Using Machine Learning to Provide Differentiated Services in SDN-like Publish/Subscribe Systems for IoT

Yulong Shi[1,2(✉)], Yang Zhang[1], Hans-Arno Jacobsen[2], Bo Han[1], Mengxi Wei[1], Runyuan Li[1], and Junliang Chen[1]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
{shiyulong2015,yangzhang,hanbo92,wmx,lirunyuan,chjl}@bupt.edu.cn
[2] Middleware Systems Research Group, University of Toronto, Toronto M5S 1A1, Canada
jacobsen@eecg.toronto.edu

Abstract. At present, most publish/subscribe systems assume that all participants have the same Quality of Service (QoS) requirements. However, in many real-world IoT service scenarios, different users may have different delay requirements. How to provide differentiated services has become an urgent problem. The rise of Software Defined Networking (SDN) provides endless possibilities for meeting customized services due to greater programmability. In this paper, we first propose two new methods to predict the queuing delay of switches. One is an improvement of the traditional Random Early Detection (RED) algorithm; the other is a machine learning method using the eXtreme Gradient Boosting (XGBoost) model. Then we describe an SDN-like publish/subscribe system architecture and priority queues supported by OpenFlow switches to realize differentiated services. In order to guarantee QoS, we present a two-layer queue management mechanism based on user requirements. In the end, we compare our delay prediction methods with the RED method and verify the effectiveness of the two-layer queue management mechanism. Experimental results show that our solution is effective.

Keywords: Publish/Subscribe · Software Defined Networking
Quality of Service · Queue management · Machine learning

1 Introduction

Internet of Things (IoT) is the third wave of the world's information industry revolution following computers and the Internet. Especially in recent years, with the widespread popularity of smart phones and the development of sensing technology, such as Radio Frequency Identification (RFID), barcodes, and Quick Response (QR) codes, IoT devices and services have increased explosively. Different devices, end users and application scenarios have different Quality of

C. Pahl et al. (Eds.): ICSOC 2018, LNCS 11236, pp. 532–540, 2018.
https://doi.org/10.1007/978-3-030-03596-9_39

Service (QoS) requirements. However, how to meet these requirements is a huge challenge. It is a good way to solve these problems by providing QoS-ware differentiated services.

Middleware systems play an important role between the network layer and application layer of IoT. The publish/subscribe (pub/sub) system is an event-driven middleware system which provides distributed, asynchronous, loosely coupled communication between message producers (publishers) and consumers (subscribers). Publishers publish events, subscribers receive events which they express their interests in. The full decoupling provided by a pub/sub paradigm in time, space and synchronization between publishers and subscribers makes it particularly suitable for large-scale distributed IoT service deployments.

Software Defined Networking (SDN) is an emerging networking paradigm in which the control plane is separated from the forwarding plane. In this way, SDN simplifies the design, management of networks and also makes the network have more programmability. SDN-like is a new pub/sub model [5] that further extends pub/sub decoupling. We can make full use of the programmability of SDN-like to provide differentiated services for customized user requirements.

Most pub/sub systems consider that all subscribers have the same QoS requirements [3]. However, in real-world scenarios, different users may have different delay requirements. Many delay-sensitive IoT applications need real-time response to anomalies which should be dealt with high priority to prevent any danger. Wang et al. [4] tried to use the Random Early Detection (RED) algorithm to predict the queuing delay of switches. However, it is inaccurate because we cannot get the enqueued and dequeued data at the same time, there is often a significant difference compared to the real delay. In this paper, we propose an improved RED algorithm and a machine learning method to predict the delay. In order to guarantee QoS, we present a two-layer queue management mechanism. The evaluations demonstrate the effectiveness of our solution.

The major contributions of this paper are as follows:

- To the best of our knowledge, we are the first to predict the queuing delay of switches using the eXtreme Gradient Boosting (XGBoost) model of machine learning. We also propose the Incremental Difference Method (IDM), an improvement of the RED algorithm, and compare the performance of them.
- We describe an SDN-like pub/sub system architecture and how to use priority queues to provide differentiated services for subscribers.
- We present a two-layer queue management mechanism based on user requirements from two different perspectives: (1) The local queue bandwidth adjustment algorithm for a single switch in the SDN controller; (2) The global QoS control strategy for all switches on the path from a publisher to a subscriber in the administrator of the system.

The remainder of the paper is organized as follows. Section 2 describes the preliminaries. Section 3 proposes an SDN-like pub/sub system architecture. Section 4 introduces the queuing delay prediction method. Section 5 presents the two-layer queue management mechanism. Section 6 provides the experimental results. Section 7 concludes this paper with an outlook on future research.

2 Preliminaries

XGBoost Model. XGBoost model is an effective machine learning method proposed by Tianqi Chen in 2016 [2], which can solve regression prediction problems. In [2], the objective function is proposed as shown in Eq. (1). We use the classic Root Mean Square Error (RMSE) loss function as the evaluation function, as shown in Eq. (2).

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \sum_{k=1}^{K} \Omega(f_k) + const \tag{1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - y_i)^2} \tag{2}$$

Here, $l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)$ is the loss function, $\Omega(f_k)$ is the regularization term. $RMSE$ is the square root of the mean of the squares of the errors between the prediction $\hat{y_i}$ and the target y_i.

RED Algorithm. The basic principle is that it monitors the average queue length to reflect the queue congestion. The RED formula is shown in Eq. (3).

$$avgQ = (1 - w) * avgQ + w * qLen \tag{3}$$

$$qLen = enQ - deQ \tag{4}$$

$$Delay = avgQ/Width \tag{5}$$

Where $avgQ$ is the average queue length, $qLen$ is the real-time queue length, w is the weight, enQ and deQ are the total number of bytes enqueued and dequeued, respectively. $Delay$ is the queuing delay. $Width$ is the queue bandwidth.

Incremental Difference Method. Equation (4) can be improved as follows:

$$qLen = \Delta enQ - \Delta deQ + qLen \tag{6}$$

Where ΔenQ and ΔdeQ are the increment of enQ and deQ, respectively. This method does not need to guarantee the simultaneity of getting the enqueued and dequeued data, avoiding the influence of measurement time difference on data.

3 SDN-like Pub/Sub System Architecture

SDN-like Pub/Sub System Architecture. The SDN-like pub/sub system architecture is shown in Fig. 1, which includes one administrator and several clusters. The administrator is responsible for the global network management and interacts with the controller of each cluster. A cluster contains a controller, several switches, publishers and subscribers. Border switches are used to interconnect clusters. Users communicate with the system by Web Service Notification (WSN). The system contains three layers: global management layer, control

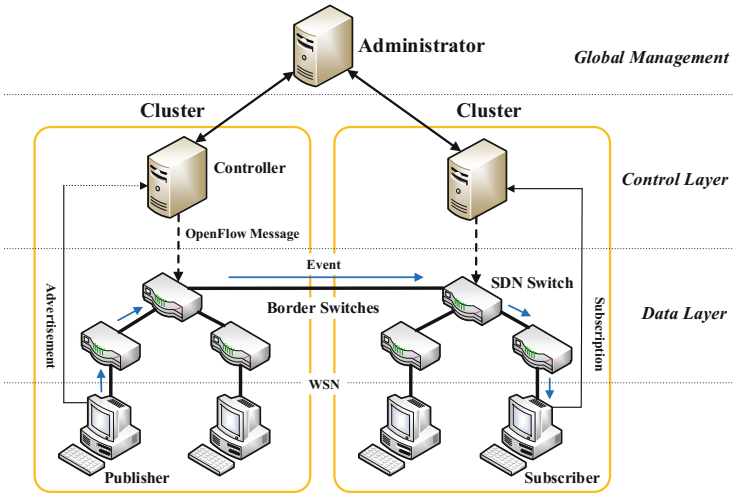

Fig. 1. SDN-like pub/sub system architecture

layer and data layer. Our SDN-like pub/sub system is mainly implemented in the control layer, namely, SDN controllers.

Two-Layer Queue Management Mechanism. The two-layer queue management mechanism is shown in Fig. 2. It is implemented in the control layer and the application layer. Specifically, the local queue bandwidth adjustment is implemented in the control layer; the global QoS control is implemented in the application layer. The priority queue is mainly implemented in SDN switches.

Topic Encoding. In our topic-based pub/sub system, topics are represented as a Lightweight Directory Access Protocol (LDAP) topic tree. The topic, event type and queue priority are encoded into binary strings of the 128 bits IPv6 multicast address in the header of packets. They are used to match flow tables directly when forwarding.

Priority Queue. Different priority queues have different bandwidths. The bandwidth size determines the forwarding capability of queues. In this way differentiation services are provided. OpenFlow switches can support up to 8 priority queues per port. These queues are numbered from 0 to 7, and the larger the queue number is, the higher the priority is. Messages are divided into three levels according to their emergency degrees: low, medium, high. The low priority messages enter queue 5, the medium enter queue 6, and the high enter queue 7, as shown in Fig. 2.

4 Queuing Delay Prediction Using XGBoost

Data Preprocessing. We collect a large amount of real data, such as bandwidth, package size by capturing data packets and logging once per monitoring

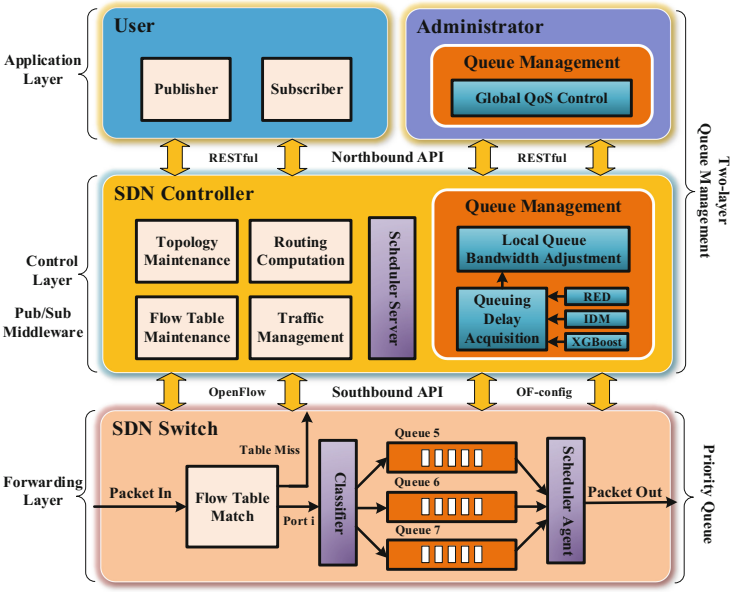

Fig. 2. Two-layer queue management mechanism

period (200 ms). Then we get the queue data after preprocessing by cleaning dummy data, filling missing values and calculation.

Feature Selection. Packets distribution means the distribution of packet transmission time intervals. The packets distribution shows periodicity, so we use the Autoregressive Integrated Moving Average (ARIMA) model [1] to obtain the cycle. We also perform a covariance test on the cycle between two adjacent packets. The correlation coefficient is 0.87293211. This higher value shows that there is little difference in their waveform distribution, and this method is reasonable. so we use packets distribution as a feature. In raw data, there are many features represented by string that the XGBoost model cannot receive, so we encode them into integer.

Model Training and Parameter Adjustment. We use the XGBoost model for training. The tree model is easily overfitting, so we divide training set by 20% as validation set and set it as watchlist to obtain the optimal number of iterations. We also use the

Table 1. Training results

XGBoost parameter	RMSE
min_child_weight=10; subsample=0.7; colsample_bytree=0.7; scale_pos_weight=0.8; max_depth=4; eta=0.1; early_stopping_rounds=30;	5.42028e+07
min_child_weight=10; subsample=0.7; colsample_bytree=0.7; scale_pos_weight=0.8; max_depth=6; eta=0.1; early_stopping_rounds=40;	5.59389e+07
min_child_weight=10; subsample=1; colsample_bytree=1; scale_pos_weight=1; max_depth=10; eta=0.1; early_stopping_rounds=50;	7.49699e+07

score of the verification set no longer declining for 10 generations consecutively as a criterion for early stop. The training results are shown in Table 1. The smaller the RMSE is, the closer the prediction is to the real value, so we choose the first row in Table 1.

5 Two-Layer Queue Management Mechanism Based on User Requirements

We achieve the two-layer queue management mechanism from two perspectives. One is the local bandwidth adjustment for a single switch, SDN controllers adjust the bandwidth according to the queue priority and the queuing delay. The other is the global control for all switches on the path. The administrator configures the delay constraint of each switch as the local bandwidth adjustment reference.

Local Queue Bandwidth Adjustment Algorithm. The bandwidth of each queue needs to be readjusted according to the delay and the queue priority. The constraints for queues are as follows:

$$w_q * t_q = AvgQ_q, q = 5, 6, 7 \tag{7}$$

$$t_q \leq T_q, q = 5, 6, 7 \tag{8}$$

$$\sum_{q=5}^{7} w_q = Port \tag{9}$$

$$w_q > 0, q = 5, 6, 7 \tag{10}$$

$$\text{Minimize}(c_5 * t_5 + c_6 * t_6 + c_7 * t_7) \tag{11}$$

Where w_q is the bandwidth of queue q, t_q is the queuing delay, $AvgQ_q$ is the average queue length, T_q is the delay constraint, $Port$ is the total bandwidth of each switch port. Equation (11) is the adjustment goal, namely, minimizing the weighted delay of queues. c_q is queue weight (coefficient).

Algorithm 1. Local Queue Bandwidth Adjustment Algorithm

Input: w_q, t_q, c_q, T_q, $q = 5, 6, 7$ //t_q is predicted by the XGBoost model, IDM or RED algorithm.
Output: B_q, D_q
1: **Initialize** $B_q = w_q$, $D_q = 0$, $Port = 100$
2: $AvgQ_q = w_q * t_q$
3: $Sum = \sum_{q=5}^{7} (AvgQ_q * c_q)^{\frac{1}{2}}$
4: **if** $t_q \leq T_q$ **then** //Lines 4∼6, use Eqs. (9)∼(11) to reason
5: $B_q = Port * (AvgQ_q * c_q)^{\frac{1}{2}} / Sum$ //calculate the new bandwidth B_q
6: $D_q = Sum * (AvgQ_q / c_q)^{\frac{1}{2}} / Port$ //calculate the new delay D_q
7: **else**
8: notify the administrator to adjust T_q
9: **end if**

In Algorithm 1, B_q is sent to the switch, and D_q is fed back to the administrator. The complexity of Algorithm 1 is $O(1)$.

Global QoS Control Strategy. We use U_j to represent the delay requirements proposed by subscriber j. There is a lower delay limit t_i for hop (switch) i. We use f_i to represent the queuing delay fed back by the controller where switch i resides. For the whole path, the constraints are as follows:

$$\sum_{i=1}^{n} T_i \leq U_j, j = 1, 2, ..., m \tag{12}$$

$$T_i \geq t_i, 1 \leq i \leq n \tag{13}$$

$$\text{Minimize}(U_j - \sum_{i=1}^{n} T_i) \tag{14}$$

$$T_i' = \frac{f_i U_j}{\sum_{i=1}^{n} f_i} T_i \tag{15}$$

Where n is the number of hops, m is the number of subscribers of a topic. T_i' is the new delay constraint, T_i is the last one. We use the best adaptation principle to adjust the bandwidth, as shown in Eq. (14). If it has a solution, the administrator will take T_i as the T_q of node i, and send it to the controller. If no solution, the administrator will notify the subscriber by controller to resubmit a new one. The administrator recalculates T_i' of each switch according to Eq. (15).

The global QoS control strategy is shown in Algorithm 2. The complexity of Algorithm 2 is $O(n)$.

Algorithm 2. Dynamic Threshold Calculation Algorithm

Input: $CurrentDelay$, $LastDelayConstraint$, $UserDelay$, $Path$, $Priority$, $ConstraintTable$
Output: Res //the new delay constraint
1: **Initialize** $Res = 0$, $temp = 0$
2: **for** $Switch$ in $Path$ **do** //calculate the sum of the last delay constraint on the path
3: $Con = ConstraintTable.get(Switch).get(Priority)$
4: $temp = temp + Con$
5: **end for**
6: $Res = CurrentDelay * UserDelay * temp / LastDelayConstraint$

This strategy makes full use of the administrator's characteristics which can control the global network. In this way, the delay of the entire path can satisfy the user needs, realizing the SDN-like topic-based differentiated services.

6 Performance Evaluation

Experiment Setup. We use three SDN-enabled physical switches and several PCs to setup the experiment topology as shown in Fig. 3. Each OpenDayLight controller, switches and some hosts form a cluster such as G1. The switch model is Pica8-p3290, the bandwidth of each switch port is 100 Mb/s.

Queuing Delay Prediction Methods Comparison. In this experiment, we set the bandwidths of queue 5, 6 and 7 are 10 Mb/s, 30 Mb/s and 60 Mb/s, respectively. For each queue we run three queuing delay prediction methods. The packet size is 1 KB. The experimental results about queue 5 are shown in Fig. 4. We can conclude that the two new methods are both better than the RED method, and XGBoost is better than IDM, so we choose the XGBoost method for the following experiments.

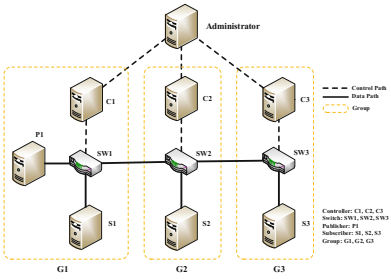

Fig. 3. Experiment topology

Local Queue Bandwidth Adjustment Algorithm Verification. For each combination of three priority queues, we compare the delay and packet loss rate under different frequencies of sending packets. The experimental results about one queue congestion are shown in Fig. 5. The bandwidth of queue 7 is 60 Mb/s before adjustment. When the frequency is between 5000 and 10000, queue 7 starts congestion, the delay remains at 125 milliseconds. After the adjustment, the queue starts becoming congested when the frequency is between 10000 and 20000, the delay remains at 83 milliseconds, the packet loss rate drops significantly and the bandwidth of this queue is 88 Mb/s. The data show that this algorithm is effective.

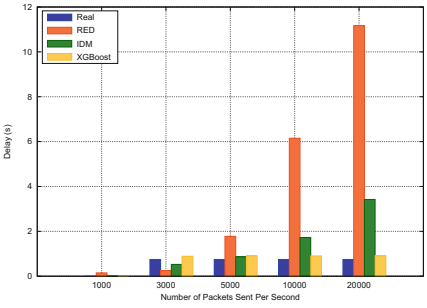

Fig. 4. Prediction methods comparison

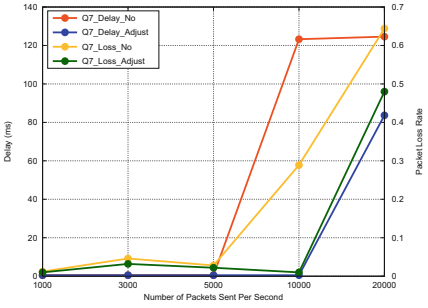

Fig. 5. Local queue bandwidth adjustment

7 Conclusion

We propose an effective machine learning method using the XGBoost model to predict the queuing delay of switches. Experiments show that it is better than IDM and the traditional RED method. We also present an SDN-like pub/sub

system architecture and a two-layer queue management mechanism based on user requirements to provide differentiated IoT services. Experimental results show that our solution is effective. However, we only use three OpenFlow physical switches to do the experiments due to their high costs, therefore it is difficult to involve routing problems. In the future, we will try to solve these problems. On the other hand, we can combine the local queue bandwidth adjustment algorithm and routing algorithms to improve the QoS of pub/sub systems.

Acknowledgement. This research is supported by the National Key Research and Development Program of China (No. 2018YFB1003800), the State Scholarship Fund of China Scholarship Council (No. 201706470069), China Postdoctoral Science Foundation (No. 2017M620617). The authors would like to thank Geoffrey Elliott at the University of Toronto and the anonymous reviewers for reviewing this manuscript.

References

1. Bartholomew, D.: Time series analysis forecasting and control. J. Oper. Res. Soc. **22**(2), 199–201 (1971)
2. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM (2016)
3. Tariq, M.A., Koldehofe, B., Koch, G.G., Khan, I., Rothermel, K.: Meeting subscriber-defined QoS constraints in publish/subscribe systems. Concurr. Comput.: Pract. Exp. **23**(17), 2140–2153 (2011)
4. Wang, Y., Zhang, Y., Chen, J.: Pursuing differentiated services in a SDN-based IoT-oriented pub/sub system. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 906–909. IEEE (2017)
5. Zhang, K., Jacobsen, H.A.: SDN-like: the next generation of pub/sub. arXiv preprint arXiv:1308.0056 (2013)