社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－DPS－138 (36)
2009－CSEC－ 44 (36)
2009／3／6

Models for Multimedia Streaming in P2P Networks

Alireza Goudarzi Nemati†, Tomoya Enokido*, and Makoto Takizawi‡
†‡Seikei University, *Rissho University
†alireza.gn@computer.org, *eno@ris.ac.jp, ‡makoto.takizawa@st.seikei.ac.jp

In peer-to-peer (P2P) overlay networks, multimedia objects are distributed in peers and source peers transmit a multimedia object to receiver peers. In this paper, we would like to discuss serial and parallel types of multi-source streaming (MSS) models where multiples source peers deliver multimedia objects to receiver peers with enough QoS in a streaming model. A multimedia object is realized in a sequence of primitive objects. In the serial MSS model, a receiver peer receives each primitive object from one of the source peers at a time. If a current source peer is expected to support lower QoS than required or not to support a succeeding primitive object, another source peer starts sending primitive objects. In the parallel MSS model, multiple source peers in parallel send primitive object units to the receiver peer.

P2P ネットワークでのマルチメディアストリーミングモデル

グダルジーネマテイ　アリレザ†, 榎戸 智也*, 滝沢　誠†
† 成蹊大学, * 立正大学

P2P ネットワークでは、マルチメディアオブジェクトがピアに分散され、ソースピアから受信ピアにストリーミングサービスが提供される。マルチメ ディアオブジェクトは、 P2P のオーバーレイ層の伝送単位である基本オブジェクトから構成される。本論文では、直列と並列転送を行う多ソースストリーミング (MSS) モデル について論じる。直列 MSS モデルでは、各基本オブジェクトを、受信ピアは一つのソースピアから受信する。一方、並列 MSS モデル では、受信ピアは複数のソースピアから並列に基本オブジェクトを受信する。

1. Introduction

A peer-to-peer (P2P) system [5] is composed of peer processes (abbreviated *peers*) interconnected in P2P overlay networks. Here, peers can not only obtain multimedia objects from other peers but also peers holding multimedia objects can support other peers with the multimedia objects. Objects are thus in nature distributed to peers with various ways like downloading and caching. In addition, a receiver peer p_r would like to obtain a multimedia object c from a source peer in a streaming way [2, 9]. A multimedia object c is realized in a sequence of primitive objects $c_1, \cdots, c_m$ ($m \geq 1$). A primitive object is a unit of data transmission among peers at P2P overlay layer. There are multiple source peers $p_1, \cdots, p_n$ ($n \geq 1$). In traditional ways, one source peer p_i sends the multimedia object c to the receiver peer p_r. Here, the receiver peer p_r may not receive the multimedia object c with enough QoS, e.g. due to the movement and faults. We discuss the multi-source streaming (MSS) model [2] where multiple source peers send a multimedia object to a receiver peer so that at least one source peer delivers the multimedia object with enough QoS. There are *serial* MSS (SMSS) and *parallel* MSS (PMSS) models. In the SMSS model, a receiver peer p_r receives each primitive object from one of the source peers, say p_i at a time. If the current source peer p_i might not support the receiver peer p_r with enough QoS or not hold a primitive object, p_r finds another source peer p_j which can support the target primitive object of enough QoS. Then, the source peer p_j starts transmitting primitive objects to p_r. In the paper [1], the authors discuss how to dynamically switch current source peers in change of QoS supported by each source peer. In this paper, we discuss a scheduling method by which source peers send primitive objects to the receiver peer.

In PMSS model, each primitive object is redundantly transmitted by multiple source peers or non-redundantly transmitted by a source peer. In the redundant transmission way, even if some source peer is faulty and some primitive object is lost in the network, the receiver peer p_r can receive every primitive object of the multimedia object c. In addition, primitive objects are in parallel transmitted by multiple source peers. Hence, we can get the higher bandwidth even

if each channel from a source peer p_i to the receiver peer p_r is too slow to send the multimedia object. The paper [2], discuss how multiple source peers send redundantly primitive objects to the receiver peer. In this paper, we discuss a scheduling way on how the receiver peer to make the parallel transmission schedule of multiple source peers.

In section 2, we discuss multimedia objects. In section 3, we present MSS models. In sections 4 and 5, we discuss the serial and parallel MSS models.

2. Multimedia Objects

In distributed multimedia services, a receiver peer p_r receives a multimedia object from a source peer p_i. The source peer p_i sends a multimedia object c to p_r if p_i could provide p_r with the multimedia object c. In the streaming service [3, 4, 9], it is critical for source peers to continuously deliver a multimedia object to receiver peers in real time manner. The streaming service is more significant and economical than the downloading service.

A multimedia object c is decomposed into a sequence S^c of primitive objects $c_1, \cdots, c_m$. A primitive object c_h is a unit of transmission of a peer in an P2P overlay network. A source peer p_i of a multimedia object c holds a subsequence S_i^c of the primitive objects, $S_i^c = \langle c_{k_1 i}, \cdots, c_{k_{l_i} i} \rangle$ $(l_i \geq 1)$ where each c_{ji} stands for an *instance* of a primitive object c_j $(j = k_1, \cdots, k_{l_i}, k_s \in \{1. \cdots, m\})$. A *segment* $\langle c_{k_h i}, c_{k_{h+1} i}, \cdots, c_{k_l i} \rangle$ $(1 \leq h, l \leq l_i)$ is a contingent subsequence of primitive object instances in the subsequence S_i^c if $k_{s+1} = k_s + 1$ for $s = 1, \cdots, l-1$. Here, if $k_{h-1} \neq k_h - 1$ and $k_{l+1} \neq k_l + 1$, the segment is *maximally contingent* in S_i^c. For example, suppose S_i^c is a subsequence $\langle c_{4i}, c_{6i}, c_{7i}, c_{8i}, c_{10i} \rangle$ of primitive object instances. $\langle c_{6i}, c_{7i} \rangle$ is a segment but not maximally contingent. A segment $\langle c_{6i}, c_{7i}, c_{8i} \rangle$ is maximally contingent in S_i^c. Let $Q(c_{ji})$ be QoS of a primitive object instance c_{ji}. Each primitive object instance c_{hi} of a source peer p_i is assumed to support the same QoS Q_{irc}, i.e. $Q(c_{hi}) = Q_{irc}$. For each primitive object c_h, $SP(c_h)$ shows a subset of the source peers, each of which supports a primitive object c_h, i.e. $\{ p_i \mid c_{ih} \in S_i^c \}$. A receiver peer p_r can receive each primitive object c_h from at least one source peer in the source peer set $SP(c_h)$ where Q_{irc} is larger than QoS RQ_{rc} required by p_r $(Q_{irc} \geq RQ_{rc})$. In Figure 1, a multimedia object c is realized in a sequence $\langle c_1, c_2, c_3, c_4, c_5, c_6, c_7 \rangle$ of primitive objects. Let c_{ji} show a primitive object instance of a primitive object c_j held by a source peer p_i. c_{11} and c_{31} show the instances of the primitive object c_1 held by p_1 and p_3, respectively, $SP(c_1) = \{p_1, p_3\}$. $SP(c_2) = \{p_1, p_2, p_3\}$. The peer p_1 holds a subsequence of $S_1 = \langle c_{11}, c_{21}, c_{31}, c_{41}, c_{71} \rangle$. A *segment* is a contingent subsequence like $\langle c_{11}, c_{21}, c_{31}, c_{41} \rangle$.

In the underlying network, a primitive object is transmitted by using protocols like TCP [8].

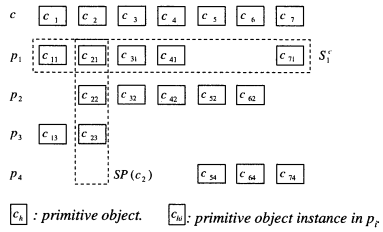

Figure 1. Primitive objects.

3. Multi-Source Streaming (MSS) Models

A receiver peer p_r receives primitive objects of a multimedia object c from source peers $p_1, \cdots, p_n$. A source peer p_i holds all or some primitive objects of the multimedia object c. In the single-source streaming (SSS) model, only one source peer p_i sends the primitive objects of c to the receiver peer p_r. If the source peer p_i is faulty or QoS supported by p_i is degraded, p_r cannot receive the multimedia object c with enough QoS. On the other hand, multiple source peers send primitive objects of the multimedia object c to p_r in the multi-source streaming (MSS) model. Here, even if a source peer p_i transmitting primitive object instances is faulty, another source peer sends the primitive object instance which p_i sends to p_r. In addition, if each of multiple source peers concurrently sends different primitive object instances to p_r, the total throughput can be increased in addition to being tolerant of faults of the source peers.

There are *serial* MSS (SMSS) and *parallel* MSS (PMSS) models. In the MSS model, each primitive object c_h is sent to the receiver peer p_r by one source peer in the source peer set $SP(c_h)$ while a pair of different primitive object instances may be sent by different source peers. Figure 2 shows the SMSS model for primitive objects $c = \langle c_1, c_2, c_3, c_4, c_5, c_6, c_7 \rangle$. If a source peer p_i currently sending primitive objects gets less qualified in terms of bandwidth, another source peer well qualified starts transmitting primitive objects to the receiver peer p_r [1]. Thus, each primitive object is sent by one source peer.

On the other hand, multiple source peers in parallel send instances of multiple primitive objects or multiple instances of a primitive object to the receiver peer p_r in the PMSS model. Let $SSP(c_h)$, be a subset of the source peers in the source peer set $SP(c_h)$, each

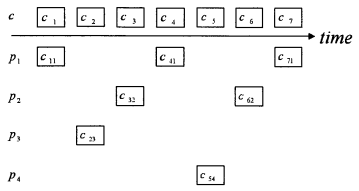

Figure 2. Serial transmission model.

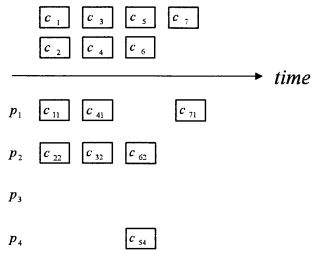

Figure 3. Non-redundant PMSS model.

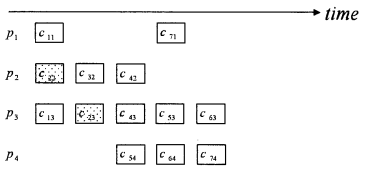

Figure 4. Redundant PMSS model.

of which sends a primitive object c_h to p_r. For example, a pair of the primitive objects c_1 and c_2 are in parallel sent by the source peers p_1 and p_3, respectively. The receiver peer p_r simultaneously receives instances c_{11} and c_{22} from p_1 and p_2, respectively. Here, $SSP(c_1) = \{p_1\}$ and $SSP(c_2) = \{p_2\}$. Since c_1 and c_2 are sent to p_r at a time, the throughput can be increased. On the other hand, a primitive object c_1 is in parallel sent by p_1 and p_2, $SSP(c_1) = \{p_1, p_2\}$. Here, even if p_r fails to receive one instance, say c_{11}, p_r can receive the other instance c_{12}. Thus, the reliability and availability can be increased.

There are furthermore non-redundant PMSS (NPMSS) and redundant PMSS (RPMSS) models. In the NPMSS model, every source peer sends a primitive object c_h different from the other source peers in the subset *SSP*(c_h). Every primitive object c_h is sent by only one source peer. Figure 3 shows the NPMSS model of the primitive objects $c_1, c_2, c_3, c_4, c_5, c_6, c_7$. Here, c_1 and c_2 are in parallel sent by p_1 and p_3, respectively. The primitive objects c_3 and c_4 are in parallel sent by p_1 and p_2, respectively. Primitive objects which are in parallel sent by multiple source peers are referred to as *concurrent*. Compared with the SMSS model, it takes shorter time to deliver every primitive object to the receiver peer p_r in the PMSS model. Here, even if the bandwidth Q_{irc} between p_i and p_r is smaller than the required QoS RQ_{rc}, enough bandwidth can be supported by multiple source peers through in parallel sending primitive objects. The paper [2] discusses how to allocate primitive object instances to multiple source peers whose bandwidths are different.

Each packet of a primitive object c_h is redundantly transmitted by more than one source peer in the source peer subset *SSP*(c_h) with the RPMSS model. Here, the receiver peer p_r redundantly receives each instance sent by multiple source peers. Figure 4 shows an example of the RPMSS model for Figure 1. Two instances c_{11} and c_{13} are simultaneously sent by p_1 and p_3, respectively. The primitive object c_2 is sent by p_2 in parallel with the instances c_{11} and c_{13}. c_2 is in addition sent by p_3. Thus, each primitive object is sent by two source peers. Even if a primitive object instance is lost and a source peer is faulty, at least one instance for each primitive object can be delivered to the receiver peer p_r in the RPMSS model. On the other hand, the number of messages and the processing overhead of p_r are increased.

4. Serial MSS Model

4.1. Dynamic model

In a serial multi-source streaming (SMSS) model [1], one source peer supports the receiver peer p_r with each primitive object c_h in the source peer set $SP(c_h)$ at a time while a pair of primitive objects may be sent by different source peers. Here, we have to discuss which source peer sends a primitive object c_h to p_r. In addition, suppose a source peer p_i sends a primitive object c_h to p_r. Then, another source peer p_j is selected to send a succeeding primitive object c_{h+1}. The receiver peer p_r has to receive c_{h+1} from p_j without any gap as shown in Figure 5. First, we discuss how to select a source peer p_i which to send each primitive object c_h. There are the following ways to select a source peer for each primitive object c_h:

1. Dynamic way: A current source peer is dynamically selected for each c_h while transmission.
2. Static way: A source peer for each primitive object is *a priori* fixed before transmission.

Suppose a receiver peer p_r is receiving a primitive object c_h from a current source peer p_i. In the dynamic

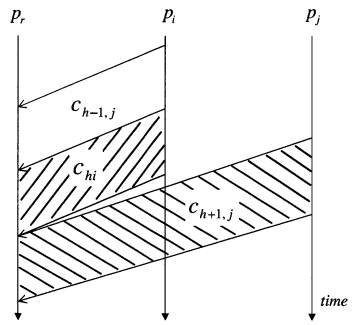

Figure 5. Non-gap transmission.

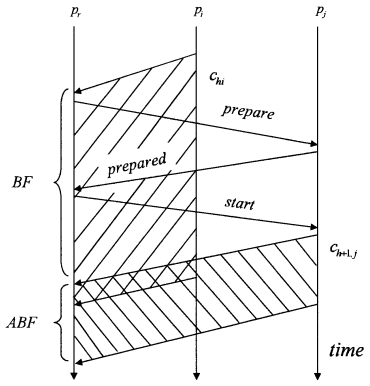

Figure 6. Source peer switching.

way, p_r selects another source peer p_j for c_{h+1} if the one of the following conditions holds:

1. The current source peer p_i does not support the succeeding primitive object c_{h+1}.
2. The current source peer p_i may not support enough QoS of c_{h+1} even if p_i holds c_{h+1}.

Here, the receiver peer p_r has to find another source peer p_j which can support p_r with the primitive object c_{h+1} of enough QoS and to do the negotiation with p_j before the end of the transmission of the current primitive object c_h. The authors discuss the acquaintance-based protocol for detecting source peers and switching source peers at the underlying network level [1].

The receiver peer p_r receives primitive object instances from the current source peer p_i and stores them to the buffer BF. In order to realize non-gap delivery of the primitive object instance $c_{h+1,j}$, p_r has to receive both the instances c_{hj} from p_i and $c_{h+1,j}$ from p_j as shown in Figure 6. The receiver peer p_r is equipped with an auxiliary buffer ABF to store primitive object instances sent by another source peer. Suppose a receiver peer p_r is receiving a primitive object c_{hi} from the current source peer p_i and expects to receive $c_{h+1,j}$ from another source peer p_j. The receiver peer p_r sends a *prepare* message to p_j. On receipt of *prepare*, p_j prepares the transmission of the primitive objects and sends the *prepared* message to p_r. If the source peer p_j sends *prepared* to p_r, p_r sends a *start* message to p_j. On receipt of *start*, p_j starts sending the primitive object instance $c_{h+1,j}$ to p_r. The receiver peer p_r stores the primitive object instance $c_{h+1,j}$ sent by p_j to *ABF* as shown in Figure 6. On receipt of the *end* of the instance c_{hj}, p_r takes the succeeding instance $c_{h+1,j}$ from ABF.

4.2. Transmission schedules

In the static way, a receiver peer p_r makes a serial transmission schedule SS_c which shows what source peer supports the receiver peer p_r with each primitive object c_h of a multimedia object c. SS_c shows a sequence of source peers $\langle p_{1k_1}, \cdots, p_{mk_m} \rangle$ where each p_{hk_h} stands for a source peer in $SP(c_h)$ which would send a primitive object c_h.

First, the receiver peer p_r collects QoS Q_{irc} of each source peer p_i in the source peer set $SP(c_h)$ for each primitive object c_h by using the following algorithm [6, 7, 10]. Then, the receiver peer p_r makes a schedule $SSC_c = \langle p_{1k_1}, \cdots, p_{mk_m} \rangle$ (each $p_{ik_i} \in \{p_1, \cdots, p_m\}$) from the multimedia object $c = \langle c_1, \cdots, c_m \rangle$ based on the QoS information as follows:

[Serial schedule SS_c]

1. For the first primitive object c_1, select a source peer p_k in $SP(c_1)$ such that QoS Q_{krc} is greater than the required QoS RQ_{rc} and the segment $\langle c_{1k}, c_{2k}, \cdots \rangle$ in the primitive object sequence S_i^c is the largest.
2. For each primitive object c_h ($h = 1, \cdots, m$), select a source peer p_k in $SP(c_h)$ such that p_k is in the source peer set $SP(c_h)$, $Q_{hrc} \geq RQ_{rc}$, p_k is selected for the preceding primitive object c_{h-1}. Otherwise, select a source peer p_k in $SP(c_h)$ such that $Q_{krc} > RQ_{rc}$, and a maximally contingent segment $\langle c_{hk}, c_{h+1,k}, \cdots, c_{h+l,k} \rangle$ in the primitive object sequence S_i^c is the largest.

4.3. Transmission of primitive objects

According to the serial transmission schedule SS_c $\langle p_{1k_1}, \cdots, p_{mk_m} \rangle$, the receiver peer p_r negotiates with each source peer p_{ik_i} to support the primitive object c_i of the multimedia object c ($i = 1, \cdots, m$). If each source peer p_{ik_i} in the schedule SS_c agrees on sending the primitive object c_i to p_r, p_r starts receiving the primitive objects from the source peers. If some

source peer p_{ih_i} does not agree, the schedule SS_c for $c_{1k_1}, \cdots, c_{mk_m}$ is reconstructed.

In the previous example, the serial schedule SS_c is $\langle p_1, p_1, p_1, p_1, p_2, p_2, p_1 \rangle$ for the primitive objects $c = \langle c_1, c_2, c_3, c_4, c_5, c_6, c_7 \rangle$. First, p_r sends a *start* message with $\langle 1, 4 \rangle$ to the source peer p_1. That is, p_r asks p_1 to send the primitive objects c_1 to c_4, i.e. instances c_{11}, c_{21}, c_{31} and c_{41}. Then, p_1 starts sending the instances $c_{11}, c_{21}, c_{31}, c_{41}$ to p_r. Before receiving c_4, p_r asks the source peer p_2 to prepare transmitting c_{52} and c_{62}. In order to synchronize the transmissions of the instances c_{41} and c_{32} from different source peers p_4 and p_3, p_r uses one auxiliary buffer ABF to store the instance c_{32}. Before receiving the end of c_{41}, p_i starts sending c_{32}. The receiver peer p_r receives the instance c_{32} in ABF.

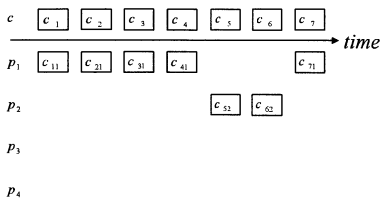

Figure 7. Serial MSS model.

The source peers $p_1, \cdots, p_n$ send primitive objects to p_r according to the schedule SS_c as follows:

[Serial transmission]

1. The receiver peer p_r collects QoS information on the multimedia object $c = \langle c_1, \cdots, c_m \rangle$. Then, p_r makes a serial transmission schedule $SS_c = \langle p_{1k_1}, \cdots, p_{mk_m} \rangle$ for $\langle c_1, \cdots, c_m \rangle$ according to the scheduling algorithm.
2. For every source peer p_{hk_h} in $SS_c = \langle p_{1k_1}, p_{2k_2}, \cdots, p_{mk_m} \rangle$, p_r sends a *prepare* message to p_{hk_h} $(h = 1, \cdots, m)$.
3. On receipt of *prepare*, the source peer p_{hk_h} prepares transmitting a primitive object c_h required by p_r. Then, p_{hk_h} sends *prepared* to p_r. If not prepared, p_{hk_h} sends *no* to p_r.
4. On receipt of *no*, p_r makes the schedule SS_c again and goto step 2.
5. On receipt of a *prepared* message from every source peer in SS_c, $h = 1$.
6. The receiver peer p_r sends a *start* message of c_h to p_{hk_h} in SS_c.
7. On receipt of the *start* message of c_h from p_r, p_{hk_h} starts transmitting c_h to p_r.
8. On receipt of messages of the current primitive object c_h from p_{hk_h}, the messages of c_h are stored in the buffer BF.
9. Before receiving the *end* message of c_h, p_r sends a *start* message to $p_{h+1,k_{h+1}}$ in SS_c. $h = h + 1$; goto step 7.
10. On receipt of messages of c_{h+1} from the source peer $p_{h+1k_{h+1}}$, messages of c_{h+1} are stored in the additional buffer ABF.
11. The source peer p_{hk_h} sends the *end* message of c_h if p_{hk_h} sends up c_h to p_r.
12. On receipt of the *end* message of c_h from p_{hk_h}, p_r sends an *end* message to p_{hk_h}. p_r takes ABF as the buffer BF. If $h = m$, p_r terminates.

5. Parallel MSS Model

The receiver peer p_r receives multiple primitive objects from multiple source peers in the PMSS model. In the non-redundant PMSS (NPMSS) model, each primitive object c_h is sent by one source peer p_i in the source peer set $SP(c_h)$ $(h = 1, \cdots, m)$. Some number of primitive objects are in parallel sent by multiple source peers. In the static way, p_r makes a parallel transmission schedule PS_c for each source peer to transmit primitive objects to p_r. Let MP_r show the maximum number of primitive object instances which p_r can simultaneously receive. The parallel transmission schedule PS_c is made as follows:

[Parallel schedule PS_c]. Initially, $t = 0$.

1. $h = t \cdot MP_r$. Primitive objects $c_{h+1}, \cdots, c_{h+MP_r}$ are taken from $S^c = \langle c_1, \cdots, c_n \rangle$ and are stored in P_t.
2. For each primitive object c_k in P_t, one source peer p_i is selected in $SP(c_k)$ where $Q_{irc} > RQ_{rc}$ and p_i sends some primitive object at round $t - 1$. Here, each source peer is allowed to send at most one primitive object in P_t.
3. $t = t+1$; If $t \cdot MP_r \leq m$, goto step 1. Otherwise terminate.

In the example of Figure 1, $MP_r = 2$, i.e. the receiver peer can receive at most two primitive objects at a time. First, a source peer of the primitive objects c_1 and c_2 are taken in the set P_0 at round 0. For c_1 and c_2, the source peers, say p_1 and p_3 are selected, respectively. Then, c_3 and c_4 are taken in P_2. The source peer p_1 is selected for c_3 because p_2 is taken at round $t = 0$. Then, p_2 is selected for c_4. For c_5 and c_6, p_2 and p_4 are taken, respectively. Then, p_4 is selected for c_7. Figure 8 shows the parallel transmission schedule PS_c obtained here. The transmission time of the multimedia object can be reduced in the PMSS model. For example, it takes four units in Figure 8 while it takes seven units in the serial MSS model of Figure 7.

In the RPMSS model, each primitive object c_h is sent by multiple source peers. Even if the receiver peer p_r fails to receive a primitive object instance c_{hj} of c_h

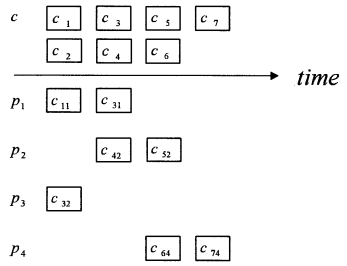

Figure 8. Parallel MSS model.

from a source peer p_j, p_r can receive an instance c_{hi} of c_h from another p_i. Suppose there are d primitive objects $c_1, \cdots, c_d$. Let $c_{[1 \cdots d]}$ be a *parity* object for the primitive object $c_1, \cdots, c_d$, i.e. obtained by taking the exclusive or of $c_1, \cdots, c_d$. Here, even if one object out of $c_1, \cdots, c_d$, $c_{[1 \cdots d]}$ is lost, the lost object can be recovered from the other objects [10]. A segment composed of d contingent primitive objects is a *recovery segment*. In the sequence of primitive objects, we put a parity object $c_{[1 \cdots d]}$ of each recovery segment $c_1, \cdots, c_d$. A sequence obtained by putting parity objects to c is an *enhanced sequence* $\bar{c}$. A subsegment $\langle c_1, \cdots, c_d, c_{[1 \cdots d]} \rangle$ is a *recovery unit*. In Figure 9, c_{12}, c_{34}, c_{56}, and c_{77} are parity objects. $\langle c_1, c_2, c_{12} \rangle$ is a recovery unit. Here, if one of the source peers is faulty or one object in a recovery unit is lost, the receiver peer p_r can receive every primitive object of the multimedia object ($d < m$).

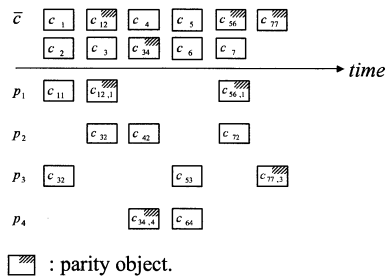

Figure 9. Redundant PMSS model.

6. Concluding Remarks

In this paper, we discussed how to support a receiver peer with enough QoS of the multimedia steaming service by cooperation of multiple source peers. We proposed SMSS and PMSS models where a receiver peer can receive primitive objects of a multimedia object c with enough QoS from one of multiple source peers. A multimedia object is realized in a sequence of primitive objects. In the SMSS model, only one source peer sends a primitive object. In this paper, we discussed the static way where the receiver peer makes a transmission schedule showing which source peer sends each primitive object. In the PMSS model, multiple primitive objects are in parallel sent by multiple source peers. We also presented the scheduling algorithm for the NPMSS model.

References

[1] A. Goudarzinemati, T. Enokido, and M. Takizawa. A multi-source streaming model for mobile peer-to-peer (p2p) overlay networks. In *Proc. Network-Based Information Systems (NBiS-2008)*, pages 122–131, 2008.

[2] S. Itaya, N. Hayashibara, T. Enokido, and M. Takizawa. Distributed coordination protocols to realize scalable multimedia streaming in peer-to-peer overlay networks. In Proc. of ICPP2006 (2006), pages 569–576, 2006.

[3] H. Montes, G. Gomez, and D. Fernandez. An end-to-end qos framework for multimedia streaming services in 3g networks. In *Proc. 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 4, pages 1904–1908, 2002.

[4] S. Mungee, N. Surendran, and D. Schmidt. The design and performance of a CORBA audio/video streaming service. In *Proc. 32nd Annual Hawaii International Conference on HICSS-32 System Sciences*, page 14pp., Volume Track 8 1999.

[5] Napster. http://www.napster.com.

[6] Y. Nishimura, T. Enokido, and M. Takizawa. Striping communication protocol for reliable multimedia communication in a hierarchical group. *In proc. 25th IEEE International Conference on Distributed Computing Systems Workshops,*, 7:734–740, 2005.

[7] Y. Nishimura, N. Hayashibara, T. Enokido, and M. Takizaw. Design of hierarchical group to realize a scalable group. *Journal of Mobile Multimedia (JMM)*, 1:180–197, 2005.

[8] J. Postel. RFC 761: DoD standard transmission control protocol, Jan. 1980.

[9] R. Ramanujan, J. Newhouse, M. Kaddoura, A. Ahamad, E. Chartier, and K. Thurber. Adaptive streaming of mpeg video over ip networks. In *Proc. 2nd Annual Conference on Local Computer Networks*, pages 398–409, 2–5 Nov. 1997.

[10] T. Tojo, T. Enokido, and M. Takizawa. Notification-based qos control protocol for multimedia group communication in high-speed networks. *In Proc. 24th IEEE International Conference on Distributed Computing Systems (ICDCS2004)*, 0:644–651, 2004.