Interfaz de conexión

Perspectiva: (cliente — servidor)

Comandos

Se dividen en tres etapas:

Etapa de inicio

- maquina_nueva (>)
- maquina_nueva (<)
- inicio (>)

Etapa de configuración

- configurar (<)

Etapa de funcionamiento

- solicitar (<)
- informar (>)

Flujo de comandos

Etapa de inicio

Primero al instalar el cliente se debe ingresar la IP y el puerto del servidor. Esta información va a guardarse en un archivo de configuración de cada cliente que debería estar protegido y no sufrir modificaciones por los usuarios del sistema.

Al iniciar el cliente por primera vez y encontrar el servidor, se conecta un socket TCP con el mismo, y manda un comando “maquina_nueva”. Como respuesta a éste, el servidor envía un comando “maquina_nueva” indicando un ID único para ese cliente. Ese ID será enviado en los comandos futuros. El ID recibido por el cliente se guarda en el archivo de configuración del mismo. Nótese que una misma máquina va a tener tantas instancias en el servidor como instalaciones tenga (es decir, si se instala en Windows y Linux tendrá una misma máquinas dos registros en el servidor con dos IDs distintos).

Estos dos comandos sólo ocurren cuando es la primera vez en que se inicia el programa cliente. Para cuando se inicie por segunda vez, el cliente encontrará en su archivo de configuración su ID y podrá funcionar normalmente.

Un tercer comando es enviado siempre que inicie el cliente, es el comando “inicio” el cual indica al servidor cuál es su archivo de configuracion actual y deja pie para que el servidor envié nuevas configuraciones, progresando hacia la etapa siguiente.

Etapa de configuración

El servidor envía al cliente cuál debe ser su configuración actual. Esta configuración debe ser flexible para permitir nuevas opciones en el futuro.

Lac configuración prevista incluye: establecer informes programados por cliente, especificar qué información se enviará al servidor,

establecer frecuencia de informes.

Tras recibir el comando “inicio”, el servidor establece una configuración por defecto en el cliente, calcula un hash con MD5, la almacena y luego envía al cliente con el comando “configurar”. El cliente guarda esta configuración en un archivo local (protegido), la ubicación del mismo es la misma para todas las instalaciones: en la raíz de la instalación.

Etapa de funcionamiento

Existen dos modos de funcionamiento:

Modo activo

El servidor envía un comando “solicitar” al cliente pidiendo que le envíe su estado actual.

El comando lleva asociado un ID de solicitud que el servidor crea, también se indica qué información de la máquina se solicita.

El cliente responde con un comando “informar” donde aparece tanto el ID de solictud del servidor, como la información requerida por el mismo.

Modo pasivo

El cliente envía un cmomando “informar” con la información de su estado actual sin que el servidor lo haya solicitado previamente (como ocurre en modo activo).

Aquí el cliente determina (a partir de su archivo de configuración) cuándo enviar este comando y con qué información. No se indica un ID

de solicitud, pues no responde a un comando “solicitar”, sino más bien a algo planificado (como ser una actualización programada o al inicio del sistema).

Nota: en ambos modos de funcionamiento cuando el cliente envía el comando “informar” asocia al mismo el hash de su configuración como cliente. El servidor calcula un hash de la configuración que él posee apra ese cliente y verifica si coincide con la que recibió. Ante cualquier discrepancia, el servidor envía la “correcta” configuración al cliente mediante el comando “configurar” (etapa de configuración). No obstante, es necesario notar que cuando ocurre esta diferencia en el modo pasivo (únicamente) el servidor también debe verificar si es correcto recibir este informe dada su configuración que él posee para ese cliente, en caso de que no sea correcto debería descartar el informe y enviar un comando “configurar”, en caso de que sea correcta simplemente debe enviar el comando “configurar” y aceptar el informe.

Detalles de comandos

Etapa de inicio

maquina_nueva (>)

Al iniciar por primera vez el cliente (o al iniciarlo y no hallar su archivo de configuración) se conecta con el servidor para solicitar su configuración (y su ID único).

> *{*
> *comando: “maquina_nueva”*
> *}*

maquina_nueva (<)

Respuesta del servidor ante el comando “maquina_nueva” indicando su ID. Luego el servidor debe enviar un comando “configurar” para establecer la configuración al nuevo cliente.

> *{*
> *comando: “maquina_nueva”,*
> *datos: {*
> *id: “id de la maquina”*
> *}*
> *}*

inicio (>)

El cliente indica al servidor que está listo para avanzar a las siguientes etapas.

> *{*
> *comando: “inicio”*
> *}*

Etapa de configuración

configurar (<)

El servidor informa la configuración correcta al cliente, éste debe sobreescribir su configuración local.

> *{*
> *comando: “configurar”,*

> *datos: {*
>
> *configuracion: { json de configuracion }*
>
> *}*
>
> *}*

Etapa de funcionamiento

solicitar (<)

El servidor solicita la información del cliente, indicando qué información desea del mismo.

> *{*
>
> *comando: “solicitar”,*
>
> *datos: {*
>
> *id_solicitud: “id de la solicitud”,*
>
> *informacion: [“procesador”, “memorias_ram”, “discos_duros”, “otro_componente...”],*
>
> *}*
>
> *}*

informar (>)

El cliente envía su información: tanto ante un solicitud (para cual agrega el id de solicitud, modo activo), o ante una condición en su configuración (modo pasivo). A su vez envía el “hash” de su archivo de configuración para que el servidor verifique si es o no correcto (en caso de no ser correcto, el servidor envía un comando “configurar” para reconfigurar el cliente).

> *{*

comando: “informar”,

datos: {

[id_solicitud: “id de la solicutud a la cual responde en modo activo”,]

hash_configuracion: “hash md5 de su archivo de configuración local”,

informacion: [

{

procesador: {

nombre: “”,

descripcion: “”,

fabricante: “”,

arquitectura: “”,

cantidad_nucleos: “int”,

cantidad_procesadores: “int”,

velocidad: “en MHz”,

tamanio_cache: “en kB”,

}

},

{

discos_duros: [

{

fabricante: “”,

modelo: “”,

numero_serie: “”,

tipo_interfaz: “SCSI, IDE”,

firmware: “”,

cantidad_particiones: “”,

```
            tamanio: “”,
        }
        ],
            memorias_ram: [
            {
            banco: “”,
            tecnolgia: “EPROM, VRAM”,
            fabricante: “”,
            numero_serie: “”,
            tamanio_bus_datos: “bits”,
            velocidad: “en MHz”,
            tamanio: “en B”
            }
        ]
        }
}
```

Estructura del archivo de configuración

El archivo almacenado en el cliente tiene como fin guardar el ID del cliente, además de qué información va a reportar (en modo pasivo) y cuándo lo hará.

```
{
id: “id del cliente”,
configuracion: {
    informes: [
            {
            id: “id del informe, decidido por el cliente e informado al
```

> *servidor”,*
> *informacion: [“procesador”, “memorias_ram”, “discos_duros”, “otro_componente...”],*
> *tipo: “programado|inicio_sistema|inicio_sesion|apagado”,*
> *hora: “hora del informe programado”*
> *}*
> *]*
> *}*
> *}*

Nota: la configuración más simple es no tener ningún informe programado, en este caso el servidor al enviar el comando “configurar” pasa como dato un json de configuración de la forma que está representado arriba (dentro de la propiedad “configuración” y debe incluir dentro la propiedad “informes” con valor igual a un arreglo vacío.

Desafíos a la integridad del sistema

- Borrado del archivo de configuración del cliente

Situación: en caso de que un cliente no encuentre su archivo de configuración, el mismo volverá a enviar un comnado “maquina_nueva” solicitando un nuevo ID, y el servidor no tiene forma de saber si se trata de una máquina nueva o una máquina ya registrada cuyo ID fue borrado.

Solución: no tiene un solución directa, en el futuro es posible hacer

conjeturas en base al estado de inactividad de tal máquina dado su ID y permitir el “sincronismo” manual de dos o más IDs registrados en el servidor por si poseen características similares (como ser IP, MAC, sistema operativo, etc.). Por ahora lo ideal sería impedir el borrado del archivo de configuración teniendo un buen sistema de manejo de persmisos y usuarioes en las máquinas cliente.

- Suplantanción de identidad:

Situación: un usuario roba el archivo de configuración de un cliente y lo usa para hacerse pasar por él (algo así como el robo de una cookie), por lo tanto el servidor creerá que el cliente del usuario malicioso es el cliente verdadero.

Solución: el archivo de configuración debe estar almacenado encriptado con datos de la máquina cliente verdadera (como ser el número de serie de la BIOS), por lo que ese archivo sólo podrá ser leído por el cliente verdadero cuya clave posea (sólo un cliente sabe su BIOS). Esto podría llevar a que el usuario malicioso intente también robar la clave (datos de la BIOS por ejemplo) del cliente verdader para desencripatr el archivo y volver a encriptar con sus propios datos. Esta situación aún no está contemplada.

- Alteración del archivo de configuración del cliente:

Situación: el archivo de configuración del cliente es distinto al almacenado en el servidor, por lo tanto el cliente no funcionaría de la manera que el servidor espera.

Solución: siempre el servidor tiene la configuración correcta por cliente. Por lo tanto, ante cualquier diferencia el servidor debe darle al cliente la configuración que él posee guardada. La diferencia entre los archivos cliente y servidor se hace mediante un hash, el cliente calcula un hash (usando el algoritmo MD5) de su archivo de configuración y la envía al servidor, el servidor posee el detalle de la configuración de ese cliente junto con el hash de ese detalle, por lo que verifica si son coincides y en caso de discrepancia le avisa al cliente cuál es la configuración correcta.

Consideraciones

- Los comandos se mandan con notación JSON.
- El archivo de configuración debe ser estricto en cuanto su convensión (debe coincidir en el cliente y servidor), su notación es JSON.
- Por ahora se manda todo en texto plano (no usamos SSL).
- El tipo de socket que usamos es TCP, por lo tanto no se usarán paquetes “acknowledge” (o aviso de recibo) aprovechando la fiabilidad de TCP.