Comparative Study of High-Speed TCP Variants in Multi-Hop Wireless Networks

Mohit P. Tahiliani, K. C. Shet, and T. G. Basavaraju

***Abstract*—Multi-hop wireless networks have evolved as a key and promising wireless technology for a large variety of applications ranging from home networking to transportation systems, defense and medical systems. Transmission Control Protocol (TCP) is a widely deployed transport protocol and its congestion control mechanisms guarantee reliable delivery of data and efficient allocation of network resources. Congestion control mechanisms implemented in TCP have evolved significantly to better the performance of TCP on different types of communication networks. Recently, a lot of research has focused on improving the performance of TCP connections with large congestion windows, resulting in new variants called "*high-speed*" TCP variants. In this paper, we study the performance of high-speed TCP variants in multi-hop wireless networks in terms of network throughput. Another metric, *expected throughput* is used for comparison of throughput when nodes are mobile.**

***Index Terms*—Multi-hop wireless networks, TCP congestion control, high-speed TCP variants.**

I. Introduction

Wireless internet has become popular in recent years due to the tremendous growth in the number of mobile computing devices and high demand for continuous network connectivity regardless of physical locations. Multi-hop wireless networks such as Wireless Mesh Networks (WMNs), Mobile Ad-hoc Networks (MANETs), etc have emerged as a promising wireless technology for a large variety of applications. Applications of multi-hop wireless networks range from broadband home networking, community networking and enterprise networking to medical systems, security surveillance systems, transportation systems, defense and building automation [1].

TCP has been widely adopted as a reliable data transfer protocol for most of the communication networks. However, effectively and fairly allocating resources of a network (e.g. bandwidth) among a collection of competing users are major issues for all types of communication network.

A network is said to be *congested* when the traffic offered to it exceeds the available capacity [2]. Van Jacobson [3] laid the cornerstone for congestion control research. He proposed a new principle called "Conservation of Packets", which means that a new packet is not injected into the network until an old packet leaves the network. This principle leads to the formation of a key mechanism called "Self-Clocking", which means that the source uses acknowledgements (ACKs) as a clock to determine when to send new packets into the network.

Van Jacobson proposed three algorithms for congestion avoidance and control: Slow-Start, Congestion Avoidance and Fast Retransmit. Slow-Start algorithm is designed to start the Self-Clocking mechanism. This algorithm quickly fills the empty pipeline (network is viewed as a pipeline) at the beginning of transmission or after a retransmission timeout to bring the connection towards its equilibrium (a connection is said to be in equilibrium if it is running stably with a full window of data in transit). Congestion Avoidance algorithm, also known as Additive Increase/Multiplicative Decrease (AIMD) algorithm, closely obeys the "Conservation of Packets" principle once the connection is in equilibrium. Fast Retransmit algorithm considers duplicate acknowledgements [4] as a sign of packet loss in the network and retransmits the lost packet without waiting for a retransmission timer to expire. Since then, TCP congestion control mechanisms have undergone several modifications to improve the performance of TCP on different types of communication networks [5].

Recent work in the area of congestion control focuses on improving the performance of TCP connections with large congestion windows (*cwnd*) [6]. The improvements are focused on enhancing the basic mechanism of AIMD to efficiently maintain the connection at equilibrium. In AIMD mechanism, TCP sender updates the congestion window (*cwnd*) if an ACK is received or if the congestion is detected. For each ACK received, *cwnd* is updated as

$$cwnd \leftarrow cwnd + \frac{1}{cwnd} \quad (1)$$

This is known as Additive Increase phase of the AIMD algorithm. When congestion is detected either through timeout or duplicate acknowledgements (*dupacks*) [4] or Selective Acknowledgements (SACK) [4], *cwnd* is updated as

$$cwnd \leftarrow \frac{cwnd}{2} \quad (2)$$

This is known as Multiplicative Decrease phase of the AIMD algorithm. In large congestion windows, the *time taken to reach the same sending rate following the detection of congestion* may be in orders of minutes [2]. This conservative approach of AIMD may lead to under utilization of the available resources and hence result in significant performance degradation in the network.

Recently, different approaches have been proposed to address this drawback. One class of approaches optimizes the increase/decrease parameters of AIMD algorithm. These approaches are known as loss-based approaches [7] since

Manuscript received November 18, 2012; revised March 8, 2013.

Mohit P. Tahiliani and K. C. Shet are with the National Institute of Technology Karnataka, Surathkal, Mangalore, 575025, India (e-mail: tahiliani.nitk@gmail.com, kcshet@nitk.ac.in).

T. G. Basavaraju is with the Govt. SKSJ Technological Institute, Bangalore, 560001, India (e-mail: tg.braju@gmail.com).

they use packet loss as the indication of congestion, e.g., HighSpeed TCP (HSTCP) [8], Scalable TCP (STCP) [2], Binary Increase Congestion Control (BIC) TCP [9] and CUBIC TCP [10]. Another class of approaches uses Round Trip Time (RTT) variations as network congestion estimator and accordingly reduces the transmission rate. These approaches are known as delay-based approaches [7], e.g., FAST [11]. Loss-based approaches are aggressive as compared to delay-based approaches because the latter approach reduces the sending rate to avoid self-induced packet losses [7]. Compound TCP (CTCP) is a synergy of loss-based and delay-based approach [7].

The characteristics of wireless networks largely differ from those of a wired network. Data transfer rate in wireless networks is still restricted to a few mega bits per second (Mbps). Moreover, non-congestion packet losses due to reasons such as transmission errors, collisions, link failures and handoffs further complicate the design and development of congestion control mechanisms.

In this paper we study the performance of HSTCP, STCP, CUBIC and CTCP in multi-hop wireless networks. CUBIC-TCP, an enhanced version of BIC-TCP, is used as the default TCP in modern Linux operating systems [10], including Android. It overcomes the RTT unfairness problem [9] in HSTCP and STCP. CTCP, though disabled by default, is implemented in Microsoft Windows Vista and Windows 7 [12].

We analyze the performance of above mentioned high-speed TCP variants by varying the routing protocols in static as well as mobile topologies. The performance is measured in terms of overall network throughput. In mobile topologies, a metric called *expected throughput* [13] is used to analyze the performance of TCP variants.

The remainder of the paper is organized as follows. In Section II we discuss the congestion control mechanisms of each high-speed TCP variant. In Section III we brief about the wireless routing protocols. Section IV presents different simulation environment designed for multi-hop wireless networks and describes the performance metrics in detail. Section V discusses the simulation results. Section VI gives conclusion and possible future directions.

II. High-Speed TCP Variants

A. High Speed TCP (HSTCP)

High speed TCP is proposed in [8] to improve the performance of TCP connections with large congestion windows. HSTCP introduces a relation between an average congestion window and packet drop rate. If packet drop rates are more than 10^{-3} HSTCP follows the basic AIMD algorithm by increasing *cwnd* as in (1) and by decreasing *cwnd* as in (2). However, for packet drop rates less than 10^{-3}, HSTCP adopts a more aggressive increase/decrease algorithm: When an ACK received, *cwnd* is updated as

$$cwnd \leftarrow cwnd + \frac{a(cwnd)}{cwnd} \quad (3)$$

and when congestion is detected, *cwnd* is updated as

$$cwnd \leftarrow cwnd - b(cwnd) \times cwnd \quad (4)$$

The canonical values for a and b are 1 and 0.5 respectively. As the *cwnd* size increases beyond certain threshold, the value of b decreases from 0.5 to 0.1, while the value of a increases accordingly [7]. A detailed study of HSTCP is presented in [14].

The major drawbacks of HSTCP are: Round Trip Time (RTT) unfairness problem and TCP unfariness problem. RTT unfairness is defined as the ratio of *cwnd* in terms of RTT ratio of multiple TCP connections [9]. Moreover, the performance of regular TCP flows is largely affected by the aggressiveness of HSTCP. This is known as TCP-unfairness [15].

B. Scalable TCP (STCP)

Scalable TCP is quite similar to HSTCP's aggressive increase/decrease algorithm. However, the increase/decrease parameters in STCP are *constant* rather than HSTCP's parameterization by the current congestion window [2]. When an ACK received, *cwnd* is updated as

$$cwnd \leftarrow cwnd + a \quad (5)$$

and when congestion is detected, *cwnd* is updated as

$$cwnd \leftarrow cwnd - (b \times cwnd) \quad (6)$$

The values of a and b are *fixed* to 0.01 and 0.125 respectively. The motivation behind the choice of values 0.01 and 0.125 is described in [2]. The authors in [2] claim that the time taken by STCP source to double its sending rate is about 70 RTTs for any rate and hence the proposed algorithm is scalable. However, like HSTCP, RTT unfairness problem and TCP-unfairness problem are major drawbacks in STCP as well [9].

C. CUBIC TCP

CUBIC TCP is an enhanced version of Binary Increase Congestion Control (BIC) TCP. BIC TCP, proposed in [9], focuses on solving the RTT unfairness problem. It combines two algorithms called additive increase and binary search increase. Additive increase ensures linear RTT fairness when *cwnd* is large and binary search increase ensures TCP-friendliness when *cwnd* is small. Binary search increase algorithm is described in detail in [9].

CUBIC further improves the performance of BIC TCP with respect to RTT unfairness problem by incrementing *cwnd* independent of RTT [10]. During steady state, CUBIC increases *cwnd* size aggressively if it is far from equilibrium and slowly when it is close to equilibrium [10]. However, TCP-unfairness problem is not addressed by CUBIC TCP.

D. Compound TCP (CTCP)

Compound TCP is a synergy of loss-based and delay-based congestion avoidance approaches [7][12]. It addresses the RTT unfairness problem and TCP-unfairness problem by adding a new scalable delay-based component to the standard TCP. This delay-based component acts as an auto-tuning knob [7] by rapidly increasing *cwnd* when the network is under-utilized and gracefully decreases *cwnd* once the congestion is detected.

CTCP retains the basic Slow Start and Congestion Avoidance phases. During congestion avoidance phase, the

update of *cwnd* is based on (1) and (2). However, CTCP may send (*cwnd* + *dwnd*) packets in one RTT (instead of 1 packet) where *dwnd* represents the delay window which controls delay-based component. Therefore, the update of *cwnd* when an ACK is received is modified accordingly:

$$cwnd \leftarrow cwnd + \frac{1}{(cwnd + dwnd)} \quad (7)$$

A detailed explanation regarding the delay-based component is provided in [2].

III. Routing Protocols in Wireless Networks

Routing protocols in multi-hop wireless networks are mainly classified as:

A. Table Driven Routing Protocols

These protocols maintain consistent, up-to-date routing information from each node to every other node in the network. These nodes maintain routing tables and respond to the changes in the network topology by propagating updates throughout the network to maintain a consistent view of the network.

Different table-driven routing protocols differ in the number of routing tables and the methods by which changes in the network topology are broadcasted. Examples of table driven routing protocols are: Destination Sequenced Distance Vector (DSDV), Optimized Link State Routing (OLSR), etc.

In DSDV routing protocol, each node in the network maintains a routing table that consists of available destinations and the number of hops needed to get to each of them. Each entry in the route table is tagged with a sequence number that is originated by the destination node. These sequence numbers distinguish the stale routes from the new ones and thus avoid routing loops. A more detailed description about DSDV is presented in [16].

The disadvantages of table driven routing protocols are: such protocols require more memory to maintain the routing information and they react very slowly on restructuring or route failure in the network.

B. Demand Driven Routing Protocols

These protocols create routes only when desired by the source. There are two main phases in demand driven routing protocols: route discovery and route maintenance. The source node initiates route discovery when it requires a route to the destination. This process is completed when a route is found or when all the possible routes are examined. The process of route maintenance is carried out to maintain the established routes until either the destination becomes unavailable or when the route is no longer required. Examples of demand driven routing protocols are: Ad hoc on demand Distance Vector (AODV), Dynamic Source Routing (DSR), etc. We choose both AODV and DSR since they are widely accepted as the standard demand driven routing protocols.

AODV routing protocol as described in [16] is a modified version of the DSDV and aims at reducing system wide broadcasts that are a feature in DSDV. Routes are discovered only when there is a demand and are maintained only as long as they are necessary. Each node maintains monotonically increasing sequence numbers and this number increases as it learns about a change in the topology of its neighborhood. This sequence number ensures that the most recent route is selected whenever route discovery is initiated. This protocol is used for unicast, multicast and broadcast communication.

DSR is a simple and efficient routing protocol [16] similar to AODV except that in DSR, each data packet sent carries in its header, the complete ordered list of nodes through which the packet must pass to reach destination. Since the source route is included in the header, other nodes hearing this transmission can cache this information in their routing table for future use.

The disadvantages of demand driven routing protocols are: they incur initial delay in establishing the route before sending the actual data packets and they induce more control overhead in scenarios where route failures are not frequent.

IV. Simulation Setup and Methodology

The results in this paper are based on the simulations done on ns-2, a discrete event simulator [17]. We have chosen static as well as mobile topologies for the study.

A. Static Topologies

We have designed a linear string topology of 8 nodes, similar to that in [15]. We consider a single TCP connection that covers a variable number of hops, from 1 to 7. The nodes are configured to use 802.11 MAC protocol. The distance between two nodes is equal to the transmission range which is by default set to 250 meters. The channel data rate is 11 Mbps. TCP packet size is fixed to 1500 bytes. Keeping all the above mentioned parameters fixed we switch TCP variant and the routing protocol. Simulation results are discussed in Section V.

B. Mobile Topologies

In mobile topologies we designed a network model consisting of 30 nodes in a 1500 × 300 meter flat, rectangular area. Our network model is analogous to the one in [8]. The mobility patterns are generated using the mobility pattern generator provided in ns-2. This generator is designed based on random waypoint mobility model. The mean speed with which nodes move is 10 m/s. We generate 25 such mobility patterns and our simulation results are based on the average throughput of these 25 mobility patterns. Other parameters are same as mentioned above for static topologies. Simulation results are discussed in Section V.

C. Performance Metric

The performance metric used in our study is throughput. In static topologies we measure the throughput of TCP connection and compare the changes observed on increasing the number of hops (from 1 to 7).

But in mobile topologies the distance between the source and destination keeps varying. The number of hops on the path from source to destination may increase or decrease. Hence, we use another performance metric called expected throughput as defined in [13]. It is calculated as follows:

Let T_i denote the throughput obtained for the string topology, where i denotes the number of hops and $1 \leq i \leq \infty$. When $i = \infty$ it means that the network is partitioned and hence throughput $T_\infty = 0$. Let t_i be the duration for which the shortest distance between source and destination in mobile topology

is i hops ($1 \leq i \leq \infty$). The expected throughput is then calculated as:

$$\frac{\sum_{i=1}^{\infty}(t_i \times T_i)}{\sum_{i=1}^{\infty}(t_i)} \quad (8)$$

The throughput measure obtained by simulations is called actual throughput. This actual throughput is then compared with the expected throughput.

V. Results and Analysis

A. Static Topologies

Table I shows the throughput (in Kbps) obtained for each high-speed TCP variant with DSDV routing protocol. Table II and Table III show the throughput obtained for each high-speed TCP variant with AODV and DSR routing protocol respectively.

Throughput decreases as we increase the number of hops for all high-speed TCP variants as shown in the tables. Our studies show that all high-speed TCP variants consistently perform better with DSDV. One of the reasons is that DSDV maintains a routing table and hence avoids the initial delay in discovering a route as in AODV and DSR.

Comparing throughput values of AODV with DSR, it is observed that better throughput is achieved by all high-speed TCP variants with AODV. The reason behind DSR's poor performance is that the data packet carries entire route information from source to destination in its header. This leads to severe degradation of throughput as the number of hops increase. It can be seen in Table II and III, when H > 2, the throughput values for DSR decrease drastically.

STCP gives the best performance when DSDV routing protocol is used because it is far more aggressive than other high-speed TCP variants due to its *fixed* increase/decrease parameters. Also the packet drop rate is low since the topology is static in nature and hence the overhead of control packets (routing packets) is minimal in DSDV. This allows STCP to reach the equilibrium aggressively in less number of RTTs and thus achieve better throughput. However, when AODV and DSR are used, the packet drop rate increases due to increase in the control overhead (DSR has less control overhead than AODV because DSR maintains entire route information in its cache). Since STCP's increase/decrease parameters are fixed, its aggressiveness leads to frequent packet drops and thus degrades the overall throughput of the network. HSTCP performs better than STCP when AODV and DSR routing protocols are used because it varies the increase/decrease parameters depending on packet drop rate.

TABLE I: Throughput (in Kbps) Using DSDV

No. of Hops, H	HSTCP	STCP	CUBIC	CTCP
1	2995.20	2995.39	2994.32	2995.20
2	1508.34	1508.23	1508.69	1508.34
3	899.86	902.53	901.49	899.86
4	681.79	691.83	682.46	683.16
5	594.01	558.24	595.59	544.61
6	516.79	541.37	539.00	541.74
7	395.05	450.74	449.46	424.92

TABLE II: Throughput (in Kbps) Using AODV

No. of Hops, H	HSTCP	STCP	CUBIC	CTCP
1	2996.51	2996.51	2996.51	2996.51
2	1507.82	1507.82	1507.82	1507.82
3	898.50	912.42	904.04	903.92
4	612.29	603.51	616.63	612.99
5	539.44	512.88	540.04	528.53
6	478.06	447.65	487.99	486.23
7	398.80	391.00	382.66	417.07

TABLE III: Throughput (in Kbps) Using DSR

No. of Hops, H	HSTCP	STCP	CUBIC	CTCP
1	2996.42	2996.42	2997.22	2996.42
2	1507.46	1508.44	1508.10	1507.46
3	817.21	824.51	862.09	817.21
4	603.66	570.67	616.62	587.38
5	432.86	401.08	458.55	416.92
6	377.98	344.28	413.15	374.08
7	377.10	339.59	379.55	361.48

When DSR routing protocol is used, CUBIC TCP gives the best performance. With DSDV, the performance of CUBIC is similar to that of STCP. DSDV and DSR have low control overhead as compared to AODV. Since AODV has the highest amount of control overhead, more packets are dropped due to collision and hence CUBIC increases *cwnd* slowly rather than aggressively. Thus CUBIC achieves least throughput with AODV routing protocol.

When AODV routing protocol is used, CTCP gives the best performance because of its delay-based component. This delay-based component reduces the sending rate to avoid packet drops caused due to increased control overhead in AODV. HSTCP, STCP and CUBIC do not reduce the sending rate till a packet drop occurs and thus increasing *cwnd* aggressively, results in more packet drops and degradation in throughput.

B. Mobile Topologies

TABLE IV: Throughput (in Kbps) Using DSDV

TCP Variant	Expected Throughput	Actual Throughput	Percentage Achieved
HSTCP	1385.606	872.5324	62.97
STCP	1387.243	799.5872	57.63
CUBIC	1390.356	645.1164	46.40
CTCP	1382.826	884.4412	63.95

TABLE V: Throughput (in Kbps) Using AODV

TCP Variant	Expected Throughput	Actual Throughput	Percentage Achieved
HSTCP	1365.179	1137.103	83.29
STCP	1359.197	1145.236	84.25
CUBIC	1367.008	1170.215	85.60
CTCP	1366.152	1149.400	84.13

TABLE VI: Throughput (in Kbps) Using DSR

TCP Variant	Expected Throughput	Actual Throughput	Percentage Achieved
HSTCP	1326.714	1310.200	98.75
STCP	1314.514	1308.454	99.53
CUBIC	1342.538	1272.218	94.76
CTCP	1321.394	1315.010	99.51

Table IV to VI show the expected throughput, actual throughput (in Kbps) obtained and the percentage of expected throughput achieved for each high-speed TCP variant with DSDV, AODV and DSR respectively.

We observe that the actual throughput obtained for high-speed TCP variants with DSR is almost similar to expected throughput. Actual throughput obtained with AODV is lower than DSR but higher than DSDV. The basic reason why DSDV results in least throughput is explained as follows:

Mobility causes frequent route failures. Table driven routing protocols like DSDV do not send the data packets till the routing table is updated with new routes. This introduces large delays and thus degrades the overall throughput of the network. Demand driven routing protocols like AODV and DSR do not maintain a routing table and hence, in case of frequent route failures, do not induce large delays. Thus the performance of high-speed TCP variants degrades with DSDV.

CTCP performs better than other variants when DSDV is used. This is because CTCP has a delay-based component that avoids packet drops and consequently increases the throughput. The performance of all high-speed TCP variants is almost similar when AODV and DSR are used. CUBIC and STCP give slightly better performance with AODV and DSR respectively.

VI. Conclusions and Future Work

Through simulations we have studied the behavior of high-speed TCP variants in multi-hop wireless networks by varying the routing protocols such as Destination Sequenced Distance Vector (DSDV), Ad hoc On demand Distance Vector (AODV) and Dynamic Source Routing (DSR) routing protocols. We have evaluated the performance of high-speed TCP variants in terms of throughput for static as well as mobile topologies. It is observed that the performance of TCP largely depends on routing protocols.

Each routing protocol varies in the way it reacts to link failures. Routing protocols also differ in the way they form the routes. More routing overhead reduces the overall throughput of the network. More number of collisions due to increased routing overload makes the situation worse for TCP performance.

In this study we have not considered the effects of non-congestion losses on the performance of high-speed TCP variants. Also the performance of high-speed TCP variants is not studied with respect to parameters such as Convergence speed, RTT fairness and TCP fairness. In future, we intend to study the performance of high-speed TCP variants with above mentioned parameters and also the effects of non-congestion losses on the performance of TCP.

References

[1] I. F. Akylidiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks*, Elsevier, pp. 445-487, January 2005.

[2] T. Kelly, "Scalable TCP: Improving Performance in High Speed Wide Area Networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83-91, 2003.

[3] V. Jacobson, "Congestion avoidance and control," *Proceedings of SIGCOMM '88,* ACM, Stanford, CA, Aug. 1988.

[4] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP," *ACM Computer Communication Review*, vol. 26, no. 3, pp. 5-21, 1996.

[5] S. Floyd and T. Henderson, "The newreno modification to TCP's fast recovery algorithm," Request for Comments 2582, Experimental, April 1999.

[6] L. Brakmo and L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, pp. 1465-1480, Oct. 1995.

[7] K. T. J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proceedings of PFLDNet*, 2006.

[8] S. Floyd, "Highspeed TCP for Large Congestion Windows," Request for Comments 3649, Experimental, 2003.

[9] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for fast long-distance networks," in *Proceedings of IEEE INFOCOM*, Hong Kong, 2004.

[10] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *Proceedings of the third PFLDNet Workshop*, France, 2005.

[11] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM*, Hong Kong, 2004.

[12] *TCP Evolution and Comparison, Which TCP will Scale to Meet the Demands of Today's Internet?* Whitepaper, FastSoft, Pasadena, 2008.

[13] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," *ACM/IEEE MOBICOM '99,* Seattle, Washington, Aug. 1999.

[14] E. D. Souza and D. Agarwal, "A HighSpeed TCP Study: Characteristics and Deployment Issues," LBNL Technical Report, Berkeley, 2003.

[15] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multi-hop Networks," in *Proceedings of IEEE WMCSA '99,* New Orleans, LA, Feb. 1999.

[16] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu and J. Jetcheva, "A Performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. ACM/IEEE Int. Conf. on Mobile Computing and Networking*, pp. 85-97, October 1998.

[17] K. Fall and K. Vardhan, "The *ns* Manual," The VINT Project, January 2009.

Mohit P. Tahiliani completed his B.E and M.Tech in Computer Science and Engineering from Visvesvaraya Technological University, Belgaum, India in the year 2007 and 2009 respectively. Currently he is pursuing Ph.D at National Institute of Technology Karnataka, Surathkal, India. His areas of interest include: Congestion control algorithms, Active Queue Management (AQM) mechanisms and Routing protocols for Multi-hop wireless networks. He is a Student Member of IEEE and has served as Reviewer for ADCONS 2011, ITCS 2012 and Student Research Symposium Chair for ADCONS 2011. He has also delivered several Talks in National / International Conferences and Workshops.

K. C. Shet completed his B.E, M.Sc. Engg and Ph.D in Electronics and Communication from Mysore University, Sambalpur University and IIT Bombay, India in the year 1972, 1979 and 1989 respectively. Currently he is working as a Professor in Department of Computer Science and Engineering at National Institute of Technology Karnataka, Surathkal, India. His areas of interest include Software testing, Wireless networks, Securing web services and Anti spam solutions. He has published several papers in International Journals and Conferences and has also served as Reviewer, Keynote Speaker and Chair for several reputed conferences.

T. G. Basavaraju completed his B.E, M.E and Ph.D in Computer Science and Engineering from UBDT Davangere, UVCE, Bangalore and Jadavpur University, Kolkata respectively. Currently he is working as Professor and Head in Department of Computer Science and Engineering at SKSJ Technological Institute, Bangalore, India. His areas of interest include Ad hoc and Sensor Networks, Mesh Networks and Network Management. He has authored several textbooks on Computer Networks and has also published several papers in International Journals and Conferences. He has also served as Reviewer, Keynote Speaker and Chair for several reputed conferences.