EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems
Research Article EAI.EU

An Efficient Traffic-aware Caching Mechanism For Information-centric Wireless Sensor Networks

Ngoc-Thanh Dinh

[1]School of Electronic Engineering, Soongsil University, Korea

Abstract

This paper proposes a traffic-aware caching mechanism (TCM) for resource-constrained nodes in information-centric WSNs. TCM pushes up popular upstream content objects to be cached nearby the sink. Less popular content objects are cached farther from the sink compared to popular content objects. TCM also pushes down popular downstream content objects to be cached inside the network. The objective of TCM is to reduce the number of interest messages required forwarding in multiple hops inside WSNs to optimize stretch ratio, energy efficiency, and content retrieval latency. We implement TCM on the top of existing popularity-based caching schemes to improve their performance. Through analysis and experimental results, we show that TCM achieves a significant improvement in terms of energy efficiency, content retrieval latency, cache hit ratio, and stretch ratio compared to state-of-the-art popularity-based caching schemes.

Received on 03 February 2022; accepted on 11 April 2022; published on 21 April 2022

Keywords: Internet of Things, wireless sensor networks, information centric networking, caching scheme, content store, traffic aware

doi:10.4108/eetinis.v9i30.561

1. Introduction

Information-centric Networking (ICN) is considered as one of the most promising network paradigms for Internet of Things (IoT). The authors in [1–3] used ICN to implement smart city applications. The authors in [4] used ICN to implement applications for smart homes based on hierarchical naming and multi-party routing schemes to retrieve content from multiple content producers. In [5, 6], the authors presented benefits of ICN in industrial automation and intelligent transportation systems. In ICN, in-network caching is the key component that enables routers to cache content objects at any intermediate node in the network and make cached content objects accessible for requests to enhance the network performance. Based on the literature [7–9], we classify existing ICN caching studies for IoT into four categories including graph-based caching, label-based caching, probabilistic caching, and popularity-based caching. Literature studies [7–9] classified ICN caching mechanisms for IoT into popularity-based caching, probabilistic caching, label-based caching, and graph-based caching. Popularity-based caching stands out as one of the most efficient approaches for in-network caching in IoT [9, 10]. In Precache [11], the authors implemented an ICN caching mechanism relying on content relevance. In [12], MPC is proposed as a caching mechanism that counts the number of incoming requests for a content object as its popularity. MPC determines a threshold to classify popular and less popular content objects. Based on the content popularity, MPC makes recommendations for nodes receiving popular content objects to cache them. In [13], CPCCS utilizes a dynamic threshold for the number of content requests to classify optimal popular content (OPC) and least popular content (LPC). Based on the classification, CPCCS suggests all routers on routing paths to cache OPC objects and fewer routers along a routing path to cache LPC objects.

Above ICN caching schemes motivate multiple routers to make a caching decision to cache a popular content object. As a result, the content diversity per a storage unit of content store (CS) of nodes is limited. This may be inefficient in resource-constrained WSNs where nodes have limited storage capacity. However,

*Corresponding author. Email: thanhdcn@dcn.ssu.ac.kr

above caching schemes may not be efficient for resource-constrained IoT devices because the caching schemes recommend multiple routers to cache a popular content object. This reduces the number of unique content objects to be cached and lowers the content diversity in the content store (CS) of nodes. In the previous work [14, 15], we proposed to address the above issue by enabling CS information of nodes in the data plane and coordinating content caching with interest packet forwarding to increase the network performance. Although above schemes enhance the usage of popular content objects to improve the cache hit ratio, the traffic pattern and node properties of resource-constrained IoT networks are not considered. In resource-constrained IoT such as wireless sensor networks, there are two main types of traffic patterns. The first one is upstream traffic of sensing data in which content requests are normally forwarded to IoT devices through the sink node. The second one is downstream traffic sent to sensors and actuators for network configuration or commands. Considering characteristics of traffic patterns, although popular content objects are cached inside the network, it would be inefficient if they are cached at a router that is far from their consumers. The reason is that resource-constrained IoT devices are normally low power, many nodes involved in forwarding an interest message and content object consume a significant amount of energy of the network. In addition, IoT nodes (i.e., sensors or actuators) may sleep and wakeup following a wakeup schedule, so interest messages that require forwarding in multiple hops inside WSNs may experience long delay and consume an additional significant energy amount of resource-constrained IoT nodes. Consumers in the first traffic type are normally sent requests through the sink node. Consumers in the second traffic type are nodes inside the network. We suggest that popular content objects should be cached as closer to consumers corresponding with each traffic type as possible to increase their availability to consumers anytime and to reduce the energy consumption of nodes as well as the content retrieval latency.

This paper takes into account the traffic pattern and node properties of wireless sensor networks in optimizing ICN caching for WSNs. We propose a traffic-aware caching mechanism (TCM) for resource-constrained nodes in information-centric WSNs. TCM can be implemented on the top of existing popularity-based caching schemes to enhance their performance [12, 13], with two strategies as follows. TCM pushes up popular upstream content objects to be cached nearby the sink. Less popular content objects are cached farther from the sink compared to popular content objects. TCM also pushes down popular downstream content objects to be cached inside the network. In this way, TCM helps not only improve the cache hit ratio but also the average stretch ratio, the average hop count required to forward an interest message inside a WSN. The objective of TCM is to reduce the number of interest messages required forwarding in multiple hops inside WSNs to optimize stretch ratio, energy efficiency, and content retrieval latency of the network. We implement TCM on the top of state-of-the-art popularity-based caching schemes including MPC [12] and CPCCS [13], namely MPC-TCM and CPCCS-TCM, respectively. Through analysis and experimental results, we show that TCM achieves a significant improvement in terms of energy efficiency, content retrieval latency, cache hit ratio, and stretch ratio compared to state-of-the-art popularity-based caching schemes.

The rest of this paper is organized as follows. In section 2, we discuss related works. In section 3, we provide the overview and the detailed design of the proposed traffic-aware caching mechanism (TCM). Section 4 describes our analysis, experiments and obtained results. Finally, Section 5 concludes the paper.

2. Related Work

In-network caching is the key component in ICN that enables routers to cache content objects around the network and make them accessible for requests to improve the content availability and content retrieval latency [9, 10, 21]. Literature studies [7–9] classified ICN caching mechanisms for IoT into popularity-based caching, probabilistic caching, label-based caching, and graph-based caching.

Probabilistic caching mechanisms are implemented based on a probability p for making caching placement decisions. IN LCE [16, 17], the authors introduced simple and popular rules for probabilistic caching where ICN routers in the network greedily cache new content objects that are not existing in their content store (CS). However, its tradeoff is a high rate of redundancy. In [18–20], the authors enhanced LCE by introducing various dynamic caching policies which change caching probability dynamically. In HCP [19, 20], the authors optimize the network performance by using $CacheWeight_y$ to reduce the number of content redundancy and using $CacheWeight_M$ to lower the stretch length between content consumers and providers. Label-based caching mechanisms are implemented based on policies to label content objects with certain properties. Based on the labels, nodes classify content objects with special policies. For example, proposed mechanisms in [22, 23] consider the network topology for labeling and recognizing network context to enhance caching decisions. Graph-based caching mechanisms take into account network structure and routing paths to enhance caching decisions. In [**?**], the authors designed edge caching policies for caching placement. In [**?**], the authors

proposed different ways to adapt caching positions and centrality of network nodes. Popularity-based caching stands out as one of the most efficient approach for in-network caching in IoT [9, 10]. In Precache [11], the authors implemented an ICN caching mechanism relying on content relevance. In [12], MPC is proposed as a caching mechanism that counts the number of incoming requests for a content object as its popularity. MPC determines a threshold to classify popular and less popular content objects. Based on the content popularity, MPC makes recommendations for nodes receiving popular content objects to cache them. In [13], CPCCS utilizes a dynamic threshold for the number of content requests to classify optimal popular content (OPC) and least popular content (LPC). Based on the classification, CPCCS suggests all routers on routing paths to cache OPC objects and less routers along a routing path to cache LPC objects.

Above caching mechanisms may show limitations when applying for resource-constrained IoT devices because caching schemes recommend multiple routers to cache a popular content object and don't consider the traffic pattern and node properties of resource-constrained WSNs. In resource-constrained IoT such as wireless sensor networks, content requests are normally forwarded to IoT devices through the sink node. Therefore, although popular content objects are cached inside the network, it would be inefficient if they are cached at a router that is far from the sink node. This paper takes into account the traffic pattern and node properties of wireless sensor networks in optimizing ICN caching for WSNs. We propose a traffic-aware caching mechanism (TCM) mechanism for resource-constrained WSNs to push unique popular content objects toward the sink node to improve the overall performance of ICN popularity-based caching schemes for WSNs.

3. Popularity-based Cache Placement Strategy

In popularity-based caching, nodes decide which content objects should be cached based on content frequency and interest message distribution. This approach is to increase the usage of popular content objects to increase the cache hit ratio. Popularity-based caching mechanisms make caching decision based on content request distribution and content access frequency. Mechanisms belonging to this category optimize the usage of popular content objects to improve cache hit ratio. In existing popularity-based caching schemes, there are several ways to determine a popular content object. However, they share the same mechanism to sample the popularity of content object. In particular, each node counts locally the number of requests for each content name c, namely content access frequency (f_c). When f_c is higher than or equal to a value, namely a threshold, c is classified as a popular content object. Each popularity-based cache placement strategy may have different rules to guide nodes to cache content objects. In MPC [12], nodes counts the number of incoming requests for a content object as its popularity and determine a threshold to classify popular and less popular content objects. Based on the content popularity, MPC makes recommendations for nodes receiving popular content objects to cache them. In [13], CPCCS utilizes a dynamic threshold for the number of content requests to classify optimal popular content (OPC) and least popular content (LPC). Based on the classification, CPCCS suggests all routers on routing paths to cache OPC objects and less routers along a routing path to cache LPC objects. CPCCS recommends all routers along the routing path should cache OPC content objects while fewer routers should cache LPC content objects. We implement TCM on the top of existing cache placement strategies such as MPC and CPCCS to improve their performance in wireless sensor networks.

3.1. Unique Popular Content Caching Policy

To improve the diversity of cached content in a neighborhood, we reuse the idea of CS information exchanging which is implemented in our previous work [14, 15]. Our proposed idea makes CS information available in the data plane to coordinate CS of nodes within its neighborhood. We propose efficient procedures for CS information exchanging using counting bloom filter (CBF) [28].

A bloom filter (BF) is well known as a probabilistic data structure designed to enable rapid checking of whether an element is present in a set. A bloom filter is a very space efficient structure consisting of only a m-bit array. A counting bloom filter uses the same functions of bloom filters with counters to enable deleting elements from its data structure. Due to resource-constrained storage, this property is necessary in IoT because content objects can be deleted or added to a CS. We use CBF as a data structure to summarize a compact name set of content objects being cached in the content store of a node [28]. As a result, we only need to use CBF for storing as well as exchanging CS information among nodes in the network. This helps reduce the amount of CS information required to be exchanged and stored.

Each node in the network summarizes information of its CS into a CBF, called a local CBF. The local CBF of a node is updated when the node adds or deletes content objects from its CS. For CS information exchanging in the data plane, the node advertises the compressed version of its local CBF [29] with its neighbor nodes within a neighborhood size defined by the mechanism, for example N hops distance from the node. To save energy, the node only piggybacks

the compressed version of its local CBF in existing advertisement messages available in the signaling channel at the lower layer protocol as described in our previous study [30]. Specifically, the resulted data is piggybacked and encapsulated into signaling messages of the 802.15.4 MAC layer at the sender and then decapsulated at the receivers. In this way, our design for CS information exchange does not incur additional packet transmission.

We create a new table structure for each node, called neighbor content store (NCS) table containing CS information received from its neighbor nodes. NCS is a lightweight table structure which consists of one or several CBFs. Each CBF is associated with a communication interface of the node and contains CS information of neighbors coming from the corresponding interface by merging local CBFs received from each neighbor node associated with the interface. Other data structures such as pending interest table (PIT), forwarding information base, and content store follow the conventional design of ICN [16, 31].

The objective of our unique caching policy is to maximize the number of popular content objects being cached within a neighborhood. Based on top of above popularity-based caching placement strategies, each node contributes to the objective of its neighborhood by optimizing the number of unique popular content objects cached in its CS. For that objective, before deciding to catch a popular content object following a popularity-based caching strategy, a node first checks whether or not the object has been cached by its neighbor by validating the content object in its neighbor CS table. If the content object is not cached by its neighborhood, the node decides to catch the object in its content store. If the content object is available in the CS of its neighbors, the node decides not to catch it. When the node has a full storage, the node replace the cache if the new content object has a higher popularity and has not been cached in its neighbors' content store.

3.2. Traffic-aware caching mechanism

For the upstream data traffic, TCM performs push up strategy to push popular content objects to be cached nearby the sink node. Whenever the content access frequency of content object c, f_c, cached in the CS of a node s achieves a certain popularity level threshold f^m, node s forwards c with its f_c information to the upper node r toward the sink through the interface where the interest message for c was received. We call s as the cached object sender (COS). If the upper node r has available storage in its CS or r finds c more popular than existing content objects cached in its CS, the node caches c or replaces a content object in its CS with c. We call r as the cached object receiver (COR) and the operation of migrating c from s to r as cache migration. COR r then replies COS s with an acknowledge message. According to receiving the acknowledge message, COS s deletes c from its CS. r continues to push c to upper nodes toward the sink following above procedures. The migration trial stops when c doesn't have a higher popularity f_c than cached content objects at the upper node, so the upper node doesn't cache c in its CS and there is no acknowledge messages replied. In this case, the sender s keeps c in its CS and stop pushing c toward the sink. For the downstream traffic, TCM performs push down strategy to push content objects to be cached by nodes inside the network. This type of content is normally sent to nodes inside the network for coordination, configuration or command. In other words, consumers of this type of information located inside the network. Each consumer normally requests each object belonging to this type of information once. For this type of traffic, whenever a node s, which caches a content object c belonging to the downstream traffic, receives an interest message for c from a downstream node, s responds with c. After receiving an acknowledgment from the downstream node, s deletes c from its CS and the downstream node caches c instead. In this way, c is pushed down to the network so that consumers can retrieve c from a shorter distance. In addition, nodes nearby the sink node have more storage capacity to cache popular content objects of upstream traffic.

4. Performance Evaluation

4.1. Implementation and configuration

We implement TCM in Contiki [31] on the top of state-of-the-art popularity-based caching scheme, MPC [12] and CPCCS [13], namely MPC-TCM and CPCCS-TCM, respectively. We note that TCM is proposed as a complementary component to improve the performance of existing popularity-based caching schemes, not to replace them. In this implementation of TCM, a node migrates a content object c in its CS to an upper node r toward the sink if the popularity level of c, f_c, is higher than or equal to f_c1, the content popularity level of $c1$ in the CS of r. We conduct simulations using COOJA simulator [31] consisting of 1050 nodes deployed in an area with the size of 1000 x 1000 meters. Sensing correlation among nodes are generated randomly based on collected sensing data from IntelLab[32]. Content requests are generated randomly from nodes following Zipf-like distribution with a coefficient parameter of α. We vary the CS size of nodes from 5 to 25 content objects. For content requests through the sink node, we use the HTTP-CoAP converter for converting requests of consumers in HTTP to CoAP, as presented in our previous study [30]. Consumers' requests through the sink node are encoded with templates using extensible markup language

Table 1. Parameters.

Parameter	Value
Number of nodes	1050
CCA check	400 times
α	0.2 - 1
cache size p	5 – 25 objects
Wakeup interval	0.5-2.5 s
MAC	LPL
Neighborhood size	3
noise model	CPM

(XML) and are decoded with SensorML interpreter [33]. We use CTP and LPL [30] for data collection protocol and 802.15.4 MAC (Media Access Control). The radio noise model is configured using closest-fit-pattern matching (CPM) and we set CCA (clear channel assessment) [30] value up to 400 times for check parameter. We present the detailed parameter configurations for simulations in Table 1. For other parameters, we use default configurations in Contiki CC2420 radio model [30]. In default cases, we set the cache size of 20 objects, neighborhood size of 3 hops and wakeup interval of 1s if they are not specified in detail. Obtained results are reported with 96% confidence interval. We compare the performance of TCM with state-of-the-art popularity-based caching schemes incluing CPCCS [13] and MPC [12].

4.2. Performance Analysis

We evaluate TCM using the following metrics.

Average cache hit ratio (CHR): CHR is an important metric to evaluate the performance of a caching scheme. CHR measures the response rate by the in-network caching storage where content objects are cached locally. A cache hit occurs when an interest message is satisfied by a network router's cache. The router plays the role of a content provider by responding with the requested content object to the content requester. We compute average cache hit ratio as follows.

$$CHR_{average} = \frac{\sum_{i=1}^{m} c_i}{p} \quad (1)$$

where m is the total number of nodes in the network, c_i is total number of cache hit by the CS of node i and $p = \sum_{i=1}^{m} p_i$ is the total number of interest messages sent by all consumers to the network.

Average stretch ratio (ST): stretch is known as the hop distance to forward an interest packet from a content consumer toward its provider. We compute stretch ratio as follows.

$$ST_{average} = \frac{\sum_{i=1}^{I} \frac{H_i^{forwarded}}{H_i^{c-p}}}{I} \quad (2)$$

where I is total number of interest packets transmitted in the network, $H_i^{forwarded}$ is total hop count required to forward an interest packet i until it is satisfied, H_i^{c-p} is total number of hop count from the content consumer to the corresponding content producer of interest packet i.

Average content retrieval latency: indicates average time needed for interest packets to get satisfied with a content object retrieved from a cache or from its content publisher. We compute average content retrieval latency as follows.

$$L_{average} = \frac{\sum_{c=1}^{p} L_c}{p} \quad (3)$$

where L_c is the latency to retrieve content c.

Average radio duty cycle: indicates energy efficiency of mechanisms [35]. To calculate radio duty cycle of nodes, we use a counter to track and accumulate time period in each radio state of nodes. Radio duty cycle of a node is calculated as the ratio of radio active period and the cycle time period. The overall duty cycle (DC) of a node i is calculated using (4) by simply adding duty cycles for each radio operation: listening (DC_{lx}), transmitting (DC_{tx}), receiving (DC_{rx}), overhearing (DC_{over}), and additional operations (DC_{add}) [35].

$$DC_i = DC_i^{lx} + DC_i^{tx} + DC_i^{rx} + DC_i^{over} + DC_i^{add} \quad (4)$$

At the end of simulation, we compute average radio duty cycle and report average results which are calculated as follows.

$$DC_{average} = \frac{\sum_{i=1}^{m} DC_i}{m} \quad (5)$$

where m is the total number of IoT nodes.

4.3. Obtained Results

Average duty cycle. We first evaluate the energy efficiency of TCM under various wakeup intervals. We change the wakeup interval configuration for sensors from 0.5 s to 2.5s and measure the average duty cycle of nodes. Figure 1 depicts average duty cycle results of MPC, MPC-TCM, CPCCS, and CPCCS-TCM. Average duty cycle results of nodes decrease gradually when we increase the wakeup interval. This is due to the fact that at a long duty cycle, nodes can sleep more and wakeup less frequent to save energy. We obtain that TCM achieves a significant energy efficient improvement for MPC and CPCCS. At the wakeup interval of 1 s, TCM helps reduce the average duty cycle of nodes running

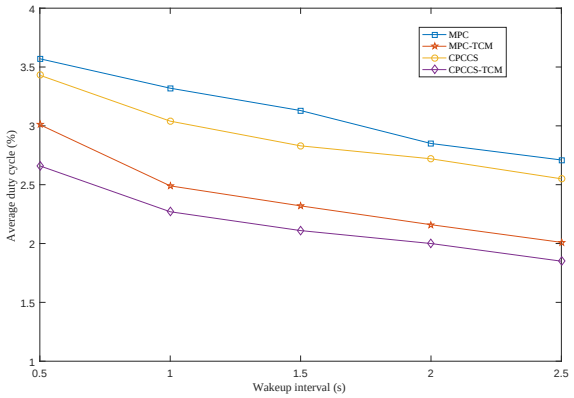

Figure 1. Average duty cycle of nodes running MPC, MPC-TCM, CPCCS, CPCCS-TCM under various wakeup intervals

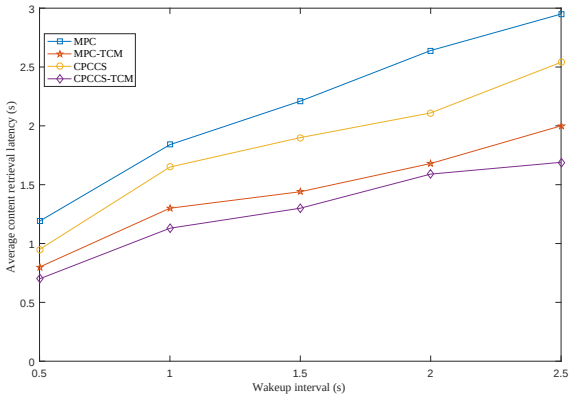

Figure 2. Average content retrieval latency of MPC, MPC-TCM, CPCCS, and CPCCS-TCM under various wakeup intervals

MPC from 3.32 % to 2.48 %, and of nodes running CPCCS from 3.04 % to 2.27 %. An interesting result is that the longer the wakeup interval is set, the greater the energy efficient improvement ratio TCM achieves. In particular, the result of MPC-TCM is lower than that of MPC 18 % at wakeup interval of 0.5 s while the result of MPC-TCM is lower than that of MPC 29.5 % at wakeup interval of 2.5 s. The reason is that energy consumption of a node to forward a message in a long wakeup interval condition is higher than that in a short wakeup interval condition, considered the same forwarding distance. At a long wakeup interval condition, the sender has to wait and be awake for a longer time period to pass the message to the next hop. By coordinating popular content objects to be cached nearby the sink node, TCM helps increase the number of interest messages that are satisfied by nodes nearby the sink, thus shorten average forwarding distance for interest messages.

Average content retrieval latency. Figure 2 presents average content retrieval latency of MPC, MPC-TCM, CPCCS, and CPCCS-TCM under various wakeup intervals. The figure shows that TCM also helps reduce average content retrieval latency of MPC and CPCCS significantly. In particular, at the wakeup interval of 1 s, average content retrieval latency in the case of MPC-TCM is 1.30 s compared to that of MPC is 1.84 s, and average content retrieval latency in the case of CPCCS-TCM is 1.13 s compared to that of CPCCS is 1.65 s. We observe the similar pattern of average content retrieval latency results compared to the previous figure when the wakeup interval increases from 0.5 s to 2.5 s. The longer the wakeup interval is used, the greater the content retrieval latency improvement ratio TCM achieves. This is due to the fact that forwarding latency of interest messages and data messages is much longer when we increase the wakeup interval of nodes. By reducing the number of forwarding hops of messages, TCM helps reduce average content retrieval latency significantly.

Average cache hit ratio. We now fix the wakeup interval of 1 s and vary the number of cache size to investigate average cache hit ratio. Figure 3 presents average cache hit ratio results obtained with MPC, MPC-TCM, CPCCS, and CPCCS-TCM. Cache hit ratio results increase gradually when the cache size is increased. For all cases, TCM achieves a significant improvement of cache hit ratio compared to MPC and CPCCS. The figure shows that CPCCS-TCM witnesses the highest cache hit ratio while MPC experiences the lowest result. The cache hit ratios of MPC, MPC-TCM, CPCCS, and CPCCS-TCM at the cache size of 20 objects are 23.6 %, 31 %, 28.9 %, and 34 %, respectively. We observe that the lower the cache size is used the higher the cache hit ratio improvement TCM achieves. In particular, at the cache size of 5 objects, MPC-TCM and CPCCS-TCM achieve around 28.4 % in term of cache hit ratio improvement compared to MPC and CPCCS while the cache hit ratio improvement is just around 19 % at the cache size of 20 objects. The reason is that TCM effectively coordinates content stores of nodes to increase their content diversity and pushes popular content objects closer to consumers. Therefore, TCM migrates more and more popular content objects to nearby consumers within a limited storage capacity of nodes. When storage capacity of nodes is higher, they can store more content objects in their CS, so cache hit ratio increases automatically, thus the improvement ratio of TCM also decreases.

Average stretch ratio. The average stretch ratio is measured as the ratio between (1) the actual hop distance required for an interest packet to be forwarded from a content consumer to provider and (2) the hop count from a content consumer to corresponding

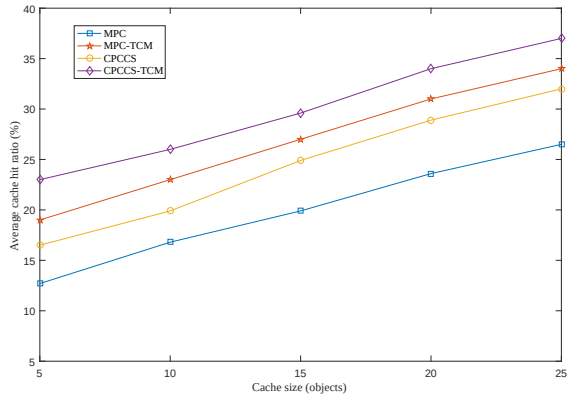

Figure 3. Average cache hit ratio of MPC, MPC-TCM, CPCCS, and CPCCS-TCM under various cache sizes

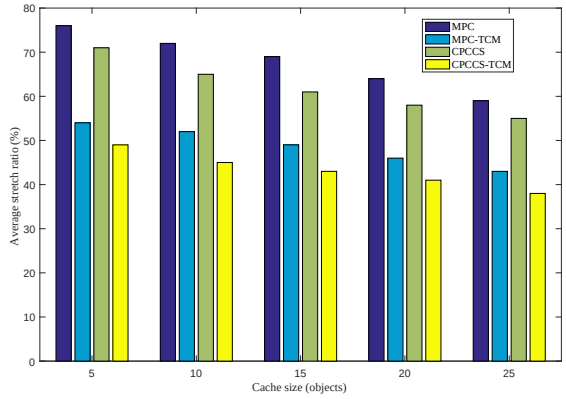

Figure 4. Average stretch ratio of MPC, MPC-TCM, CPCCS, and CPCCS-TCM under various cache sizes

content producer. We note that a content provider in ICN can be any intermediate router. Figure 4 shows the stretch ratios of MPC, MPC-TCM, CPCCS, and CPCCS-TCM under various cache sizes. Comparing figure 4 and figure 3, we find that stretch ratio results increase gradually when we increase the cache size, proportionally with cache hit ratio results. TCM improves stretch ratio of MPC and CPCCS in all cases. This indicates that TCM improves not only average cache hit ratio but also average forwarding distance for interest and data messages. An interesting observation is that the improvement ratio of TCM in term of stretch ratio is higher than the improvement ratio of TCM in term of cache hit ratio. The reason is that TCM pushes popular content objects to be cached nearby the sink, so the forwarding distance of interest and data message is optimized.

5. Discussion and Conclusions

This paper proposes to an efficient traffic-aware caching mechanism (TCM) for ICN in wireless sensor networks. The proposed caching mechanism pushes unique popular upstream content objects to be cached nearby the sink node. Less popular content are placed farther from the sink compared to popular content. TCM also pushes down popular downstream content objects to be cached inside the network. Through experiments, we find that these two strategies can be implemented on the top of two existing state-of-the-art popularity-based caching schemes such as MPC and CPCCS to enhance their network performance and energy saving significantly.

5.1. Acknowledgements

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2021-2017-0-01633) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

References

[1] T. Mick, R. Tourani and S. Misra, "LASeR: Lightweight Authentication and Secured Routing for NDN IoT in Smart Cities," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 755-764, April 2018, doi: 10.1109/JIOT.2017.2725238.

[2] Zhang, Y.; Xiong, Z.; Niyato, D.; Wang, P.; Han, Z. Information Trading in Internet of Things for Smart Cities: A Market-Oriented Analysis. IEEE Network 2020, 34, 122–129.

[3] Akhtar, N.; Khan, M.A.; Ullah, A.; Javed, M.Y. Congestion Avoidance for Smart Devices by Caching Information in MANETS and IoT. IEEE Access 2019, 7, 71459–71471.

[4] Amadeo, M.; Campolo, C.; Iera, A.; Molinaro, A. Information Centric Networking in IoT scenarios: The case of a smart home. 2015 IEEE International Conference on Communications (ICC), 2015, pp. 648–653

[5] Nagaraj, A.H.; Tahiliani, M.P.; Tandur, D.; Satheesh, H. Leveraging Named Data Networking for Industrial Automation: Opportunities and Challenges. 2020 IEEE International Conference on Communications Workshops (ICC Workshops), 2020, pp. 1–6.

[6] Bouk, S.H.; Ahmed, S.H.; Kim, D.; Song, H. Named-Data-Networking-Based ITS for Smart Cities. IEEE Communications Magazine 2017, 55, 105–111.

[7] Ioannou, A.; Weber, S. A Survey of Caching Policies and Forwarding Mechanisms in Information-Centric Networking. IEEE Communications Surveys Tutorials 2016, 18, 2847–2886

[8] Aboodi, A.; Wan, T.; Sodhy, G. Survey on the Incorporation of NDN/CCN in IoT. IEEE Access 2019, 7, 71827–71858.

[9] Zhang, M.; Luo, H.; Zhang, H. A Survey of Caching Mechanisms in Information-Centric Networking. IEEE Communications Surveys Tutorials 2015, 17, 1473–1499.

[10] Zhang, G.; Li, Y.; Lin, T. Caching in information centric networking: A survey. Computer Networks 2013, 57, 3128 – 3141.
[11] Zhang, M.; Hao, B.; Wang, R.; Wang, Y. A Pre-Caching Strategy Based on the Content Relevance of Smart Device Request in Information-Centric IoT.´ IEEE Access 2020, 8, 75761–75771.
[12] Bernardini, C.; Silverston, T.; Festor, O. MPC: Popularity-based caching strategy for content centric networks. 2013 IEEE International Conference on Communications (ICC), 2013, pp. 3619–3623.
[13] Naeem, M.A.; Nor, S.A.; Hassan, S.; Kim, B.S. Compound Popular Content Caching Strategy in Named Data Networking. Electronics 2019, 8.
[14] Dinh, T.; Kim, Y. N-Hop Content Store-Based Caching Policy and Routing Protocol for ICN. Proceedings of the 6th ACM Conference on Information-Centric Networking; Association for Computing Machinery: New York, NY, USA, 2019; ICN '19, p. 192-193.˘
[15] Dinh, N.T.; Kim, Y. An Efficient Content Store-Based Forwarding Scheme for Internet of Things. Sensors 2021, 21.
[16] Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking Named Content. Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies; ACM: New York, NY, USA, 2009; CoNEXT '09, pp. 1–12.
[17] Li, Z.; Simon, G. Time-Shifted TV in Content Centric Networks: The Case for Cooperative In-Network Caching. 2011 IEEE International Conference on Communications (ICC), 2011, pp. 1–6.
[18] Psaras, I.; Chai, W.K.; Pavlou, G. In-Network Cache Management and Resource Allocation for Information-Centric Networks. IEEE Transactions on Parallel and Distributed Systems 2014, 25, 2920–2931.
[19] Meng, Y.; Naeem, M.A.; Ali, R.; Kim, B.S. EHCP: An Efficient Hybrid Content Placement Strategy in Named Data Network Caching. IEEE Access 2019, 7, 155601–155611.
[20] Wang, Y.; Xu, M.; Feng, Z. Hop-based Probabilistic Caching for Information-Centric Networks. 2013 IEEE Global Communications Conference (GLOBECOM), 2013, pp. 2102–2107.
[21] Trung Q. Duong, Kyeong Jin Kim, Zeeshan Kaleem, Minh-Phung Bui, Nguyen-Son Vo, UAV caching in 6G networks: A Survey on models, techniques, and applications, Physical Communication, Volume 51, 2022, 101532, ISSN 1874-4907.
[22] Saino, L.; Psaras, I.; Pavlou, G. Hash-Routing Schemes for Information Centric Networking. Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking; Association for Computing Machinery: New York, NY, USA, 2013; ICN '13, p. 27-32.˘
[23] Anand, A.; Sekar, V.; Akella, A. SmartRE: An Architecture for Coordinated Network-Wide Redundancy Elimination. SIGCOMM Comput. Commun. Rev. 2009, 39, 87-98.˘
[24] Li, J.; Wu, H.; Liu, B.; Lu, J.; Wang, Y.; Wang, X.; Zhang, Y.; Dong, L. Popularity-driven coordinated caching in Named Data Networking. 2012 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2012, pp. 15–26.
[25] S. Guo, H.X.; Shi, G. Collaborative forwarding and caching in content centric networks. Proceedings of the 11th International IFIP TC 6 Networking Conference, Prague, Czech Republic, 2012, p. 41-55
[26] Thar, K.; Tran, N.H.; Ullah, S.; Oo, T.Z.; Hong, C.S. Online Caching and Cooperative Forwarding in Information Centric Networking. IEEE Access 2018, 6, 59679–59694.
[27] Ming, Z.; Xu, M.; Wang, D. Age-based cooperative caching in information-centric networking. 2014 23rd International Conference on Computer Communication and Networks (ICCCN), 2014, pp. 1–8.
[28] Alamer, A.; Basudan, S.; Hung, P.C.K. A Secure Tracing Method in Fog Computing Network for the IoT Devices. Proceedings of the 12th International Conference on Management of Digital EcoSystems; Association for Computing Machinery: New York, NY, USA, 2020; MEDES '20, p. 104-110.
[29] Mitzenmacher, M. Compressed Bloom filters. IEEE/ACM Transactions on Networking 2002, 10, 604–612.
[30] Dinh, T.; Kim, Y.; Gu, T.; Vasilakos, A.V. An Adaptive Low-Power Listening Protocol for Wireless Sensor Networks in Noisy Environments. IEEE Systems Journal 2017, PP, 1–12.
[31] Ahlgren, B.; Lindgren, A.; Wu, Y. Demo: Experimental Feasibility Study of CCN-Lite on Contiki Motes for IoT Data Streams. Proceedings of the 3rd ACM Conference on Information-Centric Networking; Association for Computing Machinery: New York, NY, USA, 2016; ACM-ICN '16, p. 221-222.˘
[32] Madden, S. Intel Berkeley Research Lab Sensor Networks. [Online]; Available: http://db.csail.mit.edu/labdata/labdata.html.
[33] Dinh, N.; Kim, Y. An Energy Efficient Integration Model for Sensor Cloud Systems. IEEE Access 2019, 7, 3018–3030.
[34] Dinh, N.; Kim, Y. Potential of information-centric wireless sensor and actor networking. 2013 International Conference on Computing, Management and Telecommunications (ComManTel), 2013, pp. 163–168.
[35] Dinh, T.; Kim, Y.; Gu, T.; Vasilakos, A.V. L-MAC: A wake-up time self-learning MAC protocol for wireless sensor networks. Computer Networks 2016, 105, 33–46.