Distributed Systems Group
Internet-Draft

Intended status: Informational
Expires: March 2016

S. Qanbari
S. Mahdizadeh
S. Dustdar
TU Wien
N. Behinaein
R. Rahimzadeh
BIHE

September 20, 2015

Diameter of Things (DoT): A Protocol for Real-time Telemetry of IoT Applications
draft-tuwien-dsg-diameterofthings-01

Abstract

The Diameter of Things (DoT) protocol is intended to provide a 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 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.

Status of This Memo

This Internet-Draft will expire on March 20, 2016.

Copyright Notice

Table of Contents

1. Introduction

Utility computing\cite{Buyya:2009:CCE:1528937.1529211} 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 are being utilized as metered services. Such awareness mechanisms enable IoT platforms to adjusts the resource leveling to not exceed the elasticity constraints such that stringent QoS is achievable.

Respecting resource elasticity requirements on edge devices establishes a firm requirement of a lightweight protocol with support of fine-grained telemetry for 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 a DoT protocol that can be incorporated to implement real-time AAA of IoT services for prepaid subscribers to make and enforce trade-off decisions.

Based on this protocol, IoT infrastructure providers are able to collect the emitted usage data, then generate billable artifacts from a chain of metering transaction tokens. Finally it permits micro-payments to take place in parallel. This document specifies a DoT application that can be used to implement real-time telemetry for a variety of IoT applications.

1.1. Quest for Metering

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] to an IoT domain. There are several established Diameter base protocol applications like Mobile IPv4 [RFC4004], Diameter Credit-Control [RFC4006] and Session Initiation Protocol (SIP) [RFC4740] 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 focus 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.

1.2. Terminology

AAA
: Authentication, Authorization, and Accounting for a specific IoT application

AA answer

AA answer generically refers to a service specific authorization and authentication answer. AA answer commands are defined in service specific authorization applications, e.g., [NASREQ] and [DIAMMIP].

AA request

AA request generically refers to a service specific authorization and authentication request. AA request commands are defined in service specific authorization applications e.g., [NASREQ] and [DIAMMIP].

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 resources) 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.

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 resource usage. It basically transfers a certain amount of trade in the payWord. In the Pay Word 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 a Requesting DoT Node and a Responding DoT Node.

2. Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

3. Architecture Models

Figure 1 illustrates a schematic view on collaborating components of DoT architecture. It contains of a DoT client, Provisioning server, Resource control server, Metering server, and a Payment server.

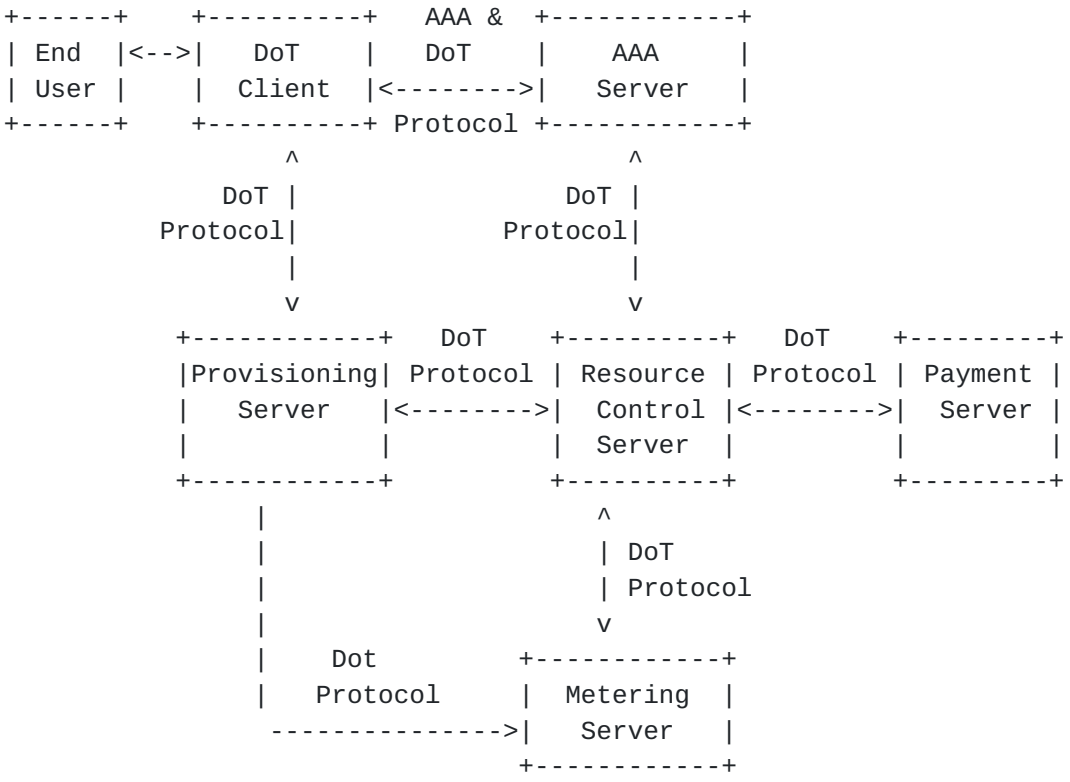


Figure 1: Typical Diameter of Things (DoT) application architecture

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 AAA 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 resources that need to be allocated to the service.

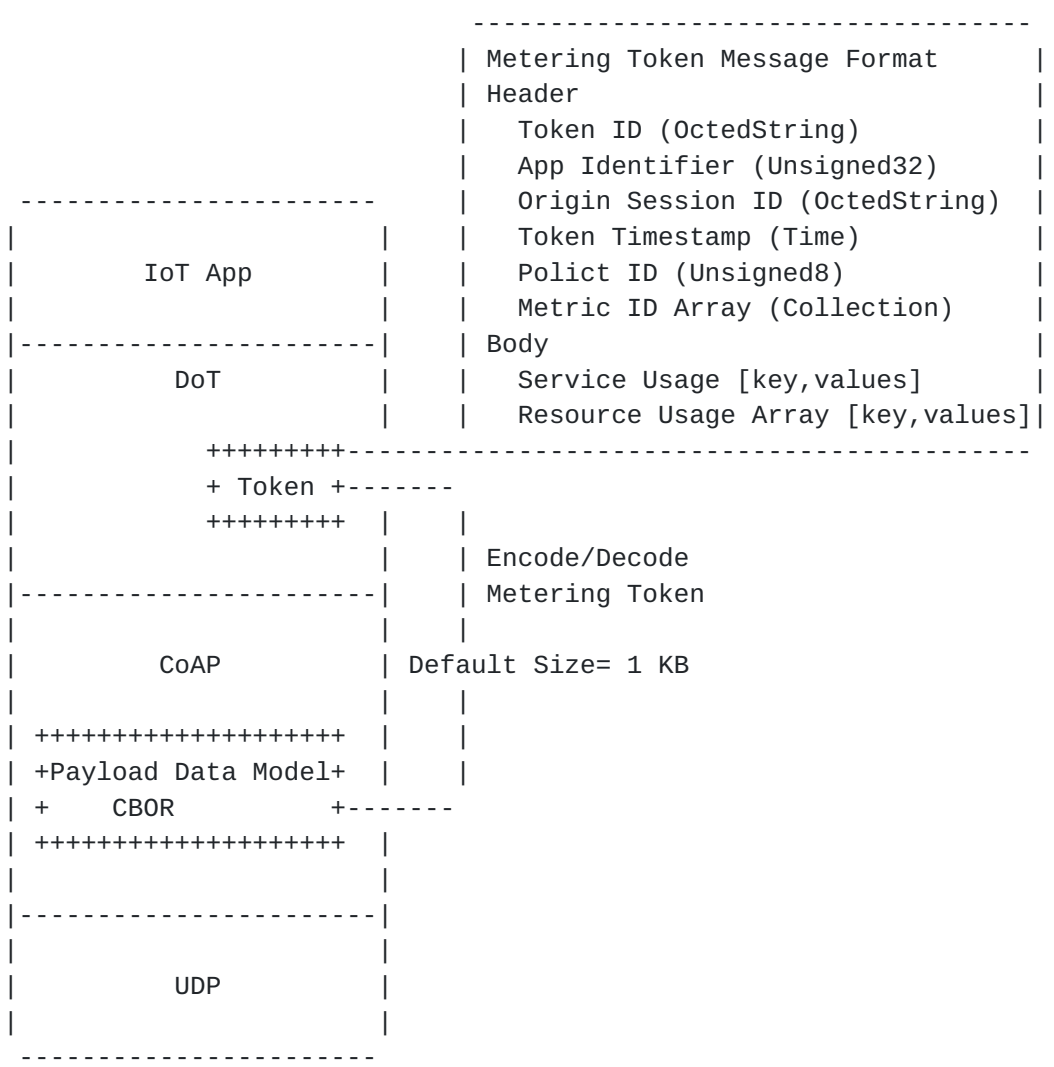
Finally the DoT client receives the grant resource message and informs the end user that the request has been granted. Next, the IoT application is deployed and instantiated, then the submitted topology is registered to metering server for telemetry and credit control purposes.

The metering server is responsible to perform the metering transactions according to the submitted topology and meter the services by calling metering task 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 termination request, the resource control server releases the resources 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.

4. DoT Metering Token Messages

In DoT application architecture, metered values are transfered via tokens. This section defines DoT token message attributes that MUST be supported by all DoT implementations that conform to this specification. The CBOR [RFC7049] message format is considered for the metering tokens transmission since it can decrease payload sizes compared to other data formats.




```
<DoT-Token-Message> ::=   < Token-Id >
                             { App-Identifier }
                             { Origin-Session-Id }
                             { Token-Timestamp }
                             { Policy-Id }
                             { Metric-Id-Array }
                             { Service-Usage }
                             { Resource-Usage-Array}
```

Figure 2: Diameter of Things (DoT) Metering Token Structure

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.

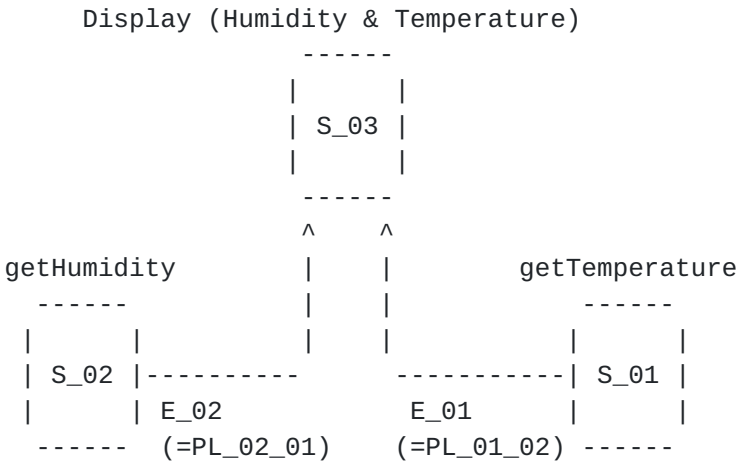
Each IoT microservice is predefined with a detailed specification such as its ID, constraints and various usage patterns and policies. For instance, as defined in List 1. they can be advertised with diverse pricing models due to the event-based or time-based patterns for specific subscribers. The IoT microservices elasticity requirements and constraints (hardware or software resources) are also defined in the topology which will dictate our schedule 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.

```
{   "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":"session mode - $2 per week"},
                {"policyID":"PL_01_02",
                 "cost":"0.01cent/invoke",
                 "desc":"event mode - 0.01cent per invoke"},
                {"policyID":"PL_01_03",
                 "cost":"$1",
                 "desc":"subscription fee"},
                ], }
```

List 1. Sample IoT microservice elasticity requirement definition

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. The sample of dependency graph in the form of incidence matrix is shown in the Figure 3.



	E_01	E_02
S_03	(PL_01_02)	(PL_02_01)
S_02	0	-(PL_02_01)
S_01	-(PL_01_02)	0

Figure 3: Typical DoT Application Topology and its Incident Matrix

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.

5.2. DoT-based Metering Plan

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 can be a subscription fee for such 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.

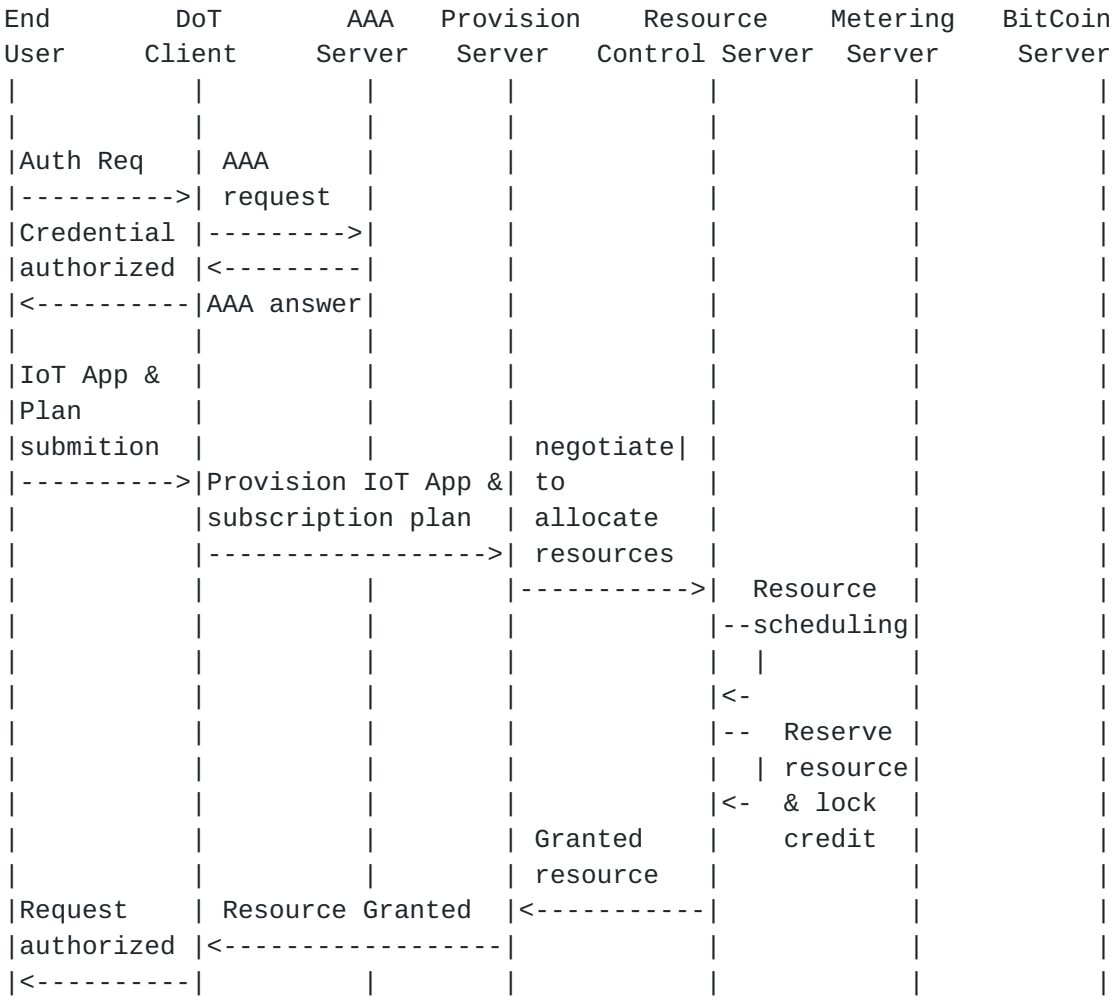
The subscribed metering plan indicates: (i) the price of granted units for every IoT application constituent microservice. For instance, 5 hours for Humidity sensor and 10 invocations for Chiller On/Off actuator. (ii) the resource Used Unit Update (U3) frequency for all associated units which are defined 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.

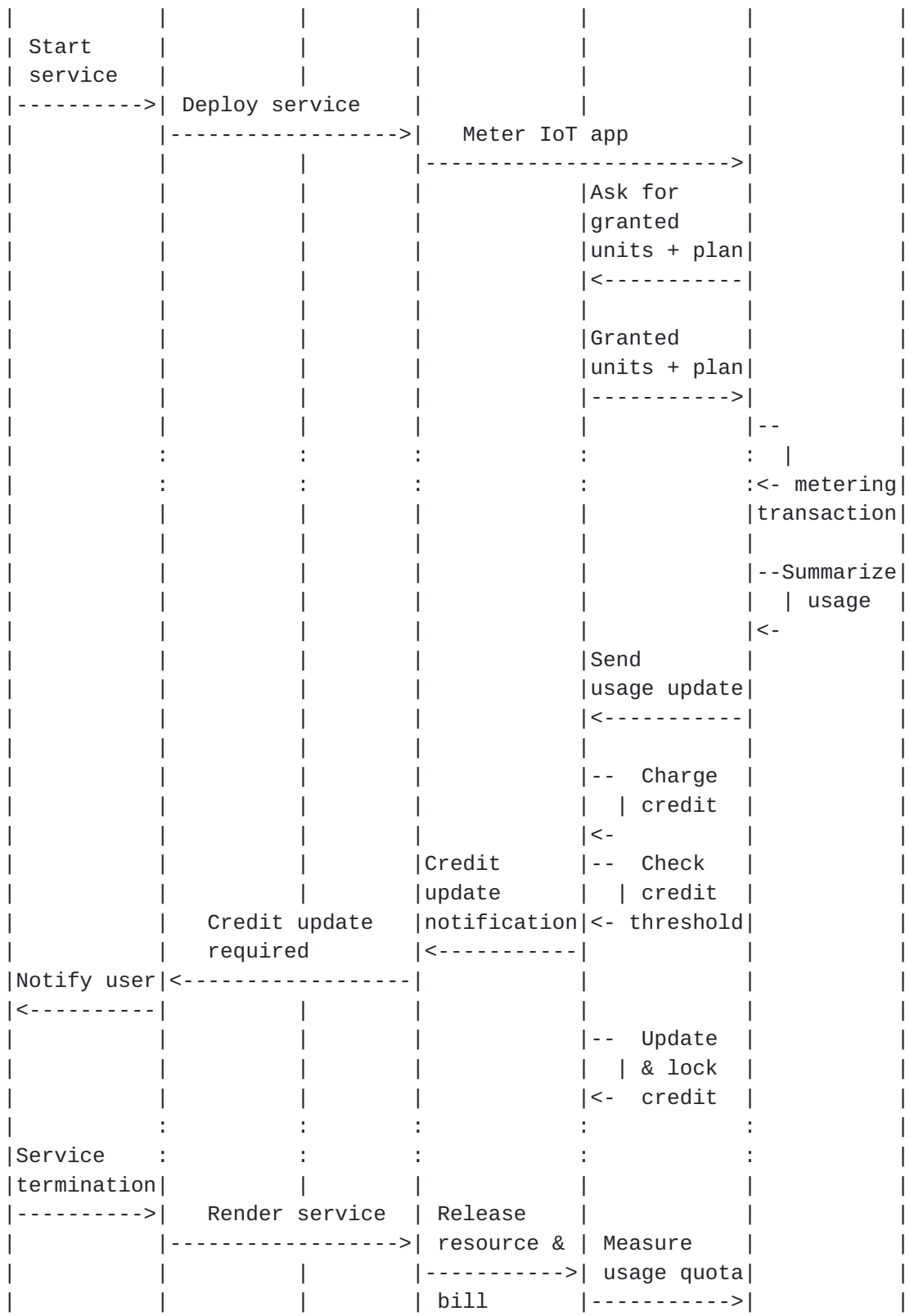
6. DoT Interrogations

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 in prepaid model and pay-per-use model are presented in Figure 4 and Figure 5 respectively.




Start service
Deploy service
Meter IoT app
Ask for granted units + plan
Granted units + plan
metering transaction
Summarize usage
Send usage update
Charge credit
Credit update notification
Check credit threshold
Credit update required
Notify user
Update & lock credit
Service termination
Render service
Release resource & bill
Measure usage quota

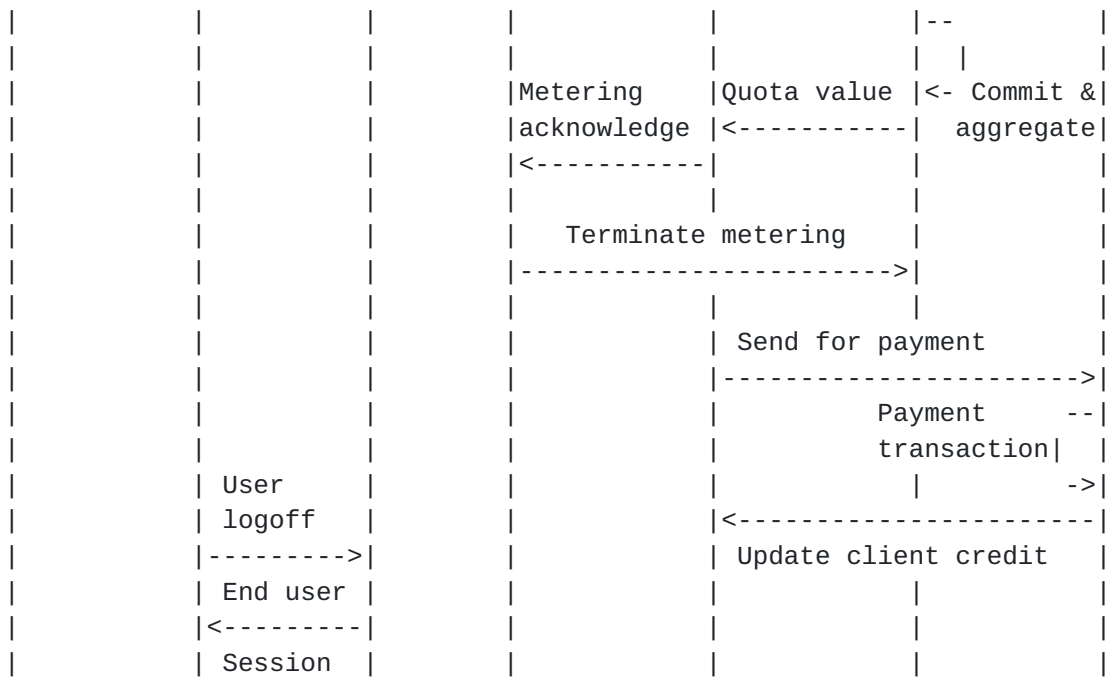
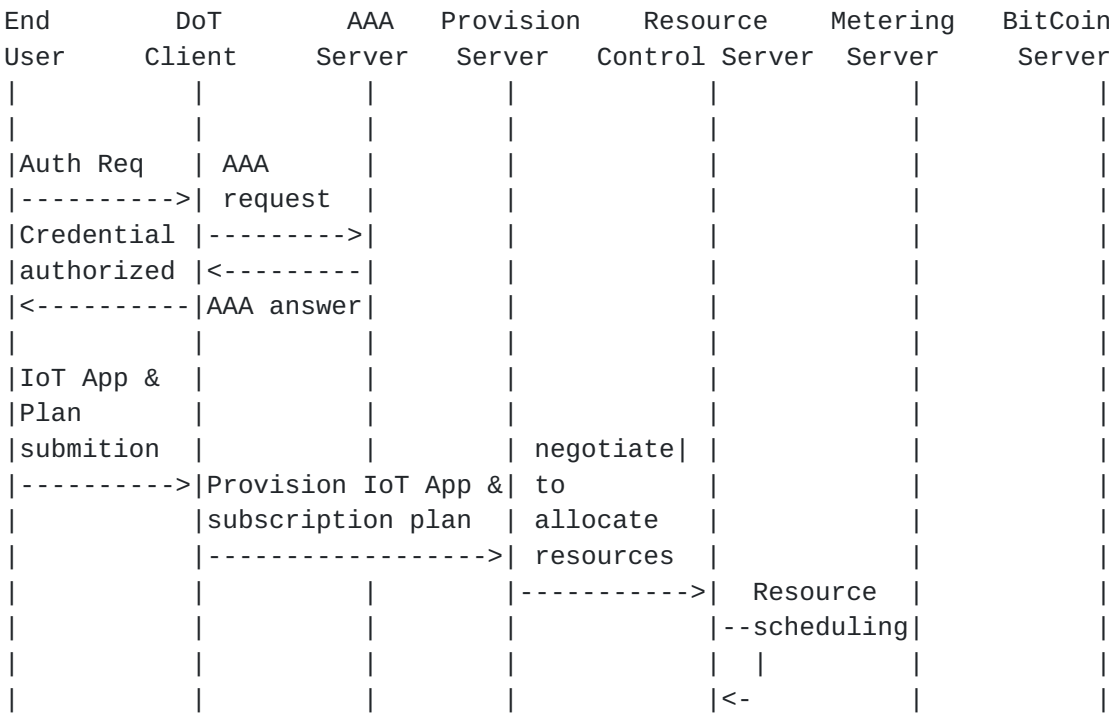
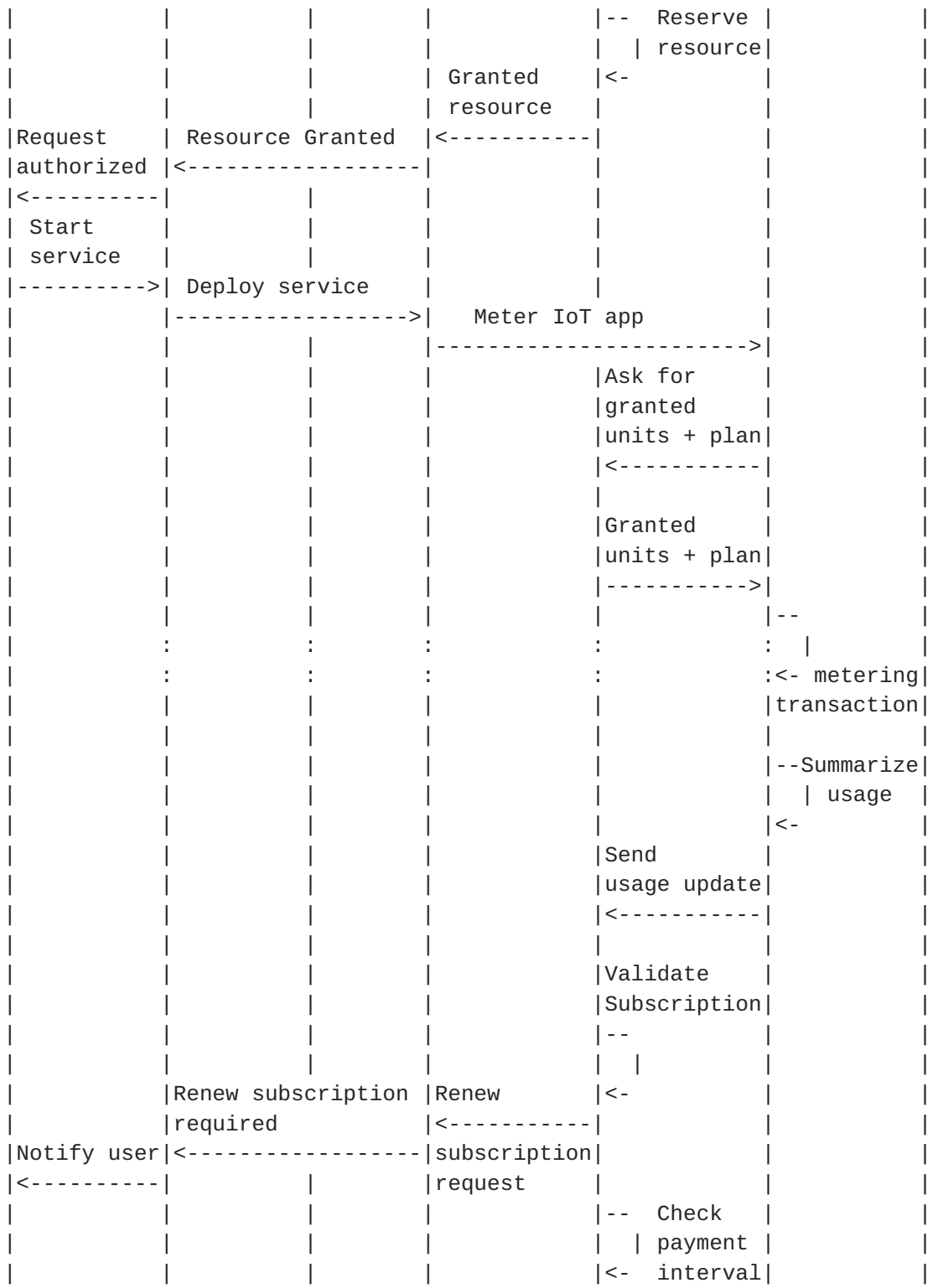


Figure 4. The sequence of DoT interrogations in prepaid model to enable hybrid metering




Reserve resource
Granted resource
Request authorized
Resource Granted
Start service
Deploy service
Meter IoT app
Ask for granted units + plan
Granted units + plan
metering transaction
Summarize usage
Send usage update
Validate Subscription
Renew subscription required
Renew subscription request
Notify user
Check payment interval

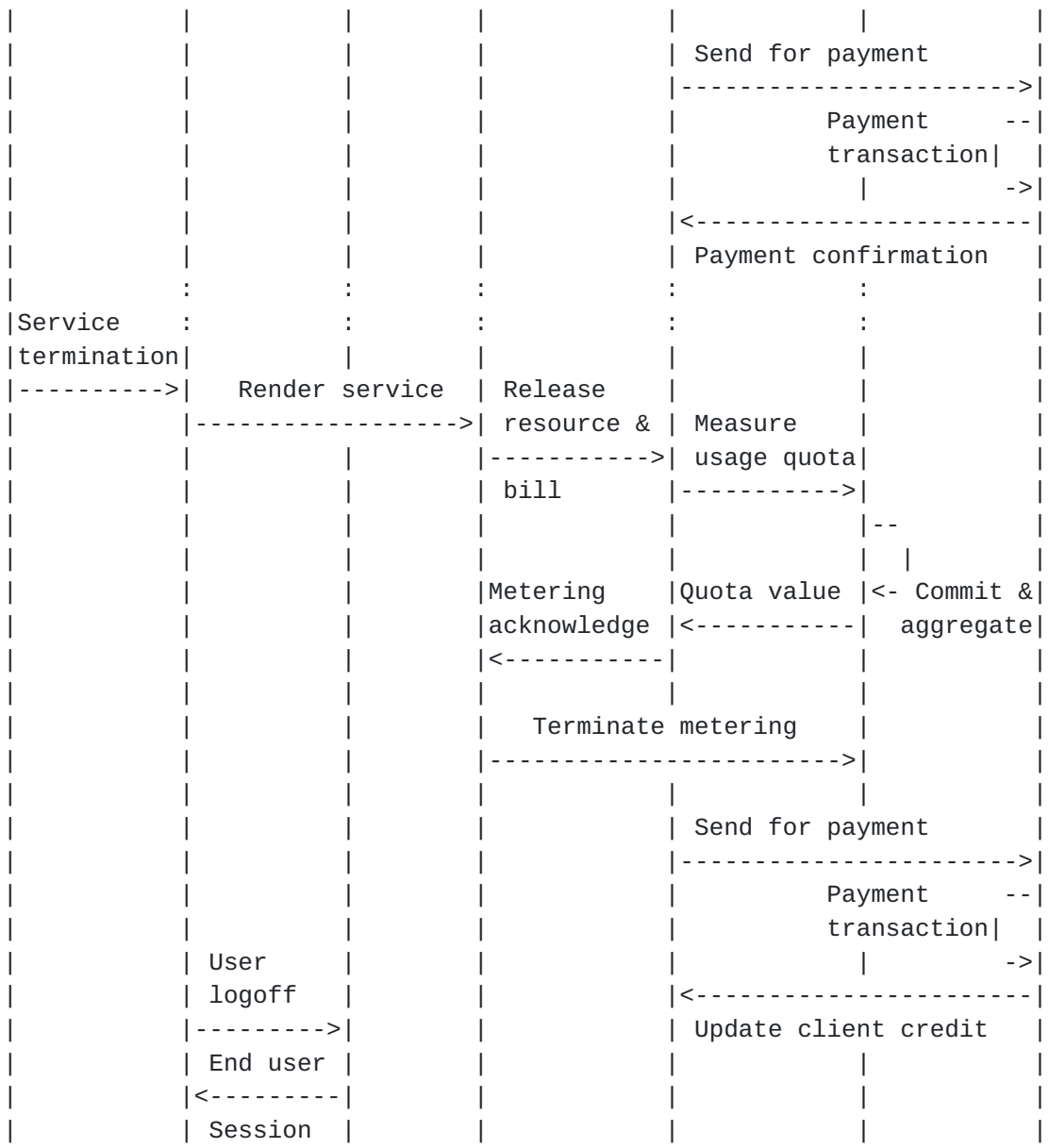


Figure 4. The sequence of DoT interrogations in pay-per-use model to enable hybrid metering

6.1. Initial Identification

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 sub-session-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.

6.3. Telemetry Transmission (TT)

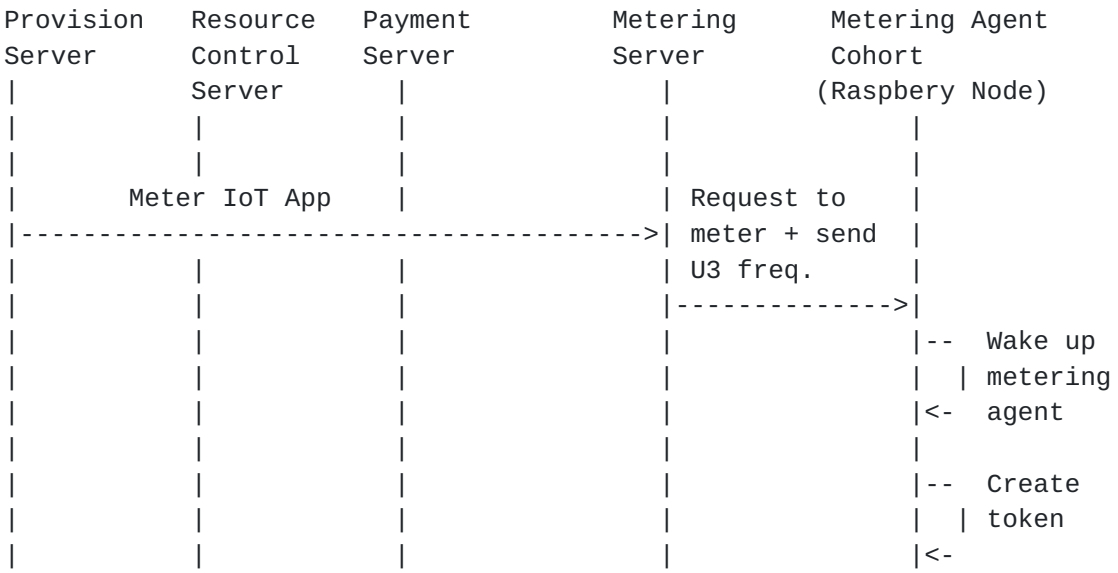
As soon as the end user sends the Start service request to the DoT client, the DoT Client asks the Provisioning server to initiate the service and start monitoring and metering processes. In this regard, the Provisioning server submits the IoT application to the Metering server and asks to establish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

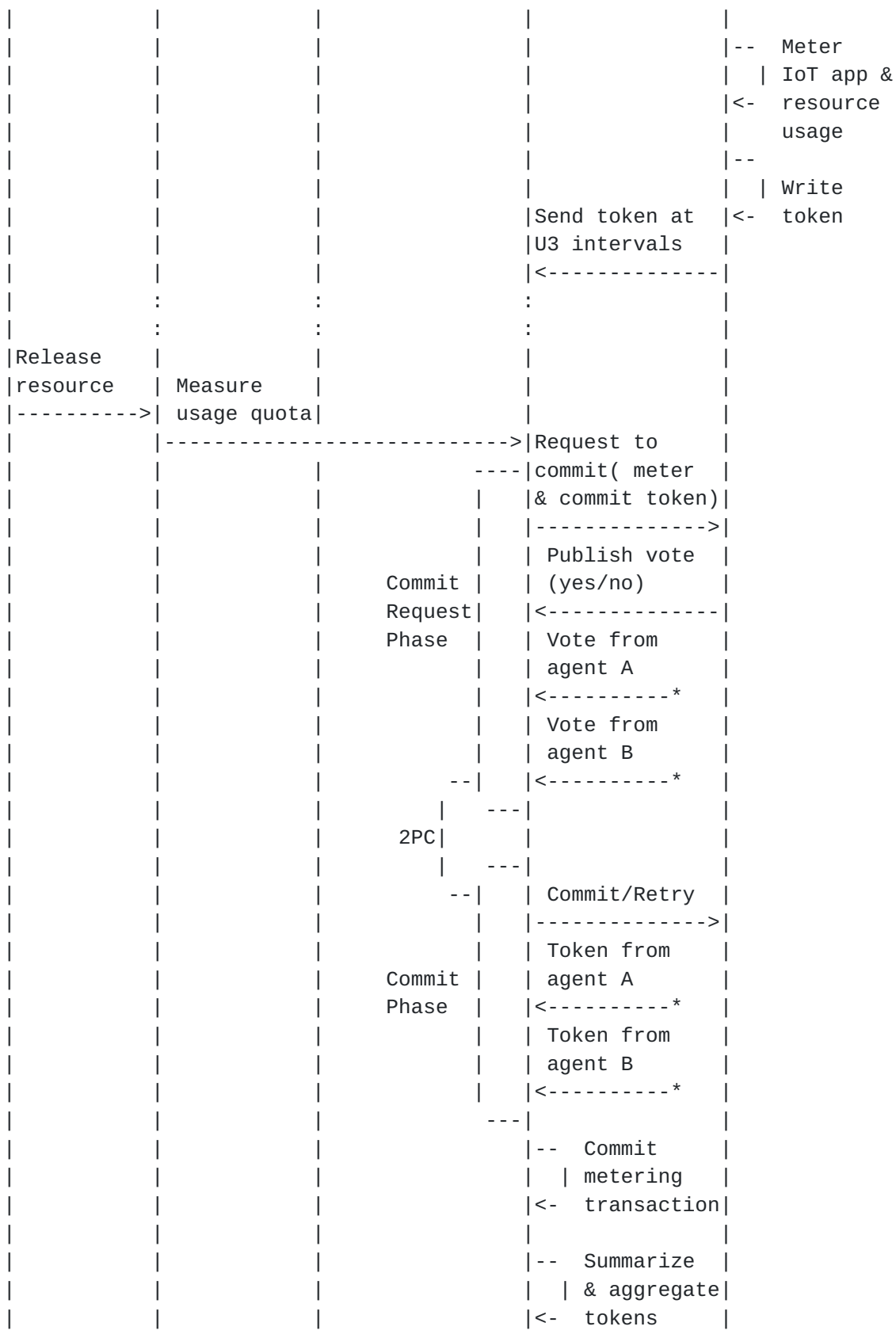
As soon as an IoT application is deployed, metering server monitors the real usage of each service element including service usage and resource usage at a certain frequency rate called Unit Usage Update (U3). The U3 frequency determines the rate of sending updates regarding unit usage of each service. 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 minutes, the usage update should be provided every 25 minutes; 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 which are reserved for all microservices involved in the application as well as the plan, which includes the U3 value for each microsevice 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 Figure 6.




Meter
IoT app &
resource
usage
Write
token
Send token at
U3 intervals
Release
resource
Measure
usage quota
Request to
commit(meter
& commit token)
Publish vote
(yes/no)
Commit
Request
Phase
Vote from
agent A
Vote from
agent B
2PC
Commit/Retry
Token from
agent A
Token from
agent B
Commit
Phase
Commit
metering
transaction
Summarize
& aggregate
tokens

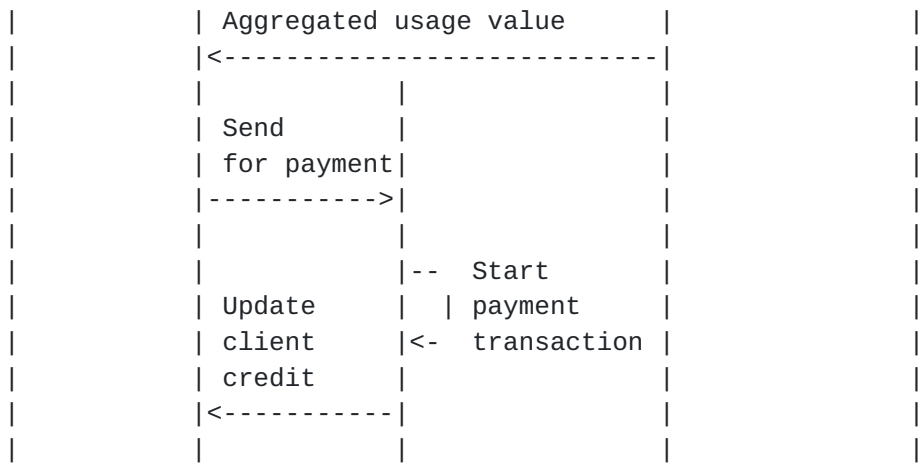


Figure 6. DoT Hybrid Metering - 2PC transaction model

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 the 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. Meanwhile DoT Client drops the user session throughout 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.

7. DoT 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.

The following command codes are defined in DoT application:

Abbr	Command Name	Sender	Receiver
PATR	Provision-Application-Topology-Request	DoT client	Provisioning server
PATA	Provision-Application-Topology-Answer	Provisioning server	DoT client
ARAR	App-Resource-Allocation-Request	Provisioning server	Resource Control server
ARAA	App-Resource-Allocation-Answer	Resource Control server	Provisioning server
SIAR	Start-IoT-App-Request	DoT client	Provisioning server
SIAA	Start-IoT-App-Answer	Provisioning server	DoT client
TIAR	Terminate-IoT-App-Request	DoT client	Provisioning server
CUSR	Close-User-Session-Request	DoT client	AAA server
ARRR	App-Resource-Release-Request	Provisioning server	Resource Control server
TIAA	Terminate-IoT-App-Answer	Provisioning server	DoT client
CUSA	Close-User-Session-Answer	AAA server	DoT client

ARRA	App-Resource-Release-Answer	Resource Control server	Provisioning server
SBPR	Start-Bill-Payment-Request	Resource Control server	Payment server
SBPA	Start-Bill-Payment-Answer	Payment server	Resource Control Server
CAMR	Commit-App-Metering-Request	Resource Control server	Metering server
CAMA	Commit-App-Metering-Answer	Metering server	Resource Control Server
SAMR	Start-App-Metering-Request	Provisioning server	Metering server
SAMA	Start-App-Metering-Answer	Metering server	Provisioning server
UACR	Update-Allocation-Confirmation -Request	Resource control server	Rrovisioning server
NUAR	Notify-User-Allocation-Request	Provisioning server	DoT client
UACA	Update-Allocation-Confirmation -Answer	Provisioning server	Resource Control Server
NUAA	Notify-User-Allocation-Answer	DoT client	Provisioning server
GASR	Get-App-Specification-Request	Metering server	Resource control server
GASA	Get-App-Specification-Answer	resource control server	Metering server

PUUR	Publish-Usage-Update-Request	Metering server	resource control server
PUUA	Publish-Usage-Update-Answer	resource control server	Metering server
TAMR	Terminate-App-Metering-Request	Provisioning server	Metering server
TAMA	Terminate-App-Metering-Answer	Metering server	Provisioning server

Four main commands are explained here in more details.

7.1. Provision-Application-Topology-Request/Answer

The PATR 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 ARAR command indicates the operation communicated between the Provisioning and Resource control server. This command aims to reserve resources including underlying resources as well as credit allocation for the provisioned application.

7.3. Commit-App-Metering-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.

7.4. 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 Application State Machine

This section defines the DoT application protocol state machines. There are five different state machines for each entity in DoT protocol. The first state machine describes the states of DoT client. The second one describes the Provisioning server state machine. The third one is the state machine of Resource control server. The fourth and fifth state machines describe the Metering server and Payment server accordingly.

In DoT client state machine, the states PendingI, PendingP, PendingD, PendingT stand for pending states to wait for an answer to an already sent request related to Initial, Provisioning, Deployment, Termination requests.

In Provisioning server state machine, the states PendingR and PendingM stands for states of waiting for an answer from the Resource control server and the Metering server respectively.

In Resource control server state machine, the states PendingPY and PendingPM correspond to the state of waiting for the payment and metering answer.

In Metering server state machine, the states PendingG and PendingUU stand for pending states for Granted unit and Usage Update information

The event 'Tw expired, Failure to send, temporary error' in the state machines indicates the situation where there is a problem in network, preventing one entity to communicate with the desired entity, or the cases where the destination entity is too busy and cannot handle the request at that time.

Dot client

State	Event	Action	New State
Idle	AA request received from end user	Send AA request to AA server, start Tw	PendingI
pendingI	Successful AA answer received	Submit application topology as well as the plan to Provisioning sever (PATR), restart Tw	PendingP
pendingI	Tw expired, Failure to send, temporary error	Retry sending AA request to AA server, restart Tw	PendingI
PendingP	Answer received from Provisioning server (PATA)	Submit request to start the IoT application to Provisioning server (SIAR),restart Tw	PendingD
PendingP	Answer received from Provisioning server (PATA) with result code != SUCCESS	Fix the problem and send the request again, restart Tw	PendingD
PendingD	Answer received from Provisioning server (SIAA)	Stop Tw, Inform end user that service & monitoring have beed started	Open
PendingD	Answer received from Provisioning server (SIAA) with result code != SUCCESS	Fix the problem and send the request again, restart Tw	PendingD
PendingD	Tw expired, Failure to send, temporary error	Retry sending Provisioning request to Provisioning server (SIAR), restart Tw	pendingD

Open	Update notification received from Provisioning server (NUAR) with USER_INPUT equal to False	Inform user about the update, send answer back to Provisioning server (NUAA)	Open
Open	User confirmation request recieved for the updating credit from Provisioning server (NUAR) with USER_INPUT equal to True	Send update confirmation request to user	Open
Open	User confirmation request recieved (NUAR) but not successfully processed	Send answer back to Provisioning server (NUAA) with result code !=SUCCESS	Open
Open	User confirmed the update	Send update confirmation answer (NUAA) with status=CONFIRM to Provisioning server	Open
Open	User rejected the update	Send update confirmation answer (NUAA) with status=REJECT to Provisioning server	Open
Open	User sends termination request	Send termination request to Provisioning server (TIAR), start Tw	PendingT
PendingT	Answer received from Provisioning server (TIAA)	Inform user about termination, stop Tw	Idle
PendingT	Answer received from Provisioning server (TIAA) with result != SUCCESS	Fix the problem and send the request again, restart Tw	PendingT
PendingT	Tw expired, Failure to send, temporary error	Retry sending termination request to Provisioning server (TIAR), restart Tw	PendingT

Provisioning server

State	Event	Action	New State
Idle	Request to provision application topology from DoT client (PATR) received and successfully processed	Send the resource allocation request to resource control server (ARAR), Start Tw	PendingR
Idle	Request to provision application topology from DoT client (PATR) received but not successfully processed	Send the provision topology answer to DoT client (PATA) with Result-Code!=SUCCESS	Idle
PendingR	Answer of resource allocation from resource control server (ARAA) received	Send the provision topology answer to DoT client (PATA), Stop Tw	Idle
PendingR	Answer of resource allocation from resource control server (ARAA) received with result code!=SUCCESS	Fix the problem and send the resource allocation request to resource control server (ARAR) again, Restart Tw	PendingR
PendingR	Tw expired, Failure to send, temporary error	Restart Tw, Retry sending the resource allocation request to resource control server (ARAR)	PendingR
Idle	Request to start the IoT application from DoT client (SIAR) received and successfully processed	Send the start metering request to metering server(SAMR), Start Tw	PendingM
Idle	Request to start the IoT application from DoT client(SIAR) received but not successfully processed	Send the start app answer to DoT client (SIAA) with Result-Code!= SUCCESS	Idle

PendingM	Answer of starting IoT application from metering server (SAMA) received	Send the start app answer to DoT client (SIAA), Stop Tw	Open
PendingM	Answer of starting IoT application from metering server (SAMA) received with result code!=SUCCESS	Fix the problem and send the start metering request to metering server (SAMR) again, Restart Tw	PendingM
PendingM	Tw expired, Failure to send, temporary error	Restart Tw, Retry sending the start metering request to metering server (SAMR) again	PendingM
Open	Request to notify user about updating resource allocation from resource control server (UACR) received and successfully processed	Send the allocation notification to DoT client (NUAR)	Open
Open	Request to terminate IoT application from DoT client (TIAR) received and successfully processed	Send the resource release request to resource control server (ARRR), Start Tw	PendingR
Open	Request to terminate IoT application from DoT client (TIAR) received but not successfully processed	Send the terminate app answer to DoT client (TIAA) with Result-Code!= SUCCESS	Open
PendingR	Answer of confirming the release of allocated resources from resource control server (ARRA) received	Send the terminate metering request to metering server (TAMR), Start Tw	PendingM
PendingR	Answer of confirming the release of allocated resources from resource control server (ARRA) received with result code!=SUCCESS	Fix the problem and send the resource release request to resource control server (ARRR) again, Restart Tw	PendingR

PendingR	Tw expired, Failure to send, temporary error	Restart Tw, Retry sending a Request to release the allocated resources to resource control server (ARRR)	PendingR
PendingM	Answer of confirming metering termination from metering server (TAMA) received	Send the terminate app answer to DoT client (TIAA), Stop Tw	Idle
PendingM	Answer of confirming metering termination from metering server (TAMA) received with result code!=SUCCESS	Fix the problem and send the terminate metering request to metering server (TAMR) again, Restart Tw	PendingM
PendingM	Tw expired, Failure to send, temporary error	Terminate the service, Send the terminate app answer to DoT client (TIAA)	Idle

Resource control server

State	Event	Action	New State
Idle	Request to reserve resources from provisioning server(ARAR) received and successfully processed	Perform resource scheduling, reserve resources (based on plan) , lock the credit, send the resources allocation acknowledgement answer to provisioning server (ARAA)	Open
Idle	Request to reserve resources from provisioning server(ARAR) received but not successfully processed	send the resources allocation answer to provisioning server (ARAA) with Result-Code!= SUCCESS	Idle

Open	Request to retrieve the Specification information from metering server (GASR) received and successfully processed	Send the answer containing granted unis and plan to metering server(GASA)	Open
Open	Request to retrieve the Specification information from metering server (GASR) received but not successfully processed	Send the getting specification answer to metering server (GASA) with Result-Code!= SUCCESS	Open
Open	Request to publish usage update from metering server (PUUR) received and successfully processed	Charge credit according the last sent usage, Send the acknowledge answer to metering server (PUUA)	Open
Open	Request to publish usage update from metering server (PUUR) received but not successfully processed	Send the usage update answer to metering server (PUUA) with Result-Code!=SUCCESS	Open
Open	Credit threshold is met	Update and lock credit, Send update notification to provisioning server (UACR)	Open
Open	Request to release the allocated resources from provisioning server(ARRR) received and successfully processed	Release the allocated resources, Send the commit request to metering server (CAMR), Start Tw	PendingM
Open	Request to release the allocated resources from provisioning server (ARRR) received but not successfully processed	Send the resource release answer to provisioning server (ARRA) with Result-Code!= SUCCESS	Open

PendingM	Answer of commiting metering containing the quota values from metering server (CAMA) received	Send the resource release answer to provisioning server (ARRA), Send a request to payment server to charge the client based on billable artifact (SBPR), Restart Tw	PendingPY
PendingM	Answer of commiting metering from metering server (CAMA) received with result code!=SUCCESS	Fix the problem and send the commit request to metering server(CAMR) again, Restart Tw	PendingM
PendingM	Tw expired, Failure to send, temporary error	Retry sending the commit metering request to metering server(CAMR), Restart Tw	PendingM
PendingPY	Answer of billing payment from payment server (SBPA) received	StopTw	Idle
PendingPY	Answer of billing payment from payment server (SBPA) received with result code!=SUCCESS	Fix the problem and send a payment request to payment server (SBPR) again, Restart Tw	PendingPY
PendingPY	Tw expired, Failure to send, temporary error	Retry sending a payment request to payment server (SBPR), Restart Tw	PendingPY

Metering server

State	Event	Action	New State
Idle	Request to start metering received from Provisioning server (SAMR)	Send answer to the Provisioning server (SAMA), send request to Resource ctrl server asking for granted Units and plan (GASR),start Tw	PendingG
Idle	Request to start metering received (SAMR) but not successfully processed	Send answer to the Provisioning server (SAMA) with result code!=SUCCESS	Idle
PendingG	Answer received from Resource ctrl server including granted units and plan (GASA)	Send service specific and U3 to each metering agent, start metering transaction, stop Tw	Open
PendingG	Answer received from Resource ctrl server with result code!=SUCCESS	Fix the problem andsend the request again, restart Tw	PendingG
PendingG	Tw expired, Failure to send, temporary error	Retry sending request(GASR) again, restart Tw	PendingG
Open	Unit usage update message recieved from a metering agent	Summarize the usage and send it to Resource ctrl server (PUUR), start Tw	PendingUU
PendingUU	Answer received from Resource ctrl server	Stop Tw	Open
PendingUU	Tw expired, Failure to send, temporary error	Retry sending request(PUUR) again, restart Tw	PendingUU

PendingUU	Answer received from Resource ctrl server with result code !=SUCCESS	Fix the problem and send the request again, restart Tw	PendingUU
Open	Request to commit the measured metering tokens and report the quota value received from Resource ctrl. server (CAMR)	Aggregate tokens and send the answer back to Resource ctrl server (CAMA)	Open
Open	CAMR request recieved but not successfully processed	Send the answer back to Resource ctrl server (CAMA) with result code !=SUCCESS	Open
Open	Request to terminate metering received from Provisioning server (TAMR)	Terminate metering, send Answer back to Provisioning server (TAMA)	Idle

Payment server

State	Event	Action	New State
Idle	Request to charge the client based on billable artifact from resource control server (SBPR) received and successfully processed	Generate a bill by Invoking the payment transaction, Deduct credit from the end user's account and refund unused reserved credit to user's account, Send start bill payment answer to resource control server(SBPA)	Idle
Idle	Request to charge the client based on billable artifact from resource control server (SBPR) received but not successfully processed	Send start bill payment answer to resource control server(SBPA) with Result-Code!= SUCCESS	Idle

9. Formal Syntax

The following syntax specification uses the JavaScript Object Notation (JSON) Data Interchange Format as described in RFC-7159 [RFC7159].

List 2 presents the sample dependency graph together with its metering policies in the form of a JSON object.

```
{
 "graphLabel":"home_temp_hum_display",
 "nodes":[{ "id":"S_01",
          "name":"getTemperature",
          "type":"node",},
        { "id":"S_02",
          "name":"getHumidity"
          "type":"node",},
        { "id":"S_03",
          "name":"Display",
          "type":"node",}],
 "edges":[{"id":"E_01",
         "directed":"True",
         "source":"S_01",
         "destination":"S_03",
         "label":"PL_01_02"},
        {"id":"E_02",
         "directed":"True",
         "source":"S_02",
         "destination":"S_03",
         "label":"PL_02_01"
         }]
}
```

List 2. Sample IoT microservice elasticity requirement definition

10. Security Considerations

This document has not conducted its security analysis, yet.

11. IANA Considerations

This draft does not specify its IANA considerations, yet.

12. Conclusions

In this draft, the authors offer a protocol entitled "Diameter of Things (DoT)" that realizes the telemetry mechanisms to the Internet of Things (IoT) domain. The TU Wien DSG will provide further improvements and implementations to the DoT protocol respectively.

13. References

13.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[NASREQ] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", RFC 4005, August 2005.

[RFC6733] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[RFC4004] Calhoun, P., Johansson, T., Perkins, C., Hiller, T.,and P. McCann, "Diameter Mobile IPv4 Application", RFC 4004, August 2005.

[RFC4006] Hakala, H., Mattila, L., Koskinen, J-P., Stura, M., and J. Loughney, "Diameter Credit-Control Application", RFC 4006, August 2005.

[RFC4740] Garcia-Martin, M., "Diameter Session Initiation Protocol (SIP) Application", RFC 4740, April 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Paul Hoffman, "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, Internet Engineering Task Force (IETF), March 2014.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

13.2. Informative References

[DIAMMIP] Calhoun, P., Johansson, T., Perkins, C., Hiller, T., and P. McCann, "Diameter Mobile IPv4 Application", RFC 4004, August 2005.

14. Acknowledgments

We truly appreciate and commend the insight provided by Diameter implementors who have motivated us to incorporate the Diameter mechanisms in IoT domain. The result is an extension to Diameter base protocol in which its specifications are proposed in this document.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Soheil Qanbari
Distributed Systems Group
Technical University of Vienna (TUWien)
Argentinierstrasse 8 / 184-1
1040 Vienna
Austria

Phone: +43-1-58801-18402
EMail: soheil@dsg.tuwien.ac.at

Soheil Qanbari
Distributed Systems Group
Technical University of Vienna (TUWien)
Argentinierstrasse 8 / 184-1
1040 Vienna
Austria

Phone: +43-1-58801-18402
EMail: soheil@dsg.tuwien.ac.at

Samira Mahdizadeh
Distributed Systems Group
Technical University of Vienna (TUWien)
Argentinierstrasse 8 / 184-1
1040 Vienna
Austria

Phone: +43-1-58801-18402
EMail: e1329639@student.tuwien.ac.at

Schahram Dustdar
Distributed Systems Group
Technical University of Vienna (TUWien)
Argentinierstrasse 8 / 184-1
1040 Vienna
Austria

Phone: +43-1-58801-18414
EMail: dustdar@dsg.tuwien.ac.at

Negar Behinaein
Computer Science Group
Baha'i Institute for Higher Education (BIHE)
Tehran/Iran

Email: negar.behinaein@bihe.org

Rabee Rahimzadeh
Computer Science Group
Baha'i Institute for Higher Education (BIHE)
Tehran/Iran

EMail: rabee.rahimzadeh@bihe.org