Capítulo 1

Diseño

Contar que es una arquitectura modular y comparación con una arquitectura cognitiva.

1.1. Arquitectura cognitiva

Explicación de los diferentes bloques de una arquitectura cognitiva: Repositorio, ejecutor, optimizador....

1.2. Arquitectura implementada

Características de nuestra arquitectura: modular, homogénea, escalable (hasta 7 nodos) lo que le hace especialmente compatible a cambios en el futuro ya que los procesos están fuertemente marcados e independientes.

1.2.1. Comparación ambas arquitecturas

Clasificar nuestros metodos en los bloques de una arquitectura cognitiva.

1.3. Detalles de implementación

Servicio vs activity.... multi app...

1.4. Cómo montar una aplicación sobre el servicio cognitivo, definición de la API

Tipos de mensajes que se intercambian, y como tienen que ir rellenos estos.

1.5. Procesos de red

1.5.1. Registro de una aplicación en el servicio cognitivo

Para que una aplicación pueda registrarse de forma correcta en el servicio debe cumplir un cierto *handshaking* consistente en el intercambio de tres mensajes, el primero servirá para que el servicio nos tenga en cuenta, el segundo para informarle de nuestros parámetros de

aplicación y el tercero será una confirmación del servicio hacia la aplicación informando de todos los parámetros del servicio respecto a nuestra aplicación.

Tras la petición de *bind*, que inicializará el servicio si no estuviese arracando ya, obtendremos el *Messenger* del servicio, indispensable para poder comunicarnos con él. Tras esto debemos mandar un primer mensaje para registrar nuestra aplicación en el servicio, donde éste, registrará nuestro messenger para habilitar la comunicación en sentido contrario y nos incluirá en su lista de aplicaciones registradas. Los detalles de este mensaje son:

Campos mensaje				
What	*arg1*	*arg2*	*obj*	*replyTo*
1 = REGISTER_CLIENT	No usado/indiferente	No usado/indiferente	No usado/indiferente	Messenger de la aplicación

Tabla 1.1: Mensaje registro cliente

En el siguiente mensaje que debemos enviar, informaremos acerca de nuestros parámetros de aplicación como es: nuestro papel, un entero cuyo valor 0 corresponde a papel secundario y el valor 1 corresponde al papel primario y nuestro código de aplicación, útil para que el servicio nos entregue sólo los mensajes que nos atañen.

Campos mensaje				
What	*arg1*	*arg2*	*obj*	*replyTo*
3 = REGISTER_EXCHANGE	0=secundario / 1=primario	No usado/indiferente	No usado/indiferente	Messenger de la aplicación
Extras				
key	*value*			
appCode	Una string con el código de aplicación			

Tabla 1.2: Mensaje intercambio parámetros aplicación

Este mensaje ha sido separado del anterior para poder reutilizarlo e introducirlo en el flujo descrito en 1.5.5. Con esta información el servicio dependiendo del punto en que se encuentre actuará de una forma u otra.

Punto de partida inicial, CWSN no establecida

Esta situación, (la más común) es cuando nuestra petición de *bind* ha arrancado el servicio y por lo tanto estamos en una situación inicial o hay aplicaciones ya montadas sobre el servicio pero no hay conectividad con otros nodos, en otras palabras, estamos solos en la red. (Hay otra forma de llegar a esta situación que veremos en 1.5.5, que no explicamos ahora para no enmarañar el texto).

En esta situación, configuraremos de nuevo la interfaz, actualizaremos los parámetros pasados en el anterior mensaje y lanzaremos nuevamente los procesos de registro en la red para intentar establecer una CWSN. Al finalizar éstos, el servicio nos devolverá (a todas las aplicaciones montadas) información acerca de todos los parámetros y el estado actual de la red. A saber:

Estado de la interfaz Un entero que representa el estado de la interfaz 0 = Down, 1 =

Idle (no en red), 2 = Idle (en red), 3 = conectando (estado visto sólo en *Bluetooth*), 4 = Enviando, 5 = Reciviendo.

Interfaz Un entero cuyo valor 0 representa a *Bluetooth*, el valor 1 representa a *WiFi* y -1 representa a una inerfaz desconicida (útil para casos de error y cambios de contexto)

Papel del nodo Un entero cuyo valor 0 = secundario y 1 = primario

Tipo de nodo Un entero cuyo valor 0 = normal, 1 = coordinador y 2 = coordinador temporal en *Bluetooth* (no usado en estos momentos)

Periodo tarea cognitiva Un *double* que representa los segundos que transcurren entre ejecuciones de la tarea cognitiva

Nombre del nodo en la red Una *cadena de carácteres* con el nombre del nodo en la red

El detalle del mensaje, queda:

Campos mensaje				
What	*arg1*	*arg2*	*obj*	*replyTo*
3 REGISTER_EXCHANGE	Estado de la interfaz	Interfaz	**False**	No usado/indiferente

Extras

key	*value*
nodeRole	Papel del nodo
nodeType	Tipo de nodo
periodTask	Periodo tarea cognitiva
nodeName	Nombre del nodo en la red

Tabla 1.3: Mensaje finalización de *handshaking*: CWSN no establecida

CWSN previamente configurada

Puede darse el caso que en este punto del proceso, ya haya aplicaciones montadas sobre el servicio y estén cooperando en una CWSN, en este caso el servicio no tiene que configurar nada y se delimita a ver si satisfacer las necesidades que le acaba de transmitir la nueva aplicación que acaba de registrar, es decir, cambiará el papel del nodo a primario si este era secundario, un cambio en sentido contrario será ignorado. Una vez hecho esto nos devolverá (en exclusiva) los parámetros del servicio (ver lista página 2) más estos que listamos a continuación:

Cógigo de la aplicación Un *cadena de carácteres* que representa el código de la aplicación

Lista nodos TODO TODO TODO en el código

El detalle del mensaje queda [1]:

[1]Notar que el campo del mensaje *obj* viene informado con el valor booleano *True* lo que nos ayuda a distinguir si previamente el servicio ya cooperaba en una red, a parte que esta respuesta es casi inmediata, a diferencia de la anterior.

Campos mensaje				
What	*arg1*	*arg2*	*obj*	*replyTo*
3 REGISTER_EXCHANGE	Estado de la interfaz	Interfaz	**True**	No usado/indiferente
Extras				
key	*value*			
nodeRole	Papel del nodo			
nodeType	Tipo de nodo			
periodTask	Periodo tarea cognitiva			
nodeName	Nombre del nodo en la red			
codeName	Código de la aplicación			
nodeNamesList	Array de *cadena de carácteres* con los nombres de los nodos			
nodeIdsList	Array de enteros con los identificadores de los nodos			

Tabla 1.4: Mensaje finalización de *handshaking*: CWSN establecida

1.5.2. Configuración de la interfaz de comunicación

WiFi

Bluetooth

1.5.3. Registro de un nodo en la red

1.5.4. Salida de un nodo de la red

WiFi

voluntaria / no voluntaria

Bluetooth

voluntaria = no voluntaria

1.5.5. Actualización de parámetros en la red

Hablar que en wifi con la entrada se actualizan y en bt no.

1.5.6. Intercambio de mensajes

1.5.7. Sensado del entorno

1.5.8. Cambios de contexto

Cambio de contexto con interfaz destino WiFi

Cambio de contexto con interfaz destino Bluetooth

References