INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 MPEG2015/M37248
October 2015, Geneva, Switzerland

Source	**Canon Research Centre France (CRF), Telecom ParisTech (TPT)**
Status	**Proposal to DASH Core Experiment on Full-duplex HTTP**
Title	**Editorial comments on DASH FDH Working Draft**
Authors	Franck Denoual (CRF), Hervé Ruellan (CRF), Youenn Fablet (CRF), Frédéric Mazé (CRF), Cyril Concolato (TPT), Jean Le Feuvre (TPT),

1 Introduction

After the last MPEG#112 meeting, a 2nd Working Draft on DASH for Full Duplex Protocols was produced [1]. This draft describes an abstract protocol composed as a set of messages.
This contribution provides comments, section by section, on this draft to try to improve readability and better define the scope of this Part-6.

2 General comments

1. First, concerning the scope of this specification: current draft contains “server MUST”, “client MUST”: **it should be clarified if we want to standardize client and/or server behavior or rather focus on the messages**.
    a. Up to now, DASH did not specify server and/or client behavior but rather formats and/or messages. In FDH draft, for example in section 7.3.4, current text constrains the server behavior.
    b. **Open question**: do we include in this FDH specification’s scope the server and/or client behavior or should we just focus on FDH messages?

2. Moreover, we should clarify the terminology and the way to define our protocol, talking about defining an API is confusing, indeed we are not defining functions on the client or the server but rather messages exchanges between clients and servers

3. We’re not sure about the relevance of the section related to the abstract protocol specification: a more concise part would be better to reach more rapidly the use of DASH FDH on concrete protocols (since abstract part has many redundancies with each specific binding).

4. Fundamentally, as such WebSocket is not an “HTTP-based protocol”, this is rather a TCP-based protocol compatible with HTTP-based infrastructure (through the upgrade mechanism).
   The working draft should avoid such misuse of terms.
   We propose replacing “HTTP-based” by “HTTP-compatible” everywhere in the document (including the title “Part 6: DASH over Full Duplex *HTTP-compatible* Protocols (FDH)”).

Considering that for HTTP/2 the only normative element defined in the specification is a new header, the length of the specification (30 pages) seems unnecessary. The group should consider making a very concise specification defining HTTP/2 only and a separate specification for WebSocket only.

3 Specific comments, section by section

3.1 Section 2: Normative References

Replace:
"*IETF Approved Working Draft, Hypertext Transfer Protocol version 2 (draft-ietf-httpbis-http2-17), February 2015*"

With

"*IETF RFC 7540, Hypertext Transfer Protocol Version 2 (HTTP/2), May 2015*"

3.2 Section 3.1: Terms and definitions

In this section, we should distinguish concepts (like "server push" and "push strategy") from parameters and/or types for this specification (like "push directive", "push acknowledge").

In 3.1.1, the term "Full Duplex HTTP" is never used. We suggest removing this definition (and any other definition not used).

In 3.1.2: HTTP/2 could be defined as:
Version 2 of the HTTP protocol, as defined by the IETF in RFC 7540.

As well, the 3.1.8 on WebSocket could be:
The WebSocket protocol, as defined by the IETF in RFC 6455.

In 3.1.4: Push Acknowledge
Should we generalize (or rename?) by saying that it can be:

- Indeed an acknowledgement from a server to a client's request for server push
- But also an announcement from a server of a push policy to be used (even when no client's request for push; i.e. server-initiated push)

We could rewrite into:
A response modifier, sent from a server to a client, which enables a server to state the push policy used when processing a request.

In 3.1.5: Push Directive:
A request modifier, sent from a client to a server, which enables a client to express its expectations regarding the server's push strategy for processing a request.

In 3.1.6: push strategy:
Current definition still considers the possibility of MPD push, even if there is no description/example in the specification. => should we remove or keep this part:
It was agreed during MPEG#112 [3] that "*MPD push was not going to be included in the initial solution. The existing mechanism for MPD update was considered sufficient*". We should clarify what it is allowed (WebSocket binding seems to use it).

We think “agreed upon by the client and server” should be removed since it is indication from the clients, and may not be agreed or applied by server.

We propose the following rewording for push strategy:
a segment and/or MPD transmission strategy that defines how segments or MPDs may be pushed from a server to a client.

In 3.1.7: Server Push
Same comment as above on segment and MPD.

3.3 Section 4: Introduction

Not sure, we should describe the upgrade to HTTP/2 or WebSocket: we can consider the full-duplex connection is already established (this is out of DASH scope).
We propose to modify the figure 1 to simplify with common behavior; i.e. having a loop on segment request instead of MPD request.
Note that on response to MPD request by the server, we could also illustrate the fast start possibility by adding a push strategy to the mpd file. As well a push of init segments from the server to the client could be added.
Another alternative is to simply remove this figure since a bit redundant with figure 2 in section 6.1.
The current draft seems to assume a specific architecture on the client (and maybe on the server) with some intermediate component between the DASH access engine and the network, realizing some abstraction of the actual protocol used below (HTTP/2 or WebSocket). A diagram describing such architecture would be useful to understand the model used in the specification and in particular, what is normative and what is not. We suggest adding such diagram in the introduction, similar to the Figure 2 of Part 1.

3.4 Section 5: Specification structure

We propose the following formulation to replace the current one:
“This specification defines an abstract protocol through a set of messages between a client and a server in order to drive the delivery of an MPEG DASH media presentation over full-duplex HTTP-compatible protocols. This set of messages is described in a first section and bindings to specific protocols are described, respectively for HTTP/2 and WebSocket in section 7 and section 8.”

This section could also be merged with introduction if finally figures 1 and 2 were merged (since also redundant).

3.5 Section 6.1: FDH Message flow

Replace the following sentence:
“The APIs defined are the absolute minimum changes required to DASH client and the server to take advantage of the additional useful features in a full-duplex HTTP based protocol”

With:

“This abstract specification defines the minimum set of messages that can be exchanged between a DASH client and a server to take advantage of the additional useful features in a full-duplex HTTP compatible protocol”.

In addition, on the figure describing the message flow, we propose to remove the upgrade from HTTP/1.1 to the bidirectional protocol to focus only on the DASH part. Connection establishment is out of scope (idem for description text). Additionally, the current diagram is

wrong. In case of upgrade, the client does not issue 2 separate messages, the upgrade is done as part of the request for the MPD.
To simplify, we may not put the list of possible pushDirectives in the request for segment #i.
At the bottom of the figure the push of the MPD update should be removed (or at least clarified) since it was decided during MPEG#112 [3] that current MPD update mechanisms were sufficient.

3.6 Section 6.2: Protocol scheme Identifiers

We propose to remove this section and rather consider a naming of the pushType that integrates this version number (for example: URN-based naming like: "urn:mpeg:dash:fdh:2015:push-XXXX").
In case this section is kept, a forward reference to where the identifiers are actually used would help the reader.

3.7 Section 6.3: Data Type definitions

Replace (or even remove):
"Details for implementing these primitives for a given concrete protocol binding may be found in the section of this specification defining that binding."

With:
"The implementation of these primitives for a given concrete protocol binding may be found in the section of this specification defining that binding."

On data types in Table 2:

- The BinaryObject type does not seem to be used except to redefine data types already defined in DASH: MPD and Segment. It may be useless to redefine those. We then suggest to remove BinaryObject data type plus MPD and Segment.
- If the section 6.2 is removed, then remove SchemeID parameter.
- The pushAck data type could be generalized into: "A response from the server to the client that contains the value describing the push strategy used by the server in response to a client request. See Table 3 for valid values of this type."
- The pushTemplate is more a new kind of pushDirective (pushType) than a new data type: could be removed from the table 2.

3.7.1 Section 6.3.2 on definition of pushDirective:

Note: This section does not contain any normative statement. The definition of a push directive is not formal. We suggest using a clear syntax for defining what a pushDirective is, such as a BNF grammar.

The term "approved" doesn't seem appropriate for a protocol specification.
Replace:
"This section provides push directives for approved push strategies"

With:
"This section provides push directives for push strategies defined in this specification"

Replace:
"We define the push strategy as a type (PushType), followed by a payload (PushParams)."

With:
"A pushDirective is defined as a type (pushType), optionally followed by a payload (PushParams)."

Indeed, push-none and push-fast-start don't have any parameters.

We suggest removing the inclusion of the scheme identifier. It is preferable, if versioning is needed, to have a pushDirective identifier integrating a version number (for example a URN as proposed in comments to section 6.2).

3.7.2 Section 6.3.3 on pushAck definition:

As for pushDirective, pushAck contains a type and optionally a payload.
In addition, the definition should reflect that PushAck can be used as an acknowledgment of PushDirective but also as an indication from the server of the Push Strategy it used.

3.7.3 Section 6.3.4. on pushTemplate

PushTemplate should be defined as specific kind (pushType) of pushDirective. The exact syntax remains to be defined (ex: URI template as defined by IETF) as well as the way to specify the values:

- Single value
- Range of values
- List of values
- Values pairs in case of multiple template parameters to vary.

This section should at the end include a reference to the informative section 13.

3.7.4 List of defined push strategies (Table 3) at end of section 6.3.

Table 3 should be updated with the push-fast-start pushType as proposed in contribution m36966.
We're wondering where this table should be placed:

- inside the section 6.3.2 that defines the pushDirective?
- Create a dedicated section on "defined push strategies for FDH"?

Moreover, defining both client and server side interpretation of the pushDirectives may improve understanding and would be justified since these can be used in client to server messages (the pushDirective parameter) but also in server to client messages (the pushAck parameter).
As a consequence, the different pushDirective shouldn't be defined only as request but rather as an indication of the push strategy.
The (tentative) resulting table below considers the above comments (alphabetical order):

PushType	**PushParams**	**Description**
push-fast-start	N/A	Indication that initialization data is considered for push to help client starting more rapidly. A server receiving such push directive may push what it considers the most appropriate for the client. A client receiving such push directive is informed that server intends to push initialization data.
push-next	K:Number	Indication that the the next K segments, using the requested segment as the initial index are considered for

		push. A value of 0 means that the server may elect to push indefinitely*. A server receiving such push directive may push consecutive segments to the requested one. A client receiving such push directive is informed that server intends to push the next segments consecutive to the requested one.
push-none	N/A	Indication that no push should occur. A server receiving such push directive should prevent from pushing. A client receiving such push directive is informed that server does not intend to push.
push-template	S:String	Indication that some segments as described by the URI template S are considered for push. A server receiving such push directive may use it to identify some resources to push. A client receiving such push directive can be informed on the resources the server intends to push.
push-time	T:Number	Indication that the next segments until the specified segmentTime (presentation time of the first frame) of a segment exceeds time T, beginning with the requested segment, are considered for push. A value of 0 means the server may elect to push indefinitely*. A server receiving such push directive may push a given duration of media segments. A client receiving such push directive is informed that server intends to push a given duration of media segments.

*Should the infinite push be kept? (push-next with K=0 or push_time with T=0).

3.8 Section 6.4: Message definitions

3.8.1 General comment

The use of the "preconditions", "post-conditions", "exceptions" and "errors" is unclear. We suggest removing them, as they contain no normative statements. If they are not removed, we suggest clarifying the behaviors in normative terms.

3.8.2 Section 6.4.1 on MPD request message

This section should include input text from contribution m36966 as agreed during the last conf call on October 7th, 2015.
This text is provided here for convenience:

The MPD request message initiates the request for a DASH MPD file. A push directive is allowed on the MPD request.

- Message Name: get_mpd

- Supplied Arguments

Argument Name	Argument Type	Description
mpd_uri	URI	The full URI for the MPD being requested
push_directive	PushDirective	A push strategy for suggesting server to anticipate the sending of initialization data. A push directive set to “push-none” explicitly indicates that the client does not intend to use push for fast start. (See Table 3)

- Preconditions
  - None

- Postconditions
  - The MPD request is initiated and pending all requested new_mpd messages are sent from the server to the client. The new_mpd message indicates that the server has responded with a requested MPD.

  - A push acknowledgment in the corresponding new_mpd message may indicate that server understood and applied the pushDirective indicated by the client.

- Errors/Exceptions
  - None

Moreover, the “pending all requested new mpd messages” in the first post condition requires clarification: In HTTP/2 there may be only one; is it then related to WebSocket?

3.8.3 Section 6.4.2 on Segment request message

We suggest removing the scheme parameter from the get_segment message. We think that, if needed, having a push type indicating the version of the protocol would provide versioning and extensibility.

Moreover, we may authorize more than one PushDirective in this message, for example to indicate preferred Push Strategies (see proposal from m36966).

3.8.4 Section 6.4.3 on MPD received message

Following the inclusion of the new push-fast-start, a pushAck should be included as optional parameter of the new_mpd message. The contribution m36966 provides the following text for that purpose:

This message represents the server's response from a previous get_mpd message sent by the client.

- Message Name: new_mpd

- Supplied Arguments

Argument Name	Argument Type	Description
mpd	MPD	The MPD returned by the server
push_acknowledge	PushAcknowledge	The push strategy that the server will follow

- Preconditions
  - The client requests an MPD by sending the *get_mpd* message.

- Postconditions
  - The client is ready to parse the received MPD.
  - The client is informed through the push_acknowledge parameter either on resource selected by server for fast start or on the fact that no push is done.

- Errors/Exceptions
  - None

3.9 Section 7.1: General Method

3.10 Section 7.2: System Architecture for HTTP/2

Should we keep this section (does not bring so useful information on FDH messages)?

If we decide to keep it:
Replace:
"Unlike HTTP 1.0/1.1 streaming, in HTTP/2 the server (origin or cache) can actively push segments (or MPDs) to the client (or the CDN) as soon as they are generated…"

With
"Unlike HTTP 1.0/1.1 streaming, in HTTP/2 the server (origin or cache) can actively push segments ~~(or MPDs)~~ to the client (or the CDN) as soon as they are generated"

The figure 4 should move in example sections: Informative annex of section 11.
Here,we should go directly on mapping of FDH messages onto the HTTP headers.
Some comments only duplicate what is said in HTTP/2 spec: should be removed.

Section 7.2.1 should include the generic header definition from contribution m36966 as agreed during last conf call on 7th of October.

Replace:
DASH-PUSH:scheme=<SchemeID>,type=<PushType>,params=<PushParams>

With:
"Accept-Push-Policy" ":" <unique_name> [";" <PushParams>*]

Note that the <unique_name> above would have to be discussed: could be pushType, could be pushType plus a version number, could be a URN… and PushParams are specified as defined in Table 3.

Section 7.2.2:
We propose for this section to align the HTTP/2 binding of the pushAck with the Accept-Push-Policy proposed above as follows (using a different header name);

"Push-Policy" ":" <unique_name> [";"<pushParams>*]

Where unique_name is TBD and PushParams are specified as defined in Table 3.

Section 7.2.3 is useless: this is standard HTTP/2 behavior.

Section 7.2.4 is useless: in case of a previously promised resource that would be unavailable, natively in HTTP/2, a server can send a RST_STREAM for the stream identifier it reserved with the PUSH_PROMISE..

3.11 Section 7.3: Message Bindings

3.11.1 Section 7.3.1 on MPD request message

We can now add (since we allow a PushDirective for fast start use case):
"The push directive parameter, if present, MUST be provided in the manner specified in section 7.2.1."

3.11.2 Section 7.3.2 on segment request message

OK

3.11.3 Section 7.3.3 on MPD received message

It was agreed during MPEG#112 [3] that MPD push was not going to be included in the initial solution. The existing mechanism for MPD update was considered sufficient.

Then, replace:
"An HTTP/2 server sends this message either in response to a client MPD request message (as in section 7.2.1) or as a server-initiated MPD update."

With
"An HTTP/2 server sends this message ~~either~~ in response to a client MPD request message (as in section 7.2.1) ~~or as a server-initiated MPD update~~."

There is a strong requirement on server behavior in this section:
"In the case where the server sends this message in response to a client request, the server SHALL return the MPD as the body of the HTTP response, just as it would when communicating with an HTTP 1.1 DASH client."

That could be rewritten as follows (if we decide to focus on FDH messages and consider client/server behavior as out of scope):
"This message shall contain the requested MPD in the body of the HTTP response (as in DASH over HTTP/1.1)."

In addition, we should add following paragraph:

"If a push directive was specified in the client request, and the server supports the push directive, it MAY insert a push acknowledgement, as described in section 7.2.2 in this message. Otherwise it MAY contain a push acknowledgement to inform the client on default push strategy used by the server."

3.11.4 Section 7.3.4 on segment received message

Again, this section contains strong requirements on server behavior which until now remained out of scope of DASH specifications: do we want this FDH specification to constrain the server behavior or should we just provide guidelines to optimize the delivery when using FDH messages?

Depending on the above choice, this section should be rewritten.

Moreover, we could remove all references to "MPD update" in this section.

3.11.5 Section 7.3.5 on segment cancel message

The Segment Cancel message can be mapped directly to RST_STREAM frame. Then, a simple sentence like the one below should be sufficient (instead of current paragraph):

"It is possible for a client to cancel a push sequence by sending RST_STREAM frames each referencing the promised stream identifiers as specified in HTTP/2."

3.12 Section 8: Protocol Binding for WebSocket

This section has not been deeply reviewed.

Should HTTP header value be strictly aligned with JSON Name Value Pair?
- We think it might be better.

3.13 Section 9: Considered use cases

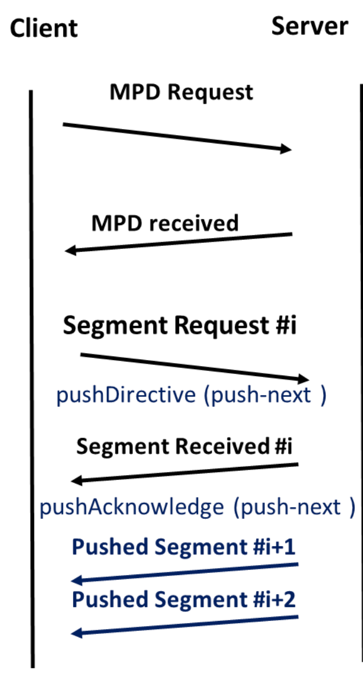
no specific comments

3.14 Section 10: Examples of abstract protocol client/server behavior

Maybe figures like the one on figure 4b could be provided to describe each behavior.
We propose the following list of examples, organized into good cases and error/mismatch cases:

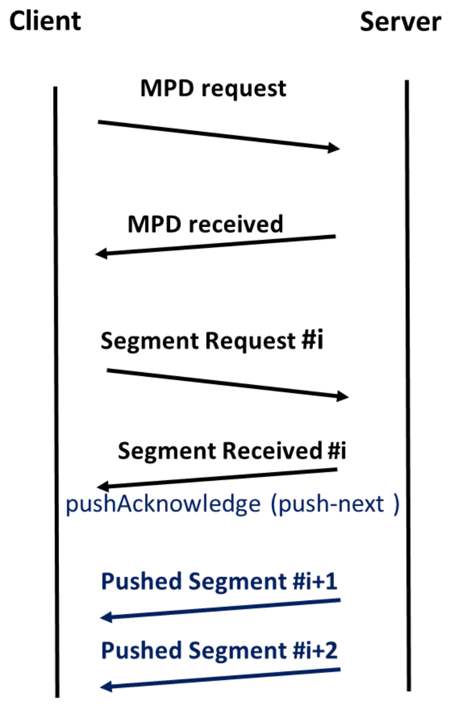
<u>Define a sub-section for "good cases"</u>
These examples would correspond to good cases where client and server both support FDH messages and server implements one or more push strategies.

- Keep the 11.1 Example of a **client-initiated** push request **on segment** (push-next or time or template)



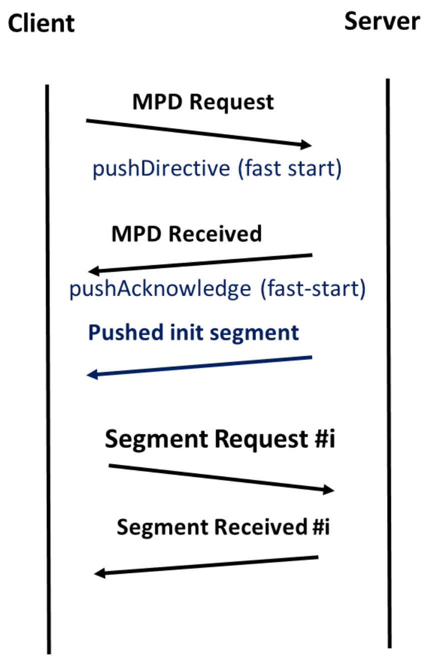
Client asking for segment push

- Add a new example for a server-initiated push request on segment



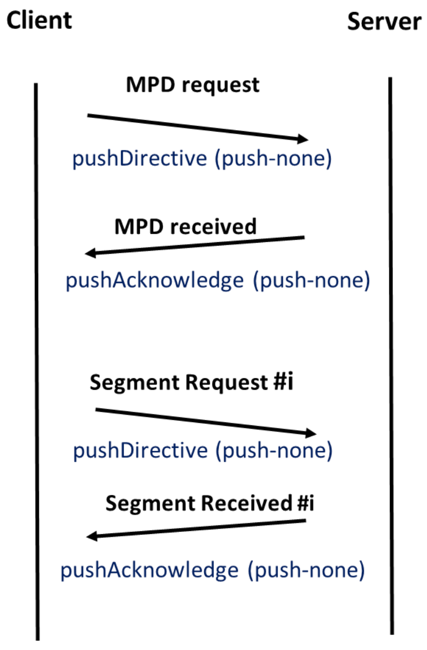
Server-initiated segment push

Add a new example for a client-initiated push request when downloading MPD (fast-start use case)



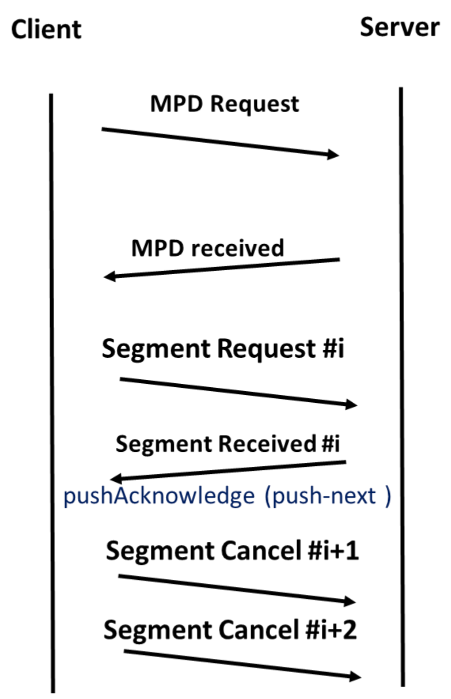
Client-initiated fast start

- Add a new example for a client indicating that it does not want any push at all (use of push-none)


Client
Server
MPD request
pushDirective (push-none)
MPD received
pushAcknowledge (push-none)
Segment Request #i
pushDirective (push-none)
Segment Received #i
pushAcknowledge (push-none)

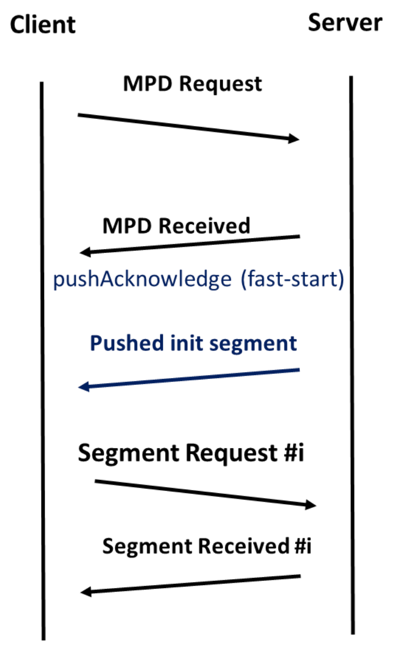
Client indicating no-push at all

- Keep the 11.3 example of cancelling a push request: server took the initiative to push, client does not accept the push by immediately sending a segment cancel message



Push cancel by client

- Add a new example for server-initiated fast start: no directive from client, server takes the initiative of pushing initialization data and informs client in the *new_mpd* message.

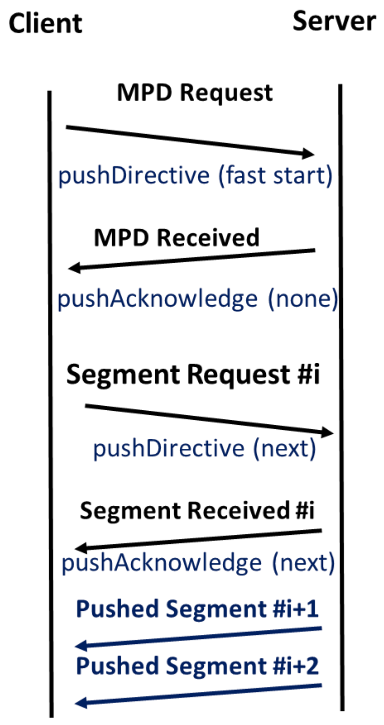


Server-initiated fast start

Define a sub-section for error cases:

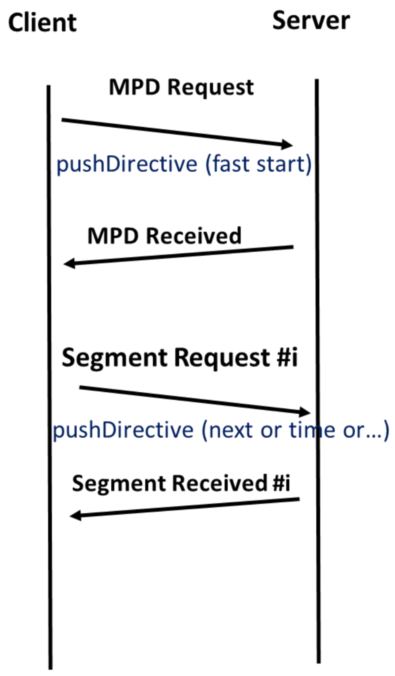
These examples would correspond to error or mismatch cases where server does not support FDH messages or server does not implement the pushDirectives indicated by the client or pushes resources that do not match client expectations.

- Add an example of a client-initiated fast-start request with apush capable serverbut that does not support push-fast-start.



No fast start at server

- Keep the 11.2 example of a client-initiated push request on segment with a server that does not support push at all.



No push at server

3.15 Section 13: Examples of Push Template

We propose to insert the examples from m36966 (using URI template from IETF)

13.1 Example of push template with segment number addressing

"../rep1/segment{**Number**}.mp4"; {2-4} to push segment**2**, segment**3** and segment**4** from Representation "rep1".

13.2 Example of push template with time-based addressing

"../rep1/segment{**Time**}.mp4" ; 1000: 10000 to push all segments between 1 and 10 seconds or "../rep1/segment{**Time**}.mp4" ; : 10000 to push all segments between requested segment and 10 seconds later.

13.3 Example of push template with multiple representations

To push rep1/segment2.mp4 and rep2/segment3.mp4 and rep3/segment4.mp4:
../rep{**id**}/segment{**Number**}.mp4; (1,2) : (2,3) : (3,4)
Where {id} and {Number} template parameters are successively replaced with the couple given as arguments: (id=1, number=2) and (id=2, number=3) and (id=3, number=4).

4 Conclusion

We would like DASH experts to consider the above comments and proposals to generate an improved version of the FDH working draft.

5 References

[1] w15532 "*Working Draft for 23009-6: DASH over Full Duplex HTTP-based Protocols (FDH)*" by I. Bouazizi, K. Streeter and V. Swaminathan, MPEG#112, Warsaw, July 2015

[2] "*Hypertext Transfer Protocol version 2 (HTTP/2)*" RFC 7540, by M. Belshe, R. peon and M. Thompson, May 2015; available at https://tools.ietf.org/html/rfc7540

[3] m36858 "*Descriptions of Core Experiments on DASH-FDH CE*", Warsaw, July 2015.