C2 REDES

Nombre: Patricio Espinoza A.

P1. DNS

Pruebe el dominio scout.cl y haga un diagnóstico de su configuración. Explique los pasos que siguió, los resultados que obtuvo y, finalmente, liste todos los errores encontrados.

Pasos:

1. Ir a la página www.diggui.com
2. Escribir scout.cl en Hostname or IP address
3. Seleccionar los distintos Record Types

Record Type: A

A:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1630
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      A

;; ANSWER SECTION:
scout.cl.               120     IN      A       45.7.229.184

;; Query time: 129 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 15:30:07 UTC 2025
;; MSG SIZE  rcvd: 53
```

Obs: Se detectó una respuesta válida para la dirección

Record Type: NS

NS:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl NS
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26960
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      NS

;; ANSWER SECTION:
scout.cl.               120     IN      NS      ns1.scout.cl.
scout.cl.               120     IN      NS      server.jpfcine.cl.
scout.cl.               120     IN      NS      secundario.nic.cl.

;; Query time: 223 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 15:32:34 UTC 2025
;; MSG SIZE  rcvd: 113
```

Obs: Muestra múltiples servidores autoritativos, indicando presencia de delegación.

Record Type: SOA

SOA:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl SOA
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41254
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      SOA

;; ANSWER SECTION:
scout.cl.               120     IN      SOA     server.scout.cl. admin.scout.cl. 2025052404 180 300 300 360

;; Query time: 129 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 15:33:37 UTC 2025
;; MSG SIZE  rcvd: 86
```

Obs: Se detectó que el servidor primario es server.scout.cl y que el administrativo es admin.scout.cl

Record Type: MX

MX:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl MX
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56058
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      MX

;; AUTHORITY SECTION:
scout.cl.               120     IN      SOA     server.scout.cl. admin.scout.cl. 2025052404 180 300 300 360

;; Query time: 136 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 15:34:33 UTC 2025
;; MSG SIZE  rcvd: 86
```

No se obtuvo ningún registro MX, lo que implica que el servidor no posee una configuración de correo para recibir o enviarlos directamente.

Record Type: DNSSEC

Como no está explicito, verificaremos manualmente DNSkey, DS, RRSIG, NSEC/NSEC3

DNSKEY:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl DNSKEY
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55251
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      DNSKEY

;; AUTHORITY SECTION:
scout.cl.               120     IN      SOA     server.scout.cl. admin.scout.cl. 2025052404 180 300 300 360

;; Query time: 135 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 19:40:35 UTC 2025
;; MSG SIZE  rcvd: 86
```

DS:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl DS
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21532
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      DS

;; AUTHORITY SECTION:
cl.                     900     IN      SOA     a.nic.cl. dnsadmin.nic.cl. 2025052731 1200 300 2592000 900

;; Query time: 9 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 19:42:21 UTC 2025
;; MSG SIZE  rcvd: 88
```

RRSIG:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl RRSIG
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55583
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      RRSIG

;; AUTHORITY SECTION:
scout.cl.               120     IN      SOA     server.scout.cl. admin.scout.cl. 2025052404 180 300 300 360

;; Query time: 129 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 19:42:58 UTC 2025
;; MSG SIZE  rcvd: 86
```

NSEC:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl NSEC
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36471
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      NSEC

;; AUTHORITY SECTION:
scout.cl.               120     IN      SOA     server.scout.cl. admin.scout.cl. 2025052404 180 300 300 360

;; Query time: 133 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 19:43:33 UTC 2025
;; MSG SIZE  rcvd: 86
```

NSEC3:scout.cl@8.8.8.8

```
; <<>> DiG diggui.com <<>> @8.8.8.8 scout.cl NSEC3
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10062
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;scout.cl.                      IN      NSEC3

;; AUTHORITY SECTION:
scout.cl.               120     IN      SOA     server.scout.cl. admin.scout.cl. 2025052404 180 300 300 360

;; Query time: 136 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue May 27 19:44:04 UTC 2025
;; MSG SIZE  rcvd: 86
```

La falta de cada registro implica que:

- ➔ DNSKEY: No hay claves públicas DNSSEC configuradas
- ➔ DS: El dominio padre (.cl) no tiene identificado a scout.cl como que use DNSKEY, y por tanto no es posible hacer una validación ascendente.
- ➔ RRSIG: Indicó que no hay registros digitales firmados
- ➔ DNSEC: Muestra que no hay una implementación para la denegación de existencia
- ➔ DNSEC3: No hay denegación de existencia para registros mediante hashing.

Esto indica que además de faltar MX tampoco está habilitado DNSSEC ya que el dominio no tiene en sus registros DNSKEY, DS, RRSIG, NSEC/NSEC3. Una vulnerabilidad de esto última es que el dominio queda expuesto a ataques, como, por ejemplo, cache poisoning o spoofing.

P2. Protocolos Clásicos

1. ¿Funcionaría sin generar errores en cualquier escenario en que un Selective Repeat clásico funciona?

El Selective Repeat clásico funciona cuando:

- El receptor tiene una ventana mayor o igual a N, tal que pueda recibir y almacenar cualquier paquete en el rango.
- El receptor debe ser capaz de distinguir los paquetes (nuevos y duplicados), y para ello el tamaño de la ventana de recepción debe ser menor o igual a N/2.

Ingeniero 1: Tamaño ventana enviador = N, tamaño ventana receptor = N/2, BDP = N

Como la ventana del receptor es más pequeña que la del enviador, es posible que este último rechace paquetes que lleguen fuera del rango de su ventana actual, lo cual afectaría a la pérdida de paquetes por una implementación que no sigue los requerimientos del Selective Repeat clásico. Por tanto, su funcionamiento no estaría libre de errores.

Ingeniero 2: Tamaño ventana enviador = N/2, tamaño ventana receptor = N, BDP = N

El rango de la ventana del receptor es mucho más grande que el del enviador, esto concuerda con lo requerido por el Selective Repeat Clásico, y por tanto, funcionará sin errores.

2. Si no hay pérdidas ni desorden de paquetes: ¿Es mejor que Go-Back-N? ¿Es peor que Selective Repeat clásico?

Ingeniero 1:

El emisor puede enviar N paquetes, sin embargo, el receptor solo puede aceptar N/2 paquetes antes de que la ventana se llene. Aún si no hay pérdidas ni desorden de paquetes, como el emisor se debe detener hasta que la haya espacio en la ventana (cuando reciba un ACK) la implementación no utilizará todo el BDP debido a la formación de un “cuello de botella”.

Esto hace que sea mejor que Go-Back-N ya que puede ir confirmando los paquetes individualmente sin la necesidad de esperar por uno perdido, pero peor que el Selective Repeat clásico ya que se limita el rendimiento debido a los tamaños dispares de las ventanas.

Ingeniero 2:

En este caso, como se limita la ventana del enviador, el receptor recibe y responde sin saturaciones en su ventana, sin embargo, esto representa un problema a su manera ya que no se aprovecha totalmente el BDP = N y por tanto es ineficiente.

Al comparar con Go-Back-N tenemos que esta implementación es más eficiente pues se mantiene el sistema de ACK individuales por paquete (que solventa envíos innecesarios, aunque no aplicaría para este caso para el Selective Repeat personalizado, pero sí para el GBN).

Por otra parte, es peor que un Selective Repeat clásico pues no llega al BDP óptimo ya que se limitó la ventana del receptor arbitrariamente.

3. Si hay pérdidas de paquetes, pero no desorden: ¿Es mejor que Go Back-N? ¿Es peor que Selective Repeat clásico?

El protocolo Go Back-N funciona tal que, al perder un paquete, el receptor descarta los paquetes siguientes, retransmitiendo todos los paquetes a partir del que se perdió. Es decir, basta un paquete perdido para que ya se vuelva ineficiente.

Por otro lado, el protocolo Selective Repeat clásico retransmite solo el paquete que se perdió, y gracias al uso de ACKS individuales el protocolo puede seguir aceptando y almacenando otros paquetes fuera de orden.

Teniendo esto en cuenta, para el **ingeniero 1** sucederá que habrán más paquetes siendo enviados, lo que llevará a descartar los sobrantes una vez se llene el buffer de la ventana del receptor. El manejo para la pérdida de paquetes resultará mas eficiente que Go Back-N, sin embargo, no será mejor al Selective Repeat clásico debido a que perderá más paquetes a causa del limite en la ventana del receptor.

Para el **ingeniero 2** sucederá que habrán menos paquetes enviados y una ventana del receptor no saturada. Si se pierde un paquete, solo se deberá retransmitir este, y la ventana del receptor no saturada no generará paquetes retransmitidos extras. Por tanto, será más eficiente que un Go Back-N. Por otro lado, la única razón por la que no sería igual de óptimo que un Selective Repeat clásico es porque no alcanza el BDP óptimo de N, ya que la retransmisión de paquete será notablemente mejor que para la propuesta del ingeniero 1.

4. Si hay desorden de paquetes, pero no pérdidas: ¿Es mejor que Go Back-N? ¿Es peor que Selective Repeat clásico?

De acuerdo con la definición de Go Back-N, este no tolera el desorden de paquetes, si uno llega desordenado será descartado y continuará con el siguiente en la secuencia esperada. Por otro lado, Selective Repeat clásico si que tolera el desorden por su sistema de buffers.

Para la propuesta del ingeniero 1 llegarán mas paquetes de los que se pueden recibir, perdiéndose paquetes en desorden, lo que simularía parcialmente un Go Back-N con la única diferencia de que esta propuesta de protocolo tolera más el desorden y por tanto no se caería como el GBN clásico en este caso. Su comportamiento respecto al Selective Repeat clásico será peor ya que el clásico no se vería tan afectado por el desorden al tener un buffer mayor donde guardar los paquetes desordenados, los almacenaría y enviaría en el momento adecuado. El propuesto por el ingeniero podría tener el problema de que hay menos ventana de buffers y un desorden muy aleatorio en donde guardarlos no resulte tan eficiente (por ejemplo, guardar los paquetes 91, 92, ..., 100 y solo tener 2 buffers en la ventana con los paquetes 1 y 2. En este caso los debería guardar hasta que sea el momento en donde puedan enviarlos y sería un consumo de recursos muy ineficiente. El SR clásico al tener una ventana mayor será más eficiente en dicha situación.

Para la propuesta del ingeniero 2, aunque se tiene que el emisor tiene una ventana N/2 y el receptor de N (con BDP=N), se repite el hecho de que GBN no puede manejar desorden, y por tanto el protocolo propuesto será más efectivo en este contexto. Sin embargo, al compararlo con el SR clásico la única diferencia será la eficiencia del canal, es decir, el BDP no será óptimo. Esto sucede ya que el emisor enviará N/2 paquetes desordenados, pero el receptor tendrá una ventana de N buffers en donde sobrará el espacio para almacenarlos y gestionarlos adecuadamente.

5. En resumen, ¿este protocolo es efectivamente mejor que Go-Back-N?

Si, ambos protocolos propuestos resultan más eficientes que Go-Back-N, la única diferencia entre ambos es el desempeño dado un contexto en donde se pierden más paquetes o uno donde hay mayor desorden, mostrando la importancia entre la relación de las ventanas de emisión y recepción. De todas formas, ambos manejan el desorden que no es posible con GBN, reenvíos más eficientes ante pérdidas y un BDP mayor.

6. Finalmente, si el BDP del enlace es N/2, ¿cambia alguna respuesta?

Las respuestas cambiarían en el sentido de que ahora el protocolo del ingeniero 2 si sería óptimo, en el resto del análisis se mantendrían igual, donde el protocolo 1 puede mantenerse ineficiente de todas formas.

P3. Congestión

1. ¿Está correcto ese análisis?

El BDP se obtiene al multiplicar el ancho de banda con el RTT: BDP = BW * RTT, esta métrica nos permite saber cuantos datos es posible enviar sin saturar un canal. En una situación donde ocurre congestión el RTT aumenta ya que los buffers se van ocupando, lo que implica que una vez llenados existirá un tiempo de cola donde el canal estará saturado. Por consecuencia uno esperaría que el BDP, dependiente de RTT, aumente también, sin embargo, el BDP disminuye a causa de las retransmisiones necesarias y de los paquetes que son descartados debido al buffer lleno.

Por tanto, si la ventana de congestión se aumenta, se introducirán aun más datos a un canal ya saturado provocando que se sature aún más. En definitiva, el análisis es incorrecto y lo que se debe de hacer es disminuir la ventana de congestión para evitar que la red colapse.

2. ¿Qué pasaría si agrandamos la ventana frente a la congestión?

Si se agranda la ventana frente a una congestión ocurrirá que esta empeorará aún más y la red este en peligro de colapsar. Esto ya que agrandarla implica que entraran más datos en la red, generando un mayor RTT, una mayor pérdida de paquetes y un ciclo de colas más largas con retransmisiones innecesarias.

3. ¿Podríamos considerar que la pérdida de paquetes es equivalente a una disminución del ancho de banda del enlace virtual?

Al ver el ancho de banda para TCP tenemos que una pérdida de paquetes representa que hay un nivel de congestión debido a que la red se ha saturado, y esto efectivamente representa una disminución en el ancho de banda efectivo en el flujo.

Los paquetes perdidos denotarán congestión, la cual se traducirá en una disminución de la ventana de congestión (disminuyendo la cantidad de datos que pueden pasar), y por consiguiente el ancho de banda del enlace virtual.

4. Uno podría decir que hay un BDP variable asociado al enlace virtual. Cuando hay pérdidas, cambia el BDP. ¿Puede que la ventana de congestión busque aproximarse al nuevo BDP?

Si es correcto decir que la ventana de congestión busca aproximarse al BDP variable, esto ya que la ventana de TCP se adapta continuamente de acuerdo a las condiciones de la red, estas adaptaciones buscan aproximarse al BDP efectivo del enlace virtual, que será aquel que permita un flujo correcto.

P4. Simulador

4.1 ACKs y CACKs

Un ACK (Acknowledgment): Permite un reconocimiento parcial al indicar el número del último paquete recibido (en orden).

Un CACKs (Cumulative ACK): Corresponde al último paquete recibido correctamente en orden. Indica que todos los frames previos fueron recibidos. Reduce la pérdida de tiempo y de ancho de banda al solo necesitar la última confirmación.

La propuesta mencionada implicaría que, si se esta recibiendo el paquete n°10 fuera de orden (faltan algunos previos), se debe enviar:

- ACK parcial = 10
- Último CACK = 5

¿En que situaciones sirve?

Sería útil cuando la red tiene un envío de paquetes extremadamente desordenado, permitiéndonos observar que efectivamente haya un progreso en el flujo. También podría ser útil para observar que paquetes son los que se están retransmitiendo o que se pierdan reiteradas veces, dando la posibilidad de optimizar la retransmisión exacta de dichos paquetes.

Dentro de los protocolos vistos, esta combinación resulta particularmente útil para el Selective Repeat, ya que en él se permiten paquetes desordenados, podría resultar óptimo para las retransmisiones, pero a su vez más pesado en tamaño, lo cual dependiendo del volumen de datos es algo a considerar como una desventaja.

Para protocolos como Go-Back-N no es útil ya que este requiere estrictamente de un orden de los paquetes para que el flujo continue correctamente. En este caso tener tanto ACK como CACK seria redundante.

¿Podría dar una ventaja de eficiencia importante?

Puede dar una ventaja de eficiencia solo en la recuperación de paquetes perdidos para su retransmisión en un contexto donde esto ocurre frecuentemente por paquetes desordenados que requieren tener claridad en su orden. En otra situación donde la red es estable el costo añadido de tener ACK y CACK combinados implicaría que el tamaño añadido no es rentable.

¿Repite información que ya se conocía?

Si la red es estable y no hay desorden ni pérdida de paquetes, entonces el ACK si contendrá información implícita del CACK, y por tanto serán redundantes. En caso contrario (como el ejemplo inicial) se puede obtener información adicional mediante CACK que no era accesible solo con ACK.

2.2 Retransmisión en Go-Back-N

Este caso se pudo replicar cuando había una probabilidad de pérdida alta del paquete y a su vez el timeout era bajo, lo que provocaba que el paquete en camino del emisor al receptor estuviera en el trayecto y en poco tiempo ya se hubiese generado otro paquete retransmitido (aunque no se hubiese detectado una pérdida del paquete).

Algo a considerar de Go-Back-N es que no utiliza ACKs individuales para cada paquete, sino que implementa ACKs acumulativos. Lo que significa que si no se recibió un ACK a tiempo no hay conocimiento sobre lo que sucedió con el paquete, y por tanto se generará el retransmitido.

No sería un error del protocolo, sino que de la implementación dados los parámetros con los que se define que trabaje. Esto es similar a como con Selective Repeat antes no se alcanzaba el ancho de banda óptimo debido a las condiciones forzadas al protocolo para la prueba, y por tanto, con parámetros más flexibles el protocolo no debería de mostrar este comportamiento.

La solución para este problema es ajustar los timeouts para una mayor flexibilidad, incluso se podría proponer que sean adaptativos de acuerdo con el comportamiento del RTT. Por ejemplo, que se actualice basándose en la información de aquellos paquetes recibidos correctamente. También se puede sugerir observar los cambios en la ventana para deducir si el paquete sigue en tránsito y que por tanto no sea necesario retransmitirlo aún, aunque esto podría no resultar del todo eficaz en contextos no estables o desordenados, junto al hecho de que da más flexibilidad, pero añade incertidumbre respecto a cuando retransmitir.