# Load Balancing in Internet Using Adaptive Packet Scheduling and Bursty Traffic Splitting

**M. Azath**
Research Scholar, Anna University,
Coimbatore, India

**Dr.R.S.D.Wahida Banu**
Research Supervisor, Anna University,
Coimbatore, India

**Summary**
In this paper, we propose an architecture for load balancing, which contains an adaptive packet scheduler with a bursty traffic splitting algorithm. The scheduler has one classifier which classifies the flows into aggressive and normal flow. Aggressive flows are treated as high priority flows. Based on the buffer occupancy threshold, a trigger handler checks for load un-balance of the network and automatically triggers the load adapter. The load adapter reroutes the high-priority aggressive flows into the least loaded best path, using the bursty traffic splitter algorithm. The bursty traffic splitting algorithm splits the aggressive flows over multiple parallel paths, based on a split vector. In this algorithm, instead of switching packets or flows, it switches packet bursts. Since the packet bursts are smaller in size, the algorithm splits the traffic dynamically and accurately. At the same time, the condition forced on their latency difference, ensures that no packets are reordered. To achieve fair bandwidth allocations, load balancing is attained in the system since the high-rate aggressive traffic flows are splitted along multiple parallel paths. The proposed switching technique is executed in the edge and core routers. We will show by simulations that our adaptive packet scheduler performs better than the standard fair-queuing techniques.

***Keywords:***
*Load Balancing, Splitting, Bursty Traffic, Aggressive Flows, Scheduler*

## 1. Introduction

The performance optimization of operational networks is referred to as Traffic engineering. Traffic presented between origin and destination nodes, loads the network in one way and alternatively, this traffic has to be carried in the network in a way that performance intentions are satisfied.

So as to obtain optimal resource utilization, throughput, or response time [1], load balancing is a method in computer networking to spread work between two or more computers, network links, CPUs, hard drives, or other resources. Rather than a single component, multiple components with load balancing are utilized to enhance reliability through redundancy. A devoted program or hardware device will provide the balancing service.

In ISP networks, load balancing is common. It is a key constituent of traffic engineering, link bundling, and equal cost multi-path routing [2]. Moving traffic from overcrowded links to other parts of the network in a well-controlled way is the initiative of load balancing. The load balancing can be devised as an optimization problem, if the traffic stipulates are recognized. Though, knowledge of traffic stipulates is frequently deficient Current trends in load balancing are pointing towards dynamic protocols. The traffic of an ingress-egress router pair onto multiple paths is mapped by these protocols and to evade hot-spots, they acclimatize the share of each path in real-time and deal with failures. The schemes ripping traffic across multiple paths at a well granularity has been required by the dynamic load balancing.

However, to divide the traffic and their capability in shunning packet reorganizing, recent traffic splitting schemes display a fight among the granularities. The preferred load share to each path has been rapidly allocated through packet-based splitting. However, splitting at packet granularity can reorganize a huge number of packets at what time paths vary in delay. TCP mystifies this reordering as a sign of congestion and ensuing in degraded performance. Some UDP-based applications, like VoIP, are sensitive to packet reordering. Conversely, flow-based splitting pins every flow to a particular path and shirks packet reordering. However, flows vary extensively in their sizes and rates. A flow continues on the path all through its existence, if it is once assigned. Therefore, flow-based splitting may not succeed to rapidly re-balance the load in the face of altering demands [6] or allot erroneous amounts of traffic to every path. This lack of ability speedily respond to traffic spikes congests links and diminishes network good put.

To distribute load in network systems, hashing is a fashionable means. On the contrary to round-robin or minimum-load mapping, hashing is used in parallel IP forwarding systems due to its capability in sustaining the packet order of individual TCP connections. Hashing operates at *flow level*. Hashing only is not competent to balance workload under extremely changeable Internet traffic [6]. To accommodate the burstiness and the

Manuscript received October 5, 2008
Manuscript revised October 20, 2008

presence of enormously large flows, adaptive schemes are required.

In this paper, we propose architecture for load balancing which contains an Adaptive Packet Scheduler (APS) and Bursty Traffic Splitter. The APS contains the following components: Classifier, Trigger Handler and Load Adapter. The trigger handler checks for load un-balance of the network and automatically triggers the load adapter. The classifier first classifies the flows into aggressive and normal flow. Aggressive flows are treated as high-priority flows and rerouted into the best path, using the bursty traffic splitter algorithm. The bursty traffic splitting algorithm resides on the router and splits the aggressive flows over multiple parallel paths. In this algorithm, instead of switching packets or flows, it switches packet bursts. In this algorithm, instead of switching packets or flows, it switches packet bursts. Since the packet bursts are smaller in size, the algorithm splits the traffic dynamically and accurately. At the same time, the condition forced on their latency difference, ensures that no packets are reordered. Since the high-rate flows are splitted along multiple parallel paths, load balancing is attained in the system, there by achieving fair bandwidth allocations. The proposed switching technique is executed in the edge and core routers.

## 2. Related Work

Using a type of weighted round-robin or shortfall round robin [3], [4] scheduling, a few proposals for traffic splitting forward packets onto multiple paths in datagram networks. These schemes are not used actually, since they cause considerable packet reorganizing.

A distributed randomized scheme has been presented in [5], which constantly rebalances the lengths of intervals of a Distributed Hash Table based on a ring topology. They have confirmed that the scheme works with high probability. Moreover, its cost calculated in the number of migrated nodes is similar to the best probable. However, the appearing constants from the analysis are huge and the analysis demands the harmonization of the algorithm. Counting of nodes is one more issue mislaid here.

For resolving the memory performance problem in high-speed routers, a theoretical foundation has been set in [8]. It carries under a general umbrella numerous high-speed router architectures and introduces a general principle called “constraint sets” to scrutinize them. But this paper converses just memory issues which are not bursty bursting traffic.

For distributed data storage in P2P systems, they have specified numerous provably competent load balancing protocols in [10]. The notion of random load balancing is used in this.

Deeming two typical load balancing approaches - static and dynamic, the performance analysis of a variety of load balancing algorithms based on dissimilar parameters have been presented in [11]. The presence of advantages and drawbacks of both static and dynamic types of algorithm over each other has been specified through the analysis. Based on the type of parallel applications, the implementing algorithm types would be chosen to resolve. Through learning the behavior of various offered algorithms, serving in design of new algorithms in future is the major intention of this paper.

At what time the input traffic is modeled by a Markov-modulated Poisson process (MMPP), the split traffic routed by a probabilistic routing algorithm has been characterized in [12]. They have find out that every split traffic under such splitting mechanism is also an MMPP with a proper parameter modification which is investigated in this paper. They have applied the consequence to enable in obtaining network-wise performance procedures, continuous sojourn delay and delay variation in networks employing probabilistic routing algorithms through an estimated approach, later than having characterized the split traffic. But this solution is based on Packet-based splitting. So it can root huge amount of packets rearranging.

## 3. System Model

We consider a parallel forwarding system where $m$ disjoint paths $(P_1, P_2, ...., P_m)$ concurrently process the packets dispatched from the scheduler. The aggregate traffic arriving at a router, at rate r, is composed of a number of distinct transport-layer flows of varying rates. For the m disjoint paths $P_i$ and a split vector $f = (f_1, f_2, ....., f_m)$, where $f_i \in [0,1]$, find a packet-to-path assignment such that the traffic rate on $P_i$ is equal to the Filter $f_i \times R$. Let us suppose that the path $P_i$ contains $n$ nodes $(N_{i1}, N_{i2}, ....., N_{in})$. Then, a packet d estined to the node $N_{ik}$ is processed at once if $N_{ik}$ is idle; otherwise, it is stored in a shared buffer of size $B$ (in packets) in front of the node. Logically, the packet is also appended to the input queue of the node $N_{ik}$. Since the buffer size is fixed, the length of an input queue is between zero and the buffer size. At any time, the limit of a queue’s length depends on the number of packets in other queues.

A chief constituent of the load balancing problem is the traffic splitting. Balancing the load across multiple paths needs a mechanism to find the splitting vector $F$ as well as a traffic splitter. In case of balancing the load across a link bundle linking two routers, the splitting vector is static usually. Since decided by the adaptive packet scheduler, it will dynamically adopt to the congestion state along each path.

## 4. Adaptive Packet Scheduling

### 4. 1 Flow Classifier

Our system contains a flow classifier which divides internet flows into two categories: the aggressive and the normal. By applying different forwarding policies to the two classes of flows, the adaptive packet scheduler achieves load balancing effectively and efficiently.

We define aggressive flows as high-rate flows. Flows that are both large and fast are the source of long-term load imbalance and are most effective when shifted to balance load. These flows are similar to the alpha flows in [12]. In addition, taking the bursty nature of Internet traffic into consideration, we also classify flows that are smaller in size but are fast enough to cause short-term load imbalance or buffer-overflow as aggressive flows.

The incoming IP traffic is considered as a sequence of windows $W_1, W_2, .....$, each containing $w$ packets. For a receiving window $W_i$, we find the largest flows and form a set $FL_i$. Let $F$ be the size of the set $FL_i$. At the end of the window $W_i$, the flows stored in the flow table are replaced by the flows of $FL_i$. Due to the packet train [24] behavior of network flows, it is possible that some of the flows of $FL_i$ may appear in $FL_{i+1}$

$$\text{ie. } FL_i \bigcap FL_{i+1} \neq \{\}$$

Let $\pi_i = \left| FL_i \bigcap FL_{i+1} \right|$

Then we define $\phi = \dfrac{\sum_{i=1}^{n} \pi_i / F}{n}$

Where, $n = NoP/w$ and $NoP$ is the number of packets forwarded during the measurement. Thus, larger the value of $\phi$, the better flow information collected in the current window predicts aggressive flows for the next window.

### 4.2 Load Adapter

When the system is in a balanced state, packets flows are mapped to the normal path as per the bursty splitter algorithm. When the system is unbalanced, the triggering policy invokes the load adapter. The load adapter becomes active and may decide to override the decisions of the splitter. It checks each passing packet to see whether it belongs to one of the high-rate flows identified by the classifier. If the packet belongs to one of these flows, the load adapter reroutes these flows into the congestion free least loaded path.

An important design parameter is $F$, the size of the balancer's flow table. Generally, shifting more aggressive flows, i.e., having more flows in the table, is more effective as far as load balancing is concerned.

### 4.3. Triggering Policy

There are multiple choices for deciding when the system is unbalanced and the adapter should be activated to redirect packets. We use the *Buffer Occupancy Threshold* (BOT) for measuring the load unbalance. The adapter is invoked if the buffer is filled above some percentage.

Eliminate users whose queues are empty. Only active users, who have packets for transmission, are taken into the selection procedure. Check queue occupancies of the active users. If the queue sizes of these users exceed their buffer occupancy threshold, it signifies the cause of load unbalance and the load adapter is invoked. A fixed buffer allocation strategy is exploited which has the queuing threshold as 70% of total buffer spaces allocated to each flow.

## 5. Traffic Splitting Algorithm

### 5.1 Bursty Splitting

To avoid packet reordering, our bursty splitting algorithm combines the effectiveness of both packet-based and flow-based splitting.

A packet-burst is a burst of packets from a given transport layer flow.

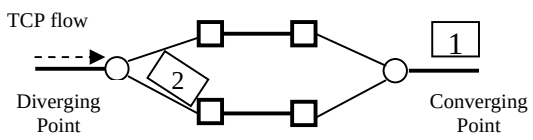


Fig.1 Packet flow on parallel paths

Consider the scenario in Fig.1 where a set of parallel paths diverge at a particular point and converge later. Each path

contains some number of hops. Given two consecutive packets in a flow, if the first packet leaves the convergence point before the second packet reaches the divergence point, one can route the second packet and subsequent packets from this flow on to any available path without of reordering. Let $T$ be the maximum latency difference of the parallel paths. So if we choose the timeout value of the packet-burst such that $t_i > T$, consecutive packet-bursts can be switched independently without reordering.

### 5.2 Load Splitting and Routing

The bursty splitting algorithm is a traffic splitting mechanism executed on a router which accepts multiple parallel paths. When a packet is received, the algorithm forwards the packet to the best path, by accepting a split vector as input. The mechanism is based on abstraction to dynamically and accurately split traffic along multiple paths without reordering. The main functions of the algorithm are:

(a) **Estimating Latency Difference:** To estimate the latency difference between the traffic splitted parallel paths, the algorithm uses periodic pings. It sets the packet-burst timeout value $t_i$ to $T$, where $T$ is the maximum latency difference between the parallel paths. So if we choose $t_i > T$, packet-bursts of the same flow can be switched independently without packet reordering.

(b) **Path Assignment:** It uses a hash table that maps packet-bursts to paths. On packet arrival, the algorithm computes the 16-bit hash value as

Hash ( src_ip, dest_ip , src_port, dest_port)

This hash is used as the key into the packet-burst table. The table contains the following two fields: $P_{id}$ and $L_t$, where $P_{id}$ is the path id and $L_t$ is the last seen time. Let $C_t$ is the current time. A token $tc_i$ is maintained for each path $P_i$, to estimate the load deviation of the path. The following algorithm illustrates the path assignment process.

1. If $C_t < (L_t + t_i)$, then,

1.1 Send packet through the path $P_i(P_{id})$

1.2 $L_t = C_t$.

2. Otherwise,

2.1 Find the path $P_{\max}$, with $tc_i = \max(tc_i)$

2.2 $P_i(P_{id}) = P_{\max}(P_{id})$

2.3 $L_t = C_t$.

2.4 Send packet through the path $P_{\max}(P_{id})$

(c) **Token-counting algorithm:** To estimate the load deviation of the path, a token $tc_i$ is maintained for each path $P_i$ of the parallel paths.

For every packet of size $bw$ bytes, all tokens are updated as follows:

$tc_i = tc_i + B_i \times bw \qquad ,\mathrm{i} = 1,2,.....$

Where $B_i$ is the fraction of the load to be sent on path $P_i$.

The packet is assigned to a path according to the path assignment algorithm. Once the packet has been assigned to a particular path $P_i$, the corresponding token is decremented by the size of the packet:

$$tc_i = tc_i - bw$$

The tokens successfully track the load assigned to a path. When a path gets a share $B_1$ which is less than the desired load $B$, then the token is calculated as,

$$tc_i = B_1 - B$$

Whenever a new packet-burst arrives, it is assigned to the path with the maximum number of remaining tokens. The tokens are reset for every measurement interval $t_m$.

## 6. Experimental Results

### 6.1 Simulation Setup

We simulated the design of our markers with the *ns-2* [14] network simulator 1. The topology used in our experiments is depicted in Figure 2.

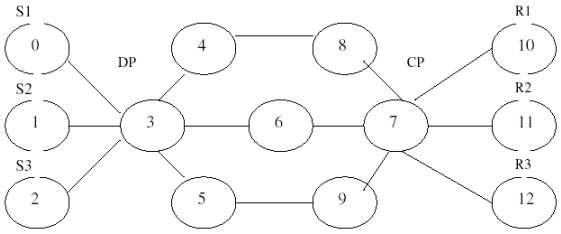


Fig.2 Simulation Topology

The topology consists of 3 senders S1, S2, S3 and 3 receivers R1, R2, R3. The senders are connected to a router DP, which is a divergent point. The receivers are connected to a router CP, which is a convergent point. There are 3 parallel paths $P_1 = \{3,4,8,7\}$, $P_2 = \{3,6,7\}$ and $P_3 = \{3,5,9,7\}$. The requested bandwidth of the senders S1, S2, S3 are 10Mb,8Mb and

5Mb, respectively. Different link bandwidth and delay are set for the 3 paths.

## 6.2 Simulation Parameters

We have taken the metrics received bandwidth and packet loss for evaluation. We compared our results with the standard queuing techniques CSFQ [4] and RED [15]. In our experiments, we vary the buffer size, traffic rate and packet size. The results are described in the next section.

## 6.3 Results

### A. Buffer Size

In the first experiment, we set the buffer size of the router as 50, and measure the received bandwidth. Figure 3, gives the result of 3 schemes in various time intervals. From the figure, we can see that, our APS algorithm has more received bandwidth than the CSFQ and RED schemes. Similar result is achieved in figure 4 for buffer size 100.

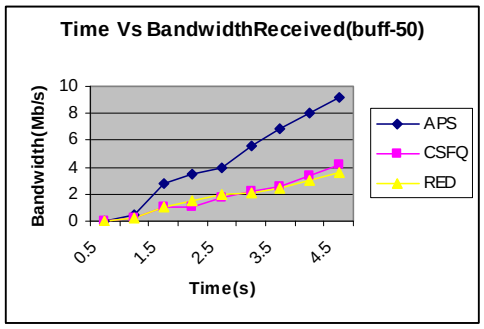


Fig. 3 Time Vs Bandwidth (for buffer size 50)

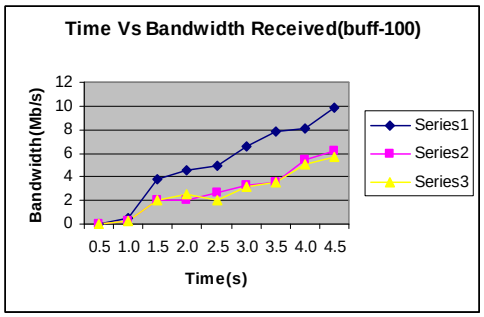


Fig. 4 Time Vs Bandwidth (for buffersize100)

### B. Packet Size

In the next experiment, we vary the packet size as 250,500,750 and 1000 and measure the received bandwidth and packet loss. Figure 5 shows that, the packet loss is more in RED, followed by CSFQ and is the least in APS. Figure 6 shows that the received bandwidth is more for APS, than CSFQ and RED.

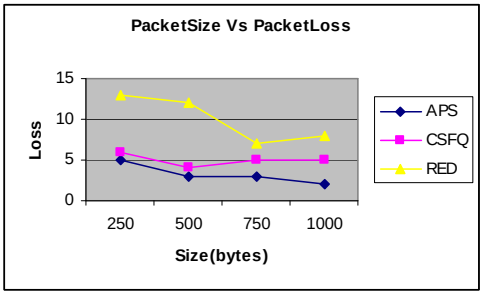


Fig. 5 Packet Size Vs Packet Loss

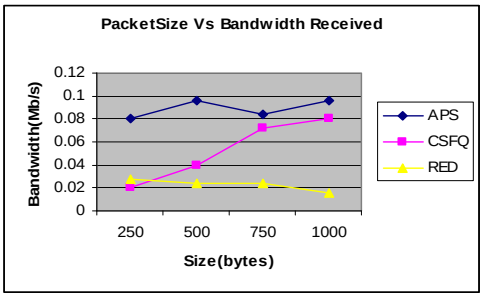


Fig. 6 Packet Size Vs Bandwidth

### C. Traffic Rate

In the next experiment, we vary the traffic sending rate as 100,200…500 (kb) and measure the received bandwidth and packet loss. From Figure 7, we see that the packet loss is more in RED, followed by CSFQ and is least in APS. Figure 8 shows that the received bandwidth is more for APS, than CSFQ and RED.

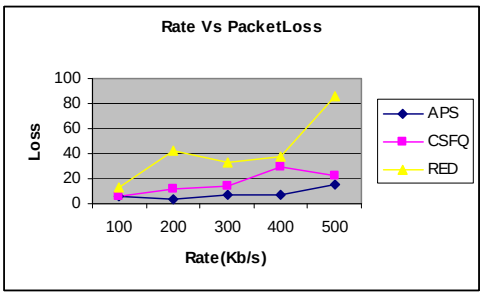


Fig. 7 Rate Vs Packet Loss

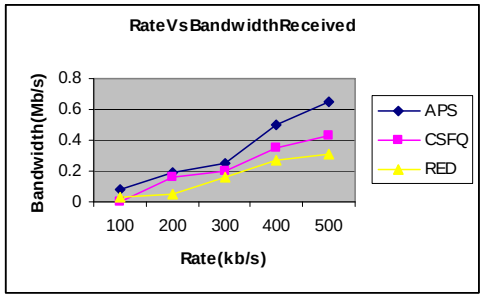


Fig. 8 Rate Vs Bandwidth

**D. Flows**

Finally, we measure the packet loss for increasing no. of flows. From Figure 9, we observe that the packet loss is more in RED, followed by CSFQ and is least in APS.

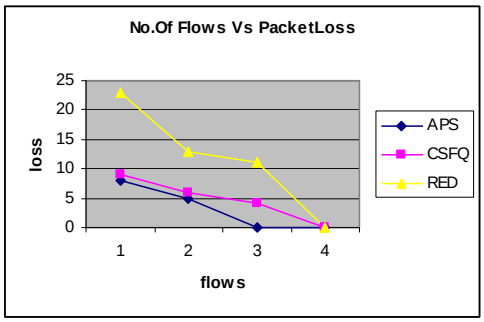


Fig. 9 Flows Vs Bandwidth

## 7. Conclusion

In this paper, we have designed an adaptive packet scheduler with a bursty traffic splitting algorithm for load balancing. The scheduler has one classifier which classifies the flows into aggressive and normal flow. A trigger handler checks for load un-balance of the network and automatically triggers the load adapter. The load adapter reroutes the high-priority aggressive flows into the least loaded best path, using the bursty traffic splitter algorithm. The bursty traffic splitting algorithm splits the aggressive flows over multiple parallel paths, based on a split vector. In this algorithm, instead of switching packets or flows, it switches packet bursts. Since the packet bursts are small in size, the algorithm split the traffic dynamically and accurately. At the same time, the condition forced on their latency difference, ensures that no packets are reordered. To achieve fair bandwidth allocations, load balancing is attained in the system since the high-rate aggressive traffic flows are splitted along multiple parallel paths. By simulations, we have shown that our adaptive packet scheduler performs better than standard fair-queuing techniques.

## References

[1] Http://En.Wikipedia.Org/Wiki

[2] Srikanth Kandula, Dina Katabi, Shantanu Sinha, Arthur Berger ”Dynamic Load Balancing Without Packet Reordering” Acm Sigcomm Computer Communication Review, Volume 37, Issue 2 (April 2007).

[3] S. Ramabhadran and J. Pasquale, "Stratified Round Robin: A Low Complexity Packet Scheduler with Bandwidth Fairness and Bounded Delay," Proc. Acm Communications Architectures and Protocols Conf. (Sigcomm), Karlsruhe, Germany, Pp. 239-249, Aug. 2003

[4] Yannick Blanpain, Hung-Yun Hsieh, Raghupathy Sivakumar “The Incremental Deployability of Core-Stateless Fair Queuing” Proceedings of the First International Conference on Networking-Part 2, Year of Publication: 2001, ISBN:3-540-42303-6.

[5] Bienkowski, Marcin; Korzeniowski, Miroslaw; Meyer Auf Der Heide, Friedhelm: “Dynamic Load Balancing In Distributed Hash Tables”, In: Proc. Of The 4th Annual International Workshop On Peer-To-Peer Systems (Iptps), 2005, S. 217-225

[6] Shi, W. Macgregor, M.H. Gburzynski, P. “Load Balancing For Parallel Forwarding” Networking, Ieee/Acm Transactions On Publication Date: Aug. 2005.

[7] K. Devine, E. Boman, R. Heaphy, B. Hendrickson, J. Teresco, J. Faik, J. Flaherty, L. Gervasio “New Challenges In Dynamic Load Balancing” Applied Numerical Mathematics, Vol. 52, Issues 2-3, Pp. 133-152, 2005.

[8] Load Balancing and Parallelism for the Internet” Nov 2007, Mendocino, CA.

[9] “Controlling Burstiness in Fair Queuing Scheduling” Cisco Systems, Bangalore, India, Oct 2007.

[10] David R. Karger And Matthias Ruhl “Simple Efficient Load Balancing Algorithms For Peer-To-Peer Systems “Acm Symposium On Parallel Algorithms And Architectures, Proceedings Of The Sixteenth Annual Acm Symposium On Parallelism In Algorithms And Architectures, 2004 .

[11] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma “Performance Analysis of Load Balancing Algorithms” Proceedings of World Academy of Science, Engineering and Technology Volume 28 April 2008 Issn 1307-6884

[12] Huei-Wen Ferng and Cheng-Ching Peng, “Traffic Splitting and Its Application to Network-Wise Performance Analysis,” In Proc. Scs International Symposium on Performance Evaluation of Computer and Telecommunication Systems (Spects) 2003, Montreal, Canada, July 2003.

[13] J. C. N. Clímaco, J. M. F. Craveirinha, M. M. B. Pascoal and L. Martins “Traffic Splitting In Mpls Networks - A Hierarchical Multicriteria Approach” Journal of Telecommunications and Information Technology, 4:3-10, 2007

[14] http://www.isi.edu/nsnam

**M.Azath** received his Master of Computer Science and Engineering degree from Anna University, on June 2007. Currently, he is doing his research in the area of Networking under Anna University, Coimbatore. Earlier he completed his B.Tech, in SSM College of Engineering from Anna University of Information Technology (IT), Chennai on April 2005. Later, he joined as Lecturer at VMU University in IT department from 2006 and still serves at the same university. His research interest includes Networking, Wireless networks, Mobile Computing and Network Security. He is a member of the Computer society of India, Salem.

**Dr.R.S.D.Wahida Banu** obtained B.E. degree in 1981 and her M.E. degree in Jan '85 from GCT, Coimbatore, Madras University. She got the Ph.D. degree in 1998 from Anna University, Chennai. First lady to acquire Ph.D. in Chennai zone and second qualified Ph.D. supervisor in the area of Computer Science and Engineering related areas. As expertise is less it continues in the Directorate of Technical Education, Tamilnadu. She is the member of ISOC, IAENG, VDAT and life member of ISTE, IE, CSI and SSI. She is currently working as Professor and Head of Electronics and Communication Engineering, Government College of Engineering, Salem.