Análisis del protocolo TCP aplicado al tráfico inter-hospitalario asociado a los nuevos servicios sanitarios de telemedicina

I. Martínez, J. García

Grupo de Tecnología de las Comunicaciones (GTC). Instituto de Investigación de Ingeniería en Aragón (I3A)
Centro Politécnico Superior (CPS). Universidad de Zaragoza (UZ).
Edificio Ada Byron. Campus Río Ebro. c/María de Luna 3, 50.018 – Zaragoza (Spain)
Teléfono: 976 76 19 45 Fax: 976 76 21 11 E-mail: imr@unizar.es

Resumen

With the advance of the network technologies, multimedia applications require efficient mechanisms of flow and congestion control, usually over Transmission Control Protocol (TCP), and efficient distribution of available resources. In this work, we have studied different versions of TCP and buffer priority methods applied to the traffic associated to the new services of telemedicine. The state of the art in queue politics and control mechanisms has been reviewed. Advanced traffic models and representative network topologies related to multiple clinical communications, have been included in the simulation environments (over NS). Thus, the results obtained suggest that TCP-Vegas and CBQ queue priority mechanism may offer the best behaviour for the proposed telemedicine scenarios.

1. Introducción

Transmission Control Protocol (TCP) se define como un protocolo de transporte que proporciona entrega de paquetes fiable extremo a extremo usando *Internet Protocol* (IP), que da servicios de conmutación de paquetes sobre redes heterogéneas. Inicialmente estuvo pensado para proporcionar servicios fiables de datos sobre redes cableadas convencionales y con un rango limitado de tasas de transmisión y retardos de propagación. Así, uno de los puntos fuertes de TCP reside en su mecanismo de control de congestión propuesto por primera vez por Van Jacobson [1]. Actualmente, las nuevas tecnologías soportadas sobre redes con retardos altos, elevada ocupación y restrictivos requisitos de calidad de servicio (*Quality of Service*, QoS) para el tráfico multimedia interactivo, hacen que el ámbito de aplicación de TCP comience a necesitar ser evaluado y adaptado a estos nuevos entornos, como se trata ampliamente en muchos de los últimos estudios publicados [2]-[6].

Esta situación se ajusta a los nuevos servicios de telemedicina que suelen estar basados en tecnologías multimedia, frecuentemente soportadas en TCP/IP, y diseñados para soportar múltiples aplicaciones clínicas sobre diferentes topologías de red. Estos entornos heterogéneos requieren diferentes requisitos de QoS y, por tanto, se hace necesario una estimación precisa del funcionamiento de la red para el éxito de dichos servicios de telemedicina [7]-[8].
El control de congestión usado en TCP/IP se basa en algoritmos de ventana deslizante y detección de paquetes perdidos para declarar congestión; es decir, los extremos de la red incrementan gradualmente el tamaño de las ventanas de transmisión hasta que se produce congestión y, por tanto, pérdidas [9]-[10].

Para estas situaciones tan variables se han propuesto múltiples versiones del protocolo buscando adecuarse a la casuística observada en las redes actuales. Algunos estudios adaptan los parámetros de los algoritmos a sus valores óptimos según el ancho de banda (*BandWidth*, BW) disponible a cada momento [11]. Otros trabajos proponen minimizan los retardos con mejoras en los mecanismos que actúan ante situaciones de pérdidas, para simultanear el envío de datos y el número de retransmisiones [12]. Sin embargo, cada estudio debe ser particular para el tipo de escenario que se evalúa, según se den pérdidas continuadas, a ráfagas, aleatorias, etc. o retardos elevados, pequeños, variables en el tiempo, etc. [13].
Esto es fundamental en entornos de telemedicina que combinan servicios TCP/IP con aplicaciones a tiempo real basadas en *User Datagram Protocol* (UDP). UDP suele generar gran cantidad de tráfico a ráfagas que tiende a ser prioritario y congestiona los *buffers*. Esta situación conjunta TCP/UDP es la que estudia este trabajo evaluando los algoritmos más óptimos adecuando la configuración de los parámetros de generación (tamaño, tasa, mecanismos de control, valores iniciales, etc.) y de los *buffers* intermedios (disciplina de servicio, asignación de prioridades,etc.) a los recursos disponibles según los requisitos de QoS
En la sección 2 se describe la topología básica del escenario de estudio y la casuística contemplada, representativa de un servicio de telemedicina que integre las aplicaciones más significativas. En las secciones 3 y 4 se revisa el estado de las investigaciones sobre los mecanismos de control de flujo y las distintas versiones de TCP implementadas. Los resultados obtenidos y su evaluación sobre entornos médicos se discuten en la sección 5, y en la sección 6 se muestran las conclusiones finales.

2. Escenario de estudio

El análisis de QoS parte de un trabajo anterior [14] que plantea una metodología de evaluación técnica sobre un escenario como el que muestra Fig. 1. Sobre esta topología genérica se ha simulado con la herramienta NS-2 [15] diversas situaciones de simultaneidad para caracterizar un servicio de telemedicina lo más completamente posible. Así, se han incluido las siguientes características:

- **Modelo de aplicación:** Servicios *Real Time*, RT, sobre UDP y basados en *codecs* de audio y vídeo (de calidad media -tipo1- y alta -tipo2-) asociados a videoconferencias médicas, telediagnóstico, etc.; y servicios *Store&Forward*, SF, sobre TCP como:
  - *HyperText Transfer Protocol* (HTTP), para consultas a bases de datos médicas y gestión del Historial Clínico Electrónico del paciente (HCE), etc., según un modelo de tres niveles basado en distribuciones log-normal, gamma y Pareto [16].
  - Telnet, para acceso remoto a equipos clínicos, según un modelo basado en *on-off* y Pareto [17].
  - *File Transfer Protocol* (FTP), para transferencia de archivos médicos y pruebas relacionadas con el paciente (ver Tabla 1), basado en dos modelos: tipo1, de tasa constante (*Constant Bit Rate*, CBR) y prioridad alta; y tipo2, de tasa variable (*Variable Bit Rate*, VBR) y baja prioridad [18].
- **Modelo de red de acceso:** Ethernet conmutada, correspondiente a un acceso de *Local Area Network* (LAN) intra-hospitalaria a 10Mbps. Sobre él se multiplexa un número variable de conexiones RT y SF concentradas en un *switch Fast Ethernet* que proporciona distintos enlaces a 2Mbps hacia el *front-end* exterior (*edge router*) [19].
- **Modelo de red troncal:** Basada en tecnologías *Frame Relay* (FR), correspondientes a conexiones *Wide Area Network* (WAN) inter-hospitalarias. Se ha considerado un *Committed Information Rate* (CIR) variable de n·64Kbps (un valor habitual es 256Kbps) y múltiples políticas de asignación de prioridades implementadas en NS-2: *First Input First Output* (FIFO), *Random Early Detection* (RED), *Fair Queuing* (FQ), *Stochastic FQ* (SFQ), *Weighted FQ* (WFQ), *Deficit Round Robin* (DRR) y *Class Based Queuing* (CBQ) [20].

Siguiendo todas estas consideraciones, se ha buscado plantear el mayor número posible de situaciones distintas en la evaluación de los tráficos presentes en el escenario. Así se propone en Fig. 2 un cronograma de tiempos representativo de las potenciales simultaneidades que puede darse en 90 min. de observación de un servicio completo de telemedicina.

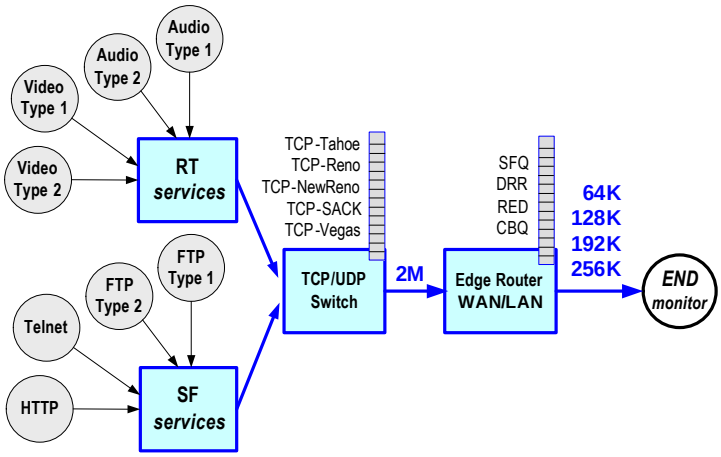

Fig. 1. Topología de simulación sobre NS-2.

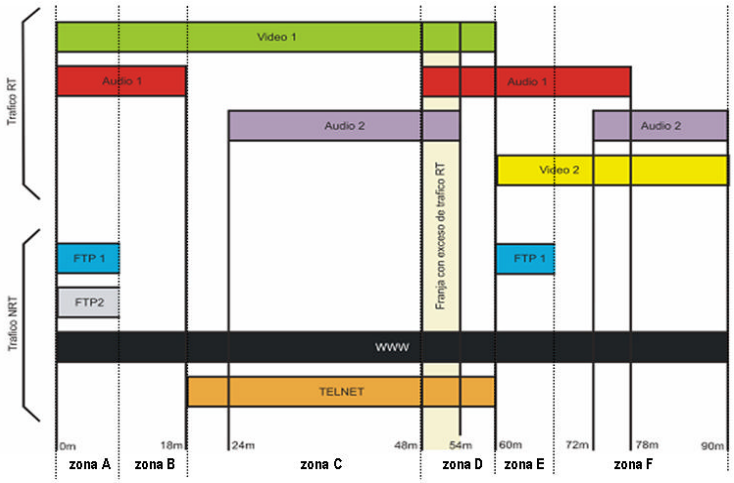

Fig. 2. Cronograma de simulación para un servicio de telemedicina

prueba médica	(MB)	S0	S1	S2
imagen simple	67.1			
imagen tórax	268.4			
radiografía	*2.5*	**720**	**240**	**96**
imagen simple	400			
mamografía	*5*	**2000**	**670**	**267**
estándar	2.1			
ecografía	*24·60*	**3020**	**1008**	**384**
slice prueba	4.2			
TAC	*40*	**168**	**56**	**14**
slice prueba	4.2			
RNM	*100*	**420**	**140**	**28**
estándar	8.4			
DIVAS	*50*	**420**	**140**	**28**
estándar	2.1			
gammagrafía	*50*	**105**	**35**	**3.5**
ECG ordinario	media	**40.5**	**13.5**	**1.7**
holter	media	**692**	**23.1**	**3.4**
EEG ordinario	media	**1.9**	**0.7**	**0.7**
estudio sueño	media	**664**	**222**	**3.4**
digitalizador	media	**67.2**		
texto WORD	media	**0.29**		
texto PDF	media	**0.08**		

Tabla 1. Para cada prueba médica: tamaños originales S0 (con 16 bits/píxel para color y 8 para niveles de grises), completos S1 (para médico especialista, aplicando compresión sin pérdidas 3:1), y simplificados S2 (para médico de cabecera junto al diagnóstico en texto).

3. Revisión de mecanismos de TCP

En la evolución del funcionamiento de TCP se han incluido los siguientes algoritmos [21]:

3.1. Slow Start / Congestion Avoidance

Las versiones iniciales enviaban segmentos TCP de forma continua atendiendo sólo al tamaño de ventana en el receptor (*window*). Las siguientes mejoras ya implementan mecanismos para consultar o detectar el estado de la red y adecuar, en consecuencia, el flujo. Así surge *Slow Start* (*SS*) que define la ventana de congestión (*cwnd*) indicador del número de segmentos TCP que se envían. Toma un valor inicial según (1) a partir del tamaño máximo que el emisor pueda enviar *(Sender Maximun Segment Size, SMSS*). Este valor se incrementa al menos en SMSS por cada segmento confirmado (*ACKnowledge, ACK*) desde el receptor, por lo que el tamaño de *cwnd* se puede aproximar a una progresión según (2). Así, dado que el aumento de *cwnd* depende de los ACK recibidos, *SS* controla la comunicación respecto al estado de congestión. En general, dicha congestión se detecta cuando no se recibe ningún ACK en el tiempo esperado o cuando se recibe por multiplicado.

Esto hace necesario implementar otro mecanismo, *Congestion Avoidance* (*CA*), que define un umbral de *SS* (*ssthresh*), de valor inicial alto (64K), indicador del tamaño umbral de referencia para *cwnd*. Mientras *cwnd* ≤ *ssthresh* y no se detecte congestión, actúa SS. Si *cwnd* supera dicho umbral o se detecta congestión, actuará *CA* incrementando *cwnd* de forma lineal según (3). Además, ante congestión, *CA* reinicia *cwnd* según (1) y reduce *ssthresh* según (4), con DnACK el tamaño de los datos enviados que no han llegado al receptor. Tras este proceso, vuelve a actuar *SS* aumentando *ssthresh* en tanto la congestión vaya desapareciendo y *cwnd* sea mayor. Este proceso se resume como:

1. Send one segment (at the beginning, or after *time-out*)
2. For each received ACK, send two segments.
3. Every time a new ACK is received:
   if (cwnd < ssthresh) then
      cwnd+ = 1; /*multiplicative increase*/
   else cwnd+ = 1/cwnd; /*linear increase*/
4. If a timeout occurs (a segment has been dropped)
   ssthresh = cwnd/2; cwnd = 1;

En Fig. 3 se muestra el funcionamiento de *SS* y *CA* en una situación de congestión variante, para entornos hospitalarios con transferencias a ráfagas y acceso múltiple de usuarios, donde se aprecia cómo el sistema ajusta un reparto equitativo de recursos.

$$cwnd[0] = \text{mín}\{2\text{SMSS}, 2 \text{ segmentos TCP}\} \quad (1)$$

$$cwnd[t+1] = 2 \cdot cwnd[t] \quad (2)$$

$$cwnd[t+1] = cwnd[t] + \frac{\text{SMSS}^2}{cwnd[t]} \quad (3)$$

$$ssthresh[0] = \text{máx}\{2\text{SMSS}, \tfrac{1}{2}\text{DnACK}\} \quad (4)$$

$$cwnd[t+1] = \tfrac{1}{2}\, ssthresh[t] + 3\text{SMSS} \quad (5)$$

3.2. Fast Retransmit / Recovery

Cuando el receptor recibe un segmento desordenado, genera un ACK de vuelta para indicarlo al emisor, que puede interpretar dos casos: que el segmento se ha desechado (y, por tanto, lo retransmitirá), o que ha habido reordenación y el esperado está aún por llegar (por lo que el emisor no debe retransmitir salvo que expire el *time-out*). Así, se entiende que el segmento se ha perdido si recibe más de dos ACK duplicados, en cuyo caso no hay que esperar al *time-out* sino que se puede retransmitir en ese mismo momento. Este mecanismo se conoce como *Fast Retransmit* al que se suele añadir la técnica *Fast Recovery*, para entornos de congestión moderada y a ráfagas, en los que se evita bajar *cwnd* al mínimo y entrar en *SS*.

Según *Fast Recovery*, cuando se reciben 3 ACK de un mismo segmento, se detecta que se ha perdido y se reduce *ssthresh* pero no se entra en *SS* sino que se retransmite el segmento solicitado y se adapta *cwnd* según (5) ya que si se reciben 3 ACKs es porque han llegado los 3 SMSS siguientes (única ocasión en que el receptor puede avisar de falta de un paquete). Si se siguen recibiendo ACKs del mismo segmento (debidos a posibles segmentos existentes en la red todavía por llegar al destino), se incrementa *cwnd* en SMSS por cada uno de estos ACK. Así se continua hasta recibir el ACK de un segmento nuevo, que será el retransmitido y hará igualar *cwnd = ssthresh.*

En Fig. 4 se muestra el citado funcionamiento de *Fast Recovery* y se observa cómo tras los 3 ACK el umbral de *ssthresh* cae a la mitad y la ventana actual *cwnd* cae al valor de (5) en lugar de (1), como sería en el caso de no utilizar esta técnica. Así no se entra en *SS* sino que se continúa en *CA* lo que mejora ampliamente las características de la transmisión.

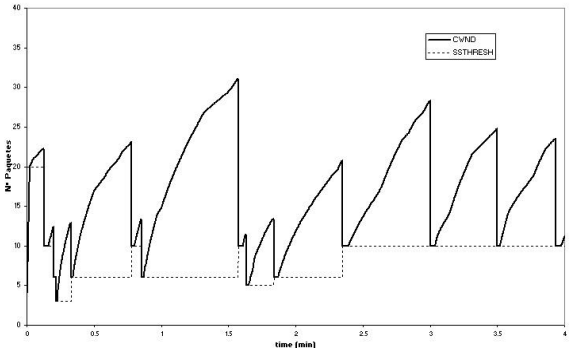

Fig. 3. Ejemplo de evolución de cwnd y sstresh con SS y CA.

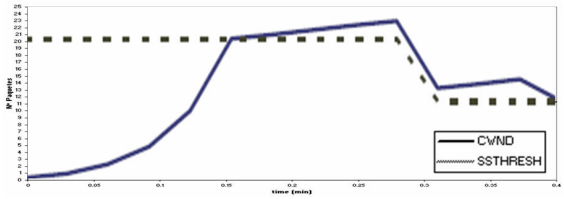

Fig. 4. Funcionamiento de TCP Fast Recovery

4. Revisión de protocolos de TCP

Se presenta una breve explicación teórica de cada versión de TCP discutidas en este estudio:

- **TCP *Berkeley***, una las primeras versiones, disminuye la complejidad ya que intervienen los dos extremos de la transmisión pero no ofrece buenas prestaciones en redes heterogéneas o en congestión, ya que sólo tiene en cuenta el tamaño de ventana en el receptor, pero no el estado de red.
- **TCP *Tahoe*** [22] (incluida en NET/1), soluciona el problema detectando el estado de la red añadiendo los algoritmos *SS* y *CA* y las variables *cwnd* y *ssthresh*. También incluye *Fast Retransmit* para retransmitir segmentos perdidos más rápido, sin esperar a que *timeout* expire y se hayan tenido que recibir multitud de ACK indicativos.
- **TCP *Reno*** [23] y la posterior NET/2, añaden a Tahoe el algoritmo *Fast Recovery*, que evita en lo posible que *cwnd* caiga a 2 y actué *SS* para redes con congestión moderada y a ráfagas. La mejora de NET/3 añade soporte para tráfico *multicast* y extensiones para *long-delay path* [24], buscando superar el límite máximo de 64K (un *handicap* en redes de alto retardo y ancho de banda, que prefieren mandar pocos segmentos grandes por la fuerte incidencia del *Round Trip Time*, RTT [25]).
- **TCP *New Reno*** [26] intenta solucionar los dos principales problemas de *SS* y *CA*: si *ssthresh* es pequeño, *SS* actúa poco, el aumento de *cwnd* se realiza en *CA*, y resulta muy lento; si *ssthresh* es muy alto, el aumento de *cwnd* se realiza muy rápido en *SS*, pero propicia la posible congestión de red. Dado que las sesiones TCP suelen ser cortas, el valor inicial de *ssthresh* y el proceso *SS* son determinantes. Así, esta versión propone buscar el tamaño inicial óptimo de *ssthresh* según el estado inicial que presente la red [27]-[28]. Esto se consigue con el algoritmo *Packet Pair* [29] en que emisor envía series de 2 paquetes conociendo su intervalo temporal. Al recibir sendos ACK, según cada retardo, es capaz de conocer el estado de la red y alcance de la congestión, etc. Con estos parámetros obtiene *ssthresh*[0] óptimo y establece la "fase de retransmisión rápida", equivalente a *Fast Retrasmit/Recovery* para más de 1 segmento de la misma ventana de datos que se han perdido.
- **TCP *SACK*** [30] implementa todo lo anterior e innova en redes muy extensas con tráfico elevado y pérdidas excesivas y continuadas. *Selective ACK* (SACK) funciona cuando el receptor recibe datos contiguos tras la pérdida de un segmento. Entonces envía un ACK duplicado (DUACK) indicando qué segmentos han sido correctamente recibidos. Así, un triple ACK indica un dato perdido y el emisor sabe exactamente qué segmentos han llegado bien y cuáles retransmitir. SACK permite un envío continuo y simultáneo a las retransmisiones siempre que este volumen de datos estimado sea menor que la ventana *cwnd*.
- **TCP *Vegas*** [31], es una de las últimas propuestas al clásico TCP. Consiste en un diseño experimental en el que el emisor podría detectar por anticipado una situación de congestión, comprobando en cada instante la diferencia entre la tasa de transferencia potencial (que espera enviar) y real (que consigue efectivamente). Con esta anticipación se propone transmitir una cantidad de segmentos que mantenga en régimen más o menos permanente un número pequeño de paquetes en los *buffers*, otorgándoles protagonismo en el funcionamiento del protocolo. Esto puede tener dos problemas potenciales: TCP "*per se*" funciona a ráfagas, lo que permite acceso múltiple y reparto de recursos. El intento de Vegas de converger a un flujo continuo podría ser inestable o alejarse de la equidad (*best-effort*). Por otro lado, Vegas podría no ser efectivo en situaciones de congestión permanente.

Las investigaciones actuales buscan mantener anchos de banda estables y equitativos con *buffers* suficientes. En caso contrario, se proponen políticas como Sack o NewReno. Aún así, como Vegas da implícito el problema de generar colas persistentes, en paralelo, se estudian políticas de gestión de *buffers* para implementar conjuntamente, como:

- **SFQ** [32], (mejora de FQ) propone colas distintas para cada flujo de datos, atendiendo cada paquete según algoritmos de turno circular *Round Robin* y garantizando retransmisiones en orden equitativo. El número de *buffers* es fijo y una función *hash* asocia a cada flujo una cola para atender todos los flujos potenciales simultáneos estadísticamente.
- **DRR** [33] (mejora computacional de WFQ) soluciona el problema del anterior para casos en que los tamaños de los paquetes no son fijos ya que ahora Round Robin no reparte por paquete sino por tamaño; así, transmite equitativamente la misma información por unidad de tiempo y flujo.
- **RED** [34], detecta por anticipado la congestión, monitorizando el tamaño medio de cola. Mantiene un número no muy grande de paquetes y permite limitadas ráfagas más o menos intensas. Al conocer la saturación en sus colas puede informar del estado de congestión incluso sin necesidad de eliminar paquetes, sino sólo marcándolos.
- **CBQ** [35] propone crear clases de flujos y priorizar unos más que otros asignando más o menos porcentaje de capacidad a cada uno. Esta mejora es la única que garantiza que la congestión de un tráfico no afecte al resto y, por tanto, se puedan asegurar requisitos de QoS propios para cada Tipo de Servicio (*Type of Service*, ToS).

En resumen, la detención anticipada de congestión y el reparto adaptado de los recursos según los tipos de servicio, parecen las estrategias a seguir [36]. Con esa idea, este trabajo valora las mejores opciones para heterogéneos entornos hospitalarios que soportan los nuevos servicios de telemedicina.

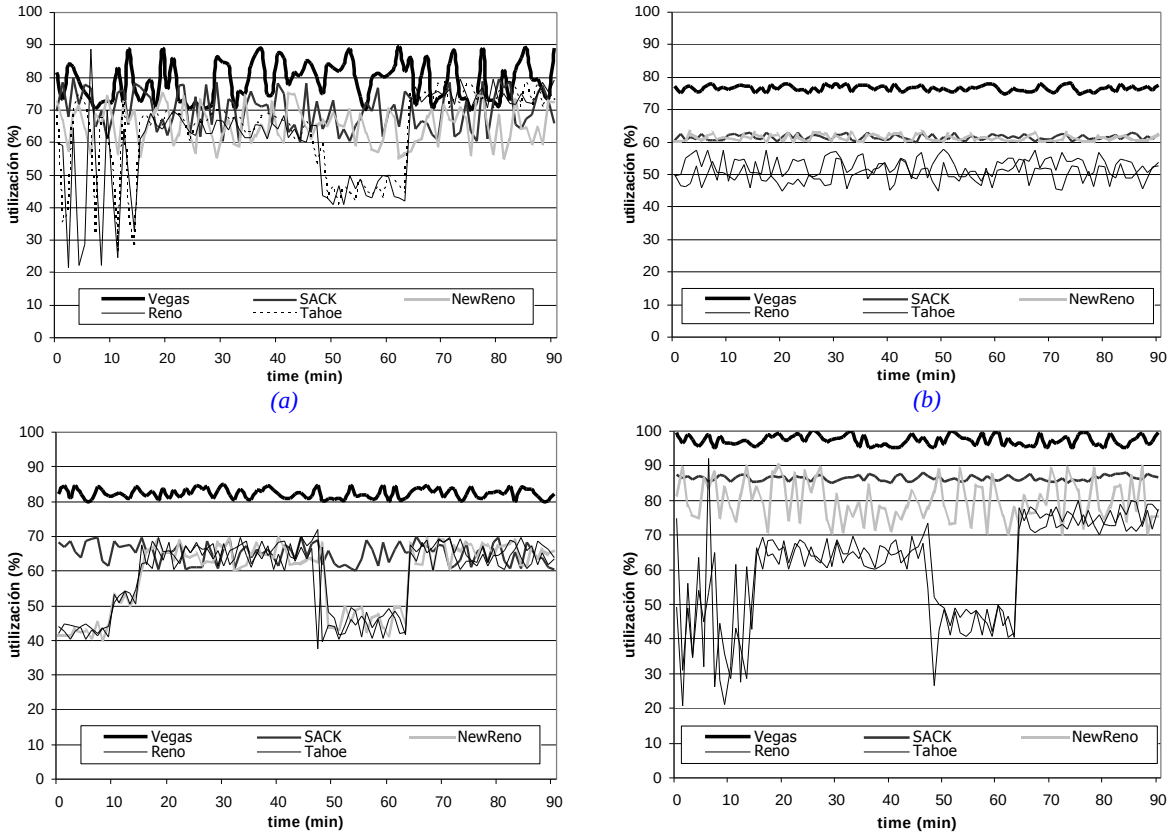

Fig. 5. Comparativa de r para Vegas, SACK, NewReno, Reno y Tahoe con política (a)SFQ. (b)DRR. (c)RED. (d)CBQ.

5. Resultados y discusión

A partir del escenario planteado en la sección 2, en primer lugar se analiza el factor de utilización (r) de los *buffer* intermedios y, por tanto, cómo aprovecha los recursos cada técnica de TCP para las disciplinas de servicio revisadas. En Fig. 5(a), SFQ asocia a cada cola un flujo de tráfico pero, al haber sólo dos *buffers* (TCP y UDP, ya que más división haría compleja su implementación real) hace un reparto probabilístico demasiado equitativo. En Fig. 5(b), DRR pesa más proporcionalmente los distintos tráficos y elimina los saltos bruscos de r en las situaciones de congestión, a cambio de disminuir la utilización media del *buffer*. En Fig. 5(c), RED se adapta al SMSS y mejora a los anteriores, sobre todo en el caso FTP2 basado en VBR (zona B en Fig. 2), pero falla en la asignación proporcional de la capacidad según la simultaneidad de los flujos de datos (zonas A, D y E en Fig. 2). En Fig. 5(d), CBQ da las mejores tendencias ya que hace un reparto clasificado por tipo de servicio, y se ajusta a la heterogeneidad planteada. En todos los casos, Sack y Vegas dan los mejores resultados ya que intentan mantener el flujo continuo de datos. Incluso parece mejor Vegas en el escenario propuesto ya que tiende a detectar anticipadamente la congestión y permite sólo un número reducido de paquetes en cola, haciendo que el *buffer* se ocupe casi continuamente al 100%. Por todo ello, de aquí en adelante se eligen ambos tipos de TCP como mecanismos de control, y CBQ como disciplina a configurar en los *buffers*.

Con estas premisas, se evalúan los parámetros de interés para optimizar el servicio: ancho de banda (BW), retardo en lazo cerrado (RTT) y los valores de la ventana de congestión (*cwnd* y *ssthresh*). Como lo que se pretende es ver en qué situaciones es más propicio cada mecanismo TCP, se simula con diferentes tamaños de *buffers* (q, en número de segmentos SMSS) según los valores iniciales, tanto de SMSS como de *cwnd* y *sstresh.* También se incluyen diferentes niveles de congestión según las recomendaciones de la *Unión Internacional de Telecomunicaciones* (UIT) [37] para tasas de pérdidas (*Packet Loss Rates*, PLR). Se analizan en este trabajo los resultados más significativos para situaciones cruzadas de congestión baja-leve ($PLR<0.03$) o moderada-alta ($PLR<0.10$), y tamaños de *buffer* medios ($q=10$) o altos ($q=30$).

En Fig. 6 se muestra el reparto de los recursos disponibles para cada tipo de servicio planteado en este estudio: transferencias médicas (prioritarias en FTP1 y no prioritarias en FTP2), consultas a bases de datos (WWW) y acceso remoto a equipos (Telnet). Se aprecia cómo los tráficos FTP1 y WWW, más prioritarios, se complementan inicialmente (zona A en Fig. 2). Sin embargo, dicho reparto no es equitativo, sino que varía significativamente según la versión del protocolo utilizada. En el caso de Vegas, FTP1 consigue más recursos por anticipado y, por tanto, restringe e BW dedicado al servicio WWW.

Esto es debido, como se ha comentado, a que es tráfico a ráfagas y el protocolo está diseñado para mantener un flujo de datos constante. Al contrario, en el caso de utilizar SACK, este rafagueo se adecua con más precisión ya que se adapta mejor a incrementos bruscos en los recursos disponibles.

También puede destacarse el comportamiento para FTP1 en ausencia de FTP2 menos prioritario (zona E en Fig. 2) en el que SACK presenta una pendiente mayor, mientras que Vegas crece más lentamente por su tendencia a tener un número reducido de paquetes en los nodos. En cualquier caso, esta tendencia no es realmente significativa e, incluso en las situaciones planteadas de mayor congestión en las que el uso de Vegas podría ser cuestionable, las curvas mostradas en Fig. 6(b) y 6(d) son muy similares. Esto respalda la elección de Vegas dado que su comportamiento es mejor para el resto de casos y, además, a ello se suma el propio de CBQ que enfatiza la diferencia entre ToS, priorizando los que presentan mayores restricciones para cumplir los requisitos de QoS.

Así, aunque la conjunción Vegas-CBQ parece adecuada en el escenario propuesto, habría que garantizar su correcto funcionamiento en casos de congestión permanente o poca capacidad de *buffers*, por lo que se mantienen ambos métodos en la evaluación del resto de parámetros.

A continuación se muestra en Fig. 7 el análisis del parámetro RTT para el servicio FTP1, dado que en el resto de los casos las tendencias son equivalentes. Se presentan de nuevo los casos de q y PLR planteados para el análisis anterior.

Para las situaciones de baja congestión, ver Fig. 7(a) y 7(c), en el caso Vegas se constata que son menores los valores estimados para RTT y, también, que las retransmisiones finalizan antes. Igualmente ocurre si se incrementa el nivel de congestión, ver Fig. 7(b) y 7(d), que Vegas mantiene su mejoría. Igualmente, para los casos de tamaños pequeños de *buffer*, q=10 en Fig. 7(c) y 7(d), Vegas responde muy bien a las exigencias manteniendo siempre pocos paquetes en cola y disminuyendo el RTT < 100ms. Esto permitiría garantizar que las transmisiones FTP cumplen los requisitos de QoS y podría concluirse que el protocolo Vegas aporta un mejor comportamiento global para el servicio.

Finalmente se muestra en Fig. 8 la evolución de *cwnd* y *sstresh*, para los dos casos más restrictivos del estudio con *q=30, PLR<0.03* y *q=10, PLR<0.10*. En el primer caso para *q*=30, se aprecia de nuevo que la retransmisión del tráfico FTP finaliza antes en Vegas, ya que no se producen descartes de paquetes pues *cwnd* es muy inferior que el límite *q* propuesto. Sack sí descarta paquetes ya que *cwnd* > 30 en varias ocasiones. Para el segundo caso de más congestión, se aprecia en Fig. 8(b) para SACK cómo *cwnd* sobrepasa continuamente el tamaño del *buffer*. Esto produce descartes, se activa el protocolo, y cae el valor de *cwnd*. Esta situación genera retransmisiones que sobrecargan el enlace innecesariamente. Vegas, que podría pensarse peor en situaciones de pérdidas elevadas, se adapta mejor a los umbrales por su tendencia a reducir el número de paquetes (aproximadamente la mitad que en el caso de SACK), como se aprecia en la comparativa.

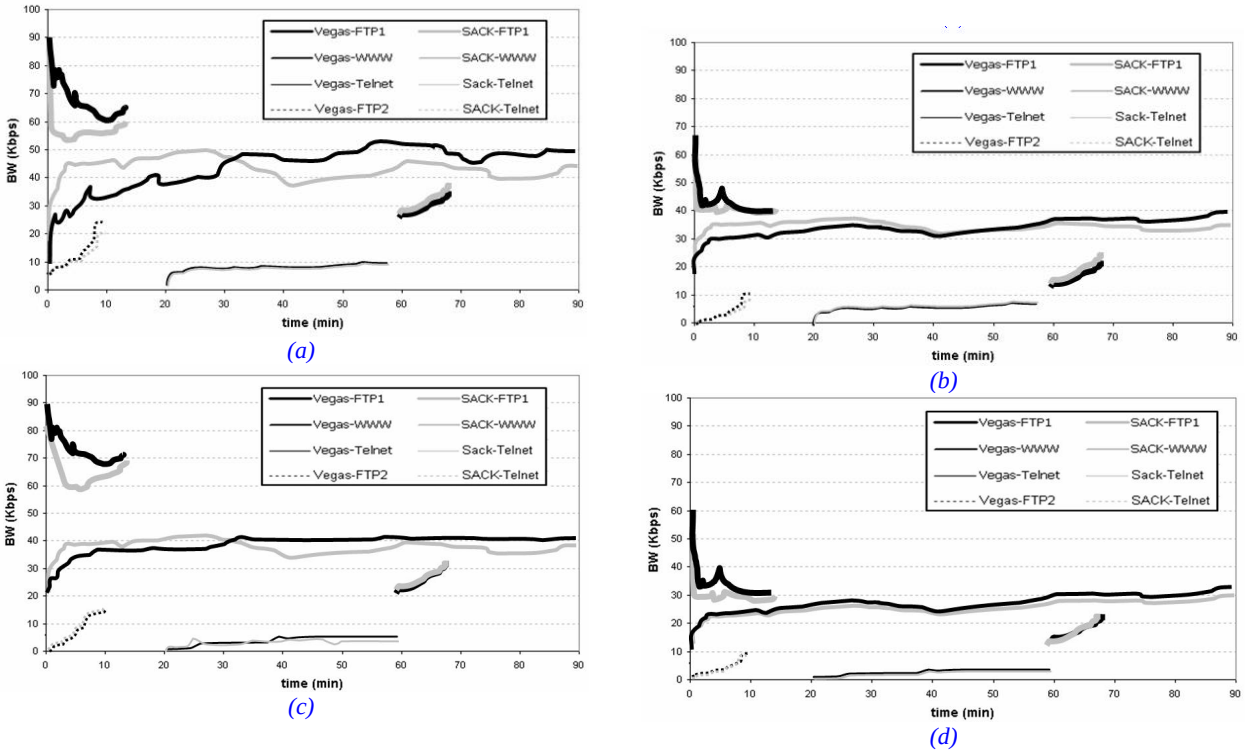

Fig. 6. Comparativa de BW para (a) q=30, PLR<0.03 (b) q=30, PLR<0.10 (c) q=10, PLR<0.03 (d) q=10, PLR<0.10.

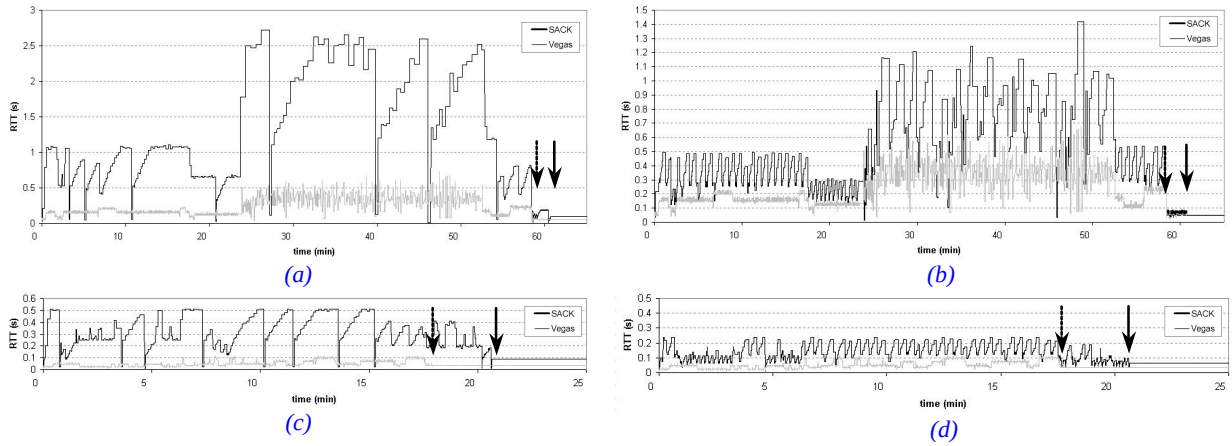

Fig. 7. Comparativa de RTT para (a) q=30, PLR<0.03 (b) q=30, PLR<0.10 (c) q=10, PLR<0.03 (d) q=10, PLR<0.10

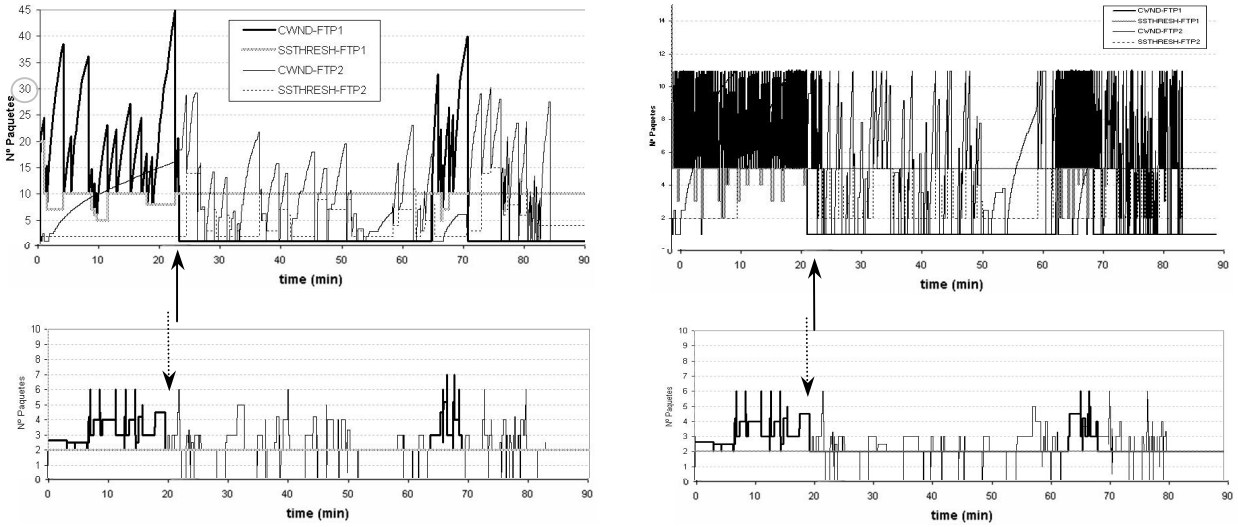

Fig. 8. Evolución de cwnd y sstresh para SACK (superior) y Vegas (inferior) según (a) q=30, PLR<0.03 (b) q=10, PLR<0.10

Finalmente, se pretende evaluar no sólo los distintos niveles de congestión sino también las fluctuaciones en la capacidad de red. Se ha considerado un CIR variable de n·64Kbps (hasta ahora el valor elegido era 256Kbps) y, manteniendo las condiciones más restrictivas de los casos anteriores, se ha simulado con múltiples valores de n.

Se presentan en Fig. 9 las situaciones más representativas para los casos de CIR=384Kbps (a), y CIR=512kbps (b). En el primer caso, se puede apreciar cómo el tiempo de transmisión se reduce considerablemente al aumentar el CIR, como se observa en las flechas indicativas de Vegas y SACK. En la evaluación también se observó que los valores de *cwnd* siguen siendo muy pequeños y que la estimación de RTT es óptima para todos los tipos de tráficos asociados a cada servicio.

Aunque este caso ya podría cubrir las necesidades planteadas, se presentan en Fig. 9(b) los últimos resultados obtenidos para dar una idea de la mejora que conlleva un incremento en el ancho de banda. Así, se sigue apreciando un comportamiento similar al caso anterior, con retardos significativamente menores que cumplen perfectamente los requisitos establecidos de QoS.

De todo ello se constata que Vegas es capaz de mantener un número muy pequeño de paquetes en los nodos y, aún así, seguir aprovechando los recursos disponibles, gracias a su anticipación a la congestión, complementado por la política CBQ de clasificación y priorización de tipos de servicios.

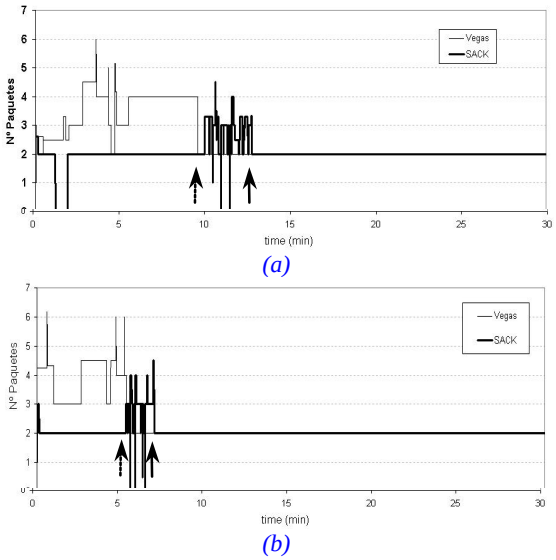

Fig.9. Evolución de cwnd y sstresh según (a)CIR=384Kbps. (b)CIR= 512Kbps.

6. Conclusiones

En este trabajo se han estudiado las distintas versiones de TCP y de disciplinas de prioridades aplicadas al tráfico hospitalario asociado a los nuevos servicios de telemedicina. Se ha revisado el estado del arte en el ámbito de los métodos de control de congestión. Y se han planteado escenarios de simulación que incluyen modelos avanzados de tráfico y red, y situaciones de simultaneidad representativas de telemedicina en entornos rurales o geográficamente dispersos.

Siguiendo una metodología de evaluación técnica desarrollada en trabajos anteriores, se han obtenido los mejores resultados para el protocolo TCP-Vegas en conjunción con disciplinas de colas CBQ en términos de tiempo de transmisión, y de utilización de los *buffers* intermedios, de enorme utilidad para tráficos simultáneos compartiendo los recursos, como es el caso de los escenarios planteados.

Para los casos de variaciones bruscas en el ancho de banda, se ha visto un mejor aprovechamiento de los recursos por parte del protocolo Sack que, a su vez, son contrarrestadas por Vegas si se da congestión o si disminuye el tamaño disponible en los *buffers*. En cualquier caso, dado que Vegas se encuentra en fase de estudio y no está tan extendido como el resto de versiones, en futuros trabajos sería interesante estudiar su implementación práctica y seguir evaluando Sack, ampliamente aceptado y utilizado en redes de propósito general como Internet.

Agradecimientos

Este trabajo ha recibido el apoyo de proyectos de la Comisión Interministerial de Ciencia y Tecnología (CICYT) y de los Fondos Europeos de Desarrollo Regional (FEDER) TSI2004-04940-C02-01, y de los Fondos de Investigación Sanitaria (FIS) FISG03/117. Los autores desean agradecer a Santiago Estela Pérez-Aradros su colaboración técnica en este artículo.

Referencias

[1] V. Jacobson, "Congestion Avoidance and Control". *ACM Computer Communications Review*, 18(4):314–329, 1988.

[2] T. Bonald, "Comparison of TCP Reno and TCP Vegas: Efficiency and Fairness", *Proc. PERFORMANCE,*1999.

[3] U. Hengartner, J. Bolliger and T. Gross, "TCP Vegas Revisited", *Proc. IEEE INFOCOM*, 2000.

[4] M. Gerla, R. Lo Cigno, S. Mascolo and W. Weng, "Generalized Window Advertising for TCP Congestion Control", *CSD-TR990012*, 1999.

[5] L. Kalampoukas, A. Varma, and K.K. Ramakrishnan, "Explicit Window Adaptation: A Method to Enhance TCP Performance", *Proc. IEEE INFOCOM*, 1998.

[6] T. Goff, J. Moronski, D. S. Phatak and V. Gupta, "Freeze-TCP: a True End-to-end TCP Enhancement Mechanism for Mobile Environments", *Proc. IEEE INFOCOM*, 2000.

[7] D. Caramella and S. Giordano "An advanced IP based telemedicine trial supporting QoS for multimedia teleconsulting", *International Conference EuroPACS,*2000.

[8] W.R. McDermott et al., "Optimization of Wide-Area ATM and Local-Area Ethernet/FDDI Network Configurations for High-Speed Telemedicine Communications Employing NASA's ACTS", *IEEE Network*, 13(4):30-38, 1999.

[9] S. Ubik and J. Klaban,"Experience with using simulations for congestion control research", *Cesnet Technical Report*, 26/2003.

[10] J.C. Hoe. "Improving the Start-up Behavior of a Congestion Control Scheme for TCP", *Proc. ACM SIGCOMM*, 1996.

[11] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput", *Proc. SIGCOMM*, 2002.

[12] W.R. Stevens. B. Dempsey, J. Liebeherr and A. Weaver, "On Retransmission based Error Control for Continuous Media Traffic in Packet-switching Networks", *Computer Networks and ISDN Systems*, vol. 28, no. 5, 1996.

[13] C. Zhang and V. Tsaoussidis, "TCP-Real: Improving real-time capabilities of TCP over Heterogeneous Networks" *Proc. NOSSDAV-International Workshop on network and operating systems support for digital audio & video*, 2001.

[14] I. Martínez J. García and J. Fernández. "QoS Evaluation Methodology for Multimedia Telemedicine Services". *IEEE Transactions on Multimedia*, submitted, 2005.

[15] NS-2. Network Simulator, http://www.isi.edu/nsnam/ns/ns-documentation (v. 2) http://www-mash.cs.berkeley.edu/ns.

[16] A. Reyes. "Modelado de tráfico de clientes WWW", *PhD Thesis*, Universidad de Málaga, 2001.

[17] M.E. Crovella and A. Bestavros, "Self-Similarity in WWW. Evidence and Possible Causes", *IEEE/ACM Transactions on Networking*, 5(6): 835-846, 1997.

[18] I. Martínez, J. Salvador, J. Fernández, J. García, "Traffic requirements evaluation for a Telemedicine network", *International Congress on Computational Bioengineering ICCB'03*, pp. 389-394, 2003.

[19] F.J. Martón y J. García, "Diseño de una red telemática global para el sistema de centros sanitarios de Aragón", *XX Congreso Anual de la SEIB*, pp. 389-392, 2002.

[20] K. Fall, "Network emulation in the VINT/ns simulator", *IV IEEE Symposium on Computers and Communications (ISCC'99*), pp. 59-67, 1999.

[21] W.R. Stevens, "TCP Slow Start, Congestion Avoidance, Fast retransmit and Fast recovery Algorithms", *RFC-2001*, 1997. www.ietf.org/rfc/rfc2001.txt. Last access 30/03/05.

[22] W.R. Stevens, "TCP/IP Illustrated", *Addison-Wesley*, 1994.

[23] M. Allman, V. Paxson, W. Stevens, "TCP congestion control", *RFC-2581*, 1999. http://www.ietf.org/rfc/rfc2581.txt. Last access 30/03/05.

[24] V. Jacobson and R.T. Braden, "TCP extension for long-delay paths", *RFC-1072*, 1998 http://www.ietf.org/rfc/rfc1072.txt. Last access 30/03/05.

[25] P. Karn and C. Partridge. "Estimating Round-Trip Times in reliable transport protocols", *Proceedings SIGCOMM*,1987.

[26] S.Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm", *RFC-2582*, 1999. http://www.ietf.org/rfc/rfc2582.txt. Last access 30/03/05.

[27] M. Allman, S. Floyd and C. Partridge, "Increasing TCP's initial window", *RFC-2414*, 1998. http://www.ietf.org/rfc/rfc2414.txt. Last access 30/03/05

[28] K. Poduri, K. Nichols, "Simulation studies of increased initial TCP window size", *RFC-2415*, 1998. http://www.ietf.org/rfc/rfc2415.txt. Last access 30/03/05.

[29] Srinivasan Keshav, "Packet Pair Flow Control", AT&T Bell Laboratories, 1999.

[30] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgement option", *RFC-2018*, 1996. http://www.ietf.org/rfc/rfc2018.txt. Last access 30/03/05.

[31] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance", *Proc. ACM SIGCOMM '94*, 1994.

[32] Y. Afek, Y. Mansour and Z. Ostfeld "Space Efficient Fair Queuing by Stochastic Memory Multiplexing" Computer Science Department. Tel-aviv University (Israel), 1998.

[33] A. Kantawala, "Queue State Deficit Round Robin (QSDRR)", *Applied Research Lab,Washington Univ*, 2002.

[34] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance". *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397-413, 1993.

[35] F. Risso, "Decoupling bandwidth and delay properties in Class Based Queuing". Dipartimento di Automatica e Informatica, Politecnico di Torino (Italia).

[36] Stephen H. Low, "A duality model of TCP and queue management Algorithms", IEEE/ACM Trans, 11 (4), 2003.

[37] D. Wright, "Informe UIT sobre telemedicina en los países en desarrollo", *Journal of Telemedicine and Telecare* 4(1), 1998 [En español, *International Telemedicine*, 7-8, 1998].