Speeding-up Cache Lookups in Wireless Ad-Hoc Routing using Bloom Filters

Evangelos Papapetrou, *Member, IEEE*, Evaggelia Pitoura, Kwstas Lillis

***Abstract*— On demand routing protocols that exploit local caches have received a lot of attention lately in wireless ad-hoc networking. In this paper, we specifically address cache management, an issue that has been a main source of criticism for the applicability of such protocols. In particular, we tackle the problem of accessing the cache content efficiently. To this end, we propose summarizing the cache content so that we achieve efficient lookups. This not only saves both the restrictive resources of the wireless devices such as computational power and energy but also improves the overall protocol performance. We use Bloom filters as summaries. Our experimental results using the ns simulator show that both resource savings and performance improvements are attained when such filters are integrated within the DSR protocol which is one the most widely used instance of an on demand protocol.**

***Index Terms*— Wireless, mobile, ad-hoc, routing, Bloom filters, cache memory**

I. Introduction

Advances in manufacturing of portable devices equipped with wireless interfaces boost the widespread use of different kinds of wireless networks. Ad-hoc connectivity has received wide acceptance over the last years for meeting the communication needs of mobile users using wireless devices. Several scenarios such as disaster relief, law enforcement, battlefield operations and networking in conferences are possible applications of self organizing networks. However, their nature imposes strict requirements on most aspects of their operation. Mobility, distributed operation and absence of any infrastructure shape an extremely challenging context for network management and especially for routing. Every mobile node must operate as a router in a distributed fashion and overcome frequent and sudden link failures that form a stochastically varying network.

On-demand routing protocols provide a reliable solution for coping with network variation and achieving efficient operation [1],[2]. Contrary to table driven protocols, routing information is discovered only when needed. Three basic mechanisms are implemented, namely route request, route reply and route maintenance. When a node in the network is in need of a path to a specific destination, it broadcasts a request packet. The destination then replies to the request packet with a unicast reply packet to form the communication path. Since routes are discovered only when needed, the overall overhead induced by the routing algorithm is minimized [1],[2],[3], in order to enhance network scalability. Several on-demand protocols utilize caching of discovered paths [1]-[4],[5],[6] for exploiting collected routing information in subsequent routing operations. Use of local caches in each mobile node has been proved to benefit network performance in terms of delivery ratio [3]. Caching is also an important attribute for future configurations that are foreseen to implement p2p systems over ad-hoc networks [7]. Moreover, the cross layer architecture proposed for such systems [8] merges application and network layer routing and renders caching as an essential network mechanism.

However, advantages of caching are not free of cost. Part of the critique on routing protocols using caching has been focused on implementation and feasibility issues related to highly populated node caches and the cost for accessing their contents [1],[2],[3]. Routing operation favors the increase of cache population for augmenting the delivery ratio and at the same time suppressing incurred routing overhead. A key issue is the cost of accessing local caches and more specifically the cost of searching in them. Many routing protocol mechanisms necessitate frequent cache lookups. For example, one of the most representative on-demand routing protocols utilizing local caches, DSR [4], makes frequent use of cache lookups in several of its mechanisms such as packet assembly, route discovery, route establishment and route as well as cache maintenance. Frequent lookup operations combined with the increased size of node caches are known to drain system resources such as computational power and energy reserves. The situation is worst in the context of a mobile node where both of those resources are restrained by technological limitations. Furthermore, since cache access is involved in protocol mechanisms, its impact on the the routing protocol performance itself is also important.

In this paper, motivated by the observation that many cache lookups end-up with a negative result, we propose using a mechanism for quickly determining cache membership. In this way, we can avert negative lookups and therefore ease the computational burden. The cornerstone of our mechanism are Bloom filters which provide summaries of the cache content for efficient testing cache membership. Although extensively used in the past [9], Bloom filters have never been used in wireless ad-hoc routing.

The rest of the paper is structured as follows. In Section II, we formulate the problem of cache lookups in the context of an on-demand routing protocol, present its implications on node resource management and protocol performance and propose the use of Bloom filters for alleviating the overhead of cache access. Then in Section III, we address some implementation issues. Results on feasibility and expected benefits of such an implementation are presented in Section IV. Finally, useful conclusions are drawn in Section V.

II. Problem Formulation and Proposed Solution

Caching of discovered routes is utilized by some on-demand protocols [4],[5],[6] to suppress the routing overhead and at the same time maximize delivery ratio, by further exploiting routing information collected in one route request-route reply cycle. This is achieved by allowing intermediate nodes having a cached path to destination, to reply to request packets. The most representative routing algorithm of this category is the DSR protocol [4]. DSR has been proved to outperform many protocols in terms of both delivery ratio and routing overhead confirming the benefits of caching. However, many concerns are raised related to the growth of the cache size and of the associated cost of cache accesses. Clearly, it is desirable that caches contain as much information as possible for the routing protocol to take advantage of it. The cache size is also augmented by the spread of stale routing information across the network [6],[3]. DSR manages the cache size by imposing a strict upper limit on the number of paths that can be cached. Newly discovered paths are added in the cache only in the expense of older ones. This approach decreases the cached routing information but provides a feasible solution for implementation.

In this paper, we address the cost of cache accesses. Nearly every mechanism of routing protocols accesses caches by means of a search or lookup for a specific node. Specifically in the case of DSR, cache lookups are performed in the route request phase, in route insertions and deletions from cache as well as in packet assembly since source routing is implemented. Furthermore, search frequency is increased since in a distributed environment each mobile node performs cache lookups to serve its peers. Search frequency combined with the cache size can be considered as an important factor encumbering node operation. Implications of cache lookups are twofold. On one hand, energy and computational power consumption are of extreme importance in the context of a mobile node where this kind of resources are in lack. On the other hand, computational burden may increase the overall processing time in a node and therefore affect the delay involved in a single cache lookup. This delay is critical for the protocol performance. For example, during packet assembly, searching for the desired route to the destination increases the packet delivery delay. Another important aspect of lookup delay relates to the overall time needed by the protocol to discover a route to destination. During a route request phase the request packet is propagated through a number of nodes. Each intermediate node is required to search for an available cached path before forwarding the request packet. Therefore the processing time of the packet in each intermediate node is increased, affecting the overall discovery time. Route discoveries are performed either for a new connection or for fixing a broken route to destination. In the first case, the so-called *path set-up delay* is increased. In fact path set-up delay is considered the major disadvantage of reactive protocols compared to proactive ones. In the case of route repair, delayed route discovery may result in the drop of packets waiting in buffers for a route to destination.

There are two approaches for alleviating the computational burden involved in cache accesses. The first one is to derive an efficient search algorithm for minimizing the cost involved in each single lookup. The basic disadvantage of this approach is that such an algorithm would depend on the cache structure and since there is no relevant specification, its applicability would be limited to a single routing protocol. The second approach is to minimize, if possible, the number of performed cache lookups. The on-demand behavior of routing protocols and the mobility of nodes results in a large number of performed lookups failing to locate a node. For example, when a route to destination does not exist, the search performed for finding the source route to be used by the data packet, is negative. Furthermore, many studies [6],[3] have proved that the number of route discoveries increases as a function of network mobility and size. This confirms the significant increase of negative result lookups. Therefore eliminating such lookups may be beneficial to the network performance. Based on the fact that every cache search is performed based on a node id, we propose the implementation of a mechanism to quickly and efficiently determine the membership of a node in the set of nodes comprising a cache. It is clear that negative membership decisions indicate that the related cache lookups would have a negative result therefore their actual execution can be avoided. Another advantage of the proposed solution is that it does not depend on the routing protocol and the cache structure. For implementing the described mechanism, we propose using Bloom filters.

A. Bloom filters and Cache access

Bloom filters are small compact data structures for fast membership decisions. Their applications range from rule-based systems to text analysis [9]. Consider a set $A = \{a_1, a_2, ..., a_n\}$ of n members. The Bloom filter is a vector v of N elements (bits), initially all set to 0. A number M_h of independent hash functions, $h_1, h_2, \ldots, h_{M_h}$, are used, each with range 1 to N. For each member $a \in A$, the bits at positions $h_1(a), h_2(a), \ldots, h_{M_h}(a)$ in v are set to 1. A particular bit may be set to 1 many times. Given a membership decision on b, the bits at positions $h_1(b), h_2(b), \ldots, h_{M_h}(b)$ are checked. If any of them is 0, then certainly $b \notin A$. Otherwise, we conjecture that b is in the set although there is a certain probability that we are wrong. This is called a *false positive*.

In the proposed implementation, a Bloom filter is maintained in each mobile node (Fig. 1). This filter provides a summary of the node's cache content. The identification number of each node in the cache is hashed, which may as well be its IP address or any other identification supported by the routing protocol. Each lookup request triggers a membership test through the Bloom filter. If the decision is negative, the search is not performed. Otherwise, the lookup is performed in the way specified by the routing protocol. In particular, in the proposed implementation, we do not use bit vectors, since many instances of a node may reside in a cache memory and the cache contents vary over time. Instead, we use counting Bloom filters [10] where each element of the filter is a small counter instead of a single bit. Each time an element of the filter must be set, the default action is to increase the counter

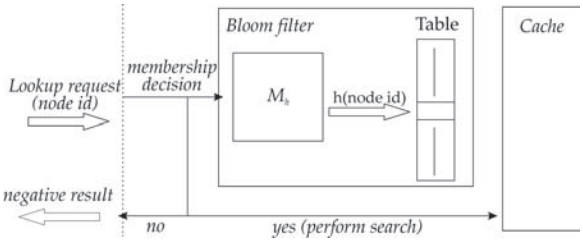

Fig. 1. Proposed Implementation

contained in the element. It can be proved that the analysis and the design of Bloom filters with more complicated elements is the same as that of simple Bloom filters [11]. Finally, note that to support updates of the cache contents, the Bloom filter must be accordingly maintained each time paths are added or removed from the cache.

Apparently, the proposed solution imposes two tradeoffs. The first is the cost of performing a membership test before a lookup that will result in a positive outcome and the second is the cost of maintaining the filter. These issues are addressed in the next section along with some feasibility considerations.

III. Tuning the Filters

Several issues arise regarding the proposed solution. The implementation of Bloom filters comes at a cost, therefore the following implementation aspects must be considered:

- Determination of the filter size
- Impact of false positive probability
- Cost of maintaining the filter
- Cost of performing the membership test

Regarding the filter size, suppose that the number of hash functions used is M_h, the number of elements in the filter is N and the number of possible set members is n. It is clear that in the worst case n is the number of nodes. To achieve optimal filter performance in terms of false positive probability, the following equation must be valid for the filter size [9]:

$$N = \frac{M_h n}{ln2} \tag{1}$$

The corresponding false positive rate is then:

$$P_{fp} = 2^{-M_h} \tag{2}$$

Given the network sizes that current routing protocols support and based on the previous equations, we can infer that the filter size (in terms of the number N of filter elements) can be kept very low and at the same time achieve a very small false positive rate. Assume that each of the N elements of the filter is of size s_{el}. Based on the cache size limitations imposed by the routing protocol and on the analysis presented in [10], we can derive that values of 4 or 8 bits are adequate for s_{el}, thus resulting in a small overall filter size. Furthermore, note that since false positives do not have a direct impact on the protocol operation, we can relax the requirement for small false positive probabilities. This allows each node to easily trade off between filter size and time complexity which depends on the number of used hash functions.

As mentioned previously, the implementation of Bloom filters involves two tradeoffs. The first one is related to the filter maintenance, that is, the time needed for inserting and deleting members in the filter. These operations require just the computation of M_h hash transforms and therefore require $O(M_h)$ time, where M_h is a small constant.The second tradeoff refers to the time needed to perform a membership test before each cache lookup. Assume that the cache size is N_{cache}, then the time for a cache lookup is $O(N_{cache})$. The membership test involves the computation of M_h hash transforms. If the test is positive, the time for the test ($O(M_h)$) is added to the time needed for the actual lookup ($O(N_{cache})$). On the other hand, if the test is negative, the actual cache lookup is skipped. In this case only two hash transforms are required on average for rejecting a non-member [9]. As mentioned before the size N_{cache} is desired to be large. On the contrary the number of used hash functions can be very small even for large networks. Therefore it is better to economize on lookups rather than filter operations. It is clear that the overall performance of the proposed solution is determined by the actual number of negative and positive searches performed and the frequency of update operations on the filter. In the next section, we will present some simulation results to evaluate in more detail the benefits of utilizing the use of Bloom filters in the context of an on-demand routing protocols.

IV. Simulation Results

A. Simulation Model and Evaluation framework

This section is devoted in analyzing and evaluating the benefits of the proposed solution. To this end we simulated the performance of DSR, the most popular on-demand protocol employing caching. The simulation results were obtained using the well-known Ns simulator [12]. Various simulation scenarios were studied by varying parameters related to geometry and mobility of network nodes. The simulation time for each scenario was 900 secs. The parameters adopted for simulating each aspect of the network are common in the literature [3],[6]. For the network geometry we used two different cases. The first one concerns 50, 75 or 100 nodes moving randomly in an area of 670x670 m^2. In the second one, the same number of nodes move randomly in an area of 1500x300 m^2. These two configurations provide the capability of comparing DSR behavior in networks with different sizes, same node density and different mean path length. It is clear that in the area of 1500x300m^2 the paths formed are in principal longer than in the case of the 670x670m^2 area. The nominal communication range of each host was 150 m. We simulated 25 connections, each one emerging from different hosts. Constant bit-rate sources were used for the generation of data packets with rate of 1 packet/sec for each connection. The channel capacity was set to 2 Mbits/s, while the packet size was 512 bytes. Host movement was based on Random Waypoint algorithm [3]. The maximum speed (u_{max}) was varied from 5 to 20 m/sec while the pause time of a host was set to 0 secs which corresponds to maximum mobility.

The suitability of the proposed solution depends on: a) the assessment of cache size and therefore the Bloom filter size,

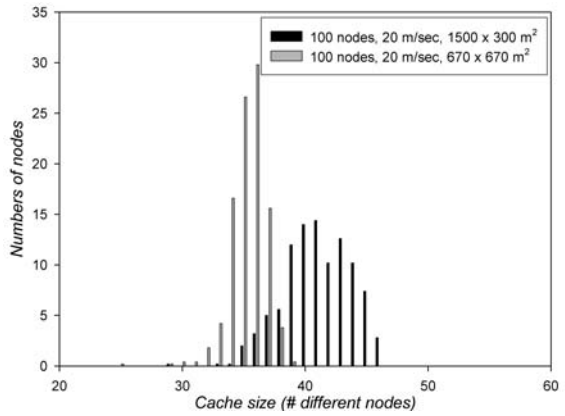

Fig. 2. Distribution of mobile nodes over the cache size

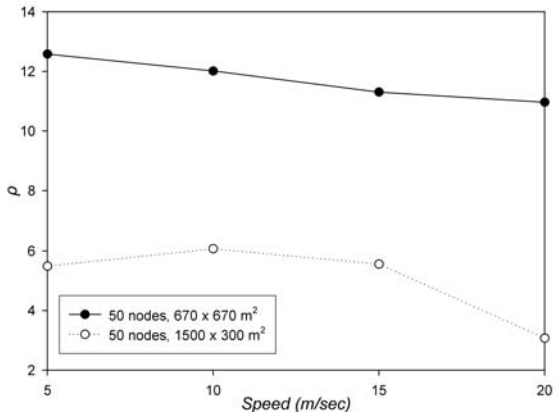

Fig. 3. Parameter ρ vs speed

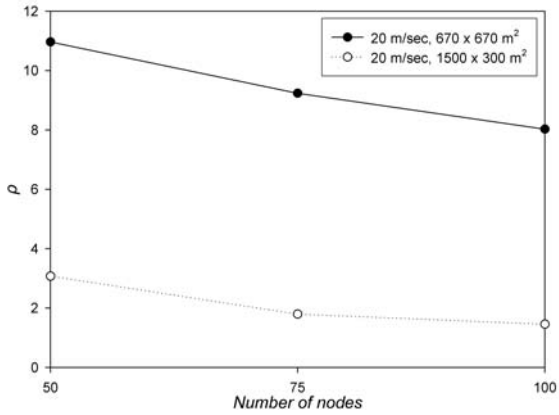

Fig. 4. Parameter ρ vs number of nodes

and b) the evaluation of the cost for maintaining the filter. Two different experiments were conducted. The first one assesses the performance in different mobility levels and the second for different number of nodes in the network. The cache size used through out these experiments was set to 30 paths. However, the size of the path is not limited.

B. Cache Content

The first objective of the study was to evaluate the mean cache size in a node. This size is necessary for determining the Bloom filter size based on Eq. 1 and 2. In particular, it must be clear that many instances of a node may reside in a cache. Therefore we evaluate size based on the number of different nodes and the total number of nodes in a cache. The first metric is the one to be used directly in Eq. 1 and 2 as n. The second one can be used in conjunction with the first to determine the size of the filter elements. Simulation results prove that the total number of nodes is clearly greater than the different nodes in cache (≈70-100 and ≈25-40 respectively). The ratio of the two metrics justify the assumption that only two or three bits are adequate for filter elements ($s_{el} \approx 2-3$ bits). Furthermore, based on Eq. 1 it is clear that even for the value of 40 nodes (maximum of different nodes recorded) we can construct filters of some bytes and at the same time achieve false positive rates of less than 10^{-3}. Both metrics present stability over the entire mobility range. This result is expected since mobility only affects the rate at which the cache is updated and not its actual content since the mean path length is the same regardless of mobility. We must note that the cache content is greater in the case of the second geometry scenario (1500x300 m^2) since the average path length is greater. Cache size (both different and total nodes) presents a slight increase over network size. This is owned to the increased network density. Each node has more neighbors and therefore is able of snooping more discovered paths that consist of more nodes. The stability of cache size over mobility and its slight increase with the number of network nodes present a useful attribute . It allows for the choice of filter parameters able to provide satisfactory performance in different network conditions. The latter conclusion is enhanced by Fig. 2 which shows that the majority of network nodes have similar cache size (in terms of different nodes in cache). This allows for a common strategy in the choice of the filter parameters.

C. Maintenance Cost and Implementation Benefits

From the analysis presented in Section III it is clear that the disadvantage of the proposed implementation is the need of performing a number (S_M) of insertions and deletions from the filter and a membership test for each of the successful lookups (S_P). All these operations require the same time T_C which is $O(M_h)$. The advantage is that for each of the negative lookups S_N we economize on time T_G which is $O(N_{cache})$. The proposed solution is suitable only if $k = S_N T_G/[(S_M + S_P)T_C] > 1$. Let us introduce the metric $1/\rho = S_N/(S_M + S_P)$. Then $T_G/T_C > \rho$ must be valid in order for the Bloom filter implementation to be meaningful. In Fig. 3 and 4 we present parameter ρ for different mobility rates and for different network sizes, respectively. In the worst case ρ reaches up to the value of 12. That means that time T_G for performing a lookup must be twelve times greater than time T_C for performing M_h hash transforms. This requirement can be easily fulfilled since T_G is $O(N_{cache})$ and T_C is $O(M_h)$. M_h is usually a small integer (see Section IV-B). On the contrary, N_{cache} which is the cache size at the time of the lookup, is usually a much greater number. This is confirmed in Fig. 5 where this size is presented for different network sizes. Over the entire range the cache size is over 75. Moreover, in

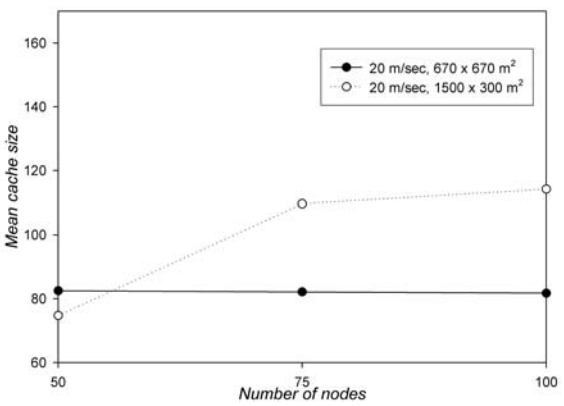

Fig. 5. Cache size at the time of a negative lookup vs number of nodes

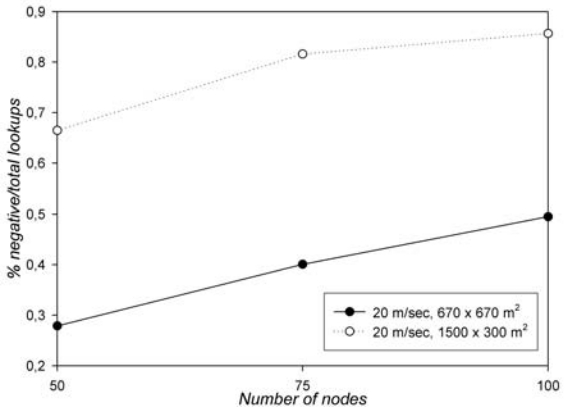

Fig. 6. Percentage of negative lookups vs number of nodes

the case of the 1500x300 m^2 scenario the average path length is greater than in the 670x670 m^2 scenario. Therefore paths are less resilient to mobility. As a result more route requests, which produce lookups in intermediate nodes, are performed, raising the mean cache size. Indeed in the 1500x300 m^2 scenario the increase on performed route requests is up to 380% contrary to the 670x670 m^2 scenario where the same value is 30%. Over the mobility range the cache size is stable around the value of 75. This stability is owned to the same reasons mentioned in Section IV-B.

Based on Fig. 3 and 4 we can conclude that the advantages of the proposed implementation tend to become more important for increased mobility and large networks, which is a very important characteristic. The requirement for ρ can be as low as $\approx$1 when network size and mobility increase. In other words Bloom filters are suitable even if $T_G = T_C$, that is even if the cost of a cache lookup is as cheap as performing M_h hash transforms. It must be noted that in the more demanding scenario of the 1500x300 m^2 area the proposed implementation is favored. These results combined with Fig. 5 further justify the suitability of Bloom filters. The behavior presented in Fig. 3 and 4 results from the degraded performance of routing protocols in challenging environments of large networks of high mobility. Routing information is not efficiently spread to node caches and negative lookups grow. This is confirmed in Fig. 6 where the percentage of lookups that resulted in a negative outcome is presented for different network sizes. The same behavior is present also over the entire mobility range.d to node caches and negative lookups grow. This is confirmed in Fig. 6 where the percentage of lookups that resulted in a negative outcome is presented for different network sizes. The same behavior is present also over the entire mobility range.

V. Conclusion

In this paper we proposed a new technique for accessing cache memories in on-demand routing protocols utilizing caching of routing information. The proposed technique is based on the implementation of Bloom filters. Although such filters have been extensively used in the past in various applications, they have never been proposed for augmenting the operation of a routing protocol in the context of a MANET. Specifically, in our implementation we take advantage of fast membership decisions that a Bloom filter can provide to eliminate cache lookups that induce computational and time overhead in every node implementing the routing protocol. We evaluated the impact of such an implementation through extensive simulation results, using the DSR protocol which is the most representative of those utilizing caching. The presented results sufficiently justify the feasibility of the proposed approach as well as its beneficial impact on the protocol performance.

References

[1] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, 1999.

[2] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks*, vol. 2, pp. 1–22, 2004.

[3] S.R.Das, C.E.Perkins, and E.M.Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *Proc. IEEE (INFOCOM' 00)*, 2000, pp. 3–12.

[4] D. B. Johnshon, D. A. Maltz, and Y.-C. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)," RFC, July 2004. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt

[5] S. Agarwal, A. Ahuja, J. P. Singh, and R. Shorey, "Route-lifetime assessment based routing (RABR) protocol for mobile ad-hoc networks," in *Proc. IEEE (ICC' 00)*, 2000, pp. 1697–1701.

[6] E. Papapetrou and F.-N. Pavlidou, "A novel approach to source routing for multi-hop ad hoc networks," *IEEE Communications Letters*, vol. 7, no. 10, pp. 472–474, 2003.

[7] G. Ding and B. Bhargava, "Peer-to-peer file-sharing over mobile ad hoc networks," in *Proc. IEEE Annual Conference on Pervasive Computing and Communications Workshops*, March 2004, pp. 104–109.

[8] K. Chen, S. H. Shah, and K. Nahrstedt, "Cross-layer design for data accessibility in mobile ad hoc networks," *Journal of Wireless Personal Communications, Special Issue on Multimedia Network Protocols and Enabling Radio Technologies*, vol. 21, pp. 49–75, 2002.

[9] J. Blustein and A. El-Maazawi, "Bloom filters. a tutorial, analysis, and survey," Technical Report CS-2002-10. [Online]. Available: http://www.cs.dal.ca/research/techreports/2002/CS-2002-10.pdf

[10] L. Fan, P. Cao, J. Almeida, , and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.

[11] B. H. Bloom, "Space/time trade-offs in hashing coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[12] K. Fall and K. Varadhan, "The ns manual," VINT Project, Univ. California, Berkeley, CA, 2001. [Online]. Available: http://www.isi.edu/nsnam/ns/ns-documentation.html