Volume 4, Issue 4, April 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Intelligent End-to-End Congestion Control in Wireless Mesh Networks

Tom Jacob Thomas, M. Balasubramani
PSNA College of Engineering and Technology, Dindigul, India

Abstract— Wireless mesh networks have unique characteristics such as the lossy nature of the communication medium, absence of a base station, etc and due to this, maintaining the performance of reliable transport protocols, such as TCP, over wireless mesh networks is a challenging problem. One of the reasons for poor performance of conventional TCP variants over WMN is that the congestion control mechanisms in conventional TCP variants do not explicitly account for these unique characteristics. To address this problem, a novel neural network based congestion control technique is proposed for reliable data transfer over WMN. The proposed technique will be analyzed in detail and is incorporated into TCP to create a variant that we name as intelligent TCP or iTCP. Also, in a wireless communication, any link failure will suspend data delivery completely and thus the overhead of route discovery cannot be neglected. The routing should take place reducing the routing overhead providing a suitable rebroadcast delay for determining the rebroadcast order and thus decreasing the number of retransmissions. So the objective is to control the congestion while providing a more flexible routing to improve the routing performance, while not to violate the basic principles of proposed protocol. The performance of iTCP is evaluated using ns-2 simulations and the results demonstrate that our proposed technique exhibit significant improvement in total network throughput and average residual energy.

Keywords— Wireless mesh networks, transmission control protocol, neural networks, congestion control, upstream coverage ratio

I. INTRODUCTION

A typical Wireless Mesh Networks (WMNs) consists of a set of mesh routers that create a wireless backbone. The backbone offers services to clients, e.g., access to the Internet and other networks via gateways. The network is self-organized and self-configured. The network is comprised of mesh routers (mesh nodes), which are stationary, and mesh clients, which can be mobile. Few of the mesh nodes are connected to the Internet (Internet gateways), while the rest of the nodes rely on multi-hop wireless paths to reach Internet connected nodes. WMNs are becoming an accepted communication paradigm for a variety of diverse applications. The main thing is many of these applications demand reliable data delivery. Several variants of the Transmission Control Protocol (TCP) have been proposed which will ensure reliable data delivery over wired and infrastructure-based wireless networks. However, relatively only a little work has addressed adapting TCP for WMNs. Data transmissions will experience a completely different environment in WMNs compared to wired networks. The wired medium is almost lossless and, therefore, results in mostly deterministic transmissions. Further, nodes in a wide-area wired network, such as the Internet exhibit significant heterogeneity in the data traffic experienced by them. WMNs, on the other hand, are exposed to a truly unpredictable and lossy medium. The lossy environment and non-deterministic nature of WMNs inspires us to explore the use of an artificial intelligence method for congestion control in contrast to the deterministic equation-based method used in existing TCP variants. Based on our work, this paper makes the following contributions: (a) We propose a new congestion control technique using a Neural Network (NN) to regulate reliable data traffic in WMNs, (b) We analyze the proposed NN-based congestion control technique in detail and incorporate it into TCP to create a variant that we name intelligent TCP (iTCP), which specifically considers the unique characteristics of WMNs. (c) We propose a novel scheme that calculate rebroadcast delay for flexible routing.

II. RELATED WORKS

For the reliable data transmission in the Internet, already many TCP variants have been proposed. Congestion avoidance with slow start is used for the control of congestion in internet. But it will aggressively cut the congestion window down to 1 in the presence of a timeout to react to a failed transmission. Fast retransmit and fast recovery is introduced by TCP Reno [9] with the help of duplicate acknowledgements (ACKs). TCP Newreno [4] improves [9] by maintaining fast recovery while acknowledging outstanding data at the time of entering fast recovery. TCP Newreno is modified by [5] with selective acknowledgements. However, none of these techniques consider any of the distinctive properties of wireless links, let alone of WMNs. Snoop [6] was the first attempt to improve TCP over wireless links. However, it requires modification of the network layer at the base station, which is not consistent with the characteristics

of WMNs where there is no base station. Another approach [3] for wireless links also suffers from the involvement of base stations for error recovery. TCP Westwood [1] eliminates the assumption of a base station. However, it assumes a fixed segment size, which may not be feasible in WMNs.

Attempt has been made in [2] to design a TCP variant for WMNs. However, it uses a simple approach depending on explicit notification from the routing layer about congestion, which violates the end-to-end principle and, thus, the thin layer property of the network layer according to the hour-glass model. Also in [10] a non-deterministic TCP variant is proposed but any failures in link will completely suspend the delivery of data and recovery operation may also incur much overhead which may result in congestions in network.

III. MOTIVATION

Some of the distinguished characteristics have to be considered in order to perform the congestion control in WMN. The main one is that WMNs experience failed data transmissions due to various reasons other than congestion, such as propagation loss, multi path effect, hidden station problem, etc. While all these other reasons result in frequent data transmission failures in WMNs, rarely only they result in consecutive failures, which mainly occur due to congestion. Therefore, if we shrink the congestion window in the case of all failures irrespective of whether the failures are isolated or consecutive, then it will unnecessarily slow the transmission rate and also decrease in the bandwidth utilization. Consequently, we should give much importance to consecutive failures than isolated failures while trying to detect the congestion. By considering these characteristics, we design a novel end-to-end congestion control mechanism using a NN, which completely eliminates the conventional notions of slow start, congestion avoidance, etc. Along with that we maximise the utilization of bandwidth by selecting those paths that have maximum unutilized bandwidth. Paths are also selected keeping in mind the power levels of the mesh nodes so that low power nodes will be used only in the case if there are no other alternative paths.

IV. DESIGN OF NN TO CONTROL CONGESTION

Our proposed architecture uses a multi-layer, feed-forward, and zero *bias* NN with reinforcement learning for intelligent end-to-end congestion control in WMNs. We utilize the number of consecutive timeouts, number of duplicate ACKs, and current congestion window (*cwnd*) size as the inputs of the NN to compute the optimal next *cwnd* size at the output. The next *cwnd* size may increase, decrease, or remain unchanged relative to the current *cwnd* size. There would be an optimal extent of update on the current *cwnd* size in case of either increase or decrease. Besides, the next *cwnd* size solely depends on the current inputs, which are completely independent of any intermediate output in the NN. This direction of information dependence justifies the choice of feed-forward NN. Finally, we dynamically obtain two inputs of the NN, namely the number of timeouts and number of duplicate ACKs from the performance of the underlying network in response to the *cwnd* size computed by the NN.

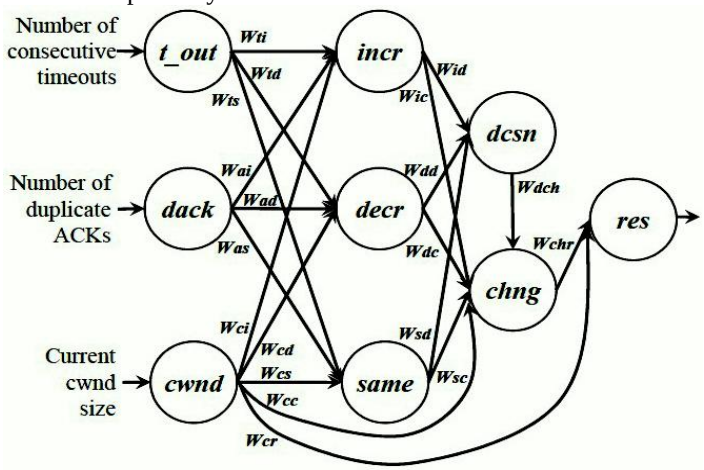


Fig. 1 Multi-layer feed-forward NN for iTCP

Fig. 1 shows the design of the multi-layer, feed-forward NN for iTCP. The NN has one input layer, two hidden layers, and one output layer. The input layer consists of three neurons (*t_out*, *dack*, and *cwnd*) to separately process three different inputs. The first hidden layer determines the relative scale or order of each type of update to be optimal to apply on the current *cwnd* size. We consider three types of update - increase, decrease, and keep the current *cwnd* size unchanged. Three different neurons (*incr*, *decr*, and *same*) separately determine the relative orders of the three types of update. All of these three neurons take inputs from each neuron in the input layer and propagate their outputs to each neuron in the second hidden layer. The tasks of second hidden layer are to determine the type of highest order update and the extent of that update. Therefore, we choose two neurons for this layer to accomplish those two different tasks. One of the neurons (*dcsn*) determines the type of the highest order update based only on the outputs provided by all neurons in the first hidden layer. The second neuron (*chng*) regulates the extent of the update depending on the type of the highest order update, current *cwnd* size, and outputs of all the neurons in the first hidden layer. Finally, the output layer utilizes only one neuron (*res*) to compute the next *cwnd* size based on the extent of update and the current *cwnd* size. The accuracy of this design, to compute the next *cwnd* size, crucially depends on the suitable adjustments of weights assigned to each flow and judicious selection of the activation functions utilized in the neurons.

A. *Adjustment of weights*

We utilize the value of the current *cwnd* size directly to determine the relative orders of different types of update, the extent of the optimal update, and the next *cwnd* size. Therefore, we assign a constant value of 1.0 to all the weights associated to the flows emanating from *cwnd* neuron. Besides, we also directly use the outputs of both *incr* and *decr* neurons to determine the type and extent of the highest order update. Therefore, to all weights that are bound with these neurons we assign the value as 1.0. We assign a value of 1.0 to the weight associated with the flow from *same* to *dcsn* neuron and 0.0 to the weight associated with the flow from *same* to *chng* neuron. Finally, we assign a value of 1.0 to all weights that are bound with those flows emerging from both *dcsn* and *chng* neurons due to their direct impact on the extent of the highest order update and the next *cwnd* size accordingly.

Three neurons - *incr*, *decr*, and *same* determine the relative orders of three types of updates depending on the responses from the network to the current *cwnd* size. We consider the weighted sum of number of consecutive timeouts, number of duplicate ACKs, and current *cwnd* size in all of *incr*, *decr*, and *same* neurons to determine the relative orders of the three types of update. The occurrence of either a timeout or a duplicate ACK should intensify the relative order of the decision to decrease the *cwnd* size and diminish the relative orders of the decisions to increase the current *cwnd* size and to retain the current *cwnd* unchanged. However, the intensity of the impacts on the relative orders are not same. We serialize the decisions in descending order according to the intensities of the impacts as - to decrease, to increase and to retain unchanged. Regardless, impact of each timeout is considered as twice the impact of each duplicate ACK.

B. Choice of activation functions

We carefully choose activation functions in all the neurons to be consistent with the choices of the weights. For all neurons which are present in input layer, the nonnegative weighted sums of inputs have to be directly propagated as the output. Therefore, we use a positive linear function. The most critical part in our design of the NN is the choice of activation functions for the neurons in the hidden layers. We prefer a negative linear activation function for the *decr* neuron and a tailored logarithmic activation function for the *incr* neuron. The reason behind these choices is the fact that timeout and duplicate ACKs may frequently occur due to different factors other than congestion in WMNs. However, if the number of consecutive timeouts or duplicate ACKs grows high, then the possibility of congestion becomes more appealing. The choice of the pair of activation functions for the *decr* and *incr* neurons helps to exploit this fact in WMNs as the growth rate of negative linear function remains constant whereas the growth rate of logarithmic function decreases gradually. Therefore, when we compare the outputs of these activation functions in the *dcsn* neuron, we would get higher preference for increasing *cwnd* size rather decreasing it in the presence of temporary timeouts and duplicate ACKs. However, the preference to decrease becomes higher than to increase with the continuity of timeouts and duplicate ACKs due to the output patterns of the activation functions.

The relative order of the decision to keep the *cwnd* size unchanged mainly depends on the proximity of the weighted sum of inputs to zero in the *dcsn* neuron. Therefore, we choose a symmetric exponential function. The *dcsn* neuron compares its inputs obtained from all the neurons in the first hidden layer to decide on the best type of update on the current *cwnd* size. The *chng* neuron determines the extent of the optimal update on the current *cwnd* size. Its activation function simply generates an output of 0 if the optimal decision is to keep the current *cwnd* size unchanged. Finally, the activation function in *res* neuron is similar to the positive linear function to directly pass the weighted sum of its inputs to its output.

C. Rebroadcast Protocol

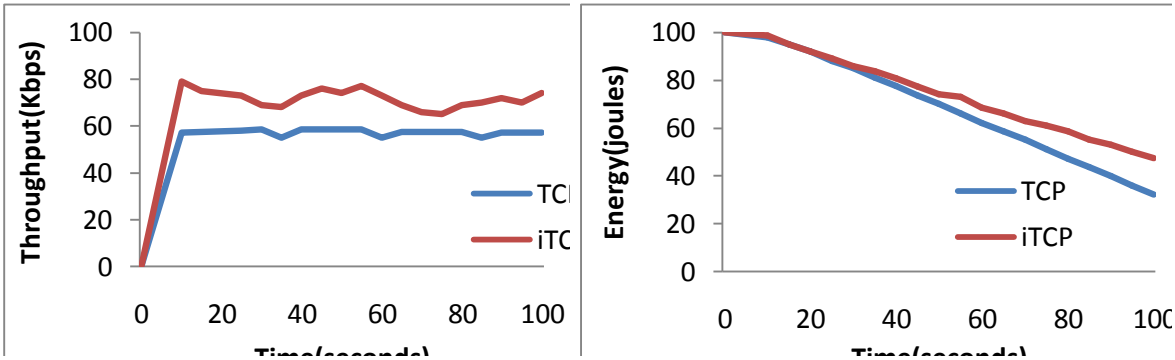
The upstream coverage ratio of the RREQ packet which is received from a previous node will be used to calculate the rebroadcast delay. When a node n_i receives the RREQ packet from *s* which is its previous node, the neighbor list in RREQ packet can be used to estimate the number of its neighbors which are not covered by RREQ packet from *s*. Suppose node n_i is having more number of uncovered neighbors by the RREQ packet from node *s*, it means that if n_i rebroadcasts RREQ packet, this rebroadcasted packet is able to reach more additional neighbor nodes. The node n_i is able to receive duplicate RREQ packets from the neighbors, because of the broadcast characteristics of RREQ packet. With the help of neighbor knowledge, the node n_i may further adjust its uncovered neighbor set. In order to avoid the channel collisions and to sufficiently exploit the knowledge about neighbors, a rebroadcast delay should be set to each node. The key to success is the choice of proper delay because the scheme that determines delay time will affect dissemination of the neighbor coverage knowledge. So, as a neighbor node receives the RREQ packet, it can calculate the rebroadcast delay in accordance with the neighbor list present in RREQ packet and also its own neighbor list.

The delay time is used to determine transmission order of the nodes. In order to exploit the neighbor coverage knowledge, the delay time should be disseminated as possible. When the nodes sends an RREQ packet, all its neighbors, n_i,i=1,2,...,|N(*s*)| will receive and then process the RREQ packet. When the node *n* rebroadcasts RREQ packet, more number of nodes are present to receive it, since node n_k has the largest number of common neighbors. Therefore, more nodes are present that will exploit the neighbor knowledge so as to adjust their UCN sets. The main objective of this rebroadcast delay is not just to rebroadcast the RREQ packet to more number of nodes, but is to disseminate neighbor coverage knowledge as quickly as possible.

V. EXPERIMENTAL EVALUATION

The performance of iTCP is evaluated and is compared with the existing TCP variant through *ns-2* simulations. We measure two evaluation metrics-total throughput and average energy consumption. Each node of the network is equipped with a 2 Mbps 802.11 radio with an omnidirectional antenna. Nodes will have 250m as the transmission range and 550m as their sensing range. The two-ray radio propagation model is used. The interference queue length is chosen as 50 packets in each node.We consider the traffic type as constant bit rate (CBR) data flows. We take the packet size as 1024

bytes at a packet rate of 8 packets per second. The minimum speed is taken as 5 m/s whereas the maximum speed is 8m/s. We collect data from the simulation run of 100 seconds. We carefully set the parameters used in the NN of iTCP. We conducted our simulation in a random topology with an area of *1000m × 1000m.* We place n × n mesh and set as if the sender is on one edge and receiver on the opposite edge and take the number of nodes in network as 25. Fig.2 shows the performance of iTCP over the TCP variant where we set the congestion window constant at 1.



(a) Total Network Throughput (b) Average Energy Residual

Fig. 2 Performance of iTCP in random topology

The total network throughput for the TCP variant and iTCP is compared in Fig. 2a. It is clear that the proposed protocol achieves higher throughput than the other variant in almost all the cases. Similarly Fig. 2b depicts the average energy residual of nodes in the network and indicates that iTCP have more energy residual in its nodes, or in other words, iTCP consumes less energy than the TCP variant.

VI. CONCLUSION

A number of TCP variants have already been proposed for reliable data transmissions over the Internet. They can address the congestion problem in wired and wireless network, but are not so efficient in wireless networks especially in the field of wireless mesh networks (WMN). So the proposed work designs a network which can address the problem of congestion control in WMN using a neural network and adding a rebroadcast delay to decrease the routing overhead. The network can increase total network throughput and decrease average energy consumption of nodes in WMNs compared to conventional TCP variant incurring almost no additional overhead.

ACKNOWLEDGMENT

We would like to express our gratitude to all those who gave us the possibility to complete this paper.

REFERENCES

[1] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang. Tcp westwood: end-to-end congestion control for wired/wireless networks. Wireless Networks, 8(5):467–479, September 2002.

[2] C. Liu, F. Shen, and M.-T. Sun. A unified tcp enhancement for wireless mesh networks. In Proceedings of the Internationall Conference Parallel Processing, pages 71–76, September 2007.

[3] B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan. Improving performance of tcp over wireless networks. In Proceedings of the 17th International Conference on Distributed Computing Systems, page 365, May 27–30 1997.

[4] S. Floyd and T. Henderson. Rfc 2582: The newreno modification to tcp's fast recovery algorithm. April 1999.

[5] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. Rfc 2883: An extension to the selective acknowledgement (sack) option for tcp. July 2000.

[6] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving tcp/ip performance over wireless networks. In Proceedings of the 1st annual international conference on Mobile computing and networking, pages 2–11, November 13–15 1995.

[7] L. L. Peterson and B. S. Davie. Computer networks: a systems approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2007.

[8] P. Sarolahti and A. Kuznetsov. Congestion control in linux tcp. In Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference, pages 49–62, June 10–15 2002.

[9] W. Stevens. Rfc 2001: Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. January 1997.

[10] A. B. M. Alim Al Islam and Vijay Raghunathan School of ECE, Purdue University, West Lafayette. End-to-End Congestion Control in Wireless Mesh Networks using a Neural Network 2011.

[11] X.M. Zhang, E.B. Wang, J.J. Xia, and D.K. Sung, "A Neighbor Coverage-Based Probabilistic Rebroadcast for Reducing Routing Overhead in Mobile Ad Hoc Networks" IEEE Trans. On Mobile Computing,vol. 12, no. 3, pp. 424-433, March 2013.

[12] F. Xue and P.R. Kumar, "The Number of Neighbors Needed for Connectivity of Wireless Networks,"Wireless Networks,vol. 10, no. 2, pp. 169-181, 2004.

