International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue 1, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

# An Efficient Congestion Control Protocol for Wired/Wireless Networks

**Hanaa Torkey, Gamal ATTIYA, Ahmed Abdel Nabi**

***Abstract* – Recently, wide spectrum of heterogeneous wireless access networks integrate with high speed wired networks to deliver Internet services. End-to-end service delivery with satisfactory quality is challenging issue in such network architectures. Although the Internet transport control protocol (TCP) addresses such challenge, it has poor performance with high speed wired networks (i.e. high bandwidth-delay product). Moreover, it behaves badly with wireless access networks (i.e. misinterpretation of packet loss occurrence that could be interpreted in wrong way as network congestion). Thus, adapting TCP in terns of accurate interpretation of network status has drawn a significant attention. This paper addresses the problem of adapting congestion window size as an interpretation of frequent fluctuations of network load. The work in this paper proposes congestion window size (*cwnd*) adjustment in terms of instantaneous network load (i.e. bandwidth estimation). The aims are to improve End-to-End TCP throughput as well as maximum resource utilization. With different scenarios, NS-2 simulator has been adopted to evaluate proposed TCP protocol. Notable improvement has been demonstrated by comparing proposed protocol to TCP NewReno benchmark and TCP Vegas.**

***Keywords* – Transmission Control Protocol (TCP), Network Protocol, Internet, Congestion Collapse, Congestion Control.**

## I. INTRODUCTION

Transmission Control Protocol (TCP) is the dominant transport protocol in the Internet. It supports most of the popular Internet applications, such as file transfer and e-mail. Recently, Internet applications have evolved from standard document-retrieval functionality to advanced multimedia services. The rapid growth of the Internet and the increasing of the traffic demand led to a serious problem called congestion collapse [1]. This problem causes unacceptable long response time particularly for real-time applications such as audio and video streaming. Congestion Control is thus proposed as a remedy to congestion collapse and ensures the Internet stability in terms of throughput performance improvement.

Over years, continuous efforts have been done to avoid congestion problem. In 1988, a congestion control mechanism, called TCP Tahoe, was introduced [2]. It composed of three phases; slow start, congestion avoidance, and fast retransmit. Two years later, a new TCP version, called TCP Reno, is developed by incorporating a fast recovery algorithm to Tahoe [3]. Several implementation versions are developed and refined by researchers aiming to overcome the performance degradation problem of Reno. Some researchers attempted to refine the fast retransmit and fast recovery algorithms while other researchers attempted to refine the slow start and congestion avoidance algorithms [4]. In [5], a congestion control mechanism, called TCP NewReno, is developed by refining the fast recovery algorithm of TCP Reno to recover multiple packet losses from a window of data. In [6], an extension of Reno, called TCP SACK (TCP with Selective Acknowledgement), is developed by modifying the fast recovery algorithm of Reno keeping the other algorithms unchanged. Similar to NewReno, TCP SACK handles multiple packet losses from the same window but it has a better estimation capability for the number of outstanding packets. Additional mechanisms include Forward Acknowledgment (FACK) [7], Selective Acknowledgment (SACK) [6], dynamic recovery [8], an Extension to the SACK Option (D–SACK) [9], TCP with Faster Recovery (FR–TCP) [10], Reordering–Robust TCP (RR–TCP) [11], Duplicate Acknowledgment Counting (DAC) [12], TCP SACK+ [13], TCP Vegas [14,15]. Other proposals, such as TCP Westwood [**16**] and TCP WestwoodNew [**17**] are developed to overcome congestion in wireless networks. A survey of TCP Reno, NewReno and SACK over mobile ad-hoc network is investigated in [**18**]. Comparative studies between the different proposals for TCP congestion control are presented in [19-21].

Although several variants were developed and refined, most of them employ Additive Increase Multiplicative Decrease (AIMD) strategy for adjusting the congestion window size (cwnd). In other words, they use the AIMD to change the rate at which packets injected into the network. This strategy is inefficient in terms of utilization of link capacity and unfair in throughput. This is because the AIMD strategy blindly updates the congestion window statically by a fixed value regardless the network status. That is, the mechanism increases the congestion window by one segment during the congestion avoidance phase as long as no congestion is detected, and decreases the congestion window to half of its value, in entering the fast recovery, as long as congestion is detected by 3 DACK.

In this paper, an adaptive congestion control mechanism is developed to avoid congestion problem in wired and mixed wired/wireless networks. The proposed mechanism does not use the AIMD strategy in updating the ***cwnd*** like other variants but it dynamically estimates the available network bandwidth using the received acknowledgments and hence adjusts the congestion window (cwnd) of the TCP sender accordingly based on the available capacity of the network. The purpose is to maximize the bandwidth utilization and the throughput of the TCP connection.

The performance of the proposed mechanism is evaluated by simulation studies, under different network scenarios, using the well known Network Simulator NS-2. The simulation results show that, the proposed mechanism achieves better throughput, efficient utilization of the network resources, fair bandwidth sharing, and lower packet losses and delay time.

The rest of this paper is organized as follows; Section 2 presents an overview of the most widespread congestion control protocol; TCP-NewReno. Section 3 discusses the drawbacks of the TCP-NewReno while the proposed mechanism is described in Section 4. Section 5 presents the simulation results. Finally, Section 6 presents the paper conclusions.

## II. TCP NewReno

The TCP NewReno is the enhanced version of the TCP Reno to combat multiple packets dropped from a single window without entering into fast recovery multiple times [20]. It composed of 4 algorithms: Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery. The first two algorithms are used by a TCP sender to control the amount of data injected into the network, the third algorithm is used to resend what appears to be lost without the need for Retransmission Time Outs (RTO) while the fourth algorithm allows the sender to work in congestion avoidance instead of returning to slow start as a congestion is detected by 3 DACk to improve the network throughput.

### *A. Slow Start*

In Slow Start, the congestion window (cwnd) is first initialized by one segment [15]. Then, the sender transmits one segment and waits the acknowledgement from the receiver. When the acknowledgement is returned, the congestion window is incremented by one, and two segments are sent. For each acknowledgement returned from the receiver, the congestion window is increased by one segment, implying doubling of cwnd per Round Trip Time RTT. This behavior continues until the cwnd reaches the slow start threshold (ssthresh). At this point, the NewReno enters into Congestion Avoidance phase to slow the increasing rate of the cwnd and hence the sending rates of packets.

### *B. Congestion Avoidance*

During congestion avoidance, the congestion window increases linearly by one segment per round trip time (RTT). This linear increase continues until a packet loss is detected by duplicate acknowledgement or retransmission timer expiration. If the congestion was indicated by a timeout, the ssthresh is set to one half of the current window and the congestion window (cwnd) is reset to one segment, and the sender automatically puts into slow start mode. But, if the congestion was indicated by duplicate acknowledgements, fast retransmit is invoked.

### *C. Fast Retransmit*

During fast retransmit, the TCP sender retransmits (what appears to be missing segment) the lost packet and enters the fast recovery algorithm after setting the values of the cwnd and the ssthresh. The purpose of fast retransmit is to speed up the retransmission of lost packet instead of waiting the expiration of timeout.

### *D. Fast Recovery*

In entering fast recovery, the ssthresh is set to one-half the current congestion window (cwnd) but no less than two segments and the cwnd is set to the ssthresh plus three. TCP NewReno continues in fast recovery until all the packets which were outstanding during the start of the fast recovery have been acknowledged. This strategy helps to combat multiple losses without entering into fast recovery multiple times as TCP Reno or causing timeout. During fast recovery, the TCP NewReno distinguishes between a partial Ack and a full Ack. A partial acknowledgement is considered as an indication that the packet following the acknowledged one in the sequence space has been lost and should be retransmitted. Therefore, TCP NewReno immediately retransmits the other lost packet indicated by the partial acknowledgement and remains in fast recovery. On the other hand, a full acknowledgement acknowledges some but not all of the outstanding data. TCP NewReno exits fast recovery when all data in the window is acknowledged. When the sender receives new acknowledgement, it will exit that phase putting its cwnd to ssthresh and start congestion avoidance algorithm. Note that, whenever a timeout occur, the ssthresh is set to one half of the current congestion window and the congestion window is set to one and the sender enters into the slow-start phase.

## III. Drawbacks of TCP NewReno

The comparative studies in [21] show that the TCP NewReno provides better performance than previous TCP variants. However, it has been found that the TCP NewReno is inefficient in terms of utilization of link capacity and unfair in its throughput performance. This is because; in NewReno, congestion control is based on the packet loss detection and the NewReno updates its transmission rate (congestion window size) using blind mechanism, regardless to the current load at the network.
Briefly, TCP NewReno suffers from two main drawbacks in the behavior of both the congestion avoidance and the fast recover phases.

- During the congestion avoidance phase, the sender updates the congestion window (cwnd) blindly by constant value, where the sender increases the congestion window size linearly by one segment with each ACK, as long as no losses are detected. This strategy makes TCP NewReno inefficient in term of network utilization. This is because, increasing the congestion window by one segment makes the sending rate at the sender to be lower than the available capacity of the TCP connection especially when the network is lightly loaded.
- During the fast recovery phase, TCP sender halves its congestion window irrespective to the state of the network, as long as a packet loss is detected by 3 DACK. This strategy degrades the network throughput. In entering Fast Recovery, the process of setting the cwnd to the half of its value as congestion is detected by 3 DACK, degrades the network throughput. This strategy makes TCP NewReno inefficient in term of network utilization.

Another problem arises with NewReno is that, when there are no packet losses but instead packets are reordered by more than three duplicate acknowledgments, TCP sender mistakenly enters the fast recovery and half its congestion window [22, 23].

International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue 1, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

## IV. Proposed Mechanism

The proposed mechanism is developed to overcome the main drawbacks that arise in both the congestion avoidance and the fast recovery algorithms of the TCP NewReno.

### *A. Modified Congestion Avoidance*

The key idea of modification in congestion avoidance algorithm is to adjust the *congestion window* at the TCP sender dynamically according to the available connection capacity at any time. In other words, the strategy is to adjust the source's sending rate (congestion window) in an attempt to dynamically adapt the congestion window based on the network load, and keeping number of packets buffered in the routers along the transmission path.

Briefly, as soon as the congestion window crosses the slow start threshold (ssthresh), TCP goes into congestion avoidance phase. In this phase, (i) the sender monitors the acknowledgment stream that it receives from the destination (the source performs an end-to-end estimate of the transfer rate achieved along a TCP connection), (ii) the sender estimates the data transfer rate currently available by the TCP connection, and (iii) the sender adjusts the cwnd according to the currently available capacity/rate of the TCP connection.

By applying these modifications, the greater the bandwidth of a given path, the higher the data transfer rate of the sender.

### *B. Modified Fast Recovery*

The crux of idea of modification is to decrease the congestion window (cwnd) and hence the ssthresh, as network congestion is detected, based on the state of the network instead of decreasing cwnd to half of its value. In other words, the sender determines changing degree in the network traffic load using the change in the Round Trip Time (RTT) and changes the congestion window size accordingly. Briefly, since all the packets make a round trip from the sender to the receiver and back to the sender, hence, the round trip time changes during the TCP connection as network traffic load changes. Where, the value of Round Trip Time (RTT) increases with the increasing of the network load. So it could be used to reflect the network status.

In entering the Fast Recovery algorithm, the sender can detect the change in the RTT and decrease the congestion window by a value related to the increases in the RTT. The process is that, the sender continuously monitors the RTT from the receiving acknowledgments and keeps up with an array of the last few numbers of RTTs values. In entering fast recovery, the sender uses the queued values of the RTTs to compute the average value. Then, it calculates the deference between the last RTT value, just right before detecting congestion, and the average RTT. The sender then uses the change in the RTT to calculate the increasing factor according to the change in RTT and finally decreases the congestion window (cwnd) by a value proportional to the increasing factor. Finally, the ssthresh is updated as the maximum of two segments and the difference between the current congestion window (cwnd) and the increasing factor.

## V. Performance Evaluation

In this section, the performance of TCP EnewReno will be investigated over asymmetric networks, mixed Wired/Wireless Network Topology. To demonstrate the effectiveness of the proposed mechanism (EnewReno), the proposed mechanism as well as the existing mechanisms are implemented and included within the well known network simulator NS-2 [24, 25]. The proposed mechanism (EnewReno) and the existing mechanisms (NewReno and Vegas) are evaluated considering throughput as the evaluation criteria.

### *Mixed Wired/Wireless Network Topology:*

Figure 1 shows a mixed network topology including wired and wireless portions. The wired portion links source nodes and a base station with 10MB bandwidth. The propagation time over the wired link is initially assumed to be 5ms. The wireless portion of the network is assumed to connect the base station to a destination mobile terminal. The wireless link is a very short 5MB wireless link with a propagation time of 2ms. Errors are assumed to be burst, and to occur in both directions of the wireless link. The sources are at the wired and wireless nodes, while the destinations are chosen in one wireless node. The network topology is described in Fig. 1. In the topology, there are two TCP connections, one connection wired and wireless node, while the other between two wireless nodes.

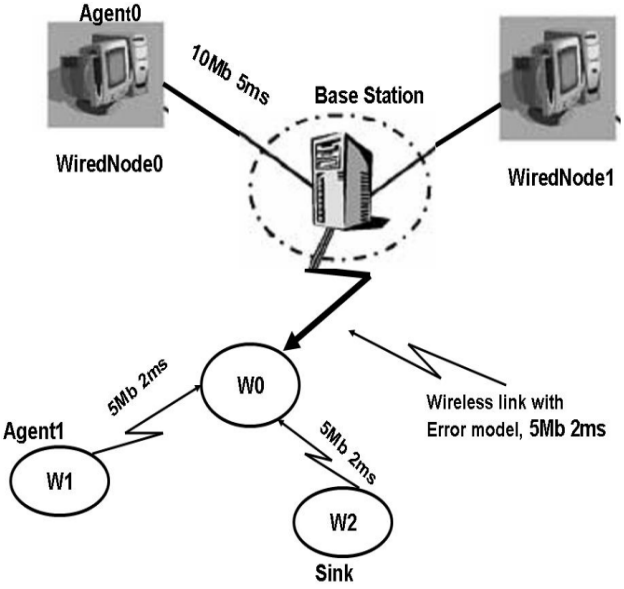


Fig.1. Mixed Wired/Wireless network model

### *Wireless link characteristics:*

When a particular link is used for IP transport, the three most important characteristics of the link are the capacity, the delay, and the loss probability. For wired links, these are constant, and the loss probability is usually very small. For wireless links, all three are time varying. In recent times there has been a great proliferation of wireless devices, and growing popularity of 802.11 based wireless networks. These networks come in the form of enterprise networks, wireless hotspots, and more recently wireless mesh networks. The common underlying requirement of

each of these networks is to provide good performance either improved throughput or satisfying QOS constraints, like acceptable throughput under high losses probability, low delay.

However the nature of the wireless medium, and 802.11 MAC (Media Access Control) protocol make it much harder to provide these guarantees when compared to a wired network. A wireless link experiences diversity, i.e. highly varying loss rate and bandwidth. Its statistics depends on the tradeoffs made in the link layer design. E.g., the introduction of link layer retransmissions reduces the loss probability, and pays the price of larger delay and larger delay variations, while adaptation of the parameters for modulation and forward error correction can trade capacity for a reduced loss probability.

Communication end-nodes may have some direct knowledge of the link they are directly attached to, but they primarily observe the characteristics of the entire network path they are using. With no cross traffic, the observed capacity would be the capacity of the smallest link on the path, but with cross traffic, the observed capacity depends not only on the links on the path, but also on the cross traffic, which in turn depends on other parts of the network topology. The observed delay is the sum of the delay of each traversed link, and the queuing delay. The observed loss rate is the combination of the loss rate at each link (links can usually be considered independent).

So, the protocols developed for wired networks should be carefully examined to be account for the characteristics of the wireless channel. This is important to define their usability over wireless networks.

## VI. Simulation Result

Figure 2 shows the throughput of EnewReno, Vegas, and NewReno with the time. As shown from the figure, TCP Vegas suffer poor performance over the wireless network. That is due to the sending rates for data packets during the congestion avoidance algorithm of Vegas. It is important to emphasize that the aggressive natural of NewReno make its performance better than TCP Vegas. But with the high probability of loses, NewReno with each losses detection cut its congestion window to half. However, the sending rate of TCP should not be unnecessarily aggressive. Otherwise, retransmission may become excessive and will hurt the overall throughput. On the other hand, TCP EnewReno get higher throughput than NewReno and Vegas. As shown in the figure, EnewReno starts with throughput less than NewReno in the first 10 seconds of simulation. Since, NewReno continuously increases its cwnd unlike EnewReno which eliminate that unnecessarily aggressive sending rate. With the time, EnewReno achieve higher throughput due to the enhanced fast recovery algorithm, while EnewReno decreasing its cwnd as a function of the network status.

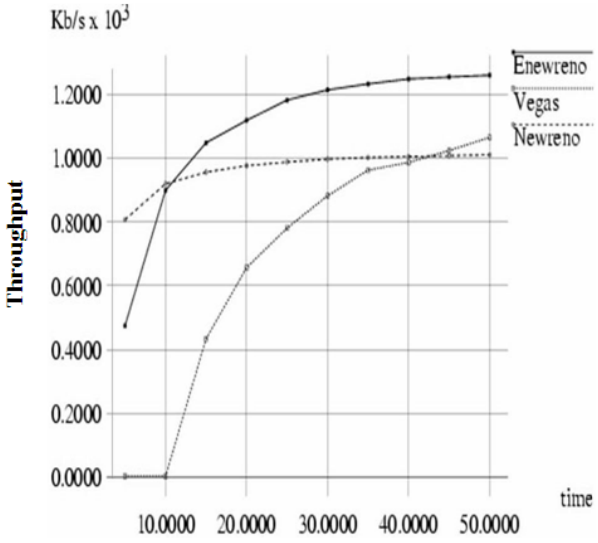


Fig.2. throughput with time

The natural of wireless infrastructure characteristic increasing data corruption probability, the TCP cannot distinguish packet losses caused by congestion from those caused by corruption. These matrices have been analyzed to show the effect of none zero packet loss rates on the average throughput that can be achieved. Figure 3 shows the change of throughput with the packet loss rate. As shown from the figure, packet losses rate have undesirable effect on the throughput, as it could be noted that the throughout for all variant decreased with increasing the packet losses rate. But, EnewReno get better throughout than the others.

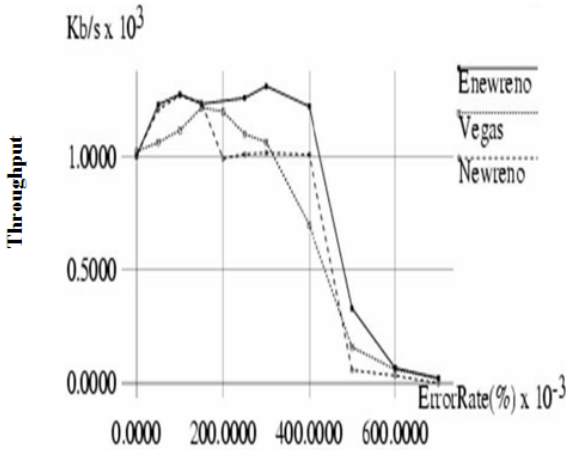


Fig.3. The change of throughput VS. Packet loss rate

## VII. Conclusions

In this paper, an efficient congestion control mechanism is developed. The algorithm dynamically estimates the available network bandwidth at any time and adapts the sending rate at the TCP sender according to the changes in traffic load so as to avoid network collapse. The proposed mechanism efficiently utilizes available capacity of the network as it adapts the sending rate dynamically. The results indicate that the proposed mechanism is more effective than NewReno and Vegas in wired and mixed wired/wireless networks.

## REFERENCES

[1] J. Postel, “Transmission Control Protocol,” IETF RFC 793, September 1981.
[2] V. Jacobson, “Congestion Avoidance and Control,” ACM SIGCOMM Computer Communication Review, Vol. 18, No. 4, pp. 314-329, August 1988.
[3] V. Jacobson, "Berkeley TCP Evolution from 4.3-Tahoe to 4.3 Reno," Proceedings of the 18th Internet Engineering Task Force, University of British Columbia, Vancouver, BC, Aug. 1990.
[4] Cheng-Yuan Ho, Yaw-Chung Chen, Yi-Cheng Chan, Cheng-Yun Ho, "Fast retransmit and fast recovery schemes of transport protocols: A survey and taxonomy," Computer Networks, Vol. 52, pp.1308–1327, 2008.
[5] J. Hoe, “Start-up Dynamics of TCP’s Congestion Control and Avoidance Schemes,” Master Theses, Massachusetts Institute of Technology, 1995.
[6] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, “TCP Selective Acknowledgment Options,” RFC 2018, Internet Engineering Task Force, October 1996.
[7] M. Mathis, J. Mahdavi, "Forward acknowledgement: refining TCP congestion control," Proceedings of ACM SIGCOMM, pp. 181-191, 1996.
[8] H. Wang, C.L. Williamson, “A new scheme for TCP congestion control: smooth-start and dynamic recovery,” Proceedings of IEEE MASCOTS, pp. 69–76, 1998.
[9] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky, “An extension to the selective acknowledgement (SACK) option for TCP,” IETF RFC 2883, July 2000.
[10] C. Casetti, M. Geria, S.S. Lee, S. Mascolo, M. Sanadidi, “TCP with faster recovery,” Proceedings of IEEE MILCOM, vol. 1, pp. 320–324, 2000.
[11] M. Zhang, B. Karp, S. Floyd, L.L. Peterson, “RR–TCP: a reordering–robust TCP with DSACK,” Proceedings of IEEE ICNP, pp. 95–106, 2003.
[12] B. Kim, J. Lee, “Retransmission loss recovery by duplicate acknowledgment counting,” IEEE Communications Letters, vol. 8, No. 1, pp. 69–71, 2004.
[13] B. Kim, D. Kim, J. Lee, “Lost retransmission detection for TCP SACK,” IEEE Communications Letters, vol. 8, No. 9, pp. 600–602, 2004.
[14] L. S. Brakmo, S. W. O’Malley and L. L. Peterson, “TCP Vegas: New Techniques for Congestion Detection and Avoidance,” Proceedings of ACM SIGCOMM, London, August 31-September 2, pp. 24–35, 1994.
[15] L. Brakmo and L. Peterson, “TCP Vegas: End-to-End Congestion Avoidance on Global Internet,” IEEE Journal on Selected Areas in Communications, Vol. 13, No. 8, pp. 1465-1480, 1995.
[16] George Xylomenos and George C. Polyzos, "TCP Performance Issues over Wireless Links," IEEE Communications Magazine, pp. 52-58, April 2001.
[17] Shimaa Hagag and Ayman El-Sayed, "Enhanced TCP Westwood Congestion Avoidance Mechanism (TCP WestwoodNew)", International Journal of Computer Applications, Volume 45–No.5, 0975 – 8887, May 2012.
[18] Shugong Xu and Tarek Saadawi, "Performance evaluation of TCP Algorithm in Muli-hop wireless Packet Networks," Wireless Communications and Mobile Computing, Vol. 2, pp. 85-100, 2002.
[19] K. Fall and S. Floyd, “Simulation Based Comparisons of Tahoe, Reno and SACK TCP,” ACM SIGCOMM Computer Communication Review, Vol. 26, No. 3, July 1996.
[20] J. Mo, R.J. La, V. Anantharam, J. Walrand, "Analysis and comparison of TCP Reno and Vegas," Proceedings of IEEE INFOCOM, 1999, pp. 1556–1563.
[21] Hanaa A. Torkey, Gamal M. Attiya and I. Z. Morsi, "Performance Evaluation of End-to-End Congestion Control Protocols", Minufiya Journal of Electronic Engineering Research (MJEER), Vol. 18, No. 2, pp. 99-118, July 2008.
[22] D. Roman, K. Yevgeni, and H. Jarmo, ”TCP NewReno Throughput in the Presence of Correlated Losses: The Slow-but-Steady Variant”, IEEE International Conference on Computer Communications INFOCOM, pp. 1- 6, April 2006.
[23] M. Niels, B. Chadi, A. Konstantin, and A. Eitan, ”Inter-protocol fairness between TCP NewReno and TCP Westwood+”, 3rd EuroNGI Conference on Next Generation Internet Networks, Vol.1, pp. 21-23, May 2007.
[24] K. Fall, and K. Varadhan, “The ns Manual (formerly ns Notes and Documentation)”, UC Berkeley, LBL, USC/ISI, and Xerox PARC, December 2006.
[25] GT-ITM “Georgia Tech Internetwork Topology”, http://www.cc.gatech.edu/project/gtitm.

## AUTHOR’S PROFILE

### Hanaa Torkey
is a M.Sc, Dept. of Computer Science and Engineering, faculty of Electronic Engineering, Menoufia University, Egypt. Ph.D. Candidate.

### Gamal ATTIYA
is a lecturer, Dept. of Computer Science and Engineering, faculty of Electronic Engineering, Menoufia University, Egypt.
Email: gamal.attiya@yahoo.com

### Ahmed M. ABDEL NABI
is an Associated Professor at City for Scientific Research and Technological Applications, Informatics Institute, Head of Network and Distributed Systems Department, Alexandria Egypt.
Email: iplanetfit@yahoo.com