Power Consumption Design Metric for Wireless Network Devices Using Adhoc Protocol Stack

Shivanand B Lamani¹, Patil Manikrao²
Faculty in Computer Science Department, Akkamahadevi Women's University, Vijayapura¹
Assistant Professor, Department of Information Technology, GNITC Hyderabad²
arathod74@gmail.com¹, manikpatil@gmail.com²

Abstract - Low power consumption is a key design metric for wireless network devices that have limited battery energy. The problem of reducing power consumption needs to be addressed at every level of system design. This paper investigates the issues of designing low power protocols in the context of the PEN system, a mobile ad hoc network developed at AT&T Laboratories Cambridge. It describes the ad hoc protocols that have been implemented, outlining both the design of individual protocols and the structure of the overall stack. The power-relevant mechanisms from the various protocols are collated in a summary.

Key words : Adhoc Protocol stack, SDU, PDU

I. INTRODUCTION

In a mobile embedded networking environment, the communicating nodes are small and rely on limited battery energy for their operation. Since energy is a limited resource, and battery technology has been slow to improve, the design of low power architectures and protocols has become a pressing issue. This paper investigates the issues of designing a low power protocol stack for the PEN¹ system, a mobile ad hoc radio network developed at AT&T Laboratories Cambridge [1]. An embedded network is one made up of a set of autonomous, embeddable, and possibly mobile, nodes that communicate intermittently. Nodes are radio-peers: each node is obliged neither to listen nor to transmit continuously. The maximum achievable bandwidth on an embedded network is quite low, making it more suited for control rather than data applications. To reduce power consumption, PEN nodes are powered down for much of their duty cycle and awake periodically to rendezvous with other nodes. This has interesting implications on the design of Medium Access Control (MAC) and higher-level protocols, requiring them to deal with the problem of nodes being powered down and thus unable to respond immediately.

A prime design requirement for the protocols is that they are as simple as possible in order to minimise power consumption and maintain the overall energy efficient properties of the stack. For this reason, existing de-facto protocols are not really suited for the envisaged energy-constrained environment. IP, for instance, is just too complex, particularly with respect to its routing functions; a typical implementation yields far greater code size and results in system demands that will be too excessive for a PEN node.

Although they are by no means standard, the function of many of the layers in the PEN protocol stack are similar to those in the Open Systems Interconnection (OSI) seven layer protocol stack [5]. In particular the PEN MAC layer has a similar function to the OSI Link layer – to share the physical communication medium; the PEN Routing layer has a similar function to the OSI Network layer – to route over subnetworks; and, the PEN Transport layer has a similar function to the OSI Transport layer – to provide a required quality of service. One major difference, however, is the presence of an additional layer between the MAC layer and the Routing layer called the Rendezvous layer which is responsible for scheduling and forecasting times of inactivity and thus has a major role in power saving.

![Fig. 1. The PEN protocol stack](image)

Each protocol in the stack has been implemented as an independent module and they are accessed through Application Program Interfaces (APIs) that are supersets of the one used to access the Transport layer in an end system. This means that many of the layers can be omitted from a system build should the function they provide not be required. For example the Routing layer can be omitted when the system requires interaction only with local nodes, or the Transport layer can be omitted if large Service Data Units (SDUs) and data integrity are not required. Reduced software size can have a beneficial
effect on power consumption if it results in a lower processor overhead for the periods during which a node is active.

II. PHYSICAL LAYER

In order to simplify development, the PEN hardware was designed to make use of off-the-shelf components, whilst at the same time striving to minimise power consumption and maximise functionality. A Radiometrix ‘BIM-418’ module operating at 418MHz using narrow-band frequency modulation provides the radio interface. This is capable of a raw data-rate of 40kbit/s, half-duplex, but requires a balanced data stream so a 4 bit to 6 bit encoding is used, reducing the usable data-rate to around 25kbit/s.

Although the BIM provides a carrier detect signal, it was found to be extremely unreliable, so a data-sense technique had to be used to detect channel utilisation. This relies upon an out-of-band 62 bit preamble at the start of each packet, which, upon reception, allows synchronization of the receive clock and then detection of a start of packet marker.

A Xilinx 3090 Field Programmable Gate Array (FPGA) implements the encoding which also provides an effective error detection method since only 1 in 4 6-bit codes are valid. The FPGA also handles preamble generation and detection, as well as implementing some of the MAC features discussed in the next section.

![Physical layer PDU format](image)

### A. Beacons

The basis for this design is introduced in [12] which explores the behaviour of two rendezvous systems: ‘server beaconing’ and ‘client beaconing’, as the rate of interaction increases. ‘Server beaconing’ has the server node sleeping most of the time, but periodically waking to send out a broadcast packet and advertise its presence. Following this broadcast, the node will listen briefly for replies from interested clients. Any client that wishes to use a particular server must switch on its receiver and listen for beacons from the server. ‘Client beaconing’ reverses the above strategy. The server listens continuously for broadcasts from clients wishing to use its service. Whilst this results in high power consumption for a server node, it minimises the power consumption of client nodes. As a result, this scheme offers lower overall power consumption when there are a large number of clients trying to access a relatively small number of servers.

In the PEN environment, it is anticipated that the rate of interaction will be fairly low. A scheduled rendezvous can easily be negotiated in cases where an ongoing association is required, so the basic rendezvous scheme is intended for the remaining first contact scenarios. As such, a variation on ‘server beaconing’ is the best solution. In PEN, ‘server beaconing’ is actually applied to any node that could accept Rendezvous layer datagram SDUs, whether or not it is a server. Such nodes will send out periodic beacons using the MAC multicast service, on a well-known multicast address, to all Rendezvous layer equipped nodes. Any node wishing to send Rendezvous layer datagram SDUs would listen for beacons from such nodes, and then forward its SDUs. Since reception is comparatively expensive relative to transmission, this ensures that the default state for a server node is one in which the minimum power is being expended. Nodes wishing to send SDUs must expend more power listening for beacons, and so the penalty, in terms of power, is placed upon the node originating the exchange. Note that this system takes advantage of the fact that PEN operates on a single frequency: frequency hopping systems would need to isolate a control frequency on which to broadcast availability.

Following the transmission of its beacon a node must listen for replies before powering down. This listen window must be long enough to allow nearby nodes to send their replies, but at the same time be minimised to avoid wasting power. To resolve these requirements a sliding window system is used. A minimum listen duration $T_{\text{min}}$ is specified, along with a window duration $T_{\text{window}}$.

![Rendezvous layer PDU formats](image)

The node will remain listening for at least $T_{\text{max}}$ and until an interval $T_{\text{window}}$ has passed with no further PDUs received.
B. Discovery and Description

An important part of PEN is the process of discovering nearby nodes that offer facilities with which one wishes to interact. It is of course important that this is a power efficient process. By having each node send out beacons that contain a list of the services offered by that node, the processes of discovery and description are combined. This avoids the need to expend power querying the capabilities of a node after its presence is discovered. To minimise the size of the beacon PDUs, short identifiers called Application Protocol IDentifiers (APIIDs) are used to identify different service types. When an application wishes to receive incoming datagram SDUs, it specifies the APIID that it wishes to listen on. This APIID is then appended to the beacon PDU, advertising the service to interested clients. APIIDs are similar to TCP port numbers in that they determine both the application and the protocol being communicated with.

By varying the interval between beacons it is possible to customise the power usage profile of a node. It is assumed that nodes spend most of their time idle. Whilst idle the only protocol activity is waking up periodically to send out a beacon, so if the interval between beacons is increased, so is battery life. However, if the beacon period is too long, nodes wishing to communicate must spend longer waiting to hear a beacon and so will consume more power. It is important to note that this also increases the latency of the communication. As a result there is a tradeoff between battery life and communication latency.

C. Application specified QOS

Different applications will have differing requirements for latency and battery life, so the selection of beacon interval is best left to the application. This also ensures that application developers are aware of the power decisions that they are making, leading to power efficient applications [8]. However, there is a need to reconcile differing requirements when a node contains many applications. This is fairly straightforward as the overhead in advertising multiple services in a single beacon is minimal, so only a single beacon is required. The interval for this beacon should be the minimum of those requested by the applications. Since at least one application has requested this interval, there is a justification for expending the power needed to satisfy it, and all other applications will benefit from the reduced latency it provides.

D. Transmission Modes

The beacon system discussed above offers a number of useful transmission modes when sending a datagram SDU. Firstly there is a unicast form of transmission called TX_ANY. The sender specifies a Rendezvous address consisting of the destination MAC address and destination APIID (with associated interval I), both of which must match at the receiver. The sender will then listen for a beacon from the requested destination and forward the Rendezvous SDU if the requested APIID is advertised. Under ideal conditions, this should result in the sending node listening for an average time of I/2.

A second mode of transmission offers a less directed form of unicast termed TX_ANY. The sender specifies a Rendezvous address consisting of just a destination APIID (and associated interval); it does not specify a destination MAC address. As before, the node will listen for beacons but the datagram SDU is forwarded to the first recipient found to offer that APIID. This mode will reduce the time spent listening when there are many suitable recipients and is useful for service discovery when any instance of a given service is acceptable.

The final mode offered is the least power efficient, but offers a form of broadcast that may be useful in some cases. It is called TX_ALL and attempts to forward a copy of the Rendezvous SDU to all recipients within range offering the requested APIID. To accomplish this, the node must listen for at least a full beacon interval and forward a copy of the datagram SDU to each suitable recipient detected. Upon completion a list of nodes that the SDU was sent to is returned to the application, along with the MAC return code for each attempt.

In summary, the Rendezvous layer:

- permits communication between nodes that spend much of their time asleep
- provides a mechanism for node discovery
- allows nodes to vary their power usage depending upon their application

V. ROUTING LAYER

The PEN Routing and Relay layer has been specifically designed to route data between nodes in a power-managed network. Logically this layer is above the Rendezvous layer, but in practice its implementation is closely integrated with that of the Rendezvous layer.

The Routing protocol allows nodes to communicate with neighbours more than one radio hop away without placing a significant overhead on the energy resources of the network. Most importantly, the Routing algorithm allows nodes to continue operating in a low power mode. Furthermore, the per-hop routing delay is comparable to the Rendezvous layer synchronization delay between nodes.

In addition to data delivery, the PEN Routing layer consists of two further functions: route discovery and route maintenance. Routes are discovered on demand as this makes a more efficient use of the energy in nodes since they do not have to expend energy discovering and maintaining routes they may never use. The penalty paid for this approach is a greater delivery delay when a node does not have a route to a destination as it must first perform the route discovery. Route maintenance also takes an on demand approach. A route is detected as broken because a node tried to use it and failed, and only then are other nodes notified or the route fixed.

Nodes may be mobile, but one major assumption is that they do not move very frequently or with a great speed. In a large network this is liable to be the case. Node mobility degrades the performance of any routing algorithm as
routes are broken more frequently and have to be reconfigured, but it does not add any extra complexity to the routing problem: route maintenance is still required to handle cases where the network topology does change, but more importantly - because of the unpredictable nature of radio as a communication medium - it must be able to reconfigure a route in the case where intermediate relays experience so much radio channel interference that they are unable to function.

When any transmission using the Rendezvous layer is made, a far larger amount of energy is expended in synchronizing with the receiving node than in transmitting the data PDU. This implies that the PDUs periodically used to synchronize nodes should be used to exchange as much routing information as possible. The functions of the different Rendezvous layer PDUs, pleas, offers and beacons, have been described in the previous section and the routing algorithm uses these to carry different types of routing information.

The Routing layer service achieves low power operation by accessing details of lower layers that might otherwise have been unavailable. The first is the use of Rendezvous layer protocol primitives (Beacon, Plea and Offer) to exchange routing information. The second involves the use of information gathered at the MAC layer to make the routing protocol more robust. In summary the Routing layer provides power efficient relay service.

VI. TRANSPORT LAYER

Higher up the stack, the Transport protocol provides a data integrity service that enables reliable end-to-end transmission of application-specific Transport Service Data Units (SDUs) over the inherently unreliable radio channel.

The Transport protocol is designed to operate on a homogeneous network where the endpoints are PEN or PEN-like devices, and are addressed using a combination of MAC address and Transport Port number. To reduce energy consumption the protocol adopts a two, rather than three, phase approach. It uses a two-way handshake to set up a temporary connection between the transmitter or sending end of the connection, and the recipient or receiving end. Setting up the connection involves the transmitter sending a single packet connection request to the recipient specifying the desired Transport Port. The recipient will respond with a confirmation indicating that it is ready to receive data on that port. The data transfer phase can then proceed, with the protocol performing the required segmentation and reassembly functions. In particular, the transport service data unit is segmented into a number of transport datagram PDUs, which are then transmitted block by block. The Maximum Transfer Unit (MTU) size is fixed; since heterogeneous networks are not involved, there is no need to consider refragmentation of PDUs as they go from a subnetwork with a greater MTU size to one with a smaller one, i.e. the MTU size does not need to be negotiated. At the receiving end, transport PDUs are reassembled into a transport SDU that is then passed to the receiving application.

Since the radio channel is prone to errors, and PDUs can be lost or duplicated, it is important that the Transport protocol ensures that the SDU is received intact at the recipient end. To achieve this, each transport PDU is assigned a unique sequence number; the sequence number is incremented after the transmission of the PDU. At the destination, the sequence numbers are used to identify and suppress duplicates. The recipient also maintains a bitmap of the sequence numbers of PDUs it has received so far; it periodically sends the transmitter an acknowledgement containing the sequence number of the first PDU being acknowledged along with a delivery bitmap indicating the receipt status of the next $N$ PDUs where $N$ is the size of the bitmap. When the transmitter receives an acknowledgement, it examines the bitmap, discards any undiscarded PDU that was successfully received, and retransmits any unsuccessful ones. The Transport protocol ensures that an SDU is delivered only once to its destination by associating a unique connection identifier with the temporary connection created when an SDU is first transmitted; when a recipient receives a subsequent SDU with the same connection identifier, it assumes that it is a duplicate and discards it.

The acknowledgement need not be sent in response to every PDU, and can be relatively infrequent (helping to reduce power use), but a transmitter is not allowed to have more than a certain number (window size) of outstanding PDUs at a time. At the start of the data transfer, the allowed window size is initialised to a fixed value. Each time the transmitter transmits a PDU, the window size is decremented by one. Once the window size becomes zero, the transmitter is not allowed to transmit any more Transport PDUs until it receives the necessary information from the recipient that it can increase its window again. Such a mechanism provides a basic form of flow control and liveness control indication on a per connection basis.

A transmitter with a pending data transaction has a single outstanding timer associated with it. If the transmitter has just started a transmission and a request for connection has been sent to the intended recipient, the timer is set to the expected response time of the recipient. In case of a timeout, the request is retransmitted and a new
timer is set; there are a fixed number of retransmission attempts before the connection request is aborted. Once the connection is set up and data transfer is in progress, a timer is set for every block of PDUs that is sent out.

Similarly a recipient with a pending data transfer has one outstanding timer. The timer is set as soon as the recipient has sent back a response to a connection request from the transmitter. Subsequently, the timer is reset each time a new Transport PDU is received. If the timer expires, meaning that a PDU has not been received for some time, the recipient sends back an ACK bitmap to the transmitter indicating that it is still expecting more PDUs. The ACK bitmap specifies the last block of PDUs the recipient has received, so the transmitter knows which PDUs need to be sent again.

As soon as a recipient receives the last PDU for a connection, it sends back an ACK to the transmitter and closes its end of the connection. On receiving that last ACK, the transmitter terminates its side of the connection. There is no explicit ‘closing’ of a Transport connection, thus avoiding the use of extra protocol primitives.

In summary the Transport layer:
- provides a connectionless service that has, in effect, no maximum SDU size
- provides guaranteed exactly once delivery
- uses lazy acknowledgements and windowing to reduce the power used by ACKs
- avoids explicit three-phase connection operation in order to eliminate extra protocol primitives

**VII. RELATED WORK**

The design of low power architectures and protocols has been the subject of a lot of recent research work. Stemm et al. [10] have observed that a significant fraction of a wireless node’s power is consumed by its wireless interface cards, and have investigated methods of reducing the power consumption of network interfaces, with emphasis on hand-held devices.

Various energy efficient protocols have been proposed at the MAC level. In [4], an energy efficient method for broadcasting data through indexing is proposed where a base station computes a transmission schedule and broadcasts it to mobiles. The mobiles are then able to stay in standby mode and conserve energy until it is their time to transmit. Various reservation and scheduling algorithms have also been proposed for wireless ATM networks. Scheduled access is generally good for low power consumption since the number of collisions are reduced and key components of the wireless interface hardware can be powered down based on the knowledge of the location of surrounding nodes and their transmission schedules. In their comparison of various MAC protocols [9], Sivalingam et al. conclude that contention usually results in high energy consumption while reservation and scheduling methods perform better in a low-power environment.

Zorzi [13] proposes transmission channel probing to overcome the problem of unnecessary MAC retransmissions when the radio channel quality is impaired. The idea is to send a short low-power probe packet to check the channel quality; if the probe is successfully sent, indicating that the channel quality is good, MAC retransmissions are scheduled. As a result, energy is not wasted for retransmission when the channel is impaired although energy is used to transmit the probes. At the software level, data compression techniques are considered useful for conserving power as they reduce the required transmission time. This is however at the expense of the additional CPU cycles for performing the compression which on smaller, embedded systems can itself make for significant power drain.

An application-level control approach is proposed in [6]. In this solution, the mobile host decides when to suspend and resume the communication device. Hence the mobile host acts as the master while the base station is the slave. Whenever the mobile host is asleep, the base station has to buffer any data sent to the mobile host. When the mobile host wakes up, it queries the base station to receive its data. Since the base station has no way of restarting Communication if it has run out of buffer space, it is up to the mobile host to understand the base station’s communication patterns so that there is no buffer overflow at the base station. Essentially the suspend/resume cycles have to be matched to the communication patterns between the mobile host and the base station.

IEEE 802.11 [3] defines wireless MAC and PHY layers intended for a range of applications, including mobile stations. The PHY layer specifies a frequency-hopping spread spectrum system at 2.4GHz. This provides a considerably higher data rate and superior resilience to interference than that offered by the existing PEN Physical layer, but is also inherently more power hungry. The MAC layer, Bluetooth [2] was initially conceived as a cable replacement architecture, with a range of 10m and sufficient bandwidth to carry a real-time voice channel. It has since evolved from that into architecture for creating personal ad hoc networks, and has had a higher power mode added giving a range of around 100m. Originally envisaged as having a unit cost of below $5 and ‘low’ power usage, products have been delayed in reaching the marketplace due to difficulties in achieving these requirements.

Other emerging wireless communication standards have also tackled the power saving issue. The HiperLAN is the European Telecommunications (ETSI) RES 10, with an operational frequency band of 5.2 GHz. Since HiperLAN offers a higher bit rate (over 20 Mb/s) compared to 2 Mb/s for IEEE 802.11, receiving data is more expensive as it requires equalisation (an equaliser being the most expensive part of the receiver in terms of power consumption). The solution is to divide the packet into a low-bit-rate part for control information and a high-bit rate part for data. The MAC evaluates the control information and, if the packet is addressed to the node, the equaliser is turned on and the packet is received at the higher data rate. This ensures that no energy is wasted on receiving packets that are not destined for a node. In HiperLAN, there is also the notion of high-power supporter nodes (p-supporter) and nodes that aim to save power (p-saver). Basically there is a contract between a p-
supporter and a p-saver. The p-saver is only active during prearranged intervals, and the p-supporter has to queue all packets destined for a p-saver and schedule their transmission during the active intervals of the p-saver.

From this survey of existing power efficient protocols, it can be seen that most existing protocols assume some kind of infrastructure network where there is a base-station or an elected node that co-ordinates the action of the other nodes and helps them to control their power consumption. In the embedded environment considered here, it cannot be assumed that such a node would be available; the PEN algorithms are therefore asynchronous in nature, each node decides by itself when to power on or off. This approach is more flexible since each node can control its own power usage independently and can operate in a truly ad hoc fashion.

CONCLUSION

In this paper an energy efficient MAC protocol was first proposed that saves power by minimizing the likelihood of collision and reducing the number of packet retransmissions. Above that, a Rendezvous protocol was designed that enables PEN nodes to remain asleep for much of their duty cycle and switch on their transceiver only when it is required. An optional Routing protocol uses routing information opportunistically gleaned during lower Independent Basic Service Set layer protocol exchanges and provides a power saving even when used in a relatively well-connected network. A Transport protocol has been built on top of these protocols and provides energy efficient transfer of arbitrarily sized SDUs on an end-to-end basis.

To summarize the experience gained in the construction of the PEN communications infrastructure the following, although sometimes obvious, were each found to have a role in reducing power consumption.

**Turn everything off:** The majority of power saving simply comes from the availability of hardware that allows the selective powering down of different system elements and the use of software that monitors the use of each hardware element so as to power down the ones that are not required.

**Use simple hardware:** There are a number of requirements that protocols can make of hardware, such as timer precision, complex modulation techniques, frequency hopping etc. that, when omitted, permit lower power solutions.

**Eliminate radio reception requirements:** It is sometimes assumed that transmission is more expensive in terms of power than reception. In the area of low power, low range radio, reception is likely to be at least as expensive as transmission (and is likely to have greater expense in hardware more advanced than that used in the PEN project). Identifying reception as an expensive activity can have a significant impact on protocol and system design.

**Schedule Rendezvous times:** Protocols that establish periods of availability enable periods of radio dumbness and radio silence to be implemented relatively low down in a protocol stack. In lightly loaded nodes such periods can allow power to be removed from the rest of the system.

**Consider re-using events from underlying protocol layers:** Although it may break the ideals of ‘information hiding’ it seems that making information, that might otherwise be discarded, available to higher protocol layers can often save power. In terms of the design of a protocol stack this may involve the explicit promotion of details, initially thought to be specific to a protocol, to aspects of their layer service.

Examples from the PEN protocol stack include: the use of the end-of-reception event (no matter for whom the reception was intended) as a ‘free’ synchronization hint; the accumulation of neighbourhood information available from redundant receptions in the MAC protocol (which is used in the Routing protocol to provide a liveness hint); and the use of the protocol primitives used to exchange beacons in the Rendezvous protocol to exchange routing information.

**Provide a relaying service:** In a radio environment with no global synchronization nodes have to expend a significant amount of energy in synchronizing with communication partners. Relaying through ubiquitous neighbours decreases the amount of time, and thus energy, necessary before communication can occur. If the overhead in operating the relaying protocol is small our measurements show that the overall power consumption in the network (as opposed to per-node power consumption) is also lower.

**Provide and use cheap low-quality options:** If it is simple to provide ‘low priority’ communication primitives more cheaply (in terms of their power use) than ‘normal’ primitives, consider providing them. Some protocol elements may require only a ‘least effort’ delivery service: so as to carry a hint to peers that can be safely ignored (by the recipients) with a small performance penalty, for example.

**Acknowledge streamed data infrequently:** When protocol data units are expected back-to-back, generating an acknowledgement for each of them uses power unnecessarily. A windowing protocol involving ‘lazy’ acknowledgements will save power.

**Avoid contention:** When competing for unique access to a shared channel, determining the level of contention for the channel and delaying transmission for as long as that level is high can save transmission power. The alternative mechanism of relying only on acknowledgements and retransmission wastes power on discarded transmissions although it may provide lower latency.

**Use channel availability hints:** The ‘hidden terminal’ problem [11] means that no radio event can be guaranteed to be shared by all those sharing the communications medium so protocols must be designed without an absolute requirement for synchronization via radio signals. However, the probability that most of its users do observe
the same events can be used in protocols to provide hints of channel availability.

**Keep protocols simple:** Protocols are commonly connectionless (single phase: data transfer) or 'soft state' (two phase: including explicit connection establishment) or connection orientated (three phase: including explicit connection release). Additional protocol elements supporting more phases, particularly those involved only in maintaining the shared knowledge that a connection is in place, represent additional power use: not only in their transmission and reception; but also, because their regular occurrence may prevent a node from powering down.

Despite the current limitations of our hardware, our empirical measurements show that these principles have enabled us to extend the battery life of PEN nodes significantly.

Further work is required to evaluate the effect that purpose built, more integrated, hardware would have on power consumption both through a higher quality radio transceiver and through more efficient components. For example, higher bandwidth or faster processors – although they might require additional power – may enable a node’s workload to be dispatched more quickly and thus enable it to power down sooner. Also the current Rendezvous layer does not take advantage of the predictable nature of other nodes’ availability when scheduling transmissions – modified protocol implementations that do are currently under investigation.

**REFERENCES**


**Authors:**

Mr. Shivanand B Lamani, is working as Assistant Professor in the Akkamahadevi Womens University, Vijaypura. He was awarded the M.Tech from V.T.U Belgaum, Karnataka India. He has published 3 international journals(IEEE & Springer publications). Having 4 years of teaching experience.

Mr. Patil Manikrao, is working as Assistant Professor in the Department of Information Technology, Guru Nanak Institute of Technical Campus. He was awarded the M.Tech from V.T.U Belgaum, Karnataka, India. He had published 2 international journals. He is a life member in Computer Society of India(CSI). He is having 7 years of experience which includes two years in software industry.