Middle-East Journal of Scientific Research 20 (1): 100-107, 2014
ISSN 1990-9233
© IDOSI Publications, 2014
DOI: 10.5829/idosi.mejsr.2014.20.01.11284

# An Innovative of Low-Priority Service via End-Point Congestion Control for TCP-LP

*E. Fathima and K.P. Kaliyamoorthie*

Department of Computer Science, Bharath University, India

**Abstract:** Service prioritization among different traffic classes is an important goal for the Internet. Conventional approaches to solving this problem consider the existing best-effort class as the low-priority class and attempt to develop mechanisms that provide "better-than-best-effort" service. In this project, we explore the opposite approach and devise a new distributed algorithm to realize a low-priority service (as compared to the existing best effort) from the network endpoints. To this end, we develop TCP Low Priority (TCP-LP), a distributed algorithm whose goal is to utilize only the excess network bandwidth as compared to the "fair share" of bandwidth as targeted by TCP. The key mechanisms unique to TCP-LP congestion control are the use of one-way packet delays for early congestion indications and a TCP-transparent congestion avoidance policy. The results of our simulation and Internet experiments show that that: TCP-LP is largely non-intrusive to TCP traffic; both single and aggregate TCPLP flows are able to successfully utilize excess network bandwidth; moreover, multiple TCP-LP flows share excess bandwidth fairly; substantial amounts of excess bandwidth are available to the low-priority class, even in the presence of "greedy" TCP flows; The response times of web connections in the best-effort class decrease by up to 90% when long-lived bulk data transfers use TCP-LP rather than TCP; despite their low-priority nature, TCP-LP flows are able to utilize significant amounts of available bandwidth in a wide-area network environment.

**Key words:** TCP-LP congestion · Substantial amounts · To develop mechanisms · TCP flows

## INTRODUCTION

**Problem Definition:** Motivated by the diversity of networked applications, a significant effort has been made to provide differentiation mechanisms in the Internet. However, despite the availability of simple and scalable solutions, deployment has not been forthcoming. A key reason is the heterogeneity of the Internet itself: with vastly different link capacities, congestion levels, etc., a single mechanism is unlikely to be uniformly applicable to all network elements. We devise TCP-LP (Low Priority), an end-point protocol that achieves two-class service prioritization without any support from the network.

The key observation is that end-to-end differentiation can be achieved by having different end-host applications employ different congestion control algorithms as dictated by their performance objectives. Since TCP is the dominant protocol for best-effort traffic, we design TCP-LP to realize a low-priority service as compared to the existing best effort service. Namely, the objective is for TCP-LP flows to utilize the bandwidth left unused by TCP flows in a non-intrusive, or TCP-transparent, fashion. Moreover, TCP-LP is a distributed algorithm that is realized as a sender-side modification of the TCP protocol.

One class of applications of TCP-LP is low-priority file transfer over the Internet. For network clients on low-speed access links, TCP-LP provides a mechanism to retain faster response times for interactive applications using TCP, while simultaneously making progress on background file transfers using TCPLP.

Similarly, in enterprise networks, TCP-LP enables large file backups to proceed without impeding interactive applications, a functionality that would otherwise require a multi-priority or separate network. Finally, institutions often rate-limit certain applications (e.g., peer-to-peer file sharing applications) such that they do not degrade the performance of other applications. In contrast, TCP-LP allows low priority applications to use all excess capacity while also remaining transparent to TCP flows.

**Corresponding Author:** E. Fathima, Department of Computer Science, Bharath University, India.

A second class of applications of TCP-LP is inference of available bandwidth for network monitoring, end-point admission control and performance optimization (e.g., to select a mirror server with the highest available bandwidth). Current techniques e.g., [1], estimate available bandwidth by making statistical inferences on measurements of the delay or loss characteristics of a sequence of transmitted probe packets.

In contrast, TCP-LP is algorithmic with the goal of transmitting at the rate of the available bandwidth. Consequently, competing TCP-LP flows obtain their fair share of the available bandwidth, as opposed to probing flows which infer the total available bandwidth, overestimating the fraction actually available individually when many flows are simultaneously probing. Moreover, as the available bandwidth changes over time, TCP-LP provides a mechanism to continuously adapt to changing network conditions [2-5].

**Project Description:** Our methodology for developing TCP-LP is as follows. First, we develop a reference model to formalize the two design objectives: TCP-LP transparency to TCP and (TCP-like) fairness among multiple TCP-LP flows competing to share the excess bandwidth. The reference model consists of a two level hierarchical scheduler in which the first level provides TCP packets with strict priority over TCP-LP packets and the second level provides fairness among micro flows within each class. TCP-LP aims to achieve this behavior in networks with no differentiated (first-come-first-serve) service. Next, to approximate the reference model from a distributed end-point protocol, TCP-LP employs two new mechanisms [6].

First, in order to provide TCP-transparent low-priority service, TCP-LP flows must detect oncoming congestion prior to TCP flows. Consequently, TCP-LP uses inferences of one-way packet delays as early indications of network congestion rather than packet losses as used by TCP. We develop a simple analytical model to show that due to the non-linear relationship between throughput and round-trip time, TCP-LP can maintain TCP-transparency even if TCP-LP flows have larger round-trip times than TCP flows. Moreover, a desirable consequence of early congestion inferences via one-way delay measurements is that they detect congestion only on the forward path (from the source to the destination) and prevent false early congestion indications from reverse cross-traffic [7]. TCP-LP's second mechanism is a novel congestion avoidance policy with three objectives:

- Quickly back off in the presence of congestion from TCP flows
- Quickly utilize the available excess bandwidth in the absence of sufficient TCP traffic and
- Achieve fairness among TCP-LP flows. To achieve these objectives, TCP-LP's congestion avoidance policy modifies the additive-increase multiplicative-decrease policy of TCP via the addition of an inference phase and use of a modified back-off policy.

Furthermore, we perform an extensive set of ns-2 simulation experiments and study TCP-LP's characteristics in a variety of scenarios (single and multiple bottlenecks, short- and long-lived TCP flows, etc.). First, in our experiments with greedy TCP flows (FTP downloads), we show that TCP-LP is largely no intrusive to TCP traffic and that TCP flows achieve approximately the same throughput whether or not TCP-LP flows are present. Second, we explore TCP-LP's dynamic behavior using experiments with artificial "square-wave" background traffic. We show that single and aggregate TCP-LP flows can successfully track and utilize the excess network bandwidth.

Finally, in our experiments with HTTP background traffic, we show that flows in the best-effort class can benefit significantly from the two-class service prioritization scheme. For example, the response times of web connections in the best-effort class decrease by up to 90% when long-lived bulk data transfers use TCP-LP rather than TCP.

Finally, we implement TCP-LP in Linux and evaluate it both in a tested as well as on the Internet. In the tested, we perform experiments with many TCP and TCP-LP flows and show that TCP-LP remains its TCP-transparent property even in such large-aggregation regimes.

Likewise, our Internet experiments show that TCP-LP remains non-intrusive in a wide-area network environment, while being able to utilize substantial amounts of the available spare network bandwidth. For example, when compared to TCP, TCP-LP is able to utilize approximately 45% of the TCP throughput on average during working-hours (8 a.m. to 5 p.m.) and as much as 75% outside this interval. Thus, the results from both our simulation and Internet experiments confirm that TCP-LP is a practically applicable protocol that accurately approximates the functionality of the reference model. The reminder of this paper is organized as follows.

## System Design

**TCP-LP:** TCP-LP is a distributed algorithm that is realized as a sender-side modification of the TCP protocol. One class of applications of TCP-LP is low-priority file transfer over the Internet. For network clients on low-speed access links, TCP-LP provides a mechanism to retain faster response times for interactive applications using TCP, while simultaneously making progress on background file transfers using TCP-LP. Similarly, in enterprise networks, TCP-LP enables large file backups to proceed without impeding interactive applications, a functionality that would otherwise require a multi-priority or separate network. In contrast, TCP-LP allows low priority applications to use all excess capacity while also remaining transparent to TCP flows [8].

A second class of applications of TCP-LP is inference of available bandwidth for network monitoring, end-point admission control and performance optimization (e.g., to select a mirror server with the highest available bandwidth). Current techniques estimate available bandwidth by making statistical inferences on measurements of the delay or loss characteristics of a sequence of transmitted probe packets. In contrast, TCP-LP is algorithmic with the goal of transmitting at the rate of the available bandwidth. Consequently, competing TCP-LP flows obtain their fair share of the available bandwidth, as opposed to probing flows which infer the total available bandwidth, overestimating the fraction actually available individually when many flows are simultaneously probing. Moreover, as the available bandwidth changes over time, TCP-LP provides a mechanism to continuously adapt to changing network conditions [9].

**Reference Model and Design Objectives:** LP ows should each obtain a fair bandwidth share compared to left unutilized by non TCP-LP ows thereby making TCP-LP ows transparent to TCP and UDP ows. This design objective is formalized in Figure 2(a) which depicts a two-class hierarchical scheduling model ([10]) that achieves the idealized system functionality. In the reference system, there is a high-priority and low-priority class, with the former obtaining strict priority service over the latter. Within each class, service is fair among competing ows-controlled ows. As networks do not typically employ such scheduling mechanisms, the objective of TCP-LP is to obtain an approximation to the reference model's behavior via an end-point congestion control algorithm. As depicted in Figure 2(b), in the actual

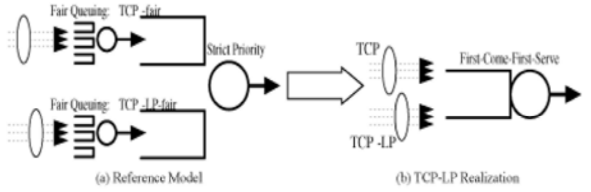


Fig. 2: Reference Model and TCP-LP Realization

system, all ows (high and low priority ) are multiplexed into a single first-come-first-serve queue and service approximating that of the reference model is obtained via the use of two different congestion control protocols, TCP an d TCP-LP. In other words, TCP ows should obtain strict priority service over TCP-LP ows and competing TCP-LP ows should each obtain a fair bandwidth share compared to.

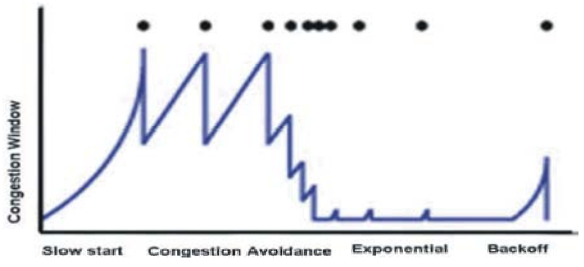
**TCP:** The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite. TCP provides reliable, in-order delivery of a stream of bytes, making it suitable for applications like file transfer and e-mail. It is so important in the Internet protocol suite that sometimes the entire suite is referred to as "the TCP/IP protocol suite."

TCP is used extensively by many of the Internet's most popular application protocols and resulting applications, including the World Wide Web, E-mail, File Transfer Protocol, Secure Shell and some streaming media applications [9].

However, because TCP is optimized for accurate delivery rather than timely delivery, TCP sometimes incurs long delays while waiting for out-of-order messages or retransmissions of lost messages and it is not particularly suitable for real-time applications such as Voice over IP. For such applications, protocols like the Real-time Transport Protocol (RTP) running over the User Datagram Protocol (UDP) are usually recommended instead [11].

TCP also distinguishes data for multiple connections by concurrent applications (e.g., Web server and e-mail server) running on the same host. In the Internet protocol suite, TCP is the intermediate layer between the Internet Protocol (IP) below it and an application above it. Applications often need reliable pipe-like connections to each other, whereas the Internet Protocol does not provide such streams, but rather only best effort delivery (i.e., unreliable packets). TCP does the task of the transport layer in the simplified OSI model of computer networks. The other main transport-level Internet protocols are UDP and SCTP.

**TCP Congestion Control:** Figure 1 show a temporal view of the TCP/Reno congestion window behavior at different stages with points on the top indicating packet losses.1 Data transfer begins with the slow-start phase in which TCP increases its sending rate exponentially until it encounters the first loss or maximum window size. From this pointon, TCP enters the congestion-avoidance phase and uses an additive-increase multiplicative-decrease policy to adapt to congestion. Losses are detected via either time-out from non-receipt of an acknowledgment, or by receipt of a triple-duplicate acknowledgment. If loss occurs and less than three duplicate ACKs are received, TCP reduces its congestion window to one segment and waits for a period of retransmission Time out (RTO), after which the packet is resent. In the case that another time out occurs before successfully retransmitting the packet, TCP enters the exponential-back off phase and doubles RTO until the packet is successfully acknowledged.



**Available Bandwidth:** The available bandwidth is an important performance metric for a network path and different tools are already developed to perform this kind of measurements. The available bandwidth (avail-bw) in a network path is of major importance in congestion control, streaming applications, QoS verification, server selection and overlay networks Measuring available bandwidth (Abw) is not only for knowing the network status, but also to provide information to network applications on how to control their outgoing traffic and fairly share the network bandwidth.

Some applications require very short sampling time and high sampling frequency, not only measurement accuracy. In this workshop, we propose to discuss what bandwidth estimation algorithms and sampling methods can be used with different application requirements. There are roughly 3 classes of use cases for bandwidth estimation algorithms: instantaneous, short term and long term.

**Service Prioritization:** The aim of service prioritization is to give resource priority to certain traffic types. For example, the voice service can have higher priority than the data service, Web surfing can have higher priority than FTP and e-mail applications, or some users can have higher priority than others. Prioritization alone won't eliminate all of bandwidth blues. If congestion is a real problem for the network, we need to address it with bigger network pipes, not just prioritization services. If we are losing packets now, then we will still lose packets after implementing prioritization services. The difference is that prioritization allows us to decide which packets we are willing to lose [10]. If not willing to throw away any of them, we must add bandwidth. However, if the problem is not sustained congestion but sporadic bursts that do not yet justify the cost of additional bandwidth, our focus should definitely be on using prioritization to ensure smooth operations.

**TCP-Transparency:** An achieve TCP-transparent behavior in large aggregation regimes, TCP-LP reduces its window size to one packet per RTT in the presence of TCP flows and decreases packet size to 64Bytes. Here, we consider scenarios with many flows to evaluate whether TCP-LP remains non-intrusive in such regimes. We perform experiments with simultaneous TCP and TCPLP file transfers where the number of flows in the system (both TCP and TCP-LP) increases from one to 20.

LP flow utilizes only excess bandwidth

- Does not reduce the throughput of TCP flows

In presence of TCP cross-traffic:

- TCP achieves fairness
- LP achieves TCP-transparency

**Modules Description**
**Modules:**

- Topology creation and transmission
- Tcp-lp implementation with congestion
- Tcp-lp implementation without congestion

**Topology Creation and Transmission:** In this module, we first describe the implementation details of TCP-LP in Linux and explain the specific use of the TCP window-scaling and time stamping options from. Next, we examine the protocol performance on a tesbed and evaluate three important TCP-LP features. The first is non-intrusiveness to TCP traffic in large aggregation regimes; the second is the ability of both a single TCP-LP flow and aggregates to utilize.

The available bandwidth in the presence of highly dynamic cross traffic; the third is fairness among TCP-LP flows.

To the best of our knowledge, TCP-LP is the first TCP stack that uses time stamping option for computing one-way delays. This option is originally developed to alleviate computation of round-trip times. We emphasize the fact that the use of one-way delays is an essential requirement for low-priority transport protocols in the Internet. Next, we evaluate TCP-LP's ability to utilize the available bandwidth in the presence of extremely dynamic cross-traffic. To this end, we perform an experiment similar to the one from Section V-B with square-wave UDP cross-traffic oscillating between 0 and 2/3 of the link capacity and where the period of oscillation changes from 50ms up to 51.2 sec [11].

**TCP-LP Implementation with Congestion:** To achieve low priority service in the presence of TCP traffic, it is necessary for TCP-LP to infer congestion earlier than TCP. In principle, the network could provide such early congestion indicators. For example, TCP-LP flows could use a type-of service bit to indicate low priority and routers could use Early Congestion Notification (ECN) messages to inform TCPLP flows of lesser congestion levels than TCP flows.

However, given the absence of such network support, we devise an endpoint realization of this functionality by using packet delays as early indicators for TCP-LP, as compared to packet drops used by TCP. In this way, TCP-LP and TCP implicitly coordinate in a distributed manner to provide the desired priority levels. TCP-LP measures one-way packet delays and employs a simple delay threshold-based method for early inference of congestion. Denote di as the one-way delay of the packet with sequence number i and dmin and dmax as the minimum and maximum one-way packet delays experienced throughout the connection's lifetime. Thus, dmin is an estimate of the one way propagation delay and dmax - dmin is an estimate of the maximum queuing delay. TCP-LP obtains samples of one-way packet delays using the TCP timestamp option. Each TCP packet carries two four byte timestamp fields. A TCP-LP sender timestamps one of these fields with its current clock value when it sends a data packet. On the other side, the receiver echoes back this timestamp value and in addition timestamps the ACK packet with its own current time.

In this way, the TCP-LP sender measures one-way packet delays. Note that the sender and receiver clocks do not have to be synchronized since we are only interested in the relative time difference. Moreover, a drift between the two clocks is not significant here as resets of dmin and dmax on timescales of minutes can be applied. Finally, we note that by using *one-way* packet delay measurements instead of round-trip times, cross-traffic in the reverse direction does not influence TCP-LP's inference of early congestion.

**TCP-LP Implementation Without Congestion:** TCP-LP is an end-point algorithm that aims to emulate the functionality of the reference-scheduling model. Consider for simplicity a scenario with one TCP-LP and one TCP flow. The reference strict priority scheduler serves TCP-LP packets only when there are no TCP packets in the system. However, whenever TCP packets arrive, the scheduler immediately begins service of higher priority TCP packets.

Similarly, after serving the last packet from the TCP class, the strict priority scheduler immediately starts serving TCP-LP packets. Note that it is impossible to exactly achieve this behavior from the network endpoints as TCP-LP operates on timescales of round-trip times, while the reference scheduling model operates on time-scales of packet transmission times. Thus, our goal is to develop a congestion control policy that is able to *approximate* the desired dynamic behavior. TCP-LP must react quickly to early congestion indicators to achieve TCP-transparency. However, simply decreasing the congestion window promptly to zero packets after the receipt of an early congestion indication (as implied by the reference scheduling model) unnecessarily inhibits the throughput of TCP-LP flows. This is because a single early congestion indication cannot be considered as a reliable indication of network congestion given the complex dynamics of cross traffic [12].

On the other hand, halving the congestion window of TCP-LP flows upon the congestion indication, as recommended for ECN flows, would result in too slow a response to achieve TCP transparency. To compromise between the two extremes, TCP-LP employs the following algorithm. After receipt of the initial early congestion indication, TCP-LP halves its congestion window and enters an *inference phase* by starting an *inference time-out timer*. During this inference period, TCP-LP only observes responses from the network, without increasing its congestion window. If it receives another early congestion indication before the inference timer expires, this indicates the activity of cross traffic and TCP-LP decreases its congestion window to one packet. Thus, with persistent congestion, it takes two round-trip times for a TCP-LP flow to decrease its window to 1. Otherwise, after expiration of the inference timer, TCP-LP enters the additive increase congestion avoidance phase and increases its congestion window by one per round-trip time (as with TCP flows in this phase). We observe that as with router-assisted early congestion indication, consecutive packets from the same flow often experience similar network congestion state. Consequently, as suggested for ECN flows, TCP-LP also reacts to a congestion indication event at most once per round-trip time. Thus, in order to prevent TCP-LP from over-reacting to bursts of congestion indicated packets, TCP-LP ignores succeeding congestion indications if the source has reacted to a previous delay-based congestion indication or to a dropped packet in the last round-trip time. Finally, the minimum congestion window for TCP-LP flows in the inference phase is set to 1. In this way, TCP-LP flows conservatively ensure that an excess bandwidth of at least one packet per round-trip time is available before probing for additional bandwidth. We denote *cwnd* as congestion window size and *itti* as the inference time-out timer state indicator. It is set to one when the timer is initiated and to zero when the timer expires.

Further, a schematic view of TCP-LP's congestion window behavior at different stages, where points on the top mark early congestion indications and the inference timer period is labeled *itti*. For example, with the first early congestion indicator, this flow enters the inference phase. It later successfully exits the inference phase into additive increase as no further early congestion indicators occur.

**Comparing and Reporting:**

- TCP-LP is largely non-intrusive to TCP traffic
- Both single and aggregate TCPLP flows are able to successfully utilize excess network bandwidth; moreover, multiple TCP-LP flows share excess bandwidth fairly
- Substantial amounts of excess bandwidth are available to the low-priority class, even in the presence of "greedy" TCP flows
- The response times of web connections in the best-effort class decrease by up to 90% when long-lived bulk data transfers use TCP-LP rather than TCP

**System Testing:** System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation begins. Software testing can look upon as many process of software development organization performed, that provide the last opportunity to correct any flaws in the developed system. Testing account requires largest percentage of technical effort in the software development process. Testing begins at a module level and word towards the integration of the entire computer based system [7].

Testing and debugging are different activities, but any testing strangely includes debugging strategy. Software testing must accommodate low level test that are necessary to validate that a small source code segment has been correctly implemented as well as high level test that validate major system function, against customer requirements. No testing is complete without the verification and validation. The goals of verification and validation activities are to access and improve the quality of the work product generated during development and modification of software.

**Test Data and Output:** Preparation of test data plays a vital role in System testing. After preparation of test data, system under study is tested using test data. While testing the system using test data, errors are uncovered and corrected by using the following method and correction are noted for future. The system has been verified and validated by running.

**Types of Testing:** During the development of software, errors of various types can occur at any stage. At each phase, different are used to detect the errors. The first major hurdle in the process of implementation is the period of testing the system[5]. The debugging process is the most unpredictable of testing procedure. To make the system develop here to reliable and acceptable, various testing methods are used. The three basic methods out of them are:

- Running the program to identify the errors that might have occurred while feeding the program into the system.
- Applying the screen formats to regular users to gauge the extent to which the screen was comprehensible to the user.
- Presenting the format to the administration for the purpose of obtaining approval and checking if any modifications have to be done or whether the proposed system serves their purpose accurately.

**Unit Testing:** Unit testing involves the tests carried out on modules/programs, which make up a system. The unit test focuses verification effort on the smallest unit of software design model. Using the procedural design description as a guide, important control paths are tested to uncover errors within the boundary of the modules. The unit testing is normally white box oriented and the step can be conducted in parallel of multiple modules.

Unit testing comprises the set of tests performed by an individual prior to the integration of the units into a larger system. There are four types of test a programmer can perform on a program unit. They are functional test, performance test, stress test, structure test. Module interface is tested to ensure that information properly flows in and out of the program. Local data structures are examined to ensure that data stored temporarily maintain its integrity during all steps in an algorithm execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. All error-handling paths are tested.

All independent paths through the control structure are exercised to ensure that all statement in a module have been exercised at least once.

**White Box Testing:**

- By using the white box testing technique, it was that all the individual logical paths were executed at least once.
- All the logical decisions were tested on both their true and false sides.
- All the loops were tested with input data in between the ranges and especially at the boundary values are also tested.

**Black Box Testing:** By the use of the black box testing technique the missing functions were identified and placed in their actual positions. The errors in the interfaces were identified and corrected accurately. This technique was also used to identify the initialization and termination errors and correct them.

**System Testing:** System Testing consists of different sets whose purpose test is to exercise the computer based system fully. There are two kinds of testing. They are integration testing and acceptance testing. Proper planning is required to ensure that each sub units in a module will be available for integration in to evolving a software product. Acceptance testing involve planning and execution of functional tests, performance tests and stress test to verify that the implemented system satisfies the requirement.

**Integration Testing:** In this the different modules of a system are integrating using an integration plan. The integration plan specifies the steps and the order in which modules are combined to realize the full screen. After each integration step, the partially integration system is tested. The primary objective of integration testing is to text the module interface.

An important factor that guides the integration plan is the module dependency graph. The modules dependency graph denotes the order in which different modules call each other. A structure chart is a form of a modules dependency graph.

**Verification:** It refers whether the software functions properly. It gives the details whether the software is correct or not. Similarly it ensures that all testing done is proper. All Types of error are identified and corrected. Integration testing is also done to recover errors associated with the interface.

**Validation:** Validation ensures whether the product meets the requirements of the customer. Validation refers whether the right product is built. After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specifications and are accepted.
- A Validation form specification is uncovered and a deficiency created.

The testing process for TCP-LP: low Priority service via end point congestion control:

TCP-LP Low Priority Service Via End Point Congestion Control
Overlay Network

```
ipaddress========200.200.200.56
Rec=====overlays$OverlayNodeRecord@1833955
Overlay nodes:
SRIRAM - 200.200.200.56
sFileName=====test2.txt
File Name========D:\workspace\TCP-LP\test2.txt

File sent Successfully
```

## CONCLUSION

TCP-LP, a protocol designed to achieve low priority service from endpoints utilize excess bandwidth without significantly perturbing non-TCP-LP flows. a sender-side modification of the TCP congestion control requires no functionality from the network routers nor any other protocol changes [13-17].

## REFERENCES

1. Alouf, S., P. Nain and D. Towsley, 2001. Inferring network characteristics via moment-based estimators. In Proceedings of IEEE INFOCOM ’01, Anchorage, Alaska.
2. Bansal, D., H. Balakrishnan, S. Floyd and S. Shenker, 2001. Dynamic behavior of slowly-responsive congestion control algorithms. In Proceedings of ACM SIGCOMM ’01, San Diego, CA.
3. Brakmo, L. and L. Peterson, 1995. TCP Vegas: End to end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communications, 13(8): 1465-1480.
4. Breslau, L., E. Knightly, S. Shenker, I. Stoica and H. Zhang, 2000. Endpoint admission control: Architectural issues and performance. In Proceedings of ACM SIGCOMM ’00, Stockholm, Sweden.
5. Clark, D.D. and W. Fang, 1998. Explicit allocation of best-effort packet delivery service. IEEE/ACM Transactions on Networking, 6(4): 362-373.
6. Feldmann, A., A. Gilbert, P. Huang and W. Willinger, 1999. Dynamics of IP traffic: A study of the role of variability and the impact of control. In Proceedings of ACM SIGCOMM ’99, Vancouver, British Columbia.
7. Firoiu, V., J.Y. Le Boudec, D. Towsley and Z.L. Zhang, 2002. Theories and models for Internet quality of service. IEEE, 90(9): 1565-1591.
8. Kerana Hanirex, D. and K.P. Kaliyamurthie, 2013. Multi-classification approach for detecting thyroid attacks, International Journal of Pharma and Bio Sciences, 4(3): B1246-B1251.
9. Khanaa, V., K. Mohanta and T. Saravanan, 2013. Comparative study of uwb communications over fiber using direct and external modulations, Indian Journal of Science and Technology, 6(l 6): 4845-4847.
10. Kumar, Giri R. and M. Saikia, 2013. Multipath routing for admission control and load balancing in wireless mesh networks, International Review on Computers and Software, 8(3): 779-785.
11. Kumarave, A. and K. Rangarajan, 2013. Routing alogrithm over semi-regular tessellations, 2013 IEEE Conference on Information and Communication Technologies, ICT 2013.
12. Kumarave, A. and K. Rangarajan, 2013. Algorithm for automaton specification for exploring dynamic labyrinths, Indian Journal of Science and Technology, 6(l 6).
13. Shafaq Sherazi and Habib Ahmad, 2014. Volatility of Stock Market and Capital Flow Middle-East Journal of Scientific Research, 19(5): 688-692.
14. Kishwar Sultana, Najm ul Hassan Khan and Khadija Shahid, 2013. Efficient Solvent Free Synthesis and X Ray Crystal Structure of Some Cyclic Moieties Containing N-Aryl Imide and Amide, Middle-East Journal of Scientific Research, 18(4): 438-443.
15. Pattanayak, Monalisa and P.L. Nayak, 2013. Green Synthesis of Gold Nanoparticles Using Elettaria cardamomum (ELAICHI) Aqueous Extract World Journal of Nano Science & Technology, 2(1): 01-05.
16. Chahataray, Rajashree. and P.L. Nayak, 2013. Synthesis and Characterization of Conducting Polymers Multi Walled Carbon Nanotube-Chitosan Composites Coupled with Poly (P-Aminophenol) World Journal of Nano Science & Technology, 2(1): 18-25.
17. Parida, Umesh Kumar, S.K. Biswal, P.L. Nayak and B.K. Bindhani, 2013. Gold Nano Particles for Biomedical Applications World Journal of Nano Science & Technology, 2(1): 47-57.