社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2004－AVM－47（1）
2004／12／9

MANETにおけるP2Pファイル共有のための最適インデックスサーバ数

太田　能[†]　Zihui Ge[††]　Yang Guo[†††]　JimKurose[†††]

† 神戸大学工学部 〒657-8501 神戸市灘区六甲台町1－1
†† AT&T Labs - Research Florham Park, NJ 07932-0971, USA
††† University of Massachusetts at Amherst Amherst, MA 01003-9264, USA
E-mail: †c-ohta@cs.kobe-u.ac.jp, ††gezihui@att.research.com, †††{yguo,kurose}@cs.umass.edu

あらまし　本稿では，モバイルアドホックネットワーク（MANET）におけるピア・ツー・ピアファイル共有のための二つの基本的な情報検索方式を比較している．インデックスサーバを利用することで，インデックスサーバ内のファイルインデックスを高速に検索することが可能になるが，キャッシュの一貫性を保つためにオーバヘッドが生じる．本稿では，二つのインデックスサーバキャッシュ方式（CC（Consistent Caching），LC（Local Caching））とフラッディング方式の比較をおこなった．本稿では，インデックスサーバ方式により，検索のオーバヘッドがフラッディング方式に比べてどのくらい削減できるのかを明らかにするとともに，最適インデックスサーバ数が，ネットワークサイズ，キュエリレート，インデックス生成レートに従ってどのように変化するかを調査した．比較にあたっては，二つのキュエリ方式，HQ（History Query）とLQ（Latest Query）を対象とした．数値結果から，ネットワークサイズ，インデックス生成レート，キュエリレートに応じて，CCとLCを選択すべきであることがわかった．
キーワード　ピア・ツー・ピア，インデックスサーバ，モバイルアドホックネットワーク

Index-Server Optimization for P2P File Sharing in Mobile Ad Hoc Networks

Chikara OHTA[†], Zihui GE[††], Yang GUO[†††], and Jim KUROSE[†††]

† 1-1 Rokkoudai, Nada, Kobe, Hyogo 657-8501, Japan
†† AT&T Labs - Research Florham Park, NJ 07932-0971, USA
††† University of Massachusetts at Amherst Amherst, MA 01003-9264, USA
E-mail: †c-ohta@cs.kobe-u.ac.jp, ††gezihui@att.research.com, †††{yguo,kurose}@cs.umass.edu

Abstract　In this paper, we compare two basic approaches towards providing peer-to-peer file-sharing (or more generally, information search) in mobile ad-hoc networks (MANETs). The use of index servers presents the possibility of locating a file index quickly in an index server cache, but requires additional overhead to maintain cache consistency. We compare the performance of the flooding approach to two index-server caching approaches: consistent caching and local caching. We quantify the reduction in search overhead using the index-server scheme rather than flooding in MANETs, and study how the optimal number of index servers varies according to network size, query rate, and index generation rate. We compare the flooding scheme and the consistent caching and local caching schemes, for two types of queries: history queries and latest queries. Numerical results show how one can choose between the alternatives of consistent caching and local caching depending on network size, index generation rate and query rate.
Key words　peer-to-peer, index server, mobile ad-hoc networks

1. Introduction

Mobile ad-hoc networks (MANETs) are self-organizing networks that consist of mobile nodes with routing and forwarding functions. In MANETs, two nodes can communicate with each other with the assistance of the other nodes' forwarding function even if these two nodes can not communicate directly (over a single hop) with each other. Thus, MANETs provide the freedom of mobility, and enable nodes to be quickly deployed in the field without any infrastructure [5]. Peer-to-peer (P2P) systems such as Gnutella [2], KaZaA [9],

and Freenet [6] have been developed to provide distributed file and information sharing. These systems are tolerant to failure of some nodes, since each node can act as both a server and a client. Such a distributed architecture enables nodes to share information to the greatest extent possible, even if the network is segmented due to node mobility. Thus P2P file sharing has great potential as a technology upon which one can build numerous MANET applications.

A fundamental problem in P2P file sharing is to determine where a needed file (or piece of information) is located. Thus nodes collect indices of which nodes hold which files. But given that a node needs to determine where a file is stored, how can it do so? Flooding is perhaps the simplest approach, although it incurs the overhead of involving every node in the search (since searches are broadcast to all network nodes). An alternative is to introduce additional servers, known as *index servers,* into the MANET. Index servers cache directory information (i.e., indices) about which nodes have which files. With index servers, a node wishing to locate a file first queries its local index server, which then queries other index servers, as needed. The use of index servers presents the possibility of locating a file index quickly in an index server cache, but requires additional overhead to maintain cache consistency. In this paper we investigate and compare the performance of the flooding approach to approaches based on index servers. We note that while we will refer to the object being searched for as a "file," there is no difference between searching for a file name, or an arbitrary piece of information that is associated with a node. In either case, the goal is to determine where the file, or piece of information is stored, after which the file/information can be retrieved. We will use the term "file" throughout this paper for ease of exposition, but stress that the techniques we study are equally applicable to searches for general pieces of information as well.

Our index-server system can be considered as a hybrid P2P system as KaZaA, which has client nodes as well as super-nodes [9]. A number of P2P research works have studied super-node systems (e.g. [7], [10]) in wired networks. In this work, we address the following questions *in the context of a mobile ad hoc network:* 1) To what extent does the index-server approach reduce search overhead in comparison with the flooding approach? 2) What forms of caching are most effective in the index-server approach? 3) What is the optimal number of index servers that minimizes search overhead?

In this paper, we analyze the search overhead of the flooding scheme and two different index-server caching schemes. We consider two types of queries: history queries and latest queries. We also obtain expressions for the optimal number of index servers, and show how the optimal number of index servers varies according to network size, query rate, and index generation rate.

This paper is organized as follows: Section 2. describes the flooding scheme and two types of caching for the index-server scheme. In Section 3., we analyze the cost (total amount of search overhead) for six combinations of three schemes (flooding scheme and two caching schemes of index-server) and two types of queries. We derive expressions for the optimal number of index servers. In Section 4., we present numerical results. Finally, we conclude in Section 5..

2. System Description

Fig. 1 shows the sequence of actions involved in file retrieval using the flooding scheme. Each node issues a query which includes the description of its user's interest. A query is flooded into the whole network using a simple scheme: if a node receives the query for the first time, it forwards it to the neighbors, otherwise it discards the query. If the node has any files that match the query, it replies to the query originator with index information about the file. For example, the index might contain the location and other characteristics of the file. If the original node needs actual files, it initiates a file request to each file holder. Each requested node replies by sending the file corresponding to the request.

In our index-server system, we assume that index servers are selected among all of the nodes according to several conditions, such as the size of network, query rate and index generation rate. In this paper, we refer to non-server nodes as clients. A client issues a query to its nearest index server. The nearest index server acts as a proxy to resolve queries on behalf of the client, and to respond to queries from other clients and other index servers. When a node (or index-server) generates any new files, it registers their indices with its nearest index server (or itself). Each index server caches indices. In this paper, we consider two types of caching: consistent caching and local caching.

Consistent caching: Each index server tries to maintain consistent indices among the index servers. Fig. 2 shows the sequence of events involved in file retrieval. If an index server receives new or updated index registrations from a client in its area, the index server immediately forwards the new/updated index to other index servers over the spanning tree that connects all index servers. Consequently, every index server has the same set of indices. The nearest index server to the query originator replies to the querier with the indices that match the query.

Local caching: Each index server acts as a query proxy on behalf of the clients in its area. Fig. 3 shows the sequence of events

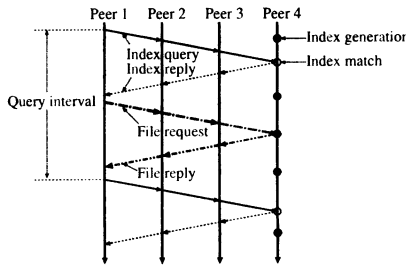

Figure 1 Time diagram of flooding-reply scheme

involved in file retrieval. An index server receiving a query from a client in its area forwards the query to other index servers. If an index server has indices that match the query, it replies by sending the indices to the proxy index server. After collecting the indices, the proxy index server forwards the indices to the query originator.

In the following analysis, we will consider two types of queries: history queries and latest queries.

History query: A node is interested in all information (subject to some criteria) including every updates after the previous query. We refer to the type of query for such information as a "history query." For instance, a node may need to know the exact fluctuation of temperature over time in a certain place.

Latest query: A node is only interested in the latest information at the moment of a query. We refer to the type of query that requires the present information as a "latest query". For example, a node may want to know the current temperature in a certain place. In this case, the latest index will be returned even if the temperature has not changed since the previous query.

3. Analysis

3.1 Motivation of Model

We are seeking a fundamental understanding of the system performance under different query schemes and various system parameters (e.g., network size, query rate, index generation rate). We note that once a querying node has received the results (indices) of its query, the subsequent file requests and replies are the same for the flooding scheme and the two kinds of index-server schemes. Therefore, to evaluate and compare these systems, we only need to focus on the message overhead associated with resolving a query.

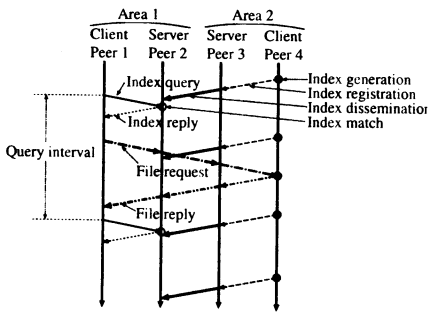

Figure 2 Time diagram of consistent caching of index-server scheme

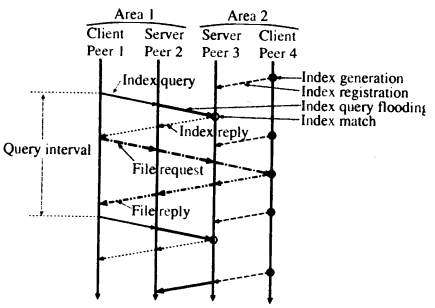

Figure 3 Time diagram of local caching of index-server scheme

In order to build a simple model to understand the performance tradeoffs in terms of search overhead and compare the three query schemes, we consider a simple grid network topology and assume uniform and static query rate and index generation rate.

Although we do not consider dynamic topology and uneven (both spatially and temporally) query and index generation rate, our analysis can still provide a good insight in such scenarios. In particular, since an index server can observe the statistics of local query rate and index generation rate, it can compute the optimal local index-server-density (as we will see in Section 3.4.6), which forms the basis of a dynamic index-server election/placement protocol (e.g., by adapting a distributed self-stabilizing algorithm in [4]).

3.2 Model Description

We consider an $n \times n$ grid topology, where on each lattice point of the grid sits a node (as shown in Fig. 4). Thus, there are a total of $N(= n \times n)$ nodes in the network. Each node can only communicate with its adjacent nodes directly. In index-server schemes, we assume that the network can be equally divided into $M(= m \times m)$ areas. An index server is placed at the central lattice point of each area. This index server serves all clients in the area, including itself.

Each node in the network generates index queries at a rate of μ bit/s, where μ can be considered as the product of the average number of query packets per second and the length of query messages in bits. New files and thus new file indices are generated on each node. The rate that a new file index is produced at one node is λ bit/s, where λ is the product of the mean number of files generated per second and the mean index length in bits. We denote the ratio of λ to μ by $\rho = \lambda/\mu$.

We define p as the probability that a file index generated in the last query interval matches a given history query. We define k as the coefficient such that $k\mu$ represents the mean rate of query replies between a query replier to a query originator in latest queries. The value of k can be considered as the mean number of file sources that match a latest query multiplied by the ratio of the file-index-length to the query-message-length.

We summarize the notation as follows:

- $N = n \times n$: the number of nodes
- $M = m \times m$: the number of index-servers (the number of areas)
- μ bit/s: mean query rate per node
- λ bit/s: mean index generation rate per node
- $\rho = \lambda/\mu$: ratio of λ to μ
- p: probability that an object generated during a query interval matches a history query.
- k: the coefficient such that $k\mu$ is the mean rate of query replies between a query replier to a query originator in latest queries.

3.3 Flooding Scheme

3.3.1 Flooding Cost

We assume that each node can flood a query to its adjacent neighbors using a link-level broadcast mechanism. Thus, each node only needs to broadcast a query once – either as the initiator when a query is generated or as a forwarder when it receives a flooded query for the first time. Since each node generates queries at rate μ, the overall query message rate is μN. Thus, the total flooding cost is

$$C_{\text{flooding}} = \mu N^2. \tag{1}$$

3.3.2 Reply Cost

a) History Query

Recall that each node generates new indices or updates existing indices at the index generation rate, λ. Thus, the mean rate of query replies in responding to one query originator from any one of the $N-1$ repliers (excluding the query originator) is $p\lambda$. Furthermore, the number of hops that a query-reply message travels on average (among the $N(N-1)$ number of possible originator-replier pairs) is $\frac{2}{3}\sqrt{N}$ [11]. The total reply cost C_{reply} can thus be computed by

$$C_{\text{reply}} = \frac{2}{3}p\lambda N^{\frac{3}{2}}(N-1). \tag{2}$$

b) Latest Query

By definition, the mean rate of query replies in responding to one query originator from any one of the $N-1$ repliers is $k\mu$. Thus, the total reply cost C_{reply} is

$$C_{\text{reply}} = \frac{2}{3}k\mu N^{\frac{3}{2}}(N-1). \tag{3}$$

3.3.3 Total Cost

a) History Query

By summing up (1) and (2), we have the total cost C_{total} as

$$C_{\text{total}} = \mu\left\{N^2 + \frac{2}{3}p\rho N^{\frac{3}{2}}(N-1)\right\}. \tag{4}$$

b) Latest Query

By summing up (1) and (3), we have the total cost C_{total} as

$$C_{\text{total}} = \mu\left\{N^2 + \frac{2}{3}kN^{\frac{3}{2}}(N-1)\right\}. \tag{5}$$

3.4 Consistent Caching Scheme

3.4.1 Registration Cost

When a new file is generated on a client node, the client needs to register this index with its nearest index server. We assume that the index server is located in the center of its area(注1). As shown in [11], the average hop count from a node to its index-server in the area, $\overline{h}_{\text{server-client}}$, is

$$\overline{h}_{\text{server-client}} = \frac{1}{2}\sqrt{\frac{N}{M}}. \tag{6}$$

Since the number of client nodes in the network is $N-M$ and the index generation rate per node is λ, the total registration cost $C_{\text{registration}}$ is

$$C_{\text{registration}} = \frac{1}{2}\lambda(N-M)\sqrt{\frac{N}{M}}. \tag{7}$$

3.4.2 Query Cost

Each node sends queries to its nearest index-server at rate μ. Similar to the registration cost, we can compute the query cost C_{query} as

$$C_{\text{query}} = \frac{1}{2}\mu(N-M)\sqrt{\frac{N}{M}}. \tag{8}$$

3.4.3 Reply Cost

a) History Query

As described in Section a), the mean rate of query replies in responding to one query originator from the other $N-1$ nodes is $p\lambda(N-1)$. Each query reply travels $\overline{h}_{\text{server-client}}$ hops on average from an index server to a client in the area. Thus the total reply cost C_{reply} is

$$C_{\text{reply}} = \frac{1}{2}p\lambda(N-1)(N-M)\sqrt{\frac{N}{M}}. \tag{9}$$

b) Latest Query

Similar to the case of history queries, the mean rate of query replies in responding to one query originator from the rest of the nodes is $k\mu(N-1)$. Thus, the reply cost C_{reply} is

$$C_{\text{reply}} = \frac{1}{2}k\mu(N-1)(N-M)\sqrt{\frac{N}{M}}. \tag{10}$$

3.4.4 Index-distribution Cost

Index-servers are logically connected to each other through a minimum spanning tree, where there are $(M-1)$ "links" and each "link" consists of $\sqrt{N/M}$ number of physical hops on average (since the index servers are evenly distributed in the network). When an index is generated/updated on an index server – by either the index server node itself or a client in the area, this index is disseminated over the spanning tree to other index servers. This means each index need to be propagated $(M-1)\sqrt{N/M}$ hops on average in order to reach all index servers. Since there are N nodes, and each node registers indices at rate λ, by multiplying them, we have the index-distribution cost $C_{\text{distribution}}$ as

$$C_{\text{distribution}} = \lambda N^{\frac{3}{2}}\frac{M-1}{\sqrt{M}}. \tag{11}$$

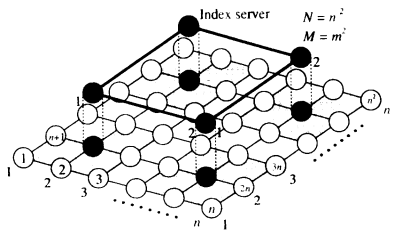

Figure 4 Topology Model

(注1) : For the area that includes node 1, the index sever is set at $(\frac{n-m}{2m}, \frac{n-m}{2m})$. For simplicity, we treat $(\frac{n-m}{2m})$ as a real number although it may not be an integer.

3.4.5 Total Cost

a) History Query

By summing (7) through (9) and (11), we have the total cost C_{total} as

$$C_{\text{total}} = \mu\left[\frac{1}{2}\{\rho+1+p\rho(N-1)\}(N-M)\sqrt{\frac{N}{M}} + \rho N^{\frac{3}{2}}\frac{M-1}{\sqrt{M}}\right]. \quad (12)$$

b) Latest Query

By summing (7), (8), (10) and (11), we have the total cost C_{total} as

$$C_{\text{total}} = \mu\left[\frac{1}{2}\{\rho+1+k(N-1)\}(N-M)\sqrt{\frac{N}{M}} + \rho N^{\frac{3}{2}}\frac{M-1}{\sqrt{M}}\right]. \quad (13)$$

3.4.6 Optimal Number of Index-Servers

In this section, we derive the expression for the optimal number of index servers, M^*, such that the total cost is minimized when consistent caching scheme is used.

a) History Query Case

Note that as M increases from 1 to N, the first term of (12) monotonically decreases while the second term monotonically increases. In the sense that the total cost C_{total} is minimized, the optimal value M^* can be obtained by solving the derivative of $\frac{dC_{\text{total}}}{dM} = 0$. Let

$$\gamma_1 = N\frac{\rho\{p(N-1)-1\}+1}{\rho\{N+(N-1)(1-p)\}-1}. \quad (14)$$

We have $M^* = \gamma_1$ if $1 \leqq \gamma_1 \leqq N$. In the case of $\gamma_1 < 1$ or $\gamma_1 > N$, if $C_{\text{total}}|_{M=1} \leqq C_{\text{total}}|_{M=N}$, $M^* = 1$; otherwise, $M^* = N$.

Taking a closer look at (14), we find that when $1 \ll N$,

$$M^* \simeq \max\left\{1, \frac{p}{2-p}N\right\} \quad (15)$$

Thus, M^* increases proportionally with N, suggesting that the optimal density of the index servers is independent of the network size when the network is sufficiently large.

Furthermore, taking limit on γ_1, we have

$$M^* \to \frac{N\{p(N-1)-1\}}{N+(N-1)(1-p)} \quad \text{as} \quad \rho \to \infty, \quad (16)$$

which corresponds to a system with a very large indices generation rate in comparison with the query rate.

b) Latest Query

Apply the above technique in latest query scheme, we obtain the optimal number of index servers M^* as follows:

$$\gamma_2 = N\frac{k(N-1)-\rho+1}{\rho(2N-1)-k(N-1)-1}, \quad (17)$$

$M^* = \gamma_2$ if $1 \leqq \gamma_2 \leqq N$. In the case of $\gamma_2 < 1$ or $\gamma_2 > N$, if $C_{\text{total}}|_{M=1} \leqq C_{\text{total}}|_{M=N}$, $M^* = 1$; otherwise, $M^* = N$.

When $1 \ll N$, we have

$$M^* \simeq \max\left\{1, \frac{k}{2\rho-k}N\right\}, \quad (18)$$

indicating that M^* also increases according to $O(N)$. Furthermore, taking limits on γ_2, we have

$$M^* \to 1 \quad \text{as} \quad \rho \to \infty. \quad (19)$$

3.5 Local Caching Scheme

3.5.1 Registration, Query, and Reply Costs

For the local caching scheme, the registration cost, $C_{\text{registration}}$, the query cost, C_{query}, and the reply cost, C_{reply}, are the same as those of the consistent caching scheme.

3.5.2 Server Flooding Cost

A query message, once received at an index server, is "flooded" among the logical index server network. That is, when an index server receives a "flooded" query from another index server for the first time, it forwards the query to all adjacent index servers except the one from which it has received the query. Thus, each query should eventually travels all logical links in the index server network, where there are $2(M-\sqrt{M})$ links in total and each virtual link corresponds to $\sqrt{N/M}$ physical hops on average. Since the query rate on each node is μ. the total flooding cost $C_{\text{server-flooding}}$ is

$$C_{\text{server-flooding}} = 2\mu N^{\frac{3}{2}}(\sqrt{M}-1). \quad (20)$$

3.5.3 Server Reply Cost

Similar to that of the flooding scheme in Section a), the number of logical hops that a reply message travels on average (among all pairs of originator and replier) in the $m \times m$ index server network is $\frac{2}{3}\sqrt{M}$ [11]. Since each logical hop corresponds to $\sqrt{N/M}$ number of physical hops, we have the average reply distance in physical hop counts as

$$\overline{h}_{\text{server-reply}} = \frac{2}{3}\sqrt{N}. \quad (21)$$

a) History Query

Each node generates indices and registers the indices at its index server at rate λ. Thus, each index server receives registrations from the clients (and itself) in its area at rate $\lambda\frac{N}{M}$. The reply rate from any one of the index servers to the proxy index server (to which a query is first directed) is $p\lambda\frac{N}{M}$ on average. The number of possible pairs of proxy-replier index servers is $M(M-1)$. The mean length of reply path is $\frac{2}{3}\sqrt{N}$. Therefore, the total reply cost $C_{\text{server-reply}}$ is

$$C_{\text{server-reply}} = \frac{2}{3}p\lambda N^{\frac{3}{2}}(M-1). \quad (22)$$

b) Latest Query

By definition, the reply rate from a node is $k\mu$. Since $\frac{N}{M}$ nodes exist in each area and all indices in the area that match a query are replied to the proxy index server, the reply rate from any one of the

index servers to the proxy index server is $k\mu\frac{N}{M}$ on average. Similar to the above history query case, we have the total reply cost $C_{\text{server-reply}}$ as

$$C_{\text{server-reply}} = \frac{2}{3}k\mu N^{\frac{3}{2}}(M-1). \quad (23)$$

3.5.4 Total Cost

a) History Query

By summing (7) through (9), (20) and (22), we have the total cost C_{total} as

$$C_{\text{total}} = \mu\left[\frac{1}{2}\{\rho+1+p\rho(N-1)\}(N-M)\sqrt{\frac{N}{M}} + 2N^{\frac{3}{2}}(\sqrt{M}-1)+\frac{2}{3}p\rho N^{\frac{3}{2}}(M-1)\right]. \quad (24)$$

b) Latest Query

By summing (7), (8), (10), (20) and (23), we have the total cost C_{total} as

$$C_{\text{total}} = \mu\left[\frac{1}{2}\{\rho+1+k(N-1)\}(N-M)\sqrt{\frac{N}{M}} + 2N^{\frac{3}{2}}(\sqrt{M}-1)+\frac{2}{3}kN^{\frac{3}{2}}(M-1)\right]. \quad (25)$$

3.5.5 Optimal Number of Index-Servers

Now we derive the expression for the optimal number of index servers, M^*, when local caching scheme is used. As we did in Section 3.4.6, we first seek a solution for $\frac{dC_{\text{total}}}{dM}=0$.

For both the history query scheme and the latest query scheme, we obtain a cubic equation in the following form:

$$x^3+ax^2+b=0, \quad (26)$$

where $x=\sqrt{M^*}$.

If (26) has a real solution x_{R} satisfying $1 \leqq x_{\text{R}} \leqq \sqrt{N}$, then the optimal value M^* is given by $M^*=\sqrt{x_{\text{R}}}$ (unless $C_{\text{total}}|_{M=1} \leqq C_{\text{total}}|_{M=\sqrt{x_{\text{R}}}}$ or $C_{\text{total}}|_{M=N} \leqq C_{\text{total}}|_{M=\sqrt{x_{\text{R}}}}$). For the solution of (26), please refer to [3].

a) History Query

For the history query scheme, we have a and b in (26) as

$$a = \frac{3}{8p}\left\{\frac{4}{\rho}-p-\frac{1}{N}\left(1+\frac{1}{\rho}-p\right)\right\}, \quad (27)$$

$$b = -\frac{3}{8p}\left(1+\frac{1}{\rho}-p+pN\right). \quad (28)$$

From (26), (27) and (28), we have

$$M^* \simeq \left(\frac{3}{8}N\right)^{\frac{2}{3}} \quad \text{as} \quad 1 \ll N. \quad (29)$$

Thus, M^* increases according to $O(N^{\frac{2}{3}})$. Furthermore, we have

$$M^* \to \left(\tilde{A}^{\frac{1}{3}}+\frac{\tilde{a}^2}{9\tilde{A}^{\frac{1}{3}}}-\frac{\tilde{a}}{3}\right)^2 \quad \text{as} \quad \rho\to\infty, \quad (30)$$

where

$$\tilde{a} = -\frac{3}{8}\cdot\frac{1+p(N-1)}{pN}, \quad (31)$$

$$\tilde{A} = \sqrt{\frac{\tilde{a}^4}{27}N+\frac{\tilde{a}^2}{4}N^2}+\left|\frac{\tilde{a}^3}{27}+\frac{\tilde{a}}{2}N\right|. \quad (32)$$

b) Latest Query

For the latest query scheme, we have a and b in (26) as follows:

$$a = \frac{3}{8p}\left(\frac{4}{k}-1-\frac{\rho+1}{k}+\frac{1}{N}\right), \quad (33)$$

$$b = -\frac{3}{8p}\left(\frac{\rho+1}{k}+N-1\right). \quad (34)$$

From (26), (33) and (34), we have

$$M^* \simeq \left(\frac{3}{8}N\right)^{\frac{2}{3}} \quad \text{as} \quad 1 \ll N. \quad (35)$$

Thus, M^* increases according to $O(N^{\frac{2}{3}})$. Furthermore, we have

$$M^* \to N \quad \text{as} \quad \rho\to\infty. \quad (36)$$

4. Numerical Results

In the following, we vary the value of ρ with the value of μ fixed. This is because the rest of C_{total} factorized by μ can be represented using ρ without μ in all cases.

4.1 Total Cost Characteristics

In the following numerical examples, we computed and used the corresponding optimal number M^* for each index server, and we set $N=100$, $\mu=256\times 8/30$ bit/s, $p=1.0$ and $k=1.0$.

Figs. 5 and 6 show characteristics of total cost for the cases of history queries and latest queries, respectively. In both cases, consistent caching provides lower cost than local caching, when the value

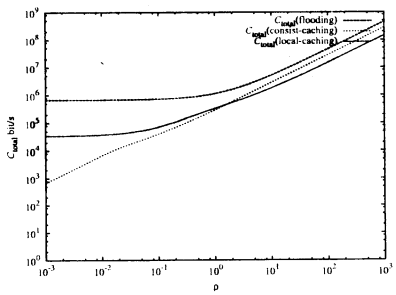

Figure 5 Characteristics of total cost of flooding, consistent caching and local caching in history query scheme as function of the value of ρ. ($N=100, \mu=256\times 8/30$ bit/s, $k=1.0$)

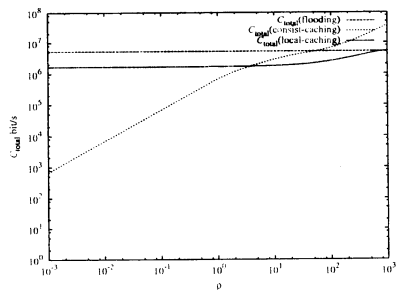

Figure 6 Characteristics of total cost of flooding, consistent caching and local caching in latest query scheme as function of the value of ρ. ($N=100, \mu=256\times 8/30$ bit/s, $k=1.0$)

of ρ is small. This relationship is reversed in case of high value of ρ. We will discuss the crossover point in Section 4. 2. Note that, in the latest query case, the cost of flooding scheme is constant for any ρ with μ fixed as shown in (5). In this case, the consistent caching scheme has higher cost than even the flooding scheme, as well as the local caching scheme in the case of a high value of ρ. This is because the index-distribution cost brings heavy burden. The cost of local caching approaches to that of flooding scheme as the value of ρ increases. In case of higher value of ρ, as shown in the following, the optimal number M^* of index-servers reaches the number of nodes (N), where server flooding cost in the local caching scheme coincides with flooding cost in flooding scheme.

4.2 Crossover Point between Consistent and Local Caching

Figs. 7 and 8 plot the ratio of the cost of consistent caching to that of local caching for history queries and for latest queries, respectively, in cases of $N = 10, 100, 1,000$ and $10,000$. Note that in case where the ratio is greater than one, consistent caching has a higher cost than local caching. The crossover point varies widely according to the value of N in the case of a history query, while such variation is relatively small in the case of a latest query. This is because, in the latest query scheme, regardless of index generation rate, only the latest indices are retrieved. Thus, the effect of index generation rate is relatively small, and the value of total cost mainly depends on the number of nodes – N. On the other hand, for the history query, the total cost depends on the values of both λ and N.

4.3 Optimal Number of Index-servers

Figs. 9 and 10 show the optimal number M^* of index-servers for history queries and latest queries, respectively, with the consistent caching scheme. In the history query scheme, the optimal number M^* converges to an asymptotic value as shown in (15), while it reaches one in the latest query scheme as shown in (16). This can be explained as follows: in this scheme, a smaller number of index-servers can bring a smaller index-distribution cost but higher other costs. Note that the reply cost of the history query is dependent on the value of the index generation rate, while that of the latest query is independent of this value. In case of a higher index generation rate λ, the reply cost is negligible in the latest query scheme, while it is still dominant in the history query scheme. Therefore, suppressing index-distribution costs reduces the total cost so that the value of M^* reaches to one in latest caching scheme. On the other hand, the smaller number of index-servers brings more reply cost so that M^* converges to an asymptotic value.

In the history query scheme, the optimal number M^* converges to an asymptotic value as shown in (30), while it reaches to N in latest query scheme as shown in (36). We can have a similar discussion to consistent caching for this reason, but we omit it due to

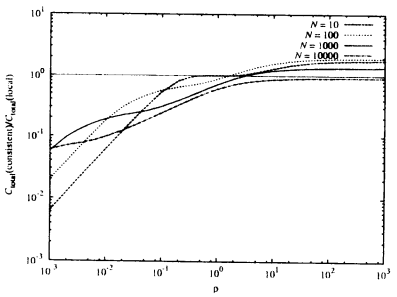

Figure 7 Characteristics of ratio of total cost of consistent caching to that of local caching in history query scheme as functions of the number of nodes and the value of ρ. ($\mu = 256 \times 8/30$ bit/s, $p = 0.1$)

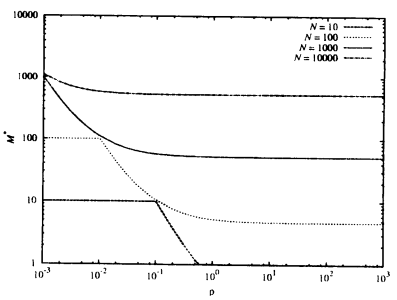

Figure 9 Characteristics of optimal number M^* of index-serves in case of consistent caching scheme and history query scheme as functions of the value of ρ. ($\mu = 256 \times 8/30$ bit/s, $p = 0.1$)

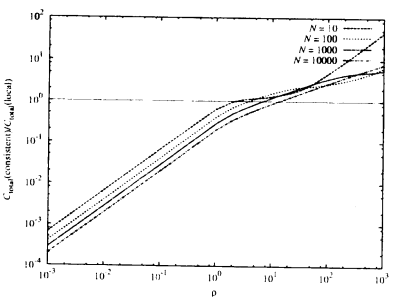

Figure 8 Characteristics of ratio of total cost of consistent caching to that of local caching in latest query scheme as functions of the number of nodes and the value of ρ. ($\mu = 256 \times 8/30$ bit/s, $k = 1.0$)

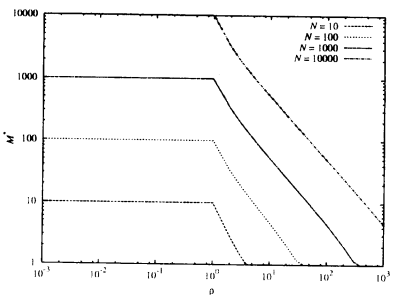

Figure 10 Characteristics of optimal number M^* of index-serves in case of consistent caching scheme and latest query scheme as functions of the value of ρ. ($\mu = 256 \times 8/30$ bit/s, $k = 1.0$)

limited space.

We confirm that the optimal numbers of index-servers given in Sections 4. 3 and 3. 4. 6 coincide to those by Brent method which is one of numerical analysis methods [12].

5. Conclusions

In order to explore the feasibility of an index-server approach for P2P file sharing in MANETs, we analyzed a flooding scheme and two index-server schemes (consistent caching and local caching) for history queries and latest queries from the viewpoint of search overhead on a grid topology. Our results show that we should carefully choose the alternative of consistent caching or local caching depending on the size of the network, the index generation rate and the query rate.

As future work, we will construct a protocol to select index servers dynamically using the results of this paper.

Acknowledgement The part of this research work was supported by the Ministry of Education, Culture, Sports, Science and Technology, Japan, Grant-in-Aid for Young Scientists (B), No. 16700066, 2004.

References

[1] "The gnutella protocol specification v0.4," "http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf."

[2] Gnutella webpage, "http://gnutella.wego.com."

[3] J. W. Harris and H. Stocker, Handbook of mathematics and computational science, Springer, 1998.

[4] B. -J. Ko and D. Rubenstein, "Distributed, self-stabilizing placement of replicated resources in emerging networks," IEEE ICNP 2003, pp.6–15, 2003.

[5] S. Corson and J. Macker, "Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations," IETF RFC2501, Jan. 1999.

[6] Freenet webpage. "http://www.freenetproject.org."

[7] B. Yang and H. Garcia-Molina, "Designing an Super-Peer Network," Proc. of IEEE ICDE 2003, March 2003.

[8] Y. Yi, X. Hong, and M. Gerla, "Scalable team multicast in wireless ad hoc networks exploiting coordinated motion," Proc. of Fourth International Workshop on Networked Group Communication, Oct. 2002.

[9] KaZaA webpage, "http://www.kazaa.com."

[10] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser, "Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks," Proc. of the 12th International World Wide Web Conference, May 2003.

[11] C. Ohta , Z. Ge , Y. Guo . and J. Kurose, "Index Server Optimization for P2P File Sharing in Mobile Ad Hoc Networks," University of Massachusetts, CMPSCI Technical Report, TR04-07, 2004.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, Numerical Recipes in C (2nd edition), Cambridge University Press, 1992.