QoS Dynamic Perception Scheduling Strategy for Edge Intelligent Computing

Wanwan Hou¹*, Depeng Sun² and Mengxue Sheng³

¹Beijing Engineering Research Center for IoT Software and Systems, Beijing University of Technology, Beijing, 100124, China
²Beijing Engineering Research Center for IoT Software and Systems, Beijing University of Technology, Beijing, 100124, China
³Beijing Engineering Research Center for IoT Software and Systems, Beijing University of Technology, Beijing, 100124, China

*Corresponding author’s e-mail: houwanwan@126.com

Abstract. Edge intelligent computing devices are often deployed in some extreme environments, where the transmission network bandwidth is low or the network environment changes greatly. Therefore, the traditional queue scheduling algorithms cannot guarantee the QoS of edge intelligent computing. WF²Q+ allocates bandwidth according to a fixed weight, which causes real-time data flow delay to increase when the network is unstable. The dynamic perception scheduling strategy proposed in this paper is to dynamically change the weight of WF²Q+ by dynamically sensing the backlog length of the queue. At the same time, combined with the queue scheduling algorithm of PQ, this algorithm can prioritize the transmission of real-time data with certain fairness. In addition, the token bucket algorithm is used to limit the sending rate of the device and prevent network congestion caused by burst data injection into the network. After experimental simulation, the improved algorithm can achieve good results on the delay index.

1. Introduction

Edge intelligent devices are deployed in a variety of network environments, with very great differences in bandwidth , large changes in network rates, and quality of service (QoS) for real-time data cannot be guaranteed. Currently more commonly used scheduling algorithms are: first-in-first-out queue algorithm (FIFO), priority queue algorithm (PQ), and weighted fair queue (WFQ) algorithm. Due to the fixed weight of the data stream, the traditional fair queue scheduling algorithm cannot guarantee the real-time service delay when the data flow is burst. Literature [6] proposed WFQ based on service classification and bandwidth allocation, however, it is necessary to measure the receiving rate and available bandwidth of each data flow in real time and accurately before iteratively allocating bandwidth for each data flow. Literature [10] proposed to dynamically modify the weight of WFQ, but it is necessary to calculate the packet loss rate in real time first, and then adjust the queue length to change the weight. In this paper, an improved WF²Q+ combined PQ queue scheduling algorithm is proposed, which can dynamically adjust the weight of data flow according to the length of real-time queue backlog to meet the real-time requirements of edge intelligent computing. The algorithm is simple to implement, which can not only guarantee the real-time requirements, but also has certain
fairness. Finally, the token bucket algorithm is used to limit the transmission rate of devices and prevent the network congestion caused by burst data injection into the network.

2. Related technologies

The idea of algorithm improvement proposed in this paper is to synthesize the advantages of the following common queue scheduling algorithms.

2.1. First in first out queue (FIFO)

The FIFO queue uses the default queuing method that first come first served, which only provides the simplest store and forward. The advantage is simple and low cost.

2.2. Strict priority scheduling algorithm (PQ)

In the strict priority scheduling algorithm, the priority of each queue is gradually reduced. If there is data in the high priority all the time, the low priority queue will always be unserviced and hungry, which is very useful for low latency services.

2.3. General processor sharing algorithm (GPS)

GPS algorithm is a queue scheduling algorithm based on ideal flow. Assume that the data packet can be subdivided infinitely, and the processing rate of the scheduler is constant r. At time t, if there are data packets in the queue of data flow i waiting to be processed, it is said that data flow i is active state at time t. Assume that the positive integers $\Phi_1, \Phi_2, \Phi_3, \ldots, \Phi_N$ respectively represent the weight of each data flow, and $S_i(\tau, t)$ is the service obtained by data flow i within the time interval $(\tau, t)$. The GPS definition is as follows:

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} \geq \frac{\Phi_i}{\Phi_j}, j = 1, 2, \ldots, N \quad (1)$$

Because r is the processing rate of the scheduler, accumulating the data flow j yields $S_i(\tau, t) \sum_j \Phi_j \geq (t - \tau) r \Phi_i$. According to the rotation swap, the minimum service rate of any data flow i is $r_i = \frac{\Phi_i}{\sum \Phi_i} r$.

2.4. Weighted fair queue algorithm (WFQ)

WFQ algorithm is a kind of data packet-based algorithm implementation of GPS. It adopts work-conserving mode, that is, the scheduler will work continuously as long as there is data in the candidate queue. We define the arrival or departure of a data packet as the occurrence of an event, and the event $j$th occurrence time is recorded as time $t_j$. The time of the first event is recorded as $t_1 = 0$. The activation state of these data flow in the time interval $(t_{j-1}, t_j)$ remains unchanged, and we use $B_j$ to represent When the scheduler is idle, the virtual time $V(t)$ is reset to 0, and the $V(t)$ of WFQ is calculated as follows:

$$\begin{cases} 
V(0) = 0 \\
V(t_j + \tau) = V(t_{j-1}) + \frac{\tau}{\sum_{i=B_j} \Phi_i} r, \tau \leq t_j - t_{j-1}, j = 2, 3, \ldots \quad (2) 
\end{cases}$$

Suppose the arrival time of the k-th packet of data flow i is $a_i^k$, the length is $L_i^k$, the virtual start time is $S_i^k$, and the virtual end time is $F_i^k$. If $F_i^0 = 0$, then $S_i^k$ and $F_i^k$ are calculated by the following formula:

$$\begin{cases} 
S_i^k = \max(F_i^{k-1}, V(a_i^k)) \\
F_i^k = S_i^k + \frac{L_i^k}{\Phi_i} \quad (3)
\end{cases}$$

The scheduler selects the packet with the smallest virtual end time to send according to the SFF policy (Small virtual Finish time First).
2.5. Weighted fair queue algorithm (WF²Q)

Based on the analysis of GPS and WFQ, it is found that when WFQ selects a data packet using the SFF strategy, the data packet that has not been serviced in the GPS reference system will be sent ahead of time, so that there is a certain difference between the departure order of packets and the departure order in the GPS reference system. Therefore, the WF²Q algorithm uses the SEFF (Small Eligible virtual Finish time First) strategy to select packets with the smallest virtual end time to reduce this difference.

WF²Q makes the following improvements. When the scheduler selects data packets, it does not select all blocking sessions, but selects the packet with the smallest virtual end time from the blocking sessions that have already started serving in the GPS reference system. At time \( t \), a packet is selected according to the following formula.

\[
\{ p_i^k | b_{GPS}^k \ll \tau \ll b_{WFQ}^k \} \quad (4)
\]

2.6. Weighted fair queue algorithm (WF²Q+)

Although WF²Q+ simplifies the complexity of WF²Q implementation, it can provide almost the same performance. The definition of the virtual time function is somewhat changed, \( h_i(t) \) represents the sequence number of the data packet in the head of session \( i \), and \( B(t) \) represents the backlog of sessions. As follows:

\[
V_{WF²Q+}(t + \tau) = \max \left( V_{WF²Q+}(t) + \tau, \min_{i \in B(t)} \left( h_i(t) \right) \right) \quad (5)
\]

The current virtual time is greater than or equal to the minimum virtual start time of the packet at the head of each session queue. This property ensures that a valid packet can be selected each time when using the SEFF policy.

To simplify the implementation, the virtual start time and end time of WF²Q+ are modified as follows:

\[
S_i = \begin{cases} 
F_i & \text{if } Q(a_i^k - ) \neq 0 \\
\max(F_i, V(a_i^k)) & \text{if } Q(a_i^k - ) = 0 
\end{cases} \quad (6)
\]

\[
F_i = S_i + \frac{l_k}{r_i} \quad (7)
\]

\( Q(a_i^k - ) \) is the sum of the lengths of all data packets in session \( i \) before the data packets reach session \( i \). \( S_i \) and \( F_i \) are the virtual start time and end time of the first packet at the head of the queue.

2.7. Token bucket

Token buckets can implement traffic shaping, which can smooth a large number of burst data flows and send data at a constant rate. When the token bucket is not full, the token is injected into the token bucket at a certain rate until the token bucket is full. After the data packet arrives, if the token in the token bucket is greater than or equal to the length of the data packet, the data packet can be sent, and the same number of tokens is subtracted from the token bucket. Otherwise the data packet will continue to wait or be dropped directly. When the token generation rate in the token bucket is greater than the data flow arrival rate, all data packets can be forwarded in time, and the token bucket will eventually be filled or overflowed, which allows the token bucket to cope with some burst flows.

3. Research on improved WF²Q+ combined with PQ algorithm

For WF²Q+, this paper proposes several problems that need to be improved in the algorithm:

- The algorithm cannot distinguish between core real-time services, non-core real-time services and non-real-time services.
- In the case of a low sending rate, if the WF²Q+ service is continued according to the established weight, the real-time service delay and packet loss rate will increase.
- If there is a burst of data flow, it will lead to a large number of data injected into the network, resulting in network congestion or even crashes.
Therefore, an algorithm needs to be designed in the system to solve the above problems at the same time. Aiming at the problem that the core real-time services cannot be served in time, this article proposes to add a separate queue to store the core real-time services and give it absolute priority. Because the data of the core real-time services in the system is less, other services will not starve. In the case of low transmission rate, a fixed weight will cause a backlog of non-core real-time service queues and increase the packet loss rate. A strategy is proposed to dynamically modify the weight of non-core real-time service based on the length of the queue backlog. Aiming at the problem that burst data flow injected into the network will cause network congestion, a token bucket is added before the data packet is sent to control the burst flow and make the sending rate relatively stable.

The core of the algorithm is to add an absolute priority queue of core real-time services, dynamically modify the weights of non-core real-time services, and add the token bucket. To address the shortcomings of WF2Q+, the following mechanisms are proposed:

- **Core real-time queue mechanism**
  The core real-time queue has the highest priority. As long as the core real-time queue is not empty, it will be scheduled ahead of other queues.

- **Non-core real-time queue mechanism**
  From equation 7, it can be known that the larger weight of the queue, the larger service rate, and the smaller virtual end time of the data packet. The weight adjustment strategy can make non-core real-time services get better services. When the non-core real-time service queue generates backlog, the weight is changed dynamically according to the length of the queue backlog. The longer backlog queue length, the greater weight.

The maximum queue length of non-core real-time services is Lmax, the lower bound of dynamic adjustment queue interval is Lmid, Lnow is the current queue backlog length, the new weight is $\phi_i'$. The weight is proportional to the service rate, the weight increases, and the corresponding service rate also increases.

$$\phi_i' = \frac{L_{max}}{L_{max} - (L_{now} - L_{mid})} \phi_i \quad 1 \leq L_{mid} \leq L_{now} \leq L_{max} \quad (8)$$

Through the weight adjustment, it can provide larger weight for non-core real-time queue with longer backlog queue, more bandwidth can be allocated, and the virtual end time can be reduced to ensure the fairness of services.

![Figure 1. Multiplication factor.](image)

When Lmax=50 and Lmid=0, from the relationship between Lnow and the weight coefficient, we can see that the larger Lnow, the faster weight value increases. When Lmid is greater than or equal to Lmax, the non-core real-time queue degenerates into a fixed weight.

- **Token bucket mechanism**
  Token bucket can be used for traffic shaping. When the data package is to be sent, the corresponding number of tokens will be obtained according to the length of the data package. If the
number of tokens is not enough, the scheduler will wait for the number of tokens in the token bucket to be greater than or equal to the length of the data package to be sent. At the same time, the corresponding number of tokens need to be subtracted from the token bucket. Because the token bucket has a certain capacity, a certain amount of burst data flow is allowed; the overall data sending rate is limited to a certain rate.

Based on the above considerations, the following queue structure is proposed:

![Figure 2. Improved WF²Q⁺ queue scheduling algorithm.](image)

Data packets come through the classifier for classification, and then enter different queues for queuing according to the classification results. Each queue is queued according to the FIFO principle. If there are data packets in the core real-time queue, scheduling is prioritized; other queues are scheduled according to the fair queue scheduling algorithm, in which the weight of non-core real-time queue is dynamically adjusted based on the backlog length, and the weight of non-real-time queue is fixed. The general principle is to select the core real-time queue first, and then select the data packets according to the SEFF principle of WF²Q⁺. After selecting the data packets, the traffic is shaped by token bucket algorithm, and the output rate is controlled within a certain rate.

### 4. Simulation experiments and results analysis

In the Linux environment, a network simulation environment is set up to simulate a total of 6 data flows. The generation rate of each data flow is 1280 Byte/s. Each data flow generates a total of 50 data packets. The length of each data packet is 0-1024 bytes. The weights of the six data flows are (0, 3, 2, 3, 2, 1), data flow 0 is absolute priority queue, and the weight is 0. Adjust the sending rate of the scheduler through the traffic shaping to 640 Byte/s, 1280 Byte/s, 2560 Byte/s, 5120 Byte/s, Lmid is 0, 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, Lmax is 50. The structure of the simulation experiment is as follows:

![Figure 3. Simulation experiment structure.](image)

In each experiment, we first determine the sending rate of the scheduler and the Lmid, and 6 data flows are sent synchronously. According to the classification result of the classifier, it is queued in different queues. The scheduler selects the appropriate packet scheduling according to the improved
WF$^2$Q+ algorithm, and adjusts the sending rate of the packets to be scheduled through the traffic shaping. After the receiver has completely received all 50 packets of 6 data flows, it calculates the total delay of 50 packets of each data flow. The delay statistics of the improved WF$^2$Q+ experiment results are shown in the following figure:

The following conclusions can be summarized from the figure:

- When the sending rate and Lmid are different, the delay of core real-time traffic data flow 0 is at the lowest level. When the sending rate is constant, the delay is almost stable near a value.

- When the sending rate is the same, the delay of non-core real-time service data flow 1 and 2 is positively correlated with Lmid. This shows that when Lmid is small, the delay of non-core real-time services will be relatively reduced, while the delay of the other three non-real-time data flows will increase.

- When the Lmid is in the range of 5-20 and the sending rate is low, it can be found that the delay of non core real-time services 1 and 2 is in the second and third low. This advantage will decrease as the sending rate increases, but the total delay will decrease. Therefore, in the case of low sending rate, the algorithm allocates more bandwidth to these non-core real-time service data flows by increasing the weight of the non-core real-time service data flows.

- In the case of a fixed sending rate, with the increase of Lmid, the delay will be more and more inclined to the fair scheduling algorithm in the case of fixed weight. Especially when the sending rate
is fixed and $L_{mid}$ is equal to 50, it can be seen from the figure that the algorithm degenerates into a
fair queue scheduling algorithm, and the delay is negatively related to the weight. Therefore, this value
can be adjusted to determine whether to make non-core real-time service data flow more sensitive to
backlog length.

5. Summary
The improved WF2Q + algorithm can dynamically sense the backlog of real-time data flow. Especially
when the sending rate is low, the priority is to ensure the core real-time queue, and the weight of the
non-core real-time queue is dynamically increased. Therefore, the algorithm has both real-time
guarantee and certain fairness. Through the token bucket algorithm, a large number of bursty data flow
can be effectively prevented from being sent to the network. In practical applications, the value of
$L_{mid}$ can be modified as needed to adjust the sensitivity of the non-core real-time queue to the
backlog length.

References
[1] Parekh, A.K., Gallager, R.G. (1992) A Generalized processor sharing approach to flow
control in integrated services networks. [Proceedings] IEEE INFOCOM ’92: The
Conference on Computer Communications, 2: 915-924.
[2] Ma, Y. (2007) Research on the wired packet scheduling algorithm based on GPS. Tianjin
University, Tianjin.
[3] Bennett, J.C.R., Zhang, H. (1997) Hierarchical Packet Fair Queuing Algorithms.
IEEE/ACM Transactions on Networking, 5: 675-689.
[4] Zhong, S., Yue, X. (2006) Research on WFQ Traffic Scheduling Algorithm. Research on
Optical Communications, 32: 19-21.
[5] Lin, Y.W., Luo, Y.Y. (2017) Research on WFQ algorithm based on priority bandwidth
guarantee and weight improvement. Journal of Guangxi University for Nationalities
(Natural Science Edition), 23: 82-86.
[6] Qin, J.X., Song, J.L., Dai, X.W. (2019) Application Research of Improved WFQ Algorithm
in Train Network Scheduling. Application Research of Computers, 36: 3302-3305.
[7] Bennet, J.C.R. (1996) WF2Q:Worst case fair weighted fair queuing. In: Proceedings of
IEEE INFOCOM ’96. Conference on Computer Communications. San Francisco, CA,
USA. pp. 120-128.
[8] Song, M., Song, J., Li, H.W. (2003) Implementing a high performance scheduling
discipline WF2Q+ in FPGA. In: Canadian Conference on Electrical and Computer
Engineering. Canada. pp. 187-190.
[9] Zheng, F.F. (2016) QoS Implementation Based on W–2Q+ Scheduling Algorithm.
Heilongjiang Science and Technology Information, 10: 141-141.
[10] Meng, G.D. (2015) Research on detection queue scheduling algorithm of video surveillance
intelligent detection system. Ocean University of China, Qingdao.