
Speculative Packet Dispatch for Virtual Output Queuing Architecture Using LSTM Recurrent Neural Network

Alex Sumarsono^{*}, Mario Rodriguez

Department of Computer Engineering, California State University East Bay, Hayward, USA

Email address:

alex.sumarsono@csueastbay.edu (A. Sumarsono), mrodriguez115@horizon.csueastbay.edu (M. Rodriguez)

^{*}Corresponding author

To cite this article:

Alex Sumarsono, Mario Rodriguez. Speculative Packet Dispatch for Virtual Output Queuing Architecture Using LSTM Recurrent Neural Network. *International Journal of Information and Communication Sciences*. Vol. 6, No. 2, 2021, pp. 38-45. doi: 10.11648/j.ijics.20210602.13

Received: April 27, 2021; **Accepted:** May 17, 2021; **Published:** May 27, 2021

Abstract: Virtual Output Queuing (VOQ) is an architecture widely employed in modern networking products. Traffic from every ingress port is stored in a set of queues mirroring the structure of the egress ports. This architecture allows congestion on one egress port to be isolated from the other ports. A request-grant protocol is used to route packets from ingress to egress. When a packet is received, a request signal is issued. After the request reaches the egress side, a grant signal is generated based on some fixed threshold indicating there is space in the egress buffer to absorb the largest packet size dispatched from ingress. The buffer must be sized deep enough to accommodate in-flight traffic associated with a scenario where heavy congestion is found after the grant is issued. Awaiting a grant signal to arrive before dispatching packets incurs significant end-to-end latency. To alleviate this problem, a speculative packet dispatch approach (SPD) is proposed in which the request grant protocol is completely eliminated. Packets are dispatched speculatively from ingress to egress based on predictions that there is enough space in the egress buffer. This is achieved by incorporating an LSTM recurrent neural network as part of the VOQ controller. The LSTM is trained by time-series data sets generated from past observations on the queue occupancy. The experimental results show that SPD delivers excellent improvement on the system performance, reduces buffering requirements and preserves the property of VOQ.

Keywords: Computer Networks, Virtual Output Queuing, Long Short Term Memory, Machine Learning, Recurrent Neural Network

1. Introduction

Higher demands on network switches and routers continue to increase since the early days of the Internet. With the emergence of new generations of use cases such as cloud computing, high-definition video streaming, artificial intelligence applications and 5G mobile communications, switching devices with larger capacity and higher bandwidth are required to meet these demands [1-3].

The fundamental technology for high-speed packet switching is based on a switch with N input (ingress) ports and N output (egress) ports. Packets are received by the ingress ports and placed in the ingress queues. After they are classified by a forwarding engine, the packets are forwarded via an $N \times N$ cross-bar to the appropriate egress queues corresponding to the destination egress ports. There are various ways to implement this architecture. Some implementations assign a set of dedicated buffers for the

ingress queues and another set for the egress queues. Other implementations employ more of a shared buffer approach where the memory that stores the packets are shared among the ingress and egress ports [4, 6, 7].

Two modes of operation normally supported by modern switches are *cut-through* and *store-and-forward* [2]. In the *cut-through* mode, packets are sent to the egress ports as soon as possible to minimize the input-output delay. Because packet inspection and packet transmission occur at the same time, a mechanism is put in place to indicate to the receiver whether the packet being transmitted is a good packet or a corrupted packet. If it is a corrupted packet, the error checking mechanism puts invalid error check bits at the of the transmitted packet. The *store-and-forward* mode, on the other hand, does not send the packet out until the entire packet has been received and inspected, which causes a significant increase in the input-output delay. This mode allows the switch to drop corrupted packets internally. Hence, the switch

guarantees that only good packets are sent out. Regardless of whether the architecture is *cut-through* or *store-and-forward*, packet queuing is mandatory to manage congestion. Assuming the ingress and egress ports run at the same speed, traffic from two or more ingress ports destined to the same egress port will create congestion. If a downstream device asserts a flow control signal to an egress port, congestion is also created because the ingress rate is now larger than the egress rate. In both cases, packets need to be temporarily stored in the ingress or egress queues. To avoid packet loss in the switch, the queues must be large enough to absorb the in-flight traffic during congestion, which is assumed to be transitory. Packet loss cannot be avoided in sustained congestion since it would require infinite amount of buffering [14].

A simple ingress-egress queueing structure has an inherent problem known as Head-of-Line (HOL) blocking, in which congestion in one egress port can cause considerable throughput degradation in other egress ports [10, 11]. To solve this problem, an architecture known as Virtual Output Queuing (VOQ) was developed [5, 6]. In this architecture every ingress port has a set of queues, each of which is associated with an egress port. Received packets are classified and placed in their appropriate virtual output queues. Although referred to as *output* queues, it is important to note that these queues are actually located in the ingress side, hence, the name *virtual output* queues. The packets will be waiting in these queues until there is space in the egress queues before they are dispatched to the egress side. This technique ensures that egress ports that are congested are isolated.

VOQ is a powerful architecture that has been employed in networking products for many years. However, its performance is heavily dependent on various factors such as round-trip latency, egress queue occupancy and packet size [15]. This paper suggests a way to overcome these limitations by proposing a method called Speculative Packet Dispatch (SPD) where packets are speculatively dispatched from a VOQ on the ingress side. The accuracy of this method is largely determined by the predicted accuracy of a dispatch scheduler that constantly monitors the status of every destination egress queue. The scheduler is driven by a Long Short Term Memory (LSTM) network, which is a class of Recurrent Neural Networks (RNN). The LSTM is trained by the egress queue occupancy status represented as time-series data.

The rest of this paper is organized as follows: sections 2 and 3 describe the VOQ and Time-Series LSTM, respectively, section 4 presents the proposed technique, section 5 discusses the experimental results and section 6 ends the paper with some concluding remarks.

2. Virtual Output Queuing

Most modern switches provide deep packet buffering before the cross-bar and shallow buffering on the output side before the egress ports. The forwarding engine and the scheduler control how to fetch a packet from an input queue, route it through the cross-bar to a particular destination port. If

none of egress ports are congested, then traffic can flow at line-rate and port fairness is achieved. Preserving port fairness in the presence of congestion is a non-negotiable requirement for modern switches and routers. Congestion that occurs on one egress port should not affect the traffic going to other unrelated egress ports. Port fairness cannot be guaranteed in the traditional method of ingress buffering.

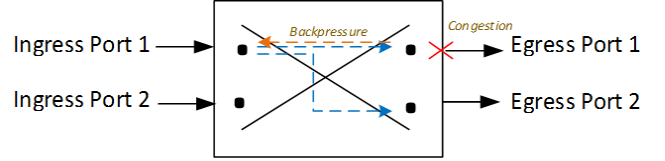


Figure 1. Effects of Flow Control on Egress Ports.

Consider a scenario depicted in Figure 1. Packets from Ingress Port 1 (IP1) are going to Egress Port 1 (EP1) and Egress Port 2 (EP2). A flow control issued by a congested downstream device on EP1 stops the switch from transmitting on that port but traffic from IP1 to EP2 should still flow. However, this flow control on EP1 causes all packets to wait in IP1's queue including those packets whose destination is EP2 until the flow control is deasserted [13]. This condition is called Head-of-Line-Blocking (HOLB).

There is another scenario that creates performance degradation without the presence of HOLB. Consider four flows: IP1 to EP1, IP1 to EP2, IP2 to EP1 and IP2 to EP2. Assume 75% of the packets from IP1 and IP2 are destined for EP1, while the rest goes to EP2. The throughput of each flow to EP1 will drop by 25%, which is acceptable since EP1 is 50% oversubscribed. However, the throughput of the other two flows to EP2 is also affected. This is not acceptable since EP2 is not oversubscribed.

The solution for all these limitations is to create a set of output queues placed in the ingress side that mirrors the egress ports as shown in Figure 2 [5, 10]. These queues are referred to as *virtual* queues [8, 9, 12], hence, the term Virtual Output Queue (VOQ).

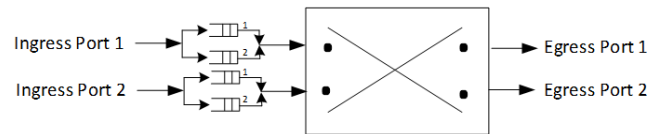


Figure 2. Virtual Output Queue Architecture.

It is easy to see that a flow control on EP1, which blocks all EP1's *virtual* output queues from sending packets, does not affect the switch to route packets destined for EP2 since those packets are stored in different *virtual* queues. Furthermore, EP2 is also impervious to an oversubscription condition on EP1. This is an elegant architecture that provides port fairness across all ingress and egress ports. The cost is a significant increase in the number of queues in the switch. If a switch has N ingress ports and N egress ports, then N^2 queues are required to support VOQ instead of N queues in the non-VOQ architecture.

3. Long Short Term Memory for Time Series Predictions

An artificial neural network consists of an input layer, an output layer and one or more in-between layers referred to as hidden layers [16-18]. The output of a given layer, which is a function of the weighted sum of its input, becomes the input of the next layer. The weights are learned using training examples via gradient descent based optimization. The gradients are computed starting from the output layer backwards all the way to the input layer. This is called backward propagation. A recurrent neural network is a neural network that has an additional path connecting the output back to itself. For each time step t a recurrent neuron receives the input as well as its own output from the previous step $t - 1$ as shown in Figure 3 (left). The network can be unfolded as depicted in Figure 3 (right) to explicitly show the information flow [19, 21-23].

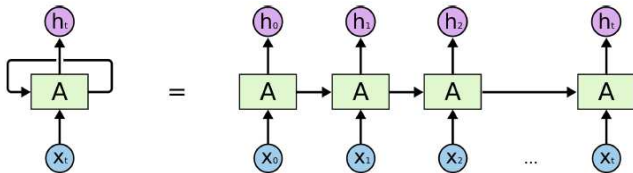


Figure 3. Recurrent Neural Network.

The output of the network can be written as

$$h_t = f(W_x^T x_t + W_y^T y_{t-1} + b)$$

where W_x is the weight matrix of the inputs of the current time step, W_y is the weight matrix of the outputs of the previous time step and b is the neuron's bias term.

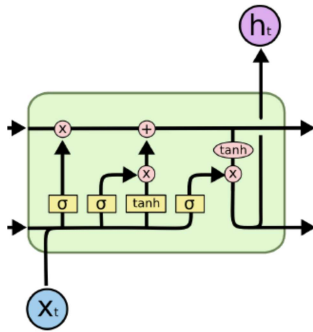


Figure 4. LSTM Cell.

Training an RNN is achieved by backward propagation through time [21]. Although it is conceptually straight forward, in practice it is very challenging due the vanishing and exploding gradient problems. As the gradients are computed recurrently, they can become extremely small or extremely large causing the training to become totally ineffective where the network can no longer capture long-term dependencies.

Long Short Term Memory (LSTM) is a special type of RNN shown in Figure 4 [19]. In addition to output h_t , cell state c_t is provided. The architecture is designed to mitigate the vanishing and exploding gradients by incorporating gates to

regulate the information flow into and out of the cell. A gate consists of a sigmoid function followed by a pointwise multiplication operation.

An LSTM cell has three types of gates: forget gate, input gate and output gate [20]. The forget gate, controlled by f_t , controls what information should be discarded from the cell.

$$f_t = \sigma(W_{xf}^T x_t + W_{hf}^T h_{t-1} + b_f)$$

The input gate, controlled by i_t , controls how much of new information should be added into the cell state from the input.

$$i_t = \sigma(W_{xi}^T x_t + W_{hi}^T h_{t-1} + b_i)$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

$$\tilde{c}_t = \tanh(W_{xg}^T x_t + W_{hg}^T h_{t-1} + b_g)$$

The output gate, controlled by o_t , determines what to output from the cell.

$$o_t = \sigma(W_{xo}^T x_t + W_{ho}^T h_{t-1} + b_o)$$

$$y_t = h_t = o_t * \tanh(c_t)$$

where

$W_{xf}, W_{xi}, W_{xg}, W_{xo}$ are the weight matrices of each layer corresponding to input x_t .

$W_{hf}, W_{hi}, W_{hg}, W_{ho}$ are the weight matrices of each layer corresponding to the previous time step.

b_f, b_i, b_g, b_o are the bias terms of each layer.

Due to its ability to handle long term dependencies, an LSTM network can be used to model a univariate time series problem [22, 24]. The model learns from a series of past observations to make a new prediction. The training dataset, which includes the labels, is constructed from these observations. Table 1 shows a training dataset to predict the egress buffer occupancy. It consists of a set of observations X with label y from a series $s_1, s_2, s_3, \dots, s_k$ with a window size of w where $k \bmod w$ is equal to 0.

Table 1. Predicted Egress Buffer Occupancy.

X	y
$s_1, s_2, s_3, \dots, s_{w-1}$	s_w
$s_2, s_3, s_4, \dots, s_w$	s_{w+1}
$s_3, s_4, s_5, \dots, s_{w+1}$	s_{w+2}
\dots	\dots
$s_{k-w+1}, s_{k-w+2}, s_{k-w+3}, \dots, s_{k-1}$	s_k

4. Proposed Architecture

As discussed earlier, VOQ should be employed as an architecture of choice in a networking device due to efficient packet transmission from the ingress side to the egress side. It is generally understood that higher egress buffer occupancy can be caused by either a flow control assertion from a downstream device or an oversubscription due to the dynamic nature of the traffic. As shown in Figure 5, buffer occupancy has a direct impact on throughput and latency, which constitute the overall system performance.

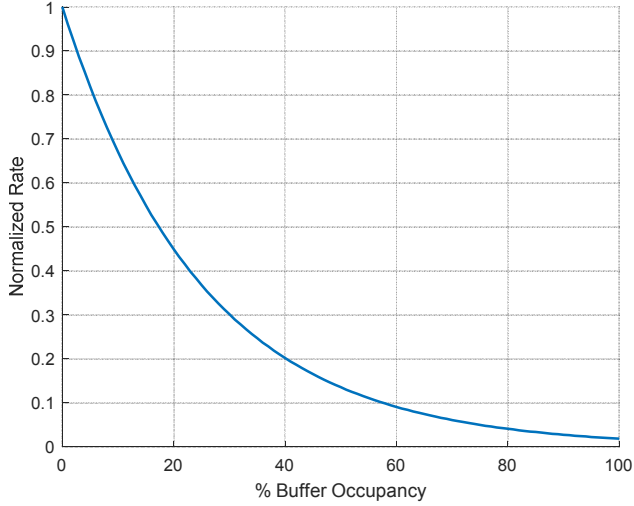


Figure 5. Transmission Rate as a Function of Buffer Occupancy.

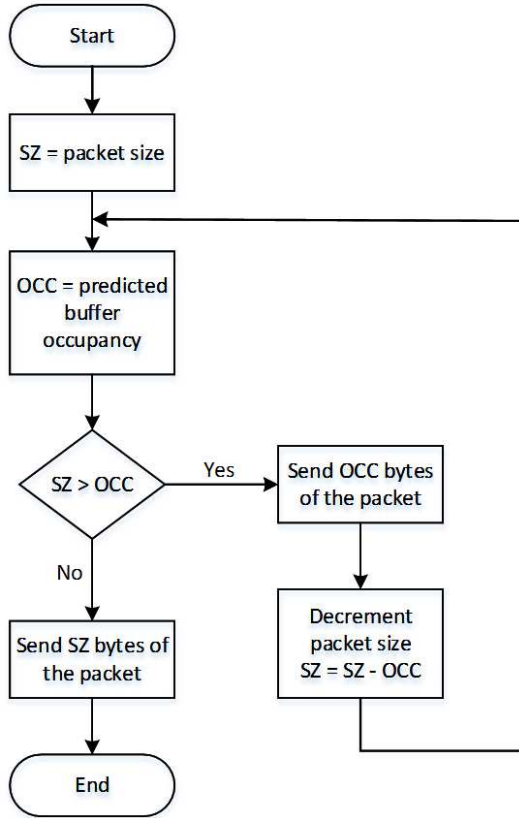


Figure 6. Speculative Packet Dispatch Flow Chart.

The performance drops rapidly as the buffer fills up indicating that the system cannot drain the data fast enough. Without intervention, this condition eventually leads to an overflow causing packet drop. Dropped packets have to be retransmitted from ingress incurring further performance hits. Alternatively, ingress packet transmission is stopped if a predetermined threshold on the egress buffer is exceeded. This approach prevents the egress buffer from overflowing but it requires ingress to have prior knowledge on the buffer occupancy before sending the packet. In other words, ingress sends a request, waits for a grant from egress then dispatches the packet. This sequence

of events is typically referred to as a request-grant handshake. Again, performance degradation is unavoidable.

The proposed architecture presented in this paper will minimize the performance hit by eliminating the need of having a request-grant handshake. Ingress speculatively dispatches full or partial packets based on the predicted egress buffer occupancy, hence, the name Speculative Packet Dispatch (SPD). A time-series LSTM network functions as a predictor. Based on a series of occupancy status over time, the LSTM makes predictions of the current value. A flow chart is shown in Figure 6.

Assuming *store and forward* and neglecting the delay through the cross-bar, the performance of a typical VOQ switch is given by the pseudo-code shown in Figure 7.

Non Speculative Packet Dispatch	Speculative Packet Dispatch
if $T_{ipkt} < RG_{Dly}$ Latency = $BfrDepth / PktLen \times RG_{Dly}$ Throughput = $T_{ipkt} / RG_{Dly} \times 100\%$ else Latency = $T_{ipkt} + RG_{Dly}$ Throughput = 100%	Latency = T_{ipkt} Throughput = 100%

Figure 7. Non-SPD and SPD Pseudo Code.

where

T_{ipkt} is the ingress packet time (the time it takes to transmit the entire packet)

RG_{Dly} is the request-grant latency

$BfrDepth$ is the depth of the buffer

$PktLen$ is the length of the packet

In a non-speculative approach, when ingress is ready to send a packet, it issues a request then waits for a grant before dispatching the packet. Another factor to be considered is the size of the egress buffer. Unlike SPD, which is a predictive algorithm, a non-speculative approach must allocate a fixed amount of storage space to absorb the in-flight data. The minimum depth of the buffer is given by

$$\begin{aligned} \text{Minimum Depth} \\ = (1 - R_{egr} / \epsilon) \\ \times \text{Maximum Packet Size} + \epsilon \end{aligned}$$

where

R_{igr} is the ingress rate

R_{egr} is the egress rate

ϵ is the safety margin

The term $(1 - R_{egr} / \epsilon)$ describes the average congestion experienced by the system.

5. Experimental Results and Analysis

Four areas were investigated to verify the viability of the proposed architecture: egress buffer occupancy prediction, request-grant delay, VOQ performance and minimum buffer size requirements. Simulations were performed using TensorFlow running on Google Colab for the LSTM-based egress buffer occupancy prediction and using System Verilog

simulated with Mentor Graphics ModelSim Verilog Simulator for the VOQ model. The results were collected and analyzed below.

5.1. Egress Buffer Occupancy Prediction

The first five rows of Table 2 are cases where the congestion starts to take place. Data is entering into the buffer at a higher rate than it is exiting causing the buffer to fill up. Eventually, ingress will stop sending traffic. The last five rows correspond to the scenarios where the egress rate is higher than the ingress rate because either the congestion has begun to subside or the ingress traffic has slowed down.

Given past buffer occupancies, the Time-Series LSTM makes its predictions on the most likely current occupancy levels as shown in columns 1 and 2, respectively. These predictions are quite accurate relative to the actual buffer occupancy listed in column 3. This is important as it relates directly to the overall performance of the system.

Table 2. Predicted Egress Buffer Occupancy.

Previous buffer occupancy (%)	Predicted current occupancy (%)	Actual current occupancy (%)
34, 36, 37, 39, 40	44	42
50, 52, 52, 54, 56	58	57
60, 62, 62, 63, 64	65	65
75, 76, 77, 78, 76	77	77
80, 82, 82, 84, 86	84	87
85, 85, 83, 80, 76	72	74
80, 79, 78, 77, 76	73	75
67, 65, 64, 61, 60	59	58
53, 54, 51, 51, 50	48	49
30, 29, 29, 27, 27	26	26

5.2. Request Grant Delay

Figure 8 shows the system throughput as a function of the request-grant round trip time (RTT) normalized by the ingress packet time. In non-speculative systems this quantity becomes a critical path as soon as the ratio exceeds 1. If the ingress packet time is larger than the request-grant RTT, the grant for the next packet is received while the transmission of the next packet is still in progress. The transmission of the next packet can proceed immediately yielding 100% throughput. However, if the request-grant RTT is larger than the packet time, an idle period (bubble) corresponding to the waiting time for the grant is inserted. This translates to permanent bandwidth loss that cannot be recovered causing the throughput to suffer. The severity of the throughput drop depends on the request-grant RTT to the ingress packet time ratio. Two factors that affect the ingress packet time are packet size and port speed. For the same packet size, packet time can be assumed to be linearly proportional to port speed. Any additional overhead associated with internal processing can be compensated by other means, such as internal speedups, hence, it can be neglected when considering the system performance. On the other hand, speculative systems do not have this issue since the request-grant loop has been removed completely and ingress now relies on the LSTM predictor. As expected, 100% throughput is maintained regardless of the size of the packet or

the port speed.

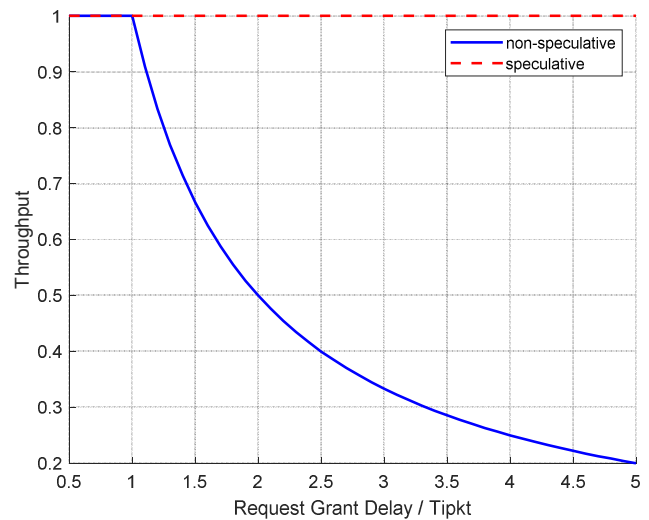


Figure 8. Effect of Request-Grant RTT on Throughput.

Throughput can also be observed on the egress port rates as depicted in Figure 9. The request-grant RTT does not vary with port speeds since it is an internal system parameter. If the request-grant RTT is 25 ns, then the minimum packet size for a 10 Gbps port and a 40 Gbps port to reach 100% egress rate is 250 bytes and 1000 bytes, respectively, in non-speculative VOQ systems. The packet time from a 100 Gbps port is always less than 25 ns. Therefore, the egress rate can never reach 100%. Once again speculative VOQ systems can sustain 100% egress rates throughout because they are not susceptible to packet sizes.

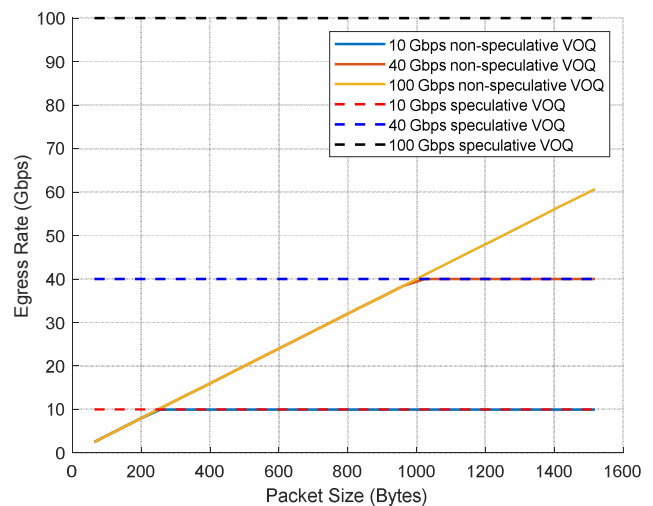


Figure 9. Effect of Packet Size on Egress Rate for 25 ns Request-Grant RTT.

In addition to throughput, the overall VOQ system performance is also measured with respect to end-to-end latency, which is the elapsed time from the moment a packet arrives on an ingress port until it exits out of an egress port. Since a packet cannot be dispatched until its grant is received in a non-speculative system, if the request-grant RTT is longer than the packet time, then the latency becomes unbounded. This

can be viewed as unrecoverable loss of bandwidth. In practice, latency is bounded by the size of the input buffer. This buffer limitation governs the ingress rate in which packets can be processed from the source. At steady state the ingress rate must be equivalent to the egress rate. Although latency seems to be better with a smaller buffer size, a system with smaller buffering is less tolerant to bursty traffic. Figure 10 shows the latency values for different packet sizes as a function the ingress buffer depth for a fixed request-grant RTT of 791 ns. That particular request-grant RTT is chosen to match the packet time of a 791-byte packet, which represents an average size between 64 bytes and 1518 bytes, the minimum and the maximum Ethernet packet sizes, respectively.

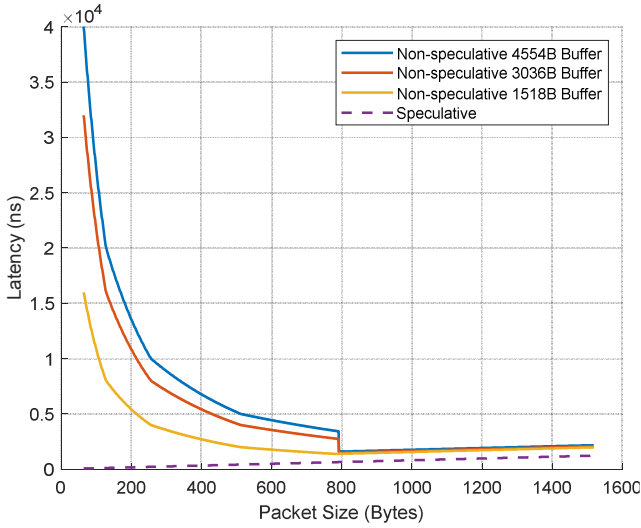


Figure 10. Effect of Ingress Buffer Size on Latency.

For non-speculative VOQ with *store-and-forward*, the latency grows exponentially as the ingress buffer depth increases for smaller packet sizes where the packet times are less than the request-grant RTT. Larger packet sizes whose packet times are greater than the request-grant RTT grow linearly. A speculative system shows a much better behavior. The latency is lower and grows linearly in proportion to the packet size regardless of the depth of the ingress buffer.

5.3. VOQ Performance

The buffer size must be sized to accommodate at least one maximum packet size (1518 bytes), also known as Maximum Transmission Unit (MTU). The simulation results for a 10 Gbps port with no congestion are shown in Table 3. The request-grant RTT of 741 ns is again assumed. For non-speculative systems, the latency grows exponentially with smaller packet sizes since these packets are affected much more severely by the request-grant RTT. The throughput increases with the packet size because larger packets would have smaller the idle period relative to the grant waiting time. Beyond 791 bytes, the latency is just the sum of packet time and request-grant RTT. Consequently, the throughput reaches 100%. Speculative

systems does not have this limitation again because the request-grant loop has been eliminated.

Table 3. VOQ Performance (No Congestion).

Packet size	Non Speculative		Speculative	
	Latency	Throughput	Latency	Throughput
64 bytes	15498 ns	8%	67 ns	100%
128 bytes	7740 ns	16%	118 ns	100%
256 bytes	3681 ns	33%	221 ns	100%
512 bytes	1921 ns	65%	426 ns	100%
640 bytes	1608 ns	81%	528 ns	100%
791 bytes	1293 ns	100%	649 ns	100%
1024 bytes	1483 ns	100%	835 ns	100%
1518 bytes	1887 ns	100%	1230 ns	100%

Another important attribute that needs to be preserved with the proposed SPD method is the fundamental property of VOQ in the presence of congestion. A congested port should not affect the throughput of uncongested ports. Six cases are shown in Table 4. The first two rows show that the throughput of egress port 1 drops in proportion to the severity of the flow control. Port 0 is unaffected since there is no flow control generated for this port. Rows 3 and 4 depict the same behavior if the flow control destination is reversed. The last two rows show that the throughput of ports 0 and 1 are determined by the respective flow control signals.

Table 4. Predicted Speculative VOQ Throughput in Presence of Flow Control.

Flow Control 0	Flow Control 1	Egress Port 0	Egress Port 1
0%	33%	100%	33%
0%	66%	100%	66%
33%	0%	33%	100%
66%	0%	66%	100%
33%	66%	33%	66%
66%	33%	66%	33%

5.4. Minimum Egress Buffer Size Requirements

Table 5 shows the required minimum egress buffer size relative to different levels of flow control for 10 Gbps ingress traffic. For non-speculative systems, the egress buffer must be large enough to absorb one MTU. Higher the congestion requires deeper the egress buffer in order to avoid an overrun condition. Table 6 shows the sustained congestion results caused by oversubscription. The 2x oversubscription non-speculative result is the same to that of 50% flow control in Table 5 since it effectively creates an identical system behavior.

Table 5. Egress Buffer Size for Intermittent Congestion Due To Flow Control.

Flow Control	Non Speculative	Speculative
25%	380 bytes	19 bytes
50%	759 bytes	26 bytes
75%	1139 bytes	19 bytes

Table 6. Egress Buffer Size for Sustained Congestion Due To Oversubscription.

Oversubscription	Non Speculative	Speculative
2x	760 bytes	26 bytes
3x	380 bytes	22 bytes
5x	304 bytes	16 bytes

Intermittent congestion can be alleviated by providing large enough buffering in the system such that the source traffic does not need to be halted. Since infinite amount of buffering is not possible, sustained congestion will eventually cause the source traffic to be intermittently stopped. Speculative systems inherently can provide guaranteed space without the risk of buffer overrun. Therefore, shallow egress buffers to incorporate some guard banding is sufficient as seen in Tables 5 and 6.

6. Conclusion and Future Work

Throughput and latency of a VOQ-based system that employs the request-grant protocol is susceptible to packet size variation, which typically is solved by introducing considerable internal bandwidth speed-up and deep buffering. This solution is expensive and leads to a significantly more complex implementation. This paper proposes a superior method referred to as the speculative packet dispatch (SPD) method. It speculatively dispatches packets from the ingress side to the egress side based on the predictions from an LSTM time-series network that has been trained by queue occupancy data sets.

The experimental results confirm that SPD-based systems have lower latency and higher throughput regardless of the size of the packet. They also require smaller amount of buffering and preserve the congestion-related per-port isolation property of VOQ.

Investigation on how to apply SPD to solve other networking problems is warranted. Since SPD can accurately predict various congestion scenarios, it might be possible to use it a system-level congestion avoidance algorithm. SPD could also be altered to perform speculative priority-based arbitration tasks to support quality of service for certain class of traffic.

Acknowledgements

This work was supported by the CSU East Bay Collaborative Research Grant (number D0208).

References

- [1] Zheng, L., Qiu, Z., Sun, S., Pan W. & Zhang, Z. (2018). Design and Analysis of a Parallel Hybrid Memory Architecture for Per-Flow Buffering in High-Speed Switches and Routers. *Journal of Communications and Networks*, 20 (6), 578-592.
- [2] Stallings, W. (2014). *Data and Computer Communications*, Pearson Education Inc.
- [3] Keith, R., & James, K. (2017). *Computer Networking: A Top-Down Approach*, Pearson Education Inc.
- [4] Jamali, M., & Ghiasan, A. (2019). Randomised Scheduling Algorithm for Virtual Output Queuing Switch at the Presence of Non-uniform Traffic. *IET Networks*, 8 (2), 138-142.
- [5] Juniper Networks. (2017, August). *Understanding CoS Virtual Output Queues (VOQs) on QFX10000 Switches*. Traffic Management Feature Guide.
- [6] Cisco Systems. (2016, July). *End-to-End QoS Implementation and Operation with Nexus* [Unpublished White Paper]. USA.
- [7] Cisco Systems. (2010). *Cisco Nexus 5548P Switch Architecture* [Unpublished White Paper]. USA.
- [8] Yoshigoe, K., & Christensen, K. (2003). An Evolution to Crossbar Switches with Virtual Output Queueing and Buffered Cross Points. *IEEE Network*, 17 (5), 48-56.
- [9] Xu, D. (2008). A Novel Scheduling Algorithm Based on Buffered Crossbar Switch Fabric in EPFTS Switch. *Proceedings of International Conference on Computer Science and Software Engineering*, China. <https://doi.org/10.1109/CSSE.2008.1316>.
- [10] Yin, Y., & Yin, S. (2010). Packet Processing Method Used in Input Terminal Buffering Queue. *Proceedings of International Conference on Industrial and Information Systems*, China. <https://doi.org/10.1109/INDUSIS.2010.5565899>.
- [11] Kyriakos, A., Patronas, I., Tzimas, G., Kitsakis, C., & Reisis, D. (2017). Realizing Virtual Output Queues in High Throughput Data Center Nodes. *Proceedings of Panhellenic Conference on Electronics and Telecommunications*, Greece. <https://doi.org/10.1109/PACET.2017.8259971>.
- [12] Do, V., & Yun, K. (2002). Packet Latency Optimization for VOQs in Variable Length Packet Switches. *Proceedings of IEEE Workshop on High Performance Switching and Routing*, Japan. <https://doi.org/10.1109/HPSR.2002.1024212>.
- [13] Cisco Systems. (2017). *Load Balancing in Multipath Switch Fabric* [Unpublished White Paper]. USA.
- [14] Lemeshko, O., Lebedenko, T., Mersni, A., & Hailan, A. (2019). Mathematical Optimization Model of Congestion Management, Resource Allocation and Congestion Avoidance on Network Routers. *Proceedings of International Conference on Information and Telecommunication Technologies and Radio Electronics*, Ukraine. <https://doi.org/10.1109/UkrMiCo47782.2019.9165445>.
- [15] Cisco Systems. (2014). *Multi-stage Congestion Management and Avoidance*. [Unpublished White Paper]. USA.
- [16] Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*, MIT Press.
- [17] Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn & TensorFlow*, O'Reilly Media Inc.
- [18] Haykin, S. (2009). *Neural Networks and Learning Machines*, Prentice Hall.
- [19] Tax, N. (2018). Human Activity Prediction in Smart Home Environments with LSTM Neural Networks. *Proceedings of International Conference on Intelligent Environments*, Italy. <https://doi.org/10.1109/IE.2018.00014>.
- [20] Olah, C. (2015). *Understanding LSTM Networks*. Available online: <https://colah.github.io/posts/2015-08-Understanding-LSTMs>.
- [21] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9 (8), 1735-1780.
- [22] Charniak, E. (2018) *Introduction to Deep Learning*. MIT Press.

- [23] Rikatsih, N., & Supianto, A. (2018). Classification of Posture Reconstruction with Univariate Time Series Data Type. Proceedings of International Conference on Sustainable Information Engineering and Technology, Indonesia. <https://doi.org/10.1109/SIET.2018.8693174>.
- [24] Karim, F., Majumdar, S., & Darabi, H. (2019). Insights into LSTM Fully Convolutional Networks for Time Series Classification. IEEE Access, (7), 67718-67725.