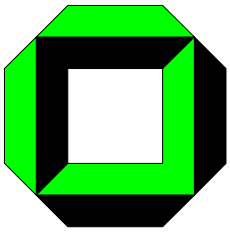
Transaction Protocols for Self-Organizing Systems of Autonomous Entities

Philipp Obreiter, Ioana Nistoreanu
obreiter@ipd.uni-karlsruhe.de, Iona.Nistoreanu@insa-lyon.fr

July 23, 2004

Technical Report Nr. 2004-11



University of Karlsruhe
Faculty of Informatics
Institute for Program Structures and Data Organization
D-76128 Karlsruhe, Germany

Abstract

Self-organizing systems of autonomous entities have gained wide-spread attention in the research community. The most difficult problem of such systems is that autonomous entities may choose between cooperation and defection in the transactions they participate. In internet based eCommerce, transaction protocols are applied for this purpose. Yet, it has been repeatedly conjectured that such protocols are not applicable to self-organizing systems. Distributed reputation systems have been proposed as a means of compensating for such lack of applicability. Still, there is no analysis of the applicability of transaction protocols and their relationship with distributed reputation systems.

Therefore, in this report, we identify the characteristics of the internet based transaction protocols and show that they are only partly applicable to self-organizing systems. We discuss conventional and evidence-aware distributed reputation systems and point out their pivotal role in facilitating self-organizing punishment of defective behavior. This leads us to regard on transaction protocols as a means to complement distributed reputation systems by preempting and perceiving defective behavior. Based on this observation, we conceive five transaction protocols that comply with these demands and find different tradeoffs. The key properties of the protocols are illustrated and discussed by a schematic visualization technique. Finally, we allow for a circumstance-dependent choice of the most appropriate transaction protocol by quantifying the tradeoffs of each protocol and illustrating the protocols' dominance structure.

Contents

Chapter 1

Introduction

We start this report with a motivating scenario of self-organizing systems of autonomous entities. This motivates the system model of Chapter 2. We conclude this chapter by providing an outline of this report.

1.1 Motivation

The Diane project [1] aims at developing concepts for ad hoc networks, which enable the efficient discovery, description and processing of information services. This kind of services allow the access to and the transaction of information through electronic means.

An ad hoc network is formed of a set of mobile, autonomous devices, which communicate with each other without relying on an existing infrastructure. The lack of a communication backbone requires constant cooperation from devices, i.e., each device has to fulfill services in the benefit of the other devices. For instance, a basic requirement is the execution of networking services: two nodes can communicate only if the intermediate nodes agree to forward the packets. But, as resources are limited, the mobile devices often tend to reduce their cooperation, mostly in the purpose of saving these resources.

Let's take the example of a student, X, connected to an ad hoc network, who wants to gather C programming notes, which are spread across the network. His service requirement leads to valuable resources losses (eg. energy, processing power) for the devices executing this service. Due to the high costs, few participants will agree to fulfill the service for X, unless he gives something in return. In other words, he should provide at his turn a service for the notes' owners, as a compensation for their resource losses.

The transactions appear to be a simple process if we assume that every device executes the service it has been asked for. But if we consider that the gained utility is higher for the user that does not execute the service, the defection tends to be preferred to the cooperation. This could lead to heavy negative effects for the performance of the network.

We conclude that cooperation enforcement is crucial for the DIANE network.

1.2 Outline

In Chapter 2, we abstract from the scenario and present its underlying system model. In Chapter 3, we identify the characteristics of internet based transaction protocols and describe their

most common instances. Furthermore, we compare the setting of the internet with the one of self-organizing systems. In Chapter 4, we discuss both conventional and evidence-aware distributed reputation systems and point out the role of transaction protocols in self-organizing systems. This leads us to conceive transaction protocols for such systems in Chapter 5. In this context, we illustrate and discuss the key properties of the protocols with a schematic visualization technique. In Chapter 6, we solve to the problem of choosing the transaction protocol that is most appropriate for the given circumstances of a transaction. Furthermore, we illustrate the principles of tradeoff-aware choice by providing dominance graphs for the proposed transaction protocols and roles. Finally, in Chapter 7, we summarize the report and point out future work.

In the appendix, we provide a glossary (Appendix B) and the implementation details of the protocols (Appendix A).

Chapter 2

System Model

The motivating scenario of the last chapter exemplifies the problem domain of this report. In this chapter, we provide a model of the problem domain in order to identify the key characteristics of the problem. The system model includes a model of the participating entities and their cooperation. Subsequently, we propose models for the items that the entities may exchange in the course of a transaction. Finally, we take a closer look at actions since they are the core ingredient of cooperation.

2.1 Model of Entities and Cooperation

We assume a system as it is described in [2, 3]. Figure 2.1 illustrates the key properties of the system model.

In the following, we introduce the key notions of the model:

- **Entities:** An entity is a component of the system. In the context of the scenario of Section 1.1, an entity is a user agent that resides on the PDA or laptop of its human principal.
- **Cooperation, transactions and protocols:** A pair of entities may cooperate by taking part in a transaction. The pair of entities (transaction peers) exchange one or several items that are mutually beneficial for them. In the scenario, a transaction consists of the proliferation of the programming notes by entity Y and a service in return by student X. A transaction protocol defines the steps that are processed in order to perform a transaction.
- **Reachability:** An entity may enter or leave the system at any time. Furthermore, partitioning may occur at any time.
- **Autonomy:** Each entity is autonomous. This means that an entity is free to *cooperate* or *defect* in the course of a transaction. Defection refers to the premature abandonment of the transaction protocol. In the scenario, take for example two PDAs that agree on exchanging a pair of files. After having received the file of the transaction partner, a transaction peer may defect by refusing to transmit the promised file.
- **Self-organization:** There is no available infrastructural entity. The system's organization is emergent because the entities have to assume infrastructural tasks. As a result, they organize themselves. The ad hoc network of the scenario inherently does not offer any

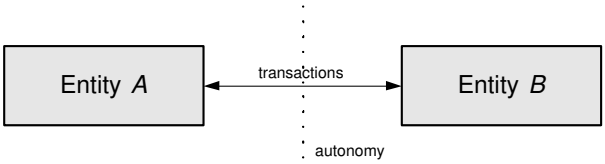


Figure 2.1: The system model

infrastructure. The means of self-organization is apparent on the network layer: Each entity has to route packets of other entities in order to enable inter-entity communication.

- **Identities and security:** Each entity has an identity[1] and is able to perform cryptographic operations. With respect to security, this means that **(1)** entities communicate over confidential channels that cannot be overheard by other entities and **(2)** entities may authenticate and issue non-repudiable tokens.

2.2 Item Model

An *item* is an action or a token that is exchanged in the course of a transaction. In this section, we classify items according to their characteristics. Based on the classification, we show which types of items are part of the system model.

2.2.1 Item Classes

In the following, we propose several classes of items and classify them in a taxonomy. Except for deliverable items, the item classes have been introduced in [4].

2.2.1.1 Deliverable Items

The terminology of exchanges and items suggests that items are exchanged in messages. However, there are actions that do not consist of their return values. Such actions are executed locally and, due to their non-perceptibility, the peers cannot verify the effectiveness of the execution[2].

A *deliverable item* is an item the effectiveness of which may be checked upon receival. In this regard, an item is effective if it complies with its specification.

An action that consists of side-effects in the real world is no deliverable item. For example, it is imperceptible for the user agent whether a document is printed. Even actions that remain within the information system might be unobservable, e.g., forwarding packets in the absence of observation and receipts [5]. Furthermore, the verification of an action's compliance to its specification is often daunting. For instance, even for a document service, it is difficult to check automatically whether it complies with its specification.

2.2.1.2 Forwardable Items

In the context of failure resolution, exchange protocols might demand for passing on an item several times. Therefore, the benefits and costs that arise from passing on an item should not

[1] Identities either are initially distributed by a third party or each entity provides itself with an identity.

[2] For example, an action may consist of storing a service advertisement. However, the advertising entity cannot perceive whether its advertisement is really stored or not [5].

depend on how many times it has been passed on. This calls for a specialization of deliverable items.

A *forwardable item* is a deliverable item that is *idempotent* with respect to the number of its receptions. This means that it does not matter whether it is received once or several times.

As a prerequisite for the forwardability criterion, the overhead of passing on a deliverable item has to be negligible. This is not the case for actions that mainly consist of sending a message. For example, in the context of a document service, most resources might be consumed for transmitting the respective document.

2.2.1.3 Commitment Items

In the course of a transaction protocol, the peers might have to issue commitments in order to coordinate their behavior.

A *commitment item* is an item that contains a non-repudiable commitment.

Commitment items are an important subclass of forwardable items[3]. For instance, contracts are commitment items since they represent a commitment to the processing of a transaction.

2.2.1.4 Generatable Items and Permits

Let us consider an entity that issues a commitment item but fails to comply with its commitment. We assume that its commitment is about promising the passing of a specific item. It might be desirable to enforce that the promised item may be derived from the commitment item.

A *generatable item* is a deliverable item that may be derived from a commitment item[4]. Such commitment item is a commitment to the delivery of the generatable item and is called *permit*. An entity that is able to derive a generatable item is called a *generator*.

In this context, generation of an item refers to signing a commitment item or executing an action. The class of generatable items is defined as a subclass of deliverable items. Otherwise, a generator does not know whether it has to generate the item.

The generation might be intransparent to the recipient of the generatable item. This means that the generatable item leaks the information whether it has been passed on by the issuer of the permit or whether it has been generated by the generator. Such intransparent generation is called *invasive*. Invasiveness is mandatory if the fact that an item is generated has to be deducible from the item itself.

Generatability is easily achieved for forwardable items if the permit contains the encryption of the generatable item. The encryption is such that only the generators are able to decrypt it. However, unless it is a generator, a peer that receives a permit is not able to verify whether the item that is generatable from the permit comes up with its expectations. This problem is partly solvable if the permit contains a description of the generatable item that is intelligible to both the generators and the non-generators. If generators recognize a discrepancy between the description and the generated item, they may disrecommend the misbehaving peer.

The problem of permit verification is fully solvable if, in addition, the generators are able to generate the item based on its description. This means that the permit only contains the description of the generatable item but not the item itself. For example, let us assume that the generatable item is the action of forwarding a packet to a destination. In this context, a generator

[3]The deliverability and forwardability criterion are both fulfilled for commitment items.

[4]The procedure of deriving a generatable item from a commitment item is called generation.

generates the item by forwarding the packet to the destination. The permit only has to contain the packet's content and the destination.

2.2.1.5 Revocable Items

A defector typically fails to pass on its item after having acquired its peers' items. There are two methods of countering defection. Generatability introduces a means of enforcing the passing of the defector's item. Alternatively, it might be possible to revoke the items that a defector has acquired.

A deliverable[5] item is *revocable* if it may be rendered useless after it has been acquired by an entity.

An item is revoked by the issuance a *revocation certificate*. An entity that possesses a revocation certificate for a given revocable item is called *revocation holder*.

At first glance, revocability requires that an effect of an item can be undone later on. Such items are called *undoable items*. There are few items that fulfill the undoability condition. An example for such items are checks [2]. If the agent acquires a check without executing its action, the principal might be able to gather evidence of the agent's defection. Upon delivery of such evidence to the bank, the check is tagged as invalid. If the agent's already has been credited, the transaction is undone.

The second subclass of revocable items are *invalidatable items*. The validity of a invalidatable item has to be thoroughly verified since its effect cannot be undone later on. Hence, reasoning about invalidatable items has to occur before the item is rendered effective. Consequently, such reasoning is contrary to reasoning about undoable items. The acquirer of the invalidatable item brings in the claim that the item is valid. The acquirer renders the invalidatable item effective if it validates the claim. Such reasoning is challenging since it does not suffice to base its decision upon the item itself. In addition, the potential revocation holders have to be contacted in order to get known whether the item has been revoked.

2.2.1.6 Divisible Items

Some exchange protocols are based on the interleaved exchange of items. This means that an item does not have to be completely passed on.

A deliverable item is *divisible* if it is composed of subitems and, for each subitem, its recipient has a positive gain from obtaining it.

The notion of subitems only makes sense for deliverable items. Therefore, the definition of divisible items is built on deliverable items.

An item that may be represented as bitarray is not necessarily a divisible item. For example, a contract is only valuable as a whole and every prefix of its bitarray representation is useless. Consequently, a contract is not a divisible item. The same consideration also holds for bonds [2]. In this regard, the difference of divisibility and granularity [6] becomes obvious. For example, checks offer fine granularity while being indivisible.

2.2.1.7 Interrelationship of Item Classes

We conclude the discussion of item classes with the illustration of their interrelationship in Figure 2.2. The arrows depict to the subsumption of item classes.

[5]The restriction of revocable items to deliverable items is made since revocable non-deliverable items are not significant.

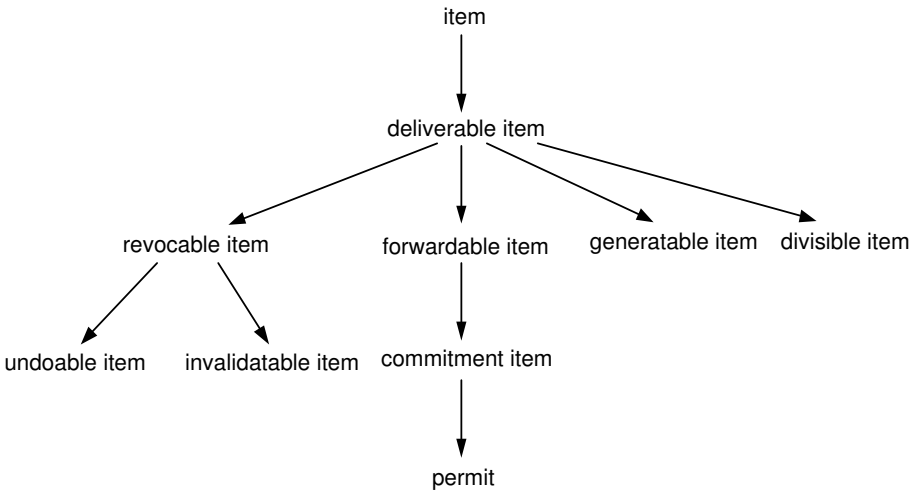


Figure 2.2: The subsumption of item classes

2.2.2 Restriction of Item Classes

For the remainder of this report, we consider the following types of items.

Contracts and receipts. Both contracts and receipts are commitment items. The semantics of these items is as follows: A transaction peer that issues a contract agrees to the terms of the transaction. If a peer issues a receipt, it acknowledges that its transaction partner has complied with its commitment.

Actions. We assume that an action is a deliverable item that is neither generatable nor revocable nor divisible. This assumption is justified as follows:

- **Deliverability:** Programming notes constitute a deliverable item since they can be checked upon receival. For the forwarding of packets, the active receipt mechanism of [5] is assumed.
- **Non-generatability:** Non-generatability means that, even if entity X commits to an action, the action cannot be executed by any other en lieu of entity X itself. This assumption makes sense since **(1)** an action like routing is inherently bound to the entity that commits to it and **(2)**, even if the effects of an action like the programming notes may be extracted from the commitment[6], it is not clear how a transaction peer is able to check whether the desired action has actually been executed[7].
- **Non-revocability:** In the scenario, actions cannot be invalidated since they are executed and rendered effective immediately during the transaction. Therefore, non-revocability means that the effects of executing an action cannot be undone. This assumption holds for the handover of programming notes or the forwarding of a packet.

[6] In such a case, the commitment contains an encryption of the action's effects. Only a dedicated entity is able to decrypt them.

[7] The transaction peer cannot decrypt the action's effects and, thus, has no idea whether its transaction partner defected or not.

Table 2.1: Cost tendencies for executing and losing control of various items

Item	Costs of Execution	Costs of Lost Control
Handover of a note[1]	↓	↑
Issuance and handover of a contract or receipt	↓	↑
Provision of an existing document[2]	↓	↑
Provision of an existing and commonly known document	↓	↓
Provision of a non-existing document[3]	↑	↑
Provision of personalized service[4]	↑	↓
Forwarding a packet	↑	↓

[1] A note is a non-repudiable promise of some later action [2]. The handover of a note is inexpensive with respect to execution since it only comprises a sign operation. However, losing control of a note means that oneself has committed to provide a later action.
[2] The execution costs only comprise the recall of the document. Still, the provision of the document incurs opportunity costs since the document may be further distributed by the transaction partner, thus yielding monopoly costs.
[3] The non-existing document has to be generated first. This usually incurs work by the user which is costly.
[4] A mailbox management service exemplifies this category. It corresponds to the provision of a non-existing document. However, there are no monopoly costs since the service will never be requested (with the same parameters) afterwards by another entity.

- **Non-divisibility:** Non-divisibility means that the execution of an action cannot be decomposed into meaningful parts. In the scenario, only the forwarding of a full network packet or the proliferation of a full programming note[8] yield benefits to the transaction partner.

2.2.3 Utility of Exchanging Items

In the following, we provide a utility-based model of exchanging an item. It consists of two states:

- **Execution of an item:** If the item is a contract or a receipt, the execution of an item means the contract or receipt is issued. Such issuance consists of cryptographically signing the commitment. This operation incurs relatively few costs. If the item is an action, the execution costs may vary.
- **Losing control of an item:** The incidence that the transaction partner acquires the item and may make use of it.

In Table 2.1, we exemplify the costs of executing and losing control of various items. It appears that they are not directly correlated.

[8] We assume here that a programming note is the smallest unit of consistency. For example, a programming note may consist of the documentation of a structure. In general, such note is only useful if the full text of the structure documentation is available.

Chapter 3

Transaction Protocols in the Internet

Nowadays, transaction protocols are applied for internet based eCommerce. In this chapter, we take a closer look at the setting of the internet since it differs from our system model. Subsequently, we identify the characteristics of internet based transaction protocols and describe the their most common instances. Finally, we examine whether and to which degree these transaction protocols can be applied to the self-organizing system that we have described in our system model.

3.1 Setting

The setting of the internet is described in [4]. It differs from the system model of Chapter 2. In the following, we highlight the differences:

- **Infrastructure:** In the internet, transaction peers may turn to dedicated infrastructural entities. There are two types of such entities:
  - **Trusted third parties:** These are entities of the information system. They are both commonly known and trusted. Furthermore, they may take an arbitration role during transactions.
  - **Law enforcement:** The entities that enforce law reside outside of the information system. They ensure that identifiable misbehavior becomes punished. Ultimately, they are necessary in order to guarantee high defection costs.
- **Transaction peers:** In several settings, there are more than just two transaction peers.
- **Item model:** Actions may be generatable, divisible, or revocable. This applies to some types of actions, e.g., eCash.
- **Reachability:** If communication breaks up in the internet, entities are certain that they are eventually able to re-establish their communication. However, this assumption does not hold for ad hoc networks.

3.2 Characteristics of Transaction Protocols

In this section, we identify the characteristics of internet based transaction protocols. This will provide a starting point for developing transaction protocols for the system model of Chapter 2.

In the literature, transaction protocols are often referred to as exchange protocols [4]. The two terms are generally interchangeable. Still, transaction protocols refer to the exchange of actions and, optionally, of the exchange of contracts and receipts. In this regard, they are more specific than exchange protocols. In the following discussion of characteristics, we keep to the notion of exchange protocols.

3.2.1 General Considerations

Exchange protocols aim at minimizing the benefits or losses that arise from defections. There are two approaches for such minimization:

- **Direct minimization:** The exchange protocol itself ensures that the effects of defections are minimized.
- **Evidence based minimization:** The exchange protocol allows for the collection of evidences [3], i.e., non-repudiable tokens. By this means, defective behavior becomes identifiable.

Regardless of which approach is pursued, exchange protocols have to consider *information asymmetry* with regard to the benefits and losses that arise from defections. This means that an entity does not know its peers' assessments of benefits and costs that ensue from a defection. They depend on the assessment of the exchanged items' values, the costs of processing the transaction, and the detriments that arise from the retaliation for the defection.

The analysis of exchange protocols is typically restricted to the exchange of deliverable items. This stems from the need for verifying the validity of items upon receipt. Hence, the following examination of exchange protocols is focussed on such exchanges.

3.2.2 Gradual Exchange Protocols vs. Third Party Exchange Protocols

We discern two types of exchange protocols that allow for the minimization of the benefits and losses that arise from defection. On the one hand, *gradual exchange protocols* call for an interleaved exchange of the items. If an entity defects, the ensuing benefits and losses are restricted to those of a subitem. On the other hand, *third party exchange protocols* make use of an additional entity that prevents or handles defections.

Gradual exchange protocols. By definition, gradual exchange protocols require *divisible items* for the interleaved exchange. In case of a defection, a peer might only have obtained parts of the desired item. The definition of divisible items ensures that the possession of such parts is profitable to such a peer.

In some environments, the peers are able to recover from premature failure of the exchange protocol. Then, each peer computes the remaining parts of the obtained subitems. Apparently, such protocols introduce asymmetric defection costs if the peers possess differing computation power.

Third party exchange protocols. An exchange protocol may utilize a *third party*, i.e., an entity that is not a transaction peer. Such a third party is able to prevent or handle the defection of transaction peers. Third parties are predestined to assume the role of the generator and the role of the issuer of revocation certificates.

There are two problems that arise from the use of third parties:

- **Trustworthiness:** The transaction peers have to rely on the exhibition of a pre-defined behavior by the third party. For the peers, the means of monitoring the behavior of the third party are usually restricted.
- **Reciprocation:** The prevention or handling of defections might not be profitable for the third party. Therefore, an entity only assumes the role of the third party if it is reciprocated. In a broader sense, the prevention or handling of defection represent the execution of an action so that the peers have to reciprocate the third party in the context of an transaction. Yet, such transaction may not be based on a further third party[1]. Apparently, the third party's reciprocation should be implicit[2] so that a further exchange of deliverable items becomes unnecessary.

The two problems show that the use of an infrastructural entity for the assumption of the third party role is not necessary but desirable.

Comparison of the two types of exchange protocols. Third party exchange protocols provide a better foundation for the evidence based minimization of the benefits and losses that arise from defections. However, third party exchange protocols demand for trust in the third party and for its reciprocation. These demands are not made by gradual exchange protocols. However, they need divisible items and generally introduce a larger overhead by the division of items into subitems and their interleaved exchange.

3.2.3 General Properties of Exchange Protocols

In the following, the properties of exchange protocols are examined. For this purpose, we introduce several terms that enable a concise discussion of the exchange protocol's properties.

Fairness. There are two notions of *fairness* [4].

- **Weak Fairness:** A correctly behaving entity cannot suffer any disadvantages from a defecting peer. In this regard, after handing over its item, the correctly behaving entity either receives the peer's item or it is able to gather evidence of the peer's defection.
- **Strong Fairness:** In addition to weak fairness, it is guaranteed that, if an entity hands over its item, its peer's item is acquired.

By definition, fairness does not consider when the correctly behaving entity acquires the evidence or the item. Yet, in some environments, the time of such acquisition is not negligible.

Rationality. In analogy to fairness, there are also two notions of *rationality* [7]:

- **Weak Rationality:** An entity cannot gain any advantage from defecting.
- **Strong Rationality:** A defecting entity suffers a disadvantage. Therefore, it is profitable to exhibit cooperative behavior.

The difference of weak and strong rationality ensues from situations in which defection is as beneficial as the exhibition of cooperative behavior.

[1] The recursion of third party exchange protocols is limited by the available resources.
[2] Implicit reciprocation is provided by the community pattern [2].

Network model. Each exchange protocol is based on one of three *network models*. A network model captures the ability of entities to communicate. In the context of the network models, we refer to the elementary constituents of communication as messages[3]. Hence, the communication is composed of a sequence of message transmissions. Each transmission is directed from the *sender* to the *recipient*. We propose three generic network models that are distinguishable with regard to the types of time guarantees they give:

- **Synchronous network model:** There is a time bound within which the delivery of every message is guaranteed. Therefore, protocol steps do not only depict logical time but they also coupled with the real time. Consequently, protocols for the synchronous network model are often modelled in *rounds* in lieu of steps. In this regard, each round lasts as long as specified by the message delivery time bound plus the maximal time for executing a protocol step.

- **Asynchronous network model:** Each message is eventually delivered. This means that there may not be one time bound within which the delivery of every message can be guaranteed. Consequently, protocols cannot be modelled in rounds of fixed length. Therefore, the concept of rounds is only applicable by lifting the foreseeable interrelation between logical time and real time. As a result, the remainder of this report adheres to the term *step* and uses the term *round* only if the synchronous network model is assumed.

- **Network model without guarantees:** It is not guaranteed that a message is eventually delivered. In general, this means that the recipient of the message has left the network or that the network is perpetually partitioned.

Participants. In general, several peers may participate in an exchange. Such exchange protocols are called *multilateral* exchange protocols. However, an important subset of exchange protocols presumes that exactly two peers participate in the exchange. Such exchange protocols are *bilateral* exchange protocols.

Complexity. The overhead of an exchange protocol is captured by this property. It consists of the quantification of the *steps* and the *messages* that the exchange protocol requires in the absence of failures. This means that the complexity of failure recovery is not considered.

3.2.4 Properties of Third Party Exchange Protocols

The introduction of a third party yields several properties that are specific to third party exchange protocols.

Trust in the third party. Some exchange protocols assume that the third party is *unconditionally* trustworthy. Such trusted third parties (TTP) always behave as specified by the protocol and, thus, are not autonomous. However, other exchange protocols assert that the third party's behavior is verifiable up to a certain degree. Then, the third party only has to be trusted *conditionally*.

[3] Again, the term has to be treated in a broad sense. On the link layer and network layer, a message is a packet.

Optimism. This property describes the involvement of the third party. It may be needed either for every protocol run or only in case of some failure[4]. Such exchange protocols are called *pessimistic* and *optimistic* respectively.

For example, a simple optimistic exchange protocol is as follows. One peer optimistically hands over its item. If the other peer defects by refusing to deliver its item, the defected entity contacts the third party.

3.2.5 Properties of Exchange Protocols that Make Use of Claims

For some items, the benefits that arise from their possession depend on other entities' estimation of their validity. For example, a peer that has acquired a check has to convince its bank of its validity. In this regard, the peer that possesses the item brings in the claim and the bank is the arbiter.

Item verification. The validity of the exchanged items may or may not be verifiable *locally*, i.e., only with the involvement of the entity that brings in the claim and the verifier. If not, verification includes *three parties* since the potential revocation holder has to be contacted in order to get known whether the item has been revoked. Exchange protocols for three party verification are only required for revocable items.

Abortion verification. The third party might issue a certificate of the abortion of the exchange. In most exchange protocols, the third party subsequently sticks to its decision to abort the exchange. For such protocols, it suffices to present an abortion certificate in order to prove that the exchange was aborted. Hence, the abortion of an exchange can be verified *locally*. Still, in some exchange protocols, the third party is able to override such abortion certificate if it figures out that the abortion certificate was handed out to a defecting peer. Consequently, the verification of an exchange's abortion might include *three parties* since the potential *affidavit*[5] holder has to be contacted.

3.2.6 Overview of the Properties of Exchange Protocols

We have proposed a set of properties of exchange protocols. Table 3.1 gives an overview of these properties and their values. For enumerable values, the table introduces capital letters.

Some properties are given a priori by the transaction *environment*. Such properties are the network model, the item properties, and the number of participants. The remaining properties only depend on the *preferences* of the transaction peers. These properties may be regarded as the peers' preferences with regard to the properties of the exchange protocol. For example, a risk averse entity might demand for a fair processing protocol.

3.3 Instances of Transaction Protocols

In the following, we take a closer look at the most common instances of transaction protocols. In order to use a terminology that is consistent with the literature, we refer to the protocols again as exchange protocols.

[4]Except from defection, a failure might occur in the context of the asynchronous network model due to timeouts.
[5]Affidavits are introduced in [3]. The affidavit that is referred to here is an overriding affidavit.

Table 3.1: Overview of the properties of exchange protocols

Property	**Values**	**Type**
Fairness	none (**N**), weak (**W**), strong (**S**)	Preferences
Rationality	none (**N**), weak (**W**), strong (**S**)	Preferences
Network model	synchronous (**S**), asynchronous (**A**), without guarantees (**W**)	Environment
Participants	multilateral (**M**), bilateral (**B**)	Environment
Complexity	number of steps and messages	Preferences
Trust in the third party	conditional (**C**), unconditional (**U**)	Preferences
Optimism	optimistic (**O**), pessimistic (**P**)	Preferences
Item verification	local (**L**), three-party (**T**)	Preferences
Abortion verification	local (**L**), three-party (**T**)	Preferences

3.3.1 Asokan's Exchange Protocol

An optimistic exchange protocol for forwardable items is proposed in [4]. The asynchronous network model is assumed. The exchange protocol fulfills the weak fairness criterion. Furthermore, the validity of the items is locally verifiable.

In order to achieve weak fairness, a trusted third party may have to issue an *attestation affidavit* of the peer's defection. The exchange protocol is optimistic since the third party is only contacted in case of defection.

For generatable items, the exchange protocol is extended in order to enforce strong fairness. In the following, the ensuing exchange protocol is called *Asokan's exchange protocol*. In a nutshell, the trusted third party is able to generate the item by using the permit and, thus, does not have to issue the attestation affidavit. The protocol steps are as follows:

1. Peer **A** passes on a permit of its own item **a** to peer **B**.
2. Peer **B** passes on a permit of its own item **b** to peer **A**.
3. Peer **A** hands over item **a** to peer **B**.
4. Peer **B** hands over item **b** to peer **A**.

Let us consider the case that peer **B** omits step **(2)**. It directly turns to the third party in order to acquire item **a**. Yet, the exchange protocol specifies that the third party only generates items if it has got the permits of both items. Hence, **B** has to give the permit of item **b** to the third party in order to acquire item **a**. After having waited for the message of step **(2)** for a certain time, peer **A** turns to the third party. Since peer **B** already possesses item **a**, peer **A** has to acquire item **b** in order to enforce strong fairness. The third party has a permit of item **b**. Thus, it generates item **b** and hands it over to peer **A**. On the other hand, if peer **A** contacts the third party before peer **B**, the exchange is aborted. In this regard, the third party does not generate item **a** even if peer **B** presents both permits. Consequently, Asokan's exchange protocol calls for a *state keeping*[6] third party.

[6] A more detailed discussion of the requirement that the third party keeps the state of an exchange is given in [8].

3.3.2 Optimal Strongly Fair Exchange Protocols

A survey of strongly fair contract signing protocols that make use of a third party is given in [8]. Since contracts are forwardable and potentially generatable items, the suggested protocols might be regarded as exchange protocols for such items. The following discussion of the proposed protocols takes such a generic view.

The proposed protocols terminate and are strongly fair. The authors identify different settings with regard to the network model (synchronous vs. asynchronous), optimism (pessimistic vs. optimistic), and item verifiability (local vs. three parties). In the following, we structure the triples that define the setting as *(network model, optimism, item verification)*. For each setting, the authors propose a protocol and prove its best case optimality with respect to the messages sent or the involved steps. In this context, the term best case refers to a protocol run during which no failure arises. The protocols for the most important settings are as follows[7]:

- **(A,P,L):** Each peer hands over its item to the third party. Only if it received both items, the third party passes on the items to their destined peer. The protocol requires at best four messages and two steps.
- **(A,O,L):**
  - **Message optimal protocol:** The message optimal protocol requires four messages in as many steps. It is identical to Asokan's exchange protocol [4] described before.
  - **Step optimal Protocol:** The step optimal protocol requires at best three steps and six messages. The steps are as follows:
    1. Each peer sends the permit of its item.
    2. Each peer commits to the successfulness of the exchange by signing both permits together. Thus, each peer gives up the right to turn to the third party and demand the abortion of the exchange.
    3. Each peer hands over its item.

    The third party generates an item only on presentation of its permit and the item owner's abandonment of the right to abort the exchange. The need for step **(2)** arises from the following consideration. Peer **A** might refuse to hand over its item **a** in step **(3)**. Yet, it might still receive peer **B**'s item **b**. Then, peer **A** turns to the third party in order to abort the exchange and pretends that it has not received any message from peer **B** in the preceding steps. If the third party has not been contacted by peer **B** before, it issues an abortion certificate. Therefore, peer **A** possesses item **b** and the abortion certificate of the exchange. In order to recover strong fairness by the means of generatability, peer **B** has to acquire item **a** and an overriding affidavit when contacting the third party. Apparently, this exchange protocol calls for three party verification of abortion certificates.
- **(S,O,L):**
  - **Message optimal protocol:** The message optimal protocol requires at best three messages in as many rounds. The protocol corresponds to the steps **(2)**-**(4)** of Asokan's exchange protocol:

[7] For clarity reasons, the triples use the initial letters of the properties, as they are introduced in Table 3.1.

1. Peer **A** passes on a permit of its own item **a** to peer **B**.
2. Peer **B** hands over item **b** to peer **A**.
3. Peer **A** hands over item **a** to peer **B**.

Since peer **A** acquires item **b** in step **(2)**, it is not allowed to abort the exchange. Otherwise, strong fairness cannot be asserted any more. Instead of that, peer **A** makes use of the synchronous network model in order to recognize when the transaction is aborted. As a prerequisite, the third party resolves failures differently. If peer **B** has not received item **a** in round **(4)**, it shows to the third party item **a**'s permit and item **b**. The third party generates item **a** and hands it over to peer **B**. In addition, the third party proactively hands over item **b** to peer **A**. If peer **A** has not received item **b** until round **(5)**, it knows that peer **B** has not obtained item **a** and, thus, that the exchange is not successful, i.e., it has been aborted. Apparently, the protocol requires the synchronous network model. On the one hand, the third party may only generate item **a** before round **(5)**. On the other hand, peer **A** knows whether the transaction is successful after round **(5)**. As a result, the protocol is not applicable to the asynchronous network model.

- **Round optimal Protocol:** The round optimal protocol requires at best two rounds and four messages. In the first round, each peer sends a permit of its item. The second round consists of handing over the item. Therefore, the second round of the corresponding protocol for the asynchronous network model is omitted. In analogy to the message optimal exchange protocol, this is facilitated by eliminating the need for the abortion of the exchange by the third party.

- **(S,O,T):** The round and message optimal protocol requires at best two messages in one round. The peers exchange their items in the first round. If a peer **A** has not received the item **b**, it contacts the third party in order to acquire a certificate of its own item **a**'s revocation. The third party first tries to obtain item **b** from peer **B**. If peer **B** hands it over, the third party passes it on to peer **A**. In other words, the third party gives a second chance to **B** to hand over item **b**. This is necessary since **A** might have falsely claimed that it did not receive item **b**. However, if peer **B** refuses to hand over item **b** to the third party, a revocation certificate for item **a** is issued by the third party. Apparently, the protocol requires the synchronous network model. Otherwise, the third party cannot perceive whether peer **B** will answer to its request. Furthermore, a peer knows that the acquired item is valid if it is not contacted by the third party in round **(3)**.

- **(A,O,T):** The message optimal protocol requires at best three messages. The protocol's steps are identical to the message optimal protocol for the **(S,O,A)** setting. However, the third party handles failure resolution as specified in Asokan's exchange protocol. The only remaining problem is how to deal with peer **A**'s request for abortion after it has received item **b** and before peer **B** contacted the third party for exchange resolution. For this purpose, the third party issues a revocation certificate of item **b** if peer **A** requests for exchange abortion. Since peer **B**'s request for exchange resolution will be eventually obtained by the third party, peer **B** is able to acquire the revocation certificate.

3.3.3 Asokan's Multilateral Exchange Protocol

An optimistic third party exchange protocol for multiple peers is proposed in [9]. The protocol allows for the most general item exchange since every pair of participating peers may exchange

items. Strong fairness is only asserted if the items are generatable or revocable. Otherwise, only weak fairness is enforced. In any case, the protocol assume that items are forwardable. The exchange protocol consists of two rounds[8]:

1. Each peer **A** sends to every other peer **B** the commitment token of item $\mathbf{a}_b$.
2. For every peer **B**, each peer **A** hands over item $\mathbf{a}_b$.

Apparently, the exchange protocol extends the round optimal bilateral **(S,O,L)** exchange protocol. The assumption of the synchronous network model is crucial for the protocol. The third party is able to resolve failures by contacting certain peers. This is not possible in the asynchronous network model[9]. Furthermore, fairness may only established within a predefined time interval. Exchange abortion or resolution by the third party has to be completed in this time interval. Consequently, the peers and the third party possess synchronized clocks and the messages they interchange are delivered within specific time bounds.

3.3.4 Buttyan's Protocol

A bilateral protocol for checks is proposed in [7]. In the following, it is called *Buttyan's protocol*. The assertion of rationality is based on the following concept. The agent determines the amount that is withdrawn from the principal's account, whereas the principal decides about the amount that is added to the agent's account. In this regard, each peer cannot decide whether its account is credited/charged.

For clarity reasons, the following description of the protocol assumes that the accounts are jointly managed by a bank[10]:

1. The principal hands over a deposit check[11] to the agent.
2. The agent executes the action.
3. The principal hands over an ordinary check that corresponds to the deposit check.
4. The agent presents this check and its account is credited.

The protocol is pessimistic since the bank constitutes a third party. The involvement of the bank ensues from the application of checks as the remuneration type [2].

If the execution of the transaction protocol does not incur any costs, e.g., for the communication, it is shown that the proposed protocol is rational. However, we have to note that it is not strongly rational. In order to show this, we take a closer look at step **(3)**. The principal has no incentive to hand over the check since the action has already been executed and the check's value equals the one of the deposit check.

It might be argued that the value of the deposit check should be increased in order to make the processing of step **(3)** beneficial for the principal. However, this implicates that step **(3)**

[8] In [9], the authors describe three rounds. Their first rounds deals with achieving a global view of which items will be exchanged. However, the protocol descriptions of this section assume that each party knows about this from the beginning.

[9] For the asynchronous network model, it has been shown [8] that recovery from protocol failure may only include the third party and the peer that contacted it. This restriction does not hold for the synchronous network model since the third party is able to recognize whether the second peer is willing to respond.

[10] The protocol is only based on the assumption that the banks are able to communicate.

[11] A deposit check assumes the role of a deposit since the agent cannot cash the check at the bank. Upon presentation of the deposit check, the bank would only debit the principal's account.

is processed regardless of the effectiveness of step **(2)**. Then, the processing of step **(2)** is not beneficial for the agent. Therefore, increasing the deposit check's value renders the processing protocol even irrational.

In this regard, the assumption of protocol execution without incurring costs becomes critical. Even if the costs of each step are infinitesimal, the protocol ceases to be rational. As it becomes clear from the upper discussion of achieving a strongly rational protocol, even an increase of the deposit check's value cannot restore the property of rationality.

Generally speaking, deposits increase the defection costs so that transactions are processed to the end. However, in such settings, the agent and the principal tend to declare every transaction successful in order to get back the deposit.

3.3.5 Syverson's Protocol

Syverson's protocol of [10] applies weakly secret bit commitment. The protocol relies on the following assumptions:

- The action solely consists of returning a value **v**. In this regard, the action is forwardable.
- There exists a well-known temporarily secret bit commitment function **w**.
- If the principal acquires **w(v)**, it is not induced to compute **v**. Therefore, the principal valuates the remuneration that it has to hand over less than the timely receipt of the return value or the costs for computing it.
- After having disclosed **w(v)** to the principal, the agent does not have any disadvantage from disclosing **v** to the principal.
- The processing of protocol steps does not incur any costs.

Based on these assumptions, the processing protocol is presented. It consists of three steps:

1. The agent executes the action and, thus, obtains the return value. It hands over **w(v)** to the principal.
2. The principal hands over the remuneration.
3. The agent discloses the return value to the principal.

The need for the principal's valuation of time or computation becomes obvious in step **(2)**. The principal processes this step only if the assumption holds. In [7], it is proved that Syverson's processing protocol is weakly rational. Apparently, it is not strongly rational. For instance, the agent has no incentive to execute step **(3)**. In analogy to the discussion for Buttyan's processing protocol, it can be shown that Syverson's processing protocol ceases to be rational if the processing of protocol steps incurs costs.

3.3.6 Classification

The discussion of the exchange protocols is summarized in Table 3.2. The characteristics that apply to the protocols are marked by an **x**. The protocols' properties are indicated by the initial letters, as they were introduced in Table 3.1.

Table 3.2: Comparison and classification of existing exchange protocols

Protocol	Com - plexity		Items				Assertions		Further properties				
	Steps	Messages	forwardable	generatable	revocable	divisable	fairness	rationality	netw. model	item verif.	abort. verif.	optimism	trust in TP
Optimal (A,P,L)	2	4	x	-	-	-	strong	-	A	L	L	P	U
Message optimal (A,O,L) (Asokan)	4	4	x	-/ x[1]	-/ x[1]	-	weak/ strong [1]	-	A	L/ T[1]	L	O	C
Round optimal (A,O,L)	3	6	x	x	x	-	strong	-	A	L	T	O	U
Message optimal (S,O,L)	3	3	x	x	-	-	strong	-	S	L	L	O	U
Round optimal (S,O,L)	2	4	x	x	-	-	strong	-	S	L	L	O	U
Message & round optimal (S,O,T)	1	2	x	-	x	-	strong	-	S	T	L	O	U
Message optimal (A,O,T)	3	3	x	x	x	-	strong	-	A	T	L	O	U
Asokan's multi - party exch. prot.	3	2*n* (n-1)	x	-/ x[1]	-/ x[1]	-	weak/ strong [1]	-	S	L/ T[1]	L	O	C
Asokan's contract signing	4	4	x	x	-	-	strong	-	A	L/ T[1]	L	O	C
Buttyan's processing prot.	4	4	-/ x[3]	-	-	-	-	weak [4]	A	L	-	P	U
Syverson's processing prot.	3	3	x	-	-	-	-	weak [4]	A	L	-	-	-

[1] Strong fairness is asserted if the items are generatable or revocable. If the third party is allowed to revoke items, the validity of items involves three parties.
[2] The account manager (bank) constitutes the third party. The protocol assumes that the bank cannot verify the validity of the agent's item, i.e., effectiveness of the action.
[3] The check has to be forwardable.
[4] Only if the costs of sending messages are negligible.

3.4 Applicability to the System Model

In the previous section, we have discussed several internet based protocols for performing transactions. To which degree are they applicable to our setting of Chapter 2? In order to answer this question, we have to revisit the relationship between our setting and the setting of internet based eCommerce.

According to Section 3.1, there four differences between the settings. In the following, we discuss their impact on the applicability of internet based transaction protocols.

Infrastructure. In a self-organizing system, there are neither law enforcing entities nor third parties that are commonly known or trusted. This yields the following consequences:

- **Self-organizing punishment:** In the absence of law enforcing entities, we need a means of punishing defections in a self-organizing manner. Ultimately, this means has to be established in order to ensure sufficiently high defection costs.

- **Self-organized choice of third parties:** If transaction protocols are still to be based on third parties, the third party has to be chosen among the set of potentially equal entities. The transaction peers may choose the third party as they like to.

- **Coping with untrustworthy third parties:** We need a new means of dealing with potentially untrustworthy third parties. Even if the transaction peers choose an entity that seems reasonably trustworthy to both of them, full trustworthiness cannot be guaranteed. In this context, the approaches towards conditional trust of Section 3.3 are not useful since they are only applicable to generatable items[12].

  It seems promising to punish misbehaving third parties in a manner that is in line with the means of self-organizing punishment for defecting transaction peers. In such a case, the behavior of third parties would not have to be considered separately.

- **Reciprocation of third parties:** Since third parties are chosen among the potentially equal entities of the system, we are able to solve the problem of reciprocation. More specifically, a third party X may be reciprocated as follows: The transaction peers promise to act as a third party in subsequent transactions in which the entity X acts as a transaction peer. If such promise is non-repudiable, it corresponds the incentive pattern of notes [2].

  In systems with dedicated third parties, such reciprocation would not be applicable. This is because such dedicated third parties never take part in transactions in the role of a transaction peer.

- **Applicability of deposits:** Buttyan's protocol is built on the assumption that an action allows for the use of a deposit check. However, the notion of deposit checks is tightly bound to third parties: Let us assume that an entity X wanted to hand over a note to entity Y. The meaning of a deposit note would be as follows: Entity X is committed to execute the promised action in return on demand. However, entity Y is not allowed to request such action before it acquires the note itself. Therefore, the question arises who should be able to demand for the honoring of the deposit note. Such a role would have to be assumed by a third party. It is unclear what happens if an arbitrary third party assumes this role.

[12] The conditionality of trust arises from the invasiveness of generating items [4].

Transaction peers. Since our system model assumes bilateral transactions, we ignore multilateral transaction protocols in the following. It is noteworthy that research of internet based eCommerce has laid the same stress on bilateral transactions.

Item model. According to our setting of Chapter 2, items are neither divisible nor generatable nor revocable. The consequences are as follows:

- **Fairness:** Most protocols require either generatability or revocability in order to assert strong fairness. Otherwise, only weak fairness can be guaranteed[13]. Still, such weak fairness is based on the capability of a trusted third party to issue effectual affidavits. Hence, the absence of trusted third parties implies that even weak fairness cannot be guaranteed.

- **Interpretation of permits:** If items are non-generatable, the internet based protocols propose a different role for permits. A permit becomes a mere commitment item and is, de facto, a contract. More specifically, it is a non-repudiable commitment of its issuer that he agrees to the terms of the transaction.

- **Optimistic protocols:** Optimistic protocols cease to be useful if items are neither generatable nor revocable.

  There is a further reason for turning to pessimistic transaction protocols. In contrast to dedicated third parties, the arbitrarily chosen third parties are close to the transaction peers so that communication with them is as costly as among the peers. Furthermore, third parties do not represent a bottleneck any more since they are arbitrarily chosen.

Reachability. Internet based protocols assume that the network model is at least the asynchronous one. However, our system model presumes the absence of any guarantee regarding reachability, which corresponds to the network model without guarantees. Hence, the termination of the proposed transaction protocols can only be achieved by the means of timeouts. In turn, this means that the protocols may produce reports of defections even if the transaction peers behave well but become unreachable. In the following, we call such reports false positives. This principle is illustrated in Figure 3.1. The problem corresponds to the one of assessing venial (i.e., non voluntary) noncooperation if an entity's will is imperceptible [6].

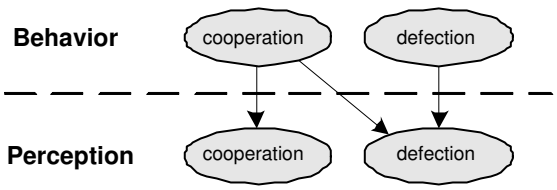


Figure 3.1: Restricted reachability implies a biased discrepancy between behavior and its perception

How may protocols cope with the noisy perception of transactional behavior? Actually, they cannot do anything about it. The means of distributed punishment has to take into account that negative reports may actually not be linked to defective behavior[14].

[13] Actually, weak fairness can only be guaranteed under such circumstances if each transaction peer can be forced to eventually hand over an item, regardless of whether it is the item it committed to or not [4].

[14] The problem of more forgiving punishment has been thoroughly analyzed in game theory [11].

Summary. There are several differences between the setting of internet based eCommerce and our setting of Chapter 2. Most importantly, it appears that transaction protocols cease to be sufficient in order to curb defections. Hence, transaction protocols have to be complemented by a means of self-organizing punishment. We discuss such means and its interrelationship with transaction protocols in the next chapter.

The differences of the settings imply that we will have to propose different transaction protocols for self-organizing systems. In Section 5.4, we will compare these protocols with the internet based protocols of this chapter.

Chapter 4

Self-organized Systems and Transaction Protocols

As we have seen, the absence of infrastructure implies that defective transactional behavior has to be punished in a self-organizing manner. Only if such a self-organizing punishment exists, the identification of misbehavior (by the means of transaction protocols) makes sense.

In this chapter, we take a closer look at distributed reputation systems that allow for self-organizing punishment. More specifically, we discuss conventional distributed reputation systems that are based on plausibility and an evidence-aware extension of such conventional systems. Finally, we stress the role of transaction protocols for evidence-aware distributed reputation systems.

4.1 Conventional Distributed Reputation Systems

A *reputation system* keeps track of defections in order to caution the entities about the defectors. In the absence of any central component, the reputation system is distributed to the entities themselves. More specifically, each entity runs a local instance of the reputation system. These instances may cooperate by exchanging recommendations. By this means, the reputation system provides a means for self-organizing punishment. The considered system is illustrated for two entities in Figure 4.1.

The issuer of a recommendation (*recommender*) communicates the trustworthiness of a certain agent to the *recipient* of the recommendation. Recommendations may be untruthful. Therefore, the recipient has to assess the truthfulness of the recommendation before taking it into account.

The existing approaches for distributed reputation systems (e.g., [12, 13]) make use of plausibility considerations in order to provide for such assessment. This means that the impact of a recommendation is contingent upon its plausibility which, in turn, depends on its compatibility

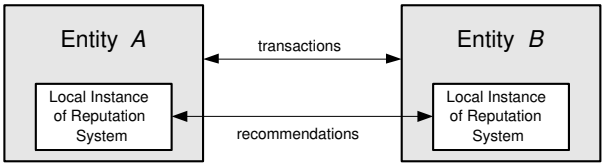


Figure 4.1: Model of a distributed reputation system

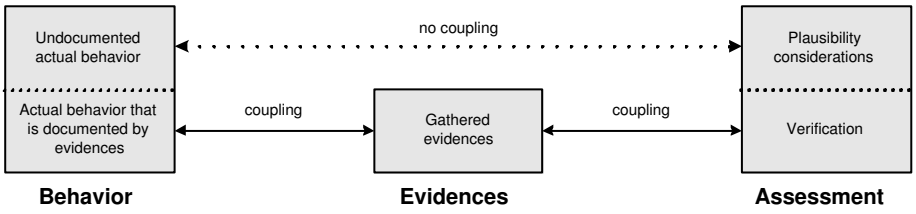


Figure 4.2: Evidences as a means of coupling the assessment of recommendations with the actual underlying behavior

with prior beliefs. More specifically, the considerations comprise two parts. On the one hand, a recommendation is assessed as rather trustworthy if it is compatible to the first hand experiences made by the assessor itself. On the other hand, the more the recommender is trusted the more the recommendation is regarded as truthful. In [3, 14], the limitations of plausibility considerations are identified and discussed. In the next section, we examine an extension that applies non-repudiable tokens in order to overcome most of the limitations.

4.2 Evidence-Aware Distributed Reputation Systems

In [3], it is proposed to make use of evidences in order to overcome the limitations of plausibility considerations. An *evidence* is a non-repudiable token that may be arbitrarily transferred[1]. In contrast to plausibility considerations, the application of evidences achieves a coupling between the actual behavior and the assessment of recommendations about it. This is illustrated in Figure 4.2.

As a prerequisite for the application of evidence-aware distributed reputation systems, evidences have to be collected in the course of transactions. According to [3], the two most important transactional evidences are receipts and contracts. After having agreed on the terms of the transaction, the transaction peers may commit to the terms by mutually issuing non-repudiable commitments (*contracts*). A *receipt* is an evidence that confirms that the transaction partner has actually executed the action it committed to.

4.3 The Role of Transaction Protocols

It has been conjectured [3] that the application of evidences increases the effectiveness of self-organizing punishment. In this regard, the question arises which role remains for transaction protocols.

Architectural overview. In Figure 4.3, we perceive that an entity is decomposed as follows: The decision maker decides with whom to transact under which circumstances. The component that applies transaction protocols processes the transaction and gathers knowledge and evidences of the transaction partner's behavior. This provides the local instance of the reputation system with the track record from which the trustworthiness of other entities.

The architectural overview allows for the identification of the roles of transaction protocols.

[1]The term evidence has been used differently in reputation systems. In [15, 13], it depicts witnessed circumstances, i.e., first hand experiences and recommendations.

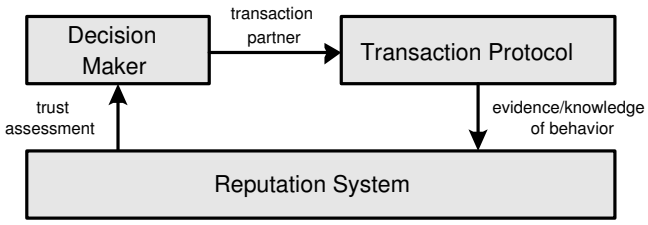


Figure 4.3: Architectural overview of an entity

Role 1: Preemption.
The first role of transaction protocols consists of making some sorts of defection impossible[2]. This is achieved by coupling the transaction steps of the peers. For example, the subsequent execution of actions by the peers implies that only the second peer is able to uni-laterally defect. The benefits of such preemption seem to be marginal. Nevertheless, they should not be underestimated since only the second peer has to be trustworthy.

Pre-transactional and transactional phases. In order to describe the second role of transaction protocols, we discuss the pre-transactional and transactional phases.

After having decided on the transaction partner, an entity processes four phases, as it is shown in Figure 4.4(a):

1. **Agreement on transaction terms:** The entity that has a transaction opportunity chooses a transaction peer and its proposed terms are agreed on (or refused) by the potential transaction peer. This is a pre-phase for the transaction in the sense that the later phases are only processed in case of an agreement. In such a case, the peers mutually know that they are committed to the transaction. Furthermore, this phase is not part of the transaction protocol.

2. **Exchange of contracts:** Each transaction peer signs the agreement and hands over the ensuing contract.

3. **Exchange of actions:** Each transaction peer executes the action it agreed to execute and hands it over. In this regard, the transaction peers comply with their commitment.

4. **Exchange of receipts:** Each transaction peer hands over a non-repudiable receipt that the respective parter has complied with its commitment.

Without the use of evidences (contracts and receipts), participation in transaction only encompasses two phases, as shown in Figure 4.4(b).

From the figures, it appears that evidences provide a means of extending the scope of knowledge: After phase one and three, only the transaction peers know about their respective partner's behavior (i.e., their commitment or compliance). We refer to this situation as mutual knowledge. After phase two and four, the respective knowledge has become globally accessible and comprehensible (due to the transferability of evidences). This kind of knowledge is called global knowledge.

[2]This role corresponds to the approach of direct minimization, as it is described in Section 3.2.

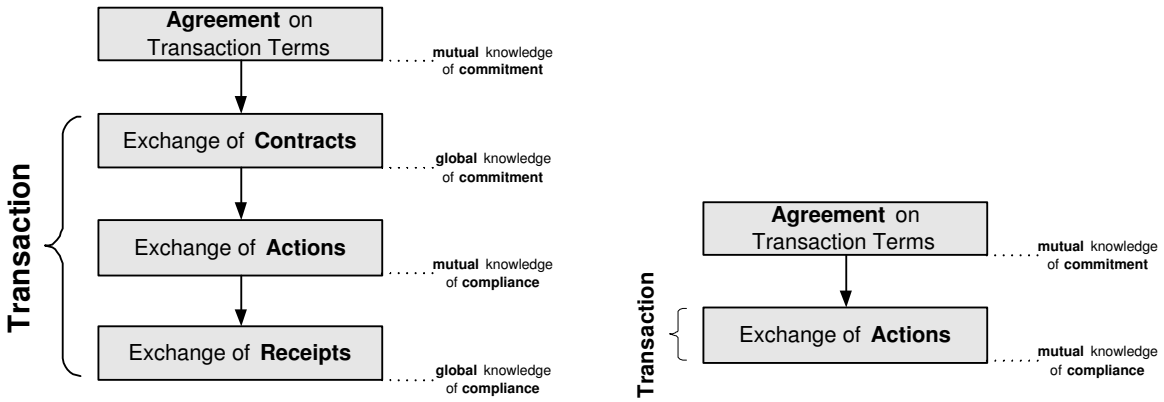


Figure 4.4: The phases of participating in a transaction **(a)** with and **(b)** without evidences.

Role 2: Perception.
The transactional phases of exchanging contracts and receipts enable that mutual knowledge becomes global knowledge[3]. In this regard, the set of entities that are able to perceive transactional behavior is extended from the transaction peers to every entity of the system. Therefore, the second role of transaction protocols is to ensure the perceptibility of transactional behavior. Such perceptibility may leverage the lack of trust between the peers. In addition, perceptibility builds the foundation of achieving weak fairness[4].

We conclude that transaction protocols complement the distributed reputation system since perceptibility of behavior is a prerequisite for self-organizing punishment.

[3]This role of transaction protocols corresponds to the approach of evidence-based minimization, as it is described in Section 3.2.

[4]Ultimately, weak fairness is achieved by the social control of distributed reputation systems.

Chapter 5

Conception of Transaction Protocols for Self-organizing Systems

In the previous chapter, we have explained the role that transaction protocols play in self-organizing systems. They have to preempt defections by coupling transaction steps of the peers. In addition, transaction protocols have furnish evidences of the peers' behavior in order to assert that their behavior is perceptible.

In this chapter, we conceive transaction protocols that comply with these demands. For this purpose, we identify the characteristics of the transaction protocols and, by this means, the design space of transaction protocols. Based on that, we propose five transaction protocols. Subsequently, the key properties of the protocols are illustrated and discussed by a schematic visualization technique. Finally, we interrelate the proposed protocols and their properties with the internet based protocols of Chapter 3.

5.1 Characteristics of Transaction Protocols

Before conceiving transaction protocols, we identify their characteristics. By this means, we are able to capture the design space of transaction protocols. In the following, we discuss four characteristics that have dialectic parameter values.

Bilateral vs. Multilateral. This characteristic captures whether there is a pair of transaction peers (*bilateral*) or whether multiple parties participate in the transaction (*multilateral*). According to the system model of Chapter 2, pairwise transactions are assumed. Therefore, we will only consider bilateral transaction protocols.

Direct vs. Indirect. The communication between the transaction peers may be mediated by a third party. In general, it has no incentive to defect since it is not interested in the exchanged items[1]. We call third party protocols *indirect* transaction protocols[2]. If the protocol is not mediated by a third party, it is called *direct* transaction protocol.

Global vs. Mutual. Figure 4.4 illustrates that the phases for the exchange of contracts and receipts are optional. If they are omitted, the knowledge about commitments and compliance

[1] This line of argument corresponds to the weak rationality criterion of Section 3.2.

[2] In the terminology of Chapter 3, indirect transaction protocols are pessimistic third party protocols.

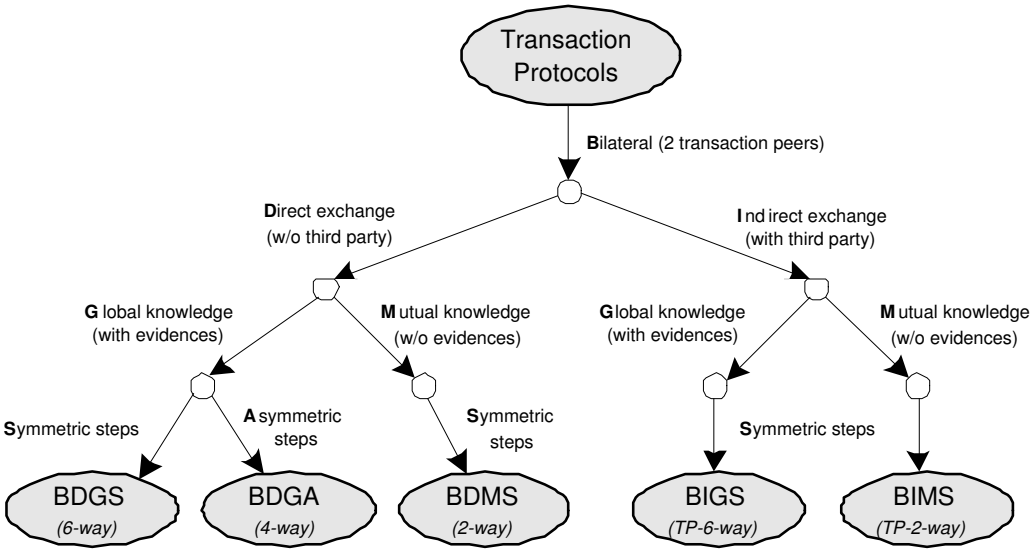


Figure 5.1: Taxonomic classification of the proposed transaction protocols

remains *mutual*. Otherwise, knowledge becomes *global*. Consequently, this characteristic captures whether the transaction protocol stipulates the exchange of contracts and receipts or not.

Symmetric vs. Asymmetric. A transaction protocol is *symmetric* if the transaction peers subsequently process the same protocol steps. In contrast, *asymmetric* transaction protocols contain steps that are not processed by every peer. The nuances of this characteristic are made more obvious by the schematic visualization technique of Section 5.3.

5.2 Proposed Transaction Protocols

According to the design space of transaction protocols, we may anticipate that there exist 8 instances of bilateral transaction protocols. Instead, it appears that only 5 instances make sense. This is because asymmetry does not fit to most parameter values of the other characteristics: **(1)** Mutual transaction protocols solely consist of the exchange of actions. Hence, each peer only processes one protocol step, i.e., the execution and handover of the action. As a result, the transaction protocol cannot be asymmetric. **(2)** Indirect transaction protocols are mediated by a third party. The aim of the third party is to synchronize the peers' protocol steps. Each synchronization restores fairness among the transaction peers. Consequently, each step of the peers has to be symmetric. This rules out asymmetric indirect transaction protocols.

The remaining five instances are taxonomically classified in Figure 5.1. The Appendix A.3 illustrates the protocol diagrams for these protocols. In Section 5.4, we will discuss the interrelationship of the proposed protocols with the internet based transaction protocols.

5.3 Schematic Visualization of Transaction Protocols

In this section, we propose a means of clarifying the characteristics of the transaction protocols. For this purpose, a schematic visualization technique is suggested.

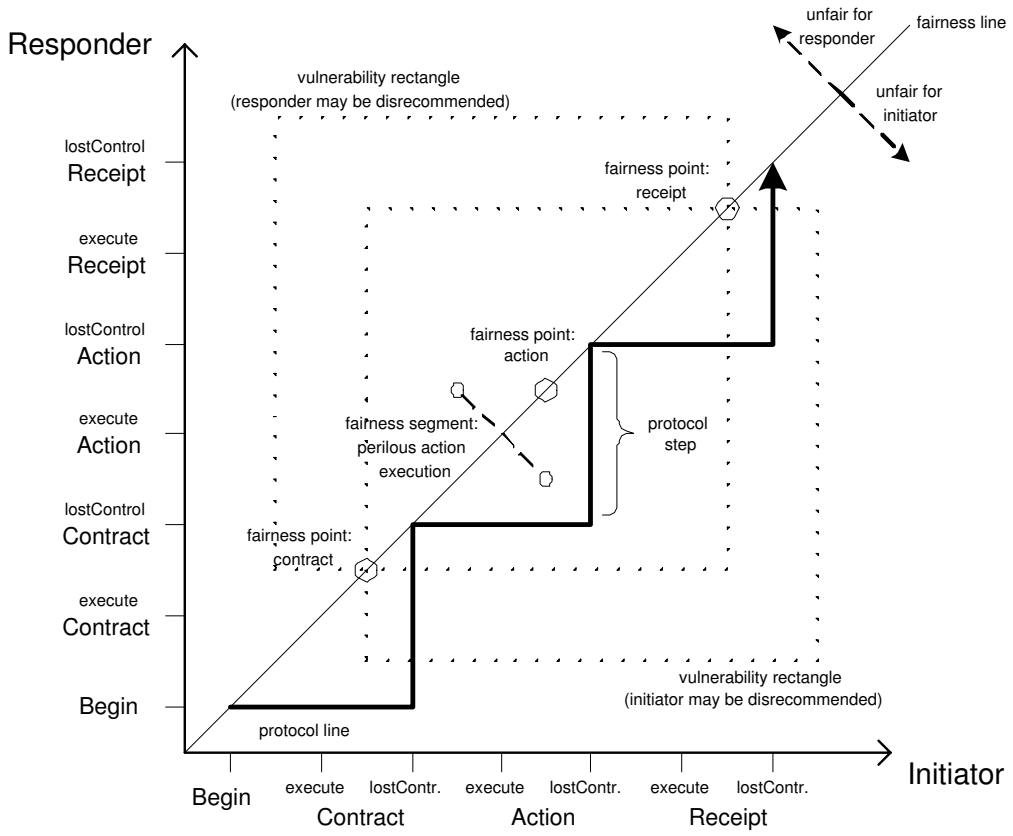


Figure 5.2: A schematic visualization of bilateral transaction protocols with annotation

5.3.1 The Schematic Visualization

Figure 5.2 illustrates the schematic visualization of bilateral transactions protocols. For better intelligibility, the parts of the visualization are annotated. In the following, we discuss each part.

Axes. The x-axis shows the activity of the transaction peer that processes the first step of the transaction (*initiator*). The activity of the other transaction peer (*responder*) is illustrated by the y-axis. For each phase of the exchange, we distinguish between two steps: In the first step, the item is executed. This means that the contract or the receipt is cryptographically signed or that the action is executed. In the second step, the transaction partner acquires the item so that the peer loses its control. For direct protocols, the handover of the item coincides with losing its control. However, for indirect protocols, the control of an item is only lost if the third party forwards it to the transaction partner.

Protocol line. The processing steps of a transaction protocol is shown by a zigzag line between *(Begin,Begin)* and *(lostControlReceipt,lostControlReceipt)*. In the figure, the protocol line of the BDGS protocol is shown. Note that, due to the peers' autonomy, the zigzag line may interrupted at any point. This means that an unsuccessful protocol run produces a prefix of the shown zigzag line.

Each segment of the line is either horizontal (protocol steps of the initiator) or vertical (pro-

tocol steps of the responder). Hence, the number of horizontal (vertical) segments corresponds to the number of protocol steps that the initiator (responder) has to process. For the BDGS protocol, each transaction peer has to process three protocol steps.

In case of an indirect transaction protocol, we assume a well-behaving third party for drawing the line. This makes sense since, in the 2-dimensional visualization, we focus on the protocol steps of the transaction peers. For indirect protocols, the processing order of the transaction peers is not defined. Hence, both possible orders are illustrated. Figure 5.3(b) shows the visualization of an indirect protocol.

Fairness line. The fairness line shows at which points the initiator and the responder have processed the same steps. This means that the situation is fair whenever the protocol line intersects with the fairness line. The more the protocol line deviates from the fairness line in the lower right (upper left) direction the more the situation is unfair for the initiator (responder). Therefore, a comparison of the protocol line and the fairness line shows which peer is in an unfair position to which degree and at which steps.

Fairness points. There are three characteristic points on the fairness line. They mark a situation in which both transaction peers are about to lose control of their contracts, actions or receipts respectively. The comparison of the fairness points with the protocol line shows which peer loses the control of its item first. If the protocol line passes the fairness point on the right side, the initiator is disadvantaged, whereas the responder is disadvantaged if the fairness point is passed on the left side. Apparently, the BDGS-protocol line always passes on the right side of the fairness points. Hence, it disadvantages the initiator with respect to the contract, the action and the receipt.

Fairness segment (perilous action execution). The execution of an action is perilous if the executing transaction peer is not guaranteed to gain control of the transaction partner's action. Since the execution of actions is inherently locally, it is impossible to make the action executions of both peers non-perilous. Therefore, the action execution is either unilaterally perilous or mutually perilous. From a fairness point of view, mutual perilousness is desirable. If the protocol line intersect the fairness segment (of perilous action execution), the action execution is indeed mutually perilous. If the protocol line passes the segment on the right (on the left), only the action execution of the initiator (of the responder) is perilous. The BDGS-protocol line passes the segment on the right. Hence, the execution of the initiator's action is perilous, whereas the responder does not experience such perilousness.

Vulnerability rectangles. An important property of transaction protocols is at which protocol steps which transaction peer is vulnerable to disrecommendations by its transaction partner. According to the norms of the evidence-aware distributed reputation system [3], an entity may only be disrecommended if it has issued a contract. However, after having received the peer's receipt, a transaction peer may refute the disrecommendation. Therefore, a transaction peer can only be effectively disrecommended after having issued the contract and before gaining control of the peer's receipt. In the figure, the areas of vulnerability are illustrated for the initiator and the responder. They are rectangles. The intersection of the rectangles indicates situations in which both peers may be disrecommended. Yet, there are areas in which only one transaction peer can be disrecommended.

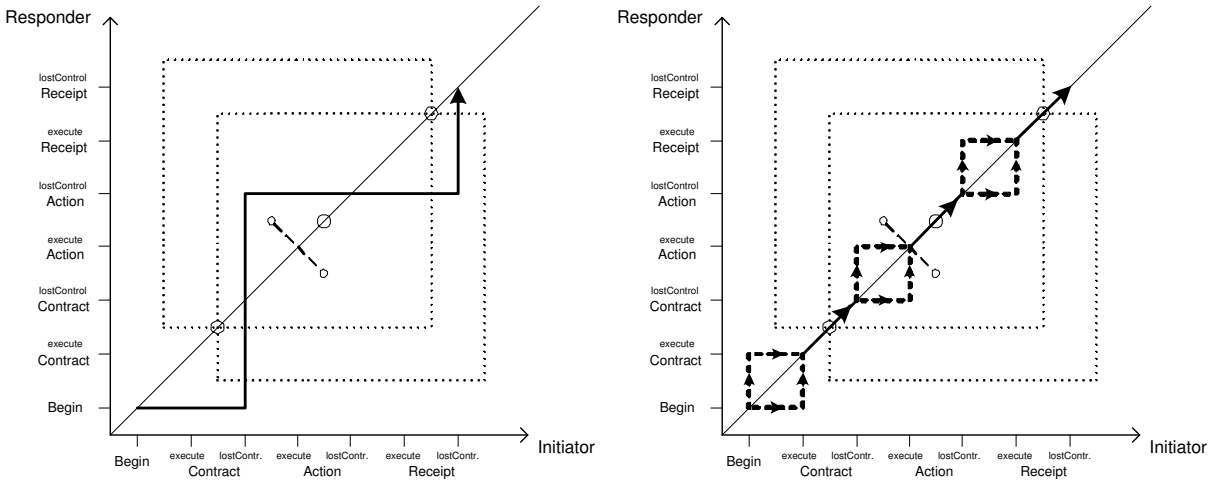


Figure 5.3: A schematic visualization of **(a)** the BDGA protocol and **(b)** the BIGS protocol

5.3.2 Discussion of the Protocols' Visualizations

In Figure 5.2 and Figure 5.3, the schematic visualizations of the BDGS protocol, the BDGA protocol and the BIGS protocol are shown. The visualization of mutual transaction protocols is omitted since it is easily derived from their global counterparts[3]. In the following, we stress the key properties of the visualized protocols:

Protocol steps. For the BDGS protocol and BDGA protocol, each transaction peer has to process three and two protocol steps[4] respectively. Since the diagonal segments of the BIGS protocol line are third party steps, there are also three segments and, thus, three protocol steps for each transaction peer in the BIGS protocol.

Fairness line, fairness points, and fairness segment. In the BDGS protocol, only the initiator is burdened with unfairness. The responder may wait until gaining control of the initiator's item before executing its own item.

For the BDGA protocol, the situation is mixed. Although the initiator still has to precede with the contract and the receipt, the responder bears the unfairness of losing control of its action first. This is shown in the visualization since the protocol line passes the fairness points on the right and, subsequently, on the left and, lastly, on the right. Not only the action fairness point is passed on the left. The same applies to the fairness segment of perilous action execution. Therefore, only the execution of the responder's action is perilous.

The visualization of the BIGS protocol illustrates the desirable properties of indirect protocols if a well-behaving third party is available. Each fairness point is passed simultaneously. This is because the third party is supposed to forward either both items simultaneously or none of them. Furthermore, the protocol line intersects with the fairness segment of perilous action execution. This means that the actions of both the initiator and the responder are executed perilously. As

[3]The visualizations for the BIMS protocol and the BDMS protocol are the action specific portions of the BIGS protocol and BDGS protocol respectively.

[4]In contrast to the notion of steps of Section 3.2 that includes every transaction peer, we refer to protocol steps as the steps that one transaction peer has to perform.

Transaction Protocols / Key Properties	Bilateral Transaction Protocols				
	Direct Exchange			Indirect Exchange	
	Global Knowledge		Mutual Knowledge	Global Knowledge	Mutual Knowledge
	Symmetric Steps	Asymmetric Steps	Symmetric Steps	Symmetric Steps	Symmetric Steps
Protocol Name	BDGS	BDGA	BDMS	BIGS	BIMS
Protocol Steps per Peer	3	2	1	3	1
Unfairness of Contracts for	initiator	initiator	-	*none*	-
Unfairness of Actions for	initiator	responder	initiator	*none*	*none*
Unfairness of Receipts for	initiator	initiator	-	*none*	-
Perilous Action Execution for	initiator	responder	initiator	init./ resp .	init./ resp.
First to enter the Vulnerability Box	initiator	initiator	-	init./ resp .	-
Last to leave the Vulnerability Box	initiator	initiator	-	init./ resp .	-

Table 5.1: Summary of the key properties of the transaction protocols

a result, unfairness may arise if a peer executes its action, whereas the other peer refrains from the execution of its action. Although this is harmful for the compliant peer, the defecting peer does not gain any advantage from such behavior since it is denied the control of the action of the compliant peer[5].

Vulnerability rectangles. The visualization emphases that, with regard to the vulnerability to disrecommendations, the properties of the BDGS protocol and the BDGA protocol are the same. In both protocols, the initiator enters the vulnerability box as the first and leaves it as the last. For the BIGS protocol, the visualization illustrates that the transaction peers enter and leave the vulnerability rectangle at the same time. Therefore, the transaction protocol ensures fairness with regard to vulnerability, as long as the third party behaves well.

Summary. In Table 5.3.2, the key properties of the transaction protocols are summarized. The values of the table are directly derived from the discussion of the protocols' schematic visualization. The key properties of mutual transaction protocols are included in the table even though we do not present and discuss their visualization. This is possible since mutual and global transaction protocols have a corresponding visualization and, thus, corresponding key properties with regard to the exchange of actions.

5.3.3 Visualization of the Symmetry of a Transaction Protocol

A side-effect of the schematic visualization is that the characteristic of symmetry versus asymmetry becomes clearer. For a symmetric protocol, the protocol line consists of an alteration of horizontal and vertical segments that stand for equivalent protocol steps. In Figure 5.2 and Figure 5.3, the protocol lines of both the BDGS protocol and the BIGS protocol comply with this

[5]This line of argument is based on the criterion of weak rationality. We have used such argumentation before in order to answer the question why the third party should forward the items it receives. Apparently, third party protocols are tightly bound to the criterion of weak rationality.

demand. In contrast, the protocol line of asymmetric protocols consists of vertical (horizontal) segments that are not matched by corresponding horizontal (vertical) segments. For example, in the BDGA protocol, the responder's step from generating the contract to losing control of its action has no correspondence for the initiator.

5.4 Interrelation with Internet based Protocols

In the previous sections, we have proposed and discussed several protocols for self-organizing systems. In the following, we discuss their interrelationship with the internet based protocols. The discussion is structured with respect to several aspects. It is more specific than the setting-based discussion of Section 3.4.

Item verification. The verification of items can always be performed locally. For contracts and receipts, this stems from the properties of their non-repudiability. Actions are always verified locally due to the assumption of deliverability, as introduced in Section 2.2.

Abortion verification. The proposed protocols do not consider the use of third party statements regarding the outcome of the transaction. The use of affidavits for this purpose has been proposed in [3]. The inclusion of this approach into the indirect transaction protocols is future work. Nevertheless, affidavits are less effective in self-organizing systems since we cannot rely on the trustworthiness of the third party.

Combination of global and indirect protocols. For internet based eCommerce, it does not make sense to make use of the BIGS protocol that is both global and indirect. This is evinced by the fact that the (A,P,L) protocol of Section 3.3 does not make use of contracts and receipts. The reason is that, in the presence of a trusted third party, global knowledge can be achieved by the issuance of affidavits.

Relevance of the trivial protocol. The BIMS protocol represents the trivial transaction protocol since the transaction peers exchange their actions directly and only with mutual knowledge. The protocol is not considered in academic work on internet based eCommerce. This is because the suggested internet based protocols aim at asserting fairness or rationality. Neither criterion can be fulfilled by the BIMS protocol.

However, the BIMS protocol makes sense for the setting of this report because some entities may be highly overhead-averse or certain about their partner's trustworthiness.

The use of receipts. In internet based eCommerce, receipts do not play such an important role as contracts[6]. Yet, in self-organizing systems, the use of receipts is crucial since they except a transaction peer from being punished. The other way round, receipts are used in order to self-recommend by making one's own good behavior perceptible.

[6]Still, a receipt-based extension of Asokan's protocol has been proposed in [4]. In a fifth protocol step, peer **A** hands over a receipt. Yet, peer **B** is not supposed to hand over a receipt because it is assumed that its action serves as receipt.

Protocol correspondences. If we abstract from some protocol steps or re-interpret them, we are able to identify correspondences between the proposed protocols and the internet based protocols. In the following, we do so and clarify the necessary abstractions and re-interpretations:

- **BDGS protocol and Asokan's protocol:** If we abstract from the exchange of receipts and interpret permits as mere contracts[7].
- **BDGA protocol and the (S,O,L) protocol:** The same abstraction and re-interpretation as above. The correspondence with a protocol that assumes the synchronous network model is unproblematic. This is because the synchronousness is only required for generatability. Hence, the BDGA protocol is not bound to the synchronous network model.
- **BIMS protocol and the (A,P,L) protocol:** The protocols are identical.

Rational protocols. Apparently, the proposed protocols are neither based on Buttyan's nor on Syverson's protocol. This stems from our incertitude regarding the appropriateness of deposits and weakly secret bit commitment. For Buttyan's protocol, it is unclear whether deposit notes should benefit an arbitrary third party. For Syverson's protocol, the difficulty of computing the actual return value has to be adjusted to the transaction peers' utility structure.

[7] Apart from being non-repudiable, a permit ensures the generatability of an item. This is not the case for contracts.

Chapter 6

The Tradeoffs of the Transaction Protocols

In the previous chapter, we have identified and analyzed several transaction protocols for self-organizing systems. The problem that remains to be resolved is the following: Given the circumstances of a transaction, which transaction protocol and which role allocation is the most appropriate one?

In this chapter, we solve this problem in two steps. First, we categorize the properties of transaction protocols. The categories either indicate the benefits or the costs of transaction protocols. Second, the protocols' key properties are captured by quantifying to which degree the respective protocol is associated to each category. As a result, it becomes possible to assign benefits, costs, and a residual utility to a transaction protocol for a specific set of circumstances. Consequently, the transaction peers may choose the transaction protocol that has the highest residual utility. Finally, we illustrate the principles of tradeoff-aware choice. For this purpose, we show the dominance graphs for the proposed transaction protocols and roles.

6.1 Benefit and Cost Categories

In order to choose transaction protocols with respect to a specific transaction environment, we have to develop the notion of appropriateness for transaction protocols. In Section 5.3, the key properties of the transaction protocols have been identified. In the following, we associate these key properties to benefit categories and cost categories. By this means, we valuate the significance of the key properties both qualitatively and quantitatively.

6.1.1 Benefit Categories

The benefits of a transaction protocol consist of its assertions. Based on the numerous key properties that are identified in Section 5.3, we propose an aggregation to three categories regarding such assertions. It is clear that the quantification of these categories depends on the role of the respective transaction peer.

Furthermore, we have to take into account the trustworthiness of the third party for the quantification of the categories. The trustworthiness is denoted as $\mathbf{t}_{TP}$ and is defined as the probability that the third party behaves correctly. There are two kinds of misbehavior by a third party: **(1)** It may retain an item or the pair of items. **(2)** It may forward an item even though the other item has not been received.

In the following, we introduce the three benefit categories.

Sufficiency of own action execution.

- **Definition:** If a transaction peer executes its own action, is it certain to gain control of the action of the transaction partner? In such a case, the execution of one's own action is sufficient in order to gain such control.

- **Quantification:** Full sufficiency is quantified as **1**, whereas full absence of sufficiency is quantified as **0**. According to the schematic visualization, sufficiency exists if and only if the execution of one's own action is not perilous.

Sufficiency of losing control of one's own action.

- **Definition:** If a transaction peer loses control of its own action, is it certain to gain control of the action of the transaction partner? In such a case, losing control of one's own action is sufficient in order to gain control of the partner's action.

- **Quantification:** Full sufficiency is quantified as **1**, whereas full absence of sufficiency is quantified as **0**. According to the schematic visualization, sufficiency exists if and only if one's own role does not suffer unfairness with respect to the actions. For indirect protocols, the additional condition of well-behavior by the third party has to be fulfilled. Since the fairness point of actions is always traversed for indirect transaction protocols, their quantification of this category is $\mathbf{t}_{TP}$.

Deterrence.

- **Definition:** If a transaction peer is defected, to which degree may it take revenge based on self-organized punishment? We distinguish between two types of deterrence that are quantified separately.
  - **Knowledge by a third party:** Is a third party able to fully perceive the defection?
  - **Ability to recommend:** May the defector be disrecommended?

- **Quantification:** We propose the following weighting[1]:
  - **Knowledge by a third party:** If fully yes **0.3**, otherwise **0**. This means that the quantification is **0.3*t**$_{TP}$ for indirect protocols.
  - **Ability to recommend:** If fully yes **0.7**, otherwise **0**. The quantity is the sum of the following:
    - **Disrecommendation for defection on action:** If the transaction partner does not provide its action, oneself may have a contract that allows for disrecommending it. This holds for global protocols (quantification **0.5**), whereas, for mutual protocols, disrecommendations are not possible (quantification **0**).

[1]The weights are set rather arbitrarily. A thorough quantification of the weights is interesting yet future work.

* **Untruthful disrecommendation:** If the transaction partner got one's own contract without losing control of its own contract, only the transaction partner could disrecommend oneself. According to the schematic visualization, this holds for an unfair position regarding contracts. In such a case, the quantification is **0**. If one also has the contract of the transaction partner, the quantification is **0.1**. For indirect protocols, the quantification is **0.1*t**$_{TP}$ since they are fair if the third party is well-behaving.
* **Disrecommendation for defection on receipts:** If the transaction partner got one's own receipt without losing control of its receipt, only the transaction partner cannot be disrecommended any more. According to the schematic visualization, this holds for an unfair position regarding receipts. In such a case, the quantification is **0**. If one also has the receipt of the transaction partner, the quantification is **0.1**. For indirect protocols, the quantification is **0.1*t**$_{TP}$ since they are fair if the third party is well-behaving.

6.1.2 Cost Categories

The cost categories capture which kinds of costs the processing of the transaction protocol incurs on the transaction peers. We distinguish between three cost categories; all of them are bound to transaction roles:

Number of messages.

- **Definition:** How many messages does a transaction peer have to send[2]?
- **Quantification:** If the transaction peer is the initiator (responder), this number corresponds to the number of horizontal (vertical) segments of the protocol line of Section 5.3.

Number of sign operations.

- **Definition:** How many times does a transaction peer have to issue an evidence and, thus, execute a sign operation?
- **Quantification:** In mutual transaction protocols, no evidences are issued. In global transaction protocols, each peer issues exactly two evidences, i.e., a contract and a receipt.

Number of unsign operations.

- **Definition:** How many times does a transaction peer have to check the content of an evidence and, thus, execute an unsign operation?
- **Quantification:** In mutual transaction protocols, no evidences are issued nor checked. In global transaction protocols, each peer has to check exactly two evidences, i.e., a contract and a receipt.

[2]This category corresponds to the complexity category of Section 3.2. The number of rounds is not considered explicitly since it is proportional to the number of messages that an entity has to send. This means that, for the assessment of the quantification of this category, one does not only have to take the costs of sending a message into account. In addition, the delay in getting control of the desired action has also to be considered.

Initiator Roles / Characteristics	Bilateral Transaction Protocols				
	Direct Exchange			Indirect Exchange	
	Global Knowledge		Mutual Knowledge	Global Knowledge	Mutual Knowledge
	Symmetric Steps	Asymmetric Steps	Symmetric Steps	Symmetric Steps	Symmetric Steps
Role Name	BDGS/P $_I$	BDGA/P $_I$	BDMS/P $_I$	BIGS/P	BIMS/P
Sufficiency of Execution	0	1	0	0	0
Sufficiency of Lost Control	0	1	0	t_{TP}	t_{TP}
Deterrence	0.5	0.5	0	0.5+0.5* t_{TP}	0.3*t_{TP}
Messages Sent	3	2	1	3	1
Sign Operations	2	2	0	2	0
Unsign Operations	2	2	0	2	0

Responder Roles / Characteristics	Bilateral Transaction Protocols				
	Direct Exchange			Indirect Exchange	
	Global Knowledge		Mutual Knowledge	Global Knowledge	Mutual Knowledge
	Symmetric Steps	Asymmetric Steps	Symmetric Steps	Symmetric Steps	Symmetric Steps
Role Name	BDGS/P $_R$	BDGA/P $_R$	BDMS/P $_R$	BIGS/P	BIMS/P
Sufficiency of Execution	1	0	1	0	0
Sufficiency of Lost Control	1	0	1	t_{TP}	t_{TP}
Deterrence	0.7	0.7	0	0.5+0.5* t_{TP}	0.3*t_{TP}
Messages Sent	3	2	1	3	1
Sign Operations	2	2	0	2	0
Unsign Operations	2	2	0	2	0

Table 6.1: Quantification of the benefits and costs of the proposed transaction protocols for the initiator and responder role

6.1.3 Quantification for the Respective Transaction Protocols

The quantification of the benefit and cost categories is directly related to the protocols' key properties that are summarized in Section 5.3. Therefore, the categories' quantification for the respective protocols is straightforward. It is shown in Figure 6.1.3 for the initiator and responder roles.

6.2 Tradeoff-aware Choice of Transaction Protocols

The categoric quantification of the protocols' benefits and costs provides a guideline of which transaction protocol suits best to the circumstances of a transaction. The decision making component of an entity (see Figure 4.3) only has to decide how it valuates the quantification of the benefits and costs. For example, a PDA that lack resources could want to minimize costs by choosing mutual protocols. On the other hand, a notebook that is about to perform a crucial transaction could aim at maximizing security by laying stress on the sufficiency and deterrence

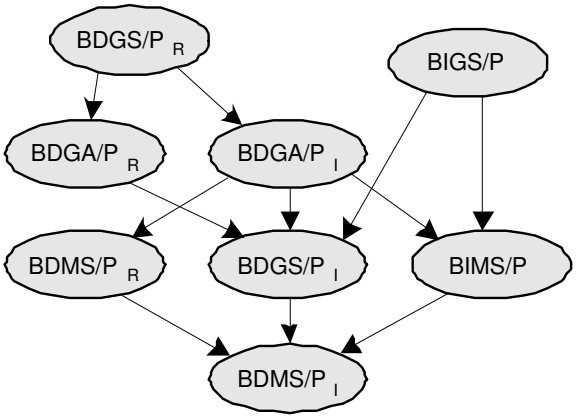


Figure 6.1: The dominance graph with respect to the benefits of the transaction roles

categories.

Apparently, the choice depends on the entity's preferences. Still, some protocol roles dominate other protocol roles with respect to several or all categories. This means that some or all quantities of the dominating protocol role are superior[3].

In the following, we illustrate such dominance by providing several dominance graphs. They give a rough estimation of the tradeoff-driven choice of transaction protocols.

Dominance regarding benefits. Figure 6.1 shows the dominance graphs with respect to the benefit categories.

For the integration of indirect protocols, the following rule is applied: **(1)** The role of an indirect protocol dominates another role if, regardless of the trustworthiness of the third party, its quantities are superior. **(2)** The role of an indirect protocol is dominated by another role if, regardless of the trustworthiness of the third party, its quantities are inferior.

The dominance graph may be interpreted as follows: Dominance induces the preference order of an entity that is not concerned about the costs of a transaction protocol.

Dominance regarding costs. Figure 6.2 illustrates the dominance graph with respect to the cost categories. Several roles share the same quantities. Hence, the figure shows that there are three equivalence classes with respect to costs.

The dominance graph may be interpreted as follows: Dominance induces the preference order of an entity that is not concerned about the benefits of a transaction protocol. This may be an entity that is fully convinced of the trustworthiness of its transaction partner.

Overall dominance. Figure 6.3 shows the overall dominance graph that results from the superposition of the benefit and cost dominance graphs respectively. Hence, the graph shows the preference order that every entity has regardless of its cost/benefit preferences.

We observe that, even though some roles are dominated, there is no transaction protocol that has all of its roles dominated. Hence, no transaction protocol is superfluous and could be chosen by the transaction peers.

[3]For benefit categories, a quantity is superior if it is higher. For cost categories, a quantity is superior if it is lower.

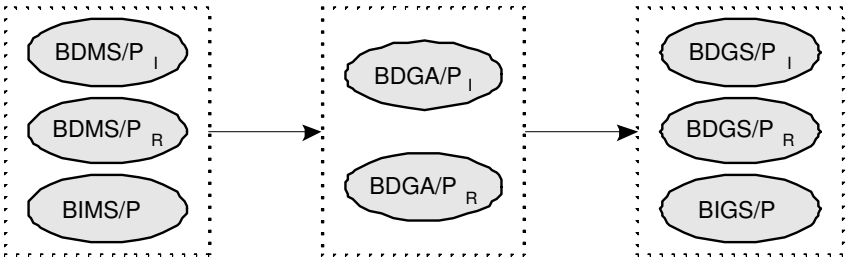


Figure 6.2: The dominance graph with respect to the costs of the transaction roles

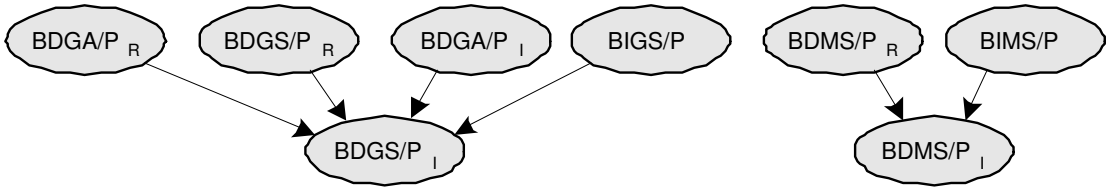


Figure 6.3: The dominance graph with respect to the residual utility of the transaction roles

Chapter 7

Conclusion

7.1 Summary

Self-organizing systems of autonomous entities have gained wide-spread attention in the research community. The most difficult problem of such systems is that autonomous entities may choose between cooperation and defection in the transactions they participate.

In internet based eCommerce, transaction protocols are applied for this purpose. In this report, we have identified the characteristics of internet based transaction protocols and have described their most common instances. However, we have shown that the setting of the internet differs from the one of self-organizing systems. Therefore, these transaction protocols can only be applied to a certain degree to self-organizing systems. The absence of infrastructure implies that defective transactional behavior has to be punished in a self-organizing manner. Only if such a self-organizing punishment exists, the identification of misbehavior (by the means of transaction protocols) makes sense.

We have discussed both conventional and evidence-aware distributed reputation systems since they allow for self-organizing punishment. This led us to stress the role of transaction protocols in a self-organizing system. The protocols have to preempt defections by coupling the transactional steps of the peers. In addition, transaction protocols have furnish evidences of the peers' behavior in order to make behavior perceptible.

Based on this observation, we have conceived transaction protocols that comply with these demands. For this purpose, we have identified the characteristics of the transaction protocols. Based on that, we have proposed five transaction protocols. The key properties of the protocols have been illustrated and discussed by a schematic visualization technique. The proposed protocols have been compared to the internet based protocols.

To this point, the problem that remained to be resolved was the following: Given the circumstances of a transaction, which transaction protocol and which role allocation is the most appropriate one? We have solved this problem in two steps. First, we have categorized the circumstances of transactions. The categories either indicate the benefits or the costs of transaction protocols. Second, the protocols' key properties have been captured by quantifying to which degree the respective protocol is associated to each category. As a result, it has become possible to assign benefits, costs, and a residual utility to a transaction protocol for a specific set of circumstances. Consequently, the transaction peers may choose the transaction protocol that has the highest residual utility. We have illustrated the principles of tradeoff-aware choice by providing dominance graphs for the proposed transaction protocols and roles.

7.2 Future Work

Evaluation. This report proposes transaction protocols for self-organizing systems and makes their tradeoffs explicit. Still, the ultimate aim of this work is to curb defective behavior in the course of transactions. This aim can only be attained if an evidence-aware distributed reputation system is available. Therefore, we plan to develop such a system so that we can measure the effects of the transaction protocols on the entities' behavior. In this regard, we will evaluate which protocols are chosen under which circumstances. Furthermore, we will measure the impact of the availability or non-availability of certain transaction protocols.

Affidavits. In addition, the set of proposed transaction protocols could be expanded by developing indirect protocols that make use of affidavits. This would entail the availability of a further type of evidence which, in turn, influences the deliberation processes of the reputation system.

Deposits and weakly secret bit commitment. The inclusion of Buttyan's and Syverson's protocols is a further promising idea that could be followed in the future. For Buttyan's protocol, we would have to evaluate whether it makes sense to apply deposit notes that benefit an arbitrary third party. For Syverson's protocol, the difficulty of computing the actual return value has to be adjusted to the transaction peers' utility structure.

Multilateral transaction protocols. Another line of future work is to examine what happens if we relax some of the assumptions of our system model. For example, it could be researched which kinds of multilateral transaction protocols fit to self-organizing systems.

Revisiting the deterrence category. In this report, the weights of the benefit category of deterrence are set rather arbitrarily. In the future, it might make sense to revisit the deterrence category and propose a more thorough weighting.

Acknowledgement

The work done for this report is partially sponsored by the German Research Community (DFG) in the context of the priority programs (SPP) no. 1140.

Bibliography

[1] Institute for Program Structures and Data Organization, Universität Karlsruhe: DIANE Project. http://www.ipd.uni-karlsruhe.de/DIANE/en (2003)

[2] Obreiter, P., Nimis, J.: A taxonomy of incentive patterns - the design space of incentives for cooperation. In: Second Intl. Workshop on Agents and Peer-to-Peer Computing (AP2PC'03), Springer LNCS 2872, Melbourne, Australia (2003)

[3] Obreiter, P.: A case for evidence-aware distributed reputation systems. In: Second International Conference on Trust Management (iTrust'04), Oxford, UK, Springer LNCS 2995 (2004) 33–47

[4] Asokan, N.: Fairness in Electronic Commerce. PhD thesis, University of Waterloo (1998)

[5] Obreiter, P., König-Ries, B., Papadopoulos, G.: Engineering incentive schemes for ad hoc networks - a case study for the lanes overlay. In: First EDBT-Workshop on Pervasive Information Management, To appear in post-proceedings, Greece (2004)

[6] Obreiter, P., König-Ries, B., Klein, M.: Stimulating cooperative behavior of autonomous devices - an analysis of requirements and existing approaches. In: Proceedings of the Second International Workshop on Wireless Information Systems (WIS2003), Angers, France (2003) 71–82

[7] Buttyan, L.: Building Blocks for Secure Services: Authenticated Key Transport and Rational Exchange Protocols. PhD thesis, EPFL (2001)

[8] Pfitzmann, B., Schunter, M., Waidner, M.: Optimal efficiency of optimistic contract signing. In: Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing (PODC), Puerto Vallarta, Mexico (1998) 113–122

[9] Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for multi-party fair exchange. Technical Report RZ 2892 (# 90840) (1996)

[10] Syverson, P.: Weakly secret bit commitment: Applications to lotteries and fair exchange. In: Proceedings of the IEEE Computer Security Foundations Workshop. (1998) 1–13

[11] Wu, J., Axelrod, R.: How to cope with noise in the iterated prisoner's dilemma. Journal of Conflict Resolution **39** (1995) 183–189

[12] Kinateder, M., Rothermel, K.: Architecture and algorithms for a distributed reputation system. In Nixon, P., Terzis, S., eds.: Proc. Of the First Intl. Conf. On Trust Management (iTrust), Heraklion, Greece, Springer LNCS 2692 (2003) 1–16

[13] Yu, B., Singh, M.P.: An evidential model of distributed reputation management. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02), Bologna, Italy (2002) 294–301

[14] Obreiter, P., Fähnrich, S., Nimis, J.: How social structure improves distributed reputation systems - three hypotheses. In: Third Intl. Workshop on Agents and Peer-to-Peer Computing (AP2PC'04), To appear in post-proceedings, New York (2004)

[15] English, C., Wagealla, W., Nixon, P., Terzis, S., Lowe, H., McGettrick, A.: Trusting collaboration in global computing systems. In: Proc. of the First Intl. Conf. on Trust Management (iTrust), Heraklion, Crete, Greece (2003) 136–149

Appendix A

Implementation and Use of Transaction Protocols

A.1 The Pre-transaction Phase: Agreement on the Terms of the Transaction

Up to now, the implementation assumes that the entity that has a transaction opportunity (proposer) coincides with the initiator. Furthermore, the agreement on the terms is implemented as a proposal-acceptance/rejection process. In Figure A.1, the agreement phase is visualized in a sequence diagram. Note that the initiator and the responder assume the same role for third party protocols.

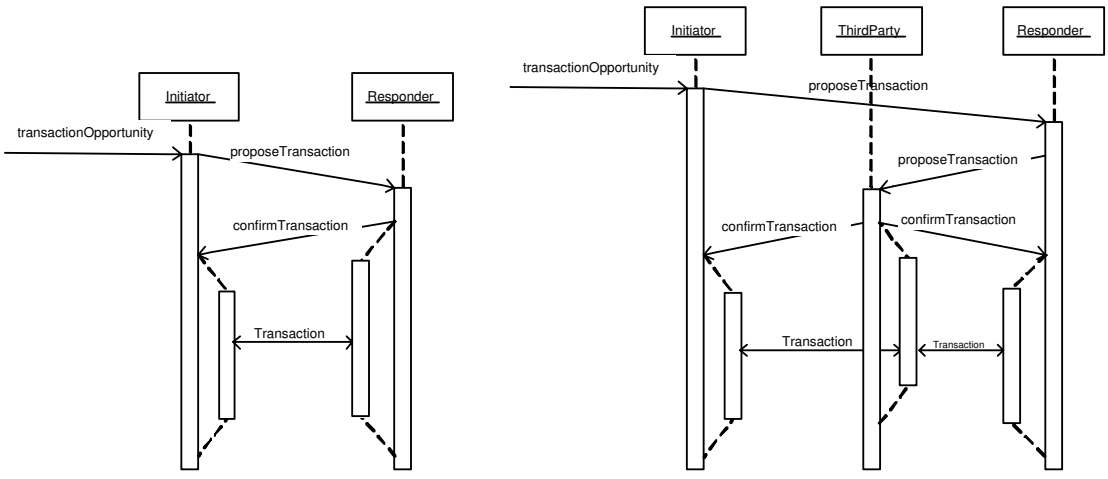


Figure A.1: (Left) without third party; (right) with third party

A.2 Algorithms of the Proposed Transaction Protocols

We only present the algorithm of the BDGS protocol since it contains every complexity that is found for the other transaction protocols.

A.2.1 BDGS (6-way)

Procedure $performTransaction - BDGS/P_i$
$(peerID, context, myAction, peerActionDescription)$

1: $myTID \leftarrow newTransactionID$
2: $handover(issueEvidence(contract(peerID, myTID, context)))$
3: $receivedContract \leftarrow receive(peerID)$
4: **if** $(timeoutWithoutContract \parallel receivedContract \neq (peerID, myID, ?, context, ?)$ **then**
5: $reportTransaction(peerID, context, DEFECTED)$
6: QUIT
7: **end if**
8: $addEvidence(receivedContract)$
9: **if** $(ValuatorManager(\text{"}DoAction\text{"}, (context, peerID, RISK\_POSITION)) \geq 0)$ **then**
10: $handover(myAction)$
11: **end if**
12: $receivedAction \leftarrow receive(peerID)$
13: **if** $(timeoutWithoutAction \parallel \neg\, receivedAction.match(peerActionDescription))$ **then**
14: $addKnowledge(noAction(peerID, receivedContract.peerTID))$
15: $reportTransaction(peerID, context, DEFECTED)$
16: QUIT
17: **end if**
18: **if** $(ValuatorManager(\text{"}DoReceipt\text{"}, (context, peerID, RISK\_POSITION)) \geq 0)$ **then**
19: $handover(issueEvidence(receipt(peerID, receivedContract.peerTID)))$
20: **end if**
21: $receivedReceipt \leftarrow receive(peerID)$
22: **if** $(timeoutWithoutReceipt \parallel receivedReceipt \neq (peerID, myID, myTID))$ **then**
23: $reportTransaction(peerID, context, DEFECTED)$
24: **else**
25: $addEvidence(receivedReceipt)$
26: $reportTransaction(peerID, context, GOOD)$
27: **end if**

Procedure $performTransaction - BDGS/P_r$
$(peerID, context, myAction, peerActionDescription)$

1: $receivedContract \leftarrow receive(peerID)$
2: **if** $(timeoutWithoutContract \parallel receivedContract \neq (peerID, myID, ?, context, ?))$ **then**
3: QUIT
4: **end if**
5: $addEvidence(receivedContract)$
6: $myTID \leftarrow newTransactionID$
7: **if** $(ValuatorManager("DoContract", (context, peerID, SAFE\_POSITION)) \geq 0)$ **then**
8: $handover(issueEvidence(contract(peerID, myTID, context)))$
9: **end if**
10: $receivedAction \leftarrow receive(peerID)$
11: **if** $(timeoutWithoutAction \parallel \neg\, receivedAction.match(peerActionDescription))$ **then**
12: $addKnowledge(noAction(peerID, receivedContract.peerTID))$
13: $reportTransaction(peerID, context, DEFECTED)$
14: QUIT
15: **end if**
16: **if** $(ValuatorManager("DoAction", (context, peerID, SAFE\_POSITION)) \geq 0)$ **then**
17: $handover(myAction)$
18: **end if**
19: $receivedReceipt \leftarrow receive(peerID)$
20: **if** $(timeoutWithoutReceipt \parallel receivedReceipt \neq (peerID, myID, myTID))$ **then**
21: $addKnowledge(noAction(peerID, receivedContract.peerTID))$
22: $reportTransaction(peerID, context, DEFECTED)$
23: **else**
24: $addEvidence(receivedReceipt)$
25: **if** $(ValuatorManager("DoReceipt", (context, peerID, SAFE\_POSITION)) \geq 0)$ **then**
26: $handover(issueEvidence(receipt(peerID, receivedContract.peerTID)))$
27: $reportTransaction(peerID, context, GOOD)$
28: **else**
29: $addKnowledge(noAction(peerID, receivedContract.peerTID))$
30: $reportTransaction(peerID, context, DEFECTED)$
31: **end if**
32: **end if**

A.3 Diagrams of the Proposed Transaction Protocols

The notations used for various functions and objects in the protocol diagrams are the following:

Entities.

- **I**: Initiator
- **R** : Responder
- **T** : Third Party

Items.

- **ContractI/R** The contract of the Initiator, Responder respectively.
- **ActionI/R** The action of the Initiator,Responder respectively.
- **ReceiptI/R** The receipt of the Initiator, Responder respectively.

Valuators.

- **DoContract/DoAction/DoReceipt(name of the peer to receive)** : Functions evaluating whether the contract or action or receipt of the peer calling the respective function are to be sent to the partner peer. For example, *DoContract(R)=true or DoAction(R)=true or DoReceipt(R)=true* by the Initiator leads to the handover of Initiator's contract or action or receipt to the Responder. On the other hand, a false evaluation of one of these functions (*DoContract(R)=false or DoAction(R)=false or DoReceipt(R)=false*) causes a defection on Initiator's side, i.e. the Initiator fails to send his contract or action or receipt to the Responder.
- **DoForward(name of the peer to receive)** : Function called exclusively by the entity assuming the role of third party. If this function evaluates to true (*DoForward(R)=true*), then the third party forwards the respective evidence or action to the receiving peer(in this example, the Responder). The defection on the third party side is caused by a false evaluation (*DoForward(R)=false*). In this case the third party fails in forwarding the evidence or action (in our example, fails to forward to the Responder).

Verificators.

- **TestContract/TestAction/TestReceipt(name of the peer having sent)** : Functions called in the purpose of verifying the received contract or action or receipt. For example, *TestContract(I)* is called by the Responder for the verification of the Initiator's contract. Respectively, *TestContract(R)* is called by the Initiator for the verification of the Responder's contract. On the Initiator's side *TestAction(I)* compares the received Responder's action with the action description in the possession of the Initiator. Finally, *TestReceipt()* verifies the received receipt.

Reputation system.

- **ReportTransaction(name of the partner peer or of the third party, transaction general conclusion)** In case of a complete transaction, i.e. the partner peer or the peer itself correctly fulfilled each required protocol step, the general conclusion is positive(*true*). For example, if the Responder ends a successful transaction with the Initiator, the function becomes *ResportTransaction(I,true)*. If the partner peer committed defection or if the peer itself wants to throw the guilt of a defected transaction on another peer, the transaction is concluded as being *false*. In our example, *ReportTransaction(I, false)*.

- **AddNoActionKnowledge(name of the partner peer)** This function is used to add knowledge to the evidence repository about the assumed misbehavior of the partner peer. Knowledge is added only when the peer, let's say the Initiator has received the Responder's contract but did not receive the Responder's action. In this case, the function becomes *AddNoActionKnowledge(R)* on the Initiator's side.

A.3.1 BDGS (6-way)

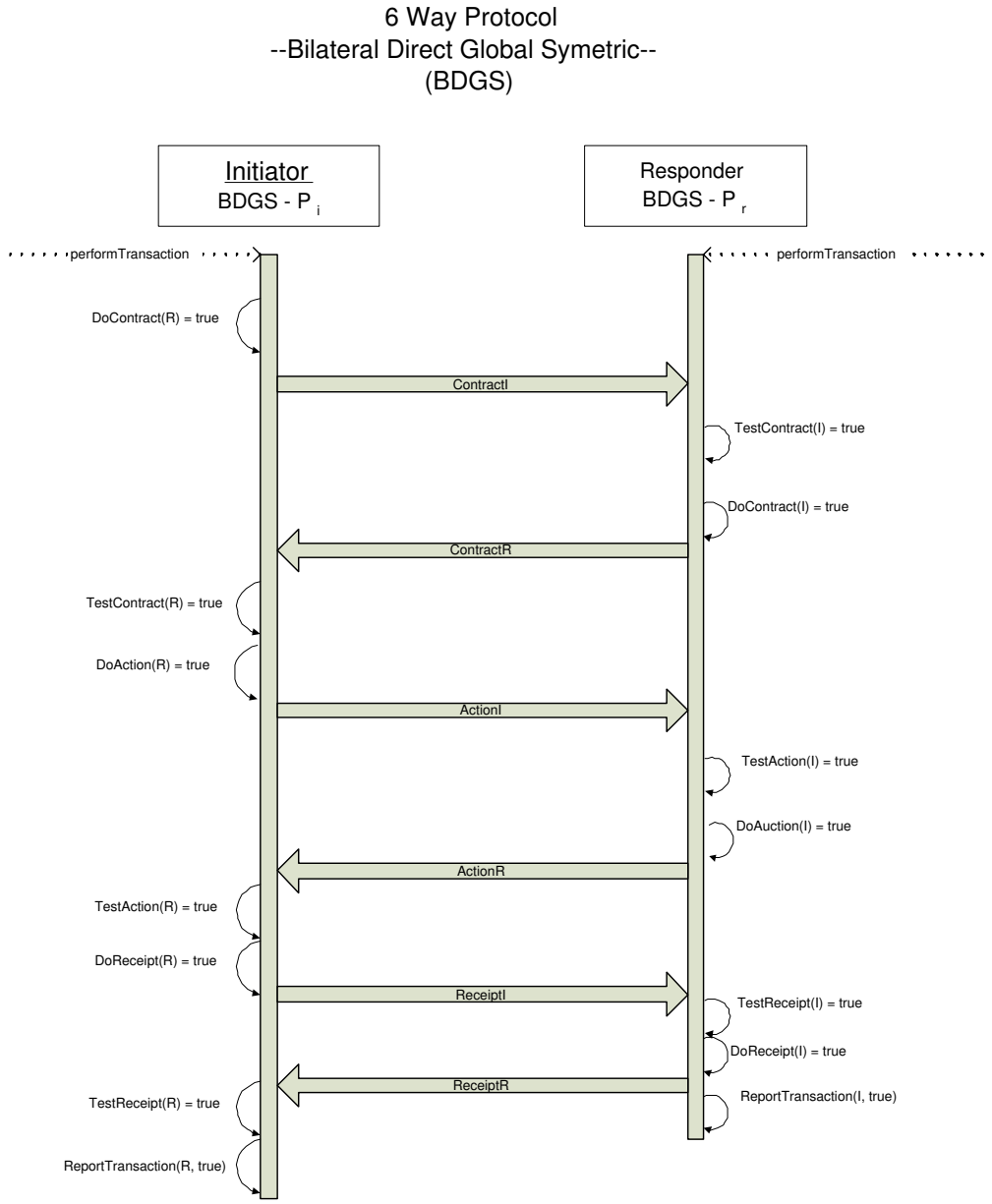


Figure A.2: Sequence diagram of the BDGS transaction protocol without defection

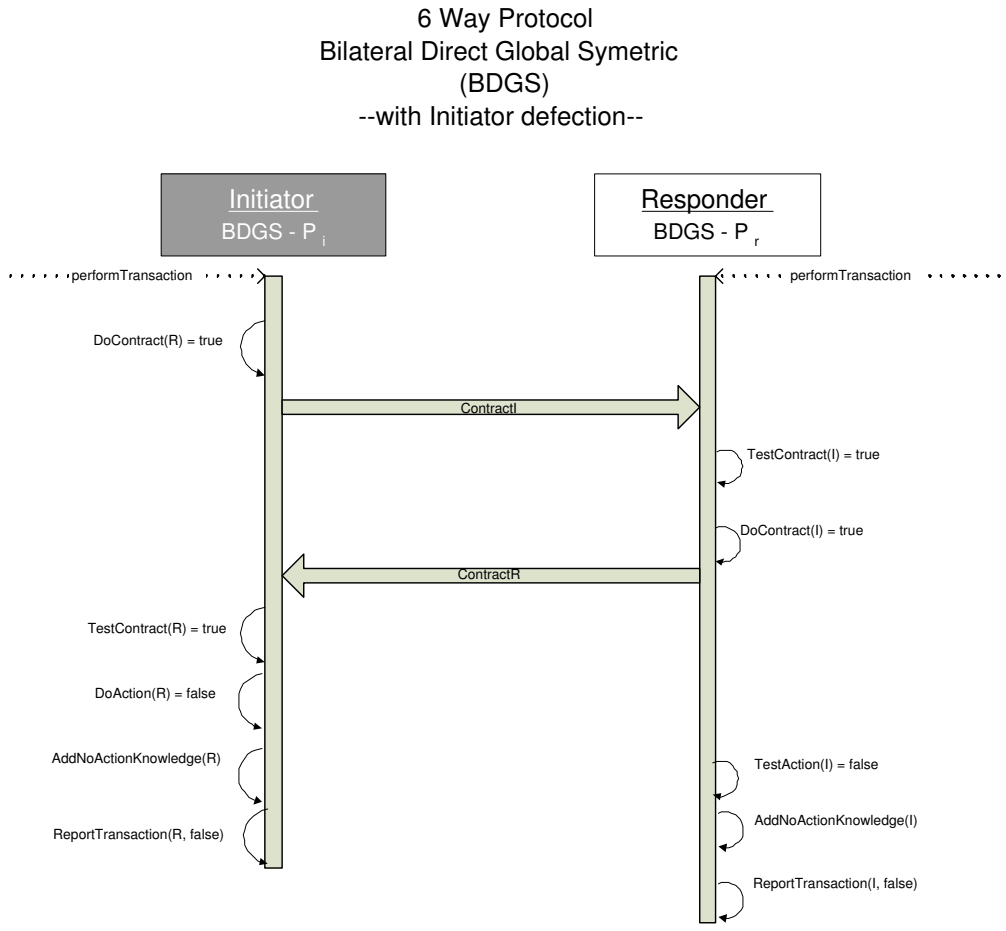


Figure A.3: Sequence diagram of the BDGS transaction protocol with defecting initiator

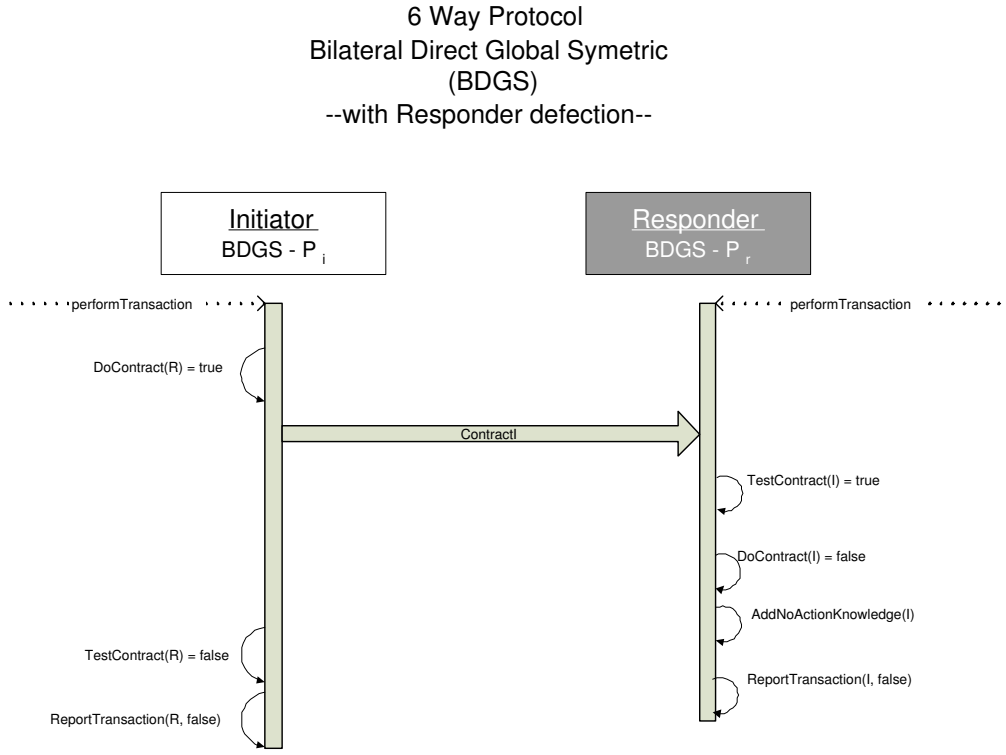


Figure A.4: Sequence diagram of the BDGS transaction protocol with defecting responder

A.3.2 BDGA (4-way)

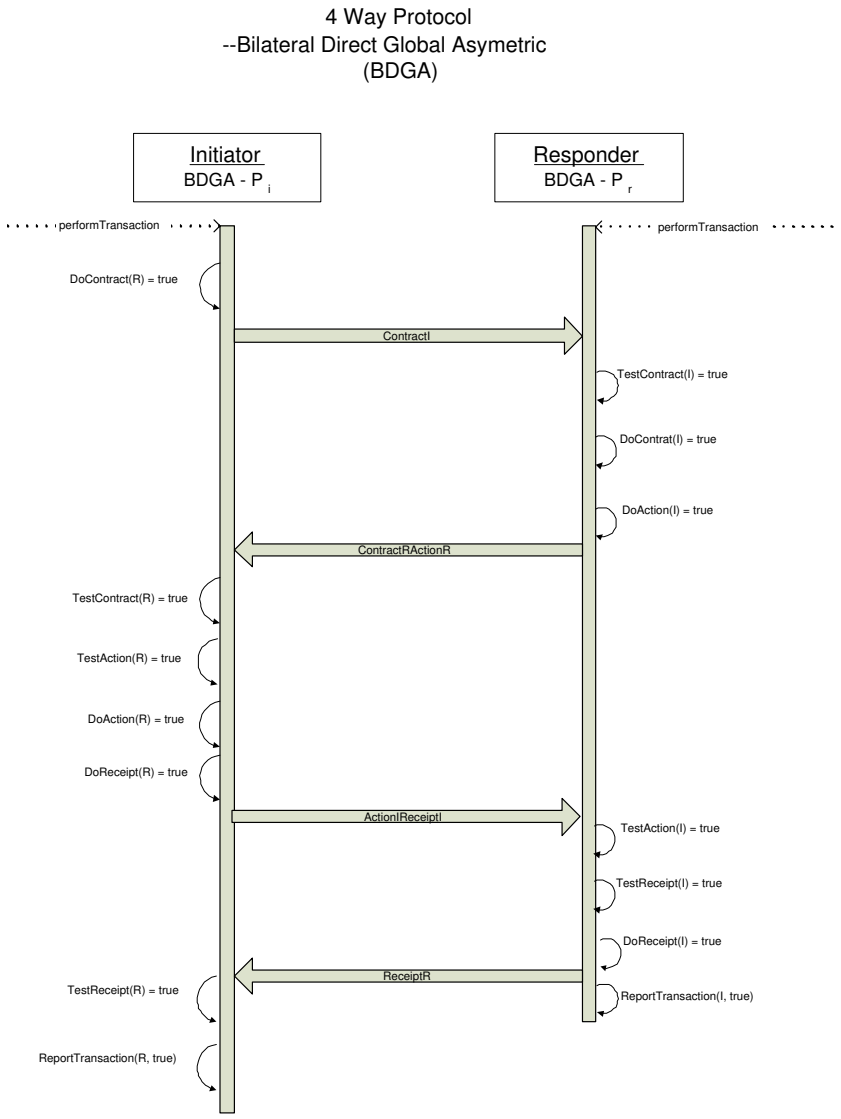


Figure A.5: Sequence diagram of the BDGA transaction protocol without defection

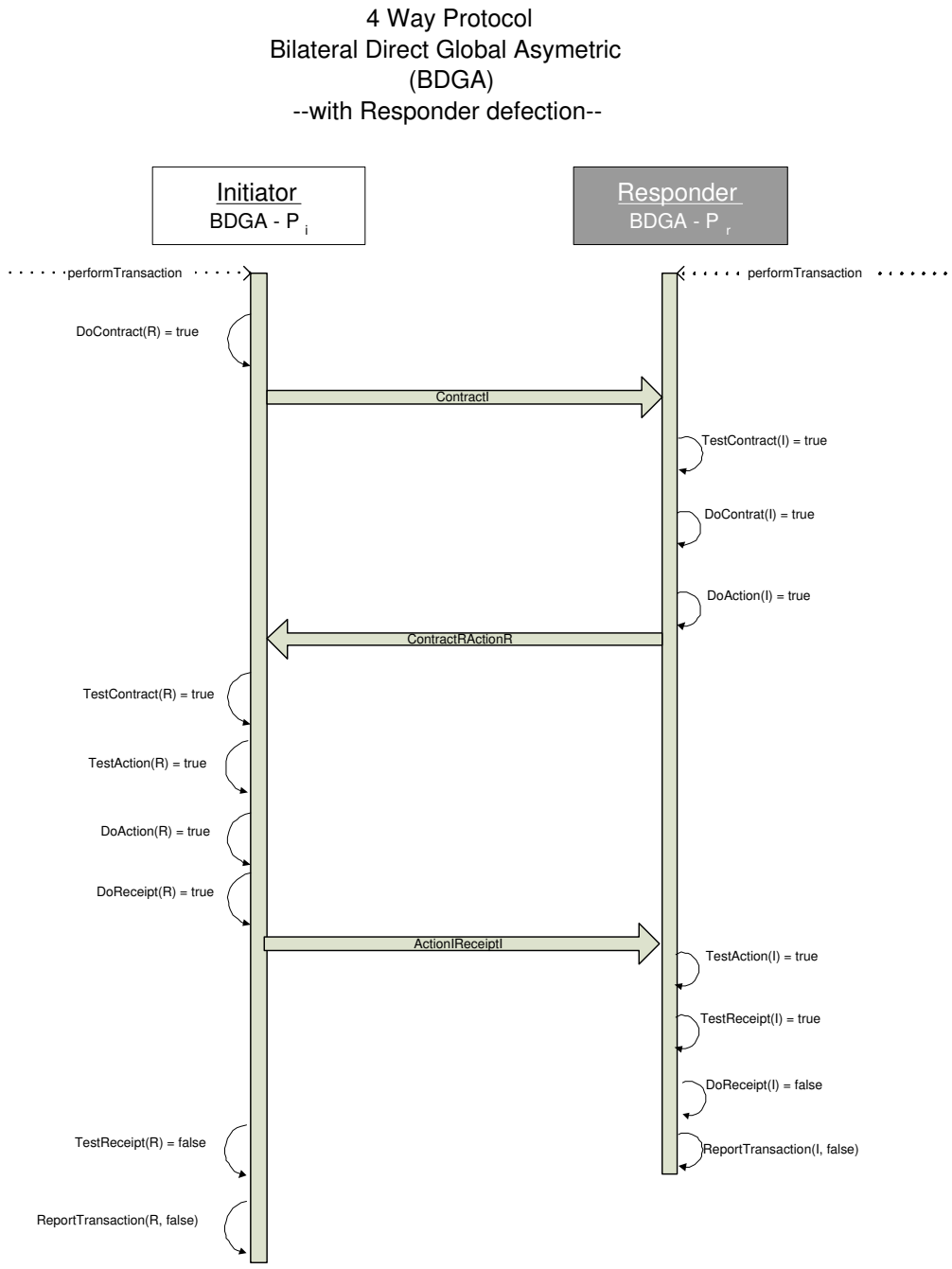


Figure A.6: Sequence diagram of the BDGA transaction protocol with defecting responder

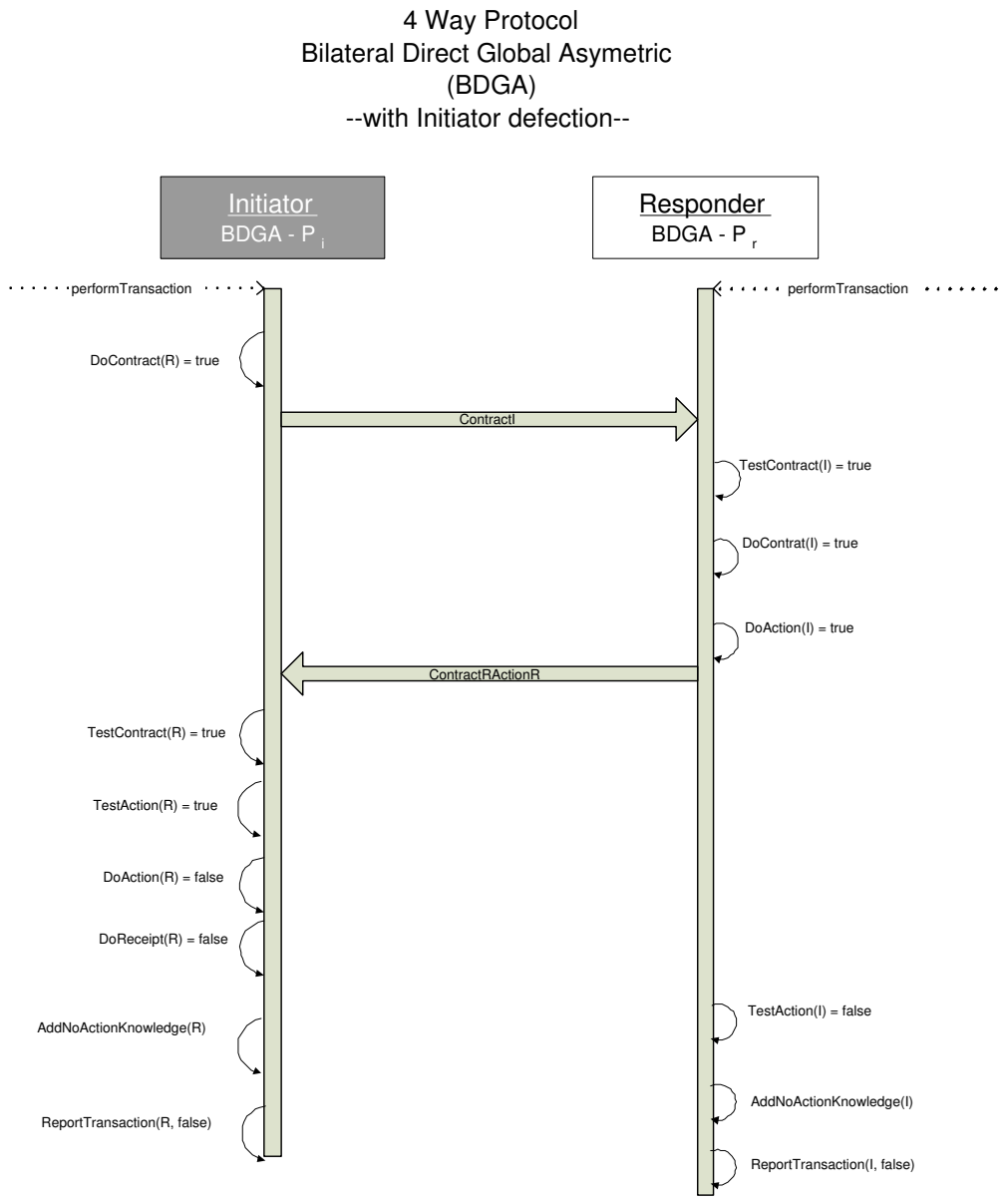


Figure A.7: Sequence diagram of the BDGA transaction protocol with defecting initiator

A.3.3 BDMS (2-way)

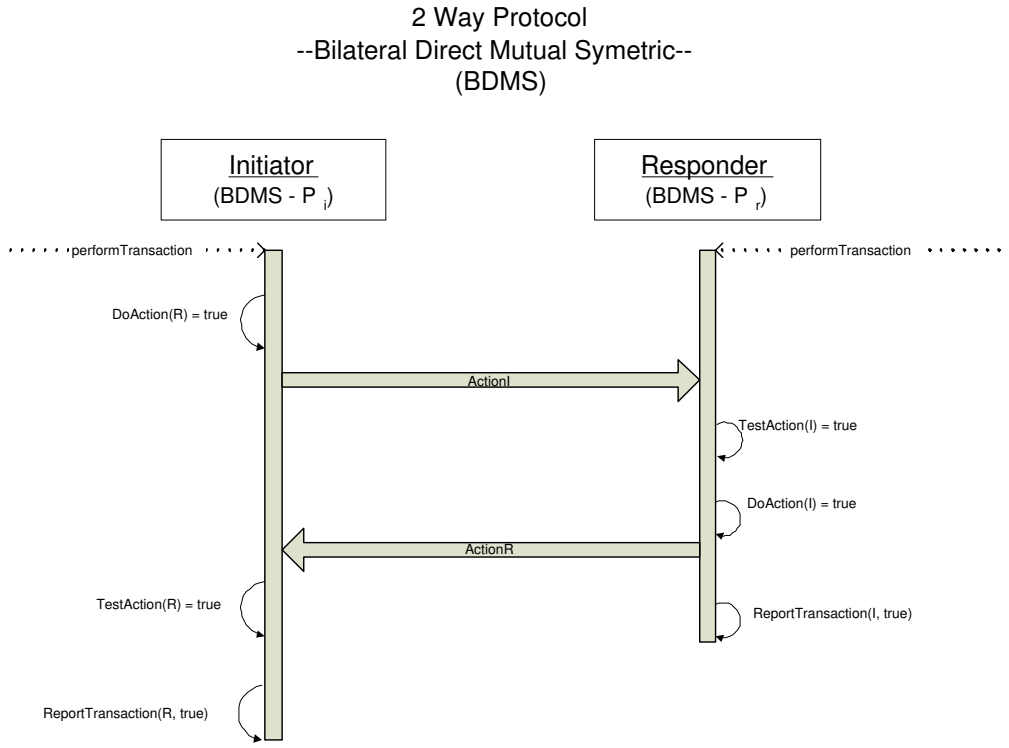


Figure A.8: Sequence diagram of the BDMS transaction protocol without defection

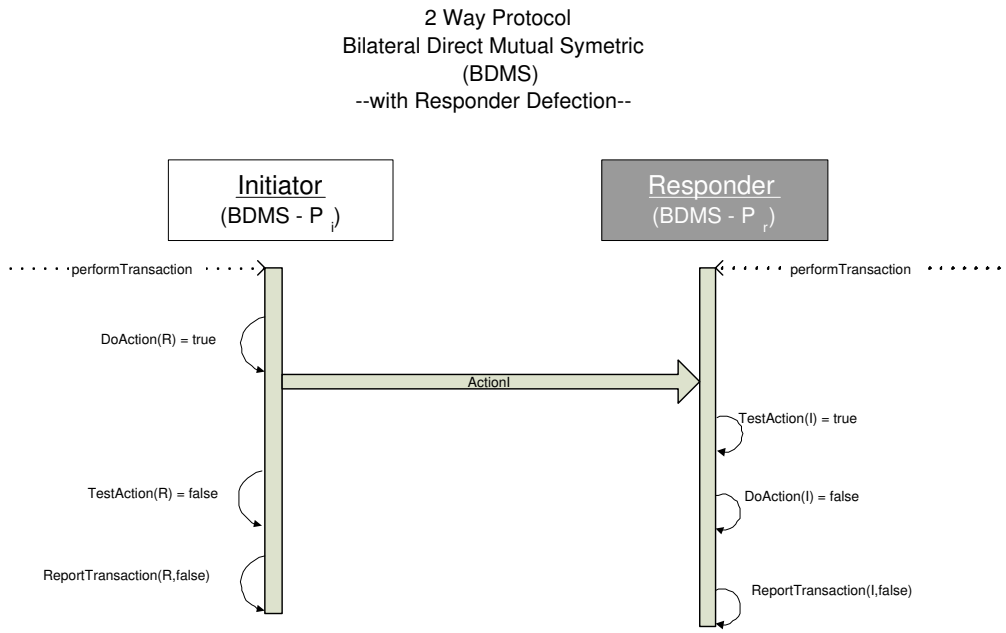


Figure A.9: Sequence diagram of the BDMS transaction protocol with defecting responder

A.3.4 BIGS (tp-6-way)

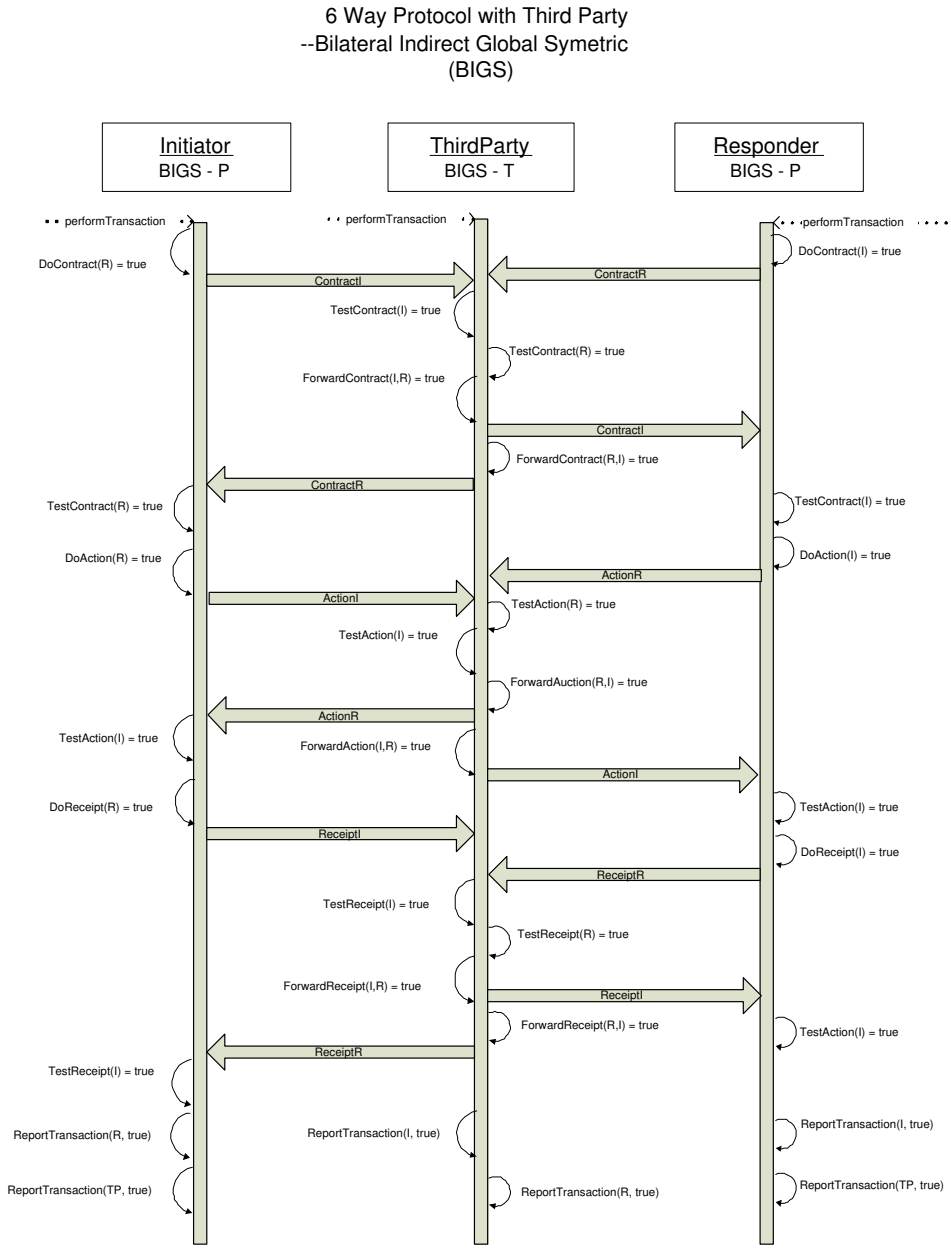


Figure A.10: Sequence diagram of the BIGS transaction protocol without defection

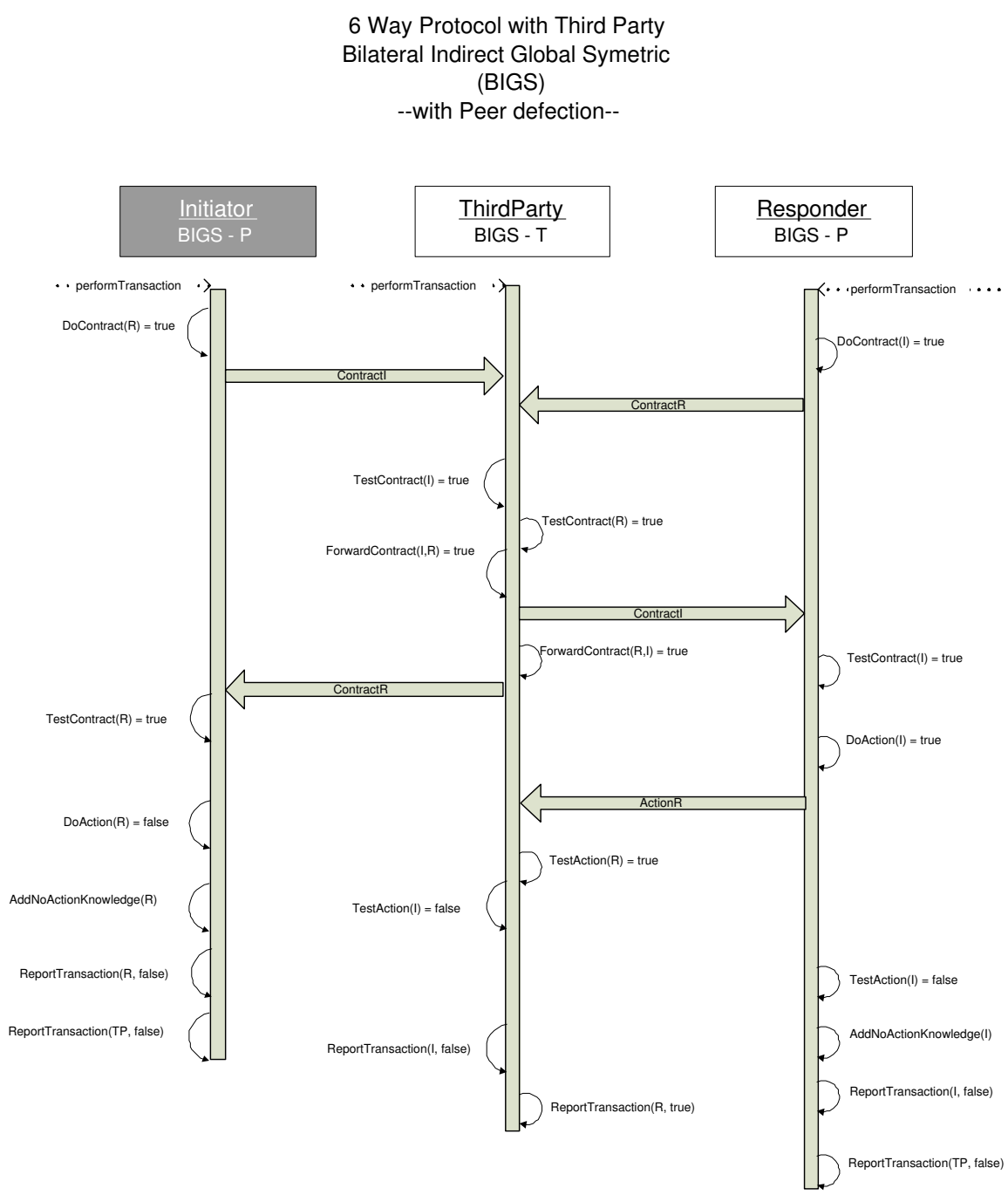


Figure A.11: Sequence diagram of the BIGS transaction protocol with defecting peer

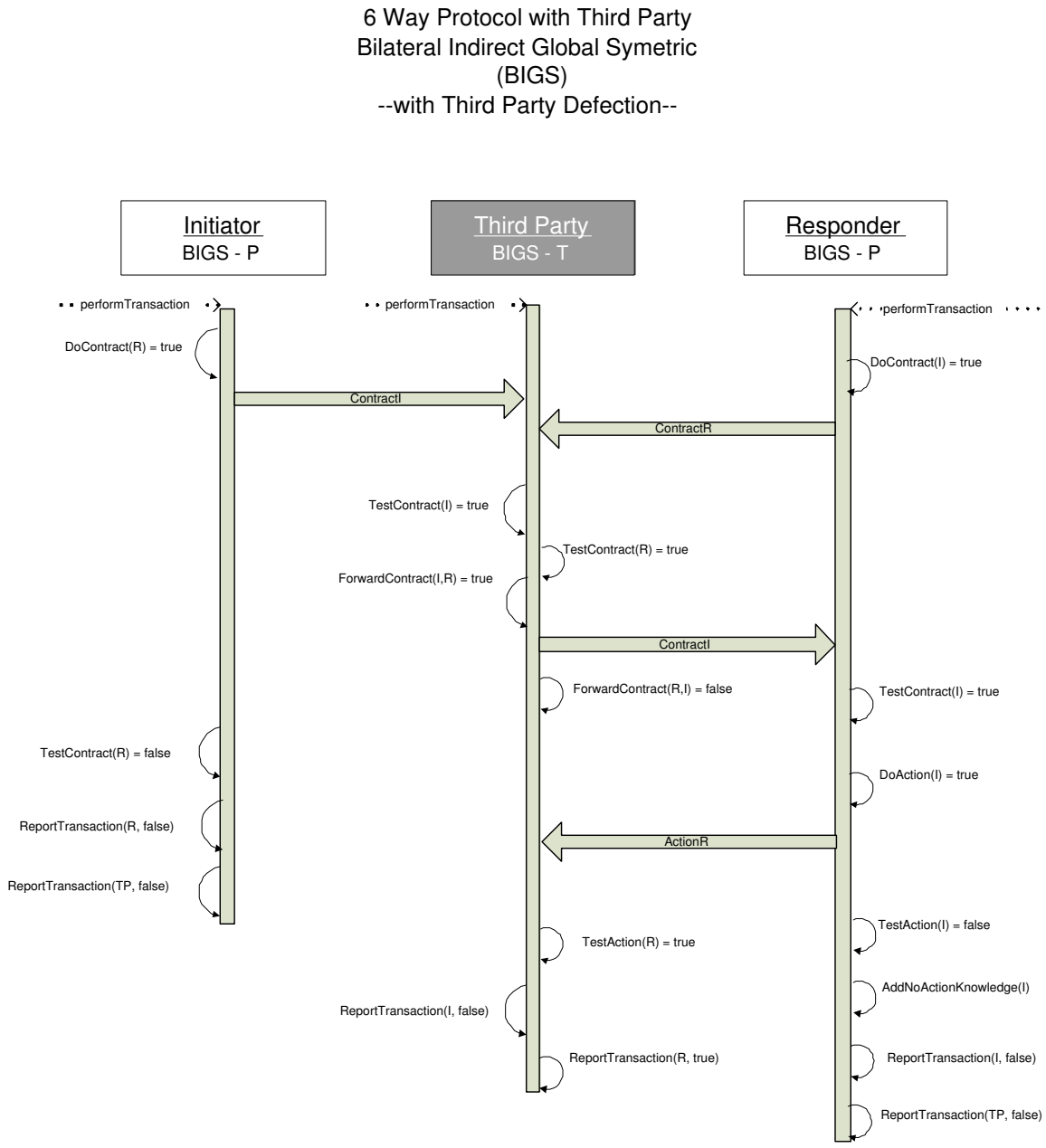


Figure A.12: Sequence diagram of the BIGS transaction protocol with defecting third party

A.3.5 BIMS (tp-2-way)

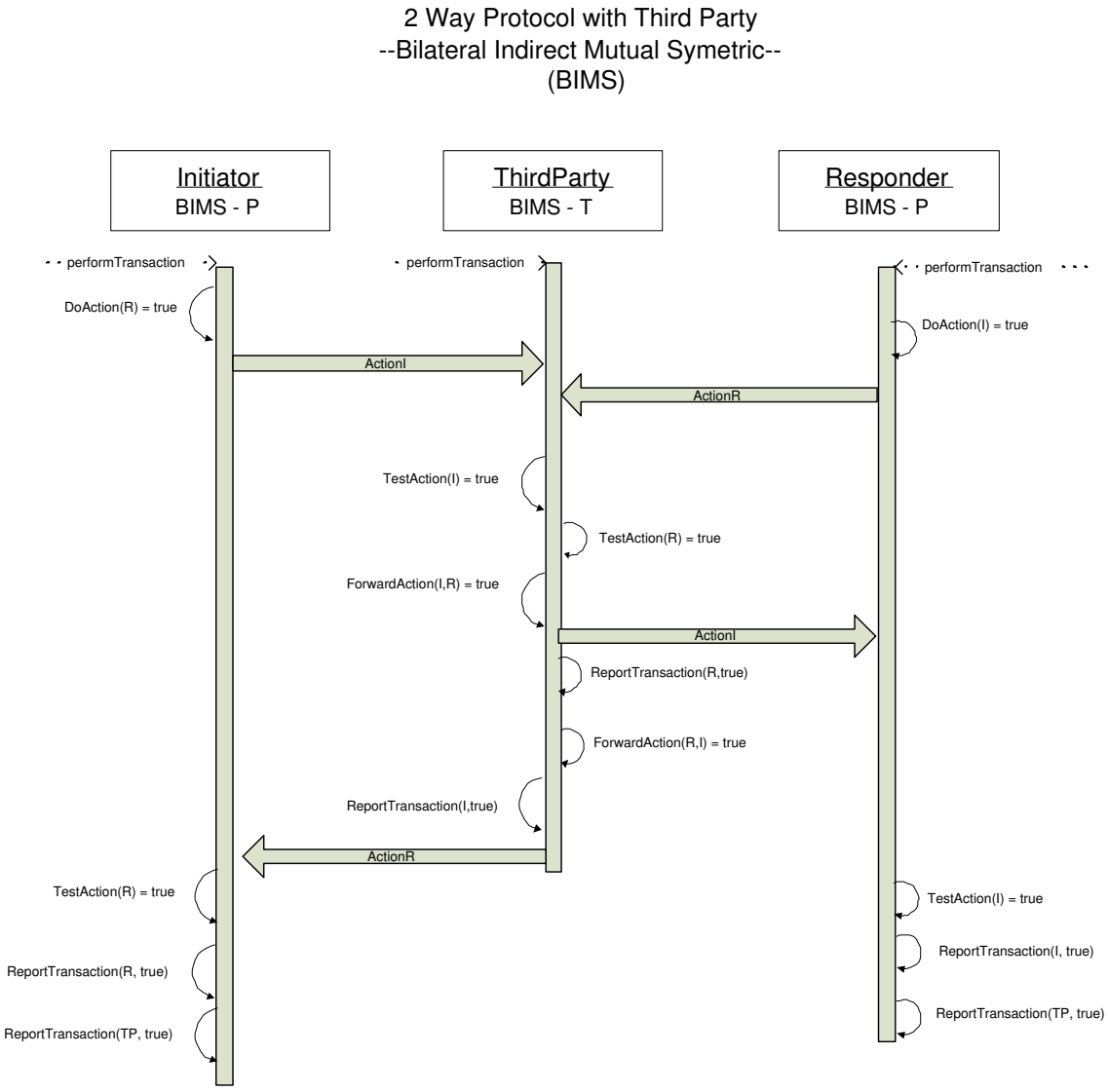


Figure A.13: Sequence diagram of the BIMS transaction protocol without defection

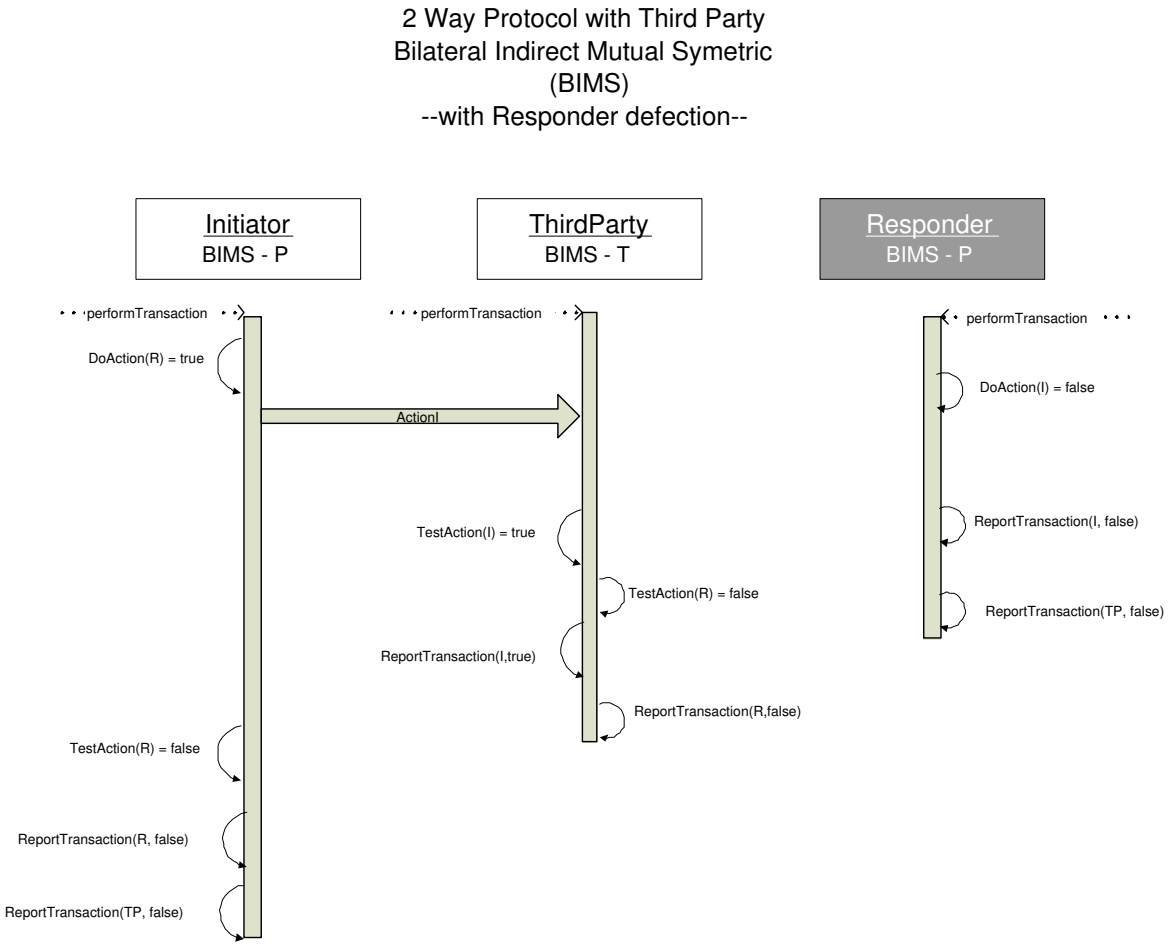


Figure A.14: Sequence diagram of the BIMS transaction protocol with defecting peer

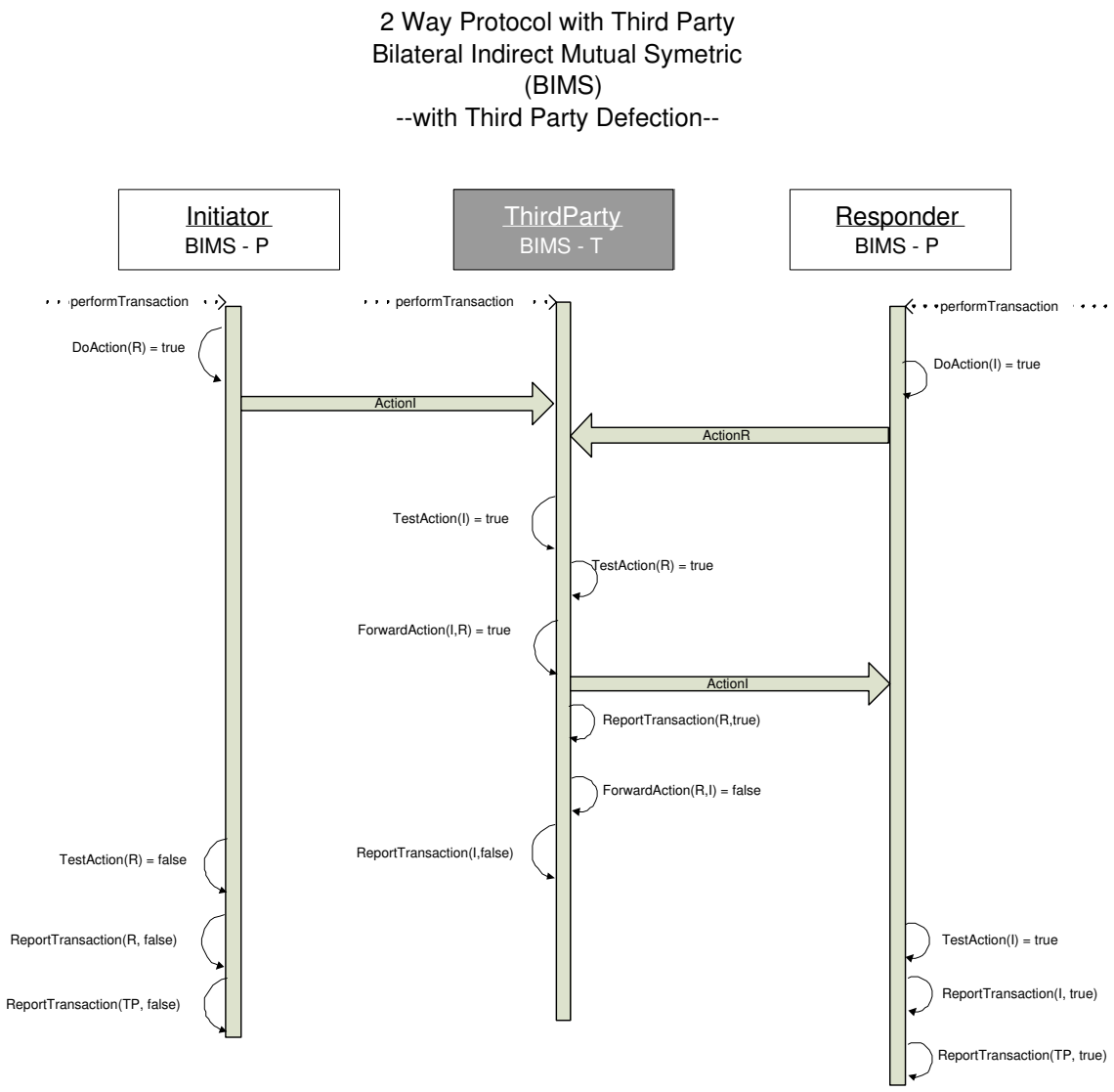


Figure A.15: Sequence diagram of the BIMS transaction protocol with defecting third party

Appendix B

Glossary

Action: *(Aktion)* A resource consuming activity which is beneficial for another entity.

Autonomy: *(Autonomie)* Each entity is autonomous. This means that an entity is free to *cooperate* or *defect* in the course of a transaction. Defection refers to the premature abandonment of a transaction.

Entity: *(Einheit)* A component of the system. It consumes resources in order to attain a goal. Typically, this involves transactions with other entities.

Evidence: *(Beweismittel)* An non-repudiable token about a specific aspect of another entity's behavior.

Exchange Protocols: *(Austauschprotokoll)* A protocol that defines the exchange of items. The procedure of item exchange is often described in subsequent protocol steps.

Execution of an item: *(Gegenstandsausführung)* Depending on whether the item is an action, a contract or a receipt, the execution of an item means the generation of a contract or receipt or the execution of an action.

Initiator (role): *(Initiator)* The entity that processes the first step of the transaction protocol.

Item: *(Gegenstand)* An abstract term for referring to a contract, an action or a receipt.

Lost control of an item: *(Verlorene Kontrolle eines Gegenstandes)* The incidence that the transaction partner acquires the item. For direct protocols, this coincides with the handover of the item. However, for indirect protocols, the control of an item is only lost if the third party forwards it to the transaction partner.

Responder (role): *(Antwortender)* The entity that does not process the first step of the transaction protocol.

Self-organization: *(Selbstorganisation)* The system's emergent organization if no infrastructural entity is available. In such a case, the entities have to assume infrastructural tasks. As a result, they organize themselves.

Third party (role): *(Dritter)* An entity that assists the transaction peers in performing the transaction.

Transaction: *(Transaktion)* In a transaction, a pair of entities (transaction peers) execute actions that are mutually beneficial for them.

Transaction peer: *(Transaktionsteilnehmer)* An entity that participates in a transaction in order to have an action executed by another entity.

Transaction protocol: *(Transaktionsprotokoll)* A protocol for performing transactions. It consists of the exchange of actions and, optionally, of the exchange of contracts and receipts. In this regard, a transaction protocol is more specific than the notion of exchange protocols.

Transaction role: *(Transaktionsrolle)* The role that an entity exhibits in a specific transaction protocol. For direct transaction protocols, this is either the initiator role or the responder role. For indirect transaction protocols, the initiator role and responder roles coincide. A further role for such indirect transaction protocols is the third party role.