Offloading Encryption to QUIC Enabled NICs

Andy Gospodarek – Software Architect, Broadcom
Eric Davis – System Architect, Broadcom
Eric Spada – Distinguished Engineer, Broadcom
Overview of QUIC
Transport protocol that runs over UDP
Utilizes TLS1.3 for handshake/key exchange
Transmit and Receive Connection IDs are used to identify unique connections
Encryption is self-contained within each packet
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Percent Packets</th>
<th>Packets</th>
<th>Percent Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame</td>
<td>100.0</td>
<td>36214</td>
<td>100.0</td>
</tr>
<tr>
<td>Ethernet</td>
<td>100.0</td>
<td>36214</td>
<td>1.6</td>
</tr>
<tr>
<td>Address Resolution Protocol</td>
<td>4.5</td>
<td>1631</td>
<td>0.2</td>
</tr>
<tr>
<td>Internet Protocol Version 4</td>
<td>95.5</td>
<td>34569</td>
<td>2.0</td>
</tr>
<tr>
<td>Transmission Control Protocol</td>
<td>14.6</td>
<td>5289</td>
<td>8.1</td>
</tr>
<tr>
<td>Data</td>
<td>0.0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>Malformed Packet</td>
<td>0.1</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>Transport Layer Security</td>
<td>6.8</td>
<td>2447</td>
<td>7.6</td>
</tr>
<tr>
<td>User Datagram Protocol</td>
<td>80.8</td>
<td>29276</td>
<td>0.7</td>
</tr>
<tr>
<td>Data</td>
<td>4.1</td>
<td>1500</td>
<td>2.6</td>
</tr>
<tr>
<td>Domain Name System</td>
<td>0.4</td>
<td>160</td>
<td>0.0</td>
</tr>
<tr>
<td>Multicast Domain Name System</td>
<td>0.0</td>
<td>14</td>
<td>0.0</td>
</tr>
<tr>
<td>QUIC IETF</td>
<td>76.2</td>
<td>27600</td>
<td>84.8</td>
</tr>
<tr>
<td>Simple Service Discovery Protocol</td>
<td>0.0</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>Internet Protocol Version 6</td>
<td>0.0</td>
<td>14</td>
<td>0.0</td>
</tr>
<tr>
<td>User Datagram Protocol</td>
<td>0.0</td>
<td>14</td>
<td>0.0</td>
</tr>
<tr>
<td>Multicast Domain Name System</td>
<td>0.0</td>
<td>14</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Hardware offload good; software only bad
Checksum Offload, LRO/GRO, GSO, Tunnel Offload, nTuple filtering, HW Timestamping, kTLS, TC, ...
What could QUIC hardware offload look like?
Leverage experience from kTLS Offload to support QUIC Offload
quic kernel module with userspace/netlink and driver interface
Tracks flow, connection ID, direction and QUIC session/crypto information for offload
#define QUIC_MAX_CONN_ID_LEN 20
#define QUIC_MAX_PKT_NUM_LEN 8

struct quic_flow_ctx {
    u8 conn_id_len;
    u8 conn_id[QUIC_MAX_CONN_ID_LEN];
    u8 pkt_num[QUIC_MAX_PKT_NUM_LEN];
    u8 key_phase;
    u8 cipher;
    u16 xid;

    union {
        struct cipher_aes_gcm_128 aes_gcm_128;
        struct cipher_aes_ccm_128 aes_ccm_128;
        struct cipher_chacha20_poly1305 chacha20_poly1305;
    } cipher_info;
};
Receives offload requests via netlink
# ip crypto show mode quic
4ed45802: rx 10.0.0.1:53227 10.0.0.100:443 cid e45ea092c337f4eb cipher aes_gcm_128 dev ens7f0np0
Adds offloaded connection info to drivers and hardware
enum quic_offload_dir {
    QUIC_OFFLOAD_DIR_RX,
    QUIC_OFFLOAD_DIR_TX
};

int (*quic_dev_add)(struct net_device *netdev,
                     struct sock *sk,
                     struct quic_flow_ctx *flow_ctx
                     enum quic_offload_dir direction);
Returns XID to caller if flow can be offloaded
QUIC userspace components
Application/Library does initial connection setup/handshake
If application wants to offload; send request to offload via netlink
Application stores XID returned in netlink ext_ack cookie
Transmit datapath
XID passed into cmsghdr if data is to be encrypted by hardware
XID added to sk_buff extension in socket layer
Network driver formats TX buffer descriptor as needed based on sk_buff extension data
Hardware sends encrypted packet to peer
Receive datapath
Frames will be decrypted by hardware and validation reported in buffer descriptor
Driver will populate sk_buff extension with decrypt and authentication status from buffer descriptor
Socket will set cmsghdr and pass back to application, indicating XID, encryption, validation status
Sounds simple, right?
Conclusions and Next Steps
QUIC is well suited for device offload
Modifications to kernel and network drivers are clear
QUIC device offload will be part of bnxt_en driver updates
QUIC Libraries may take more work to increase adoption and usage
FIN