Retrieval and Pre-fetching algorithms for Segmented Streaming in Mobile Peer-to-Peer Networks

Zhou Su, Jiro Katto and Yasuhiko Yasuda

Graduate School of Science and Engineering, Waseda University

1． Introduction

In contrast to conventional P2P systems in wired networks that consist of static peers, the mobile P2P is subjected to the limitations of battery power, wireless bandwidth, and the dynamically changed network topology. Challenges arise in how to improve the source discovery and data replication [1][2].

In this paper, we talk about an integrated searching and pre-fetching algorithm for the segmented streaming in mobile Peer-to-Peer (P2P) Networks. Firstly, each stream is divided into several segments and each segment is assigned a priority based on theory analyses. Then, for a given segment, the different number of queries is sent to search it and the length of the query for this segment is also dynamically decided by the segment-priority to avoid the unnecessary overhead. Next, along the path where a stream is sent from the requester node, parts of the nodes on this path are selected to pre-fetch the requested segment to reduce the user delay for the next possible request. Finally, Simulation results show that better performance than the conventional methods can be achieved.

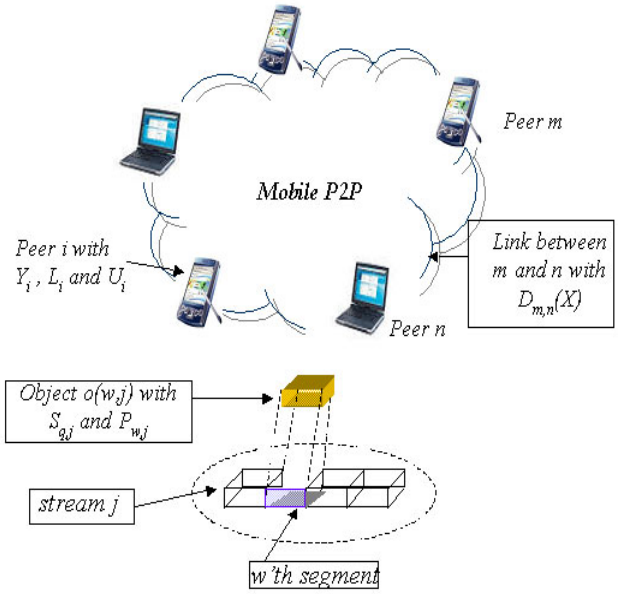

Figure1: Segmented Streaming in Mobile Peer-to-Peer Networks

2． Theory Analysis

2.1: query number priority (*QNP*)

Let $\wedge=\sum_i \lambda_i$ be the total request rate of all peers. A up ratio U_i is defined as the fraction of the time that Peer i is up (on line). Assume that Peer *i* has different battery power Y_i and L_i different links with its neighbor peers, each neighbor peer has a serial number l ($l=1,\ldots,L_i$), let's denote each link's RTT value as R_l ($l=1,\ldots,L_i$).A query number priority (*QNP*) for each neighbor peer *i* as follows:

$$QNP(l) = \beta / R_l + \alpha \cdot Y_i + (1-\beta-\alpha)\cdot U_l \cdot \lambda_l / \wedge \quad \text{(Eq.1)}$$

If a peer with high *QNP*, the query will first send this peer because it has both a high active ratio and fast link.

2.2: query number priority (*QNP*)

We define *TST* as the number of total queries sent by users in a test period. Let v_j denote the request times of the request for stream *j* during the above period. Then, we can get:

$$p(j) = g(j)/(\sum_w g(w)) \quad \text{(Eq.2)}$$

$$g(j) = v_j / TST \quad \text{(Eq.3)}$$

Assume that stream *j* has been requested by *Q* times within a fixed period and each time the client stopped watching stream *j* after the playback of segment $S_{q,j}$ ($q=(1,\ldots,W_j)$, $j=(1,\ldots,J)$)., the priority index of segment can be decided by

$$p(w|j)p(j) = (\left|\{S_{q,j} \,|\, w \le S_{q,j}\}\right| / Q) g(j) / (\sum_w (g(w)) \quad \text{(Eq.4)}$$

As *Q*, *TST* and the summation of $g(w)$ over *w* are identical for all streams in the above Equation, for convenience, we define a replication priority (*RP*) for the query of segment *w* of steam *j* as follows:

$$RP(w,j) = \left|\{S_{q,j} \,|\, w \le S_{q,j}\}\right| v_j / TST \quad \text{(Eq.5)}$$

3. Proposal

3.1:Search Algorithm

When a peer i $i=\{1,...,I\}$sends queries (walkers) to its neighbor peer l $(l=1,...,L_i)$ to request the w-th segment of stream j,

1) $QNP(l)$ in Eq.(6) is calculated. A threshold T_{QNP} is set up, if $QNP(l) > T_{QNP,}$ peer i will send query (walker) to peer l, otherwise, the query will not be sent. Then, the number of walkers can be controlled

2) A query to peer l $(l=1,...,L_i)$ is decided to be sent, the total length of this query (walker) will be calculated by the $RP(w,j)$ in Eq.(14). If we denote the total number of the search-step as ST, then the length at each step is $RP(w,j)/ST$.

The above algorithm is different from the conventional method, *Random Walks* algorithm, which forwards a query message to a randomly chosen neighbor at each step. Since both the number and the length of each walker in our proposal are varied according to the situation of the requested stream and neighbor peer, we name the proposed algorithm *Dynamic Streaming Walker*.

3.2:Pre-fetching Algorithm

Assume that a peer i ($i=\{1,...,I\}$) requests the w-th segment of stream j and the requested segment is sent from a peer pa_F along the path that consists of the following peers : peer pa_1, peer pa_2...peer pa_f...peer pa_F $(F<= I)$.

The length $Di,pa_f(\boldsymbol{X})$ is denoted as the shortest distance in AS hops from peer i to peer pa_f under the placement $\boldsymbol{X}$

STEP1: Along the path consists of peer pa_1, peer pa_2...peer pa_f...peer pa_F $(F<= I)$, their $RP(w, pa_f)$ will be calculated. Only the peers which are qualified to the following equation will be selected to go to *STEP2*.

$RP(w, pa_f) > thresh\ 1$

STEP2: Among the peers selected by *STEP1*, the peer which has the highest $RP(w, pa_f)$ will be selected to replicate segment w.

STEP3: The peer which distance is far away from threshold will be slected to replicate segment w

$D_{paf,\ max\{\ RP(w,\ paf)\ \}} > thresh\ 2$

Here, $max\{\ RP(w,\ paf)\ \}$ means the peer which was slected in *STEP2*

4. Evaluation of Proposal

In simulation experiments, we assume following conditions: There are 1000 nodes (peers) in our network simulator. As the recent study showed that most communication networks have Power-Law link distributions, network topology in our simulation is decided according to the Power-Law Random Graph: the i'th most connected node has Ω/r_i^{β} neighbors, and β is set to be 0.8.

The distribution of web request is decided by Zipf distribution, which states that the relative probability of a request for the j'th most popular page is proportional to Ω/r_j^{α}. Besides, the parameter of Zipf distribution is 0.8.

As for the streaming contents, there are 1000 different streams with the rate of 384kbps. The length of each stream is uniformly distributed from one minute to ten minutes and the size of one segment is 288kbyte. In our simulation, clients often stop watching a streaming content after playback. The position where a client stops watching a stream is decided at random. Client requests arrive according to a Poisson process. The total request times in the simulations are 10000. Note that we set the number of total queries (walkers), which can be used in the following three algorithms, to be the same, if the available queries are used up, the simulation will also be ended. That is to say: we test the performance of each algorithm based on the same condition, where the available resource (query) is fixed.

We show the performance of different replication algorithm in Fig.2. Three replication algorithms are compared. We define a relative performance, which is the ratio of the search success to the replication cost. From the result, we can know that our algorithm can obtain the highest relative percentage. For the *Owner Replication*, although the replication is low as only the owner can replicate the requested stream, its search success percentage is very low, resulting a low relative performance.

On the other hand, for *Path Replication*, although its search success percentage is very high because every node in the path replicates the requested node, its replication is very high. Therefore, the relative performance is decreased.

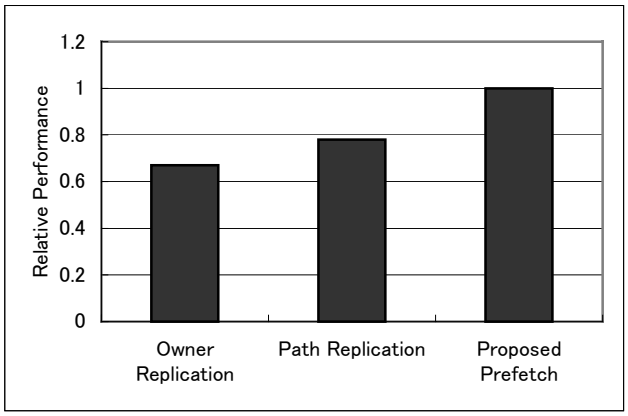

Fig. 2. Comparison of Relative Performance among Different Replication Algorithms

5.Reference

[1] Z,Jing, W,Yijie, L,Xicheng, and Y,Kan, “A Dynamic Adaptive Replica Allocation Algorithm in Mobile Ad Hoc Networks”, Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW, 2004)

[2] H.Shen, M.S.Joseph, M.Kumar, and S.K. Das, “PReCinCt: A Scheme for Cooperative Caching in Mobile Peer-to-Peer Systems”, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS, 2005)