Redes de Computadoras
Letra y Soluciones – 14 de diciembre de 2022

Pregunta 1 (10 puntos)

Suponga que conecta su computador personal a una red pública, en la que el router gateway está en manos de un operador desconocido que puede usarlo para capturar (esnifear) el tráfico que pasa por el. Usted abre su navegador y escribe en la barra de URL: *https://www.fing.edu.uy/docs/reporte.pdf*

Responda a las siguientes preguntas de forma justificada:

a) ¿El operador puede saber a qué servidor se conectó? ¿Cómo?
b) ¿El operador sabe el protocolo que se usó durante la descarga? ¿Cómo?
c) ¿El operador tiene acceso al nombre del documento que descargó? ¿Cómo?
d) ¿El operador tiene acceso al contenido del documento que descargó? ¿Cómo?
e) ¿El operador tiene acceso al tamaño aproximado del documento que descargó? ¿Cómo?

Solución Pregunta 1

a) Si, ya sea interceptando la posible consulta DNS, o el three-way handshake del establecimiento de la conexión TCP. Notar que se está usando HTTPS, por lo que no es posible obtener información del cabezal HTTP debido a que el request viaja encriptado.

b) No de menera determinante, aunque puede suponer que es HTTP si el puerto de la conexión TCP es el 80, así como inferir que está encriptado de alguna forma (por ejemplo HTTPS).

c) No, el nombre del documento es un parámetro de la operación GET y viaja encriptado porque el protocolo es HTTPS.

d) No, como el protocolo es HTTPS el response va encriptado.

e) Puede estimarlo midiendo la cantidad de bytes transmitidos desde el servidor al cliente. Es una estimación porque el tamaño del cabezal HTTP es desconocido.

Pregunta 2 (10 puntos)

a) Dada una red de conmutación de paquetes, mencione y defina los 4 componentes fundamentales del retardo que sufre un paquete que viaja por ella.
b) Indique cuál/es de los retardos puede variar paquete a paquete, por qué y de qué depende.
c) Considere los siguientes 2 escenarios y analice las componentes más relevantes del retardo en ambos escenarios:
    1) más de 2 routers, todos de alta capacidad, ubicados en un datacenter en Miami e interconectados entre sí mediante un hub,
    2) los mismos routers ahora ubicados algunos en un datacenter en Miami y otros en un datacenter en Madrid, interconectados mediante un switch.

Solución Pregunta 2

a) Cuando un paquete viaja de un nodo (host o router) al siguiente nodo (host o router) a lo largo de esa ruta, el paquete sufre varios tipos de retardo en cada uno de los nodos de dicha ruta. Los más importantes de estos retardos son: **el retardo de procesamiento nodal, el retardo de cola, el retardo de transmisión y el retardo de propagación**; todos estos retardos se suman para proporcionar el retardo nodal total.

El tiempo requerido para examinar la cabecera del paquete y determinar dónde hay que enviarlo es parte del retardo de procesamiento. El retardo de procesamiento puede también incluir otros factores, como el tiempo necesario para comprobar los errores de nivel de bit del paquete que se hayan producido al transmitir los bits del paquete desde el nodo situado aguas arriba hasta el siguiente router.

En la cola, el paquete experimenta un retardo de cola mientras espera para ser transmitido a través del enlace. La duración del retardo de cola para un determinado paquete dependerá del número de paquetes que hayan llegado antes a la cola y que estén esperando para ser transmitidos por el enlace.

Sea la longitud del paquete igual a L bits y la velocidad de transmisión del enlace del router A hasta el router B igual a R bits/segundo. El retardo de transmisión será igual a L/R. Este es el tiempo necesario para introducir (es decir, transmitir) todos los bits del paquete en el enlace.

Una vez que un bit ha entrado en el enlace, tiene que propagarse hasta el próximo router. El tiempo necesario para propagarse desde el principio del enlace hasta el router B es el retardo de propagación.
El retardo de propagación es igual a la distancia entre dos routers dividida por la velocidad de propagación. Es decir, el retardo de propagación es igual a d/s, donde d es la distancia entre un router y el router siguiente, y s es la velocidad de propagación del enlace.

b) El retardo de cola.
Por ejemplo, si llegan 10 paquetes a una cola vacía al mismo tiempo, el primer paquete transmitido no sufrirá retardo de cola, mientras que el último paquete transmitido sufrirá un retardo de cola relativamente largo (mientras espera a que los restantes nueve paquetes sean transmitidos).
Depende de la velocidad a la que llega el tráfico a la cola, de la velocidad de transmisión del enlace y de la naturaleza del tráfico entrante, es decir, de si el tráfico llega periódicamente o a ráfagas.

Si el tamaño de los paquetes varía, se puede considerar que el retardo de transmisión varía entre los paquetes de distinto tamaño. Sin embargo, esta variación es determinista y solo depende del tamaño del paquete, no del estado “desconocido” de la red.

c) Routers de alta capacidad: el retardo de procesamiento se considera despreciable.
Al ser en el mismo Datacenter, el retardo de propagación es despreciable.

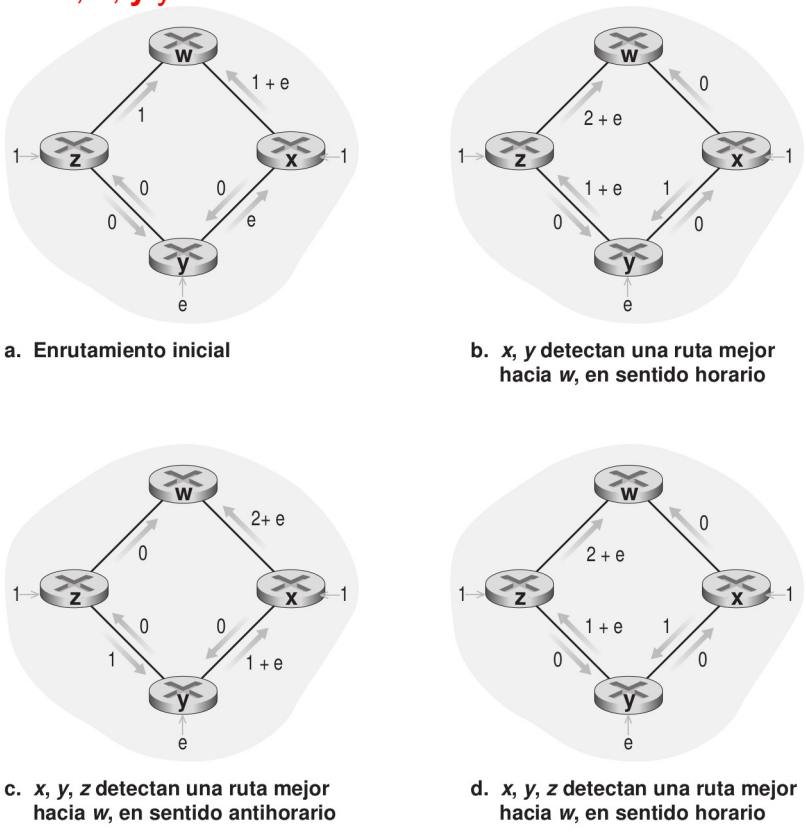
Caso Hub: el retardo de cola puede ser significativo si el acceso al medio mediante CSMA/CD comienza a mostrar señales de colisión.

Caso Switch: el retardo de cola deja de ser significativo (ya no hay colisiones). Comienza a ser significativo el retardo de propagación en el caso que se comunican routers ubicados en datacenter distintos.

Pregunta 3 (10 puntos)
Considere un algoritmo de enrutamiento de estado de enlace, explique mediante un ejemplo sencillo por qué es posible generar oscilaciones en el enrutamiento cuando se utilizan costos dinámicos para los enlaces, proporcionales al tráfico cursado por ellos.

Solución Pregunta 3

La Figura 5.5 (del libro del curso 7a edición) muestra una topología de red simple donde el costo de cada enlace es igual a la carga transportada por el enlace. En este ejemplo, los costos de los enlaces no son simétricos; es decir, **c(u,v)** es igual a **c(v,u)** solo si la carga transportada en ambas direcciones del enlace **(u,v)** es la misma. En este ejemplo, el nodo **z** origina una unidad de tráfico destinada a **w**, el nodo **x** también origina una unidad de tráfico destinada a **w**, y el nodo **y** inyecta una cantidad de tráfico igual a **e**, también destinado a **w**.
El enrutamiento inicial se muestra en la Figura 5.5(a), correspondiendo los costos de los enlaces a la cantidad de tráfico transportado.
Cuando se vuelve a ejecutar el algoritmo LS, el nodo **y** determina, basándose en los costos de enlace mostrados en la Figura 5.5(a), que la ruta en sentido horario a **w** tiene un costo de **1**, mientras que la ruta en sentido antihorario a **w** (que era la que había estado utilizando) tiene un coste de **1+e**. Por tanto, la ruta de coste mínimo de **y** a **w** ahora va en sentido horario. De forma similar, **x** determina que ahora su nueva ruta de costo mínimo a **w** va en sentido horario, dando como resultado los costos mostrados en la Figura 5.5(b). Cuando el algoritmo LS se ejecuta otra vez, los nodos **x**, **y** y **z** detectan una ruta de coste cero a **w** en el sentido antihorario y todos ellos envían su tráfico a las rutas en sentido antihorario. La siguiente vez que se ejecuta el algoritmo LS, **x**, **y** y **z** enrutan su tráfico a las rutas en sentido horario.



a. Enrutamiento inicial

b. *x*, *y* detectan una ruta mejor hacia *w*, en sentido horario

c. *x*, *y*, *z* detectan una ruta mejor hacia *w*, en sentido antihorario

d. *x*, *y*, *z* detectan una ruta mejor hacia *w*, en sentido horario

Figura 5.5 ♦ Oscilaciones con enrutamiento sensible a la congestión.

Pregunta 4 (10 puntos)
a) Discuta la eficiencia de los protocolos de transporte de tipo “parada y espera” (stop & wait), en comparación con protocolos “en cadena” (pipeline).
b) Discuta ventajas y desventajas de los protocolos de tipo “retroceder N” (Go Back N - GBN) y los de “repetición selectiva” (Selective Repeat).

Solución Pregunta 4

a) En los protocolos stop&wait se debe esperar el ACK del segmento actual para enviar un nuevo segmento. En los casos con grandes retardos de propagación (RTTs grandes), típico en enlaces de largas distancias (miles de km), este comportamiento es inaceptable, ya que implica que el enlace está la mayor parte del tiempo vacío; en efecto, si despreciamos el tamaño (y por tanto el tiempo de trasmisión) de los ACKs, y consideramos solo los retardos de propagación, la tasa de utilización de un protocolo stop&wait se puede aproximar por Uemisor=L/R/(RTT+L/R), es decir, el tiempo de trasmisión de un segmento dividido por el tiempo total que hay que esperar para volver a trasmitir un nuevo segmento. Por ejemplo para los valores RTT=30ms, L=1000bytes (8000 bits), R=1Gbps, la tasa de utilización es de 0,008ms/30,008ms=0,00027, es decir 0,27%, inadmisible.
Los protocolos en pipeline, en cambio, admiten tener muchos segmentos “en viajes” sin que hayan llegado los ACKs correspondientes, lo cual permite llenar el canal y por lo tanto aumentar la tasa de utilización.

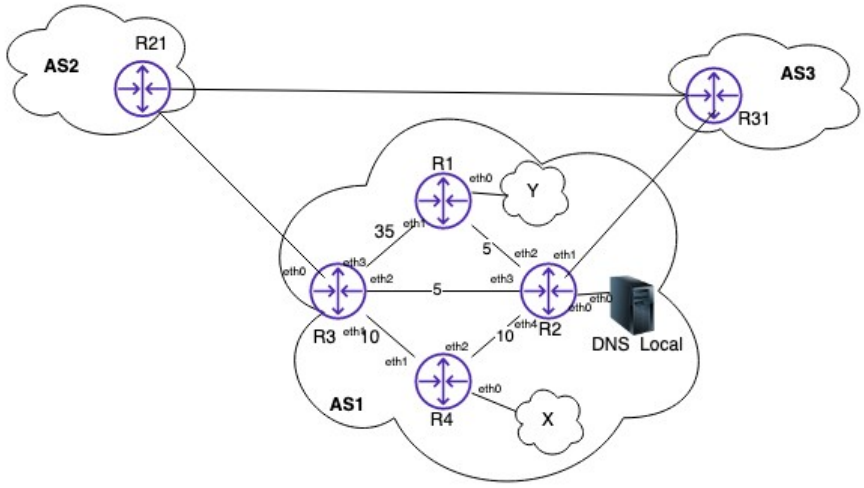
b) Los protocolos de tipo Go Back N funcionan de modo que si un segmento con número de secuencia n se recibe correctamente y en orden (es decir, los últimos datos entregados a la capa superior proceden de un segmento con el número de secuencia n-1), el receptor envía un ACK para el segmento n y entrega la parte de los datos del segmento a la capa superior. En todos los restantes casos, el receptor descarta el segmento y reenvía un mensaje ACK para el segmento recibido en orden más recientemente, es decir, descarta los paquetes que no están en orden. La ventaja de este método es la simplicidad del almacenamiento en el buffer del receptor (el receptor no necesita almacenar en el buffer ninguno de los segmentos entregados desordenados), sin embargo, si los segmentos no llegan en orden, se debe descartar toda la “ventana” de segmentos recibidos, bajando la eficiencia del protocolo. En efecto, un único segmento erróneo podría hacer que el protocolo GBN retransmitiera una gran cantidad de segmentos, muchos de ellos de forma innecesaria.

En cambio, los protocolos de repetición selectiva evitan las retransmisiones innecesarias haciendo que el emisor únicamente retransmita aquellos segmentos que se sospeche que llegaron al receptor con error (es decir, que se perdieron o estaban corrompidos). Esta retransmisión individualizada y necesaria requerirá que el receptor confirme individualmente qué paquetes ha recibido correctamente. Esta característica mejora la eficiencia del uso del canal, pero la implementación es más compleja que GBN.

Problema 1 (30 puntos)

Se cuenta con el esquema de red presentado en la figura, donde los routers frontera mantienen sesiones eBGP activas con sus pares de otro sistema autónomo y sesiones iBGP con los routers internos al AS. Los sistemas autónomos AS1 y AS3 son ISPs de acceso (redes terminales).
El AS3 es dueño del prefijo 19.0.0.0/16, el AS2 del 18.0.0.0/18 y el AS1 del 17.0.0.0/22.

Se sabe que el AS1 internamente ejecuta un algoritmo de tipo vector de distancias mientras que el AS2 y el AS3 uno de tipo estado de enlaces. El AS1 tiene como clientes a las redes X e Y, y aloja un servidor DNS Local. Los valores de los enlaces del AS1 son los costos asociados a los mismos.



a) Numere todas las interfaces del AS1 que correspondan y asigne prefijos a las redes X e Y sabiendo que la red X requiere de 80 direcciones IP para alojar servicios públicos y la red Y requiere de 126 direcciones IP para sus servidores web.

b) Muestre las **rutas** BGP que anuncian y reciben los sistemas autónomos de la figura. Justifique su respuesta.

c) Escriba la tabla de forwarding (reenvío) del DNS Local y del router R4 asumiendo que los algoritmos de routing ya convergieron. Los costos de los enlaces son los indicados en la figura. Justifique brevemente cada entrada.

d) Suponga ahora que el enlace entre R1 y R2 (dentro del AS1) se cae debido a una falla en una de las interfaces.

1. Muestre las tablas del algoritmo de vector distancia en R1, R2 y R3 antes de la caída del enlace.
2. Muestre los vectores de distancia intercambiados entre R2 y R3 hasta alcanzar la convergencia. Justifique los pasos seguidos.

e) Mencione alguna técnica que podría utilizar para reducir las iteraciones luego de la falla. Muestre las tablas del algoritmo de vector distancia en R1, R2 y R3 cuando el algoritmo converge si se aplica esta técnica. No considere la caída del enlace para este caso.

Solución Problema 1

a) El AS1 es dueño del prefijo 17.0.0.0/22 y debe proveer un rango IP público para sus clientes (X e Y) y para su servidor DNS Local. Para las subredes internas al backbone se utilizará direccionamiento privado con el objetivo de preservar las direcciones IP públicas para los clientes.

X necesita 80 IP públicas + red + broadcas + router = 83 → 7 bits para hosts → el prefijo más ajustado es un /25
Y necesita 126 IP públicas + red + broadcas + router = 129→ 8 bits para hosts → el prefijo más ajustado es un /24
Para la sub red DNS-R2: 2 ips.
Una asignación eficiente partiendo de 17.0.0.0/22 es tomar 17.0.0.0/24 para Y, 17.0.1.0/25 para X, y 17.0.1.128/30 para R2-DNS, es decir, tomando direcciones de una “rama” del árbol de particiones (dejando un /23 disponible).
IP s: DNS-R2: .129 para el lado R2 .130 para el DNS.

Privadas:
R1-R2: 10.0.0.0/30 (Ips: .1 para el lado R1 .2 para el lado R2)
R2-R4: 10.0.1.0/30 (Ips: .1 para el lado R2 .2 para el lado R4)
R4-R3: 10.0.2.0/30 (Ips: .1 para el lado R4 .2 para el lado R3)
R2-R3: 10.0.3.0/30 (Ips: .1 para el lado R2 .2 para el lado R3)

b)
Como AS1 y AS3 son ISPs de acceso (redes terminales) el tránsito en ellos deben ser paquetes salientes de o destinados a ellos. Las rutas BGP están compuestas por un prefijo + atributos (AS-PATH - next-hop).

AS1 → AS3
17.0.0.0/22 – AS1 - R2_eth1
AS1 → AS2
17.0.0.0/22 – AS1 - R3_eth0

AS3 → AS1
19.0.0.0/16 – AS3 - R31_if0
AS3 → AS2
19.0.0.0/16 – AS3 - R31_if1

AS2 → AS1
19.0.0.0/16 – AS2, AS3 - R21_if0
17.0.0.0/22 – AS2, AS1 - R21_if0
18.0.0.0/18 – AS2 - R21_if0

AS2 → AS3
19.0.0.0/16 – AS2, AS3 - R21_if1
17.0.0.0/22 – AS2, AS1 - R21_if1
18.0.0.0/18 - AS2 - R21_if1

c)
Asigno R2_eth0 17.0.1.129
DNS Local 17.0.1.130

Tabla DNS Local
Prefijo – NH – if salida
17.0.1.128/30 – DC - eth0
0.0.0.0/0 - 17.0.1.129 – eth0

Tabla R4
Prefijo – NH – if salida
17.0.0.0/24 – IP_R2_eth4 - eth2
17.0.1.0/25 – DC – eth0
17.0.1.128/30 – IP_R2_eth4 - eth2
10.0.0.0/30 – IP_R2_eth4 - eth2
10.0.1.0/30 – DC - eth2
10.0.2.0/30 – DC – eth1
10.0.3.0/30 - IP_R3_eth1 – eth1
19.0.0.0/16 - IP_R2_eth4 – eth2 (iBGP)
18.0.0.0/18 - IP_R3_eth1 – eth1 (iBGP desempata por AS-PATH)

d)
i)

Tabla de R1

	R1	R2	R3	R4
R1	0	5	10	15
R2	5	0	5	10
R3	10	5	0	10

Tabla de R2

	R1	R2	R3	R4
R1	0	5	10	15
R2	5	0	5	10
R3	10	5	0	10
R4	15	10	10	0

Tabla de R3

	R1	R2	R3	R4
R1	0	5	10	15
R2	5	0	5	10
R3	10	5	0	10
R4	15	10	10	0

ii)

Cae el enlace R1, R2. Esto hace que desde ellos detectan que llegan al otro “directo” con costo infinito. Por lo tanto, utilizan Bellman-Ford para recalcular las rutas, quedando de la siguiente forma:

Tabla de R1

	R1	R2	R3	R4
R1	0	**40**	**35**	**45**
R2	5	0	5	10
R3	10	5	0	10

Tabla de R2

	R1	R2	R3	R4
R1	0	5	10	15
R2	**15**	0	5	10
R3	10	5	0	10
R4	15	10	10	0

Al haber un cambio, R1 y R2 envían sus vectores distancias a sus vecinos. A partir de este momento R1 y R2 dejan de ser vecinos y ya no reciben mas los DV entre ellos.

1) En particular, R2 envía a R3 el siguiente vector:

R2	**15**	0	5	10

R3 recalcula su vector a partir de la nueva información

Tabla de R3

	R1	R2	R3	R4
R1	0	40	35	45
R2	**15**	0	5	10
R3	**20**	5	0	10
R4	15	10	10	0

2) R3 envía el nuevo vector a R2:

R3	**20**	5	0	10

R2 recalcula su vector a partir de la nueva información

Tabla de R2

	R1	**R2**	**R3**	**R4**
R2	**25**	0	5	10
R3	**20**	5	0	10
R4	25	10	10	0

3) Se lo envía a R3

R2	**25**	0	5	10

Tabla de R3

	R1	**R2**	**R3**	**R4**
R1	0	40	35	45
R2	**25**	0	5	10
R3	**30**	5	0	10
R4	25	10	10	0

4) Se lo envía a R2

R3	**30**	5	0	10

Tabla de R2

	R1	**R2**	**R3**	**R4**
R2	**35**	0	5	10
R3	**30**	5	0	10
R4	35	10	10	0

5) Se lo envía a R3

R2	**35**	0	5	10

Tabla de R3

	R1	**R2**	**R3**	**R4**
R1	0	40	35	45
R2	**35**	0	5	10
R3	**35 (directo por R1)**	5	0	10
R4	25	10	10	0

6) Se lo envía a R2

R3	**35**	5	0	10

Tabla de R2

	R1	**R2**	**R3**	**R4**
R2	**40**	0	5	10
R3	**35**	5	0	10
R4	35	10	10	0

7) Se lo envía a R3

R2	**40**	5	0	10

Tabla de R3

	R1	R2	R3	R4
R1	0	40	35	45
R2	**40**	0	5	10
R3	35	5	0	10
R4	35	10	10	0

No altera su DV por lo tanto se estabiliza.

e)
Una técnica a utilizar es la reversa envenenada. En este caso, un nodo cuando envía su DV a un vecino, cambia las distancias a infinito de los destinos cuyos caminos pasan por ese vecino.

Tabla de R1

	R1	R2	R3	R4
R1	0	5 (por R2)	10 (por R2)	15 (por R2)
R2	infinito	0	5	10
R3	10	5	0	10

Tabla de R2

	R1	R2	R3	R4
R1	0	Infinito	**Infinito**	**infinito**
R2	5 (por R1)	0	5 (por R3)	10 (por R4)
R3	**Infinito**	Infinito	0	10
R4	**Infinito**	Infinito	10	0

Tabla de R3

	R1	R2	R3	R4
R1	0	5	10	15
R2	5	0	Infinito	10
R3	10 (por R2)	5 (por R2)	0	10 (por R4)
R4	15	10	Infinito	0

Tabla de R4

	R1	R2	R3	R4
R1	0	5	10	15
R2	5	0	5	Infinito
R3	10	5	0	Infinito
R4	15 (por R2)	10 (por R2)	10 (por R3)	0

Problema 2 (30 puntos)

Considere un cliente HTTP cuyo objetivo es, dada la URL de una página HTML, guardar recursivamente a disco todas la páginas enlazadas. Cada página puede contener varios *links*, y el proceso termina cuando no hay más páginas para descargar. El cliente usa HTTP1.0, y descarga una página por vez (sin descargas simultáneas).

Un enlace en una página HTML tiene el formato:
`<a href="example.com/path/file">un texto</a>`

Cuando un navegador detecta ese texto en un objeto de tipo "text/html", escribe en pantalla "un texto" donde este apunta a `example.com/path/file`

Para escribir en archivos dispone de la llamada:
appendfile (filename, s)
Si el archivo *filename* no existe lo crea. El string *s* es agregado al final del contenido del archivo *filename*.

Se pide:

Implemente el programa cliente en un lenguaje de alto nivel usando la API de sockets del curso. Además dispone de APIs auxiliares para parsear strings, estructuras de datos, etc.

Solución Problema 2

```
urls = new queue()   -- urls pendientes de descarga
urls.push( arg[1] )  -- el parametro es la página incial

function download(url)
  -- separamos el protocolo, servidor y path
  proto  = string.sub(url, 1, string.pos('://')-1)
  url    = string.sub(url, string.pos('://')+3, string.len(url))
  server = string.sub(url, 1, string.pos('/')-1)
  path   = string.sub(url, string.pos('/'), string.len(url))

  master = socket.tcp()
  client = master.connect(server, 80) -- Según la API del curso, el
connect acepta un hostname como parámetro

  -- armo mensaje GET
  getstring = 'GET '..url..' HTTP1.0\r\nHost: '..server..'\r\n'

  repeat --envío el comando por la conexión
    getstring, err = client.send(getstring)
    if (err == 'closed') then
      client.close()
      return
    end
  until getstring == ''

  buff = ''
  type = ''
  repeat
    fragment, err = client.receive();
    if fragment then
      buff = buff + fragment
    end

    if not type then
      --expresion regular para leer tipo
      type = string.match(buff, "content-type=(.*)\n")
      if type and type!='text/html' then
        client:close()
        return
      end
```

```
    end
  until err=='closed'

  -- saltearse header
  buff = string.sub(string.pos(buff, '\r\n\r\n'+4), string.len(buff))

  -- una expresion regular adecuada para extraer los links
  for link in string.match(buff, "<a href='(.*))'>") do
    url.push(link)
  end
  appendfile (buildfilename(url), buff)
end


repeat
  url = urls.pop()
  download(url)
until urls.empty()
```