A TREAP-BASED CONGESTION CONTROL MODEL FOR M/G/k QUEUE NETWORK

[1] *Gbadebo D. A., [1] Abimbola G. O., [1] Iposu O. N. & [2]Salami, I. A.
[1]*Department of Computer Science, College of Information and Technology Education,*
[2]*Department of Mathematics, College of Science Education,*
Lagos State University of Education, Oto / Ijanikin, Lagos State, Nigeria.
**Corresponding author: gbuyi2010@yahoo.com*

ABSTRACT

This study presents a Treap-based Congestion Control Model (Tb-CCM) for ensuring stability in M/G/k queue network. In this model, packets of varying sizes of data were transmitted from *N* sources over time into the network, consequently resulting in traffic. Since the model has one server at inception, it could not process more than one packet at a time. Packets arriving in the network whenever the server is busy were arranged as nodes in a tree. These packets arrive in a Poisson distribution process of $\alpha(k) = (\lambda^k/k!)e^{-\lambda}$ and the general distribution process for packet's service time is an arbitrary probability density function $f_y(\gamma)$. The model combines the features of max- heap and binary search trees which are manipulated to manage and transmit nodes to the server based on node rotations. The algorithm of the proposed model and that of Random Early Detection with Reconfigurable Maximum Dropping Probability (RRMDP) were implemented using Optimized Network Evaluation Tool (OPNET) 14.5 simulator. The performance of both methods in network throughput, latency, average queue size management and queuing delay were compared. Simulation results showed a considerable improvement in the performance of the proposed model in network throughput, latency, queuing delay as well as average queue size. Consequently, it was concluded that the proposed model is more effective in the management of M/G/k queue networks.

Keywords: Congestion, binary search tree, heap, tree, packets, sub-destination

INTRODUCTION

Advancement in telecommunication technology is now being directed at adapting traffic patterns to network congestion control. Congestion management schemes are designed such that transmission rates increase linearly when there are no congestion signals. Conversely, when congestion is detected, the rate decreases by a multiplicative factor. This is the case of transmission control protocol (TCP) in the Internet as congestion is detected at the source through signals such as packet losses or some negative acknowledgment mechanisms. Owing to the continuous increase in the rate of data transmission over networks, there is the need to ensure network congestions are addressed as they occur. TCP congestion control mechanism is used to prevent congestion collapse while Active Queue Management (AQM) schemes have been proposed to complement the TCP network congestion control (Hamdi, et al. 2020). In AQM schemes, performance of two thresholds over single threshold is reported. Two thresholds can always be adjusted to give a lower delay for the same throughput. Luo, et al. (2018) opine that AQM scheme used with priority structures is able to provide better quality of service (QoS), reduce traffic congestion and packet delays.

In a study on methods of avoiding queue overflow and reducing queue delay at evolved-NodeB (eNodeB) in Long-Term Evolution (LTE) networks using congestion feedback mechanism, Adesh and Renuka (2019) suggested the use of controlled de- lay in which the channel rate is changed such that

the channel queue capacity can be adapted based on data weight. Result showed that the proposed algorithm outperformed Random Early Detection (RED) gateway. Similarly, Miao (2019) had proposed an improvement of service quality using queue method in internet topology. The study used First-In, First-Out (FIFO), Random RED and Per-Connection Queue (PCQ) methods. The system allowed several users to simultaneously download data at a rate of 128 kilobytes per second. From the results, RED performed better compared to PCQ and FIFO when several users were simultaneously downloading data in the queue.

Okokpujie, Chukwu and Noma-Osaghae (2018) proposed two adaptive TCP models and compared their performance with RED and fixed-parameter Proportional Integral (PI) on a MATLAB platform. From the results, the two adaptive TCP models performed better than the fixed-parameter PI and RED controllers. An Adaptive Queue Management algorithm with Random Dropping (AQMRD) was proposed by Patel and Bhatnagar (2017). The AQMRD incorporates information about the average queue size and its rate of change to threshold level that falls in-between the minimum and maximum thresholds. The AQMRD was able to minimize packets' dropping by managing the difference between minimum and maximum thresholds.

Kamal and Murshed (2005) stated that several researches had shown that RED can improve TCP performance under certain parameter settings and network circumstances, yet the basic RED algorithm is subject to several problems including sensitivity to its control parameters, bandwidth unfairness and low throughput. To overcome some of these problems, some researchers had proposed several variants of RED while others suggested making modifications to it. RED with Reconfigurable Maximum Dropping Probability (RRMDP) which aims at average queue size reduction and delay time scheduling without any effect on the rate of packet dropping and link utilization was recently proposed by Al-Allaf and Jabbar (2019). The simulation was performed in OPNET. The results of the simulation showed that the RRMDP has a more controllable average queue size, which led to lower queuing delay without affecting the link utilization and throughput. In addition, the controllable average queue size keeps the router queue away from buffer overrun, even in the case of severe congestion (Al-Allaf and Jabbar, 2019).

Many studies had been done on the management of network congestion and queues using methods as feedback mechanism for congestion detection and consequent reduction of packets' sending rate, process timing and packets' dropping in order to avoid buffer overflow. Unlike previous works, this study focuses on minimizing packets' average queue size with minimal cost while also ensuring optimal usage of available servers irrespective of the rate at which packets are injected into the network. This is done by creating a tree structure which is responsible for ensuring network stability using insertion, rotation and removal processes to manage available packets as nodes. The process is the same when node(s) is/are removed for transmission to the server. The proposed system is dynamic by ensuring that it is capable of advising management to increase the number of server(s) by one unit each time the average queue size exceeds the recommended value in order to ensure that packets do not over-stay in the network while also ensuring that available server(s) is / are put to optimal use.

The contribution of this study is that it proposes a method of prevention of packets' losses arising from the emergence of queues using Treap data structure to arrange in-coming packets in the form of a tree whenever the rate of packets' arrival is higher than the service capacity of the server. These packets are rotated from time to time based on tree properties of the Treap and are transmitted to the server based on rotations. This is unlike in previous studies where network congestion using methods as reduction of packets' sending rate, process timing and packets' dropping in order to avoid buffer overflow.

METHOD

This section is discussed under the following inter-related sub-sections.

Proposed congestion control model for M/G/k queue network

The proposed model considers a case of N packets traveling through and contending for service in an M/G/k queue network. Figure 1 is the schematic structure of the proposed congestion control model for the M/G/k network.

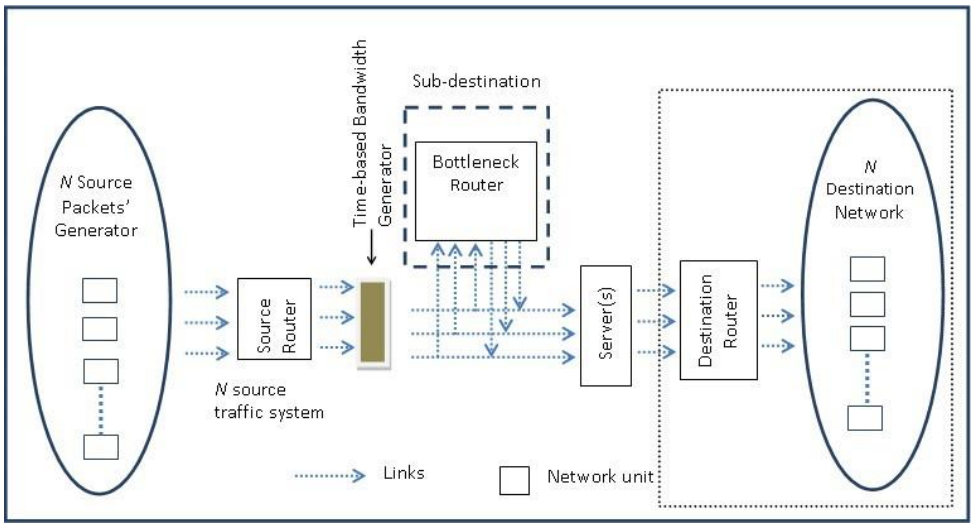


Figure 1: Schematic structure of the proposed congestion control model for the M/G/k network

The proposed model in figure 1 has the following components:

1. N source packets' generator: This generates sequence of random bits of packets of varying sizes and transmits them into the network over available links. Packets generated are assumed to be injected into the network by N users.

2. N source traffic system: This is a source router used to transmit packets to the sub-destination or server, as the case may be. When packets are injected into the network, they are transmitted by the source router to either the sub-destination or server depending on whether or not the packets were injected into the network at a rate higher than the capacity of the server.

3. Time-based bandwidth generator: This generates random numbers typical of bandwidth sizes or capacity over a known range of time $0 \le t \le T$.

4. Sub-destination: When packets are arriving the network at a rate exceeding the processing capacity of the server, they are first transmitted into the sub-destination and are arranged as nodes in a tree. In this case, the packets are made to 'wait' as nodes in a buffer.

5. Server: This processes packets as they are transmitted from the sub- destination or from the source packets' generator. However, when the rate of packets' arrival exceeds the processing

capacity of the server, the packets are first transmitted to the sub-destination where they are arranged as nodes in a tree and are transmitted one after the other to the server in decreasing order of priority.

6. N destination network: This routes processed packets to their respective destinations. The proposed structure is a general-topology network with end-to-end connection of source and destination linked by several routes and nodes. A multiple destination system is assumed for this work.

Queue and tree structure

In the M/G/k network, 'M' is the inter-arrival time distribution i.e. Poison arrival process with intensity (λ), 'G' is the mean service time distribution i.e. $\bar{S} = 1/\mu$, 'k' is the number of server(s). In this network, $\bar{x}$ is the average time a packet spends in service while $\bar{t}$ is the average time between packets' arrivals. Often, the following rate notations are used for these quantities:

$\bar{x} = 1/\mu$, (where μ is the service rate) and $\bar{t} = 1/\lambda$ (where λ is thearrival rate). Combining these two quantities, ρ (utilization factor) can be defined as $\rho = \bar{x}/\bar{t} = \lambda/\mu$ and this is the system efficiency. In general, for the network to be stable, it requires $\rho < 1$, otherwise if $\rho > 1$, then the network is unstable. The Poisson distribution process for packet arrivals in the M/G/k queue network is given as:

$$\alpha(k) = (\lambda^k/k!)e^{-\lambda}$$

where k = arrivals in one time slot and λ = arrival rate per time slot. The general distribution process for packet service time in the network is given as an arbitrary probability density function of $f_y(\gamma)$.

A tree is a widely used abstract data type that simulates a hierarchical structure with a root value and sub-trees of children with a parent node, represented as a set of linked nodes. James, Prakash and Nandakumar (2019) defined a tree as a non-linear and hierarchical data structure consisting of a collection of nodes such that each node of the tree stores a value which is a list of references to the nodes i.e. the children. It consists of a root and zero or more sub-trees i.e. $T_1, T_2, ..., T_n$, such that there is an edge from the root of the tree to the root of each sub-tree. In the design of the proposed model, the structure adopted is a tree with several nodes which represent network packets.

A packet carries important information such as size, time-to-live, source, destination IP address, checksum for error detection, 16-bit identification number, etc. It may also indicate whether or not the packet can be fragmented while also including information about re-assembling fragmented packets, i.e. fragmentation offsets (Dordal, 2021). However, since packets are treated as nodes in the tree, packets' information considered in the design of the proposed model are node size (NS) and node time (NT). In essence, every node in the tree has two parameters i.e. NS and NT which are used as basis for node selection, removal from the tree and consequent transmission to server. The Treap-based structure of N nodes in the tree is represented mathematically as:

$$Tree \quad (T) = \{(a_1, b_1, (a_2, b_2, ..., (a_n, b_n)\}$$

where a is the node size i.e. NS and b is node time i.e. NT.

The NS refers to the size of packet (in bytes), treated as nodes in the tree. The minimum size of an

internet protocol packet is 21 bytes (which includes 20 bytes for the header and 1 byte of data) while the maximum size is 65,535 bytes (Dordal, 2021). However in practice, most packets are about 1500 bytes. For the purpose of this model, the size of packets, treated as nodes does not include the size of the header but rather the size of data only. The NT is the total time a packet had spent in traveling from the source to the sub-destination, in addition to the time it spent 'waiting' in the tree as a node to be transmitted to the server.

The NS of the tree structure arranged max-heap order while the NT are arranged in binary search tree (BST) order. Figure 2 shows the Treap structure of each node in the proposed model.

Figure 2: The Treap structure of each node in the proposed model

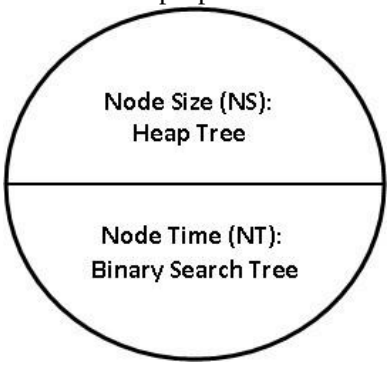


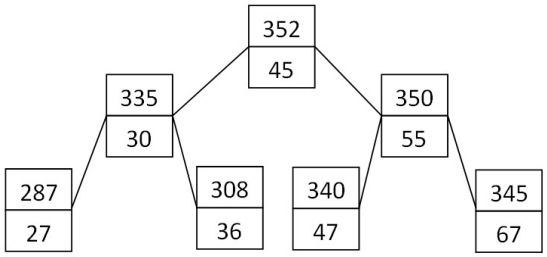
Table 1 shows the first six of an infinite number of packets, treated as nodes in a tree.

Table 1: Sample packets treated as nodes in a tree

Nodes	NS (bytes)	NT (microsecond)
1	352	45
2	335	30
3	350	47
4	287	27
5	308	36
6	340	55
7	345	67
.	.	.
.	.	.
.	.	.

N_n	NS_n	NT_n

These packets are arranged as nodes in the tree given in figure 3.



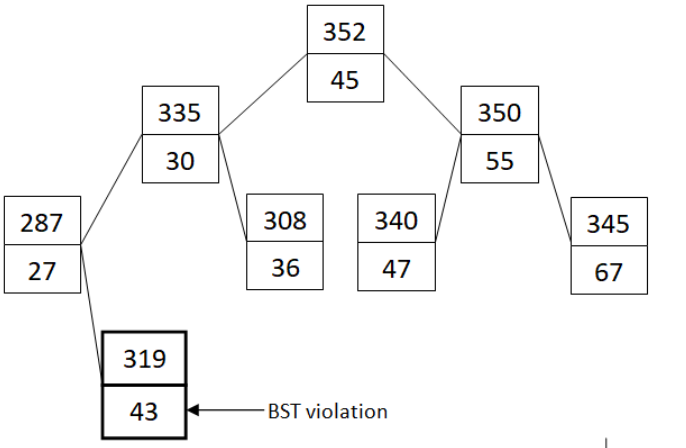
Array	1	2	3	4	5	6	7
NS	352	335	350	287	308	340	345
NT	45	30	55	27	36	47	67

Figure 3: A tree showing NS and NT values arranged as max-heap and BSTs respectively with corresponding array representation

Tree operations of node insertion

When a packet contending for service arrives the network and finds the server busy, it is transmitted to the sub-destination. In this case, the packet is seen as a node by the system. When a minimum of two nodes are in the sub-destination, a tree with one parent node and one child node is formed. As more packets arrive the sub-destination while the server is busy, the tree becomes bigger with parent(s) and child nodes. Consequently, each time a node is added to the tree, it is rotated, if need be, in order to ensure that the tree is balanced.

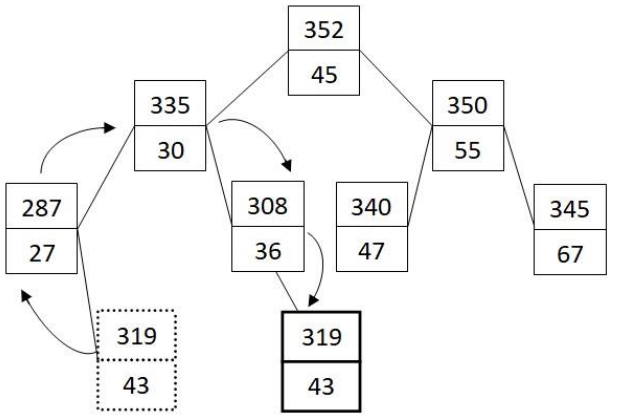
Node insertion is done by first considering the BST property of NT values of all nodes in the tree before consideration of the max-heap order i.e. NS. In this model, insertion is done mainly by attaching the new node to the leftmost node in the BST. The NT value of the new node is compared with that of the parent node. Consequently the new node is placed based on the BST property of the tree. For instance, if node NS 319 : NT 43 arrives the sub-destination to join the tree in figure 3 after 0.*17* ms of the formation of the tree, then it is first attached to the leftmost node i.e. NS 287 : NT 31 as a right leaf node. Preference for placement is first made based on BST property of the NT value. The process is depicted in figure 4.



Array	1	2	3	4	5	6	7	9
NS	352	335	350	287	308	340	345	319
NT	45	30	55	27	36	47	67	43

Figure 4: Tree showing the placement of the new node *NS* 319 : *NT* 43 based on BST property of the immediate parent node only with corresponding array representation

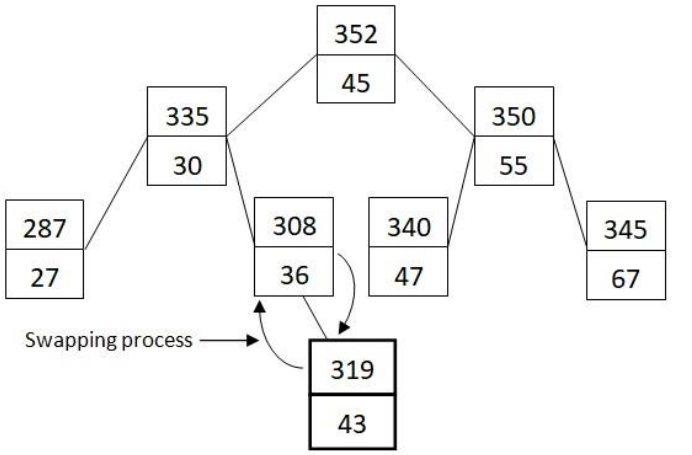
In figure 4 above, the new node is attached to the leftmost node in the tree i.e. *NS* 287 : *NT* 27 as a right child node, thus making the new node to occupy the ninth cell in the array. In this case, the BST property of the NT values is violated because *NT* 43 cannot be the left child of *NT* 30 which is a 'grand parent' node to the newly attached node in the tree. Consequently, the NT value of the new node i.e. *NT* 43 is compared with the NT value of its parent i.e. *NT* 27 and the root node of the left sub-tree i.e. *NT* 30 as well as the NT value of the right child node of the left sub-tree i.e. *NT* 36. The movement of the node results in another tree in which the NT value of the new node based on BST property within the tree structure as depicted in figure 5.



Array	1	2	3	4	5	6	7	11
NS	352	335	350	287	308	340	345	319
NT	45	30	55	27	36	47	67	43

Figure 5: The placement of the NT value of the new node based on BST property of the entire left sub-tree with corresponding array representation

It is observed that the max-heap tree property in the tree structure had been violated as indicated in figure 5. Consequently, the second stage of the placement of the new node based on max-heap tree property is started. Since the node in a heap should be greater than the children nodes, the new node i.e. NS 319 : NT 43 is swapped with the parent node i.e. NS 308 : NT_36. The process is shown in figure 6.

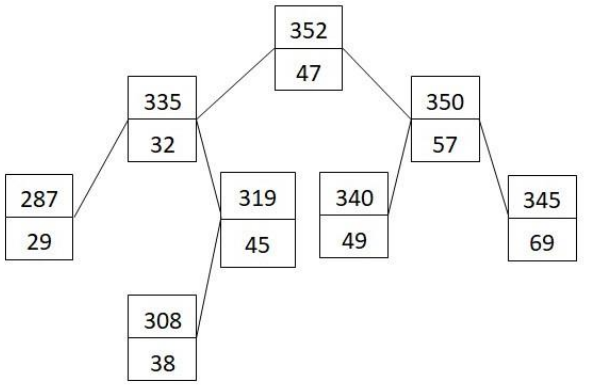


Array	1	2	3	4	5	6	7	11
NS	352	335	350	287	308	340	345	319
NT	45	30	55	27	36	47	67	43

Figure 6: Swapping of new node NS 319 : NT 43 with the parent node NS 308 : NT 38 for max-heap

placement with corresponding array representation

Let us assume that the time it took the new node i.e. *NS* 319 : *NT* 43 a total of 2.15*ms* to join and be properly placed in the tree, then the new tree after the proper placement is as depicted in figure 7 below.



Array	1	2	3	4	5	6	7	10
NS	352	335	350	287	319	340	345	308
NT	47	32	57	29	45	49	69	38

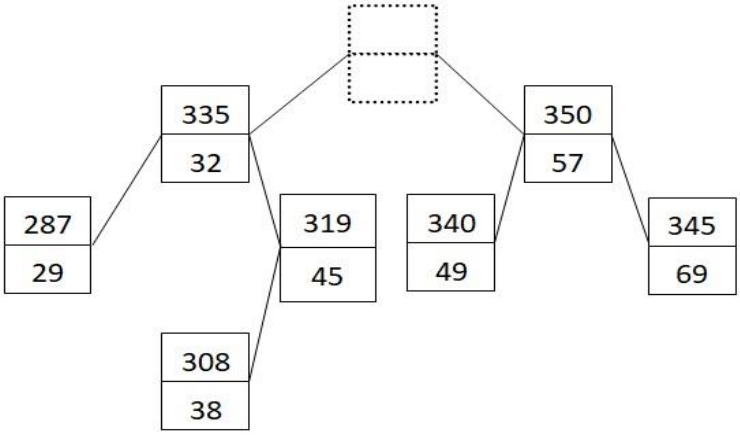
Figure 7: Final placement of the new node i.e. *NS* 319 : *NT* 43 based on both the BST and max-heap tree properties of the tree with corresponding array representation

Node removal algorithm and transmission to server

In order to transmit a node to a server, it has to be removed from the tree. After this, it is transmitted to the server. In order to remove a node in the proposed model, the root node is removed being the node with the highest NS value. The same applies for the in-order of keys. The following algorithm is adopted for the removal of root node in the max-heap tree of the proposed model.

Procedure: Node Removal (NS : NT, Node : tree)
If tnull → NT ← NS Node Removal (NS, Node) *If* NS < Node → NT *then*
Node Removal (NT, Node → *l*child)
else if NS > Node → NT *then* Node Removal(NT, Node → *rchild*) *else* Root Removal(T)
Procedure Root Removal (Node : Tree)
if Is Leaf Or Null(Node) *then* Node ← tnull
else if T → *lchild* → NS > T → *rchild* → NS *then*
Remove Left (Node→ *rchild*)
else Remove Right (Node → *lchild*).

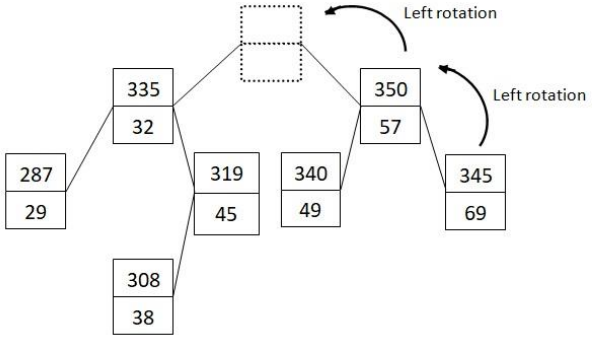
In figure 7, the node with the highest priority is *NS* 352 : *NT* 47. This node is removed for transmission to server. The resulting tree is given in figure 8.



Array	1	2	3	4	5	6	7	10
NS		335	350	287	319	340	345	308
NT		32	57	29	45	49	69	38

Figure 8: The resulting tree after the removal of root node *NS* 352 : *NT* 47 for transmission to server with corresponding array representation

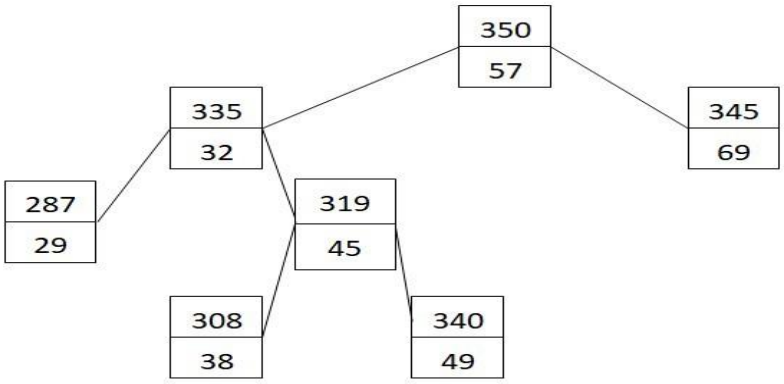
In order to replace the node that was removed, a left-left rotation is done on the node with the highest NS i.e. *NS* 350 : *NT* 57 on order to take the position of the root node. The processes involved are indicated in figure 9.



Array	1	2	3	4	5	6	7	10
NS		335	350	287	319	340	345	308
NT		32	57	29	45	49	69	38

Figure 9: Process of replacement of the root node with node *NS_350; NT_57* using left-left rotation with corresponding array representation

The resulting tree after the left-left rotation of node *NS*_350 : *NT_57 i*s as given in figure 10.



Array	1	2	3	4	5	10	11
NS	350	335	345	287	319	308	340
NT	57	32	69	29	45	38	49

Figure 10: The resulting tree after replacing the root node with *NS* 350 : *NT* 57 using left-left rotation with corresponding array representation

The Treap-based Congestion Control Module

The Treap-based Congestion Control Module (Tb-CCM) is proposed for stabilizing the M/G/k network when packets arrive at a rate higher than the capacity of the server. Initially, the method is not applicable if packets do not arrive at a rate higher than the capacity of the server. In this case, the network is stable and packets are served as they arrive. However, tree development using packets as nodes within the sub-destination to ensure that packets' arrival at a rate higher than the capacity of the server neither destabilize the network nor results in packets' dropping.

The Tb-CCM is given below:

Algorithm: The Treap-based Congestion Control Module Input: s_t, i_t, Q
Output: Tree
for each packet arrival do *{*
calculate the average service time of the processor, s_t;
calculate the inter-arrival time of the packet, i_t;
calculate network queue stability Q;
$Q = s_t/i_t$
if $Q < 1$ *then*
{
serve the packet;
}
else
Tree then
t++;
}
end
return Q;

Dataset

The data set used in the simulation was randomly generated. There were 1,000 files used. Each file

was randomly loaded with different sizes of bytes of data as follows:

1 packet = 1024 bytes;

2 packets = 2048 bytes;

3 packets = 3072 bytes;

4 packets = 4096 bytes;

5 packets = 5120 bytes, etc.

These data sets were loaded and injected into the network through N sources as in figure 1.

Implementation and simulation

Simulations were done using the network topology in figure 1. In the simulations, there were seven traffic sources, seven traffic destination points, one source router, one bottleneck router and one destination router. The link between the source and destination routers is the bottleneck router. The traffic sources are connected to the source router with 10 megabyte per second (mbs) link. The link between the source router and the bottleneck router is 2.5 mbs while the link between the bottleneck router and destination router is 10mbs. The propagation delay between the source router and the bottleneck router is 20ms while the propagation delay from the bottleneck router to the destination router was randomly generated.

Packets of varying size were injected at different simulation time to study the performance of the proposed system under different traffic load. This results in different levels of congestion on the bottleneck link between source and destination routers. The minimum threshold (min_{th}) and maximum threshold (max_{th}) for both methods were set as min_{th} = 50 while max_{th} = 150 (i.e. the max_{th} is triple of the value of min_{th}) as suggested in Floyd, Gummadi and Shenker (2001). The value of max_p was set at 0.1. Both the traffic generating sources and destination have packet size and acknowledgment size of 1448 byte and 40 byte respectively. Packets were injected concurrently through N sources into the network over a period of 45 seconds. While some packets initially by-passed the bottleneck router when the network was stable, others had to pass through the bottleneck router before being served as a result of packets arriving the network at a rate exceeding the capacity of the server.

RESULTS

The performance of the proposed method as well as that of RRMDP is considered under: network throughput, latency and average queue size.

Network throughput

The network throughput increases gradually as the network load in- creases correspondingly. The performance of Tb-CCM is higher compared to RRMDP in this regard. This is shown in figure 11 below

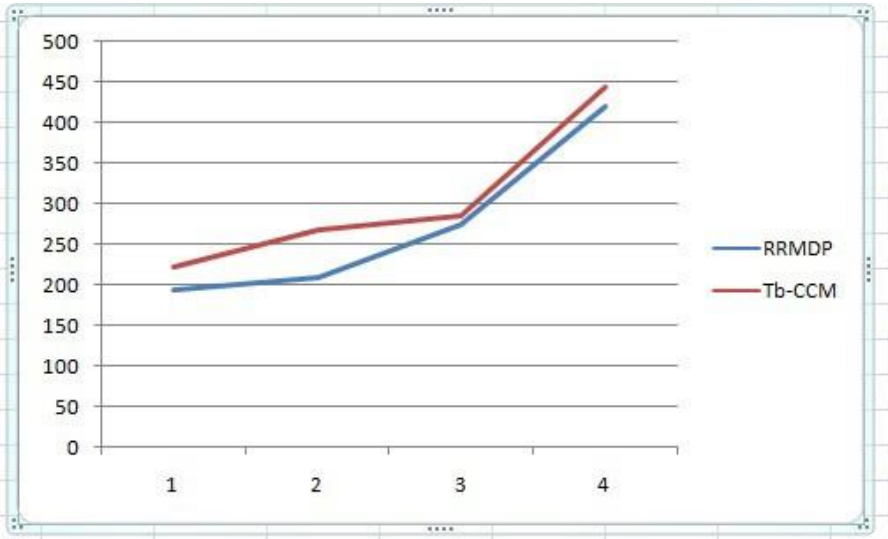


Figure 11: Comparison of network throughput of Tb-CCM and RRMDP

While RRMDP has 195, 210, 275 and 420 kilobytes per second (kbs) for the first four iterations, Tb-CCM has 222, 267, 286 and 445 kbs for the first four iterations respectively. This indicates a significant improvement in the performance of Tb-CCM over RRMDP.

Latency

Latency is the delay in network communication. It shows the time that data takes to transfer across the network. In essence, it is the amount of time taken for a packet of data to travel through multiple devices and then be received at this destination and decoded. The latency of the network using both methods differ significantly. Tb-CCM has a lesser latency compared to RRMDP. This is shown in figure 12 below.

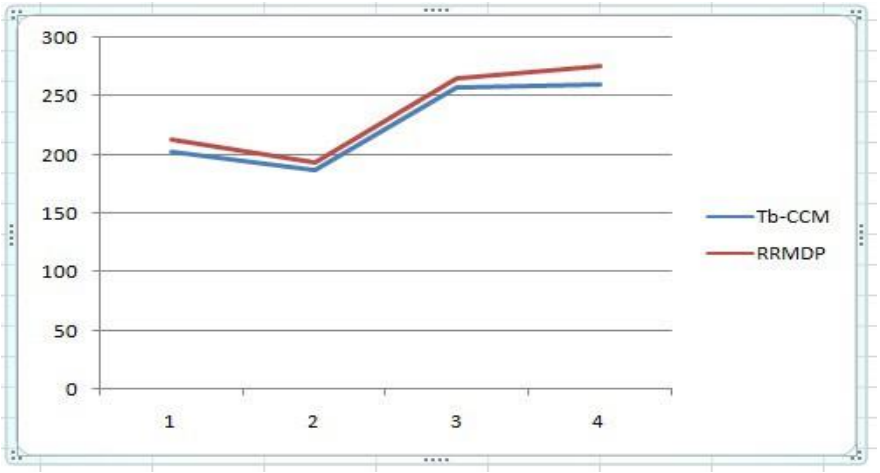


Figure 12: Comparison of latency of Tb-CCM and RRMDP

While RRMDP has 213, 193, 265 and 275 nanoseconds for the first four iterations, Tb-CCM has 202, 187, 257, 260 nanoseconds for the first four iterations respectively. This indicates a significant reduction in latency for Tb-CCM over RRMDP.

Average queue size

Average queue size is the average time spent by packets sitting in a queue waiting to be transmitted onto the link. The amount of time needed to wait depends on the size of the queue. If the queue is empty, then packets transmitted immediately. The average queue size of both methods differ significantly. Simulation results shows that Tb-CCM has a lesser average queue size compared to RRMDP. This is shown in figure 13 below

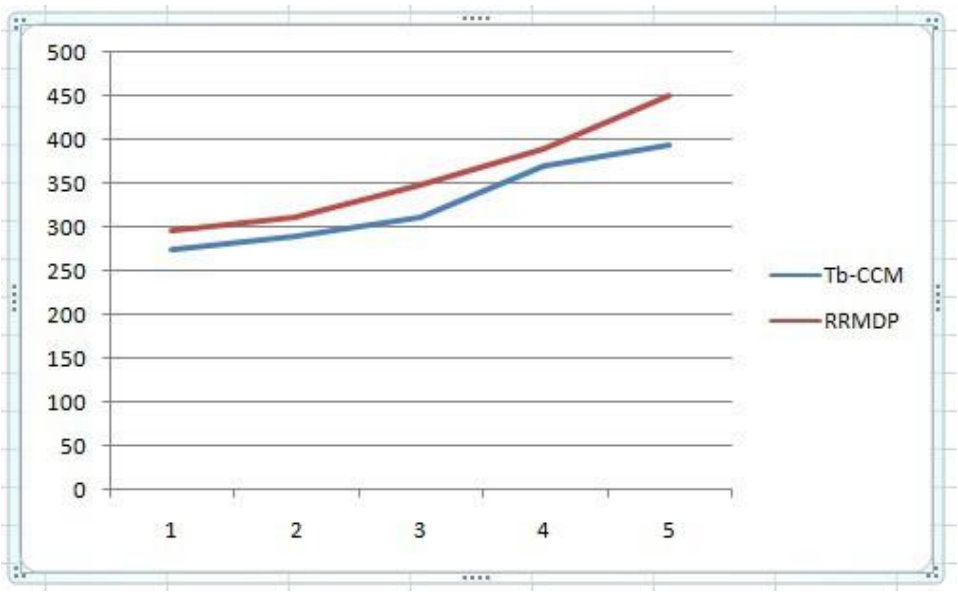


Figure 13: Comparison of average queue size of Tb-CCM and RRMDP in the queue network

While RRMDP has 297, 312, 350, 391 and 451 kbs for the first five iterations, Tb-CCM has 274, 289, 312, 370 and 398 kbs for the first five iterations respectively. This indicates a significant reduction in the average queue size of Tb-CCM over RRMDP.

CONCLUSION

This study proposes a Treap-based Congestion Control Model for M/G/k queue network. The proposed model prevents congestion in the network by using a tree which nodes are treated as packets taking cognisance of their arrival times and sizes. Simulation results show a considerable improvement in network throughput, latency and average queue size of the proposed model in addition to ensuring stability of the queue network. Consequently, it was concluded that the proposed model is more effective in the management of M/G/k queue networks.

REFERENCES

Al-Allaf, A. F. and Jabbar, A. A. (2019). RED with Reconfigurable Maximum Dropping Probability. International Journal of Computing and Digital Systems 8(1). 61-72.

Adesh, N. and Renuka. A. (2019). Avoiding Queue Overflow and Reducing Queuing Delay at eNodeB in LTE Networks Using Congestion Feedback Mechanism. *Computer Communications*. 146(1). 131-143.

Dordal, P. L. 2021. An Introduction to Computer Networks. Department of Computer Science, Loyola University, Chicago. page 537.

Floyd, S. Gummadi, R. and Shenker, S. (2001). Adaptive RED: An Algorithm for increasing the robustness of REDs Active Queue Management. *Technical report, ICSI*. Available at: http://www.icir.org/floyd.

Floyd, S. and Jacobson, V. 1993. Random Early detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking* 1(4). 397-403

Hamdi, M. M., Mahdi, H. F., Abood, M. S., Mohammed, R. Q, Abbas, A. D. and Mohammed, A. H. (2020). Review on Queue Management Algorithms in Large Networks. *2nd International Scientific Conference of Engineering Sciences (ISCES)* pp. 110. doi: 10.1088/1757-899X/1076/1/012034

James, S., Prakash, P. and Nandakumar, R. (2019) The Tree List: Introducing a Data Structure. *International Journal of Recent Technology and Engineering (IJRTE)* 7(6). 1093 - 1095.

Kamal, A. and Murshed, M. (2005) Adaptive RED with Dynamic Threshold Adjustment. *A Research Report, Iowa State University*. 1- 42.

Luo, Y., Zhou, R., Liu, J., Qiu, S. and Cao. Y. (2018). An efficient and self-adapting colour image encryption algorithm based on chaos and interactions among multiple layers. *Multimedia Tools and Applications* 77 (20). 26191-26217.

Miao, W. (2019). Stochastic Performance Analysis of Network Function Virtualization in Future Internet," *IEEE Journal on Selected Areas in Communications.* 37(3). 613- 626.

Okokpujie, K. O., Chukwu, E. C. and Noma-Osaghae, E. (2018). Novel Active Queue Management Scheme for Routers in Wireless Networks. *International Journal on Communications Antenna and Propagation*. 8(1). 53-61.

Patel, S. and Bhatnagar, S. (2017). Adaptive Mean Queue Size and its Rate of Change: Queue Management with Random Dropping. *Telecommunication Systems*. 65(2) pp. 281-295.