Improving the Performance of Interactive TCP Applications using End-point Based and Network Level Techniques

Varun G Menon
Assistant Professor
Department of Computer Science and Engineering
M.E.T'S School of Engineering,Thrissur,India
varungmenon46@gmail.com

***Abstract*---Recent measurement based studies reveal that most of the Internet connections are short in terms of the amount of traffic they carry, while a small fraction of the connections are carrying a large portion of the traffic.. Most of these short flows are from interactive applications like telnet, gaming that use TCP protocol for connection establishment and data transfer. These short TCP flows suffer from severe response-time performance degradations when multiplexed with long-lived flows during times of network congestion. The reasons for this problem is that, in the absence of large number of packets the short flows are unable to get a detailed knowledge about the level of underlying network congestion and even a single packet loss forces long retransmission timeouts. Also as the numbers of packets are less they are not able to develop large congestion windows and thus unable to jumpstart the next data burst. Due to this, clients of interactive applications suffer from increased response time for data packets sent and they try to upgrade their short flows to long flows by sending dummy packets into the network even when they do not have any data to send. This behavior can lead to severe congestion in the network and causes harm to statistical multiplexing in the Internet. This paper aims at providing easy to implement techniques that can be used by the clients of interactive applications to get much better performance without causing any serious congestion in the network.**

.

***Keywords*---Interactive application, Latency, Performance Degradation, Retransmission Timeout, TCP.**

I. INTRODUCTION

Most of the flows in today's Internet are short in terms of the amount of traffic they carry. Majority of the applications in the Internet uses short data flows and sends packets in short burst. They often use TCP protocol for connection establishment and data transfer. It is well known that short TCP flows may experience significant performance degradations when they are multiplexed with long-lived TCP flows in the Internet. The root of the problem is the lack of knowledge about the level of the underlying network congestion. In absence of the large number of packets characteristic for long-lived flows, even a single packet loss can force a short-lived TCP flow to experience long retransmission timeouts [1], which in turn significantly increase a client's perceived response time. Another reason for this problem is that the TCP relies on its own packet samples to estimate an appropriate retransmission timeout (RTO) value. For the first control packets (SYN, SYN-ACK), and the first data packet, since no sampling data is available, TCP has to use a conservatively estimated initial timeout value as RTO. Losing these packets can have a disastrous effect on short connection performance due to the large timeout period. It is due to these reasons that we experiences large and variable delays when using interactive Internet applications, such as Telnet and the Web. As the numbers of short flows in the Internet are very high, it is very important to reduce this delay.

The growing importance of these and similar Internet applications also encourage us to improve their performance. Providing preferential treatment to these short flows is one way to reduce this delay experience by the short TCP flows. Various queuing mechanisms can also be tried out to reduce this delay. Another way of improving the performance of short flows is to keep them unaware of network congestion. Classifying the flows in the Internet and informing only the short flows about congestion would help short flows to avoid unnecessary delays. Most of these methods have a number of limitations and most of them cannot be deployed in Internet due to its complexity and various other problems.

Due to all these factors clients of interactive applications always tries to upgrade their short flows to long flows simply by sending dummy packets into the network even when they do not have any data to send. This technique is easy to implement and is not detectable by any monitoring authorities. The major problem with this technique is that, when followed by a large number of users, it can lead to severe congestion and can cause serious harm to statistical multiplexing in the Internet. This paper provides two easy to implement approaches with different variations that can be used to get better performance than the previous technique without causing any serious congestion in the network. Different variations of these techniques are also proposed in this paper which can provide much better performance during different network conditions. Through simulation using the ns-2 tool we show that the two new approaches have a better performance than all the previous approaches. The two approaches can be implemented at the application level as well as the network level. Using Response time, average end to end delay and Packet Delivery Ratio (PDR) as the performance parameters we show that the proposed methods have better performance than all the previous methods. Also the simulations results shows that the new approaches are capable of de-motivating the clients of interactive applications from upgrading their short flows to long flows.

This paper is organized as follows: Section II describes the previous methods used to reduce the latency experienced by the short TCP flows. The drawbacks of these methods are also explained in this section. Section III consists of the problem origins and its implications. Section IV, consists of a comparison of response time of short and long TCP flows. In Section V, we explain the proposed approaches, evaluate the performance of various methods and a comparison is made. Finally, in Section VI we conclude.

II. RELATED WORK

In this section we describe the various methods that were used to improve the performance of short TCP flows**.** Guo and Matta [3] give a method for preferential treatment of short TCP flows in the network using the RIO-PS queuing mechanism. This method requires large changes to the existing network infrastructure. The mechanism proposed appears to address the problem of short flows, but not the interactive flows because its packet/flow classification technique is based on packet counting, rather than the sending rate of the flow. It would classify a long-lived interactive connection, even if sending packets at a very low-rate, into the category of long flows, once the packet counter exceeds the threshold value. Noureddine and Tobagi [5] propose application and TCP-level marking to give strict priority to interactive applications in the network. In addition to requiring per-user traffic policing at the network edge, the authors assume a widespread network support for multipriority services in the Internet.

Le, Aikat and Jeffay [4] propose an AQM scheme which gives a strict priority to short flows, and applying congestion only to long flows. The major advantage of this scheme is that this does not require the support of the end-points; it distinguishes short flows from long flows by tracking the number of packets that have recently been seen from each flow at the router. The major problem with this approach is that in addition to provoking potential security and stability side effects, the proposed scheme requires to be implemented in the network core; unfortunately, no strong incentives for such a deployment exist. Similarly, it has been shown that using the Explicit Congestion Notification (ECN) for marking, instead of dropping, could significantly improve the performance of short flows [6]. Unfortunately, ECN is not widely deployed in today's Internet due to a number of possible limitations and restrictions.

Mellia, Meo and Casseti [7] gave a segmentation algorithm, TCP Smart Framing (TCP-SF) to reduce TCP's delay. The basic idea of the algorithm was to segment data into smaller segments when TCP's congestion window is less than four maximum segment sizes (MSS). This helps TCP to recover from packet losses during the slow start period. In essence, this is an attempt to ensure that there are always enough packets in the network to trigger the fast retransmit. As the congestion window grows, the algorithm increases the segment size. This technique has been shown to perform well in reducing timeout events during the slow-start phase when the end point has new data to send in response to an acknowledgement. But it is seen that TCP-SF even degrades the response times for interactive applications. The key reason is that the probability of getting one of the multiple small segments lost in the network is even higher than when entire data is packed into a single packet. Savage, Cardwell and Anderson [9] propose the Informed Congestion Control mechanism that aims at improving the performance of data transfers over the Internet by collecting information from many co-located hosts. Although this method has a number of advantages the major problem is that the connections using this approach would converge quickly compared to the other flows.

Despite the fact that all of the above endpoint approaches enable protocol support for improving the performance of short or interactive flows, the key problem remains: the application-level data starvation can prevent clients from experiencing any benefits from the above designs. In particular, the burst periods of interactive flows are typically small enough to fit into a single packet. As a result, an increased congestion

window, a more aggressive TCP, or a history-based approach cannot help. If a packet gets lost in the network, the sender must rely on the mechanism before retransmitting the packet back into the network, thus experiencing significant performance degradations.

III. PROBLEM ORIGINS AND IMPLICATIONS

A. Problem Origins

TCP has two mechanisms for packet loss detection: Fast Retransmit and timeout (RTO). TCP interprets receipt of three duplicate ACKs as an indication of a packet loss. It retransmits the lost packet immediately upon the receipt of the third duplicate ACK. This mechanism is called Fast Retransmit; it detects a packet loss and reacts to it on the order of a flow's RTT. Another mechanism to detect a packet loss is the timeout mechanism. TCP sender starts a retransmission timer when it sends a packet. In case the acknowledgement for the packet is not received and the timer expires, it resends the packet. Comparing both the approaches the Fast Retransmit approach is much faster than the RTO approach.

The major reason for the performance degradation of short TCP flows is that, as the number of packets are very less in the flow they are not able to get a detailed knowledge about the level of congestion in the network. Also as the number of packets are very less they are not able to develop large congestion windows to quick start the next data burst. Indeed, given that such flows only have a few packets to send, in case a packet gets lost in the network, they have no other option but to wait for the RTO to expire. In other words, they are unable to resend the packet immediately after one RTT, because the three duplicate ACKs may never return and as also due to the fact that the values of RTO are much higher degrades the response time of the short TCP flows.

A. Implications

All these reasons encourage the users of interactive applications to upgrade their short flows to long flows. The logic behind the misbehavior is very simple; the users send dummy packets into the network when they do not have any data to send. That is they fill up the entire interval between two data packets with dummy packets so that the flow is not terminated. Whenever data packets are available, its send with strict priority and when not available, dummy packets are send

Incentives for clients to apply this approach are many. First, by adopting fully backlogged approach, clients avoid losing memory in moments of data starvation. It helps them to develop larger congestion windows that can help "jumpstart" an actual data burst arriving from the application. Second, "dummy" packets following data packets may significantly increase the probability that a potential packet loss will be detected via the triple-duplicate ACK mechanism rather than the RTO mechanism. Finally, clients can freely apply this approach, without any fear of "getting caught." This is because both network- and endpoint-based schemes designed to check for TCP-fairness compliance would detect no violations.

The major problem with this approach is that as the number of short flows in the Internet is so high and if every user starts using this approach, Internet will get totally congested. Moreover the statistical multiplexing principle of the Internet will be affected greatly. Another major issue is that implementing this approach is very easy for the user. Client-side only implementations, both at the TCP and the application levels are straight forward. Such designs could improve the times required to "push" packets to servers, a feature of particular interest to online gaming players. Moreover servers can be configured to send dummy packets automatically.

IV. COMPARISON OF PERFORMANCE OF SHORT AND LONG TCP FLOWS

Here we compare the performance of short TCP flows with the long TCP flow in a network using a shared link. The comparison is required to prove that the short flows suffer from severe performance degradation when multiplexed with long-lived flows over the Internet. The key performance metric used for comparison is the response time, defined as the time that elapses between sending a data packet into the network and receiving a corresponding acknowledgement. We use the ns-2 simulator for generating the results for comparison. Initially we create a topology with a shared link and generate both short and long flows over the shared link. The bandwidth of the shared link is adjusted and an error probability is defined for that link. We then measure the response time for packets of both the flows from the trace file using a java script file. We obtain different values of response time for both the flows using different probability of loss values. Fig. 1 shows the comparison of response time for both the flows as a function of probability of loss. The graph clearly shows that the response time for short TCP flows is much greater compared to the long lived flows over a shared link.

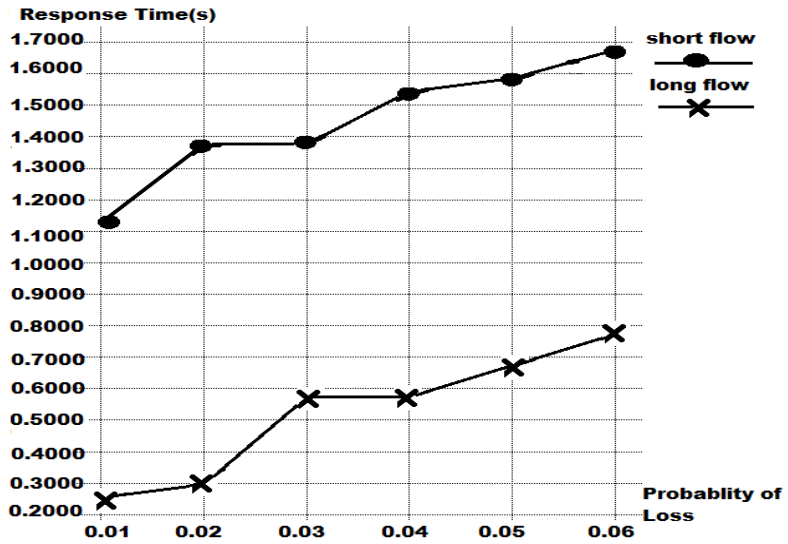

Fig. 1 Simulation: Response Time as a function of loss rate

V. PROPOSED APPROACHES AND PERFORMANCE EVALUATION

A. Three dummy packet approach

The main purpose of this method is to improve the performance of short TCP flows by increasing the probability that a packet loss is detected via the fast retransmit mechanism. Considering the fact that the Fast Retransmit approach is much faster than the retransmission timeout, this approach helps in improving the performance of interactive applications that uses short flows. In this approach we append application data with three light weight dummy packets. Three light weight dummy packets are send immediately after each data packet. Indeed, RFC 3390 [12] enables setting TCP's initial congestion window size to four packets when TCP starts a new connection or restarts a connection after a long idle period. The main idea behind this approach is that the three dummy packets will increase the probability of triggering three duplicate acknowledgements from the receiver to the sender in case the data packet is lost. Another major advantage of this approach is that this technique does not cause any serious congestion in the Internet.

Users of interactive applications can easily use this technique at the application layer itself. The users can configure their application to send three dummy packets after every data packet. As the interval between data packets is large for interactive applications, this approach does not cause much increase in load in the network. Also this approach does not require any kernel level TCP change. Using this approach, users can get better performance for their interactive applications compared to the fully backlogged approach.

B. Duplicate Copy Approach

In this approach we send more than one copy of the data packet to the destination. The key idea behind this approach is to increase the probability that at least one of the copies of the packet will make it to the receiver. Even if the original data packet is lost, there is a high probability that one of its copies reaches the destination and the transfer gets completed. This can be done by modifying the TCP sender to send more than one copy. Number of copies to be send can be decided based on the level of congestion in the network. Based on the current network status the TCP sender can be modified to send more than one copy of the data packet fro the interactive applications. Another feature of this approach is that, if all packets are lost, TCP undergoes retransmission timeout and cuts down the congestion window to one. Hence, in the following retransmission rounds, it retransmits the packet only once. This helps to maintain the stability of the network and prevents it from overloading.

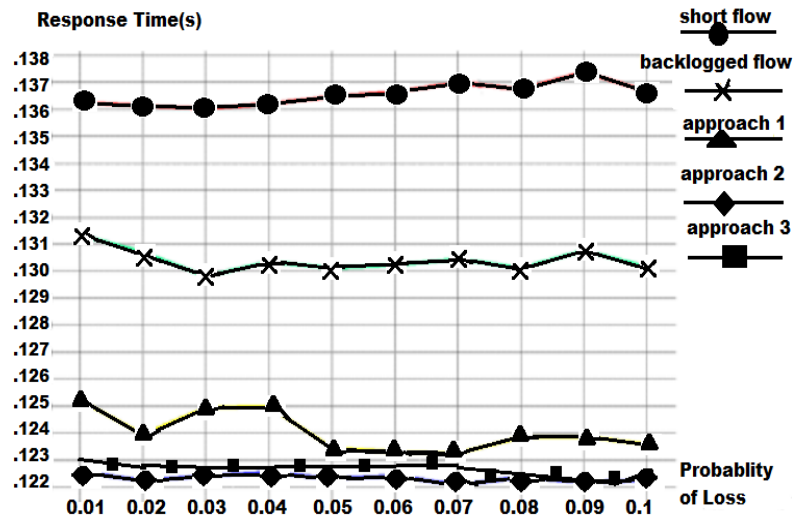

Fig. 2 Simulation: Response time as a function of loss rate

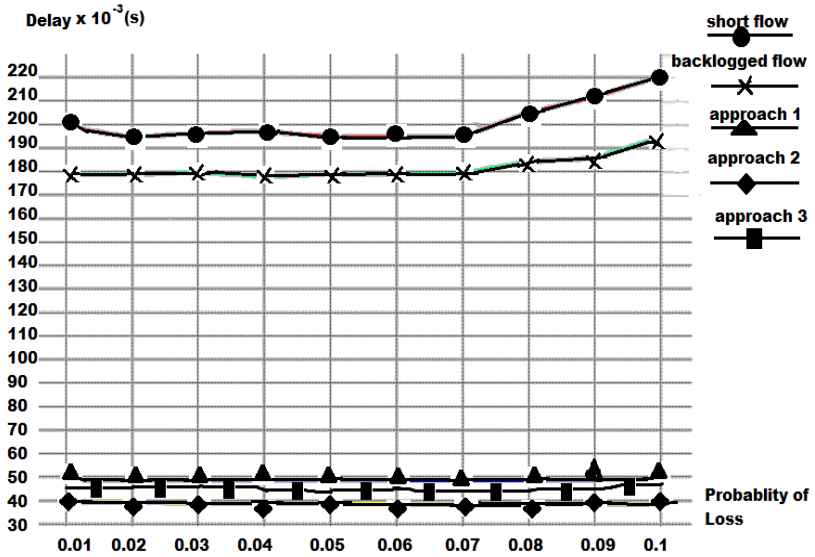

Fig. 3 Simulation: Delay as a function of loss rate

C. Implementation at the Network Level

To get even better performance we implement the padding approach at the network level in the gateway routers. Here the gateway router is configured to monitor all the flows coming from the network and also coming into the network. The gateway router tracks the number of packets that have been seen from each flow. The router also maintains a particular threshold value for classification of the flows as short and long. If this count exceeds a threshold, the flow is considered to be a long-lived flow. The router sends three dummy messages after each data packet to all the other flows that is the short flows so that their performance is improved

Implementing this approach in the gateway routers of each network has a number of advantages over the previous implementations. Implementation at the network level would help to reduce the work of the users as there is no need for each user to configure their systems independently. Also all the modifications can be made for the entire network at a common place. This would also help to reduce the amount of traffic in the network.

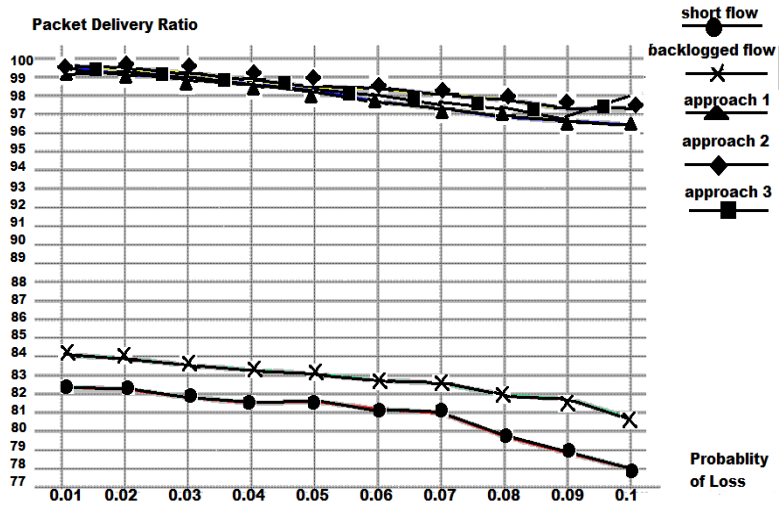

Fig. 4 Simulation: Packet Delivery Ratio as a function of loss rate

E. Performance Evaluation

Here we evaluate all the proposed approaches using extensive simulation experiments. We use the ns-2 simulator to obtain the various results. The topology consists of several networks having servers and clients and connected to each other using routers. TCP agents are attached to the sending and receiving nodes. We use ns-2's TCP/FullTcpAgent. We then define the capacities of various links and also vary the bottleneck link capacity from 1.5 to 10 Mbps. For interactive traffic, we open a telnet connection. The telnet client generates packets using an exponential distribution with average inter arrival time of 1 s. For fully backlogged TCP connections, we open FTP connections between a pair of nodes, one each from the server and the client pool. The three performance metrics used for comparison are Round Trip Time (RTT), Average End to End Delay and the Packet Delivery Ratio (PDR)

Fig. 2 plots the simulation results for the response time (the *y*-axis in the figure) as a function of the packet loss rate (the *x*-axis in the figure).We generate short flow, fully backlogged flow ,the three dummy packet flow and the duplicate copy flow from the source to the destination. The background traffic is composed of http, ftp, and low-rate udp traffic. For each of the flows, we log the experienced packet loss ratio and response-time. Then, we run the experiment multiple times and aggregate the results to plot the graph.

Fig. 2 clearly shows that the response time of the short TCP flows is much higher compared to the fully backlogged flows. This justifies the fact that the users of interactive applications always tries to upgrade their flows to fully backlogged flows. Approach 1 is the three dummy packet approach and approach 2 is the duplicate copy approach. But the key observation is that the three dummy packet and the duplicate copy approaches outperform the fully backlogged approach. In this way, two goals are achieved: 1) The interactive-application clients no longer have incentives to generate fully backlogged flows. Indeed, why convert to fully backlogged when the proposed approaches are better. 2) The new approaches also preserve the idea of statistical multiplexing.

For further performance analysis we calculate the average end to end delay and the Packet Delivery Ratio of the two approaches. Fig. 3 shows that the delay of flows of the proposed approaches is much lower compared to the fully backlogged approach. It also shows that the short TCP flows experience a higher delay compared to all the other flows. Fig. 4 clearly shows that the Packet Delivery Ratio of both the approaches is much higher compared to the fully backlogged approach and the short TCP flows. These results adds to the fact that the two proposed approaches have better performance than the fully backlogged approach without causing serious congestion in the Internet. From Fig.2,fig 3,fig 4 we can observe that the padding approach implemented at the network level as the approach three has better performance compared to all the previous approaches.

.

VI. CONCLUSION

Reasons for performance degradation of short TCP flows when multiplexed with long lived flows in the Internet were discussed in this paper. We demonstrated the response time performance degradation of the short TCP flows compared to the long flows over a shared link. The various reasons for the users to upgrade their short flows to long flows or the fully backlogged flows were explained in detail. Problems caused due to this approach on the Internet were pointed out. The problem is imminent because this misbehavior of upgrading to fully backlogged flows is hard to detect, given that flows are TCP friendly. The problem is serious because it has the potential to jeopardize one of the core principles that today's Internet is built upon—statistical multiplexing. We also showed that all the users of interactive applications always have an incentive to send at a TCP-fair rate, because the corresponding response-time performance always outperforms the pure interactive approach. Finally we demonstrated two easy to implement approaches that can be used to get better performance for the short TCP flows without causing any congestion. Application level and network level implementation of this approaches were also discussed. We showed that the two new approaches have a better performance than the fully backlogged approach and both the approaches are capable of de-motivating the clients from using the fully backlogged approach. Implementing these approaches in the gateway routers of each network would help to reduce the work of the users and also helps to make modifications for the entire network at a common place.

REFERENCES

[1] A.Mondal and A.Kuzmanovic, "Upgrading Mice to Elephants: Effects and End-Point Solutions ," IEEE/ACM Transactions on Networking, vol.18, no. 2, Apr. 2010.

[2] A.Mondal and A.Kuzmanovic, "When TCP friendliness becomes harmful," in Proc.IEEE INFOCOM, Anchorage, AK, May 2007,pp.152-160.

[3] L.Guo and I.Matta, "The war between mice and elephants," in Proc.9th IEEE International Conference on Network Protocols (ICNP), Riverside, CA, Nov. 2001, pp. 180–188.

[4] L. Le, J. Aikat, K. Jeffay, and F. Smith, "Differential congestion notification: Taming the elephants," in Proc. IEEE ICNP, Berlin, Germany, Oct. 2004, pp. 118–128.

[5] W. Noureddine and F. Tobagi," Improving the performance of interactive TCP applications using service diffrentiation ," in *Proc. IEEE, INFOCOM*, New York, Jun. 2002, vol. 1, pp. 31–40.

[6] X. Chen and J. Heidemann, "Preferential treatment for short flows to reduce web latency," Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 41, no. 6, pp.779-794, July 2002.

[7] M. Mellia, M. Meo, and C. Casetti, "TCP smart framing: a segmentation algorithm to reduce TCP latency," IEEE/ACM Transactions on Networking, vol.13, no. 2, pp.316-329, Apr. 2005.

[8] M. Mellia, M. Meo, and D.Ciullo, "Two schemes to reduce latency in short lived TCP Flows," IEEE Communications Letters, vol.13, no. 10, Oct. 2009.

[9] S. Savage, N. Cardwell, and Anderson, "The case for informed transport protocols," in Proc. HotOS, Rio Rico, AZ, Mar. 1999, p. 58.

[10] B. Kim and J. Lee, "Retransmission loss recovery by duplicate counting," IEEE Communications Letter, vol. 8, no. 1, pp.69–71, Jan. 2004.

[11] L. Le, J. Aikat, K. Jeffay, and F. Smith, "The effects of active queue management on Web performance," in Proc. ACM SIGCOMM, Karlsruhe, Germany, Aug. 2003, pp. 265-276.

[12] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's initial window," Internet RFC 3390, 2002.

Varun G Menon is currently working as an Assistant Professor at M.E.T'S School of Engineering, Kerala. He has done his M.Tech degree in Computer and Communication Engineering from Karunya University, Tamil Nadu India and received his B.Tech degree from Anna University TamilNadu India. His main research interest includes networking, network security, wireless networks, use of transport layer protocols in the Internet.