Presence and Availability Service at the Network Edge

Evelina Pencheva
Todor Kableshkov University of Transport,
Sofia, Bulgaria
evelina.nik.pencheva@gmail.com

Ivaylo Atanasov, Vladislav Vladislavov, Ventsislav Trifonov
Balkantel Ltd
Sofia, Bulgaria
i.i.a@abv.bg, {vladislav.vladislavov, ventsislav.trifonov}@balkantel.net

***Abstract*—Presence service enables to publish, store, and access presence information. Usually, it is used by humans along with voice and instant messaging services, but with the ubiquitous penetration of machine type communications it may by used also by smart devices. The paper presents a RESTful Presence service which may be deployed at the network edge. The main benefits are that the proposed service does not require deployment of Internet Protocol Multimedia Subsystem and saves network resources. In addition to information provided by the presentity, the service enables extension of published presence status with information provided by the radio access network.**

I. Introduction

Presence services enables subscribers to upload, store and watch presence information. Presence information reveals the presence supplier willingness and ability to communicate by preferred means. Using a presence application, the presence supplier publishes his/her dynamic profile, including device and personal status, device capabilities, location, context and preferred communication methods, via a network connection and the presence service stores this information. The presence service enables authorized presence consumers to access the published data and to initiate e.g., "polite" call, instant messaging, or gaming. The service allows users to control flexibly with whom and by what means they want to communicate. With presence service, team working is more efficient as it enables information sharing beyond availability e.g., future plans, meeting location and topic etc. In addition, presence service creates an alternative channel for information-sharing and advertising.

Having a close relationship to the mobility information, presence has its own specifics and thus, can serve as a "brick" in building of different types of services, including either new ICT services, or "upgrading" existing ones. When looking from the standpoint of new communications services then the presence service turns to be a key component if it is in question to conclude i.e., to infer the eventual acceptance for and level of engagement of given user in taking part in new types of communications based on the accessible presence information e. g. bot-based chat or voice-bot is the right "front-man" for given user, or when to switch to more appropriate one. The new information services might incorporate presence service as an ingredient when e.g. "abstract" entities have to provide services in the context of mobility - enhanced augmented/virtual reality site-seeing, traffic congestion information provisioning etc.

The Presence service may be deployed on Internet Protocol Multimedia Subsystem (IMS) platform. IMS users can share presence and content as a part of their multimedia communications. In this paper, we present an approach to define a new mobile edge service for presence management. One of the main advantages of the proposed approach is that it does not require deployment of distributed IMS functionality and is based on the capabilities exposed by Multi-access Edge Computing (MEC).

The paper is organized as follows. First, we present the related works on presence service. Next, a brief review on the presence service standards is given. Then, a detailed service description is provided where the service functionality is illustrated by typical use cases. The data model and data types, and supported methods are part of service Application Programming Interfaces (APIs). Some remarks on implementation of presence applications are provided.

II. Related works

The Presence service is mainly used in Voice over IP and Instant messaging services [1]–[3]. The Presence service may be deployed on Internet Protocol Multimedia Subsystem (IMS) platform. IMS users can share presence and content as a part of their multimedia communications. Different aspects of IMS-based presence service are studied in [4]–[7]. While humans are ordinary users of presence information, presence and content aware applications can enhance communications between smart devices which interact with each other [8]–[10].

The user point of view at a service that incorporates presence service consists of two main functionalities: a) a higher-level management of access of his/her own presence information that is provided to other services/users, say group level; b) a finer management level of granularity by explicit presence status provisioning to specific services/users, say individual level. As far as these management functionalities are granted to a big number of users, it becomes obvious that the number of combinations is even bigger, and this presumes a great number of interactions. One of the ways to make the Presence service scalable, to manage the needed resources in a flexible manner, and to handle with the anticipated load is the

cloud computing. A case study on presence service in cloud environment is presented in [11].

To reduce the Presence service traffic in the core network and to be more effective at the place where it is used, the presence service may be deployed at the network edge. Multi-access Edge Computing (MEC) as a key ingredient of the next generation networks moves intelligence (computing, storage and networking) close to the users, devices and applications and thus satisfying the requirements for high bandwidth and low latency [12], [13].

The MEC platform which provides an environment for running MEC applications and services may be co-located with distributed and virtualized core network functionality. The proposed new MEC service may use the capabilities of network exposure to enable publishing and access to presence information without the need to use IMS functionality. Further, the proposed service is enhanced with information provided by network elements about the presence supplier status such as handover status, used bearers, positioning data etc., available through subscription/notification from other MEC services.

III. Presence Service Standards

The presence service both as concept and as practice is in use by Internet world for long time but it is kept with no interoperability in mind and with a high level of fragmentation [14]–[17]. If one has to put the interoperability as a leading principle when specifying presence service then the network convergence, i.e. fixed, mobile, and wireless, while interacting with external ones, may form a different set of requirements than the regular one with respect to the information attributes as location information, multi-media components, etc.

The aim to provide common view on concepts, entities involved and their basic functions, and to enable interoperability between different Presence systems drives the definition of Presence service by IETF [18], [19]. The service has two types of clients, clients that supply (publish) presence information (called presentities) and clients that consume presence information from the service (called watchers). Watchers may fetch the current value of published presence information or may subscribe to receive notifications about changes in values of presence information. The service also provides watcher information which may be accessed by a simple request or distributed to subscribers via notifications. The IETF presence service model defines the format of presence information as a container of arbitrary number of elements. Each element consists of a status marker (such as offline/busy/online/do not disturb, etc.) and optional address for communication including a method for communications (e.g. telephony, instant messaging), and an address to contact via communication means (e.g. telephone number, the address of instant messaging box). The access rules, defined by the presentity, constrain the presence information availability to watchers, and the visibility rules which determine constraints on watcher information available.

3GPP defines a set of requirements to support the presence service, applicable to home environment (services in the home network), network manufacturers and devices [20]. The home environment may act as presentities (to supply presence information) and as watchers (to consume presence information). The requirements to home environment for the Presence service support include general requirements, management requirements for access control to presence information, for supplying of presence information and requesting presence data, and requirements for notifications about monitoring requests and presence data modification, and acknowledgments. The presentity may configure the necessity of authorization before providing presence information.

The protocol details for Presence service provisioning in IMS (Internet Protocol Multimedia Subsystem), based on Session Initiation Protocol (SIP) and Presence event package, are provided in [21]. SIP and XML Configuration Access Protocol provide methods for presence status management.

The open access to presence management for third party applications may be provided by Open Service Access (OSA) Application Programming Interfaces (APIs) [22]. The OSA Presence and Availability Management (PAM) service capability feature is defined by sequence diagrams, class diagrams, interface specification including detailed description of methods, and data definitions. The OSA PAM specifications make difference between presence and availability. While the presence includes static information about user identities, and dynamic information about user activity status and contextual information such as disposition, the availability denotes the user willingness and availability to share information and to communicate with others and enables privacy control. Thus, presence is required for availability, but does not imply availability to all watchers. The OSA API provides low level of network functionality abstraction, methods are asynchronous relying on a large set of data types and require a detailed control of presence information management.

Parlay X Presence web service provides a greater level of abstraction of presence information management with methods that are simple in operation, with plain semantics and use synchronous invocations [23]. In addition to the capabilities of presence information supplying and consuming, and monitoring on watcher information, the Parlay X presence web service can directly communicate with IMS presence network elements using SIP interface. The Parlay X Presence interfaces are defined using WSDL (Web Services Definition Language) which is XML based and use the Simple Object Access Protocol (SOAP) to access web services via HTTP.

REST (Representational State Transfer) is a client-server architectural style, which uses a resource representation to transfer the resource state at the server into the resource state at the client application. The interactions are based on HTTP methods and REST APIs make it easier to create complex queries. While SOAP uses web service interfaces to reveal its functionality to applications, REST uses Uniform Resource Identifiers (URIs) to access resources. SOAP needs more bandwidth while REST is more frugal in this regard.

In this paper, we propose a RESTful web service for management of presence information which may be deployed

at the network edge. Presence information at the network edge besides for a real subscriber, might be published in favor of an application or service, say highway jam information, 'flashing news' distribution, local advertisements, location of like-minded users in the vicinity, playing and/or content sharing, etc.

IV. Detailed Service Description

The proposed Presence and Availability Service (PAS) allows for presence information to be published by a user and to be obtained by others. The typical client of PAS interfaces is either a presentity or a watcher of the presence information. Using the PAS interfaces the user's, say Martin's presence application publishes a set of attributes reflecting both his wish to be available, or not, for communication, and his willingness to do so. The presence status of Martin may aggregate both static and dynamic information provided by the network and by himself. Part of the dynamic information might be fetched from different sources (e.g. device attach/detach procedures, keying activity, handover procedures etc.). A fixed telephone number is a trivial illustration for the static information could look like. Another type of presence information, provided by the network, might also be based on user's patterns. More details on how this information is captured are provided later in this section. Martin's privacy and confidentiality protection might be obtained by controlling over whether any group of watchers, here Eve, Lily, and Nick, are allowed to fetch his presence information. Moreover, Martin might define different access levels for different watchers, e.g. Eve is allowed to access the entire presence information of Martin, Lily can fetch partly, and Nick is not allowed at all. Thus, Martin defines (watcher per watcher) his presence information exposure at the level of granularity of his own will, and he might limit the access of specific watchers down to the level he wants, or to none. In fact, Martin might decide to make his presence information look in a multi-facet way and formulating access rules in such way that some watchers are provided with different information (e.g. Martin is not available for Lily, but he is available for others).

In case Eve, Lily, and Nick (e.g. Martin's call parties or chat peers) need to know Martin's wish and ability to be in touch might do so by fetching his presence status at the moment, stored by the PAS. Thus, they are Martin's watchers. In addition to request for presence status on demand, the watcher may subscribe for notifications about changes in the Martin's presence status. Both the request and subscription have to be authorized by the presentity.

Using the PAS APIs, the presentity may observe watcher information by retrieving data about his/her current watchers and/or by starting/stopping notifications regarding watchers.

The PAS enables watcher authorization by the presentity. Fig. 1 shows the flow of authorizing a new watcher request. First the presentity application starts watcher notifications and upon a request for presence information from the watcher application, the PAS notifies the presentity application, which updates the authorization rules, and the PAS provides the requested information to the watcher application.

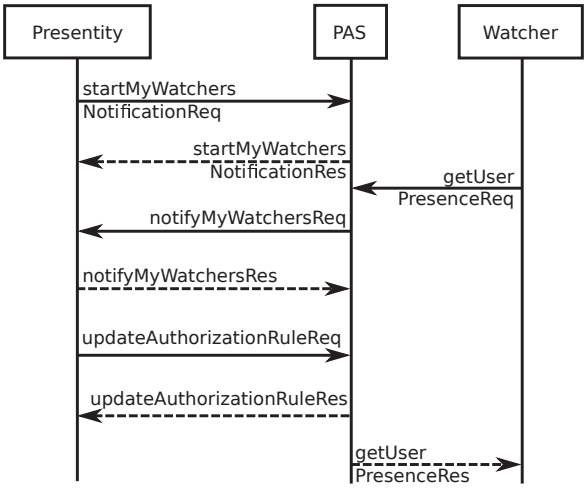

Fig. 1. Presentity authorizes a new watcher request

Fig. 2 shows the flow of publishing of presence information by a presentity application and updating an existing authorization rule. The presentity application publishes user status and retrieves a list of watchers of the presentity status. Next the application gets attributes of an existing watcher subscription and updates his/her authorization rules.

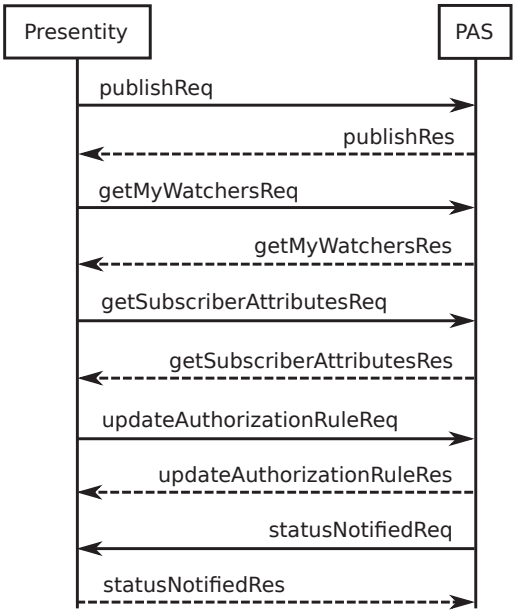

Fig. 2. Presentity publishes presence data and updates existing watcher authorization

Fig. 3 shows the flow of authorizing a new watcher subscription. The presentity application starts watcher notifications. Upon a request for subscription to presence information the PAS notifies the presentity applications, which approves the

subscription by updating the authorization rules and the PAS acknowledges the subscription request of the watcher.

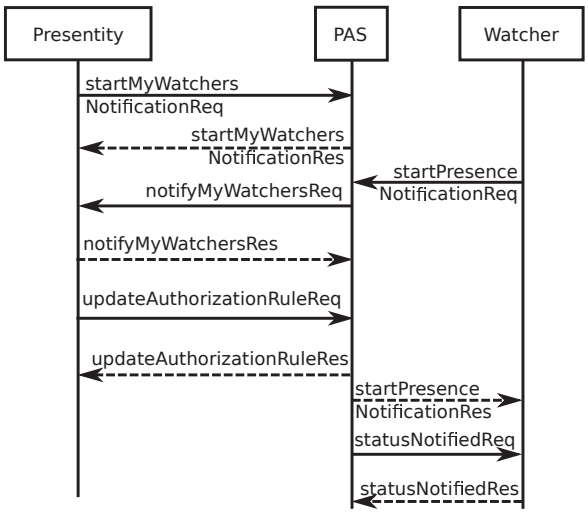

Fig. 3. Presentity authorizes a new watcher subscription

Fig. 4 shows the flow of notifying about presence status change, where having an active subscription, the watcher application is notified upon updates in the presence information.

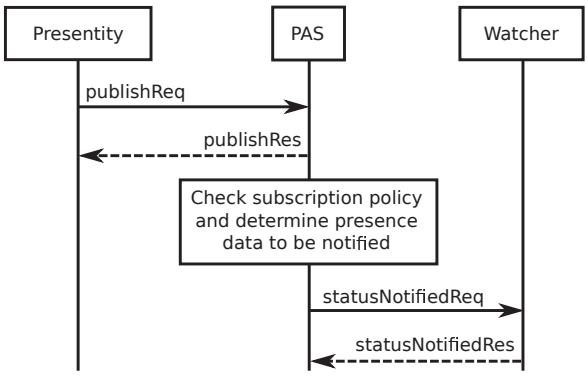

Fig. 4. Watcher is notified upon presence status change

Either the watcher or the presentity may terminate an existing presence status subscription and watcher information subscription as shown in Fig. 5 and Fig. 6, respectively.

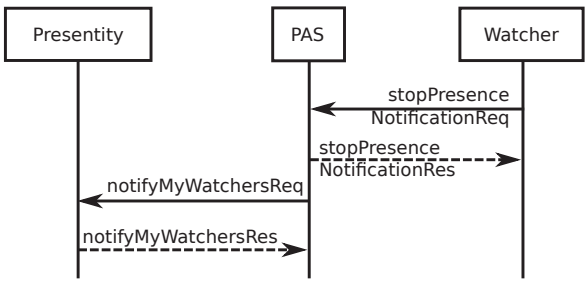

Fig. 5. Watcher terminates presence notifications

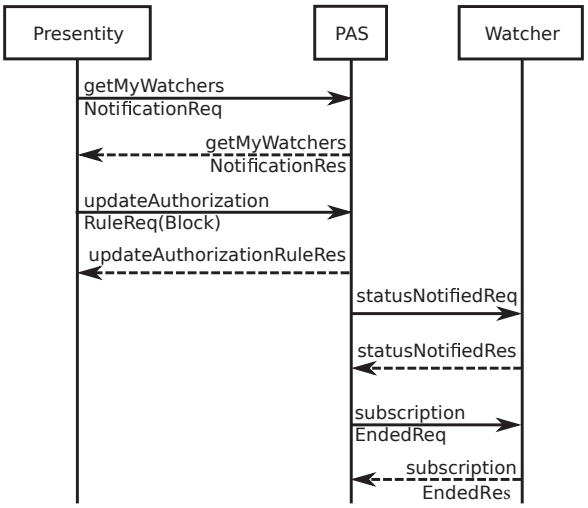

Fig. 6. Presentity terminates an existing watcher subscription

In the latter case, when the presentity application requests termination of subscription for presence information by updating the respective authorization rule, the PAS notifies the watcher application about the ended subscription.

In case the presentity wishes to publish extended presence information gathered from network elements it indicates to the PAS its intention. The PAS in turn subscribes to receive notifications about user location changes with the standardized MEC Location Service, and notifications about user mobility and activity with the standardized MEC Radio Network Information service (RNIS). Fig. 7 shows the flow of enabling notifications from the network about user location and mobility. First, the presentity application using the PAS interfaces enables notifications indicating the notification types (location and mobility). Next, the PAS subscribes to receive notifications from Location Service and RNIS and having an active subscription the PAS receives notifications when the user changes his/her location and when he/she changes the serving cell.

When the presentity does not want to publish presence information from the network elements he/she disables notifications.

V. Service Resource Model

REST uses the concept of resource representing a physical or logical entity. Service resources are organized in a tree structure, where each uniquely identified resource follows the service root URI that may be found in a service directory. This section describes the PAS resources and applicable methods which are part of PAS API definition.

The PAS resources are grouped for watchers and presentities.

Fig. 8 shows the resource structure for watchers. The watchers resource represents information for all watchers of presence data and has two sub-resources: subscriptions and presenceInfoRequests.

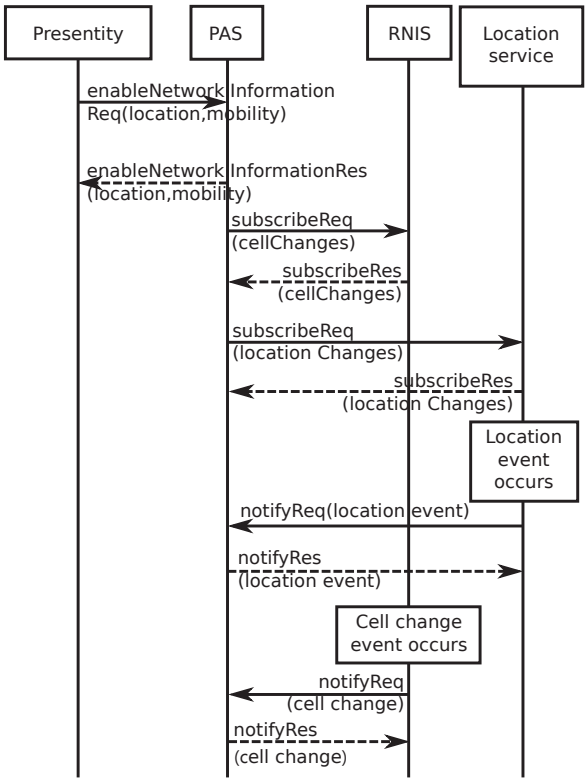

Fig. 7. Receiving presence information from the network

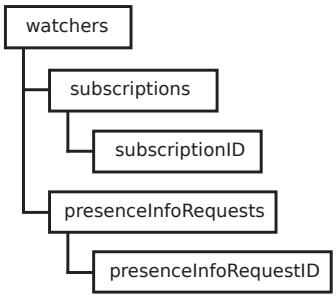

Fig. 8. Resource structure for watchers

The subscriptions resource represents information for all watcher subscriptions for notifications about changes in presence data, and the subscriptionID represents an existing watcher subscription. The presenceInfoRequests resource represents all requests for presence data issued by watchers, while the presenceInfoRequestID represents an existing request. When a presentity application wants to retrieve information about all watcher subscriptions it sends an HTTP GET method to the subscriptions resource and as a result the PAS answers with 200 OK response where the message body contains the list of watcher subscriptions. When a watcher application wants to create a new subscription for presence data, it sends an HTTP POST method to the subscriptions resource, and the PAS creates a subscription and returns 201 Created response with the address of the created subscriptionID resource. An existing subscriptionID resource may be updated using an HTTP PUT method and also may be deleted using an HTTP DELETE method.

The presenceInfoRequests resource represents information about all requests for presence data issued by watchers, and the presenceInfoRequestID resource represents an existing watcher's presence request. The HTTP GET method on the presenceInfoRequests resource or on the presenceInfoRequestID retrieves a list of all watcher's requests or information about existing watcher's request, respectively. An HTTP POST method invoked on the presenceInfoRequests resource creates a new request for presence data, while an HTTP PUT or DELETE method, changes or deletes an existing watcher's request.

Fig. 9 shows the resource structure for presentities. The presentities resource represents information for all presentities and has six sub-resources: publishInfoRequests, myWatcherNotificationRequests, getMyWatcherRequests, subscriptionEndedRequests, updateAuthorizationRuleRequests, and networkInfoRequests.

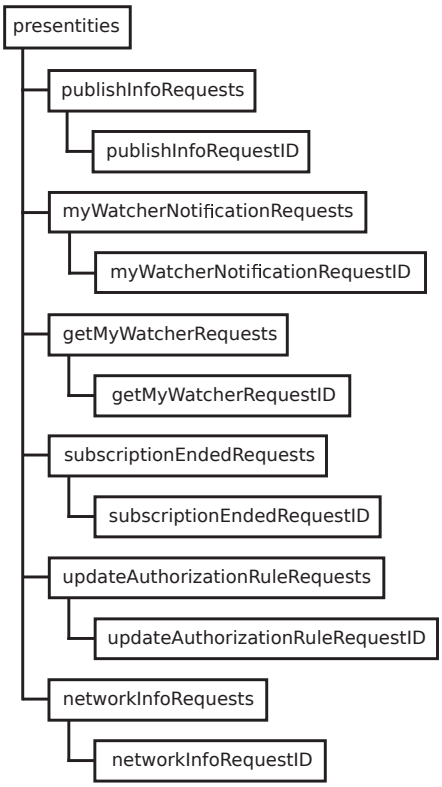

Fig. 9. Resource structure for presentities

The publishInfoRequests resource is a container for all publishInfoRequestID resources each of which represents an existing request for publishing presence information by a presentity. An invocation of HTTP POST method on the publishInfoRequests resource creates a new resource representing a request for publishing presence information, while an invocation of HTTP PUT or DELETE method on the pub-

lishInfoRequestID resource, updates or deletes the resource, respectively.

The myWatcherNotificationsRequests resource represents all requests issued by presentities for notifications about their watchers, and the myWatcherNotificationsRequestID resource represents an existing request for notifications about watcher information. To enable notifications about watchers, the presentity application sends an HTTP POST request to the myWatcherNotificationsRequests resource. The application updates or disables an existing notification about watcher information changes by sending a HTTP PUT or DELETE method to the myWatcherNotificationsRequestID.

The getMyWatcherRequests resource represents all requests for watcher information issued by presentities, and its sub-resource getMyWatcherRequestID resource represents an existing request for watcher information.

The subscriptionEndedRequests resource stands for all requests sent by presentities to terminate an existing watcher subscription. Its sub-resource subscriptionEndedRequestID serves for a particular request.

The updateAuthorizationRuleRequests resource consists of all resources representing presentity's requests for updating authorization rules each of which is denoted by updateAuthorizationRuleRequestID resource.

The networkInfoRequests resource is a container for all requests issued by presentities to enable enrichment of published presence data with information provided by the network elements. Both radio access network and core network elements may supply user information. The paper describes the way PAS service may receive user information from existing MEC services, but with deployment of distributed core network functionality, Network Exposure Function may expose another user information (e.g. user registration status). To enable specific notifications from the network element, the presentity application sends an HTTP POST method to the networkInfoRequests resource and the PAS returns the identification of the newly created networkInfoRequestID resource.

Table I summarizes the PAS resources and applicable HTTP methods.

TABLE I. PAS RESOURCES AND APPLICABLE METHODS

Resource URI	**HTTP method**	**Meaning**
/watchers /subscriptions	GET	Retrieves list of all subscriptions for changes in presence information.
	POST	Creates a new subscription.
/watchers /subscriptions /subscriptionID	GET	Retrieves information about existing subscription for notifications on presence info changes.
	PUT	Modifies existing subscription.
	DELETE	Terminate existing subscription.
/watchers /presenceInfoRequests	GET	Retrieves list of all watcher requests for presence info
	POST	Creates a new request for presence info
/watchers /presenceInfoRequests /presenceInfoRequestID	GET	Retrieves information about existing watcher's request.
	PUT	Updates existing request.
	DELETE	Deletes existing request.
/presentities /publishInfoRequests	GET	Retrieves list of all requests issued by presentities for presence publishing.
	POST	Creates a new request for presence publishing.
/presentities /publishInfoRequests /publishInfoRequestID	GET	Retrieves information about existing request.
	PUT	Updates existing request.
	DELETE	Deletes existing request.
/presentities /myWatcherNotification Requests	GET	Retrieves list of all requests issued by presentities for notifications on watcher info changes.
	POST	Creates a new request for enabling notifications.
/presentities/myWatcher NotificationRequests /myWatcherNotification RequestID	GET	Retrieves information about existing request.
	PUT	Updates existing request.
	DELETE	Deletes existing request.
/presentities /getMyWatcherRequests	GET	Retrieves list of all requests for watcher information.
	POST	Creates a new request for watcher information.
/presentities/getMy WatcherRequests/get MyWatcherRequestsID	GET	Retrieves information about existing request.
	PUT	Updates existing request.
	DELETE	Deletes existing request.
/presentities /subscriptionEnded Requests	GET	Retrieves list of all requests for subscription termination.
	POST	Creates a new request for subscription termination.
/presentities /subscriptionEnded Requests/subscription EndedRequestID	GET	Retrieves information about existing request.
	PUT	Updates existing request.
	DELETE	Deletes existing request.
/presentities /updateAuthorization RuleRequests	GET	Retrieves list of all requests for authorization rule update issued by presentities.
	POST	Creates a new request for authorization rule update.
/presentities /networkInfoRequests /networkInfoRequestID	GET	Retrieves information about existing request.
	PUT	Updates existing request.
	DELETE	Deletes existing request.
/presentities/update AuthorizationRule Requests/updateAuthor izationRuleRequestsID	GET	Retrieves information about existing request.
	PUT	Updates existing request.
	DELETE	Deletes existing request.

VI. RESOURCE, SUBSCRIPTION AND NOTIFICATION DATA TYPES

The published presence information is a set of attributes that describe the presentity status as current disposition, activity, communication means, contacts. Ideas for attributes characterizing the presence status are provided in [19], [22]. Some attributes may have multiple values.

The "activity" attribute describes current user activity, and the user may be involved in a multiple activity simultaneously. The "device" attribute references the devices that provide a particular service. The "mood" attribute stands for presentity's mood. The "place-is" attribute provides some information about the presentity's environment and the "place-type" attribute describes the type of the place where currently is the presentity. The "privacy-type" attribute indicates the level of privacy the presentity currently has. The "validity" attribute describes the time boundary beyond which the presence information is not valid.

TABLE I. (CONT.)

Resource URI	HTTP method	Meaning
/presentities /networkInfo Requests	GET	Retrieves list of all requests to enable publishing of presence information supplied by network elements.
	POST	Creates a new request to enable publishing of presence information supplied by network elements.

An example of presence status description in JSON format is as follows:

```
{ "status": {
   "activity":["vocation","traveling"],
   "device":["phone","tablet"],
   "mood":"happy",
   "place-is":"noisy",
   "place-type":"car"
   }
}
```

The subscription data type represents a subscription for presence notifications. It is a structure that contains the URI of the watcher who wants to gain access to presence data, URIs of observed presentities, the attributes the watcher wants to access, the watcher application identifier, maximum frequency of notifications, subscription expiry deadline and the URI selected by the watcher to receive notifications on subscribed presence information.

The presenceNotification data type represents a notification with regards to presence information change. Its attributes include identifiers to associate the presence related event, time stamp and presence status.

The subscriptionEnded data type is used to notify the watcher about subscription termination and its attributes include the watcher URI whose subscription is terminated, the owner of the presence information to which the terminated subscription refers to, and the reason (blocked, give up, probation, timeout, deactivated).

The watcherSubscriptionStatus data type indicates the status of the watcher subscription to presence information and it is an enumerated type with possible values of authorized (by the presentity), blocked (currently by the presentity), polite blocked, pending (waiting for authorization decision), active (approved by the presentity), waiting (timeout occurs while waiting for authorization), and terminated.

The watcherInfo data type is used by the presentity to request information about watchers that want to subscribe or are subscribed to his/her presence data. Its attributes are the URI of the presentity who wants watcher information and the watcherSubscriptionStatus data type.

The authorizationValue data type represents the authorization actions applied to the watcher by the presentity. Possible values are “block” (the watcher is not allowed to access presentity’s data), “confirm” (the watcher has to wait for presentity input), “politeBlock” (the presentity blocks the watcher but this fact is not revealed to the watcher and the watcher receives limited or no presence information) and “allow” (the watcher is allowed to access the presence data).

The authorizationRule data type is used to update an authorization rule. Its attributes indicate the presentity URI, the rule ID, the watcher URIs, authorization value and the respective presence attributes.

The watcherSubscriptionData data type represents a subscription to notifications for the status of watcher subscription. It allows the presentity to filter watchers whose information the presenty is interested in. Its attributes include the presentity URI, the watcherSubscriptionStatus as described above and the subscription duration.

VII. Presence Subscription Models

The PAS needs to be aware about the watcher subscription status to notify the watcher about presence information change. This status has to be synchronized with the watcher’s application view on presence subscription status.

Fig. 10 shows the simplified watchers view on subscription status for presence information.

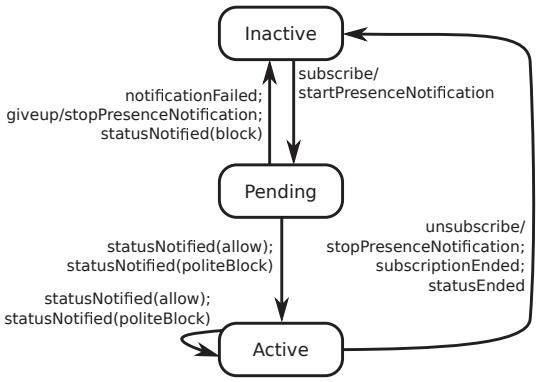

Fig. 10. Watcher’s subscription status as seen by the watcher application

In Inactive state, the presence subscription does not exist, and the PAS does not send any notifications about presence information change. When the watcher decides to subscribe to published presentity’s data, his/her watcher application starts presence notifications, and the subscription status becomes Pending. In Pending state, the watcher application waits for subscription authorization. The watcher may give up a pending subscription or a timeout may occur (notificationFailed). The presentity’s authorization decision may be one of the following (allow, block or polite block). If the authorization decision is polite block, the presentity has blocked the watcher, but this information is not revealed to the watcher and he/she receives none or limited presence information. So, from watcher point of view there is no difference between active subscription and politely blocked subscription. In Active state, the subscription is authorized, and the PAS calls statusNotified operation to notify the watcher application about the requested presence changes. When the subscription expires (statusEnd operation),

or the presentity decides to terminate the subscription (subscriptionEnded operation) the watcher application is notified. When the watcher does not want to receive any notifications about presentity's status, his/her application stops presence notifications. In Active state, the presentity may update the watcher subscription, the watcher may unsubscribe for receiving presence notifications, or the presentity may cancel the subscription, or the subscription may expire.

From the PAS point of view the watcher subscription status may be in Terminated, Pending, Waiting, Active or PoliteBlocked state. Fig. 11 shows the simplified PAS view on the presence subscription status of a watcher.

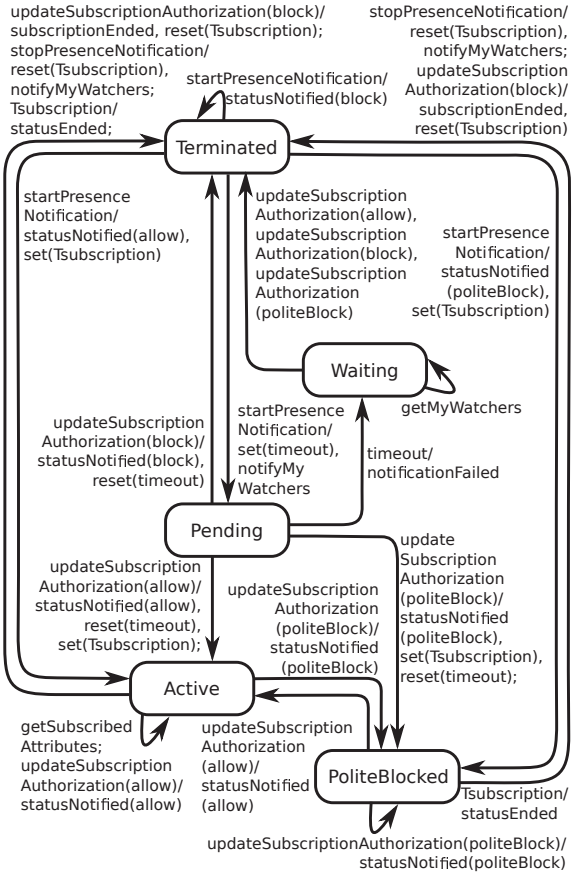

Fig. 11. Watcher's subscription status as seen by the service

The Terminated state indicates that the subscription is terminated or currently blocked and the PAS does not send any notifications to the watcher application about presence changes. If a presence subscription request is received and there is an authorization policy, the watcher application is notified. In Pending state, the presence subscription waits for authorization decision by the presentity. In Pending state, the subscription request may time out, or a watcher may give up. The Active state indicates active watcher subscription authorized by the presentity, and in this state the PAS notifies the watcher application about presence changes (not shown in Fig. 11). The Tsubscription timer indicates the watcher subscription duration. In Waiting state, the watcher subscription has expired while waiting for presentity authorization. In PoliteBlocked state, the watcher subscription is politely blocked by the presentity. In Active state and in PoliteBlock state, the presentity may change some subscription attributes or may change the subscription status. It is also possible, the watcher to stop presence notifications.

We use formal model representation as finite state machines to mathematically prove that both views on the watcher's subscription status are synchronized. A finite state machine (FSM) is represented by a set of states, a set of events, a set of transitions and an initial state.

By $FSM_w = \left(S_w, E_w, T_w, s_w^0\right)$ it is denoted an FSM modeling the watcher's view on the subscription status, where the names of states and events are given in brackets:

$$S_w = \{Inactive\,[s_1^w], Pending\,[s_2^w], Active\,[s_3^w]\};$$
$$E_w = \{subscribe\,[e_1^w], notificationFailed\,[e_2^w], giveup\,[e_3^w], statusNotified\,(block)\,[e_4^w], statusNotified\,(allow)\,[e_5^w], statusNotified\,(politeBlock)\,[e_6^w], subscriptionEnded\,[e_7^w], statusEnd\,[e_8^w], unsubscribe\,[e_9^w]\};$$
$$T_w = \{(s_1^w e_1^w s_2^w), (s_2^w e_2^w s_1^w), (s_2^w e_3^w s_1^w), (s_2^w e_4^w s_1^w), (s_2^w e_5^w s_3^w), (s_2^w e_6^w s_3^w), (s_3^w e_5^w s_3^w), (s_3^w e_6^w s_3^w), (s_3^w e_7^w s_1^w), (s_3^w e_8^w s_1^w), (s_3^w e_9^w s_1^w)\};$$
$$s_w^0 = \{s_1^w\}.$$

By $FSM_s = \left(S_s, E_s, T_s, s_s^0\right)$ it is denoted an FSM modeling the PAS's view on the subscription status, where:

$$S_s = \{Terminated\,[s_1^s], Pending\,[s_2^s], Active\,[s_3^s], Waiting\,[s_4^s], PoliteBlocked\,[s_5^s]\};$$
$$E_s = \{startPresenceNotification\,[e_1^s], updateSubscriptionAuthorization\,(allow)\,[e_2^s], updateSubscriptionAuthorization\,(politeBlock)\,[e_3^s], updateSubscriptionAuthorization\,(block)\,[e_4^s], stopPresenceNotification\,[e_5^s], getSubscribedAttributes\,[e_6^s], Tsubscription\,[e_7^s], timeout\,[e_8^s], getMyWatchers\,[e_9^s]\};$$
$$T_s = \{(s_1^s e_1^s s_1^s), (s_1^s e_1^s s_2^s), (s_1^s e_2^s s_3^s), (s_1^s e_3^s s_5^s), (s_2^s e_3^s s_1^s), (s_2^s e_2^s s_3^s), (s_2^s e_3^s s_5^s), (s_2^s e_8^s s_4^s), (s_2^s e_5^s s_1^s), (s_4^s e_9^s s_4^s), (s_4^s e_4^s s_1^s), (s_4^s e_3^s s_1^s), (s_3^s e_2^s s_3^s), (s_3^s e_6^s s_3^s), (s_3^s e_3^s s_5^s), (s_3^s e_4^s s_1^s), (s_3^s e_5^s s_1^s), (s_3^s e_7^s s_1^s), (s_5^s e_3^s s_5^s), (s_5^s e_2^s s_3^s), (s_5^s e_4^s s_1^s), (s_5^s e_5^s s_1^s), (s_5^s e_7^s s_3^s)\};$$
$$s_s^0 = \{s_1^s\}.$$

By definition, two FSMs have a weak bisimulation relationship between each other if there are tuples of states in FSMs, where for each state tuple there exist a transition sequence in the first FSM and a respective transition sequence in the second FSM.

Proposition 1. *FSM_w and FSM_s have weak bisimulation relationship.*

Proof: Let $R = \{(s_1^w, s_1^s), (s_2^w, s_2^s), (s_3^w, s_3^s), (s_3^w, s_5^s)\}$ is a set of state tuples. Then, the following transitions in each FSM start and terminate in the states in R:

1) The watcher sends a presence subscription request and waits for presentity's confirmation: $\forall (s_1^w e_1^w s_2^w) \exists (s_1^s e_1^s s_2^s)$.
2) The watcher sends a presence subscription request and there is a blocking policy: $\forall (s_1^w e_1^w s_2^w) \sqcap (s_2^w e_4^w s_1^w) \exists (s_1^s e_1^s s_1^s)$.
3) The watcher sends a presence subscription request and there is an enabling policy: $\forall (s_1^w e_1^w s_2^w) \sqcap (s_2^w e_5^w s_3^w) \exists (s_1^s e_1^s s_3^s)$.
4) The watcher sends a presence subscription request and there is a policy for polite blocking: $\forall (s_1^w e_1^w s_2^w) \sqcap (s_2^w e_6^w s_3^w) \exists (s_1^s e_1^s s_5^s)$.
5) The watcher gives up a pending subscription request: $\forall (s_2^w e_3^w s_1^w) \exists (s_2^s e_5^s s_1^s)$.
6) The presentity allows a pending presence subscription: $\forall (s_2^w e_4^w s_3^w) \exists (s_2^s e_2^s s_3^s)$.
7) The presentity blocks a pending presence subscription: $\forall (s_2^w e_4^w s_1^w) \exists (s_2^s e_4^s s_1^s)$.
8) The presentity politely blocks a pending presence subscription: $\forall (s_2^w e_2^w s_1^w) \exists (s_2^s e_3^s s_5^s)$.
9) The watcher's subscription request has expired while waiting for authorization decision. Later, the presentity defines an authorization policy for the presence subscription: $\forall (s_2^w e_2^w s_1^w) \exists ((s_2^s e_8^s s_4^s) \sqcup (s_4^s e_2^s s_1^s)) \sqcap ((s_2^s e_8^s s_4^s) \sqcup (s_4^s e_3^s s_1^s)) \sqcap ((s_2^s e_8^s s_4^s) \sqcup (s_4^s e_4^s s_1^s))$.
10) The presentity updates attributes of active or politely blocked presence subscription: $\forall ((s_3^s e_5^s s_3^s) \sqcup (s_3^s e_6^s s_3^s)) \exists ((s_3^s e_6^s s_3^s) \sqcup (s_3^s e_2^s s_3^s)) \sqcap ((s_5^s e_6^s s_5^s) \sqcup (s_5^s e_5^s s_5^s))$.
11) The presentity updates status of active or polite blocked presence subscription: $\forall ((s_3^s e_5^s s_3^s) \sqcup (s_3^s e_6^s s_3^s)) \exists ((s_3^s e_6^s s_3^s) \sqcup (s_3^s e_3^s s_5^s)) \sqcap ((s_5^s e_6^s s_5^s) \sqcup (s_5^s e_2^s s_3^s))$.
12) The presentity terminates an active or politely blocked presence subscription: $\forall (s_3^s e_7^s s_1^s) \exists (s_3^s e_4^s s_1^s) \sqcup (s_5^s e_4^s s_1^s)$.
13) The presence subscription expires: $\forall (s_3^s e_8^s s_1^s) \exists (s_3^s e_7^s s_1^s) \sqcup (s_5^s e_7^s s_1^s)$.
14) The watcher terminates his/her presence subscription: $\forall (s_3^s e_9^s s_1^s) \exists (s_3^s e_5^s s_1^s) \sqcup (s_5^s e_5^s s_1^s)$.

Therefore, FSM_w and FSM_s are weakly bisimilar. ■

VIII. Conclusion

The paper presents an approach to define a RESTful service that enables supplying presence information and observing the published information. The service also allows the presentity to authorize and monitor his/her watchers who observe the presentity information. In addition to standardized presence service, the proposed approach enables publishing and additional information about presentity activity, mobility and location gathered from network elements. The main benefits of the proposed Presence and Availability API is that the API deployment does not require IMS infrastructure at all, each presence application with IP connectivity may use the PAS API. Next, the approach features inherent MEC benefits related to reduced latency and saving transmission resources from the edge to the core. The API may be used in the world of Internet of Things for innovative solutions in smart cities, smart homes, intelligent transportation systems, and ambient assisted living. Examples include smart devices equipped by air quality sensors that publish data about exceeding the permissible thresholds in close vicinity to watchers, and context aware applications which are more user adaptive.

Acknowledgment

The research is part of project grant KP-06-H37/33, funded by Bulgarian National Science Fund.

References

[1] Ye Yang, Jiangtao Luo, Jin Peng and Jian Huang, "Research and implementation on PRESENCE service of IMS," 2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), Beijing, China, 2010, pp. 803-806, doi: 10.1109/ICBNMT.2010.5705201.

[2] Daeseung Yoo, Jinkyu Choi, Byungtae Jang and Soonghwan Ro, "A Study of Presence Services in SIP-Based Group Communication Systems," Future Information Technology, Application, and Service, 2012, pp. 301-308, doi: 10.1007/978-94-007-5064-7_42.

[3] Patricia E. Figueroa and Jesús A. Pérez, "Architecture for Interoperability between Instant Messaging and Presence Protocols," Networked Digital Technologies, 2010, pp. 306-320, doi: 10.1007/978-3-642-14306-9_32.

[4] K. Imran and T. Jensen, "Performance of Parallel Signaling between IMS Presence Server and Web Services," 2011 Sixth IEEE International Symposium on Electronic Design, Test and Application, Queenstown, New Zealand, 2011, pp. 248-253, doi: 10.1109/DELTA.2011.52.

[5] Andrés Garcia, José Santa, Antonio Moragón and Antonio F. Gómez-Skarmeta, "IMS and Presence Service Integration on Intelligent Transportation Systems for Future Services," Advances in Computing and Communications, 2011, pp. 664-675, doi: 10.1007/978-3-642-22720-2_70.

[6] Kun-Che Hsu and Jenq-Shiou Leu, "Improving the efficiency of presence service in IMS by JSON," 2015 Seventh International Conference on Ubiquitous and Future Networks, Sapporo, Japan, 2015, pp. 547-550, doi: 10.1109/ICUFN.2015.7182604.

[7] David Petras, Ivan Baronak, Erik Chromy, "Presence Service in IMS," The Scientific World Journal, vol. 2013, Article ID 606790, 8 pages, 2013, doi: 10.1155/2013/606790.

[8] M. Happenhofer, C. Egger, F. Wegscheider and R. Gabner, "Presence distribution network," 2012 International Conference on Selected Topics in Mobile and Wireless Networking, Avignon, France, 2012, pp. 107-112, doi: 10.1109/iCOST.2012.6271276.

[9] Ernesto García Davis and Anna Calveras Augé, "Presence-Based Architecture for Wireless Sensor Networks Using Publish/Subscribe Paradigm," Wired/Wireless Internet Communications, 2011, pp. 27-38, doi: 10.1007/978-3-642-21560-5_3.

[10] P. A. Moreno, M. E. Hernando, and E. J. Gómez, "AALUMO: A User Model Ontology for Ambient Assisted Living Services Supported in Next-Generation Networks," XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013, 2014, pp. 1217-1220, doi: 10.1007/978-3-319-00846-2_301.

[11] S. S. Chauhan, S. Yangui, R. H. Glitho and C. Wette, "A case study for a presence service in the cloud," 2016 7th International Conference on the Network of the Future (NOF), Buzios, Brazil, 2016, pp. 1-7, doi: 10.1109/NOF.2016.7810124.

[12] I. Atanasov, E. Pencheva, D. Velkova and V. Trifonov, "Programmability of Multi-Connectivity in 5G," 2020 26th Conference of Open Innovations Association (FRUCT), Yaroslavl, Russia, 2020, pp. 38-45, doi: 10.23919/FRUCT48808.2020.9087445.

[13] E. Pencheva, I. Atanasov and V. Vladislavov, “Mission Critical Messaging Using Multi-Access Edge Computing,” Cybernetics and Information Technology, 2019, vol. 19, no. 4, pp. 73-89, doi: 10.2478/cait-2019-0037.

[14] EnGenius, Presence service. [Online]. Available: https://docs.engenius.ai/whitepapers/presence-service.

[15] Cisco, *Microsoft Exchange for IM and Presence Service on Cisco Unified Communications Manager*, 2014, *Release 10.5(1)*.

[16] Idency, *BEMS in a Good Control and Good Proxy environment*, 2017, *Configuration Guide*.

[17] Oracle, *WebLogic Communication Services, Developer’s Guide, 2020*. [Online]. Available: https://docs.oracle.com/cd/E28280_01/doc.1111/e13807/toc.htm.

[18] M. Day, J. Rosenberg and H. Sugano, “A model for Presence and Instant Messaging,” IETF, Request for Comments (RFC 2778), February, 2000.

[19] H. Schulzrinne, P. Kyzivat, J. Rosenberg and V. Gurbani, “RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF),” IETF, Request for Comments (RFC 4480), July, 2006.

[20] 3GPP TS 22.141, “Presence Service; Stage 1,” Rel. 16, v16.0.0, July, 2020.

[21] 3GPP TS 23.141, “Presence Service; Architecture and functional description,” Rel. 16, v16.0.0, July, 2020.

[22] 3GPP TS 29.198-14, “Open Service Access; Application Programming Interface (API); Part 14: Presence and Availability Management (PAM); Service Capability Feature (SCF),” Rel. 9, v9.0.0, 2009.

[23] 3GPP TS 29.199-14, “Open Service Access (OSA); Parlay X Web Services; Part 14: Presence,” Rel. 9, v9.0.0, 2009.