International Journal of Computer Applications (0975 - 8887)
Volume 186 - No.65, January 2025

Adaptive Congestion Control Protocol based on Meta-Reinforcement Learning for Data Communication Networks

Mugoh Mwaura
Jomo Kenyatta University of Agriculture and Technology
Nairobi, Kenya

Stephen Kiambi
Jomo Kenyatta University of Agriculture and Technology
Nairobi, Kenya

Hezekiah Nganga
Jomo Kenyatta University of Agriculture and Technology
Nairobi, Kenya

ABSTRACT

Congestion control protocols aim to optimize network capacity utilization and maximize throughput by selecting appropriate transmission rates for the sender. When approached as a Reinforcement Learning (RL) problem, congestion control policies derived from this framework exhibit superior performance compared to manually designed protocols. However, the practical implementation of RL algorithms encounters a significant challenge due to the dynamic nature of network conditions, which hampers their ability to generalize to new scenarios. This study presents a solution to this issue by considering the vast range of network conditions as an unknown task, treating it as a concealed variable that can be inferred from observed network history. By acquiring the capability to estimate this task as an underlying state and conditioning the protocol to respond accordingly, the proposed approach achieves continuous adaptation to evolving network conditions. The results demonstrate that this method not only enhances the utilization of network capacity in congestion control algorithms but also ensures protocol consistency across diverse network characteristics.

Keywords

Congestion control, reinforcement learning, latent models, meta-learning

1. INTRODUCTION

In a communication link, each connection has a sender which streams packets of traffic onto the link, and a receiver that sends packets of acknowledgement (ACKs) to the source of traffic. The sender responds to this feedback by adjusting its sending rate to ensure effective utilization of the channel.
The adjustment of the transmission rate in a communication link is determined by a congestion control protocol implemented at each endpoint of the connection [3, 12]. This protocol utilizes network parameters, such as bandwidth, inferred from the acknowledgment history, to determine the next transmission sending rate. It can be denoted as a function f_ψ:

$$f_\psi : \mathbf{x}_{t:t-H} \rightarrow k_t \tag{1}$$

k_t : The sending rate at time t
$\mathbf{x}_{t:t-H}$: Perceived ACKs history from time t to the size of the history length $t - H$
H : Number of buffered ACKs

The objective of congestion control protocols is to select an optimal rate that meets the node's throughput and latency requirements based on the specific application. An optimal transmission rate should be sufficiently high to efficiently utilize the link capacity, while ensuring that the traffic in transit does not exceed the bandwidth-delay product of the link [11]. However, since the bandwidth is not directly observed and requires estimation techniques [21], achieving a balance between bandwidth utilization and congestion avoidance is challenging for the congestion control protocol. The protocol aims to address this challenge through the design of the control parameters ψ in Equation 1.
The protocol's parameters ψ can be manually configured using Additive Increase Multiplicative Decrease (AIMD) algorithms [5]. AIMD-based algorithms linearly increase the congestion window and exponentially decrease it when congestion is detected. Examples of protocols that employ AIMD include Binary Increase Congestion Control (BIC) [3] and Transmission Colntrol Protocol based on cubic window growth function (TCP CUBIC) [12]. However, a drawback of these algorithms is that they decrease the sending rate whenever a packet loss occurs, even though not all losses are necessarily caused by congestion. Increasing the sending rate can effectively mitigate losses unrelated to congestion.
The challenge of congestion control has been approached by reframing it as an RL problem [26], aiming to learn an optimal congestion control policy that determines the appropriate action for adjusting the sending rate. RL-based control policies have shown superior performance compared to manually designed protocols [17, 15, 28, 16]. In this context, the congestion control function f_ψ in Equation 1 is redefined as a policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$, which maps the current network history $\mathbf{s}_t = \mathbf{x}_{t:t-H}$ to the optimal control action a_t (next sending rate).
Aurora [15] applies RL to internet congestion control and demonstrates its ability to capture complex network patterns, surpassing widely adopted and state-of-the-art protocols [12, 3, 7] at the time of publication. However, Aurora does not address the challenge of

generalizing to new network conditions. When the congestion control protocol operates in a network beyond its training domain, it fails to achieve the same competitive performance.
Conventional on-policy and off-policy RL algorithms [27, 19, 23] are incapable of generalizing to novel environments. Consequently, an RL-based congestion control protocol may perform poorly in unfamiliar network conditions, potentially even worse than a manually designed protocol.
To tackle this issue, the proposed approach focuses on learning a control policy that can adapt to variations in network conditions over time. This approach extends the capabilities of Aurora, an RL-based control protocol, by leveraging meta-learning, where algorithms adapt their behavior based on past experiences to match the current task [10, 22, 13]. Specifically, the approach initially focuses on variations in link bandwidth to simplify the handling of network dynamics. However, since the solution effectively adapts to network changes, it should deliver desirable performance when all key link parameters (bandwidth, latency, random packet loss and queue size) [29] vary significantly. This adaptability arises from treating these parameters as hidden latent variables inferred from experience. Incorporating additional variables merely increases the number of network dimensions the algorithm needs to learn, necessitating more interaction with the environment.
The objective of the approach is to create a congestion control algorithm that can continuously adapt to non-trivial changes in networks and minimize congestion while ensuring efficient bandwidth utilization, across networks with different characteristics.
The remaining sections of the document are structured as follows: In Chapter 2, the proposed method is examined. Chapter 3 showcases the results and discussion of the experiments conducted, while Chapter 4 comprises the conclusion.

2. META RL-BASED ADAPTIVE CONGESTION CONTROL

Meta RL has had successful application in robotics for continuous task adaptation and generalisation across unseen agents in multi-agent RL [1]. These settings show the potential of application of meta RL to congestion control which shares a similar problem in terms of task adaptation and generalisation to novel environment characteristics.

2.1 Congestion control as Reinforcement Learning

A Markov Decision Process (MDP), characterized by the property that the subsequent state $\mathbf{s}_{t+1}$ is solely determined by the current state $\mathbf{s}_t$ and action $\mathbf{a}_t$, can be expressed as

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, p_0, \gamma, T) \tag{2}$$

where $\mathcal{S}$ is the state space while $\mathcal{A}$ represents the action space. R represents the reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, P is the environment transition dynamics function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ sutton$_b ach_b arto_2$018.p$_0$ defines the initial state distribution, γ the cumulative reward discount factor and T the horizon of the environment episode. The objective of RL methods is to find a policy that maximizes the expected cumulative reward over trajectories $\tau = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_T, \mathbf{a}_T)$ induced by the policy $R(\tau) = \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)$.
To apply the MDP framework to congestion control, this work adopts an approach inspired by Aurora:

—$\mathcal{A}$: The action space $\mathcal{A}$ represents changes to the transmission rate k_t. Time is divided into monitor intervals (MI) dong2015,

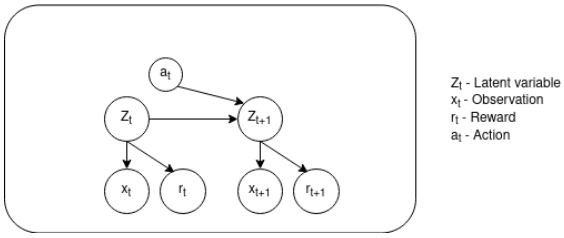


Fig. 1: Tasks as partially observable MDPs.

where k_t remains fixed within each MI. The sending rate is updated by the Aurora algorithm as follows:

$$k_{t+1} = \begin{cases} x_t \times (1 + \alpha \mathbf{a}_{t+1}), \mathbf{a}_{t+1} > 0 \\ x_t / (1 - \alpha \mathbf{a}_{t+1}), a_{t+1} < 0 \end{cases} \tag{3}$$

$\mathbf{a}_t$: proposed next agent action
α : dampening factor controlling oscillations

—$\mathcal{S}$: The state space $\mathcal{S}$ consists of a history of network statistics derived from packet acknowledgements. These statistics include: (i) the derivative of latency with respect to time, (ii) the ratio of sent to acknowledged packets, and (iii) the ratio of the mean latency of the current MI to the minimum recorded MI mean latency.
The history length is a hyper-parameter. Longer histories are expected to improve performance.

—$r(\mathbf{s}_t, \mathbf{a}_t)$: The reward function is designed to maximize throughput while minimizing latency and packet loss, expressed as $throughput - latency - loss$.

2.2 Meta-RL of Congestion Control with Latent Dynamics Models

The fundamental concept is that the process of inferring the state of a network and predicting significant variations in the network based on observations and rewards is essentially the same as the task of estimating hidden variables from accumulated experience. This concept is demonstrated in Figure 1, where latent variables $\mathbf{z}_t$ in a partially observable environment can provide insights into the true state by examining the history of observations $(\mathbf{x}_0, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_T)$. The task, which is obtained from the distribution of the environment $\mathcal{T} \sim p(\mathcal{T})$, is considered as a component of $\mathbf{z}_t$. Therefore, the latent variable encompasses both the hidden state $\mathbf{s}_t$ and the task. The proposed method aims to learn valuable representations of the network statistics described in Section 2.1 and encode them in a latent variable $\mathbf{z}_t$.
The latent variable model [8] makes inferences on the belief over the latent $\mathbf{z}_t$ and adjusts the posterior of this belief at each time step based on the incomplete observation from the current time step. By incorporating rewards into the input of the latent model, the hidden state $\mathbf{z}_t$ can encompass task-related information, thereby facilitating meta-learning.
The maximum log-likelihood of the input data, consisting of observations and rewards, is achieved by learning the latent state model over the inferred variables $\mathbf{z}_t$

$$\max_{\phi} \mathop{\mathbb{E}}_{\mathcal{T}\sim p(\mathcal{T})} \left[\mathop{\mathbb{E}}_{\substack{\mathbf{a}_t\sim\pi_\theta(\cdot|\mathbf{g}_t)\\ \mathbf{x}_t\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t)\\ \mathbf{s}_{t+1}\sim p_{\mathcal{T}}(\cdot|s_t,a_t)\\ r_t\sim r_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)}} \left[\log p_\phi\left(\mathbf{x}_{1:T}, r_{1:T}|\mathbf{a}_{1:T-1}\right)\right] \right] \quad (4)$$

Given an observation $\mathbf{x}_t$ and reward r_t from the current timestep, the agent uses the posterior from the previous time step $\mathbf{z}_{t-1}$ to update its posterior belief. Posterior inference shows the belief $\mathbf{g}_t$ over the current time-step hidden variable $\mathbf{z}_t$, i.e.

$$\mathbf{g}_t \sim p(\mathbf{z}_t|\mathbf{x}_{1:t}, r_{1:t}, \mathbf{a}_{t-1}) \quad (5)$$

Conditioning the congestion policy on the inferred belief i.e. $\pi_\theta(\mathbf{a}_t|\mathbf{g}_t)$, gives it the ability to adapt its behaviour in varying network conditions and uses. The policy adaptation objective incorporates posterior inference on the latent state model, giving the following meta-learning objective:

$$\max_{\theta} \mathop{\mathbb{E}}_{\mathcal{T}\sim p(\mathcal{T})} \left[\mathop{\mathbb{E}}_{\substack{\mathbf{a}_t\sim\pi_\theta(\cdot|\mathbf{g}_t)\\ \mathbf{x}_t\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t)\\ \mathbf{s}_{t+1}\sim p_{\mathcal{T}}(\cdot|s_t,a_t)\\ r_t\sim r_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)}} \left[\sum_{t=1}^{T} \gamma^t r_t\right] \right] \quad (6)$$

$$\text{where } \mathbf{g}_t = p(\mathbf{z}_t|\mathbf{x}_{1:t}, r_{1:t}, \mathbf{a}_{t-1})$$

2.2.1 Variational Inference of the Latent State. Variational inference [2] converts the inference problem into an optimization problem with the aim of maximizing the evidence lower bound (ELBO) [6] on the log-likelihood of the meta-RL objective. The optimization of the ELBO, as presented in [31], is applied to the log-likelihood objective described in Equation 4, maximizing the ELBO objective in the following manner:

$$\mathbb{E}_{\mathbf{z}_{1:T}\sim q_\phi}\left[\log p(\mathbf{x}_{1:T}, r_{1:T}|\mathbf{a}_{1:T-1}\right] \geq \mathcal{L}_{model} \quad (7)$$

where $\mathcal{L}_{model}$ is the training loss of the latent state model and is defined as:

$$\begin{aligned} &\mathcal{L}_{model}(\mathbf{x}_{1:T}, r_{1:T}, \mathbf{a}_{1:T-1}) \\ =&\mathbb{E}_{\mathbf{z}_{1:T}\sim q_\phi} \sum_{t=1}^{T} p_\phi(\mathbf{x}_t|\mathbf{z}_t) + \log p_\phi(r_t|\mathbf{z}_t) \\ &- D_{KL}(q_\phi(\mathbf{z}_1|\mathbf{x}_1, r_1)||p(\mathbf{z}_1) \\ &- \sum_{t=1}^{T} D_{KL}\left(q_\phi(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, r_{t+1}, \mathbf{z}_t, \mathbf{a}_t)||p_\phi(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)\right) \end{aligned} \quad (8)$$

The terms $p_\phi(\mathbf{x}_t|\mathbf{z}_t)$ and $p_\phi(r_t|\mathbf{z}_t)$ serve to incentivize the model to extract valuable information from the inputs $(\mathbf{x}_t, r_t)$, which is condensed in the latent variable. The inference function $q_\phi(\mathbf{z}_t|\mathbf{a}_{t-1}, r_t)$ and the latent dynamics function $p_\phi(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ are both fully connected neural networks. The model employs separate encoders for the reward $p(\tilde{r}_t|r_t)$ and for the observation $p(\tilde{\mathbf{x}}_t|\mathbf{x}_t)$, as well as decoders $p_\phi(\mathbf{x}_t|\mathbf{z}_t)$ and $p_\phi(r_t|\mathbf{z}_t)$ for observations and rewards, respectively. The policy is learned using Soft Actor Critic (SAC) haarnoja2018sac, which is advantageous for its off-policy nature, resulting in improved sample efficiency.
The learning procedure cycles between using the current policy to collect trajectories, using the objective in Equation (6) to train the latent state model and updating the policy as outlined in Algorithm 1.

Algorithm 1 RL congestion control with latent representations

Require: A distribution of network training tasks $p(\mathcal{T})$
Initialize the model q_ϕ, critic Q_φ and policy π_θ
Initialize the memory buffers $\mathcal{B}_i$ for each task
repeat
  for each task $\mathcal{T}_i \in p(\mathcal{T})$ **do** ▷ Collect data
    Infer first posterior $\mathbf{g}_1 = q_\phi(\mathbf{z}_1|\mathbf{x}_1, r_1)$
    Take action $\mathbf{a}_1 \sim \pi_\theta(\mathbf{a}_1|\mathbf{g}_1)$
    for each step $t = 2$ to max steps $T-1$ **do**
      $\mathbf{v}_t \leftarrow r_t, \mathbf{a}_{t-1}, \mathbf{x}_t$
      Infer the posterior $\mathbf{g}_t \sim q_\phi(\mathbf{z}|\mathbf{z}_{t-1}, \mathbf{v}_t)$
      Sample the action $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}|\mathbf{g}_t)$
    end for
    Add transitions $\{\mathbf{x}_{1:T}, r_{1:T}, a_{1:T-1}\}$ to the current task's buffer $\mathcal{B}_i$
  end for

  for each training step to max steps **do**
    for each task $\mathcal{T}_i \in p(\mathcal{T})$ **do** ▷ Train the model
      Sample trajectories $\{\mathbf{x}_{1:T}, r_{1:T}, \mathbf{a}_{1:T-1}\} \sim \mathcal{B}_i$ from task buffer
      $\mathbf{v}_{1:T} \leftarrow (r_{1:T}, \mathbf{a}_{1:T-1}, \mathbf{x}_{1:T})$
      Infer posteriors $\mathbf{g}_{1:T} = q_\phi(\mathbf{z}|\mathbf{z}_{1:T-1}, \mathbf{v}_{1:T})$
      Reconstruct observations $\tilde{\mathbf{x}}_{1:T}$ and rewards $\tilde{r}_{1:T}$
      Minimize the reconstruction error
      $\mathcal{L}_i = \mathcal{L}_{model}(((\tilde{\mathbf{x}}_{1:T}, \tilde{r}_{1:T}), (\mathbf{x}_{1:T}, r_{1:T})))$
    end for
    Update the model
    $\phi \leftarrow \phi - \alpha\nabla_\phi \sum_i \mathcal{L}_i$
  end for
  for each policy training step t to max steps **do**
    for each task $\mathcal{T}_i \in p(\mathcal{T})$ **do** ▷ Train the actor-critic
      Train the actor $\pi_\theta(\mathbf{a}_t|\mathbf{g}_t))$ and critic $Q_\psi(\mathbf{a}_t, \mathbf{g}_t)$
      conditioned on the latent representation $\mathbf{g}_{1:T} \sim q_\phi$
    end for
    Update the actor
    Update the critic
  end for
until performance is desirable

Algorithm 2 Meta-testing congestion control

Require: Test network task $\mathcal{T} \sim p(\mathcal{T})$
Infer first posterior $\mathbf{g}_1 = q_\phi(\mathbf{z}_1|\mathbf{x}_1, r_1)$
Take action $\mathbf{a}_1 \sim \pi_\theta(\mathbf{a}_1|\mathbf{g}_1)$
for each step $t = 2$ to max steps $T-1$ **do**
  $\mathbf{v}_t \leftarrow r_t, \mathbf{a}_{t-1}, \mathbf{x}_t$
  Infer the posterior $\mathbf{g}_t \sim q_\phi(\mathbf{z}|\mathbf{z}_{t-1}, \mathbf{v}_t)$
  Sample the action $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}|\mathbf{g}_t)$, get $\mathbf{x}_{t+1}$, reward r_{t+1}
end for

2.3 Running environment

The congestion control protocol was trained using Aurora's environment [15], which emulates a solitary traffic source with varying network parameters. To evaluate the trained protocols, distinct environments with tasks different from the training ones were employed.

3. RESULTS AND DISCUSSION

The study aimed to explore the adaptability of congestion control protocols to substantial network changes through the use of meta-learning latent representations of the network. The experimental analysis sought to compare the performance of a meta RL agent with that of a standard RL agent. Specifically, the study examined how the actions taken by each agent impacted the utilization of network capacity and the ability to control congestion in network patterns that were not encountered during the training phase.

3.1 Environment setup

The network environment simulated by [15] was expanded to facilitate the training and evaluation of both the meta-learned agent and Aurora's agent. This environment is initialized with a designated task, where a task refers to a collection of pairs consisting of minimum and maximum values for various network parameters such as bandwidth, latency, queue size, and random loss rate. At each time step, the current value of a network parameter is randomly chosen from within the specified minimum and maximum values for that parameter.

For the experiments conducted, the variations in tasks were limited to the bandwidth alone. This was done to simplify the environment and accelerate the learning process of the congestion control agents. Consequently, the remaining network parameters were kept constant within a predetermined range of minimum and maximum values, while the bandwidth minimum and maximum range differed for each task. After the completion of each training episode, a new task was chosen for the subsequent episode.

In the course of this evaluation, the algorithm under investigation in this study is referred to as MLLD (Meta-learning Latent Dynamics), and PPO (Proximal Policy Gradients) as the algorithm employed by Aurora, against which the comparison of congestion control agents is conducted. The term "agent" is used interchangeably with congestion control protocol.

3.2 Evaluating the protocol in unseen network conditions

The evaluation process involved testing the trained congestion control policy in a network environment with random bandwidth variations that fall within the same distribution as those encountered during the training phase. The evaluation episodes were limited to a length of 150, and a total of 20 evaluation tasks were performed. For each task, the agent underwent evaluation in four episodes. Each task consisted of a random pair of minimum and maximum bandwidth values. These minimum/maximum bandwidths thresholds were set to deviate within a 50% range from a central bandwidth value (c_{bw}) $c_{bw} : c_{bw} \pm c_{bw} \times \text{rand}(0, 0.5)$. The center bandwidth is selected from a set of n equally spaced bandwidth values ranging from 1 to 1000 Mbps, spaced out by pre-set ranges of $1000/n$. Thus, the bandwidth task is a set bw_{set} comprising n pairs of minimum-maximum bandwidth thresholds.

$$bw_{set} = \{(\min_{bw_0}, \max_{bw_0}), \ldots, (\min_{bw_n}, \max_{bw_n})\}$$

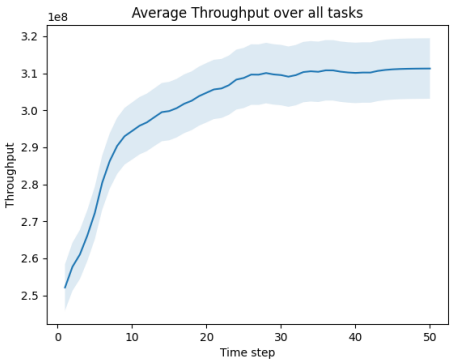
At the conclusion of each evaluation episode, the network environment was reset, and a new random pair of bandwidth thresholds was generated based on the current task. The findings of the evaluation report the averaged values of selected network features across the time steps of each evaluation task.

3.2.1 Throughput. In Aurora environment used, throughput is directly proportional to the quantity of bytes acknowledged and inversely proportional to the size of the network receive-window. If more data is transmitted per unit monitor-interval-time in the network, the resulting throughput will be higher.

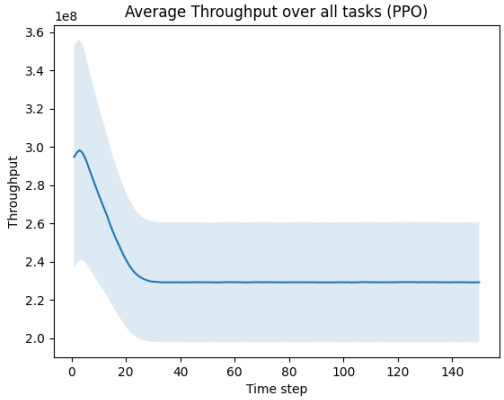
$$\text{throughput} = (\text{bytes\_ack} - \text{packet\_size})/\text{receive duration} \quad (9)$$

Improved utilisation of the link capacity with increase in bandwidth results in higher throughput. MLLD has a throughput that steadily increases over the course of evaluation, while PPO exhibits an initial high throughput that quickly decreases as it encounters new network variations as illustrated in Figure 2. This behaviour demonstrates MLLD 's agent's ability to identify it's current network pattern, act to maximize throughput, and get better with more environment interaction.

The sharp decline in the PPO agent can be attributed to its inability to derive meaningful network representations from states that differ from those encountered during training. Acting outside the training distribution, PPO tends to take actions that disregard the training objective, negatively impacting the protocol's throughput. Past the first 40 evaluation episodes in Figure 2b, it attained a constant throughput, indicating that the agent takes similar actions when optimising for network throughput regardless of the current bandwidth.



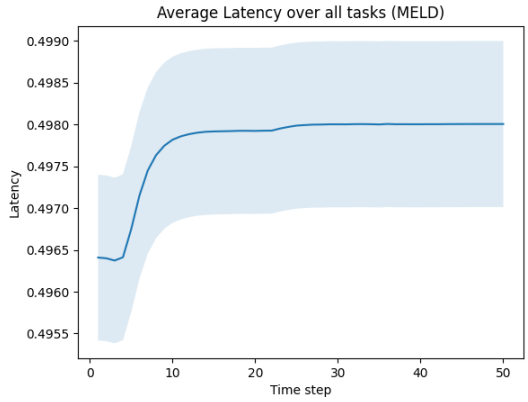
(a) MLLD



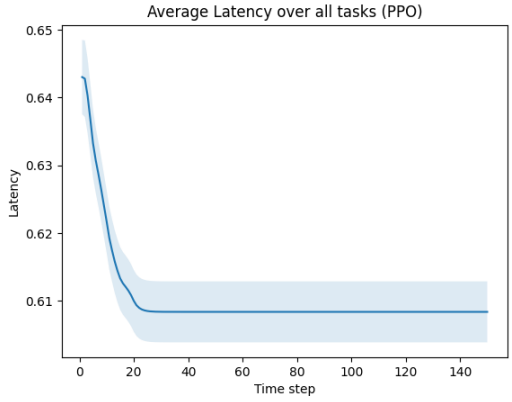
(b) PPO

Fig. 2: Comparison of task-averaged evaluation throughput of MLLD and PPO on the same network tasks

3.2.2 Loss rate. The loss rate of the PPO agent is an order of magnitude lower than MLLD 's as illustrated in Figure 4. This loss-rate difference can be attributed to the lower utilization of the link by PPO when the bandwidth is varied, as opposed to the MLLD which has higher link utilisation, that increases with increase in the link bandwidth. A constant loss rate percentage at a given bandwidth, will theoretically result in a rise in the amount of packets lost as the bandwidth gets higher. This interpretation fits the loss-rate curve of MLLD . As further illustrated in Figure 5 MLLD has a higher sending ratio $(1.5 - 3.5)$ compared to PPO's $(1.0 - 1.1)$. PPO's sending ratio also declines as evaluation progresses — the inverse of the sending ratio behaviour with the protocol trained on MLLD . Thus, though MLLD has a higher loss rate than PPO, the ratio of packets sent to those transmitted successfully is much higher than that of PPO.



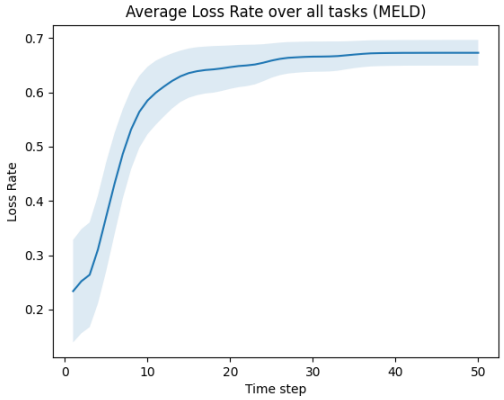
(a) MLLD



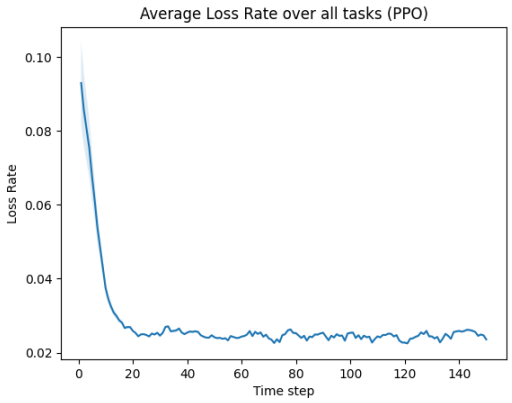
(b) PPO

Fig. 3: Comparison of task-averaged evaluation latency on the same network tasks

3.2.3 Latency. The MELD agent demonstrated a lower latency threshold $(0.4955 - 0.4990)$ which remained almost constant in previously unseen environments compared to the PPO agent whose latency ranged between $(0.5900 - 0.6450)$. This is shown in Figure 3. The lower variance in latency shows MLLD is able to make more informed decisions on which actions in the current network satisfy a low latency property while still utilizing the link efficiently.

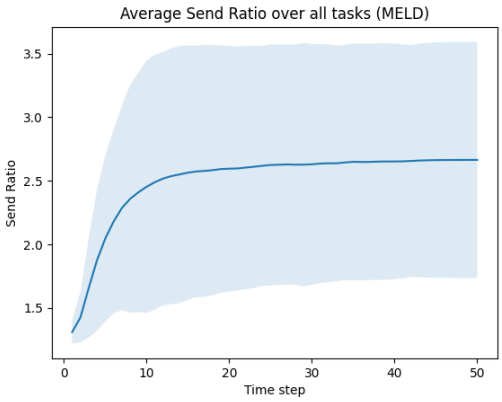


(a) MLLD

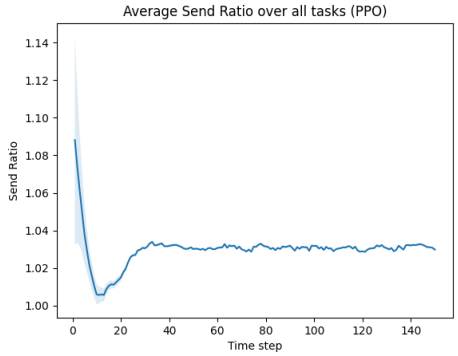


(b) PPO

Fig. 4: Comparison of task-averaged evaluation loss rate on the same network tasks



(a) MLLD



(b) PPO

Fig. 5: Comparison evaluation sending ratio across task-averaged episodes

4. CONCLUSION

This work has presented a novel approach for learning congestion control protocols that can continuously adapt to changing network conditions by extracting valuable latent features related to the network state. The variations in the network are modeled as partially observable task distributions, which are learned by an adaptation function based on the latent features. By treating significant network variations as hidden variables, latent variable models can be employed to extract meaningful network representations from the observed network history and estimate the unknown variables efficiently. By conditioning the meta-learner on these variables and enabling it to explore different network characteristics, the learner becomes capable of adapting to new network patterns and ultimately generalizing to unseen dynamic network conditions. The approach can be applied in real-world networks to create adaptive congestion control protocols that optimally utilize the network capacity while generalizing to network variations.

For future work, attention mechanisms can be used to aid with credit assignment to improve the learning ability of a long sequence of optimal sending rates. Reward-shaping methods could also be used to investigate how the optimizing of a certain network feature may affect other feature. Contrastive learning methods could be applied together with auto-encoders to improve the learning of better network representations based on the network dynamics. These methods would allow capturing of more task-relevant features.

Reward shaping and credit assignment: Since a congestion control protocol relies on long-term credit assignment, it's difficult to learn a long sequence of optimal sending rates. Attention mechanisms [14] can be used to aid with credit assignment over the long network horizons.

Single objective reward functions could be explored to examine how the optimizing for a certain network feature like throughput affects the rest of the features. Reward shaping methods [4] would prove useful in this area.

Learning better network representations: Selection of network state features used for conditioning the policy is done on the basis of domain knowledge. This may include inductive biases. A solution would be looking into how to incorporate the relevant information from all the state features efficiently to learn a good representation [24] for the network dynamics. Methods on constrastive learning [25, 9] could help learn more useful representations. Loss auto-encoders could further be used to capture more task-relevant features for invariant representation learning [30].

Generalizing to non-stationary network environments: With network dynamics changing over time, the congestion control policy should be robust to the non-stationality of the environment. In this work, we explored the changing environment representations as latent variables. An alternative approach would be assumption of uncertainty in the transitions and learn robust agents that consider worst-case environment situations [18]. The use of randomization and state augmentation during training [20] can also be tried as an approach to generalizing the policies in real networks.

The present study has examined the capacity of the congestion control protocol to adapt to significant variations in bandwidth. An area for further investigation would involve assessing the agent's capability to fulfill evolving network demands that encompass multiple objectives for the congestion control protocol, such as diverse combinations of user requirements for network bandwidth, latency, sending rate, or buffer queue size. Conducting tests on the protocol within an actual network environment would represent a progression towards its practical implementation.

The approach explored changing environment representations as latent variables. An alternative method would be assumption of uncertainty in the transitions and learning of robust agents that consider worst-case environment situations. Reward shaping and credit asignment method can also be examined. For instance, single objective reward functions could be explored to examine how the optimizing for a certain network feature like throughput affects the rest of the features.

5. REFERENCES

[1] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning, 2023.

[2] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Apr 2017.

[3] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control. *Queue*, 14(5):20–53, 2016.

[4] Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 4(2):2007–2014, 2019.

[5] Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, 1989.

[6] Justin Domke and Daniel Sheldon. Importance weighting and variational inference. *CoRR*, abs/1808.09034, 2018.

[7] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. PCC vivace: Online-learning congestion control. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 343–356, Renton, WA, April 2018. USENIX Association.

[8] B. Everett. *An Introduction to Latent Variable Models*. Springer Science & Business Media, Mar 2013.

[9] Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning, 2023.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017.

[11] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[12] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: A new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[13] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(09):5149–5169, sep 2022.

[14] Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature Communications*, 10(1), 2019.

[15] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. A deep reinforcement learning perspective on internet congestion control. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3050–3059. PMLR, 09–15 Jun 2019.

[16] 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.

[17] Wei Li, Fan Zhou, Kaushik Roy Chowdhury, and Waleed Meleis. Qtcp: Adaptive congestion control with reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 6(3):445–458, 2019.

[18] Daniel J. Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. Robust reinforcement learning for continuous control with model misspecification, 2020.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[20] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[21] R. Prosad, C. Davrolis, M. Murray, and K.c. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network*, 17(6):27–35, 2003.

[22] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[24] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning, 2020.

[25] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning, 2020.

[26] Richard S. Sutton, Francis Bach, and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press Ltd, 2018.

[27] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning (reinforce). *Reinforcement Learning*, page 5–32, 1992.

[28] Zhenchang Xia, Libing Wu, Fei Wang, Xudong Liao, Haiyan Hu, Jia Wu, and Dan Wu. Glider: rethinking congestion control with deep reinforcement learning. *World Wide Web*, 26(1):115–137, Jun 2022.

[29] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. Pantheon: the training ground for internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 731–743, Boston, MA, July 2018. USENIX Association.

[30] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction, 2021.

[31] Tony Z. Zhao, Anusha Nagabandi, Kate Rakelly, Chelsea Finn, and Sergey Levine. Meld: Meta-reinforcement learning from images via latent state models, 2021.