Host-to-Crowd Congestion Control for TCP

#C.Soumi[1] **,** M.Tech (S.E), E mail: pvkishorehod@gmail.com
# Ch.Raja Jacob[2]**,Professor,** HOD CSE department**,** e mail: ch.rajajacobnova@gmail.com
#K. Subhashini[3]**, Asst.Professor,** CSE Dept, E mail: ksubhashininova@gmail.com
# Nova College of Engineering, West Godavari , A.P, INDIA

Abstract:- ***The Transmission Control Protocol (TCP) carries most Internet traffic, so performance of the Internet depends to a great extent on how well TCP works. Performance characteristics of a particular version of TCP are defined by the congestion control algorithm it employs. This paper presents a survey of various congestion control proposals that preserve the original host-to-host idea of TCP—namely, that neither sender nor receiver relies on any explicit notification from the network. The proposed solutions focus on a variety of problems, starting with the basic problem of eliminating the phenomenon of congestion collapse, and also include the problems of effectively using the available network resources in different types of environments (wired, wireless, high-speed, long-delay, etc.). In a shared, highly distributed, and heterogeneous environment such as the Internet, effective network use depends not only on how well a single TCP based application can utilize the network capacity, but also on how well it cooperates with other applications transmitting data through the same network. Our survey shows that over the last 20 years many host-to-host techniques have been developed that address several problems with different levels of reliability and precision. There have been enhancements allowing senders to detect fast packet losses and route changes. Other techniques have the ability to estimate the loss rate, the bottleneck buffer size, and level of congestion. The survey describes each congestion control alternative, its strengths and its weaknesses. Additionally, techniques that are in common use or available for testing are described.***

1. INTRODUCTION

MOST CURRENT Internet applications rely on the Transmission Control Protocol (TCP) [1] to deliver data reliably across the network. Although it was not part of its initial design, the most essential element of TCP is congestion control; it defines TCP's performance characteristics. In this paper we present a survey of the congestion control proposals for TCP that preserve its fundamental host-to-host principle, meaning they do not rely on any kind of explicit signaling from the network.1 The proposed algorithms introduce a wide variety of techniques that allow senders to detect loss events, congestion state, and route changes, as well as measure the loss rate, the RTT, the RTT variation, bottleneck buffer sizes, and congestion level with different levels of reliability and precision. The key feature of TCP is its ability to provide a reliable, bi-directional, virtual channel between any two hosts on the Internet. Since the protocol works over the IP network [3], which provides only best-effort service for delivering packets across the network, the TCP standard [1] specifies a sliding window based flow control. This flow control has several mechanisms. First, the sender buffers all data before the transmission, assigning a sequence number to each buffered byte. Continuous blocks of the buffered data are packetized into TCP packets that include a sequence number of the first data byte in the packet. Second, a portion (window) of the prepared packets is transmitted to the receiver using the IP protocol. As soon as the sender receives delivery confirmation for at least one data packet, it transmits a new portion of packets (the window "slides" along the sender's buffer, Figure 1). Finally, the sender holds responsibility for a data block until the receiver explicitly confirms delivery of the block. As a result, the sender may eventually decide that a particular unacknowledged data block has been lost and start recovery procedures (e.g., retransmit one or several packets).To acknowledge data delivery, the receiver forms an ACK packet that carries one sequence number and (optionally) several pairs of sequence numbers. The former, a *cumulative ACK*, indicates that all data blocks having smaller sequence numbers have already been delivered. The latter, a *selective ACK* (Section II-E—a TCP extension standardized 15 years after the introduction of TCP itself), explicitly indicates the ranges of sequence numbers of delivered data packets. To be more precise, TCP does not have a separate ACK packet, but rather uses flags and option fields in the common TCP header for acknowledgment purposes. (A TCP packet can be both a data packet and an ACK packet at the same time.) However, without loss of generality, we will discuss a notion of ACK packets as a separate entity. Although a sliding window based flow control is relatively simple, it has several conflicting objectives. For example, on the one hand, throughput of a TCP flow should be maximized. This essentially requires that the size of a sliding window also be maximized. (It can be shown that the maximum throughput of a TCP flow depends directly on the sliding window size and inversely on the round-trip time of the network path.) On the other hand, if the sliding window is too large, there is a high probability of packet loss because the network and the receiver have resource limitations. Thus, minimization of packet losses requires minimizing the sliding window. Therefore, the problem is finding an optimal value for the sliding window (which is usually referred to as the *congestion window*) that provides good throughput, yet does not overwhelm the network and the receiver. Additionally, TCP should be able to recover from packet losses in a timely fashion. This means that the shorter the interval between packet transmission and loss detection, the faster TCP can recover. However, this interval cannot be too short, or otherwise the sender may detect a loss prematurely and retransmit the corresponding packet unnecessarily. This overreaction simply wastes network resources and may induce high congestion in the network. In other words, when and how a sender detects packet losses is another hard problem for TCP.

2. SYSTEM ANALYSIS

2.1 Existing system:

Existing TCP specification: The standard already requires receivers to report the sequence number of the last in-order delivered data packet each time a packet is received, even if received out of order . For example, in response to a data packet sequence 5,6,7,10,11,12, the receiver will ACK the packet sequence 5,6,7,7,7,7. In the idealized case, the absence of reordering guarantees that an out-of-order delivery occurs only if some packet has been lost. Thus, if the sender sees several ACKs carrying the same sequence numbers (duplicate ACKs), it can be sure that the network has failed to deliver some data and can act accordingly.

2.2 Proposed system:

The Host to Host congestion control proposals that build a foundation for all currently known host-to-host algorithms. This foundation includes

a) The basic principle of probing the available network resources,

b) Loss-based and delay-based techniques to estimate the congestion state in the network,

c) Techniques to detect packet losses quickly

Modules:

2.3Tcp Host to host Network module:

Host-to-host principle, meaning they do not rely on any kind of explicit signaling from the network.1 The proposed algorithms introduce a wide variety of techniques that allow senders to detect loss events, congestion state, and route changes, as well as measure the loss rate, the RTT, the RTT variation, bottleneck buffer sizes, and congestion level

2.2 .Congestion Collapse module:

TCP sender's estimate of the number of data packets the network can accept for delivery without becoming congested. In the special case where the flow control limit (the socalled receiver window) is less than the congestion control limit (i.e., the congestion window), the former is considered a real bound for outstanding data packets. Although this is a formal definition of the real TCP rate bound, we will only consider the congestion window as a rate limiting factor,

assuming that in most cases the processing rate of end-hosts is several orders of magnitude higher than the data transfer rate that the network can potentially offer. Additionally, we will compare different algorithms, focusing on the congestion window dynamics as a measure of the particular congestion control algorithm effectiveness

2.4. Congestion Avoidance & Packet Recovery Module:

Congestion control algorithm, the purpose of which is to reduce consumption of network resources in complex congestion situations. But this expectation rests on the assumption that congestion states, as deduced from each detected loss, are independent, and in the example above this does not hold true. All packet losses from the original data bundle (i.e., from those data packets outstanding at the moment of loss detection) have a high probability of being caused by a single congestion event. Thus, the second and third losses from the example above should be treated only as requests to retransmit data and not as congestion indicators.

Moreover, reducing the congestion window does not guarantee the instant release of network resources. All packets sent before the congestion window reduction are still in transit. Before the new congestion window size becomes effective, we should not apply any additional rate reduction policies. This can be interpreted as reducing the congestion window no more often than once per one-way propagation delay or approximately RTT/2

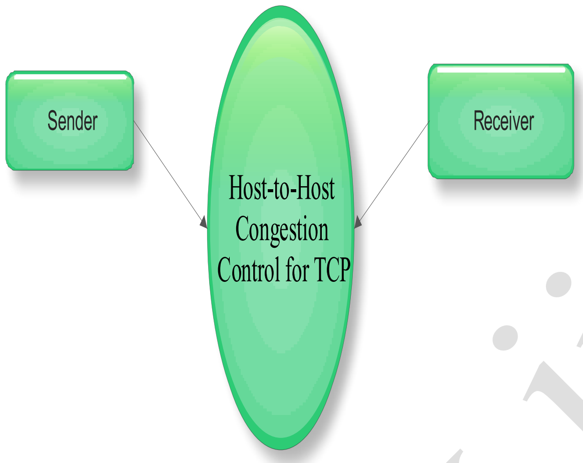
3. SYSTEM DESIGN

3.1 Design Overview

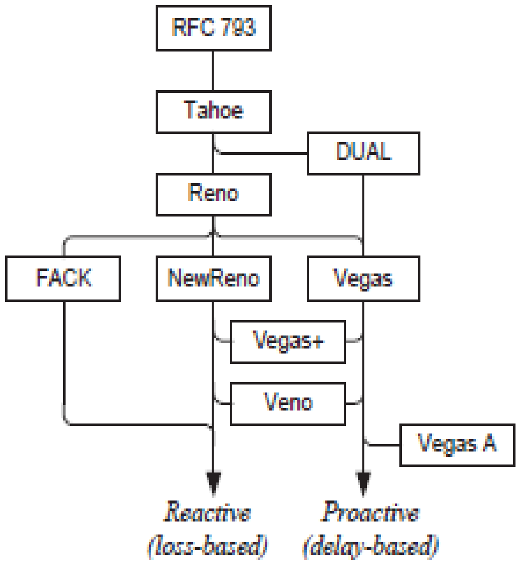
Design involves identification of classes, their relationships as well as their collaboration. In objectiory, classes were divided into entity classs, interface classes and the control classes. The computer Aided Software Engineering tools that are available commercially do not provide any assistance in this transition. Even research CASE tools take advantage of meta modeling are helpful only after the construction of class diagram is completed. In the fusion method, it used some object-oriented approaches like object modeling Technique

(OMT),Clas_Responsibility_Collaborator(CRC) and objectory used the term Agents to represent some of the hardware and software systems. In fusion method, there was no requirement phase, where in a user will supply the initial requirement document. Any software project is worked out by both analyst and designer. The analyst creates the user case diagram. The designer creates the class diagram. But the designer can do this only after

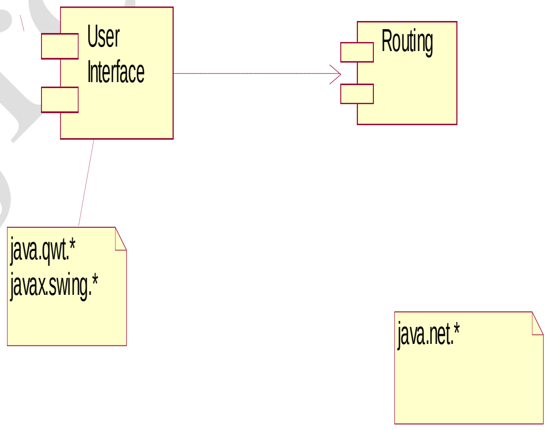
3.2 Data Flow Diagram



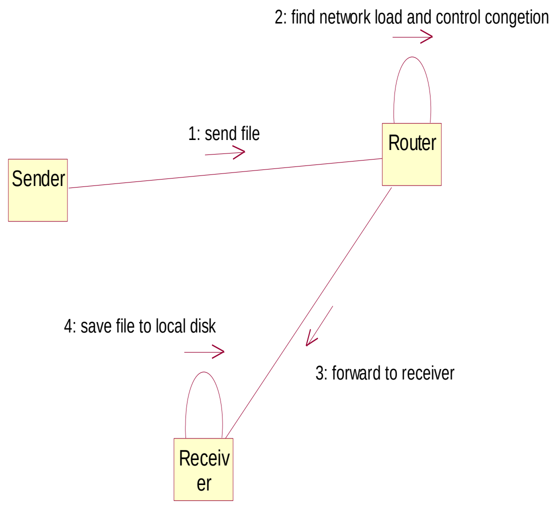
System Architecture



Component Diagram:



Collaboration Diagram:


2: find network load and control congetion
1: send file
Sender
Router
4: save file to local disk
3: forward to receiver
Receiver

4. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5. TYPES OF TESTS

5.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

5.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

5.3 Functional test

Functional tests provide a systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation , and user manuals.
Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify
Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.4 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5.5 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

5.6 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

5.7 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.8 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6. CONCLUSION

In this work we have presented a survey of various approaches to TCP congestion control that do not rely on any explicit signaling from the network. The survey highlighted the fact that the research focus has changed with the development of the Internet, from the basic problem of eliminating the congestion collapse phenomenon to problems of using available network resources effectively in different types of environments (wired, wireless, high-speed, long-delay, etc).

7. REFERENCES

[1] G. Yang, R. Wang, M. Sanadidi, and M. Gerla, "TCPW with bulk repeat in next generation wireless networks," *IEEE International Conference on Communications 2003*, vol. 1, pp. 674–678, May 2003.
[2] R. Ludwig and R. H. Katz, "The Eifel algorithm: making TCP robustagainst spurious retransmissions," *SIGCOMM Computer CommunicationReview*, vol. 30, no. 1, pp. 30–36, 2000.
[3] R. Ludwig and A. Gurtov, "RFC4015—the Eifel response algorithm for TCP," *RFC*, 2005.
[4] F. Wang and Y. Zhang, "Improving TCP performance over mobile adhoc networks with out-of-order detection and response," in *Proceedingsof the 3rd ACM international symposium on mobile ad hoc networking& computing*, New York, NY, 2002, pp. 217–225.
[5] S. Bohacek, J. Hespanha, J. Lee, C. Lim, and K. Obraczka, "TCPPR:TCP for Persistent Packet Reordering," in *Proc. InternationalConference on Distributed Computing Systems*, vol. 23, 2003, pp. 222–233.
[6] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "RFC2883—AnExtension to the Selective Acknowledgement (SACK)," *RFC*, 2000.
[7] M. Zhang, B. Karp, S. Floyd, and L. Peterson, "RR-TCP: a reordering robust TCP with DSACK," International Computer Science Institute,Tech. Rep. TR-02-006, July 2002.
[8] "RR-TCP: a reordering-robust TCP with DSACK," in *Proc. 11th IEEE International Conference on Network Protocols (ICNP)*, 2003, pp. 95–106.
[9] A. Kuzmanovic and E. Knightly, "TCP-LP: low-priority service via end-point congestion control," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 739–752, 2006.
[10] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *EEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 362–373, 1998. [11] X. Xiao and L. Ni, "Internet QoS: A big picture," *IEEE Network*, vol. 13, no. 2, pp. 8–18, 1999.
[12] B. Davie, "Deployment experience with differentiated services," in *Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care?*, New York, NY, 2003, pp. 131–136.
[13] A. Venkataramani, R. Kokku, and M. Dahlin, "TCP Nice: A Mechanism for Background Transfers," *Operating Systems Review*, vol. 36, pp. 329–344, 2002.
[14] A. Kuzmanovic and E. W. Knightly, "TCP-LP: a distributed algorithm for low priority data transfer," in *Proc. IEEE INFOCOM*, April 2003.
[15] J.-H. Choi and C. Yoo, "One-way delay estimation and its application," *Computer Communications*, vol. 28, no. 7, pp. 819–828, 2005.
[16] A. Kuzmanovic, E. Knightly, and R. Les Cottrell, "HSTCP-LP: A protocol for low-priority bulk data transfer in high-speed high-RTT networks."
[17] K. Ramakrishnan, S. Floyd, and D. Black, "RFC3168—the addition of explicit congestion notification (ECN)," *RFC*, 2001.

Bibliography:

1. **c.soumi** received her B.Tech.degree in Computer Science Engineering from JNTUA, Andra Pradesh, INDIA in 2008. She is currently doing M.Tech degree (Software Engineering) in NOVA College of Engineering, JNTUH, Jangareddi Gudem, West Godhavari(Dist), , Andra Pradesh, INDIA.

2. **Ch.Raja Jacob** working as a Professor & HOD,CSE department, NOVA College of Engineering, JNTUH, Jangareddi Gudem, West Godhavari(Dist), , Andra Pradesh, INDIA.
3.
3.K.Subhashini working as a Asst. Professor in CSE department, NOVA College of Engineering, JNTUH, Jangareddi Gudem, West Godhavari(Dist), Andra Pradesh, INDIA.