Cache Coherence in Machine-to-Machine Information Centric Networks

Maroua Meddeb*‡, Amine Dhraief*, Abdelfettah Belghith*†, Thierry Monteil‡§and Khalil Drira‡
*HANA Lab, Univeristy of Manouba, Tunisia
†College of Computer and Information Sciences, King Saud University, Saudi Arabia
‡CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
§Univ de Toulouse, INSA, F-31400 Toulouse, France
Email: amine.dhraief@hanalab.org, abelghith@ksu.edu.sa, {mmeddeb, monteil, khalil}@laas.fr

Abstract—**Information-Centric Networking (ICN) is a new paradigm proposing a shift in the main Internet architecture from a host-centric communication model to a content-centric model. ICN architectures target to meet user demands for accessing the information regardless of its location. A major building block of ICNs concerns caching strategies. Concomitantly, Machine-to-Machine (M2M) technologies are considered the main pattern for the Internet of Things (IoT). Unifying M2M and ICN into a single framework raises the challenge of cache coherence. In this paper, we propose a novel cache coherence mechanism to check the validity of cache contents. We also propose a caching strategy suitable to M2M environment. Extensive experimentations are conducted to evaluate the performance of our proposals. They show that the combination of our two proposed schemes results in a notable improvement in content validity at the expenses of a certain degradation in both server hit and hop reduction ratios.**

I. Introduction

Internet was originally designed to share scarce resources among a limited number of static nodes with trusted communications channels. Efficient and scalable content distribution, mobility and security still challenge the current Internet architecture. Hence, Internet re-architecture is no more and option and has become an imperative requirement to the healthy growth of the Internet of the future.

This has lead researchers to introduce a novel paradigm namely Information Centric Networking (ICN) [7]. The cornerstone of ICN is the content which the user requests rather than the location of the node that provides this content: it is then a shift from a host-centric view of the network to a content-centric one.

In order to alleviate the pressure on the network bandwidth caused by the tremendous traffic growth, ICN provides in-network caching to distribute content in a scalable, cost-efficient and secure manner. Caching is not a revolutionary concept; it has already been widely used by the host-centric Internet in the Web, P2P systems and Content Distribution Networks (CDNs). However, with the absence of unique identification, current caching system may handle the same content at different network location as different contents. In addition, existing cache systems are designed for a specific traffic and applications, they are not generic. Furthermore, in traditional caching systems, cache nodes are predetermined and fixed, unlike in ICN caching, caches are ubiquitous. Each node can be a cache and while forwarding a content, the node can choose to cache a copy or not [7]. Unique identification, transparency and ubiquity are new features that make ICN caching more challenging than already existing cache systems.

ICN in-network caching strategies are based on two caching strategies: *On-path* and *Off-path* caching. The On-path caching strategy uses nodes/routers belonging to the reverse path pursued by the request. While forwarding a content, an intermediate router has the ability to choose between either caching the content or not and this is according to its own caching policy. The Off-path caching strategy reserve a distant cache node, so the publisher multicasts the content to both the client and the cache [6].

At a given node, the decision is made based on a caching strategy which we detail some of them in the following: (1)Leave Copy Everywhere (LCE) [7]; A copy of the content is stored in every node in the path towards the user. (2)Leave Copy Down (LCD) [7]; When a content is found at a node in the request path, a copy of that content is stored on one level down in the reverse path towards the user. (3) Probabilistic cache (ProbCache) [7]; A copy of a content is cached in a node with a probability. This probability is not constant, it is inversely proportional to the distance from the user to the server. As a consequence, nodes close to the user have a greater chance to be a cache node. (4) Betweenness Centrality (Btw) [1]; The betweenness centrality parameter is pre-calculated to each node. It measures the number of times that a node belongs to a path between all pairs of nodes in a network topology. On the way of the response towards the user, a copy of a content is cached only in nodes whit the greatest betweenness centrality.

Meanwhile, the number of sensors and actuators which can communicate through the Internet infrastructure is in constant increase and thus presaging Internet of things(IoT). Machine-To-Machine (M2M) is believed to be the main framework of IoT [4]. It refers to the technology that provides machines with the ability of interacting with each other in order to exchange information and perform actions with minimal human intervention. Using ICN in M2M networks allow us to get M2M data from any cache, not essentially from the M2M server. In this context, ICN is used to alleviate the pressure on the M2M server and make faster reactions in term of traversed hops and

978-1-4673-6770-7/15/$31.00 ©2015 IEEE

response time.

In-network caching has been proved to be quite difficult to use. According to A. Dingle et al. in [2], difficulty lies in the fact that copies of contents in a cache can be out of date. This problem is called cache coherence which is the consistency of a shared resource data stored in multiple caches.

Unlike [2], we focus on caching in ICN rather than web but the cache coherence problem remains the same. ICN cache coherence largely impacts M2M networks. As the current status of the network can change at any time, we need to check that stored state in caches are valid. To the best of our knowledge, this is the first research work which focuses on cache coherence of ICN. In the following, we focus on existing coherence mechanism used by distributed systems [2]: (1) Validation check; When a content is stored in a cache, it is marked with a timestamp. To verify the coherence, the timestamp of the content is sent to the server to be sure that the data has not been modified since it was cached. (2) Callback mechanism; The server notifies all caches when the content is updated. It is unsuitable that the server holds a list of all caches to notify them. (3) HTTP Headers; To request a content, a conditional Get is sent towards the server; an If-Modified-Since header is appended to the Get request. When the request reaches a cache, it verifies if the last modification time of the content is greater than the time in the header, in this case the content is returned. (4) "Naive" coherence; This coherence mechanism is used with an hierarchical topology. When the content is found in a cache, this latter sends a conditional GET message to the next higher cache or server. The last modification time of the cached content is passed as the If-Modified-Since header time. If the content is returned to the cache so it is considered as coherent. (5) Expiration-based coherence; Each content is marked with an expiration time. Before the expiration time elapses, the content is assumed valid and any requests made are automatically answered.

This paper is structured as follows. In section 2, we explain client-cache caching strategy and event-based coherence mechanism. We proceed to show and analyse simulation results in section 3. We finally conclude in section 4.

II. A COHERENT CACHING STRATEGY

In this section, we detail our proposed coherent caching strategy for ICN M2M networks. Our main objective is to enhance the percentage of validity of requested content while maintaining system performances.

A. Client-cache strategy

In the following, we present our caching strategy which we call *client-cache*. Client-cache design is based on three hypothesis. It is (i) an on-path caching strategy which (ii) reduces the number of cache nodes and (iii) selects caches close to users.

First of all, unlike on-path caching, off-path caching will serve only local clients. In addition, it requires additional meta-data to inform client which cache element is associated with a desired source. For this, we choose, as a first hypothesis, the on-path caching strategy. Second, in [1] authors demonstrated that caching less performs better results. Even random caching is better than LCE. This is explained by the fact that the cache size is limited and the number of contents to be cached is more important so the cache replacement error increases. In our strategy, we choose to reduce the number of cache nodes and to focus on the most important nodes. Third, experiment results in [3] show that using edge caching can save most of the caching benefits of ICN. Since users are located at network edges, important nodes are connected to clients. The third hypothesis is to cache close to users.

Considering these hypothesis, client-cache proposes to cache contents on nodes in request reverse path which have a connection with a client.

B. Event-based coherence mechanism

Event-based coherence mechanism, as its name indicates, depends on the data flows. Before explaining our proposed coherence mechanism, we firstly detail the different M2M traffic pattern.

1) M2M traffic pattern: M2M data flow are continuous, periodic or OnOff. In continuous transmission, like video streaming, content's chunks are sent continuously. In periodic transmission, the content is sent every fixed period of time. Finally, in the OnOff transmission mode, the content value is sent only when it changes (i.e: presence sensor).
This three modes can be classified in two major modes periodic and OnOff transmissions. Continuous transmission is in fact a periodic one with a very short period.

2) Event-based coherence algorithm: Our proposed event-based coherence algorithm is an expiration-based coherence where the expiration time is not constant.

The validity of a content is checked at the moment when a requested content is found in a cache. A content is considered valid when the lifetime of its version in the cache is lower than the lifetime of its last version in the source. With the time of storing a content in a cache *cache-time* we can calculate for how long a content is cached ($Now - cache\_time$).

In the case of a periodic transmission, the lifetime in the origin source can be easily calculated since the update of data is periodic with a constant period T. So, the period must be appended to the content. We note by Tv this lifetime and $Tv = Now - T * E(Now/T)$. As we have already mentioned, the content is considered valid when $Tv > Now - cache\_time$, then it is returned to the client.

In the case of an OnOff transmission, the sensor behavior is not predicted and values updates can be performed at any time. For this reason, when a request matches a data in a cache, we can not deduce the time of last happened events in the source to verify the validity. To do so, we propose to store, for each content in the source, the time of happened events in a list $lastEvents$. As a concequence, we will have past events and we can predict the time of the future one which we note $Tevent$. If $Tevent > Now$, that means that pearhaps an update is performed in the source. The content is so considered invalid.

There are two time series models designated for prediction which are ARMA (Autoregressive Moving Average) model and Exponential smoothing. To predict future value, the exponential smoothing takes into account all past data while ARMA can only use k past data points. In addition, the exponential smoothing is poor in long-term precision in comparison with ARMA which is consistent even with long series. Finally, ARMA can be used with all type of series while the exponential smoothing is used only on adjusted series. In our proposal we used the ARMA model.

III. Performance evaluation

In this section, we describe the adopted metric, the simulation scenario and the obtained results.

A. *Performance metrics*

In our performance evaluation, we measure the hop reduction ratio, the server hit reduction ratio and the response latency against the Zipf distribution [3] coefficient.

The hop reduction ratio A represents how faster, in term of hops, a content is fetched from a cache than from the server. It is analytically represented by Eq. 1. Each client i sends R requests. For each request r from i the hop reduction ratio is calculated. It is the ratio of the path length from the client to the first content cache h_{ir} over the path length from the client to the server H_{ir}.

$$A(\alpha) = \frac{\sum_{i=1}^{N} \frac{\sum_{r=1}^{R} \frac{h_{ir}(\alpha)}{H_{ir}(\alpha)}}{R}}{N} \tag{1}$$

The server hit reduction ratio B expresses the alleviation of the server load. It is the ratio of the number of requests satisfied by the server $serverhit$ over the totality of requests, which represent requests satisfied by the server and by a cache (see Eq. 2).

$$B(\alpha) = \frac{\sum_{i=1}^{N} serverhit_i(\alpha)}{\sum_{i=1}^{N} serverhit_i(\alpha) + cachehit_i(\alpha)} \tag{2}$$

The response latency is the duration between the delivery of a content request from a client and the response, knowing that the transmission delay between nodes are randomly set.

The last metric is used to evaluate the coherence mechanism. It is the validity percentage of contents. We calculate the ratio of the number of valid contents over all provided contents.

B. *Simulation scenario*

S. K. Fayazbakhsh et al. have experimentally proved in [3] that the request popularity distribution across different geographical locations is close to a Zipfian distribution. In our simulation, content requests are generated following a Zipf distribution where α varies in a rather wide range $[0.5; 2.5]$ [5].

For our simulations, we choose the cache size $C = 1000$ chunks, the file number $\mid F \mid = 10^4$ files and the file size $F = 1$ chunk. We set $\lambda = 1$, the cache replacement policy is LRU (Least Recently Used) and the forwarding strategy is SPR (Shortest Path Routing). We consider a scale-free topology following the Barabasi-Albert (B-A) model with N=100 nodes, m_0=3 and m=2. Concerning the transmission delay, we are based on the transmission delays used in real network topology. We note that delay $\in [0; 20]\, ms$

C. *Results collection and analysis*

Our objective is not to have a better caching strategy but to propose a coherent caching strategy which maximizes the percentage of contents validity. Fig.1 confirms the advantage of using event-based coherence mechanism.

We observe from Fig. 1a, that our coherent client-cache strategy provides the best validity results with 98%. We also notice that even without coherence mechanism, the result with client-cache is not so bad as with other strategies, the validity is over 61%. We can see that the percentage of validity with other caching schemes is very low. In Fig. 1a, we report from left to right, 40% under the LCE strategy, 34% using LCD, 31% with ProbCache strategy and 51% with Btw strategy. We conclude that under our coherent client-cache caching strategy, almost all requested contents are valid, however with other strategies, it is unlikely to receive a coherent data.

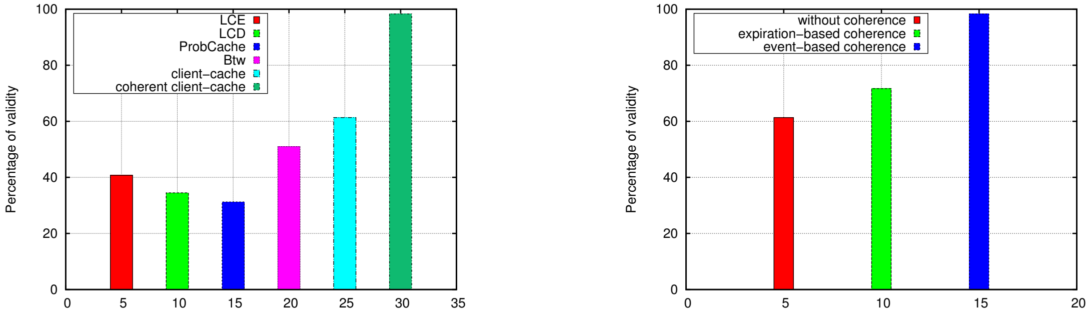
Fig. 1b portrays the results of client-cache strategy with different coherence mechanisms. These results confirm the performance of our event-based coherence mechanism. In fact, using the expiration-based coherence mechanism has improved the content validity with 71% but our coherence mechanism still performs better results with 98%. As a deduction, we can say that even with the existing coherence mechanism, the probability to get a valid response is not sufficient.

Now, we evaluate these caching strategies without the coherence mechanism and our coherent caching strategy. Results are depicted in Fig. 2. We reminder that these simulations have been carried out to evaluate the server-hit reduction, hop reduction and responce latency against the variation of the popularity of contents.

In Fig. 2a, we see the server-hit performance. The difference between simple caching strategies balances around a reduction of 4-10% for client-cache against the ProbCache strategy, while, it is smaller compared to LCD and Btw. With this two latter strategies, results are the same with respectively $\alpha \geq 1.75$ and $\alpha \leq 1.5$. Using LCE, the reduction is of 3% better. With the coherent client-cache strategy, the reduction decreases against the simple client-cache strategy by 10-40%.

The hop reduction results are showed in Fig. 2b. We notice that the more popular the content is the faster it is found. With $\alpha \geq 2$, the content is found in the first hop towards the server. The difference is roughly 4-6% against ProbCache. LCD and LCE perform better results with a reduction of 12-15% while results are similar to Btw. Similar to the first metric, the hop reduction diminishes using coherent client-cache by 10-15%.

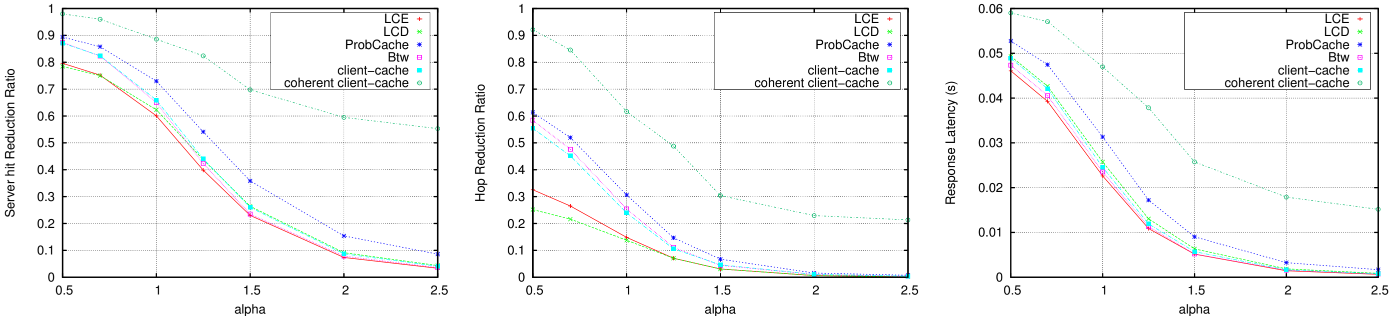
Client-cache reduces the response latency of 3% in comparison to ProbCache, whereas, results are very similar with other strategies Fig. 2c. We report that we lose arround 0.01 s of the response time under the coherent client-cache strategy.



(a) Validity % with different caching strategies

(b) Validity % with client-cache strategy

Fig. 1. System performances with different chaching strategies



(a) Server hit reduction ratio

(b) Hop reduction ratio

(c) Response latency

Fig. 2. System performances with different chaching strategies

We conclude that there is degradation in system performance with our proposal, but it still better than without caching and the degradation is very small. In other words, our strategy like other strategies reduces the number of traversed hops and the server hits, in addition, it increases the percentage of validity. In term of coherence, it is better to make one extra hop in order to have a valid content.

IV. Conclusion

In this paper, we have outlined the in-networking caching in information-centric M2M networks, emphasizing its support for cache coherence mechanism. We have explained the impact of the lack of mechanisms to check content's coherence, especially in M2M networks. We have proposed an event-based coherence mechanism, an algorithm that verifies if a cached content is valid or not and it is based on M2M data flow. In addition, we have suggested to join to this mechanism a client-cache caching strategy. Our caching scheme propose to store a copy of a content where there is a client attached to that node. We report that the combination of our two propositions provides users with 98% of valid contents, although only around 40% using other caching strategies without coherence mechanism and approximately 60% with expiration-based coherence. In our future work, it is intended to improve existing caching strategies with our coherence mechanism and to compare the network resources consumed in different caching strategies. As a challenge we propose to focus on name-based routing in ICN. We aim to conceive a routing strategy that look for the nearest coherent cache.

References

[1] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," in *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I*, ser. IFIP'12. Springer-Verlag, 2012, pp. 27–40.

[2] A. Dingle and T. Pártl, "Web cache coherence," *Comput. Netw. ISDN Syst.*, vol. 28, no. 7-11, pp. 907–920, May 1996.

[3] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 147–158, Aug. 2013.

[4] M. Meddeb, M. B. Alaya, T. Monteil, A. Dhraief, and K. Drira, "M2m platform with autonomic device management service," *Procedia Computer Science*, vol. 32, no. 0, pp. 1063 – 1070, 2014, the 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014).

[5] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing," Telecom ParisTech, Tech. Rep., 2011.

[6] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. Siris, and G. Polyzos, "Caching and mobility support in a publish-subscribe internet architecture," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 52–58, July 2012.

[7] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128 – 3141, 2013, information Centric Networking.