I like this analogy.

FedEx Analogy – 1

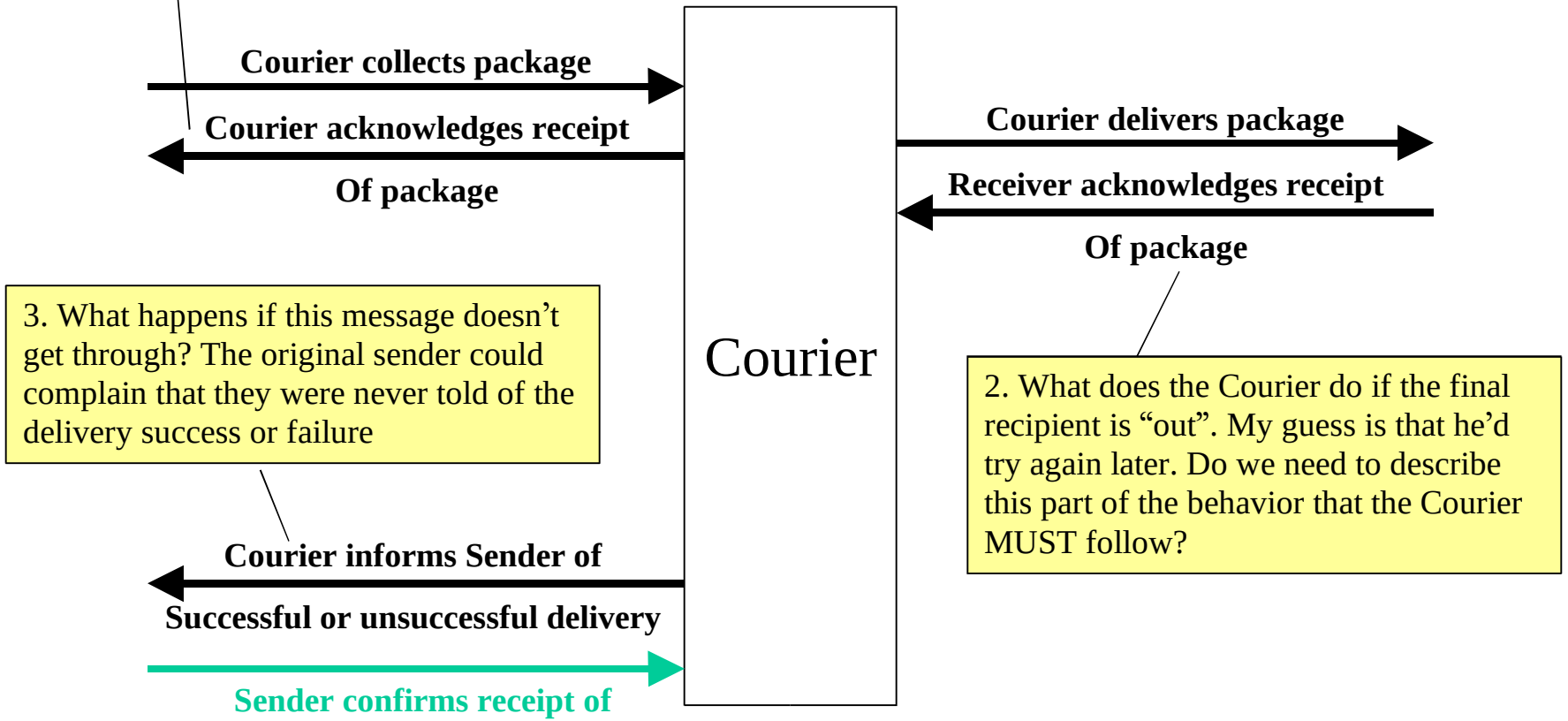
When using a courier service such as FedEx, the sender of a package has no control over how they implement the delivery. It is simply a black box between the Sender (From) and Receiver (To) parties. The Sender has a contractual relationship with the courier.

2. The Courier will deliver the package and not lose it.
3. The Courier will accept delivery of the package from the Sender and issue them with a receipt for it. Once the package is in the Courier's possession, the Sender assumes it will be delivered under the terms of the contract.
4. If required, the Sender can request an acknowledgement from the Courier that confirms the package has been delivered to the ultimate recipient.
5. If for some reason the package cannot be delivered, then the Courier will inform the Sender and may ultimately return the package to them.
6. The recipient will acknowledge to the courier that they have taken possession of the package.

to get the receipt. Unfortunately, if the receipt is being sent electronically, you might not. Do we describe what the Sender should do if they don't get a receipt?

FedEx Analogy – 2

This corresponds to an interface of:

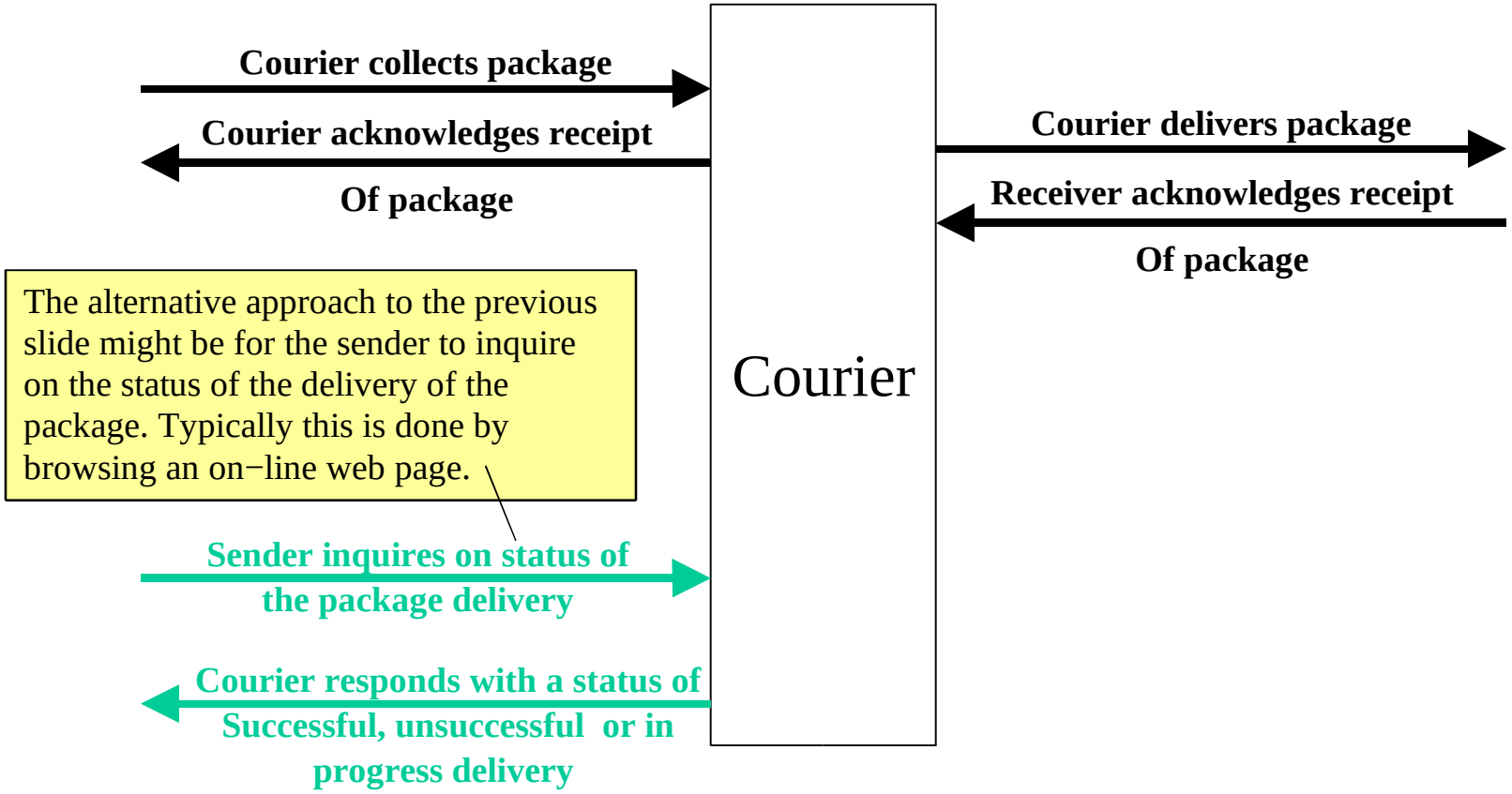


3. What happens if this message doesn't get through? The original sender could complain that they were never told of the delivery success or failure

2. What does the Courier do if the final recipient is "out". My guess is that he'd try again later. Do we need to describe this part of the behavior that the Courier MUST follow?

4. Perhaps we need a response from the sender, that the "delivery result" message was received. If Fed Ex does not get the response, then it should follow a backup procedure to inform the sender by other means if it is important.

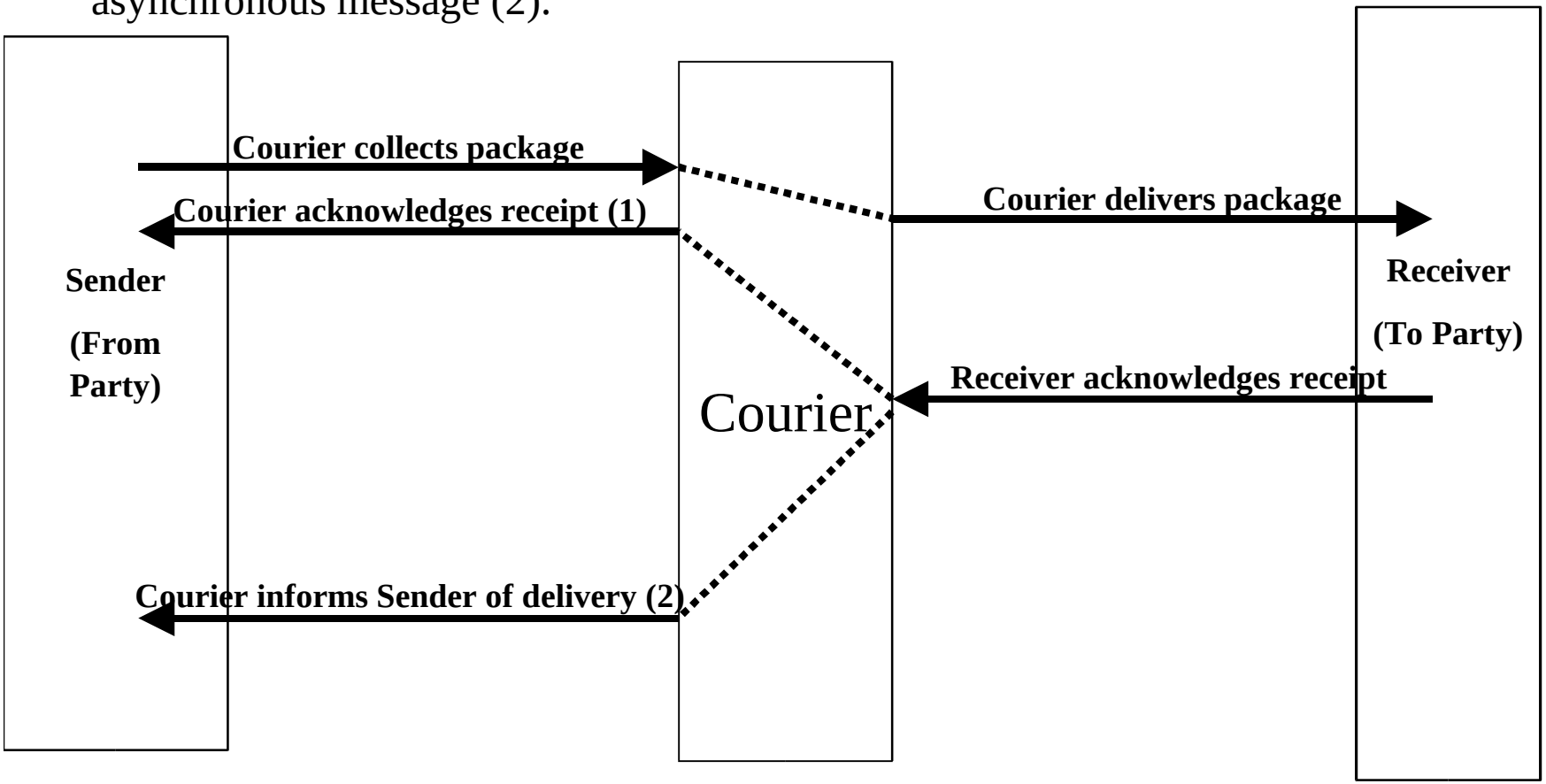
FedEx Analogy – 2a



the messages were being sent over SMTP? Can we actually separate synchronous/asynchronous behavior from the underlying transport protocol? I think we should. Thoughts?

Single H_

Note that if the courier is a simple pass−through then this becomes the single hop case. Receipt acknowledgement may be either synchronous (1) or a separate asynchronous message (2).



Black Box Proposal – Sender (From Party)

Using the FedEx analogy, there are two things the Sender (From party) can do:

2. Hand over the message for delivery (reliably or best effort) to FedEx

   a. Receive an acknowledgement that FedEx have got it.

c. Request from FedEx that a receipt be obtained from the To party and returned to the Sender (by FedEx) once delivery completed.

- The Sender will ultimately receive confirmation that the message was delivered.
- Alternatively an error message will inform the sender that the message wasn't delivered. Note that the error and confirmation will originate from the Courier since that is who the Sender has the contract with. A Sender will then have their own strategy (complementing the Courier) for retry.
- An implied error will exist if acknowledgments are not received from the Courier within some agreed time or a failure occurs in the reliable delivery semantics.

The implied error is, I think "delivery status unknown". I think this will frequently be insufficient as the sender does not know what to do to recover. An Inquiry transaction would allow the Sender to discover the status, even if there is no confirmation of delivery.

Black Box Proposal – Receiver (To Party)

As the ultimate Receiver (To party), I can receive a message sent to me by FedEx and send an

Acknowledgement of that receipt to FedEx. What FedEx does with that receipt is not my concern.

They may simply use it for internal auditing or forward it back to the original sender (From party)

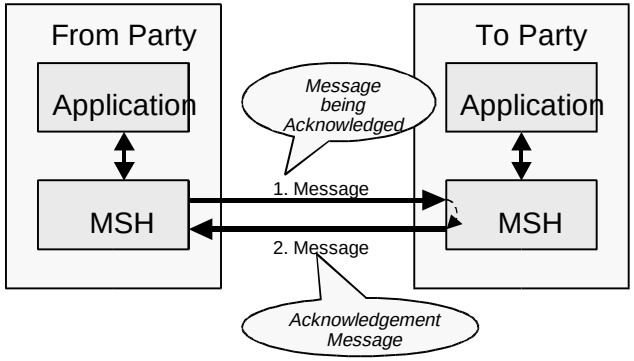
If they requested a copy of the receipt

One area where the analogy with the Real World breaks down, is that if you are sending a physical package, you can't replicate the package and send it twice.

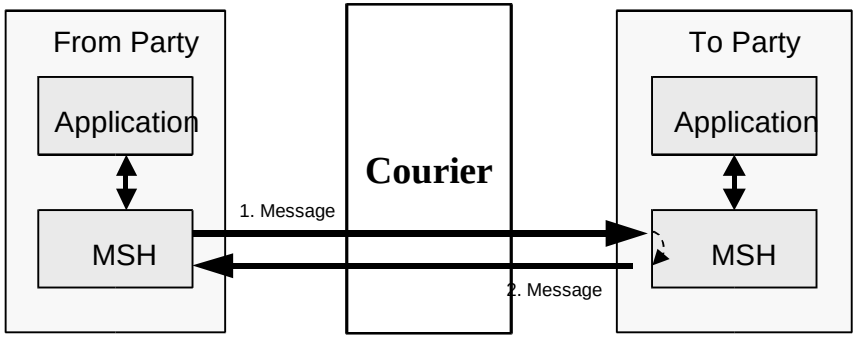
In the virtual world, you can, therefore you can receive the same message twice. However you will want to deliver the message only once to the end application. Therefore you need to filter duplicates, and keep message ids for received messages in storage.

None of this applies in the real world.

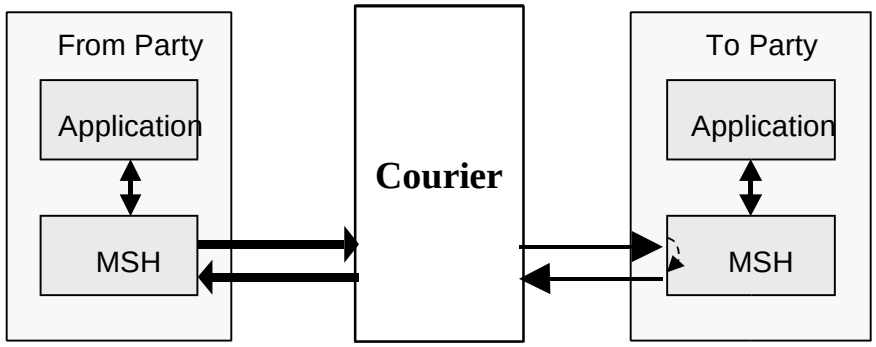
Alternative Approach …..



1. Start with the single hop case



2. Add a "null" intermediary that acts as a pass−through



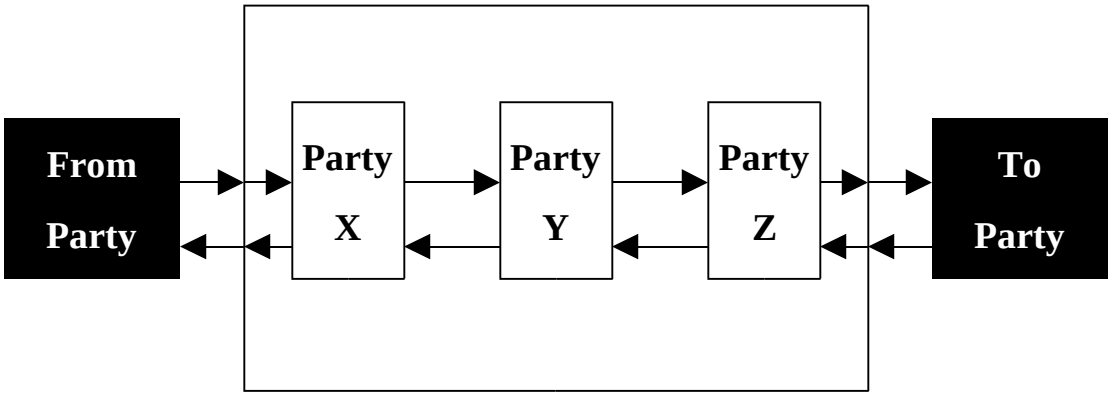
3. Intermediary becomes a black box and its boundary

Establishes the interface.

I like this way of expressing the idea. We should put it in the spec.

Extending to multi−hop

Can now extend this to the multi−hop (intermediary) situation. Providing interface between the From and To parties is maintained, then anything can happen within the black box. We can also thing of the multi−hop model as being nested (black boxes within black boxes) or cascaded (a mapping of the nested case).



I like this too.