An RT-based Policy Model for Converged Networks

San-Tsai Sun and Konstantin Beznosov*
{santsais,beznosov}@ece.ubc.ca

Laboratory for Education and Research in Secure Systems Engineering
University of British Columbia
Vancouver, Canada

Technical report LERSSE-TR-2010-001†

Last Modification Date: 2009/01/02
Revision: #21

Abstract

Technologies advanced in communication devices and wireless networks enable telecommunication network operators to provide rich personalized multimedia services. To attract potential customers and increase average revenue per customer, network operators will provide personalized services as differentiating factors in the near future. To accommodate the diversity and complexity of future networks, it is desirable to have an unified access management framework for supporting current and future network operations. As part of efforts in developing access management framework for a large telecommunication company in Canada, we developed policy models for current and future operation modes of converged networks. The proposed policy models are used as the basis for the specification of access control policies in a larger project of access management framework. The relations between elements in the proposed policy model are expressed formally using *RT* framework. Policy model use-cases are used to demonstrate how *RT* credentials and policies can be developed based on the proposed policy models.

*Contact author.
†This and other LERSSE publications can be found at lersse-dl.ece.ubc.ca

Contents

1 Introduction

Telecommunication network operators currently provide a set of voice and data services for fixed-line and wireless subscribers. Technologies advanced in wireless networks enable network operators to provide rich personalized multimedia services. For instance, the specification of an IP based-multimedia subsystem (IMS) was released in the fifth release of the 3GPP specification [3GP06]. One of the key contributions of this release was the IMS supports simultaneous delivery of multimedia services by allowing separate negotiation of QoS and other delivery parameters for each multimedia service requested by a subscriber [3GP03]. The IMS also supports interworking between public switched telephony networks (PSTN), analog and digital cellular networks, and the Internet domains. This last points allows a convergence of Internet and Telecommunications networks, hence the term "*converged networks*". An important consequence of improved interfaces between packet-switched telecom networks and the Internet is that externally hosted multimedia service providers (EMSPs) will be able to offer services to subscribers on any access network (i.e., cellular, fixed-line, DSL) through gateways provided by the IMS. It is anticipated that the availability of personalized multimedia services will become commodities.

Technologies advanced in communication devices is another enabling factor for personalized services from converged networks. for instance, future mobiles phones and PDAs will typically have an integrated Assisted-GPS (A-GPS) chip that enable location based services. Location based services include services to identify a location of a person or object, such as discovering the nearest banking cash machine or personalized weather services and even location-based games. Physical presence information also allow network operators to deliver tailored personal services. For example, a network operator can provide personal call routing that routes voice calls automatically according to physical location (e.g., home, office or roaming), who is calling and urgent level settings.

To attract potential customers and increase average revenue per customer, network operators will provide personalized services such as personal call routing, presence enabled converged communication, or third-party service personalization/recommendation as differentiating factors in the near future. To accommodate the diversity and complexities of future networks, it is desirable to have an unified access management framework for current and future network infrastructures. As part of efforts in developing access management framework for a large telecommunication company in Canada, we developed policy models for current and future operation modes of converged networks.

In this paper, the term *policy model* refers to a set of inter-related elements that access control policies building upon. A well evaluated and validated policy model can serve as a valuable input during the policy-writing stage, and is of vital to the development of access management policies. The work developed in this paper is part of a larger project to develop and evaluate a policy-based access management framework (AMF) for facilitating the management of subscribers accesses to both voice and data services via converged networks. The proposed policy models are used as the basis for the specification of access control policies for AMF.

Converged networks are becoming increasedly complex, large-scale and heterogeneous. Tomorrow's networks will have to support millions of subscribers with diverse devices and service needs. Currently, policy-based network management system that separate policies from network components are commonly deployed to manage such network. Access control is achieved by mean of policies. Formal access control specification are authored using policy language such as Ponder [DDLS01], PDL [LBN99], DAP [KL03], or ECPL, and are deployed throughout the system to make network authorization decisions. Policy-based network management systems are generally effective. In order to specify correct and complete policies for the current and future converged networks, a well evaluated and validated policy model is required. We developed the proposed policy models based on data provided by the network operator, and validated through a series of case-study. Each case-study features a typical use-case scenarios the system has to support. The relations between elements in the proposed policy models are expressed formally using RT framework [LMW02].

RT framework is a family of role-based trust-management language for expressing policies in distributed environments. The main reason behind using RT as policy language is that for future operation mode, the access framework must allow subscribers to express their own credentials and access-control policies. The simplicity and expressiveness power of RT language are suitable for this purpose.

The rest of the paper is organized as follows. Section 2 explains the background of RT framework. Section 3 and Section 4 presents policy models for current and future operation mode respectively. Section 5 demonstrate how RT credentials and policies can be implemented based on the proposed policy models. Section 6 discuss conformance checking on service-plan policy model. Section 7 summarizes the paper and outlines future work.

2 Background on RT

In this work, RT [LMW02] is employed as the policy language for expressing access-control policies. RT framework is a family of role-based trust-management language for representing policies and credentials in distributed authorization. RT combines the strength of role-based access control (RBAC [SCFY96]) and trust-management (TM) systems [BFL96] to form a concise and expressive language.

Traditional access control mechanism make authorizations based on the identity of the request. However, in de-centralized or multicentric environments such as telecommunication networks, the resource owner and the request often are unknown to one another. In the "trust-management" approach, a requester submits a request to an authorizer, who specifies access rules (also called policies) to govern access to protected resources. During a resource request, the authorizer and the requester jointly compute a set of credentials (signed policy statements) to determine whether the request should be authorized. From TM, RT borrows principles of managing distributed authority through the use of credentials. From RBAC, it borrows the notions of role, interposed in the assignment of permissions to users to aid organizing those assignments, and of sessions and selective role activations.

All policy statements and credentials in RT take the form, $A.r \longleftarrow e$, where A

is an entity (also called principal), r is a role, and e is a role expression (a sequence of entities and roles). In this paper, we capitalized first character to denote entities, and use lower-case to represents roles. The terms *policy statement* and *credential*, and *entity* and *principal* are used interchangeably. The above credential $A.r \longleftarrow e$ means that $members(A.r) \supseteq members(e)$, i.e., e is a member of $A.r$. An entity in RT is a uniquely identified individual or process which can issue credentials and make requests, and a role is a set of entities who are members of this role. Entities in RT correspond to users in RBAC, and roles in RT can represent both roles and permissions from RBAC. For instance, company A issues a credential $A.manager \longleftarrow U$ to assign an entity U (user) to a *manager* role, and issues a policy $A.writeCheck \leftarrow A.manager$ to assign the role *manager* to anther role "*writeCheck*" (permission). Based on above credentials, one can conclude $A.writeCheck \longleftarrow U$, which means an entity U has "*writeCheck*" permission since U is in A's *manager* role. RBAC hierarchy can be also expressed by RT in a form of $A.r1 \longleftarrow A.r2$. For example, $A.employee \longleftarrow A.manager$ means *manager* role has every permission an *employee* role has. Delegation is expressed in form of $A.r \longleftarrow B.r$, which states that A delegate its authority to B over r.

RT framework includes RT_0, RT_1, RT_2, RT^T, and RT^D. RT_1 adds to RT_0 parameterized roles, which can express attribute fields. RT_2 adds to RT_1 logical objects, which can group logically related objects together so that permissions about them can be assigned together. RT^T provides manifold roles and role-product operators, which can express threshold and separation-of-duty policies. RT^D provides delegation of role activations, which can express selective use of capacities and delegation of these capacities. RT_0 is the most basic form of RT. There are four types of credentials in RT_0, each corresponding to a different way of defining role membership:

- Simple member $A.r \longleftarrow B$: defines an entity B to be the member of role r.
- Simple containment $A.r \longleftarrow B.r_1$: defines the role $A.r$ to contain (every identity that is a member of) the role of $B.r_1$.
- Linking containment $A.r \longleftarrow A.r_1.r_2$: defines $A.r$ to contain $B.r_2$ for every B that is a member of $A.r_1$.
- Intersection containment $A.r \longleftarrow B_1.r_1 \cap \cdots \cap B_k.r_k$: defines $A.r$ to contain the intersection of all the roles $B_1.r_1, \cdots, B_k.r_k$.

A *role expression* e in a RT policy statement $A.r \longleftarrow e$ can be a principal, a role, a linked role, or an intersection. We say that each policy statement defines the role $A.r$. Given a set $\mathcal{P}$ of policy statements, we denote $principals(\mathcal{P})$ is the set of principals in $\mathcal{P}$, $names(\mathcal{P})$ is the set of role name in $\mathcal{P}$, and $roles(\mathcal{P}) = \{A.r |\ A \in principals(\mathcal{P}), r \in names(\mathcal{P})\}$.

3 Policy Model for Current Operation Mode

Currently, network operators provide data and voice services through fixed-line and wireless subscriptions. From access control point of view, the resources to be protected are network bandwidth, and the subjects that make requests to the resources

are subscribers. When a subscriber initiates a request session, the subscriber authenticates to the network operator using identities enrolled during registration process such as wireless phone number or broadband user account. Request authorizations are based on service plans that are purchased by the subscribers. As a common use case, a residential user *Alice* desires access to the Internet from her home. In this case, *Alice* is subscribed to a "basic" home internet plan. The network operator in turn assigns *Alice* the role "internet/basic", and a new user name/passowrd for configuring the communication device (i.e., router). Once properly configured, *Alice* begins a connection by initiating a "data" session from her home computer. As part of this session, *Alice* generates requests that need to be routed to the appropriate resource (i.e., the Internet). Before routing the requests to the Internet, her identity (in this case, username/password), would be used to retrieve the associated service plans, and subsequently, the corresponding roles would be activated for the current session. Service plans might change frequently for the purpose of marketing, but a user role typically has a few fixed service levels within a network operator. When a request is processing , *Alice*'s profile information and her location along with the associated roles are passed to a policy enforcement point (PEP). PEP authenticates the request in question, and determines whether the request should be granted. The authentication is performed using the identity sent by the communication device (i.e., router) and/or the location information (device MAC and IP addresses) that are transmitted as part of the request. Once authenticated, PEP solicits authorization decision by passing the request in question to a policy decision point (PDP). Authorization is performed based on the service-plan *Alice* has signed up for (the roles) and the context information of the request. Context information is meta-data for the request, such as request payloads or the time of day when the message arrives. Depending upon the role of the user (e.g., quality of service the user has purchased) and the network traffic, the request may or may not be allowed through. Once the request is completed, accounting is turned on in order to measure the users usage and finally generate the monthly bill.

Similar case-studies can be established for (1) organizational subscribers accessing the resources provided by the service network, and (2) home and organizational subscribers using the network to make voice calls. In the voice-call case, voice-based sessions would have to be considered, and the credentials would be the telephone number of a user.

3.1 Current Operation Mode Policy Model

Figure 1 illustrates the proposed policy model for the current operation mode in converged networks. The corresponding RT credentials for the relations between elements are discussed in Section 3.2.

In the current operation mode of a converged network, a *user* is a personal or business subscriber. A user enrolls a set of *identities* (e.g., phone number, user name/password) that could be used to represent the user. Each subscriber purchases one or more *service plans* in order to obtain services from the network operator. A service plan assures different quality levels of voice/data services. Once a service plan

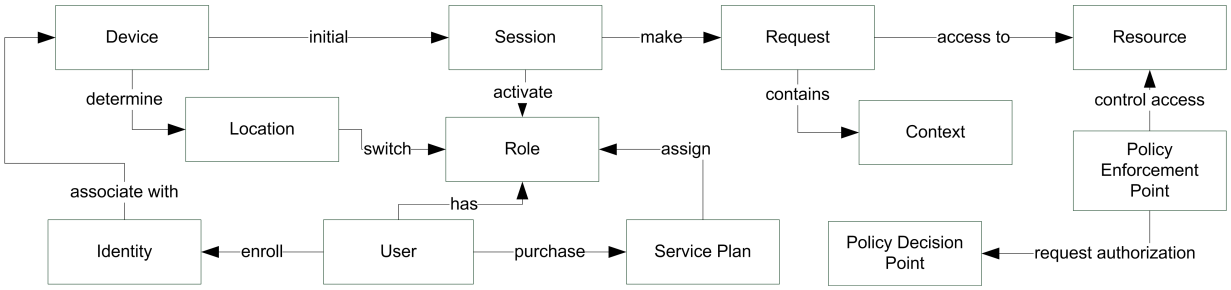

Figure 1: A policy model for the current operation mode in converged networks. Each arrowed line represents an action relation between two elements.

is purchased, a corresponding set of *roles* is assigned to the user. Before initiating a session, a user configures each identity obtained during enrollment to one or more communication *devices*. A *session* in a converged network could be either voice or data session. Based on the identity transmitted by the device and the subscribed service plans, a session automatically activate appropriate *roles* for the current session. The active roles in a session could be changed by user's current *location*, which could be determined by the device's physical location. Multiple *requests* can be made during a session. Each request contains the identity sent by the device, the associated roles, and the *context* information. Context information is the meta-data of a request, such as user-specific profile settings, usage statistics, request payload, or date/time of current request. A request is for a *resource* protected by a policy enforcement point (PEP). A PEP requests authorization decisions with respect to the request from a policy decision point (PDP) to enforce the decision. A PDP responds an authorization decision based on the request and a set of pre-defined access control rules.

3.2 Current Operation Mode RT Credentials

In this section, we formally express relations between elements in the proposed policy model using RT credentials. To clearly distinguish entities from roles in a *RT* credential, in this paper, we capitalize first character to denote entities, and use lowercase letter to denote role names—unless the role name is an acronym such as *ADSL*.

The relation that a user U enrolls a set of identities id_i from an enrollment entity E can be expressed as:

$$E.U \longleftarrow \bigcup_{i=1}^{n} U.id_i$$

For instance, the fact that *Alice* enrolls a mobile phone number and an account for ADSL connection can be expressed by two credentials: $E.Alice \longleftarrow Alice.mobilePhoneNo$ and $E.Alice \longleftarrow Alice.ADSLUserName$. These two credentials state that both $Alice.mobilePhoneNo$ and $Alice.ADSLUserName$ can represent $E.Alice$. $E.Alice$ is a subscriber defined by the enrollment entity E. The enrollment entity can also include other credentials issued by a trusted entity to represent $E.Alice$. For example, enrollment entity E could issue

$E.Alice \longleftarrow CA.userCert(Alice)$ to use a certificate issued by CA to speak for $E.Alice$. Entity CA might generate a public key for *Alice*'s certificate by issuing $CA.userCert(Alice) \longleftarrow PublicKey_{Alice}$.

A user purchases one or more service plans, and service plan authority S in turn assigns a set of roles r_i to a user U. The relation of service plan to role assignment can be specified by:

$$\bigcup_{i=1}^{n}(S.r_i \longleftarrow E.U)$$

where $E.U$ is a role definition credential issued by entity E for a user U. Assume *Alice* purchased "ADSL basic" and "Mobility premium" service plans, and S in turn assigns "ADSLBasic", "mobilePremium" roles to $E.Alice$. This scenario can be expressed by using $S.ADSLBasic \longleftarrow E.Alice$ and $S.mobilePremium \longleftarrow E.Alice$ credentials.

Before initialing a session, user U configures a set of identities id_i to a set of devices D_i, which can be specified as:

$$\bigcup_{i=1}^{n}(U.id_i \longleftarrow D_i)$$

To follow the example scenario above, *Alice* uses a cellular phone $Mobile_{Alice}$ and a router $Router_{Alice}$ for the subscribed services respectively. *Alice* configures mobile phone number $Alice.mobilePhoneNo$ to the phone $Mobile_{Alice}$, and setups the router using the user name/password obtained during enrollment process. This scenario can be represented by $Alice.mobilePhoneNo \longleftarrow Mobile_{Alice}$ and $Alice.ADSL \longleftarrow Router_{Alice}$. From the credential chain $E.Alice \longleftarrow Alice.mobilePhoneNo \longleftarrow Mobile_{Alice}$, $Mobile_{Alice}$ now has the same authority as $E.Alice$; in other words, $Mobile_{Alice}$ speaks for $E.Alice$.

The fact that a user initiates a session s_0 by using device D_i can be expressed using RT delegation credentials:

$$D_i \xrightarrow{D_i \text{ as } U.id_i} s_0$$

Delegation credentials in RT^D can be used to express user-to-session and process-to-process delegation of capability. An entity E activates the role $A.r$ to use in a session s_0 can be represented by $E \xrightarrow{E \text{ as } A.r} s_0$. We call $EasA.r$ a role activation. For instance, *Alice* initiates a "data" session from her computer to access Internet, which triggers a connection from the router to the network operator. This action can be expressed by $Router_{Alice} \xrightarrow{Router_{Alice} \text{ as } Alice.ADSL} s_0$.

A session s_0 activates an appropriate role $S.r_i$ and makes a request req based on the identity of the current session and the corresponding service plan can be expressed as:

$$s_0 \xrightarrow{s_0 \text{ as } S.r_i} req$$

Thus, the credential $s_0 \xrightarrow{s_0 \text{ as } S.ADSLBasic} internet$ represents an internet access request made by a session, in which $S.ADSLBasic$ is activated.

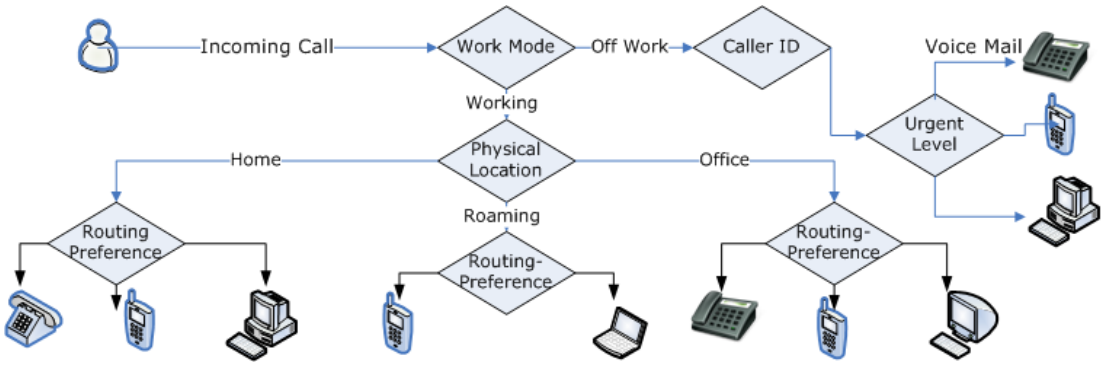

Figure 2: Decision tree of personal call routing scenario.

In addition to the delegation credentials discussed above, other entities might also issue some credentials during a resource request. For instance, an accounting entity A may issue a role activation credential $A \xrightarrow{A \text{ as } A.inGoodStanding} s_0$ to certify that the user of current session (i.e., *Alice*) is in good standing.

4 Policy Model for Future Operation Mode

For future operation mode (FOM), the resources under protection are a set of user profiles that enable personalized services such as address books, contact channels, usage behaviors, presence information, and preference settings. The subjects that make requests to the resources are other subscribers in the same or different network, as well as third-party solution providers. For personal and business subscribers, the same identities in current operation mode could be used in the authentication process. However, identities for third-party solution providers are required in order for subscribers to specify access policies and for network operator to enforce access control. In the future service scenarios, subscribers maintain ownership over their own profile data, and are able to specify access-control policies. The authorization decisions made by policy decision points (PDP) are based on access polices authored by subscribers.

4.1 Future Operation Mode Use Cases

Personal call routing: Alice starts work at home before office hours. PC knows Alice is away from office, it signs on to service network and fetches last work screen. Phone calls are routed to PC/home/mobile phone automatically depending on user setting and physical location. Full automatic transition of office experience to home with minimal user interaction. In the afternoon, Alice works in a cafe. She turns on her Ultra-Mobile PC (UMPC), the last session at home is transferred automatically to UMPC. The mobile knows Alice is away from home, and her UMPC is turned on. It routes phone call and "office social networking" to either mobile or UMPC depending on Alice's preference. When Alice decides she is done for the day, she

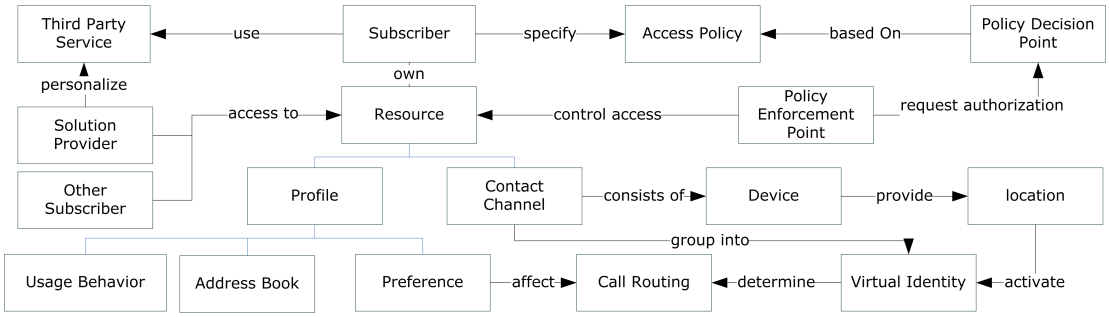

Figure 3: A policy model for future operation mode of telecommunication networks. Each arrowed line represents an action relation between two elements.

turns off UMPC and the session is available for use from within office tomorrow. All voice calls are now routed according to who's calling and "urgent level" settings. Figure 2 illustrates the decision tree of personal call routing scenario.

Presence enabled converged communication: Bob enters Alice's phone number (wireless, landline), email address, physical address, instant message (IM) information into mobile handset. Bob expects the address book to be fully accessible in all his devices (PC, TV, home phone, mobile) for initiating a communication session. Bob is able to see Alice's presence. Bob has the option to specify which method to contact Alice (email, IM, phone). If Bob does not specify which method he would like to use, the service network decides automatically by understanding Bob and Alice's current preference. Alice's preferred method of contact dependent on her online calendar, geo-location, and device used. If automatic method selection is not used, Bob can uses his computer at home on the service broadband to access address book information on Alice's email address. To send her an email, Bob' email client automatically sync-up to network address book to provide the proper email address. Bob can access the same address book while on the road by remotely verifying Bob's identity with the service network and he is able to access the same converged experience.

Third-party service personalization/recommendation: Alice logs into service network and starts surfing the web for blogs. As she looks into different blogs, the widgets on her desktop recommends related web sites, books, and other commercialized sites. The widget recommendation comes from the network operator's collection of previous user behavior and the history is funneled to trusted third-party solution providers. Alice decides to buy a book through widget recommendation. All personal info is seamlessly transferred to this trusted third party (authorized by Alice's decision to purchase). The book's billing is automatically attached to Alice's mobile phone bill or her own Mastercard (another trusted partner of the network operator) depending on Alice's personal preference. The network operator keeps track of Alice's purchase behavior.

4.2 Future Operation Mode Policy Model

The proposed policy model for the future operation mode in converged networks is illustrated in Figure 3. In current operation mode (COM), a subscriber is the subject that initiates requests to the resources owned by the network operator. Contrast to COM, a subscriber is a resource owner and is the authority that specifies access-control polices in future operation mode (FOM).

A subscriber owns a set of resources such as contact channel, usage behavior data, and location information. A *contact channel* is a channel such as phone number or instant message account within the service network, and through which the user can be contacted. Each contact channel is associated with a *communication device* to accept requests. In this sense, a contact channel is a resource that could be protected by access control policies. A subscriber can group logically related contact channels into a *virtual identity*. For instance, a subscriber can group home phone and PC into "home" virtual identity, and group mobile phone and laptop as "roaming" identity. A subscriber can also specify that the laptop is the only contact channel when she is in the "meeting" identity. An A-GPS equipped communication device provides *location* information. This physical presence information can then be used to activate a specific virtual identity. The activated virtual identity and preference settings then determine how voice calls should be routed automatically.

The subjects that make request in the FOM service network are other callers in the same or different networks, and third-party solution providers. Other callers attempt to access a subscriber's contact channel, and third-party solution providers require subscriber's profile data and usage behavior information in order to provide personalized services and recommendations. The subjects in the FOM service network can be grouped by a subscriber into roles via address books. Each role can be subsequently organized into role hierarchy. To protect possesed resources, a subscriber issues credentials and access-control polices. For instance, a subscriber can specify only callers in VIP role can contact her during meeting. As it might exists important callers that are unknown to a subscriber, the subscriber can delegate the role-member definitions to trusted subscribers to prevent missing important calls.

4.3 Future Operation Mode RT Credentials

The RT credentials for FOM service network including virtual identity association, caller role grouping , caller role hierarchy and third-party solution provider identification. The relation that a subscriber U associate a set of contact channels into a virtual identity v can be expressed as:

$$U.v \longleftarrow \bigcup_{i=1}^{n} D_i$$

For instance, *Alice* associates home phone and PC into "home" virtual identity, and use her laptop as the contact channel when she is in "roaming" can be expressed by two credentials: $Alice.virtual(home) \longleftarrow HomePhone_{Alice} \cup HomePC_{Alice}$ and $Alice.virtual(roaming) \longleftarrow Laptop_{Alice}$.

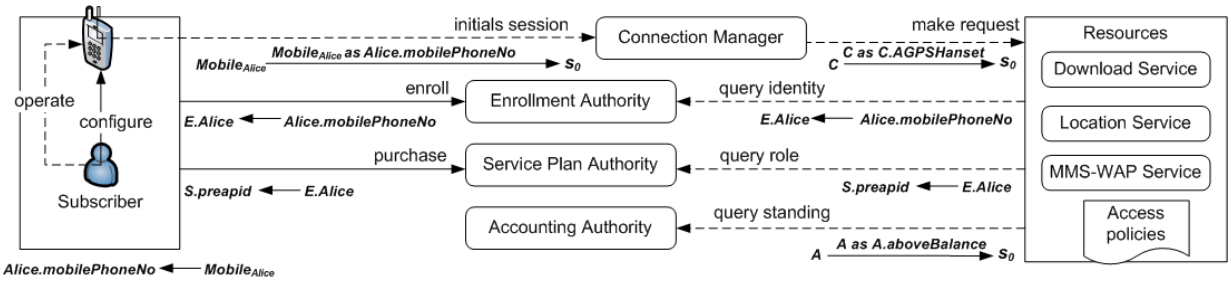

Figure 4: An use case of the proposed policy model. Solid lines represent actions during subscribers setup time, and dashed lines are request actions at runtime. For each action, a sample RT credential is presented.

A subscriber U groups callers into a role r can be specified as:

$$U.r \longleftarrow \bigcup_{i=1}^{n} E.U_i$$

For example, credentials $Alice.boss \longleftarrow E.Bob$ and $Alice.friend \longleftarrow E.Fred \cup E.Friend$ can be used to represent scenarios that $Alice$ assigns Bob as $boss$, and $Fred$ and $Frank$ as $friend$.

Similar to RBAC, the role in RT can be organized into role hierarchy. The role in the higher level dominates the role below, which has every permission the role below has. Role hierarch can be expressed as:

$$U.r \longleftarrow \bigcup_{i=1}^{n} U.r_i$$

Assume $Alice$ wants both $boss$ and $manager$ roles have every permission the role $colleague$ has, the corresponding credential is $Alice.colleague \longleftarrow Alice.boss \cup Alice.manager$.

Third-Party solution providers require identity credentials issued by the network operator to access a subscriber's profile data. To do that, a solution manager P in the network operator could issue $P.3Wave \longleftarrow CA.userCert(3Wave)$ to use a certificate provided by a trusted CA to speak for a provider $3Wave$. The CA might generates a public key for $3Wave$'s certificate by issuing $CA.userCert(3Wave) \longleftarrow PublickKey_{3Wave}$. The solution manager P then issues another credential $P.trustedProvider \longleftarrow P.3Wave$ to indicate $3Wave$ is indeed a trusted solution provider. If $Alice$ wants to allow all trusted provider to access to her profile data, $Alice$ could author a policy as: $Alice.allow \longleftarrow P.trustedProvider$.

5 Policy Model Use Case

In this section, we used a use case to demonstrate how credential and policies can be developed based on the proposed policy models. For the purpose of discussion, we

assume that the network infrastructure provides a set of *logical* entities as illustrated in Figure 4. The mapping between logical and physical entities is currently of no concern, and each logical entity is functioning as follows:

- `Enrollment Authority` (E): Acts as a subscriber registration authority. This entity issues subscriber's credentials during registration, such as $E.Alice \longleftarrow Alice.mobilePhoneNo$.
- `Service Plan Authority` (S): Acts as a service plan subscription authority. This entity defines $S.prepaid$ and $S.postpaid$ roles, and assign subscribers into these roles during service plan subscription, such as $S.prepaid \longleftarrow E.Alice$
- `Accounting` (A): Acts as an accounting authority. This entity defines $A.aboveBalance$ role for the subscriber who has a positive balance. `Accounting` also defines $A.goodStanding$ role as:

$$A.goodStanding \longleftarrow (S.prepaid \cap A.aboveBalance) \cup S.postpaid \quad (1)$$

  `Accounting` issues credentials for subscribers during a service request session to indicate whether the requesting subscriber is in good standing.

- `Connection Manager` (C): Acts as a connection authority. This entity defines $C.AGPSHandset$, and $C.AGPSTower$ roles. `Connection Manager` detects the capabilities of a device, and issues credentials for devices during a service request session.
- `Download Service` (D): This entity provides content download services for subscribers. The decision whether a content download should be granted to a subscriber is made according to the following rules:
  - checks if subscriber is on prepaid or postpaid
  - if is prepaid, checks against balance of account
    - if there is balance, allows access.
    - if no balance, denies access.
  - if is postpaid, allows access.

  `Download Service` references roles defined by `Accounting Authority`, and specifies access control policies based on above rules as follows:

$$D.allow \longleftarrow A.goodStanding \quad (2)$$

- `Location Service` (L): This entity provides location-based services for subscribers. The decision whether a location-based service should be provided to a subscriber is made according to the following rule:
  - Determine whether A-GPS is supported or not or plain old cell tower location

  According to the rule above, this service references roles defined by `Accounting Authority` and `Connection Manager`, and specifies access control policy as follows:

$$L.allow \longleftarrow A.goodStanding \cap C.AGPSHandset \cap C.AGPSTower \quad (3)$$

- `Subscribers`: `Alice` is a prepaid subscriber with balance greater than zero, and `Bob` and `Charlie` are postpaid subscribers. `Bob` is `Alice`'s boss and `Charlie` is `Bob`'s VIP customer. `Alice` authors $Alice.virtual(meeting) \longleftarrow Mobile_{Alice}$ to specify mobile phone is the only contact channel available when she is in the meeting, and then issues the following policy to represent only callers in *vip* and *boss* roles are able to contact her.

$$Alice.allow(Alice.virtual(meeting)) \longleftarrow Alice.vip \cup Alice.boss \tag{4}$$

Our model design assumes that each of the above policy enforcement points (i.e., `Location` and `Download Service`) is served by a dedicated policy decision point (PDP). The separation between PEP and PDP enables several or more instances of same PEP (e.g., `Download Service`) to be served by a (logically) single PDP, which enables enforcement of consistent policies. From now on, we will refer to the above services as single entities but the reader should keep it in mind that our design allows multiple instances of same service.

5.1 Credentials In Action

Credentials are issued by the entities in Figure 4 during setup process (e.g., identity enrollment and service plan subscription) or within a resource request session. Actions performed during subscriber setup time are depicted in solid lines, while dashed lines are used to represent request actions at runtime.

As illustrated in Figure 4, `Enrollment Entity` (E) issues subscriber's identities during user registration:

$$E.Alice \longleftarrow Alice.mobilePhoneNo \tag{5}$$

$$E.Bob \longleftarrow Bob.mobilePhoneNo \tag{6}$$

$$E.Bob \longleftarrow Charlie.mobilePhoneNo \tag{7}$$

`Alice` might also enroll an identity for her home land-line phone, and another identity for ADSL internet connection:

$$E.Alice \longleftarrow Alice.landPhoneNo$$

$$E.Alice \longleftarrow Alice.ADSLUserName$$

Each user (`Alice` and `Bob`) subscribes to a service plan, and `Service Plan Authority` (S) in turn assigns $E.Alice$ as a $S.prepaid$ user, and $E.bob$ as $S.postpaid$:

$$S.prepaid \longleftarrow E.Alice \tag{8}$$

$$S.postpaid \longleftarrow E.Bob \tag{9}$$

Before initiating a request session, `Alice` setups her cellular phone using mobile phone number provided by the network operator:

$$Alice.mobilePhoneNo \longleftarrow Mobile_{Alice} \tag{10}$$

`Bob` also configures his cellular phone as:

$$Bob.mobilePhoneNo \longleftarrow Mobile_{Bob} \tag{11}$$

`Alice` operates her mobile phone $Mobile_{Alice}$ to initiate a session s_0:

$$Mobile_{Alice} \xrightarrow{Mobile_{Alice} \text{ as } Alice.mobilePhoneNo} s_0 \tag{12}$$

`Bob` uses his mobile phone $Mobile_{Bob}$ to initiate a session s_1:

$$Mobile_{Bob} \xrightarrow{Mobile_{Bob} \text{ as } Bob.mobilePhoneNo} s_1 \tag{13}$$

Assume `Alice`'s cellular phone supports A-GPS (while `Bob`'s does not), and is current located in the A-GPS enabled tower, thus `Connection Manger (`C`)` issues device capacity credentials for request session s_0:

$$C \xrightarrow{C \text{ as } C.AGPSHanset} s_0 \tag{14}$$

$$C \xrightarrow{C \text{ as } C.AGPSTower} s_0 \tag{15}$$

In response to the queries performed by policy enforcement points, `Accounting Authority (`A`)` issues balance standing credentials for request session s_0:

$$A \xrightarrow{A \text{ as } A.aboveBalance} s_0 \tag{16}$$

Bob specifies `Charlie` as his *vip* caller, `Alice` assigns `Bob` is her *boss* and delegates her *vip* role-member definition to the member of her *boss* role can be represented using the following credentials respectively:

$$Bob.vip \longleftarrow E.Charlie \tag{17}$$

$$Bob.vip \longleftarrow E.Charlie \tag{18}$$

$$Alice.vip \longleftarrow Alice.boss.vip \tag{19}$$

5.2 Request Proving

Based on the credentials issued in the previous section, this section discusses the steps a PDP would take to conclude whether a request should be granted or denied. A PDP decides whether to authorize a specific request by answering the *proof-of-compliance* question: "Does the access rules and credentials authorize the request?"

***Request 1*: Should the request made by a session initiated by `Alice`'s cellular phone to `Download Service` be granted? Goal:** $D.allow \longleftarrow s0$

Proof In according with the identity credential issued by `Enrollment Authority` in (5), identity to device association (10), and delegation credential for the request session (12), `Download Service` derives a credential chain as follows:

$$E.Alice \xleftarrow{(5)} Alice.mobilePhoneNo \xleftarrow{(10)} Mobile_{Alice} \xrightarrow{Mobile_{Alice} \text{ as } Alice.mobilePhoneNo\ (12)} s_0$$

Based on the above credential chain, `Download Service` proves s_0 speaks for $E.Alice$:

$$E.Alice \longleftarrow s_0 \tag{20}$$

From `Alice`'s service plan (8), and s_0 speaks for $E.Alice$ (20), `Download Service` derives the following credential chain:

$$S.prepaid \xleftarrow{(8)} E.Alice \xleftarrow{(20)} s_0$$

Based on the above credential chain, `Download Service` proves that $S.prepaid$ role is activated in s_0 :

$$S.prepaid \longleftarrow s_0 \tag{21}$$

According to the delegation credential issued by `Accounting Authority (A)` for s_0 (16), `Download service` knows s_0 is a member of $A.aboveBalance$:

$$A.aboveBalance \longleftarrow s_0 \tag{22}$$

From the good standing role defined by `Accounting Authority` in (1), the facts that $S.prepaid$ role is activated in s_0 (21), and s_0 is above balance (22), `Download Service` proves s_0 is in good standing:

$$A.goodStanding \longleftarrow s0 \tag{23}$$

Based on access policy in (2) and s_0 is in good standing (23), `Download Service` concludes that the request made by s_0 should be granted. ■

***Request 2*: Should the request made by a session initiated by `Alice`'s cellular phone to `Location Service` be granted? Goal:** $L.allow \longleftarrow s0$

Proof From the device capability credentials issued by `Connection Manager` in (14) and (15), `Location Service` knows s_0 is in the role of $C.AGPSHanset$ and $C.AGPSTower$:

$$C.AGPSHanset \longleftarrow s_0 \wedge C.AGPSTower \longleftarrow s_0 \tag{24}$$

From access policy in (3), the fact that s_0 is in good standing (23), and s_0 is in the role of $C.AGPSHanset$ and $C.AGPSTower$ (24), `Location Service` concludes that the request made by a session initiated by Alice's cellular phone should be allowed. ■

***Request 3*: Should the request made by a session initiated by the `Bob`'s cellular phone to `Download Service` be granted? Goal:** $D.allow \longleftarrow s_1$

Proof From the `Bob`'s identity credential (6), `Bob`'s identity to device association (11), and the delegation credential for the request session initiated by the `Bob`'s cellular phone (13), `Download Service` proves s_1 represents $E.Bob$:

$$E.Bob \longleftarrow s_1 \tag{25}$$

`Bob` subscribes to a postpaid service plan (9), and s_1 represents $E.Bob$ (25), `Download Service` infers that $S.postpaid$ role is activated in s_1:

$$S.postpaid \longleftarrow s_1 \tag{26}$$

From the good standing role defined by `Accounting Authority` in (1) and the fact that $S.postpaid$ role is activated in s_1 (26), `Download Service` proves s_1 is in good standing:

$$A.goodStanding \longleftarrow s1 \tag{27}$$

Based on access policy in (2) and the fact that s_1 is in good standing (27), `Download Service` concludes the request made by Bob's cellular phone should be permitted. ■

***Request 4*: Should the request made by a session initiated by the `Bob`'s cellular phone to `Location Service` be granted? Goal:** $L.allow \longleftarrow s_1$

Proof The access policy in (3) states that in order to grant a access, the session that makes the request has to be a member of $A.goodStanding$, and the device must support A-GPS ($C.AGPSHandset$) and currently located in a A-GPS enabled tower ($C.AGPSTower$). Although based on (27), s_1 is in good standing, but since `Bob`'s cellular phone does not support A-GPS, the required role $C.AGPSHandset$ is not presented in s_1. Thus, the request made by s_1 to `Location Service` should be denied. ■

***Request 5*: Should the request made by `Charlie`'s cellular phone to `Alice` mobile phone be granted when she is in the meeting? Goal:** $Alice.allow(Alice.virtual(meeting)) \longleftarrow Mobile_{Charlie}$

Proof `Alice`'s call policy in (4) states that only callers in `Alice`'s *boss* and *vip* roles can contact her during meeting. `Charlie` is not in either role directly. However, `Alice` delegates *vip* role definition to the member of her *boss* role (19). `Bob` is in `Alice`'s *boss* role (18), and `Bob` defines `Charlie` is in his *vip* role (17). Based on this credential chain, PDP concludes `Charlie` is in `Alice`'s *vip* role. Thus, the request made by `Charlie`'s cellular phone to `Alice` mobile phone should be granted during meeting. ■

6 Service-Plan Conformance Checking

Due to the complexity and diversity nature of telecommunication network, it is possible that services subscribed by a user might not match the services that are actually allowed. In this section, we discuss conformance checking on *service-plan policy model*. A service plan is a set of *services* a subscriber can subscribe. Once subscribed, a set of corresponding *roles* are in turn assigned to the subscriber. The subscriber uses assigned roles to request services that are protected by access-control policies expressed in RT. Access-control policies require decisions (RT credentials/policy statements) issued by multiple entities within a network operator to make final access decisions. For example, `location` service might require a subscriber in good standing (defined by `accounting` entity), and the device equipped with Assisted-GPS (defined by `connection tower`) in order to decide whether a request should be granted or not. The goal of conformance checking is to verify whether this inconsistent situation exists for all reachable states in the system. In other words, we check whether the configuration of service plans is conformed to a given set of RT policies, for all possible changes in the RT policies.

In order to achieve our goal, we implemented tools for authoring RT polices and configuring service plans. We then designed a service-plan *model class* in Java which is executed and verified in Java PathFinder [MGPMS08]. The service-plan model class models the state behaviors of the system. A state is represented by the set of service plan subscribed by a user (*service-plan state*) and a set of RT policy statements (*policy state*) within the policy model. Both service-plan state and policy state evolve over time. The model *invariant* is the services subscribed by a user must equal to the services allowed. Given a configuration of service plans and initial RT policies, the model class checks whether the model invariant holds for all reachable states.

6.1 Problem Formulation

Figure 5 illustrates service plan policy model. A *service plan* contains a set of voice/data *services*, and associated with a set *roles*. Service plans might change frequently for the purpose of marketing, but a role typically has a few service levels within a network operator. When a service plan is subscribed by a subscriber, the corresponding set of roles is assigned to the user. The roles are defined by *management authority* such as accounting or enrollment entities. Access to a service is mediated by *access-control policies* which are authored by *service authority*. A *user*

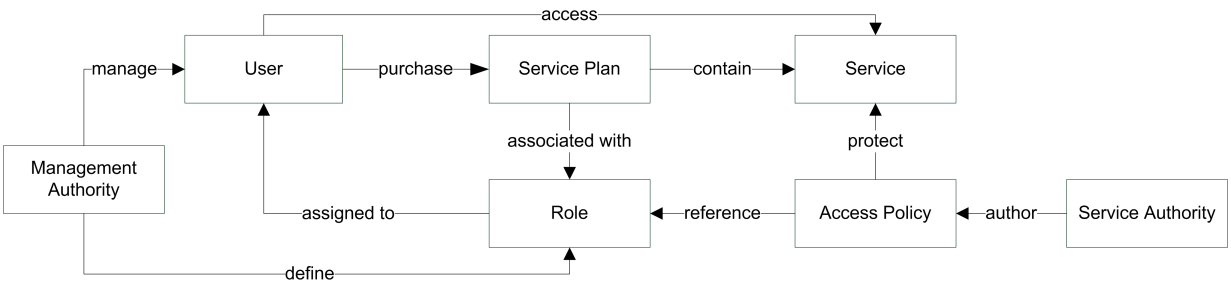

Figure 5: Service-plan policy model. Each arrowed line represents an action relation between two elements.

in the model is a personal or business subscriber that are managed by the management authority. A user subscribed one or more services plans, the service plans in turn assign the corresponding set of roles to the user. A user then initiates a request using the assigned roles.

The proposed service-plan policy model is formally expressed by RT policy statements. An RT policy consists of a set of RT policy statements. Based on the use-case scenarios provided by the network operator, there are four types of RT policies within service plan model:

- Management Policy $\mathcal{P}_{\mathcal{M}}$. This policy consists of policy statements issued by management entities to defining roles and delegations between them. For instance, `Accounting` entity A can define $A.aboveBalance$ role for the subscriber who has a positive balance. `Accounting` might also defines $A.goodStanding$ role as $A.goodStanding \longleftarrow (S.prepaid \cap A.aboveBalance) \cup S.postpaid$ by delegating $S.prepaid$ and $S.postpaid$ roles to service plan authority S.
- User Management Policy $\mathcal{P}_{\mathcal{U}}$. This policy consists of policy statements issued by management entities to manage users. For instance, `Enrollment` entity E can issues subscriber's credentials during registration, such as $E.Alice \longleftarrow Alice.mobilePhoneNo$. `Accounting` A might issues credentials for subscribers during a service request session to indicate whether the requesting subscriber is in good standing, such as $A.abovebalance \longleftarrow E.Alice$.
- Access-control Policy $\mathcal{P}_{\mathcal{A}}$. The policy statements in this policy are used to mediate access to services. For example, `Download Service` D might allow access for all subscribers in good standing by specifying $D.allow \longleftarrow A.goodStanding$. On the other hand, `Location Service` L might require the devices making the request are equipped with Assisted-GPS chips which is determined by `Connection Manager` C, such as $L.allow \longleftarrow A.goodStanding \cap C.AGPSHandset$.
- User Service Subscription Policy $\mathcal{P}_{\mathcal{S}}$. This policy consists of role assignment policy statements issued by service plan authority to users. For instance, `Alice` and `Bob` each subscribes to a different service plan, and `Service Plan Authority` S in turn assigns $E.Alice$ as a $S.prepaid$ user, and $E.bob$ as $S.postpaid$ by $S.prepaid \longleftarrow E.Alice$ and $S.postpaid \longleftarrow E.Bob$

In this paper, we define a set $\mathcal{P}$ of policy statements a *state* of the policy model. Given a state $\mathcal{P}$, the state can evolve into another state by adding or removing existing statements. The state evolution is guarded by a restriction rule $\mathcal{R} = (\mathcal{G}_R, \mathcal{S}_R)$ where $\mathcal{G}_R$ is a set of *growth-restricted* roles, and $\mathcal{S}_R$ is a set of *shrink-restricted* roles. Given a state $\mathcal{P}$ and a restriction rule $\mathcal{R}$, we denote $\mathcal{P} \mapsto_{\mathcal{R}} \mathcal{P}'$ if change from $\mathcal{P}'$ to $\mathcal{P}'$ is allowed by $\mathcal{R}$. We denote $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ if a sequence of zero or more allowed changes from $\mathcal{P}'$ to $\mathcal{P}'$. If $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$, we say $\mathcal{P}'$ is $\mathcal{R}$-reachable, or simply $\mathcal{P}'$ is reachable when $\mathcal{P}$ and $\mathcal{R}$ are clear from context.

Given a state $\mathcal{P}$ and a query $\mathcal{Q}$, the relation $\mathcal{P} \vdash \mathcal{Q}$ means $\mathcal{Q}$ is true in $\mathcal{P}$. The query $\mathcal{Q}$ takes the form $\alpha \sqsupseteq \beta$, in which α and β are either role or explicit set of principals. For instance, $A.r \sqsupseteq \{u\}$ holds if principal u is a member of $A.r$ in state $\mathcal{P}$.

The set of services and service plans provided by a network operation is denoted by $\mathcal{S}$ and $\mathcal{SP}$ respectively. A service plan $sp \in \mathcal{SP}$ is a tuple of (S, R) where $S = \{s|\ s \in \mathcal{S}\}$ is a set of associated services and $R = \{r|\ r \in roles(\mathcal{P}_M)\}$ is the set of corresponding roles that are defined by management authority. Each service has a permission role $s.permission \in roles(\mathcal{P}_M)$ specified by service authority. The set of service plans purchased by a user u is denoted by $\mathcal{SP}_u = \{sp|\ sp \in \mathcal{SP}\}$. Given a $\mathcal{SP}_u$, the set of subscribed services is $S_u = \{s|\ s \in \bigcup_{i=1}^{n} sp_i.S, sp_i \in \mathcal{SP}_u\}$, and the corresponding roles $R_u = \{r|\ r \in \bigcup_{i=1}^{n} sp_i.R, sp_i \in \mathcal{SP}_u\}$ can be obtained. For each role $r \in R_u$, there is a corresponding policy statement added to $\mathcal{P}_S$. Thus, $\mathcal{P}_S = \bigcup_{i=1}^{n} r_i \longleftarrow u,\ r_i \in R_u$.

In service-plan policy model, the policy state is the set of policy statements $\mathcal{P} = \mathcal{P}_\mathcal{S} \bigcup \mathcal{P}_\mathcal{M} \bigcup \mathcal{P}_\mathcal{U} \bigcup \mathcal{P}_\mathcal{A}$. Assume a user can purchase any combination of service plans, and the management entities can delegate authorities between each other. We also assume access-control policy $\mathcal{P}_\mathcal{A}$, and user roles defined by management authority $\mathcal{P}_\mathcal{U}$ remain fixed in every state. Thus, the restriction rule for our model is $\mathcal{R} = (\mathcal{G}_R, \mathcal{S}_R) = (roles(\mathcal{P}_\mathcal{U} \bigcup \mathcal{P}_\mathcal{A} \bigcup \mathcal{P}_\mathcal{S}), roles(\mathcal{P}_\mathcal{U} \bigcup \mathcal{P}_\mathcal{A} \bigcup \mathcal{P}_\mathcal{S}))$. For each reachable state $\mathcal{P} \mapsto_{\mathcal{R}} \mathcal{P}'$, the allowed set of services for a subscriber is $S_a = \{s|\ s \in \mathcal{S}, \mathcal{P} \vdash s.permission \sqsupseteq \{u\}\}$. The goal of conformance checking is to verify whether the following model invariant holds for every reachable state.

$$S_u = \{s|\ s \in \bigcup_{i=1}^{n} sp_i.S, sp_i \in \mathcal{SP}_u\} \equiv S_a = \{s|\ s \in \mathcal{S}, \mathcal{P} \vdash s.permission \sqsupseteq \{u\}\}$$

6.2 Approach

To enable conformance checking on RT-based service-plan policy model, we (1) implemented tools for authoring RT policy statements and configuring service plans, (2) extended RT inference engine to answer containment queries, and (3) designed a *model class* in Java, which is then executed and verified in Java PathFinder [MGPMS08]. Figure 6 illustrates the main components in our conformance checking system architecture. In this paper, we focus the discussions on the model class. The service-plan model class takes a configuration of service plans and a set of RT polices as inputs,

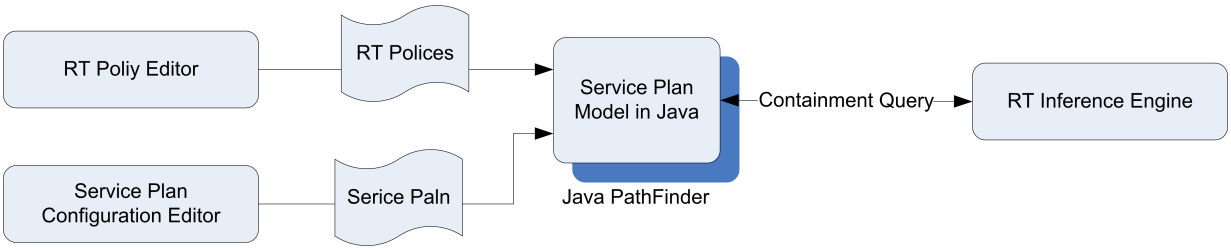

Figure 6: System architecture for RT-based service plan conformance checking.

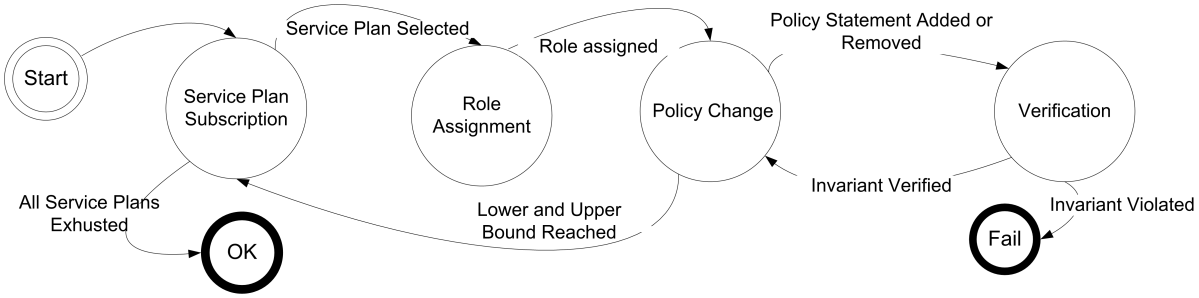

Figure 7: Conformance checking flow of RT-based service plan model.

and checks whether the model invariant holds for all reachable states. During conformance checking, the RT inference engine is employed to answer containment queries in form of $\mathcal{P} \vdash s.permission \sqsupseteq \{u\}$.

Java PathFinder (JPF) is an explicit-state software model checker for verifying Java bye-code programs. It is a custom-made Java Virtual Machine that support all features of Java in addition to *non-deterministic choices*. Non-deterministic choices represents the states of a program under verification in JPF. JPF explorers all execution paths of a given program. It backtracks when a previous explored state is visited. Developers can guide the search by defining different search heuristics or specifying conditions when the search backtracks. JPF provides a set of non-deterministic data choice generators in the class `gov.nasa.jpf.jvm.Verify`. For verification under JPF, the main idea is to obtain non-deterministic input data values from JPF in a way that it can systematically analyze all relevant choices. JPF non-deterministic choices are used in our service-plan model class to represent the state of RT polices and the subscription status of service plans.

6.2.1 Conformance Checking Flow and Algorithms

Figure 7 shows the checking flow performed by the proposed model class. Given by a set of service plans ($\mathcal{SP}$) and RT policies ($\mathcal{P}_\mathcal{A}$, $\mathcal{P}_\mathcal{M}$, $\mathcal{P}_\mathcal{U}$), the model class subscribes a set of service plan $\mathcal{SP}_u$. The procedure for selecting $\mathcal{SP}_u$ is illustrated in Algorithm 1. Based on $\mathcal{SP}_u$, the union set of subscribed services S_u and roles R_u are derived. For each $r \in R_u$, a new policy statement in form of $r \longleftarrow u$ is add to policy $\mathcal{P}_\mathcal{S}$. The model class then evolves to next state by adding or removing policy statements from policy $\mathcal{P}_\mathcal{M}$. To guard policy state evolutions, the model class first

computes the lower and upper-bound of policy $\mathcal{P}_{\mathcal{M}}$ based on the procedure described by Li et at. [LMW05].

Algorithm 1: next_service_plan_state

```
1: Input: a set of service plans SP
2: Output: next random combination of service plans SP_u
3: SP_u ← empty;
4: for i = 1 to |SP| do
5:    if JPF.Verify.getBoolean() == true then
6:       SP_u.add( SP[i]) ;
7:    end if
8: end for
9: return SP_u;
```

A role's lower-bound is the set of principals that are members of the role in every reachable state. A policy $\mathcal{P}$ contains a set of policy statement that defines role membership. The lower-bound of a policy $\mathcal{P}$ can be obtained by removing all statements that defining shrink-unrestrict roles. Algorithm 2 illustrates the algorithm for computing lower-bound of a policy.

Algorithm 2: lower_bound_policy

```
1: Input: a set of policy statements P
2: Input: a set of shrink-restricted roles S_R
3: Output: a lower-bound set of policy statements P_⊥
4: P_⊥ = P;
5: R = roles(P);
6: P = principles(P);
7: for all role r in R do
8:    if r ∉ S_R then
9:       for all principle p in P do
10:         if P_⊥.contains(r, p) == true then
11:            P_⊥.remove(policy_statement(r, p)) ;
12:         end if
13:      end for
14:   end if
15: end for
16: return P_⊥;
```

Contrast to shrink-unrestrict role, a growth-unrestrict role could have every principal as its member. The upper-bound of a policy $\mathcal{P}$ can be obtained by adding every principal to growth-unrestrict roles in a policy. Algorithm 3 illustrates the algorithm for computing upper-bound of a policy.

Algorithm 3: upper_bound_policy

```
1: Input: a set of policy statements 𝒫
2: Input: a set of growth-restricted roles 𝒢_ℛ
3: Output: an upper-bound set of policy statements 𝒫_⊤
4: 𝒫_⊤ = 𝒫;
5: R = roles(𝒫);
6: P = principles(𝒫);
7: for all role r in R do
8:   if r ∉ 𝒢_ℛ then
9:     for all principle p in P do
10:      if 𝒫_⊤.contains(r, p) == false then
11:        𝒫_⊤.add(policy_statement(r, p)) ;
12:      end if
13:    end for
14:  end if
15: end for
16: return 𝒫_⊤;
```

Once the lower and upper-bound of policy $\mathcal{P}_{\mathcal{M}}$ are obtained, the model class evolve to next state $\mathcal{P}'$ by adding or removing one statement from the original policy $\mathcal{P}_{\mathcal{M}}$, as illustrated in Algorithm 4.

For all reachable state, the model class check if the invariant $S_u = S_a$ holds. If the invariant is violated, the checking process stops, and the corresponding counter-example is presented. If the invariant hold for that state, the model class evolves to next state until all states in $\mathcal{P}_{\mathcal{M}}$ are reached. The model class then selects next possible combination of service plans. The iteration continue until all possible service plan subscriptions are exhausted. Algorithm 5 shows the checking flow of the model class.

6.2.2 Conformance Checking Examples

In this section, we demonstrate two examples that violate model invariant, and is detected by the model class during verification.

Example 1: inconsistent between $\mathcal{SP}$ and $\mathcal{P}_A$

Service Plan: Service Plan 1
- Services included: $S1$, $S2$
- Roles to be assigned: $A.r1$, $A.r2$

Access-Control Policy $\mathcal{P}_A$:
(1): $S1.allow \longleftarrow A.r1$
(2): $S2.allow \longleftarrow A.r2$

Algorithm 4: next_policy_state

```
1: Input: a set of policy statements P
2: Input: a lower-bound set of policy statements P_⊥
3: Input: an upper-bound set of policy statements P_⊤
4: Output: next policy state P'
5: P' = P ;
6: △_⊥ = P − P_⊥ // set of policy statements to be removed;
7: △_⊤ = P_⊤ − P // set of policy statements to be added;
8: for i = 1 to |△_⊥| do
9:    if JPF.Verify.getBoolean() == true then
10:      P'.remove(△_⊥[i]) ;
11:   end if
12: end for
13: for i = 1 to |△_⊤| do
14:   if JPF.Verify.getBoolean() == true then
15:      P'.add(△_⊤[i]) ;
16:   end if
17: end for
18: return P';
```

(3): $S3.allow \longleftarrow A.r1 \cap A.r2$

When a user u subscribes Service Plan 1, the User Service Subscription Policy $\mathcal{P}_{\mathcal{S}}$ contains:
(1): $A.r1 \longleftarrow u$
(2): $A.r2 \longleftarrow u$

In this initial state, the model class detects an inconsistency:
$S_u = \{S1, S2\} \neq S_a = \{S1, S2, S3\}$

The reason for this inconsistency is because based on (3):$S3.allow \longleftarrow A.r1 \cap A.r2$ in $\mathcal{P}_A$ and policy statements in $\mathcal{P}_{\mathcal{S}}$, the model class can concludes $S3.allow \longleftarrow u$.

Example 2: inconsistent between $\mathcal{SP}$ and $\mathcal{P}_M$

Service Plan: Service Plan 1
- Services included: $S1$, $S2$
- Roles to be assigned: $A.r1$, $A.r2, B.r1$

Access-Control Policy $\mathcal{P}_A$:
(1): $S1.allow \longleftarrow A.r1$
(2): $S2.allow \longleftarrow A.r2$
(3): $S3.allow \longleftarrow A.r3$

Algorithm 5: model_checking

1: **Input**: a set of service $\mathcal{S}$
2: **Input**: a set of service plans $\mathcal{SP}$
3: **Input**: a set of management delegation policy statements $\mathcal{P}_M$
4: **Input**: a set of access-control policy statements $\mathcal{P}_A$
5: **Input**: a set of user status policy statements $\mathcal{P}_U$
6: **Output**: boolean value indicates the result of model checking
7: u = new subscriber("Alice") ;
8: $\mathcal{SP}_u$ = next_service_plan_set($\mathcal{SP}$) ;
9: S_u = subscribed_services($\mathcal{SP}_u$) ;
10: R_u = subscribed_roles($\mathcal{SP}_u$) ;
11: $\mathcal{P}_S$ = add_policy_statement(u, R_u);
12: $\mathcal{P}_{M\perp}$ = lower_bound_policy($\mathcal{P}_M$);
13: $\mathcal{P}_{M\top}$ = upper_bound_policy($\mathcal{P}_M$);
14: $\mathcal{P}'_M$ = next_policy_state($\mathcal{P}_M$, $\mathcal{P}_{M\perp}$, $\mathcal{P}_{M\top}$);
15: $S_a \leftarrow$ empty;
16: **for all** service s in S **do**
17: $r = s$.permission_role;
18: **if** is_allow(r, u, $\mathcal{P}'_M + \mathcal{P}_S + \mathcal{P}_A + \mathcal{P}_U$) == true **then**
19: S_a.add(s) ;
20: **end if**
21: **end for**
22: **return** assert($S_u = S'_a$);

Management Policy $\mathcal{P}_\mathcal{M}$:
(1): $B.r1 \longleftarrow C$
(2): $A.r3 \longleftarrow C.r1.r1$

When a user u subscribes Service Plan 1, the User Service Subscription Policy $\mathcal{P}_\mathcal{S}$ contains:
(1): $A.r1 \longleftarrow u$
(2): $A.r2 \longleftarrow u$
(3): $B.r1 \longleftarrow u$

In this initial state, the system is consistency where $S_u = \{S1, S2\} \equiv S_a = \{S1, S2\}$. However, when $\mathcal{P}_\mathcal{M}$ evolves over time and a new policy statement $C.r1 \longleftarrow B$ is added, the model class detects an inconsistency:
$S_u = \{S1, S2\} \neq S_a = \{S1, S2, S3\}$

The reason for this inconsistency is because based on (2): $A.r3 \longleftarrow C.r1.r1$ in $\mathcal{P}_M$, (3): $B.r1 \longleftarrow u$ in $\mathcal{P}_\mathcal{S}$, and $C.r1 \longleftarrow B$, the model class can concludes $S3.allow \longleftarrow$

u.

6.2.3 Complexity and State-Space Analysis

The algorithm for selecting a random combination of service plans requires looping through every service plan. The complexity for this step is $O(|\mathcal{SP}|)$. To compute lower-bound of a policy, Algorithm 2 loops through each role in the policy, which is $O(|\mathcal{P}|)$. For each role r that is shrink-unrestricted, the algorithm loops through all principal in the policy, which is $O(|\mathcal{P}|)$. For each principal p, the algorithm checks whether the policy statement in form of $r \longleftarrow p$ exists in the lower-bound set, which is $(O(|\mathcal{P}|))$ in worst case. If the policy statement exists, the algorithm removes it from lower-bound set. Thus, the worse-case complexity for computing lower-bound set of a policy is $O(|\mathcal{P}|^3)$. Similarly, the worse-case complexity for computing upper-bound set of a policy is $O(|\mathcal{P}|^3)$ as well. To evolve to next policy state, Algorithm 4 loops through lower and upper-bound set of a policy to remove or add one policy statement, which is $(O(|\mathcal{P}|))$. For each policy state, the model checking algorithm loops through all services to check whether a specific service is allowed, which is $(O(|\mathcal{S}|))$. Thus, the total complexity of conformance checking algorithm is $O(|\mathcal{SP}| + 2|\mathcal{P}|^3 + |\mathcal{P}| + |\mathcal{S}|)$.

The number of state in the model class is determined by the number of service plans and the number of statements of lower and upper-bound set of a policy, which is $2^{|\mathcal{SP}|} * 2^{|\mathcal{P}_\perp|+|\mathcal{P}_\top|}$.

7 Conclusions

In this paper, we have presented RT-based policy models for current and future converged networks. The proposed policy model is a set of inter-related elements that access control policies building upon. The relations between elements in the proposed policy model are expressed formally using RT framework. RT framework is a family of role-based trust-management language for representing policies and credentials in distributed authorization. RT combines the strength of role-based access control and trust-management systems to form a concise and expressive language. The policy model is used as the basis for the specification of access-control policies. Based on the proposed policy model, we developed access-control polices for the use case scenarios using RT credentials to demonstrate the applicability of the proposed model.

For future works, we plan to implement an RT inference engine that supports RT_0, RT_1, RT_2, RT^D, and takes RT credentials in RTML form as input. Based on this RT inference engine, we will compile the proposed credentials, validate the policies, and infer whether a specific request should be granted. The resulted work will be incorporated into the unified access management framework we are currently working on.

References

[3GP03] 3GPP. Overview of 3GPP Release 5: Sumary of all Release 5 Features. http://www.3gpp.org/specs/releases-contents.htm, September 9 2003.

[3GP06] 3GPP. 3rd Generation Partnership Project. http://www.3gpp.org/, 2006.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Washington DC, USA, 1996.

[DDLS01] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In Morris Sloman, editor, *Workshop on Policies for Distributed Systems and Networks*, Bristol UK, 2001.

[KL03] Madhur Kohli and Jorge Lobo. Realizing network control policies using distributed action plans. *Journal of Network and Systems Management*, 11(3):305–327, September 2003.

[LBN99] J. Lobo, R. Bhatia, and S. Navqi. A policy description language. In *Proceedings of the National Conference on AI*, pages 291–298, Orlando, FL, 1999.

[LMW02] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 114, 2002.

[LMW05] NingHui Li, John C. Mitchell, and William H. Winsborough. Beyond proof-of-compliance: Security analysis in trust management. *Journal of the ACM*, 52(3):474–514, 2005.

[MGPMS08] Peter Mehlitz, Dimitra Giannakopoulou, Corina Pasareanu, and Masoud Mansouri-Samani. Java pathfinder, 2008.

[SCFY96] Ravi Sandhu, Edward Coyne, Hal Feinstein, and Charles Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.