ISSN 2278-3091
Volume 8, No.6, November – December 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse132862019.pdf
https://doi.org/10.30534/ijatcse/2019/132862019

TCP with Machine Learning - Advances and Opportunities

Hardik K. Molia[1,2], Dr. Amit D. Kothari[1]
[1]Gujarat Technological University – Ahmedabad, Gujarat, INDIA
[2]Government Engineering College – Rajkot, Gujarat, INDIA, hardik.molia@gmail.com

ABSTRACT

TCP-Transmission Control Protocol is a connection oriented and reliable transport layer protocol to ensure process to process communication. Improving TCP's performance for wireless networks is always been a challenging task for the researchers. The primary reason of the TCP's performance degradation in wireless networks is its inability to differentiate losses to act accordingly. TCP suffers from inaccurate bandwidth estimation and fairness issue in wireless networks too. Over the years, researchers have proposed many TCP variants to enable TCP to maintain satisfactory performance in wireless environments. In the last decade, ML-Machine Learning techniques are started being applied in every possible area due to their numerous advantages. Many researchers have tried to apply various ML techniques to improve TCP's performance for wireless and wired networks. This paper discusses the fundamentals of ML techniques and their applicability with TCP. Some of the recent ML based TCP variants are discussed. The paper concludes with some future directions for researchers.

Key words : TCP , Wireless Network , Machine Learning , Supervised Learning , Reinforcement Learning

1. INTRODUCTION

TCP-Transmission Control Protocol ensures connection oriented and reliable communication between two processes running at two end devices. Being a connection oriented protocol, it facilitates connection management to allow full duplex and stream communication. Being a reliable protocol, it facilitates congestion control, flow control and error control [1][2][3]. TCP was initially written for wired networks where the primary cause of packet losses is network congestion. TCP's primary focus is also on congestion control where it considers every packet loss as a cause of congestion. When the same TCP was used with wireless networks, a significant amount of performance degradation was found. The reason was TCP's inability to handle various non congestion issues of wireless networks. TCP's congestion control centric architecture was needed revisions to be suitable with the issues related with the wireless networks. These issues include packet loss classification, congestion window update and congestion inference. Packet loss classification is the ability to handle non congestion losses such as channel losses, losses due to route failures etc. TCP's strong consideration of any loss as a cause of network congestion reduces the transmission rate unnecessarily in presence of packet losses due to channel issues or route failures. Congestion window should be set considering the present status of the wireless networks. Congestion detection is also a critical task in wireless networks due to possibilities of other issues [4][5][6].

Over the years researchers have proposed various TCP variants which could be primarily classified into cross-layer approaches and layered approaches. Cross layer TCP variants receive feedback information (network state) from the lower layers for decision making where as layered TCP variants are not dependent on lower layers. TCP variants could be also classified according to how they handle various packet losses. These approaches could be classified based on loss differentiation, loss prediction or loss avoidance approach. A detail discussion of TCP variants is given in [7][8].

In recent years, ML is introduced with the computer networks too. The concept of ML based intelligent network protocol or application design is being applied for traffic analysis, network security and protocol optimization [9][10].

ML - Machine Learning is a group of statistical techniques which let computers learn from the available training data without being explicitly programmed. The goal is to let computers build a decision model themselves which will be keep improving with the experiments. ML focuses on exploration of training data to derive hidden patterns. These patterns are used to label unknown data through the decision model. ML techniques are started being used for many real world applications in the field of bioinformatics, computer vision, speech recognition, data analytic etc. From stock market predictions to the weather forecast, Students' performance predictions to medical diagnosis, Machine Learning is every where due to its numerous advantages such as flexility, accuracy and efficiency. Offline learning such as Supervised learning processes the training data to build the model which is used for decision making for future data. Online learning such as Reinforcement learning learns directly via interaction with the environment. The learning happens on trial basis without any prior model building phase[11][12].

This paper discusses how ML techniques are incorporated for TCP performance improvement. Section 2 and 3 discusses online learning based and offline learning based TCP variants. Section 4 compares these TCP variants. The paper concludes with future directions.

2. TCP WITH ONLINE LEARNING

This section discusses TCP variants with online learning. These variants try to overcome following limitations of rule based or TCP variants with offline learning.

1. Inability to adapt to new or unseen network scenario.
2. Inability to learn from the past experiences.

2.1 Learning TCP

Learning TCP [13] has a learning automaton to set value of $Cwnd$ -Congestion Window dynamically. The automaton interacts with the environment by selecting an action probabilistically. Probabilities are maintained for the selection of actions as per the response received from the environment. The focus is to select a current action based on past actions and responses. Analysis of IAT - Inter Arrival Times of ACKs - Acknowledgements is performed to determine network condition.

A $Time\_Window$ is maintained to record N most recent IATs of ACKs. $Mean$ - Average and $Stddev$ - Standard Deviation of these values are calculated. A value of IAT is bounded in the range $[Mean - Stddev\ ,\ Mean + Stddev]$ where it is rounded to nearest value to minimize effect of very low (or high) IAT. The network state is defined as $Time\_Ratio$ and γ.

$$Time\_Ratio = \frac{Mean - Current\_IAT}{Mean} \quad (1)$$

$$\gamma = \frac{Time\_Ratio - \delta}{1 - \delta} \quad (2)$$

It is assumed that the maximum IAT is thrice of $Mean$. So upper bound of $Time\_Ratio$ is 1 and lower bound of $Time\_Ratio$ is -2. If IAT is small compared to of $Mean$, $Time\_Ratio$ becomes near to 1 indicating early ACK. An ACK for which IAT is higher than of thrice of $Mean$ is considered as a late ACK. $Time\_Ratio$ is -2 for late ACK. γ is normalized network state. Value of δ is set to -2. In absence of heavy network load, early ACKs lead to the value of γ to be closed to 1. Subsequently $Cwnd$ is increased.

The performance index $\beta(n)$, at time step n is set to either γ (Increase state) or to $1 - \gamma$ (Decrease state). The CALA - Continuous Action Learning Automata algorithm uses $\beta(n)$ to update the $\mu(n)$ - Mean and $sigma(n)$ - Deviation of update in $Cwnd$. The detail arithmetic is discussed in [13].

2.2 ML Framework for RTT Estimation

A Machine Learning framework for RTT-Round Trip Time estimation [14] is proposed using experts framework method. Each of the experts guesses a value. RTT is derived using weighted average of these values. Weights are updated based on the difference between estimated RTT and actual RTT. A weight decides the confidence for an expert in terms of its accuracy of prediction. As RTT value is used to calculate RTO value, the accurate RTT estimation indirectly avoids unnecessary pauses(long RTO) or frequent retransmissions (small RTO).

At every trial t, a set of N experts predict their values $x_i \forall i \in \{1, \ldots, N\}$ with weights $w_{t,i}$. The expert's prediction y'_t is compared with actual value y_t using a loss function L to estimate error. $L_{i,t}(x_i, y_t)$ refers to the error at trial t for expert i with value x_i.

$$y'_t = \frac{\sum_{i=1}^{N} w_{t,i} * x_i}{\sum_{i=1}^{N} w_{t,i}} \quad (3)$$

$$L_{i,t}(x_i, y_t) = \begin{cases} (x_i - y_t)^2, & \text{for} x_i \geq y_t \\ 2 * y_t, & \text{for} x_i < y_t \end{cases} \quad (4)$$

The weights are updated with a learning rate $\eta, \eta > 0$. Weights are also shared among experts for the prediction improvement with a weight sharing rate $\alpha, 0 \leq \alpha \leq 1$.

$$w'_{t,i} = w_{t,i} * e^{-\eta * L_{i,t}(x_i, y_t)} \quad (5)$$

$$pool = \sum_{i=1}^{N} \alpha * w'_{t,i} \quad (6)$$

$$w_{t+1,i} = (1 - \alpha) * w'_{t,i} + \frac{1}{N} * pool \quad (7)$$

Values for the three parameters N - Number of experts,η - Learning rate and α - Weight sharing rate are set after experiments with several combinations. These values are set as $N = 100$, $\eta = 2$ and $\alpha = 0.8$.The discussion is given in [14].

2.3 Intelligent TCP

i-TCP - Intelligent TCP [15] is a NN-Neural Network based end to end congestion control solution for adhoc multihop wireless mesh networks. iTCP tries to improve congestion control for non deterministic and lossy wireless environment. A reinforcement learning based multi layer, feed forward and zero bias NN is designed to set $Cwnd$ in the form of increase, decrease or remain unchanged update.

A. Neural Network Design

The NN has one input layer, two hidden layers and one output layer. The input layer has three neurons corresponding to three inputs: t_out -Number of consecutive timeouts, $dack$-Number of duplicate ACKs and current $Cwnd$. The first hidden layer has three neurons recommending strength to change $Cwnd$ in the form of $incr$-Increase, $decr$-Decrease and $same$-No change. The second hidden layer has two neurons: $dcsn$ - determines the type of update with highest relative strength based on the outputs of neurons of first hidden layer. $chng$ -regulates the strongest update with

reference of the output of $dcsn$, output of all neurons of first hidden layer and current $cwnd$. The output layer has a neuron res to set $Cwnd$ based on its current value and output of second hidden layer.

Reinforcement learning based adjustment of weights is performed. Dynamic weights are used for the neurons processing network feedback (Neurons t_out and $dack$). Weights for rest of the neurons are static. A detail discussion of weights adjustment and activation functions is given in [15]. The simulation is performed with NS2 and a real testbed.

2.4 TCP – PCC

PCC - Performance Oriented Congestion Control [16] is a new architecture to understand what rate control actions can improve the performance. PCC sets a transmission rate and observes the results to calculate value for a utility function with an objective like high throughput and low loss rate. PCC is flexible as we can set different objectives by selecting utility function accordingly.

PCC executes a set of micro experiments. A micro experiment shows sending at rate r, produces the utility μ. PCC selects the rate which leads to the highest utility. The time is divided into MI-Monitor Intervals of one or two RTTs. For a MI, PCC sends data at the rate r and observes SACK, throughput, loss rate and latency to calculate value μ of a utility function. One possible utility function is $\mu = T - L$ (T is Throughput and L is Loss rate). PCC has an online learning algorithm similar to of gradient ascent. PCC starts with a rate r and tests rates $(1 + \epsilon) * r$ (higher) and $(1 - \epsilon) * r$ (lower) to select next rate, whichever gives higher utility. It continues till the utility is increasing. Once the utility is decreased, it returns to the state from where it continues testing both higher and lower rates to select best to continue with. The utility function is set with the goal of high throughput and low loss as,

$$U_i(x_i) = T_i * Sigmoid_\alpha(L_i - 0.05) - x_i * L_i \quad (8)$$

x_i, L_i and $T_i = x_i * (1 - L_i)$ are sending rate, loss rate and throughput of i^{th} sender respectively. The sigmoid function is defined as,

$$Sigmoid_\alpha(y) = \frac{1}{1+e^{\alpha * y}} \quad (9)$$

The detail discussion on selection of value of α and appropriateness of utility function is discussed in [16]. PCC's control algorithm has three states: Starting state, Decision making state and Rate adjusting state.

A. Starting State

PCC starts with $2 * MSS/RTT$ and doubles rate every MI-Monitor Interval. Instead of exit from the starting state on a packet loss, it continues till utility decreases. Once the utility is decreased, it returns to the rate which had higher utility and switches to the Decision making state.

B. Decision Making State

PCC executes multiple RCTs - Randomized Controlled Trials to determine direction and amount of change in rate. Four consecutive MIs are divided into two pairs, each of having two MIs. For each pair, PCC tries higher rate $(1 + \epsilon) * r$ and lower rate $(1 - \epsilon) * r$ to get utility values U^+ and U^- respectively. For each pair $i \in 1,2$, utility values U_i^+ and U_i^- are compared.

1. If $U_1^+ > U_1^-$ and $U_2^+ > U_2^-$ then $r_{new} = (1 + \epsilon) * r$.
2. If $U_1^+ < U_1^-$ and $U_2^+ < U_2^-$ then $r_{new} = (1 - \epsilon) * r$.
3. If $U_1^+ > U_1^-$ and $U_2^+ < U_2^-$ then no change in r.
4. If $U_1^+ < U_1^-$ and $U_2^+ > U_2^-$ then no change in r.

On no change of rate event, PCC continues in decision making state with increase in experiment granularity ϵ. $\epsilon = \epsilon + \epsilon_{min}$. The initial value of ϵ is 0.01. The minimum ϵ_{min} is 0.01 and ϵ_{max} is 0.05.

C. Rate Adjusting State

The n^{th} rate can be set as,

$$r_n = r_{n-1} * (1 + n * \epsilon_{min} * dir) \quad (10)$$

$dir = \pm$ is the moving direction. If utility is decreased ($U_r < U_{r-1}$), rate is set to r_{n-1} and PCC switches to Decision making state.

Performance of PCC has been evaluated with 8 real world challenging network scenarios such as Inter data center environment, satellite links, unreliable lossy links etc. The detail discussion is given in [16].

2.5 Q-Learning TCP

Q-Learning TCP [17] deals with TCP's fairness issue of favoritism of flows with small number of hops as compared to of flows with large number of hops. It is a cross layer, reinforcement learning based distributed network monitoring solution for fair resource allocation for wireless mesh networks and wireless multi hop networks. Each TCP sender represents the network as a MDP-Markov Decision Process and applies Q-Learning to maintain transition probabilities.Q-learning is a reinforcement learning algorithm where an agent learns via interaction with the environment. In this approach, TCP sender is the agent and network is the environment. The interaction helps the agent to build states-actions mapping called policies. The time is divided into decision epochs. In every epoch, TCP sender receives network statistics in the form of state space variables.

A. States

The state space is in the form of Fairness Index and Aggressiveness Index. TCP sender uses Jain's fairness index. The Jain's fairness index at node k at decision ephoch t is,

$$J_k^t(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{i=n} x_i)^2}{n * \sum_{i=1}^{i=n} x_i^2} \quad (11)$$

Where n is the number of flows originated or forwarded from node k and x_i is data rate of flow i. The fairness index is a continuous number between 0 and 1. The index as 0 and 1 represents worst and best conditions respectively. To represent fairness index in the form of discrete state space, the [0,1] interval is divided into p sub intervals $[0,f_1],(f_1,f_2],...,(f_{p-1},1]$.

The Aggressiveness Index of i^{th} TCP sender is measured as,

$$G(i) = \frac{\text{No.of packets originated from node i}}{\text{No.of packets forwarded from node i}} \quad (12)$$

To represent aggressiveness index in the form of discrete state space, the [0,1] interval is divided into q sub intervals $[0,g_1],(g_1,g_2],...,(g_{q-1},1]$.

Selection of values for p and q is a critical task as choosing small values shrink the state space and limit the convergence rate. Through extensive simulations and experiments these values are set as, $3 \leq p,q \leq 4$. Thus the state space is of size $p * q$ as,

$$S = \{(f_t, g_t) | f_t \in \{0, f_1, \dots f_p\} \text{and} g_t \in \{0, g_1, \dots g_p q\}\} \quad (13)$$

B. Actions

Q-learning TCP sets the $Cwnd_{Max}$ - Maximum value for Congestion Window to achieve fairness without interfering in the congestion control algorithm. The action set is of Increase, Decrease, Stay operations. The increase and decrease factor is set to 50% of the current $Cwnd_{Max}$.

The detail of integration of Q-learning in TCP and reward function is given in [17]. The approach is evaluated with large number of simulations and real test bed experiments.

2.6 Xavier TCP

Xavier TCP [18] is a reinforcement learning based adaptable congestion control solution. The congestion control problem is described as a MDP and Q-learning technique is used to form the policies.

A. States

The state space has two variables. $EWMA\_RTT$ - Exponential Weighted Moving Average of RTT is a proxy for delay. RTT_RATIO is a proxy for congestion. Both of these variables are updated on arrival of an ACK.

$$EWMA\_RTT = (0.8 * EWMA\_RTT) + (0.2 * RTT_{Current}) \quad (14)$$

$$RTT\_RATIO = RTT_{Current} / RTT_{Smallest} \quad (15)$$

B. Actions

Xavier selects an action out of four actions on arrival of a non duplicate ACK.

1.Exponential Growth: $Cwnd = Cwnd + 1$
2.Linear Growth: $Cwnd = Cwnd + \frac{1}{Cwnd}$
3.Linear Decrease: $Cwnd = Cwnd - \frac{1}{Cwnd}$
4.No Change: $Cwnd = Cwnd$

The reward function is set with reference of the change in EWMA_RTT over time. The detail discussion is given in [18]. The simulation has been done with NS2.

2.7 Learning based and Data driven TCP

Learning based and Data driven TCP [19] is a Q-learning based solution to set $Cwnd$. It is shown that how function approximation can reduce the memory requirement of a learning protocol without compromising with the performance.

A. States

The state space has four variables. Each of these variables is discretized into 10 intervals.

1. Moving average of inter arrival times between new ACKs.
2. Moving average of inter arrival times between sent packets.
3. Ratio between current RTT and best RTT so far.
4. Slowstart Threshold.

B. Actions

There are five possible actions to define change in $Cwnd$ in terms of number of bytes.

1.Reduce by -1.
2.No change.
3.Increase by 5.
4.Increase by 10.
5.Increase by 15.

A state-action space with continuous variables requires a large memory to store Q-table. This problem can be reduced by discretization of continuous variables. Further for the reduction of memory requirement, function approximation technique is used to store approximation of the Q-table which requires less memory without compromising with the performance. Three approaches: TCP learning without function approximation, TCP learning with function approximation (CMAC and Fuzzy) are proposed for memory constraint IoT based application. The detail algorithm and selection of function approximation technique are discussed in [19]. The simulation has been performed with NS3.

2.8 TCP G-Vegas

TCP G-Vegas[20] enhances TCP Vegas[21] with grey prediction. Inaccurate bandwidth estimation due to node mobility leads to performance degradation of TCP Vegas in wireless multihop adhoc networks with mobility. G-Vegas has two parts: Congestion control is enhanced with expected throughput, grey prediction and residual modification model. Adaptive $Cwnd$ is achieved with quantification and

reinforcement learning model. The detail arithmetic is given in [20]. G-Vegas is implemented with NS2 and evaluated for various scenarios such as chain topology, reference point group mobility model and wireless mobile adhoc networks.

2.9 Neural Network Based Reliable Transport Layer Protocol

A neural network based reliable transport layer protocol for MANET [22] is proposed to recognize and capture mobility patterns of nodes to differentiate packet losses. This solution tries to avoid unnecessary timeouts and $Cwnd$ reduction in presence of link failures due to mobility.

Each node maintains a M_Win - Mobility Window to capture mobility pattern. M_Win works as a left shift register and gets updated every Hello interval. 1 is inserted if node's entry is present in neighbor table otherwise 0 is inserted. A node with high mobility has more number of 1's in its M_Win leading to the W_Win - Average Weighted Mean of M_Win closer to 1. W_Win of node n_i at time t is calculated as,

$$W\_Win_{(n_i,t)} = \alpha * M\_Win_{n_i} + (1-\alpha) * W\_Win_{(n_i,t-1)} \quad (16)$$

A. Neural Network Design

A single layer, feed forward, biased and reinforcement learning based neural network is designed to compute $ECwnd$ - Estimated Congestion Window. $ECwnd$ is in the form of increase, decrease or no change action with reference to the current $Cwnd$. The inputs to the neural network are,

1.$Error\_Rate$ - Rate of error due to congestion or link failure.
2.C_M_Win - Cumulative moving average of M_Win from the path from source to destination.
3.Current value of $Cwnd$.

C_M_Win closer to 0 and 1 represent packet loss due to congestion and link failure respectively. The input layer has three neurons corresponding to three inputs: $Error\_Rate$, C_M_Win and $Cwnd$. The hidden layer has three neurons to define strength of recommendation for each of the three updates: Increase, Decrease and no change. The output layer has one neuron to define $ECwnd$. The detail discussion of weights adjustment and activation functions is given in [22]. Simulation has been performed with Qualnet and neural network is validated with MATLAB.

2.10 RL-TCP

RL-TCP [23] is a reinforcement learning based TCP for wired networks with dynamic environment. RL-TCP has three components: sensing engine, learner and actuator. The sensing engine receives ACKs.

A. States

The inputs to the learner are,
1. EWMA of ACK inter arrival time.
2. EWMA of packet inter sending time.
3. Ratio of current RTT and minimum RTT.
4. Slowstart Threshold.
5. Current value of $Cwnd$.

B. Actions

The actuator has an action space of four actions to update value of $Cwnd$. $Cwnd = Cwnd + x$.

1. Decrease by -1.
2. No change.
3. Increase by +1.
4. Increase by +3.

The detail discussion on SARSA based Q-function design and evaluation is given in [23]. The simulation has been performed with NS2.

2.11 Q-TCP

Q-TCP [24] - Q-Learning based TCP lets the TCP sender learns the optimal congestion control using reinforcement learning. Q-learning is used to generalize TCP under wide range of network scenarios. Kanerva coding function approximation is used for the reduction of complexity and state space size.

A. States

Q-TCP's state space has three variables.
1. avg_send - Average interval between sending two packets.
2. avg_ack - Average interval between receiving two consecutive ACKs.
3. avg_rtt - Average RTT.

The logic is that in absence of congestion, avg_send and avg_ack should be same. If $avg\_send < avg\_ack$, possibility of congestion can be considered.

B. Actions

The action space has three actions to update $Cwnd$ in terms of number of bytes.
1.Decrease by -1.
2.No change.
3.Increase by 10.

The detail discussion on using Q-learning, reward function and kanerva coding based function approximation is given in [24]. The simulation has been done with NS3.

3 TCP WITH OFFLINE LEARNING

This section discusses TCP variants with offline learning.

3.1 Bayesian Packet Loss Detection

Bayesian Packet Loss Detection for TCP [25] differentiates congestion loss and packet reordering event. Packet loss detection through time out is time consuming. Fast

retransmission is suitable to detect small number of lost packets. Packet reordering event may trigger fast retransmission unnecessarily. It is necessary to detect an event and infer the reason behind it. Two probabilities $P(y|\theta = loss)$ and $P(y|\theta = reorder)$ are defined corresponding to lost packet and reordered packet events where y refers to RTT- Round Trip Time value. Bayesian framework based inference mechanism analyzes distribution of RTT values to identify cause of DACK-Duplicate ACKs. The bayes detector find probability of packet loss P_L. Probability of packet reorder $P_{L'}$ will be $1 - P_L$. Traces of TCP connections provided by NLANR -National Laboratory for Advanced Network Research and collected at Boston University are used as dataset. The detail arithmetic of probability distribution functions for bayes detector is given in [25].

3.2 TCP with Packet Loss Classifier

A Packet Loss Classifier, TCP + Classifier [26] differentiates congestion loss and loss due to a link error on arrival of three DACKs. Congestion control is not activated if cause of a loss is classified as link errors. A decision tree based classifier processes the information of packets which caused three DACKs event and packets precede it. Average, minimum, maximum and standard deviation of one way delay and inter arrival time are maintained for most recent two RTTs. The input parameters are:

1. L^d, L^1, L^2 refer to the list of 3 packets which generated DACKs, packets received in previous RTT and packets received in previous of previous RTT respectively.
2. BL and BL_{qd} refers to packet received before loss and its queuing delay respectively.
3. L^x_{ip} and L^x_{qd} refers to inter arrival times and queuing delays of packets L^x.

A function Fct is defined as,

$$Fct(X, L) = \frac{Average(L) - Average(X)}{Standard\ Deviation(L)} \quad (17)$$

The decision tree is a set of tests to classify a loss. A few conditions are discussed here.

If $Maximum(L^d_{ip})/Average(L^1_{ip}) < 1$, loss is more likely to be due to congestion. This condition describes that the packets which causes DACKs (after loss) are received in more confined way as compared to of packets prior to a loss. Further conditions are checked to conclude cause of a loss.

If $BL_{qd}/Minimum(L^d_{qd}) < 1$, loss is more likely to be due to link error. This condition describes a situation where a packet (after loss) has higher queuing delay as compared to of a lost packet's queuing delay.

Fct based rules are used to find how far the lists of packets are from each other in terms of inter arrival times. For example, if the average of the inter arrival times of packets after a loss is very high as compared to of packets preceded a loss, congestion loss is considered.

The dataset is composed of losses collected by simulation of large number of random networks with NS2. The detail information of dataset and classifier is given in [26].

3.3 TCP ex Machina

TCP ex Machina [27] introduces the concept of computer generated congestion control with its program named as Remy. This solution discovers congestion control rules from the prior network assumptions, traffic model and objective. The objective could be to achieve high throughput and low queuing delay. Remy takes these models as input to generate most appropriate congestion control algorithm Remy-CC (for TCP sender) which will maximize the total expected value of the objective function. As this process is being done prior to the implementation with an actual network, it is considered as an offline optimization solution. The prior network assumptions also called design range includes information of speeds of bottleneck links, prorogation delays of paths, queue sizes, degree of multiplexing etc. Traffic model specifies the load as a stochastic process. Alpha fairness metric evaluates the throughput on shared links.

RemyCC Memory records sender state information with three variables which are updated every ACK. These variables are used as congestion signals.

1. ack_ewma is EWMA - Exponentially Weighted Moving Average of inter arrival times of ACKs.
2. $send\_ewma$ is EWMA of time between TCP sender timestamps reflected in ACKs.
3. rtt_ratio is the ratio of current RTT and minimum RTT seen so far.

RemyCC defines the mapping between state and action with a look up table. On ACK, memory is updated and a suitable action is taken. The action has three components.

1. A multiple m to the current $Cwnd$. ($m \geq 0$).
2. An increment b to the current $Cwnd$. b can be negative.
3. A lower bound r to set time interval between successive sends. ($r > 0$).

RemyCC defines a set of rules to map three dimensional memory with three dimensional action. The structure of rules is:

$$(ack\_ewma, send\_ewma, rtt\_ratio) \Rightarrow (m, b, r) \quad (18)$$

Remy uses a large number of randomly generated network configurations for evaluation. At the end of the simulation, objective function of every sender is summed to determine overall merit of goodness for a RemyCC. The detail of Remy's design procedure and simulation with NS2 is given in [27].

3.4 LP-TCP

LP-TCP [23] Loss Predictor based TCP CC predicts the probability (loss probability) of how likely a packet will be

lost if sent. The loss predictor is designed with random forests technique to derive loss probability. A network state is updated every ACK with following parameters.

1.$Cwnd$ value and current packet in $Cwnd$.
2.EWMA, TS-Time Series and Minimum of ACK inter arrival times.
3.EWMA, TS and Minimum of packet inter sending times.
4.TS and minimum of RTT.
5.TS of ratios of ACK inter arrival times.
6.TS of ratios of packet inter sending times.
7.TS of ratios of RTTs.

Time series of a parameter includes 8 most recent samples. If the loss probability is higher than of a decision threshold th, a packet send is postponed. To derive value of th accurately, throughput delay tradeoff metric M_e is defined. tp is throughput and d is delay set to $RTT - RTT_{Min}$.

$$M_e = log(E(tp)) - 0.1 * log(E(d)) \quad (19)$$

Threshold th is selected which maximizes value of M_e. The implementation is done with NS2. The detail of threshold and experimental setup is given in [23].

4. COMPARISION

The performance of a TCP variant depends on the type of network it is being implemented for. As TCP's inherent architecture is more suitable for the wired networks, it is required to analyse appropriateness and performance of a TCP variant in wireless environment. Further to the traditional wireless networks (infrastructure based), wireless adhoc networks such as MANETs (infrastructure less with mobility of nodes) add more challenges. This section discusses how suitable various ML based TCP variants are to be deployed for MANETs. Table 1 discusses TCP variants with online learning. Table 2 discusses TCP variants with offline learning.

Applying ML-Machine Learning with a network is always challenging. Selection of a ML type needs to be done with the consideration of its appropriateness and scope of learning in our application. It is also challenging to select the most suitable ML technique of a selected ML type. This needs to be done based on the purpose of learning and resource constraints of our application.

5. FUTURE DIRECTIONS

ML-Machine Learning based network protocol design is still in its initial phase of implementation. Most of the proposed solutions are not widely implemented in real world networks too. The most of the ML based TCP variants focus on setting value of transmission rate (Cwnd). TCP's loss differentiation limitation could be solved by using ML. The intra flow contention issue could be solved by ML based ACK thinning schemes. TCP's security and QoS based parameters could also be set using ML. The present challenge is to select a ML algorithm which is best suited for TCP. At a network protocol level, either ML could be used to set values of various parameters or to design a protocol itself. There is still a lot to research in the direction of automatic protocol design to produce a generalized and adaptable solution for dynamic networks. A systematic analysis of performance of TCP with various ML algorithms may help researchers for intelligent TCP design.

REFERENCES

1. Behrouz Forouzan. **Tcp/ip protocol suite**. *McGraw-Hill*, 2009.
2. W. Richard Stevens. **Tcp/ip illustrated, vol. 1: The protocols.** *Addison-Wesley Professional Computing Series*, 2000.
3. B. Qureshi, M. Othman, and N. A. W. Hamid. **Progress in various tcp variants.** *2nd IEEE International Conference on Computer, Control and Communication*, pages 1–6, 2009.
4. Noor Mast and Thomas J Owens. **A survey of performance enhancement of transmission control protocol (tcp) in wireless adhoc networks.** *EURASIP Journal on Wireless Communications and Networking-Springer*, 2011.
5. Ammar Mohammed Al-Jubari, Mohamed Othman,Borhanuddin Mohd Ali, and Nor Asilah Wati Abdul Hamid. **Tcp performance in multi-hop wireless ad hoc networks challenges and solution.** *EURASIP Journal on Wireless Communications and Networking-Springer*, 2011. https://doi.org/10.1186/1687-1499-2011-198
6. Vassilis Tsaoussidis and Ibrahim Matta. **Open issues on tcp for mobile computing.** *Journal on Wireless Communications and Mobile Computing*, 2:3–20, 2002.
7. Dimitris Kanellopoulos. **Congestion control for manets:An overview.** *ICT Express - Elsevier*, 2018.
8. Hardik K. Molia and Amit D. Kothari. **Tcp variants for mobile adhoc networks: Challenges and solutions**. *Wireless Personal Communications - Springer*, 100:1791– 1836, June 2018.
9. M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang. **Machine learning for networking: Workflow, advances and opportunities.** *IEEE Network*, 32(2):92–99, 2018.
10. Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe EstradaSolano, and Oscar M. Caicedo. **A comprehensive survey on machine learning for networking: evolution, applications and research opportunities.** *Journal of Internet Services and Applications - Springer*, 2018. https://doi.org/10.1186/s13174-018-0087-2
11. S. Wang, W. Chaovalitwongse, and R. Babuska. **Machine learning algorithms in bipedal robot control.** *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(5):728–743, 2012.
12. Jiawei Han, Micheline Kamber, and Jian Pei. **Data mining: Concepts and techniques.** *Morgan Kaufmann Publishers Inc.*, 2011.

13. B. Venkata Ramana and C. Siva Ram Murthy. **Learning-tcp: A novel learning automata based congestion window updating mechanism for ad hoc wireless networks.** *High Performance Computing HiPC 2005.Lecture Notes in Computer Science-Springer*, 2005.
14. Bruno Astuto Arouche Nunes, Kerry Veenstra, WilliamBallenthin, Stephanie Lukin, and Katia Obraczka. **A machine learning framework for tcp round-trip time estimation.** *EURASIP Journal on Wireless Communications and Networking-Springer*, 2014.
15. A. B. M. Alim Al Islam and Vijay Raghunathan. **itcp: an intelligent tcp with neural network based end-to-end congestion control for ad-hoc multi-hop wireless mesh networks.** *Wireless Networks-Springer*, 21:581–610, 2015.
16. Mo Dong, Qingxi Li, Doron Zarchy, Brighten Godfrey,and Michael Schapira. **Pcc re-architecting congestion control for consistent high performance.** *Computing Research Repository Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation*, abs/1409.7092, 2014.
17. Nasim Arianpoo and Victor C.M. Leung. **How network monitoring and reinforcement learning can improve tcp fairness in wireless multi-hop networks.** *EURASIP Journal on Wireless Communications and Networking- Springer*, 2016.
18. Agrawal Akshay. Xavier: **A reinforcement-learning approach to tcp congestion control.** *Technical Report - Stanford University*, 2016.
19. W. Li, F. Zhou, W. Meleis, and K. Chowdhury. **Learning based and data-driven tcp design for memory constrained iot.** *International Conference on Distributed Computing in Sensor Systems (DCOSS)-IEEE Explore*, pages 199– 205, May 2016.
20. Hong Jiang, Ying Luo, QiuYun Zhang, MingYong Yin, and Chun Wu. **Tcp-gvegas with prediction and adapta- tion in multi-hop ad hoc networks.** *Wireless Networks- Springer*, 23:15351548, July 2017. https://doi.org/10.1007/s11276-016-1242-y
21. Lawrence S. Brakmo, Sean W. OMalley, and Larry L. Peterson. **Tcp vegas: New techniques for congestion detection and avoidance**. *SIGCOMM '94 the conference on Communications architectures, protocols and applications*, pages 24–35, 1994.
22. P. Kumar, S. Tripathi, and P. Pal. **Neural network based reliable transport layer protocol for manet.** *4th International Conference on Recent Advances in Information Technology IIT(ISM), Dhanbad-IEEE Explore*, pages 1– 6, March 2018.
23. Yiming Kong, Hui Zang, and Xiaoli Ma. **Improving tcp congestion control with machine intelligence.** *Proceedings of the 2018 Workshop on Network Meets AI & ML - NetAI'18*, pages 60–66, 2018.
24. W. Li, F. Zhou, K. R. Chowdhury, and W. M. Meleis. **Qtcp: Adaptive congestion control with reinforcement learning.** *IEEE Transactions on Network Science and Engineering*, 2018.
25. N. Fonseca and M. Crovella. **Bayesian packet loss detection for tcp.** *IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, 3:1826–1837, 2005.
26. Ibtissam El Khayat, Pierre Geurts, and Guy Leduc. **Enhancement of tcp over wired/wireless networks with packet loss classifiers inferred by supervised learning.** *Wireless Networks-Springer*, 16:273–290, 2010.
27. Keith Winstein and Hari Balakrishnan.**Tcp ex achina: Computer-generated congestion control.** *Proceedings of the ACM SIGCOMM 2013 Conference*, pages 123–134,2013.

Table 1 Online Learning based TCP Variants

Sr	TCP Variant	Features	Comments
1	Learning TCP [13]	Network state is decided based on the observation of IAT- Inter Arrival Time between two successive TCP ACKs. Performance Index is calculated based the analysis of series of IAT values. Continuous Action Updating Algorithm changes Cwnd probabilistically based on the Performance Index.	This approach is complex for MANETs. Only Inter Arrival Time based analysis is inefficient in MANETs.
2	ML Framework for RTT Estimation [14]	Experts framework using fixed share experts algorithm is introduced for RTT estimation. Weights are updated based on difference between estimated RTT and actual RTT.	RTT estimation is not accurate is MANETs. The relationship of Machine Learning based RTT with RTO can be de- rived.
3	Intelligent TCP [15]	A multi-layer, feed forward and reinforcement learning based Neural Network is introduced to estimate Cwnd. Simple inputs and activation functions are used. RTT is not used as input due to inefficient estimation.	Only number of consecutive RTO and DACKs are used for reinforcement learning. More parameters could be used to improve accuracy. Mobility based input could be introduced to make it suitable for MANETs.
4	TCP - PCC [16] Performance oriented Congestion Control	It is based on trials with different sending rates to find the best rate according to the feedback utility function.	It can be implemented only with TCP variants supporting SACK.

Sr	TCP Variant	Features	Comments
5	Q-Learning TCP [17]	It is reinforcement learning based, cross-layer, distributed solution for TCP fair- ness. Fairness index and Aggressiveness index states are introduced to set Maxi- mum Cwnd size dynamically.	Model could be used to control Cwnd too.To achieve fairness, a node may need to compromise with throughput. Other TCP related issues could be addressed along with this solution.
6	Xavier TCP [18]	Objective is to achieve high throughput and low delay. It is a generalize approach across varying network topologies. It has simple inputs and algorithm based on analysis of RTT values.	Reward function can be designed based on other inputs related with throughput. Non linear function approximation can be used to handle other issues like route failures, channel issues.
7	Learning-based and Data-driven TCP [19]	Q-Learning based TCP is proposed todetermine cwnd based on past network state. Two variations based on Cerebellar Model Articulation Controller and Fuzzy Kanerva based function approximation are proposed to reduce memory requirement in building the exploration space.	This variant is proposed for IoT applications. It may not be suitable directly for MANET due to its complexity.
8	TCP - GVegas [20]	The prediction of future throughput based on grey prediction is used to promote the online control. The optimal exploration method based on Q-Learning and RTT quantizer are applied to search for the more reasonable changing size of congestion window.	It is Specific to TCP Vegas. It is a computationally complex approach.
9	Neural Network based Reliable Transport Layer Protocol [22]	It recognizes and captures the mobility behavior of nodes. The captured mobility behavior is used to identify the cause of packet loss. Overcome the issue of band- width under utilization due to link failure caused by dynamic mobility behavior.	It has no support for identification of packet loss due to channel issues. Mobility behavior analysis could be done in simple way to be suitable for resource restricted MANETs.
10	Rl-TCP [23] Reinforcement Learning based TCP	The input parameters are easy to calculate. It continuously learns and adapts in a dynamic network environment.	This variant is proposed for wired networks. It is required to check its suitability with MANETs.
11	Q-TCP [24]	Function approximation based on Kanerva coding is used to reduce number of states for making Q-learning tractable. Dynamic Generalization Kanerva Coding Algorithm is proposed for performance improvement.	This variant is proposed for wired networks. It may not be suitable directly for MANET due to its complexity.

Table 2 Offline Learning based TCP Variants

Sr	TCP Variant	Features	Comments
1	Bayesian Packet Loss Detection [25]	The limitations of Timeout and Duplicate ACK based loss detection are discussed. RTT based Bayesian network is designed to find cause of a Duplicate ACK: Packet loss Vs Packet Reorder Event.	RTT estimation is not accurate in MANETs. Other parameters should be considered with RTT.
2	TCP with Packet Loss Classifier [26]	Supervised learning based classification (decision tree) model to differentiate a loss into channel loss or congestion loss is pro- posed.	Supervised learning based model may not be suitable for all scenarios of MANETs.
3	TCP exMachina [27]	Analysis of network assumptions and traffic detail in offline mode is performed to build a Computer Generated Congestion Control algorithm. No learning is required at run time. Packet loss and RTT are not directly used for congestion control.	The learning phase is very time consuming. Designed for specific networks. Performance may degrades if network conditions change.
4	LP - TCP [23] Loss Predictor based TCP	The main purpose is to find how likely packet will be lost if sent.	Due to supervised learning, When the topology and parameters of a network change, Loss Predictor needs to be relearned. This variant is proposed for wired networks. It is required to check its suitability with MANETs.