Open Philosophies for Associative Autopoietic Digital Ecosystems

Contract n° 034824

Workpackage 4: Distributed Accountability, Identity, and Trust

Deliverable D4.2: Distributed Accountability Model for an Autopoietic Peer-to-Peer Network

Project funded by the European Community under the "Information Society Technology" Programme

Contract Number: 034824

Project Acronym: OPAALS

Title: Open Philosophies for Associative Autopoietic Digital Ecosystems

Deliverable N°: D4.2

Due dates: 5/2007

Delivery Date: 08/2007

Short Description:

This document reviews accountability models and protocols in the computing and software engineering domains through a literature review.

The document then proposes a model for accountability for peer-to-peer autopoietic networks.

We conclude with a summary of the work and an indication of future work required in this field

Author:WIT

Partners contributed: WIT

Made available to: Public

Versioning		
Version	Date	Author, Organisation
0.1	**05/2007**	**Paul Malone, WIT**
0.2	**06/2007**	**Paul Malone & Brendan Jennings, WIT**
0.3	**07/2007**	**Paul Malone, WIT**
0.4	**08/2007**	**Paul Malone, WIT**
0.5	**08/2007**	**Post internal reviewers Comments Addressed, Paul Malone, WIT**
1.0	**08/2007**	**Final Version for delivery**

Quality check:

1st Internal Reviewer : Oxana Lapteva, University of Kassel

2nd Internal Reviewer: Sotiris Moschoyiannis, University of Surrey

Table of Contents

Illustration Index

1 Introduction

Many in the business and research communities are pursuing the vision of digital ecosystems – distributed software environments through which organisations can seamlessly access customised, potentially disposable, services and knowledge. Peer-to-peer networks are a desirable choice for deployment of such digital ecosystems, due to the inherent decentralised, resource sharing and scalability nature of these networks. However, interactions between entities in such a peer-to-peer digital space require that there is trust established not only between the entities, but also in the platform upon which these interactions occur. In addition, where these environments are being used for economic transferral of value, it is essential that interactions can be accurately monitored and independently validated against Service Level Agreements (SLAs). Robust accountability mechanisms must be made available to accurately provide the means through which these interactions can be logged. Such accountability facilities can also enable trust levels to be established, monitored and evolve. In the case where interactions fail, accountability provides the means through which the cause of this failure can be clearly and unambiguously established.

The purpose of this document is examine accountability in the computer science and software engineering domains with a view to establishing a model of accountability which provides the required functionality for peer-to-peer digital ecosystem deployment.

This remainder of this document is structured as follows: Section 2 provides a literature review of accountability in the Computer Science domain, including non-repudiation protocols, service accountability and accountability research in peer-to-peer networks. Section 3 provides an initial model for accountability in peer-to-peer digital ecosystems drawing from the background research in Section 2. Section 4 then provides a conclusion and indicates where further work is required to refine this model beyond the current status.

2 Accountability

Accountability \Ac*count`a*bil"i*ty\, n.

The state of being accountable; liability to be called on to render an account; the obligation to bear the consequences for failure to perform as expected; accountableness.

[1913 Webster]

The above definition from the Merriem-Webster Dictionary gives a generic definition of accountability. Depending on the context in which the term accountability is used, there are various refinements of this definition appropriate to the context in question. Accountability has been the subject of philosophical and legal dissertations for centuries and is in general related to the concepts of blame, responsibility and liability.

In recent times the terms "accountability" and "governance" and their inter-relationship have become increasingly popular across a wide range of disciplines. As an indicator of the importance of this relationship, accountability is one of the six World Governance Indicators used by the World Bank in measuring the governance of nations[1]. Bullock [BULL-06] provides a model linking governance, accountability and legitimacy (trust in the system), which is a useful reference point in showing the relationship between these subjects and how changes in governance can ultimately reduce trust in the system.

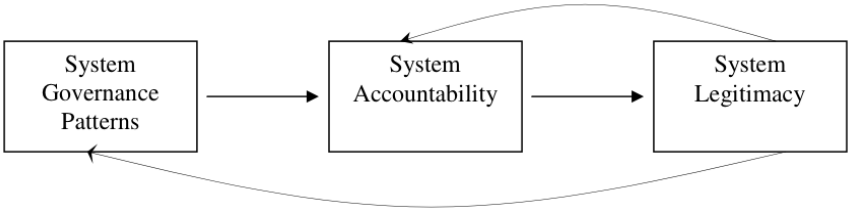

Figure 1: Bullock's Governance, Accountability and Legitimacy Model

In the model (Figure 1), changes in a system's governance pattern lead to a loss of accountability, which in turn depending on the degree of accountability loss can lead to a loss in the systems legitimacy (trust) in the system.

Wang and Singh [WANG-07] recognised this relationship between trust and accountability and view trust in federated systems as a reputation mechanism incorporating the knowledge of others in the system when deciding whether to trust in another party. They augment this view of trust in peer-to-peer systems with "belief" or expectation that the other peers output will provide the expected result. This belief approach is underpinned by available "evidence". Strong accountability provides trustworthy evidence and in turn provides accurate assessment of trustworthiness of peers and the system also.

[1] The Worldwide Governance Indicators (WGI) project, http://info.worldbank.org/governance/wgi2007/

Koppell [KOPP-05] cites The Public Administration Dictionary's definition of accountability as *"a condition in which individuals who exercise power are constrained by external means and by internal norms"* and makes the point that the external means (e.g. citizens, legislature, courts) may have conflicting interests. Furthermore he makes the point that "*Some analyses treat "accountability" as a synonym for law-abiding while others envision responsiveness to popular demands"*. Given the lack of conceptual clarity and the multiple meanings of the term accountability Koppel introduces the problem of Multiple Accountability Disorder (MAD) whereby an organisation suffering from MAD moves between behaviours influenced by conflicting concepts of accountability.

Koppell argues that providing a single definition for the term "accountability" renders the concept meaningless and provides five dimensions to accountability to enable clearer discussions:

1. *Transparency*: Did the organisation reveal the facts of its performance?
2. *Liability*: Did the organisation face consequences for its performance?
3. *Controllability*: Did the organisation do what the principal desired?
4. *Responsibility*: Did the organisation follow the rules?
5. *Responsiveness*: Did the organisation fulfil the substantive expectation?

For the purpose of this document the primary concern is that of accountability within distributed digital communities for the purposes of knowledge sharing, service provision and consumption, and collaboration. A useful definition of accountability in this context is given by Schedler [SCHE-99] as *"A is accountable to B when A is obliged to inform B about A's (past or future) actions and decisions, to justify them, and to suffer punishment in the case of eventual misconduct"*. Within our context *A* and *B* could represent individual players in the ecosystem, or *A* could represent a participant and *B* represent the ecosystem or community, or indeed both could represent ecosystems and their accountability to each other.

With this in mind, this chapter will provide a view of how accountability has been interpreted in the computing domain, drawing on a set of various previous works relevant to the ultimate purpose of this document; to model an accountability framework capable of providing the facilities to create trust in digital ecosystems, not only between the participants but in the ecosystems and the underlying platform itself.

2.1 Accountability in Computing Systems

Since the proliferation of computer systems in the 1970s in business and governmental support systems, an acknowledgement for the need of accountability in these systems has steadily grown. This requirement has been further accelerated by the availability and general ubiquity of the Personal Computer through the 1980s and 1990s through to today's widespread availability of broadband Internet access for small businesses and individual domestic users. An early example of this recognised need for accountability in computing systems is a 1994 article by Helen Nissenbaum [NISS-94]. Nissenbaum recognised that a community which insists on accountability where participants are answerable for their actions provides a signal for high quality work and

encourages responsible actions among the participants. The absence of such accountability provides a situation where no participant is answerable for risks or harm. She believed that accountability was being undermined in the emerging computerised society and that this undermining was due to three underlying factors rather than emerging computer system proliferation:

1. *A narrow understanding of the concept of accountability*
2. *A set of assumptions concerning the capabilities and shortcomings of computer systems*
3. *An acceptance that producers of computer systems were not fully answerable for the consequences of the usage of those systems*

Nissenbaum argued that life-critical systems required accountability not only to provide trust in those systems, but also to provide a means through which the end user had recourse in the event of malfunction. She identified four barriers to the adoption of rigorous accountability mechanisms being put in place for computerised systems:

1. *The problem of "many hands"*: Systems are most often created in an organisational setting.
2. *Bugs*: Bugs cause problems but are also commonly regarded as an inherent element of computer systems
3. *The computer as scapegoat*: The willingness to accept that the computer is at fault.
4. *Ownership without responsibility*: While systems providers were eager to assert rights of ownership, the responsibilities of ownership were being neglected.

Providing accountability, according to Nissenbaum, was not ultimately concerned with pinning blame on someone, rather ensuring an approach that diffused the absence of such accountability. Nissenbaum provides three recommendations to encourage accountability in the emerging computerised society:

1. *Keep accountability distinct from liability to compensate.*
2. *Clarify and promote a substantive standard-of-care.*
3. *Impose strict liability for defective consumer-oriented software and software whose impact on society and individuals is great.*

An implementation of a software design and development accountability model at Lockheed Martin's Modular Mission Computer Team is provided by Parris [PAR-96]. This model, as shown below in Figure 2, is provided as being prescriptive rather than descriptive, showing how things should be rather than how things actually are. This model views accountability as a proactive process rather than a concept of obligation. The participants in this process are the *Team* and the *Stakeholders*. The *Team* are those responsible for providing the desired result and the *Stakeholders* those effected by that result.

The process is an ongoing one of *accountings* and *consequences*. An accounting is a report on activity, past present or future. A consequence is a feedback on these reports. The process was put in place and iteratively adjusted as the process evolved. Barriers to the process came in the form of attitudes borne out of previous work practices. An awareness of the inter-dependencies between accountability and trust and accountability and learning is acknowledged as being fundamental in successful implementation of the provided model. In terms of

trust, the team must trust the stakeholders in believing that consequences are fair, that assistance will be helpful and that direction will be knowledgeable and consistent. Equally the stakeholders must trust that the accountings are accurate and honest, that no important information is suppressed and that the competing interests between stakeholders will be balanced.

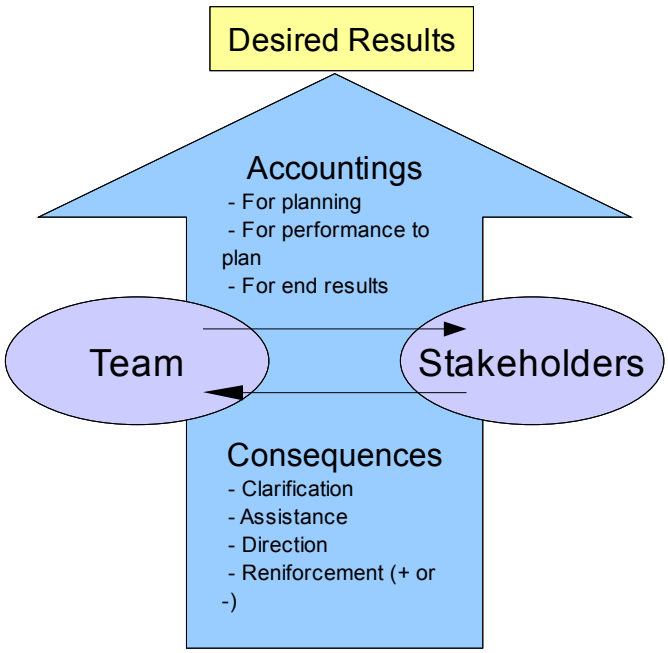

Figure 2: Lockheed Martin's MMC Accountability Model

The above views of accountability is largely concerned with holding the producers of software systems accountable for the behaviour and design of their systems for delivery to stakeholders. In a modern context the accountability of the end-users of such systems is also of considerable concern in providing trust in systems and in other users of the system. The rest of this chapter will be largely concerned with those issues particularly in the context of electronic transactions in current and network service provision platforms.

2.2 Accountability Protocols

The purpose of traditional network security protocols is to provide secure communications in insecure networks. These protocols prevent unauthorised agents or persons from obtaining private data or impersonating another entity. These protocols provide protection where there is trust established between the network resource and the authorised user. They do not provide any protection where there is no such trust. In a scenario where two participants in an electronic exchange of data or services are unknown to each other, more stringent protocols are required to ensure the identity of the participants and to account for their actions during the interaction. One

method of strengthening trust and achieving this goal is a preliminary registration process such as SET[2] or Visa 3-D[3]. Accountability protocols go further than this and provide the ability to ensure two further elements are present in the interaction [BELLA-06]:

1. *Non-repudiation of actions*: neither party can deny their actions in the interaction.
2. *Fairness*: Both participants receive the expected outcome of the transaction, or neither do.

An accountability protocol provides lasting evidence about actions performed by the peers. In general such protocols involve three parties, the two participant peers and a trusted third party (TTP). [KREM-02] provides a set of definitions relating to non-repudiation including definitions of different types of TTPs:

- **Inline TTP**: A TTP involved in each message transmitted during the protocol.
- **Online TTP**: A TTP involved in each session of the protocol but not each message.
- **Offline TTP**: A TTP only involved in a protocol in the case of incorrect or dishonest behaviour, or in the case of network failure.
- **Neutral TTP**: A TTP whose assistance is not pre-conditioned by knowledge of the the information to be exchanged.
- **Transparent TTP**: An offline TTP which produces evidence indistinguishable from the message exchange in a faultless case.

In a message interchange between A and B, non-repudiation of origin (NRO) provides a validation to B that the message originated from A, while non-repudiation of receipt (NRR) provides validation to A that B received the message. Such non-repudiation is normally brought about through the use of signatures, encryption, notarisation and data integrity mechanisms. [KREM-02] provides a detailed description of several fair non-repudiation protocols with no TTP, with inline, online and offline TTPS. Two such protocols using online TTPs are *the fair non-repudiation protocol* [ZHOU-96] and *the certified email protocol* [ABADI-02]. These are both described briefly here. It should be pointed out that for the requirements of a suitable auotopoietic peer-to-peer system under investigation, reliance on a TTP is undesirable. With this in mind two further protocols without the need for a TTP are also discussed, namely the *Markowitch and Roggeman protocol* [MARK-99] and the *Mitsianis protocol* [MITS-01]. Further details of all these protocols are available in the corresponding papers and of other protocols in [KREM-02].

[2]Secure Electronic Transaction (SET) is a standard protocol for securing credit card transactions over insecure networks, specifically, the Internet.

[3]Visa 3-D Secure is an authentication technology that uses Secure Sockets Layer (SSL/TLS) encryption and a Merchant Server Plug-in

2.2.1 The fair non-repudiation protocol

The fair non-repudiation protocol [ZHOU-96] provides the ability nether the sender or receiver of a message to obtain an advantage due to the transmission. The protocol makes use of a TTP . The main idea behind the protocol is to split the message definition into two parts, a commitment and a key. The commitment is exchanged between the two participants and the key is lodged with the TTP.

Notation:

A: originator of the non-repudiation exchange

B: receiver of the non-repudiation exchange

TTP: on-line trusted third party

M: message sent from *A* to *B*

C: commitment (ciphertext) for message *M* (*M* encrypted with key K)

K: message key defined by *A*

$NRO = sS_A(f_{NRO}, B, L, C)$: Non-repudiation of Receipt of Origin for M

$NRR = sS_B(f_{NRR}, A, L, C)$: Non-repudiation of Receipt of Receipt for M

$sub\_K = sS_A(f_{SUB}, B, L, K)$: proof of submission

$con\_K = sS_T(f_{CON}, A, B, L, K)$: confirmation of *K* issued by *TTP*

An assumption is made that all parties have a private signature key and relevant public keys. A label *L* links all messages in a session together. Flags (e.g. f_{SUB}) indicate the purpose of the message.

The Protocol:

1. $A \rightarrow B$: f_{NRO}, B, L, C, NRO
2. $B \rightarrow A$: f_{NRR}, A, L, C, NRR
3. $A \rightarrow TTP$: f_{SUB}, B, L, K, sub_K
4. $B \leftrightarrow TTP$: $f_{CON}, A, B, L, K, con\_K$
5. $A \leftrightarrow TTP$: $f_{CON}, A, B, L, K, con\_K$

The Steps:

1. A → B: A sends *NRO* to *B*.
2. B → A: After receiving the *NRO* and *C* from *A, B* can decide to acknowledge and send *NRR* to *A*. *B* does not know the meaning of *C* without *K*.
3. A → TTP : After comparing the label *L*, *A* lodges its private key with *TTP*
4. B ↔TTP: Using ftp *B* collects the key *K* from *TTP* and can decipher *C*.
5. A ↔TTP: Using ftp *A* collects retrieves *con*_k from *TTP*.

At any of the steps 1 through 4, either party can halt the process without giving an advantage to the other, providing fairness.

2.2.2 The certified email protocol

The certified email protocol [ABADI-02] provides non-repudiation for email delivery. Unlike the previous protocol, no public key infrastructure is used in this protocol. While the TTP has encryption and signature keys, A and B only have a password to TTP. In order for the protocol to work correctly A and B must agree in advance on a challenge response mechanism.

Notation:

$c = mk$	ciphertext of m using key K
$h_A = hash(q, r, c)$	hash of challenge q, response r, cipher c
$h_B = hash(q, r, c)$	hash of challenge q, response r, cipher c
$S2TTP = \{S, k, R, h_A\}\ eK_{TTP}$	certificate of *TTP*, signed with *TTP*'s public key
$RR = S2TTP\ sK^{-}_{TTP}$	return receipt, signed with *TTP*'s private key

The Protocol:

1. $A \rightarrow B$: TTP, $c, q, S2TTP$
2. $B \rightarrow TTP$: $S2TTP, pwd_B, h_B$
3. $TTP \rightarrow B$: k, h_B
4. $TTP \rightarrow A$: RR

The Steps:

1. $A \rightarrow B$: *A* sends *B* the encrypted mail *c*, a challenge *q* and a certificate for *TTP* containing the symmetric key *k* and a hash linking *c* to the response *r*.
2. $B \rightarrow TTP$: *B* calculates the response *r* to the challenge *q* and creates the hash which is passed to the *TTP* with the *TTP* password, pwd_R. This is done over a secure SSH channel.
3. $TTP \rightarrow B$: The *TTP* decrypts and verifies the certificate and verifies that h_A matches h_B. When this is verified, the *TTP* replies with the key *k* over the ssh channel set up in the previous step.
4. $TTP \rightarrow A$: The TTP returns the signed return receipt *RR* to A. This RR is an NNR. If the certificate received in the *RR* by *A* matches its stored certificate *A* will authenticate *B*.

In the case of the certified email protocol a sender A can send an email to a receiver B, so that B reads the message only if A receives the corresponding return receipt.

In both of the above protocols, the TTP does not see the encrypted message or email, thus reducing the need for strong trust in the TTP.

2.2.3 Markowitch and Roggeman protocol

The Markowitch and Roggeman protocol [MARK-99] is a non-repudiation protocol designed to provide probabilistic fairness without the need for a TTP. The protocol is iterative and is such that except at the final iteration, neither entity is at an advantage, thus providing fairness.

During the set-up, *A* who wants to send a message m to B, picks a random number n which will determines the number of iterations of the protocol. A keeps the value of n as a secret and selects a set of random n-1 values ri and a key k, the random values and the key having the same size.

Notation:

$X \rightarrow Y$	transmission for *X* to *Y*
$h()$	a collision resistant one-way hash function
$E_k()$	a symmetric-key encryption function under key *k*
$D_k()$	a symmetric-key decryption function under key *k*
$E_X()$	a public-key encryption function under *X*'s public key
$D_X()$	a public-key decryption function under *X*'s private key
$S_X()$	the signature function of entity *X*
m	the message sent
k	the session key
$c = E_k(m)$	the cipher of *m* under session key *k*
$l = h(m,k)$	a label that in conjunction with (*X,Y*) uniquely identifies a protocol run
f	a flag indicating the purpose of a message

The Protocol:

1. A → B: $f_{EOO}, B, l, c,$ EOO
2. B → A: $f_{EOR}, A, l,$ EOR
3. A → B: $f_{EOOk,1}, B, l,$ l, $r_1,$ $EOO_{k,1}$
4. B → A: $f_{EORk,1}, A, l,$ $EOR_{k,1}$

...

2n-1. A → B: $f_{EOOk,n-1}, B, l, n$-l, $r_{n-1},$ $EOO_{k,n-1}$

2n. B → A: $f_{EORk,n-1}, A, l,$ $EOR_{k,n-1}$

2n+1. A → B: $f_{EOOk,n}, B, l, n, k,$ $EOO_{k,n}$

2n+1. B → A: $f_{EORk,n}, A, l,$ $EOR_{k,n}$

The Steps:

1. A → B:	*A* initiates the protocol by sending the cipher *c* to *B*
2. B → A:	*B* acknowledges the receipt of the cipher
3. A → B:	*A* sends the first of the *n*-1 random values r_1 as well as the evidence of origin for r_1
3. B → A:	*B* confirms the reception of this value and the process continues
.	
.	
2n-1. *A → B:*	*A* transmits the final random value r_{n-1} and the corresponding EOO
2n. B → A:	*B* acknowledges the receipt of r_{n-1}
2n+1. *A → B:*	*A* transmits the the deciphering key *k*
2n+2. B → A:	*B* acknowledges receipt of *k* (which is indistinguishable from the the random values. After a predetermined delay or having received a notification from *A*, *B* calculates *m* = *Dk(c)*.

Before the final *2n*+1 step, B did not receive anything anything usable. The only way B can detect whether he received the key is by deciphering c using the value he has received. However, this computation is supposed too long compared to the time before A received the EOR and will not continue.

2.2.4 Mitsianis protocol

The *Mitsianis* protocol [MITS-01] is similar to the Markowitch and Roggeman protocol in that it uses a iterative approach to passing elements from *A* to *B*. Initially *A* sends the cipher *c* of the message *m* under a secret key K_δ to *B*. Once *B* acknowledges the receipt of the cipher, *A* adds a padding of size ω to K_δ to obtain K'_δ. The size of the padding chosen secretly by *A*. *A* ciphers this new padded key with a shared session key K_λ and obtains the ciphered key K_ς. *A* now chooses secretly a random value *n* and splits K_ς into *n* parts k_i of different random lengths. Now *n* 2-moves iterations begin. Each iteration involves *A* a k_i (in the order of the split) and *B* acknowledges the receipt. When *B* acknowledges receipt of the last parts, k_n of K_ς, *A* possesses the NOR of of the message *m* and *B* can retrieve the session key to recover the message.. This is done by composing K_ς by concatenating all the parts k_i received, deciphered using K_λ giving K'_δ. As *B* knows the size of the session key K_δ, the padding can be extracted from K'_δ to obtain K_δ and *B* can decipher the message.

2.3 Accountability in Network Services

As previously noted, traditional security mechanisms are not sufficient in providing adequate accountability facilities in electronic communications. This is becoming more evident as services grow more complex and inter-dependent. In emerging service provision platforms, services often consist of a set of interacting components spanning several trust domains. In grid computing or peer-to-peer systems, these components may run on infrastructures controlled by untrusted third parties. In this scenario fully accountable services are crucial in providing guarantees of correctness and also for increasing trust among participants to wards each other and in the system as a whole.

Yumerefendi and Chase [YUM-04] provide 3 fundamental properties of accountability in terms of service provision:

- **Undeniable**: The actions of an accountable actor (a service or its clients) are provable and non-repudiable.
- **Certifiable**: A client, peer or external auditor can verify that an accountable service is behaving correctly and prove any deviations from this correct behaviour to a third party.
- **Tamper-evident**: Any attempt to corrupt a service state should carry a high probability of detection.

Yumerefendi and Chase argue that *"system builders should view accountability as a first-class design goal of services and federated distributed systems..."*. In examining accountability as a general design goal, they conclude that *"a key limitation of accountable design is that it is not fully general and that it must be 'designed in' to application structure and protocols"*. They provide a framework for accountable services which is an extension of an approach know as KASTS [MAN-02]. The purpose of this framework is to provide accountability for a general service class which accesses and updates internal state in response to client or peer requests. The process is shown below in Figure 3. Execution occurs in a sequence of numbered rounds. In each round the service updates one internal state variable. At the end of each round the service publishes a signed, timestamped non-repudiable digest of its internal state and provides this to an external observer.

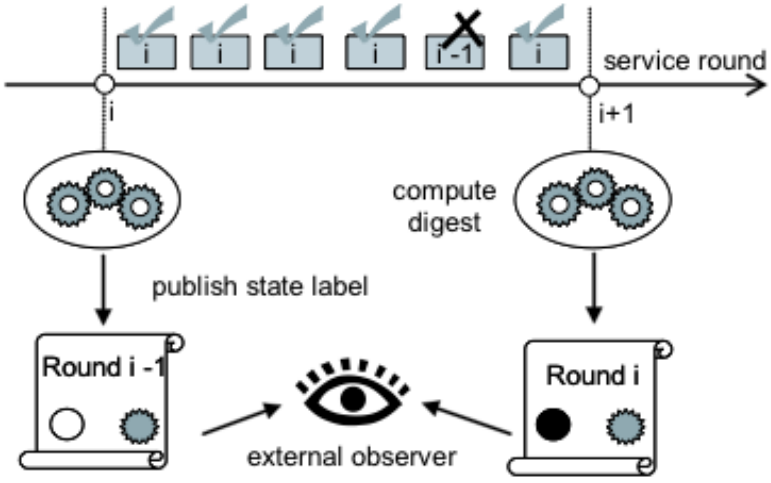

Figure 3: Round Processing

When a client requires an execution from a service, it digitally signs the request. After verifying that the request is valid the service executes the request and on completion returns a result with cryptographic evidence certifying its correctness relative to the published round requests. It should be noted that this approach requires access to the internal state of the services in question and as such is unsuitable for the needs of the service driven peer-to-peer approach being pursued.

PlanetFlow

A real world implementation of a network service accountability platform is that of PlanetFlow [HUA-06], which is the network auditing tool for PlanetLab.

PlanetLab [BAV-04] is a geographically distributed overlay network supporting deployment and evaluation of new planetary-scale network services. At the time of writing PlanetLab consists of 808 nodes at 401 sites. The purpose of the PlanetFlow tool is to audit the PlanetLab network resources to facilitate complaint resolution, limit liability and minimise problematic behaviour. As PlanetLab is a wide-area network of shared resources interacting with the public Internet a suitable accountability framework is necessary for resolving complaints. The policy is to put people who complain about PlanetLab traffic in direct contact with the researchers whose nodes are responsible for the traffic in question. The PlanetFlow service runs on every PlanetLab node. The PlanetFlow components can be seen below in Figure 4. The components are briefly described as:

- A flow collector classifying outgoing packets into IP flows
- A database to store the flows
- Web and administration tools to query the database
- A central server for storing, querying, archiving flow data from all nodes

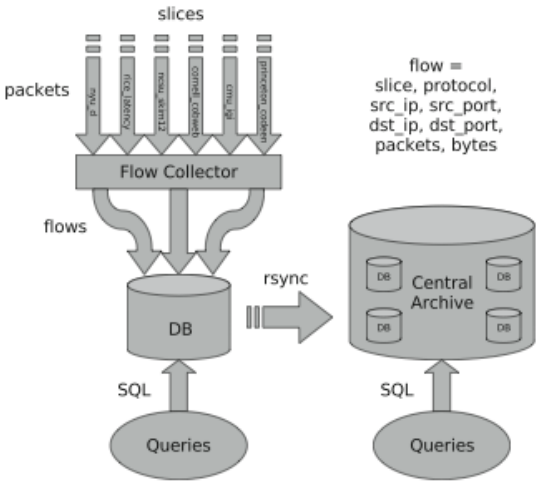

Figure 4: PlanetFlow Components

2.4 Accountability and Service Oriented Architectures

The rise of Web Services and an increased interest in Service Oriented Architectures (SOA) in general has provided business-to-business (B2B) type transactions with a set of open inter-operable standards through which Internet business transactions can be performed. As an approach to building IT systems, SOA connects

applications across a network via a common communications protocol, allowing organisations to reuse old software. Web Services have in fact become the de-facto standard for such B2B interactions. This proliferation of Web Services in providing inter-operable, scalable and composed B2B services has a drawback in terms of providing accountability; there is no standardised approach to providing accountability within the current set of protocols and standards. While there exist specifications on Security [WSS] and on Trust [WST], these are only applicable within trusted domains. This lack of clearly specified accountability models has prompted some work in the area of accountability in Web Services/SOA (e.g. [ROB-05], [ZHANG-06], [ZHANG-07]).

It has become common to standardise B2B interaction with message-exchange patterns. RosettaNet define a set of externally observable elements of B2B exchange in terms through a set of Partner Interface Processes (PIPs). Typically each message is acknowledged with a confirmation of receipt. Figure 5, below shows the delivery of a business message and its associated acknowledgements. The initial message receipt is acknowledged by B, who in turn sends a message signifying whether the original message was valid or invalid. Upon receipt of this message, A again acknowledges receipt.

Figure 5: Business Message Delivery Pattern

2.4.1 Fair Non-repudiable Web Services

Robinson et al, [ROB-05] describe a protocol (which they call WS-NRExchange) to enable non-repudiable and fair interactions with Web Services. Their protocol makes use of the non-repudiation protocol of Zhou and Gollman [ZHOU-96] and is based on applying such a protocol to the message exchange pattern shown in Figure 5. This is achieved by the introduction of an inline TTP, the Delivery Agent (DA). This is shown in Figure 6, below.

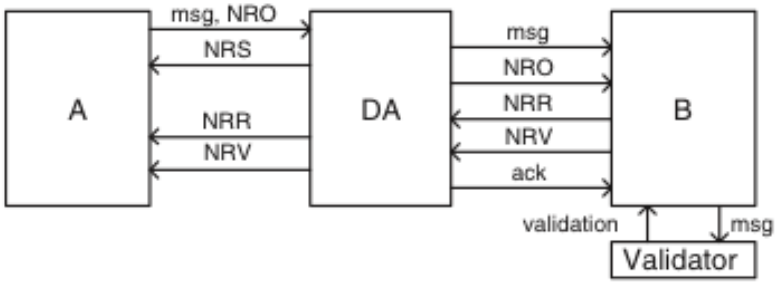

Figure 6: Business Message Pattern with inline TTP for Non-repudiation

This approach provides four types of evidence for non-repudiation.

1. NRO: non-repudiation of origin; *msg* originated at A
2. NRS: non-repudiation of submission: *msg* was submitted by A
3. NRR: non-repudiation of receipt: *msg* was received by B
4. NRV: non-repudiation of validation: *msg* was validated (positively or negatively) by B

In terms of implementation Robinson et al suggest the use of interceptors to abstract the non-repudiable message exchange from the service endpoints. Most implementations of web services have such interceptors available (e.g. Axis Handlers). This approach is shown below in Figure 7, which depicts the proposed WS-NRExchange architecture.

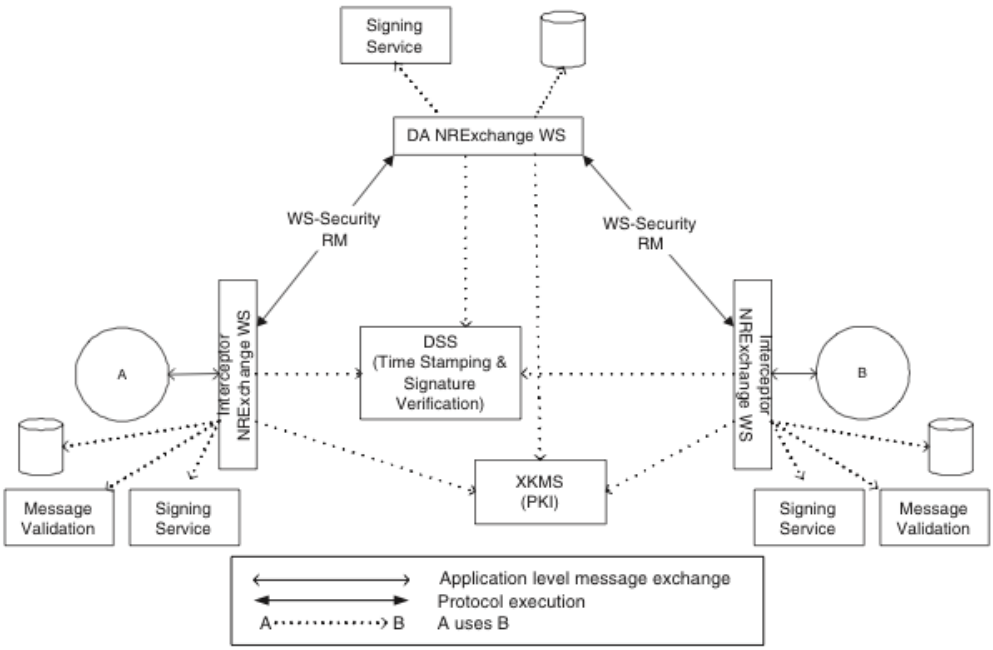

Figure 7: WS-NRExchange Architecture

2.4.2 Accountability in SOAs

Zhang et al, [ZHANG-06], [ZHANG-07] present a model framework for accountability in SOAs which leverage Bayesian Networks in diagnosis of faults and a learning process linked with reputation. This model is more concerned with the self-healing aspect of autonomics through the monitoring of faults and reasoning, than providing a system of non-repudiation service provision.

They recognise that in the real world, business processes are often naturally organised hierarchically. This hierarchical management is efficient in that it aids the large-scale service failure as it facilitates a divide and conquer approach.

They present a 3-D approach (Detect, Diagnose and Defuse), which is implemented as follows:

1. **Detect**: Agents monitor behaviour of atomic services according to a predefined and report exceptions to an Accountability Authority (AA).
2. **Diagnose**: The AA makes use of Bayesian Network reasoning to discover the root of the problem when Service Level Agreement (SLA) violation occurs. This reasoning makes use of reported monitoring information gathered at the Detect phase.
3. **Defuse**: The AA updates the reputation of each atomic service based on the outcome of the Diagnosis phase. The AA then re-composes the service through selection of services least likely to violate SLAs, based on reputation.

In [ZHANG-07], they identify a set of new challenges for accountability techniques when deployed in SOAs:

- **Causality**: The outputs from upstream services have impacts on downstream services. Accountability approaches need to be capable of dealing with these causal relationships in these graph model based systems.
- **Scalability**: Fault diagnosis mechanisms need to scale to large-scale SOA systems.
- **Efficiency**: In terms of gathering runtime status information on the performance of services in large-scale SOA deployments, strategies must be designed to collect appropriate subsets of this status information while still allowing for accurate fault diagnosis.
- **Uncertainty**: In SOAs, the behaviour of services is often uncertain at runtime. Accountability mechanisms need to built on a probability and statistical theory basis.
- **Dynamicism**: Service behaviour in SOAs is not static and evolves over time. Mechanisms are required to evaluate trustworthiness of services as their behaviour evolves.

They designed a Hierarchical Diagnosis Algorithm (HDA) and a Accountability Authority (AA) to which faults are reported in order to address these issues in SOAs.

The architecture is depicted below in Figure 8.

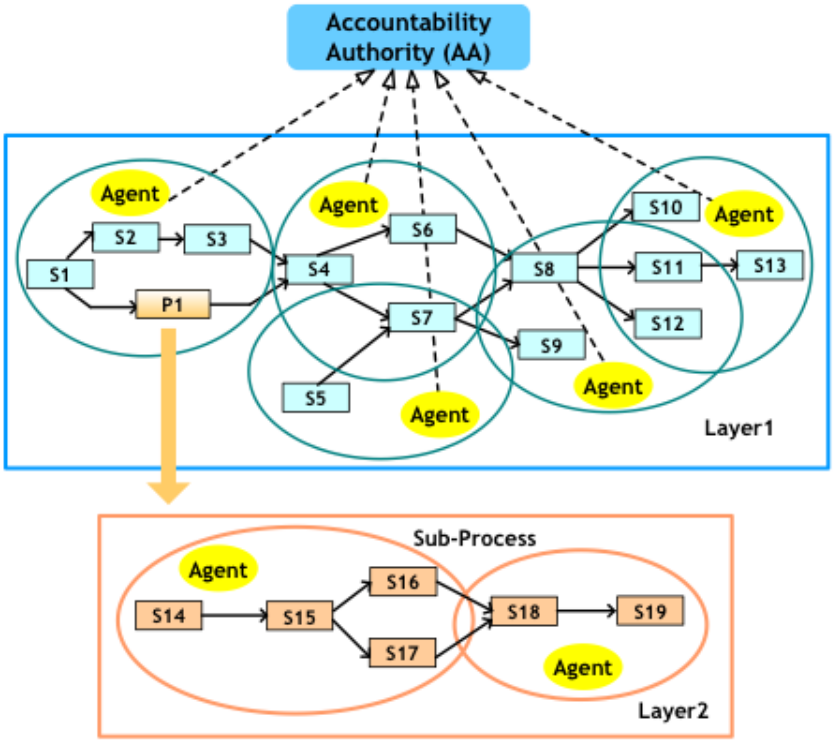

Figure 8: An SOA Accountability Architecture [ZHANG-07]

2.5 Accountability and Peer-to-Peer Systems

Peer \Peer\, n.

One of the same rank, quality, endowments, character, etc.; an equal; a match; a mate.

[1913 Webster]

Peer-to-peer networks operate through the leveraging of resources made available by its participants rather than relying on a set of servers providing a set of centralised finite resources. A pure peer-to-peer network operates on the basis that all peers are equal and each operates as both a client and a server in terms of providing and consuming services and/or content. These types of (pure peer-to-peer) networks have become the subject of much research and commercial interest in recent times due to their inherent lack of a single point of failure as resources are replicated across nodes, low cost of deployment in large-scale roll-out (due to the non-requirement of centralised management systems) and the fact that as more resources join the network, the greater the capacity of the network (due to the fact that it is the participants who provide resources). Another element of peer-to-peer technology that is of benefit on a social level is that, by its nature, it empowers its users who in effect collectively own the network and its resources. It is partly this reason that peer-to-peer networks have become the technologies of choice in the area of digital ecosystems.

In terms of accountability, peer-to-peer networks bring the same challenges that are in place for accountability in traditional client server models, i.e. Non-repudiation, fairness, etc. In fact, in peer-to-peer the requirements could be considered more important due to the fact that participants now operate in entirely untrusted environments

without any centralised authentication mechanisms providing trust and strong user identity. In addition, the lack of centralised control brings its own challenges in terms of replication of (sometimes sensitive) data and resources across peers in untrusted environments.

The remainder of this section will present research published in the area of accounting and accountability in peer-to-peer networks and also projects which have been active in this field. While much of the work on accountability in peer-to-peer systems is motivated by a desire to reduce the 'freeloading' which can occur in resource sharing systems, there are also stronger real world economic drivers where these systems are used in sustaining digital communities with service provision.

2.5.1 JXTA

Project JXTA[4] started as a research project incubated at Sun Microsystems to address the peer-to-peer space. JXTA technology is a set of open, generalised peer-to-peer protocols that allow any connected devices on the network to communicate and collaborate. JXTA peers create a virtual network where any peer can interact with other peers and resources directly even when some of the peers and resources are behind firewalls and Network Address Translations (NATs) or are on different network transports.

2.5.1.1 JXTA Compute Power Market Project

The Compute Power Market (CPM)[5] Project uses an economic theory approach in managing computational resource consumption and provision in peer-to-peer computing. It allows application users to access computing power with ease and simplicity. It also allows consumers to choose computing power/resource providers that offer cost-effective services on demand. The project is concerned with creating a competitive market approach to service oriented peer-to-peer computing. A position paper explaining the project is available [BUY-01].

2.5.1.2 JXTA Balance Project

Project Balance is a new project born out of issues raised in accounting for peer-to-peer networks in other projects such as CPM and Payment project[6]. The Balance project[7] intends to invent a new form of peer-to-peer based eCommerce (pCommerce) to address the accounting and payment issues in peer-to-peer networks. A close collaboration is intended with *Payment project* and the *CPM project.*

2.5.1.3 JXTA Metering and Monitoring Project

The primary goal of the JXTA Metering and Monitoring project[8] is to provide a dynamic and extendible framework for gathering and reporting metrics about JXTA services running within groups of peers (called PeerGroups). The types of metrics maintained for a service will be defined on a peer service implementation with

---

4 *Project JXTA,* http://www.jxta.org/

5 *The Compute Power Market (CPM) Project ,* http://www.computepower.com/

6 *JXTA Payment Project*, http://payment.jxta.org/

7 *JXTA Balance Project*, http://balance.jxta.org/

8 *JXTA Metering and Monitoring Project*, http://meter.jxta.org/

an XML representation of each type of metric. In addition to providing an API for obtaining metrics from PeerGroups running locally, the optional JXTA Peer Information Protocol (PIP)[9] is a specified means for obtaining these metrics from remote peers.

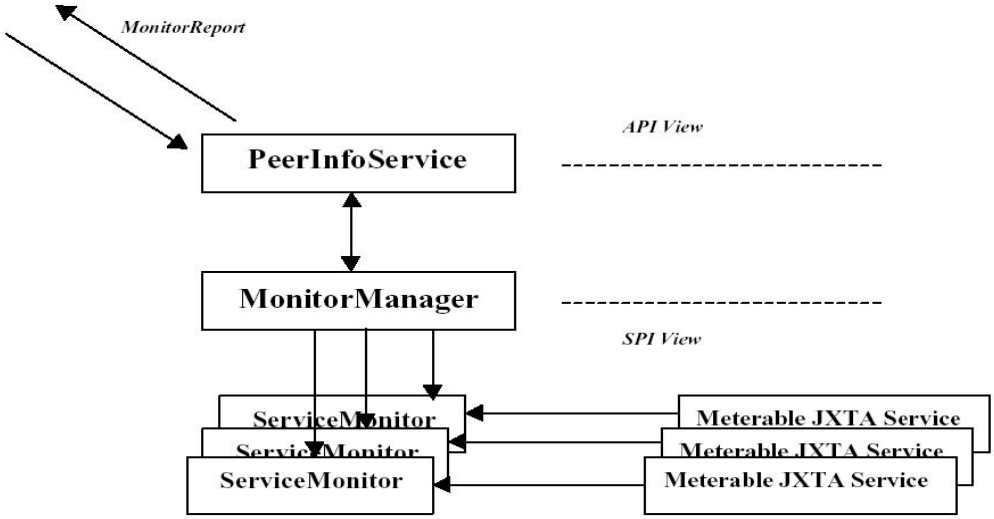

Figure 9: JXTA MMP Architecture

The MonitorManager choreographs all cumulative and asynchronous reporting of metered data. It provides a Service Provider Interface (SPI) view to the underlying Services via ServiceMonitors. The MonitorManager interacts with all registered ServiceMonitors delegating appropriate information and requests to the service-specific ServiceMonitors which will internally optimise and collect their own data.

2.5.1.4 JXTA-Monitor

JXTA-Monitor[10] is a monitoring tool that captures messages between JXTA Peers. While the project seems to focus on these messages as a means of debugging JXTA applications, it is a simple step to adapt this as a means of metering JXTA services for accounting purposes. The architecture is shown below and takes the approach of inserting the monitor as an intermediary between the peers exchanging messages by implementing it as a JXTA RendezvousPeer in the netPeerGroup. The project has provide a GUI implementation of this monitor, primarily used for debugging. The JXTA applications themselves do not have to be modified in order for the monitor to extract the messaging data.

---

[9] *JXTA Peer Information Protocol,* http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.html#proto-pip

[10] *JXTA-Monitor,* http://jxta-monitor.jxta.org/

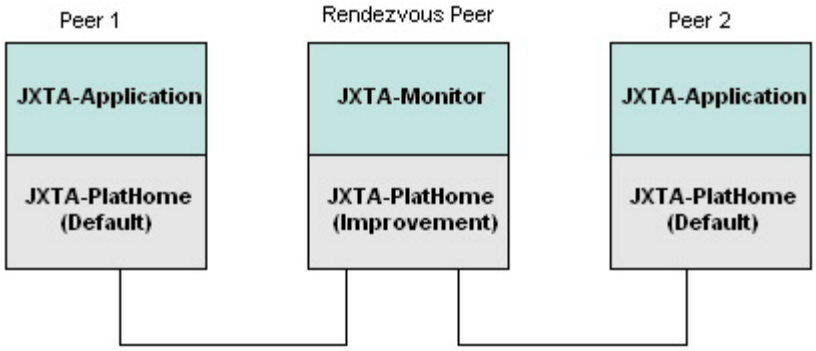

Figure 10: JXTA-Monitor Architecture

2.5.2 MMAPPS

The IST MMAPPS (Market Management of Peer to Peer Services) project[11] looked at how services can be traded in peer-to-peer applications. The project viewed the provision and consumption of services in peer-to-peer environments as a marketplace and enhanced schemes, such as micro-payments, with more innovative, non payment-based accounting schemes such as ratings, where individual members of a particular community receive a rating score based upon their contribution. This rating then affects how other members provide services to that individual.

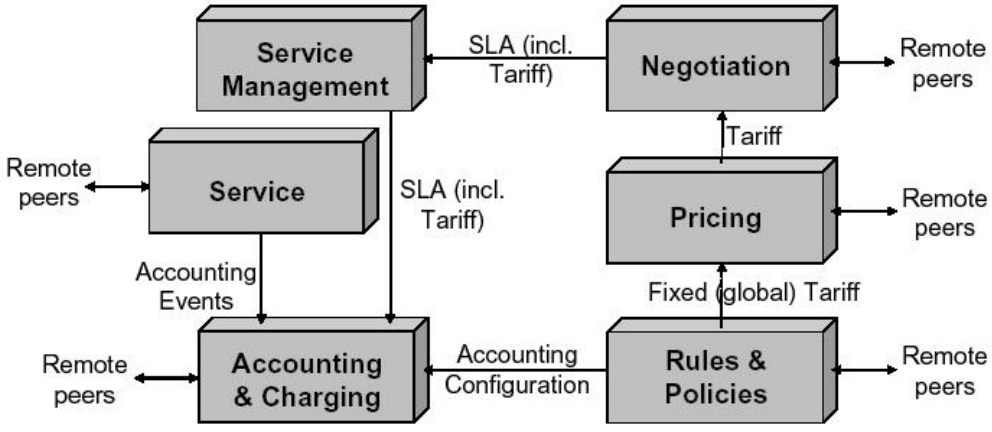

Figure 11: The Accounting and Charging module with associated modules in MMAPPS

During the service preparation phase, the pricing module selects the tariff which will be used to account for usage of a particular service. As the tariff has to know the characteristics of the service to account for, it is developed in advance closely together with the service. The corresponding tariff parameters, however, are

[11] *MMAPPS (Market Management of Peer-to-Peer Services),* http://www.mmapps.org *(domain has lapsed)*

usually set dynamically and can be different on every peer. Optionally, fixed application-wide tariff parameters can be used which need to be specified by the Rules & Policies module [MMAPPS-10].

MMAPPS have also adopted a token-based approach to provide accounting and distributed pricing facilities to peer-to-peer service provision. With this approach, tokens are traded for service usage. A file-sharing scenario is discussed in [HAUS-03]. The project has defined an abstract architecture [STRU-03] satisfying this token based approach which can be seen in the following diagram.

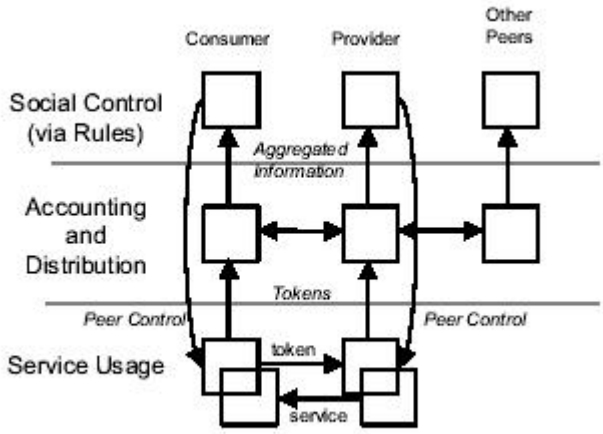

Figure 12: MMAPPS Token-based Peer-to-Peer Economics Architecture

2.5.3 Karma

Karma [VISH-03] is a framework for sustaining resource sharing peer-to-peer networks using a secure economic model. The motivation for the work is the avoidance of freeloading is resource sharing networks. The system s economic in that it keeps track of the resource purchasing power of peers in the network.

Each peer is represented by a scalar value called a *karma* (much like a community currency) which represents the purchasing power of that peer. If a peer requests a resource, but does not have sufficient karma to purchase the resource, the transaction may not proceed. Participants are thereby incentivised to contribute as many resources (or more) as they consume. The framework provides properties of non-repudiation, certification and atomicity, protecting against both malicious providers and consumers. In addition, periodic correction to the outstanding karma in the system is performed in order to reduce the effects of inflation or deflation. Inflation can occur when peers use up their balance to consume resources and then leave the system. Deflation can occur when peers accrue large balances and then leave the system.

It is assumed that the minimum number of nodes in the network is k. Karma maintains its internal state via a peer-to-peer distributed hash table (DHT). The bank-set $Bank_A$ is a set of peers that independently maintain the karma balance of peer A. Each participant is assigned a unique identifier in a circular identifier space e.g. *NodeID(A)*. The bank set $Bank_A$ is the k closest nodes in the identifier space to *HASH(NodeID(A))*. This

mapping is chosen as it allows the implementation to be layered on top of an existing DHT such as Pastry [ROWS-01]. Using this method the routing to each of the bank-set nodes can be performed efficiently. Each node in the bank-set $Bank_A$ independently stores the karma balance of *A*, signed by *A*'s private key (making the value tamper resistant). Each bank-set node also contains a transaction log of recent payments. Karma is not concerned with how the resources are shared or how a price is agreed upon for resource.

When a new node joins the system, a randomly selected public/private key pair is provided and using a value *x*, such that $md5(K_{public})$ *equals* $md5(x)$, in the lower *n* digits, where *n* is a system wide parameter, the new node's *nodeID* is set to $md5(K_{public},x)$. The node certifies that this calculation has been performed by signing challenges from its allocated bank-set with its private key.

The value/resource transfer protocol is shown in Figure 13.

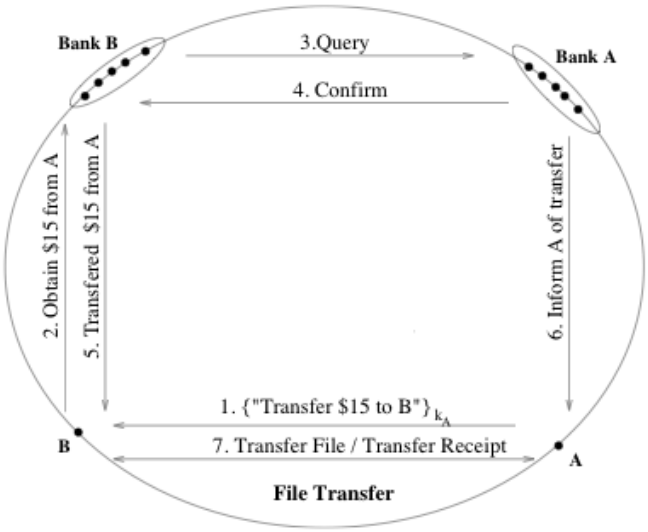

Figure 13: KARMA resource/value exchange protocol

Initially *A* sends *B* a signed authorisation for $Bank_A$ to transfer the agreed karma amount to $Bank_B$. *B* forwards this message to $Bank_B$, which in turn contacts $Bank_A$. If *A* has sufficient karma to satisfied the transfer amount, the amount is deducted from $Bank_A$'s balance and credited to $Bank_B$. $Bank_B$ then notifies *B* that the transfer took place and *B* releases the resource to *A*. The initial transfer from *A* includes the balance *A* will have at the end of the successful transaction (signed by *A*'s private key). Similarly when *B* passes this message to its bank-set, it includes a signed version of its own expected balance at the end of the transaction. This mechanism ensures that each bank node contains the latest balance corresponding to their client node.

2.5.4 PeerMint

In 2005 Hausheer and Stiller published a decentralised scheme for accountability in peer-to-peer networks which they called *PeerMint* [HAUS-05]. The work was performed as part of the MMAPPS[12] project and was concerned with accountability mechanisms for commercial peer-to-peer applications. The provided scheme can be used as a means of ensuring fair sharing of resources or as a means of applying charging and payment

[12] *MMAPPS (Market Management of Peer-to-Peer Services),* http://www.mmapps.org *(domain has lapsed)*

mechanisms to peer-to-peer applications. The work introduces the concept of session peers in addition to account peers in order to reduce significantly the possibility of collusion between peers. The session peers are responsible for maintaining balances and updates for the current session and for validating actions against a predetermined Service Level Agreement (SLA) between the peers.

While the accounting data is secure in that it ensures availability and integrity of data, it does not provide mechanisms for confidentiality or privacy. The interface to the accounting mechanisms is generic and can be easily applied to various environments.

PeerMint takes a redundant approach to both session peers and account peers through the replication of data across a set of peers. The selection of account peers is made in a similar way to to the hashing method of Karma [VISH-03] in order to optimise routing. The selection of session peers also uses this approach but the hash is performed on a combination of the peer IDs and a timestamp denoting the point of session initiation. In both cases (account and session peers), when a peer goes offline a new peer is selected to take its place. The new peer obtains the balance from the remaining peers associated with the account or session. The model is shown below in Figure 14. New peers are shown as a dashed circle.

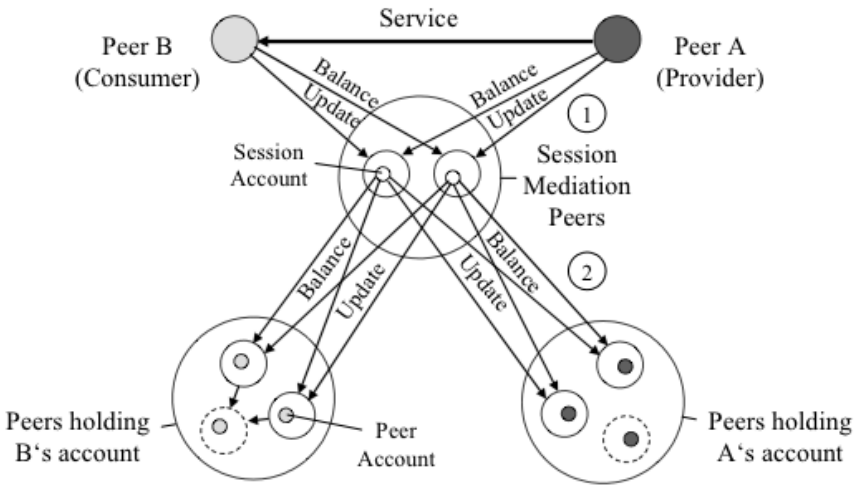

Figure 14: PeerMint Distributed Redundant Accounting

The PeerMint scheme was implemented in MMAPPS using a local instance of FreePastry[13], an open source implementation of Pastry [ROWS-01]. The experimental results showed that the scheme's message overhead increased slowly in small networks, but that this overhead levels off as the network grows, thus showing that the scheme provides scalability. No real world implementation is available.

[13] FreePastry, open-source implementation of Pastry, http://freepastry.rice.edu/FreePastry/

3 Accountability Model for Digital Ecosystems

This section will examine the requirements for providing accountability in peer-to-peer digital ecosystems and the selection of previously developed frameworks to provide this functionality. The initial model for accountability is provided which is largely based on adapting two of the works researched in Section 2 (PeerMint and SOA Accountability Architecture). The limitations of this model are discussed with a view to integrating a suitable solution in the next iteration. It should be noted that this model is an initial general approach to providing accountability in autopoietic peer-to-peer networks and its evolution is dependent on decisions of network design coming from WP3.

3.1 Requirements

Below is a list of the technical requirements for accountability in peer-to-peer deployed digital ecosystems.

3.1.1 Decentralised

One of the fundamental requirements for digital ecosystems is that there is no single point of failure. If any one node (or any sets of nodes) becomes unavailable, the system must be capable of recovering completely without any external intervention. A suitable accountability solution requires therefore that there is no dependence on a single centralised accountability manager or co-ordinator and that all functionality is distributed across the system.

3.1.2 Service Composition

A crucial aspect of digital ecosystems is collaboration between participants. In this sense it is important that services and resources can be combined to create new services. This service composition needs to be dynamic and cannot be known in advance. An accountability solution requires that such a dynamic composition of services can be seamlessly and efficiently accounted for.

3.1.3 Scalability

The number of peers in a digital ecosystem is arbitrary, ranging from a handful to several thousand. A suitable solution needs to work for large sets of peers as well as large service compositions.

3.1.4 Security

The implications for security when providing accountability in dynamic composing in digital ecosystems are wide ranging. Integrity of accounted data, availability of accounted data, confidentiality of accounted data, privacy of transactions al need to be considered when designing a suitable accountability model.

3.1.5 Contracts

Exchange of contracts or service level agreements prior to service consumption is a vital aspect of a commercial application of digital ecosystem deployment. It is desirable that aspects of these agreements can be assessed at runtime to ensure that parties are operating within the bounds of the agreement.

3.2 Model Considerations

Taking into account the above requirements, the relevant approaches outlined in Section 2 are addressed here with a view to adopting and extending the most appropriate ones for our needs. The ones related to peer-to-peer networks and SOA are discussed here.

WS-NRExchange

The fair non-repudiable web services framework described by Robinson et al, [ROB-05] provides a useful solution and architecture to providing accountability in webs services. It has one major drawback for our needs in that it uses an inline TTP as an integral part of its architecture. Given the purely distributed nature of our desired infrastructure, this approach is not compatible with our needs.

SOA Accountability Architecture

This approach is a very useful contribution in that it addresses directly the challenges of Causality, Scalability, Efficiency, Uncertainty and Dynamicism as identified by [ZHANG-07]. One element of this framework that is of concern is the centralised nature of the Accounting Authority. It can be that the functions of this element can be deployed as a distributed service and thus over come this drawback.

MMAPPS

The *MMAPPS* project examined how services can be traded in peer-to-peer applications. While the project itself proved a useful architecture for accounting and charging for these services, it did not directly address accountability in terms of non-repudiation etc. A separate element of the project provided such a mechanism (*PeerMint*) that in fact does provide such non-repudiation elements, although service composition is not addressed.

JXTA

JXTA and its associated accounting projects provide some degree of accounting for and management of computational resources and provision in peer-to-peer networks. Most of this work is concentrated on accounting mechanisms and does not address further issues of non-repudiation and fairness required to provide rigorous accountability functionality. Several of these sub-projects do not appear to have made much progress in the last 2 years. JXTA does not address service composition.

KARMA

Karma provides a framework for non-repudiable resource sharing among peers and uses a secure economic model to achieve this end. The approach satisfies much of what is required, but again does not address service composition.

PeerMint

PeerMint is a decentralised scheme for non-repudiation accountability in peer-to-peer networks. The *PeerMint* approach is very similar to that of *Karma*, but goes further through the introduction of session peers in order to reduce the possibility of collusion among peers in creating reputation and value. *PeerMint* does not consider service composition in its approach.

None of the above approaches satisfy all our needs. However by combing and adopting some of these approaches we can move closer to satisfying our requirements. The two models which most closely satisfy the requirements for our accountability model are that of *PeerMint* [HAUS-05] (for distributed accountability in peer-to-peer networks) and the *SOA Accountability Architecture* (which satisfies service composition accountability) developed in [ZHANG-07]. Here we analyse those models in terms of our challenges defined above. The table below shows a comparison of these approaches with respect to our requirements.

Requirement	*PeerMint*	*SOA Accountability Architecture*
Decentralised	**Yes** The model is fully distributed as it takes a redundant approach to the distribution of accountability data.	**No** Not a distributed approach as the Accountability Authority is centralised
Service Composition	**No** The model does not consider Service Composition as it is primarily concerned with binary transactions between peers.	**Yes** The primary concern of this architecture is to account for composed services in an SOA environment. In addition this approach address the challenges for accountability in SOAs, namely Causality, Scalability, Efficiency, Uncertainty and Dynamicism
Scalability	**Yes** The model has been tested to show that the message overhead levels off for larger scale deployments	**Yes** The model uses Bayesian Networks in diagnosis of faults and demonstrates in this regard.
Security	**Partly** Data integrity and availability are satisfied by this approach. Confidentiality and privacy aspects are not considered.	**No** No security aspects were considered
Contracts	**Yes** SLAs are exchanged before the transaction takes place.	**No** Contracts are not considered as a part of this design

Table 1: PeerMint and SOA Accountability Architecture Comparison

3.3 Model Description

From Table 1 we can see that some of our requirements are largely satisfied by one or both of these approaches. As *PeerMint* provides a fully decentralised approach that is suitable for our needs and the *SOA Accountability Architecture* provides a framework for accounting for composed services, it is likely that a combination of these approaches will provide most of what is required.

The model under development is a combination of the *PeerMint* model published by Hausheer and and Stiller [HAUS-05] combined with the *SOA Accountability Architecture* desigend by Zhang et al [ZHANG-07]. The

idea here is to take the concept of the *Accounting Authourity* introduced by Zhang and to deploy this as a distributed service and combine this with the distributed *PeerMint* model. The model is shown below in Figure 15. Although the diagram shows a simple transaction between 2 services, the introduction of the *Accounting Authority* provides the required functionality necessary for composed services. The role of the accounting authority in a composed service scenario is provided by the super-set of mediation peers.

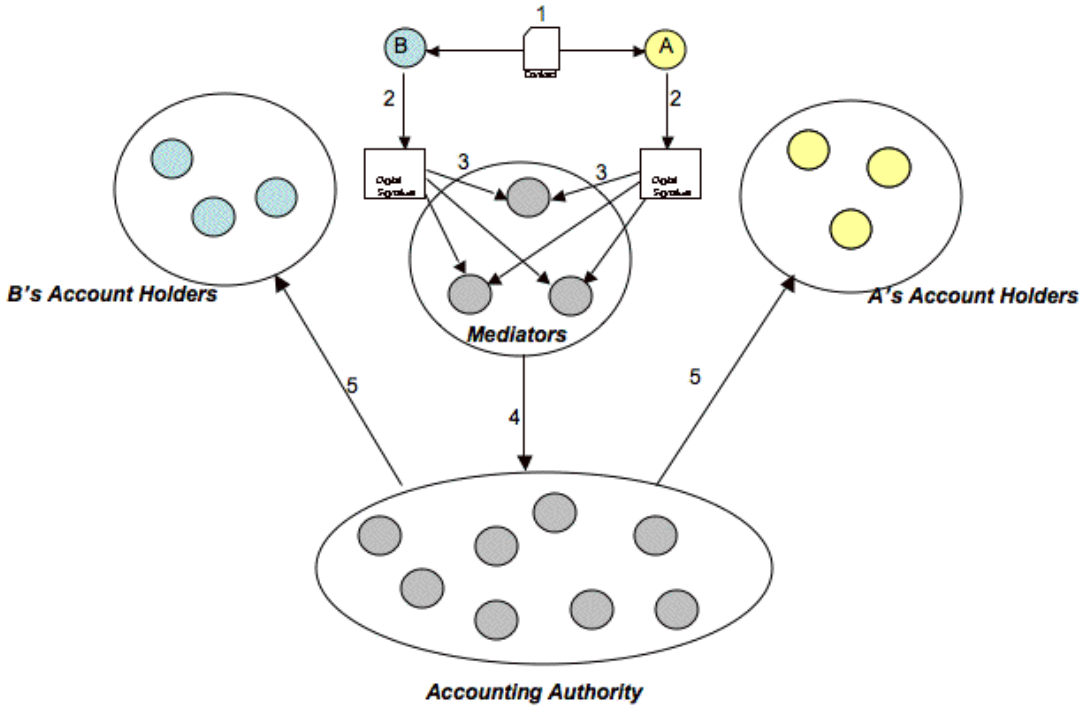

Figure 15: Distributed Accountability Model

The model shows two peers (A and B) interacting in a peer-to-peer services trading environment.

Prior to interaction between the peers, each peer is assigned a set of trusted peers as account holders. This assignment is based on historical availability of peers and this set of peers is updated over time as the system evolves. Upon joining the network, each peer is assigned a unique 128-bit peer ID calculated fron that peer's public key using a secure hashing method. This is the approach taken by both The public key is provided by the peer's identity (modelled through Task 4.1, Distributed Identity Model in OPAALS). The original selection of account holders is based on a combination of trustworthiness and leaf sets closest to the peer to be accounted for. The trustworthiness of these peers is determined based on the distributed trust model being developed within workpackage 4 in OPAALS. In a similar way (as in *PeerMint*) the session peers are selected using a hash on the combination of the interacting peers IDs in combination with a timestamp. Both the account holders and the session peers are normal peers in the network. An interaction between A and B is shown in the diagram and is explained as follows:

1. Initially the peers exchange a contract which includes the charging schemes they agree for service consumption. At runtime when the peers agree to interact, they are assigned a set of session mediation

peers. These mediation peers are provided with a copy of the contract. These peers are responsible for holding accounting data from both A and B for the current interaction session.

2. Each message passed between A and B each session is metered and the resulting usage data is digitally signed by the peer itself. In combination with the distributed identity model being developed and the availability of peers' public keys, these signatures can be verified.
3. The digitally signed usage data is then passed to the mediation agents who compare A's usage with B's to check that both are agreeing on how they are interacting and that these interactions conform with the previously agreed contract.
4. As the session evolves, the accounted data is passed to the super-set of mediation peers operating as the Accounting Authority and checked for integrity with respect to the overall service composition.
5. At the end of the session, the usage data is consolidated into a charge record by the Accounting Authority. These charge records are delivered to the corresponding set of account holders for both A and B.

3.4 Further Requirements

The above described model provides most of the functionality required to allow for distributed accountability in peer-to-peer networks where dynamic service composition is the enabler for multi-peer collaboration. However, some requirements are not satisfied fully by this model, namely security issues. Privacy is a security aspect which also need to be addressed in a future evolution of this model. In the model provided here, any peer which holds *A*'s public key can, in theory, access details of *A*'s account from its account holders. In the next evolution of this model an access control overlay needs to be added to overcome this issue. Furthermore, decisions on hashing methods, routing tables, leaf-sets and selection of account and session holders need to further considered as the network design decisions emerge from Workpackage 3.

In addition, further work needs to be undertaken with regard to receiving inputs from other parts of the OPAALS project team to better understand the requirements in terms of business models for community networks (Workpackage 7) and also socio-economic aspects (particularly that of community currencies) addressed in Workpackage 6.

4 Concluding Remarks and Future Directions

This document has provided an overview of accountability in computing and service driven architectures and introduced a model for accountability in peer-to-peer networks providing. Concepts and previous works on accountability in the computing domain have been researched and examined for their suitability for the task at hand. Some accountability protocols have been presented as well as frameworks for accountability in service oriented architectures and peer-to-peer networks. From this research and a set of requirements for accountability in digital ecosystems, an initial model has been developed drawing from two previously published models, *PeerMint* published by Hausheer and and Stiller [HAUS-05] combined with the *SOA Accountability Architecture* desigend by Zhang et al [ZHANG-07]. While this model does not satisfy all of the requirements it represents an evolving work in progress which needs to be augmented to include confidentiality in terms of access control in a future version.

Further work that needs to be conducted in future includes providing detailed specification of this model in terms of technical requirements for implementation through work package 5. This specification has considerable implications for the requirements of the peer-to-peer platform being developed and is essential for successful platform development. In addition the development of suitable contract models needs to be conducted. A model developed during the DBE project will be used as a basis for this contractual model specification.

In addition it is required that the output of this accountability data can be used in the development of reputation and trustworthiness of peers and services.

More work also needs to be conducted in bringing the models for trust, identity and accountability together in providing an evolving reputation and trust system which can be used in the selection of services for automatically composed services. Providing this functionality whilst maintaining security in terms of access to the accounted data is a considerable challenge.

Another potential avenue for future research is is where this accounted value transfer can be analysed with a view to optimising future interactions between participants with a view to sustaining the community as a whole.

Further work needs to be undertaken with regard to receiving inputs from other parts of the OPAALS project team to better understand the requirements in terms of business models for community networks (workpackage 7) and also socio-economic aspects (particularly that of community currencies) addressed in workpackage 6.

5 References

[ABADI-02] Abadi, M., Glew N., Horne B., Pikas, B., 2002, *Certified email with a light on-line trusted third party: design and implementation*, in Proceedings of the 11th international conference on World Wide Web, ACM Press, New York

[BAV-04] Bavier, A., Bowman, M., CUller, D., Chun, B., Krlin, S., Muir, S., Peterson, L. L., Roscoe, 2004, *Operating System Suport for Planteray-Scale Network Services*, in Proceedings of the First Symposium on Networked Systems Design and Implementation, ACM Press, New York

[BELLA-06] Bella, G ., Paulson, L. C., 2006, *Accountability protocols: Formalized and verified*, in ACM Transactions on Information and System Securit, Volume 9, pp 138 - 161, ACM Press, New York

[BULL-06] Bullock, G., 2006, *Governance, Accountability, and Legitimacy*, Working Paper Series, Consumer Information Laboratory, University of California, Berkeley, available at nature.berkeley.edu/infolab/files/u3/InfoLab_WP06-01_Governance.pdf (checked 13/08/2007)

[BUY-01] Buyya, R., Vazhkudai, S., 2001, *Compute Power Market: Towards a Market-Oriented Grid*, in Proceedings of The First IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Press

[HAUS-05] Hausheer, D., Stiller B., 2005, *PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications*, in NETWORKING 2005, pp 40,52, Springer Berlin / Heidelberg, ISBN: 978-3-540-25809-4

[HAUS-03] David Hausheer, Nicolas C. Liebau, Andreas Mauthe, Ralf Steinmetz, Burkhard Stiller, 2003, *Token-based Accounting and Distributed Pricing to Introduce Market Mechanisms in a Peer-to-Peer File Sharing Scenario*, in , In Proceedings 3rd IEEE International Conference on Peer-to-Peer Computing, Linkoping

[HUA-06] Huang, M., Baviar, A., Peterson, L., 2006, *PlanetFlow: maintaining accountability for network services*, in ACM SIGOPS Operating Systems Review, Volume 40, pp 89-94, ACM Press New York, NY, USA

[KOPP-05] Koppell, J., 2005, *Public Administration Review*, in Public Administration Review, Volume , pp 94-108, Blackwell Publishing

[KREM-02] Kremer, S., Markowitch, O., Zhou, J., 2002, *An intensive survey of fair non-repudiation protocols*, in Computer Communications, Volume 25, pp 1606-1621, Elsevier

[MAN-02] Maniaitis, P. and Baker, M., 2002, *Enabling the Archival Storage of SIgned Documents*, in Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST 2002), USENIX Association, Berkeley, CA, USA

[MARK-99] Markowitch, O., Roggeman, Y., 1999, *Probabalistic non-Repudiation without Trusted Third Party*, in Proceedings of Second COnference on Security in Communication Networks,

[MITS-01] MItsianis, J., 2001, *A New Approach to Enforcing non-Repudiation of Receipt*,

[MMAPPS-10] Gerke J.(Editor), 2004, *Specification and Implementation of the Peer-to-Peer Middleware; Deliverable 10 of the MMAPPS Project*, http://www.mmapps.org/results/WP4.1-D10-P2P-Middle-1.0.pdf

[NISS-94] Nissenbaum, H., 1994, *Computing and Accountability*, in Communications of the ACM, Volume 37, pp 72-80, ACM Press, New York

[PAR-96] Parris, K. V. C., 1996, *Implementing Accountability in Software Development*, in IEEE Software, Volume 13, pp 83-93, IEEE Press

[ROB-05] Robinson, P., Cook, N., Shrivastrava, S., 2005, *Implementing fair non-repudiable interactions with Web services*, in Proceedings of Ninth IEEE International EDOC Enterprise Computing Conference, IEEE Press

[ROWS-01] Rowstron, A., Druschel, P., 2001, *Pastry, Scalable, distributed object location and routing for large-scale peer-to-peer systems*, in Proceedings of IFIP/ACM Middleware, Springer

[SCHE-99] Schedler, A., 1999, , in he Self-Restraining State: Power and Accountability in New Democracies, pp 13-28, Lynne Reiner Pulishers, ISBN: 1-55587-773-7

[STRU-03] Strulo, B., Smith, A., Farr, J., 2003, *An Architecture for Peer-to-Peer Economies*, in , In proceedings 3rd IEEE International Conference on Peer-to-Peer Computing, Linkoping, Sweden

[VISH-03] Vishnumurthy, V., Chandrakumar, S., Sirer, E. G., 2003, *KARMA: Asecure Economic Framework for Peer-to-Peer Resources*, in Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA.,

[WANG-07] Wang, Y., Singh, M., 2007, *Formal trust Model for Multiagent Systems*, in Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'07),

[WSS] Web Services Security (WSS) TC, 2006, *Web Services Security: SOAP Message Security 1.1*, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[WST] OASIS Web Services Secure Exchange TC, 2007, *WS-Trust 1.3*, http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf

[YUM-04] Yumerefendi, A.R., Chase, J. S., 2004, *Trust but verify: accountability for network services*, in Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC, 2004, ACM Press, New York

[ZHANG-07] Zhang, Y., Lin, K.,, 2007, *Hierarchical Management of Service Accountability in Service Oriented Architectures*, in Proceedings of IEEE International Conference on Service-Oriented Computing and Applications, IEEE Press

[ZHANG-06] Zhang, Y., Lin, K., Yu, T., 2006, *Accountability in Service Oriented Architecture: Computing with Reasoning and Reputation*, in Proceedings of IEEE Conference on e-Business Engineering, IEEE Press

[ZHOU-96] Zhou J. and Gollman D., 1996, *A fair non-repudiation protocol*, in Proceedings of IEEE Symposioum on Security and Privacy, IEEE Press