A Simulation Study of TCP Performance over UMTS Downlink

B. J. Prabhu*, E. Altman*, K. Avrachenkov*, J. Abadia Dominguez†
* INRIA, 2004 Route des Lucioles, 06902 Sophia Antipolis Cedex, France.
Email: {bprabhu, altman, k.avrachenkov}@sophia.inria.fr
† France Telecom R&D, 38, rue du General-Leclerc, 92794 Issy-les-Moulineaux Cedex 9, France.
Email: javier.abadia@rd.francetelecom.com

***Abstract*— Data transfer on the UMTS downlink can be done either through dedicated channels (DCH) or shared channel (FACH). The data transfers on the DCHs can receive very high throughputs. However, a setup time of the order of 250ms is required before data transfer can begin on the DCH. For very short transfers it may be better to use the shared channel. However, for long data sessions it might be better to allocate a dedicated channel. In this paper, we propose a threshold policy to determine which sessions should use the dedicated channels. Initially, we use the FACH for a given connection. Then if we get an indication that the current burst might be long (for example, we observe a long queue from that source), then beyond some threshold we shall try to allocate a DCH to that connection (if there is one available). We also present and evaluate the performance of algorithms in which a timer is set when the queue size falls below a threshold and the connection is on DCH. The use of timer allows a connection to remain for a longer duration on DCH and thus improve performance. We use ns-2 simulations to obtain the performance measures for the threshold and timer based policies.**

I. Introduction

UMTS is the 3^{rd} generation of cellular wireless networks which aims to provide high speed data access along with real time voice calls [1]. On the UMTS downlink, data can be transferred through dedicated channels (DCH) or a shared channel (FACH). Dedicated channels offer higher transfer speeds but require a setup time which is significant (of the order of 250ms). Shared channels, on the other hand, have a low setup time and also low transfer speeds. For sporadic packets (i.e., files or transfers that are very short) it is efficient to use the shared channel. However, for long data sessions it may be better to allocate a dedicated channel. Most data transfers over the Internet are performed today with TCP. Data connections on the Internet are known to be bursty where the basic burst size is best described with a Pareto distribution:

$$Pr[burst\ size > s] = (k/s)^{\beta},$$

where k is the minimum size and $1 < \beta \leq 2$ (for Internet traffic) [2]. We use the traffic description of Appendix B of [3] which is the standard for UMTS data structure. It is also based on the Pareto distribution for the transfer size. Our starting point is the observation that long bursts are in some sense "predictable": if we observe that the current burst has already transfered many packets, say x, then the distribution of the remaining size of the burst $R = burst\ size - x$ is given by

$$Pr(R > x + s | R > x) = \left(\frac{x}{x+s}\right)^{\beta}$$

so for a fixed value of s, as x increases (to infinity), $Pr(R > x + s | R > x)$ increases to 1. (This is a general property of heavy tailed distributions [4]). Therefore, it seems natural to implement the following policy: start using the FACH for a given connection if a DCH is not available. Then if we get an indication that the current burst might be long (for example, we observe a long queue from that source), then beyond some threshold we shall try to allocate a DCH to that connection (if there is one available). When the queue size of a connection falls below a threshold we switch it back to FACH. We also present and evaluate the performance of a modified threshold algorithm. In the modified threshold algorithm, instead of switching immediately when the queue size falls below a threshold, a timer is set and the connection remains on the DCH for this period. If there are no new arrivals within this timeout period, the connection is switched back to FACH. The timer is used in order to allow the ACKs to reach the TCP source and new packets to be released by the source.

The rest of the paper is organized as follows. In Section II, we present the network configuration used in the simulations and describe the methods used in the implementation of the code. In Section III, we describe the source traffic model, the simulation parameters and give the values used in the simulations. In Section IV, we present the results of the threshold algorithm. In Section V, we describe the modified threshold policies wherein a timer is used along with the threshold on the DCH. In Section VI, we present the results of the modified threshold policies and compare them with the results of the original threshold policy. In Section VII, we present the conclusions obtained from this work.

II. Network Model

We consider a network model with N_{tcp} TCP sources which need to send data to mobile receivers. We assume a single cell scenario with one base station and several mobile stations which act as destinations for TCP traffic. The TCP sources are assumed to be connected to the base station of the cell with a high speed (5mbps, 30ms) link. The base station can transfer data from a TCP source on either DCH or FACH at a given time. There is one FACH and N_{dch} DCHs in the system. The FACH is a time division multiplexed channel. In addition to any TCP connections which may be present on a FACH, there is signaling traffic which must be transmitted on the FACH. The signaling traffic has priority over the TCP connections. During the silence periods of the signaling traffic, data from one or more TCP connections can be transmitted on the FACH. On the FACH, data from the TCP connections is assumed to be transmitted in a round- robin fashion. If all the DCHs have a TCP connection configured, a connection on DCH should be first switched to FACH before a session from FACH can be switched on to a particular DCH. This means that a switch can take up to 500ms (if there is already a TCP connection configured on the DCH).

Switching from one channel to another is costly in time and in signaling. In the model we assume that there exists a queue corresponding to each TCP connection in the base station. The base station is hence able to track the queue length of each connection. During the switching time (of around 250ms) from one channel to another, no packets from the queue of the TCP connection being switched can be transmitted. We, therefore, try to avoid quick switching and propose the following channel allocation policy. There are two thresholds T_h and T_l. If the number of packets in the queue of a connection using FACH reaches T_h and a DCH is available then we initiate a switch to the DCH. If the number of packets in the queue of a connection using DCH drops below T_l and the number of allocated DCHs equals the N_{dch}, then we switch the connection back to FACH. The simulation setup for the network is presented in Fig. 1.

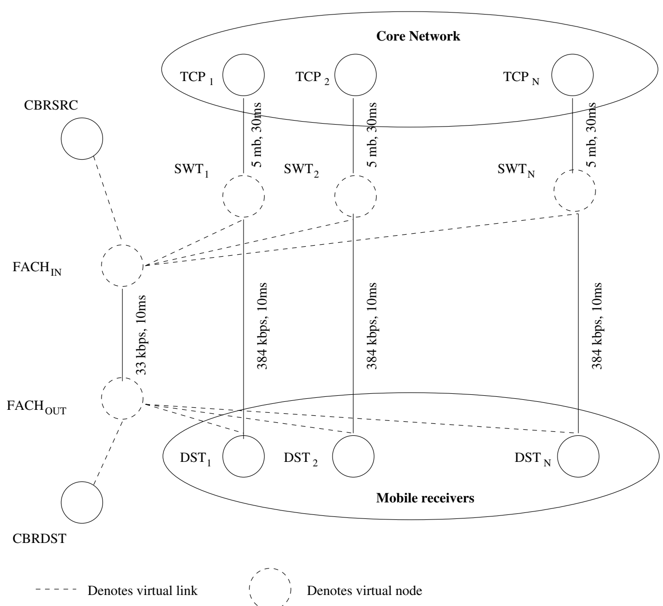

Fig. 1. Simulation Setup

Each TCP source node, TCP_i is connected to a routing node called Switch (SWT_i). SWT_i is present inside the base station and can be connected either to the $FACH_{IN}$ or directly to the TCP destination via the DCH. The SWI_i node is created to simplify the simulations and may not be present inside a real base station. The $FACH_{IN}$ is another virtual node which simulates the prioritized round robin service discipline taking place on the FACH. This is a service discipline which we had to implement. In this discipline, the node $FACH_{IN}$ gives priority to the traffic from CBRSRC while serving the packets from the SWI_i's (only those which are currently not transmitting on DCH) in a round-robin manner. We note that there are no queues at $FACH_{IN}$ and all the packets are either queued at SWT_i or at the $CBRSRC$. The $CBRSRC$ simulates a constant bit rate source of signaling/control traffic. It generates packets at rate R_{sig} and is assumed to be present within the base station. The signaling traffic flows from the base station to the mobile receivers. Even though we model the destination of the signaling traffic ($CBRDST$) as one node different from DST_i, we note that it does not affect the simulations as simultaneous transfer of data and control packets to the same mobile receiver is possible when different channels are used. The links $SWI_i - FACH_{IN}$ and $CBRSRC - FACH_{IN}$ are virtual links within the base station and thus have zero delay. We note that the data from SWT_i to DST_i can take two different routes i.e., $SWI_i - FACH_{IN} - FACH_{OUT} - DST_i$ (via FACH) or $SWT_i - DST_i$ (via DCH). At any given time only one route from the above two can be active. Although in the simulation scenario we have as many DCH links as TCP source nodes, the simulation allows us to connect not more than N_{dch} DCH channels at a time, which may be chosen strictly smaller than the number of TCP sources (N_{tcp}). In the simulations we switch over from FACH to DCH by changing the cost of the links and recomputing the routes. This is done as follows: Initially, the cost of the direct path from the Switch to the TCP destination is set to 10 and the cost of all other links to 1. Hence, the traffic gets routed through the FACH. When a switch is desired, the cost of the DCH is set to 1 and the routes are recomputed. This activates the DCH and the traffic gets routed on the DCH.

III. Input for the simulation

We use ns-2 [5] for our simulation study with the following parameters:

- Number of available DCH channels (N_{dch}) is taken to be 1. The number of simultaneous TCP connections, N_{tcp} varies between 2 to 8.
- Values of threshold T_h for switching between channels is varied between 0 and 15. A threshold value of 0 indicates that we try to switch a connection to DCH as soon as it arrives.
- The duration of the simulation is taken 200000 secs in order to reach stationarity. The large time needed to get good accuracy is partially due to the nature of the long range dependence of the traffic.
- The time it takes to switch between channels (D_{sw}) is 250ms.
- We consider the background (non TCP traffic source) traffic source, that uses the FACH, to be CBR source with rate $R_{sig} = 24$ kbps. It sends a 1kB packet at an interval of $1/3$secs. It has non pre-emptive priority over TCP.

TABLE I

NETWORK PARAMETERS

N_{dch}	1
R_{dch}	384 kbps
R_{fach}	33 kbps
R_{sig}	24 kbps
T_l	1
D_{sw}	.25 s

TABLE II

TRAFFIC SOURCE PARAMETERS

T_{on}	$0.3s$
T_{off}	$5s$
P_{off}	0.33
FS_{avg}	30 kB
k	1.1
Pkt_{size}	320 B

- The TCP connection traffic model is as follows: On a TCP connection, data arrives in bursts. The number of packets in a burst has a Pareto distribution. The shape parameter is generally taken to be between 1.04 and 1.14 [2], [6]. We take the shape parameter to be $k = 1.1$ [3] and the average file (burst) size is taken to be $FS_{avg} = 30kB$ [2]. We do not take all the parameters from [3] as they are not adapted to TCP traffic. The model in [3] considers packets of size 81.5 bytes whereas TCP does not send short packets [8]. A TCP connection alternates between "ON" and "OFF" states. In the ON state, the interarrival time between successive bursts is exponentially distributed with mean T_{bur}. At the end of each burst, the connection goes into OFF state with probability P_{off}. It remains in the OFF state for an exponentially distributed duration with mean T_{sil}. At the end of this duration the connection goes to ON state which is marked by the arrival of a burst. Fig. 2 shows the traffic model used at each source node.
- The TCP packet size is taken to be 320 bytes which is of the same size as a RLC data segment in UMTS [9].
- The schedule of service of packet in the FACH is taken to be round robin. The implementation in ns-2 is not standard and we had to change some link element features of ns-2 in order to implement this schedule.

Tables I and II give the values of the network parameters and traffic parameters, respectively.

IV. SIMULATION RESULTS I

Fig. 3 shows the average burst delay versus T_h for different number of TCP connections, N_{tcp}. For a given value of N_{tcp}, it is observed that initially the delay reduces and then increases with increasing values of T_h. This suggests an optimal value of the threshold at which the delay is minimum. At higher values of T_h an increase in the burst delay is observed because a higher value of T_h implies more time is spent in the FACH. The FACH is a low bandwidth channel which has

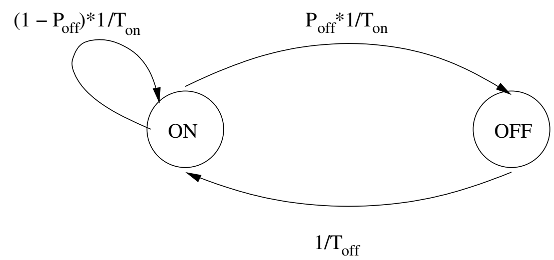

Fig. 2. Traffic source model

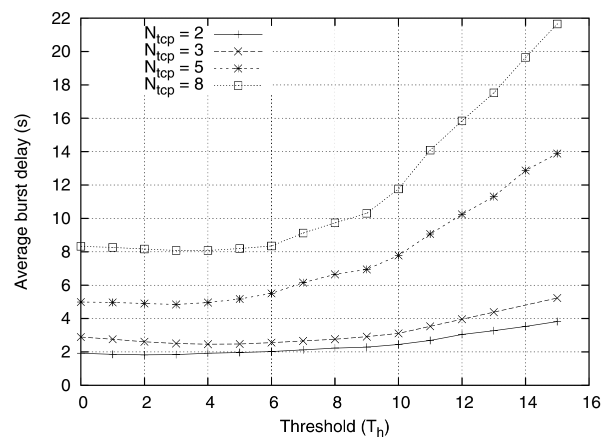

Fig. 3. Average Delay vs. T_h. $N_{dch} = 1$, $D_w = 250ms$.

high priority signaling traffic on it. This results in low average bandwidth being shared amongst the TCP connections. For a TCP connection, the switch to DCH is based on its the current buffer size which in turn depends on its current window size. The current window size is incremented whenever an ACK is received by the sender. When a TCP connection is on a low bandwidth link, the window builds up slowly due to delay in receiving an ACK. This slow buildup of the window size results in slow buildup of the current buffer size. As the value of T_h is increased, a TCP connection has to spend more time on the slow FACH before it builds up its buffer size resulting in a higher delay.

We also observe a decrease in the burst delay for small values of T_h. This dip is observed due to the high speed of the DCH and the delay before ACKs can reach the source. Before the source can release more packets to the DCH queue it must receive an ACK from the TCP_{dest}. However, there is a link delay and transmission delay before the ACK is received by the source. If, in this period, the DCH queue becomes empty, the session is switched back to the FACH. As soon as the source receives the ACK it releases packets to the queue thus prompting a switch back to DCH. This redundant switch consumes $500ms$ and thus adds to the delay of the burst. However, if there are enough packets present in the queue such that the queue does not become empty prior to the arrival of the ACK, there is no switch back and forth. The number of packets which ensures that this back and forth switch does not take place is 5 and hence we see an optimal value at $T_h = 4$.

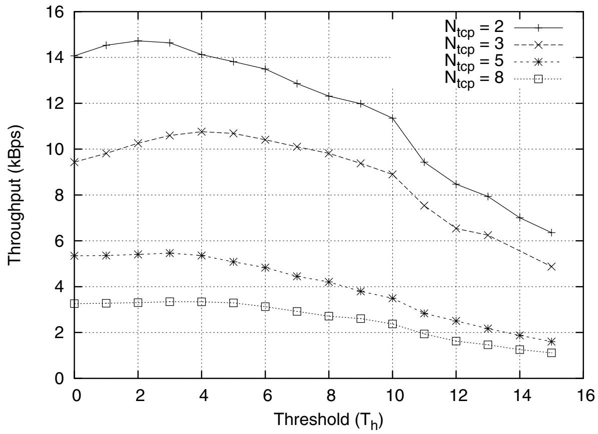

Fig. 4. Throughput vs. T_h. $N_{dch} = 1$, $D_w = 250ms$.

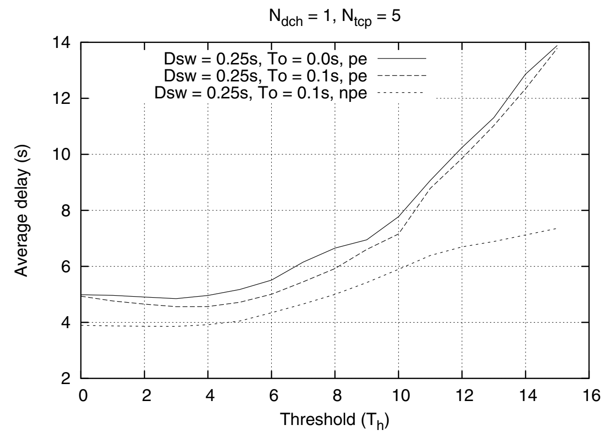

Fig. 5. Average Delay vs. T_h. $N_{dch} = 1$,$N_{tcp} = 5$, $D_w = 250ms$.

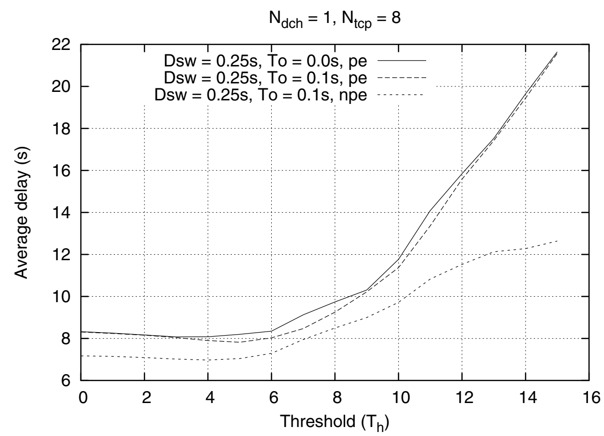

Fig. 6. Average Delay vs. T_h. $N_{dch} = 1$,$N_{tcp} = 8$, $D_w = 250ms$.

For a given value of T_h, the average burst delay increases as a function of N_{tcp}. As the number of TCP connections on the FACH increase, the available bit-rate for each connection reduces thereby increasing the time taken to increase the window size and hence to exceed the threshold.

Fig. 4 shows the throughput as a function of the T_h for different values of the number of TCP connections, N_{tcp}. The throughput first increases and after an optimal value of T_h it decreases. The improvement in throughput at the optimum threshold is more prominent compared to improvement of delay. For a given threshold, the throughput decreases with increasing number of TCP connections.

V. Modified threshold policy

In the previous section we observed that the use of the *original* threshold policy resulted in a switch from DCH to FACH even though the source had packets to send. This resulted in a poor delay performance for the policy. In this section we propose a *modified* threshold policy which aims to ameliorate this flaw of the *original* threshold policy. In the *modified* threshold policy, the switch from the FACH to DCH takes place according to the same criterion as the *original* threshold policy. However, when the queue length at SWI_i drops below T_l and the connection i is on DCH we start a timer for a duration T_{out}. If there are packet arrivals at the queue (SWI_i) during this period, we reset the timer to 0. We simulate two policies which decide on the switch back to FACH.

In the *pre-emptive scheme*, if during the timeout period there is another TCP connection which requires a switch to DCH, we initiate a switch back to FACH.

In the *non pre-emptive* scheme, we wait till the end of the timer before we initiate the switch back to FACH. Also, at the end of the timer if there are no other requests for the DCH, we start the timer for a duration of T_{out} thus allowing the connection to remain on DCH.

A timeout value of 0 results in the *original* threshold policy. The use of timeout periods is motivated by the fact that there is a delay before the ACKs reach the source. Hence, even though the queue at the base station is empty the source may have packets to send and may be waiting to receive the ACKs before it sends the data packets. The timer, to some extent, waits for the ACKs to reach the source and the source to release the packets.

VI. Simulation results II

Figs 5 and 6 show the average delay performance of the *pre-emptive (pe)* and *non pre-emptive (npe)* schemes of the modified threshold policy. We again note that the case of $T_o = 0$ corresponds to the *original* threshold policy. From the figures, we observe that the *modified* threshold policies also have a optimal value of the threshold as was observed in the case of the *original* threshold policy. As compared to the original policy, the modified policy with *pre-emptive* scheme performs marginally better. As the number of TCP connections increases, the probability of a request for a switch also increases. Hence, the *pre-emptive* scheme, in which a switch is initiated as soon as there is a request, performs very close to the original policy when $N_{tcp} = 8$. With $N_{tcp} = 5$, the duration of the idle period on the DCH (i.e., when the timer is on and no requests have been received) is longer and thus the connection is able to receive packets from the source and remain on the DCH and avoid a costly (in terms of time) switch back to FACH. The

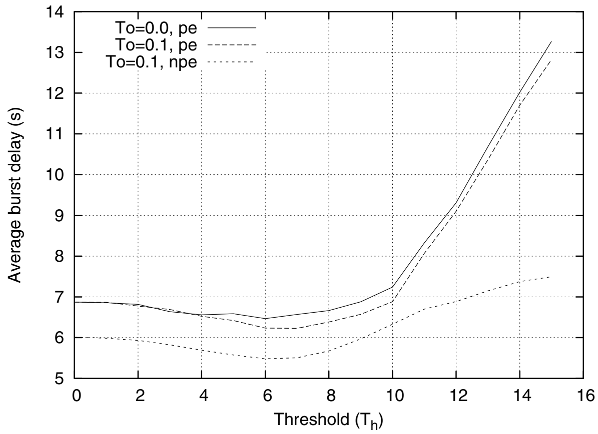

Fig. 7. Average Delay vs. T_h. $N_{dch} = 1$,$N_{tcp} = 5$, $D_w = 500ms$.

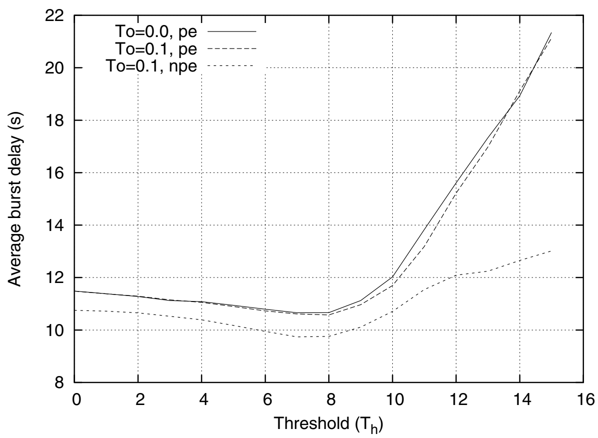

Fig. 8. Average Delay vs. T_h. $N_{dch} = 1$,$N_{tcp} = 8$, $D_w = 500ms$.

non pre-emptive scheme performs better than both the *original* policy and the *pre-emptive* scheme. The performance gains are close to 20% at $T_h = 0$ and $N_{tcp} = 5$. In the *non-preemptive* scheme, the base station waits till the end of the timer before it initiates a switch. The timeout duration ensures to some extent that the TCP session has ended and thus can be switched back to FACH. Hence, a TCP session which is once switched to DCH remains on the high speed DCH till the end of the session thereby reducing its average delay.

Figs. 7 and 8 show the performance of the policies when the switching delay, D_w is $500ms$. At a higher switching delay, the optimal threshold shifts to right. As the cost of shifting to DCH and back is high, it is preferable to remain on the FACH when the connections have fewer packets to send. However, we still note that the *non pre-emptive scheme* performs better than the other schemes. With a higher switching delay, the performance gain obtained by using the optimal threshold is higher and is observed to be around 10% compared to the no threshold case.

Figure 9 shows the average delay with $N_{dch} = 2$, $N_{tcp} = 8$, and $D_w = 250ms$. We observe a similar behavior as in the case of $N_{dch} = 1$.

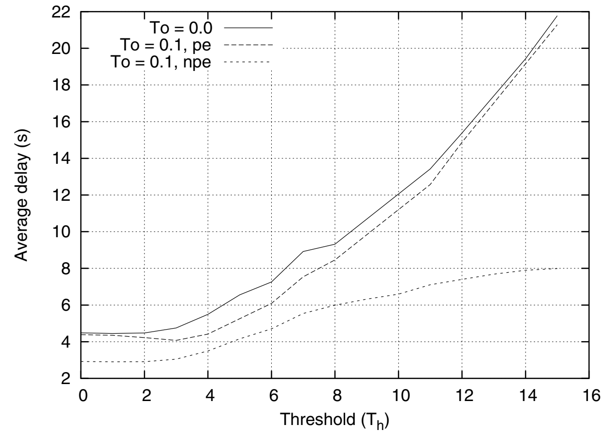

Fig. 9. Average Delay vs. T_h. $N_{dch} = 2$,$N_{tcp} = 8$, $D_w = 250ms$.

Also, in general, we observe that there is significant loss in delay performance at threshold values higher than the optimal value.

VII. CONCLUSIONS

In this paper, we evaluated, through simulations, the performance TCP data transfers on the UMTS downlink. We used a threshold policy to determine which TCP connections should use the high speed dedicated channels. The results indicated the presence of a optimal value of the threshold for minimum burst delay. However, the threshold policy resulted in redundant switches between the FACH and DCH, and therefore in increased average delay for the file transfers. We used a timer on the DCH to prevent these frequent switches. We observed that the use of a timer on the DCH resulted in improved delay performance for the bursts (files). The results indicate that for the simulated parameters the use of a threshold did not significantly improve the delay performance and suggest that use of a timer with a threshold can give improved performance.

REFERENCES

[1] H. Holma and A. Toskala, *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*, John Wiley & Sons, 2001.

[2] B. A. Mah, "An Empirical Model of HTTP Network Traffic", *Proc. IEEE INFOCOM'97*, April 1997.

[3] UMTS 30.03 version 3.2.0, Technical Report TR 101 112, ETSI, April 1998.

[4] C. M. Goldie and C. Kluppelberg, "Subexponential Distributions", in *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, ed. R. L. Adler, R. Feldman, M. S. Taqqu, pp. 435-459. Birkhuser, Boston, 1998.

[5] S. McCanne and S. Floyd. ns Network Simulator. http://www.isi.edu/nsnam/ns/

[6] M.E. Gomez and V. Santoja, "Analysis of Self-similarity in I/O Workload using Structural Modeling", 7^{th} International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, March 24-28, 1999, College Park, Maryland.

[7] D. P Heyman, T. V. Lakshman and A. L. Neidhardt, ".A New Method for Analysing Feedback-based Protocols with Applications to Engineering Web Traffic over the Internet", *Proc. ACM SIGMETRICS'97* Seattle, USA.

[8] R. Braden, "RFC 1122: Requirements for Internet Hosts – Communication Layers", October 1989.

[9] 3GPP TS 34.108, "Common Test Environment for User Equipment (UE) Conformance Testing", version 4.1.0, Release 1999.