

AN APPLICATION LEVEL CACHING SCHEME IMPLEMENTATION FOR MANETS USING NS-2

F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera

Dpto. Tecnología Electrónica, University of Málaga

ETSI Telecomunicación, Campus de Teatinos, Universidad de Málaga, 29071 Málaga (Spain)

ABSTRACT

In this paper, we describe the implementation of a caching scheme for ad hoc networks. In this scheme the mobile nodes implement a local cache. This feature allows to intercept the forwarding requests and serve the documents requested directly using their local cache. Using the information obtained by the documents forwarded the nodes can redirect the requests to other nodes that are known to have the document requested. This caching scheme has been implemented using the network simulator NS-2. This paper describes the architecture, implementation details and customization parameters.

KEYWORDS

Caching, ad hoc networks, NS-2, replacement policies.

1. INTRODUCTION

The success of wireless technologies has prompted the use of mobile devices anywhere and anytime. Aiming at communicating distant wireless devices in an autonomous way, a MANET (Mobile Ad Hoc Network) can be formed. A MANET is composed of wireless devices that cooperate in the retransmission and routing of packets. In this way, wireless devices are able to communicate without any infrastructure.

Most of the research activities related to MANETs are accomplished by means of simulations. In this sense, the simulations tools already implement MANET technologies such as the routing protocols or the MAC layer (NS-2) (OMNET++). However, these two tools by themselves become insufficient to evaluate important application procedures, for instance, the caching schemes. When devices are equipped with caches, they can store documents that have been previously served by a host. Considering the mobile nature of the devices in a MANET, caching allows nodes to access some documents even when the host that keeps it is not reachable. This behaviour is expected to improve the performance of MANET applications.

In order to evaluate the benefits of using caching in MANETs, we have implemented a caching scheme for wireless networks (González-Cañete, 2009) using NS-2, that is one of the most popular network simulators for ad hoc networks (Kurkowski, 2005). The implemented caching scheme supports local caching and an interception model by which requests can be served by nodes in the MANET without accessing the server. On the other hand, the nodes also store information about the documents others nodes have in their local caches in order to redirect the requests they have to forward. In this way, the conventional cooperation of MANET in the routing procedures is also extended to the caching operations. Based on the implementation, we can compare the performance of the caching schemes in MANETs.

The rest of the paper is structured as follows. Section 2 details the architecture implemented as well as the messages and configuration parameters. Section 3 describes the relationship between the classes implemented. Finally, section 3 outlines the main conclusion of our work.

2. ARCHITECTURE

Figure 1 represents the architecture implemented to add a caching mechanism in ad hoc networks using the network simulator NS-2. The class *CacheNode* implements the application level protocol that generates

requests to the servers as well as the server behavior. As routing protocol, *CacheNode* utilizes the *AODV* class (Perkins, 2003) but any other routing protocol could be employed.

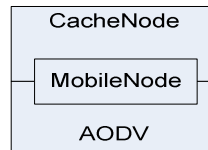


Figure 1. Protocol architecture

The class *MobileNode* implements all the caching and redirection capabilities interacting with *CacheNode* as well as with *AODV*. The procedure implemented by *CacheNode* is a request-response protocol. The mobile nodes send requests using a typical HTTP GET message. This GET message includes the numerical document identification. Before sending each request, *CacheNode* checks if the document is stored in the node's local cache. Once the request reaches the destination, the data server or another intermediate node in the path to the server responds with a RESP message, which contains the size and expiration time of the document.

Each time a node receives a GET or a RESP message it informs the class *MobileNode* in order to add an entry for the document. If the message is a GET, the source of the request and the distance in hops is annotated. If the message received is a RESP message the source of the response, the distance and the expiration time of the document are stored. In addition if there is an annotation for a document yet and a RESP message related to that document is received, the expiration time for the entry is updated. In that way the node knows which node requested the document and which node responded and how far they are.

When a node receives a GET message to forward (i.e. the node is not the destination of the GET) at *AODV* level, the *AODV* class consults the *MobileNode* if there is an updated (not expired) copy of the document requested in the local cache. In this case, *AODV* informs the *CacheNode* module that it can directly respond to the request so that the node does not forward the request.

On the other hand, if the document requested is not in the local cache of the forwarding node, *AODV* also asks the *MobileNode* for another node that is known to have the document and that is closer than the destination node. In this case, the node redirects the request to the closer node having the document. When a node receives a redirected request it responds with the document requested if there is a valid copy in its cache. However, if the document was evicted from the cache, the node sends an *ERROR_GET* message to the source of the request. In the path from the source of the request to the destination node, a list of the nodes that perform a redirection is stored in the GET message so that the *ERROR_GET* message follows the reversal path to the origin of the request passing through the nodes that redirected the request. Those nodes annotate that the document is no longer available for redirection to this node.

The node that receives an *ERROR_GET* message will have to request the document again. Two alternatives can be selected: to send another GET message or to send a *NO_REDIRECTION_GET*. This last message prevents the request from being redirected in the network.

Finally, the requests of the documents are governed by a timeout so that the document is requested again if no RESP or *ERROR_GET* is received before the timeout.

The classes *CacheNode* and *MobileNode* can be configured to change the behavior of the network and the caches. There are five parameters of *CacheNode* that must be configured:

- *pathFileSizes*, *pathFileTTL*, *pathRequestDirectory* and *pathWaitTimeDirectory* – Configuration files where the file sizes, TTLs, list of requests and time between requests are defined.
- *nodeType* – The nodes can be defined as CLIENT or SERVER. The main differences between clients and servers are: servers do not generate requests, the servers do not cache documents and each server has a portion of the documents existing in the network.

The *MobileNode* class can be configured setting the next parameters:

- *remoteCacheInterception* (ON/OFF) – Enables or disables the interception of messages.
- *cacheDestRedirectionOnlyWithRoute* (ON/OFF) – Enables or disables the redirection to a node even if there is not an active route to it.
- *cacheDestMode* – Configures how the cache of redirections selects a node to redirect the request. Eight values can be specified: *GET_ONLY* redirects the request only to a node that once requests the document; *RESP_ONLY* only redirects to the node that once served the response; *GET_FIRST* tries to

redirect the request to a node that solicited the document. If the document it is not available the message is redirected to the node that responded; `RESP_FIRST` is equivalent to the previous one but the responding node is tried first and the node that requested the document is tried next; `MIN_NHOPS` redirects the message to the closest node between the requesting and responding nodes; `MAX_EXP_TIME` redirects the request to the node with a version of the document with a higher expiration time; `NO_REDIRECT` disables the redirection capability.

- `maxRedirections` – The maximum number of times that a requests can be redirected.
- `minHopsProfit2Redirect` – When a node is going to redirect a request it calculates the difference of distance in hops between the destination node and the node where the message is going to be redirected. This difference is the benefit in hops. The `minHopsProfit2Redirect` parameter sets the minimum benefit that is required to redirect the request.
- `promiscuousMode (ON/OFF)` – Enables or disables the promiscuous mode in order to learn where the documents are located, not only using the requests and responses that pass through the node but also using the information of the adjacent nodes.
- `recvTimer` – It configures the timeout to wait for a response.
- `porcWarmUp` – It specifies the percentage of requests that must be used to warm up the local cache of each node before starting the simulation.
- `set-cache` – It defines the replacement policy used for the cache and the storage space.

In order to evaluate the performance of the networks simulated by this caching scheme, five trace and information files are generated during the simulations. Using this information the effect of using local cache, interception or redirection over the AODV or the CacheNode traffic as well as the variation of the delay or number of hops could be studied. Also the performance of the local cache, interception and redirection can be studied as a function of the network topology, the mean time between requests and the popularity distribution of the documents.

3. IMPLEMENTATION DETAILS

In order to implement the local cache for each node a proxy cache simulator implemented by the authors (González-Cañete, 2008) was utilized. This simulator was developed in C++ using the Borland C++ Builder IDE and the VLC library supplied by this development environment. The implementation of the replacement policies was adapted and the VLC classes utilized for the management of the storage were replaced by other classes implemented by the authors. Figure 2 shows the class diagram implemented.

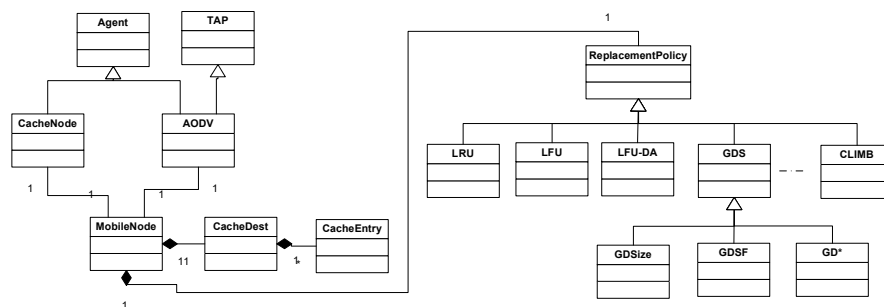


Figure 2. Class diagram implemented

The *CacheNode* class inherits from the NS-2 class *Agent* as well as the *AODV* class, which also inherits from the NS-2 *TAP* class to obtain the promiscuous mode. Both classes, *CacheNode* and *AODV* interact with *MobileNode* in order to interchange information about the local cache and the destinations cache. The *MobileNode* class contains an instance of the *ReplacementPolicy* class. The *ReplacementPolicy* class is an abstract class that implements most methods necessary to manage the data structures of the replacement policies. The *ReplacementPolicy* class is a super class for the classes that implement each replacement policy. There is a class for each replacement policy although the *GDS* class is an abstract class that implements a generalization of the GDS family (*GDSsize*, *GDSF* and *GD**).

As can be observed, the modular architecture developed allows adding new replacement policies by just deriving from the base class and overriding the necessary methods. Nowadays the replacement policies implemented are: FIFO (first Input First Output), LRU (Least Recently Used), LFU (Least Frequently Used), LFF (Largest File First), LDU-DA (LFU with Dynamic-Aging) (Arlitt, 1997), GD-SIZE (Greedy-Dual Size) (Cao, 1997), GDSF (Greedy-Dual Size with Frequency) (Cherkasova, 1998), GD* (Greedy-Dual*) (Jin, 2001), RANDOM and CLIMB.

The *MobileNode* class also contains an instance of the *CacheDest* class that manages a list of *CacheEntry* instances. Each *CacheEntry* instance contains the information about a particular document location. The address of the node that requested the document, the distance in hops and the expiration time of the document in the local cache of the requesting node are stored. The class also stores the same information for the responding node.

Each time a GET message is going to be forwarded at AODV level the *AODV* class asks *MobileNode* for an alternative destination stored in the destination cache. The *CacheDest* class will return the address of a new destination according to the configured *cacheDestMode* parameter. On the other hand if the message is not redirected *AODV* urges *CacheDest* to add or update the entry corresponding to the document. The *CacheNode* class also consults *MobileNode* in order to obtain a destination closer than the data server when a GET message is going to be sent. Finally, an ERROR_GET message received by the node causes the deletion of the document entry in the destination cache corresponding to the source address of the message.

4. CONCLUSIONS

In this paper we have presented the implementation of a caching scheme for ad hoc networks using the network simulator NS-2. This caching scheme includes local caching at each node, interception and redirection of the requests. We have defined the architecture and the functionality of each class implemented as well as the interaction between them. The implementation defines a set of parameters that can be modified in order to simulate and study a great variety of situations.

REFERENCES

- Arlitt, M. et al. 1997. *Internet Web Servers: Workload Characterization and Performance Implications*, IEEE/ACM Transactions on Networking.
- Cao, P. 1997. Cost-Aware WWW Proxy Caching Algorithms, *USENIX Symposium on Internet Technologies and Systems*
- Cherkasova, L. 1998. Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy. Technical Report HP Labs HPL-98-69.
- González-Cañete, F.J. et al. 2008. A Windows Based Web Cache Simulator Tool. Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTOOLS 2008), Marseille, France.
- González-Cañete, F.J. et al. 2009. Proposal and Evaluation of an Application Level Caching Scheme for Ad Hoc Networks. Proceedings of the 5th International Wireless Communications and Mobile Computing Conference (IWCMC 2009), Leipzig, Germany, pp. 952-957
- Jin, S. et al. 2001. *GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams*, International' Journal of Computer Communications, Vol. 24, No. 2, pp. 174-183
- Khayari, R.A. 2003. *Workload-Driven Design and Evaluation of Web-Based Systems*, Der Andere Verlag, Osnabrueck, Germany.
- Kurkowski, S. et al. 2005. MANET Simulation Studies: The Incredibles. *ACM's Mobile Computing and Communications Review*, vol. 9, no. 4, pp. 50-61, 2005.
- Luotonen, A. et al. 1994. World-Wide Web Proxies. *Computer Networks and ISDN Systems*, Vol. 27, No. 4, pp. 147-154.
- NS-2 Home page: <http://isi.edu/nsnam/ns/>
- OMNET++ Home page: <http://www.omnetpp.org/>
- Perkins, C. E. et al. 2003. Ad Hoc On Demand Distance Vector (AODV) Routing. IETF RFC 3561.