Diameter of Things (DoT): A Protocol for Real-Time Telemetry of IoT Applications

Soheil Qanbari[1(✉)], Samira Mahdizadeh[1], Rabee Rahimzadeh[2], Negar Behinaein[2], and Schahram Dustdar[1]

[1] Vienna University of Technology, Vienna, Austria
{qanbari,dustdar}@dsg.tuwien.ac.at, e1329639@student.tuwien.ac.at
http://dsg.tuwien.ac.at
[2] Baha'i Institute for Higher Education (BIHE), Tehran, Iran
{rabee.rahimzadeh,negar.behinaein}@bihe.org
http://www.bihe.org

Abstract. The Diameter of Things (DoT) protocol is intended to provide a near real-time metering framework for IoT applications in resource-constraint gateways. Respecting resource capacity constraints on edge devices establishes a firm requirement for a lightweight protocol in support of fine-grained telemetry of IoT deployment units. Such metering capability is needed when lack of resources among competing applications dictates our schedule and credit allocation. In response to these findings, the authors offer the DoT protocol that can be incorporated to implement real-time metering of IoT services for prepaid subscribers as well as Pay-per-use economic models. The DoT employs mechanisms to handle the IoT composite application resource usage units consumed/charged against a single user balance. Such charging methods come in two models of time-based and event-based patterns. The former is used for scenarios where the charged units are continuously consumed while the latter is typically used when units are implicit invocation events. The DoT-enabled platform performs a chained metering transaction on a graph of dependent IoT microservices, collects the emitted usage data, then generates billable artifacts from the chain of metering tokens. Finally it permits micropayments to take place in parallel.

Keywords: Diameter protocol · Service metering · Internet of things

1 Introduction

Utility computing [1] is an evolving facet of ubiquitous computing that aims to converge with emerging Internet of Things (IoT) infrastructure and applications for sensor-equipped edge devices. The agility and flexibility to quickly provision IoT services on such gateways requires an awareness of how underlying resources as well as the IoT applications are being utilized as metered services.

J. Altmann et al. (Eds.): GECON 2015, LNCS 9512, pp. 207–222, 2016.
DOI: 10.1007/978-3-319-43177-2_14

Such awareness mechanisms enable IoT platforms to adjust the resource leveling to not exceed the elasticity constraints such that stringent QoS is achievable.

The quest for telemetry of the client's IoT application resource usage becomes more challenging when the job is deployed and processed in a constrained environment. Such applications collect data via sensors and control actuators for more utilization in home automation, industrial control systems, smart cities and other IoT deployments. In this context, telemetry enables a Pay-per-use or utility-based pricing model through metered data to achieve more financial transparency for resource-constrained applications.

Metering measures rates of resource utilization via metrics, such as number of application invocations, data storage or memory usage consumed by the IoT service subscribers. Metrics are statistical units that indicate how consumption is measured and priced. Furthermore, metering is the process of measuring and recording the usage of an entire IoT application topology, individual parts of the topology, or specific services, tasks and resources. From the provider view, the metering mechanisms for service usage differ widely, due to their offerings which are influenced by their IoT business models. Such mechanisms range from usage over time, invocation-basis to subscription models. Thus, IoT application providers are encouraged to offer reasonable pricing models to monetize the corresponding metering model.

To fulfill such requirements, we have incorporated and extended the Diameter base protocol defined in RFC6733[1] to an IoT domain. There are several established Diameter base protocol applications like Mobile IPv4 [2], Credit-Control [3] and Session Initiation Protocol (SIP) [4] applications. However, none of them completely conforms to the IoT application metering models.

The current accounting models specified in the Diameter base are not sufficient for real-time resource usage control, where credit allocation is to be determined prior to and after the service invocation. In this sense, the existing Diameter base applications do not provide dynamic metering policy enforcement in resource and credit allocations for prepaid IoT users. Diameter extensibility allows us to define any protocol above the Diameter base protocol. Along with this idea, in order to support real-time metering, credit control and resource allocation, we have extended the Diameter to Diameter of Things (DoT) protocol by adding four new types of servers in the AAA infrastructure: DoT provisioning server, DoT resource control server, DoT metering server and DoT payment server. Further details regarding the aforementioned entities will be discussed in a later section of DoT architecture models. In summary, our main contribution is the specification of an extended Diameter base protocol to an IoT application domain. This contributes to fully implementing a real-time policy-based telemetry and resource control for a variety of IoT applications. Our protocol supports both the prepaid and cloud pay-per-use economic models. However, in this paper, we address the metering for the prepaid model.

With this motivation in mind, the paper continues in Sect. 2, with a brief review on how the Diameter base protocol functions. Next, we introduce the

[1] https://tools.ietf.org/html/rfc6733.

terms and preliminaries defined in this study at Sect. 3. With some definitive clues on the Diameter architecture, we propose Diameter of Things (DoT), an extension to the Diameter on how IoT applications are to be metered and monetized. The DoT framework layered architecture together with its interacting entities are detailed in Sect. 4. In support of our model, we have defined a DoT-based IoT application topology and its associated hybrid metering policies in Sect. 5. This lets the application telemetry policies vary independently from clients as well as applications that use it. To enable this, DoT performs several interrogations which are detailed in Sect. 6. DoT commands are then described in Sect. 7. Next, in Sect. 8, we express formally the DoT application transaction models to achieve better telemetry control over computing resources. Subsequently, Sect. 9 surveys related work. Finally, Sect. 10 concludes the paper and presents an outlook on future research directions.

2 The Utility of Diameter

The Remote Authentication Dial In User Service (RADIUS) [5] as per RFC2865 is a simple but most deployed protocol which provides network access control using client/server Authentication, Authorization and Accounting (AAA) model. The IETF[2] has standardized the Diameter base protocol [6] as an enhanced version of RADIUS providing a flexible peer-to-peer operation model. It is featuring intermediary agents (Relay, Proxy, Redirect and Translation) and capabilities negotiation among servers. It enables reliable transport layer using TCP or SCTP connection. The Diameter peer connections are ensured using a Keep-alive mechanism. It also supports the dynamic peer discovery and configurations in a valid session. Such specifications are achieved using set of commands and Attribute-Value-Pairs (AVPs) by collaborating and negotiating peers. Diameter has the concept of "applications", which is entirely missing in RADIUS. The protocol is enriched with a globally unique application ID. These applications benefit from the general capabilities of the Diameter base protocol while defining their own extensions on top of the base.

Some of the main features offered by Diameter are dynamic routing based on Realm, session management, accounting, agent support. It is based on Peer-To-Peer architecture as illustrated in Fig. 1, in a way that each Diameter node can behave as either a client or server based on the current deployment model. Diameter nodes can be of the type of Diameter client, Diameter server and Diameter agent. Diameter client is the node that receives the user connection request. Diameter server is the one serving the request e.g. performing user authentication based on provided information. Diameter agents themselves divide into four types of Relay, Proxy, Redirect and Translate agents. A relay agent is used to forward messages to other Diameter nodes based on the information provided in the message. Proxy agent can also act like a relay agent with the extra functionality of policy enforcement implementation via message content modification. Redirect

[2] Internet Engineering Task Force (https://www.ietf.org).

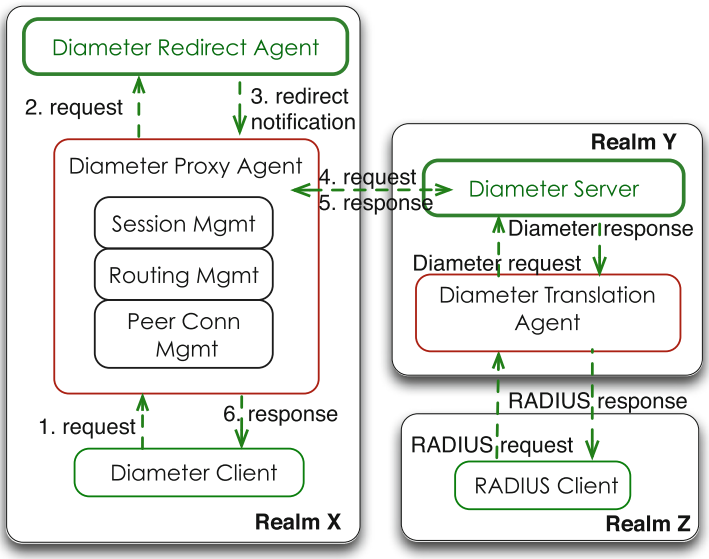

Fig. 1. Diameter base protocol architecture

agents act as a centralized configuration repository by returning information necessary for Diameter agents to communicate directly with another node. Translation agents provide translation between two distinct AAA protocols.

3 DoT Preliminaries and Terms

In this section we present basic conventions, terms together with their definitions considered in the DoT protocol:

- **Diameter of Things (DoT)**: DoT implements a mechanism to provision IoT deployment units, control resource elasticity, meter usage, and charge the user credit for the rendered IoT applications.
- **IoT Microservice**: A fine-grained atomic task performed by an IoT service on a device.
- **IoT Application Topology**: It contains the composition of hybrid collaborating IoT microservices to meet the user's request. The topology is packages with the elasticity requirements and constraints (hardware or software) which will dictate our schedule and credit allocation within the runtime environment.
- **Metering Server**: A DoT metering server performs real-time metering and rating of IoT applications deployment.
- **Metering Agent**: The agent transfers the metered values to the metering server via tiny tokens.
- **Provisioning Server**: Provisioning server refers to initial configuration, deployment and management of IoT applications for subscribers. It also deals with ensuring the underlying IoT device layer is available to serve.

- **Payment Server**: The micropayment transaction charges subscribers upon relatively small amounts for a unit of usage. It basically transfers a certain amount of trade in the payWord or microMint micropayment schemes [7]. In the payWord scheme a payment order consists of two parts, a digitally signed payment authority and a separate payment token which determines the amount. A chained hash function, is used to authenticate the token. The server then calculates a chain of payment tokens or paychain. Payments are made by revealing successive paychain tokens.
- **Rating**: The process of giving price to an IoT application usage events. This applies to service usage as well as underlying resource usage.
- **Resource-Control Server**: Resource control server implements a mechanism that interacts in real-time with a resource and credit allocation to an account as well as the IoT application. It controls the charges related to the specific IoT application usage.
- **One-Cycle Event**: It indicates a single-request-response message exchange pattern which one specific service is invoked by one consumer at a time while no session state is maintained. One message is exchanged in each direction between requesting and responding DoT nodes.

4 DoT Architecture Models

Figure 2 illustrates a schematic view on collaborating components of our proposed DoT architecture. It contains of a DoT client, Provisioning server, Resource control server, Metering server, and a Payment server.

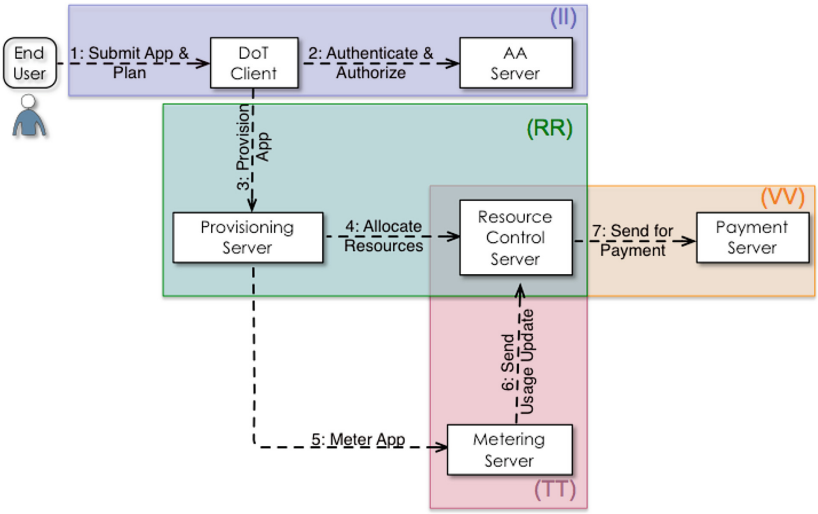

Fig. 2. Typical diameter of things (DoT) application architecture.

As the end user defines and composes an IoT application, the request is forwarded to the DoT client. The DoT client submits the composed application to the DoT infrastructure which determines possible charges, verifies user accounts, controls the resource allocation to the application, meters the usage, and finally generates a bill and deducts the corresponding credit from the end user's account balance.

DoT client contacts the AAA server with the AA protocol to authenticate and authorize the end user. When the end user submits the IoT application topology graph, the DoT client contacts the provisioning server to submit an application topology. Afterwards, the provisioning server contacts the resource control server with information of required resource units. The resource control server reserves the resources that need to be allocated to the service. When user's credit is locked for the application provisioning, the DoT client receives the grant resource message and informs the end user that the request has been granted. As soon as the IoT application is deployed and instantiated, the submitted topology is registered to the metering server for telemetry and credit control purposes.

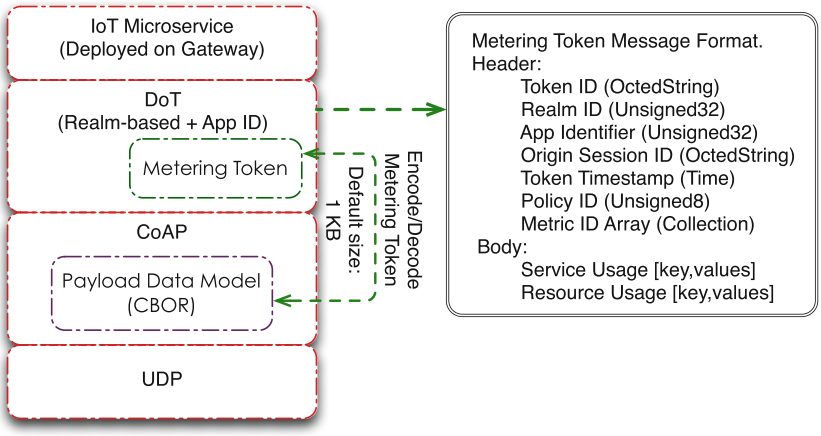

Fig. 3. Typical DoT metering token structure.

The metering server is responsible to perform the metering transaction according to the submitted topology and meter the services by calling metering the tasks of each service in the chain. Metering transactions will remain running until the termination request is sent from DoT client to the provisioning server. After receiving a termination request, the resource control server releases the resource and sends the billable artifacts related to the user usage to the payment server. The payment server, then, invokes the payment transaction and deducts credit from the end user's account and refunds unused reserved credit to the user's account.

In DoT application architecture, metered values are transfered via tokens. The metering token message attributes as shown in Fig. 3 must be supported

by all DoT implementations that conform to this specification. The CBOR[3] message format is considered for the metering tokens transmission as it can decrease payload size compared to other data formats.

5 DoT-Based IoT Application Overview

The DoT application defined in this specification implements flexible metering policy as well as the definition and constraints of the application topology.

5.1 DoT-Based Application Topology

The main responsibility of the provisioning server is the actual provision of the requested IoT application package. It contains the composition of collaborating IoT microservices to meet the user's request. Each IoT microservice is predefined with a detailed specification such as its ID, constraints and various usage patterns and policies. For instance, as defined in Listing 1.1 they can be advertised with diverse pricing models due to the event-based or time-based patterns for specific subscribers. The IoT microservices elasticity requirements and constraints (hardware or software resources) are also defined in the topology which will dictate our schedule and credit allocation within the runtime environment. The end user's request can be received in the form of a JSON object. It contains user information as well as the user requirements in terms of IoT composite application topology and its specification, to realize the intended behavior. After receiving the request, a provisioning server generates a dependency graph of the application topology complying with its specification.

The Dependency graph displays dependencies between different microservices which are requested to be in topology. In the DoT protocol, this dependency graph is used in forming the transaction model for metering the IoT deployment unit. The dependency graph is a directional graph where each node of the graph represents an available microservice in the service package registry. Similarly, each edge of the graph shows dependencies between two microservices (two nodes). The edge has a direction that shows the execution order of microservices involved in this edge. Additionally, each edge has a label which shows the policy to be in effect for this connection. The Resource control server realizes such metering policies using a predefined incident matrix. This incident matrix represents the metering policies for our directed acyclic graph (DAG) of IoT services. The metering policy incident matrix P_t is a $n * m$ matrix of P_{ij} policies, where n is the number of nodes (vertices) and m is number of lines (Edges). In the cell of N_i and V_j, the P_{ij} indicates the rate of call per granted unit (time &events). It enables each service to invoke its neighbor with the attached policy. Therefore, when a client sends a request containing a tailored IoT application topology, the Resource control server is able to rate the request based on the enforced metering policies of time (duration) and event-based usage patterns.

[3] The Concise Binary Object Representation: http://cbor.io.

5.2 DoT-Based Metering Plans

The metering plans define the allocation mechanism for granting the required resource units to an IoT application/constituent microservices. It is an indication that the following assumptions underlying our IoT telemetry solution has been considered for the proper positioning of DoT protocol. The IoT applications are advertised with an associated charging plans. In case of cloud-based model, there may be a subscription fee for pay-per-use plans. The cost of obtaining such plans is known as the plan's "premium" which is the price that is calculated and offered in the subscription phase by the provider. The estimation of the plan's premium is out of the scope of the DoT protocol. The plan indicates the composed services pricing schema and comes in two models of predefined as well as customized. The plan will be built to be consistent with the composed application topology in the rate setting.

The subscribed metering plan indicates: (i) the price of granted units for every IoT application constituent microservice. For instance, 5 h for Humidity sensor and 10 invocations for Chiller On/Off actuator. (ii) the resource Used Unit Update (U3) frequency for all associated units which are defined in the provider's plan. (iii) the manual/automatic payment configuration. So that the provider can handle the payment transactions automatically while informing the user. (iv) container instance fee, which is the fee that user pays for underlying resource usage. Instance usage can be time based or underlying resource-usage based which is defined by the provider's policy. (v) finally, subscription time for pay-per-use model and subscription fee for prepaid model.

Listing 1.1. Excerpt of an IoT application spec and policy in a JSON object.

```
{   "microserviceID":"01",
    "microserviceName":"getTemperature",
    "uri":"getTemperature.py",
    "execute":"python getTemperature.py",
    "constraints":{"runtime": "python 2.7","memory": "..."}
    "policies":[{"policyID":"PL_01_01",
                 "cost":"$2/week",
                 "desc":"time mode - $2 per week"},
                {"policyID":"PL_01_02",
                 "cost":"0.01 cent/invoke",
                 "desc":"event mode - 0.01 cent per invoke"},
                {"policyID":"PL_01_03",
                 "cost":"$1",
                 "desc":"subscription fee"}
                ] }
```

6 DoT Interrogations

For a Hybrid DoT, four main interrogations are performed for a well-functioning protocol. The first interrogation is called Initial Identification (II) which

basically deals with clients' identification, for instance the user authentication and authorization processes. The second interrogation is Request Realization (RR) that aims to provision the clients' IoT applications as well as scheduling their resource allocation upon agreed terms and subscribed plan. The third interrogation is called Telemetry Transmission (TT) that deals with metering the running services as well as the granted units usage data transmission for charging purposes. The final interrogation, entitled Value Verification (VV), ensures value generation and delivery to the interested stakeholders. The hybrid metering is carried out in main DoT sessions which hold globally unique and constant Session-IDs. The whole DoT-based metering life-cycle including the II, RR, TT, and VV interrogations are presented in Fig. 4.

6.1 Initial Identification (II)

The end user subscribes the application as well as the chosen plan to the DoT client. The DoT client submits the IoT deployment units to the Provisioning server to ask for the required resource units it needs to run. In this case, the Provisioning server queries for resources (including underlying resources and credit allocation) from Resource control server. The Resource control server is responsible for the device resource reservation. It also keeps track of user credit fluctuations.

In this phase, the end user requirements are modeled into an application topology using a directed acyclic graph. This graph can connect various IoT microservices available in diverse usage units. The deployment of such hybrid applications will result in one global constant Session-ID followed by related subsession-IDs as well as transaction-IDs. Note that the IoT application might send a (re)authorization request to the AAA server to establish or maintain a valid DoT session. However, this process does not influence the credit allocation that is streaming between the DoT client and provisioning server, as it has already been authorized for the whole transaction before.

6.2 Request Realization (RR)

The Resource control server analyzes the IoT service and allocates resources to the requested service. It also considers the subscription time/fee in order to notify user when this value elapses.

When the new usage update arrives at the Resource control server, it charges the credit based on the usage summary received from the Metering server. It also validates subscription by checking the status of credit (as in prepaid model) or elapsed time (in pay-per-use model) against a certain threshold. Upon reaching the threshold, the Resource control server sends an update notification request to the user. DoT protocol makes it possible to define a default action for this purpose. This default action can be to perform update automatically or to ask user to take the desired action.

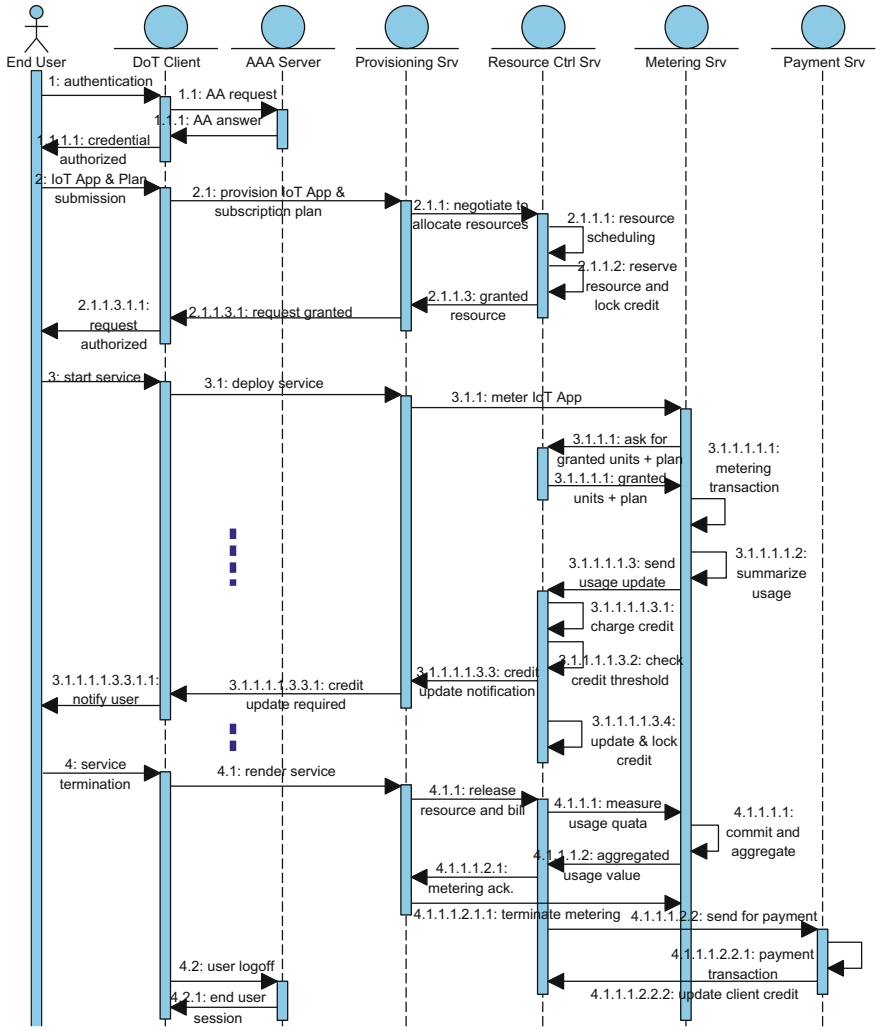

Fig. 4. The sequence of DoT interrogations to enable hybrid metering.

6.3 Telemetry Transmission (TT)

As soon as the end user sends the Start service request to the DoT client, the DoT Client asks the Provisioning server to initiate the service and start monitoring and metering processes. In this regard, the Provisioning server submits the IoT application to the Metering server and asks to establish a metering mechanism for the newly opened session.

Having an IoT application deployed, the metering server monitors the real usage of each service element including service usage and resource usage at a certain frequency rate called Unit Usage Update (U3). The U3 frequency determines

the rate of sending updates regarding unit usage of each service. It is independent of the type of service. For example, the $U3$ set to 25 %, implies that for a time-based microservice with granted units of 100 min, the usage update should be provided every 25 min; and for an event-based microservice with granted units of 100 invocations, the usage update will be sent after every 25 invocations.

To make it more clear, after identification of an IoT application to the Metering server, the Metering server sends a request to the Resource control server asking for the amount of granted units as well as the plan, which includes the U3 value for each microservice as defined by the provider. The U3 values are then provided to the Metering agents, as the metering server starts the metering transaction.

During deployment of an IoT application, each Metering agent meters the actual resource and service usage of its associated/assigned microservice. If the actual service usage value reaches an integer multiples of $U3$, the Metering agent will send a notification message to the Metering server. The Metering server then uses these feedbacks to gain a realistic perception of the usage of each microservice and to charge the user credit accordingly. Next, if the application usage of a microservice reaches the threshold value, the Metering Server informs the Resource control server issuing a resource-update request for the service. For instance, when the actual usage of a certain microservice reaches a certain threshold, e.g. more than 70 %, metering server informs the resource control server. This contributes to a continuous and consistent service delivery. The detailed flow of TT phase is presented in Fig. 5.

In the DoT credit allocation model, the provisioning server asks the resource control server to reserve resources and to lock a suitable amount of the user's credit. Then it returns the corresponding amount of credit resources in the form of service specific usage units (e.g., number of invocations, duration) to be metered. The granted and allocated unit type(s) should not be changed during an ongoing DoT session.

6.4 Value Verification (VV)

When the end user terminates the service session, the DoT client must send a final service rendering request to the Provisioning server. The Provisioning server should ask the Resource Control server to release all the allocated resources to an IoT application and perform payment transaction. As such, the Resource control server deallocates the granted resources and asks the Metering server to commit the measured metering tokens and report the quota value to it. Then the Resource Control server sends the billable artifacts to the Payment Server to charge the client account respectively. Finally, the Payment server sends the updated client credit to the Resource control server. Meanwhile, the DoT Client drops the user session via AAA server.

Upon each deduction from user credit, the DoT protocol verifies the credit value. As soon as the credit value drops below a certain threshold, it informs the user to perform credit update automatically or to take the desired action manually.

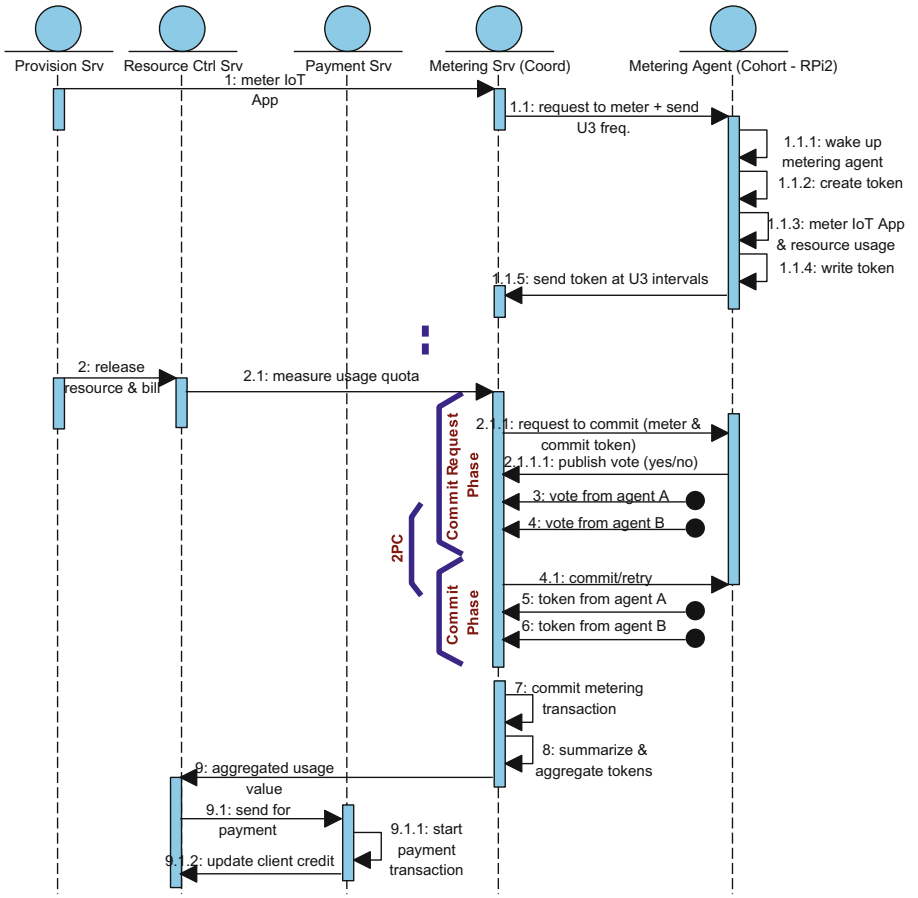

Fig. 5. DoT hybrid metering - 2PC transaction model.

7 DoT Command Messages

The DoT messages contain commands and Attribute Value Pairs (AVP) to enable metering and payment transactions for DoT-based applications. The messaging structure should be supported by all the collaborating peers in the domain architecture. Four main commands are explained here in more details.

7.1 Provision-Application-Topology-Request/Answer

The SATR command is a message between DoT client and Provisioning server. Via this command, the DoT client submits the IoT App (defined by the Client) to the Provisioning server and inquiry resources needed for the application delivery in II or RR interrogations. Following the request, the PATA response with an acknowledgment of resources reserved for the client's IoT App request.

For instance, the PATR message format is: [<Session-Id>, <Client-Id>, <Request-action>, <Dest-Realm>, <User-Name>, <IoT-App-Id>, <AVPs>]. In return, the PATA response will contain the <Granted-Service-Units> attribute in addition to the original request.

7.2 App-Resource-Allocation-Request/Answer

The CAMR command is used in RR interrogation. As soon as the Provisioning server receives Terminate-IoT-App-Request command, it sends a Commit-App-Metering-Request command message to the Metering server asking resource usage quota measurement for a specific user. Another use case is to ask for resource usage during service delivery. In return, the Commit-App-Metering-Answer command is used to report the measured quota value for the requested user and the metered IoT application.

7.3 Commit-App-Metering-Request/Answer

The SMR command is used in RR interrogation. As soon as the Provisioning server receives Terminate-Service-Request command, it sends a Start-Metering-Request command message to the Metering server asking resource usage quota measurement for a specific user. Another use case is to ask for resource usage during service delivery. In return, the Start-Metering-Answer command is used to report the measured quota value for the requested user and a metered IoT application.

7.4 Start-Bill-Payment-Request/Answer

The Start-Bill-Payment-Request command which should be invoked in VV interrogation, is used between the Resource control server and the Payment server to establish a payment mechanism and initiate a fund transfer. In response to the request, the Start-Bill-Payment-Answer command contains the state of payment transaction and the updated user credit information.

8 DoT Transaction Model

The runtime of a DoT application is carried out in four "nested chained" transactions. In this respect, the DoT transaction model preserves consistency by defining conflict serializability. In order to prevent the conflict between operations in the DoT transaction model, a schedule pattern is defined to force the logical temporal order in transactions execution. In this case, recoverability is an argument in favor of transaction processing with a rollback mechanism. In the DoT model there exists four transactions prone to inconsistency: (T1) Identification, (T2) Provisioning, (T3) Metering and (T4) Payment transaction. The Identification transaction embeds two sub-transactions of Authorization and Authentication. As such, the Payment transaction has also two sub-transactions

Table 1. The scheduled chronological execution sequence of DoT transactions.

T1: Identification	T2: Provisioning	T3: Metering	T4: Payment
:	:	:	:
Lock-S (credit)	Lock-X (credit)	Lock-X (credit)	Lock-X (credit)
Read (credit)	Read (credit)	Read (credit)	Read (credit)
Authenticate and authorize	Reserve (premium)	Calculate (credit)	Paychain (credit/debit)
Unlock (credit)	Deploy (IoT appID)	Write (credit)	Write (credit)
:	Write (credit)	Commit	Commit
	Commit	Unlock (credit)	Unlock (credit)
	Unlock (credit)	:	:
	:		

of Credit and Debit. Here is the schedule of conflict serializability for read(), write(), and commit() operations on the credit resource object of the client.

We use the following notation: Let T_i be a transaction and *credit* be a relation or a data resource of a relation. Assuming $O_{ij} \in R(credit), W(credit)$ be an atomic read/write operation of T_i on data item *credit*. Then the two operations O_{ij} and O_{ik} on the same *credit* data resource are in conflict if at least one of them is a write operation. To avoid such conflicts, we have come up with the DoT transaction schedule in Table 1.

9 Related Work

In relation to our work, there is some commendable research regarding the cloud service usage metering. Elmsroth et al. [8] proposed a loosely coupled architecture solution for an accounting and billing system for use in the RESERVOIR [9] project. There are some alternatives that propose billing and metering solutions, Narayan et al. [10]. Petersson [11] describes cloud metering and billing solution. Naik et al. [12] proposed a solution for metering of services delivered from multiple cloud providers. They incorporate the cloud service broker together with a metering control system to report metered data at configurable intervals. Meanwhile, there are some prominent studies on capturing pricing models for IoT domains. Aazam and Huh [13] provided a resource prediction, resource price estimation and reservation for the Fog which resides between underlying IoTs and the cloud. Their proposed model does not support the real-time session and event-based metering models.

Mazhelis et al. [14] studies the applicability of the Constrained Application Protocol (CoAP), a lightweight transfer protocol under development by IETF, for efficiently retrieving monitoring and accounting data from constrained devices. Their results indicate that CoAP is suited for efficiently transferring monitoring and accounting data, both due to a small energy footprint and a memory-wise compact implementation. This work relies on using the accounting and monitoring infrastructure produced by the Accounting and Monitoring

of Authentication and Authorization Infrastructure Services (AMAAIS) project [15]. Our work elevates the metering to an IoT domain by proposing an extended Diameter protocol that enables IoT infrastructures to incorporate the DoT protocol in their deployment models. This contributes to a real-time resource utilization awareness by constructing and allocating flexible metering models to IoT deployment units.

10 Conclusion

In this study, we have presented a metering protocol, called the Diameter of Things (DoT), that enables real-time telemetry and automatic resource allocation control of IoT applications. The DoT is designed to enhance the resource utilization by extending the Diameter base protocol. The authors offered the DoT architecture, its interrogations as well as its transaction model for accounting for the resource usage of constrained devices. The DoT offers considerable benefits, such as granular metering of lightweight applications, real-time transparency over resource usage in edge devices and transporting and exchanging application-specific metering policies in an IoT domain. As an outlook, our future work includes the DoT implementation together with an evaluation of the proposed DoT architecture in terms of scalability, performance and session management. A concise evaluation of the requirements[4] for the Diameter-based protocols is proposed that will be considered for the DoT implementation as well as evaluation purposes. We envision the DoT protocol to gaining acceptance as a de facto metering standard in IoT.

Acknowledgments. The research leading to these results is sponsored by the Doctoral College of Adaptive Distributed Systems (ADSys) (http://www.big.tuwien.ac.at/adaptive) at the Vienna University of Technology.

References

1. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **25**(6), 599–616 (2009)
2. Calhoun, P., Johansson, T., Perkins, C., Hiller, T., Mccann, P.: Diameter mobile IPv4 application. In: IETF, RFC 4004, August 2005
3. Hakala, H., Mattila, L., Koskinen, J.-P., Stura, M., Loughney, J.: Diameter credit-control application. In: IETF, RFC 4006, August 2005
4. Garcia-Martin, M.: Diameter session initiation protocol (SIP) application. In: IETF, RFC 4740, April 2006
5. Rigney, C., Rubens, A., Simpson, W., Willens, S.: Remote authentication dial in user service (RADIUS). In: IETF, RFC 2138, June 2000
6. Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.: Diameter base protocol. In: IETF, RFC 3588, September 2003

[4] https://tools.ietf.org/html/draft-zander-ipfix-diameter-eval-00.

7. Rivest, R.L., Shamir, A.: PayWord and MicroMint: two simple micropayment schemes. In: Lomas, M. (ed.) Proceedings of the International Workshop on Security Protocols, pp. 69–87. Springer, London (1997)
8. Elmroth, E., Marquez, F.G., Henriksson, D., Ferrera, D.P.: Accounting and billing for federated cloud infrastructures. In: Proceedings of the 2009 Eighth International Conference on Grid and Cooperative Computing, GCC 2009, Washington, DC, USA, pp. 268–275. IEEE Computer Society (2009)
9. Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I.M., Montero, R., Wolfsthal, Y., Elmroth, E., Cáceres, J., Ben-Yehuda, M., Emmerich, W., Galán, F.: The reservoir model and architecture for open federated cloud computing. IBM J. Res. Dev. **53**(4), 535–545 (2009)
10. Narayan, A., Rao, S., Ranjan, G., Dheenadayalan, K.: Smart metering of cloud services. In: 2012 IEEE International Systems Conference (SysCon), pp. 1–7, March 2012
11. Petersson, J.: Cloud metering and billing. http://www.ibm.com/developerworks/cloud/library/cl-cloudmetering. Accessed 08 Aug 2011
12. Naik, V.K., Beaty, K., Kundu, A.: Service usage metering in hybrid cloud environments. In: 2014 IEEE International Conference on Cloud Engineering (IC2E), pp. 253–260, March 2014
13. Aazam, M., Huh, E.-N.: Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications (AINA), pp. 687–694, March 2015
14. Mazhelis, O., Waldburger, M., Machado, G.S., Stiller, B., Tyrväinen, P.: Retrieving monitoring and accounting information from constrained devices in internet-of-things applications. In: Doyen, G., Waldburger, M., Čeleda, P., Sperotto, A., Stiller, B. (eds.) AIMS 2013. LNCS, vol. 7943, pp. 136–147. Springer, Heidelberg (2013)
15. Stiller, B.: Accounting and monitoring of AAI services. Switch J. **2010**(2), 12–13 (2010). IETF