CoRE A. Bhattacha
Internet Draft S. Agr
Intended status: Standards Track H.
Expires: August 2019 A.
B. Purushoth
TATA CONSULTANCY SERVICES
February 4,

Adaptive RESTful Real-time Live Streaming for Things (A-REaLiST)

draft-bhattacharyya-core-a-realist-01

Abstract

This draft presents extensions to Constrained Application Protocol (CoAP) to enable RESTful Real-time Live Streaming for improving the Quality of Experie (QoE) for delay-sensitive Internet of Things (IoT) applications. The overall architecture is termed ''Adaptive RESTful Real-time Live Streaming for Thing REaLiST)''. It is particularly designed for applications which rely on real- augmented vision through live First Person View (FPV) feed from constrained agents like Unmanned Aerial Vehicle (UAV), etc. These extensions provide the necessary hooks to help solution designers ensure low-latency transfer of st and, for contents like video, a quick recovery from freeze and corruption wi incurring undue lag. A-REaLiST is an attempt to provide an integrated approa maintain the balance amongst QoE, resource-efficiency and loss resilience. I 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 convent Internet.

Status of this Memo

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 tr in industry and research community realize the importance of live visual dat important sensory information. There are many discourses available to suppor observation [Murphy]. Live First Person View (FPV) from Unmanned Aerial Vehi (UAV) and dumb robot terminals are being used for futuristic remote control actuation applications for Augmented Reality (AR), Visual Simultaneous Localization and Mapping (VSLAM), UAV based surveillance, etc. Efficacy of t applications depends on resource-efficient, low-latency, yet high QoE transf the FPV over the Internet (or IP networks in general). Contrary to the tradi video streaming applications, the UAV-like end-points (henceforth referred a '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 ' consumer') for these applications are quite different from traditional applications. For example, in case of highly delay sensitive AR applications human brain may not tolerate a noticeable video freeze or delayed reception, might have been overlooked for usual content delivery service like a YouTube video. Such delay may result in wrong actuation. For example, delayed FPV fr UAV may lead to wrong control commands leading to catastrophic consequences. addition, the communication should be as light-weight as possible to optimiz usage of on-board computing and energy resources of the UAV. So, real-time v transmissions for IoT applications require special treatment [Pereira]. Howe as revealed through a detail analysis of the state-of-the-art in the next se 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 a few new header options which, taken together, can be conceived to form a conceptual 'Stream' extension on CoAP (Fig. 1).


```
                +----------------------+
                |      Application     |
                +----------------------+   ----
                |        Stream        |       \
                |----------------------| \      |CoAP
                |  Requests/Responses  | |      |extended
                |----------------------| | CoAP |for
                |       Messages       | |     / A-REaLiST
                +----------------------+ / ----
                +----------------------+
                |         UDP          |
                +----------------------+
```

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

Though primarily designed for video streaming, these extensions can also be 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 pe 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 suita 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 lik [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 messagin UDP.

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

(iv) The base CoAP specification is inherently designed for resource constr devices. Hence, a streaming protocol using the stateless RESTful semantic CoAP makes the solution inherently lightweight. So, unlike conventional approach the designers can use a single stack that is equally efficient f 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 del of segments rather than maintaining the real-time deadlines. However, this i right now the prevalent approach as it treats video and other streams as gen Internet traffic. So, streaming can seamlessly co-exist with the existing In architecture. Also, since TCP takes care of ordered delivery, the end-applic 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 s 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 delive

and RTP-on-UDP (best-effort delivery). Further CoAP allows the switching bet
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 t
RESTful request/response semantics of CoAP. The ''producer'' chunks the cont
information stream into segments as per the agreed maximum payload size sugg
in [RFC7252]. Each chunk is transmitted as a CoAP request to a given resourc
the ''consumer''. This draft provides the necessary header extensions that e
the ''consumer'' to maintain the sequence of the information segments in tim
space.

3.1. Optional Context Aware Semantic Switch

Before forming the CoAP message for each segment, the streaming application
use a real-time analytics module (henceforth referred as 'analytics module')
may provide inference to the ''Stream'' layer to decide the exchange semanti
the current segment. The message is sent reliably (CON message) or as best-e

(NON message with No-Response option) based on the segment's information
criticality. Criticality is measured in terms of importance of the segment-c
in reconstruction of the frames at the consumer. However, determination of
criticality can be done on many aspects involving several application featur
like the source encoding type, the rendering logic at the consumer, etc. Thi
the over-all balance between QoE and resource-consumption may be maintained.
2 explains the idea with conceptual blocks. The overall concept and its effi
has been explained with experimental results in [Wi-UAV-Globecom]

```
+-----------------------+
|      Application      | Information segment    ---------
+-----------------------+ ===================> |Real-time|
|         Stream        | <=================== |Analytics|
|-----------------------|  Reliable/             ---------
|   Requests/Responses  |  Best-effort?
|-----------------------|
|        Messages       |
+-----------------------+
```

Figure 2: Illustrating the concept for context aware switching

Some examples are:

Example-1: Temporally compressed videos like MPEG consist of Group of Pictur (GoP) which comprises I-frames (Intra-frames) or key-frames, P-frames (Predicted frames) and B-frames (Bidirectional frames). Out of these 3 ty of frames I-frames are most critical in terms of synchronizing with the G 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 B frames may be transferred as best-effort requests.

Example-2: Let us consider a Motion JPEG (MJPEG) stream. In this case all th frames are independent JPEG frames and there is no temporal compression. analytics module may treat the segments containing MJPEG meta-data for ea frame as critical segments and transfer them through reliable messaging. of the segments may be transferred as best-effort requests. An intelligen rendering engine at the ''consumer'' application may compensate for / con any possible loss of non-meta-data (non-critical) segments using the reli received meta-data and rest of the non-meta-data segments received throug best-effort. This way high QoE can be ensured despite reduced resource us

4. The Options Introduced

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

1) Stream_info: Consumes one unsigned byte. It maintains the stream identity indicates the present phase of exchange. It is both a request and respons 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 stre 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.

2) Time-stamp: It consumes 32-bit unsigned integer. It is a request option. 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 be accompanied with the Time-stamp option. It is a combination of two fie The 15-MSBs indicate the ''offset'' at which the present segment is place the frame corresponding to the given timestamp. The LSB indicates if the current segment is the last segment of the frame corresponding to the giv 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 prese video transfer is initiated by the ''producer'' which acts as the client.

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. example, while a fixed infrastructure system is using streamed FPV feed from there may be situations where each time a UAV is low on resources (energy an computation, a new UAV with better state of resources (fresh battery, etc.) commissioned. The overall operation becomes simple if the newly commissioned readily starts its job by streaming to the same resource at the fixed

infrastructure. It can be easily configured to determine whether the consume
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 commissione
consumer has to re-initiate the request again.

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

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 reque
Exact format of the script is application dependent and is not part of this

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

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


```
Client (Producer)                                   Server (Consumer)
|                                                   |
| POST: CON;                                        |
|       URI=/video;                                 |
|       Stream-info = <5-bit ID>000;                |
|       Payload= CBOR or JSON                       |\
+-------------------------------------------------->| |
|                                                   | |Stream
```

```
   | ACK;                                             | |negotiation
   | Response = 2.04 CHANGED                          | |
   | Steam-info = <5-bit ID>001                       | |
   |<-------------------------------------------------|/
   :                                                  :
   :                                                  :
   |(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.)                                       | |
   |                                                  | | Stream
   | POST: NON;                                       | | ongoing
   |       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>           | |
   +------------------------------------------------->| |
   |                                                  | |
   :                                                  : |
```


Figure 3: Example showing successful negotiation of streaming parameters fol
by transmission of video information and control. It is assumed that the seg
size negotiated as 1024 at the initiation. So, the position of the 2nd block

1024. Note the use of No-response option with NON request for the non-critic
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 ha when the consumer application may need a less frame-rate than what is propos the producer. So, the ''consumer'' may request a lower frame-rate and thereb 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 consumer requests more frame-rate than what is initially proposed by the pro then the producer may insist on the lower frame-rate. Renegotiation may also 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 se to the consumer.

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

Fig. 4 illustrates exemplary exchanges for re-negotiation.

```
Client (Producer)                                         Server (Consumer)
|                                                         |
| POST: CON;                                              |
|       URI=/video;                                       |
|       Stream-info = <5-bit ID>000;                      |
|       Payload= CBOR or JSON                             |\ Initial
+-------------------------------------------------------->| |negotiation
|                                                         | |followed by
| ACK;                                                    | |renegotiation
| Response = 2.04 CHANGED                                 | |request with
| Steam-info = <5-bit ID>010                              | |revised
| Payload= CBOR or JSON                                   | |params.
|<--------------------------------------------------------|/
|                                                         |
| POST: CON;                                              |
|       URI=/video;                                       |
|       Stream-info = <5-bit ID>010;                      |
|       Payload= CBOR or JSON                             |\ Successful
+-------------------------------------------------------->| |renegotiation
|                                                         | |as the
| ACK;                                                    | |consumer
| Response = 2.04 CHANGED                                 | |agrees to the
| Steam-info = <5-bit ID>001                              | |revised
|<--------------------------------------------------------|/ proposal.
:                                                         :
:               (Streaming starts)                        :
```

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

Fig. 5 illustrates exemplary exchanges when a stream negotiation is unsucces
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 is not in the scope of this draft.

```
Client (Producer)                                        Server (Consumer)
|                                                        |
| POST: CON;                                             |
|       URI=/video;                                      |
|       Stream-info = <5-bit ID>000;                     |
|       Payload= CBOR or JSON                            |\ Unsuccessful
+------------------------------------------------------->| |negotiation.
|                                                        | |The request
| ACK;                                                   | |is successful.
| Response = 2.04 CHANGED                                | |But consumer
| Steam-info = <5-bit ID>011                             | |may reject
| Payload= CBOR or JSON                                  | |for some
|<-------------------------------------------------------|/ reason
|                                                        |  mentioned in
                                                            Script.
```

Figure 5: Example showing unsuccessful renegotiation despite successful resp
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 th best-effort delivery on UDP. Despite that the application designer can make REaLiST implicitly congestion aware and proactively avoid congestion. CoAP h basic congestion avoidance mechanism which uses exponential back off to incr the timeout for retransmissions. However, that works only for CON messages.

The implicit congestion avoidance works like this: In case the producer fai successfully transfer a critical segment of a frame within the MAX_TRANSMIT_ 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 fr 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 i 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, n critical are delivered with best-effort in an open-loop exchange. Also, the 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 basic approach should be: Buffer till a critical time to iron out the jitter

out-of-order arrival of the segments, play out from the appropriate buffer a constant rate determined by the frame-rate of the video. There may be intell algorithms to play-out with high QoE despite non-arrival of non-critical seg 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 Globecom]. The play-out architecture leverages on the design assumption abou '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 l 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 t uncontrolled fragmentation leading to undesired explosion of traffic in the network. For IPV6 network, the MTU can be determined using Path MTU Discover (PMTUD) [RFC8201] which bestows the responsibility of determining the path M the end-points itself.

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

7. IANA Considerations

The IANA is requested to assign numbers to the three options introduced in t 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
base CoAP specification [RFC7252].

9. References

9.1. Normative References

[RFC7252]

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

[RFC7967]

Bhattacharyya, A., Bandyopadhyay, S., Pal, A., Bose, T., ''Constrained Appli
Protocol (CoAP) Option for No Server Response'', RFC 7967, August, 2016.

[RFC8201]

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

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 S
Object Networking", RFC 7452, March, 2015.

[Murphy]

Murphy, C., ''Internet of Things: Are you underestimating video?'', Availabl
online:
http://www.informationweek.com/bigdata/bigdataanalytics/internetofthingsarey
restimatingvideo/a/d-id/1269508, June, 2014.

[Pereira]

Pereira, R., Pereira, E. G., ''Video Streaming Considerations for Internet o
Things'', International Conference on Future Internet of Things and Cloud, p
52, August, 2014.

[RFC7959]

Bormann, C., Shelby, Z., ''Block-Wise Transfers in the Constrained Applicati
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
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
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-streamin
Experience for Delay-sensitive IoT Applications : A RESTful Approach'', acce
Globecom (Wi-UAV workshop), Dec., 2018.

[Perkins]

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

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: arpan.pal@tcs.com