

Self-Organization in Peer-to-Peer Systems*

CHING-WEI HUANG and WUU YANG
National Chiao-Tung University
HsinChu, Taiwan, R.O.C.

Abstract

Peer-to-peer systems now become one of the most popular Internet applications. However, these systems are consuming a major part of the Internet bandwidth. Additionally, many P2P systems suffer from the inefficient and unstable performance of searching. These two problems are highly related to the topological structure of a P2P system. Due to the lack of central servers, P2P systems need self-organization mechanisms to maintain an appropriate topological structure and to guarantee the correctness and performance of message communication. We propose a self-organization mechanism that addresses four aspects: node identification, network topology control, message routing and broadcasting, and network connectedness. The performance of the self-organization mechanism is also discussed.

Keywords: Peer-to-Peer System, Self-Organization, Gnutella, Small World, Distributed System.

1 Introduction

Peer-to-peer(P2P) systems are one of the most popular Internet inventions. The P2P model provides a solution to one of the major problems of the traditional client-server model: the bottleneck of services on servers. Although the P2P model provides a new solution, it also brings up new challenges. Without the help of servers, all communication between nodes in a P2P system is completely dependent on the message exchange among nodes. Moreover, the control and maintenance of the whole P2P system is distributed among all the nodes. All nodes in a P2P system must co-operate a *self-organization* mechanism.

Traditional network services, such as FTP, HTTP, DNS, etc., now have a chance to be implemented in the P2P model. Underlying these high-level applications, the self-organization mechanism provides an infrastructure for distributed systems. A well-designed self-organization mechanism provides good performance and reduces the cost of services. We study four aspects of self-organization: node identification, network topology control, message routing and broadcasting, and connectedness of the P2P system. We will discuss the four aspects in the remainder of this section.

Node Identification In the traditional client-server model, the servers can identify their clients by their IP addresses. A more advanced model is to provide a registration mechanism and to generate identifications for registered clients. On the other hand, because in the P2P model, there is no server in charge of identification or registration, the identifications of nodes must be generated in a distributed way. Two related problems are (1) how to generate a unique identification for each node and (2) how to connect nodes after they obtain their identifications. An advanced issue is the anonymity. An anonymous P2P system protects users' privacy. However, communication between nodes becomes more difficult in an anonymous P2P system because it is difficult to locate the source and the destination of messages. The anonymity problem is addressed in a separate paper [20].

*This work was supported, in part, by National Science Council, Taiwan, R.O.C., under grant NSC 91-2213-E-009-068

Network Topology Control The network topology significantly affects the performance of a P2P system. One obvious example is the Gnutella network, one of the most famous P2P systems. Recent research [1] shows that the message broadcasting protocol of the Gnutella generates a huge number of redundant broadcast messages. The number of broadcast messages originating from one request in the Gnutella network is estimated as $2 * \sum_{i=0}^{TTL} C * (C - 1)^i$, where C is the average number of neighbors per node and TTL is the average number of hops per message. Another research [5] shows that 95% of all the pairs of nodes could exchange messages in less than 7 hops. Hence, it is reasonable to limit TTL to 7. If the average number of neighbors per node is 4, the estimated number of broadcast messages originating from one request is 26,240 in Gnutella. However, if the average number of neighbors per node is 8, the estimated number of broadcast messages becomes 15,372,800.

A plausible cause of this problem is that Gnutella does not control network topology. Random connections will easily generate a huge number of redundant broadcast messages. The Internet is a collection of autonomous systems connected by routers. An Autonomous System (AS) [5][13] is a set of routers under a single technical administration that uses an interior gateway protocol and common metrics to route packets within the AS and an exterior gateway protocol to route packets to other AS's. If the messages travel across different autonomous systems frequently, the communication will be prohibitively expensive. The research [1] shows that only 2 to 5 percent of connections in Gnutella are within a single AS.

An appropriate network topology, such as a tree, could provide efficient message delivery and hence reduce the number of redundant broadcast messages. However, additional messages are required to maintain the appropriate topology of the network. In order to achieve better performance, a good P2P system should keep physically close nodes logically close. In other words, nodes within the same AS should be kept in the same cluster in a P2P system.

Message Routing and Broadcasting Message delivery plays a key role in a P2P system. In order to analyze the performance of message delivery in P2P systems, three measurements are commonly used: the average number of hops[2][3], the clustering coefficient of the network topology[2][3], and the number of broadcast messages originating from one request. The average number of hops per request is the average path length of message propagation of a request. The fewer the average number of hops the lower the communication cost. The clustering coefficient¹ of a graph is the degree of the clustering effect of the graph. The higher the clustering coefficient of a graph the more nodes are connected in a regular fashion[3]. The lower the clustering coefficient of a graph the more nodes are connected randomly. Recent research [3] shows that a small-world network could have the benefits of both the higher clustering coefficient and the lower average number of hops.

In a P2P system, a request is delivered through successive message broadcasts. Redundant broadcast messages are inevitable. However, without control, the redundant broadcast messages may consume the major part of the network bandwidth. There are several message broadcasting algorithms proposed in the literature, including Freenet[8], Tapestry[9], Pastry[10], Chord[11], etc. Each addresses a different problem. In this paper, we propose a new message broadcasting algorithm that tries to reduce the number of redundant broadcast messages originating from one request.

Connectedness of the P2P system The last issue we are concerned with is the connectedness of the P2P system. The frequent connecting and disconnecting operations of nodes may possibly split the P2P system. In a disconnected system, the request messages could not reach every node. This leads to the poor search accuracy. An ideal but infeasible solution is to keep all the nodes always connected to the system. A more practical solution is to tolerate a disconnected system and develop a recovery mechanism to re-connect the disconnected system.

¹The clustering coefficient [2] of a graph is the average of the clustering coefficients of all nodes. The clustering coefficient of a node N in a graph is the ratio of the number of edges among N's neighbors to $k(k-1)/2$, where k is the number of N's neighbors.

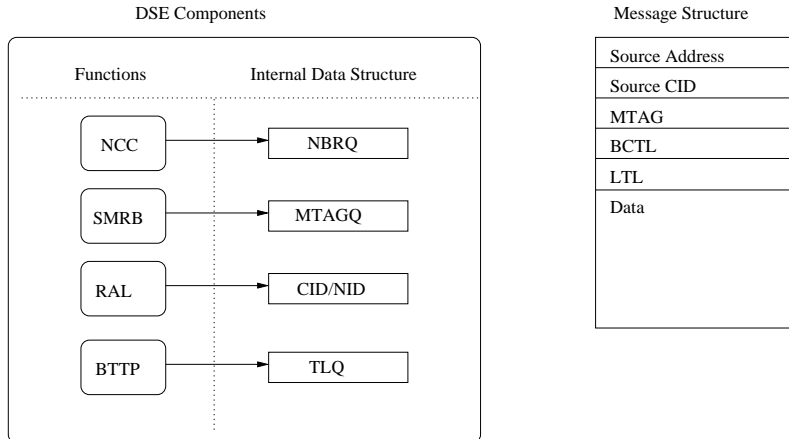


Figure 1: The DSE architecture and message structure

In this paper, we present DSE (Distributed Search Environment), our P2P system. DSE concentrates on the above four aspects and tries to bring up new solutions to the respective problems. The remainder of this paper is organized as follows. Section 2 illustrates the architecture and design of our DSE system. Connection structure of the system and maintaining system connectedness are two major issues. Section 3 introduces the implementation of DSE. Section 4 analyzes the performance of a simulated DSE system. Section 5 is the conclusion.

2 System Architecture and Design

This section describes the DSE architecture, which is shown in Figure 1. Self-organization in DSE is implemented in four parts: Node registration and initial connection, the neighbor clustering control (NCC) mechanism, smart message routing and broadcasting (SMRB), and the ring-around-leader (RAL) algorithm.

When a user runs the DSE software for the first time, the node registration process is activated, which assigns the user a unique ID. After registration completes, the node attempts to connect to DSE through some default nodes (currently there are ten default nodes in DSE). If the node connects to any of the default nodes successfully, the node becomes part of the DSE system. After the initial connections are established, the NCC, SMRB and RAL mechanisms are activated.

A standard time is very important in a fully distributed system. The DSE system assigns a globally standard time to all nodes with the Network Time Protocol (NTP) [12]. All nodes periodically query one of the NTP servers to obtain this globally standard time.

The NCC mechanism controls the number of neighbors of a node based on their physical distances. As a result, physically close nodes will be placed in the same cluster. On the other hand, SMRB focuses on the performance and efficiency of message broadcasting and routing. Every broadcast message contains a small area, called the *broadcast travelling list* (BCTL), which records the visited nodes. Every broadcast message also has a unique tag, denoted by MTAG, which is generated by the node that initiates the request. SMRB makes use of BCTL, MTAG and the identification of a node to reduce redundant broadcast messages.

The RAL mechanism tries to use the information exchange between neighbors to keep the whole system connected. The RAL mechanism elects the node that has the maximal identification as the leader of the component. The leader's identification servers as the component ID (that is, the identification of the connected component in the P2P system). The component ID is timestamped before it is transmitted over the network. By checking the timestamp of the component ID, nodes could detect the disconnection of the system.

2.1 Node Identification

The DSE system adopts the distributed hash table (DHT) [4] to generate node identifications from two pieces of information: (1) IP address of the host and the standard timestamp when the new user registers. (2) user behavior related information. DSE asks the user who wants to join the DSE system to type some arbitrary sentences. The characteristics of typing these sentences is fed into a hash function to generate an identification for the user. Since some behaviours of human beings are unique, the generated hash value could be unique to serve as the identification of the user(node). The behavior-related information includes the typing speed, the trace of writing, the keyboard pressure from the fingers, etc.

There are two possible hash functions that we could use: MD5 [14] and SHA1 [15]. MD5 is a fast function; however, recent research [16] shows that MD5 could be cracked with brute-force algorithms. It is also possible that different samples could be mapped to the same MD5 value. SHA1, therefore, is a better choice because it provides a longer hash value (so that collision is rarer) and is very difficult to crack.

Since there is no server to provide a list of initial neighbors for a new node, the DSE system chooses some volunteer nodes to serve as the initial neighbors for every new node. These volunteers perform exactly the same function as other nodes except that all new nodes connect to them initially. Once a new node attempt to connects to any of these volunteers, it joins the DSE system successfully.

Although the node identification is generated by the DSE software locally, the hashing property of SHA1 guarantees the global uniqueness of the identification as long as the input message of SHA1 is distinct. Also, the identifications are strings that can be compared to one another. The unique identification gives us great support in reducing redundant broadcast messages.

2.2 Node Clustering

DSE divides nodes into clusters. Every node maintains a list of its current neighbors. Every message in the DSE system contains a header, which records the source (i.e., the identification and the IP address of the source node), shown in Figure 1. Every node examines the headers of the incoming messages. If the source of the message is not a current neighbor of the node, then the node adds the source of the incoming message to its list of current neighbors. The node will also attempts to establish a connection to the source of the message. An important feature of the self-organization mechanism of the DSE system is the *Neighbor Clustering Control* (NCC) mechanism. NCC tries to group nodes based on their physical distances. For a given node, those connected neighbors can be classified into two kinds: a very large percentage (e.g., 98%) are the physically close nodes, and a very small percentage (e.g., 2%) are arbitrarily chosen from the remaining nodes. By the small-world effect [3], the DSE system performs well on keeping the characteristic path length[3] of the request low and the clustering coefficient high. On the other hand, the Internet is a collection of interconnected autonomous systems(ASs). The cost of message delivery across different ASs is more expensive than within the same AS. The NCC mechanism could reduce this cost significantly because connections are built within the same ASs whenever possible. Two questions that we may ask are (1) the physical distance between two nodes and (2) a method to measure the distance.

Our approach is to use the ICMP protocol [7]. The ICMP protocol estimates the physical distance between two nodes with two pieces of information: the number of routers and the packet transport time on the connection. Although the measurements obtained through the ICMP protocol may not be the exact cost of the message transport between two nodes, it is a meaningful approximation to the physical distance of two nodes.

The NCC mechanism also controls the number of neighbors for each node, There is a lower bound (called NBLB) and an upper bound (call NBUB) on the number of neighbors per node. Once the number of neighbors exceeds NBUB, the neighbors are re-clustered by NCC to reduce the number of neighbors per node below NBUB. On the other hand, it is not necessary to worry about the case that the number of neighbors falls below NBLB because the incoming messages

```

function insertNBR(node N) {
    insert(NBRQ, N);          // insert the new node N into the set NBRQ
};

function NCC() {
    if (sizeof(NBRQ)<NBUB) return;
    Q=sort_by_distance(NBRQ); // sort the NBRQ by their distances (from local
node)
    R=0.98;                    // 98% are closer nodes
    S=get_closer_nodes(Q, NBLB*R); // get closer nodes from Q (e.g. 98%)
    Q=Q-S;
    V=get_random_nodes(Q, NBLB*(1-R)); // get nodes from Q randomly (e.g. 2%)
    NBRQ=S ∪ V;
};

```

Figure 2: The NCC mechanism

bring the node chances of adding new neighbors.

The algorithm of the NCC mechanism is summarized in Figure 2. Let NBRQ be the set of neighbors of a node. When a message arrives at a node, the function insertNBR() inserts the source address of the message into the current NBRQ. The function NCC executes periodically to reduce the size of current NBRQ.

2.3 Smart Message Routing and Broadcasting

Broadcasting is the most widely used and the most powerful communication method in a P2P system. However, incautious design could result in a huge number of redundant broadcast messages. The SMRB mechanism provides two functions to solve this redundancy problem: avoiding duplicated broadcast messages, and preventing redundant message delivery. The difference between these two functions is that the prevention of redundant delivery works before a message is broadcast out (in this stage, we do not know whether the message is redundant or not), while the avoidance of duplicated broadcasting works after a message arrives at a node (in this stage, we already know it is a redundant message).

Avoid Duplicated Broadcasting DSE associates every broadcast message with a unique tag, called the *MTAG*, which is generated by the node that initiates the request. Every node has a queue for storing the MTAGs of the arriving broadcast messages. Since the topology of a P2P network may contain loops, duplicated broadcast messages are inevitable. The SMRB mechanism will check the queue to see if a message with the same MTAG has arrived before. If so, the duplicated messages will not be further broadcast to the neighbors.

Prevent Redundant Delivery The second part of the SMRB mechanism focuses on preventing redundant message delivery. Every broadcast message reserves a small area, called the *broadcast travel list (BCTL)*, which records the visited nodes and the nodes where the message will be sent to next. The function of BCTL is to prevent a message from being sent to the same node more than once. The performance of BCTL depends on the clustering coefficient of the system. A system with a higher clustering coefficient would benefit more from the BCTL mechanism because the more nodes are in the same cluster, the more redundant messages could be filtered out. Obviously,

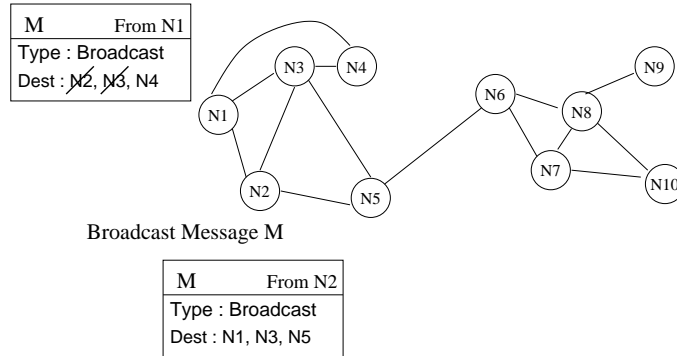


Figure 3: Broadcast in intra-cluster and inter-cluster

the capacity of BCTL also determines the number of redundant messages that can be filtered out. We set the capacity of BCTL as a multiple of NBUB because every node maintains at most NBUB neighbors. We say that BCTL is of multiple K if the capacity of the BCTL is K times NBUB. BCTL of multiple 0 (i.e., no BCTL at all) means that no broadcast messages will be filtered out. For easier explanation in later discussion, the notation $BCTL(M, N, 0)$ means the contents of BCTL before message M is sent to node N , and $BCTL(M, N, 1)$ means the contents of BCTL after message M is sent out from node N .

There are two stages in preventing redundant broadcast messages. For a given node, the first stage focuses on the nodes to which the message will be sent out from the local node. Before the message is sent by the local node, the current neighbors of the local node will be inserted into the BCTL of the message. In the mean time, any neighbor which is already in the BCTL of the message will be filtered out because the message has already been sent there.

The second stage focuses on the nodes from which the message is sent to the local node. When a node prepares to forward a message, if it knows that one of its neighbors will also send the same message to the same destination, there should be only one node that will actually send the message. The choice depends on the comparison of their identifications. When two or more nodes will send the same message to the same destination, the one with the largest identification will actually send the message. All other nodes should not send the message to that destination. A prerequisite of the filtering on stage 2 is that all node must know all of the neighbors of its neighbors. Figure 5 will explains this prerequisite later.

In the example in Figure 3, there are two clusters $\{N1, N2, N3, N4, N5\}$ and $\{N6, N7, N8, N9, N10\}$. Node $N2$ begins to broadcast a new message M . The message M is first sent to $N2$'s neighbors $N1, N3$, and $N5$. Hence, $BCTL(M, N2, 1) = \{N1, N2, N3, N5\}$. After $N1$ receives M , $N1$ will not forward M to $N3$ because $N3$ is in $BCTL(M, N1, 0)$.

The capacity of BCTL determines the filtering effect of redundant messages. However it does not need to be very large. The NCC mechanism clusters all nodes ideally so that the number of neighbors per node is between NBLB and NBUB. BCTL of multiple 1 should suffice to filter out most redundant messages. Furthermore, in addition to the closer nodes, the NCC mechanism keeps a small percentage of neighbors that are randomly chosen in order to create the small-world effect [3]. Moreover, the DSE system is not always ideally clustered because of the dynamic addition of new neighbors to nodes. According to our simulation results, BCTL of multiple 1 reduces approximately $2/3$ of the redundant broadcast messages. However, BCTL of multiple $(K+1)$ reduces less additional broadcast messages than the BCTL of multiple K when $K > 1$. The best choice based on our experiment is BCTL of multiple 2.

In the example of Figure 4, suppose NBUB is 4 and we use BCTL of multiple 1. Node $N1$ first broadcasts message M to it neighbors $N2, N3$ and $N4$. Also, we assume that the message travels on paths $N1-N3-N6$ and $N1-N2-N6$ faster than on path $N1-N4-N6$. By the filtering on stage 1, the messages travelling on paths $N2 \rightarrow N3$ and $N3 \rightarrow N2$ are filtered out by nodes $N2$ and $N3$, respectively. Message M is then forwarded to node $N6$ from nodes $N2$ and $N3$. Since $BCTL(M,$

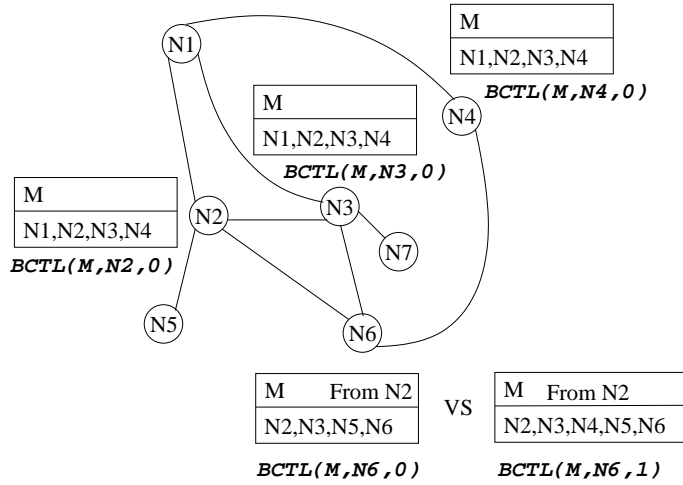


Figure 4: Filtering on stage 1

$BCTL(M, N2, 1)$ and $BCTL(M, N3, 1)$ are $\{N2, N3, N5, N6\}$ and $\{N2, N3, N6, N7\}$, respectively, neither contains $N4$. Therefore, before message M is sent to $N6$ from $N4$, node $N6$ may have already sent M to $N4$. The redundant message transport from $N4$ to $N6$ will occur. If we use $BCTL$ of multiple 2, this situation can be avoided because, in this case, both $BCTL(M, N2, 1)$ and $BCTL(M, N3, 1)$ contain node $N4$.

In addition to the filtering on stage 1, which focuses on the nodes to which the message is broadcast, the filtering on stage 2 focuses on the nodes from which the arrived message is sent. There are two requirements for fulfilling the filtering on stage 2: First, every node owns a unique identification that can be compared. Second, every node knows the neighbors of its neighbors. Let $NBRQ(N)$ be the set of neighbors of node N , $TS(M, N)$ be $NBRQ(N) - BCTL(M, N, 0)$ (here - means set difference) and $NS(M, N)$ be $NBRQ(N) \cap BCTL(M, N, 0)$. $TS(M, N)$ represents the remaining nodes to which the message M will be broadcast from node N after stage 1. $NS(M, N)$ represents the neighbors of node N that are already in $BCTL(M, N, 0)$. The idea of the filtering on stage 2 is that for every node Y in $TS(M, N)$, if there exists a node X in $NS(M, N)$ such that Y is a neighbor of X and the identification of X is greater than that of N , then node N will not forward message M to node Y .

For the example in Figure 5, the numbers inside the angle brackets, e.g. $\langle 461 \rangle$, are the identifications of the nodes. When message M is initially broadcast from node $N1$ and then arrives at nodes $N2, N3, N4$ and $N5$, the broadcast messages along paths $N2-N3$ and $N3-N4$ will be filtered out during stage 1. Now consider the broadcast messages along the connections $N2-N6$, $N3-N6$ and $N4-N6$. Note that $BCTL(M, N2, 0) = BCTL(M, N3, 0) = BCTL(M, N4, 0) = \{N1, N2, N3, N4, N5\}$. The message M could be redundantly sent to $N6$ by nodes $N2, N3$ and $N4$. At this moment, $NBRQ(N3) = \{N1, N2, N4, N6\}$, $TS(M, N3) = NBRQ(N3) - BCTL(M, N3, 0) = \{N1, N2, N4, N6\} - \{N1, N2, N3, N4, N5\} = \{N6\}$, and $NS(M, N3) = \{N1, N2, N4, N6\} \cap \{N1, N2, N3, N4, N5\} = \{N2, N4\}$. Without the filtering on stage 2, both nodes $N2$ and $N3$ will forward message M to node $N6$. However, with the filtering on stage 2, both nodes $N2$ and $N3$ discover two facts: first, the identification of $N2$ (which is 2192), is greater than that of $N3$ (which is 47), and second, $NBRQ(N2)$ and $NBRQ(N3)$ both contain the destination node $N6$. Therefore, node $N3$ will not forward M to $N6$. As a result, message M is sent to node $N6$ from only node $N2$. The algorithm of the SMRB mechanism is summarized in Figure 6.

2.4 Network Connectedness

The performance of searching on a P2P system is heavily dependent on the topology of the system. If the system is split into n components of equal size, the search hit ratio drops to $1/n$

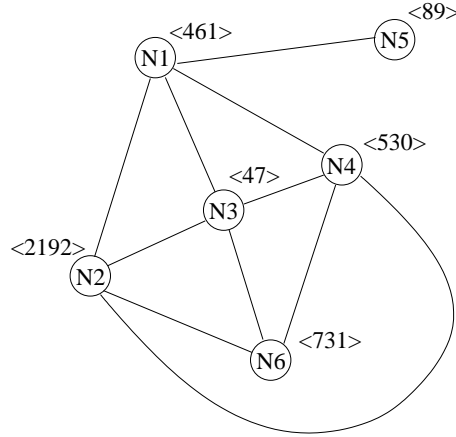


Figure 5: Filtering on stage 2

```

function broadcast(message M) {
  // L is the local node

  TS:=NBRQ - M.BCTL;
  NS:=NBRQ ∩ M.BCTL;
  SQ={};

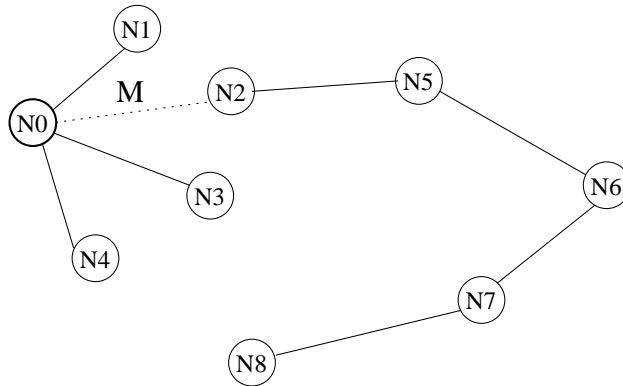
  for (node X in TS) {
    if (there is a node G in NS such that G.IDENTIFIER>L.IDENTIFIER and X is a neighbor
of G) continue;
    SQ:=SQ ∪ {X};
  };

  M.BCTL:=M.BCTL ∪ SQ;
  reduceBCTLsize(M.BCTL, 2*NBUB); // reduce the size of M.BCTL to 2*NBUB

  for (node Y in SQ) {
    sendto(Y, M); // send message M to node Y
  };
};

```

Figure 6: Redundant delivery prevention in the SMRB mechanism (assume BCTL of multiple 2)



M : Last Message send from node N2

The Travelling Path of M : (N8,N5,N2)

The Neighbors of Node N0 : {N1,N2,N3,N4}

Figure 7: Back track the travelling path

because a search request can reach to the nodes in the same component. When a P2P system splits, reconnecting the components is essential. DSE implements two distributed algorithms for reconnecting the components. One algorithm tracks back the travelling paths of messages, and the other reconnects the broken connections between components via information exchange of nodes.

Back Tracking the Travelling Paths As a broadcast message may travel through hundreds of nodes, the visited nodes themselves become good clues for re-connecting the broken network. Every broadcast message in DSE contains a small record, called the *travelling list* (TL), which is a list of *clues*. A clue consists of three node addresses: the source node (S) where the message was initiated, the last node (L) before the message arrives at the current node, and the previous node (P) before node L. A clue (S, P, L) is used for the back-tracking on the fail nodes whose connections are lost.

Every node in DSE system analyzes all broadcast messages and stores their TLs in a queue (which is called the TLQ). If a node detects one of its neighbors lost connection, it will search the lost node in TLQ to find any matched clues. If a clue (S, P, L) is found such that the lost neighbor is L, then the local node tries to connect to nodes S and P. This recovery mechanism, called BTTP (back-tracking the travelling path), reconnects the split system.

In Figure 7, suppose node N0 detects that its neighbor N2 lost the connection. After checking the TLQ, the clue (N8, N5, N2) is found. The node N0 will try to re-build the connections to nodes N8, N6 and N5 and the disconnected network becomes connected again if any of the connections succeeds.

The Ring-Around-Leader Algorithm Even with the help of back tracking on the travelling paths, the recovery of a disconnected network may still fail. We propose another efficient algorithm, called the Ring-Around-Leader (RAL) algorithm, to maintain the connection of the DSE system.

Every node in the DSE system maintains two important pieces of information. The first is the Node Identification(NID). The second is the Component Identification(CID), which is the largest NID that the node knows in the same component. Ideally, every node shares the same CID because the whole DSE system forms a single connected component. The CID is updated by the request message broadcast and periodical information exchange between each pair of connected nodes.

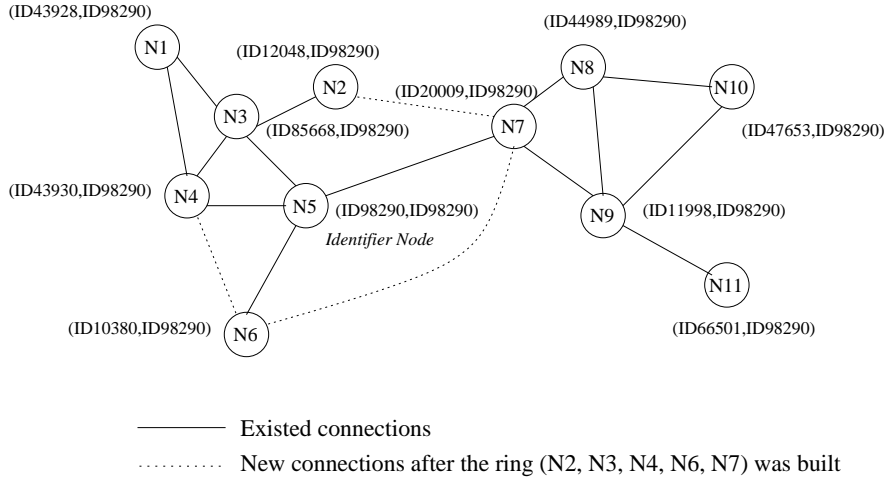


Figure 8: CID, NID and the leader

Every node in the DSE system broadcasts its current CID timestamped with latest standard time to its neighbors. When a node X tries to join the DSE system, the initial value of its CID, denoted as $CID(X)$, is set to its own NID, denoted as $NID(X)$. For each pair of connected nodes A and B , $CID(A)$ is compared to $NID(B)$ periodically. If $CID(A)$ is greater than $NID(B)$, $CID(B)$ will be set to $CID(A)$. Similarly, $CID(B)$ is compared to $NID(A)$. Suppose that the DSE system is connected, the CIDs of all nodes will be the largest NID of all nodes eventually. The node that owns the largest NID in the component is called the *leader* of the component.

The timestamp of the CID is used to check whether the CID is out of date. The leader of the component periodically broadcasts its timestamped CID (which is also its NID) to all nodes. Every node will obtain the CID with the latest timestamp periodically. The time of spreading the largest CID to every node depends on the size of the network. We can estimate an appropriate upper bound of the time to be the time-limit of the timestamp of the CID. Every node checks the timestamp of its CID. If its CID is out of date, the new CID (i.e., its NID) replaces the old.

Since the CID exchange only occurs on the neighbors, the bandwidth cost is relatively less than that by broadcast. However, if a user initialize and broadcast a request, the CID of the user is attached on that request. When the broadcast message arrives at a node, the comparison of the attached CID and the local NID occur. Such design speed up the CID exchange.

On the other hand, every node periodically checks whether its CID is equal to its NID. If so, the node is the leader of the component. Once a node assumes the leadership of the component, it immediately informs all of its neighbors to build a ring connection around the leader. This ring around the leader assures the connectedness of the component when the leader fails. We explain this in the following example.

For the example in Figure 8, each node is annotated with its (NID, CID) pair. As we can see, all nodes form a single connected component $\{N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, N11\}$. Node $N8$ has NID "ID44989" and CID "ID98290". Node $N5$ has the same CID and NID value: "ID98290", which means that $N5$ is the leader of the component. Once node $N5$ assumes the leadership of the component, it informs all of its neighbors to form a ring connection ($N2, N3, N4, N6, N7$).

Some events of the nodes, such as logout, network failure, and being attacked, etc., may break down the connection. We call the events that break down the connections as the failure of a node. Any disconnection may split the P2P system. The RAL algorithm tries to reconnect the components upon detecting node failures. There are two possible cases of a failed node:

case 1 : The failed node is not the leader of the component.

In this case, the neighbors of the failed node will detect the failure and try to connect to the leader. If the failure splits the system, the connection to the leader will help the components to join again.

In the example of Figure 8, if node N7 fails, the whole system will be split into two components {N1, N2, N3, N4, N5, N6} and {N8, N9, N10, N11}. The neighbors of node N7 including nodes N2, N5, N6, N8 and N9 will detect the failure of N7 and try to re-connect to the leader N5. Once the connection (N5, N8) or (N5, N9) have been built successfully, the two components join again. Thereafter, CIDs of all nodes will be updated (in this case, CID is still ID98290).

Nodes in a P2P system may fail frequently. If every node failure causes a connection to the leader of the component, the large number of connections to the leader will make it a bottleneck of the whole system. An efficient method is to check the expiring of the local CID instead of the failure of neighbors. If a disconnection of a node separates the system into two components, a node in one of the components will eventually detect that its CID is out of date. The node who first found this condition tries to connect to the leader of the CID.

In the example of Figure 8, if node N7 fails, no node of the component {N8, N9, N10, N11} will receive the new CID from the leader N5 of the original component. The CIDs of nodes N8, N9, N10 and N11 will expire eventually. Any one of them, for example, node N9, who first discovers its CID expires will attempt to contact the leader N5. Node N9 first queries its neighbors N8, N10 and N11 whether any of them has a connection to the leader N5. If so, node N9 drops the connection attempt. Otherwise, N9 builds a connection to the leader N5. This indirect connection prevents the leader from becoming a bottleneck. In the example in Figure 8, the first of nodes N8, N9, N10 and N11 who finds its CID expires will build a connection to the leader N5.

case 2: The failed node is the leader of the component.

The failure of the leader of a component will never split the component because the leader's neighbors have formed a ring connection. This guarantees the component remains connected even if the leader fails. In the example in Figure 9, if the leader N5 fails, the ring around N5 (N2, N3, N4, N6, N7) will prevent the component from being split. After node N5 fails, nodes in the ring around N5 will find its CID "ID98290" expires eventually and replace its CID with its own NID. After several message exchanges, a new leader N3 will be elected with the new CID "ID85668".

The RAL algorithm can recover from a single-node failure. However, it can not handle massive and simultaneous failures which include the leader and other nodes in the ring. In the example of Figure 9, if nodes {N2, N3, N4, N5, N6, N7} all fail simultaneously, the DSE system will still be split into two components. The simultaneous failures of nodes are not considered in the RAL algorithm.

3 Implementation of DSE

We have implemented DSE in PHP [17] on multiple platforms to validate our approach. The code name of our DSE project is *apia* (Advanced P2P infrastructure and architecture) [18]. Apia supports Windows, Linux, FreeBSD, Solaris and Mac OS X. Apia is a multi-purpose framework for applications. Users could develop their own applications with the functions apia provides. Currently a file-sharing service is the first experimental application on apia. Other applications under development include a P2P web service, a DNS service, and a grid computing service.

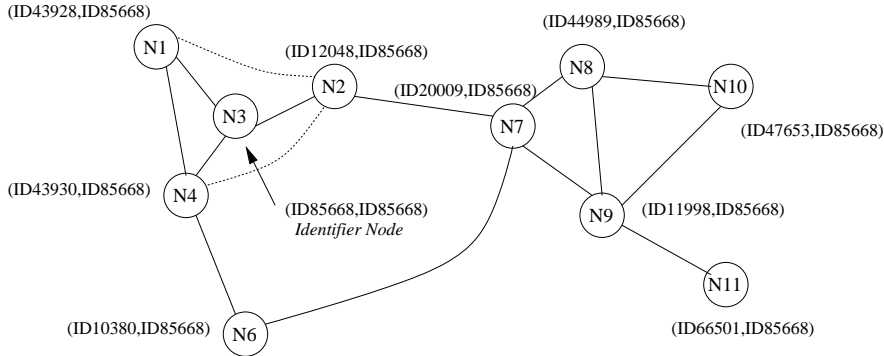


Figure 9: Generation of a new leader

The core component of *apia* is a self-organization system which performs the NCC, SMRB, and RAL mechanisms. Another component of *apia* is a set of API functions which provide basic control for self-organization. The set of API functions can be extended to provide additional functions for different applications.

4 Performance

In order to determine the performance of the DSE system, we build a network environment to simulate the behavior of nodes. The number of default nodes which serves as the initial neighbors for the new nodes is 5. All nodes broadcast messages periodically between 20 to 60 seconds. All nodes run the NCC, SMRB and RAL mechanisms continuously.

The following experiments try to analyze the NCC, SMRB and RAL mechanisms in 6 aspects: the relation between the average number of broadcast messages which originates from one request (AMOR) and the number of nodes in the DSE system, the relation between the average number of hops of request and the number of nodes, the relation between the clustering coefficient and the number of nodes, the relation between AMOR and BCTL of multiple K, and the relation between AMOR and the dynamic behaviors of nodes.

4.1 Observations of the NCC mechanism

In experiment 1, we generate 1000 nodes. One node is generated to connect to DSE every second. When the number of nodes is a multiple of 100, the system suspends generating new nodes for a period of time.

There are three measurements to be observed: the average number of broadcast messages which originates from one request (AMOR), the clustering coefficient, and the average number of hops that a message travels (HOPS). Figure 10 shows that AMOR is highly dependent on two factors: the dynamic behaviors of the system (i.e., the connections and failures of the nodes) and the number of nodes in the system. Consider the two periods of times, P1 and P2, in Figure 10. When the number of nodes increases from 200 to 300 in period P1, AMOR increases very fast. In period P2, the number of nodes remains 300 and AMOR drops to a stable lower bound quickly. In P1, the DSE system is not stable because nodes keep on joining DSE. The NCC mechanism continuously clusters the system in order to induce the small-world effect. But the dynamic behaviors of the system counteracts the NCC mechanism. Once no new node joins the system, the system enters a stable stage (as in period P2) and the small-world effect reduces AMOR.

Another important observation is that the lower bound of AMOR in the stable period is approximately $O(n)$, where n is the number of nodes. This property helps the DSE system scales up smoothly.

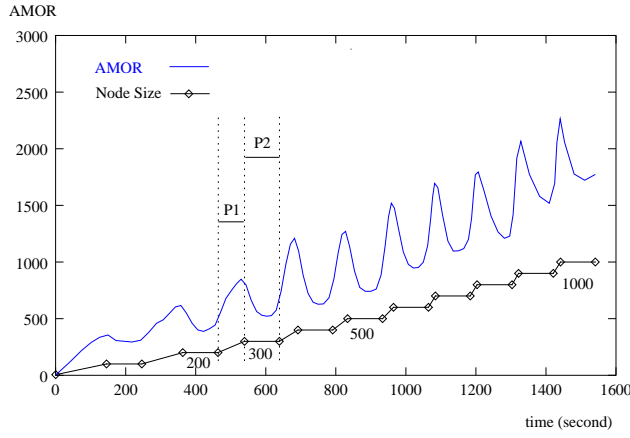


Figure 10: Relationship between AMOR and node size

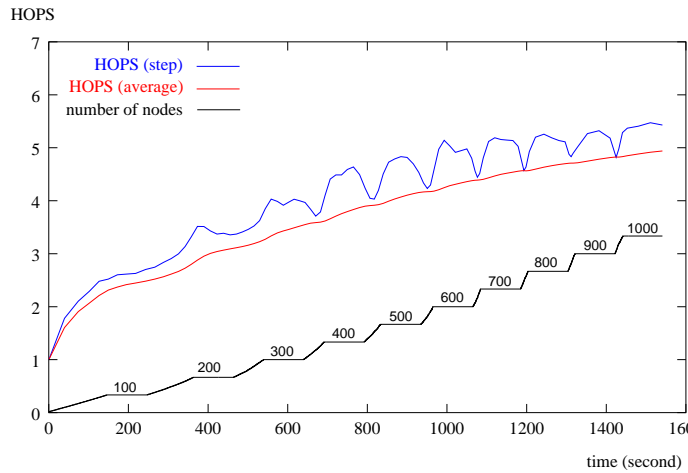


Figure 11: Relationship between HOPS and node size

The second measurement is the relation between the average number of hops and the number of nodes. New nodes are generated in the same way as in the experiment in Figure 10. Figure 11 shows that the average number of hops is less than $O(n)$, where n is the number of nodes. Just like AMOR, the number of hops is also dependent on the dynamic behaviors of the system. During the period when new nodes are generated, the number of hops grows quickly. In the period when the number of nodes remains fixed, the average number of hops drops and remains stable gradually.

The third measurement is the relation between the clustering coefficient and the number of nodes. Figure 12 shows that the clustering coefficient is also dependent on the dynamic behaviors of the system. During the period when new nodes are generated, the clustering coefficient drops quickly. In the period when the number of nodes remains unchanged, the clustering coefficient remains stable. In average, the clustering coefficient remains below a constant upper bound, which is about 0.7. This result shows that the NCC mechanism actually clusters nodes into groups. From the results in Figure 11 and Figure 12, we can see the NCC mechanism organizes all nodes into a system with the small-world effect.

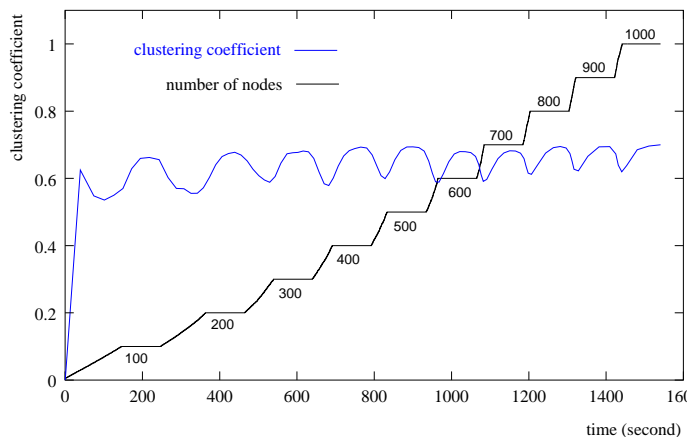


Figure 12: Relationship between clustering coefficient and node size

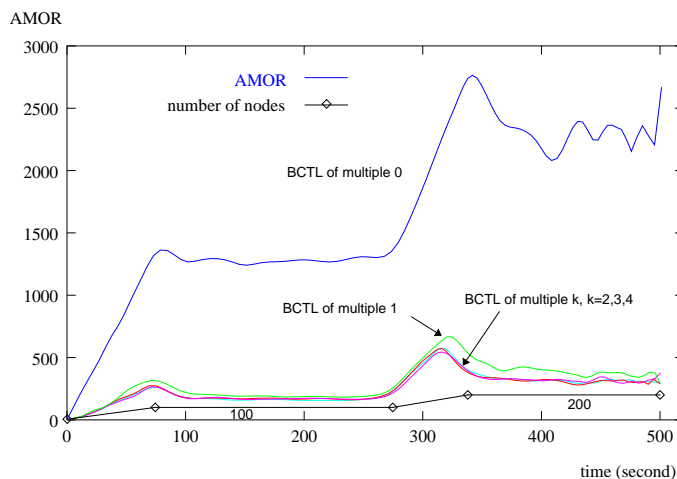


Figure 13: Relationship between AMOR and BCTL of multiple k

4.2 Observations of the SMRB mechanism

The SMRB mechanism provides a low-cost message routing and broadcasting method. We evaluate this cost by analyzing the relation between AMOR and BCTL of multiple K. The first part of the SMRB mechanism, the avoidance of duplicated broadcasting, is always enabled. The second part, the prevention of redundant delivery, has two filtering stages. In the experiment, we analyze how stages 1 and 2 can reduce the redundant broadcast messages.

Initially, we disable the second filtering stage and focus on the performance of the first filtering stage. We examine AMOR under five conditions: BCTL of multiple K, where $K=0, 1, 2, 3, 4$. Notice that the condition $K=0$ is equivalent to disabling the first filtering stage. Figure 13 shows that AMOR under BCTL of multiple 0 is 6 to 7 times the value under BCTL of multiple 1. AMOR under BCTL of multiple 2 is lower than that under BCTL of multiple 1, but AMORs under BCTL of multiple K remain almost the same value for $K=2, 3, 4$.

In the second part of this experiment, we focus on the performance of the second filtering stage. We compare AMOR under two conditions: BCTL of multiple 2 with and without the second filtering stage. Figure 14 shows that BCTL of multiple 2 with the second filtering stage could reduce AMOR approximately to a half of AMOR under BCTL of multiple 2 without the filtering of stage 2.

The dynamic variation of the system also affects AMOR. We start the experiment by generating

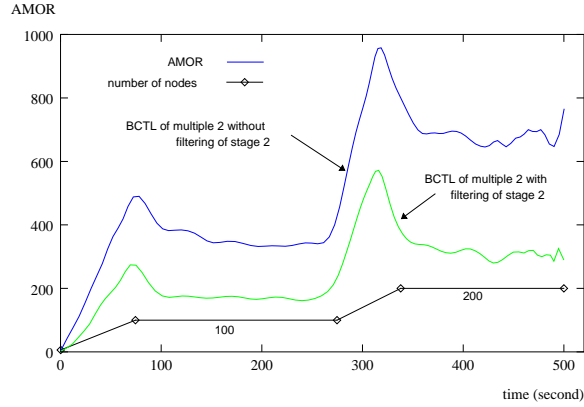


Figure 14: Relationship between AMOR and filtering of stage 2

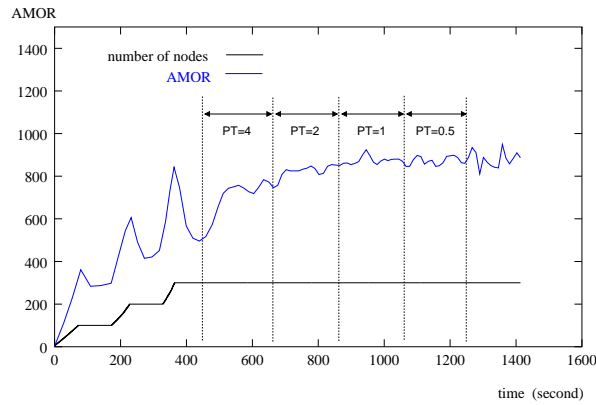


Figure 15: Relationship between AMOR and dynamic variation

300 nodes initially. After the system is stable, the system enters the dynamic variation stage by failing one node and generating a new node simultaneously per PT seconds, where $PT=4, 2, 1, 0.5$. Figure 15 shows that AMOR increases when PT decreases. This situation is obvious when PT changes from 4 to 2 seconds and from 2 to 1 second. However, the increase is getting smaller. This result shows that the SMRB mechanism adapts well to the dynamic variation of the system.

4.3 Observations of the RAL mechanism

The RAL mechanism provides the ability of reconnecting the fragmented system. We set up a system of 800 nodes, which will be removed one by one. By observing the number of distinct CIDs (ignoring the attached timestamps) with and without the RAL mechanism, we can verify the function of the RAL mechanism. A system of K components will lead to K distinct CIDs.

The experiment consists of four stages. In the first stage, 800 nodes are generated and connect to the system one by one. In the second stage, the whole system runs into a stable status. In the meanwhile, we begin to record the number of distinct CIDs. From the beginning of the third stage, nodes are removed from the system randomly. At the end of the third stage, 90% of the

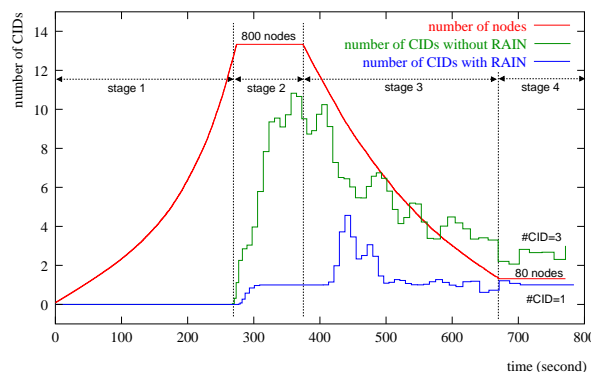


Figure 16: The number of distinct CIDs under the dynamic behaviors of the system

nodes (i.e., 720 nodes) are removed. The system runs into another stable status from the beginning of the fourth stage.

Figure 16 shows that the number of distinct CIDs, without the RAL mechanism, grows fast in the second stage. That value remains larger than 1 from the second stage to the end of the fourth stage. The average number of distinct CIDs is 5.74, which means that the system is split into 5.74 components in average. When the RAL mechanism is enabled, the number of distinct CIDs remains 1 most of the time and at the end of the experiment. The average number of components is 1.48 when the RAL mechanism is enabled.

The RAL mechanism elects one unique CID on a component by information exchange or request broadcasts. However the dynamic variation of the system will affect the performance of RAL. In the example of Figure 16, there are two types of dynamic variations: the disconnections of neighbors with longer distances by the NCC mechanism and the failures of nodes in stage 3. These dynamic variations make the number of distinct CIDs pulsated during the period when the RAL mechanism is enabled. When the system runs into a stable status (the number of nodes is fixed and the neighbors of every node mostly remained unchanged), the pulsation of the number of distinct CIDs will stop changing and the number of distinct CIDs becomes 1 gradually.

The last important observation is the relation between the failure of nodes and the average number of hops per request. Two well-known P2P projects, Gnutella and Freenet, both suffer from the failure of nodes. The research [6] shows that when the number of failed nodes exceeds a certain percentage (40% in that experiment [6]), the average number of hops grows extremely fast in both Gnutella and Freenet. Figure 17 shows that with the cooperation of the NCC and RAL mechanisms, the average number of hops per request seems to remain under a constant upper bound.

5 Conclusion

The DSE system is a completely distributed, self-organizing P2P system. The NCC mechanism clusters nodes into groups. The SMRB mechanism provides a method to broadcast messages at low cost. The RAL mechanism guarantees the connectedness of the P2P system and the performance of searching on it.

The DSE system focuses on the infrastructure of a distributed, self-organizing system. Any P2P application can use the infrastructure to simplify the design and the implementation.

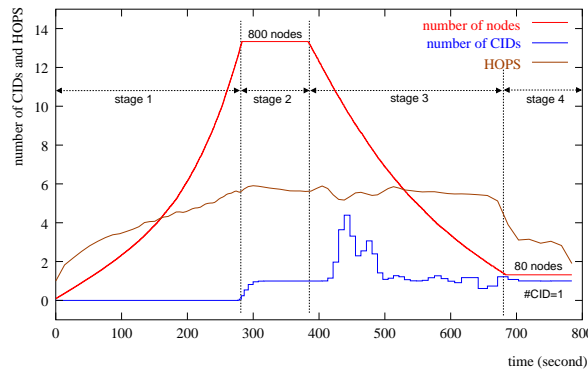


Figure 17: Connectedness and HOPS under the dynamic behaviors of the system

References

- [1] Karl Aberer, Magdalena Puceva, Manfred Hauswirth and Roman Schmidt. Improving Data Access in P2P. *IEEE Internet Computing*, 6(1), 2002.
- [2] Andy Oram. *Peer-to-Peer Harnessing the Power of Distributed Technologies*. O'Reilly 2001.
- [3] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, vol. 363, pp. 202-204.
- [4] Sylvia Ratnasamy, Scott Shenker and Ion Stoica. Routing Algorithms for DHTs: Some Open Questions. *First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.
- [5] Matei Ripeanu, Ian Foster and Adriana Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. *IEEE Internet Computing Journal*, 6(1), 2002.
- [6] Ian Clarke, Oskar Sandberg, Brandon Wiley and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Lecture Notes in Computer Science*, vol. 2009, pp. 46+ ,2001.
- [7] RFC 792. Internet Control Message Protocol. <http://rfc.sunsite.dk/rfc/rfc792.html>
- [8] The Free Network Project. <http://freenetproject.org/>
- [9] Tapestry. <http://jakarta.apache.org/proposals/tapestry/>
- [10] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, vol. 2218, pp. 329+, 2001.
- [11] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan. Chord: A Scalable Peertopeer Lookup Service for Internet Applications. Technical Report TR-819, MIT, March 2001.
- [12] RFC 1305. Network Time Protocol. <http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1305.html>
- [13] RFC 1772. Autonomous System. <http://www.faqs.org/rfcs/rfc1772.html>
- [14] RFC 1321. The MD5 Message-Digest Algorithm. <http://www.faqs.org/rfcs/rfc1321.html>

- [15] RFC 3174. US Secure Hash Algorithm 1 (SHA1). <http://www.faqs.org/rfcs/rfc3174.html>
- [16] Bruteforce your hashes. <http://mdcrack.df.ru/>
- [17] PHP: Hypertext Preprocessor. <http://www.php.net/>
- [18] APIA: Advanced P2P infrastructure and architecture. <http://clonefab.net/>
- [19] Kunwadee Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. The O'Reilly Peer-to-Peer and Web Services Conference, September 2001.
- [20] Ching-Wei Huang and Wu Yang. Efficient Anonymity Implementation in Peer-to-Peer Systems, in preparation, 2003.