Journal of Network and Computer Applications 35 (2012) 1990–2002

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A hierarchical overlay with cluster-based reputation tree for dynamic peer-to-peer systems

Chih-Lin Hu *, Tzu-Han Kuo

Department of Communication Engineering, National Central University, No. 300, Jung-da Rd, Jung-li City, Taoyuan 32001, Taiwan, ROC

ARTICLE INFO

Article history:
Received 25 November 2011
Received in revised form
3 June 2012
Accepted 31 July 2012
Available online 9 August 2012

Keywords:
Overlay
Reputation system
Peer-to-peer system
Mobile peer-to-peer computing
Distributed system

ABSTRACT

Traditional peer-to-peer technologies and systems assume that people operate with desktop computers in fixed broadband networks. When people with modern mobile devices now access Internet and Web services much in the manner they used to on desktop computers, the classical peer-to-peer overlay models can be vulnerable in wireless and mobile networks. This paper proposes a hierarchical overlay architecture based on partially central and semi-structured overlay models for the deployment of peer-to-peer systems in dynamic network environments. To keep up system scalability and efficacy, this architecture design exploits peer locality and network proximity, and contends with several problems of peer churn, peer mobility, search redundancy and traffic overhead that become much stickier in dynamic network environments. This design also integrates the reputation notion to mitigate the free-riding problem in peer-to-peer systems. According to a special cluster-based reputation tree, the hierarchical overlay is adjustable to moderate unfair or imbalanced resource utilization over the system. Furthermore, the cluster hierarchy is resilient to any points of failure at peer clusters in the overlay topology. Therefore, the effort of this study achieves an efficient and robust overlay architecture in dynamic network environments. Simulation results show that the proposed architecture is not only scalable to peer population, but also sustainable to peer- and network-initiated dynamics and influences in peer-to-peer systems.

1. Introduction

The wide deployment of many peer-to-peer (P2P) file sharing, media streaming and live streaming applications (Liu et al., 2008; Androutsellis-Theotokis and Spinellis, 2004), e.g., BitTorrent, eDonkey, Gnutella, Napster, PPstream and PPLive, dominates a major part of today's Internet traffic (ipoque GmbH, 2009). The success of P2P methodologies resorts to users' free willingness of sharing media contents, storage and computation resources. P2P systems enable the collective ability of a network of user-provided devices, so-called *peers* hereinafter, to offer highly distributed services and contents, drastically reducing the need for expensive servers and network infrastructures that traditional client–server systems often get entangled in.

A P2P system operates as an application-level *overlay* network (Lua et al., 2005) that provides connectivity, routing, and messaging among peers that are addressable on top of the IP layer. Ideally, a P2P overlay would be autonomous with self-organization and self-management against the churning problem, i.e., dynamics of peer leaving and joining. For resource provisioning, the feature set and performance are determined by initial configuration, local capability assessment, and peer exchange of measurements. For service quality, content replication and discovery mechanisms are used to serve peers in finding contents of interest in the proximity. Furthermore, fault detection and repair facilities are made automatically to avoid information loss and sustain system performance.

1.1. P2P critique

In reality, P2P systems encounter several critical issues, including topology mismatch, peer churn, and free-riding problems, as mentioned below.

Although an overlay separate from the underlying network topology eases the deployment of P2P systems, topology mismatch affects the system performance because of arbitrary organizations in unstructured systems or distributed hash-based properties in structured systems (Han et al., 2005; Hsiao et al., 2009; Liu et al., 2005; Qiu et al., 2007; Lua and Zhou, 2007; Papadakis et al., 2012). The neighborhood of two peers does not reflect location proximity in physical networks. Peer messaging in a small number of logical hops not only endures long-delay transmission but also induces redundant

* Corresponding author. Tel.: +886 3 4227151x35513; fax: +886 3 4229187.
E-mail addresses: charleshu911@gmail.com, clhu@ce.ncu.edu.tw (C.-L. Hu), kinhoco2b@gmail.com (T.-H. Kuo).

1084-8045/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2012.07.022

traffic to a considerable extent. The report of Gnutella in Bolla et al. (2009) pointed that only 3–4 percent of peers belong to the same autonomous systems. Thus, it is difficult to support content services in proximity, e.g., location-based query and response processes in P2P systems (Lv et al., 2002; Chow et al., 2011; Nghiem and Waluyo, 2011).

User-driven dynamics of peer participation or *churn*[1] is an inherent property in P2P systems (Stutzbach and Rejaie, 2006). Due to the autonomous nature of peers, their mutual dependency and large peer population, peer churn has a great impact on the overlay structure design, resiliency, and system performance (Bustamante and Qiao, 2008). The degree of churn, i.e., the stability of a peer's neighboring set, affects the satisfaction level of caches, replications, and query processing in the context of P2P content distribution. Notice that peer churn becomes much stickier under peer mobility and roaming in wireless and mobile environments. It is not judicious to separate the mobile P2P overlay from physical infrastructure and geographical location partitions (Huang et al., 2007). Hence, a P2P overlay design must wrestle with drastic variances of churn rate in dynamic P2P networks (Biskupski et al., 2007).

The free riding problem is difficult to be resolved in P2P systems, where rational participants may refuse to contribute their fair share of resource but act to maximize their own utility (Adar and Huberman, 2000; Feldman and Chuang, 2005). Participants in doing so are termed "free-riders." P2P networks are rife with free riders, since individual rationality is in conflict with social welfare. A very small proportion of the peers are responsible for the vast majority of the sharing (Hughes et al., 2005). Thus, incentive approaches are presented to restrain free riders and to encourage peers' contributions for service fairness in P2P systems (Feldman and Chuang, 2005).

The literature review shows that most incentive approaches are designed upon monetary payment and reciprocity-based schemes. Monetary payment schemes dictatee that service recipients have to pay service providers for resources they consume. This type of incentive method often uses the game theory, e.g. Nash equilibrium, to study the potential benefits of micropayment techniques in P2P file systems (Eger and Killat, 2008; Golle et al., 2001). In reciprocity-based schemes, peers refer to the histories of peers' reciprocal behavior during their decision making processes. A *direct*-reciprocity scheme is fit for applications with long session durations, as they provide ample opportunities for reciprocation between pairs of peers. A typical example is the tit-for-tat incentive method (Cohen, 2003) used in BitTorrent-like applications that make uploading as an intrinsic part of the rarest first and choke algorithms (Legout et al., 2006). By contrast, an *indirect*-reciprocity scheme, a.k.a. reputation systems (Resnick et al., 2000; Marti and Garcia-Molina, 2006), determines reputation scores/credits of a peer's contribution to the system, and then performs mapping of scores to different strategies. According to reputation differentiation, reputable peers are rewarded better service than disreputable peers by the free-riding severity (Hughes et al., 2005). Remarkably, the growing recognition of reciprocity-based schemes has led to significant research efforts in this area. Many related studies (Yu et al., 2010; Jin and Chan, 2010; Satsiou and Tassiulas, 2010; Hu et al., 2010) addressed the feasibility and reliability for P2P systems with large peer population, highly dynamic memberships, and infrequent repeated transactions.

1.2. Our scheme and contribution

This paper accounts for peer- and network-initiated dynamics and influences in dynamic P2P networks, containing a mixture of dynamic factors such as peer capacity, peer churn, peer mobility, network proximity, delivery latency, and traffic redundancy. However, a majority of P2P studies in the literature simply assume that *stationary* peer applications run on desktop computers with constant and plentiful bandwidth, and storage resources. Their research efforts are dedicated to performance optimization in static environments. Besides, many conventional P2P overlay designs[2] use unstructured and structured topologies (Androutsellis-Theotokis and Spinellis, 2004; Lua et al., 2005). Unstructured topologies are not scalable because they opt to flood redundant traffic regardless of peer population and network size (Barjini et al., 2012). Structured overlays are sophisticated and induce costly maintenance and management expenses. Accordingly, overlay maintenance and dynamics (Buford et al., 2009) are crucial, but not addressed well in P2P research communities. This situation motivates the thinking of a multi-dimensional way to cope with these factors in dynamic P2P networks. Not to patch up the P2P systems with separate means, the goal of this paper aims to design a systematic solution from the fundamental of P2P overlay infrastructures.

This paper proposes a hierarchical and cluster-based overlay architecture, abbreviated as HiCO, for P2P networking and content delivery services in a large-scale network environment. The design of a HiCO architecture exploits topological proximity, peer locality and reputation system, resulting in an adaptive overlay capable of self-organization and self-managing in dynamic networks. Referring to P2P overlay categories (Androutsellis-Theotokis and Spinellis, 2004), the HiCO is a *partially centralized* and *semi-structured* overlay architecture, where peers in a local cluster are not tightly controlled and the placement of media objects is not precisely determined to some tolerant degree. With topological and geographical partitions (Liu et al., 2003; Sharma et al., 2006), the way of peer clustering supports a variety of region scale. Peers in the same region form a peer cluster, i.e., a local peer community, and a peer's network-aware search can be confined in its physical network proximity. In concord with the specific cluster-based *reputation tree*, a hierarchical representation of peer clusters further turns into a hierarchical overlay. Hence, this elegant overlay design integrates three architectural functionalities below to address various factors in dynamic P2P networks.

- Partially centralized architecture
- Semi-structured architecture
- Reputation hierarchy

The design of a partially centralized P2P system specifies the role of superpeer, acting as a local central directory and a caching proxy for peers in each cluster of the P2P network. Superpeers are elected corresponding to reputation hierarchy if they have good repute and sufficient resources. They make indices of media objects shared by their local peers connected to them. All peers' queries are destined to superpeers in charge of content searching in the system. Thus, there are several benefits in a partially centralized system. First, the system gets rid of any single point of failure, since superpeers are dynamically assigned and can be replaced if they go down or fail. Second, superpeers undertake a

[1] A well metric of churn is node session time or life span – the time from the node's joining to its subsequent leaving from the system – varying from several hours to one minute (Bustamante and Qiao, 2008).

[2] For example, Napster and Freenet used pure unstructured topologies, BitTorrent, Gnutella (after 0.6 version) and Envoy (Joung and Lin, 2010) used two-tier topologies, and Chord, AnySee (Liao et al., 2006) and mTreebone (Wang et al., 2007) carried on ring, mesh, tree or hybrid structured topologies.

portion of entire network workload, unlike in a purely decentralized network where all peers can be equally and heavily loaded regardless of their service capabilities. Finally, superpeers can tackle inherent heterogeneity between different P2P systems when they are coupled with bridging functions for overlay federation in large-scale networks (Braun et al., 2008).

The semi-structured architecture strikes the balance between structured and unstructured ones from the viewpoints of search efficiency, system scalability, and resilience. A semi-structured overlay is built inside a "superpeer network" that consists of only superpeers in a P2P system. Such a overlay is tightly controlled under cluster hierarchy. Media objects, or pointers to them, are distributed to precise clusters and peers through superpeers, essentially providing a mapping between content and location references by means of distributed routing tables. Superpeers can route queries to target clusters and peers for the desired content. Thus, the superpeer network results in less cost of routing maintenance, rather than interconnecting all peers in a structured architecture. In addition, a semi-structured architecture has less maintenance overhead in response to varied peer population, unlike an unstructured design that is unscalable to the explosive content volume and network size (Lv et al., 2002). Finally but not at least important, as compared with the costly centralized directory, a semi-structured data management will not cause extra cost when peers still move in the same clusters in spite of high dynamics of peer population, churn or mobility.

The hierarchical organization is based on the cluster-based reputation tree to mitigate the free-riding problem. Since a peer cluster is the building block of the HiCO architecture, its reputation reckons in all peers' reputation credits in a cluster. The highest reputable peer in a cluster serves as a superpeer. Superpeers in the neighborhood further decide a new upper-level superpeer out of them to serve in an extended cluster that includes neighboring superpeers' clusters. Likewise, upper-level superpeers recursively decide their superpeer and organize the cluster hierarchy. Therefore, a peer's hop distance to its target superpeer is reversely proportionate to its reputation in the P2P system. Reputable peers have preemption in service with better quality of service and user experience, e.g., prioritized download capacity and fast query/response time. Punishment policy may be imposed on disreputable peers by different punitive levels of the free-riding severity, e.g., limiting message propagation from them, ignoring their queries, and disconnecting malicious/unproductive peers from the network (Hughes et al., 2005). Note that the cluster hierarchy is resilient to any points of failure at superpeers. A tie-in superpeer election process can fast coalesce the broken path. In addition, the cluster hierarchy is adaptive because the topology control process can timely perform cluster division and union according to peer proximity and balanced resource utilization.

The efforts of this study result in a hierarchical overlay architecture which is able to contend with the topological mismatch, peer churn and free riding problems in dynamic P2P systems. The proposed architecture not only takes advantage of joint partially centralized and semi-structured models to maintain system performance, but also uses the reputation hierarchy to guarantee service differentiation. This overlay design is flexible and resilient with self-organization. It can adjust the cluster hierarchy against varied peer population, unbalanced resource utilization, and any points of broken failure along topological paths. Furthermore, the HiCO architecture employs a special cluster-based signature technique for efficient P2P content search. Extensive simulations have been conducted to evaluate the HiCO architecture in terms of cluster scale, peer churn, query processing, and reputation incentive. Performance results show that the proposed overlay architecture is functional in system scalability, reliability, and service fairness.

The rest of this paper is organized as follows. Section 2 mentions the HiCO system model, terminology, and notation. Section 3 describes the HiCO overlay organization, components, and interactive processes. Section 4 shows the experimental results and performance evaluation. Conclusion is given in Section 5.

2. HiCO system model

This section describes the HiCO system model based on the joint partially centralized and semi-structured models. This model constructs a hierarchical overlay over peer population in light of peer locality, network proximity, and a specific reputation-tree structure.

Fig. 1 depicts the proposed overlay design with cluster-based, network-aware organization, where the cluster scale refers to varied peer population in network proximity rather than an uniform geographic size. For simplicity, this study adopts a grid mesh model for peer clustering with the maximal fan-out degree $d=4$. A cluster C is divided into or composed of four sub-clusters C_1, C_2, C_3 and C_4 in proximity where each sub-cluster is indexed counterclockwise. In a cluster hierarchy, the root cluster at the top level is denoted as C^0. The symbol $C^i_{j_1,\ldots,j_i}$ means that a cluster is located in a pairwise coordinate $\langle i\text{-th level}, j_1,j_2,\ldots j_i\text{-th sub-cluster}\rangle$ where the i-th level and the list of $j_1,j_2,\ldots j_i$ are figured out in a top-down sequence from the root. For example, as shown in Fig. 1, $C^3_{3,3,1}$ is an identifier of sub-cluster C_1 belonging to cluster $C^2_{3,3}$. Likewise, $C^2_{3,3}$ is a sub-cluster C_3 in C^1_3, and C^1_3 in C^0.

There are two basic types of peer: superpeer S, and local peer p or peer briefly. Every cluster $C^i_{j_1,\ldots,j_i}$ has a superpeer $S^i_{j_1,\ldots,j_i}$, whatever cluster scale it is. Local peers in the same cluster $C^i_{j_1,\ldots,j_i}$ are supposed to connect to $S^i_{j_1,\ldots,j_i}$. When a peer can reside in a location belonging to multiple clusters that are superposed with different region scales, a "direct peer" further defines a local peer that directly connects to its superpeer in a single-hop distance. For example, a peer in $C^3_{3,3,1}$ can reach another peer in C^1_2 in a multi-hop path by firstly connecting to $S^3_{3,3,1}$, ($S^2_{3,3}$), S^1_3, S^0, and S^1_2 in a four-hop path, where $S^2_{3,3}$ is identical to $S^3_{3,3,1}$ that also serves as the superpeer in $C^2_{3,3}$.

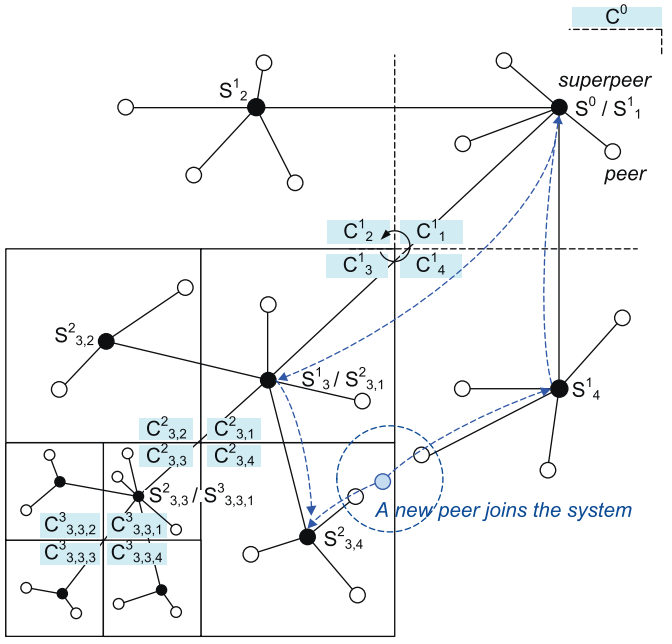

Fig. 1. Hierarchical P2P overlay topology with cluster-based and network-aware organization.

In a cluster $C^i_{j_1,\ldots,j_i}$, the peer with the highest reputation is served as the superpeer $S^i_{j_1,\ldots,j_i}$. Let $R^i_{j_1,\ldots,j_i}$ denote the reputation aggregate of all peers in cluster $C^i_{j_1,\ldots,j_i}$. With a set of four neighboring clusters $\{C^{i+1}_{j_1,\ldots,j_i(1|2|3|4)}\}$ that form in a super-cluster $C^i_{j_1,\ldots,j_i}$, their superpeers $\{S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}\}$ elect the one of the highest reputation as the upper-level, *parent* superpeer $S^i_{j_1,\ldots,j_i}$, and others become *child* superpeers in contrast to their parent. For example, as shown in Fig. 1, given with $C^2_{3,1}$ $C^2_{3,2}$ $C^2_{3,3}$ and $C^2_{3,4}$ in C^1_3, S^1_3 is identical to $S^2_{3,1}$ as the upper-level superpeer with its aggregate reputation of $R^1_3 = \sum_{k=1}^{d=4} R^2_{3,k}$. The same election process will run upward till the root level is reached. Let $N^i_{j_1,\ldots,j_i}$ denote the peer population, i.e., the number of peers in cluster $C^i_{j_1,\ldots,j_i}$. The peer population in a P2P system is $N = \sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \cdots \sum_{j_i=1}^{d} N^i_{j_1 j_2,\ldots,j_i}$ for any specific degree.

In regard to reputation representation, the assignment of a peer's reputation credit depends on which reputation system and incentive methods to be used. While the study of reputation system itself is important for P2P applications (Jin and Chan, 2010; Satsiou and Tassiulas, 2010), it is parallel to the work of this paper. The HiCO design does not intend to prescribe any new reputation mechanisms, but adopts a common measure of uploading contribution to reflect a peer's reputation credit. This allows the work of this paper to focus on the overlay design and development. For simplicity, this paper assumes that there exists prior knowledge and statistics of reputation records in P2P systems. All peers are assigned credits during system initialization.

3. HiCO architecture design

This section designs the HiCO system which consists of four primary mechanisms, including the basic peer clustering, topology control, content query and search, and resilience processes, respectively, described in Sections 3.2–3.5. Appendix A enclosed in "Supplemental Material" presents algorithmic forms for reference.

3.1. Design overview

The proposed HiCO architecture addresses peer- and network-initiated dynamics and influences in dynamic P2P network environments. To our best knowledge (Androutsellis-Theotokis and Spinellis, 2004; Lua et al., 2005; Buford et al., 2009), the design in this paper is the first attempt to integrate the hierarchical overlay with reputation system and network proximity simultaneously. This design focuses on hierarchical overlay construction, topology self-organization, and message protocols, and provides several major mechanisms:

- dynamic cluster division and union against peer churn and mobility,
- fault tolerance and resilience requirement, and
- reputation-based incentives for content query, search and delivery.

Accordingly, the HiCO system performs four primary processes – basic overlay organization, topology control, content search, and resilience – to fulfill the system functionality. Fig. 2 depicts the procedural flow among processes by system and peer sides.

- Basic overlay organization: *Join_Cluster* and *Election* processes are two fundamental processes called by other processes. The former is called into action by a direct peer or child superpeer to join a cluster. The latter is run by a group of direct peers or child superpeers to elect a new superpeer for their cluster. *Join_System process* is for a new peer to participate in the HiCO system. *Change_Cluster* process is for a peer's movement out of its original cluster to join another cluster.
- Topology control: A superpeer in charge can perform *Cluster_Division* and *Cluster_Union* processes to control resource utilization and avoid service loading saturation. *Re-election* process tries to edge out the cluster with less contribution to the topological boundary of the HiCO system.
- Content search: *Query_Handle* process is called when a superpeer receives a *query* message, from its direct peers or child superpeers, in quest of target content in the system.
- Resilience: *Failure_Recovery* process makes the system resilient to peer and network dynamics. A superpeer periodically sends a *probing* message to its direct peers and child superpeers to check their availability. Receiving peers send back *probing_response* messages to notify their superpeer of specific measure information, such as location, service capacity, resource utilization, etc.

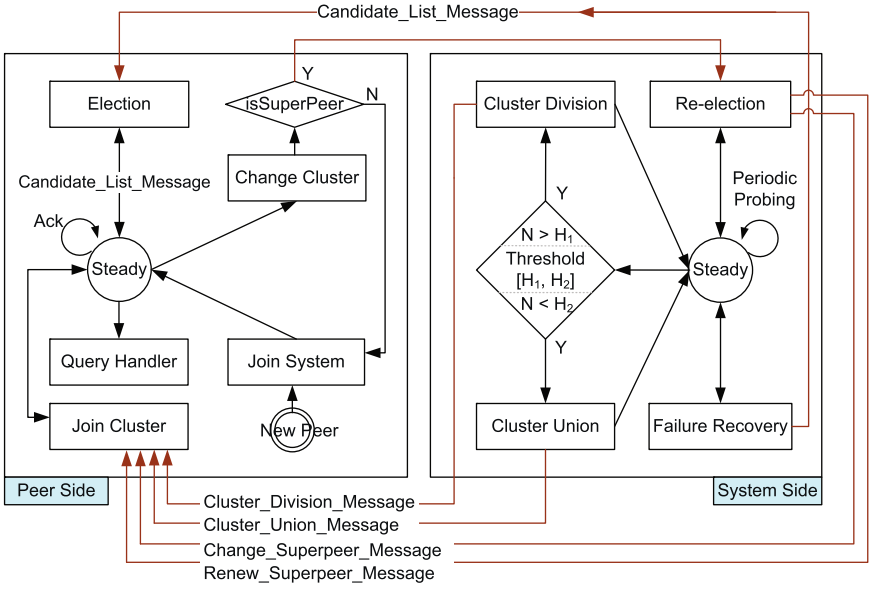

Fig. 2. Functional flowchart in the HiCO system.

3.2. Basic overlay organization

The HiCO system develops the overlay, basically and initially, by *Join_System*, *Change_Cluster*, *Join_Cluster*, and *Election* processes that operate at the peer side.

3.2.1. Join_System process

This process provides the *bootstrapping* service which performs in four steps for a new peer p to join the HiCO system. Firstly, when p appears in the network, it finds and connects with the superpeer S responsible for the current region. Assume that p is able to discover neighboring peers in its surrounding through bootstrapping means (Cramer et al., 2004; Wua et al., 2008). For example in Fig. 1, p can randomly choose some peers in its surrounding and try to connect with these peers. Secondly, after finding a neighboring peer that already exists in the system, p sends it a *join_system* message, including p's location reference. Thirdly, if the receiving peer is a direct peer, the *join_system* message is forwarded to its superpeer S. The superpeer checks p's location reference and decides if p can become its direct peer. If true, S replies to p by a *join_system_success* message including this superpeer's identifier $S^i_{j_1,\ldots,j_i}$ in a return path. If false, S checks whether p can become as a peer of one of its child superpeers $\{S^{i+1}_{j_1,\ldots,j_i,j_{(1|2|3|4)}}\}$. Finally, if p can be served by S's child superpeer, S forwards the *join_system* message to the target child superpeer. Otherwise, S will pass the *join_system* message upward to its parent superpeer $S^{i-1}_{j_1,\ldots,j_{i-1}}$ for further resolution in the same way. In light of the above procedures, the *Join_System* process can eventually determine the superpeer in charge of the cluster that p belongs to.

3.2.2. Change_Cluster process

When a peer p moves out of the boundary of its original cluster, its original superpeer shall coordinate p to connect with the target superpeer. In the design, a superpeer periodically probes its direct peers and child superpeers. Every peer adds its location information in the *probing_response* message. A superpeer is then aware of whether p is out of its cluster and further looks up which superpeer can take over to serve p. If yes, it sends p a *change_cluster* message including the target superpeer's location reference. As receiving this message, p sends a *join_system* message to the target superpeer to re-join in the system. Note that if the leaving peer is a superpeer, it still follows the above procedure to fulfill the cluster change. However, the old cluster needs a new superpeer in replace to serve other peers. This situation is similar to the case of a single point of failure that Section 3.5 will mention.

3.2.3. Join_Cluster process

This process is run by direct peers $p^i_{j_1,\ldots,j_i}$ or child superpeers $\{S^{i+1}_{j_1,\ldots,j_i,j_{(1|2|3|4)}}\}$ to join their upper-level cluster $C^i_{j_1,\ldots,j_i}$. As Fig. 2 shows, suppose that direct peers or child superpeers have received specific messages from other processes, i.e., *cluster_union*, *cluster_division*, *change_superpeer* and *renew_superpeer* messages that include the identifier of a target parent superpeer $S^i_{j_1,\ldots,j_i}$ to be connected. They send *join_cluster* messages to $S^i_{j_1,\ldots,j_i}$. This *join_cluster* message contains the sender's identifier and, if the sender is a child superpeer, the sub-cluster identifier. $S^i_{j_1,\ldots,j_i}$ then acknowledges those direct peers or child superpeers as new members in $C^i_{j_1,\ldots,j_i}$.

3.2.4. Election process

In cluster initialization and failure recovery processes, the system can deliver *candidate_list* messages to direct peers or child superpeers in a cluster to ask for superpeer election.

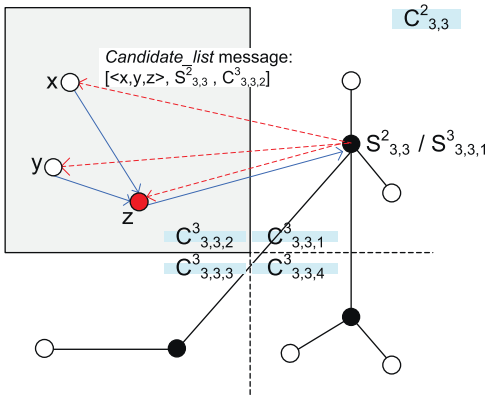

Fig. 3. Superpeer election procedure.

A *candidate_list* message includes three fields: a list of all candidates' identifiers, an identifier of an interim superpeer (i.e., an unsolved superpeer's parent), and an identifier of the cluster where the election is on-going. All peers in the candidate list report their profiles, including reputation and service capability, to the interim superpeer. According to specific policies, the peer with the highest credit is chosen as the new superpeer in the cluster. After that, the interim superpeer informs all peers in the cluster to invoke *Join_Cluster* process to connect with the new superpeer. This new superpeer also runs *Join_Cluster* process to connect with its parent superpeer specified in the *candidate_list* message.

To ease understanding, Fig. 3 depicts an example of superpeer election in an unsolved cluster. Peers x, y and z receive *candidate_list* messages, e.g., $[\langle x,y,z\rangle, S^2_{3,3}=S^3_{3,3,1}, C^3_{3,3,2}]$, from an interim superpeer $S^2_{3,3}$. Then, x, y and z report their profiles to $S^2_{3,3}$. Let z be the peer of the highest credit and elected as the new superpeer $S^3_{3,3,2}$. Peers x and y run *Join_Cluster* process to connect with $S^3_{3,3,2}$. Peer z also runs this process to become a child superpeer of $S^2_{3,3}$. Alternatively, peers in a candidate list may connect each other to elect their superpeer, if no interim superpeer is assigned. This case is possible when peers are in C^0 initially, or when no superpeer is adequate due to some kind of failure that will be resolved in Section 3.5.

3.3. Topology control

To cope with the problems of peer churn, mobility and free riding, this design proposes a topological control mechanism to avoid service saturation and improve resource utilization in the system. This mechanism performs two mutual procedures: *dynamic peer clustering* and *cluster marginalization* procedures.

- Dynamic peer clustering: The HiCO system adjusts the hierarchical overlay using a couple of *Cluster_Union* and *Cluster_Division* processes to perform the peer clustering procedure. A superpeer can divide its cluster $C^i_{j_1,\ldots,j_i}$ in a smaller granularity[3] when the number of peers in $C^i_{j_1,\ldots,j_i}$, i.e., $N^i_{j_1,\ldots,j_i}$, exceeds a specific threshold H_1. Oppositely, a union process can merge neighboring sub-clusters into a large one when $N^{i+1}_{j_1,\ldots,j_{i+1}}$ is lower than $H_2=H_1/d=H_1/4$ as $d=4$. Peer population $N^i_{j_1,\ldots,j_i}$ in a cluster $C^i_{j_1,\ldots,j_i}$ is thus controlled within a specific range $[\max,\min]=[H_1,H_2]$ to avoid fluctuant control between cluster union and division operations.

[3] The number of peers in a bottom/leaf cluster is controlled in a specific range. Although bottom clusters can have different geographic sizes, they have the same scale in terms of peer population.

- Cluster marginalization: The HiCO system integrates a specific reputation-tree incentive to moderate the free-riding problem. A cluster marginalization approach edges out the cluster with less contribution to the topological boundary. Functionally, a superpeer runs a periodic *Re-election* process to modify the topological levels among clusters in reflective response to their cluster reputations. A cluster with higher reputation can be rewarded higher priority in service. In a tree-like topology, therefore, a highly reputable cluster is arranged in an upper level than others in lower levels under its tree sibling. Peers in an upper cluster can receive faster responses to their queries and obtain their superpeers' resource and service capacities in an earlier stage.

In what follows, the subsections specify the cluster reputation formulation, and describe *Cluster_Division*, *Cluster_Union*, and *Re-election* processes that can execute in parallel to self-organize the overlay topology.

3.3.1. Cluster reputation formulation

Clusters at different levels have different peer populations and geographic sizes. The design of cluster reputation formulation reckons in average cluster reputation and peer density in a basic unit of cluster scale. Basically, a cluster reputation $R^i_{j_1,\ldots,j_i}$ presents the numerical measure of contributions to a particular cluster $C^i_{j_1,\ldots,j_i}$ where all peers' reputations aggregate in $R^i_{j_1,\ldots,j_i}$. Since clusters have various peer populations, an average cluster reputation, i.e., a cluster's reputation aggregate divided by its number of peers $N^i_{j_1,\ldots,j_i}$, is used instead to mitigate biased reputation patterns. For instance, few peers may contribute very high reputation credits in a cluster. Promoting such a cluster to an upper level leads to an unfair situation because most peers in this cluster have very less contribution. Thus, the use of average cluster reputation is beneficial in this situation. In addition, neighboring clusters at different levels need to be examined at the same time during overlay adjustment and management. Whereas an upper cluster may contain a larger amount of peers than a lower cluster has, peer density $D^i_{j_1,\ldots,j_i}$ in a basic unit of cluster scale can provide a relative measure in place of peer population. Accordingly, the measure function for the use in topology control process is expressed as $F(C^i_{j_1,\ldots,j_i}) = \alpha R^i_{j_1,\ldots,j_i}/N^i_{j_1,\ldots,j_i} + (1-\alpha)D^i_{j_1,\ldots,j_i}$, where α is tunable to upscale or downscale their weights of average cluster reputation and peer density, respectively.

3.3.2. Cluster_Division process

Fig. 4(a) and (b) illustrates the *Cluster_Division* process. In this design, a superpeer $S^i_{j_1,\ldots,j_i}$ periodically probes all direct peers in $C^i_{j_1,\ldots,j_i}$ to know their location references and reputation credits. If the average number of $N^i_{j_1,\ldots,j_i}$ is larger than H_1, $S^i_{j_1,\ldots,j_i}$ can start a *Cluster_Union* process. During the *Cluster_Division* process, as shown in Fig. 4(a), the superpeer classifies all peers in $C^i_{j_1,\ldots,j_i}$ into four sub-clusters $\{C^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}\}$ with their locations. In each sub-cluster, a peer with the highest reputation is chosen as the child superpeer $S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$. Then, the superpeers $S^i_{j_1,\ldots,j_i}$ sends *cluster_division* messages, indicating the identifiers of sub-cluster and child superpeer, to all peers in each sub-cluster $C^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$. As Fig. 4(b) shows, after receiving the *cluster_division* messages, peers in $C^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$ run the *Join_Cluster* process to connect with the newly assigned child superpeer $S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$. New child superpeers also operate *Join_Cluster* process to communicate with their parent superpeer $S^i_{j_1,\ldots,j_i}$. Till now, the *Cluster_Division* process is completed.

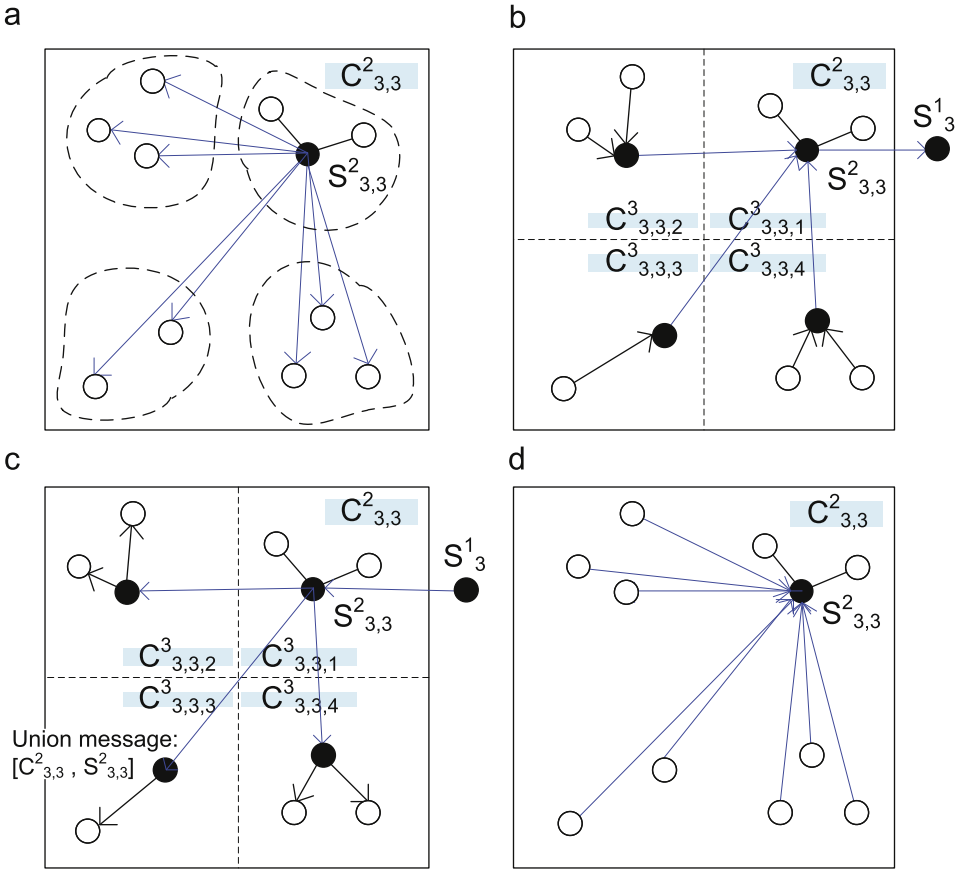

Fig. 4. *Cluster_Division* and *Cluster_Union* processes: (a) before division, (b) after division, (c) before union, and (d) after union.

3.3.3. Cluster_Union process

A super-cluster performs the *Cluster_Union* process to merge its direct peers and sub-clusters into a large cluster where all peers become direct peers. In each sub-cluster $C^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$, the child superpeer $S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$ senses the variance of $N^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$ through a periodic probing manner. Likewise, the parent superpeer $S^{i}_{j_1,\ldots,j_i}$ senses the variance of $N^{i}_{j_1,\ldots,j_i} = \sum N^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$. If the average number of $N^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}$ is smaller than H_2, $S^{i}_{j_1,\ldots,j_i}$ can force $\{S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}\}$ to start a *Cluster_Union* process. As Fig. 4(c) shows, $S^{i}_{j_1,\ldots,j_i}$ notifies all its direct peers and child superpeers to be merged by sending out *cluster_union* messages that include the cluster identifier $C^{i}_{j_1,\ldots,j_i}$ and the superpeer identifier itself to be connected with, e.g., $C^{2}_{3,3}$ and $S^{2}_{3,3}$. Every peer then sends $S^{i}_{j_1,\ldots,j_i}$ a *join_cluster* message to become as a direct peer in $C^{i}_{j_1,\ldots,j_i}$ as shown in Fig. 4(d).

3.3.4. Re-election process

The *Re-election* process adjusts the hierarchical topology of the superpeer network in compliance with specific cluster-based reputation incentives. With periodic probing information, all superpeers execute in parallel the *Re-election* process in bottom-up order upon the overlay hierarchy. In reference to Fig. 2, the superpeer in a bottom cluster first sends a *change_superpeer* message, including the cluster identifier, parent superpeer identifier and cluster signature, to some direct peer with the highest reputation as the new superpeer. The old superpeer then sends out *renew_superpeer* messages to notify all direct peers of the newly elected superpeer. When a direct peer receives a *renew_superpeer* message, it runs the *Join_Cluster process* to connect with the new superpeer. In addition, this new superpeer notifies the parent superpeer that it takes over the cluster using a *Join_Cluster process* as well. Further, a parent superpeer may further need to select a new superpeer reflectively in response to an update of its child superpeer. Same as the above steps, an old parent superpeer sends a *change_superpeer* message to the new one, and also sends *renew_superpeer* messages to its child superpeers. Then, all child superpeers and the new superpeer run the *Join_Cluster process* to connect with their parent superpeers. Likewise, intermediate superpeers on the upward path towards the root superpeer may correspondingly perform the *Re-election* process to adjust the topology.

As a result of the periodic *Re-election* process, the system keeps a cluster with less contribution as lower as possible in the cluster hierarchy. The *Re-election* process is enforced in each cluster level. The higher a cluster level is, the more the average cluster reputation it has. The root superpeer is the peer having the highest reputation in the network. The HiCO design guarantees that peers in upper clusters obtain better P2P service than those in lower clusters.

3.4. Content query and search

Content searching in the HiCO is based on the semi-structured superpeer network. This design employs the *signature* technique (Frakes and Baeza-Yates, 1992; Li et al., 2007) to improve search scalability and efficacy in dynamic P2P networks, rather than distributed hash tables (DHT), CAN and Chord for structured P2P systems (Lua et al., 2005) and flooding and random walks for unstructured ones (Barjini et al., 2012; Cholvi1 et al., 2004; Lv et al., 2002). Specifically, each object is assigned a bitwise signature using the bloom filter technique. All object identifiers in a cluster are compressed in a *cluster signature* (Chow et al., 2007) that is stored at this cluster's superpeer. A parent superpeer compresses many sub-cluster signatures from its child superpeers into a *compound cluster signature* by bitwise superimposition. Likewise, upper-level superpeers generate compound cluster signatures in larger cluster scales. In light of the cluster hierarchy, this design conducts content search using cluster signatures.

Only superpeers need to process the tasks of storing, exchanging and maintaining cluster signatures inside the superpeer network. During content searching, the hierarchical overlay routes any query messages over the superpeer network; no local peers are involved except the original peer sending the query and target peers. Hence, the cluster signature approach can better tolerate peer churn, mobility and failures in dynamic P2P networks. In addition, this approach can reduce database cost, allowing the P2P system to accommodate the increasing volume of P2P content.

3.4.1. Query handle

The query handle procedure includes three steps, as follows.

1. For accessing an object o_i, a peer $p^{i}_{j_1,\ldots,j_i}$ sends a *query* message to its superpeer $S^{i}_{j_1,\ldots,j_i}$. $S^{i}_{j_1,\ldots,j_i}$ first transforms o_i's identifier into a bitwise signature $g(o_i)$ using the bloom filter technique.
2. $S^{i}_{j_1,\ldots,j_i}$ performs a matching comparison between o_i's signature and cluster signature $g^{i}_{j_1,\ldots,j_i}$ to check if o_i is stored in the local cluster.
   (a) In case of matching, $S^{i}_{j_1,\ldots,j_i}$ replies $p^{i}_{j_1,\ldots,j_i}$ with a *query_response* message including o_i's location from which $p^{i}_{j_1,\ldots,j_i}$ can get o_i.
   (b) Otherwise, $S^{i}_{j_1,\ldots,j_i}$ forwards this *query* message, together with the indicated o_i's signature, to its parent superpeer $S^{i-1}_{j_1,\ldots,j_{i-1}}$.
3. $S^{i-1}_{j_1,\ldots,j_{i-1}}$ similarly compares o_i's signature with its compound cluster signature $g^{i-1}_{j_1,\ldots,j_{i-1}}$ in an extended super-cluster.
   (a) In an unmatched case, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ forwards the query message to its parent superpeer, upward and iteratively, to search the target object, till the root superpeer is reached.
   (b) Else, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ finds out which child superpeer *may* contain o_i, by comparing o_i's signature with each $g^{i}_{j_1,\ldots,j_{i-1},(1|2|3|4)}$. Then, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ sends a query message to every matched child superpeer. The child superpeer runs the same checking process, downward and iteratively, till the target cluster is located or that a query already reaches the bottom cluster.

Accordingly, the content search performs a series of upward and downward query operations to look for the target object along a hierarchical path, thereby resulting in a logarithmic-scale search cost.

Fig. 5 instantiates the query handle for exposition. Let a peer x in $C^{3}_{3,3,4}$ query its superpeer $S^{3}_{3,3,4}$ about an object o_1. $S^{3}_{3,3,4}$ generates o_1's signature $g(o_1) = 110011000001$ and compares $g(o_1)$ with this cluster signature $g^{3}_{3,3,4}$ by a bitwise *AND* operation. $S^{3}_{3,3,4}$ finds that o_1 does not exist in $C^{3}_{3,3,4}$, and forwards the query and $g(o_1)$ to its parent superpeer. $S^{2}_{3,3}$ runs an *AND* operation on $g(o_1)$ and its compound signature $g^{2}_{3,3}$. A positive results means that o_i *may* exist in $C^{2}_{3,3}$. Then, $S^{2}_{3,3}$ runs *AND* operations on $g(o_1)$ and every sub-cluster signature $g^{3}_{3,3,(1|2|3|4)}$. $S^{2}_{3,3}$ gets a positive result in the case of $g^{3}_{3,3,2}$ and further routes the query to $S^{3}_{3,3,2}$. $S^{3}_{3,3,2}$ looks $g(o_1)$ up in its signature table and gets a positive match. Because $C^{3}_{3,3,2}$ is a bottom cluster, $S^{3}_{3,3,2}$ can ascertain whether o_1 is locally available or not due to unpredictable peer churn or mobility. If true, eventually, a positive *query_response* message, including o_1's location reference, is sent directly from $S^{3}_{3,3,2}$ to $S^{3}_{3,3,4}$ and peer x, and alternatively backward along the query forwarded path to peer x. Otherwise, a false *query_response* is returned backward by the routing path to peer x. Every intermediate superpeer $S^{i-1}_{j_1,\ldots,j_{i-1}}$ can log a misleading record of

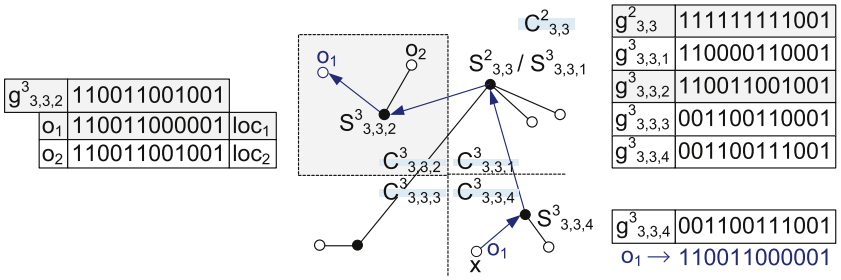

Fig. 5. Content search process.

$[o_1, g(o_i), S^i_{j_1,\ldots,j_i}]$ in the false routing table to avoid forwarding the same query in the future.

3.4.2. Usage on cluster signature

The generation of object and cluster signatures applies the bloom filter technique to represent a set of n data objects, $O=\{o_1,o_2,\ldots,o_n\}$, where every object o_i is assigned a bitwise signature $g(o_i)$ with m bits. A cluster signature $g^i_{j_1,\ldots,j_i}$ is a bloom filter that superimposes all object signatures in $C^i_{j_1,\ldots,j_i}$. Specifically, given with an object set O, a vector with m bits, $V=\{v_1,v_2,\ldots,v_m\}$, is initially set to zero. There are k independent hash functions, $h_1,h_2,\ldots,h_k$, which have the same codomain between 1 and m. To construct a bloom filter, every object o_i in O is hashed by k hash functions to produce k hash values, $h_1(o_i),h_2(o_i),\ldots,h_k(o_i)$. Then, the corresponding bits in V at the positions of k hash values are set to be one, i.e., $\forall j\in[1,k]$, $v_{h_j(o_i)}=1$. Thus, a cluster signature is produced after all objects in O are reflected by V.

To check whether an object o_i is subsumed in V, o_i is first hashed by the k hash functions. If all bits at positions of the k hash values are set in V, o_i is "probably" in V. Otherwise, o_i is definitely not in V. Note that a bloom filter may result in a *false positive* – the existence of an object, but it is actually not. Due to the property of bitwise superimposition, there is a probability that some bits in V are randomly set to be one at multiple times by different objects in O. When the bits at the positions of $h_1(o_i),\ldots,h_k(o_i)$ in V have already been set by other objects, a bitwise comparison between $g(o_i)$ and V still indicates a match, albeit it is not true.

For mathematical simplicity, it is commonly assumed that all bits are independently set to zero or one (Chow et al., 2007; Fan et al., 2000). After hashing all elements in O into V, the probability that a certain bit in V still remains zero is $(1-1/m)^{nk}$. The false-positive probability is given as $p_{error}=\left[1-(1-1/m)^{nk}\right]^k$, where m is the bit length of a signature for $g(o_i)$ in V, n is the number of objects in O, and k is the number of hash functions. Furthermore, the optimal number of hash functions is minimized as $k=\ln 2\times m/n$. As noted in Fan et al. (2000), Table 1 presents some false-positive data that Section 4.5 will refer to.

Hence, the signature-based content search mechanism has several properties. First, a misleading false-positive case may occur owing to the signature methodology (Frakes and Baeza-Yates, 1992). Second, it is possible that multiple cluster signatures come out the true in process of content search. Only one out of the matched cluster signatures is correct; other misleading cases may cause futile traffic overhead. Nevertheless, the queried object can be found as long as it exists in the system. Third, the misleading probability can be reduced to an accepted extent by several approaches, e.g., increasing signature length, manipulating appropriate hashing functions, and putting down a misleading log. Finally, a signature cache can further assist in query handling and improve the system performance, especially while object popularity and skewed access patterns are considered (Chow et al., 2007).

Table 1
False-positive probability.

$\frac{m}{n}$	k	p_{error}
6	4	0.0561
8	6	0.0215
12	8	0.00314

3.5. Resilience

The HiCO architecture develops the *Failure_Recovery* process to ensure resilience against unpredictable faults in dynamic environments, e.g., peer failure and disconnected topology. To remedy such situations, the *Failure_Recovery* process specifies two types of resilience process, *Single_Failure_Recovery* and *Multi_Failure_Recovery* processes. The effect of recovery processing is confined in a coverage of contiguous clusters without relation to other siblings in a tree-like topology. Thus, the recovery processes can run in parallel to resolve concurrent points of failure in the system.

3.5.1. Recovery process to single point of failure

Let a peer x in the system run into failure. The system remains unaffected if x is a direct peer. Otherwise, if x is a superpeer $S^i_{j_1,\ldots,j_i}$ in a cluster $C^i_{j_1,\ldots,j_i}$, the system reacts to a single point of failure. Remember that a superpeer periodically probes its direct peers and child superpeers $\{S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}\}$ to check their availability. A *probing* message contains the identifier and location reference of its parent superpeer, its identifier and location reference, and its cluster identifier. When x's parent superpeer perceives the absence of some child superpeer, it invokes the *Single_Failure_Recovery* process to repair the disconnected topology.

The *Single_Failure_Recovery* process performs two phases in response to any single point of failure.

- Temporary recovery phase: Direct peers and child superpeers of a failed superpeer $S^i_{j_1,\ldots,j_i}$ in $C^i_{j_1,\ldots,j_i}$ temporarily connect to the parent superpeer $S^{i-1}_{j_1,\ldots,j_{i-1}}$.
- Fully recovery phase: $S^{i-1}_{j_1,\ldots,j_{i-1}}$ coordinates all direct peers and child superpeers in $C^i_{j_1,\ldots,j_i}$ to re-elect a new superpeer.

Temporary recovery phase: Assume that direct peers and child superpeers in $C^i_{j_1,\ldots,j_i}$ have not received any *probing* messages from their superpeer $S^i_{j_1,\ldots,j_i}$ over a time threshold. They consider that $S^i_{j_1,\ldots,j_i}$ is failed or inexistent due to unpredictable reasons such as outage, churn and mobility. With $S^{i-1}_{j_1,\ldots,j_{i-1}}$'s identifier and location reference indicated in previous *probing* messages, $S^i_{j_1,\ldots,j_i}$'s direct

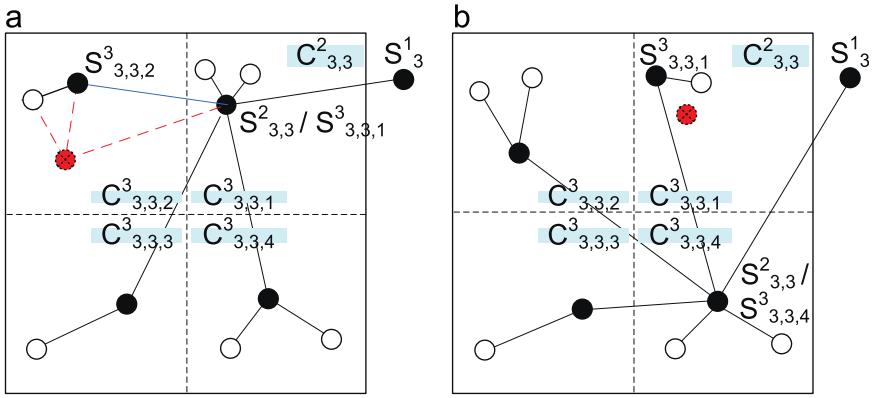

Fig. 6. *Single_Failure_Recovery* process: (a) $S^i_{j_1,\ldots,j_i}$ is a simple superpeer in $C^i_{j_1,\ldots,j_i}$, and (b) $S^i_{j_1,\ldots,j_i}$ is a dual superpeer in $C^i_{j_1,\ldots,j_i}$ and $C^{i+1}_{j_1,\ldots,j_{i+1}}$.

peers and child superpeers connect to $S^{i-1}_{j_1,\ldots,j_{i-1}}$ by the same way as the *Join_Cluster* process. During the recovery transition, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ serves an interim superpeer to acknowledge those direct peers and child superpeers as new members in $C^{i-1}_{j_1,\ldots,j_{i-1}}$. After this phase, the disconnected topology affected by the failed $S^i_{j_1,\ldots,j_i}$ is temporarily fixed. $S^{i-1}_{j_1,\ldots,j_{i-1}}$ handles any content queries from $C^i_{j_1,\ldots,j_i}$ before the second phase decides a new $S^i_{j_1,\ldots,j_i}$ to supplant the interim one.

Fully recovery phase: $S^{i-1}_{j_1,\ldots,j_{i-1}}$ is aware of the failure of $S^i_{j_1,\ldots,j_i}$ after receiving many *join_cluster* messages from $C^i_{j_1,\ldots,j_i}$. $S^{i-1}_{j_1,\ldots,j_{i-1}}$ coordinates the new superpeer election and topology recovery in $C^i_{j_1,\ldots,j_i}$. Particularly, this phase treats two cases, Cases 1 and 2, as Fig. 6 illustrates.

1. When $S^i_{j_1,\ldots,j_i}$ is a "simple superpeer" in $C^i_{j_1,\ldots,j_i}$, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ starts an *Election* process to determine a new superpeer in place of the failed one. For example in Fig. 6(a), $S^2_{3,3}$ coordinates the *Election* process to elect a new $S^3_{3,3,2}$.
2. When $S^i_{j_1,\ldots,j_i}$ is a "dual superpeer" in both $C^i_{j_1,\ldots,j_i}$ and $C^{i+1}_{j_1,\ldots,j_{i+1}}$, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ runs a two-step election process. Firstly, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ coordinates the *Election* process to elect a new $S^{i+1}_{j_1,\ldots,j_{i+1}}$. Secondly, with all child superpeers $\{S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}\}$ of the old failed $S^i_{j_1,\ldots,j_i}$, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ determines the one of the highest cluster reputation as the new $S^i_{j_1,\ldots,j_i}$. After that, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ informs $\{S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}\}$ to run *Join_Cluster* processes to link up with $S^i_{j_1,\ldots,j_i}$. As the example in Fig. 6(b) where $S^2_{3,3}/S^3_{3,3,1}$ was absent, S^1_3 coordinates the *Election* process to elect a new $S^3_{3,3,1}$ in $C^3_{3,3,1}$ as well as a new $S^2_{3,3}/S^3_{3,3,4}$ in $C^2_{3,3}$. Then, the other child superpeers $\{S^3_{3,3,(1|2|3)}\}$ connect with $S^2_{3,3}$ as their parent superpeer $S^2_{3,3}$.

3.5.2. Recovery process to multiple points of failure

The *Multiple_Failure_Recovery* process resolves the critical situation by multiple consecutive points of failure on the hierarchy topology. To ease exposition, the recovery process in a simple case of two consecutive points of failure is described firstly. Then, a generic recovery process is given to resolve any number of consecutive points of failure along a topological path.

Given with two consecutive failed points $S^i_{j_1,\ldots,j_i}$ and $S^{i+1}_{j_1,\ldots,j_{i+1}}$, as shown in Fig. 7(a), direct peers in $C^{i+1}_{j_1,\ldots,j_{i+1}}$ of the failed $S^{i+1}_{j_1,\ldots,j_{i+1}}$ and child superpeers $\{S^{i+2}_{j_1,\ldots,j_{i+1},(1|2|3|4)}\}$ under $S^{i+1}_{j_1,\ldots,j_{i+1}}$ will not receive any probing messages from $S^{i+1}_{j_1,\ldots,j_{i+1}}$. Besides, all of them cannot appeal to $S^i_{j_1,\ldots,j_i}$ for failure recovery, while $S^i_{j_1,\ldots,j_i}$ is failed too. In this situation, however, direct peers in $C^i_{j_1,\ldots,j_i}$ of the failed $S^i_{j_1,\ldots,j_i}$ and other alive child superpeers $\{S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}\}$ under $S^i_{j_1,\ldots,j_i}$ can still remember $S^i_{j_1,\ldots,j_i}$'s parent superpeer $S^{i-1}_{j_1,\ldots,j_{i-1}}$ that is indicated in previous probing messages. They can appeal to $S^{i-1}_{j_1,\ldots,j_{i-1}}$ to serve

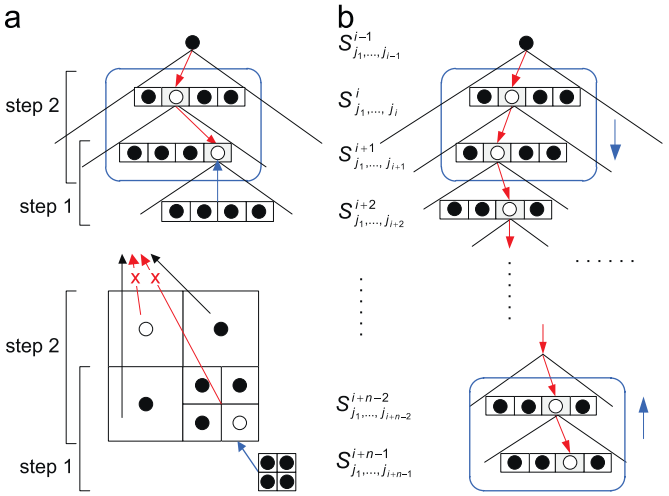

Fig. 7. *Multiple_Failure_Recovery* process: resolving multiple failures in sequence along the tree topology. (a) Two consecutive points of failure and (b) recursive recovery.

as an interim superpeer. Thus, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ is able to reflectively learn which sub-cluster $C^{i+1}_{j_1,\ldots,j_{i+1}}$ encounters a failed $S^{i+1}_{j_1,\ldots,j_{i+1}}$ and needs a new superpeer. $S^{i-1}_{j_1,\ldots,j_{i-1}}$ then enforces all peers in the target $C^{i+1}_{j_1,\ldots,j_{i+1}}$, including direct peers and child superpeers $\{S^{i+2}_{j_1,\ldots,j_{i+1},(1|2|3|4)}\}$, to run *Election* processes to appoint a new superpeer $S^{i+1}_{j_1,\ldots,j_{i+1}}$. As $S^{i-1}_{j_1,\ldots,j_{i-1}}$ has all child superpeers $\{S^{i+1}_{j_1,\ldots,j_i,(1|2|3|4)}\}$, the one of the highest cluster reputation is chosen as the new $S^i_{j_1,\ldots,j_i}$ in $C^i_{j_1,\ldots,j_i}$.

The generic resolution is a downward *recursive procedure* that repeatedly operates the "recovery pattern" – the process of remedying two consecutive failed superpeers. Let the system suffer from a sequence of n failed superpeers downward along a topological path, i.e., $S^i_{j_1 j_2,\ldots,j_i},\ldots,S^{i+n-1}_{j_1 j_2,\ldots,j_{i+n-1}}$, as depicted in Fig. 7(b). Firstly, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ follows the recovery pattern to resolve concurrent failures at $S^i_{j_1,\ldots,j_i}$ and $S^{i+1}_{j_1,\ldots,j_{i+1}}$. Then, $S^{i-1}_{j_1,\ldots,j_{i-1}}$ can have three out of four child superpeers in sub-clusters $\{C^i_{j_1,\ldots,j_{i-1},(1|2|3|4)}\}$. To resolve the failed one in $\{C^i_{j_1,\ldots,j_{i-1},(1|2|3|4)}\}$, $S^i_{j_1,\ldots,j_i}$ has to perform the recovery pattern to recover concurrent failures at $S^{i+1}_{j_1,\ldots,j_{i+1}}$ and $S^{i+2}_{j_1,\ldots,j_{i+2}}$ because $S^{i+2}_{j_1,\ldots,j_{i+2}}$ is also failed. Likewise, $S^{i+1}_{j_1,\ldots,j_{i+1}}$ performs the recovery pattern again to recover $S^{i+2}_{j_1,\ldots,j_{i+2}}$ and $S^{i+3}_{j_1,\ldots,j_{i+3}}$. Notice that this procedure leads to a recursion of operating recovery patterns till the last and lowest superpeer is met and determined as $S^{i+n-1}_{j_1 j_2,\ldots,j_{i+n-1}}$. Then, the recursive procedure returns to the preceding step in $C^{i+n-2}_{j_1 j_2,\ldots,j_{i+n-2}}$ to figure out a new $S^{i+n-2}_{j_1 j_2,\ldots,j_{i+n-2}}$. The return to a preceding step is run over and over till the first and highest $S^i_{j_1 j_2,\ldots,j_i}$ is determined.

4. Performance results

This section examines the proposed HiCO architecture in terms of performance sensitivities to system scalability by peer population, service fairness by reputation incentive, time and messaging traffic overheads by peer cluster adjustment, and query response time and false positive rate by content search.

4.1. Simulation environment

This study develops a simple simulator according to the pseudo-programs in Appendix (Supplemental Material) to evaluate the proposed HiCO architecture under a dynamic P2P network environment. The simulator conducts extensive experiments in a synthetical P2P system based on a discrete and slotted time-based model. The simulated system contains 16 384 (2^{14}) peers in a spatial scope of 10000×10000 square units. For generality, peer mobility pattern follows the time-based random waypoint model (Bettstetter et al., 2003; Nayebi et al., 2007). Peers move and pause independently and randomly in a time horizon. As the motion step starts, every peer randomly chooses a destination in the space, a motion time T_m in the interval $[T_{min},T_{max}]=[20,50]$, and a motion distance D_m in the interval $[D_{min},D_{max}]=[10,150]$; accordingly, the peer adjusts its motion velocity motion distance $V_m=D_m/T_m$. At the end of each motion, every peer pauses for some time span T_p that is chosen uniformly at random in the interval $[0,T_m]$, and then repeats the motion step until the simulation is terminated.

The simulation performs in two phases. Firstly, each peer is free to join the system at random. A new peer selects a target cluster and sends *join_system* request messages to neighboring peers in its surrounding. When an existent peers receives this message, it is obligated to forward this message to the local superpeer in charge. Reflectively, the HiCO architecture may perform cluster division, union and/or re-election processes to adjust the overlay topology. Secondly, when the system reaches a stable state, peers are free to launch query messages to their superpeers which handle content search and query processing tasks.

4.2. Sensitivity to cluster scale

This subsection inspects the scalability of the HiCO architecture for hierarchical overlay organization under variance of cluster scale, i.e., the maximal number of peers in a bottom cluster. Basically, the topology depth is smaller with a larger cluster scale; so is the smaller amount of *join_system* messages forwarded from the superpeers in charge to the root. Fig. 8(a) depicts the number of the bottom clusters and the average number of peers in each bottom cluster, i.e., peer density in the system. As the division threshold increases, a cluster can contain more peers and be much tolerant to dynamics of peer churn and mobility that will be further investigated in Section 4.3. Meanwhile, the depth of hierarchical overlay is smaller, and so is the number of clusters. A new peer can thus faster reach the target superpeer in the system. The results in Fig. 8(b) confirm this observation that the average response time for a *join_system* request decreases by order of magnitude when the cluster scale varies from 32 to 16 384. As the cluster scale is 16 384, the system turns into a centralized system with only one superpeer. It then takes 3 time units: one to send, one to process, and one to reply to a *join_system* request under a discrete-event simulation model. Obviously, the HiCO system conforms to $O(\log N)$ complexity of running time under a hierarchical structure.

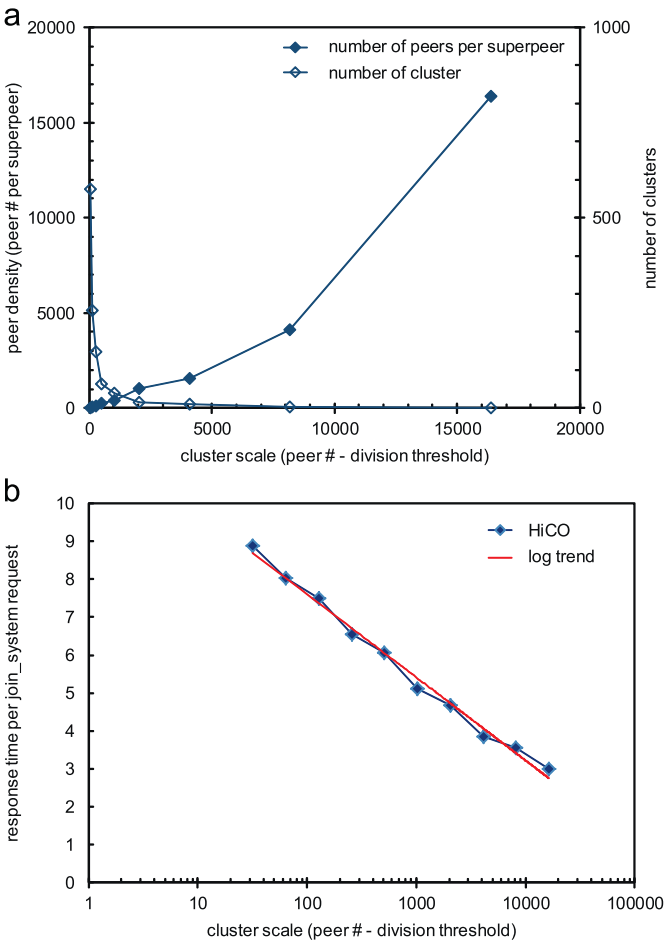

Fig. 8. Sensitivity to cluster scale (2^5–2^{14}). (a) Number of bottom clusters versus peer density and (b) response time for joining the system.

4.3. Sensitivity to peer churn and mobility

This subsection examines the sensitivity to peer churn and mobility in terms of times of cluster division/union and messaging overheads. In simulation, at each time unit, every peer moves to a new coordinate according to its mobility pattern. As a result of a peer's leaving, if the number of peers in the original cluster is lower than a specific union threshold, the peer's original cluster is merged with neighboring clusters. Simultaneously, with a peer's joining in a new target cluster, if the number of peers in the new cluster exceeds the specific division threshold, this cluster will be divided into four clusters. Since both union and division processes can recursively involve many clusters, superpeers and peers according to the overlay topology, these processes can undergo various times and induce messaging overheads, the event which is investigated below.

Fig. 9 depicts the totals of times of cluster division/union and messages, generated by cluster division and union processes, over the simulation that executes 1800 time units after all peers have entered the system. To facilitate the scrutiny of threshold granularity, the difference between union and division thresholds, called as *delta threshold*, is given as $1-\textit{union threshold}/\textit{division threshold}$ with a value range of [0,0.5] to present a comparative baseline. That is, union threshold is a factor of division threshold.

As shown, the increase of delta threshold reduces the times of cluster division and union since the gap between division and union threshold is larger. Both time and message costs become lower. However, this situation can degrade the system efficiency because the overlay topology tends to be steady with rare changes.

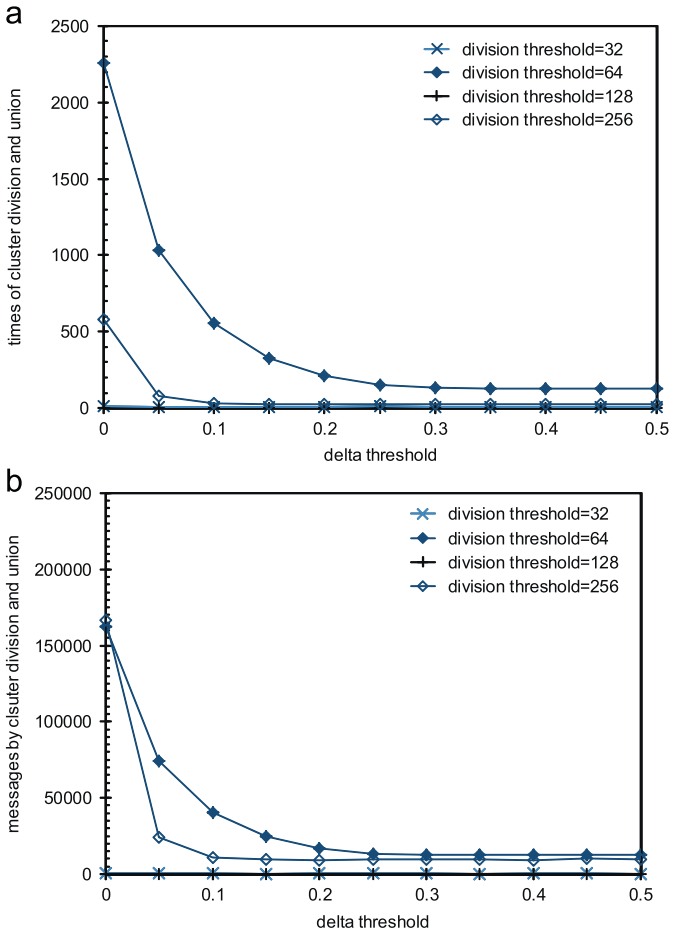

Fig. 9. Sensitivity to threshold variance of cluster division and union. (a) Times of cluster division and union and (b) total messages by cluster division and union.

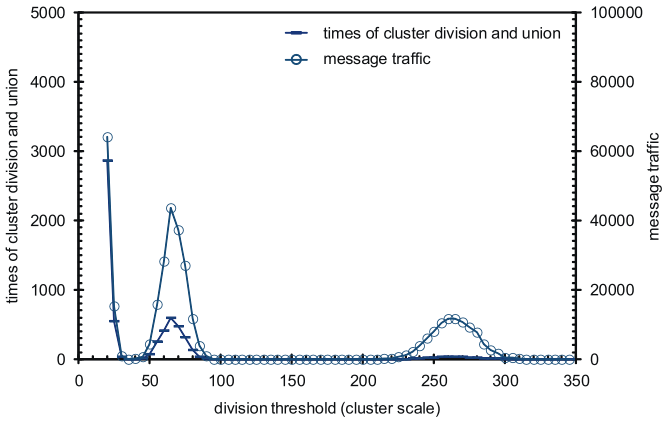

Fig. 10. Sensitivity to peer churn and mobility.

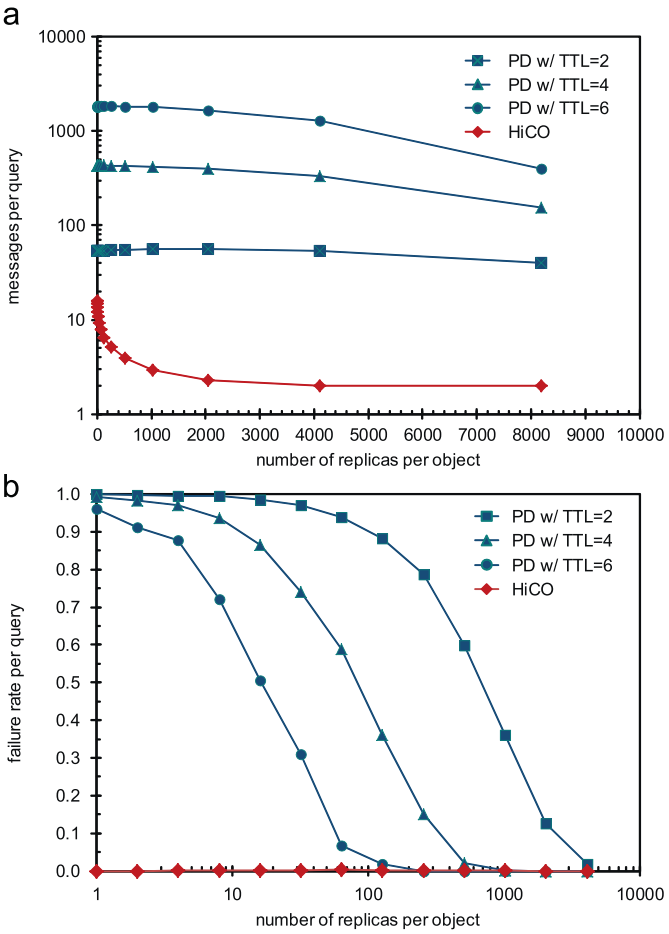

Fig. 11. Sensitivity to query performance. (a) Message traffic and (b) failure rate.

Accordingly, there should be some appropriate values of delta threshold in response to various division/union thresholds. To scrutinize this observation, Fig. 10 shows additional experiments where delta threshold is set constantly at 0.1 and the values of division threshold are tuned from 20 to 345. Time and messaging costs in most cases decrease with a larger division threshold. Notwithstanding, it is worthy to note that the curves rise sharply when the division threshold is near to either 64 or 256. It is found that in the cases of 64 and 256 the number of peers in the bottom cluster is very close to the division and union thresholds. Such a cluster is too sensitive to be divided and merged in response to peer mobility. This observation implies that the use of an appropriate division threshold can avoid the fluctuant adjustment of overlay topology. Experimental results show that it is beneficial to choose a division threshold whose value of log(*peer population*/*division threshold*) is apart from the power of fan-out degree used in the topology construction. For example, as depicted in Fig. 10, 64(= 16 384/256) and 256(= 16 384/64) are not suitable because they are in the power of $d=4$ with an exponent 3 and 4, respectively.

4.4. Effect by query process

This subsection examines the effect of content searching in the HiCO system in terms of query response time and failure rate. The purely distributed (PD) P2P model with the flooding search and time-to-live (TTL) limits is referred as a comparative baseline. Fig. 11 shows the relative performance under a sequence of queries from different peers' asking for the same object. Let a superpeer keep a replica of answered object for its peer's query. The more queries for that object, the more number of object replicas the system contains. As Fig. 11(a) shows, the HiCO is much superior to the PD scheme in terms of message traffic because the HiCO forwards query messages in a hierarchical superpeer network in contrast with the PD scheme that broadcasts numerous messages to neighboring peers in the system. Significantly, the HiCO need not concern the query failure owing to its semi-structured and tie-in fault recovery mechanisms.

The PD scheme, however, induces a high failure rate under variances of the value of TTL limit and the amount of object replicas in the system. Comparing Fig. 11(a) and (b) shows that although increasing TTL limits can lower the failure rate of the PD scheme, the traffic overhead rises drastically, resulting in a trade-off between traffic redundancy and reliability.

4.5. Effect by signature's bitwise length

This subsection further scrutinizes the searching efficiency influenced by the cluster signature approach. Fig. 12 depicts the performance sensitivity to the ratio (m/n) of the signature's bit length m to the number of object types n in the system. When the bit length of a signature is large, the false positive rate and searching failure rate are small, since a long signature can present much more objects than a short one. The HiCO attains a lower failure rate with a smaller value of m/n in comparison with the results in Table 1. For example, given with $m/n=6$, the HiCO gets a failure rate <0.01 that is much lower than 0.056 in Table 1. In addition, as $m/n=3$, the false positive rate is less than 3.5%, and the searching failure rate is less than 1.6%. This can be explained that the HiCO distributes objects in different clusters, so that the number of object types in a certain cluster is smaller than the total of object types in the system. Thus, a superpeer in a cluster-based topology can still achieve searching efficiency with a shorter signature.

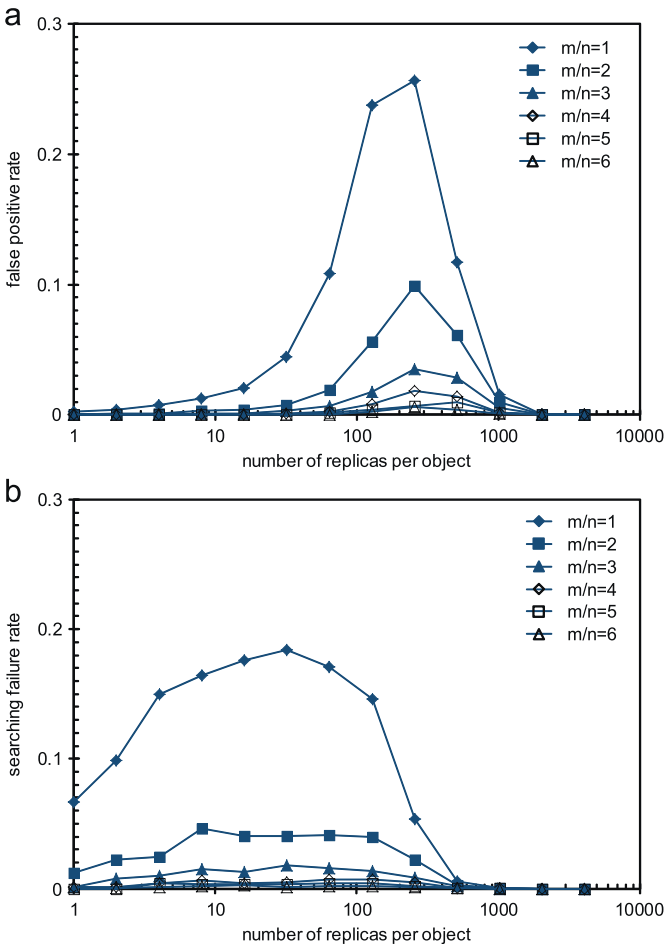

Fig. 12. Sensitivity to signature settings of m/n. (a) False positive rate and (b) failure rate.

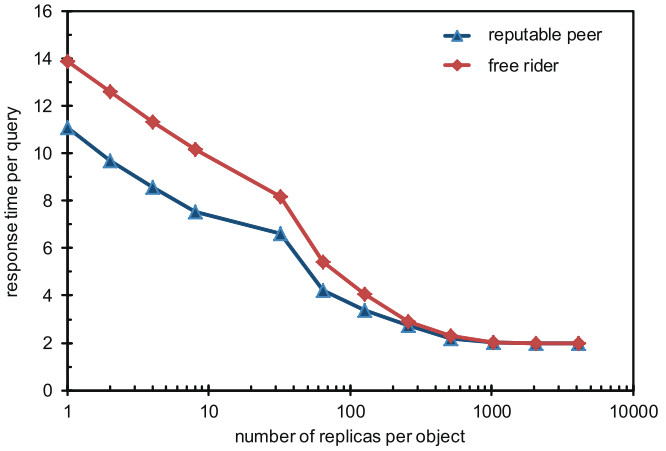

Fig. 13. Sensitivity to reputation incentive (100 000 peers, Zipf skew coefficient $\theta=1$, and cluster scale 16).

4.6. Effect by reputation incentive

This subsection investigates the effects of the cluster-based reputation incentive to alleviate the free-riding problem. The simulation utilizes the Zipf-like distribution (Zipf, 1949; Yu et al., 2006) to assign the reputation value of each bottom cluster in the HiCO system. Peers in the clusters of the top 10% reputation are considered reputable peers, and those peers in the clusters of the last 10% reputation are free riders.

The effects by cluster-based reputation incentive are prominent. When *Re-election* process executes, the topological control mechanism elevates the levels of clusters with high reputations in the overlay hierarchy. A peer located in a high-reputation cluster can obtain better quality of content searching than other peers with lower reputations. As Fig. 13 depicts, a reputable peer achieves lower query response time than free riders. Peers in a lower cluster traverse a longer distance, upward to the root and then downward to the target cluster, resulting in a large number of hops. By contrast, peers in an upper cluster can faster find out its target object through a shorter topological path. When the number of replicas increases, the time difference between these two cases remains distinct. As a result of this cluster-based reputation incentive, the HiCO is able to perform service differentiation between reputable peers and free riders.

5. Conclusion

This paper has proposed the HiCO architecture which is a *partially centralized* and *semi-structured* overlay architecture for large-scale P2P services in a dynamic network environment. The HiCO design integrates topological proximity, peer locality and mobility, and reputation-based incentive simultaneously. In compliance with cluster-based reputation hierarchy, the HiCO overlay is capable of self-organization and self-management by dynamic cluster adjustment and superpeer election mechanisms. In addition, the HiCO overlay is resilient to any points of superpeer failure by specific recovery pattern and resolution mechanisms. Therefore, the HiCO system can not only provide network-aware P2P services, but also alleviate the peer churn and free riding problems to a significant extent. Extensive simulations have evaluated its system performance in terms of overlay maintenance, content query, and traffic redundancy. Experimental results confirm that the HiCO architecture attains scaled-up performance and achieves service efficacy, fairness, and reliability for dynamic P2P systems.

Acknowledgement

This work was supported in part by the National Science Council of Taiwan, ROC, under Contracts NSC99-2221-E-008-011 and NSC100-2221-E-008-085-MY3.

Appendix A. Supplementary material

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.jnca.2012.07.022.

References

Adar E, Huberman BA. Free riding on gnutella. First Monday 5, October 2000.
Androutsellis-Theotokis S, Spinellis D. A survey of peer-to-peer content distribution technologies. ACM Computing Surveys 2004;36(December):335–71.
Barjini H, Othman M, Ibrahim H, Udzir NI. Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured P2P networks. Peer-to-Peer Networking and Applications 2012;5:1–13.
Bettstetter C, Resta G, Santi P. The node distribution of the random waypoint mobility model for wireless ad hoc networks. IEEE Transactions on Mobile Computing 2003;2(July–September):257–69.
Biskupski B, Dowling J, Sacha J. Properties and mechanisms of self-organizing MANET and P2P systems. ACM Transactions on Autonomous and Adaptive Systems 2007;2(March).
Bolla R, Gaeta R, Magnetto A, Sciuto M, Sereno M. A measurement study supporting P2P file-sharing community models. Computer Networks 2009;53:485–500.
Braun D, et al. UP2P: a peer-to-peer overlay architecture for ubiquitous communications and networking. IEEE Communications Magazine 2008;46(December):32–9.
Buford JF, Yu H, Lua EK. P2P Networking and Applications. USA: Morgan Kaufmann Publishers; 2009.
Bustamante FE, Qiao Y. Designing less-structured P2P systems for the expected high churn. IEEE/ACM Transactions on Networking 2008;16(June):617–27.
Cholvi1 V, Felber P, Biersack E. Efficient search in unstructured peer-to-peer networks. European Transactions on Telecommunications 2004;15:535–48.
Chow CY, Leong HV, Chan AT. Grococa: group-based peer-to-peer cooperative caching in mobile environment. IEEE Journal on Selected Areas in Communications 2007;25(January):179–91.
Chow CY, Mokbel MF, Leong HV. On efficient and scalable support of continuous queries in mobile peer-to-peer environments. IEEE Transactions on Mobile Computing 2011;10:1473–87.
Cohen B. Incentives build robustness in BitTorrent. In: Proceedings of the 1st workshop on economics of peer-to-peer systems, 2003.
Cramer C, Kutzner K, Fuhrmann T. Bootstrapping locality-aware P2P networks. In: Proceedings of the 12th IEEE international conference on networks, 2004.
Eger K, Killat U. Bandwidth trading in BitTorrent-like P2P networks for content distribution. Computer Communications 2008;31(February):201–11.
Fan L, Cao P, Almeida J, Broder AZ. Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Transactions on Networking 2000;8(June):281–93.
Feldman M, Chuang J. Overcoming free-riding behavior in peer-to-peer systems. ACM SIGecom Exchanges 2005;5(July):41–50.
Frakes W, Baeza-Yates R. Information retrieval: data structures and algorithms. NJ, USA: Prentice Hall; 1992.
Golle P, Leyton-Brown K, Mironov I, Lillibridge M. Incentives for sharing in peer-to-peer networks. In: Proceedings of the 2nd international workshop on electronic commerce. Lecture notes in computer science, 2001. p. 75–87.
Han J, Watson D, Jahanian F. Topology aware overlay networks. In: Proceedings of IEEE INFOCOM'05, 2005. p. 2554–65.
Hsiao HC, Liao H, Huang CC. Resolving the topology mismatch problem in unstructured peer-to-peer networks. IEEE Transactions on Parallel and Distributed Systems 2009;20:1668–81.
Hu CL, Chen DY, Chang YH, Chen YW. Fair peer assignment scheme for peer-to-peer file sharing. KSII Transactions on Internet and Information Systems 2010;4:709–36.
Huang CM, Hsu TH, Hsu MF. Network-aware P2P file sharing over the wireless mobile networks. IEEE Journal on Selected Areas in Communications 2007;25(January):72–93.
Hughes D, Coulson G, Walkerdine J. Free riding on gnutella revisited: the bell tolls? IEEE Distributed Systems Online 2005;6(June).
ipoque GmbH. Internet study 2008/2009. 〈http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009〉, 2009.
Jin X, Chan SHG. Reputation estimation and query in peer-to-peer networks. IEEE Communications Magazine 2010;48(April):122–7.
Joung YJ, Lin ZW. On the self-organization of a hybrid peer-to-peer system. Journal of Network and Computer Applications 2010;33:183–202.
Legout A, Urvoy-Keller G, Michiardi P. Rarest first and choke algorithms are enough. In: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, 2006.
Li M, Lee WC, Sivasubramaniam A. Performance evaluation of neighborhood signature techniques for peer-to-peer search. Journal of Computers 2007;17(January):11–36.
Liao X, Jin H, Liu Y, Ni LM, Deng D. Anysee: peer-to-peer live streaming. In: Proceedings of IEEE INFOCOM'06, 2006. p. 1–10.
Liu J, Zhang X, Li B, Zhang Q, Zhu W. Distributed distance measurement for large-scale networks. Computer Networks 2003;41:177–92.
Liu Y, Guo Y, Liang C. A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications 2008;1(March):18–28.
Liu Y, Xiao L, Liu X, Ni X. Location awareness in unstructured peer-to-peer systems. IEEE Transactions on Parallel and Distributed Systems 2005;16(February):163–74.
Lua EK, Crowcroft J, Pias M, Sharma R, Lim S. A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications on Tutorials and Surveys 2005;7(July):72–93.
Lua EK, Zhou X. Network-aware superpeers-peers geometric overlay network. In: Proceedings of 16th international conference on computer communications and networks, 2007. p. 141–8.
Lv Q, Cao P, Cohen E, Li K, Shenker S. Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th ACM international conference on supercomputing, 2002. p. 84–95.
Marti S, Garcia-Molina H. Taxonomy of trust: categorizing peer-to-peer reputation systems. Computer Networks 2006;50(March):472–84.
Nayebi A, Rahimi MR, Azad HS. Analysis of time-based random waypoint mobility model for wireless mobile networks. In: Proceedings of the 4th international conference on information technology: new generations, 2007. p. 42–7.
Nghiem TP, Waluyo AB. A pure P2P paradigm for query processing in mobile ad-hoc networks. In: Proceedings of the 9th international conference on advances in mobile computing and multimedia, 2011. p. 182–9.
Papadakis H, Fragopoulou P, Markatos E, Roussopoulos M. Ita: innocuous topology awareness for unstructured P2P networks. IEEE Transactions on Parallel and Distributed Systems 2012, http://dx.doi.org/10.1109/TPDS.2012.137.
Qiu T, Chen G, Ye M, Chan E, Zhao BY. Towards location-aware topology in both unstructured and structured p2p systems. In: Proceedings of international conference on parallel processing, 2007. p. 133–44.
Resnick P, Kuwabara K, Zeckhauser R, Friedman E. Reputation systems. Communications of the ACM 2000;43(12 (December)):45–8.
Satsiou A, Tassiulas L. Reputation-based resource allocation in P2P systems of rational users. IEEE Transactions on Parallel and Distributed Systems 2010;21:466–79.
Sharma P, Xu Z, Banerjee S, Lee SJ. Estimating network proximity and latency. ACM SIGCOMM Computer Communication Review 2006;36(July):39–50.
Stutzbach D, Rejaie R. Understanding churn in peer-to-peer networks. In: Proceedings of the 6th ACM SIGCOMM conference on internet measurement, 2006. p. 189–202.
Wang F, Xiong Y, Liu J. mtreebone: a hybrid tree/mesh overlay for application-layer live video multicast. In: Proceedings of the 27th IEEE international conference on distributed computing systems, 2007.
Wua CH, Chianga K, Yua RJ, Wang SD. Locality and resource aware peer-to-peer overlay networks. Journal of the Chinese Institute of Engineers 2008;31(October):1207–17.
Yu H, Shen Z, Miao C, Leung C, Niyato D. A survey of trust and reputation management systems in wireless communications. Proceedings of the IEEE 2010;98(October):1755–7172.
Yu H, Zheng D, Zhao BY, Zheng W. Understanding user behavior in large-scale video-on-demand systems. ACM SIGOPS Operating Systems Review 2006;40(October):333–44.
Zipf GK. Human behaviour and the principle of least effort.Reading, MA: Addison-Wesley; 1949.