Análisis de TCP CUBIC usando un Modelo de Pérdida Determinístico

CUBIC TCP Analysis Using a Deterministic Model of Loss

Rodolfo Ignacio Ledesma-Goyzueta[1]

Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú

***Resumen*— CUBIC es uno de los protocolos más importantes desde que fue incorporado como opción predeterminada en el kernel Linux. El presente trabajo apunta a un análisis de la región cóncava del estado *"congestion avoidance"* para así entender el funcionamiento y mecanismo de CUBIC. El análisis matemático permite hallar el tamaño medio de ventana y así comprobar que el desempeño de CUBIC es mejor en redes de alto retardo y alto BDP.** [1]

***Abstract*— CUBIC is one of the most important protocols since it was set as default option in the Linux kernel. This paper aims at an analysis of the concave region of congestion avoidance to understand the functioning and mechanism of CUBIC. The mathematical analysis allows us to find the average size of window and thus verify that the performance of CUBIC is best in high retardation, and high BDP networks.**

***Palabras Clave*— AIMD, MIMD, TCP, control de congestión, cadena de Markov.**

***Key Words* — AIMD, MIMD, TCP, congestion control, Markov chain.**

I. Introducción

Las redes de altas velocidades están caracterizadas por un elevado producto de ancho de banda y retardo (BDP). Entiéndase este producto como la cantidad de datos que debe mantenerse en la red para que el enlace sea utilizado a plenitud. Hoy en día el gran problema es que las redes son utilizadas por debajo de sus capacidades, debido a las características, aún limitadas, de algunos protocolos de transporte. Los protocolos AIMD, MIMD oderivados (NewReno [1], [2], HTCP [3], HSTCP [4], STCP [5]) son muy drásticos ante un evento de pérdida, haciendo que el flujo de datos disminuya de tal manera que el enlace queda inutilizado por un periodo de tiempo, hasta que vuelve a llegar a un umbral, donde comienza el periodo de pérdida inminente. Asimismo, debido a los estudios, otras propuestas también compiten por llegar a un algoritmo que permita la plena utilización del enlace (e.g. FAST [6], Vegas [7], Westwood [8], BIC [9], CUBIC [10]). El punto crítico de un protocolo es siempre mantener el flujo de datos estable. La estabilidad de CUBIC radica en que el flujo tiene un crecimiento lento cerca del punto umbral entre la región cóncava y convexa, asimismo, tiene un parámetro de decremento multiplicativo bajo.

Las características de CUBIC son peculiares, comenzando por su función de crecimiento, la cual es cúbica, además de características como una región compatible con TCP estándar.

La idea principal para analizar este protocolo a través de este trabajo es que en el mundo la mayoría de los servidores son GNU/Linux, y este protocolo de control de congestión está anidado en el kernel [11] como configuración predeterminada. Por ello es importante analizar este protocolo, ya que su operación [12] afecta a redes WAN [13]. Sin embargo, como se sabe, no es el único, ya que Internet es una red muy heterogénea.

El presente trabajo se estructura de la siguiente manera, en la Sección II se describe de modo breve el funcionamiento del algoritmo de ajuste de ventana de CUBIC, tanto su función de crecimiento de ventana como sus parámetros. En la Sección III se realiza el análisis para llegar al modelo de pérdida determinístico, lo cual permite un entendimiento del funcionamiento de CUBIC, asimismo se logra un modelo con el que se prueba el mecanismo de CUBIC frente a cambios en alguno de sus parámetros. Ya en la Sección IV, se realiza un breve análisis del mecanismo de convergencia rápida (*fast convergence*). Posteriormente en la Sección V se realizan las validaciones empíricas de nuestro modelo comparado con modelos determinísticos de TCP Reno y STCP. Finalmente, en la Sección VI se exponen las

[1] Rodolfo Ignacio Ledesma-Goyzueta. E-mail: rilcdesmag@iece.org
Recepción: Setiembre 2009/Aceptación: Noviembre2010

conclusiones del trabajo después de analizar las validaciones.

II. Control de Congestión CUBIC

D CUBIC mantiene el sincronismo por ACK para incrementar su tamaño de ventana. Además, el protocolo no realiza cambios sobre las etapas de recuperación rápida (*fast recovery*) y retransmisión de TCP NewReno [2] y TCP SACK [14].

La función de la tasa de crecimiento de ventana de CUBIC se aprecia en la Fig. 1. En ella se muestra claramente la función cúbica con un punto de inflexión con ordenada W_{max}.

La operación de CUBIC se explica a partir de un evento de pérdida, que es cuando se registra W_{max} como el tamaño de ventana en donde se suscitó el evento de pérdida, después se realiza el decremento multiplicativo del tamaño de ventana por un factor β tal que β es un parámetro real. Después de esto, el' mecanismo de control de congestión ingresa a la etapa de *congestion avoidance* desde la etapa de *fast recovery*, donde luego se comienza a incrementar el tamaño de ventana utilizando la función cúbica en su región cóncava. La función cúbica toma como referencia de su incremento a W_{max}. Luego que la función alcanza el valor W_{max} se inicia el crecimiento en la región convexa. A la región cóncava se le conoce como región de estado estable, ya que el crecimiento tiene un valor máximo de referencia, en cambio a la región convexa se le conoce como región de sondeo máximo, porque es cuando la función está explorando el enlace más allá del último máximo. Esto se verifica además viendo la Fig. 1.

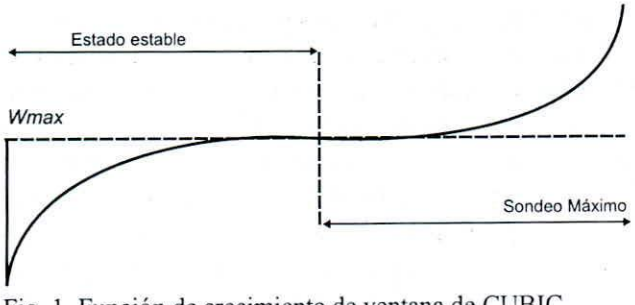


Fig. 1. Función de crecimiento de ventana de CUBIC.

La función cúbica permite un crecimiento lento cerca del último valor máximo, donde ocurrió una pérdida. Por ello, el tamaño de ventana permanece casi constante en esa zona. Durante la etapa de *congestion avoidance*, después de la etapa de *fast recovery*, CUBIC cambia el algoritmo de ajuste de ventana del TCP estándar. Se supone que W_{max} es el tamaño de ventana antes que el tamaño sea reducido en la etapa de *fast recovery*. Entonces la función de crecimiento de ventana de CUBIC es la siguiente función:

$$W(t) = C(t-K)^3 + W_{\max} \qquad (1)$$

donde C es una constante fija que determina la velocidad (agresividad) del crecimiento de ventana en redes de alto BDP, t es el tiempo transcurrido desde la última reducción del tamaño de ventana, y K es el periodo de tiempo estimado que tomaría la función en alcanzar el tamaño de ventana W_{max}, asumiendo que no haya pérdidas futuras dentro de ese lapso de tiempo. Entonces K se calcula con la siguiente ecuación:

$$K = \sqrt[3]{\frac{W_{\max}\beta}{C}} \qquad (2)$$

Tras recibir un ACK durante la etapa de *congestion avoidance*, CUBIC calcula la tasa de crecimiento de ventana durante el siguiente periodo RTT utilizando (1). El valor $W(t+\text{RTT})$ es la opción para ser el tamaño de ventana. Se asume que el tamaño de ventana actual es *cwnd*, entonces CUBIC opera en tres posibles modos diferentes, dependiendo del valor de *cwnd*. Primero, si *cwnd* es menor que el tamaño de ventana que alcanzaría TCP estándar en un tiempo t después del último evento de pérdida, entonces CUBIC está en la región compatible con TCP (*TCP friendly region*). En otro caso, si *cwnd* es menor que W_{max} entonces CUBIC está en la región cóncava y si *cwnd* es mayor, CUBIC se encuentra en la región convexa.

La Tabla I muestra la evolución de CUBIC y de sus versiones que fueron incorporadas en el kernel Linux, con lo cual se aprecia cómo el protocolo ha sido mejorado y se han corregido ciertos errores de implementación.

La Fig. 2 muestra el pseudocódigo del algoritmo de ajuste de ventana implementado en el kernel Linux.

TABLA I

EVOLUCIÓN DE LAS VERSIONES DE CUBIC

Versión	Kernel	Actualizaciones
pre -2.0	2.6.13	Se lanza la primera implementación de CUBIC en Linux [15].
2.0	2.6.15	CUBIC es oficialmente incluido en el kernel Linux.
	2.6.18	CUBIC reemplaza a BIC-TCP como protocolo TCP predeterminado en el kernel Linux.
	2.6.19	La implementación de CUBIC tiene un error de escala. Toma un mes solucionar el error desde que BIC-TCP fue reemplazado.
	2.6.21	La implementación original es optimizada por los desarrolladores de Linux para un mejor desempeño [16], [17]. El cálculo de la raíz cúbica de CUBIC mediante el método de bisección es reemplazado por el método de Newton-Raphson. En promedio, el método de bisección cuesta 1032 ciclos de reloj, mientras que el método N-R solamente 79.
2.1	2.6.22	La implementación original de CUBIC impuso el incremento de la ventana máximo a 32 paquetes por RTT. Esto lo heredó de BIC-TCP. Después de experimentos, se decide la eliminación en la región cóncava. Ello mejora la escalabilidad sobre redes de alto BDP.
	2.6.22-rc4	CUBIC mejora las etapas de *slow start* para un inicio rápido eliminando el valor *initial_sstresh*.
	2.6.23	Se elimina el uso del valor de la marca de tiempo (*timestamp*) recibido proveniente del cálculo del RTT para prevenir ataques de receptores malintencionados que reportan marcas erróneas con el fin de reducir los RTTs para mayor tasa de transferencia
2.2	2.6.25-rc3	La sujeción del crecimiento de ventana también es eliminada de la región convexa. Esta característica permite a CUBIC mejorar su velocidad de convergencia mientras mantiene su imparcialidad (*fairness*) y compatibilidad con TCP estándar.

Algoritmo de Linux CUBIC (v2.2)

Initialization:
  $tcp\_friendliness \longleftarrow 1,\ \beta \longleftarrow 0.2$
  $fast\_convergence \longleftarrow 1,\ C \longleftarrow 0.4$
  $cubic\_reset()$
On each ACK:
begin
  if $dMin$ **then** $dMin \longleftarrow min(dMin, RTT)$
  else $dMin \longleftarrow RTT$
  if $cwnd \leq ssthresh$ **then** $cwnd \longleftarrow cwnd + 1$
  else
    $cnt \longleftarrow cubic\_update()$
    if $cwnd\_cnt > cnt$ **then**
      $cwnd \longleftarrow cwnd + 1, cwnd\_cnt \longleftarrow 0$
    else $cwnd\_cnt \longleftarrow cwnd\_cnt + 1$
end
Packet loss:
begin
  $epoch\_start \longleftarrow 0$
  if $cwnd < W_{last\_max}$ **and** $fast\_convergence$ **then**
    $W_{last\_max} \longleftarrow cwnd * \frac{(2-\beta)}{2}$
  else $W_{last\_max} \longleftarrow cwnd$
  $ssthresh \longleftarrow cwnd \longleftarrow cwnd * (1-\beta)$
end
Timeout:
begin
  $cubic\_reset()$
end
$cubic\_update()$:
begin
  $ack\_cnt \longleftarrow ack\_cnt + 1$
  if $epoch\_start \leq 0$ **then**
    $epoch\_start \longleftarrow tcp\_time\_stamp$
    if $cwnd < W_{last\_max}$ **then**
      $K \longleftarrow \sqrt[3]{\frac{W_{last\_max} - cwnd}{C}}$
      $origin\_point \longleftarrow W_{last\_max}$
    else
      $K \longleftarrow 0$
      $origin\_point \longleftarrow cwnd$
    $ack\_cnt \longleftarrow 1$
    $W_{tcp} \longleftarrow cwnd$
  $t \longleftarrow tcp\_time\_stamp + dMin - epoch\_start$
  $target \longleftarrow origin\_point + C(t-K)^3$
  if $target > cwnd$ **then** $cnt \longleftarrow \frac{cwnd}{target - cwnd}$
  else $cnt \longleftarrow 100 * cwnd$
  if $tcp\_friendliness$ **then** $cubic\_tcp\_friendliness()$
end
$cubic\_tcp\_friendliness()$:
begin
  $W_{tcp} \longleftarrow W_{tcp} + \frac{3\beta}{2-\beta} * \frac{ack\_cnt}{cwnd}$
  $ack\_cnt \longleftarrow 0$
  if $W_{tcp} > cwnd$ **then**
    $max\_cnt \longleftarrow \frac{cwnd}{W_{tcp} - cwnd}$
    if $cnt > max\_cnt$ **then** $cnt \longleftarrow max\_cnt$
end
$cubic\_reset()$:
begin
  $W_{last\_max} \longleftarrow 0,\ epoch\_start \longleftarrow 0,\ origin\_point \longleftarrow 0$
  $dMin \longleftarrow 0,\ W_{tcp} \longleftarrow 0,\ K \longleftarrow 0,\ ack\_cnt \longleftarrow 0$
end

Fig. 2. Pseudocódigo del algoritmo de CUBIC.

A. Regiones

El protocolo se encuentra en la región compatible con TCP, si *cwnd* es menor que $W_{TCP}(t)$, entonces *cwnd* toma el valor de $W_{TCP}(t)$ en cada recepción de un ACK. Entiéndase por $W_{TCP}(t)$ como la tasa de crecimiento de TCP estándar. En la Fig. 2 en *cubic_tcp_friendliness()* se describe el comportamiento en dicha región.

Si el protocolo no se encuentra en la región compatible con TCP y además *cwnd* es menor que W_{max} entonces el protocolo está en la región cóncava. En esta región *cwnd* es incrementado siguiendo (1) hasta alcanzar W_{max}.

Una vez que *cwnd* alcanza el valor W_{max} o es mayor, el protocolo se encuentra en la región convexa. Esto puede significar más ancho de banda disponible, debido a cambios en las condiciones de la red. Comienza el sondeo de la red, a causa de modificaciones en el ambiente de la red, ya que Internet tiene muchas fluctuaciones de entrada y salida de flujos de la red. Como se aprecia en la Fig. 1 y como ya se mencionó, la tasa de crecimiento pasado W_{max} es más lento al principio.

B. Convergencia rápida

CUBIC introduce heurística en el protocolo para mejorar la velocidad de convergencia.

Cuando un flujo entra a la red, entra en competencia por el ancho de banda del enlace. Asimismo, los flujos existentes en la red deben compartir el ancho de banda y desprenderse del ancho de banda adjudicado para permitir el crecimiento de flujos entrantes. Para ello, CUBIC añade el mecanismo de convergencia rápida (*fast convergence*).

Con el mecanismo de convergencia rápida, cuando ocurre un evento de pérdida, se hace una comparación entre el último W_{max} guardado con el último W_{max} alcanzado antes de la reducción multiplicativa. Si este último es menor que el W_{max} guardado, entonces es un indicador que hay menos ancho de banda disponible, con lo que se deduce acerca de los cambios en el uso del enlace. Esto se puede entender con el siguiente algoritmo:

```
if (W_max < W_last_max){
W_last_max = W_max;
W_max = W_max*(2-beta)/2;
} else
W_last_max = W_max;
```

Denotamos para el algoritmo anterior `W_max` (entero) como el tamaño de ventana máximo del evento ocurrido y `W_last_max` (entero) como el último valor máximo guardado, `beta` representa el factor de decremento β.

III. MODELO

Se asume que la probabilidad de pérdida de un paquete es p, y esas pérdidas aleatorias son independientes. Consideramos entonces un flujo CUBIC de larga duración, que opera en la región cóncava del estado de control de congestión. El modelamiento de CUBIC se simplificará en el modo matemático al realizarlo en la región cóncava, pero no se perderá la perspectiva del análisis del mecanismo de control de congestión de CUBIC.

El crecimiento del tamaño de ventana puede ser visto como una cadena de Markov [18]. Sin embargo, para simplificar el análisis se desarrolla considerando los valores promedios.

Asumiendo que el tamaño de ventana inicial es de w, se explica las siguientes funciones:

$W(r, w)$, tamaño de ventana después de r paquetes reconocidos.

$T(r, w)$, tiempo que toman los r paquetes para ser enviados y reconocidos.

$R_{max}(w)$, número de paquetes reconocidos antes de una pérdida.

Se considera que las perdidas pueden ser originadas, e. g. por un desborde del *buffer* o por otro fenómeno perjudicial que evite el correcto reconocimiento de un paquete. Adicionalmente, basta que se pierda un paquete para que el protocolo de transporte decida sobre esa pérdida. Se sabe entonces que R_i es una variante de carácter dicotómico, cuyas concreciones serán A(ck), L(oss), por lo tanto las correspondientes probabilidades $q=1-p$ y p son:

$$P[R_j = A] = q \qquad (3.a)$$

$$P[R_j = L] = p \qquad (3.b)$$

Sabiendo eso, la probabilidad del número de paquetes reconocidos sigue una ley geométrica [18], [19] y considerando que $(r-1)$ paquetes $(\leq R_{max})$ han sido reconocidos, se define la función de densidad de probabilidad de R, tal que R es una variable aleatoria independiente e idénticamente distribuida (i.i.d.), por lo tanto:

$$P[R = r] = (1-p)^{r-1} p \quad , r \leq R_{\max} + 1 \qquad (4)$$

Entonces el ciclo concluye en un periodo $T(R, w)$ y con un tamaño de ventana $W(R, w)$.

Considerando una cadena de Markov homogénea en el tiempo conformada por los estados $\{w_n\}$, se especifica la evolución de w_n y T_n:

$$w_{n+1} = W(R_n, w_n) \quad (5.a)$$

$$T_{n+1} = T(R_n, w_n) \quad (5.b)$$

La cadena de Markov tiene como probabilidades de transición a (4) y (5.a). Se considera que $W(r, w)$ y $T(r, w)$ son variables aleatorias i.i.d.. La distribución de $T(r, w)$ tiene un valor esperado $E[T]$. Por lo tanto, se calcula la tasa de transferencia λ considerando los valores esperados de las variables aleatorias R y T:

$$x = \frac{E[R]}{E[T]} \quad (6)$$

Para facilitar el cálculo, se realiza el siguiente razonamiento, partiendo de la ecuación de la tasa de crecimiento de ventana de CUBIC, el cual no depende de RTT, solo del tiempo continuo que demora en alcanzar desde $w=(1-\beta)\omega$ hasta $W_{max}=\omega$

Por lo tanto, el periodo promedio es K, sabiendo que cuando llega a $W_{max}=\omega$ pasa del estado ω_i a ω_i+1 y entonces el ciclo comienza en $w=(1-\beta)\omega$ nuevamente, después de $R_{max}+1=R_i+1=1/p$ paquetes:

$$K = \sqrt[3]{\frac{\omega\beta}{C}} \quad (7)$$

Con ello:

$$W\left(\frac{1}{p}, w\right) = W_{\max} \quad (8)$$

Además, sabiendo que la función de la tasa de crecimiento está sujeta a la periodicidad de la pérdida, se puede derivar que $E[T]=K$.

El número promedio de paquetes enviados es:

$$E[R] = rP[R = r] \quad (9)$$

Como el número de paquetes enviados antes de un pérdida está ya establecido, asumiendo que se pierde un paquete en cada ciclo de la función de la tasa de crecimiento, entonces resulta $1/p=E[R]$.

Sabiendo que la dinámica del protocolo en cada RTT tiene una función de crecimiento de:

$$W(t) = C\,(t.RTT - \sqrt[3]{\frac{\omega\beta}{C}})^3 + \omega \quad (10)$$

y asumiendo que el protocolo solo trabaja en la región cóncava del estado de *congestion avoidance*, conociendo el periodo de cada ciclo, entonces la cantidad de paquetes transmitidos en cada ciclo es:

$$\frac{1}{p} = \int_0^{\frac{1}{RTT}\sqrt[3]{\frac{\omega\beta}{C}}} C\,(t.RTT - \sqrt[3]{\frac{\omega\beta}{C}})^3 + \omega\, dt \quad (11)$$

La expresión anterior (11) es la cantidad de paquetes por ciclo. Como se puede apreciar de la ecuación, se ha modificado la función de crecimiento para que dependa de RTT y la unidad de referencia de tiempo sea también RTT. Esta modificación no afecta a los cálculos principales. Para graficar lo mencionado se muestra la Fig. 3, siendo la curva de (10), para el valor predeterminado por el protocolo β= 0.2.

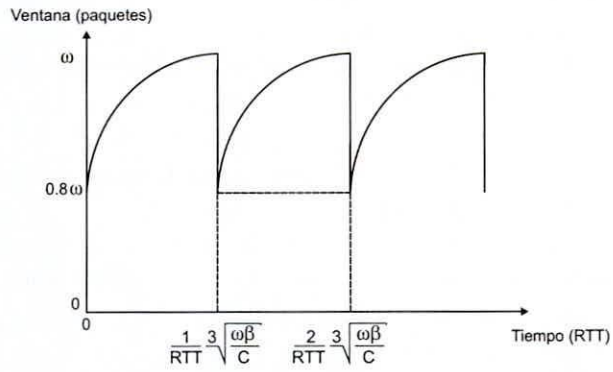


Fig. 3. Crecimiento de ventana bajo pérdidas periódicas.

El tamaño de ventana promedio es igual a la cantidad de paquetes transmitidos durante un lapso de tiempo (teniendo como base temporal el RTT) hasta que un evento de pérdida ocurre, por lo tanto:

$$E[W_{CUBIC}] = \frac{1}{\frac{1}{RTT}\sqrt[3]{\frac{\omega\beta}{C}}} \int_0^{\frac{1}{RTT}\sqrt[3]{\frac{\omega\beta}{C}}} C\,(t.RTT - \sqrt[3]{\frac{\omega\beta}{C}})^3 + \omega\, dt \quad (12)$$

Resolviendo (12) se puede llegar a la expresión del tamaño medio de ventana:

$$E[W_{CUBIC}] = \sqrt[4]{C\,\frac{RTT^3}{p}\,\frac{4-\beta}{\beta}} \quad (13)$$

Adicionalmente, se considera que los paquetes viajan a través de un enlace disponible para dicho flujo, y dichos paquetes llegan al enlace siguiendo una distribución de Poisson, con una tasa de llegada λ, y donde este parámetro λ depende del tamaño medio de ventana de (13), por lo tanto:

$$x = \frac{E[R]}{E[T]} = \frac{E[W]}{RTT} \qquad (14)$$

IV. Análisis de Fast Convergence

El mecanismo de *fast convergence*, como se indicó en la Sección II-B, le añade heurística al protocolo, para así dar lugar al crecimiento de otros flujos entrantes en la red. Se considera que todas las condiciones se cumplen para que se ejecute el mecanismo de *fast convergence*.

El desprendimiento de ancho de banda tiene un límite de tiempo, pues, un flujo que sufre una pérdida, puede tener dos comportamientos, uno que parte utilizando *fast convergence* y el otro no. Entonces, pasado un lapso de tiempo va a llegar a un umbral donde el tamaño de ventana de ambos flujos es igual. Pasado dicho umbral, el crecimiento del flujo que partió utilizando *fast convergence* se hace más agresivo que el mismo flujo si no hubiera partido usando el mecanismo de *fast convergence*.

Se considera K_1 y K_2 como los periodos de tiempo estimado utilizando *fast convergence* y no utilizándolo, respectivamente. Asimismo, ω_1 y ω_2 son los tamaños máximos de ventana de referencia de la función cúbica, utilizando *fast convergence* y no utilizándolo, respectivamente.

Entonces, se puede hallar el umbral de tiempo, sabiendo que en ese punto el tamaño de ventana alcanzado es igual:

$$C(t-K_1)^3 + \omega_1 = C(t-K_2)^3 + \omega_2 \qquad (15)$$

Resolviendo (15), se puede hallar la ecuación del lapso de tiempo hasta llegar al umbral, y así poder hallar el tiempo límite de *fast convergence*, por lo tanto:

$$at^2 + bt + c = 0 \quad ; \qquad (16.a)$$

$$a = 1, \quad b = -\frac{K_2^2 - K_1^2}{K_2 - K_1}, \quad c = \frac{K_2^2 - K_1^2}{3(K_2 - K_1)} - \frac{\omega_2 - \omega_1}{3C(K_2 - K_1)} \qquad (16.b)$$

La ecuación (16) tiene por única solución:

$$t_{\lim} = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad (17)$$

El resultado anterior es despreciable, después de un evento de pérdida o si el flujo concluye la transferencia de datos antes del tiempo límite de *fast convergence*. Por lo tanto, el resultado de (17) se puede considerar como el tiempo límite t_{lim} o duración de *fast convergence*.

V. Validación

La validación en esta sección es empírica, ya que se basa en los resultados del análisis en las secciones previas. El análisis matemático previo nos permite obtener resultados teóricos que sirven para el entendimiento de las características del protocolo de control de congestión a analizar, CUBIC.

A. Comportamiento de CUBIC

Para apreciar la influencia de la probabilidad periódica sobre (13), se consideran dos escenarios. En el primer escenario se considera un flujo con RTT=10 ms. En la Fig. 4 se muestra la evolución del tamaño medio de ventana con relación a la tasa de pérdida. En el segundo escenario se considera un flujo con un RTT mayor, RTT= 100 ms. En la Fig. 5 se muestra el comportamiento del tamaño medio de ventana bajo dichas condiciones.

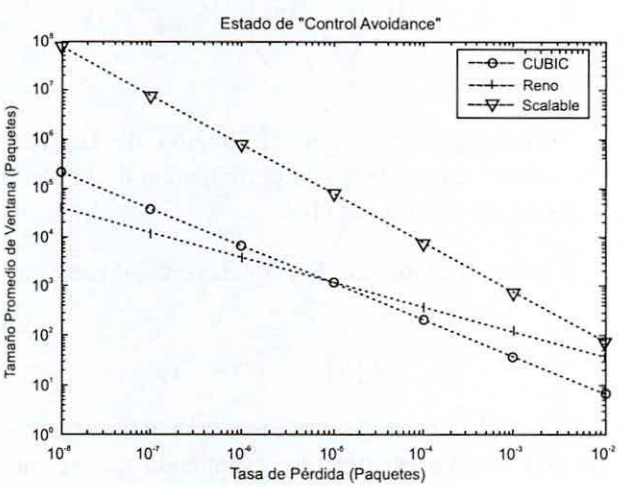


Fig. 4. RTT= 10 ms.

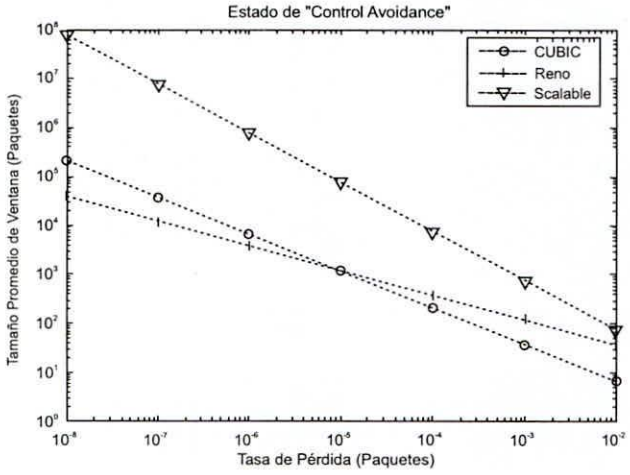


Fig. 5. RTT= 100ms.

Como se puede apreciar en la Fig. 4, el tamaño medio de ventana de CUBIC tiene una evolución menor que las de STCP y Reno TCP. El primero es muy agresivo, mientras que el segundo mantiene un crecimiento más cauteloso, pero todavía muy agresivo. Ya en la Fig. 5 se aprecia un mejor desempeño de CUBIC, teniendo un comportamiento más agresivo en redes de alto retardo, STCP tiene un comportamiento aún más agresivo, pero con menor desempeño que en redes de bajo retardo. Además, TCP Reno tiene el más pobre desempeño en redes de alto BDP, con alto retardo. Esto demuestra fácilmente la falta de ecuanimidad tanto de STCP como TCP Reno con flujos sobre enlaces con alto retardo.

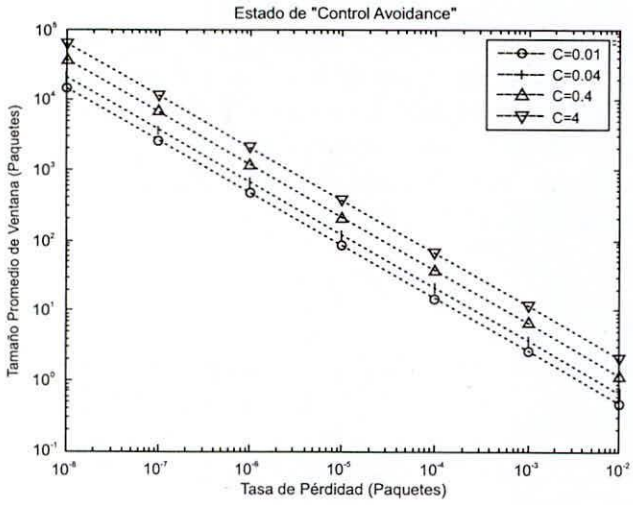


Fig. 6. RTT= 10 ms.

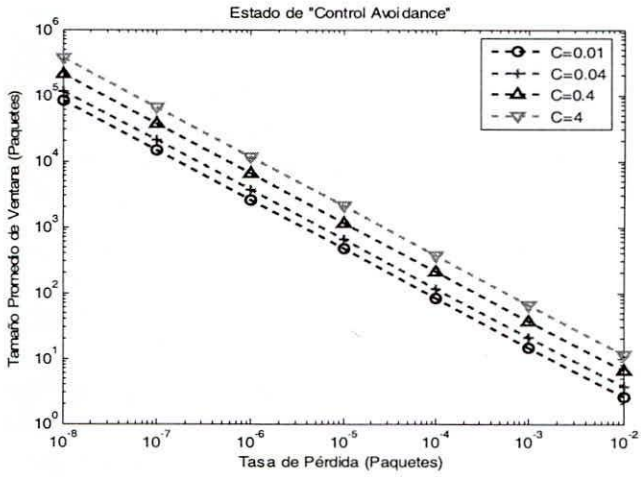


Fig. 7. RTT= 100 ms.

El parámetro C es una constante definida por CUBIC, ello detona la agresividad del crecimiento de ventana. La Fig. 6 muestra la respuesta de una red con RTT= 10 ms., ante diferentes valores de C. La Fig. 7 denota una mejora de desempeño en redes de mayor retardo, RTT= 100 ms. Para entender y apreciar la distribución de los parámetros K y C a través de diferentes tasas de pérdida p, se muestra la Fig. 8. El retardo escogido es RTT= 100 ms., pues como se pudo ver, es el tipo de red donde CUBIC tiene un mejor desempeño como protocolo de control de congestión. Por ejemplo, se deduce entonces que para $p=10^{-8}$ y C=0.04, resulta K= 99 seg., el cual es el tiempo estimado para llegar a W_{max} sin pérdidas más adelante. Asimismo, para $p=10^{-8}$ yC =0.4, que es el valor predeterminado del protocolo, se obtiene K= 55 seg. Para un valor constante de la tasa de pérdida p, haciendo la variación de C, se aprecia que la agresividad del crecimiento del tamaño de ventana se refleja en un tiempo estimado K menor.

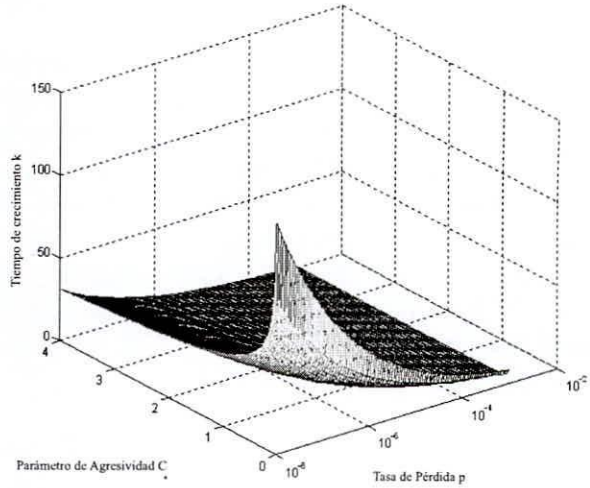


Fig. 8. Relación entre C, p y K. RTT= 100 ms.

B. Caso particular, fast convergence

Debido a la heurística incorporada en el protocolo de control de congestión por CUBIC, y después del breve análisis del tiempo límite de *fast convergence*, se realiza la validación empírica del análisis de la Sección IV.

La validación se realiza dentro del intervalo de tiempo Δt, $\Delta t = [0; t_{lim}]$. Dicho intervalo denota por consiguiente la duración de *fast convergence*. En la Fig. 9 se muestra los paquetes disponibles antes los diferentes intervalos de duración de *fast convergence*. Se refiere a paquetes disponibles al espacio de ancho de banda medido en paquetes que se le permite dentro del intervalo que dura *fast convergence* para que otros flujos entrantes en la red puedan ocuparlos y poder así crecer, permitiendo una distribución de ancho de banda que permita un equilibrio de utilización entre todos los flujos en el enlace. Como se sabe, de lo explicado en la sección anterior, el tiempo límite o duración del mecanismo de *fast convergence* se cumple si se asume la no existencia de un evento de pérdida antes de dicho periodo. En la Fig. 10 se representan los paquetes disponibles en función del tamaño máximo de ventana del último evento de pérdida, W_{max}. Ello nos da una mejor apreciación de los paquetes disponibles a ser ocupados por otros flujos desde el punto de vista del flujo dador y de su tamaño máximo de ventana referencial, cuyo ancho de banda o paquetes disponibles son puestos a disposición de otros flujos entrantes. Como se afirmó en la Sección II-B, la heurística añadida al mecanismo de control de congestión detecta un flujo entrante mediante la comparación del tamaño máximo de ventana último y el corriente y comprobándose el decremento en comparación al último.

C. Utilización

La utilización que un protocolo puede alcanzar sobre los recursos de un enlace es un parámetro importante en redes de alta velocidad. En esta sección se realizan las simulaciones utilizando el simulador *ns-2* [20], haciendo uso del módulo TCP-Linux. Las simulaciones se realizaron sobre enlaces dedicados de 150 Mbps para cada protocolo. Para la Fig. 11 y Fig. 12 el RTT base que se considera es de 20 ms. Con ello se toma una red de bajo BDP.

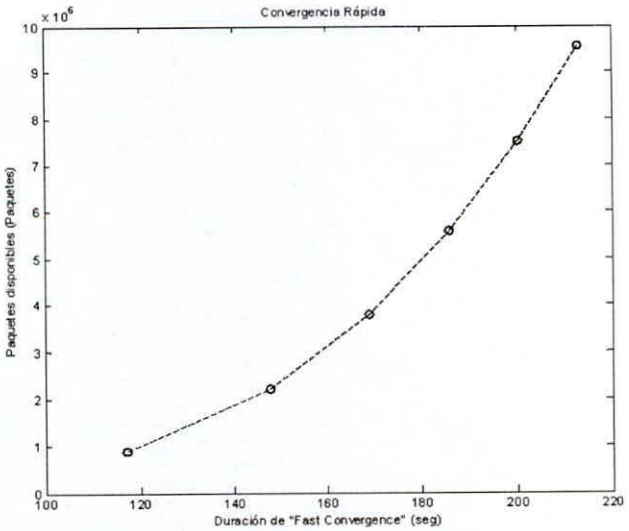


Fig. 9. Paquetes disponibles dentro del intervalo Δt.

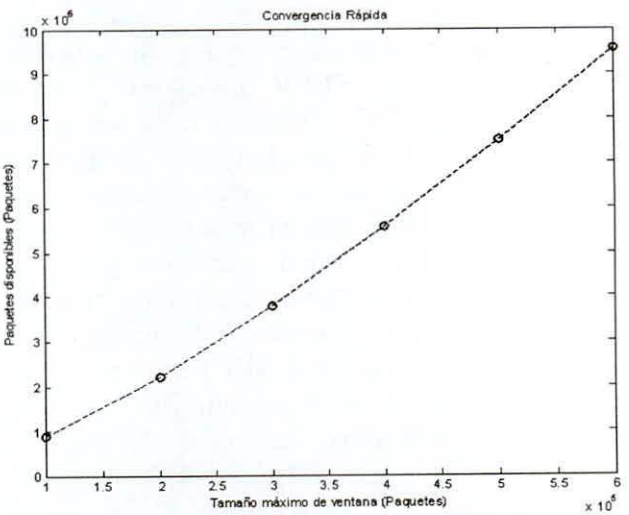
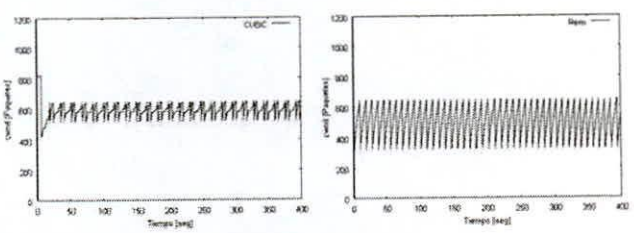


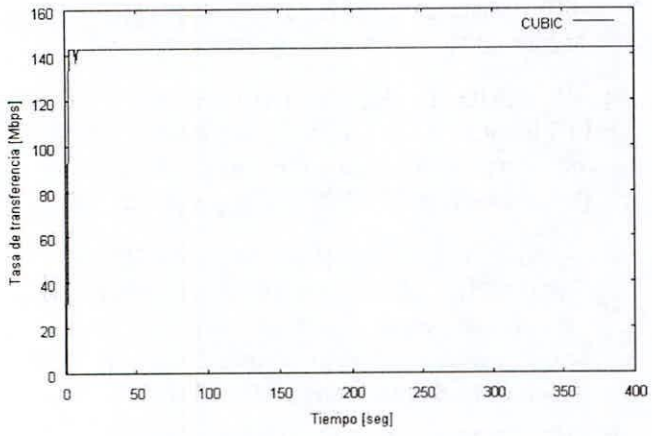
Fig. 10. Paquetes disponibles con relación al tamaño máximo de ventana referencial, W_{max}.

Como se puede apreciar de la Fig. 11, se registran pérdidas periódicas para ambos flujos. Asimismo, la Fig. 12, tanto CUBIC como TCP Reno tienen una utilización del enlace cercano al ideal, siendo un entorno de red favorable a TCP Reno por las características, cf. Sección V-A.

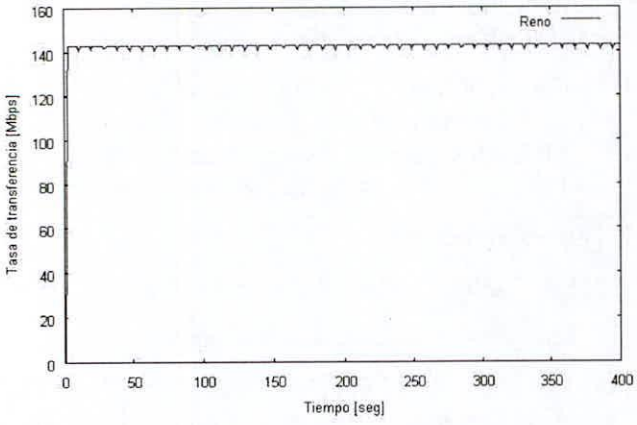


(a) (b)

Fig. 11. Curvas de tamaños de ventana. (a) CUBIC. (b) TCP Reno.



(a)



(b)

Fig. 12. (a) Tasa de transferencia de CUBIC. (b) Tasa de transferencia de TCP Reno.

La utilización promedio para el flujo CUBIC de la Fig. 12(a) es de 95.1% y del flujo TCP Reno de la Fig. 13(b) se calcula en 94.9%.

Ya para el caso de un entorno de red de alto BDP, se mantiene la capacidad del enlace dedicado en 150 Mbps y además se considera un RTT de 380 ms. En la Fig. 13(a) se registran pérdidas periódicas solamente para el flujo CUBIC. Sin embargo, para el flujo Reno de la Fig. 13(b) se tiene un crecimiento continuo del tamaño de ventana.

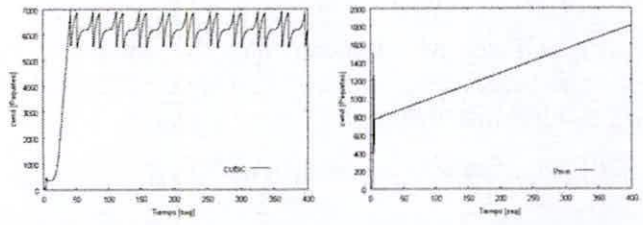
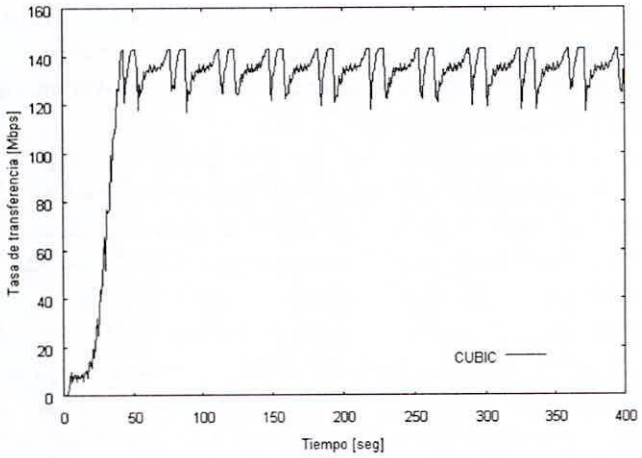
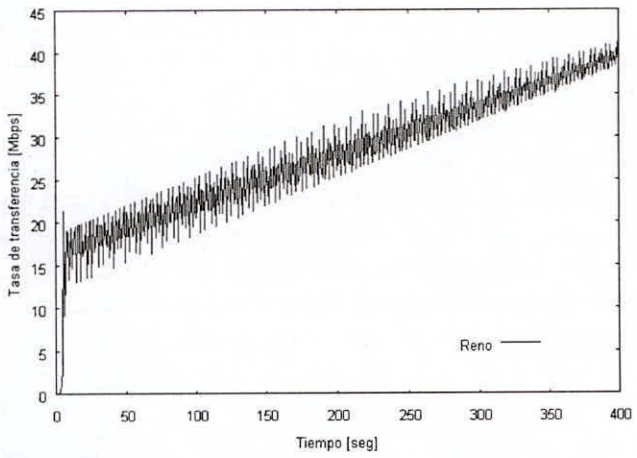


Fig. 13. Curvas de tamaños de ventana. (a) CUBIC. (b) TCP Reno.



(a)



(b)

Fig. 14. (a) Tasa de transferencia de CUBIC. (b) Tasa de transferencia de TCP Reno.

Debido a que para este caso el entorno de red es favorable a CUBIC, cf. Sección V-A, en la Fig. 14(a) se registra que la tasa de transferencia en CUBIC tiene un mayor aprovechamiento de los recursos del enlace, teniendo una utilización promedio de 83.5%, que a comparación de la utilización promedio de TCP Reno, Fig. 14(b), el cual es de 18.5%, se puede apreciar un decremento en el desempeño de Reno, el cual mantiene su enlace subutilizado.

VI. CONCLUSIÓN

En este trabajo se ha analizado un protocolo de control de congestión basado en pérdidas, CUBIC. El modelo determinístico de CUBIC desarrollado, considerando la variable de la tasa de pérdida de paquetes, permitió estudiar sus características de ajuste de ventana.

Asimismo, se presentó en la Sección IV un análisis de la heurística de *fast convergence* insertada en este algoritmo de control de congestión, con ello se pudo comprobar la importancia de este mecanismo que

apunta a mejorar la equidad en la distribución de ancho de banda, considerando los espacios liberados en unidades de paquetes, los cuales serán aprovechados por otros flujos nacientes.

Las validaciones de la Sección V-[A, B] se realizaron en GNU Octave para demostrar el modelo al cual se llegó en la Sección III. Con dichas gráficas se puede concluir que el desempeño de CUBIC en estado estacionario mejora en redes de alto RTT. Ya en la Sección V-C se hizo uso del simulador ns-2 para un breve análisis de la utilización del enlace para el desempeño dinámico de dos protocolos, CUBIC y TCP Reno. Ambos son protocolos basados en pérdidas. Las simulaciones mostraron que el desempeño dinámico de CUBIC es mejor que el de TCP Reno en redes de alto RTT, ende un mayor aprovechamiento del ancho de banda en redes de alto RTT.

Actualmente CUBIC sigue siendo un protocolo en desarrollo, a pesar que es la opción de protocolo predeterminada en el kernel Linux. Cabe mencionar además, que como propuesta de futuros trabajos se pueden hacer modelos estocásticos de CUBIC basados en pérdidas aleatorias y hacer estudios del desempeño de este protocolo en redes inalámbricas.

Referencias

[1] JACOBSON V. *Congestion avoidance and control*, in Proceedings of SIGCOMM '88, Standford CA, ACM, 1988.

[2] FLOYD S., HENDERSON T., GURTOV A. *The NewReno Modification to TCP's Fast Recovery Algorithm*, RFC 3782, Abril 2004.

[3] SHORTEN R. N., LEITH D. J., *H-TCP: TCP for high-speed and long-distance networks*, in Proceedings of he Second PFLDNet Workshop, Argonne, Illinois, Febrero 2004.

[4] FLOYD S., *HighSpeed TCP for Large Congestion Windows*, RFC 3649, Diciembre 2003.

[5] KELLY T., *Scalable TCP: Improving performance in highspeed wide area networks*, ACM SIGCOMM Computer Communication Review 33, 2, 83–91, Abril 2003.

[6] JIN C.,. WEI D. X, LOW S. H. FAST TCP: motivation, architecture, algorithms, performance, in Proceedings of IEEE INFOCOM, Hong Kong, Marzo 2004.

[7] BRAKMO L. S., O'MALLEY S. W., AND PETERSON L. L.,*TCP Vegas: new techniques for congestion detection and avoidance*, in Proceedings ACM SIGCOMM pp. 24-35, 1994.

[8] CASETTI C., GERLA M., MASCOLO S., SANADIDI M. Y., WANG R. *TCP Westwood: Bandwidth estimation for enhanced transport over wireless links* in Proceedings of ACM Mobicom, Roma, Italia, Julio 2001.

[9] XU L., HARFOUSH K., RHEE I. *Binary increase congestion control for fast long-distance networks*, in Proceedings of IEEE INFOCOM, Hong Kong, Marzo 2004.

[10] HA S., RHEE I. AND XU L. *CUBIC: A New TCP-Friendly High-Speed TCP Variant*, ACM SIGOPS Operating System Review, Volume 42, Issue 5, pp. 64–74, Julio 2008.

[11] www.kernel.org

[12] http://lxr.linux.no/linux+v2.6.28.7/net/ipv4/tcp_ipv4.c

[13] http://netsrv.csc.ncsu.edu/wiki/index.php/TCP_Testing

[14] MATHIS M., FLOYD S., ROMANOW A. *TCP Selective Acknowledgment Options*, RFC 2018, Octubre 2006.

[15] http://netsrv.csc.ncsu.edu/twiki/pub/Main/BIC/cubic-kernel-2.6.13.patch

[16] HEMMINGER S. Cubic root benchmark code, [http://lkml.org/lkml/2007/3/13/331]

[17] TARREAU C. W. Cubic optimization, [http://git.kernel.org/?p=linux/kernel/git/davem/net-2.6.git;a=commit;h=7e58886b45bc4a309aeaa8178ef89ff767daaf7f]

[18]. HINES W. W AND MONTGOMERY D. C. Probability and Statistics in Engineering and Management Science (traducido por Act. Ma. De Lourdes Fournier de Fournier), 1era. Edición, CECSA, Mayo 1986.

[19]. LOPEZ M. Fundamentos y Métodos de Estadística, 3era. edición, Ediciones PIRAMIDE, 1981.

[20] http://www.isi.edu/nsnam/ns/