CoAP Message Transport with Packet Wash

Lijun Dong, Richard Li
Futurewei Technologies Inc.
2330 Central Expressway
Santa Clara, CA, U.S.A
Email: {lijun.dong, richard.li}@futurewei.com

Abstract—**Network congestion in the current Internet usually causes packet loss and prolonged packet delivery latency due to retransmission. Qualitative Communication as one of the major features in New IP reduces the network layer operation from the packet level to much smaller granularity, namely packet wash. This paper illustrates such possibility by using Constrained Application Protocol (CoAP) as an example, which would be of wide usage owing to the thriving of Internet of Things (IoT) applications. The paper proposes two aspects of packet wash actions to CoAP packet: (1) header compression facilitated by cross-layer design; (2) partial packet dropping empowered by payload chunkification and New IP metadata. The paper elaborates on the detailed mechanisms, which provide many benefits with very slim overhead.**

Keywords- In-network; New IP; CoAP; block-wise transfer; packet wash; Qualitative Communication; UDP; IP; partial dropping; caching.

I. INTRODUCTION

Recently, New IP [1][2][3] has been proposed as an advanced network protocol specification to modernize the network layer without changing the fundamental Internet architecture. The New IP framework ensures the backward compatibility with the current Internet protocols and requires minimum effort to upgrade. It brings the intelligence of user and application awareness, continuity of services into the network. The concise New IP packet format as shown in Fig. 1 is designed to include three major elements:

1) New IP allows for hybrid formats of source and destination addresses according to the functionality and network interfaces of the communicating parties. The New IP addressing allows different types of addresses to be integrated and communicated in a flexible way.
2) New IP contract enables new types of modern courier-like network services at the finest packet-level granularity, e.g., High Precision Communication [4][5][6], Qualitative Communication[7][8][9]. The network and routers fulfill the contract. The Contract could include contract clause(s) and associated metadata. A contract clause specifies how the routers process the packet as it is forwarded in the network based on the configured triggering event and condition. The "Metadata" contains data about the packet, as well as the contextual information about the user/application, etc. And it can also allow New IP packet to collect the customized statistics about the flow on intermediate hops.
3) The New IP Payload can be the same as that in the existing IP, but can also be a sequence of sub-payloads, which share the same sources and destinations, but may be chuckified from currently intact payload. By reducing the granularity of operation on packet payload, the transport paradigm switches to the new Qualitative Communication [7][8][9]. Packet wash action in Qualitative Communication is designed to be taken when facing network congestion. A packet wash truncates a packet and only drops some portions of the packet to mitigate entire data loss. This has the effect of reducing the packet size until that packet is small enough to be stored and/or transmitted by the network node even though there is network congestion.

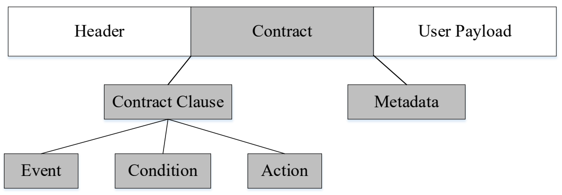

Figure 1. New IP: unified framework for future IP packet

Ideally, packets are transferred in their entirety without losing any part in the packet. However, under certain circumstances the controlled dropping of certain portions or fields in the packet as a last resort may be preferable over losing packet in their entirety, in particular when this means that extra delay due to the need for retransmission can be avoided. In our previous works on Qualitative Communication [8][9], the major effort has been focused on how to design the packetization methods to facilitate the chuckification of packet payload. Packet Wash is mainly applied to packet payload. However, Packet Wash could also exhibit certain commonalities with performing on-demand lossful compression.

In this paper, we use Constrained Application Protocol (CoAP) [10] as an example to show that the packets transported on the Internet may be subject to packet wash operation when network congestion presents. CoAP is specifically designed to satisfy the urge for connecting and integrating the low-power and constrained devices such as sensors and actuators at a global scale, which has pushed towards the Internet of Things (IoT) vision. The CoAP message format and its major features are summarized in Section II. The paper proposes a cross-layer design of Internet protocol stack to expose the upper layer headers to the network layer, some fields that are not absolutely necessary may be partially removed to reduce the headers' size. The paper presents such methods of further compressing CoAP header, correspondingly User Datagram Protocol (UDP) header, and traditional IPv6 header. On the other hand, with the adoption of New IP framework, the application layer contexts may be made aware to the network layer, which facilitates appropriate actions on the payload, including washing (partial dropping), caching, proactive re-delivery, etc., which will be described in detail in Section III. In Section IV, the overhead and benefits that are brought by the proposed methods are discussed. Section V concludes the paper.

II. CoAP Message

Internet Engineering Task Force (IETF) Constrained RESTful Environments (CoRE) Working Group (WG) [12] defines CoAP as a customized web transfer protocol for use in IoT nodes.

The CoAP messaging model is based on the exchange of messages running over UDP between endpoints. CoAP uses a short fixed-length binary header of four bytes that might be succeeded by a variable-length Token value between 0 to 8 bytes, compact binary options in TLV (Type-Length-Value) format and a payload filling the rest of the datagram. The CoAP message format is shown in TABLE I.

- Type (T): 2-bit unsigned integer, which indicates the type of the message, whether it is a Confirmable (0), Non-confirmable (1), Acknowledgement (2), or Reset (3) type.
- Token Length (TKL): 4-bit unsigned integer. The Token is used to match a response with a request.
- Code: 8-bit unsigned integer, which is split into a 3-bit class (most significant bits) and a 5-bit detail (least significant bits). There are 4 types of method codes with the most three significant bits set to 0: 0.01 GET, 0.02 POST, 0.03 PUT, 0.04 DELETE.
- Message ID: 16-bit unsigned integer, which is used to detect message duplication and to match messages of type Acknowledgement/Reset to messages of type Confirmable/Non-Confirmable.

CoAP defines less options than Hypertext Transfer Protocol (HTTP), i.e. If-Match, Uri-Host, ETag, If-None-Match, Observe, Uri-Port, Location-Path, Uri-Path, Content-Format, Max-Age, Uri-Query, Accept, Location-Query, Proxy-Uri, Proxy-scheme and Size1. The detailed definition and usage of those options could be found in the specification [10], or in this survey [11] for a short reading.

When included in a GET request, the Observe Option [13] requests the server to notify the client if changes happen to the target resource. Otherwise, the request falls back to a normal GET request. When included in a response, the Observe Option identifies the message as a notification. The present of Observe Option can differentiate a notification from a normal response.

CoAP also defines the block wise transfer [14] to limit the size of datagrams in constrained networks: by the maximum datagram size (˜ 64 KB for UDP), or to avoid IP fragmentation (MTU of 1280 bytes for IPv6), or to avoid adaptation-layer fragmentation (60-80 bytes for 6LoWPAN). Two options (Block1, Block2) are used: when Block1 is present in a request or Block2 in a response, it indicates a block-wise transfer and describes which part of the entire payload this specific block-wise occupies.

TABLE I. CoAP Message format

Ver	T	TKL	Code	Message ID
Token (if any)				
Options (if any)				
1 1 1 1 1 1 1 1			Payload (if any)…	

III. System Architecture and Functionalities

In this section, we will present the cross-layer design for Internet protocol stack and in-network packet wash mechanisms for CoAP packets.

A. Cross-Layer Design for Internet Protocol Stack

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the US-letter paper size.

When two parties initiate a communication between them, the application layer header and application transport layer (CoAP, HTTP etc.) header are set up properly. The transport layer adds a Transmission Control Protocol (TCP) or UDP header in front of the application transport layer header and treats the application transport layer header and payload as opaque. The network layer adds an IP header on top of the TCP or UDP header, the application transport layer header and payload, and treats them as opaque. In principle, one layer is not able to access the fields in the header of another layer, either it is upper layer or lower layer. The methodology of layered protocol design possesses some advantages from the protocol transparency perspective. For example, protocols in one layer can be designed, improved, or even substituted without imposing any influence on other protocol layers. However, it is likely that the information from one layer may be useful to another layer. As a result, the performance optimization between different protocol layers becomes impossible under such methodology of layered protocol design, which can significantly degrade the network performance. This unavoidably leads to the cross-layer design. The concept of cross layer design is about sharing of information among different protocol layers for adaptation purposes and to increase the inter-layer interactions.

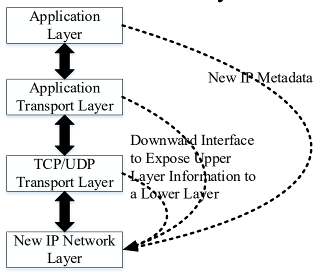
Application Layer
Application Transport Layer
TCP/UDP Transport Layer
New IP Network Layer
New IP Metadata
Downward Interface to Expose Upper Layer Information to a Lower Layer

Figure 2. Cross-layer design for information sharing across layers

To avoid above restriction, Figure 2. shows that alternatively the application layer may a maintain downward interfaces to expose upper layer information to the network layer, such that the network layer is able to utilize the information from the application layer such as the type of payload data, characteristics of payload data, user's requirement on performances such as latency and quality of the data, etc., as well as its tolerance level on the disparity from the exact requirement. If the underlying network layer supports New IP framework, the application layer information can be naturally passed to the network layer through New IP Metadata. The application transport layer, and TCP/UDP transport layer protocol layer instead may maintain downward interfaces to expose their headers to IP protocol layer to enable the possible packet wash mechanisms proposed in this paper.

The paper mainly focuses on the protocol stack with CoAP protocol as the application transport protocol, correspondingly UDP as the transport layer protocol and proposes the in-network packet wash mechanisms for CoAP messages.

B. *Details of In-Network Packet Wash Mechanisms*

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

The in-network packet wash proposed in this paper mainly aims to design the optimal mechanisms to compress the headers, drop/cache the least important information and transport the most necessary information to the destination with the shortest latency when the network condition is not desirable. As shown in Figure 3. , between the two CoAP end nodes, there exists the network nodes (e.g., base station, gateway, routers) that would route and forward the CoAP messages. The in-network packet wash enables each of the intermediate network nodes to have the capability of differentiating multiple blocks in the packet and dispose them discriminately. Note, for the purpose of evolving deployment of such capability, some of the network nodes may remain with legacy implementation, which will simply either forward or drop the packet completely. For those network nodes (i.e., Router 2, 3, 5 and Base Station 6) which are enabled with the in-network packet wash capability for CoAP messages, the packet is processed with certain level of compression and discard when facing network congestion, which we will discuss in the later section.

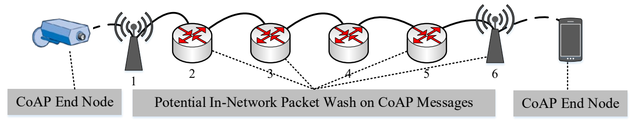

Figure 3. Illustrative example

Due to the simplicity and RESTful nature of CoAP protocols, the most common CoAP messages that are sent between two end nodes are the combinations of: [Confirmable, Non-Confirmable, Acknowledgement, Reset]+[GET, PUT, POST, DELETE]+[Request, Response]. Since the only difference between Confirmable and Non-Confirmable messages is that whether the message is acknowledged by the recipient upon arrival, we don't distinguish them later. Among those messages, the GET response, POST request, PUT request may contain data in the payload, which the paper will focus on. Given the GET response, POST request, PUT request messages have similar syntax, depending on whether it is initiated by the client or server, in the paper, we use the GET response message as the example to illustrate how a CoAP message may be washed by the network nodes. Other types of CoAP messages can be treated similarly.

The GET response message is sent by a CoAP end node upon receiving a GET request message or triggered by a notification due to an existing subscription. The only difference between those two cases is that whether the Observe Option is included in the message. In order to facilitate the in-network packet wash on the GET response message, we propose that in the corresponding GET request message, the requesting end node's application layer will optionally pass the following parameters to the New IP Metadata:

- Type of requested data that corresponds to the application (e.g., surveillance video, temperature)
- Latency budget for returned data
- Required quality of returned data (e.g., for video data, view angle, resolution etc.)
- Tolerance degree on the distortion of the data quality

When packet wash capable network node (e.g., Router 5 in Figure 3.) receives a GET response packet, it may make decisions on processing the packet, depending on the current network condition. The actions it may take include:

- Compress the CoAP header by accessing the CoAP header fields through the downward interface from the application transport layer to New IP network layer.
- Compress the UDP header by accessing the CoAP header fields through the downward interface from the application layer to New IP network layer.
- Compress the IP header.
- Cache the payload data and remove the payload from the packet.
- Truncate the payload partially or completely.

1) *New IP Metadata Design*

Status (6 bits): It indicates whether the packet has been washed by the previous network nodes which have forwarded the packet. The very first bit indicates whether the packet is original or not (e.g., 000000 denotes that the packet is original). The second bit indicates whether the packet's CoAP header is compressed or not (e.g., 110000 denotes that the packet's CoAP header has been compressed). The third bit indicates whether the packet's UDP header is compressed or not (e.g., 101000 denotes that the packet's UDP header has been compressed). The fourth bit indicates whether the packet's IP header is compressed or not (e.g., 100100 denotes that the packet's IP header has been compressed). The fifth bit indicates whether the packet's payload is partially dropped or not (e.g., 100010 denotes that the packet's payload has been partially dropped). The sixth bit indicates whether the packet's payload is completely dropped (e.g., 100001 denotes that the packet's payload has been completely dropped). Any combination of the 5 types of revisions to the original packet could exist by setting the corresponding bit to 1 and results in the first bit to be 1.

CoAPWash (2 bits): Each bit indicates whether a field is removed/dropped from the original CoAP header. The first bit indicates whether that the *TKL*, *MessagID* and *Tokens* fields are removed. The second bit indicates whether the *content-format*, *max-age*, and *ETag* options are removed and cached.

UDPWash (4 bits): The first bit indicates whether the source port is compressed to 4 bits. The second bit indicates whether the destination port is compressed to 4 bits. The third bit indicates whether the *Length* field is removed. The fourth bit indicates whether the *Checksum* field is removed.

IPWash (6 bits): The first bit indicates whether the *Traffic Class* field is removed. The second bit indicates whether the *Flow Label* is removed. The third field indicates that the *Payload Length* field is removed. The fourth bit indicates whether the *Next Header* is removed and cached. The fifth bit indicates whether the *Hop Limit* field is removed and cached. The six bit indicates whether the *Source Address* and *Destination Address* fields are removed.

IPExtCache (4 bits): Each bit indicates whether the extension header is removed and cached, following the sequence as shown in TABLE II. The CoAP options that could be included in a GET response message are: *Content-Format*, *Max-Age*, *Etag* and *Observe*.

Multi (1 bit): It indicates whether there are simultaneous requests between the requesting end node and responding end node. If the bit is set, the *Tokens* field must be included in the CoAP header and cannot be removed. If the bit is not set, a network node may remove the *TKL* and *Tokens* field in the CoAP header. On the other hand, if there is only one request between the two end nodes, the *Message ID* field can also be removed since the requesting end node can match the response to the request by the responding end node's address.

TABLE II. OPTIONS IN COAP GET MESSAGE

Name	**Length (bytes)**
Etag	1-8
Content-Format	0-2
Max-Age	0-4
Observe	0-3

TagCache (1-8 bytes): The identifier of the cached copy of payload and other information stored in an intermediate network node, which can be used by the requesting node to retrieve the payload later when the network condition becomes satisfactory. Such identifier could be very short, as far as it is unique among all the cached content stored in the intermediate network nodes between source and destination. The field may contain multiple identifiers of cached portions of the current CoAP message.

Significance (1-8 bits): If the message is one block in the block-wise transfer, this field can be used to indicate the significance of the block in recovering or interpreting the original data. A network node can use this field to decide whether the payload in the message may be dropped.

Selects (length varies): This field is used to include any possible requirements from the requesting node (client), as well as properties related to the actual data being returned by the responding node (server). The *Type-of-Data* lets the network nodes understand the type of data included in the payload, which in turn may decide on the priority of the packet. For a multimedia type of data, the examples of the application layer parameters are proposed and shown in TABLE III. The *Tolerance-Degree* option following the previous option is to indicate the requesting node could have some level of tolerance if the data is not exactly matching its requirement. This *Tolerance-Degree* could give the flexibility to the network node to selectively drop some parts of the packet payload to fit the current network condition, with some sacrifice to the multimedia data' quality in resolution or view-angle, or performance in latency.

TABLE III. SELECTS IN METADATA SEGMENT

No.	Name	Format	Length
1	Type-of-Data	string	0-1
2	Latency-Budget	unit	0-1
3	Tolerance-Degree	unit	0-1
4	View-Angle	unit	0-2
5	Tolerance-Degree	unit	0-1
6	Resolution	unit	1
7	Tolerance-Degree	unit	0-1

2) *Actions on Payload*

A network node can take the following actions on the payload, the overall procedure is shown in Figure 4. .

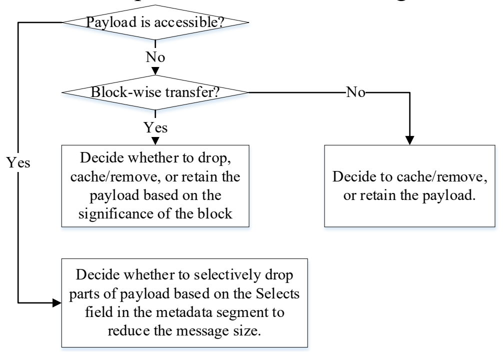
Payload is accessible?
No
Block-wise transfer?
No
Yes
Yes
Decide whether to drop, cache/remove, or retain the payload based on the significance of the block
Decide to cache/remove, or retain the payload.
Decide whether to selectively drop parts of payload based on the Selects field in the metadata segment to reduce the message size.

Figure 4. Actions on the payload

If the payload is encrypted, the data itself is invisible to the network node and cannot be processed in any method. However, the network node can decide to whether to cache the payload and other associating information in the CoAP header, as well as remove it completely from the original message. If the network node caches the payload and other associating information in the CoAP header (e.g., *Content-Format*, *Max-Age*, *ETag*) and removes the payload from the original message, the network node will set the very first bit in the *Status* field in the metadata segment to 1 to indicate such action. On the other hand, the network node can set the *TagCache* field in the metadata segment to include its local identifier of the cached content for the requesting node to retrieve the data when the network condition becomes better. If the *TagCache* is not setup after the network node drops the payload from the original message, it indicates that the network node is in charge of sending the data contained in the payload to the requesting node after it sees the network condition becomes better. The requesting node only needs to expect and wait for the data to be delivered later.

If the message is one block in the block-wise transfer, we propose that the responding node may organize the blocks such that each of the blocks may contain different parts of the requested data with different importance levels. In the *Significance* field of the metadata segment in the message, it will indicate the relative importance of the current block compared to the other blocks. The larger the *Significance* value is, the more relatively important the block is to recover the information contained in the data. The network node could decide whether to drop the block given that the network condition is not satisfactory to transport the block to the next hop. If the block is relatively important, the network node needs to try its best to retain the payload and send to the requesting node. Otherwise, the payload could be dropped, removed/cached for later retrieval or delivery.

A network node may need to decide on which CoAP messages to perform packet wash operation on, if there are multiple candidates from different flows waiting in the outgoing queue. The relative significance of a CoAP message is determined as shown in (1).

$$sig = \begin{cases} 1, & \text{if non} - \text{blockwise} \\ Significance, & \text{if blockwise} \end{cases} \quad (1)$$

$$Significance = \alpha * \frac{sig_{block}}{num} \quad (2)$$

If a CoAP message does not have block-wise transfer options (i.e., block1 and block2 options), then the significance level of the message is regarded as 1. Otherwise, the relative significance of the CoAP message among all other blocks from the same flow is indicated in the Significance metadata field, which is basically assigned by the sender as the significance level of the block divided by the number of blocks in the flow. The factor α would make sure that the significance level of the blocks which the sender considers as the most important and are preferred to be delivered without re-transmission is larger than 1, as shown in (2).

If the payload is not encrypted or different chunks within the payload are encrypted independently, the network nodes can selectively drop parts of the payload based on the network condition, user's requirement and application layer parameters included in the metadata segment of the message. The *Selects* field is used to specify those proposed requirements, which could be set up by the requesting node when the CoAP request message is sent. The responding node prepares the data according to the requesting node's requirements specified in the *Selects* field. When the response message is sent, the actual properties of the data (e.g., *Type-of-Data*, *Resolution*, *View-Angle*) are setup accordingly. The related *Tolerance-Degree* is copied from the request message. On the other hand, the *Latency-Budget* is modified for the response message to reach the requesting node by deducting the used time from the original latency budget. Based on the *Tolerance-Degree*, the message size may be reduced. For example, the resolution may be adapted by selectively dropping some parts of payload, such that the resolution could be lower than the current value, but higher than the requesting node's tolerable value. Consequently, the packet size is reduced, and the packet can be avoided being completely dropped.

3) CoAP Header Compression

The CoAP header of a GET response packet could be possibly compressed to reduce message size, under the condition that some of the fields in the header may be removed without influencing the processing or understanding of the CoAP header at the receiver side. If the CoAP header is not compressed by any previous network node on the path from the sender to the receiver, based on the status field in the New IP metadata (the first and second bit in *Status* field), the currently received message might be eligible for CoAP header compression. Otherwise, the network node moves to the UDP header, which will be discussed in Section 4). The proposed algorithm of compressing CoAP header of GET response message is shown in Figure 5.

```
Algorithm 1 CoAP Header Compression
If the first two bits of Status is "11", then
    The CoAP header has been compressed and no further
    actions needed on the CoAP header.
else if the second bit of Status is "0", then
    if $multi = 0$, then
        remove TKL, MessageID and Tokens field if they exist.
        Set the first bit of CoAPWash to "1".
    end
    if payload is removed from the message and cached, then
        Remove content-format, max-age, and ETag options.
        Cache content-format, max-age and ETag.
        Set the second bit of CoAPWash to "1".
    else
        Retain content-format and max-age options
        Remove ETag option.
    end
        Set the first and second bit in the Status field is set to
        "11".
end
```

Figure 5. Algorithm 1

In the CoAP header, the *TKL*, *MessagID* and *Tokens* field can be removed if there are no simultaneous requests between the requesting end node and responding end node, which is indicated in the *Mutli* field in the metadata by the responding end node.

The CoAP options that could be included in a GET response message are: *Observe*, *Content-Format*, *Max-Age* and *ETag*. The *Observe* Option indicates that this is a notification for the subscription. This option should not be removed. The *Content-Format* Option indicates that the representation format of the message payload. The representation format is given as a numeric content format identifier that is defined in the "CoAP Content-Formats" registry [15]. It should not be removed if the payload stays in the message. The *Max-Age* Option indicates that the maximum time a response may be cached before it is considered not fresh. It should not be removed if the payload stays in the message. The *ETag* Option is generated by the responding node and used as a resource-local identifier for differentiating between representations of the same resource that vary over time. The *ETag* Option in a response provides the current value of the entity-tag for the requested resource representation in the payload. The *ETag* Option can be removed, resulting in that the requesting node is not aware of the entity-tag of the received data, which is not crucial information in interpreting the representation.

After the CoAP header is processed and compressed according to the algorithm proposed above, the first two bits in the *CoAPWash* field in metadata segment of the message is set up accordingly and the first and second bit in the *Status* field is set to "11". For the future network nodes, after they detect the second bit in the *Status* field is set up, they would not act on the CoAP header anymore.

4) UDP Header Compression

UDP (both source and destination) ports may be compressed to 4 bits, if the requesting and responding nodes agree to only use 16 number of specified ports for different applications which are using CoAP as the application layer protocol. Other than the source port and destination port, there are two other fields in the UDP header:

- Length: It indicates the length in bytes of the UDP header and the encapsulated data. The minimum value for this field is 8. This field can be removed without influencing packet interpretation.
- Checksum: This is computed as the 16-bit one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes. If the checksum is set to zero, then checksuming is disabled. If the computed checksum is zero, then this field must be set to 0xFFFF. Since a network node may partially drop the IP header, UDP header and the payload based on the mechanisms proposed in the paper, the Checksum field needs to be disabled if any modifications happen to the original packet, thus is removed (in this paper, removing Checksum field means disabling the checksuming).

After a network node processes the UDP header, the *UDPWash* field in the metadata segment is setup correspondingly.

5) IP Header Compression

The standard IPv6 header is composed of the fields as shown in Figure 6. .

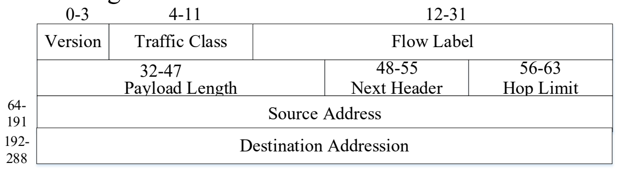

Figure 6. IPv6 header

- Version (4-bits): It represents the version of Internet protocol, i.e., 0110, which can be fixed, thus can be removed.
- Traffic Class (8-bits): These 8 bits are divided into two parts. The most significant 6 bits are used for Type of Service to let the routers know what services should be provided to this packet. The least significant 2 bits are used for Explicit Congestion Notification (ECN). For a CoAP message, if this field is set up by the source, it should be not removed, and the routers need to take actions according to the Type of Service and Explicit Congestion Notification to forward the message. Otherwise, this field can be removed.
- Flow Label (20-bits): This label is used to maintain the sequential flow of the packets belonging to a communication. The source labels the sequence to help the router identify that a specific packet belonging to a particular flow of information. This field helps avoid re-ordering of data packets. Since the data can be carried in one CoAP message or multiple ones with block-wise transfer, the Flow Label is not necessary for the requesting node to recover the data accurately. The Block2 option contains the order information of the blocks. Thus, the Flow Label field can be removed.

TABLE IV. IPV6 EXTENSION HEADER TYPES AND SEQUENCE IN THE MESSAGE

IPv6 header
Hop-by-Hop Options header
Destination Options header
Routing header
Fragment header
Authentication header
Encapsulating Security Payload header
Destination Options header
Upper-layer header

- Payload Length (16-bits): This field is used to tell the routers how much information a particular packet contains in its payload. Payload is composed of Extension Headers and Upper Layer data. The field needs to be changed if the upper layer header compression and partial payload dropping happen.
- Next Header (8-bits): This field is used to indicate whether the type of Extension Header is present. The types and sequence of the Extension Headers are shown in TABLE IV. If we assume that the blocks in the block-wise transfer share the same values for the Extension Headers if they are included in the IPv6 header, then the Extension Headers only need to be transferred once in the first block in the block-wise transfer. The Extension Headers can be extracted from the cached copy stored in either the receiving node (in the scenario of successful delivery of the first block) or the intermediate network node (in the scenario of the Extension Headers of the first block are washed and cached during the transmission). This proposal also applies to the Hop Limit, Source Address and Destination Address.
- Hop Limit (8-bits): This field is used to prevent a packet to loop in the network endlessly. The value of Hop Limit field is decremented by 1 as it passes a hop. When the field reaches 0 the packet is discarded. This field cannot be removed.
- Source Address (128-bits): This field indicates the address of originator of the packet.
- Destination Address (128-bits): This field provides the address of intended recipient of the packet.
- After a network node processes the IP header, the IPWash and IPExtCache fields in the metadata segment are setup correspondingly.

IV. Discussions on Overhead and Benefits

The proposed methods rely on the New IP metadata inserted in the packet to instruct the intermediate network nodes to take actions when network congestion appears, which inevitably increases the packet size. However, we can affirm that such size overhead is extremely light and compatible with the existing packet size configurations, which will be explained below.

For a CoAP message, it usually fits within a single IP packet to avoid IP fragmentation (MTU of 1280 bytes for IPv6). The good upper bounds are 1152 bytes for the message size and 1024 bytes for the payload size. If the data is larger, the block-wise transfer would be implemented. The CoAP header size has a fixed value of 4 bytes. Followed by the fixed header fields, there could exist token and options. The token size can be as large as 8 bytes, while the options could be of varying sizes and the total size of options in TABLE II. could reach 17 bytes. By abstracting 1152 from 1280 bytes, the capacity that could be used by the New IP metadata and lower layer headers is 128 bytes. The UDP header size is 2 bytes. The IPv6 header size could be 80 bytes, including the extension headers. The total size of the New IP metadata proposed in Section III.B.1) can be as high as 20 bytes (less than 46=128-2-80). Thus, the New IP metadata size is small enough to be carried in the CoAP message.

CoAP is bound to unreliable transports such as UDP, it implements a lightweight reliability mechanism, without trying to re-create the full feature set of transport like TCP. It adopts a simple stop-and-wait retransmission reliability with exponential back-off for Confirmable messages. Qualitative Communication does not entirely eliminate the need for re-transmission since it cannot mitigate against irrecoverable loss of critical elements of the packet. However, the amount of information needing retransmission is dramatically reduced, since the critical information that is contained in CoAP blocks has higher priority/significance and is less prone to discards than before. As proposed in (2), the most important block(s) in a block-wise transfer would have higher significance level than the other concurrent CoAP messages queued in a congested network node, which ensures that they are more likely to be retained as much as possible and reach the receiver. Then the receiver is able to obtain the most crucial information without any extended delay caused by packet loss and retransmission.

V. Conclusions

Qualitative Communication has been proposed to mitigate the performance degradation in throughput, latency due to entire packet dropping caused by network congestion. Instead of dropping packets completely, packet wash proposed for Qualitative Communication is an action triggered by network congestion to partially remove some parts from the packet such that the packet can be retained in the outgoing queue and survive the deteriorating network condition. In this paper, we propose that packet wash could be applied to both packet headers and payload. Through the enablement of cross-layer design, the upper layer headers are potentially visible and compressible by the network nodes. We use CoAP protocol as an example to show the details of header compression mechanisms, i.e., to the CoAP header, UDP and IPv6 header correspondingly. On the other hand, thanks to the unified New IP framework, CoAP message could be encapsulated in New IP packet with metadata, which can be leveraged to pass some application or user's context to the network node to make intelligent and more effective packet washing operation on the packet payload. The paper also discusses the overhead might be brought by the proposed mechanism, as well as its feasibility from the perspective of packet size limit. In the last but not least, the benefits are summarized. A proof-of-concept prototype is planned as future works.

References

[1] R. Li, A. Clemm, U. Chunduri and L. Dong , "New IP: Enabling the Next Wave of Networking Innovation," book chapter in "Design Innovation and Network Architecture for the Future Internet", pp. 1-42, IGI Global.

[2] R. Li, K. Makhijani and L. Dong, "New IP: A Data Packet Framework to Evolve the Internet," IEEE HPSR 2020.

[3] R. Li, A. Clemm, U. Chunduri, L. Dong, and K. Makhijani, "A New Framework And Protocol For Future Networking Applications," ACM Sigcomm Workshop on Networking for Emerging Applications and Technologies (NEAT 2018).

[4] L. Han, Y. Qu, L. Dong and R. Li, "A Framework to Realize the Guaranteed Service for Bandwidth and Latency for Future IP network," 2020 Infocom workshop on New IP: The Next Step.

[5] A. Clemm and T. Eckert, "High-Precision Latency Forwarding over Packet - Programmable Networks," IEEE/IFIP Network Operations and Management Symposium, 2020.

[6] L. Dong and R. Li, "Packet Level In-Time Guarantee: Algorithm and Theorems", IEEE Globecom 2020.

[7] R. Li, K. Makhijani, H. Yousefi, C. Westphal, L. Dong, T. Wauters, and F. De Turck, "A Framework For Qualitative Communications Using Big Packet Protocol," ACM Sigcomm Workshop on Networking for Emerging Applications and Technologies (NEAT 2019).

[8] L. Dong and R. Li, "In-Packet Network Coding for Effective Packet Wash and Packet Enrichment," 2019 IEEE Globecom Workshop on Future Internet Architecture, Technologies and Services for 2030 and Beyond.

[9] L. Dong, K. Makhijani and R. Li, "Qualitative Communication Via Network Coding and New IP," IEEE HPSR 2020.

[10] Z. Shelby, K. Hartke and C. Bormann, "The Constrained Application Protocol (CoAP)," IETF RFC 7252.

[11] L. Dong and R. Li, "A Survey on IETF Standardization for Connecting and Integrating the Low-Power and Constrained IoT Devices," 2020 International Conference on Computing, Networking and Communications (ICNC).

[12] Constrained RESTful Environments (CoRE), IETF, https://datatracker.ietf.org/wg/core/about/.

[13] K. Hartke, "Observing Resources in the Constrained Application Protocol (CoAP)," IETF RFC 7641.

[14] C. Bormann amd Z. Shelby, "Block-Wise Transfers in the Constrained Application Protocol (CoAP)," IETF RFC 7959.

[15] Z. Shelby, "Constrained RESTful Environments (CoRE) Link Format," IETF RFC 6690.