Offloading QUIC – AN IMPLEMENTATION GUIDE

Manasi Deval, Gregory Bowers

IETF 104, Prague, March 2019

Ethernet Networking Division

Agenda

- Review the challenges and solution to the proof of concept
- Three types of possible solutions
  - Agree on the possible solutions across multiple implementations.

Outcome ?

- Possible solutions are classified into:
  - Using meta-data to solve the problem.
    - Generic meta-data processing imposed by Linux. Limited by the OS upstreaming barriers.
  - Modification of the implementation guideline
    - With large number of stacks, can we agree on a few implementation rules.
  - Modification of the header.
    - Folks in this room do not like this solution ☺

Connection Id has16 different sizes in the short header

Connection Id size varies on both Tx and Rx

Solve the problem with some meta data and implementation rule:

- Transmit solution
  - Augment the meta data with the CID size
- Receive solution
  - Header parsing will be programmed with a single size, for a server.

Connection Id has16 different sizes in the short header

Protocol solution: Encode connection Id with a varint

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|1|S|R|R|K|P P|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Destination Connection ID with a varint encoding (0..144)   ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           Packet Number (8/16/24/32)           ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           Protected Payload (*)          ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Optional connection Id in the short header

- The hardware has a classification schema to identify the crypto key.
- The current spec imposes that the hardware implements multiple schemas
  - Connection id match is higher priority over the outer 4 tuple

- Transmit solution
  - Meta data flowing with the packet identifies the size of CID.
- Receive solution
  - Packets receive at server will always have CID
  - Single size of CID

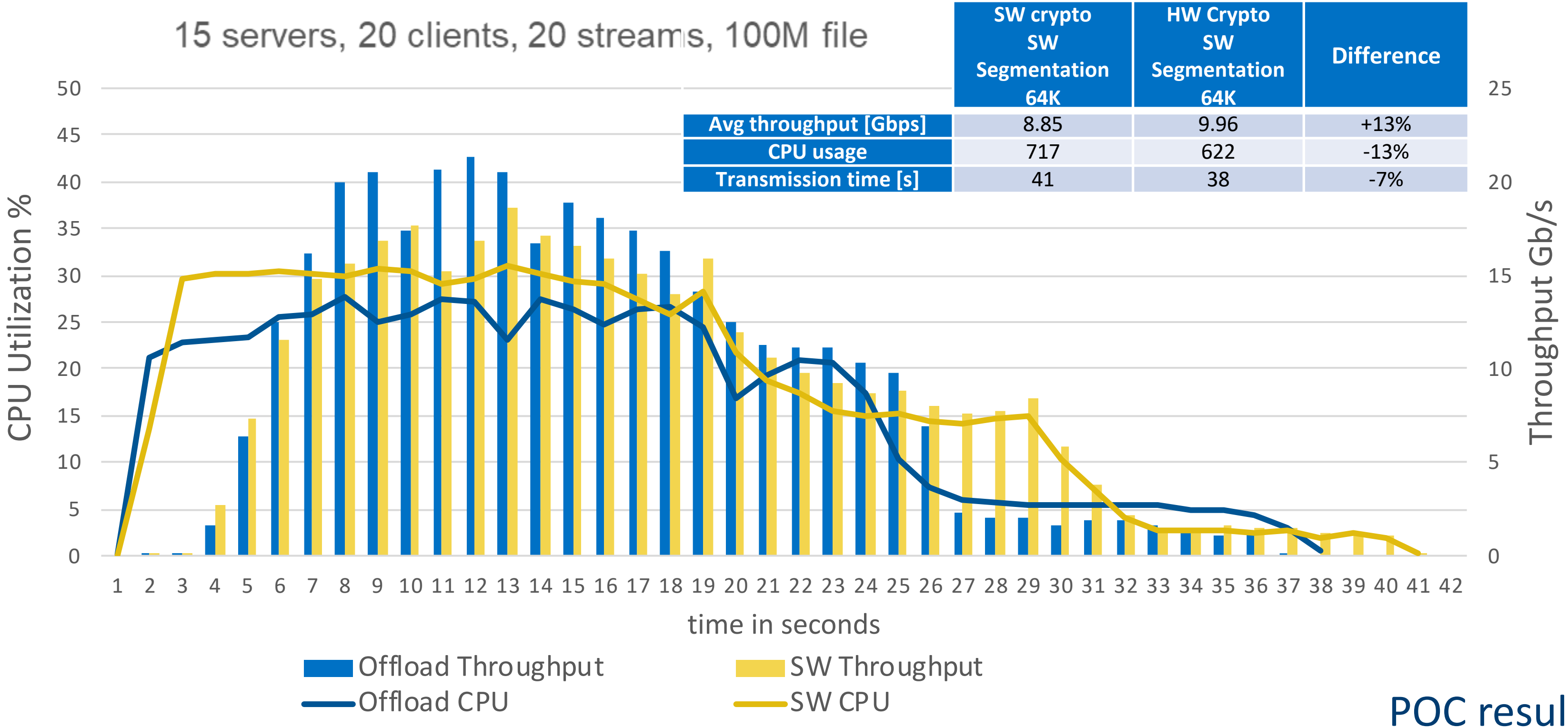
Experiment with Chromium

Chromium serving large number of clients.

- Saturate the network bandwidth with minimal clients.
- Each client does an HTTP request.
- Measurements on Tx side.

QUIC Crypto Offload Performance

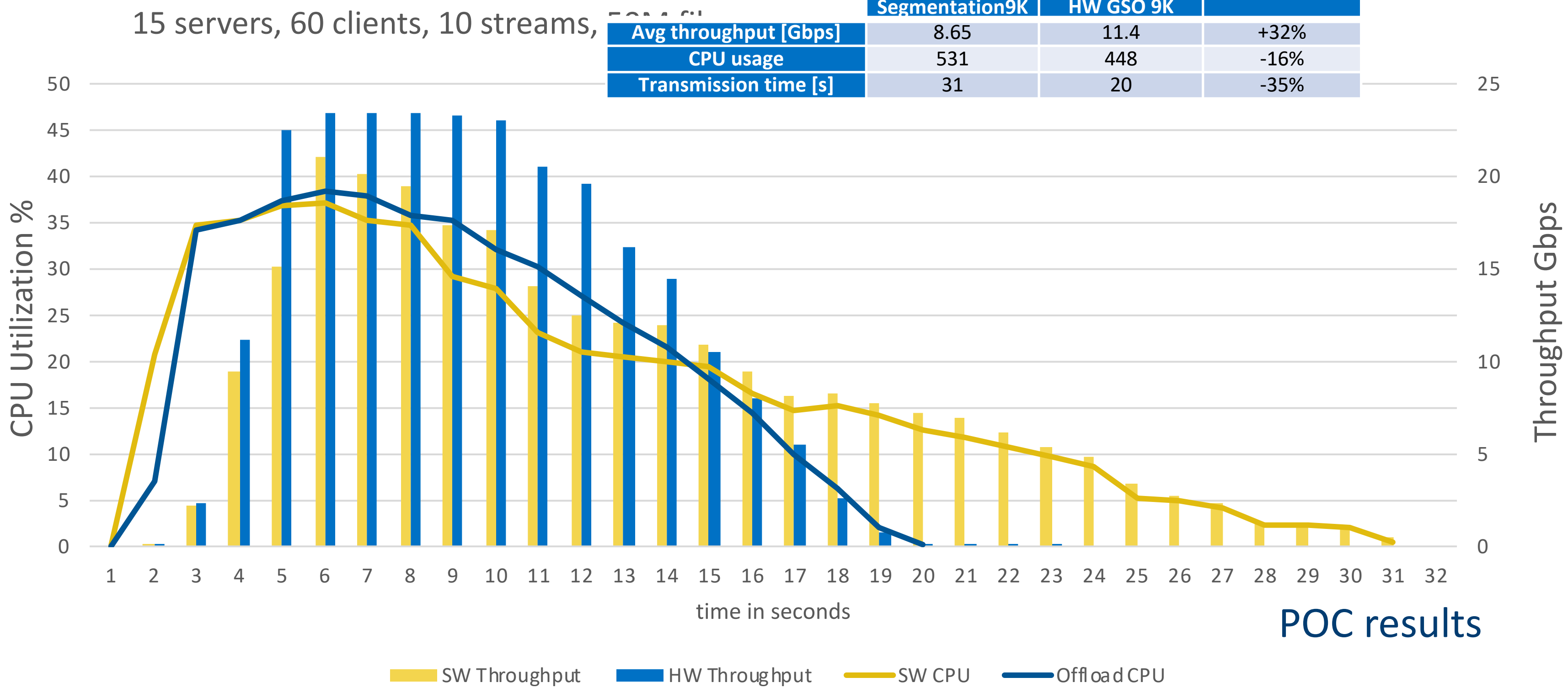
	SW crypto SW Segmentation 64K	HW Crypto SW Segmentation 64K	Difference
Avg throughput [Gbps]	8.85	9.96	+13%
CPU usage	717	622	-13%
Transmission time [s]	41	38	-7%



POC results

Now let's put it all together

	SW crypto SW Segmentation9K	HW Crypto HW GSO 9K	Difference
Avg throughput [Gbps]	8.65	11.4	+32%
CPU usage	531	448	-16%
Transmission time [s]	31	20	-35%

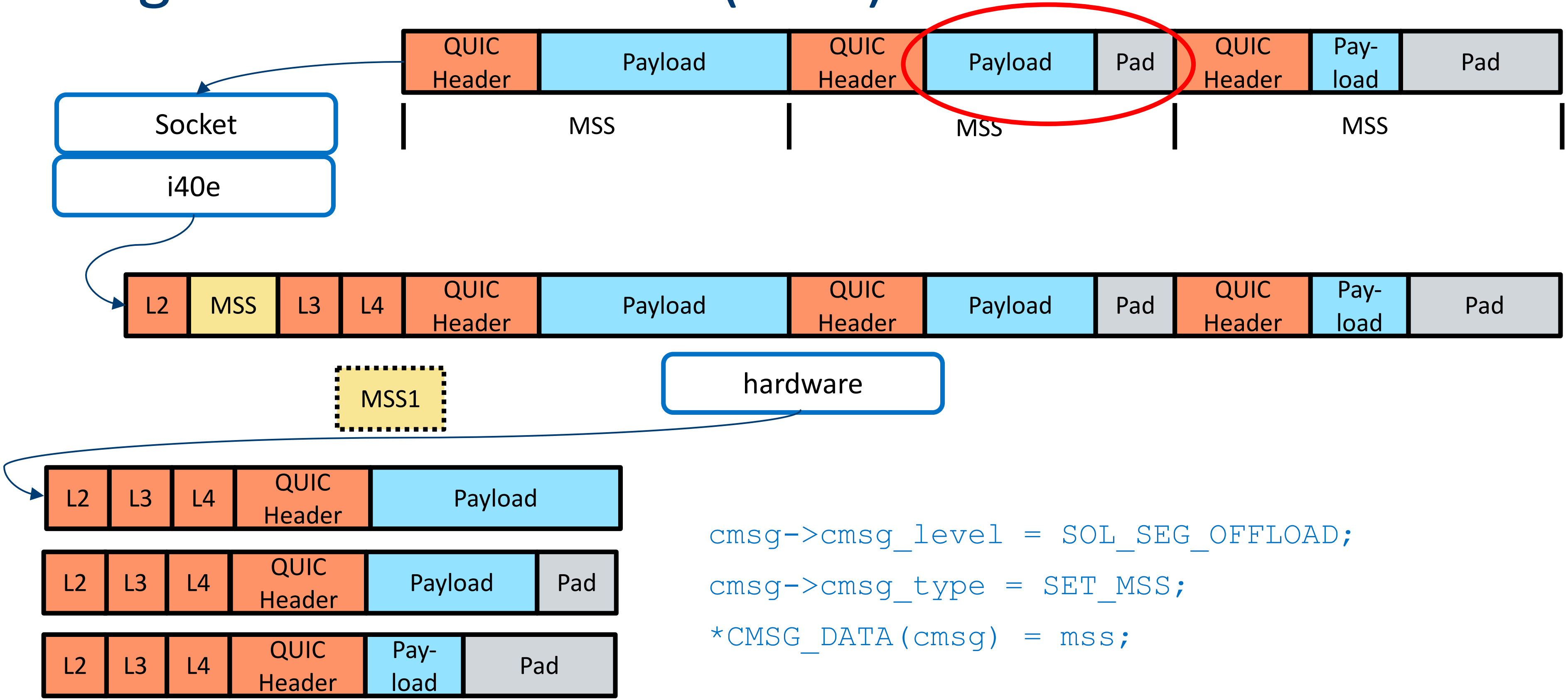


POC results

Programming the HW – User space to Driver

```
                                                        struct ulp_offload_devops {
setsockopt(sk, SOL_OFFLOAD, INIT_DEVICE, &init, ...)       ---->     int (*ulp_offload_init)(struct net_device *netdev, ...);

setsockopt(sk, SOL_OFFLOAD, ADD_SA_{TX|RX}, &addsa, ...)    ---->     int (*ulp_add_sa)(struct net_device *netdev, ...);

setsockopt(sk, SOL_OFFLOAD, UPDATE_SA_{TX|RX}, &upsa, ...)  ---->     int (*ulp_update_sa)(struct net_device *netdev, ...);

setsockopt(sk, SOL_OFFLOAD, DEL_SA_{TX|RX}, &delsa, ...)    ---->     int (*ulp_del_sa)(struct net_device *netdev, ...);

getsockopt(sk, SOL_OFFLOAD, GET_CAPS, &capabilities, ...)   ---->     int (*ulp_get_caps)(struct net_device *netdev, ...);

getsockopt(sk, SOL_OFFLOAD, OFFLOAD_OK, &status, ...)       ---->     bool (*ulp_offload_ok)(struct net_device *netdev);
                                                        };
```

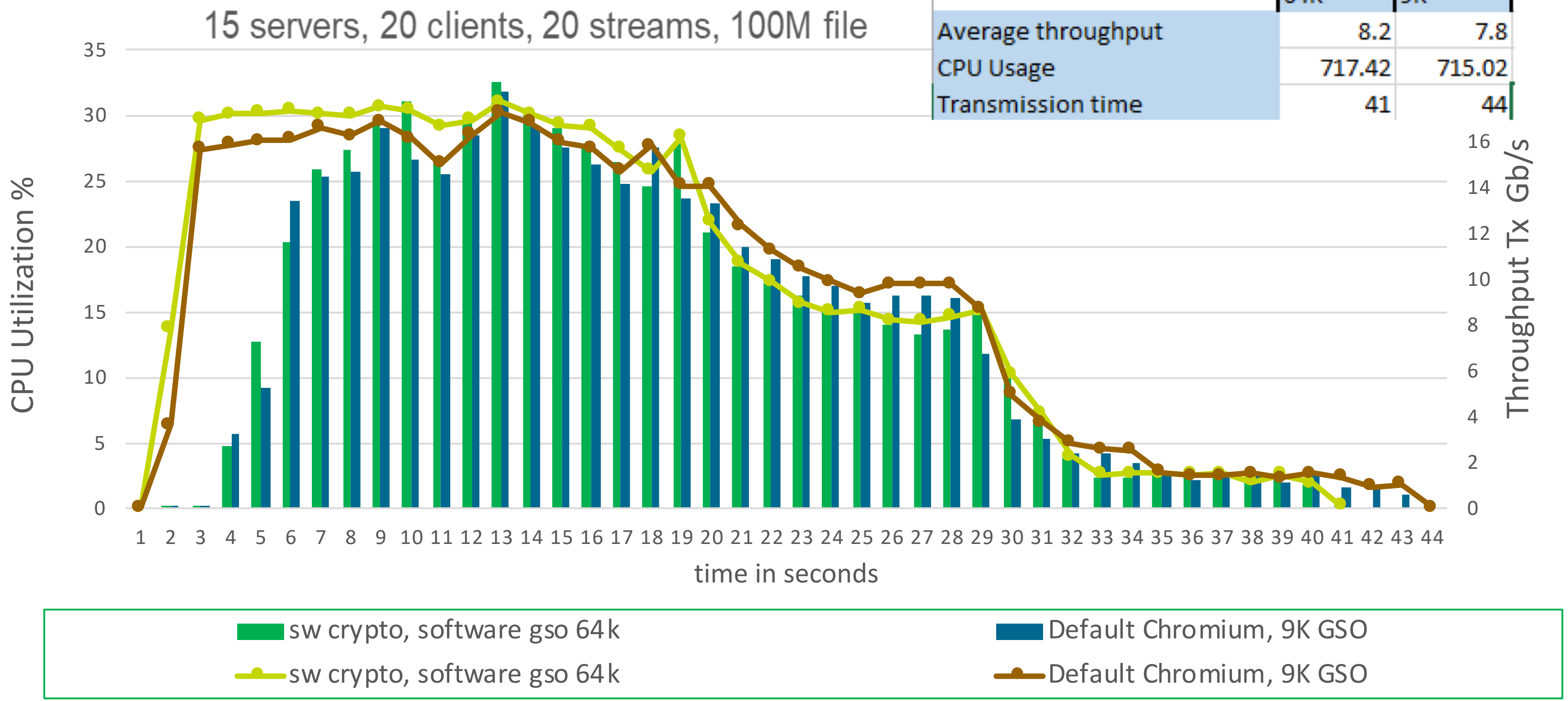
Segmentation Offload (USO)



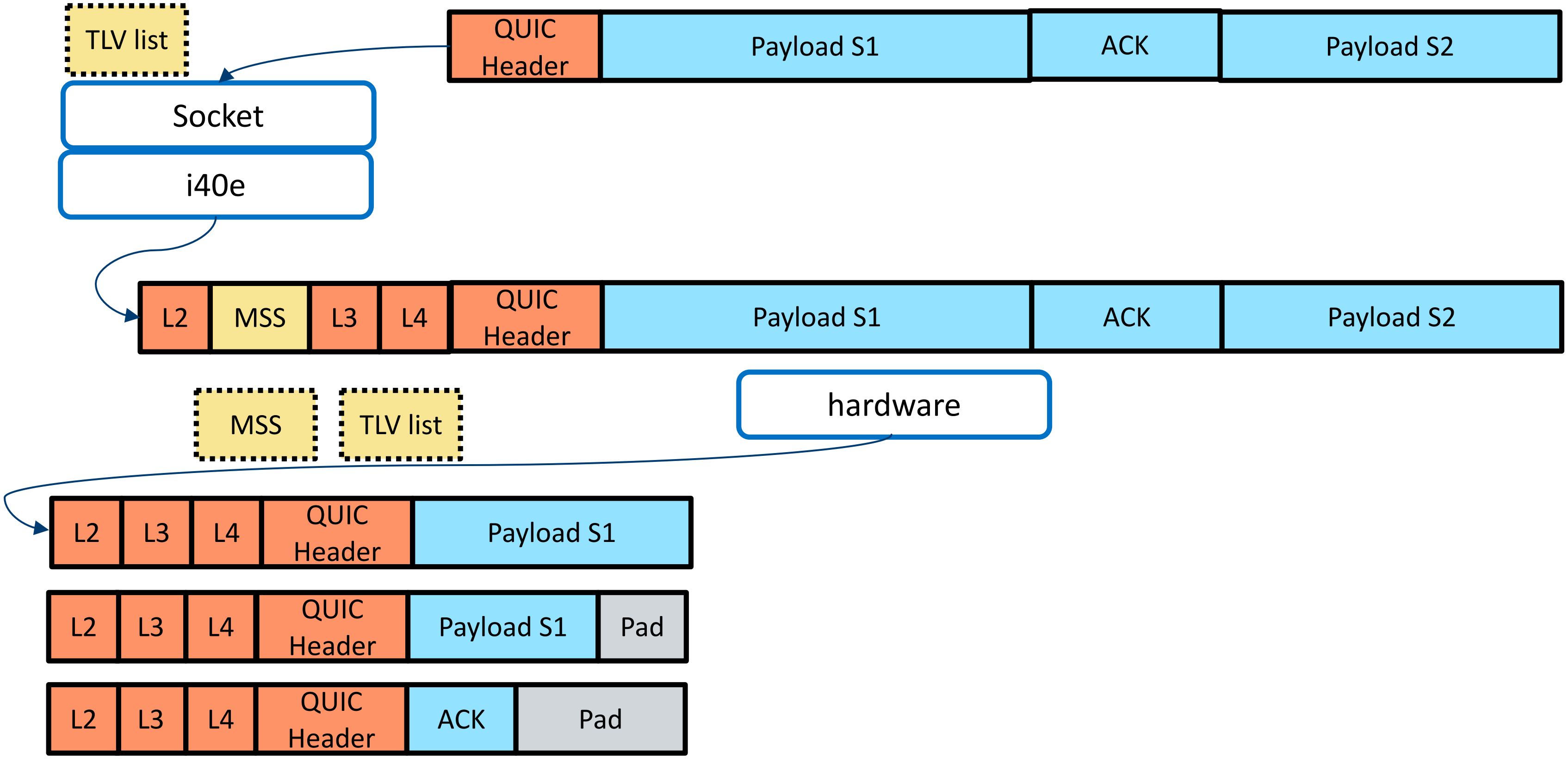

```
cmsg->cmsg_level = SOL_SEG_OFFLOAD;
cmsg->cmsg_type = SET_MSS;
*CMSG_DATA(cmsg) = mss;
```

Comparing 9K GSO with 64K GSO

	SW Crypto SW GSO 64K	SW Crypto SW GSO 9K
Average throughput	8.2	7.8
CPU Usage	717.42	715.02
Transmission time	41	44



Segmentation Offload like TCP



Segmentation Options

- Option 1: Only segment a single stream in a GSO offload
- Option 2: Limit the segmentation to only use ACK and stream frames
  - Only a single ACK is present at the start in the offloaded buffer

- TLV meta data to present the object list
  - Kernel folks pushed back on sending a large meta – data
- Having a length in the ACK would help
  - Small surgical change to simplify reduce the TLV style meta data

More Discussion?

Detailed discussion today at 4pm @ Congress Hall 3

Ingress Metadata

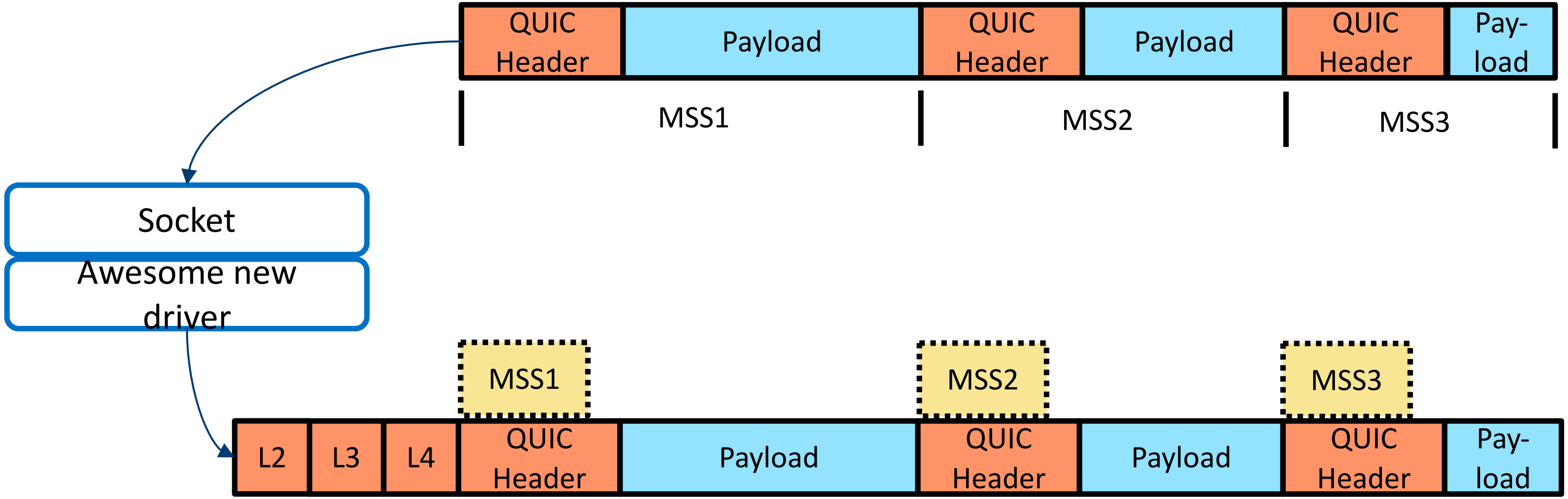
- Agent passes status to driver: authentication status, decryption status, protocol errors
- How does the driver communicate that status to the stack per packet?
  - Store status in control buffer in skb
  - Create cmsg header once skb hits socket layer
  - Stack uses recvmsg and extracts metadata from cmsg header

Flexible Interfaces for Flexible HW

- Abstract nonce from HW, send multiple MSS
  - Sent per segment via cmsg headers from the user space stack?

```
int send_quic(socket,
              quic_header = {flags, CID},
              nonces = [nonce1, nonce2, nonce3],
              quic_data = [
                    {QUIC_segment_1, mss1},
                    {QUIC_segment_2, mss2},
                    {QUIC_segment_3, mss3}],
              npackets = 3);
```

Segmentation Offload – Future Interface




```
for (i = 0; i < numpkts; i++) {
        cmsg->cmsg_level = SOL_SEG_OFFLOAD;
        cmsg->cmsg_type = SET_MSS;
        *CMSG_DATA(cmsg) = mss[i];
}
```

Takeaways

- Offload saves **~16%** CPU Usage, improves throughput by ~**32%** in certain test cases
- Segmentation interface and crypto interface are independent
- Crypto offload is impossible without an interface to get crypto parameters from user space to hardware
- A generic interface can enable crypto and segmentation offloads for other protocols
- Opens
  - What is the best way to bind a socket to an interface?
  - Is there a better way to do ingress metadata?
  - Are there other protocols that could make use of such an interface?

THANK YOU!

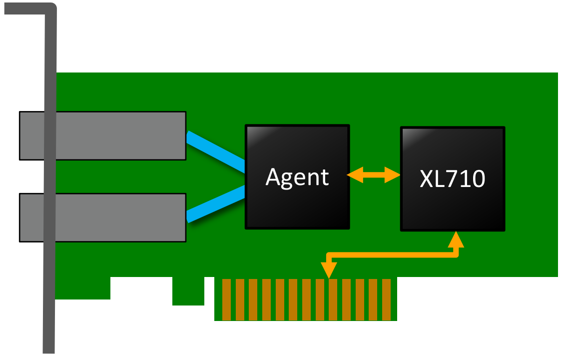
BACKUP

Programming the HW – User space to Driver

```
                                                        struct ulp_offload_devops {
setsockopt(sk, SOL_OFFLOAD, INIT_DEVICE, &init, ...)       ---->     int (*ulp_ofload_init)(struct net_device *netdev,
                                                                                      struct ulp_offload_init *init);
setsockopt(sk, SOL_OFFLOAD, ADD_SA_{TX|RX}, &addsa, ...)    ---->     int (*ulp_add_sa)(struct net_device *netdev,
                                                                                 struct ulp_sa_context *sa,
                                                                                 struct ulp_crypto_info *crypto_info);
setsockopt(sk, SOL_OFFLOAD, UPDATE_SA_{TX|RX}, &upsa, ...)  ---->     int (*ulp_update_sa)(struct net_device *netdev,
                                                                                    struct ulp_sa_context *sa,
                                                                                    struct ulp_sa_update *update);
setsockopt(sk, SOL_OFFLOAD, DEL_SA_{TX|RX}, &delsa, ...)    ---->     int (*ulp_del_sa)(struct net_device *netdev,
                                                                                 struct ulp_sa_context *sa);
getsockopt(sk, SOL_OFFLOAD, GET_CAPS, &capabilities, ...)   ---->     int (*ulp_get_caps)(struct net_device *netdev,
                                                                                   struct ulp_offload_caps *caps);
getsockopt(sk, SOL_OFFLOAD, OFFLOAD_OK, &status, ...)       ---->     bool (*ulp_offload_ok)(struct net_device *netdev);
                                                        };
```

Connectivity with our QUIC Agent

- No separate control plane for Configuration and Metadata
- All control data has to go through the MAC to get to the Agent
- Use one L2 tag to denote Control packets
- Different L2 tag to insert Metadata into a packet



- See https://www.netdevconf.org/2.2/slides/klassert_ipsec_workshop03.pdf for more info about agent connectivity

SW segmentation – 9K vs 64K