International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-5, January 2020

SDNOFS: Software Defined Networking with Openflow Switches & BCN-ECN with ALTQ for Congestion Avoidance

Vamshi Krishna Sabbi, Azad Shrivastava, Sunil J. Wagh, T. V. Prasad

Abstract: ***High-performance computing cluster in a cloud environment. High-performance computing (HPC) helps scientists and researchers to solve complex problems involving multiple computational capabilities. The main reason for using a message passing model is to promote application development, porting, and execution on the variety of parallel computers that can support the paradigm. Since congestion avoidance is critical for the efficient use of different applications, an efficient method for congestion management in software-defined networks based on Open Flow protocol has been presented. This paper proposed two methods; initially, to avoid the congestion problem used by Software Defined Networks (SDN) with open flow switches, this method was originally defined as a communication protocol in SDN environments which allows the SDN controller to interact directly with the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based), so that it could better adapt to changing business requirements.. Second, to enhance the quality of service and avoid the congestion problem used BCN-ECN with ALTQ. While comparing the existing method, the SDN open flow switches and BCN-ECN with ALTQ provides 98 % accuracy. Usage of these proposed methods will enhance the parameters structures delay time, level of congestion quality time and execution time***

Keywords: Message Passing Interface, Software Defined Networking, Backward Congestion Network, Explicit Congestion Network, ALTernate Queueing

I. INTRODUCTION

The Information and communication technology infrastructures are expanding continuously with the increase in a number of devices and applications for internet-of-things (IOT) and cyber physical systems. SDN is an advanced network technology that offers high interoperability and cost-effective means of user control and network programming in data center networks.

Manuscript received on January 02, 2020.
Revised Manuscript received on January 15, 2020.
Manuscript published on January 30, 2020.
*Corresponding Author
Vamshi Krishna Sabbi*, Department of Electronics and Communication, Godavari Institute of Engineering and Technology, Rajahmundry (Andhra Pradesh) India. E-mail: research.svk@gmail.com
Azad Shrivastava, HPC Division, Aura Emanating Teknology Pvt. Ltd. New Delhi, India. E-mail: azad.tech@gmail.com
Sunil J. Wagh, Department of Electronics and Communication, Mahatma Gandhi Mission' s College of Engineering and Technology, Noida (Uttar Pradesh) India. E-mail: sunilwagh@ yahoo.com
T. V. Prasad, Department of Computer Science Engineering, GIET Institutions, Rajahmundry (Andhra Pradesh) India.
E-mail: tvprasad2002@yahoo.com

The main difference between traditional networking and SDN-based networking is distinction in SDN-based networks of the data plane and the control plane. Software-defined networking (SDN) is deemed to be a technology that can efficiently manage the entire network and transform the complex network architecture into the simple and manageable one that will enable to prevent congestion with primary advantages of centralized method provisioning,
Lower operation cost, guaranteed content delivery [1]. Recent SDN studies show that single point of failure in typical networks is not able to meet the growing demand-based security issues of all components in the network, creating a complex structure that is difficult to manage. [2] That's right. the generic Open Flow network comprises with three components: the Open Flow controller, the Open Flow switch and the hosts. Each of the switches Hot spot traffic patterns, network business, defective area routing and frequency / voltage connection scaling (lower connection speed to save power) both can make a significant contribution to congestion.[3]. If all of these factors are known in advance, the network administrator may mitigate the consequences by effectively balancing traffic loads, but this is not usually the case. However, in situations where several nodes send more data Any adaptive re-routing can be done to a single destination that the node can handle to prevent network congestion. It becomes even worse when there are several different jobs in a parallel system as an on-demand service (e.g. cloud computing), where the resulting traffic pattern is unpredictable. In order to avoid congestion problems with the use of open flow switches, ECN-BCN achieves an improvement in the quality of services with ALTQ in the queuing level based on the priority queue.

High-performance computing (HPC) employs parallel processing to efficiently, reliably and quickly run advanced application programs. High-performance computing (HPC) incorporates parallel processing to drive advanced application programs efficiently, reliably, and quickly. Generally, HPC systems are built from many individual computers, relative in capacity for many personal computers, connected by a certain network. Each of these individual computers is often called a node [4]. HPC system ms frequently comprises several different node types that are specialized for different purposes. Head (or front-end or login) nodes to communicate are related to the HPC system. There are computation nodes in which the actual calculation is taken out. You don't always have direct access to computer nodes in general and especially— a scheduler or batch system manages access to these resources.

Message passing, and the Message Passing Interface (MPI), in short, is the dominant method for designing parallel applications on massive high-performance computing (HPC) systems [5]. Most MPI applications have historically relied on rollback recovery to recover from errors, a technique that can be accomplished without MPI library support. Nevertheless, recent studies suggest that, despite the estimated mean time between exascale HPC systems failures (MTBF) and beyond,
rollback-based checkpoint recovery could be underperformed to the extent that replication becomes a compelling option.

The scope of techniques used will be very huge, including checkpoint-prelaunch variations dependent on uncoordinated replication of rollback recovery, fault tolerance based on algorithms where mathematical properties are leveraged to escape checkpoints Another common characteristic found in most of these advanced methods for recovery of failure, as compared to traditional rollback recovery where the entire machine is used the program is expected to continue running despite processor failures, minimizing the I / O, downtime and overhead losses incurred [6]. The MPI Standard does not, however, Nevertheless, once affected, the MPI Standard does not specify the exact actions of MPI implementations, by failures. As a consequence, the implementation of advanced fault-tolerance technologies is difficult, with a burden on software development efficiency in many applied science environments, and fault-tolerant systems suffering from a lack for portability of ad hoc solutions.

ALTQ is a framework which continues to support a broad range of BSD-based QoS (Quality of Service) modules. The objective is to have a well-designed system for more study and experiments and realistic components of QoS. ALTQ started as a research forum for QoS, but developed into a regular traffic management subsystem. The three main components are the ALTQ system [7]. The code at the underside is responsible for the interfaces to the operating system to make use of QoS frameworks. Furthermore, the components of QoS provide QoS. User space management tools are responsible for interfaces with human or other management software.

Many MPI systems need a ratio of communication-to-computation-time to either the number of nodes they use. For these parallel applications, network latency and bandwidth can inhibit the desired linear performance acceleration with cluster size. Two major network topologies use HPC clusters: fat trees and 3D toriFat trees is used for different traffic patterns, while tori are often used for problems that have been restricted to communicating with just the nearest neighbor communication[8]. A recent study assessed network performance degradation with the increase in HPC cluster volume, down to 60% of the bandwidth provided by fat trees for MPI communication. The source of degradation is attributed to cases in which traffic from multiple sources congested a specific network connection, creating a" hot spot'.' Simulations suggest that the total loss is 40% of the marginal network.

AhmadAfsahi *et al…* [9] Static and dynamic approach to message matching development. The static approach works on the basis of the knowledge from the profiling stage and the dynamic approach to the message matching model. The drawback of the proposed solution decreases the search time for long-list traversals while preserving the performance for short-list traversals.

Bertilet and Tan al… [10] Introduced a new parallel model for medical image analysis. The system uses MPI to merge centralized and shared memory architectures and interposes the process of data switching (IPC).The algorithm illustrates how this architecture can be effectively converted to a medical endoscopic image analysis algorithm. Drawbacks of this work are only effective in connecting single application-level.

Eita et al… [11] Congestion-free routing scheme for international collectives in completely or partially settled single-work fat trees. The suggested technique of using coherent node-ordering, D-Mod-K routing and collective MPI permutation sequences offers congestion-free traffic for single or multiple jobs running on fully or partially populated Real-Life Fat-Trees. This result which removes any contention from the network and provides full network bandwidth improves the network performance.
Ernst Gunnar Gran *et al…* [12] ECOCC, a modern, reliable and cost-effective CC technique combining throttling injection with congested flow insulation. The weaknesses of two of the most successful CC approaches is analyzed and proposed a novel technique for CC that combines both approaches to overcome their respective shortcomings, so that the problems derived from congestion are more efficiently eliminated. The slow reaction time of injection throttling is the problem of this method.

Gustavo and Carsten et al… [13] DPI, an interface providing a set of Simple yet powerful abstractions versatile enough to take advantage of the features of modern networks appropriate for data processing. Looking at the concept behind the MPI used extensively in HPC, DPI is an interface definition rather than an implementation in order to bridge and develop different networking technologies. Problem occurs when the congestion boundary line to stable point are also helpful to scale other congestion control algorithms to ultra-high speed networks.

Puneet Sharma et al [14]…have proposed SDN is designed to address networking needs that are poorly addressed by existing networks, so the Open Flow protocol looks more like a tightly linked API. The goal of representing a wide range of network devices, hardware and software, and the Extensibility WG volunteer process make the design process very difficult to handle. One of the major drawbacks is that execution takes more time.

M Mubarak et al… [15] Discrete event simulation is becoming an increasingly important method for exploring potential supercomputers' development space, and many complex scientific applications are now relying heavily on collective MPI operations. Therefore, we have extended the ability of our large-scale torus and dragonfly network models to support collective communication operations in this project. But in real traces of application communication it is not applicable.

LCui et al [16] have proposed the basic features and new trends of big data and SDN in this article. So, they present some strong SDN features (separation of control and data planes, logically centralized control, global network view, network programming capabilities, etc.) that can support big data applications, including large data storage in cloud data centers, data distribution, joint management, technical big data architectures, and scheduling issues. Throughout controller operations, it is not scalable.

JBruck et al… [17] Described a way to perform *parallel* computation using the MPI over a reliable software package that allows us to take advantage of an underlying broadcast medium. Our performance measurements demonstrate the feasibility of such an approach and show reasonably effective bandwidth utilization for patterns of communication required by representative parallel programs.

BV Protopopov[18].Suggested improved message-passing efficiency using MPI implementations. According to the principles of MPI communications, ordered message transmission should only be applied within a communicator between a pair of client processes. This approach decreases the locking granularity within the implementation in addition to an increased degree of communication competition. Drawback is Granularity of lock occurs enable more space and time.

Kim et al [19]…Model for SDN-based network management. Network operators can use four control domains, time, data usage, authentication status, and traffic flow to implement and enforce a reactive network policy based on functional reactive programming with our high-level configuration language. For interaction between the Procera controller and the underlying network switches, we use the Open Flow protocol. Deals with the inherent delay caused by the control plane communication

F.O'Carroll et al [20] MPI using a zero-copy message primitive called Zero Copy MPI. A zero-copy message primitive demands that the message area be pinned down to the physical memory. In order to avoid such a deadlock, They implemented the following strategies, i) splitting the total pin-down memory size for sending and receiving to ensure that at least one sending and receiving is always active at the same time and ii) delayed queues to accommodate the transit operations of the postponed message. The main disadvantage of this project is that it is not ideal for wide range of applications and does not support MPI wildcards.

Chan-Haeng et al... [21] Introduced a new MVAPICH2 model that extends the ADI3 layer of the MPICH2. The overall design is demonstrated and the advantages of our new design are explained. Within the model we are proposing, most of the performance enhancements related to hardware can be integrated seamlessly into the system layer, resulting in high efficiency and scalability. The drawback of this work is that many performance improvements on this layer have limited implementation.

ShadiAttarha et al… [22] Suggested a new and effective algorithm to avoid overuse of links in the SDN environment. The proposed algorithm selects the minimum number of flows to overcome the congestion and redirects it to the best backup routes. This mechanism therefore has short running time and is quick and efficient. Problem, increased loss of packets and increased throughput.

The remaining portion of the paper continues as follows: Section III sets out our proposed approach to traffic control and quality improvement, followed by a review of the proposed solution. Subsequently, the outcome of the proposed solution is presented and discussed in section IV. At the end of the day, Section V presents the idea and further functions.

II. PROPOSED METHODOLOGY

HPC in Fat Network connector supercomputer processors concerning data dependence and lagging of Time in MPI communication hardware collective congestion that causes HPC systems to lower their QoS and increase congestion during computation SDN uses OpenFlow switches to avoid congestion in the network, which will further reduce congestion and reach a high standard of assured content delivery, lower capital costs and lower operating costs. While, reliability is our main aspect of networking, as enhancement uses Crossbreed BCN-ECN with ALTQ, which prevents data packet queuing, transmission bandwidth from being lost. Using this new methodology decreases the level of congestion and improves the level of accuracy with the improvement in quality to stop Queueing.

A summary of the HPC application framework development at a significant level is shown in Figure 1. Host A and Host B, Switches and Network Unit are controlled by the SDN Controller. The network component development system consists of two basic modules that can be modularly modified and repeated, i.e. a switch module and an adapter module for the network. Such two basic modules were designed in such a way that every specific network interconnection topology can be dynamically grouped in to a network configuration by mentioning the connections among multiple switch instantiations and network adapter module explicitly or algorithmically, network modules that define the overall topology of the HPC device model, i.e. its subcomponents, and how they are preseeded. In HPC devices, the flow to be split as tasks based on the BCN-ECN message.

To send and receive MPI messages, use a switch network adapter. MPI messages are then sent to origin network adapter's arrival portion, when they are divided into smaller network packets called flits. Flits traverse its switch modules in one or more hops until finally they reach a egress portion of the destination network adapter, where they are reconfigured into the initial MPI messages MPI application partition running in the presence of artificial random background traffic generated by such an additional level of generator-configured nodes. This is far more feasible than running an MPI application through an empty process and less complex than running multiple MPI applications to generate ambient traffic simultaneously.

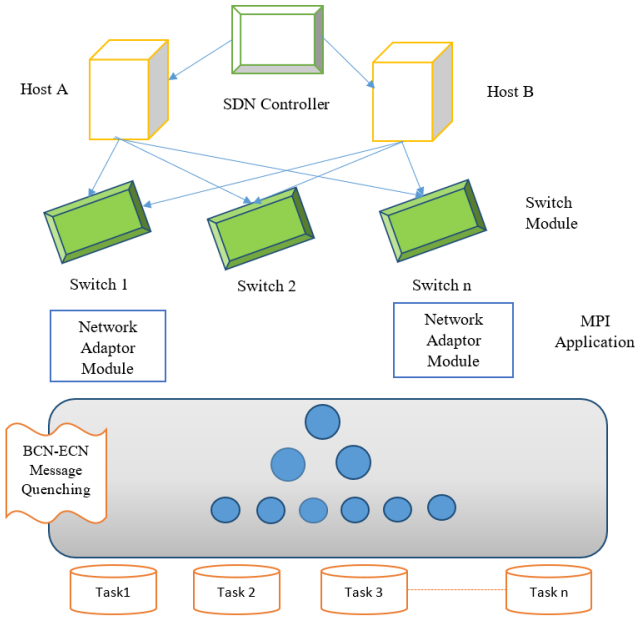

Figure 1: HPC with MPI based on SDN Controller and BCN-ECN

A. SDN controller with OpenFlow Switches

An SDN controller is a stage for SDN design that oversees flow control for improved system and device output. Software Defined Networking (SDN) is an innovation made for high-bandwidth speed and dynamic applications in the advanced world. Changing existing slowed down systems administration framework to a dynamic and sensible one is created in the historical context. The center innovation is to isolate software from hardware network device dependent on OpenFlow convention, which permits usefulness indicated by SDN support software. While the system management framework portrayed by innovation throughout this way turns out to be rather versatile and deft. Establishment of this latest worldview is the SDN operator. Typically there are two types of SDN controllers: SDN controllers for a data center's NFV framework, traditional SDN controllers to regulate the network's programmable switches.

SDN provides a widespread separation between two essential characteristics of such networking facilities, namely data transmission and decision-making. The data transmission segment of SDN is called Data Plane, and the other section is called Control Plane (Figure. 2), which decides how each data should be transmitted in a network and communicates the decision to appropriate networking facilities. As a software program, Control Plane is usually implemented, hence why it is called Software. The ability to organize a whole network globally is another beneficial aspect of SDN, which is more applicable to the study of this paper. In SDN, the Control Plane can collect information across the entire network and use the information to calculate the optimized allocation of valuable network resources.

SDN controllers direct traffic as indicated by sending strategies that a network administrator sets up, in this manner limiting manual setups for individual network gadgets. The combined centralized controller encourages automated network management by removing the control plane from the network hardware and running it as programming as shown in Figure 1 and makes it easier to incorporate and manage business applications. The SDN controller essentially fills in as a type of system operating system (OS).

Isolating the Data Plane and the Control Plane may bring multiple benefits to entities who build and manage PC networks, as shown in Figure 2 below. Most importantly, the Control Plane separation from external network management facilities, such as Ethernet switches, makes it feasible to substitute protocol handling of modules.

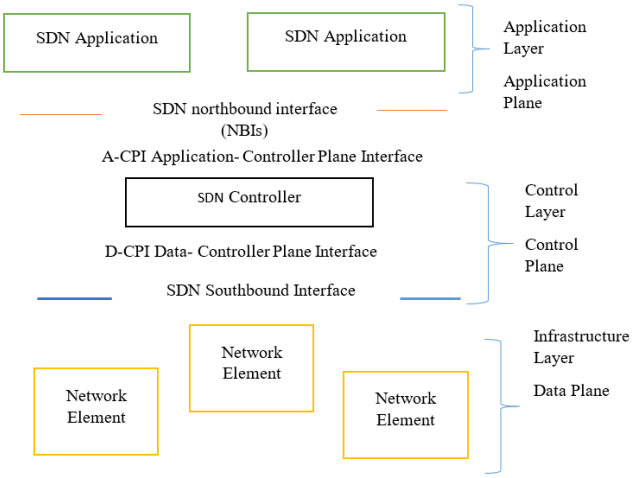

Figure 2: SDN Controllers Functionality

This should be complete rapidly by displacing the software program introduced in the Control Plane, without refreshing any firmware or setups on the real systems networking facilities. This component is very valuable for operators who need to understand their own automated network management the board appropriate for their specific organizations, or researchers who need to attempt their new systems administration conventions or traffic the executives plot.

An OpenFlow switch is an hardware or software program that transfers packages in a SDN circumstances. OpenFlow switches are either good with or dependent on the OpenFlow conventionAn OpenFlow Logical Switch consisting of at least one flow tables and a community table that fulfills packet searches and transfers, and at least one OpenFlow stream to an external controller (Figure 3). With the controller, the switch communicates and the controller directs the progress for the OpenFlow transition through the convention.

Utilizing the OpenFlow switch convention, the controller can proactively or responsively embed, alter and expel flow entries in flow tables. Each flow table throughout the switch includes a progression of flow inputs; each flow input involves flow fields, counters, and many matching packet procedures. OpenFlow's physical ports are switch-characterized ports relative to a application interface of the switch. For example, physical ports map one - to-one on an Ethernet switch to the Ethernet interfaces. In some organizations, the OpenFlow switch can be virtualized over the switch hardware.

Controller functionality as just a single point of failure, so maintaining it is necessary for any SDN. Any individual who has the controller approaches the network as a whole. This implies which organizing operators have to create security and control approaches to ensure that access is only open to the ideal people. The controller is the focal point of a code-specific system. Such residual pieces between network devices on one side of the network and on the other end of uses. The controller must be used to communicate between applications and network devices. One of the significant points of interest of SDN with OpenFlow controllers is that the centralized controller knows about all conceivable network ways and can direct traffic-based packets. It can naturally change traffic flows and help arrange network operators to remember congested connections because of the controller's perceivability into the network.

Although SDN and OpenFlow couple bolster some restricted QoS abilities it enables us to acquire per stream blockage control in a progressively adaptable, adaptable dependent on the investigation of accessible materials on 24 SDN/OpenFlow controllers.

Each instruction for flow entry requires either actions or changes the pipeline processing. The guidelines describe packet forwarding, packet modifications and team table storage. Processing instructions for pipelines allow packets to be sent to subsequent tables for further processing and allow the sharing of information in the form of metadata between tables.

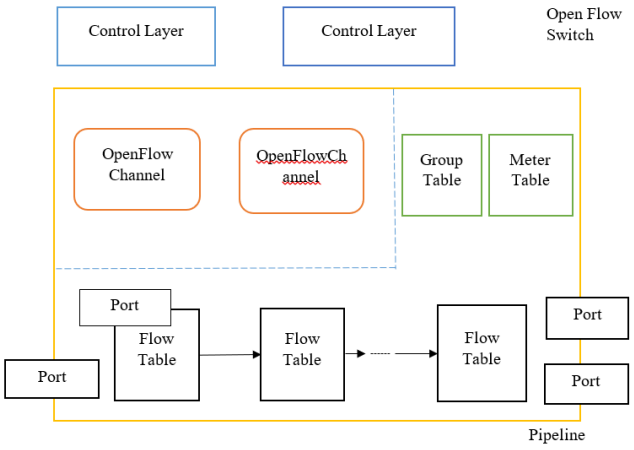

Figure 3: Open Switch Components

Table pipeline processing stops when the instruction set associated with a corresponding flow entry does not indicate a next table; at this stage, the packet is typically changed and forwarded. Flow entries can be forwarded to a channel. The OpenFlow channel is the interface that connects to an OpenFlow controller every logical OpenFlow switch. The controller configures and tracks the switch through this interface, detects events from the switch and sends out the packets for the switch. The switch's Control Channel can support a single OpenFlow channel with one controller and multiple OpenFlow channels to share with multiple controllers the management of the switch.

B. Crossbreed BECN-ECN with ALTQ

The congestion existed is clarified by SDN Openflow switches yet Crossbreed BCN-ECN with ALTQ is to presume so for this intention BCN-ECN (Backward Congestion Notification-Explicit Congestion Network) is congested with ALTernate Queuing. BCN Drag congestion from both the core of the network edges utilizing edge congestion-causing flows and configure rate-limiters relying on congestion point input, whereas ECN enables end-to-end network congestion notification without falling packets. This continues to offer a level of improvement in performance over a drop in the packet. Based on the difference between packet transfer and congestion gate manipulation, even with the packets to be dropped afterwards. It provides a level of quality control over the package's downturn. The transition is ambitious with extensive data transfers, based on differences among packet retransmission and congestion-window modification. Information on congestion with ECN can be easier because designed to detect a dropped packet requires a timeout.

ECN reduces as expected by limiting retransmission, latency and, in addition with, jitter. Unless the path has a single exceptional portion, it will have the most severe impact ALTQ comprises the queueing disciplines and other elements which are applicable to the quality of service (QoS) needed for resource allocation.

a. Backward Congestion Notification-Explicit Congestion Notification (BCN-ECN)

Another proposed congestion mechanism is the Backward Congestion Notification (BCN). This uses ICMP source quench messaging as an IP signaling mechanism to adopt a specific ECN system for IP networks, hold congestion warnings at just the IP level Negotiations on network endpoints are not included. Active notifications of congestion can be transmitted to layer protocols

BCN's central function is discussed here. As shown in Fig.4, BCN consists of:

- The congestion point (CP) makes a reference to the congestion switch. CP's role is to recognize congestion, introduce massages data, then distribute them to that of the reaction stage.
- ECN may keep changing the coding point to CP rather than dropping the text. This act is described as a "passage" to warn of unceasing congestion at a requesting exit point. All the protocols of the upper layer (transport layer protocol) at the receiving access point only provide the recognition of congestion and should be evoked back to the transmitting node to reduce its transmission rate. The ECN defines the intended bit when transmitting the message leveraging ECT (ECN Capable Transport) and Congestion Experienced (CE) as shown in table of 10' and 01' code points to show that perhaps the endpoints of the network are ECN-capable; ECT(0) and ECT(1). Use ECT(0) for protocols and senders requiring just one ECT code point. The point 00 non-ECT code indicates a packet that does not use ECN. The 11' CE code pointing to the end nodes is designed to reflect a router's congestion.

In the absence of ECN, routers with a packet that enters a complete queue sever the packet.

Table 1: ECN bits

	ECN FIELD	
ECT	CE	[Obsolete] ECN bits
0	0	Not-ECT
0	1	ECT (1)
1	0	ECT (0)
1	1	CE

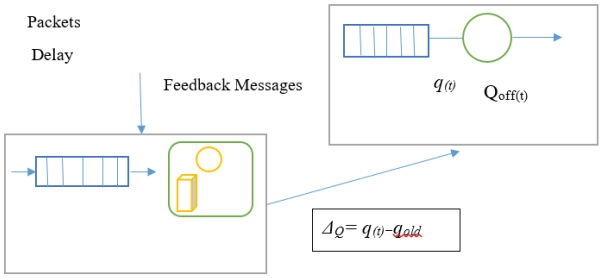

Figure 4: BCN Framework

At CP, this switch tracks incoming messages from both the $q_{(t)}$instant queue and probability p packages The congestion is assessed by F_b, which would be the current offset of the queue $(Q_{off} = q_{(t)} - q_0)$ and the deviation of the queue length in a sampling interval $(\Delta_Q = q_{(t)} - q_{old})$,where q_0 is the duration of the target queue and q_{old} is the last time the length of the queue in the sampling. F_b is provided by

$$F_b = -(Q_{off} + w * \Delta_Q) \quad (1)$$

Where w seems to be the weight, Q_{off}Queue length offset. The message of congestion that included F_b is developed and transmitted to sampled packet source, it would be likely that ECN must be applied progressively, it is essential to accommodate migration. Many routers can only ever drop the length for the queue to guarantee congestion which may not be capable of ECN-by certain end systems. The most viable strategy is one that accommodates gradual deployment without having to comply with either ECN-capable or ECN-capable. Modern traffic management and demand reduction systems need to coexist and work with existing mechanisms for congestion control

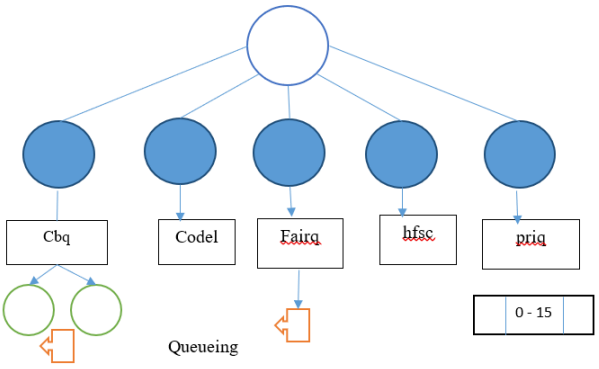

Figure 5: Queueing Discipline Levels

Queueing Discipline Levels is shown in above figure 5. Via packet scheduling and queue buffer management, a queueing discipline regulates outgoing traffic. ALTQ embraces other areas of queueing, especially for research purposes in particular. Enabling the composition of ALTQ traffic adds an unfair burden on both the infrastructure and an overall potential network performance loss will occur. In order to prevent these issues uses Queueing Discipline levels. Cbq- Class Based Queueing Queues added to an interface create a stack, allowing for additional child queues to be connected to each other. A precedent and a bandwidth can be provided for each queue. **Codel** — Delay tested CoDel does not have clear controls or choices for configuration. When enabled for a queue, it will try to manage traffic automatically.

Fair Queuing — Equal in FAIRQ, queues are monitored from highest to lowest priority, but the scheduler strives to distribute bandwidth across all connections equally. If there is no fight about bandwidth, FAIRQ will send all waiting packets. **Hfsc** — Hierarchical Fair Service Curve. Queues tethered to an interface create a tree, allowing each queue to have additional queues for kids**.** It seems that every queue can be issued a priority and a bandwidth. Priority mainly governs the time it would take for packets being sent out, whereas bandwidth mainly affects the efficiency**. Priq** — Priority queueing. Assigns numerous queues to such a network interface with priority level assigned to each queue. A queue with a higher priority is often treated rather than a queue with a lower priority. A specific number is given to the queue, ranging from 0 to 15. Packets are sorted with the highest priority in the queue first. The system also segregates the flow to misbehavior and further prevents other traffic inorder to Prevent Congestion, By using the above Levels such as Cbq, Codel, Fairq, Hfsc, Priq the notification levels Dropping a packet and allocating bandwidth is one of the most important goals of a queueing discipline.

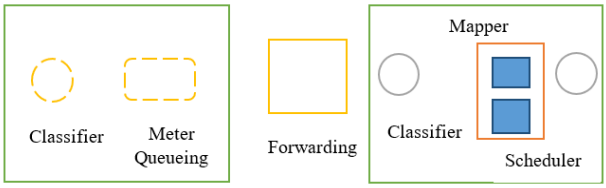

Figure 6: Queueing Architecture

The queue delay (waiting time in the queue) by magnitude orders is greater than the forwarding delay. The delay and jitter of a flow can be reduced by exchanging the appropriate network resources as shown in Figure 6. Control of admission is deliberate to determine whether it would be necessary to allocate all the required assets. It is also important to attempt to control by shaping the reserved flow rate. The incoming frequency must be significantly lower than that of the assigned speed in order to avoid delay caused by the current's own traffic. A buffer-sized leaky bucket is a simple, finite mechanism for shaping.

ALTQ performs Queueing Discipline, which can often be matched across different queueing disciplines to facilitate a series of queueing techniques with different components: flow identification, packet handling, and device driver support. Scientists will therefore be willing to implement a new queueing technique without the details on the implementation of the kernel being known. The architecture was also designed to support research as well as operation. While such a device can be remotely manipulated, it is easy to select real machines and networks through simulation. The overall design of ALTQ for conducting and testing research output is very easy to define; the queueing interface often becomes a change to a number of queueing disciplines shown in Figure 7. Alternate queueing is being seen in a network interface output queue. The queue input becomes less critical as it only operates at the entrance at some type of congestion point. IF ENQUEUE) (and IF DEQUEUE) (macros control the queue structure when snd. Such macros were used by two functions specified in structifnet, once output but when beginning; output described for each link type performs the enqueue operation; and when starting specified as part of the network process driver perform the dequeue operation;

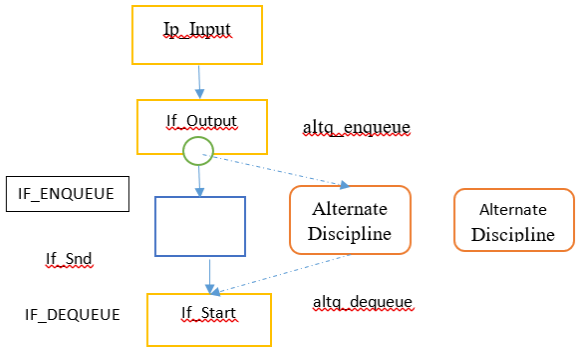

Figure 7: ALTernate Queueing Architecture

It could be assumed that it may be sufficient to consider replacing IF ENQUEUE (and IF DEQUEUE), (but sadly this is not the case. The difficulty is that both are the queuing up operations used inside the kernel being questioned and dequeuted, but the enqueue operation in a standard if the output illustrates the problem.

```
S = splimp();
if (IF QFULL(&ifp->if snd))
 {
IF DROP(&ifp->if snd);
splx(s);
m_freem(m);
return(ENOBUFS);
}
if (ifp->if snd, m);
if (ifp->if flags) = = 0) (* ifp->if start)(ifp);
splx(s);
```

Pseudocode: Enqueue Operation if_output

The application completes three operations of queueing. Check if IF (QFULL) (fills a queue then, if the latter, lowers the forthcoming packet Ask the IF ENQUEUE) as shown in the pseudocode above. Call the device operator to transmit the packet if the operator is not already occupied. The program follows tail-drop rule, i.e. dropping the packet that arrives. Yet deciding to fall and choose a victim package should be up to a queueing technique. However, the drop cycle often happens in a random drop rule after inquiring into an arriving packet. The order of the two operations, even relying on a queueing method.

```
s = splimp();
#ifdef ALTQ
   if
(ALTQ_IS_ON(ifp)) {
      error = (*ifp-
>if_altqenqueue)(ifp, m,
&pr_hdr,
ALTEQ_NORMAL);
      if (error) {
splx(s);
         return (error);
      }
   }
   else {
#endif
      if
(IF_QFULL(&ifp-
>if_snd)) {
         IF_DROP(&ifp-
>if_snd);
splx(s);
m_freem(m);

return(ENOBUFS);
      }

IF_ENQUEUE(&ifp-
>if_snd, m);
      if ((ifp->if_flags&
IFF_OACTIVE) == 0)
         (*ifp-
>if_start)(ifp);
#ifdef ALTQ
   }
#endif
splx(s);
```

```
#ifdef ALTQ
   if (ALTQ_IS_ON(ifp))
      m = (*ifp-
>if_altqdequeue)(ifp,

ALTDQ_DEQUEUE);
   else
#endif
      IF_DEQUEUE(&ifp-
>if_snd, m);
```

Pseudocode: Enqueue and dequeue of if_output and if_start

Two changes to whether output routines are being made. One is to pass information on the protocol header to an enqueue system. The protocol header data consists of a packet's address family and a pointer to the packet's network layer header. Packet classifiers can use this documentation to effectively extract the required fields from a packet header. Packet labeling can indeed be added. If the flow information are obtained in the network layer, it is always necessary to change the output interface to pass information to the output or to introduce new data to the mbuf structure that affects a fairly large part of the kernel code.

On the other hand, if the sequencing of the flow information is carried out entirely in enqueue operations, the location of the header of the network layer requires different link-level headers. Since need to change if output routines to benefit the enqueue operation of ALTQ, the choice is to save necessary information in if output is prepended before the reference header and pass the protocol header information to the enqueue method. Queueing disciplines are monitored by ioctl system calls by a queueing device (e.g./dev / cbq). ALTQ is represented as the character system between each queueing discipline and is established as a minor ALTQ device. A privileged user program opens the discipline-related queue module to trigger an alternate queueing discipline, instead connects the discipline to such an interface and activates it through the correct system calls. BCN-ECN

III. EXPERIMENTAL EVALUATION

In this section the performance Evaluation of SDN OpenFlow switches and BCN-ECN with ALTQ and Experimental Evaluation was carried out by Mininet. The measurement is carried out by two steps first, In view of the straightforward communing and customizing the vast majority of parts. SDN with OpenFlow switches specifies the flow of control based on the dynamical flow of OpenFlow switches that enable higher congestion avoidance level , Second In order to avoid traffic within the system and enhance the Quality uses Crossbreed BCN-ECN with ALTQ, this emulator can be applied in different advancement, training, and research ventures in the wake of making the system from OpenFlow switches, and have to control these switches, Mininet develops multiple networks on a single machine, including base, switch, user codes etc. Our proposed system's overall performance computation manages to achieve better performance time, congestion avoidance, Service quality. The proposed scheme achieves 97.02 % better performance time.

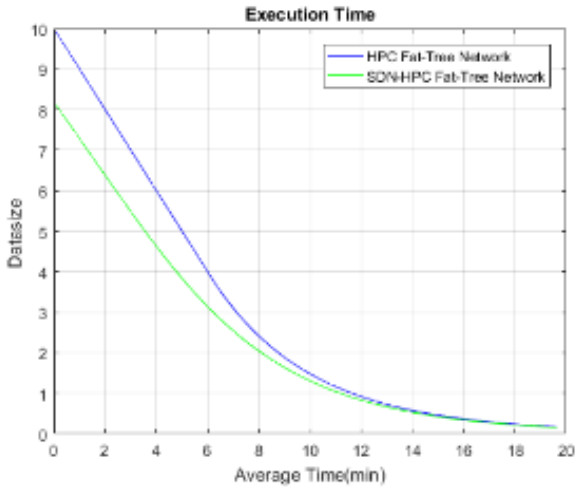

Figure 8: Execution time of MPI program on Fat-Tree network (Traditional vs SDN

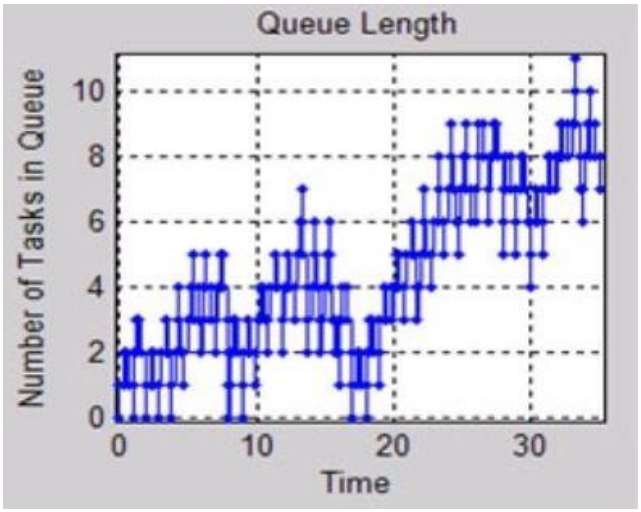

Figure 9: Queue profile of ALTQ

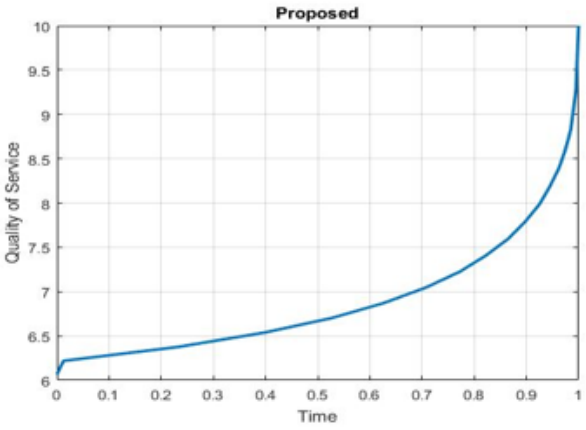

Figure 10: Quality Comparison of Proposed

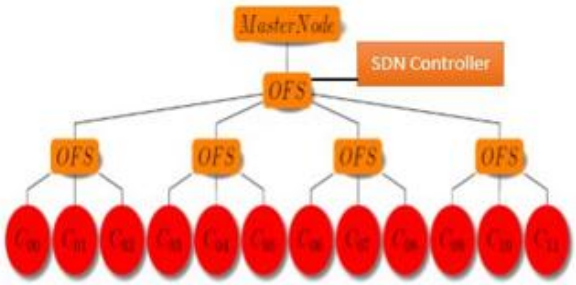

Figure 11: Compute Node and OpenFlow Switches in SDN

The above figure 8 shows the average execution time accomplished by the Proposed based on the sizes used to specify MPI and SDN OpenFlow switches with various sizes of the dataset. MPI Communication problem triggers network delay, on the basis of which runs SDN with OpenFlow switches to control network time lag, Figure 9 above shows the Queue length to be assigned in the delay tolerance specification. ALTQ achieves greater task to be assumed while congestion occurs with greater performance of 97.45%.

Quality in network is lowered while using QCN and FCN but by Crossbreeding BCN-ECN quality level is increased within the time. Proposed Scheme as shown in below Figure 10 ensures that the BCN-ECN achieves higher Quality level 98.87% Figure 11 below shows the implementation of Fat Network Switches in The number of ports included in each switch Switches with 12 number of computing nodes are called k-port Fat network topology in a Fat tree topology.
OpenFlow switches each switch, contain same number of ports in it so the notification level of the congestion decreases.

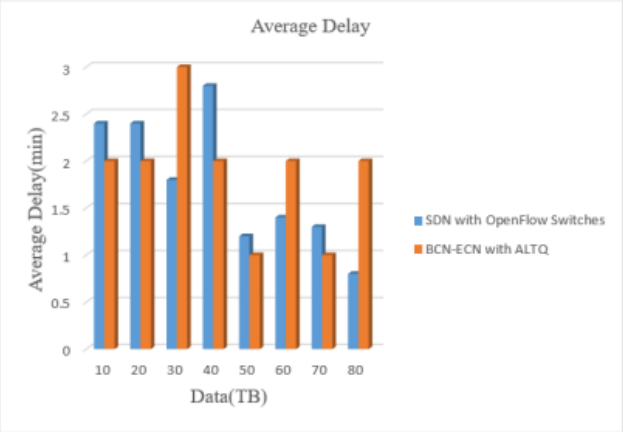

Figure 12: Average Delay of SDN with OpenFlow switches and BCN-ECN with ALTQ

The above figure 12 specifying the average delay of Proposed System, accurately declare that the Delay time in OpenFlow-switch tried to give the controller a congestion message and both the Methodology achieves better delay Lowrance level of 98.32%.

Table 1: Evaluation Results in Tabular Form

Delay Tolerance level of Proposed SDN with Open Flow Switches and BCN-ECN with ALTQ		
Throughput(KB)	Quality Time	Ratio of Congestion
72.50	3.40	23%
73.10	3.20	20%

Above table 1 specifies the evaluation of Parameters such as Throughput, Quality Time and Ratio of Congestion to be lay down in Values as per the satisfaction of Proposed work.
SDN uses the plane of flow with OpenFlow switches so the level of congestion occurring degrades than other forms and the Execution time increases and delay time decreases based on the controllers in SDN, After the congestion avoidance the overflow of traffic and the queuing levels is controlled based on BCN-ECN with ALTQ Prevention of congestion takes place.

IV. CONCLUSION

In this paper, Initially resolving the problem of MPI using SDN controller with Openflow Switches, here SDN Controller enables the layer level flow of control while passing a message and Openflow Switches provides the Message by labeling in Table and functioning the flow of control inorder to avoid congestion the implementation results shows that the SDN Controller with Openflow Switches achieves congestion avoidance rate of 96.76%. Second, after the avoidance of congestion Quality is boosted and occurring of congestion is avoided by BCN-ECN with ALTQ enables the notification of control to be send as a message to the network inorder to prevent congestion and enhancing Queueing Disciplines with the help of ALTernate Queueing that enables the enqueue and dequeue within the flow of control in the network. By evaluating the effectiveness of our approach by using a series of switches, demonstrating that this method has potential for improvement, particularly in the advancement of the output and the reduction of average congestion avoidance. The Overall implementation results shows that the BCN-ECN with ALTQ achieves 97.05% level of quality is achieved.

REFERENCES

1. Rawat, Danda B., and Swetha R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey", *IEEE Communications Surveys & Tutorials*, Vol. 19, no. 1, pp. 325-346,2016
2. Gholami, Masoumeh, and Behzad Akbari, "Congestion control in software defined data center networks through flow rerouting." In *2015 23rd Iranian Conference on Electrical Engineering*, pp. 654-657. IEEE, 2015.
3. Gran, Ernst Gunnar, Sven-Arne Reinemo, Olav Lysne, Tor Skeie, Eitan Zahavi, and Gilad Shainer. "Exploring the scope of the InfiniBand congestion control mechanism." In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, pp. 1131-1143. IEEE, 2012.
4. Rizwan A Ashraf., Saurabh Hukerikar, and Christian Engelmann. " Shrink or substitute: Handling process failures in HPC systems using in-situ recovery", *Proc. of 26th EuromicroInt. Conf. on Parallel, Distributed and Network-based Processing (PDP)*, pp. 178-185.IEEE, 2018
5. Lauria, Mario, Scott Pakin, and Andrew Chien. "Efficient layering for high speed Communication: the MPI over Fast Messages (FM) experience.*" Cluster Computing*, Vol. 2, no.2 (1999): 107-116
6. Weifeng Liu, Jie Zhou, and MengGuo. "Topology-Aware Strategy for MPI-IO Operations in Clusters." *J. of Optimization*, Vol. 5552, 2018.
7. Cho, Kenjiro. "Fitting Theory into Reality in the ALTQ case." In *Proceedings of ASIA BSD conference*. 2004.
8. K Takahashi, DashdavaaKhureltulga, Yasuhiro Watashiba, Yoshiyuki Kido, Susumu Date, and Shinji Shimojo. "Performance evaluations of SDN-enhanced MPI all reduce on a cluster system with fat-tree interconnect",*Proc. of Int. Conf. on High Perf. Computing & Simulation (HPCS)*, pp. 784-792. IEEE, 2014.
9. AhmadAfsahi,S.MahdiehGhazimirsaeed,an dH. Mirsadeghi Seyed,. "Communication-aware message matching in MPI." *Concurrency and Computation: Practice and Experience: e4862*, 2018.
10. Bertil Schmidt and Tan, Sean Chiew Seng. "Combining message-passing and inter-process communication in SMP-hybrid cluster for efficient parallel medical image analysis." In *Applications of Digital Image Processing XXVII*, vol. 5558, pp. 181-190. International Society for Optics and Photonics, 2004.
11. EitanZahavi "Fat-tree routing and node ordering providing contention free traffic for MPI global collectives", *J. of Parallel and Distributed Computing*, Vol. 72, no. 11, pp. 1423-1432, 2012

12. Ernst Gunnar Gran, Jesus Escudero-Sahuquillo, Pedro Javier Garcia, Jose Flich, Tor Skeie, Olav Lysne, Francisco Jose Quiles, and Jose Duato. "Combining congested-flow isolation and injection throttling in HPC interconnection networks", *Proc. of Int. Conf. on Parallel Processing*, pp. 662-672. IEEE, 2011.
13. Fengyuan Ren, Wanchun Jiang, Xin Yue, and Chuang Lin. "Scale Congestion Control to Ultra-High Speed Ethernet", *arXiv preprint arXiv:1405.1127*, 2014
14. Sujata Banerjee, Tourrilhes Jean, Puneet Sharma, and Justin Pettit. "The evolution of SDN and OpenFlow: a standards perspective." *IEEE Computer Society*, Vol. 47, no. 11, pp. 22-29, 2014.
15. M Mubarak, Christopher D. Carothers, Robert B. Ross, and Philip Carns. "Using massively parallel simulation for MPI collective communication modelling in extreme-scale networks" *Proc. of the Winter Simulation Conf.*, pp. 3107-3118. IEEE, 2014.
16. L. Cui, Qiao Yanand F. Richard Yu. "When big data meets software-defined networking: SDN for big data and big data for SDN" *IEEE Network*, Vol. 30, no. 1, pp. 58-65, 2016
17. Bland, Wesley, AurelienBouteiller, Thomas Herault, George Bosilca, and Jack Dongarra. "Post-failure recovery of MPI communication capability: Design and rationale", *Int. J. of High-Perf. Computing Applications*, Vol. 27, no. 3, pp. 244-254,2013.
18. BV Protopopov and Anthony Skjellum. Multithreaded message passing interface (MPI) architecture: Performance and program issues", *J. of Parallel and Distributed Computing*, Vol. 61, no. 4, pp. 449-466, 2001
19. H. Kim and Nick Feamster. "Improving network management with software defined networking.", *IEEE Comm. Magazine*, Vol. 51, no. 2, pp. 114-119,2013
20. F.O'Carroll, Hiroshi Tezuka, Atsushi Hori, and Yutaka Ishikawa. "The design and implementation of zero copy MPI using commodity hardware with a high performance network", *Proc. of 12th Int.Conf. on Supercomputing, ACM*, pp. 243-250,1998.
21. Ching-Tien Ho,Ray Strong J Bruck, Danny Dolev, and Marcel-CătălinRoșu. "Efficient message passing interface (MPI) for parallel computing on clusters of workstations", *J. of Parallel and Distributed Computing*, Vol. 40, no. 1, pp. 19-34, 1997
22. ShadiAttarha, Koosha Haji Hosseiny, GhasemMirjalily, and KiarashMizanian. "A load balanced congestion aware routing mechanism for Software Defined Networks." *In 2017 Iranian Conference on Electrical Engineering (ICEE)*, pp. 2206-2210. IEEE, 2017.

AUTHORS PROFILE

Sabbi Vamshi Krishna, is an associate professor in Electronics and Communication Department at Godavari Institute of Engineering and Technology, Rajahmundry, India since 2016. He joined Doctoral Program in 2015 at Dr. A.P.J Abdul Kalam Technical University, Lucknow, India. He received Bachelor of Engineering in Electronics and Telecommunication from Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India in 2002 and Masters of Technology from C-DAC, Noida affiliated to Guru Govind Singh Indraprastha University, New Delhi, India in 2008. He has good experience of computer network deployment, CCNA, Switches, Routers and testing with Rooman Technology training center, New Delhi. He has good working experience on FPGA (Xilinx and Actel).. His areas of interest are Advanced Microachitecture of Processor, High Performance Computing, Software Defined Networking, Open Flow Standards, and Database. He has also experience as an independent technical consultant on network operations, HPC and management for small and medium enterprises. Email: research.svk@gmail.com

Dr. Azad Shrivasatva, is National Head of High Performance Computing and Open Source Development Practices at CMC Limited, New Delhi, India. He received Ph.D degree from ABV-Indian Institute of Information Technology, Gwalior, India in 2009. He received his MS Degree C.E.U, MCSI Ft. Lauderdale, Florida and Bachelor of Engineering in Electronics & Communications, Government Engineering College Jabalpur from University of Jabalpur Madhya Pradesh, India. He has vast experience of 35 years in providing HPC solution based on CPU and GPU installation, implementing of Private and public cloud and Database solutions. He is providing consultancy to NRCPB, IMD, PRL and IIT Jodhpur. He chaired national HPC conferences in 2011&2013. His passion toward education, derived him to give active participation in Engineering education. He is academic council members of Mewar University Rajasthan and Vikrant Group of Institutions. He has given training on Big-data, HPC, Luster file system, GPFS, Cloud Computing and many more. Email: azad.tech@gmail.com

Dr. Sunil J. Wagh, is Vice Principal at Mahatma Gandhi Mission's College of Engineering and Technology, Noida (NCR) India. He was born in Sakri, District Dhule, Maharashtra on May 10, 1963. He received Ph.D Degree in the area of Wireless Communication Network from Swami Ramanand Treeth Marathwada University, India in 2015. He received M.Sc (Electronics) and M.Tech (Electronics Design and Technology) in 1987 and 1997 respectively from Marathwada University, Aurangadad, India. He joined as lecturer in Electronics and Communication Department at Mahatma Gandhi Mission's College of Engineering, Nanded, India. His area of interest are Wireless Communication Network, Ad hoc sensor network , High Performance Computing, Computer Architecture and Database. Out of the academia, he has also experience as an independent technical consultant on network operations and management for small and medium enterprises.
Email: sunilwagh@yahoo.com

Dr. T. V. Prasad, is working in capacity of Director, Quality Assurance at GIET Campus, Rajahmundry India. He received his Ph. D in Computer Science Bioinformatics from Jamia Milia Islamia University, New Delhi in 2006.His areas of interest are Supercomputing, Natural Language Processing, Bioinformatics, and Human Computer Interaction etc. He had worked with Bureau of Indian Standard in capacity of Dy. Director for eight years. He has published more than 100 papers in reputed international journals, national and international conferences. He authored three books with reputed publishers. He has filled 7 patents till date and one is granted to him. Four Ph. D degrees are awarded under his guidance and three scholars are doing their Ph. D under his guidance.
Email Id: tvprasad2002@yahoo.com