行政院國家科學委員會專題研究計畫 成果報告

無線感測網路之關鍵技術及在社區照護之應用--子計畫四：無線感測網路之中介軟體及在社區照護之應用(3/3)
研究成果報告(完整版)

計 畫 類 別 ：整合型
計 畫 編 號 ：NSC 95-2221-E-002-061-
執 行 期 間 ：95 年 08 月 01 日至 96 年 07 月 31 日
執 行 單 位 ：國立臺灣大學電機工程學系暨研究所

計 畫 主 持 人：王勝德

計畫參與人員：博士班研究生-兼任助理：黃正一、莊勝彥
碩士班研究生-兼任助理：王皓漢、楊逢振、楊進福

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 96 年 10 月 30 日

行政院國家科學委員會補助專題研究計畫 ☑成果報告 □期中進度報告

無線感測網路之關鍵技術及在社區照護之應用-子計畫四：無線感測網路之中介軟體及在社區照護之應用

計畫類別： □個別型計畫 ☑ 整合型計畫
計畫編號：NSC 95-2221-E-002 -061
執行期間： 95年08月01日至96年07月31日

計畫主持人：王勝德
共同主持人：
計畫參與人員： 王浩漢 莊勝彥 楊進福 楊逢振 黃正一

成果報告類型(依經費核定清單規定繳交)：□精簡報告 ☑完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：國立台灣大學 電機工程學系
中 華 民 國 96 年 10 月 22 日

This work “The Modified Grid Location Service for Mobile Ad-Hoc Networks” by Hau-Han Wang and Sheng-De Wang

Has been published by

Lecture Notes in Computer Science
Springer Berlin / Heidelberg
ISSN 0302-9743 (Print) 1611-3349 (Online)
Volume 4459/2007
Advances in Grid and Pervasive Computing

ISBN 978-3-540-72359-2
pp. 334-347

The Modified Grid Location Service for Mobile Ad-Hoc Networks

Hau-Han Wang and Sheng-De Wang

Department of Eletrical Engineering
National Taiwan University, Taipei, Taiwan
sdwang@ntu.edu.tw

Abstract. Position-based routing has been proven to be a scalable and efficient solution for packet routing in mobile ad hoc networks (MANETs) by utilizing location information of mobile nodes. The location service provides geographic locations for all nodes and is therefore critical to position-based routing. In general, the control overhead in a position-based routing protocol is mainly dominated by location updates. In this paper, we propose a location service called Modified Grid Location Service (MGLS), which employs a binary grid partitioning scheme to reduce the control overhead associated with the location management and supports large scale ad hoc networks. We then use a theoretical model to analyze both MGLS and GLS. Both theoretical analysis and simulation results show that MGLS can reduce the location update overhead in location services.

1 Introduction

Routing protocols in MANETs are commonly categorized into two different types: *topology-based* and *position-based routing*. M. Mauve et al. [1] has presented such an overview of ad hoc routing protocols. The routing performance can be significantly improved by utilizing location information of nodes. That is, if each node is aware of the location of the destination and all its one-hop neighbors in the network, it can geographically forward a packet toward its destination. Position-based routing algorithms uses such additional location information to eliminate the limitations of topology-based routing. Commonly, each node determines its own position through the use of GPS (Global Positioning System). Before sending a packet to the destination, senders always include the location of destination which is provided by the so-called location service in the header of outgoing packets. The routing decision at each node is then based on the destination's position contained in the packet and the position of the forwarding node's neighbors. Position-based routing thus does not require the establishment or maintenance of routes; furthermore, it scales well even if the network is highly dynamic.

Location services provide the positions of the destination nodes to senders all around the geographic region. Existing location services can be classified according to the number of nodes that host the service and the range of nodes that is maintained by one location server. This can be either some specific nodes or all nodes of the network. Thus there are the four possible combinations as some-for-some, some-for-all, all-for-some, and all-for-all in the of location services. Recent algorithms [2]-[5] present some possible ways of finding destination and distributing location updates.

The Grid Location Service (GLS[2]), which provides a location service by mapping from node_id to current location. GLS divides the area that contains the ad hoc network into a hierarchy of squares. Each node maintains its current location at a small subset of network nodes, called the node's location servers. Location Servers for a node are relatively dense near the node and sparse farther away from the node. The route discovery for a destination is then equivalent to recursively querying the location servers until the query packet arrives at the one having the destination's location. Quorum systems[3][4], which route most packets through arbitrary participants. This reduces the danger that the special participants may become a bottleneck. The role of the special participants is limited to storing location tables and computing routes through the general network. DREAM[5] forces nodes to proactively flood their current location information over the entire network, enabling each node to build a complete location database. However, DREAM does not scale well to large networks due to its use of global flooding.

Forwarding strategies help nodes make routing decisions based on the destination's position included in the packet and the position of their neighbors. The Location Aided Routing (LAR[6]) uses geographic location to determine the search space for a destination, hence reducing the number of route-discovery packets of reactive ad hoc routing approaches. Besides, LAR restricts the search for a route to a so-called request zone which is determined based on the expected location of the destination node at the time of route discover. However, LAR uses flooding as a means of route discovery. This is done in a fashion similar to that of the DREAM approach. [7] had presented a complete comparison between these two schemes, because of the similarity of DREAM and LAR.

The Greedy Perimeter Stateless Routing (GPSR[8]) is such an instance of greedy packet forwarding, which uses a planer subgraph of the wireless network graph to route around dead-end. In GPSR, senders first include the approximate destination positions obtained from a location service into packets. Nodes then use the positions of routers and packets' destinations to make packet forwarding decisions; forward the received packet to a neighbor lying in the direction of the destination until the destination has been reached.

In geographic forwarding, a node announces its current position and velocity to its neighbors by broadcasting periodic HELLO packets. Each node maintains a table of its current neighbors' identities and geographic positions. Therefore, nodes may learn about two hop neighbors: nodes that cannot be reached directly, but can be reached in two hops via the neighbor that sent the HELLO message, it's called 2-hop distance vector. 2-hop distance vector helps alleviate holes in the topology and ensures that each node knows the location of all nodes in its own smallest grid. The header of a packet destined for a particular node contains the destination's identity as well as its geographic position. When node needs to forward a packet to location D, the node consults its neighbor table and chooses the neighbor closest to D. It then forwards the packet to that neighbor, which itself applies the same forwarding algorithm. The packet stops when it reaches the destination. GLS adopts geographic forwarding as its forwarding strategy. Actually, both geographic forwarding and GLS belong to the GRID project[9].

Another survey of position-based routing in ad hoc networks was presented by I. Stojmenovic[10]. T. Park et al. proposed a hybrid routing protocol[11] constructed by combining well-known location-update schemes, which minimizes the overall routing overhead in terms of location-update thresholds. Some location services with fixed static hierarchy such as DLM[12], SLURP[13], SLALoM[14] and HIGH-GRADE[15] are compared systematically in [16].

In this paper, we proposed a distributed location service scheme for position-based routing in mobile ad hoc networks, called Modified Grid Location Service (MGLS) which is an improvement to GLS. Similar to GLS, in our scheme, the entire network is partitioned into hierarchical grids. Each node is randomly assigned an integer as its node ID and is placed at uniformly random location over the network. These nodes act as end systems and routers at the same time. In order to maintain the location information in a decentralized way, each node has several location servers in the network. As nodes move, this location information is constantly updated. Before sending a packet to a node, the sender first queries the destination's location and then uses the geographic routing protocol to forward the packet to the destination. Since the cost of location management usually dominates the overall protocol overheads. MGLS was designed to reduce the amount of location updates with a delicate grid hierarchy. We also use a theoretical model for studying the location service scalability, based on which we analyze our scheme as well as GLS. The analytical results are then validated by simulation in medium to large size networks.

2 Overview of MGLS Scheme

MGLS exploits geographic forwarding as its forwarding strategy. First, all nodes know the same global partitioning of the ad hoc network into a hierarchy of grids, as we will describe in the following section. Next, since every node in the network acts as an end system and a location server of other nodes at the same time,

the mechanism of location server selection has to be defined clearly. Nodes will periodically update their location servers with their current location obtained by GPS. Finally, if one node A wants to transmit a packet to another node B, A queries the location servers of node B for B's current location before using geographic forwarding. Actually, every node in the network has a predefined unique ID in integer, as well as our wireless card has an unique MAC address in a wireless network.

2.1 Grid Hierarchy

The whole network is partitioned into grids as shown in Figure 1. The grids in the figure are unit grids in the network referred to as *level-0* grids with the ratio $1 : \sqrt{2}$ in width and length. Two *level-0* grids adjoined on the larger side make up a *level-1* grid, two *level-1* grids adjoined on the larger side make up a *level-2* grid, and so on. Obviously, our grid hierarchy has a characteristic of recurrence. Grids of all levels keep the same ratio of $1 : \sqrt{2}$, and the area of *level-(n+1)* is twice as large as level-n.

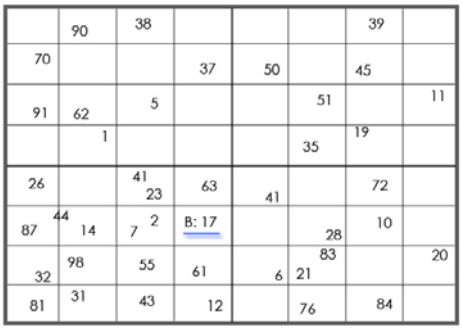
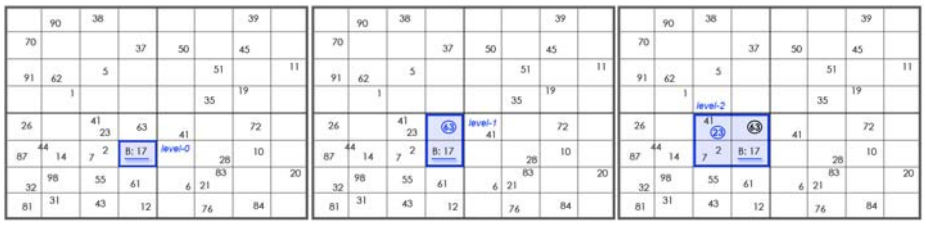


Fig. 1. Formatting an ad hoc network

2.2 Location Servers

We believe that using centralized location servers is not a good idea. Due to the limitation of radio transmission range, the only one location sever may be out of reach of most mobile nodes. Besides, a single server is too weak to provide reliability of location service, it is unlikely to scale a large number of mobile nodes. In order to offer a fault-tolerant scheme, we have to make our location service distributed. That is, one mobile node has multiple location servers located in the whole network. So that MGLS can provide distributed lookup service by replicating the information of nodes' current locations.

Selecting Location Servers and Updating Location Information Every node uses its ID and the predefined grid hierarchy to determine which nodes are its location servers. In the Figure 2, node B has an ID of 17 and wants to update its location servers after moving a certain distance. The strategy is that one node picks one other node with ID "least greater" than its own ID to be its location server for each level of the grid hierarchy. Note that the ID space is ordered in a circular fashion. We defined 2 is closer to 17 than 7 is to 17.



(a) (b) (c)

Fig. 2. A flow diagram illustrates how does a node B seek its location servers. The nodes which become B's location servers are circled.

Here is an example. Let's start from the Figure 2(a). B is located in its own *level-0* grid. Then in Figure 2(b), the *level-0* grid of node B "grows" to be a *level-1* grid containing another node 63. Since 63 is the "least greater" node in ID space than B, so 63 is selected as a location sever of B in its *level-1* grid. In Figure 2(c), 23 is the least greater node than B again, following a rational line, 23 is B's location server in its *level-2* grid, and so on. The same location server selection process repeats until the *level-i* grid of B covers the whole network, where i is supposed here to be 6 in our example.

Grid Location Service (GLS) divides an a network into a hierarchy grid of squares, too. The *level-i* square is recursively divided into 4 *level-(i-1)* squares until *level-0* squares are reached, forming a so-called quad-tree. In each *level-i* square, node B selects 3 location servers, one in each *level-(i-1)* square that B isn't in. However, in both schemes, the number of location servers that a node must recruit is equal to the number of neighbors per level in the geographic hierarchy multiplied by the number of levels in the hierarchy. For GLS, this means that a node must maintain $3 \log_4 n$ location servers in a network. While MGLS, which splits the network in half at each level, rather than in fourths, by using rectangles with an aspect ratio of $1 : \sqrt{2}$. This leads to a network in which nodes recruit only $\log_2 n$ location servers, that is, 2/3 the number of location servers needed in GLS. Figure 3 gives a contrast to GLS.

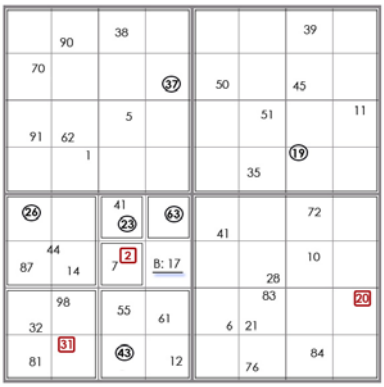


Fig. 3. The same case of GLS, location server 2, 20, 31 are demanded in addition for node B.

As a node moves, it must update its location servers. Nodes avoid generating excessive amounts of update packets by bounding their location update rates to their traveled distance. A node updates its *level-1* location servers every time after moving a particular threshold distance δ since sending the last update. The node updates its *level- 2* servers after each movement of $\sqrt{2}\delta$. In general, a node updates its *level-i* servers after each movement of $\sqrt{2}^{i-1}\delta$. As a result, a node sends out updates at a rate proportional to its speed and that updates are sent to distant location servers less often than to local servers.

Location Query In Figure 4, each node is shown with the list of nodes for which it has up-to-date location information. To perform a location query, node A sends a request by using geographic forwarding to the least greater node than B for which A has location information. That node forwards the query in the same way. In the end, the query will reach a location server of B which will forward the query to B. Since the query contains the location of A, B thus can respond to A directly using geographic forwarding.

2.3 Design Tradeoffs

As we have seen, MGLS changed the grid organization from quad- to binary-partitioning. As a result, the number of location servers kept by each node is reduced and thus the cost of location maintenance for MGLS may be redueced. However, MGLS may come with an increased query path length due to the decrement of the number of location servers, as shown in Figure 4, where a location query packet was sent from node C (with ID: 76) to 21. It was then forwarded to node 20, a location server of B in GLS, so that this query packet could be forwarded directly to the query destination in one hop earlier than the query packet in MGLS.

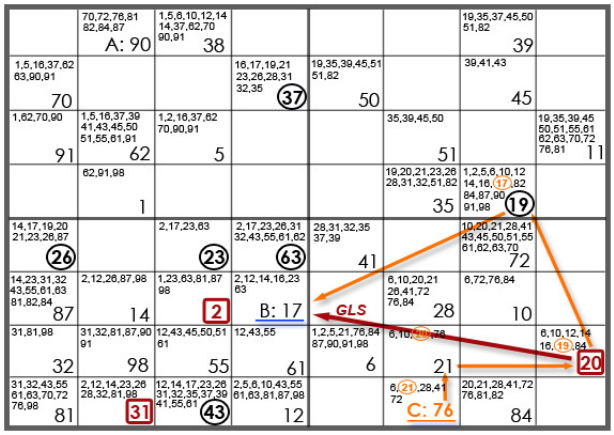


Fig. 4. An example of location querying operations in MGLS.

3 Comparisons Based On a Theoretical Model

In this section, we exploit a developed theoretical model [16] to analyze the scalibility of MGLS and GLS. The focus of this analytical work is to demonstrate how design choices affect the protocol costs of the two schemes.

3.1 Metrics

We first define three metrics to be the criteria of evaluating the scalability of each scheme.

Definition - *Location Maintenance Cost:* The location maintenance cost C_m is defined as the number of forwarding operations each node needs to perform in one second to deal with the location update packets. It can be regarded as the cost of maintaining up-to-date location information on location servers in the network.

Definition - *Location Query Cost:* The location query cost C_q is defined as the number of packet forwarding operations due to location queries each node needs to perform in one second. It can be regarded as the cost of acquiring location information from location servers before sending data packets to other nodes in the network.

Definition - *Storage Cost:* The storage requirement cost C_s of a location service is defined as the number of location records a node needs to store as a location server. We measure this metric by counting the number of entries instead of calculating the bytes of location tables.

We separate the location maintenance and query costs for one reason. We believe that the location query cost is relatively easy to be reduced in a location service

scheme by employing various caching strategies, while the location maintenance cost is not. Thus, we will focus on the location maintenance cost in the following.

3.2 Model Assumptions

The rest of this section derives the expected values of the first and the third metrics as functions of N and v. The node density γ is supposed to be a constant. We also assume that γ is high enough that geographic forwarding is operational. (According to GLS, geographic forwarding works fine only if $\gamma \geq 50$ $nodes/km^2$. Actually, the variable γ approaches 100 $nodes/km^2$ in our experiments.) We assume that nodes are moving according to a simplified random way-point mobility model. Each node picks a random point in the network and moves toward it with a random velocity v chosen uniformly between [0, v_{max}]. After the point is reached, node selects a new random point with zero pause time. Let $P_i, \forall i = 0, \cdots, H$ denote the probability that node B (the querying node) and A (the node being queried) are co-located in the same $level - i$ grid. Based on the size of the $level - i$ grids, P_i can be easily estimated as:

Lemma 1: *(Grid Coexistence Probability).*
The probability of the querying node and the queried node are located in the same $level - i$ grid is

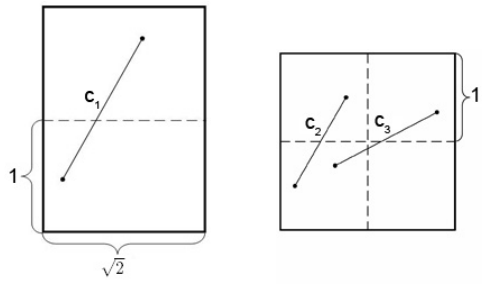
$$P_i = \begin{cases} \frac{1}{2^{H-i}} \text{ if MGLS} \\ \frac{1}{4^{H-i}} \text{ if GLS} \end{cases} \quad \forall i = 0 \cdots H$$

3.3 MGLS

Location Maintenance Cost As we described in Section 2.1, MGLS uses binary grid-partitioned algorithm instead of quad grid-partitioned. A node A selects one location server in each $level - i$ grid ($i = 1 \cdots H$). Since all location servers of A have to store the current location positions of A, they are expected to be updated periodically to ensure freshness of location information and to reduce the query failure rate. In MGLS, A updates its $level - i$ server after each movement of $(\sqrt{2}^{i-1} \cdot \delta)$, where δ represents the update threshold which can probably be a few hundreds of meters. The updating period is set as the expected time a node moves a distance of $(\sqrt{2}^{i-1} \cdot \delta)$, namely $(\sqrt{2}^{i-1} \cdot \delta)/v$.

Theorem 1. *For MGLS,* $E(C_m) = \dfrac{c_1 \cdot \sqrt{2} \cdot R}{\delta \cdot z} \cdot v \log N$; $E(C_s) = \log N$.

Proof: To compute the location maintenance cost C_m, we first consider the expected distance that an updating packet has to travel in the $level - i$ grid, denoted as $E(d_i^u)$, and the average number of hops a updating packet takes from node A to $A's$ location server in the $level - i$ grid, denoted as $E(n_i^u)$. Since one node may be randomly located anywhere in a $level - i$ grid, we can view d_i^u as the distance between two random points in two $level - i$ grid adjoined on a side. Therefore,



(a) (b)

Fig. 5. (a)Constant c_1 - random distance between a pair of nodes in two MGLS unit squares adjoined on a side. (b)c_2 - random distance between a pair of nodes in two GLS unit squares adjoined on a side; c_3 - random distance between a pair of nodes in two unit squares adjoined on a corner.

$$\begin{aligned} E(d_i^u) &= \sqrt{2}^i R \int_0^{\sqrt{2}} \int_0^1 \int_0^{\sqrt{2}} \int_0^1 \sqrt{(x_1-x_2)^2+(y_1-y_2)^2} dx_1 dy_1 dx_2 dy_2 \\ &= c_1 \cdot \sqrt{2}^i R \end{aligned}$$

where R is a constant representing the shorter side length of a $level-0$ grid. Since the size lengths of $level-i$ grid are in the ratio of $1:\sqrt{2}$, the term $\sqrt{2}^i R$ thus corrects the computation of integral in any $level-i$ of grid. And c_1 is a constant factor representing the average random distance between two neighboring grids, as shown in Fig.5(a), $c_1 \leq \sqrt{6}$.

The expected number of hops in forwarding the packet is the expected distance divided by z, the average progress of each hop, which can be viewed as a function of the radio transmission range and the node density. Since we assume both as constants in our model, so is z. Thus,

$$E(n_i^u) = \frac{E(d_i^u)}{z} = \frac{c_1 \cdot \sqrt{2}^i R}{z}$$

Since updates are sent out at a rate of $v/(\sqrt{2}^{i-1} \cdot \delta)$ (δ represents the update threshold), we have

$$\begin{aligned} E(C_m) &= \sum_{i=1}^{H} \frac{v}{\sqrt{2}^{i-1} \cdot \delta} \cdot E(n_i^u) \\ &= \sum_{i=1}^{H} \frac{v}{\sqrt{2}^{i-1} \cdot \delta} \cdot \frac{c_1 \cdot \sqrt{2}^i R}{z} \\ &= \frac{c_1 \cdot \sqrt{2} \cdot R}{\delta \cdot z} \cdot \sum_{i=1}^{H} v \\ &= \frac{c_1 \cdot \sqrt{2} \cdot R}{\delta \cdot z} \cdot vH \end{aligned}$$

$$= \frac{c_1 \cdot \sqrt{2} \cdot R}{\delta \cdot z} \cdot v \log N$$

where $H = \log N$.

As for the Storage Requirement Cost: C_s, remember that the storage requirement is defined as the number of location records a node needs to store as a location server. The average number of records a node stores is the total number of records stored in the network divided by the total number of nodes. Since every node has one location server in each level, we have

$$E(C_s) = \frac{N \cdot H}{N} = \log N. \qquad \blacksquare$$

3.4 GLS

The GLS scheme uses a similar multilevel structure of the grid hierarchy as MGLS. A node A selects three location servers in each $level - i$ square, one in each $level - (i-1)$ squares quadrants that A is not in, as shown in Figure 3. An important difference between GLS and MGLS is the distinct hierarchies of the grid structure. The same as MGLS, all the location servers need to be updated periodically in order to ensure freshness of location information and to reduce the query failure rate. We now prove the following theorem for GLS.

Theorem 2. *For GLS,* $E(C_m) = \frac{(2c_2 + c_3) \cdot R}{z \cdot \delta} \cdot v \log \sqrt{N}$; $E(C_s) = \frac{3}{2} \log N$;

Proof: We first consider the location maintenance cost C_m. According to the GLS algorithm, all moving nodes update their location servers after the distance of $(2^{i-1} \cdot \delta)$; at a period of $(2^{i-1} \cdot \delta)/v$. Consider the expected distances the three update packets traveled to update the three locations servers in the $level - i$ square, denoted $E(d_i)$. We have

$$E(d_i^u) = (2c_2 + c_3) \cdot 2^i R \quad \text{, and}$$
$$E(n_i^u) = \frac{E(d_i^u)}{z} = \frac{(2c_2 + c_3) \cdot 2^i R}{z},$$

where $2^i R$ is the side length of a $level - i$ square, c_2 and c_3 are two constant factors representing the average random distance between two points in two neighboring squares, as shown in Figure 5(b). Simply, we have $c_2 \leq \sqrt{5}$, and $c_3 \leq 2\sqrt{2}$. Since updates are sent out at a rate of $v/(2^{i-1} \cdot \delta)$, we have

$$\begin{aligned} E(C_m) &= \frac{v}{(2^{i-1} \cdot \delta)} \cdot \sum_{i=1}^{H} \frac{(2c_2 + c_3) \cdot 2^i R}{z} \\ &= \frac{(2c_2 + c_3) \cdot R}{\delta \cdot z} \cdot \sum_{i=1}^{H} \frac{v \cdot 2^i}{2^{i-1}} \\ &= \frac{(2c_2 + c_3) \cdot R}{\delta \cdot z} \cdot 2vH \\ &= \frac{(2c_2 + c_3) \cdot R}{z \cdot \delta} \cdot v \log \sqrt{N} \end{aligned}$$

where $H = (1/2)\log\sqrt{N}$, since GLS use a quad-grid partitioning. Finally, since every node in GLS has three location servers in each level, the expected value of the storage cost for GLS is,

$$E(C_s) = \frac{N \cdot 3H}{N} = \tfrac{3}{2}\log N. \qquad \blacksquare$$

3.5 Summary of Theoretical Analyses

The analytical results of MGLS and GLS share the same asymptotic costs, as their designs exhibit the same philosophy. However, the constant factors in the cost are different. It is obviously that the storage cost of MGLS is smaller than that of GLS. As for the location update cost, which is usually the dominating overhead in location services, MGLS are also smaller than GLS since $c_1 \cdot \sqrt{2}$ is smaller than $2c_2 + c_3$ in the worst case, where $c_1 \leq \sqrt{6}$, $c_2 \leq \sqrt{5}$, and $c_3 \leq 2\sqrt{2}$.

4 Performance Evaluation using Simulation

This section presents simulation results for both MGLS and GLS. The GLS implementation we used for simulation is that of [17, NS-2 simulation for Grid]. An outstanding study of GLS's simulator was presented by M. Kasemann et al.[18]. Our MGLS simulation was implemented by making some necessary modifications to the GLS simulator.

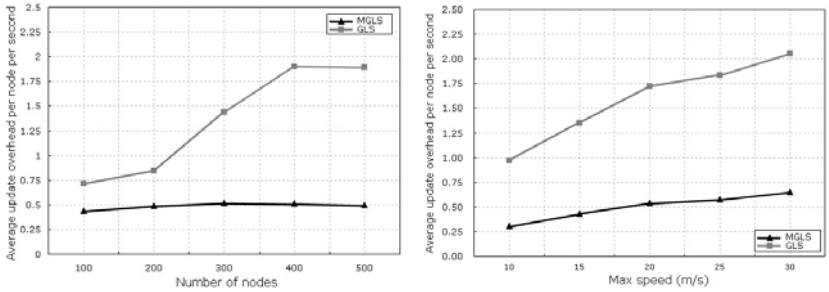
Simulation Settings The simulations use CMU's wireless extensions for the NS-2 simulator. The radio transmission range for each node is generally acknowledged $250m$. The simulations use 2 Megabits per second radios. Each simulation runs for 300 seconds, during which time, each node generates on average 4 data packets to other nodes per second. Nodes move according to the random waypoint model. Each time a random target is chosen, a moving speed is selected between zero and a maximum moving speed, where the maximum moving speed of the simulation is $30m/s$ by default. When the node reaches the destination, it chooses a new destination and begins moving toward it immediately, with no pause time.

Protocol Constants All nodes are initially randomly placed across the entire network area. For all the simulation runs, the initial node density is about $100 nodes/km^2$. One reason for this choice is that we intend the system to be used over relatively large areas such as a campus or municipality, rather than in concentrated locations such as a conference hall. Therefore, the size of network area increases linearly with the number of nodes. For a network of 500 nodes in MGLS, which is the biggest simulation we have done, the grid hierarchy goes up to $level-7$ in a universe of $2800m \times 2000m$. For both MGLS and GLS, the side length of a $level-0$ grid is set to be $250m$ (in MGLS, it would be $354m \times 250m$). The location updating threshold is $150m$ in both schemes.

Performance Metrics We considered the performance metrics, includeing average update cost and the qurery success rate [2][15]. In order to have precise experimental results, we created three levels of traffic loadings in our simulation: *100%, 50%, 10%* of N. We make this by giving three distinct bounds (which can be set in the CBR scenario files) to the number of connections between mobile nodes. For the case of high loading in the simulation, the number of maximum connections between nodes is set to be equal to the total number of nodes. The number of maximum connections equals half the total number of nodes in the case of medium loading. In the low loading network, the number of maximum connections is only one-tenth the number of nodes. Each data point in each of the three levels of traffic loading networks is an average of five simulation runs. In the results presented below, each data point is an average of the three scales traffic loadings. The simulations will demonstrate that MGLS fulfills an impressive balance between designing choice against N and v.

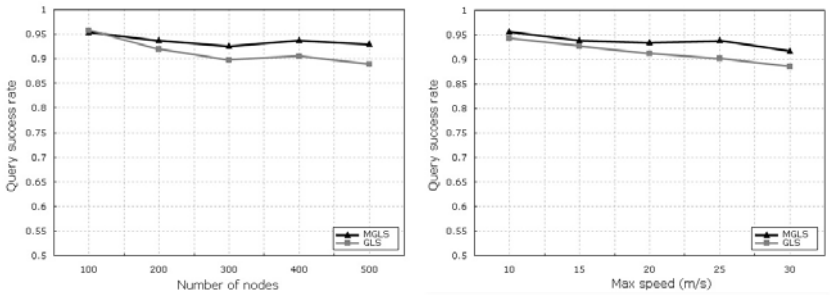
We are interested in the effects of mobility in nodes. High mobility will result in a significant protocol overhead. Dealing with mobility needs a tradeoff between the quality of location maintenance and the bandwidth available for data packets. Aggressive updating can increase query success rate but will occupy the bandwidth shared with data packets, while loosely location updates may have an opposite effect.

Protocol Overhead Figure 6 shows the average location update cost as a function of (a) the total number of nodes N and (b) maximum moving speeds of nodes v. The location update cost of MGLS is smaller than that of GLS as expected in our analysis.



(a) (b)

Fig. 6. Average location update cost as a function of total number of nodes and the nodes moving speeds.



(a) (b)

Fig. 7. Query success rate as a function of total number of nodes and the nodes moving speeds.

Protocol Performance Figure 7 shows the query success rate for both two schemes, as a function of (a) the total number of nodes N and (b) maximum moving speeds of nodes v. Most query failures are due to stale location information stored on the servers. Both schemes maintain quite satisfactory query success rate, around *90%* or above, where the MGLS has a little bit better query success rate than GLS. This result may be due to the lower overhead associated with the MGLS.

5 Conclusions

In this paper, we presented the design and performance of an efficient location service for mobile ad hoc networks. We also used a theoretical model to analyze the behaviors of both MGLS and GLS. With an enhanced grid partitioning scheme and reasonable tradeoffs, MGLS reduces the protocol overheads in comparison with GLS. Mathematical analysis and simulation results confirmed the performance advantages of our scheme. Future work may be aimed at supporting energy-efficient or quality-of-service (QoS) for discovering routes, where single-path routing used in both MGLS and GLS.

References

1. M. Mauve, J. Widmer, and H. Hartenstein. *A survey on position-based routing in mobile ad hoc networks.* IEEE Network Magazine, p30-39, November 2001.
2. J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. *A scalable location service for geographic ad-hoc routing.* In Proceedings of ACM MobiCom, p120-130, August 2000.
3. Z. J. Haas and B. Liang. *Ad Hoc Mobility Management with Uniform Quorum Systems.* IEEE/ACM Trans. Net., vol. 7, no. 2, p228-240, Apr. 1999.
4. I. Stojmenovic et al.. *A routing strategy and quorum based location update scheme for ad hoc wireless networks.* SITE, University of Ottawa, Tech. Rep. TR-99-09, September 1999.

5. S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. *A distance routing effect algorithm for mobility (DREAM).* In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom), p76-84, 1998.
6. Y.Ko and N.H.V aidya. *Location-aided routing (LAR) in mobile ad hoc networks.* In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom), p66-75, 1998.
7. T. Camp, J. Boleng, B. Williams, L. Wilcox, and W. Navidi. *Performance comparison of two location based routing protocols for Ad Hoc networks.* In Proceedings of the IEEE INFOCOM, 2002.
8. B. Karp and H. T. Kung. *GPSR: greedy perimeter stateless routing for wireless networks.* In International Conference on Mobile Computing and Networking (MobiCom 2000).
9. The grid project homepage, http://www.pdos.lcs.mit.edu/grid
10. I. Stojmenovic. *Position based routing in ad hoc networks.* IEEE Commmunications Magazine, Vol. 40, No. 7, p128-134, July 2002.
11. Taejoon Park , Kang G. Shin. *Optimal tradeoffs for location-based routing in large-scale ad hoc networks.* IEEE/ACM Transactions on Networking (TON), v.13 n.2, p.398-410, April 2005.
12. Y. Xue, B. Li, and K. Nahrstedt. *A scalable location management scheme (DLM) in mobile ad-hoc networks.* In Proceedings of the IEEE Conference on Local Computer Networks (LCN '01), 2001.
13. Seung-Chul M. Woo and Suresh Singh. *Scalable routing protocol (SLURP) for ad hoc networks.* Wireless Networks, 7(5):513-529, 2001.
14. Christine T. Cheng, H. L. Lemberg, Sumesh J. Philip, E. van den Berg, and T. Zhang. *SLALoM: A scalable location management scheme for large mobile ad-hoc networks.* In Proceedings of IEEE WCNC, March 2002.
15. Yinzhe Yu, Guor-Huar Lu, and Zhi-Li Zhang. *Enhancing Location Service Scalability with HIGH-GRADE.* Dept. of Comp. Sci. & Eng., U of Minnesota, Technical Report TR-04-002, 2004.
16. Y Yu, GH Lu, ZL Zhang. *Location Service in Ad-Hoc Networks: Modeling and Analysis.* In Proceeding of NSF Workshop on Theoretical and Algorithm Aspect of Ad Hoc Wireless Networks, Chicago, June 2004.
17. NS-2 simulation for Grid, http://pdos.csail.mit.edu/grid/sim/index.html
18. H. Hartenstein, M. Kasemann, H. Fubler, and M. Mauve. *A simulation study of a location service for position-based routing in mobile ad hoc networks.* Technical report, Department of Science, University of Mannheim, TR-02-007, July 2002.

會議名稱：**The 12th IEEE International Symposium on Pacific Rim Dependable Computing (PRDC'06)**

出席報告

會議日期：December 18-20, 2006

會議地點：University of California, Riverside, USA

- 主辦單位：
  - **IEEE Computer Society Technical Committee on Fault-Tolerant Computing**
  - **University of California, Riverside**

    In Cooperation with

    IFIP WG 10.4 on Dependable Computing and Fault Tolerance
- 姓名：王勝德
- 服務單位：國立台灣大學電機系
- 職稱：教授

PRDC'06 從投稿 117 篇論文中接受了 42 篇論文，接受率 35%，發表的論文爲 IEEE Proceedings on PRDC'06 所收錄。

本次議題包括：

- Software and hardware reliability, testing, verification and validation
- Dependability measurement, modeling and evaluation
- Safety-critical systems and software
- Architecture and system design for dependability
- Fault tolerant algorithms and protocols
- Tools for design and evaluation of dependable systems
- Reliability in Internet and Web systems and applications
- Dependability issues in computer networks and communications
- Dependability issues in distributed and parallel systems
- Dependability issues in real-time systems, database and transaction processing systems

本人所發表paper屬於Architecture and system design for dependability, 本會邀請了三場 keynote speech:

Keynote 1: December 18, 2006

Title: Quality Wave in Light of The Nano Development
Speaker: Professor Way Kuo

Abstract: Nano technologies are a driving force for strong economic growth in the world, and some analysts predict that its impact will bring to us the next industrial revolution. In the 2005 National Academy's publication of Keck Futures Initiative, reliability is cited as the key element of the success of nano fabrication and manufacturing. In this keynote speech, we will address both the historical review of the neno technologies ever since the industrial revolution and recent development, particularly those events which have great impacts on quality and computing. Some new challenges will be discussed as well.

Keynote 2: December 19, 2006

Title: "Challenges in Dependability of Future Networked Systems"
Speaker: Professor Takashi Nanya

Abstract: As networked systems pervade every aspect of the modern information society, we are faced with serious threats to dependability due to problems caused by accidental events such as human mistakes and physical malfunctions or by intentional behavior being either malicious or non-malicious. In this talk, we discuss major challenges and give views of future directions in research on the dependability of evolving networked systems toward an advanced information society.

Keynote 3: December 20, 2006

TITLE -- Code Coverage: The Missing Link Between Software Testing and Software Reliability?

SPEAKER -- Professor Michael R. Lyu
Computer Science & Engineering Department, The Chinese University of Hong Kong

ABSTRACT --
While hardware testing and reliability techniques are closely related, software testing and reliability approaches were developed independently, sometimes with conflicting principles. Software testers spend their most testing efforts in exceptional test cases, while software reliability engineers require software to be tested under a normal operational profile. Software testers are interested in knowing how software testing covers the development requirements. Software reliability engineers are interested in how software reliability is perceived from customer views. Software testers do not trust numbers. Software reliability engineers insist software quality cannot be an objective attribute without creditable reliability measures.

The main issues in software testing are the design and evaluation of effective test cases, and relating software testing with the resulting reliability. Code coverage was proposed as an estimator for testing effectiveness, But it remains a controversial metric in linking testing with reliability. In this talk, we focus our research questions

regarding the measure of code coverage on testing effectiveness under various testing strategies, and evaluate the influence of code coverage to software reliability measurement. We conduct experiments to investigate the relationship between code coverage and fault detection capability under different testing profiles. From our experimental data, code coverage is merely a moderate indicator for fault detection regarding the overall testing strategies examined on the whole test set. However, it is clearly a good fault detection estimator with exceptional test cases. Moreover, we analyze the effects of different coverage metrics and how coverage can be used to in reliability measurement, and establish a new reliability model incorporating both testing time and code coverage. New research directions in software testing and reliability will also be given.

這演講的三個主題反應了不同的研究主題，都很具代表性，例如 第一個主題主要講nano technology的可靠性議題，第二個的主題有關網路資訊系統的可靠性議題，第三個主題則有關軟體測試與軟體可靠度，雖然本人的研究興趣慢慢轉移到嵌入式系統，但在嵌入式系統設計也有可靠度的問題，所以本人對軟體測試與軟體可靠度主題，感到興趣，另外在會議的一部份主題也探討到如何利用設計方法來提高可靠度，這方面也跟軟硬體共同設計和特殊電路設計有關，這些方面都與嵌入式系統相關。

本次會議地點在UC Riverside 舉行，主辦單位借電機系的教室為會場，沒有一般大型會議在飯店舉辦的方便，但比較樸實，有點像我們在台灣常常為了節省經費，使用學校的場地來開會一樣。UC Riverside 比較內陸，從洛杉磯要到會場，以自行開車比較方便，我也是從機場租車前往，回程時順道參觀了西來寺，感覺佛光山的確很有規模，在西方能夠建立如此宏偉的道場，不禁令人肅然起敬。

會議攜回資料: 會議論文CD.

附錄：所發表之論文

A Dependable Outbound Bandwidth Based Approach for Peer to Peer Media Streaming

Zheng-Yi Huang and Sheng-De Wang
Department of Electrical Engineering
National Taiwan University, Taipei, TAIWAN
Email: sdwang@ntu.edu.tw

***Abstract*— A fundamental problem in peer-to-peer streaming is how to select peers with desired media data so that the best possible streaming quality can be maintained. In this paper, we propose an outbound bandwidth based streaming model in which peers are layered according to their offered outbound bandwidth and are permitted to request data of peers from upper layers peers only. Based on the layered approach, a media data assignment algorithm for the subset of media data is presented to select qualified sending peers to ensure that they will be received before their scheduled playback time. We also present two resolutions for request conflicts, which arise when there are more than one peer simultaneously requesting data from the same sending peer that can't afford outbound bandwidth for all requests. We evaluated the proposed streaming model through simulations. Experimental results show that streaming quality of the proposed streaming model is excellent and the properties of scalability as well as robustness are obtained even in a highly dynamic environment where peers join and leave frequently.**

1. Introduction

Recently, the peer-to-peer networking paradigm offers the alternative possibility for streaming media over the network due to the most important inherent characteristic: resources are shared among participants. Peers that simultaneously function as both clients and servers share their resources (e.g., computing power, bandwidth, storages and contents) with others. This important characteristic avoids dedicated replication servers altogether and hence it does not have the bottleneck of client/server streaming architecture. Unlike peer-to-peer file sharing systems, a peer-to-peer media streaming system is operated in a kind of play-while-downloading mode ([1]). It is implied that the outbound bandwidth (upstream bandwidth) that a peer can acquire is a crucial factor for streaming quality. Another element of diversity is that the local storage of each peer is leveraged as a cache-and-relay mechanism ([4]) in which requesting peers request media data and cache the most recently played media data during streaming. The cached content can then be relayed to later peers that request the same content. But the most distinct feature of all is that from peer-to-peer streaming systems, users not only enjoy the availability of media content but also the high quality of media streaming. The media streaming quality depends on a combination of factors, ranging from the characteristics of the streaming sources (e.g., link capacity, availability, accessible outbound bandwidth) to the characteristics of the network paths (e.g., available bandwidth, packet loss rate, and overlap of paths from multiple sources to receiver [7]). Thus the main challenge is to design peer selection strategies to realize high streaming quality. Much research has been done to optimize solutions for streaming that aim at providing a minimal delay time for buffering media data. We argue that such optimal solutions are adequate only in some particular situations such as the number of requesting peers is relatively less than the number of sending peers. In addition, in order to get an optimal solution, one trends to exhaust peers with powerful capability, for example peers with high outbound bandwidth.

In this paper, we study the peers selection problem as well as the media data assignment problem as in [1]. Given requesting peers and sending peers (supplying peers) with heterogeneous outbound bandwidth, we proposed to select sending peers and assign a subset of media data to them based on the outbound bandwidth layered hierarchy. Different from other optimal peer selection algorithms which result in optimal solutions for an individual requesting peer, our goal is to provide a high streaming quality for whole participants and to maintain the fairness among peers according to their contribution to the network. Also unlike other elaborate streaming architectures, the control overhead of the proposed streaming model is extremely low and it can apply to any existing peer-to-peer overlay network. Briefly, peers participating in the streaming network are layered according their offered outbound bandwidth to form a hierarchy in which they are permitted to request data from peers in higher layers only. To avoid peers from requesting data blindly and from accessing resource greedily, a media data assignment algorithm is presented such that peers can get their required media data with limited resource and media data can be received before their scheduled playback time so that the streaming quality can be maintained. Since there is limited capacity (hereafter we will use outbound bandwidth and capacity alternately) of a sending peer, request conflicts would occur frequently and inevitably. A request conflict occurs when there are more than one peer that simultaneously request data from the same sending peer that can't afford enough capacity for all requirements. In this paper, we propose two possible mechanisms for resolving request conflicts, namely capacity sharing and capacity preemption which enlarges the accessible capacity of a requesting peer and gives peers chances to preempt other peers with lower priority, respectively. Both of these mechanisms aim at maintaining a best possible streaming quality for all participants in the network. The features of the proposed outbound bandwidth

based streaming model are:

- The construction of the outbound bandwidth based layer is distributed and the communication overhead of the streaming process is extremely low.
- The proposed streaming model is scalable and resilient to peer failures.
- The obtained streaming quality is high and stable even if in a highly dynamic environment where peers join or leave frequently.
- Fairness is achieved for all peers, where the peers that contribute more capacity will have a higher streaming quality.

The rest of this paper is organized as follows. Section 2 defines the assumed media streaming model. Section 3 presents the proposed layered streaming hierarchy. In Section 4, we present two possible technologies for request conflicts resolution and the layered outbound bandwidth based media streaming. The results of simulations are showed in Section 5. Finally, Section 6 describes related work and Section 7 concludes this paper.

2. Peer-to-Peer Media Streaming Model

In this section, we describe the proposed peer-to-peer media streaming model and the terminologies of this paper.

A. Media streaming model

We assume that the media data can be partitioned into small sequential sessions of variable sizes and each session can be segmented into small sequential segments of equal sizes. A session is referred to as size κ if it is composed of κ segments and a segment is the unit of transmission in the streaming model. Each segment is attached with a unique sequence number which determines the playback order and ranges from zero to M-1, given that the media is of M-segments. Segments seg_i is thought to play before seg_j if $i<j$. We also assume that the media data is of constant-bit-rate and with multiple description coding (MDC) and the playback rate is R_0. If the size of a segment is M_0, then its playback time is $T_0 = \frac{M_0}{R_0}$.

B. Playback model

During the playback of session s, peer P_i requests and caches segments of session $s+1$ and then the cached segments are available for being requested from other peers. P_i repeats the streaming processes session by session until the completion of the playback or the leaving of the peer. If the scheduled playback time of the first segment of session s is $\tau_{1,s}$, then the scheduled playback time of j-th segment of s will be $\tau_{j,s} = \tau_{1,s} + j-1$. And $\tau_{1,s}$ is of course $\tau_{1,s-1} + \kappa$ if session $s-1$ is of size κ. Let $\rho_{j,s}$ be the received time of j-th segment of session s. Then we say that segment j is a valid segment if and only if $\rho_{j,s} \leq \tau_{j,s}$, since a segment arriving after its scheduled playback time is useless and considered lost. P_i will discard invalid segments and won't cache them even if they have arrived later.

3. The outbound bandwidth based streaming layer

In this section, we propose the outbound bandwidth based hierarchy as the backbone for streaming as well as the media data assignment algorithm for selecting qualified peers with required segments.

A. The outbound bandwidth based layer

When a peer P_i joins the streaming network, it sets aside bandwidth as its offered outbound bandwidth and which has one of the following values:$\{\frac{R_0}{2}, \frac{R_0}{2^2}, \frac{R_0}{2^3}, \cdots \frac{R_0}{2^N}\}$. To simplify notations, we use R_ℓ as the abbreviation for $\frac{R_0}{2^\ell}$ and say $P_i \in (R_\ell)$ if P_i is a peer of outbound bandwidth R_ℓ. We assume that there are source peers $\in (R_0)$ that possess a copy of media file in the network. Peers are divided into layers according to their outbound bandwidth and P_i is said to in layer ℓ if $P_i \in (R_\ell)$. We use the notation of the *smaller* the ℓ, the *higher* the layer and let $\mathcal{L}_{P_i}$ denotes the layered value of P_i. Now, we give the layered relation for requesting and sending segments as the backbone for peer-to-peer streaming.

Definition Let (P_i) be the set of accessible peers of P_i. Then, for all P_j in the streaming network

$$(P_i) = \begin{cases} \{P_j : 1 \leq \mathcal{L}_{P_j} < \mathcal{L}_{P_i}\} & \text{if } \mathcal{L}_{P_i} \geq 2 \\ \{P_j : \mathcal{L}_{P_j} \leq \mathcal{L}_{P_i}\} & \text{if } \mathcal{L}_{P_i} \leq 1 \end{cases}$$

Peers in layer one are permitted to request segments from source peers directly and peers in other layers except layer one can request segments from peers of upper layers but not source peers. In another word, peers are allowed to send segments to peers of lower layers. For example, the accessible set of peers for a layer-4 peer is $\{(R_1), (R_2), (R_3)\}$. Therefore, the set of available outbound bandwidth is $\{R_1, R_2, R_3\}$.

Based on the layered relation defined above, we generalize three characteristics of the outbound bandwidth based layer. First, the direction of streaming is endowed since segments will stream from the highest layer down to the lowest layer. In other words, peers in higher layers will have newer segments than peers in lower layers. Second, peers in higher layers can have a smaller standard deviation of transmission time if the size of the requested session is equal to the size of the accessible set and each requested layer is responsible for delivering one segment. For example, the accessible set of a layer-4 peer is $\{(R_1), (R_2), (R_3)\}$, then the transmission time of each layer is therefore $\{2T_0, 4T_0, 8T_0\}$. Thus, the standard deviation of transmission time will be $2.49T_0$ under the above assumptions. But for a layer-5 peer, it will be $5.36T_0$. It implies that peers in a higher layer will have a shorter waiting time for completing one streaming process of a session. Third, peers in a higher layer take more opportunities to send segments towards peers in lower layers. Hence, peers in a higher layer will take charge of more dissemination of segments. To summarize, the goal of layered hierarchy is to keep track of fairness such that peers that volunteer to offer higher outbound bandwidth will make use of higher streaming quality but they will also be burdened with heavier responsibilities for data dissemination. Another goal is to rule the streaming direction such that peers can avoid both

TABLE I

NEIGHBOR TABLE AND ACCESSIBLE LAYERS OF P_i

Neighbors		(P_i)	
Peer	Layer	Layer	Peer
P_1	1	1	P_1
P_2	2	2	P_2, P_3
P_3	2	3	P_4
P_4	3	-	-
P_5	4	-	-

blind pull(request) or push(send). This can tend to result in extremely communication overhead and will cause streaming quality to be degraded significantly.

Each peer has its own layered hierarchy and the establishing process is distributed with little overheads. In addition, the layered hierarchy can be found on any existing peer-to-peer overlay network. In this paper, however, we adopt the gossip-based approach for the construction of the overlay since its excellent robustness, scalability and reliability as evaluated in detailed by Laurent Massouli et al. ([2]). Upon peer P_i setting up its neighbor table (or partial view [2]), P_i constructs its layer by information stored in its neighbor table and then prepares its media streaming. For example, if neighbors of P_i are as the left-half part of table I and if P_i is of layer four, then its accessible set will be { (R_1), (R_2), (R_3)}. Therefore, its outbound bandwidth based layer will be the right-half part of table I. It is evident that the layer constructing process is distributed and the additional overhead is to exchange layer information with the neighbors. And due to duplicated peers of each layer, the layered streaming model will be reliable even in the present of peer failures.

B. Media Segments Assignment for Layers

Unlike other segments assignment research, the problem we study is how to assign media segments to layers such that segments can be received before their scheduled playback time, given an admissible buffering time of a session and its size. An admissible buffering time is the maximum time a peer is allowed to request and receive segments for a session to such an extent that each segments will be cached on time.

To prevent peers from requesting peers from particular layers arbitrarily and greedily, a maximum accessible peers for the layers are defined for each session. Algorithm 1 defines the maximum peers for each layer that a requesting peer P_i can request, namely (P_i), where σ, called *outbound degree*, defines the scope of claimable outbound bandwidth. It is obvious that if $P_i \in (R_\ell)$, then $|(P_i)| = \sigma \cdot \ell$. For example, if $P_i \in (R_4)$ and $\sigma = 1$, then $|(P_i)|$=4, i.e. (P_i)={ (R_1), (R_2), (R_3), (R_3)}. That is to say, P_i can use 1, 1 and 2 peers from layer 1, layer 2, and layer 3 respectively during a streaming process. And their total outbound bandwidth is R_0, of course.

Algorithm 2, executed by requesting peers, assigns media segments to layers. Its intrinsic idea is very intuitive and simple: just make on time segment by segment. Giving the size of a session, the assignment algorithm selects layers for

```
1: procedure AccessiblePeers
2:     total ← 0;
3:     (P_i) ← ∅;
4:     while total < σ do
5:         ℓ ← round robin select a layer in [L_{P_i} − 1 , 1];
6:         if total + R_ℓ ≤ σ then
7:             let P_c be a peer ∈ (R_ℓ);
8:             (P_i) ← (P_i) ∪ P_c;
9:             total ← total + R_ℓ;
10:        end if
11:        if (total MOD 1)=0 then
12:            set selected index in [L_{P_i} − 1 , 1] to L_{P_i} − 1;
13:        end if
14:    end while
15: end procedure
```

Algorithm 1: maximum accessible peers

segments such that they pass for being received before their scheduled playback time in the view of the requesting peers. In the algorithm, there are two auxiliary values associated for a suppling peer. δ_{P_s} is the time it takes P_s to deliver a segment. If $P_s \in (R_r)$ then $\delta_{P_s} = 2^r T_0$. ϕ_{P_s} indicates the next free time of P_s and is maintained by requesting peer locally. By locally, we mean that it won't be influenced or considering requesting statuses of other peers. However, it also means that a local free status might not match the actual free status of the supplying peer. We will propose mechanisms to fix this problem in the next session. Initially, δ_{P_s} is zero, if a segment is assigned to sending peer P_s, then the local next free time of P_s will be updated to current $\delta_{P_s} + \phi_{P_s}$.

```
1: procedure Assign(size)                          ▷ the size of session_s
2:     if size = 1 then
3:         Segment_{size−1} ← CANDIDATE(0);
4:     else
5:         Assign(size − 1);
6:         Segment_{size−1} ← CANDIDATE(SIZE-1);
7:     end if
8: end procedure
```

Algorithm 2: Assigning Layers for Session

```
1: procedure Candidate(seq)                        ▷ the sequence of segment
2:     τ ← τ_{seq,s} + seq;                        ▷ the scheduled playback time
3:     for all P_c ∈ (P_i) do
4:         select a P_c if δ_{P_s} + φ_{P_s} ≤ τ
5:     end for
6:     φ_{P_c} ← δ_{P_c} + φ_{P_c};
7:     return L_{P_c}                              ▷ return the layer of selected peer
8: end procedure
```

Algorithm 3: Selecting Candidate Layer

When there are more than one candidate layers for a segment, a decision must be made to select one for it. The following are four possible selection strategies for this situation.

- Lowest Layer First: to select a lowest layer from multiple candidate layers. The delivering time of this layer is the

longest among other strategies but the burdened outbound bandwidth is lightest.
- Highest Layer First: to select a highest layer from multiple candidates. Opposite to lowest layer first, this strategy has the shortest delivering time and heaviest burdened outbound bandwidth.
- Random: select a layer randomly.
- Least Recently Assigned: according to previous selection history, it selects a layer which has least recently been assigned on account of balancing utilization of outbound bandwidth.

A comparison of performance among different strategies will be discussed in Section 5. In addition, if the layer selection strategy is the highest layer first, buffer size is 2^k and outbound degree is R_o, then our assignment algorithm has the same result as the optimal algorithm in [1].

Now, we prove that the media segments assignment algorithm always generates valid segment assignments. To simplify notations, we will use the local sequence of segments instead of the original sequence of segments in the proof. Given a session s, the local sequence of segment k of s is equal to k minus the original sequence of the first segment of s. If we can prove that for each segment $k, 0 \le k \le \mathcal{K}-1$, of a sized $\mathcal{K}$ session, there exist some peer which can make segment k valid, then the property is proved. we prove it by mathematical induction.

Theorem 1: The media segments assignment algorithm shown in Algorithms 2 and 3 always generates valid segment assignments.

Proof Let (S_k) denote the set of candidate sending peers for segment k and let $\mathcal{B}_{size}$ denote the size of a session. For $\tau_{1,s} > 3$, $\mathcal{L}_{P_i} \ge 3$. and $\sigma \ge 1$.

1) When $\mathcal{B}_{size} = 1$. Initially, ϕ_{P_c}=0, $\forall P_c \in (P_i)$. Then $(S_0) = \{P_c : \phi_{P_c} + \delta_{P_c} \le \tau_{1,s}\}$. Obviously that $|(S_0)| \ge 1$ since $\exists P_c \in (R_2)$ such that $\delta_{P_c} + 0 \le \tau_{1,s}$.
2) Assume when $\mathcal{B}_{size} = \mathcal{K}$ is true. That is $\rho_{\mathcal{K}-1,s} \le \tau_{\mathcal{K}-1,s} = \tau_{1,s} + \mathcal{K} - 1$. Then ϕ_{P_c}=$\{\tau_{1,s} + \mathcal{K} - 1 - c_j : 0 \le c_j \le \tau_{1,s} + \mathcal{K} - 1\}$. Let $c_j = j$. Then we have $|c_0| \le 1$ and $1 \le |c_j| \le |(P_i)|$, for $1 \le j \le \tau_{1,s} + \mathcal{K} - 1$.
3) Consider $\mathcal{B}_{size} = \mathcal{K} + 1$. $(S_{\mathcal{K}}) = \{P_c : \phi_{P_c} + \delta_{P_c} \le \tau_{1,s} + \mathcal{K}\}$. That is $(S_{\mathcal{K}}) = \{P_c : \tau_{1,s} + \mathcal{K} - 1 - c_j + \delta_{P_c} \le \tau_{1,s} + \mathcal{K}, 0 \le c_j \le \tau_{1,s} + \mathcal{K} - 1\}$. Consider $\mathcal{F}(P_c, c_j) = -1 - c_j + \delta_{P_c}$. We have $|\mathcal{F}(P_s, c_j) \le 0| \ge 1$ since $\exists c_j$, $j \ge 1$ and $P_c \in (R_r), r \ge 1$ such that $\mathcal{F}(P_s, c_j) \le 0$. That is $|(S_{\mathcal{K}})| \ge 1$, $\exists P_c$ can make segment k as a valid one.

By the mathematical induction, the algorithm is proved.

4. THE OUTBOUND BANDWIDTH BASED MEDIA STREAMING

In this section, we present two mechanisms for request conflict resolution as well as the layered outbound bandwidth based streaming model. They all aim at maintaining a best possible streaming quality for participants in the network.

A. Request Conflict Resolution Mechanisms

One of the major issues with media streaming when taking whole participants into account is how to resolve request conflict when it occurs. Peers will encounter request conflict unexpectedly and frequently since they request their desired segments in a distributed and un-synchronous fashion and timing constraints of playback will force peers to grab segments through their best efforts. Moreover, due to the limited capacity of a sending peer, it occurs inevitably. It should be resolved obviously otherwise streaming quality will degrade and none will get stable quality. To overcome this problem, we propose two possible technologies to resolve it, namely *capacity sharing* and *capacity preemption*. These two mechanisms will be considered only when there is a request conflict, and just one of them or none will be executed depending on conditions at the time.

1) Capacity Sharing: During a normal request, a requesting peer, P_i, will take whole parts of required segments from peers that have been requested. However, with capacity sharing, P_i may take partial segments from peers that have been requested and the remainder from peers not been requested who have the same required segments and the same requested target. Upon receiving a request from P_i, a sending peer, P_s, determines whether data can be shared among P_i and peer being serviced. If it is sharable, P_s accepts the request of P_i and notifies these sharable peers to sending partial segments to each other. The following algorithm determines whether the request from P_i is sharable with peers being serviced.

```
1: procedure SharableSet( P_i)
2:     ← ∅;
3:   for all P_j ∈ P_s do
4:     for all k ∈ P_i do
5:       if δ_{P_j} + φ_{P_j} < τ_{k,s,P_i} then
6:         ← ∪ {{k, P_j, P_i}};
7:         φ_{P_j} ← δ_{P_j} + φ_{P_j};
8:       else if δ_{P_i} + φ_{P_i} < τ_{k,s,P_j} then
9:         ← ∪ {{k, P_i, P_j}};
10:        φ_{P_i} ← δ_{P_i} + φ_{P_i};
11:      end if
12:    end for
13:  end for
14: end procedure
```

Algorithm 4: Determining Sharable Set for P_i

After determining the sharable set for P_i, P_s execute the following procedure to send segments to peers and to notify peers to share segments with shared target.

```
1: procedure SharableSending
2:   for all {{k, P_j, P_l}} ∈ do
3:     send k to P_j;
4:     notify P_j to send k to P_l;
5:   end for
6: end procedure
```

Algorithm 5: Segments sending for sharable peers

It's obvious that there is no harm for sending peers to perform capacity sharing mechanisms since all one needs

to do is to notify sharable peers with little communication overhead. It won't lengthen delivering time for sending peers nor shorten it. The benefit of capacity sharing is the reduction of opportunities being rejected while peers issue requests. But the major drawback of this is the reduction of network capacity implicitly since more are capacities being consumed in order to share segments with each other.

2) Capacity Preemption: In the capacity preemption mechanism, priority is assigned to each peer according to which layer it locates. Peers in higher layers deserve higher priority to access resources. Upon receiving a request from P_i, P_s determines whether there are preemptable peers based on the following procedure. Preempted peers will lose their required

```
1: procedure PREEMPT(P_i)
2:     for all P_j ∈     do
3:         select one P_j such that L_{P_j} < L_{P_i};
4:     end for
5:     if such P_j exists then
6:         P_i preempts P_j;
7:     end if
8: end procedure
```

Algorithm 6: Capacity preemption

segments requested from P_s and of course their streaming quality will degrade. However, from experimental results in section 5, we will see that capacity preemption is the decisive factor which maintains a best streaming quality for whole participants in our layered streaming model.

B. On outbound bandwidth based peer-to-peer streaming

When peer P_i joins to a media streaming network, it establishes its neighbor membership by the underlined peer-to-peer overlay protocols. In the meantime, P_i exchanges the information of it's located layer with its neighbors to build its own outbound bandwidth based layer. Upon receiving the layer information of P_i, peers should update their layers to reflect the effect that P_i will contribute and consume capacity for the network. Before the playback of the media data, P_i must determine which segment should be its initial playback segment ς_{P_i} suggested by the following equations, assuming that $\mathcal{L}_{P_i} = \ell$.

$$max(\varsigma_{P_k}) < \varsigma_{P_i} < min(\varsigma_{P_j}) \quad \forall P_k \in \quad (R_{\ell+1}), \forall P_j \in \quad (R_{\ell-1})$$

The reason why the initial playback segment of P_i is not as live as source peers is that since P_i will become a sending peer who sends segments to others located in lower layers, it also will be a requesting peer to request segments from peers of higher layers. Therefore, the initial playback segment should not be too new otherwise there will be few peers or none can satisfy its requirement and it should not be too old or it can't provide any segments for peers in lower layers since their desired segments will be newer than cached by P_i. Thus, a eclectic strategy for determining a initial playback segment is to choose a segment with sequence less than the least sequence of segments among it's neighbors of an immediate upper layer and greater than the largest sequence of a segment among it's

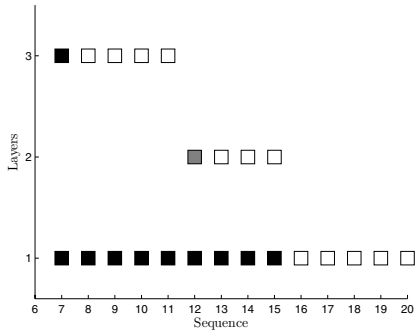


Fig. 1. The best initial segment of P_i

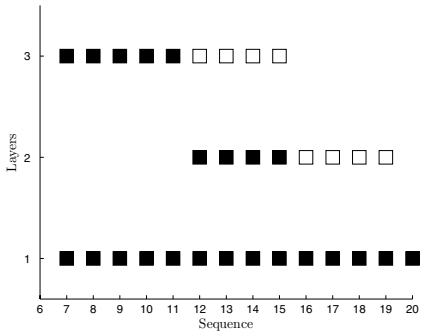


Fig. 2. cached segments after 4 T_0

neighbors of an immediate lower layer. For example, if the new joining peer P_i belongs to (R_2), and the largest initial segment of its immediate low layer is 8, and the least initial segment of its immediate up layer is 16, then the best initial segment of P_i will be 12 as shown in Fig 1 assuming that peers in layer 3 requested a four-size session. Peers in layer 3 who request their sessions initialed from segment 8, won't request data from P_i, since when they schedule their request targets (during play segments from 4 to 7), P_i does not exist in their neighbor table, and therefore it will avoid quality degrade since P_i still does not have enough data. But in Fig 2, peers in layer 3 may and can request segments from P_i since P_i had cached their desired segment in its cache. Consider the relations between P_i and its immediate up layer, layer 1. If P_i requests a session with initial segment larger than 12, then P_i won't get its required data since none have cached them.

After initializations described above, P_i can perform streaming processes and will act as both a requesting peer and a sending peer by performing procedure 7 and 8, respectively. P_i executes the requesting procedure for buffering next session during playback of current session. As described in the previous section, P_i issues requests relying on its local free statuses of sending peers. In case of having the required segments not being received which are requested from some sending peer, says $P_{s'}$, P_i will record $P_{s'}$ as an ill-reputation sending peer and it will try to select a peer with a best reputation for requesting next time. This selection strategy makes our

streaming model more resilient since it is self-adapting to avoid requesting from unstable situations such as high frequent request conflict, high end-to-end delay or peers which may have failed or left. Upon P_s receiving a request from P_i, P_s

```
1: procedure REQUEST(session_s)
2:     do layers assignment for session_s;
3:     for all k ∈ session_s do
4:         for all P_j ∈ Neighbors and L_{P_j} =k.layer do
5:             select a P_j with best reputation;
6:         end for
7:         request k from P_j;
8:     end for
9: end procedure
```

Algorithm 7: Requesting procedure of requesting peer

checks whether the requested segments were cached in its buffer or not. If the answer is no, P_i won't get anything and therefore its streaming quality will be degraded. Otherwise, P_s determines whether it can afford enough outbound bandwidth to deliver segments or is there any preemption or sharing possible depending on conditions mentioned earlier?

```
1: procedure SEND(k)                          ▷ k is the requested segment
2:     if k ∈ cache then
3:         if δ_{P_s} + φ_{P_s} < τ_{k,s} then
4:             send k to P_i;
5:         else if preemptable then
6:             do preemption;
7:         else if sharable then
8:             do sharing;
9:         end if
10:    end if
11: end procedure
```

Algorithm 8: Sending procedure of sending peer

5. EVALUATION

To investigate the performance of the proposed streaming model, we have carried out extensive simulations under various scenarios and the detailed experimental results are presented in this section. For each experiment, we report the mean value of results obtained through 10 runs with network size varying from 10,000 peers to 25,000 peers. Peers belong to one of following layers R_1, R_2, R_3, R_4, R_5 and their distribution is 12.5%, 20%, 35%, 20%, 12.5%, respectively, under the assumption of a normal distribution. In addition, there are source peers possessing a copy of a media file in the network and their total outbound bandwidth is $100R_0$. In the first set of experiments, we study behaviors of the streaming model under three aspects: conflict resolving mechanism, outbound degree and layers selection strategies respectively. And a highly dynamic environment in which peers join or leave(failure) frequently was simulated in the second set of experiments. Finally, we show fairness and the streaming direction through experimental results. To quantify the performance of media streaming system, we define quality of a streaming session as

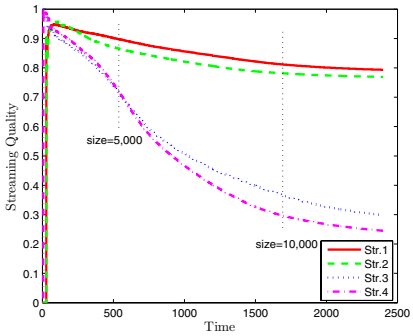


Fig. 3. Comparison of Request Conflict Resolutions

in [8]

$$Q = \frac{\sum_{i=0}^{\kappa-1} Z_i}{\kappa} \quad (1)$$

where κ is the number of segments of a media session and Z_i is a variable that takes value 1 if segment i arrives at the receiver before its scheduled playback time, and 0 otherwise. A segment that misses its deadline is discarded and, therefore, does not contribute to the quality of a streaming session. The streaming quality of a requesting peer is the mean quality of all its required sessions. And the *streaming quality* showed in the experimental results is defined as the mean quality of all requesting peers in the network.

The goal of simulations is to confirm that our streaming model possesses four features: *high and stable streaming quality, scalability, robustness* and *fairness*.

A. The Investigation of System's Behavior

In the first set of experiments, we investigate streaming quality in terms of a variant model's behaviors. The goal of these experiments is to study the effect of the model's parameters and to understand their implicit meanings through experimental results.

1) The influence of request conflict and its resolutions: As mentioned in the previous section, peers' streaming behaviors reveal their distributed and unpredicted nature; there is no way to know the streaming intention of other peers, such as when to issue requests and who is the requested target. Hence, request conflict is inevitable during streaming. To inspect the impact of request conflicting and the improvement by conflict resolutions, the experiment was made by streaming models with conflict resolving mechanisms versus a streaming model without any request conflict resolution. Fig 3 exhibits the compared results of four strategies. Strategy 1 is a streaming environment adopting both preemptive mechanism and capacity sharing as discussed earlier; when there is a choice that should be made among them, preemptive mechanism will be considered first. Strategy 2 adopts preemptive mechanism only and Strategy 3 adopts just capacity sharing. Strategy 4 is a environment without handling request conflict. From the experimental results, Strategy 1 outperforms others. But when considering the improved streaming quality compared

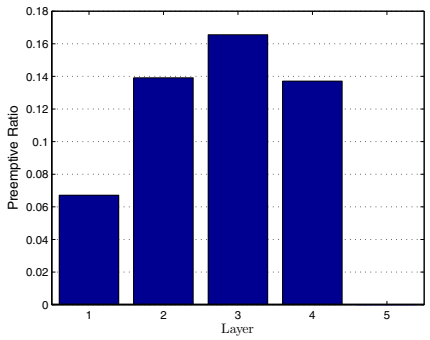


Fig. 4. Preemption ratio of each layer

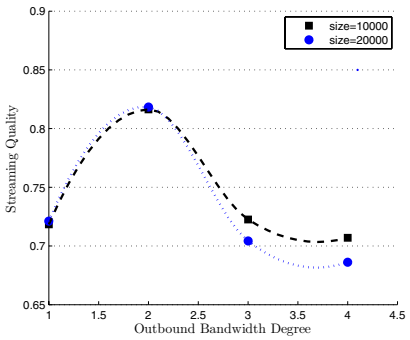


Fig. 5. The Effect of outbound degree

with Strategy 4, Strategy 2 has an impressive improvement. The reason is that, in our streaming model, peers are allowed requesting media segments only from peers in a higher layer, and there will be a chain of segments lost in the case of peers in a higher layer that could not acquire their desired media segments. This situation will cause streaming quality to degrade significantly if request conflict takes place frequently and peers in a higher layer does not have higher priority to access resources. Although preemption will also bring about streaming quality to degrade in the preempted peers, the proportion of preemption is slight as show in Fig. 4. To go a step further in discussion the reason why the proportion of preemption is slight is that the buffering time and delay time of peers is basing on layers of peers, peers in a lower layer will have longer buffering time and therefore delay time, which means that they can request media segments from peers in "lower layer" (but higher than themselves) and it will reduce opportunities been preempted. For this reason, the preemptive mechanism improves streaming quality considerably. Strategy 3 also improves streaming quality although its improvement is not as a good deal as Strategy 2. Since when performing a sharing process, there are at least three peers involving in one media segments, say one sending peer and two requesting peers, it will consume the capacity of the streaming network indirectly. Moreover, if there are a lot of sharing processes in a higher layer, it will reduce the ability to service peers in lower layers. It is evident that ignoring the resource conflict phenomenon is not a good idea because of its terrible streaming quality.

2) The Impact of Outbound Degree: The outbound degree defines the accessible scope a peer can access for one session. That is, it is the number of peers involved during streaming in one session. The larger the outbound degree, the more the peers will be involved. We would like to know what is the most suitable outbound degree for a streaming system and how it is related to the streaming quality. Will a larger outbound degree imply a higher streaming quality? To understand these questions, we ran experiments of different outbound degrees with network sizes of 10,000 peers and 20,000 peers respectively. Fig. 5 gives the answers. If all participants of the network grab outbound bandwidth greedily, that is, with a large possible outbound degree, the result is that there will be lots of request conflicts and that will exhaust the capacities of peers. Thus the streaming quality will degrade rapidly. Although peers with smaller outbound degrees will limit their accessible scope, it is indeed beneficial to the streaming quality of the whole streaming network, since it also reduces request conflicts. But with an exact outbound degree, it will also limit the accessible scope of peers which leads to limiting the ability of peers to handle request conflict and ill-reputation peers. That is why the smallest outbound degree can't bring the highest streaming quality.

3) Comparison of Layer Selection Strategies: In this set of experiments, we investigate the impact of layers selecting strategies. A layer selecting process will be performed when there is more than one candidate layer for a requested segment. Layer selection strategies been compared are Lowest Layer First(LLF), Highest Layer First(HLF), Random, and Least Recently Assigned(LRA), they all relate to outbound bandwidth only and we consider neither end-to-end delay nor underlining topology. As we can see from Fig. 6 the LRA layers selection strategy has the best streaming quality among all strategies and the LLF strategy has the worst streaming quality. The reason is that the LRA strategy distributes the consumed capacity loading among the accessible layers of a request peers and hence it avoids exhausting the resource of particular layers. Both LLF and HLF tend to consume capacity of the highest layer and lowest layer respectively and therefore they won't have a good streaming quality.

B. Scalability and Resilience to Peer Failures

Attractive features of the proposed streaming model are its scalability to network growing and robustness to peer failures. It is self-growing since requesting peers joining later will become sending peers; therefore the streaming network's total capacity will be amplified; it is self-adapting because detected failures of peers will cause requesting peers to switch their request target to seek more stable sources. These features make for a high and stable streaming quality in highly dynamic environments where peers disconnect or connect arbitrarily and frequently. The goal of this set of experiments is to attest to scalability and robustness of our streaming model.

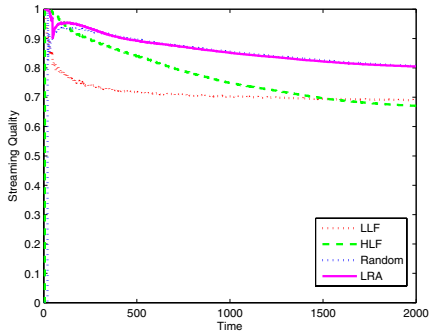


Fig. 6. Comparisons of Layer Assignment Strategy

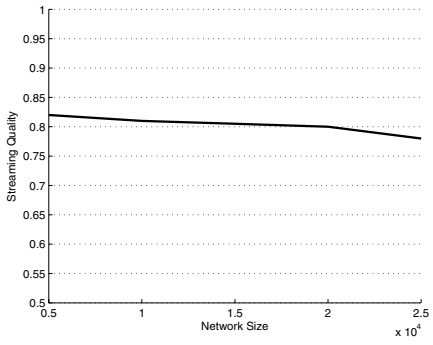


Fig. 7. Scalability

To evaluate scalability, we ran experiments of network with size growing from 5000 peers to 25,000 peers. Fig. 7 depicts the streaming quality versus network size. Realize that the total capacity of the network won't match the demanded capacity of all peers due to asymmetry between contributed outbound bandwidth and demanded inbound bandwidth of peers. For instance, a peer $\in \quad (R_\ell)$ will consume at least R_0 for playback of one segment but only contribute $\frac{R_0}{2^\ell}$ for delivering one segment. The streaming quality degrading is inevitable while the network grows rapidly. Thus the important issue of a good streaming system is that the streaming quality should degrade smoothly while the network grows rapidly. As show in Fig. 7, the streaming quality of our streaming model degrades gracefully and is almost stable. For this reason, we say the proposed streaming mode is with scalability.

In the robustness experiments, we ran experiments for network with size 15,000 peers and up to 20 percent peer failures. The result is shown in Fig 8. Although, the leave(failure) of peers will reduce request conflicts, they also lessen the capacity of the network. Therefore, the streaming quality goes down. But because there are duplicated peers for layers in a peer's outbound bandwidth layer, and each peer will choose peers with the best reputation for each accessible layer for requesting, the streaming quality won't get worst and it shows that our proposed model is resilience to peer failures.

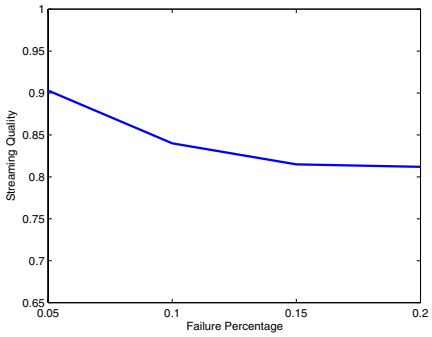


Fig. 8. Streaming Quality with peer failure

TABLE II

EXPERIMENTAL RESULTS OF FAIRNESS

Layer	Quality	Freshness	Loading	delay(T_0)
source	-	1	0.25	0
1	0.928	0.987	0.366	0
2	0.904	0.957	0.203	4
3	0.854	0.932	0.158	8
4	0.799	0.925	0.023	16
5	0.714	0	0.000	32

C. The Fairness

One promise of our streaming model is to keep fairness among peers. By fairness, we mean that peers with higher offered outbound bandwidth will have the benefit of higher streaming quality and have more live media data. Nevertheless, they also burden heavier loading for disseminating segments to other peers. Table II exhibit fairness exists in layers. The field of Freshness indicates the accuracy of the streaming direction. The value of this field shows the percentage of peers who have newer segments than peers in lower layer. It is clear that peers in layer 1 have the highest quality as well as shortest delay time but they have the heaviest loading. On the contrary, peers in the lowest layer have the worst streaming quality and longest delay time but their loading is negligible.

6. RELATED WORK

Several techniques have been proposed to address the problem of peer-to-peer media streaming by adopting application level multicast(ALM). In an ALM-based streaming system, a multicast tree is constructed for media delivering over the network and therefore how to build and maintain a multicast tree efficiently and with scalable control overhead is a critical issue. Nice[11] and Zigzag[10] both adopt hierarchical distribution trees in which peers are organized in a hierarchy of bounded-size clusters but are fundamentally different due to their own multicast tree construction and maintenance strategies. SpreadIt [12] builds a single distribution tree in which a joining process is done by traversing the tree nodes downward from the source until reaching a node that is unsaturated and could accommodate the request. A deleting process is

performed with a redirect process while a child detects the parent failure.

In layered media streaming, cumulative streaming and non-cumulative streaming are two possible approaches [14]. In the cumulative layering case, the media data is encoded in a base layer and one or more enhancement layers. Unlike the base layer which can be decoded independently, enhancement layers are decoded cumulatively such that layer k can only be decoded along with the layer 1 to the layer k-1. Receivers join at least one layer and add/drop other layers according to proposed strategies. In a non-cumulative streaming case, each layer is independent and the receiver need only subscribe to one layer. Receivers in [15] add and drop layers according to conditions on congestion and on spare capacity, respectively. PALS [16] allows requesting peers to orchestrate coordinated delivery by monitoring the overall throughput and periodically determining what is the target overall quality that can be delivered from all sending peers. K. Nahrstedt et al. [13] introduced requesting times of peers to determining a set of qualified sending peers for a requesting peer.

Much research has addressed the problems of the media data assignment and the peers selection. B. Bhargava. et al. [1] proposes an optimal media data assignment algorithm which leads to the minimum buffering delay for a requesting peer. Peers are classified into N classes according to the N possible values of their outbound bandwidth offer and data assignment is done under considering the available set of sending peer and the buffered size. B. Bhargava et al. [7] studies three possible peers selection techniques, namely random, end-to-end, and topology-aware with different goodness estimations for sending peers. Much research has addressed the problems of the media data assignment and the peers selection. B. Bhargava. et al. [1] proposes an optimal media data assignment algorithm which leads to the minimum buffering delay for a requesting peer. Peers are classified into N classes according to the N possible values of their outbound bandwidth offer and data assignment is done with considering the available set of sending peers and the buffered size. B. Bhargava et al. [7] studies three possible peer selection techniques, namely random, end-to-end, and topology-aware with different goodness estimations for sending peers.

7. Conclusion

In this paper, we study the peers selection problem and propose an outbound bandwidth based approach for peer-to-peer media streaming. The contributions of our work are:

- We proposed an outbound bandwidth based streaming model with properties of high streaming quality, robustness, scalability and reliability.
- We studied the problem of request conflicts and presented possible mechanisms for resolutions.
- We pointed out that grabbing outbound bandwidth greedily does not necessarily ensure high streaming quality. We showed by simulations that there exists a suitable outbound degree for the proposed streaming model.

This is our initial work for peer-to-peer media streaming since there are still several consideration we don't take into account but we have them in mind. For example, the influence of end-to-end delay, overlap of paths from multiple sources to receiver as mentioned in [7]. We also do not take advantage of topology to improve performance. These will be given consideration in our future works.

References

[1] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. *On peer-to-peer media streaming*. In Proc. of IEEE ICDCS'02, Vienna, Austria, July 2002.

[2] Peer-to-Peer Membership Management for Gossip-Based Protocols Ayalvadi J. Ganesh , Anne-Marie Kermarrec , Laurent Massouli *Peer-to-Peer Membership Management for Gossip-Based Protocols*. IEEE Transactions on Computers, v.52 n.2, p.139-149, February 2003.

[3] http://www.skype.com/

[4] Changxi Zheng, Guobin Shen, Shipeng Li. *Distributed Prefetching Scheme for Random Seek Support in Peer to Peer Streaming Applications*. In Proc. ACM P2PMMS'05, November 2005.

[5] Chuan Wu, Baochun Li. *Peer-to-peer tree construction: Optimal peer selection for minimum-delay peer-to-peer streaming with rateless codes*. In Proc. ACM P2PMMS'05, November 2005.

[6] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, Atul Singh. *SplitStream: high-bandwidth multicast in cooperative environments*. Proceedings of the nineteenth ACM symposium on Operating systems principles, October 19-22, 2003, Bolton Landing, NY, USA.

[7] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. *PROMISE: Peer-to-Peer Media Streaming Using CollectCast*. In Proc. of ACM Multimedia 2003, November 2003.

[8] Habib, A.; Chuang, J. *Incentive Mechanism for Peer-to-Peer Media streaming*. Quality of Service, 2004. IWQOS 2004. Twelfth IEEE International Workshop on 7-9 June 2004 Page(s):171 - 180.

[9] Meng Zhang, Li Zhao, Yun Tang, Jian-Guang Luo, Shi-Qiang Yang. *Large-Scale Live Media Streaming over Peer-to-Peer Networks through Global Internet* In Proc. of ACM P2PMMS'05, November 2005.

[10] D. A. Tran, K. A. Hua, and T. Do. *ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming*. In Proc. of IEEE INFOCOM 2003, March 2003.

[11] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. *Scalable Application Layer Multicast*. In Proc. of ACM SIGCOMM 2002, August 2002.

[12] H. Deshpande, M. Bawa, and H. Garcia-Molina. *Streaming Live Media over a Peer-to-Peer Network*, Technical report, Standford Database Group 2001-20, August 2001.

[13] Y. Cui and K. Nahrstedt, *Layered peer-to-peer streaming*. In Proc. of ACM NOSSDAV, 2003.

[14] T. Kim and M. Ammar. *A comparison of layering and stream replication video multicast scheme*. In International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2001.

[15] S. McCanne, V. Jacobson and M. Vetterli. *Receiver-driven layered multicast*. In ACM SIGCOMM, 1996.

[16] R. Rejaie and A. Ortega. *PALS: Peer to peer adaptive layered streaming* in NOSSDAV, June 2003.