Downloaded from http://iranpaper.ir

Computer Communications xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Content centric routing in IoT networks and its integration in RPL

Q1 Yichao Jin[a,*], Sedat Gormus[b,1], Parag Kulkarni[a], Mahesh Sooriyabandara[a]

[a] Toshiba Telecommunications Research Laboratory, 32 Queen Square, Bristol BS1 4ND, United Kingdom
[b] Karadeniz Technical University, Trabzon, Turkey

ARTICLE INFO

Article history:
Available online xxx

Keywords:
IoT
Routing
Content centric
Data aggregation
Implementation

ABSTRACT

Internet of Things (IoT) networks can be used for many applications across different industry domains including infrastructure monitoring, civil service, security and surveillance applications etc. However, gathering large amounts of data from such networks including images and videos often cause traffic congestion in the central network area. In order to solve this problem, we proposed the content centric routing (CCR) technology, where routing paths are determined by content. By routing the correlated data to intermediate relay nodes for processing, a higher data aggregation ratio can be obtained, hence effectively reducing the traffic in the network. As a result, significant latency reduction can be achieved. Moreover, redundant data transmissions can also be eliminated after data aggregation which reduces the energy consumption spent predominantly on wireless communication thereby conserving limited battery. CCR was further integrated with the IETF RPL protocol and implemented in Contiki OS using the TelosB platform. Finally, both simulation and implementation results prove the superior performance of CCR in terms of low network latency, high energy efficiency, and high reliability.

© 2016 Published by Elsevier B.V.

1. Introduction

Distributed computing in wireless networks has recently been attracting a lot of attention, especially in the emerging paradigm of the Internet of Things (IoT) communications where IoT devices are equipped with independent processing, communication, and storage capabilities [1]. The key idea is that rather than sending all raw data directly across an expensive (multi-hop) wireless network which is usually correlated with high energy consumption and time delays, a more cost-effective way is to first reduce the data volume locally via in-network processing and subsequently forward only the processed result. Therefore, we can save bandwidth and energy, reduce latency and extend the network life in resource constrained IoT network [2].

In many cases, data collected for the same application tends to be highly correlated [3] and therefore can be combined or jointly processed while forwarding to the sink. For example, fusing together multiple sensor readings related to the same physical event. Such *data aggregation* process can reduce the total amount of messages to be sent over expensive wireless links, which has a significant impact on energy consumption as well as overall network efficiency. On the other hand, uncorrelated packets might not be simply aggregated from the processing point of view, e.g. it is not meaningful to calculate an average of a temperature and a humidity reading. Therefore, one critical issue in data aggregation is to determine an optimized information flow and communication topology in order to efficiently route the correlated data to the intended processing nodes in the network. Let's take the tunnel monitoring system as an example to give more insight. As shown in Fig. 1, a variety of sensor nodes and cameras are installed to monitor two key tunnel assessments: tunnel structure safety and traffic management, where a huge amount of real-time sensory data including images and video streams needs to be delivered to a remote control centee. Traditionally, the server first collects all the data via the same routing topology regardless of whether the data is used for tunnel safety or traffic management. This case is illustrated in Fig. 1(A). Take Node 1 for example, it sends both Tunnel safety data A and traffic data B to node 5 as they are treated as the same. Once all data reaches the destination, the final results are computed at the server side. However, this is very likely to create a 'hot-spot' problem, where heavy network traffic in the central area results in higher energy consumption on the neck nodes and is also prone to traffic congestion events. This is due to the fact that the neck nodes are geographically closer to the access point/server, thus they have to forward messages coming from the outer regions (Fig. 1(A)).

Q2 * Corresponding author. Tel.: +44 1179060780.
E-mail addresses: yichao.jin@toshiba-trel.com, yichao.jin@hotmail.com (Y. Jin), sedatgormus@ktu.edu.tr (S. Gormus), parag.kulkarni@toshiba-trel.com (P. Kulkarni), mahesh@toshiba-trel.com (M. Sooriyabandara).

[1] This work was carried out when the author was with Toshiba Research Europe Ltd.

http://dx.doi.org/10.1016/j.comcom.2016.03.005
0140-3664/© 2016 Published by Elsevier B.V.

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

UNCORRECTED PROOF

Downloaded from http://iranpaper.ir

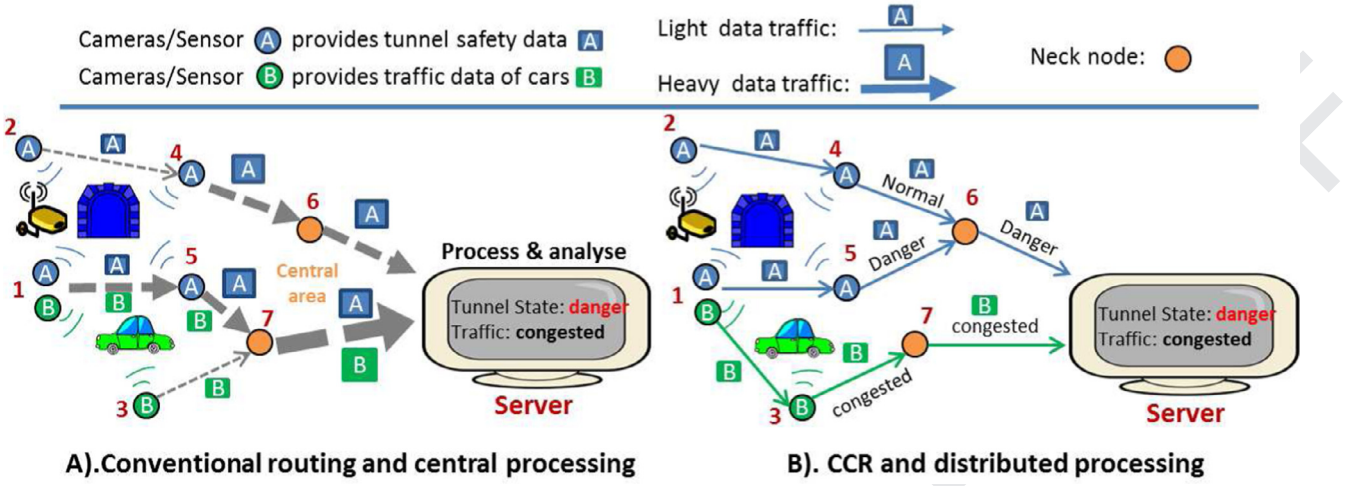


Fig. 1. Conventional routing vs. CCR.

To overcome such problem, we proposed a content centric routing (CCR) protocol in [4] for efficient data aggregation in multi-hop IoT networks. We differentiate data by its content while routing information, and correlated data are termed as the data of the same content. As shown in Fig. 1(B), since both Node 1 and Node 3 provide information for traffic conditions, rather than sending content B to Node 5 as shown in Fig. 1(A), Node 1 sends it to Node 3 where they can be combined or aggregated while forwarding to the server. Intermediate results such as heavy traffic warnings can be computed within the network. As a result, two distinctive routing topologies based on content A and B are created in CCR. This can help to reduce the amount of redundant data sent over the network and also the time lag in the communication system, saving limited node energy and extending the network lifetime.

CCR provides a paradigm shift from traditional ways of data collection to content oriented data aggregation and retrieval. This change could bring several attractive advantages such as energy efficiency, fast system response, long network lifetime etc. and provides a way to solve the data explosion problem [5] for the future IoT network. In [4], we proposed a multi-objective function to provide optimized in-network data aggregation with the aim of reducing latency, load balancing and extending the network lifetime. Using which, each node can refine its routing strategy according to neighboring traffic patterns and the energy availability of the neighboring nodes. In addition, a routing candidate selection mechanism was developed in order to avoid communication loops, and the signaling cost of local message gossiping is controlled to conserve limited node energy and resources. With respect to our previous work, in this paper, we further extend the analysis and protocol explanations in [4], including updated frameworks, signaling and control message details, and trigger function flows etc. Furthermore, we presented a possible integration of CCR in a real industrial standard (the IETF RPL protocol [6]). The implementation is based on the Contiki OS [2] and TelosB platform. Last but not least, CCR is evaluated by both simulation and implementation experiments. To the best of our knowledge, this paper is the first to present how to implement a content based routing protocol for in-network data aggregation.

The rest of the paper is organized as follows. Section 2 covers the related work. In Section 3, we present our models and assumptions following which we present the CCR protocol in Section 4. The CCR implementation and integration with RPL is illustrated in Section 5. The efficacy of the design is illustrated in Section 6 with both simulation and emulation results. The paper finally concludes in Section 7 highlighting some of the key findings.

2. Related work

Different from recent popular notion of Content-Centric Networking (CCN) or Named Data Networks (NDN) [8]–[10] which has a aim of caching and subscribing data based on content rather than the host. The main focus of the proposed CCR technology is to provide optimized routing topology to facilitate in-network data aggregation and reduce the network traffic.

Routing and data aggregation mechanisms have received considerable attention in the literature [3], [11] in the context of wireless sensor networks. The existing body of work can be broadly classified into two categories – centralized and distributed approaches. Centralized approaches [12]–[17] usually pre-compute and construct the optimal appropriate routing structure before the network starts to operate. In [13], a network lifetime maximum aggregation tree solution is proposed. Load balancing is considered in [15], and authors in [17] further took the aggregation computational cost into account. However, global network information is often required for above literatures which can introduce significant control overhead. In order to reduce control overhead, distributed clustering approaches such as [18]–[24] resort to forming hierarchical routing topologies via local message gossiping. However, only a simple shortest path tree topology is adopted in [22]. In [23], a dynamic clustering based approach is proposed. However, the clustering process is triggered per event or application, resulting in large transmission cost in forming the clusters. In addition, similar to the clustering approach, tree ([13], [14], [25]) or Direct Acyclic Graph (DAG)[26] based approaches also require a specific routing topology to operate, and hence limits their abilities to cope with dynamic network conditions. This is because each time a network change happens such as link breakage or early energy depletion of some critical routing nodes, the network topology information needs to be updated to reflect the prevailing conditions. This, however, has the cost of introducing additional control traffic to the network as well as incurs additional delays.

[2] Contiki is an operating system to network embedded devices. It is highly portable and can be ported to more than 12 different microprocessor and microcontroller architectures. [7]

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

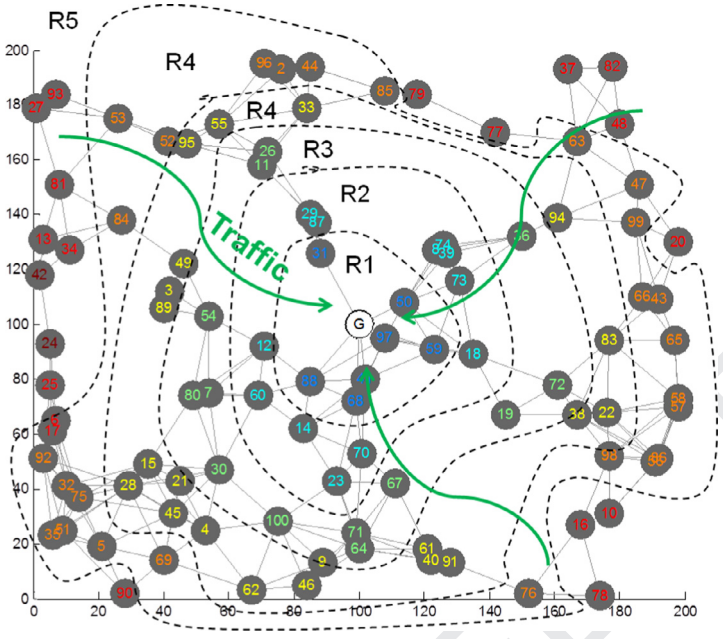


Fig. 2. Hop distance based ring topology [4].

A key shortcoming of existing solutions is that they often assume homogeneous network conditions, for example: homogenous traffic or universal node processing capability. However, traffic is dynamic in nature. Each time the traffic of an application changes or a new application arrives, the optimized network structure should, ideally, also be re-formed. In a dynamic environment with multiple applications co-existing, different data aggregation paths are required for efficient delivery of different types of data and better organization of heterogeneous traffic flows. A pre-optimized static structure cannot satisfy the requirements in such a scenario. On the other hand, it is not computationally efficient to frequently reconstruct a global network topology or to compute and build multiple overlaid topologies because such approach would be expensive to maintain in lossy environments.

On a related note, a perfect channel condition is also usually assumed [4], [21], [27], which may not always be the case in the real world as communication link quality can vary over time. Such an assumption can jeopardize the delivery of a message and can potentially result in re-transmissions which result in further energy depletion. Routing packets based on link quality and connectivity can improve communication reliability [6] but it does not necessarily lead to energy efficient routing. Thus, in addition to improving communication reliability, conserving the limited on-board energy of the battery powered nodes is an important requirement in order to keep the nodes alive and running in such resource constrained networks.

To overcome some of the problems above, the CCR algorithm [4] was introduced to provide content based information flow and optimized in-network data aggregation with the aim of avoiding the transmission of redundant network traffic, reducing network delay, conserving limited energy resource and extending the network lifetime. Furthermore, compared with our previous work [4], we provide analysis on a possible integration of CCR in a real IoT protocol stack. CCR is modified such that it can be integrated with the RPL protocol and show its compatibility with a standard based protocol stack. Its effectiveness is evaluated via both Matlab simulation and more practical Contiki Cooja based emulation. A small scale CCR implementation Demo based on real hardware (TelosB mote) was developed and demonstrated on various occasions.

3. System models and assumptions

3.1. Network model

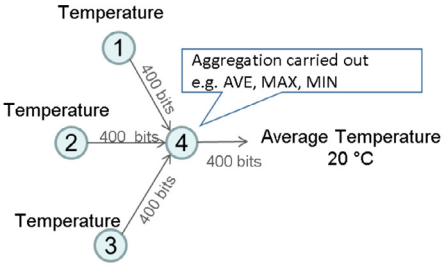
We assume that nodes send data to a gateway node residing at the centre of the topology. The gateway node is much more capable (e.g. in terms of processing power, memory etc.) than the individual nodes themselves and has access to mains power. Nodes are battery powered and have a finite and heterogeneous initial energy supply $E(i)$. Transmission power control is not enabled and therefore all nodes have a fixed communication range. We also assume heterogeneous node processing capability, e.g., a node may only be capable of processing one or a few particular types of content due to hardware or software constraints. In case a node receives a data packet that it cannot process, it simply relays the packet. To begin with, the gateway broadcasts a radio ranging message to help the nodes ascertain how many hops away are they located from the gateway. Nodes that receive this message are assigned with a layer ID. This layer ID represents the minimum hop distance between a node and the gateway. Subsequent to assignment of the layer ID, the node forwards this message with its own layer ID included in this message. Such a wave like propagation, results in the formation of a ring type of topology as shown in Fig. 2.

3.2. Application and aggregation mode

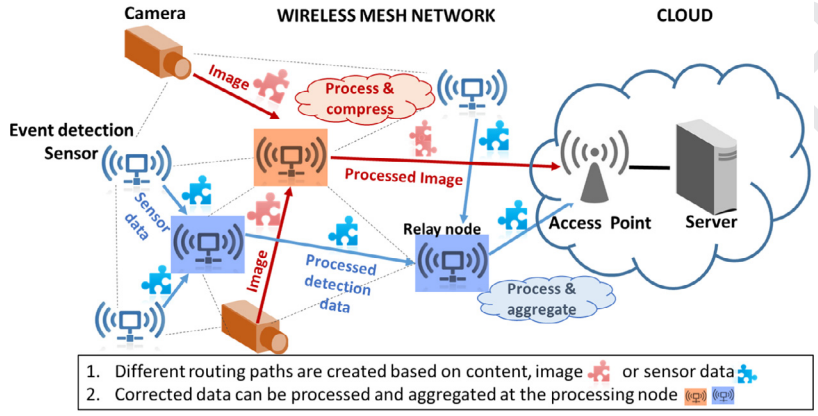
We mainly consider monitoring applications (multi-point to point data collection scenario) in this paper for data aggregation purpose. All data packets associated to the same processing objective are termed as the packet of the same content, which then can be processed by a corresponding processing node. For example, same type of data (temperature readings) gathered in a building can be treated as the same content if an application requires the average building temperature. For simplicity, we assume that each application running in the network only has a single processing objective, but multiple applications can co-exist. A total number of K applications $A = \{a_k \mid k = 1, 2, 3, \ldots\}$ have the same poisson arrival rate of λ, but with different running duration of $T = \{t_k \mid k = 1, 2, 3, \ldots\}$ and heterogeneous traffic traffic data rates R_k. Depending on the accuracy requirement of an application,

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir



(a) Simple data aggregation model



(b) Sophisticated in-network processing

Fig. 3. Aggregation model examples for different applications.

different data aggregation functions can be applied considering lossy or lossless aggregation processes [28]. For each node i with processing function s, a generic data aggregation model is defined below:

$$R_i^{out} = \omega_s \times R_i^{in}$$
$$0 < \omega_s \leq 1 \quad (1)$$

where R_i^{in} and R_i^{out} represent the incoming and outgoing traffic rate, respectively. ω_s is the data aggregation or compression rate depends on the processing function s. If $\omega_s = 1$, it means that the current content cannot be processed by s. ω_s can also be a variable depending on the processing function. For example, there may be considerable correlation of data streams comprising data reports of AVERAGE or MAX readings for monitoring applications, which can aggregate multiple incoming messages into a single outgoing message. In such cases, depending on the total received message number (assuming M) on an aggregation node, ω_s could be $\frac{1}{M}$. Nevertheless, we assume only messages from the same application can be aggregated. For reasons, different data types may not be easily processed or just not possible to do so in some cases. For example, it is not meaningful to calculate the average value of a temperature and a humidity reading. An example of data aggregation can be found in Fig. 3.

In order to have a good data aggregation opportunity, it is assumed that aggregation is carried out in a periodic manner at each hop, i.e. each node waits for a pre-defined period of time to gather information [29] and then performs the aggregation. A timeout clock is used in case some packets get lost during transmission.

4. The proposed CCR protocol

CCR is a distributed process, when an application arrives at the gateway following which a default routing structure is first used to initiate data collection. The focus of the subsequent phases is about optimizing this routing structure. Each node has a probability p_t to refine its next hop relay by executing an objective function F. The node that intends to execute F (referred to as an objective node hereafter), first broadcasts a local query message to its one-hop neighboring nodes. The query message comprises the objective node's outgoing traffic content types and corresponding traffic volume, and the candidate selection criterion (TTGF bits described in Section 4.3). The qualified next hop candidate will then respond to it with an *ACK* message, which consists the information required by F such as the responder's ID and the estimated node lifetime of the responder if the designated traffic was sent to that candidate node. Using this information as the input of the objective function, candidate rankings are produced and the one with the highest ranking is selected to relay the corresponding traffic. Finally, the objective node updates the routing table and broadcasts a route update announcement message containing new next hop node ID for corresponding traffic content. Subsequently, the new next hop nodes reply with *JOIN ACK* messages and the previous relay nodes send *LEAVE ACK* messages. As a result of this process, an overlaid tree topology for multiple traffic content types can be updated dynamically. Details of the message sequence signaling involved in this process is illustrated in Fig. 4.

CCR's operation includes three main functions: trigger function, objective function, and routing updates with loop detection function, and its system architecture is shown in Fig. 5. The trigger function decides how frequently to execute the CCR objective function. It ensures a high execution frequency under dynamic network conditions in order to keep the routing table up-to-date, and a low execution frequency when the network stabilizes to reduce the cost of local signaling. Once the objective function is triggered, the node queries its neighbors to provide some information such as data traffic status, remaining battery power level and the content in their own routing tables. Using this information as the input of the objective function, candidate rankings are produced and

UNCORRECTED PROOF

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

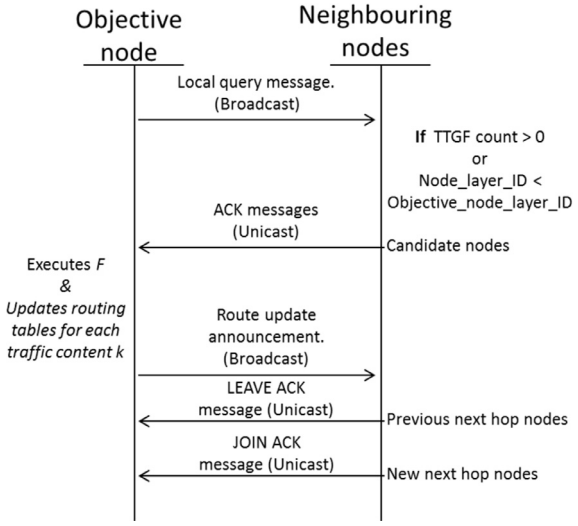


Fig. 4. CCR signaling messages.

the one with the highest ranking is selected to route the corresponding content. Hence, the objective function constructs a separate routing entry for each content in the routing table. Lastly, an effective loop avoidance mechanism is designed in order to detect communication loops and conserve the limited amount of energy stored at each node. Details of each function are described in the following sections.

4.1. Trigger function

In dynamic network environments, most routing protocols periodically update their routing information and keep the routing table up to date. This, however, incurs additional control overheads. In resource constrained networks with lossy links, these signaling messages should be controlled in order to conserve limited onboard node energy.

The frequency of executing the objective function at a time t is controlled by a probability p_t. This probability is calculated independently on each node and does not require any local or global network information. p_t is defined as below:

$$p_t = \min\left(\left(\sum_{t_1}^{t} |\Delta_k| + 1\right) \times p_{default}, 1\right) \quad (2)$$

Where Δ_k is the traffic content variation of each node at a time round.[3] t is the current time instance while t_1 is the previous time instance when the node ran the objective function. $p_{default}$ is the default probability determined by system parameters, details of this can found in Section 6.1.2. An example of variation of Δ_k is shown in Fig. 6(a), and the system function flow to execute F based on p_t is illustrated in Fig. 6(b).

Thus, the probability of executing the objective function increases if a large value of Δ_k is produced due to traffic content variation. On the other hand, when the network stabilizes (small Δ_k), the probability p_t decreases. It becomes $p_{default}$ if no changes occur, which effectively reduces control overhead. This ensures that even in a slow changing environment, the routing table is still kept updated. Yet, the probability to execute the objective function is much lower in a stable network compared to a dynamically changing one.

[3] A round is a basic time unit defined in this paper.

4.2. The objective function (F)

The objective function F is executed on an objective node i in order to find out the most suitable next hop node j for each traffic content k among N neighboring candidates. Since the traffic is differentiated by its content type, the objective node maintains a separate routing entry for each content k and updates it by executing the objective function. Details of the objective function are described as below in (3).

$$F(k) = \max_{j \in N}\left(\tilde{g}_j^k - g_j^k + \beta\frac{\tilde{l}_j^k - l_{j^*}}{\tilde{l}_j^k} + \varepsilon_j^k\right) \quad (3)$$

where the first term $\tilde{g}_j^k - g_j^k$ calculates the normalized communication data reduction via aggregation process which is called as the marginal processing gain. The second term $\frac{\tilde{l}_j^k - l_{j^*}}{\tilde{l}_j^k}$ is the link quality aware local network lifetime gain estimation; while β is a tuning parameter to provide weights between the two parameters. Finally, ε_j^k is a reward parameter. We'll explain each parameter in the following sections.

4.2.1. The processing gain

The first term in (3), $\tilde{g}_j^k$ is the processing gain by allocating application content k's traffic to Node j, and g_j^k is the processing gain without allocating content k's traffic to j, where g_j^k is calculated as:

$$g_j^k = \frac{\sum_{k\in K} R_j^{in}(k) - \sum_{k\in K} R_j^{out}(k)}{\sum_{k\in K} R_j^{in}(k)} \quad (4)$$

Basically, $\sum_{k\in K} R_j^{in}(k)$ and $\sum_{k\in K} R_j^{out}(k)$ stand for the total amount of incoming and outgoing traffic for total K applications on Node j, respectively. Therefore, the numerator of (4) represents the total amount of traffic reduction via data aggregation at node j. This value is then divided by the total incoming traffic. The rationale behind this parameter is:

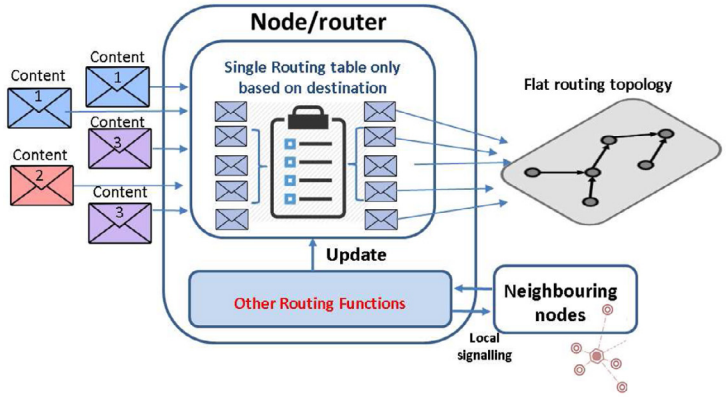
- It is the normalization process which makes the marginal processing gain numerically comparable with the local lifetime gain (second term shown in (3)) and therefore facilitates computation of a multi-gain and,
- For load balancing purposes, it is preferable to relay traffic to a node that can provide the same processing gain (reduce the same amount of data), but with less traffic than is already assigned to it. In other words, with the same amount of reduction in traffic achievable through aggregation, the more incoming traffic a node has, the smaller processing gain it can obtain.

An example illustrating the above concept is shown in Fig. 7. Two applications (a_1 & a_2) are collecting data in a network formed by 6 nodes. Nodes 1–3 are the source nodes of a_1 and Node 4 is the source node for application a_2. It is assumed that both Nodes 4 and 5 can process a_1 and a_2 with $\omega_{a_1} = \omega_{a_2} = 1/M$.[4] Assume that Node 3 is executing the objective function to determine which of the two nodes (Node 4 or Node 5) should forward its traffic (a_1). Clearly, in this example, Node 5 will be selected as it has a better processing gain due to the fact that it is only ferrying traffic for a single application type and therefore can aggregate information unlike Node 4 which cannot aggregate traffic as it is transporting data for two different types of application.

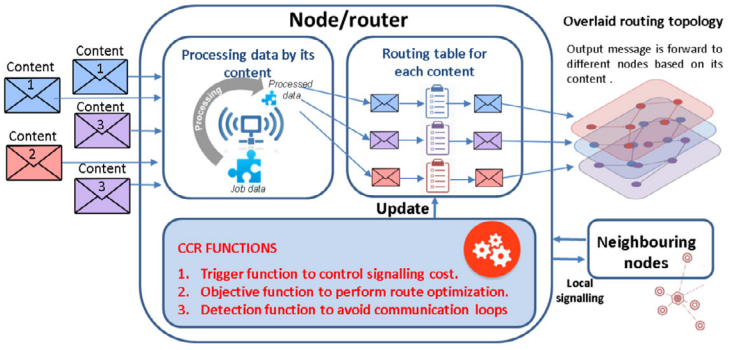
[4] M is the total number of messages received for the same application as described in Section 3.2.

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

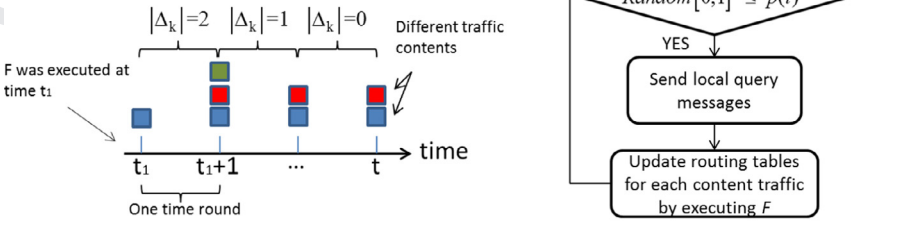


(a) Conventional approach with single routing topology

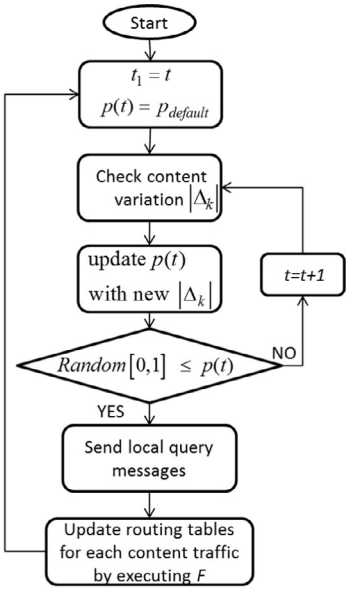


(b) CCR creates an overlaid routing topology based on content

Fig. 5. CCR architecture and operational difference.



(a) Example of traffic content variation



(b) Function flow

Fig. 6. Dynamic trigger function.

By taking advantage of the processing gain, the amount of communication traffic can be reduced by aggregating correlated data. However, this does not necessarily imply that a longer network lifetime can be achieved because energy consumption could be higher if the selected link has a very poor channel quality. Furthermore, heterogeneous node energy levels need to be considered as, in principle, if a node is equipped with more energy it can relay and process more information compared with those with less on-board energy. Therefore, another parameter is introduced in the objective function shown in Eq. (3), known as the link quality aware local lifetime gain.

4.2.2. Link quality aware local lifetime estimation

Due to the inherent nature of the wireless medium, link quality can vary. There are various link quality estimation methods. For example, ETX (Expected Transmission count) [30] is a popular link

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

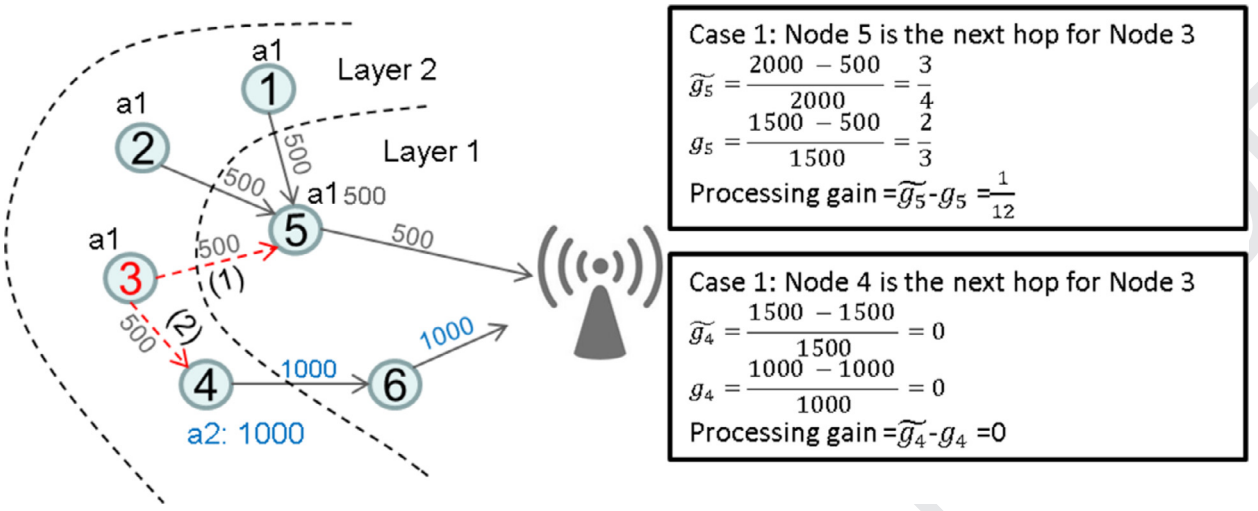


Fig. 7. Examples for processing gain.

quality/reliability parameter used in many routing protocols such as RPL [6]. ETX is the average number of transmissions required by a sender to successfully deliver a message to the destination.

Since the ETX value of a link can be easily converted to the average amount of energy spent on transmissions per packet via that link, we chose this parameter to assess the communication link quality. This further contributes to estimation of the local network lifetime. Even though ETX is used in this work, it should be possible to use other link quality metrics with simple modification to the estimation function.

The local lifetime gain parameter $\frac{\tilde{l}_j^k - l_{j^*}}{\tilde{l}_j^k}$ shown in (3) is defined as the minimum node lifetime among the objective node i and its N qualified neighboring candidate nodes.

$$l = \min\left(\frac{E_i}{e_i}, \min_{j \in N}\left(\frac{E_j}{e_j}\right)\right) \tag{5}$$

where E_i and E_j is the current battery energy level for the objective node and the candidate node respectively; and e_i and e_j is the total energy consumption including processing, transmission and reception costs. In (3), l_{j^*} stands for the current local network lifetime which can be simply obtained via the query process shown in Fig. 4. While $\tilde{l}_j^k$ is the estimated local network lifetime if the content k's traffic is allocated to a new candidate j rather than j^*. In order to calculate $\tilde{l}_j^k$, the estimated energy consumption of the objective node $\tilde{e}_i$ and each candidate node $\tilde{e}_j$ have to be estimated by considering switching the content traffic k from the current next hop node j^* to a new candidate node j. Thus, for each candidate node j ($j \neq j^*$), $\tilde{e}_i^k$ can be calculated as:

$$\tilde{e}_i^k = e_i^* - (ETX_i^{j^*} - ETX_i^j) \times U^k \times e_t \tag{6}$$

where $ETX_i^{j^*}$ and ETX_i^j stand for the ETX value of the current link from i to j^* and the link from i to a candidate node j respectively, U^k is the total amount of data for traffic k and e_t is the average energy consumption to transmit one bit of data.

Similarly, the estimated energy consumption $\tilde{e}_j^k$ ($j \neq j^*$) can be obtained as shown in (7).

$$\tilde{e}_j^k = e_j^* + U^k \times e_r + U^k \times e_p + ETX_j^{NextHop} \times U_p^k \times e_t \tag{7}$$

where e_r, e_p are the energy consumption to receive and process one bit of data and, U_p^k is the amount of additional data after processing which j has to send to its next hop node. If node j cannot process content k, then $U_p^k = U^k$.

4.2.3. Reward parameter ε

The reward parameter ε_j^k is introduced in order to accommodate the heterogeneous processing capability of nodes. Certain nodes, for example, may only be capable of processing specific types of content, due to hardware or software constraints, while other nodes may not be able to process any type of data. The reward parameter ε_j^k is used to give additional credit for a node j that can process the corresponding content k. The value of the reward parameter is set as follows:

$$\varepsilon_j^k = \begin{cases} 0, & \text{if } j \text{ cannot process } k \\ \sigma, & \text{if } j \text{ can process } k \ (\sigma \text{ is a constant}) \end{cases}$$

Please note that the marginal processing gain in F already gives credit to a Node j that is able to process content k, providing a traffic reduction of k can be produced on j. Therefore, even in the absence of the reward parameter, traffic is more likely to be forwarded to nodes that are capable of processing the data in addition to merely relaying it. An example of this is illustrated in Fig. 8(a).[5] However, if a Node j has the capability of processing content k but there is currently no other traffic k routed via j, the processing gain is zero because there is no traffic reduction. In this instance, the reward parameter can help to ensure that traffic is still forwarded to that Node j as shown in Fig. 8(b). Thus, although the value of σ could be relatively small compared to the other parameters in F, it provides a bias to forward traffic to nodes that are capable of processing the particular type of content in question.

4.3. Candidate selection and loop avoidance

Communication loops can cause several problems such as traffic congestion, packet loss (due to Time-To-Live expiry), and additional energy consumed in repeatedly processing and transmission of looping messages. In RPL [6], a popular routing protocol used in lossy wireless networks, a message header is used to detect communication loops. RPL does not allow messages to be routed 'down' to a child node, if these are supposed to be sent 'up' towards the root. If a loop is detected, the message is discarded and a local repair is carried out. However, such a loop avoidance scheme limits the number of neighbors that can be selected as the next hop relay. Consequently, this limits the possibility to perform distributed processing and to reduce the network traffic volume.

[5] Note that for simplicity, in this example, the link quality aware local network lifetime estimation is omitted from consideration.

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

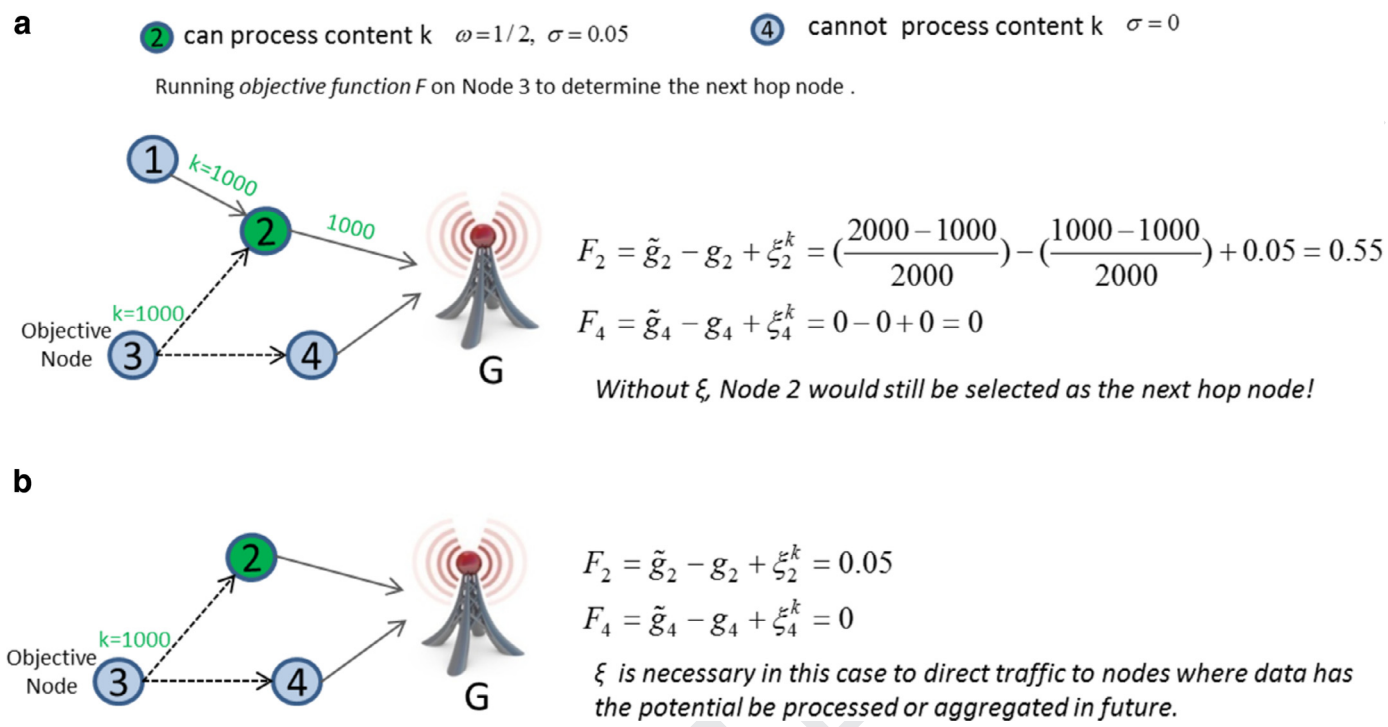


Fig. 8. Example of the benefit of ε.

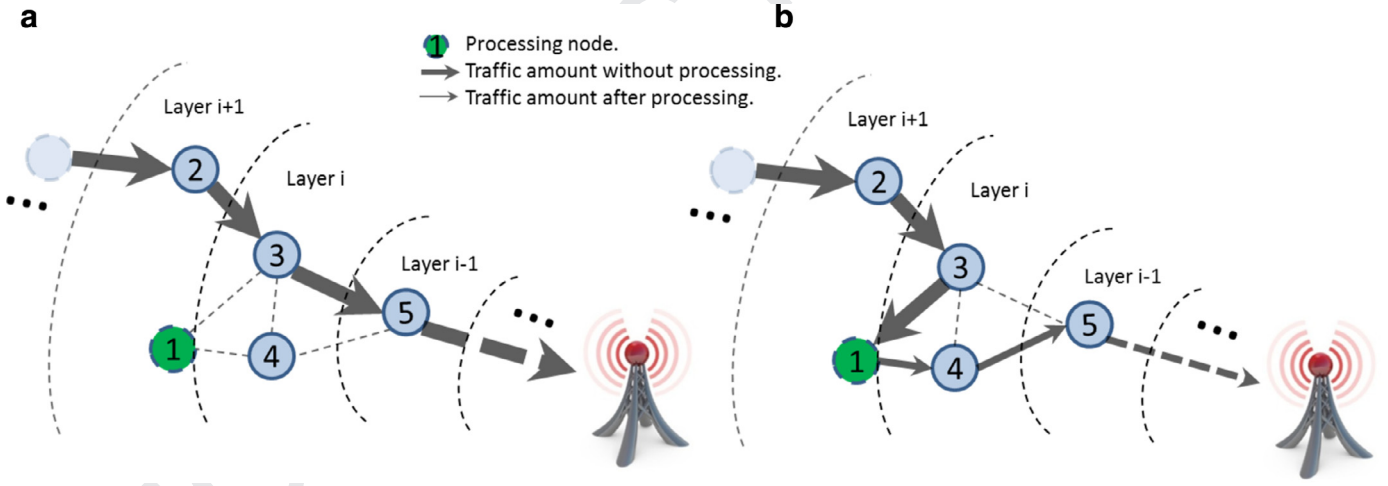


Fig. 9. Example shown the advantage of TTGF loop avoidance.

To overcome this limitation, we introduce the Time-To-Go-Forward (TTGF) in order to select appropriate neighboring nodes (as candidate next hop nodes) that would respond to the local query message and avoid communication loops. TTGF relaxes the restriction introduced in RPL that traffic bound for upward nodes cannot be routed to a downward node, provided a higher processing gain can be achieved within the TTGF tolerance range. Fig. 9 shows an example of the difference between the TTGF approach (Fig. 9(a)) and the conventional RPL loop avoidance approach (Fig. 9(b)).

TTGF is a similar notion to the Time-To-Live (TTL) metric which can be added to the header of the data packet. It works together with the node layer ID, which represents the minimum number of hops required for each node to reach the sink. Details of how to obtain this node layer ID is described in Section 3. TTGF contains two parameters: (1) the TTGF layer ID, (2) the TTGF count. The TTGF layer ID points to the lowest node layer ID that a message reaches. The value of TTGF layer ID is updated as the packet is forwarded on the way to the sink. If a successful forward transmission is made (the current/recipient node layer ID < TTGF layer ID), the value of the TTGF layer ID is updated by the current node layer ID. The TTGF count works as a 'count down' parameter. In case the recipient has the same or higher node layer ID compared to the TTGF layer ID, the message has not reached 'closer' to the sink. Therefore, the TTGF count is reduced by one. Once the TTGF count reaches zero, only those with a lower layer ID compared to the objective node's layer ID can be chosen as the next hop candidate. The TTGF count is set to the preset default value once the TTGF layer ID is updated.

By using TTGF, we allow messages to be relayed to nodes with the same or even higher depth of the network layer within the TTGF tolerance value, such that a proper processing node can be found in order to aggregate data. On the other hand, if a message that has not been forwarded any 'closer' to the sink within TTGF hops, is forced to do so by selecting a lower layer node as the next

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

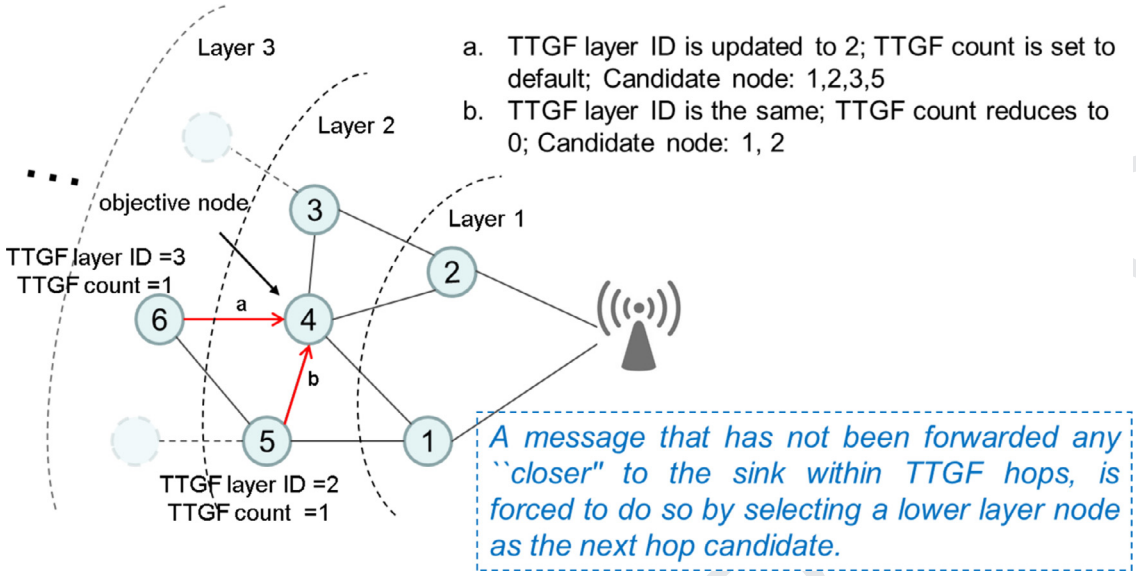


Fig. 10. Loop avoidance with TTGF.

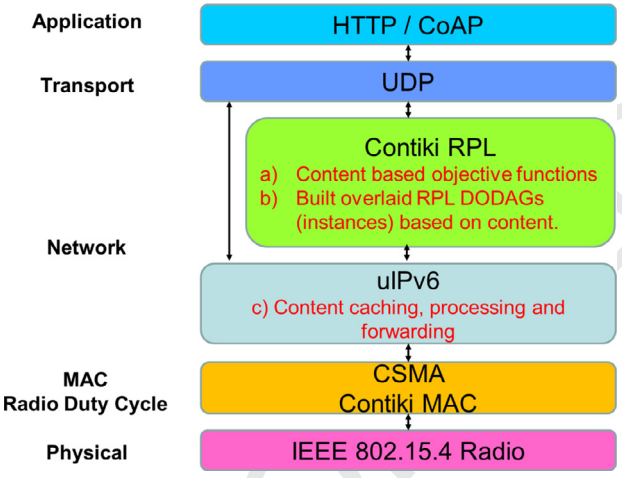


Fig. 11. The integration of CCR components into Contiki and RPL architecture.

hop candidate. An example of using TTGF is shown in Fig. 10. In case of event (a) shown in Fig. 10, a forward transmission is made from node 1 (layer 3) to node 4 (layer 2). Hence, the TTGF layer ID becomes 2 and TTGF count is set to default. Therefore, all the one hop neighboring nodes apart from the previous sender can be the next hop candidate for node 4. For case (b) shown in Fig. 10, the TTGF layer ID is the same while the TTGF count is reduced by 1 and becomes 0 when it reaches node 4. Thus, only those with a lower layer ID (Node 5 and 6) are qualified for the candidate selection and respond to the query message.

5. CCR implementation and integration in RPL

5.1. Summery of RPL

The Routing Protocol for Low-power and Lossy Networks (RPL) is developed and standardized by IETF for enabling connectivity in IoT mesh networks. RPL uses a proactive process to construct and maintain a Destination-Oriented Directed Acyclic Graph (DODAG) routing topology, where the data concentrator sits at the root of the DODAG and the edges form a path from each node to the DAG root. The DODAG construction starts with the root broadcasting a DIO (DODAG Information Object) control message. Any node received this message can choose to join the DODAG by adding the DIO sender to its parent list, and computes its rank relative to the parent node based on an objective function. It then further forward the DIO message with updated rank information. Once the DODAG is in place, nodes can send data via their relay parents until it reach the root. If the network is in a steady state, RPL uses a low-rate DIO beacon process controlled by Trickle timer in order to maintain the DODAG routing topology. However, RPL would temporarily increase the DIO sending frequency by resetting the tricker time if network inconsistencies were detected. This process allows RPL to dynamically adjust its control operations and reduces unnecessary control overheads.

5.2. CCR's operation in Contiki RPL

In the following, the integration of CCR components into the industrial RPL protocol is described. The detailed system architecture is shown in Fig. 11, where our contributions are highlighted, notably we have: a) Developed a content based objective function for the RPL protocol; b) Enabled multiple RPL instances on the same RPL root, such that messages with different content types are routed via different RPL DODAG instances. c) Caching, processing and forwarding functions are added to the network layer to facilitate in-network data aggregation. Details are described below.

There are a few ways to implement CCR, one way is to specify a content Byte in the packet routing header, and the procedures described in Section 4 can be used. However, a standard based implementation is more promising. Therefore, we adopt the existing Contiki RPL standard as a baseline, which is augmented with more functionality without affecting backwards compatibility.

To meet Contiki RPL's implementation style, we revised CCR's messaging exchange procedure and adopted the existing DIO and DAO (DODAG Destination Advertisement Object) control messages proposed in RPL. In addition, since RPL supports multi-topology routing over the same physical mesh network by using an instance-id, we assign instance-id to each content type and create multiple overlaid routing topologies (DODAGs) based on content.

For each content (RPL instance) c, the root node first starts advertising the information about the DODAG graph using the DIO message. Any node within the listening vicinity receives the message, and then it processes the message and makes a decision

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

whether or not to join the graph according to the objective function. Node rank is computed representing the relative position in the DODAG with respect to the root. For simplicity, the objective function in (3) is simplified for CCR implementation and the node rank for each instance c can be calculated as per (9).

$$R_{Node}(c) = R_{parent}(c) - N_{content}(c) * \omega + O_{rank}(c) * \eta + BaseHop \quad (9)$$

where R_{parent} is a parent node's rank; $N_{content}$ is the total number of child nodes associated with the parent node for the same content c (parent node is inclusive if it is also the source node of content c). This means if a parent node having more children for the same content, it would have a larger value of $N_{content}$, hence a lower rank (a better parent) in the DODAG graph; O_{rank} stands for any other objective function (3) related parameters, for instance, ETX or Residual battery level etc.; ω and η are weighting parameters; and BaseHop is a constant. If BaseHop is not equal to 0, the R_{Node} increases by adding more hops.

An example of CCR based multi-DODAG construction is shown in Fig. 12, where Nodes 2, 4, 7, generates content A, Nodes 3, 5, 6, generate content B, and Node 8 generates both content A and content B. The initial single DODAG routing topology without CCR is shown in Fig. 12(1). Now, when we apply CCR, different instance-ids are given to content A and content B, hence the DODAG shown in Fig. 12(1) can be separated into two as shown in Fig. 12(2) and (3). For simplicity, we assume $O_{rank} = 0$, root rank is 10, and BaseHop is set to 4. Each parent will pass its own rank (R_{parent}) along with the Content factor ($N_{content}$) in the DIO message. Let us take content A for example as shown in Fig. 12(3). Since the root node only has Node 2 to provide content A messages, the rank of Node 2 $R_2(A)$ can be calculated by using (9) as $R_2(A) = R_{root}(A) - 1 + 4 = 13$. Similarly, the rank of Node 4 and Node 5 can be calculated accordingly. Now, in order to choose the best parent for Node 8, it needs to calculate its rank based on whether to choose Node 4 or Node 5 as its parent. Since, there would be 3 nodes (Nodes 4, 7, 8) sending content A messages to Node 4, hence $N_{content}(A) = 3$. In contrast, Node 5 would only have Node 8 to send content A packets if it was chosen as parent, therefore $N_{content}(A) = 1$. Obviously, Node 4 is a better parent compared with Node 5 based on the objective function. As a result, Node 8 will switch its parent from Node 5 to Node 4 as shown in Fig. 12(5). Same parent selection process is carried out for Content B in Fig. 12(2) and (4). Eventually, two distinctive DODAGs for content A & B are formed as shown in Fig. 12(6).

5.3. CCR core modules

Memory allocation: a set of memory blocks is statically allocated for buffers, caches, and other data structure to handle CCR's communication.

Content entry: this module includes three main functions, Node_Has_Content( ), *Content_Type_Get() and Node_Add_Content( ). The first function checks whether a parent Node has already cached the same content of a received message from its child. If so, the second function is called to retrieve the pointer pointing to the corresponding content in the memory. Otherwise, the Node_Add_Content( ) is used to allocate memory to the new content. Due to memory limitation of the Hardware platform, the maximum number of content is set to 3 in this implementation.

Content caching: a content table is built with an unique index assigned to each content object. The received packet payload is then copied to the content table with a matching to the corresponding content type.

Content processing and forwarding: cached data for each content will be processed if the parent node receives packets from its children or the maximum number of message stored in the buffer for the corresponding content is reached, or a timeout() is triggered. The processed data will be forwarded based on the destination IP address and destination port.

Table 1
Energy consumption of different operations per Byte.

Operations	Energy consumption (μJ)
Transmission	9.72
Reception	8.22
Flash write	0.445
Flash read	0.315
Data aggregation	0.0011

6. Performance evaluation

6.1. Simulation results

A simulation based study was first carried out in order to evaluate the performance of the proposed CCR protocol with the conventional methods. A global network lifetime maximization tree algorithm [27] (referred to as *Static Tree* hereafter) and the traditional centralized processing scheme (referred to as *Central* hereafter) were chosen as benchmarks.

The Static Tree is a centralized algorithm which pre-constructs a maximum-lifetime data gathering tree before the network starts to operate. In order to achieve a fair comparison, we added the data aggregation to the Static Tree approach. In addition, since global knowledge is required to perform the optimization for Static Tree, which can be very time consuming to re-compute the optimal tree each time there is a change in the network such as node/link failures. To mitigate this issue, we slightly modified the static tree algorithm to adapt to such failure cases, and apply a simple but fast recovery mechanism to randomly choose the next hop node with a lower layer ID if a failure event happens. On the other hand, the central algorithm first gathers all the data at the sink and then carries out processing to compute the results.

6.1.1. Simulation parameters

Unless specified otherwise, a network deployment comprising of 200 nodes with nodes being uniformly distributed with a $200 \times 200\,m^2$ area was assumed. Three applications with heterogeneous traffic rates were considered in the simulations. Nodes were assumed to have unequal energy levels at the startup time in the r ange 4–6 J. The TTGF count was set to 2 and the control packet size was assumed to be 500 bits. $p_{default}$ was set to 0.05 and the value of β was set to 2. A simple data aggregation function which computes the maximum and the average of the sensed values was considered in the simulations. A variable data aggregation rate $\omega = \frac{1}{M}$ was used, where M is the total number of messages received for the same application on a processing node. Tmote Sky node was chosen as our basic node model which is equipped with an MSP430 processor and CC2420 radio chip. The energy consumed by the different operations (per Byte) for the Tmote Sky platform are adopt from [31], [32] and listed in Table 1. Finally, the performance of the proposed CCR algorithm was compared with a centralized lifetime maximization tree algorithm [27] (referred to as *Static Tree* hereafter) and the traditional centralized processing scheme (referred to as *Central* hereafter). All simulation results were averaged from 200 test runs which show 99% confidence interval with about mean-value ± 10% precision. The implementation experiments were run for more than 40 times which show 95% confidence interval within ± 10% of the sample mean.

UNCORRECTED PROOF

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

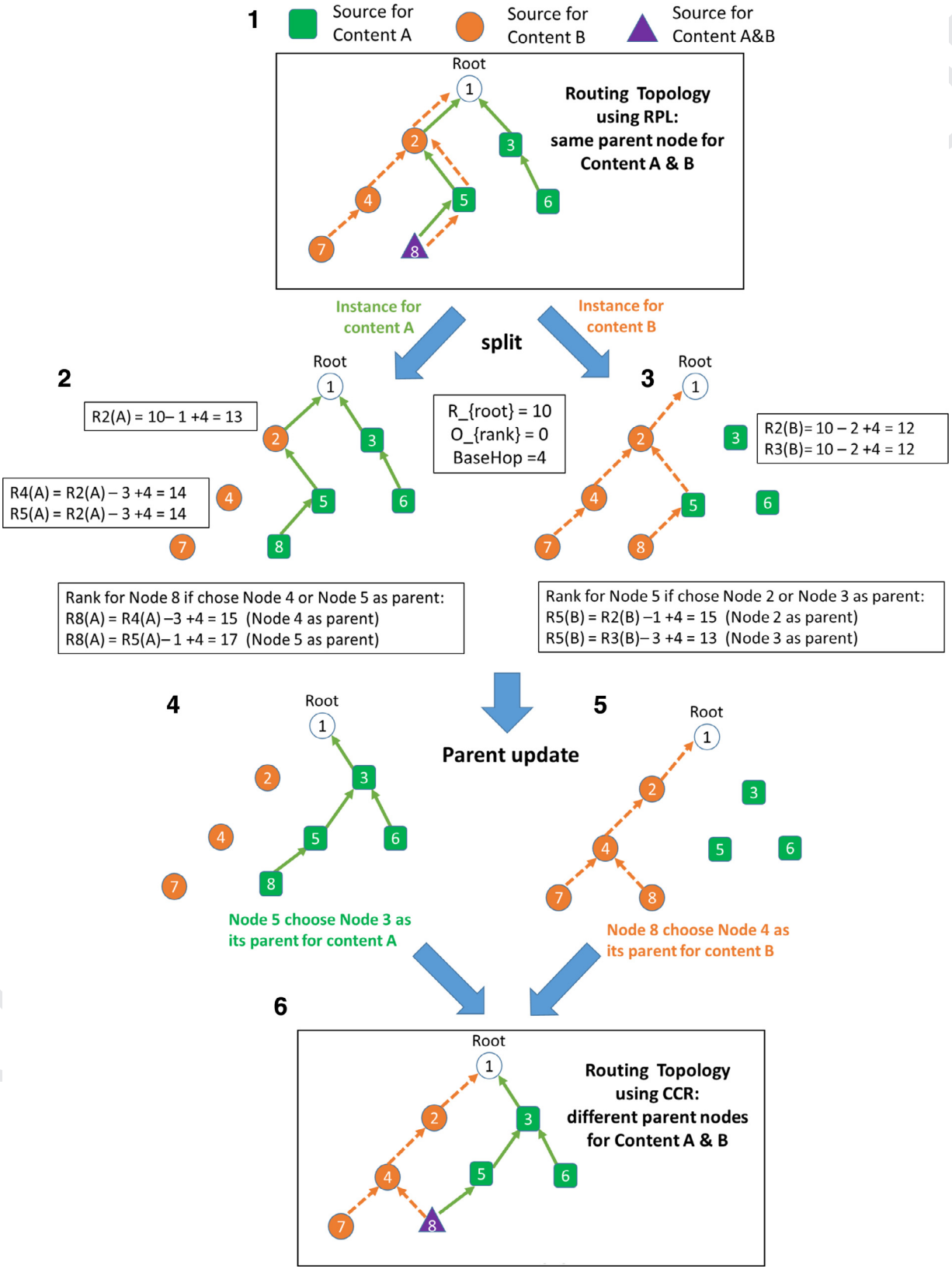
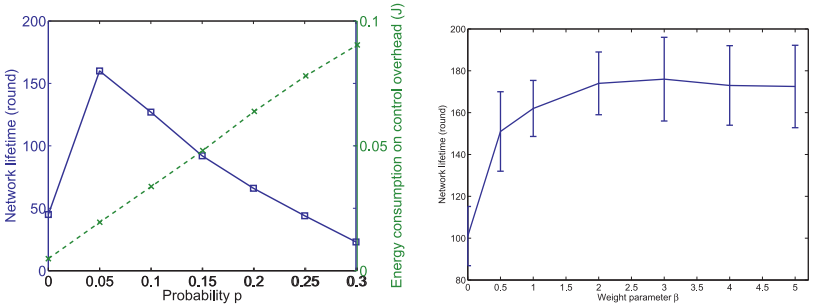


Fig. 12. Mutil-DODAG graphs building process based on content types.

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir



(a) Test for $p_{default}$ (b) Test for β

Fig. 13. Impact of p and β on network lifetime.

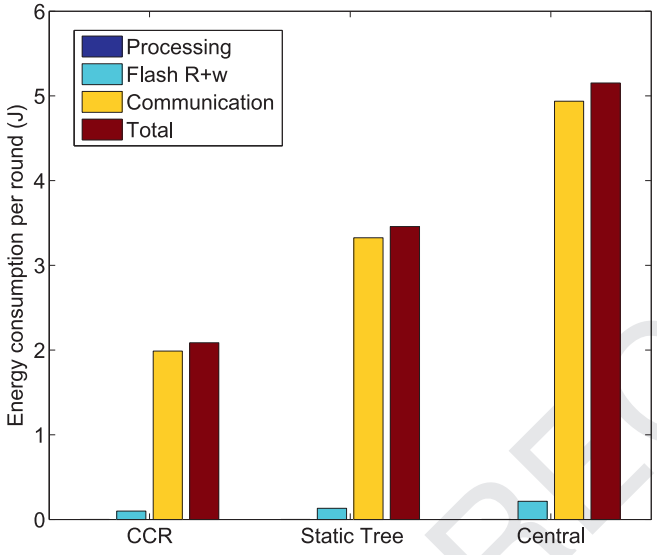


Fig. 14. Energy consumption per round for different processes.

6.1.2. Impact of $p_{default}$ and β on lifetime

To begin with, simulations were run to determine the probability $p_{default}$ of executing the objective function F, and the weight parameter β to decide the tradeoffs between the processing gain and the local lifetime gain. $p_t = p_{default}$ when the network is in a stable status. Intuitively, a larger value of $p_{default}$ gives more chances for each node to select the best hop candidate in terms of having a higher processing gain as well as a balanced lifetime. However, as seen from Fig. 13(a), the network lifetime first increases when $p_{default}$ is set to 0.05, but then it drops very quickly as the value of $p_{default}$ further increases. This stems from the fact that when $p_{default}$ increases, more energy is spent on the control overhead to gather local information in order to execute F. As evident from Fig. 13(a), the energy consumption related to the increased control overhead also increases linearly with $p_{default}$ which implies that the node energy depletes.

A suitable value of β is also needed as it has an impact on the local lifetime gain. A large value of β puts more weight on the local network lifetime gain which consequently produces a smaller value of F for the bottleneck node. However, when β is large enough to avoid overloading the bottleneck node, this increase of β value has no further impact on the network lifetime as evident from Fig. 13(b).

Therefore, application or network specific configurations might be required before the network start to operate. Nevertheless, such simple configuration is acceptable by the industry, for example, the Trickle timer parameter can be tuned in the RPL protocol in order to achieve the best performance result. Please note that in the following experiments, corresponding $p_{default}$ and β values determined in this section are used.

6.1.3. Energy consumptions

Simulations were run to identify the contribution of each of the data processing, flash read/write and communication operations to the total energy consumption. Fig. 14 shows the per round energy consumption of processing, flash read/write and communication operations and, the sum total of these. It is evident from this figure that the energy consumed by communications significantly dominates the energy consumed by the other operations. The central algorithm gathers all the data at the sink and then carries out processing as a result of which it exhibits the highest communication cost. By taking advantage of content-centric data aggregation to reduce the volume of data that needs to be transported, CCR saves more than half of the energy spent on communication in comparison to the Central approach, about a third in comparison to Static tree. Although in-network data aggregation is enabled in Static Tree, it still incurs higher communication cost in comparison to the proposed algorithm. This is because it employs a centralized routing optimization approach, i.e. tree does not adapt to changes in the underlying network (traffic dynamics). Global knowledge of the network is needed to perform the optimization. In a dynamic network environment, it can be time consuming to recompute the optimal tree each time there is a change in the network such as node/link failures, or arrival of a new application. Thus, the performance of the pre-optimized network topology in the Static Tree approach degrades over time. Furthermore, since the processing cost is too small to be spotted in Fig. 14, we point out that the Central approach spent only 31 μJ of energy on processing. This is only half of the processing cost compared with CCR and Static tree. The key point to take note of here is that even a small increase in processing cost for CCR and Static tree translates to large increase in energy saving gains on communication.

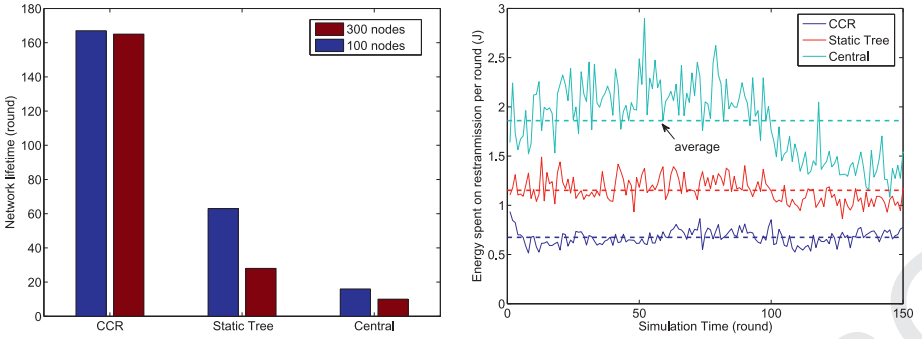
6.1.4. Network lifetime

The network lifetime is defined as the time duration between when the network starts to operate until the first node dies due to energy depletion. As evident from this Fig. 15(a), CCR provides a significant increase in network lifetime compared with the other two. Furthermore, we observe that CCR has a smaller gap when the network is scaled from 100 nodes to 300 nodes. This boils down to the ability of CCR to reduce considerable amount of traffic in the network as a result of using content-centric data aggregation. Additionally, Fig. 15(b) shows the total energy spent on retransmissions per simulation round. As evident from this figure, CCR spends the least amount of energy on retransmissions. This is attributed to its ability to form a more reliable routing topology by taking link quality into account. The central algorithm has the highest energy

UNCORRECTED PROOF

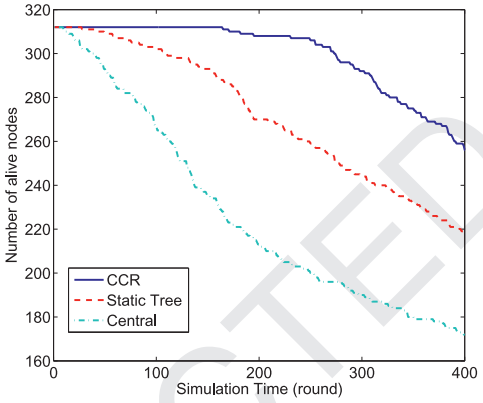
Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir



(a) First Node network lifetime

(b) Energy consumption spent on retransmissions per round



(c) Number of alive node over simulation time

Fig. 15. Network lifetime comparison.

spent on retransmissions due to heavy traffic on links. However, it drops significantly at the latter stage. This stems from the fact that the number of alive nodes in the network is significantly decreased, hence less traffic is generated in the network compared to the other two. Finally, the number of alive nodes against simulation time is shown in Fig. 15(c). Clearly, CCR conserves more energy resources for nodes which is vital for resource constrained devices.

6.1.5. Graphical network traffic comparison

Fig. 16 provides visualized traffic maps for a simulated network with nodes generating three different types of contents. The line width in the traffic map represents the volume of data flowing through that link, i.e., the thicker the line, the higher the volume of traffic flowing through it. Nodes marked by red color in each content traffic map are those that can process the corresponding content. By using CCR, it can be observed that traffic flows of the same content are more likely to be routed to those red processing nodes and correlated data is more likely to be aggregated within the network resulting a much less traffic amount as shown in Fig. 16.

Fig. 17 provides traffic map comparison at different network operation time instances. The 'x' mark indicates a dead node that has already depleted its energy. We can see that the Central approach has much heavier communication traffic compared with CCR. In addition, since the nodes in the area that are closer to the sink need to relay information for those located in the outer region. Massive traffic can be observed at the centre of the network for the Central method. This could easily cause the hot-spot problem, while CCR and Static tree have much less communication data volume after aggregation. Furthermore, by observing the number of dead nodes in Fig. 17, we can clearly tell that CCR performs much better in conserving node energy as well as in providing full network coverage.

Table 2
Experiment setup.

Experiment setup	Parameters
High data rate	1 Packet per second per node
Low data rate	1 Packet per 5 seconds per node
Traffic type	CBR
Number of nodes	10(3Hops), 15(4Hops), 20(5Hops)
Types of contents	2
Maximum cached messages per content	3
Baseline benchmark	RPL

6.2. Implementation results

In this section, we experiment on the evaluation of CCR's performance in Contiki Cooja Emulator based on the TelosB (also known as Tmote Sky) Platform. A screen shot of the emulator is shown in Fig. 18 and details of the experiment setup are shown in Table 2. Since it is not straight forward to implement the *Static Tree* in contiki cooja, we choose the *RPL* standard as the main competitor in our implementation based experiments. Similar to the *Central* approach, RPL does not process data while routing packets.

Fig. 19 presents the number of average transmitted packets over a 10 s of period in the network. It can be observed that CCR is able to significantly reduce the amount of traffic. As a results, CCR spends less energy on communication and prolongs network lifetime as shown in Fig. 20. Although it can be noticed that CCR outperforms RPL in extending the network lifetime, the performance gain is reduced compared with our simulation results. This

UNCORRECTED PROOF

Downloaded from http://iranpaper.ir

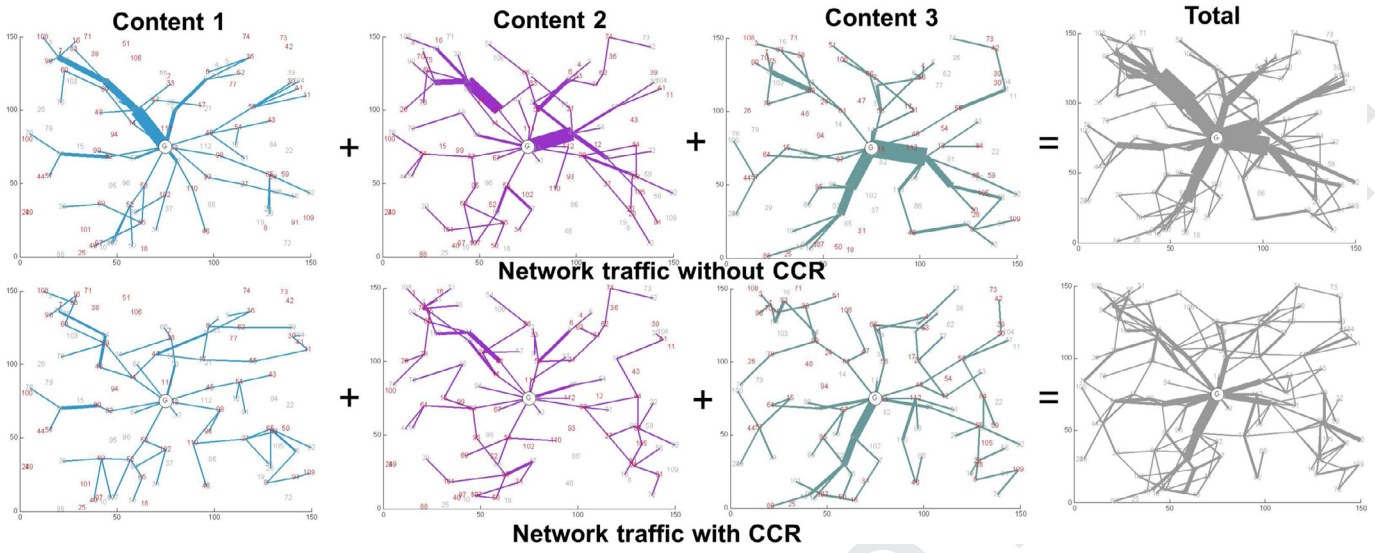
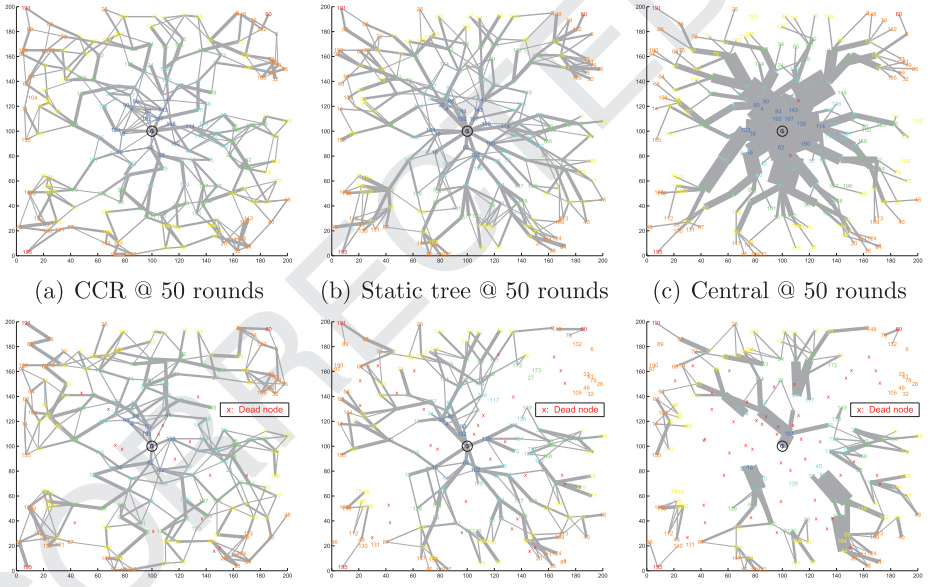


Fig. 16. Network traffic maps with 3 different contents. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).



(a) CCR @ 50 rounds (b) Static tree @ 50 rounds (c) Central @ 50 rounds

(d) CCR @ 650 rounds, No network Partition (e) Static tree @ 650 rounds, Small Network Partition (f) Central @ 300 rounds, Large Network Partition

Fig. 17. Graphic comparison of traffic maps at different time instances.

is mainly due to the reason that we adopted the RPL based message signaling in order to make it compliant with the standard. However, this incurs additional costs on control overhead.

Two additional experiments were conducted in this implementation evaluation, which could not be performed in the previous simulation based tests. We first measure the average packet delay. In this test, a special message is generated on the farthest leaf node, the average reception time on the root node is recorded. In order to have a correct and fair measurement of network latency, this special message is not cached in CCR as this incurs additional waiting time, while the other messages are cached and processed in the network. The results obtained are shown in Fig. 21.

Although the network latency increases when the number of hops increases in the case of both approaches (CCR & RPL), CCR outperforms RPL in both low data rate case and high data rate case. This is because the network is less congested by using CCR, therefore a low network latency can be achieved. We also observe that the magnitude of the improvement increases with an increase in the scale of the network. With 5 Hops (20 nodes) and high data rate setting, CCR can provide as much as twice the reduction in network latency. This indicates the scalability of CCR in dense deployments. We further measured the packet delivery ratio and the outcome is illustrated in Fig. 22. RPL has a rapid performance degradation when the number of nodes increase to 20, only 46% and 20% of the messages will eventually reach the sink in the low data rate and high data rate case, respectively. This is because the central area is highly congested due to heavy network traffic, which RPL is not able to accommodate. In contrast, CCR still achieves over 90% of PDR in the low data rate case with 20 nodes and 60% of PDR in high data rate case.

6.3. CCR demonstration

Finally, we ported our implementation codes to the real hardware (TelosB node) and developed a Demo. Since it is very challenging to deploy a large-scale multi-hop network with real

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

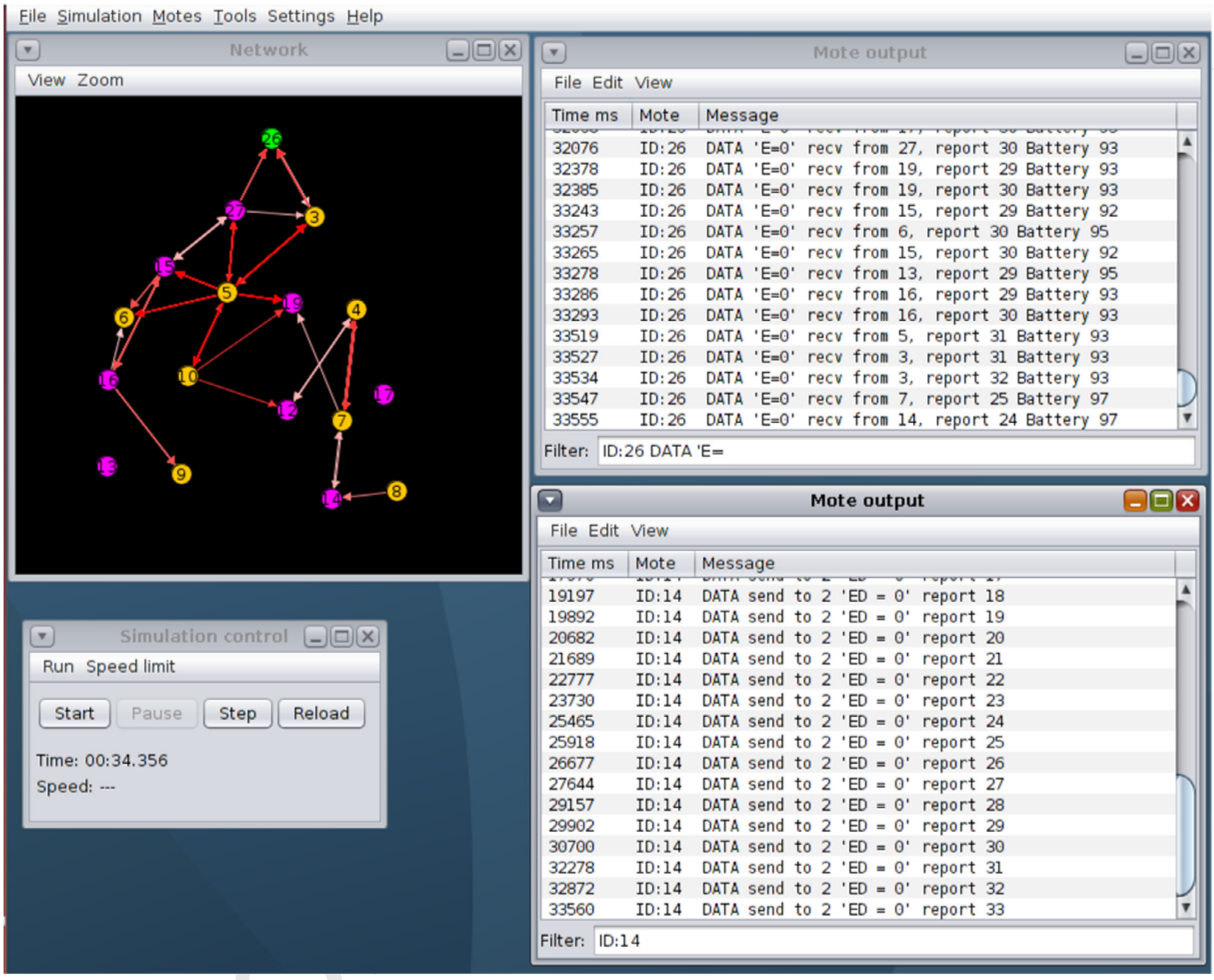


Fig. 18. CCR's evaluation on Contiki Cooja.

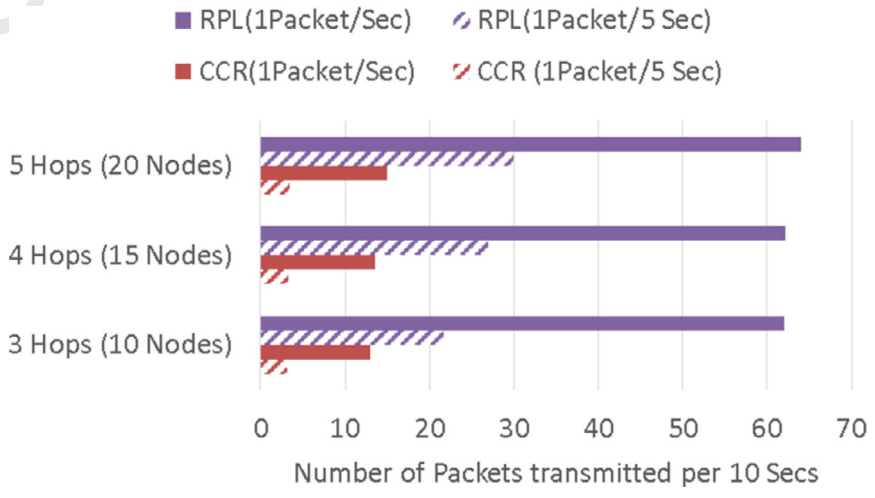


Fig. 19. Experimental results – aggregation gain.

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

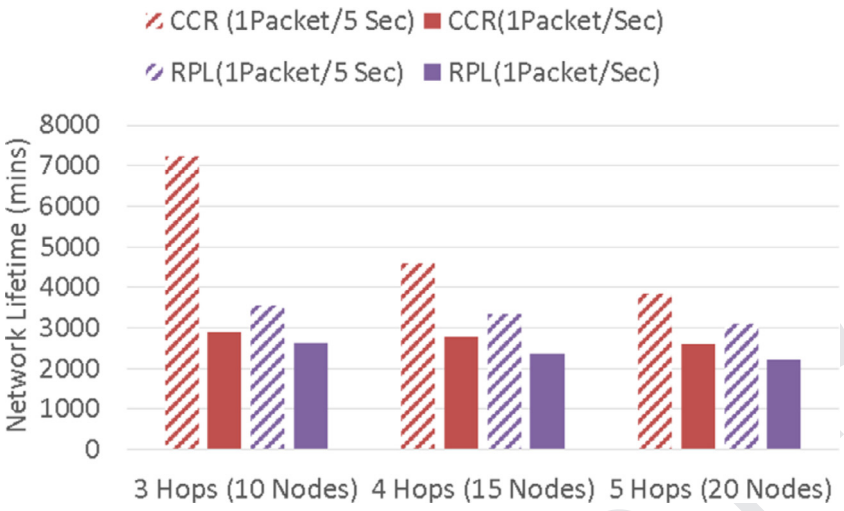


Fig. 20. Experimental results – network lifetime.

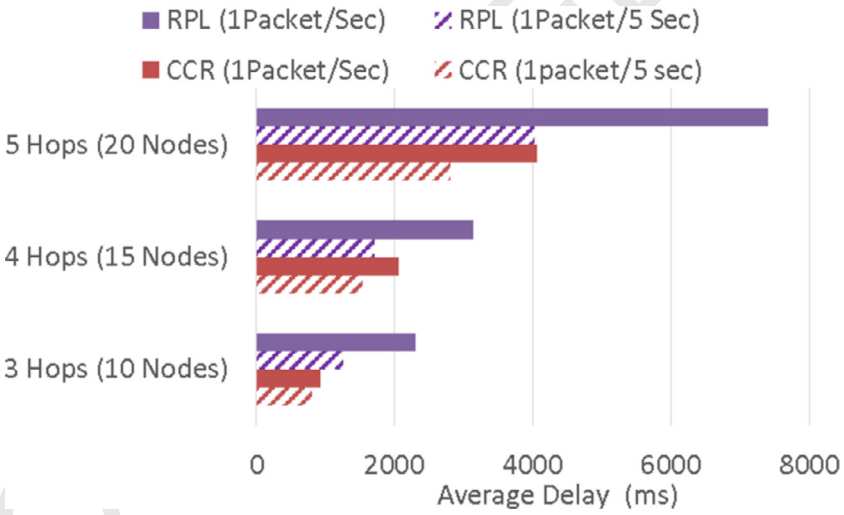


Fig. 21. Experimental results – network latency.

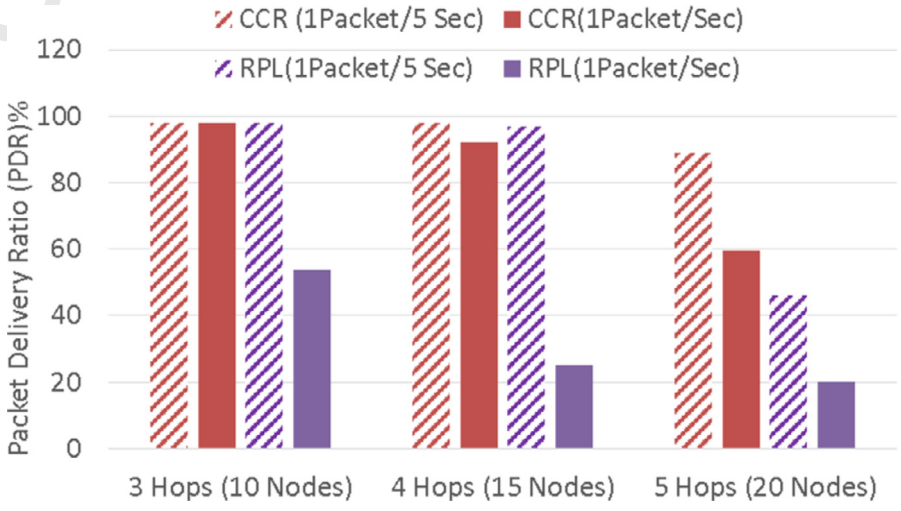


Fig. 22. Experimental results – packet delivery ratio.

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

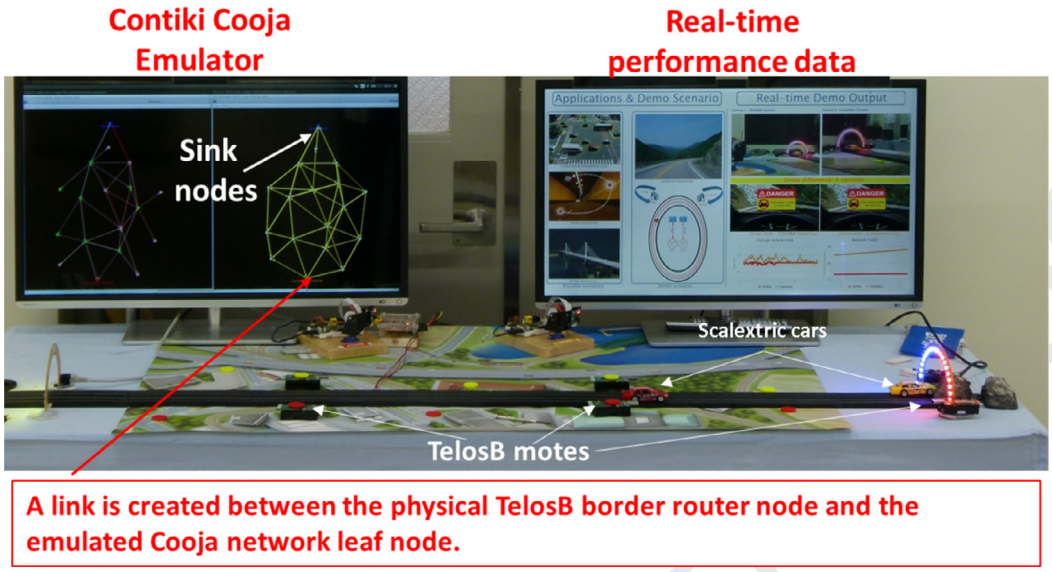


Fig. 23. CCR demo steup.

hardware nodes, as a first step we found a hybrid solution with a Contiki Cooja emulator emulating a large-scale mesh network along with a few physical nodes to form the outer part of the mesh network. Both the physical node and emulated node are running the same CCR code. The boarder router of the physical network is connected to one of the leaf nodes emulated in the Contiki Cooja emulator. Hence, all packets sent by the telosB motes will route through both the small scale real mesh network and the emulated large-scale wireless mesh network. By such approach, we can demonstrate CCR technological benefit for a large scale mesh network, a demo setup picture can be seen in Fig. 23. We have successfully showcased the CCR demo in various occasions including Venturefest Bristol and Bath, 2015.

7. Conclusion

In this paper, we proposed and studied the performance of an efficient data aggregation and reliable data delivery scheme CCR for a deployment scenario where data from the IoT network endpoints such as sensors have to traverse over wireless lossy links on the way to the other endpoint hosting the IoT application. In particular, CCR is a distributed approach which considers the traffic reduction gain achieved through content-centric data aggregation when routing traffic over reliable communication links by incorporating link quality information. Based on the content of a message, each node constructs a separate routing entry for each content type by running the proposed novel objective function; the key idea being to route heterogeneous types of content via selected reliable communication links to nodes which are capable of aggregating and processing the information before forwarding the summary information. This greatly reduces redundant communication traffic and reduces retransmissions as a positive side-effect. Both simulation and implementation results confirm that CCR can significantly extend the network lifetime, reduce network latency and improve communication reliability.

For future work, hardware motes using the CCR protocol will be deployed in our office premises in order to collect more data. In addition, the impact of the number of content types will be investigated with the support of new hardware with larger memory. Last but not the least, technologies such as data mining, fuzzy logic will be explored to support in-network processing. We expect the performance of CCR to improve further with the implementation of more advanced processing functions and a better content definition supporting more content types in the network.

Uncited Refrerences

Ref. [9],[16],[19],[20]

References

[1] D. Datla, X. Chen, T. Tsou, S. Raghunandan, S. Hasan, J. Reed, C. Dietrich, T. Bose, B. Fette, J. Kim, Wireless distributed computing: a survey of research challenges, IEEE Commun. Mag. 50 (1) (2012) 144–152, doi:10.1109/MCOM.2012.6122545.
[2] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Netw. 52 (12) (2008) 2292–2330.
[3] E. Fasolo, M. Rossi, J. Widmer, M. Zorzi, In-network aggregation techniques for wireless sensor networks: a survey, IEEE Wirel. Commun. 14 (2) (2007) 70–87, doi:10.1109/MWC.2007.358967.
[4] Y. Jin, P. Kulkarni, S. Gormus, M. Sooriyabandara, Content centric and load-balancing aware dynamic data aggregation in multihop wireless networks, in: Proceedings of the IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2012.
[5] C. Alvarado, J. Teevan, M.S. Ackerman, D. Karger, Surviving the Information Explosion: How People Find Their Electronic Information, MIT, USA, 2003.
[6] T. Winter, P. Thubert, etal, RpI: Ipv6 Routing Protocol for Low-power and Lossy Networks, (rfc 6550), http://www.ietf.org/rfc/rfc6550.txt, 2012.
[7] A. Dunkels, B. Gronvall, T. Voigt, Contiki – a lightweight and flexible operating system for tiny networked sensors, in: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, 2004.
[8] CCNx Project, http://www.ccnx.org/
[9] G. Mishra, M. Dave, A review on content centric networking and caching strategies, in: Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT), 2015, pp. 925–929, doi:10.1109/CSNT.2015.119.
[10] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, G. Polyzos, A survey of information-centric networking research, IEEE Commun. Surv. Tutor. 16 (2) (2014) 1024–1049, doi:10.1109/SURV.2013.070813.00063.
[11] R. Rajagopalan, P. Varshney, Data-aggregation techniques in sensor networks: a survey, IEEE Commun. Surv. Tutor. 8 (4) (2006) 48–63, doi:10.1109/COMST.2006.283821.
[12] S. Lindsey, C. Raghavendra, K. Sivalingam, Data gathering algorithms in sensor networks using energy metrics, IEEE Trans. Parallel Distrib. Syst. 13 (9) (2002) 924–935, doi:10.1109/TPDS.2002.1036066.
[13] H.-C. Lin, F.-J. Li, K.-Y. Wang, Constructing maximum-lifetime data gathering trees in sensor networks with data aggregation, in: Proceedings of the 2010 IEEE International Conference on Communications (ICC), 2010, pp. 1–6, doi:10.1109/ICC.2010.5502286.
[14] H.O. Tan, I. Körpeoğlu, Power efficient data gathering and aggregation in wireless sensor networks, SIGMOD Rec. 32 (4) (2003) 66–71, doi:10.1145/959060.959072.
[15] J. He, S. Ji, Y. Pan, Y. Li, Constructing load-balanced data aggregation trees in probabilistic wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 25 (7) (2014) 1681–1690.

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005

Downloaded from http://iranpaper.ir

[16] B. Zhang, W. Guo, G. Chen, J. Li, In-network data aggregation route strategy based on energy balance in WSNS, in: Proceedings of the 2013 11th International Symposium on Modeling Optimization in Mobile, Ad Hoc Wireless Networks (WiOpt), 2013, pp. 540–547.
[17] S. Ji, J.S. He, Y. Pan, Y. Li, Continuous data aggregation and capacity in probabilistic wireless sensor networks, J. Parallel Distrib. Comput. 73 (6) (2013) 729–745.
[18] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Trans. Wirel. Commun. 1 (4) (2002) 660–670, doi:10.1109/TWC.2002.804190.
[19] O. Younis, S. Fahmy, Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks, IEEE Trans. Mob. Comput. 3 (4) (2004) 366–379, doi:10.1109/TMC.2004.41.
[20] M. Ye, C. Li, G. Chen, J. Wu, Eecs: an energy efficient clustering scheme in wireless sensor networks, in: Proceedings of the 24th IEEE International Performance, Computing, and Communications Conference, 2005 (IPCCC 2005), 2005, pp. 535–540, doi:10.1109/PCCC.2005.1460630.
[21] D. Wei, Y. Jin, S. Vural, K. Moessner, R. Tafazolli, An energy-efficient clustering solution for wireless sensor networks, IEEE Trans. Wirel. Commun. 10 (11) (2011) 3973–3983, doi:10.1109/TWC.2011.092011.110717.
[22] Y. Abidy, B. Saadallahy, A. Lahmadi, O. Festor, Named data aggregation in wireless sensor networks, in: Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–8.
[23] L. Villas, A. Boukerche, H. Ramos, H. de Oliveira, R. de Araujo, A. Loureiro, Drina: a lightweight and reliable routing approach for in-network aggregation in wireless sensor networks, IEEE Trans. Comput. 62 (4) (2013) 676–689.
[24] Y. Jin, S. Gormus, P. Kulkarni, M. Sooriyabandara, Link quality aware and content centric data aggregation in lossy wireless networks, in: Proceedings of the 2014 IEEE Wireless Communications and Networking Conference (WCNC), 2014, pp. 3082–3087.
[25] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, Tag: a tiny aggregation service for ad-hoc sensor networks, SIGOPS Oper. Syst. Rev. 36 (SI) (2002) 131–146, doi:10.1145/844128.844142. URL http://doi.acm.org/10.1145/844128.844142.
[26] S. Motegi, K. Yoshihara, H. Horiuchi, Dag based in-network aggregation for sensor network monitoring, in: Proceedings of the International Symposium on Applications on Internet (SAINT '06), IEEE Computer Society, Washington, DC, USA, 2006, pp. 292–299, doi:10.1109/SAINT.2006.20. URL http://dx.doi.org/10.1109/SAINT.2006.20.
[27] Y. Wu, Z. Mao, S. Fahmy, N. Shroff, Constructing maximum-lifetime data-gathering forests in sensor networks, IEEE/ACM Trans. Netw. 18 (5) (2010) 1571–1584, doi:10.1109/TNET.2010.2045896.
[28] A. Sharaf, J. Beaver, A. Labrinidis, K. Chrysanthis, Balancing energy efficiency and quality of aggregate data in sensor networks, VLDB J. 13 (4) (2004) 384–403, doi:10.1007/s00778-004-0138-0.
[29] I. Solis, K. Obraczka, The impact of timing in data aggregation for sensor networks, in: Proceedings of the 2004 IEEE International Conference on Communications, 6, 2004, pp. 3640–3645Vol.6, doi:10.1109/ICC.2004.1313222.
[30] D.S.J. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom '03), ACM, New York, NY, USA, 2003, pp. 134–146.
[31] N. Tsiftes, A. Dunkels, A database in every sensor, in: Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys '11), ACM, New York, NY, USA, 2011, pp. 316–332, doi:10.1145/2070942.2070974.
[32] S. Nath, Energy efficient sensor data logging with amnesic flash storage, in: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks (IPSN '09), IEEE Computer Society, Washington, DC, USA, 2009, pp. 157–168.

UNCORRECTED PROOF

Please cite this article as: Y. Jin et al., Content centric routing in IoT networks and its integration in RPL, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.005