Proxy con capacidad de ejecutar SIP Servlets para dar servicios de llamada

El objetivo de esta tercera práctica es dotar al proxy SIP con capacidad de ejecutar SIPServlets de la práctica 2 con un SIPServlet genérico que adapta su ejecución a las instrucciones leídas de un fichero de servicio especificado en XML. La filosofía que subyace detrás de esta práctica es permitir a cualquier usuario de telefonía IP, aun sin conocimientos programáticos, que pueda definirse sus propios servicios de atención de sus llamadas entrantes y salientes. El usuario, normalmente a través de una interfaz gráfica, en la realización de la práctica editando directamente el fichero XML, podrá especificar las horas en las que está disponible para realizar y recibir llamadas así como la lista de usuarios permitidos o prohibidos. La información definida por el usuario se guardará en un fichero XML. El SIPServlet genérico a desarrollar leerá el contenido de estos ficheros XML de servicio de usuario, para llamante o llamada según se explica más adelante, y reaccionará en función de lo que allí se especifique, controlando el flujo de atención de llamada a través del API de SIPServlets definido para la práctica 2.

El SIPServlet genérico

Se trata de codificar la clase GenericSIPServlet que implemente la interfaz SIPServlet y cuyo método doInvite() lea el nombre del llamado y del llamante y ejecute el servicio definido para el llamado o si no hay servicio definido para el llamado(no existe el fichero de servicio XML asociado) ejecute el servicio definido para el llamante (en caso de no haber servicio ni para el llamado ni para el llamante se aceptará la llamada y se progresará según las reglas de la práctica 1). La especificación del servicio para un usuario se hará mediante una sintaxis XML según el siguiente Schema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://sma.org/service.xsd"
xmlns=" http://sma.org/service.xsd" elementFormDefault="qualified">

 <xs:element name="service" type="serviceType"/>

 <xs:complexType name="serviceType">
  <xs:sequence>
   <xs:element name="inbound" type="inboundType"/>
   <xs:element name="outgoing" type="outgoingType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="inboundType">
  <xs:sequence>
   <xs:element name="availableFrom" type="xs:string"/>
   <xs:element name="availableUntil" type=" xs:string"/>
   <xs:choice>
      <xs:element name="banedUser" type="xs:string" minOccurs=”0”
maxOccurs=”unbounded”>
      <xs:element name="allowedUser" type="xs:string" minOccurs=”0”
maxOccurs=”unbounded”>
   </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="outgoingType">
  <xs:sequence>
   <xs:element name="availableFrom" type="xs:string"/>
   <xs:element name="availableUntil" type=" xs:string"/>
   <xs:choice>
      <xs:element name="banedUser" type="xs:string" minOccurs=”0”
maxOccurs=”unbounded”>
```

```
      <xs:element name="allowedUser" type="xs:string" minOccurs="0"
maxOccurs="unbounded">
   </xs:choice>
  </xs:sequence>
</xs:complexType>


</xs:schema>
```

Un ejemplo de fichero de servicio según la especificación anterior podría ser (se permiten llamadas entrantes y salientes de 10 de la mañana a 10 de la noche, se podrá llamar solamente al jefe pero se podrán recibir llamadas de todo el mundo salvo de dos usuarios prohibidos):

```
<?xml version="1.0"?>
<service>
   <inbound>
        <availableFrom>10</availableFrom>
        <availableUntil>22</availableUntil>
        <banedUser>sip:jones@domain.org</banedUser>
        <banedUser>sip:james@dom.org</banedUser>
   </inbound>
   <outgoing>
        <availableFrom>10</availableFrom>
        <availableUntil>22</availableUntil>
        <allowedUser>sip:boss@domain.org</banedUser>
   </outgoing>
</service>
```

Los ficheros definiendo el servicio de procesamiento de llamadas entrantes y salientes para un usuario según la sintaxis anterior se nombrarán según el formato: usuario.xml. De esta forma, el GenericSIPServlet leerá primero el usuario llamado y mirará si existe el fichero de servicio XML asociado. Si existe ejecutará el servicio especificado para la parte inbound. Si no existe el fichero, mirará si existe el fichero de servicio XML para el llamante. Si existe ejecutará la parte outgoing de dicho fichero. Si no existe ni fichero de servicio para llamante ni llamado se progresará la llamada al igual que se hacía en la práctica 1.

La implementación del GenericSIPServlet usará el parser SAX para leer la información del fichero de servicio que necesite ejecutar y hará las comprobaciones apropiadas para saber si tiene que invocar el método getProxy o createResponse del objeto SIPServletRequest que recibe en el INVITE.

Probando la práctica

Para poder comprobar el correcto funcionamiento de la práctica, se requiere que el alumno implemente dos ficheros de servicios para dos usuarios. Para un usuario U1 se le dejará llamar a U2 pero no a U3 y recibir llamadas de U2 y U3. Para un usuario U2 se le dejará llamar a U3 pero no a U1 y recibir llamadas de U1 pero no de U3. Hacer pruebas entre U1, U2 y U3 de dos en dos como llamantes y llamados y verificar el correcto funcionamiento. Nótese que para las llamadas de U2 a U1 hay una diferencia entre la visión de cada uno de los usuarios en sus ficheros de servicios pues U1 sí que permite recibir llamadas de U2 pero U2 no permite hacer llamadas a U1. Esto se ha definido así para verificar que se comprueba solo el llamado en caso de tener servicio definido tanto para el llamante como para el llamado y por lo tanto en este caso se aceptaría la llamada.