2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

Exploring UVM TLM2 based Sequence, Sequencer and Driver in UVM

N Goyal, J Refice

Nvidia Corporation

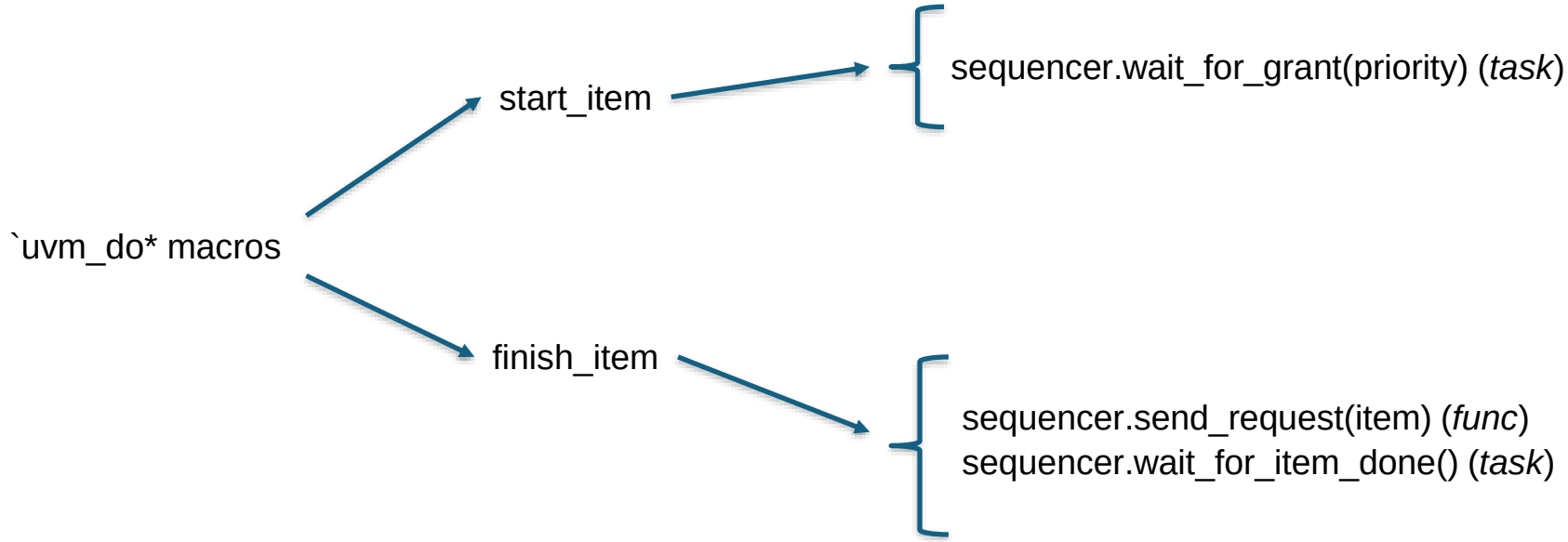
NVIDIA.

What is TLM1 and TLM2?

- TLM (Transaction Level Modeling)

	TLM1	TLM2
Data Passing	Pass by value	Pass by reference
Transaction	User-defined	tlm_generic_payload
Connection	Ports	Sockets
Features	Simple, message-passing	Completion semantics, phase/time awareness and interoperability

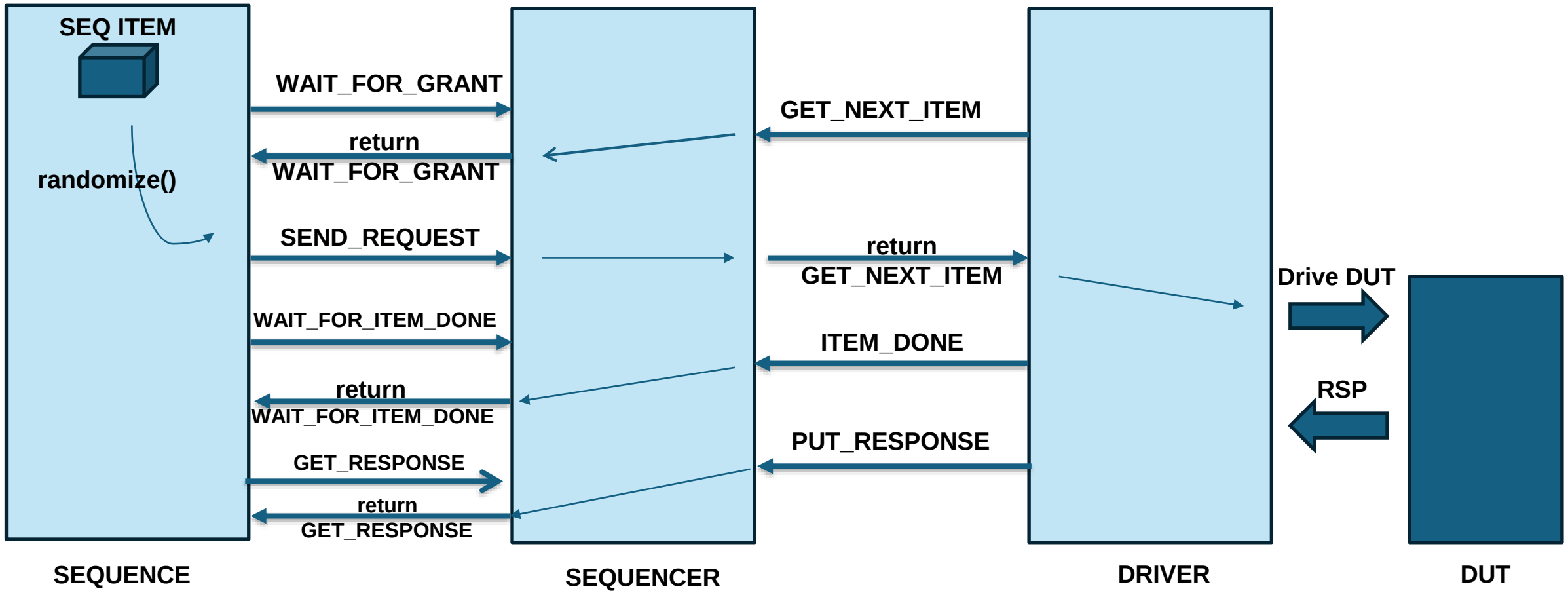
Sequence- Sequencer Communication



Sequencer – Driver Communication

- Special interface uvm_seq_if_base#(T1,T2)
- Methods:
  - get_next_item(), try_next_item()
  - item_done()
  - put_response()

What exists today?



Why couldn’t we use TLM1?

- Bi-directional handshake
- Late randomization
- Response Item

TLM2 Procotol

- Sockets
  - `uvm_tlm_initiator_socket`
  - `uvm_tlm_target_socket`
- Generic Payload
  - `uvm_tlm_generic_payload`
- Interfaces (blocking, non blocking)
  - Non –blocking (`nb_transport_fw/bw`) – with phase (`BEGIN_REQ, END_REQ, BEGIN_RESP, END_RESP`)
- Responses – `UVM_TLM_ACCEPTED, UVM_TLM_UPDATED, UVM_TLM_COMPLETED`

Why explore TLM2?

- Timing semantics
  - No delta cycles, non-blocking functions with begin-end
- Response mapping
  - Request and Response are a part of the payload
- Interoperability
  - Standardized communication

Proposed TLM2 protocol

- Initiator and Target Sockets
- New TLM2 payload class
- Use `nb_transport_fw/bw` interface
- Updated TLM2 phases

TLM2 payload class

```
class uvm_tlm2_sequence_payload #(type REQ=uvm_sequence_item, type
RSP=REQ) extends uvm_sequence_item;

// Request and Response item
REQ req_item;
RSP rsp_item;
//Add variable for the three calls
int pri_req = -1; // priority for wait_for_grant
bit lock_request = 0; // lock request for wait_for_grant
bit rerandomize = 0; // rerandomize request for send_request

endclass
```

Proposed TLM2 phases

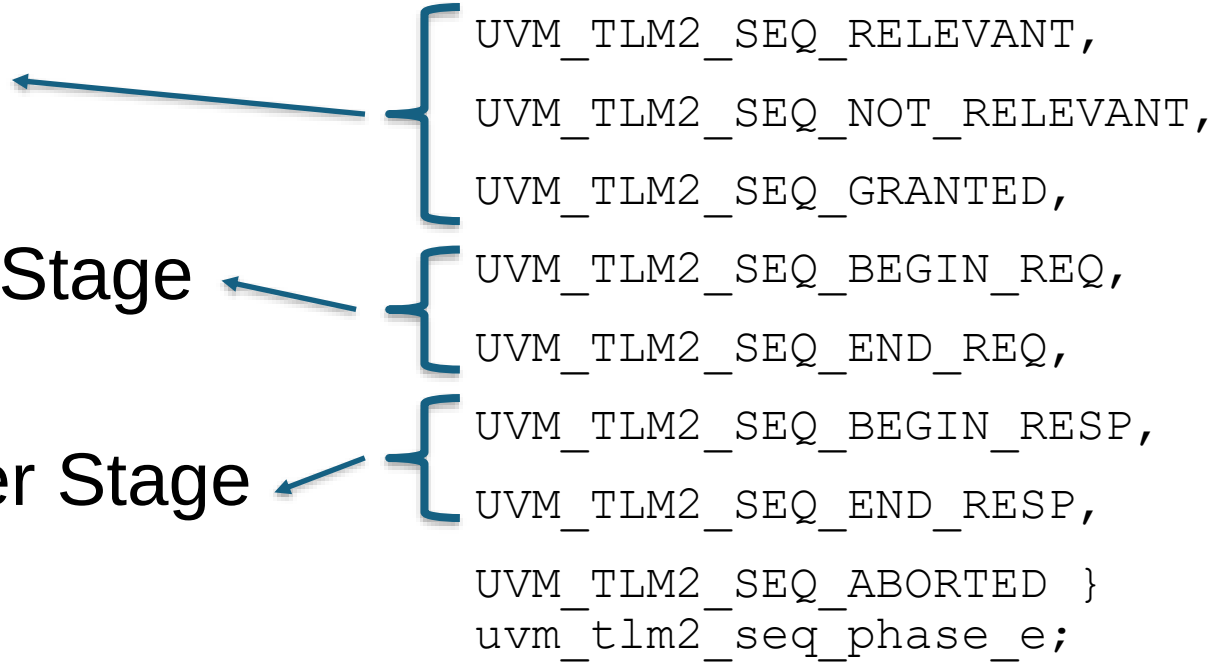
```
typedef enum {
UVM_TLM2_SEQ_RELEVANT,
UVM_TLM2_SEQ_NOT_RELEVANT,
UVM_TLM2_SEQ_GRANTED,
UVM_TLM2_SEQ_BEGIN_REQ,
UVM_TLM2_SEQ_END_REQ,
UVM_TLM2_SEQ_BEGIN_RESP,
UVM_TLM2_SEQ_END_RESP,
UVM_TLM2_SEQ_ABORTED } uvm_tlm2_seq_phase_e;
```

Stages of the proposed TLM2 protocol

- Arbitration Stage
- Request Transfer Stage
- Response Transfer Stage

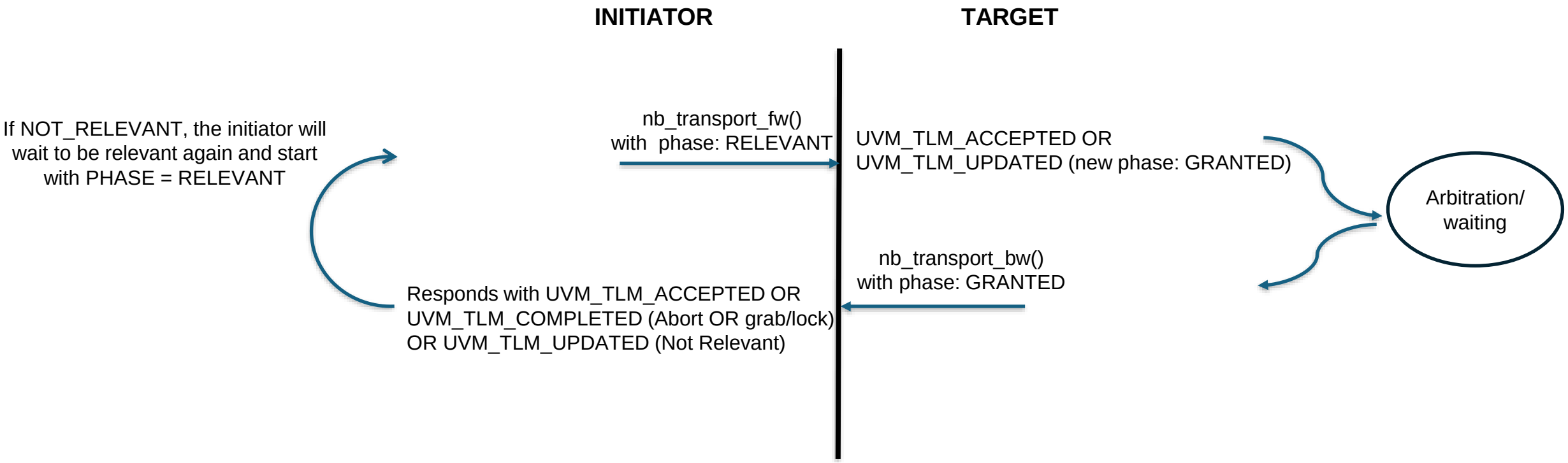
Proposed TLM2 protocol

- Arbitration Stage
- Request Transfer Stage
- Response Transfer Stage

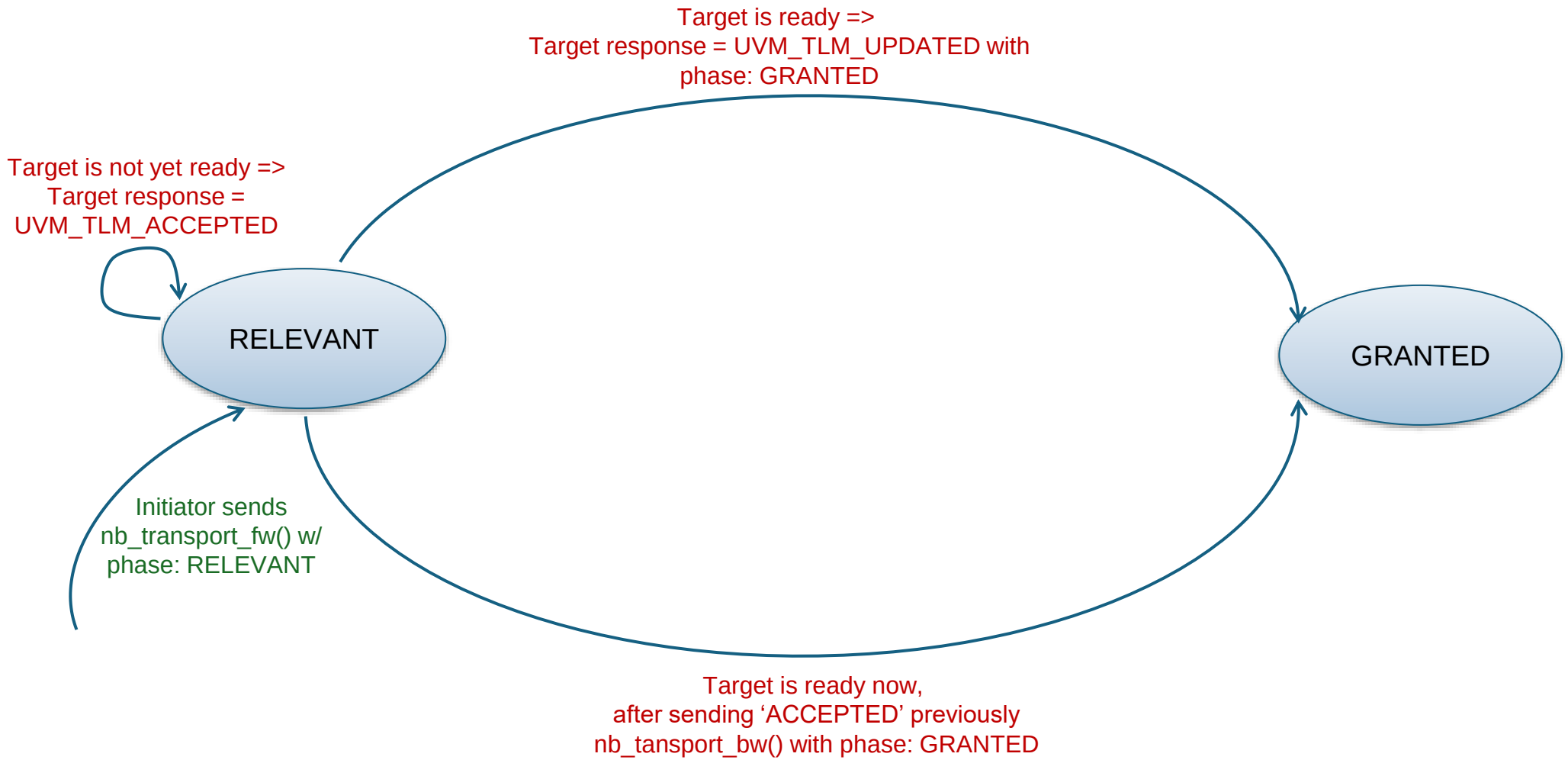


```
typedef enum {
UVM_TLM2_SEQ_RELEVANT,
UVM_TLM2_SEQ_NOT_RELEVANT,
UVM_TLM2_SEQ_GRANTED,
UVM_TLM2_SEQ_BEGIN_REQ,
UVM_TLM2_SEQ_END_REQ,
UVM_TLM2_SEQ_BEGIN_RESP,
UVM_TLM2_SEQ_END_RESP,
UVM_TLM2_SEQ_ABORTED }
uvm_tlm2_seq_phase_e;
```

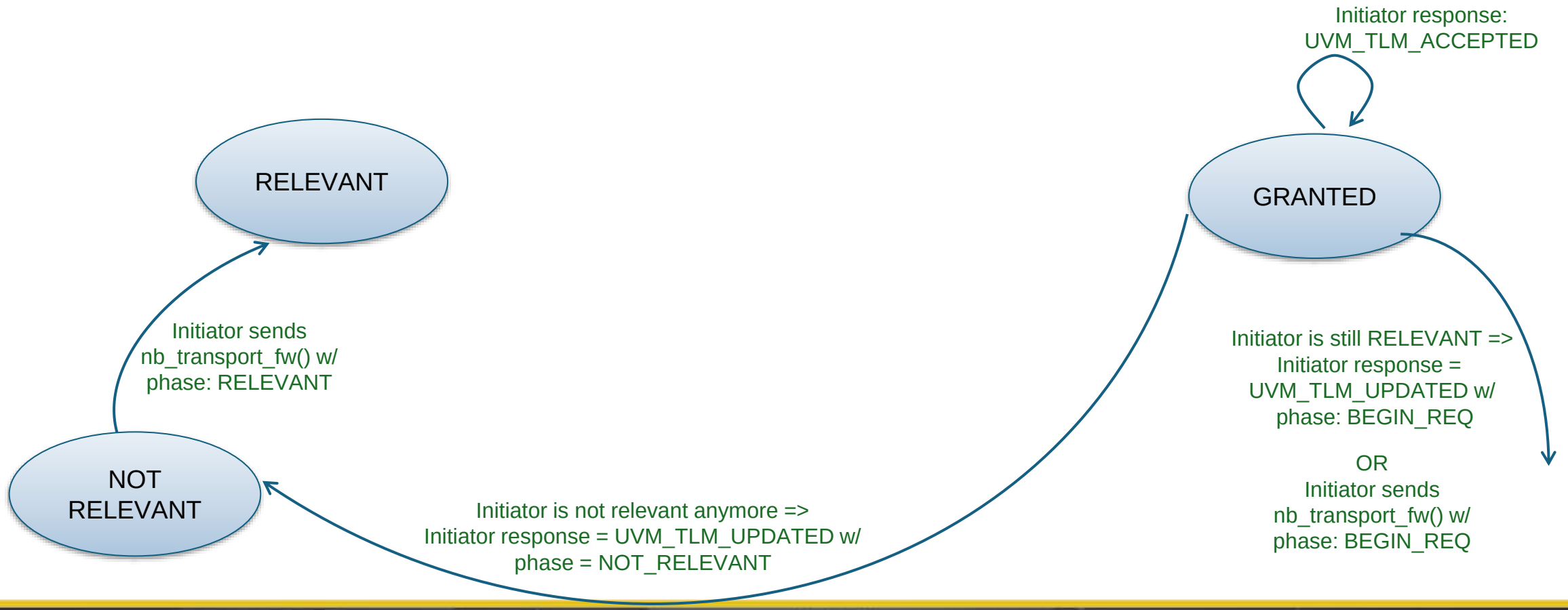
Arbitration Stage



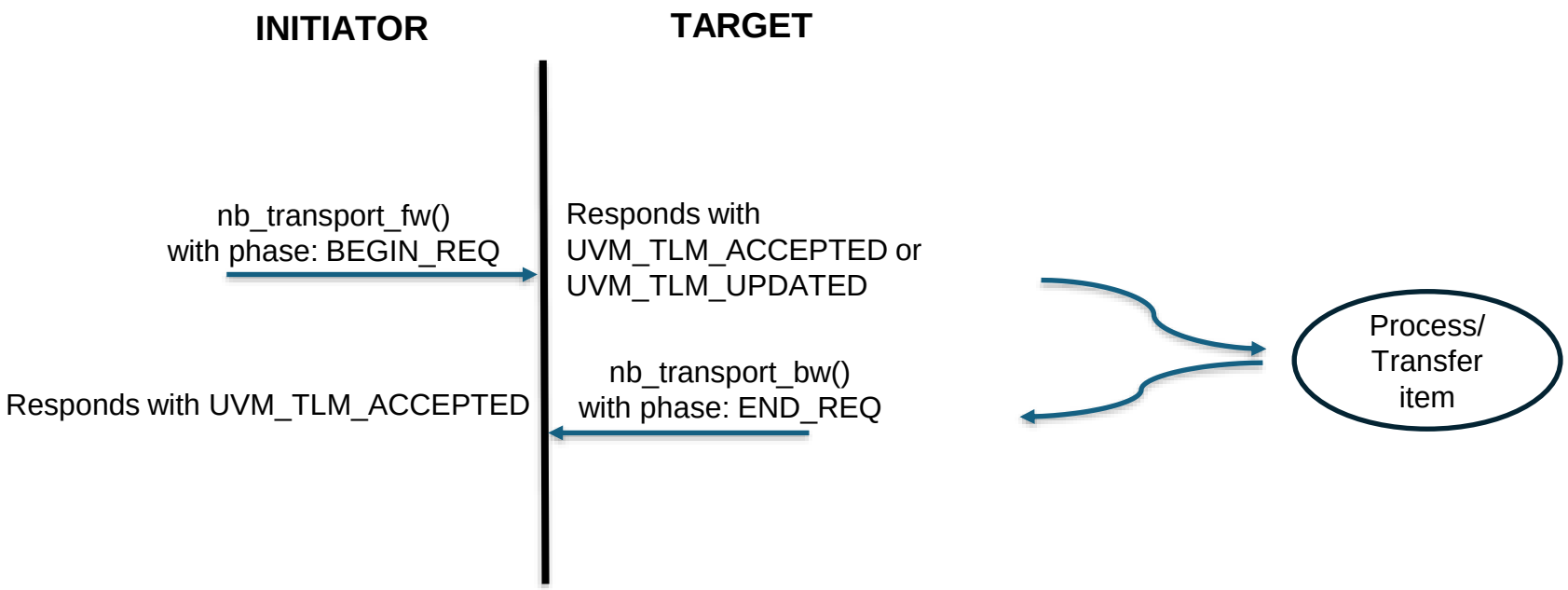
ARBITRATION STAGE



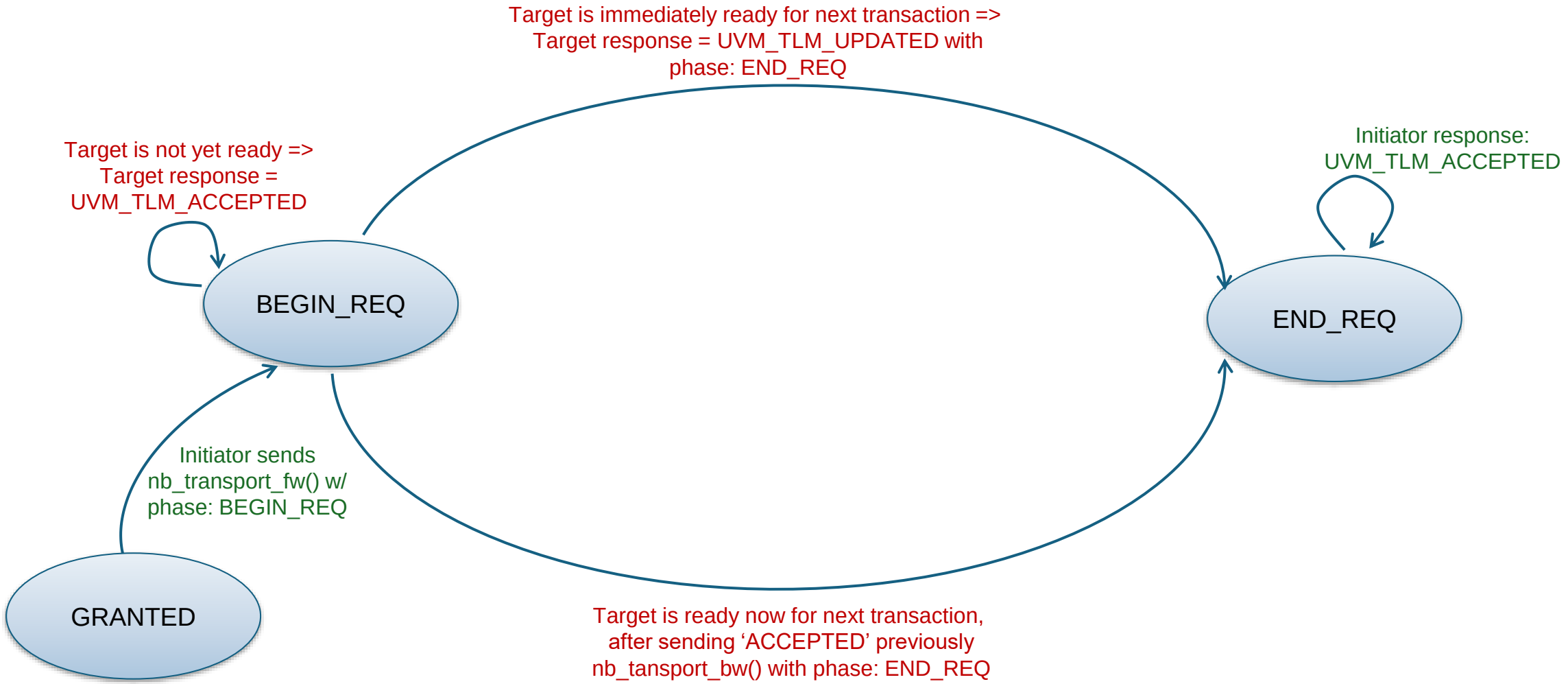
ARBITRATION STAGE



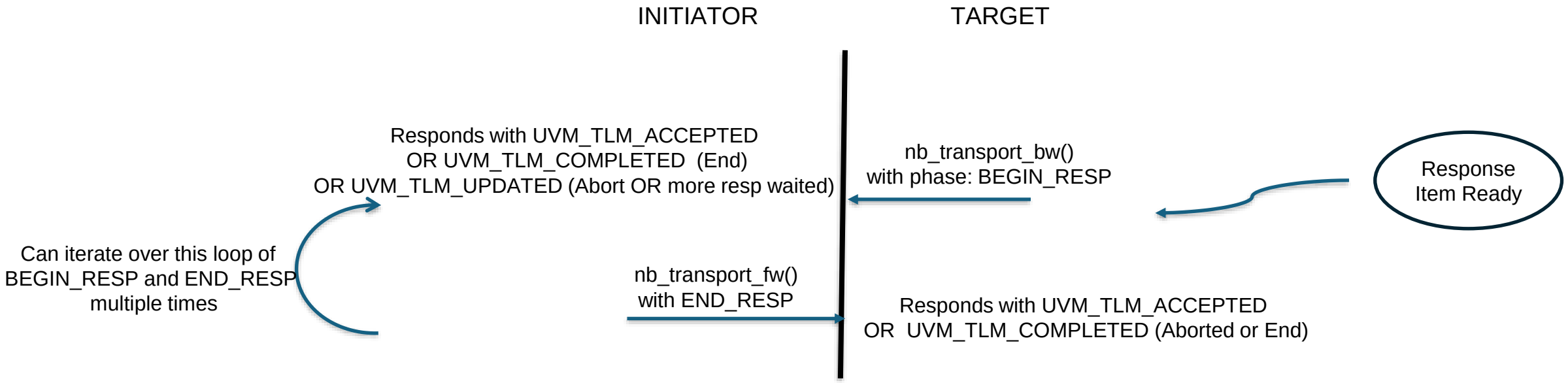
Request Transfer Stage



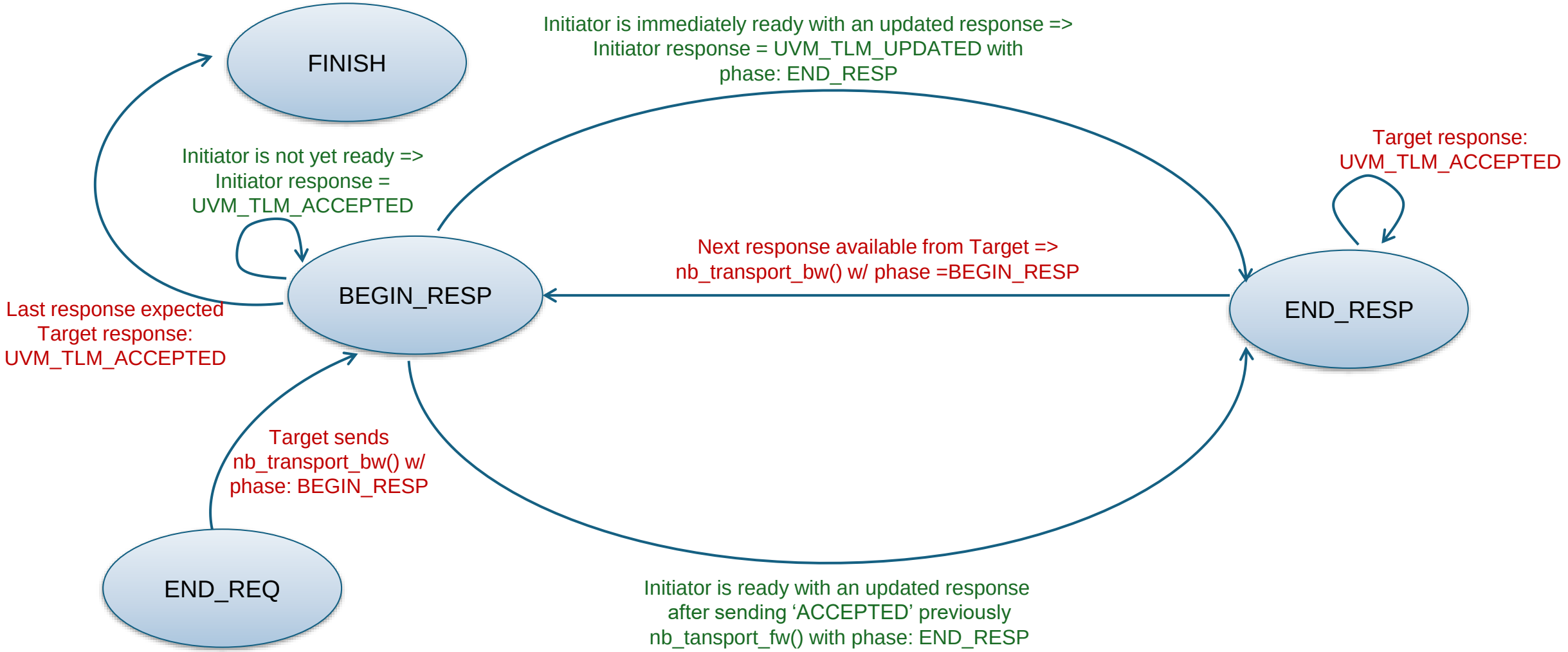
REQ TRANSFER STAGE



Response Transfer Stage

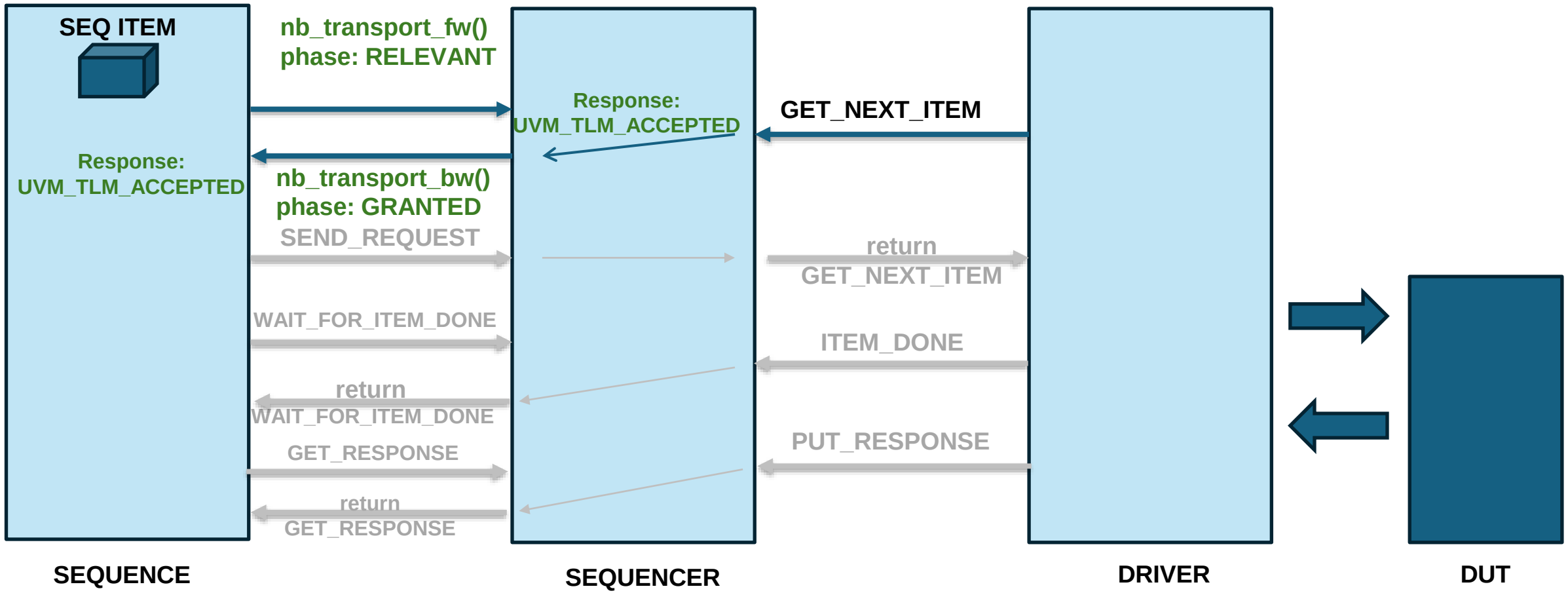


RESP TRANSFER STAGE

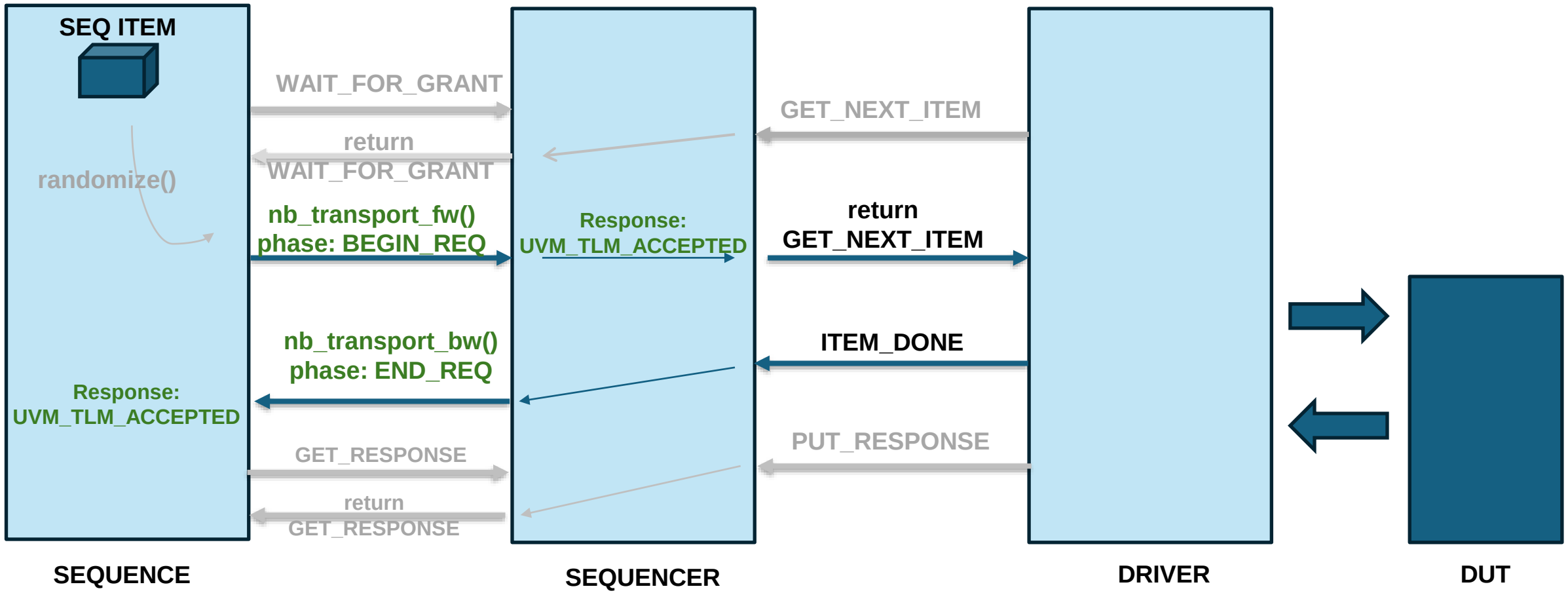


Mapping the proposed TLM2 protocol to the sequence-sequencer interaction

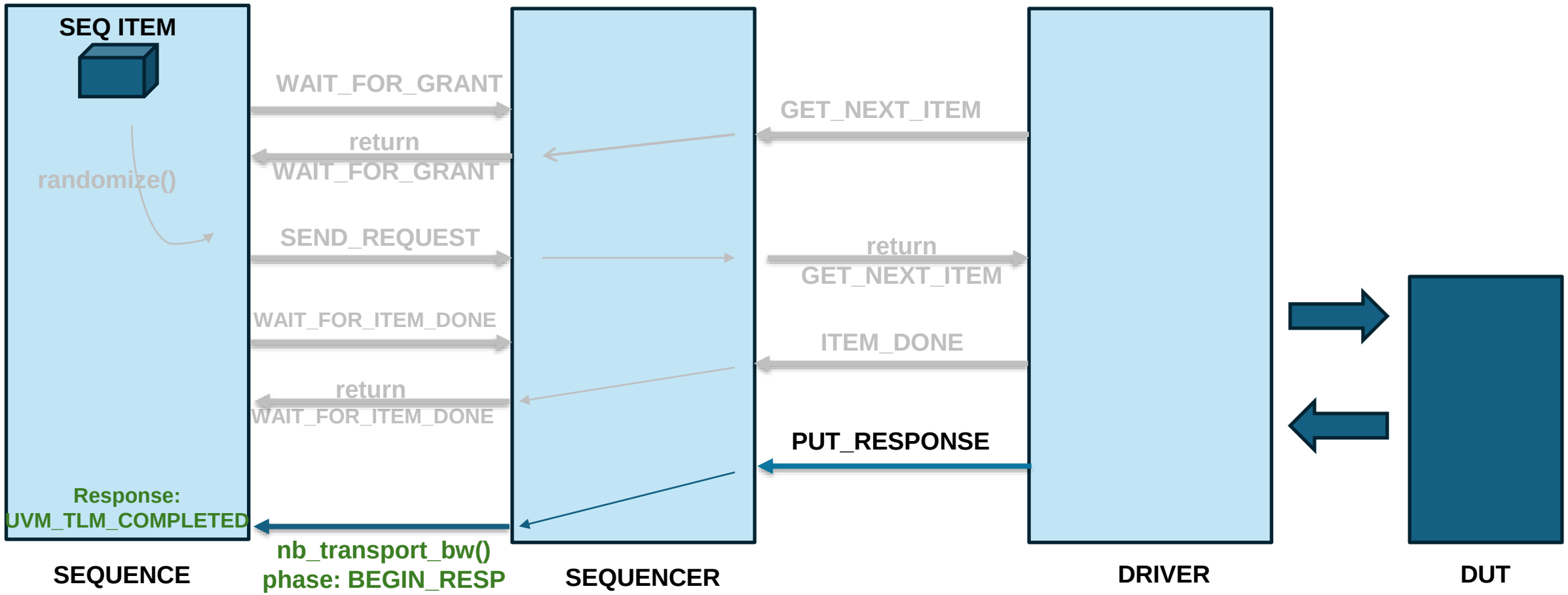
Mapping to Seq-Seqr

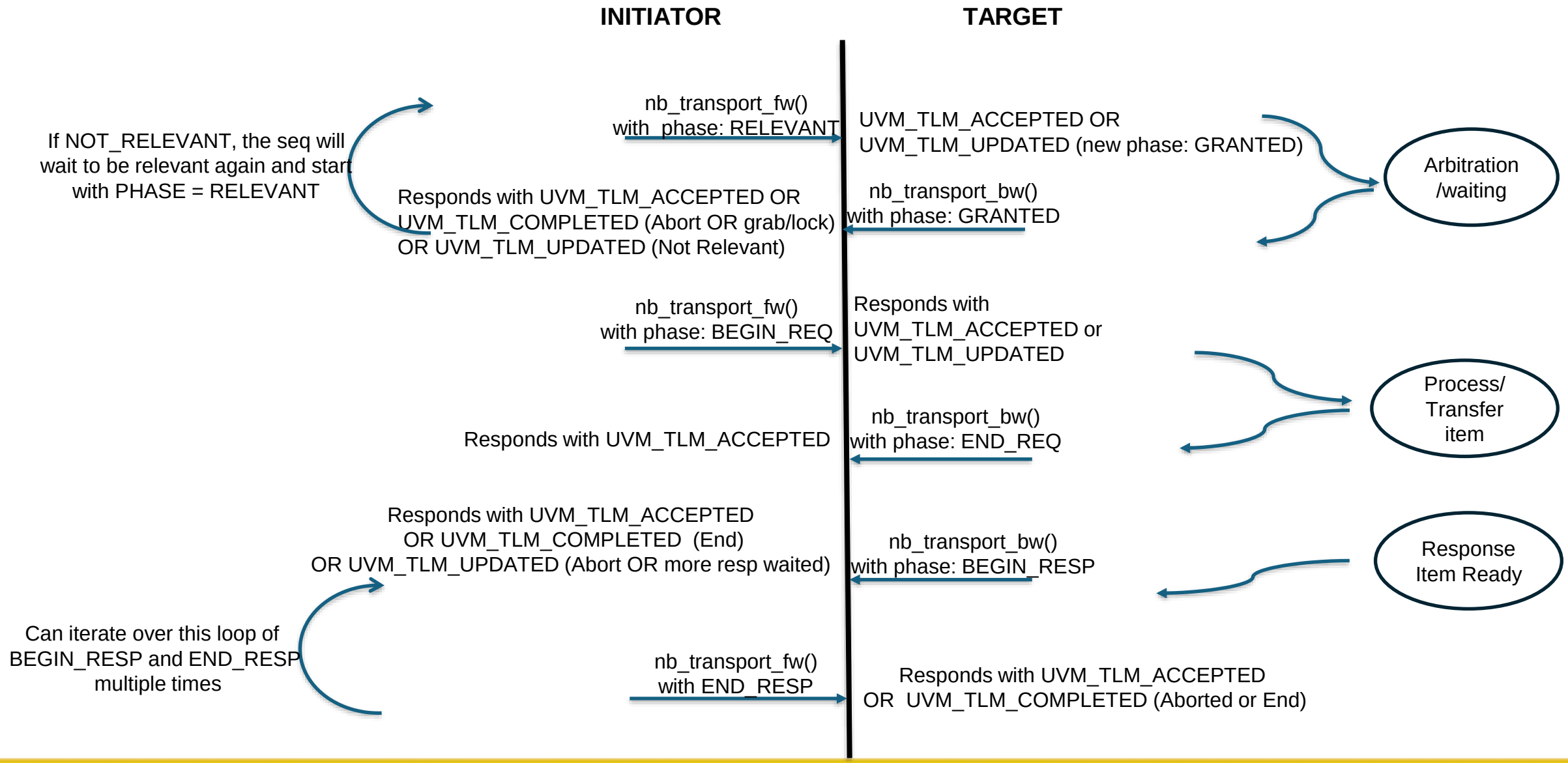


Mapping to Seq-Seqr



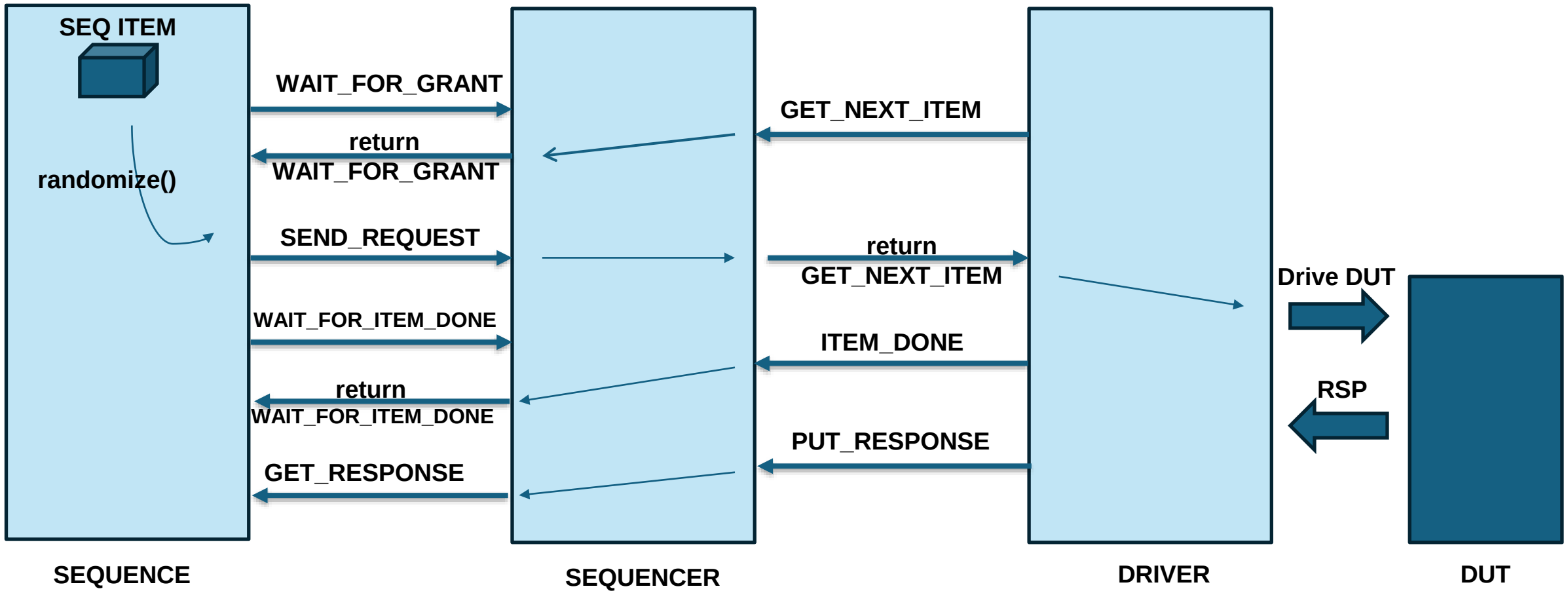
Mapping to Seq-Seqr



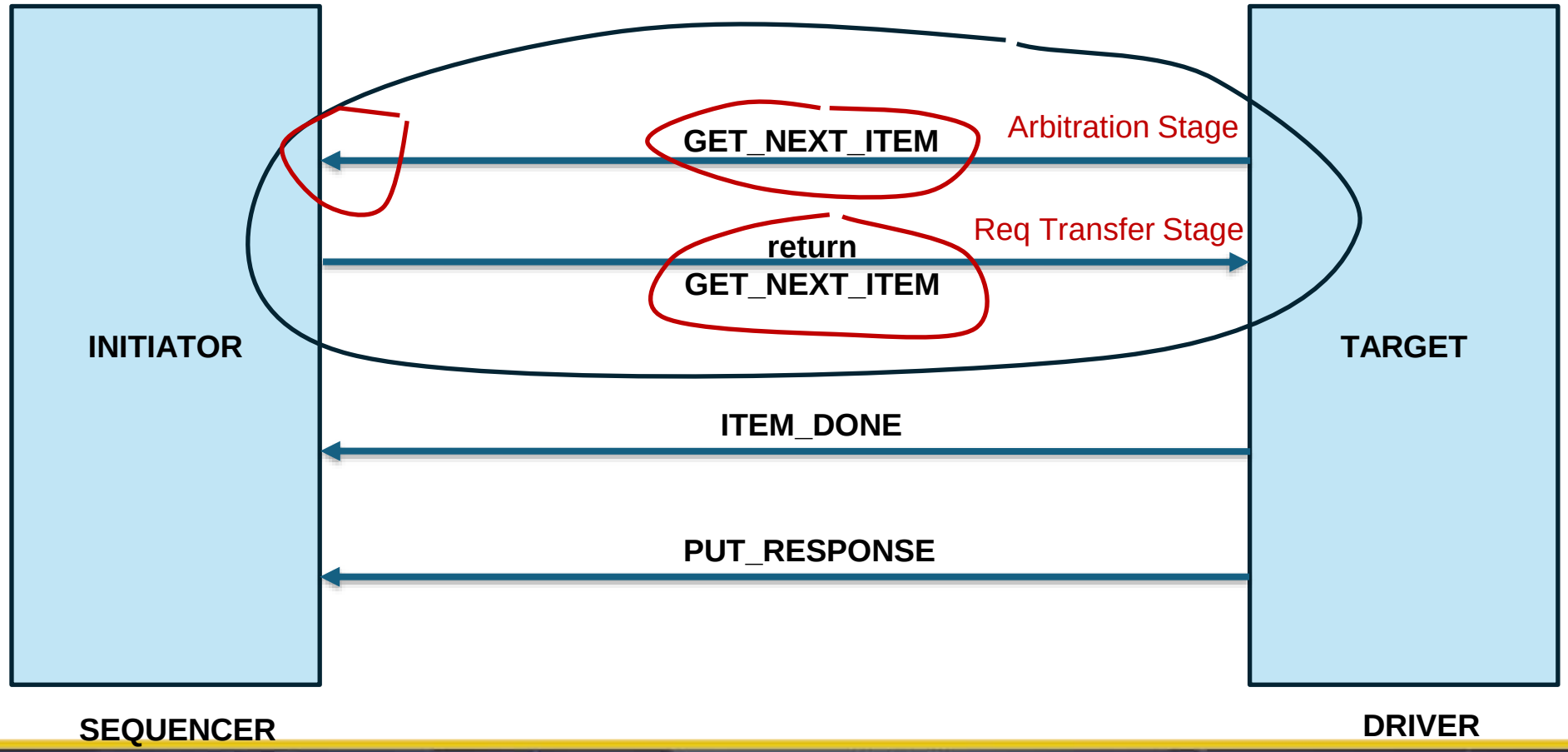

INITIATOR
TARGET
If NOT_RELEVANT, the seq will wait to be relevant again and start with PHASE = RELEVANT
nb_transport_fw() with_phase: RELEVANT
UVM_TLM_ACCEPTED OR UVM_TLM_UPDATED (new phase: GRANTED)
Arbitration /waiting
Responds with UVM_TLM_ACCEPTED OR UVM_TLM_COMPLETED (Abort OR grab/lock) OR UVM_TLM_UPDATED (Not Relevant)
nb_transport_bw() with phase: GRANTED
nb_transport_fw() with phase: BEGIN_REQ
Responds with UVM_TLM_ACCEPTED or UVM_TLM_UPDATED
Process/ Transfer item
Responds with UVM_TLM_ACCEPTED
nb_transport_bw() with phase: END_REQ
Responds with UVM_TLM_ACCEPTED OR UVM_TLM_COMPLETED (End) OR UVM_TLM_UPDATED (Abort OR more resp waited)
nb_transport_bw() with phase: BEGIN_RESP
Response Item Ready
Can iterate over this loop of BEGIN_RESP and END_RESP multiple times
nb_transport_fw() with END_RESP
Responds with UVM_TLM_ACCEPTED OR UVM_TLM_COMPLETED (Aborted or End)

Mapping the proposed TLM2 protocol to the driver-sequencer interaction

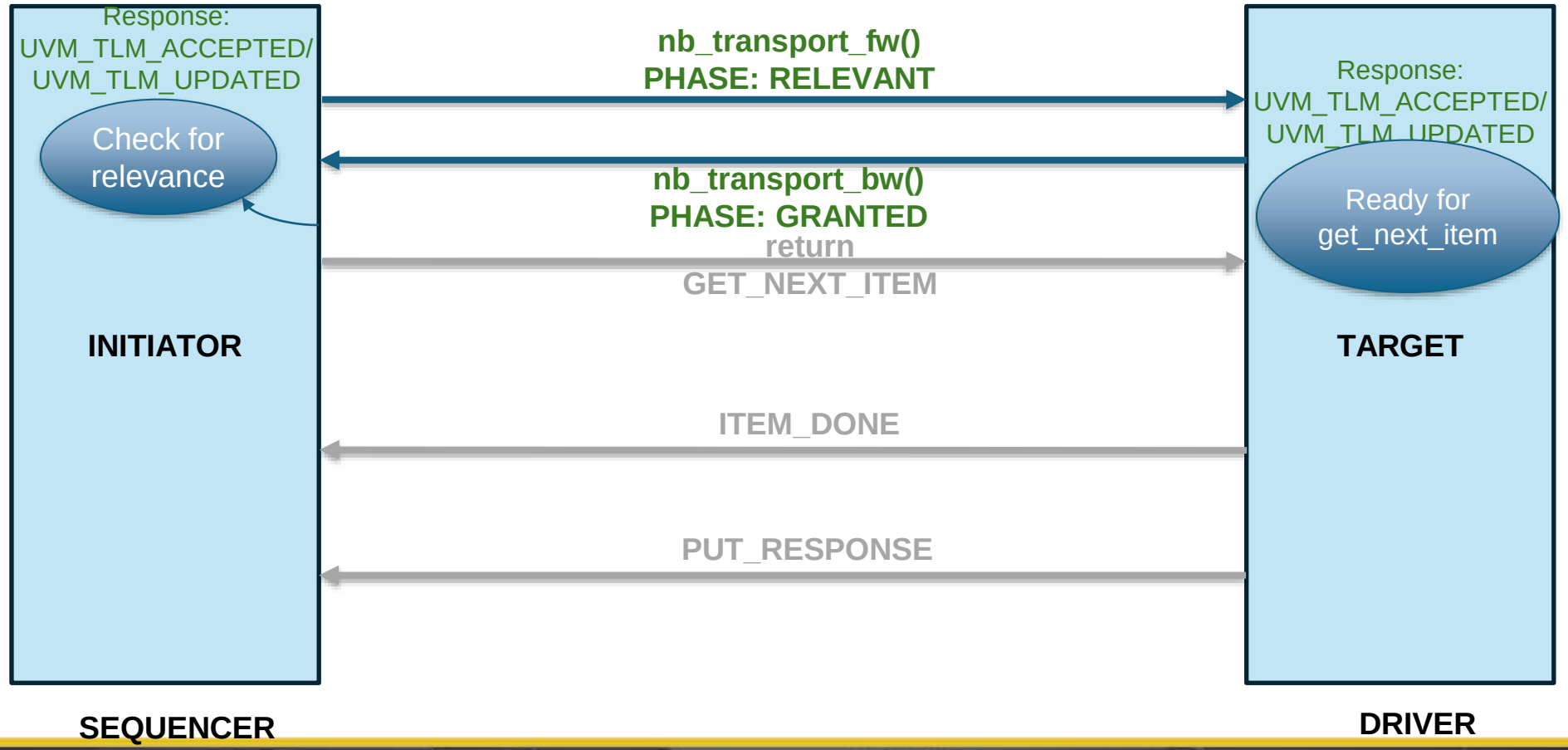
TLM2 protocol for Sequencer - Driver



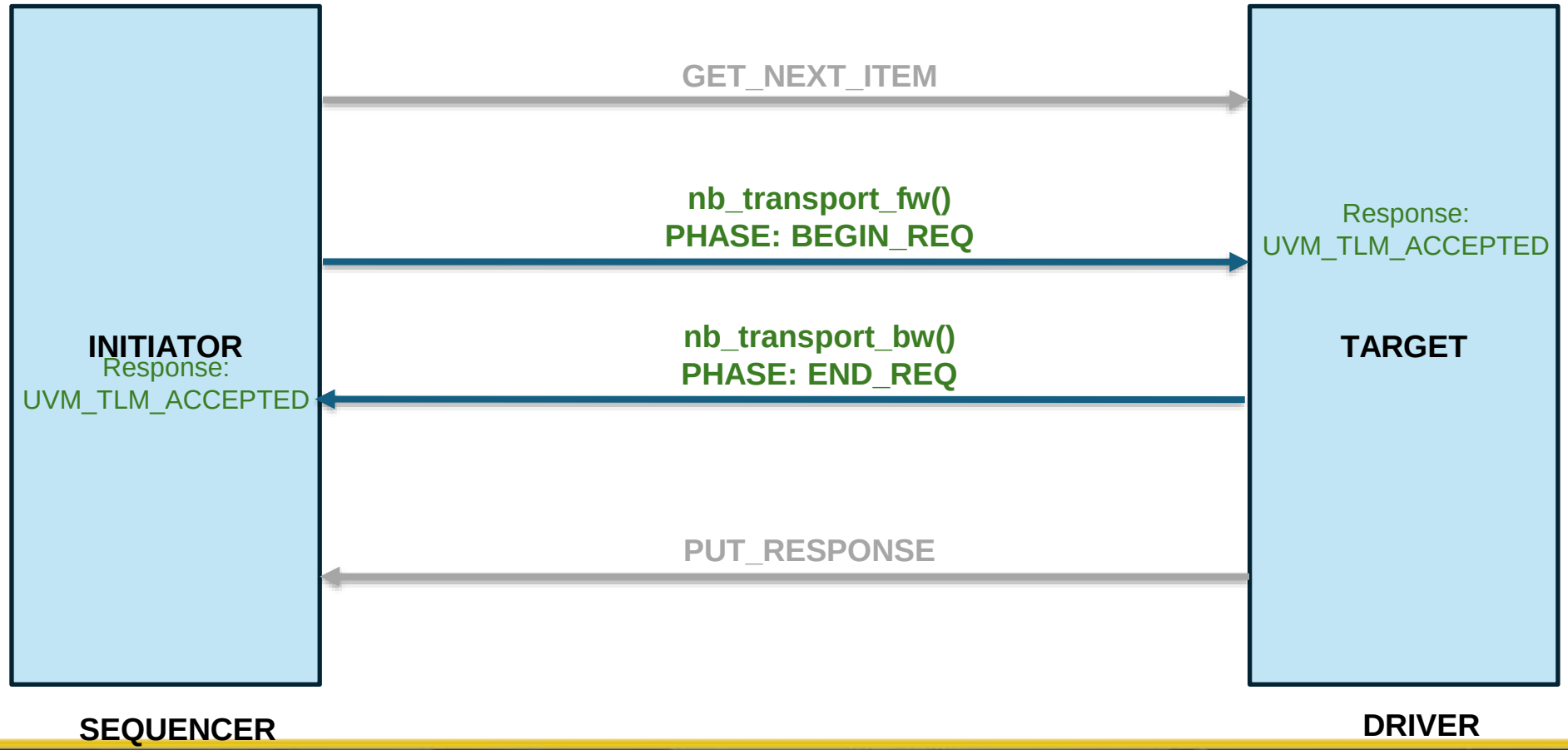
TLM2 protocol for sequencer-driver



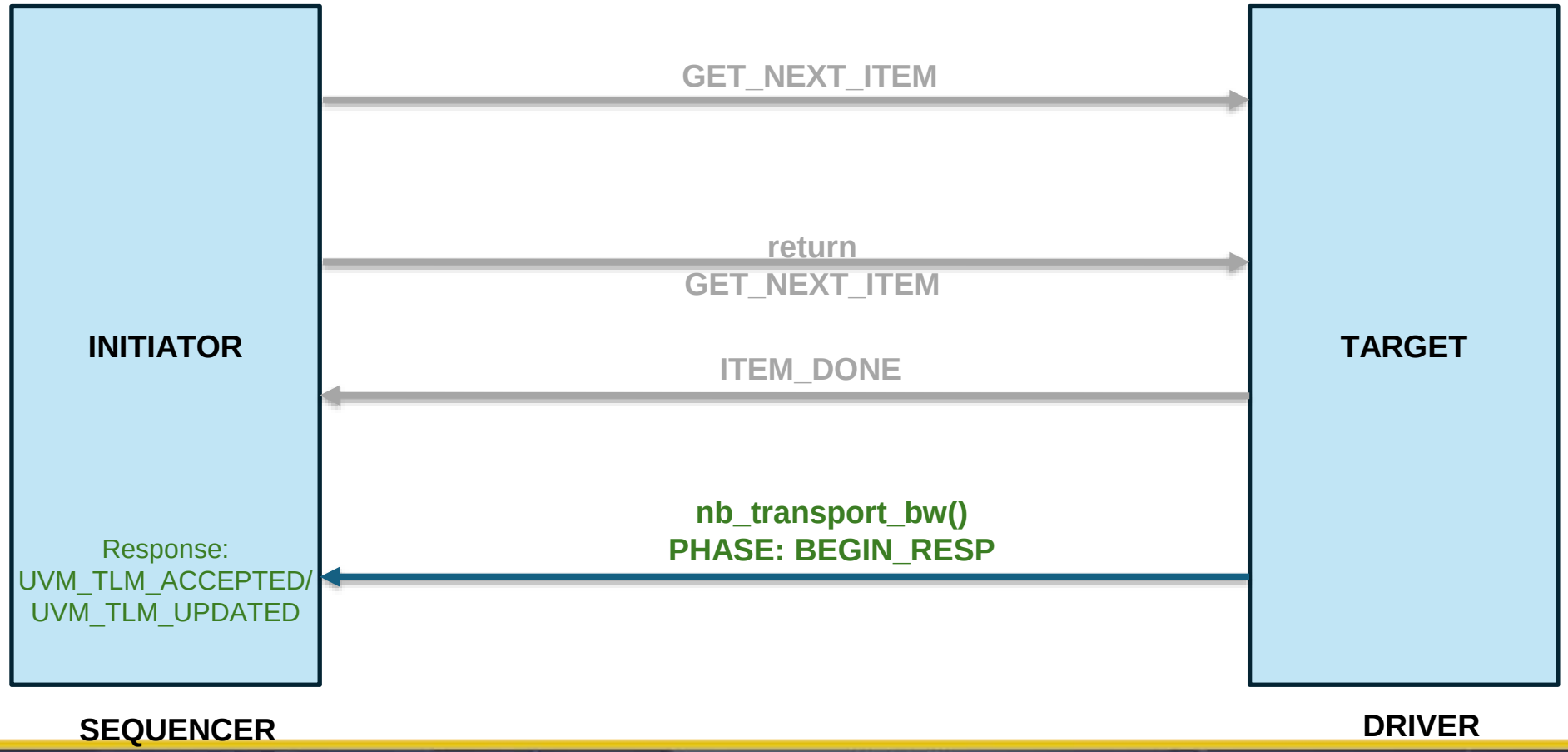
TLM2 protocol for sequencer-driver

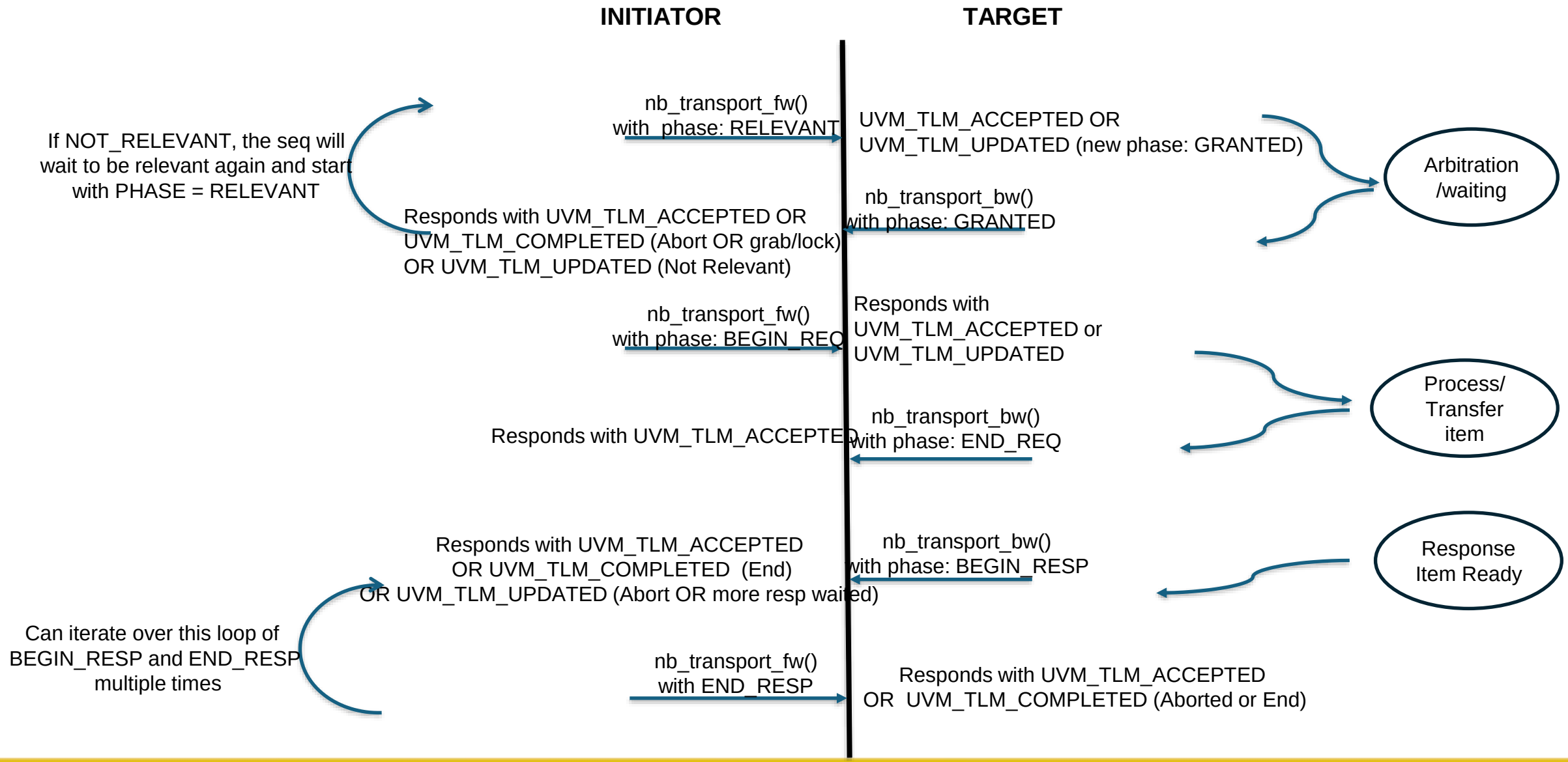


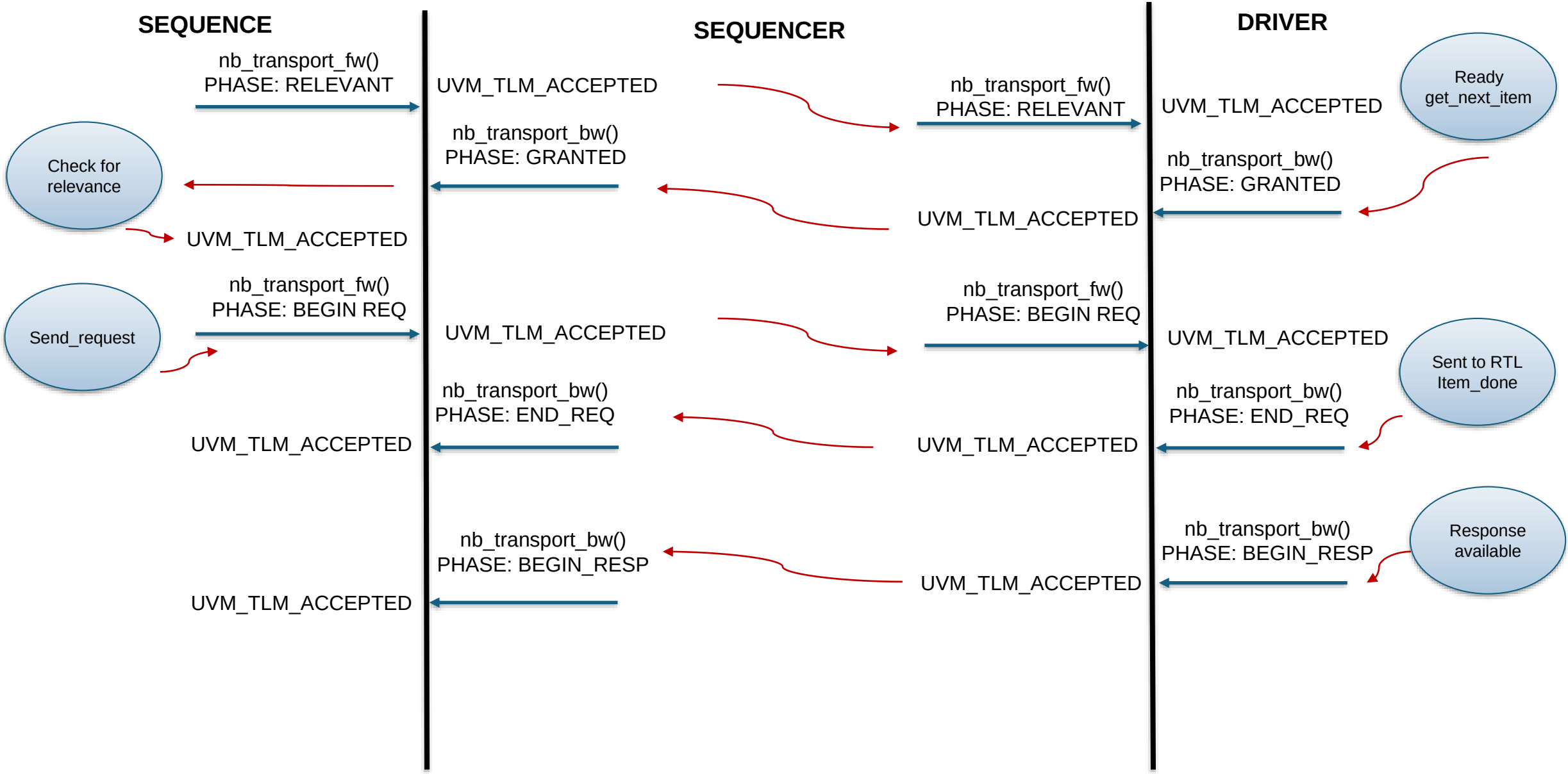
TLM2 protocol for sequencer-driver



TLM2 protocol for sequencer-driver




INITIATOR
TARGET
If NOT_RELEVANT, the seq will wait to be relevant again and start with PHASE = RELEVANT
nb_transport_fw() with phase: RELEVANT
UVM_TLM_ACCEPTED OR UVM_TLM_UPDATED (new phase: GRANTED)
Arbitration /waiting
nb_transport_bw() with phase: GRANTED
Responds with UVM_TLM_ACCEPTED OR UVM_TLM_COMPLETED (Abort OR grab/lock) OR UVM_TLM_UPDATED (Not Relevant)
nb_transport_fw() with phase: BEGIN_REQ
Responds with UVM_TLM_ACCEPTED or UVM_TLM_UPDATED
Process/ Transfer item
nb_transport_bw() with phase: END_REQ
Responds with UVM_TLM_ACCEPTED
Response Item Ready
nb_transport_bw() with phase: BEGIN_RESP
Responds with UVM_TLM_ACCEPTED OR UVM_TLM_COMPLETED (End) OR UVM_TLM_UPDATED (Abort OR more resp waited)
Can iterate over this loop of BEGIN_RESP and END_RESP multiple times
nb_transport_fw() with END_RESP
Responds with UVM_TLM_ACCEPTED OR UVM_TLM_COMPLETED (Aborted or End)


SEQUENCE
SEQUENCER
DRIVER
nb_transport_fw()
PHASE: RELEVANT
UVM_TLM_ACCEPTED
nb_transport_fw()
PHASE: RELEVANT
UVM_TLM_ACCEPTED
Ready
get_next_item
nb_transport_bw()
PHASE: GRANTED
Check for
relevance
nb_transport_bw()
PHASE: GRANTED
UVM_TLM_ACCEPTED
UVM_TLM_ACCEPTED
nb_transport_fw()
PHASE: BEGIN REQ
Send_request
nb_transport_fw()
PHASE: BEGIN REQ
UVM_TLM_ACCEPTED
UVM_TLM_ACCEPTED
Sent to RTL
Item_done
nb_transport_bw()
PHASE: END_REQ
nb_transport_bw()
PHASE: END_REQ
UVM_TLM_ACCEPTED
UVM_TLM_ACCEPTED
nb_transport_bw()
PHASE: BEGIN_RESP
Response
available
nb_transport_bw()
PHASE: BEGIN_RESP
UVM_TLM_ACCEPTED
UVM_TLM_ACCEPTED

What did we observe?

- Current communication is TLM1-like at best
- TLM2 implementation will use non-blocking methods, more TLM2 phases and a new payload class
- Both the seq-seqr and seqr-drvr communication can be mapped to TLM2

Conclusion

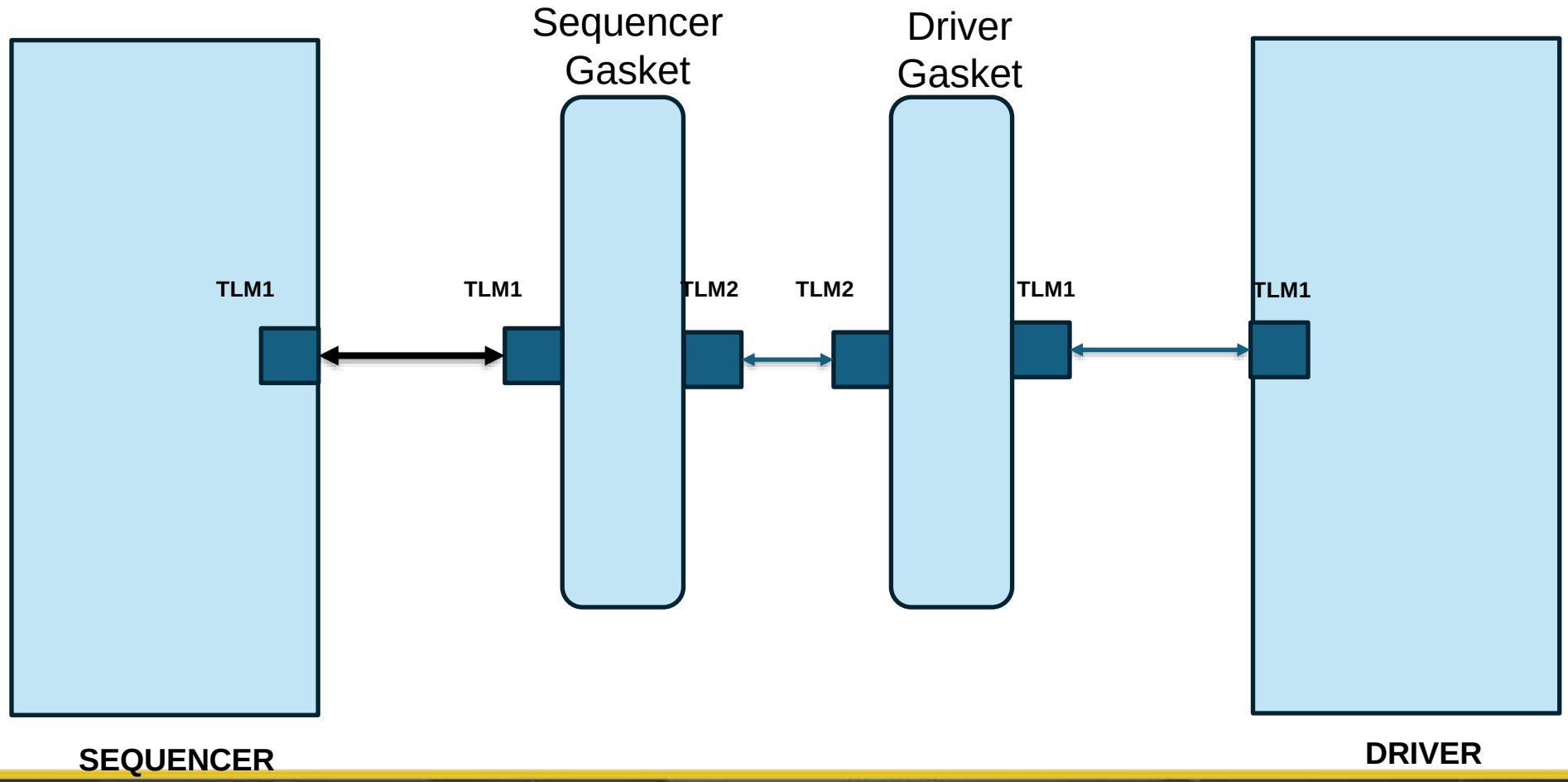
- It is possible to move UVM sequence-sequencer and sequencer-driver communication to TLM2
  - Yes, it can be broken down to phases, have a payload class and communicate via non-blocking calls.
  - It will bring interoperability between language boundaries, like system C and system Verilog.
  - It will overcome the limitations of the current implementation

Q & A

Other Topics

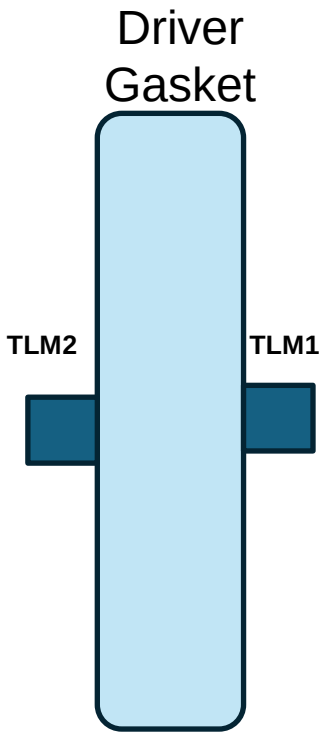
- Driver and Sequencer Gaskets
- Another approach to Seq-Seqr TLM2 adoption

Sequencer and Driver Gaskets



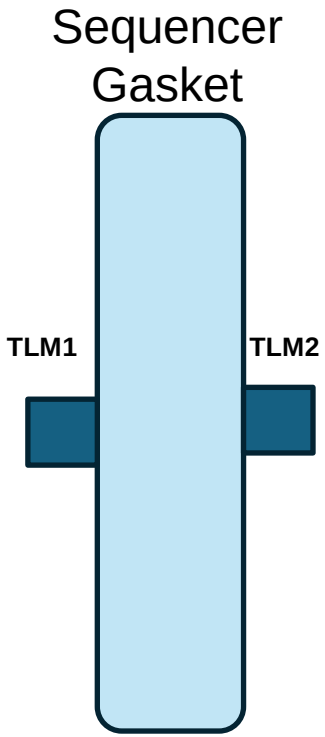
Driver gasket

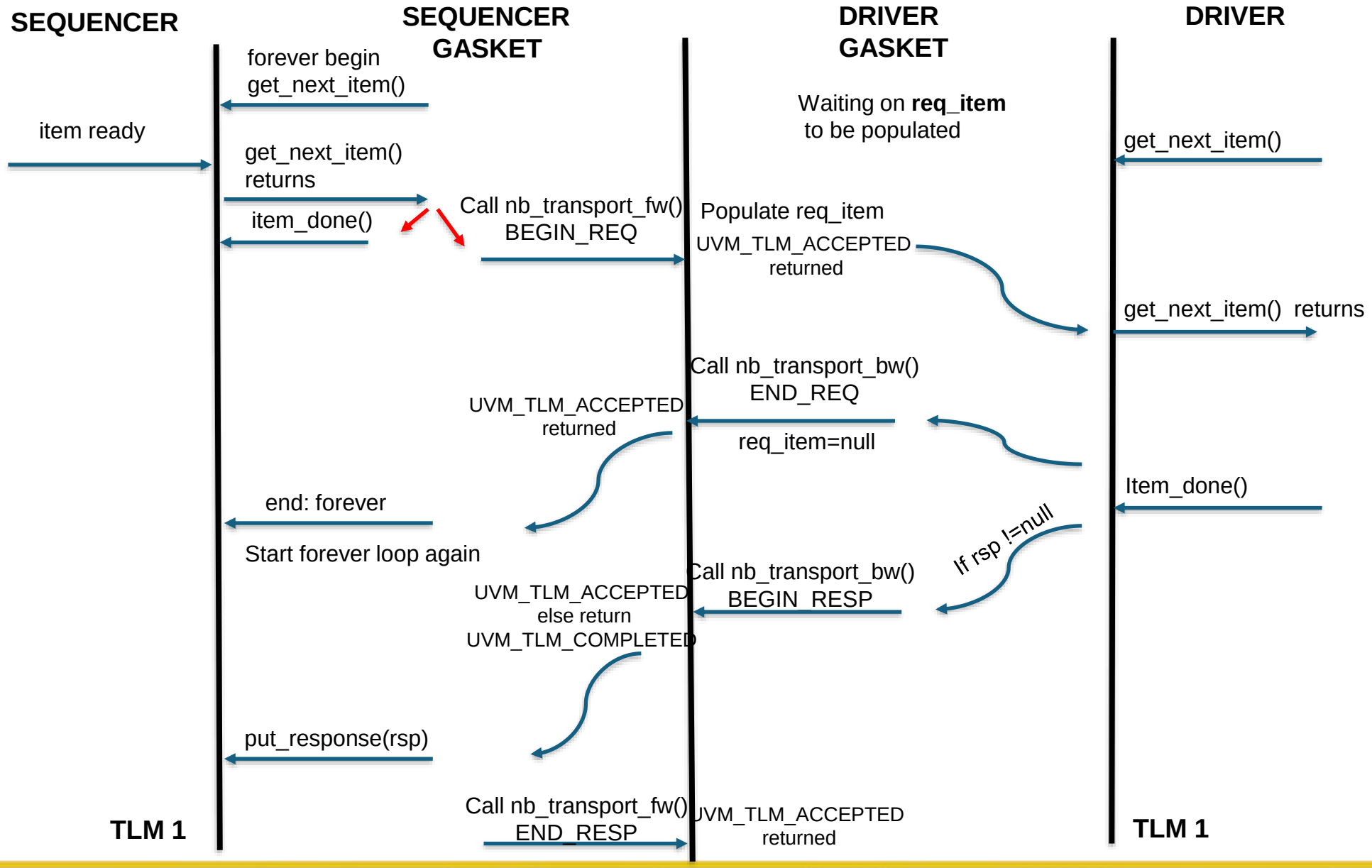
- TLM1 sequencer for the driver
  - `seq_item_pull_imp` with `sqr_if` functions
- TLM2 driver for the sequencer gasket
  - Target socket with `nb_transport_fw()` implementation



Sequencer gasket

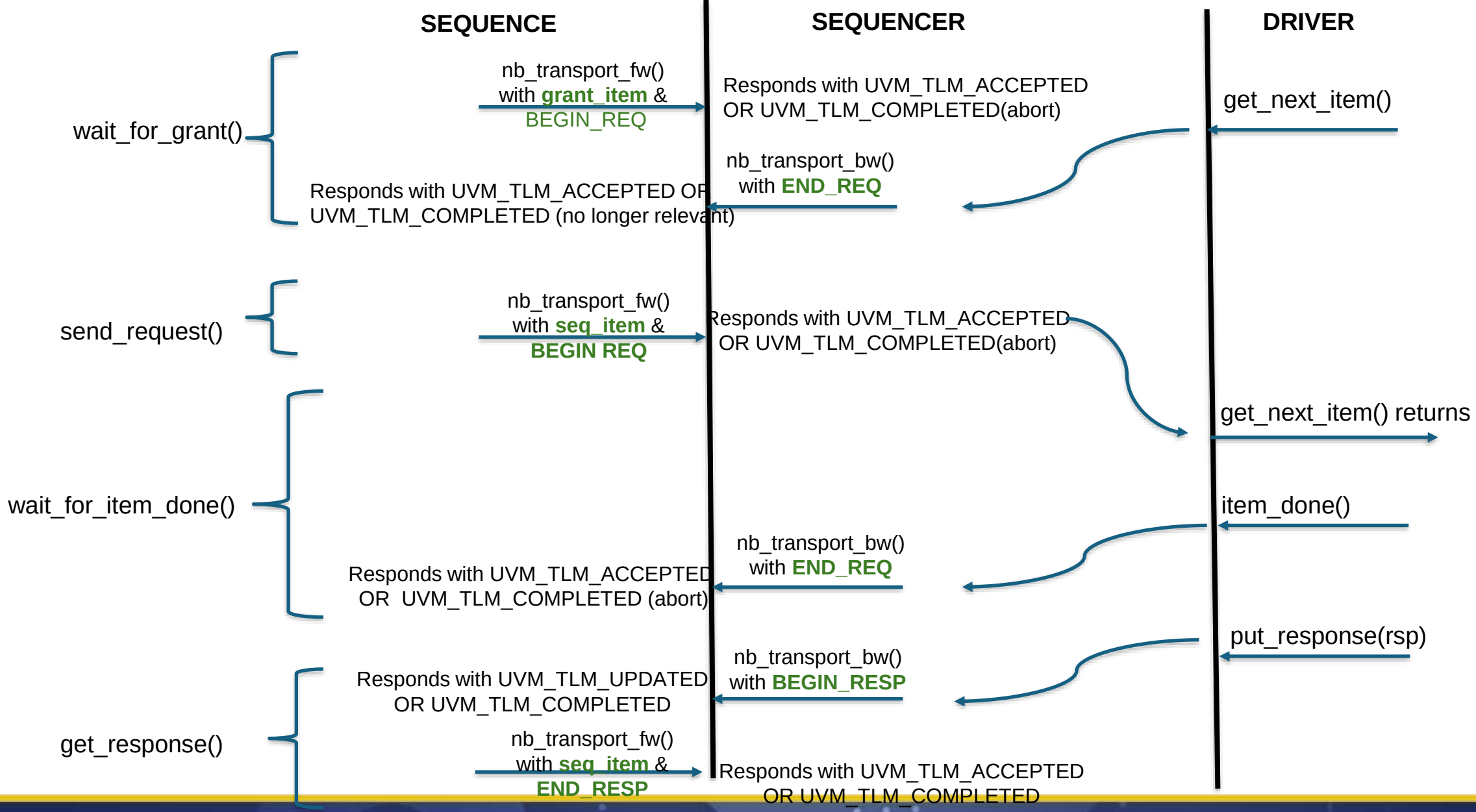
- TLM2 sequencer for the driver gasket
  - `seq_item_port`
- TLM1 driver for the TLM1 sequencer
  - Initiator socket with `nb_transport_bw()` implementation




SEQUENCER
SEQUENCER GASKET
DRIVER GASKET
DRIVER
forever begin
get_next_item()
item ready
Waiting on req_item to be populated
get_next_item()
get_next_item() returns
Call nb_transport_fw() BEGIN_REQ
item_done()
Populate req_item
UVM_TLM_ACCEPTED returned
get_next_item() returns
Call nb_transport_bw() END_REQ
UVM_TLM_ACCEPTED returned
req_item=null
Item_done()
end: forever
If rsp !=null
Start forever loop again
Call nb_transport_bw() BEGIN_RESP
UVM_TLM_ACCEPTED else return UVM_TLM_COMPLETED
put_response(rsp)
Call nb_transport_fw() END_RESP
UVM_TLM_ACCEPTED returned
TLM 1
TLM 1
TLM 2

Another implementation for Seq-Seqr

- Keep the same TLM2 phases as of today
- Use different payload classes


SEQUENCE
SEQUENCER
DRIVER
wait_for_grant()
nb_transport_fw()
with grant_item &
BEGIN_REQ
Responds with UVM_TLM_ACCEPTED
OR UVM_TLM_COMPLETED(abort)
get_next_item()
nb_transport_bw()
with END_REQ
Responds with UVM_TLM_ACCEPTED OR
UVM_TLM_COMPLETED (no longer relevant)
send_request()
nb_transport_fw()
with seq_item &
BEGIN REQ
Responds with UVM_TLM_ACCEPTED
OR UVM_TLM_COMPLETED(abort)
get_next_item() returns
wait_for_item_done()
item_done()
nb_transport_bw()
with END_REQ
Responds with UVM_TLM_ACCEPTED
OR UVM_TLM_COMPLETED (abort)
put_response(rsp)
nb_transport_bw()
with BEGIN_RESP
Responds with UVM_TLM_UPDATED
OR UVM_TLM_COMPLETED
get_response()
nb_transport_fw()
with seq_item &
END_RESP
Responds with UVM_TLM_ACCEPTED
OR UVM_TLM_COMPLETED