EFFECTIVE TOPOLOGY-AWARE PEER SELECTION IN UNSTRUCTURED PEER-TO-PEER SYSTEMS

Yen-Cheng Chen and Chen-Chih Liao

Department of Information Management, National Chi Nan University, Nantou, Taiwan
ycchen@ncnu.edu.tw, catsky@yahoo.com.tw

ABSTRACT

Peer-to-Peer systems form logical overlay networks on top of the Internet. Essentially, peers randomly choose logical neighbours without any knowledge about underlying physical topology. This may cause inefficient communications among peers. This topology mismatch problem may result in poor performance and scalability for Peer-to-Peer systems. A possible way to improve the performance of Peer-to-Peer systems is the overlay network construction based on the knowledge of the physical network topology. In this paper, we will propose the use of the "Record Route" and "Timestamp" options supported in the IP protocol to explore the paths between peers. By the topology-aware peer selection, our approach outperforms traditional P2P systems using random peer selection. Our approach only incurs a low overhead and can be deployed easily in various P2P systems.

KEYWORDS

Peer to Peer, Topology Awareness, Peer Selection, IP, Route Record, Timestamp

1. INTRODUCTION

Peer-to-Peer (P2P) applications have become very popular in the Internet. Some traffic measurements show that P2P traffic is starting to dominate the bandwidth in certain segments of the Internet [1]. In a typical P2P system, a peer realizes its neighbouring peers at the application level. Accordingly, peers only know the existence of other peers somewhere in the Internet, but are not aware of how far the peers are physically. The network logically constructed by peers is called an overlay network. Each logical link of the overlay network corresponds to a path in the underlying Internet. Since there exists a topology mismatch between the overlay network and the Internet, communications through an overlay network may be inefficient and may consume much overall traffic. Therefore, such a P2P file sharing mechanism will incur a large amount of unnecessary traffic and may spend much more time in downloading files. This mismatch problem has been described in [2]. Since peer nodes of an overlay network may be distributed over the Internet, P2P traffic flows may spread over a number of ASs (Autonomous Systems). Consequently, P2P applications may consume more overall Internet traffic than client-server applications. It is shown in [3] and [4] that P2P traffic contributes the largest portion of the Internet traffic, based on measurements on some popular P2P systems, including KaZaA [5], Gnutella [6], and DirectConnect [7]. For better understanding of bandwidth use in P2P communications, a mathematical model for P2P bandwidth is proposed in [8]. Measurements in [9] also show that the flooding-based routing generates 330 TB/month in a Gnutella network with only 5000 nodes even if 95% of any two nodes are less than 7 hops distant. This heavy use of Internet traffic gets worse as the number of peer nodes increases. This leads to poor scalability in unstructured P2P systems [10]. This also raises a demand for a better use of Internet bandwidth for P2P communications.

There are three major categories of P2P systems: centralized, decentralized structured and decentralized unstructured. In the centralized model, centralized index servers are employed to maintain a directory of shared files stored in peers so that a peer can easily find the location of a desired content from an index server. But, this model suffers from the risk of a single point of

DOI : 10.5121/ijp2p.2011.2101

failure. In decentralized P2P systems, without the need of any centralized index server, a peer finds desired contents and downloads them directly from other peers. In the decentralized structured model, such as Chord [11], Pastry [12], Tapestry [13], and CAN [14], the shared data placement and topology characteristics of the network are strictly controlled based on distributed hash functions. On the other hand, the decentralized unstructured architecture doesn't have any hierarchical structure among peers. In other words, all peers are of the same role [15] and files are distributed in peers in a random manner, independent of the actual network topology of peers [16]. Decentralized unstructured P2P systems are most commonly seen in today's Internet. The main theme of this study is the peer selection issue in decentralized unstructured P2P systems.

Among unstructured P2P systems, BitTorrent [17][18] is one of the most popular P2P applications nowadays [19]. We choose BitTorrent as the target P2P system whose performance is evaluated and improved by our proposed scheme. In fact, BitTorrent is a P2P communication protocol, which is supported by a lot of BitTorrent client programs. In this paper, the BitTorrent protocol is first reviewed and an improved scheme is then proposed to be compliant with the current BitTorrent protocol. Without any change of the BitTorrent protocol, the proposed approach is developed based on the option support in IP headers, which are available in all BitTorrent messages. By enabling the "Record Route" (RR) option of the IP header, we can infer the length of the path between two peers during their initial BitTorrent conversation. By enabling the "Timestamp" option of the IP header, we can further infer the latency of message transmissions between two peers. Based on the path and latency information, the proposed solution constructs an effective overlay network with a logical topology partially resembling the physical network topology. It will be shown that the traffic and latency incurred by BitTorrent communication can be reduced by selecting near peers in terms of path length and transmission latency.

2. PRELIMINARIES

In this section, we will review the BitTorrent protocol and give a brief overview of the Record Route and Timestamp options in the IP header.

2.1. Overview of BitTorrent

BitTorrent is a P2P application designed to facilitate fast downloading of popular files. The basic idea of BitTorrent is to divide a single file into pieces with size at most 256 KB. A BitTorrent client, who attempts to download a file, will first get a seed-file for the file. The seed-file describes the location of a Tracker server and the hash values of each piece of the file. A Tracker server is responsible for maintaining a list of peers who are participating in the sharing of a certain file. Figure 1 illustrates the entities involved in BitTorrent and the initial procedure for downloading files. After obtaining the seed-file for a file to download, a client will get connected to the corresponding Tracker server. The Tracker server will give the client a list of peer IDs which are participating in the file sharing. The default number of peer IDs included in a list is 50. The number can be changed if required. If there are fewer peers in the current torrent of file sharing, a shorter list will be returned. If more than 50 peers are participating in file sharing, the Tracker will randomly pick up 50 peers to include their ID in the list. After receiving a list of peer IDs, the client then send Handshake messages to all the peers to gather further information for effective downloading. A Handshake message consists of the peer's ID and some hash values which are transmitted in tracker requests. If a peer receives a Handshake with the hash values that it is not currently serving or the peer's ID does not match the expected peer's IDs, then the peer must drop the connection. After handshaking is completed, each of the requested peers will immediately send a bit-field message to the client. Each bit of the bit-field message, corresponding to a specific piece of the wanted file, indicates

whether a specific piece is owned by the requested peer. Thus, from the received bit-field messages, the client could determine the most effective way to download all the pieces of a file from a set of peers. To be allowed to download a file piece, the client will send an "interested" message to the peers who have the specific piece. After completing the download of a piece, the client verifies the downloaded piece by comparing its hashed value with the hashed value declared in the seed-file. The client then sends a HAVE message to other peers to announce that the client has a copy of the piece.

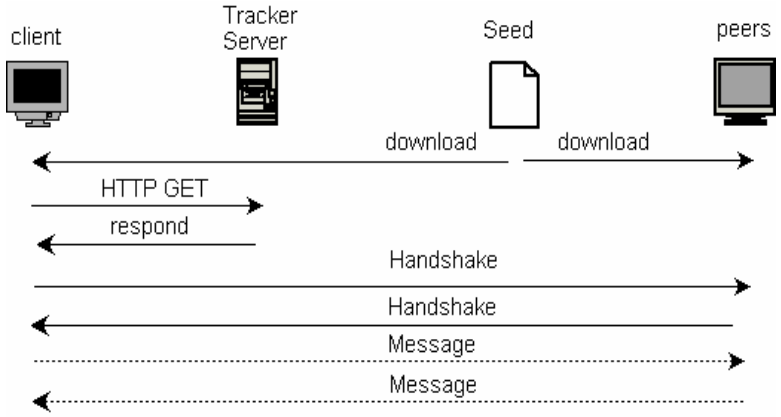

Figure 1. The initial procedure of BitTorrent downloading

In order to facilitate the effective distribution of file pieces, a Rarest First algorithm [17] is used to determine which piece should be downloaded first. By keeping the received bit-field messages and updating them whenever receiving any HAVE message, a client knows which piece is indicated least frequently in those bit-field messages. The piece will be downloaded first. In addition, each peer allows at most four clients to download pieces from it at the same time. If there are more than four clients attempting to download pieces from the same peer simultaneously, these clients will be scheduled to download pieces in turns with an interval of 30 seconds. For the purpose, a peer allows or disallows downloading requests by sending back "Unchoking" or "Choking" messages respectively. The reciprocation of Choking and Unchoking is maintained in four flags for each client (called remote peer).

- am_choking: the local peer is choking the remote peer
- am_interested: the local peer is interested in the remote peer
- peer_choking: the remote peer is choking the local peer
- peer_interested: the remote peer is interested in the local peer

Basically, a peer allows four remote peers for downloading at the same time and changes the allowed peers in periods of 30 seconds. This rule can be overwritten by the Optimistic Unchoking algorithm [17], which first selects the interested peers with best upload rates. To ensure fairness, each peer uses a tit-for-tat algorithm to get a consistent download rate. In addition, there are several criteria that a good choking algorithm should meet. First, the number of simultaneous uploads should be limited for good TCP performance. Choking and unchoking quickly, known as "fibrillation", should be avoided. All the peers should have opportunity to download pieces. Finally, choked connections could be unchoked if they are anticipated to have better throughput than the current unchoked ones. According to BitTorrent's specification, whenever there a request from a client, the Tracker server randomly picks up 50 peers among the current peers participating the sharing of a file. Therefore, it is very possible that the selected neighboring peers are far from the client in the physical network topology. This will increase both transmission latency and traffic overhead for file sharing.

2.2. IP Header's Options: Record Route and Timestamp

In this section, a brief introduction of the options in the IP header is given. As defined in RFC 791 [20], the last part of the IP header is a list of optional fields. Several types of IP header options are defined. Each type of options is identified by the option-type octet, which appears at the first octet of an option. The option-type octet consists of three fields: copied flag (one bit), option class (two bits) and option number (five bits). The copied flag indicates if the option should be copied to each fragment of an IP packet. Option class and number are used to classify options. Table 1 lists the option class, option number, length, and description of each option. In this paper, we will adopt the *Record Route* and *Timestamp* options for path cost evaluation.

Table 1. Options in the IP header.

Class	Number	Length	Description
0	0	-	End of Option list. This option occupies only 1 octet; it has no length octet.
0	1	-	No Operation. This option occupies only 1 octet; it has no length octet.
0	2	11	Security. Used to carry Security, Compartmentation, User Group (TCC), and Handling Restriction Codes compatible with DOD requirements.
0	3	var.	Loose Source Routing. Used to route the internet datagram based on information supplied by the source.
0	9	var.	Strict Source Routing. Used to route the internet datagram based on information supplied by the source.
0	7	var.	Record Route. Used to trace the route an internet datagram takes.
0	8	4	Stream ID. Used to carry the stream identifier.
2	4	var.	Internet Timestamp.

The *Record Route* option allows an IP packet to record its route in the IP header. If this option is present in an IP packet, each router which forwards this packet will append its IP address to the option. As depicted in Figure 2, the *Record Route* option begins with an option type of value 7. Following the option type field is the length field to specify the total length of the option, including option fields and appending route data. The pointer field indicates the position in the route data where the next route address should be placed. A recorded route is composed of a sequence of IP addresses. If the value of the pointer is greater than the length, the recorded route area becomes full. To accommodate all the IP addresses of the routers in a route, the originating host of the IP packet should specify the length field with a value large enough to place IP addresses in the route data area.

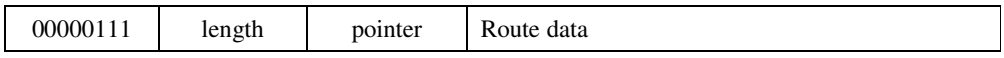

Figure 2. The Record Route option

When forwarding a datagram, a router checks to see if the record route option is present. If present, the router inserts its own IP address into the record route option beginning at the octet indicated by the pointer and increments the pointer by four. If the route data area is already full, the datagram is forwarded without inserting the IP address into the record route.

Similar to *Record Route*, the *Timestamp* option is used to record the timestamps of the routers within the route of an IP packet. Whenever receiving an IP datagram with the *Timestamp* option in its header, a router will inserts the current timestamp into the routed datagram. As illustrated

in Figure 3, the option-type code of *Timestamp* is 131. The length field is to indicate the number of octets of the option (maximum length 40). The pointer field indicate the position where the next timestamp will be appended. The timestamp area is full when the value of the pointer is greater than the length. The Overflow (oflw) field (4 bits) indicates the number of IP modules (i.e. routers) that cannot append timestamps into the option due to lack of space. The Flag (flg) is used to specify the way of appending timestamps:

0: timestamp only, stored in consecutive 32-bit words.
1: Each timestamp is preceded with the IP address of the registering entry.
2: The IP address fields are pre-specified. An IP module only registers its timestamp if it matches its own address with the next specified IP address

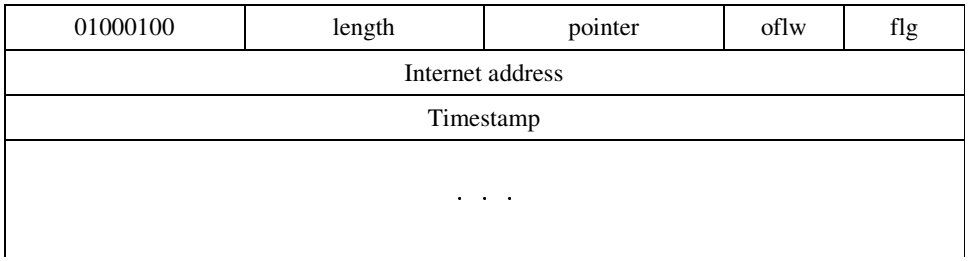

Figure 3. The Timestamp option

A timestamp is a right-justified, 32-bit one in milliseconds since midnight UT (Universal Time). If the time is not available in milliseconds or can't be provided with respect to midnight UT, then any time may be inserted as a timestamp provided the high order bit of the timestamp field is set to one to indicate the use of a non-standard value. If the timestamp data area of the forwarded datagram is full, the datagram is forwarded without inserting the timestamp, but the overflow count is increased by one.

3. RELATED WORK

There have been a number of approaches to improve the performance of unstructured P2P overlay networks [2][21][22][23]. Most of these approaches use proposed algorithms or particular protocols to avoid the waste of unnecessary traffic in long-distance message travelling over the Internet. In these approaches, physical network distances are estimated and near peers are selected for downloading files more quickly. Essentially, a peer will pick up the closest peer from its neighbours if network topology information is available. Indeed, we can manually use the *ping* or *traceroute* tool to estimate the network distance between source and destined nodes. However, these network tools were designed primarily for network diagnostics and are too heavy-weighted to be used by large-scale applications. For example, the use of BGP routing table dumps [24] can provide comprehensive topology information, but incurs the same problem. Furthermore, such topology information is obtained or learned from the underlying Internet layer and therefore cannot be directly available to end-user applications.

In [23], the authors proposed many approaches, including off-line and on-line ones, to perform accurate topology-aware operations for server selection using only static information. Based on the four metrics, namely geographic distance (DIST), number of hops (HOPS), number of Autonomous Systems (AS), and DEPTH, each peer could estimate the network distance between peers by regression analysis, which is also evaluated in [23]. Another approach proposed in [25] is the use of an extended DNS system providing location information of hosts, networks, and subnets. The geographic information of hosts and networks are added to the DNS system using the extension to the protocol as defined in RFC 1876 [26] (also see [26] for a similar implementation). That is, each network element will be associated with static location

information, such as latitude and longitude. The major idea is that two geographically near peers are most probably two ones with shorter network distance in the physical network. Many Internet users get connected to the Internet via dialup, DSL, ADSL or cable modem. DHCP (Dynamic Host Configuration Protocol) protocol is typically used for dynamically assigning IP address to these hosts. For use in the extended DNS system, all IP addresses available to a DHCP server can be associated with a single geographic location. We can query the DNS server to obtain the geographic information of particular peers. Therefore, one can select the closest server using regression analysis, as proposed in [23], to calculate the network distance metrics (i.e. round trip time) approximately between any pair of peers. It was shown that 90% accuracy for an allowed deviation of 30ms can be achieved to utilize geographic information in the evaluation of transmission latency based on the best off-line model proposed in [23]. This approach significantly outperforms the RANDOM approach, where the server is chosen randomly. The off-line approach can achieve the same performance as the best on-line model, Nevertheless, this off-line approach has an obvious disadvantage: a geographic information protocol, i.e., RFC 1876, should be supported in each DNS server. In practice, it is very difficult to deploy this approach in the Internet.

In order to estimate the network distance accurately for a better construction of an overlay network, SL algorithm and BinSL algorithm were proposed in [21]. In SL algorithm, a node picks its ***k*** neighbours by picking the $k/2$ nodes in the system closest to itself (these connections are called *short links*) and then picks another $k/2$ nodes at random (these connections are called *long links*). It was shown in [27] that if a greedy algorithm is used to merely pick up closer peer nodes, the constructed overlay network will become disconnected. Therefore, the SL algorithm picks up $k/2$ random links to keep the graph connected. This algorithm assumes that the latency between any peer pairs is available in an $n \times n$ IP-latency matrix. Obviously, SL algorithm is more efficient than the RAN algorithm, which chooses all nodes at random. The major drawback of the SL algorithm is the difficulty of maintaining the $n \times n$ IP-latency matrix, which also results in poor scalability [28].

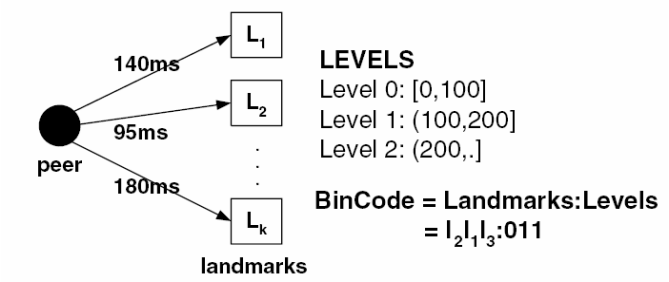

Figure 4. An example of the BinSL algorithm in [22]

Ratnasamy et al. [21] propose the distributed BinSL algorithm, which was based on the evaluated end-to-end delays using a system of k landmarks spread across the Internet. Each node measures the round trip times (RTTs) from itself to k well-known landmarks on the Internet. The ordering of the landmarks by the latencies represents the "bin" that the node belongs to. That is, nodes in the same bin are those with the same ordering of landmarks. The major idea of the BinSL algorithm is that each node will pick up $k/2$ nodes in the bin it belongs to and then picks up other $k/2$ nodes at random. In the algorithm, latencies are classified into *levels* by ranges of latency values. For example, we might divide possible latency values into 3 levels: level 0 for latencies in the range [0,100] ms, level 1 for latencies between (100,200] ms, and level 2 for latencies greater than 200ms. The landmark ordering for a node is then augmented with a level vector. Each level value of the vector corresponds to a landmark in the ordering. For example, the landmark ordering for the peer shown in Figure 4 is $l_2 l_1 l_3$. If three levels as defined above are used, the corresponding level vector will be "0 1 1". Thus, the bin of

the peer is denoted by "$l_2l_1l_3$:011". We can infer that nodes belonging to the same bin are usually topologically close to each other. It is possible that some nodes do belong to the same "bin" but are not topologically close to each other. In [22], it is seen that fewer landmarks will degrade the performance of the binning scheme and may increase the probability of improper binning. The performance of the BinSL algorithm is a little worse than SL Algorithm while constructing the overlay network. Each landmark should sustain the load of being pinged by all nodes to determine landmark orderings. In fact, the overhead is substantial. For example, assume there are a million nodes and each node needs 10 Pings sent to a landmark to obtain a good sample of RTT. If nodes refresh their bin information once per hour, it would impose a load of 2700 Pings per second on each landmark [21]. Although this drawback could be possibly overcome by the use of multiple collocated nodes acting as a single logical landmark, the overall communication cost is still high and the deployment of numerous landmark nodes complicates the implementation of the BinSL algorithm.

Zeinalipour-Yazti et al. [22] propose the use of domain names to select near peers. The proposed DDNO (Distributed Domain Name Order) method is a distributed technique to make unstructured overlay networks topologically aware. Each node participating in a DDNO topology has a domain name (*dn*), which is a string that conforms to the syntax rules of RFC 1035 [29]. In order to determine whether two peers, saying peer A and peer B with domain names dn_A and dn_B respectively, belong to the same domain, this approach defines two functions, called *Split-hash* and *dnMatch*. The Split-hash function is to split a domain name *dn* into *k* hashes, where *k* is the number of sub-domain strings in *dn*. After splitting the domain name, the procedure will use the $dnMatch(dn_A, dn_B)$ comparison function to compare the individual hashes of sub-domains in dn_A and dn_B. For example, if dn_A = "a.aol.com" and dn_B = "b.aol.com", then $dnMatch(dn_A, dn_B) = true$. If dn_A = "a.yahoo.de" and dn_B = "a.yahoo.com", then $dnMatch(dn_A, dn_B) = false$. In addition to the use of *Split-hash* and *dnMatch*, a node may send a *lookupDN* message to discover other sibling nodes under the same sub-domain. The *lookupDN* message is flooding over the overlay network. The flooding may traverse a number of randomly selected neighbours before the node finds its siblings. By DDNO, a node tries to connect to *k*/2 nodes among the sibling nodes found by *lookupDN* and picks up other *k*/2 nodes at random. Similar to the SL algorithm and BinSL algorithm, the DDNO algorithm makes well-connected clusters of nodes that are topologically close to each other without jeopardizing network connectivity. Moreover, DDNO only uses the DNS system in selecting peers. Therefore, neither the $n \times n$ IP-latency matrix nor additional landmarks are required in DDNO. This leads to good scalability but incurs a little performance degradation compared to SL and BinSL algorithms. Another disadvantage of the DDNO algorithm is the additional traffic introduced by the flooding of *lookupDN* messages, although the authors declared that the lookups in a completely decentralized fashion will not actually impose large communication overhead even in larger topologies.

In [2], the authors propose the location-aware topology matching (LTM) algorithm based on the use of timestamps. A message called TTL2-detector is used in LTM to record timestamps of nodes for measuring latencies between nodes. These measured latencies will help the construction of a better overlay network, which could topologically match the physical underlying network. It is assumed that the clocks in all nodes are synchronized by NTP (Network Time Protocol) [30]. When a TTL2-detector message traverses a node, the node will record its timestamp in the message such that the latencies among these traversed nodes can be calculated. Thus, the topology of the overlay network is constructed by pruning the connections with a large cost in terms of latency and bandwidth. Indeed, it is confirmed that LTM achieves the same performance improvement as Gnutella-like overlay networks. However, it is not applicable to utilize LTM in some P2P overlay networks, like BT-like applications, which don't use flooding query messages. In addition, connection pruning used by LTM may jeopardize network connectivity and probably result in a shattered topology.

4. THE PROPOSED SCHEME

P2P protocols on the Internet are built over the IP layer. That is, P2P protocol messages are carried by IP packets. The proposed scheme is developed based on the inherent supports in the IP protocol to gather topological information of the P2P overlay network. The proposed scheme enables the *Timestamp* option of the IP header to record the current time of each router within the path from a peer to another one. Thus, we can evaluate the latency of data transmission from those timestamp records. Another approach of the proposed scheme is the use of the *Record Route* option of the IP header to record the routers which a packet traverses, which can be used to evaluate the network distance between peers. The proposed scheme is described in three aspects.

Selection of IP Options

***Timestamp* (TS)**: IP supports the *Timestamp* option to allow routers to record their current times in IP packets. For a proper use of timestamps, the clocks in all routers should be synchronized using time synchronization techniques [30] in an acceptable accuracy. If the timestamp option is present in the IP header of a packet, any router forwarding the packet will put its IP address and current timestamp in the IP header. For example, if there are n routers between two peers, timestamp option of the IP header will be ($Router_1$, $Timestamp_1$, $Router_2$, $Timestamp_2$, …, $Router_n$, $Timestamp_r$). Note that the length of the option in the IP header should not be more than 40 bytes. If a packet traverse many routers, it is possible that there is no room to allow all timestamps to be recorded. For saving spaces, recording only timestamps in the header can be adopted. In this case, the timestamp option area will be ($Timestamp_1$, $Timestamp_2$, …, $Timestamp_t$), which allows more timestamps included. Each timestamp is 32-bit timestamp string in milliseconds. Thus, at most 10 timestamps can be contained in the IP header option. The overflow flag field in the timestamp option is used to record the number of the routers which fail to add timestamps into the option due to the lack of space.

***Record Route* (RR)**: IP supports the *Record Route* option to allow routers to record their IP addresses in IP packets. If the *Record Route* option is present in the IP header of a packet, any router forwarding the packet will put its IP address in the IP header. The *Record Route* option will looks like ($Router_1$, $Router_2$, …, $Router_r$). The space limitation problem also exists in the use of the *Record Route* option.

Selection of P2P Messages to Enable IP Options

Both TS and RR scheme can be realized in all messages of a P2P protocol. In fact, it is not necessary to implement the schemes in each P2P message. To minimize communication and processing overheads, the proposed approach enables the IP options only during the very initial message exchange between peers before file downloading. In the BitTorrent protocol, before selecting peers for downloading files, a peer has to handshake with other peers in the peer-list given by a tracker. These two IP options can be enabled only in the handshaking packets because the handshaking is the initial and mandatory message exchange between peers. Let p denotes a peer which wants to download a file available in a BitTorrent P2P network N. Suppose that tracker will provide p with a list of participating peers, $L = \{p_1, p_2, \ldots, p_n\}$, where n is the maximum number of peers that a tracker will provide in the list. The default value of n is 50 by the BitTorrent specification. A peer may request a longer list of peers by specifying the *NumWant* parameter in the request message to the tracker. If there are fewer peers participating in the sharing of a specific file, then a shorter list will be provided. By default, the tracker will randomly select 50 peers to fill into the peer-list and send the peer-list to peer p. Upon receiving the peer-list, peer p then sends handshake messages to all the peers in list L to request connections. Those recipients will send responses to peer p. By the propose scheme, the TS or RR option is enabled in those recipients' responses. Therefore, after the handshaking procedure, peer p can acquire the latency or the number of hops between peer p and each peer in list L. For

Gnutella protocol, the TS or RR option will be enabled in Response messages from peers are. The rationale of enabling IP options in the Response messages is that it is an initial message before file downloading and is the only one message replied from other peers.

Selection of Peers

After the initial message exchanges between peers, the peer *p* will obtain a set of estimated latency or path length data for all the other peers. Then, peer *p* can select a certain number of peers with lower latency or shorter path length for downloading files concurrently. For a widespread distribution of files and better network connectivity, it is not recommended that only the near peers be selected for file sharing. The idea of our approach is similar to the SL algorithm. Suppose the number of peers to be selected is *k*. Given a list of *n* peers with $n \geq k$, peer *p* will select the first *k/2* peers with lower latencies or shorter path length and randomly select *k*/2 peers from the other *n* - *k*/2 peers. Recall that more peers randomly picked in file sharing will increase connectivity, but may lead to worse downloading performance. On the other hand, more near peers involved in downloading file pieces may achieve better performance. However, the connectivity of the overlay network will decrease. As shown in [22], the selection of *k/2* random peers in file sharing is a good trade-off between P2P communication performance and network connectivity.

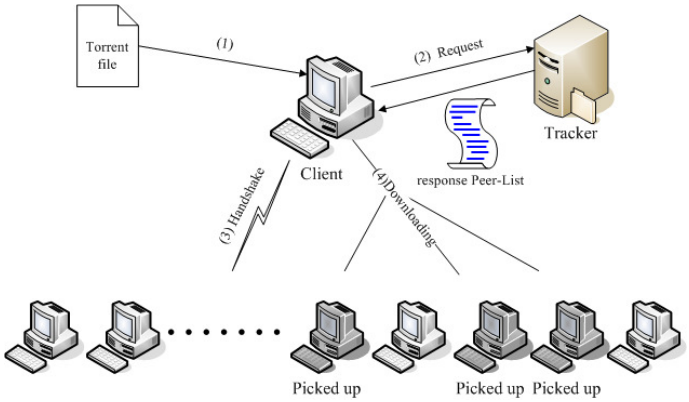

Figure 5. Procedures of the proposed schemes

Figure 5 depicts the message communications of the proposed scheme. In summary, the proposed scheme is performed in four steps.

1. A client peer gets the torrent file for a specific file on the web..
2. The client peer sends a request to a tracker to ask for a peer list for downloading the file of interest. By setting a bigger value of the "NumWant" parameter, the client gets a longer peer list than the original BT system does.
3. The client peer sends a Handshake message to each peer specified in the given peer list. Each peer responds to the handshaking by sending a response in an IP packet with *Record Route* or *Timestamp* option enabled.
4. The client picks up the first *k*/2 peers with the shorter latencies or the smaller number of hops measured by the TS or RR scheme. The client peer further picks up another *k/2* peers randomly from the remaining *n* - *k*/2 peers. Those *k* selected peers form a new peer list. The client peer starts to download file pieces from the peers in the new peer list.

One may argue that the latency evaluated via only one handshaking message is not accurate as the actual time spent in file downloading. The evaluated latency is still a valuable assessment for peer selection. In fact, the construction of an optimal overlay network is known to be NP-complete [30]. The proposed scheme can be regarded as a heuristic approach for better

performance than current BitTorrent-based P2P systems. In addition, if we keep measuring the latency in other messages, e.g. keep-alive and HAVE messages, considerable traffic will be introduced due to the extra 40 bytes needed in each IP packet. The traffic overhead is also proportional to the number of participating peers.

In practice, it is difficult to get the latency metric via the *Timestamp* option of the IP header. The prerequisite for proper use of timestamps is that the clocks of all the routers should be synchronized. The requirement is hard to be satisfied in the public Internet, which is constructed by a lot of networks under different administrative domains. Thus, we prefer the use of the *Record Route* option of the IP header for the selection of peers. By counting the entries in the *Record Route* data area, we can determine the number of hops for each route. A route with a smaller hop number is considered as a better one for file sharing.

5. SIMULATION

5.1. Performance Metrics

A study in [19] presents the importance of scalability and efficiency in P2P file sharing systems. These issues receive more attention recently because of the increased Internet bandwidth consumption by P2P systems. In fact, BitTorrent provides a built-in incentive unchoking algorithm to encourage users to share files more efficiently. Each peer prefers uploading to the peers with a higher downloading rate, since it usually implies more available bandwidth and better downloading performance. However, it doesn't imply a better usage of the overall Internet traffic. To evaluate the effectiveness of the Internet traffic usage, we introduce a new performance metric *Overall Traffic* based on the aggregated link costs of downloading file pieces. The overall traffic consumption of P2P communications becomes one of the important parameters concerned by network administrators. Network administrators usually use bandwidth shaping tools to limit the overall bandwidth usage of P2P systems. This still allows proper operations of P2P services but limits their scalability [10]. Therefore, a better traffic usage is crucial for the development of contemporary P2P systems. The overall traffic metric for use in our experiments is defined as follows.

$$Overall\ Traffic = \left(\sum_{i=0}^{p}\sum_{j=0}^{k} PieceLink[i][j]\right) \times 256\text{KB} \qquad (1)$$

$$PieceLink[i][j] \neq \infty$$

where *PieceLink* is a $p*k$ traffic matrix, assuming that there are p file pieces transferred within a network with k links. $PieceLink[i][j]$ indicates the consumed traffic in link j for the transmission of file piece i. In other words, the *Overall Traffic* metric represents the total traffic volumes for downloading a file with p pieces within a network with k links. We recommend that *Overall Traffic* metric be minimized for better performance of BitTorrent-based P2P systems.

In practice, there are a number of downloading events occurring simultaneously, A piece will become available in a peer after the peer finishes downloading of the piece. Therefore, a downloader can soon become an uploader. As the downloading events proceed, a peer can choose more nearer peers to download file pieces. Therefore, the time required to download a file will be related to the number of downloading events. Therefore, we use the average downloading time of all the downloading events to evaluate the downloading performance of the P2P system. In a downloading event, a user may give up to download a file if the downloading has taken a long time. Therefore, only successful downloading events are involved in the following performance evaluation. Assumes the number of successful downloading events is e, and the downloading time of the i-th event is dt_i, $0 \leq i < e$. The average downloading time DT_{avg} is defined as follows.

$$DT_{avg} = (dt_0 + dt_1 + \ldots + dt_{e-1})/e \qquad (2)$$

In addition, we take into account the average number of links required to download a file piece. It is an important metric in the evaluation of the proposed *Record Route* scheme. The average number of links, called $Link_{avg}$, is defined as follows.

$$Link_{avg} = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} Link[i][j]}{n} \quad (3)$$

where *Link* is an $n \times m$ matrix, assuming that there are *n* pieces transferred over a network with *m* links. *Links*[*i*][*j*] = 1 if link *j* is within the route of transmitting the file piece *i.*

Table 2. Configuration of the simulation.

Parameter	Value
delay / bandwidth transit-transit: transit-stub: within stub:	 5ms / 1000Mbps 10ms / 100Mbps 30ms / 10Mbps
number of peer	300
number of agents per peer	10
DFN-Like topology: number of nodes number of edges	 420 2100
number of nodes (overlay network)	300
Number of peers in a peer list	100 (RR, TS), 50 (BT)
Number of peers selected	50

5.2 Simulation Configuration

In the simulation, we use the Georgia Tech Internetwork Topology Model (GT-ITM) [32] to generate network topologies. There are two topology models generated by GT-ITM: Waxman model [33] and Transit-Stub (TS) model [34]. The Waxman model describes the Internet as a graph in which the nodes in the network are placed at random points in a two-dimensional grid and the links are added to the network by connecting any pair of nodes. This model has a probability function used to decide whether a link should exist. This probability function involves that how far apart the two nodes are and how many links are expected to exist in the whole network [35]. The Transit-Stub (TS) model generates a network topology from the top level of a hierarchy, i.e. Transit domain or WAN, and proceeds downward to the lowest level of the hierarchy, i.e. Stub domain or LAN. Each node in the generated topology is labelled with a string of identifiers: an identifier indicating whether it is a transit of Stub-node, a global identifier of the domain to which it belongs, and a domain-local identifier. Each edge of the generated topology is associated with a routing policy weight, which can be used to find routes that follow the standard domain-based routing outlined earlier. In the simulation, we adopt the TS model as our topology model because the Internet is constructed hierarchically. In addition, the topology is generated based on the parameters specified in [36] to simulate a real ISP network, DFN G-Win (German research network). As indicated in [36], the topology generated by the parameters achieves a high degree of similarity: 0.966. In addition to the topology generator, we conduct experiments using the General Peer-to-Peer Simulator (GPS), which is a message-level and event driven P2P simulation framework to model P2P protocols, including the BitTorrent protocol. In the simulation on GPS, we assume that there are some peers randomly located in the nodes of the generated topology. A node is also randomly selected as the tracker. Initially, a document of size 500,000 KB is stored in some peer. The performance of

the P2P system is evaluated regarding different number of downloading events for the document. Because of the use of an unchoking algorithm in the simulation, the simulator needs a way to calculate the available bandwidth, as defined follows.

$$BW = \frac{MSS}{RTT}\frac{C}{\sqrt{p}} \qquad (4)$$

where BW is the available bandwidth, MSS is the maximum segment size of TCP, RTT is the round trip time, C is a constant for TCP flavor, and p is the estimated packet loss rate. In our simulation, we set MSS=536, RTT=1.7, C=1.22, and p=0.001.Three schemes are implemented in the simulation: Route Record (RR), Timestamp (TS), and the original BitTorrent scheme (BT). Table 2 lists the settings of the simulation.

5.3 Simulation Results

In the following, the simulation results of average downloading time, overall traffic, and average number of links of three schemes are presented, with respect to four different numbers of downloading events: 50, 85, 113, and 149. These four numbers are obtained by counting only the successful events of all the downloading events in the simulation.

5.3.1. Downloading Time

Figure 6 shows the average downloading times of three schemes in four cases of downloading events. It can be seen that the proposed TS and RR schemes achieve downloading times slightly smaller than the BT scheme in the simulation of 50 downloading events. In fact, in the case of 50 downloading events, all the three schemes choose the same peers to download file pieces. The better performance achieved is due to the different orders of peers involved in file sharing. When the TS or RR scheme is used, the client will first choose closer peers to download file pieces, although in all the three schemes downloading connections will be rotated among all participating peers if the number of downloading events is below 50. Therefore, the proposed schemes can achieve smaller downloading times slightly even if the number of all participating peers is below 50. As the number of downloading events increases, more peers become uploaders and the TS and RR schemes take effect in the selection of peers. By TS and RR schemes, more nearer peers will be selected in file sharing than the BT scheme. Therefore, compared to the BT scheme, both TS and RR schemes can achieve smaller downloading times. In summary, the simulation results show that the average downloading times of the proposed TS and RR schemes are smaller than the BT scheme in all cases of downloading events. The improvements of downloading times over the BT scheme are about 3~21% and 2~21% respectively.

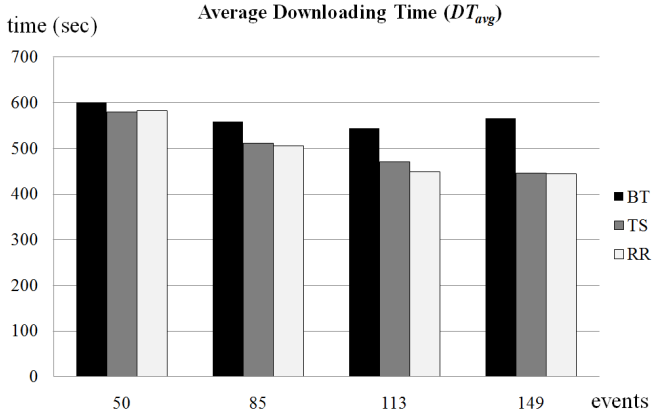

Figure 6. Average downloading time

5.3.2. Overall Traffic

The overall traffic evaluation of our simulations is to investigate the total network traffic consumed by the proposed schemes. The simulation results of overall traffic are shown in Figure 7. The improvement of overall traffic achieved by our schemes is not significant in the case of 50 downloading events. This is because all peers are selected in file sharing. When the number of downloading events increases, the overall traffic consumed can be reduced by the proposed schemes. It is shown that our schemes reduce 7%~18% of the overall traffic compared to the BT scheme in the cases of 85, 113, and 149 downloading events. Furthermore, comparing the two proposed schemes, we find that the RR scheme can reduce more overall traffic than the TS scheme with respect to the same number of downloading events. As expected, the RR scheme provides an accurate estimation of number of links used in file sharing. Therefore, the client can exactly select the peers with less overall traffic usage.

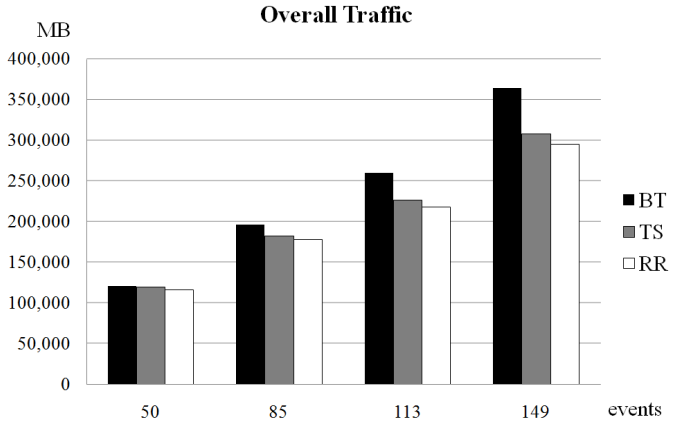

Figure 7. Overall traffic

5.3.3. Average Number of Links

The last simulation result is the average number of links required to transmit a file piece. The simulation results are shown in Figure 8. In average, the number of links by the proposed schemes is about 7% ~ 18% shorter than the BT scheme for more than 50 downloading events. In addition, the RR scheme slightly outperforms the TS scheme in this simulation. In fact, the evaluation of average number of links is very similar to the overall traffic evaluation. This measuring metric is to reveal the degree of topology mismatch between the P2P overlay network and physical network. A smaller number of links between peers implies a better topology consistency. Thus, this explains why the performance of P2P systems can be improved by the proposed schemes.

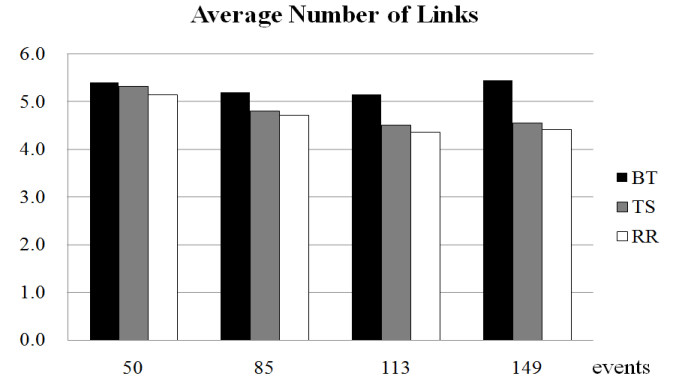

Figure 8. Average number of links

6. CONCLUSIONS

This paper has presented a simple and scalable approach to inferring network proximity information. We apply two schemes to the construction of topologically-aware overlay networks to achieve better usage of overall traffic. Simulations have shown that the performance improvement can be achieved by our schemes. The performance metrics of our research are based on three parameters, overall traffic, average downloading time, and average number of links. Our simulations indicated that our schemes are attractive implementations for topologically-aware overlay construction as they reduce about 7% ~ 18% of overall traffic and downloading time. This paper takes the BitTorrent protocol as an unstructured P2P example in our schemes and simulations. Conceptually, our approaches can be applied in a variety of P2P systems. In practice, there is a need for a more generalized scheme applicable for different P2P frameworks, including centralized, decentralized and semi-centralized P2P systems. In the future, we will further investigate the developement of the proposed schemes in other P2P systems, such as Gnutella, Foxy, and eMule.

REFERENCES

[1] Top applications (bytes) for subinterface: SD-NAP traffic, 2006. http://www.caida.org./analysis/workload/byapplication/sdnap.

[2] Yumhao Liu, Xiaomei Liu, Li Xiao, Lionel M. Ni, Xizodong Zhang, "Location-Aware Topology Matching in P2P Systems," in *Proceedings of IEEE INFOCOM 2004*, 2004.

[3] S. Sen and J. Wang, "Availability and locality measurements of peer-to-peer file systems," *in Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2002.

[4] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," *in Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, 2002.

[5] KaZaA, http://www.kazaa.com/.

[6] Gnutella, http://gnutella.wego.com/.

[7] DirectConnect, http://www.neo-modus.com/.

[8] Kiarash Mizanian, Mehdi Vasef and Morteza Analoui, "Bandwidth Modeling and Estimation in Peer to Peer Networks," *International Journal of Computer Networks & Communications* (*IJCNC*), Volume 2, Number 3, May 2010, pp. 65-83.

[9] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing*, Vol. 6, No.1, 2002, pp.50-64.

[10] Ritter, "Why Gnutella can't scale. No, really", http://www.darkridge.com/~jpr5/doc/gnutella.html.

[11] R. M. Stoica, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," *in Proceedings of ACM SIGCOMM'01*, 2001.

[12] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *in Proceedings of International Conference on Distributed Systems Platforms*, 2001.

[13] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing," *Technical Report UCB//CSD-01-1141*, U.C.Berkely, 2001.

[14] R. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *in Proceedings of ACM SIGCOMM'01*, 2001.

[15] L. Mekouar, Y. Iraqi, R. Boutaba, "Peer-to-peer's most wanted: Malicious peers", *The International Journal of Computer and Telecommunications Networking*, Vol. 50 , No. 4 , March 2006.

[16] B. Yang and H Gracia-Molina, "Efficient search in peer-to-peer networks," *in Proceedings of International Conference on Distributed Computing Systems* (*ICDCS'02*), 2002.

[17] BitTorrent official website, http://www.BitTorrent.com/, 2005.

[18] B. Cohen. Incentives to Build Robustness in BitTorrent, May 2003. http://bitconjurer.org/BitTorrent/BitTorrentecon.pdf.

[19] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," *in Proceedings of ACM SIGCOMM'04*, 2004.

[20] J. Postel, "Internet Protocol," RFC 791, Sep. 1981.

[21] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, "Topologically-aware Overlay Construction and Server Selection," *INFOCOM 2002. in Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies,* Vol. 3, 2002.

[22] D. Zeinalipour-Yazti, V. Kalogeraki, "Structuring topologically aware overlay networks using domain names," *The International Journal of Computer and Telecommunications Networking archive,* Vol. 50 , No. 16, Nov. 2006.

[23] P. Sinha, D. Raz, N. Choudhuri, "Estimation of network distances using off-line measurements," *Computer Communications*, Vol. 29, 2006, pp. 3295-3305.

[24] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," *in Proceedings of SIGCOMM '00*, Stockholm, Sweden, Aug. 2000.

[25] C. Davis, P. Vixie, T. Goodwin, I. Dickinson, "A Means for Expressing Location Information in the Domain Name System," RFC 1876, Jan. 1996.

[26] C. Davis, H. Rose, "DNS LOC: geo-enabling the domain name system," http://www.ckdhr.com/dns-loc/.

[27] V. Batagelj, A.Mrvar. "PAJEK-Program for large network analysis," *Connections* 21, 1998.

[28] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker, "Making Gnutella-like P2P system scalable," *in Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications,* Karlsruhe, Germany, 2003, pp.407-418

[29] P. Mockapetris. "Domain names – Implementation and specification," *RFC 1035, Network Working Group*, November 1987.

[30] David L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," RFC 1305, March 1992.

[31] M. R. Garey, D. S. Johnson, Computers and Intractability. "Guide to the Theory of NP-Completeness," W. H. Freeman San Francisco, CA, 1979.

[32] GT-ITM: Georgia Tech Internetwork Topology Models, http://www.cc.gatech.edu/projects/gtitm/.

[33] B. M. Waxman, "Routing of Multipoint Connections," *IEEE JSAC*, col. No. 9, 1988, pp. 1617-22.

[34] D. Knuth, "The Stanford GraphBase: A Platform for Combinatorial Computing," Reading, MA: Addison-Wesley, 1994.

[35] Ken Calvert, Matt Doar and Ellen W. Zegura. ,"Modeling Internet Topology," *IEEE Communications Magazine*, June 1997.

[36] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "Generating Realistic ISP-Level Network Topology", *IEEE Communications Letters*, Vol. 7, No. 7, July 2003.