RDMA and QP timeouts

General resources

NOTES:

- Most of settings can be tracked from application to verbs API down to PRM and IB spec.
- API that can be used by an application: either verbs based or rdma_cm based IP, both available in user space and kernel.
  The API signature and capabilities are slightly different between user space and kernel.

1. For user space - all up to date systems use rdma-core upstream based solution.
   Thus man pages of rdma-core is the place to look:
   https://github.com/linux-rdma/rdma-core/tree/master

2. RDMA Aware programming guide is now available in html format online
   https://docs.mellanox.com/display/RDMAAwareProgrammingv17

3. RDMA mojo – outdated blog, but has very good basic answers for RDMA
   https://www.rdmamojo.com/

4. PRM – should be accessible under NDA

5. IB spec

Timeout relevant parameters

Four main parameters that impact the timeout are explained very well in RDMAMojo and other resources per below summary:

1. **min_rnr_timer** - if a receiving QP doesn't have WQE - enough buffers to receive requests, this timer will expire, and a NACK will be sent to the other side.
Default: 0
Relevance: QP type: RC and DC.
IB spec reference:
"11.2.5.2 MODIFY QUEUE PAIR" – look for key words "Minimum RNR NAK"
"Table 64 Connection Parameters by Transport Service" – "RNR NAK retry time" for QP relevance

2. **timeout** - minimum time QP waits for any kind of response (ack or nack) until it tries to retransmit the packet.
The value is numeric with equation converting to usec.
An application must set the qp timeout value during QP transition from RTR to RTS.
If value is not set, FW will apply the default, which is the minimal value – at this time 16.
This minimal value can be seen in ibv_devinfo per below example:

```
ibv_devinfo -v | grep ack_delay
local_ca_ack_delay:          16
```

Default: If rdma_cm 19, If no rdma_cm – 16 from FW.
Relevance: QP type: RC
IB spec reference:
"12.7.34 LOCAL ACK TIMEOUT" - look for key words "ACK delay"
"Table 67 Packet Fields, Queue Parameters, and their Sources" - "Local ACK Timeout" – for QP relevance
Meaning: The delay value in microseconds is computed using 4.096us * 2^(local_ca_ack_delay).

3. **retry_cnt** - number of times that QP will try to resend packet before erroring out.
It means that each time, we will first wait for #2 - timeout to expire then will try to resend - totally number of times of this counter.
Default: 7
Relevance: QP type: RC and XRC
IB spec reference:
"12.7.38 RETRY COUNT" - look for key words "retry count"
"9.9.2.1.1 REQUESTER ERROR RETRY COUNTERS"
"Table 67 Packet Fields, Queue Parameters, and their Sources" – "Error Retries" - for QP relevance
"9.7.5.2.8 RNR NAK"

4. **rnr_retry** - number of times that will try to resend if rnr nack was received.
Meaning the other side had min_rnr_timer expired, sent rnr_nack and as a result this counter will be used.
Default: 7
Relevance: QP type: RC
IB spec reference:
"12.7.39 RNR RETRY COUNT" - look for key words "rnr retry count"
"9.9.2.1.1 REQUESTER ERROR RETRY COUNTERS"
"Table 67 Packet Fields, Queue Parameters, and their Sources" - "RNR Retry counter" – for QP relevance

Controlling the parameters in API

- Verbs
  Above all four parameters' default values can be changed when calling to **ibv_modify_qp** during RTR to RTS transition.
  Also there is an explanation in:
  - Rdma-core libibverbs github man pages of ibv_modify_qp
  - RDMA aware programming guide – "RTR to RTS"
  - RDMAmojo.

  Relevant API struct

```
struct ibv_qp_attr {
	enum ibv_qp_state	qp_state;
	…
	uint8_t		min_rnr_timer;
	uint8_t		port_num;
	uint8_t		timeout;
	uint8_t		retry_cnt;
	uint8_t		rnr_retry;
	…
};
```

  Example of user space application code can be seen in:
  - Rc_pingpog in libibverbs/examples
  - Perftest ctx_modify_qp_to_rts function

  Similar option exists in kernel, function **ib_modify_qp** (without v).

- RDMA_CM

  In case rdma_cm is used, only some of the above parameters can be controlled by use of RDMA_CM API.

  User space

  #3 – **retry_cnt** (default value 7) and #4 **rnr_retry** (default value 7) from above, can be provided in rdma_cm API.
  These parameters can be modified by use of **rdma_connect** function, per below guidance:

  1. rdma_connect man pages in rdma-core github
  2. rdma_connect in RDMA Aware programming guide

  Relevant API struct

```
struct rdma_conn_param {
    const void *private_data;
    uint8_t private_data_len;
    uint8_t responder_resources;
    uint8_t initiator_depth;
    uint8_t flow_control;
    uint8_t retry_count;                /* ignored when accepting */
    uint8_t rnr_retry_count;
    /* Fields below ignored if a QP is created on the rdma_cm_id. */
    uint8_t srq;
    uint32_t qp_num;
};
```

  Example application code can be seen in:

  - Rping in librdmacm/examples, function rping_init_conn_param
  - Perftest

  #2 - timeout (default for RoCE is 19) can be controlled by use of **rdma_set_option** function, per below guidance:

  1. Rdma_set_option man pages in rdma-core github

  Relevant API

```
enum {
    RDMA_OPTION_ID_TOS       = 0,          /* uint8_t: RFC 2474 */
    RDMA_OPTION_ID_REUSEADDR = 1,          /* int: ~SO_REUSEADDR */
    RDMA_OPTION_ID_AFONLY    = 2,          /* int: ~IPV6_V6ONLY */
    RDMA_OPTION_ID_ACK_TIMEOUT = 3  /* uint8_t */
};
```

  Example application code can be seen in:

  - cma_tos in librdmacm/examples , function addr_handler

  Kernel space

  Same function and API exists in kernel as above in user space for controlling **retry_cnt** and **rnr_retry** values.

  In addition to that there is an option to control **#2 - timeout** (default for RoCE is 19) by use of **rdma_set_ack_timeout** function directly instead of rdma_set_option from user space.
  This function API can be seen in include/rdma/rdma_cm.h

__Equation – calculating actual timeout for QP__
Below is relevant for ConnectX-5 and on cards and might slightly be different depends on the FW version and other factors.

Below is assuming latest FW:

QP timeout = (<u>enforced min timeout</u>) * (<u>num retries configured</u>) * ~2(<u>average timeout multiplier</u>)

<u>“enforced min timeout” per try</u> = (4.096us * 2 ^ (qp_attr.timeout)) – per IB spec
qp_attr.timeout attribute has an enforced minimal value of 16, if we configure <= 16, 16 will be used.
If we configure > 16, FW uses the configured value.

<u>"num retries configured"</u> = qp_attr.retry_cnt

<u>“average timeout multiplier”</u> = enforced by FW