Middleware to Support Accountability of Business to Business Interactions

In Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

School of Computing Science

Derek John Mortimer

April 2013

Newcastle University

To Dad, Mum and Scott.

Acknowledgements

I have spent close to 10 years at Newcastle University throughout all my studies, I have many people to thank for their support, motivation and guidance through it all.

First and foremost I would like to thank my PhD supervisor Doctor Nick Cook for immeasurable support and guidance and always having an open door when I needed it, I could not have had anyone more fitting for the role.

Of my friends at the University I would like to particularly thank Matthew, Robert and Dylan for help with proof reading, allowing me to bounce ideas off them and help keeping me generally sane. Anna and Matthew deserve special mention for keeping me well fed and sociable towards the end of writing up when I neglected to do so myself.

Thanks are also due to Professor's Santosh Shrivastava and Aad van Moorsel for much appreciated guidance and feedback on papers and this thesis and I would like to graciously thank Doctor Mark Little, RedHat's Newcastle operation and EPSRC for funding my studies under CASE/CNA/07/71 ("Service Oriented Architectures for Business-to-Business Collaboration"), allowing me the time to complete them.

I owe tremendous thanks to my parents for always pushing me to discover and pursue my own goals, always supporting me in my efforts to do so and I wish very much that my Mum were around to see my PhD completed.

The path through my thesis has not been the smoothest and I am very lucky to have a rich circle of support including my family, Scott, Nicole, Aaron, Chris, Ben and Debs, Matthew and Anna, all of whom have been there for me through the difficult and wonderful times of my life while completing my studies, I love you all. I thank Scott especially for his patience over these last few months, it means more than I am able to express.

To the numerous people I have crossed paths with during my time at Newcastle, I would like to say thank you and dedicate this thesis to everyone who has steered me towards its completion, I am eternally grateful.

Abstract

Enabling technologies have driven standardisation efforts specifying B2B interactions between organisations including the information to be exchanged and its associated business level requirements. These interactions are encoded as conversations to which organisations agree and execute. It is pivotal to continued cooperation with these interactions that their regulation be supported; minimally, that all actions taken are held accountable and no participant is placed at a disadvantage having remained compliant.

Technical protocols exist to support regulation (e.g., provide fairness and accountability). However, such protocols incur expertise, infrastructure and integration requirements, possibly diverting an organisation's attention from fulfilling obligations to interactions in which they are involved. Guarantees provided by these protocols can be paired with functional properties, declaratively describing the support they provide. By encapsulating properties and protocols in intermediaries through which messages are routed, expertise, infrastructure and integration requirements can be alleviated from interacting organisations while their interactions are transparently provided with additional support.

Previous work focused on supporting individual issues without tackling concerns of asynchronicity, transparency and loose coupling. This thesis develops on previous work by designing generalised intermediary middleware capable of intercepting messages and transparently satisfying supportive properties. By enforcing loose coupling and transparency, all interactions may be provided with additional support without modification, independent of the higher level (i.e., B2B) standards in use and existing work may be expressed as instances of the proposed generalised design. This support will be provided at lower levels, justified by a survey of B2B and messaging standards. Proof of concept implementations will demonstrate the suitability of the approach. The work will demonstrate that providing transparent, decoupled support at lower levels of abstraction is useful and can be applied to domains beyond B2B and message oriented interactions.

Contents

List of Figures

List of Tables

1 Introduction

Contracts and agreements have long been the standard construct by which parties agree on a set of terms and conditions pertaining to exchanges resulting in mutually beneficial outcomes [Hor07]. The terms and conditions in these contracts seek to simultaneously allow for beneficial outcomes while also protecting the interests of those who do adhere to expected behaviour against those who do not. In an electronic business-to-business (B2B) context, these agreements are descriptions dictating the flow of communication between organisations resulting in outcomes such as the exchange of goods, services and information.

1.1 B2B Interactions

Open networks and enabling technologies have allowed B2B interactions to move increasingly into an electronic setting. Organisations wishing to interact in this manner create appropriate digital business messages containing business documents such as purchase orders or invoices and exchange them with their intended recipient. Recipients process these messages and sanction a response (e.g., a corresponding response message or a real world action such as delivery). These exchanges are enabled by the development of loosely coupled business services to asynchronously transmit, receive and process business messages. These factors enable B2B interactions to become more streamlined; however, higher degrees of automation require that more aspects of interactions are agreed upon beforehand.

For B2B conducted via the exchange of messages as previously described, agreements are encoded as conversations. Conversations describe a complete set of exchanges to achieve desired outcomes, minimally including: the expected contents of messages (syntactic and semantic) and the order in which they are to be exchanged. Higher degrees of automation and regulation may require the specification of elements such as: deadlines for acknowledgement or processing messages, security requirements, evidence of origin or receipt and actions to take should exceptional circumstances arise.

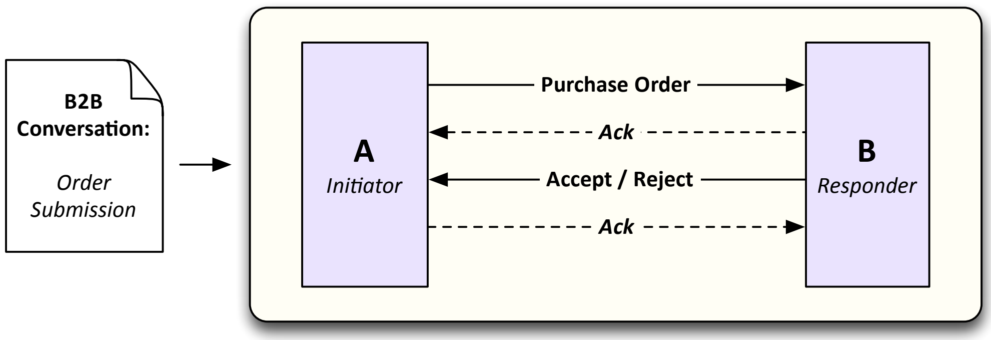


Figure 1.1: *A* and *B* execute an agreed upon *'Order Submission'* conversation.

Figure 1.1 illustrates an example conversation between two organisations for the submission of a purchase order. The conversation specifies that an initiator, *A*, transmits a message containing a purchase order which is acknowledged as being received. Upon processing, the responder, *B*, will transmit a response message indicating whether the initial purchase order was accepted or rejected (i.e., whether the contents were syntactically and semantically valid and the order can be met). The response message is acknowledged as being received, thus terminating the conversation. Possible conversation elements such as deadlines and encryption are omitted for the sake of clarity.

The requirement for complete conversation descriptions, and the benefits of a common base of understanding, drove the development of standards seeking to provide tools to allow the definition and agreement of conversations and their requirements. Available B2B standards range in scope from specifying only the format of messages(e.g., EDIFACT

[Uni11]), specifying a set of permitted conversations including their requirements and how they be satisfied (e.g., RosettaNet [Ros02, Ros09]) and general standards capable of expressing any conversations and their requirements (e.g., ebXML [OAS01a, OAS07b]).

This thesis adopts ebXML as the general B2B standard, capable of specifying any B2B conversation and its requirements[1]. Other standards discussed including RosettaNet and Open Travel Alliance represent subsets of what may be expressed using ebXML [Ros09, Ope11b] and are surveyed with regards to what requirements they support in their respective domains such that commonly supported requirements may be generalised to support the widest possible range of B2B interactions.

ebXML provides a toolbox of building blocks upon which organisations can define and agree upon conversations tailored to their specific needs [OAS01a]. ebXML provides a set of general exchange types to be composed in any order into conversations. These conversations may specify business level requirements, but do not how their execution and support **must** be carried out, ebXML decouples this into conformance profiles specifying message formats, transports and how supported requirements must be satisfied, allowing conversations and their requirements to be tailored as needed. Interacting organisations wishing to use ebXML must agree on a set of conversations to support and an ebXML conformance profile to determine how conversations and requirements are executed.

Domain Specific Standards represent subsets of ebXML by providing a permitted list of conversations and instructions for how these conversations **must** be executed and how the requirements **must** be supported. RosettaNet is a domain-specific standard aimed at the supply chain of major Computer and Consumer Electronics, Electronic Components, Semiconductor Manufacturing, Telecommunications and Logistics companies and the conversations it permits are designed to address these

[1]The requirements permitted are restricted to an ebXML approved list of functional requirements, more on this in Section 2.8.

needs [Ros02, Ros09]. Open Travel Alliance is a domain-specific standard aimed at all aspects of the travel industry. Other domain-specific B2B standards include CIDX aimed at the Chemical Industry, PIDX aimed at the Petroleum Industry. An exhaustive list of all domain-specific standards is beyond the scope of this thesis. Where domain-specific standards are discussed, it is with a view to generalising commonly supported B2B requirements and message exchange patterns.

Following from the ebXML and domain specific standards, this thesis considers the execution and regulation of B2B interactions to have two key aspects:

1. Specifications supporting high-level encoding of agreements as conversations, providing a definition for how organisations must be observed to behave, and

2. Low-level mechanisms enabling the execution of these conversations, the satisfaction of their requirements and the support of their regulation. These mechanisms will generate, exchange and store evidence irrefutably demonstrating where obligations, requirements and properties have (or have not) been satisfied.

B2B standards address the first point, providing definitions of conversations including their functional requirements in well known format. Chapter 3 will generalise common exchange patterns and supported requirements from varying B2B standards discussed in Section 2.7.

Some B2B standards (e.g., RosettaNet) partially address the second point (e.g., by specifying how certain requirements are satisfied); however, this thesis is concerned with providing a generalised middleware design capable of satisfying multiple requirements (e.g., fairness and accountability), with potentially stronger guarantees (e.g., accountability protected from cryptographic key revocation) such that regulation of all B2B interactions may be supported, independent of the B2B standard in use.

Independence from B2B standards requires the middleware designed in this thesis to operate in a decoupled and transparent manner, while this is considered a sound engineering decision [SRC84], there are complexities (e.g., what is the most appropriate point and mechanism of interception) and trade-offs for doing so (e.g., is it possible to rely on standard-specific information while also operating independently of any standard). These complexities and trade-offs are discussed in the remainder of this thesis.

1.2 Supporting B2B Interactions

The ease with which businesses can make themselves available online presents increased opportunity for organisations to involve themselves in B2B collaboration. This can lead to a significant investment of resources into collaborations. While undertaking these high value interactions organisations are assumed to continue to operate autonomously and are likely to privilege their own interests over those of their partners. These factors can lead to a tension between the desire to cooperate and the need to ensure their own interests remain protected.

In addition to this tension, there may be occasions where obligations are not met by one or more participants. Where organisations have complete trust in each other, they are safe in the knowledge that where obligations are not met, responsibility will be taken by the correct participants. Realistically, however, organisations may not have complete trust in each other for reasons including: protecting their own interests first or a lack of previous experience upon which trust can be established. In the face of these concerns, fairness and accountability become critical issues to tackle. If satisfied, they allow the execution and governance of interactions to be supported while also addressing the issues of trust (or lack thereof) and conflicts with a desire to cooperate while protecting one's interests.

Fairness is defined as the property that no well-behaved participant in an interaction is placed at a disadvantage as a result of misbehaviour by another [Aso98]. For example, if two participants are exchanging goods, fairness dictates that at the end of the exchange, both have the desired item or neither do. Any other outcome would be unfair to one of the participants [GPV99].

Accountability is the property that all actions taken within a system are *undeniable*, *certifiable* and *tamper-evident* [YC04]. Accountability is generally achieved by the generation, exchange and logging of evidence binding participants to the actions they take (e.g., proof of origin and proof of receipt). Such evidence forms an audit trail allowing the resolution of disputes by demonstrating where obligations were or were not met. Without this ability to irrefutably resolve disputes, agreements become unenforceable [PG99].

In this thesis, we consider fairness and accountability as desirable concerns to address, in the context of B2B interactions, such that participants are assured that their interests will remain protected in the face of misbehaviour and that all agreements in place remain unambiguously enforceable. Motivation for addressing fairness and accountability (i.e., what happens without such guarantees) is discussed in Section 2.5.2 and 2.5.3.

Implementations of middleware designed in this thesis will use fairness and accountability as examples properties, satisfied by the use of suitable technical protocols. However, the generalised middleware design is capable of allowing other concerns to be addressed. Section 2.5.5 and 3.3.4 discuss how consistency, based on previous work [MJSC07], could be supported as proof of concept of the extensibility of the approach taken in this thesis.

1.3 Support Intermediaries

Current business involves the use of third parties for services such as: identity and credit checks, payment processing and document notarisation. Communication with these third parties can be thought of as the satisfaction of properties supporting an interaction (e.g., a credit check allows a purchase to proceed).

Accountability and Fairness can be considered properties in the same way, their satisfaction allows interactions to proceed with guarantees that agreements remain enforceable and no participants will be disadvantaged as a result of compliance. These guarantees allow organisations to interact focusing primarily on their fulfilling business objectives and obligations while additional support is provided by service providers who deal with the expertise and underlying technicalities of the support they provide.

Of the possible modes of interaction with third party services, intermediaries present a particularly useful approach to supporting interactions. That is, a message being delivered from its sender to recipient may be routed through one or more intermediaries who act upon messages passing through them (e.g., logging, notarising or validation). ebXML and domain specific standards acknowledge the importance of intermediaries in this capacity and discuss their possible involvement [OAS07b, Ros02, Ope11b]. Figure 1.2 illustrates a message passing through multiple intermediaries during its delivery from A to B:

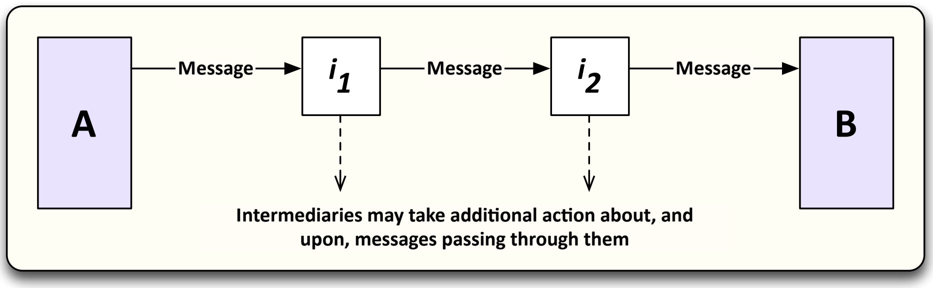

A
Message
i_1
Message
i_2
Message
B
Intermediaries may take additional action about, and upon, messages passing through them

Figure 1.2: A message sent from A to B passing through multiple intermediaries.

1.3.1 Transparent Support Intermediaries

Intermediaries, such as those in Figure 1.2, in the delivery path of a message have the potential to alter the contents of messages routed through them. Any change to the contents of a message may have an unintended impact on the semantics of the message.

By considering the capability to alter contents, we can define a *transparent* intermediary as one whose operation does not alter the contents of the intercepted transmission.

In the example of Figure 1.2, this means the contents of the message, as it arrives at B, are exactly the same as when it was transmitted by A, regardless the number of intermediaries passed through. An important consideration is that A or B may still be able detect intermediary involvement other ways (e.g., increased delivery time for a specific message against a known median) and A and B must trust some subset of intermediary support and connecting infrastructure. These concerns are discussed in Section 4.3 and 2.4.

Transparency can be applied to preserve levels of abstraction, for example, business level transparency ensures that business level operation remains unchanged even when introducing additional lower level elements to provide support such as fairness and accountability in a transparent manner (i.e., by not altering the contents of any intercepted business transmissions).

Provided the above can be achieved, transparent intermediary middleware provides an ideal place to support the execution and regulation of B2B interactions. Support can be offered to existing and new types of B2B interactions, independent of specific standards in use, without any alterations to an organisation's business level operation. Additionally, existing support can be modified or extended by the reconfiguration of existing intermediaries in the delivery process, still maintaining business level transparency.

The middleware designed in this thesis will operate as a set of transparent and decoupled intermediaries, promoting separation of concern and allowing the support offered to be independent of specific B2B standards. The intermediaries will operate asynchronously, just as the B2B services implemented by organisations do, and will use composition to demonstrate support for different properties (e.g., fairness, accountability or consistency) and combinations of properties.

1.4 Intermediary Accountability Support

Previous sections have introduced B2B interactions, supporting their execution and regulation, how this can be achieved and why it is desirable. This section will use the support of *accountability*, satisfied by a transparent intermediary to introduce the topics explored and discussed in the subsequent chapters of this thesis. The intermediary discussed here is simplistic in that it employs only two protocols to satisfy accountability with coarsely defined characteristics. Discussion in subsequent chapters will expand upon the topics introduced here by considering: multiple properties (and multiple characteristics of those properties), multiple protocols to satisfy those properties and characteristics, support for more B2B standards, and multiple mechanisms for declaring when supported properties should be satisfied. The challenges associated are discussed in Section 1.5.

1.4.1 Accountability as a Functional Property

Accountability, as introduced above, can be described as the functional definition that participants are held accountable for their actions. Where this thesis refers to *functional properties*, this indicates a business level concern or requirement (e.g., accountability or fairness) considered in functional terms (i.e., the details of how this is achieved are irrelevant to the discussion).

Thus, for any functional property, its satisfaction ultimately depends on the specification of the imperative details (e.g., protocols, formats and transports). Combinations of different imperative details may satisfy properties with varying characteristics. For example, trust and fairness are discussed below as characteristics affecting how accountability can be satisfied.

Firstly, we classify *trust* as a characteristic whose value is either ‘trusted’ (the sender has complete trust in the recipient) or ‘untrusted’ (all other cases).

Fairness is defined as a binary characteristic defining whether the release of accountability evidence (and the business message being sent) should done in such a way that no participant gains an advantage by misbehaving.

Using these characteristics, we can reason that accountability, without complete *trust*, should be delivered with *fairness*.

Non-repudiation protocols represent suitable protocols to satisfy accountability [KMZ02]. These protocols generate and exchange evidence irrefutably binding participants to the actions they perform while interacting. Two non-repudiation protocols here for the sake of brevity, both resulting in the generation and exchange of a message and two kinds of evidence: *non-repudiation of origin*, irrefutably binding a message to its originator and *non-repudiation of receipt*, irrefutably binding the receipt of a message to its recipient.

The first protocol is the Coffey-Saidha non-repudiation protocol [CS96], mandating the involvement of a mutually trusted third party (TTP) to ensure a message and its associated evidence are exchanged fairly. The second protocol is named "voluntary exchange" in which evidence is exchanged directly (and voluntarily) between participants without a third party guaranteeing the exchange [CRS06]. The involvement of a TTP to ensure fairness, and the type of its involvement (inline, discussed in Section 2.6.3) mean that the Coffey-Saidha protocol incurs a higher cost in terms of messages passed and computational complexity (additional decryption and encryption at the TTP). For the above reasons, it is assumed preferable to perform voluntary exchange where trust relationships allow it.

As per the accountability property and its characteristics described above, voluntary exchange is suitable only where the recipient is *trusted* at the time of exchange (i.e., fairness does not need to be guaranteed) and the Coffey-Saidha protocol is suitable in situations where fairness is required in the exchange of message and its non-repudiation evidence. For simplicity, we assume here that the trust relationship between two organisations is known. The complexities of defining and expressing trust relationships are discussed in Section 2.5.1 and 3.3.1.

1.4.2 When to Provide Accountability Support?

The final piece is determining when accountability should be satisfied for an intercepted message. Section 3.4 discusses all mechanisms for declaring when supported properties should be satisfied. Here we consider the use of existing business level requirements to create a rule inferring when accountability is required, as proof of concept for supporting existing B2B interactions.

ebXML allows messages to specify whether they require “non-repudiation of origin” and/or “non-repudiation of receipt” as binary parameters [OAS07b]. Where a message specifies either of these parameters, the intermediary assumes that accountability is desired and invokes the correct protocol to satisfy it with the desired characteristics (e.g., fairness guaranteed without complete trust in recipient).

A valid question here is why, if ebXML specifies generation and exchange of non-repudiation evidence, would we engage another non-repudiation protocol to satisfy accountability? Section 2.7 will demonstrate that the support specified by ebXML and other B2B standards is susceptible to cryptographic key revocation (rendering evidence unusable) and exchanged without any fairness guarantees.

Transparent intermediaries allow stronger levels of support to be offered (e.g., Coffey-Saidha guarantees fairness and key revocation can be prevented) to better address identified concerns of B2B interactions, discussed in Section 2.5, 2.7 and 3.3.

1.4.3 Intercepted Message Flow

The previous section functionally defined accountability and specified two protocols supporting its satisfaction under different trust characteristics defining when fairness guarantees are required. Rules were also defined for how the requirement for accountability will be inferred. This allows the definition of intermediary accountability support for intercepted messages. Figure 1.3 illustrates intermediary accountability support for an intercepted message:

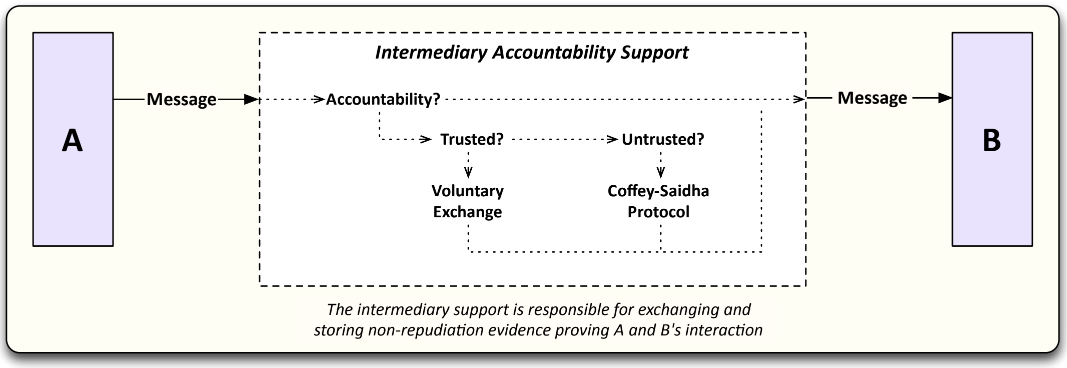

Intermediary Accountability Support
A
Message
Accountability?
Trusted?
Untrusted?
Voluntary Exchange
Coffey-Saidha Protocol
Message
B
The intermediary support is responsible for exchanging and storing non-repudiation evidence proving A and B's interaction

Figure 1.3: An intercepted message being provided accountability by an intermediary.

1. A message is generated by its sender (A) and transmitted.
2. The message is routed through the intermediary support.
3. Once intercepted, the message is analysed to determine if accountability should be satisfied by:
   a) Checking the message to see if its business level requirements infer accountability is required.
4. If accountability is required, which protocol should be invoked?
   a) If the recipient is completely trusted, use voluntary exchange knowing that will provide their associated evidence.
   b) For all other situations, use Coffey-Saidha to guarantee fairness to both participants.
5. Deliver the original message to its recipient (unaltered, to maintain business level transparency between A and B) and reliably store any intermediary generated evidence.

The figure shows a message, transmitted from A to B, passing through intermediary support whose determines whether accountability is required, satisfies the property where necessary using a suitable protocol and finally delivers the original message on to B. All messages are offered three routes through the intermediary support: without accountability, accountability via the Coffey-Saidha protocol and accountability via voluntary exchange and all routes result in the original message being delivered to B, maintaining business level transparency.

This provides a general set of processing instructions for the intermediary support of functional properties, for any intercepted message: determine the required properties (and characteristics), satisfy them appropriately through the execution of technical protocols and finally deliver the intercepted message onwards to maintain transparency.

The dashed edge of the intermediary support in Figure 1.3 denotes that the intermediary support is a composition of smaller services, discussed in the next section.

1.4.4 Composed Intermediary Accountability Support

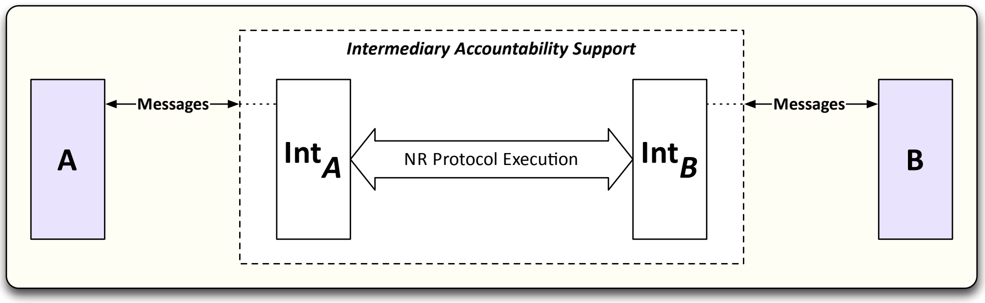


Figure 1.4: An intercepted message being provided accountability by an intermediary.

Figure 1.4 represents a simple decomposition of the provision of intermediary accountability support. A's messages pass through Int_A, a transparent intermediary acting on behalf of A to support its interactions. Similarly, Int_B acts on behalf of B to transparently support its interactions. Int_A and Int_B communicate with each other to execute technical protocols satisfying supported functional properties. Specifically for accountability as discussed in the previous section, Int_A and Int_B communicate to ensure the message is exchanged and the correct non-repudiation protocol is invoked where required. The composition of Int_A and Int_B provide the abstraction of “intermediary accountability support”.

Henceforth, when referring to “A’s intermediary” or Int_A, we are referring to any intermediary acting on behalf of A. Specifically, the intermediary is not necessarily owned or operated by A but A is assumed to have some degree of control over Int_A. For example, declaring when supported properties should be satisfied for intercepted messages, configuring acceptable timeouts or exercising control over the cryptographic keys used to generate evidence on its behalf.

Int_A and Int_B provide a known boundary at which business level transparency can be guaranteed for A and B. That is, for a message transmitted from A to B, routed through Int_A and Int_B, the last point at which transparency can be guaranteed (by intermediary support) is when Int_B emits the message, and vice versa. Additional intermediaries may exist between A and Int_A (and B and Int_B), these are beyond the control of provided intermediary support.

The provisioning of Int_A and Int_B in this manner match the loosely coupled asynchronous exchange of messages facilitating the execution of B2B interactions. A, B, Int_A and Int_B are all standalone entities who may communicate but are not all required to be online at the same time to do so.

The composition of Int_A and Int_B to provide accountability support also illustrates that the decomposition may be recursively applied. That is, Int_A and Int_B are themselves composed of smaller services (discussed in Section 4.8.1).

A strong motivator for this approach is the possible configurations for the location and operation of components, they may all be hosted by the same provider, split across multiple providers or hosted within their respective organisations. This allows elements to be composed across multiple domains of control (e.g., Int_A’s composition may be split across A and a dedicated provider). Section 4.10 discusses such configurations and demonstrates that previous work can be expressed as instances of these configurations.

Section 2.4 discusses issues relating to the trust of components between one or more providers.

1.4.5 Summary

While Int_A and Int_B communicate to execute non-repudiation protocols to provide accountability to A and B, the execution of other technical protocols would allow different properties to be satisfied (e.g., synchronisation protocols to ensure consistency).

Although omitted for brevity, communication between Int_A and Int_B may engage a trusted third party (TTP) to guarantee fairness [PG99] the important elements here were A, B, their respective intermediaries and the illustration of composition to provide intermediary support abstractions. A TTP would simply represent another composition of services to provide required functionality.

"Intermediary Accountability Support" was used as a label in Figure 1.3 and 1.4 to show that Int_A and Int_B (and TTPs) were composed to provide the required paradigm (i.e., using non-repudiation protocols to satisfy accountability). Following this, "intermediary support" in used to mean any support offered to interacting organisations (e.g., A and B) through the use of intermediary components acting on their behalf to execute protocols satisfying one or more support properties with varying characteristics.

1.5 Objectives of Work

Previous work has focused on supporting interactions through the provision of intermediaries. This includes the execution of non-repudiation protocols [CRS06, NZB04], contract monitoring [Str09], synchronisation for consistency management [MJS06], augmenting message delivery with additional capabilities [TMRS02] and ensuring agreements are never violated [MU00]. These works have generally aimed to support one property (e.g., non-repudiation) and placed significant requirements on organisations in terms of:

Expertise: What protocols should be used, what properties do they provide and when should they be used?

Infrastructure: The computing power required to executed the protocols rendering the required support.

Integration: How is the offered support integrated into new and/or existing business processes.

This thesis aims to improve on previous work by designing and implementing generalised support middleware that does the following:

1. Supports multiple functional properties (e.g., accountability or fairness) with multiple characteristics (e.g., accountability when the recipient is untrusted)
2. Provides the desired support through the use of intermediaries which are composed to satisfy the supported functional properties by implementing the required technical protocols.
3. Operates transparently
4. Operates decoupled from B2B standards
5. Operates asynchronously
6. Alleviates expertise, infrastructure and integration requirements placed upon interacting organisations

For **point (1),** the challenges are identifying which properties to support, which protocols satisfy these properties (with what characteristics) and establishing suitable mappings. Discussed in Section 2.4, 2.5, 2.6, 2.7, 3.2, 3.3 and 4.10. A generalised design in which this is possible will be capable of expressing previous work as instances of functional properties to be supported. Discussed in Section 2.12 and 4.12.

For **point (2),** the challenges are decomposing the required support into individual components that may be re-used to provide the widest range of support with minimal duplication of effort or functionality. Discussed in Section 1.4.4, 2.3, 4.2, 4.3, 4.8.1, 4.9, 4.12, 5.3, 5.4 and5.5. Expressing previous work within this generalised design becomes a matter of implementing components to be composed to provide the required functionality. Discussed in Section 2.12, 4.8.1 and 4.12.

For **point (3)**, the challenges are identifying points of interception at which the required information is available to the intermediary support and the desired support can be provided to the interacting organisations, independent of B2B standard, while ensuring business level transparency is maintained. Discussed in Section 1.3.1, 2.2, 2.7, 2.12, 3.4, 3.5, 3.6, 4.7 and5.6. By enforcing transparency in the intermediary support, existing and new B2B interactions can be supported by placing intermediary support in the delivery path of B2B interactions. Discussed in Section 1.4, 2.7, 3.1, 3.4, 3.5 and 3.6.

For **point (4)**, the challenges are ensuring that support offered is not dependent upon any specific B2B standard, that some useful minimal level of support is available to all interactions and also that available business level information may be capitalised upon where available. Discussed in Section 2.7, 3.2, 3.3, 3.4, 3.6 and 4.3. By ensuring the intermediary support is decoupled from any B2B standard, existing and new standards can be supported (assuming they communicate using message oriented middleware). Discussed in Section 3.4, 3.5, 3.6 and 4.3.1.

For **point (5)**, the challenges fall under the realm of an engineering challenge. It is best to implement intermediary support asychronously simply as message oriented middleware and all surveyed B2B standards operate asynchronously. The benefits of doing so are that all components are not required to be online at the same time. However, we assume any component will *eventually* be online such that it can process messages and proceed with executing or supporting interactions. Discussed in Section 2.1, 2.4, 3.1, and 4.3.1.

For **point (6)**, we consider that expertise and infrastructure requirements can be satisfied by dedicated service providers. Such providers would be responsible for implementing component services and composing them into intermediary support that delivered the required levels of support to interacting organisations. This thesis assumes the role of one or more dedicated service providers for the purposes of supporting interacting organisations. Discussed in Section 2.3, 2.11 and 4.11.

By enforcing transparency and decoupling, integration requirements can be minimised to requiring only that an organisation's messages are routed through the intermediary support. Other scenarios for integration (including situations in which organisations wish to retain control over specific aspects) are discussed in Section 2.4, 2.5.3, 2.6.3, 2.11 and 4.11.

The proposed approach to providing transparent, decoupled, asynchronous support at lower levels, using middleware intermediaries, will be demonstrated as fit for purpose and also applicable to domains beyond B2B and message oriented middleware.

Additionally:

1. Fairness and accountability will be developed as proof-of-concept properties alongside technical protocols supporting their satisfaction. They will demonstrate that multiple properties (and characteristics) can be supported independently of B2B standards in use, all while maintaining transparency, loose-coupling and asynchronicity. Consistency will be described as a property that could be implemented (with examples), demonstrating the extensibility of the generalised middleware design.

2. Example implementations will be created, representing different instances of composition and deployment of components to provide intermediary support for B2B interactions. These implementations will be evaluated with regards to how well they function compared to previous work and what impact they may have on B2B interactions.

1.5.1 Summary

In short, the major contributions are a generalised intermediary middleware design for supporting interactions using by mapping functional properties to technical protocols allowing their satisfaction and declaration mechanisms specifying when properties are required for intercepted transmissions.

The challenges of transparency, asynchronicity and loose-coupling are addressed in the support offered, allowing:

- Existing and new interactions to be supported seamlessly
- Previous work to be expressed as instances of composition and deployment within the generalised design.
- Expertise, infrastructure and integration requirements to be alleviated from interacting organisations.

Example properties, protocols and implementations, coupled with generalisation of previous work and surveyed standards will demonstrate the suitability of low-level approach to providing support. Cross domain applications of the work will be discussed in Section 6.3.

1.6 Thesis Structure and Contents

The work presented in this thesis expands upon the topics discussed in this chapter and is structured as follows:

Chapter 2 introduces general B2B terminology, concepts and assumptions made throughout this thesis. Concerns relevant to supporting B2B interactions are presented alongside technical protocols for their satisfaction and considerations for integration into intermediary support. B2B standards are surveyed, reporting on message exchange patterns, supported requirements and extensibility. Cloud computing is briefly discussed including benefits and issues relating to trust, regulation and operational requirements. The chapter finishes by surveying related work.

Chapter 3 generalises the results of the surveyed B2B standards to generalise common exchange patterns, business level requirements and abstractions. These generalisations are to define declaration mechanisms, specifying when supported properties should be satisfied. Hierarchies for *Fairness* and *Accountability* properties are defined and mapped to technical protocols providing their with different characteristics. The chapter finishes with discussions on supporting individual transmissions (as a building block for grander support) and use of the proposed declaration mechanisms.

Chapter 4 presents the generalised intermediary support middleware design, beginning with its positioning within the a B2B stack and a breakdown of its layers of abstraction. Various design decisions are discussed with regard to their effects on business level operation. Individual components are discussed including minimal components required to facilitate interception of messages and how functional properties are supported by composing additional components to execute technical protocols. Configurations for deployment will be discussed followed by a demonstration that previous work can be expressed as instances of the proposed generalised design.

Chapter 5 discusses three implementations representing instances of the design in Chapter 4. These first is a decentralised solution providing accountability where organisations retain control of their own cryptographic operations generation while still maintaining business level transparency[MC09]. The second implementation is a centralised solution providing accountability built using cloud computing as to minimise requirements on an organisation [MC10]. The third implementation is an adaptation of the second with extremely granular timing information. All implementations are evaluated on their functionality and the third is evaluated on its performance with regards to its impact on typical B2B interactions.

Chapter 6 concludes the work with a summary of contributions. An overview of contributions will be followed by a description of contribution by chapter. Contributions will be summarised and evaluated against the aims and objectives in this chapter. The chapter will finish with a conclusion and an overview of future work.

2 Background

This chapter begins with a discussion of B2B terminology in Section 2.1. Section 2.2 and 2.3 discuss layers of abstraction involved in B2B interactions and their associated components. Section 2.4 discusses relevant business and technical level assumptions.

Section 2.5 discusses relevant concerns to address when supporting B2B interactions including fairness, accountability and consistency. Section 2.6 discusses technical protocols suitable to solving the identified concerns and how they can be integrated into intermediary support while maintaining transparency.

Section 2.7 surveys current B2B standards, using ebXML as the general case and examining domain specific standards as subsets of ebXML. The survey reports on message exchange patterns, supported business requirements, how B2B concerns are addressed and how the B2B standards may be supported or have their own support extended.

Section 2.11 briefly discusses cloud computing with regards to its use for hosting components of intermediary support. The discussion considers benefits typically associated with cloud computing and where operational or compliancy issues may impact its use.

The chapter finishes with a survey of related work in Section 2.12 how it informs this thesis. Section 2.13 summarises key points discussed in the chapter.

2.1 B2B Terminology

Before discussing issues associated with supporting the execution and regulation of interactions, this section introduces some relevant terminology.

2.1.1 Organisational Identity

For the purposes of this thesis an *organisation* is an autonomous body, responsible for the business services they provision to enable their participation in B2B interactions. To facilitate participation in these interactions, organisations are represented by a sufficiently unique *identity,* this identity is authenticated and authorised within a given context (e.g., through the use of standards such as X.509 infrastructure, enabling the authoritative issuing of *identity certificates*) [SHF02]. In order to allow organisations to maintain their autonomy, the internal assignment of an identity within an organisation (i.e., whether it represents the entire organisation or some subset thereof) is considered out of scope, the authorised use of an identity representing any part of an organisation is their own responsibility.

2.1.2 B2B Terminology

B2B standards generally have their own terminology, these relate to general terminology laid out in this section including relevant relationships and assumptions:

Participant A participant is a single organisation (represented by an identity) interacting with another for the purposes of conducting business. This thesis assumes interactions to contain two participants, generally an initiator and a recipient.

Intermediary An intermediary is an entity through which interactions may be routed. The intermediary may take actions upon the intercepted information before passing in on for eventual delivery to its intended recipient.

Document A business document is a self contained document constructed by one participant to be sent to another. Example business documents include purchase orders or invoices.

Message Participants interact by sending business messages to each other. A business message contains one or more business documents and required information to ensure messages can be delivered to their intended recipient (i.e., the identity of the sender and the recipient).

Signal Business signals are used to indicate significant events, these signals indicate both positive and negative statuses including: the indication a participant is ready for some message to be sent, the acknowledgement of receipt of a previously sent business message or the acceptance or rejection of some business message's contents.

Transmission A transmission constitutes the transmission of some business contents (i.e., a message or signal) from one participant to another (e.g., $A \to B$). It is the smallest observable unit of communication between two participants.

Exchange An exchange between A and B involves an initial transmission of a message or signal from $A \to B$ and then a subsequent transmission of a response signal or message from $B \to A$. The response from B must explicitly reference the request from A.

Conversation A conversations is a sequence of one or more transmissions and exchanges (and their associated requirements), executed in order to achieve mutually beneficial business outcomes.

Interaction Two participants engaging in a transmission, exchange or conversation are interacting. Thus, interaction is used as a general term where the distinction is irrelevant.

Additional terms are introduced in subsequent sections. The above definitions provide a starting point for discussion related to B2B interactions. The survey of B2B standards in Section 2.7 will relate general terms to standards-specific counterparts where illustrative.

2.1.2.1 Conversations, Agreements and Contracts

Until this point, the terms conversation, agreement and contract have been used interchangeably without clarification. For the purposes of this work, where organisations conduct business through the exchange of messages, ‘conversation’ and ‘agreement’ may be used interchangeably depending on the context of the discussion. Where the focus of a discussion is on the need for organisations to agree, agreement will be the preferred term and where the focus is on the exchange of messages to facilitate business, conversation will be the preferred term.

Contracts are considered minimally to specify the conversations (or agreements) two organisations agree to execute, the execution of which be affected by legal requirements. This thesis assumes legal requirements affecting the business level definition and execution of a conversation are translated into specific business requirements (e.g., if a response is legally required within a certain time period, this can be translated to an appropriate deadline requirement) or are otherwise incorporated into the implementation of an organisation’s business services (e.g., laws affecting time periods for data retention). To avoid confusion, only the terms ‘conversation’ and ‘agreement’ will be subsequently used with the focuses described in this section.

2.2 Levels of Abstraction

Section 1.1 defined two key aspects to enabling (and supporting) the execution and regulation of B2B interactions. Specifically, high-level mechanisms allowing the definition of conversations specifying how participants must be observed to behave and low-level mechanisms enabling their realisation. That is, the satisfaction of requirements and properties enabling their execution and regulation by being able irrefutably prove that expected behaviour was observed, or indicate where expected behaviour was deviated from. These high and low levels can be more descriptively defined:

Business Level The business (high) level comprises the implementation of business services by an organisation and a specific set of B2B standards (e.g., ebXML) under which the services are designed. That is, an ebXML business service is implemented capable of supporting ebXML conversations through the exchange and processing of ebXML messages. This level deals in business abstractions (e.g., business documents, business messages and business conversations) enabling a business service to be constructed with details such as formats, protocols and transports abstracted by the lower levels.

Technical Level The technical (low) level comprises enabling technology allowing the business level standards and services to operate. This includes all formats, protocols and transports necessary to physically deliver the message from one participant to another, allowing them to process it at the business level and sanction an appropriate response. B2B standards may mandate the use of specific technical level technologies (e.g., RosettaNet mandates MIME message format and HTTP transport protocol) to satisfy their own business level requirements, but they still maintain the business and technical abstractions.

Figure 2.1 illustrates the layers in a generic B2B stack, indicating which layers belong to which level of abstraction. This layered stack is similar in approach to layered network stacks such as the OSI and TCI/IP models [Tan03].

The points in the following list correspond to the numbering on the layers shown in Figure 2.1:

1. A's business service is implemented to participate conversations defined by a B2B standard (e.g., RosettaNet), the chosen B2B standard dictates the contents and the formatting of the business messages to be exchanged, facilitating the execution of agreed upon conversations.

2. Messaging standards provide a general message format used to encapsulate all

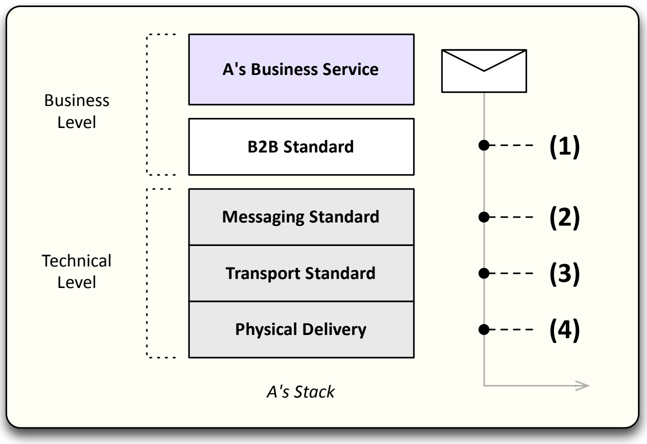


Figure 2.1: The components of *A*'s B2B software stack.

of the business level information ready for transport. For example, RosettaNet messages are encapsulated as MIME messages.

3. Messages ready for delivery can be transmitted across a number of different transports, matching specific requirements. For example, RosettaNet specifies that MIME messages be transmitted using HTTP or HTTPS where transport security is required.

4. With messaging standards and transports chosen, the physical delivery of a message to its recipient can occur, typically using TCP/IP or UDP.

Some Message oriented Middleware (MoM) standards specify their own wire-format [AMQ09]. That is, they define the messaging and transport layers (3 and 4) as shown in Figure 2.1.

It is important to note that while B2B standards may mandate technical level elements, these are kept abstracted from the business level elements. Where technical elements are mandated, it is for the reasons of satisfying specific requirements supported by the B2B standards. For example, RosettaNet allows a message to specify it requires secure transport, HTTPS is mandated as a technical requirement to guarantee a business level notion of transport security. Conversations defined in RosettaNet need not be aware that HTTPS is being used, they simply know their security requirement is being fulfilled. This mirrors the approach taken by support middleware designed in this thesis. That is, additional supported requirements are declared functionally, satisfied transparently by lower level elements to maintain the appropriate abstractions.

The survey in Section 2.7 will demonstrate that B2B standards represent concretisations of the general stack shown in Figure 2.1 where certain technical level aspects are specified to satisfy supported business level requirements (and the execution of conversations). It follows then, that two organisations interacting using the same B2B standard will both implement B2B stacks containing a minimally required group of the same components (i.e., they will both feature agreed upon elements of 1-4 as shown in Figure 2.1).

2.2.1 Message and Signal Types

Using these levels of abstraction, we define two message and signal types:

Business Message: A message whose contents and metadata drive operation at the business (high) level of abstraction. Business messages are exchanged and processed by business services.

Technical Message: A message whose contents drive operation at the technical (lower) level of abstraction, such as individual protocol messages. Technical messages are exchange and processed by technical level elements. Technical messages are associated with business messages (e.g., a technical message facilitating protocol execution related to a specific business transmission) and may contain parts of business messages, to be reassembled before being passed to the business layer.

Business Signal: A signal indicating an event at the business level of operation (e.g., the acknowledgement of receipt of a business message, the rejection of contents of a message or an indication of readiness to receive a business message).

Technical Signal: A signal indicating an event at the technical level of operation (e.g., the acknowledgement of receipt, the completion of a protocol or the indication of some exceptional circumstance).

All technical level elements are related to some business level elements. For example, a technical signal may refer to a technical message which is part of some protocol execution associated with the delivery of a business message or signal. The associations and their multiplicities are discussed in Section 4.6.1. This thesis will refer to messages where the distinction is irrelevant, otherwise technical- or business- prefixes will be used to indicate the applicable scope.

2.2.2 Conversation Outcome Types

The previous abstractions allow the definition of three outcomes for all B2B interactions:

Business Success: A business success (*BizSucc*) outcome indicates that all business interactions were successful and the desired business outcome was achieved.

Business Failure: A business failure (*BizFail*) indicates a problem at the business level of operation. For example, unexpected, malformed or erroneous business messages, rejection or failure due to deadlines or the inability to supply an order.

Technical Failure A technical failure (*TechFail*) indicates a problem at the technical level of operation. For example, unexpected, malformed or corrupt technical messages, technical incompatibilities or failures in communication, processing and storage.

Technical failures are difficult to deal with due to their potential impact on the business level, easily giving rise to business failures (e.g., a broken connection leading to a missed business deadline). These abstraction and outcome types are used in Section 2.5.5 and

4.6 when discussing the consistency of conversation outcomes and events whose effects cross layers of abstraction.

2.3 B2B Support Service Types

Steinauer et al. discussed the role of third parties in electronic commerce (E-Commerce) as *trust enhancers*, independent of and trusted by both participants in an interaction [SWR97], these trust enhancers were envisioned to provide additional guarantees or alleviate concerns regarding trusting interactions and their participants. They identify support of regulation through accountability and fairness as key concerns that can be addressed by the involvement of trusted third parties.

Subsequent work at Hewlett-Packard [BBMS01] went on to propose an ecosystem of security services to support E-Commerce. They envisaged the emergence of *security service providers* offering *security services* for a number of useful B2B roles including: identity, authorisation, guaranteed message delivery, notarisation, storage, audit, receipt generation and time-stamping. Desirable requirements for these services were identified including: ease of use to emphasise ease of integration and composition, survivability (long-lived availability and reliability), confidentiality and privacy including protection against insider threats at the service provider.

The constituents composed into Int_A, Int_B and "Intermediary Accountability Support" as discussed in Section 1.4 are examples of composed security services capable of providing support to B2B interactions.

The above allows the specification of different types of services involved in B2B interactions and their support. These types indicate domains of responsibility (separation of concern) and define specifically the type of service this thesis implements:

Business Services are constructed using B2B standards such as RosettaNet and ebXML to enabled the execution of agreed upon conversations allowing business to be conducted between organisations. These services create and process only business messages.

Security Services aid the execution and regulation of interactions conducted by business services. These security services may render their support transparently (as in this work) or be explicitly involved in them (e.g., the use of a payment processing service between two participants)[1].

Both of these service types require infrastructure to enable their operation, prompting the consideration of another service (and provider) type:

Infrastructure Services provide computation, storage and communication as a service. When infrastructure services are supplied by cloud providers, consumers pay for only the resources they consume (i.e., bandwidth, storage space and CPU cycles) and can be freed from provisioning their own hardware.

The definition of an ecosystem and security services allow functional properties to be satisfied by their sensible composition (e.g., time-stamping, receipt generation, fair exchange and evidence storage could be composed to provide a type of accountability). The design and implementations in Chapter 4 and 5 will make use of the composition of component security services when satisfying the functional properties it supports, to reduce redundant implementation of supporting features.

Each of the service types here has an associated service provider whose responsibility it is to satisfy the expertise and infrastructure requirements of operating their service. Business services are the responsibility of their respective organisation participating in B2B interactions, security services are the responsibility of the security service provider.

In this thesis, we assume the role of one or more security service providers, developing component security services and composing them to provide intermediary support

[1]In the course of its transparent operation, intermediary support may explicitly interact with security services. However, business level transparency will always be maintained to participants.

capable of supporting B2B interactions by transparently satisfying multiple functional properties. This thesis makes no assumption about the infrastructure services used by organisations or other security service providers to enable their business and security services, they may use cloud computing, private hosting or combinations thereof.

Where external infrastructure services are used by the security services developed in this thesis, they will be provisioned by cloud providers (e.g., Amazon Web Services), the use and impacts of which are discussed specifically in Section 2.11 and 4.11.

2.4 Assumptions

Before continuing with discussion on the concerns associated with B2B interactions, and technical protocols to address them, this section will state the assumptions upon which the work in this thesis is developed.

2.4.1 Business Level Assumptions

As stated in 2.1.1, organisations are assumed to use a sufficiently unique identity, represented by a mechanism such as X.509 infrastructure certification [SHF02]. Organisations are assumed to retain autonomy and be responsible for the internal assignment of such identities.

We assume all B2B interactions can be described in terms business level and technical level abstractions as discussed in Section 1.1 and 2.2 such that messages and signals may be classified as discussed in Section 2.2.1 and outcomes may be classified as discussed in Section 2.2.2.

Organisations are assumed to trust a critical subset of the intermediary support components acting on their behalf. This includes components responsible for generating and exchanging cryptographic data on behalf of an organisation and any other trusted third parties involved in the provision of support (e.g., a TTP ensuring fairness in the execution of a Coffey-Saidha non-repudiation protocol execution). This thesis assumes existing techniques for establishing or verifying trust, or mitigating risk, such as those discussed in [Sv10, BBMS01, JsIB07, SGR09] could be adapted to the work presented in this thesis.

For the mapping of functional properties (e.g., fairness and accountability) to technical protocols providing their satisfaction (e.g., non-repudiation), such as in Section 3.3.3 and 3.3.2, we assume formal modelling could be applied to verify the mappings (e.g., sufficient coverage of characteristics). Similarly, we assume that declarations using business level to infer functional properties (e.g., using ebXML non-repudiation requirement to infer accountability) can be verified or validated in the same way. This thesis develops middleware capable of realising provided mappings and declarations, applying methods for verification and validation of these is considered beyond scope.

2.4.2 Technical Level Assumptions

We assume two failure types regarding communication channels and nodes (i.e., all business services and intermediaries) [MJSC07]:

Permanent Failures: Communication channels do not heal and nodes do not recover their execution.

Temporary Failures: Communication channels will eventually heal and nodes will eventually recover their processing.

In the face of permanent failures, neither support or execution of B2B interactions can be guaranteed to complete, thus the following assumptions are made:

- The communication channels between well-behaved nodes provide eventual message delivery. That is, we assume channels are susceptible to temporary failures and there is a known bound on the number of temporary failures experienced by well-behaved parties.

- Well-behaved nodes have persistent storage for messages and ensure that messages are available for as long as is needed to fulfil obligations to activities in which they are engaged. That is, nodes are susceptible to temporary failures but may recover their execution.

- Well-behaved participants only send and process messages that comply with the specification of activities in which they are engaged (e.g., at the business level only messages complying with supported conversations are exchanged and processed and at the technical level all messages adhere to executing or supporting B2B interactions).

Trusted third parties involved in intermediary support are well-behaved by definition. There is no assumption (or restriction) regarding the number of misbehaving nodes.

By assuming temporary failures on communication channels, and the use of sufficiently unique message identifiers for all business and technical messages we can assume *at-least-once* delivery for messages and *at-most-once* processing for messages (on well-behaved nodes). Asynchronicity is assumed in the exchange and processing of messages such that channels and nodes do not need to be online at the same time but they are assumed to eventually be online.

For the modelling of protocol execution, I adopt the model formalised in [CD04]. The model is based on a modification of the Dolev-Yao model [DY83] in which intruders are *non-blocking*. Participants in protocol executions may misbehave (and collaborate) but eventually messages between well-behaved participants will be delivered. It is assumed intruders are capable of eavesdropping and replaying on any channel, the cryptographic primitives and capabilities described in Section 2.6.1 are used to secure transmissions across insecure channels where required.

2.5 Concerns when Supporting B2B Interactions

Section 2.1 through 2.3 defined concepts, classifications and assumptions relevant to the work undertaken. This section motivates supporting the execution of B2B interactions, discussing trust, fairness, accountability and consistency concerns.

2.5.1 Trust Concerns

Section 1.4 introduced the notion of classifying the level of trust between interacting organisations. This has an impact on what guarantees must be enforced when they interact. To this end we use three classifications of trust as described in [FR97]:

Trusted parties are assumed to behave at all times and will be honest about their responsibilities when compliance is broken (e.g., if they miss a deadline, they will admit to doing so)

Semi-trusted parties are assumed to be able to misbehave individually, but will not conspire with others to do so.

Untrusted parties are assumed to be able to misbehave individually and in collaboration with others.

From intermediary support's point of view, semi-trusted and untrusted parties are classified as untrusted. That is, fairness guarantees should only be relaxed when complete trust is present. Interacting organisations may be classified as semi-trusted in that they may privilege their own interests ahead of others, as described in Section 1.2.

This thesis does not require or assume any specific trust relationship between organisations, characteristics will be defined for fairness and accountability properties allowing the relationship between two organisations to be expressed.

Importantly, we assume that trust relationships change over time. The intermediary support will take this into account and use the trust relationship at the time of transmission when choosing the best protocol to satisfy the required properties. Section 3.3.1

and 3.4 discuss how intermediary support characterises the trust relationship between participants, affecting the satisfaction of fairness and accountability properties.

It is required that an organisation trusts its own intermediary and TTPs engaged by its intermediary. For example, A trusts intermediaries acting on its behalf and all TTPs engaged with to provide extra guarantees (e.g., deterministic fairness). Components requiring trust will be kept to a minimum, discussed in Section 4.9.

2.5.2 Fairness Concerns

We begin with the following definition:

Fairness: For a fixed quality communication channel, at the end of an exchange protocol, either all involved parties obtain their expected items or none, even a part, of the information to be exchanged is revealed. [MGK02].

Informally, we are expressing a desire that no participant is placed at a disadvantage as a result of having co-operated with agreements in place. That is, there is no benefit to deviating from the expected behaviour for participants who choose to do so.

Various work on fairness has defined some important characteristics regarding how it is guaranteed [MGK02, Aso98, FR97, PVG03]:

Strong: The requirement that fairness is never violated during an exchange

Weak: Allows fairness to be violated during an exchange with assurances that it can be re-established

Optimistic: The guarantee that TTPs will not become involved in an exchange unless absolutely required

Transparent: The guarantee that where TTPs are involved, their involvement cannot be detected by the participants

Weak fairness is intended for situations in which the overhead of providing strong fairness is deemed unsuitable. For example, if the value of an interaction were sufficiently low or the cost of support too high.

The most desirable combination of these characteristics, for supporting fairness in B2B interactions is strong, optimistic and transparent. That is, TTPs will only become involved when absolutely necessary to maintain strong fairness and their intervention remains undetectable to the participants. Transparency in TTP involvement protects against situations in which detecting the involvement of a TTP (e.g., in generated evidence) may harm the reputation of participants.

Section 3.3.2 discusses a mapping of Fairness as a property to a number of protocols providing its satisfaction under the characteristics discussed in this section. As an aside, various work has demonstrated that a TTP is required to guarantee deterministic fairness in an exchange [PG99, EY80, Sch00].

If fairness is not guaranteed for an exchange, either participant can gain an unfair advantage over the other, Figure 2.2 illustrates two examples, the first in which A offers proof of origin (PoO) alongside the message it delivers and B is supposed to respond with proof of receipt (PoR). In the second conversation B offers PoR as a show of good faith before A is supposed to divulge PoO.

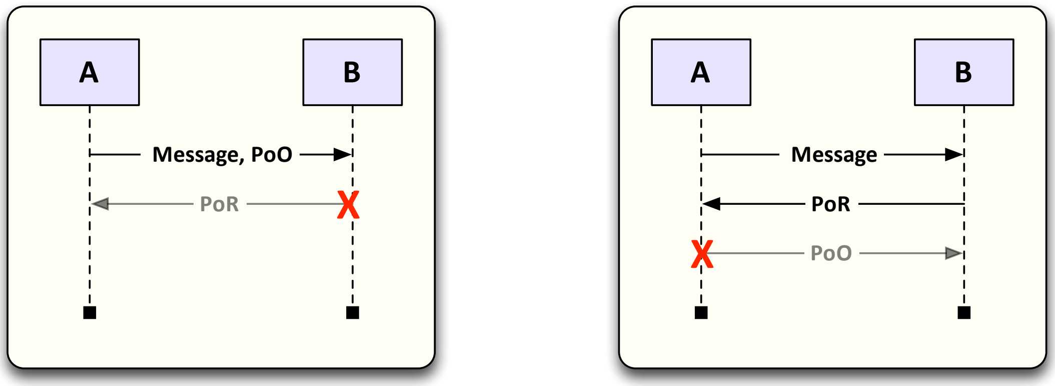


Figure 2.2: Two flows in which a participant gains an unfair advantage.

In the first exchange, B gains an unfair advantage in that it can choose to withhold proof of receipt from A and A has no irrefutable recourse. Similarly, B is placed at a disadvantage in the second exchange by A's decision to withhold proof of origin from B. These examples describe the *selective receipt* problem [Aso98]. Fairness will be used in this thesis to protect participants during their interactions, that is, they are able to interact under the assumption that their interests are protected in the face of misbehaviour by other participants.

2.5.3 Accountability Concerns

Accountability, as defined by [YC04], denotes "assurances of semantic behaviour that extend beyond basic perimeter security and the mechanisms for message authentication, encryption, and integrity". That is, the behaviour, state, and actions of accountable systems should be:

Undeniable: Actions of an accountable actor are provable and non-repudiable. That is, a service or its clients cannot plausibly deny their actions, and those actions may be legally binding.

Certifiable: A client, peer, or external auditor may verify that an accountable service is behaving correctly, and prove any misbehaviour to an arbitrary third party. For example, a service may be prompted to prove cryptographically that its actions are justified by the sequence of operations issued by its clients, in accordance with its defined semantics.

Tamper-evident: Any attempt to corrupt the service state incurs a high probability of detection. In particular, an external auditor may determine if the internal state could or could not result from the sequence of operations issued on the service.

Thus, accountability can be satisfied by the generation, exchange and storage of evidence satisfying these properties for all B2B interactions. We begin by defining two kinds of evidence as discussed in [CS96, ZG96b, ZG97b, KMZ02]:

Proof of Origin and Contents binding a message's origin and contents to a participant (the sender)

Proof of Receipt binding a message's receipt to a participant (the recipient)

Additional kinds of evidence may be generated in the creation of an audit trail used to irrefutably demonstrate accountability [Zho01, Coo06], including:

Proof of Submission evidence that a message was submitted to some delivery agent for onward delivery to its recipient, usually generated by some delivery intermediary or TTP.

Proof of Delivery evidence that a message was ultimately delivered to its intended recipient, again usually generated by some delivery intermediary or TTP.

These evidence types bind the associated action to an identity such as those discussed in Section 2.1.1. Such evidence allows participants and/or auditors (e.g., TTPs) to render transmissions *undeniable*, *certifiable* and *tamper-evident* as previously described. Evidence satisfying these properties allows the irrefutable demonstration of compliance with agreements (i.e., conversations) in place, without it, agreements become unenforceable.

There is generally an agreed upon period of time for which accountability evidence should be stored, usually some number of years [OAS07b, Ros02]. The potential longevity of this evidence highlights an issue regarding its validity as generation of these evidence types relies on the use of cryptographic operations. Cryptographic keys used to generate this evidence may be revoked at any time, rendering evidence unusable if its associated keys are no longer valid.

The issue becomes demonstrating that evidence was valid at the time of generation (or some witnessed time before key revocation). Without such guarantees, a system is vulnerable to *key revocation* [Coo06].

Section 2.6.2 discusses time-stamping authorities, designed to irrefutably witness information at a given time, ensuring its validity against subsequent key revocations.

The specific protocols used to achieve accountability , discussed in Section 2.6.3, may interact with TTPs and TSAs to ensure that generated evidence is protected against selective receipt and key revocation.

2.5.4 Accountability and Fairness: Why Both?

A valid question may be why guarantee accountability for exchanges where strong fairness can be guaranteed. The critical issues are the following:

1. An exchange is not the smallest unit of B2B interaction that can occur
2. Strong fairness only protects participants during an exchange, it is not sufficient to ensure agreements remain irrefutably enforceable

Section 2.1 defined incremental building blocks for composing conversations: transmissions, exchanges and conversations. A conversation is a sequence of transmissions (which have no response) and exchanges (request transmissions with corresponding response transmissions). The smallest unit of business communication we can assume between two participants is a single transmission.

The issue here being that fairness as defined in Section 2.5.2 is only applicable to exchanges. That is, there must be at least one piece of information, per participant, to be exchanged to be able to guarantee fairness by the definition that everyone is guaranteed their desired information, or nobody is.

If we consider that accountability is satisfied by the exchange of evidence, irrefutably binding participants to the actions they take, a business level transmission can be supported at the technical level as an exchange of a message and its associated proof of origin for proof of its receipt. In doing so, business level transparency is maintained for the smallest unit of business interaction (i.e., a transmission) while allowing fairness and accountability to be provided. Section 3.5 will discuss this in details.

The second point above stated that strong fairness only protects during an exchange and is not sufficient to ensure agreements remain irrefutably enforceable. What we mean here is that, without accountability evidence to irrefutably bind participants to their actions, it cannot be demonstrated whether a participant complied with agreements in place for their interactions. Thus, accountability is required to protect participants after interactions while fairness protects them during an exchange by ensuring that messages, signals and associated evidence aren't released prematurely, granting anyone an unfair advantage.

2.5.5 Consistency Concerns

The B2B interactions considered in this thesis are executed via the asynchronous exchange and processing of messages. This asynchronicity can lead to unpredictable delays in processing and communications and lead to situations in which participants have conflicting views of state. Figure 2.3 illustrates a conversation in which A and B terminate with conflicting views of the outcome.

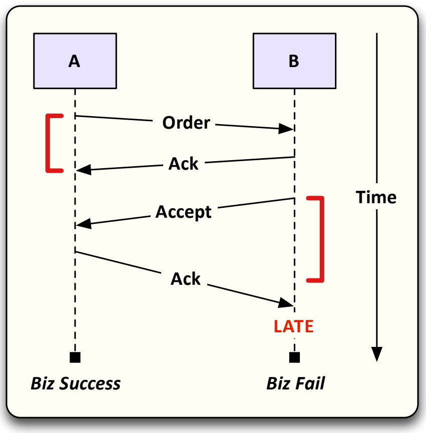


Figure 2.3: A and B terminate in different states.

The problem demonstrated in Figure 2.3 is an instance of the *last-ack* (or *two gen-*

erals) problem [Tan03, MJSC07]. The red brackets represent deadlines by which an acknowledgement is expected to be received for some previously sent message. The final acknowledgement from A to B is seen to arrive beyond the end of the second deadline, meaning A believes the conversation terminated successfully (a business success in terms of the outcomes discussed in Section 2.2) whereas B treats the late arrival as a failure to meet a business deadline and assumes the outcome is a business failure.

Based on the above, we define two levels of consistency for use in this work:

Total Consistency: The requirement that all participants are prevented from entering conflicting states regarding ongoing interactions.

Outcome Consistency: The requirement that inconsistencies may arise during interactions but the outcome is consistent for all participants.

Similarly to strong and weak fairness as discussed in Section 2.5.2, total consistency would require additional complexity over outcome consistency. It may be permissible to relax consistency requirements to apply only to outcomes, or even further depending on factors such as the perceived value of an interaction.

B2B standards acknowledge that inconsistency may occur and a corrective approach to the issue. Participants are able to notify others of failures regarding previous interactions. The invocation of such mechanisms depends upon the eventual detection of the inconsistency and will most likely result in corrective action being taken, potentially at a cost of computation, reputation or money. For the above reasons, consistency is desirable such that inconsistencies be prevented from manifesting at the business level.

Consistency is discussed as a proof of concept that properties beyond fairness and accountability can be satisfied by the intermediary middleware designed in this thesis. Section 2.6.4 and 3.3.4 will discuss how this could be achieved.

2.6 Technical Protocols

The issues discussed in Section 2.5 are not without technical solutions. However, these solutions incur expertise, infrastructure and integration requirements as discussed in Section 1.5. This section discusses technical protocols and techniques that will be used to address the concerns of fairness, accountability and consistency. Protocols providing fairness and accountability will be mapped with functional properties in Chapter 3, specifying how the properties (with varying characteristics) are satisfied.

2.6.1 Primitives and Capabilities

For the protocols discussed in the remainder of this thesis, here we define supporting notation, primitives and capabilities including secure hash functions, digital signature schemes, secure pseudo-random sequence generators [Sch96, Gol99].

Secure hash functions have the following properties:

Ease of Computation: Given x, it is easy to compute $hash\,(x)$

Compression: Given an arbitrary string, the function produces a fixed length string as output (a.k.a a message digest)

Preimage Resistance: Given h, it is computationally infeasible to find x such that $h = hash\,(x)$

Second Preimage Resistance: Given x and $hash\,(x)$ it is computationally infeasible to find $y \neq x$ such that $hash\,(x) = hash\,(y)$

Collision Resistance: If $hash\,(x) = hash\,(y)$, then $x = y$ with an effective probability of 1

Public key cryptography pairs together public and private keys to be used for asymmetric cryptographic operations. These *key pairs* have the following properties:

- It is computationally infeasible to compute a private key from its corresponding public key
- Cipher-text generated using a public key can only be decrypted using the corresponding private key
- Cipher-text generated using a private key can only be decrypted using the corresponding public key

The final of these properties supports *digital signature schemes* comprised of two algorithms:

1. The signing algorithm where some data is encrypted using a private key to generate a verifiable signature and
2. The verification algorithm which uses the corresponding public key to decrypt the received signature

Verification allows a participant to confirm that only the signing participant's private key could have been used to generate the signature.

Secure pseudo-random sequence generators generate sequences of bits with the following properties:

Pseudo-randomness: The generated sequence is statistically random

Unpredictability: Given complete algorithmic, hardware and previous generation knowledge it is computationally infeasible to predict the next bit of the sequence

Following these properties, we can specify the following notation used by protocols referenced in this thesis, shown in table 2.1.

For the involvement of public key cryptography we assume the existence of some certificate authority (*CA*) and some public key infrastructure (*PKI*) allowing identity certificates, key-pairs and public keys to be issued, validated, revoked and distributed as discussed in this section and in Section 2.1.1.

Notation	Description
$h\left(x\right)$	A secure hash of x
$sig_P\left(x\right)$	Participant P's signature over x using P's private key
$enc_P\left(x\right)$	Encryption of x with P's public key
$rn_{[P]}$	A secure pseudo-random number with optional identity of participant P who generated rn
T_g	Time of generation of some information
$P \rightarrow Q : x$	The transmission of x from P to Q
x, y, z	The concatenation of x, y and z.

Table 2.1: Cryptography and Communication Notation

2.6.2 Time-stamping: Time-stamping Authorities

Section 2.5.3 discussed the notion of key revocations subsequent to the generation of evidence rendering them invalid. For this reason we consider the involvement of a specialised kind of TTP known as a Time-stamping Authority (*TSA*) . *TSA*s serve to act as a witness to information presented to them at some time T. Specifically for accountability, *TSA*s are used to witness the time of generation (T_g) of accountability evidence [ZG97b]. Revocations will be executed authoritatively by *CA*s and thus, all evidence witnessed before the time of revocation can be irrefutably demonstrated as valid. Table 2.2 describes and defines notation for interacting with *TSA*s to witness information.

Interaction with *TSA*s may be in-line (all interactions between nodes passes through the *TSA*) or on-line (nodes consult the *TSA* but not all messages are required to pass through it), illustrated in Figure 2.4.

The existence of *TSA*s and the consistency issues discussed in Section 2.6.4 assume there is a global time base from which clocks are synchronised to a reasonable (known and bounded) accuracy.

Notation	Description
$ts_{TSA}(x)$	The time-stamp on x generated by TSA to witness the generation, T_g, of x

Notation	Definition
$ts_{TSA}(x)$	$\{T_g, sig_{TSA}(x, T_g)\}$

Table 2.2: Time-stamping Authority notation

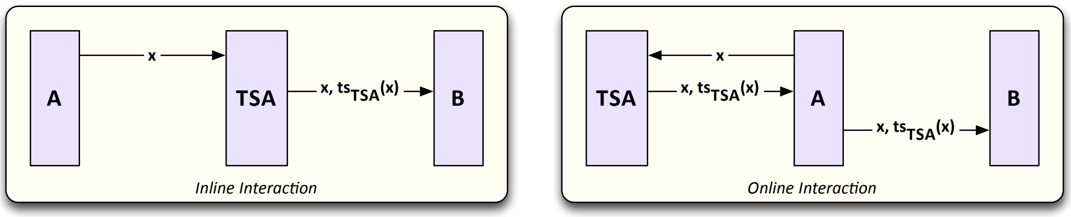


Figure 2.4: Styles of interaction with a *TSA*.

2.6.3 Accountability: Non-repudiation Protocols

The accountability example in Section 1.3 introduced the notion of using non-repudiation protocols for the satisfaction of accountability as a property. Non-repudiation is defined to be the inability to subsequently deny an action or event. ISO 7489-2 identified non-repudiation as a primary security concern alongside four others (authentication, access control, confidentiality and data integrity) [ISO89]. Kremer et al. conducted an intensive survey of non-repudiation protocols and classified them in terms of their requirements and the properties they provide [KMZ02].

For each of the evidences detailed in Section 2.5.3 (proof of origin, receipt, submission and delivery) the non-repudiation specific counterparts are similarly named non-repudiation of origin (*NRO*), non-repudiation of receipt (*NRR*), non-repudiation of submission (*NRS*) and non-repudiation of delivery (*NRD*).

They classify the following fairness properties:

Probabilistic at the end of an exchange there is some probability, p, that a message and its *NRO* were fairly exchanged for its *NRR*

Strong Fairness states that at the end of an exchange a message and its *NRO* are guaranteed to have been fairly exchanged for its *NRR* or no valuable information has been obtained by either participant. A *TTP* must be available to deterministically guarantee fairness [PG99] but they may not be required to intervene in all cases

True Fairness states that strong fairness must be provided and, at the end of an exchange, it is indistinguishable from the evidence and execution whether the *TTP* intervened or not

Alongside there fairness properties, they classify *TTP*s as having three modes of interaction for the execution of non-repudiation protocols:

Inline in which all communication between nodes engaged pass through the *TTP*

Online in which a *TTP* is involved in every execution of a protocol but all communication between nodes does not pass through it

Offline in which a *TTP* is only engaged when necessary to guarantee a protocol terminates and fairness is guaranteed

Transparent in which is it indistinguishable when an *offline* *TTP* becomes involved in a protocol execution from when it does not

Offline and transparent *TTP*s are synonymous with optimistic and transparent modes of operation from the fairness discussion in Section 2.5.2.

Communication channels are classified as:

Unreliable channels are able to permanently fail or entirely lose messages

Resilient channels may temporarily fail, but will eventually heal and deliver messages with an unknown but finite delay

Operational channels guarantee delivery with a known delay

Table 2.3 selects protocols for different modes of *TTP* interaction from [KMZ02]. For multiple protocols satisfying the same kind of TTP interaction, the protocol chosen is the one guaranteeing timeliness alongside having the lowest communication channel requirements (i.e., unreliable < resilient < operational).

Protocol	**Fairness**	**TTP**	**Channel**
Coffey-Saidha[CS96]	Strong	Inline	Resilient
Zhou-Gollman [ZG96a]	Strong	Online	Resilient
Kremer-Markowitch [KM00]	Strong	Offline	Resilient
Markowitch-Kremer [MK01]	True	Transparent	Resilient
Mitsianis [Mit01]	Probabilistic	None	Unreliable

Table 2.3: Non-repudiation protocols, their characteristics and channel requirements

Voluntary exchange is omitted from this table as the survey cited details only exchange protocols that guarantee some level of fairness (even probabilistic). Voluntary exchange has no channel requirements (i.e., unreliable), involves no *TTP* interaction and guarantees no fairness.

Each of these protocols is used to provide accountability with different characteristics, Section 3.3.3 will define the characteristics and provide a complete mapping of accountability to all available protocols able to satisfy it. As previously discussed, voluntary exchange is only suitable when a participant has complete trust in the recipient. In all other circumstances one of the protocols in table 2.3 providing deterministic fairness guarantees will be used.

In terms of the best protocol to use to guarantee fairness where required, the Markowitch-Kremer protocol is the theoretical best, providing strong fairness through an offline *TTP* acting transparently. Transparency is cited as an important concern in the execution of non-repudiation protocols and the generation of their evidence as being able to discern when a *TTP* had to intervene versus when it did not may have an adverse affect on the reputation of participants [KMZ02, VPG99].

In reality, complexity trade-offs such as performance, message and computational may make different fairness guaranteeing protocols suitable under different conditions. Full protocol definitions are omitted from this section for brevity although Chapter 5 will detail non-repudiation protocols implemented to provide accountability. Importantly, the middleware design in Chapter 4, and constructs presented in Chapter 3 provide a framework in which any non-repudiation protocol can be implemented and executed to provide accountability while still maintaining business level transparency.

The intermediary accountability support example showed only two intermediaries, acting on behalf of their respective organisations, communicating with each other to execute non-repudiation protocols to provide accountability. In reality, TTPs (and other security services such as TSAs and CAs) will be engaged with during the execution of technical protocols. That is, the composition of intermediaries acting on behalf of organisations and security services is used to satisfy some support property as an abstraction like accountability. Figure 2.5 illustrates an example TTP being possibly involved in the execution of a non-repudiation protocol.

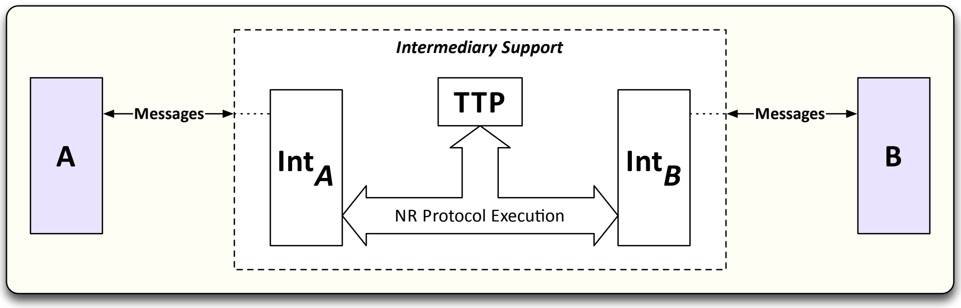


Figure 2.5: Non-repudiation execution possibly engages a *TTP* to guarantee fairness.

2.6.3.1 Protecting Against Key Revocation

*TSA*s offer protection against key revocation in non-repudiation protocols regardless of the type of interaction with the *TTP*, the survey of non-repudiation protocols discusses alternative schemes that either are not applicable to all modes of *TTP* interaction or add complexity that does not contribute to the aims of this thesis. That is, the ability within the intermediary support to safeguard evidence from key revocations is enough. As such, where required unless otherwise specified, *TSA*s as described in Section 2.6.2 will be engaged to witness the generation of evidence and protect it against subsequent key revocation.

2.6.4 Consistency: Synchronisation Protocols

Section 2.5.5 demonstrated that participants in B2B interactions may obtain inconsistent view of state. Synchronisation can be used as a preventative approach in which inconsistent views are prevented from manifesting at the business level. Using the outcomes defined in Section 2.2.2, any interaction can results in a business success, a business failure or a technical failure. These outcomes are assigned the following precedence:

$$\mathit{TechFail} > \mathit{BizFail} > \mathit{BizSucc}$$

That is, failures take precedence over a success and technical failures are more severe than business failures. This allows a framework to be established for synchronising the outcomes of B2B interactions [MJSC07]. By performing a three-way handshake, two participants can obtain a consistent view on the state of an interaction (using the above precedence).

In doing so, inconsistent views can be prevented from manifesting at the business level. *BizSucc* outcomes are never propagated to the business level where any participant is in a failure state. Similarly, a *BizFail* outcome is never propagated to the business level if any participant is in the *TechFail* state.

Molina et al. augmented existing RosettaNet interactions to provide synchronisation of outcomes while also maintaining business level autonomy and transparency [MJSC07]. This is done through the implementation of a conversation management layer (CML) at lower levels, each participant is equipped with a CML that acts on their behalf. The CML ascertains its organisation's view on the outcome of an interaction and performs a three-way handshake with other CMLs to ensure that a consistent outcome is propagated. Figure 2.6 illustrates an example B2B interaction with the additional synchronisation steps taken at the end of the interaction.

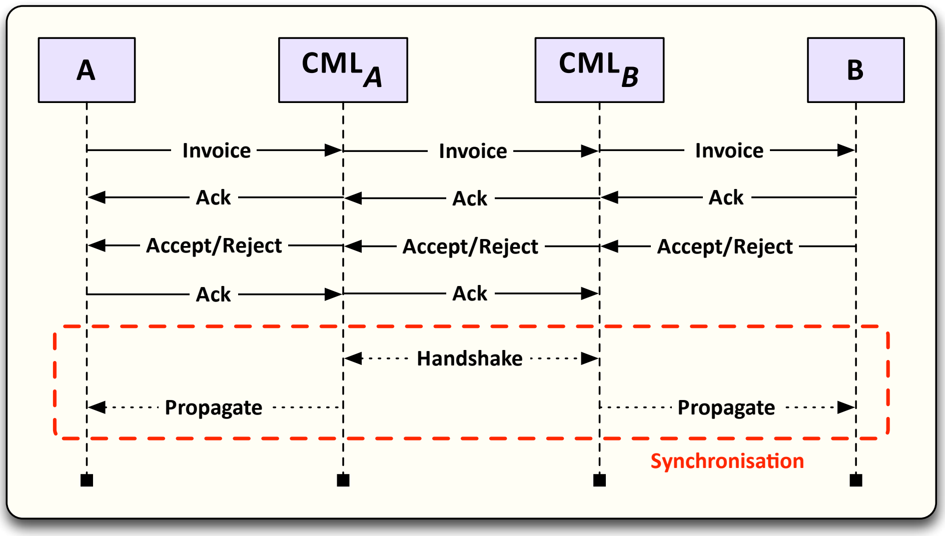


Figure 2.6: Synchronisation of outcomes by CMLs on behalf of A and B

Molina et al. use the outcome of a conversation as the chosen point of synchronisation. By choosing other points of synchronisation, it may be possible to obtain a stronger level of consistency guarantee such as total consistency as discussed in Section 2.5.5.

This approach to synchronisation is an instance of consensus within a distributed asynchronous system, the desire is to reach consensus in a finite time in the face of temporary failures. However, it must be considered that if nodes are prone to failure, reaching consensus in finite time in an asynchronous system can be impossible [FLP85].

Even assuming nodes never fail but communications may temporarily fail, it may still be impossible to establish consensus in finite time (two-army/last-ack problem)[Tan03]. There is a small risk that, by assuming temporary failures, an agreed outcome cannot be reached, this risk can be arbitrarily reduced by increasing the timeouts used for all communications [MJS06].

This thesis will not implement support for the consistency property. However, it will be considered as proof that additional capabilities could be implemented such as consistency support through synchronisation/consensus. Section 3.3.4 will discuss a possible example consistency property and some of the complexities of supporting it while ensuring support remains decoupled from specific B2B standards.

2.7 Survey of B2B Standards

While the proposed support offered to B2B interactions in this thesis will be done so independent of specific B2B standards, this following sections will survey existing B2B standards in order to better generalise message exchange patterns, supported requirements and technical level elements discussed later in the thesis. The survey will ensure the support offered has a real world grounding and, with the rest of the work presented in this chapter will be used to address the challenges identified in Section 1.5.

ebXML is surveyed as the general case for defining B2B interactions and their requirements, RosettaNet and Open Travel Alliance will be surveyed as a subset of what is expressible using ebXML and where they differ (or refine) ebXML specifications for their own purposes.

The survey will discuss the following areas:

- A general overview discussing the aims and target audience of standard and relating domain specific terminology to the general definitions in Section 2.1.
- An illustration of the standard's B2B stack as a concretisation of the general B2B

stack in Section 2.2, indicating any required technical level elements.

- What messages exchange patterns are used or supported in the construction of conversations?
- What support is offered for the following requirements and how well are the requirements satisfied?:
  - Security for messages and their delivery.
  - Reliability for the reliable delivery of messages to their recipients.
  - Accountability to ensure participants are bound to their actions.
  - Fairness guarantees for all participants.
- How are the supported requirements specified for messages and conversations?
- Can business level requirement support be improved and if not, how can intermediary middleware better support the standard.

The survey will finish with a summary, demonstrating the suitability for a lower level approach.

2.7.1 Surveyed Documents

The surveyed ebXML documents are:

1. ebXML Messaging Services 3.0 [OAS07b]
2. ebXML Business Process Specification Schema [OAS01a]
3. ebXML Messaging Services 3.0 Conformance Profiles [OAS07a]
4. ebXML Collaboration Protocol Profile and Agreement Specification [OAS02]

The surveyed RosettaNet documents are:

1. RosettaNet Implementation Framework 2.0 [Ros02]

2. RosettaNet Partner Interface Process [Ros09]

The surveyed Open Travel Alliance documents is:

1. OpenTravel Schema 2011B [Ope11b].

2.8 Survey: ebXML

ebXML comprises a set of standards designed to allow the definition of arbitrary conversations and the processing of messages facilitating their execution. It makes recommendations for how some business level requirements may be satisfied leaves imperative details as a matter of conformance profiles [OAS07a]. Standards such as RosettaNet and Open Travel Alliance can be represented as an agreed upon set of ebXML conversations and a conformance profile dictating how conversations are executed and supported, discussed in Section 2.9.

Conformance profiles specify details relating to handling of business messages including transports to use (e.g., HTTP), general messaging versions (e.g., SOAP 1.x) and how requirements are supported (e.g., WS-Reliability for reliability, HTTPS for transport security or algorithms for generating non-repudiation evidence). The separation of functional and imperative concerns allows ebXML to be adapted to new situations are required. Section 2.8.8 briefly discusses conformance profiles.

Collaboration profiles allow organisations to provide a profile of all of the ebXML capabilities they support (e.g., conformance profiles). A collaboration profile agreement is an agreed upon intersection of two collaboration profile documents, allowing organisations to agree on what capabilities conversations they define can make use of. These specified mechanisms allow organisations to quickly agree upon a on a common base of understanding and begin developing conversations [OAS02].

2.8.1 Terminology and Stack

ebXML's terminology is similar to that introduced in Section 2.1, participants in may be referred to as *partners,* business messages may also be called *ebXML messages* and conversations are called *business processes*, still composed of sequences of exchanges. Figure 2.7 relates the ebXML stack to the general B2B stack in Figure 2.1, the subsequent list points correspond to their element in the figure.

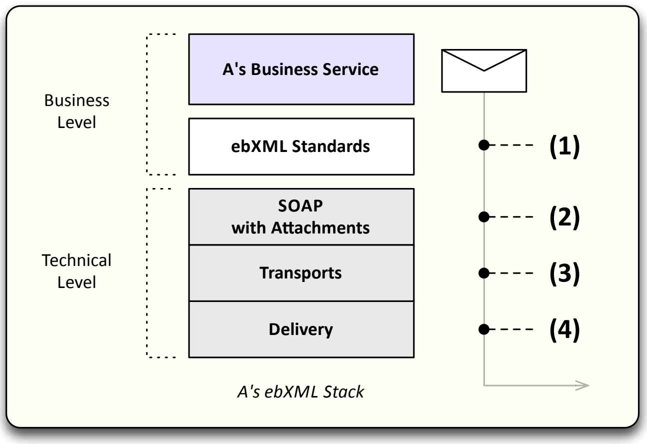


Figure 2.7: ebXML's concretised B2B stack.

1. The *ebXML Messaging Service* and *Business Process Specification Schema* standards provide functional definitions for conversations and their requirements. *Conformance and collaboration profiles* allow organisations to agree upon how conversations are executed and requirements supported.

2. All ebXML messages are encapsulated as SOAP messages (specifically SOAP with Attachments [Wor00]). Conformance profiles dictate the version of SOAP (1.1 or 1.2 recommended). Their processing supports the use of any standards and techniques applicable to SOAP messages, pending agreement with other participants through collaboration profiles.

3. The *ebXML Messaging Service* standard abstracts away specific transport stand-

ards and categorises all transports as *one-way* (e.g., SMTP) or *two-way* (e.g., HTTP). The use of HTTP, SMTP, FTP and even IIOP are discussed but ultimately a result of conformance and collaboration agreements.

4. As with the transport layer, ebXML abstracts away specification of physical delivery as a matter of conformance and collaboration profile, or possibly a natural consequence of the transports in use (e.g., HTTP over TCP/IP). TCP/IP and UDP are highlighted as the most likely choices [OAS07b].

The *ebXML Standards* layer in Figure 2.7 can be further decomposed to illustrate how the *ebXML Messaging Service* standard abstracts away messaging details, shown in Figure 2.8 with subsequent discussion:

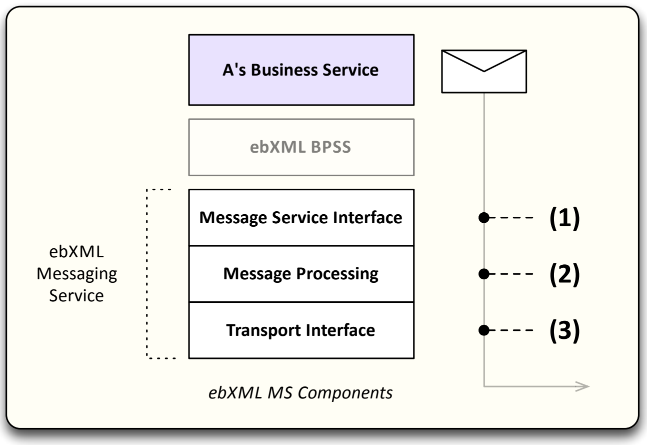


Figure 2.8: Layers of the ebXML Messaging Service.

1. The *Messaging Service Interface* marks the point at which a full formed ebXML business message is passed from a business service for processing and delivery.

2. *Message Processing* involves extra processing required by the message, this may involve the generation of signatures and receipts, encryption of contents or additional SOAP-style operations on the ebXML message supported by WS-* standards.

3. The *Transport Interface* provides a transport independent mechanism for ebXML messages to be dispatched for delivery, a binding from the transport interface onto specific standards such as HTTP, SMTP or FTP enables their use by ebXML messaging.

Elements (3) and (4) of Figure 2.7, and elements (1), (2) and (3) of Figure 2.8 are specified by conformance and collaboration agreements between organisations. The *ebXML Messaging Service* and *ebXML Business Process Specification Schema* abstract away the imperative details of conversation execution and requirement support from their specification.

2.8.2 Message Exchange Patterns

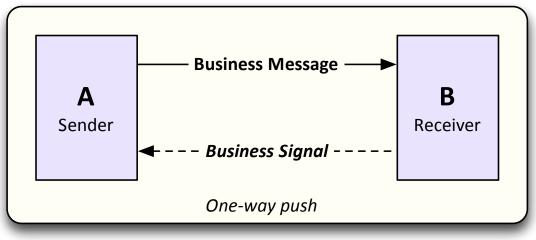
Conversations specified using ebXML are created by interacting organisations to suit their specific needs. Their specification may be informed by documents such as the ebXML catalog of common business processes, discussed later, but ultimately are the responsibility of the interacting organisations. As with the general definitions in Section 2.1.2, conversations are defined as sequences of exchanges. The constituent exchange patterns in ebXML consider two aspects:

Response required? Is any kind of response correlated to the original request required?

Push, Pull or Synchronous? Is the exchange categorised as push, pull or synchronous?

Based on these two aspects, ebXML considers the following exchange patterns: one-way *push*, one-way *pull*, two-way *push-then-push*, two-way *push-then-pull,* two-way *pull-then-push* and two-way *synchronous.* Two-way patterns can be considered in terms of the one-way patterns:

For one-way *push* exchanges, some *sender* transmits a business message to a *receiver* and the receiver responds with some kind of signal (e.g., acknowledgement of receipt, proof of receipt or an exception).



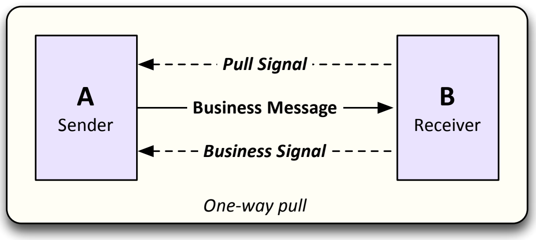


Figure 2.9: One-way push and one-way pull exchange patterns.

For a one-way *pull* exchange, the *receiver* makes aware the *sender* that it is ready to receive a business message. The *sender* then transmits its business message to the *receiver* who responds with a signal as in the one-way *push* exchange. Importantly, ebXML requires that the *pull signal* and *business* message occur over the **same connection**. That is, a two-way connection (e.g., HTTP), as discussed in Section 2.8.1 is required.

The supported two-way exchanges are compositions of the one-way patterns (i.e., *push*-then-*push*, *push*-then-*pull* and *pull*-then-*push*). ebXML distinguishes two-way exchanges by specifying that the second business message must make explicit reference to the first (using its message identifier).

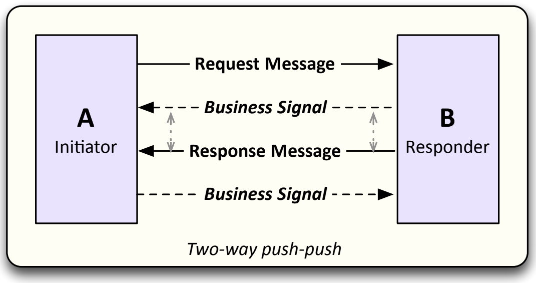


Figure 2.10: Two-way push-push exchange pattern.

Figure 2.10 illustrates a two-way *push-then-push* exchange, a composition of two one-way *push* operations, the *response message* from the *responder* must contain an explicit reference to the *request message.* The figure also indicates an association between the first business signal (indicating the *responder's* reaction to the *request* message) and the *response* message. ebXML states that the signal from the *responder* to *initiator* **may be** piggy-backed on the *response* message, conformance and collaboration profiles can strictly prohibit this behaviour if desired.

This optimisation may also be applied to the two-way *pull-then-push* pattern shown in Figure 2.11 but not the *push-then-pull* pattern. The intervening *pull signal* from *initiator* to *responder* prevents the business signal being piggy-backed onto the *response* message.

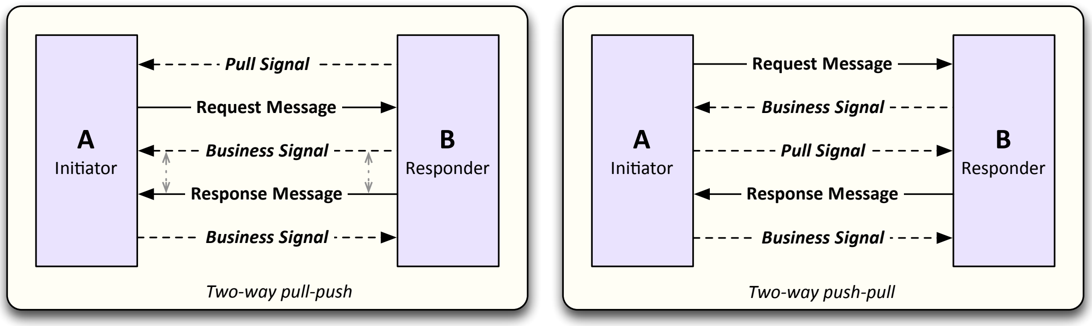


Figure 2.11: Two-way pull-push (left) and push-pull (right) exchange patterns.

The two-way synchronous exchange pattern is a synchronous execution of the two-way *push-then-push* exchange, requiring the *initiator* to transmit its *request* message and block its execution until the *responder* returns a *response message* and *signal* piggy-backed together.

2.8.3 Security

The ebXML messaging service defines the format for ebXML business messages as an extension to the SOAP with Attachments specification. As such, all applicable SOAP security standards can be applied to ebXML messages. Additionally, all information within an ebXML message, except that required to facilitate delivery to the intended recipient, may be encrypted.

Messages are allowed to specify binary parameters requiring content security and transport security. Characteristic of ebXML's approach, how this is achieved is left as a matter of conformance and collaboration profiles. Standards such XML Encryption, XML Signatures and HTTPS are discussed as suitable examples [Wor02, Wor08].

2.8.4 Reliability and Timeliness

ebXML discusses multiple methods for achieving reliable delivery including an ebXML Reliable Messaging protocol and existing technologies designed to reliably deliver SOAP messages. A message can specify that is requires reliable delivery alongside four parameters regarding the delivery and processing:

Time to Acknowledge Receipt: A deadline within which receipt of the message must be acknowledged.

Time to Acknowledge Acceptance: A deadline by which the message must have finished processing and been acknowledged as such.

Time to Perform: A deadline by which a response message must be received for a request message and is measured from when the original message is successfully transmitted.

Retry Count: The maximum number of times redelivery of a message may be attempted in the face of timeouts and expired deadlines.

Retry count specifies *may* rather than *will* for redelivery as it may not be appropriate to retransmit the original messages. For example, if a message has been acknowledged as received but not as processed. The correct action here may be to abandon the interaction altogether if some critical window of opportunity has expired, but this is a business level decision.

2.8.5 Accountability Support

ebXML allows messages to specify boolean parameters that they require non-repudiation of origin, receipt or both. These parameters default to *false.* ebXML specifies that non-repudiation of origin be attached to the outgoing message and must contain some form of signature over it. The non-repudiation of receipt must contain a signature over the original message and be returned in a signal indicating acknowledgement of receipt of the message, possibly piggy-backed onto a response business message or signal.

By specifying that evidence **must be** attached to messages and responses, non-repudiation support in ebXML is vulnerable to selective receipt as discussed in Section 2.5.2. The evidence generated **may be** susceptible to key revocation if conformance and collaboration agreements in use do not mandate the involvement of a time-stamping authority to witness evidence as discussed in Section 2.5.3.

2.8.6 Fairness Support

ebXML has no notion of fairness at the business level. Conversations could be designed such that their flow guaranteed fairness but this places an expertise requirement on participants, abandons separation of concern (the conversation itself implements fairness as opposed to specifying it as a requirement) and will break business level transparency. This does not preclude the use of lower level techniques to guarantee fairness, but such support must maintain business level transparency.

ebXML does mention that TTPs may be involved in the execution of B2B interactions but makes absolutely no assumptions to what extent and how this would be achieved. The mechanisms for doing so are stated as entirely proprietary between interacting organisations, there is no mechanism within conformance and collaboration profiles to regarding TTP involvement.

2.8.7 Extensibility Support

ebXML aims to be completely extensible in that new conformance and collaboration profiles may be created and agreed upon by organisations. However, the lack of a functional consideration for fairness, and the inability to specify TTP (specifically, TSA) involvement means that conformance and collaboration profiles cannot protect against issues including selective receipt and key revocation.

Furthermore, the adoption of new conformance profiles would require organisations to commit to new conformance and collaboration agreements, possibly impacting existing interactions specified using ebXML, this thesis aims to transparently support all existing and new B2B interactions independently of specific standards, rendering this approach entirely unsuitable.

2.8.8 Summary

ebXML leaves significant amount of room for organisations to agree on how best execute conversations and support their requirements. However, the requirement that non-repudiation evidence be attached to messages and the inability to specify TTP involvement mean regardless of conformance profile in effect, ebXML exchanges will always be susceptible to selective receipt and key revocation.

The specification of certain business level requirements in functional terms (e.g., requirement for non-repudiation of origin/receipt or requirement for secure transport) is useful for inference in that functional requirements from ebXML can be aligned with functional properties provided by support middleware, this allows domain specific knowledge to be capitalised upon and is discussed in Section 3.4 and 3.6.

Critically, however, is the possibility that all information in an ebXML message (except that required to deliver it to its intended recipient) may be encrypted. Intermediary support must be able to provide the support it offers to transmissions whose contents are completely opaque to it. A message's sender, recipient and unique identifier are left in the clear such that intermediaries are able to deliver messages onwards to their final destination [OAS07b].

It is useful to consider official ebXML conformance profiles as they may indicate a suitable set of transport protocols to support. AS4 represents the most recent official conformance profile [OAS11] and it supports the use of transports including HTTP, HTTPS and SMTP, these transport choices are also supported by RosettaNet and Open Travel Alliance, discussed in the subsequent sections.

2.9 Survey: Domain Specific B2B Standards

RosettaNet and Open Travel Alliance (OTA) represent two domain specific B2B standards, designed to be completely prescriptive about all possible conversations between two participants, how these conversations must be executed and how their requirements are satisfied. These specifications allow organisations within a business domain to quickly enable interactions with other compliant partners.

2.9.1 Terminology and Stack

OTA uses exactly the same terminology as ebXML while RosettaNet refers to participants as (RosettaNet) partners. Messages are distinguished as either RosettaNet messages or general messages and exchanges and conversations are termed *activities* and *Partner Interface Processes* (PIPs) respectively. Figure 2.12 illustrates the RosettaNet stack as a concretisation of the general B2B stack described in Figure 2.1, with subsequent list points discussing their corresponding element in the figure.

1. The RosettaNet Implementation Framework and Partner Interface Process standards [Ros02, Ros09] define fully a set allowed conversations and their requirements, the syntax for conversations and messages and how the conversations and their requirements must be executed.

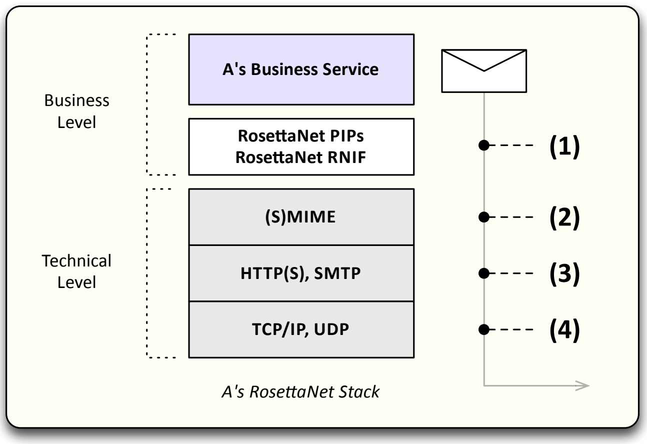


Figure 2.12: RosettaNet's concretised B2B stack.

2. RosettaNet specifies that all RosettaNet content will be encapsulated using the MIME message format, possibly Secure MIME (SMIME) if message security is required.

3. RosettaNet specifies that compliant implementations should prefer to use HTTP transport, possibly HTTPS if transport security is required. The use of SMTP is also supported but due to SMTP secured using TLS being an ad-hoc combination, HTTP should be preferred by organisations implementing their business services to use RosettaNet.

4. RosettaNet assumes that TCP/IP is used to enact the delivery of messages, it does not mandate it but says the semantics are undefined if transports use alternatives for their delivery.

OTA specifies the same technical level elements of the stack as RosettaNet, OTAs equivalent of the RNIF and PIPs standard is the single OTA 2011B document [Ope11b]. Any organisation wishing to be interact using RosettaNet or OTA or will implement its services using the chosen B2B standards and support their execution through the use of the required technical level requirements (e.g., MIME, HTTP, SMTP and TCP/IP). All stack elements for RosettaNet and OTA can be expressed using ebXML conformance and collaboration profiles.

2.9.2 Message Exchange Patterns

As with ebXML, conversations in RosettaNet and OTA are broken down into sequences of exchanges. These exchanges fall into a number of predefined patterns. RosettaNet specifies four exchange patterns of which all its conversations are composed. They are classified by two factors:

Synchronicity Is the exchange synchronous or asynchronous?

Substantive Response Is there a substantive response to the initial business message (i.e., a corresponding business message in response to the original request)?

These two factors lead to the definition of four exchange patterns: (1) synchronous one-way, (2) synchronous two-way, (3) asynchronous one-way and (4) asynchronous two-way.

One-way exchange patterns constitute a business message being sent from some *sender* to a *recipient* and the optional transmission of a business signal acknowledging *receipt* of the message back to the *sender*. In the synchronous variant, the *sender* blocks until the *recipient* responds.

Two-way exchange patterns constitute a sequence of two one-way exchanges (request and response, performed synchronously or asynchronously). As with ebXML the response must explicitly reference the request. If a *response* business message is transmitted back to the *initiator*, it provides a signal acknowledging receipt of the *response* message back to the *responder*.

OTA supports the same one-way and two-way exchanges except it does not allow synchronous blocking. That is, it conversations may only be composed using asynchronous one-way and two-way exchange patterns. RosettaNet specifies that synchronous exchanges should only be used they can be completed quickly (within seconds).

2.9.3 Security

RosettaNet messages comprise three header sections and a business payload. The payload contains business documents (and optional attachments) being transmitted from one participant to another. RosettaNet messages are constructed as a MIME Multipart/Related messages [Lev98] and may only be transmitted using HTTP, HTTPS or SMTP [FGM+99, Res00, Kle01].

The header sections are separated by duty:

Preamble Header: identifying the version of RosettaNet in use

Delivery Header: identifying the sender, recipient, whether secure transport is required, a timestamp for message creation and a unique message identifier

Service Header containing process context for a message including which specific PIP is being executed, information about the transmitting and receiving business partner and unused (as of the latest RNIF version) quality of service negotiation information.

As with ebXML, all elements of a RosettaNet message apart from the Preamble and Delivery headers may be encrypted. All RosettaNet messages are multipart MIME messages and security is achieved using secure MIME (SMIME) [Lev98] and HTTPs [Res00] where required. A secured version of SMTP exists [Hof02] but RosettaNet makes no mention of it and does not support its use.

OTA uses the same messaging format as ebXML. That is, all OTA messages are SOAP messages (SOAP with Attachments, specifically) to be transported using HTTP, HTTPS or SMTP. The same techniques described in 2.8.3 are used to secure OTA messages and their transport.

2.9.4 Reliability and Timeliness

RosettaNet provides a subset of the reliability and timeliness parameters specified by ebXML, with slightly restricted semantics:

Time to Acknowledge specifies a deadline within which a partner must acknowledge the receipt of a message. This deadline begins counting down when a message has successfully been transmitted.

Retry Count specifies the maximum number of times a message may be retransmitted beyond the initial attempt, such a retransmission would occur if the *Time to Acknowledge* deadline expired.

Time to Perform specifies a deadline within which an activity must be performed by a receiving partner. That is, for any business document requiring a corresponding response document, the deadline by which that response must be received at the initial sender after the initial message has been acknowledged and processed. For business documents requiring only acknowledgement (i.e., no corresponding response document) this parameter will be ignored.

For RosettaNet's synchronous actions, *Time to Acknowledge* and *Time to Perform* are expected to be identical and no longer than an acceptable timeout for a typical HTTP request. *Retry Count* is expected to be set to 0 as the synchronous connection blocks until a response (valid or exceptional) is received or the connection breaks.

OTA supports the same reliability and timeliness parameters as ebXML, discussed in Section 2.8.4, but it prohibits the use of ebXML's reliable messaging capabilities.

2.9.5 Accountability Support

Accountability support in RosettaNet and OTA consists of the voluntary exchange of non-repudiation evidence, exactly as with ebXML. Messages may specify boolean parameters that they require non-repudiation of origin, receipt or both. These parameters default to *false*.

The generation of the evidence is completely specified, using S/MIME message signatures [Lev98] for RosettaNet and XMLSIG and XMLENC [Wor08, Wor02] without the use of a TSA to protect against key revocation. Non-repudiation of receipt evidence must contain an MD5 or SHA-1 digest of the original message. These pieces of evidence are expected to be stored at each participant for an agreed upon amount of time (typically three to seven years according to the ebXML, RosettaNet and OTA documents).

As per discussions in Section 2.5 and2.8.5, voluntary exchange and the lack of a TSA to witness evidence renders the accountability support in RosettaNet and OTA susceptible to selective receipt and key revocation.

As a final note the RosettaNet indicates that it intends to support an additional kind of evidence demonstrating non-repudiation of routing by intermediaries in the future. As of now, “hubs are responsible for solving this in private ways.”.

2.9.6 Fairness Support

RosettaNet and OTA have no notion of fairness at the business level, participants cannot be guaranteed deterministic fairness by use of either standard. Any deviation from expected behaviour would constitute both non-compliance and break business level transparency.

Additionally, the prescriptive nature of RosettaNet and OTA means it is not possible to define a conversation whose explicit flow guarantees fairness, even though the approach is unsuitable as per discussion in Section 2.8.6.

2.9.7 Extensibility Support

RosettaNet and OTA are fully prescriptive in the syntax and semantics of all aspects of interaction including formats, protocols, transports and even cryptographic algorithms (i.e., how evidence is generated). In order to remain compliant, the observed behaviour at the business level cannot deviate from what compliant participants expect as defined by RosettaNet or OTA. Following the above, RosettaNet and OTA are not extensible and no extra support can be provided at the business level without modifications to the B2B standards themselves, motivating a lower level approach that can be applied independent of specific B2B standards or versions.

2.9.8 Summary

Domain specific standards such as RosettaNet and OTA succeed in being prescriptive in their definitions of possible conversations and exactly how these should be executed. However, there remain issues such as those discussed in [HKVO07] relating to the heterogeneity that occurs even within predefined message formats. Optional parameters can lead to situations in which organisations cannot be sure (or must otherwise agree) on which subset of optional parameters they will both support. This extra agreement is in direct contrast with the aims of removing all ambiguity. Haller et al. also highlighted RosettaNet's lack of support for indicating the units of some parameters (e.g., currency) in their work, they attempt to solve this using an intermediary driven approach employing ontologies to automatically transform RosettaNet messages between participants.

The lack of extensibility, susceptibility to selective receipt and key exchange motivate the pursuit of a lower level approach, such as the intermediary middleware designed by this thesis. By operating as transparent, decoupled intermediaries, stronger (and new) kinds of support (e.g., fairness and accountability) can be offered seamlessly to existing and new domain specific B2B standards.

As a final comment on the ability of RosettaNet and OTA to be expressed as a subset of ebXML, both domain specific standards make use of ebXML file formats for the definition of all of their conversations. That is, all OTA and RosettaNet conversations are specifies in ebXML Business Process Specification files. Domain specific standards were surveyed here to demonstrate that they share common concerns with general standards for supporting their B2B interactions.

2.10 Summary of B2B Standards Survey

Section 2.8 and 2.9 demonstrated that while B2B standards acknowledge and attempt to address common business level requirements, the mechanisms for doing so vary between standards and may not adequately address concerns such as fairness and accountability, demonstrated to be vital in supporting the regulation of B2B interactions in Section 1.2 and 2.5. The support for non-repudiation in ebXML, RosettaNet and OTA is specified in such a way that does not protect against key revocation or selective receipt. These issues motivate the pursuit of a lower level approach, such as transparent and decoupled intermediaries designed and implemented in this thesis.

The most useful aspects of the survey are in the discovery of common transport protocols, messaging formats and message exchange patterns. Chapter 3 and 4 will generalise and integrate these findings into the middleware design such that the support remains transparent, decoupled and asynchronous while supporting a useful range of B2B standards (e.g., by supporting common transports such as HTTP, HTTPs and SMTP and understanding SOAP and MIME messages where possible).

While the contents of intercepted messages may be completely opaque (through content or transport security), there may be instances in which additional information contained within a message is available to support intermediaries. The knowledge learned from surveying specific standards can be used to enrich the functionality provided by intermediaries (e.g., by allowing B2B standards level requirements to infer functional properties offered by intermediary support).

Briefly, surveying current B2B standards yielded a list of general message exchange patterns, useful transports to facilitate transparent interception, common requirements for supporting B2B interactions and demonstrated that a lower level approach is suitable for providing stronger and additional guarantees to a range of existing B2B standards.

2.11 Cloud Computing

Section 2.3 discussed business services and security services and that providers of these services are responsible for satisfying their infrastructure requirements. These infrastructure requirements can be broadly broken down into processing, storage and communication requirements. Service providers (e.g., security service providers) can use cloud computing to provide any or all of these requirements, alleviating them from having to deploy and maintain their own hardware. Cloud computing allows services to be delivered to cloud users (e.g., business and security service providers in this thesis) under three models [AFG$^+$10], listed in increasing level of abstraction:

Infrastructure as a Service (IaaS) where a provider offers infrastructure level resources (e.g., virtual machines, block storage and networks), on demand. The cloud user is responsible for configuring OSs and execution environments of the infrastructure provisioned to them.

Platform as a Service (PaaS) where a provider offers a pre-configured platform (or execution environment) into which cloud users can deploy and execute their service(s). The infrastructure underlying the provisioned platform is automatically scaled to match demand such that the cloud user does not have to manually request and allocate resources from the cloud provider.

Software as a Service where application software is provided to cloud users as a service and the underlying platform and infrastructure are managed by the cloud provider.

The work carried out in this thesis is concerned with the use of IaaS and PaaS as an enabler for deploying and executing security services which are composed to provide intermediary support. Security and Business services as described in Section 2.3 may be provided under the SaaS model but where I am concerned with provisioning security services, IaaS and PaaS are the only suitable methods for doing so in the cloud.

The benefits of using cloud computing include alleviating physical deployment requirements from a service provider, allowing services to be operated to a cost of only what they consume (e.g., computation, bandwidth and storage), on demand scaling (in all models of delivery) and elasticity. Elasticity refers to the ability to scale up and down across all tiers of an application, that is, any part of the software may be scaled up and down as needed independently of another. The ability to scale down is vital to prevent underutilization and ensure cloud users are not charged for resources they do not need or utilise. That is, when elements are scaled up to cope with a surge, they do not remain scaled up longer than required at a cost to the cloud user.

Enabling elasticity depends on the model of cloud computing in use, under IaaS the cloud user can simply request more virtual machines or increasingly powerful virtual machines but this does not automatically software deployed on them will scale in the manner required. Under PaaS elasticity can be more automated as the infrastructure underlying specific aspects of the platform can be automatically scaled up and down as required.

2.11.1 Trust, Privacy and Compliance

Santos et al. discussed the trust concerns users may have in deploying and developing their applications to leverage cloud computing [SGR09]. The main concerns regard trust in the cloud environment even against insider threat. That is, can a cloud user trust that their execution environment within the cloud is secure (and trusted)? Privacy concerns also factor in to critical data which at some point may exist unencrypted within the cloud either in the memory of a virtual machine or stored on some cloud storage. Chapter 6 discusses work and approaches taken to securing execution environments within the cloud and addressing issues of trust.

Beyond trust and privacy concerns, cloud users deploying services within the cloud must be aware of compliance issues. Legal issues regarding the storage and processing of data (e.g., Payment Card Information Data Security Standard (PCI DSS) and Data Protection Acts (DPA) in Europe and Health Information Portability and Accountability Act (HIPAA) and Sarbanes-Oxley Act (SOX) in the US) may affect to what extent cloud computing may be leveraged or how it is used [SW07].

It is beyond the scope of this thesis to provide solutions catering to specific compliance legislation but the existence of such legislation makes required that hybrid approaches must be available. That is, approaches in which elements of infrastructure are hosted by a service provider or organisation. The design presented in Chapter 4 discusses deployment and composition configurations for intermediary support allowing for these hybrid approaches.

2.12 Related Work

2.12.1 Law Governed Interaction

Minsky and Ungureanu developed a system described as Law Governed Interaction (LGI) representing one of the earliest attempts at to support regulation between autonomous organisations [MU00]. Under this system, all organisations have their messages routed through trusted controllers. These controllers contain a set of rules, machine readable representations of some law to be enforced among the controllers.

When organisations (or agents, in their terminology) dispatch messages for delivery, a controller intercepts the message and ensures that its contents and delivery will not violate the defined rules and takes the appropriate action (the rules are captured as event-condition-action statements). Thus, misbehaviour (against some globally agreed upon laws) is prevented by the actions of the trusted controllers.

The work enforces specific message formats, transports and a language for the expression of rules representing laws (a restricted form of Prolog). Critically, it requires all possible state transitions (events) to be know such that event-condition-action rules can be programmed for each event. While the work is not aimed specifically at supporting conversations or B2B interactions, the principles of satisfaction (or enforcement) of requirements by intermediaries acting on behalf of participants is similar to the technique explored in this thesis.

The LGI implementation assumes one global set of laws to which everyone must adhere. Controllers facilitate this by storing control states (CS) for each agent. The CS for each agent represents the actions they are allowed to take from their current state. Critically, the requirements apply to all participants and there is no capability for expressing individual requirements per participant.

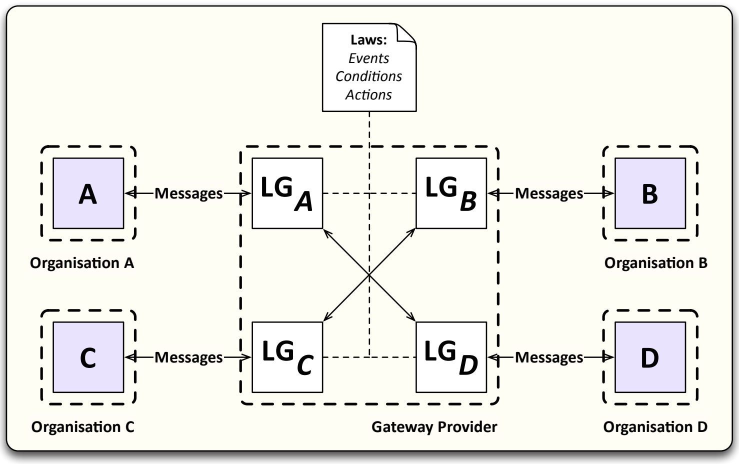


Figure 2.13: Each *LG* gateway acts to ensure only legal (valid) actions on behalf of its organisation.

The intermediary support in this thesis supports organisations both individual requirements and decoupled operation. That is, organisations are free to use whichever B2B standard(s) they desire (and associated messaging and transport standards) and express their own requirements about how their interactions are supported. Additionally, the support can be provided without the requirement for prior knowledge about all possible interactions. Section 3.6 will demonstrate that any interaction can be supported without any requirement of knowledge regarding the conversation it belongs to.

2.12.2 Conversation Support for Business Process Integration

Hanson et al. commented on the value of supporting B2B interactions at lower levels of abstraction, specifically that organisations be allowed to retain autonomy and "sovereignty" in their business level operations [HNK02]. Their work expresses business conversations (including converting existing conversations) as state machines known as *conversation policies* (CPs), these policies are described as machine readable representations of message exchange patterns. Similarly to Section 2.2 they identify B2B as consisting of separate layers including high (business) and low (support) levels.

Gateways are provisioned to act on behalf of interacting organisations, messages flow through these gateways who act to enforce the expected flow of a CP (i.e., that attempted state transitions are allowed). The gateways enable interoperability and interworking by allowing organisations to engage in conversations over different transports and B2B standards. In a sense, the work functions as an enterprise service bus through which all messages pass and conversation flow is enforced for the participants. Messages not adhering to the expected flow will never propagate to the business level. Figure 2.14 illustrates the approach, showing that only valid and expected messages are passed back to A.

The desire to allow organisations to retain autonomy is beneficial as they are freed to use the most suitable standards and conversations supporting their interactions. The offered interoperability and interworking across multiple transports and B2B standards is theoretically an enabler for collaboration between organisations. However, the requirement that all conversations must be translated into a corresponding CP representation represent integration and expertise requirements upon interacting organisations. For ex-

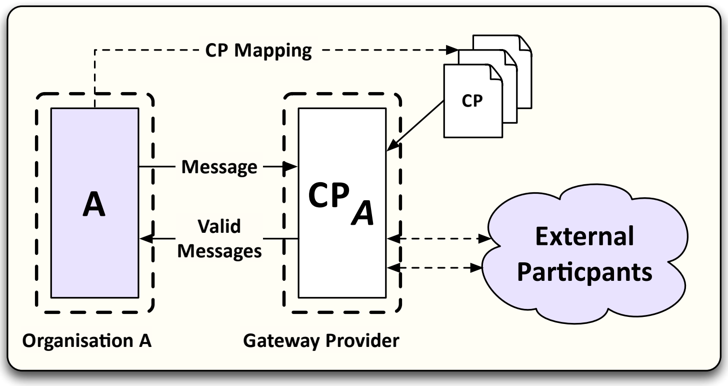


Figure 2.14: CP_a acts on behalf of A to prevent it from seeing late or unexpected business messages.

ample, they would have to ensure that CPs representing conversations were valid, that the same conversations using different B2B standards resulted in the same generation of the same CP and ensuring CPs are updated whenever conversation definitions are modified.

The CPs in the work presented allow the contents, order and delivery deadlines for messages in a conversation to be defined but do not support the expression of other business requirements. This means the gateways do not allow an individual organisation's requirements to be supported. The work focuses purely on ensuring the expected order (and timing) of conversations by preventing unexpected or late messages from being propagated to the business level. Requirements such as fairness, accountability and consistency are not supported or discussed.

Additionally, the functionality of the gateways relies completely on the ability to understand the contents of intercepted messages such that the state of a conversation can be discerned. This requires that some contents of the business message are unencrypted (or can be decrypted). The intermediary support middleware developed in this thesis enables support for all B2B interactions including transmissions whose contents are opaque. However, where the contents of a transmission are available, domain specific knowledge will be capitalised upon in order to improve support.

In summary, the approach enforces transparency and allows the underlying B2B standard, messaging formats and transport protocols to be abstracted away. The caveat is that the gateways are not decoupled in so much as they require the contents of a message to be both available and understood to discern conversation state and prevent invalid messages and actions.

The support offered in this work is improved over LGI, discussed in the previous section, in that individual supports for organisations can be addressed. That is, the CPs used are specific to the message flow expected in and out of the supported organisation. This thesis will further improve support by enforcing decoupling, allowing opaque transmissions to be supported, and by allowing intermediaries to provide other kinds of support (e.g., fairness and accountability).

2.12.3 FIDES Fair Exchange System

The FIDES system represents a bespoke client-server based approach to implementing non-repudiation between organisations [NZB04], completely abandoning transparency and automation in favour of providing strong fairness guarantees. Each organisation deploys a FIDES server within their own infrastructure and FIDES client software within their business level. The FIDES client provides an interface into which items can be uploaded and their recipient designated. The recipient must also use the FIDES system and have their own FIDES servers and clients configured.

The FIDES client will engage an organisation's local FIDES server to communicate with the recipient's FIDES server to execute some protocol (engaging with FIDES TTP servers to guarantee fairness where required) to non-repudiably transmit information from sender to recipient. The specific non-repudiation protocols employed by FIDES produce transparent evidence, discussed in Section 2.6.3, avoiding situations in which ascertaining TTP involvement may cause inadvertent discrimination again a participant.

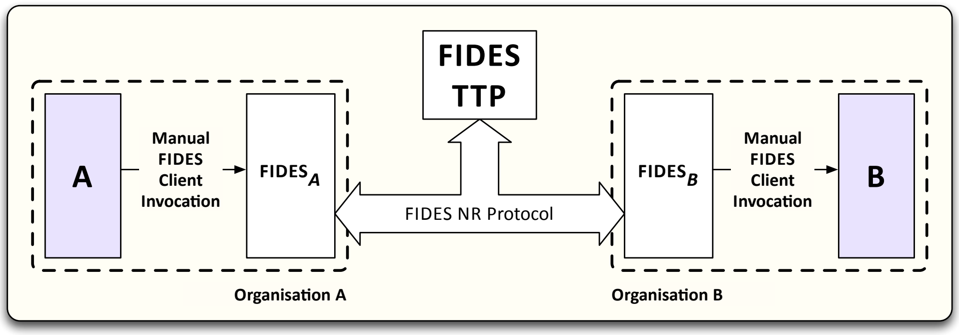


Figure 2.15: Organisations employ FIDES clients, servers and TTPs to facilitate non-repudiable exchange of information.

A drawback to this approach is that no modes of interaction with the FIDES servers are offered except the FIDES client application. This makes it impossible to integrate FIDES into an automated B2B flow. Manual intervention would be required to load each document generated by a business service into the FIDES client, specify its recipient and then invoke the exchange. Similarly, the recipient would have to take the received document and feed it into their receiving business service. This breaks business level transparency and introduces potentially redundant information. For example, a business document will contain information about its recipient, this information must then be (correctly) duplicated in the FIDES client interface in order to deliver the document to its intended destination.

FIDES uses the Java Messaging Service (JMS) standard for all of its communication within and between organisations, Section 5.3 contains a detailed summary of why JMS is ill-suited to facilitating communication between different organisations. Specifically, it lacks a standardised remote addressing mechanism and specifies no transport or wire level format, meaning two vendors' implementations of the same JMS reference API will be incompatible, causing tight coupling between organisations wishing to employ the non-repudiation support offered. Furthermore, it does not render all JMS deliveries non-repudiable, it simply uses JMS to facilitate its own application level messages resulting in the non-repudiable delivery of some business item(s) from sender to recipient. This thesis will extend on FIDES by supporting automation, provision of support independent of specific B2B standards, asynchronous operation and the use of multiple messaging standards

2.12.4 Interceptor Based support for Non-repudiation Protocols

Cook [Coo06] (and Cook et al. subsequently in [CRS06]) developed interceptor-based support to execute non-repudiation protocols in the delivery of SOAP and Java Enterprise Edition 2.0 (J2EE) messages. Different non-repudiation protocols are used to render service invocations and updates to shared information non-repudiable. The system employs one interceptor per participant, responsible for execution NR protocols and engaging with TTPs where required). Under this work all components of an organisation's intermediary are co-located with their business services and the interceptor components are configured as part of the organisation's J2EE or SOAP stacks.

In addition to the execution of non-repudiation at lower levels, allowing organisations to retain autonomy at their business level operation (and ensuring business level transparency), business level validation results may be fed into the lower level non-repudiation support to trigger early protocol termination when a business level failure has occurred. Mechanisms for doing this are provided specifically for the technologies used in the implementations (SOAP and J2EE), Section 4.6 discusses similar aspects in the consideration of how to deal with events that cross layers of abstraction and whether these should be hidden, passed up or invoke other actions.

The support offered by the interceptors functions in a synchronous manner and intermediaries acting on behalf of organisations must all be online at the same time to execute the non-repudiation protocol. In a sense, the work acts to guarantee non-repudiation for object invocation middleware (it renders the invocations non-repudiable). The work was specifically written to deal with J2EE and SOAP messaging although the design can be

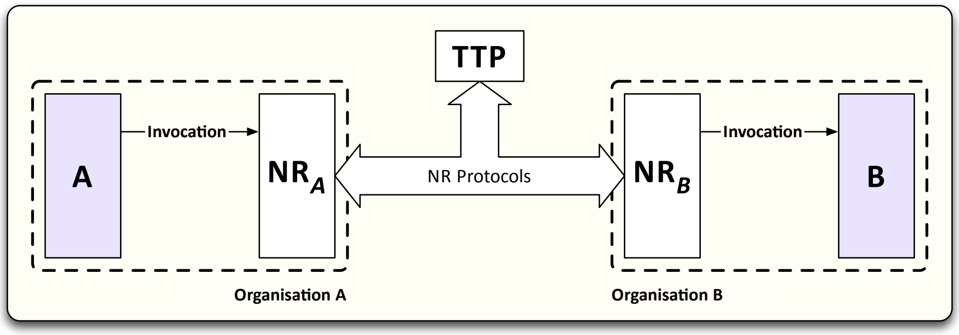


Figure 2.16: Intermediaries deployed within organisations interact (synchronously) to execute non-repudiation protocols for service invocations.

adapted to other technologies. The intermediary support in this thesis is designed such that additional properties can be provided to multiple messaging and transport formats and standards, presuming interactions using them can be intercepted. Chapter 3 and 4 discuss these issues in more detail.

Cook's work represents an improvement over the FIDES work in that it supports automated operation in existing and new B2B interactions and the use of more than one non-repudiation protocol to address different situations (non-repudiable service invocation vs non-repudiable updates to share information). Cook's work was used as a starting point for this thesis, we seek to expand upon the approach by providing a generalised middleware in which the previous work can be expressed as a mapping of functional properties (e.g., accountability) to a set of protocols (e.g., non-repudiation). Section 4.12 demonstrates Cook's work can be expressed as an instance of deployment and composition within the middleware designed by this thesis. Beyond this, middleware designed in this thesis will operate asynchronously and be constructed in such a way that it can employed by an organisation simply by being placed in the delivery path of its messages, as opposed to requiring integration into their B2B stack.

2.12.5 Accountability as a Service for the Cloud

Yao et al. propose a solution to provide accountability as a service to interacting participants [YCW$^+$10]. The work defines two domains, the business service domain (BSD) and the accountability service domain (ASD), referring to both domains as part of a Trusted Service Oriented Architecture (TSOA). The ASD contains one or more accountability services, supplied with accountability evidence for each action taken by all participants. That is, for a message transmitted from A to B, A is expected to send accountability evidence to accountability services within the ASD. All evidence is generated by the interacting organisations and there is an assumption that this evidence is irrefutable but no discussion of how to achieve this. In the face of a dispute, one or more accountability services within the same domain would reach consensus and provide a decision on who was to be held account as a resolution of the dispute.

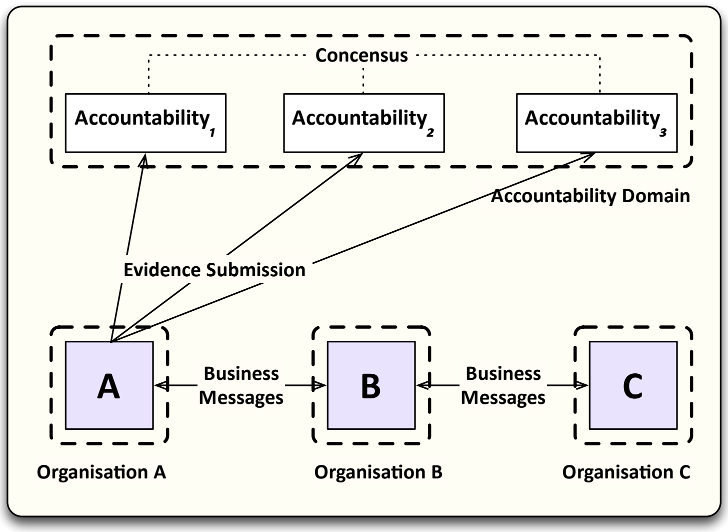


Figure 2.17: Yao's work

An issue here is the lack of fairness guarantees, all evidence submission is voluntary. Any participant providing evidence during in an interaction is provided with no guarantees that others will do the same, in the case of a dispute it becomes impossible to say a participant who did not provide evidence specifically did misbehave (i.e., a lack of evidence of behaviour does not constitute evidence of misbehaviour) and agreements can become unenforceable. Work in this thesis seeks to address accountability and fairness guarantees such that agreements always remain enforceable and well behaved participants are never placed at a disadvantage as a result of correct behaviour.

When contrasted with the aims of this thesis in Section 1.5, the lack of transparency requires that organisations augment their business level operation specifically to generate and emit the accountability evidence associated with every action they take. The intermediary support designed in this thesis allows these operations to be undertaken by intermediaries but also supports the use case that an organisation may wish to generate and emit their own evidence (e.g., limited trust in intermediaries or wishing to retain private keys used to generate accountability evidence), discussed in Section 4.10.

The approach taken by Yao could be encapsulated within the intermediary based design developed in this thesis. In such a scenario, intermediaries acting on behalf of participants would generate and submit evidence on their behalf to accountability services. This would allow Yao's system to be used transparently but does not address the lack of fairness guarantees where other participants are not completely trusted. Section 4.12 discusses the encapsulation of Yao's work within the generalised design proposed in this thesis.

Yao's approach is useful in that the protocol execution is offloaded in a decoupled manner and abstracted away from organisations (i.e., "as a Service"). Evidence is provided to the accountability domain and decisions are emitted from the accountability domain where required.

Work in this thesis goes a step further by supporting the case where the evidence generation, submission and storage can be transparently encapsulated at lower levels but still be available to organisations should they need it. In this sense, the technical layer of the thesis can be considered the provision of supportive properties as services (e.g., accountability as a service, fairness as a service, consistency as a service), with intermediaries serving to transparently encapsulate the invocation of these services on behalf of interacting participants. Chapter 4 and 5 will discuss this approach to invoking underlying support.

2.12.6 Extending Messaging with Application Conditions

Tai et al. propose an architecture in which conditions can be annotated onto outgoing messages as metadata to be satisfied in the course of their delivery[TMRS02]. The satisfaction of these conditions is performed and verified by one or more intermediaries in the delivery of a message from its sender to recipient. This system is unique among the surveyed related work in that it allows the sender of a message to express their individual requirements on any outgoing message and have them satisfied by the lower levels of the system. The strict use of only message oriented middleware (specifically using only queues and topics for point-to-point and publish-subscribe communication respectively) renders the approach asynchronous.

In the face of no intermediaries to provide additional support for the expressed conditions, messages are delivered as they would have been normally with no additional support available. I discuss the inclusion of this consideration in Section 4.4 when considering interaction with an organisation who does not employ intermediary support for their B2B interactions. In such cases, strict transparency would dictate that interactions continue unsupported (alternatives are also discussed in which an organisation may opt not to continue interacting if properties cannot be guaranteed).

The work is not targeted specifically at B2B and as such, the example requirements are proof of concept (examples include deadlines for delivery and ensuring a minimum number of recipients). However, the design presents useful considerations for adding support to existing message oriented systems while preserving transparency at higher level of abstraction (e.g., business level transparency).

2.12.7 Summary of Related Work

The related work represents approaches that attempt to satisfy single properties [YCW+10, CRS06, NZB04], use intermediaries to ensure established conditions are not violated for observed interactions [MU00, HNK02] or allow individual messages to express additional capabilities [TMRS02]. Yao's work presents useful considerations for separation of support offered (e.g., accountability as a service) from its invocation (e.g., an intermediary whose purpose is to transparently encapsulate the invocation of services on behalf of some organisation).

The middleware developed in this thesis will demonstrate that individual concerns can be addressed by the mapping of functional properties to technical protocols within middleware intermediaries. By addressing the challenges of transparency, loose coupling and asynchronicity (as discussed in Section 1.5), the support will be demonstrated as applicable to all B2B interactions and more generally, any message oriented interactions. Section 6.3 will discuss applications beyond messaging middleware for future work.

2.13 Summary

This chapter introduced terminology, concepts and assumptions regarding the execution and support B2B interactions and their regulation. Fairness, accountability and consistency were demonstrated as useful concerns to address for B2B interactions alongside technical protocols capable of satisfying these requirements. The B2B standards survey motivated the lower level intermediary approach explored by this thesis.

The B2B standards survey and related work inform decisions taken in the theoretical and practical elements of this thesis and serve as a base from which useful generalisations can be made. These generalisations are discussed throughout Chapter 3 and 4.

3 Generalisation of Survey and Background to Enable Support

This chapter combines surveyed B2B standards, identified support concerns, technical protocols and related work to generalise message exchange patterns, common requirements, example functional properties and declaration mechanisms. The generalisations and mechanisms are used to design and implement the intermediary support middleware in Chapter 4 and 5.

Section 3.1 generalises the exchange patterns that are characteristic of B2B conversations expressed by the surveyed standards. These exchange patterns are decomposed into the constituent units of communication when dealing with B2B interactions. By supporting the smallest unit of interaction, support for bigger units of interaction can be composed. Section 3.2 generalises the common requirements supported by surveyed B2B standards. These generalisations are compared to the concerns of fairness, accountability and consistency motivated in Section 2.5.

Section 3.3 proposes hierarchies mapping properties of fairness, accountability and consistency to technical protocols that provide their satisfaction with varying characteristics. These mappings will be realised by the implementations in Chapter 5.

Section 3.4 introduces the declaration mechanisms allowing the specification of when supported properties (e.g., accountability) should be satisfied for intercepted interactions. Two methods of declaration are discussed allowing individual transmissions to be annotated with extra requirements and allowing the creation of decoupled rules ca be matched against any intercepted transmission to see if support should be satisfied.

Section 3.5 and 3.6 contain discussions around supporting individual business transmissions as the fundamental unit of business communication and the various ways in which the declaration mechanisms may be used such that support can be offered to opaque transmissions, capitalise on business level information where available and even support new B2B standards through the use of content matching.

3.1 Generalised Message Exchange Patterns

All surveyed standards classified one and two-way exchanges as the exchange of a single business message or the exchange of two related business messages respectively. The survey demonstrated that the two-way exchanges can be decomposed into two (related) one-way associations. The one-way exchanges can be further decomposed into sequences of transmissions. Figure 3.1 illustrates the decomposition:

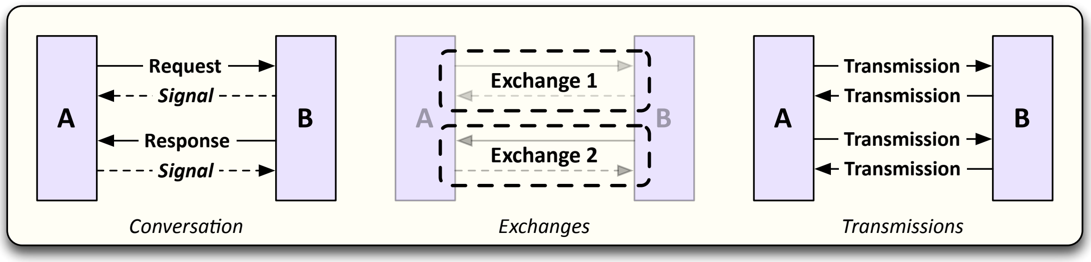


Figure 3.1: Recursive decomposition of conversations and exchanges.

By supporting individual business level transmissions, support can be provided to exchanges (as sequences of transmissions) and conversations (as sequences exchanges). Section 3.5 discusses supporting individual transmissions and fairness and accountability concerns.

3.2 Generalised Business Requirements

The B2B standards survey demonstrated that supported conversations are allowed to specify similar types of requirements addressing areas including reliability, security, timeliness, non-repudiation and consistency. This section discusses those common requirements such that the example properties for fairness, accountability and fairness may be defined in Section 3.3, attempting to provide stronger guarantees where B2B standards and generalised requirements fall short.

3.2.1 Reliability

Reliability concerns to messages and conversations are generally tackled in two ways. The first is a binary requirement that a message should be 'reliably transmitted', usually involving the invocation of some reliable delivery mechanisms (e.g., ebXML reliable delivery protocol) designed to achieve at-least-once message delivery.

To maintain business level transparency, underlying support intermediaries must use transports compatible with the business level requirements of standards in use. Reliable messaging protocols invoked at (or by) the business level remain a business level concern and will ultimately be delivered to recipients as individual transmissions.

3.2.2 Timeliness

The timeliness of message delivery, acknowledgement and processing is facilitated by the following parameters:

Timeout A timeout is the length of time after which a single delivery attempt of a message should be considered to have failed

Deadlines for Acknowledgement defines a deadline by which the receipt of a message must be acknowledged to the sender by its recipient

Deadlines for Processing defines a deadline by which the recipient of a message must have processed the contents of a message acknowledge this fact to its sender

Retry Count The retry count dictates the number of times attempted retransmission of a message should occur in the face of timeouts or communication failures

Ignore Duplicates Repeated attempted delivery may result in the delivery of duplicate messages to an organisation, this field indicates that duplicates should be discarded, guaranteeing at-most-one processing of a message

The passing of deadlines or timeouts will result in the retransmission of a message until the specified retry count has been reached.

These parameters are business level concerns but underlying support middleware must be mindful that its functionality (e.g., the execution of additional protocols to provide support) does not cause timeouts or deadlines to expire. Section 5.5.5 discusses typical lifespans and deadlines of B2B conversations and compares them to timing measurements taken from implementations to evaluate the impact of providing extra support to B2B interactions.

3.2.3 Security

The end-to-end security of messages exchanged during interactions is ensured through two means:

Content Security in which the partial or entire contents of a message are encrypted. This allows their delivery over insecure channels without compromising their contents. Where content security is in place, minimal information about the message must be left unencrypted to facilitate its delivery. This minimal information comprises: the identity of the sender, identity of the recipient and the unique message identifier.

Transport Security is designed so that the contents of all transmissions across a channel do not have their contents compromised. Where content security allows messages to be transmitted over unsecured channels, transport security allows the transport of unencrypted messages over secured channels without compromising their contents. As with content security, the sender, recipient and message identifier of the message are always known.

Intermediary support must be able to function using the minimal available information for intercepted transmissions although other information may be used where available, to enrich functionality. Section 3.6 discusses capitalising on available business information while ensuring support can be offered to all transmissions.

3.2.4 Accountability

All surveyed B2B standards acknowledge accountability and attempt to satisfy it using non-repudiation evidence. All standards mandate this evidence is exchanged voluntarily owing to the complexities of specifying TTP involvement and guaranteeing fairness. As such, all standards support two binary properties:

Non-repudiation of Origin Required indicating whether a sender is expected to provide evidence binding itself to the origin of a given message

Non-repudiation of Receipt Required indicated whether a recipient is expected to provide evidence binding itself to the receipt of a given message

The surveyed standards (including real world ebXML conformance profiles) specify the generation of evidence that is both susceptible to key revocation and exchange without fairness guarantees.

Intermediary support in this thesis will consider the appearance of requirement for any kind of non-repudiation evidence as an indication that accountability should be provided for both participants with the strongest guarantees necessary, protecting against selective receipt and key revocation issues as discussed in Section 2.5.

3.2.5 Fairness

The surveyed B2B standards have no business level notion of fairness, this is due to the complexity of guaranteeing fairness itself and the impacts it would have on allowable conversations. That is, fairness for B2B interactions can be provided when there is at least one transmission in either direction between two participants.

The importance of fairness, discussed in Section 2.5.2, prompts the discussion in Section 3.5 about how fairness can be provided by individual transmissions. This is achieved by maintaining the business level abstraction of an individual transmission while altering the operation at technical levels. Where insufficient levels of trust are expressed, intermediary support will attempt to provide the strongest levels of fairness it is capable of.

3.2.6 Consistency

As with fairness, the surveyed standards have no business level notion of consistency requirements. However, they do acknowledge notifications mechanisms are required to invoke corrective measures when problems (such as inconsistency) are detected.

Section 2.6.4 discussed that where consensus can be quickly and unambiguously established, the time to detect failures or inconsistencies can be reduced or even prevented from manifesting at the business level.

The cost of synchronisation means that the frequency with which it occurs may have an impact on deadlines, timeouts and a conversation's overall execution time. Section 3.3.4 discusses potential characteristics of consistency as a property to address these issues.

3.2.7 Summary of Generalised Requirements

Reliability and timeliness concerns prove difficult to provide additional support for as they closely coupled to the business level. That is, the expiration of deadlines or timeouts will generally trigger a business level redelivery of a message. From a technical level perspective this constitutes an entirely new transmission and should be supported as such.

Intermediary support should, naturally, attempt to minimise its impact on the delivery of a message (e.g., avoid causing message delivery to fail), Section 4.5 more thoroughly discusses the possible effects of intermediaries on deadlines, timeouts and processing.

Similarly, little additional support can be offered to the two security options for messages. If a message requires content security, it will have been encrypted before being routed through the intermediary support, thus the intermediary needs only pass the message onwards once it has finished processing the data (e.g., executed the required protocols to satisfy the desired support). Where transport security is required, the intermediaries must support the transport being used and allow its secure operation.

Fairness, accountability and consistency all represent concerns that can be aided by intermediary support. Accountability and fairness can, notably, be provided to any transmission even if the contents are encrypted. For the above reasons, accountability and fairness properties are defined in the next section and implemented in Chapter 5. Discussion of consistency as a property is included in the next section as an example of how the design proposed in Chapter 4 could be extended to provide additional support.

3.3 Fairness, Accountability and Consistency as Functional Property Hierarchies

Section 2.5 motivated the satisfaction of fairness, accountability and consistency when supporting B2B interactions. The previous section demonstrated that intermediary support provides an ideal place to transparently provide support for these concerns. This section defines hierarchies of functional properties for satisfying fairness, accountability and consistency with varying characteristics and which protocols are used to satisfy those combinations. Trust will be used as the determining factor for when fairness is required (both as a property and as a characteristic of accountability).

3.3.1 Expressing Levels of Trust

This work uses the three levels of trust discussed in Section 2.5.1.

For the purposes of fairness guarantees, semi-trusted participants are considered untrusted. That is, fairness guarantees will only be relaxed in the presence of complete trust. Semi-trusted means an entity is expected to misbehave on their own but will not collaborate to do so, this reflects the concern that most organisations will privilege protecting their own interests above engaging in an interaction.

Importantly, these trust properties are not symmetric, a sender may have complete trust in a recipient while the recipient has no trust in the sender. The support for each organisation to specify their own declarations allow this expression.

Table 3.1 defines the values representing the trust relationship between two participants.

Characteristic	**Example**
FullyTrusted	Sender's complete trust in recipient
SemiTrusted	Sender's semi-trust in recipient
UnTrusted	Sender does not trust recipient

Table 3.1: Functional characteristics representing varying levels of *Trust*

Where unspecified, the default trust relationship between participants is assumed to be *UnTrusted*, requiring be it explicitly established by declaration where complete trust is present.

It is possible that these three levels are too coarsely grained and a more suitable metric for trust could be used. Potentially some probability $0 \leq p \leq 1$ that a recipient can be trusted, Section 2.4 and 6.3 discuss different possibilities for expressing trust metrics between different participants.

3.3.2 Fairness Hierarchy

Figure 3.2 illustrates the hierarchy of fairness characteristics identified by this work. These match the definitions provided in the discussion of fairness in section 2.5.2. Table 3.2 summarises the characteristics shown in the hierarchy.

Characteristic	**Description**
Probabilistic	No deterministic fairness guarantees, some p of fairness after an exchange
Deterministic	Fairness is guaranteed during or after an exchange
Strong	Fairness is never violated during an exchange
Weak	Fairness may be lost during an exchange but must be subsequently recovered
Transparent	Involvement by TTP is indistinguishable
Optimistic	TTP is only involved when required

Table 3.2: Characteristics for the *Fairness* property

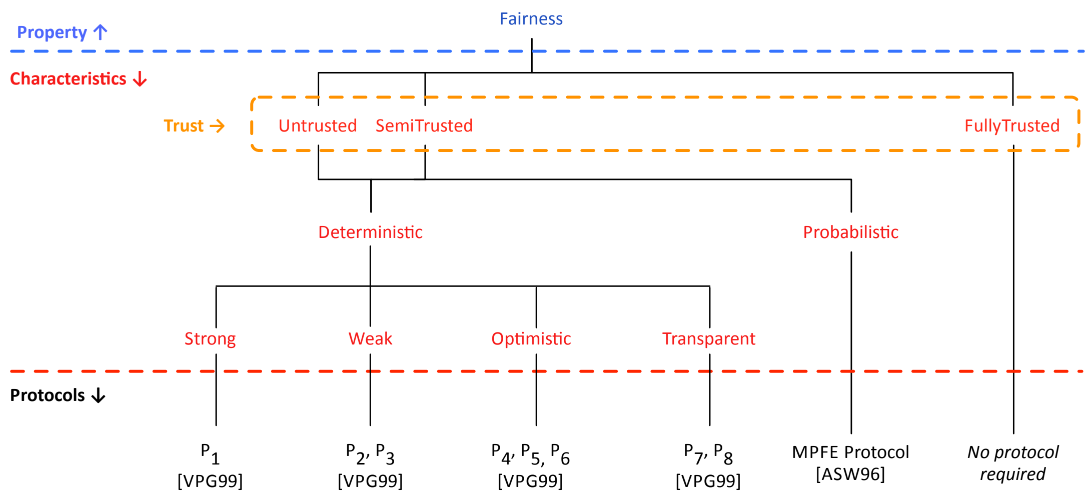


Figure 3.2: *Fairness* property hierarchy

While omitted for clarity, some protocols from [VPG99] satisfy multiple deterministic characteristics, summarised in the Table 3.3:

Protocol	Characteristics Satisfied
P_3, P_5	*Deterministic*, *Weak*, *Optimistic*
P_6	*Deterministic*, *Strong*, *Optimistic*
P_7	*Deterministic*, *Weak*, *Transparent*, *Optimistic*
P_8	*Deterministic*, *Strong*, *Transparent*, *Optimistic*

Table 3.3: Multiple characteristics satisfied by fair exchange protocols in [VPG99].

3.3.3 Accountability Hierarchy

Figure 3.3 illustrates the hierarchy of properties, characteristics and protocol mappings for accountability.

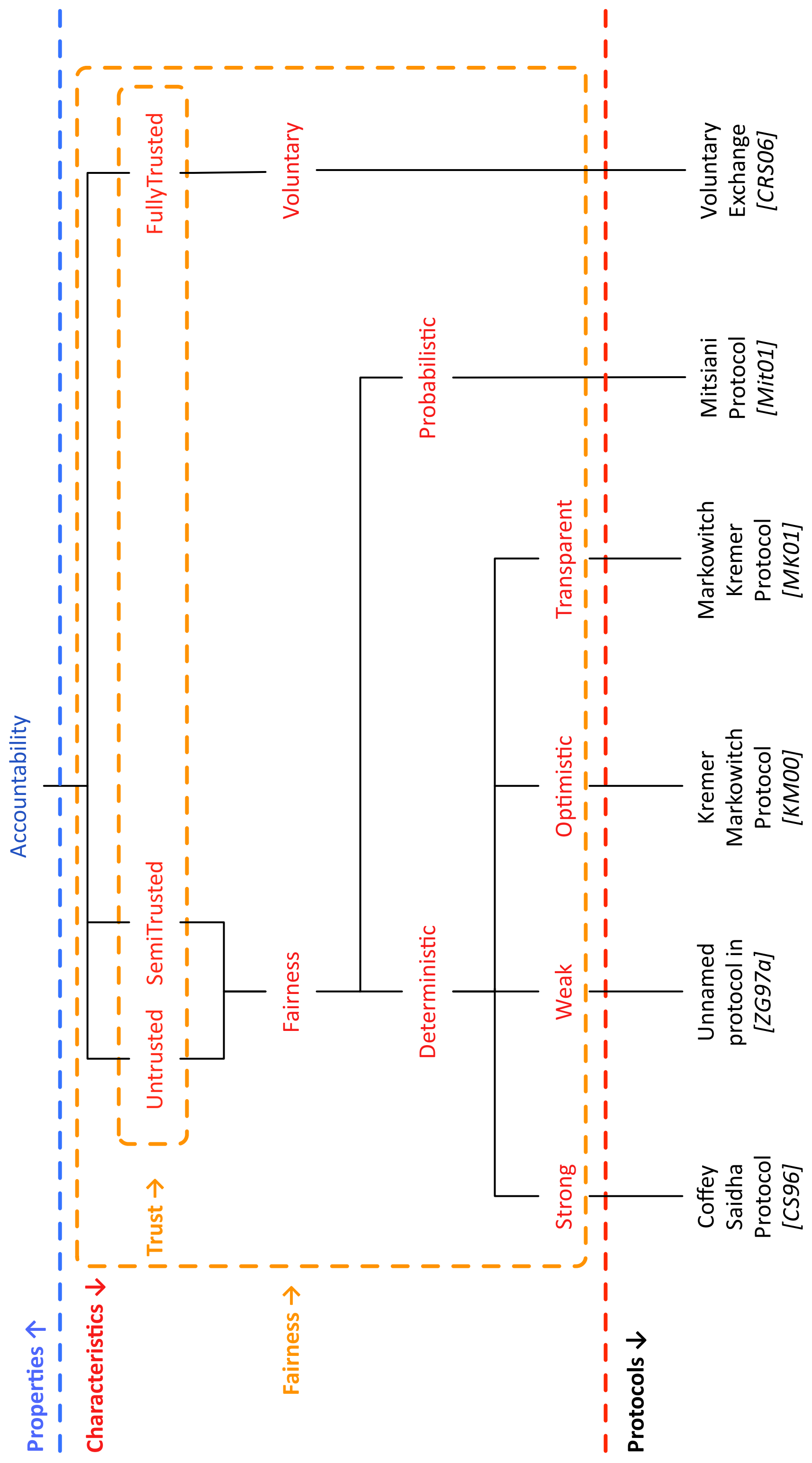


Figure 3.3: *Accountability* property hierarchy

Figure 3.3 indicates the protocols used to satisfy accountability with varying characteristics. Table 3.4 summarises the hierarchy.

Protocol Name	**Characteristics Satisfied**
Coffey-Saidha	*UnTrusted* or *SemiTrusted*, *Fairness*, *Deterministic*, *Strong*(*Inline*)
Zhou-Gollman	*UnTrusted* or *SemiTrusted*, *Fairness*, *Deterministic*, *Strong*(*Online*)
Weak Protocol in [ZG97a]	*UnTrusted* or *SemiTrusted*, *Fairness*, *Deterministic*, *Weak*
Kremer-Markowitch	*UnTrusted* or *SemiTrusted*, *Fairness*, *Deterministic*, *Strong*, *Optimistic*
Markowitch-Kremer	*UnTrusted* or *SemiTrusted*, *Fairness*, *Deterministic*, *Strong*, *Optimistic*, *Transparent*
Mitsiani	*UnTrusted* or *SemiTrusted*, *Fairness*, *Probabilistic*
Voluntary Exchange	*FullyTrusted*, *Voluntary*

Table 3.4: Non-repudiation protocols and the *Accountability* characteristics they satisfy

The Zhou-Gollman protocol [ZG97b] is not indicated in Figure 3.3 for the sake of space, it would occupy the same space as the Coffey-Saidha protocol. It would also be possible to insert additional characteristics into the hierarchy to force the use of protocols with *Inline* or *Online TTP* requirements.

The fairness hierarchy in Figure 3.2 can be seen within the accountability hierarchy shown in Figure 3.3. This references the discussion in Section 2.5.4 where accountability and fairness are guaranteed for transmissions to protect both during and after exchanges.

The implementations in Chapter 5 will default to providing deterministic strong fairness where finer grained fairness characteristics are unspecified. That is, probabilistic or weak protocols will only ever be used when explicitly specified.

This hierarchy serves as an example mapping of functional properties (and characteristics) to technical protocols, implemented in Chapter 5 to satisfy fairness and accountability requirements, acting as proof of concept of the approach to supporting multiple B2B support properties specified in this manner.

3.3.4 Partial Consistency Hierarchy

Consistency presents an interesting discussion with regards to provisioning it within intermediary support. Section 2.6.4 described the use of a three-way handshake on the outcome of an interaction. Specifically, the synchronisation takes place to ensure the outcome with the highest precedence (*TechFail* > *BizFail* > *BizSucc*) is the one that is propagated to the business level.

This section considers a partial consistency hierarchy as an example of a direction the intermediary support could be extended in. Table X defines the characteristics and Figure Z illustrates the hierarchy.

Characteristic	**Description**
Outcome	The outcome of an interaction must be consistent for all participants
Total	Participants must maintain a consistent state at all times during interactions

Table 3.5: Characteristics for the *Consistency* property

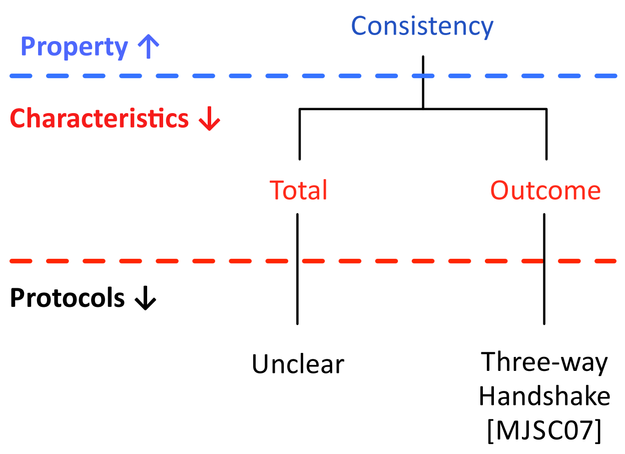


Figure 3.4: *Consistency* property hierarchy

The reason consistency is explored as an example property rather than implemented, as fairness and accountability are in Chapter 5, is that the approach specified by [MJSC07] cannot be applied in a decoupled manner to intercepted transmissions. That is, intermediaries would require that all messages be sent unencrypted, or could be decrypted and must also be able to discern which conversation is being executed and at what stage the current execution is. This also implies that intermediaries are aware of all possible conversations that can occur, where the aim of this thesis is to produce intermediaries that can support transmissions independent of their B2B standard including which conversation(s) are being executed with the optional ability to capitalise on conversation definitions to enhance functionality.

An approach that provided consistency that could be applied independently to B2B interactions, the characteristic of total consistency could be tackled by adding new points of synchronisation. For example, after every transmission or after every exchange. This would incur communication and computational overheads but could allow consistency to be maintained for participants during an entire interaction.

The development of suitable protocols to satisfy consistency is beyond the scope of this thesis, its consideration as a property is to indicate ways in which the proposed intermediary support could be extended where applicable protocols are available.

3.4 Declaration Mechanisms

This section discusses two mechanisms by which organisations can specify which available functional properties should be satisfied for their intercepted transmissions: predicates and annotations.

Declaration via predicates are defined as pairings of predicates and functional properties (and their characteristics). Intercepted transmissions are tested against the specified predicate to be if the paired functional properties should be satisfied for it. Section 3.4.1 and 3.4.2 provide generalised abstractions upon which the predicates may be constructed to match intercepted transmissions. 3.4.3 will demonstrate example predicates and discuss their application.

Declaration via annotation encapsulates a business transmission and annotates the desired properties onto the encapsulated transmission, the intermediary support will unpack the original transmission and delivery it to ensure business level transparency is maintained. Annotations are discussed in Section 3.4.3.

3.4.1 Generalised B2B Abstractions

The following abstractions are generalised from the surveyed B2B standards:

- Message Identifier
- Conversation Name
- Conversation Version
- Conversation Identifier
- Business Identity (minimally of sender and recipient)

- Business Role
- Business Requirement
- Raw Business Message
- Raw Conversation Definition

For all transmitted messages the following parameters are always unencrypted: *business identity of sender*, *business identity of recipient* and *message identifier*. These are required to facilitate delivery even if all remaining data and metadata is encrypted. All B2B standards analysed express this requirement and I assume it to be true for all messages under the message oriented middleware paradigm.

Conversation name and *version* apply to the definitions of conversations whereas *conversation identifier* is used to identify messages as belonging to an instance of a conversation. Where readable, this allows messages belonging to the same instance of a conversation to be correlated.

*Business role*s are generally expressed at the conversation level, they are used to indicate actions expected by each participant. In previous examples such roles have included 'sender' with 'recipient' and 'initiator' with 'responder'. They are identity-agnostic names for participants that allow conversations to be applied to any two participants fulfilling their roles.

*Business requirement*s are named requirements that are expressed either on conversations or individual messages, these may represent B2B standard independent requirements (e.g., non-repudiation of receipt) or reference a specific standard (e.g., ebXML non-repudiation of receipt).

Section 3.4.3 will use the abstractions to construct example predicates, illustrating how they are used to support B2B interactions.

3.4.2 Intermediary Support Abstractions

In addition to the generalised abstractions in the previous section, we specify the following abstractions:

- Property Name (with characteristics)
- Protocol Name
- Protocol Identifier (i.e., an identifier referencing a specific instance of any protocol execution)

Property name refers to any available support property the intermediary support is able to satisfy for intercepted transmissions (e.g., accountability), properties are declared (via annotation or predicates) by name as required for the matching interactions.

Protocol name and *identifier* are not used in the definition of predicates. However, these abstractions are used in Section 4.6 when discussing cross layer events. Specifically, it is useful to be able to map protocol identifiers to message and conversation identifiers, allowing automation and inference where appropriate.

3.4.3 Declaration via Predicates

By combining the abstractions from Section 3.4.1 and 3.4.2, declarations can be defined using predicates to specify when matched interactions require specific properties to be satisfied. Table 3.6 lists some example predicates and desired properties.

The final two examples represent inference using existing business level requirements. They state that any transmissions detected to specify generic or ebXML business level non-repudiation of origin should be provided with accountability with strong fairness guarantees. This follows earlier discussion about capitalising on business information where available to provider richer functionality (i.e., the inference of functional requirements from business level requirements).

Predicate	**Properties Required**
Any Message	*Accountability*
Recipient Identity = 'Unknown Organisation'	*Accountability(Fairness, Strong)*
Recipient Identity = 'Sister Organisation'	*Accountability(FullyTrusted)*
Conversation Name = 'SubmitPurchaseOrder'	*Accountability(Strong), Consistency*
Conversation Name = 'SensitiveExchange'	*Accountability(Strong, Transparent), Consistency*
Business Requirement = 'Non-repudiation of Origin'	*Accountability(Strong)*
Business Requirement = 'ebXML Non-repudiation of Origin'	*Accountability(Strong)*

Table 3.6: Example declarations using predicates

These predicates are intended to be evaluated quickly, allowing the desired properties to be determined as quickly as possible and satisfied in the most appropriate manner. For multiple predicates matching a transmission the intermediary support has the option to attempt to combine the associated properties in the best manner possible (e.g., strongest requirements take precedence) or simply allow multiple different protocols to execute (e.g., a protocol providing weakly fair accountability and a protocol providing strongly fair accountability).

These declarations should be stored within an intermediary acting on behalf of an organisation. For example, Int_A stores declarations checked against all of A's transmissions. That is, all transmissions from A can be checked against the declarations within its intermediary to determine if support properties are required. This keeps support transparent (i.e., predicates are matched against intercepted transmissions within an intermediary) and decoupled (i.e., the predicates are written separately from B2B transmissions and can be constructed using standards-independent abstractions such as recipient identity). The implementations in Chapter 5 contained pre-programmed predicates for deciding when fairness and accountability should be satisfied. A production implementation would most likely use a rule-based evaluation system such as JBoss Drools. Future work in Chapter 6 will discuss possible refinements to predicate based declarations.

3.4.4 Declarations via Annotation

Declaration via annotation involves encapsulating an entire transmission and annotating the container with the desired properties that should be satisfied by intermediary support. Within the intermediary support the desired properties could be separated from the original transmission, the required support rendered and the original transmission delivered onwards to its intended recipient. Encapsulation of the entire original transmission is preferred as annotating desire properties into a transmission may not be possible (e.g., if the contents are encrypted) or may be obtrusive (e.g., affect the generation of signatures and digests).

Specifically, for business transmission whose entire contents are represented by $\{data\}$, annotation encapsulates the transmission inside a technical level message whose contents are $\{desiredProperties, \{data\}\}$. For some transmission of $\{data\}$ from $A \rightarrow B$, it would be encapsulated as $\{desiredProperties, \{data\}\}$. Int_A would separate $\{data\}$ and satisfy the *desiredProperties* before delivering it onwards to B. B receives $\{data\}$ and continues its normal operation.

Declaration via annotation is useful for its ability to specify required properties on a per-transmission basis. Declaration via predicates, stored within an organisation's intermediary, will generally apply to multiple transmissions (i.e., any that match the predicate).

Declaration via annotation does place specification requirements on an the transmitting organisation, which may be considered to break levels of transparency. However, declaration by annotation still specifies functional properties. The specified properties are satisfied by intermediary support and this is preferable to interacting organisations having to implement and integrate technical protocols into their own stack.

3.4.5 Asymmetry and Decoupling of Declarations

Declarations via predicates and annotations, as discussed in the previous sections, are defined on a per-participant level. Declarations via predicates will be stored within an intermediary acting on behalf of an organisation with the ability to be match against that organisation's incoming and outgoing transmissions. Declarations via annotations allow an organisation's individual outgoing transmissions to specify a list of properties to be satisfied for that transmission.

The above mechanisms facilitate asymmetric predicates for organisations. For example, A may fully trust B but B may not trust A at all. This is represented by A and B having separate sets of declarations (within their respective intermediaries and on their own transmissions).

By storing declarations within the intermediaries acting on participants' behalf, and ensuring encapsulation where properties are annotated on to transmissions, the declarations are entirely decoupled from a transmission's contents (including which B2B standards it uses) but are still able to capitalise on transmission information where available.

3.5 Discussion: Supporting Individual Transmissions

This section discusses the ways in which individual transmissions can be supported including examples of providing fairness and accountability and potential considerations for optimisation.

Consider a one-way exchange pattern, as discussed in Section 2.8.2, 2.9.2 and 3.1, in which a business message is sent including its business level non-repudiation of origin evidence, in response to this message a business signal is generated and transmitted containing the business level non-repudiation of receipt for the original message.

We define *bizMSG* to represent a business message, *bizNRO* to represent non-repudiation of origin evidence (at the business level, such as ebXML non-repudiation evidence), *bizSIG* to represent a response signal and *bizNRR* to represent non-repudiation of receipt evidence. Figure 3.5 shows two possible conversations in which a message is exchanged for a signal with optional non-repudiation evidence attached to the respective transmissions:

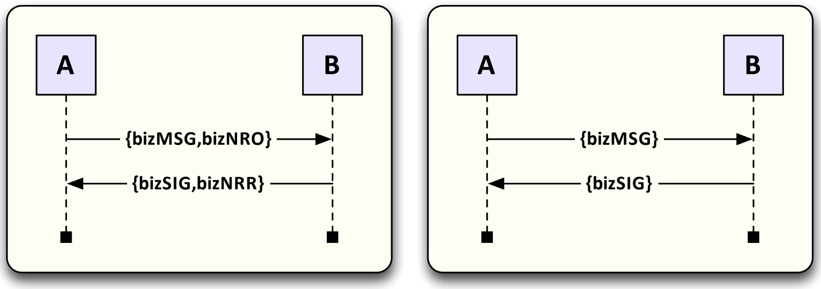


Figure 3.5: Business exchanges, possibly including non-repudiation evidence.

Both conversation consists of two transmissions ($A \rightarrow B$ and then $B \rightarrow A$). After the first transmission in either conversation, B has an advantage over A in that it can choose not to transmit *bizSIG* or *bizNRR* (i.e., selective receipt) and A has no recourse. What is demonstrated here is that the most granular level of support required is for a single transmission. Taking accountability as an example, there must be proof to show that A initiated a single transmission and proof that B accepted it.

To this end, we define the contents of a transmission to be arbitrary (and opaque) data such that $data_n$ represents the data contained in the n^{th} transmission. In the figures above, $data_1$ would represent $\{bizMSG, bizNRO\}$ or $\{bizMSG\}$. Following this, we define $iNRO_n$ and $iNRR_n$ to represent non-repudiation of origin and receipt (respectively) of the n^{th} transmission. The i prefix on these evidence types denotes that they are generated at lower-levels, by the intermediary support (where the *biz* prefix indicates business level elements).

By considering the example intermediary accountability support in Section 1.3, Figure 3.6 illustrates the exchanges between A, B, Int_A, Int_B and TTP to render the transmission of $data_1$ from A to B accountable (by executing the Coffey-Saidha non-repudiation protocol between Int_A and Int_B, engaging TTP):

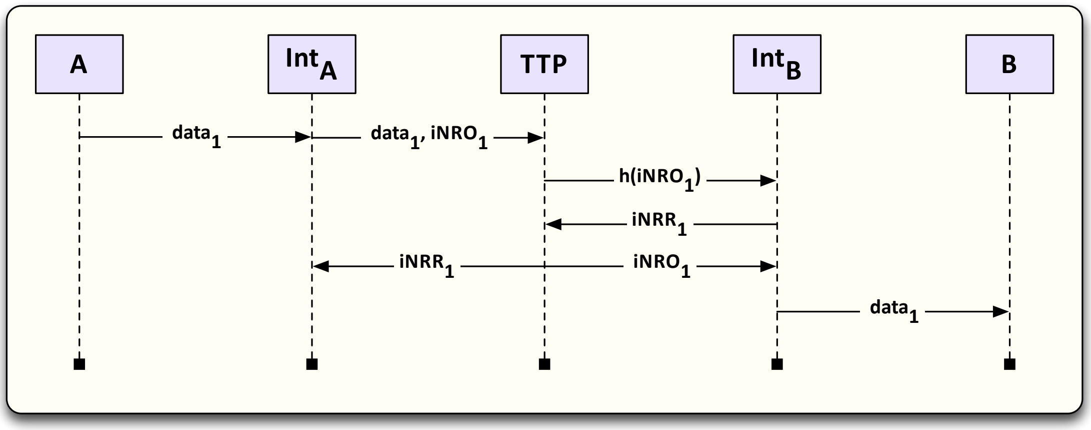


Figure 3.6: A single transmission rendered accountable by intermediary support.

The composition of Int_A, Int_B and TTP form the 'Intermediary Accountability Support' abstraction as defined in the example in Section 1.3. All communication between Int_A, Int_B and TTP is the execution of protocols designed to satisfy the required properties. In this example accountability is satisfied by executing the Coffey-Saidha non-repudiation protocol, fully specified in Section 5.2.

The outcome of this flow is $data_1$ is transmitted from A to B and Int_A, Int_B and TTP have copies of *iNRO₁* and *iNRR₁* proving irrefutably that A originated the transmission and B accepted receipt of its contents.

It follows then that $data_2$ represents $\{bizSIG, bizNRR\}$ or $\{bizSIG\}$ as in the second transmissions from the conversations in Figure 3.5. The flow illustrated in Figure 3.6 could be repeated by exchanging $data_2$ to yield evidences $iNRO_2$ and $iNRR_2$ for the second transmission.

Assuming both transmissions between A and B occur successfully, business level transparency is achieved and they both receive the contents they expect (*bizMSG*, *bizSIG* and possibly *bizNRO* and *bizNRR*) without it being observable that Int_A, Int_B and TTP intervened in the transmissions to provide stronger accountability guarantees. Similarly, unfair outcomes are prevented from occurring: evidence exists that could demonstrate irrefutably that B received $data_1$ but did not respond with $data_2$ and B is unable to initially acquire $data_1$ without $iNRR_1$ being proffered on its behalf. Chapter 4 discusses the actions that can be taken in the face of non-compliance at the technical level. For example, should the transmission of a message from A to B be aborted if B won't proffer $iNRR_1$ even if this breaks transparency?

This illustrates that by providing support for each individual transmission (i.e., transmission 1 is supported by $iNRO_1$ and $iNRR_1$ and transmission 2 by $iNRO_2$ and $iNRR_2$), support for the entire exchange has been achieved. This applies again to exchanges, by supporting each exchange, entire conversations can be supported.

This means the intermediary support operates agnostic of the semantics of both the transmission and the contents being transmitted. That is, it doesn't matter if a transmission stands alone, is part of an exchange or part of a conversation, all supported properties are available for all transmissions and all supported properties can be satisfied regardless of transmission contents (i.e., the protocols can all operate while considering data to be opaque). Interestingly, this potentially allows transport protocols to be changed during a conversation's execution and also supports asymmetry in transports used by both participants (i.e., A may transmit to B using a different transport than B uses to transmit to A).

Following this, the analysis of B2B standards may seem somewhat redundant. However, the analysis yields that where the contents of a transmission are available (i.e., unencrypted), the service can inspect the contents to support its execution. This allows the intermediary support to be able to correlate protocol executions to transmissions,

transmissions to exchanges and exchanges to conversations. Such correlations are not required but may be extremely beneficial to the retrieval of data from Int_A, Int_B and TTP (e.g., A can query for all evidence relating to a single conversation identifier).

While the example above uses the Coffey-Saidha non-repudiation protocol for accountability satisfaction, the other non-repudiation protocols guaranteeing fairness discussed in Section 2.6.3 are all equally suitable since the contents of $data_n$ can be completely opaque to the protocol in use. However, Coffey-Saidha provides suitable clarity for the flow diagram in this example.

3.5.1 Side Discussion: Optimisations

As hinted at above, and discussed in Section 2.10, the intermediary support may inspect the contents of $data_n$ where unencrypted and use this information to make more informed execution decisions. An application of this is optimising the application of Figure 3.6 to the conversations in Figure 3.5.

The first possible optimisation we will consider is optimising up through layers of abstraction. That is, it may be possible to replace $bizNRO$ with $iNRO_1$ and $bizNRR$ with $iNRR_1$.

The second kind of optimisation considered is optimising down through the layers of abstraction. That is, it may be possible to replace $iNRO_1$ with $bizNRO$. It is not possible to replace $iNRR_1$ with $bizNRR$ since $bizNRR$ is not generated until the transmission of $data_2$. $iNRR_2$ cannot be replaced with $bizNRO$ as $iNRR_2$ is proof for B where $bizNRO$ is proof **for** A.

Both of these optimisations aim to deduplicate evidence exchange during interactions. However, they are protocol dependent, the above optimisations work specifically for Coffey-Saidha but are not possible for the other non-repudiation protocols used in this work. Optimising up through layers of abstraction may break business level transparency if the evidence is generated differently from how a B2B standard specifies. Optimising down through layers of abstraction may be possible but when replacing lower level evidence, we must ensure the replacing evidence is sufficiently strong (e.g., for non-repudiation evidence, it must be witnessed by a *TSA* to protect against key revocation).

Another optimisation may involve not providing the transmission of $data_2$ with accountability guarantees since its contents simply inform A of $data_1$'s receipt and *iNRR₁* serves the same purpose. This requires the intermediary support to have sufficient business level understanding to correlate the transmission of $data_2$ as a response to $data_1$.

An extreme optimisation, following the previous example, may be that transmission of $data_2$ to A can be completely abandoned as $iNRR_1$ serves as sufficient proof. This would likely result in breaking business level transparency, the trade-off is whether saving the transmission of $data_2$ is worth it with the additional complexity that at least Int_B must receive $data_2$ and be able to discern that its transmission should be abandoned (i.e., it relies on Int_B being able to view the contents of $data_2$ and also reason about them by both correlating the transmission as a response to $data_1$ and determining that it is suitable not to transmit $data_2$ as a response).

Generally these optimisations come down to a minimal saving in overhead versus a big increase in the complexity of understanding required to implement them.

For the case of optimisation via evidence replacement, evidences can only be treated as redundant if they provide the same strength of guarantees. That is, $iNRO_n$ and $iNRR_n$ use *TSA* witnessing to protect against key revocation where *bizNRO* and *bizNRR* may not. In this case, no evidence generate at lower levels is redundant. If the evidences do have equal strength, then there are only two pieces of redundant evidence generated (some $iNRO_n$ that matches *bizNRO* and some $iNRR_n$ matching *bizNRR*).

For such small overheads on top of conducting fair accountability, it is worth incurring the small overhead for an overall larger gain in the simplicity of treating all transmissions individually and transmission data opaquely.

3.5.2 Side Discussion: Fairness for Individual Transmissions

As introduced in Section 2.5.4, a single business transmission is the smallest guaranteed unit of communication between two interacting participants, not an exchange. To guarantee fairness, the transmission must be executed as some kind of exchange at lower levels. Furthermore, fairness only serves to protect during an exchange, accountability is required to protect after an exchange and ensure agreements remain enforceable.

The abstraction of a single business level transmission is maintained by saying that for the transmission of $data_n$ from $A \rightarrow B$, the intermediary support guarantees the fair exchange of $\{data_n, iNRO_n\}$ for $iNRR_n$ with guarantees that $iNRO_n$ and $iNRR_n$ remain valid in the future.

3.5.3 Side Discussion: Semantics of Accountability Evidence

This section illustrates that treating the contents of a transmission as opaque allows all interactions to be supported even when the contents are secured. The business semantics of accountability evidence generated for a single transmission are also of no concern to the intermediary support. The contents could include a message belonging to an instance of a conversation or the transmission or agreement of new conversations, by providing evidence and being able to guarantee its fair exchange in a semantic agnostic manner, interacting organisations are able to use the evidence to whatever ends they please.

For example, by combining proof of the origin and receipt of a transmission, and demonstrating the transmission's contents were a message belonging to an instance of a conversation, the evidence generated by the intermediary support can be used to demonstrate compliance with a given conversation, allowing dispute resolution where required.

3.6 Discussion: Predicate Declarations

This section demonstrates the use of declaration via predicates to infer when accountability should be satisfied using RosettaNet business level requirements. Firstly we consider an existing RosettaNet conversation: RosettaNet PIP 3A4 (or 'RequestPurchaseOrder') and describe the submission of a purchase order, acknowledgement of its receipt and subsequent acceptance or rejection based on the validity of the contents of the purchase order.

Figure 3.7 includes a snippet of the conversation definition from which predicates to facilitate inference are established. The XML snippet indicates that non-repudiation of origin and receipt are required for both the requesting message ('PurchaseOrderRequest') and the response message ('PurchaseOrderConfirmation'). Lines 4, 5, 9 and 10 mark the exact location of parameters specifying the requirements[1].

```
 1  <ProcessSpecification name="3A4">
 2      <RequestingBusinessActivity
 3          nameID="InitiatePurchaseOrderRequest"
 4          isNonRepudiationRequired="true"
 5          isNonRepudiationReceiptRequired="true" />
 6
 7      <RespondingBusinessActivity
 8          nameID="InitiatePurchaseOrderConfirmation"
 9          isNonRepudiationRequired="true"
10          isNonRepudiationReceiptRequired="true" />
11  </ProcessSpecification>
```

Figure 3.7: An excerpt from the XML definition of PIP 3A4.

This constitutes a specification at the conversation level that messages require non-repudiation evidence. It is also possible that individual RosettaNet messages contain the same attributes in their XML definition. That is, we can tell if a RosettaNet message requires non-repudiation by either (a) identifying an intercepted message as belonging to

[1]XML elements and attributes beyond the scope of the example have been omitted for brevity.

a known RosettaNet conversation specifying the requirement for non-repudiation evidence or (b) detecting the inclusion of the non-repudiation parameters of a RosettaNet message itself.

Section 2.9.2 described three kinds of headers included in every RosettaNet message, the *service header* contains information including which PIP is in execution and a unique identifier for this particular instance of that PIP. That is, a conversation name, version and identifier as described in Section 3.4.1. Figure 3.8 contains an XML snippet of a service header for a message belonging to an instance of the RosettaNet PIP 3A4 conversation.

```xml
<ServiceHeader>
    <ProcessControl>
        <pipCode>3A4</pipCode>
        <pipVersion>1.2</pipVersion>
        <pipInstanceId>12345</pipInstanceId>
    </ProcessControl>
</ServiceHeader>
```

Figure 3.8: An excerpt from an example RosettaNet Message Service Header in XML.

As per RosettaNet's specification, the service header may be encrypted. However, where it is readable, reasoning can be established to drive inference. Specifically we can specify predicates to infer when accountability is required. The following list contains two examples of using predicates to establish when the declaration should apply:

1. By conversation name
    a) if *Conversation Name* = *'3A4'* then require *Accountability(Strong)*
2. By business level requirement
    a) if *Business Requirement* = *'non-repudiation of origin'* then require *Accountability*

b) if *Business Requirement = 'non-repudiation of receipt'* then require *Accountability*

Both cases rely on abstractions that can be extracted from known business level information. That is, intermediary support must be able to extract the information from transmissions (i.e., sufficiently unencrypted transmission elements) and know how to match these.

For point (1a) the intermediary support must understand where a conversation's name is defined (Line 1 in Figure 3.7) and similarly, which parameter in a message indicates the PIP and instance thereof it belongs to (Lines 3 and 5 in Figure3.8).

For points (2a) and (2b) the intermediary support must understand what parameters specify non-repudiation of origin and receipt for a message (Lines 4, 5, 9 and 10 in Figure 3.7).

For points 1 and 2, participants can be provided with additional accountability guarantees with no extra specification cost on their part. That is, the declarations exist within intermediary support and can infer the additional requirements for any intercepted transmission where sufficient information information is available. In short, once the declaration is defined within an organisation's intermediary, all (or a part of) their transmissions may be supported with no extra specification effort on their part.

Naturally, there are costs in terms of additional computation, storage and timing (e.g., the generation, exchange and storage of evidence) but, importantly, the business level continues to operate as normal while still being provided with potentially stronger levels of support. Section 5.5.5 empirically evaluates the impact of one of the implementations on typical business conversations by calculating computation and communications overheads and comparing them to existing delays and conversation lifespans.

3.6.1 Side Discussion: Opaque Transmissions

While the example declarations in the previous section rely on the contents of a transmission being unencrypted and understandable, it is still possible to support opaque transmissions by limiting predicates to using only sender identity, recipient identity and message identifier. For example:

- All messages require *Accountability(Fairness, Strong)*

Referencing the hierarchy in Figure 3.3, we can see that this specifies accountability refined by the available characteristics. In our hierarchy, this makes any protocol satisfying the 'strong' characteristic of accountability suitable, specifically, Coffey-Saidha, Zhou-Gollman, Kremer-Markowitch or Markowitch-Kremer. Importantly, this allows strongly fair accountability to be provided to all outgoing transmissions from an organisation regardless of their contents.

The requirement that *sender identity*, *recipient identity* and *message identifier* is mandated by the use of message oriented middleware. That is, a message must expose enough information to facilitate delivery its intended recipient(s).

3.6.2 Side Discussion: Extracting Knowledge versus Raw Content Matching

The predicates within the declarations specified in Section 3.6 rely on the abstractions exposed by the intermediary support, these abstractions are enabled by the intermediary support's ability to understand and extract information relating to specific B2B standards. That is, the intermediary support knows how to identify elements such as business level requirements and conversation name for specific standards such as ebXML and RosettaNet.

An alternative for specifying the same declarations relies on raw content matching on conversation definitions and messages for predicates. Using the abstractions provided, the following declaration was previously defined:

- if *Conversation Name = '3A4'* then require *Accountability*.

We can redefine this using raw content matching as the following:

- if *Raw Message contains*: ("`<pipCode>3A4</pipCode>`") then require *Accountability*

In the latter version, the intermediary support does not need to be able to reason about the B2B contents of the intercepted transmission although the entity creating the declaration does (to create the pattern described the contents to be matched). Accountability will be satisfied for any message whose contents are unencrypted and found to contain the desired contents. A benefit to this approach is that messages belonging to B2B standards not understood by the intermediary support can still be supported, relying on content matching as opposed to knowledge extraction.

All of the declaration via predicate examples demonstrate the ability of the intermediary support to capitalise on available information regarding the transmissions it intercepts, enriching the functionality where suitable.

3.7 Summary

Section 3.1 and 3.2 generalised message exchange patterns and common requirements of the surveyed B2B standards. These generalisations allow the smallest unit of supported business interaction to be determined (a business transmission) and highlighted how fairness, accountability and consistency are particularly suitable concerns to support via transparent, decoupled intermediaries.

Section 3.3 specified hierarchies of properties for addressing concerns of fairness and accountability with a discussion for how consistency could be supported in the future. The hierarchies included varying characteristics and were mapped to technical protocols providing their satisfaction under these characteristics. The implementations in Chapter

5 will realise a subset of the accountability hierarchy (Figure 3.3) to provide fairness and accountability support for intercepted transmissions.

Section 3.4 discussed predicate and annotation mechanisms for declaring when supported properties should be satisfied for intercepted transmissions. Section 3.5 and 3.6 demonstrate the general applicability of the proposed support to all B2B interactions, situations in which business level knowledge may be capitalised upon and even that the predicates and individual support transmission other types of message oriented interactions (beyond B2B) to be supported. Elements of this chapter are referenced in Chapter 4 in discussions about design decisions (trade-offs) and events that cross layers of abstraction (e.g., technical events that have a consequence at the business layer).

4 Designing and Discussion of Intermediary Support

Section 4.1 and 4.2 begin by discussing the positioning of the support with regards to each participant's B2B stack and the layers of abstraction within the intermediary support's own stack, used to determine and provide the required support. Section 4.3 discusses communication with and within the intermediary support including details of interception and assumptions for communication between components of the intermediary support.

Section 4.5 through 4.7 discuss design decisions relating to how support is offered to participants and how to deal with events that may require notification or may cross levels of abstraction and the associated trade-offs.

Section 4.8 and 4.9 discuss components and their role within the intermediary support. Section 4.10 discusses configurations for the deployment and composition of these components. Section 4.12 demonstrates that related work can be expressed as instances of composition and deployment within the proposed generalised design and Section 4.13 details the instances that will be implemented in Chapter 5.

Section 4.14 summarises the design and discussions in this chapter, discussing how their contribution (i.e., how they address the challenges outlined in Section 1.5).

4.1 Conceptual Middleware Positioning

This section discusses the conceptual positioning in relation to general B2B stacks as discussed in Section 2.2. This will illustrate how support can be rendered transparently to interacting organisations at lower levels, independently of the B2B standards in use by supporting interception and provision of support at the transport level. Figure 4.1 illustrates the transmission of a message from A to B through general B2B stacks with zero intervening intermediaries.

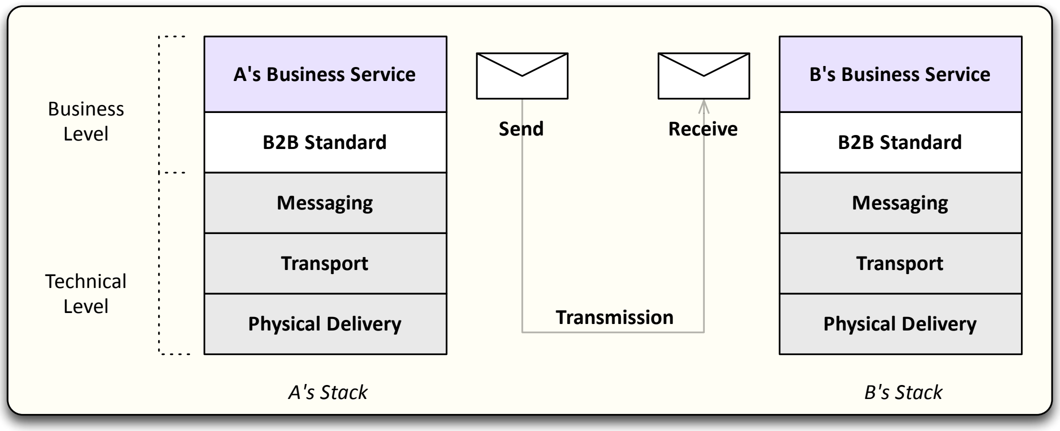


Figure 4.1: Transmission from $A \rightarrow B$ passing through their B2B stacks.

As per Section 2.2, a business message (or signal) is created by A's business service. This creation occurs at the business level, the business transmission has some business semantics and may specify some business level requirements. A then dispatches its transmission for delivery, incurring the use of specified message and transport formats and protocols, abstracted away by the business level.

Based on work in Section 3.5, intermediary support must render the support it offers to transmissions at the transport level if it is to do so transparently and independent of the B2B standards in use. This is achieved by placing intermediary support in the delivery path of B2B interactions, requiring that intermediary support exposes compatible transports and can determine the sender, recipient and transmission (or message) identifier.

By also considering that intermediary support may read the contents of outgoing transmissions (and messages) and that certain transports may require certain types of physical delivery, the support intermediary can be considered to be its own stack as shown in Figure 4.2.

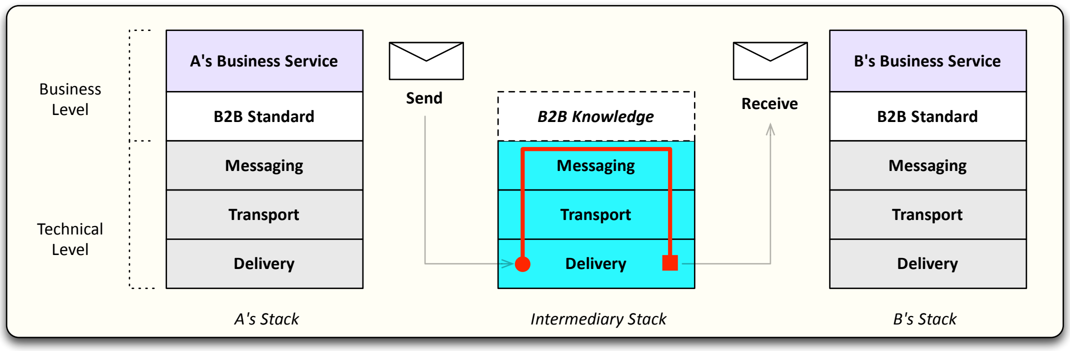


Figure 4.2: Intermediary support's positioning within the general B2B stack.

The blue stack depicts elements of the intermediary support. The solid red line indicates where the intercepted transmission deviates from its normal delivery path with regards to its delivery from A to B. It is important to note that the entire intermediary stack, and elements thereof can be distributed between A, B and service providers. It is simply useful for this discussion to consider the support offered as an intermediary stack through which transmissions can be routed. Critically, the grey line indicating transmission from A to B leaves A's and enters B's in the same places as in Figure 4.1.

Figure 4.2 illustrates that the intermediary support does not replace any layers within the B2B stack, it provides a complete parallel support, offering the transport layer such that compatible transmissions could be intercepted, but transmission without intermediary involvement is still supported.

While shown in the intermediary stack, the B2B knowledge layer is not required to be involved in interception, it represents the ability to use available business information in intercepted transmissions to drive capabilities such as business level requirement to functional requirement inference as discussed in Section 3.6. Similarly, the messaging layer may have minimal involvement such as being used to determine only the sender, recipient and message identifier.

The delivery layer element of the intermediary is present as specific transports may mandate the use of specific delivery standards, formats or protocols (e.g., TCP, UDP, IPv4 or IPv6). Intermediary support may be required to use obey such requirements to ensure business level transparency is maintained.

As an example, intermediary support illustrated in Figure 4.2 could support all RosettaNet and OTA interactions by supporting interception HTTP, HTTPS and SMTP transports, MIME, S/MIME and SOAP messages and TCP, UDP and IPv4 delivery standards. Declarations could be developed based on the recipient of messages such that all of an organisation's outgoing messages could be supported by fairness and accountability, even when encrypted or part of an unknown conversation.

4.2 Middleware Layers of Abstraction

Given the intermediary support's positioning within the B2B stack, this section defines internal layers of abstraction used to facilitate its functionality and discusses them in the order that they are generally encountered during normal execution of supporting B2B interactions. Specifics regarding communication with and within the intermediary support are discussed in detail in Section 4.3. Figure 4.3 illustrates the abstract layers within an intermediary acting on behalf of an organisation (e.g., Int_A).

The flow as illustrated in Figure 4.3 shows an intercepted transmission ($data_n$) passing through the conversation and exchange pattern knowledge layer that is able to extract and reason about the contents of a transmission, the declarations and properties layer is responsible for determining which properties (if any) are required for the intercep-

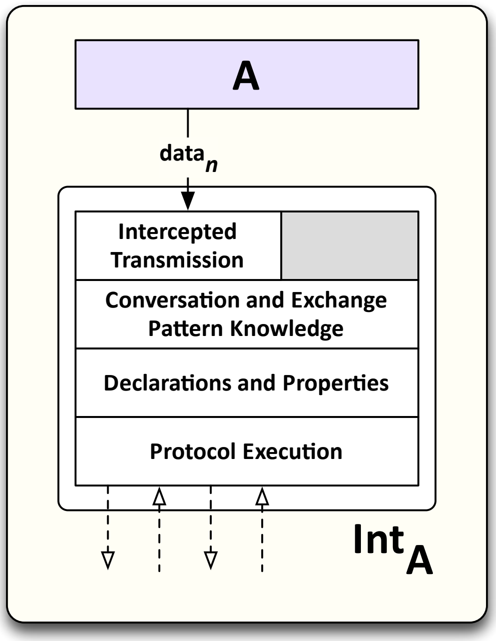

A
data$_n$
Intercepted Transmission
Conversation and Exchange Pattern Knowledge
Declarations and Properties
Protocol Execution
Int$_A$

Figure 4.3: Layers of abstraction within Int_A.

ted transmission and finally the protocol execution layer executes protocols, interacting with another organisation's intermediary (e.g., Int_B) and security services to satisfy the required properties. At the recipient's end, their intermediary will ensure business level transparency is maintained (e.g., Int_B will maintain business level transparency for B by ensuring $data_n$ is passed to the business level unaltered, as if delivered directly from A).

Of note here (and also demonstrated in the fair accountable transmission example in Section 3.5) is that a single business level transmission (e.g., $data_n$ transmitted from $A \to B$) may result in multiple transmissions back and forth between Int_A and Int_B in the execution of the required technical protocols. Critically to ensuring business level transparency is that preservation of the abstraction of a single business level transmission from $A \to B$ regardless of the underlying execution. As in previous examples, the stack based approach with layers of abstraction is similar to layered network stacks.

Section 4.2.1 through 4.2.3 discuss the three abstract layers in detail.

4.2.1 Conversation and Exchange Pattern Layer

The conversation and exchange pattern layer (or MEP layer) is responsible for extracting and interpreting the contents of a transmission where available (i.e., unencrypted). It will always determine values for the sender identity, recipient identity and message identifier and attempt to determine values for all other abstractions defined in Section 3.4.1 (e.g., conversation name) to be used when determining which declarations using predicates apply to an intercepted transmission.

The MEP layer contains instances of domain specific message exchange patterns (i.e., conversations in a specific B2B standard) and also of more general message exchange patterns (i.e., transmissions, exchanges, one/two-way actions as described in Section 3.1). Knowledge of these patterns is used to enhance the functionality of the intermediary support where possible. For example, when a transmission can be identified as part of an exchange or conversation, the intermediary support can aid consistency support (i.e., trigger synchronisation at the end of a conversation to ensure a consistent outcome). The ability to correlate related protocol executions, transmissions and exchanges and conversations can improve the retrieval of evidence from intermediary support.

It is assumed that two interacting organisations agree to use identical (or compatible) versions of conversations defined in their desired B2B standard and that these versions are also known by the intermediary support (specifically by the organisations' intermediaries). It is reasonable that intermediaries could expose functionality by which their organisation could upload and update conversation definitions to be used in the course of supporting B2B interactions. Future work in Chapter 6 discusses possible developments relating to mutually agreed updates and the signing of conversations and agreements along these lines.

The MEP layer is only capable of providing support for the B2B standards, conversations and general exchange patterns it is aware of. However, it may still intercept transmissions in known message and transport formats for B2B standards that it is not aware of. In these cases there are still two possibilities by which intermediary support can be configured to provide additional properties to these transmissions:

1. Declarations via annotation. With known transport protocols, it is possible to encapsulate the contents of a transmission and annotate them with required properties to be satisfied. By being able to function while treating the contents of any transmission as opaque, intermediary support can support arbitrary transmissions including B2B standards it can not reason about and general (i.e., non-B2B) transmissions.

2. Declarations via predicates, specifically using raw content matching. An organisation can define declarations within their intermediary that perform raw content matching on intercepted transmissions. The content matching relies on the knowledge of the organisation to express their desired requirements but enables them to specify ad-hoc predicates matching content that cannot be reasoned about by the intermediary support.

In summary, the MEP layer exists to determine minimally required information (i.e., sender, recipient, message identifier) and, where possible, capitalise on higher level and domain specific knowledge to better inform the lower levels of execution in supporting B2B interactions transparently.

4.2.2 Declaration and Properties Layer

The declaration and properties layer contains the available functional support properties (and maps them to technical protocols), the defined declarations using predicates and is able to determine the overall required properties for an intercepted transmission (i.e., the combination of annotated requirements and requirements specified using predicates).

For example, when Int_A intercepts a transmission $A \rightarrow B$ it will firstly check if the contents (i.e., $data_n$) are annotated with requirement declarations. These annotated properties are added to a list of properties that must be satisfied for this transmission. Following this, the layer will check the contents of the transmission against defined predicates, for all satisfied predicates the declared requirements will also be added to the list of required properties for this transmission. After these operations, the layer can invoke the correct technical protocols (in the protocol execution layer) to satisfy the required properties (and their given characteristics).

It may be possible to optimise on the required list of properties to satisfy once they have all been determined. Optimisations in this case may apply simple heuristics such as "the strongest requirement wins" where characteristics such as "strong, optimistic and transparent fairness" would win out over lesser characteristics (e.g., just "strong and optimistic"). Without any optimisations the intermediary support will simply execute all protocols required to satisfy all of the properties that have been declared as required. Common sense will be applied such that the same technical protocols are not invoked twice for the same transmission.

4.2.3 Protocol Execution Layer

The protocol execution layer is responsible for the execution of technical protocols to satisfy the required functional support properties (and their characteristics). The intermediary support will invoke the protocols based on the results of the previously discussed declaration and properties layer. The layer contains all available technical protocols for execution and the components required to execute them by communicating with other intermediaries acting on behalf of organisations and required security services.

Section 3.4.2 defined the notion of a protocol identifier. This is some sufficiently unique identifier that can be used to identify individual instances of protocol execution within the intermediary support. This allows the association of protocol identifiers with business level elements where possible. If a conversation identifier is known, association allows all protocol executions related to the conversation to be referenced or vice versa. Similarly, protocol identifiers can be associated with message identifier to facilitate referencing. These identifiers and their association will be referenced in Section 4.6 to automate the handling of events crossing layers of abstraction.

A final note about the protocol execution layer is that it may be useful to allow organisations to configure parameters affecting the execution of technical protocols. Examples of these parameters would be acceptable timeouts or the maximum number of retries in the fact of exceptional behaviour. Reasons for allowing this configuration will be discussed in Section 4.5 through 4.7. Referring back to the discussion of using synchronisation protocols to obtain consistency for B2B conversations in Section 2.6.4. It was said that in the face of temporary failures, it may not be possible to guarantee consensus in a distributed asynchronous system but that this risk can be arbitrarily reduced by increasing the timeouts allowed in synchronisation protocols. Allowing the configuration of the protocol layer gives an organisation the ability to configure these timeouts for satisfactory performance.

4.3 Communication With and Within the Intermediary Support

This section discusses communication with the intermediary support (i.e., facilitating interception) and within it (i.e., how do security services within the intermediary support communicate with each other). Discussion in Section2.8, 2.9, 3.1 and 4.2 shows that by exposing specific transports, transmissions from an organisation can be intercepted and supported. Section 4.3.1 discusses some assumptions and considerations for facilitating interception by intermediaries.

4.3.1 Interception and Encapsulation of Transmissions

It is assumed in this thesis that conversations are executed via the asynchronous delivery and processing of messages. That is, we adopt a message oriented middleware (MoM) view of communication by organisations in which messages are asynchronously deposited in to endpoints and subsequently processed by some recipient or other delivery intermediary [Tan03]. For example, the transmission of a message from $A \rightarrow B$ constitutes the depositing of some message on a queue from which B will subsequently claim and process it. Thus, interception under MoM is quite easily facilitated by redirecting the flow of the message leaving A. For example, a message from A can be deposited on a queue Int_A who collects it and executes protocols to provide additional guarantees before the message ends up deposited within Int_B. From Int_B the message can be deposited on B's queue for normal processing and both A and B's business level remain unaware that redirection or extra support took place at lower levels. Figure 4.4 illustrates the examples.

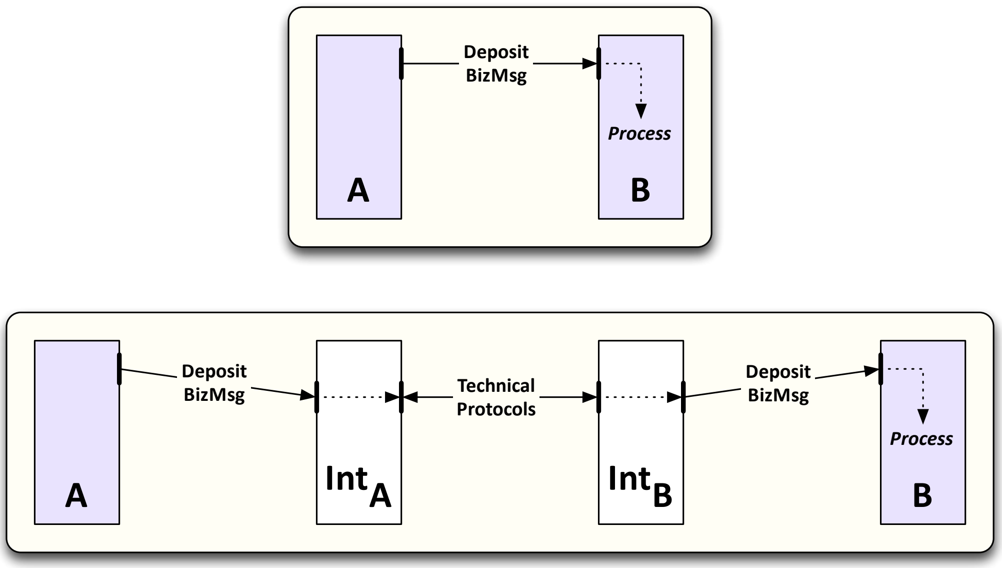


Figure 4.4: Intercepting the delivery of a messaging under MoM.

Figure 4.4 illustrates that participants and intermediaries may have multiple queues for different purposes. For example, A has a queue containing outgoing messages, Int_A and Int_B have a queue containing messages for and from A and B as well as an internal queues for technical protocol execution. The implementations in Chapter 5 adopt similar approaches for internal communication and facilitating interception with supported organisations.

As previously discussed, encapsulation under MoM is easy to achieve, a message can be considered the combination of some payload and its meta-data. Encapsulation can be achieved by treating the entire message as the payload in a larger container message whose meta-data constitutes the declaration of require properties for this transmission. This ensures at the recipient side, the original message is unpacked unaltered to be passed to the recipient organisation. Such encapsulation (and the annotation of required properties) can occur at the technical level within an organisation before the message is transmitted. Chapter 5 discusses some caveats with encapsulation under specific transports (e.g., HTTP and SMTP encapsulation must use transparent proxies as intermediary endpoints). In all cases, the abstraction of a single transmission from sender to intended recipient is maintained regardless of the amount of additional support provisioned at lower levels.

4.3.2 Communication within the Intermediary Support

Communication between components within the intermediary support will be done exclusively using the MoM paradigm. Specifically using message, queue and topic abstractions. Messages are deposited onto queues and topics who facilitate different modes of communication.

A message may only be received from a queue by one recipient. That is, there may be multiple nodes reading from a queue but a message will only be received and processed by one of these[1].

A message deposited onto a topic may be received by zero or more subscribers to the topic. That is, every subscriber will receive a copy of a message deposited on to a topic.

Message delivery and processing is asynchronous, meaning all components of the intermediary support need not be online at the same time although, as per the assumptions in Section 2.4.2, they are assumed to eventually come online to process queued messages.

4.4 Assumptions and Compatibility Between Participants

This section discusses compatibility considerations between organisations. Specifically, how to deal with situations in which organisations do not employ the intermediary support designed in this chapter. In this thesis we assume that both organisations agree to employ compatible middleware intermediaries to aid in supporting the execution and regulation of their interactions. However, it is still useful to consider how to handle situations in which this is not always possible.

4.4.1 Both Participants employ Support

The first scenario considered is that both participants employ my intermediary support. This completely facilitates the provision of support for all available requirements. Just as organisations agree to use specific B2B standards such as RosettaNet or ebXML, it is assumed they would agree to use intermediary support to provide them both with additional guarantees.

4.4.2 One Participant employs Support

The second scenario considered is one in which a single participant employs my intermediary support but the other does not (e.g. they run a plain B2B RosettaNet or ebXML

[1]For high throughput applications there may be instances in which a message is accidentally retrieved from a queue by more than one recipient, message identifiers are used to prevent the message being processed multiple times in this case.

stack). Under this scenario, the important decision is what to do in the face of being unable to provision additional support. Under this scenario there are two choices.

Firstly, interaction could be abandoned with the participant not employing intermediary support under the decision that where stronger guarantees cannot be provided, the potential loss of accountability or fairness make it too risky to continue. However, this approach makes it likely that business level transparency will be compromised when an organisation must be informed why all of their interactions with a specific participant fail.

The second choice is simply to continue without being able to provision any extra support, this approach has the benefit of maintaining transparency and the intermediary support is capable of “vanilla” execution by definition. That is, if no additional properties are required then regular execution takes place anyway.

4.4.3 Neither or Other Support

It is beyond the scope of this thesis to deal with scenarios in which alternative additional support for B2B interactions is provisioned and trivial to deal with situations in which no additional support is provisioned (i.e., two vanilla B2B stacks interacting normally as per their chosen standard).

4.5 Intermediary Impact upon Deadlines and Timeouts

The execution of one or more technical protocols at lower levels naturally has an associated time cost. This may impact business level deadlines and timeouts. For example, if a technical protocol takes too long to terminate, a delivery deadline may be missed and a business level retransmission would occur. Obviously, implementation will place a premium on efficient implementation to attempt to minimise the time required for technical protocols to execute.

Due to the possible encryption of a transmissions contents, intermediary support cannot reliably determine the deadlines for business transmissions. This renders scheduling or prioritising of protocol execution potentially wasted effort. Similarly, it is impossible for the intermediary support to controller higher level elements of processing. That is, the intermediary support may execute as fast as possible only for another intermediary beyond our control to delay the delivery and cause a deadline to be missed.

For these reasons, configuration of the protocol layer as discussed in Section 4.2.3 is considered particularly useful as it allows an organisation to place their own reasonable restrictions on the length of time protocols are allowed to execute for before timing out (e.g., All non-repudiation protocols must terminate within sixty seconds). Beyond this, the execution of protocols could be profiled to determine average execution times which could potentially be factored into the definition of reasonable business level deadlines within a conversation.

4.6 Handling Events Across Layers of Abstraction

In supporting the execution and regulation of B2B interactions, there may be eventualities occurring that have impacts beyond their layer of abstraction. For example, if technical protocols are being executed and a business conversation is abandoned, the intended behaviour should be that all technical protocols associated with that conversation be terminated as soon as possible. Similarly, if the execution of a technical protocol fails, this may in turn cause a business failure to occur. This section discusses possibilities within the middleware design of the intermediary support for dealing with these events automatically and allowing participants to manually trigger specific behaviour.

4.6.1 Tracking and Automating Behaviour

Referring to the discussion in Section 4.2.3 regarding the association of protocol, message and conversation identifiers. Here we establish multiplicities for the relationships between all identifier types:

- **one** protocol identifier maps to **one** message identifier
- **one** protocol identifier maps to **zero or one** conversation identifiers
- **one** message identifier maps to **zero or more** protocol identifiers
- **one** message identifier maps to **zero or one** conversation identifier
- **one** conversation identifier maps to **zero or more** protocol identifiers
- **one** conversation identifier maps to **one or more** message identifiers

Not all of these mappings will always be available. That is, a conversation identifier cannot always be determined and mappings involving this abstraction may not exist. However, where any of these mappings exist, elements can be associated across both levels of abstraction (i.e., protocol executions at the technical level and messages and conversations at the business level). This allows behaviour to be inferred when events happen at either level. The following list contains examples:

- If a business level failure is detected for a given message or conversation identifier, any associated executing technical protocols may need to be informed of the eventuality (e.g., to trigger aborts or terminations)
- If a technical failure occurs, this may require corrective or preventative action to be taken about business level elements (associated by message of conversation identifier) such as aborting the execution of a conversation or preventing an exception from manifesting at the business level

There operations are an instance of capitalising on higher level knowledge to enhance the functionality of intermediary support. Only by determining this information and associating it can this type of functionality by automated to aid in the transparent support of B2B interactions.

4.6.2 Notification and Manual Behaviour Triggering

The previous section discussed using association of identifiers to respond automatically to failures at either level of abstraction. Another possibility is that an organisation may opt to simply be notified of these eventualities such that they can respond appropriately.

This would likely entail exposing points through which organisations can manually trigger (or inject) the resolution of technical protocols by providing a protocol, message or conversation identifier. These points of interaction can be accessed while still maintaining business level transparency. However, they do require effort on the part of an organisation in that they must subscribe to relevant notifications about the state of their interactions, understand how to respond appropriately to relevant notifications and how to trigger the desired behaviour within the intermediary (e.g., retry, terminate or abort). Future work in Chapter 6 discusses the integration of work specifically designed to monitor contracts (conversations and agreements modelled a specific way) and notify participants when violations occur.

4.7 When to Maintain or Relax Transparency

The previous sections on compatibility, impacts on processing and deadlines and cross layer events prompt a discussion the possible approaches towards maintaining business level transparency and how involved or aware an organisation is with regards to technical level execution. A motivating example not discussed explicitly so far is what action to take when a property declared as required for a given interaction cannot be satisfied? The following sections will discuss strictly maintaining transparency versus potentially relaxing it to better inform an organisation's business level processes using this example and others discussed so far.

4.7.1 Maintaining at all Costs

The first possibility discussed is maintaining transparency at all costs. A motivation for this approach is to shield all lower level details (including exceptions and failures) from the higher (i.e., business) levels. When considering eventualities such as cross layer events, and impacts upon processing and deadlines, intermediary support must be able to appropriately control respond to technical level events (e.g., taking corrective measures and possibly preventing their propagation upwards).

What constitutes the correct action to take is subjective, it could be argued that for eventualities such as an inability to satisfy a required property (e.g., strong fair accountability) can be handled either by allowing interaction to continue without additional guarantees or simply causing the interaction to technically fail (i.e., *TechFail*). The argument hinges upon whether a precedence be place on the additional guarantees (e.g., fairness and accountability) or an organisation continuing to engage in B2B without extra technical effort on their part. Causing an interaction to fail does not break business transparency in so much as the business level services continue to operate without modification. However, it is a consideration that a business service with intermediary support operating in this manner will simply try (unsuccessfully) to repeat the business transmission until retry limits or timeouts have been reached. Ultimately a decision must be made with regards to this trade-off.

The implementations in Chapter 5 opt to place precedence on the additional guarantees provided by intermediary support, treating the inability to satisfy a required property as a technical failure of a business interaction. This is strictly true in that the failure to execute protocols at the technical level has occurred although the grey area is in that the technical failure occurred *because* the intermediary support become involved in delivery and could not successfully execute the required protocols for whatever reason(s).

4.7.2 Relaxing for Benefits to Interacting Organisations

The second approach considers when business transparency may be broken. This may involve feeding notifications from intermediary support directly into the flow of business conversations. That is, a business service implemented by an organisation may choose to use notifications from intermediary support to inform or direct the execution of conversations. This approach would allow more descriptive errors to be passed to the business layer. For example, technical failures could be split into support failures (e.g., property cannot be satisfied) and technical failures (e.g., transmission timeout).

These intermediary support notifications (including potentially descriptive errors) could be consumed by the business level and integrated into business services and the flow of conversations. This could also potentially make the properties offered by intermediary support available as business level requirements directly within conversation definitions.

4.7.3 Summary

As previously stated, the implementations in Chapter 5 will act to maintain transparency, treat the inability to satisfy required properties as technical failures and allow these failures to propagate to the business level, potentially triggering business level retransmissions until retry limits or timeouts are reached or the required properties are satisfied for an intercepted transmission. It is possible that intermediary support could automatically retry specific technical level operations where doing so would prevent a technical failure propagating upwards and potentially causing a business interaction to abort, the argument for when to propagate versus when to mask an event from higher levels is a well discussed software engineering topic [SRC84].

Future work in Chapter 6 will discuss potential refinements of the available declarations to provide finer grained specification of required properties to an organisation while still placing a premium on simplicity.

4.8 Design of an Organisation's Intermediary

This section will decompose intermediaries acting on behalf of an organisation (e.g., Int_A) into the minimally required components in order for them to function in their intended capacity. Before breaking down the individual components of an organisation's intermediary, it is useful to consider the flow of execution for an intercepted transmission:

1. A transmission is intercepted by (or routed through) the intermediary support
2. If encapsulated, the intermediary support extracts the contents of the transmission from the annotated list of required properties for this transmission
3. The contents of the transmission are checked against the declarations (using predicates) stored within the intermediary to see if any additional properties are required
4. From the list of required properties, determine the correct technical protocols to execute
5. For each protocol to be executed, invoke its execution over the transmission's contents, through some handler
    - In the course of execution, the handler may communicate with other security services (e.g., TTPs)
6. If required, pass data to the business, preserving business transparency
    - Possibly emit notifications for interested parties about the events occurring within the intermediary support

This flow of execution allows the main components of an organisation's intermediary to be drawn out, listed in the following section.

4.8.1 Middleware Components

All communication between components is conducted through the use of messages, queues and topics as discussed in Section 4.3.

1. Endpoints through which an organisation's transmissions are intercepted and routed (incoming and outgoing).
2. Internal queues and topics connecting the components of the intermediary.
    a) Including a topic through which intermediary support can emit notifications, providing a known point of subscription to which interested parties can subscribe.
3. A component to extract the encapsulated incoming transmission from property annotations, if required
4. A component to determine which predicates the transmission matches and which properties are required
5. A component to determine which protocols satisfy the required set of properties
6. A component to invoke the required technical protocols on the intercepted transmission
7. For **each** technical protocol required, a component to handle that protocol's execution
    a) Each protocol handler may interact with required security services, other intermediaries and other TTPs
8. A component to deliver the transmission's unaltered contents on to the recipient if required, maintaining business level transparency

As a side note, where multiple protocols are invoked, they may be executed sequentially or in parallel, they subject data for each protocol execution will always be the opaque contents of the original transmission. That is, the protocols invoked will take action **about** the transmission's contents but never **upon** it, allowing parallel execution to occur and reducing execution time.

As an example of components to enable the execution of non-repudiation, the following list serves as an example:

1. A component acting as protocol handler for **Coffey-Saidha Non-repudiation Protocol** executions
   a) Interaction with a component acting as an inline TTP to guarantee fairness
   b) Interaction with a component to generate evidence on behalf of the organisation being supported (e.g., evidence representing A is generated within Int_A[2])
   c) Interaction with a component to store the evidence associated with a protocol run

In fact all non-repudiation protocol handlers (with the exception of Mistianis' probabilistic protocol and voluntary exchange) will require components to generate evidence, securely store evidence and act as a TTP guarantee fairness in its exchange.

There is a design decision as to whether a single handler is implemented to support all non-repudiation protocols or whether one handler per protocol be implemented. Owing to composition I opt to implement a single component for each available protocol that could be composed into a virtual "non-repudiation" component providing multiple protocols.

Figure 4.5 shows the flow and connectivity of components within an organisations intermediary. The figure uses organisation A and their corresponding Int_A as the example.

[2]This can be done with A remaining in full control of the evidence generation, discussed in Section 4.10.

It should be noted that components 3-5 in the above list are likely to be implemented as a single component whose responsibility is "determine protocols to execute" and declarations via annotation and predicates are simply part of this process.

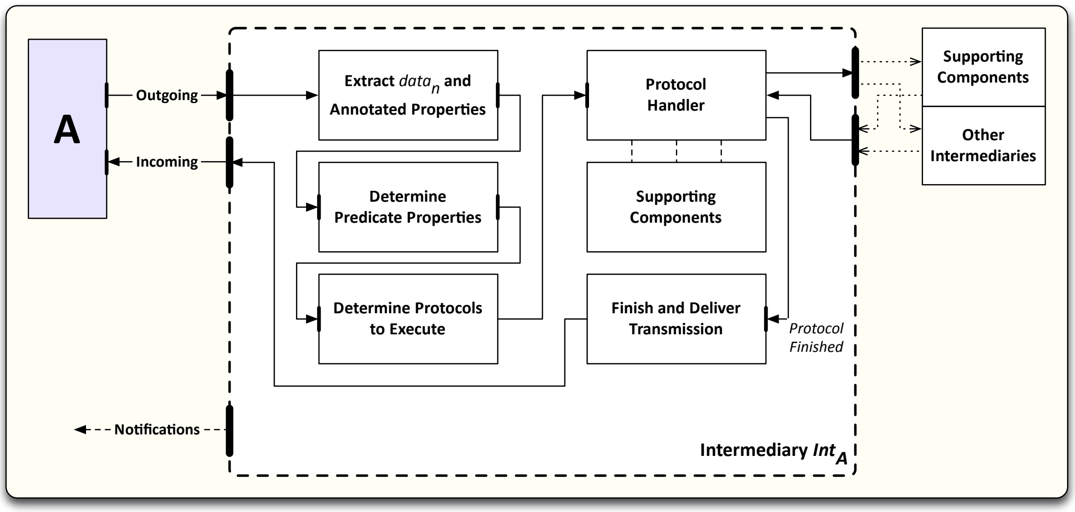


Figure 4.5: Connectivity of the components of an organisation's intermediary.

The notifications topic, seen on the bottom left of Figure 4.5 is a topic onto which any component of Int_A can publish events relevant to its operation. Connecting lines from every component are omitted for clarity. This design makes no assumption about where the components are hosted, Section 4.10 discusses the available configurations for deployment and how these satisfy different operational requirements.

Supporting components are shown both as part of the intermediary and externally to it. An example internal support component would be evidence generation or storage where an example external support component would be TTPs to guarantee fairness, TSAs to witness evidence or CAs and PKIs to facilitate public key cryptography.

4.9 Supporting Middleware Components

In addition to the components of an intermediary acting on behalf of an organisation, external supporting components exist. The composition of intermediaries (containing the necessary protocol handler components) and these external supporting components allow abstractions such as "transparent intermediary accountability support" (see Section 1.3) or "transparent intermediary consistency support" between interacting organisations.

The following list contains external supporting components that may be interacted with:

1. Certificate Authorities and Public Key Infrastructures to issue, verify, distribute and revoke identity certificates, public and private keys. This enables the use of public key cryptography.

2. TTPs to guarantee fairness for non-repudiation protocols

   a) As with protocol handlers within an organisation's intermediary, there may be multiple component types representing TTPs for different non-repudiation protocols.

3. TSAs to witness evidence generated by non-repudiation protocols

   N.B. TSA engagement will be online, not inline.

4. Secure logging or storage of evidence

   N.B. TTPs should always, where possible, log non-repudiation evidence in the process of protocol execution. Evidence storage **must** exist within an organisation's intermediary and **may** exist within the TTP. That is, it makes no sense for an organisation's intermediary not to store evidence relating to their interactions but it is not strictly required that a TTP also retains copies of all evidence it sees. Also, under offline TTP engagement, a TTP may never see evidence.

5. Identity Services (e.g., authorisation and authentication)

6. Consensus services

   N.B. Consistency support as described in Section 2.6.4 can support more than two participants through the use of a component deciding the outcome for all participants

These components can all be considered security services in the ecosystem described in Section 2.3, discussed by [BBMS01].

4.10 Composing Components to Provide Intermediary Support

This section will discuss the composition and deployment configurations that enable the intermediary support to satisfy the properties it offers and how the components of these compositions are deployed between one or more service providers and within organisations.

4.10.1 Composition to Enable Interception and Generalised Protocol Execution

Section 4.8 and illustrated Figure 4.5 describe and illustrate the minimum required components to facilitate the interception of a transmission between two organisations. For example, with only the components listed in the aforementioned section and figure, A and B would have their messages routed through Int_A and Int_B. However, without any protocol handlers no extra support can be provided. The internal processing by an organisation's intermediary can be considered to have three phases:

Pre-processing in which it is determined what protocols should be executed for the intercepted transmission

Protocol Execution in which the required protocols are executed to satisfy the properties declared for the intercepted transmission

Post-processing in which business transparency is guaranteed by ensuring only the original transmission is passed to the recipient organisation

Figure 4.6 illustrates this, providing a base from which accountability and consistency support will be described in the following sections.

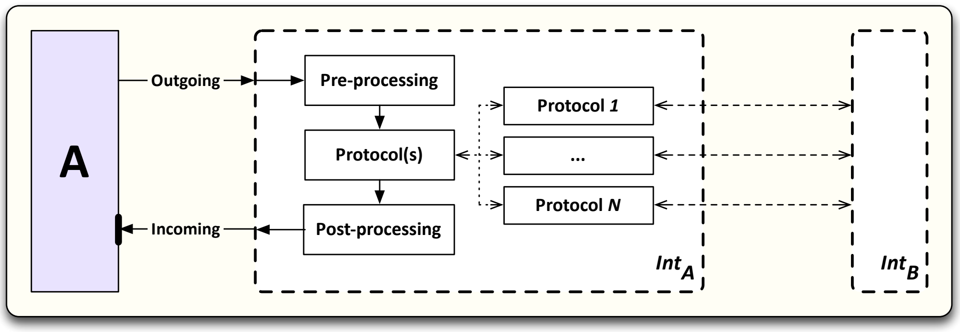


Figure 4.6: The components of Int_A interacting to execute protocols with Int_B.

4.10.2 Composition to Enable Fairness and Accountability Support

In addition to the components in the previous section, components supporting non-repudiation protocols such as those discussed in the hierarchy in Section 3.3.3 would be implemented as protocol handlers to enable Fairness and Accountability support. Non-repudiation protocol handlers would be co-located with $Protocol_1$, ..., $Protocol_N$ handlers shown in Figure 4.6. Support components providing evidence generation and evidence storage on behalf of a supported organisation would be required, their location determined by (de)centralisation of components and providers.

External support components must include TTP and TSA, the TTP may or may not employ its own evidence storage into which copies of evidence are reliable stored. These components would be involved in the protocol executions between Int_A and Int_B.

Assuming the above components are present, intermediary support can provide Fairness and Accountability support for intercepted transmissions.

4.10.3 Potential Composition to Enable Consistency Support

Consistency support, as described in Section 2.6.4 is achieved by including a synchronisation protocol handler within an intermediary acting on behalf of an organisation (e.g., Int_A). This synchronisation protocol handler acts as the conversation management layer (CML) from the referenced work [MJS06, MJSC07].

With the presence of the synchronisation protocol handler, intermediary support can offer and satisfy the *Consistency* property for intercepted transmissions.

Referencing Figure 4.6, the synchronisation protocol handler would replace one of the placeholder protocol handlers. For synchronisation involving more than two participants, an external consensus component would be engaged.

4.10.4 Summary

The accountability and consistency support described in Section 4.10.2 and 4.10.3 can be composed to provide intermediary support capable of satisfying accountability and consistency properties. Thus, supporting new properties within the intermediary support requires the definition of a new functional property, implementation of the necessary protocol handler(s) to execute protocols satisfying the property (and supporting components). Section 4.11 considers the potential locations for deployment of all components within one or more providers and within interacting organisations.

4.11 Deployment of Middleware Components

With the components and their compositions into intermediary support defined, the next consideration in the design is that of deployment locations. Specifically, restrictions regarding where components may be deployed.

4.11.1 Deployment within Supported Organisations

Two major restrictions relate to the deployment of components within organisations. Firstly, by definition, no part of any mutually trusted third party components (e.g., TTPs, TSAs, CAs and PKIs) may be hosted within an organisation participating in interactions that it wishes to support. These component must be completely independent of interacting organisations to ensure neutrality and prevent bias.

The second restriction relates to the degree of control an organisation wishes to retain over aspects of supporting interactions in which it is engaged. In the case of accountability support, there is a requirement that an evidence generation component be present. The non-repudiation protocols covered in Section 2.6.3 use the public and private keys (sensibly) of the participants to generate and validate evidence. An organisation may not wish to divulge private keys to a component such as this, acting on their behalf. In this case, they would host a security service representing the evidence generation component.

4.11.2 Deployment within Security Service Providers

Any of the aforementioned components can be hosted by security service providers (see Section 2.3). An entire intermediary support solution may be hosted by one provider (i.e., a centralised solution), by multiple providers (i.e., a decentralised solution), or split between one or more providers and interacting organisations. In the case of a centralised solution using a single provider, the provider must be trusted by, and independent of, both interacting organisations. Otherwise components cannot truly function as TTPs.

The deployment of components within multiple security providers (i.e., decentralised intermediary support) has the benefits of allowing organisations to choose a provider who offers some service(s) closely matching their desired requirements. The trade-off here is in complexity, interaction between all components must be well defined. Additionally, there may be situations in which parts of intermediary support offered by different providers different or incompatible protocols to satisfy their requirements. Possibly requiring some form of brokering and agreement on what additional capabilities are available for interactions between two supported organisations.

Message oriented middleware is a particularly useful communication paradigm in enabling the components of the intermediary support to span multiple providers and organisations and communicate the same regardless of this. That is, messages are consumed and deposited from and to arbitrarily addressed queues and topics which may be co-located with components of the intermediary support or hosted elsewhere. Thus, components are loosely coupled and asynchronous execution is allowed.

Providers and organisations may use cloud computing to operate their components as discussed in Section 2.11, allowing the components to operate at a cost of only the resources they consume (i.e., computation, storage and communication). Additionally, intermediary support (and the components thereof) can benefit from aspects such as scalability and elasticity. The use of open standards such as AMQP allow their backing to be provided by cloud providers, private providers or a combination thereof completely transparently to the intermediary support. That is, it is inconsequential what kind of provider a topic or queue identifier is backed by. Chapter 5 discusses these implementation focussed issues in greater detail.

4.12 Discussion: Previous Work as Instances of the Generalised Design

Previous work supporting B2B interactions discussed in Section 2.12 can be expressed as instances within the generalised design proposed in this chapter.

Cook's work [Coo06] can be implemented by the composition of protocols implementing the Coffey-Saidha non-repudiation protocol to be executed between interception components. Yao's work [YCW^{+}10] could be implemented completely as-is (that is, with no transparency) as a set of distributed consensus components into which business processes pass evidence allowing consensus to be reached. It is possible to adapt Yao's work and encapsulate its invocation inside transparent interception such that the service it provides is called while interacting organisations retain transparency. The second implementation in Section 5.4 adopts this approach of transparently invoking software as a service (providing fairness and accountability). FIDES work could be implemented by the creation of components to execute the FIDES non-repudiation protocol (providing strong and transparent accountability and fairness). Note that none of these works allow the capabilities they provide to be expressed (this is because they are designed to do one thing) as functional properties with characteristics as in this work. In order to actually implement previous work, the support they provide would have to be characterised in terms of functional hierarchies such as those discussed in Section 3.3.

4.13 Instances Chosen for Implementation

Using work from [Coo06] as a starting point, the first implementation provides a decentralised service constructed using JMS using no cloud hosting. Each intermediary acting on behalf of their organisation is hosted by their own provider and support components (e.g., TTPs and TSAs) are hosted by another provider. That is, five providers in total (A, B, Int_A , Int_B and support component provider).

The second implementation provides a centralised solution using cloud computing to satisfy all infrastructure requirements of the support service. All components of the intermediary support are satisfied by a single security service provider (independent to interacting participants). That is, all components of Int_A , Int_B and external support are hosted by a single provider. A and B both agree to use the centralised support for their B2B interactions.

The third implementation is a variant of the second in which timing measurements are taken for all aspects of support to ascertain the communication and computation overheads of providing additional support to B2B interactions and how these overheads compare to (and affect) the original interactions.

Beyond the implemented configurations, other useful instances include decentralised using cloud computing to allow variation in a choice of service providers and the potential benefits of cloud computing. Centralised without cloud computing offers solutions to situations such as those where compliance issues prevent the use of cloud computing (see Section 2.11.1). Similarly, decentralised or centralised approaches using a hybrid of cloud and otherwise hosted infrastructure can customise elements of the system as needed (e.g., transition them to or from the cloud) to cater to operational requirements possibly determined by compliance or other issues.

4.14 Summary

This chapter illustrated the conceptual positioning and layers of the intermediary support with regards to the general B2B stack. That is, distributed intermediary support components may be considered to be an intermediary stack for the purposes of supporting B2B interactions. The discussions in Section 4.1 through 4.7 relate to the challenges and considerations associated with achieving transparent, asynchronous, decoupled operation within the intermediary support.

Section 4.8 through 4.11 discuss components, their composition and deployment to form intermediary support middleware addressing varying operational requirements, followed by examples of achieving fairness and accountability support by composing specific protocol components with general interception components and declaration mechanisms to determine required properties. Section 4.12 demonstrates previous work also as instances of composition and deployment before Section d4.13 discusses the instances of composition and deployment chosen for implementation in Chapter 5.

5 Intermediary Middleware Implementations

This chapter discusses three implementations representing two instances of the possibilities discussed in Section 4.13. The first implementation is decentralised solution constructed entirely without the use of cloud computing providers. The service uses Java Enterprise Edition 5.0 (JEE) and the Java Messaging Service (JMS) standards to facilitate communication. The use of non-repudiation protocols allows the delivery of all JMS transmissions to be rendered accountable with fairness guarantees [MC09].

The second implementation is a centralised solution constructed using cloud computing for infrastructure requirements (i.e., computation, storage and communication). The solution is written in Scala [Ode10] and uses Amazon Web Services as the cloud services provider. Non-repudiation protocols are used to ensure the fair and accountable delivery of documents between between participants, the implementation was published in [MC10].

The third implementation is a variation on the second with timings for all computation and communication overheads. The locations of components are varied to demonstrate that the processing costs of the implemented support remain steady while the communication overheads vary depending on component locations. The timings are compared to typical B2B conversation deadlines and lifespans to evaluate the acceptability of the intermediary implementation.

The implementations will be evaluated on their functionality, how they improve over previous work and what the performance impacts associated with doing so. Section 5.6 will summarise the results and evaluations.

5.1 A Common Business Message Format

For purposes of clarity in this chapter we define here a simple messaging format using XML. Support for inspecting MIME, S/MIME and SOAP messages (such as those used by ebXML, RosettaNet and OTA) could be added seamlessly to the implementation. Figure 5.1 contains an example of the simple message format.

```xml
<Message
    senderID="org-a.com"
    recipientID="org-b.com"
    messageID="msg1">

    <BizInfo
        msgName="EXAMPLE_MSG"
        convName="EXAMPLE_CONV"
        convID="conv1"
        requireBizNRO="true"
        requireBizNRR="true" />

    <!-- Message Body -->
</Message>
```

Figure 5.1: A simple message example.

As with all surveyed B2B standards, we assume the entire contents of the `<Message>` tag in the above example may be encrypted and thus, unreadable by intermediary support. That is, only the `senderID`, `recipientID` and `messageID` parameters are always guaranteed to be readable.

Figure 5.2 shows an example business signal with a similar format.

As with the earlier message example, the contents of the `<Signal>` tag may be encrypted but `senderID`, `recipientID` and `signalID` parameters will always be available.

```
<Signal
    senderID="org-b.com"
    recipientID="org-a.com"
    signalID="sig1">

    <BizInfo
        signalName="MSG_ACK_RECEIPT"
        messageID="msg1"
        convName="EXAMPLE_CONV"
        convID="conv1"
        requireBizNRO="true"
        requireBizNRR="true" />
</Message>
```

Figure 5.2: A simple signal example.

There is no body in a signal message, the signal name in combination with the conversation and message identifiers provide context enabling a recipient to process the signal. The example in Figure 5.2 is a `MSG_ACK_RECEIPT` event applicable to some message with ID `msg1` in some conversation with ID `conv1`.

All business messages and signals in the implementations discussed in this chapter will use for formats defined in Figure 5.1 and 5.2.

5.1.1 Common Declarations via Annotations and Predicates

As discussed in Section 3.5 and 3.6, support offered by intermediaries will be applicable to opaque transmissions. However, where business requirements are readable, predicates can be created to declare when accountability should be satisfied by intermediary support. All implementations will support annotation of requirements onto individual transmissions (using encapsulation as discussed in Section 3.4.4) and will be programmed with the following predicate based declarations for the purposes of testing:

1. if *recipient* = *'org-b.com'* then require *Accountability(Fairness,Strong)*

2. if *business requirement = 'requireBizNRO' or 'requireBizNRR'* then require *Accountability*

3. if *conversation name = 'important_conversation'* then require *Accountability(Strong)*

Declarations 2 and 3 constitute using business level information to infer which properties the intermediary support should satisfy. Declaration 1 is applicable to all intercepted transmissions as recipient is always unencrypted information.

5.2 Coffey-Saidha Protocol Definition

The Coffey-Saidha protocol is used to provide accountability with strong fairness guarantees, Figure 5.3 contains the non-repudiation protocol definition [CS96]. The Coffey-Saidha protocol involves a *TTP* to guarantee fairness and *TSA* to protect evidence against key revocation (as discussed in Section 2.5.2 and 2.6.2).

1	Int_A	$\rightarrow$	TTP	:	$enc_{TTP}(id, A, B, data_n, NRO, ts_{TSA}(NRO))$
2	TTP	$\rightarrow$	Int_B	:	$id, A, B, h(NRO)$
3	Int_B	$\rightarrow$	TTP	:	$enc_{TTP}(id, A, B, NRR, ts_{TSA}(NRR))$
4	TTP	$\rightarrow$	Int_B	:	$enc_B(id, A, B, data_n, NRO, ts_{TSA}(NRO))$
5	TTP	$\rightarrow$	Int_A	:	$enc_A(id, A, B, NRR, ts_{TSA}(NRR))$

Figure 5.3: Coffey-Saidha non-repudiation protocol definition.

id represents a unique protocol identifier, meaning each protocol message contains the context of the protocol instance to which it belongs. *A* and *B* represent the identity of the participants on whose behalf the non-repudiation protocol is being executed. This protocol is implemented using constructs described in the following sections.

5.3 Implementation 1: Decentralised JMS

The first implementation is a decentralised solution where each intermediary acting on behalf of an organisation is hosted by their own provider alongside a single provider hosting external and trusted support components (i.e., TTPs, TSAs, CAs, PKIs). Figure 5.4 illustrates high level view of the implementation.

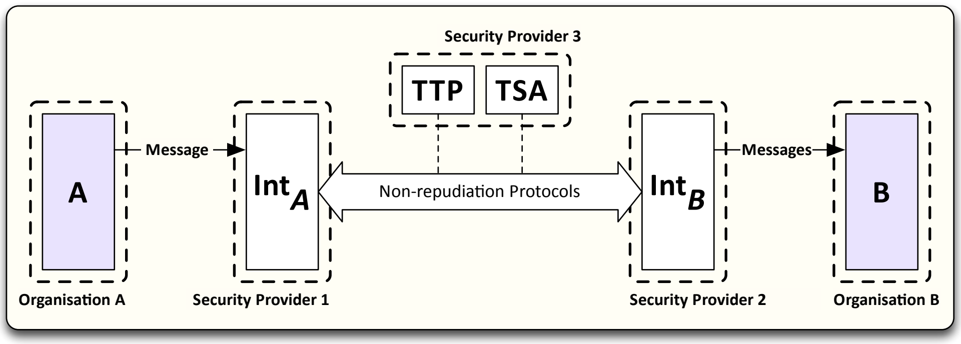


Figure 5.4: Int_A and Int_B collaborate on behalf of A and B to provide accountability to message delivery.

The implementation provides accountability with fairness guarantees through the execution of the Coffey-Saidha non-repudiation protocol, where required. By using transport level interception as discussed in Section 4.1 and 4.3.1, any JMS transmission routed through the intermediary support can be provided with fairness and accountability.

Int_A, Int_B and the components hosted by service provider 3 execute the non-repudiation protocol to generate, exchange and store evidence guaranteeing fairness and accountability. A and B exchange JMS messages whose contents are business messages and signals as discussed in Section 5.1.

5.3.1 Message Processors and Groups

The implementation is based around the notion of processors and processor groups. Processors are invoked with some intercepted message to take some action about it. Groups of processors are logically partitioned to form components. To facilitate loose coupling and asynchronous execution, processor groups are connected using JMS queues and topics. A processor group has four connected locations:

Input: A mandatory location specifying where messages to be processed by a processor group should be collected from.

Output: An optional location specifying where the intercepted, unaltered, message should be deposited once processing is complete.

Signal: An optional topic into which significant events during a processor group's execution are deposited for interested subscribers.

Audit: An optional topic into which extremely detailed messages about a processor group's execution are deposited, allowing its behaviour to be audited.

Figure 5.5 illustrates example processor groups and their interconnection using messaging queues. The figure omits signal and audit topics for clarity, their use is discussed in Section 5.3.2 and 5.3.3.

Intermediary support components, as described in Section 4.8 and 4.9, are implemented as processor groups.

The pre-processing group in Figure 5.5 is responsible for determining which protocols to execute, programmed with the predicate declarations discussed in Section 5.1.1 and capable of supporting annotation via encapsulating an entire JMS transmission within another JMS transmission. The processing group is responsible for executing the Coffey-Saidha protocol to satisfy fairness and accountability where required. The post-processing group is responsible for ensuring that the original business level transmission

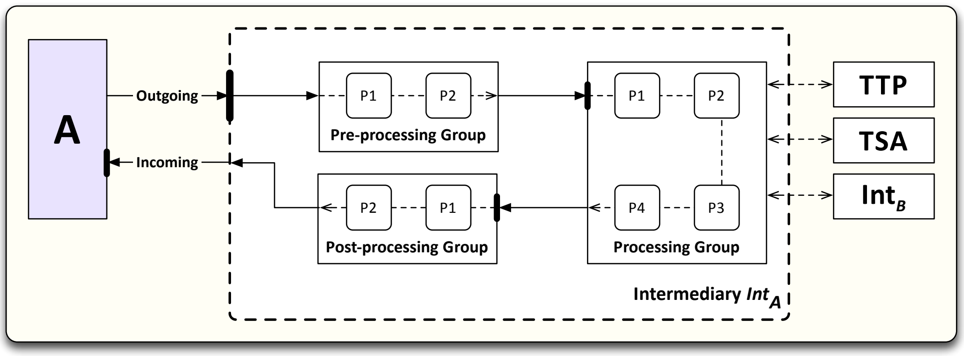


Figure 5.5: An example intermediary with three processor groups.

is passed upwards to maintain transparency.

Processor groups are connected using JMS queues and topics (thick black lines at the border of the organisation, intermediary and processor groups) although the processors inside groups may, where required, connect to arbitrary JMS queues and topics to deposit and retrieve additional messages.

5.3.2 Signal and Audit Topics

Signal and audit topics decouple processor groups from event notification and auditing concerns (e.g., logging). Processors simply emit events and audit messages and these are routed to their processor group's signal and audit topics if defined.

This implementation defines a single signal and audit topic for an organisation's intermediary. An authorised logger is subscribed to the audit topic to log all messages generated and sent during the execution of intermediary support. This logging will include all evidence generated and exchanged in the execution of the Coffey-Saidha non-repudiation protocol. Authorised parties may also subscribe to signal topics to be notified of significant events within the intermediary support (e.g., protocol invoked, protocol terminated and protocol failed).

Access control restricts the subscription to sensitive topics to prevent unauthorised

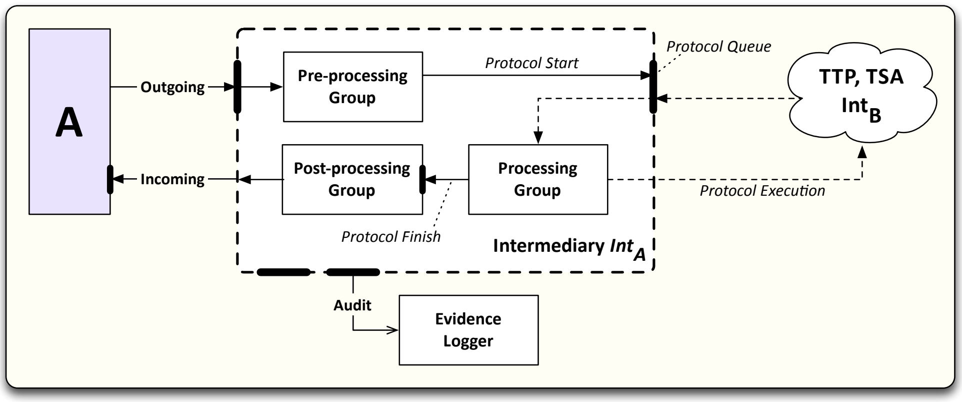


Figure 5.6: An evidence logger attached to the intermediary's audit topic.

parties from obtaining information such as non-repudiation evidence and prevent fairness from being compromised.

5.3.3 Fairness and Accountability Support

The support for fairness and accountability are implemented in three processor groups corresponding to the general areas outlined in Section 5.3.1.

NR-Pre-Process: This group determines if fairness and accountability are required. This is determined either by annotation (in JMS this is achieved by key-value pairs in the encapsulating message's meta-data) or by predicates programmed to implement the declarations specified in Section 5.1.1.

NR-Process This processor group executes the Coffey-Saidha non-repudiation protocol in cooperation with the recipient's intermediary (e.g., Int_B), a TTP and a TSA. If no non-repudiation is required the message will be delivered straight to the recipient's intermediary who will pass it through to the business level.

NR-Post-Process: Post-processing ensures only original transmissions are passed to the organisation. For example, a transmission from $A \rightarrow B$ will be passed to B's business level by Int_B's post-processing group once a non-repudiation protocol has successfully executed.

An evidence logger, as shown in 5.6 is implemented simply as a JMS subscriber to the audit topic that logs received audit messages into a MySQL database.

TTPs and TSAs involved in the execution of the Coffey-Saidha protocol are only ever concerned with technical level messages. As such, TTPs and TSAs are implemented as single processor groups who collect messages from an input queue and deposit their results on determined protocol queues.

For the TTP, the output destination depends on the current step within the Coffey-Saidha protocol (as per the definition in Section 5.2). The TSA's output destination will correspond to whoever used it to witness their evidence. That is, if A sends a message to the TSA to be witnessed, the TSA will deposit the witnessed message back to A.

5.3.4 Testing

The implementation was tested to ensure that it functioned as intended and also that is functioned under a variety of transmissions it could intercept. To this end, transmissions and evidence stored in the MySQL audit database will be compared to test data to ensure the correct evidence was generated, exchanged and stored providing fairness and accountability for the intercepted transmissions.

The test data includes various transmission contents to ensure:

1. That predicates function for opaque transmissions where only the sender, recipient and message ID are known

2. That predicates function for clear transmissions and can correctly use business level information

3. That support can be provided to both opaque and clear transmissions

Based on the above, the following transmissions were constructed for testing, using the message from Section 5.1 as a template:

1. An encrypted transmission whose recipient is 'org-b.com' to trigger predicate (1) in Section 5.1.1
2. An unencrypted transmission whose recipient is 'org-b.com' to trigger predicate (1) in Section 5.1.1
3. An unencrypted transmission expressing a business level requirement for NRO evidence to trigger predicate (2) in Section 5.1.1
4. An unencrypted transmission expressing a business level requirement for NRR evidence to trigger predicate (2) in Section 5.1.1
5. An unencrypted transmission whose conversation name is 'important_conversation' to trigger predicate (3) in Section 5.1.1
6. An encrypted transmission with an annotated requirement for *Accountability*
7. An unencrypted transmission with an annotated requirement for *Accountability*
8. An encrypted transmission that does not require additional support to be provided
9. An unencrypted transmission that does not require additional support to be provided

Testing the intermediary support with each of these transmissions and validating the evidence will address the points discussed at the beginning of this section.

For each of the above transmissions, those requiring support will result in the generation, exchange and storage of non-repudiation evidence providing fairness and accountability. The evidence is stored in a MySQL audit database as described in the previous section. This means for each transmission requiring support, the database will contain two rows:

1. A row containing the intercepted transmission, a protocol run identifier and TSA-witnessed NRO evidence

2. A row containing the intercepted transmission, a protocol run identifier and TSA-witnessed NRR evidence

The protocol run identifier for corresponding NRO and NRR rows will be identical as they are produced during the same protocol execution.

For all transmissions requiring support (1-7 in the above list), both associated rows were retrieved from the database and validated to ensure that they correctly represented the intercepted transmission. That is, the NRO evidence correctly represented the intercepted transmission and the NRR evidence correctly represented the NRO evidence and intercepted transmission.

Logging within the implementation verified that the correct predicates were being triggered for test transmissions 1-5 and that transmissions requiring support (1-7) were only delivered to their recipient upon successful completion of the Coffey-Saidha protocol, providing fairness and accountability.

5.3.5 Evaluation and Summary

Based on the testing in the previous section, this implementation is capable of intercepting any JMS transmission and providing it with fairness and accountability support, where required, by invoking the Coffey-Saidha non-repudiation protocol. The implementation allows the declaration of when fairness and accountability are required using both annotations and predicates against which intercepted transmissions are tested. The resulting evidence which is exchanged on behalf of interacting organisations and stored was correctly associated to the intercepted transmissions and the NRR evidence was correctly associated to its corresponding NRO evidence. While B2B interactions are supported such that predicates like those in 5.1.1 can be specified, any intercepted

JMS transmission can be supported and predicate support could be widened to extract information from messages belonging to other application domains.

In comparison with [YCW$^+$10] (Yao), this implementation provides stronger accountability guarantees by engaging with a TTP to ensure deterministic fairness and a TSA to witness evidence to address selective receipt and key revocation issues. The implementation operates transparently whereas Yao's implementation required business processes to explicitly emit accountability evidence. Both this and Yao's work operate asynchronously and independent of B2B standards in use.

In comparison with [Coo06] (WSNR) and [NZB04] (FIDES), this implementation is improved in its transparent and asynchronous operation. Compared specifically to FIDES this implementation improves by offering automated support that can be place in the delivery path of B2B interactions. WSNR and FIDES both provide deterministic strong guarantees but place integration requirements on interacting organisations. WSNR requires a SOAP interceptor to be integrated into an organisation's B2B stack and FIDES requires an manual workflow to be defined in which the proprietary FIDES clients are used to engage with FIDES servers to facilitate non-repudiable exchanges. WSNR and FIDES both execute synchronously, requiring all involved components (organisations and intercepting components) to be online at the same time.

All works, including this implementation, require some minimal set of trusted components to be hosted in a location independent from all interacting organisations. This work specifically considers decentralised operation as an example that an organisation can be supported by an external intermediary service provider acting upon its behalf in combination with other providers hosting trusted components (e.g., TTPs and TSAs) and intermediaries for other organisations.

On a more technical level, the design of processors and processor groups could be improved to support parallel execution of processors within groups and also to provide more flexible interconnection between processor groups. In this implementation a transmission flows through all groups even if those groups were to implement a protocol not required for the intercepted transmission. The design in Chapter 4 dispatches transmissions only to protocol handlers that are required to be invoked, the improvement was a consequence of this evaluation.

The use of JMS was problematic in a number of ways. First and foremost is that JMS defines no wire-level format for transmission. That is, two vendors implementing the same reference interface for JMS may not interoperate. The requirement that everyone uses the same vendor's (and version) implementation of JMS is not realistic in open heterogeneous networks. Beyond this, JMS has no standardised mechanism for remote addressing. There is no standardised way to pass a portable URL referencing a JMS queue or topic. Ad-hoc solutions exist, generally relying on Java Naming and Directory Interface (JNDI) standard to perform remote lookups. The JMS reference API also contains no standardised mechanisms for connecting to JMS services, it simply treats them as an implementation specific detail. As with the URL issue, ad-hoc solutions exist that abstract away vendor specific details but no standardised solution is available. As a final minor issue, JMS messages are not serialisable, this makes their use in operations such as signatures, encryption, hashes and digests an ad-hoc solution alongside URL schemes and dealing with remote or different JMS services. The use of JMS for heterogenous networks relies on additional support to marshall between vendor implementations.

In summary, the implementation addresses the requirements of intercepting transmissions and transparently supporting them. The support rendered is offered via functional properties (fairness and accountability), paired with technical protocols providing their satisfaction (Coffey-Saidha) and declared as required in a decoupled manner through the use of annotations and predicates. The MoM based communication renders the implementation asynchronous.

5.4 Implementation 2: Centralised, Cloud (AWS)

This implementation provides a centralised solution using cloud computing for all infrastructure requirements. Amazon Web Services (AWS) are used as the cloud provider for the following services: Simple Queueing Service (SQS), Simple Notification Service (SNS), SimpleDB Service (SDB), RelationalDB Service (RDS), Simple Storage Service (S3) and Elastic Compute Cloud (EC2).

The implementation is written in a combination of Scala and Java [Ode10]. Both languages compile to Java byte-code, Section 6.3 discusses the use of Scala (a functional language) and the possibility of leveraging it to provide domain specific APIs onto B2B and other conversational standards. Open Java libraries (e.g., Typica) are used to access Amazon Web Services.

All components of the intermediary support are satisfied by a single security service provider (independent to participants supporting their B2B interactions). That is, all Int_A , Int_B and other support components (i.e., TTP, TSA) are hosted by a single provider. A and B both agree to use, and trust, the same provider and the centralised support it provides. The security service provider is assumed to trust the infrastructure upon which their intermediary support operates (trust in the cloud is discussed in Section 6.3). Figure 5.7 illustrates the high level view of the system. As with the JMS implementation in Section 5.3, this implementation uses the message formats, predicates and protocols defined in Section 5.1, 5.1.1 and 5.2.

5.4.1 Interaction with the Service

Figure 5.7 illustrates the provision of components acting on behalf of A and B within the intermediary support (i.e., Int_A and Int_B). The centralised nature of the intermediary support means that it is not necessary to deploy a complete set of these components for each organisation. Instead, it can provision what is minimally required to allow an organisation their own point of interaction with the service.

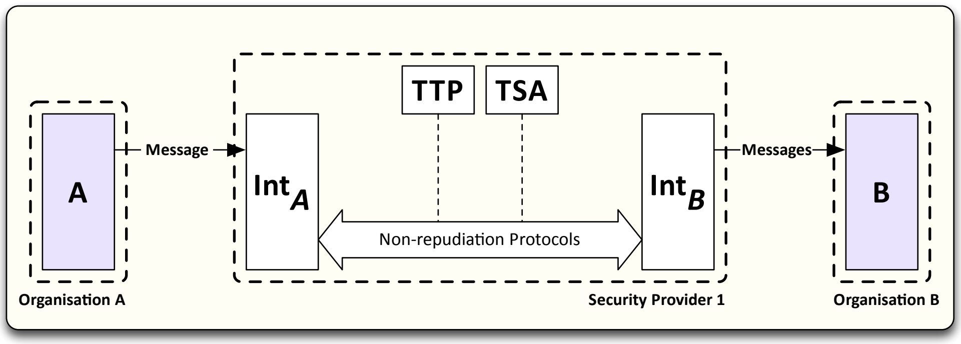


Figure 5.7: A single provider allows A and B to interact with accountability and fairness guarantees.

Since communication with and within the service is message based (using queues and topics), each user of the intermediary support is provisioned with the following:

Outgoing Queue: onto which outgoing messages are deposited

Incoming Queue: onto which incoming messages are deposited by the intermediary support

Notification Topic: allowing a user to subscribe to events generated by the intermediary support

Configuration Data: containing all predicate declarations and configuration for a particular organisation (e.g., cryptographic keys or acceptable timeouts for protocols)

Figure 5.8 illustrates the provision of queues and topics for organisations. Queue and topic names are prefixed with the identity of their respective organisations to prevent naming collisions and mechanisms exist to prevent unauthorised parties interacting with the queues and topics (either encryption of contents or the use of authentication and authorisation mechanisms).

Compared to the JMS implementation it may be noted there is now a single notification topic as opposed to an event and audit topic. This single topic employs *message prefixes*

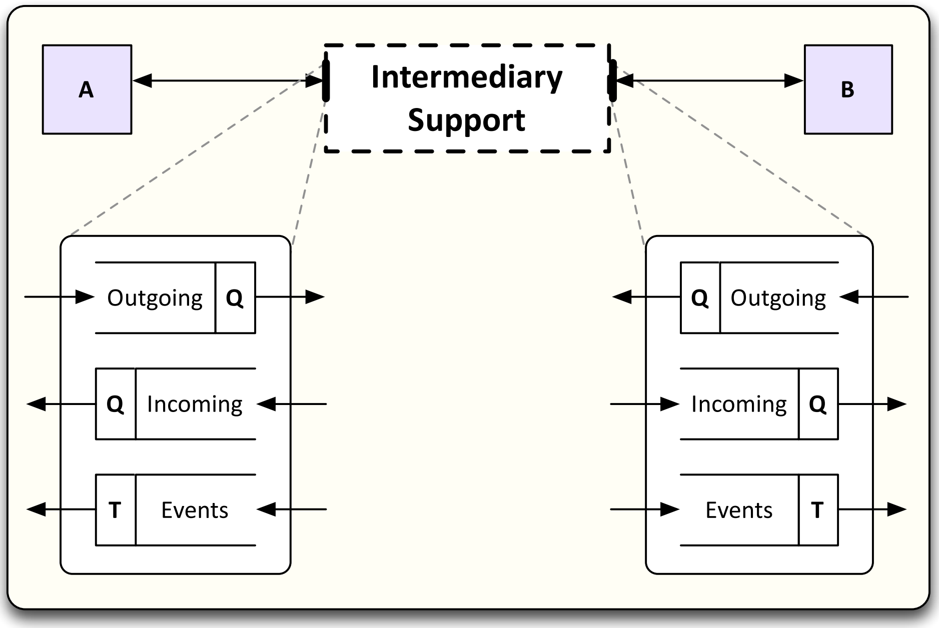


Figure 5.8: The provisioning of points of interaction per organisation as opposed to complete sets of components.

to provide a desired verbosity of messages received by a subscriber. That is, a receiver can choose to receive messages from a topic or queue with a given prefix. This approach is similar to logging architectures in which logs capture events at a desired levels of granularity (e.g., critical events, exceptional events, debugging information and complete trace information).

The centralised support uses reliable storage for all generated evidence and provides mechanisms for its retrieval by authorised parties, removing the requirement for a logging component to be attached to the notification topic to log all generated evidence.

Under Amazon Web Services, Simple Queue Service (SQS) and Simple Notification Service (SNS) are used to facilitate queue and topic based communication, respectively. SQS queues and SNS topics are addressed and accessed using HTTP. To this end, the implementation will intercept transmissions over HTTP and HTTPS using transparent HTTP(S) proxies.

5.4.2 Message Processing

Messages processors in this implementation present a simpler but more powerful refinement over the JMS implementation. Processors are not organised into groups, but the same behaviour may be achieved if desired. In this implementation a message processor is created with a single parameter, an incoming location (i.e., queue or topic) from which it should retrieve messages and engage in its execution. Under this implementation, processor groups, as in the JMS implementation, could be achieved by a linear arrangement of multiple processors.

Processors may query for global configuration parameters (e.g., configuration specified by the security service provider) or organisation-specific configuration parameters (e.g., configuration specified by users of the intermediary support). The output destinations of processors are dependent upon the context of their execution (i.e., their role, the identities of the involved participants from the intercepted transmission and available configuration parameters).

Critically, messages are not removed from queues until their processing is complete. They are flagged as “in use” by a receiving processor, preventing the same message being processed by other processors at the same time. As a message contains the complete context for its processing (e.g., a business transmission to be supported or a protocol message with protocol information), this allows message processors to arbitrarily crash and be recreated without a loss of messages within the system. Messages that are not removed from a queue will eventually have their “in use” marker removed and be processed. This also allows multiple processors (of the same functionality) to be subscribed to the same queue, providing horizontal load balancing to handle increased workloads.

5.4.3 Centralised Support Components

Figure 5.9 details the components (i.e., queues, topics, message processors and support components) of the centralised intermediary support.

The provision of queues and topics for each organisation maintain the abstraction of intermediaries acting on their behalf. Internally, however, messages may be processed from A and B's queues by the same message processor. This is a consequence of the centralised design, it is entirely possible to spawn individual message processors per organisation but not required in this implementation. This would also allow an easy migration to a decentralised deployment if desired.

The centralised design also facilitates the complete execution of non-repudiation protocol by a single processor. That is, the centralised support contains components capable of generating evidence on behalf of all interacting organisations.

The implementation does, however, also support evidence generation outside the centralised support, this is facilitated by providing a technical API to organisations which can be used to drive exchanges if desired. This API maintains business level transparency but does require an organisation to become involved in driving the execution of exchanges and reacting accordingly to generated events. It breaks the strictly centralised design of the support but the sensitive nature of divulging the required cryptographic keys involved in evidence generation make it a potentially useful feature, allowing organisations to retain control of their own private keys.

The *Work* queue shown in Figure 5.9 is fed by all queues that intercept messages outgoing from organisations. That is, the *Work* queue is an aggregation of all intercepted messages requiring non-repudiation protocol execution. Messages outgoing from A or B would go through the pre-processing message processor and, if required, be passed to the *Work* queue for processing by a non-repudiation protocol processor. It is also possible that a message not requiring any additional support is passed straight from pre-processing to post-processing for delivery to its recipient. The aggregation of messages awaiting protocol execution into a single queue allows the number of non-repudiation processors to be increased arbitrarily to deal with processor crashes and/or workload balancing.

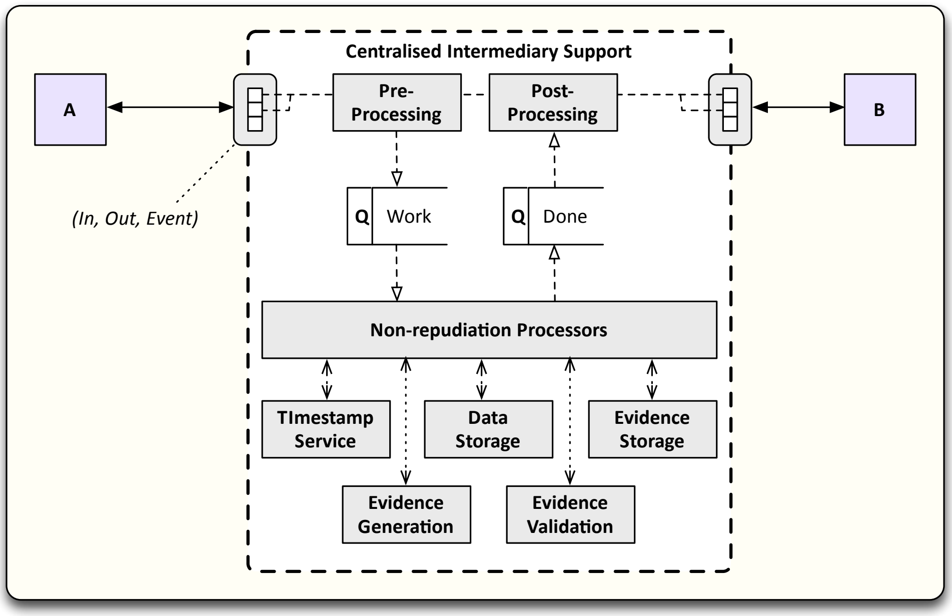


Figure 5.9: Exchange service internal components and message flow.

Similarly, the *Done* queue is an aggregation of all messages that have successfully had non-repudiation protocols executed for their contents. The messages are distributed to their respective organisations' queues.

The Amazon Web Services are leveraged in the following manner in the centralised intermediary support implementation:

Simple Queueing Service (SQS) queue based MoM communication

Simple Notification Service (SNS) topic based messages between nodes.

SimpleDB Service (SDB) global configuration and organisation-specific configuration storage and querying

RelationalDB Service (RDBS) SQL databases for storing logs and non-repudiation evidence

N.B. Non-repudiation evidence storage period can be configured per-organisation and defaults to 5 years

Simple Storage Service (S3) storing copies of the intercepted data unless configured not to

Elastic Compute Cloud (EC2) to host linux virtual machines executing all intermediary support components (Java 6, Scala 2.8)

Amazon was chosen as the cloud provider simply for ease of use, they are assumed to be sufficiently trusted as a cloud provider. All services apart from EC2 scale automatically. That is, SQS and SNS scale with increased communication demand, SDB and RDB will scale to increased database read or write activity and S3 will scale to meet increased storage demand. EC2 requires manual intervention to increase the power of the currently executing VMs or deploy new ones. Each deployed VM must also have its operating environment configured (i.e., Linux, Java and Scala).

5.4.4 Testing

Testing for the cloud based implementation was exactly the same as that described in Section 5.3.4. That is, the same 9 test transmissions were used to ensure correct functionality across encrypted and unencrypted transmissions where support is declared using annotations, predicates or not required at all. For each transmission, the two generated rows of evidence were extracted and validated to ensure the NRO and NRR evidences correctly associated with each other and with the originally intercepted transmission.

All transmissions were intercepted as required.

The first 7 transmissions had their required properties identified (i.e., the correct predicates were matched or annotations extracted) and satisfied by the intermediary support. Furthermore, messages were only delivered to their recipient upon successful completion of the protocols. Transmissions 8 and 9 were successfully passed straight through the intermediary support without traversing the protocol processors.

5.4.5 Evaluation and Summary

Based on the successful testing of this implementation, HTTP transmissions can be intercepted and provided with fairness and accountability support, where required, by invoking the Coffey-Saidha non-repudiation protocol. As with the JMS implementation, transmissions other than those in the B2B domain could be intercepted and supported.

This work is more in line with [YCW$^+$10] (Yao) than the JMS implementation. That is, this implementation provides accountability and fairness support in the cloud as a service. Additional components intercept transmissions between organisations and transparently invoke the service, where required, using the contents of the intercepted transmission. Such an improvement was discussed in Section 2.12 that would allow Yao's work to be invoked transparently. As it is, this implementation improves over Yao's support by transparently providing fairness guarantees and stronger accountability guarantees (i.e., the use of TTPs and TSAs to protect against selective receipt and key revocation). This implementation represents the same asynchronous, automated and transparent improvements over [Coo06] and [NZB04] as the JMS implementation.

The use of centralisation requires that interacting organisations fully trust the single service provider although the components could be decentralised as in the JMS implementation. The centralised intermediary support allows the expedited execution of the Coffey-Saidha non-repudiation protocol where all information is available within the service. As it is still possible for an organisation to retain control of evidence generation (through the use of the technical API), expedited execution may not always be possible.

The implementation leverages cloud hosting for all aspects. All cloud services used scale automatically to cope with demand except EC2 although services exist to provide automatic scaling of EC2 deployments. Compared specifically to the JMS implementation, cloud computing made the acquisition and configuration of resources easier and quicker.

The use of SQS and SNS mean that messages are not lost within the system in the face of processor crashes. New processors spawn and will eventually process messages that are flagged as re-available for processing.

The manner in which the support is provided (i.e., fairness and accountability as a service with components to facilitate transparent interception and invocation) prompt the consideration that the intermediary support in this thesis could also be realised as "Delivery as a Service", this is discussed in Section 6.3.

The use of Amazon Web Services for all services brings with it the advantages that Amazon do not charge for data that passes within their own services and they can provide improved quality of service (e.g., they guarantee instant delivery of notifications from EC2 instances to SNS topics and they guarantee delivery of messages from SNS topics into SQS queues). The other side of the argument is that by choosing a single provider for all infrastructure needs, a service provider is subject to all of their guarantees regarding availability, reliability, durability and performance.

In summary, as with the JMS implementation, support is successfully offered to intercepted HTTP transmissions in a transparent manner. Realised by the mapping of functional properties to technical protocols and declared as required through both predicates and explicit annotations on transmissions. The implementation improves on previous work by addressing the challenges of transparency, asynchronous operation, loose coupling and automated operation.

5.5 Implementation 3: Centralised, Cloud (AWS) with Timing Measurements

The JMS and AWS Cloud implementations discussed in the previous sections were tested (in Section 5.3.4 and 5.4.4) in terms of their functionality. That is, they correctly identified when support was required and provided it in the correct manner (ensuring delivery after non-repudiation protocols had executed). This implementation modifies the centralised implementation discussed in the previous section to remove logging statements allowing accurate timing data to be collected.

Finely grained timing data will be taken for communication and processing such that overheads for each can be calculated. From these timings we will extrapolate the total impact of intermediary support for a variety of configurations (e.g., centralised, decentralised and over varying locations) and compare them to deadlines and lifespans for typical B2B conversations.

This implementation will be timed using two transmissions:

1. An unencrypted transmission expressing a business level requirement for NRO evidence to trigger predicate (2) in Section 5.1.1
2. An unencrypted transmission with an annotated requirement for *Accountability*

These transmissions declare the same requirements from intermediary support (i.e., *Accountability*) using both predicates and annotations. Both transmissions will have a fixed size of 512 kilobytes, this is slightly larger than the largest text-based messages seen in conversations specified in [OAS01b, Ros09, Ope11b].

This implementation will consider two locations for the operation of example business services (i.e., the services whose interactions will be supported) and two locations for the operation of the centralised intermediary support. For business services, servers located in Newcastle upon Tyne, England (Newcastle) and London, England (London) will be used. For intermediary support, servers located in Amazon's EU Ireland (AWS-EU) and US-East Northern Virginia (AWS-US) locations will be used. Each AWS virtual machine is configured using the `m1.medium` profile providing 3.75Gb memory, 1 AWS virtual core with 2 EC2 Compute Units and 410Gb storage space [SGR09, AFG$^+$10].

For each business service location (London and Newcastle), 10,000 transmissions will be sent to each intermediary support location (AWS-EU and AWS-US). Providing a total of 40,000 transmissions whose communication and processing overheads will be timed for evaluation. Relevant latencies between all four sites will be calculated to demonstrate that processing overheads are stable while communication overheads vary depending on locations.

5.5.1 Measurements Taken

The implementation will take the following communication measurements:

1. Communication latency between the transmitting organisation and the intermediary support.
2. Communication latency within the intermediary support. That is, inter-component latencies within the intermediary service provider.
   - **N.B.** In decentralised systems these could also be inter-provider latencies.
3. Communication latency between the intermediary support and the recipient organisation.
4. Communication latency between transmitting organisation and recipient organisation (for comparison).

The following processing measurements will be taken:

1. Time taken to facilitate interception (i.e., to get the transmission in to the intermediary support).
2. Time taken to determine the desired support properties to satisfy (predicate or annotation driven).
3. Time taken to execute the required protocols.
4. Time taken to execute post-processing cleanup before delivery towards recipient.

Composite timings will be taken including the following:

1. End-to-end business transmission time. That is, the total time taken between dispatching a transmission from a sending organisation and its arrival at the recipient organisation.
    - **N.B.** This timing does not include business level processing, this occurs *before* transmission at the sender and *after* receipt at the recipient.
2. End-to-end intermediary support timing. That is, total time taken to intercept, process and forward to transmission towards its recipient.

All of the above measurements will be taken for 10,000 transmissions under the following four experiments:

1. Business services operating on **Newcastle** servers, Intermediary support operating on **AWS-EU** servers.
2. Business services operating on **Newcastle** servers, Intermediary support operating on **AWS-US** servers.
3. Business services operating on **London** servers, Intermediary support operating on **AWS-EU** servers.
4. Business services operating on **London** servers, Intermediary support operating on **AWS-US** servers.

5.5.2 Results

Table 5.1 shows the median observed communication latencies between the four locations.

* The observed latencies *within* the AWS-EU and AWS-US locations (3ms and 2ms) actually represent local connections to HTTP servers running on the same machines, not communication between different servers within the server locations. These figures are included as they comprise portions of the total time taken to render support to

Location	Newcastle	London	AWS-EU	AWS-US
Newcastle	9	39	99	172
London	39	15	119	220
AWS-EU	99	119	3*	-
AWS-US	172	220	-	2*

Table 5.1: Latencies between server locations, measured in milliseconds.

intercepted transmissions. Timings between AWS-EU and AWS-US zones are not taken as this experiment assumes centralised providers.

Table 5.2 shows the observed median processing times for intermediary support operating on AWS-EU and AWS-US servers.

	AWS-EU	AWS-US
Interception Processing	3	1
Determine Properties (Predicates)	12	11
Determine Properties (Annotations)	15	18
Execute Protocols	38	46
Post-processing	1	2

Table 5.2: Median processing times for intermediary support operating in AWS-EU and AWS-US locations, measured in milliseconds.

Medians calculated for interception processing, protocol execution and post-processing are averaged across 20,000 transmissions (2 business service locations tested per intermediary support location at 10,000 transmissions each).

Medians calculated for determining properties via predicates and annotations are 10,000 each (the transmissions from each business service location are half annotation based and half predicate based).

Table 5.3 shows the median end-to-end times for all four experiments.

	AWS-EU		AWS-US	
	Newcastle	**London**	**Newcastle**	**London**
Business Transmission Time	251	296	406	510
Intermediary Support Time	53	58	68	62

Table 5.3: Median end-to-end times for intermediary support involvement, measured in milliseconds.

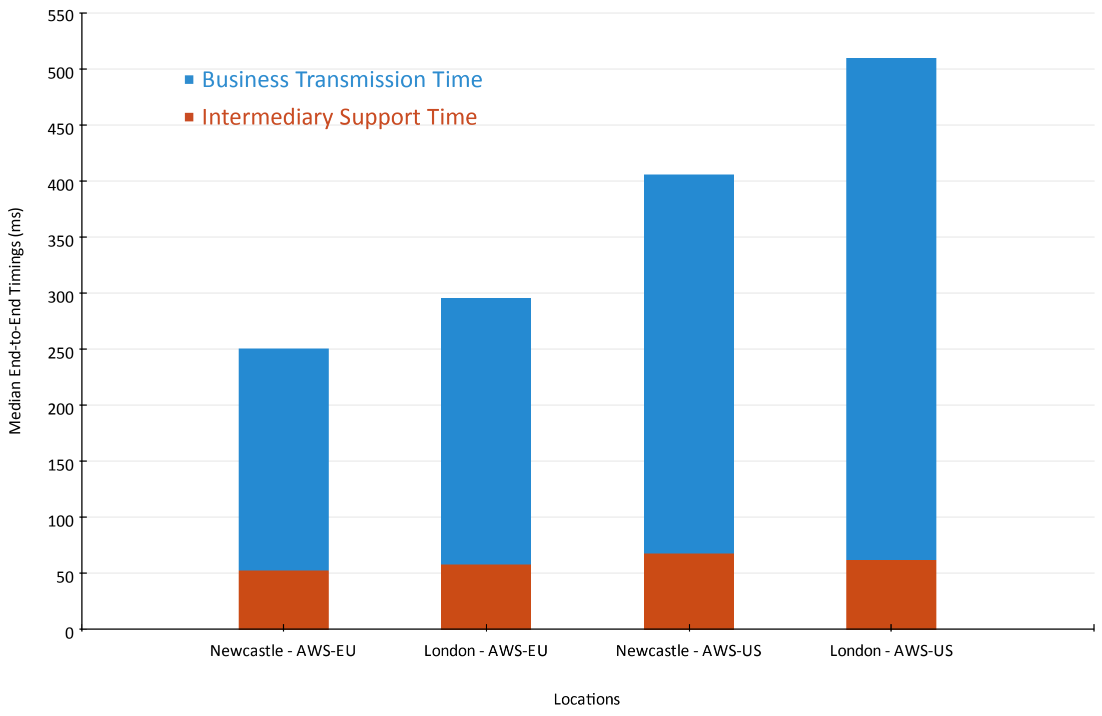


Figure 5.10: Graph comparing end-to-end time measurements taken, measure in milliseconds.

Table 5.3 and Figure 5.10 indicate that the total time taken by the centralised intermediary support remains stable where end-to-end business transmission time is most significantly impacted by communication overheads.

5.5.3 Centralised Support Costs

The performance of the intermediary support was stable on both servers, owing to the same AWS EC2 hardware provision profile (m1.medium). The expected result for centralised intermediary support running on equally provisioned and prioritised virtual machines was that the intermediary support operation would be similar across both servers. Table 5.3 and Figure 5.10 demonstrate this, showing that total time taken for intermediary support to fully process intercepted transmissions averaged (median) between 53 and 68ms where the remaining communication overheads constitute the majority of the end-to-end business transmission time taken.

The blue portion of the bars in Figure 5.10 comprise transmission from sender to intermediary support and transmission from intermediary to recipient. That is, we would expect the blue bar to be approximately double the median latency between support and interacting organisations. This is observed for all four scenarios.

For example, communication latency between Newcastle and AWS-EU is 99ms (Table 5.1), median total time taken by intermediary support is 53ms and median total business transmission time is 251ms (Table 5.3). Following this we would expect the total business transmission time minus time taken by intermediary support to be approximately two times the communication latency (sender to support followed by support to recipient) giving:

$$251ms - 53ms = 198ms = 2 * 99ms$$

The centralised nature of the support has helped to keep total intermediary support time low. The communication latency for requests within the centralised support was shown as 2ms for AWS-US and 3ms for AWS-EU in Table 5.1. Each supported transmissions results in a call to evidence generation, time-stamping and evidence storage components within the centralised design as shown in Figure 5.9.

The centralised implementation optimises support component invocation and only calls each of these components once per protocol execution to generate, timestamp and store both kinds of evidence, resulting in three requests each with a request and response component with communication latency in both directions.

For the AWS-US server, three invocations with internal communication latency in both directions gives: $(2 * 2ms) * 3 = 12ms$. That is, a median 12ms of the total time taken by intermediary support operating on the AWS-US server was internal communication. For the AWS-EU server (3ms) the corresponding time is $(2 * 3ms) * 3 = 18ms$. These figures can be subtracted from the total time taken by intermediary support to give a median total computation time within intermediary support.

5.5.4 Estimating Decentralised Support Costs

Following the above, we can begin to extrapolate and estimate the costs associated with decentralising components in terms of the messages that are required to be passed. Considering the Newcastle to AWS-EU scenario in Table 5.3, end-to-end business transmission time was 251ms and total intermediary support time was 53ms. Of the 53ms, 18ms of that was communication latencies as discussed in the previous section. This gives $53ms - 18ms = 35ms$ of pure computation time within intermediary support.

Using 35ms as a base and assuming that all components run on hardware matching the AWS EC2 m1.medium profile, we can estimate time taken by this intermediary support running in decentralised configurations by calculating number of messages to be passed and the latencies between the component locations.

The Coffey-Saidha protocol defined in Section 5.2 requires five protocol messages to be passed. Additionally, two invocations of component services for evidence generation, time-stamping and storage are required (with communication latency in both directions) and two transmissions delivering the transmission from sender to intermediary support and from intermediary support to recipient.

If we imagine an example set of distinct providers with an average inter-provider communication latency of 20ms, and provider-to-organisation latencies of 50ms, executing the Coffey-Saidha non-repudiation protocol, with individual components for evidence generation, time-stamping (witnessing) and storage, total intermediary support time would be:

$$\begin{array}{rrr} & 35ms & \textit{Base processing time} \\ + & 5 * 20ms & \textit{Coffey-Saidha Message Passing} \\ + & (2 * 20ms) * 6 & \textit{Support Component Communication} \\ + & 2 * 50ms & \textit{Provider to Organisation Latency} \\ \cline{2-2} = & 835ms & \textit{Business Transmission Time} \end{array}$$

Using the communication latencies in Table 5.1, a worst case scenario using the observed values would involve all communication having a median latency of 220ms. Inserting 220ms for all communication into the above expression would result in an end-to-end business transmission time of 4215ms. Similarly, a best case fully decentralised system using the lowest cross-location latency (39ms between Newcastle and London) would result in an end-to-end business transmission time of 776ms. If we considered different providers all located in Newcastle (9ms median communication latency between machines) the end-to-end business transmission time would be 206ms.

In summary, the total intermediary support time is a function of base processing time of protocol(s) to be executed combined with the latencies for inter-component (and potentially inter-provider) communication. End-to-end business transmission time becomes a function of total intermediary support time and communication latencies between interacting organisations and support intermediaries.

5.5.5 Evaluation and Summary

The functionality of this implementation was tested in Section 5.4.4. This implementation disabled logging and console output and deferred output of timing data to ensure that data collected was accurate and that finer grained timing operations did not unintentionally increase the timing of larger operations.

The impact of support on interactions between organisations in any two locations may be calculated by subtracting the communication latency between the two organisations from the end-to-end business transmission time involving intermediary support between the organisations. If we consider a business transmission between two organisations in Newcastle, with centralised intermediary support hosted on AWS-EU servers, we know median latency directly between organisations is 9ms (Table 5.1), and end-to-end business transmission time via intermediary support is 251ms (Table 5.3), giving an impact of $251ms - 9ms = 242ms$.

OTA and ebXML Common Business Processes do not specify default timeouts or lifespans for their conversations except for acknowledgement that they may be "long-lived" due to asynchronicity [Ope11b, OAS01b]. RosettaNet does not specify expected life spans for entire conversations but does mandate a default two hours timeout on individual exchanges. Given that the worst case estimated timings were measures in the seconds (4215ms in the previous section), the impact of the intermediary support on B2B conversations can be deemed acceptable (0.000583% of a default two hour deadline). We assume that for additional protocols, and more realistic (i.e., powerful) servers, acceptable impacts on B2B interactions could be maintained.

5.6 Summary

The implementations in Section 5.3 and 5.4 had their functionality tested in Section 5.3.4 and 5.4.5, demonstrating that the evidence was correctly generated, stored and that transmissions were not delivered until protocols had successfully executed. Essentially, fairness and accountability were provided successfully to all intercepted transmissions requiring them.

The implementation in Section 5.5 was tested in terms of its performance and impact upon typical B2B transmissions and evaluated to be acceptable in these terms. That is, support was provided in between hundreds to thousands of milliseconds and the default deadlines and timeouts of B2B interactions far exceed these [Ros02, Ros09]. The size of the transmissions used in performance testing was larger than any message observed in example conversations in ebXML, RosettaNet and OTA.

In terms of previous work, the implementations improve over [Coo06] by being transparent, asynchronous and loosely coupled. [NZB04] is improved over by the implementations being automated, transparent, asynchronous and potentially supporting additional protocols and properties. [YCW+10] is improved over by transparently encapsulating the invocation of underlying support and providing stronger accountability (and fairness) guarantees. Both [Coo06] and [YCW+10] allow multiple protocols but do not provide a notion of functional properties satisfied by the supported protocols. [Coo06] essentially supports synchronous execution of arbitrary protocols over an intercepted SOAP message without a notion of what support the protocols provide. [YCW+10] provides a system that is implicitly programmed to deliver Accountability with no refinements on the type of support offered although execution among the Accountability services in the Accountability Service Domain is abstracted away meaning it could be improved (to address key revocation) but would still be invoked in a non-transparent manner (i.e., business processes are still expected to emit their own evidence).

The implementations have the predicates described in Section 5.1.1 explicitly programmed in to them. The use of a rules engine to allow dynamic definition of predicates would likely incur an increase in execution time by intermediary support.

A limitation in terms of real world use is that all interacting participants must employ compatible intermediary support in order to have stronger guarantees provided for their transmissions. Properties cannot be guaranteed without cooperation from intermediaries acting on behalf of both sides of an interaction (and independently trusted components). Section 4.4 discussed these issues but the best approach is subjective.

In order to enable the functionality of declarations by predicates intermediary support must be programmed with the knowledge to extract and understand information from intercepted transmissions where available such that predicates relying on that information can successfully be interpreted. While it is beneficial that the implemented intermediary support provides its functionality at the transport level (i.e., all JMS or HTTP transmissions can be intercepted, regardless of B2B or other content), in order to capitalise on any domain specific information, the intermediary support must understand how to extract that information. Similarly, if defining predicates using raw content matching, as discussed in Section 3.6, those predicates require that the defining party assumes specific knowledge about the contents of intercepted transmissions.

6 Summary and Future Work

6.1 Summary of Contributions

This thesis sought to alleviate the requirements of expertise, infrastructure and integration placed on interacting participants who wished to support their interactions. This is achieved by the design and implementation of intermediary support middleware that operates asynchronously, transparently and decoupled from higher level communication standards (e.g., B2B). By enforcing these characteristics, support intermediaries can be provided by support service providers through which transmissions are routed and provided with additional support. These combination of factors allow the aforementioned requirements (expertise, infrastructure and integration) to be alleviated from interacting participants.

The support offered by the middleware intermediaries was abstracted into mappings of functional properties (describing what support is offered) and technical protocols (defining how the support is satisfied). The hierarchies in Section 3.3 represent an application of functional properties mapped to technical protocols in the B2B domain. That is, these concerns were identified as relevant to B2B interactions in order to support regulation, and suitable protocols were identified that could be encapsulated into intermediary support to address these concerns with added challenges of transparency and loose-coupling.

The generalised middleware design proposed in Chapter 4 is capable of expressing related work upon which this thesis builds, as demonstrated in Section 4.12.

Issues of transparency are motivated by a desire to ensure separation of concern

(Chapter 1 and 2). In doing so, higher levels of abstraction (e.g., B2B) can be supported at lower levels (i.e., transmission level by the intermediary support in this thesis) as demonstrated in Chapter 3 through 5.

Ensuring loose-coupling is also motivated by a desire to ensure separation of concern, discussed throughout Chapter 1 through 3. In doing so, transmissions in any application domain can be supported simply by intermediary support exposing the correct transport standards (e.g., JMS or HTTP) and the declaration via annotation and predicate mechanisms discussed in Chapter 3 through 5.

By adopting a message oriented middleware approach, and assuming participants will eventually be online to exchange messages (Section 2.4), asynchronicity is supported within intermediary support, better supporting the potential longevity of interactions such as those in the B2B domain. Additional benefits of supporting interactions using intermediaries as done in this thesis include the expression of asymmetric requirements by interacting participants and the support for asymmetric transports within the generalised middleware design.

The implementations in Chapter 5 server as proof of concept of the approaches discussed throughout the thesis and are demonstrated as functioning as expected (Section 5.3.4 and 5.4.4) and performing acceptably with regards to their impact in the chosen application domain (i.e., B2B, Section 5.5).

The approach of providing intermediary support at lower levels (e.g., transport level) allow the support offered to be tailored to other application domains relying on message oriented middleware seamlessly. Similarly, the notion of altering execution at lower levels to provide additional or altered functionality while maintaining higher level operation may be applied beyond the message oriented middleware domain. For example, Platform as a Service offerings may be adapted to become Accountable Platform as a Service, discussed in future work in Section 6.3.

6.2 Summary by Chapter

This section summarises the chapters of this thesis.

Chapter 1 introduced the general problem area (supporting message oriented interactions) and an application domain of such issues (B2B). The approach of using transparent intermediaries was introduced and discussed in the context of providing fairness and accountability to B2B interactions.

Chapter 2 more thoroughly defined the B2B application domain including terminology, assumptions, concerns, technical protocols and a survey of existing B2B standards to motivate the pursuit of a lower level approach that was standards independent and application domain agnostic (i.e., not just B2B). Related work was surveyed both in the domain of supporting B2B interactions and supporting general message oriented interactions to investigate suitable methods for ensuring transparency and loose-coupling.

Chapter 3 generalised the background and surveyed material to better understand how all interactions could be supported (i.e., the smallest unit of message oriented interaction that must be supported) in the general domain, and how such support could be applied to all transmissions (both clear and opaque) using declaration mechanisms. B2B specific property hierarchies for fairness, accountability and consistency were devised to be later implemented as an example of providing intermediary support in the B2B domain (i.e., functional properties paired with technical protocols realised by support intermediaries). Section 3.5 and 3.6 discussed supporting individual transmissions and declaration mechanisms.

Chapter 4 discussed considerations for compatibility and ensuring transparency within intermediary middleware and went on to propose components that could facilitate transparent interception to be subsequently composed with components implementing specific protocols to satisfy the functional properties they were mapped from. Configurations for composition and deployment were discussed to demonstrate the flexibility of the proposed design and previous work demonstrate to be instances of components implementing specific protocols.

Chapter 5 implemented proof of concept systems and evaluated them in terms of functionality and performance to gauge how support is improved over previous work (specifically over [Coo06, YCW^{+}10, NZB04]) and what its effect on interactions in the B2B domain might be (the comparison of median support time versus unsupported transmission time).

6.3 Future Work

Section 2.11 discussed the various levels of delivery associated with cloud computing usage. Of these models, Platform as a Service provides a monitored execution environment into which specially packaged applications are deposited to be automatically deployed and executed (e.g., Google Application Engine). Following the theme of providing support at lower levels, it may be possible to provide an Accountable Platform as a Service into which arbitrary applications are deployed. As the application makes use of the features provided by the platform, additional functionality could also generate accountability evidence. All of this could occur transparently to the deployed service. This represents an application of the work beyond the domain of message oriented middleware.

The centralised cloud based implementation was implemented in two pieces, an intermediary providing Accountability as a Service and boundary interceptors marshalling incoming and outgoing communication. This presented the possibility to represent exchanges (or protocol executions) as resources using Restful architectures. This would require well defined semantics for the resources and operations (i.e. the HATEOAS constraint discussed in [Fie00]). Similarly, protocols can be adapted for execution via HTTP (although not requiring Restful architectural constraints) providing a well understood and accessible representations through which participants may easily interact.

The centralised cloud implementation also prompted the exploration of domain specific APIs to programatically drive conversations under specific B2B standards. The use of the Scala language was suited to this. Such work would take conversations defined in well-known B2B standards such as RosettaNet or ebXML and produce an API used to drive them. For example, RosettaNet PIP 3A4 may be turned into an API that capitalises on the names of the messages (e.g., SubmitPurchaseOrder) to provide more intuitive method names (e.g., submitPurchaseOrder(RequiredItems)) and automate the generation of RosettaNet messages and send them for delivery. This approach completely breaks transparency but does offer the ability to hide technical details and allow engagement in conversations using familiar terminology capitalising on domain specific knowledge.

Another interesting problem is the possible calculation of a conversation's structure simply by observing transmissions between participants. That is, is it possible to determine a conversation's structure by observing transmissions and if so, can properties then be attributed to that conversation based on its associated transmissions' requirements. It may then be possible to associate transmissions with a calculated conversation to determine potential requirements.

On a more technical level within the intermediary support itself, declaration mechanisms within the support could be refined to allow finger grains of expression (e.g., required versus desired properties) and specify whether certain characteristics are already satisfied by specific criteria in intercepted transmissions (e.g., does this standard already provide strong fairness?).

The satisfaction of consistency as property using synchronisation as discussed in [MJSC07] are not applicable to all intercepted transmissions (e.g., if the contents of a transmission are opaque), it would be preferable to find alternative techniques applicable to all interactions although this was beyond the scope of this thesis. Additionally, techniques may exist for better ascertaining ideal points of synchronisation for a given flow of execution (e.g., a conversation).

The use of cloud computing prompts the requirement that service providers trust the infrastructure upon which their services are deployed. Work is ongoing into ensuring execution environments are trusted that must be integrated into any realistic offering hoping to support real world interactions [SGR09, CS11]. Organisations cannot be realistically expected to trust service providers who can provide no proof of trust in the infrastructure upon which the services are deployed.

The capturing of trust as a property with three levels of granularity (full, semi and none) may be inadequate for future requirements. It may be better to model trust as some metric or heuristic (e.g., express trust a some probability p) [JsIB07]. This may be of use when considering when it is worth invoking extra support versus when it is not. For example, a sufficient probability of trust may allow the use of probabilistic protocols for accountability and fairness.

The intermediary support could provide support for fairness through the support for fair exchange protocols similar to those discussed by [VPG99]. These protocols support the definition of item descriptions that can be used by a TTP to verify that the items being exchanged are the expected items before passing the exchange items to their respective recipients. Support for this was not included in this thesis as evidence demonstrating accountability cryptographically binds the contents of the transmission to its origin and receipt meaning there is proof any participant sent an unexpected or invalid item. [VPG99] also discussed the modular composition of protocols from smaller building blocks, this would be interesting in terms of dynamically composing protocols with new previously unsolved combinations of characteristics.

Strano's work, discussed in [Str09] and shown as an instance of deployment and composition configurations in Section 4.12, constitutes a service designed specifically to react to events and how they affect current engaged contracts. That is, participants emit events when they take an action (e.g., "I sent this type of message"), and a centralised contract monitor evaluates this event against a contract. Contracts are modelled as EROP sets (electronic rights, obligations and prohibitions) which dictate the actions that can, must and should never be taken. An issue with this work was that participants communicate directly (e.g., $A \rightarrow B$ and $B \rightarrow A$) and emit their own, untrusted, events. The work could be adapted such that messages from A and B are intercepted by intermediary support who is then responsible for generating and emitted trusted events to provide irrefutable contract monitoring. This would entail determining the context of the intercepted transmission to sufficiently advise some central (trusted by and independent to all participants) monitor.

Bibliography

[AFG$^+$10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, and David A Patterson. A View of Cloud Computing. *Commun. ACM*, 53(4):50–58, 2010.

[AMQ09] AMQP. AMQP A General-Purpose Middleware Standard, 2009.

[Aso98] N Asokan. *Fairness in Electronic Commerce.* PhD thesis, University of Waterloo, June 1998.

[ASW96] N Asokan, Matthias Schunter, and Michael Waidner. Optimistic Protocols for Multi-Party Fair Exchange. Research Report RZ2892, IBM Zurich Research Lab, 1996.

[ASW98] N Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *Proc. IEEE Symp. on Research in Security and Privacy*, pages 86–99, Los Alamitos, CA, USA, 1998.

[BBMS01] Adrian Baldwin, Yolanta Beres, Marco Casassa Mont, and Simon Shiu. Trust Services: A Trust Infrastructure for E-Commerce. Technical Report HPL-2001-198, Hewlett Packard Laboratories, Bristol, UK, 2001.

[CD04] Jan Cederquist and Muhammad Torabi Dashti. Formal Analysis of a Fair Payment Protocol. In *Proc. IFIP World Comp. Congress Workshop on Formal Aspects in Security and Trust (FAST)*, Toulouse, France, 2004.

[Coo06] Nick Cook. *Middleware support for non-repudiable business-to-business interactions.* PhD thesis, Newcastle University, 2006.

[CRS06] Nick Cook, Paul Robinson, and Santosh K Shrivastava. Design and Implementation of Web Services Middleware to Support Fair Non-repudiable Interactions. *Int. J. Cooperative Information Systems (IJCIS) Special Issue on Enterprise Distributed Computing*, 15(4):565–597, December 2006.

[CS96] Tom Coffey and Puneet Saidha. Non-repudiation with mandatory proof of receipt. *ACM SIGCOMM Comp. Commun. Review*, 26(1):6–17, January 1996.

[CS11] Christian Cachin and Matthias Schunter. A Cloud You Can Trust. *IEEE Spectrum*, 2011.

[CSB03] Tom Coffey, Puneet Saidha, and Peter Burrows. Analysing the Security of a Non-repudiation Communication Protocol with Mandatory Proof of Receipt. In *Proc. 1st ACM Int. Symp. on Information and Communication Technologies*, Dublin, Ireland, 2003.

[DA99] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176.

[DY83] Danny Dolev and Andy C Yao. On the Security of Public Key Protocols. *IEEE Trans. Inf. Theory*, 29(2):198–208, 1983.

[EY80] Shimon Even and Yacov Yacobi. Relations among public key signature systems. Technical report, Computer Science Department, Technicon, Haifa, Israel, January 1980.

[FGM$^+$99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785, 6266, 6585.

[Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[FLP85] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2):374–382, April 1985.

[FR97] Matthew Franklin and Michael Reiter. Fair Exchange with a Semi-Trusted Third Party. In *Proc. ACM Conf. on Comp. and Comm. Security (CCS)*, Zurich, Switzerland, 1997.

[Gol99] Dieter Gollmann. *Computer Security.* John Wiley and Sons, 1999.

[GPV99] Felix Gärtner, Henning Pagnia, and Holger Vogt. Approaching a formal definition of fairness in electronic commerce. *Reliable Distributed Systems, 1999. Proceedings of the 18th IEEE Symposium on DOI - 10.1109/RELDIS.1999.805123*, pages 354–359, 1999.

[HKVO07] Armin Haller, Paavo Kotinurmi, Tomas Vitvar, and Eyal Oren. Handling heterogeneity in RosettaNet messages. In *Proceedings of the 2007 ACM symposium on Applied computing - SAC '07*, page 1368, New York, New York, USA, March 2007. ACM Press.

[HNK02] James E Hanson, Prabir Nandi, and Santhosh Kumaran. Conversation Support for Business Process Integration. In *Proc. 6th IEEE Int. Enterprise Distributed Object Computing Conf. (EDOC)*, Lausanne, Switzerland, 2002.

[Hof02] P. Hoffman. SMTP Service Extension for Secure SMTP over Transport Layer Security. RFC 3207 (Proposed Standard), February 2002.

[Hor07] Morton J Horwitz. The Historical Foundations of Modern Contract Law. *Harvard Law Review*, 87(5):917–956, October 2007.

[HV04] T. Hansen and G. Vaudreuil. Message Disposition Notification. RFC 3798 (Draft Standard), May 2004. Updated by RFCs 5337, 6533.

[ISO89] ISO. *Information Processing Systems - Open Systems Interconnection - Security Frameworks for Open Systems - Basic Reference Model - Part 2: Security Architecture.* ISO 7498-2, 1989.

[JsIB07] Audun Jø sang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, March 2007.

[Kle01] J. Klensin. Simple Mail Transfer Protocol. RFC 2821 (Proposed Standard), April 2001. Obsoleted by RFC 5321, updated by RFC 5336.

[KM00] Steve Kremer and Olivier Markowitch. Optimistic non-repudiable information exchange. In *J. Biemond, (Ed.), 21st Symp. on Information Theory in the Benelux, Werkgemeenschap Informatieen Communicatietheorie*, pages 139—-146, Enschede, Netherlands, 2000.

[KMZ02] Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, 2002.

[Lev98] E. Levinson. The MIME Multipart/Related Content-type. RFC 2387 (Proposed Standard), August 1998.

[MC09] Derek Mortimer and Nick Cook. CS-TR-1214: A Declarative Approach to Configuring Business-to-Business Conversations. Technical report, Newcastle University, Newcastle upon Tyne, 2009.

[MC10] Derek Mortimer and Nick Cook. Supporting accountable business to business document exchange in the cloud. In *2010 IEEE International Conference* *on Service-Oriented Computing and Applications (SOCA)*, pages 1–8. IEEE, December 2010.

[MGK02] Olivier Markowitch, Dieter Gollmann, and Steve Kremer. On fairness in exchange protocols. In *Proc. 5th Int. Conf. on Information Security and Cryptology (ISISC 2002)*, Springer LNCS 2587, pages 451–465, Seoul, Korea, 2002.

[Mit01] John Mitsianis. A new approach to enforcing non-repudiation of receipt. Manuscript, 2001.

[MJS06] Carlos Molina-Jimenez and Santosh K Shrivastava. Maintaining Consistency Between Loosely Coupled Services in the Presence of Timing Constraints and Validation Errors. In *Proc. IEEE European Conf. on Web Services (ECOWS)*, pages 148–157, Zurich, Switzerland, 2006.

[MJSC07] Carlos Molina-Jimenez, Santosh K Shrivastava, and Nick Cook. Implementing Business Conversations with Consistency Guarantees using Message-oriented Middleware. In *Proc. 11th IEEE Int. EDOC Enterprise Computing Conf.*, pages 51–62, Annapolis, MD, USA, 2007.

[MJSW05] Carlos Molina-Jimenez, Santosh K Shrivastava, and John Warne. A Method for Specifying Contract Mediated Interactions. In *Proc. 9th IEEE Int. EDOC Enterprise Computing Conf.*, Enschede, Netherlands, 2005.

[MK01] Olivier Markowitch and Steve Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In *Information Security Conference 2001, Lecture Note in Computer Science*, Berlin, Germany, 2001. Springer.

[MS01] Olivier Markowitch and Shahrokh Saeednia. Optimistic Fair Exchange with Transparent Signature Recovery. In *5th International Conference, Financial Cryptography 2001, Lecture Notes in Computing Science*, volume 2339, pages 339 – 350, 2001.

[MU00] Naftaly Minsky and Victoria Ungureanu. Law-Governed Interaction: A Coordination and Control Mechanism for Heterogeneous Distributed Systems. *ACM Trans. Software Eng. and Methodology*, 9(3):273–305, 2000.

[NZB04] Aleksandra Nenadic, Ning Zhang, and Stephen Barton. FIDES - A Middleware E-Commerce Security Solution. In *Proc. 3rd European Conf. on Inf. Warfare and Security (ECIW)*, London, UK, 2004.

[OAS01a] OASIS. ebXML Business Process Specification Schema. OASIS ebXML Specifications, July 2001.

[OAS01b] OASIS. ebXML Catalog of Common Business Processes. OASIS ebXML Specifications, July 2001.

[OAS01c] OASIS. ebXML Technical Architecture Specification. OASIS ebXML Specifications, February 2001.

[OAS02] OASIS. Collaboration-Protocol Profile and Agreement Specification. OASIS ebXML Specifications, September 2002.

[OAS07a] OASIS. ebXML Messaging Services 3.0 Conformance Profiles. OASIS ebXML Specifications, 2007.

[OAS07b] OASIS. ebXML Messaging Services Version 3.0: Part 1, Core Features. OASIS ebXML Specifications, October 2007.

[OAS11] OASIS. AS4 Draft Conformance Profile of ebMS 3.0. OASIS ebXML Specifications, 2011.

[Ode10] Martin Odersky. The Scala Language Specification, 2010.

[Ope11a] Open Applications Group. Open Applications Group Integration Specification 9.5.1, 2011.

[Ope11b] Open Travel Alliance. OpenTravel Schema 2011B, 2011.

[PG99] Henning Pagnia and Felix Gärtner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Dept. of Computer Science, TU Darmstadt, 1999.

[PR85] J. Postel and J. Reynolds. File Transfer Protocol. RFC 959 (INTERNET STANDARD), October 1985. Updated by RFCs 2228, 2640, 2773, 3659, 5797.

[PVG03] Henning Pagnia, Holger Vogt, and Felix Gärtner. Fair Exchange. *The Computer Journal*, 46(1):55–75, 2003.

[Res00] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000. Updated by RFC 5785.

[Ros02] RosettaNet. RosettaNet Implementation Framework: Core Specification, February 2002.

[Ros09] RosettaNet. RosettaNet Overview: Clusters, Segments and Pips, January 2009.

[Sch96] Bruce Schneier. *Applied Cryptography.* John Wiley and Sons, 2nd edition, 1996.

[Sch00] Matthias Schunter. *Optimistic Fair Exchange.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2000.

[SGR09] Nuno Santos, Krishna P. Gummadi, and Rodrigo Rodrigues. Towards Trusted Cloud Computing. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, San Diego, CA, USA, 2009. USENIX Association.

[SHF02] David Solo, Russell Housley, and Warwick Ford. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2002.

[SRC84] Jerome Saltzer, David Reed, and David Clark. End-to-End Arguments in System Design. *ACM Trans. Computer Systems (TOCS)*, 2(4):277–288, 1984.

[SSCE07] James Skene, Allan Skene, Jason Crampton, and Wolfgang Emmerich. The monitorability of service-level agreements for application-service provision. In *Proceedings of the 6th international workshop on Software and performance - WOSP '07*, page 3, New York, New York, USA, February 2007. ACM Press.

[Str09] Massimo Strano. *Contract Specification for Compliance Checking of Business Interactions.* PhD thesis, Newcastle University, 2009.

[Sv10] Chris Smith and Aad van Moorsel. Mitigating provider uncertainty in service provision contracts. *Economic Models and Algorithms for Distributed Systems*, pages 143–159, 2010.

[SW07] Radu Sion and Marianne Winslett. Regulatory-compliant data management. *Very Large Databases*, pages 1433–1434, 2007.

[SWR97] Dennis D Steinauer, Shukri A Wakid, and Stanley Rasberry. Trust and traceability in electronic commerce. *StandardView*, 5(3):118–124, September 1997.

[Tan03] Andrew Tannenbaum. *Computer Networks.* Prentice Hall, 2003.

[TMRS02] Stefan Tai, Thomas Mikalsen, Isabelle Rouvellou, and Stanley Sutton. Conditional messaging: extending reliable messaging with application conditions. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on DOI - 10.1109/ICDCS.2002.1022249.*, pages 123–132, 2002.

[Uni11] United Nations Economic Commission for Europe. UN/EDIFACT D.11A, 2011.

[VPG99] Holger Vogt, Henning Pagnia, and Felix Gärtner. Modular Fair Exchange

Protocols for Electronic Commerce. In *Proc. IEEE Annual Comp. Security Applications Conf.*, pages 3–11, Phoenix, AZ, USA, 1999.

[Wor00] World Wide Web Consortium. SOAP with Attachments, 2000.

[Wor02] World Wide Web Consortium. XML Encryption Syntax and Processing, 2002.

[Wor08] World Wide Web Consortium. XML Signature Syntax and Processing (Second Edition), 2008.

[YC04] Aydan Yumerefendi and Jeffrey Chase. Trust but Verify: Accountability for Network Services. In *Proc. 11th ACM SIGOPS European Workshop*, Leuven, Belgium, 2004.

[YCW$^+$10] Jinhui Yao, Shiping Chen, Chen Wang, David Levy, and John Zic. Accountability as a Service for the Cloud. In *2010 IEEE International Conference on Services Computing*, pages 81–88. IEEE, July 2010.

[ZG96a] Jianying Zhou and Dieter Gollmann. A fair non-repudiation protocol. *IEEE Symposium on Security and Privacy, Research in Security and Privacy, IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Security Press*, pages 55–61, 1996.

[ZG96b] Jianying Zhou and Dieter Gollmann. Observations on non-repudiation. In *Advances in Cryptology—ASIACRYPT'96*, Springer LNCS 1163, pages 133–144, Kyongju, Korea, 1996.

[ZG97a] Jianying Zhou and Dieter Gollmann. An efficient non-repudiation protocol. In *The 10th Computer Security Foundations Workshop, IEEE Computer Society Press, Silver Spring*, pages 126–132, 1997.

[ZG97b] Jianying Zhou and Dieter Gollmann. Evidence and non-repudiation. *J. ˜Network and Comp. Applications*, 20(3):267–281, 1997.

[Zho01] Jianying Zhou. *Non-repudiation in Electronic Commerce.* Artech House Computer Security Series, 2001.