CoRE
Internet Draft
Intended status: Standards Track
Expires: August 2019

A. Bhattacharyya
S. Agrawal
H. Rath
A. Pal
B. Purushothaman
TATA CONSULTANCY SERVICES LTD.
February 6, 2019

Adaptive RESTful Real-time Live Streaming for Things (A-REaLiST)
draft-bhattacharyya-core-a-realist-02

Abstract

This draft presents extensions to Constrained Application Protocol (CoAP) to enable RESTful Real-time Live Streaming for improving the Quality of Experience (QoE) for delay-sensitive Internet of Things (IoT) applications. The overall architecture is termed "Adaptive RESTful Real-time Live Streaming for Things (A-REaLiST)". It is particularly designed for applications which rely on real-time augmented vision through live First Person View (FPV) feed from constrained remote agents like Unmanned Aerial Vehicle (UAV), etc. These extensions provide the necessary hooks to help solution designers ensure low-latency transfer of streams and, for contents like video, a quick recovery from freeze and corruption without incurring undue lag. A-REaLiST is an attempt to provide an integrated approach to maintain the balance amongst QoE, resource-efficiency and loss resilience. It provides the necessary hooks to optimize system performance by leveraging contextual intelligence inferred from instantaneous information segments in flight. These extensions equip CoAP with a standard for efficient RESTful streaming for Internet of Things (IoT) contrary to HTTP-streaming in conventional Internet.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 6, 2019.

Copyright Notice

Table of Contents

1. Introduction

IoT emerged to facilitate exchange of frequent-but-small sensory information amongst numerous constrained sensors [IOT-ISOC][RFC7452]. However, recent trends in industry and research community realize the importance of live visual data as important sensory information. There are many discourses available to support this observation [Murphy]. Live First Person View (FPV) from Unmanned Aerial Vehicles (UAV) and dumb robot terminals are being used for futuristic remote control and actuation applications for Augmented Reality (AR), Visual Simultaneous Localization and Mapping (VSLAM), UAV based surveillance, etc. Efficacy of these applications depends on resource-efficient, low-latency, yet high QoE transfer of the FPV over the Internet (or IP networks in general). Contrary to the traditional video streaming applications, the UAV-like end-points (henceforth referred as 'video producer') that capture and transmit the FPV are resource constrained devices. Moreover, the producer may work in a lossy environment marred with fluctuating radio connectivity and disruptions due network congestion.

The QoE considerations of the video rendering unit (henceforth referred as 'video consumer') for these applications are quite different from traditional applications. For example, in case of

highly delay sensitive AR applications, a human brain may not tolerate a noticeable video freeze or delayed reception, which might have been overlooked for usual content delivery service like a YouTube video. Such delay may result in wrong actuation. For example, delayed FPV from a UAV may lead to wrong control commands leading to catastrophic consequences. In addition, the communication should be as light-weight as possible to optimize the usage of on-board computing and energy resources of the UAV. So, real-time video transmissions for IoT applications require special treatment

[Pereira]. However, as revealed through a detail analysis of the state-of-the-art in the next section, the existing solutions do not address such special requirements. This draft attempts to bridge this important gap by extending CoAP [RFC7252].

To realize its purpose, the A-REaLiST architecture relies on [RFC7967] and adds few new header options which, taken together, can be conceived to form a conceptual 'Stream' extension on CoAP (Fig. 1).

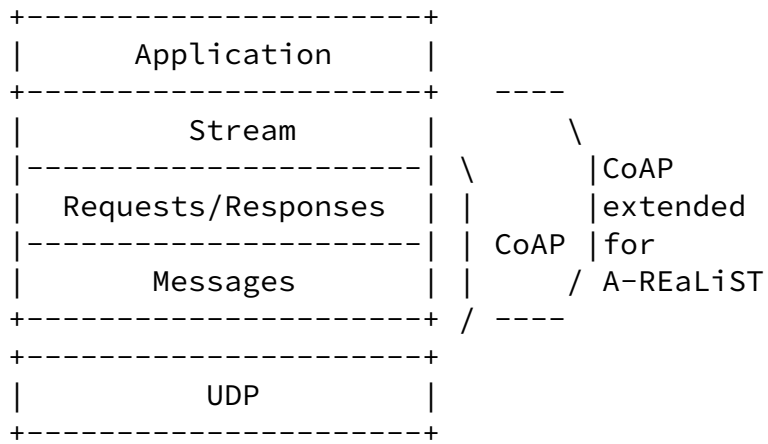

Figure 1: Abstract extended layering of CoAP for A-REaLiST with the conceptual layer for streaming.

Though primarily designed for video streaming, these extensions can also be used to allow streaming of time-series information on CoAP.

Note: Block-wise transfer [RFC7959] is a standardized extension to

CoAP for transferring large application data. The cited use case for this is to perform firmware upgrade for a large number of constrained devices. Block-wise transfer is primarily concerned with reliable delivery of information. It works in synchronized manner. If a message remains unacknowledged despite retransmissions then the whole exchange is cancelled. So, it is not suitable for real-time delivery [GIoTS] which is requirement for many time-series information streams including video.

2. Revisiting CoAP

2.1. Some Interesting Aspects of CoAP

(i) CoAP allows both confirmable (CON) and non-confirmable (NON) messaging.

(ii) CON mode enables CoAP with an option for reliable RESTful delivery like HTTP [RFC2616]on TCP. On the other hand, intelligent use of No-Response option [RFC7967] along with NON mode can create an RTP like best-effort messaging on UDP.

(iii) Context based switching between the reliable and best-effort semantics can be executed from the end-application level. This way an optimum balance between reliability delay-performance can be maintained to improve the overall Quality of Experience (QoE).

(iv) The base CoAP specification is inherently designed for resource constrained devices. Hence, a streaming protocol using the stateless RESTful semantics on CoAP makes the solution inherently lightweight. So, unlike conventional approach the designers can use a single stack that is equally efficient for sending the small data out of sensors, as well as, infinite visual stream.

2.2. The Prevalent Approaches for Streaming over Internet

The two prevalent approaches for streaming over the Internet are as

below.

First approach is to send the information segment over HTTP which uses the reliability feature of the underlying Transmission Control Protocol (TCP) transport. In this case TCP state-machine puts more emphasis on reliable delivery of segments rather than maintaining the real-time deadlines. However, this is right now the prevalent approach as it treats video and other streams as general Internet traffic. So, streaming can seamlessly co-exist with the existing Internet architecture. Also, since TCP takes care of ordered delivery, the end-application does not need to worry about these matters.

The other approach is to use a specialized protocol like Real-time Transport Protocol (RTP) [RFC3550]. It treats video and other real-time streams as a special type of traffic. To ensure real-time delivery, the data is delivered in best-effort manner on top of UDP. So, reliable delivery is undermined.

2.3. CoAP as the Best of Two Worlds

It can be conjectured, tallying the above with previous section, that CoAP inherently imbibes the functional features from HTTP-on-TCP (reliable delivery) and RTP-on-UDP (best-effort delivery). Further CoAP allows the switching between these two seamlessly just by maneuvering the header options.

3. The Approach behind A-REaLiST

The design stems from the principles of "progressive download" on top of the RESTful request/response semantics of CoAP. The "producer" chunks the continuous information stream into segments as per the agreed maximum payload size suggested in [RFC7252]. Each chunk is transmitted as a CoAP request to a given resource at the "consumer". This draft provides the necessary header extensions that enable the "consumer" to maintain the sequence of the information segments in time and space.

3.1. Optional Context Aware Semantic Switch

Before forming the CoAP message for each segment, the streaming application may use a real-time analytics module (henceforth

referred as 'analytics module') which may provide inference to the "Stream" layer to decide the exchange semantics for the current segment. The message is sent reliably (CON message) or as best-effort (NON message with No-Response option) based on the segment's information criticality. Criticality is measured in terms of importance of the segment-content in reconstruction of the frames at the consumer. However, determination of criticality can be done on many aspects involving several application features like the source encoding type, the rendering logic at the consumer, etc. This way the over-all balance between QoE and resource-consumption may be maintained. Fig. 2 explains the idea with conceptual blocks. The overall concept and its efficacy has been explained with experimental results in [Wi-UAV-Globecom]

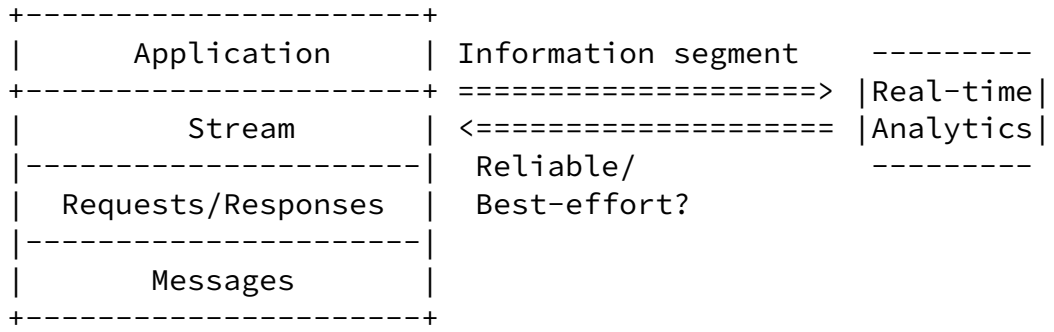

Figure 2: Illustrating the concept for context aware switching

Some examples are:

Example-1: Temporally compressed videos like MPEG consist of Group of Pictures (GoP) which comprises I-frames (Intra-frames) or key-frames, P-frames (Predicted frames) and B-frames (Bidirectional frames). Out of these 3 types of frames I-frames are most critical in terms of synchronizing with the GoP at the receiver end for successful rendering. So, an analytics module at the "video producer" end may infer each information segments of I-frames as critical and send those segments reliably. The segments corresponding to P and B frames may be transferred as best-effort requests.

Example-2: Let us consider a Motion JPEG (MJPEG) stream. In this case all the frames are independent JPEG frames and there is no temporal compression. The analytics module may treat the segments

containing MJPEG meta-data for each frame as critical segments and transfer them through reliable messaging. Rest of the segments may be transferred as best-effort requests. An intelligent rendering engine at the "consumer" application may compensate for / conceal any possible loss of non-meta-data (non-critical) segments using the reliably received meta-data and rest of the non-meta-data segments received through best-effort. This way high QoE can be ensured despite reduced resource usage.

4. The Options Introduced

To achieve the purpose of the Stream layer, three new protocol header options have been proposed as below:

1) Stream_info: Consumes one unsigned byte. It maintains the stream identity and indicates the present phase of exchange. It is both a request and response option. It has two fields. The 3-LSBs indicate the state of exchange (Stream_state) and 5-MSBs indicate an identifier (Stream_id) for the stream. The identifier remains unchanged for the entire stream. So,

```
Stream_id = Stream_info >> 3;
Stream_ state = Stream_info & 0x7.

Interpretation of Stream_state bits are :
000=> stream initiation (always with request);

001=> initiation accepted (always with response);
```



```
010=> initiation rejected (always with response);

011=> stream re-negotiation (with request or response);

100=> stream ongoing.
```

Note: While Stream_id field enables to uniquely identify an information stream, it may also be used by an application to relate the context of different sub-streams (may be with varying QoS) and produce a combined rendering at the consumer-end.

2) Time-stamp: It consumes 32-bit unsigned integer. It is a request option. It relates a particular application information segment to the corresponding frame in the play sequence.

3) Position: It consumes 16-bit unsigned integer. It is a request option and MUST be accompanied with the Time-stamp option. It is a combination of two fields. The 15-MSBs indicate the "offset" at which the present segment is placed in the frame corresponding to the given timestamp. The LSB indicates if the current segment is the last segment of the frame corresponding to the given timestamp. Hence,
Last_segment = Position &0x01 ? True : False;
Offset = (Position >> 1).

No.	C	U	N	R	Name	Format	Length	Default
TBD	X		-		Stream-info	uint	1	(none)
TBD	X		-		Time-stamp	uint	4	(none)
TBD	X		-		Position	uint	2	(none)

Table 1: Option Properties

5. The Handshake and Exchange Semantics

As per the design considerations (in view of the scenarios conceived at present) video transfer is initiated by the "producer" which acts as the client.

Each segment is transmitted to the "video consumer" as a POST request. The Time-stamp and Position options help sequential ordering of the segments at the consumer.

Note: The design considerations are driven by the experiences drawn from the applications where live video feeds are transmitted from battery operated constrained "video producers" like UAVs and dumb robotic terminals, etc. For example, while a fixed infrastructure system is using streamed FPV feed from UAVs, there

may be situations where each time a UAV is low on resources (energy and computation, a new UAV with better state of resources (fresh battery, etc.) is commissioned. The overall operation becomes simple if the newly commissioned UAV readily starts its job by streaming to the same resource at the fixed infrastructure. It can be easily configured to determine whether the consumer is up and watching by observing the responses to the CON requests. In case the exchange is initiated by the consumer then whenever a new UAV is commissioned, the consumer has to re-initiate the request again.

5.1. Initial Negotiation

Initial negotiations for frame rate, video type, encoding details, etc., are performed by exchanging configuration scripts (cbor or json) over POST request. Exact format of the script is application dependent and is not part of this draft.

Fig. 3 illustrates the exemplary exchanges related to handshakes for connection initiation.

Note: All reliable transfers are in blocking mode. So, the producer MUST wait to send any further segment (critical/ on-critical) till the response is received for the critical segment. Please refer to Section 6 for suggested behavior in case a reliable transfer fails.

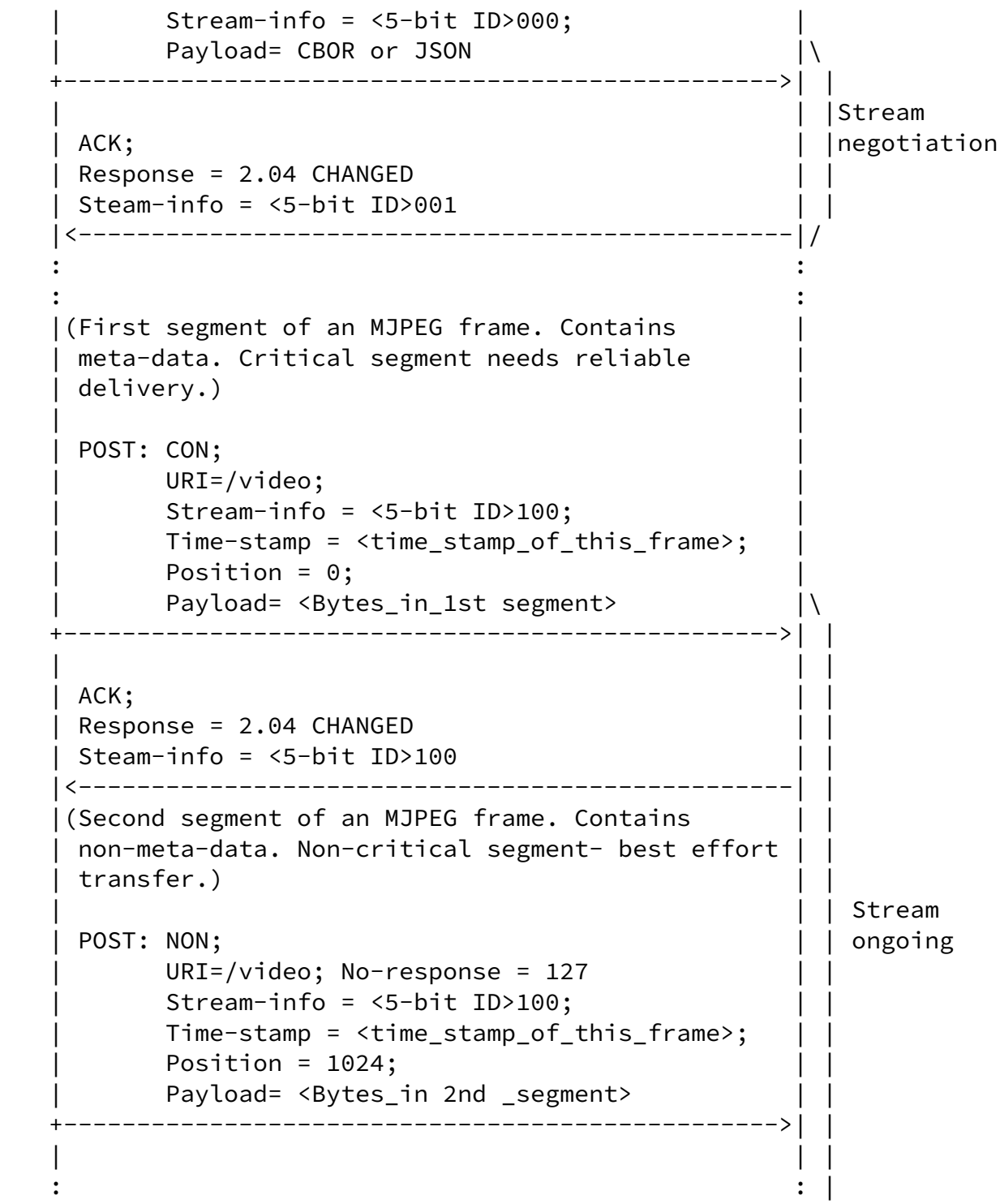
Stream-info = <5-bit ID>000;
Payload= CBOR or JSON
ACK;
Response = 2.04 CHANGED
Steam-info = <5-bit ID>001
Stream
negotiation
(First segment of an MJPEG frame. Contains
meta-data. Critical segment needs reliable
delivery.)
POST: CON;
URI=/video;
Stream-info = <5-bit ID>100;
Time-stamp = <time_stamp_of_this_frame>;
Position = 0;
Payload= <Bytes_in_1st segment>
ACK;
Response = 2.04 CHANGED
Steam-info = <5-bit ID>100
(Second segment of an MJPEG frame. Contains
non-meta-data. Non-critical segment- best effort
transfer.)
POST: NON;
URI=/video; No-response = 127
Stream-info = <5-bit ID>100;
Time-stamp = <time_stamp_of_this_frame>;
Position = 1024;
Payload= <Bytes_in 2nd _segment>
Stream
ongoing

Figure 3: Example showing successful negotiation of streaming parameters followed by transmission of video information and control. It is assumed that the segment size negotiated as 1024 at the initiation. So, the position of the 2nd block is 1024. Note the use of No-response option with NON request for the non-critical segment.

5.2. Renegotiation

The renegotiation phase may occur when the "consumer" does not agree to parameters proposed by the producer and proposes a modified set. This may happen when the consumer application may need a less frame-rate than what is proposed by the producer. So, the "consumer" may request a lower frame-rate and thereby avoid unnecessary traffic in the network. The reduction may also be driven by the processing load on the producer which is anyway a constrained device. So, if a consumer requests more frame-rate than what is initially proposed by the producer, then the producer may insist on the lower frame-rate. Renegotiation may also occur if, during a stream, the producer senses a change in the end-to-end channel condition and proposes a new set of best possible parameters that can be served to the consumer.

Note that, that the consumer is never allowed to exceed the limits advertised by the producer.

Fig. 4 illustrates exemplary exchanges for re-negotiation.

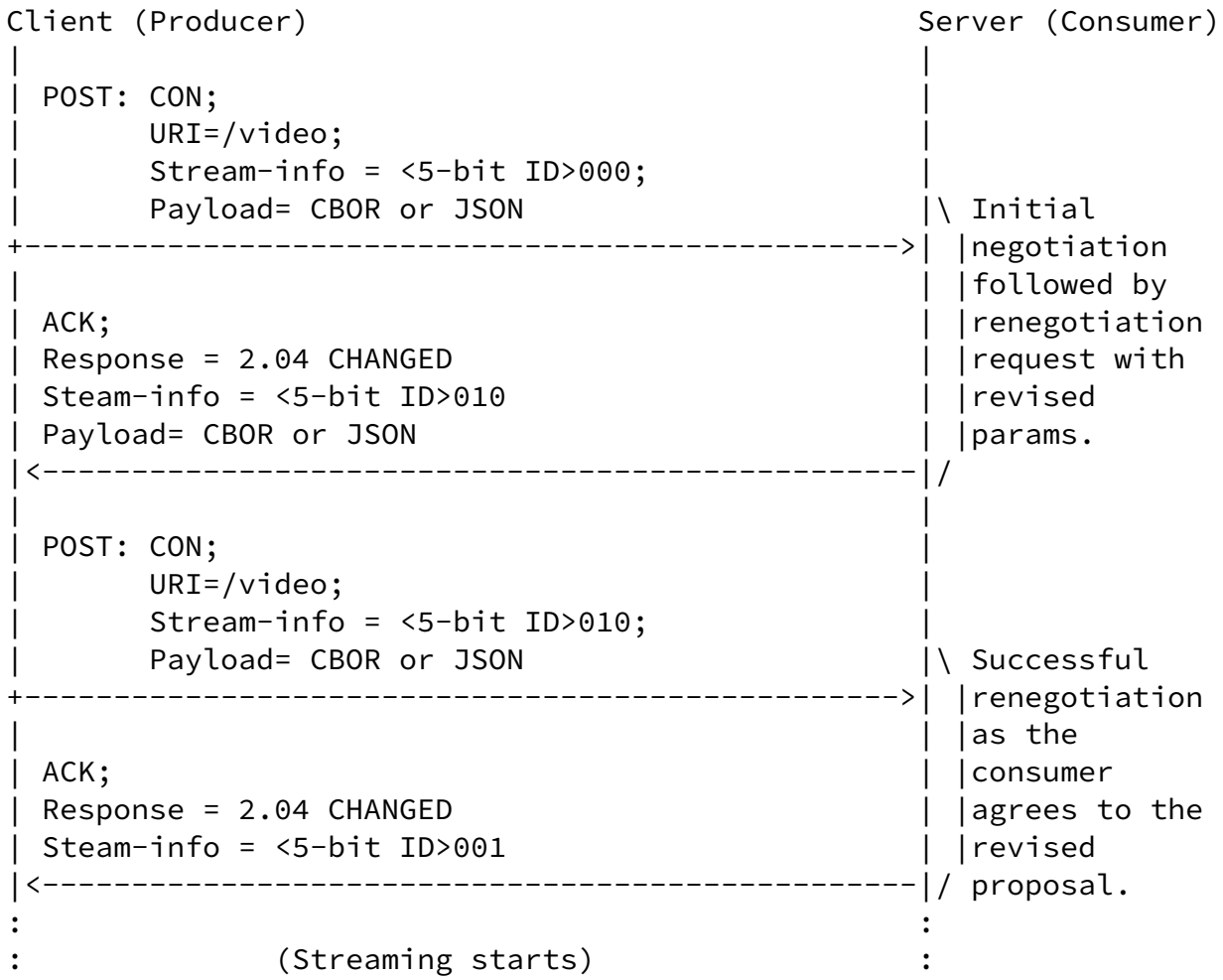

Figure 4: Example showing successful renegotiation of streaming parameters. Note the maneuvering of the Stream-info bit patterns.

Fig. 5 illustrates exemplary exchanges when a stream negotiation is unsuccessful. The accompanied script may provide hints to the reason for unsuccessful negotiations. A simple case of unsuccessful attempt may be observed if the resource on the "consumer" side is not ready. The exact formatting of the script is not in the scope of this draft.

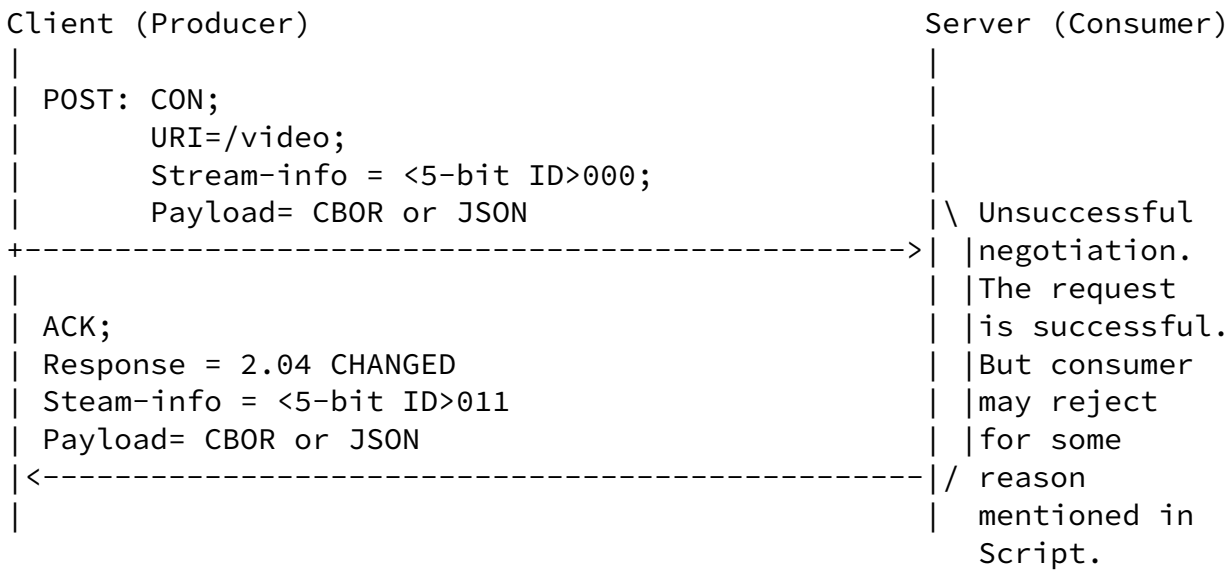

Figure 5: Example showing unsuccessful renegotiation despite successful response code against the initiation request.

6. Some Design Guidelines

6.1. Implicit Congestion Avoidance

The throughput and resource optimization for A-REaLiST depends largely on the best-effort delivery on UDP. Despite that the application designer can make A-REaLiST implicitly congestion aware and proactively avoid congestion. CoAP has a basic congestion avoidance mechanism which uses exponential back off to increase the timeout for retransmissions. However, that works only for CON messages.

The implicit congestion avoidance works like this: In case the producer fails to successfully transfer a critical segment of a frame within the MAX_TRANSMIT_SPAN as well as within MAX_RETRANSMIT [RFC7252] attempts, the producer drops transmission of rest of the segments in that frame and waits for the next frame to be ready. The rationale is, since the critical segment is not delivered, the consumer will fail to reconstruct this frame anyway. So, there is no point in clogging the network with rest of the segments.

6.2. Considerations for Consumer-side Rendering

While the critical segments are delivered reliably in a sequential manner, non-critical are delivered with best-effort in an open-loop exchange. Also, the whole frame can be dropped to avoid congestion. Hence, the application at the "consumer" end-point (server) needs to

deal with issues like out-of-order delivery, frame/segment loss, asynchronous segment arrival.

The issues mentioned above have been discussed in literatures [Perkins]. So the basic approach should be: Buffer till a critical time to iron out the jittery, out-of-order arrival of the segments, play out from the appropriate buffer at a constant rate determined by the frame-rate of the video. There may be intelligent algorithms to play-out with high QoE despite non-arrival of non-critical segments within the play-out deadline. This draft provides the hooks to create such designs. Reference architecture of the play-out mechanism is provided in [Wi-UAV-Globecom]. The play-out architecture leverages on the design assumption about the 'less-constrained' nature of the consumer in terms of memory and processor.

6.3. Determining the segment size

Size of the information segment in a CoAP message should be limited by the least possible MTU for the end-to-end channel. This is to ensure that there is no undesired conversation state at the lower layers of the protocol stack due to uncontrolled fragmentation leading to undesired explosion of traffic in the network. For IPV6 network, the MTU can be determined using Path MTU Discovery (PMTUD) [RFC8201] which bestows the responsibility of determining the path MTU on the end-points itself.

The size of the segment should be guided by the recommendations as specified in Section 4.6 of [RFC7252].

7. IANA Considerations

The IANA is requested to assign numbers to the three options introduced in this draft for inclusion in the "CoAP Option Numbers" registry as shown below.

Number	Name	Reference
TBD	Stream-info	Section 4
TBD	Time-stamp	Section 4
TBD	Position	Section 4

8. Security Considerations

This draft presents no security considerations beyond those in Section 11 of the base CoAP specification [RFC7252].

9. References

9.1. Normative References

[RFC7252]

Shelby, Z., Hartke, K. and Bormann, C.,"Constrained Application Protocol (CoAP)", RFC 7252, June, 2014.

[RFC7967]

Bhattacharyya, A., Bandyopadhyay, S., Pal, A., Bose, T., "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, August, 2016.

9.2. Informative References

[IOT-ISOC]

Rose, K., Eldridge, S., Chapin, L., "The Internet of Things: an overview", Internet Society, pp.1-50, October, 2015.

[RFC7452]

Tschofenig, H., Arkko, J., McPherson, D., "Architectural Considerations in Smart Object Networking", RFC 7452, March, 2015.

[Murphy]

Murphy, C., "Internet of Things: Are you underestimating video?", Available online:

http://www.informationweek.com/bigdata/bigdataanalytics/internetofthingsareyouunderestimatingvideo/a/d-id/1269508, June, 2014.

[Pereira]

Pereira, R., Pereira, E. G., "Video Streaming Considerations for Internet of Things", International Conference on Future Internet of Things and Cloud, pp. 48-52, August, 2014.

[RFC7959]

Bormann, C., Shelby, Z., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, August, 2016.

[GIoTS]

Dey, S., Bhattacharyya, A., Mukherjee, A., "Semantic data exchange between collaborative robots in fog environment: Can CoAP be a choice?", Global IoTS, pp. 1-6, June, 2017.

[RFC2616]

Fielding, R., Irvine, U.C., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June, 1999.

[RFC3550]

Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July, 2003.

[Wi-UAV-Globecom]

Bhattacharyya, A., Agrawal, S., Rath, H., Pal, A., "Improving Live-streaming Experience for Delay-sensitive IoT Applications : A RESTful Approach", accepted in Globecom (Wi-UAV workshop), Dec., 2018.

[Perkins]

Perkins, C., "RTP: Audio and Video for the Internet", Addison-Wesley, 2003.

[RFC8201]

McCann, J., et al., "Path MTU Discovery for IP version 6", RFC 8201, July, 2017.

Authors' Addresses

Abhijan Bhattacharyya
Tata Consultancy Services Ltd.
Kolkata, India

Email: abhijan.bhattacharyya@tcs.com

Suvrat Agrawal
Tata Consultancy Services Ltd.
Bangalore, India

Email: suvrat.a@tcs.com

Hemant Rath
Tata Consultancy Services Ltd.
Bhubaneswar, India

Email: hemant.rath@tcs.com

Arpan Pal
Tata Consultancy Services Ltd.
Kolkata, India

Email: arpan.pal@tcs.com

Balamurali Purushothaman
Tata Consultancy Services Ltd.
Bangalore, India

Email: balamurali.p@tcs.com