International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.609.4

(01/2018)

SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

OSI networking and system aspects – Networking

Managed P2P communications: Multimedia streaming peer protocol

Recommendation ITU-T X.609.4

ITU-T

Recommendation ITU-T X.609.4

Managed P2P communications: Multimedia streaming peer protocol

Summary

Recommendation ITU-T X.609.4 specifies the peer protocol for distributed multimedia streaming services over managed peer-to-peer (P2P) architecture as defined in Recommendation ITU-T X.609. This Recommendation satisfies the signalling requirements that are specified in Recommendation ITU-T X.609.3. This Recommendation also specifies protocol messages among peers, behaviours, and the types of peer relationship.

History

Edition	Recommendation	Approval	Study Group	Unique ID[*]
1.0	ITU-T X.609.4	2018-01-13	11	11.1002/1000/13493

Keywords

Distributed multimedia streaming, managed P2P, peer protocol.

[*] To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

Table of Contents

Recommendation ITU-T X.609.4

Managed P2P communications: Multimedia streaming peer protocol

1 Scope

This Recommendation specifies a peer protocol for multimedia streaming services over managed peer-to-peer networking infrastructure. It describes the following:

– overview of the peer protocol;

– peer protocol elements such as types of peer, types of peer relationship and buffermap structure;

– protocol messages and parameters;

– protocol behaviours.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T X.609]	Recommendation ITU-T X.609 (2015), *Managed peer-to-peer (P2P) communications: Functional architecture*.
[ITU-T X.609.3]	Recommendation ITU-T X.609.3 (2017), *Managed P2P communications: Multimedia streaming signalling requirements*.
[IETF RFC 7159]	IETF RFC 7159 (2014), *The JavaScript Object Notation (JSON) Data Interchange Format*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 overlay network [b-ITU-T X.1162]: An overlay network is a virtual network that runs on top of another network. Like any other network, the overlay network comprises a set of nodes and links between them. Because the links are logical ones, they may correspond to many physical links of the underlying network.

3.1.2 peer [b-ITU-T X.1161]: Communication node on P2P network that functions simultaneously as both "client" and "server" to the other nodes on the network.

3.1.3 peer-to-peer (P2P) [b-ITU-T Y.2206]: A system is considered to be P2P if the nodes of the system share their resources in order to provide the service the system supports. The nodes in the system both provide services to other nodes and request services from other nodes.

NOTE – Peer is the node in a P2P system.

3.1.4 managed P2P [b-ISO/IEC TR 20002]: P2P with manageability features to manage the P2P-based service and P2P network by the P2P participants such as P2P service provider, ISP and peer.

3.1.5 buffermap [ITU-T X.609]: A map showing downloading status of fragments comprising a shared content.

3.1.6 fragment [ITU-T X.609]: A piece of the shared content.

3.1.7 source peer [ITU-T X.609.3]: A peer that streams the multimedia contents to the overlay network. The peer only provides content data to other peers and does not receive it. This peer generates fragments using the multimedia data received from the contents source.

3.1.8 client peer [ITU-T X.609.3]: A peer that sends fragments received from other peers to other peers, and does not generate its own fragments.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 requesting peer: A peer that sends a request to the other corresponding peer.

3.2.2 corresponding peer: A peer that performs an operation on the request received from the requesting peer

3.2.3 candidate peer: A peer that is participating in an overlay network for which no specific peer relationship has yet been established but can be attempted whenever necessary.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

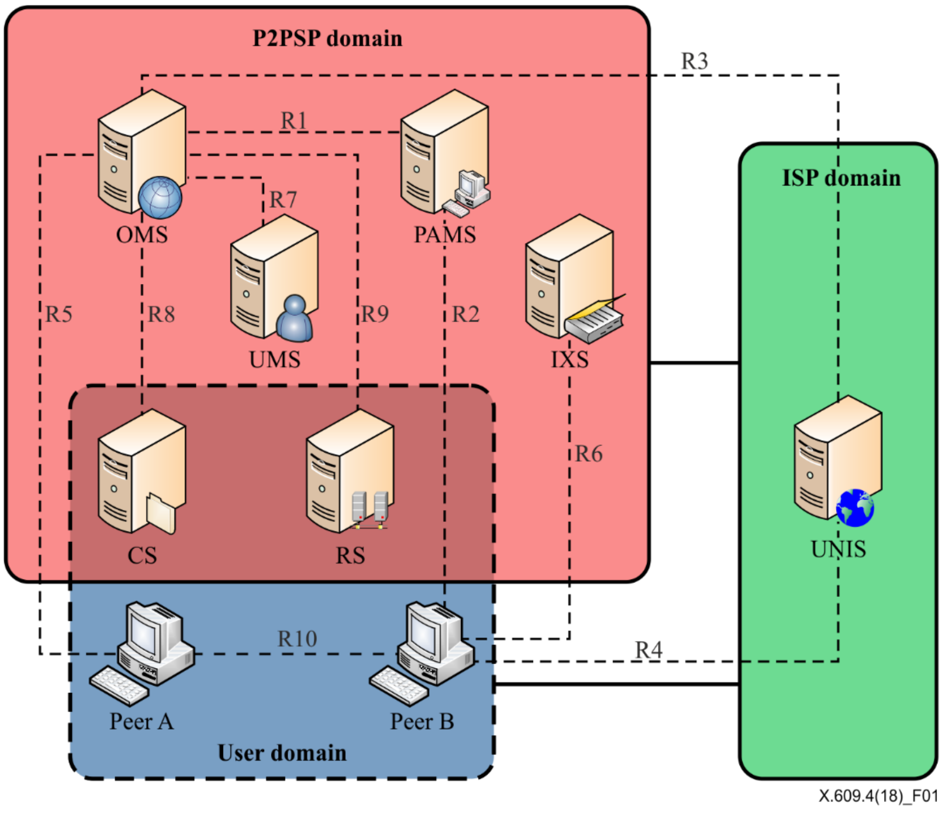
ADR	Average Download Rate
APMR	Average Play point Move Rate
BSON	Binary JavaScript Object Notation
BTT	Buffermap Timetable
CP	Completed Point
CS	Cache Server
DP	Downloading Point
EP	End Point
FD	Forced Delay
JSON	JavaScript Object Notation
MP2P	Managed Peer-to-Peer
NAT	Network Address Translation
NTP	Network Time Protocol
P2P	Peer-to-Peer
PP	Play Point
SHA	Secure Hash Algorithm
SP	Starting Point
TCP	Transmission Control Protocol

5 Conventions

None.

6 Overview

Figure 1 shows framework and reference points of managed P2P defined in [ITU-T X.609]. This Recommendation defines the multimedia streaming peer protocol for reference point R10.



UMS: user profile management server
OMS: overlay management server
PAMS: peer activity management server
RS: relay server
CS: cache server
IXS: index server
UNIS: underlying network information server

Figure 1 – Framework and reference points of managed P2P [ITU-T X.609]

[ITU-T X.609.3] describes signalling requirements for multimedia services over managed P2P architecture defined in [ITU-T X.609].

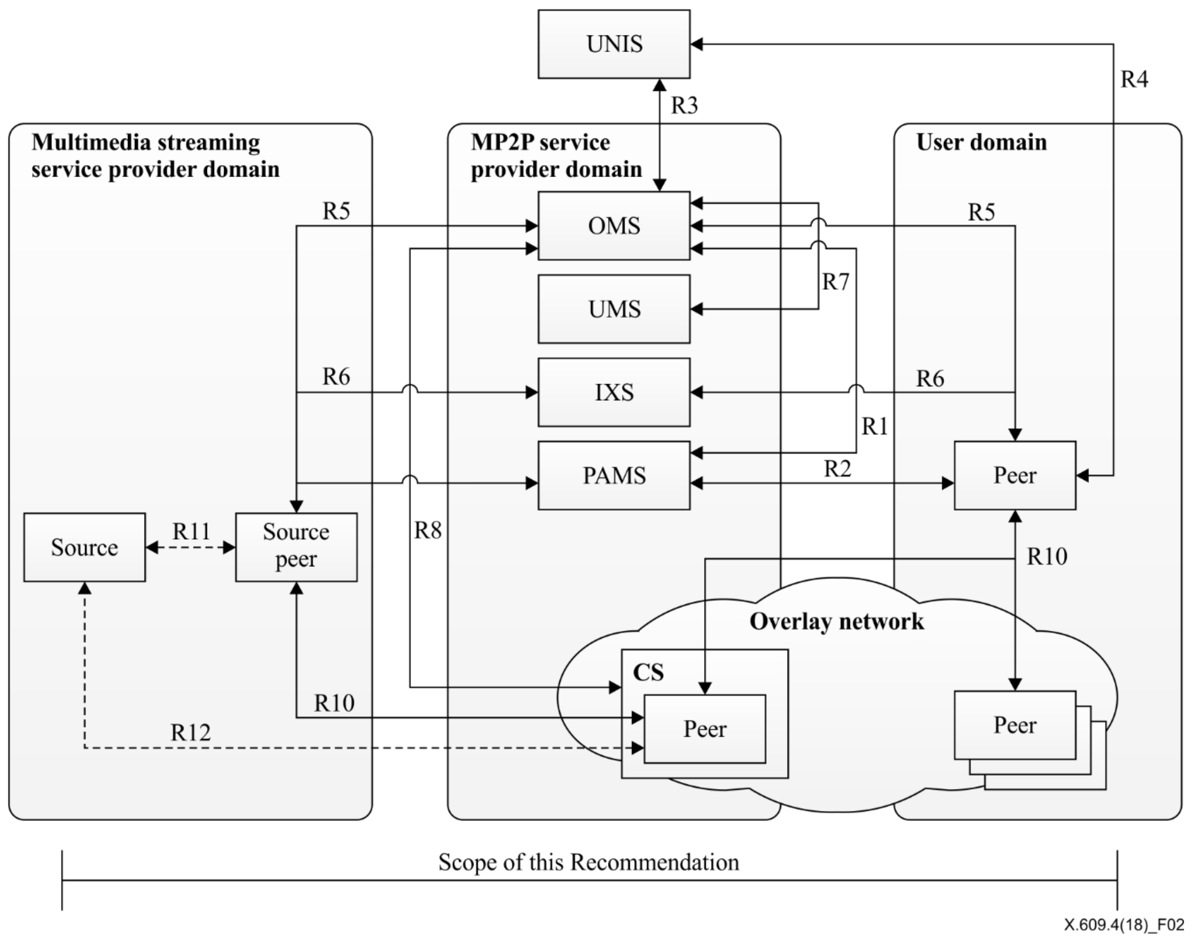


Figure 2 – Architectural overview of multimedia streaming service on MP2P architecture [ITU-T X.609.3]

Figure 2 shows an architectural overview of multimedia streaming service on managed peer-to-peer (MP2P) architecture defined in [ITU-T X.609.3]. This Recommendation defines the protocol for reference point R10.

7 Protocol

This clause describes messages to be exchanged among peers and behaviours of peers on sending and receiving those messages for multimedia streaming services.

The peer protocol consists of three phases as follows:

– In the negotiation phase, each peer exchanges its buffermap with other peers. On receiving the buffermap of a corresponding peer, each peer figures out whether it contains any fragment that it does not have. If that is not the case, it just disconnects and communicates with another peer;

– In the streaming phase, a peer requests the corresponding peer to send a specific fragment that it does not have. In this phase, relationships between peers are classified as neighbourship and partnership. After establishment of a partnership with the other peer, the corresponding peer sends a notification to the partner peer on acquiring new fragments from the other peers;

– In the termination phase, a peer sends termination requests to all corresponding peers to release its resources.

Figure 3 shows the overall procedures of the peer protocol for negotiation, streaming and termination of the multimedia streaming service.

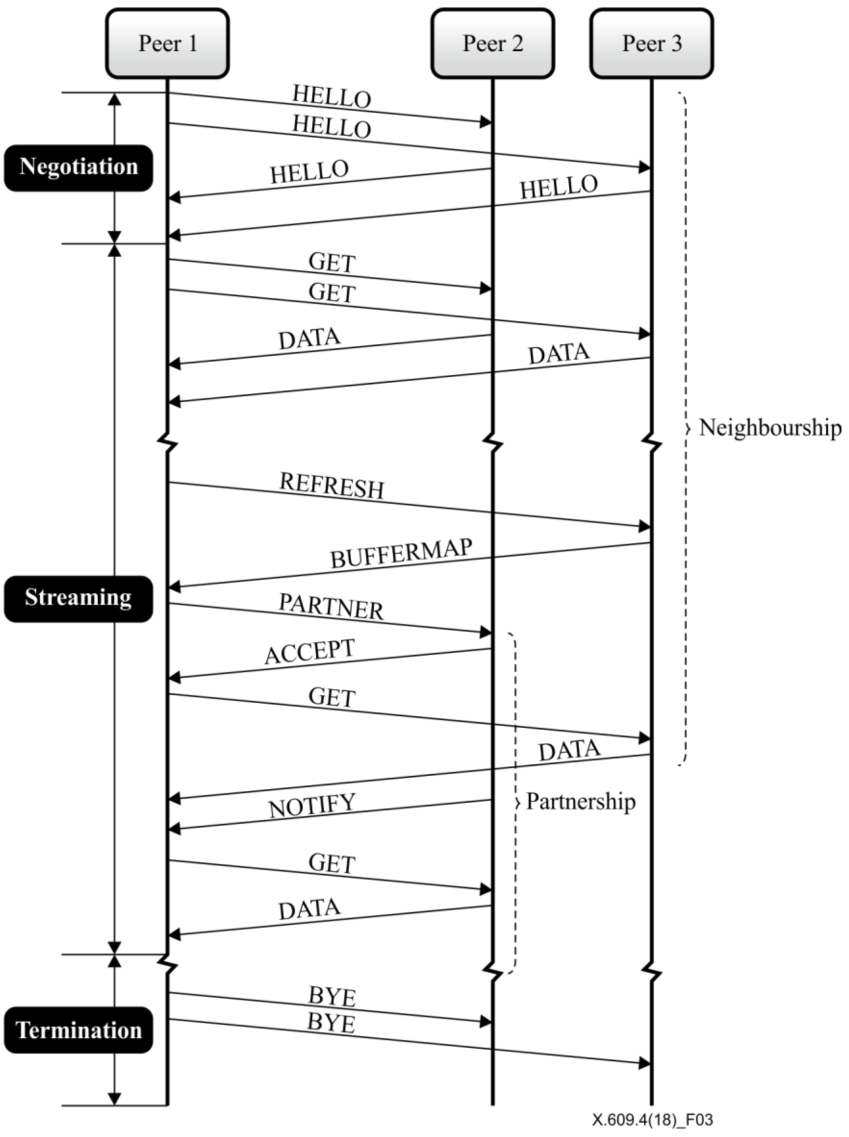


Figure 3 – Overall message flows of peer protocol

7.1 Peer relationships and buffermaps

This clause describes the types of peer and the structure of a buffermap.

7.1.1 Peer relationships

In the multimedia streaming service, new pieces are sequentially generated at appropriate time intervals. Therefore, the operation at the time of the first service entry is different from the operation at the time of having the latest buffer. A combination of a query-based request method and a notification-based request method in requesting the fragment can reduce unnecessary exchange of signalling messages, thereby enabling a more stable operation. Each peer works by dividing its relationship with other peers into two relationships as follows:

– Neighbourship: A peer exchanges fragments with another peer without partnership, and it pulls fragments based on a buffermap;

– Partnership: A peer sends fragments or notifications to another peer with partnership, and does not exchange a buffermap with the corresponding peer.

7.1.2 Structure of a buffermap

Every peer maintains a buffermap to represent the fragments owned by each peer. Peers exchange buffermaps with each other to find fragments to pull. Buffermap types are classified as follows:

– Local buffermap: The buffermap of a local peer;

– Remote buffermap: The buffermap of a corresponding peer.

On receiving a remote buffermap from a corresponding peer, the peer extracts fragment numbers by comparing the local buffermap with the remote buffermap.

A buffermap consists of three sections:

– Completed section: A section with fragments that a peer possesses continuously;

– Downloading section: A section with fragments that a peer does not possess continuously;

– Empty section: A section that will be used for buffering to prevent the downloading point (DP) from overriding the starting point (SP). This section may have valid data but could be overwritten at any time.

Figure 4 shows the structure of a buffermap and its attributes that are used for buffermap negotiations with other peers.

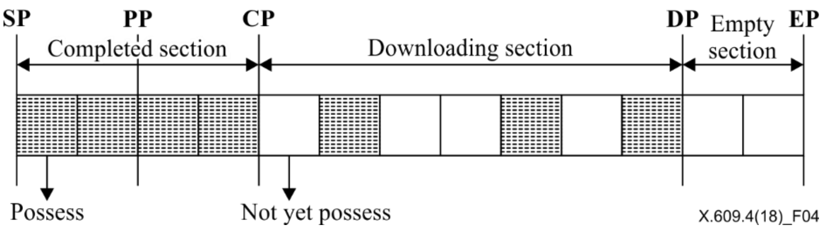


Figure 4 – Structure of buffermap and its attributes

Each buffermap has the following attributes that point to specific fragment identifiers:

– Completed point (CP): This value points to the end of the completed section in the buffermap;

– Downloading point (DP): This value points to the end of the downloading section in the buffermap;

– Starting point (SP): This value points to the identifier at the beginning of the buffermap;

– Play point (PP): This value points to the identifier that passed to the player in the buffermap. This value is used for flow control with other peers.

7.2 Messages

The peer protocol has 14 messages (see clauses 7.2.1 to 7.2.14) for multimedia streaming services which are represented as JavaScript object notation (JSON) [IETF RFC 7159]. Since JSON itself cannot accommodate binary data, messages are converted to binary format such as binary JavaScript object notation (BSON) [b-BSON 1.1].

7.2.1 HELLO

This message is used for establishing neighbourship between peers. When a client peer establishes a transmission control protocol (TCP) connection with a corresponding peer, it sends this message, and the corresponding peer also responds with a HELLO message. The syntax of the message is as follows:

```
{
   "method": "HELLO",
   "proto-version": integer,
   "peer-id": string,
   "overlay-id": string,
   "valid-time": integer,
   "sp-index": integer,
   "cp-length": integer,
   "dp-index": integer,
   "ds-length": integer,
   "buffermap": binary
   "req-btt"; boolean,
   "req-btt-span": {
    "start" : integer,
    "end" : integer
     },
   "btt" : [ {  "piece-id" : integer, "timestamp" : integer }, … ]
}
```

– *peer-id* is the identifier of the peer.
– *overlay-id* is the identifier of overlay network.
– *valid-time* indicates the valid time of the buffermap in seconds.
– *sp-index* indicates the starting index number of the fragment represented by the buffermap.
– *cp-length* indicates the length of completed section within the buffermap beginning *sp-index*.
– *dp-index* indicates the starting index number of the downloading section.
– *ds-length* indicates the length of the downloading section.
– *buffermap* represents the possession status of fragments within the downloading section. If the peer possesses a particular fragment, it is set to '1' and is set to '0', if not.
– *req-btt* indicates whether it wants to receive the buffermap timetable or not.
– *req-btt-span* is used when requesting a buffermap corresponding to a specific time zone.

NOTE 1 – If the value of *req-btt* is true, but the *req-btt-span* is not present, the corresponding peer sends the buffermap timetable for its entire buffermap.

NOTE 2 – If a requesting peer wants to have the buffermap timetable after a certain time, the *end* parameter is not included in the HELLO message.

– *btt* is used to include its own buffermap timetable, when a peer sends its HELLO on receiving a HELLO with *req-btt* of "true".

7.2.2 PELLO

This message is used to establish neighbourship with a peer located behind a network address translation (NAT) or firewall. If the source peer is located behind a NAT or firewall, other peers, including the virtual peer of the cache server, cannot connect to the source peer. The peer receiving this message acts the same as when sending a HELLO message to the peer and uses the same TCP socket to request and receive fragments of the multimedia stream. That is, a source peer actively makes TCP connections with other peers and sends this message to the corresponding peer. If the corresponding peer finds that this peer has a data it needs, it sends a GET request without sending a HELLO message.

The syntax of the message is as follows:

```
{
    "method": "PELLO",
    "proto-version": integer,
    "peer-id": string,
    "overlay-id": string,
    "valid-time": integer,
    "sp-index": integer,
    "cp-length": integer,
    "dp-index": integer,
    "ds-length": integer,
    "buffermap": binary
}
```

– *peer-id* is the identifier of the peer.
– *overlay-id* is the identifier of the overlay network.
– *valid-time* indicates the valid time of the buffermap in seconds.
– *sp-index* indicates the starting index number of the fragment represented by the buffermap.
– *cp-length* indicates the length of the completed section within the buffermap beginning *sp-index*.
– *dp-index* indicates the starting index number of the downloading section.
– *ds-length* indicates the length of the downloading section.
– *buffermap* represents the possession status of fragments within the downloading section. If the peer possesses a particular fragment, it is set to '1' and set to '0', if not.

7.2.3 REFRESH

This message is used to update the buffermap of a neighbour peer. When the requesting peer receives all fragments of the neighbour peer, it requests to update the buffermap status by sending this message. The syntax of the message is as follows:

```
{
   "method": "REFRESH",
   "piece-index"; integer,
   "piece-number": integer
}
```

– *piece-index* is an index number of the fragment that is of interest to the requesting peer.
– *piece-number* indicates the number of fragments to be represented by the buffermap. If this value is set to '0', the corresponding peer sends the whole buffermap after *piece-index*.

7.2.4 BUFFERMAP

This message is used to deliver the buffermap of the peer to refresh information about the fragment possession status. Since this protocol is for providing multimedia streaming services, fragments are created sequentially. This means that the buffermap can be simplified by compressing the information about the completed section. The syntax of the message is as follows:

```
{
  "method": "BUFFERMAP"
  "piece-index": integer,
  "cp-length": integer,
  "dp-index": integer,
  "ds-length": integer,
  "buffermap": binary,
  "timestamp": string
}
```

– *piece-index* is an index number requested by the REFRESH message. If *piece-index* of the REFRESH message is set to '0', this will be *sp-index* of the local buffermap.
– *cp-length* indicates the length of the completed section within the buffermap.
– *dp-index* indicates the starting index number of the downloading section.
– *ds-length* indicates the length of the downloading section.
– *buffermap* represents the possession status of fragments within the downloading section. If the peer possesses a particular fragment, it is set to '1' and set to '0', if not.
– *timestamp* includes the network time protocol (NTP) timestamp of the first fragment of the buffermap.

7.2.5 GET

This message is used to request a particular fragment from a corresponding peer. The syntax of the message is as follows:

```
{
  "method": "GET"
  "piece-index": integer,
  "offset": integer
}
```

– *piece-index* is the index number of fragment being requested.
– *offset* is set to 0(zero) if it needs all of the data of the fragment. If it has part of the fragment, a requesting peer specifies the offset value to this field and then the requested peer sends the fragment from the offset.

7.2.6 BUSY

This message is used when rejecting neighbourship establishment with a requesting peer due to lack of resources of the peer. When the corresponding peer sends this message, it responds with a reason. The syntax of the message is as follows:

```
{
  "method": "BUSY"
  "reason": string
}
```

– *reason* is set to convey reasons of denial. The reason can be 'shortage of bandwidth', 'the number of concurrent connections has been exceeded' or 'internal resources such as storage and CPU are busy'.

7.2.7 DATA

This message is used for delivering fragments. The peer can use the hash value to verify the integrity of the fragment from a remote peer. This message may contain a signature that are signed by a private key of the source peer on the hash value to prevent malicious manipulation. The syntax of the message is as follows:

```
{
  "method": "DATA"
  "piece-index": integer,
  "offset": integer,
  "data-size": integer,
  "timestamp": string,
  "hop-count": integer,
  "hash": string,
  "signature": string,
  "encrypted-hash": string,
  "data": binary
}
```

– *piece-index* is an index number of the fragment.
– *offset* indicates the offset for a particular fragment.
– *data-size* indicates the size of the fragment.
– *timestamp* indicates the creation time of fragments with NTP timestamp format.
– *hop-count* indicates the number of steps from the source peer. On receiving a fragment, the hop count is incremented by 1.
– *hash* contains a SHA-1 hashing value for the fragment.
– *signature* contains the signed hash value by digital signature of the source peer.
– *encrypted-hash* contains encrypted hash value by digital signature of the source peer.
– *data* contains the fragment.

7.2.8 CANCEL

This message is used to revoke a pending request on a particular fragment to a remote peer. The syntax of the message is as follows:

```
{
  "method": "CANCEL",
  "piece-index": integer,
  "offset": integer
}
```

– *piece-index* is an index number of the fragment.
– *offset* indicates the offset for a particular fragment.

7.2.9 PARTNERreq

This message is used to establish partnership with a corresponding peer. When a peer requests a partnership with the corresponding peer, it sends this message with the identifier of the fragment that starts the partnership. The syntax for the message is as shown in below:

```
{
  "method": "PARTNERreq",
  "start-piece-index": integer
}
```

– *start-piece-index* is an index number of the fragment that starts the partnership.

7.2.10 ACCEPT

This message is used to grant a partnership with the requesting peer upon receipt of a PARTNETreq message. The syntax of the message is as follows:

```
{
  "method": "ACCEPT",
  "result": boolean,
  "reason": string
}
```

7.2.11 NOTIFY

This message is used to notify the acquisition of new fragment to a partner peer. The syntax of the message is as follows:

```
{
  "method": "NOTIFY",
  "piece-index": integer
}
```

7.2.12 BYE

This message is used for the release peer relationship with a corresponding peer. The syntax of the message is as follows:

```
{
  "method": "BYE"
}
```

7.2.13 GETBTT

This message is used to request buffermap timetable information of the corresponding peer separately. The syntax of the message is as follows:

```
{
    "method": "GETBTT",
    "overlay-id": string,
    "req-btt-span": {
     "start": integer,
        "end": integer
         }
    }
```

– *overlay-id* is the identifier of the overlay network.
– *req-btt* indicates whether it wants to receive a buffermap timestamp or not.
– *req-btt-span* is used when requesting a buffermap corresponding to a specific time zone.

NOTE – When sending a HELLO message, a requesting peer only requests a buffermap after a certain time. In this case, the *end* parameter is not included.

7.2.14 BTT

This message is used to provide the buffermap timetable information to a requesting peer when receiving a GETBTT message. It is also possible to send this message without receiving a GETBTT message. The syntax of the message is as follows:

```
{
    "method": "BTT",
    "peer-id": string,
    "overlay-id": string,
    "btt" : [ {  "piece-id" : integer, "timestamp" : integer }, … ]
 }
```

– *overlay-id* is the identifier of the overlay network.
– *peer-id* is the identifier of the peer.
– *btt* is used to include its own buffermap timetable.

7.3 Behaviour of peers

This clause describes the behaviour of the peers for multimedia stream distribution. The behaviour of a peer varies depending on the type of peer. Figure 5 and Figure 6 show state transition diagrams on sending/receiving messages from the corresponding/requesting peer.

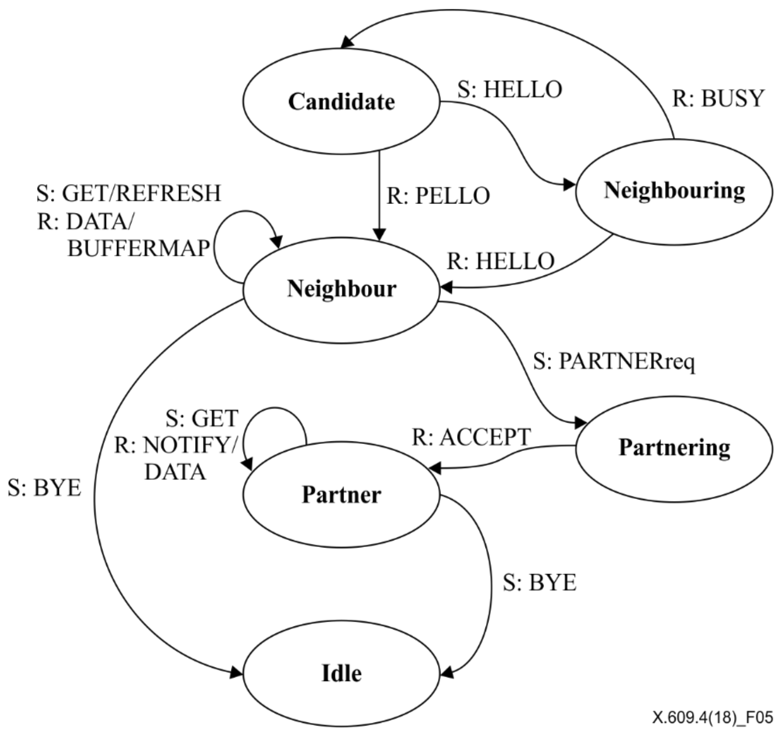


Figure 5 – State machine diagram for a requesting peer

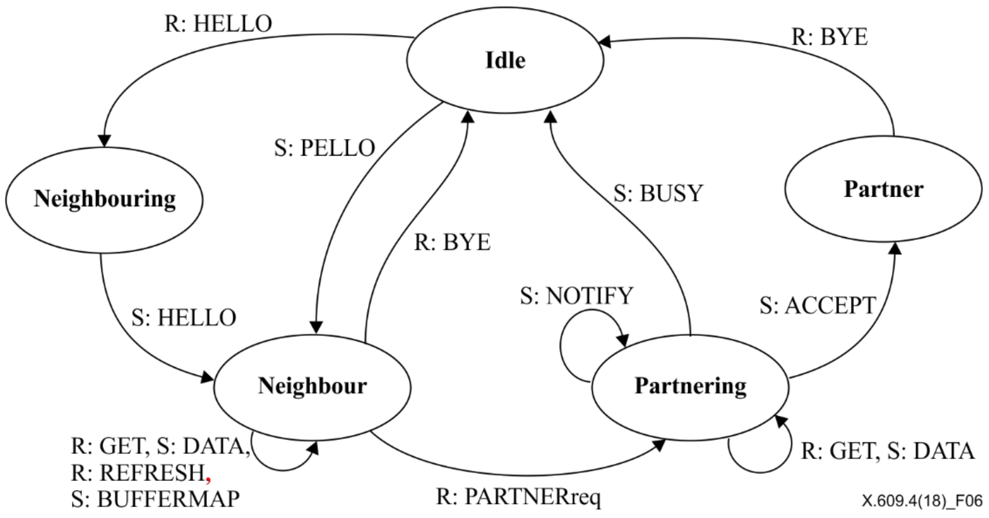


Figure 6 – State machine diagram for a corresponding peer

7.3.1 Negotiation phase

During the negotiation phase, each peer exchanges their buffermap to find the peer that has necessary fragments after connection establishment. In this clause, it is assumed that the peer initiating the HELLO message is a requesting peer.

7.3.1.1 Requesting peer

When a client peer establishes a TCP connection with another peer, it sends a HELLO message that contains the local buffermap and waits for a response from the corresponding peer. For a source peer, that does not need the buffermap of corresponding peer, the source peer can send a PELLO message to the corresponding peer. In this case, the corresponding peer acts as if it sent a HELLO message as described in clause 7.3.1.2.

7.3.1.2 Corresponding peer

On receiving a HELLO message from the requesting peer, the corresponding peer sends a HELLO message containing the local buffermap. If there are no resources available, it responds with a BUSY message and disconnects the TCP connection. On receiving a PELLO message from the requesting peer, it compares the local buffermap to the remote buffermap to find the index number of the required fragments. The peer that received the PELLO message behaves as if it sent the HELLO message without sending the HELLO message.

7.3.1.2.1 Validation checks

If the *valid-time* of the HELLO message is not equal to that of the requesting peer, the corresponding peer proceeds to the termination phase or disconnects the TCP connection with the requesting peer.

7.3.1.2.2 Starting point initialization

The SP value plays an important role in determining the sliding position of the buffermap. The buffermap of a peer should be in company with the buffermaps of other peers with some margin. When a new peer joins a particular overlay network, it does not have SP for its local buffermap and the SP value is determined during buffermap negotiation with other peers. The initial SP value is determined in a different manner depending on the characteristics of the service. For example, if a user wants live broadcasting, the largest value of the corresponding peer's buffermap would be selected and if synchronized play is required, the SP value closest to the specified synchronization time is selected. Figure 7 shows a logical structure of a buffermap received from a corresponding peer.

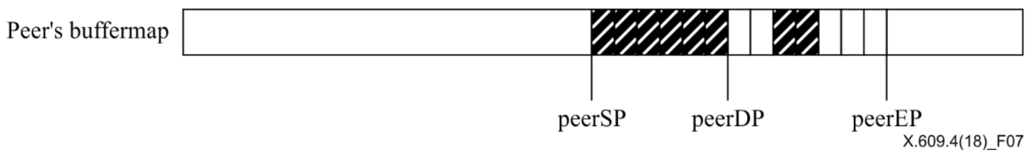


Figure 7 – Logical structure of a buffermap from a corresponding peer

If the peer has just been initiated and there is no buffermap timetable information of the corresponding peer, the initial SP value is determined as follows:

$$SP = peerDP - ROUND[\ ½ * \max\{(peerDP - peerSP), 1\}\ ]$$

When it comes to provide synchronized distribution, it uses timetable information that is exchanged during the buffermap negotiation. Since a source peer embeds an NTP timestamp into the DATA message on creation of the piece, it is possible to calculate the approximate presentation timing. Furthermore, it is also capable of calculating the corresponding peer's buffermap timetable (BTT) using the HELLO or GETBTT messages. However, unlike the server-client streaming model, P2P-based streaming has some limitations in providing hard real-time streaming. Therefore, a forced delay (FD) needs to be provided and this delay will be chosen by the service provider or the peer. If a peer has buffermap timetable information of the first contacted corresponding peer, the initial SP value is calculated according to one of the three cases (see Figure 8) that can happen on initializing the SP for synchronization among multiple peers.

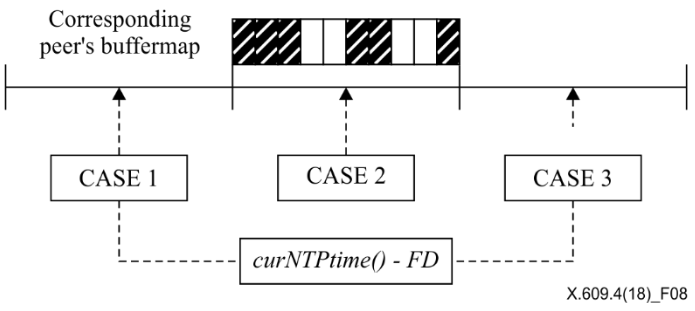


Figure 8 – Initializations of SP for synchronization among multiple peers

– CASE 1: If the buffermap of the peer already passes the selected SP value of this peer, the initial SP value is chosen using the average of the time deviations of the fragmentation time included in the BTT and the fragment number corresponding to *curNTPtime() – FD* is inferred. As there is no data to receive from this peer, the connection with the corresponding peer is terminated and another peer is connected to receive the data.

– CASE 2: If there is a fragment selected as the SP value of this peer, it starts with the request fragment starting with this SP value.

– CASE 3: In this case, it uses the average of the fragmentation time deviations included in the BTT to derive the fragment number corresponding to *curNTPtime() – FD* and requests the fragments received by the currently connected peer. In addition, connections to other peers are requested in parallel to fill the buffermap.

7.3.2 Streaming phase

In the streaming phase, the peers exchange fragments with each other. In this phase, two kinds of relationship can be made; neighbourship and partnership.

7.3.2.1 Requesting peer

If there are fragments that the requesting peer does not have, it sends a GET message repeatedly. If this peer has multiple neighbour peers, it does not request the same fragment from multiple neighbour peers. The client peer can establish a partnership or neighbourship with multiple peers at the same time. If it fails to receive a particular fragment due to network loss with a corresponding peer, it immediately requests the fragment from another peer.

7.3.2.1.1 Requesting fragments from a neighbour

The requesting peer sends a GET message to the corresponding peer to get the fragments it needs. In response, the corresponding peer embeds the fragment data in a DATA message and delivers it.

If the requesting peer receives all the fragments held by the corresponding peer, it sends a REFRESH message to know the status of the peer's most recent buffermap.

7.3.2.1.2 Transition to partnership

If a corresponding neighbour peer satisfies a certain condition, it can be switched to partner peer. The condition depends on the implementation that reflects the characteristics, such as bitrate, of the multimedia stream. For example, when the requesting peer successfully receives fragments in series with an appropriate bitrate, it sends a PARTNETreq message to the corresponding peer to establish a partnership.

7.3.2.1.3 Starting point adaptation

If the buffermap of the current negotiating peer is ahead of the local buffermap, it moves the SP of the peer forward. As shown in the Figure 9 and Figure 10, if the buffermap of the corresponding peer is ahead of the buffermap of P1, the SP of this peer is moved to SP' and the buffermap of P1 is moved accordingly. If real-time performance is important, such as live broadcasting, it is necessary to adjust the SP' to indicate latest value as much as possible as shown in Figure 9. If it is important to provide smooth service rather than real-time performance, readjust the SP' to the section where enough pieces are secured as shown in Figure 10.

Because of the nature of streaming, the buffermap continues to move with time, if the SP value of the corresponding peer is used as the SP of the local buffer map, the fragment corresponding to the SP may not be included in the buffermap as it moves during buffermap adaptation. Therefore, it is necessary to set the value of SP' by jumping an appropriate interval (N) in order to prevent this situation from occurring.

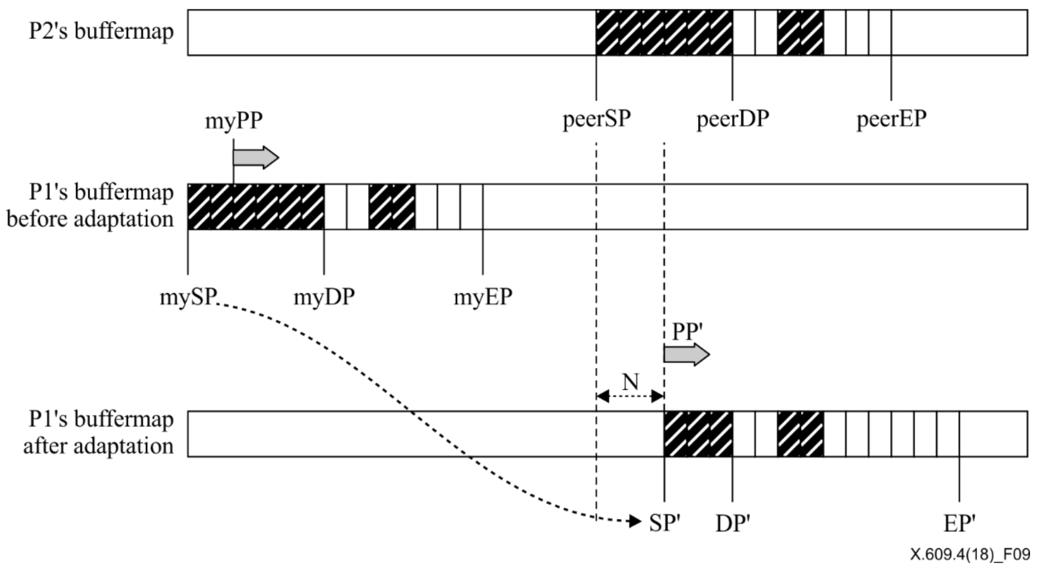


Figure 9 – Adjusting start point by jumping N fragments from SP of a corresponding peer

In addition, if the SP' is set too close to *peerDP*, there is a high probability that the fragments will not be retained at the required time. Therefore, it is necessary to have some margin in setting the SP' value.

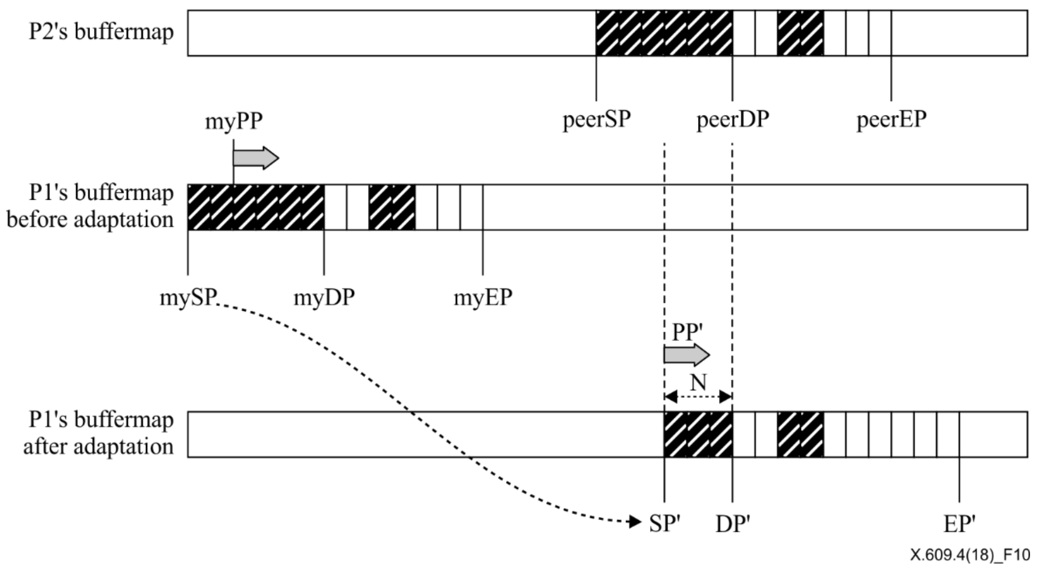


Figure 10 – Adjusting start point by jumping N fragments from DP of a corresponding peer

For a dynamic and autonomous decision of the SP, it needs to flexibly derive the SP' value by taking into consideration average play point move rate (APMR) and average download rate (ADR). Because APMR is almost the same as the generation rate of fragments, it is possible to deduce the reproduction bitrate of the media using APMR and ADR. That is, the combination of these values can determine the optimal value suited to the multimedia characteristics of the session in consideration of the network situation.

If mySP is not yet set, the SP value is calculated as follows:

$$SP = peerDP - ROUND[1/2 * \max\{(peerDP - peerSP), 1\}]$$

If *myEP* is smaller than *peerSP*, the *SP'* value is recalculated as follows;

```
R= APMR/ADR; /* ( 0 < R < 1 ) */
N= ROUND[max{(peerDP - peerSP) x R, 1}];
if (ADR > Limit)  SP' = peerDP - N;
else              SP' = peerSP + N;
```

If the ADR is higher than a pre-specified limit, the SP' is set at a point close to the *peerDP*. If the ADR is lower than the limit, the value is determined at a point close to the *peerSP*, so that the stability and real-time characteristics of the service can be flexibly selected. The limit value is calculated by use of APMR.

7.3.2.1.4 Requesting fragment from a partner

Upon receiving a NOTIFY message from the corresponding peer, the requesting peer delivers a GET message. The corresponding peer that receives the GET message, sends the fragment data to a DATA message. If the piece identifier is larger than the PP of the requesting peer, then the partnership with the corresponding peer is released.

7.3.2.2 Corresponding peer

When the corresponding peer receives a GET message from the requesting peer, it sends the requested fragment within the DATA message. On receiving a REFRESH message from the requesting peer, it sends a BUFFERMAP message that contains the latest buffermap status of the corresponding peer. On receiving a PARTNERreq message from the requesting peer, it sends an ACCEPT message if it has enough resources. On successful establishment of partnership, it sends a NOTIFY message on acquiring a specific fragment from another peer. The peer sends NOTIFY messages to each of the partner peers on receiving a new fragment.

7.3.3 Termination phase

In the termination phase, the peer trying to terminate the peer relationship explicitly sends a BYE message to the corresponding peer. If the connection is lost without receiving a BYE message, the peer behaves as if it received a BYE message.

There are two major cases where the relationship with the peer ends.

– Corresponding peers are no longer interested in fragments

– No notifications from partner peers for a certain period.

7.3.3.1 Requesting peer

The peer that wants to terminate the relationship sends a BYE message to the corresponding peer. If the TCP connection is lost, the peer assumes that the corresponding peer has sent a BYE message.

7.3.3.2 Corresponding peer

On receiving a BYE message from a requesting peer, the corresponding peer releases the connection with the requesting peer. If the connection with requesting peer is lost without receiving a BYE message, the peer assumes that it has received a BYE message from the requesting peer.

Bibliography

[b-ITU-T X.1161]	Recommendation ITU-T X.1161 (2008), *Framework for secure peer-to-peer communications*.
[b-ITU-T X.1162]	Recommendation ITU-T X.1162 (2008), *Security architecture and operations for peer-to-peer networks*.
[b-ITU-T Y.2206]	Recommendation ITU-T Y.2206 (2010), *Requirements for distributed service networking capabilities*.
[b-ISO/IEC TR 20002]	ISO/IEC TR 20002 (2012), *Information technology - Telecommunications and information exchange between systems - Managed P2P: Framework.* https://www.iso.org/standard/50950.html
[b-BSON 1.1]	*BSON – Binary JSON specification Version 1.1 (2013),* http://bsonspec.org

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks, open system communications and security**
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2018