ABSTRACT

Mobile networks are becoming increasingly popular as a means for distributing information to a large number of users. In comparison to wired networks, mobile networks are distinguished by a potentially much higher variability in client demand due to user mobility. Most previous content distribution techniques assume a static client demand distribution and, thus, may not perform well in mobile networks.

This paper proposes and analyzes a Mobile Dynamic Content Distribution Network model, which takes demand variations into account to decide whether to replicate a content and whether to remove previously created replicas in order to minimize total network traffic. We develop two solutions to our model: an offline optimal, which provides an ideal lower-bound on the total traffic, and a practical heuristic online algorithm, which uses demand forecasting to make replication decisions. We provide a thorough evaluation of our solutions, comparing them against ACDN, the only previous dynamic content placement algorithm targeting bandwidth minimization that we are aware of. Our results show that our online algorithm significantly outperforms the ACDN one, reducing total network traffic by up to 85% in a number of experiments covering a large system design space.

Categories and Subject Descriptors
I.6 [Computing Methodologies]: SIMULATION AND MODELING

General Terms
Dynamic Content Placement

Keywords
Dynamic Content Placement, CDN, Mobile Network, Online Algorithm, Simulation, Demand Forecasting

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSWiM’04, October 4-6, 2004, Venezia, Italy.
Copyright 2004 ACM 1-58113-953-5/04/0010 ...$5.00.

1. INTRODUCTION

As the number of mobile users grows and high bandwidth, third-generation mobile networks become available, new content delivery techniques are necessary to meet the increasing demand. One particular characteristic of such networks, which distinguish them from traditional wired networks, is a typically much higher variability in client demand. In mobile networks, variations in client demand are caused not only by changes in content popularity but also by user mobility. Therefore, content distribution techniques for mobile networks must take into account potentially very high spatial and temporal demand variations.

A Content Distribution Network, or simply CDN, replicates popular content at a number of servers in order to reduce user perceived latency and network bandwidth requirements. CDNs have been shown to be effective for managing content distribution to large numbers of users in wired networks [5, 9, 12, 15, 17]. However, most previous proposals assume a static client demand distribution. To the best of our knowledge, content placement for dynamically changing client demands, typical scenario in mobile networks, has only been addressed in a few recent studies [5, 7, 16, 10, 20]. Moreover, the only general solution that aims at minimizing total traffic we are aware of is the ACDN algorithm proposed in [16].

This paper proposes and analyzes a Mobile Dynamic Content Distribution Network (MDCDN) model which takes both temporal and spatial variations of client demand into account to dynamically reconfigure the system in order to minimize the total traffic over the network backbone. We provide two solutions to our mobile model: an offline optimal and an heuristic online solution. The offline solution is introduced as a mathematical programming model. We also propose a new online algorithm that approximates the optimal solution given by the offline optimal.

To evaluate our MDCDN solutions, we use a previously developed mobility simulator which models a city with a fixed number of users and a fixed number of previously placed servers [13]. We compare the total traffic generated in the solutions provided by our new online algorithm against the offline optimal and against the ACDN algorithm [16] for a large number of system configurations.

We found that our online algorithm produces solutions that are within 2.7 times the offline optimal, in terms of the total traffic generated over the network backbone. Fur-
Furthermore, in all of our experiments, our online algorithm produces systems with total network traffic up to 85% lower than the corresponding ACDN solutions.

In summary, this paper makes the following key contributions:

- A dynamic content distribution model for mobile networks, aiming at reducing total network traffic and providing a soft upper bound in the servers load.
- A optimization model that provides a lower bound on the traffic generated for dynamic content replication algorithms.
- A novel and efficient online algorithm based on demand forecasting.
- A thorough evaluation of our online algorithm, comparing it not only against the offline optimal solution but also against the ACDN algorithm\[?\], the only previous online dynamic content placement algorithm we are aware of.

The remaining of this paper is organized as follows. Next section background on CDN strategies. Section 3 introduces the mathematical programming model that implements the offline optimal solution as well as the online heuristic algorithm. Section 4 presents a description of our simulation environment. Section 5 presents the main results of a thorough performance evaluation of the MDCDN algorithms. Conclusions and future work are offered in Section 7.

2. BACKGROUND

A Content Distribution Network (CDN) is a system that allocates replicas of its content over servers geographically scattered, placed close to high demand sites in order to reduce client latency and network traffic [9]. The three basic CDN design problems, namely, server content, content placement and server selection, have been addressed by a large number of previous studies, mainly for wired networks [2, 4, 8, 9, 11, 12, 15, 17, 21, 18]. This paper focuses on content placement for mobile networks i.e., determine the most cost effective number and placement of each content replica, a non trivial design problem that has not been fully addressed in the past.

The addition of a new content replica is expected to reduce client latency and network traffic to serve client requests. However, it also incurs an extra traffic during content replication and maintenance operations. Thus, the decision of adding a new replica to a CDN involves the solution of a complex tradeoff between forwarding requests to remote servers and replicating and maintaining a new replica.

The content placement problem, previously shown to be NP-Complete [6], has been studied mostly for static client demands [8, 11, 12, 17, 15]. Content placement for dynamically changing demand have only been studied very recently [5, 7, 16].

In [5], the authors address the dynamic content placement problem in a tree topology aiming at minimizing the number of replicas deployed while meeting client QoS and server capacity constraints. Heuristics for dynamic content placement replication inside a cluster based cache are proposed in [7]. The only previous solution for general topologies we are aware of is the ACDN heuristic algorithm, proposed in [16].

The ACDN algorithm dynamically places replicas based on past observed demand. Like in the other two previous studies, the ACDN algorithm is proposed and evaluated in the context of wired networks. Although it could be applied to mobile networks, an evaluation of its performance in a mobile environment remains to be made.

CDN design strategies targeting, specifically, mobile networks have been addressed only by a few previous studies. In [19], the authors address server selection in a hierarchical streaming media mobile CDN aiming at reducing the number of mobile client handoffs. In [22], the authors propose a mobile streaming media CDN architecture, with the goal of supporting client mobility while efficiently utilizing network resources.

This work attempts to fill the gap left by previous work and develops not only an online dynamic content placement algorithm, designed for mobile networks, but also an offline optimal solution. It also provides an extensive performance evaluation of the proposed solutions, comparing them with the online ACDN algorithm, applied to mobile networks.

3. NEW CONTENT PLACEMENT TECHNIQUES FOR MOBILE NETWORKS

This section introduces the Mobile Dynamic Content Distribution Network (MDCDN) model. Section 3.1 gives an overview of the model. The offline optimal and online heuristic solutions proposed to implement MDCDN are described in sections 3.3 and 3.2.

3.1 Overview of the MDCDN Model

The MDCDN model is a framework for dynamic placement of content replicas in a mobile network. The model assumes a pre-defined fixed-location number of servers and coordinates content replication in these servers over time with the goals of minimizing total network traffic and providing soft upper bound on the servers load.

The MDCDN model uses the general term “content” to represent a collection of objects, which may change over time. A content could represent a collection of object of a given site or of a certain internet application (in comparison with [16]). Each content is stored in at least one server in the system, which we refer to as the origin server for that content. Mobile users request individual objects of a content. Each user request is always sent to the currently closest access point, due to wireless coverage limitation. We assumed that each access point may store some content locally and, thus, is also called a replica server. Therefore, it may or may not have a replica of the request content. If it has, the request is served locally. Otherwise, it forwards the request to the closest server that has the content replica and is not overloaded (see below), and relays the response to the client. Note that, in case of a hit in the closest access point, no traffic is generated over the network backbone. In case the request must be forwarded to a remote server, the indirect response generates traffic over the network backbone between the two servers.

In response to the current spatial distribution of client demand for objects of different contents, MDCDN performs content replication and/or removal of previously created replicas in the servers. The model allows for either total or partial content replication. If total content replication is performed, all the objects of a content are copied in a single
replication process. In case of partial content replication, individual popular objects are copied separately. In either case, the data is copied from the closest replica. Whenever a content is modified in its origin server, new content replicas or simply “patches” with the changes must be copied from the origin to each content replica server.

MDCDN characterizes each content $c$ by three size parameters: (1) the number of bytes transmitted during a replication ($sr^c$), (2) the number of bytes transmitted when a request is indirectly served ($si^c$), and (3) the number of bytes transmitted in each maintenance operation ($sm^c$). Partial and total replication can be implemented by setting $sr^c$ and $si^c$ appropriately. In case of partial replication, indirect request service and replication are performed on a per-object basis and, thus, $sr^c = si^c$. Otherwise, replication is performed for the whole content at once (i.e. $sr^c \geq si^c$).

In the MDCDN model, the total traffic generated over the backbone can be divided into three components: (1) the total traffic generated during content replications, (2) the total traffic generated by indirect client responses to requests for an object of the content, and (3) the total traffic generated for maintaining the existing content replicas up-to-date. Each traffic component is estimated as the product of the number of bytes transferred in each operation, the “distance” between the two servers involved in the operation and the number of operations. The distance between two servers can represent number of hops, geographical distance or any other measure that expresses the relative cost for transmitting between different pairs of servers.

As discussed in Section 2, the decision of adding a new replica implies the solution of a tradeoff between decreasing the traffic generated by indirect client responses and increasing the total traffic generated by replication and maintenance operations. There is a similar tradeoff for the removal of indirect service and maintenance operations over content.

The notation used in the model is:

\[ x_{ij}^c, y_{ij}^c, m_{ij}^c \]: Number of bytes transferred during a replication, indirect service and maintenance operations over content $c \in C$, respectively.

\[ dist_{ij} \]: Distance between the servers $i \in S$ and $j \in S$;

\[ o^c \]: Origin server of content $c \in C$;

\[ br_{ij}^c \]: Traffic generated if server $j \in S$ indirectly serves a request, forwarded by server $i \in S$, for an object of content $c \in C$;

\[ bm_{ij}^c \]: Traffic generated if server $i \in S$ has to update a local replica of content $c \in C$;

\[ y_{ij}^c \]: Indicates whether server $i \in S$ replicates content $c \in C$ from server $j \in S$ during period $t \in T$;

\[ a_{ij}^c \]: Indicates whether server $i \in S$ has the content $c \in C$ during period $t \in T$;

\[ z_{ij}^c \]: Number of times that server $i \in S$ forwards a request for an object of content $c \in C$ to server $j \in S$ during period $t \in T$.

The model inputs include the sets of contents, fixed location servers and reconfiguration periods. For each period, the set of client demands ($d^c_t$) as well as flags indicating whether each content is modified in its origin during the period are also inputs to the model. The size components characterizing each content, the distance between each pair of servers and the origin server for each content complete the list of input parameters. The remaining notations are used to represent the traffic, calculated as described in Section 3.1, generated by each indirect service, replication and maintenance operations, and variables used in the model formulation.

The offline optimal MDCDN solution is next:

\[
\begin{align*}
\min & \quad \sum_{c \in C} \sum_{t \in T} \sum_{i \in S} \sum_{j \in S} x_{ij}^c y_{ij}^c + \\
& \quad \sum_{c \in C} \sum_{t \in T} \sum_{i \in S} \sum_{j \in S} y_{ij}^c m_{ij}^c + \sum_{c \in C} \sum_{t \in T} \sum_{i \in S} \sum_{j \in S} a_{ij}^c m_{ij}^c b_{ij}^c
\end{align*}
\]

Subject to:

\[
\begin{align*}
& x_{ij}^c = sr^c \times dist_{ij}^c, & \forall c \in C, \forall i \in S, \forall j \in S \quad (1) \\
& br_{ij}^c = sr^c \times dist_{ij}^c, & \forall c \in C, \forall i \in S, \forall j \in S \quad (2) \\
& bm_{ij}^c = sm^c \times dist_{ij}^c, & \forall c \in C, \forall i \in S \quad (3) \\
& a_{ij}^c = 1, & \forall t \in T, \forall c \in C \quad (4) \\
& \sum_{j \in S} x_{ij}^c + d_i^c - a_{ij}^c = d_i^c, & \forall t \in T, \forall c \in C, \forall i \in S \quad (5) \\
& x_{ij}^c \leq d_i^c - a_{ij}^c, & \forall t \in T, \forall c \in C, \forall i \in S, \forall j \in S \quad (6) \\
& y_{ij}^c \leq a_{ij}^c, & \forall t \in T, \forall c \in C, \forall i \in S \quad (7) \\
& \sum_{j \in S} \sum_{c \in C} x_{ij}^c \leq HW, & \forall c \in C, \forall t \in T, \forall i \in S \quad (8) \\
& a_{ij}^{t+1} \geq 0, & \forall t \in T, \forall c \in C, \forall i \in S, \forall j \in S \quad (9) \\
& y_{ij}^c \in \{0, 1\}, & \forall t \in T, \forall c \in C, \forall i \in S, \forall j \in S \quad (10) \\
& a_{ij}^c \in \{0, 1\}, & \forall t \in T, \forall c \in C, \forall i \in S \quad (11)
\end{align*}
\]

The objective function minimizes the total traffic over the network backbone. The three sets of summations represent
the total traffic generated by indirect request service, content replication and replica maintenance.

The sets of constraints (1), (2) and (3) compute the traffic generated by each indirect request service, replication and replica maintenance, respectively. The set of constraints (4) ensures that each content is stored in it origin server. Constraints (5) ensure that client requests are served, either locally or by remote servers. The set of constraints (6) ensures that a server only forwards requests to servers containing the requested content. Constraints (7) guarantee that each replication creates a new replica. Constraints (8) ensure that the content being stored in the originating server.

The online algorithm is distributed and executed simultaneously by each server. Each server keeps track of the number of local client requests to each content received on each period and DES method to predict future demand for each content. Based on the predictions for a given content, the server estimates the total traffic generated to replicate the content and to maintain the new replica or to simply maintain the replica, if it already exists. It also estimates the total traffic generated if it does not replicate and, thus, has to forward future requests to the closest replica. The replica is copied to (removed from) the server only if the estimated traffic incurred by replication (or removal) is lower than the estimated traffic generated by the predicted indirect service. Thus, the algorithm is greedy, as each server independently makes a decision that, at the time, minimizes the total traffic generated for serving its predicted future demand.

Algorithm 1 New Online MDCDN

for ∀ content c ∈ C. do {Executed by each server i ∈ S at the ending of each period t ∈ T}

for k = 1, . . . , δ do

\( \bar{y}_{i+k} = \text{Forecast Local Demand}(c, i, t, k) \)

end for

\( d^* = \sum_{k=1}^{\delta} \bar{y}_{\text{LocalContent}(c, t, k)} \)
\( b^r = d^* \times s^r \times \text{dist}_{i,j} \)
\( br^r = s^r \times \text{dist}_{i,j} \)
\( bm^r = \sum_{k=1}^{\delta} m^{i+k} \times s^m \times \text{dist}_{i,o} \)

if \( i \) does not have a replica of content \( c \) then

if \( b^r > (br^r + bm^r) \) then

Replicate content \( c \) from the closest replica

end if

end if

end if

end if

end if

end for

while \( \text{load}_i \geq HW \) and content \( c \) has not been examined do

max\( \bar{y}^r = \max(d^r) \)

\( b^r = d^r \times s^r \times \text{dist}_{i,j} \)
\( bm^r = \sum_{k=1}^{\delta} m^{i+k} \times s^m \times \text{dist}_{i,o} \)

if \( d^r > bm^r \) then

Replicate content \( c \) on server \( j \)

end if

end while

end for
the server creates new content replicas on those servers that have the major contribution to its predicted future indirect load in order to alternate its future load.

Our new online heuristic algorithm is shown in Algorithm 1. Note that it takes a different approach from the previously proposed ACDN dynamic content placement algorithm [16]. In particular the ACDN algorithm makes placement decisions based on the demand observed in the past whereas our algorithm uses prediction of future demands, which, in turn, are based on the evolution of past demand. Moreover, the ACDN algorithm uses a “push” model, that is, at each reconfiguration period, the servers that have content replicas decide whether they should push a copy of the content to other servers where high demands were observed in the past. In contrast, our algorithm follows a “pull” model, i.e., each server decides to add a local replica by requesting it from the closest server that currently has it. A “push” strategy is used only when a server detects it may become overloaded in the following periods. Nevertheless, both algorithms have the same worst-case time complexity, i.e., $O(\log |S| \times |C|)$, where $|S|$ and $|C|$ are the numbers of servers and contents, respectively. In Section 5, the efficiency of both algorithms are evaluated, contrasting the quality of their solutions to the optimal offline, obtained by solving the optimization model described in the section 3.2.

4. SIMULATION ENVIRONMENT

The evaluation of new MD CDN, offline optimal and heuristic solutions and the previous static and ACDN algorithms is performed using a mobility simulation developed in [13]. It models a twenty-kilometer radial city, divided in area zones based on population density and natural limits (e.g., rivers, highways, etc). The area zones are grouped into four area types: center urban, suburban and rural. The simulator also include a number of content servers with fixed locations. Figure 1 shows a representation of the modeled city with the server locations indicated by small circles.

Different types of movement attraction points, i.e., places where people typically spend time in the city (e.g., residences, workplaces and schools). Figure 2 presents the frequencies of different types of movement attraction points in each city area type.

The user population is divided into four groups according to their mobility and demand characteristics. The four types of groups are: a) 5% of users are 24-hour delivery boys, b) 60% are common workers, c) 30% are housekeepers and d) 5% are taxi drivers. A movement table is associated to each group in order to determine its typical mobility behavior [15]. This was done by dividing the day into time periods and associating a probability that a user of the group is at a certain location at any time within the period.

During simulation, the per-user call inter-arrival times follow Poisson distribution with means 14, 7, 18 and 18 minutes for groups a), b), c) and d), respectively. Once connected, a user issues a number of requests at rate 1 request per second. The number of requests issued during a call depends on its duration, which is also exponentially distributed with parameter $\mu$. In particular the ACDN algorithm makes placement decisions based on the demand observed in the past whereas our algorithm uses prediction of future demands, which, in turn, are based on the evolution of past demand. Moreover, the ACDN algorithm uses a “push” model, that is, at each reconfiguration period, the servers that have content replicas decide whether they should push a copy of the content to other servers where high demands were observed in the past. In contrast, our algorithm follows a “pull” model, i.e., each server decides to add a local replica by requesting it from the closest server that currently has it. A “push” strategy is used only when a server detects it may become overloaded in the following periods. Nevertheless, both algorithms have the same worst-case time complexity, i.e., $O(\log |S| \times |C|)$, where $|S|$ and $|C|$ are the numbers of servers and contents, respectively. In Section 5, the efficiency of both algorithms are evaluated, contrasting the quality of their solutions to the optimal offline, obtained by solving the optimization model described in the section 3.2.

5. PERFORMANCE EVALUATION

This section evaluates the performance of the new online content placement heuristic for mobile networks. We compare the total traffic the online method generates over the backbone with the offline optimal traffic. We also compare our new algorithm with the previously proposed ACDN algorithm [16] and also with a static content placement approach, where there are multiple fixed replica servers for each content. In our experiments, we consider static content placement with 2 and 4 replica servers, placed at the more central locations in the simulated city (see Figure 1). A validation of our ACDN algorithm implementation is given in an extended version of this paper [1].

All contents have homogeneous sizes and have the same origin server. The geographical distance between two servers is used as a multiplier factor in the calculation of the traffic between them. We assume the worst-case scenario in which maintenance operations must be performed for every replica at each reconfiguration period.

Unless otherwise stated, in all experiments, we set the number of contents to 24 and the number of mobile users to 500 thousand, resulting in an average of 35700 requests per second in the network (see Section 4). The reconfiguration period is set to 10 minutes and the simulation runs for 80 periods. The $\alpha$ and $\delta$ parameters of the demand forecasting method are set to 0.2 and 7, respectively, and the HW parameter is set to 1000 requests per period.

5.1 Impact of Replication, Indirect Service and Maintenance Content Sizes

Recall that each content is characterized by three size parameters representing the number of bytes transmitted over the backbone during a replication ($sr^c$), indirect service ($sr^c$) and replica maintenance operation ($sm^c$). In this section, we refer to them as simply the replication, indirect service and maintenance sizes of a content. These parameters have direct impact on a server’s decision as to whether replicate/remove a content or forward a client request to another server. In the following we analyze the impact of these parameters in the five algorithms considered, offline optimal, online heuristic, ACDN and static placement with two and four servers, assuming a baseline configuration where $sr^c = sm^c = 1Kbyte$, for all $c \in C$.

We start by analyzing the impact of the replication size ($sr^c$), keeping the indirect service and maintenance sizes at the baseline. The total traffic generated all five algorithms as a function of $sr^c$ is shown in Figure 3. As $sr^c$ increases, the
replication component of the total traffic starts to dominate. This has no impact on the static approaches but the dynamic algorithms tend to reduce the number of replications. The offline optimal solution takes advantage of the knowledge of all future demands to perform replications that will lead to significant reductions in the future traffic, especially given the low maintenance size. The online algorithm, on the other hand, predicts client demands only for the following δ periods and thus makes some sub-optimal replication decisions. Nevertheless, note that, in the configurations shown in Figure 3, our new online algorithm results in a total traffic that is only up to slightly more than two times and the offline optimal (up to 85% lower) is significantly lower than the one generated by the ACDN algorithm. These results seem to indicate that replicating based on demand forecast instead of past observed demands leads to significant improvements. Finally, as expected, the savings in total traffic obtained with our online algorithm over the best static approach analyzed is as high as 97%.

![Figure 3: Total Traffic as a Function of Replication Size](image1)

![Figure 4: Total Traffic as a Function of Maintenance Size](image2)

Starting with a replication size equal to 10000 times the baseline, we now evaluate the impact of increasing the content maintenance size on the total network traffic generated by each algorithm. Figure 4 shows the results. As maintenance size increases, it becomes less cost-effective to maintain a replica in a server with low client demand. In this scenario, total network traffic for the dynamic algorithms increases significantly. In particular, the ACDN algorithm performs very poorly, even compared to the static baseline, we now evaluate the impact of increasing the configuration parameter equal to 10 Kbytes. Figure 6 shows the results. The dynamic algorithms scale much better than the static content placement approaches, as expected. For a system with 200 thousand users, the online algorithm results in a total traffic which is only 52% higher than the offline optimal. In contrast, for the same configuration, the ACDN algorithm results in a total traffic which is 126% higher than the optimal. These results show that our online algorithm is superior to both the static approach and the ACDN algorithm for a large range of number of users.

![Figure 5: Total Traffic as a Function of Indirect Service Size](image3)

![Figure 6: Total Traffic as a Function of the Number of Users](image4)

**5.2 Impact of the Number of Mobile Users**

This section, analyzes the total network traffic generated for each algorithm as the number of mobile users increases, assuming all size parameter equal to 10 Kbytes. Figure 6 shows the results. The dynamic algorithms scale much better than the static content placement approaches, as expected. For a system with 200 thousand users, the online algorithm results in a total traffic which is only 52% higher than the offline optimal. In contrast, for the same configuration, the ACDN algorithm results in a total traffic which is 126% higher than the optimal. These results show that our online algorithm is superior to both the static approach and the ACDN algorithm for a large range of number of users.

**5.3 Overhead of Management Operations**

Content replication and maintenance are management operations which incurs an overhead to the total traffic generated in a CDN. If this overhead is compensated with a significant reduction in the traffic of indirect client responses over the backbone, the management operations are beneficial to the system. Otherwise, total traffic increases and dynamic content placement may perform worst than static placement. Figure 7 shows the portion of total network traffic that is due to management operations, for a configuration of 300000 mobile units and the same size parameters used in Section 5.2. The dynamic placement approaches incur significant management overhead. In particular, practically all the traffic generated by the offline optimal solution is due to management operations, implying that the replication decisions are payed off by the reductions in indirect request service. The online algorithm generates the same management overhead. However, some replication decisions are not payed and the total traffic that is 37% higher than the offline optimal. In contrast, the ACDN algorithm results in higher management overhead and total traffic. Again, by using demand forecasting, instead of simply the past observed demand, our online algorithm avoids performing some replications that do not lead to traffic savings.

**5.4 Server Overloading**

Recall that our online algorithm imposes a soft upper-bound on the number of indirect requests each server can attend in any period, performing, thus, load balancing among multiple replicas of the same content. A similar approach is also used in the ACDN algorithm [16]. The upper-bound is soft because it is applied to estimates of future (indirect)
load. If the estimate is not accurate, a server may still become overloaded.

This section evaluates the ACDN and our online algorithms with respect to the number of overloaded servers during each simulated period. Figure 8 shows the results for the same configuration parameters used in section 5.3. The number of overloaded servers is significantly lower in the online algorithm in most simulation periods. The highly accurate demand forecasting method makes our algorithm much more competitive than the ACDN method not only for network traffic reduction but also for controlling server overload.

![Figure 7: Management Operations Overhead](image)

![Figure 8: Number of Servers Overloaded](image)

### 5.5 Impact of Demand Forecasting

The previous sections showed the performance gains of our online heuristic over the ACDN algorithm which is probably due to the more accurate future demand estimates. The accuracy of the demand forecasting method used is then key to the performance of our heuristic. This section evaluates the sensitivity of our algorithm to the \( \alpha \) and \( \delta \) parameters used in the forecasting method. We also analyze the accuracy of the forecasting method by comparing the total traffic generated with the online algorithm operating in two scenarios: (1) using the Double Exponential Smoothing (DES) method to predict future demand and (2) using perfect knowledge of the future demand. In both cases, we use the same window in the future (i.e., same \( \delta \) value).

Recall that the \( \alpha \) parameter is the weight given to recent observations in the prediction of future demand. The \( \alpha \) value that produces the best results depends on the typical variations in demand observed. If client request arrivals are fairly stable, an \( \alpha \) value close to 0 may produce better results. Values of \( \alpha \) close to 1 are more adequate if demand increases/decreases gradually. Figure 9 shows the increase in the total traffic generated, compared to the offline optimal, by the online algorithm using different values of \( \alpha \) and a configuration with 300000 users and indirect service, replication and maintenance sizes equal to 20KB, 10MB and 500KB, respectively. In the mobile environment simulated, client demand varies smoothly. Thus, an \( \alpha \) value close to 0 produces the best results. However, it is important to note that the method may perform poorly if the value of \( \alpha \) is not chosen appropriately (e.g., 33% traffic increase if \( \alpha = 0.9 \)).

![Figure 9: Impact of the DSE \( \alpha \) Parameter](image)

![Figure 10: Impact of the DSE \( \delta \) Parameter](image)

We now evaluate increase in total traffic over the optimal as the \( \delta \) parameter increases, for the same configuration as above. As shown in Figure 10 the online algorithm performs poorly if \( \delta = 1 \). As the value of \( \delta \) increases, the algorithm looks farther in the future and its performance improves. However, in our mobility simulation environment, most performance gains are obtained by increasing \( \delta \) to 2, further increases in \( \delta \) produces only minor improvements. Nevertheless, our online algorithm was very insensitive to the value of \( \delta \) for \( 2 \leq \delta \leq 20 \), producing results that are within 5% of the optimal.

Finally, we evaluate the accuracy of the DES forecasting method by comparing it with perfect forecasting, where the real client demands for the future \( \delta \) periods are known a priori. We compare both methods using a configuration with 12900 users and replication, indirect service and maintenance sizes equal to 2 MB, 20 KB and 900 KB, respectively. Figure 11 shows the total network traffic in each simulation period obtained with both methods. For comparison, it also shows the offline optimal results. Perfect demand forecasting reduces total traffic in only up to 9.7% compared to the DES method. Thus, the forecasting method used in our algorithm is highly accurate. Note that as simulation progresses, the online algorithm approximates to the offline optimal. The gap between them reduces to only 55% after 150 periods.

### 6. ACKNOWLEDGMENTS

Wagner M. Aioffi, Geraldo R. Mateus and Jussara M. Almeida are supported by CNPq/Brazil.

### 7. CONCLUSIONS AND FUTURE WORK

In this paper we applied content distribution techniques to build a Mobile Dynamic Content Distribution Network Model (MDCDN) that supports the increasing and variable demand of mobile networks. The MDCDN model takes both
temporal and spatial variations of client demand into account to dynamically add/remove content replicas to/from the network in order to minimize the total traffic over the backbone. It also imposes a soft upper-bound on the load on each server and performs load balancing to improve service provided to users. We develop two solutions for MDCDN: an offline optimal and an online heuristic, which is based on demand forecasting.

Using simulation, we compare our MDCDN solution to the previous ACDN algorithm [16] and to static content placement approaches. The performance evaluation showed that our new online algorithm is not only superior to the static approaches but also outperforms the ACDN algorithm in all cases considered, reducing total network traffic in up to 85%. Furthermore, our online algorithm produces solutions that are, in the worst-case, within 2.7 times the offline optimal. Server load control is also more effective in our online heuristic than in the ACDN algorithm, due to the highly accurate demand forecasting method used.

A possible direction for future work is to extend the performance comparison of our algorithms and the ACDN algorithm to theworst-case, within 2.7 times the offline optimal. 

## 8. REFERENCES


