Trabajo práctico 2: Diseño
Enrutador de tráfico IP

Normativa

Fecha de entrega: Miercoles 11 de noviembre, **de 17 a 18 horas**.

Normas de entrega: Las contenidas en la página web de la materia.

Enunciado

El objetivo de este trabajo práctico consiste en realizar el diseño completo del módulo que se explica con el TAD Router (ver especificación adjunta[1]), así como de todos aquellos módulos necesarios para la tarea.

Las interfaces serán representadas como números naturales (instancias del TAD Nat). Se puede asumir que el conjunto de interfaces de un enrutador son naturales consecutivos comenzando desde 0. Es válido reemplazar el conjunto de interfaces (en el generador y el observador correspondiente) por un número indicando el máximo o la cantidad o algo equivalente.

El diseño debe respetar los siguientes requerimientos de complejidad (siempre en *peor caso*):

- Las operaciones **versiones**, **interfaces**, **eventos**, **tiempoCaída** y **estaCaída** deben realizarse en $\mathcal{O}(1)$.
- La operación **nuevo** debe realizarse en $\mathcal{O}(n)$ donde n es la cantidad de interfaces recibidas como parámetro.
- Las operaciones **agVersión**, **agRegla** y **enrutar** deben cumplir su tarea en $\mathcal{O}(v)$, donde v es la versión a agregar, la versión de la regla a agregar o la versión de la dirección IP solicitada, respectivamente.
- La operación **agEvento** debe cumplir su tarea en $\mathcal{O}(a)$, donde a es la antigüedad del evento a agregar. Esto quiere decir que $a = c - t$ donde c es el timestamp actual (es decir, alguno mayor a todos los existentes en el enrutador) y t es el timestamp del evento a agregar.

Para los cálculos de complejidad no hace falta considerar el tiempo requerido para alojar o desalojar memoria, pero sí para inicializarla.

Recuerden que para cumplir los requisitos de complejidad les puede convenir pedirle a una estructura una operación que usualmente no tiene o que no está especificada en el TAD correspondiente.

Para la confección de este trabajo práctico se deberán tener en cuenta las siguientes pautas:

- El TP debe contener la definición de todos los módulos utilizados, a excepción de los descriptos en el apunte de diseño de la materia y de interfaz, dirIp, reglaDir y eventos, que tampoco hace falta diseñar (dado que son renombres de cosas diseñadas en el apunte, se puede obtener de allí las complejidades de las operaciones).
- La interfaz de cada módulo debe contener, para cada operación: pre y postcondición, complejidad temporal e información sobre aliasing.
- La complejidad de cada operación deberá ser justificada adecuadamente, en función de su algoritmo y de la complejidad de las operaciones de los módulos utilizados. Esto puede hacerse directamente sobre el pseudocódigo o en un texto separado.
- El invariante de representación de las estructuras recursivas (aquellas que involucren punteros) puede escribirse en castellano. Solo puede escribirse en castellano la parte del invariante que involucra punteros, para el resto (de existir) debe usarse lógica.
- Que un módulo se explique con un TAD dado no quiere decir que sea obligatorio diseñarlo usando exactamente las funciones de dicho TAD. Solo es obligatorio diseñar aquellas funciones que precisen utilizar. Para el caso particular del enrutador, es obligatorio incluir las operaciones para las cuales hay una restricción de complejidad en este enunciado. Para los tipos que se usan como resultado de dichas operaciones (secuencia y conjunto), solo es obligatorio que diseñen un módulo análogo con los generadores y observadores básicos.

[1] Esta especificación NO es una solución del TP1. Es sólo una guía para uniformizar los criterios de todos los grupos.

Nota: El 4 de noviembre, a más tardar, los grupos deberán estar en condiciones de presentar informalmente a los docentes un esquema completo del diseño que han concebido. No se requieren los detalles, sino una idea general que incluya un diagrama de módulos y la estrategia, a grandes rasgos, que planean aplicar para poder cumplir los requerimientos de complejidad. Sugerimos fuertemente tenerlo listo anteriormente e ir consultando con los docentes el diseño mientras evoluciona, a fin de llegar correctamente con los tiempos y tener un diseño que permita cumplir con todos los requisitos.

Especificación

Por simplicidad, se utilizan funciones estándar para los Nats que no aparecen en el apunte de TADs básicos (como divisón entera y módulo). También por esa razón, no se especifican las restricciones para validar que las direcciones y reglas sean válidas. Pueden asumirse esas restricciones adicionales para el diseño.

TAD Router

exporta router, generadores, observadores, estaCaída?, tiempoCaída

géneros router

igualdad observacional

$$(\forall r1, r2 : \text{router}) \left(r1 =_{\text{obs}} r2 \iff \begin{array}{c} (versiones(r1) =_{\text{obs}} versiones(r2) \\ \wedge\ interfaces(r1) =_{\text{obs}} interfaces(r2) \\ \wedge\ ((\forall d : dirIp) enrutar(r1, d) =_{\text{obs}} enrutar(r2, d)) \\ \wedge_{\text{L}}\ ((\forall i : interfaz) i \in interfaces(r1) \Rightarrow_{\text{L}} eventos(r1, i) =_{\text{obs}} \\ eventos(r2, i))) \end{array} \right)$$

observadores básicos

versiones	: router	$\longrightarrow$	conj(nat)
interfaces	: router	$\longrightarrow$	conj(interfaz)
enrutar	: router $\times$ DirIp	$\longrightarrow$	respuestaDir
eventos	: router $\times$ interfaz	$\longrightarrow$	secu(evento)

generadores

nuevo	: conj(interfaz)	$\longrightarrow$	router
agVersión	: router r $\times$ versión v	$\longrightarrow$	router $\quad$ v $\notin$ versiones(r)
agRegla	: router r $\times$ reglaDir u	$\longrightarrow$	router $\quad$ versión(dirIp(u)) $\in$ versiones(r) $\wedge$ inter(u) $\in$ interfaces(r)
agEvento	: router r $\times$ evento e	$\longrightarrow$	router $\quad$ ($\neg\exists$ i : interfaz, i $\in$ interfaces(r) $\wedge_{\text{L}}$ existeTimestamp?(eventos(r,i), timestamp(e)))

otras operaciones

tieneRegla?	: router $\times$ dirIp	$\longrightarrow$	bool	
queInterfaz	: router r $\times$ dirIP d	$\longrightarrow$	interfaz	TieneRegla?(r,d)
primerosBitsIguales?	: dirIp d1 $\times$ dirIp d2 $\times$ nat n	$\longrightarrow$	bool	versión(d1) = versión(d2) $\wedge$ versión(d1) $\times$ 8 $\geq$ n
estaCaída?	: router r $\times$ interfaz i	$\longrightarrow$	bool	i $\in$ interfaces(r)
tiempoCaída	: router r $\times$ interfaz i	$\longrightarrow$	nat	i $\in$ interfaces(r)
sumarCaídas	: secu(evento)	$\longrightarrow$	nat	
existeTimestamp?	: secu(evento)	$\longrightarrow$	bool	

axiomas $\forall(r : router, c : conj(interfaz), v, n : nat, re : regla, e : evento, d : dirIp, s : secu(evento))$

versiones(nuevo) $\equiv \emptyset$

versiones(agVersión(r,v)) $\equiv$ ag(v,versiones(r))

versiones(agRegla(r,re)) $\equiv$ versiones(r)

versiones(agEvento(r,e)) $\equiv$ versiones(r)

interfaces(nuevo(c)) $\equiv$ c

interfaces(agVersión(r,v)) $\equiv$ interfaces(r)

interfaces(agRegla(r,re)) $\equiv$ interfaces(r)

interfaces(agEvento(r,e)) $\equiv$ interfaces(r)

enrutar(r,d) $\equiv$ **if** versión(d) $\notin$ versiones(r) **then**
DirecciónNoSoportada
else if $\neg$tieneRegla?(r,d) **then**
interfazDeSalidaNoEncontrada
else if $\neg$estaCaída?(r,queInterfaz(r,d)) **then**
InterfazDeSalidaCaída(queInterfaz(r,d))
else
SalidaPorInterfaz(queInterfaz(r,d))
fi fi fi

eventos(nuevo(c),i) $\equiv$ <>

eventos(agVersión(r,v),i) $\equiv$ eventos(r,i)

eventos(agRegla(r,re),i) $\equiv$ eventos(r,i)

eventos(agEvento(r,e),i) $\equiv$ **if** inter(e) = i **then** ordernarPorTimestamp(e $\bullet$ eventos(r,i)) **else** eventos(r,i) **fi**

tieneRegla?(nuevo(c), d) $\equiv$ false

tieneRegla?(agVersión(r,v), d) $\equiv$ tieneRegla?(r,d)

tieneRegla?(agRegla(r,re),d) $\equiv$ primerosBitsIguales(dirIp(re), d, cantBits(re)) $\vee_L$ tieneRegla?(r,d)

tieneRegla?(agEvento(r,e),d) $\equiv$ tieneRegla?(r,d)

queInterfaz(agVersión(r,v), d) $\equiv$ queInterfaz(r,d)

queInterfaz(agRegla(r,re),d) $\equiv$ **if** primerosBitsIguales(dirIp(re), d, cantBits(re)) **then** inter(re)
else queInterfaz(r,d) **fi**

queInterfaz(agEvento(r,e),d) $\equiv$ queInterfaz(r,d)

primerosBitsIguales?(d1, d2, n) $\equiv$ **if** $n = 0$? **then**
true
else
pasarA8Bits(d1[n $\div$ 8])[n mod 8] = pasarA8Bits(d2[n $\div$ 8])[n mod 8]
$\wedge$ primerosBitsIguales(d1, d2, pred(n))
fi

tiempoCaída(r,i) $\equiv$ sumarCaídas(eventos(r,i))

estaCaída(r,i) $\equiv \neg$vacía?(eventos(r,i)) $\wedge_L$ caída?(ult(eventos(r,i)))

sumarCaídas(s,i) $\equiv$ **if** vacía?(s) $\vee_L$ vacía?(fin(s)) **then** 0 **else**
(**if** caída?(fin(s)) **then** timestamp(prim(fin(s))) - timestamp(prim(s))
else 0 **fi**) + sumarCaídas(fin(s))
fi

existeTimestamp?(s,n) $\equiv \neg$vacía?(s) $\wedge_L$ (n = prim(s) $\vee$ existeTimestamp?(fin(s), n))

Fin TAD