

Multimedia Resource Replication Strategy for a Pervasive Peer-to-Peer Environment

Letian Rong

Arles Group, INRIA, Rocquencourt, France

Email: Letian.Rong@inria.fr

Abstract— The computer world is experiencing a paradigm shift towards context-aware pervasive computing, where diverse computer devices communicate with each other through different network access technologies, and allow resource access based on user preferences. P2P computing is an area which will be affected by this new trend. In this paper, we proposed a resource replication strategy for context-aware P2P computing in a pervasive environment. The replication strategy takes an active approach through using the resource request rate as the primary metric to trigger the replication process, and then adaptively replicate resources into various configuration states according to the properties of peers (i.e., terminal capabilities and user preferences) and the size of peer clusters. Also, the strategy uses peer related information stored on super peers to determine which peers should be selected to perform adaptive replications and where the resulting replicas should be stored. Simulation results show that the proposed strategy reduces network delays while increasing resource hit rate in comparison to commercial super peer P2P systems and random replication strategies.

Index Terms—P2P, pervasive computing, context-aware, replication strategy

I. INTRODUCTION

P2P systems have been rapidly gaining popularity on the Internet, due to the wide success of Napster [1], KaZaA [7] and other multimedia resource sharing P2P systems. Currently, the multimedia resource sharing P2P systems are mainly designed for high-end computer devices such as PCs and notebooks. However, as computing devices becoming more diverse, soon mobile and embedded devices such as mobile phones and MP4 players will also participate as peers in P2P-based sharing. Therefore, it is important that the P2P systems consider providing multimedia resource sharing suitable for heterogeneous devices. There are also other factors which are not dependent on devices but could also be subject to vast variations. For example, the network connection bandwidth of peer devices could vary from 14.4kbps to 100Mbps, while users can also have different preferences about the consumed resources. In summary, a broad range of usage environment characteristics should be considered for multimedia resource consumption in a P2P environment. In particular, multimedia resources should be adapted to suit the usage environment of peer

devices and provide users with the best possible experience.

A super peer-based architecture was proposed in our previous work [2][3] to facilitate dynamic multimedia resource adaptation in a P2P environment based on the surrounding usage environment attributes (i.e., user preferences, terminal capabilities etc.). In this paper, we focus on the resource replication aspect of the proposed architecture. As the proposed architecture aims to provide a P2P infrastructure enabling multimedia access in a pervasive computing environment, the replication strategy needs to take the heterogeneity of peers into consideration when creating resource replicas. In addition, it should also use the properties of peers (i.e., terminal capabilities) and their surrounding usage environment attributes (i.e., network attributes) to determine which peers should be selected to perform adaptive replications and where the resulting replicas should be stored. This is an area that has not been explored, to our knowledge, in other work.

The paper is structured as follows: section II briefly describes the previously proposed dynamic resource adaptation architecture. Section III then compares the related work in the field of P2P-based resource replications, and Section IV outlines the shortcomings and challenges currently being faced. In section V, the proposed resource replication strategy is explained. Simulation results are shown in section VI with section VII drawing conclusions.

II. DYNAMIC RESOURCE ADAPTATION P2P ARCHITECTURE

P2P systems can be classified into two main types based on the level of 'structured-ness' of peers and their data in a P2P network. These are known as structured P2P systems (e.g., Tapstry [20] and CAN [21]) and unstructured P2P systems (e.g., Napster [1], Gnutella [23] and KaZaA [7]). Structured P2P systems are not well suited for mass-market resource sharing, due to their rigid structures, despite being more effective in finding resources [22]. On the other hand, all the widely deployed P2P systems on the Internet today are unstructured P2P systems (e.g., Gnutella [23] and KaZaA [7]) and their success has proven the feasibility of these systems. One subset of structured P2P systems is the super peer systems [24] (e.g., KaZaA [7] and eDonkey [13]) which

further improve the scalability of P2P-based sharing by nominating powerful peers in the network as super peers to handle search requests and other administrative duties.

We previously proposed a dynamic P2P architecture to support resource adaptation/personalization according to the surrounding usage environment and user preferences [2]. The proposed P2P architecture is based on the super peer design [24], peers are grouped into clusters and super peers are nominated to be in charge of one or more clusters. Two types of super peers are defined in the architecture: category super peers (C_SP) and locality super peers (L_SP). A C_SP or a number of C_SPs govern peers which share content that belongs to a particular category, while each L_SP is in charge of peers within a certain locality range. Metrics such as physical distance, IP address and RTT can be used as locality attributes and their feasibility and performance in our system is currently under investigation [3].

When new peers join the network, they must first register with a corresponding L_SP based on their locality attributes and then submit related usage environment attributes (i.e., user preferences) to that L_SP. Peers can then register information of their shared content with one or more C_SPs; this includes submitting related usage environment attributes (i.e., terminal capabilities). The super peers then use pull or pushed adaptation approaches based on the registered information to adapt search results and select peers to send/adapt resources to the request peers [2].

In the following sections, first we review related work in resource replication strategies and then propose a replication strategy for unstructured super peer systems based on the proposed P2P architecture.

III. RELATED WORK IN RESOURCE REPLICATION STRATEGIES

P2P replication puts copies of objects closer to the peers to minimize the connection distance between the peers requesting and providing the objects. It helps to minimize the distance delay by sending requests to closely located peers. Also, replication assists with the disappearance of peers, as peers tend to be user machines rather than dedicated servers; therefore, there is no guarantee that the peers will not be randomly disconnected from the network. Coordinating the replication of resources is essential to a scalable P2P network.

P2P based replication strategy is the topic of various active research projects and most of these projects emphasize replication in structured P2P systems [4]-[6]. For unstructured P2P systems, the most common replication strategy is to allow the resource to replicate itself across the peers without coordination. This strategy is deployed in commercial P2P systems such as Napster [1] and KaZaA [7]. With this approach, a peer replicates an object only when it is downloaded to the local storage. However, this approach could easily lead to uneven distribution of replicas across the network, as well as a shortage or excess replication of resources. Excessive

replication can cause wastage of network and peer resources.

There are several research proposals that aim to improve the existing replication approaches in an unstructured P2P network [8]-[12]. Lv et al. [8] and Cohen and Shenker [9] study optimal replication in order to reduce random search times in an unstructured P2P network. Two common replication methods, Uniform and Proportional replication strategies were considered in the work, and a third replication strategy, Square-root replication was proposed to provide the optimal search size on successful resource request queries and minimize the overall search traffic. Tewari et al. [11] investigate the effect of the number of file replicas on search performance in unstructured P2P networks. It is shown that for a P2P network with a random graph topology where file replicas are uniformly distributed, the hop distance to a replica of a file is logarithmic in the number of replicas. Furthermore, it is shown that flooding-based search is optimized when the number of replicas is proportional to the file request rates. On et al. [10] propose a set of Quality of Service (QoS) based parameters and metrics to evaluate the performance of resource replication in a P2P network. They focus on P2P systems where peer storage is very limited and use replica placement as the main strategy to improve resource replications by satisfying the proposed QoS parameters. In [12], Thampi et al. propose to deploy mobile agents in unstructured P2P networks to replicate resources. The agents use machine learning techniques to determine suitable sites for replications.

Commercially available super-peer based P2P system such as KaZaA [7] and eDonkey [13] use super-peers in the network to replicate the index of several other peers. It is evident that searching is more effective when a larger index can be probed. However, even though per-probe capacity is increased in these systems, the same resource replication strategy as Gnutella is deployed for the actual replication of resources. Yang et al. [14] compare the performance of super peer systems with different replication and server organization schemes. Four different types of architectures are proposed based on server organization, query forwarding and object index replication: Chained Architecture, Full Replication Architecture, Hash Architecture and Unchained Architecture. However, the replication of the actual resources is not addressed.

IV. ANALYSIS AND REQUIREMENTS OF MULTIMEDIA RESOURCE REPLICATION IN A DYNAMIC RESOURCE ADAPTATION P2P ARCHITECTURE

When designing a replication strategy, the following important aspects should be taken into consideration:

1. What to replicate (replica selection problem)?
2. Where to place the replica (replica placement problem)?
3. When and how to update the replica (update control problem)?

Since the proposed architecture focuses on the consumption of multimedia resources in a public P2P

network, old resource would usually be replaced by new resource in a short timeframe. Therefore, updating the resource is not considered to be a crucial aspect. On the other hand, the first two aspects for resource replication (i.e., 1. what to replicate, and 2. where to place the replica) have a direct impact on the performance of the system, and thus should become the main focus of the proposed replication strategy.

In terms of ‘what to replicate’, distributed resources should be replicated according to the resource popularities in order to maximize the probability that requests from peers will be satisfied (i.e., hit rate of requests). This is particularly true for popular resources that might have a limited number of replicas in a P2P network when first introduced, and overwhelming requests from peers can cause network congestions and slow download speed. Furthermore, in the proposed dynamic resource adaptation P2P architecture, the replication strategy needs to consider the heterogeneity of a P2P environment where replications should be adaptively created in proportion to the percentage of various peer classes (i.e., desktop computer, mobile phone etc.) and based on their usage environment factors (i.e., terminal capabilities). For instance, a movie trailer should be replicated in various bit-rates/resolutions so that it can be played not only on desktop computers, but PDAs and mobile phones which all connect to the same P2P network. Appropriate resource variation replication can reduce the overhead introduced when adapting replicas into correct variation versions later on when the resource is requested.

In terms of ‘where to place the replica’, the replicated copies should be placed close in proximity to peers who are likely to request the resource. This allows peers to be able to search and find desired resources, and reduces delays occurring during search and downloading. Also, as the proposed dynamic resource adaptation P2P architecture is required to adapt replicas into various variations, the replication strategy should use the properties of peers (i.e., terminal capabilities) and their surrounding usage environment attributes (i.e., network attributes) to determine which peers should be selected to perform adaptive replications and where the resulting replicas should be stored.

Last, the mentioned projects take a passive replication approach which only triggers the replication process when a resource is downloaded by a peer and creates at most one resource replica. However, it is essential to provide a more assertive replication strategy to cope with the dynamic changes of resource popularity in a multimedia sharing P2P network. The following section outlines the design of a replication strategy based on these requirements.

V. ADAPTIVE RESOURCE REPLICATION STRATEGY

Based on the observations in Section IV, we propose a novel adaptive resource replication strategy that improves the resource availability and reduces network delays.

A. Resource Replicatoin based on Request Rate

In order to accommodate dynamic changes of resource popularity in a P2P network, a more assertive replication approach is taken in the replication strategy which uses request rate as the metric to determine when replication should be performed. It is shown in [15] that when the number of replicas of each file is proportional to the request rate for that file, each peer operates at a uniform utilization level independent of which files it has in its storage, regardless of the type of search method deployed. Thus, the use of request rate as the main metric to replicate resources should minimize the average download time in the P2P network since the resources of host peers are uniformly utilized. The request rate is calculated as the number of requests received by the C_SP for a particular resource divided by the time elapsed.

In the proposed replication strategy, each C_SP has a request rate table which contains entries that store both the total number of requests and request rates of resources that belong to the category; the table is updated every time when a C_SP receives a resource request. When the total number of requests for a resource reaches a certain predefined threshold (i.e., every 20 requests), it triggers the C_SP to check if resource replication for the particular resource is required. Presently, the following simple formula is used to determine the number of replicas ($Required_{Total}$) that are required globally in the network for a given request rate of a resource ($Request_Rate$):

$$Required_{Total} = Request_Rate * K$$

K is a numeric value that determines how aggressively the replication strategy should be undertaken. If the total number of current resource variations is smaller than $Required_{Total}$, then the super peer starts the adaptive resource replication process.

B. Steps of the Adaptive Resource Replicatoin Strategy

The replication process is divided into four steps:

Step1. Overall replica number estimation for each locality cluster: the C_SP estimates the number of resources that are required to be replicated in each locality cluster based on $Required_{Total}$ and the size of each locality cluster ($Size_{Cluster}$) in comparison to the total network size ($Size_{Total}$). For instance, the required number of resources in locality cluster i ($Required_{Cluster-i}$) is calculated as:

$$Required_{Cluster-i} = Required_{Total} * (Size_{Cluster} / Size_{Total})$$

The C_SP then requests each L_SP to replicate $Required_{Cluster}$ number of replicas.

Step2. Individual replica number estimation based on each resource variation: once an L_SP receives the order from the C_SP, it starts the second stage of the replication process by calculating the number of replicas that are required for each resource variation (i.e., resource with

different configuration state). As explained in Section II, every peer is required to register its related usage environment descriptors (i.e., terminal capabilities and user preferences) to a corresponding L_SP; therefore the L_SP uses the peer information to calculate the number of peers ($Size_{Peer-Variation}$) which fit a resource variation. For instance, a movie trailer with resolution of 320x240 pixels is suitable for peers running on PDAs. The required number of replicas for resource variation A ($Required_{Variation-A}$) in cluster i of size $Size_{Cluster-i}$ is calculated as:

$$Required_{Variation-A} = Required_{Cluster-i} * (Size_{Peer-Variation-A} / Size_{Cluster-i})$$

Step3. Allocate replica senders, adaptors and receivers: when $Required_{Variation}$ is determined for each resource variation in a locality cluster, the L_SP selects replica senders, adaptors and receivers respectively. Replica adaptor is defined as the peer who is able to adapt a resource to a different resource variation (i.e., transcoding of video streams). The replica senders, adaptors and receivers are selected based on their capability descriptors (i.e., bandwidth, processing power, storage capacity and availability etc.) which are stored on the L_SP. The selection process continues until $Required_{Cluster}$ is reached in the locality cluster. This selection process occurs in each locality cluster and the adaptive replication process is completed when all replicas are transmitted to the replica receivers in all locality clusters.

Step4. Resource replacement strategy: the resource replacement strategy is deployed when the storage capacity of a peer has reached its maximum and some resources stored on the peer are required to be removed to create space for the resource replica. The peer retrieves request rates of stored local resources from corresponding C_SPs and removes resource(s) with the lowest request rate(s) until enough space is freed for a new replica.

VI. SIMULATION RESULTS

A P2P simulation is created in our work to evaluate the performance of the proposed dynamic resource adaptation architecture. The following subsections first describe the implementation details of the P2P simulation and then show simulation results based on the evaluations of the proposed resource replication strategy.

A. P2P Simulation

The simulation is implemented in Java 2 Platform SE 1.5 [18] and in particular, uses its multi-thread technology to model the peers and super peers in the network. Therefore, each peer in the simulated network is initiated as a single thread in the Java runtime environment and all threads run concurrently to emulate simultaneous peer execution in a real P2P network. Threads interact with each other by invoking methods of other threads through referencing; this emulates the interactions among peers. Peer properties, shared resources and usage environment attributes such as

network conditions are also modeled in the simulation through variable definitions. In addition, a set of arrays are used in the main simulation file to store monitored information of each resource transaction between peers, as well as other information such as the number of resource variations and network delays in the network. The stored information is used to generate statistical data about the simulation at the end of each simulation. Furthermore, a simulation configuration file which contains various simulation settings (i.e., number of peers, super peer percentage, Zipf distribution [16] value etc.) is used to customize the simulation to model different environment.

TABLE I. SIMULATION SETTINGS

	Processor	Primary Storage	Secondary Storage	Connection Bandwidth
PC	3.2 GHz	2 GB	200 GB	100 Mbps (LAN)
Notebook	1.6 GHz	512 MB	40 GB	20 Mbps (WiFi)
PDA	400 MHz	64 MB	512 MB	400 Kbps (3G)
Mobile Phone	100 MHz	6 MB	64 MB	20 Kbps (GPRS)

When the simulation is executed, it is initialized by preloading the network with a specified number of normal and super peers based on the configuration file. Normal peers are further classified into provider peers and freeloaders according to the percentage values in the configuration file. Resources with various sizes are then generated and populated into the network according to the Zipf distribution [16] to simulate different shared resources in a typical P2P network (i.e., music samples and video files). The resources are proportionally assigned to the provider peers and freeloaders as specified in the configuration file. To simulate a heterogeneous usage environment, a set of device classes are currently used in the simulation. Devices from different classes possess different device capabilities and usage environment attributes. The current implementation of the simulation simulates up to four device classes to correspond to PC, notebook, PDA and mobile phone. In addition, network links between peers also vary in bandwidth and network conditions. The simulated device classes and the values of their capabilities are presented in Table I.

When a high-end peer simulates resource adaptation for a low-end peer, extra time spent on the adaptation is counted toward the measurements. The adaptation time of a resource is in proportion to its size and the processing speed of the adaptation device. For example, in the simulation it takes a PC 10 minutes to adapt a resource of 100MB. In order to achieve uniform results, the simulation does not consider different adaptation costs for different resource types. When the adaptation of a resource is completed, an adapted version of the resource is generated. Furthermore, popularity values are assigned to resources to simulate the popularities of old resources gradually decreasing during the simulation. These system settings and peer behaviors conform to the findings in the

survey conducted by Saroiu et al. [17]. During the simulation, requests are generated by peers in the system at random intervals and a predefined number of randomly generated/selected peers with resources join/leave the network. The simulation terminates when all requests are completed.

The simulation implementation is further expanded using the ‘thread pool’ technology in the newly implemented java.util.concurrent package of Java 2 Platform SE 1.5 [18]. With this approach, the simulation creates a ‘thread pool’. When a peer object is required to simulate resource exchange with other peers, the simulation instantiates the peer object as a thread and ‘injects’ it into the pool. Once the peer object finishes simulating resource exchanging, it is removed from the pool. In this way, threads are created for only peer objects that are simulating resource exchanging. Furthermore, the Excelsior JET native code compiler [19] is used to tune the performance of the simulation. The end result is that the network size in the simulation increased by more than tenfold from 7,000 peers in our initial simulation implementation [3] to 100,000 peers with up to 20,000 peers (i.e., threads) simultaneously exchanging resources.

B. Simulation Results of the Adaptive Resource Replication Strategy

TABLE II. SIMULATION SETTINGS

Total no. of peers	100,000
No. of super peers (including locality super peers)	100
No. of locality super peers	100
No. of provider peers	18,000
No. of freeloaderes	80,000
No. of unique resources (initially during startup)	15,000
No. of peer classes	4
Resource size	10-200 Mbs
Percentage of total resources owned by freeloaderes	30%
Percentage of total resources owned by provider peers/super peers	70%
Zipf skew factor	0.5
No. of new peers joined the network while running	20,000
No. of peers left the network while running	20,000
Total no. of requests	500,000

The results below were obtained based on the simulation settings in Table II. Other peer compositions have also been simulated, and the simulation results shown in the section demonstrate the typical trend.

In the simulation, the proposed replication strategy is compared with replication in commercial super peer P2P systems (i.e., KaZaA [7]) and a random replication strategy. Commercial super peer P2P systems always make a replica for each requested resource. The request peer tries to store a new resource into the local storage. However, when there is not enough storage space for the

new resource, a Least Recent Used (LRU) replacement policy is used to remove old resources. In the random replication approach, the requested resource is randomly assigned (i.e., yes or no) to be replicated into the local storage.

Average hit rate is used as a performance metric to measure the performance of the proposed replication strategy. The average hit rate is calculated based on the percentage of times a required resource is available in the locality cluster of the request peer.

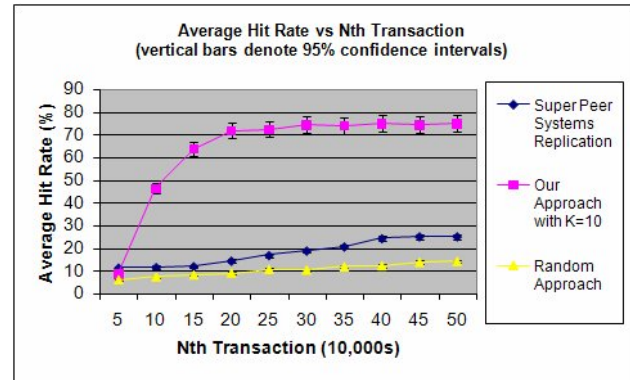


Figure 1. Average hit rates of different approaches

Fig. 1 shows that when K is set to 10, the hit rate increases by 3-4 times in comparison to the commercial P2P system replication strategy and 4-5 times in comparison to the random replication approach. The increase of resource availability also contributes to the decrease of average network delay which is evident in Fig. 2, where the network delay of the proposed approach is approximately twice and three times lower than the commercial P2P system replication strategy and random replication approach respectively. Also, the proposed approach shows a higher network delay in comparison to the commercial P2P system replication strategy in the beginning of the simulation. This is because the initial request rates of resources do not necessarily reflect the demand of resources. However, the network delay of the proposed approach quickly drops below the commercial P2P system replication strategy after the first 50,000 requests as the replication strategy distributes resources evenly across the network.

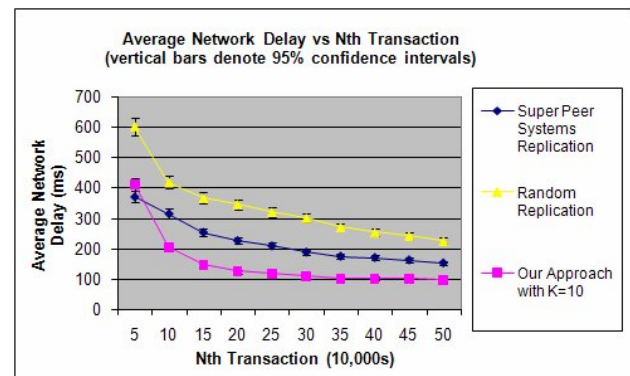


Figure 2. Average network delay of different approaches

The second comparison is made between the proposed replication strategy and a modified replication strategy which follows *step1* of Section IV.B, but replaces *step2* with choosing resource variations randomly to replicate in each locality clusters. Average download speed is used as a performance metric and it is shown in Fig. 3 that the proposed approach has a 10-15% higher download speed than the modified replication strategy, as peers are able to locate appropriate resource variations within their locality clusters.

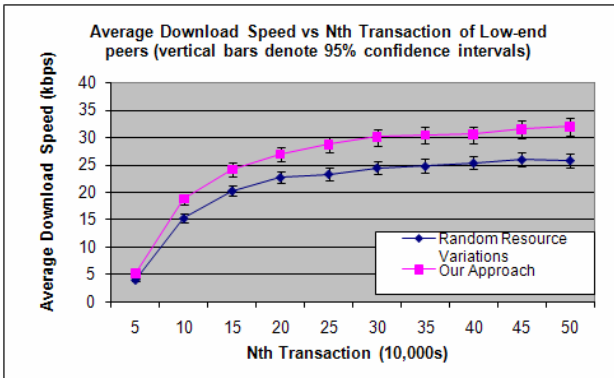


Figure 3. Average download speed comparison of low-end peers of different approaches

Next, the K value of the proposed approach is varied and the results are compared. Fig. 4 shows that the hit rate increases steadily as the value of K increases, until it reaches beyond a value of 10; after that, the hit rate does not show evident improvements; this is revealed in the graph when the K is set to 15. Simulation results for other peer compositions also indicate that the K value has a threshold. Similarly, it shows in Fig. 5 that network delay also holds a threshold for the K value. Currently we are investigating using formulas to determine the threshold value of the constant K.

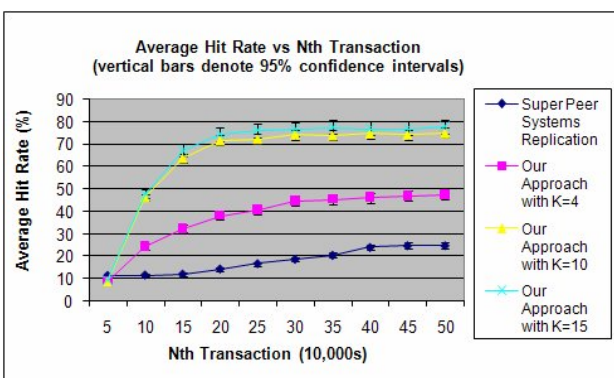


Figure 4. Average hit rate comparison of the proposed replication strategy with different K value

Finally, the simulation also shows that by deploying the proposed replacement strategy as described in *step4* of Section V, the number of resources in the network is sustained at a certain level (i.e., when K is set to 4 and 10, there is a 4% and 9% increase in number of resources respectively, in comparison to when no replication is used) while retaining the average hit rate performance

and keeping the average network delay low as mentioned above.

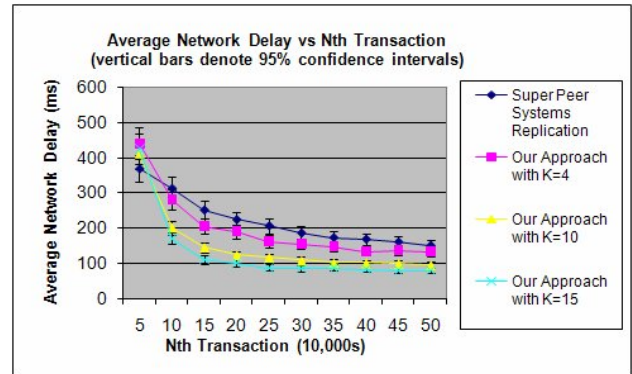


Figure 5. Average network delay comparison of the proposed replication strategy with different K value

VII. CONCLUSION

In this paper, we have proposed an adaptive resource replication strategy to facilitate resource replication in a dynamic resource adaptation P2P network. The replication strategy takes an active approach through using resource request rate as the prime metric to trigger the replication process and then adaptively creating resource variations in the network according to the heterogeneities of peers.

A multi-thread based Java P2P simulation is created to evaluate the performance of the proposed resource replication strategy. Simulation results have shown that the proposed replication strategy could greatly improve the average hit rate of resources while keeping the average network delays low in comparison to the replication in commercial super peer systems such as KaZaA and random replication approaches. Also, the results show that replicating resources into resource variations in proportion to the heterogeneities of peers improves the average download speed, compared to the random resource variation replication approach. Furthermore, the replacement strategy maintains the total number of resources in the network at a reasonable level.

REFERENCES

- [1] T. Standage, "Why Napster Matters," Prospect, pp. 50-53, October, 2000.
- [2] L. Rong and I. Burnett, "Facilitating Universal Multimedia Adaptation (UMA) in a Heterogeneous Peer-to-Peer Network," International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS'06), Leeds, UK, December, 2006.
- [3] L. Rong and I. Burnett, "Improved Dynamic Multimedia Resource Adaptation-based Peer-to-Peer System through Locality-based Clustering and Service," IEEE International Region 10 Conference (Tencon'05), Melbourne, Australia, November, 2005.
- [4] B. A. Alqaralleh, C. Wang, B. B. Zhou and A. Y. Zomaya, "Effects of Replica Placement Algorithms on Performance of structured Overlay Networks," in Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS2007), CA, USA, March, 2007.
- [5] Z. Xiang, Q. Zhang and W. Zhu, "Replication Strategies for Peer-to-Peer Based Multimedia Distribution Service,"

- in Proceedings of the 2003 International Conference on Multimedia and Expo (ICME'03), Baltimore, Maryland, USA, July, 2003.
- [6] V. Vlassov, L. Dong, K. Popov and S. Haridi, "A Scalable Autonomous Replica Management Framework for Grids," in Proceedings of the IEEE 2006 JVA International Symposium on Modern Computing (JVA'06), Sofia, Bulgaria, October, 2006.
- [7] KaZaA, <http://www.kazaa.com>.
- [8] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," in Proceedings of the ACM International Conference on Supercomputing 2002, New York City, USA, June, 2002.
- [9] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," in Proceedings of SIGCOMM'02, Pittsburgh, Kansas, USA, August, 2002.
- [10] G. On, J. Schmitt and R. Steinmetz, "The Effectiveness of Realistic Replication Strategies on Quality of Availability for Peer-to-Peer Systems," in Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03), Linköping, Sweden, September, 2003.
- [11] S. Tewari and L. Kleinrock, "Analysis of Search and Replication in Unstructured Peer-to-Peer Networks," in Proceedings of ACM SIGMETRICS, Alberta, Canada, June, 2005.
- [12] S. M. Thampi and K. C. Sekaran, "Autonomous Data Replication Using Q-Learning for Unstructured P2P Networks," in Proceedings of the 6th IEEE International Symposium on Network Computing and Applications (IEEE NCA07), Cambridge, USA, July, 2007.
- [13] Overnet home page, <http://www.overnet.com>.
- [14] B. Yang and H. Garcia-Molina, "Comparing Hybrid Peer-to-Peer Systems," in Proceedings of the 27th International Conference on Very Large Databases, Rome, Italy, September, 2001.
- [15] S. Tewari and L. Kleinrock, "On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks," in Proceedings of IFIP Networking, Waterloo, Canada, May, 2005.
- [16] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, "Web Caching and Zipf-like Distribution: Evidence and Implications," in Proceedings of the IEEE INFOCOM'99, New York, USA, March, 1999.
- [17] S. Saroiu, P. K. Gummadi and S. D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts," in Multimedia Systems Journal, Volume 9, Number 2, pp. 170-184 August 2003, Springer-Verlag.
- [18] Java 2 Platform Standard Edition (J2SE), <http://java.sun.com/j2se/1.5.0/>.
- [19] Excelsior JET – Java Virtual Machine (JVM) and Native Code Compiler, <http://www.excelsior-usa.com/jet.html>.
- [20] B. Y. Zhao, L. Huang and J. D. Kubiatowicz, "Tapestry: A Resilient Global-Scale Overlay for Service Deployment," IEEE Journal on Selected Areas in Communications, Vol. 22, No. 1, pp. 41-53, January, 2004.
- [21] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A Scalable Content Addressable Network," in Proceedings of SIGCOMM'01, San Diego, California, USA, August, 2001.
- [22] P. Heleher, B. Bhattacharjee and B. Silaghi, "Are Virtualized Overlay Networks Too Much of a Good Thing?," in Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, Massachusetts, USA, March, 2002.
- [23] Gnutella, <http://www.gnutella.co.uk/>.
- [24] B. Yang, and H. Garcia-Molina, "Design a Super-Peer Network," in proceeding of the 19th International Conference on Data Engineering (ICDE'03), Bangalore, India, March, 2003.

Dr. Letian Rong holds a BEng (Hons) in Computer Engineering from the University of New South Wales, Australia (2002), a Graduate Business Certificate from the University of Wollongong, Australia (2005), and a PhD in Computer Engineering from the University of Wollongong, Australia (2007).

He currently works in the ARLES group at INRIA, Versailles in France, where he is a research engineer (Ingénieur Expert), working on the mobility management aspect of B3G networks, as part of the EU IST-6 project PLASTIC (www.ist-plastic.org). His current research interests include mobile computing, wireless communications, P2P systems, multimedia systems and pervasive computing.