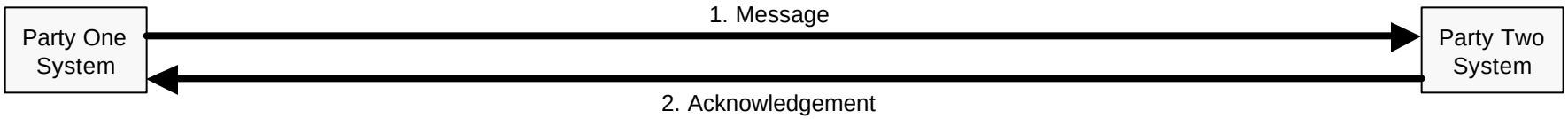
Multi-Hop Reliable Messaging Use Cases
Discussion paper for Tokyo F2F

David Burdett

Commerce One

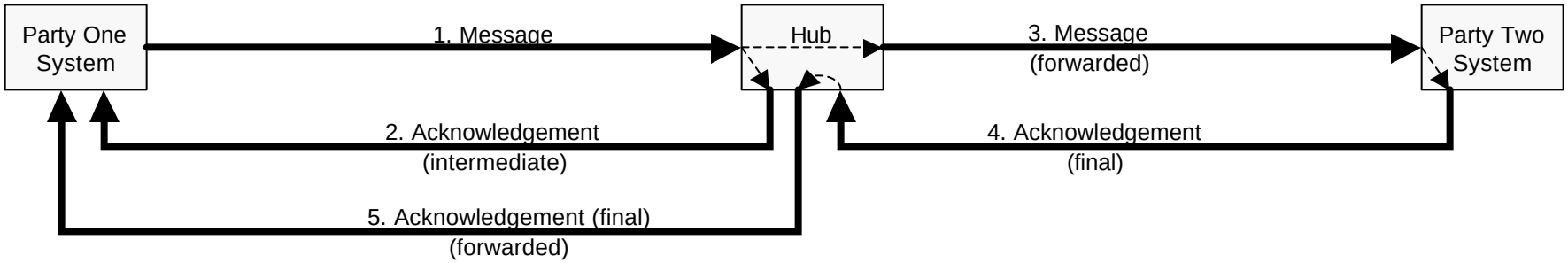
November 2, 2000

Use Cases – Explicit Acknowledgements



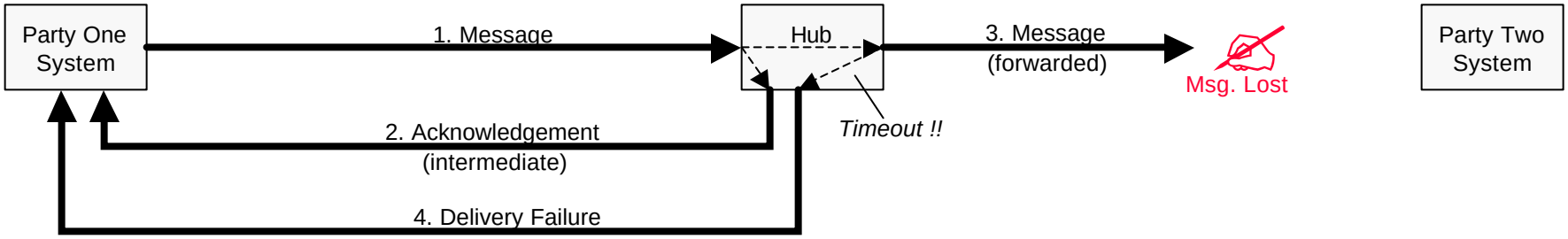
- A sender of a message may need to know that it has reached its final destination
  - requires final destination to acknowledge receipt
- ***Note*** *that the various internet servers, firewalls, DMZ's etc through which the message goes have not been shown as none of these would actually "look" at the ebXML header or process it in anyway.*

Use Case 1 - Acknowledgement by Destination



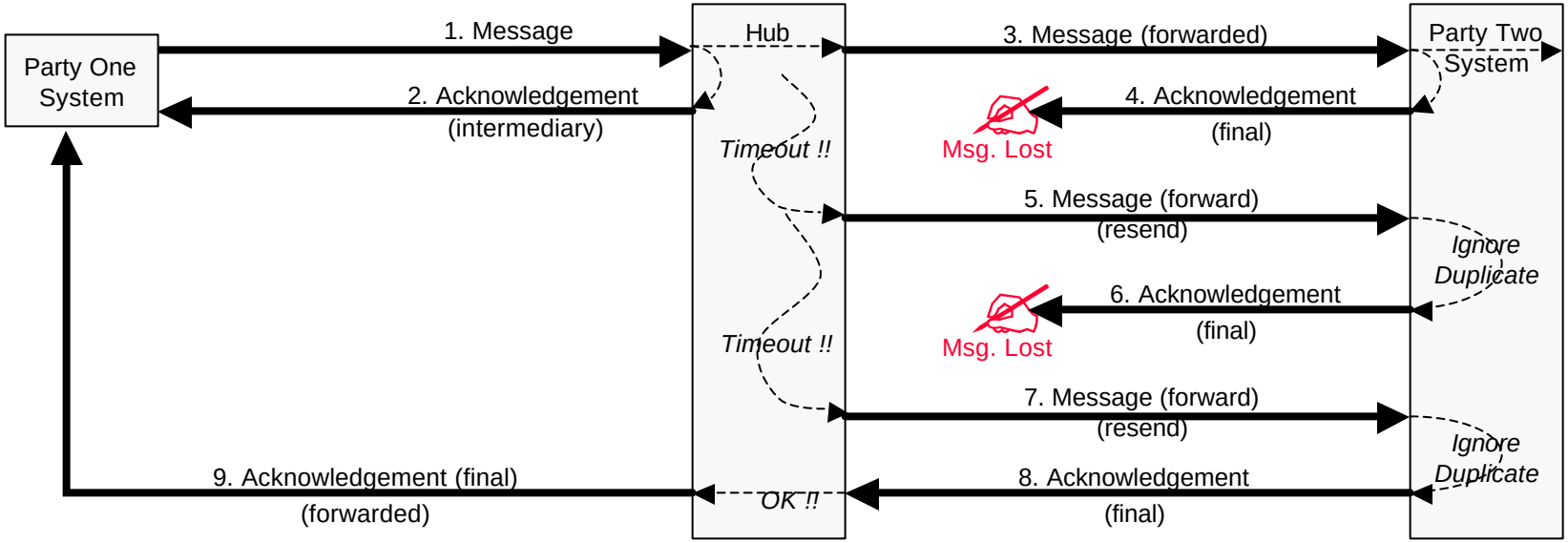
- A sender of a message may want to know that the message has reached an intermediate point - for example if one leg of trip uses a slow protocol such as SMTP
- Requires intermediate destination (hub) to:
  - acknowledge initial receipt, and
  - forward receipt from final destination
- ***Note*** *Forwarded messages (e.g. Message 3) contain the same message header as the message that caused it to be sent (e.g Message 1). Only the routing header changes (or a new one added).*

Use Case 2 - Intermediate Acknowledgement



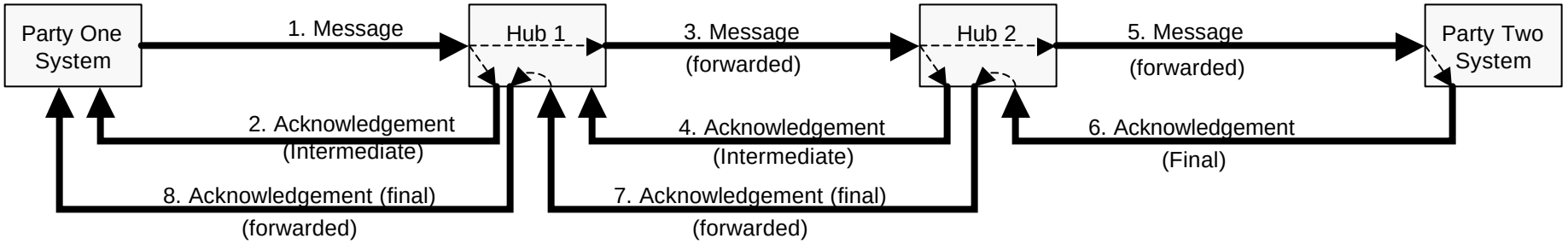
- A sender of a message may need to know if a message could not be delivered to its final destination
- Require intermediate destination (e.g. a hub) to:
  - notify the original sender of a message of its failure to forward it to its final destination

Use Case 3 - Reporting failed message delivery



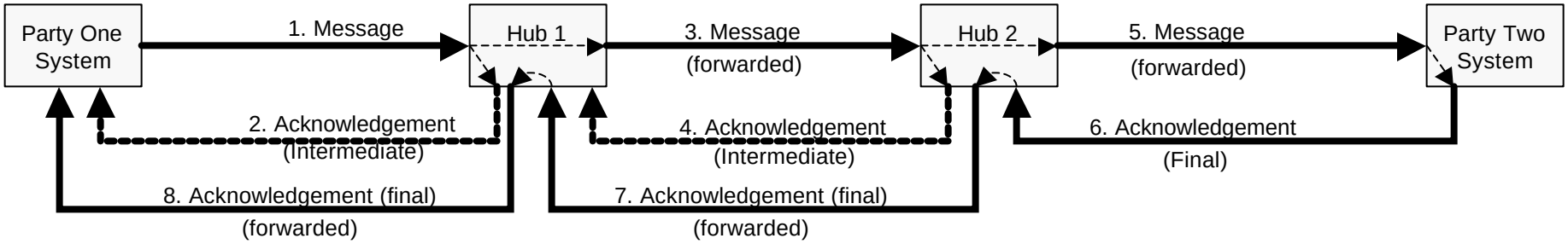
- A sender of a message should try several times before reporting failure
  - a second or later try might work
- Must work for inbound and outbound messages
- Requires recipient to ignore duplicates

Use Case 4 - Retry after failed delivery



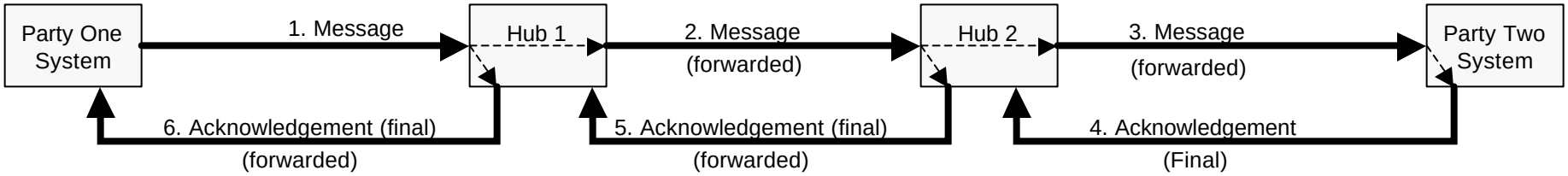
- The Final acknowledgement from Party Two should be sent back to Party One
- Earlier requirements (2, 3 and 4) must work equally well:
  - Over multiple hops
  - If different protocols/standards are used along the way

Use Case 5 - Support for multiple protocols



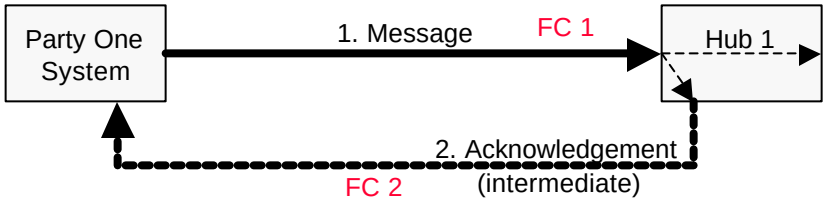
- Intermediate acknowledgements (messages 2 & 4 in diagram) are not needed if final acknowledgement can be returned within the time required

Use Case 6 – Optional intermediate acks



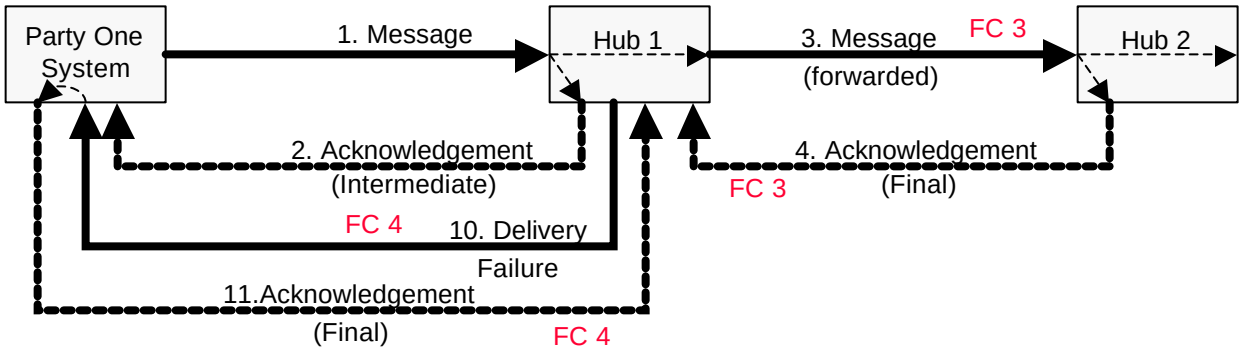
- There may be no need to send back the Final Acknowledgement to party one if:
  - Party one has received an intermediate ack, and
  - Party one knows that he will be informed of delivery failures

Use Case 7 – Optional Final Ack



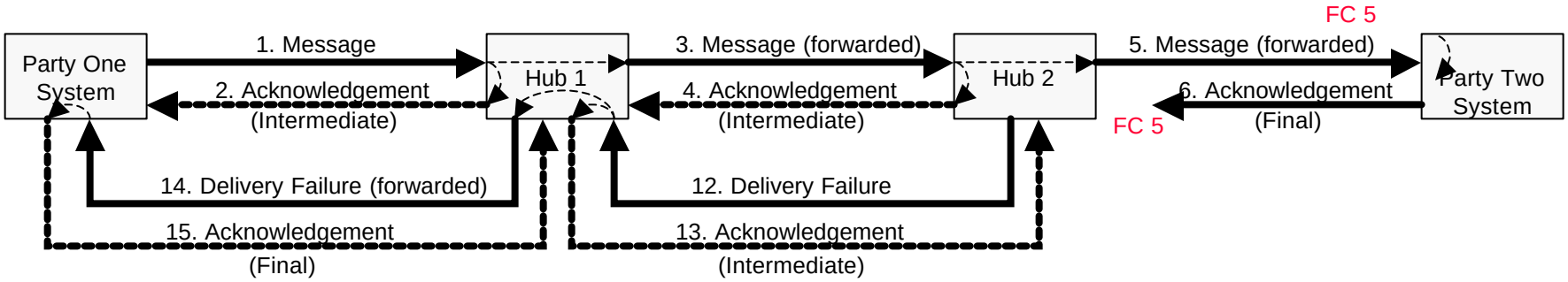
- Fail Case 1: If message 1 is not delivered:
  - re-send message 1 until message 2 is received, or
  - eventually give up
- Fail Case 2: If message 2 is not received:
  - same as Fail Case 1

Failure Analysis - 1



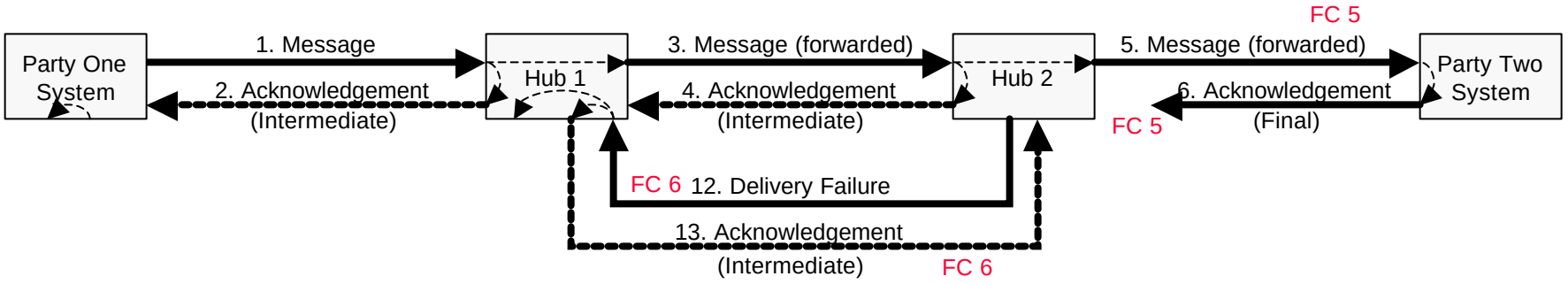
- Fail Case 3: If message 3 is not delivered or message 4 not received:
  - re-send message 3 until message 4 is received, or
  - eventually give up and send message 10 Delivery Failure
- Fail Case 4: If message 10 not delivered, or message 11 not received
  - re-send message 10 until message 11 received, or
  - eventually give up and notify TP1 by other means if required

Failure Analysis - 2



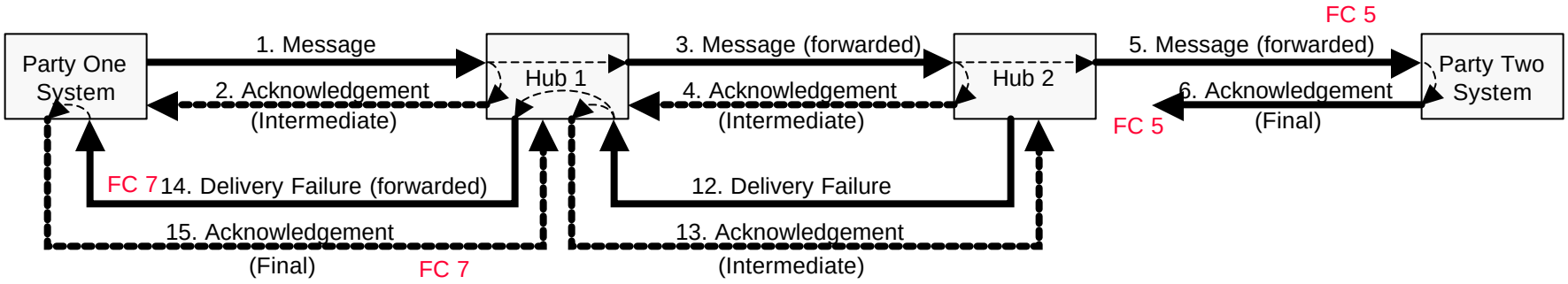
- Fail Case 5: Message 5 not delivered or message 6 not received:
  - re-send message 5 until message 6 received, or
  - eventually give up and send message 12
- ***Note*** *The Delivery Failure needs to be acknowledged since Party One wants to know of the failure*
- ***Note*** *The Delivery Failure indicates that the message <u>probably</u> didn't get through. There is still a possibility that message was delivered and just the ack failed.*

Failure Analysis - 3



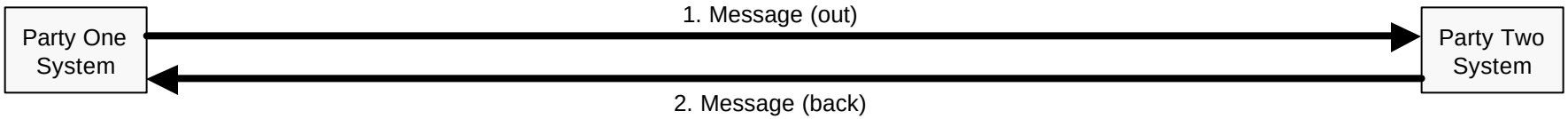
- Fail Case 6: Message 12 (delivery failure) not delivered or message 13 not received:
  - re-send message 12 until message 13 received, or
  - eventually give up and rectify by other means, e.g. Hub 2 operator contacts hub 1 by email, telephone, etc.

Failure Analysis 4



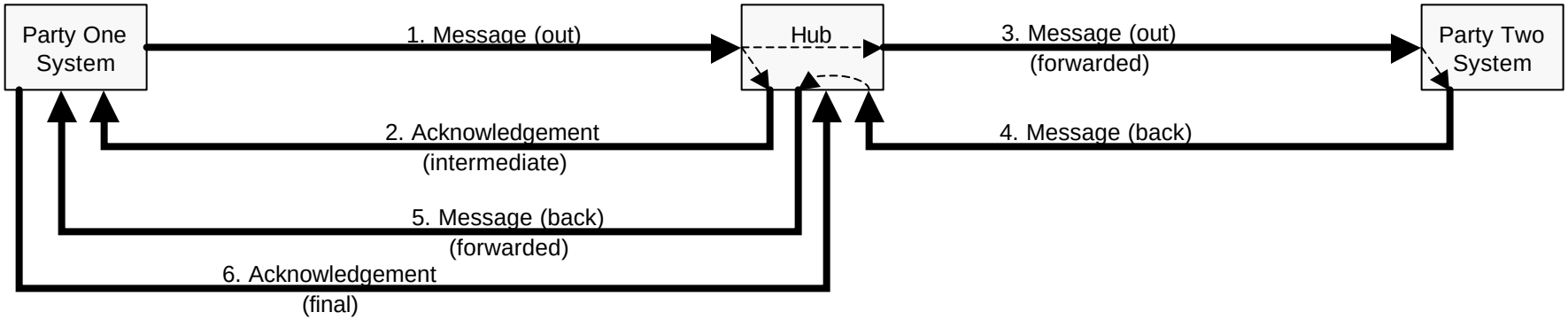
- Fail Case 7. Forwarded Delivery Failure (or its acknowledgement) is not delivered.
  - Resend message 14 until message 15 arrives, or
  - Eventually give up and rectify by other means
- ***Note*** *Hub 1 could in this case report to Hub 2 the delivery failure of the delivery failure message (no 14) but this is probably going too far ...*

Use Cases – Implicit Acknowledgements



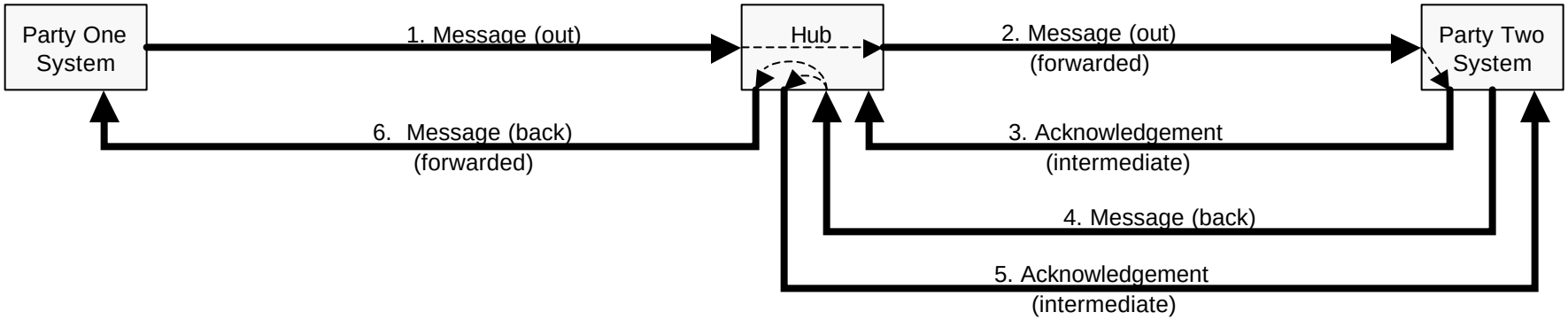
- The Message (back) acts as an implied acknowledgement for the Message (out).
- Party One knows, once the Message (back) has been received that Party Two received the Message (out)

Use Case 8 – Implicit Acknowledgements



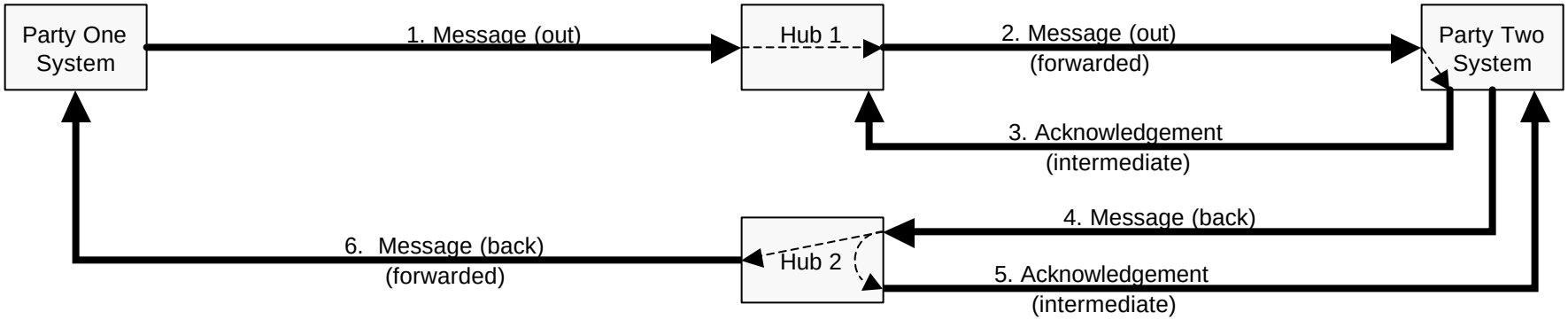
- An intermediate Ack may still be required if, for example, one leg of trip uses a slow protocol such as SMTP
- Message 5 also needs an ack (message 6) if it is to be delivered reliably
- ***Note*** *Message 4 acts as an ack for Message 3. See later use case for what happens if Message 3 (or 4) fails to be delivered*
- ***Note*** *Forwarded messages (e.g. Message 3) contain the same message header as the message that caused it to be sent (e.g Message 1)*

Use Case 9 – Intermediate Acks



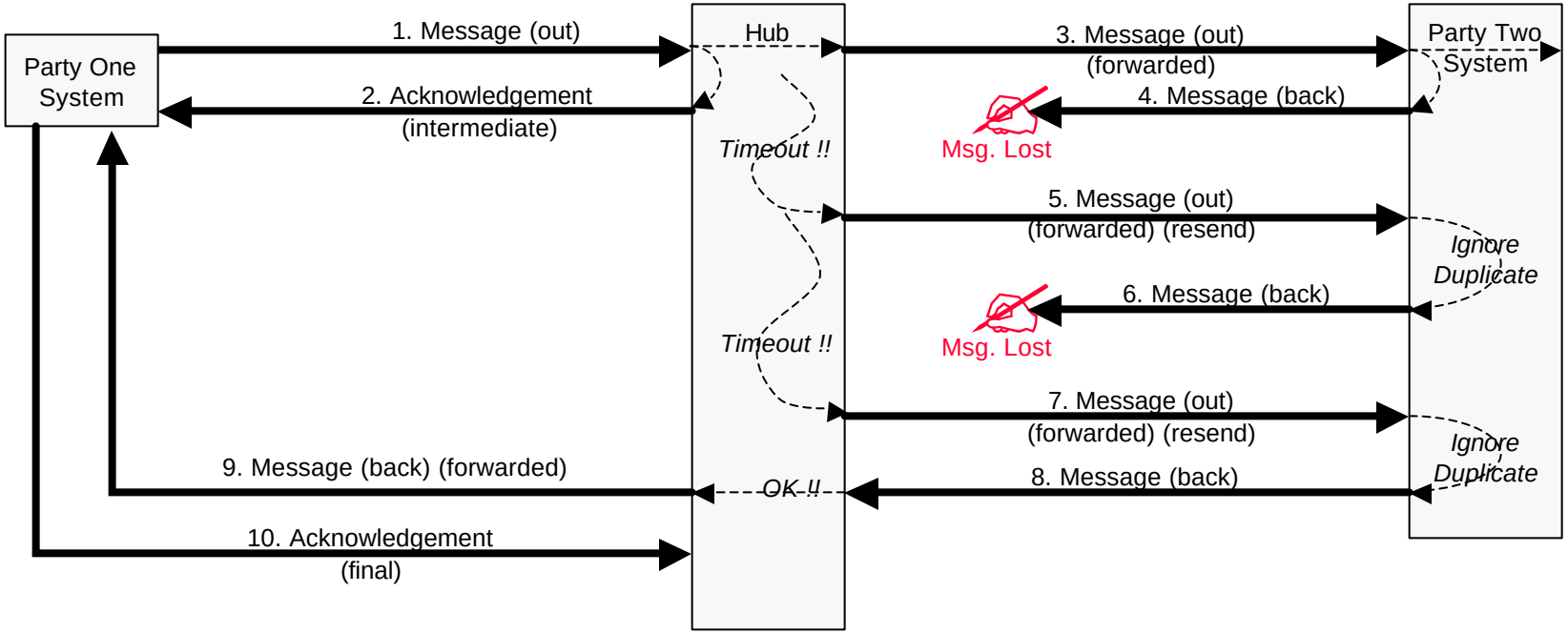
- Implicit and explicit acks may be combined on a separate hops in a multi-hop routing, as only one message can be the “ack” for an earlier message
- Implicit and explicit acks must not be mixed on a single conversation over one hop

Use Case 10 – Acks on any hop



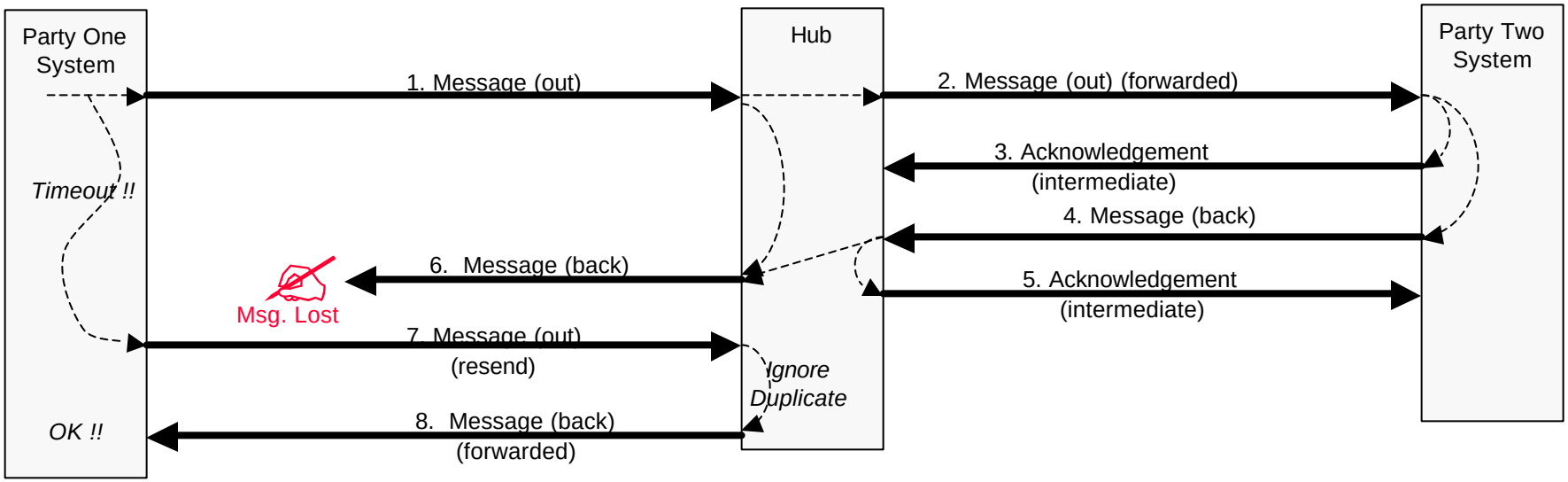
- A message may be returned by a different route to the route that was used to send the original message

Use Case 11 – Different return route



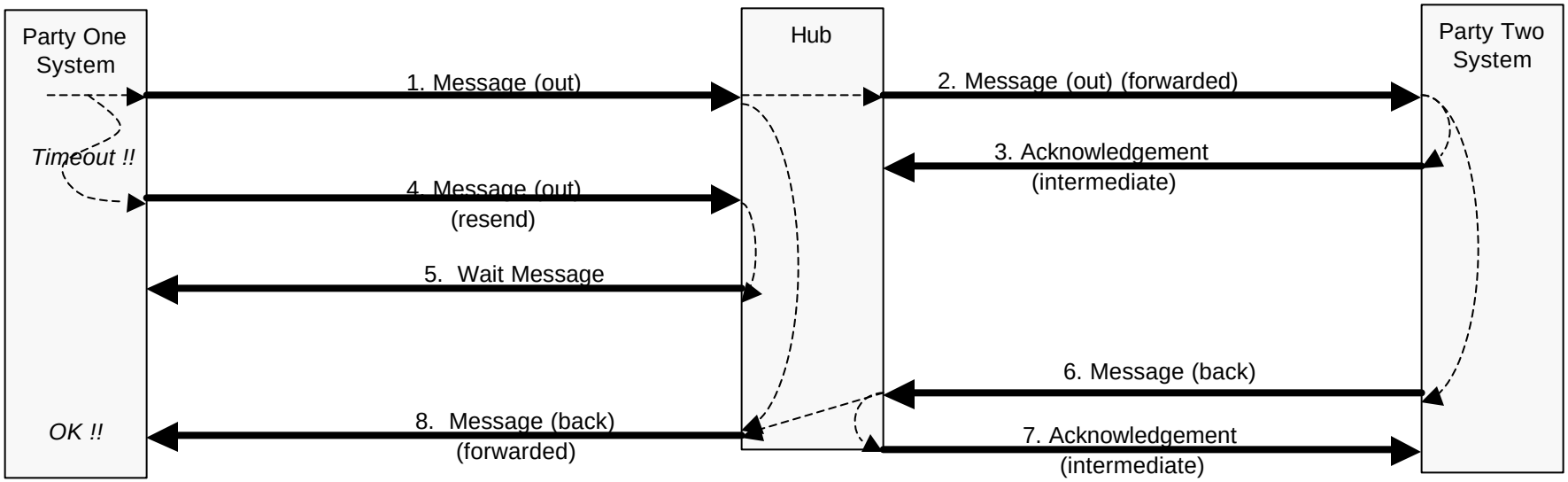
- Message 8 is an implicit Ack of Message 7
- Message 9 also needs an Ack if it must be delivered reliably (message 10)
- *(no new requirements)*

Use Case 12 – recovery and implicit acks



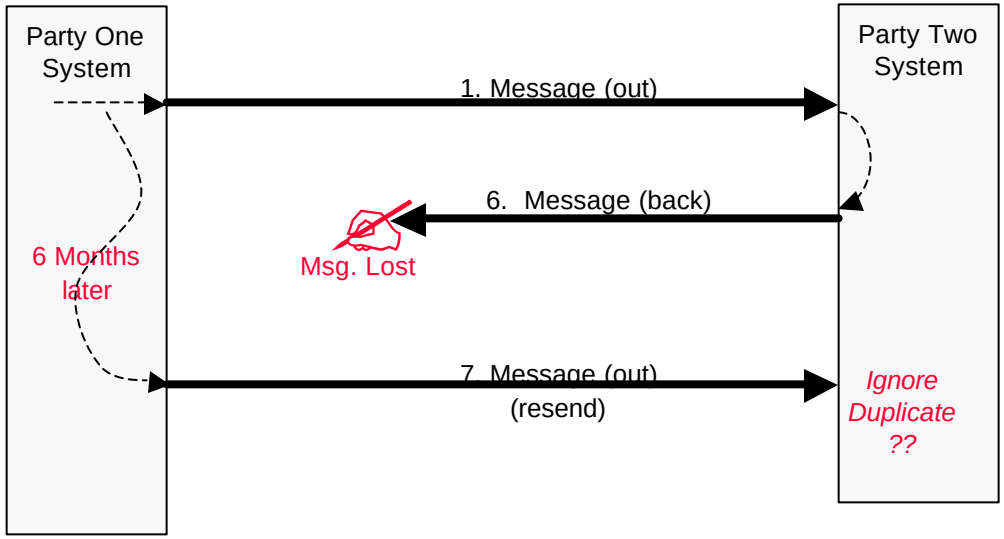
- Intermediate MSHs needs to know that Message 4 from Party Two is implicit ack to Message 1 from Party One. Requires Message 4 to contain “Ref to” to Message 2
- If an intermediate MSH receives a duplicate it should send back the last message it sent

Use Case 13 – Tying Implicit and Explicit acks together



- Wait message tells party one to stop resending and wait for message to arrive
- Same approach applies if Party Two had received a duplicate message before a response could be sent
- *(this is the reason for the "transient error" in the original Error Handling Spec – David Burdett)*
- ***Note** Message 5 acts as an implicit intermediate ack for message 1 (and 4). The implicit final ack that Message 1 has reached its final destination is provided by Message 8*

Use Case 14 – Resend before response is ready



- How long should Party Two retain a message so that it can check for duplicates – 2 hours?, 2 days?, 2 weeks?, 2 months? 2 years????
- Need a concept of a lifetime for a message for duplicate filtering purposes
- Is this set in the message?, in the CPA?, in the CPP?, or any/all of these

 Use Case 15 – extended delays in resends

Summary Requirements

- Reliable Messaging must work equally well over multiple and single hops (UC 5)
- Each hop may use a different standard/protocol (UC 5)
- The final MSH MAY acknowledge receipt of a message (UC 1)
- An intermediate MSH MAY, if required, acknowledge receipt of a message that it will forward to the MSH that sent it the message (UC 2)
- An intermediate MSH MAY omit an intermediate ack if the final acknowledgement can be returned in time (UC 6)
- An intermediate MSK MAY omit the return of a Final Ack if Delivery Failures are reported
- An intermediate MSH MAY, if required, report the failure to successfully forward a message to the MSH that sent it the message (UC 3)
- An intermediate destination MAY, if required, need to re-send a message it is forwarding before giving up (FA 1)

Summary Requirements - 1

- The “final” acknowledgement message received from the final MSH MAY, if required need to be sent – via any intermediate MSHs back to the MSH that sent the original message (UCs 1 & 5)
- A MSH that receives a delivery failure message MAY, if required, need to be sent, via any intermediate MSHs back to the MSH that sent the original message (FA 2)
- Final Acknowledgement messages and Delivery Failure messages themselves MAY, if required, need to be sent reliably using acks (FAs 2 and 3)
- An Implicit ack MAY replace an Explicit Ack (UC 8)
- Implicit acks and Explicit acks MAY be used on each leg of a multi-hop route (UCs 9 & 10)
- Implicit and explicit acks MUST NOT be mixed on a single conversation over one hop (UCs 9 & 10)

Summary Requirements - 2

- An MSH may receive a message that is an acknowledgement from a different MSH to the one that was sent the original message.
- An Intermediate MSH MUST keep track of messages that were sent as an acknowledgement to an earlier message and send back the latest message sent (UC 13)
- An MSH SHOULD send a Wait Message if a duplicate message is received before the response (either an implicit or explicit ack) is ready to be sent (UC 14)
- Messages MUST have a maximum lifetime for duplicate filtering purposes (UC 15)

Summary Requirements - 3

Sender and Receiver Rules

The following two pages contain “rules” that govern the behavior of a sender or receiver of a reliable message. They are a start and are not yet complete but reflect the ideas in earlier pages.

The behavior of the *Source MSH* that is a sender of a *normal message*, that is not a message that is acknowledging another, follows these “rules”:

1. If the message that was sent (the “*original message*”) does not result in the return of a message (the “*acknowledgement message*”) that contains a RefToMessageId of the message that was sent then resend the *original message*
2. If, after several resends of the *original message*, no *acknowledgement message* is received then, if required, inform the source of the *original message*. The source may be:
    1. An application or other process that requested the message be sent, or
    2. Another Message Service Handler.
3. If the source of the message is an application or process then the source should be informed. How this is done is implementation dependent.
4. If the source of the *original message* is another MSH (the “*source MSH*”) then, if required send the *source MSH* a Delivery Failure Message
5. If Delivery Failure Messages must be sent reliably then treat them as a *normal message* (see Rule 1 above). If after several resends, no *acknowledgement message* is received then inform the *Source MSH* by other means such as, email, telephone, etc, if required

Reliable Messaging Rules - Sender

The receiver of a *normal message* that is not acknowledging another message follows these rules:

1. If the *normal message* that is received (the *original message*) from a Source MSH contains a Message Id that has not been received *before* then:
    1. Forward the message to its *destination*, and
    2. If an *explicit acknowledgement* is required, generate an *explicit acknowledgement message* and send it to the *Source MSH*
2. The *destination* may be:
    1. An application or other process that needs to receive the message, or
    2. Another Message Service Handler (the *destination MSH*).
3. If the *destination* is a MSH then:
    1. Forward the *original message* to the *destination MSH* with a new Routing Header and treat it as a *normal message* from a *source MSH* (see previous slide)
4. If the receiver of the *normal message* is the *final destination* of the message then any *explicit acknowledgement* must be a *final acknowledgement* otherwise it is an *intermediate acknowledgement*

Reliable Messaging Rules – Receiver 1

5. If the *normal message* that is received from a Source MSH contains a Message Id that has been received before then:
   1. Do not forward the message to its *destination*
   2. If a message has been sent to the *Source MSH* that acknowledges the *original message,* then resend the message to the *Source MSH*
   3. If no message has been sent to the *Source MSH* that acknowledges the *original message,* then send a *wait message* to the *Source MSH*
6. "*Before*" implies that the MSH persists *MessageIds* for some period of time. The way this time is decided is to be determined

Reliable Messaging Rules – Receiver 2