Evaluation of a Redirection Technique in Cooperative Caching for MANETs

F.J. González-Cañete, E. Casilari
Dpto. Tecnología Electrónica
University of Málaga
Málaga, Spain
+34 952 13 71 76
{fgc, ecasilari}@uma.es

ABSTRACT
In this paper, the performance improvement of a Mobile Ad Hoc Network that implements a caching technique based on the redirection of the requests is evaluated. The redirection technique is implemented over a cooperative caching scheme that utilizes both local and redirection caching strategies. For this purpose, the proposed scheme stores information about the location of the documents in the wireless network analyzing the forwarded request and reply messages. By using this information, the documents requests can be redirected to mobile nodes that are known to be closer than the final data server. By means of simulations, the redirection technique is shown to improve the efficiency of the caching process.

Categories and Subject Descriptors
I.6.5 [***Simulation and Modeling***]: Model development C.2.1 [***Computer Communication Networks***]: Network architecture and design

General Terms
Management, Measurement, Performance.

Keywords
Cooperative caching, redirection caching, mobile ad hoc networks, replacement policy, performance evaluation.

1. INTRODUCTION
The aim of a cooperative caching scheme for MANETs (Mobile Ad hoc NETworks) is to take advantage of the cooperation among the mobile nodes in order to increase the performance of the wireless network. Mobile nodes have to collaborate in order to forward the messages from the source to the destination node in a multihop way. This cooperation is carried out by the routing protocols. Taking into consideration that every node from the source to the destination of a request or reply message has to analyze the corresponding packet in order to take a routing decision, the mobile nodes can also store information about where the requested documents are located in the network. Consequently, this information could be used to request the documents to mobile nodes that are known to be located closer than the original destination of a request (normally a data server). Furthermore, a mobile node that receives a document request to be re-forwarded could decide to redirect the request to another node (instead of sending it to the original destination).

PE-WASUN 2012, October 21–25, 2012, Paphos, Cyprus.

As the wireless medium is bandwidth-constrained, it is desirable to reduce the generated traffic load while maintaining (and even increasing) the documents' accessibility. Thus, the redirection caching technique is an appropriate option in order to obtain these objectives. In addition, the redirection technique also reduces the delay to obtain the documents as the requests are forwarded to closer nodes that are most probably accessed in less time than the original destination node.

The goal of this paper is to evaluate the performance improvement that the redirection caching technique adds to a cooperative caching scheme that includes local caching and redirection caching.

The rest of this paper is structured as follows. Section II presents some caching schemes that utilize the redirection caching technique. In section III, the caching scheme that redirects the requests is described. Section IV details the simulation model. Section V evaluates the performance improvement of a MANET when the redirection caching technique is employed. Finally, section VI outlines the main conclusions and suggests possible future work.

2. RELATED WORK
The redirection technique has been employed by some caching schemes for MANETs, although the implementation is quite different for each case.

The Distributed Greedy Algorithm (DGA) presented in [1] proposes to store, for all the documents in the network, the location (i.e. an identifier such as the IP address) of the closest node (in number of hops) and the second closest node that has a copy of the document. This information is obtained by analyzing the request and reply messages that are forwarded by the nodes. The cooperative technique COOP [2] and the caching scheme proposed in [3] behave in an analogous way but only storing the information of the closest node. Similarly, the caching services CachePath and HybridCache [4] save the location of the documents if they are stored in a node that is closer than the server. An additional condition to save this location imposes that the distance to the server (expressed in number of hops) minus the

distance to the node that stores the document has to be greater than a certain threshold.

Under the COACS (COoperative and Adaptive Caching System) [5] caching scheme, the mobile nodes are divided into QDs (Query Devices) and CNs (Caching Nodes). The QDs store the information about the location of the documents by studying the requests executed by the CNs. In that way, the CNs request the documents to its closest QD. If this QD knows where the document is stored, the request is redirected. Otherwise, the request is sent to another QD. If all the QDs have been consulted and the document has not been found, the request is sent to the server.

GroupCaching [6] proposes to form groups of nodes, where each group is composed by the nodes located at a one-hop distance. All the nodes implement a table called *group_table*. For every requested document, the table creates an entry with the following fields: *data_id, data_source, group_member_id,* and *timestamp*, where *data_id* is the document identification, *data_source* is the identifier of the node that served the document, *group_member* is the identifier of the node that has a copy of the document and *timestamp* is the time when the document was stored. In order to keep the *group_table* updated, the nodes periodically send a message to the rest of the nodes in the group informing about the documents that it stores and the remaining available cache space. Similarly, IXP (IndeX Push) [7], also utilizes a table called *IV* (Index Vector) with the fields *x, cached, cachednode* and *count*, where *x* is the document identification, the *cached* boolean variable indicates if the document is stored in the local cache, *cachednode* identifies the node that has a copy of the document and *count* describes the number of copies of the document x in the neighborhood. Whenever a node stores or deletes a document from its local cache, a message is sent to the neighbor nodes in order to update the *IV* tables. The DPIP (Data Pull/Index Push) [7] caching scheme is an evolution of IXP that proposes to update the *IV* tables using the document request instead of broadcast messages.

3. CACHING IN MANETs

In this section we define the formulation of a caching scheme for MANETs. The mathematical notation follows a modified version of that presented in [8].

Let $MN = \{w_1, w_2, ..., w_w\}$ be the set of w mobiles nodes in a MANET. Let $U = \{d_1, d_2, ..., d_n\}$ be the universe of n documents that can be requested by the mobile nodes, where *s(i)* represents the size of the document *i* with $1 \le i \le n$. *TTL(i)* is defined to be the time when the document *i* expires and becomes obsolete. $R_j = \{r_{j1}, r_{j2}, ..., r_{jm}\}$ is defined to be the sequence of requests performed by the node *j*, where r_{jk} denotes the request of mobile node *j* in the *k* instant taking into account that $r_{jk} \in U$. The model assumes that the destination of the requests in R_j is a fixed node w_{DS} (Data Server) that has access to all the documents in *U*.

3.1 Local caching

The local cache allows that future requests to the same document will be served by the local cache even if the server is not available. Thus, the request may not produce any traffic in the ad hoc network. B_{jk} denotes the set of documents stored in the cache of node *j* at time *k*, where $B_{jk} \subset U$. The set of documents stored in the caches must satisfy the properties expressed in equations (1) and (2):

$$\sum_{i \in B_{jk}} s(i) \le S_j \tag{1}$$

$$\forall i \in B_{jk} \Rightarrow TTL(i) > k \tag{2}$$

where S_j is the size of the cache in node *j*. Therefore, the sequence $(B_{j0}, B_{j1}, ..., B_{jm})$ indicates the states of the cache in node *j* as the requests in R_j are resolved. B_{j0} is the initial state when the cache is empty and B_{jm} is the state when all the requests have been served.

Let $p_{k,ij} = \{w_i, w_u, w_v, ..., w_j\}$ be the active route between the nodes *i* and *j* at instant *k*, defined as the set of mobile nodes in the route from node *j* to node *i* at the instant *k*. Only two consecutive nodes in a path are directly connected, that is, a wireless link between them is available. If $p_{k,ij} = \varnothing$ then there is no route created to reach node *j* from node *i*, otherwise $card(p_{k,ij}) \ge 2$, where $card(p_{k,ij})$ is the cardinality of the set $p_{k,ij}$. We can define the distance between node *i* and *j* at the instant *k* as (3):

$$dist(w_i, w_j)_k = card(p_{k,ij}) - 1 \tag{3}$$

This distance defines the number of hops needed to reach node *j* from node *i* at the instant *k*. When the distance between nodes *i* and *j* is one, the nodes are considered to be neighbors. On the other hand, we consider that $dist(w_i, w_j)_k = \infty$ if there is not a route created between the nodes *i* and *j* at the instant *k* ($p_{k,ij}$=∅).

In addition, we define the local cache hit sequence in node *j* to be $(h^l_{j1}, h^l_{j2}, ..., h^l_{jm})$ where h^l_{jk} is defined as:

$$h^l_{jk} = \begin{cases} 1 & \text{if } r_{jk} \in B_{j(k-1)} \\ 0 & \text{if } r_{jk} \notin B_{j(k-1)} \end{cases} \tag{4}$$

When h^l_{jk} equals *1* a local cache hit is considered in node *j*, otherwise a local cache miss is assumed as the document required is not in the local cache at that moment.

The cache replacement policy decides which documents must be evicted from the cache in order to make room for the new ones. The goal of the caching scheme is to retain in the cache the documents that have a higher probability of being requested again in the near future.

Given a request sequence R_j in node *j*, a cache of size S_j and an initial state B_{j0}, the replacement policy produces a cache state sequence $(B_{j1}, ..., B_{jm})$. For *k*=1,…,*m*, we have :

$$B_{jk} = \begin{cases} (B_{j(k-1)} - \varepsilon_{jk}) \cup \{r_{jk}\} & \text{if } r_{jk} \notin B_{j(k-1)} \\ B_{j(k-1)} & otherwise \end{cases} \tag{5}$$

In the first case considered in the previous equation, the requested document was not present in the cache of node *j* (cache miss) at the instant of the request. The term ε_{jk} (with $\varepsilon_{jk} \subset B_{j(k-1)}$) denotes the set of documents that have to be evicted from the cache in order to make room for the new one in the request r_{jk}. If there is enough room for the new document, ε_{jk} is an empty set. In

the second case there is a local cache hit and the cache remains unchanged.

3.2 Interception caching

The next step compels the mobile nodes to cooperate between them in order to respond to the requests of the other nodes by using their local caches. The forwarding nodes in the route from the requester to the destination nodes are provided with the ability to directly reply with the document if they have a valid copy in their local cache. The interception of the requests reduces the number of hops necessary to obtain the document and hence the number of forwarding messages along the network.

We define the interception cache hit sequence in node j to be $(h^i_{j1}, h^i_{j2}, ..., h^i_{jm})$ where h^i_{jk} is defined in (6) while w_i identifies an intermediate wireless node that has a copy of the document requested in its local cache. The node w_i is located in the route of the request r_{jk} from the node j to *DS*.

$$h^i_{jk} = \begin{cases} 1 & \text{if } r_{jk} \notin B_{j(k-1)} \wedge r_{jk} \in B_{i(k-1)} \\ & \quad | \, i \neq j \wedge i \neq DS \wedge w_i \in p_{k,jDS} \\ 0 & otherwise \end{cases} \qquad (6)$$

When $h^i_{jk}=1$ an interception cache hit is considered to have occurred in node j at instant k.

3.3 Redirection caching

Previous redirection caching schemes do not take into consideration the TTL of the documents or the replacement of the documents in the local caches. This could be a serious problem as the requests can be redirected to nodes that do not actually store the requested document as they have been evicted from the remote cache.

Every entry in the redirection cache stores the fields *id*, *IP_GET*, *hops_GET*, *TTL_GET*, *IP_RESP*, *hosp_RESP*, *TTL_RESP*, where *id* is the identification of the document, *IP_GET* and *hops_GET* are the IP address (i.e. the identification of a mobile node) and the distance in hops to the requester node and *TTL_GET* is the TTL of the document. Similarly, *IP_RESP*, *hosp_RESP* and *TTL_RESP*, stores the same information about the response node (if it is not a DS). Consequently, the nodes in the wireless network analyze the forwarding requests and responses to facilitate the update of the information stored in its redirection cache. In order to take into consideration the eviction of documents in the local caches, every node calculates the mean time the documents are stored in its own local cache. In that way, the TTL stored in the redirection cache will be the lowest value between the TTL of the document and the mean time that is supposed to be stored in the remote caches.

When a node receives a request to be forwarded, it first looks up in the redirection cache if there is a node with a valid copy of the document that is located closer than the destination of the request. If so, the request is redirected to that node. In order to avoid redirection loops, a message can only be redirected once. Additionally, the redirection is performed only if at least one of the associated TTL (*TTL_GET* or *TTL_RESP*) is known.

In those cases when the redirected message reaches its destination but this node has evicted the document of its local cache, a *REDIRECTION_ERROR* message is sent to the redirector node. This node will update its redirection cache deleting the associated information that caused the redirection error.

Formally, let BR_{jk} be the information about the location and TTL of some documents in the universe U stored in node j at the instant k. BR_{j0} will be the initial state of the redirection cache, that is, when the redirection cache is empty. We define the sequence of redirection hits in node j as $(h^{rd}_{j1}, h^{rd}_{j2}, ..., h^{rd}_{jm})$ where h^{rd}_{jk} is defined in (7):

$$h^{rd}_{jk} = \begin{cases} 1 & if \quad \begin{cases} \left(r_{jk} \notin B_{j(k-1)}\right) \wedge \left(r_{jk} \notin B_{i(k-1)}\right) \\ \wedge \left(r_{jk} \in BR_{i(k-1)}\right) \wedge \left(r_{jk} \in B_{r(k-1)}\right) \\ | \left(i \neq r \neq DS\right) \wedge \left(j \neq r\right) \wedge \left(w_i \in p_{k,jDS}\right) \\ \wedge \exists p_{k,ir} \wedge \exists p_{k,rj} \\ \wedge \left(dist(w_i, w_r)_k < dist(w_i, w_{DS})_k\right) \end{cases} \\ 0 & otherwise \end{cases} \qquad (7)$$

where w_i represents an intermediate node in the route of the request r_{jk} from node j to *DS*. This intermediate node w_i does not have a copy of the requested document in its local cache although it knows that node w_r has the document. Thus, node w_i redirects the request to node w_r. The redirection will be performed if a route between nodes i and r can be established and the distance in hops from i to r is shorter than the distance from i to DS, that is, $dist(w_j, w_r)_k < dist(w_j, w_{DS})_k$. When $h^{rd}_{jk}=1$ a redirection cache hit in node j is considered. The redirection cache hits are expected to reduce the delay perceived as the documents are served from a closer node.

The redirection caching structure is managed using two LRU lists: the *KNOWN_TTL* list contains the entries related to documents whose TTL values are known while the *UNKNOWN_TTL* list contains the entries to documents with unknown TTL values. The space for each structure is dynamically assigned but the memory space reserved for both structures is set to a constant value. If an entry of the *UNKNOWN_TTL* list is updated with the TTL of a document, it will be transferred to the head of the *KNOWN_TTL* list. When a new entry must be stored and there is not enough room because the reserved storage space is full, the oldest entry in the *UNKNOWN_TTL* list is evicted. If this list is empty, the oldest entry in the *KNOWN_LIST* will be deleted. An entry is also deleted if the information is obsolete because all the associated TTLs have expired.

4. SIMULATION MODEL

By means of simulations we have evaluated the influence and benefits of the redirection caching technique proposed in section III. The simulations are based on the network simulator NS-2.33 [9] which is the most popular simulator for the research on ad hoc networks [10].

Table 1 summarizes the main simulation parameters. We will consider W=50 mobiles nodes distributed in a 1000 meters by 1000 meters area. The mobile nodes follow the Random Waypoint mobility pattern [11] moving at a constant speed of 1 m/s with no pause time when they reach the destination.

We consider n=1000 different documents (identified by a specific number) distributed between two fixed (immobile) servers located at positions (x,y)=(0 m, 500 m) and (x,y)=(1000 m, 500 m) in the simulation area respectively. To distribute the traffic, documents with an odd identifier are placed in a server while even-numbered

documents are located in the other one. We have considered an exponential distribution with a mean between 250 and 2000 seconds for the TTL of the documents. In that way we model the system for both a high and low variability of the documents. In addition we also consider the case of an infinite TTL for the documents, that is to say, the case where the documents never expire. The size of all the documents is constant and set to 1000 bytes.

Table 1. Simulation parameters

Parameter	Default	Tested values
Simulation area (meters)	1000x1000	
Number of nodes	50	
Number of Documents	1000	
Documents size (bytes)	1000	
Number of requests per node	10000	
Simulation time (s)	20000	
Timeout (s)	3	
TTL (s)	2000	250-500-1000-2000-∞
Mean time between requests (s)	25	5-10-25-50
Traffic pattern (Zipf slope)	0.8	0.4-0.6-0.8-1.0
Replacement policy	LRU	
Cache size (number of documents)	35	10-20-35-50
Redirection cache size (number of registers)		10-25-50-75
Ad hoc routing protocol	AODV	
Mobility pattern (Random WayPoint)	Speed: 1m/s Pause time: 0 s	1-3-5 m/s
Warm-up (s)	4000	
Simulation time (s)	20000	

Every node that is not a server generates requests to the servers along the simulation time. When a request is served another request is generated by the same node. The idle time between the reception of a response and the next request has been modelled using an exponential distribution with a mean between 5 and 50 seconds. Using this set of values we evaluate a wide range of patterns for the node activity and, consequently, for the network load. A document is requested again if the response of the present request is not served before a defined timeout is triggered.

The pattern of requests of the documents follows a Zipf-like distribution that has been demonstrated to properly characterize the popularity of the Web documents in the Internet [12]. The Zipf law asserts that the probability *P(i)* for the *i-th* most popular document to be requested is inversely proportional to its popularity ranking as shown in eq. (8):

$$P(i) = \frac{\beta}{i^{\alpha}} \tag{8}$$

The parameter α is the slope of the log/log representation of the number of references to the documents as a function of its popularity rank (i) while the β parameter is the displacement of the function. In our simulations, the slopes selected to generate the requests are 0.4, 0.6, 0.8 and 1.0.

Finally, every node implements a local cache that employs the Least Recently Used (LRU) replacement policy. This replacement policy evicts the documents that were referenced longest ago. All nodes have the same cache size, which has been configured to fit 10, 25, 35 or 50 documents (depending on the simulated scenario). In order to avoid cold start influences, that is, cache misses because the cache is empty [13], the local caches are "warmed up" using the first 20% of the simulation time, which has been set to 20000 seconds.

5. PERFORMANCE EVALUATION

Each scenario has been executed five times using the same configuration parameters but with different seeds. The presented evaluation metrics are the mean of the results obtained for the five simulations. All the presented figures include the 95% confidence interval for every measured parameter.

As performance metrics we define the local hit ratio (LHR), the interception hit ratio (IHR) and the redirection hit ratio (RDHR) expressed as:

$$LHR = \frac{1}{w}\sum_{j=1}^{w}\frac{\sum_{k=1}^{m} h_{jk}^{l}}{card(t_j)}. \tag{9}$$

$$IHR = \frac{1}{w}\sum_{j=1}^{w}\frac{\sum_{k=1}^{m} h_{jk}^{i}}{card(t_j)} \tag{10}$$

$$RDHR = \frac{1}{w}\sum_{j=1}^{w}\frac{\sum_{k=1}^{m} h_{jk}^{rd}}{card(t_j)} \tag{11}$$

where $t_j \subset R_j$ represents the subset of requests that the node j has performed until the end of the simulation time t (with $t \leq m$).

Figure 1 shows the cache hits as a function of the speed of the nodes. The local cache hits do not change with the speed; however the interception cache hits are greater for a speed of 1 m/s. On the other hand, the redirection cache hits are slightly increased with the speed. In addition, a redirection cache of 50 registers obtains the same performance as the one with 75 registers. In the best case, the redirection cache hits reach the 5%.

Figure 2 depicts the cache hits as a function of the number of nodes in the network. For small networks with 25 nodes the local and interception hits are smaller than those obtained in more dense topologies. However, the redirection cache hits are greater. For networks with more nodes the redirection caching hits are about 5% for redirection caches with more than 50 registers.

Figure 3 represents the cache hits as a function of the mean time between requests. As the time between requests increases (the network load is decreased) the number of local and interception hits is decreased. Nevertheless, the redirection cache hits are increased. This is due to the fact that, as the traffic is diminished, the documents stored in the local cache and the information stored in the redirection caches do not change very often and the redirection cache maintains more up-to-date information.

As the TTL of the documents rises they are expected to stay longer in the local caches. Consequently, the number of cache hits will be increased. Figure 4 shows this behaviour. As in previous studies, a redirection cache of 50 entries is enough to obtain the best performance (about 5%).

Figure 5 represents the cache hits as a function of the Zipf slope. As the Zipf slope increases, the most popular documents will be requested more frequently. This behaviour implies that the local caches will obtain more local hits as the Zipf slope rises. The interception and redirection hits follow the same behaviour, obtaining the redirection cache about 5% of cache hits for the maximum slope.

Finally, Figure 6 shows the cache hit ratio as a function of the cache size. As the cache size is incremented, the number of documents and time that they are stored is increased. For very small caches (only 10 documents) the replacements in the cache occur very frequently, thus, it is very difficult to obtain redirection hits. As the cache size increases, all the metrics are incremented, including the redirection cache hits, which obtain a 6% for the cache size of 50 documents. As in the case of the other studied variables, redirection cache sizes of 50 and 75 documents obtain similar results.

6. CONCLUSIONS

In this paper we have evaluated the performance of a redirection technique for a cooperative caching scheme for MANETs. This technique stores information about the location of the documents in the wireless network analyzing the messages that are forwarded by the mobile nodes. As a novelty, this redirection technique takes into consideration the TTL of the documents as well as the estimated time that they are being stored in the remote caches.

We have evaluated the redirection cache hits (as well as the local and interception cache hits) as a function of the speed of the nodes, number of nodes in the network, mean time between requests (traffic load), the TTL of the documents, the traffic pattern (Zipf slope) and the cache sizes. All the evaluation tests conclude that the redirection caching technique achieves, in the best cases, about a 5% of redirection cache hits. Thus, the redirection caching slightly increments the performance of the network for all the studied parameters. Additionally, we have evaluated various redirection cache sizes, concluding that a cache size greater than 50 documents does not obtain better results.

As a future research topic we can propose to evaluate the influence of other parameters (such as the delay, the number of retrieved documents or the generated traffic load) on the performance of the redirection technique.

7. ACKNOWLEDGEMENTS

We would like to thank Adela Isabel Fernández Anta for revising the syntax and grammar of this paper.

This work was partially supported by the National Project No. TEC2006-12211-C02-01.

8. REFERENCES

[1] B. Tang, H. Gupta, S.R. Das, Benefit-Based Data Caching in Ad Hoc Networks, IEEE Transactions on Mobile Computing, vol. 7, 3:289-304, 2008.

[2] Y. Du, S. Gupta, COOP – A cooperative caching service in MANETs, Proceedings of the of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS-ICNS 2005), pp. 58-63, Papeete (Tahiti), 2005.

[3] Y.H. Wang, J. Chen, C.F. Chao, C.C. Chuang, A Distributed Data Caching Framework for Mobile Ad Hoc Networks, Proceedings of the 2006 International conference on Wireless Communications and Mobile Computing, pp. 1357-1362, Vancouver (Canada), 2006.

[4] L. Yin, G. Cao, Supporting Cooperative Caching in Ad Hoc Networks, IEEE Transaction on Mobile Computing, vol. 5, 1:77- 89, 2006.

[5] H. Artail, H. Safa, K. Mershad, Z. Abou-Atme, N. Sulieman, COACS: A Cooperative and Adaptive Caching Systems for MANETs, IEEE Transactions on Mobile Computing, vol. 7, 8:961-977, 2008.

[6] Y. Ting, Y. Chang, A Novel Cooperative Caching Scheme for Wireless Ad Hoc Networks: GroupCaching, Proceedings of the International Conference on Networking, Architecture and Storage (NAS 2007), pp. 62-68, Guilin (China), 2007.

[7] G. Chiu, C. Young, Exploiting In-Zone Broadcast for Cache Sharing in Mobile Ad Hoc Networks, IEEE Transactions on Mobile Computing, vol. 8, 3:384-397, 2009.

[8] S. Hosseini-Khayat, Optimal Solution of Off-line and On-line Generalized Caching, Technical Report WUCS-96-20, 1996.

[9] NS-2 Home page: http://isi.edu/nsnam/ns/

[10] S. Kurkowski, T. Camp, M. Colagrosso, MANET Simulation Studies: The Incredibles, ACM's Mobile Computing and Communications Review, vol. 9, 4:50-61, 2005.

[11] J. Broch, D. Maltz D. Johnson, Y. Hu, J. Jetcheva, Multi-Hop wireless ad hoc network routing protocols, ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98), pp. 85-97, 1998.

[12] L.A. Adamic, B.A. Huberman, Zipf's law and the Internet. Glottometrics, vol. 3, pp. 143-150, 2002.

[13] S.G. Dykes, K.A. Robbins, C.L. Jeffery, Uncacheable Documents and Cold Starts in Web Proxy Cache Simulations, Technical Report CS-2001-01, Texas University (USA), 2000.

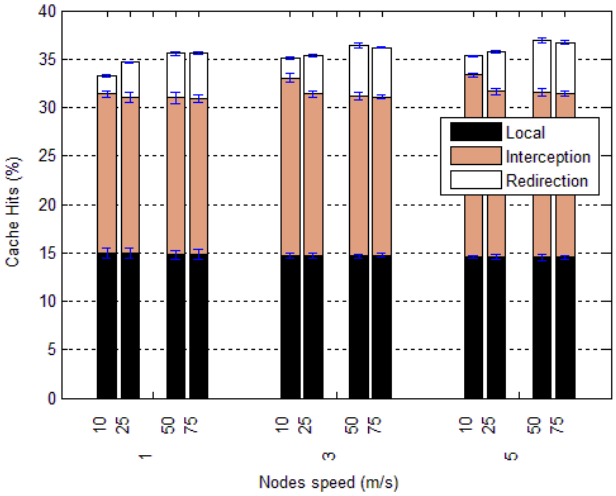

Figure 1. Hit Ratio as a function of the nodes speed.

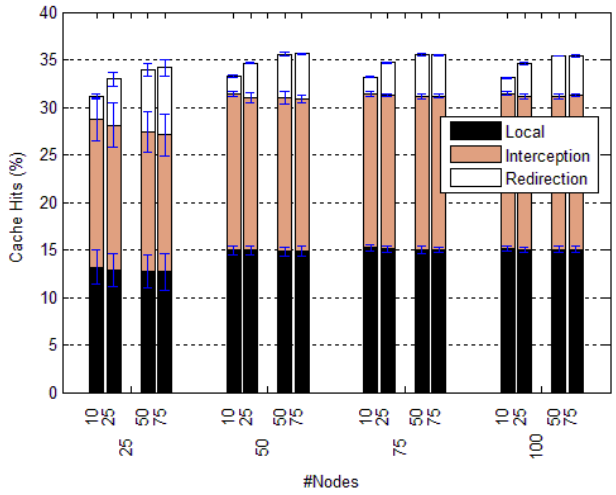

Figure 2. Hit Ratio as a function of the number of nodes.

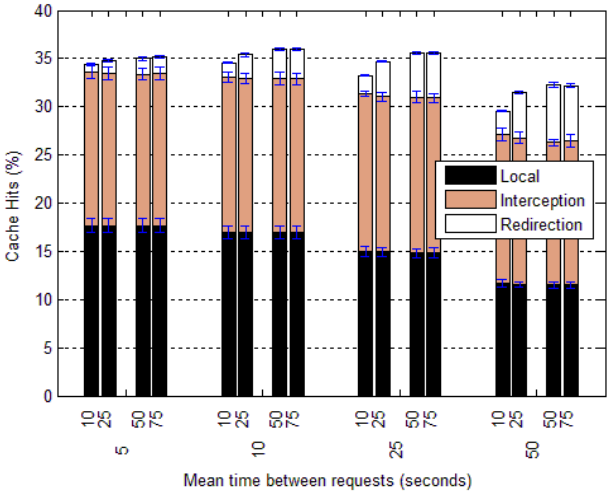

Figure 3. Hit Ratio as a function of the time between requests.

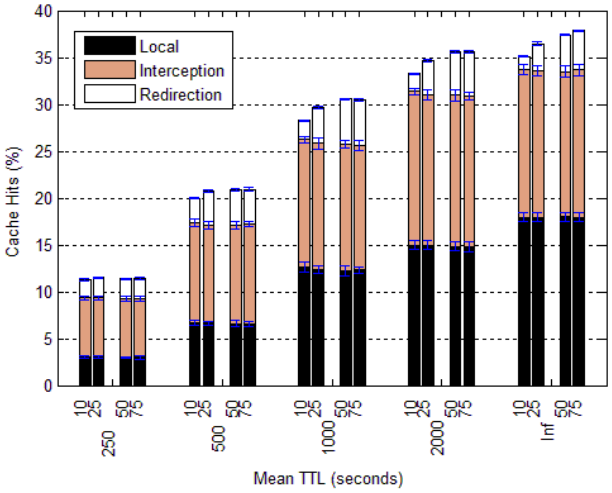

Figure 4. Hit Ratio as a function of the mean TTL of the documents.

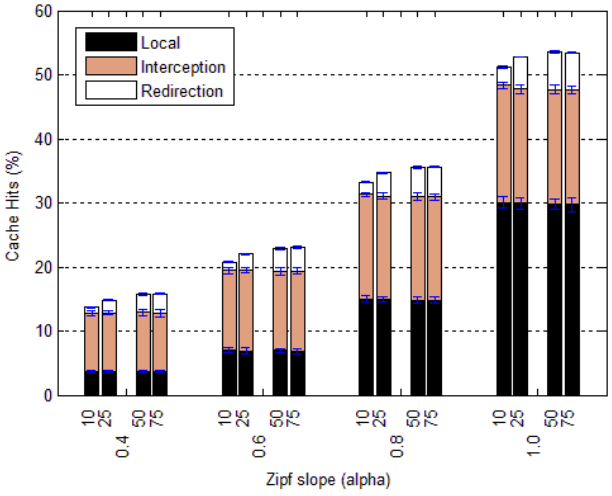

Figure 5. Hit Ratio as a function of the Zipf slope.

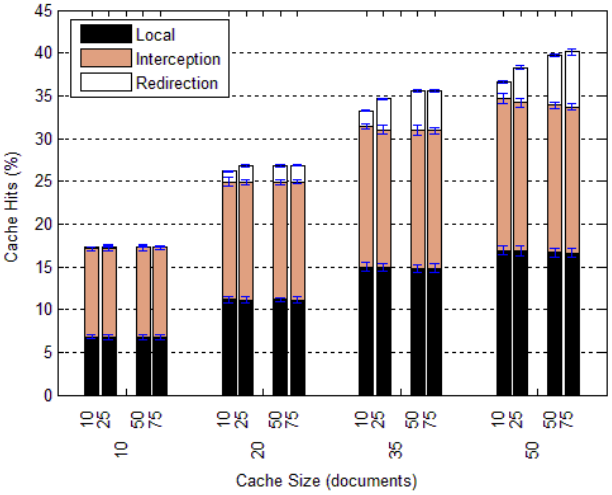

Figure 6. Hit Ratio as a function of the local cache size.