Network and Complex Systems www.iiste.org
ISSN 2224-610X (Paper) ISSN 2225-0603 (Online)
Vol.5, No.8, 2015

Simulation of an Optimized Data Packet Transmission in a Congested Network

Promise T. Akiene Ledisi G. Kabari
School of Applied Sciences, Ken Saro-Wiwa Polytechnic, P. M. B. 20, Bori, Nigeria

Abstract
Computer network and the Internet nowadays accommodate simultaneous transmission of audio, video, and data traffic among others. Efficient and reliable data transmission is essential for achieving high performance in a networked computing environment. Thus, there is need to optimized data packet transmission in the present day network. This paper simulates and demonstrates the process of optimizing data packet transmission in a congested network. It uses the modified FIFO Queue system to control data packet loss and uses the prototyping software methodology to develop software in Python Programming language for its implementation. From the simulation process, it was observed that causes of packet loss during transmission are largely dependent on protocol, congestion of traffic way, speed of the sender and speed of the receiver's machine. Thus, the paper takes advantage of the observations from simulation and presents a system that simulates control of data loss during transmission in a congested network.
Keywords: Simulation, Auxiliary Queue, Departing Packets, Arrival Packets, Packet Loss.

1. Introduction

Data loss is a problem that occurs on congested networks when users send data or contend for access to the same resources (bandwidth, buffers, and queues). It is important to avoid high data loss rate in the internet. When a packet is dropped before it reaches its destination, all of the resources it has consumed in transit have been wasted. In extreme cases, this situation can lead to congestion collapse. Finite network resources bring about unavoidable competition among transmitted packets, resulting in network congestion and hence data/packet loss. This requires the network or Internet that will guarantee no or less data/packet loss.

In packet-switched networks, packets move in and out of the buffers and queues of switching devices as they traverse the network. In fact, a packet-switched network is often referred to as a "network of queues." A characteristic of packet-switched networks is that packets may arrive in bursts from one or more sources. Buffers help routers absorb bursts until they can catch up. If traffic is excessive, buffers fill up and new incoming packets are dropped which leads to packet loss. Increasing the size of the buffers is not a solution, because excessive buffer size can lead to excessive delay.

Congestion typically occurs where multiple links feed into a single link, such as where internal LANs are connected to WAN links. Congestion also occurs at routers in core networks where nodes are subjected to more traffic than they are designed to handle. When there is congestion, there is packet drop, these dropped packets are lost. Hence the problem of packet lost.

Packet loss is the failure of one or more transmitted packets to arrive at their destination. This event can cause noticeable effects in all types of digital communications. The effects of packet loss among others are Packet loss produces errors, in videoconference environments it can create jitter, in pure audio communications, such as VoIP, it can cause jitter and frequent gaps in received speech, in the worst cases, packet loss can cause severe mutilation of received data, broken-up images, unintelligible speech or even the complete absence of a received signal.

The causes of packet loss include among others: inadequate signal strength at the destination, natural or human-made interference, excessive system noise, hardware failure, software corruption or overburdened network nodes, the protocol in use, congestion of traffic way, Speed of the sender and the speed of the receiver's machine. Often more than one of these factors is involved. In this case the only area considered is the congestion of network traffic way.

The aim of this paper is to present a simulation of an optimized data packet transmission in a congested network. The simulation demonstrates how to handle loss of packets during data transmission in a congested network. Hence the paper will examine the causes of data packet loss during transmission and develop an application program that will simulate packets loss, control and elimination of data loss during transmission. The paper proffers a means of pinning down the degree of package losses at any given router or link, within which a transmission path is seen. This goes a long way to identify which router(s) or path where congestion can be felt (where source and destination are incommunicado).

As congestion wastes the scarce energy due to a large number of retransmissions and packet drops, the proposed system protocol can save energy at each node, given the reduced number of retransmissions and packet losses.

2. Related Works

Over the last decades, Transmission Control Protocol (TCP) and its congestion control mechanisms have been instrumental in controlling packet loss and in preventing congestion collapse across the internet. Researchers have spent a great deal of effort exploring alternative and additional mechanisms for TCP and related technologies in lieu of potential network overload problems. Some techniques have been implemented; others left behind and still others remain on the drawing board.

Good and bad network performance is largely dependent on the effective implementation of network protocols. TCP, easily the most widely used protocol in the transport layer on the Internet (e.g. HTTP, TELNET, and SMTP), plays an integral role in determining overall network performance. Amazingly, TCP has changed very little since its initial design in the early 1980's. A few improvements have been added, but for the most part, the protocol has withstood the test of time. However, there are still a number of performance problems on the internet and fine tuning TCP software continues to be an area of work for a number of people (Semke et al., 1998).

2.1 Congestion

In data networking and queuing theory, network congestion occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queuing delay, packet loss or the blocking of new connections. A consequence of the latter two effects is that an incremental increase in offered load leads either only to a small increase in network throughput, or to an actual reduction in network throughput (Al-Bahaddili, 2012)

Network protocols which use aggressive retransmissions to compensate for packet loss tend to keep systems in a state of network congestion, even after the initial load has been reduced to a level which would not normally have induced network congestion. Thus, networks using these protocols can exhibit two stable states under the same level of load. The stable state with low throughput is known as congestive collapse.

Modern networks use congestion control and congestion avoidance techniques to try to avoid congestion collapse. These include: exponential back off in protocols such as 802.11 CSMA/CA and the original Ethernet, window reduction in TCP, and fair queuing in devices such as routers. Another method to avoid the negative effects of network congestion is implementing priority schemes, so that some packets are transmitted with higher priority than others. Priority schemes do not solve network congestion by themselves, but they help to alleviate the effects of congestion for some services. An example of this is 802.1p. A third method to avoid network congestion is the explicit allocation of network resources to specific flows. One example of this is the use of Contention-Free Transmission Opportunities (CFTXOPs) in the ITU-T Ghn standard, which provides high-speed (up to 1 Gbit/s) local area networking over existing home wires (power lines, phone lines and coaxial cables).

One of the key problems in network design is ensuring that available capacity is fairly and efficiently shared between competing end points. Traditionally, fairness has been achieved either in the network itself using fair queuing at routers (Kurose and Ross, 2008), or through the cooperation of end hosts using a common congestion control protocol such as TCP. Unfortunately, both approaches have significant drawbacks: fair queuing is expensive to implement, while end-host congestion control is typically far from optimal and, critically, relies on the goodwill of end hosts for success (Savage et al., 1999; Sherwood et al., 2005).

2.2 Congestion Control Mechanisms

Many Congestion Control Algorithms have been designed namely: Random Early, Detection (RED), Back Pressure Technique, Choke packet Technique, Implicit Congestion Signaling, Additive Increase and Multiplicative Decrease (AIMD), Explicit Congestion Notification (ECN) in TCP/IP, Binary Congestion Notification(BCN)(Forouzan and Fegen, 2007).

RED: This mechanism detects incipient congestion by computing average queue size and would notify connections of congestion either by dropping packets arriving at it or by setting a bit in packet headers (Sally and Van, 1993).

Backpressure Technique: If a node becomes congested then it slows down or stops receiving the packets from the nodes from which it is receiving packets. If this restriction persists for long then packet sending nodes themselves become congested which in turn propagate the restriction on their preceding nodes.

A choke packet is a packet sent by a node to the source to inform it of congestion. Note the difference between the backpressure and choke packet methods. In backpressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station. In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly.

Implicit Congestion Signaling: When the sending source comes to know of congestion at a node if the propagation delays of packets are detected that is the delay is longer than fixed propagation delay and it may ultimately lead to packet discard. But the sending node should have a mechanism to detect increased delays and packet discards.

Explicit Congestion Notification in TCP/IP (ECN): The purpose of this method is to react to congestion

in a controlled and in a fair manner. It especially operates over connection oriented networks. In this method the network alerts the end systems about the growing congestion within the network on the end systems.
Binary Congestion Notification (BCN): In TCP/IP based networks, congestion is indicated by dropping packets at congested routers. Packets are dropped when the queue of the router reaches its limit (drop tail scheme)

2.3 Open-Loop Congestion Control

In open-loop congestion control, policies are applied to prevent congestion before it happens. In these mechanisms, congestion control is handled by either the source or the destination (Anderson et al., 2006). These policies includes: Retransmission Policy, Window Policy, Acknowledgment Policy, Discarding Policy and Admission Policy.

Retransmission is sometimes unavoidable. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. Retransmission in general may increase congestion in the network. However, a good retransmission policy can prevent congestion. The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion

Window Policy: The type of window at the sender may also affect congestion. The Selective Repeat window is better than the Go-Back-N window for congestion control. In the Go-Back-N window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver. This duplication may make the congestion worse. The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted.

The acknowledgment policy imposed by the receiver may also affect congestion. If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion. Several approaches are used in this case. A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires. A receiver may decide to acknowledge only N packets at a time. We need to know that the acknowledgments are also part of the load in a network. Sending fewer acknowledgments means imposing fewer loads on the network.

A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission. For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated. An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual-circuit networks, Switches in a flow first check the resource requirement of a flow before admitting it to the network. A router can deny establishing a virtual-circuit connection if there is congestion in the network or if there is a possibility of future congestion.

2.4 Closed-Loop Congestion Control

Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols, some these policies are: Backward Signaling, Forward signaling and Congestion Window. In backward signaling, a bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets. In forward signaling, a bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion. In congestion window, the sender window size is determined by the available buffer space in the receiver. In other words, we assumed that it is only the receiver that can dictate to the sender the size of the sender's window. We totally ignored another entity here—the network. If the network cannot deliver the data as fast as they are created by the sender, it must tell the sender to slow down. In other words, in addition to the receiver, the network is a second entity that determines the size of the sender's window.

Today, the sender's window size is determined not only by the receiver but also by congestion in the network. The sender has two pieces of information: the receiver-advertised window size and the congestion window size (Allman et al., 2002).

2.5 Scheduling for Congestion Management

Scheduling policies are used to manage congestion. Some of the scheduling policies used are: First-in, first-out(FIFO) queuing, Priority Queuing and Weighted Fair Queuing.

In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded. A FIFO queue is familiar to those who have had to wait for a bus at a bus stop. Figure1 shows a conceptual view of a FIFO queue.

In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue. The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last. Note that the system does not stop serving a queue until it is empty. Figure2 shows priority queuing with two priority levels (for simplicity).

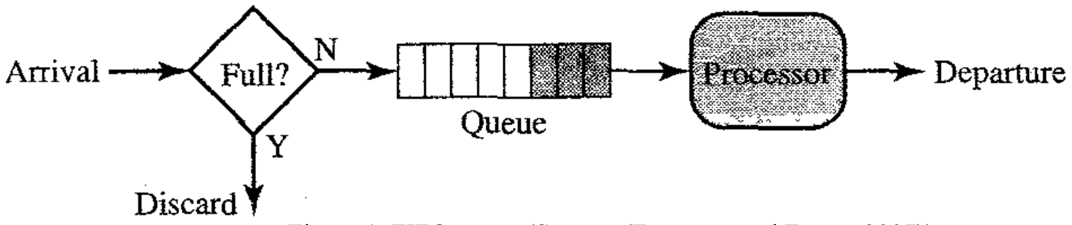


Figure 1. FIFO queue (Source: (Forouzan and Fegen, 2007))

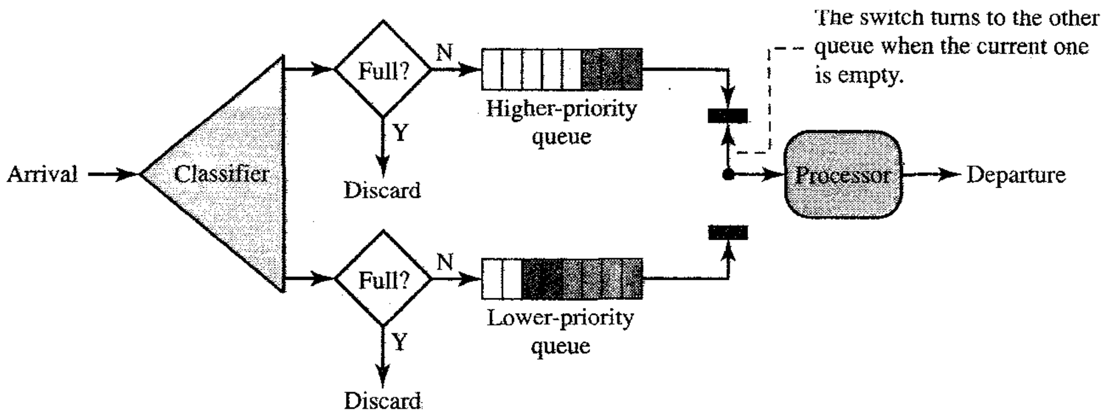


Figure 2. Priority queuing (Source(Forouzan and Fegen, 2007))

A priority queue can provide better Quality of Service (QoS) than the FIFO queue because higher- priority traffic, such as multimedia, can reach the destination with less delay. However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed. This is a condition called starvation.

A better scheduling method is weighted fair queuing. In this technique, the packets are still assigned to different classes and admitted to different queues. The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight. The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight. For example, if the weights are 3, 2, and 1, three packets are processed from the first queue, two from the second queue, and one from the third queue. If the system does not impose priority on the classes, all weights can be equal. In this way, we have fair queuing with priority. Figure3 shows this technique with three classes.

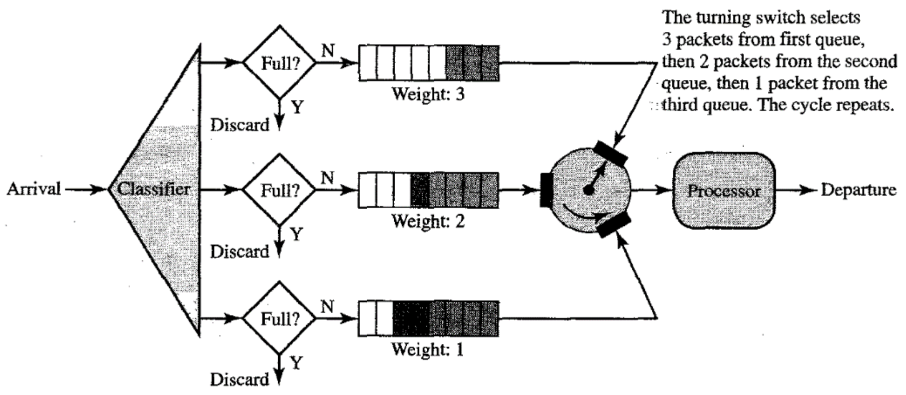

The turning switch selects 3 packets from first queue, then 2 packets from the second queue, then 1 packet from the third queue. The cycle repeats.
Arrival
Classifier
Full?
N
Y
Discard
Weight: 3
Weight: 2
Weight: 1
Processor
Departure

Figure3. Weighted fair queuing (source: (Evans and Washburn, 1993))

2.6 Traffic Shaping

Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic: leaky bucket and token bucket.

Leaky Bucket: If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate. Figure4 shows a leaky bucket implementation.

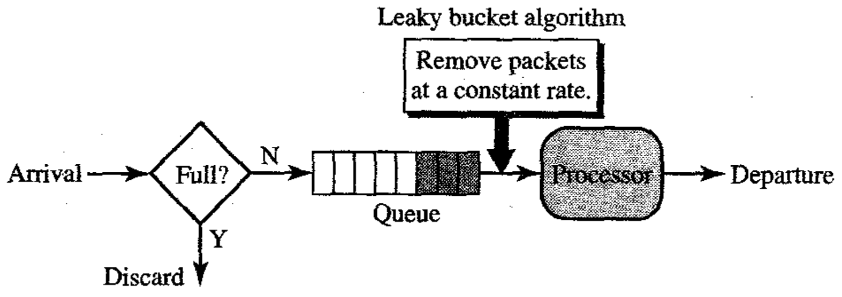


Figure 4. Leaky bucket(source:(Kauffels, 2001))

Token Bucket: The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens. For each tick of the clock, the system sends n tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if n is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens. Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick. In other words, the host can send bursty data as long as the bucket is not empty. Figure5 shows the idea. The token bucket can easily be implemented with a counter. The token is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.

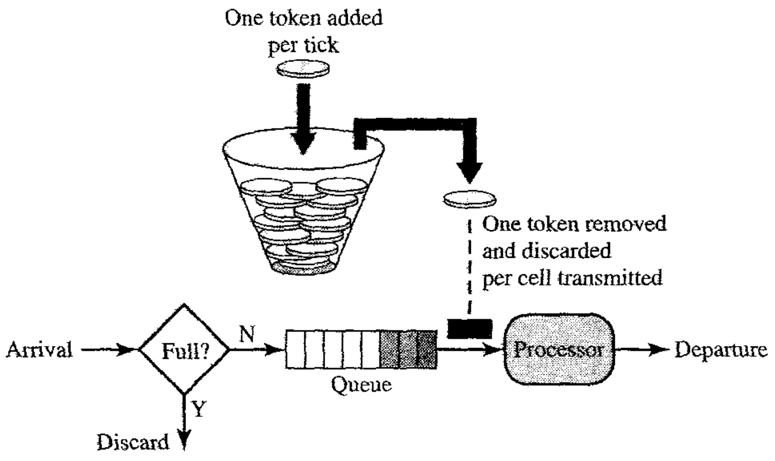


Figure 5. Token bucket(source (Kauffels, 2001))

3. System Design and Analysis

The prototyping software engineering methodology was used in order to reduce development cost and increases systems development speed. The method enabled us to detect faults early enough to avoid project abandonment and delivers product quality easily.

3.1 Weaknesses of the Present/Existing System

Packets can be lost in a network because they may be dropped when a queue in the network node overflows. Hence, it shows that more arrivals of data packets lead to overflow and network congestion. This leads to packet drop and hence packet loss. Considering equation (1)

$$N(K+1) = N(K) + \left(N_{entry}(K) - N_{exit}(K)\right) \dots (1)$$

called Queue Conservation Equation, where: N(K+1) is the expected data packets at a particular point in time, N(K) is the number of data packets in the receiver (section between the congestion point and the switch), N_{entry} is the number of data packets that are entering the queue, Whereas N_{exit} is the number of data packets that are leaving the queue.

From equation (1), it can be seen that it is expected to determine the congestion of data packets at a certain point where there is a queue. It is a data packet congestion detector that notifies the congestion state of the network and thereby causing packet drop. The dropped packets are lost, it cannot effectively manage congestion nor handle data packet loss. This is the main problem considered and identified in this old system/model. Figure6 shows a pictorial representation of equation (1).

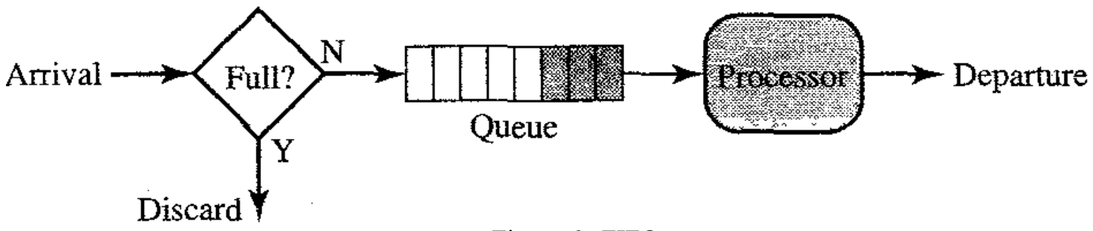


Figure 6. FIFO queue

3.2 Design of the New System

Packets can be lost in a network because they may be dropped when a queue in the network node overflows. Hence, it shows that more arrivals of data packets lead to overflow and network congestion. This leads to packet drop and hence packet loss. That is, arrival of packets is greater than the exit of packets.

Considering equation (1) for queue conservation principle, increasing the number of packets leaving/exiting the queue (N_{exit}), will definitely increase the flow in the queue. With this, there will be no overload, no congestion, no packet drop hence no packet loss. From equation (1), it can be seen that it is essential to determine the congestion of data packets at a certain point where there is a queue. A data packet congestion detector/switch that notifies the congestion state of the network and causing diversion to the second and third routes (queue) is the proposed novel model. Hence the equation (1) is now redesigned as in equation (2).

$$N(K+1) = \sum_{i=1}^{3} N(K) + \left(N_{entry} - N_{exit}(K)\right) i \quad \ldots(2)$$

Where N(K+1) is the expected data packets at a particular point in time, N(K) is the number of data packets in the receiver (section between the congestion point and the congestion detector), N_{entry} is the number of data packets that are entering the queue, Whereas N_{exit} is the number of data packets that are leaving the queue, $\sum$ is the summation of the three queues/processors so proposed. Figure7 shows a pictorial representation of equation (2) as it is simulated.

As seen in figure7 in the simulation, arrival packets are received first into a switch, which checks for the status of the main queue and makes decision whether to send arrival packet(s) to the main queue or auxiliary queues. If main queue is full then packets are sent to the auxiliary queues. In all cases, a processor is attached which processes the packet(s) before delivery. Auxiliary queues and processors are used only when main queue is full.

4. System Implementation

Python programming language was used for the implementation. Python is an open-source object oriented language. Its efficacy cuts across, Application Programming Interface (API), Platform independence, simulation, low-level programming, object linker embedding, network configuration etc. Once a Python program is developed, it does not need to be compiled each time it is about to run. The overwhelming reason for choosing Python for the implementation of this work is the way and manner in which python interacts with network devices. Besides, short codes are used to implement great task.

There are mainly two major classes to the system specification. One of it is identifying the offensive location (the nonbehaving source) and the next is simulation of practical model that demonstrates clear evidences of minimizing network congestion, packet drop, data/packet loss. The first class of specification has several modules viz connection, query module, report module. All these are batched separately. They are executed only when program is in active state.

The system checks the status of each switching device that is linked to the nonbehaving computer. The checks consider the timeout for a particular queue to finish processing. This is used to evaluate the queue status; a report is made based on this evaluation. This task has to do with the various module of workflow as mentioned earlier. Firstly, connectivity is made through which the information super highway is accessed. Another module identifies the various nodes involved; between the sending location and the receiving location. The module that checks the time-out assumes ten seconds as maximum transmission time.

The next specification class combines threads of processes by way of simulation and displays a propitious model of First-In First-Out Queue Theory; where arrival packets are screened under the proviso of queue size. This check is to ensure that when queue is full, arriving packets are redirected to auxiliary queues for onward processing.

4.1 Hardware and Software Requirement

The implementation of the system requires the following list of hardware components: Monitor, System Unit (processor speed of 1GHz and above, RAM size of 2GB and above), Keyboard/mouse, Uninterrupted power supply (UPS), Modem, Network Interface Card, Hub, Routers, Bridges, Gateway, Connecting cables, Satellite Dish.

The system equally requires the following list of software: Python Interpreter, Internet Information Service(IIS), Operating System (Unix, Linux, Windows etc.), Component driver programs, Text Editor, Integrated Development Environment, Proposes System source code, Web browser program.

4.2 System Output/Demonstration

The system was finally tested and demonstrated, a screen shot of the system are as shown in figure7, figure 8, figure 9, figure 10 and figure 11.

Figure 7 demonstrates the schematic design of the policy (system) comprising of the arrival packets using our designed policy to check whether the main queue is full or not and consequently enter into the main queue or diverted to the auxiliary queuue1 or auxiliary queue2. Figure 8 demonstrates when the main queue is full and the packets are then diverted to auxiliary queue 1 and auxiliary queue 2. Figure 9 demonstrates when auxiliary queue 1 full and packets are then diverted to auxiliary queue 2. Figure 10 demonstrates when both auxiliary queue 1 and auxiliary queue 2 are full and the main queue is now decongested. Figure 11 shows the system report.

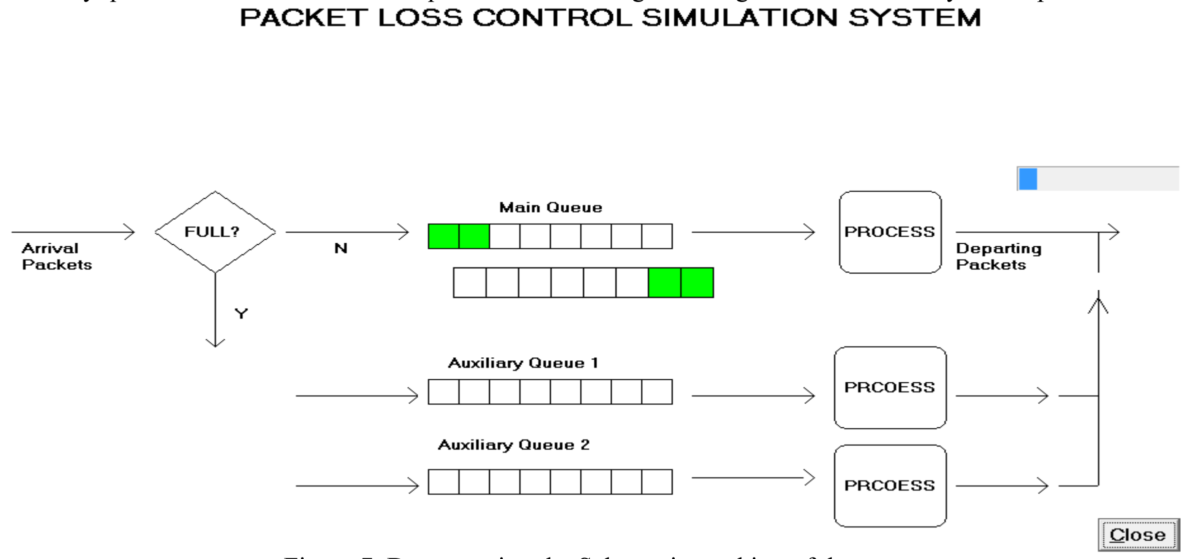


Figure 7. Demostrating the Schematic working of the sytem.

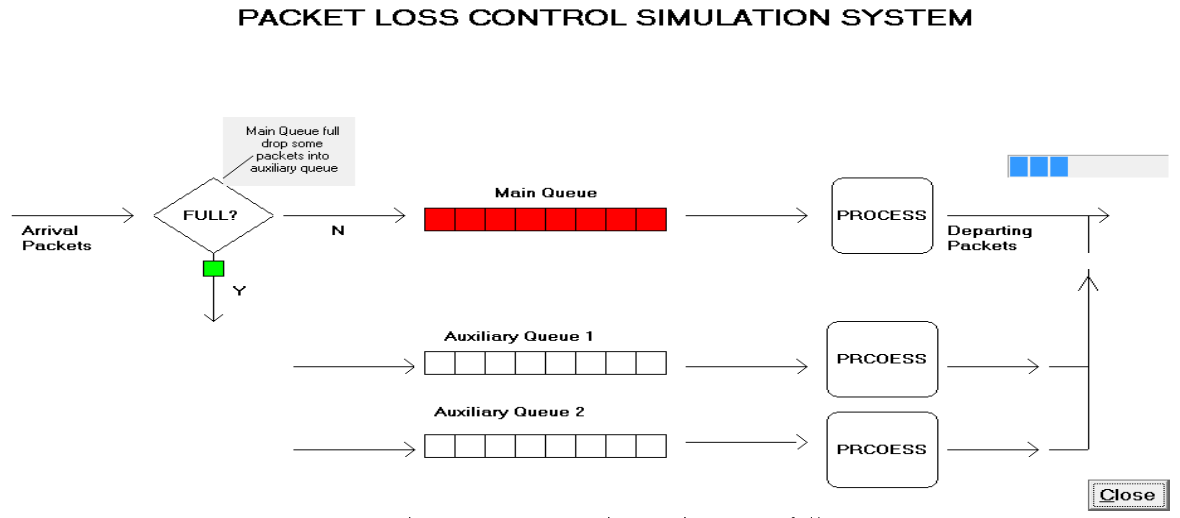


Figure 8. Demostrating Main Queue full

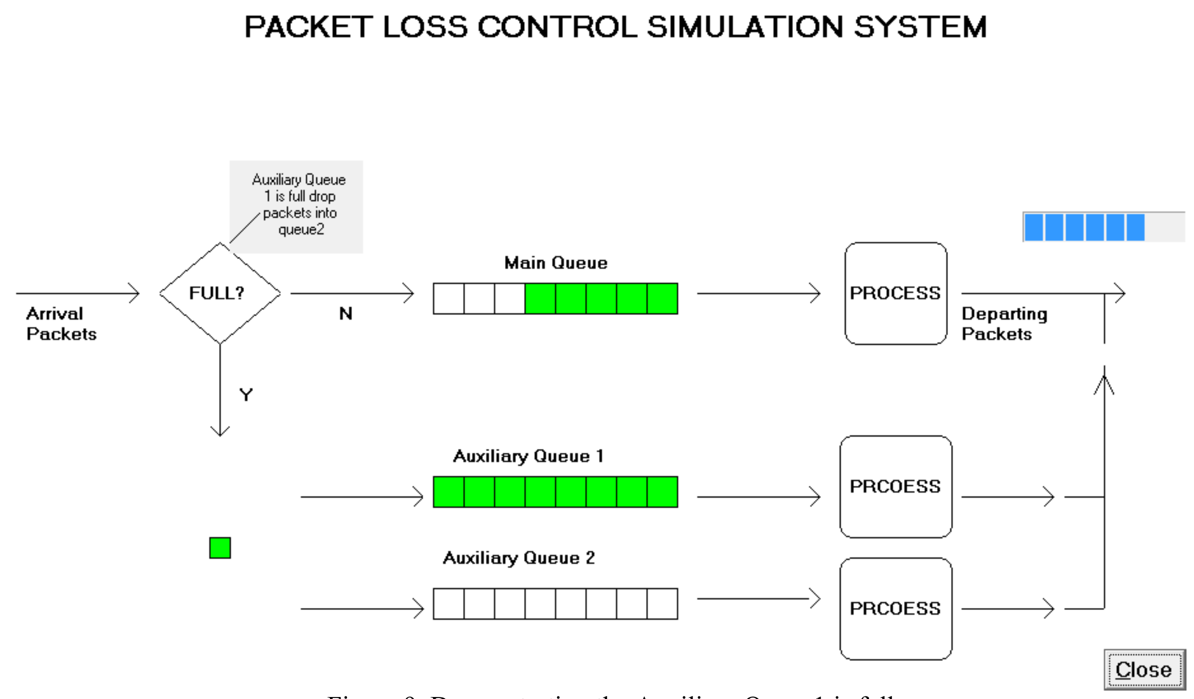


Figure 9. Demonstrating the Auxiliary Queue1 is full

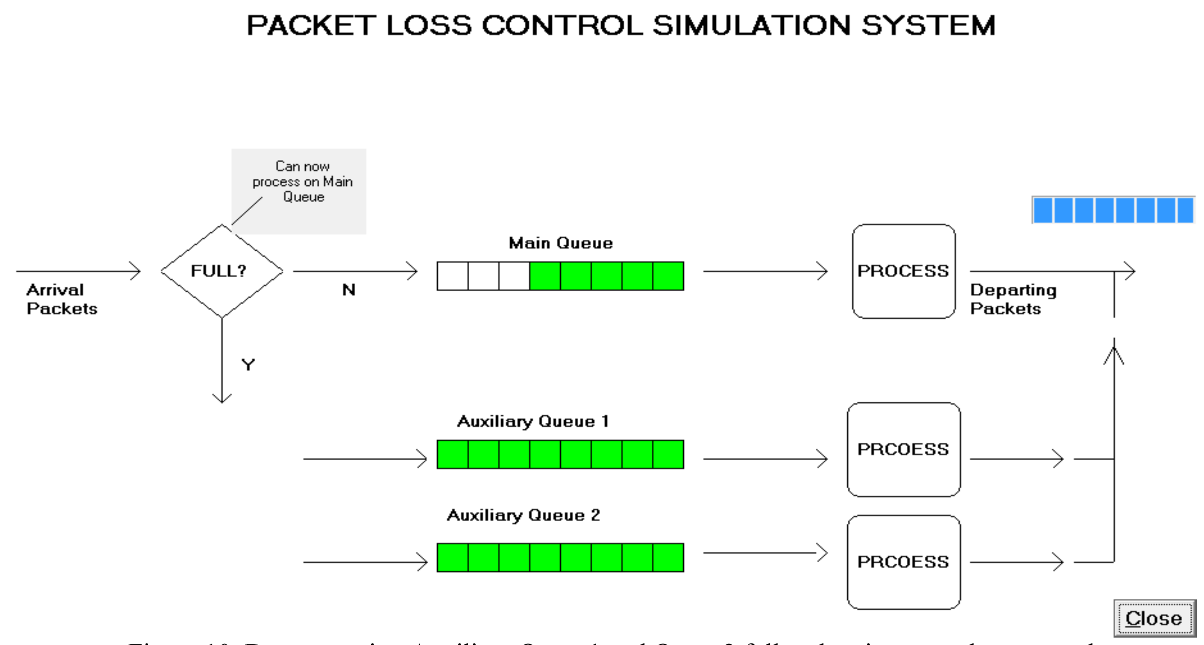
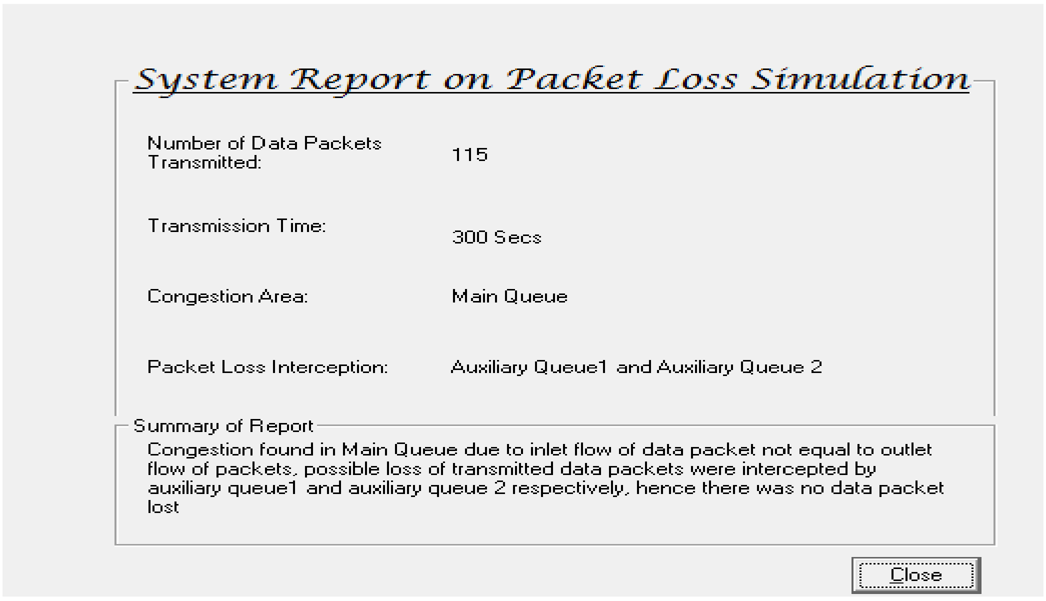


Figure 10. Demonstrating Auxiliary Queue1 and Queue2 full and main queue decongested


System Report on Packet Loss Simulation

Number of Data Packets Transmitted: 115

Transmission Time: 300 Secs

Congestion Area: Main Queue

Packet Loss Interception: Auxiliary Queue1 and Auxiliary Queue 2

Summary of Report
Congestion found in Main Queue due to inlet flow of data packet not equal to outlet flow of packets, possible loss of transmitted data packets were intercepted by auxiliary queue1 and auxiliary queue 2 respectively, hence there was no data packet lost

Figure 11. Showing System Report

5. Conclusion

The paper demonstrates in practical terms the solution to Packet/data loss which normally caused some level of psychological phobia and trauma to some network and computer users.

Packet transmission permits waiting for process completion, this is crucial to the experience of packet/data loss in a congested network. From the simulation demonstrated, it is evident that network congestion is tangential to system speed, rate of transmission, bandwidth, switching device types(router or hub), transmission medium, number of arrival packets against number of processed packet and that of departure packets. This implies that if packet arrival is greater than packet departure then there is bound to be network congestion hence packet drop and packet loss, and if packet arrival is less than packet departure then there is bound to be no network congestion hence no packet drop and no data/packet loss . The traditional packet drop as a solution to network congestion is discouraged by the simulation shown. Rather a modified FIFO Queue system used in this simulation should be adopted and implemented, which is more trusted than the former. However, if packets are constantly put below queue capacity then network congestion will be avoided and there will be no packet/data loss.

References

Al-Bahadili, H. (2012), "Simulation in computer network design and modelling: Use and Analysis", Hershey, PA: IGI Global, P. 282.

Allman, M., Floyd, S,. & Partridge, C. (2002). "Increasing tcp's initial window". RFC 3390.

Anderson, E., Greenspun, P., and Grumet, A.(2006), "Internet Application Workbook. MIT Press 2006; ISBN 0262511916 Retrieved on 13/10/2014 from http://philip.greenspun.com/internet-application-workbook/

Behrouz A. Forouzan and Sophia Chung Fegen (2007), Data Communication and Networking, 4th ed ISBN-13978-0-07-296775-3 McGraw-Hill Forouzan Networking series

Evans, J. and Washburn, K.,(1993), "TCP/IP Running A Successful Network", Addison-Wesley Publishing Company, Wokingham, England.

Kauffels, Franz-Joachim (2001), "Network Management: Problems, Standards and Strategies", Network News West Yorkshire, England :MCB Univesity Press

Kurose, James and Ross, Keith (2008), " Computer Networking – A Top-Down Approach (4th ed.). Addison Wesley. ISBN 978-0-13-607967-5.

Sally, Floyd and Van Jacobson, M. (1993), " Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking.

Savage, S., Cardwell, N., Wetherall, D. and Anderson, T(1999), "TCP congestion control with a misbehaving receiver", ACM SIGCOMMCCR, 29(5):71–78.

Semke, J. Mahdavi, J. and Mathis, M. (1998), "Automatic TCP Buffer Tuning", Computer Communications Review, ACM SIGCOMM, Volume 28, Number 4, October 1998.

Sherwood, R. Bhattacharjee, B. and Braud, R.(2005), "Misbehaving TCP receivers can cause Internet-wide congestion collapse". In Proceedings of ACM CCS, 2005