Intelligent and Converged Networks ISSN 2708-6240
2021, 2(3): 177–197 DOI: 10.23919/ICN.2021.0016

Model-based reinforcement learning for router port queue configurations

Ajay Kattepur*, Sushanth David, and Swarup Kumar Mohalik

Abstract: Fifth-generation (5G) systems have brought about new challenges toward ensuring Quality of Service (QoS) in differentiated services. This includes low latency applications, scalable machine-to-machine communication, and enhanced mobile broadband connectivity. In order to satisfy these requirements, the concept of *network slicing* has been introduced to generate slices of the network with specific characteristics. In order to meet the requirements of network slices, routers and switches must be effectively configured to provide priority queue provisioning, resource contention management and adaptation. Configuring routers from vendors, such as Ericsson, Cisco, and Juniper, have traditionally been an expert-driven process with static rules for individual flows, which are prone to sub optimal configurations with varying traffic conditions. In this paper, we model the internal ingress and egress queues within routers via a queuing model. The effects of changing queue configuration with respect to priority, weights, flow limits, and packet drops are studied in detail. This is used to train a model-based Reinforcement Learning (RL) algorithm to generate optimal policies for flow prioritization, fairness, and congestion control. The efficacy of the RL policy output is demonstrated over scenarios involving ingress queue traffic policing, egress queue traffic shaping, and one-hop router coordinated traffic conditioning. This is evaluated over a real application use case, wherein a statically configured router proved sub optimal toward desired QoS requirements. Such automated configuration of routers and switches will be critical for multiple 5G deployments with varying flow requirements and traffic patterns.

Key words: router port queues; model-based Reinforcement Learning (RL); network slicing

1 Introduction

The emergence of fifth-generation (5G) systems[1] poses a new set of challenges to telecommunication networks. Principally, Quality of Service (QoS) requirements would vary, depending on the types of flows. 5G systems include various classes of flows, including enhanced Mobile BroadBand (eMBB) for mobile data and telephony, Ultra Reliable Low Latency Communications (URLLC) for low-latency industrial connectivity, and massive Machine Type Communications (mMTC) for Internet of Things (IoT) applications. 5G network slicing[2] is an important technique to meet these diverse requirements. Network slicing depends on the accurate configuration of underlay components, such as routers and switches within large and complex networks.

Vendors of routers and switches have developed multiple models to perform traffic routing, shaping, and aggregation within 5G networks[1]. Chief among these are Ericsson 6000 series[3] and Juniper M-series[4], which provide high-capacity port interfaces for edge and aggregation routing. To provide QoS of various flows, routers must be appropriately configured for various traffic mixes and requirements. Typically, multiple routers may be configured using programmable Software-Defined Networking (SDN)[5]

• Ajay Kattepur and Swarup Kumar Mohalik are with the Artificial Intelligence System Group, Ericsson Research, Bangalore 560093, India. E-mail: ajay.kattepur@ericsson.com; swarup.kumar.mohalik@ericsson.com.
• Sushanth David is with the Ericsson Managed Services Unit in Texas, Plano, TX 75025, USA. E-mail: sushahth.s.david@ericsson.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2021-08-25; revised: 2021-09-13; accepted: 2021-09-29

frameworks. This is currently an (human) expert driven process with multiple configuration commit commands provided for each QoS flow[4, 6]. However, given the complexity and scale of networks, this is an inefficient technique that can quickly turn sub optimal to new traffic patterns. Moreover, it is a trial-and-error approach that depends on the experience of the programmer, which may fail under novel patterns. Thus, it is necessary to explore more automated techniques to manage such routers.

In this work, we explore the use of Reinforcement Learning (RL) approaches[7] to replace static configurations currently being used in routers. RL algorithms are applied to dynamically reconfigure queuing disciplines and weights, dependent on traffic patterns, queue lengths, and observed congestions. Specifically, we make use of the Partially Observable Markov Decision Process (POMDP)[8] to model states, actions, observations, and rewards within router port queues. Through action space exploration, which includes queue priority changes, weight changes, flow limitation, drop rate modification, and queuing discipline change, we demonstrate the generation of optimal policies for various traffic patterns.

This model is integrated with traffic shaping and policing algorithms typically used in router configuration[9]. This approach is demonstrated over a real use case, wherein a statically configured router is shown to cause deteriorated 5G system performance. This is specifically shown to be applicable in three aspects: (i) port ingress queue traffic policing to ensure fair allocation to all queues, (ii) port egress queue traffic shaping to prevent bottlenecks at the egress port, and (iii) one-hop routing to perform coordinated traffic shaping across multiple router ports.

The principal questions targeted in this paper are as follows:

(1) Can the configuration of the ingress and egress queues within a router be automated to adapt to different traffic conditions?

(2) Can policies generated via RL reduce congestions and packet dropping in router queues?

(3) What is the process to transform queuing model parameters to transition probabilities within model-based RL techniques?

The paper is organized as follows: The state-of-the-art is described in Section 2. Details on router queue modeling and associated protocols are studied in Section 3. A real 5G network slicing scenario and the need for router configuration are presented in Section 4. The description of the model-based RL approach for router queue configuration is presented in Section 5. Section 6 provides details on ingress traffic shaping, egress traffic policing, and one-hop traffic conditioning using the output of RL policies. The conclusions and future directions are discussed in Section 7.

2 Related work

In this section we review the related approaches in router queue modeling and the use of RL toward queuing systems.

2.1 Router modeling

To model the ingress and egress queues within routers, we make use of queuing network models. In Ref. [10], differentiated service (DiffServ) requirements at router ports were studied, including queuing types, priorities, and packet drop policies. In Ref. [11], early work on programming computations on routers and switches was introduced. The programmability of routing control has been further developed via Software-Defined Networking (SDN)[5], wherein control plane decisions may be made to route flows to specific router ports.

A scalable model used to maintain QoS in routers is DiffServ[12]. With DiffServ, the network tries to deliver a particular kind of service based on the QoS specified by each packet. Commonly used marking and policing mechanisms are single-rate 3-colour marking and dual-rate 3-colour marking[9]. The DiffServ marking has been incorporated with 5G network slicing[2] to provide guaranteed QoS to customer flow types. Network slicing builds on DiffServ by providing DiffServ at the radio access level. To this end, multiple 5G routers from Ericsson[3], Cisco[6], and Juniper[4] have been introduced. While the commands to configure routers are typically expert driven (trial-and-error), efforts are underway to make use of artificial intelligence techniques into these systems.

In this paper, we make use of the queuing modeling techniques and associated simulators to study the effect of parameter changes. The reconfiguration settings are then fed into the RL model for policy updates. The queuing network model is used to model the ingress and egress queuing outputs within the routers. This is a highly dynamic system that changes depending on the packet arrival rate, packet traffic mix, and priorities of flows. Continuously optimizing the system using rule-based or linear programming solvers would be inefficient. By contrast, RL algorithms offer multiple advantages: (i) the ability to continuously monitor the current state of the system and propose the most rewarding actions and (ii) learn about changes in a traffic mix that can cause deteriorated performance and suggest alternate configurations.

2.2 RL for queue-based systems

RL algorithms[7] have gained prominence in recent years due to their ability to generate optimal actions within use cases, such as vehicular traffic and scenario-based gaming. There have also been applications of RL algorithms toward networks. In Ref. [13], RL techniques were prosed as switch arbitrators in high speed networks. However, the effects of the queuing delay and traffic mix have not been considered. A prominent area of work is Q-routing[14], wherein RL techniques are used to provide an appropriate route considering link capacities and congestion effects. However, these techniques do not deal with the internal configurations of router ports. In Ref. [15], a survey of model-free and model-based techniques used for efficient routing over variants of the Q-routing protocol was studied. In Ref. [16], a knowledge layer was proposed over SDNs that can provide autonomous configurations via analytics, prediction, and tuning.

The use of RL in the automated configuration of complex systems has also received significant attention. In Ref. [17], the use of RL for adaptive bandwidth provisioning for DiffServ was studied. In Ref. [18], RL was used to optimize virtual machine job allocation and to preserve the fair use of CPU, memory, and disk resources. In Ref. [19], RL techniques were used to provide multi-objective optimization of distributed stream processing systems. In Ref. [20], the process of automatically adjusting the number of concurrently running jobs on a grid workstation was formulated as an RL problem. The queue length and the actions of accepting or blocking jobs were performed using hidden Markov models and Q-learning formalisms. Table 1 provides an overview of applying the state-of-art RL techniques network routing, traffic engineering, and queuing control. We provide limitations in the approaches within these set of papers.

To the best of our knowledge, we have seen limited work in the application of RL algorithms for router port queue configurations. Current static rule-based setup will not scale and cannot guarantee optimal changes with dynamic traffic patterns. It also involves human operators setting rules that may be inefficient. We propose a detailed analysis of port interface capacities, flow classes, interfaces, and scheduling models to generate appropriate policies for reconfiguration.

This paper is an extended version of the workshop paper[33] with the following additions:

(1) Additional background, configuration details, and protocols involved within router queues.

(2) Additional details on the evaluation of traffic policing for ingress queues.

(3) New evaluation results on egress queue shaping with detailed analysis.

(4) New addition of one-hop router configuration and evaluation.

3 Background

In this section, we present the background material in router queues, configuration parameters, and traffic policing/shaping.

3.1 Router queues

A router typically has two types of network components with separate processing planes[4]:

(1) Control plane: This plane maintains a routing table that lists the routes to be taken by a data packet.

Table 1 RL applied to transport network routing and traffic engineering.

Solution space	Related work	Algorithmic technique	Limitation
Intelligent routing	DROM[21], knowledge defined networking[22], multi-agent routing[23], graph based routing[24]	DROM[21] and knowledge defined networking[22] make use of Deep Deterministic Policy Gradient (DDPG) algorithms to change the weights of network links in an SDN framework. Multi-agent routing[23] upgrades Q-routing with multi-agent RL techniques for coordinated routing. Graph based routing[24] develops a model-free RL technique that considers the graph nature of the network topology tailored to the routing problem.	This set of papers makes use of RL within the routing plane of the network. There is no modeling of the internal router/switch queue configurations or machine learning thereof.
Network traffic control	QFLOW[25], LEARNET[26], multi-hop routing[27], SDN RL[28], TCP-DRINC[29]	QFLOW[25] is a platform for RL-based edge network configuration that uses queuing, learning, and scheduling to meet the quatily of experience of video streaming applications. LEARNET[26] makes use of RL for flow control in time-sensitive deterministic networks. In multi-hop routing[27], a distributed model-free solution based on stochastic policy gradient RL was proposed, which aims to minimize the E2E delay by allowing each router to send a packet to the next-hop router according to the learned optimal probability. SDN-based RL techniques[28] exploits the multi-path forwarding of SDN to increase throughput or reduce latency of packet transmission. TCP-DRINC[29] uses an RL-based congestion control to adjust the congestion window size within TCP networks.	This set of traffic engineering techniques makes use of overlay congestion control or SDN-based flow control to improve the throughput, latency, and packet drop of networks. They do not consider the bottlenecks that can arise at the port queue configuration level in underlay networks.
Queue configuration	RL-QN[30], Deep RL[31], queue-learning[32]	In Ref. [30], model-based RL was used to learn the optimal control policy of queueing networks, so that the average job delay is minimized. In Ref. [31], Proximal Policy Optimization (PPO) algorithm was tested on a parallel-server system and large-size multi-class queueing networks. The algorithm consistently generates control policies that outperform heuristics in a variety of load conditions from light to heavy traffic. Queue-learning[32] studies an RL-based service rate control algorithm for providing QoS in tandem queueing networks. The proposed method is capable of guaranteeing probabilistic upper-bounds on the end-to-end delay of the system.	These papers provide a theoretical foundation on applying RL within queueing networks. However, specific nature of protocols used for traffic policing and shaping within routers/switches are not considered.

This may be statically defined or learned based on the traffic mix. The control plane builds the Forwarding Information Base (FIB) that is fed to the forwarding plane.

(2) Forwarding plane: The router forwards data packets between incoming and outgoing port connections. Using information from the packet header, the accurate FIB is used to map the outgoing packet.

Both the switch and router ports have ingress (inbound) queues and egress (outbound) queues.

Figure 1 presents an overview of the ingress and egress queuing operations that typically occur in routers[4, 6]. When a packet arrives at the router, it typically follows the following workflow steps (see Fig. 1):

(1) Prioritization: A packet entering the router port is assigned an internal priority level and internal drop precedence. This is determined based on a class map linked to the packet header information.

(2) Ingress policing: As the packet enters the router, it is subjected to a classification filter. Packets belonging to each class can be rate-limited or marked. The per- class traffic can be treated as follows:

(a) The per-class rate limits may be set. By default, conforming traffic is marked green; exceeding traffic yellow; and violating traffic red[9]. Violating traffic can be marked red (if the violate red command is configured) or dropped immediately (if the violate drop command is configured).

(b) Packets that are not dropped because of rate limiting can have their drop precedence values

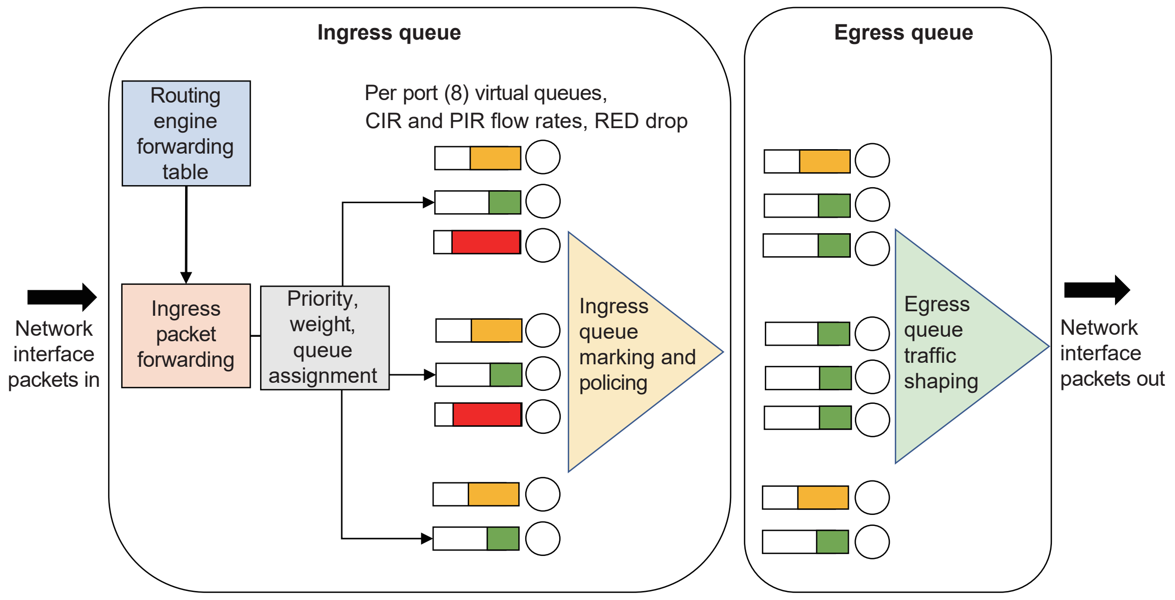

Ingress queue
Egress queue
Routing engine forwarding table
Per port (8) virtual queues, CIR and PIR flow rates, RED drop
Network interface packets in
Ingress packet forwarding
Priority, weight, queue assignment
Ingress queue marking and policing
Egress queue traffic shaping
Network interface packets out

Fig. 1 Router internal queues.

modified.

(3) The router transports the packet to the egress port.

(4) Egress scheduling: Each outgoing packet is assigned to an egress queue based on the destination information from the forwarding table. Egress queues also have associated scheduling parameters, such as rates, depths, and relative weights.

Note: In this paper, we do not target the routing table policy (extensively surveyed in Ref. [14]), but concentrate on configuring the router port queuing disciplines.

3.2 Queuing models

Both ingress and egress queues within the router port may be configured to operate on different queue types. Depending on the combination of flows and QoS requirements, the following queuing models may be used[10]:

−**First-In First-Out (FIFO)**: In FIFO queues, the packets are processed in order of arrival.

−**Priority Queuing (PQ)**: In this form of queuing, packets are assigned priorities with higher priority packets proceed before lower priority packets. The disadvantage is that the lower priority queues may not receive any service.

−**Fair Queue (FQ)**: This queuing discipline ensures that all flows have fair access to the queuing resources. This model prevents bursty traffic from overconsuming resources.

−**Weighted Fair Queuing (WFQ)**: This is a fair queuing discipline that also provides priority scheduling of packets. Higher priority packets are scheduled before a lower priority packet when they arrive at the same time.

−**Class-Based WFQ (CBWFQ)**: This provides multiple use defined queues that may be configured based on either a fixed bandwidth or shared bandwidth.

−**Priority WFQ (PWFQ)**: This provides a strict priority of queues as an extension of CBWFQ.

In most commercial routers[3, 4, 6], the PWFQ model is used. An example configuration is shown below. The first step is to create a PWFQ policy with the bandwidth limits, congestion avoidance map, and policy weights:

```
[local](config)#qos policy policy1 pwfq
[local](config-policy-pwfq)#rate pir 100000 burst
    10000
[local](config-policy-pwfq)#rate cir 50000 burst
    30000
[local](config-policy-pwfq)#congestion-map ca-map-1
[local](config-policy-pwfq)#num-queues 8
[local](config-policy-pwfq)#weight 70
```

This step is followed by specifying queues, and assigning port priorities, weights, maximum rates, and bursts.

```
[local](config-policy-pwfq)#queue 0 priority 0
      strict-priority 0
[local](config-policy-pwfq)#queue 0 rate maximum
      5000 burst 30000
[local](config-policy-pwfq)#queue 1 priority 0
      strict-priority 1
[local](config-policy-pwfq)#queue 1 rate maximum
      10000 burst 30000
[local](config-policy-pwfq)#queue 2 priority 0
      strict-priority 2
[local](config-policy-pwfq)#queue 2 rate maximum
      10000 burst 30000
[local](config-policy-pwfq)#queue 3 priority 0
      strict-priority 3
[local](config-policy-pwfq)#queue 3 rate maximum
      10000 burst 100000
```

We delve into adding additional packet queue scheduling parameters in the next section.

3.3 Packet scheduling in routers

In order to provide DiffServ-based QoS, network routers and switches make use of traffic conditioning[4]. Policing and shaping are two methods that will help reduce congestion by continuously measuring the rate at which data are sent or received.

Policing applies a hard limit to the rate at which traffic arrives or leaves an interface. Packets are either dropped (hard policing) or reclassified (soft policing) if they do not conform to the constraints. Policing is helpful under certain conditions where the neighbouring network could send more traffic than the contract specification. Policing will then enforce the contract, thus protecting the network from being overrun with too much traffic. A policer can be applied in an inbound or outbound direction.

Shaping also defines a limit to the rate at which traffic can be transmitted, but unlike policing, it acts on traffic that has already been granted access to a queue and is awaiting access to transmission resources. Shaping is useful when the neighbouring network is policing or is slower in accepting traffic. Under such conditions a shaper can delay traffic, thus preventing it from getting dropped. Therefore, shaping can be less aggressive than policing and can have fewer negative side effects.

One well-used policing policy is the two-rate three-colour marking[9] (see Fig. 2). It bases the marking on the Committed Information Rate (CIR), i.e., the average traffic rate that a customer is allowed to send into the network, and the Peak Information Rate (PIR),

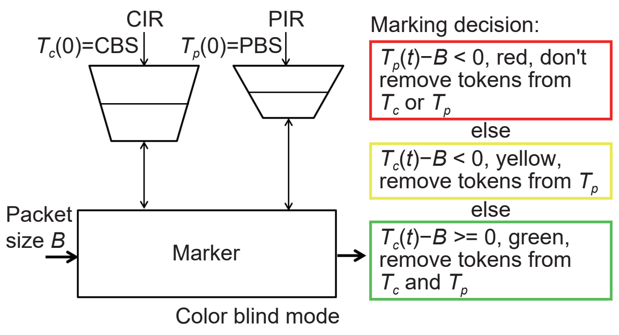

CIR
PIR
$T_c(0)$=CBS
$T_p(0)$=PBS
Marking decision:
$T_p(t)-B < 0$, red, don't remove tokens from T_c or T_p
else
$T_c(t)-B < 0$, yellow, remove tokens from T_p
else
$T_c(t)-B >= 0$, green, remove tokens from T_c and T_p
Packet size B
Marker
Color blind mode

Fig. 2 Two-rate three-color marking, where CBS is the committed burst rate and PBS is the peak burst rate.

i.e., the maximum average sending rate for the customer. Traffic bursts that exceed the CIR but remain under PIR are allowed in the network, but are marked for more aggressive discarding.

Algorithm 1 provides users with three actions for each packet: a conform action, an exceed action, and an optional violate action. If the tokens in the CIR or PIR buckets are exceeded, then tokens may be marked as yellow (exceeding) or red (violating). Users can specify these actions. Exceeding packets can the sent with a decreased priority, and violating packets can be marked for aggressive dropping.

Packet dropping policies: The router scheduler maintains an average queue length or each queue configured for a Random Early Drop (RED)[34]. When a packet is enqueued, the current queue length is weighted into the average queue length based on the average-length exponent in the drop profile. When the average queue length exceeds the minimum threshold, RED begins randomly dropping packets. While the average queue length increases toward the maximum threshold, the RED drops packets with increasing frequency, up to the maximum drop probability. When

Algorithm 1 Two-rate three-colour marking[9]

```
Input: Input packet of size $B$; number of tokens in the CIR bucket $T_c$;
    number of tokens in the PIR bucket $T_p$.
Output: Packet marked with green (conform), yellow (exceed) or red (violate)
    colours.
When a packet with size of $B$ bytes arrives at the interface, perform the following:
if $T_p < B$ then
    Mark packet with violate colour (red);
    Do not decrement $T_p$ bucket;
else if $T_c < B$ then
    Mark packet with exceed colour (yellow);
    Decrement $T_c$ bucket by $B$;
else
    Mark packet with conform colour (green);
    Decrement both $T_c$ and $T_p$ buckets by $B$
```

the average queue length exceeds the maximum drop threshold, all packets are dropped. This is also shown in Fig. 3.

In cases where the queues are congested, packed dropping strategies may be applied[10]:

−**Drop-from-tail**: easy to implement; delayed packets within the queue may “expire”.

−**Drop-from-head**: old packets purge first; good for real time; better for TCP.

−**Random drop**: fair if all sources behave; misbehaving sources are more heavily penalized.

An example profile of dropping packets for congestion avoidance is shown below. The queue depth is used along with the queue average packet size to calculate the effective queue depth. The minimum threshold sets the RED queue occupancy below, whose packets are not dropped.

```
[local](config-congestion-map)#queue 3 depth 200
[local](config-congestion-map)#queue 3 average-packet
        -size 256
[local](config-congestion-map)#queue 3 exponential-
        weight 9
[local](config-congestion-map)#queue 3 red default
        min-threshold 130 max-threshold 200 probability
        10
[local](config-congestion-map)#queue 4 depth 400
[local](config-congestion-map)#queue 4 average-packet
        -size 512
[local](config-congestion-map)#queue 4 exponential-
        weight 9
[local](config-congestion-map)#queue 4 red default
        min-threshold 300 max-threshold 400 probability
        10
```

4 5G slicing scenario

Network slicing in radio access, transport, and core subsystems is necessary to provide DiffServ performance[2]. The use case that is of interest is when various slice requirements are affected by the static configurations of a router port. In this case, the transport layer cannot meet service-level agreement due to congestion at a particular port (other diagnostic tools may be used to detect this). The objective of this work is to automatically reconfigure the port queue to alleviate this bottleneck.

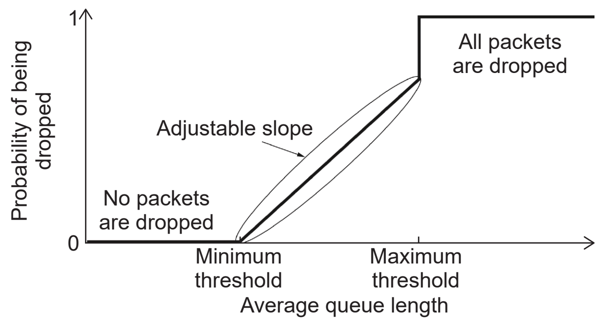


Fig. 3 RED drop probability.

Figure 4 presents an actually observed scenario on Ericsson’s internal network. Due to a statically configured edge router, the end-to-end (E2E) network slice is unable to meet the service-level QoS objectives. Although an alternative would be to re-route the traffic to another port, this is a pathological problem that would entail repeated changes to the slice.

The Ericsson 6675 router is deployed on a live Ericsson testbed network with multiple flows of traffic as shown in Fig. 5. 60 UDP users and 80 TCP users are used to generate a range of traffic pattens over the monitored period. An example of such a port configuration on an Ericsson 6675 router is shown below:

```
PORT 10
QoS policy PWFQ - 1_8 Gbps pwfq
rate PIR 1800000
rate CIR 1800000
num-queues 8
congestion-map WRED
queue 0 priority 0 weight 100
queue 1 priority 0 weight 100
queue 2 priority 0 weight 100
queue 3 priority 0 weight 100
queue 3 rate maximum 900000      (#motor)
queue 4 priority 4 weight 60
queue 5 priority 4 weight 25      (#video)
queue 6 priority 4 weight 10
queue 7 priority 4 weight 5      (#IoT sensor)
```

Due to the priority and weight settings, queues 5 and 7 receive a bulk of the traffic, which results in increased congestion in the respective queues. An example of the traffic generated is shown in Fig. 6. Such a misconfiguration is a pathological problem that requires expert root cause analysis and tuning. In the next section, we study an RL model that can potentially alleviate this issue.

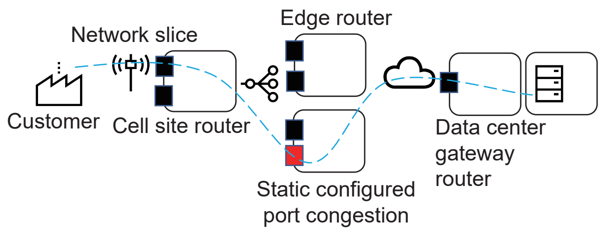
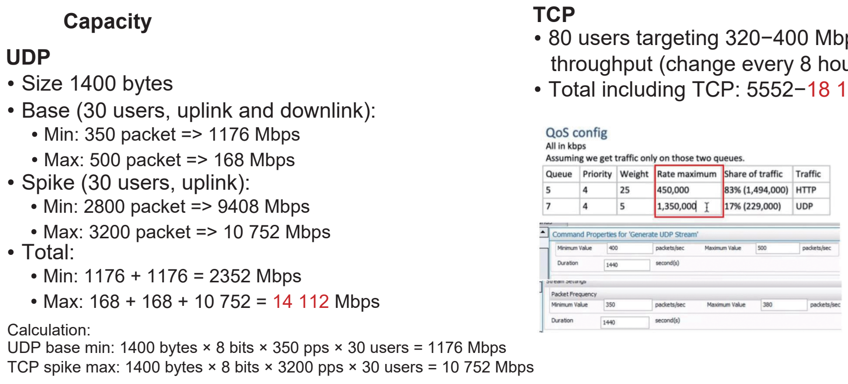


Fig. 4 Bottleneck due to statically configured router.



Capacity

UDP
- Size 1400 bytes
- Base (30 users, uplink and downlink):
  - Min: 350 packet => 1176 Mbps
  - Max: 500 packet => 168 Mbps
- Spike (30 users, uplink):
  - Min: 2800 packet => 9408 Mbps
  - Max: 3200 packet => 10 752 Mbps
- Total:
  - Min: 1176 + 1176 = 2352 Mbps
  - Max: 168 + 168 + 10 752 = 14 112 Mbps

Calculation:
UDP base min: 1400 bytes × 8 bits × 350 pps × 30 users = 1176 Mbps
TCP spike max: 1400 bytes × 8 bits × 3200 pps × 30 users = 10 752 Mbps

TCP
- 80 users targeting 320−400 Mbps throughput (change every 8 hours)
- Total including TCP: 5552−18 112 Mbps

QoS config
All in kbps
Assuming we get traffic only on those two queues.

Queue	Priority	Weight	Rate maximum	Share of traffic	Traffic
5	4	25	450,000	83% (1,494,000)	HTTP
7	4	5	1,350,000	17% (229,000)	UDP

Command Properties for 'Generate UDP Stream'
Minimum Value 400 packets/sec Maximum Value 500 packets/sec
Duration 1440 second(s)

Stream Settings
Packet Frequency
Minimum Value 350 packets/sec Maximum Value 380 packets/sec
Duration 1440 second(s)

Fig. 5 Statically configured router queues.

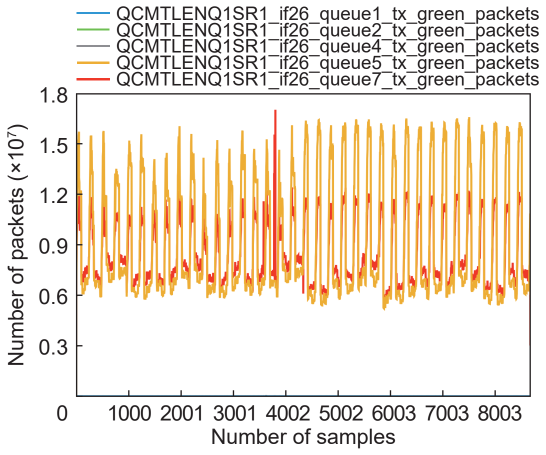


Fig. 6 Observed packet traffic across multiple queues.

5 RL model

To dynamically reconfigure the router port queues, we make use of a trained RL agent to take actions to relieve potential mis-configurations. Figure 7 presents the high level model of our proposed system, which consists of the following:

(1) Queuing network model: Based on the traffic mix and flows, a queuing network model is developed for the ingress and egress queues. The queuing model is used to study the effects of changing multiple parameters, such as queuing discipline, drop rates, and priority.

(2) RL agent: The effect of changes on queue configurations are fed into the RL model for training. It has actions spanning multiple queue configuration parameters. Observations on throughput, latency, and packet drop changes are used to provide rewards to the agent.

(3) Configuration deployment: The policy generated by the RL agent is then deployed on the router port. This can be directly run on the router

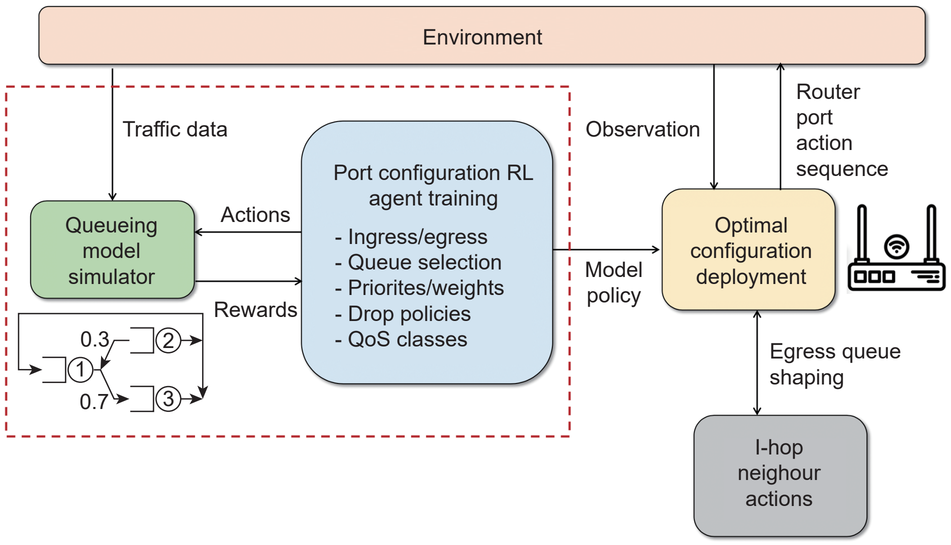


Fig. 7 RL model for port configuration.

operating system or be configured via SDN controllers. Based on the observed traffic mix, the agent's policy is deployed for ingress traffic policing and egress traffic shaping.

(4) 1-hop neighbor configuration: We further include policies that can affect another hop of the router. This reward structure ensures that the egress output does not cause a bottleneck in the next hop.

We provide further details on the model-based RL algorithms and the queuing simulator to train and evaluate this model.

5.1 POMDP

Markov Decision Processes (MDPs) form the theoretical basis for RL. A generalization of MDPs is POMDP[8], where decisions are made with partial observations of the environment. This is a more generic and scalable approach as (i) there is no assumption on agents having full observations of the environment, (ii) the system is resilient to faulty observations, and (iii) heterogeneous observations may be included. This approach is also useful in the case of routers as internal queues are seldom exposed: Through the coarse-grained characteristics of the latency, throughput, and packet drop per flow, the underlying queue characteristics can be derived.

As the agent does not directly know the exact state, it has to derive beliefs based on observations. The observations and rewards are inputs to the agents, which are then used to update current belief states. The advantage of this process is that uncertainty in observations is also included within the model.

Definition 1 A POMDP is a tuple $\langle S,A,\Omega,T,O,R\rangle$:

– S is a finite set of states,

– A is a finite set of actions,

– Ω is a finite set of observations,

– T is a transition function defined as $T: S\times A\times S\rightarrow[0,1]$,

– O is an observation function defined as $O: S\times A\times\Omega\rightarrow[0,1]$,

– R is a reward function defined as $R: S\times A\times S\rightarrow\mathbf{R}$.

Figure 8 provides an overview of the POMDP agent's interaction with its environment. With a set of states $S=\{s^1,\ldots,s^N\}$ and a set of actions $A=\{a^1,\ldots,a^K\}$,

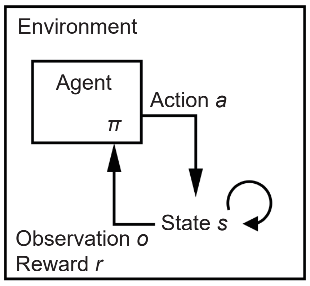


Fig. 8 POMDP agent.

when the agent takes an action a in state s, the environment transitions to state s' according to the transition function $T(s,a,s')$. The agent receives a reward $R(s,a,s')$ as a result of this action. The set of observations $\Omega=\{o^1,\ldots,o^M\}$ represent all possible sensor readings the agent can receive. The observation is dependent on the function $O: S\times A\times\Omega\rightarrow[0,1]$, with an independently tracked probability of observations.

The POMDP agent maps observations to states using the belief vector $b(s)$. The belief b is a probability distribution over S, which forms a Markovian signal for the planning task. Each POMDP problem assumes an initial belief b_0, which for instance can be set to a uniform distribution over all states (representing equal probability of stating in any of the states). Every time the agent takes action a and observes o, its belief is updated by Bayes' rule,

$$b_{ao}(s')=\frac{p(o|s',a)}{p(o|b,a)}\sum_{s\in S}p(s'|s,a)b(s) \qquad (1)$$

where $p(o|s',a)$ and $p(s'|s,a)$ are defined by parameters O and T, respectively, and the normalizing constant $p(o|b,a)$ is defined as

$$p(o|b,a)=\sum_{s'\in S}p(o|s',a)\sum_{s\in S}p(s'|s,a)b(s) \qquad (2)$$

The goal of the agent is to choose actions that fulfill its task as well as possible, thus generating an optimal policy. In POMDPs, an optimal policy $\pi^*(b)$ maps beliefs to actions over a continuous set of probability distributions over S. A policy π can be characterized by a value function $V^\pi(b)$, which is defined as the expected future discounted reward that the agent can gather by following π starting from belief b,

$$V^\pi(b)=E_\pi\left[\sum_{t=0}^{h}\gamma^t\sum_{s\in S}R(s,\pi(b_t))b_t(s)|b_0=b\right] \qquad (3)$$

A policy π that maximizes V^{π} is called an optimal policy π^*. It specifies for each b the optimal action to execute at the current step, assuming that the agent will also act optimally at future time steps. γ is the discount factor that trades off current rewards to future expected rewards. The value of the optimal policy is defined by the optimal value function V^*, that is generated using the Bellman backup operator[8],

$$V^*(b) = \max_{a\in A}\left[\sum_{s\in S} R(s,a)b(s) + \gamma\sum_{o\in O} p(o|b,a)V^*(b_{ao})\right] \quad (4)$$

Generating an exact solution for the exhaustive set of belief points is computationally expensive. A counter technique is to generate an approximate solution with only a finite set of belief points[8]. In this work, we make use of the SARSOP algorithm[35] that uses point based approximations to solve POMDPs.

A POMDP policy $\pi : \mathcal{B} \rightarrow A$ maps a belief $b \in \mathcal{B}$ to a prescribed action $a \in A$. The value function associated with the optimal policy π^* can be approximated arbitrarily closely by a convex and piecewise-linear function,

$$V(b) = \max_{\alpha\in\Gamma}(\alpha \cdot b) \quad (5)$$

where Γ is a finite set of vectors called α-vectors, b is the discrete vector representation of a belief, and $\alpha \cdot b$ is the inner product of vectors α-vector and b. Each α-vector is associated with an action. The policy can be executed by selecting the action corresponding to the best α-vector at the current belief. As provided in Ref. [35], the value update function for the α-vectors is given by the following:

$$\alpha_a(s) \leftarrow R(s,a) + \gamma\sum_{o,s'} T(s,a,s')O(s',a,o)\alpha_{a,o}(s') \quad (6)$$

We make use of the POMDP format (http://www.pomdp.org/code/pomdp-file-grammar.html) to specify a subset of states, actions, and observations that are relevant for the router port configuration. The underlying states are based on (unobservable) queue depths and queue utilization, which can affect the marking of an incoming packet, and classified as low, mid, or high. A subset of this is seen in Table 2.

With the changes in parameters via actions on a particular queue, the rest of the queues may be positively or negatively affected. These are seen as system observations used to make further configuration changes. To evaluate the probabilities of these actions, we make use of exhaustive queuing network model simulations, as described in the next section.

Table 2 States, action, and observations in POMDP format.

```
discount: 0.95
values: reward

states:
#0  Q0-3_low_Q4_low_Q5_low_Q7_low
#1  Q0-3_low_Q4_low_Q5_low_Q7_high
#2  Q0-3_low_Q4_low_Q5_high_Q7_low
#3  Q0-3_low_Q4_low_Q5_high_Q7_high
#4  Q0-3_low_Q4_high_Q5_low_Q7_low
#5  Q0-3_low_Q4_high_Q5_low_Q7_high
#6  Q0-3_low_Q4_high_Q5_high_Q7_low
#7  Q0-3_low_Q4_high_Q5_high_Q7_high
#8  Q0-3_high_Q4_high_Q5_high_Q7_high
#9  Q0-3_low_Q4_high_Q5_low_BW_Q7_high

actions:
# 0  Q5_weight_increase
# 1  Q5_weight_decrease
# 2  Q5_bandwidth_limit_decrease
# 3  Q5_bandwidth_limit_increase
# 4  Q5_interface_increase
# 5  Q5_interface_decrease
# 6  Q5_PFWQ_FIFO
# 7  Q5_FIFO_PFWQ
# 8  Q5_RED_packet_drop_high
# 9  Q5_RED_packet_drop_low

observations:
Q5_residence_time_increase
Q5_residence_time_decrease
Q5_throughput_increase
Q5_throughput_decrease
Q5_queue_drop_increase
Q5_queue_drop_decrease
Q7_residence_time_increase
Q7_residence_time_decrease
Q7_throughput_increase
Q7_throughput_decrease
Q7_queue_drop_increase
Q7_queue_drop_decrease
system_residence_time_increase
system_residence_time_decrease
system_throughput_increase
system_throughput_decrease
system_queue_drop_increase
system_queue_drop_decrease
```

5.2 Queue evaluation parameters

In order to train the POMDP model, we need a robust quqeueing modeling tool to observe changes due to configurations. We use the Java Modeling Tools (JMT)[36] queuing network simulator to model the internals of the router queue and study configuration changes as presented in the case study.

Figure 9 presents the router port ingress and egress closed queuing models in JMT. Here we consider multiple classes of customer flows (UDP, TCP, and VoIP), that must be simulated in the queuing network. By making use of the workload intensity parameter N, the average number of jobs (customers) in the flow execution is included. Priorities for processing each class of flows are specified. The arrival rate of packets

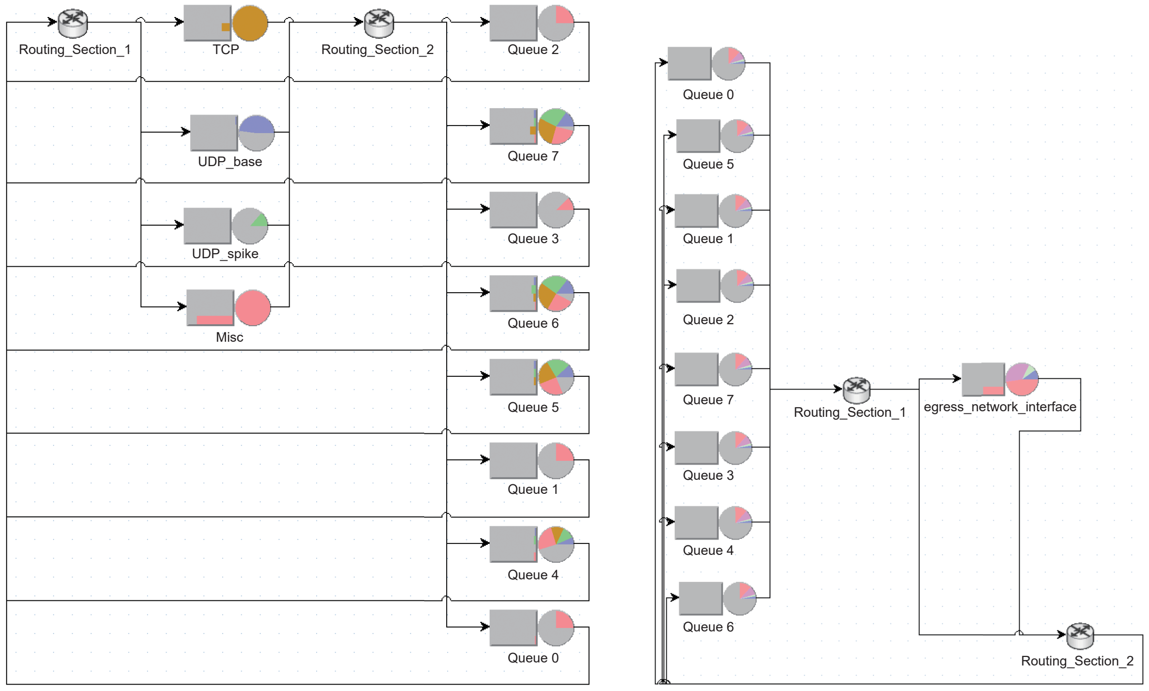


Fig. 9 Router ingress (left) and egress (right) queueing models within JMT.

is specified by a Poisson process with the service time (processing time per visit of a station).

In order to implement the routing priorities and PWFQ specifications, certain special parameters in JMT are used:

(1) Queue station: The queue station in JMT allows users to specify queues, queue capacity (finite or infinite), and scheduling policy.

(2) Queue policy: Myltiple ququing policies may be evaluated, including FIFO, FQ, and PWFQ.

(3) Routing section: Various routing algorithms, such as round robin, random, or probabilistic models, may be applied to route packets between queuing stations.

We make use of these aspects to simulate the use case as specified in Section 4.

The advantage of using the queuing network simulators is the ability to collect various performance indices: (i) **residence time:** time spent by a packet waiting and receiving service at a queuing station; (ii) **drop rate**: rate at which packets are dropped from the system; (iii) **throughput:** the number of packets processed in a time unit; and (iv) **utilization (of a station)**: percentage of time that a station is used evaluated over all the simulation runs.

5.3 Transforming queuing metrics to transition probabilities

As we now have outputs from the queuing network models, the output observations and rewards must be effectively translated into the transition probabilities. For this, we make use of some of the fundamental queuing laws[37].

Definition 2 Service demand law: The total average service time required by a customer at resource i, denoted by D_i,

$$D_i = \frac{U_i}{X} \tag{7}$$

where U_i is the utilization of resource i and X is the system throughput.

Definition 3 Little's law: It provides the relationship among the number of users in a system, throughput, and response time. If there are N users in the system, each with think time Z (time waiting between interactions with the system), and system processes at the throughput rate X producing a response time R, then the following relationship applies:

$$N = X \cdot (R + Z) \tag{8}$$

Definition 4 Queue length: Given the utilization U_i at a given station, the queue length q_i may be calculated by

$$q_i = \frac{U_i}{1 - U_i} \tag{9}$$

As the queuing length and utilization of each queue can be related, we make use of the following thresholds to relate the two-rate three-color marking and the RED packet drop probabilities for individual queues,

$$Mark(q,U) = \begin{cases} q_{low}, & \text{if } U \leqslant 0.5; \\ q_{mid}, & \text{if } 0.5 < U \leqslant 0.7; \\ q_{high}, & \text{otherwise} \end{cases} \tag{10}$$

From the observed changes in the utilization of a queue (correspondnigly the queue length), we would like to derive the conditional probabilities of the change among q_{low}, q_{mid}, and q_{high}. For instance, the probability of change from states q_{mid} to q_{low} given the action weight, the increase at state q_{mid} must be generated. Note that these thresholds may vary depending on the use case and queue deterioration thresholds.

Algorithm 2 provides this transformation from steady state queue observations to state and observation transition probabilities, which are required for POMDP models. For instance, if in state `Q1_low`, the action `weight_decrease` produces an observed utilization increase of 5%. Taking the thresholds in Eq. (10) results in the transition probabilities,

$$P(\texttt{Q1\_low}|\texttt{weight\_decrease in Q1\_low}) = 0.9,$$

$$P(\texttt{Q1\_mid}|\texttt{weight\_decrease in Q1\_low}) = 0.1.$$

This is because, in only 10% of cases with utilization in the range [0, 0.5], a 0.05 increase will result in

Algorithm 2 State transition and observation probabilities elicitation

```
Input: Queuing network with a set of states $S$, configuration change actions $A$, corresponding utilization change $\Delta U$, and corresponding observation vector (residence time, throughput, drop packets) change $\Delta O_i$, $i = 1, \ldots, k$.
Output: State and observation transition probabilities.
for each state $s \in S$ that covers all joint queues states (green, yellow, red) do
    for each action $a \in A$ do
        Perform action $a$ in state $s$;
        Determine the change in utilization $\Delta U$ for each $q \in Q$ in state $s$;
        Determine the conditional probability for each state $s$: $P$(state change | $\Delta U$);
        Record state transition probabilities after normalization;
    for each action $a \in A$ do
        Perform action $a$ in state $s$;
        for each for each observation category, $i = 1, \ldots, k$, do
            Determine the change in observation $\Delta O_i$ for each $q \in Q$ in state $s$;
            Determine the conditional probability for each observation change: $P$ (observation change|$\Delta O_i$);
            Record observation transition probabilities after normalization;
```

utilization in the range [0.51, 0.7]. This may be correspondingly marked to queue lengths and packet color markings.

Algorithm 2 is used in conjunction with the SARSOP value update function in Eq. (6) to generate optimal policies.

6 Automated configuration evaluation

In this section, we evaluate the efficacy of the model based RL technique on the traffic policing and shaping of the ingress/egress router port queues.

6.1 Ingress queue policing

In order to study the ingress queue policing, we make use of the traffic setting configuration presented in Section 4 in conjunction with the queuing model in Fig. 9. The output of a queuing simulation over the misconfigured router is shown in Fig. 10. As expected, we see increased utilization, queue length, and residence times for queue 7, despite quques 0−3 having only 25% utilization (Fig.10a). The objective is to generate a policy that would alleviate this load, while maintaining the priorities of individual flows within the system.

The first step is to collect the statistics of changes in the observed metrics as a result of configuration changes. We make use of the JMT simulator to study the improvements/deteriorations due to configuration changes, as shown in Table 3 . The following configurations are studied:

(1) Queue 5 weight increases by 10;

(2) Queue 5 weight decreases by 10;

(3) Queue 5 bandwidth increases by 50%;

(4) Queue 5 bandwidth decreases by 50%;

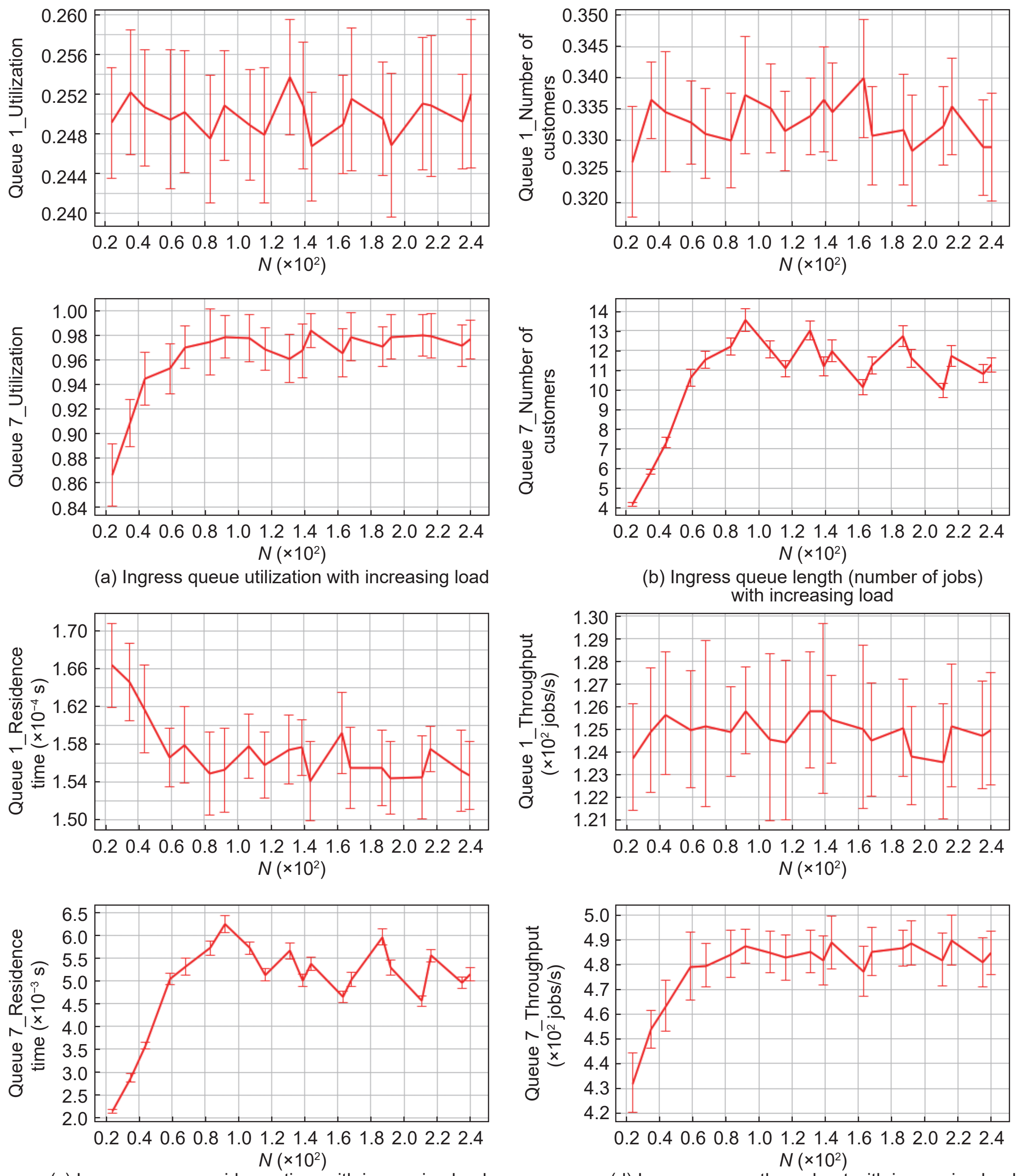
(5) Queues 5 and 7 virtual interfaces increase by 1;

(6) Queue changes from PWFQ to FCFS;

(7) Queue RED packet drop increases.

While we have primarily concentrated on changing the configuration of one queue, this can similarly be extended to combinations of various ingress queue configurations.

Once we have the steady-state metrics for configuration changes, they can be inputted as transition and observation probabilities as described in



(a) Ingress queue utilization with increasing load

(b) Ingress queue length (number of jobs) with increasing load

(c) Ingress queue residence time with increasing load

(d) Ingress queue throughput with increasing load

Fig. 10 Ingress queue with misconfigured router.

Algorithm 2. A snippet of the POMDP file format is shown in Table 4 . Note that we map the same observation to multiple possible underlying states (green, yellow, and red) of the individual queues. This method is realistic as routers do not expose individual queue utilization rather than the global metrics, such as throughput, packet drop rates, and latencies. The expert or RL model must make use of these observations to infer the appropriate configurations that would alleviate the bottleneck queue. We make use of the POMDP model, which has uncertainty built in to estimate the appropriate configuration. In a simpler case, this can be converted to a conventional MDP by mapping each observation to a particular state of the router/queue.

The SARSOP solver[36] is used to generate a policy

Table 3 Observed metrics with ingress queue configuration changes.

Configuration	Queue	Metric			
		Throughput (jobs/s)	Queue length	Residence time (s)	Utilization ratio
Baseline	Q0	124.3	0.33	0.000 70	0.240
	Q1	124.3	0.33	0.000 70	0.240
	Q2	124.3	0.33	0.000 70	0.240
	Q3	124.3	0.14	0.000 34	0.120
	Q4	281.0	1.28	0.003 00	0.560
	Q5	419.0	5.00	0.012 00	0.830
	Q6	477.0	19.40	0.046 00	0.950
	Q7	497.0	98.90	0.230 00	0.990
	UDP-base	353.0	2.22	0.100 00	0.700
	UDP-spike	379.8	0.13	0.000 32	0.110
	TCP	445.0	44.00	0.140 00	0.990
	Misc	995.0	66.90	0.160 00	0.990
1	Q0	124.3	0.33	0.000 86	0.240
	Q1	124.3	0.33	0.000 86	0.240
	Q2	124.3	0.33	0.000 86	0.240
	Q3	124.3	0.14	0.000 37	0.120
	Q4	281.0	1.28	0.003 00	0.560
	Q5	379.0	3.11	0.008 20	0.750
	Q6	477.0	19.40	0.050 00	0.950
	Q7	497.0	103.90	0.270 00	0.990
	UDP-base	336.0	1.90	0.005 10	0.670
	UDP-spike	358.0	0.13	0.000 33	0.110
	TCP	445.0	42.70	0.110 00	0.990
	Misc	995.0	66.90	0.160 00	0.990
2	Q0	124.3	0.330	0.000 72	0.240
	Q1	124.3	0.33	0.000 72	0.240
	Q2	124.3	0.33	0.000 72	0.240
	Q3	124.3	0.14	0.000 31	0.120
	Q4	281.0	1.28	0.002 80	0.560
	Q5	457.0	10.30	0.022 50	0.910
	Q6	477.0	19.10	0.040 00	0.950
	Q7	497.0	91.00	0.190 00	0.990
	UDP-base	368.0	2.60	0.005 10	0.730
	UDP-spike	401.0	0.14	0.000 31	0.120
	TCP	445.0	46.60	0.100 00	0.990
	Misc	995.0	67.60	0.140 00	0.990
3	Q0	101.6	0.25	0.001 20	0.200
	Q1	101.6	0.25	0.001 20	0.200
	Q2	101.6	0.25	0.001 20	0.200
	Q3	101.6	0.11	0.000 51	0.100
	Q4	164.0	0.48	0.002 00	0.320
	Q5	218.0	0.27	0.001 30	0.210
	Q6	242.0	0.93	0.004 30	0.480
	Q7	249.0	231.00	1.000 00	0.990
	UDP-base	100.0	0.25	0.100 00	0.200
	UDP-spike	101.0	0.03	0.000 15	0.030
	TCP	266.0	1.40	0.006 50	0.590
	Misc	812.0	4.16	0.001 90	0.820

(To be continued)

Table 3 Observed metrics with ingress queue configuration changes. (Continued)

Configuration	Queue	Metric			
		Throughput (jobs/s)	Queue length	Residence time (s)	Utilization ratio
4	Q0	101.6	0.25	0.001 20	0.200
	Q1	101.6	0.25	0.001 20	0.200
	Q2	101.6	0.25	0.001 20	0.200
	Q3	101.6	0.11	0.000 46	0.100
	Q4	181.0	0.56	0.002 30	0.360
	Q5	249.0	229.00	0.920 00	0.990
	Q6	279.0	1.26	0.005 00	0.560
	Q7	289.0	0.40	0.001 60	0.290
	UDP-base	128.0	0.34	0.001 40	0.260
	UDP-spike	129.0	0.04	0.000 17	0.040
	TCP	333.0	2.75	0.011 00	0.740
	Misc	818.0	4.30	0.017 20	0.820
5	Q0	124.3	0.33	0.006 00	0.240
	Q1	124.3	0.33	0.006 00	0.240
	Q2	124.3	0.33	0.006 00	0.240
	Q3	124.3	0.14	0.002 60	0.120
	Q4	290.0	1.37	0.018 00	0.580
	Q5	436.0	1.20	0.014 00	0.870
	Q6	498.0	114.00	1.310 00	0.990
	Q7	519.0	1.50	0.018 00	1.000
	UDP-base	380.0	2.88	0.026 50	0.700
	UDP-spike	419.0	0.13	0.001 40	0.110
	TCP	445.0	48.00	0.440 00	0.990
	Misc	995.0	66.90	1.262 50	0.990
6	Q0	452.0	8.80	0.195 00	0.900
	Q1	452.0	8.80	0.195 00	0.900
	Q2	452.0	8.80	0.195 00	0.900
	Q3	452.0	0.82	0.001 80	0.450
	Q4	452.0	8.80	0.195 00	0.900
	Q5	452.0	8.80	0.195 00	0.900
	Q6	452.0	8.80	0.195 00	0.900
	Q7	452.0	8.80	0.195 00	0.900
	UDP-base	493.0	21.42	0.047 30	0.980
	UDP-spike	1 681.0	1.00	0.002 00	0.520
	TCP	449.0	72.00	0.160 00	0.990
	Misc	995.0	82.00	0.180 00	0.990
7	Q0	224.0	0.800	0.003 60	0.450
	Q1	224.0	0.800	0.003 60	0.450
	Q2	224.0	0.80	0.003 60	0.450
	Q3	224.0	0.28	0.001 30	0.220
	Q4	224.0	0.81	0.003 60	0.450
	Q5	224.0	0.81	0.003 60	0.450
	Q6	224.0	0.81	0.003 60	0.450
	Q7	224.0	0.81	0.003 60	0.450
	UDP-base	449.0	8.60	0.030 00	0.890
	UDP-spike	449.0	16.00	0.990 00	0.140
	TCP	449.0	224.00	0.000 73	0.990
	Misc	449.0	0.81	0.030 00	0.450

Table 4 Transitions, observations, and rewards in POMDP.

```
#Transition Probabilities
T: Q5_bandwidth_limit_increase
0.77 0.0 0.23 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.66 0.0 0.0 0.0 0.34 0.0 0.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.66 0.0 0.0 0.0 0.34 0.0 0.0 0.0
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.74 0.0 0.26 0.0 0.0 0.0 0.0 0.0 0.0 0.0


T:  Q5_interface_increase
0.95 0.0 0.05 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.95 0.0 0.05 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0

# Observations
O: Q5_weight_decrease : * :
    Q5_residence_time_increase  0.125
O: Q5_weight_decrease : * :
    Q5_residence_time_decrease  0.0
O: Q5_weight_decrease : * : Q5_throughput_increase  0
    .125
O: Q5_weight_decrease : * : Q5_throughput_decrease  0
    .0
O: Q5_weight_decrease : * : Q5_queue_drop_increase  0
    .125
O: Q5_weight_decrease : * : Q5_queue_drop_decrease  0
    .0
O: Q5_weight_decrease : * :
    system_residence_time_increase  0.0
O: Q5_weight_decrease : * :
    system_residence_time_decrease  0.125
O: Q5_weight_decrease : * :
    system_throughput_increase  0.125
O: Q5_weight_decrease : * :
    system_throughput_decrease  0.0
O: Q5_weight_decrease : * :
    system_queue_drop_increase  0.125
O: Q5_weight_decrease : * :
    system_queue_drop_decrease  0.0

# Rewards
R: Q5_bandwidth_limit_decrease : Q0-3
    _green_Q4_red_Q5_green_Q7_green : * : * -20
R: Q5_bandwidth_limit_decrease : Q0-3
    _green_Q4_red_Q5_green_Q7_red : * : * -20
R: Q5_bandwidth_limit_decrease : Q0-3
    _green_Q4_red_Q5_red_Q7_green : * : * -20
R: Q5_bandwidth_limit_decrease : Q0-3
    _green_Q4_red_Q5_red_Q7_red : * : * -20
R: Q5_bandwidth_limit_decrease : Q0-3
    _red_Q4_red_Q5_red_Q7_red : * : * -20
R: Q5_bandwidth_limit_decrease : Q0-3
    _green_Q4_red_Q5_green_BW_Q7_red : * : * -100
```

that can appropriately reconfigure the system. Our rewards are configured to reward improvements in the observed throughput, residence times, and packet drop rates. The policy graph generated is shown in Fig. 11 (initialized to a statically configured state) with a simulation output of 1000 Monte Carlo runs, as shown in Table 5.

The last step is to input this policy into the queuing network model to observe improvements. The output of the RL algorithm is encoded in the configuration commit. Observations are tried out every 10 minutes

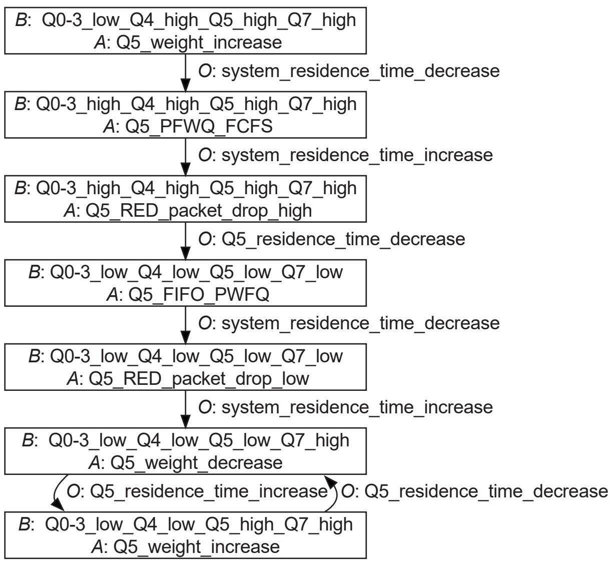

B: Q0-3_low_Q4_high_Q5_high_Q7_high
A: Q5_weight_increase
O: system_residence_time_decrease
B: Q0-3_high_Q4_high_Q5_high_Q7_high
A: Q5_PFWQ_FCFS
O: system_residence_time_increase
B: Q0-3_high_Q4_high_Q5_high_Q7_high
A: Q5_RED_packet_drop_high
O: Q5_residence_time_decrease
B: Q0-3_low_Q4_low_Q5_low_Q7_low
A: Q5_FIFO_PWFQ
O: system_residence_time_decrease
B: Q0-3_low_Q4_low_Q5_low_Q7_low
A: Q5_RED_packet_drop_low
O: system_residence_time_increase
B: Q0-3_low_Q4_low_Q5_low_Q7_high
A: Q5_weight_decrease
O: Q5_residence_time_increase
O: Q5_residence_time_decrease
B: Q0-3_low_Q4_low_Q5_high_Q7_high
A: Q5_weight_increase

Fig. 11 Ingress queue configuration policy graph.

Table 5 Simulation run and rewards for ingress queue policy.

```
---------------------------------------------------------------
 Time    |#Trial |#Backup |LBound     |UBound
 |Precision  |#Alphas |#Beliefs
---------------------------------------------------------------
 204.1    311      23369     146.172     146.999
 0.8266     275      9044
---------------------------------------------------------------
-------------------------------------------
 #Simulations  | Exp Total Reward
-------------------------------------------
 10               154.573
 20               140.178
 30               137.111
 40               137.309
 50               134.383
 60               130.251
 70               136.91
 80               139.373
 90               135.104
 100              138.198
-------------------------------------------
---------------------------------------------------------------
 #Simulations  | Exp Total Reward | 95\%
 Confidence Interval
---------------------------------------------------------------
 100         138.198        (124.735, 151.661)
---------------------------------------------------------------
```

until a steady state is reached. The outputs of the estimated queue utilization are presented in Fig. 12. The initial configuration has two queues in the red packet marking zone. After successive policy actions, all the queue utilization is brought below the 70% level. This finding demonstrates the efficacy of the RL model in ingress traffic policing.

6.2 Egress queue traffic shaping

It is also important to perform the effective shaping of egress queues to prevent packet drops at the output of routers (Fig. 9). Figure 13 provides the output of the queuing model with an initialized queue configuration.

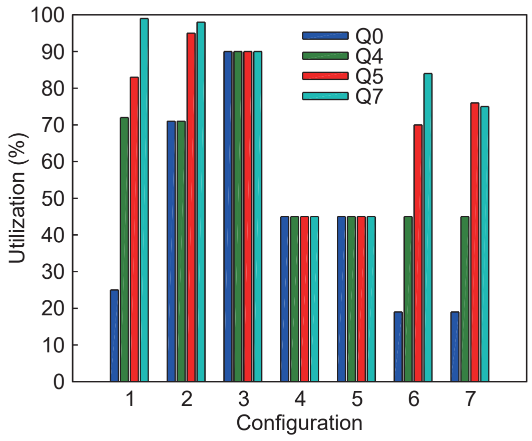


Fig. 12 Observed improvements due to automated configuration.

The results show that the egress queue reaches bottleneck capacity, whereas queues 0−7 still have capacity.

In order to reconfigure the system, we once again study the effect of configuration changes on the observed metrics. Table 6 reflects the observed changes due to the following egress queue scheduling configurations:

(1) Egress queue input routing percentage decreases by 20%;

(2) Egress queue input routing percentage decreases by 20%;

(3) Egress queue bandwidth limit increases by 50%;

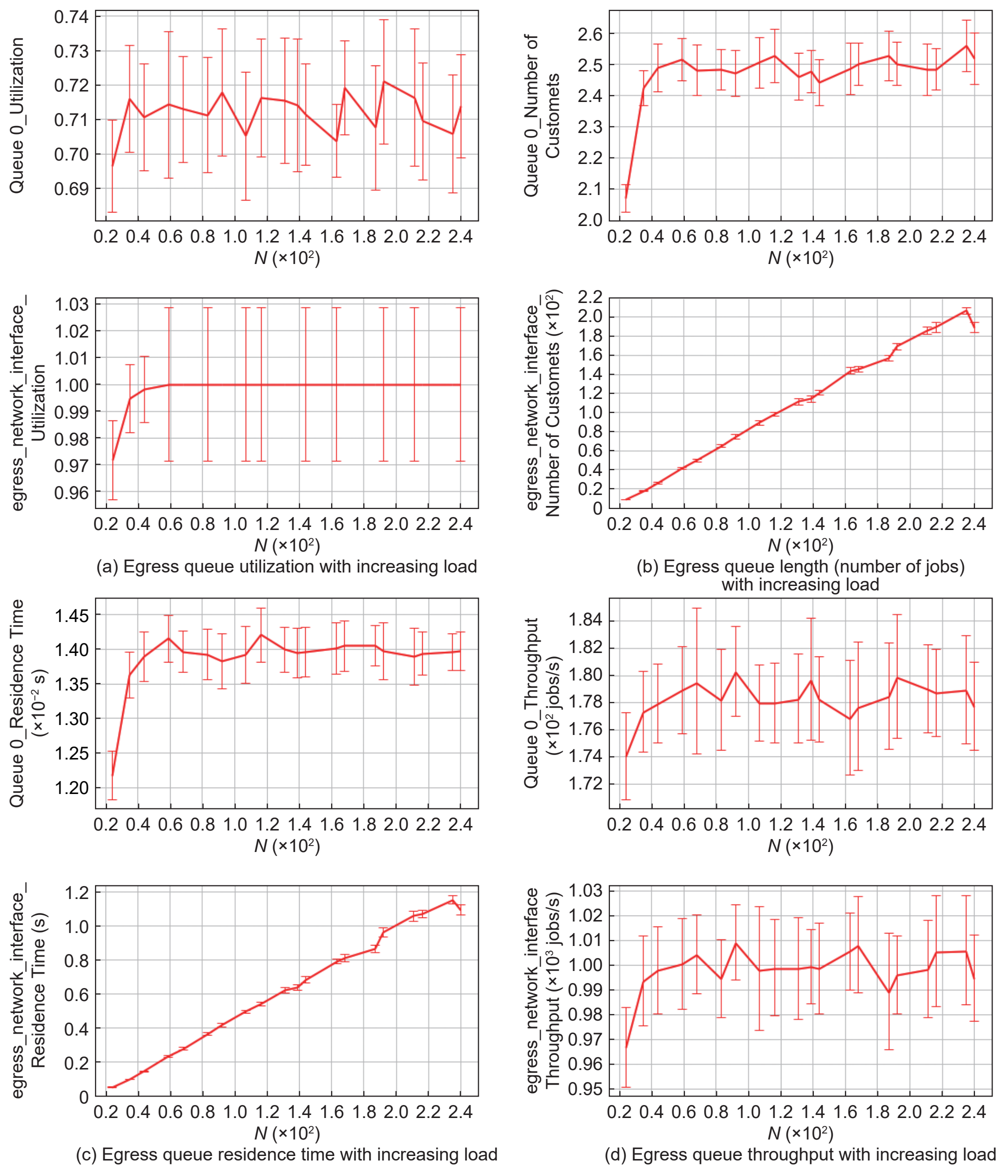


Fig. 13 Statically configured egress queue.

Table 6 Configuration changes for egress queues.

Configuration	Queue	Metric			
		Throughput (jobs/s)	Queue length	Residence time (s)	Utilization ratio
Baseline	Q0−Q7	249	1.00	0.004	0.50
	Egress	999	232.00	0.928	1.00
1	Q0−Q7	485	29.50	0.060	0.97
	Egress	786	3.43	0.007	0.77
2	Q0−Q7	416	4.80	0.010	0.83
	Egress	999	201.00	0.480	0.99
3	Q0−Q7	97	30.00	0.300	0.97
	Egress	388	0.60	0.006	0.38
4	Q0−Q7	249	0.33	0.001	0.25
	Egress	999	237.00	0.950	1.00

(4) Egress queue bandwidth limit decreases by 50%.

This model is once again translated into the POMDP format with transition and observation probabilities. Using the SARSOP solver, the optimal policy is generated, as shown in Fig. 14, with the Monte Carlo runs shown in Table 7.

Figure 15 demonstrates the effectiveness of the egress traffic shaping process. We notice that through repeated configuration changes, the utilization of both queues drops to the "green" states.

In Fig. 15 , we also provide a result with reward changes for the one-hop router configuration. This method is needed in cases where the egress queue from one router can potentially cause bottlenecks in subsequent routers by deploying excessive traffic to the network. We weave into this model the rewards when

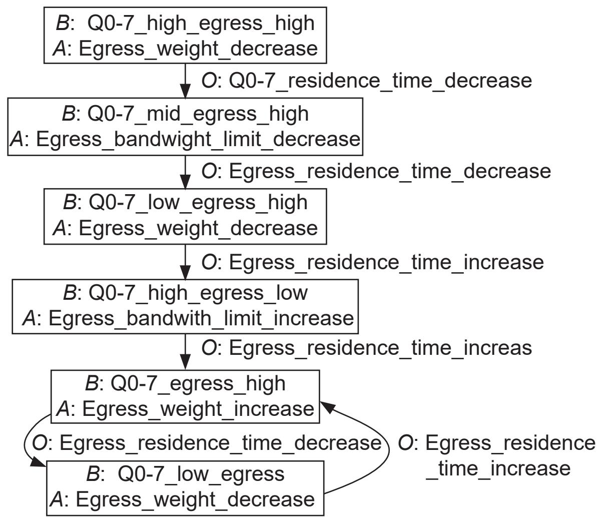

B: Q0-7_high_egress_high
A: Egress_weight_decrease
O: Q0-7_residence_time_decrease
B: Q0-7_mid_egress_high
A: Egress_bandwight_limit_decrease
O: Egress_residence_time_decrease
B: Q0-7_low_egress_high
A: Egress_weight_decrease
O: Egress_residence_time_increase
B: Q0-7_high_egress_low
A: Egress_bandwith_limit_increase
O: Egress_residence_time_increas
B: Q0-7_egress_high
A: Egress_weight_increase
O: Egress_residence_time_decrease
O: Egress_residence _time_increase
B: Q0-7_low_egress
A: Egress_weight_decrease

Fig. 14 Egress queue configuration policy graph.

Table 7 Simulation rewards for egress.

```
--------------------------------------------------------------------------
 Time     |#Trial  |#Backup  |LBound      |UBound
          |Precision   |#Alphas  |#Beliefs
--------------------------------------------------------------------------
 0.21       82        3491       208.902     208.902
        0.000816354  13          602
--------------------------------------------------------------------------
---------------------------------------------
 #Simulations    | Exp  Total  Reward
---------------------------------------------
 10                   196.246
 20                   201.31
 30                   204.546
 40                   202.937
 50                   205.736
 60                   204.843
 70                   204.305
 80                   205.793
 90                   206.368
 100                  206.478
---------------------------------
--------------------------------------------------------------------------
 #Simulations    | Exp  Total  Reward  |  95\%  Confidence
          Interval
--------------------------------------------------------------------------
 100                  206.478                 (203.317,  209.64)
--------------------------------------------------------------------------
```

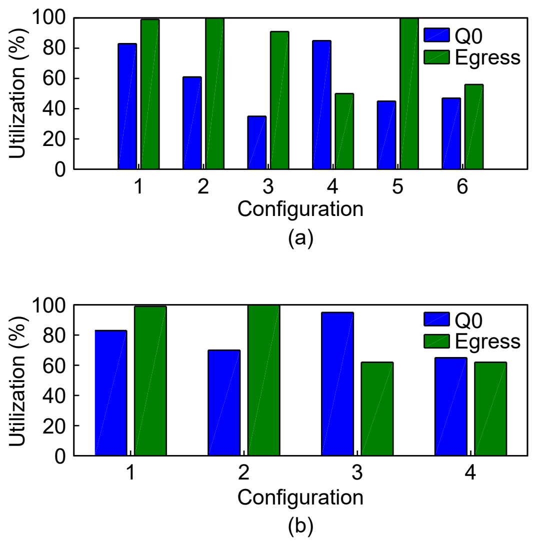


Fig. 15 Egress queue configurations with (a) single router and (b) one-hop considered.

there can be deterioration of the next-hop router policies and can effectively configure the router to mitigate this action.

6.3 Effect of traffic change

Another factor to estimate is the robustness of the system with network traffic changes. We make use of the same POMDP model with a new traffic pattern as shown in Fig. 16 . Keeping the same transition and observation probabilities as the initial model, the POMDP policy is generated with this new initial condition. The sequence of configuration changes is shown in Fig. 17 . The findings demonstrate that the system is robust enough to handle variations in traffic patterns and ensures proper configuration of ingress/egress queues.

7 Conclusion

The emergence of 5G network slicing has mandated the need for accurate configuration of routers and

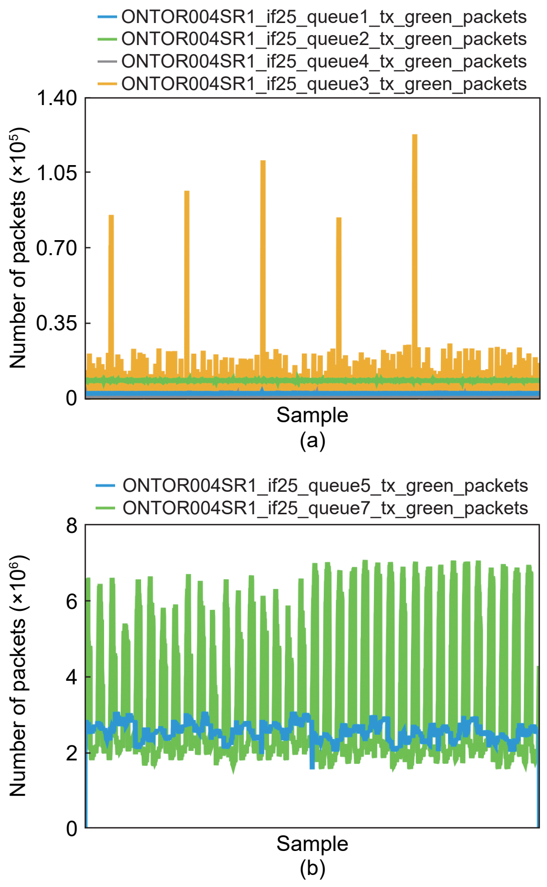


Fig. 16 Router queue processing traffic.

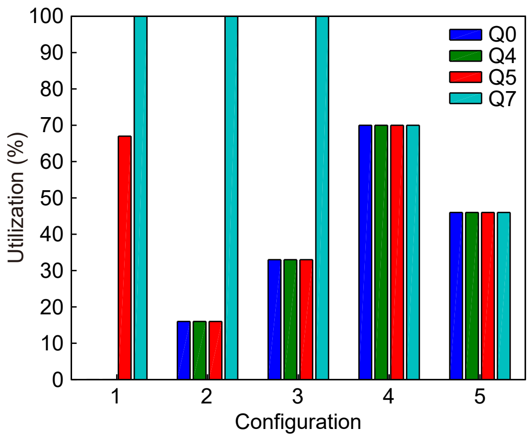


Fig. 17 Ingress queue reconfiguration.

switches to deliver optimal performance. Currently, routers and switches have static configurations, which are applied to multiple traffic mixes. This characteristic can quickly turn suboptimal, requiring trial-and-error changes by an expert. In this paper, we explore alternative RL-based models to automate configuration changes in router ports. This method is shown to be effective in traffic policing, traffic shaping and coordinated configuration changes within routers. Its performance is demonstrated over a realistic use case involving routers for 5G network slicing.

In this work, we have made use of queuing network models to train the RL model. However it would be more realistic to train it on logged network datasets that stem from Ericsson, Juniper, and Cisco router deployments. In addition, there are plans to include transfer learning to re-use policies across a sequence of routers for effective 5G slicing. This initiative may also include multi-agent RL techniques to coordinate multiple configuration agents across large networks.

References

[1] ETSI, System architecture for the 5G system, 3GPP TS 23.501, version 15.3. 0, 2018.

[2] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, Network slicing in 5G: Survey and challenges, *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, 2017.

[3] Ericsson AB, Router 6675, Technical specifications, 2019.

[4] D. R. Hanks Jr. and H. Reynolds, *Juniper MX Series*. Sebastopol, CA, USA: O'Reilly Media, 2012.

[5] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, Software-defined networking: A comprehensive survey, *Proc. IEEE*,

vol. 103, no. 1, pp. 14–76, 2015.

[6] Cisco Systems, Quality of Service (QoS) configuration guide, Cisco IOS, 2018.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning - An Introduction*. 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[8] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, Planning and acting in partially observable stochastic domains, *Artif. Intell.* , vol. 101, nos. 1&2, pp. 99–134, 1998.

[9] Cisco Systems, QoS: Color-aware policer, Cisco IOS documentation, 2005.

[10] C. Semeria, Supporting differentiated service classes: Queue scheduling disciplines, *Juniper Networks Whitepaper*, 2001.

[11] H. Zhang, Service disciplines for guaranteed performance service in packet-switching networks, *Proc. IEEE*, vol. 83, no. 10, pp. 1374–1396, 1995.

[12] Cisco Systems, DiffServ – the scalable end-to-end QoS model, *WhitePaper*, 2005.

[13] T. X. Brown, Switch packet arbitration via queue-learning, in *Proc. 14th Int. Conf. Neural Information Processing Systems: Natural and Synthetic*, Vancouver, Canada, 2001, pp. 1337–1344.

[14] J. A. Boyan and M. L. Littman, Packet routing in dynamically changing networks: A reinforcement learning approach, in *Proc. 6th Int. Conf. Neural Information Processing Systems*, Denver, CO, USA, 1993, pp. 671–678.

[15] Z. Mammeri, Reinforcement learning based routing in networks: Review and classification of approaches, *IEEE Access*, vol. 7, pp. 55916–55950, 2019.

[16] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, et al., Knowledge-defined networking, *SIGCOMM Comput. Commun. Rev.* , vol. 47, no. 3, pp. 2–10, 2017.

[17] T. C. K. Hui and C. K. Tham, Adaptive provisioning of differentiated services networks based on reinforcement learning, *IEEE Trans. Syst. Man Cybern. C* (*Appl. Rev.*), vol. 33, no. 4, pp. 492–501, 2003.

[18] J. Rao, X. P. Bu, C. Z. Xu, L. Y. Wang, and G. Yin, VCONF: A reinforcement learning approach to virtual machines auto-configuration, in *Proc. 6th Int. Conf. Autonomic Computing*, Barcelona, Spain, 2009, pp. 137–146.

[19] A. da Silva Veith, F. R. de Souza, M. D. de Assunção, L. Lefèvre, and J. C. S. dos Anjos, Multi-objective reinforcement learning for reconfiguring data stream analytics on edge computing, in *Proc. 48th Int. Conf. Parallel Processing*, Kyoto, Japan, 2019, p.106.

[20] A. Bar-Hillel, A. Di-Nur, L. Ein-Dor, R. Gilad-Bachrach, and Y. Ittach, Workstation capacity tuning using reinforcement learning, in *Proc. ACM/IEEE Conf. Supercomputing*, Reno, NV, USA, 2007, p. 32.

[21] C. H. Yu, J. L. Lan, Z. H. Guo, and Y. X. Hu, DROM: Optimizing the routing in software-defined networks with deep reinforcement learning, *IEEE Access* , vol. 6, pp. 64533–64539, 2018.

[22] T. A. Q. Pham, Y. Hadjadj-Aoul, and A. Outtagarts, Deep reinforcement learning based QoS-aware routing in knowledge-defined networking, in *Proc. 14th EAI Int. Conf. Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Ho Chi Minh City, Vietnam, 2019, pp. 14–26.

[23] X. Y. You, X. J. Li, Y. D. Xu, H. Feng, and J. Zhao, Toward packet routing with fully-distributed multi-agent deep reinforcement learning, in *Proc. of 2019 Int. Symp. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks* (*WiOPT*), Avignon, France, 2019, doi: 10.23919/WiOPT47501.2019.9144110.

[24] X. Mai, Q. Z. Fu, and Y. Chen, Packet routing with graph attention multi-agent reinforcement learning, arXiv preprint arXiv: 2107.13181, 2021.

[25] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, S. Shakkottai, D. Kalathil, R. K. P. Mok, and A. Dhamdhere, QFlow: A reinforcement learning approach to high QoE video streaming over wireless networks, in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Networking and Computing*, Catania, Italy, 2019, pp. 251–260.

[26] J. Prados-Garzon, T. Taleb, and M. Bagaa, LEARNET: Reinforcement learning based flow scheduling for asynchronous deterministic networks, in *Proc. of 2020 IEEE Int. Conf. Communications*, Dublin, Ireland, 2020, doi: 10.1109/ICC40277.2020.9149092.

[27] P. Pinyoanuntapong, M. Lee, and P. Wang, Distributed multi-hop traffic engineering via stochastic policy gradient reinforcement learning, in *Proc. of 2019 IEEE Global Communications Conf.* (*GLOBECOM*), Waikoloa, HI, USA, https://webpages.uncc.edu/pwang13/pub/routing.pdf, 2019.

[28] J. Chavula, M. Densmore, and H. Suleman, Using SDN and reinforcement learning for traffic engineering in UbuntuNet Alliance, in *Proc. of 2016 Int. Conf. Advances in Computing and Communication Engineering* (*ICACCE*), Durban, South Africa, 2016, pp. 349–355.

[29] K. F. Xiao, S. W. Mao, and J. K. Tugnait, TCP-Drinc: Smart congestion control based on deep reinforcement learning, *IEEE Access*, vol. 7, pp. 11892–11904, 2019.

[30] B. Liu, Q. M. Xie, and E. Modiano, Reinforcement learning for optimal control of Queueing systems, in *Proc.*

of the 57th Annu. Allerton Conf. Communication, Control, and Computing (*Allerton*), Monticello, IL, USA, 2019, pp. 663–670.

[31] J. G. Dai and M. Gluzman, Queueing network controls via deep reinforcement learning, arXiv preprint arXiv: 2008.01644, 2021.

[32] M. Raeis, A. Tizghadam, and A. Leon-Garcia, Queue-learning: A reinforcement learning approach for providing quality of service, *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 1, pp. 461–468, 2021.

[33] A. Kattepur, S. David, and S. Mohalik, Automated configuration of router port queues via model-based reinforcement learning, in *Proc. of 2021 IEEE Int. Conf. Communications Workshops*, Montreal, Canada, 2021, pp. 1–6.

[34] S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[35] M. Bertoli, G. Casale, and G. Serazzi, JMT: Performance engineering tools for system modeling, *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 10–15, 2009.

[36] H. Kurniawati, D. Hsu, and W. S. Lee, SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces, in *Proc. Robotics: Science and Systems IV*, Zurich, Switzerland, doi: 10.15607/RSS. 2008.IV.0092008.

[37] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance, Computer System Analysis Using Queueing Network Models*. Upper Saddle River, NJ, USA: Prentice-Hall, 1984.

Ajay Kattepur received the BEng and MEng degrees in electrical and electronic engineering from Nanyang Technological University (NTU), Singapore, and the PhD degree in computer science from Inria, France. He is currently working at the Artificial Intelligent System Group, Ericsson Research. Prior to joining Ericsson Research, he was with the Tata Consultancy Services (TCS) Research & Innovation Labs in India. He was previously a postdoctoral researcher at the French National Institute for Computer Science and Control (Inria), Paris, France. His research interests include automated planning, reinforcement learning, and verification techniques.

Sushanth David received the BEng degree in electronics and communication from VT University, India, and the MEng degree in computer science from University of Pennsylvania, USA. He is currently at the Managed Services Unit in Ericsson with focus on design and construction of reusable frameworks, packages, and components, with emphasis on applying AI-based theories and techniques to the fulfilment and assurance across all domains of the 5G network. He has served on the Linux Foundation Networking Board and contributed to ONOS and CORD initiatives. His research interests include SDN/NFVi and O-RAN, time sensitive communications, and high performance distributed compute.

Swarup Kumar Mohalik received the BEng and MEng degrees in computer science from the Indian Institute of Technology, Kanpur, India, and the PhD degree in computer science from the Institute of Mathematical Sciences, Chennai, India. He has expertise in artificial intelligence and formal methods, and he has worked on application of Internet-of-Things and service automatization within Ericsson Research. He has worked as a postdoctoral researcher at LABRI, University of Bordeaux, France. Prior to joining Ericsson, He worked at Intel, General Motors, and Hewlett-Packard. He has research experience in the areas of formal specification and verification of real-time embedded software, model-based testing, and AI planning techniques.