Recepción con *Handlers* en Lower_Layer_UDP

Programación de Sistemas de Telecomunicación
Informática II

Departamento de Sistemas Telemáticos y Computación (GSyC)

Noviembre de 2019

Contenidos

Contenidos

Llamadas bloqueantes

¿Qué es una llamada bloqueante?

Las llamadas a algunos subprogramas bloquean el flujo de ejecución del programa hasta que ocurran eventos externos al programa

- Ejemplo: la lectura de texto del teclado mediante `Ada.Text_IO.Get_Immediate` o `Ada.Text_IO.Get_Line` bloquea el flujo de ejecución del programa hasta que el usuario introduzca algún texto por el teclado:
  - Durante este tiempo el programa NO puede realizar otras acciones como recibir mensajes.

Llamadas bloqueantes

La llamada a LLU.Receive es bloqueante

Por esta razón en Mini-Chat un cliente no puede ser lector y escritor a la vez: no puede recibir los mensajes que le envía el servidor y a la vez estar leyendo cadenas de texto del teclado.

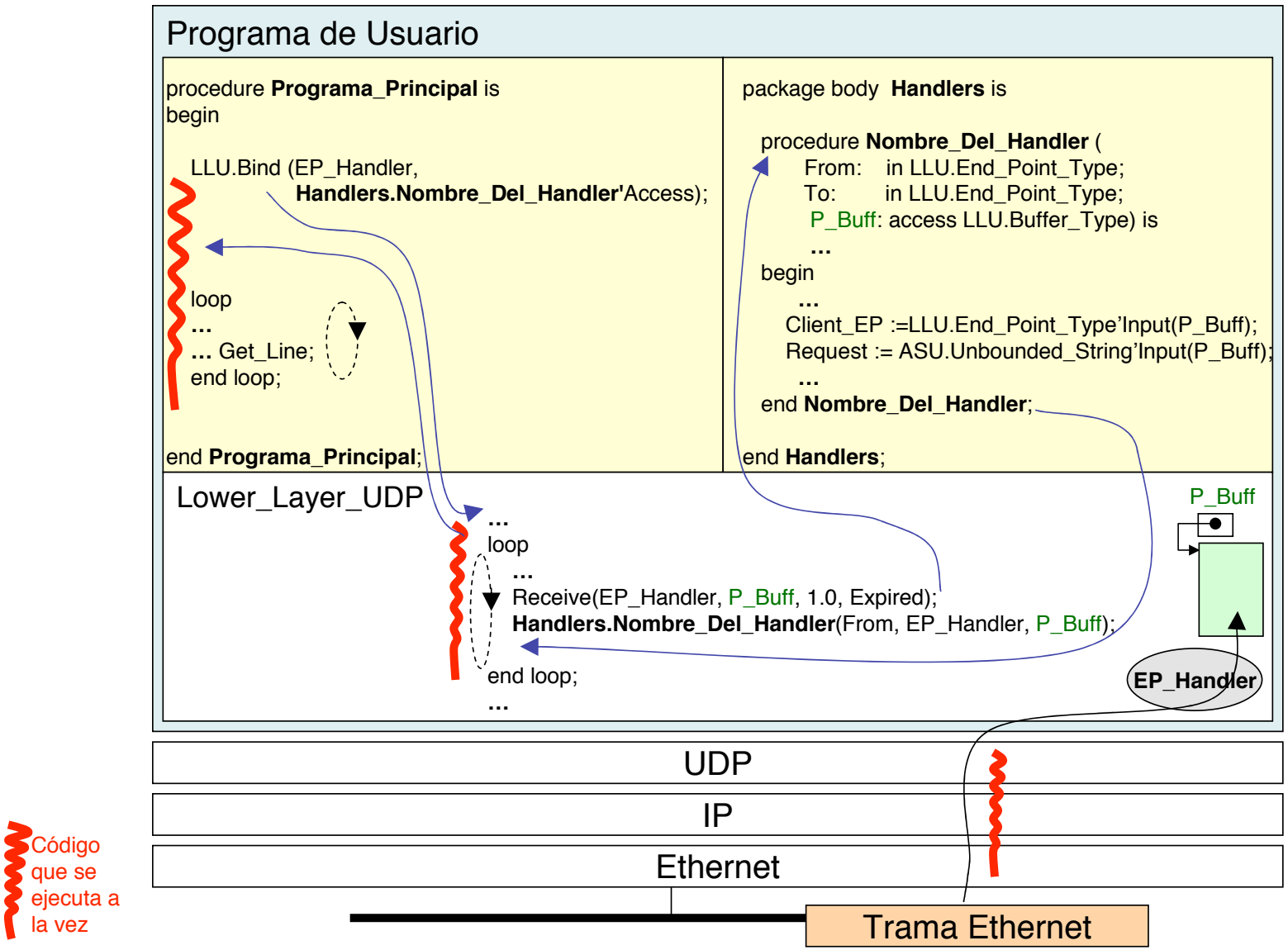
- La recepción de mensajes en Lower_Layer_UDP (LLU) mediante la llamada `LLU.Receive` bloquea el flujo de ejecución del programa hasta que se reciba algún mensaje, o venza el timeout especificado en la llamada
  - Durante este tiempo el programa NO puede realizar otras acciones como leer del teclado o enviar mensajes.
  - Cuando se usa `LLU.Receive` el programador SÍ determina el momento en el que quiere recibir un mensaje
- OJO, no siempre es malo recibir mediante `LLU.Receive`: en ocasiones un programa no puede hacer nada si no recibe un mensaje.

Contenidos

Recepción mediante *handlers* en LLU

- En Lower_Layer_UDP la recepción de mensajes mediante *handlers* o manejadores permite que el flujo de ejecución del programa no se detenga para esperar a recibir mensajes
- El programa delega en el sistema la tarea de esperar a recibir mensajes y la de llamar al *handler* justo cuando llegue uno. El *handler* contiene el código que el programador quiere que se ejecute cuando se reciba un mensaje.
- Ejecución concurrente: Mientras que el sistema está ejecutando el *handler* porque ha llegado un mensaje, el programa principal puede estar ejecutando simultáneamente cualquier otro código:
  - leer del teclado,
  - enviar mensajes,
  - recibir mensajes de manera bloqueante con `LLU.Receive` en otro End_Point,
  - e incluso recibir mensajes de manera no bloqueante con *handlers* en otro End_Point,...

Recepción mediante *handlers* en LLU



Contenidos

Programación con *handlers*

- El programador escribe el código que debe ejecutarse cuando se reciba un mensaje en un subprograma al que llamaremos *handler*
  - El código del *handler* necesariamente debe estar dentro de un paquete, no puede estar en el fichero del programa principal.
  - La especificación del *handler* está predeterminada pues hay código en el sistema que llamará al *handler* y ha de coincidir con la especificación a la que llamará este código
- La función del código que se escribe en el *handler* es procesar los contenidos de cada mensaje que vaya llegando
- El programador asocia el handler a un `End_Point` en el momento de llamar a `Bind` o a `Bind_Any`
- El programador NO escribe código para llamar al subprograma *handler*, sino que es el sistema el que llama al *handler* cada vez que llega un mensaje.

Especificación del *handler*

- La especificación del handler tiene que ser obligatoriamente la resaltada en rojo (excepto por el nombre, que elegimos nosotros):

```
with Lower_Layer_UDP;
package Handlers is
   package LLU renames Lower_Layer_UDP;

   procedure Nombre_Del_Handler (From: in LLU.End_Point_Type;
                                 To:   in LLU.End_Point_Type;
                                 P_Buffer: access Buffer_Type);
end Handlers;
```

- Parámetros:
  - `From:` `End_Point` interno de `Lower_Layer_UDP` que no utilizaremos.
  - `To:` `End_Point` al que venía destinado el mensaje recibido
  - `P_Buffer:` puntero al Buffer que contiene el mensaje recibido

Cuerpo del *handler*

- En el cuerpo del handler se escribe el código para extraer los campos del buffer que el sistema le pasa al *handler* en el parámetro `P_Buffer` con los contenidos del mensaje recibido
- El handler se invoca **automáticamente** cuando llega algo al `End_Point` asociado.
- **NO hay que llamar a LLU.Receive** para recibir los datos.
- Es importante que el *handler* termine lo antes posible, pues mientras se está ejecutando no se están procesando nuevos mensajes que pueden estar llegando a la máquina
  - En particular, es importante no hacer llamadas bloqueantes en el código del handler, como `Get_Line` o `LLU.Receive`

Cuerpo del *handler*

- Ejemplo:

```
with Ada.Text_IO;
with Ada.Strings.Unbounded;
package body Handlers is
   package ASU renames Ada.Strings.Unbounded;

   procedure Nombre_Del_Handler (From: in LLU.End_Point_Type;
                                 To:   in LLU.End_Point_Type;
                                 P_Buffer: access Buffer_Type) is
      Client_EP: LLU.End_Point_Type;
      Request  : ASU.Unbounded_String;
      Reply  : ASU.Unbounded_String := "¡Bienvenido!";
   begin
      -- Suponemos que los mensajes que se reciben
      -- contienen un End_Point y un Unbounded_String
      Client_EP := LLU.End_Point_Type'Input (P_Buffer);
      Request := ASU.Unbounded_String'Input (P_Buffer);

      Ada.Text_IO.Put_Line (ASU.To_String (Request));

      LLU.Reset (P_Buffer.all);
      ASU.Unbounded_String'Output (P_Buffer, Reply);
      LLU.Send (Client_EP, P_Buffer);

   end Nombre_Del_Handler;
end Handlers;
```

- NOTA: `P_Buffer` es un **puntero** a un `Buffer_Type`, y no un `Buffer_Type`

Asociación del *handler* a un End_Point con Bind/Bind_Any

- El programador asocia en su código el *handler* a un `End_Point` en el momento de llamar a `Bind` o `Bind_Any`.
- Desde ese momento si se reciben mensajes el sistema llamará al *handler*
- En los `End_Points` a los que el programa se ha enlazado con un *handler* no se puede recibir con `LLU.Receive`
- A `Bind`/`Bind_Any` se le pasan como argumentos un End_Point y un puntero al *handler*
- Hasta que no se llame a `LLU.Finalize` el sistema seguirá llamando a los *handlers* para procesar los mensajes que se sigan recibiendo, incluso cuando no queden más sentencias para ejecutar en el programa principal.

Asociación del *handler* a un End_Point con Bind/Bind_Any

- Ejemplo con `Bind`:

```
...
Server_EP := LLU.Build ("127.0.0.1", 6123);

-- Tras llamar a Bind ya se pueden estar recibiendo
-- mensajes automáticamente en el manejador
LLU.Bind (Server_EP, Handlers.Server_Handler'Access);
```

- Ejemplo con `Bind_Any`:

```
-- Tras llamar a Bind_Any ya se pueden estar recibiendo
-- mensajes automáticamente en el manejador
LLU.Bind_Any (Client_EP, Handlers.Client_Handler'Access);
```