Mobile Information Systems 1 (2005) 275–307
IOS Press

Supporting user mobility through cache relocation

Kwong Yuen Lai*, Zahir Tari and Peter Bertok
RMIT University, School of Computer Science and Information Technology, GPO Box 3476V, Melbourne, Vic 3001 Australia
Tel.: +614 139 996 35; Fax: +613 966 216 17; E-mail: {kwonlai, zahirt, pbertok}@cs.rmit.edu.au

Abstract. Traffic and access delay can be reduced in a mobile network by caching data objects at network nodes near the clients. Traditional caching techniques, however, are unsuitable in this environment because they do not account for the changing location of the users. To deal with this problem, cache relocation techniques can be applied to dynamically relocate data objects so they remain close to the moving clients. Existing relocation techniques rely heavily on path prediction. Unfortunately, the inaccuracy of path prediction can lead to high relocation overhead and poor response time.

This paper presents an analytical study of the effects of client mobility on network cache performance. Two new cache relocation techniques are proposed to deal with the issue of poor path prediction and reduce the overhead of existing cache relocation schemes. The first technique, 2PR, compensates for poor path prediction by temporarily moving data objects to a common parent node prior to a handover. Objects are moved to the correct destination once the client's new location has been confirmed. The second technique, ROLP, reduces the traffic overhead associated with cache relocation by ensuring duplicate objects are not relocated and relocation of objects are performed only from the nearest node to the destination.

Test results show that 2PR reduces the query delay experienced by mobile clients by 60 to 83% after handovers, while ROLP reduces the overhead of cache relocation by between 33 to 65% compared to existing schemes.

Keywords: User mobility, cache relocation, transparent handovers

1. Introduction

Today, the most popular use of wireless networks is for personal communication applications such as mobile phone calls and short messaging service (SMS) [1]. As the prices of mobile computing devices drop and the cost of accessing wireless networks decreases, users will begin looking for a wider variety of services including wireless access to the World Wide Web and mobile file access, etc. To ensure high performance for these applications, caching techniques can be applied.

In a mobile network, caching takes place at two different levels. At the device level, clients cache data objects in the memory of their mobile devices. This type of caching is important because communication in mobile environments has high energy costs. By storing data locally, remote data access is reduced and overall communication cost for mobile clients is reduced. The second benefit of performing on-board caching is that data availability can be improved. Due to limited wireless coverage, mobile clients sometimes become disconnected from the network. Caching improves data availability during disconnection because even when a connection is unavailable, data can still be accessed through the

*Corresponding author.

on-board cache, thus allowing mobile clients to continue to operate during disconnection. Thirdly, data can be access more quickly from local memory than from remote hosts. As a result, by performing caching on mobile devices, user-perceived performance can be improved through reduced access delay.

Depite the many benefits of caching o nmobile devices, due to the smaller physical size of mobile devices, on-board caches are generally limited in space. In order to compensate for this, network caches can be installed at intermediate nodes between the remote servers and the clients.

This paper focuses on improving the performance of network caches located at the base stations in a mobile network (referred to as Home Location Caches or HLCs [2]). Unlike conventional network caches, where cache space is shared by multiple clients, each home location cache is dedicated to one mobile client. This is however just a logical separation. Multiple home location caches may exist on the same physical hardware.

Home location caches are useful for storing personalised data that are otherwise too costly to store on mobile devices (due to storage limitations). For example, a sales representative carrying a PDA may keep product catelogues and inventory information in their home location cache. A business person travelling between different location may store their email inbox in their home location cache so that even when their portable device is switched off, the home location cache can still act as a proxy to receive emails which can later on be downloaded to the device. Another example is a personal mobile file system where files in a user's home directory are stored at the home location cache to provide fast access.

Past research (e.g. [3–5]) has shown there are a number of benefits in having home location caches. Firstly, home location caches allow the most frequently accessed data objects to be stored near the clients, hence reduces the latency when accessing these objects. Secondly, they reduce the number of requests that needs to be transmitted to the remote server, thus lowers the traffic in the network. Furthermore, HLCs can act as proxies on behalf of disconnected clients, which simplifies the problem of dealing with network disconnections [2]. Lastly, storing objects in home location caches reduce the storage requirement imposed on mobile clients.

A challenging problem faced by home location caches is the impact of user mobility. Existing caching techniques, such as those discussed in [6–9], focused on caching in networks with static topologies. As a result, they cannot deal with the dynamic nature of wireless networks. When a mobile client moves from one location to another, its point of attachment to the network changes. The responsibility of supporting the client is handed over from one base station to another. However, under a static caching policy, data objects stored in the home location cache at the origin base station do not move with the client to the new location. This leads to longer access delay and increased communication cost for the client after the handover.

This paper investigates the use of cache relocation techniques to deal with the problem described. Cache relocation reduces the performance degradation experienced by users after handovers, by dynamically relocating their home location caches so cached objects remain close to the users as they move.

To minimise the impact of relocation, path prediction techniques (such as those discussed in [10,11]) can be used to predict the cell to which a client is most likely to move into. This allows cache relocation to begin prior to the user entering the new cell, thus ensuring the home location cache is ready to be used as soon as the client enters the cell. Unfortunately, an incorrect path prediction will result in objects being relocated to wrong locations. Although this could be corrected once the client has moved into the new cell, poor path prediction deteriorates the performance of home location caches and wastes bandwidth [4].

In this paper, two new techniques are proposed to reduce the network traffic associated with cache relocation and lowers the risk of incorrect path predictions when performing cache relocation. The

proposed techniques allow cache relocation to be applied efficiently to combat the effects of user mobility on network caching performance. The main contributions of this paper include:

- A new object relocation approach, called Two-Phase Relocation (2PR), which compensates for the low accuracy of existing path prediction techniques when making cache relocation decisions. When a user moves from one cell to another, objects in its home location cache are first copied to a common parent node of the origin and most likely destination cells. The probabilities of the client moving into any of its neighbouring cells are considered when choosing the common parent node to ensure only a small penalty will result if the client does not move into the most probable destination cell.
- The second contribution, called Return-path Object List Passing (ROLP), reduces the overhead of cache relocation by minimizing the number of data objects relocated and reducing the distance these objects need to travel to reach the destination. Object lists in ROLP are passed from the origin node to the destination node prior to a handover. These lists indicate to each intermediate nodes which objects need to be relocated. By recursively passing the object list along the path from the origin to the destination, ROLP coordinates intermediate nodes during the relocation to ensure each data object is relocated from the nearest location to the destination. This significantly reduce the amount of traffic transmitted directly from the origin to the destination.
- Lastly, a detailed analytical study of 2PR and ROLP is presented. The models developed facilitate comparison of the different strategies. Analytical and simulation results demonstrate that 2PR reduces the penalty of poor path predictions compared to existing schemes while ROLP significantly reduces relocation overheads.

The rest of this paper is organised as follows. Section 2 provides a survey of related work. The network model used in our study is described in Section 3. The proposed techniques are presented in Sections 4 and Section 5 respectively. This is followed by an analytical study in Section 6. Section 7 presents the simulation results and finally conclusion and future work are discussed in Section 8.

2. Related work

In this section, we describe some of the major work in the area of cache relocation and path prediction and put our work in context.

2.1. Proxy caching and cache relocation algorithms

The most recent work on cache relocation is a strategy proposed in [4] to accelerate web browsing in wireless networks. Prior to a handover, all data objects cached at the base station serving the moving client is copied to the most probable destination nodes according to path prediction. This ensures objects cached at the previous location will be available as soon as the client moves into the new cell, making the handover operation transparent to the client.

A path prediction algorithm based on a learning automaton is used to determine which cell a mobile client is likely to move into and how soon it will move into the new cell, so that relocation can begin prior to the handover. However, a number of studies [10,11] have shown that the accuracy of path prediction depends on the randomness of the mobile client's movement. As mobile clients operate in an unrestricted environment where they can move freely in any direction at any speed, it is not possible to be absolutely accurate in predicting their movement pattern.

To reduce the effect of incorrect predictions, the relocation scheme proposed in [4] copies all cached objects to the most likely destination node prior to a handover. Next, the top 70% most frequently accessed objects are also copied to the next two most likely destinations. Lastly the top 30% most frequently accessed objects are copied to the remaining adjacent cells. Although this approach is able to improve the average access delay experienced by clients, the traffic overhead introduced is very high.a This is because objects are relocated to multiple different locations, resulting in some objects being relocated to locations that the client never end up going to. Apart from the waste of bandwidth, this also leads to cache space at other network nodes being wasted.

Another caching strategy that uses proxy caches to improve the performance of GPRS networks was proposed in [5]. The problem of client mobility and the issue of handovers on cache performance were identified. Caches are placed at the SGSN (Serving GPRS Support Node) in GPRS networks which enables mobile clients to move between cells serviced by the same SGSN without the need of cache relocation. Unfortunately, when a mobile client moves into cell serviced by a different SGSN, it will continue to contact the cache at the previous location. This limits the benefit of network caches due to the delay added when contacting the remote cache.

The Mobile Cache Protocol (MCP) is a dynamic object relocation protocol for wide area networks [12]. Each client periodically calculates the best location to cache an object so that efficient access to the object is achieved. The computation is based on network bandwidth and each object's access frequency. MCP differs from the approaches proposed in this paper in a number of ways. Firstly, relocation decisions are made periodically in the MCP approach, while in the proposed schemes, relocation decisions are made prior to handovers so that the impact of handovers on clients can be reduced. Secondly, in the proposed approaches, relocation decisions are made at the cache level, whereas in MCP, relocation decisions are made for each object individually, resulting in higher computational overheads.

Another strategy was proposed in [3] to determine the best set of objects to store in caches located at base stations throughout a mobile network. The strategy attempts to maximise data consistency while satisfying bandwidth requirements. The choice of which objects to access remotely and which objects to cache locally is mapped to the knapsack problem, which is then solved using a dynamic programming algorithm. The authors showed the benefits of having caches at base stations. However, they did not consider the problem of mobility and how to deal with the issue of handovers. Furthermore, only caches located at base stations were utilised in this strategy. The fact that data can also be cached at other nodes in the network were not considered.

2.2. Path prediction algorithms

The study of path prediction algorithms is an important area of work that influence the design of cache relocation strategies. By predicting the likely destination of a moving client, path prediction algorithms can help cache relocation strategies make more accurate decisions when dynamically relocating objects. It can also help manage resources in the mobile network (e.g. bandwidth reservation prior to handovers).

An interesting mobile motion prediction algorithm was proposed in [13], in which regularity detection algorithms are used to identify known movement patterns. When a mobile client follows a movement pattern which had been observed previously, their further path can be predicted by looking up the pattern database. Three types of movement pattern matching methods were devised to compare the client's actual path with the recorded paths. These include state matching, velocity matching and frequency matching. Furthermore, the proposed technique is based on a learning automaton. Each time after a path prediction, the learning automaton is given positive or negative feedback depending on the correctness of the prediction. This allow the automaton to *learn* and improve the accuracy of future predictions.

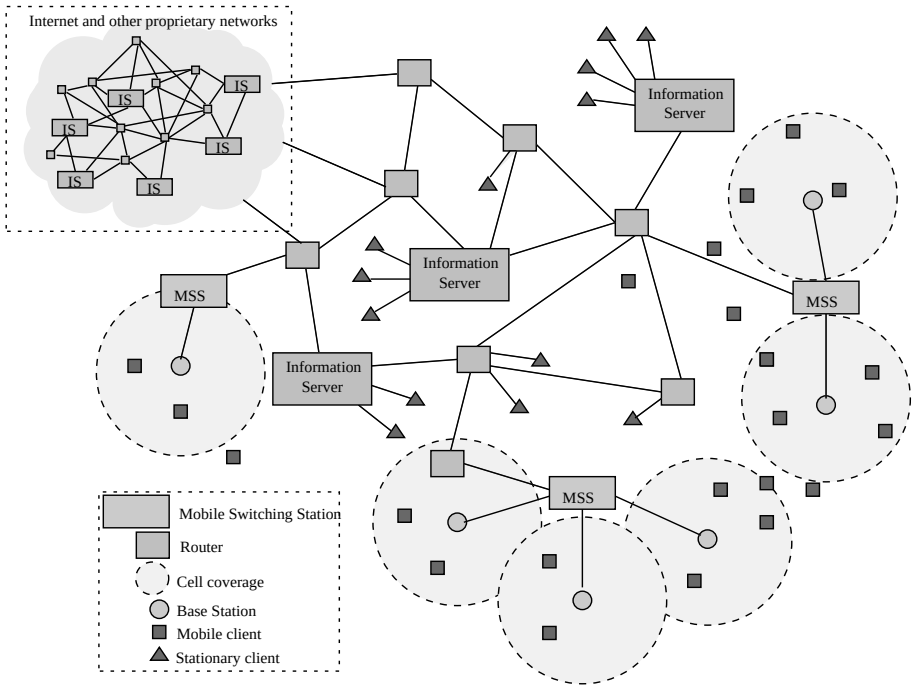


Fig. 1. A mobile computing network.

Another representative path prediction strategy is the Hierarchical Location-Prediction (HLP) scheme [10]. HLP uses a combination of client-initiated location update and location prediction to improve the resource management of wireless ATM networks. HLP deals with regular client movement patterns using an approximate pattern matching algorithm. Regular client movements are recorded in client profiles. As a client moves, its path is compared against the recorded movements to identify likely future movement paths. Furthermore, while many path prediction algorithms are unable to deal with randomness in real-world client movement, HLP uses a low level prediction algorithm to cater for this. Test results show that the HLP strategy can achieve around 75% accuracy in its next-cell predictions [10].

3. Background

Before the 2PR and ROLP schemes are presented, it is necessary to describe some background information about the network model used in this study. We consider a mobile wireless network that consists multiple cells. The size of a cell is determined by the transmission range of the base station located within the cell. Users carrying mobile computing devices connect to the network via the base stations. When a mobile client moves from one cell to another, a handover takes place. During a handover, the responsibility of taking care of the moving client is passed from one base station to another. Mobile switching centers ensure handovers happen smoothly and clients can move freely between cells without interruption to their connection. Also connected to the wired network are information servers that store data objects of interest to the mobile clients. Figure 1 illustrates the network described. The network can also be represented as a hierarchy of nodes (as illustrated in Fig. 2 where the root node is a remote information server and the leaf nodes are base stations.

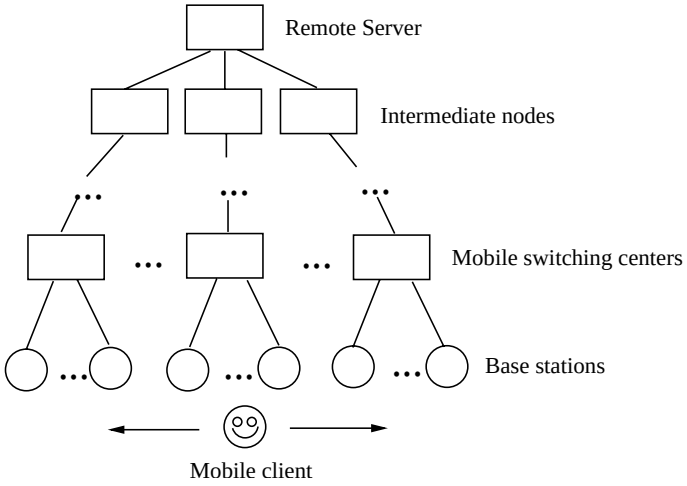


Fig. 2. A hierarchical mobile network.

When a mobile client wishes to access an object, it will first look in its own cache to search for a copy of the object. If the object is found, the query is processed locally without the need of any communication. However, if the object is not available in the cache, the client will send the request to its home location cache to ask for a copy of the object. When the home location cache receives the client's request, it checks whether the requested object is available. If the object is found, it will be sent back to the client. On the other hand, if the object is not available at the home location cache, the request is forwarded to the home location cache's parent node. This process continues up the hierarchy until the request can be answered by one of the nodes, or when the request reaches the remote server. The benefits of having a hierarchy of caches is that frequently accessed objects can be stored in caches close to the client's physical location, while objects less frequently accessed can be stored further away in caches at intermediate nodes. This is particularly important in a wireless network because mobile devices are usually limited in storage capacity. By storing objects at intermediate nodes between clients and the remote information server, the storage requirement placed on the clients is reduced. Furthermore, it allows network traffic to be distributed near the edge of the network, which significantly reduces the load at the core of the network.

4. The two-phase relocation (2PR) scheme

The 2PR strategy takes advantage of the hierarchical structure of mobile networks to reduce the impact of client mobility on home location cache performance and compensates for poor path prediction accuracy when making cache replacement decisions.

In 2PR, the relocation process is divided into two main phases. The first phase occurs before a handover takes place. The top k% most frequently accessed objects in the client's home location cache are relocated to the most likely destination nodes according to path prediction. Only the most frequently accessed objects are relocated because if the client does not enter the most likely destination cell, the impact of the incorrect path prediction can be minimised. At the same time, all the objects (including the top k%) in the client's home location cache are also copied to a common parent node until the client's new location is confirmed. The common parent node is chosen based on the client's current location, and the likelihood of it entering into any of its neighbouring cells such that on average, the total relocation

Table 1
Probability of entering neighbouring cells for the example of Fig. 3

Neighbour	Probability of entering
Cell a	0.25
Cell c	0.30
Cell d	0.20
Cell e	0.15
Cell f	0.08
Cell g	0.02

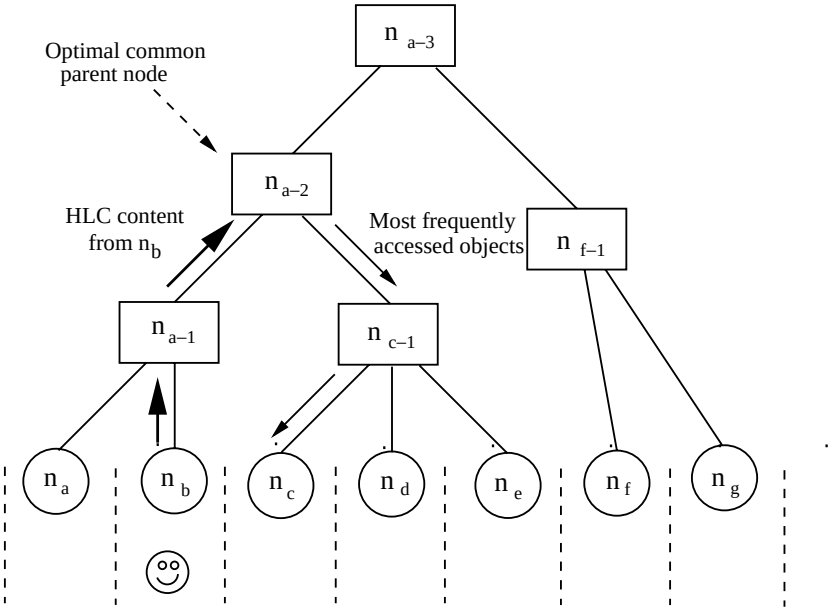


Fig. 3. Two-phase relocation.

cost is reduced. The common parent node selection algorithm is presented later on in Section 4.1. This is followed by a discussion on the effect of choosing different values for k in Section 4.2.

The second phase of the 2PR process takes place after the client has entered into the new cell. The action taken in this phase depends on whether the client has entered the most probable cell as predicted by the path prediction algorithm. If it did enter the most probable cell, the rest of the cached objects (that is, those not in the most frequently accessed set) are moved from the common parent node to the client's new location. If path prediction is incorrect, the entire content of the home location cache is relocated from the common parent node or from the client's previous location (depending on which one is nearer to the client's new location) to the new location.

Figure 3 illustrates how the two-phase relocation method works. Assume a client m_x is currently located in cell b and the path prediction algorithm predicts the client is likely to move into one of the neighbour cells with the probabilities shown in Table 1. Since cell c is the most likely destination, the most frequently accessed objects from the home location cache at n_b will be copied to n_c. Furthermore, assuming the common parent node is n_{a-2}, the objects at n_b are also copied from n_b to n_{a-2}. At this stage, the first phase of the two-phase relocation method is completed.

The second phase starts after the client is handed over to the new cell. If the client moves into cell

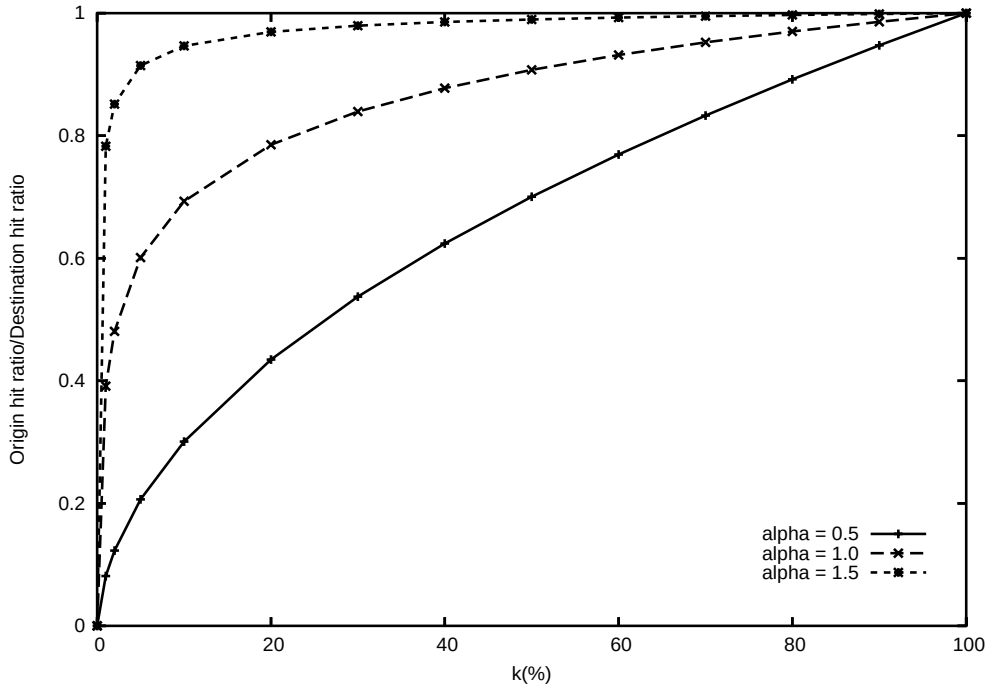


Fig. 4. Ratio of origin/destination hit rate vs. k.

1	2	3	4	5					D-1	D
1	1	0	1	0	0	1	1	1	0 ... 1	0

Fig. 5. An example bit sequence.

c as predicted, since the most frequently accessed objects have already been relocated to n_c, the client will be able to utilise these objects as soon as it enters cell c. The rest of the objects will be relocated from n_{a-2} to n_c, completing the two-phase relocation process. On the other hand, if the client entered another cell instead (say cell e), the cached object will be relocated from the common parent node, n_{a-2}

The 2PR scheme does not rely only on the most likely destination as predicted by path prediction, but also takes into consideration the probability of the client moving into any of the neighbour cells. This is a major improvement compared to the existing approach of relocating the entire cache based on the most likely destination prediction. In the case of an incorrect path prediction, 2PR ensures less bandwidth will be wasted, because only a small number of cached objects has been moved to the wrong destination. In the case of a correct path prediction, since the most frequently accessed objects have already been relocated to the predicted destination in the first phase of 2PR, these objects will be available to the client as soon as they enter the new cell resulting in lower access delay.

4.1. Selection of a common parent node

The common parent node should be chosen such that the maximum delay experienced by a client is bounded by the client's latency requirement. At the same time, the average traffic due to cache relocation should be minimised.

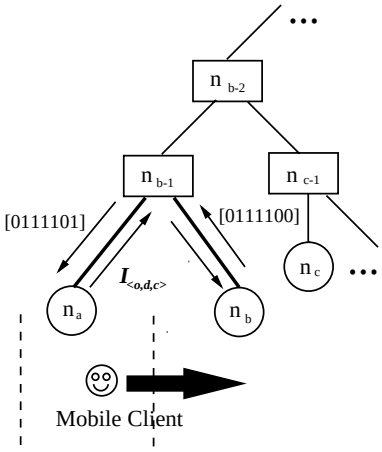
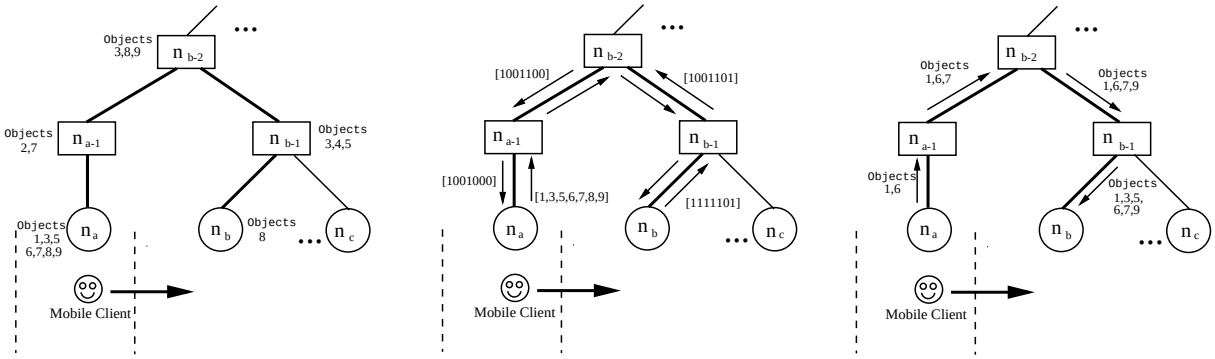


Fig. 6. Return-path object list passing.



a) Objects cached at each node

b) Return path object list passing

c) Cache relocation

Fig. 7. Rolp example.

To formally define the problem of how the common parent node should be chosen, a number of concepts must first be introduced. First, let denote by H the minimum spanning tree representing the given network. H can be represented as follows:

$$H = (N, L) \tag{1}$$

where N is the set of nodes in the network, and L is the set of links connecting these nodes together.

A link between two nodes n_i and n_j is denoted $l_{i,j}$, where $l_{i,j} \in L$ and $n_i, n_j \in N$. In a network, nodes are usually connected in a graph topology. However, H could be obtained for a graphed network using Kruskal's [14] or Prim's algorithm [15,16], by representing each link in the graph as an edge and taking the cost of transmitting an object over each link as the weight for the edges.

Assume a client m_x is currently located at cell o, and the base station serving o is n_o. The client's home location cache is denoted as HLC_x, which holds D objects with a mean size of $\mathrm{size}_{\mathrm{object}}$. M denotes the set of base stations serving the cells geographically adjacent to cell o, such that if the mobile client exits cell o, its new location will be within a cell serviced by n_d, where $n_d \in M$.

Let a path between n_o and n_d be defined as:

$$p_{o,d} = \{l_{o,o+1},\ l_{o+1,o+2},\ l_{o+2,o+3}, \ldots, l_{d-1,d}\} \tag{2}$$

such that $l_{i,i+1} \in L$ for all i.

The probability that the client will move from n_o to n_d is denoted as $Pr(n_o, n_d)$. If $n_d \notin M$, that is, if n_d is not an adjacent cell to the client's current cell, then $Pr(n_o, n_d) = 0$.

The latency of a link $l_{i,j}$ is denoted as $t_{i,j}$ and the cost of moving one object over this link is $cost_{i,j}$. The latency of a path $p_{o,d}$ can be calculated by summing the latency of all links on the path. That is:

$$t_{o,d} = \sum_{\forall l_{i,j} \in p_{o,d}} t_{i,j} \tag{3}$$

Assume each mobile client has an acceptable delay requirement of $t_{\max}$. The goal is to find a node n_q for the two-phase relocation algorithm, so that the maximum access latency the client will experience is less than $t_{\max}$ and the average cost of relocating the content of HLC_x (the client's home location cache) is minimised.

Given H, n_o and $Pr(n_o, n_d)$ for all $n_d \in M$, the problem of selecting the common parent node can be solved by finding a node n_q such that:

$$t_{q,d} \leqslant t_{\max} \qquad \forall n_d \in M \tag{4}$$

and

$$\min \left(\sum_{\forall l_{i,j} \in p_{o,q}} \text{cost}_{i,j} + \sum_{\forall n_d \in M} \left(Pr(n_o, n_d) \times \sum_{\forall l_{i,j} \in p_{q,d}} \text{cost}_{i,j} \right) \right) \quad \forall n_q \in p_{o,d},\ \forall n_d \in M \tag{5}$$

The condition specified in Eq. (4) ensures that a common parent node is selected such that the maximum latency for accessing an object from this node is within the client-specified requirement $t_{\max}$. For all nodes that satisfy this requirement, the one that incurs the lowest relocation cost should be used as the common parent node as shown in Eq. (5). The first term inside the minimise condition of Eq. (5) represents the cost of moving the objects from the origin node n_o to the selected parent node n_q. The second terms represents the expected cost of moving the objects from the selected parent node to the destination n_d. Since multiple destinations are possible, the expected cost is calculated by summing the product of the probability of the client entering any of the possible destinations and the cost of relocating the objects from the common parent node to each of these destinations.

To reduce the amount of computation required in selecting the common parent node, the 2PR scheme only considers nodes located on the paths between n_o and $n_d \in M$ as candidates. The cost of using each of the candidates as the common parent node is calculated with Eq. (5) and the one with the lowest cost is selected for the relocation.

4.2. Most frequently accessed set

In the 2PR relocation process, only the top k% most frequently accessed objects are relocated to the most likely destination node prior to the client being handed over. This section discusses the implications of choosing different values for k%.

The effect of choosing different values for k is dependent on the access distribution of the mobile client. In this thesis, it is assumed that clients access data objects following a Zipf distribution [17] and

objects are ranked based on their access probabilities. The probability that a client would access the i-th most popular object (denoted as d_i) is equal to:

$$Pr(d_i) = \frac{\frac{1}{i^\alpha}}{\sum_{x=1}^{D} \frac{1}{x^\alpha}} \qquad i \in \{1..D\} \tag{6}$$

where α is the Zipf parameter which specifies the skewness of the distribution, and D is the number of objects in the client's home location cache.

Based on Eq. (6), if k% of objects are relocated to the destination node, the probability that an object d_a cached at the origin node will also be found at the destination can be calculated as follows:

$$Pr(d_a \in \text{ destination}) = \sum_{i=1}^{D \times k\%} \frac{\frac{1}{i^\alpha}}{\sum_{x=1}^{D} \frac{1}{x^\alpha}} \tag{7}$$

Equation (7) is plotted in Fig. 4 for different values of k for a home location cache with 1000 objects. The graph shows that when $\alpha = 1.0$, the destination node can achieve 80% of hit ratio of the origin node by relocating just 25% of the home location cache.

The value for k should be chosen such that the hit ratio meets the client's minimal hit ratio requirement at the destination. Obviously, this depends on the access distribution. In the case where clients' access probability follows a Zipf distribution (such as shown in Eq. (6)), the value of k will depend on the skewness of the distribution. For example, to achieve an origin vs. destination hit ratio of 80% for $\alpha = 1$, k needs to be around 25%. To achieve the same hit ratio when $\alpha = 0.5$ will require a k value of almost 70% as shown in Fig. 4. This is because with a low α value, the client's access pattern will exhibit less locality, thus a higher k value is needed to achieve the same hit ratio requirement. On the other hand, if the value of α is large (for example, $\alpha = 1.5$), a smaller value for k will be sufficient.

5. The ROLP (Return-path Object List Passing) scheme

While the 2PR scheme reduces the impact of poor path prediction, it can still be expensive (in terms of the network traffic generated) to relocate objects in a mobile network. This is particularly the case for clients who frequently cross cell boundaries. In order to apply cache relocation to deal client mobility while keeping the network traffic overhead associated with cache relocation low, another relocation technique called Return-path Object List Passing (ROLP) is proposed in this section.

ROLP takes advantage of the fact that in a mobile network, copies of objects can exist in the caches of different nodes. When a client relocates from one cell to another, it is not always necessary to move all the objects from the origin node to the destination node. Some objects may already exist at locations close to the destination. In order to take advantage of this in the ROLP scheme, intermediate nodes between the origin and the destination node pass coordinating messages between each other to ensure objects already close to the destination are not propagated all the way from the origin.

When a client m_x moves from its current cell (n_o) to the destination cell (n_d), its home location cache also needs to be relocated from n_o to n_d. The relocation is coordinated by informing all nodes along the path from n_o to n_d which objects will be relocated. An object list containing the IDs of the objects cached by the home location cache at n_o is passed to all intermediate nodes between n_o and n_d to achieve this.

The structure of the object list, $I_{<o,d,c>}$, is in the following form:

$$I_{<o,d,c>} = \{\mathrm{ID}_i\} \qquad \forall i \in \mathrm{HLC}_x \tag{8}$$

where o is the ID of the origin node, d is the ID of the destination node, c is the ID of the moving client, HLC_x is the home location cache of the client and ID_i is the ID of an object in HLC_x.

Each intermediate node will temporarily keep a copy of the object list. When the list reaches the destination node, the cache there is compared against the list to identify overlaps. Objects already cached at the destination need not be brought all the way from the origin node. The destination node has to keep these objects in its cache to ensure they are available to the mobile client as soon as it enters the new cell. After the destination node has finished checking its cache against the list, it constructs a reply message containing a bit-sequence structure to inform the previous node in the path (denoted as n_{d-1}) about objects that still need to be sent as well as objects that already exist at the destination. Each bit in the bit-sequence corresponds to one object included in the object list. An object that already exists at the destination node is represented by a binary '1' in the bit sequence, otherwise it is represented by a binary '0'. An example bit sequence is shown in Fig. 5 where D is the number of objects that need to be relocated from the origin home location cache.

When a node n_{d-1} receives the reply message from the destination node, it maps the bit-sequence into the object list it had previously stored to find out which objects are already available at the destination and which objects still need to be sent. If the ID of an object in its cache is found in this list, n_{d-1} will forward a copy of the object to the destination node. For each of the objects forwarded, it also sets the corresponding bit in the bit-sequence to '1'. The bit-sequence is then sent to the next node along the return path. This process continues until the bit sequence finally reaches the origin node. Figure 6 shows graphically the steps described.

The object list provides a mechanism for nodes to work together in relocating the home location cache. Objects will only be copied from the closest node that has cached it. This results in reduced relocation overhead in terms of network traffic generated.

To demonstrate the benefit of using the ROLP scheme, ROLP is now compared to the method of relocating everything from the origin to destination without considering objects cached at intermediate nodes. Assume the cost of copying an object over a single hop (that is, from one node to an adjacent node) is C. In this example, C is assumed to be a constant, however a cost function can be used instead for more complex models. The cost of relocating the home location cache from a node n_a to another node n_b without considering what is cached at intermediate nodes between n_a and n_b is:

$$C_{\mathrm{nochecking}} = C \times D(n_a, n_b) \times H(n_a, n_b) \tag{9}$$

where $D(n_a, n_b)$ is the number of objects that need to be relocated from n_a to n_b and $H(n_a, n_b)$ is the number of hops between n_a and n_b.

On the other hand, the cost of relocating using ROLP in this example is equal to:

$$C_{\mathrm{ROLP}} = C \times \sum_{j=a}^{b} (D(n_j, n_b) \times H(n_j, n_b)) \tag{10}$$

A detailed example showing the operation of the ROLP scheme is depicted in Fig. 7.

Assume at time t, the client m_x is about to move from cell a to cell b. Figure 7(a) shows the objects that are currently cached at each node between n_a and n_b. Using the proposed ROLP algorithm, n_a first constructs the object list $(1, 3, 5, 6, 7, 8, 9)$. This list is passed from n_a to n_b along the highlighted path.

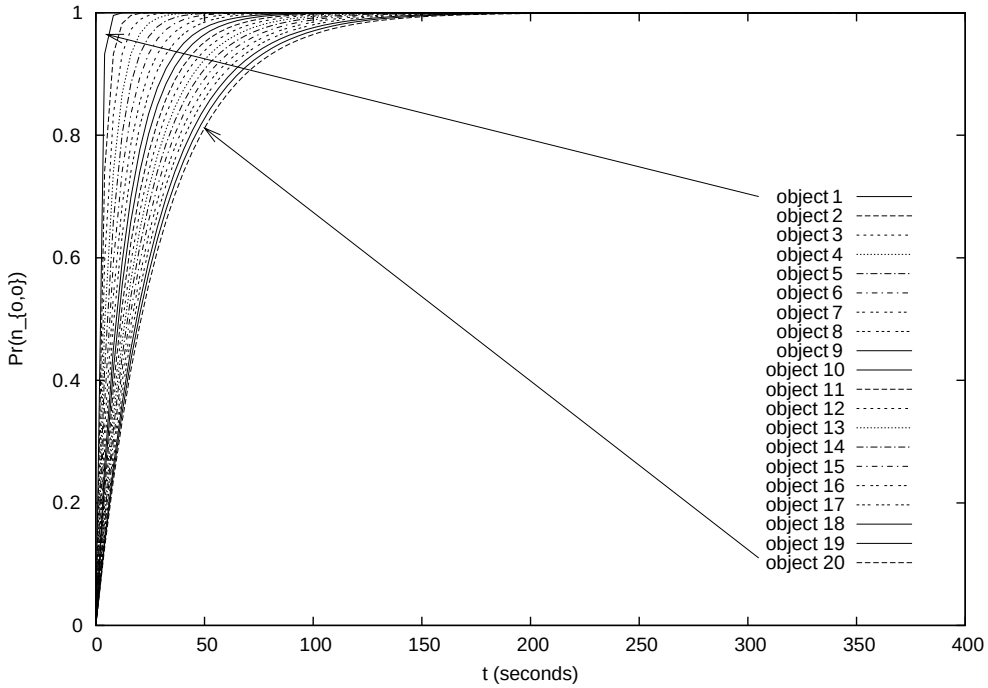


Fig. 8. Time to return to steady state for $\lambda = 5$/s.

When the list reaches n_b, it is found that object 8 is already in the cache at the destination. As a result, node b sends back the bit sequence "0000010" to n_{b-1}. Since n_{b-1} has already cached objects 3 and 5, it will send these objects to the destination node (n_b) and send the bit sequence "0110010" to the next node in the return path. This process continues until the bit sequence is returned to n_a. At this point, the only objects that needs to be relocated from n_a to n_b are objects 1 and 6. All other objects have already been provided by an intermediate node located closer to n_b.

The cost of relocation in the example in Fig. 7 can be calculated using Eqs (9) and (10):

$$\begin{aligned} C_{\text{nochecking}} &= D(n_a, n_b) \times H(n_a, n_b) \times C \\ &= 7 \times 4 \times C = 28C \\ C_{\text{ROLP}} &= C \times \sum_{j=a}^{b} (D(n_j, n_b) \times H(n_j, n_b)) \\ &= C \times (2 \times 4 + 1 \times 3 + 1 \times 2 + 2 \times 1 + 1 \times 0) = 15C \end{aligned}$$

From the above calculation, it is clear that a substantial saving can be made using ROLP. The amount of savings achieved depends on the number of objects already existing at intermediate nodes and the cost of transmitting these objects over the relocation path. Note that the cost of transmitting the object list and the return bit sequences have not been included in the calculation for simplicity. It is expected that the size of these messages is much smaller than the average size of an object, their cost is negligible. In our simulation, the cost of transmitting these messages are included to ensure the results obtained are accurate.

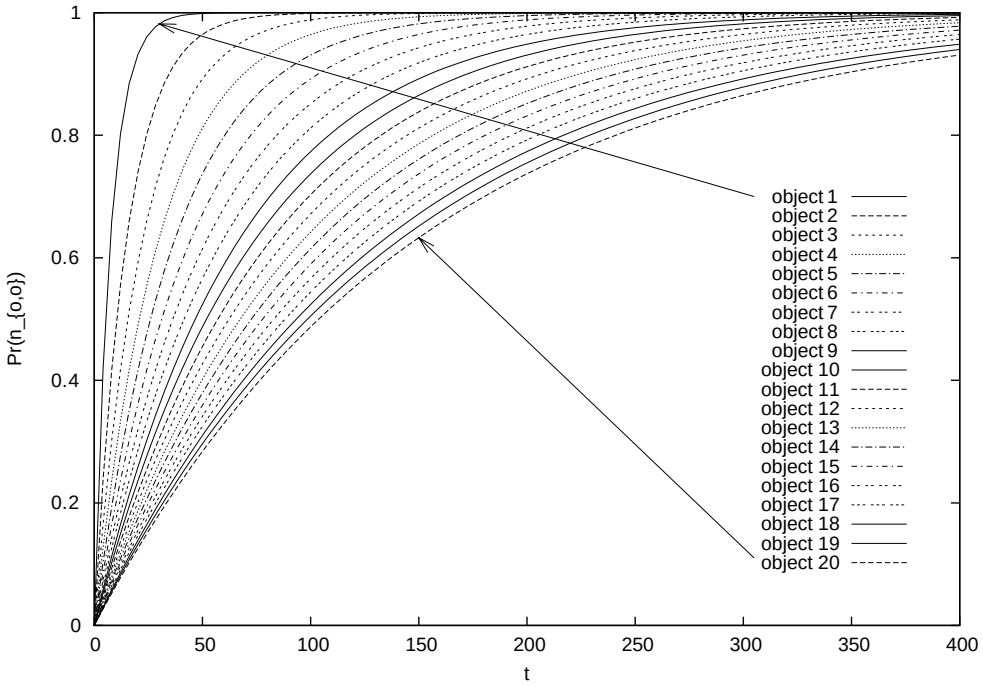


Fig. 9. Time to return to steady state for $\lambda = 1$/s.

The proposed relocation method does not consider the impact of updates to data objects. To deal with the issue of update propagation and data consistency, ROLP can be extended to provide a simple form of version control. For such an extension, the version number of each object that needs to be relocated should be sent together with the object IDs in the object list. When an intermediate node checks the object list against its own cache, only objects with a newer version compared to the origin node is relocated from the intermediate node. If the origin node has a newer copy of an object compared to all the intermediate nodes, the object will be relocated from the origin. This technique gives a best-effort attempt in providing the client with the latest copy of cached objects while providing a saving on the relocation overhead. Intermediate nodes can also benefit from this by updating their cache with more up-to-date data objects as they are being relocated.

6. Analytical study

In this section, analytical models of a number of cache relocation schemes are presented. These models will enable a fair comparison between various relocation schemes to be made without the need of simulation. They also provide a general framework for analysing the performance of these schemes under different workloads and could be used to verify the accuracy of simulation results.

Apart from the related work described in Section 2, two basic relocation schemes are also introduced for comparison purposes, namely the *Delayed* strategy and the *Greedy* strategy. These two strategies are included in the analysis because their performance provide the upper and lower bounds of the query delay and traffic overhead achievable using cache relocation.

The delayed-based cache relocation strategy takes place after a mobile client has moved into a new cell. When the client's new location has been confirmed, data objects stored in the origin home location

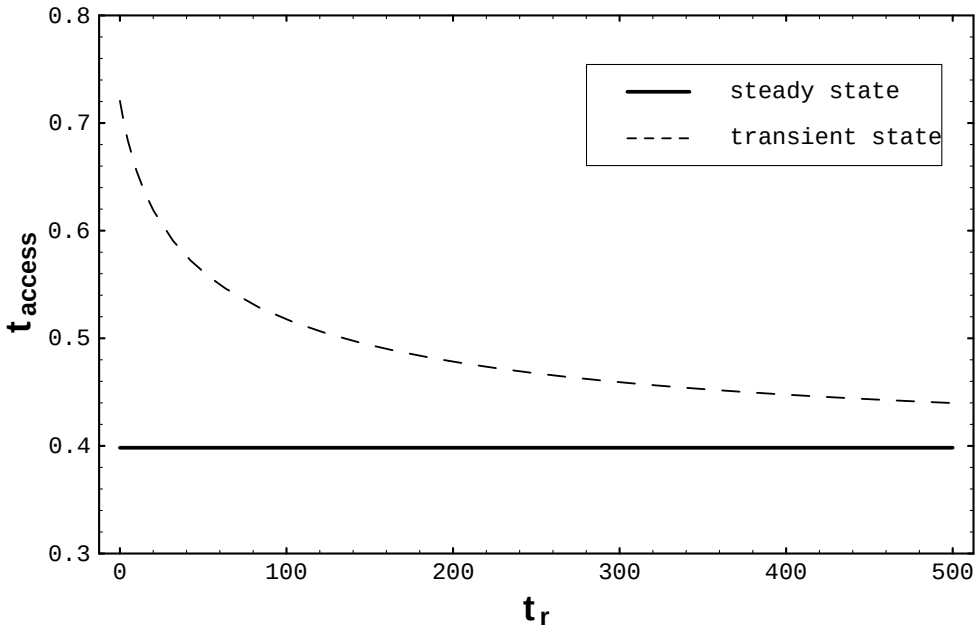


Fig. 10. Mean access delay for different t_r.

cache are relocated to the new cell. Due to the fact that relocation does not take place until after the handover, the client will experience increased access delay after entering the new cell. However, as the client's new location is known when relocation takes place, the network traffic overhead associated with relocation in the delayed technique also represents the best case achievable by a cache relocation algorithm.

In the greedy-based strategy, relocation takes place prior to a handover. Since it is not known in advance which cell a mobile client will move into, objects cached in the home location cache at the origin node are copied to all neighbouring cells. This ensures as soon as the mobile client enters the new cell, the cached objects will be readily available for access. Obviously in this strategy, the majority of objects relocated will end up unused. Only those objects relocated to the correct client destination will be used. As a result, although the *Greedy* strategy can achieve transparent cache relocation, it also represents the worst case scenarios in terms of the amount of traffic generated.

6.1. Preliminaries

Consider the underlying network as a hierarchy $H = \{N, L\}$. A single node is denoted as n_i where $n_i \in N$ and a link connecting two nodes is $l_{i,j}$ where $l_{i,j} \in L, n_i, n_j \in N$. It is assumed that from any node $n_i \in N$, there is a path $p_{i,r}$ to the root node n_r which represents the remote server that stores the master copy of all data objects.

The transmission delay associated with a link $l_{i,j}$ is denoted $t_{i,j}$. A processing delay $t(n_i)$ is also imposed on any traffic passing through a node n_i.

Associated with each client is a home location cache located at the base station (denoted n_o) servicing the client's current cell. Each client is given dedicated space on their home location cache, enough to store up to $\text{size}_{\text{cache}}$ objects. Caches at intermediate nodes further up the hierarchy (that is, nodes between n_o and n_r) are assumed to be proxy caches shared by all clients, and can store up to c objects.

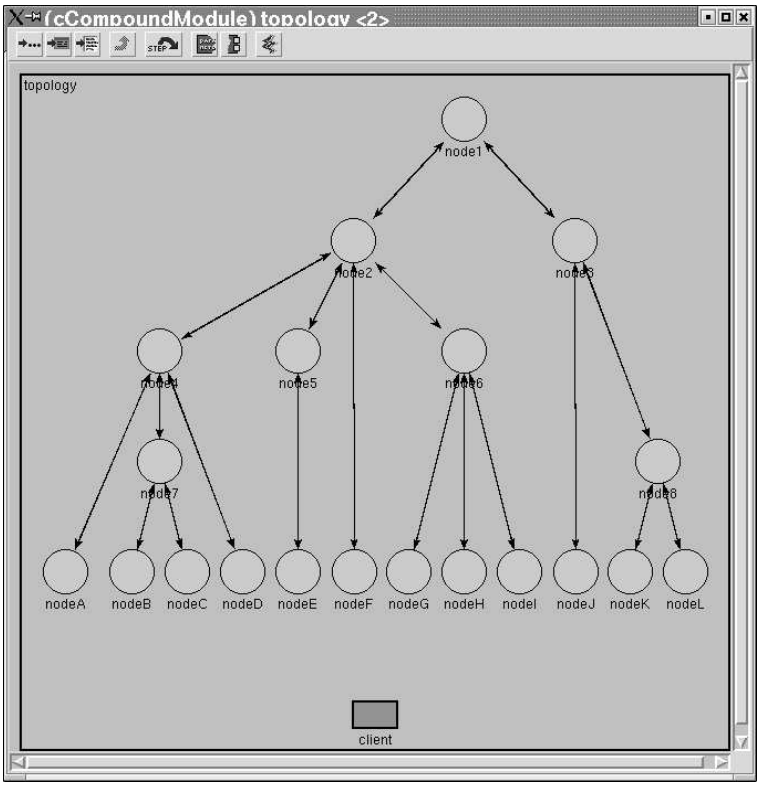


Fig. 11. Simulated Network in OMNET++.

Parameter	**Value**
Simulation duration	10000s
Number of objects	1000
Client cache size	5% of server items
Mean object size	8192 bits
Mean object ID size	32 bits
Mean time between handovers	500s
Mean time between queries	5s
κ for the 2PR method	20%
Zipf parameter(α)	1.0

Fig. 12. Simulation parameters.

The Least Frequently Used (LFU) scheme is used in all the caches when making cache replacement decisions. However, other policies such as LRU, LRU-K [18] and other cost-based cache replacement policies [19] could also be used.

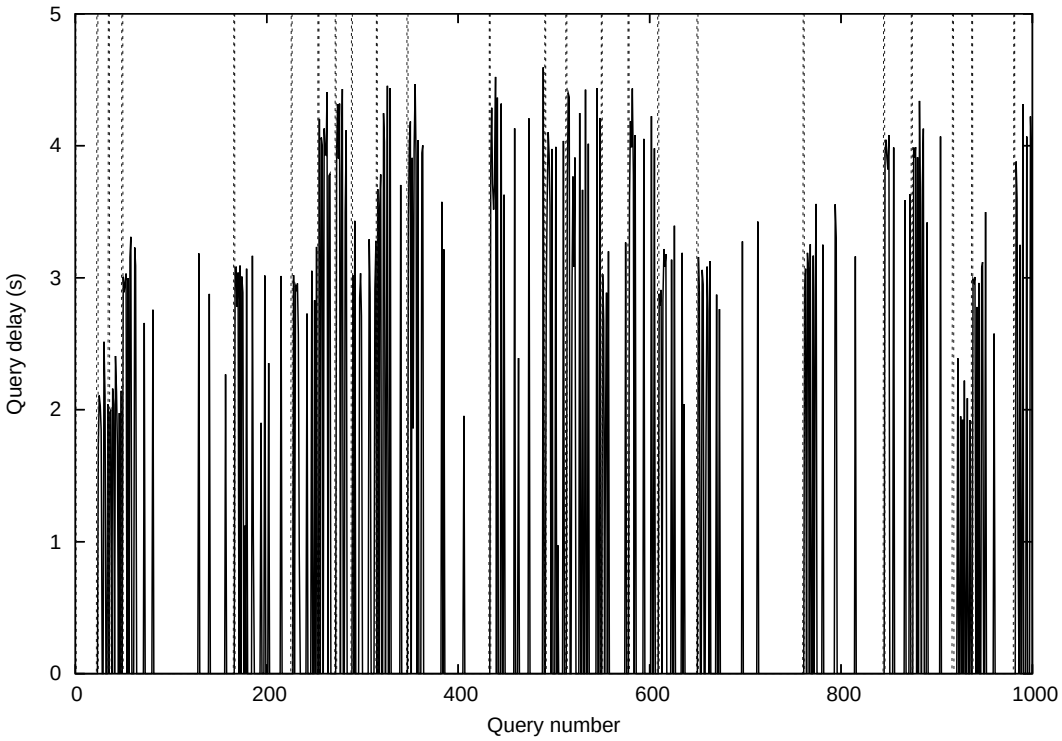


Fig. 13. Query delay for 1,000 queries without cache relocation (No Relocation).

6.2. Access delay

With the basics defined, the following analysis investigates the impact of movement on the access delay experienced by clients when fetching objects from their home location caches. Access delay is an important performance measure because it represents the response time as perceived by the clients and is a direct indication of the quality of service experienced by the clients.

Assume a mobile client is currently connected to n_o. When a request from the client arrives at n_o, the cache at this node is checked to see if it can satisfy the request. If the requested object is found, it is sent back to the client, otherwise, the request is forwarded to the next node in the path $p_{o,r}$. If none of the intermediate nodes on $p_{o,r}$ is able to answer the request, it will eventually arrive at n_r where it gets answered by the remote server. As the analysis is only concern with the query delayed experienced by clients when accessing their home location caches, only requests that are not satisfied by the clients' on-board caches are considered.

Let $Pr(n_{o,i})$ denote the probability of a request originating from n_o being answered by a node n_i. The access delay associated with this request is $t(n_{o,i})$.

Given $Pr(n_{o,i})$ and $t(n_{o,i})$, the mean access delay experience by a mobile client can be calculated using the following equation:

$$t_{\text{access}} = \sum_{i=o}^{r} (Pr(n_{o,i}) \times t(n_{o,i})) \tag{11}$$

That is, the average delay experienced by a client is equal to the the probability of the client accessing

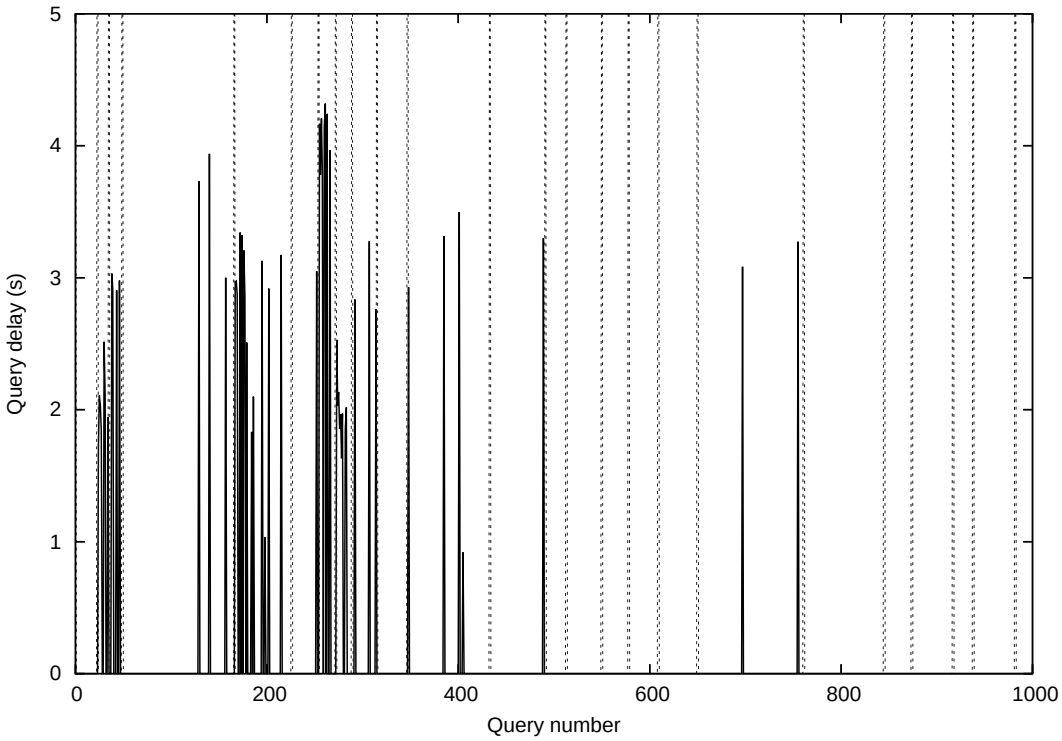


Fig. 14. Query delay for 1,000 queries using cache relocation (ROLP).

data objects from each node in the network multiplied by the delay of fetching the objects from each of those nodes.

Next, expressions for $Pr(n_{o,i})$ and $t(n_{o,i})$ need to be derived. $t(n_{o,i})$ is simply the sum of the transmission delay of all the links from n_o to n_i plus the processing delay of each node along this path.

$$t(n_{o,i}) = 2 \times \left(\sum_{j=o}^{i-1} (t_{j,j+1} + t(n_{j+1})) \right) \tag{12}$$

where $t_{j,j+1}$ is the transmission delay of the link $l_{j,j+1}$. $t(n_{j+1})$ is the processing delay at n_{j+1}. The summation in Eq. (12) is multiplied by 2 because the request first travels from n_o to n_i, and then the object is sent back from n_i to n_o.

6.2.1. Calculating $Pr(n_{o,i})$

Next, the computation of $Pr(n_{o,i})$ (the probability that a request from n_o is answered by n_i) is considered. Note that the value of $Pr(n_{o,i})$ depends on how long it has been since the client entered the current cell. Obviously if a client has been residing in the same cell for a long period of time, objects most frequently accessed by the client are likely to be found in the home location cache at the current node. On the other hand, for a client who has only recently moved into a new cell, the requests may have to travel further up the hierarchy before they get answered.

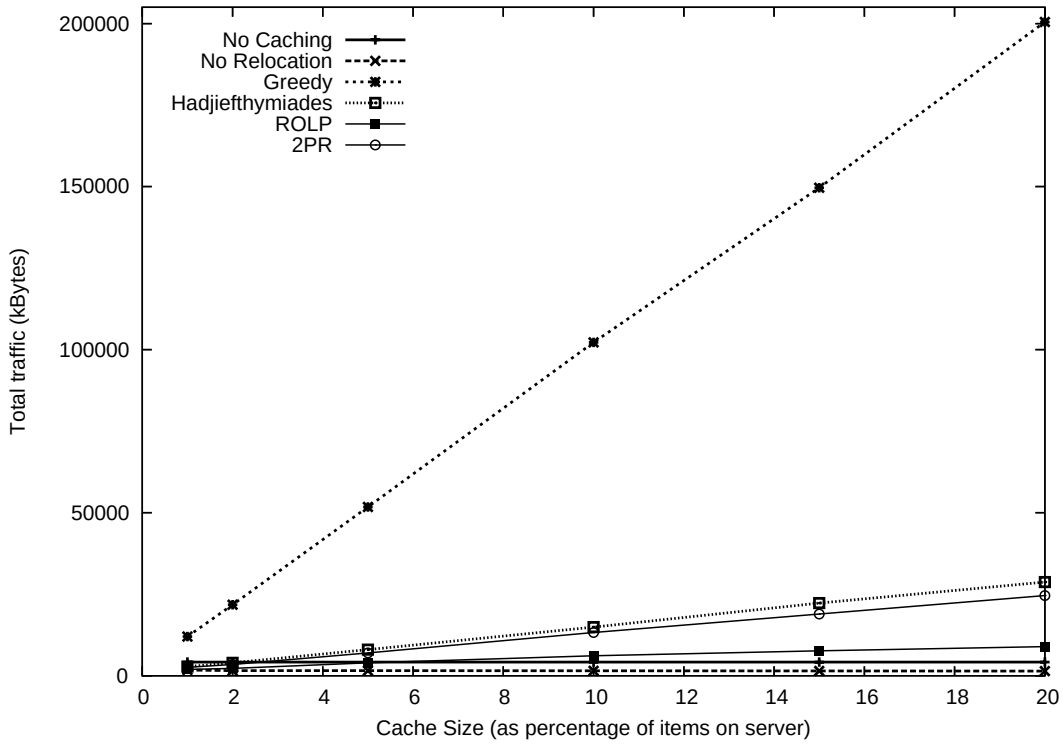


Fig. 15. Total traffic vs. cache size.

6.2.2. Steady state analysis

In this section, an equation is derived to compute the value of $Pr(n_{o,i})$ when the client has been residing in the same cell for a long period. A client is considered to be in *steady state* in this case. More precisely, a client is considered to be in *steady state* when its cache is filled with objects it most frequently accesses.

Assume that a set of D unique objects are stored at the remote server, and these objects are ranked based on their access frequencies, such that $d_x \in D$ is the x-th most frequently accessed object. Access locality is modelled by distributing client requests among objects following a Zipf distribution. Therefore, the probability of an object d_x being queried can be expressed as $Pr(d_x)$ as defined in Eq. (6).

Assuming LFU is used for cache replacement, the home location cache at n_o is expected to contain the top $\text{size}_{\text{cache}}$ most frequently accessed objects in steady state. Based on this the probability that a request can be answered by the home location cache at n_o can be expressed as:

$$Pr(n_{o,o}) = \sum_{x=1}^{\text{size}_{\text{cache}}} Pr(d_x) \tag{13}$$

where $\text{size}_{\text{cache}}$ is the size of the home location cache and $Pr(d_x)$ is the probability the x-th object is requested.

Requests that cannot be answered by n_o are forwarded to higher nodes. The cache at these nodes are shared by all clients in the hierarchy below, and they have the same size, denoted $\text{size}_{\text{intermediate}}$. A request will only reach a node n_i if the object requested cannot be found in the home location cache at

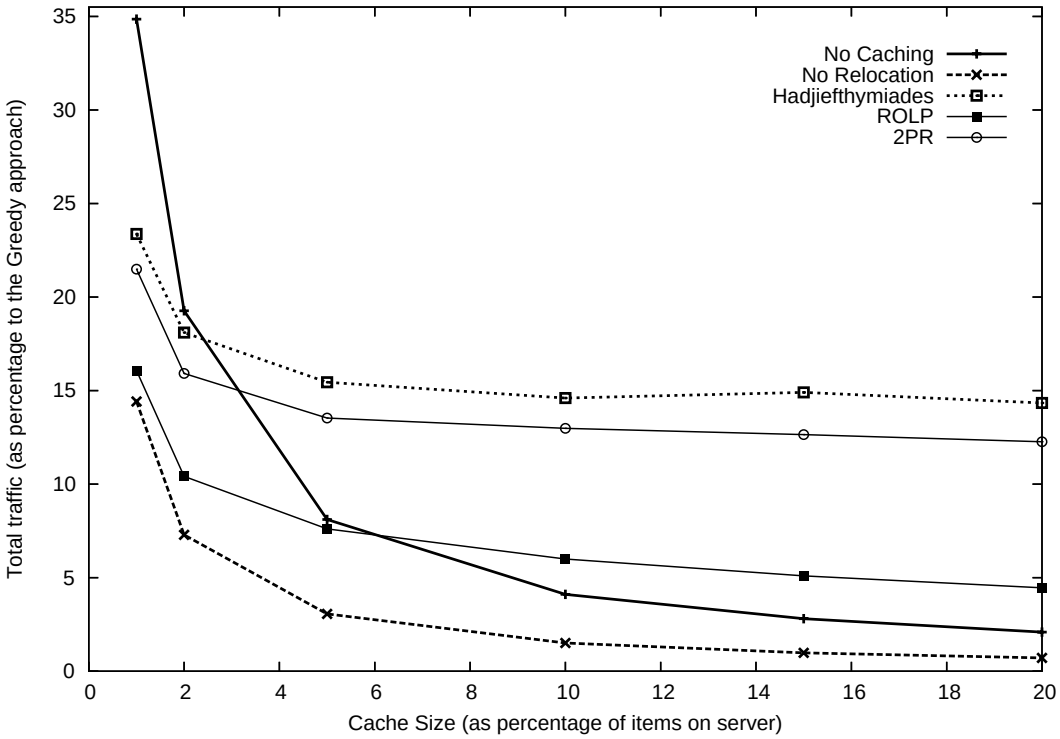


Fig. 16. Total traffic as percentage of Greedy vs. cache size.

n_o, and none of the nodes below n_i has cached the object. As a result, the probability that a request is answered by a node n_i is equal to:

$$Pr(n_{o,i}) = \left(1 - \sum_{x=1}^{\text{size}_{\text{cache}}} Pr(d_x)\right) \times \left(1 - \frac{\text{size}_{\text{intermediate}}}{D}\right)^{|p_{o,i}|-1} \times \frac{\text{size}_{\text{intermediate}}}{D} \tag{14}$$

where $|p_{o,i}|$ represents the length of the path $p_{o,i}$, in other words, the number of hops between n_o and n_i. The first term in Eq. (14) is the probability that the requested object is not found at n_o. The second term is the probability that the requested object is not found in the $i-1$ caches before n_i, and the last term is the probability that the requested object is found at node n_i. The product of these three terms gives the probability of a query from n_o being answered by an intermediate node n_i.

If none of the intermediate nodes has cached the requested object, the request will arrive at the remote server n_r. The probability of this happening can be calculated with the following equation:

$$Pr(n_{o,r}) = \left(1 - \sum_{x=1}^{\text{size}_{\text{cache}}} Pr(d_x)\right) \times \left(1 - \frac{\text{size}_{\text{intermediate}}}{D}\right)^{|p_{o,r}|-1} \tag{15}$$

That is, a request is answered by the root node (remote server) if it is not cached by the client's home location cache and not cached by the $|p_{o,r}| - 1$ intermediate nodes between n_o and n_r.

Lastly, the mean access delayed in steady state can be calculated by substituting Eqs (12), (13), (14) and (15) into Eq. (11).

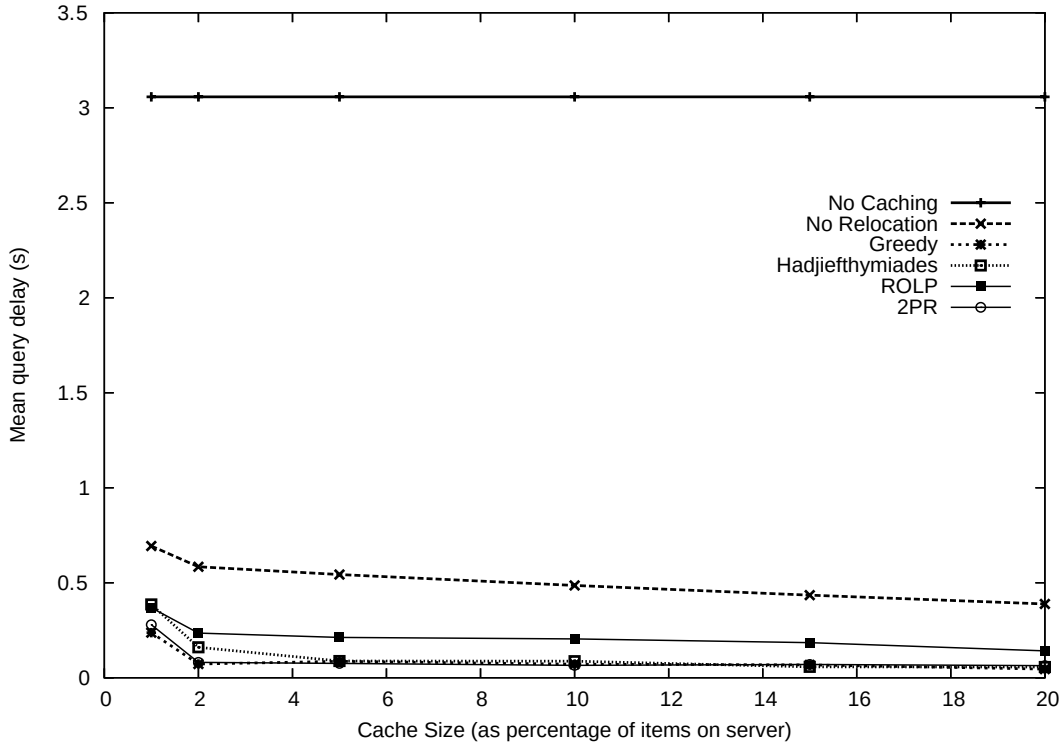


Fig. 17. Mean query delay vs. cache size.

6.2.3. Transient state analysis

Next, an analysis of how moving between cells affects the mean access delay is presented. In the early stages of a mobile client entering a new cell, the value of $Pr(n_{o,o})$ changes depending how long it has been since the client entered the new cell.

Assume clients move between cells at exponentially distributed intervals with mean time between relocation of t_r. Let τ denote the time between a request being received and the time when the client first enter the current cell. The probability that τ is equal to some $t \in \{0, \infty\}$ is:

$$Pr(\tau = t) = \frac{1}{t_r} e^{-\frac{t}{t_r}} \tag{16}$$

Requests for objects are modelled with a Poisson process. Given a request for d_i, and the client has been in the cell for t (that is $\tau = t$), the probability that d_i can be found in the cache at n_o can be expressed as:

$$Pr(n_{o,o}|\tau = t) = 1 - e^{-\lambda_i \times t} \tag{17}$$

where $\lambda_i = Pr(d_i) \times \lambda$, $Pr(d_i)$ is the probability of object d_i being requested and λ is the rate at which each client generates requests.

The value of $Pr(n_{o,o}|\tau = t)$ is plotted against t for a cache of 20 objects in Fig. 8 (for $\lambda = 5$/s) and Fig. 9 (for $\lambda = 1$/s). It can be seen from the graphs that even for a small cache with only 20 objects, it takes quite some time after the client enters the new cell $(t = 0)$ before the cache returns to steady state. This result shows the importance of object relocation. A good relocation scheme should reduce or even

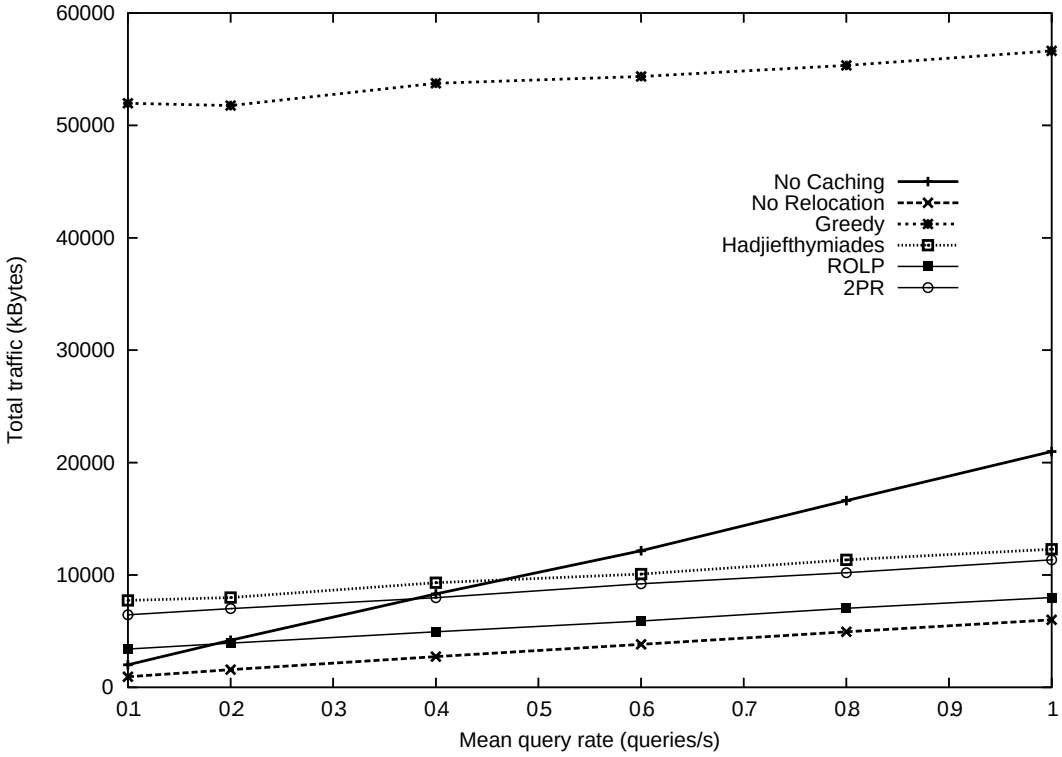


Fig. 18. Total traffic vs. client query rate.

eliminate the occurrence of the transient state, so that mobile clients do not experience any performance degradation in accessing its home location cache after moving into a new cell.

Finally, an expression for $Pr(n_{o,o})$, that is, the probability that a request is answered by the cache at n_o in transient state, is derived by combining Eqs (16) and (17).

$$Pr(n_{o,o}) = \sum_{x=1}^{\text{size}_{\text{cache}}} Pr(d_x) \times \left[\int_0^{\infty} \left(\frac{1}{t_r} e^{-\frac{t}{t_r}} \right) \left(1 - e^{-\lambda_i t} \right) dt \right] \tag{18}$$

where $\text{size}_{\text{cache}}$ is the size of the home location cache and $Pr(d_x)$ is the probability the x-th object is requested. Note that as the value of $t_r \to \infty$, Eq. (18) simplifies to $\sum_{x=1}^{\text{size}_{\text{cache}}} Pr(d_x)$, which is the same as the equation obtained in the steady state analysis. As caches at intermediate nodes further up the hierarchy are shared among all clients, the value of $Pr(n_{o,i})$ for $i \in \{(o+1)..r\}$ is not affected by client movements and has the same function as shown in Eqs (14) and (15).

In Fig. 10, the mean access delay in transient state for clients with different movement rates are shown. It can be seen that the query delay during transient state is higher compared to those in steady state. This is especially the case for clients who move between cells frequently.

6.3. Relocation overhead

In the previous section, it has been shown analytically that clients experience significant increase in access delays after handovers due to the reduced effectiveness of their home location caches. By

dynamically relocating cached objects, home location caches can remain in steady state even as mobile clients move between cells, thus hiding the effect of handovers from the clients.

In this section, the traffic overhead associated with various cache relocation schemes are compared. Specifically, the *Delayed* approach, the *Greedy* approach, the strategy proposed by *Hadjiefthymiades* [4] and the ROLP scheme are studied.

6.3.1. Delayed approach

In the $Delayed$ Relocation method, objects at the origin node are relocated to the new node after a handover takes place. Assume a client is moving from n_o to another node n_d. The cost of relocating using the *Delayed* approach can be expressed as:

$$C_{\text{delayed}} = \sum_{\forall\, l_{i,j} \in p_{o,d}} (\text{size}_{\text{cache}} \times\ \text{cost}_{i,j}) \tag{19}$$

where $\text{size}_{\text{cache}}$ is the number of objects in the home location cache at the origin node (n_o). The cost of transmitting an object over a link $l_{i,j}$ is denoted $cost_{i,j}$, and $p_{o,d}$ represents a path from n_o to n_d.

6.3.2. Greedy approach

Objects are copied to all neighbour node of the origin node in the *Greedy* approach to ensure the performance of the home location cache is not affected by handovers. As a result, the traffic generated by the *Greedy* algorithm is dependent on the number of neighbouring cells the origin cell has. Assume the client is moving from n_o with k neighbours. The set of neighbours is denoted M, such that $M = \{n_x\}$ for $x \in 1..k$. The cost of the *Greedy* approach is,

$$C_{\text{greedy}} = \sum_{x=1}^{k} \left(\sum_{\forall\, l_{i,j} \in p_{o,x}} (\text{size}_{\text{cache}} \times\ \text{cost}_{i,j}) \right) \tag{20}$$

6.3.3. Hadjiefthymiades' method

The approach proposed by Hadjiefthymiades relocates 100%, 70% or 30% of the home location cache to the neighbour nodes of the origin node depending the likelihood with which a client will move into each cell. Assume the origin cell has k neighbours, let n_a denote the most likely destination, the next two most likely nodes are denoted n_b and n_c, and the remaining nodes n_x where n_x is in the set $\{n_x | x \in 1..(k-3)\}$. The cost of relocation with this approach is equal to:

$$\begin{aligned} C_{\text{hadjief}} = &\left(\sum_{\forall\, l_{i,j} \in p_{o,a}} (\text{size}_{\text{cache}} \times\ \text{cost}_{i,j}) \right) + \left(0.7 \times \sum_{\forall\, l_{i,j} \in p_{o,b}} (\text{size}_{\text{cache}} \times\ \text{cost}_{i,j}) \right) \\ &+ \left(0.7 \times \sum_{\forall\, l_{i,j} \in p_{o,c}} (\text{size}_{\text{cache}} \times\ \text{cost}_{i,j}) \right) + \left(0.3 \times \sum_{x=1}^{k-3} \left(\sum_{\forall\, l_{i,j} \in p_{o,x}} (\text{size}_{\text{cache}} \times\ \text{cost}_{i,j}) \right) \right) \end{aligned} \tag{21}$$

6.3.4. The ROLP method

In ROLP, objects are relocated from the closest node to the destination. Assume the origin node is n_o and the destination node is n_d. To calculate the cost of relocation using ROLP, it is necessary to first calculate the probability that an object is relocated from n_y, where $n_y \in p_{o,d}$. As caches at the

intermediate nodes are shared by all clients, the probability of finding an object d_i in an intermediate node n_y is equal to $\frac{\text{size}_{\text{intermediate}}}{D}$, where $\text{size}_{\text{intermediate}}$ is the number of objects cached at n_y and D is the total number of objects at the server. Therefore, the probability of an object being relocated from an intermediate node n_y to n_o is equal to,

$$Pr_{\text{relocate}}(n_{o,d,y}) = \binom{\text{size}_{\text{intermediate}}}{N} \times \left(1 - \binom{\text{size}_{\text{intermediate}}}{N}\right)^{|p_{y,d}|} \tag{22}$$

That is, an object is relocated from n_y if the object is found in the cache of n_y and it is not found in the $|p_{y,d}|$ nodes nearer to the destination.

Any objects not relocated from an intermediate node is relocated from the home location cache at n_o. The probability of this happening is:

$$Pr_{\text{relocate}}(n_{o,d,o}) = \left(1 - \binom{\text{size}_{\text{intermediate}}}{N}\right)^{|p_{o,d}|} \tag{23}$$

Based on Eqs (22) and (23), the relocation cost of the ROLP scheme for a client moving from a node with k neighbours can be expressed as follows,

$$C_{\text{ROLP}} = \sum_{x=1}^{k} \left[\text{size}_{\text{cache}} \times \sum_{y=o}^{d} \binom{Pr_{\text{relocate}}(n_{o,d,y})}{\times \sum cost_{i,j} | \forall\, l_{i,j} \in p_{y,d}} \right] \tag{24}$$

7. Simulation and results

This section summarises some of the results obtained from the simulation of the cache relocation algorithms we have studied. Simulation is used to evaluate the performance of the proposed method as it allows us to obtain results in a controlled environment, that would otherwise not be possible in a real environment. It also allow us to repeat the same experiences for each of the techniques studied so that compariable results can be obtained. The simulation is implemented using the OMNET++ simulation package [20].

The simulation model consists of a network hierarchy of 20 nodes, which allow us to model the typical behaviour of mobile clients moving between cells. One of the nodes is selected as the master remote server holding the original copy of all data objects. Each simulation run lasted for 10000 seconds (simulation time). During this time, mobile clients move between cells in the network where the time between handovers is exponentially distributed. The mean time between handovers is set to 500 s. Mobile clients generate queries following a Poisson distribution with mean rate of 0.2 queries per second. To model access locality, queries are distributed among data objects following a Zipf distribution. The average time between handovers is set to 500 s.

Figure 11 shows a screenshot of the network modelled in OMNET++. The default value of the parameters used in the simulation are shown in Fig. 12.

In the simulation, the following schemes are compared:

- No Caching g: No network caches used.
- No Relocation: Caches used, but do not relocate.
- Greedy: Copy client's home location cache to all adjacent nodes prior to handover.

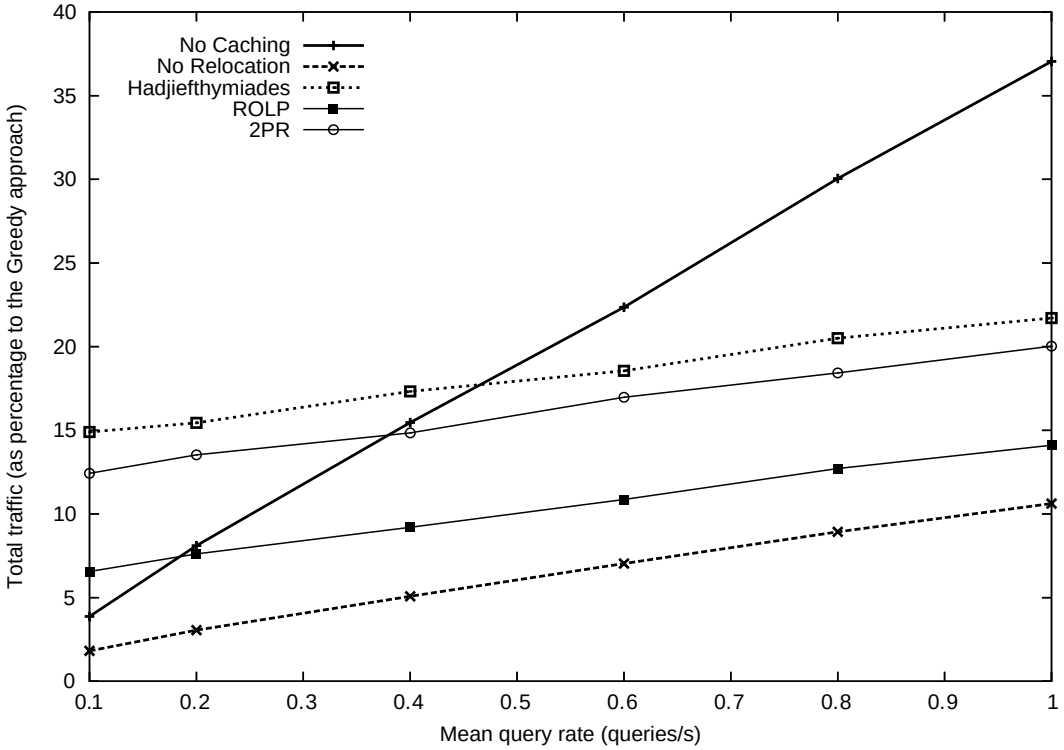


Fig. 19. Total traffic as percentage to Greedy vs. client query rate.

– Hadjiefthymiades: The method proposed in [4]. (See Section 2 for details)
– ROLP (See Section 5)
– 2PR (See Section 4)

The "No Caching", "No Relocation" and "Greedy" schemes are used in the evaluation as they represent the various extremes of cache relocation strategies. The "No Caching" scheme represents the scenario where no network-level caches are used. The "No Relocation" scheme represents most existing static caching strategies where network caches are used but are not dynamically relocated as clients move between nodes. The "Greedy" strategy ensures clients will not be affected by handovers by relocating the content of clients' home location caches to all possible destinations. However, the overhead of this strategy also represents an upper bound to cache relocation overhead. Lastly, the "Hadjiefthymiades" [4] scheme is used in the comparison as it is the most recent approach proposed to deal with cache relocation in mobile environments.

7.1. Query delay for 1,000 queries

Figures 13 and 14 show the query delay for 1,000 queries in two simulation runs. In the first run, cache relocation is not applied (*No Relocation*), while in the second run, cache relocation (the ROLP scheme) is applied. The dotted lines in the figures represent handovers. It is found that if cache relocation is not used, a period of high query delay is experienced by the mobile client each time it moves into a new cell. This is because when the home location cache is not relocated with the client's movement, it must be rebuilt at the new location. As it takes time for the cache to return to a steady state (that is, for the cache

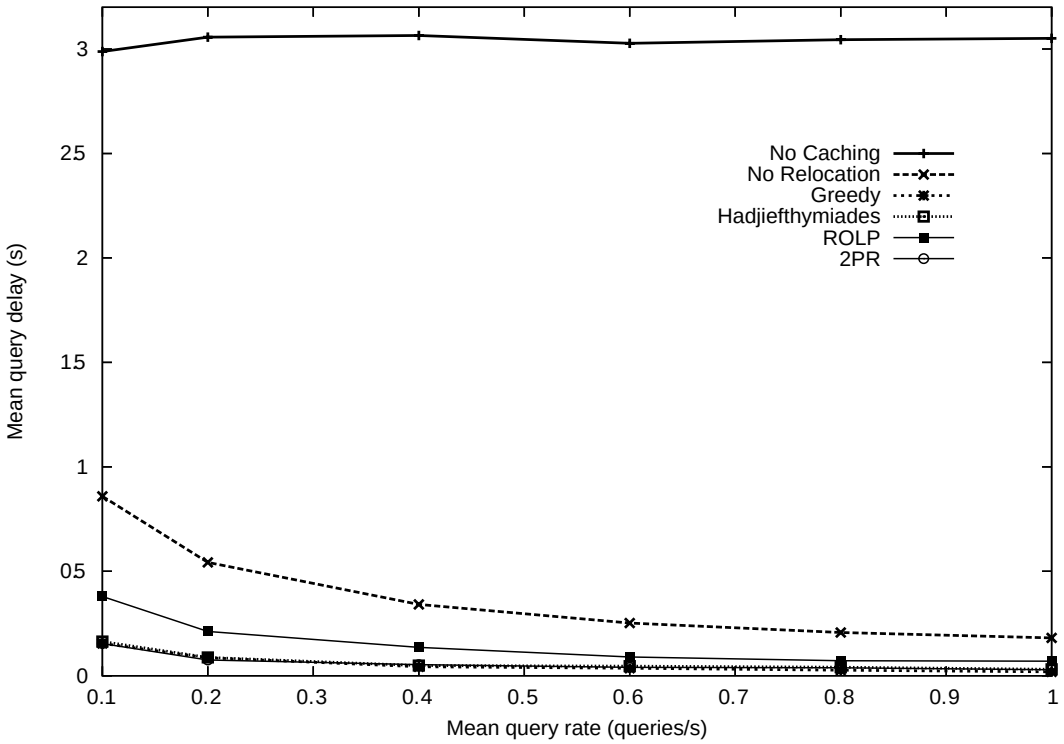


Fig. 20. Mean query delay vs. client query rate.

to be filled with the objects the client most frequently accesses), the number of cache misses increases after each handover. On the other hand, if the home location cache dynamically relocates as the client moves between cells, the amount of time it takes for the cache to return to steady state is reduced. This results in lower mean query delay, making the handover more transparent to the mobile client.

An interesting observation in this experiment is that over time, the performance of the home location cache improves when cache relocation is applied. From Figs 13 and 14, it can be seen that while query delay is similar across all the queries when cache relocation is not used, the delay drops significantly (after around 400 queries) when cache relocation is used. This can be explained by realising that as a home location cache follows a client around, over time, the set of data objects it holds become more closely matched to the objects most frequently accessed by the client. On the other hand, when the home location cache does not relocate, it must "relearn" which data objects are of interest to the client after each handover. This results in higher number of cache misses, thus higher query delays after handovers.

7.2. The effect of home location cache size

Next, the effect of different home location cache sizes is investigated. Figure 15 shows the total traffic of the various approaches during the simulation plotted against home location cache size. While larger home location caches allow more data objects to be cached at the base stations, the cost of relocation is also higher. The *Greedy* scheme clearly performs the worst in terms of traffic as it copies the objects at the home location cache to all adjacent cells, resulting in very high relocation overhead. On the other hand, the *No Caching* and *No Relocation* approaches have the lowest overheads, however as will be shown

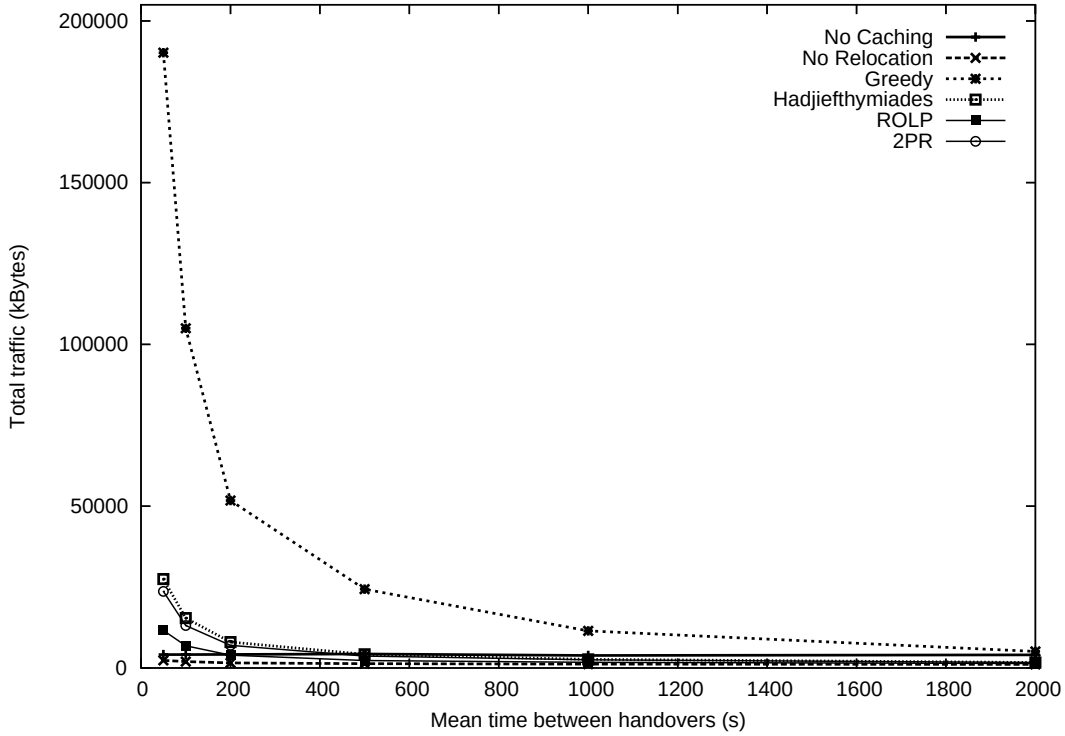


Fig. 21. Total traffic vs. mean time between handovers.

later on, they do not provide transparency for mobile clients between handovers. The proposed schemes ROLP and 2PR both perform consistently better compared to the *Greedy* algorithm and *Hadjieftymiades'* approach. This is especially the case when the size of the home location cache is large.

It was shown in Fig. 15 that the traffic overhead of *Greedy* is significantly higher than the other approaches tested. To more clearly compare the performance of the other approaches, the total traffic of these approaches are plotted as a percentage of traffic of the *Greedy* approach in Fig. 16. The results show that *No Relocation* has the lowest traffic overhead across the range of cache sizes. This is because in *No Relocation*, home location caches are not relocated as clients move, thus no traffic is caused by cache relocation. It is found the traffic overhead of ROLP is between 33 to 65% lower than Hadjiefthymiades' method, and between $\frac{1}{6}$ and $\frac{1}{20}$ of the *Greedy* approach. This saving is achieved because in ROLP, only objects that do not exist at intermediate nodes are relocated from the origin, thus the number of objects and the distance the objects need to travel during relocation is reduced. The 2PR scheme lowers total traffic by between 8 to 15% compared to Hadjiefthymiades's method by reducing the penalty of incorrect path predictions. The saving is consistent over the whole range of cache sizes.

The mean query delay for ROLP and other schemes tested are plotted against the client's HLC size in Fig. 17. In this graph, the client's HLC size is represented as a percentage of the number of data objects on the server. The graphs show that relocating home location caches always results in better query delay than not using cache relocation. Furthermore, the use of home location caches reduces mean query delay by more than six fold compared to the *No Caching* scheme. In relation to cache size, it is found that mean query delay decreases linearly as cache size increases. This can be explained by realising that as the size of the home location caches increase, more queries are answered by these caches, thus reducing the need for requests to travel all the way to the remote server which causes much higher query delays. The query

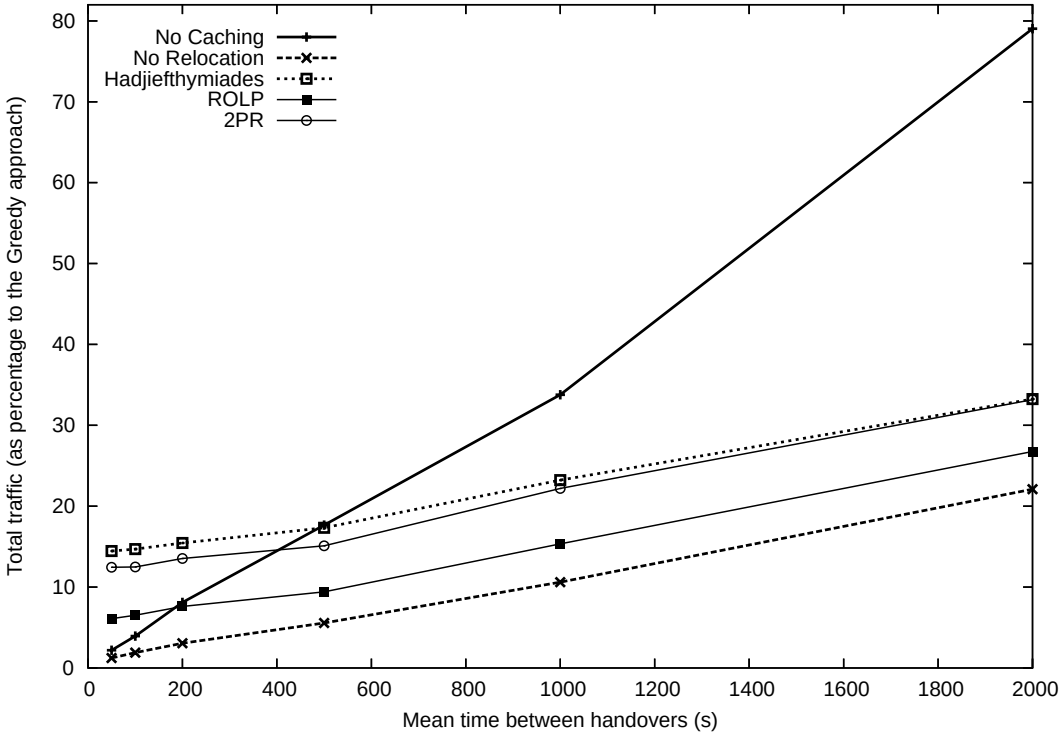


Fig. 22. Total traffic as percentage of Greedy vs. mean time between handovers.

delays achieved by ROLP is roughly double of that achieved by 2PR. The 2PR scheme achieves query delays that are very similar to those achieved by the *Greedy* scheme, however at a much lower traffic overhead as shown previously in Figs 15 and 16.

7.3. The effect of client query rate

In this section, the effect of client query rate is studied. Figure 18 shows the total traffic during the simulation against client query rate, while Fig. 19 plots the total traffic of the various schemes as a percentage of the *Greedy* approach against clients' query rate. The results show that the increase in query rate affects the various schemes in a similar fashion. It is found that the increase in total traffic grows linearly as client query rate increases. The only scheme that performs differently is the *No Caching* approach. In this case, the total traffic increases much faster compared to the other approaches as the query rate increases. This is because in the *No Caching* approach, no network-level caches are used. This means requests for data objects must travel higher up the network hierarchy. As the query rate increases, more requests are sent into the network, resulting in an increase in network traffic. On the other hand, approaches that use home location caches allow many requests to be answered near the edge of the network, thus achieving a slower increase in total traffic compared to *No HLC*. At a query rate of 1 query/s, the use of home location cache and cache relocation halves the traffic overheads compared to not using home location caches (*No Caching*). This result highlights the importance of using home location caches, especially for systems where the client query rate is high.

The mean query delay is shown against the clients' query rate in Fig. 20. Unexpectedly, the mean query delay decreases slowly as the client query rate increases. At high query rates (for example 1/s),

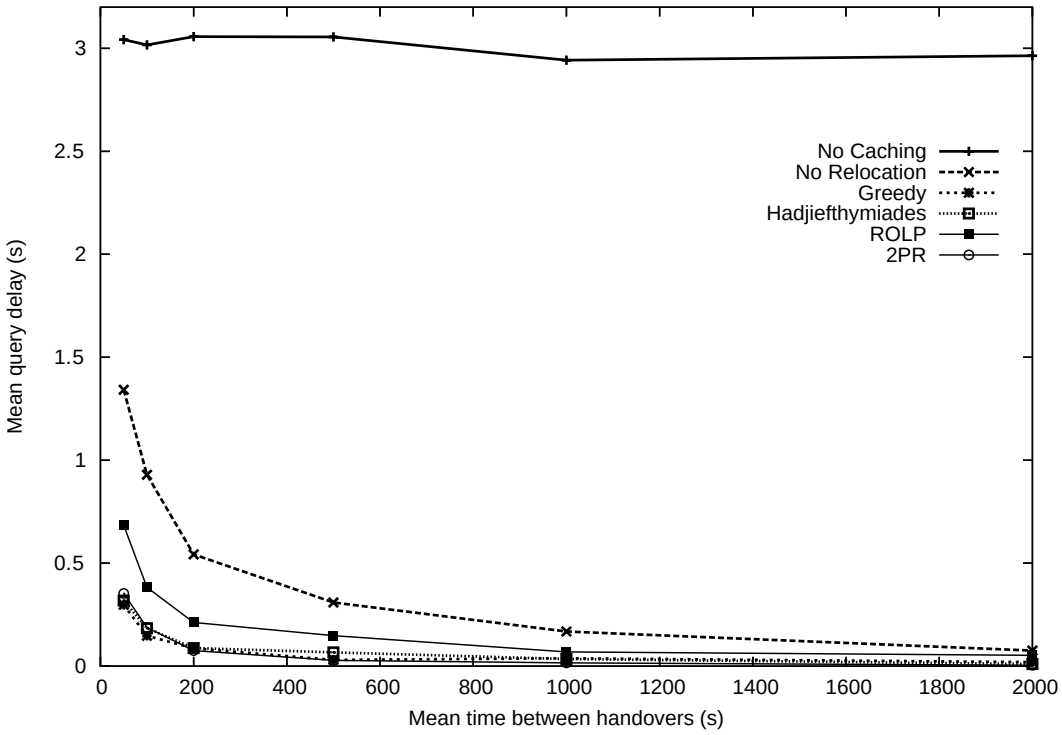


Fig. 23. Mean query delay vs. mean time between handovers.

the mean query delay is up to four times lower than when query rate is low (for example 0.1/s). It is found that this is due to the fact that at high query rates, clients perform more queries between each handover, thus allow them to build up their home locations caches and gain more benefits from them at each location. On the other hand, when cache relocation is not used, similar performance is experienced by the clients across all query rates.

7.4. The effect of mean time between handovers

So far, the results obtained have shown that cache relocation can significantly reduce the query delay experienced by mobile clients after handovers. However, this is at the expense of increased network traffic. For clients who frequently move between cells, the use of cache relocation may lead to very high relocation overheads. To gain a better understanding of the trade-off between network traffic overhead and the improvement in query delay, this section studies how the frequency of handovers affect the performance of the various cache relocation schemes.

Figures 21 and 22 show the total relocation traffic against mean time between handovers for ROLP and 2PR compared to the other schemes. When the mean time between handovers is low, the total traffic caused by cache relocations can become very high. This is because home location caches are relocated when a mobile client moves between cells. The more frequently mobile clients cross cell boundaries, the more often home location caches need to relocate, thus the higher traffic generated. As the mean time between handovers reduces, the overhead of using relocation also reduces significantly. From the graphs, it can be seen that once the mean time between handovers reaches around 200 seconds, the overhead of ROLP and 2PR becomes relatively close to the *No Relocation* approach. As cells in a mobile network are

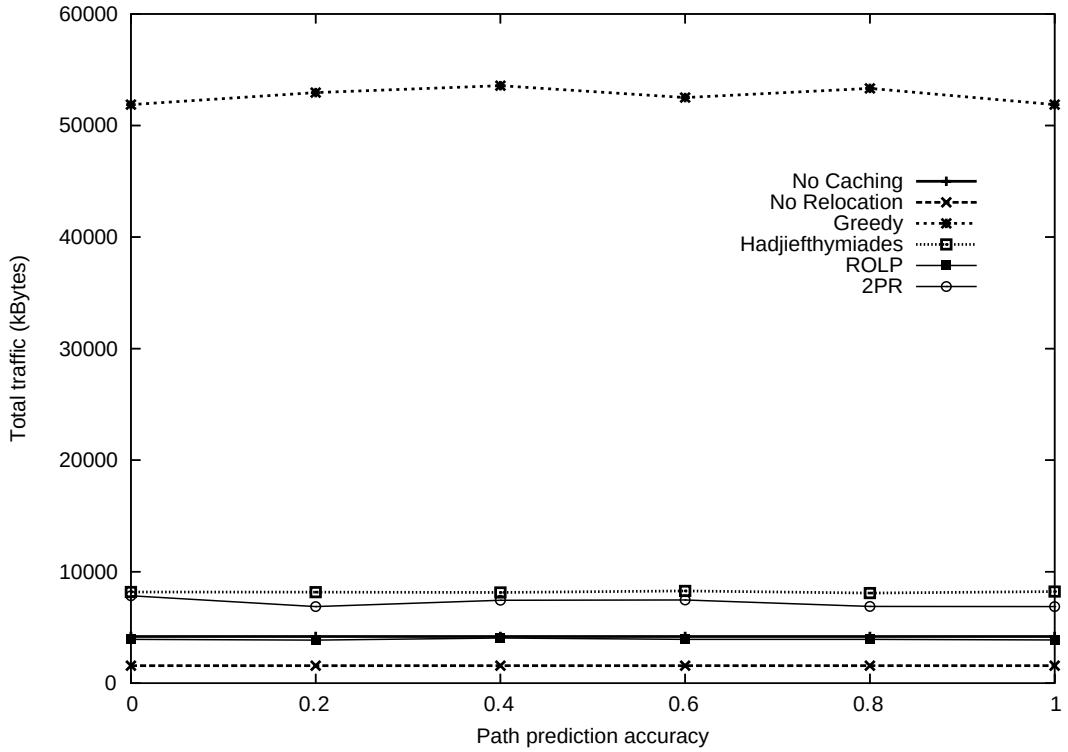


Fig. 24. Total traffic vs. path prediction accuracy.

usually a few kilometers in radius, it is unlikely for pedestrians or slow vehicles to cross cell boundaries more frequently than 200 seconds. This justifies the use of cache relocation for this type of clients. On the other hand, for high speed clients who frequently cross cell boundaries (for example vehicles on a highway), the traffic caused by cache relocation may outweigh the benefits gained from cache relocation.

Figures 23 describes how the mean query delay changes for clients with different mean time between handovers. Again, the results confirm that clients experience the highest query delay when home location caches are not used. This is due to the fact that without caching, client requests must travel longer distances in order to be answered. For the other schemes tested, it is found that the mean query delay decreases as the mean time between handovers increases. This is to be expected because when a client stays in the same cell for a long period of time, the most frequently accessed objects will become available at its home location cache, resulting in a higher cache hit ratio, thus the query delay is in turn reduced. The results show that using ROLP and 2PR, the mean query delay for clients who frequently move between cells is less than half of that compared to the case if no cache relocation is used.

7.5. The effect of path prediction accuracy

The total traffic caused by ROLP and 2PR is plotted against the probability of correct path predictions in Fig. 24. The cost of relocating using ROLP and 2PR is much lower than Hadjiefthymiades' strategy. This is because in ROLP, the number of objects relocated is reduced by using object lists to coordinate the relocation process. In 2PR, only a small portion of objects (those most frequently accessed) are relocated to the most likely destination node. The remaining objects are temporarily placed at a higher

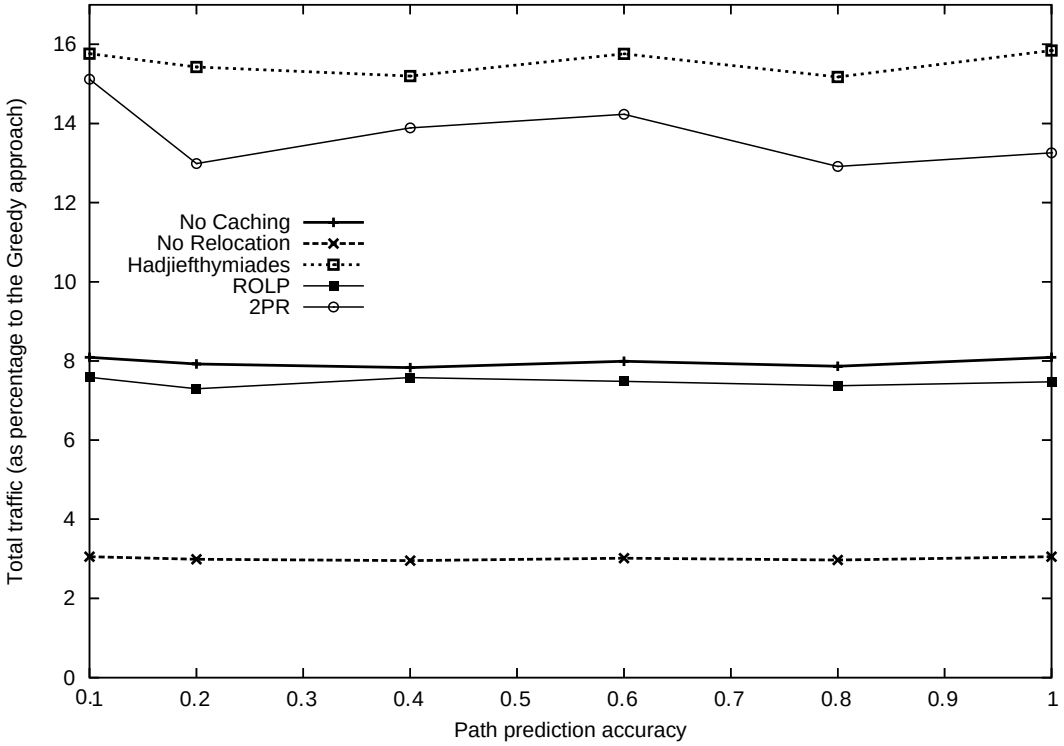


Fig. 25. Total traffic as percentage of Greedy vs. path prediction accuracy.

common parent node until the location of the client is confirmed. This method reduces the dependency on path prediction accuracy, and compensates for cases when the client does not end up going to the most likely destination. Furthermore, as it avoids the unnecessary copying objects to multiple nodes, the overhead of relocation is reduced.

Figure 25 shows the total traffic as a percentage of *Greedy* against path prediction accuracy. It is found that overall the 2PR scheme is able to reduce the traffic overhead compared to Hadjiefthymiades' algorithm. An interesting result is that the path prediction accuracy does not have a strong impact on traffic generated if the system does not try to recover from incorrect path prediction. In our implementation of the various relocation techniques, it is assumed the system does not try to correct itself if a home location cache is relocated to the wrong location. As a result, even when a path prediction is incorrect, there will only be a slight increase in network traffic due to more uplink requests at the new location. However, the effect of incorrect path predictions will become evident in the next graph.

Figure 26 shows the mean query delay of the various approaches at different path prediction accuracy. The mean query delay is high when home location caches are not used, or when home location caches do not relocate. This is due to the fact that without caching and dynamic relocation, many client queries need to travel to the remote server to be answered. On the other hand, the $Greedy$ algorithm provides the best query delay because no matter to which cell a mobile client moves, all the objects cached at the origin node will also be available at the new node. However, the *Greedy* algorithm generates a large amount of traffic overhead, making it an expensive method to use. 2PR provides a similar mean query delay compared to Hadjiefthymiades' method because at high prediction accuracy, clients are likely to move into the most probable destination cell. Since both 2PR and Hadjiefthymiades' ensure the most

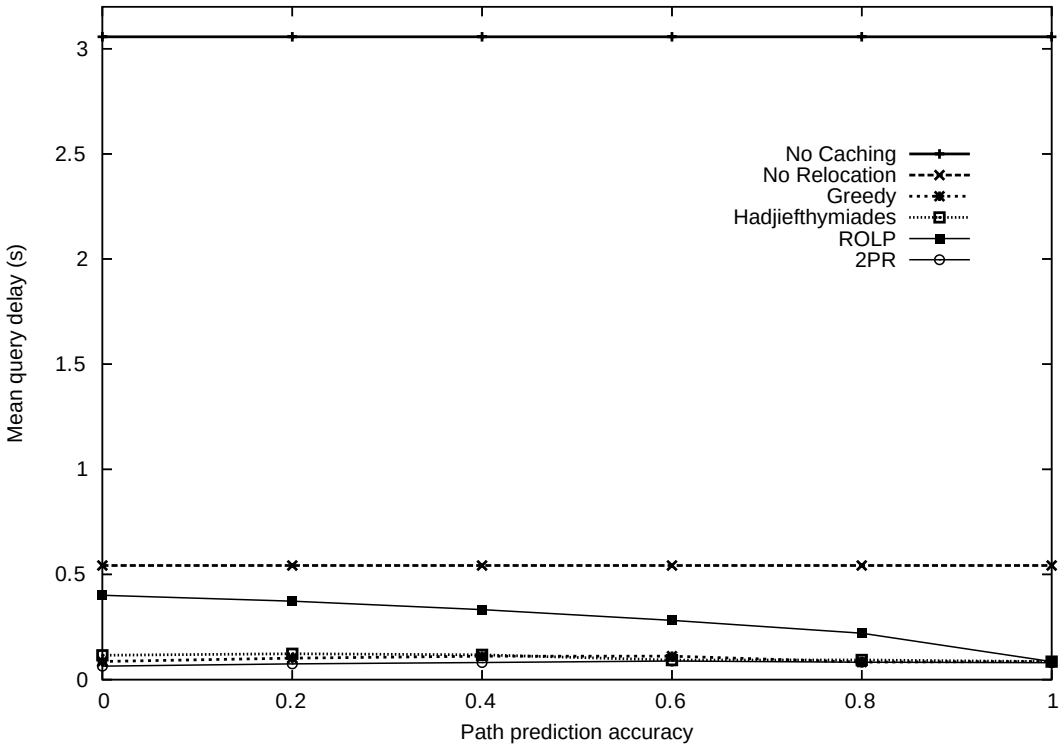


Fig. 26. Mean query delay vs. path prediction accuracy.

frequently used objects are in the most probable cell, they both provide good performance. The query delay of ROLP is high when path prediction accuracy is low, however this improves with better path prediction. The reason this occurs is because unlike 2PR and $Hadjiefthymiades$, ROLP does not have built into it a mechanism to deal with incorrect path prediction. This results in objects being relocated to the wrong destinations, thus increasing query delay.

8. Summary

This paper investigated the effects of handovers on client's perceived performance. It has shown that as mobile clients move from one cell to another, they experience increased delay when accessing objects from their home location caches. This is especially the case for mobile clients who move been cells frequently.

To deal with this problem, two new techniques are proposed to improve the performance of existing cache relocation methods for mobile networks. In the two-phase relocation method (2PR), a client's home location cache is first relocated to a common parent of the origin node and its neighbouring nodes, so that even if the client does not move into the most likely destination cell, the system will still be able to recover quickly. Simulation results demonstrate that 2PR achieves lower query delays than existing schemes by reducing the risk of incorrect path predictions. The second technique, Return-path Object List Passing (ROLP), reduces the amount of network traffic generated by cache relocation by relocating objects from intermediate nodes rather than the origin node. Test results showed that ROLP reduce the traffic associated with relocation by between 33 and 65% compared to existing schemes.

In addition to simulation, detailed analytical models have been developed to compare existing solutions based on realistic parameters, including query delay and traffic overhead. Results obtained from the analytical study and simulation both indicate that the proposed methods are effective in reducing the overhead associated with cache relocation and allow clients to move between cells without experiencing high query delays.

Acknowledgment

This project is supported by the ARC (Australian Research Council – under the Linkage-Project scheme, no. LP0218853) and SUN Microsystems grant no.7832-030217-AUS.

References

[1] G. Association, *An overview of SMS*, http://gsmworld.com/technology/sms/index.shtml, 2004.
[2] A. Kahol, S. Khurana and S. Gupta, A strategy to manage cache consistency in a disconnected distributed environment, *IEEE Transactions on Parallel and Distributed Systems* **12**(7) (July 2001), 686–700.
[3] L. Bright and L. Raschid, *Efficient remote data access in a mobile computing environment*, in Proceedings of the International Workshop on Parallel Processing, August 2000, 57–64.
[4] S. Hadjiefthymiades and L. Merakos, *Using proxy cache relocation to accelerate web browsing in wireless/mobile communications*, in Proceedings of the International World Wide Web Conference, 2001, 26–35.
[5] Y. Li, C. Chiang and M. Liu, *Effective web caching for GPRS networks*, in Proceedings of the IEEE International Conferece on Computer Networks and Mobile Computing, October 2001, 85–90.
[6] P. Cao and S. Irani, *Cost-aware WWW proxy caching algorithm*, in Proceedings of the USENIX Symposium on Internet Technologies and Systems, December 1997, 193–206.
[7] A. Rousskov and V. Soloviev, A performance study of the Squid proxy on HTTP/1.0, *World Wide Web* **2**(1–2) (1999), 47–67.
[8] M. Bowman, P. Danzig, D. Hardy, U. Manber and M. Schwartz, *The Harvest information discovery and access system*, in Proceedings of the International WWW Conference, October 1994, 17–20.
[9] D. Muntz and P. Honeyman, *Multi-level caching in distributed file systems – or – your cache ain't nuthin' but tras*, in Proceedings of the USENIX Winter Conference, January 1992, 305–313.
[10] T. Liu, P. Bahl and I. Chlamtac, Mobility modeling, location tracking, and trajectory prediction in wireless networks, *IEEE Journal on Selected Areas in Communications* **16**(6) (1998), 922–936.
[11] W. Su, S. Lee and M. Gerla, *Mobility prediction and routing in ad hoc wireless networks*, International Journal on Network Management, 2000.
[12] T. Watanabe, A. Mori and Y. Yamamoto, *Mobile cache protocol: A dynamic object relocation protocol for wide area networks*, in Proceedings of the IEEE International Conference on Distributed Computing Systems, April 2000, 420–427.
[13] G. Liu and G.M., Jr, A class of mobile motion prediction algorithms and wireless mobile computing and communications, *ACM MONET* **1**(2) (October 1996), 113–121.
[14] J. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *The American Mathematical Society* **7** (1956), 48–50.
[15] T. Cormen, C. Leiserson and R. Rivest, *Introduction to algorithms*, MIT Press, Cambridge MA, 1990.
[16] R. Prim, Shortest connection networks and some generalizations, *Bell System Technical Journal* **36** (1957), 1389–1401.
[17] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, *Web caching and Zipf-like distributions: Evidence and implications*, in Proceedings of the INFOCOM, 1999, 126–134.
[18] E. O'Neil and P. O'Neil, *The LRU-k page replacement algorithm for database disk buffering*, in Proceedings of the ACM SIGMOD conference, May 1993, 296–306.
[19] J. Xu, Q. Hu, W. Lee and D. Lee, Performance evaluation of an optimal cache replacement policy for wireless data dissemination, *IEEE Transaction on Knowledge and Data Engineering* **16**(1) (January 2004), 125–139.
[20] A. Varga, *Omnet++ user manual*, http://whale.hit.bme.hu/omnetpp, 2003.
[21] D.L. Lee, M. Zhu and H. Hu, When location-based services meet database, *Mobile Information Systems* **1**(2) (2005), 81–90.