Escuela Politécnica

Proyecto Fin de Carrera

Generación, análisis y optimización de escenarios de migración de redes IPv4/IPv6 mediante programación funcional

Diciembre 2008

José Franco Campos

Estudio del problema

Ventajas de IPv6

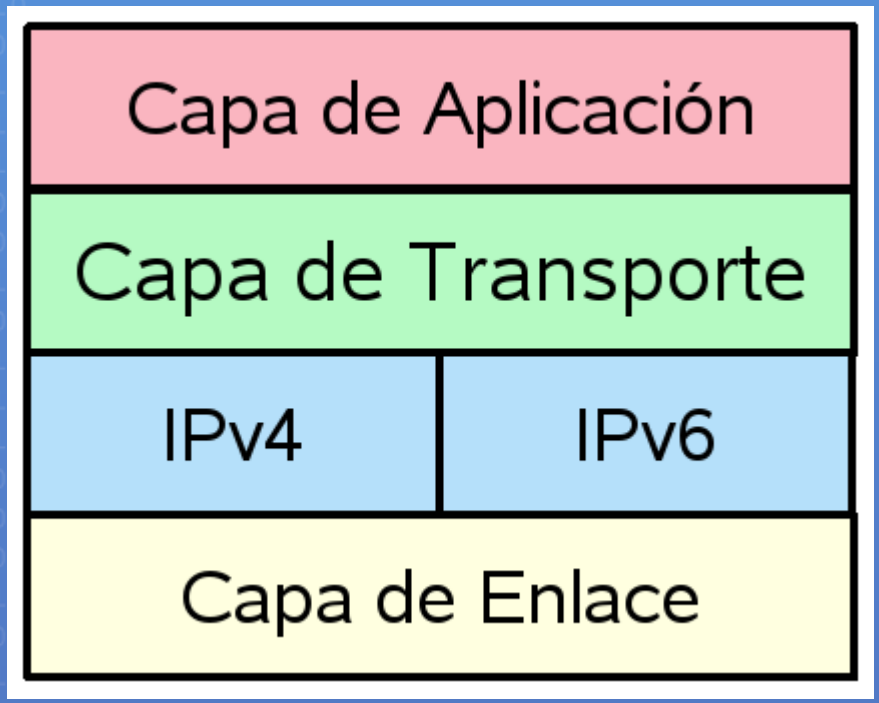
- Sustituye a IPv4.
- Expande enormemente el espacio de direcciones ($2^{32} \rightarrow 2^{128}$).
- Facilita el enrutamiento jerárquico.
- Ofrece opciones de autoconfiguración.
- Formato de cabecera más sencillo y flexible.

Transición a IPv6

- Ambas versiones son incompatibles.
- Es necesario que convivan durante un tiempo.
- Se definen varios mecanismos de transición:
  - Dual Stack.
  - Túneles.
  - Traducción de protocolos.

Dual Stack

Implementación de ambas pilas de protocolos.



Túneles

- Manuales.- Campo “protocolo” a 41.
- Broker.- Servidor centralizado de creación de túneles.
- Túneles automáticos.- Se usan cuando los extremos del túnel son los extremos de la conexión.
- 6to4.- Nodos IPv6 aislados que desean conectividad.
- DSTM.- Nodos IPv4 aislados en una red IPv6.
- ISATAP.- Nodos IPv6 dentro de una intranet.
- Teredo.- IPv6 a través de UDP IPv4 para atravesar NAT.

Traducción de protocolos

- SIIT.- Traducción de direcciones sin estado.
- NAT-PT.- Traducción de direcciones/puerto con estado.
- TRT.- Traducción a nivel de transporte.
- SOCKS.- Traducción a nivel de aplicación.

Análisis

Necesidad

- Gran variedad de mecanismos a considerar.
- Necesidad de implementar una herramienta que permita resolver escenarios de migración.
- Primero hay que formalizar el problema.

Objetivos

- Definición de escenarios de transición.
- Búsqueda de soluciones.
- Evaluación y ordenación de soluciones.

Metodología

- Usaremos la Metodología MENINA.
- Describe escenarios de transición:
  - Aplicaciones.
  - Nodos terminales.
  - Routers intermedios.
- Proporciona una base de reglas para buscar soluciones.

Definición de los escenarios

Representación algebraica de un escenario:

$$(A_1, N_1, RD_f, N_2, A_2)$$

$$A_1, A_2 \in \{A_4, A_6, A_d\}$$

$$N_1, N_2 \in \{N_4, N_6, N_d, N_{d,map}\}$$

$$R_i \in \{R_4, R_6, R_d, R_4^t, R_d^t, R_{d,tp}, R_{d,tp}^t\}$$

Fases de la metodología

- Formalización.
  - Creación de zonas.
  - Creación de los operadores de conexión.
- Canonización.
- Túneles.
- Mecanismos sobre redes finales.

Diseño

Almacenamiento de topologías

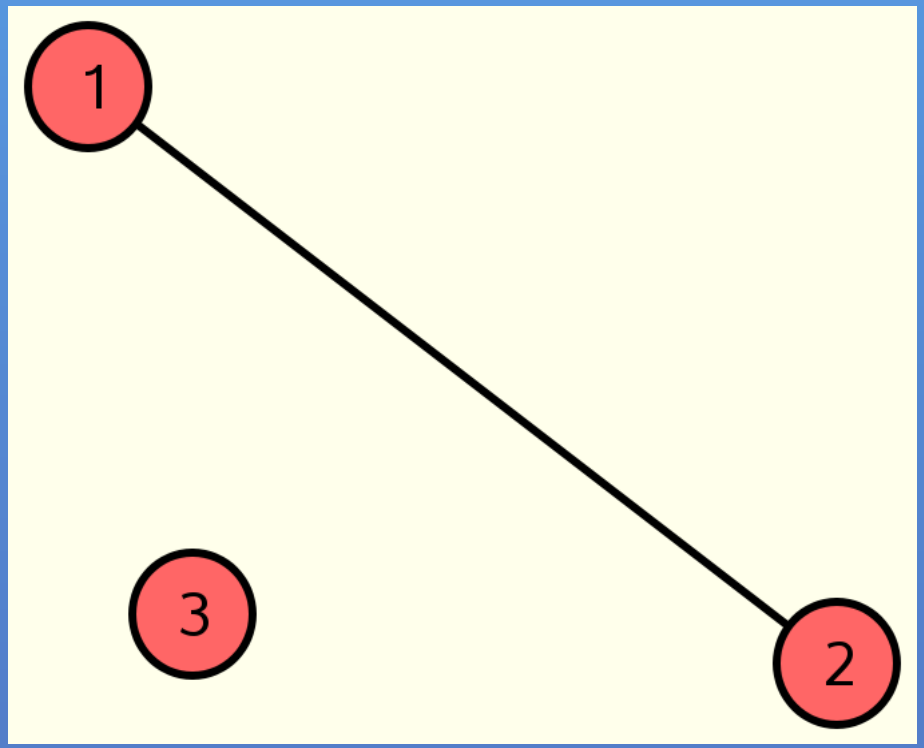
- Necesitamos poder leer y escribir topologías.
- MENINA usa escenarios lineales:
  - Necesidad de un algoritmo de búsqueda de rutas.

Graph Modelling Language (I)

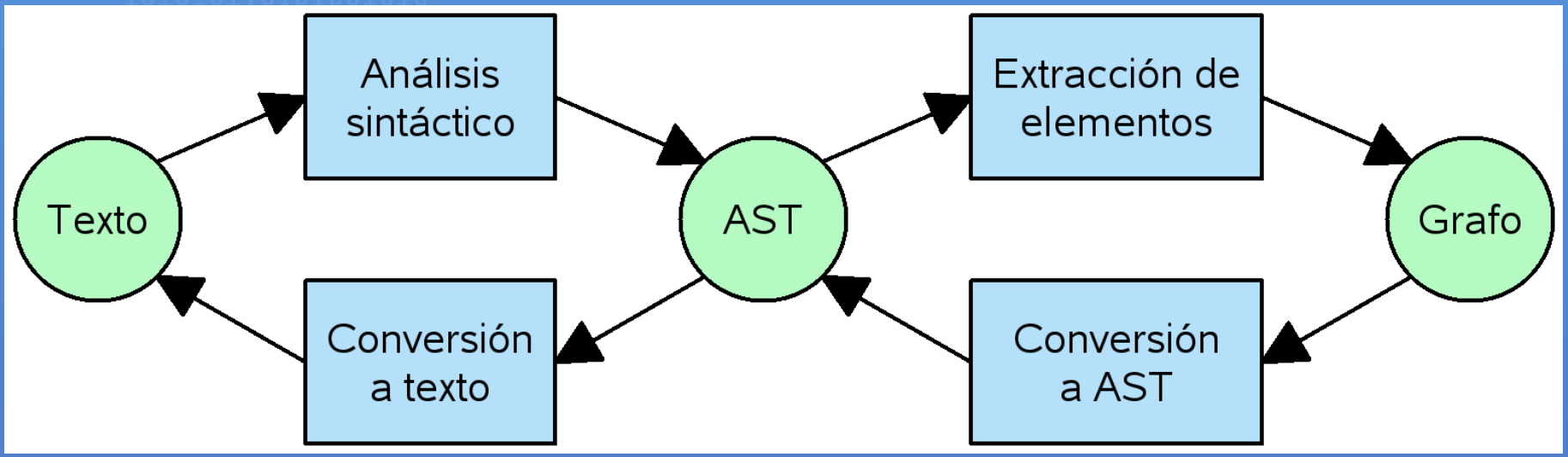
- Formato de almacenamiento de grafos genéricos.
- Muy sencillo, basado en texto estructurado.
- Posibilidad de expansión, añadiendo nuevos atributos adaptados a nuestra aplicación.

Graph Modelling Language (II)

```
graph [
    node [ id 1 ]
    node [ id 2 ]
    node [ id 3 ]
    edge [
        source 1
        target 2
    ]
```



Proceso de carga

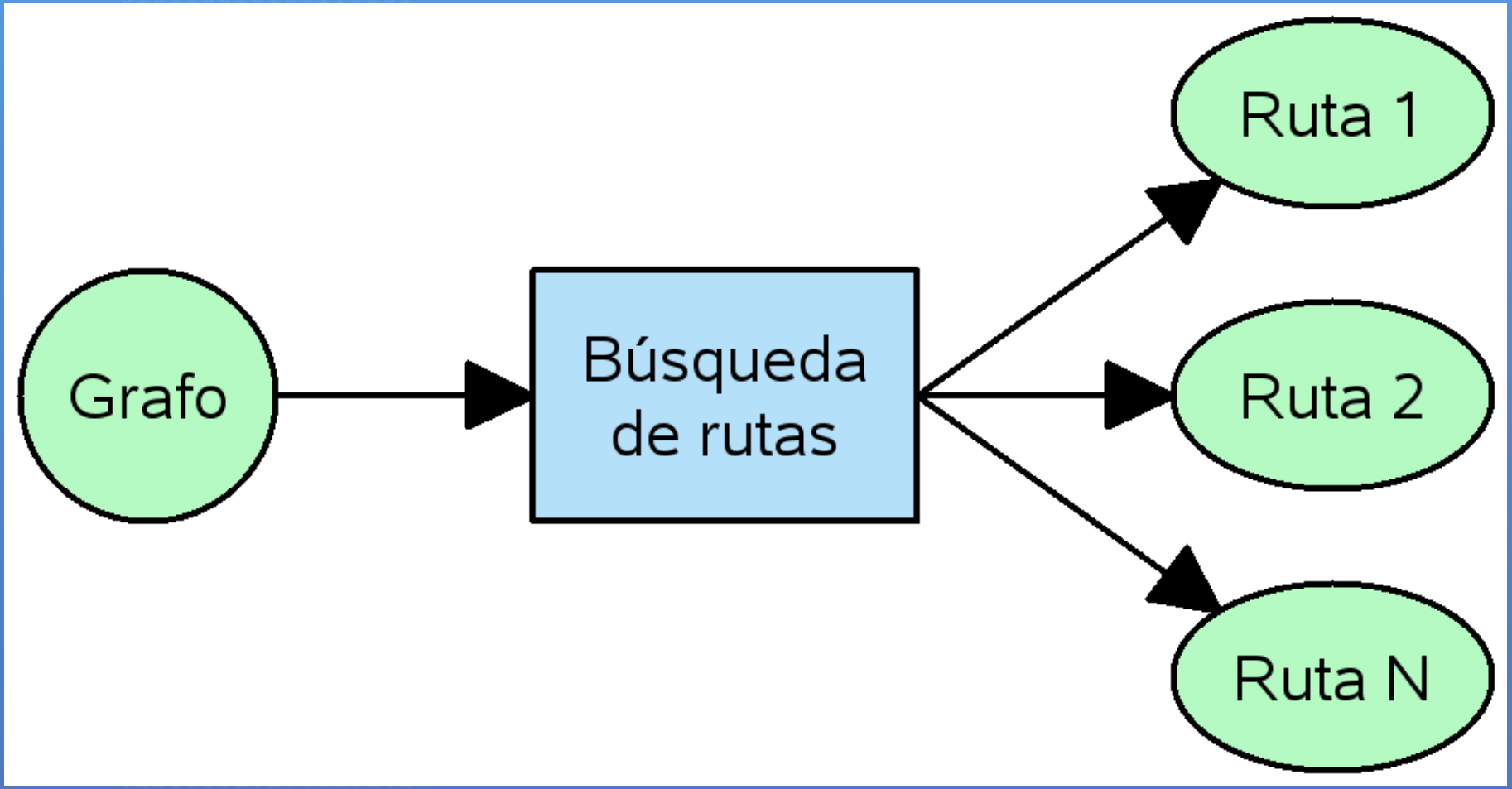


Estrategia de resolución (I)

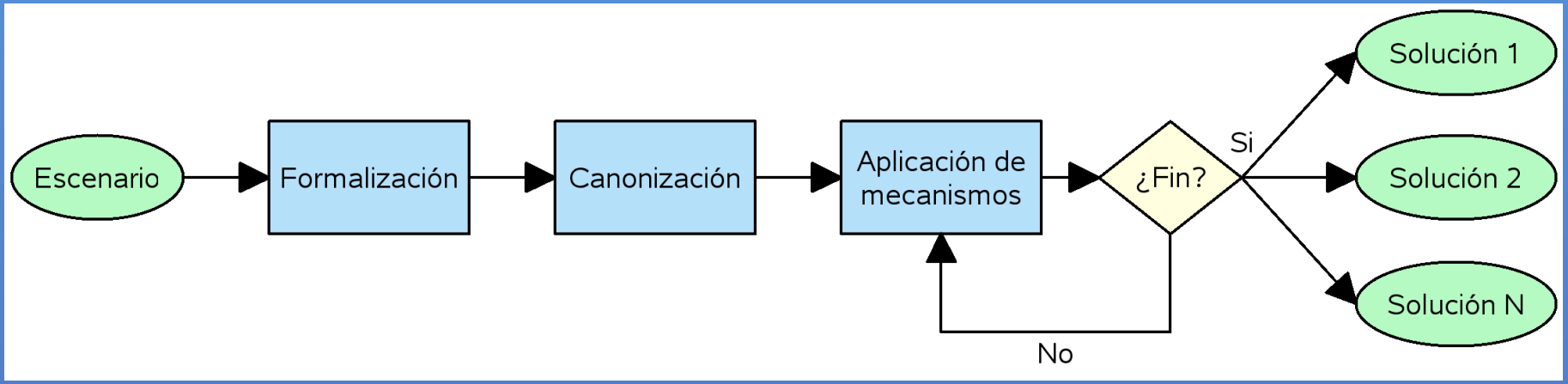
Tres pasos:

- Obtención de rutas lineales.
- Búsqueda de soluciones.
- Evaluación y ordenación de soluciones.

Estrategia de resolución (II)



Estrategia de resolución (III)



Flujo de ejecución

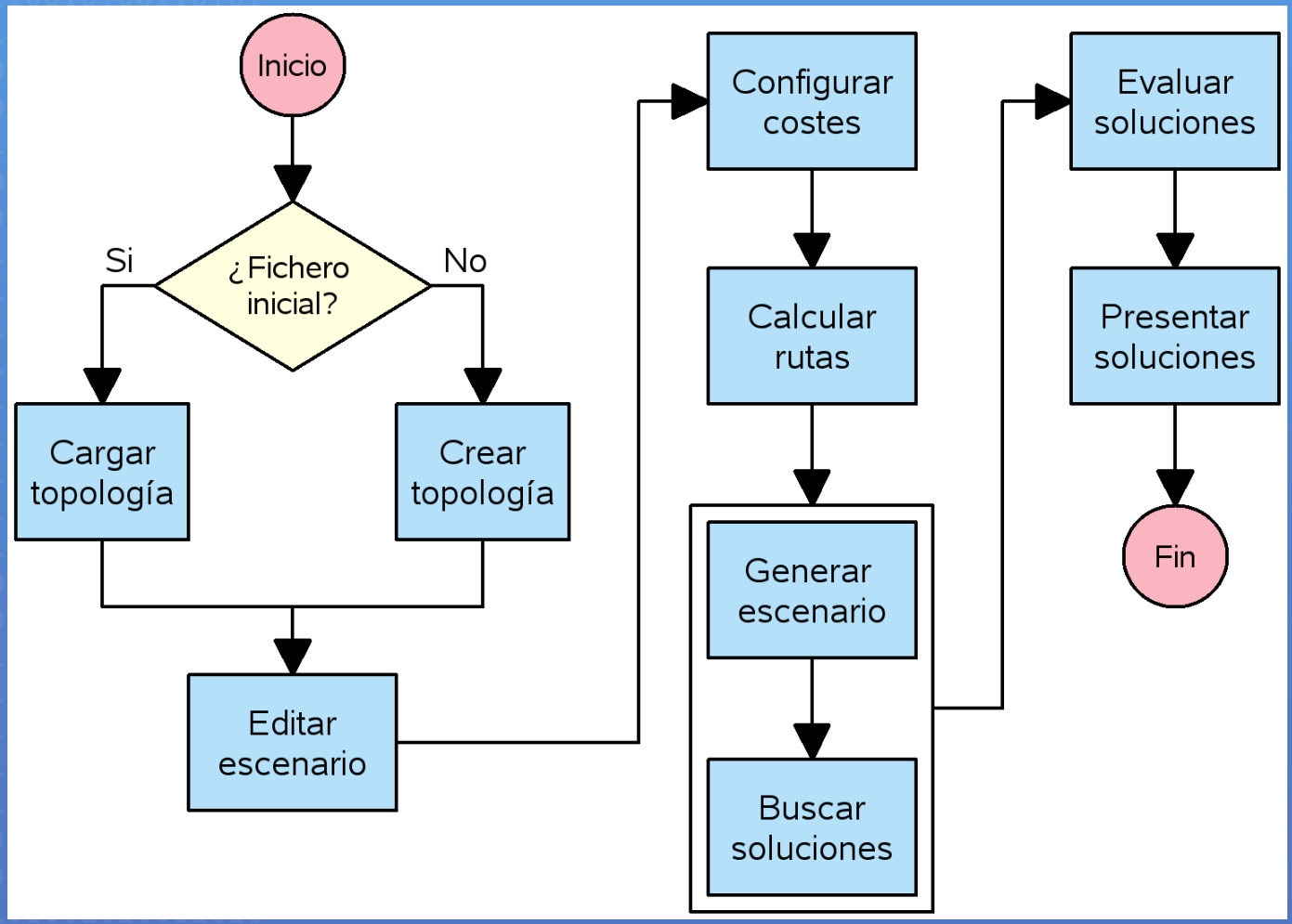
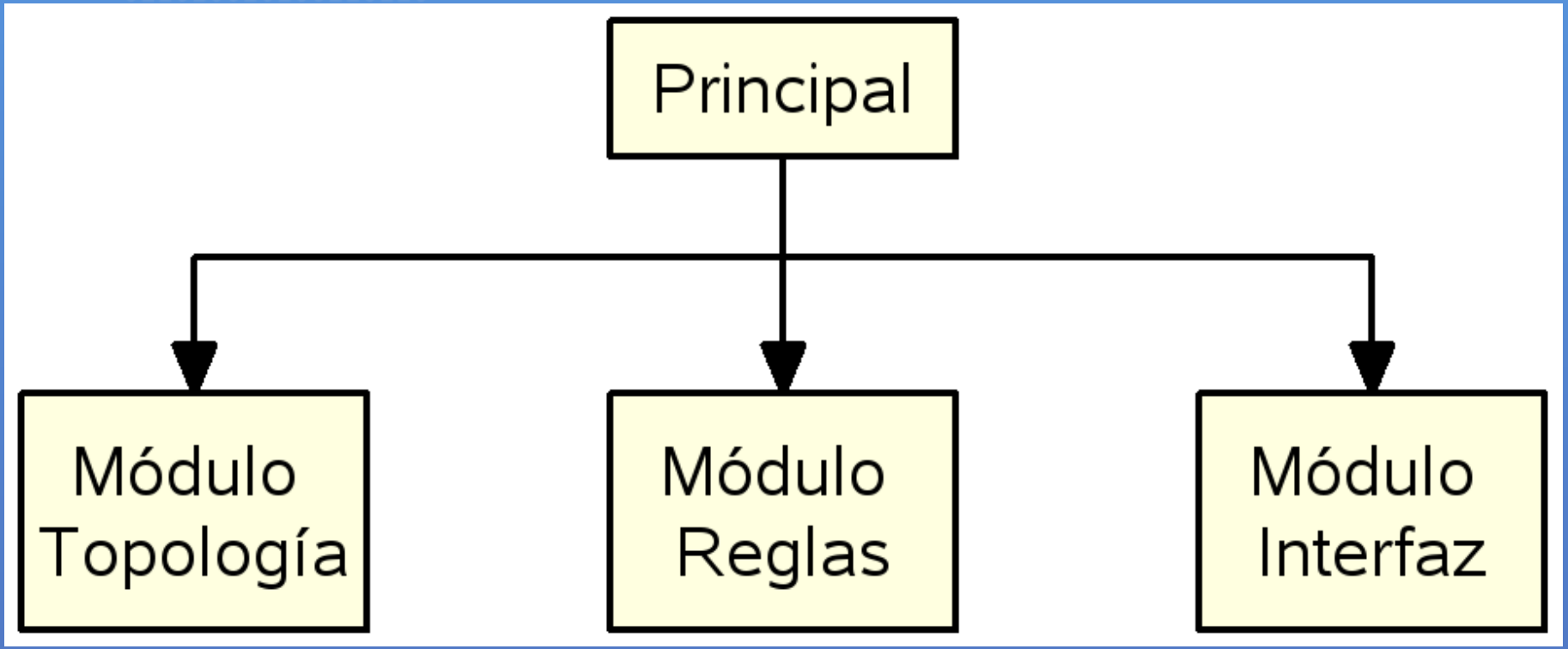


Diagrama de módulos



Implementación

Módulo Topología

Funcionalidades:

- Carga y almacenamiento de los grafos.
- Búsqueda de rutas.

Módulo Topología

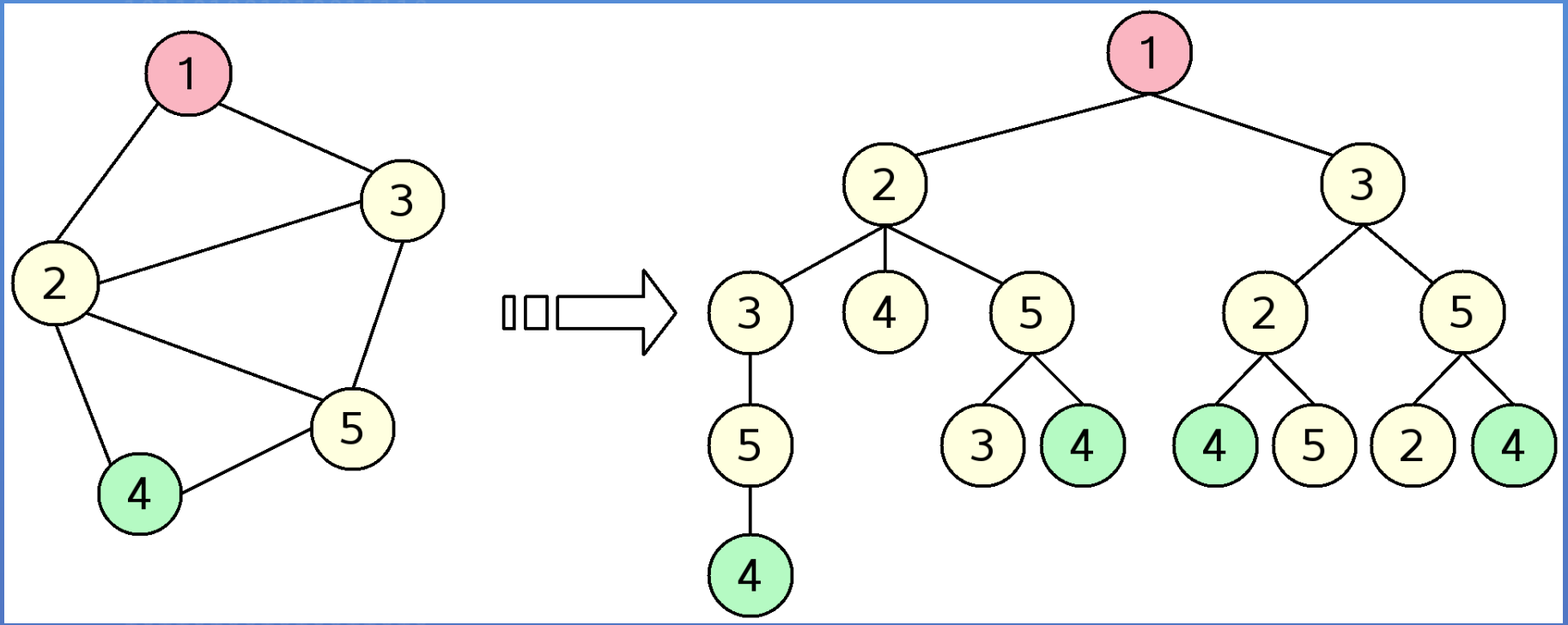
Estudio de conectividad:

Fichero	Nodos	Enlaces	Conectividad absoluta			Conectividad relativa			Densidad
			Mínima	Media	Máxima	Mínima	Media	Máxima	
abilene.gml	11	14	2	2,55	3	18,18 %	23,14 %	27,27 %	0,1209
abovenet.6461.r0.cch.gml	368	966	0	5,25	39	0,00 %	1,43 %	10,60 %	0,0008
att.7018.r0.cch.gml	731	2253	0	6,16	55	0,00 %	0,84 %	7,52 %	0,0003
att.gml	154	188	1	2,44	29	0,65 %	1,59 %	18,83 %	0,0088
colt.gml	43	59	1	2,74	8	2,33 %	6,38 %	18,61 %	0,0251
cw.gml	33	107	2	6,49	13	6,06 %	19,65 %	39,39 %	0,0058
dfn.gml	30	97	1	6,47	20	3,33 %	21,56 %	66,67 %	0,0064
ebone.1755.r0.cch.gml	161	307	0	3,81	17	0,00 %	2,37 %	10,56 %	0,0034
exodus.3967.r0.cch.gml	246	540	0	4,39	18	0,00 %	1,79 %	7,32 %	0,0017
geant-20041125.gml	23	37	2	3,22	6	8,70 %	13,99 %	26,09 %	0,0345
geant.gml	23	37	2	3,22	6	8,70 %	13,99 %	26,09 %	0,0345
level3.3356.r0.cch.gml	625	5298	0	16,95	169	0,00 %	2,71 %	27,04 %	0,0000
sprint.1239.r0.cch.gml	549	1593	0	5,8	51	0,00 %	1,06 %	9,29 %	0,0004
switch.gml	31	42	1	2,71	10	3,23 %	8,74 %	32,26 %	0,0360
telekom.gml	10	17	2	3,4	7	20,00 %	34,00 %	70,00 %	0,0735
tiscali.3257.r0.cch.gml	248	405	0	3,27	31	0,00 %	1,32 %	12,50 %	0,0030
verrio.2914.r0.cch.gml	911	2217	0	4,87	46	0,00 %	0,53 %	5,05 %	0,0004
vsnl.4755.r0.cch.gml	12	12	0	2	4	0,00 %	16,67 %	33,33 %	0,1818

Módulo Topología

Búsqueda de rutas:

- Conversión de los grafos a árboles

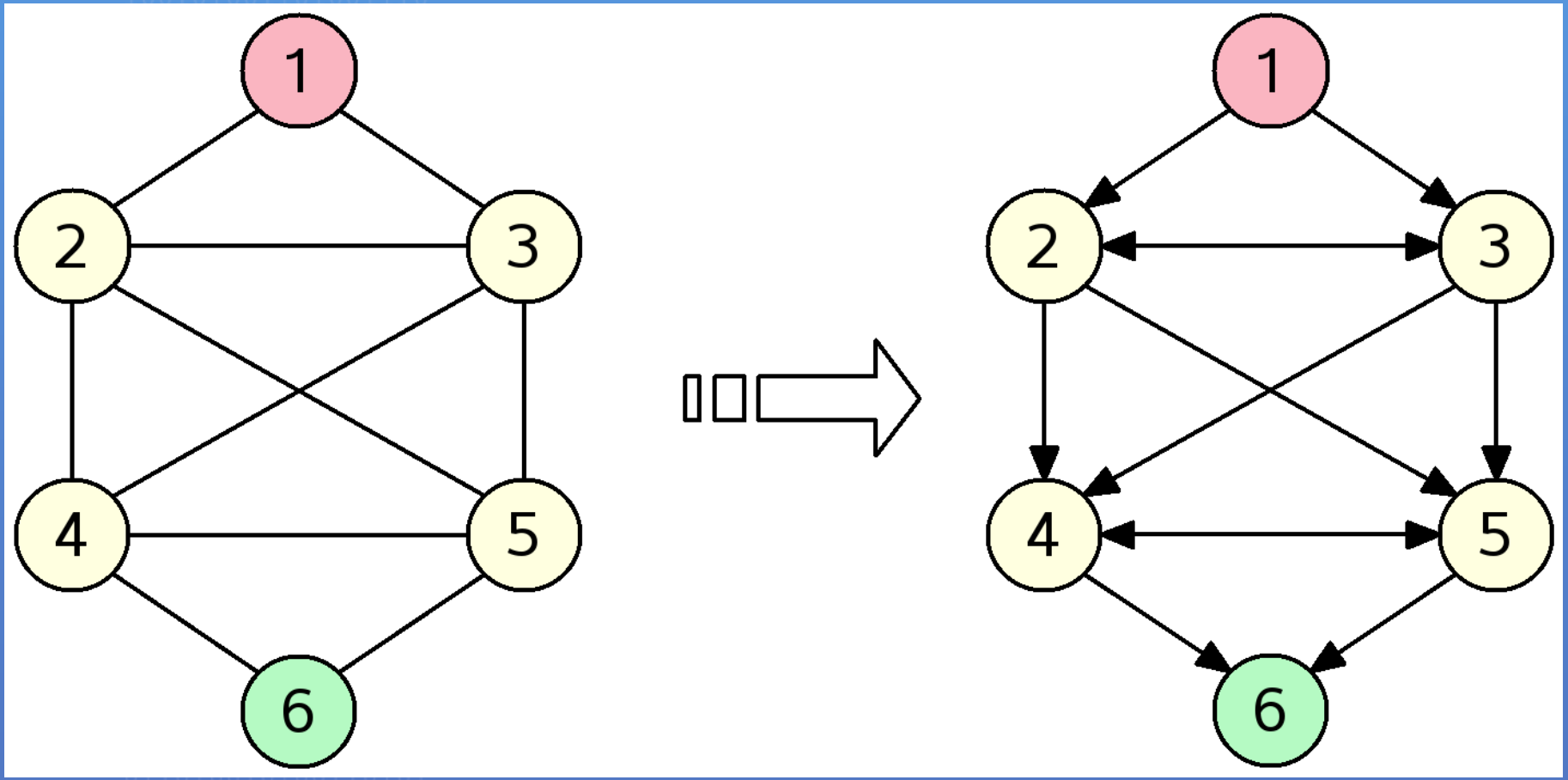


Módulo Topología

Criterios de poda:

- Forzar nodos adyacentes.
- Limitar la longitud máxima del camino.
- Eliminar bucles del grafo.

Módulo Topología



Módulo Reglas

Funcionalidades:

- Implementación de la base de reglas.
- Estrategia de resolución.
- Evaluación y ordenación de soluciones.

Módulo Reglas

Implementación de la base de reglas:

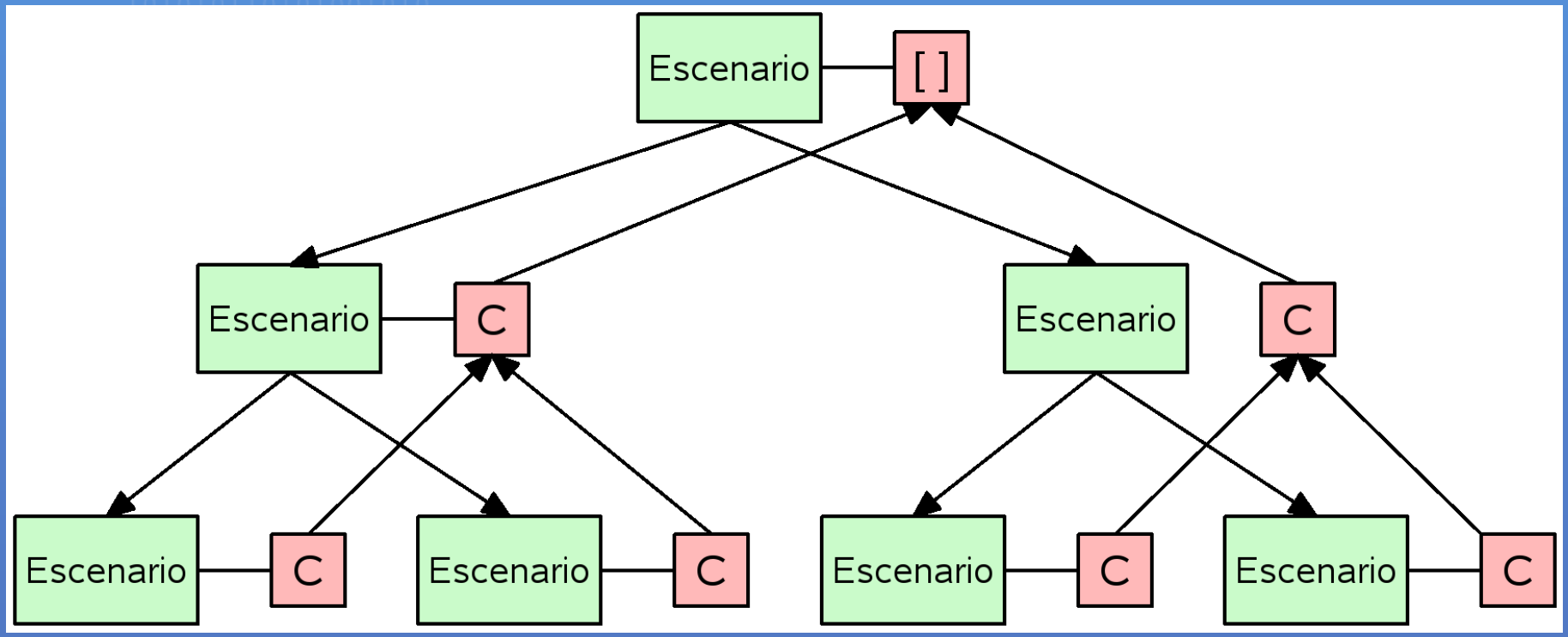
- Hemos usado HaRP.
- Nos permite escribir patrones con expresiones regulares.
- Ejemplo:

$$Z_4(\otimes_d Z_4)* = Z_4$$

```
[/ (conexion,Z4) , r@(OpD, Z4)+! ,
 rs@_* /]
```

Módulo Reglas

Seguimiento de cambios:



Módulo Interfaz

Se evaluaron cuatro alternativas:

- Gtk2hs
- QtHaskell
- WxHaskell
- Htk

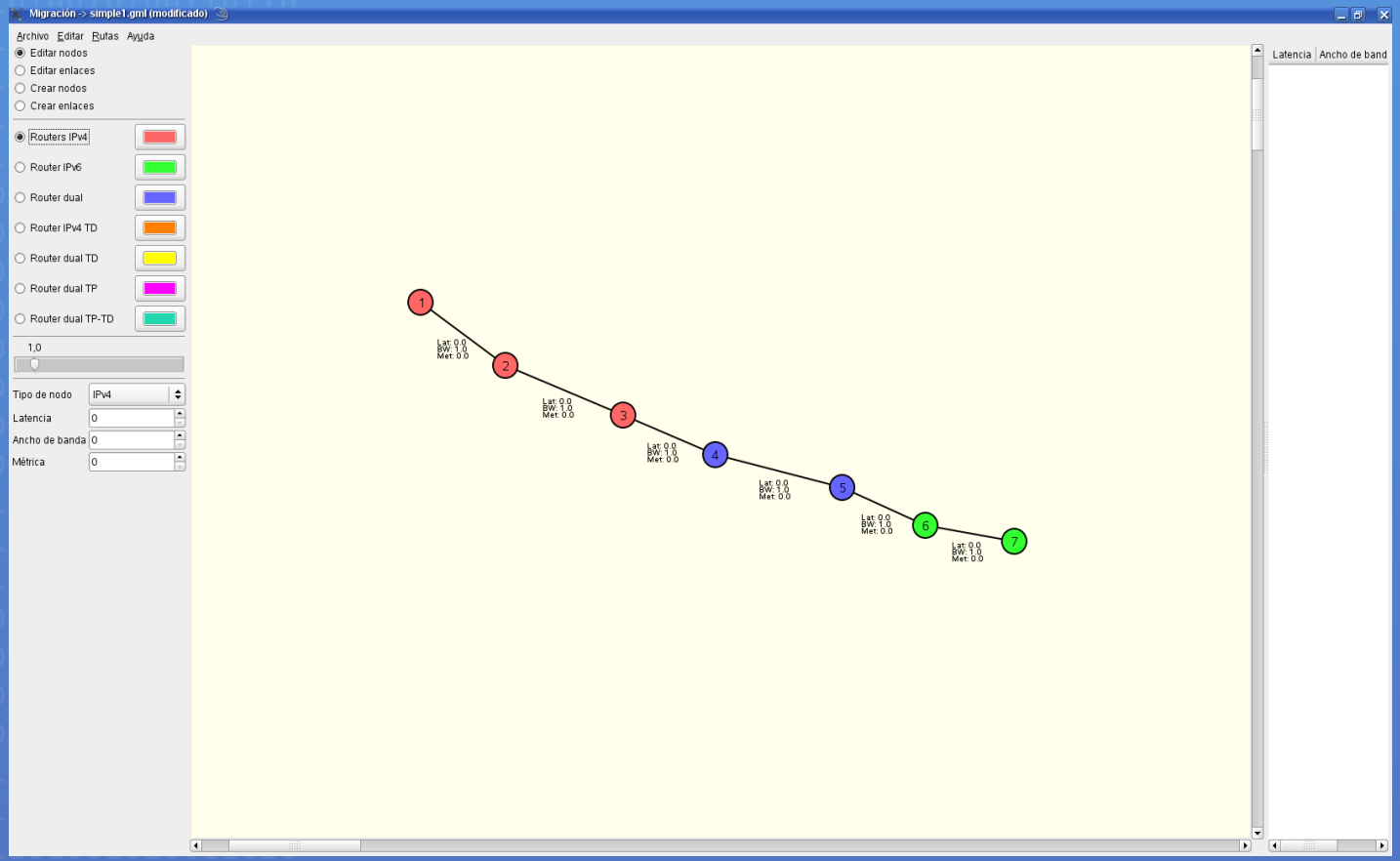
Elegimos Gtk2hs.

Módulo Interfaz

- Estilo de programación claramente imperativo.
- Aun así, el código es sencillo.
- Pasos para crear una interfaz gráfica:
  - Diseñar la interfaz con Glade y guardarla en XML.
  - Durante la ejecución:
    - La aplicación carga el fichero XML.
    - Extrae los widgets.
    - Establece los manejadores de señal.
    - Llama al bucle principal de la interfaz.

Módulo Interfaz

Resultado:



Pruebas

Pruebas con QuickCheck

- Herramienta de generación automática de bancos de pruebas.
- El programador define propiedades que deben cumplir las funciones.
- Al llamar a QuickCheck, genera pruebas aleatorias y verifica que se cumplan las propiedades para todos los casos.

Pruebas con QuickCheck

Ejemplo:

```
prob_texto :: Valor -> Bool
prob_texto a =
   (leerGML “” . escribirGML) a == a
```

Pruebas con QuickCheck

Propiedades definidas:

- La conversión Grafo→Texto→Grafo devuelve el grafo original.
- Después de la creación de zonas, no hay dos zonas contiguas del mismo tipo.
- Las reglas de canonización son conmutativas.
- La aplicación de una regla no aumenta el tamaño de un escenario.

Pruebas unitarias

Pruebas implementadas manualmente:

- Resultado de los diferentes criterios de poda.
- Situaciones concretas para cada regla.
- Ejemplos de escenarios de transición sencillos que prueban varias reglas a la vez.

Escenarios de ejemplo

- Se han creado varias topologías incorrectas, para comprobar que la aplicación no falla al cargarlas.
- Se han definido cuatro escenarios de ejemplo, para probar diversas situaciones comunes.

Conclusiones

Conclusiones

- La migración de IPv4 a IPv6 es un problema complejo, que debe afrontarse lo antes posible.
- La metodología MENINA ha resultado muy útil para tratar el problema.
- Haskell es un lenguaje potente y flexible, perfectamente capacitado para el desarrollo de aplicaciones complejas.
- Problema de Haskell: curva de aprendizaje y pocas herramientas maduras.

Fin