Using Reinforcement Learning Algorithms to Provide More Robust Congestion Control

Yihao Cai
Robotics Engineering
Worcester Polytechnic Institute
ycai5@wpi.edu

***Abstract*—As the internet plays a more crucial role in our daily life, providing a stable connection between users becomes much more important. In contrast to circuit switching (e.g. telephone line); the internet works based on packet switching connection. In packet switching; the proper sending rate of users is very critical. To obtain a reliable connection; we need to govern the congestion window of senders in the Transmission Control Protocol (TCP) layer to avoid the congestion problem. Therefore, controlling the congestion window properly would lead to a better connection. Classic congestion models were mostly based on pre-defined parameters that they don't include the dynamic of the data. In last years, using data-driven models -which define their parameters based on the data- become more and more popular. In this project, we use the Reinforcement Learning (RL) algorithm to deal with Congestion Control (CC) problem. We also benefit from the Mininet emulator to generate data and feed the RL model. At this stage, the results might not beat the classical models (like TCP cubic), but exploring such a new discipline for an existing problem would be interesting and promising.**

I. Introduction

A. Problem Description

As the most successful and advanced innovation in Industry worldwide, Internet plays an essential role for people's daily routine [10]. Different from circuit switching used in telephone network, Internet has provided a connectionless service that utilizes packet switching technology, which allows more types of data stream transferring (e.g., image, video, file, etc.) and guarantees message transmission in a more convenient and flexible way. The Internet Protocol is designed to stitch many different networks together and hide underlying technology from applications via Narrow-Waist or Hourglass model [12].

Based on the Hourglass model of computer network architecture, network layer as well as transport layer get their important role for packet transmission especially in the management of end-to-end connections for upper layer services. The performance of emerging new applications depends heavily on the interactions between the underlying network and the transport layer. There're quite a few intrinsic challenges in the network layers, one of which is Congestion Control (CC). Typically, network congestion arises in cases of traffic overloading when a router is unable to handle data that arrives at it, which would therefore cause router buffer overflow and data loss, decreasing transmission efficiency and the overall network throughput (check figure 1). The actual solution is to reduce the network load by adjusting the frequency for sending out the packets on the sender side, where congestion control could be modeled as an optimization problem. In general, there're two approaches for congestion control implementation: End-to-end (E2E) CC and Network-assisted CC [8]. End-to-end CC needs only senders and receivers getting involved without other indicators from the internal network while Network-assisted CC requires information from intermediaries through network like routers. All these two methods focus on designing a policy to enable sender send packet in such a way that the overall network resources could be leveraged to the fullest while network congestion could also be avoided in the meanwhile, which indeed helps achieve fairness and responsiveness in complex networking system.

There may be quite a few challenges remaining in E2E CC, one of which is how to utilize implicit network signals for designing the CC algorithms. Roughly, there're several types of mature E2E CC approaches on the ground: loss-based (e.g. Tahoe, Reno, BIC-TCP, Cubic, etc), latency-based (e.g. Vegas, Westwood) and link capacity-based (e.g. BBR). A classic example is TCP Reno. As the most widely used protocols in transport layer, TCP gains its advantages over ensuring a reliable transfer as well as its capacity in avoiding network congestion [5]. Through constructing a congestion window (CWND), the sender would be able to control how much data could be send out once a time (sending rate), while the window size is changing adaptively according to the ACK received, which ensures the smoothness of communication. Typically, the size of CWND would increase additively every time ACK has been acknowledged, and will reduce multiplicatively once a packet loss is detected. [4]

B. Limitations of Current Approaches

The conservative approach of CC has witnessed great impact and success in tradition network architecture and systems. However, development of modern technologies may cause the increase of complexity as well as diversity of network transmission scenarios and protocols, which brought potential challenges for protocol design, meaning that whilst tradition CC may work well in a certain network, they cannot guarantee the exact same great performance in diverse scenarios since these rule-based methods are mostly non-heuristics [20]. Additionally, even the varying traffic patterns may affect the performance as well. Therefore, an intelligent CC algorithm

is recommended which is also known as learning-based congestion control approach. Different from conservative CC methods, learning-based schemes leverage network states in real-time to control the sending rate rather the predetermined policies, which in turn provides them the availability to cater to dynamic network environments.

Recently, machine learning (ML) has made breakthroughs in a variety of application areas, such as speech/pattern recognition, computer vision, video games and robot control [15]. ML represents strong advantages in learning from collected data or the environment and building models accordingly, which is useful for producing an adaptive method in terms of tradition network congestion control rules. For example, conventional congestion control policies like TCP Reno only take into consideration monotonous metrics, for example the packet loss, acknowledgement (ACK) and/or Round-trip time (RTT) as indicators of network congestion, in which way the entire decision making process to adjust the sender's congestion window heavily relies on these measurements as well as the predefined rules based on human understanding of the existing network topologies. In addition to it, other network problems could be happening by using loss-based policy such as lower throughput due to longer RTT, slow convergence and Bufferbloat. However, by using machine learning/deep learning approaches, the agent would instead have the ability to memorize data and learn from its past experience. It thus has the capability to adjust the CWND more intelligently through choosing approprate measurements (e.g., combination of packet loss, ACK and RTT, etc.) to achieve a better balance between high network throughput and low latency. As the decision-making policy will be trained through time; it would be responsive to the sudden changes under complex and dynamic network environments [17]. Compared with loss-based methods, the approach of Bottleneck Bandwidth and Round-trip propagation time (BBR) could also mitigate the problems (Bufferbloat) by introducing Bandwidth of bottleneck link (BtlBw) together with Propagation delay along the path (RTprop) as metrics for congestion indicators. However, although BBR has the ability to significantly reduce delay among networks, it will bring smaller throughput in comparison with TCP Cubic under some certain scenarios. ML methods would instead be able to address it by allowing agents to make decisions with a trade-off between multiple elements (e.g. throughput, delay, sending rate, etc.) in a dynamical network environment.

All in all, the rule-based mechanism is more susceptible to many unpredictable factors, resulting in poor performance in a complex network scenario. ML/DL, on the other hand, aims to construct algorithms or models in such a way that agents can learn to make decisions directly from past experience or the information from network environment. It does not need accurate network models. Hence, it has the potential to outperform the rule-based mechanism.

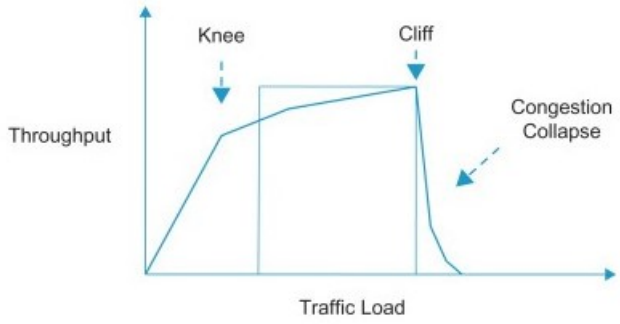

Fig. 1. **Throughput versus load** There are 2 critical points in the congestion figure. **1) knee:** after this point throughput increases very slowly and delay will be increased **1) cliff:** when we reach this point, throughput reduces to zero immediately (congestion collapse) and the delay goes to infinity [16]

C. Challenges

To apply ML approaches for controlling congestion, we need to deal with different challenges. Some of these are intrinsic characteristics about the CC problem, and the others are directly related to solving CC problem with ML/DL. We listed serveral potential challenges we may encounter in our project as below:

1) **Decrement Rule:** What is the best Traffic Rate Decrement rule in the presence of congestion [16]? This is the main concern in the Congestion Control problem. Using different parameters, e.g. initial window, packet loss or delay, or different rules would lead to completely different results. Here, we need to use the collected data and train the ML network to decide about the Decrement rule of the network.
2) **ML Method:** To solve the congestion control problem, a proper ML technique should be found. In recent years different ML techniques including online and offline approaches have been developed to deal with the CC problem [19]. Each of these methods has its own limitations and advantages, meaning that one neural network may perform well under one specific environment while it may even get worse result than rule-based policy under another scenario. Because of this, we need to find a proper ML algorithm that much caters to our situation for the project in a limited time.
3) **Data Collection:** The problem of learning a Neural Network typically has relationship with the amount of data as input. In order to leverage from Deep Neural Network (DNN) we need to generate a proper and sufficient amount of data so that we won't get result such as underfitting or overfitting [15]. To obtain data for CC, we might need to generate synthetic data or use real data, including vehicular, mobile networks, and satellite communication.
4) **Feature Selection:** We need to investigate about proper features that can be used to train ML models. In ML models we need to extract features from data and pass them to the model so as to train it properly. In addition,

we ought to choose the objective functions in which the model tries to find the best accuracy to satisfy our objective. From the literature review, usually throughput, delay, loss rate, and congestion windows are used as the objective functions. We need to find the proper features as well as the objective functions (targets) to train the model. In fact, we need to do some data pre-processing and collect desired features.

5) **Training Efficiency:** Training efficiency is directly determined by deployment of environment. Traditionally, the training process is regarded as a time/resource consuming procedure where state abstraction plays an essential role in training efficiency improvement. Parameter selection would be another approach for optimization. However, tackling this issue requires more research on the way for completing this project.
6) **Providing Test Bed:** At the end, we need to apply the proposed model on different datasets, including simulated (or real data) to check how the model is successful to improve its performance. We also will provide some comparisons between the proposed model and current models to assess the power of our model. However ML techniques acquired lots of attention in CC problem, but still these ML techniques not fully tested and developed for real data sets [19]. We hope that we will be able to run the ML algorithm on real data.

II. Literature Review

As illustrated before, the conservative rule-based policy for congestion control indicates several limitations including dynamic network environment adaptation, implicit signals as indicator of network congestion, intrinsic problems of algorithms whose performance are affected by network topology, etc. Machine Learning-based control rules have been proposed to address these potential issues under an end-to-end CC. Generally speaking, there exist three subsets of ML approaches for dealing with CC – supervised learning, unsupervised learning and reinforcement learning. Literature [8] has a complete and detailed explanation as well as implementation regrading ML-based learning approach with CC, where they use four parameters as performance metrics for network congestion: packet loss, RTT, throughput and fairness. Following the above survey, we will firstly give out a demo to show how tradition rule-based policy detects congestion and summarize the details as well as potential issues about ML/DL-based algorithms under different network scenarios respectively.

A. Conventional Rule-based Policy

One well-known and widely used implementation of rule-based policy may be TCP Reno. Taking packet-loss as indicator of congestion, TCP Reno controls the amount of data flowing into network by adjusting congestion window once it detects losses, where several phases are included – slow start, congestion prevention, packet loss detection as well as fast recovery as shown from figure 3. In slow start, Reno increases its CWND exponentially until a thresh

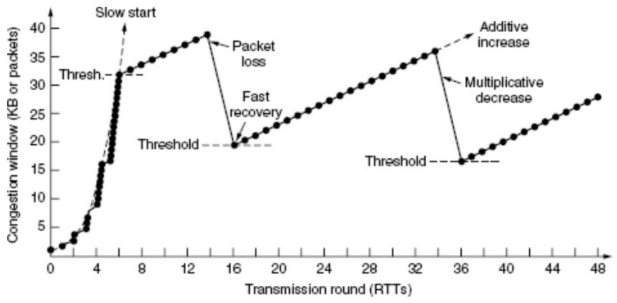

Fig. 2. TCP Reno Algorithm

value has been reached. Then it would additively increase CWND to avoid congestion. Once a packet loss has been detected, Reno would multiplicative decrease CWND (reset threshold) for next transmission, and get into Fast recovery phase which allows it again to increase CWND additively until another loss has been detected. This approach provides a linear methodology for controlling CWND size but would cause severe bufferbloat problem in routers, resulting in high latency through data transmission.

B. Learning-based algorithm

Learning-based algorithms are very useful because they can be adopted to new network situation and they are much more customizable than rule-based algorithms [11]. In general, we have three different learning based algorithms that we will introduce them in below.

1) Supervised Learning: Supervised learning is about building a model between known input and output datasets, and mapping following inputs to outputs correspondingly, which is more commonly regarded as an explicit classifier sorting input data according to their labels attached. Tradition supervised learning algorithm involves decision trees together with neural networks. Its application consist of traffic classification, loss classification and delay predication for gaining better congestion control effect, which are explained separately as below:

A **Traffic Classification:** Traffic policy plays an important role in CC because it decides allocation of network resources, and an optimal traffic classification method can help achieve better network performance as well as fairness. In order to apply supervised learning techniques for traffic, the states of elements must be predefined (e.g. [14] has defined packet-level/flow-level information). In addition, high error rate is also a common issue within such approach and a Naive Bayes Estimator could be used for improving the accuracy [13]. Moreover, a Support Vector Machine (SVM) [9] can be leveraged for reducing the computation cost if the feature (dimension) of input is sufficiently large.
B **Loss Classification:** Packet loss has always been utilized as a signal of network congestion in tradition CC algorithms (e.g. TCP Reno, Cubic). However, as a matter of fact, the cause of loss not only results

from congestion, but may be affected by other factors under different network scenarios (e.g. wireless networks, optical burst switching networks, networks with reordered events, satellite networks). Some uncertain factors may cause losses in wireless networks. They could be erroneous links, mobility, channel conditions together with interference in addition to congestion. To tackle this problem, traditional classifiers such as Biaz, Spike and ZigZag have been proposed for distinguishing the types of losses. While they may work effectively in some specific scenarios, an adaptive policy is still needed for a more complex and dynamic network environment. To apply supervised learning algorithms, delay information is the kernel state where the features of input could be extracted as RTT value, inter-packet times, inter-arrival time, queuing delay, etc. Contention loss usually happens within Optical Burst Switching network (OBS) which saves sources due to wavelength reservation. And contention loss happens because OBS lacks buffers. Supervised learning approach is trying to address it by applying Hidden Markov Model (HMM) [7] to distinguish between contention/congestion losses. When it comes to networks with multi-channel paths, then reordering loss may not be avoided once packets are reordered. One characteristic for reordering loss is the variant RTT value compared with those related to congestion. And a Bayesian algorithm can be used to distinguish between two types of losses with considerable accuracy [3]. Although supervised learning techniques show advantages in loss classification, there are still challenges it faces for real implementation. One is Misclassification where the performance of classifiers mainly depends on predefined parameters, causing poor congestion control to network if parameters are not set properly in the beginning. Another is the balance between computational complexity and prediction accuracy. Sometimes it consumes more network resources but brings limited improvement for classification accuracy, because of which we have to make a trade-off under different network scenarios.

C **Delay Prediction:** Delay of packet transmission can directly reflect the total number of in-flight data (load) through the network. Therefore it is an indispensable factor we have to consider when designing a CC algorithm. For learning-based policy, mechanism of delay prediction has been proposed for senders to react quickly to avoid network congestion in an efficient way. Multiple metrics could be effective on predicting packet delay, one of which is Round Trip Time (RTT) and another is Retransmission Timeout (RTO). Moreover, some approaches could be leveraged on measuring RTT through building the relationship between RTT and sending rate using linear regression [2], Bayesian techniques, etc. In order for Delay Prediction to achieve a great performance in a real-time networks, low computational complexity as well as high responsiveness should be guaranteed during the process.

2) Unsupervised Learning: Different from supervised learning, there's no predefined labels in unsupervised learning and it sorts out input data according to the similarity among samples so that both its intra-class gap of data will be minimized and inter-class gap would be maximized to the fullest extent. This machine learning approach is widely used where there's limited information about the networks. Commonly used clustering methods include K-means, Hierarchical Clustering, Density-based MeanShift, Density-based DBSCAN, and Expectation Maximization (EM). Each of these approaches represents different characteristics under different situations. K-means is an easy-to-implement algorithm where an unlabeled data is sorted to a sole cluster, but its result could be highly affected by initial cluster center selection. EM is more generally utilized for Gaussian Mixture Model and its data point could be mapped into multiple clusters instead of being restricted to only one type as K-means. But the limitation is that its clustering performance depends on the particular applications. Compared with AutoClass [18], K-means and DBSCAN methods converge faster. While K-means and AutoClass shows higher accuracy. However, DBSCAN produces better clusters. For their applications in congestion control, Loss Clustering and Delay Predication are two main research fields.

A **Loss Clustering:** Similar to loss classification in supervised learning, Loss clustering will cluster data collected by sender into congested losses and non-congested losses (e.g. wireless/contention loss) according to extracted features. In wireless networks, loss-delay pairs are used for clustering because wireless losses present different delay distribution than any other types, which could be captured by Hidden Markov Model. In OBS, burst amount is able to distinguish congestion/contention losses.

B **Delay Prediction:** Due to high processing demands for delay calculation in networks, the availability of applying unsupervised learning-based CC algorithms on delay prediction is limited. Some states have been listed for data clustering such as message size, message validity and message type, and by designing specific sending rate for each cluster, the network congestion control could therefore be achieved.

3) Reinforcement Learning: Reinforcement Learning (RL) is considered to be another branch of Artificial Intelligence (AI) together with ML, which learns through interaction with environment and may not have strict restrictions on input data scale. Essential elements in RL include agent (policy), state (environment), and reward. Figure 3 depicts the reinforcement paradigm. Agent plays as the brain of RL network and takes an action. Then, based on taken action and the environment (including all measures of the model, in our case packet loss, delay, ...) we calculate the reward. Now, agent updates its action based on the reward of previous action and the environment.

To have a better understanding of RL, you can assume the example of baby and mother. Baby is learning how to behave

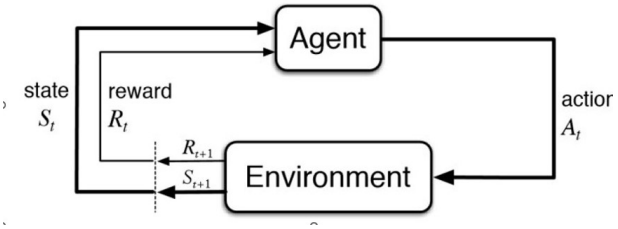

Fig. 3. Reinforcement Learning paradigm

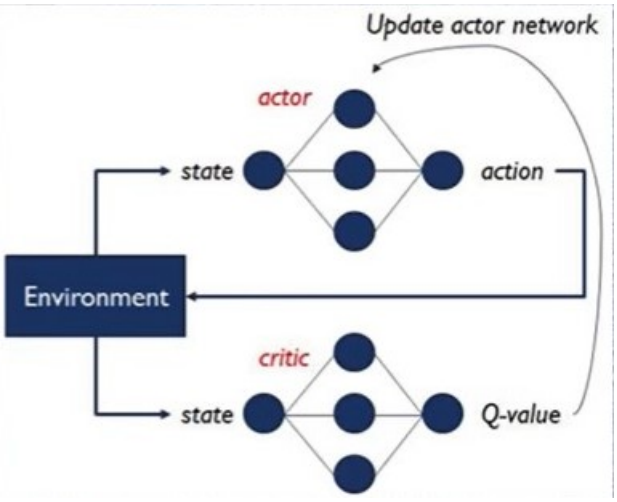

Fig. 4. Actor-critic algorithm

well under the mother's supervision, and mother serve as a mentor during his growth. Baby will be scolded when doing something wrong (penalty) or get a sweet if he behaves well (reward). According to feedback on his behaviors (action), he was able to self-learn how to better interact with his surrounding environment gradually (agent/policy). Typically, there are two schemes for implementing RL algorithm: value-based and policy-based. In policy-based schemes, the agent would try to figure out the best policy applicable for the actions. whilst in value-based schemes, the agent would instead predict action value straightly. In the methodology section, we will use the actor-critic approach; here we introduce this algorithm. This algorithm has been shown in the figure 4. Actor-critic method has 2 main parts: **Actor** use current environment and take the best action based on the current state. **Critic** plays evaluation rule by taking environment state and the action; then return a score that tells us how good is the chosen action. This score is named Q-value and we re-train our actor model based on Q-value score [6].

For its applications through network communication, RL is widely used in scenarios like 5G and edge computing for improving the overall quality-of-service (QoS). Different from traditional approaches used in ML, RL outperforms them in network through monitoring environment status continuously and providing an optimized utility function for interaction the next time, because of which RL algorithms are super adaptable to dynamic environments. So far two trends are popular for research in these areas. One is to ensure the reliability of data transmission under some specific scenarios (e.g. data center, cloud computing, etc.), and another is about designing a flexible network topology for mobile networks. Moreover, most implementations of RL algorithms are achieved by updating Congestion Window (cwnd). As RL is more flexible than other 2 learning based approaches; we utilize this powerful model in our method.

So far we have talked about conventional and learning-based approaches to the CC problem. However, each of these algorithms has their advantage and disadvantages. In the last years, there has been a new endeavor to combine these two schemes and benefit from both of them [1]. In table I some of the advantages and disadvantages of each model have been written [1], [11].

If we combine both conventional and learning-based algorithms, we can benefit from both of the aforementioned models. this model is called a Hybrid model. In the methodology, we introduce a hybrid model, that uses Reinforcement learning in conjunction with conventional TCP algorithms.

III. Methodology

In this section, we introduce a state-of-the-art model for CC named ORCA [1]. The optimal goal of the presented model is to estimate cwnd in a way that we have the best performance. In the congestion problem, we try to maximize the average delivery packets (throughput) while we minimize the delay of the network. In general, these two parameters (throughput and delay) specify the performance of the CC algorithm.

In this paper [1], one lightweight network emulator tool **Mahimahi** has been introduced to conduct the experiment for measuring the network performance using DRL-based approach. It focuses on testing real-life networks where numerous transcontinental AWS/GENI servers should be deployed and configured for cooperation. And it guarantees the effectiveness of RL algorithm to be applicable into real-world practical use. However, most of its network environments are set up upon the client-server model, which is monotonous and leaves potential scalability problem when it comes to other types of network topologies for applications. Furthermore, many configuration issues should be addressed within its original implementation. Thus to provide more convenience and flexibility, we propose a methodology based on another commonly used emulator **Mininet** (more powerful than **Mahimahi**) to test our methodology together with the result. In addition to that, we utilize Vagrant for environment setup to ensure ease-of-use and user-friendliness. That being said, we keep part

TABLE I
Comparison between conventional and learning-based models

	Conventional CC	Learning-based CC
advantage	reliable fast and real-time	capturing the net's dynamic data-based
disadvantage	not adjustable to a new net	convergence issues is not real-time

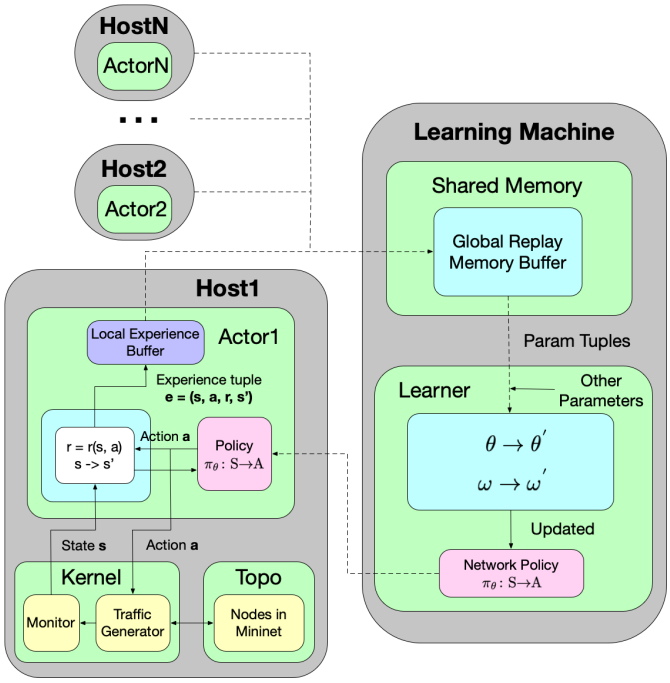

Fig. 5. Model Pipeline Design for RL

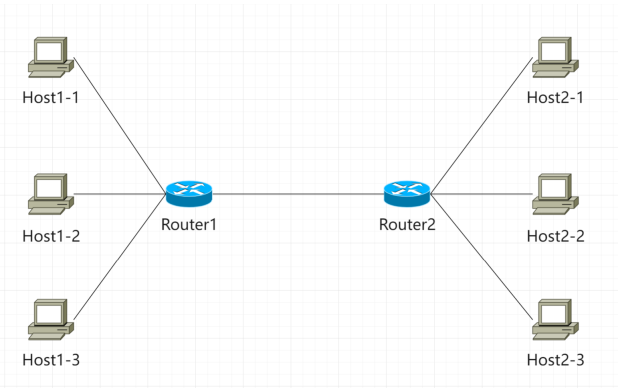

Fig. 6. Dumbbell Topology

of the architecture as origin and provide a flexible Mininet interface capable of measuring DRL-based CC performance on various network topologies. Also, we test the feasibility of our approach by parameter tuning to adapt the model for the input data we generate.

A. Model pipeline

In this section we talk about how DRL updates its parameters. Figure 5 helps us to have a better understanding about what is the DRL method, how it gets information from its environment, and how it connects to the conventional TCP controller. Here we proposed our model pipeline design by separating the learning agent and actor agent apart. Some concepts including Actor, Global Memory, Learning Machine, etc. are introduced as following.

1) **Actor:** Actors are recognized as action generators. They are deployed on hosts one for each, and playing the roles of producing the action which helps maximize its reward function using updated policies from learning model. It obtains the real-time states from TCP monitor and passes the optimized actions to traffic generator for execution where these two modules are running in kernel mode. After an action has been performed, a new network state **s'** will be thereby yielded, and the reward could be calculated through **r = (s, a)** after action execution. So far the actor agent has got an experience tuple **e = (s, a, r, s')** which is supposed to be stored in Local Experience Buffer on its host and would be synchronized on the shared memory on learning machine for training.
2) **Kernel/Topo:** Two modules are running in the kernel mode which includes the TCP monitor and the traffic generator. Moreover in the Topo module, Mininet nodes are created with predefined topology with which we wish to test the effectiveness of RL-based TCP CC algorithms. That being said, the traffic generator executes the action and yields flows interacting with other nodes in Mininet (by adjusting cwnd to send out packets and receive the ACKs), and monitor samples data to produce the observed states in real time. All these modules work cooperatively to control flows in the low-level design.
3) **Learning Machine:** Learning machine is the agent that holds the RL learner for updating the policies globally. Here policy is defined as a mapping from state to action, which exhibits various optimized choices to control the flow under different network scenarios. Learning machine should guarantee a considerable large memory which is used for storing all the experience tuples coming from different hosts as Global Replay Memory Buffer. And learner would fetch and train the parameters to yield optimized policies which they will later send back to all the running hosts. And threads synchronization mechanism should be guaranteed on learning machine to deal with potential issues caused by distribution architecture.

Based on the details explained above, we can conclude that each actor implements RL-based CC algorithm under a specific-designed network. Using this distributed architecture, we could not only test our model performance with different network topologies, but also guarantee the efficiency as well as the effectiveness of training multiple models simultaneously.

B. Network Topology

As we previously mentioned, a client-server model has been used to measure the overall performance of RL-based CC algorithm in [1], and in this project, we try to extend its scalability as well as versatility by designing an architecture that provides an easy-to-use interface based on Mininet for building diverse network topologies. And a network evaluation scheme should be set up at this time through the monitor block. As is widely acknowledged, numerous topologies exist for establishing the real-world network where each exhibits its own advantages/disadvantages. In this project, we only use

TABLE II
CONFIGURATION OF MACHINE USED TO TRAIN THE RL MODEL

Linux version	CPU	GPU	RAM
Ubuntu 20.	core i7, 7700	GeForce 1060	16 Gig

the dumbbell topology to train and test the network. However, we can extend the model by using multi-actors and assigning each actor to one topology. The other important topologies that we might extend our model to them; are introduced in the Appendix section.

Dumbbell Topology: Dumbbell Topo is more utilized in places where two agents try to make connections through world-wide internet. For instance, just imagine that you're outside for a business trip and want to connect your company network in a secure way. Then it is likely that you are supposed to establish VPN (Virtual Private Network) connection using WLAN (Wireless Local Area Network). The gateway server plays the role as NAT during the process. In our architecture, we plan to apply the RL-based model on two gateway hosts (routers) and explore their stability as well as the capability of dealing with high-concurrency problems.

IV. EMPIRICAL RESULTS

The goal of the RL method is to benefit from the presented Mininet emulator to generate enough data and apply synthetic data to the complex RL network to capture the dynamic of data. In the empirical section, at first, we talk about training configuration and the scenario that we used to test the RL method and baseline method. After it, we will talk about the metrics that we used to assess the model. In the end, the empirical results have been presented.

A. Training

Through training; as the training of the RL method is time and resource-consuming (for example in ORCA model it uses 48 CPU cores, in the combination with 256 GB RAM and GPU RTX 2080 TI); we needed to decrease the number of actors of the RL model and make the RL model much simpler than original models. the configuration of the machine that is used for training is shown in table II. For the training session, we only use an actor that can not lead to a robust result (In the ideal case, we need to train the RL model with more actors but it takes more time and needs more resources).

The other imporant aspects in training the RL model is related to chosen input signals. Based on [1] we use statistics in table III as input of the model. In the implementation, we used a third-party toolbox, Goben, to capture these statistics.

To assess the performance of the RL model, we use TCP cubic as the underlying model for comparing the results of our simulation to it. TCP cubic is used as default in Linux and we can easily use TCP cubic to evaluate the RL model.

B. Scenario

We use the dumbbell topology (with three senders; three receivers and two routers) to test the model. figure 7 and 8 illustrate the test scenario.

TABLE III
STATISTICS THAT ARE USED BY THE RL TO TAKE A NEW ACTION

statistic	description
thr	the avg of delivery rate (throughput)
l	the avg of loss rate
d	the avg of delay
n	the number of valid ACK
m	the time between the last and current report
$sRTT$	the smooth RTT of packets so far
$cwnd_u$	the current congestion window
thr_{max}	the max value of delivery rate so far
d_{min}	the maximum value of packet so far

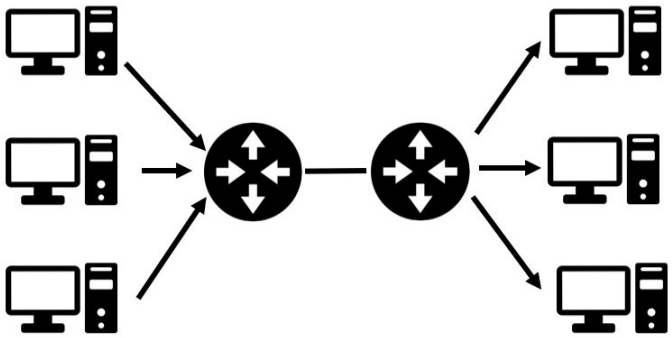

Fig. 7. Dumbbell Topology that is used for test the RL model

The capacity of the link between 2 routers is similar to sending bandwidth of each senders, for exampel in the first hour is 10 MB/s, in the second hour is 100 Mb/s, and in the last part is 1000 Mb/s. We run this scenario for 3 hours. This 180-minute is divided into 3 parts and each of these three-part has its own characteristic for senders (the bandwidth of senders will change every 60 minutes).

To collect the parameters to assess the models, after each 1 sec, we capture the value for buffer usage and throughput. We use these two as evaluation parameters of the model. These

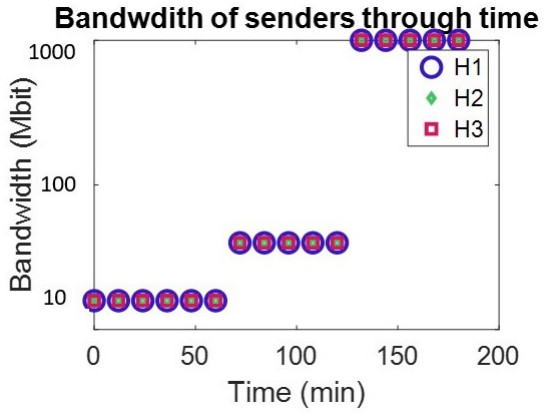

Fig. 8. Senders bandwidth for 180 minutes of test data. in the first 60 minutes, senders 30 MB/s; the second part, 300 Mb/s. In the third part total, sender's bandwidth will be 3,000 Mb/s a

TABLE IV
THROUGHPUT RESULTS COMPARISON OF RESULTS FOR BASELINE AND RL TCP. TCP CUBIC HAS BETTER RESULTS THAN THE RL MODEL

	0-60 min mean (std)	60-120 min mean (std)	120-180 min mean (std)	total (0-180 min) mean (std)
TCP cubic	0.81 (0.05)	0.94 (0.02)	0.99 (0.01)	0.91 (0.03)
TCP RL	0.78 (.08)	0.89 (.05)	0.97 (0.015)	0.88 (0.05)

Evaluation parameters are used as input for the metric.

C. Metric

In CC problem, the optimal goal is to increase throughput and decrease the buffer usage (low latency). However, reach to these two goals is kind of paradoxical but in this research, we use these two parameters as our matrices. So the metrics are

1) Bandwidth
2) Buffer Usage

To assess easily the results; all metrics are normalized between 0 and 1. For example, if buffer size=0 means that the buffer is completely empty and if buffer size=1 means that the buffer is full. This normalization helps to easily compare the results of the baseline model and TCP RL model to each other.

D. Results

In this section, we provided results of running both baselines (TCP cubic) and TCP RL for the scenario we described before. At first, we present results for the throughput. Table IV shows the results for the throughput metric. As we mentioned before, every 1 sec we capture the throughput. In the table, the mean and standard deviation is calculated by the captured throughput in the mentioned time (more detailed results for bandwidth and queue usage are presented in figures 12, and 13 in the Appendix).

With tuning RL parameter, we got better results for RL TCP in comparison to the previous report. Now, the results of TCP RL are comparable to TCP cubic. Secondly, the results for buffer usage have been shown in the table V.

However in the buffer usage table, one can see buffer usage for the RL model is less than TCP cubic. However, this different is not very distinguishable and we can say both models act similarly to each other. For BW metric, TCP cubic a little better than RL; and in the buffer usage, RL is a little better. As you see, when we tune the RL parameters; it leads to better results. To make the RL model scalable we need to train it by using more actors and benefiting from different topologies.

TABLE V
BUFFER USAGE RESULTS COMPARISON OF RESULTS FOR BASELINE AND RL TCP. IT SEEMS THAT RL MODEL HAS A BETTER BUFFER USAGE

	0-20 min mean (std)	20-40 min mean (std)	40-60 min mean (std)	total (0-60 min) mean (std)
TCP cubic	0.78 (0.24)	0.66 (0.28)	0.59 (0.18)	0.68 (0.23)
TCP RL	0.71 (.35)	0.69 (.18)	0.14 (0.105)	0.64 (0.22)

V. CONCLUSION

In this project, we benefit from a new trending Machine learning algorithm to let the model capture the dynamic of data. This data-driven model (in comparison with model-based (parameter-based) models) helps us to tune the model based on each data environment. Also, we use Mininet as the emulator that helps us to generate as much as data in a control-based environment. For training DNN models; sufficient data plays a very crucial role and using such an emulator provides numerous data sets. In addition, the most benefit of RL is learning through time; as time goes on; based on new data sets; RL agent becomes more robust and produces better results. In below; we mention some of the future works that we can do to improve this research:

1) **More sophisticated RL method** In this research, because of a lack of time and resources, we reduce the complexity of the RL model; however, in Deep Learning models; we will benefit from them by creating a large amount of data and complex network. Then, the DNN model will be able to capture hidden features of the data and lead to better results.
2) **Using multi-actor**: By increasing number of actors; we will have more robust results.
3) **Exploring other reward functions**: In the RL model; the goal is to train the agent to have the best policy and this policy is based on the definition of reward. Therefore; the definition of reward is the most crucial aspect of the RL model. We can add more metrics to the reward function and it might lead to better results.
4) **Comparision with other DNN models**: Comparing this RL model with other DNN models helps to understand how powerful is this in the domain of data-driven models.
5) **More emulators**: Using other emulators like NS2, and NS3 helps us to provide more synthetic data. If the RL method is trained with different kinds of data sets; it helps to have a more robust result. In training sessions using more emulators; will lead to better training and therefore better performance. Also, replicating results with different emulators; makes us confident about the sturdiness of the model.
6) **Chnage Input space**: Using different sets of statistics as features (environment in the RL terminology). Input features provide the most crucial role in learning of ML method. Using a different set of features might lead to better policy and therefore better results.
7) **Hyper-parameter tuning**: Tuning of DNN models is very important and has lots of effects on the final result. Tuning is somehow a trial-and-error process; it takes time to reach the proper hyper-parameter and we have not addressed tuning parameters in this work
8) **Multiple Topology for training**: if we use different actors and assign them to different topologies; it will lead to better results. Foe example, we can use 3 more actors for the star, the mesh, and the three topologies. More information about these three topologies has been

described in the Appendix section.

In the end, we acknowledge that training the RL network for TCP CC is a challenging task and might be bigger than the scope of the project of the course. However, we try to do our best, and during this time we changed multiple time the modeling to run the TCP RL. We might need more time to explore this domain. We expect to continue this interesting work even after the semester (when we have more time) to find more robust results.

A. Statement of Work

In this project Yihao Cai and Reza Saadati Fard collaborated with each other. However, after the intermediate 2 report and the class presentation, Reza Saadati Fard withdrew from the course and stopped his collaboration.

REFERENCES

[1] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. Classic meets modern: A pragmatic learning-based congestion control for the internet. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 632–647, 2020.

[2] Tongyu Dai, Xinggong Zhang, Yihang Zhang, and Zongming Guo. Statistical learning based congestion control for real-time video communication. *IEEE Transactions on Multimedia*, 22(10):2672–2683, 2020.

[3] N. Fonseca and M. Crovella. Bayesian packet loss detection for tcp. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1826–1837 vol. 3, 2005.

[4] P. Geurts, I. El Khayat, and G. Leduc. A machine learning approach to improve congestion control over wireless computer networks. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 383–386, 2004.

[5] Pierre Geurts, Ibtissam El Khayat, and Guy Leduc. A machine learning approach to improve congestion control over wireless computer networks. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 383–386. IEEE, 2004.

[6] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[7] A. Jayaraj, T. Venkatesh, and C. Siva Ram Murthy. Loss classification in optical burst switching networks using machine learning techniques: improving the performance of tcp. *IEEE Journal on Selected Areas in Communications*, 26(6):45–54, 2008.

[8] Huiling Jiang, Qing Li, Yong Jiang, GengBiao Shen, Richard Sinnott, Chen Tian, and Mingwei Xu. When machine learning meets congestion control: A survey and comparison. *Computer Networks*, 192:108033, 2021.

[9] Ning Jing, Ming Yang, Shaoyin Cheng, Qunfeng Dong, and Hui Xiong. An efficient svm-based method for multi-class network traffic classification. In *30th IEEE International Performance Computing and Communications Conference*, 2011.

[10] Leonard Kleinrock. An early history of the internet [history of communications]. *IEEE Communications Magazine*, 48(8):26–36, 2010.

[11] Josip Lorincz, Zvonimir Klarin, and Julije Ožegović. A comprehensive overview of tcp congestion control in 5g networks: Research challenges and future perspectives. *Sensors*, 21(13), 2021.

[12] Tahir Nawaz Minhas and Markus Fiedler. Quality of experience hourglass model. In *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, pages 87–92, 2013.

[13] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. *SIGMETRICS Perform. Eval. Rev.*, 33(1):50–60, jun 2005.

[14] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick Duffield. Class-of-service mapping for qos: A statistical signature-based approach to ip traffic classification. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, page 135–148, New York, NY, USA, 2004. Association for Computing Machinery.

[15] Basu Dev Shivahare, Shashikant Suman, Sai Sri Nandan Challapalli, Prakarsh Kaushik, Amar Deep Gupta, and Vimal Bibhu. Survey paper: Comparative study of machine learning techniques and its recent applications. In *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, volume 2, pages 449–454, 2022.

[16] Subir Varma. Chapter 1 - introduction. In Subir Varma, editor, *Internet Congestion Control*, pages 1–24. Morgan Kaufmann, Boston, 2015.

[17] Ruitao Xie, Xiaohua Jia, and Kaishun Wu. Adaptive online decision method for initial congestion window in 5g mobile edge computing using deep reinforcement learning. *IEEE Journal on Selected Areas in Communications*, 38(2):389–403, 2019.

[18] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l*, pages 250–257, 2005.

[19] Lei Zhang, Yong Cui, Mowei Wang, Zhenjie Yang, and Yong Jiang. Machine learning for internet congestion control: Techniques and challenges. *IEEE Internet Computing*, 23(5):59–64, 2019.

[20] Ticao Zhang and Shiwen Mao. Machine learning for end-to-end congestion control. *IEEE Communications Magazine*, 58(6):52–57, 2020.

APPENDIX

A) Topology:

Here we introduce other topologies that we can use in the multi-actor model that is presented in the paper.

1) **Star Topology:** This topology (as Figure 9) is regarded as one of the most common network setups. In the architecture, each peripheral host is connected to one centralized device, and typically it is identified as multi-client to one-server model where hosts act like clients while central device performs as a server. Star Topo holds the characteristics of: Centralized management of the network, ease of scalability and disaster recovery capability. In our proposed method, we intend to set up several connected hosts, apply our optimized RL algorithm to the centralized device and take some of the performance metrics (throughput, packet loss, burstiness and fairness,) for evaluating the real-time effect.
2) **Mesh Topology:** Unlike the previous two Topos, there's no specific agent that works for all the other network devices in mesh topology. Each host severs as a sole point on its own where every single node must connect at least two other nodes. Mesh topology provides the users with considerable reliability, but it lacks enough flexibility for maintenance and what's more, such design is also of high cost. In our proposed example test, we decide to run up six hosts under local mesh topology and randomly choose one of them as the agent running the RL-based congestion control algorithm to observe its effect on the overall network performance (QoS).
3) **Tree Topology:** Similar to Mesh Topo, no centralized process device shows up in Tree topology and each host connects to each other with shape of the structure being as an upside down tree (forest). Additionally, it is regarded as the combination of Bus Topo and Star Topo, and is also usually implemented in Data Center network. Tree Topo holds the advantages at its robustness for fault segmentation – the other nodes in a Tree Topo will not be affected if one of the nodes gets down. Furthermore, its extensible characteristic makes the network maintainable and callable. For simplicity and usability, we decide to set the top device as the intelligent CC host to valid the algorithm's impact on overall network devices.

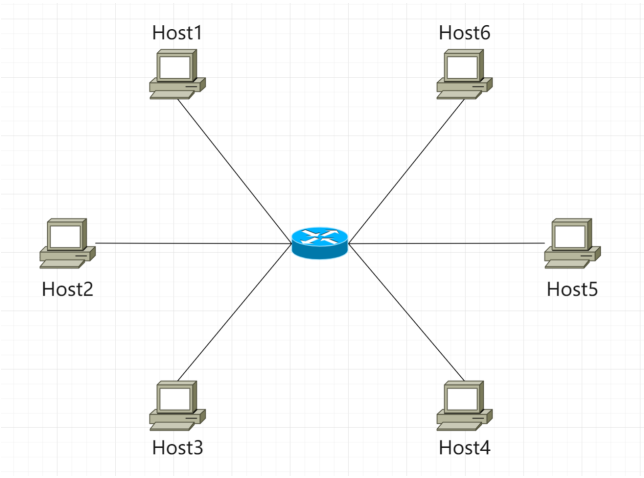

Fig. 9. Star Topology

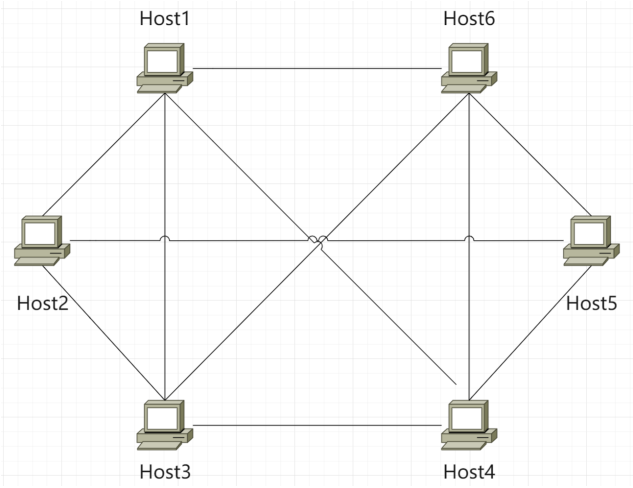

Fig. 10. Mesh Topology

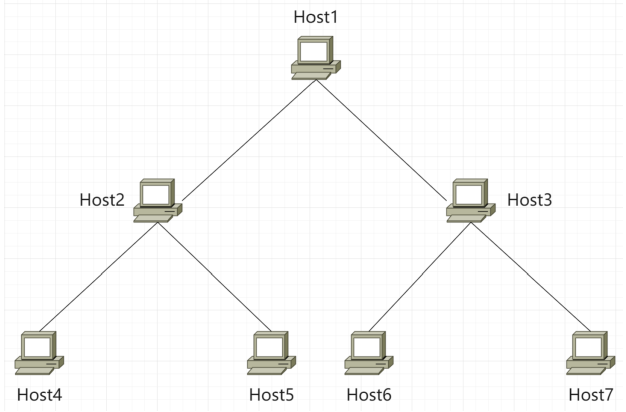

Fig. 11. Tree Topology

if we include these topologies in the training section, it can lead to more robust model. In the future; we will try to use multi-actor topology for training the model and it can lead to a better model.

B) Results: Here, we present our results for Bandwidth and Queue usage in detail. As we mentioned, we have three different parts and each part is 1 hour. We also run the RL TCP and TCP cubic for sneders with BW 10, 100, and 1000 Mbps. Figure 12 shows the results of these three tests for BW usage. In figure 13 we depict results for the queue usage. In the results section of the paper; we take the mean and standard deviation of theses data and put them into tables IV and V to make them easier to read.

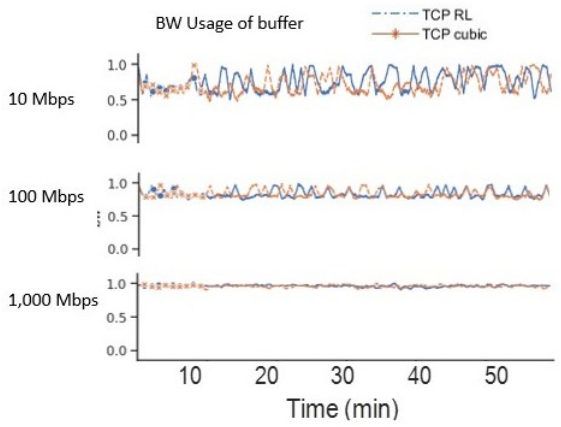

Fig. 12. Bandwidth usage for three differe

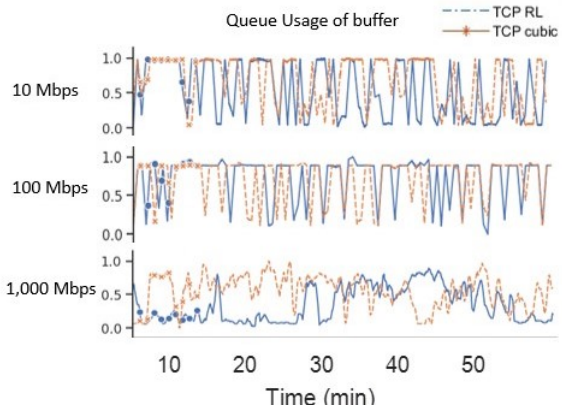

Fig. 13. Tree Topology