Trabajo de Fin de Carrera
Localizador Gráfico de Direcciones IP

Alumno
Javier Lerones Gallego (ETIG – 01/2006)

Consultor
Jesus Arribi Vilela

Indice

1. Introducción

1.1 Punto de Partida

Este documento presenta el desarrollo de un programa localizador gráfico de direcciones IP, como trabajo de fin de carrera.

En el mundo actual, la mayoría de los ordenadores están conectados a la red global Internet; para poder transmitir información entre ellos se necesita identificarlos con una dirección única, conocida como dirección IP (Internet Protocol).

Cada ordenador conectado a Internet poseé una dirección diferente, esto permite que el flujo de información entre ordenadores pueda ser en ámbos sentidos. Además los ordenadores o routers intermedios de las rutas de los datos también poseen sus respectivas direcciones IP.

Gracias a los protocolos de Internet, existen métodos desarrollados para poder saber que ruta siguen los datos entre el origen y el destino de un flujo de datos. En este punto es donde se encuadra el programa que se va a desarrollar en este proyecto.

1.2 Objetivos

Centrandonos en el objetivo del proyecto, el programa está pensado para localizar gráficamente en un mapa del globo terráqueo, la posición de un sistema indicado por el usuario que está utilizando el programa; Así mismo, también tiene que presentar la ruta que siguen los datos entre el sistema que está ejecutando el programa del usuario y el destino que ha indicado anteriormente.

Este resultado se debe conseguir cumpliendo con dos premisas fundamentales:

- Que se consiga el máximo de información disponible para cada IP.

- Que esa información sea suficiente para poder mostrarla gráficamente por pantalla al usuario final.

1.3 Enfocamiento Inicial

El entorno de ejecución escogido para el programa será el sistema operativo Windows, en cualquier versión que soporte la ejecución del .NET Framework en su reciente versión 2.0, preferentemente Windows XP Home o Profesional.

El .NET Framework se puede descargar de forma totalmente gratuita en cualquiera de las siguientes direcciones, dependiendo del tipo de procesador que tenga el sistema en el que se va a instalar:

.NET Framework 2.0 (Para procesadores x86)
.NET Framework 2.0 (Para procesadores x64 - 64 bits)

El lenguaje de programación elegido será Visual C#, y para el desarrollo del programa se utilizará el entorno de programación Visual C# Express Edition, gratuito y descargable en la siguiente dirección:

Visual C# Express Edition

Este entorno será necesario para poder recompilar el proyecto a partir de su código fuente.

1.4 Planificación del proyecto

El programa se compondrá de dos partes bien diferenciadas, el núcleo y la interfaz.

El núcleo se encarga de extraer toda la información de internet, las direcciones IP que componen la ruta y las coordenadas geográficas de cada una.

La interfaz se encargará de mostrar esa información gráficamente por pantalla.

Para el desarrollo de las funciones del núcleo se utilizarán, por una parte, los métodos de System.Net.NetworkInformation para conseguir las direcciones IP de la ruta y el destino mediante el uso de la clase Ping, y por otra parte, el componente gratuito resolvedor de DNS de Pivo.
Este componente, gratuito y de libre uso en su versión de educación, permite realizar consultas sobre hosts a los servidores DNS con cualquier tipo de registro DNS, incluido el registro LOC, que contiene las coordenadas geográficas, según se explica en el protocolo RFC 1876 (ver Apéndice).

No obstante la gran mayoría de los servidores y ordenadores conectados a internet no contienen información en el registro LOC, por lo que para mejorar el funcionamiento del programa, se ha optado por utilizar un método complementario para conseguir los datos geográficos: El webservice de la compañía Maxmind (maxmind.com).
Este webservice permite conocer los datos asociados a una determinada dirección IP, incluido país, ciudad, longitud y latitud, más que suficiente para las necesidades de nuestro programa.

1.5 Componentes de terceros utilizados

Como ya se ha indicado en el punto anterior se ha utlizado el resolvedor de DNS de Pivo.com, de libre distribución si su uso es con fines educativos y no comerciales.
Así mismo para una parte de la interfaz se ha usado la librería MagicLibrary 1.7.4, totalmente

gratuita en esa versión, aunque su uso es de escasa importancia en el proyecto.

2. Planificación y construcción

2.1 Funcionalidad básica

2.1.1 Requisitos

El funcionamiento básico del programa serguirá los sigientes pasos:

· El usuario indica una dirección IP o un nombre de Host, y pulsa el botón de “Localizar”.
· El núcleo crea un nuevo “Tab” donde mostrar la información de ese destino.
· En ese tab se inicial el proceso de búsqueda.

- Mediante el uso de la clase Ping se van consiguiendo todas las direcciones IP de la ruta.
- Para cada dirección IP de la ruta se buscan sus datos geográficos usando RFC 1876.
- Si esos datos no se encuentran, se utiliza el webservice de Maxmind.
- La interfaz crea un nuevo nodo en la lista, y lo dibuja en el mapa transformando las coordenadas conseguidas.
- El núcleo continua con la siguiente dirección IP.

Una vez terminado el proceso, el usuario podrá buscar la ruta de otra dirección o nombre de Host volviendo a pulsar en el botón de “Localizar”, se creará un nuevo “Tab” que contendrá una nueva imágen del mundo con los datos de la nueva ruta.

2.1.2 Análisis y Diseño

El programa debe comprobar que el texto que ha introducido el usuario es una dirección IP o un nombre de Host válidos, para ello se utlizarán expresiones regulares. En este documento no se entrará en detalle sobre el funcionamiento de las mismas, aunque en el apartado de Bibliografía se pueden encontrar varios enlaces relacionados con las mismas.

Así mismo, no todas las direcciones IP son útiles, el protocolo IP define cuatro clases de direcciones IP y ciertos rangos especiales como se ve a continuación:

Rangos de Clases:

Class	Leading bits	Start	End	CIDR equivalent
Class A	0	0.0.0.0	127.255.255.255	/8
Class B	10	128.0.0.0	191.255.255.255	/16
Class C	110	192.0.0.0	223.255.255.255	/24
Class D (multicast)	1110	224.0.0.0	239.255.255.255	NA
Class E (reserved)	1111	240.0.0.0	255.255.255.255	NA

Rangos Especiales:

Addresses	CIDR Equivalent	Purpose	RFC	Class	Total # of addresses
0.0.0.0 - 0.255.255.255	0.0.0.0/8	Zero Addresses	RFC 1700	A	16,777,216
10.0.0.0 - 10.255.255.255	10.0.0.0/8	Private IP addresses	RFC 1918	A	16,777,216
127.0.0.0 - 127.255.255.255	127.0.0.0/8	Localhost Loopback Address	RFC 1700	A	16,777,216
169.254.0.0 - 169.254.255.255	169.254.0.0/16	Zeroconf	RFC 3330	B	65,536
172.16.0.0 - 172.31.255.255	172.16.0.0/12	Private IP addresses	RFC 1918	B	1,048,576
192.0.2.0 - 192.0.2.255	192.0.2.0/24	Documentation and Examples	RFC 3330	C	256
192.88.99.0 - 192.88.99.255	192.88.99.0/24	IPv6 to IPv4 relay Anycast	RFC 3068	C	256
192.168.0.0 - 192.168.255.255	192.168.0.0/16	Private IP addresses	RFC 1918	B	65,536
198.18.0.0 - 198.19.255.255	198.18.0.0/15	Network Device Benchmark	RFC 2544	C	131,072
224.0.0.0 - 239.255.255.255	224.0.0.0/4	Multicast	RFC 3171	D	268,435,456
240.0.0.0 - 255.255.255.255	240.0.0.0/4	Reserved	RFC 1700	E	268,435,456

El programa tiene que comprobar que la direccion IP introducida por el usuairo no corresponde a una de las anteriores direcciones especiales, ya que no se podrá encontrar la localización de la misma.

2.1.3 Implementación

Esta comprobación usando expresiones regulares se realiza en la clase *Fmain.cs* en los métodos (ver código fuente):

```
private bool EsIPValida(string Busqueda)
```

y

```
private bool EsHostValido(string Busqueda)
```

El segundo método es trivial, en cambio el método *EsIPValida*() comprueba que el parametro es una dirección IP, acto seguido va comprobando uno por uno los rangos especiales de la tabla anterior. Si la dirección pertenece a alguno de ellos, el método devuelve false.

2.1.4 Test

A continuación se presentan una serie de pruebas realizadas para comprobar el correcto funcionamiento de las expresiones regulares.

Entrada	Salida Esperada	Salida Obtenida	Resultado
www.yahoo.com	correcto	correcto	OK
yahoo.com	correcto	correcto	OK
yahoocom	Error	Error	OK
http://www.yahoo.com	Error	Error	OK
0.0.0.0	Error	Error	OK
0.0.0.0.0.0	Error	Error	OK
0.000.0.0	Error	Error	OK
169.254.100.200	Error	Error	OK
245.0.0.0	Error	Error	OK
201.26.121.69	correcto	correcto	OK
180.180.180.180	correcto	correcto	OK

El funcionamiento de las comprobaciones en el método *EsIPValida*() es muy simple, por ese motivo no haré el análisis de los valores límite, en su lugar, incluyo las comprobaciones de error de ciertos valores de los rangos de ips especiales que servirán para comprobar que los “checks” de todos las condiciones de los “if” devuelven el resultado correcto.

2.2 Núcleo

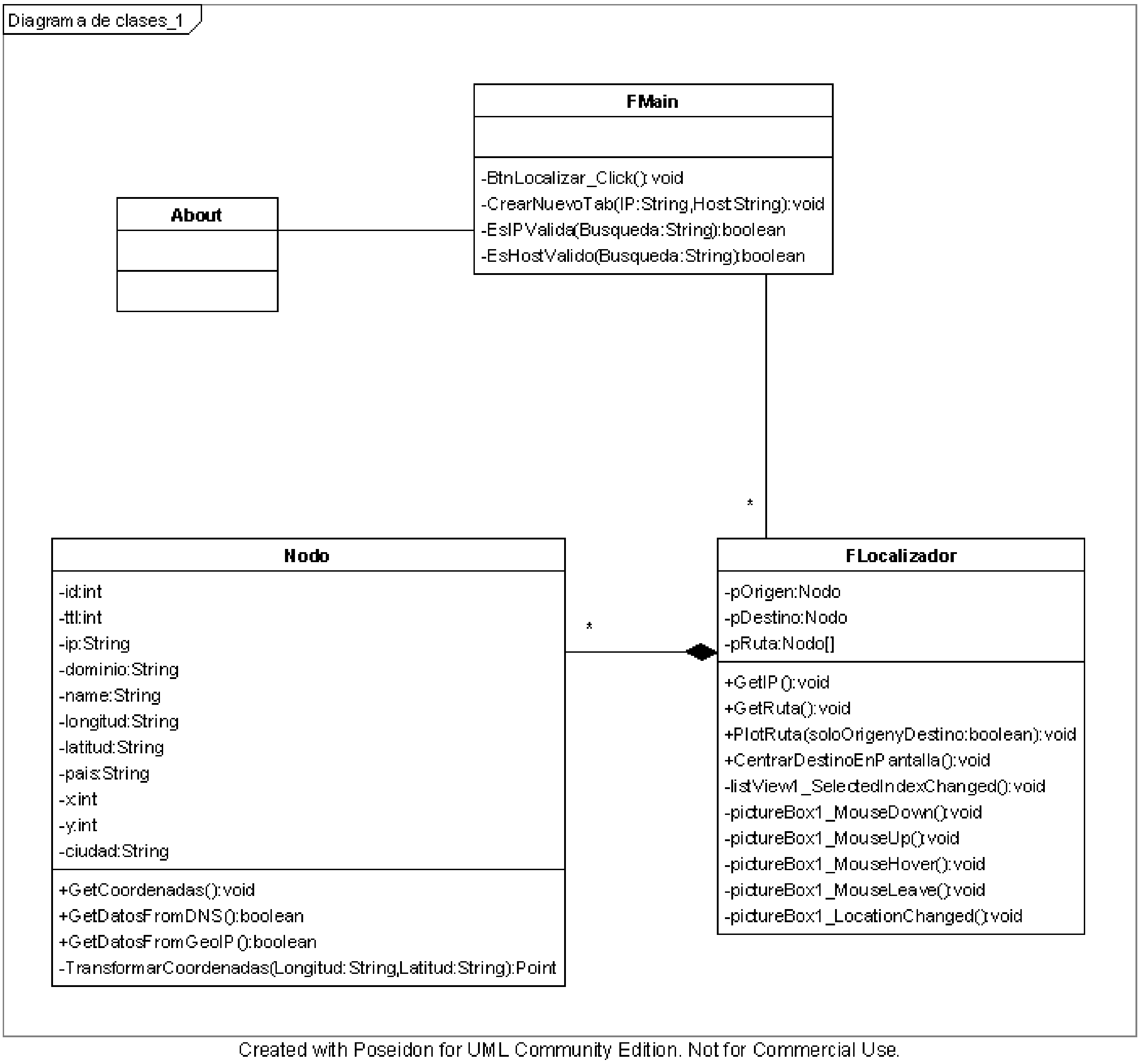
2.2.1 Requisitos

El núcleo del programa se encarga de realizar todo el procesado de la ruta y de la información geográfica de cada dirección de la misma.

Esencialmente el núcleo es el programa completo, a excepción de los métodos de presentación por pantalla que se incluyen en la clase *Flocalizador.cs*

2.2.2 Análisis y Diseño

A continuación se incluye el diagrama UML con el diseño del programa:



2.2.3 <u>Implementación</u>

Clase *Fmain.cs*

Esta clase espera a que el usuario pulse el botón de “localizar”, en ese momento, comprueba que el texto que el usuario ha introducido es una dirección IP o un nombre de Host válidos llamando a los métodos *EsIPValida*() y *EsHostValido*().

Si es correcto, entonces crea un nuevo Tab que contendrá el control del mapa mundo (*Flocalizar.cs*) con el texto que ha introducido el usuario como valor.

Tambien gestiona las opciones del menu principal de la aplicación,

Archivo -> Salir
y
Ayuda -> Acerca de

Clase *FLocalizador.cs*

Esta clase se encarga de obtener las direcciones IP que componen la ruta, para cada dirección IP crea un nuevo objeto *Nodo*, y para cada uno de esos objetos busca sus datos geograficos llamando a los metodos de la clase *nodo* correspondientes.

La ruta se almacena en una Lista que se compone de objetos Nodo.

El método GetRuta() funciona de la siguiente manera:

Inicializa el valor del TTL a 1 y entra en un bucle while con condición de salida falsa.
Para cada valor de TTL hace un *Ping* a la IP del destino y espera el resultado. Si el resultado es “TTLExpired”, entonces crea un nuevo nodo con la dirección IP que ha devuelto, busca sus datos geograficos llamando a los métodos *GetDatosFromDNS*() y *GetDatosFromGeoIP*(), y lo añade a la lista. Si el resultado es “Success” entonces puede que el tiempo haya expirado (TimedOut) o que el nodo devuelto sea el de destino, si es esta segunda opción, almacena sus datos en el objeto pDestino, lo añade a la ruta y finaliza el bucle.

Clase *Nodo.cs*

Esta clase tiene dos métodos principales, GetDatosFromDNS() y GetDatosFromGeoIP().

El primer método ejecuta el algoritmo de búsqueda especificado en el protocolo RTF 1876 para conseguir los registros LOC del Host del nodo. Este método devuelve true si ha encontrado registro LOC valido para el host actual, false si no lo ha encontrado. Así mismo si encuentra el registro, almacen los datos de longitud y latitud en las variables privadas de la clase.

El segundo método hace una llamada al Webservice de MaxMind para conseguir los datos geográficos de la ip actual. Si el webservice devuelve los datos correctamente, los almacena en las variables privadas de la clase y devuelve true, si se produce un error, o no encuentra datos para esa dirección IP, devuelve false.

2.2.4 <u>Test</u>

Para hacer un test del núcleo se necesita ejecutar el programa completo, núcleo + interfaz.
Las pruebas de ejecución se incluyen en el apartado 3 de este documento.

2.3 Interfaz

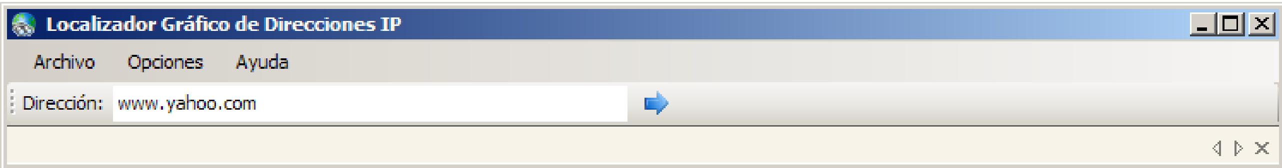
2.3.1 Requisitos

El desarrollo de la interfaz se ha realizado en su totalidad de forma visual utilizando las herramientas del entorno de desarrollo del Visual C# Express.

La interfaz debe ser agradable a la vista y muy intuitiva, las funcionalidades que debe tener este programa no son muy complejas, por lo que se debe simplificar al máximo el entorno visual que el usuario final va a utilizar.

2.3.2 Análisis y Diseño

El programa principal estará compuesto por una ventana que contenga un menú principal, y una barra justo debajo del menú en la que habrá un cuadro de texto donde el usuario escribirá la dirección IP o nombre de Host que desea buscar:

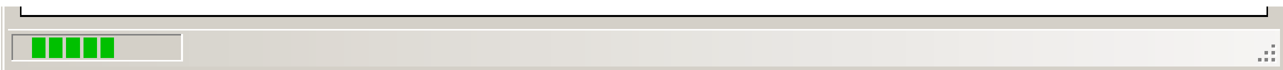


(figura 1)

Debajo de esta barra con el cuadro de texto, se presentará el mapa del mundo con el dibujo de la ruta y el listview con la información sobre los nodos de la ruta. Además se podrá hacer más de una búsqueda, con lo que cada control Flocalizador.cs se creará en un “Tab” independiente:

(figura 2)

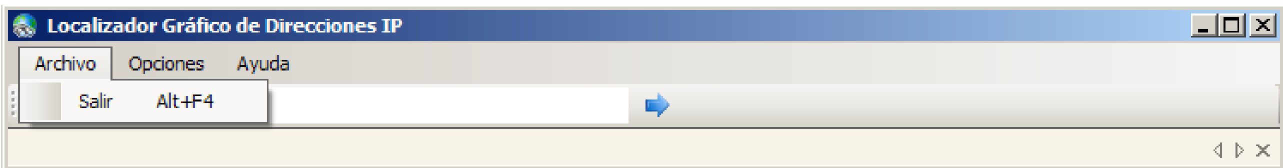
Además se presentará una barra de estado en la que se mostrará una pequeña barra de progreso que servirá para indicar que se está buscando la ruta actual en internet.



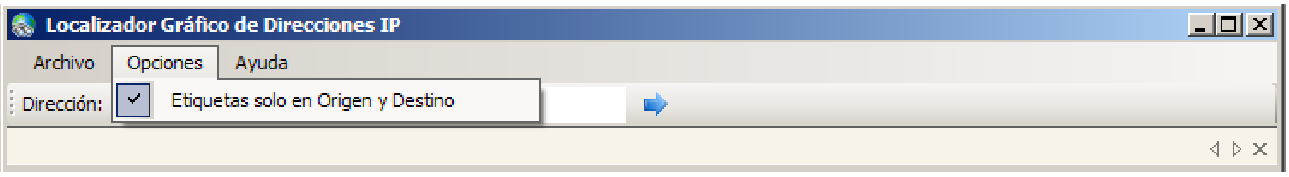
(figura 3)

El menú principal de la aplicación se compondrá de las siguientes opciones:

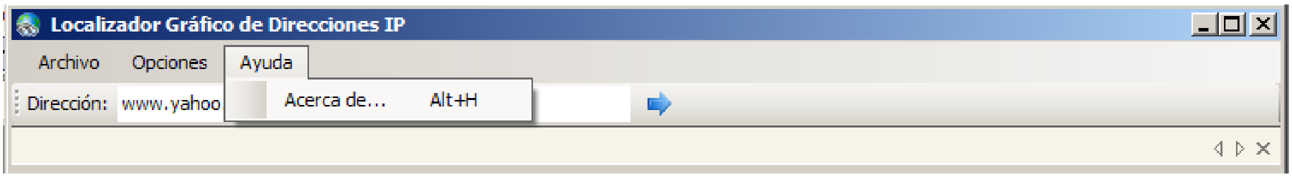
Menu Archivo:



Menu Opciones:



Menu Ayuda:



Y ya para terminar el cuadro de diálogo de Acerca de tendrá la siguiente estructura:

2.3.3 <u>Implementación</u>

Además de las funcionalidades descritas en el punto anterior, la interfaz se encarga de transformar las coordenadas de latitud y longitud a puntos del mapa.

Este proceso es muy sencillo:

Se conoce la longitud y latitud de cada extremo del mapa, 359º 59' 59'' en horizontal y 179º 59' 59'' en vertical, sabemos además que el centro del mapa es el punto 0º 0' 0''N y 0º 0' 0''E, y además sabemos que la imágen tiene un tamaño de 2400 pixels horizontales por 1600 verticales.

Sabiendo estos datos, calculamos cuantos pixels corresponden a cada segundo con la mayor precisión posible:
0.00185185185185185185185185185185519

Y teniendo esta información, transformamos las coordenadas de longitud y latitud en segundos, y multiplicamos estos segundos por la constante anterior.
Esto nos dará un desplazamiento desde el centro del mapa en dirección vertical u horizontal positiva o negativa con respecto al centro de la imágen.

Una vez hecho esto solamente hay que restar este desplazamiento al punto central de la imágen en cada coordenada X e Y y ya tendremos el punto exacto que representa esas coordenadas en el mapa.

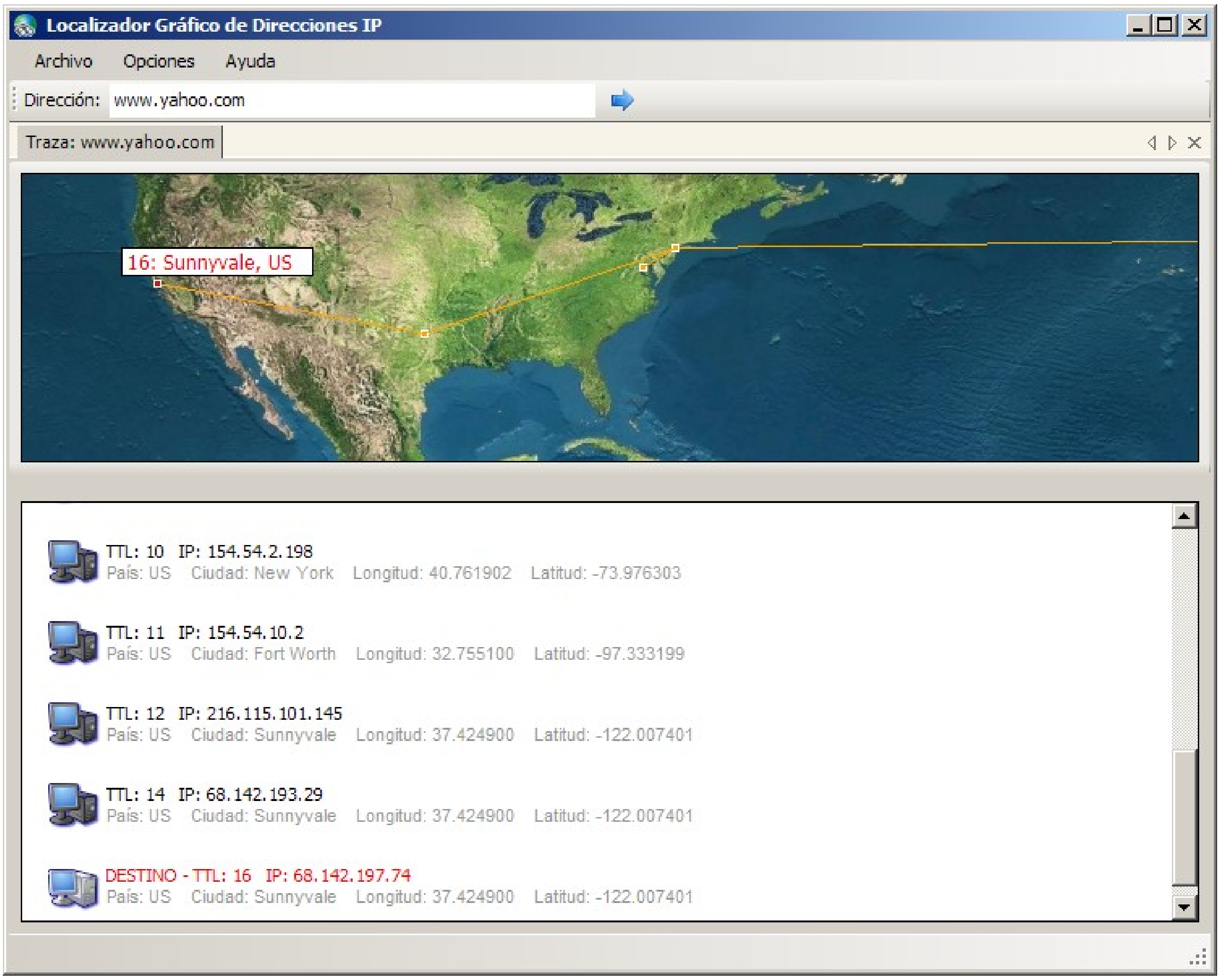
2.3.4 Test

Para hacer un test del núcleo se necesita ejecutar el programa completo, núcleo + interfaz.
Las pruebas de ejecución se incluyen en el apartado 3 de este documento.

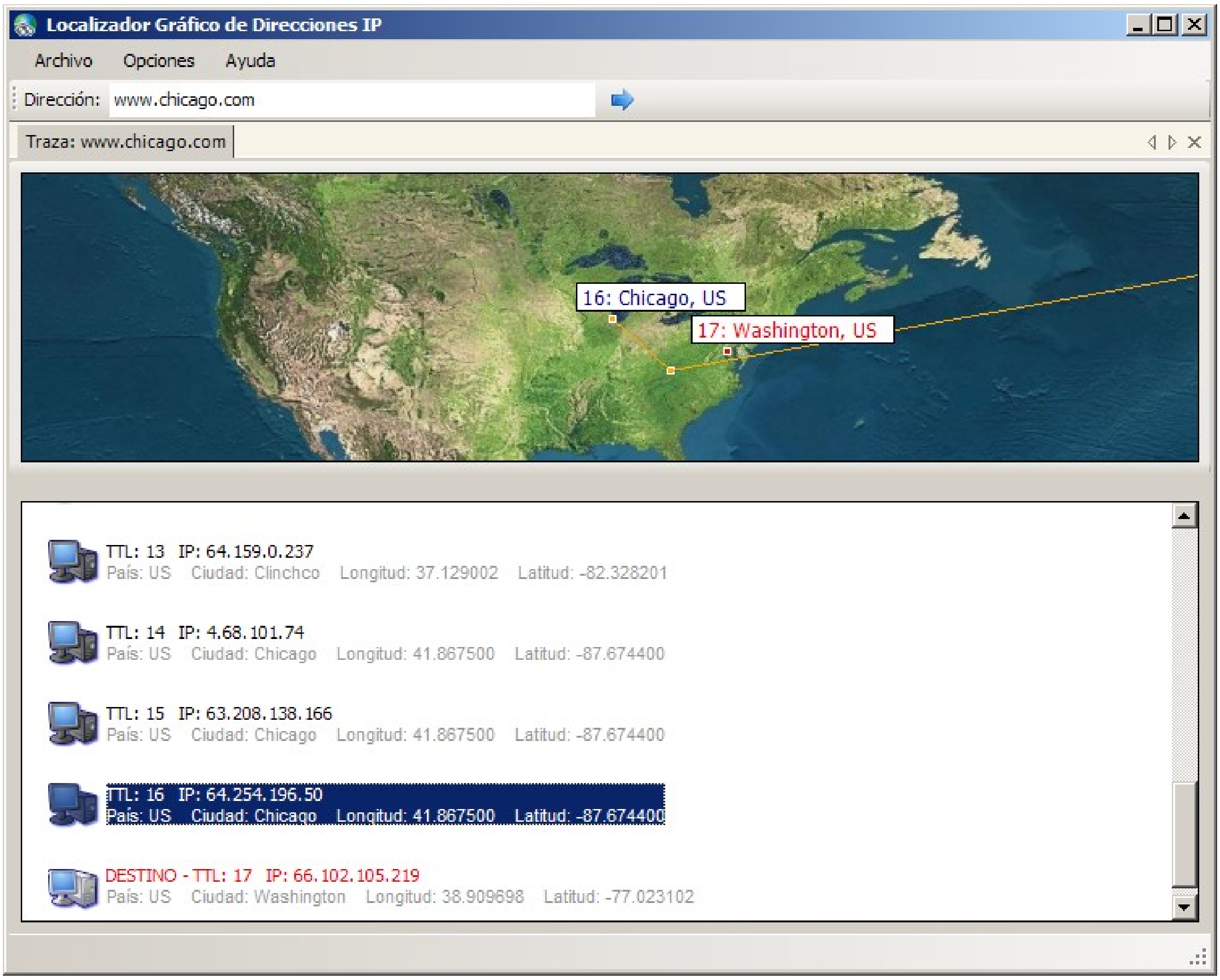
3. Batería de pruebas

A continuación se muestra el resultado de la ejecución de la aplicación para varios Hosts y Direcciones IP:

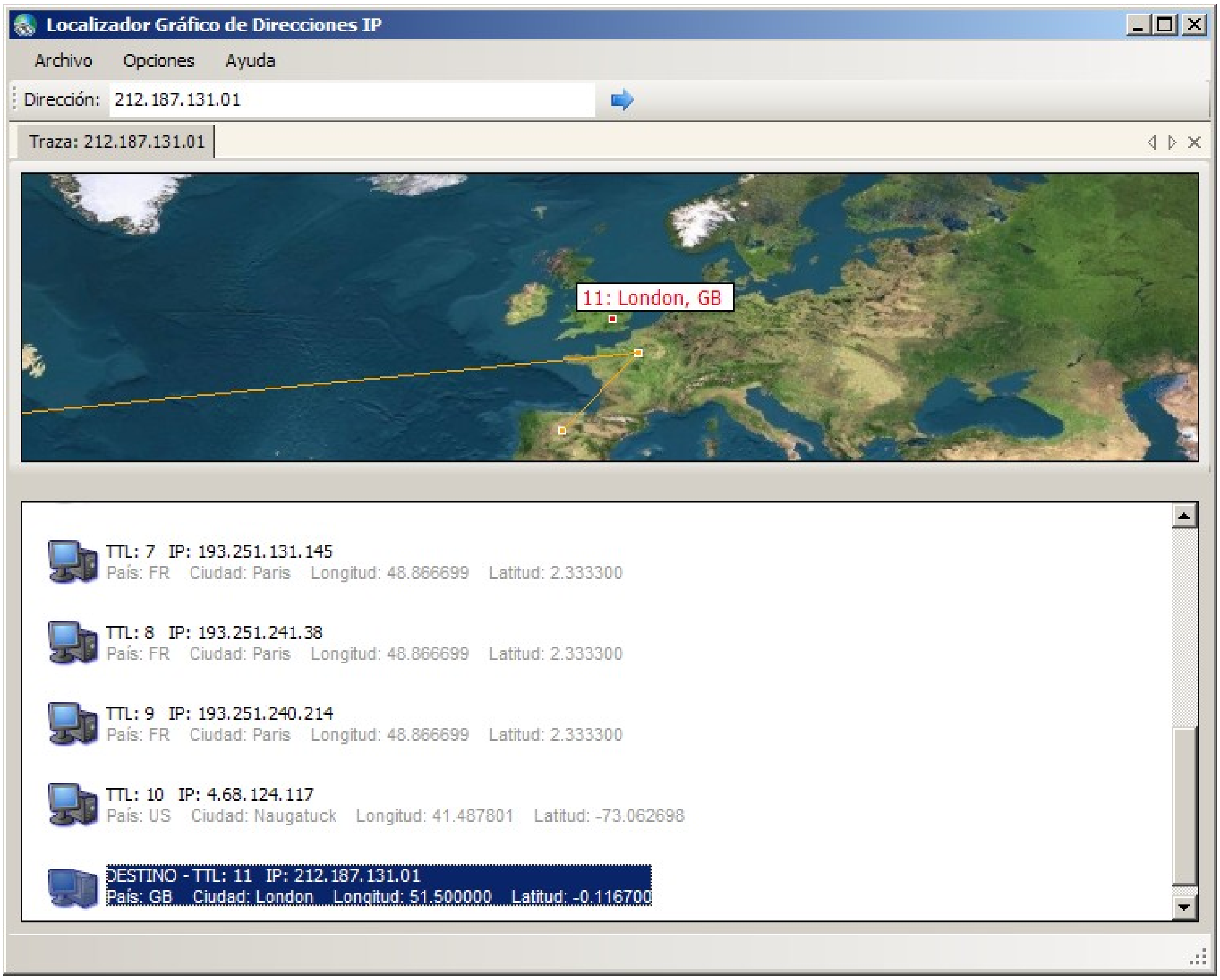
www.yahoo.com



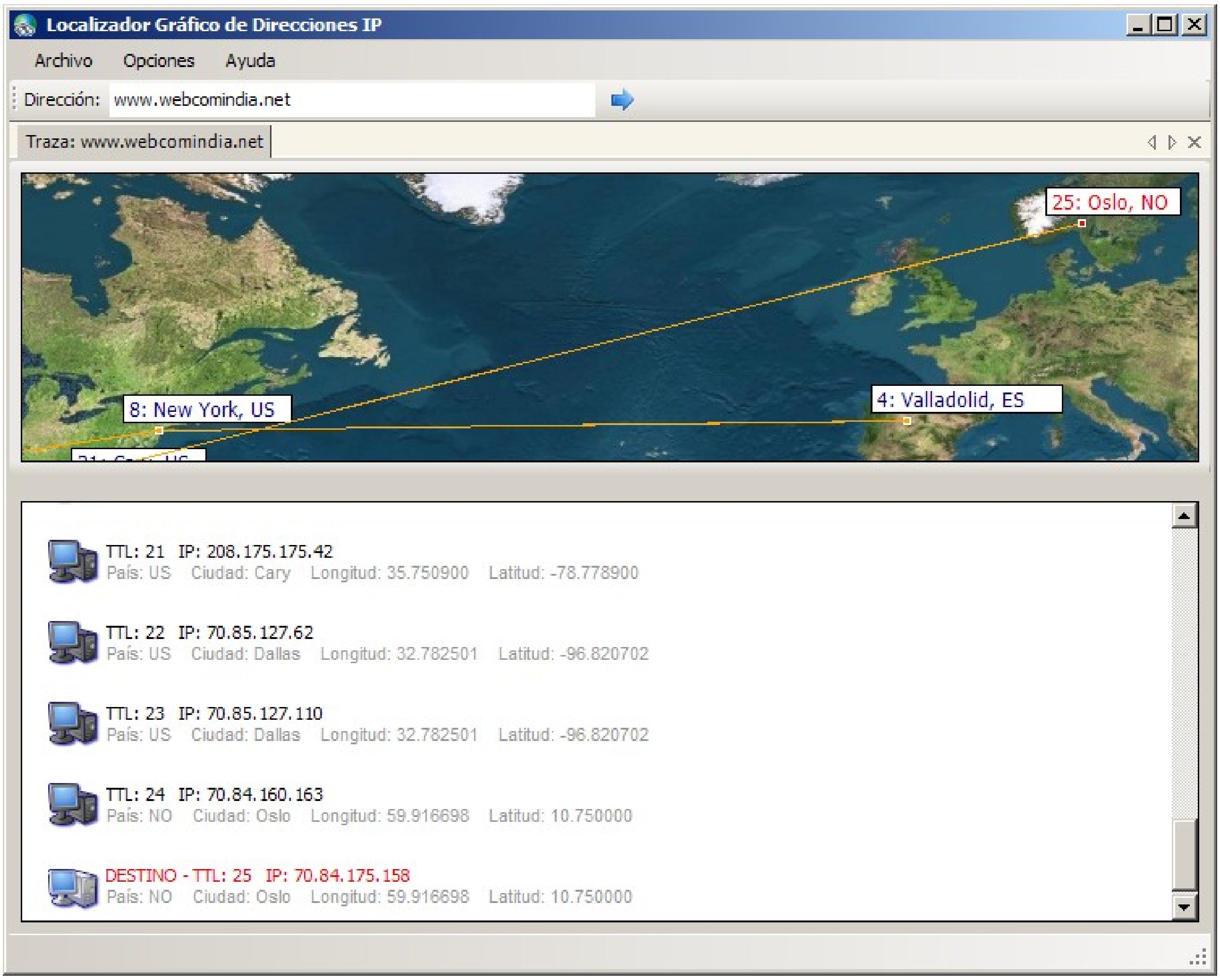
www.chicago.com


Localizador Gráfico de Direcciones IP
Archivo
Opciones
Ayuda
Dirección: www.chicago.com
Traza: www.chicago.com
16: Chicago, US
17: Washington, US
TTL: 13 IP: 64.159.0.237
País: US Ciudad: Clinchco Longitud: 37.129002 Latitud: -82.328201
TTL: 14 IP: 4.68.101.74
País: US Ciudad: Chicago Longitud: 41.867500 Latitud: -87.674400
TTL: 15 IP: 63.208.138.166
País: US Ciudad: Chicago Longitud: 41.867500 Latitud: -87.674400
TTL: 16 IP: 64.254.196.50
País: US Ciudad: Chicago Longitud: 41.867500 Latitud: -87.674400
DESTINO - TTL: 17 IP: 66.102.105.219
País: US Ciudad: Washington Longitud: 38.909698 Latitud: -77.023102

212.187.131.1



www.webcomindia.net



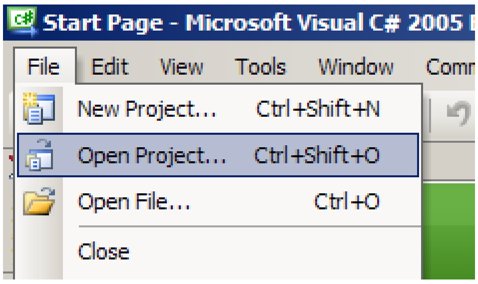
3.1 <u>Instrucciones de compilación</u>

Para poder recompilar el código fuente suministrado con esta memoria, primero se deben cumplir una serie de prerequisitos, estos son:

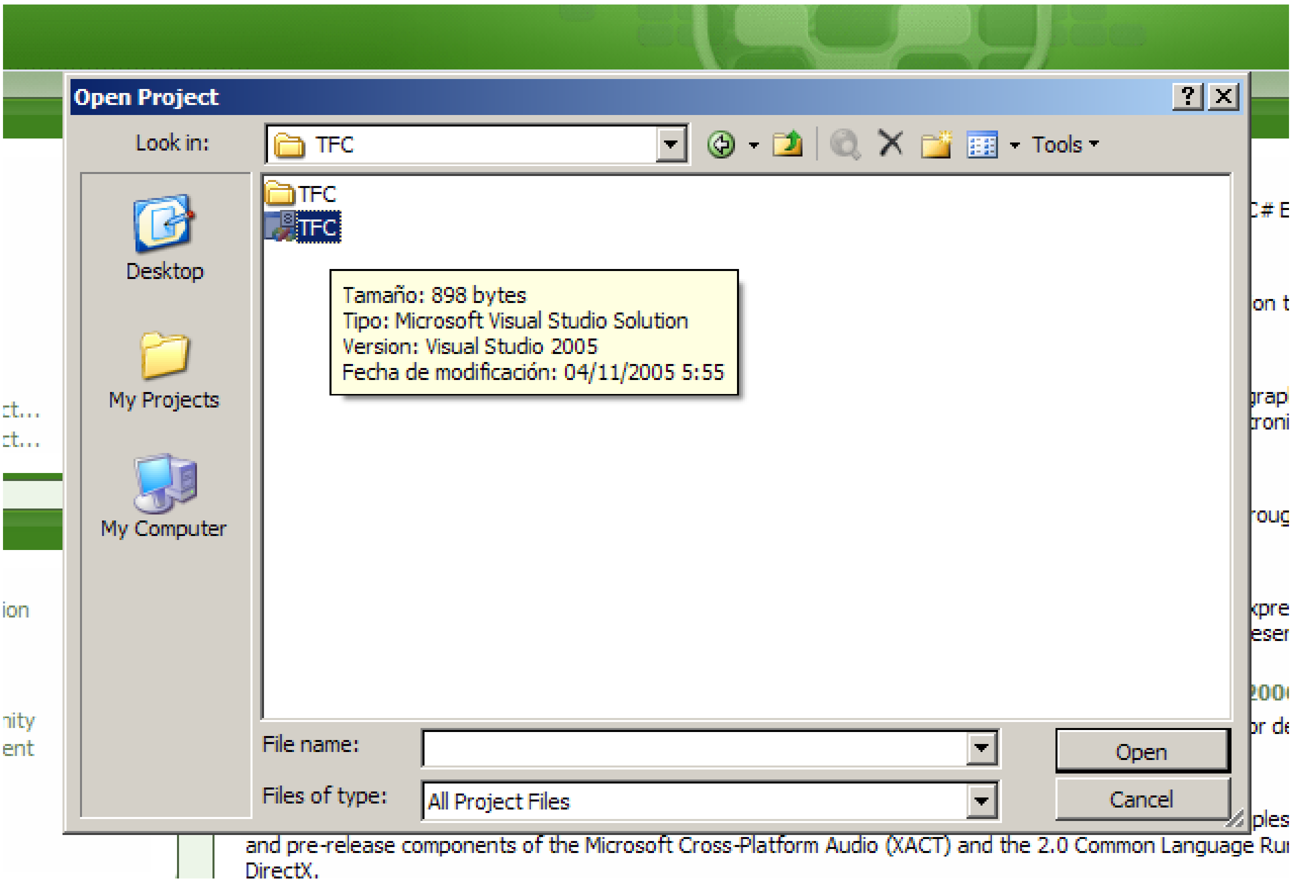
Tener instalado el .NET Framework 2.0
Tener instalado el Visual C# Express Edition

Una vez instalados, hay que descomprimir el archivo jlerones_CodigoFuente.zip en el directorio de projectos del Visual Studio, suele ser "C:\Documents and Settings\USUARIO\Mis Documentos\Visual Studio 2005\Projects".

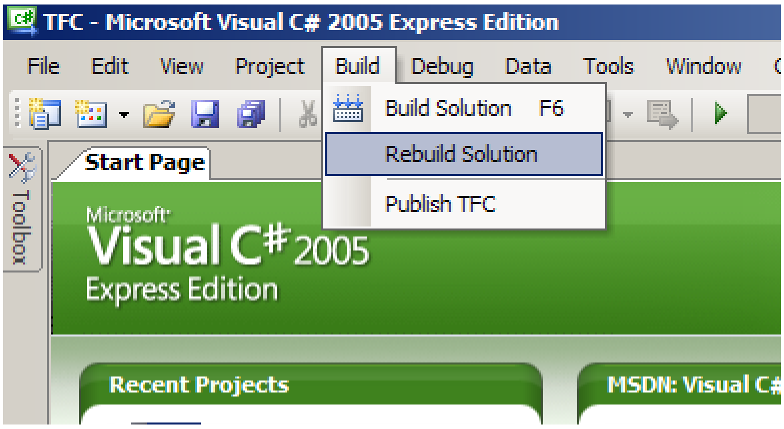
Ahora ejecutamos el Visual C# y hacemos click en la opción "Open Project" del menu "File" (figura 4). La ventana que nos aparece (figura 5) contiene los proyectos que hay instalados en el ordenador actual, navegamos a la carpeta TFC y hacemos doble click sobre el archivo de solución.
Así se abrirá el proyecto en el entorno de desarrollo, para recompilarlo una vez abierto únicamente hay que ejecutar la opción "Rebuild Solution" del menú "Build" (figura 6).



(figura 4)



(figura 5)


TFC - Microsoft Visual C# 2005 Express Edition
File
Edit
View
Project
Build
Debug
Data
Tools
Window
Build Solution F6
Rebuild Solution
Publish TFC
Start Page
Toolbox
Microsoft
Visual C# 2005
Express Edition
Recent Projects
MSDN: Visual C#

(figura 6)

4. Valoración Económica

El coste del desarrollo del programa será el siguiente:

Software

Visual C# Express Edition	Gratis
.Net Framework	Gratis
Pivo DNS Resolver	Gratis
MagicLibrary 1.7.4	Gratis
WebService Maxmind.com (50.000 requests)	18€

Mano de obra

Un programador		
horas trabajadas:	50	
sueldo por hora:	30€	
Total mano de obra:		1500€

Coste económico Total: Aproximadamente 1500€.

No se ha incluido el coste del sistema operativo Windows;
el precio máximo de la versión XP Profesional suele rondar los 150€ por licencia aproximadamente.

5. Conclusiones y posibles mejoras

En conclusión, el desarrollo del programa se ha realizado siguiendo las funcionalidades mínimas necesarias para el correcto funcionamiento del mismo. El rendimiento que se ha conseguido depende en gran medida de la velocidad de conexión a internet que posea el ordenador en el que se ejecuta el programa, así como de si se tiene acceso a los puertos que utilizan tanto el programa ping como el DNS resolver.

Viendo la evolución de las direcciones IP en el Internet del mundo del año 2005, sería la ampliación de la funcionalidad para hacerlo compatible con el protocolo IPv6.

6. Glosario

Direccion IP: Una dirección IP es un número que identifica a una interfaz de un dispositivo (habitualmente un ordenador) dentro de una red que utilice el protocolo IP.

Es habitual que un usuario que se conecta desde su hogar tenga una dirección IP que cambia cada cierto tiempo; eso es una *dirección IP dinámica* (normalmente se abrevia como *IP dinámica*).

ICMP: El Protocolo de Control de Mensajes de Internet (ICMP por sus siglas en inglés) es uno de los protocolos centrales de la suite de protocolos de Internet. Es usado principalmente por los Sistemas operativos de las computadoras en una red para enviar mensajes de error, indicando por ejemplo que un servicio determinado no está disponible ó que un router ó host no puede ser localizado.
más información: ver apéndice.

Ping: Se trata de una utilidad que comprueba el estado de la conexión con uno o varios equipos remotos, por medio de los paquetes de solicitud de eco y de respuesta de eco (definidos en el protocolo de red ICMP) para determinar si un sistema IP específico es accesible en una red. Es útil para diagnosticar los errores en redes o enrutadores IP.
más información: ver apéndice.

Traceroute: Traceroute es una herramienta de diagnóstico de redes que permite seguir la pista de los datagramas que van desde un host (punto de red) a otro. Se obtiene además una estadística de las velocidades de transmisión de esos paquetes.
más información: ver apéndice.

DNS: El Domain Name System (DNS) es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet. Aunque como base de datos el DNS es capaz de asociar distintos tipos de información a cada nombre, los usos más comunes son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico de cada dominio.
más información: ver apéndice.

7. Bibliografía

1.- **Microsoft Visual C# 2005 Step by Step (Microsoft)**
por John Sharp

2.- **Beginning Visual C# 2005 (Wrox Beginning Guides)**
por Karli Watson, Christian Nagel, Jacob Hammer Pedersen, Jon D. Reid, Morgan Skinner, Eric White.

3.- **Professional C# 2005 (Wrox Professional Guides)**
por Christian Nagel, Bill Evjen, Jay Glynn, Morgan Skinner, Karli Watson, Allen Jones.

8. Apéndices

8.1 Protocolo RFC 1876

A continuación se incluye información sobre el protocolo RFC 1876:

RFC 1876: A Means for Expressing Location Information in the Domain Name System

This memo defines an Experimental Protocol for the Internet community. This memo does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested.

1. Abstract
This memo defines a new DNS RR type for experimental purposes. This RFC describes a mechanism to allow the DNS to carry location information about hosts, networks, and subnets. Such information for a small subset of hosts is currently contained in the flat-file UUCP maps. However, just as the DNS replaced the use of HOSTS.TXT to carry host and network address information, it is possible to replace the UUCP maps as carriers of location information.
This RFC defines the format of a new Resource Record (RR) for the Domain Name System (DNS), and reserves a corresponding DNS type mnemonic (LOC) and numerical code (29).
This RFC assumes that the reader is familiar with the DNS [RFC 1034, RFC 1035]. The data shown in our examples is for pedagogical use and does not necessarily reflect the real Internet.

2. RDATA Format

```
    MSB                                           LSB
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
   0|        VERSION        |         SIZE          |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
   2|       HORIZ PRE       |       VERT PRE        |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
   4|                   LATITUDE                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
   6|                   LATITUDE                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
   8|                   LONGITUDE                   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
  10|                   LONGITUDE                   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
  12|                   ALTITUDE                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
  14|                   ALTITUDE                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

where:
VERSION: Version number of the representation. This must be zero. Implementations are required to check this field and make no assumptions about the format of unrecognized versions.

SIZE: The diameter of a sphere enclosing the described entity, in centimeters, expressed as a pair of four-bit unsigned integers, each ranging from zero to nine, with the most significant four bits representing the base and the second number representing the power of ten by which to multiply the base. This allows sizes from 0e0 (<1cm) to 9e9 (90,000km) to be expressed. This representation was chosen such that the hexadecimal representation can be read by eye; 0x15 = 1e5. Four-bit values greater than 9 are undefined, as are values with a base of zero and a non-zero exponent.
Since 20000000m (represented by the value 0x29) is greater than the equatorial diameter of the WGS 84 ellipsoid (12756274m), it is therefore suitable for use as a "worldwide" size.

HORIZ PRE: The horizontal precision of the data, in centimeters, expressed using the same representation as SIZE. This is the diameter of the horizontal "circle of error", rather than a "plus or minus" value. (This was chosen to match the interpretation of SIZE; to get a "plus or minus" value, divide by 2.)

VERT PRE: The vertical precision of the data, in centimeters, expressed using the sane representation as for SIZE. This is the total potential vertical error, rather than a "plus or minus" value. (This was chosen to match the interpretation of SIZE; to get a "plus or minus" value, divide by 2.) Note that if altitude above or below sea level is used as an approximation for altitude relative to the [WGS 84] ellipsoid, the precision value should be adjusted.

LATITUDE: The latitude of the center of the sphere described by the SIZE field, expressed as a 32-bit integer, most significant octet first (network standard byte order), in thousandths of a second of arc. 2^31 represents the equator; numbers above that are north latitude.

LONGITUDE: The longitude of the center of the sphere described by the SIZE field, expressed as a 32-bit integer, most significant octet first (network standard byte order), in thousandths of a second of arc, rounded away from the prime meridian. 2^31 represents the prime meridian; numbers above that are east longitude.

ALTITUDE: The altitude of the center of the sphere described by the SIZE field, expressed as a 32-bit integer, most significant octet first (network standard byte order), in centimeters, from a base of 100,000m below the [WGS 84] reference spheroid used by GPS (semimajor axis a=6378137.0, reciprocal flattening rf=298.257223563). Altitude above (or below) sea level may be used as an approximation of altitude relative to the the [WGS 84] spheroid, though due to the Earth's surface not being a perfect spheroid, there will be differences. (For example, the geoid (which sea level approximates) for the continental US ranges from 10 meters to 50 meters below the [WGS 84] spheroid.
Adjustments to ALTITUDE and/or VERT PRE will be necessary in most cases. The Defense Mapping Agency publishes geoid height values relative to the [WGS 84] ellipsoid.

3. Master File Format
The LOC record is expressed in a master file in the following format:

```
<owner> <TTL> <class> LOC ( d1 [m1 [s1]] {"N"|"S"} d2 [m2 [s2]] {"E"|"W"} alt["m"] [siz["m"]
                          [hp["m"] [vp["m"]]]] )
```

(The parentheses are used for multi-line data as specified in [RFC 1035] section 5.1.)

```
d1:      [0 .. 90]                          (degrees latitude)
d2:      [0 .. 180]                         (degrees longitude)
m1, m2:        [0 .. 59]                          (minutes latitude/longitude)
s1, s2:        [0 .. 59.999]                (seconds latitude/longitude)
alt:     [-100000.00 .. 42849672.95] BY .01       (altitude in meters)
siz, hp, vp: [0 .. 90000000.00]                   (size/precision in meters)
```

If omitted, minutes and seconds default to zero, size defaults to 1m, horizontal precision defaults to 10000m, and vertical precision defaults to 10m. These defaults are chosen to represent typical ZIP/postal code area sizes, since it is often easy to find approximate geographical location by ZIP/postal code.

4. Example Data
;;;
;;; note that these data would not all appear in one zone file

;;;
;; network LOC RR derived from ZIP data. note use of precision defaults

cambridge-net.kei.com.

LOC 42 21 54 N 71 06 18 W -24m 30m

;; higher-precision host LOC RR. note use of vertical precision default

loiosh.kei.com.

LOC 42 21 43.952 N 71 5 6.344 W -24m 1m 200m

pipex.net.

LOC 52 14 05 N 00 08 50 E 10m

curtin.edu.au.

LOC 32 7 19 S 116 2 25 E 10m

rwy04L.logan-airport.boston.

LOC 42 21 28.764 N 71 00 51.617 W -44m 2000m

5. Application use of the LOC RR

5.1 Suggested Uses

Some uses for the LOC RR have already been suggested, including the USENET backbone flow maps, a "visual traceroute" application showing the geographical path of an IP packet, and network management applications that could use LOC RRs to generate a map of hosts and routers being managed.

5.2 Search Algorithms

This section specifies how to use the DNS to translate domain names and/or IP addresses into location information.
If an application wishes to have a "fallback" behavior, displaying a less precise or larger area when a host does not have an associated LOC RR, it MAY support use of the algorithm in section 5.2.3, as noted in sections 5.2.1 and 5.2.2. If fallback is desired, this behaviour is the RECOMMENDED default, but in some cases it may need to be modified based on the specific requirements of the application involved.
This search algorithm is designed to allow network administrators to specify the location of a network or subnet without requiring LOC RR data for each individual host. For example, a computer lab with 24 workstations, all of which are on the same subnet and in basically the same location, would only need a LOC RR for the subnet.
(However, if the file server's location has been more precisely measured, a separate LOC RR for it can be placed in the DNS.)

5.2.1 Searching by Name

If the application is beginning with a name, rather than an IP address (as the USENET backbone flow maps do), it MUST check for a LOC RR associated with that name. (CNAME records should

be followed as for any other RR type.)
If there is no LOC RR for that name, all A records (if any) associated with the name MAY be checked for network (or subnet) LOC RRs using the "Searching by Network or Subnet" algorithm (5.2.3). If multiple A records exist and have associated network or subnet LOC RRs, the application may choose to use any, some, or all of the LOC RRs found, possibly in combination. It is suggested that multi-homed hosts have LOC RRs for their name in the DNS to avoid any ambiguity in these cases.
Note that domain names that do not have associated A records must have a LOC RR associated with their name in order for location information to be accessible.

5.2.2 Searching by Address

If the application is beginning with an IP address (as a "visual traceroute" application might be) it MUST first map the address to a name using the IN-ADDR.ARPA namespace (see [RFC 1034], section 5.2.1), then check for a LOC RR associated with that name. If there is no LOC RR for the name, the address MAY be checked for network (or subnet) LOC RRs using the "Searching by Network or Subnet" algorithm (5.2.3).

5.2.3 Searching by Network or Subnet

Even if a host's name does not have any associated LOC RRs, the network(s) or subnet(s) it is on may. If the application wishes to search for such less specific data, the following algorithm SHOULD be followed to find a network or subnet LOC RR associated with the IP address. This algorithm is adapted slightly from that specified in [RFC 1101], sections 4.3 and 4.4.
Since subnet LOC RRs are (if present) more specific than network LOC RRs, it is best to use them if available. In order to do so, we build a stack of network and subnet names found while performing the [RFC 1101] search, then work our way down the stack until a LOC RR is found.

1. create a host-zero address using the network portion of the IP address (one, two, or three bytes for class A, B, or C networks, respectively). For example, for the host 128.9.2.17, on the class B network 128.9, this would result in the address "128.9.0.0".

2. Reverse the octets, suffix IN-ADDR.ARPA, and query for PTR and A records.
Retrieve:
0.0.9.128.IN-ADDR.ARPA. PTR isi-net.isi.edu.
A 255.255.255.0
Push the name "isi-net.isi.edu" onto the stack of names to be searched for LOC RRs later.

3. Since an A RR was found, repeat using mask from RR (255.255.255.0), constructing a query for 0.2.9.128.IN-ADDR.ARPA.
Retrieve:
0.2.9.128.IN-ADDR.ARPA. PTR div2-subnet.isi.edu.
A 255.255.255.240
Push the name "div2-subnet.isi.edu" onto the stack of names to be searched for LOC RRs later.

4. Since another A RR was found, repeat using mask 255.255.255.240 (x'FFFFFFF0'), constructing a query for 16.2.9.128.IN-ADDR.ARPA.
Retrieve:
16.2.9.128.IN-ADDR.ARPA. PTR inc-subsubnet.isi.edu.
Push the name "inc-subsubnet.isi.edu" onto the stack of names to be searched for LOC RRs later.

5. Since no A RR is present at 16.2.9.128.IN-ADDR.ARPA., there are no more subnet levels to search. We now pop the top name from the stack and check for an associated LOC RR. Repeat until a LOC RR is found.

In this case, assume that inc-subsubnet.isi.edu does not have an associated LOC RR, but that div2-subnet.isi.edu does. We will then use div2-subnet.isi.edu's LOC RR as an approximation of this host's location. (Note that even if isi-net.isi.edu has a LOC RR, it will not be used if a subnet also has a LOC RR.)

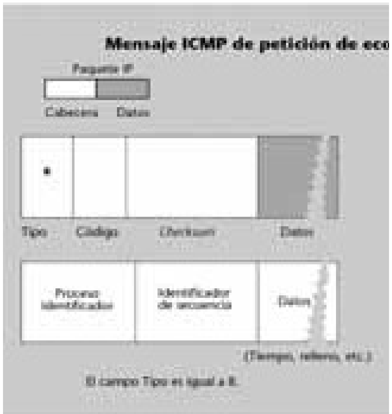
5.3 Applicability to non-IN Classes and non-IP Addresses

The LOC record is defined for all RR classes, and may be used with non-IN classes such as HS and CH. The semantics of such use are not defined by this memo.
The search algorithm in section 5.2.3 may be adapted to other addressing schemes by extending [RFC 1101]'s encoding of network names to cover those schemes. Such extensions are not defined by this memo.

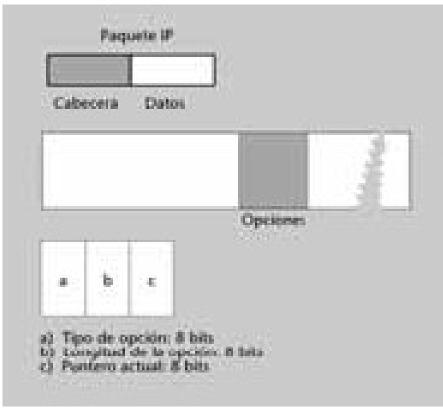
8.2 Ping

El programa ping permite descubrir si una estación se encuentra activa o no, simplemente efectuando lo siguiente:

La instrucción ping envía un mensaje ICMP del tipo 8 (petición de eco) con el destino indicado. El receptor de la petición debe responder con una respuesta de eco (ICMP tipo 0), y, cuando el ping la recibe, indica en pantalla que la estación remota está activa.



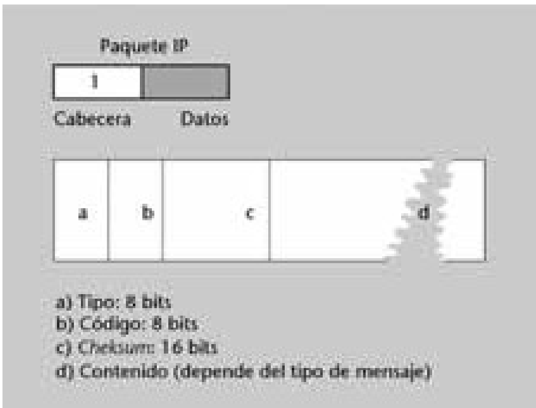
En sistemas multitarea puede haber más de una petición en curso. Para saber a cuál de las instrucciones ping debe entregarse la respuesta, cada ping asigna el identificador del proceso ping a los mensajes generados. El campo Identificador de secuencia permite al ping enviar diferentes paquetes e identificar sus respuestas (ping –t en MSWindows y ping en UNIX*). El campo de datos permite añadir una cadena arbitraria de bytes a la petición de eco (ping –s en UNIX y ping –t en MSWindows). La respuesta debe llevar una copia de dichos bytes. Con el ping tenemos otras opciones disponibles. En particular, la opción de memorización de rutas (*record route* o RR)*. Esta opción no se refleja en ninguno de los campos del mensaje ICMP, sino que se encuentra dentro de la misma cabecera IP, en el campo de opciones.
Las diferentes opciones que se encuentran disponibles en la cabecera se identifican por medio del primer byte del campo de opciones de la cabecera IP:



Si el originador quiere activar la opción RR, el primer byte debe ser 7. En todas las opciones el primer byte indica el tipo, y el segundo, la longitud en bytes de la opción. En este caso siempre se pide el máximo posible, que es 39 bytes, puesto que el campo siguiente tiene un byte (puntero), al que sigue una cadena de campos de 4 bytes (que son las direcciones IP encontradas). El campo Puntero se inicializa a 4, y dentro de los 4 bytes siguientes se guarda la dirección de la estación en que ejecutamos el ping.
Cada direccionador debe mirar dentro de los paquetes IP (ICMP o no) para ver si tienen la opción RR activada. Si un direccionador encuentra un paquete con esta opción activada, modifica la posición apuntada por el puntero con su dirección (por norma general, la dirección de salida del direccionador) e incrementa el valor del puntero en cuatro unidades. Cuando vuelve el mensaje de respuesta de eco, se ha añadido una lista con todos los saltos que ha tenido que realizar el paquete (tanto de ida, como de vuelta).
El campo de opciones tiene limitaciones de tamaño: sólo dispone de 36 bytes (39 - 3) para guardar direcciones IP. Como cada dirección ocupa 4 bytes, sólo hay espacio para nueve direcciones IP. Si a ello le añadimos que no todos los direccionadores comprueban si hay opciones dentro de los paquetes IP, o no actualizan la opción RR, hace poco útil este tipo de ping en el mundo real.

8.3 Traceroute e ICMP

Los mensajes ICMP viajan dentro de paquetes IP (al contrario de lo que sucedía con los paquetes ARP), en el campo de datos con el campo Protocolo igual a 1. El formato del mensaje presentado en la figura siguiente nos facilitará el estudio de los diferentes usos del paquete ICMP:



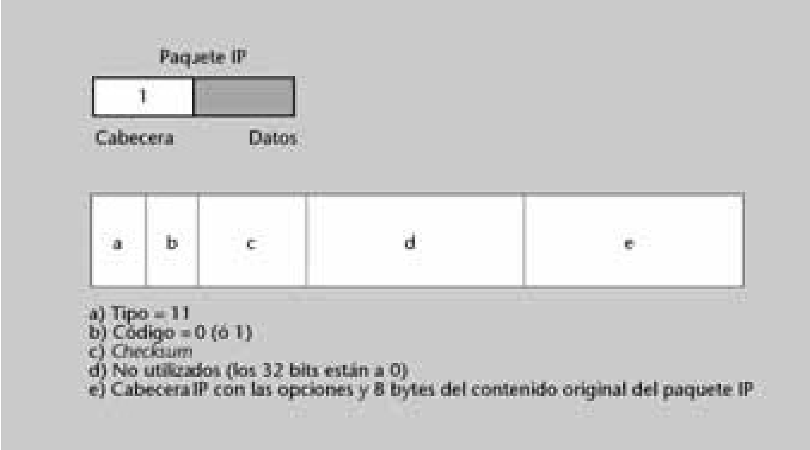
Existen trece *tipos* de mensajes ICMP en uso en la actualidad, y alrededor de una treintena de subtipos identificados con el campo Código.

Tipo	Código	Descripción	Clase
0	0	Respuesta de eco (echo reply)	Petición
8	0	Petición de eco (echo request)	Petición
3	0-15	Destino inalcanzable (unreachable destination)	Error
4	0	Petición de control de flujo	Error
5	0-3	Redireccionamiento	Error
9	0	Publicación de rutas	Petición
10	0	Petición de rutas	Petición
11	0-1	Tiempo de vida expirado (time exceeded)	Error
12	0-1	Cabecera IP incorrecta	Error
13	0	Petición de hora	Petición
Tipo	**Código**	**Descripción**	**Clase**
14	0	Respuesta de hora	Petición
17	0	Petición de máscara de subred	Petición
18	0	Respuesta de máscara de subred	Petición

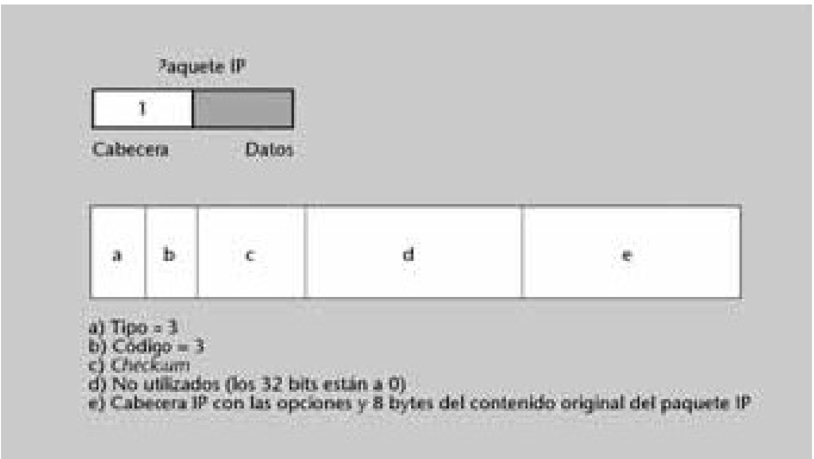
La última columna nos permite distinguir mensajes ICMP de notificación de error de mensajes que son parte de una petición (la petición o la respuesta).
Esta distinción es importante, puesto que los mensajes de error ICMP no pueden generar otros mensajes de error ICMP. En particular, no se generan mensajes de error en respuesta a los paquetes o mensajes siguientes:
• Los mensajes de error ICMP.
• Un paquete IP destinado a una dirección *broadcast* (sea un *broadcast* IP o un *broadcast* MAC).
• Un fragmento que no sea el primero.
• Una dirección de origen que no identifique una única estación. Por ejemplo, la dirección de origen válida 0.0.0.0 o la dirección de origen no válida 255.255.255.255.

El programa traceroute permite encontrar las rutas entre un origen y un destino sin ninguna de las limitaciones del ping -R (ping -r en MSWindows). Utiliza un mecanismo bastante ingenioso basado en mensajes genéricos ICMP (y no los específicos petición de eco y respuesta de eco del ping).
El funcionamiento se basa en la explotación de dos mensajes ICMP:

1) Tiempo de vida agotado (*time-exceeded*): cuando un direccionador recibe un paquete, aparte de las tareas primordiales de direccionamiento, debe reducir en una unidad el valor del campo TTL de la cabecera IP.
En caso de que el valor (después de la reducción) sea cero, el paquete debe eliminarse.
Sin embargo, esta eliminación no es silenciosa, sino que el direccionador responsable envía una notificación de la misma al originador del paquete por medio de un mensaje ICMP tipo 11 y código 0 (tiempo de vida agotado).
Este paquete ICMP contiene la cabecera del paquete IP que se ha eliminado y los primeros 8 bytes de su contenido (seguramente la cabecera UDP o los primeros bytes de la cabecera TCP).



2) Puerto inalcanzable (*unreachable-port*): cuando una estación recibe un datagrama UDP o un segmento TCP destinado a un puerto que la máquina no escucha, responde con un mensaje de error de puerto inalcanzable (tipo 3 con código 3).



El programa traceroute simplemente debe enviar paquetes al destino con TTL secuencialmente ascendentes: el paquete (con independencia del tipo que sea) que tenga el TTL = 1 será rechazado por el primer direccionador, el que tenga TTL = 2 lo será por el segundo, y así consecutivamente. Cada uno de los direccionadores devolverá un mensaje ICMP “tiempo de vida agotado”, una pista del todo suficiente para que el originador averigüe el camino que han seguido todos los paquetes.
Cuando el mensaje llega al destino, debe devolver algún mensaje para saber que la secuencia ha finalizado. Por norma general, el mensaje será “puerto inalcanzable” si el mensaje enviado era un datagrama UDP a un puerto no usado, o bien una respuesta de eco si lo que se ha enviado son paquetes ICMP de petición de eco.