CCN Traffic Optimization for IoT

Jérôme François, Thibault Cholez, Thomas Engel

CCN Traffic Optimization for IoT

Jérôme François, Thibault Cholez, Thomas Engel

Interdisciplinary Centre for Security, Reliability & Trust - University of Luxembourg, {firstname.name}@uni.lu

***Abstract*—Content-Centric Networking (CCN) recently received a lot of attention thanks to its elegant way to optimize content diffusion at the scale of Internet. However, communications occurring at the edge of Internet, in particular the Internet of Things (IoT), are also a vivid research topic. Even if CCN was not initially designed to optimize the specific traffic pattern of the IoT, it can be improved to better support these new applications. In this paper, we propose to optimize the traffic within a CCN for IoT network where information is created and consumed at different frequencies. The simulations show that our solution outperforms the vanilla CCN architecture for this generic scenario.**

I. Introduction

Network protocols for the Internet-of-Things (IoT) are still a very active field of research. Even if 6LoWPAN/RPL [1] is currently well established, some disruptive research works [2], [3], [4] apply the new Information-Centric Networking (ICN) paradigm, designed to improve the core of the Internet, to other usages at the edge of the Internet, like the IoT. ICN is however not optimized by design for IoT communications which exhibit different properties (push-based, short-lived, small contents).

In this context, our approach optimizes the diffusion of information requested at different sampling rates and sent by sensors within the network using the Content-Centic Networking (CCN) architecture. More precisely, we describe how a node in the path between a data producer and some consumers can maintain states to transmit the produced upstream data toward the downstream links with a limited overhead.

The rest of the paper is organized as follows: Section II presents the advantages of using CCN for the IoT. Section III presents our optimization reducing the number of messages as well as its evaluation through simulations. Finally, we present the related work in Section IV and conclude in Section V.

II. CCN-based IoT

A. IoT requirements

The IoT enables smart services through the interconnection of things having computational resources. Smart applications usually rely on contextual information like the geographical location or other physical sensors. Therefore, a core component of the IoT are the sensors or, in a more general way, "information producers". WSNs (Wireless Sensor Networks) are usually connected to the Internet through gateways [5]. The information may be produced in two different modes: either on demand when another entity requests it, or proactively sent to multiple subscribers.

Similarly to WSNs, the IoT undergoes scalability and low power efficiency challenges. For efficient information in WSNs, two major routing approaches [6] can be identified: data-centric and hierarchical protocols. Hierarchical approaches, as for example [7], try to cluster nodes and to elect a leader serving as a kind of proxy. Data-centric routing, like [8], advocates for a name based routing where data follows the reverse path of the corresponding query by a hop by hop mechanism.

We promote the usage of CCN for IoT due to existing implementations for computers and smartphones, CCNx [9], but also for sensors running Contiki [3]. Moreover, CCN combines the advantage of the data-oriented approach and hierarchical routing, by leveraging a scalable hierarchical naming scheme, and integrates security mechanisms and caching by design.

B. CCN

CCN is well suited for distributed environments where each node plays a role in the routing decisions. A node interested in a given content (consumer) sends an *interest* request which is forwarded until another node can satisfy it (producer) by sending a *data* message using the reverse path. Each CCN node maintains three vital types of information:

- the Pending Interest Table (PIT) tracks the forwarded *interests* not yet satisfied,
- the Forward Information Base (FIB) maintains the next hop information used to forward *interests*,
- the Content Store (CS) indexes the content which is actually stored or cached on the local node.

In CCN, a face is anything serving as medium for transmitting and receiving messages. So, when a content c is requested by an interest i on a current node n, received from another node on the face f, the steps are as follows:

1) if c is available locally (c is in the CS), n sends it by a *data* message on the face f
2) if no match for c exists in the CS but exists in the PIT, this means that n already forwarded an interest for c. Hence, no message is emitted. If the new interest i is coming from a new face f, the latter is added to the list of faces associated to c in the PIT,
3) if no match exists in the PIT and in the CS, i is forwarded to another node using the FIB except if no route exists towards c, and the corresponding entry in created in the PIT.

The *data* message sent by the content provider is forwarded back hop-by-hop using the different PIT entries on the path as they contain the faces from where this content was requested.

More precisely, CCN leverages a hierarchical naming scheme such that *data* can only "satisfy"

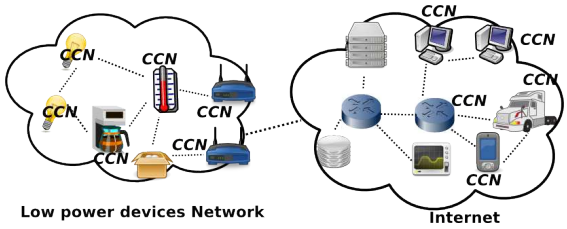


Fig. 1: A CCN enabled IoT Network

a specific *interest* whose content name is a prefix of the *data* message. For example, a content named `ccnx:/uni.lu/videos/intro.avi` defines several levels separated by `/`. To fill out the FIBs, nodes have to announce their content which is propagated to all other nodes with a limited overhead using aggregation. A typical example is the router of an organization which does not announce individual content but only a prefix like `ccnx:/uni.lu`.

C. Problem definition

A common IoT deployment is composed of communicating things in a dedicated network. Thanks to gateways, such a network is interconnected to the Internet. However, things tends also to be more and more directly connected, especially using cellular networks, like smartphones or connected cars.

We promote the use of CCN for low power devices. CCN considers every node as a router and so enhances direct neighbor to neighbor communications as in Figure 1. In addition, CCN is designed to be incrementally deployed beside IP which is necessary for traversing non-CCN paths.

However, this paper focuses on the low power devices at left hand in Figure 1. As highlighted before, CCN is well designed for distributed environments. Data-centric approaches are helpful for saving resources while the issue of a flat naming scheme is addressed with CCN. In addition, the mechanisms of CCN can route *interests* in parallel on multiple paths and faces (and so communication medium) which thus helps to improve the QoS (opportunistic routing). Besides, the meaningful naming scheme enables new context-aware routing features[10].

These low power networks are composed of information producers (like sensors) requested by low power devices (like actuators) and other Internet devices (servers, smartphones, etc). Therefore, a single information producer may have several information consumers desiring different granularities. As an example, a temperature sensor can send information to a fire detection system at a high frequency, every 10 seconds, but such an information provided only every 5 minutes is enough for automatically adjusting a heating system.

Therefore, the information should not be multi-casted to every consumer at the same regularity to limit the communication overhead. Sending information temperature every 10 seconds during five minutes represents an overhead of 29 useless messages which can be forwarded through multiple paths and so consume even more resources.

The general scenario is depicted in Figure 2 where there is an intermediate CCN node R_i which has to forward the information from M producers transmitted by a previous router R_{i-1} towards K destinations which are accessible through other downstream nodes like R_{i+1}. The upstream and downstream routers can be also other producers or consumers as highlighted before. To reduce the number of messages forwarded by R_i, we consider a sub problem qualified as local regarding one router and one producer. Resolving the more general problem with multiple producers consists in resolving multiple times the previous problem.

Formally, the problem is defined as follows. Assuming (1) an intermediate node, a router R_i, (2) one producer P able to send an information inf every x seconds, (3) N consumers $C = c_1, \ldots, c_N$ and (4) that each consumer c_i is interested to have inf every $x \times i_i$ seconds where $i_i \in \mathbb{N}$ (sampling period), the goal is to minimize the number of messages sent by R_i. Since each sensor node has a limited capacity, this optimization can only use a limited number of additional resources Cap_i. This assumes that a proportion of resources is already reserved for the common CCN operations.

III. Sampling optimization

A. Basic Forwarding strategies

In the following discussions, the targeted router for sampling optimization purposes is R_i and the considered producer is P. For sake of clarity, only one content is supposed for this producer, named $P_{content}$, but the approach can be easily applied in parallel to different contents.

By definition, *CCN* nodes are stateful and only forward *data* on the back path if an *interest* was emitted beforehand. Therefore, a basic usage of CCN requires that information consumers directly request producers as illustrated in figure 3(a). This strategy is called $pull$ and allows CCN to send data only when needed (on demand delivery) but the initial *interest* packet represents an overhead. However, an *interest* is not forwarded if a previous similar one has been already sent, and a given *data* message is sent only once per face which is then multi-casted by design. The producer also needs to initialize the FIB of the CCN nodes by announcing the hosted content. Such a step is not considered in the overhead computation because it is required only once.

Pushing data over CCN was envisioned in [2] which describes potential solutions and finally ends up designing a publish-subscribe mechanism where a node interesting in a certain content can subscribe for it, but in this case the diffusion is still based on IP. In this paper, we consider a strategy, $push$ where data is directly transmitted using the FIB. In fact, the subscription is done in the same way that the registration of contents populating the FIB and the sensors' data dissemination is similar to the propagation of interests. In particular, sensors'*data* are not cached and corresponds to a one-way message. To do so, a simple option is to disseminate such very small and ephemeral data directly inside interests like for example */roomA/temperature/ts=10/value=20* assuming that the temperature sensor has registered the content */roomA/temperature/*. Such an interest is forwarded using the FIB but without creating any entry in the PIT, since no *data* will be sent back. This can be easily done using a flag in the content name or in the *interest* header. However, a main advantage of creating an entry in the PIT, in the common usage

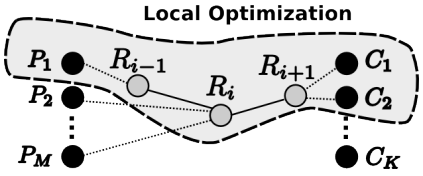
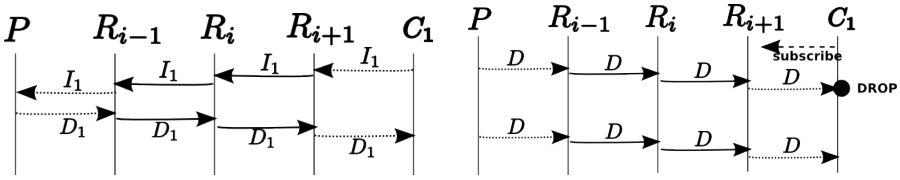


Fig. 2: Optimization problem



(a) On demand notification (pull)

(b) Subscription based mechanism (push) – similar to IP multicast

Fig. 3: Basic Forwarding scheme (I: *interest*, D: *data*) (Dotted lines represents possible multiple hops)

of CCN, is to avoid routing loops. Unlike [2] that uses standard IP unicast for avoiding routing loop, we propose to use a timestamp, *ts*, in the content name in conjunction with a new field in the FIB called *last_seen* in order to route only the latest content, which is the most valuable. Every new transmission for a given content will be checked against the *last_seen* value. More recent data will be forwarded while older obsolete ones will be discarded, thus avoiding loops and saving bandwidth.

Hence, we assume our aforementioned mechanism. This corresponds to *data* messages forwarded in Figure 3(b). However, since the information is transmitted regularly and independently of the demands, each value out of the sampling period (i_i) of a consumer c_i is dropped and represents an overhead of useless messages. Such a mechanism is comparable to use IP multi-cast where data is forwarded to all subscribers.

B. Optimal Forwarding strategy

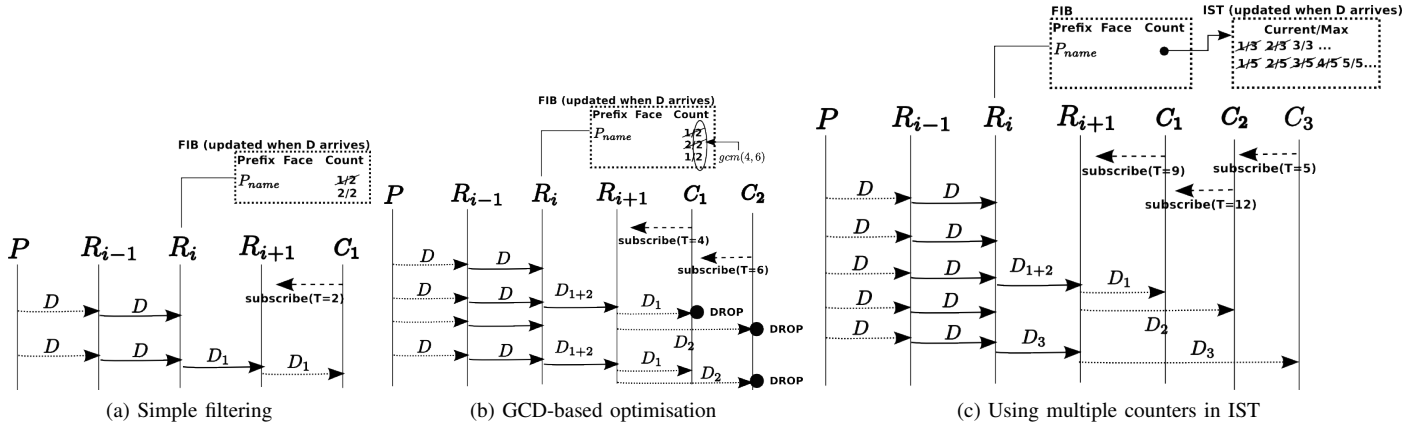
To reduce the message overhead in an optimal way, the goal is to combine the advantage of both pulling and pushing modes. The router should only forward data to C_i in pushing mode at the period t only if C_i has expressed its interest regarding this period: t is a multiple of the sampling period i_i. Hence, when the consumers subscribe, they have to specify their sampling period such that R_i can keep track of it thanks to a dedicated counter. This is shown in Figure 4(a) where C_1 subscribes to P_{name} whit $i_1 = 2$. As highlighted, the counter is directly inserted in the FIB and updated each time a message of the subscribed *data* arrives.

However, this optimal strategy needs one counter per subscription. The additional resources Cap_i are considered in terms of number of available counters, independently of the implementation. To limit the number of messages, identical messages are combined into one. Assuming $i_1 = 2$ and $i_2 = 3$, the sixth produced message is not forwarded twice as D_1 and D_2 but only once as D_{1+2} meaning that this data serves both C_1 and C_2. The different notations of the same message just help in identifying the purpose of the message. In fact, all these messages are exactly the same and will be forwarded to both consumers thanks to the propagation mechanism of CCN, which makes feasible the use of a single message D_{1+2}. This simple mechanism is also applied to the other strategies described in the next section.

The optimal strategy (one counter per content per consumer) cannot be applied with a fixed Cap_i value because the number of subscriptions can be very large and the sensors' resources are very limited. While not being practical, this strategy is considered as the baseline for evaluating the performances of the other ones.

C. Smart Forwarding strategies

Assumes a fixed maximal number of counters Cap_i, the goal of a smart forwarding strategy is to refine the pushing mode while limiting the number of forwarded messages. The worst case considers a single counter which serves to all subscribed periods. For example, with multiples like $i_1 = 2$ and $i_2 = 4$, a unique counter looping between 1 and 2 is



(a) Simple filtering

(b) GCD-based optimisation

(c) Using multiple counters in IST

Fig. 4: Sampling optimization (Dotted lines represents possible multiple hops)

enough and does not entail additional messages. This can be extended using the Greatest Common Divisor (GCD) (strategy gcd). With $i_1 = 4$ and $i_2 = 6$ as in Figure 4(b), the unique counter loops over $gcd(4, 6) = 2$ which thus avoids to forward message at every period. However, there is an overhead of useless messages like the first forwarded message D_{1+2} which is then transferred by other routers as D_1 or D_2 before being dropped by the consumers. Moreover, there is a high risk that with numerous subscriptions, the global GCD reaches 1 which so does not represent any improvement compared to the basic pushing forwarding scheme.

With more counters, *i.e.* $Cap_i > 1$, this can be extended by selecting Cap_i values, $G = \{g_1, \ldots, g_{Cap_i}\}$, such that:

$$\forall C_i \in C, \exists g_k \in G, \alpha \in \mathbb{N}, g_k \times \alpha = i_i$$

Figure 4(c) considers two counters and so introduces a specific structure, the Information Sampling Table (IST), which keeps track of them. With three consumers and $i_1 = 9$, $i_2 = 12$ and $i_3 = 5$, R_i will forward every 3 and 5 *data* messages. Again, this will entail some message overhead but it is better than having considered the unique GCD which is one.

After having filtered identical intervals and those being multiple of others, the smart forwarding strategy affects one counter per unique remaining sampling interval $G' = \{g_1, \ldots, g_M\}$. If the number of needed counters is acceptable, *i.e.* $|G'| \leq Cap_i$, the algorithm stops and $G = G'$. Otherwise it iterates until this condition is reached by merging at each step two counters, $g\_merge = gcd(g_i, g_j), (g_i, g_j) \in G' \times G', g_i \neq g_j$. These counters are selected to maximize the GCD in order to limit the message overhead:

$$\forall (g_k, g_l) \in G' \times G', (g_k, g_l) \neq (g_i, g_j), gcd(g_j, g_i) \geq gcd(g_k, g_l)$$

D. Evaluation

For evaluation purposes, a dedicated simulator has been developed which implements the only necessary functions of CCN which are used in our approach. The number msg_s of messages forwarded by R_i is considered regarding the chosen strategy s (among $pull$, $push$, $optimal$, gcd or $smart$) and a configuration defined by:

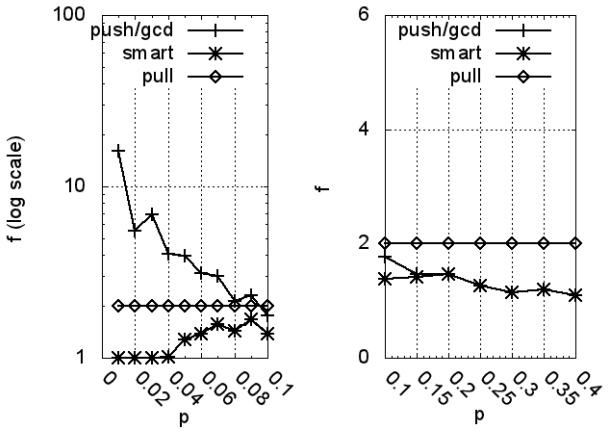
- MAX_T: the maximal length of the sampling interval that a consumer can subscribe to,
- $N = |C|$: the number of consumers which subscribes to a unique sampling interval. The latter is chosen randomly between 1 and MAX_T,
- $\#counters$: the number of available counters (only impacts the smart forwarding strategy),
- $\#steps$ is the number of simulation steps assuming that the length of a step is defined by the initial sampling period at which the sensor can produce data.

We measured the number of messages because it is clearly correlated to the energy consumption which is one of the main criteria for designing protocols for WSNs. Based on preliminary experiments, $\#steps$ was fixed to $5 \times MAX\_T$, which guarantees that every consumer has received data at least five times during the discrete events simulation. As previously said, the optimal strategy is used as the baseline to compute an overhead ratio defined as:

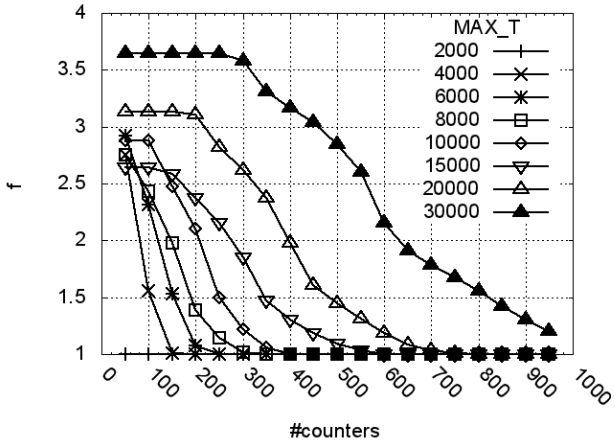
$$f_s = \frac{msg_s}{msg_{optimal}} \quad (1)$$

We always calculate the average of f over 10 experiments. The number of consumers affects our performances by increasing the number of different sampling periods and is therefore expressed as a proportion p of MAX_T.

The number of consumers is evaluated in figure 5(a) where $MAX\_T = 3000$ and $\#counters = 100$. The vanilla CCN architecture ($pull$) highlights a ratio of 2 which is coherent since every *data* is sent after an initial *interest*. When p increases, all other curves tend to one because the optimal strategy also requires to forward more messages as there are more customers. In particular, when p increases, the chance to have a consumer C_i having subscribed with an interval $i_i = 1$ is higher which thus leads to forward a message at every step. Due to the same mechanism, the benefit of smart forwarding compare to the push mode is reduced. Therefore, using optimization has logically a higher impact when the number of subscribed intervals is low. However, the worst case for smart routing is for $p = 0.08$ representing 240



(a) Impact of the number of consumers ($MAX\_T = 3000$)



(b) Impact of the number of counters (smart forwarding, $p = 0.05 \times MAX\_T$)

Fig. 5: Message overhead evaluation

distinct intervals which are handled by the 100 counters. As highlighted, the gcd strategy leads to the same result that the simple $push$ strategy which forwards every *data* message because the overall GCD computation reaches always 1 even with $p = 0.05$ due to the probability to have randomly selected one of the 430 prime numbers between 1 and 3000. Besides, as previously highlighted, the *push* mechanism is similar to IP multicast, which thus allows to compare our approach with the latter.

The smart forwarding strategy proposed in this paper always exhibits a ratio significantly lower than the classical CCN approach. It varies between 1 and 1.67. When p is very small, there are enough counters to be optimal and starting at $p = 4\%$, the optimization starts by sharing the counters for multiple intervals which entails a message overhead. Around 0.1, the optimal strategy has to forward many messages which thus highly impacts on the denominator in equation (1).

In figure 5(b), the number of counters for the smart forwarding strategy varies on x-axis and different maximal sampling intervals have been tested. The number of uniquely subscribed intervals is set to 5% of MAX_T and so varies between 100 and 1500 depending on the configuration. When the number of usable counter increases, the overhead is logically reduced, even faster if there are less subscribed intervals (lower MAX_T). Such an experiment helps to choose the right trade-off between message overhead and hardware/software additional resources (number of counters). Then, depending on the communication medium and the hardware, it would be possible possible to select the most efficient number of counters in terms of energy consumption. For instance, assuming that the maximal acceptable overhead is expressed as $f = 2$, the proper number of counters can be chosen regarding the number of uniquely subscribed intervals.

IV. Related work

As mentioned, an important missing feature to use CCN for IoT is a push-based mechanism for communications. Instead of probing a sensor regularly, which consumes energy, a better way would be to let the sensor transmit its data when needed. Doing this in CCN is not natural but yet feasible. Van Jabobson *et al.* solved a similar issue to enable CCN-based conversational communications for voice over IP [11]. They propose a solution based on two properties of CCN: on-demand publishing to request content that has not been published yet, which simulates a rendez-vous point, and a deterministic algorithm followed by the two parties to find this service name. The approach in [2] handles small ephemeral messages, more precisely events communication typically produced by IoT. This allows CCN to support publish/subscribe communications by adding a new unicast forwarding table and a new set of dedicated messages: "one way" messages for event notifications that can follow the path of subscriptions up to the subscribers. In [3], a full CCN layer is provided for Contiki which enables CCN usage for low power devices and IoT.

Our paper also deals with the problem of efficient dissemination of event notifications in a sensor network. Some papers are close to this topic even if they do not optimize the diffusion of information according to fixed sampling rates. For example, the authors in [12] used non-linear optimization theory to optimize the rate at which sensors can be probed by a control system under network limitations. Another way to save network resources in WSNs is to rely on in-network aggregation techniques which have been largely investigated through different strategies [13], even secured ones [14]. This solution is particularly efficient when data is gathered by a few sinks while we envision in our case a more neighbor-to-neighbor communication pattern between sensors.

V. Conclusion

In this paper, CCN is leveraged for distributed IoT communications, especially in the context of producers regularly updating information sent to multiple consumers. A new push-based mechanism was designed and optimized. Even under strict resource constraints, the nodes optimize the forwarding of messages based on the update frequencies required by the different consumers. Compared to the optimal but unrealistic approach, the results show a limited overhead, always under the regular CCN *pull* approach. Our future work will focus on improving our approach for multiple-content producers.

Acknowledgement: This work was partly funded by BUTLER and IoT6 FP7 EU projects under the grant agreements 287901 and 288445.

References

[1] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of ipv6 packets over ieee 802.15.4 networks," 2007. [Online]. Available: http://tools.ietf.org/html/rfc4944

[2] A. Carzaniga, M. Papalini, and A. L. Wolf, "Content-based publish/subscribe networking and information-centric networking," in *ACM SIGCOMM workshop on Information-centric networking – ICN*, 2011.

[3] B. Saadallah, A. Lahmadi, and O. Festor, "CCNx for Contiki: implementation details," INRIA, Tech. Report RT-0432, 2012.

[4] S. Y. Oh, D. Lau, and M. Gerla, "Content centric networking in tactical and emergency MANETs," in *Wireless Days (WD), 2010 IFIP*. IEEE, Oct. 2010.

[5] L. Steenkamp, S. Kaplan, and R. Wilkinson, in *AFRICON*, 2009.

[6] A. Boukerche, M. Z. Ahmad, D. Turgut, and B. Turgut, *A Taxonomy of Routing Protocols in Sensor Networks*. John Wiley & Sons, Inc., 2008, pp. 129–160.

[7] S. Soro and W. B. Heinzelman, "Cluster head election techniques for coverage preservation in wireless sensor networks," *Ad Hoc Netw.*, vol. 7, no. 5, pp. 955–972, Jul. 2009.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Conference on Mobile computing and networking*. ACM, 2000.

[9] Content Centric Networking. [Online]. Available: http://www.ccnx.org

[10] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *International conference on Emerging networking experiments and technologies – CoNEXT*. ACM, 2009.

[11] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, "VoCCN: voice-over content-centric networks," in *Workshop on Re-architecting the internet – ReArch*. ACM, 2009.

[12] J. Mao, Z. Wu, X. Wu, and S. Wang, "Sampling frequency optimization in wireless sensor network-based control system," in *International conference on Advanced Web and Network Technologies, and Applications – APWeb*. Springer, 2006.

[13] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *Wireless Communications, IEEE*, vol. 14, no. 2, pp. 70–87, 2007.

[14] L. Yu, J. Li, S. Cheng, and S. Xiong, "Secure continuous aggregation via sampling-based verification in wireless sensor networks," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 1763–1771.