IJCSN International Journal of Computer Science and Network, Volume 5, Issue 2, April 2016
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 1.02

Avoidance of Duplicate Messages In p2p Network Using Cycle Minimization

[1] C. Santhiya, [2] T. Manju

[1, 2] Information Technology, Anna University, Thaigarajar College of engineering
Madurai, Tamil Nadu, 625017, India

Abstract - Peer-to-Peer (P2P) networks, including flat and two-layer super-peer implementations are extremely popular nowadays due to their simplicity, ease of deployment and versatility. The p2p network contains many cyclic paths which introduce numerous duplicate messages in the system. While such messages can be identified and ignored, they still consume a large proportion of the bandwidth and other resources, causing bottlenecks in the entire network. In this paper We describe DCMP,a dynamic ,fully decentralized protocol which reduces significantly the duplicate messages by eliminating unnecessary cycles. As queries are transmitted through the peers ,DCMP identifies the paths to break the cycles, while maintaining the connectivity of the network. With the information collected during this process distributed maintenance is performed efficiently even if peers quit the system without notification.DCMP can be easily implemented in various existing P2P systems.

Keywords - *Network Protocols, Distributed Systems, p2p.*

1. Introduction

In a P2P system numerous nodes are interconnected and exchange data or services directly with each other. There are two major categories of P2P architectures:

(i) Hash-based systems , which assign a unique key to each file and forward queries to specific nodes based on a hash function. Although they guarantee locating content within a bounded number of hops, they require tight control of the data placement and the topology of the network.

(ii) Broadcast-based systems, which use message flooding to propagate queries. There is no specific destination; hence, every neighbor peer is contacted and forwards the message to its own neighbors until the message's lifetime expires. Such systems have been successfully deployed in practice.

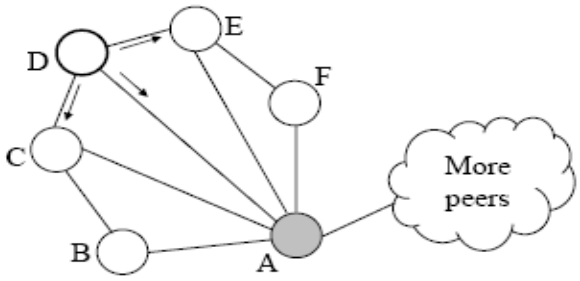


Fig 1: Example Network.

Assume the network topology of fig (1) and let peer D initialize a query message **msg**.**D** broadcasts msg to A,C times. Existing systems tag query messages with a unique identifier and each peer maintains a list of recently received messages. When a new message arrives, the peer checks whether it has already been received through another path. If this is the case, it simply ignores the incoming message. This method is called as Naïve Duplicate Elimination (NDE).

Here I focus on broadcast-based P2P architectures. My methods are also applicable to two-layer networks based on super-peers. The main characteristics of Distributed Cycle Minimization Protocol are:

(i) It reduces duplicate messages by as much as 90%,

(ii) It requires few control messages, therefore the overhead is minimal.

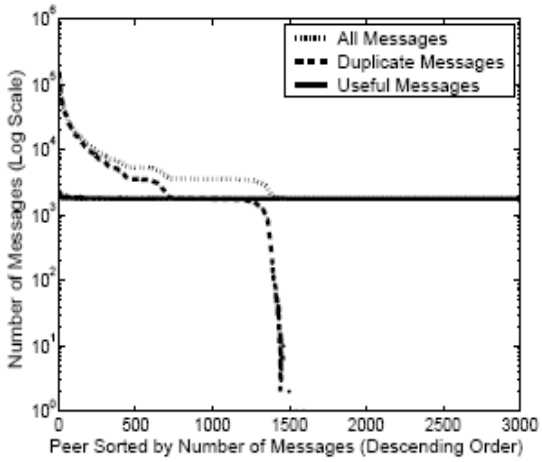


Fig 2: Total no of Duplicate messages

Duplicate messages affect severely the response time and scalability of P2P systems, since they consume bandwidth and system resources, primarily from high-degree peers which are crucial for the connectivity of the network. Distributed Cycle Minimization Protocol (DCMP) which aims at cutting the cyclic paths at strategic locations, in order to avoid introducing duplicate messages in the network. In DCMP any peer which detects a duplicate message can initiate the cutting process. This involves two steps: First, peers in the cycle elect a leader, called GatePeer. At the second step, the cycle is cut at a well-defined point with respect to the gate peer. GatePeers are also important for maintaining the connectivity and optimal structure of the network when peers enter or quit without notification. Since any peer can become GatePeer via a distributed process, the system is resilient to failures.

The rest of my paper is organized as follows: Section II presents the related work. Next, in Section III I describe the main aspects of DCMP, whereas in Section IV I discuss how DCMP deals with dynamic networks. In Sections V and VI I present he experimental results from my simulation. Finally, Section VII concludes the paper and discusses directions for future work.

2. Related Work

Yang and Garcia-Molina observed that the Gnutella protocol could be modified in order to reduce the number of nodes that receive a query, without compromising the quality of results. They proposed three techniques:

(i) Iterative Deepening, where multiple BFTs are initiated with successive larger depth.

(ii) Local Indices, where each node maintains and local index over the data of all peers within r hops of itself, allowing each search to terminate after d – r hops.

(iii) Directed BFT, where queries are propagated only to a beneficial subset of the neighbors of each node. Several heuristics for deciding these neighbors are described. A similar technique, called Interest-based Locality , contacts directly the most promising peer which is not necessarily a neighbor of the query initiator. If this does not return enough results, a normal BFT is performed. Note that none of these techniques deals with duplicate elimination, but they are orthogonal to this protocol.

Limewire maintains a table where it stores the IDs of duplicate messages and the directions (i.e.,neighbor peers) from where they arrive. Once a message is identified as duplicate, it is discarded. Further message propagation avoids the directions from where duplicates have arrived. Keeping (ID,Direction) in formation for each duplicate message requires additional memory, especially in high-degree peers as they tend to receive a lot of duplicates. Therefore, Limewire also implements a simplified version which disables those connections from where "a lot" of duplicates are arriving. In practice, it is difficult to define an ambiguously the disconnection threshold. Moreover this method may compromise the connectivity of the network, as we show in our experiments.

Let peer A receive a query message. Gia routes the message as follows: A → B → C → A. Therefore, A receives a duplicate. Since A knows that it has already sent the message to B, this time it chooses D.

3. Protocol Design

These sections describes the protocol in details and explain why it is superior to existing approaches. To assist my discussion, first I present the notation I use throughout this paper.

- When a node generates a query message msg, the message is assigned a globally unique ID denoted as: GUID(msg).

- Let A and B be two neighbor nodes (i.e., they have a direct overlay connection). The connection between them is denoted as: AB

- Let a message travel from A to B . We denote the direction of the travelled path as: A → B and the reverse direction as B → A.
- Let A receive a message msg from its neighbor B .

Then A places the following pair into the history table:

(GUID(msg), B → A)

A. DCMP: Distributed Cycle Minimization Protocol

In contrast to SCE, this protocol requires negotiation among all peers involved in a cycle about the optimal way to cut the cycle. Therefore, the probability of generating a disconnected network is minimized. The negotiation process is efficient, requiring only two messages per peer per cycle. Also, the information gathered during negotiation is used to repair the network with low overhead when peers join or fail/quit without notification.

The negotiation process can be initiated by any peer who receives a duplicate. Fig. 2 provides an example. Assume that peer A receives a message msg from B → A and, soon after, it receives the same message[2] (i.e., same GUID) from F → A. Peer A identifies msg as duplicate by performing a lookup in its history table. The first step of this protocol is to gather information from all peers in the cycle. To achieve this, I introduce a new type of control message, called Information Collecting (IC) messages.

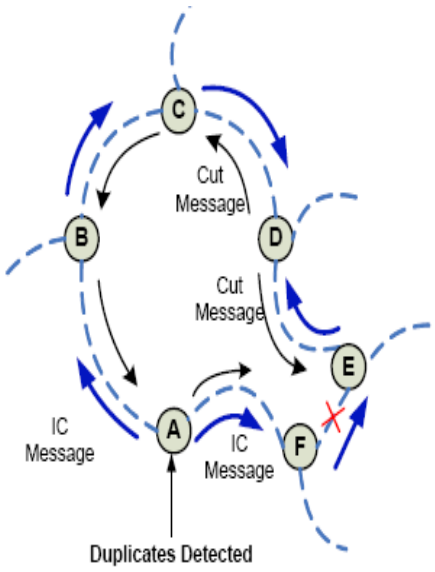


Fig 3 : DCMP Representation.

The *GUID*(*icm*) is set to be the same as the *GUID* of the duplicate *msg*. This is done in order to facilitate the propagation of *icm* by the same mechanism which handles query answers in a Gnutella-style network. Note that if *msg* travels through many cyclic paths, multiple peers will detect the duplicates. To ensure that each IC message is unique I introduce another field, called *DetectionID*, which represents the direction of the connection where the duplicate was identified.

In my example, *DetectionID*(*icm*) ≡ *F* → *A*. The last field of the IC message is the *Node Information Vector* (NIV). NIV contains information about the peers which propagated the IC message. This includes the bandwidth of each peer, the processing power, the IP address and topology information about the peer's degree and its neighbors. In our example, the NIV of *icm* initially contains information only about peer *A*. Direction is not important in this field since any of the two nodes in the pair can disable the connection. Peer *D* sends two copies of the cut message towards *D* → *C* and *D* → *E*, respectively.

Algorithm for handling IC message

Precondition: Node *N* receives an IC message *icm*, from direction *M* → *N* 1. Search the history for a recent IC message *icm'* which satisfies:

1. **GUID**(*icm*) = *GUID*(*icm'*) **and**

   DetectionID(*icm*) = *DetectionID*(*icm'*)

2. **if** *icm'* is found, **then** // a duplicate IC message is found
3. Combine NIV of *icm* and *icm'* into a single vector *v*

   At this point, *v* contains information about all the nodes in the cycle.

4. Using *v*, decide which connection in the cycle will be disabled.
5. Forward the decision to all the nodes in the cycle
6. **else** // no duplicate IC found
7. Append the node information of *N* to the NIV field in *icm*
8. Find in the history a message *msg* such that

   GUID(*msg*) = *GUID*(*icm*)

9. Assume that *icm* is an answer message for *msg*

Use Gnutella protocol to send *icm* towards the reverse path of *msg*

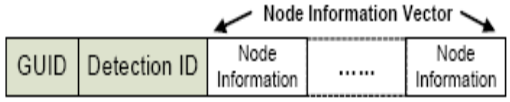


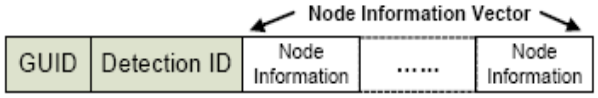
Fig 4: Structure of IC message

From the NIVs of the icm messages, D has information. all nodes in the cycle, namely A, B , C , D, E and F . Using this information D decides which connection should be disabled. For now the other peers in the cycle about the decision.

Algorithm for Cut message

Precondition: Node N receives a cut message cm

1. **if** N is involved in the connection to be disable **then**
2. **if** the corresponding connection is still active **then disable it.**
3. **else**
4. Search the history for an IC message icm such that $GUID(icm) = GUID(cm)$ **and** $DetectionID(icm) = DetectionID(cm)$
5. **if** such icm is found **then** forward cm to the reverse



$|p| = |p'| - 1 = |S_N|/2$ if $|S_N|$ is even

Peer power: The power P of a node N is given by the following formula:

$$P(N) = c_1 B + c_2 C + c_3 D$$

where B is the bandwidth, C is the CPU processing power, D is the peer's degree (i.e., maximum number of simultaneous connections) and $c_{1...3}$ are predefined constants.

It is obvious why the bandwidth and CPU power characterize how powerful a peer is. The degree factor is used, because a peer which accepts many neighbors is beneficial for low network diameter. There are several other factors which can influence the characteristics of the peer power.

GatePeer*:* The most powerful peer in a cycle is called GatePeer.The heuristic we use in our protocol is to cut cycles by disabling the connection which is opposite to the corresponding GatePeer. The intuition is that our method minimizes the average number of hops from the GatePeer to any peer in the cycle. The GatePeer, in turn, will most probably be the hub which connects the cycle to many other peers; therefore, the connectivity will be largely preserved. Also, since the GatePeer can process messages fast, the response time will not suffer.

Opposite edge: Let S_N be the set of nodes which form to it is an edge MM' such that: $M \in S_N$, $M' \in S_N$, and there is a path p from N to M and a path p' from N to M' such that $p \subset S_N$, $p' \subset S_N$ and:

$|p'| = \lceil |S_N|/2 \rceil$ if $|S_N|$ is odd.

Algorithm for selecting gate peer

Preçondition: Node N receives two IC messages icm and icm which satisfy the conditions:
GUID(icm) = GUID(icm′) and
DetectionID(icm) = DetectionID(icm′)

1. Calculate the power P_i of each peer in NIVs
2. Let the peer with Max(P_i) be the Gate Peer
3. In case of a tie, the Gate Peer is the one with the largest GUID
4. Find the position to be disabled based on the GatePeer.
5. Generate cut messages accordingly.

4. Dynamic Network

Disseminating GatePeer Information

GatePeers assist to recover from node failures and are used as entrance points in a dynamic network therefore, it is beneficial for other peers outside the cycle to know which are the nearby GatePeers. To disseminate this information with minimal overhead, we use a piggyback technique. Each GatePeer appends the messages passing through it with the following information: (NIV_{GP} , HopsNumber), where NIV_{GP} is the information vector of the GatePeer (including its IP address)and HopsNumber is an integer indicating the distance in hops from the message origin to the GatePeer. This process is called as tagging. While the overhead of tagging is only a few bytes of peer message, the GatePeer information remains relatively stable for most of the time. Therefore, we can achieve our goal by tagging messages periodically.

Observe that immediately after cycle is eliminated, most probably a new GatePeer is elected. In order to advertise fast its identity, the GatePeer performs tagging frequently. Later, the GatePeer tags messages infrequently to peers up to T T L hops away realize that it is still alive. My results suggest that the following settings provide a good tradeoff between cost and efficiency: for a period of 1 min after a new GatePeer is elected, a message is tagged every 5 sec; after that, the tagging frequency is lowered to 1 message every 10 min.

Transitive Peer:

A peer that continuously receives tagged messages from more than one direction is called a transitive peer. Peers may receive tagged messages from several gate peers continuously. If the tagged messages do not come all from the same direction, it is possible that the peer is a hub.

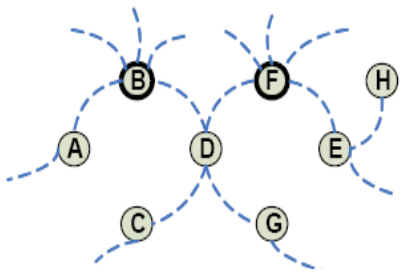


Fig 5: Example of transitive peer

GatePeer Departure

All peers, including GatePeers, receive tagged messages periodically; therefore, they have a list of near by gate peers(the transitive peers are also handled as GatePeers).From this information, a GatePeer G knows its distance to each of the nearby GatePeers. Taking into account the distance and power of these GatePeers, G generates an ordered list of backup gatepeersG broadcasts this list to its direct neighbors (i.e., only one hop away). The guideline for selection is that the backup gatepeers should be powerful enough to accept the direct neighbors of G, in case G quits. In our experiments, we found two to five backup GatePeers were usually selected, depending on the degree of G and the capacity of its neighboring GatePeers.

If G quits/fails, its neighbors attempt to repair the network.The backup GatePeers of G connect to each other .The rest of G's neighbors attempt to connect to some backup GatePeers randomly. Therefore, only a small number of peers (i.e the direct neighbors of G) are affected and the network topology does not change significantly. If for some reason this process is not successful (e.g., none of the backup GatePeers can accept more connections, because of simultaneous GatePeer failures), then the affected peers simply re-join the network using the peer arrival procedure described above.

Departure of Normal Peer

If a normal peer quits/fails we must also ensure that the network remains connected. In contrast to GatePeer failures, this case affects only neighbors whose primary direction includes the quitting node.

Peer A is a GatePeer and it is also the referred GatePeer of both C and D. Assume that B fails (B is a normal peer) and note that the primary direction of C and D is C → B and D → B , respectively. Recall that the primary direction indicates the preferred path towards the rest of the network. Therefore, B 's failure is likely to affect the connectivity for the subgraphs under C and D. In our protocol, the affected peers attempt to connect to their referred GatePeer; hence, C and D will connect to A.

Peer Arrival

In existing Gnutella-style networks, joining nodes first contact some well-known peers and send ping messages which are broadcasted in the network. Peers willing to accept the new connection, reply with a pong message.

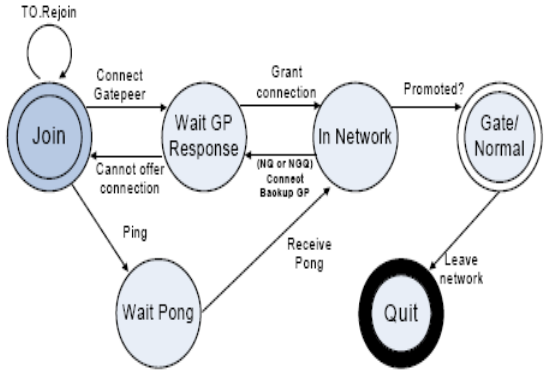


Fig 6 : Notification of peers in network

Assuming that the newcomer peer N was in the network before it is possible that it has cached the IP addresses of some GatePeers. N attempts to contact the GatePeers, hoping they are still in the network. The

intuition is that GatePeers are powerful and most probably can accept the new connection. Even if there are no free resources at the moment, a GatePeer G can recommend to N a new set of GatePeers in G's vicinity. Given that this process succeeds, N is able to join the network without the overhead of broadcasting a large number of ping messages. The savings can be substantial if nodes join/leave the network frequently.

Influence of Network Size

Fig. (7) shows the network coverage. The graph reveals that DCMP preserves short routing paths as the network size increases. DCMP eliminates only the small cycles around GatePeers, achieving almost as good coverage as Gnutella[6] . In Fig.(8) present the average number of duplicates for various network sizes.

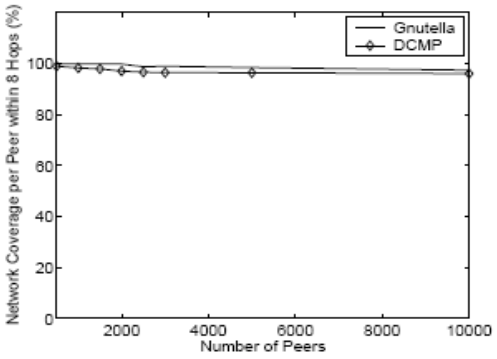


Fig 7: Connectivity

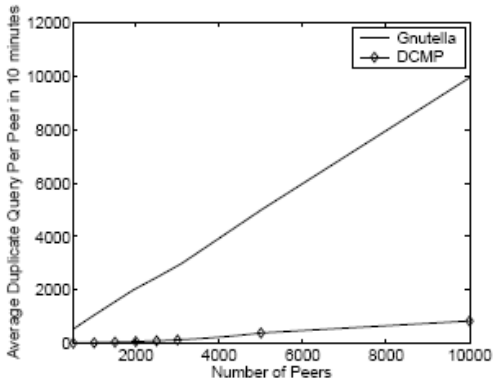


Fig 8: Duplicate query

For DCMP the number of duplicates increases very slowly, since the number of cycles with length larger than 2. TTld (i.e,the ones that introduce duplicates in DCMP)is small.

5. Symmetric Cut vs. Random Cut

Here we investigate the effectiveness of the symmetric cut heuristic employed by DCMP. The comparison of my method against cutting the cycle at a random position. The results are shown in Fig.), where we draw the network coverage for varying number of hops. By cutting cycles symmetrically to gatepeers.DCMP manages to follow closely the good coverage of Gnutella. The random heuristic, on the other hand, creates long chains of peers and network fragments, since all peers in a cycle may decide to break the cycle concurrently.

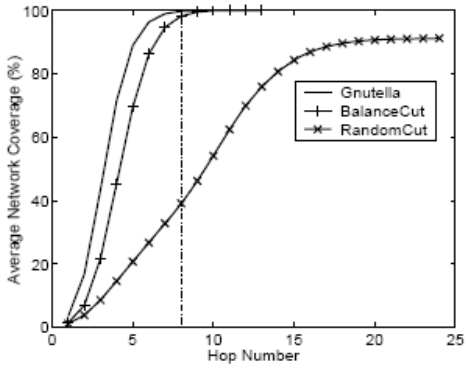


Fig 9: Randomcut

Failure and Attack Analysis

In peer-to-peer systems, peers are usually unstable and the network is very dynamic. One important requirement of the system is to be resilient to failures. The failure can be detected either when a peer sends a message or when the "KeepAlive" timer of the TCP layer expires.

Comparison with other Approaches

Random Walks (RW) for searching in unstructured P2P networks. The algorithm initiates k random walkers. In order to reduce the system load, the walkers contact the initiator node periodically, to check whether the search should stop. RW favors searches for popular objects but exhibits poor performance for rare ones. If RW is forced to transmit the same number of messages as flooding approaches, it achieves almost the same network coverage. RW does not alter the network's structure.

In LTM, the following two steps are performed at each peer:

(i) Forward a detection message: if a detection message

(received or self-created with initial T T L = 2) has not expired, the peer inserts a new timestamp and broadcasts the message to the neighbor peers.

ii) Cut a connection: upon receiving two detection messages with the same GUID, the peer drops the link with the largest delay among all traversed links, using the timestamps to calculate delays.

6. Conclusions

In this paper DCMP is described, a protocol for distributed cycle minimization in broadcast-based P2P systems. It preserves low diameter while eliminating most of the duplicate messages. The overhead due to control messages is minimal. This results in reduced response time, which in turn increases the scalability. The protocol is suitable for dynamic networks, since it handles peer joins/departures efficiently and is resilient to failures. DCMP is also designed to be as simple as possible and is independent of the search algorithm. Therefore, it can be implemented on the top of popular P2P systems. A prototype implementation is used to verify that this techniques are applicable to realistic environments.

References

[1] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. "a Scalable Content-Addressable Network," in Proc. of ACM SIGCOMM,2001pp.161-172.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer- To peer lookup Protocol for Internet Applications," IEEE/ACM Transactions on Networking, vol. 11, no. 1,pp. 17-32, 2003.

[3] Gnutella, http://www.gnutella.com/ and http://groups.yahoo.com/group/the gdf/.[4] Kazza, http://www.kazaa.com/.

[5] M. Ripeanu, A. Iamnitchi, and I. T. Foster, "Mapping the Gntella Network",IEEE Internet Computing vol6,no1,pp 50-57,2002.

[6] Q. Lv, P. Cao, E. Cohen, K. Li,and S. Shenker, "Search and Replication in Unstructured Peer-to-peer networks", in Proc. of Int. Conf. on Supercomputing (ICS), 2002, pp. 84-95.

[7] PlanetLab, http://www.planet-lab.org/.

[8] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, Shenker, "Making Gnutella-Like P2P Systems Scalable," in Proc. of ACM SIGCOMM,2003,PP.407-418.

[9] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," ACM Computing Surveys, vol. 36, no. 4, pp. 335-371, 2004.

[10] V. Cholvi, P.A. Felber, and E.W. Biersack, "Efficient Search in Unstructured Peer-to-Peer Networks," European Trans. Telecomm., vol. 15, no. 6, 2004.