

On the Feasibility of Bilaterally Agreed Accounting of Resource Consumption

Carlos Molina-Jimenez, Nick Cook, and Santosh Shrivastava

School of Computing Science, Newcastle University, UK
{Carlos.Molina,Nick.Cook,Santosh.Shrivastava}@ncl.ac.uk

Abstract. The services offered by Internet Data Centers involve the provision of storage, bandwidth and computational resources. A common business model is to charge consumers on a pay-per-use basis where they periodically pay for the resources they have consumed (as opposed to a fixed charge for service provision). The pay-per-use model raises the question of how to measure resource consumption. Currently, a widely used accounting mechanism is provider-side accounting where the provider unilaterally measures the consumer's resource consumption and presents the latter with a bill. A serious limitation of this approach is that it does not offer the consumer sufficient means of performing reasonableness checks to verify that the provider is not accidentally or maliciously overcharging. To address the problem the paper develops bilateral accounting models where both consumer and provider independently measure resource consumption, verify the equity of the accounting process and try to resolve potential conflicts emerging from the independently produced results. The paper discusses the technical issues involved in bilateral accounting.

Keywords: Service provisioning, resource accounting, unilateral resource accounting, bilateral resource accounting, storage accounting, trusted outcomes, Amazon's storage service.

1 Introduction

The focus of our research is services provided by Internet Data Centers (IDC) to remote customers over the Internet. The variety of these services is large and still growing. Leaving aside specific technical details, we consider there to be three basic services that sell storage, bandwidth and compute power to remote consumers. We are interested in pay-per-use services, as opposed to fixed charge services. In the latter, the bill is fixed irrespective of the amount of resources consumed. In the former, the bill depends on the amount of resources consumed. Pay-per-use services can be further categorised into on-demand and utility services. In the case of on-demand services, the consumer pays (normally in advance) for a fixed amount of resource (for example, 60 minutes of international phone calls) and the service is terminated when the consumer exhausts the resources. With utility services the consumer consumes as much as he needs; charges are calculated according to actual consumption and presented to the consumer at the end of an agreed upon accounting period. A well

known example from the IT field that uses the utility service model is the Amazon Simple Storage Service (Amazon's S3) [1] that sells storage space to remote users. Central to the pay-per-use model is the issue of accountability for the consumed resources: who performs the measurement and decides how much resource has been consumed — the provider, the consumer, a trusted third party (TTP), or some combination of them? Traditional utility providers such as water, gas and electricity services use provider-side accounting based on metering devices that have a certain degree of tamper-resistance and are deployed in the consumer's premises. Provider-side accounting is also widely used by phone services and Internet-based utility providers like Amazon. However, in contrast to traditional utilities, the infrastructure responsible for measuring resource consumption is deployed at the provider. The distinguishing feature of provider-side accounting is that it is unilateral. Provider-side accounting is acceptable when the consumer has good reasons to trust the provider not to accidentally or maliciously overcharge. However, we contend that there will be a class of applications, or of relationships between consumer and provider, where this assumption does not hold and where other models are needed. Unilateral consumer-side accounting can be implemented but we do not discuss it in this paper because it is not representative of practical applications — it is very unlikely that a provider would simply accept accounts unilaterally computed by a consumer. Unilateral accounting by a TTP on behalf of both consumer and provider would be more practical than consumer-side accounting, but in this paper we consider a hitherto unexplored alternative of bilateral accounting of resource consumption. We develop a new model in which the consumer and provider independently measure resource consumption, compare their outcomes and agree on a mutually trusted outcome. The problem of achieving mutual trust in accounting for resource consumption is currently neglected but is becoming important as users increasingly rely on utility (or cloud) computing for their needs. We explore the technical issues in bilateral accounting and develop a model that is abstract and general enough to apply to the different types of resources that are being offered on a utility basis. To ground our approach in current practice, we often use storage consumption as an example and, in particular, use Amazon's S3 as a case study for bilateral accounting.

Section 2 presents an overview of a generic accounting system, its components and underlying trust assumptions. Section 3 discusses our model of bilateral accounting for resource consumption and issues such as data collection and conflict resolution. In Section 4 we take Amazon's S3 storage service as a case study and discuss the feasibility of bilateral accounting of storage consumption. Section 5 presents related work. Section 6 concludes the paper with hints on future research directions.

2 Resource Accounting Services

We conceive a resource accounting system as composed out of three basic component services: metering, accounting and billing (see Fig. 1). We assume that resources are exposed as services through one or more service interfaces. As shown in the figure, the metering service collects data on resource usage at the service interfaces that it is able to access. The metering service stores the collected data for use by the accounting service. The accounting service retrieves the metering data, computes

resource consumption from the data and generates accounting data that is needed by the billing service to calculate the billing data. We assume that billing and accounting services use deterministic algorithms known to all the interested parties. Thus, given the same data, any party can compute the outcome of the relevant accounting or billing service.

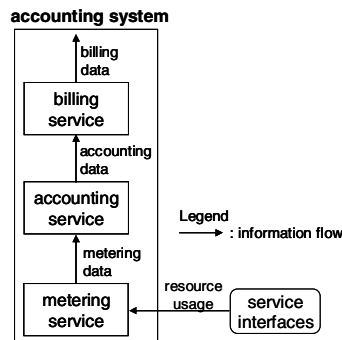


Fig. 1. Components of the resource accounting system

2.1 Trust Assumptions and Root of Trust

Regarding the trustworthiness of outcomes produced by the component services, we distinguish between **unilaterally trusted** and **mutually trusted** outcomes. A unilaterally trusted outcome is produced by a party with the help of its own component services and is not necessarily trusted by other parties. The components could be located either within or outside the party's infrastructure. In the latter case, the component's owner may need to take additional measures, such as the use of tamper-resistant mechanisms, to protect the component and its outcomes against modification by other parties [2, 3]. There are two approaches to producing a mutually trusted outcome: 1) a TTP produces the outcome using its own certified infrastructure, or 2) the parties concerned use their respective unilaterally trusted outcomes as the basis for agreement on a valid, mutually trusted outcome. This alternative is the focus of our interest and, as discussed later, requires the execution of some protocol between the participants (two in a bilateral relationship) to produce an agreed upon and non-repudiable outcome. Mutually trusted outcomes form the "root of trust" for building trusted resource accounting systems. The source of mutually trusted outcomes can be rooted at any one of the three levels shown in Fig. 1. Once a mutually trusted outcome source is available, the trustworthiness of the component services above it becomes irrelevant. Given the determinacy assumption, a party can always resort to the mutually trusted outcome to compute and verify results produced by other parties. For example, given a metering service that produces mutually trusted metering data, the accounting and billing services can be provided by any of the parties in any combination; their outcomes are verifiable by any other party.

3 Bilateral Resource Accounting

Bilateral accounting is an attractive solution in applications where mutually untrusted consumer and provider are reluctant, or unable, to use a TTP and therefore agree to deploy their own component services. A distinguishing feature of this approach is that the consumer and provider own and run their own independent but functionally equivalent component services to produce their unilaterally trusted outcomes. Bilateral agreement between the pair of component services results in the trusted outcome needed to build the whole resource accounting system. This approach presents us with two fundamental problems: (i) how do the consumer and provider collect the metering data that is essential to compute unilaterally trusted outcomes that can form the basis for agreement on resource consumption, and (ii) how do consumer and provider resolve conflicts over resource consumption.

3.1 Collection of Metering Data

As stated in Section 2, we assume that a resource user (consumer or provider) uses a resource by invoking operations at service interfaces to the resource. Consumers use a resource through one or more service interfaces exposed by the provider. In their turn, the provider accesses the resources necessary to provide the service through a set of interfaces. The consequence is that any party wishing to collect metering data is restricted to analysing operations at the service interfaces that they can access.

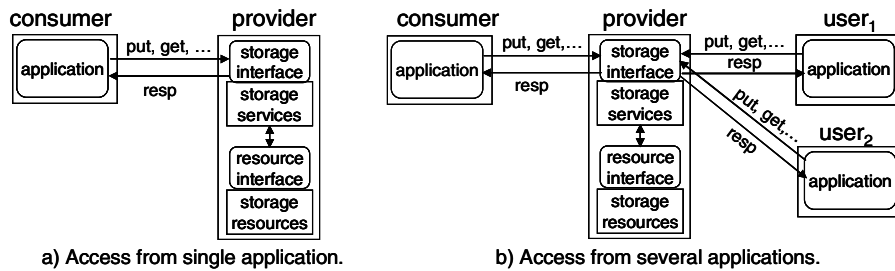


Fig. 2. Example use cases for utility storage services

For example, in Fig. 2-a, the consumer accesses the provider’s storage services through a storage interface. To generate its unilaterally trusted outcome with respect to storage consumption, the consumer must generate metering data by analysing operations at this storage service interface. In contrast, the storage provider has access both to the storage service interface it exposes to the consumer and to a resource interface for operations on its storage resources. Consequently, it can collect metering data by analysing operations at both interfaces, combine this information and produce its own unilaterally trusted outcome about the consumer’s resource consumption. For the sake of illustration, we consider two common utility storage use cases. In Fig. 2-a, a consumer hosts a single application that invokes operations on the provider’s storage service. An example of this case is the use of a storage service for backup. In Fig. 2-b, the storage is consumed by the consumer’s application and by applications

hosted by other users ($user_1$, $user_2$, etc.) that all access the storage service at the consumer's expense. An example of this case is a consumer using a storage service to provide photo or video sharing services to other users.

In the single application case, the consumer can analyse its requests to and the responses from the service over a given period. The consumer can collect metering data on the type of a request (put, get etc.), on the size of a request and of an associated response, and on the round-trip time for request/response pairs. It can compute the bandwidth used for its service invocations (for example, in GB transferred per month). Given a defined relationship between data transferred and amount of storage used (see Section 4), from request types and request/response sizes the consumer can compute storage consumption (for example, in GB stored per month). From round-trip times, it can compute average response times for its service invocations. On the other hand, the provider can compute bandwidth usage by performing essentially the same analysis as the consumer on the request/response traffic as seen by the provider. It can use the same data to compute storage consumption in the same way as the consumer. In addition, the provider is able to analyse operations at the storage resource interface. That is, the provider can compute storage consumption based on data collected at either the storage service interface or at the resource interface (which may allow a more direct measure of consumption). The choice of input data will determine the algorithm the provider uses to compute its unilaterally trusted outcome. Equally important, the choice of input data admits the possibility of a degree of divergence between the unilaterally trusted outcomes produced by consumer and provider. With respect to response time as experienced by the consumer, the provider must either: (i) compute estimates based on average network latency, or (ii) compute actual response times using data the provider collects at the consumer. The former approach can be used if the provider-side data supports computation of response times to a degree of accuracy that is acceptable to both parties. The latter approach can be used if the provider can deploy their own metering service at the consumer.

The multi-user, multi-application case shown in Fig. 2-b presents the consumer wishing to independently compute resource usage with additional problems. The applications all access the resource through the storage service interface and the consumer is accountable for all usage. However, in addition to collecting data on its own operations, the consumer must now collect data on operations performed by $user_1$ and $user_2$ (and any additional users given access). The consumer may be able to collect this data by deploying metering services with each application instance, in their own and each user's infrastructure. An advantage of this approach is that the consumer can collect data to compute bandwidth consumption, storage consumption and response times for all the applications for which it is accountable. The feasibility of the approach will depend on factors such as the degree of control the consumer has over the deployed applications and the number of users. Alternatively, the consumer may be able to deploy a metering service at the provider to collect data on user operations at the storage service interface. If this approach is adopted, the consumer can compute bandwidth and storage consumption (under the same assumptions as the single application case). They can no longer directly compute response times for any party other than themselves. As for the provider in the single application case, estimates of response times, to some agreed degree of accuracy, may be acceptable.

From the storage provider’s viewpoint, the multi-user, multi-application case is similar to the single consumer application case. The provider can compute resource consumption such as bandwidth and storage from data collected at the storage service and/or resource interfaces. The discussion of provider computation of consumer-perceived response times applies to computation of response time for other users.

The preceding discussion highlights the fact that bilateral accounting relies on the ability of the parties involved to independently collect the data necessary to produce their unilaterally trusted outcomes. Further, as noted in Section 2, data collection may require additional protection mechanisms such as tamper-resistance. We now generalise the discussion.

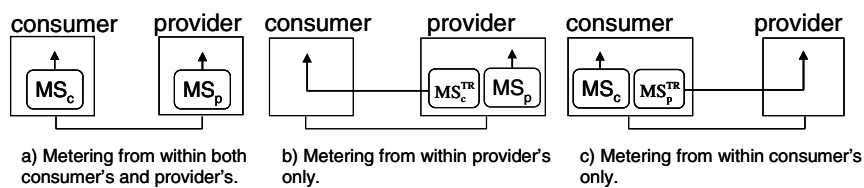


Fig. 3. Models for bilateral metering

Fig. 3 shows three possible scenarios. A provider (p) offers some service to a consumer (c). MS stands for Metering Service. MS_c is a consumer’s metering service. MS_p is a provider’s metering service. TR means tamper-resistant protection. So, MS_c^{TR} is a consumer metering service that is tamper-resistant to provider modification. The outcomes of metering services are unilaterally trusted (MS_c by c and MS_p by p) and made available to their respective accounting services (not shown). Fig. 3-a applies when metering data can be collected from within the consumer and the provider (for example, data to compute bandwidth). Fig. 3-b applies when the metering data of a required degree of accuracy can only be collected from within the provider (for example, data on access by multiple users). In this case, the provider deploys its MS_p locally, whereas the consumer performs remote metering with the help of MS_c^{TR} . Fig. 3-c mirrors Fig. 3-b: metering data of a required degree of accuracy can only be collected from within the consumer (for example, data to compute response time). In this case, MS_c performs local metering whereas the provider deploys MS_p^{TR} at the consumer.

3.2 Agreement on Mutually Trusted Accounting Outcomes

Though they are functionally equivalent, consumer and provider components do not necessarily use the same algorithms or input data to compute their unilaterally trusted outcomes. As discussed in Section 3.1, they may use data collected at different interfaces to compute an outcome. There is then the possibility of divergence between the independently computed (unilaterally trusted) outcomes. To address this problem we suggest the use of a Comparison and Conflict Resolution Protocol (CCRP). A suitable protocol will support: (i) the comparison of independently produced and unilaterally trusted outcomes to detect potential divergences; (ii) where possible, for

example, when $|Outcome_p - Outcome_c| \leq d$, (c , p , d stand for consumer, provider and agreed-upon acceptable divergence, respectively), the immediate declaration of absence of conflicts; (iii) where the divergence is greater than d , the execution of a negotiation protocol between consumer and provider with the intention of reaching agreement on a single outcome; (iv) when the negotiation protocol fails to sort out the conflict automatically, the declaration of conflict for off-line resolution; (v) production of non-repudiable mutually trusted outcome. The non-repudiation property is necessary to ensure that neither consumer nor provider can subsequently deny execution of the CCRP or the result of that execution. That is, they could not subsequently deny their agreement or otherwise to a given accounting outcome.

Our middleware for non-repudiable information sharing [4, 5] can form the basis of a CCRP. The middleware provides multi-party, non-repudiable agreement to updates to shared information which can be maintained in a distributed manner with each party holding a copy. Essentially, one party proposes a new value for the state of some information and the other parties sharing the information subject the proposed value to application-specific validation. If all parties agree to the value, then the shared view of the information is updated accordingly. Otherwise, the shared view of the information remains in the state prior to proposal of the new value. For non-repudiable agreement to a change:

1. there must be evidence that any proposed change originated at its proposer, and
2. there must be evidence that all parties agreed to any change and therefore share the same (agreed) view of information state.

That is, there must be evidence that all parties received the proposed update and that they agreed to the state change.

In our consumer-provider application the problem is for consumer and provider to reach agreement on the set of mutually trusted outcomes that will ultimately be used to compute the bill for a given accounting period. For example, if bills are computed from a set of resource usage records for a given period, the problem is to reach agreement on the valid membership of the set. Fig. 4-a shows the abstraction of a consumer and provider sharing, and adding to, a set of resource usage records; the box is shown dotted to indicate that this is a logical view; in reality each party has a local copy of the set. The non-repudiable information sharing middleware will maintain this abstraction of a set of mutually agreed records and control the addition of new records to the set. The middleware generates non-repudiation evidence to ensure that the state of the agreed set of records is binding on both parties. In the case of the provider proposing a new record, the basic two-party agreement process is:

1. The provider proposes a new record.
2. The consumer performs application specific validation of the proposed record. This validation can be arbitrarily complex. The record could be compared with an equivalent record computed locally by the consumer. Bounds on divergence between the two records could be imposed. The history of previous interactions could be used to detect any persistent over-estimation of consumption on the part of the provider.
3. The consumer returns a decision on the validity, or otherwise, of the proposed record.

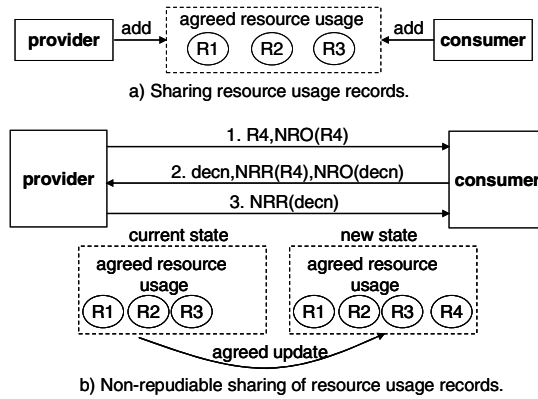


Fig. 4. Non-repudiable agreement to resource usage

The middleware uses a signed, two-phase commit protocol with application-level validation to achieve the preceding basic agreement. Fig. 4-b shows the execution of the protocol for addition of a new record, R4, to the agreed set of records {R1, R2, R3}. First, the provider proposes the addition of R4 with non-repudiation of its origin (NRO(R4)). Then, the consumer validates the record and returns a decision on its validity or otherwise (decn), non-repudiation of receipt of R4 (NRR(R4)) and non-repudiation of origin of the decision (NRO(decn)). The decision is essentially a binary yes or no value. However, the middleware supports annotation of the decision with application-specific information. For example, the decision may be annotated with the degree of divergence of the record from the consumer’s view of resource usage. These annotations form part of the evidence generated during protocol execution and may be used to support more complex negotiation built on top of the basic agreement mechanism. The protocol terminates with the provider sending non-repudiation of receipt of the validation decision to the consumer (NRR(decn)). As shown in Fig. 4-b, if the consumer decides that R4 is valid then the agreed set of records is {R1, R2, R3, R4}. Otherwise, the agreed view of the set remains unchanged ({R1, R2, R3}) and the failure to agree to the addition of R4 will be signaled to both parties. As with annotations to decisions, this failure signal can be used to build more complex negotiations. For example, the signal could trigger the proposal of a revised record or the initiation of extra-protocol dispute resolution. At the end of a protocol run, both parties have the same irrefutable (binding) view of the set of agreed records and of the validation decisions made with respect to records in the set and with respect to proposed records that have been rejected by either consumer or provider.

As discussed, the non-repudiable agreement provided by the middleware can be used as a basic building block for negotiation of a mutually trusted accounting outcome. Proposal of, and agreement to, addition of a new outcome is cheap (involving a single agreement protocol round). Disagreement to the proposed addition of an outcome to the set in one round of protocol execution can trigger further protocol rounds with revised proposals or lead to extra protocol resolution (based on evidence generated during protocol executions). The middleware supports the application-specific, autonomous imposition of conditions for validation of updates,

proposal of revised updates and termination of protocol execution. Therefore, more complex multi-round negotiation could be built on the basic agreement mechanism.

3.3 Models for Bilateral Accounting

We now complete our models for bilateral accounting by combining the component services with execution of a CCRP. Fig. 5 shows the three variants of our model when metering data is collected at both consumer and provider (as in Fig. 3-a). As before, c and p stand for consumer and provider. MS, AS and BS stand for metering service, accounting service and billing service, respectively. The difference between the three variants is the level at which the root of trust is established. This is determined by the level at which the parties execute the CCRP to agree on a mutually trusted outcome.

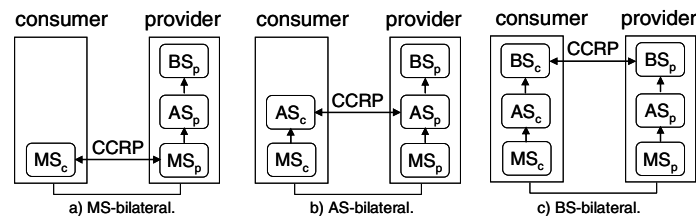


Fig. 5. Models for bilateral resource accounting

In Fig. 5-a the mutually trusted outcome is produced at the metering level, at the accounting level in Fig. 5-b and at the billing level in Fig. 5-c. In each variant, the provider deploys the component services above the level of the root of trust. As indicated in Section 2, the rationale here is that once a bilaterally trusted outcome is available, the consumer can verify any subsequent accounting computation. This verification is orthogonal to the operation of the bilateral resource accounting system. There are six further variants of the model: three for deployment of MS_c and tamper-resistant MS_p at the consumer (as in Fig. 3-c), and three for deployment of MS_p and tamper-resistant MS_c at the provider (as in Fig. 3-b). The full set of models also allows for the combination of data from mixed MS deployments to build a complete bilateral accounting system.

4 On the Feasibility of Bilateral Accounting for Amazon's S3 Storage

Amazon's S3 is currently one of the best-known services that provides storage to remote users over the Internet on a pay-per-use basis. S3 presents its users with the abstraction of buckets that contain arbitrary data objects up to 5GB in size with 2KB of metadata. Each bucket is owned by an Amazon's S3 user. A user-assigned key identifies each object in a bucket. There are both REST-style and SOAP Web service interfaces to the service. The interfaces define operations to create, list, delete and retrieve buckets and objects. There are other operations to manage object access control and metadata and to obtain limited statistics of service usage. User requests to

perform operations on the storage service are authenticated by a user signature and signed timestamp. Amazon charges S3 consumers monthly for 1) the amount of GB stored, 2) the number of *put*, *get* and *list* requests generated by the consumer, and 3) the amount of bandwidth consumed by the consumer (bytes transferred from outside Amazon into S3 and in the opposite direction). At least twice daily Amazon checks their storage resources to calculate the amount of storage occupied by a consumer's buckets and multiplies this by the amount of time elapsed since the last check. Amazon charges a flat amount for every 1000 *put* or *list* request and a flat amount for every 10000 *get* request. There is no charge for *delete* request. Download (e.g. object retrieval) is approximately twice as expensive as upload (e.g. object creation). Currently, accounting for use of Amazon's S3 is provider-side. Amazon meters the consumer's consumption, calculates the charges and presents the consumer with a bill. According to the definitions in Section 2.1, Amazon's charges are based on unilaterally trusted outcomes. Ideally, resource consumers should have mechanisms to independently measure their resource consumption and verify that they are not accidentally or maliciously overcharged. This would result in a bilateral accounting system.

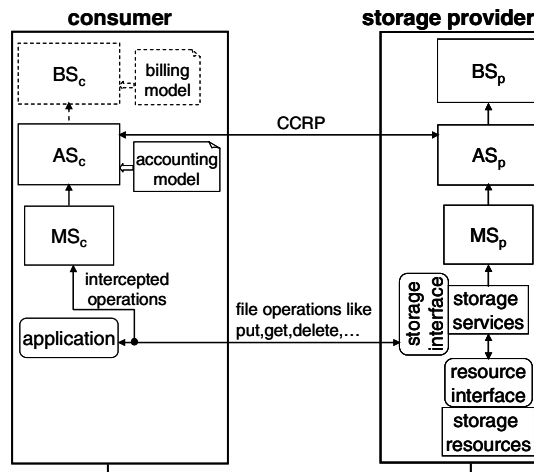


Fig. 6. Model for bilateral accounting of resource consumption in Amazon's S3

For the specific example of accounting for S3 storage consumed by an application deployed within the consumer's infrastructure (as in Fig. 2-a), we can envisage a bilateral accounting system based on the abstract model illustrated in Fig. 6. It is relatively easy to bilaterally meter the number and type of requests and bandwidth consumption. As discussed in Section 3, both consumer and provider can collect the necessary data at the storage service interface. In the case of Amazon, MS_c and MS_p can independently intercept a consumer's requests and provide the metering data to AS_c and AS_p . Given that MS_c and MS_p are intercepting the same request/response traffic, the two accounting services should arrive at the same result. Thus, it is straightforward to bilaterally account for two elements of Amazon's charging model:

numbers of requests and bandwidth. A more challenging problem is to ensure that AS_c and AS_p produce similar accounting data about storage consumption over time (the third element of Amazon's charging model). The difficulty with storage consumption is that the consumer does not have the same degree of access to the storage resources as Amazon. Amazon, as storage provider, accesses the resources through their resource interface. This allows Amazon to more directly measure resource consumption. Amazon's S3 does provide operations to list storage information, to obtain service statistics and perform some user-accessible logging. However, Amazon explicitly states that this information cannot be relied on for checking consumer's storage accounting. In any case, these operations cannot be used by MS_c as an independent means of collecting metering data about storage consumption. As suggested in Section 3.1, this leads to different strategies for collection of metering data by MS_c and MS_p . For example, MS_c can independently meter the size of upload requests to Amazon. AS_c can then use such consumer-side data to estimate storage consumption. However, it is very likely that the results produced by AS_c will diverge from those produced by AS_p because AS_p is able to rely on data collected by MS_p which has access to the resource interface within Amazon. There is clearly some relationship between the request size that a consumer can measure and the storage resource usage (which may for example include file meta data) that Amazon is able to measure more directly. Unfortunately, Amazon's published charging model does not define this relationship. Furthermore, the charging model does not provide sufficient information to the consumer about the time from which storage usage is charged. Thus, when computing the impact of a request to create or delete an object or bucket, it is unlikely that the consumer AS_c will arrive at the same result as the provider AS_p . In summary, the only independent metering data available to the consumer is based on their request traffic. The published charging does not provide them with sufficient information to use that data to perform their own storage accounting and produce results that will be compatible with those produced by Amazon. Bilateral accounting is currently possible for two elements of the Amazon's S3 charging model (number of request and bandwidth usage) but not for the third (storage usage over time).

A solution to the preceding problem is that the provider publish a reference model that allows the consumer to estimate with an agreed upon accuracy the impact of their requests on storage usage. In Fig. 6, this idea is represented by the provider-supplied accounting model that the consumer's accounting service uses to generate accounting data. For example, an Amazon's S3 accounting model for use by consumers would stipulate how to calculate the impact on storage usage of a put request given the time and size of the request. This would include the means to estimate the size of any storage metadata and the time from which Amazon would charge storage. Such a model is necessary but, given they are using different input data, AS_c and AS_p may still produce divergent unilaterally trusted outcomes. Therefore, as shown, in this case AS_c and AS_p would execute the CCRP to arrive at a mutually trusted outcome that becomes the root of trust for billing. In principle, the root of trust could be produced at the metering services or billing services level. However, in this example, reaching agreement at the metering services level is difficult because there is a significant difference between the data on storage usage collected by the consumer and provider. It is only when AS_c and AS_p apply the accounting model that outcomes will converge.

The availability of a mutually trusted outcome at the accounting level makes bilateral billing redundant. This is why we represented consumer's BS_c in dashed lines. The argument here is that the consumer can always retrieve the bilaterally trusted accounting outcome, input it into a billing service that implements the billing model and verify the bill.

5 Related Work

The topic of bilateral monitoring of resource consumption has received little attention as an object of study in its own right. In this section we will briefly mention some results that are somehow related to our work. General architectural concepts involved in monitoring service level agreements are discussed in [6]. The emphasis there is in providing the two parties with mechanisms for monitoring each other's behaviour. In [7], the authors introduce the concept of monitorability of service level agreement parameters and point out that some parameters are unmonitorable, monitorable by the consumer, monitorable by the provider or monitorable by a TTP. This relates directly to the discussion in Section 3.1 on the collection of data at the set service interfaces available to a given party. Regarding models for resource accounting, in [8] authors describe a reference model for building accounting services based on accounting policies. The layers of this model correspond to the component services of our work, however, the focus of the authors is only on unilateral accounting. To the best of our knowledge, Saksha [9] is one of the first attempts to implement bilateral accounting of storage consumption. The paper discusses only scenarios with a single application deployed within the consumer like in Fig. 2-a. The authors do not explicitly describe their trust model which appears to be a combination of what we call consumer-side and provide-side metering, where the consumer and the storage provider unilaterally measure the amount of storage consumed and freed, respectively. The merit of this work is that it raises several issues that need to be studied. Our decomposition into component services is similar to that suggested in [10] where the issue of storage accounting in Grid environments is addressed; the interest of the author is in unilateral accounting, whereas in our work, the focus is on bilateral. A general discussion on the use of strong accountability from the perspective of liability for actions (e.g., deletion of files, retrieval of the latest version, etc.) executed between a client and their storage service provider is presented in [11]; however, the issue of storage consumption accounting is not discussed. The need for bilateral accountability of storage consumption is hinted in [12] where a storage provider charges for executing their consumer's operations such as create, replace, append, delete, find and list file, etc, depending on the size of the file. The paper focuses on the payment mechanisms and overlooks the issue about the computation of the size of the file which seems to be unilaterally decided by the consumer. Comprehensive discussions of Amazon's storage and an application hosting services is presented in [13, 14]. Bilateral accounting of storage consumption is not mentioned, yet related parameters such as throughput, availability, consistency, etc. are discussed. Our work is related to research on the financial impact of IT solutions. In [15] for example, the authors study analytical models for resource allocation and job acceptance in a Web service hosting enterprise, that allow the enterprise to maximize its revenue. Bilateral accounting

opens up the possibility of empowering consumers to minimize their expenditure on IT resources. We believe that a consumer can take advantage of their metering data to infer expenditure pattern and tune the application to minimise the bill; a step further would be for the consumer to analyse accounting models offered by different providers with the intention of dynamically re-locating the application to the provider whose accounting model would minimize the expected resource consumption charges.

6 Concluding Remarks

Users are increasingly relying on pay-per-use services from utility (or cloud) computing service providers. Charging is on the basis of the provider unilaterally measuring the consumer's resource consumption and producing the accounting data. We believe that this practice of provider-side accounting will need to be supplemented by measures that enable consumers to produce their own accounting data, minimally to check the reasonableness of the provider produced data. Bilateral accounting of resource consumption is the next logical step: the consumer and the provider independently measure resource consumption, compare their outcomes and agree on a mutually trusted outcome. We developed a number of models of bilateral accounting and used Amazon's S3 storage services as a case study to highlight the issues involved. Success of bilateral accounting to a large extent will depend on two factors: the quality of accounting data consumers can collect and the availability of a relatively simple comparison and conflict resolution protocol (CCRP) to enable production of mutually agreed outcomes. Service providers can help consumers by providing (i) suitable service interfaces to enable consumer side metering, and (ii) a reference model (e.g., an accounting model) to enable consumers to estimate resource consumption and associated charges. Further, as we discussed, sometimes there is also a need for a consumer (provider) to collect metering data directly at the provider's (consumer's) premises, so suitable metering techniques will need to be developed. CCRP procedures will also need to be developed and agreed as a part of the service level agreement.

Acknowledgements

This work has been funded in part by UK Engineering and Physical Sciences Research Council (EPSRC), Platform Grant No. EP/D037743/1, "Networked Computing in Inter-organisation Settings".

References

1. Amazon Simple Storage Service (Amazon S3), <http://aws.amazon.com/s3>
2. van Oorschot, P.C.: Revisiting Software Protection. In: Boyd, C., Mao, W. (eds.) *ISC 2003*. LNCS, vol. 2851, pp. 1–13. Springer, Heidelberg (2003)
3. TCG Specification Architecture Overview. Specification Revision-1.4, <http://www.trustedcomputinggroup.org/>

4. Cook, N., Shrivastava, S.K., Wheeler, S.: Distributed Object Middleware to Support Dependable Information Sharing Between Organisations. In: IEEE International Conference on Dependable Systems and Networks (DSN 2002), Washington DC, pp. 249–258. IEEE Computer Society, Los Alamitos (2002)
5. Cook, N., Robinson, P., Shrivastava, S.K.: Design and Implementation of Web Services Middleware to Support Fair Non-repudiable Interactions. *Int. J. Cooperative Information Systems (IJCIS) Special Issue on Enterprise Distributed Computing* 15(4), 565–597 (2006)
6. Molina-Jimenez, C., Shrivastava, S.K., Crowcroft, J., Gevros, P.: On the Monitoring of Service Level Agreements. In: First IEEE International Workshop on Electronic Contracting, San Diego, pp. 1–8. IEEE Computer Society, Los Alamitos (2004)
7. Skene, J., Skene, A., Crampton, J., Emmerich, W.: The monitorability of service-level agreements for application-service provision. In: Sixth International Workshop on Software and Performance (WOSP 2007), Buenos Aires, Argentina, pp. 3–14. ACM, New York (2007)
8. Zseby, T., Zander, S., Carle, G.: Policy-Based Accounting, IETF RFC 3334 (October 2002), <http://www.ietf.org/rfc/rfc3334.txt>
9. Kher, V., Kim, Y.: Building Trust in Storage Outsourcing: Secure Accounting of Utility Storage. In: 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007), Beijing, pp. 55–64. IEEE Computer Society, Los Alamitos (2007)
10. Scibilia, F.: Accounting of Storage Resources in gLite Based Infrastructures. In: 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007), Paris, pp. 273–278. IEEE Computer Society, Los Alamitos (2007)
11. Yumerefendi, A.R., Chase, J.S.: Strong Accountability for Network Storage. *ACM Trans. on Storage* 3(3) (2007)
12. Ioannidis, J., Ioannidis, S., Keromytis, A.D., Prevelakis, V.: Fileteller: Paying and Getting Paid for File Storage. In: Blaze, M. (ed.) *FC 2002*. LNCS, vol. 2357, pp. 282–289. Springer, Heidelberg (2003)
13. Garfinkel, L.S.: An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS. Technical Report TR-08-07; Center for Research on Computing and Society, School of Engineering and Applied Science, Harvard University (2007)
14. Brantner, M., Florescu, D., Graf, D., Kossmann, D., Kraska, T.: Building a Database on S3. In: Annual ACM Conference (SIGMOD 2008), Vancouver, pp. 251–263. ACM, New York (2008)
15. Mazzucco, M., Mitrani, I., Palmer, J., Fisher, M., McKee, P.: Web Service Hosting and Revenue Maximization. In: Proceedings of the Fifth IEEE European Conference on Web Services (ECOWS 2007), Halle, Saale, Germany, pp. 45–54. IEEE Computer Society, Los Alamitos (2007)