## Architecture options discussion

Topology

Where do components sit within and between MHS and DS?

Review four options from Friday session – see next 4 pages

Use of queues

One organisation per queue?

Multiple per queue? Selective consumers?

How does the choice affect outbound queues where endpoints may be down?

Put full message in queue, or header in queue and message in DB?

How are the choices implemented in Rabbit?

Data sharing/distribution agreements

## Option 1 – each product has own queues

## 

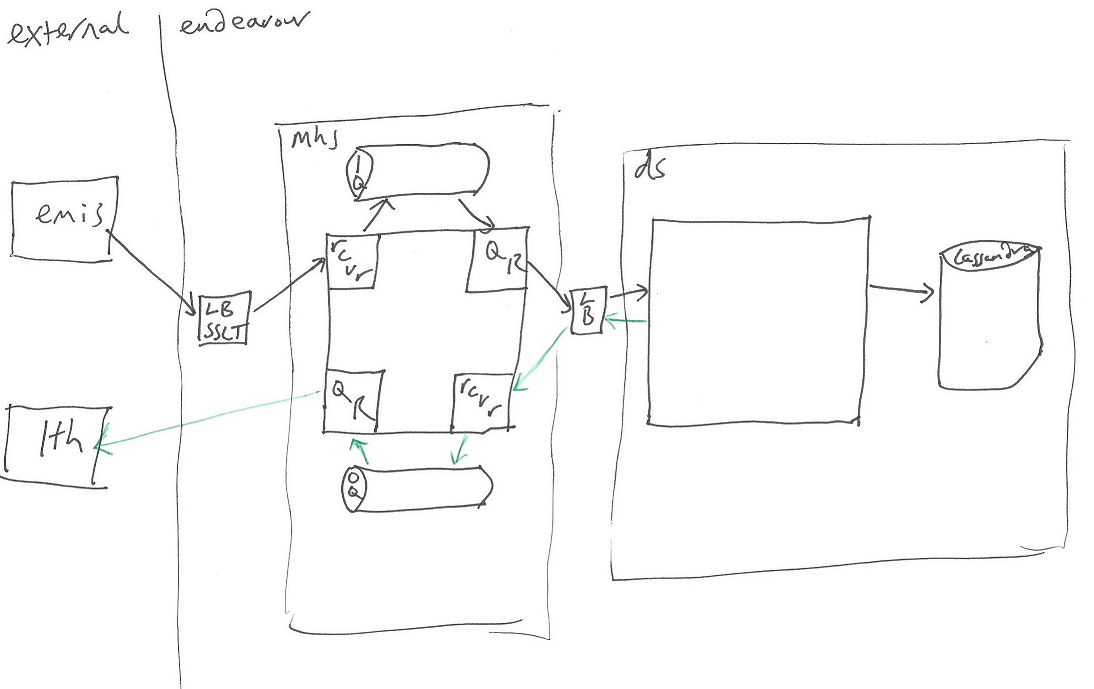
Disadvantages

* Complexity of four sets of queues
* DS must be able to handle MHS being down and implement retry

Advantages

* Consistency between sync and async interfaces – both use sockets
* Async MHS pattern re-usable for external – external “pass through” case
* MHS can receive async messages while DS is down

## Option 2 – MHS has queues, DS is synchronous



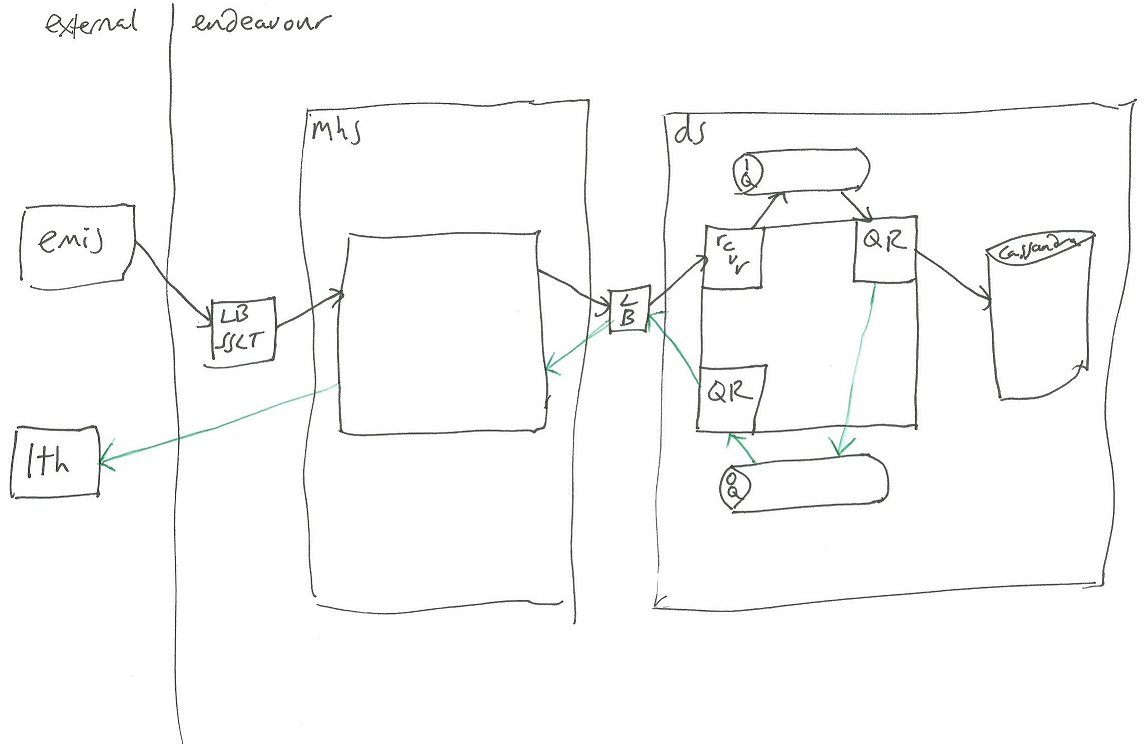
Advantages

* Consistency between sync and async interfaces – both use sockets
* Async MHS pattern re-usable for external – external “pass through” case
* DS is simpler (and doesn’t have to implement retry)
* MHS can receive async messages while DS is down

Disadvantages

* Long(-ish) lived socket connection between MHS and DS

## Option 3 – MHS is synchronous, DS has queues



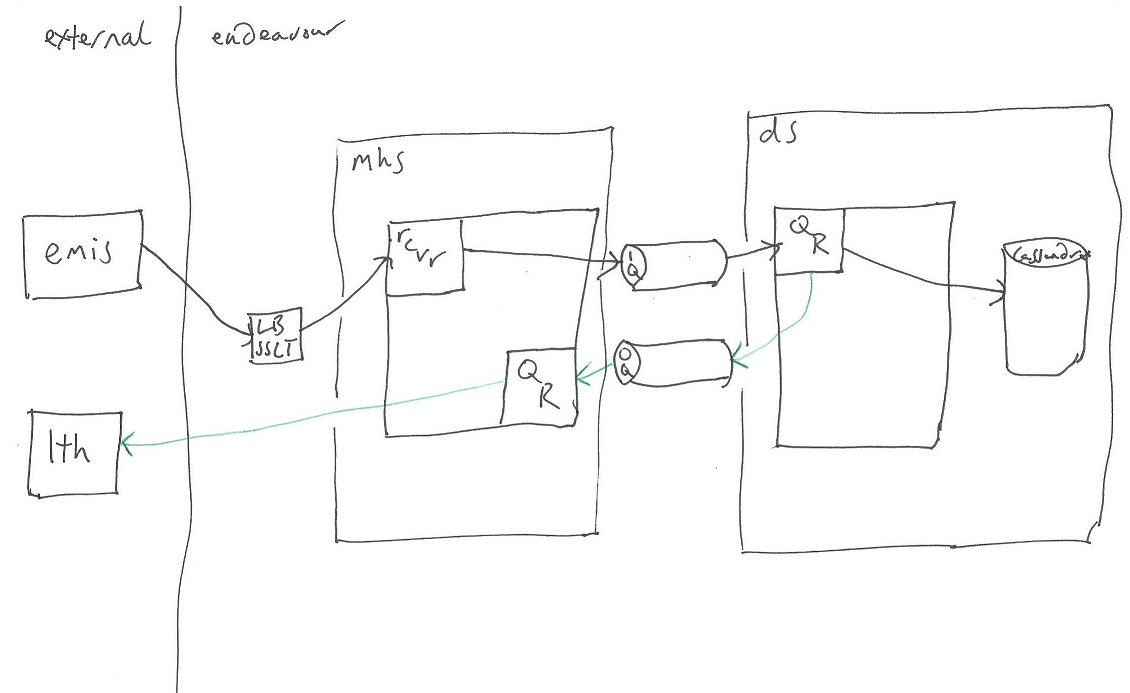
Advantages

* MHS is simpler

Disadvantages

* DS is exposed to external downtime and has to know about message/recipient specific retry mechanisms
* MHS doesn’t support async pattern
* Async message cannot be received without DS being up

## Option 4 – Queues are used as transport between DS and MHS



Disadvantages

* Different interfaces for sync and async
* MHS async pattern NOT re-usable for external – external “pass through” case

Advantages

* Simpler – only two sets of queues
* MHS can receive async messages while DS is down
* DS doesn’t have to implement retry