/%C1.de.fhl.app~1` with empty data. The application sends back a content object named `.../intra_node/%C1.de.fhl.appContent~1`

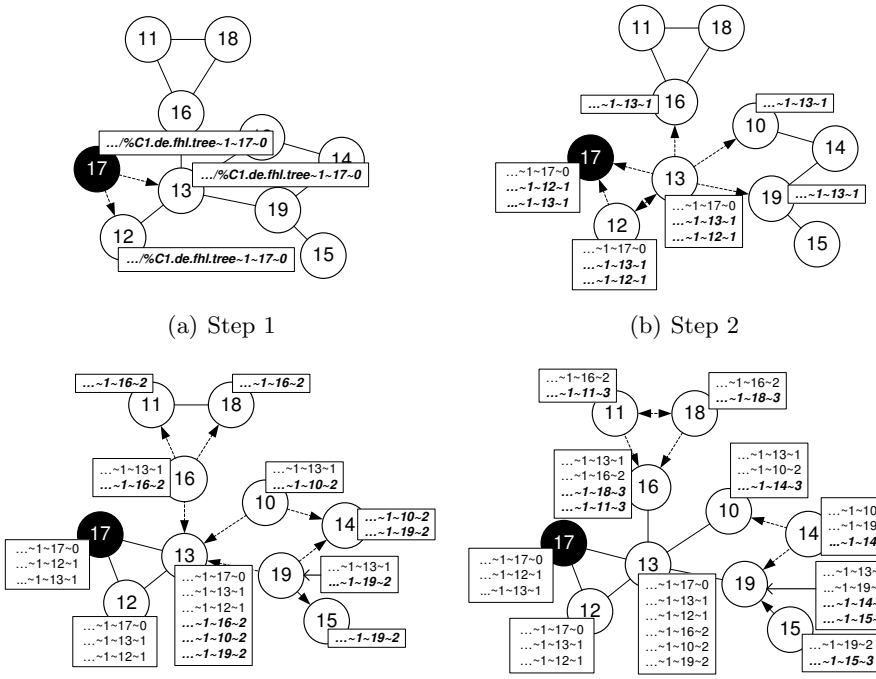


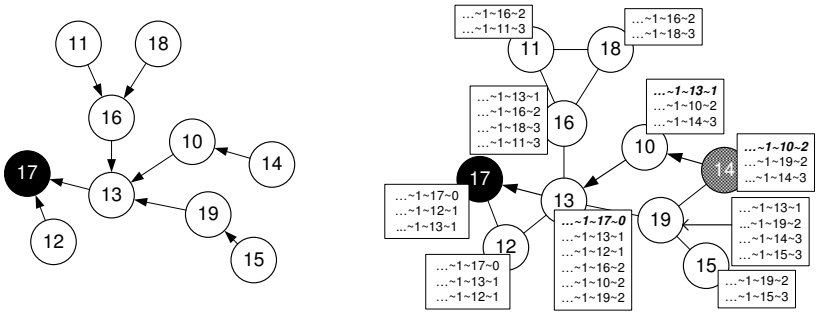
Fig. 6. Phase 1: Tree construction

containing data, in our example the node id 14. Content received from the application is forwarded by the forwarding service along the tree.

The forwarding service changes the name of the content object received by the application to `.../forward/%C1.de.fhl.tree~1~10~2` to forward the content to node 10. Node 10 is chosen because the interest from node 10 has a hop count less than the own hop count. The content object is forwarded on the unicast face the interest arrived. If several interests with a smaller hop count are available one of them is chosen.

The tree forwarding service on node 10 receives the content object. The received content objects name is changed to `.../intra_node/%C1.de.fhl.app~1` and handed to the application the same way as in node 14. At node 10 the aggregation application waits for content objects from all neighbors upwards the tree, aggregates the data and sends it to the tree forwarding service.

A special case occurs at node 19. Node 19 waits for data from node 14 and node 15. However node 14 forwarded data already to node 10. Therefore, intermediate nodes send their content object after a timeout even though they have not received the content object from all neighbors upwards the tree. Data is continuously forwarded until the sink is reached as shown in Figure 7(a).



(a) Tree graph (b) Forwarding on the tree

Fig. 7. Phase 2: Forwarding

When a content object is received the interest with the matching name is removed from the PIT. Therefore, the forwarding service resends the interest with scope 0 to be able to receive further content objects matching the interest. Resending these interests is important e.g. in Figure 7(b) node 13 will receive content objects from node 16 and 19 as well. As all interests have a certain lifetime the tree discontinues if no more content objects are sent.

In this section we presented tree forwarding as an example implementation of a forwarding service. As an example application we implemented a node discovery providing a list of all nodes available in the WSN.

8 Summary and Outlook

In this paper we have motivated the need for a further development of CCN-WSN to achieve efficient data aggregation in wireless sensor networks. Our approach extends CCN-WSN by three major components: unicast faces to reduce the number of message in a broadcast medium, forwarding service to create overlay structures on a given physical topology, and an intra-node protocol for communication between applications and the forwarding service. The daemon as the main component and the interfaces of CCN-WSN (and CCNx respectively) are unchanged in this approach. The communication between the components follows the basic mechanisms of CCNx. We have shown in an example implementation that a tree routing can be easily integrated with this approach.

In the future we will perform an evaluation against 6LoWPAN and CoAP. We foresee that using CCN straight away on top of IEEE 802.15.4 will provide an efficient solution with less overhead. Furthermore, we will add applications and semantic functionality to query wireless sensor networks. To improve the performance we will provide a compression scheme for the content names and implement persistent memory for content. Furthermore, we will add lightweight security measures in the implementation to prevent passive and active attacks against CCN and design and implement a gateway for seamless integration of CCN-WSN into CCNx implementation.

Acknowledgments. This work was funded by the Federal Ministry of Education & Research of the Federal Republic of Germany (17PNT017, DataCast). The authors alone are responsible for the content of this work.

References

1. CCNx Project: CCNx Technical Documentation (May 2013), `http://www.ccnx.org/releases/latest/doc/technical/`
2. Daras, P., Semertzidis, T., Makris, L., Strintzis, M.G.: Similarity Content Search in Content Centric Networks. In: Proceedings of the International Conference on Multimedia, MM 2010, pp. 775–778. ACM, New York (2010)
3. Garcia-Luna-Aceves, J., Spohn, M.: Source-Tree Routing in Wireless Networks. In: Proceedings of Seventh International Conference on Network Protocols, ICNP 1999, pp. 273–282 (1999)
4. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking Named Content. In: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT 2009, pp. 1–12. ACM, New York (2009)
5. Lee, G., Han, L., Park, Y., Lee, J.B., Kim, J., In, H.P.: An Energy-Efficient Routing Protocol for CCN-based MANETs. International Journal of Smart Home, 143–151 (2013)
6. Meisel, M., Pappas, V., Zhang, L.: Ad Hoc Networking via Named Data. In: Proceedings of the Fifth ACM International Workshop on Mobility in the Evolving Internet Architecture, MobiArch 2010, pp. 3–8. ACM, New York (2010)
7. Meisel, M., Pappas, V., Zhang, L.: Listen First, Broadcast Later: Topology-Agnostic Forwarding under High Dynamics. In: Annual Conference of International Technology Alliance in Network and Information Science (September 2010)
8. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D.: Transmission of IPv6 Packets over IEEE 802.15.4 Networks (RFC 4944) (September 2007)
9. Sun, L., Song, F., Yang, D., Qin, Y.: DHR-CCN, Distributed Hierarchical Routing for Content Centric Network. Journal of Internet Services and Information Security (JISIS) 3(1/2), 71–82 (2013)
10. Yu, Y.T., Dilmaghani, R.B., Calo, S., Sanadidi, M., Gerla, M.: Interest Propagation In Named Data MANETs. In: 2013 International Conference on Computing, Networking and Communications (ICNC), pp. 1118–1122 (2013)
11. Yuan, H., Song, T., Crowley, P.: Scalable NDN Forwarding: Concepts, Issues and Principles. In: 2012 21st International Conference on Computer Communications and Networks (ICCCN). pp. 1–9 (2012)
12. Zhong, R., Hail, M.A., Hellbrück, H.: CCN-WSN - a lightweight, flexible Content-Centric Networking Protocol for Wireless Sensor Networks. In: 2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (IEEE ISSNIP 2013), Melbourne, Australia (April 2013)