# CHAPTER 3

# PROFICIENT RATE CONTROL (PRC) ALGORITHM FORCONGESTIONCONTROLINWSN

## 3.1 INTRODUCTION

In Real-Time (RT) applications, Wireless Sensor Networks are capable of producing awide range of data packet types. Limitations in available network capacity necessitate theusage of priority queues to manage the flow of data in a WSN. Congestion managementstrategies that allow for the relative importance of RT packet traffic have evolvedduring the past few decades. Rather, it needs to effectively handle congestion brought on by a mix of non-real-time (NRT) and real-time (RT) packets.

To resolve this concern, The Difference of Differential Rate (DDR) of a node and the Weighted Priority (WP) of the traffic class were both made possible by the Weighted Priority Difference of Differential Rate Control (WPDDRC) technology (Swain S. K et. al.,).Nevertheless, it does not allow the buffer occupancy and queue size,which might cause significant packet loss and delay. The buffer is full while the queuelength is high. Here, a WPDDRC-based priority based queuing system as a Control PRCtechnique to address this queue length problem. This method allows two different virtual queues on a single physical queue, so that the input packets from each child node can be aggregated in a way that responds to the importance and urgency of the traffic entering from each node.The PRC checks for congestion in the virtual queue after receiving the packetand then adjusts the child's transmission rate accordingly. This PRC technique, which takes into justification the priority of each traffic type and the current queue status, may result in buffer overflow and congestion in WSNs.

The research work comprises various methodological phases, which are briefly illustrated, and the overall flow of this research is depicted in Figure 3.1.

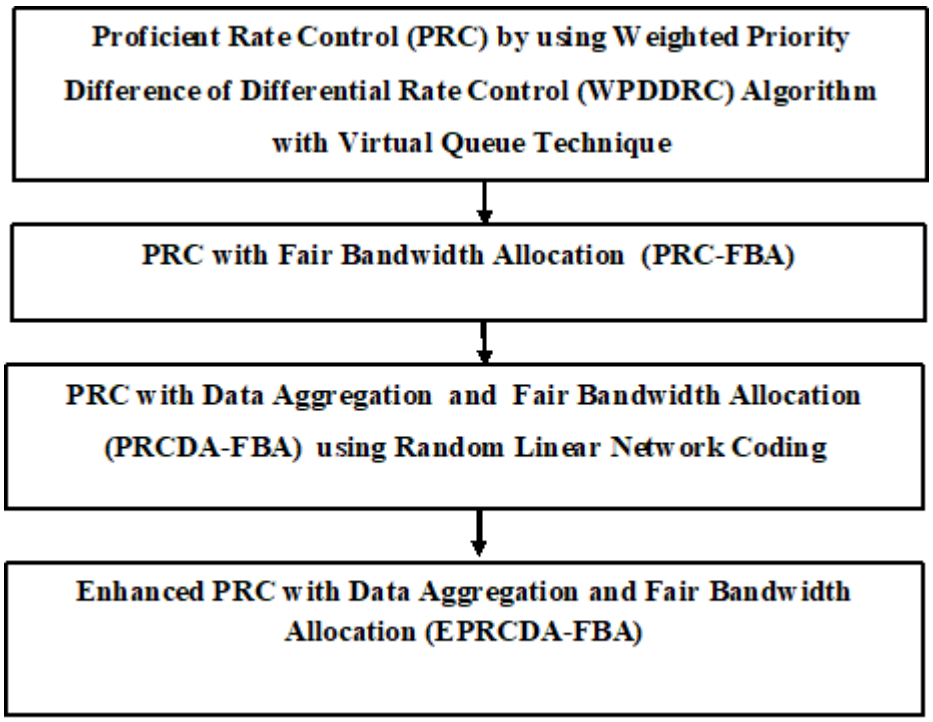


**Figure 3.1 Overall Flow of the Research work**

### 3.1.1 Proficient Rate Control (PRC) by using Virtual queue Techniques for Congestion Control inWSN

Although a large queue can occasionally fill up more than half the buffer, leading toconsiderablepacketlossanddelay,aPRCmethodisproposedbythePriority Difference of DifferentialRateControl**(**WPDDR)withanadaptivequeuingsystem.WiththePRCalgorithm,th eincoming packets from each child node are accumulated in one of two virtual queues on a singlephysicalqueue:oneforlow prioritytrafficandanotherforhigh prioritytraffic.Congestion in the virtual queue is detected when a packetis successfully received andthe PRC responds by adjusting the transmission rate of child node. It allowed therelative importance of each type of traffic and the current queue status, this PRC methodhas the potential to mitigate the negative effects of congestion and buffer overflow inWSNs.

## 3.2 PRIORITYBASEDRATECONTROLSCHEMES

In the Difference of Differential Rate control (DDRC) Algorithm, a Sink and a Source Node Directed Data Replication Difference of Differential Rate control (DDR) difference is calculated using Differential Rate Control. A node rate is determined by DDRC by using both its Global Priority and the variation in traffic volumes between the

node and the sink.This technique can be theoretically used to control higher-order derivatives of data rates from nodes near the destination and the source.It can be used to manage single Real Time packets, Real Time and Non-Real Time pairings.

The DDR at the sink and the traffic class have been combined to create a WPDDRCalgorithm. Traffic classes having higher order DRC linked to different nodes are assigned a weighted priority, WPDDRC modifies the global priority in order to grant RT traffic class priority over the NRT packets. It was achieved by prioritizing certain traffic types that were already connected to particular nodes. Higher order derivative rate control and a modified total priority form the foundation of the second strategy. However, it does not support queue size or buffer occupancy, which could result in considerable packet loss and delay if the queue is too long.

In the context of WSNs, traffic class refers to the categorization of data traffic based on certain characteristics such as priority, delay sensitivity, reliability requirements, and application type. Managing different types of traffic classes is crucial in WSNs to ensure efficient resource utilization, Quality of Service (QoS) requirements, and prolong network lifetime. Some common traffic classes in WSN are:

- Event-driven traffic
- Periodic traffic
- Query-driven traffic
- Control traffic: Control messages are used for network management, synchronization, routing updates, and other administrative purposes. These messages are essential for maintaining network connectivity and stability
- Emergency or critical traffic
- Background traffic
- Maintenance traffic

Efficient handling of traffic classes in WSNs often involves employing appropriate routing protocols, scheduling mechanisms and Quality of Service (QoS)mechanisms tailored to the specific requirements of each class. This ensures that critical data is delivered promptly, while non-critical data is transmitted efficiently without causing undue congestion or resource contention.Here, proposed a WPDDRC with a priority

queuing system,also known as a PRC algorithm. The PRC method divides asingle physical queue into two virtual queues, one for low-priority traffic and one forhigh-priority traffic, depending on the type of packet it is. Once the packet has beenreceived, the PRC determines whether or not there is congestion in the virtual queue andadjusts the child's transmission rate accordingly. This PRC approach may control bufferoverflow and congestion in WSNs by considering the priority of each traffic type and thecurrentqueuestatus.

## 3.3 PROPOSED METHODOLOGY

The primary goal of the algorithm is to handle various NRT packet types. The packet categories that are taken into consideration are High priority NRT (HNRT), Medium priority NRT (MNRT), and Low priority NRT (LNRT).These packets are dispersed according to a specific priority range. As a result, the data rates within various values are used to identify the packets.

The concept of higher order derivative is applied to RT traffic in order to calculate the rate of virtual queues in a certain node. The difference of differentials with respect to the sink is used to compute this. Additionally, weights are applied to the priorities of various traffic classes to generate new weighted traffic class priorities.The general network topology(tree topology) is shown in Figure 3.2, where $P_1$, $P_2$ and $P_3$ are parent nodes and $C_1$-$C_8$are child nodes. During packet transmission, the parent nodes forward the higher priority packets to the sink node, while the child nodes prioritize the traffic classes.

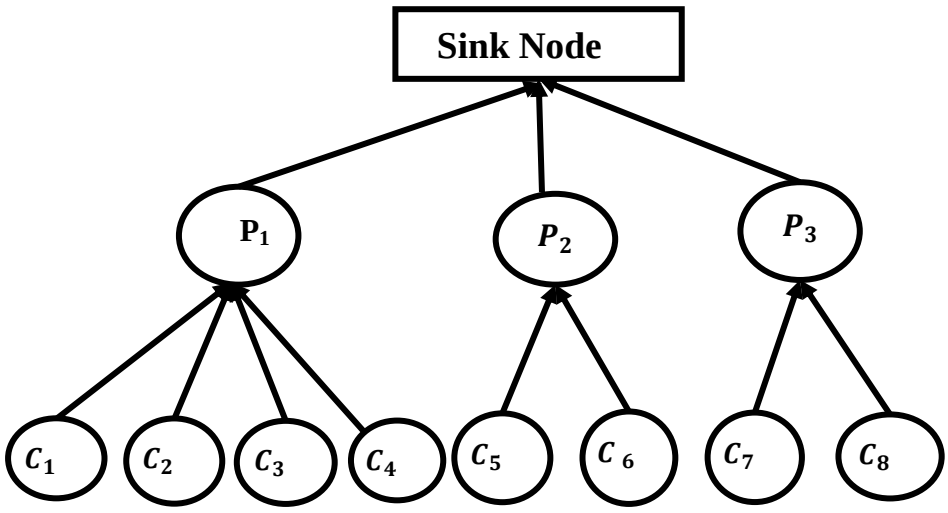


**Figure 3.2. General Network Topology**

Considering$TC\mathcal{P}t_n^k$ and $G\mathcal{P}t_n^k$ are the traffic class priority and the geographical priority $n^{th}$ virtual queue in $k^{th}$ node, accordingly. Also, consider and $\mathcal{SP}t_i^{k_n}$ is the traffic source priority of $n^{th}$ virtual queue in $k^{th}$ node where $i$ is the set of traffic classes and $i \in \{RT, HNRT, MNRT, LNRT\}$.

Initially, the traffic class priority of $n^{th}$ virtual queue in $k^{th}$node is calculated as:

$$TC\mathcal{P}t_n^k = \sum_n \sum_i \mathcal{SP}t_i^{k_n} \tag{3.1}$$

Since the RT traffic class is mostly responsible for the overall priority, the RT is given the highest priority. Furthermore, the overall traffic class priority contributing the overall priority of $n^{th}$ virtual queue in $k^{th}$ node is minimized by the weighted total number of the NRT traffic classes. Therefore, the weighted overall priority for $n^{th}$ virtual queue in $k^{th}$node ($\mathcal{WP}_n^k$) is computed as:

$$\mathcal{WP}_n^k = TC\mathcal{P}t_n^k \cdot G\mathcal{P}t_n^k + [\mathcal{W}_{RT} - \delta(\mathcal{W}_{HNRT} + \mathcal{W}_{MNRT} + \mathcal{W}_{LNRT})] \tag{3.2}$$

In Eq(3.2), $\delta$ indicates the constant $(0 \leq \delta \leq 1)$ and $\mathcal{W}_{RT}$, $\mathcal{W}_{HNRT}, \mathcal{W}_{MNRT}, \mathcal{W}_{LNRT}$ are the weights assigned to the "RT and NRT" traffic classes. Similarly, the traffic class priority of $n^{th}$ virtual queue in $l^{th}$ child node is computed as:

$$TC\mathcal{P}t_n^l = \sum_n \sum_i \mathcal{SP}t_i^{l_n} \tag{3.3}$$

In Eq. (3.3), $\mathcal{SP}t_i^{l_n}$ is the source priority of $n^{th}$ virtual queue in $l^{th}$ child nodes. The weighted overall priority for $n^{th}$ virtual queue in $l^{th}$ child node ($\mathcal{WP}_n^l$) is,

$$\mathcal{WP}_n^l = TC\mathcal{P}t_n^l \cdot G\mathcal{P}t_n^l + [\mathcal{W}_{RT} - \delta(\mathcal{W}_{HNRT} + \mathcal{W}_{MNRT} + \mathcal{W}_{LNRT})] \tag{3.4}$$

In Eq. (3.4), $\delta$ indicates the constant $(0 \leq \delta \leq 1)$ and $G\mathcal{P}t_n^l$ is the geographical priority of $n^{th}$ virtual queue in $l^{th}$ child node. The weighted global priority of $n^{th}$ virtual queue in $l^{th}$ child node ($\mathcal{WGP}_n^l$) changes to $\mathcal{WGP}_n^l = \mathcal{WP}_n^l$. The weighted global priority at the $n^{th}$ virtual queue of $k^{th}$ parent node $\mathcal{WGP}_n^k$ changes to,

$$\mathcal{WGP}_n^k = \sum_n \sum_{l \in \mathcal{C}(k)} \mathcal{WGP}_n^l + \mathcal{WP}_n^k \tag{3.5}$$

The rate of thehighest output$n^{th}$ virtual queue in $k^{th}$ weighted priority of parent node is computed as follows:

$$OR_n^k = OR_n^{sink} \cdot \frac{\mathcal{WGP}_n^k}{\mathcal{WGP}_n^{sink}} \tag{3.6}$$

In Eq. (3.6), $\mathcal{WGP}_n^{sink}$ is the weighted global priority of the $n^{th}$ virtual queue in the sink node and is calculated as:

$$\mathcal{WGP}_n^{sink} = \sum_n \sum_{k \in \mathcal{C}(sink)} \mathcal{WGP}_n^k \tag{3.7}$$

The Eq. (3.7) is the total number of the weighted global priority of the linked child nodes of the sink. Also, the input rate of the $n^{th}$ virtual queue in the sink node is given as:

$$IR_n^k = \sum_n \sum_{k \in \mathcal{C}(sink)} OR_n^k \tag{3.8}$$

In Eq. (3.8), $OR_n^k$ corresponds to the output rate of every child node. Using the modified global priority of Eq. (3.5) and rate of output(3.6), the updated rates of $n^{th}$ virtual queue in the parent nodes and the sink node are calculated as:

$$\Delta R_n^{sink} = \beta \cdot OR_n^{sink} - IR_n^{sink} \tag{3.9}$$

$$OR_n^k = OR_n^k + \Delta R_n^{sink} \cdot \frac{\mathcal{WGP}_n^k}{\mathcal{WGP}_n^{sink}} \tag{3.10}$$

$$\Delta R_n^k = \beta \cdot OR_n^k - IR_n^k \tag{3.11}$$

The updated rate is represented by,

$$OR_n^l = OR_n^l + \Delta R_n^k \frac{\mathcal{WGP}_n^l}{\mathcal{WGP}_n^k} + \mu \left[ \left( \Delta R_n^{sink} - \Delta R_n^k \right) \frac{\mathcal{WGP}_n^l}{\mathcal{WGP}_n^k} \right] \tag{3.12}$$

The corresponding parent node transmits the updated rate to its child node. By doing so, the network ensures the allocation of an appropriate data rate, effectively managing network congestion. This careful allocation helps to prevent issues such as buffer overflow and packet loss during data transmission, thereby maintaining efficient network performance and stability.

The working procedure of the proposed Proficient Rate Control (PRC) algorithm is given in Algorithm 3.1.

**Algorithm 3.1: Proficient Rate Control (PRC)**

***Step 1:*** Initialize the parameters: service time $\left(ST_n^{sink}\right)$, β, δ, μand the traffic class priorities.

***Step 2:*** Computed the mean service time of $n^{th}$ virtual queue in the sink node as:

$$\overline{ST}_n^{sink}(t+1) = (1-\alpha)\overline{ST}_n^{sink}(t) + \alpha \cdot ST_n^{sink} \quad (3.13)$$

***Step 3:*** Calculated the variance of rate $n^{th}$ virtual queue in the sink node and $k^{th}$ parent nodes by using Eq. (3.9) & Eq. (3.11).

***Step 4:*** Calculated the updated output rate of $n^{th}$ virtual queue in the $k^{th}$ parent node using Eq. (3.10).

***Step 5:*** Calculated the update rate of $n^{th}$virtual queue in the $k^{th}$ parent node propagated to the $l^{th}$ child node using Eq. (3.12).

***Step 6:*** Continued *Steps 2* to *Steps 5* till completion of the specified simulation period.

Therefore, in this algorithm, one of the major contributions is the weighted global priority of $n^{th}$virtual queue in $K^{th}$parent and $l^{th}$ child node as given in Eq. (3.2) & Eq. (3.4). Also, the novelty in the rate computation is the weighted global priority together with the difference of differential as given in Eq. (3.12). The difference of differential is the last term of Eq. (3.12).

**Major Contributions of the Algorithm:**

- **Weighted Global Priority:** A significant contribution of this algorithm is the implementation of the weighted global priority for the $n^{th}$ virtual queue in both $k^{th}$ parent and $l^{th}$ child nodes, as described by Eq. (3.2) and Eq. (3.4). This priority system helps in efficient rate allocation based on the importance of different queues.
- **Novel Rate Computation:** The algorithm introduces a novel method for rate computation, which incorporates the weighted global priority along with the difference of differentials, as shown in Eq. (3.12). The term difference of differentials refers to the last term in Eq. (3.12), which fine-tunes the rate based on dynamic network conditions.

Figure 3.3 presents the flowchart of the PRC Algorithm for Congestion Control.

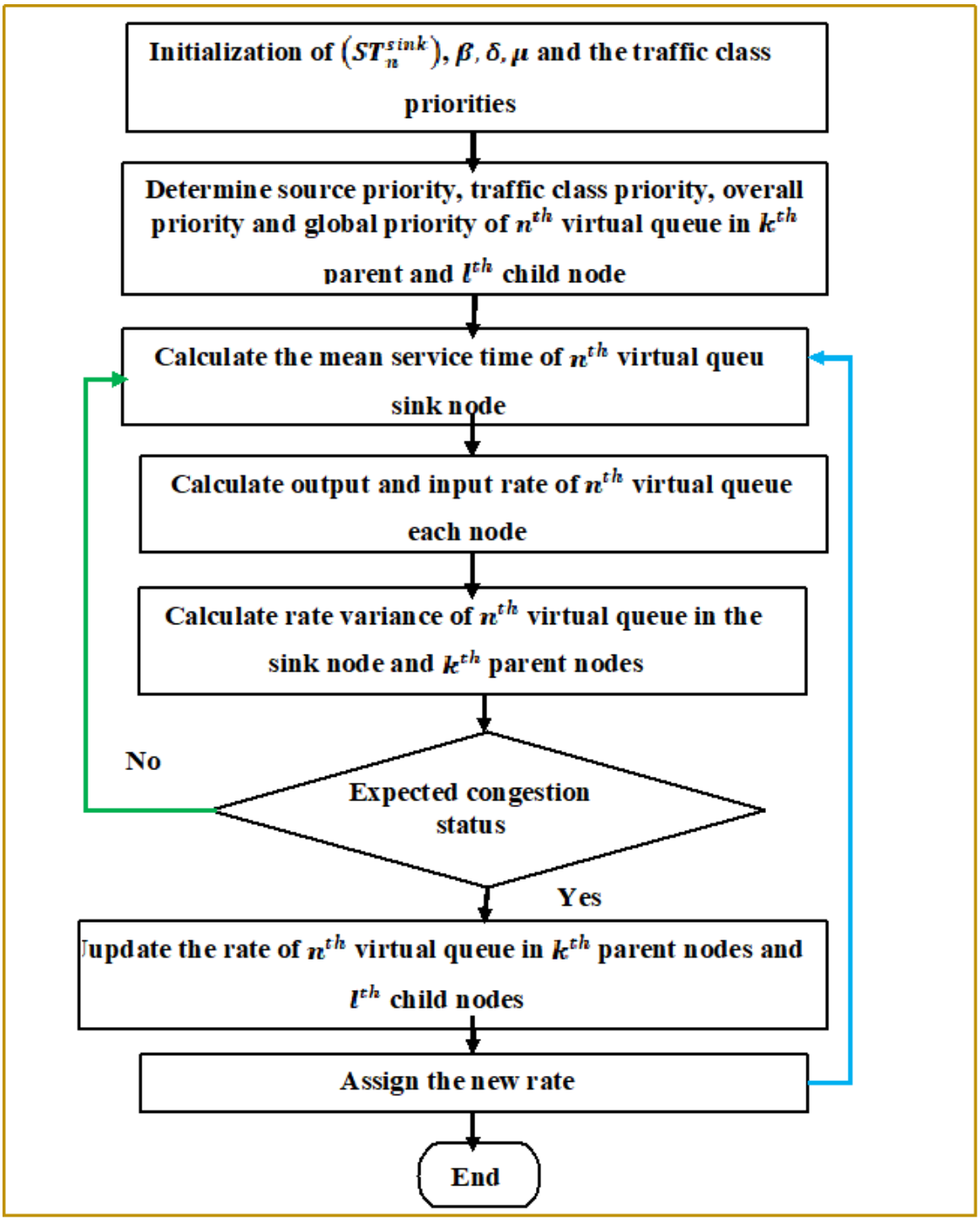


**Figure 3.3. Flowchart of PRC Algorithm for Congestion Control**

## 3.4 EXPRIMENTAL SETUP

Table 3.1 provides the simulation settings used in this study with Network Simulator 2.35 (NS2.35), encompassing various parameters essential for simulating network behavior and evaluating performance.

**Table 3.1. Simulation Parameters**

| Parameter | Range |
|---|---|
| Transmissionpower | 285.63mW |
| MAClayer | IEEE802.11 |
| Numberofnodes | 900nodes |
| Trafficsource | CBR |
| Datarate | 2Mbps |
| Packet size | 200bytes |
| Communicationrange | 300m |
| Numberoftrafficcategories | 4 |
| Simulationarea | 1000×1000m |
| Routing protocol | AODV |
| Operatingfrequency | 5GHz |
| Simulationtime | 120sec |
| Mobilityspeed | 10m/s |
| Mobilitymodel | Randomwalk |

## 3.5 PERFORMANCE METRICS

In order to demonstrate the effectiveness of the proposed algorithm, various performance metrics such as throughput, packet loss, end-to-end delay, queue size, and data transmission rate are considered.

### 3.5.1 Throughput

It defines the number of packets received by the target within a given period.

$$Throughput = \frac{Total\ number\ of\ packets\ received}{Time} \quad (3.1)$$

### 3.5.2 Packet Loss

It was indicated the quantity of packets dropped during a conversation. It was computed as:

$$packet\ loss = \frac{Number\ of\ dropped\ packets}{Number\ of\ dropped\ packets + Number\ of\ received\ packets} \quad (3.2)$$

### 3.5.3 End-to-end Delay

It defines the time taken for a packet to be sent from a source node to the sink.

$$E2E\ Delay = Time_{sink} - Time_{source} \quad (3.3)$$

In Eq. (3.3), $Time_{sink}$ is the time at the sink node when receiving the packet and $Time_{source}$ is the time at the source node when transmitting that packet.

### 3.5.4 Queue Size

It defines the number of packets in the queue. It is a key metric to estimate the delay. If the queue size is large, then it causes more delay.

### 3.5.5 Data Transfer Rate

It is the data transfer rate of origin, which handles the congestion and buffer overflow in WSN.

## 3.6 Performance Evaluation

The performance of proposed PRC algorithm was compared to that of the WPDDRC andthe Difference of differential rate control scheme (DDRC). Toevaluate and contrast the proposed work, metrics such as throughput, End-to-End (E2E) delay, queue size, packet loss, and source data transmission rate adaptation are used.

### 3.6.1 Throughput

A comparison was conducted on the throughput of DDRC, WPDDRC and PRC over a specific number of iterations. The throughput values for the proposed PRC algorithm, as well as the existing DDRC and WPDDRC algorithms, are detailed in Table 3.2.

**Table 3.2 Throughput Comparison**

| Simulation Time(sec) | DDRC (Kbps) | WPDDRC (Kbps) | PRC (Kbps) |
|---|---|---|---|
| 20 | 331 | 354 | 375 |
| 40 | 359 | 385 | 408 |
| 60 | 382 | 406 | 429 |
| 80 | 410 | 429 | 450 |
| 100 | 432 | 448 | 472 |
| 120 | 455 | 470 | 490 |

The results given in the Table 3.2 presented the throughput performance of DDRC, WPDDRC, and PRC algorithms across different simulation durations. Notably, the PRC algorithm exhibited superior throughput compared to alternative congestion control methods. For instance, at a simulation time of 120 seconds, PRC achieved a throughput 7.69% higher than DDRC and 4.26% higher than WPDDRC. This enhancement can be attributed to the dynamic adjustment of traffic class priorities within each nodes virtual queue, a feature unique to the PRC algorithm.

### 3.6.2 Packet Loss

Here, a number of iterations wasused to compare Packet Loss between DDRC, WPDDRC and PRC.Thepacketloss comparisons between the proposed PRC, DDRC and WPDDRC algorithm,alongwiththenumberofiterationswasprovidedinTable 3.3.

**Table 3.3 Packet Loss Comparison**

| Simulation Time(sec) | DDRC (%) | WPDDRC (%) | PRC (%) |
|---|---|---|---|
| 20 | 8.8 | 6.5 | 4.4 |
| 40 | 11.3 | 8.7 | 6.4 |
| 60 | 15.8 | 13.4 | 10.6 |
| 80 | 19.5 | 17.2 | 15.1 |
| 100 | 24.6 | 21.6 | 19.5 |
| 120 | 30.2 | 27.9 | 25.2 |

The findings displayed the percentage of packet loss across different simulation durations for the WPDDRC, PRC, and DDRC algorithms. The study revealed that in terms of packet loss, the PRC approach outperformed the other algorithms. Specifically, with a simulation duration of 120 seconds, the packet loss with PRC was 16.57% lower than DDRC and 9.68% lower than WPDDRC. This superior performance of PRC in minimizing packet loss can be attributed to its implementation, where each node autonomously manages the prioritization levels of traffic classes through distinct virtual queues.

### 3.6.3 End-to-end Delay

The End-to-End delay of DDRC, WPDDRCand PRC are compared for definite number of iterations and is given inTable 3.4.

**Table 3.4 E2E Delay Comparsion**

| Simulation Time(sec) | DDRC(ms) | WPDDRC (ms) | PRC (ms) |
|---|---|---|---|
| 20 | 97 | 90 | 83 |
| 40 | 119 | 110 | 101 |
| 60 | 146 | 140 | 130 |
| 80 | 158 | 150 | 142 |
| 100 | 185 | 177 | 170 |
| 120 | 198 | 190 | 183 |

The study displayed the end-to-end (E2E) latency for the DDRC, WPDDRC, and PRC algorithms across different simulation durations, measured in milliseconds. According to the findings, the PRC approach exhibited the smallest E2E delay. By the simulation time 120 sec, the E2E delay with PRC was determined to be 7.58% less than that of both DDRC and WPDDRC.

### 3.6.4 Queue Size

The queue size of DDRC, WPDDRC andPRC are compared with definite number of iterations. Table 3.5 provided the comparisonvalues of queue size for proposed PRC with the existing DDRC, WPDDRCalgorithm withthe numberofiterations.

**Table 3.5 Queue size Comparison**

| Simulation Time(sec) | DDRC (Pkts) | WPDDRC (Pkts) | PRC (Pkts) |
|---|---|---|---|
| 20 | 14 | 9 | 5 |
| 40 | 17 | 13 | 8 |
| 60 | 20 | 16 | 11 |
| 80 | 23 | 20 | 14 |
| 100 | 27 | 24 | 18 |
| 120 | 31 | 28 | 22 |

From this comparison, it was observed that the PRC algorithm exhibited the least efficiency in reducing the average queue length compared to all queueing techniques. Specifically, in a 120-second simulation, the mean queue size with PRC was found to be 29.03% smaller than that of DDRC and 21.43% smaller than that of WPDDRC. Consequently, the shorter queue length with PRC suggests a potential reduction in packet loss and end-to-end latency. It is evident that PRC offers a more consistent mean queue size and has the ability to stabilize the queue length at a targeted value.

### 3.6.5 Data Transfer Rate

Therateof data transmissionadjustmentofDDRC,WPDDRC and PRC was compared with definite iterations.Table 3.6 providedthecomparisonvaluesof datatransmissionrateadjustmentforproposedPRCwith the existingDDRC,WPDDRCalgorithmswiththenumberofiterations.

**Table 3.6 Data Transfer Rate Comparison**

| Simulation Time(sec) | DDRC (Pkts/s) | WPDDRC (Pkts/s) | PRC (Pkts/s) |
|---|---|---|---|
| 20 | 58 | 63 | 68 |
| 40 | 55 | 60 | 64 |
| 60 | 52 | 57 | 60 |
| 80 | 49 | 54 | 57 |
| 100 | 46 | 51 | 54 |
| 120 | 43 | 49 | 51 |

By managing queue length and prioritizing traffic appropriately to adjust rates, the PRC algorithm demonstrated superior performance in achieving the highest overall source data transfer rate in the analysis. Specifically, when the simulation time was set to 120 seconds, the PRC source rate was found to be 18.6% higher than that of both DDRC and WPDDRC. It is evident that PRC progressively slows down the data transfer rate compared to the initial rate of nodes. This prioritization ensures that the most critical traffic types are transported first before the transfer rate is decreased.