Wireless Pers Commun (2017) 97:4087–4109
DOI 10.1007/s11277-017-4714-1

Cooperative Caching for Efficient Data Search in Mobile P2P Networks

Kyoungsoo Bok[1] · **Jaegu Kim**[1] · **Jaesoo Yoo**[1]

Published online: 3 August 2017

Abstract In this paper, we propose a cooperative caching scheme for structured data via clusters based on peer connectivity in mobile P2P networks. In the proposed scheme, a cluster is organized for cache sharing among mobile peers with long-term connectivity, and metadata are disseminated to neighbor peers for efficient data search performance. It reduces data duplication and uses cache space efficiently through integrative cache management of peers inside the cluster. The proposed scheme reduces data replacement time in the event of changes in topology or cache data replacement using the concept of temp cache. It performs data recovery and cluster adjustment through cluster management in the event of an abrupt disconnection of a peer. In the scheme, metadata of popular data are disseminated to neighbor peers for efficient data searching. In a data search, queries are processed in the order of local cache, metadata, the cluster to which it belongs, and neighbor clusters, in accordance with cooperative caching strategy. Performance evaluation results show that the proposed scheme has a higher cache hit ratio, and lower cost for data replacement and query processing than existing schemes.

Keywords Mobile P2P · Cooperative cache · Cluster · Structured data · Temp cache

✉ Jaesoo Yoo
yjs@chungbuk.ac.kr

Kyoungsoo Bok
ksbok@chungbuk.ac.kr

Jaegu Kim
jgns0101@naver.com

[1] Department of Information and Communication Engineering, School of Information and Communication Engineering, Chungbuk National University, Chungdae-ro 1, Seowon-Gu, Cheongju, Chungbuk 28644, Korea

1 Introduction

Since Client/Server (C/S) architecture consists of a server that provides a service and a client that receives it, it cannot provide normal service in the event that a server failure or bottleneck occurs. In addition, it not only increases server loads but problems occur with scalability, reliability, and flexibility when there are more users. To address these issues, peer-to-peer (P2P) networks for each peer to share and use necessary resources autonomously have been proposed. In a P2P network, peers simultaneously function as both client and server to each other without the notion of a separate client or server [1, 2]. It takes a form in which peers share resources with one another through equivalent horizontal networks. With recent improvements in performance of mobile devices and communications, studies have been conducted on mobile P2P which combines P2P networks and the mobile environment [3–5].

Mobile P2P allows functions including file sharing, content transfer, social networking, and advertising among mobile devices, using short-range wireless communications such as IEEE 802.11, Bluetooth, and Ultra Wide Band (UWB) [3, 6]. While wired P2P networks allow stable communications by ensuring reliability, mobile P2P networks cannot guarantee network reliability due to frequently changing connections, and mobile devices have limitations in data access due to their limited power source and small storage space [3, 5, 7–9]. These limitations make it difficult to apply P2P technology used in wired networks to mobile P2P networks [7, 10–12].

To improve performance data sharing and search among mobile devices in mobile P2P networks, caching methods are used [13, 14]. Caching is a strategy to reduce disc I/O and access time by storing frequently used resources, or resources to use later, to provide fast replies to data requests [14]. Existing caching schemes hold data redundantly because they manage caches independently without considering the condition of neighbor peers. This makes popular data access easier, but reduces cache utilization in the overall network, resulting in a decreased cache hit ratio and wasted storage space [15–18]. Therefore, to improve performance in data access and search, cooperative caching is required, which not only uses one's own local cache but also shares the caches of neighbor mobile devices to process user requests. This cooperative caching increases the cache hit ratio as well as reducing the cost for communications among peers and battery consumption.

In recent years, studies have been conducted to investigate data processing schemes using cooperative caching in mobile P2P networks [15, 17, 19–24]. In an attempt to improve cluster formation and data utilization, Chow et al. [25] has defined a tightly-coupled group (TCG) as an aggregate that exhibits data preference similar to the movement patterns of peers. Caetano and Bordim [26] proposed a cluster configured with peers to maintain cached data jointly. These schemes increase cache diversity and hit ratio by joint maintenance and management of popular data in shared caches. Existing caching schemes, however, have no data exchange for clusters or popular data in the event of a change in topology in a mobile P2P network [16, 17]. As a result, when a peer leaves or connects to a cluster, duplication and loss of popular data occurs. Therefore, in mobile networks where topology changes frequently, data loss and additional communication cost occurs when peers are connected to, or leave, a cluster.

To address this problem, this paper proposes a cache sharing scheme for structured data by considering peer connectivity in mobile P2P networks. The proposed scheme improves reliability through cache sharing by configuring a cluster with peers with long-term connectivity. It reduces data duplication and uses cache space efficiently by using peers inside the cluster as a single cache. In addition, it reduces data loss by dividing cache space into data cache and temporary, and using the temporary for popular data exchange in the event of topology change or cache data replacement. It also reduces the data search cost by distributing each peer's metadata of popular data to neighbor peers. Moreover, it performs cluster management to prevent loss of shared cache data. Data dissemination is performed using Multi Point Relay (MPR) [27].

The rest of this paper is organized as follows. Section 2 provides a summary of previous related studies, and Sect. 3 describes the proposed cache sharing scheme. Section 4 describes the results of performance evaluation to demonstrate the superiority of the proposed scheme over existing schemes. Finally, Sect. 5 provides the conclusion of this paper and future directions.

2 Related Work

Chow et al. [15] proposed COoperative Caching (COCA) to share the cached data cooperatively via the P2P communication. In COCA, a cache layer is inserted between the mobile client cache layer and the MSS cache layer. Each Mobile Host (MH) and its neighboring peers are clustered to share their cached data cooperatively via the P2P communication. The peers share their cache with one another to reduce the number of server requests and power consumption. To summarize a local cache in an MH, a cache signature is generated. The cache signature is a bit string, which represents all data signatures of the data in the local cache.

In [25], GROup-based COoperative Caching (GroCoca) is proposed to support P2P based cooperative caching in mobile environments. GroCoca used Tightly Coupled Group (TCG) which is a cluster of MHs that possess similar mobility pattern and similar data access pattern. Two cooperative caching management protocols such as cooperative cache admission control and cooperative cache replacement are developed. The cooperative cache admission control provides a means for the MHs to control data replicas in their TCGs. The cooperative cache replacement allows the MH to collaborate with its TCG members to replace the least valuable data.

Shen et al. [20] and Joseph et al. [21] proposed a new cooperative caching called Proximity Regions for Caching in Cooperative MP2P Networks (PReCinCt) to efficiently support scalable data retrieval in large scale mobile P2P networks. In PReCinCt, the entire network is divided into geographical regions where each region is responsible for set of keys representing the data. A hash function is used at each peer to map a key to a region's location. PReCinCt caches relevant data among a set of peers in a region. PReCinCt suggested Greedy-Dual Least-Distance (GD-LD) replacement algorithm which consider the access count, the size, and the region-distance.

Caetano and Bordim [26] proposed a cluster based cooperative caching consisting of Shared Cache Area (SCA) and Private Cache Area (PCA) in mobile ad hoc networks. To reduce the number of duplicated data found in the neighbor peers, the cache space of all cluster member are classified into two parts: SCA and PCA. SCA is used to store data that are of interest of the members of the cluster. PCA is reserved for each peer to cache data of its particular interest and are managed as an individual cache system. The cluster head manages data which pare promoted to SCA and selects which data should be stored in SCA.

Ting and Chang [19] proposed Group Caching (GC) to maintains the local caching status of 1-hop neighbor peers among the cluster member. A cluster is organized by each

MH and its 1-hop neighbor peers. The caching status is exchanged in a cluster periodically. When an MH receives a data, it caches the data object if the cache space is enough. Otherwise, the receiving MH checks the available cache spaces of its group members. If the available cache space of any group member is sufficient to store the data, the receiving MH puts the data to the group member randomly.

Joy and Jacob [23] proposed a key based cache replacement scheme called E-LRU in mobile ad hoc networks. E-LRU considers the time interval between the recent reference, size and consistency for cache replacement. If data referenced only once has in cache space, LRU is used for replacement. If data is referenced more than once, inter arrival between the most recent two references is considered for replacement.

Paul et al. [22] proposed a service cache management scheme called SCM for mobile P2P networks to enhance the service retrieval. SCM used a Distributed Spanning Tree (DST) algorithm to convert the unstructured mobile P2P network to DST structures. The DST is optimized by Ant Colony Optimization (ACO) to find the optimal routing path between nodes in unstable networks. Every DST should have its Home Node (HN). HN manages a HN_Table storing the service item ID and LeafNodeID that has the service item. Each Leaf Node (LN) manages a LN_table to store the service item ID and the value of the service item.

Kumar et al. [16] proposed a new cooperative cache replacement scheme using multiple parameters such as size, access count, Time To Live (TTL) in mobile ad hoc networks. When a node requires data, it sends request to data server and then caches data locally. The set of one-hop neighbor nodes forms a cooperative zone. Each node in the zone maintains a caching information table (CIT), which stores n data information. Three factor such size, access count, TTL of data are considered to replace a cache in each node. When size and access count is equal, data with lower TTL is removed in a cache. When size and TTL is equal, the data with higher access count is removed. When access count and TTL is equal, the data with big size is removed.

Elfaki et al. [17] proposed a cooperative caching priority (CCP) to ensure that priority requests are served with minimum cache discovery overhead. A mobile ad hoc network consists of three nodes such as cluster header, cache node, and request node. The cluster header stores information of cached data of the cache nodes. The cluster header makes decisions in serving requests based on the needs of the requested node. CCP exchanges the information of the cached data among neighbor nodes. A number of cluster headers are used to organize the distributed indexing to store the information of the cached data. CCP serves requests based on the request's classifications either priority or normal.

Kumar and Lee [18] proposed a P2P based cooperative caching called P2PCC for Vehicular Ad Hoc Networks (VANETs) to share the traffic information among vehicles. Whenever two vehicles want to share the most commonly used information, they do the same using cooperative caching in a P2P technique. If the requested information is not found in their cache, then they request to the central server. The data with a minimum probability is selected to replace the cached data. The greater the time spent in the waiting time, the greater the frequency of the data to be accessed by the vehicle and hence the lesser its probability of replacement.

Parvathya and Kumarb [28] proposed a two-tire cooperative caching scheme called 2TierCoCS for VANETs. Each vehicle stores the information of its 1-hop neighbors by exchanging beacon messages. Each RSU maintains a Neighbor Cache Index (NCI), which represents the mapping of vehicles and the cached data and stores a popular cache which caches the most frequently requested data in its zone. Data in popular cache are prefetched from the server. Cache replacement is processed at two levels; local caches and popular cache. In vehicles, when there is no space to store new data items, the item with lowest TTL will be replaced. The popular cache is replaced by request count.

3 The Proposed Cache Sharing Scheme for Structured Data

3.1 System Architecture

In existing cooperative caching schemes, each peer in a cluster has an individual replacement strategy, meaning that a lot of duplicate data are maintained within a cluster or neighbor peers. While this makes data access easy, it reduces cache utilization and makes it difficult to store diverse data, resulting in inefficiency in terms of storage space. In addition, cost increases due to the presence of duplicate data during the data search. To address this problem, this paper proposes a cluster based cache sharing scheme by considering peer connectivity in mobile P2P networks. The proposed scheme uses multiple cache spaces as a single large cache space by eliminating duplicate data among peers included in a cluster. In addition, popular data is partitioned and maintained among peers. This allows efficient use of cache space through storage of diverse data, and increases cache utilization among peers through data partitioning, ultimately reducing load and data search cost. In terms of a data search scheme, metadata of popular data are disseminated among neighbor peers in addition to a shared cache to increase communication efficiency and reduce cost. Each peer performs the data search by utilizing the received metadata. This reduces the cost to transmit queries to cluster headers or neighbor clusters, reducing overall search cost.

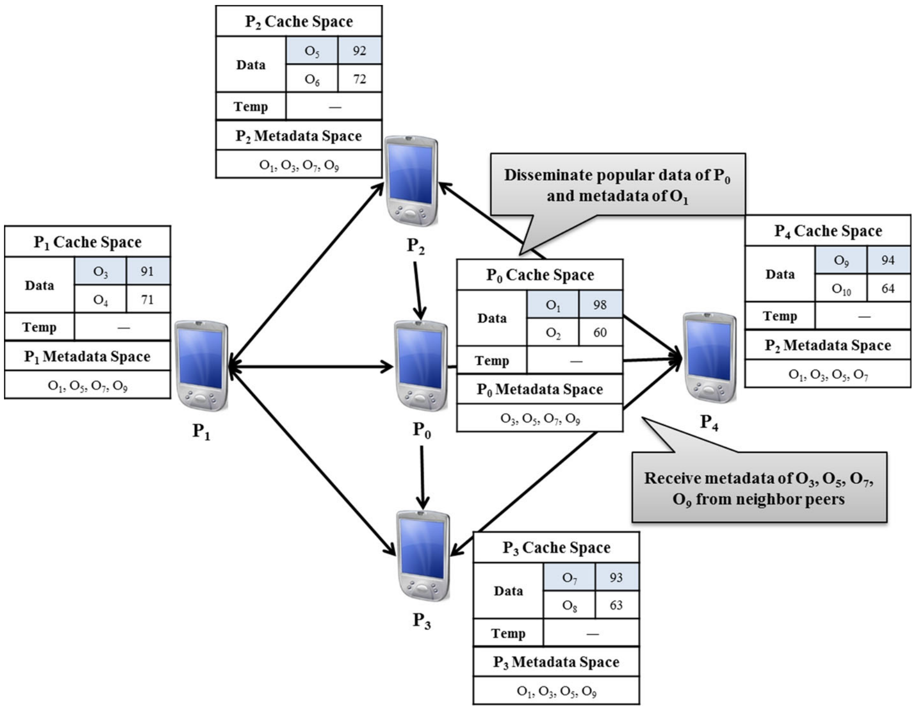


Fig. 1 Architecture of the proposed cache sharing scheme

Figure 1 shows the architecture of the proposed cache sharing scheme. The cluster is configured up to n-hop, creating small clusters. The peers formed into the cluster manage the cache data with redundancies eliminated. A cluster header manages the data of peers belonging to its cluster, and returns data using information on the cluster header's table if a peer requests data of O_1. Each peer manages cache space by dividing it into data cache and temporary. Data cache is the cache storage space that maintains data. Generally, it is the space where cache data are managed and the replace strategy is performed. Temporary is a space to store cache data temporarily, and is used like a buffer to reduce data replacement time. In addition, it is used for data exchange in the event that a peer connects to or leaves a cluster. Data dissemination can increase communication efficiency and reduce cost. To explain this from the data dissemination standpoint, the metadata of popular or frequently used data (O_1), including data identifiers, paths, and scores among the data existing in P_0 are sent to neighbor peers. When P_1 receives the metadata searches for O_1 data, P_1 can request the data from P_0 that has O_1 by utilizing the metadata, making the search fast and reducing communication cost.

3.2 Cluster Generation

In this paper, a small cluster with up to n-hop is generated by considering connectivity among peers. Clustering peers eliminates data duplication and stores diverse data in cache space, improving efficiency. In addition, peers and data belonging to the cluster are managed through a cluster header, improving efficiency in data searches. The cluster must satisfy connectivity and must not exceed the maximum number of hops. The proposed cluster generation scheme generates a cluster with peers that can maintain connectivity continuously. Connectivity refers to the time length for which direct communications among peers are possible, and is calculated with the motion vector values that represent the mobility of peers. When the connectivity value exceeds a reference threshold, peers are connected to a cluster. Once a cluster is formed, duplicate data are eliminated and the cluster is configured into a cache sharing environment. A cluster header enables fast data access when a data request is made by utilizing information on peers in the cluster and on the data that each peer has, as well as performing overall cluster management.

When a peer is in a state communicable with a specific cluster while on the move, a determination is made as to whether it can become a member of the cluster. The peer can become a member if it satisfies connectivity with the cluster and does not exceed the maximum number of hops. Figure 2 shows the cluster generation process. Figure 2a shows the connectivity determination process. If a peer P_0 satisfies connectivity and number of hops requirements, P_0 is connected to the cluster as shown in Fig. 2b. If P_0 makes the number of hops exceed the maximum number of hops for the cluster, P_0 cannot be connected to the cluster. In addition, as shown in Fig. 2d, if the connectivity between P_0 and a peer in the cluster which has the strongest connectivity with P_0 is lower than the threshold value, P_0 cannot be connected to the cluster.

The cluster header selects peers that can maintain the longest communication time among peers. It manages all peers in the cluster and communications are performed via the cluster header. In addition, it collects, manages, and maintains all information on the cluster. The cluster header manages the header table that maintains information on data and peers existing in the cluster. The header table has a structure of <*Did*, *Pid*, *Rv*>, where *Did* is data identifiers, *Pid* is peer identifiers, and *Rv* is the path from the header to a peer. Each peer maintains a peer table to manage the data it retains and information on other peers. The peer table has a structure of <*Did*, *Dv*, *Pos*, *Plist*, *CHeader*>, where *Did* is data

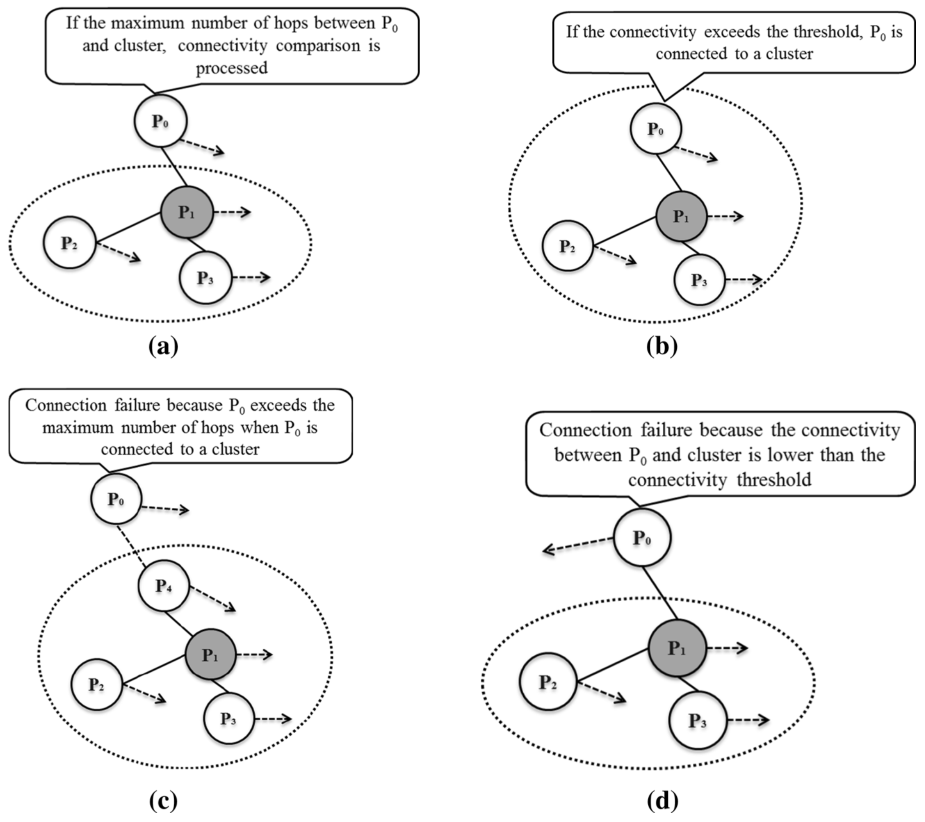

If the maximum number of hops between P_0 and cluster, connectivity comparison is processed
P_0
P_1
P_2
P_3
(a)
If the connectivity exceeds the threshold, P_0 is connected to a cluster
P_0
P_1
P_2
P_3
(b)
Connection failure because P_0 exceeds the maximum number of hops when P_0 is connected to a cluster
P_0
P_4
P_1
P_2
P_3
(c)
Connection failure because the connectivity between P_0 and cluster is lower than the connectivity threshold
P_0
P_1
P_2
P_3
(d)

Fig. 2 Cluster generation process. **a** Determination of connectivity, **b** P_0 being connected to a cluster, **c** exceeding the maximum number of hops, **d** low connectivity

identifiers, *Dv* is the value that represents data, *Pos* is motion vector information of a peer, *Plist* is the list of peers with which 1-hop communication is possible in the network, and *CHeader* is the information on the header of the cluster to which it belongs.

3.3 Basic Cache Management

Once a cluster is formed with consideration to connectivity through the proposed cluster generation scheme, peers perform cache sharing. By eliminating duplicate data among peers included in the cluster, multiple cache spaces are used as if they comprise a single large cache space. In addition, popular data are partitioned and maintained among peers. Cache space is managed by dividing into data cache and temporary. Data cache is a cache storage space that maintains data shared in the cluster, arranged by data priority. Temp cache is a space to store cache data temporarily, and a fixed portion of cache space is devoted to temporary. Initially, when a peer is connected to a cluster, an empty space is generated for the temp cache. When a particular node is connected to a new cluster, it temporarily keeps the temp cache information so that the new cluster can utilize the popular data in the previous cluster. A typical cache space executes replacement strategy if it becomes full, receiving new data and eliminating existing data. However, to replace data every time the cache is full results in a delay. Therefore, to prevent the problem, the empty space of the temporary is used to temporarily store data to replace. In addition, the temporary is used for exchange of popular data when a peer leaves a cluster as its connectivity with the cluster weakens, as well as to decompose popular data which was in the previous cluster when the peer becomes connected to a new cluster.

When the peer leaves the cluster, popular data that each peer in the cluster has are transmitted to the peer that leaves. When the peer becomes connected to another cluster, it disseminates the popular data, minimizing data loss. Figure 3 shows the situation where P_0 leaves a cluster. If there is no data exchange when P_0 leaves the cluster, a problem occurs in that P_0 and the cluster cannot access each other's upper rank data. Therefore, P_0 and other peers in the cluster exchange upper rank data. Each of the other peers in the cluster transmits their upper rank data (O_1, O_3, O_4) to the temporary of P_0, and P0 transmits its upper rank data (O_6, O_{10}) to the temporary of the cluster. P0 that leaves the cluster keeps the data in its temporary until it gets connected to a different cluster. Once P_0 becomes connected to a different cluster, it disseminates the popular data received from the previous cluster to the newly connected cluster.

When a peer becomes connected to a new cluster, it transmits its cache data information to the cluster header, and sets up an environment for cache sharing. Figure 4 shows the situation where P_0 becomes connected to a new cluster. P_0 transmits its cache data

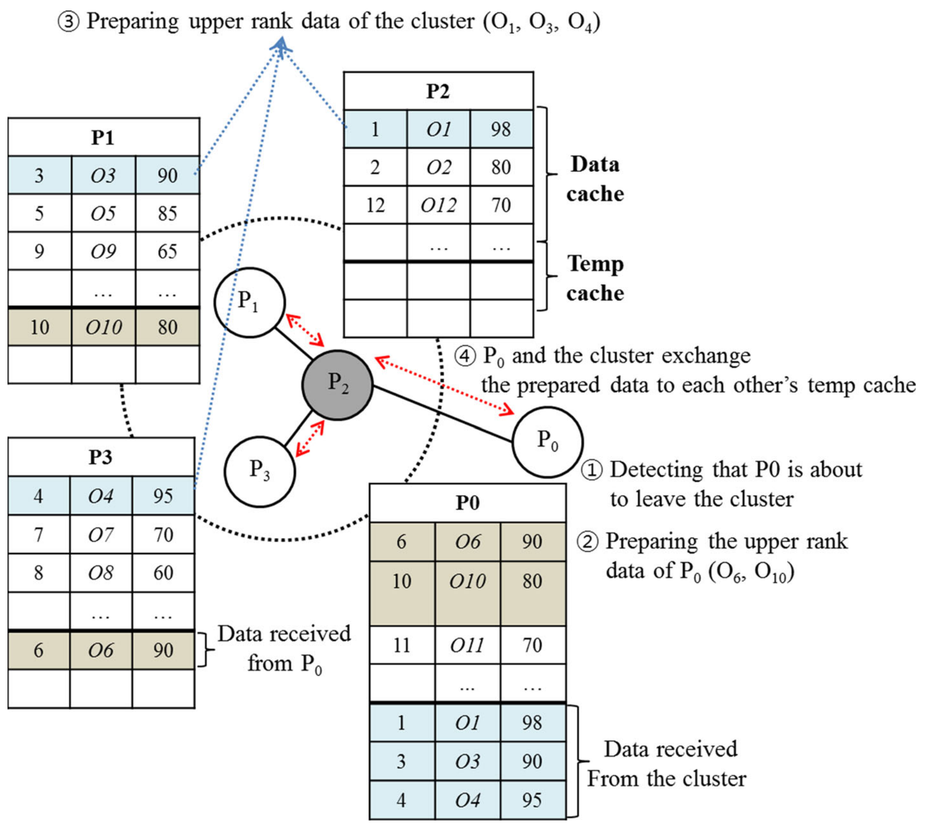

③ Preparing upper rank data of the cluster (O1, O3, O4)
P1
3 O3 90
5 O5 85
9 O9 65
… …
10 O10 80
P2
1 O1 98
2 O2 80
12 O12 70
… …
Data cache
Temp cache
P1
P2
P3
P0
④ P0 and the cluster exchange the prepared data to each other's temp cache
P3
4 O4 95
7 O7 70
8 O8 60
… …
6 O6 90
Data received from P0
① Detecting that P0 is about to leave the cluster
P0
6 O6 90
10 O10 80
11 O11 70
… …
1 O1 98
3 O3 90
4 O4 95
② Preparing the upper rank data of P0 (O6, O10)
Data received From the cluster

Fig. 3 Situation where P_0 leaves a cluster

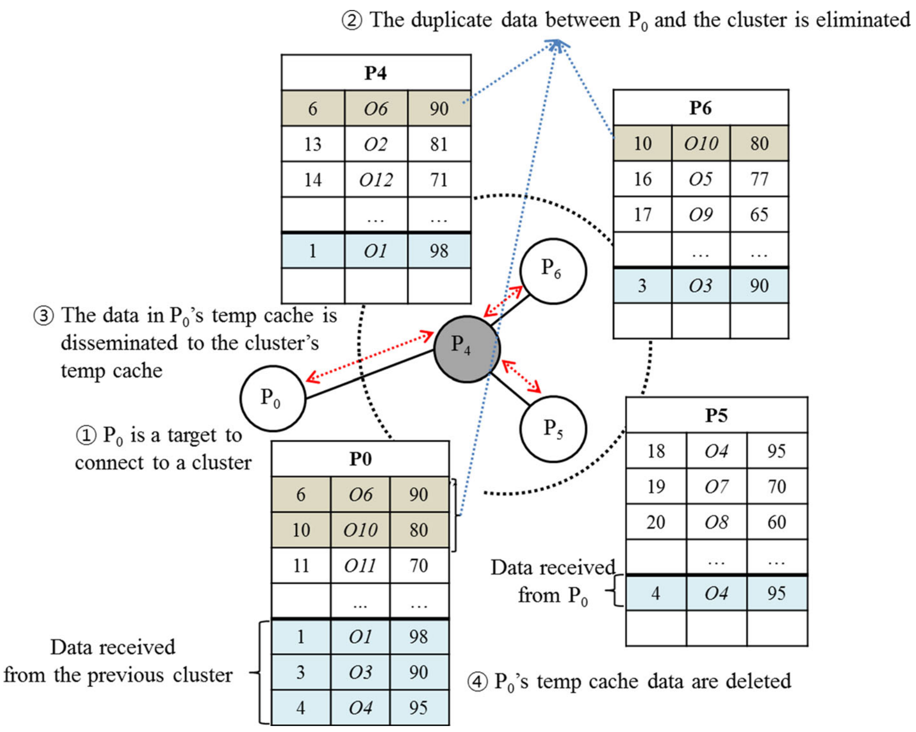

② The duplicate data between P0 and the cluster is eliminated
P4
6 O6 90
13 O2 81
14 O12 71
… …
1 O1 98
P6
10 O10 80
16 O5 77
17 O9 65
… …
3 O3 90
③ The data in P0's temp cache is disseminated to the cluster's temp cache
P6
P4
P0
P5
① P0 is a target to connect to a cluster
P0
6 O6 90
10 O10 80
11 O11 70
… …
1 O1 98
3 O3 90
4 O4 95
Data received from the previous cluster
P5
18 O4 95
19 O7 70
20 O8 60
… …
4 O4 95
Data received from P0
④ P0's temp cache data are deleted

Fig. 4 The situation where P_0 becomes connected to a cluster

information to the cluster header, and the cluster header compares it with cache data information of the cluster. If there is data that overlaps between the cache data information of P_0 and the cluster, the cluster header deletes the duplicate data in P_0's cache. Once duplicate data are deleted, P_0 disseminates its temporary data to the temporary of the peers in the cluster, providing the popular data from the previous cluster to the newly connected cluster. This can maintain recency data and can reduce communication cost in the search process. Once P_0's temporary data transmission is complete, all of P_0's temporary data are deleted. The peers that receive data from P_0 perform a local cache replacement strategy for the data in the data cache and temporary spaces, to determine which data to store in the data cache. Upon completing the local cache replacement strategy, the data in the temporary space of each peer are deleted.

3.4 Cluster Adjustment

In mobile P2P networks, disconnections occasionally occur without warning when a peer device is turned off or due to communication failure. To minimize data loss in such circumstances, a request is made to neighbor peers for the disconnected peer's upper rank data, and adjustment is performed in the event that an inefficient cluster structure occurs. Figure 5 shows the process of requesting lost data and cluster adjustment when a cluster peer is disconnected, based on a 3-hop structure. When a peer is suddenly disconnected in a cache sharing environment, some of the upper rank data are lost. In this situation, as in Fig. 5a, the cluster header recovers the data by requesting neighbor cluster headers for the

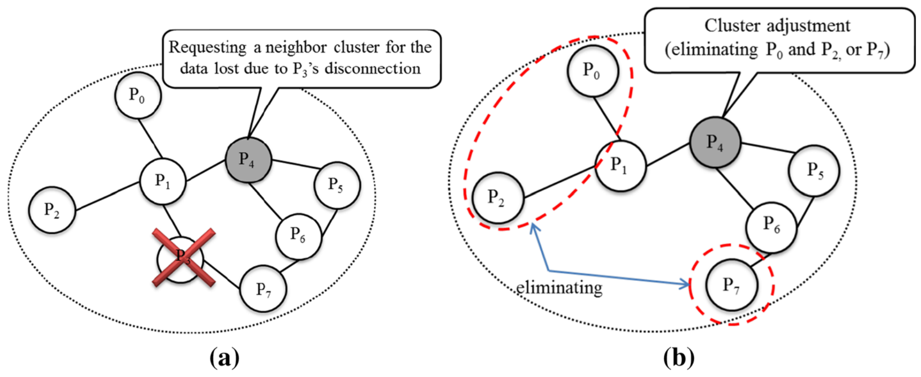

Requesting a neighbor cluster for the data lost due to P3's disconnection
P0
P1
P2
P3
P4
P5
P6
P7
(a)
Cluster adjustment (eliminating P0 and P2, or P7)
eliminating
(b)

Fig. 5 Cluster management when a peer is disconnected. **a** Requesting lost data, **b** cluster adjustment

data. When P_3 is disconnected from the cluster as in Fig. 5b, the maximum number of hops is exceeded and, therefore, the number of hops is adjusted by eliminating (P_0, P_2) or (P_7) that exist outermost of the cluster. The peers to be eliminated in the adjustment are determined by the number of peers to be adjusted and the amount of popular data.

When the cluster header is disconnected from the cluster, both the request for lost data and adjustment are difficult to perform because the cluster cannot be managed. As a result, the peers that belonged to the existing cluster generate multiple smaller clusters with peers with similar connectivity. Figure 6 shows an example in which a cluster is divided into two clusters as its cluster header is disconnected.

3.5 Data Dissemination

In a typical mobile P2P search, a peer checks both the cluster to which it belongs and neighbor clusters and, therefore, an additional search cost occurs. To solve this problem, the proposed scheme disseminates metadata of popular data of each peer to neighbor peers, preventing communication overloading of the cluster header and reducing search cost through fast search of data using metadata. The proposed data dissemination scheme distributes metadata of popular data in each peer's cache using MPR. Metadata contains

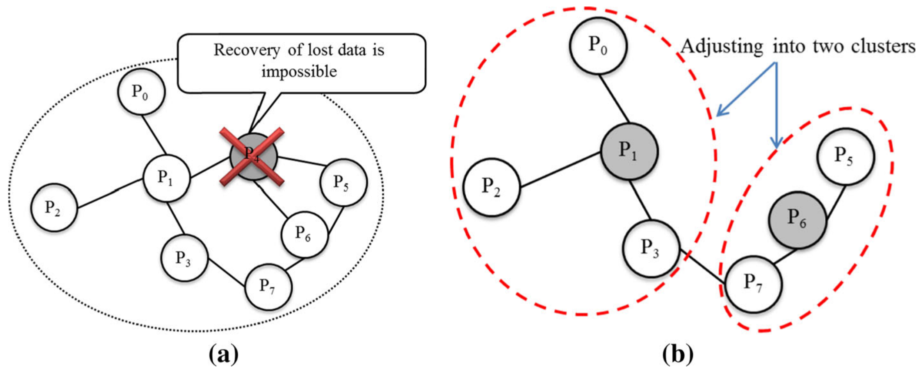

Recovery of lost data is impossible
P0
P1
P2
P3
P4
P5
P6
P7
(a)
Adjusting into two clusters
(b)

Fig. 6 Cluster management when a cluster header is disconnected. **a** Requesting lost data, **b** cluster adjustment

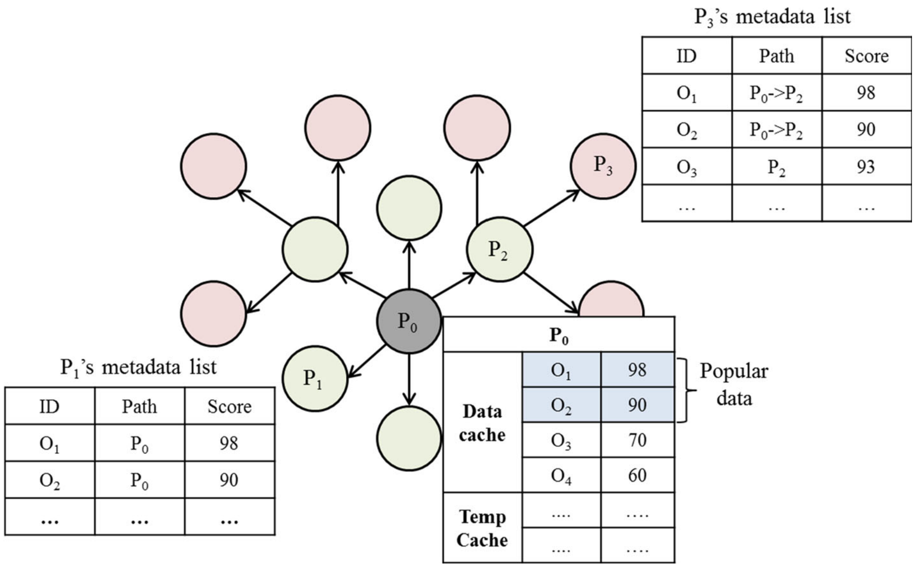
information on data identifiers, paths, and scores. The metadata received from neighbor peers are stored in cache, and improve data search performance by reducing search requests for outside clusters. In addition, the increased throughput of each peer reduces the amount of data search for the cluster header to process, preventing overloading of the header.

In this scheme, metadata on popular data or frequently used data among the data in cache are transmitted to neighbor peers. It transmits messages by selecting a specific peer using MPR because transmitting metadata with flooding increases the amount of message transmission and duplicate message transmission. This can reduce the size of data disseminated in the network and the frequency of transmission. Reduction in transmission frequency is an advantage in terms of network transmission cost; however, it can be a disadvantage in terms of reliability of transmission. In a mobile network with unreliable connections, low frequency of transmission may lead to reduced data reception. Because the proposed dissemination scheme transmits small sized data, such as metadata instead of the entire data, reception failure is less likely and communication cost are reduced.

Figure 7 shows metadata dissemination using MPR. Let us suppose that P_0 disseminates metadata of O_1 and O_2 in cache. First, 1-hop peers add peers that provide the only available paths to 2-hop peers. The peers like P_1, which has no peer to which to transmit the metadata, stores the metadata and ends the communication. In the case of P_2, it stores the received metadata and transmits it to the next peer that added P_2 as a path. Because network topology keeps changing, the scheme updates metadata to the latest information by iterating the above process at regular intervals.

3.6 Data Search

In the proposed scheme, data search is performed using metadata received from neighbor peers in addition to a shared cache to increase communication efficiency and reduce cost.



P3's metadata list

ID	Path	Score
O_1	P_0->P_2	98
O_2	P_0->P_2	90
O_3	P_2	93
…	…	…

P1's metadata list

ID	Path	Score
O_1	P_0	98
O_2	P_0	90
…	…	…

P_0		
Data cache	O_1	98
	O_2	90
	O_3	70
	O_4	60
Temp Cache	….	….
	….	….

Fig. 7 Metadata dissemination using MPR

This can reduce cost for sending queries to, and receiving replies from, the cluster header because a data search can be directly requested to the peers within a 1-hop or 2-hop cache that has the data to satisfy the query. In addition, the scheme reduces the frequency of query requests to neighbor clusters, reducing communication cost.

Figure 8 shows the data search process. As shown in Fig. 8a, when P_0 makes a query, it searches for corresponding data in its own local cache. If the local cache does not satisfy the query, it checks its metadata list (Fig. 8b). If the metadata list does not have data for the query, it sends the query to the cluster header to check the cluster cache (Fig. 8c). The cluster header that received the query compares it with data identifiers in the header table. If there is a hit for the query, the cluster header requests the peer for the data, and returns the searched data to P_0. If the cluster cache has results for the query, a query request is sent to a neighbor cluster. When the data for the query exists in the peer on the path to a neighbor cluster header, the data is returned to P_0. Otherwise, a query is sent to the neighbor cluster header, and the data search is processed in the same manner as described earlier for a within-cluster search.

Figure 9 shows the algorithm to perform the data search in cooperative caching. First, the query_processing() function is executed, and pID sends a query to look up the data search result. Lines 02–06 of query_processing() execute the data search for a the local cache. If Did equals Q in a comparison in the peer table, the data is returned to Pissue. If the algorithm fails to find a hit in local cache, it executes the data search section to check the metadata list. Pissue calls peers in Plist a forward() function, a query transmission

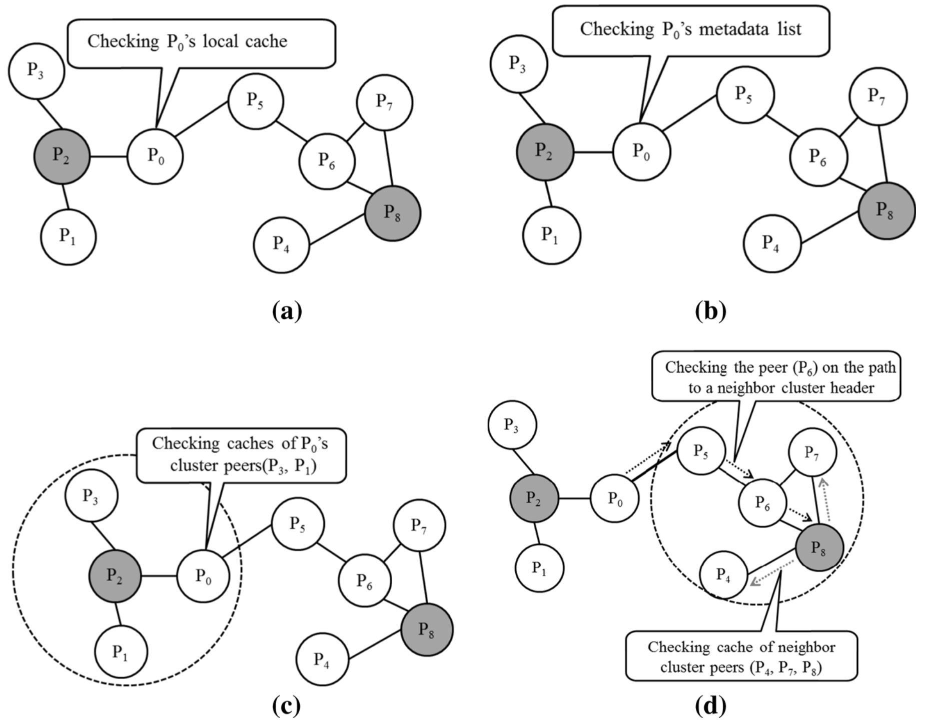

Checking P_0's local cache
P_0
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
(a)
Checking P_0's metadata list
(b)
Checking caches of P_0's cluster peers(P_3, P_1)
(c)
Checking the peer (P_6) on the path to a neighbor cluster header
Checking cache of neighbor cluster peers (P_4, P_7, P_8)
(d)

Fig. 8 P_0's data search process. **a** Checking P_0's local cache, **b** checking P_0's metadata list, **c** checking cache of the cluster, **d** checking cache of a neighbor cluster

```
01: function Query_Processing(Q, L, Plist, R)
// Search local cache data
02: for each data = L.Did[i]
03: | if $L \neq$ null and data = Q then
04: | | return Dv[i]
05: | end if
06: end for each
// Check metadata list
07: Plist $\leftarrow$ Pissue   as a list of peers in 1-hop from Pissue
08: for each Plist[i]
09: | forward(Plist[i], Pissue, “Pone”, Q)
10: end for each
11: Ptotal = Plist + Pmpr
12: Ep.add(Ptotal)
// Check the cache of the cluster it belongs to
13: if Pissue $\neq$ CHeader and Pissue $\subset$ C then
14: | forward(CHeader, Pissue, “CHeader” Q, Ep)
15: else if Pissue = CHeader then
16: | search(“CHeader”, R, Q, Ep)
17: end if
18: Clist $\leftarrow$ CHeader   as a list of peers connected to CHeader
19: Ep.add(Clist)
20: forward(NC, CHeader,“NCHeader”, Q, Ep)
```

Fig. 9 Data search algorithm in cooperative cache environment

algorithm containing Q and the type. Lines 02–05 of forward() compares Did and Q if type is Pone and, if they are equal, returns the data to Pissue. If the data search with metadata list fails to return data, it sends the query to the cluster header. To prevent duplicate query checks for the peers, the algorithm adds Ptotal to Ep, sets the type as CHeader, and then calls forward() to send the query. Lines 12–16 search data by calling search() if there is a peer on the path while the query is transmitted from Pissue to the cluster header. If the peer does not have a hit for the query, the algorithm adds the peer to Ep and then calls forward() again. Once the query has been sent to CHeader, the algorithm compares Did and Q in the cluster table and looks up Pid and Rv for the hit. For a data request, a request function is called by adding Pid and Rv to a request algorithm, request(). It gets to Pid via Rv, and returns the data to Pissue via R by calling a response algorithm, response() function. If it fails to find a hit in the cache of the cluster to which it belongs, it sends a query request to a neighbor cluster. The request process shows the same flow as the process of transmission to its own cluster header, as described above.

4 Performance Evaluation

4.1 Analysis

In recent years, various caching schemes to improve the performance of data retrieval and sharing in mobile ad hoc network have been proposed [16, 17, 22, 23, 25, 26]. In addition, caching schemes for information sharing among vehicles in VANET environments that are special types of mobile ad hoc networks have been proposed [18, 28]. The existing caching schemes provide various access methods according to cache management methods, cluster construction methods, data retrieval methods, and replacement strategies. We compare the functional differences between the proposed scheme and the existing schemes in order to prove the superiority and originality of the proposed scheme.

GroCoca is the most representative method for cooperative caching and defines tightly-coupled group (TCG) as a set for data preferences similar to the moving patterns of peers [25]. It increases data accessibility through cooperative cache management such as cooperative cache admission control and cooperative cache replacement. However, it increases data redundancy although each cache keeps meaningful data through data preferences.

CBCCA constructs clusters according to connectivity among peers and retrieves data through inter-cluster communication and intra cluster communication [26]. It divides cache space into private cache and shared cache. All peers in the shared cache cluster keep data in common. It increases the diversity of caches and cache hit ratios by keeping and managing popular data in common in the shared cache. However, as the shared cache area increases, the efficiency of storage spaces degrades. Chow et al. [25] and Caetano and Bordim [26] did not consider the popular data exchange method when a peer leaves a cluster.

SCM was proposed to support efficient data retrieval in mobile P2P networks [22]. It provides efficient cache service and retrieval in mobile P2P networks since it builds the mobile P2P networks using distributed spanning tree. However, SCM spends additional routing costs because it conducts data retrieval based on headers. Since it manages cache spaces individually, it can replace data that neighboring nodes need in cache replacement.

KCR provides E-LRU (Extended LRU) that conducts cache replacement by considering recent accesses, time intervals of the recent referenced data, and the most recent reference time. KCR does not provide cooperative cache management and construct clusters because each node keeps a cache space independently. It does not use the storage space of a peer efficiently since some parts of cache spaces remains empty in order to reduce data replacement times.

CCR constructs clusters in 1-hop in mobile ad hoc environments and has a cache replacement scheme using SAT method [16]. In CCR, each node manages information individually and retrieves cache information using the neighboring peers when requesting data. It does not provide cooperative cache management because each node manages a cache space independently. It also does not consider the request or status of a neighbor peer in cache replacement.

CCP manages the cached data of each node through multiple cluster headers [17]. A cache node caches data from a server or other cache nodes and performs the role of a requested node (RN) in a data request. Since it assumes that all cluster headers are connected to a server, it cannot be applied to mobile ad hoc networks. It spends high routing costs in data replacement because a cluster is constructed focusing on in interests. It also has additional costs that it should transfer all requests to a header since it retrieves data based on a header.

P2PCC is for P2P based cache sharing in VANET environments [18]. In P2PCC, a cluster is composed of vehicles that can communicate with RSU and each vehicle caches the data independently. 2TierCoCS manages cache spaces exchanging information among 1-hop vehicles [28]. Since each RSU manages the cached data of each vehicle, it causes data request costs to RSU in data retrieval. Since [18, 28] are based on vehicles in a road with constrained moving patterns, they increases maintenance and retrieval costs when they are applied to unconstrained mobile ad hoc networks.

The proposed scheme generates clusters and proposes a cache sharing scheme by considering peer connectivity in mobile ad hoc environments. The proposed scheme improves reliability through cache sharing by configuring a cluster with peers with long-term connectivity. It reduces data duplication and uses cache spaces efficiently by using peers inside the cluster as a single cache. In addition, it reduces data loss by dividing the cache space into data cache and temporary, and using the temporary for popular data exchange in the event of topology change or cache data replacement. It also reduces the data search cost by distributing each peer's metadata of popular data to neighbor peers. Moreover, it performs cluster management to prevent loss of shared cache data.

Table 1 shows the comparison of the proposed scheme and the existing schemes in terms of Management, Clustering, Discovery, Mobility and Replacement. Management means a cache management scheme, where Cooperative represents a cooperative strategy for cache replacement and management, Central represents a cache management strategy by a server or cluster headers, and Individual represents a strategy that each node manages a cache individually and collaborates with other nodes only in data retrieval. Clustering means properties that are considered when clusters are constructed. Discovery means a retrieval scheme using cache when data is requested. Here, Neighbor represents a scheme that retrieves the caches of neighbor peers using a P2P method, Header represents a scheme that retrieves the caches of neighbor peers using a sever or cluster headers, Neighbor + Header combines Neighbor and Header, and Neighbor + RSU represents a scheme that combines Neighbor with a retrieval method using RSU in VANET environments. Replacement means a method that is used as a replacement strategy.

4.2 Evaluation Results

Performance evaluation was conducted on a PC with Intel(R) Core(TM) i7 processor and Windows 7 with 4G memory. The experimental environment is shown in Table 2. The basic experiment setting was the number of peers, 200; the number of data, 500; cache size, 30 in a 500 × 500 space; and the number of cluster hops was configured as 2-hop. Here, the locations of peers and data were randomly generated. Many studies for efficiently supporting data sharing and retrieval in mobile ad hoc environments have been conducted. As we mentioned in Sect. 4.1, SCM [22] constructs cache spaces using DST and manages

Table 1 Comparison of cooperative caching schemes

Feature	Management	Clustering	Discovery	Mobility	Replacement
GroCoca [25]	Cooperative	Mobility	Neighbor + Header	Unconstraint	TTL
CBCCA [26]	Cooperative	Connectivity	Neighbor + Header	Unconstraint	LRU
SCM [22]	Central	DST	Header	Unconstraint	N/A
KCR [23]	Individual	N/A	Neighbor	Unconstraint	E-LRU
CCR [16]	Individual	1-hop	Neighbor + Server	Unconstraint	SAT
CCP [17]	Partial cooperative	Interest	Header	Unconstraint	LRU + TTL
P2PCC [18]	Individual	RSU	Neighbor	Constraint	Waiting time
2TierCoCS [28]	Partial cooperative	RSU	RSU	Constraint	TTL, request count
The proposed	Cooperative	Connectivity	Neighbor + Header	Unconstraint	Popularity, access time

Table 2 Experimental environment

Parameter	Value
Number of peers (count)	50–400
Number of data (count)	500–900
Cache size (number of data)	10–50
Communication range (m)	50
Network size	500 × 500
Temporary size (number of data)	4–20

them based on headers. KCR [23] and CCR [16] construct clusters and manage cache spaces individually. CCP [17] shares some cache spaces through cooperative caching but manages them based on headers. Kumar and Lee [18], Parvathya and Kumarb [28] are based on vehicles in a road with constrained moving patterns. GroCoca [25] that is the most representative cooperative caching scheme in mobile ad hoc environments constructs clusters according to peer mobility and manages cache spaces cooperatively. CBCCA [26] constructs clusters according to connectivity among peers and manages cache spaces in common by peers in a cluster. The proposed scheme provides cooperative cache sharing among nodes with unconstrained moving patterns. In this experimental evaluation, we chose GroCoca and CBCCA that are cooperative caching schemes as the existing schemes for performance comparison. The reason is that they are cooperative caching schemes that their execution environments are similar to those of the proposed scheme. We compared the performances of the existing schemes and the proposed scheme in terms of cache hit ratio, data replacement time, and energy consumption as the number of data, cache size, and the number of peers changes. The proposed scheme conducts data dissemination to improve retrieval performance. In order to prove the superiority of the data dissemination proposed in this paper, we evaluate the performance of the proposed scheme according to data dissemination.

The cache hit ratio is the most important element of the cache management scheme. Hit ratios are evaluated because higher hit ratios suggest successful searches and the superiority of the cluster structure in a mobile P2P network. The cache hit ratio means the rate that when a cluster consists of n nodes, the requested data can be found in the cluster. The cache hit ratio is calculated as Eq. 1, where CoP is the number of nodes in a cluster, and $localhit_i$ is 1 when the local cache of node i hits and is 0 otherwise.

$$hitratio = \frac{\sum_{i=1}^{n} localhit_i}{CoP} \tag{1}$$

Figure 10 shows cache hit ratios as a function of the number of data. It indicates that cache hit ratios decrease with increased numbers of data because of the limited amount of data maintained in cache. When the number of data increases, CBCCA and GroCoca show more similar hit ratios. Because CBCCA employs joint maintenance of cache in a cluster, which results in reduced diversity of data in the cluster when the number of data increases, the cache hit ratios sharply decline with the increase in the number of data. The proposed scheme was found to outperform GroCoca and CBCCA by approximately 16 and 9%, respectively, in cache hit ratio.

Figure 11 shows cache hit ratios as a function of cache size. Because CBCCA has a shared cache, the cluster maintains more diversity and greater quantities of data when the cache size increases. The proposed scheme shows high hit ratios because the entire cache is

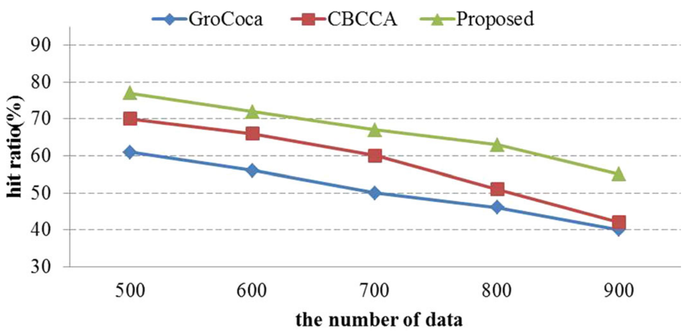


Fig. 10 Cache hit ratios for the number of data

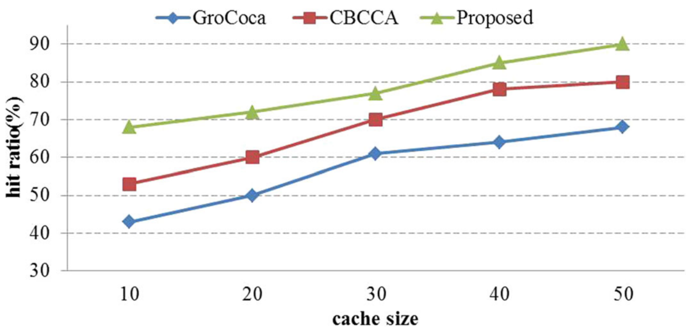


Fig. 11 Cache hit ratios for cache size

shared and, therefore, even caches that are small in size make a large cache for the entire cluster, resulting in diverse data that are maintained. The proposed scheme was found to outperform GroCoca and CBCCA by approximately 21 and 10%, respectively, in cache hit ratio.

Figure 12 shows cache hit ratios as a function of the number of peers. In an environment with a small number of peers where it is difficult to form a cluster, the conventional scheme and the proposed scheme show similar hit ratios. In CBCCA and the proposed scheme, a cluster is formed with more peers and the amount to share increases as the number of peers increases. As a result, the cache hit ratio shows a marked increase with the

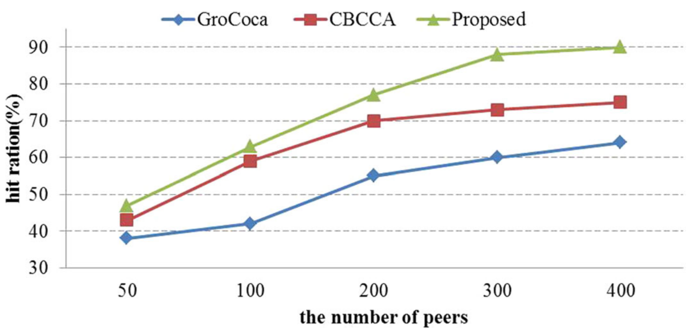


Fig. 12 Cache hit ratios for the number of peers

increase in the number of peers. The proposed scheme was found to outperform GroCoca and CBCCA by approximately 21 and 9%, respectively, in cache hit ratio.

Data replacement time refers to the amount of time used to check the cluster header for data duplication when data eligible for replacement comes into local cache, and to insert the data after deleting a data item with top priority for replacement if there is no duplication. A smaller amount of data replacement time means higher search performance and cluster management efficiency. Figure 13 shows data replacement time as a function of the number of data. It shows that an increased number of data results in reduced diversity of data in the cluster, causing more frequent data replacement and, therefore, increasing data replacement time. The proposed scheme was found to outperform GroCoca and CBCCA by reducing data replacement time by approximately 50 and 16%, respectively.

Figure 14 shows data replacement time as a function of cache size. When cache size is small, GroCoca and CBCCA show a larger amount of data replacement time compared to the proposed scheme. The proposed scheme maintains diverse data because the entire cache is shared and, therefore, even caches that are small in size make a large cache for the entire cluster. As a result, replacement strategy is not employed frequently, keeping the amount of data replacement time low. The proposed scheme was found to outperform GroCoca and CBCCA by reducing data replacement time by approximately 56 and 20%, respectively.

Figure 15 shows data replacement time as a function of the number of peers. As the number of peers increases, a cluster is formed with more peers and the amount to share

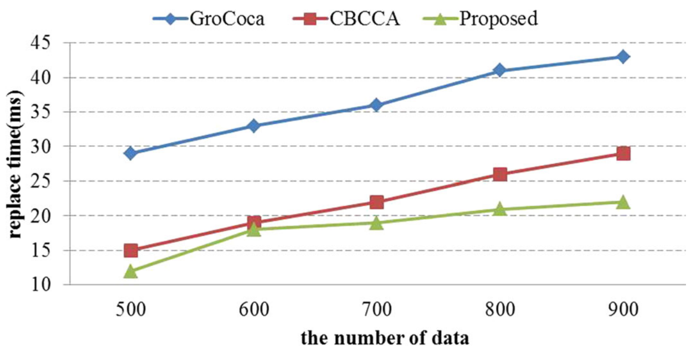


Fig. 13 Data replacement time for the number of data

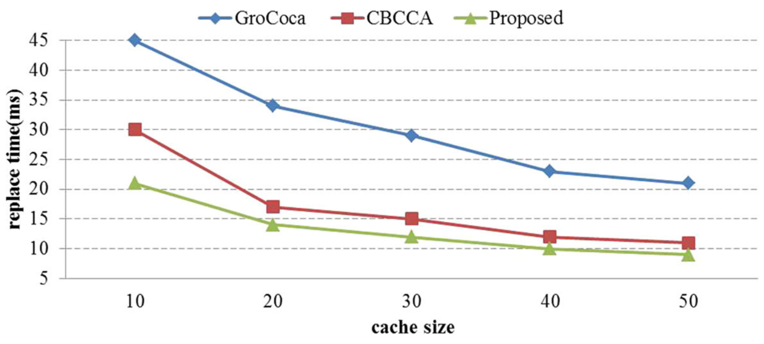


Fig. 14 Data replacement time for cache size

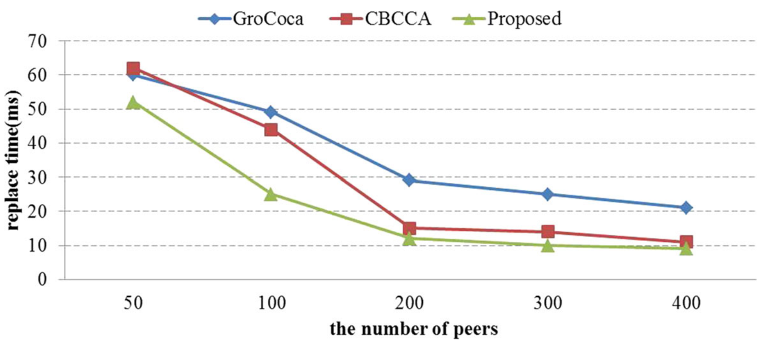


Fig. 15 Data replacement time for the number of peers

increases as the number of peers increases. As a result, replacement strategy is not employed frequently, keeping the length of data replacement time low. The proposed scheme was found to outperform GroCoca and CBCCA by reducing data replacement time by approximately 47 and 25%, respectively.

In order to show the performance superiority of the proposed scheme, we compared it with the existing schemes, GroCoca and CBCCA. We also evaluated the performance of the proposed scheme in terms of data dissemination. In performance evaluation, the proposed scheme without data dissemination is called Proposed1 and the proposed scheme with data dissemination is called Proposed2. In other words, Proposed1 is a scheme that when data is requested in a status that it does not disseminate popular data like CBCCA, it first checks the local cache and then if the data does not exist in the local cache, it transfers the request to a cluster header. Proposed2 is a scheme that when data is requested in a status that it disseminates popular data, it first checks the local cache and then if the data does not exist in the local cache, it transfers the request to cluster headers continuously until data is retrieved. As the routing cost for data retrieval increases, the energy consumption of a node increases. Therefore, we evaluate the energy consumption of the proposed scheme according to data dissemination.

Figure 16 shows the power consumption according to the number of data. GroCoca and CBCCA decrease cache hit ratios according that the number of data increases. They also increase the retrieval costs since when each peer does not have the desired data in the local cache, it transfers the retrieval request to the neighbor peers and a header. Proposed1 increases the cache hit ratio over the existing schemes, But it increases the retrieval costs. The reason is that since it does not use data dissemination, it transfers the retrieval request to the neighbor peers and a header when there is not the desired data in the local cache like the existing schemes. Proposed2 significantly reduces the energy consumption over the existing schemes. The reason is that it knows meta data and retrieval paths in advance through data dissemination. As a result, it is shown through performance evaluation that the proposed scheme reduced the power consumption by about 21 and 14% over GroCoca and CBCCA on average. Proposed2 reduced the power consumption by about 8% over Proposed1.

Figure 17 shows the power consumption according to the cache size. As shown in Fig. 17, as the cache size increases, the communication costs decrease and the power consumption reduce. The reason is that as the cache size increases, the cache can include much more data. As a result, it is shown through performance evaluation that the proposed scheme outperforms GroCoca and CBCCA by about 17 and 14% on average. When

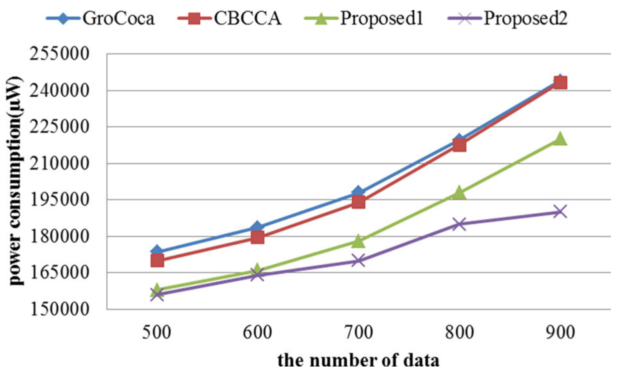


Fig. 16 Power consumption for the number of data

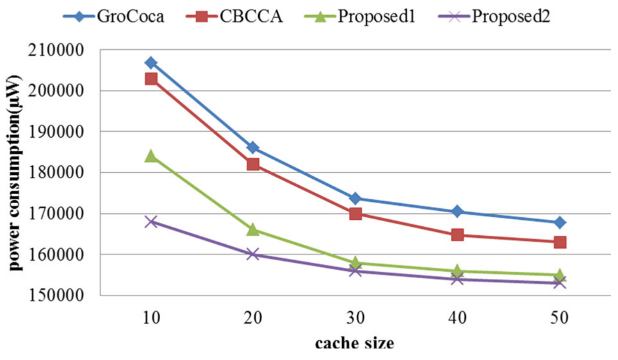


Fig. 17 Power consumption for cache size

compared to Proposed1, Proposed2 allows efficient processing even when the cache size is small because of its use of metadata. The proposed scheme was found to outperform Proposed1 by reducing power consumption by approximately 4%.

Figure 18 shows the power consumption according to the number of peers. As the number of peers in a cluster increases, the possibility that neighbor peers in the cluster

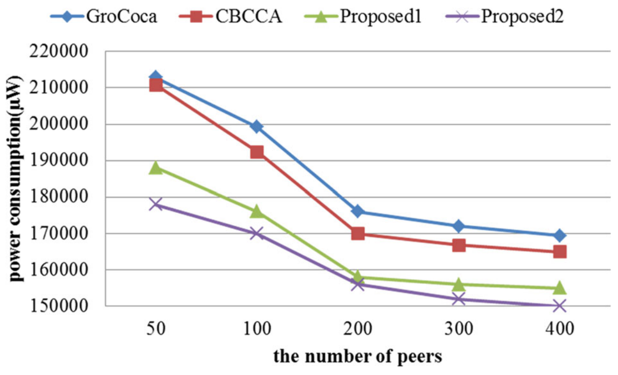


Fig. 18 Power consumption for the number of peers

provides the desired data in a retrieval request increases. Therefore, as the number of peers increases, the power consumption reduces. As a result, it is shown through performance evaluation that the proposed scheme reduced the power consumption by about 18 and 15% over GroCoca and CBCCA on average. Proposed2 reduced the power consumption by about 4% over Proposed1.

5 Conclusions

In this paper, we proposed a cache sharing scheme for structured data by considering peer connectivity in mobile P2P networks. The proposed scheme shares cache by configuring a cluster with peers that can maintain long-term connectivity. It reduces data duplication and uses cache space efficiently by using peers inside the cluster as a single cache. In addition, it uses cache space by dividing it into data cache and temporary. Data cache is a space to manage cache data and perform replacement strategy, and temporary is a fixed portion of empty space created for data replacement where popular data are temporarily stored in the event a change occurs in the topology of the cluster. In addition, when replacing cache data, the empty space is used like a buffer, reducing data replacement time. Moreover, to reduce the cost of communications to transmit queries, a data dissemination scheme was added by introducing MPR. It disseminates metadata of popular data of each peer to neighbor peers, resulting in efficient processing in data search and return. In addition, the proposed scheme performs data recovery and cluster mediation through cluster management in the event that a peer is suddenly disconnected. The scheme outperforms the existing schemes in reducing communication overheads. Results of experimental evaluation showed that the proposed scheme improved cache hit ratios by 9–25%, reduced the data replacement times by 16–56%, and reduced power consumption by 3–6%, compared to existing schemes. Further research is in plan to apply the proposed cache sharing scheme to actual mobile P2P networks.

Acknowledgements This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2013-0-00680) supervised by the IITP (Institute for Information & communications Technology Promotion) and by the "Development of biomedical data network analysis technology based on high performance computing for dementia researches (K-17-L03-C02-S02)" funded by Korea Institute of Science and Technology Information.

References

1. Meng, X., & Li, T. (2013). A dynamic load balancing scheme with incentive mechanism in heterogeneous structured P2P networks. *Computer and Electrical Engineering, 39,* 2124–2134.
2. Shojafar, M., Abawajy, J. H., Delkhah, Z., Ahmadi, A., Pooranian, Z., & Abraham, A. (2015). An efficient and distributed file search in unstructured peer-to-peer networks. *Peer-to-Peer Networking and Applications, 8,* 120–136.
3. Bok, K., Kwak, D., & Yoo, J. (2012). A resource discovery with data dissemination over unstructured mobile P2P networks. *KSII Transactions on Internet and Information Systems, 6,* 815–834.
4. Zeng, D., & Geng, Y. (2014). Content dissemination mechanism in mobile P2P network. *Journal of Networks, 9,* 1229–1236.
5. Ahmed, D. T., & Shirmohammadi, S. (2007). Design issues of peer-to-peer systems for wireless ad hoc networks. In *Proceeding of international conference on networking*, p. 26.
6. Li, H., Bok, K. S., Chung, K. Y., & Yoo, J. S. (2014). An efficient data dissemination method over wireless ad-hoc networks. *Wireless Personal Communications, 79,* 2531–2550.

7. Li, H., Bok, K. S., & Yoo, J. (2015). A mobile social network for efficient contents sharing and searches. *Computers & Electrical Engineering, 41,* 88–300.
8. Lim, H., & Kim, C. (2001). Flooding in wireless ad hoc networks. *Computer Communications, 24,* 353–363.
9. Qayyum, A., Viennot, L., & Laouiti, A. (2002). Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *Proceeding of annual Hawaii international conference on system sciences*, pp. 3866–3875.
10. Shah, B., & Kim, K. (2014). Towards enhanced searching architecture for unstructured peer-to-peer over mobile ad hoc networks. *Wireless Personal Communications, 77,* 1167–1189.
11. Liu, C., Chen, C., Chen, Y., & Wang, J. (2015). A mobile P2P semantic information retrieval system with effective updates. *KSII Transactions on Internet and Information Systems, 9,* 1807–1824.
12. Chen, K., & Shen, H. (2015). Maximizing P2P file access availability in mobile ad hoc networks though replication for efficient file sharing. *IEEE Transactions on Computers, 64,* 1029–1042.
13. Ye, F., Li, Q., & Chen, E. (2011). Benefit based cache data placement and update for mobile peer to peer networks. *World Wide Web, 14*(3), 243–259.
14. Cao, G., Yin, L., & Das, C. R. (2004). Cooperative cache-based data access in ad hoc networks. *IEEE Computer, 37,* 32–39.
15. Chow, C., Leong, H. V., & Chan, A. T. S. (2004). Cache signatures for peer-to-peer cooperative caching in mobile environments. In *Proceeding of international conference on advanced information networking and applications*, pp. 96–101.
16. Kumar, P., Chauhan, N., Awasthi, L. K., & Chand, N. (2014). Cooperative cache replacement policy for MANETs. *International Journal of Advanced Pervasive and Ubiquitous Computing, 6,* 36–47.
17. Elfaki, M. A., Ibrahim, H., Mamat, A., Othman, M., & Safa, H. (2014). Collaborative caching priority for processing requests in MANETs. *Journal of Network and Computer Applications, 40,* 85–96.
18. Kumar, N., & Lee, J. (2014). Peer-to-peer cooperative caching for data dissemination in urban vehicular communications. *IEEE Systems Journal, 8,* 1136–1144.
19. Ting, Y., & Chang, Y. K. (2007). A novel cooperative caching scheme for wireless ad hoc networks: Group caching. In *Proceeding of international conference on networking, architecture and storage*, pp. 62–68.
20. Shen, H., Joseph, M. S., Kumar, M., & Das, S. K. (2005) PReCinCt: A scheme for cooperative caching in mobile peer-to-peer systems. In *Proceeding of international parallel and distributed processing symposium*, p. 57.
21. Joseph, M. S, Kumar, M., Shen, H., & Das, S. (2005). Energy efficient data retrieval and caching in mobile peer-to-peer network. In *Proceeding of international conference on pervasive computing and communications workshops*, pp. 50–54.
22. Paul, P. V., Rajaguru, D., Saravanan, N., Baskaran, R., & Dhavachelvan, P. (2013). Efficient service cache management in mobile P2P networks. *Future Generation Computer Systems, 29,* 1505–1521.
23. Joy, P. T., & Jacob, K. P. (2013). A key based cache replacement policy for cooperative caching in mobile ad hoc networks. In *Proceeding of international advance computing conference*, pp. 383–387.
24. Chow, C. Y., Leong, H. V., & Chan, A. T. S. (2005). Distributed group based cooperative caching in a mobile broadcast environment. In *Proceeding of international conference on mobile data management*, pp. 97–106.
25. Chow, C., Leong, H. V., & Chan, A. T. S. (2007). GroCoca: Group-based peer-to-peer cooperative caching in mobile environment. *IEEE Journal on Selected Areas in Communications, 25,* 179–191.
26. Caetano, M. F., & Bordim, J. L. (2010). A cluster based collaborative cache approach for MANETs. In *Proceeding of international conference on networking and computing*, pp. 104–111.
27. Liang, O., Sekercioglu, Y. A., & Mani, N. (2006). A survey of multipoint relay based broadcast schemes in wireless ad hoc networks. *IEEE Communications Surveys and Tutorials, 8,* 30–46.
28. Parvathya, P. R., & Kumarb, K. S. A. (2015). 2TierCoCS: A two-tier cooperative caching scheme for internet-based vehicular ad hoc networks. *Procedia Computer Science, 46,* 1079–1086.

Kyoungsoo Bok He received the B.S. in Mathematics from Chungbuk National University, Korea in 1998 and also received M.S. and Ph.D. in Information and Communication Engineering from Chungbuk National University, Korea in 2000 and 2005 respectively. He is now a research professor in Information and Communication Engineering, Chungbuk National University, Korea. His research interests are sensor network, location based service, mobile ad-hoc network, social network, and big data.

Jaegu Kim He received the B.S. and the M.S in Information and Communication Engineering from Chungbuk National University, Korea in 2013 and 2015 respectively. His research interests are mobile P2P, social network, cloud computing, and big data.

Jaesoo Yoo He received M.S. and Ph.D. in Computer Science from Korea Advanced Institute of Science and Technology, Korea in 1991 and 1995 respectively. He is now a professor in Information and Communication Engineering, Chungbuk National University, Korea. His research interests are database system, storage management system, sensor network, distributed computing, and big data processing.