Lies My Teacher Told Me About The UVM

Basic Stimulus

Justin Refice
NVIDIA Corporation

Sep 24, 2015
SNUG Boston

Agenda

Part 1: Flow of stimulus in the UVM

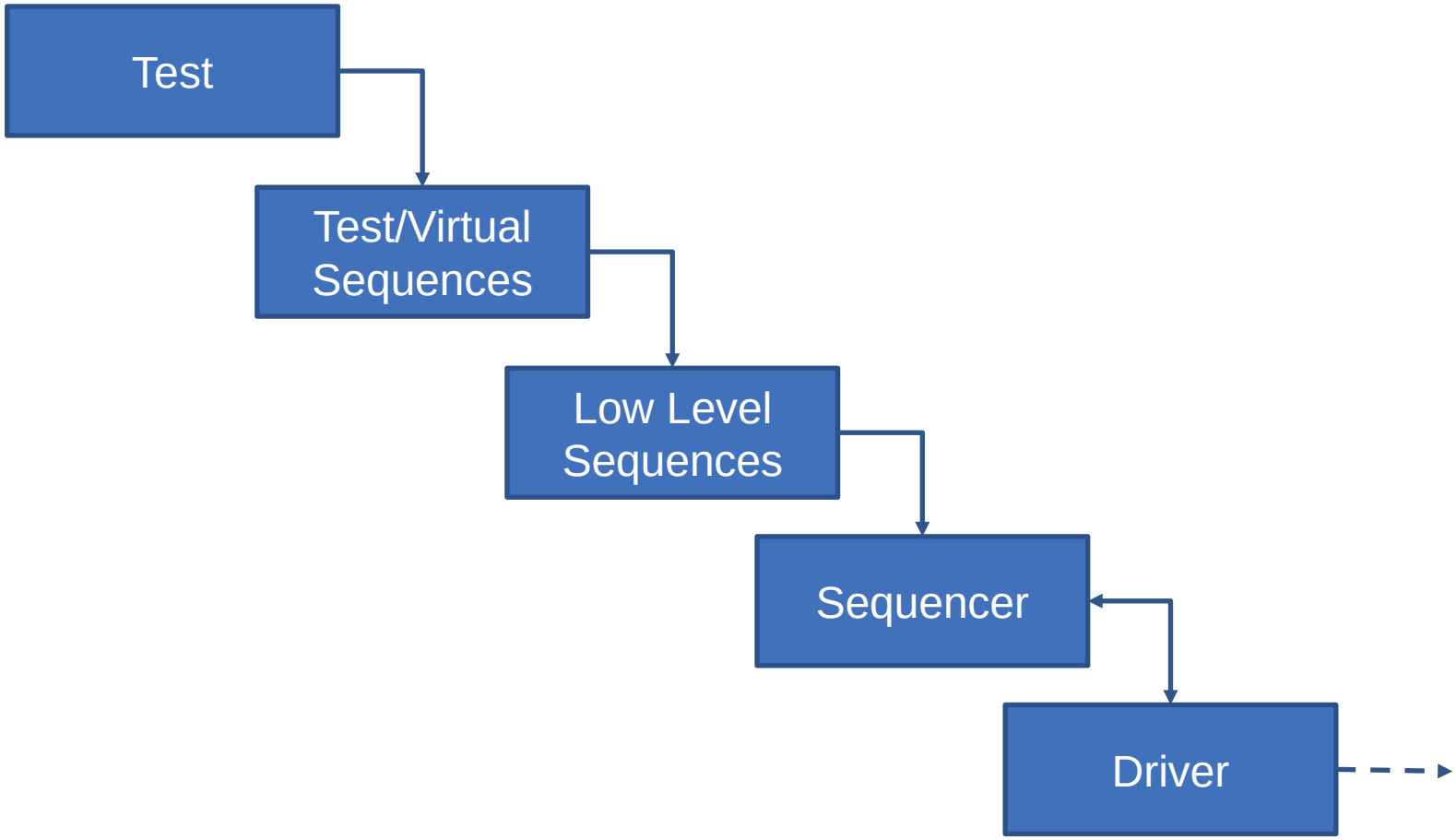
Part 2: Communicating in the real world

Part 3: Why is this important?

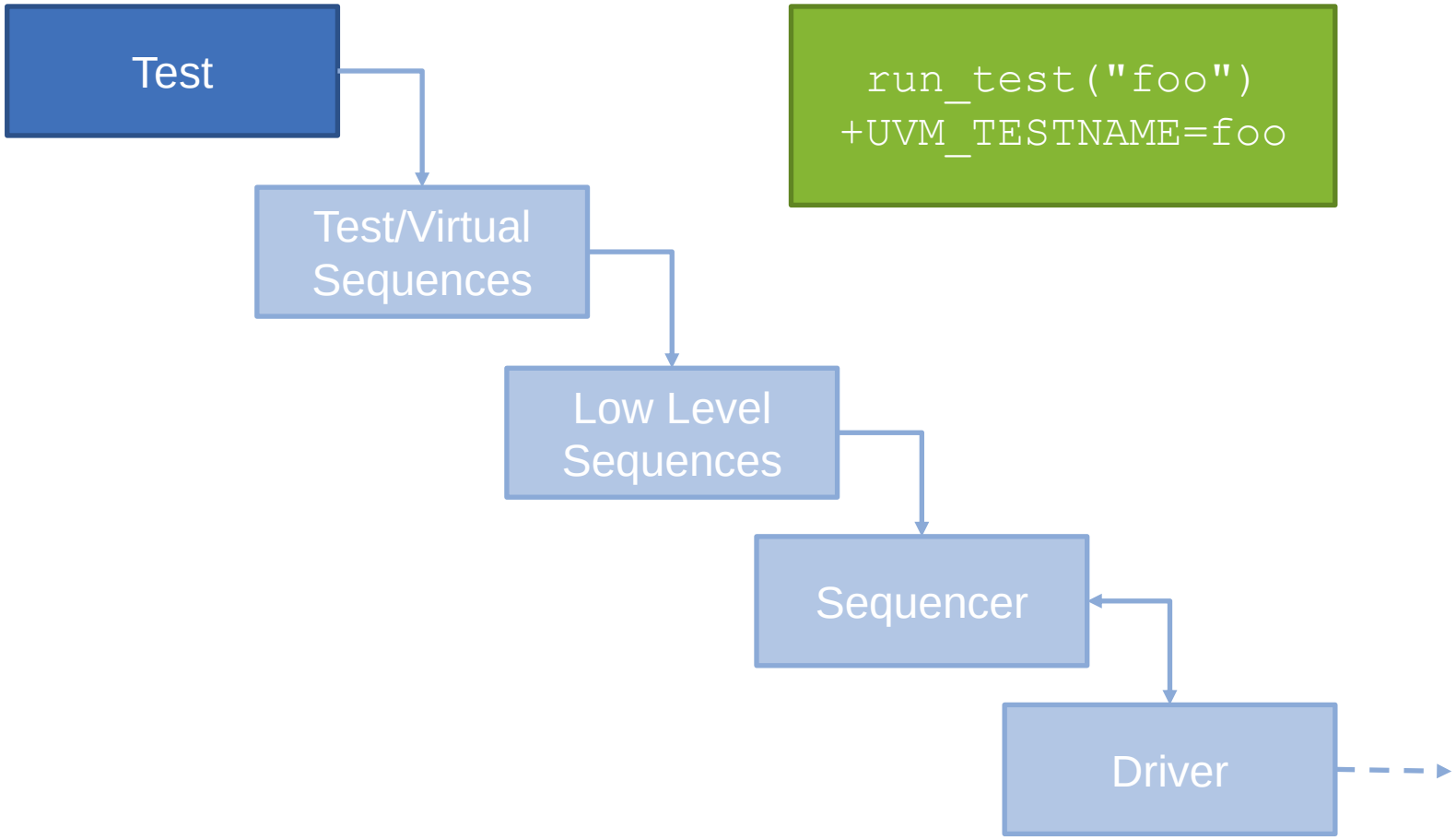
The flow of stimulus

Ready? Set? Go!

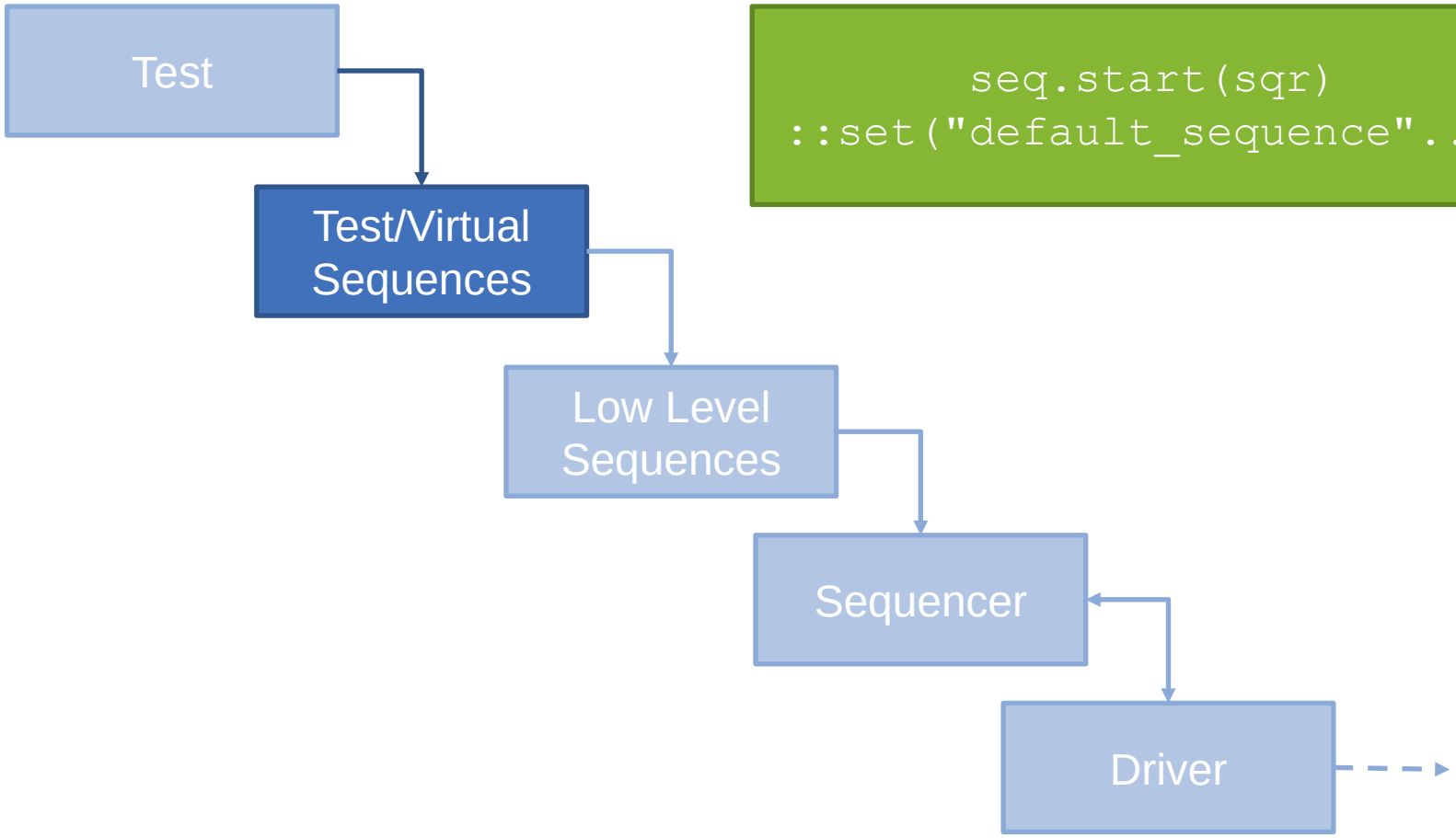
UVM Stimulus



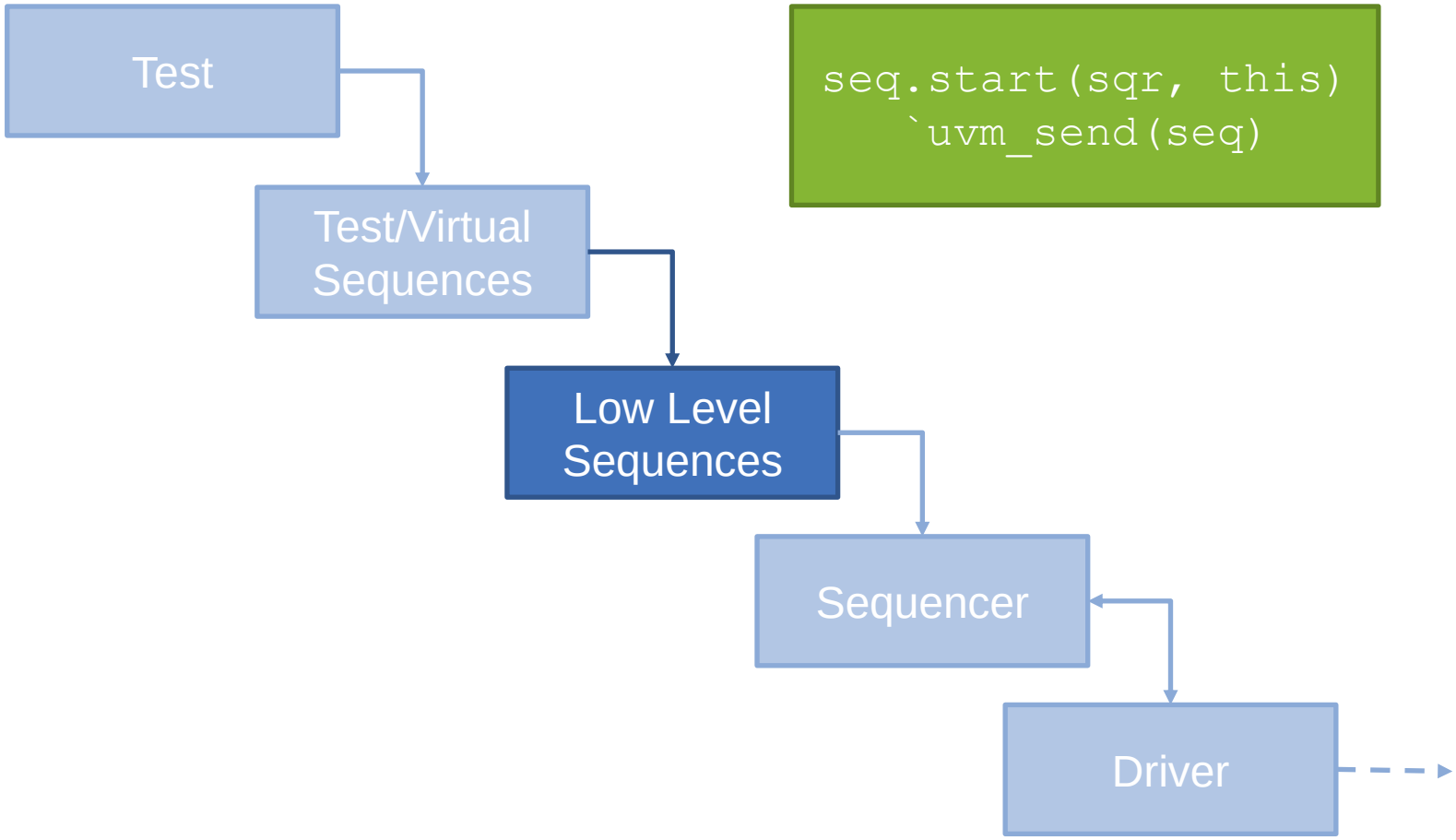
UVM Stimulus



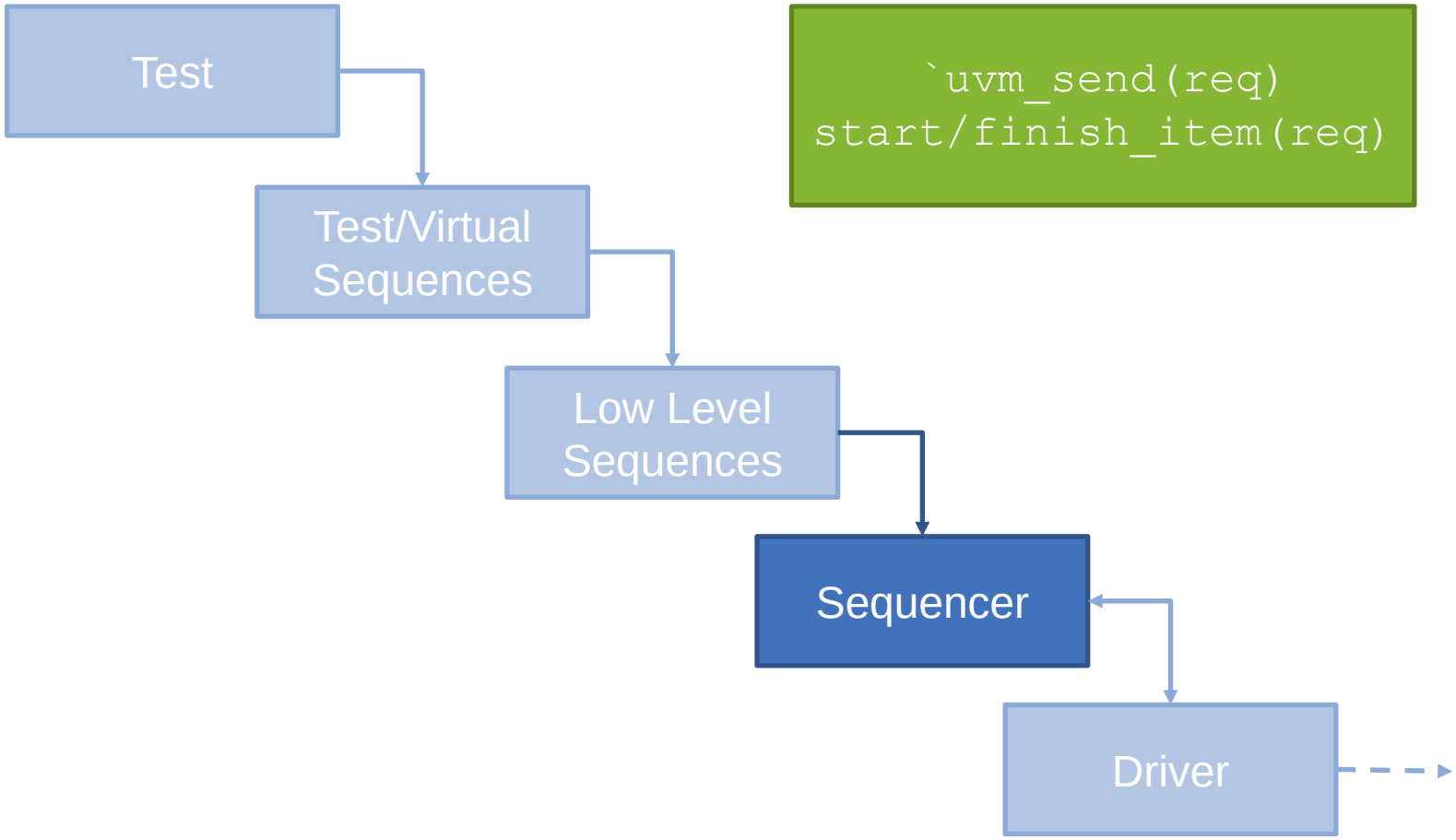
UVM Stimulus



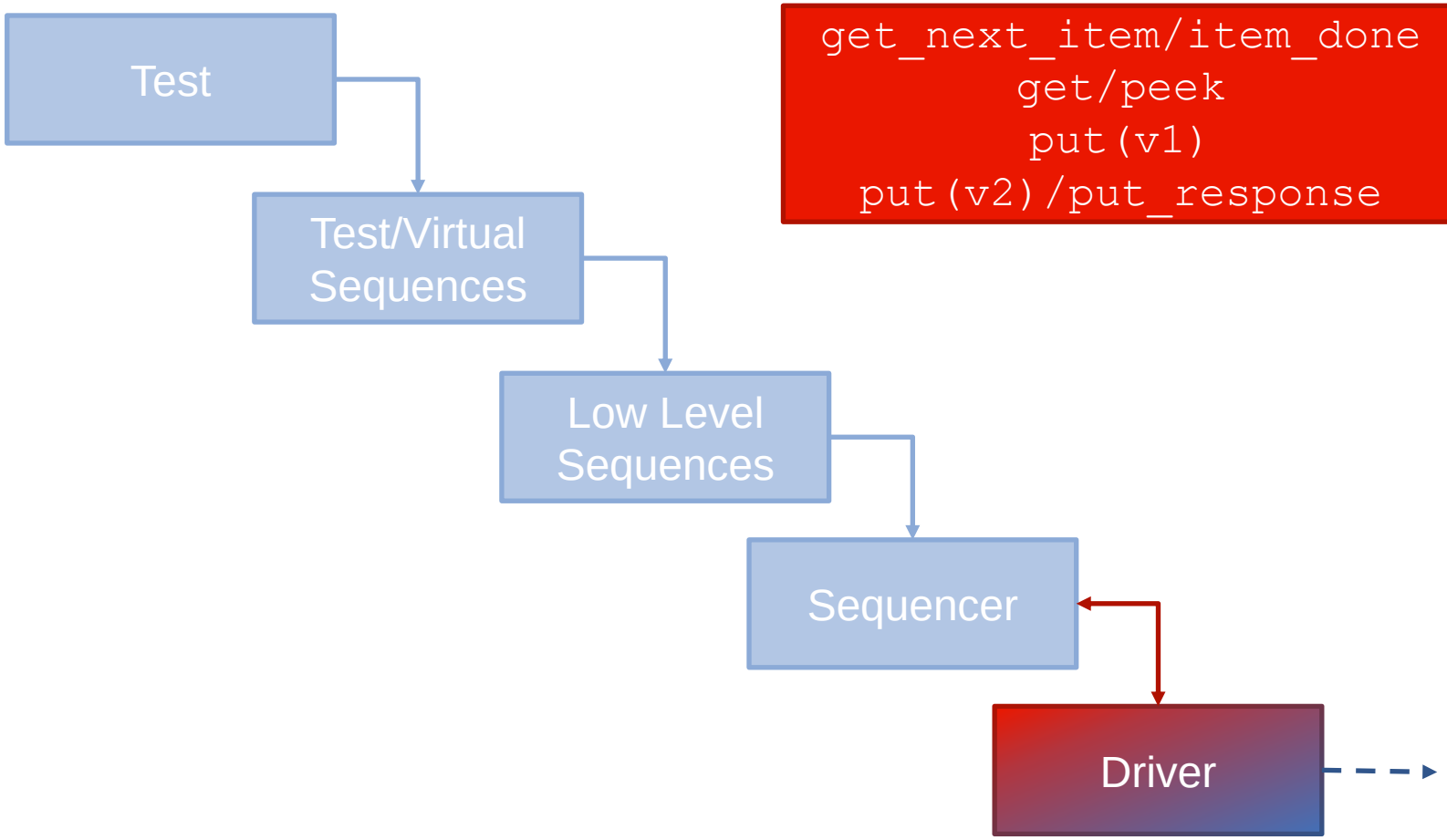
UVM Stimulus



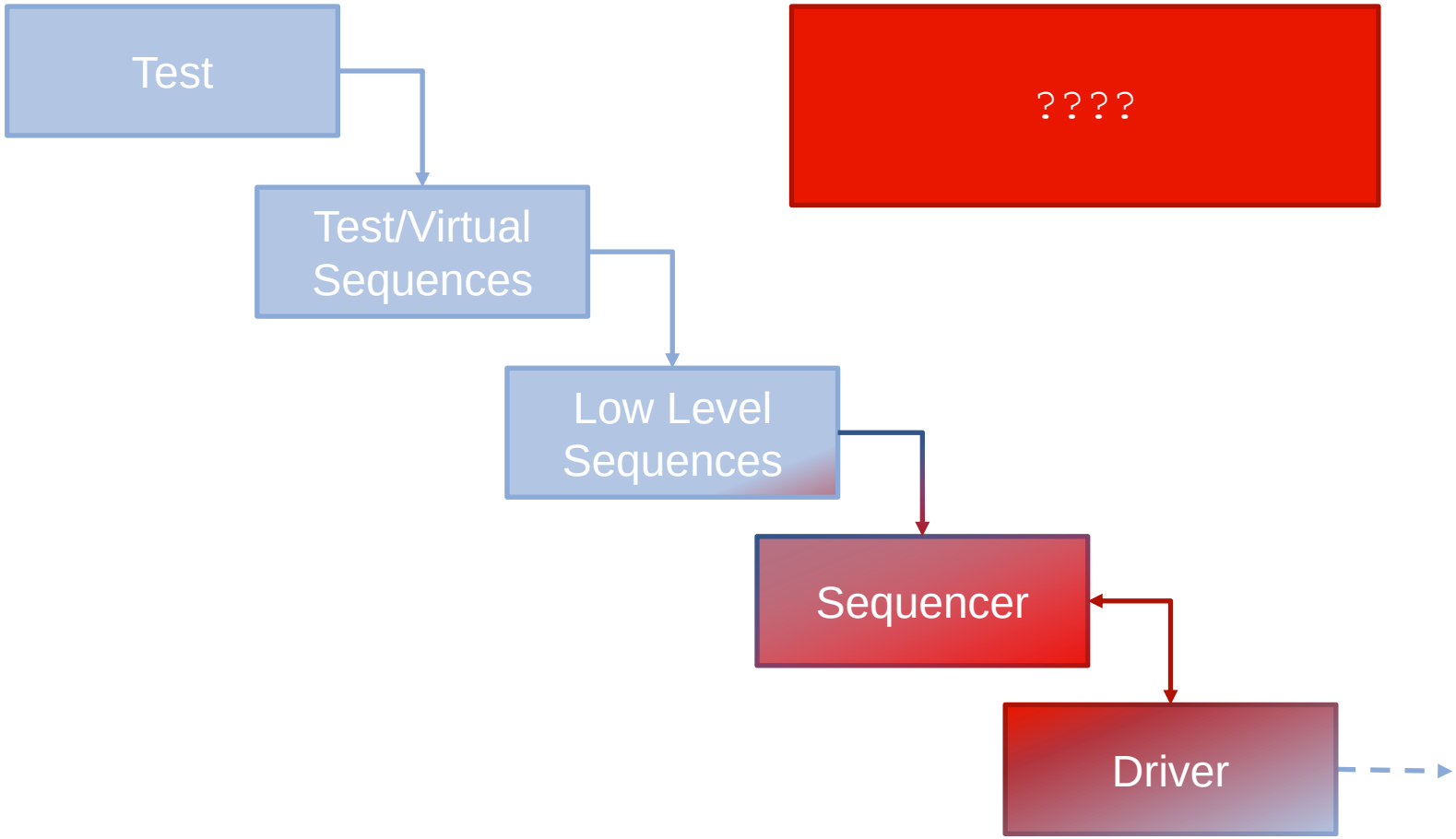
UVM Stimulus



UVM Stimulus



UVM Stimulus



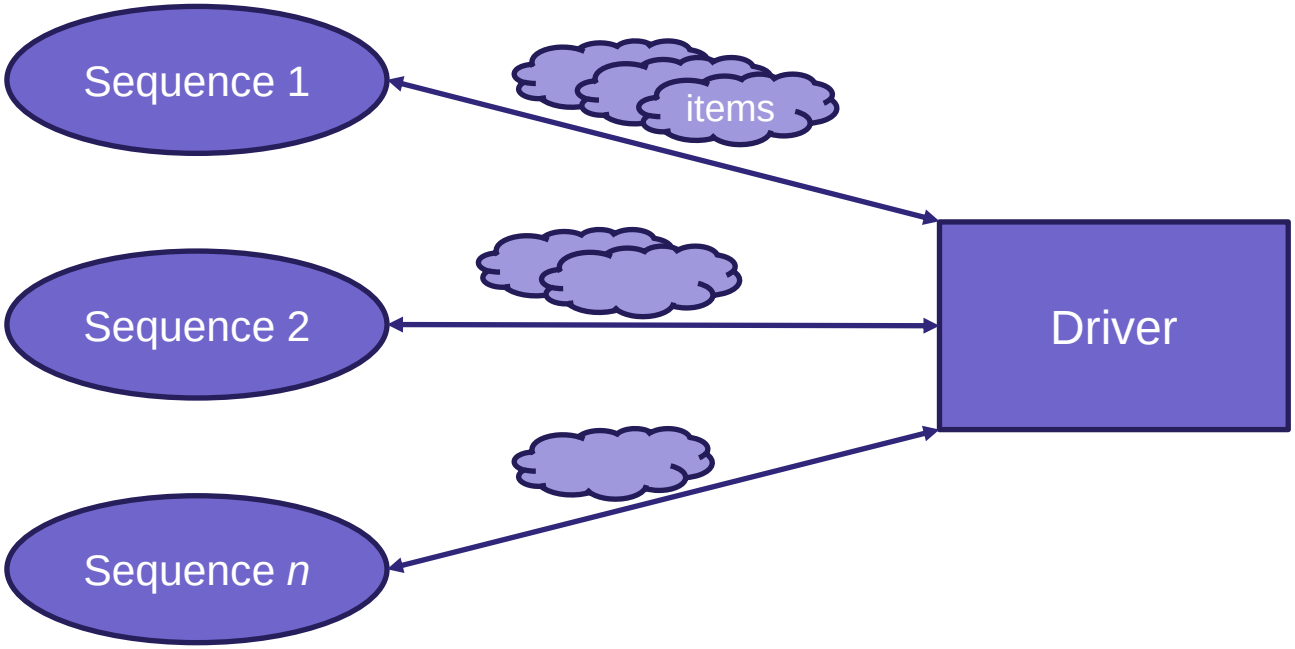
Communication in the real world

"Hello", "Goodbye" and everything in between...

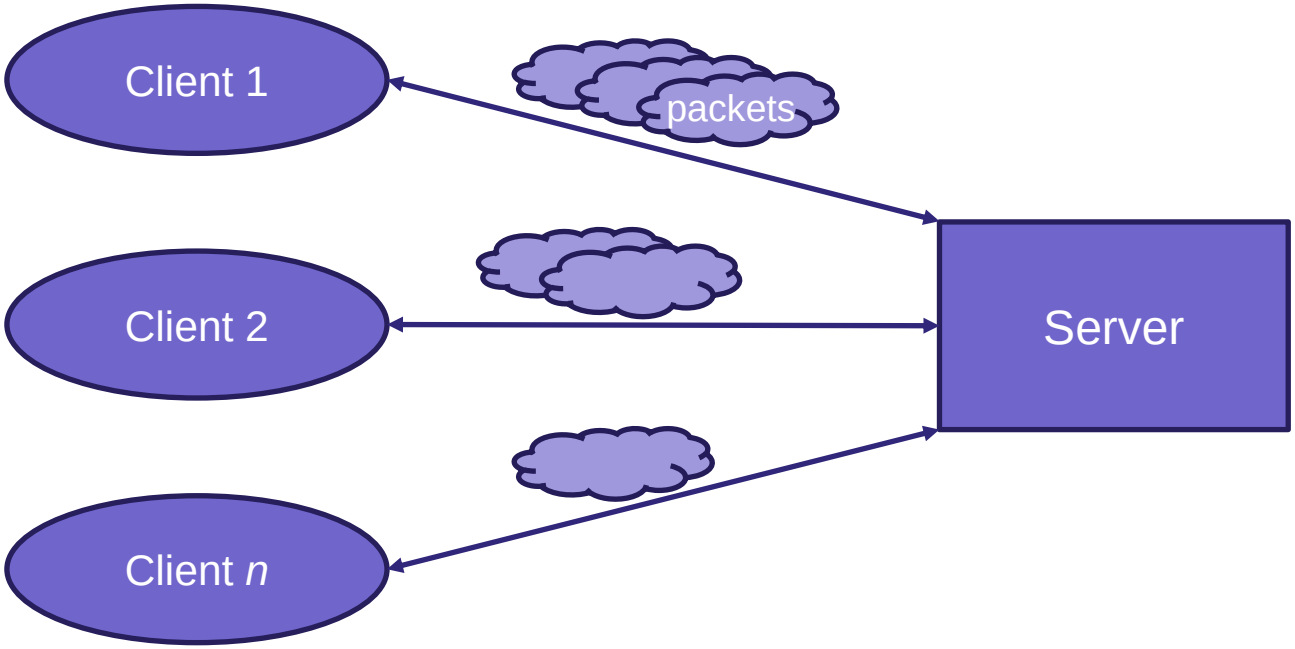
UVM Stimulus Terms

- Sequence Items
  - Abstract representation of data
- Drivers
  - Consume items and drive the bus
- Sequence
  - Produces a stream of items for the driver to consume

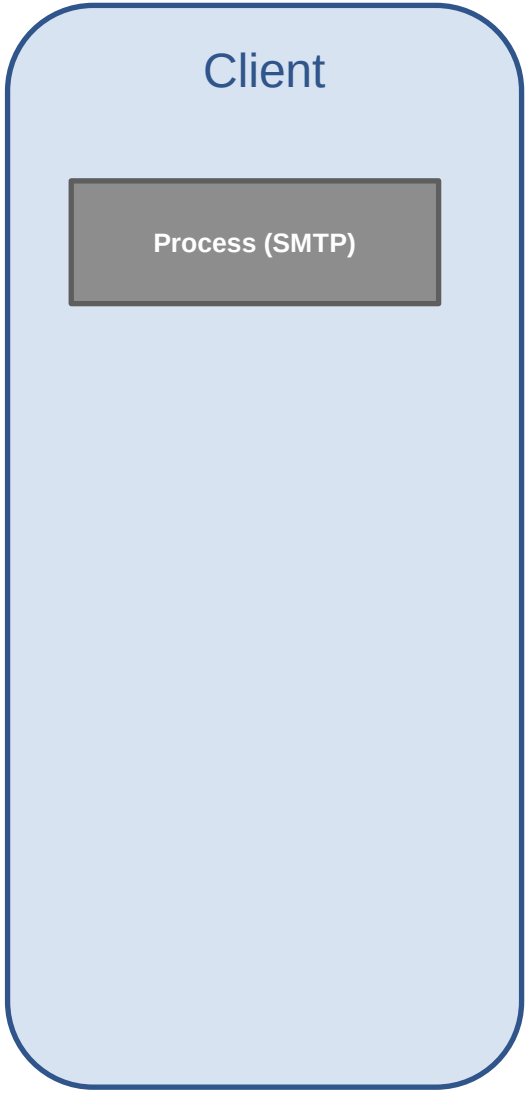
UVM Stimulus Diagram



Looking Familiar?

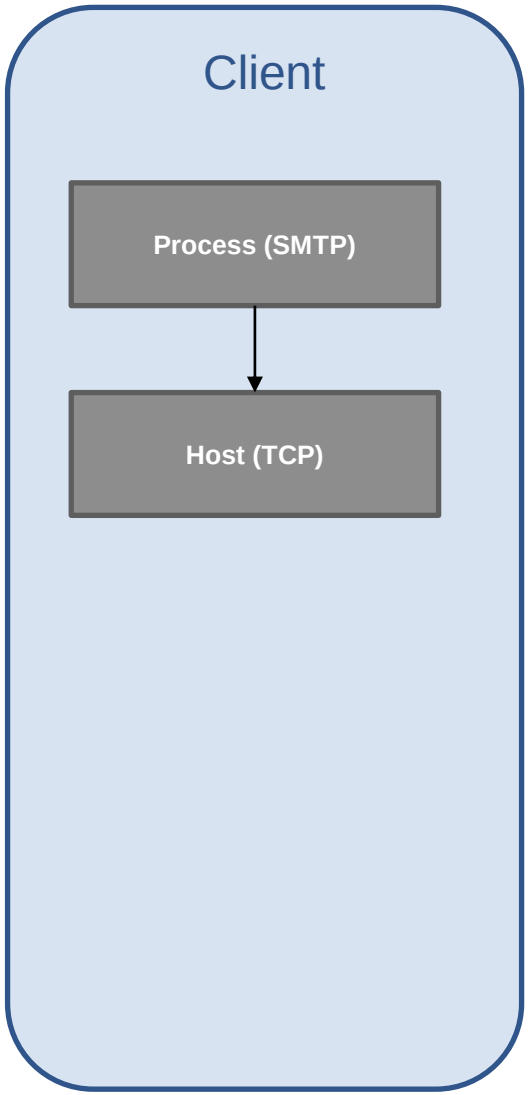


Client Email (1)




```
From: foo@bar.com
To: hello@world.com
Subject: UVM is the best!
```

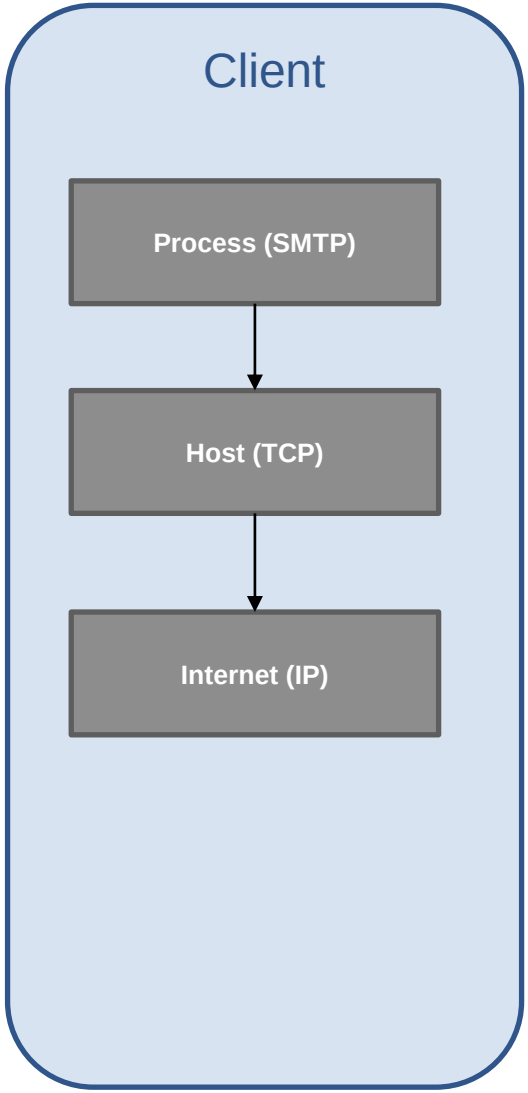
Client Email (2)




```
From: foo@bar.com
To: hello@world.com
Subject: UVM is the best!
```

```
Source Port: 64234
Destination Port: 25
```

Client Email (3)

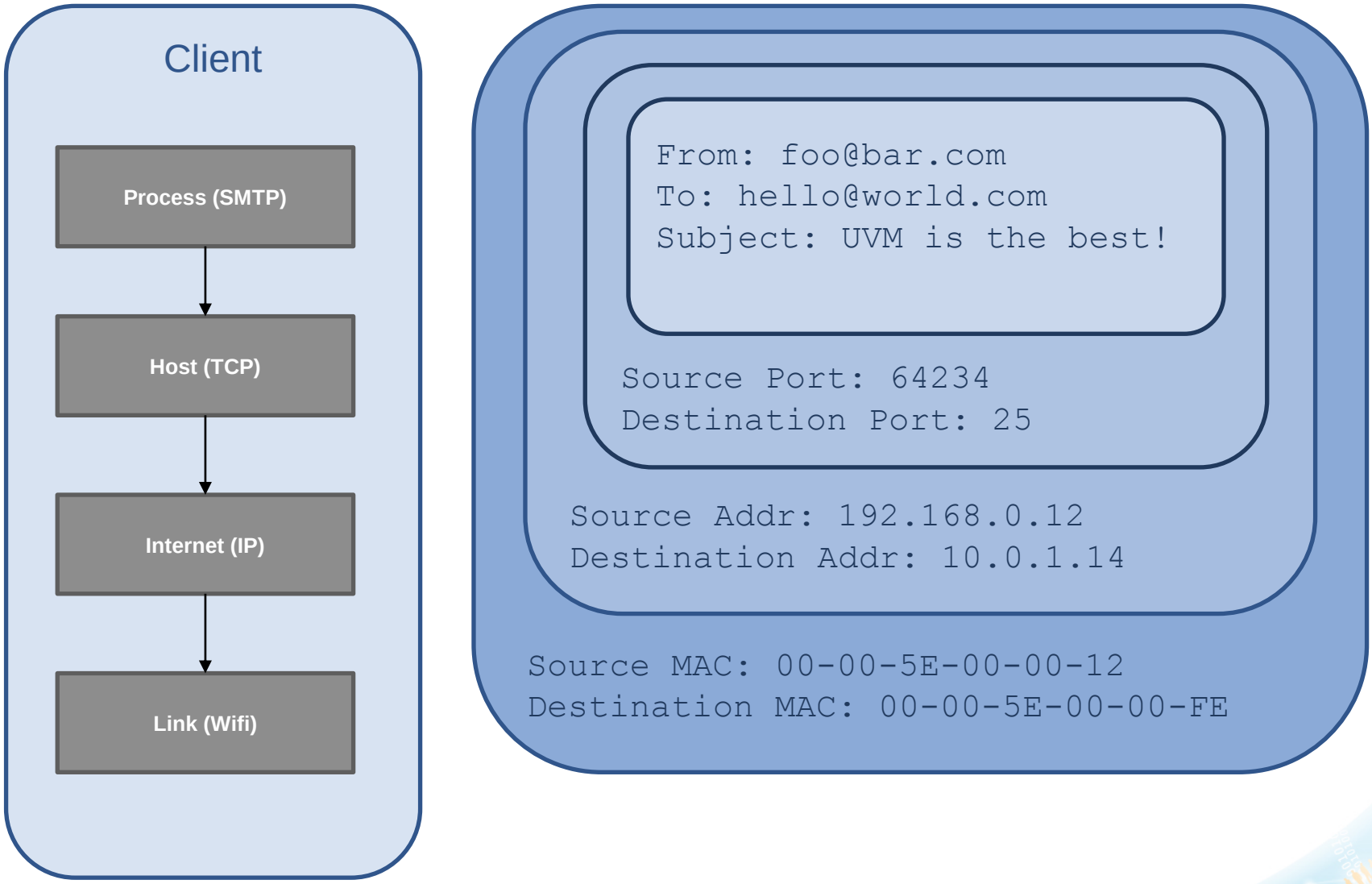



```
From: foo@bar.com
To: hello@world.com
Subject: UVM is the best!
```

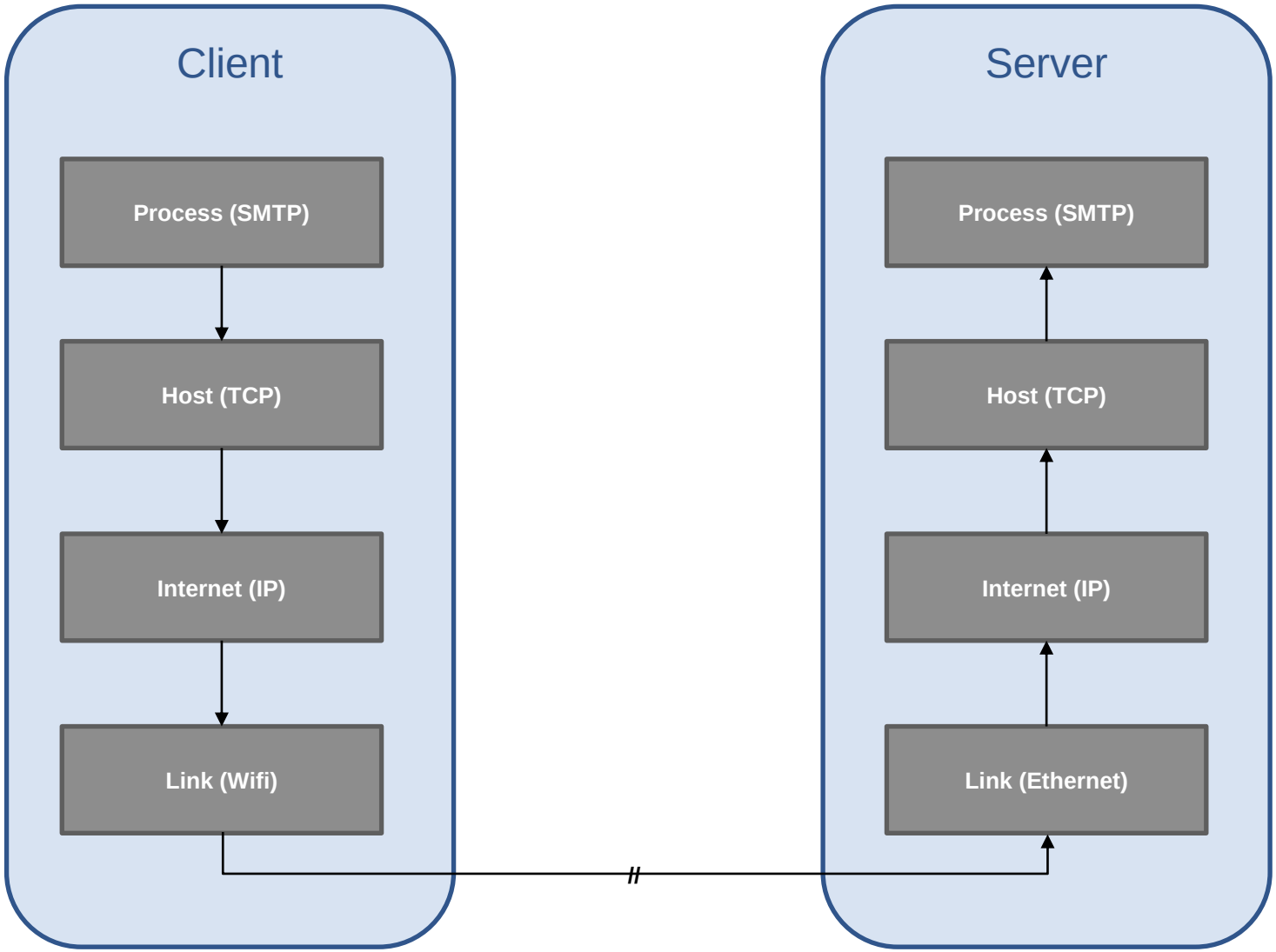
```
Source Port: 64234
Destination Port: 25
```

```
Source Addr: 192.168.0.12
Destination Addr: 10.0.1.14
```

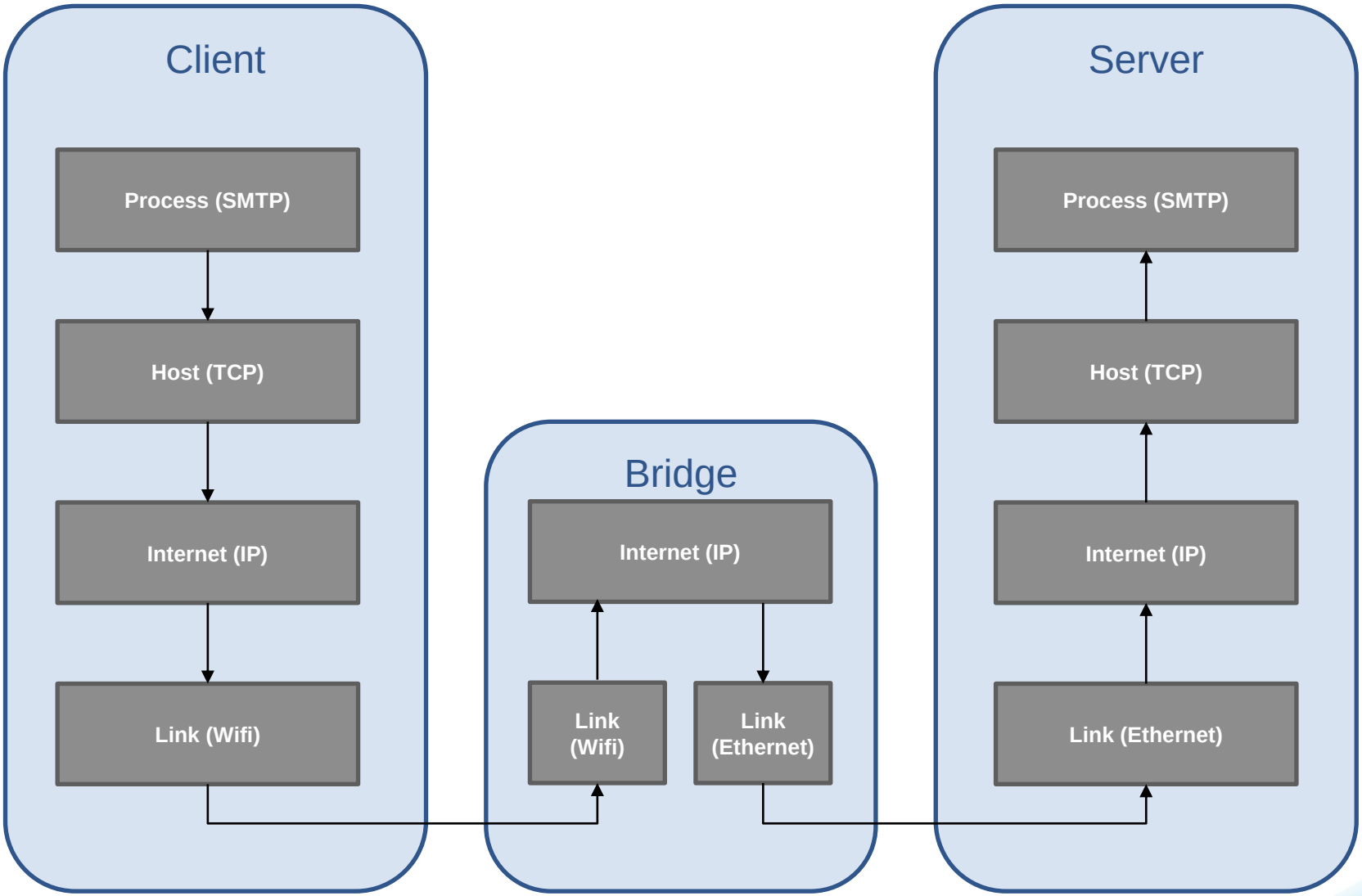
Client Email (4)



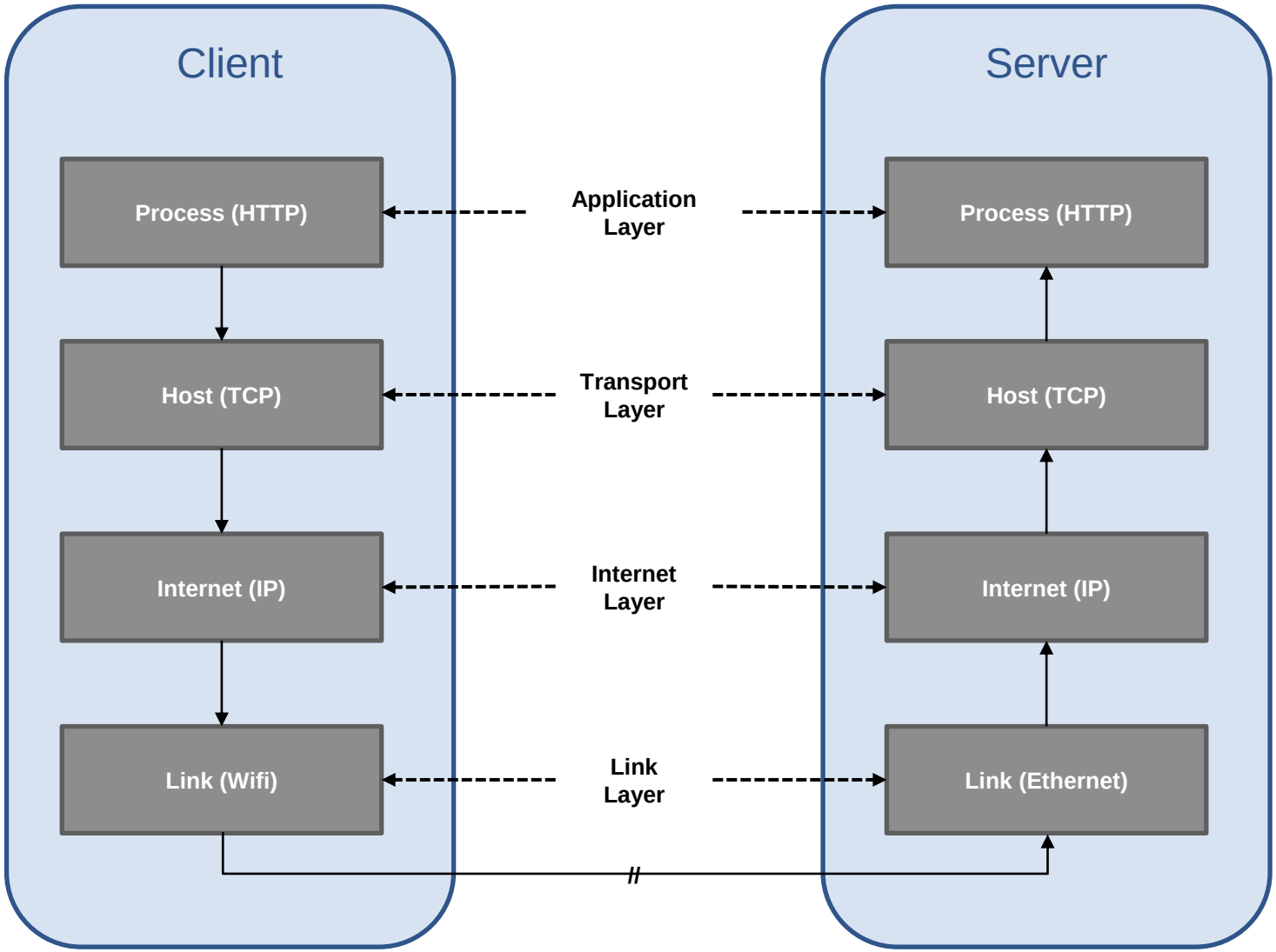
Client/Server Email



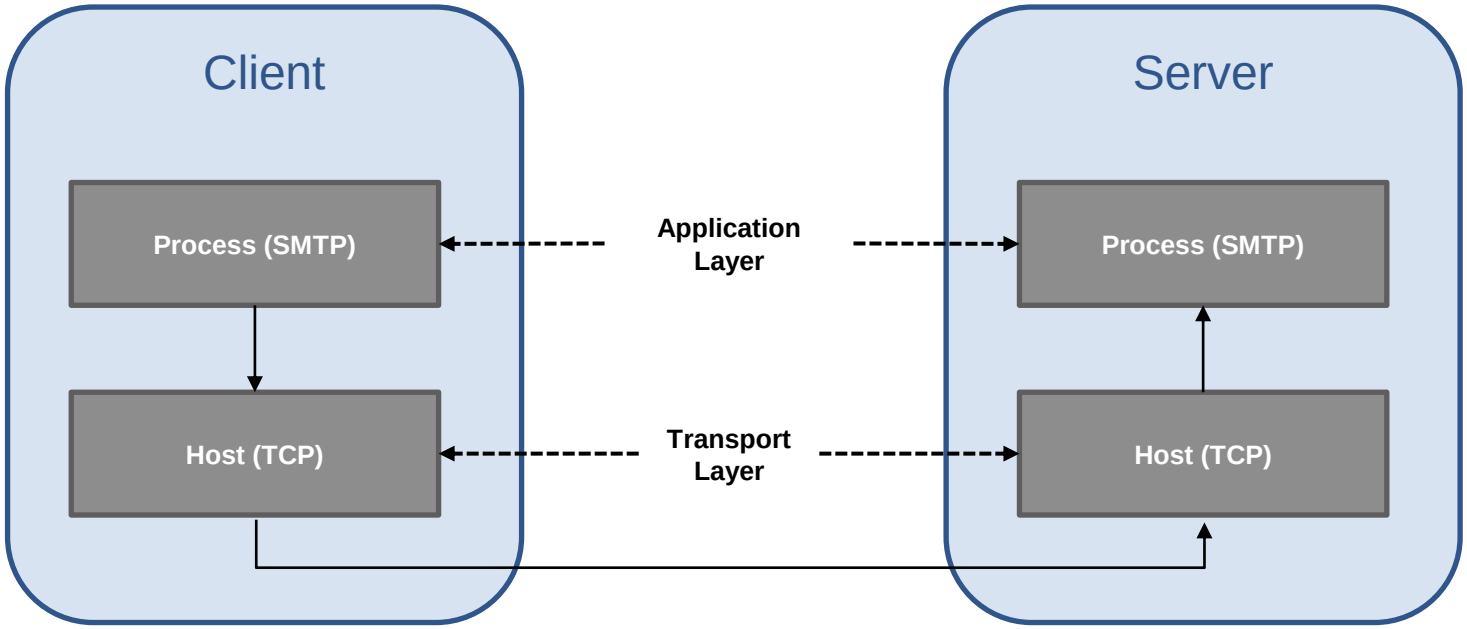
Encapsulation



TCP/IP Layers



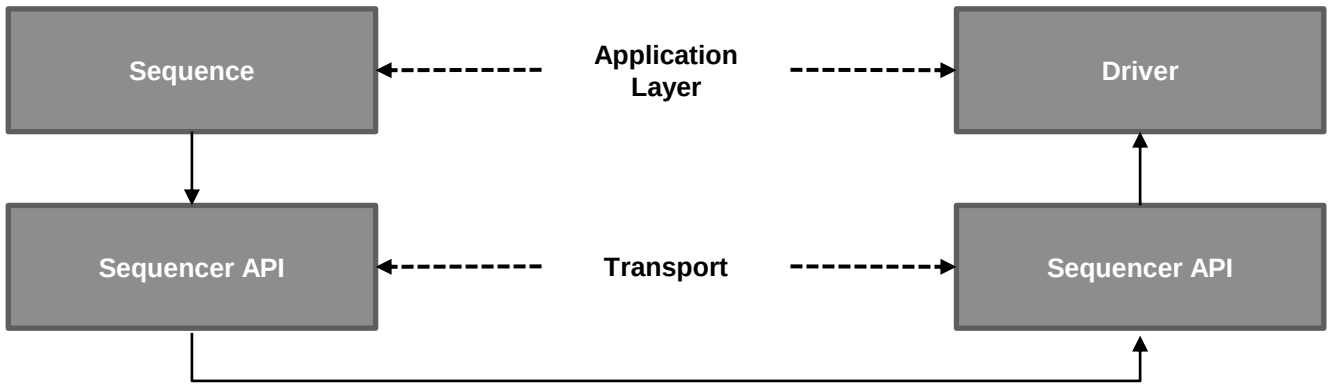
Sending email locally



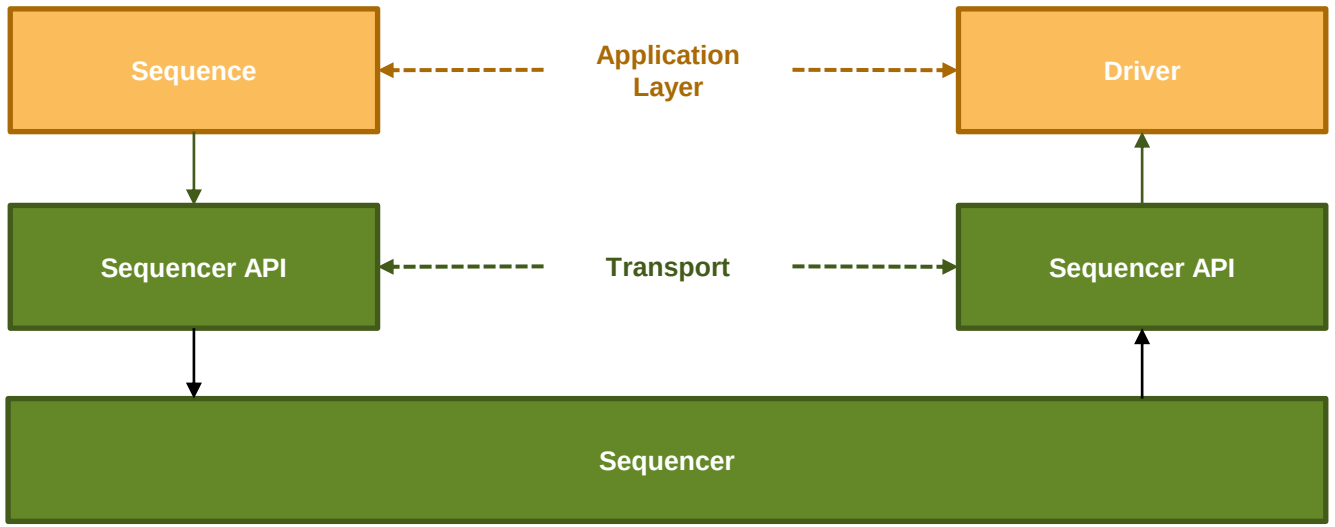
High Level Communication

UVM "Transport" Layer

UVM "Transport" Layer



UVM Data Flow



Standardized

Not Standardized

Why is this important?

... and what could go wrong if I ignore it?

To pipeline or not to pipeline...

- APB
  - Non-pipelined
  - Interface can only handle a single transaction at a time
- AHB-Lite
  - Pipelined
  - Multiple transactions can be handled simultaneously
- Both are memory mapped protocols
  - Read/Write addresses in memory

The APB Example

- Request is handled via:
  - `uvm_do(req)
  - `uvm_send(req)
  - start_item(req)/finish_item(req)

- Response information is contained within 'req'

The AHB-Lite Example

- Request is handled via:
  - `uvm_do(req)
  - `uvm_send(req)
  - start_item(req)/finish_item(req)

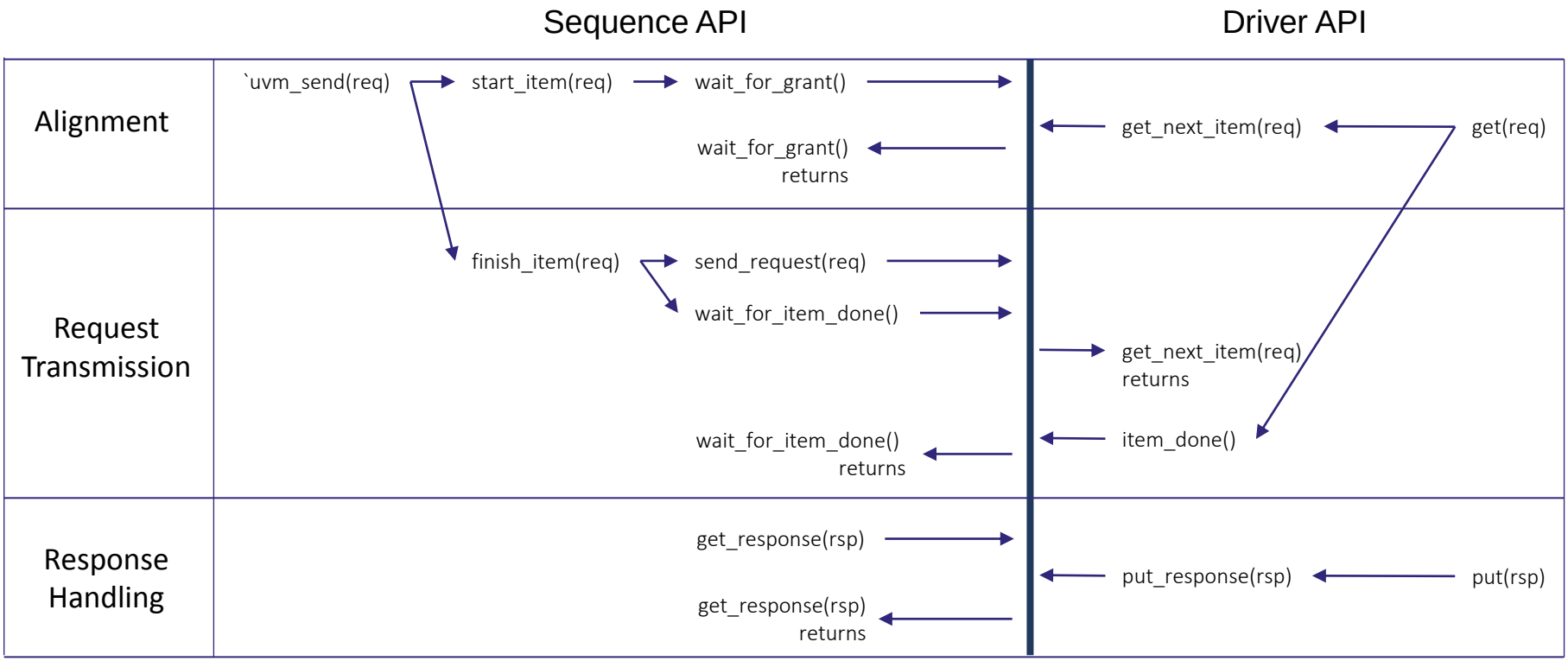
- Response information is retrieved via:
  - get_response(rsp)
  - response_handler(rsp)

Lifetime of a Session

- A "Session" is a communication from the sequence to the driver.
  - High level concept, *not* a "real" UVM term.

- Three part handshake:
  - Alignment (Is everyone ready?)
  - Request Transmission (Sequence → Driver)
  - [Optional] Responses (Driver → Sequence)

Sequencer API



Evil, thy name is item_done

- item_done "indicates that the request has been completed"
  - What the heck does that mean?
  - Short answer: A lot less than you may think it does

- In reality:
  - indicates that the driver is allowed to request another item.
  - Indicates that the sequence can move past wait_for_item_done

Embrace the pipeline

- Even if your protocol ISN'T pipelined, the sequencer API IS.
- Use separate requests and responses, regardless of low level protocol
- What does it look like...?

Many ways to skin the cat...

Request Transmission	Response Handling

`seq_item_port.get_next_item(req);`
`  //...`
`seq_item_port.item_done();`
	`seq_item_port.put_response(rsp);`

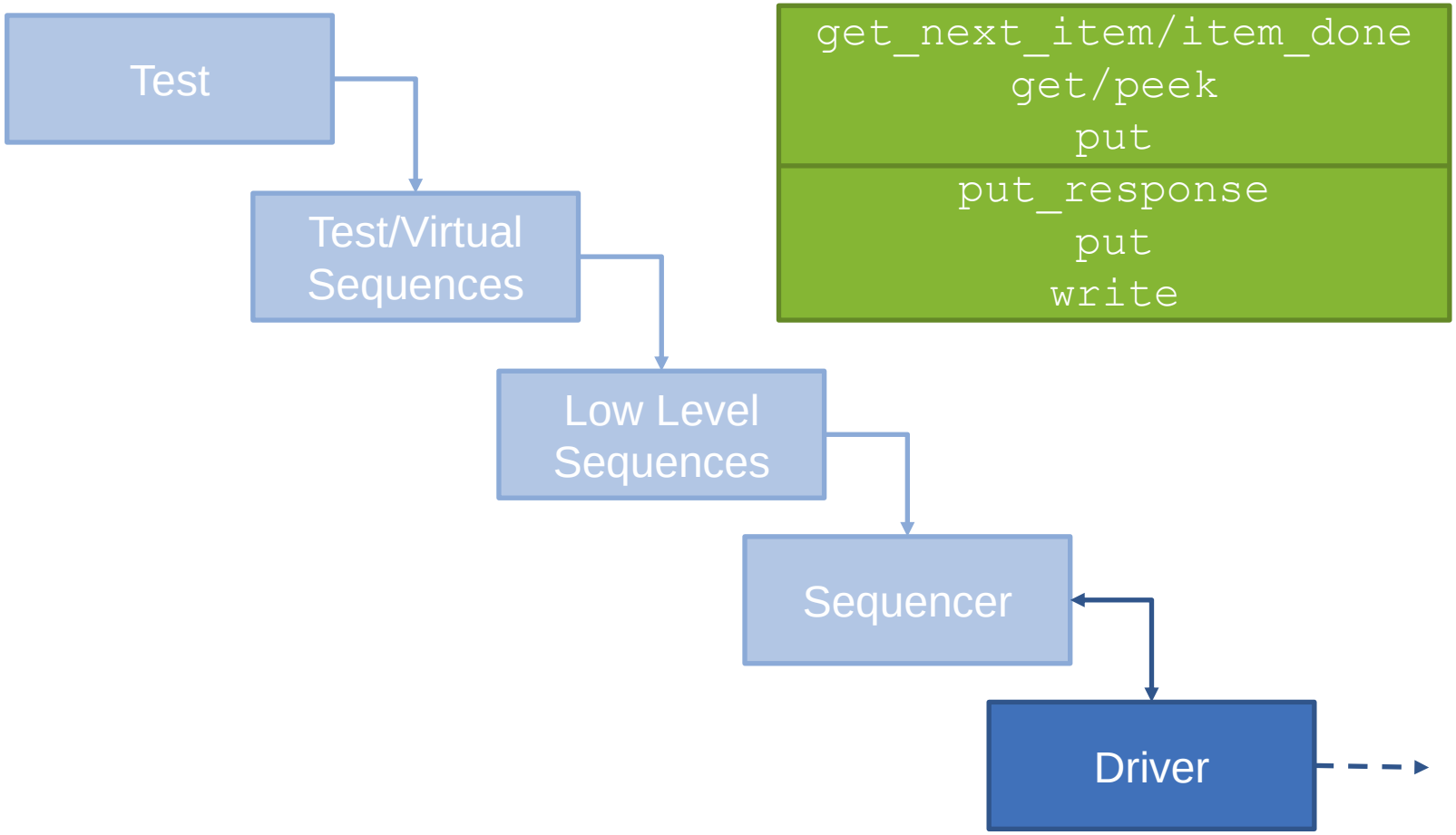
`seq_item_port.peek(req);`
`  //...`
`seq_item_port.get(req);`
	`seq_item_port.put(rsp);`

`virtual task put();`
`  //...`
`endtask : put`
	`rsp_port.write(rsp);`

Additional Benefits

- Drivers can change functionality/protocols without the sequences necessarily needing to be updated
- Extending the response objects allows extended mode drivers to send additional responses back to the test
- Transaction recording can be safely handled within the driver
  - In *addition* to the built in automatic item recording (not *instead* of)

UVM Stimulus



Thank You