A Proposal of P2P Content Delivery System for Supporting Streaming Applications

Takanori Kashiwagi*, Jun Sawamoto*, Hiroyuki Sato*, Norihisa Segawa*, Eiji Sugino*, Yuji Wada**

*Graduate School of Software and Information Science, Iwate Prefectural University
**Department of Information Environment, Tokyo Denki University
*{sawamoto, sato_h}@iwate-pu.ac.jp, **yujiwada@sie.dendai.ac.jp

Abstract - Streaming large files such as video and audio contents from the internet has become an increasingly common practice with users and content providers. Content delivery presents serious challenge for content providers, with the increased cost of hosting and transmitting large video files, the existing client server system is experiencing problems. The high server load incurred by the client model is costing hosts considerable resources. Peer to Peer (P2P) technology alleviates some of these problems by distributing transfer work among multiple hosts (peers). P2P works by sending and receiving data directly with other peers that are participating in the network. It distributes resources and load across the network. This can solve the problem of the client server system resource overload. The purpose of this research is to propose a method which is suitable for streaming using P2P and solve the problem of client server system's resource overload. We aim to realize stable video streaming, low latency playback, and reduction of the number of breaks due to buffering protocol.

Keywords: Content delivery, Streaming, Peer to Peer network, BitTorrent, BiToS.

1 INTRODUCTION

[1]The video and audio content delivery service using the internet, such as YouTube [1] and NicoNico Douga [2], has become an increasingly common practice, and it is capturing the attention from broad directions, such as political use and commercial use, etc. Moreover, by the development of broadband service and improvement of terminal performance of individual use, it is expected that the video and audio content as a medium for disseminating information continues to grow. In the prediction and investigation of Cisco [3], it is expected that two-thirds of the world's mobile data traffic will be video by 2017. Mobile video will increase 16-fold between 2012 and 2017, accounting for over 66 percent of total mobile data traffic by the end of 2017. As streaming large files such as video and audio content from the internet has become an increasingly common practice with users and content providers, the content delivery presents serious challenge for content providers, with the increased cost of hosting and transmitting large video files, the existing client server system is experiencing problems. The high server load incurred by the server-client model is costing hosts considerable resources.

Peer to Peer (P2P) technology alleviates some of these problems by distributing transfer work among multiple hosts (peers). BitTorrent [4] is one of the most popular P2P protocols. File transfer operates by splitting the file into many pieces. Peers transfer the pieces out of order in a distributed fashion then re-assemble the original file. The order of the pieces transferred is determined by the RarestFirst algorithm [5, 6]. However, it is bad for streaming because pieces are transferred out of order and it is hard to predict the next piece. BiToS (BitTorrent Streaming) [7] was proposed to solve the streaming P2P problems of BitTorrent. This allowed somewhat smoother playback, but there were still delays or pauses (breaks). And some new methods to shorten breaks' time and reduce the number of times of breaks are called for.

We propose a method which is suitable for streaming using P2P. The emphasis must be placed on reduction of the number of breaks in playback. To this end, we must do something different if there is a gap in download pieces between current playback position and the next available piece. Improved peer and piece selection methods, such as special priority for pieces near playback position may hopefully alleviate the problems with BiToS and RarestFirst algorithm. Specifically, if the piece closest to the playback position is not yet downloaded then the proposed method will set an emergency priority. Within the high priority group we must request missing pieces from the peer with the fastest connection. In order to verify the proposed method's effectiveness when compared to the established methods of RarestFirst and BiToS, we performed simulations and experiments.

The rest of this paper is organized as follows: In section 2, we describe detailed algorithm of BitTorrent and BiToS. In section 3, we present our proposed solution for better peer and piece selection. In section 4, details of the implementation on software simulator is described. In section 5, we report experimental evaluation of our proposed method. Finally, the paper is concluded in section 6.

2 BITTORRENT AND BITOS

BitTorrent is one of the most popular P2P protocols. Holding, sending, and receiving of all content are performed by only the peers. The tracker manages information about peers in a swarm; it coordinates initial connections and

[1]This work was supported by JSPS KAKENHI Grant Number 24500122.

keeps a table of connected hosts and the download/upload statistics of each peer (Fig.1).

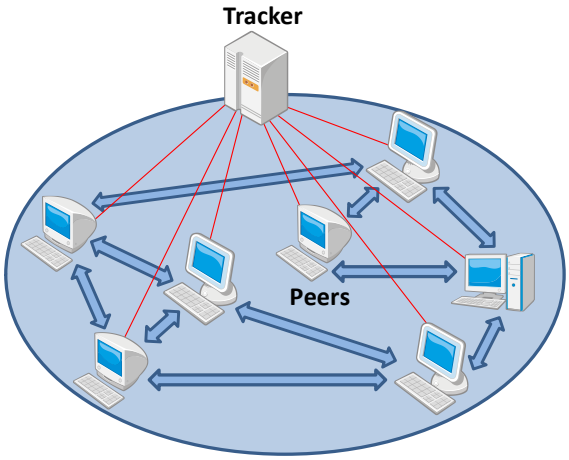


Figure 1: Network configuration of BitTorrent.

As shown in Fig.2, BitTorrent uses swarming techniques in which the torrent file (the content that is distributed), is split in pieces. A user who wants to upload a file first acts as a seed and distributes content information through BitTorrent nodes. Peers (leecher) can simultaneously download pieces from other peers. While the peer is downloading pieces of the file, it uploads the pieces that it has already acquired to its peers. Each time the peer has a new piece, it advertises this information to its peer set (the peers that the peer is connect to).

Peers transfer the pieces out of order in a distributed fashion then re-assemble the original file. This distributed method is suitable for large-capacity content delivery.

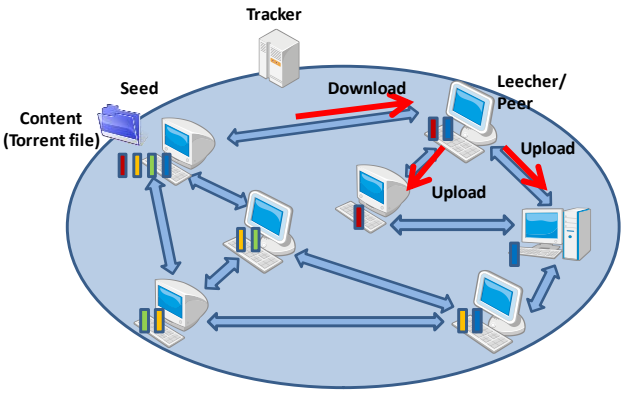


Figure 2: File transfer operates by splitting the file into many pieces

The order of the pieces transferred is determined by the RarestFirst algorithm. This algorithm tells peers to send the least common pieces amongst the swarm first, causing convergence faster. RarestFirst transfer makes P2P very efficient when compared to the random out of order method. However, it is bad for streaming because pieces are transferred out of order and it is hard to predict the next piece. Streaming requires in-order transfer for smooth playback. The method proposed in this paper aims to provide more predictable transfer to allow for smooth playback.

BiToS was a previous attempt to solve the streaming P2P problems (Fig.3). It was a research to reduce the number of breaks when streaming using BitTorrent. The BiToS method changed from RarestFirst so that pieces near deadline mark have higher priority than later pieces. This allowed somewhat smoother playback, but there were still pauses. BiToS method works by assigning a priority to two groups of pieces. If the probability of selecting a piece from the high priority group is "*p*" then low priority group probability is "*1-p*". The parameter "*p*" represents the balance between the immediate need for a piece and the future need. Within each priority group we simply use RarestFirst method.

Currently downloading pieces in high priority group and low priority group are moved to the group of received pieces after they are downloaded. If a piece cannot meet its playback deadline, then it will not be asked to be downloaded (or its download can be aborted) and will be marked Missed. A peer at any given time can have at maximum a fixed number of currently downloading pieces. The number of pieces (cardinality) of the higher priority group remains fixed. Using BiToS, we receive pieces closer to the playback position sooner. This is more suitable for content delivery than pure RarestFirst method.

However within each group the RarestFirst method is still used, so there may be breaks if the priority group has not rare pieces close to the playback position. This means pieces are still sent out of order within each priority group. This causes gaps in playback when the playback position reaches a missed piece.

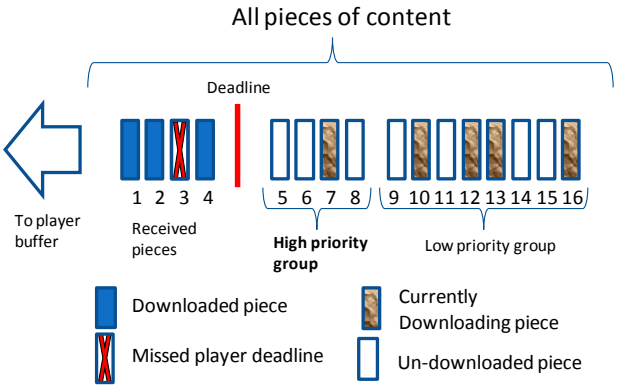


Figure 3: Outline of BiToS piece selection method.

3 PROPOSED SOLUTION

To propose a method which is suitable for streaming using P2P, emphasis must be placed on the reduction of the number of breaks in playback. To this end, we must do something different if there is a gap in download pieces between our deadline position and the next available piece. Here, the deadline is the time limit after that, the received piece is not useful and will be discarded.

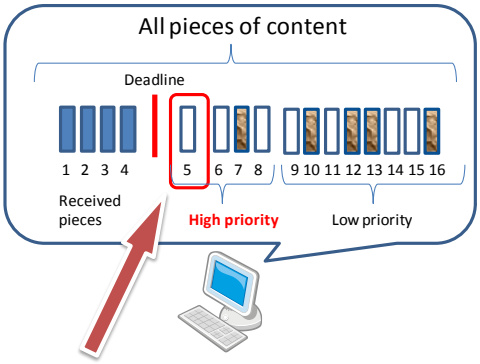


Figure 4: Introduction of emergency priority.

Improved peer and piece selection methods, such as special priority for pieces near deadline position may hopefully alleviate the problems with BiToS and RarestFirst. Specifically, if the piece closest to the deadline position is not yet downloaded then the proposed method will set an emergency priority (Fig. 4). Within the high priority group we must request emergent pieces from the peer with the fastest connection (Fig. 5).

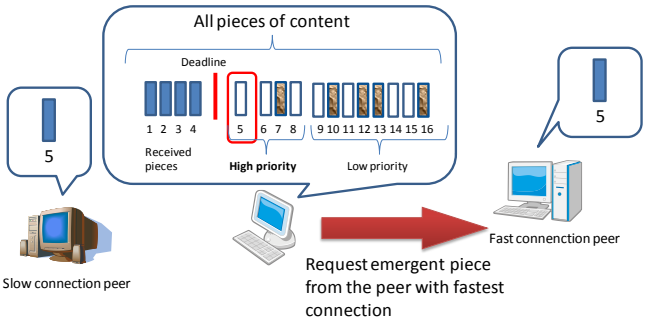


Figure 5: Request emergent pieces from the peer with the fastest connection.

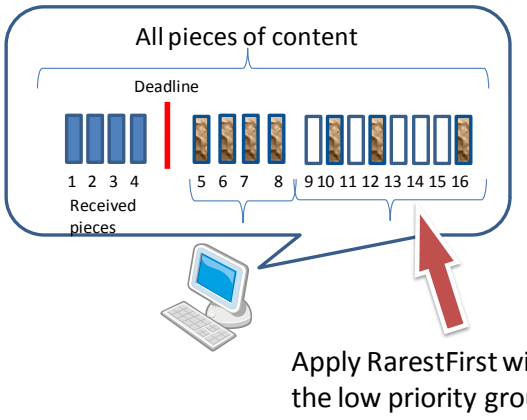


Figure 6: Enough buffered content then download pieces from a lower priority group.

If there is enough buffered content then the new method may download pieces from a lower priority group using simple RarestFirst (Fig. 6). Thus it is still possible to contribute to the distribution of rare pieces on low priority groups and improve convergence speed.

The proposed method solves the problem of BiToS where pieces close to playback position are not always chosen. This leads to a more stable delivery and smooth playback.

4 IMPLEMENTATION ON A SOFTWARE SIMULATOR

In order to verify the proposed method's effectiveness when compared to the established methods of RarestFirst and BiToS, it is necessary to perform simulations and experiments. One such proposed experiment is to provide a peer that implements each method on a software simulator. We used General Purpose Simulator for P2P network (GPS) [8] which is capable of simulating BitTorrent algorithm.

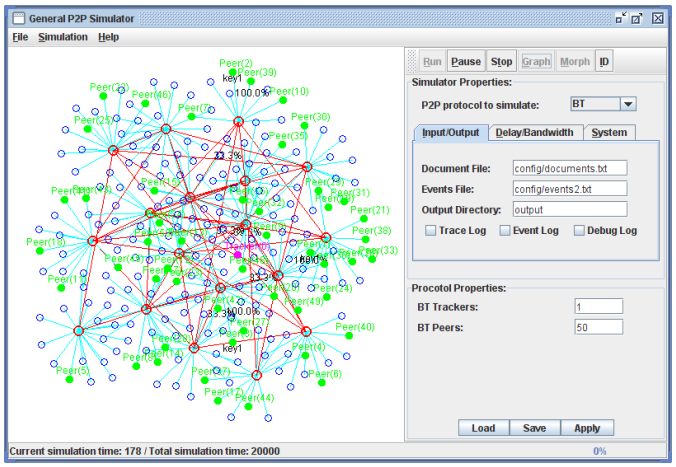


Figure 7: Display image of the simulation by General Purpose Simulator for P2P network.

As for the software structure of GPS, various search protocols such as Chord [9], CAN [10], etc. are located on top of the physical network layer at the bottom of the structure. The layer of P2P algorithms come on the search layer. Some Hybrid P2P algorithms including BitTorrent exist in the same layer as the search layer, because they don't use provided general search protocols like Chord etc. but they mostly implement original search protocols using the server systems.

The methods of previous works and our proposed method are implemented on top of the P2P algorithms layer, and they can be switched according to the experimental situation. However, it is not possible to make peers who adopt different methods on the same network at present.

Moreover, in the operation of the various methods, since it is necessary to acquire the information of the playback position, and to measure the number of times of breaks and duration and frequency of breaks, which is the evaluation indices, we added virtual video player part on top of the P2P algorithm layer.

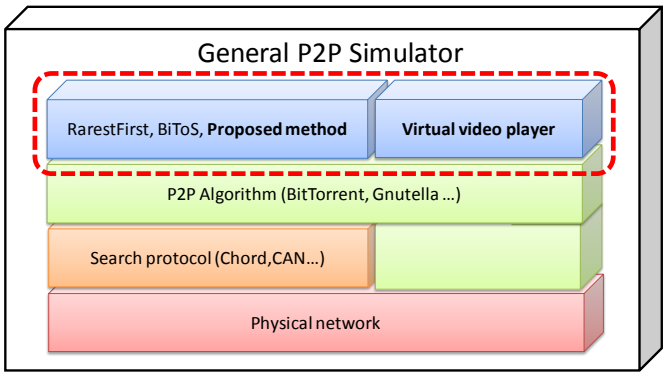


Figure 8: Software structure of GPS.

5 EXPERIMENTS

The peer and piece selection method proposed by this research, RarestFirst and BiTos are compared by measuring evaluation indices such as the number of breaks and the total duration of breaks under various video download conditions.

5.1 Outline of the Experiment

First, the peer who had all the pieces (original content) is generated on the simulator. Then a peer who does not have any piece participates one at a time to the network with certain interval and starts content downloading. Playback is started when the head piece of the content is downloaded at the peer. All the peers continue remaining in the network until the last peer completes the download. All the peers who participated to the network complete the download of whole content and finish the playback then the simulation stops.

The transmission speed of peers are classified into two types such as high speed and low speed, and randomly assigned to each peer. In the communication between low-speed peers, bandwidth is set to 5Mbps, between a high-speed peer and a low-speed peer 10Mbps, and between high-speed peers 15Mbps.

Simulations are iterated 10 times for each method respectively, and the results are compared on the average basis.

5.2 Contents and Parameters used for the Simulation

The details of parameters used for the simulation are shown in Table 1. The content sizes are two kinds, 128 MB and 256 MB.

The size per one piece, in consideration of the size length used widely when dividing a file by BitTorrent, is set as 1 MB. Even if the content size is the same, the playback time differs according to the content quality, high and low image quality. We experiment two cases of playback time, i.e., 0.5 seconds and 4 seconds per one piece, supposing two content qualities.

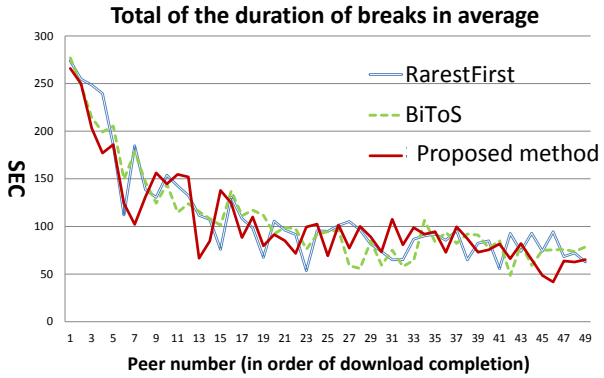
Table 1: Details of the contents and parameters used for the simulation

Content size (Mbyte)	128MB		256MB	
Size of a piece (Mbyte)	1MB			
Playback time per a piece (sec)	4	0.5	4	0.5
Number of peers	50			
Participating interval of new peers (sec)	60			
Ratio of the high priority group (%)	5			
Probability of selecting a piece from high priority group (%)	90			

5.3 Experimental Results and Evaluation

5.3.1 Content Size 128MB , 4 Seconds of Playback Time per One Piece

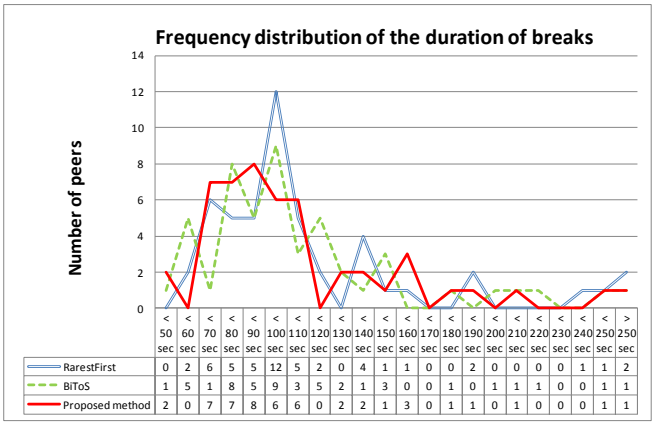
The experimental result in case of content size is 128 MB and the playback time per one piece is 4 seconds is shown here. Fig.9 is a graph of the total of the duration of breaks in average at each peer and total average of all peers during the playback by each method. The total duration of breaks at the peer which completed download earlier is large and decreases as the number of peers increases for all methods. This is because when few peers are in the network, the number of downloadable peers is small, but it increases as more peers participate to the network and the feature of P2P algorithm that a download speed rises using a communication line effectively as the number of peers increase is shown here. From the graph, significant difference is not seen as a whole by each method, but when the total average of all peers was taken for each method, it turned out that the total of the duration of breaks in average is the shortest in our proposed method (105.2) than RarestFirst (110.0) or BiToS (108.3).



Total average (seconds)		
RarestFirst	BiToS	Proposed method
110.0	108.3	105.2

Figure 9: Total of the duration of breaks in average at each peer and total average of all peers (Content size 128MB, 4 seconds of playback time per one piece).

Figure 10 shows the frequency distribution of the duration of breaks at each peer for each method. In our proposed method, many peers have shorter duration of breaks compared to other methods. For example, as shown in Table 2, 24 peers have duration of brakes less than 90 seconds in our method compared to 18 in RarestFirst and 20 in BiToS. Therefore it could be assumed that many peers have achieved shorter download time of the content as a whole.



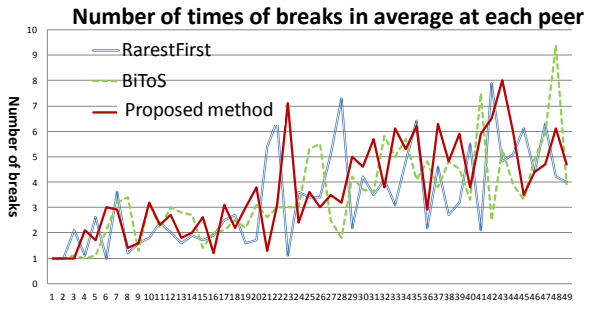
	< 50 sec	< 60 sec	< 70 sec	< 80 sec	< 90 sec	< 100 sec	< 110 sec	< 120 sec	< 130 sec	< 140 sec	< 150 sec	< 160 sec	< 170 sec	< 180 sec	< 190 sec	< 200 sec	< 210 sec	< 220 sec	< 230 sec	< 240 sec	< 250 sec	> 250 sec
RarestFirst	0	2	6	5	5	12	5	2	0	4	1	1	0	0	2	0	0	0	0	1	1	2
BiToS	1	5	1	8	5	9	3	5	2	1	3	0	0	1	0	1	1	1	0	0	1	1
Proposed method	2	0	7	7	8	6	6	0	2	2	1	3	0	1	1	0	1	0	0	0	1	1

Figure 10: Frequency distribution of the duration of breaks at each peer in average (Content size 128MB, 4 seconds of playback time per one piece).

Table 2: Accumulated number of peers of duration of brakes less than 90 and 110

	RarestFirst	BiToS	Proposed method
<90 sec	18	20	24
<110 sec	35	32	36

On the other hand, about the number of times of breaks, as shown in Fig.11 of number of times of breaks in average at each peer and total average, no method is stable and no significant difference is seen in average here.

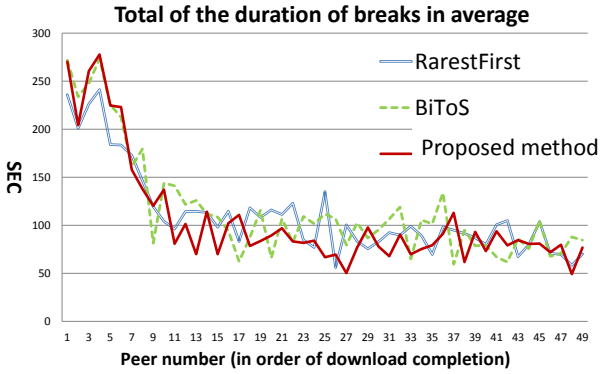


Total average (Times)		
RarestFirst	BiToS	Proposed method
3.3	3.4	3.7

Figure 11: Number of times of breaks in average at each peer and total average (Content size 128MB, 4 seconds of playback time per one piece).

5.3.2 Content Size 128MB , 0.5 Seconds of Playback Time per One Piece

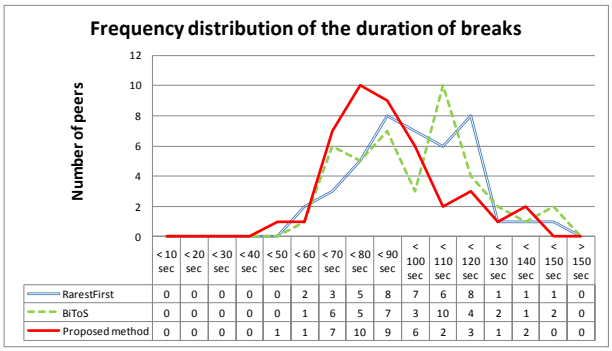
The experimental result in case of content size is 128 MB and the playback time per one piece is 0.5 seconds is shown here. Form the graph of Fig.12, the total duration of breaks in average at each peer and total average of all peers during the playback shows similar trend as the case of 4 seconds of playback time per one piece, and it turned out that the total of the duration of breaks is the shortest in average in our proposed method (106.3) than RarestFirst (110.9) or BiToS (116.2).



Total average of all peers (seconds)		
RarestFirst	BiToS	Proposed method
110.9	116.2	106.3

Figure 12: Total of the duration of breaks in average at each peer and total average of all peers (Content size 128MB, 0.5 seconds of playback time per one piece).

Fig.13 shows the frequency distribution of the duration of breaks at each peer for each method. In our proposed method, many peers have shorter duration of breaks compared to other methods. For example, 34 peers have duration of brakes less than 100 seconds in our method compared to 25 in RarestFirst and 22 in BiToS.

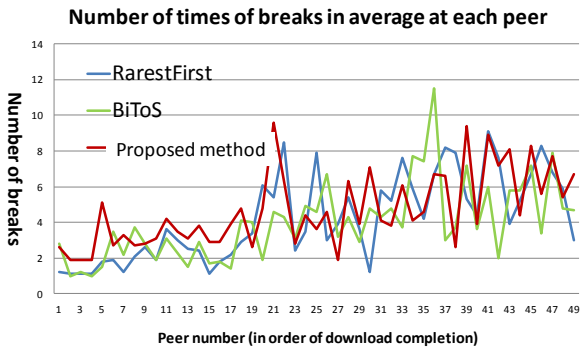


	< 10 sec	< 20 sec	< 30 sec	< 40 sec	< 50 sec	< 60 sec	< 70 sec	< 80 sec	< 90 sec	< 100 sec	< 110 sec	< 120 sec	< 130 sec	< 140 sec	< 150 sec	> 150 sec
RarestFirst	0	0	0	0	0	2	3	5	8	7	6	8	1	1	1	0
BiToS	0	0	0	0	0	1	6	5	7	3	10	4	2	1	2	0
Proposed method	0	0	0	0	1	1	7	10	9	6	2	3	1	2	0	0

Figure 13: Frequency distribution of the duration of breaks at each peer in average (Content size 128MB, 0.5 seconds of playback time per one piece).

On the other hand, about the number of times of breaks, , as shown in Fig.14 of number of times of breaks in average

at each peer and total average, no big difference is seen among methods just like the case of 4 seconds of playback time per one piece.

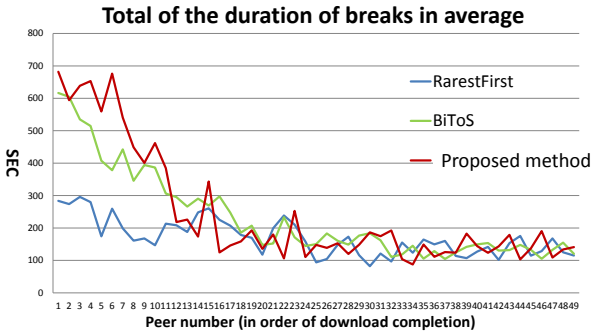


Total average (Times)		
RarestFirst	BiToS	Proposed method
4.2	4.0	4.7

Figure 14: Number of times of breaks in average at each peer and total average (Content size 128MB, 0.5 seconds of playback time per one piece).

5.3.3 Content size 256MB, 4 seconds of playback time per one piece

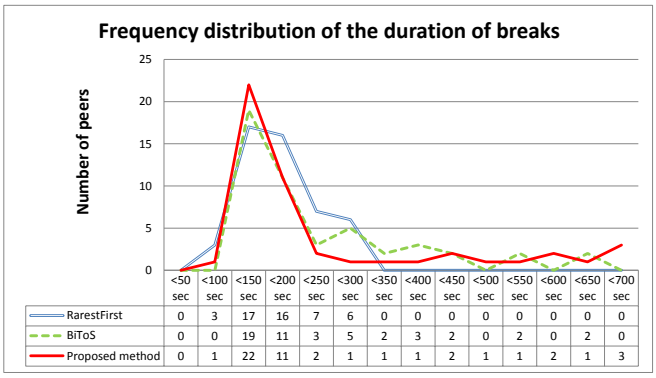
The experimental result in case of content size is 256 MB and the playback time per one piece is 4 seconds is discussed here. The proposed method has shown poor performance here and the total duration of breaks in average at each peer is the largest (Proposed metod:244.0, RarestFirst:169.8, BiToS:231.5) as shown in Fig.15.



Total average of all peers (seconds)		
RarestFirst	BiToS	Proposed method
169.8	231.5	244.0

Figure 15: Total of the duration of breaks in average at each peer and total average of all peers (Content size 256MB, 4 seconds of playback time per one piece).

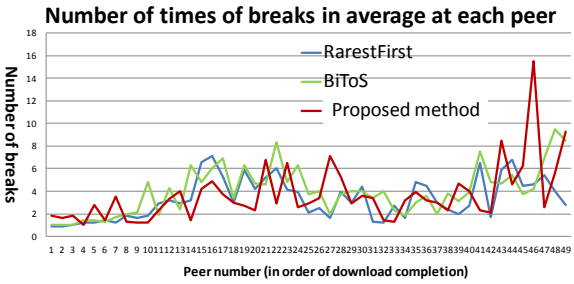
Figure 16 shows the frequency distribution of the duration of breaks at each peer for each method. For example, the number of peers of less than 150 seconds of duration of breaks is 23 in our method compared to 20 in RarestFirst and 19 in BiToS.



	<50 sec	<100 sec	<150 sec	<200 sec	<250 sec	<300 sec	<350 sec	<400 sec	<450 sec	<500 sec	<550 sec	<600 sec	<650 sec	<700 sec
RarestFirst	0	3	17	16	7	6	0	0	0	0	0	0	0	0
BiToS	0	0	19	11	3	5	2	3	2	0	2	0	2	0
Proposed method	0	1	22	11	2	1	1	1	2	1	1	2	1	3

Figure 16: Frequency distribution of the duration of breaks at each peer in average (Content size 256MB, 4 seconds of playback time per one piece).

As shown in Fig.17, the number of times of breaks in average at each peer and total average, no big difference is seen among methods (Proposed method: 3.7, RarestFirst: 3.3, BiToS: 4.0).



Total average (Time)		
RarestFirst	BiToS	Proposed method
3.3	4.0	3.7

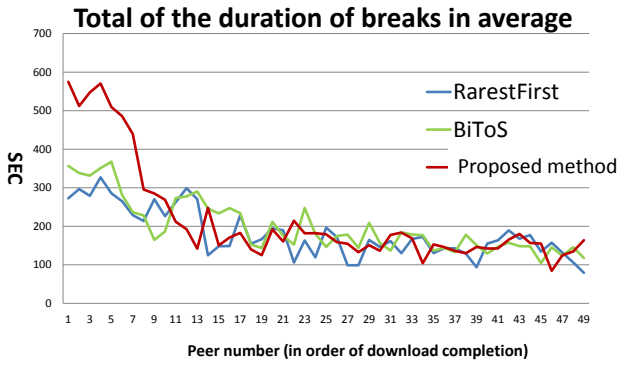
Figure 17: Number of times of breaks in average at each peer and total average (Content size 256MB, 4 seconds of playback time per one piece).

5.3.4 Content size 256MB, 0.5 seconds of playback time per one piece

The experimental result in case of content size is 256 MB and the playback time per one piece is 0.5 seconds is discussed here. Here also the proposed method performed poorly in terms of total duration of breaks in average as shown in Fig.18.

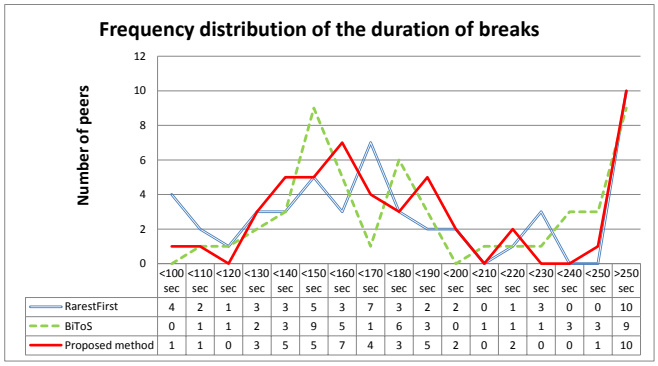
The frequency distribution of the duration of breaks shows the distribution is high in the area of 130-200 seconds and over 250 seconds area in all methods as shown in Fig.19.

As shown in Fig.20, the number of breaks in average is the smallest in our method (Proposed method: 4.0, RarestFirst: 4.1, BiToS: 4.5), but no significant difference is seen by methods here also.



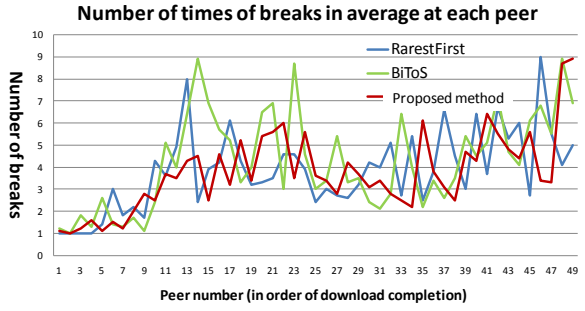
Total average of all peers (seconds)		
RarestFirst	BiToS	Proposed method
180.5	198.0	218.1

Figure 18: Total of the duration of breaks in average at each peer and total average of all peers (Content size 256MB, 0.5 seconds of playback time per one piece).



	<100 sec	<110 sec	<120 sec	<130 sec	<140 sec	<150 sec	<160 sec	<170 sec	<180 sec	<190 sec	<200 sec	<210 sec	<220 sec	<230 sec	<240 sec	<250 sec	>250 sec
RarestFirst	4	2	1	3	3	5	3	7	3	2	2	0	1	3	0	0	10
BiToS	0	1	1	2	3	9	5	1	6	3	0	1	1	1	3	3	9
Proposed method	1	1	0	3	5	5	7	4	3	5	2	0	2	0	0	1	10

Figure 19: Frequency distribution of the duration of breaks at each peer in average (Content size 256MB, 0.5 seconds of playback time per one piece).



Total average (Times)		
RarestFirst	BiToS	Proposed method
4.1	4.5	4.0

Figure 20: Number of times of breaks in average at each peer and total average (Content size 256MB, 0.5 seconds of playback time per one piece).

5.4 Consideration

In case of content size 128MB, in both cases of 4 and 0.5 seconds of playback time per one piece, the number of times of breaks is rather small in all peers and no significant difference was seen by each method. It is considered that since the communication with sufficient bandwidth is secured by any method because the size of the content is small enough for the environment with assumed number of peers and line speed. On the other hand, there is less number of times of breaks in case the playback time per one piece is 4 seconds rather than the case of 0.5 second. This indicates that long playback contents with low image quality have less frequent breaks. About the duration of breaks, in both cases of 4 and 0.5 seconds of playback time per one piece, the average duration of breaks is the smallest by our proposed method. In many peers, average duration of breaks distributes between 50 and 120 seconds. In case of 0.5, the duration came between 50 and 100 in most of peers by our method, and our proposed method performed better than other methods.

In case of content size is 256MB, in cases of 4 and 0.5 seconds of playback time per one piece, average number of times of breaks is smallest by RarestFirst and by our method respectively, but no significant difference is seen among methods. This is because the content size is rather large and pieces are too many for the assumed environment in this case. For the duration of breaks, in both cases of 4 and 0.5 seconds of playback time per one piece, the average duration of breaks is the largest by our method. And from the frequency distribution of the duration of breaks, distribution of short breaks is almost same by all methods, but breaks of long duration are seen in many peers by our method. This is considered that when the system downloads pieces with emergency priorities, download requests from other peers also swarm about a certain peer and causes a long waiting time for the download request.

6 CONCLUSION

The purpose of this research is to propose a method which is suitable for video streaming using P2P while solving the problem of client server system resource overload in the content delivery market. The research has proposed a new method of peer and piece selection in a P2P streaming environment using BitTorrent. The proposed simulations examine the effectiveness of the new methods for improving on the established BiToS and RarestFirst methods. It is the research's sincerest hope that the proposed method alleviates some of the current challenges facing streaming content delivery.

REFERENCES

[1] YouTube : http://www.youtube.com/
[2] NicoNico Douga : http://www.nicovideo.jp/
[3] Cisco®, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2012–2017," (2013).

[4] B. Cohen, “Incentives build robustness in bittorrent,” In 1st Workshop on the Economics of Peer-2-Peer Systems, Berkley, CA (2003).
[5]BitTorrentSpecifications. https://wiki.theory.org/BitTorrentSpecification .
[6]Arnaud Legout, G. UrvoyKeller, and P. Michiardi, “Rarest First and Choke Algorithms Are Enough,” IMC '06 Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pp.203–216 (2006).
[7] A. Vlavianos, M. Iliofotou, and M. Faloutsos, “BiToS: Enhancing Bittorrent for Supporting Streaming Applications,” INFOCOM 2006, Proc. of 25th IEEE International Conference on Computer Communications, pp.1–6 (2006).
[8] Weishuai Yang, Nael Abu-Ghazaleh, “GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent,” Proceedings of 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05) (2005).
[9] I. Stoica, et al. “Chord: A scalable peer-to-peer lookup service for Internet applications,” In Proceedings of ACM SIGCOMM, Volume 31 Issue 4, pp.149-160 (2001).
[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable network,” In Proceedings of ACM SIGCOMM, Volume 31 Issue 4, Pages 161-172 (2001).

(Received December 4, 2013)
(Revised July 28, 2014)

Takanori Kashiwagi received M.S. degree in 2013 from Iwate Prefectural University, Japan. Currently, he works for Caunets Corp.

Jun Sawamoto is currently a Professor of Faculty of Software and Information Science, Iwate Prefectural University, Japan. He received the B.E. and M.E. in mechanical engineering from Kyoto University in 1973 and 1975. He joined Mitsubishi Electric Corporation in 1975. He received his PhD degree from Tokyo Denki University in 2004. His research interests include ubiquitous computing, human-interface system, multi-agent systems, and cooperative problem solving. He is a member of IPSJ, IEEE-CS, ACM.

Hiroyuki Sato is currently a Professor of Faculty of Software and Information Science, Iwate Prefectural University, Japan. He received the B.E. in information engineering from Tsukuba University in 1982. He joined Mitsubishi Electric Corporation in 1982. He received his PhD degree from Tsukuba University in 2003. His research interests include parallel processing, and high performance computing. He is a member of IPSJ, IEICE and IEEE-CS.

Norihisa Segawa received Ph.D. degrees in Information Sciences from Tohoku University, Sendai, Japan, in 2004. Currently, he is working at Faculty of Software and Information Science, Iwate Prefectural University. His research interests are developments of long-range out-door sensor networks.

Eiji Sugino was a researcher at ICOT for the Japanese 5th Generation Computer Project from 1987 to 1990. He received Ph.D. in Information Science from Japan Advanced Institute of Science and Technology in 1997. He is a full-time Lecturer at the Faculty of software and information science, Iwate Prefectural University. His research interests include operating system, parallel software, and dependable computing. He is a member of IPSJ, IEICE, and IEEE.

Yuji Wada received the B.E. and the M.E. in electrical engineering from Waseda University in 1974 and 1976, respectively. He joined Mitsubishi Electric Corporation in 1976. He received the PhD degree in computer science from Shizuoka University of Japan in 1997. He is currently a Professor in the Department of Information Environment, Tokyo Denki University. His research interests include database systems, data mining, and recommendation. He is a member of the IPSJ, the IEICE, the JSAI, the JSSST and the DBSJ.