Adaptive Caching with Heterogeneous Devices in Mobile Peer to Peer Network

Fan Ye
Department of Computer Science and Technology,
University of Science & Technology of China, Hefei, China.
Joint Research Lab of Excellence,
CityU-USTC Advanced Research Institute, Suzhou, China.
Department of Computer Science and Technology,
City University of Hong Kong, Hong Kong, China.
yfan@mail.ustc.edu.cn

Qing Li
Department of Computer Science,
City University of Hong Kong,
Hong Kong, China.
Joint Research Lab of Excellence,
CityU-USTC Advanced Research Institute, Suzhou, China.
itqli@cityu.edu.hk

EnHong Chen
Department of Computer Science and Technology,
University of Science & Technology of China, Hefei, China.
Joint Research Lab of Excellence,
CityU-USTC Advanced Research Institute, Suzhou, China.
cheneh@ustc.edu.cn

ABSTRACT
In the (upcoming) 3G age, many applications such as multimedia streaming, file sharing and so on will be widely used in the wireless environments. In such a mobile environment, users may carry heterogeneous mobile devices with different transmission ranges, latency, and even cache sizes. These devices not only can get services from a Mobile Support Station (MSS), but also they can form/generate a mobile peer-to-peer (P2P) network to provide services to each other. For such a mobile P2P network, an effective cache framework that can handle heterogeneous devices is required. In this paper, we propose a flexible cache scheme which is adaptive to the actual device condition and that of its neighbors. Our scheme encourages the "strong" peers to keep hot data and do service, and meanwhile to protect and best utilize the "weak" nodes with their limited cache space. Simulation and mathematic analysis results show the effectiveness of our scheme.

Categories and Subject Descriptors
Algorithms, Design.

General Terms
Distributed Cooperative Caching Scheme, Heterogeneous Mobile Device, Mobile Peer to Peer Network.

Keywords
Heterogeneous Devices, Mobile Peer to Peer Network, Caching.

1. INTRODUCTION
The popularity of wireless networks grows with the advance of wireless technologies and Internet applications. In the (upcoming) 3G age, many applications such as multimedia streaming, file sharing and so on will be widely used in the wireless environment.

SAC'08, March 16-20, 2008, Fortaleza, Ceará, Brazil.

On the other hand, Peer-to-Peer (P2P) systems have now been widely used in lots of applications on the Internet. Structured P2P systems provide both low latency and desired load balance so that all data items are accessed with a uniform probability, which has been proved to be very effective.

However, the distribution of user demands for real data items is often skewed, and the mobile devices may have different capabilities. These can lead to poor load balancing and dropped messages. Furthermore, the situation becomes even more severe in a mobile P2P environment. In particular, user devices are often different in terms of their capabilities such as the transmission scope, power of battery, cache size, and so on. In this paper, we propose a distributed adaptive cache scheme based on the capabilities of mobile devices in a heterogeneous mobile device network. We further suggest an AFSR (for *Adaptive Frequency, Size, Recentness*) scheme, and conduct analytical studies to compare with the existing schemes of *Cooperative Uniform Frequency, Size, Recentness* (CUFSR) [7] and the popular *Distributed Least Recently Used* (DLRU) [9]. The CUFSR algorithm considers the ratios *Frequency*, *Size*, and *Recentness* and nodes can get data from neighbors' caching. The DLRU scheme use LRU(*Least Recent Used*) to update and nodes also can share data in a cooperative way. Simulation studies are conducted to evaluate the effectiveness of our flexible cache scheme.

The rest of the paper is organized as follows. Section 2 surveys some related work of caching schemes in mobile environments. Our modeling framework is explained in section 3. Section 4 describes the adaptive cache algorithm and the mathematic model. In section 5, simulation results are presented and the performance is evaluated. Finally, Section 6 offers brief concluding remarks, with an outline of our future work.

2. RELATED WORK
There have been a lot of cooperative cache schemes proposed with analogical peers, such as ad hoc network caching, Web caching, and so on. Numerous studies have been carried out on the caching and replication techniques in mobile environments.

However, previous works were mostly based on the ground that the mobile nodes in the network have the same ability, and the data transmission latency is uniform. In [2], a cooperative caching framework called "Hybrid caching" is introduced and claimed to be very effective to data availability; the caching framework is based on "CacheData" and "CachePath" in an ad hoc network. In [3,4], a replica allocation method and clustering in ad hoc networks are introduced to improve data accessibility in a mobile communication environment. Another cooperative cache scheme which is named as COCA (mobile COoperative CAching framework) for similar peers is described in [5], which combines the P2P communication technology with a conventional mobile system. However, as a mobile P2P network is formed by the mobile users with different kinds of devices, and the battery power of the same kind of devices may also be different, mobile devices in the network are heterogeneous and have different capabilities.

To address these challenges, we advocate an adaptive caching framework in which heterogonous devices with different transmission ranges, latency, and even cache sizes co-exist in the mobile P2P network, as to be detailed in the subsequent sections.

3. MODELING FRAMEWORK

Borrowing the diagram from [5], the modeling framework for our cache scheme is identical to that of [5], as shows in Figure 1. In particular, we assume that each mobile host, m_i , has a unique identifier, i.e., M= {m_1, m_2,..., $m_{NumClient}$}, where NumClient is the total number of mobile devices in the environment. Each mobile host (device) is equipped with two wireless network interface cards: one is dedicated to communications with MSS (Mobile Support Station), and the other is devoted to communication with the neighbor peers to form the P2P network. A mobile device can directly communicate with the devices which are in its transmission range; however, by using data indexing, the device can also communicate with other devices out of its transmission range through the help of other mobile hosts, with a longer network latency in most cases.

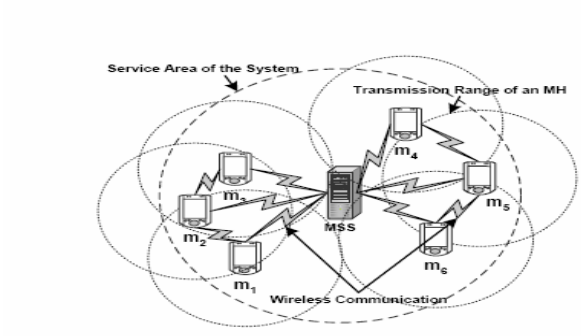

Figure 1. The Mobile P2P Environment

3.1 Communication protocol

For each data request, a mobile host m_i first tries to find the required data item from its local cache. If it encounters a local cache miss, it broadcasts a request message to its neighbor peers via P2P broadcast communication. If a neighbor peer has the required data item cached, it sends a reply message to m_i via P2P communication. However, if m_i does not get any reply from the network after the time out period, it will instead send the request to the MSS to obtain the required data.

3.2 Neighbor discovery and information exchange scheme

With all the mobile hosts (devices) having two network interfaces, they can use one of these to communicate with their neighbors. In our scheme, just as the broadcast environment, a mobile host periodically sends a "hello" message to the mobile devices in its transmission range. If a peer in the P2P network can get data files from other peers directly or through some relaying peers, then all of these peers will be regarded as neighbors, with the latency being defined as follows:

$$\sum_{i=1}^{hop} (\text{Size of Request}_i + \text{size of Reply}_i)/BW_{i,i+1} \quad (1)$$

Here, *hop* is defined as the total number of hops the data gets transmitted, and $BW_{i,i+1}$ denotes the bandwidth between two peers (mobile devices) whose Ids are i and i+1, respectively. As peers can exchange such information as the cache sizes of their neighbors, and the transmission ranges, they can negotiate and come up with a cache strategy based on the messages exchanged. As a result, better usage of their limited cache space and higher system performance can be obtained, as to be described next.

4. ADAPTIVE CACHING ALGORITHM

4.1 Preliminary

In our scheme, an important attribute for each data (text, multimedia) object is its popularity: the larger the value is, the more possible that mobile clients may access that object.

Formula (2) defines the relationship between the popularity of a data object and its ranking, which can be denoted as Zipf-low, where α is assumed to be around 0.5(or more precisely, 0.47).

$$popular = \frac{1}{rank^{\alpha}} (0 < rank < TotalObject) \quad (2)$$

We use the FSR scheme proposed in [7] as the basic cache updating algorithm for each mobile host, in which the weight W of a cached data object C_i is defined as:

$$W = F^f S^s R^r \quad (3)$$

In formula (3), F is the number of times the cached data object C_i is accessed (i.e., the Frequency), S is the size of the object (Size) and R is the counter of the request times since the last access to C_i (i.e., the Recentness). Each new request round will cause all the cached data object 's Recentness to be incremented by one; if the cached object has been actually accessed, its Recentness is reset to zero. The three exponents f, r and s (as the weights for the Frequency, Size, and Recentness, respectively) in formula (2) are chosen by trial-and-error, and in our experimental study we adopt f=2, r=0.2, and s=-1.5 empirically. To effectively use the cache, it is obvious that we should favor the replacement of a cached object (block) with a larger Recentness, as it indicates that the object may have been sitting in the cache and not accessed for a long time. Also we should choose cache blocks with small F (Frequency) and large S (Size) values as the

candidates for replacement, so that it may allow us to accommodate more "hot" yet small-sized objects in the available cache space.

Table 1 lists the various parameters and their definitions (default values) to be used by our cache scheme.

Table 1. The parameters of our scheme

Notation	Definition(Default values)
Latency	time consumed to transfer a particular data object
Cache Size	10% -30% of the total storage space used
Frequency (F)	access frequency to a cached data object C_i
Size (S)	the size of a cached data object C_i
Recentness (R)	a counter of the request times since last access to C_i
f	the weight of F
s	the weight of S
r	the weight of R
W	the weight of a cache object C_i
Zipf Factor	0.47
Total Number of Data Files	10-500
Mean Request Interval	12.5s
Simulation Time	500-2000s
(Average) File Size	uniform distribution in 1KB – 10KB
Total No. of Mobile Nodes	10-100

4.2 Algorithmic descriptions

In comparison with the existing cooperative cache schemes, one main difference of our cache scheme lies in the fact that the mobile devices in our P2P network can be heterogeneous in terms of their transmission ranges, latency, and cache sizes. As a first attempt, we consider the properties of latency and cache size as the basis to cluster the mobile peers/devices. In particular, mobile devices are divided into one group with *Low Latency* (LL(<500ms)) and *Big Cache* (BC(>=100KB)) size, another group with *High Latency* (HL(>=500ms)) and *Small Cache* (SC(<100KB)) size, and the rest into the third group. As listed in Table 1, *Latency* is the transmission delay of the link between two mobile devices, and *Cache Size* is the total cache space of a mobile device. Using these attributes, we can classify the mobile peers into "strong" peers (in the first group) , "weak" peers (in the second group), and the "normal" peers.

In Figure 2, the algorithm checks if a data request can be satisfied locally, from its neighbors, or through the Mobile Support Station (MSS). In line 2, the inLocalCache(request r) sub-route checks if the requested data object is in the local cache; similarly, the inNeighborCache(request r) sub-route in line 4 is to check if the requested data object is available from any of the other peers which have become the neighbors of the current peer.

```
Algorithm_Service_Satisfaction_Check
1. for a data request r
2. if ( inLocalCache(r) == True), then send the requested data to the user
3. else search r from the Neighbors to get the data, and update the local
     cache
4. else if ( !inNeighborCache(r) )
5. then send the request to  MSS to get the required data object;
6. update the local cache
7. End if
8. End for
```

Figure 2. Service Satisfaction Checking Algorithm

The overall algorithm to adaptively conduct cache update is given in Figure 3, with the "greedy" update algorithm (line 2) and the "conservative" update algorithm (line 4) being given in Figures 4 and 5, respectively.

```
Algorithm_Cache_Update(Latency latency, CacheSize cs, Request r)
1. If (latency<LL && cs > BC)
2. update the cache using the "Greedy" updating algorithm
3. else if (latency >HL && cs < SC)
4. update the cache using the "Conservative" updating algorithm
5. else update the cache using the basic FSR (or CUFSR) described in [7]
6. End if
```

Figure 3. Adaptive Cache Update Scheme

```
Algorithm_Greedy_CacheUpdate (file f, Cache List cl)
1. for (every object i in the Cache List cl)
2. C_i.W = (C_i.Frequency)^f / ((C_i.Size)^s * (C_i.Recentness)^r)
3. End for;
4. minW =  least Weight of the cache blocks in Cache List;
5. If (minW < δ )
6. update the Cache List with the least Ci.W in file object f ;
7. else keep the Cache List as before
8. end if
```

Figure 4. Greedy Cache Updating Scheme

```
Algorithm_Conservative_CacheUpdate(Cache List cl,file f)
1.    for(every object o in Cache List cl)
2.    if(  inNeighborCache(f) == True ), do not update cache
3.    else  for(every cache block Ci of local cache)
4.        if(inNeighborCache(Ci)==True),then updateQuene.push(Ci);
5.        for(every cache block in updateQueue)
6.          update the cache block with the least Weight W in the
      Queue
7.        end for
8.    end for
9.    end if
10.   end for
```

Figure 5. Conservative Cache Updating Scheme

In Figure 4, C_i means the i_{th} data object (block) of the Cache List. In the "Greedy" cache update, the algorithm attempts to keep most of the hot objects in the local cache of the mobile device. In particular, we set a threshold δ , so that if the cache block's weight is bigger than the threshold, it will not be updated. Furthermore, the data replication method can be used to keep the popular/hot data objects in the "strong" peers. On the other hand, in the "Conservative" updating algorithm shown in Figure 5, local cache is used to keep those objects which may not be the hot ones but are likely to be used. As a matter of fact, the approach is "conservative" in that all the nodes do their work by considering

their "stronger" neighbor's condition and then do their cache update accordingly. Note that the list "updateQueue" keeps the cache blocks which have been found in their strong neighbor's cache, therefore it could be updated (replaced). The basic FSR (or termed as CUFSR) cache updating scheme (line 5 of Figure 3) refers to the algorithm proposed in [7], which replaces the cache object (data block) of the least W (i.e., the weight calculated by formula(3) using the Frequency, Size, and Recentness). In our conservative scheme, a "weak" node is thus able to keep data objects which do not appear in the strong neighbors (nodes). As the third possibility (when the node is neither strong nor weak), the local cache is updated according to the basic FSR (or CUFSR) cache updating scheme (line 5 of Figure 3) as proposed in [7], namely, it replaces the cache object (data block) which is of the least W.

4.3 Mathematic model

In this section, we give a mathematic analysis of our model. We suppose that there are total of R data requests for the P2P network. In the system, there are totally N_s strong nodes, N_w weak nodes and N_{ser} server nodes. (A server node is an MSS which can satisfy the request of data objects not found in the local cache; generally, it is much more time-consuming to get the requested data from an MSS.) The latency for a strong node i is defined as l_{si}, the latency for a weak node i is defined as l_{wj}, while the latency for an MSS is defined as l_{ser}. For simplicity of our analysis, we suppose that the total cache space of the peers is bigger than the total file size, and the numbers of the three kinds of peers (strong peers, weak pears, server nodes/MSSs) are the same. Furthermore, we use P_{si} to denote the proportion of requests made by a strong node i relative to the total requests, P_{wj} as the proportion of requests made by a weak node j relative to the total requests, and $P_{ser.k}$ as the proportion of requests made to the k^{th} server/MSS relative to the total requests. (Note that N_{ser} is the number of the servers/MSSs in the whole system.) Then the total latency can be calculated as:

$$\sum_{i=1}^{N_s} l_{si} R * p_{si} + \sum_{j=1}^{N_w} l_{wj} R * p_{wj} + \sum_{k=1}^{N_{ser}} l_{ser.k} R * p_{ser.k} =$$

$$R \sum_{i=1}^{N_s} (l_{si} p_{si} + l_{wi} p_{wi} + l_{ser.i} p_{ser.i}) =$$

Suppose that $l_{wi} = t_i * l_{si}$ and $l_{ser} = t^a_i * l_{si}$, we then have:

$$R \sum_{i=1}^{N_s} (l_{si} p_{si} + p_{wi} * t_i * l_{si} + p_{ser.i} * t_i^{\alpha} * l_{si}) =$$

$$R \sum_{i=1}^{N_s} l_{si} (p_{si} + p_{wi} * t_i + p_{ser.i} * t_i^{\alpha}) =$$

For simplicity and clarity, we further suppose that all the strong nodes are not too different from each other, all the weak nodes are pretty much the same, and all the MSSs can be regarded as similar with respect to their latency. Thus we have $t_i = t$ and $l_{si} = l_s$. Substituting these into the above formula, we get the total latency as follows:

$$R l_s \sum_{i=1}^{N_s} (p_{si} + p_{wi} * t + p_{ser.i} * t^{\alpha}) \qquad (4)$$

In formula (4) above, the parameters t denotes the relative latency of a weak node in comparison with a strong node, and t^{α} means the relative latency of a server/MSS in comparison with a strong node. Obviously $t > 1$ and $\alpha > 1$. As with AFSR, we keep all kinds of data in the network, much smaller $P_{ser.i}$ can be obtained, for weak node need not to keep hot data, much smaller P_{wi} can be obtained in weak node. So, P_{wi} and $P_{ser.i}$ in our scheme is smaller. It follows that t^{α} is the main part of the result, and t is the second main part of the result (cf. formula (4)); as a result, our scheme still can do better in terms of the Average Request Delay (ARD), as to be verified by the empirical study next.

5. EMPIRICAL STUDY

To demonstrate the effectiveness of our proposed adaptive cache scheme, we have conducted an empirical study through simulation. The simulation test bed is designed on a 500m*500m rectangle space and the three kinds of the peers are very much of the same number. The peers are using Random Waypoint Model (or, RWM) which is a commonly used synthetic model for mobility in, e.g., ad hoc networks. It is an elementary model which describes the movement pattern of independent nodes by simple terms. In our simulation, each file object is randomly assigned with a value representing its popularity, and mobile clients choose file objects randomly based on such values. Our simulation is implemented upon JNS (java network simulator) which has quite the same characteristics as NS-2, except that it is a Java based network simulator. In the simulation study, we compare our scheme with such existing traditional algorithms as CUFSR (*Cooperative Uniform Frequency, Size, and Recentness* based algorithm) and DLRU (*Distributed Least Recently Used*), the later is especially popular in cooperative web caching. For comparison purpose, we term our scheme as AFSR (*Adaptive Frequency Size Recentness*) in the following discussions.

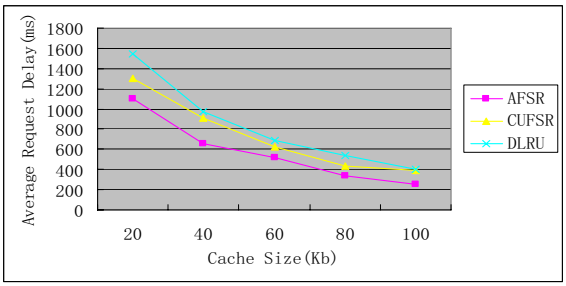

Figure 7. Average Request Delay (ARD) against Cache Size

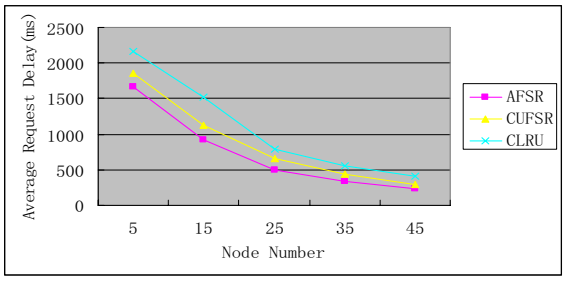

Figure 8. Average Request Delay (ARD) against Node Number

Figure 7 shows the simulation result upon cache size by using 20 peers. We use the Average Request Delay (ARD) as the determine ratio to compare these three algorithms. As can be seen, our AFSR performs much better than the basic uniform cache algorithm CUFSR and the DLRU scheme. As the average cache size becomes larger, the ARD value becomes smaller for all three algorithms, since more cache space makes the system to respond faster (i.e., with a smaller time delay).

In Figure 8, we evaluate the algorithms in terms of the node number. As can be seen, the increase of the node number can lead to the decrease of ARD value. This is natural since more nodes means more space to cache data for all the peers in the network. Notably, our AFSR algorithm does better than CUFSR in all cases, while the latter is already better than DLRU.

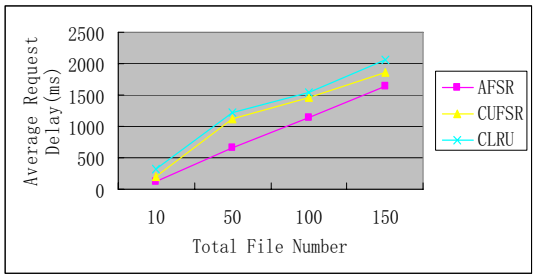

Figure 9. Average Request Delay (ARD) against File Number

To test the robustness of our cache scheme, Figure 9 shows the ARD result of the three algorithms in terms of the file number. In particular, different numbers of files (referred as the total number of files in Table 1) that users can obtain are tested to examine the performance of the three schemes. In this test, we use 40 peers whose average cache size is 40KB, with the cache sizes of the peers uniformly distributed in the range of 10KB-60KB. As we see in Figure 9, ARD becomes bigger as the file number increases. The reason is that as file number gets larger; cache miss also becomes bigger, thus causing the peer to take more time to get the required data file. However, ARD values of our AFSR algorithm are still much smaller than that of CUFSR and DLRU, indicating that the scalability of our scheme.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a flexible cache scheme which is adaptive to a mobile P2P network with heterogeneous devices. A "Greedy" scheme is devised to encourage the "strong" peers to provide more caching services to other peers, and a "Conservative" scheme is devised for the "weak" nodes, which enables them to be an effective complement to the "strong" nodes; for the rest peers, a normal scheme based on the cooperative uniform FSR[7] is used for their cache update. Mathematic analysis and simulation results also confirm and demonstrate the validity and effectiveness of our scheme. Our future work will concentrate on how to generate a more sophisticated model for heterogeneous mobile devices by considering more ratios. In addition, we will also consider combining caching and data replication techniques within a mobile network of heterogeneous devices.

ACKNOWLEDGMENTS

The work described here is supported by the National Basic Research Fund of China ("973" Program) under Grant No.2003CB317006 and the National Natural Science Foundation of China under Grant No.60573077. The research has been benefited from various discussions among the group members of the Joint Research Lab between CityU (Hong Kong) and USTC (China) in their advanced research institute in Suzhou, China.

7. REFERENCES

[1] S. Hadijiefthymiades, L. Merakos, "Using Proxy Cache Relocation to Accelerate Web Browsing in Wireless/Mobile Communications", www10, May 1-5, 2001, Hong Kong.

[2] L. Yin, G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks", IEEE TRANSACTION ON MOBILE COMPUTING. VOL 5, NO. 1, JANUARY 2006.

[3] T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility", IEEE INFOCOM 2001.

[4] J. Zheng, J. Su, X. Lu, "A Clustering-Based Data Replication Algorithm in Mobile Ad Hoc Networks for Improving Data Availability", Journal of Software, Vol.16, No.8.

[5] C. Y. Chow, H. V. Leong, A. T. S. Chan. "Distributed Group-based Cooperative Caching in a Mobile Broadcast Environment", MDM 2005, Ayia Napa,Cyprus.

[6] C.-Y. Chow, H. V. Leong , and A. T. S. Chan. Utilizing the cache space of low-activity clients in a mobile cooperative caching environment. International Journal of Wireless and Mobile Computing (IJWMC).

[7] S. Paknikar, M. Kankanhalli, K.R. Ramakrishnan, S.H.Srinivasan, L.H. Ngoh, "A Caching and Streaming Framework for Multimedia", *ACM Multimedia 2000*, Los Angeles, USA.

[8] J. Zhai, Q. Li, and X. Li, "Data Caching in selfish MANETs", ICCNMC 2005.

[9] Wijesundara, M.N.; Tay, T.T. "An object replacement strategy for global performance in distributed Web caching", Communication Technology Proceedings, 2003. ICCT 2003.

[10] K. Wu, P. S. Yu, J. L. Wolf, "Segmentation of Multimedia Streams for Proxy Caching", IEEE TRANSACTIONS ON MULTIMEDIA. VOL. 6, NO. 5, OCTOBER 2004.

[11] P. Krishnan, D. Raz, Y. Shavitt, "The Cache Location Problem", IEEE/ACM TRANSACTIONS ON NETWORKING. VOL 8, NO. 5. OCTOBER 2000

[12] C.Ye, D.M.Chiu, "Peer-to-peer Replication with Preferences", Infoscale 2007, Suzhou,China, June 2007.