International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.4101
(08/2021)

SERIES Q: SWITCHING AND SIGNALLING, AND ASSOCIATED MEASUREMENTS AND TESTS

Protocols and signalling for peer-to-peer communications

Hybrid peer-to-peer (P2P) communications: Tree and data recovery procedures

Recommendation ITU-T Q.4101

ITU-T

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING, AND ASSOCIATED MEASUREMENTS AND TESTS

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Q.4101

Hybrid peer-to-peer (P2P) communications: Tree and data recovery procedures

Summary

Recommendation ITU-T Q.4101 specifies procedures for the construction and recovery of a hybrid overlay network. The hybrid overlay network constructs a data broadcast tree among peers and calibrates the topology of the overlay network whenever peers are joining or leaving. On calibrating the hybrid overlay network, it needs to be stable even during the reconstruction process. To do this, this Recommendation specifies the procedures for construction and recovery of the tree-based hybrid overlay network, and also specifies the procedures for recovering data that may be lost during the tree recovery procedures.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Q.4101	2021-08-29	11	11.1002/1000/14766

Keywords

Data recovery, hybrid peer-to-peer, overlay network, procedures, tree recovery.

* 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.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T Q.4101

Hybrid peer-to-peer (P2P) communications: Tree and data recovery procedures

1 Scope

This Recommendation describes the data and tree recovery procedures of a hybrid overlay network as follows:

– overlay network construction procedures;

– overlay network recovery procedures;

– data recovery procedures.

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 Q.4100] Recommendation ITU-T Q.4100 (2020), *Hybrid peer-to-peer communications: Functional architecture*.

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 hybrid overlay network [ITU-T Q.4100]: A peer-to-peer overlay network that participating peers exchange data using pull and push method. The hybrid overlay network also provides a way to organize and maintain a tree-style path for pushing data to all peers without loops, as well as fetching data from other peers simultaneously.

3.1.5 reference point [b-ITU-T Y.2012]: A conceptual point at the conjunction of two non-overlapping functional entities that can be used to identify the type of information passing between these functional entities.

3.1.6 hybrid peer [ITU-T Q.4100]: A peer capable of exchanging data using mesh-based and tree-based methods running over a hybrid overlay network

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

HOMS Hybrid Overlay Management Server

HON Hybrid Overlay Network

HP2P Hybrid Peer-to-Peer

P2P Peer-to-Peer

TCP Transmission Control Protocol

WebRTC Web Real-Time Communications

5 Conventions

None.

6 Overview

The hybrid overlay network that is specified in [ITU-T Q.4100] maintains a tree-based hybrid overlay network, described in clause 6 of [ITU-T Q.4100], that is used for broadcasting data. Whenever a new hybrid peer joins the overlay network, it needs to be ensured that a loop does not occur within the tree-based overlay network. In order to keep the tree resilient, it needs to reconstruct the tree-based overlay network as soon as possible when any peer leaves the hybrid overlay network.

[ITU-T Q.4100] classifies the type of the path into a primary path and a candidate path, and the candidate path can be further classified as follows:

– Incoming candidate path: a candidate path established by a remote hybrid peer.

– Outgoing candidate path: a candidate path established by the hybrid peer itself.

The tree-based hybrid overlay network (HON) consists of hybrid peers with primary paths. To continue being resilient to any failure of peers within a tree-based hybrid overlay network, the peers in the hybrid overlay network make multiple candidate paths for rapid recovery. If any primary path of a peer fails, the peer immediately switches one of its candidate paths to a primary path. In this switching process, special consideration is also needed to avoid loops in the tree. In addition, it needs to recover data that has been lost during the tree recovery procedures. This Recommendation provides the following procedures for supporting the recovery of tree and data:

– Hybrid overlay network construction procedures;

– Hybrid overlay network recovery procedures;

– Hybrid overlay network data recovery procedures.

7 Procedures for hybrid overlay network construction and recovery

This clause describes procedures for building a tree-based hybrid overlay network and procedures for recovering from failures in any intermediate peer that provide data broadcasting services.

The shape of a hybrid overlay network will change with every joining of a new peer. It needs to guarantee that there is no loop and provide robustness on the failure of single or multiple node failures.

For fast recovery of a network, each peer establishes multiple candidate paths during the joining procedure and chooses one of them as a primary path. The primary path is used to distribute data among each other.

7.1 Hybrid overlay network construction procedures

Hybrid overlay network is constructed through (1) overlay network joining (2) overlay network participation request propagation (3) candidate path establishment through peer-to-peer connection and (4) primary path establishment process as shown in Figure 7-1.

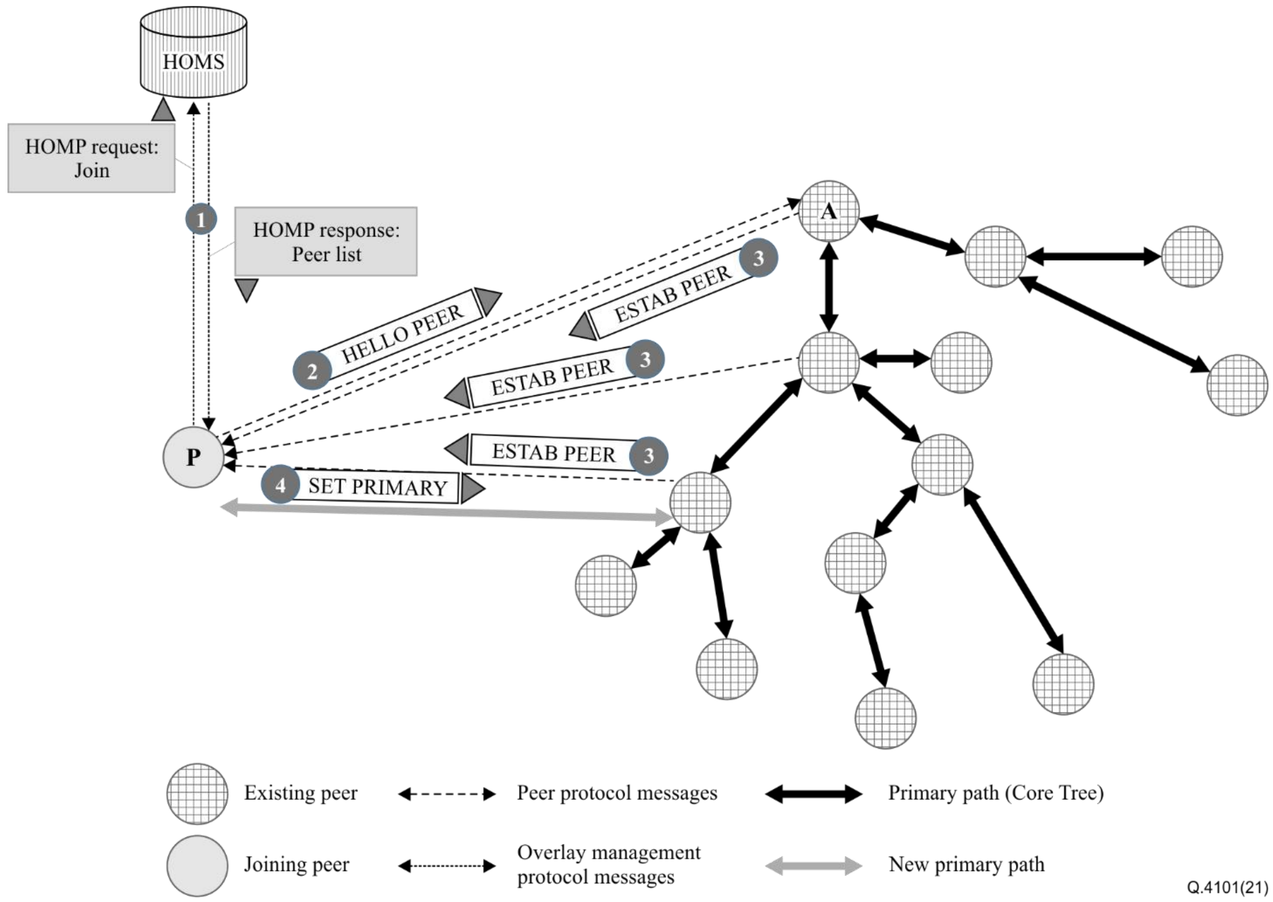


Figure 7-1 – Hybrid P2P network construction procedures

7.1.1 Hybrid overlay network joining

When a new peer joins an existing hybrid overlay network, it firstly sends a request message for joining a hybrid overlay management server (HOMS). This request message includes the following information:

– peer's identifier;

– network address information that other peers can access;

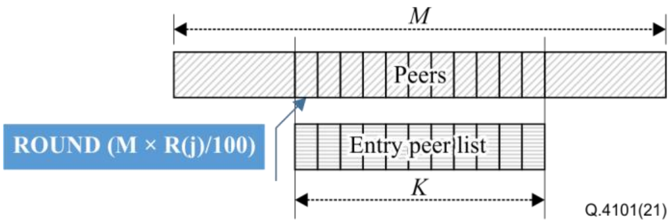
– authentication information if necessary.

The network address information describes the network address where other peers can access the peer and contains information necessary for connection establishment such as transport method (Transmission control protocol (TCP), web real-time communications (WebRTC), etc.), transmission port number, etc.

On receiving this request, HOMS provides an appropriate entry peer list that contains part of the participants' list of the overlay network and issues a new *ticket-id* sequentially to the newly joining peer. This procedure ensures that all peers have a unique *ticket-id* that will be used within that overlay network, and a balanced tree shape. Through this step, an early joined peer has the *ticket-id* lower than that of the peer joining later.

When it comes to selecting the entry peers, HOMS shall not compose the peer list with the peers having low *ticket-id* values to prevent the failure of joining incurred by the possibility that the peers with low *ticket-id* have already consumed their resources to establish primary paths with other peers. If the HOMS composes the peer list with the peers having high *ticket-id*, it also leads to failure of the joining procedures since the peers within the list may not yet be attached to the tree. Hence, it is important to provide an appropriate entry peer list.

When a new peer joins, the HOMS sends a response message with a *ticket-id T* and an *entry_peer_list* composed of the following rules given below:

 	entry peer list = [PI[ROUND(M x R(j)/100)], PI[ROUND(M x R(j)/100)+1], … PI[ROUND(M x R(j)/100)+K]]

– PI: The full set of peers currently remaining in the overlay network, sorted by *ticket-id in ascending order*;

– M: The number of PI's element;

– R(j): The relative position of peer within M, and it is expressed in percentage (%). This value is configured by an administrator of HOMS. For example, if 100 peers are currently participating and the value of R(j) is 80, this means the peer is located in the 80th position in the PI;

– K: This value indicates the maximum number of peers to be included in the *entry peer list.*

7.1.2 Participation request propagation

When the newly joining peer *P(j)* receives the peer list and its *ticket-id*, it chooses one peer *P(n)* from the *entry peer list* and sends a HELLO_PEER request message to let other peers know its participation. If *P(j)* fails to establish a connection with the chosen peer, it resumes with the next peer in the *entry peer list*. If successful, *P(j)* does not try to send the request message anymore. HELLO_PEER request message includes the following information:

– *overlay-id*: is a unique overlay identifier;

– *ttl*: describes how many peers this message propagates up to. Whenever a peer forwards the message, it decrements this value by 1. If the decremented value is 0, the peer does not forward anymore and discards the message;

– *conn_num*: describes the maximum number of connections with other peers that *P(j)* wants to establish;

– *peer's network address information*: network address information that other peers can access to *P(j)*.

On receiving the HELLO_PEER request message, *P(n)* sends a success response and adds *P(j)*'s information into its incoming candidate path list. On receiving the success response, *P(j)* adds the *P(n)*'s information into its outgoing candidate path list.

If *P(n)* has an available capacity on its resource, it sends an ESTAB_PEER request message to *P(j)* including following information:

– overlay-id;
– peer-id;
– ticket-id.

In order to prevent overhead, each peer maintains the following information and uses them to calculate its availabilities:

– the number of maximum/current primary paths;
– the number of incoming/current candidate paths;
– the number of outgoing/current candidate paths.

In addition, *P(n)* is also required to send the received HELLO_PEER request message from *P(j)* to other peers that are connected as a primary path by conducting the following operations:

– decrements the value of *ttl* if it is not zero. If the value is zero, *P(n)* does not send the HELLO_PEER request message and discards the message.
– If *P(n)* has multiple primary paths with other peers except for a newly established peer, it forwards the HELLO_PEER request message to other peers.
– If *P(n)* decides to send an ESTAB_PEER request message to *P(j)*, it subtracts the value of *conn_num* and divides the number of *P(n)*'s primary peers and replaces the *conn_num* value of a HELLO PEER with the calculated value. That is, when the *conn_num* value in the received HELLO_PEER message from *P(j)* is *N*, and *P(n)* has sent an ESTAB_PEER message, and the number of the primary path is *P*, the *conn_num* value in the HELLO_PEER request message sent to each primary peer is set to *(N-1)/P*. If *P(n)* does not have any subordinate peers that are connected as a primary path, the operation is terminated without further forwarding.

Through these procedures, *P(j)* secures multiple outgoing candidate paths, and it sends a SET_PRIMARY request message to the most appropriate peer among the candidate peers to establish a primary path. To select the most appropriate peer, *P(j)* can figure out the metrics by sending the PROBE_PEER message to all candidate peers.

On sending a PROBE_PEER message, *P(j)* inserts its local timestamp into the message, and other peers respond with the timestamp embedded. When *P(j)* receives the response, it calculates the time difference. Figure 7-2 shows an example of a constructed shape of a hybrid overlay network.

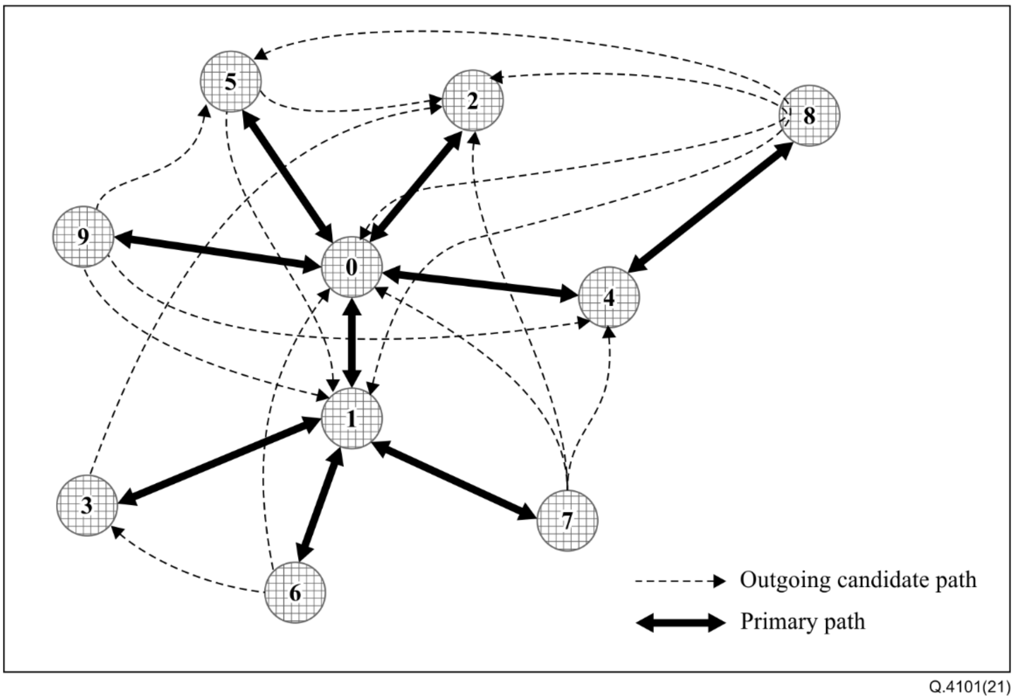


Figure 7-2 – Example shape of a constructed hybrid P2P network

7.2 Hybrid overlay network recovery procedures

A tree-based hybrid overlay network is a shape of a spanning tree that has no loop, and the shape needs to be maintained against any peer dynamics which means peers join and leave during service. That is, all hybrid peers are connected into a tree without a loop. Therefore, it is required that the network supports fast recovery on any possible events such as graceful leave of peer and abnormal termination of peer.

When peer *P(p)* detects a leave of peer *P(j)* that is attached as a primary path, it establishes a new primary path by choosing one of its outgoing candidate paths. However, if *P(p)* tries to re-establish a primary connection on every candidate path, it may lead to a collapse of the tree by the occurrence of a loop. In order to prevent the loop, *P(p)* does not perform a recovery operation, if it has a *ticket-id* value lower than that of *P(j)*. If *P(p)* has *ticket-id* value higher than that of *P(j)* and it has outgoing candidate paths, it sends a SET_PRIMARY message to the most appropriate candidate. If *P(p)* has no more outgoing candidate paths, it requests HOMS to send the latest *entry peer list*, and to do the same procedures with the joining procedures described in clause 7.1 except adding a *recovery* parameter into a HELLO_PEER request message to let the other peers know that this message is for recovery. In this case, HOMS may use a different *R(j)* value from the value of initial joining. For example, if an entry peer selects a node in the bottom 80% at the time of initial participation, the peer may select the node in the top 20% for reconstruction.

When creating an entry peer list, if there is no peer whose *ticket-id* precedes the requesting peer, the HOMS informs that the peer is at the top position by sending an empty list or an error response message. In both cases, the peer does not perform an additional operation.

When *P(n)*, included in the entry peer list from HOMS, receives the HELLO_PEER message having *recovery* parameter from *P(p)*, it adds *P(p)* to its incoming candidate path list. If its incoming candidate capacity is full, *P(n)* disconnects the incoming candidate path with the peer with the lowest *ticket-id* that is lower than *P(p)*'s *ticket-id* and adds *P(p)* to its incoming candidate path list.

Figure 7-3 shows an example of a recovered network on leaving peer 4. If peer 4 of Figure 7-3 leaves, peer 0 and peer 8, which were connected as a primary path with peer 4, detects the leave event. This event can be detected by a timeout or an explicit release message from a leaving peer. Peer 0 does not perform any operation because the *ticket-id* of the leaving peer is lower than itself. Since peer 8 has a higher *ticket-id*, peer 8 sends a SET_PRIMARY message to peer 0 to establish a new primary path with peer 0.

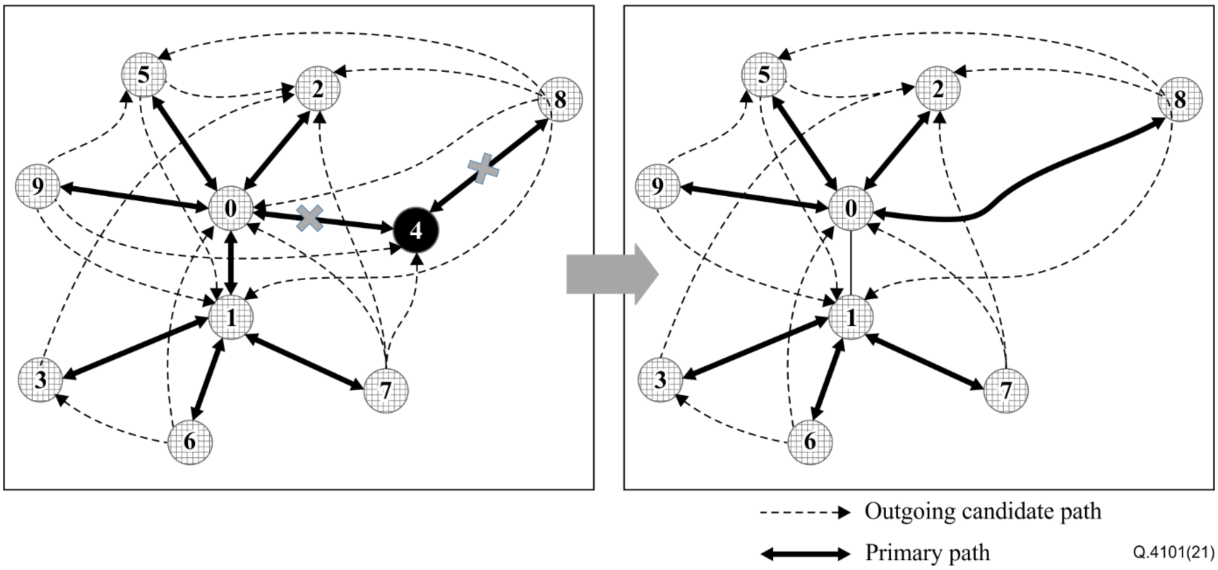


Figure 7-3 – Example shape of a recovered hybrid P2P network

8 Data recovery procedures

Any peer in a hybrid overlay network can be a data source. This means that data generated by any peer is propagated to all other peers. However, in the process of recovering the tree through the process described in clause 7.2, peers can generate data asynchronously, which leads to a segmentation problem where the data generated and distributed in each segment during the tree recovery process is not fully distributed all over the tree as shown in Figure 8-1.

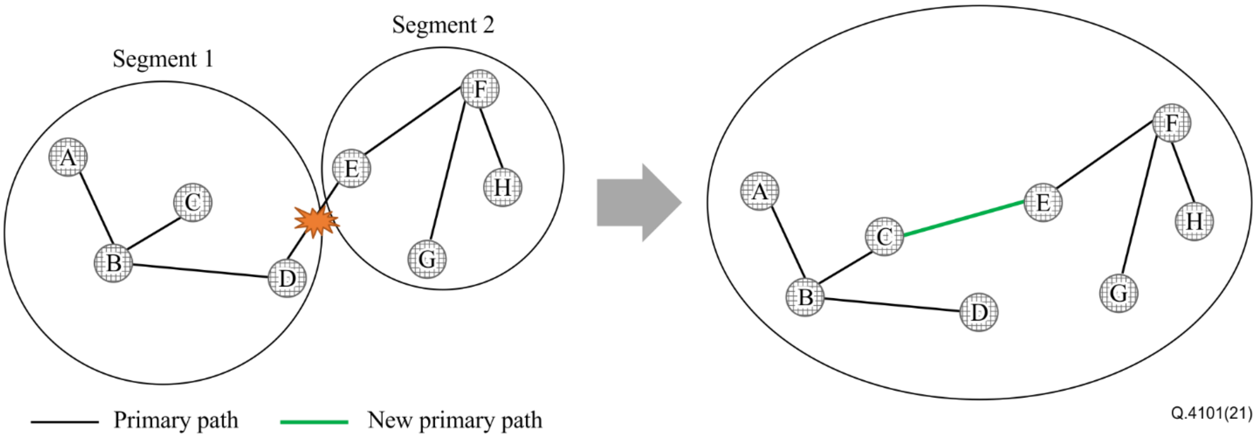


Figure 8-1 – Cache synchronization among segments after tree recovery

When peer E-D's primary path is cut off, the segment {E, F, G, H} and the segment {A, B, C, D} may have different cache due to the newly generated and partially distributed data packets that are not shared between segments. Even though the primary path between two segments is restored, there may be data loss between them. Hence, cache synchronization between two segments is required to complete segment merging.

In order to support the segments merging, it is necessary to cache a certain amount of data transmitted and received by each peer for data recovery.

This clause specifies the caching and recovery policy when creating an overlay network, caching buffer management of peers, data transmission / reception procedures, and data recovery procedures.

8.1 Configurations on hybrid overlay network caching and recovery policy

When an overlay network is being created through reference point R3 described in [ITU-T Q.4100], a peer provides two policies: caching policy and recovery policy. When a peer joins a particular hybrid overlay network, the HOMS provides the policies, and all participating peers follow the same caching and recovery policies on the same hybrid overlay network.

8.1.1 Caching policy

An owner peer that creates a hybrid overlay network specifies a caching policy that includes the following:

– Minimum number of cached data(*mN_Cache*): This parameter specifies the minimum size of caching buffer. Each peer should keep the circular queue with a size larger than the minimum value. If the value is 0, it means that the circular queue is not maintained. That is, the data recovery function is not provided. The data recovery function is disabled when data that has already been delayed is no longer valid in the delay-sensitive service such as a multimedia streaming service.

– Minimum duration of cached data(*mD_Cache*): This parameter specifies the retention time for cached data. Each peer guarantees a minimum retention time for each cached data.

If one of the two values (*mN_Cache*, *mD_Cache*) is not satisfied with respect to the caching policy, the corresponding cached data can be deleted; however, the cached data can be further maintained depending on the implementation i.e., when the value of *mD_Cache* is 10 minutes and the value of *mN_Cache* is 100. When the 101st packet arrives at the caching buffer, the oldest cached data can be deleted from the cache even though the retention time of each data packet to be erased is less than 10 minutes. In addition, even if 15 minutes of time has elapsed while there are still 50 data packets in the circular queue, the data packet older than the specified retention time can be deleted from the cache.

8.1.2 Recovery policy

The cached data can be exchanged in two ways; push and pull method. In pull mode, a peer that needs to get cached data from another peer, requests the data specifically. In push mode, a peer holding cached data broadcasts it to another peer that does not possess the data.

An owner peer that creates a hybrid overlay network specifies a recovery policy that includes the following:

– **recovery-by**: When the value is set to "push", data recovery is performed only through the primary path. This is useful when 1) the real-time data is important, 2) the frequency of data occurrence is not high, and 3) the data size is not large. When this value is set to 'pull', data recovery uses both the primary path and the candidate path. The newly created data is sent along the primary path in an existing way, and only the data to be used for recovery is delivered along the candidate path. This method is useful when data latency is not important and the size of the data is relatively large.

8.2 Data caching and recovery in the hybrid peer

In a hybrid overlay network, each hybrid peer needs to store a certain amount of data in a cache for data recovery. All peers maintain a caching buffer list consisting of a set of caching buffers in the form of a circular queue for each data source, and each caching buffer includes the following information as shown in Figure 8-2:

a) Source Peer ID;

b) DataPacket includes
   i) sequence number;
   ii) datetime that indicates the NTP time of being cached;
   iii) payload.

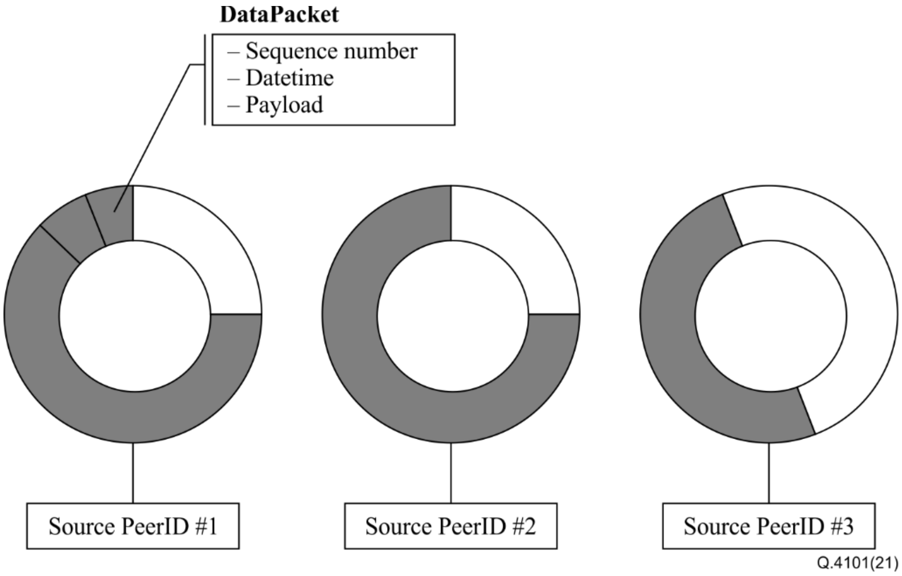


Figure 8-2 – Peer's cache buffer shape

If more than a certain number (or a certain size) is accumulated during the caching process, the oldest data will be deleted first. Also, if the difference between *datetime* of a cached data and the current NTP time exceeds the value of *mD_Cache*, the cached data will be cleared.

8.2.1 Data caching on broadcasting

In a hybrid overlay network, when a peer receives a BROADCAST_DATA message from the primary path, the peer immediately delivers the message to the other primary path. On receiving a BROADCAST_DATA message from another peer, it checks whether the DataPacket with the same source peer ID and the same sequence number already exists in the caching buffer. If it does exist, the message is received again, and the peer does not propagate to other primary paths. Otherwise, the BROADCAST_DATA message is propagated to another primary path and copied to the internal caching buffer. In addition, if the value of *mD_Cache* exists in the corresponding overlay network, it records into the *datetime* of the cached DataPacket by adding *mD_Cache* value to the current NTP time.

8.2.2 Data recovery after re-establishment of primary path

This clause describes the procedures for recovering lost data after the re-establishment of a new primary path. The recovery mode is pre-configured by an owner of the hybrid overlay network in HOMS as specified in clause 8.1.

8.2.2.1 Data recovery with push mode

This clause describes the data recovery operation after tree recovery when the value of *recovery-by* is "push". This operation is performed when the peer establishes a primary path with another peer after one of its primary paths is disconnected. In a hybrid overlay network, each peer receives a *ticket-id* from the HOMS when joining the hybrid overlay network, which is then used for tree recovery. When the primary path is disconnected, the tree recovery operation is not performed when the disappeared peer has a *ticket-id* value higher than that of itself. However, the peer switches the candidate path to the primary path or attempts to establish a new primary path with the new peer when the peer with a lower *ticket-id* value disappears.

Figure 8-3 shows procedures for data recovery when a tree is being reconstructed.

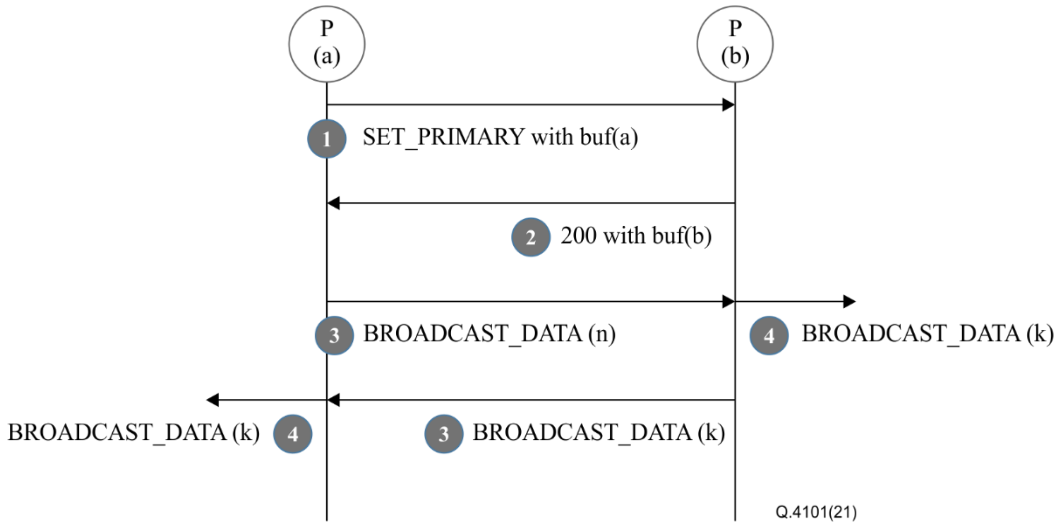


Figure 8-3 – Data recovery procedures in push mode

① *P(a)* sends its buffermap *buf(a)* when sending a SET_PRIMARY message by including a *cache-recovery* parameter with '*true*' value. If the parameter does not exist, it does not proceed with the data recovery hereinafter.

② *P(b)* that receives a SET_PRIMARY request from *P(a)* sends its buffermap *buf(b)* while sending 200 responses.

③ *P(a)* and *P(b)* send data that the corresponding peer does not have using the BROADCAST_DATA message just like normal propagations.

④ *P(a)* and *P(b)* that have received these BROADCAST_DATA messages pass the message to other primary paths as usual.

Figure 8-4 shows an example of data recovery using push mode. When peer 1 disappears, peers 0, 3, 6, and 7 that have a primary path with peer 1 initiate the path recovery procedures. In the case of peer 0, since it has a *ticket-id* value lower than that of peer 1, it does not perform the recovery procedure. In contrary, peers 3, 6, and 7 have higher *ticket-id* values, so one of the candidate paths they have secured should be converted to a primary path by sending a SET_PRIMARY message. Peers 3, 6 and 7 send their buffermap using response messages of SET_PRIMARY, and then peers 2, 0, and 4 send the BROADCAST_DATA message with the data that peers 3, 6 and 7 do not possess.

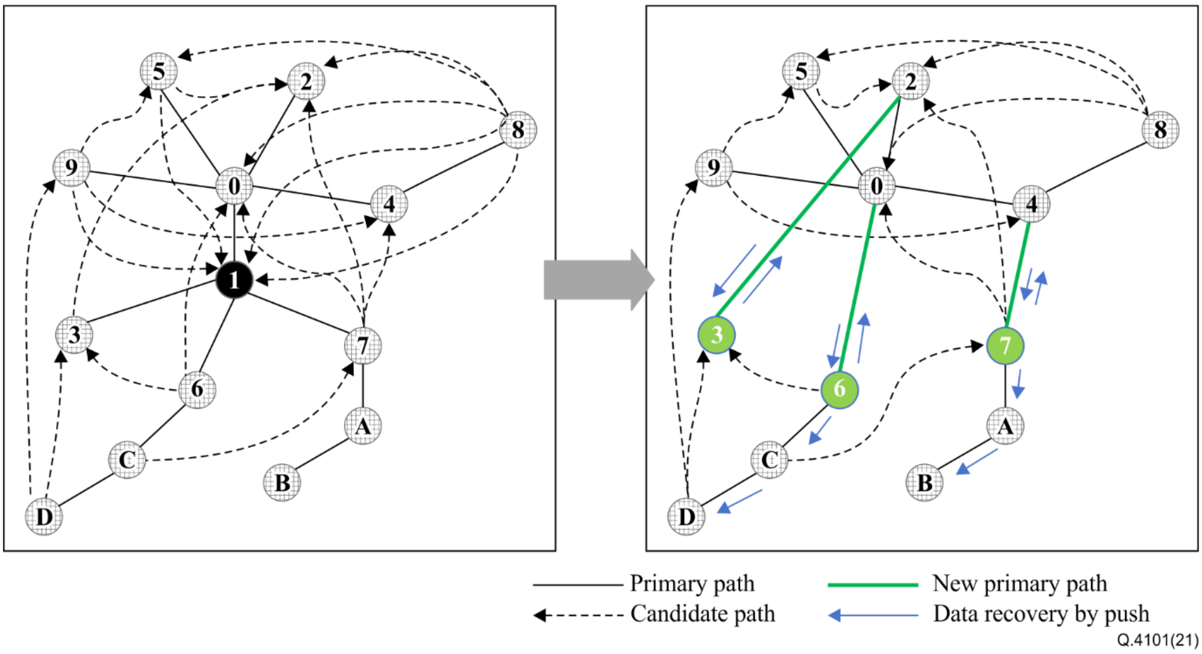


Figure 8-4 – Data recovery procedures in push mode

In addition, peers 2, 0, and 4 transmit the received BROADCAST_DATA messages to synchronize data to the opposite direction simultaneously. Peers 3, 6 and 7 that receive the BROADCAST_DATA message deliver it to other peers A, B, C, and D that have a primary path with them. Through these procedures, data synchronization among the segments is finished.

8.2.2.2 Data recovery with pull mode

This clause describes the data recovery operation when the recovery-by value is "pull". According to the construction procedures specified in clause 8.1, the *ticket-id* of a peer's primary peer and candidate peer is designed to be higher than its own *ticket-id*. That is, candidate peers connected to a specific peer have *ticket-ids* lower than that of the specific peer. On establishing a new primary path, the peer with the higher *ticket-id* retrieves the lost data by the pull method, and the lower peer recovers by the push method as specified in clause 8.2.2.1. This clause describes the pull method in detail in accordance with Figure 8-5.

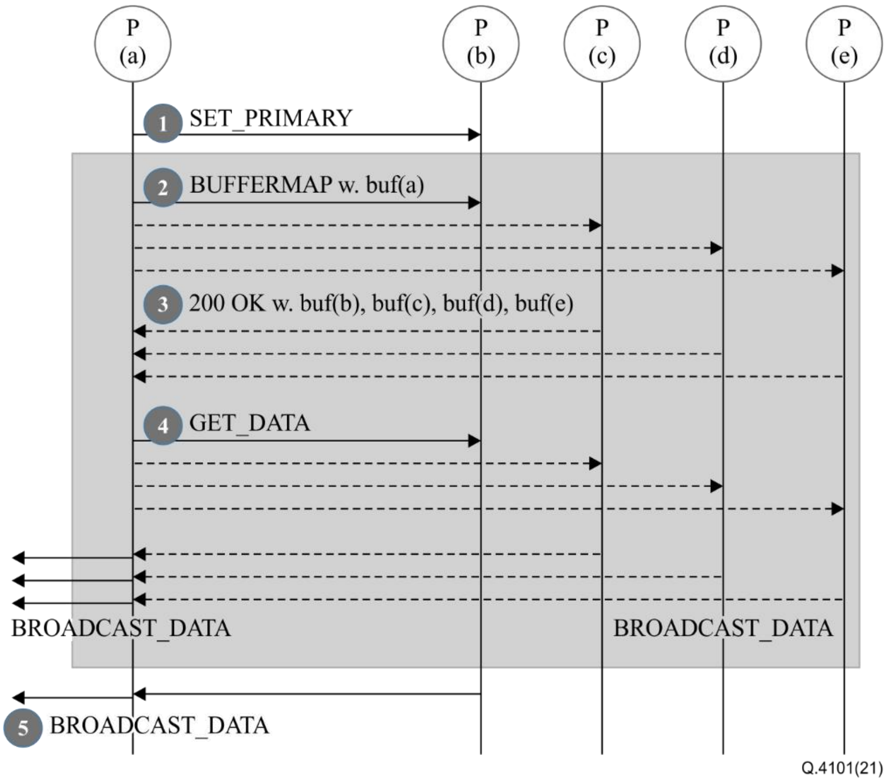


Figure 8-5 – Data recovery procedures in pull mode

When a new primary path is established, the peer recovers the lost data with the following steps:

① *P(a)* sends a SET_PRIMARY request message that does not contain its buffermap to *P(b)*.

② *P(a)* sends a BUFFERMAP request message including its buffermap to other primary paths and candidate paths connected to it.

③ All the peers that receive the BUFFERMAP message send 200 responses with their own buffermap.

④ *P(a)* preferentially sends a GET_DATA request message to candidate peers to request data that they do not have, and if there is data not acquired from candidate peers, it requests the primary peer. That is, by comparing the buffermap collected from all peers, requesting data that they do not have, and when both the candidate peer and the primary peer have it, the candidate peers are used first.

P(b) transmits data that the other peer does not have through the BROADCAST_DATA request message when there is no data held by *P(a)*. That is, push mode is used to send to a peer whose *ticket-id* value is lower than itself.

When one peer goes out, peers connected to the primary path perform tree recovery and data recovery at the same time. In this process, duplicated data can be propagated along the tree. Therefore, before transmitting to the lower level, it is checked whether the corresponding data already exists in its buffer map, and if not, it is cached in its cache buffer. If it already exists, the data is not distributed. This is a safeguard to minimize the spread of duplicated data.

Figure 8-6 shows procedures for data recovery using pull/push mode. When peer 1 disappears, peers 0, 3, 6, and 7, which are connected to the primary path with peer 1, initiate path recovery procedures. In the case of peer 0, since it has a lower value than peer 1, no additional action is performed, and peers 3, 6 and 7 convert one of the secured candidate paths to a primary path. Data generated in the segment {3}, {6, C, D} and {7, A, B} when the tree is being disconnected is propagated to all segments through peers 2, 0 and 4. In these processes, peers 3, 6, and 7 figure out the list of data they do not have through the buffermap exchanges with peers 2, 0, and 4, and request the data to the peers connected to the candidate path parallelly. The peer that received the recovered data completes all segment synchronization by passing it to other peers connected as the primary path.

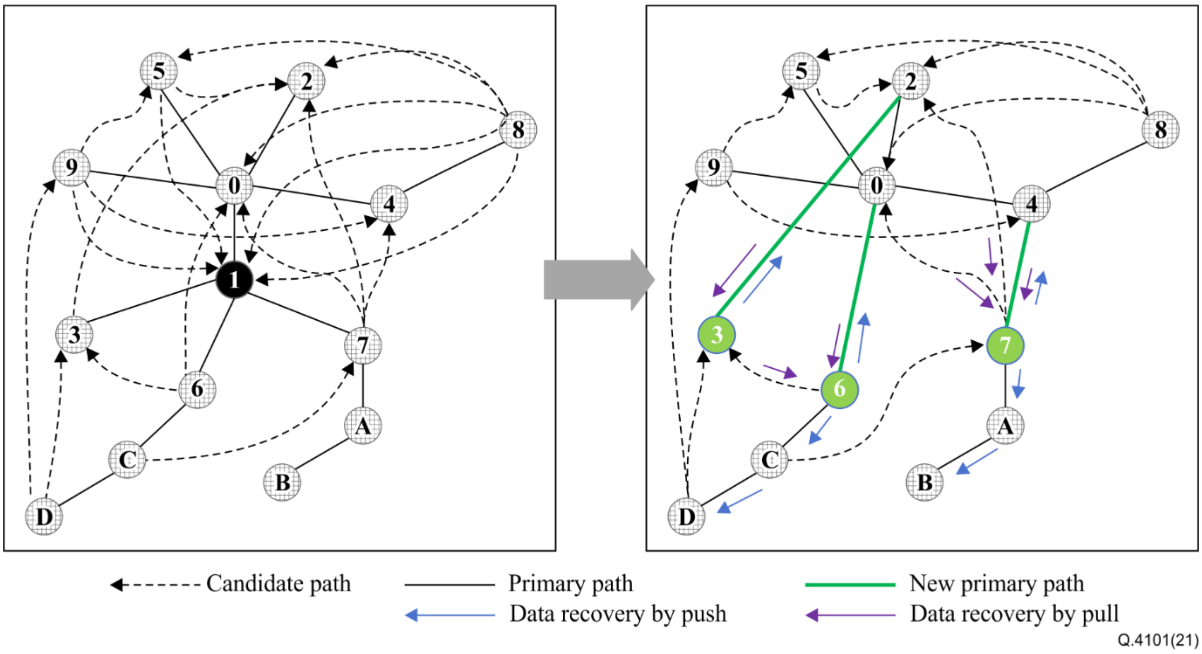


Figure 8-6 – Data recovery procedures in pull mode

8.2.3 Data synchronization of a new peer

When a new peer joins a hybrid overlay network, it needs to fully synchronize its caching buffer with other peers after establishing a primary path. Since the new peer may have multiple candidate paths and one primary path. It can make use of candidate paths for fetching data using pull mode and use the primary path for receiving newly generated latest data from the tree. Figure 8-7 shows procedures for filling the cache buffer of a newly joining peer. In this Figure, it is assumed that *P(a)* has candidate paths with *P(b)*, *P(c)*, *P(d)* and *P(e)*.

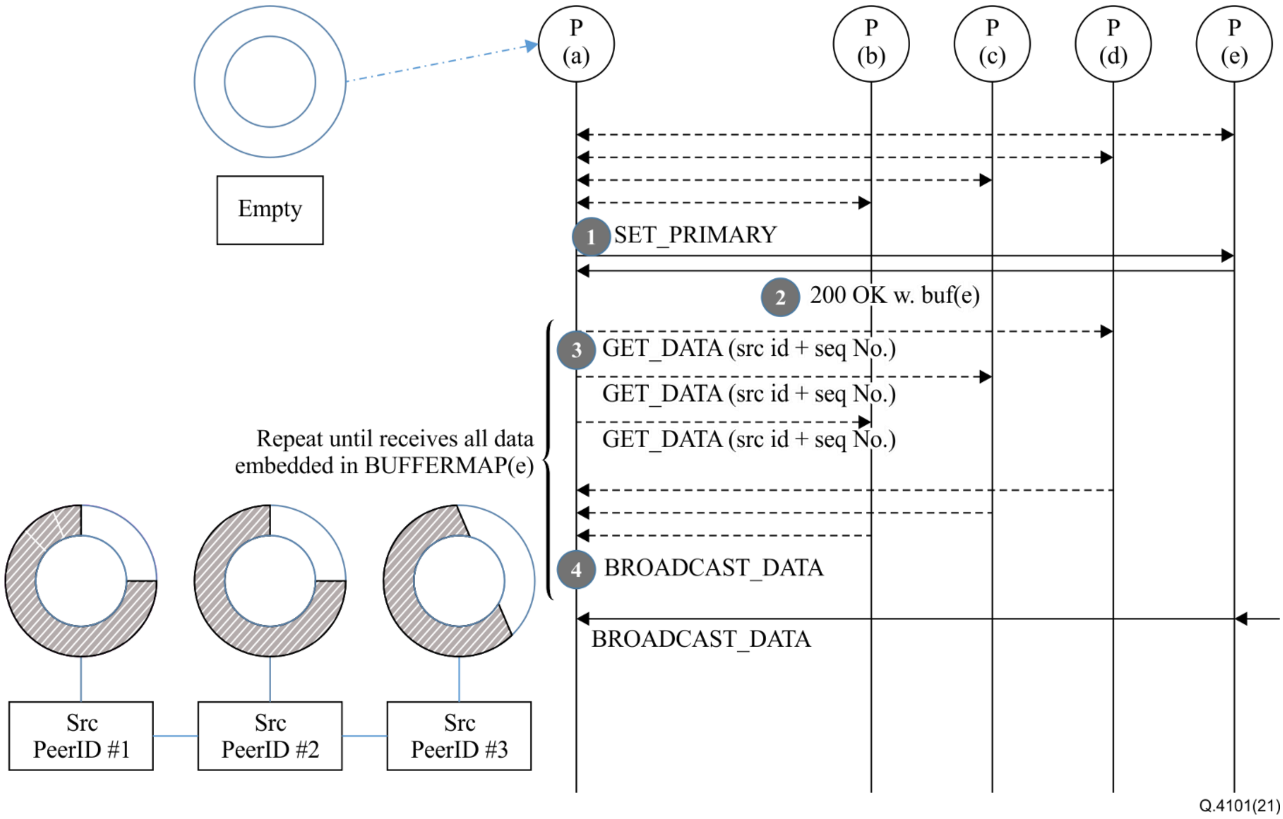


Figure 8-7 – Procedures for filling the cache buffer of a newly joining peer

① *P(a)* selects *P(e)* as a primary path among the candidate peers and sends a SET_PRIMARY request message. In this case, buffermap is not included.

② *P(e)* includes its buffermap when sending a response message for a SET_PRIMARY request message to *P(a)*.

③ *P(a)* sends a GET_DATA request message requesting the data it needs to other candidate peers. In this process, *P(a)* specifies the source peer ID and the sequence number of the data packet for the required data.

④ Candidate peers *P(b)*, *P(c)*, and *P(d)* deliver the requested data. In this process, each peer transmits the data using the BROADCAST_DATA request message.

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.2012] Recommendation ITU-T Y.2012 (2010), *Functional requirements and architecture of next generation networks.*

[b-ITU-T Y.2206] Recommendation ITU-T Y.2206 (2010), *Requirements for distributed service networking capabilities*.

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, 2021