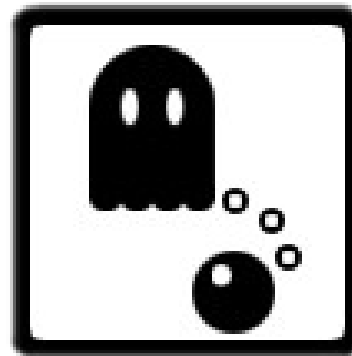


Introducción a SIP y OpenSER

Imagine there is no PSTN...



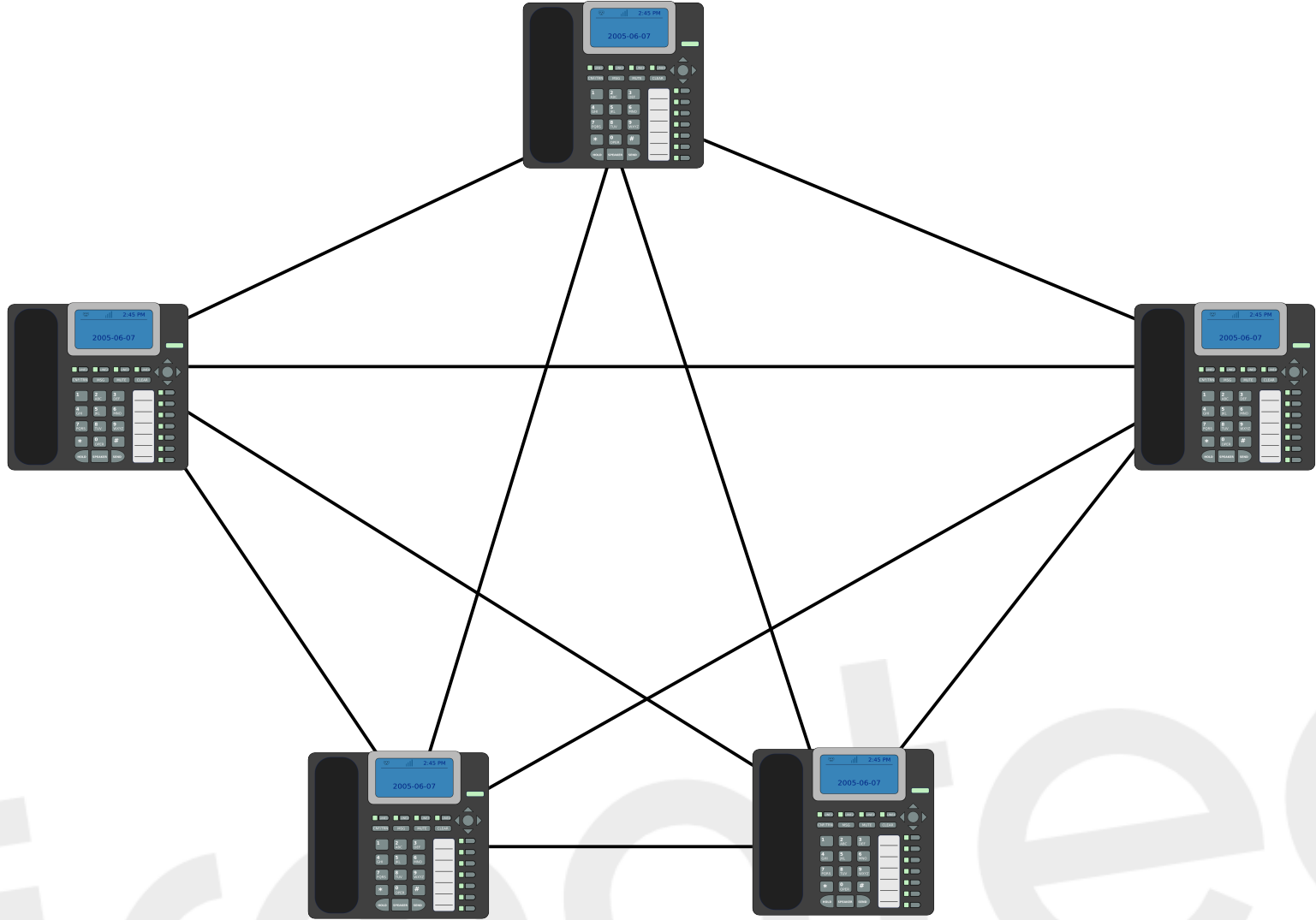
Saúl Ibarra Corretgé
<http://www.saghul.net>

Un poco de historia

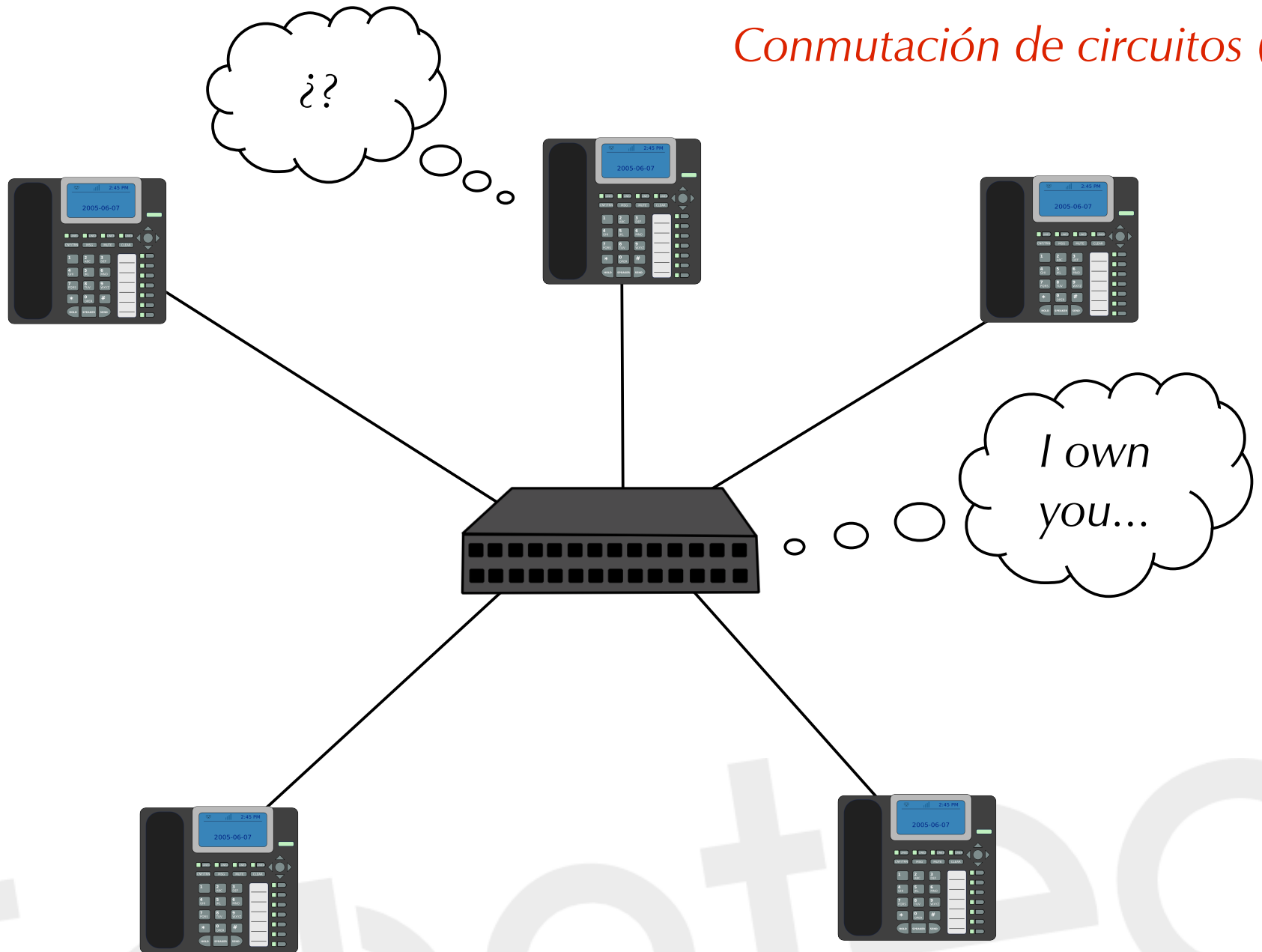
Conmutación de circuitos

- *La telefonía tradicional se basaba en conmutación de circuitos.*
- *Desde el comienzo hasta el final de una llamada se establecía un camino físico.*
 - *Consumo de recursos.*
- *Inicialmente -> redes totalmente malladas*
 - *0% escalable.*
 - *Cambio a estructura en estrella.*

Conmutación de circuitos (2)



Conmutación de circuitos (3)



Conmutación de circuitos (4)

- *Al principio, telefonía 100% analógica.*
- *Gestión del crecimiento*
 - *Analógico: FDM*
 - *Digital: TDM*
- *Posteriormente los switches se sustituyeron por switches digitales.*
- *Digital vs. Analógico*
 - *Digital es más barato.*
 - *Digital tiene mejor calidad.*
 - *Analógico más rápido (switching).*
 - *Complejidad de los terminales digitales.*
- *Solución: terminales analógicos y red troncal digital.*

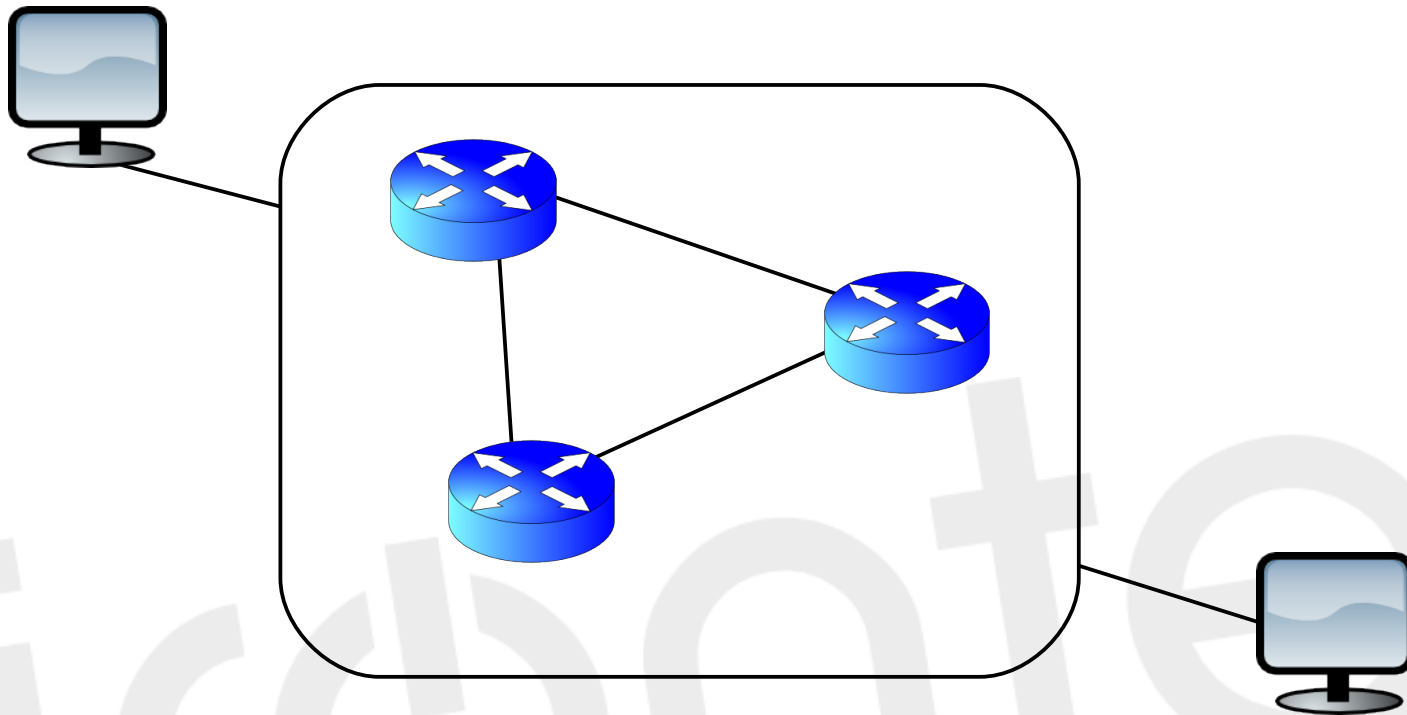
- *Necesidad de comunicación entre distintos sistemas.*
- *Ligado a la evolución de centrales/terminales de usuario*
- *Señalización analógica*
 - *Inband*
- *Señalización digital*
 - *Access signalling (del terminal a la central)*
 - *Ex. DTMF*
 - *Trunk signalling (entre centrales)*
 - *CAS (señalización asociada al canal)*
 - *CCS (señalización por canal común)*

Señalización (2)

- *Actualmente se usa SS7 (CCS)*
 - *Señalización asociada al circuito*
 - *Relativa a la llamada*
 - *Señalización no asociada al circuito*
 - *Consulta de tablas de enrutado*
 - *Servicios suplementarios*
 - *Desvíos de llamada...*
- *Paradigma de SS7*
 - *La inteligencia reside en la red (terminales 'tontos')*
 - *El acceso a la red determina los servicios disponibles*

Conmutación de paquetes

- *En conmutación de circuitos raramente se utilizaba todo el ancho de banda disponible.*
- *TDM ineficiente gestionando el uso de la red.*
- *El contenido del paquete determina la ruta.*

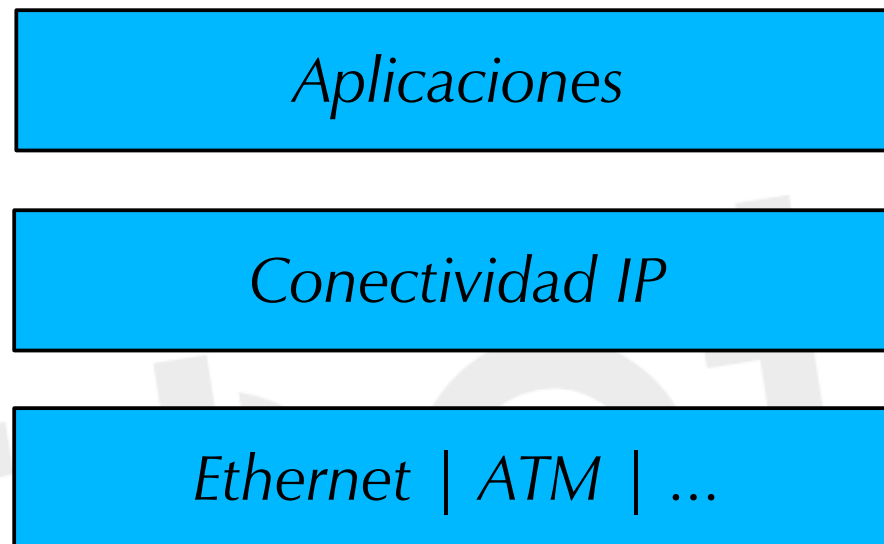


Conmutación de paquetes VS. Conmutación de circuitos

- *C. de circuitos*
 - *Más rápida*
 - *No se examina el contenido de los paquetes*
- *C. de paquetes*
 - *Mejor gestión de recursos*
 - *Precio*

El paradigma IP

- Su ÚNICO propósito es proporcionar conectividad.
- La red es independiente de la tecnología subyacente.
- Las aplicaciones pueden utilizar una infraestructura común IP.

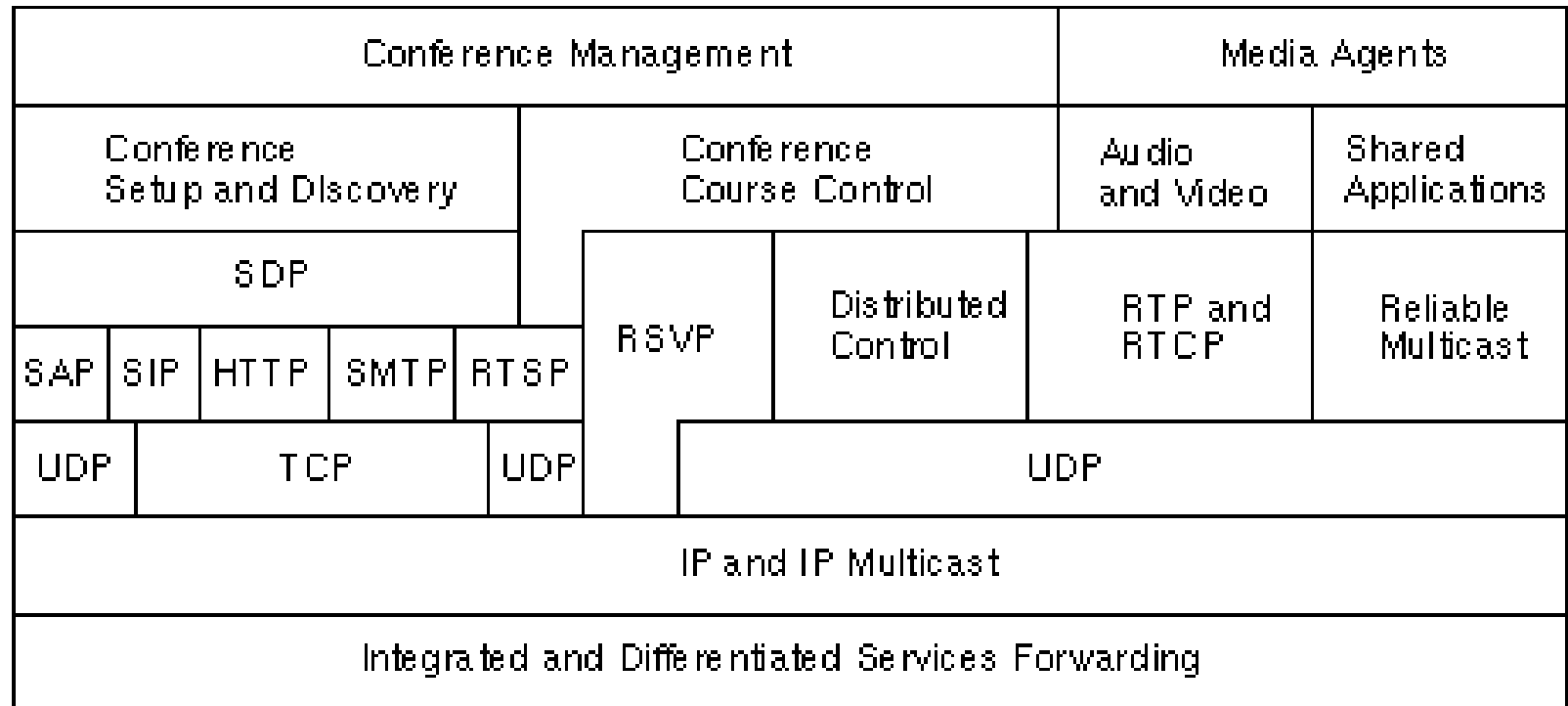


El paradigma IP (2)

- *Protocolos de extremo a extremo*
 - *IP solo 'lleva' cosas*
 - *La INTELIGENCIA esta en los extremos*
 - *Internet es idiota :)*
- *Justo lo contrario que en la telefonía tradicional...*

Arquitectura de Conferencias Multimedia en Internet

Arquitectura de Conferencias Multimedia en Internet



- *Protocolo de transporte en Tiempo Real.*
- *Requerimiento de aplicaciones con retardo ~ 0 .*
- *Internet es un medio hostil*
 - *Latencias*
 - *Jitter*
- *Para solucionarlo:*
 - *Timestamps*
 - *Números de secuencia*
- *Si tenemos varios streams de audio/vídeo, es necesaria la sincronización*
 - *RTCP*
 - *Asocia los timestamps con un RealTime Clock*

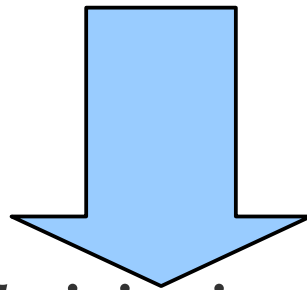
- *Session Announcement Protocol*
- *Sirve para 'anunciar' una sesión multimedia*
 - *“Hoy a las 8, película de noseke...”*
 - *Como la revista de la TV*
- *No se encarga de describir la sesión, para eso SDP*

- *Session Description Protocol*
- *Contiene toda la información que un usuario puede necesitar para unirse a una sesión multimedia.*
- *Ofrece la siguiente información*
 - *IP para conectarse a la sesión*
 - *Codecs soportados*
 - *Información descriptiva*
 - *...*

SIP: Session Initiation Protocol

Recapitulando...

- *Hasta ahora sabemos hacer 2 cosas*
 - *Anunciar una sesión multimedia*
 - *Describirla*
- *Pero... ¿como indicamos a alguien que se una?*
 - *Hay que INVITARLE a inicial una sesión*



SIP: Session Initiation Protocol

- *Para cubrir la carencia de no poder iniciar una sesión multimedia con alguien, surgió SIP.*
- *Estándar de la IETF, recogido en el RFC3261 (SIPv2)*
- *'Merge' entre*
 - *SIPv1 (Session Invitation Protocol)*
 - *SCIP (Simple Conference Invitation Protocol)*

Funcionalidades

- *SIP proporciona un mecanismo para iniciar, modificar y finalizar una sesión.*
- *Independiente del tipo de sesión multimedia y de su descripción.*
 - *Podemos invitar a alguien a una partida online de mus mediante SIP, utilizando MGD (Mus Game Description Protocol) para describir la sesión. XD*
- *Movilidad del usuario*
 - *Necesidad de conocer su localización.*
 - *SIP URLs: identificar a usuario SIP.*
sip:saghul@irontec.com
 - *Los usuarios registran su ubicación en el servidor.*

- *User-Agent: entidad con la que interactúa el usuario.*
 - *Teléfono SIP*
 - *Softphone*
- *Servidor Proxy: servidor que gestiona las invitaciones a las sesiones*
 - *Sabe donde esta el usuario destino, así que le enruta el mensaje.*
- *Registrar: servidor que acepta peticiones de registro, y guarda la ubicación del usuario.*
- *Location Server: no es una entidad SIP, pero es necesario para localizar al usuario.*
- *Normalmente los 3 anteriores son el mismo software.*

Porqué SIP mola

- *Diferencia entre el establecimiento y la descripción de la sesión*
 - *Extensible*
- *Protocolo de extremo a extremo*
 - *Un usuario ES DUEÑO DE SU SESIÓN*
 - *Paradigma IP vs. Paradigma SS7*
- *Favorece la interoperabilidad*
 - *El 'core' es ""relativamente"" sencillo: 6 métodos*
 - *Funcionalidades adicionales mediante extensiones*
- *Es escalable*
 - *La inteligencia esta en los extremos*
 - *La red guarda muy pocos datos del estado*

Funcionamiento de SIP

- *INVITE*
 - *Invita a un usuario a una sesión multimedia*
 - *Modifica una sesión multimedia existente*
- *ACK*
 - *Proporciona un 3-way-handshake en el INVITE, sirve para confirmar la recepción de una respuesta final a un INVITE*
- *CANCEL*
 - *Cancela una transacción en curso*
- *BYE*
 - *Se utilizan para abandonar una sesión*
- *REGISTER*
 - *Sirven para informar al servidor de la ubicación del usuario*
- *OPTIONS*
 - *Nos permite consultar qué métodos soporta un usuario.*

Transacciones Cliente-Servidor

- *Un cliente GENERA peticiones.*
- *Un servidor RECIBE peticiones.*
- *El UA que genera peticiones se conoce como UAC: User Agent Client.*
- *El UA que responde a las peticiones se conoce como UAS: User Agent Server.*
- *Una petición, junto con las respuestas que genera, es una TRANSACCIÓN.*

Respuestas SIP

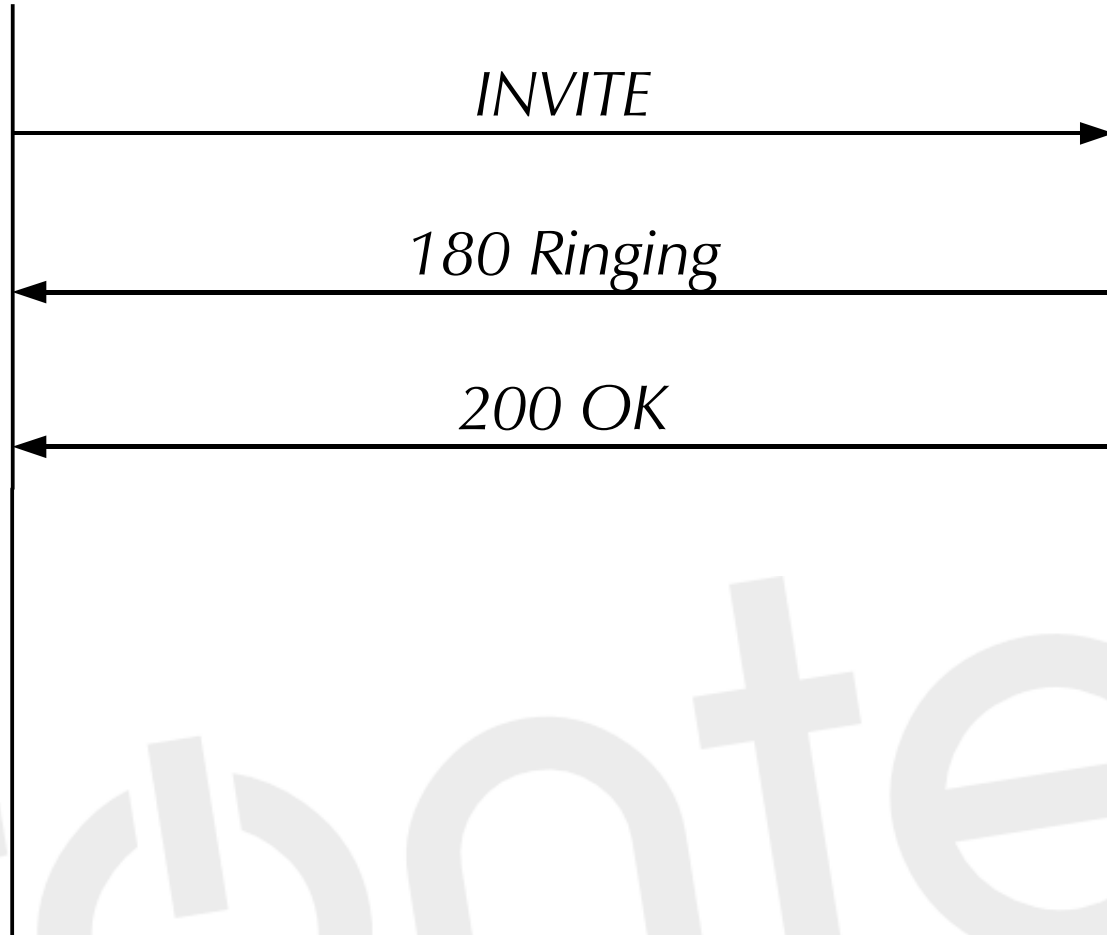
- *100 – 199: provisional e informativa*
- *200 – 299: afirmativa*
- *300 – 399: redirección*
- *400 – 499: error del cliente*
- *500 – 599: error del servidor*
- *600 – 699: fallo global*

- *Las respuestas incluyen un mensaje descriptivo, pero lo importante es el código numérico.*

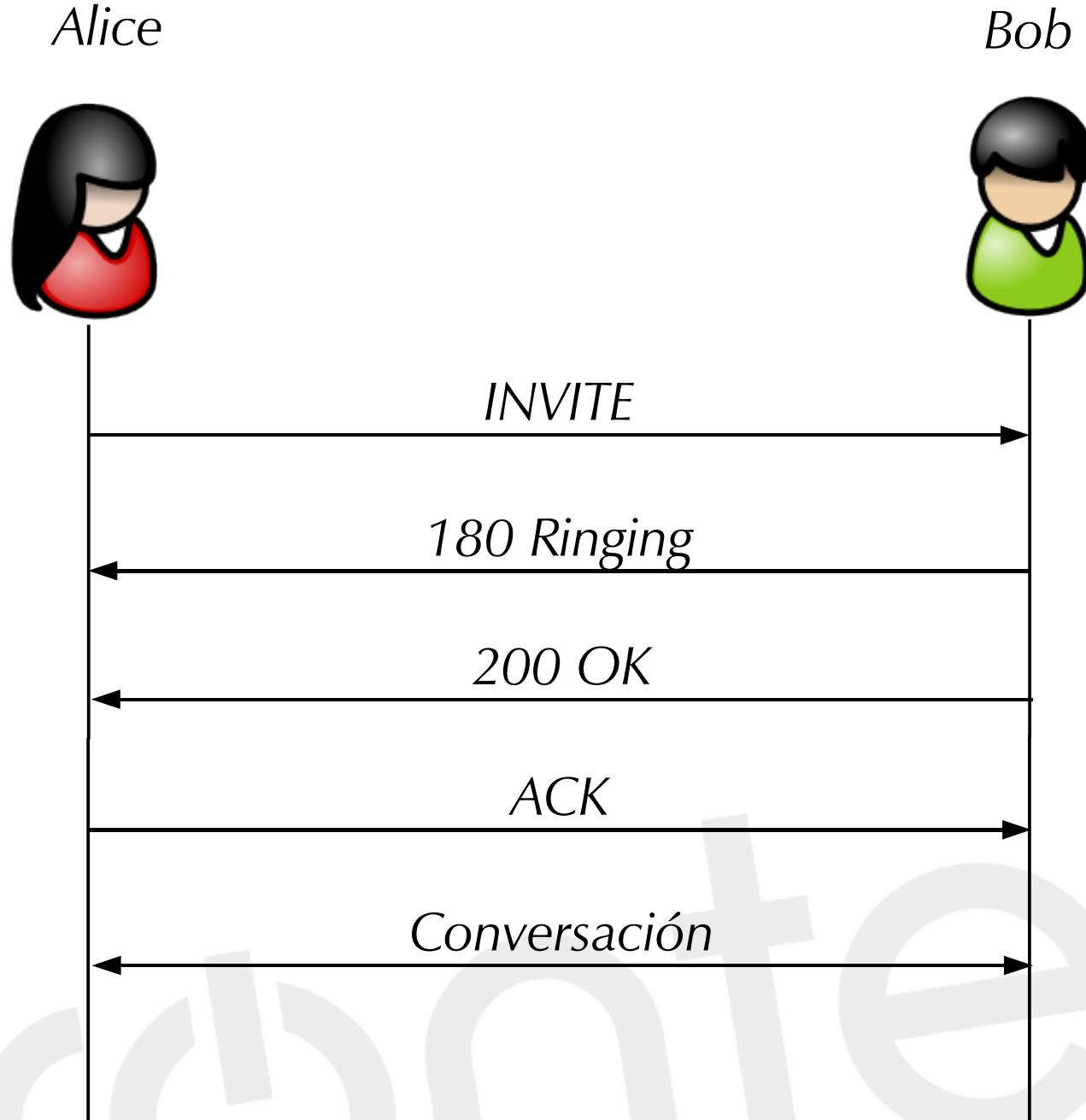
INVITE

Alice

Bob

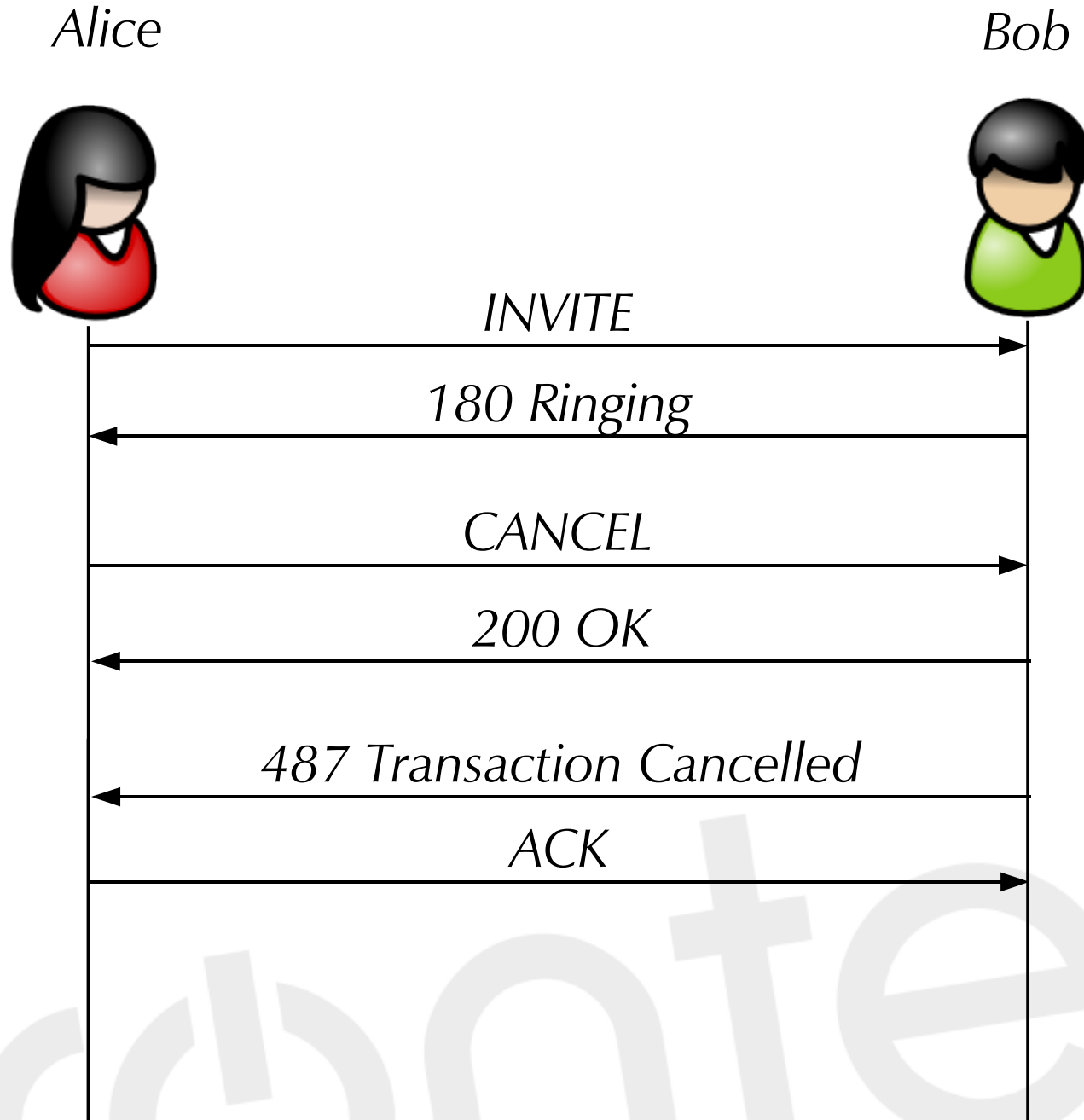


ACK

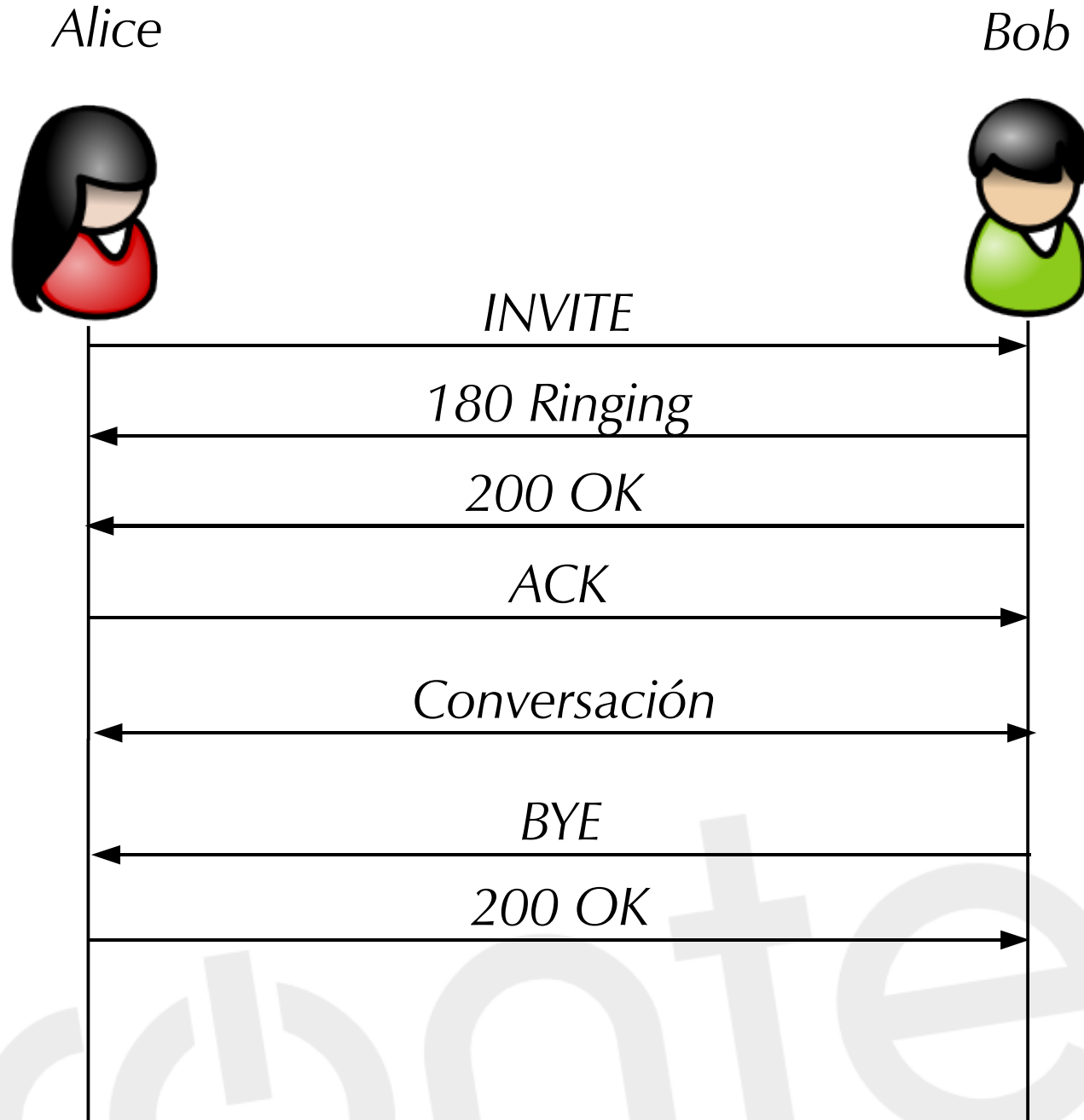


- *INVITE es el único método que utiliza 3 way handshake.*
- *El resto de mensajes esperan una respuesta veloz, pero en el caso del INVITE, esta puede tardar.*
- *El UAC manda al UAS un ACK, indicando que ha recibido su respuesta.*
- *Aseguramos el correcto establecimiento de la sesión sobre un medio no fiable: UDP*

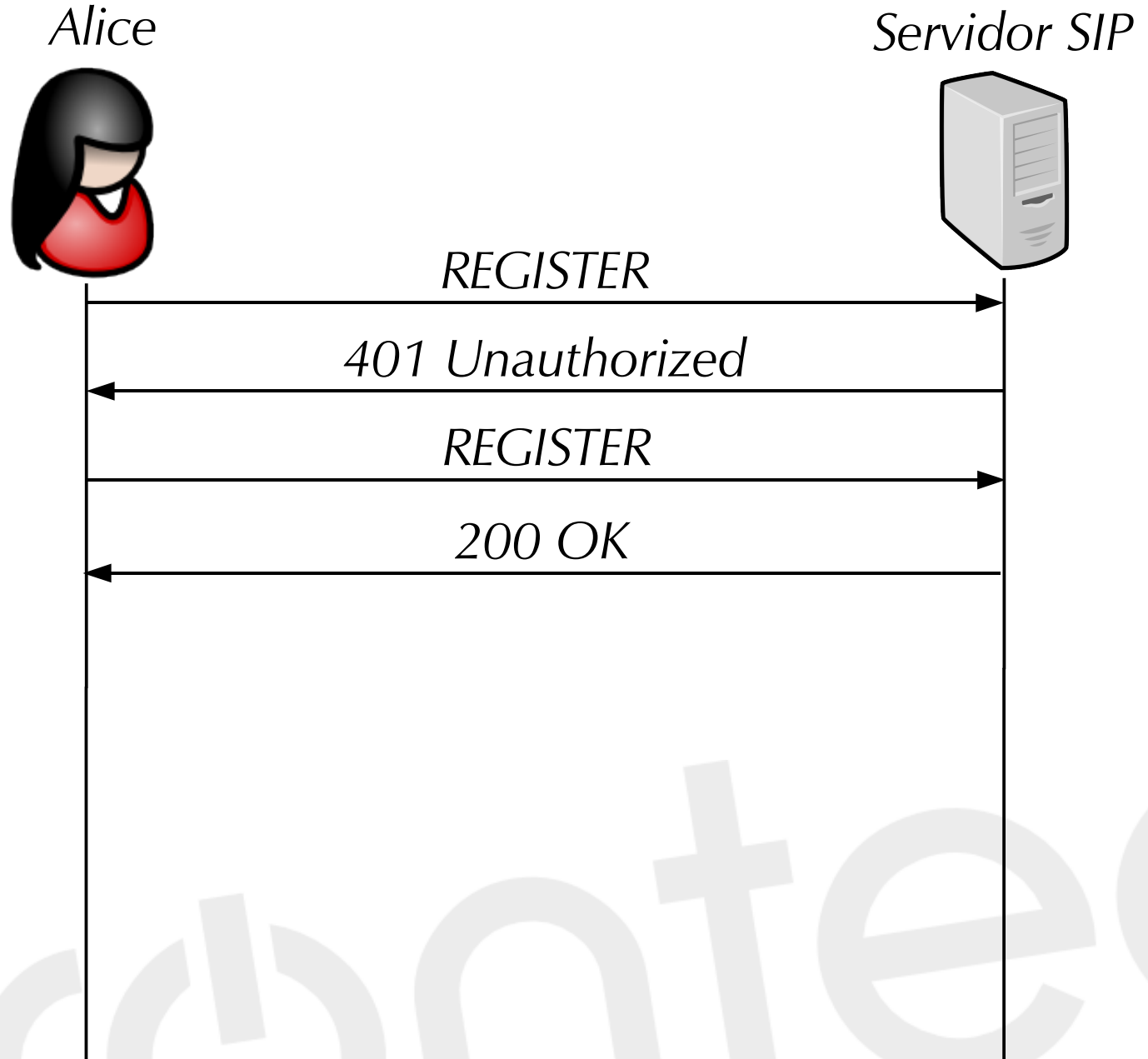
CANCEL



BYE



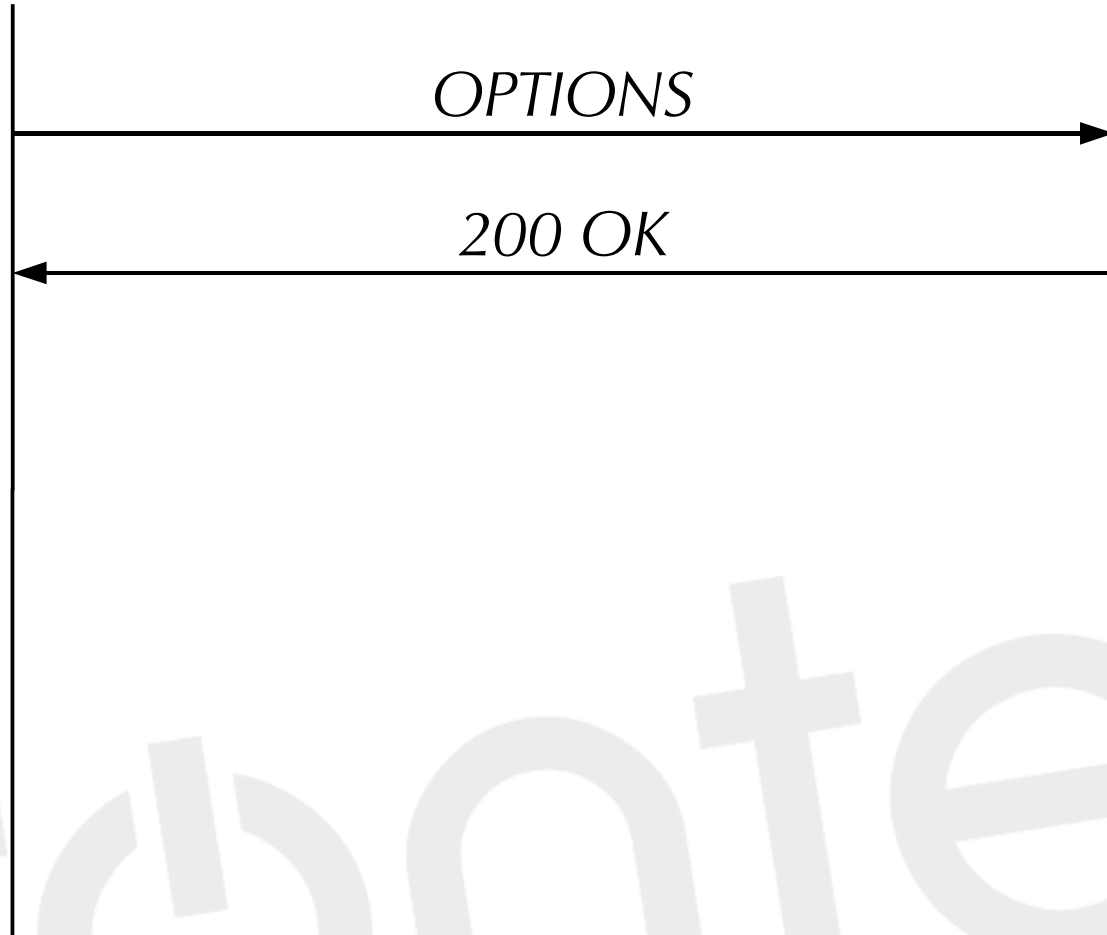
REGISTER



OPTIONS

Alice

Bob



- *From*
 - *Identifica al que origina una petición.*
- *Call-ID*
 - *Representa una relación entre 2 dispositivos SIP, relacionando un INVITE y todas las transacciones asociadas.*
- *Contact*
 - *Incluye una SIP URL, indicando donde se puede contactar con el usuario.*
- *To*
 - *Identifica al receptor de una petición.*
- *Vía*
 - *Contiene todos los proxys que han gestionado una petición.*
 - *Hace que las respuestas sigan el mismo camino que las peticiones*

Ejemplo de INVITE

INVITE sip:bob@biloxi.example.com SIP/2.0

Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9

Max-Forwards: 70

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

To: Bob <sip:bob@biloxi.example.com>

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 INVITE

Contact: <sip:alice@client.atlanta.example.com;transport=tcp>

Content-Type: application/sdp

Content-Length: 151

Ejemplo de INVITE (2)

SIP/2.0 180 Ringing

Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

To: Bob <sip:bob@biloxi.example.com>;tag=8321234356

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 INVITE

Contact: <sip:bob@client.biloxi.example.com;transport=tcp>

Content-Length: 0

SIP/2.0 200 OK

Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

To: Bob <sip:bob@biloxi.example.com>;tag=8321234356

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 INVITE

Contact: <sip:bob@client.biloxi.example.com;transport=tcp>

Content-Type: application/sdp

Ejemplo de INVITE (3)

ACK sip:bob@client.biloxi.example.com SIP/2.0

Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bd5

Max-Forwards: 70

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

To: Bob <sip:bob@biloxi.example.com>;tag=8321234356

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 ACK

Content-Length: 0

Ejemplo de INVITE (4)

BYE sip:alice@client.atlanta.example.com SIP/2.0

Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7

Max-Forwards: 70

From: Bob <sip:bob@biloxi.example.com>;tag=8321234356

To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 BYE

Content-Length: 0

SIP/2.0 200 OK

Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7

;received=192.0.2.201

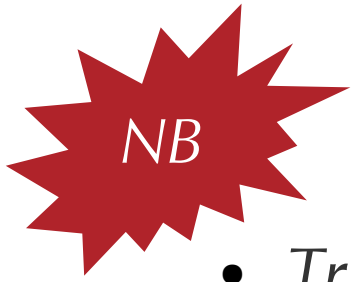
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356

To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 BYE

Content-Length: 0



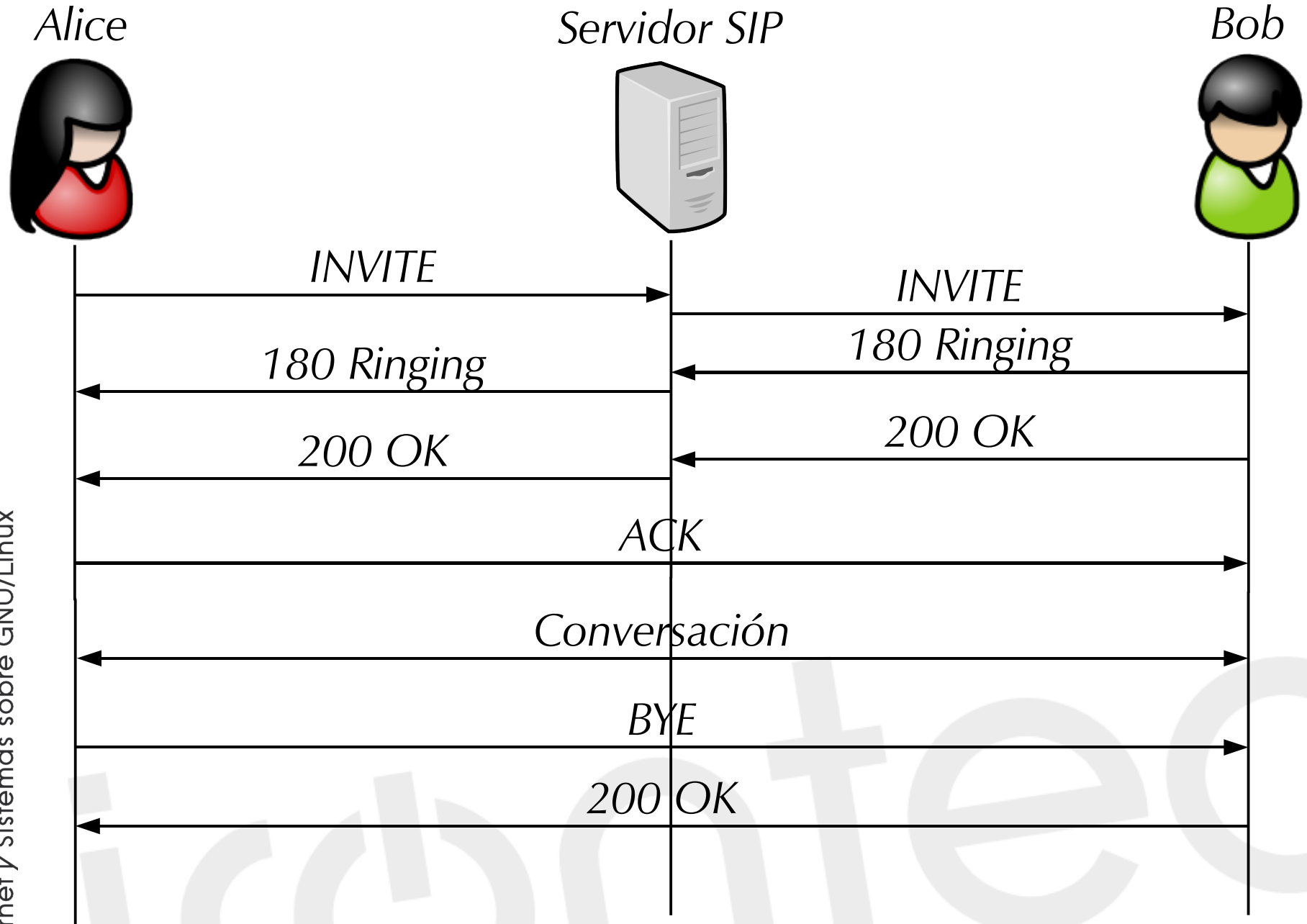
Conceptos importantes: Transacción y Diálogo

- *Transacción*
 - *Una petición + respuesta, Si la respuesta es afirmativa (INVITE + 200 OK)*
 - *Una petición + respuesta negativa + ACK (INVITE + 404 Not Found + ACK)*
 - *Identificado unívocamente por el 'branch' de la cabecera Vía.*
- *Diálogo*
 - *Concepto de 'llamada'*
 - *Identificado unívocamente por el From tag, To tag y Call-ID.*

Tipos de proxys SIP

- *Stateful Proxy*
 - *Su ámbito es la transacción.*
 - *No entiende de diálogos, pero sí de transacciones.*
- *Stateless Proxy*
 - *No guardan ningún tipo de estado.*

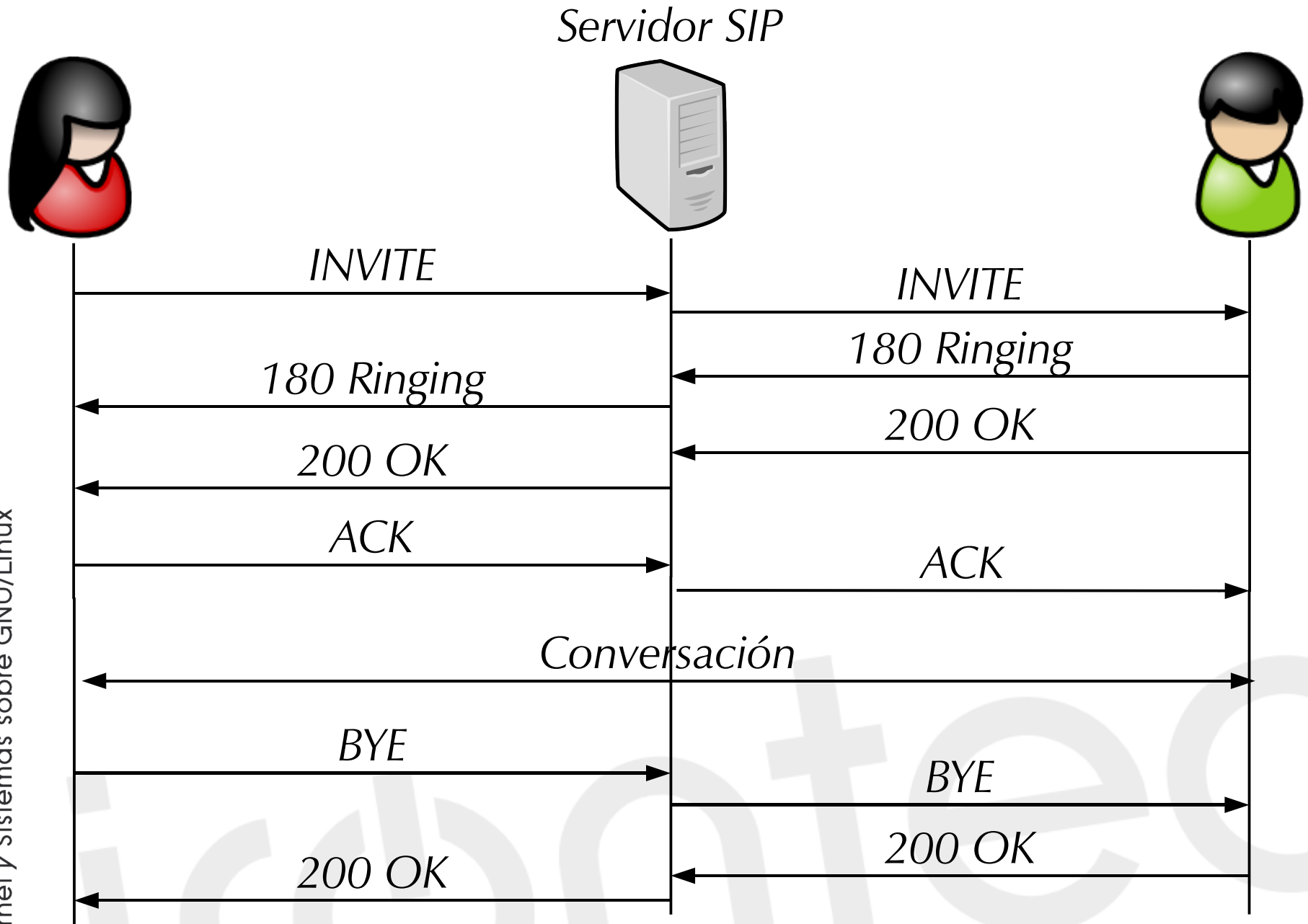
Routing de mensajes SIP



Routing de mensajes SIP (2)

- *Después del 200 OK, Alice ya sabe donde esta Bob (Contact)*
- *Las transacciones siguientes (ACK y BYE-200 OK) van directamente de extremo a extremo.*
- *Podemos alterar este comportamiento con las cabeceras Record-Route y Route*
 - *Si queremos facturar, queremos estar al tanto de la señalización...*

Routing de mensajes SIP (3)



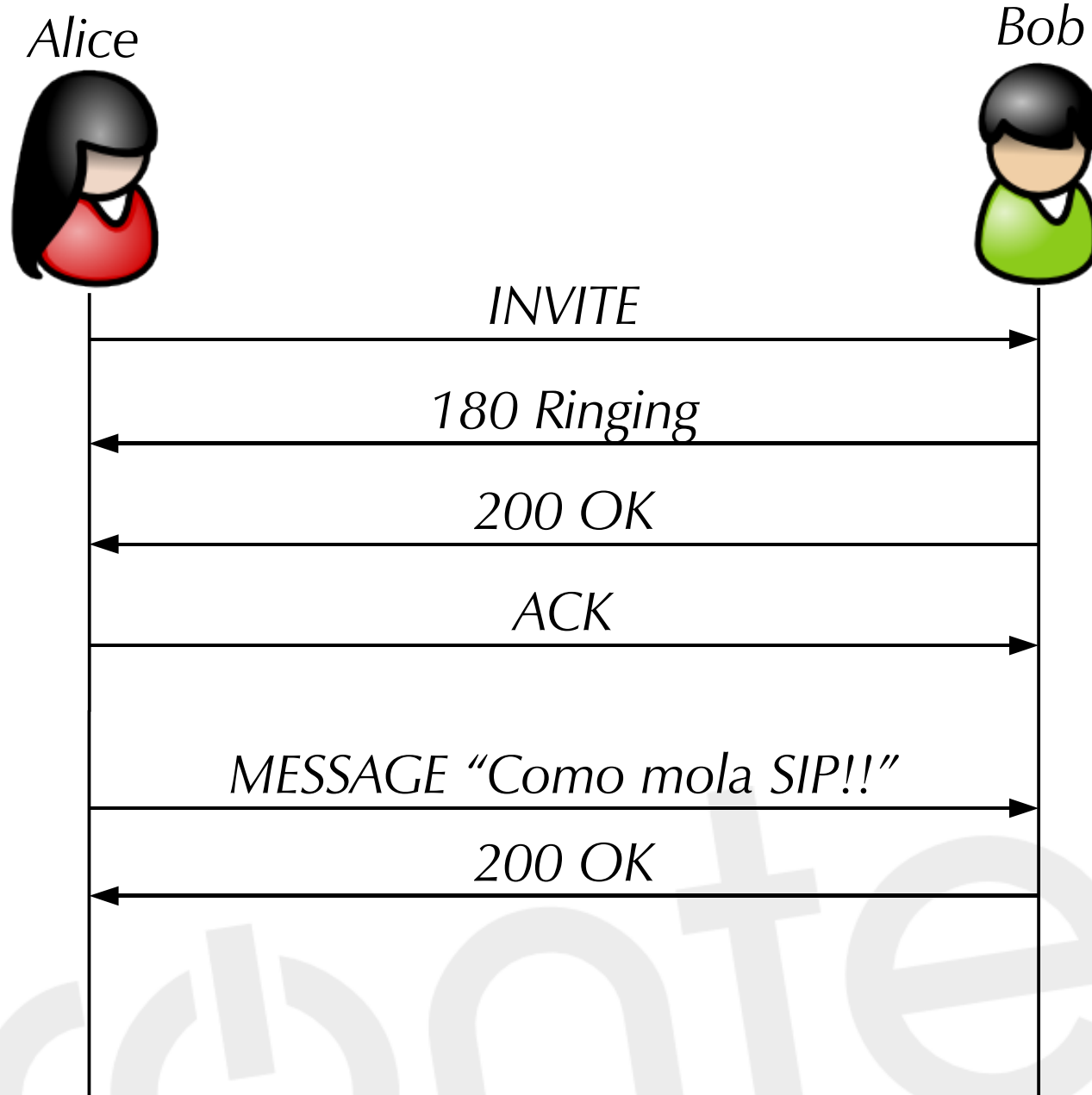
Routing de mensajes SIP (4)

- *Cada proxy que quiere quedarse 'en medio' añade una cabecera Record-Route al invite que pasa a través de él.*
- *Las cabeceras se mantienen y se envían de vuelta en la respuesta.*
- *Las siguientes transacciones se generan con la cabecera Route (en orden inverso que las Record-Route).*
- *En mensaje se envía al proxy que indica su primera cabecera Route y el proxy la elimina.*

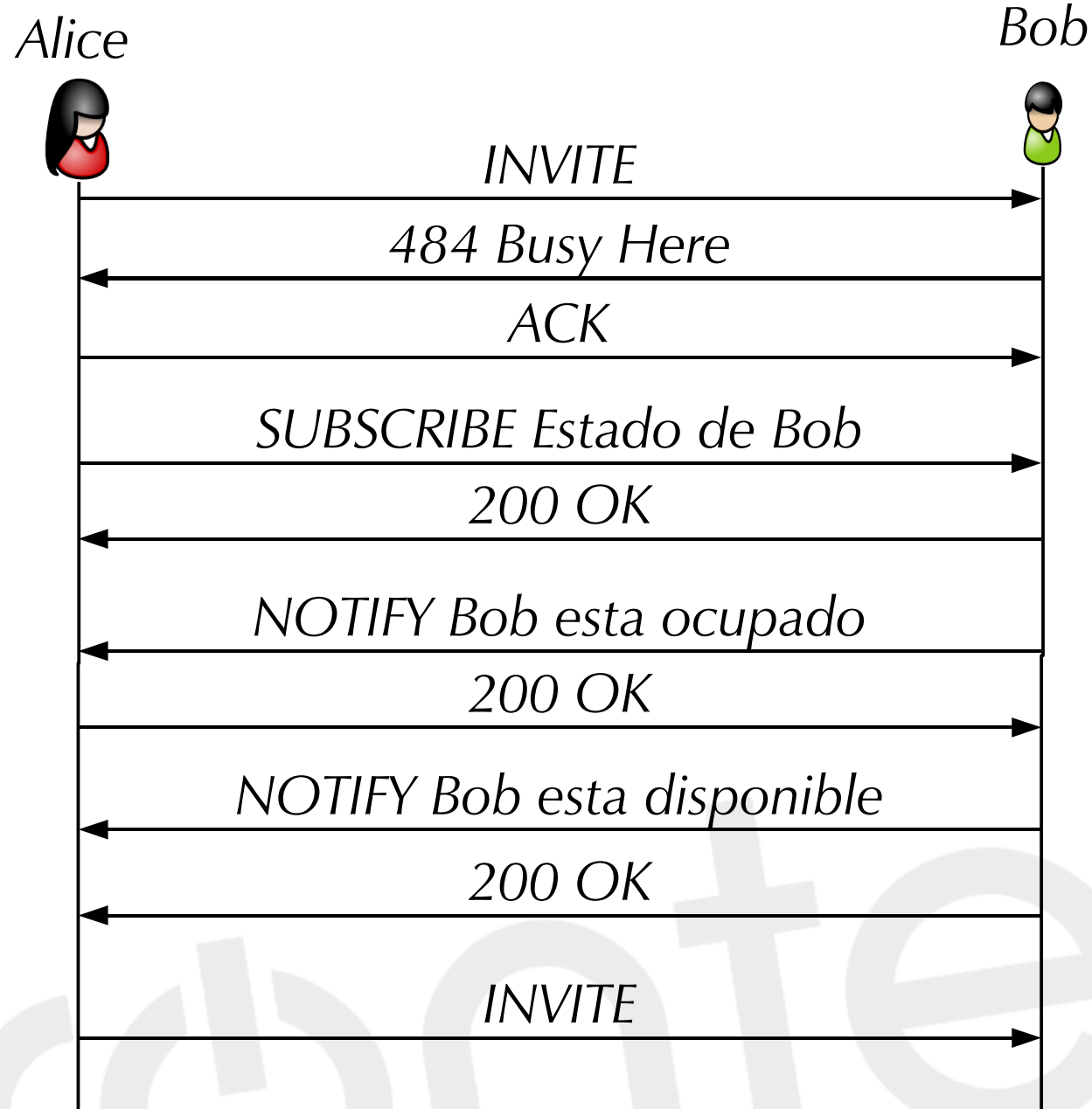
Extendiendo el RFC3261

- *Con lo visto hasta ahora, solo podemos hacer y recibir llamadas :-O*
- *PEEEERO, SIP se diseñó para ser extensible, por lo que se le han añadido servicios mediante extensiones al protocolo.*
 - *Mensajería*
 - *Notificaciones Asíncronas de Eventos*
 - *Transferencia de sesiones*
 - *...*

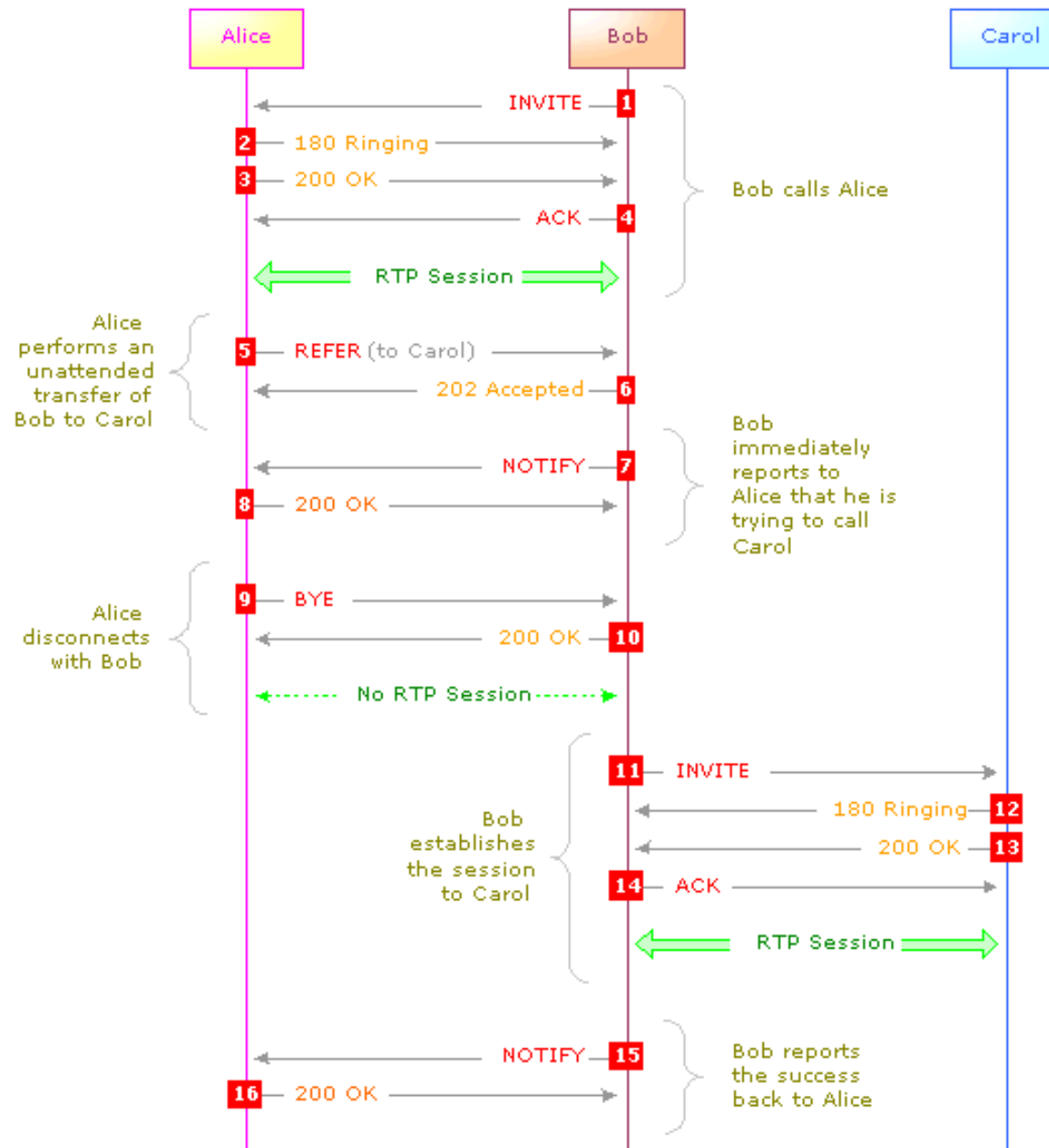
Mensajería Instantánea



Notificaciones Asíncronas de Eventos



Transferencia de sesiones



OpenSER

- *OpenSER es un fork de SER (SIP Express Router)*
- *SER fue creado en el FhG FOKUS de Berlin y liberado en 2002*
- *En 2005, 2 de sus principales desarrolladores (Daniel-Constantine Mierla y Bogdan-Andrei Iancu) decidieron crear el fork OpenSER, ya que había distintos puntos de vista en distintos aspectos de la gestión del proyecto*
- *Volvieron a su país natal (Rumanía) y fundaron Voice-System*
- *El desarrollo de OpenSER creció muy rápidamente*
 - *El primer servidor SIP Open Source con soporte para SIP sobre TLS (28 de octubre de 2005)*

SER vs OpenSER

- *OpenSER está siendo desarrollado más activamente*
- *En realidad, a efectos de configuración, son MUY parecidos*
- *OpenSER dispone de una muy buena documentación*
- *La comunidad de OpenSER es muy grande y muy activa*

¿Qué es OpenSER?

- *OpenSER es un Stateful Proxy (transaction stateful, dialog stateless*)*
- *¿Qué tiene que hacer?*
 - *Validar una petición*
 - *Preprocesar la información de enrutado.*
 - *Determinar el receptor de la petición*
 - *Encaminar la petición hacia su destino*
 - *Procesar las respuestas*

¿Que NO es OpenSER?

- *OpenSER NO es un Asterisk molón.*
- *OpenSER no procesa audio.*
- *OpenSER es un SIP Proxy/Registrar/Location Server, NO un B2BUA*
- *B2BUA*
 - *Back 2 Back User Agent*
 - *Actúa como UAC por un lado, y como UAS por el otro*
 - *Mantiene estado de las llamadas*
 - *Ofrece servicios al usuario (buzón de voz, ...)*
 - *Gestiona el media (traducción de codecs, ...)*
 - *P.e. Asterisk*
 - *1 llamada = 2 diálogos SIP (Asterisk hace un bridge)*

Configuración de OpenSER

Estructura de *openser.cfg*

```
# ----- global configuration parameters -----  
debug=3          # debug level (cmd line: -dddddddddd)  
fork=yes  
log_stderr=no    # (cmd line: -E)  
children=4  
port=5060
```

Parámetros
globales de
configuración

```
# ----- module loading -----  
#set module path  
mpath="//lib/openser/modules/"
```

Carga de
módulos

```
loadmodule "sl.so"  
loadmodule "tm.so"  
loadmodule "rr.so"  
loadmodule "maxfwd.so"  
loadmodule "usrloc.so"  
loadmodule "registrar.so"  
loadmodule "textops.so"
```

Estructura de openser.cfg (2)

```
# ----- setting module-specific parameters -----
```

```
modparam("usrloc", "db_mode", 0)
```

```
# ----- request routing logic -----
```

```
# main routing logic
```

```
route{
```

```
...
```

```
}
```

```
route[1] {
```

```
...
```

```
    t_relay();
```

```
}
```

```
onreply_route[1] {
```

```
...
```

```
}
```

```
failure_route[1] {
```

```
...
```

```
}
```

Diseñando la lógica

- *OpenSER es nuestra 'caja de herramientas'*
- *Tenemos que pensar QUÉ queremos hacer con nuestro proxy*

- *OpenSER trabajando como stateful proxy*
- *No hay almacenamiento persistente de usuarios*
- *Funcionamiento*
 - *Guarda la localización de los usuarios en memoria*
 - *Simplemente enruta los mensajes*

Contextos de enrutado

- *Transacción*
 - *Si enrutamos con `t_relay()` estamos en stateful mode*
 - *Podemos gestionar las respuestas en `onreply_route[]` y `failure_route[]`*
- *Diálogo*
 - *Si no insertamos una cabecera Record-Route no 'vemos' las siguientes peticiones (gracias a la cabecera Contact han 'aprendido' donde esta cada uno)*
 - *Con la función `loose_route` enrutamos las peticiones que contengan un conjunto de rutas (cabeceras Route)*
 - *Hay que ser más severos: ¿tiene To tag?*

Ejemplo 1.1

- *Enrutado stateless*
 - *¿Por donde pasan las respuestas?*
- *Enrutado stateless SIN Record-Route*
 - *¿Donde están las respuestas?*

- *Autenticación de REGISTER*
- *Usuarios en MySQL*
- *Fichero openserctlrc*
- *Funciones check_to y check_from*
- *Gestión de CANCEL*
- *Múltiples dominios*

- *Conexión con Asterisk Media Server*
 - *Va a ser nuestro PSTN gateway*
- *ACLs*

Ejemplo 4

- *Desvíos de llamada*
- *Voicemail*
- *Pseudo variables*
- *AVPs*

Uso de *openserctl*

- *Añadir un usuario:*
 - *openserctl add 200 1234 saghul@gmail.com*
- *Consultar usuarios online*
 - *openserctl online*
- *Gestión de ACLs*
 - *acl show <usuario>*
 - *acl grant <usuario> <grupo>*
 - *acl revoke <usuario> [<grupo>]*

- *OpenSER solo gestiona señalización.*
- *Con OpenSER es posible implementar soluciones al NAT far-end*
 - *Detectamos que un usuario esta detrás de NAT*
 - *Modificamos sus mensajes para 'evitarlo'*
 - *Uso de Media Proxys*
 - *RTPProxy*
 - *MediaProxy*

- *SIP es un protocolo sencillo en su forma pero MUY complicado en su funcionamiento*
- *Tendencia hacia estándares abiertos, interoperabilidad*
- *Aunque su aplicación principal es la VoIP, SIP tiene un futuro prometedor*
- *Es necesario conocer SIP a fondo para una buena instalación de OpenSER*
 - *Al gestionar el protocolo a tan bajo nivel, tenemos todo el control.*

Consejos... o algo así

- *Ngrep es tu nuevo muy mejor amigo*
 - *ngrep -d any -P '' -W byline -T port 5060*
- *Disponer de un entorno de pruebas adecuado*
 - *Seriously!*
- *No pienses que estas loco por leer RFCs*
- *NO uses el generador de configuraciones de SIPWise*
 - *Es preferible empezar desde abajo e ir complicándolo cada vez más*
- *Recuerda que OpenSER es un proxy SIP*
 - *Aunque creas que puedes, no hagas que OpenSER haga cosas que no le corresponden.*

Referencias y lecturas recomendadas

- *Building Telephony Systems with OpenSER* (Flavio E. Goncalves)
- *SIP Demystified* (Gonzalo Camarillo)
- *Lista SIP-ES*
 - <http://groups.google.com/group/sip-es/>
- *Lista OpenSER-ES*
 - <http://www.openser.org/cgi-bin/mailman/listinfo/users-es>
- *Listado de RFCs y Drafts*
 - <http://groups.google.com/group/sip-es/web/rfc-y-drafts>
- *Ejemplos de SIP con dibujos y demás*
 - <http://www.tech-invite.com/>
- *Por si se me olvida algo interesante*
 - <http://del.icio.us/saghul/sip>
 - <http://del.icio.us/saghul/openser>



Creative Commons BY-NC-SA

<http://creativecommons.org/licenses/by-nc-sa/2.5/es/>

¡¡Gracias!!

BYE sip:gente@192.168.1.120:23834;rinstance=c1e2af826bf4f26c SIP/2.0.

Via: SIP/2.0/UDP 192.168.1.119:14060;branch=z9hG4bK-d87543-c32b1e47453b7c41-1--d87543-;rp

Max-Forwards: 70.

Route: <sip:192.168.1.104;lr;ftag=333c3224>.

Contact: <sip:saghul@192.168.1.119:14060>.

To: "userA" <sip:gente@192.168.1.104>;tag=be72f51c.

From: "userB" <sip:saghul@192.168.1.104>;tag=333c3224.

Call-ID: NTA0YTFmYjFkMmQzY2I1NTdiZDMwZGY4ODJkNzc1NTQ..

CSeq: 2 BYE.

Reason: SIP;description="User Hung Up".

Content-Length: 0.