NS2 REPORT

RED

RED (Random Early Detection) is among the first Active Queue Management (AQM) algorithms. RED operates at the output port during the ‘enqueue’ time. When a new packet arrives, RED makes a decision; whether this packet should be enqueued or dropped? It makes this decision even if there is space in the queue. It makes this decision because if enqueuing the packet increases overall delay or makes the queue full then it will cause congestion to occur.

RED consists of following components -

- Calculation of average queue length.
- Calculation of drop probability, whether the packet will be enqueued or dropped depends on this probability.
- Decision-making logic (helps to decide whether the incoming packet should be enqueued or dropped).

SRED

SRED (Stabilized Random Early Drop) is a modified version of red. Like RED (Random Early Detection) SRED preemptively discards packets with a load-dependent probability when a buffer in a router in the Internet or an Intranet seems congested. SRED has an additional feature that over a wide range of load levels helps it stabilize its buffer occupation at a level independent of the number of active connections.

ALGORITHMIC OVERVIEW

Simple way of comparing an arriving packet with a recent other packet is to compare it with a packet still in the buffer. This makes it impossible to compare packets more than one buffer drain time apart. To give the system **longer memory**, we augment the information in the buffer with a `zombie list`. We can think of this as a list of `M` recently seen flows, with the following extra information for each flow in the list: a `count and a timestamp`. Note that this `zombie list` is small and maintaining this list is not the same as maintaining per-flow state. We call the flows in the zombie list `zombies`.

The zombie list starts out empty. As packets arrive, as long as the list is not full, for every arriving packet the packet flow identifier (`flow id`) is added to the list, the `count` of that zombie is set to zero, and its `timestamp` is set to the arrival time of the packet.

Once the `zombie list` is full it works as follows: Whenever a packet arrives, it is compared with a randomly chosen `zombie` in the `zombie list`.

- **Hit** :

If the arriving packet's flow matches the zombie we declare a "hit". In that case, the `count` of the zombie is increased by one, and the `timestamp` is reset to the arrival time of the packet in the buffer.

- **No Hit** : If the two are not of the same flow, we declare a "no hit". In that case, with probability $p_{overwrite}$ the flow identifier of the packet is overwritten over the zombie chosen for comparison. The `count` of the zombie is set to 0, and the `timestamp` is set to the arrival time at the buffer. With probability $1 - p_{overwrite}$ there is no change to the zombie list.

Irrespective of whether there was a hit or not, the packet may be dropped and it will depend on p_{zap}. The drop probability p_{zap} may depend on whether there was a hit or no hit.

We maintain an estimated $P_{hit}(t)$ for the hit frequency around the time of the arrival of the t-th packet at the buffer. For the t-th packet, let,

$$\text{Hit}(t) = \begin{cases} 0 & \text{if no hit} \\ 1 & \text{if hit} \end{cases}$$

and let

$$P_{hit}(t) = (1-\alpha)P_{hit}(t-1) + \alpha \text{ Hit } (t)$$

where,

$$\alpha \sim \frac{p_{overwrite}}{M}.$$

Here, $0 < \alpha < 1$, and M is the size of `zombie list`

We have a buffer of capacity B bytes. A function $p_{\text{sred}}\,(q)$ is defined as follows:

$$p_{\text{sred}}\,(q) = \begin{cases} p_{\max} & \text{if } \frac{1}{3}B \leq q < B \\ \frac{1}{4} \times p_{\max} & \text{if } \frac{1}{6}B \leq q < \frac{1}{3}B \\ 0 & \text{if } 0 \leq q < \frac{1}{6}B \end{cases}$$

For **Simple SRED** mode, p_{zap} is calculated is as follows

$$p_{zap} = p_{sred}(q) \times \min\left(1, \frac{1}{(256 \times P_{hit}(t))^2}\right)$$

and for **Full SRED** mode,

$$p_{zap} = p_{\text{sred}} \times \min\left(1, \frac{1}{(256 \times P_{hit}(t))^2}\right) \times \left(1 + \frac{Hit(t)}{P_{hit}(t)}\right)$$

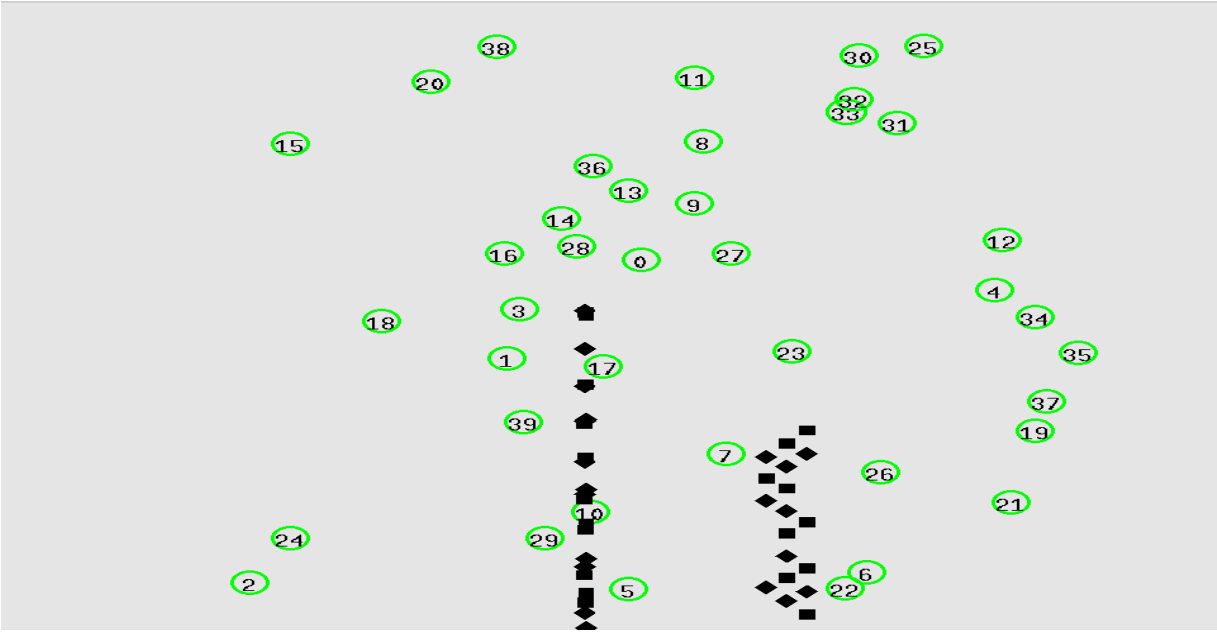
In the SRED paper, `timestamp` is not used as part of the experiments that have been conducted. We extend the SRED algorithm utilizing the `timestamp` of the zombie flows stored in our `zombie list`. We call this the Extended Stabilized RED algorithm or ESRED in short. In the original SRED algorithm, a fixed $p_{overwrite}$ is used to decide if a zombie flow needs to be overridden when there is no hit. Instead of using a fixed $p_{overwrite}$ when there is no hit, we dynamically change it according to the flow `timestamp` we are comparing with. We modify this $p_{overwrite}$ so that it is higher for flows with older `timestamp`.

TASK - 1

Topology

For task-1, the given network is **Wireless 802.11 (static)**. For a given number of nodes and flows, for each flow we randomly select two nodes and add a flow in them. So, they don't follow any specific topology.

For the transfer layer protocol we used UDP Protocol and for the application layer we used CBR Traffic. And in the network layer we used the DSDV Routing Protocol.



Parameters Under Variation

Baseline Parameters are -

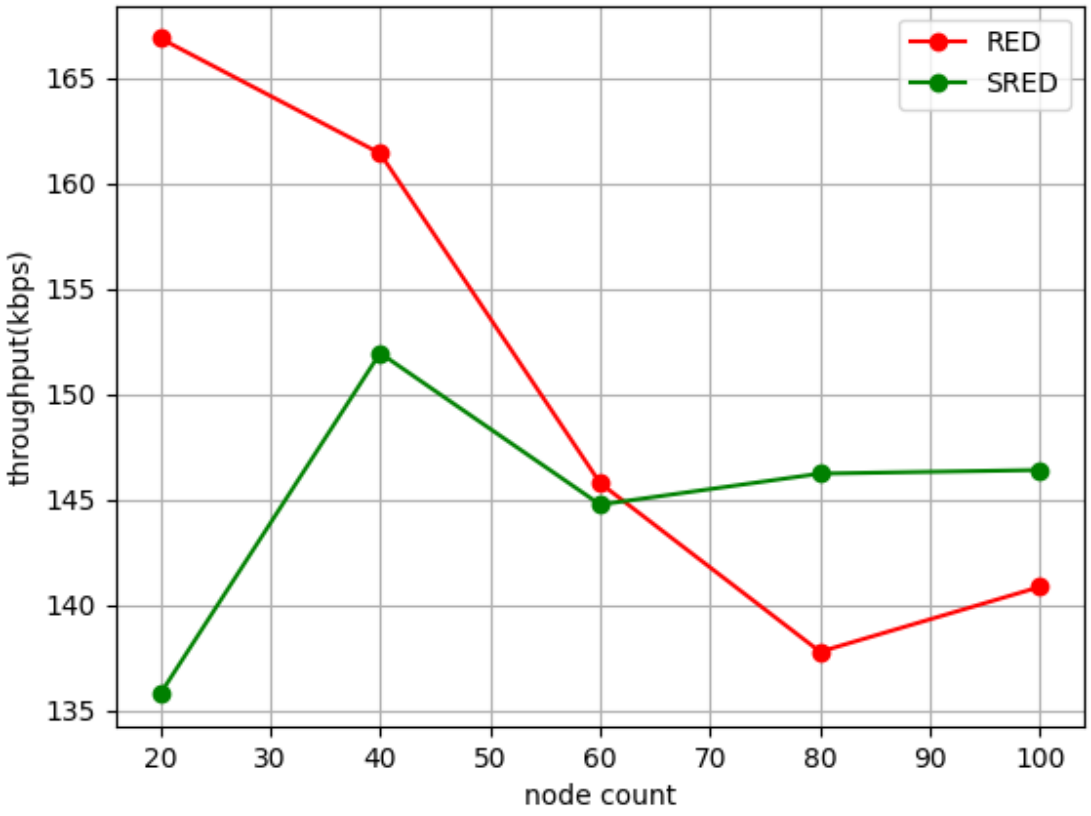
1. Number of Nodes : 40
2. Number of Flows : 30
3. Number of Packet per Second : 300
4. Coverage Area : 3 x TxRange

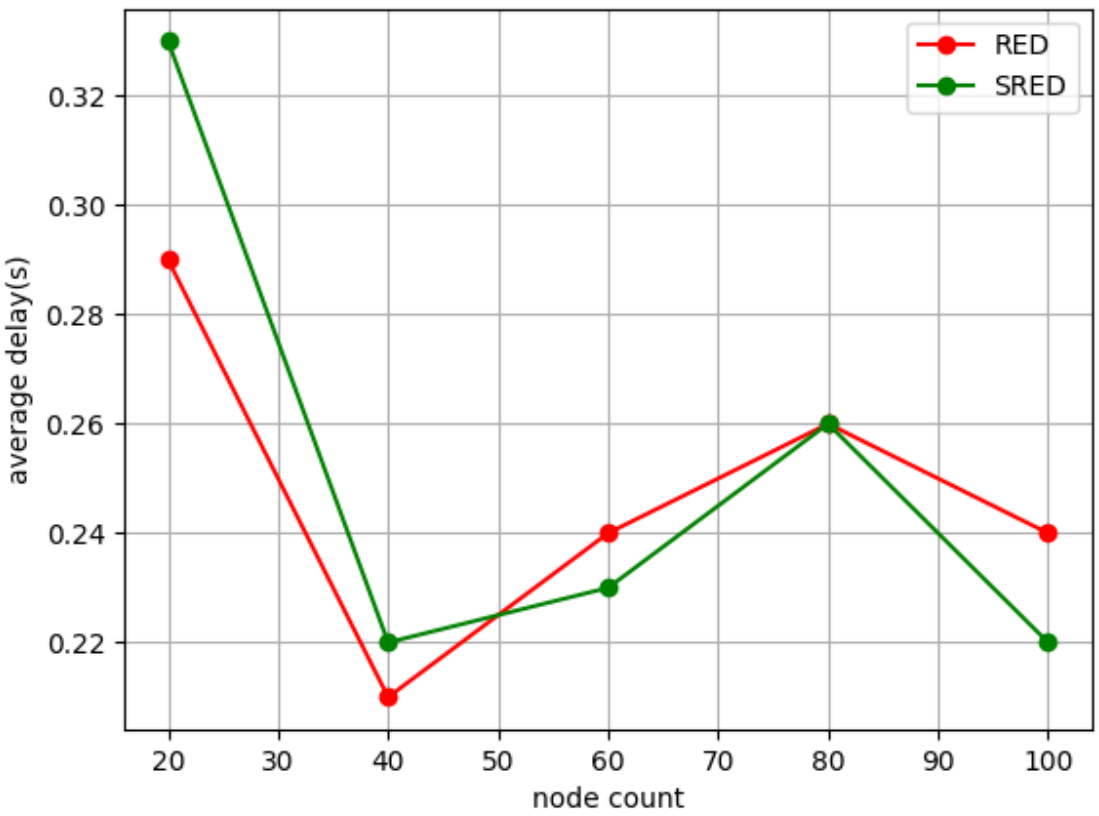
Varying Parameters -

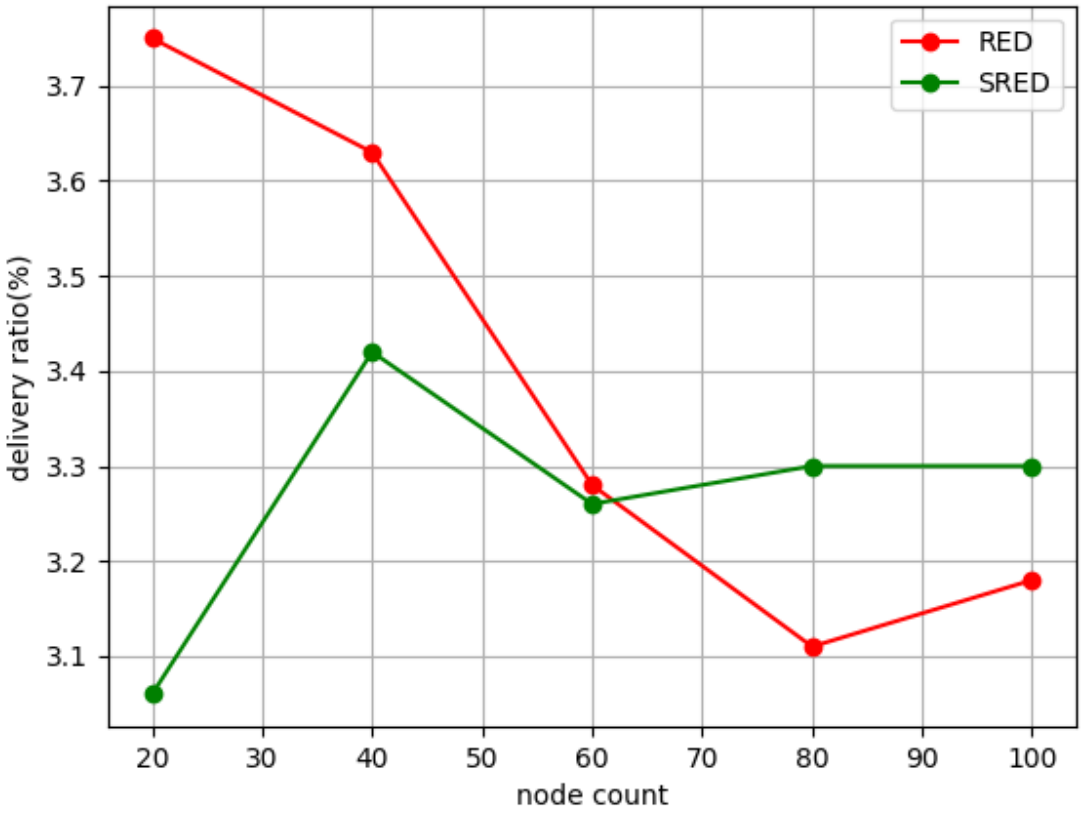
1. Number of Nodes : 20, 40, 60, 80, 100
2. Number of Flows : 10, 20, 30, 40, 50
3. Number of Packet per Second : 100, 200, 300, 400, 500
4. Coverage Area : 1 x TxRange, 2 x TxRange, 3 x TxRange, 4 x TxRange,5 x TxRange

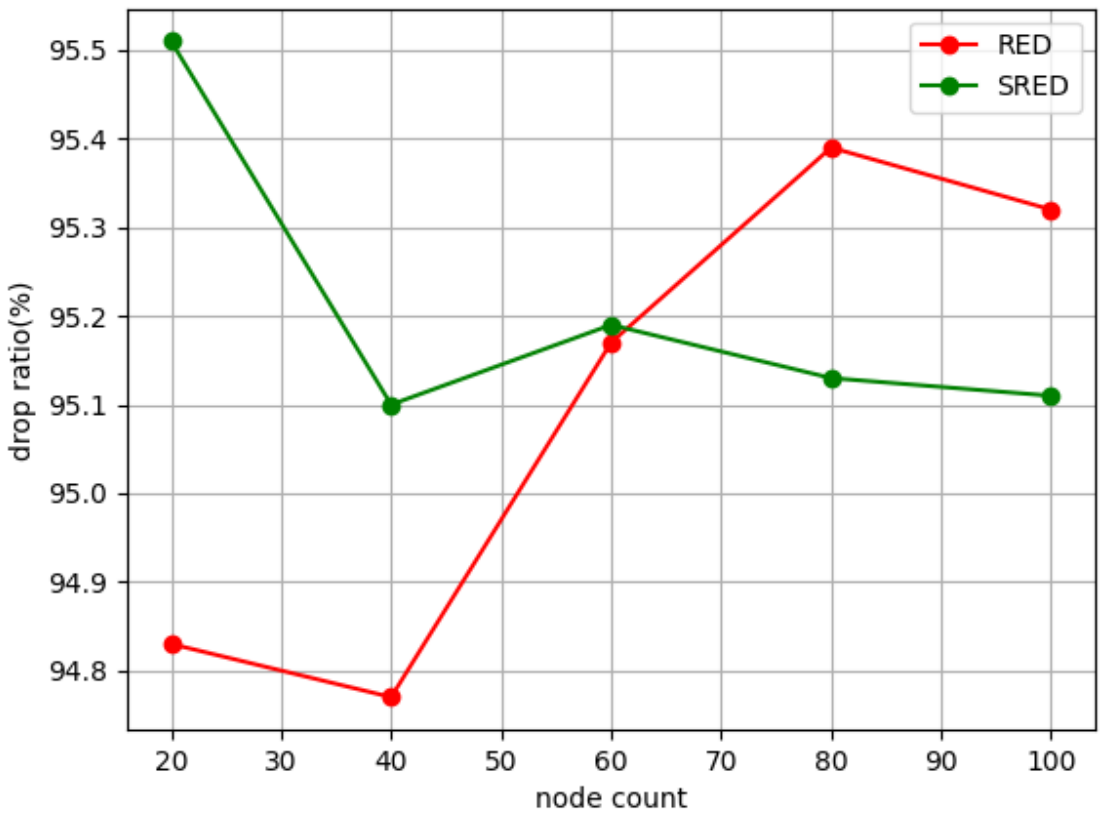
GRAPHS

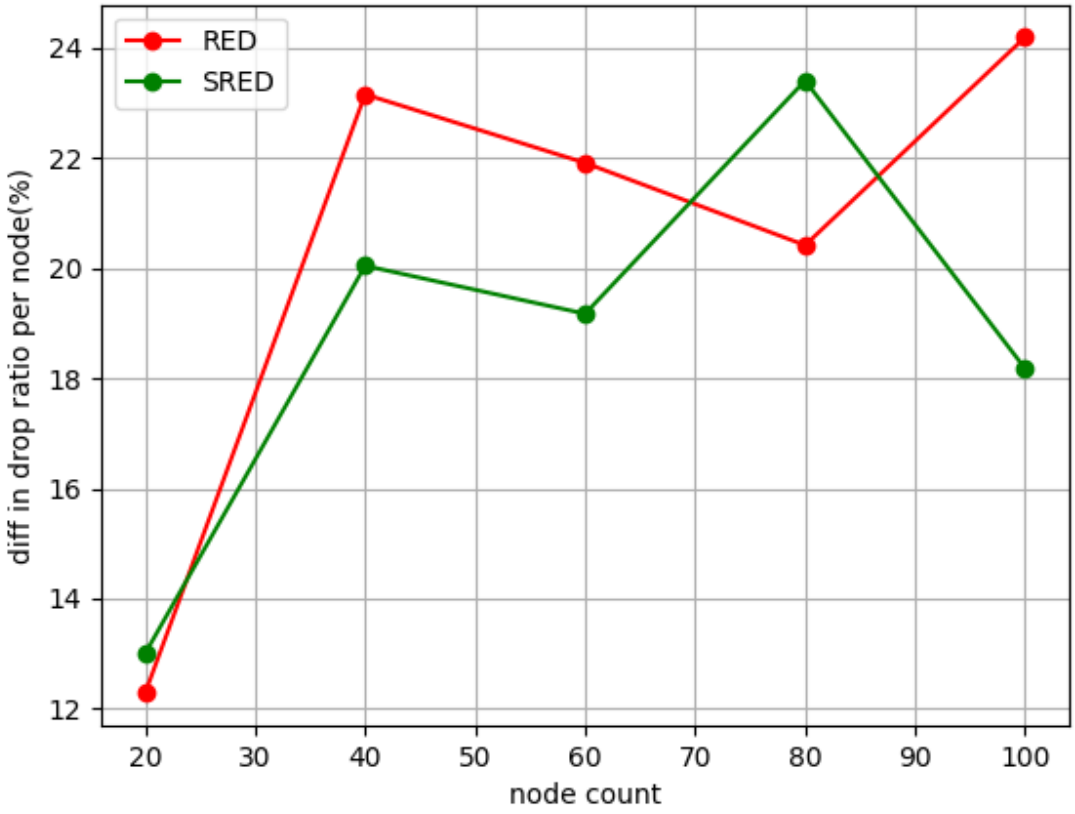
Vary Nodes

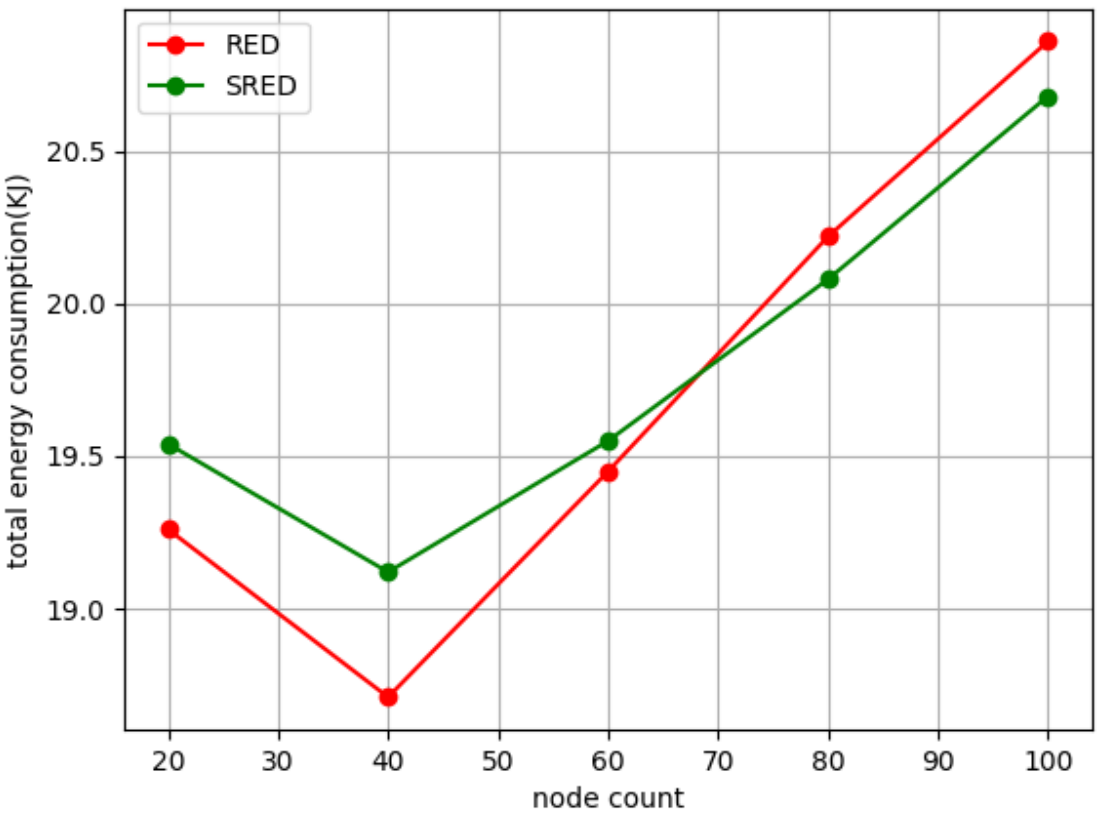



RED
SRED
0.32
0.30
0.28
0.26
0.24
0.22
average delay(s)
20
30
40
50
60
70
80
90
100
node count

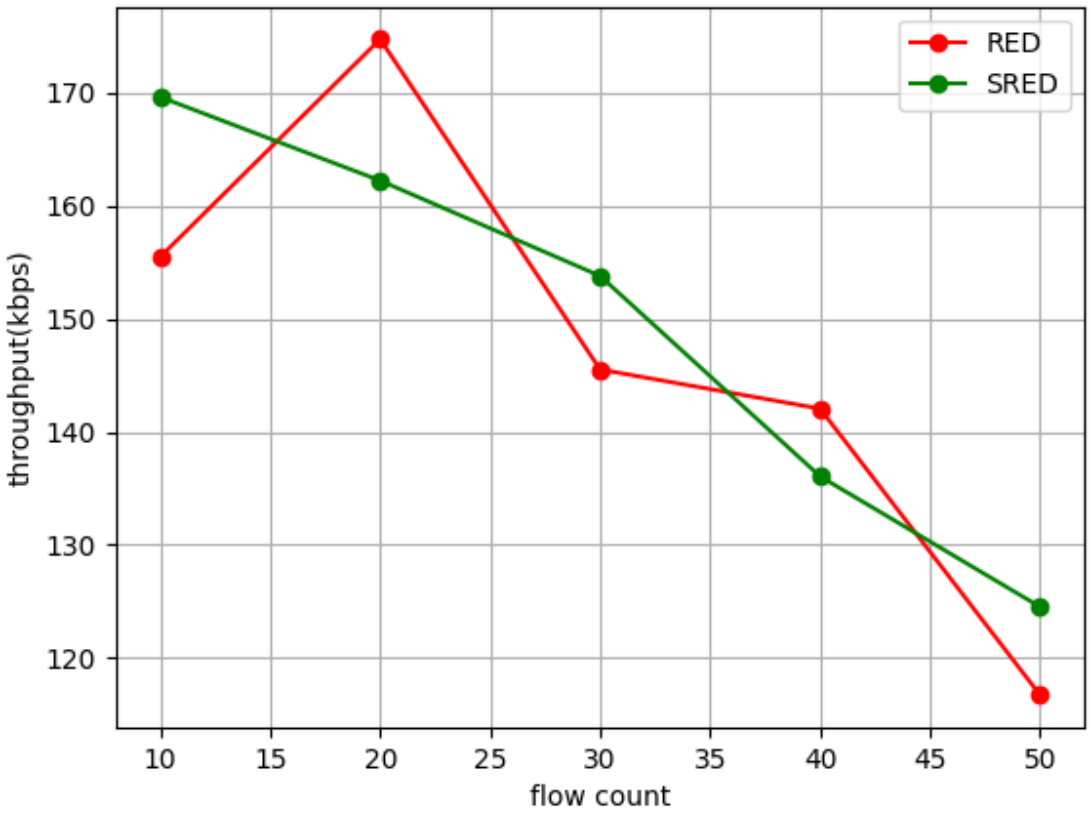

RED
SRED
delivery ratio(%)
3.7
3.6
3.5
3.4
3.3
3.2
3.1
20
30
40
50
60
70
80
90
100
node count

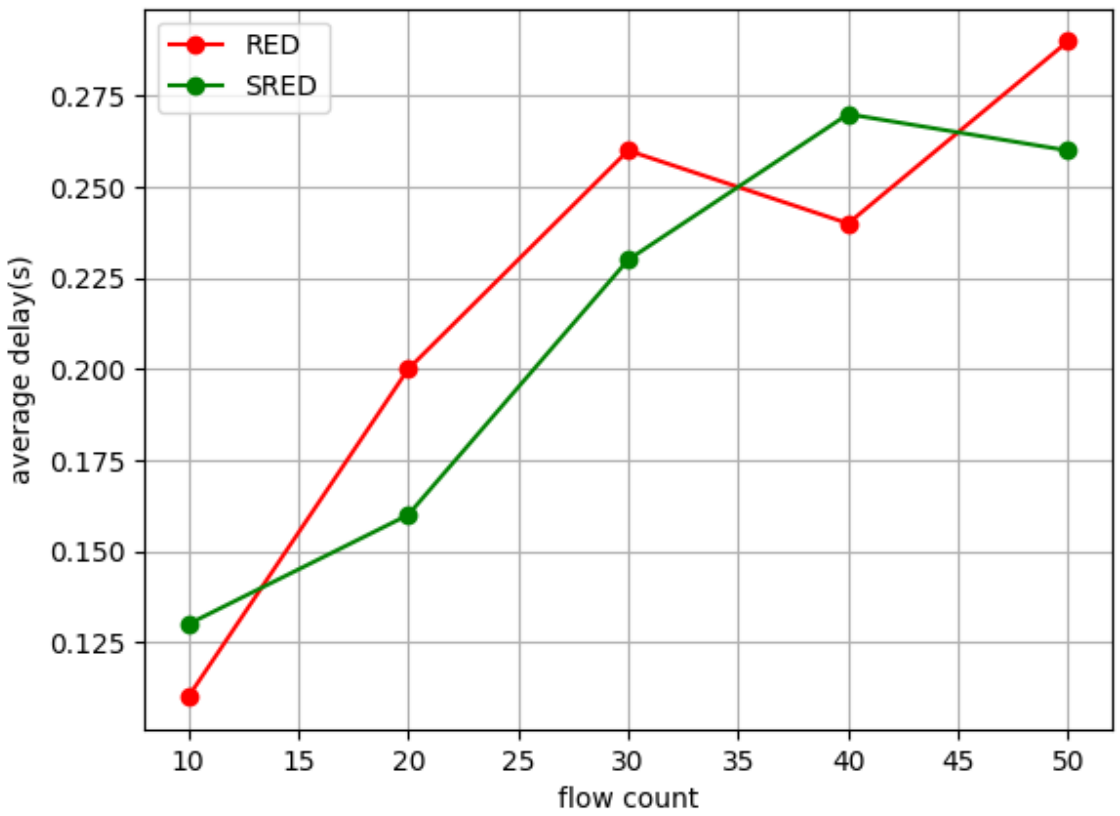

RED
SRED
95.5
95.4
95.3
95.2
95.1
95.0
94.9
94.8
drop ratio(%)
20
30
40
50
60
70
80
90
100
node count

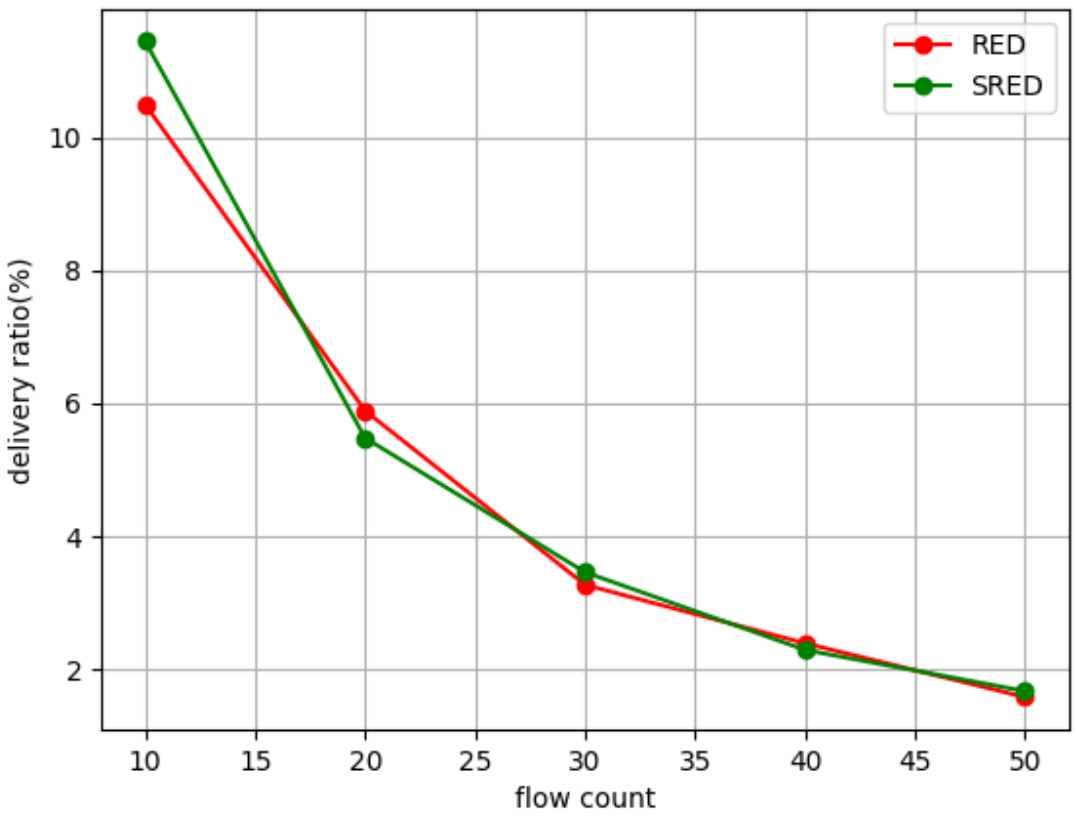

RED
SRED
diff in drop ratio per node(%)
node count
12
14
16
18
20
22
24
20
30
40
50
60
70
80
90
100

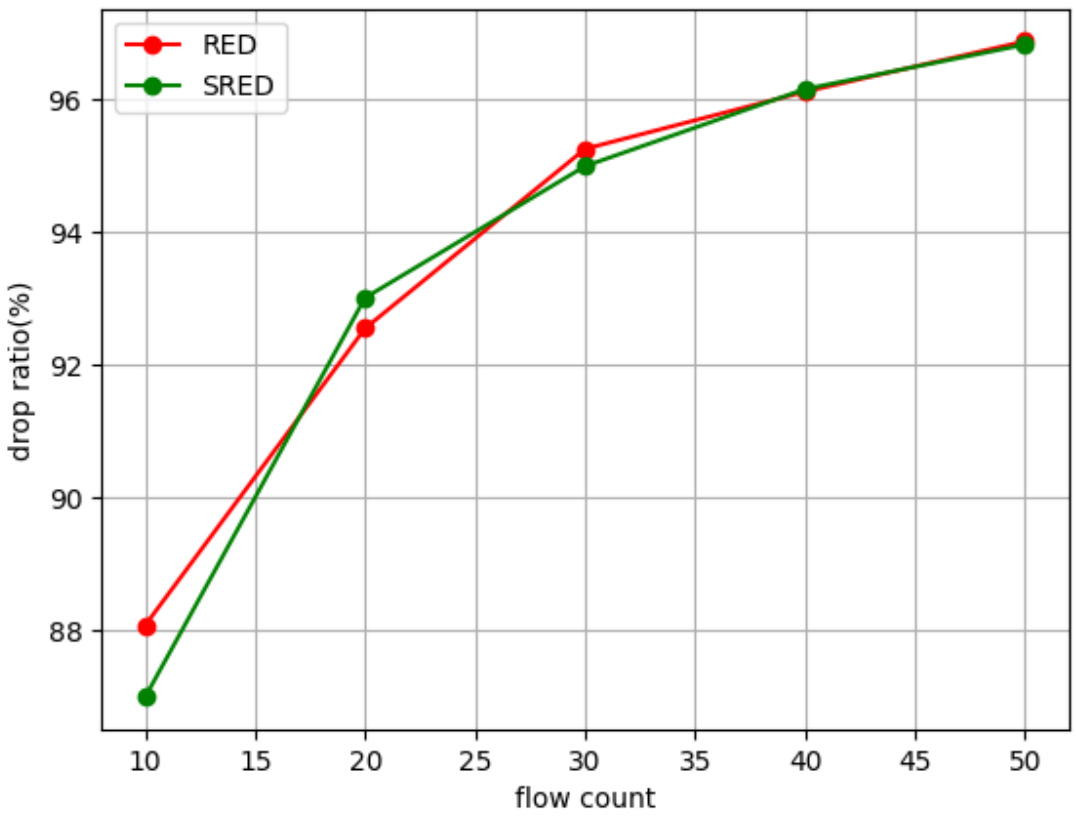

RED
SRED
20.5
20.0
19.5
19.0
total energy consumption(KJ)
20
30
40
50
60
70
80
90
100
node count

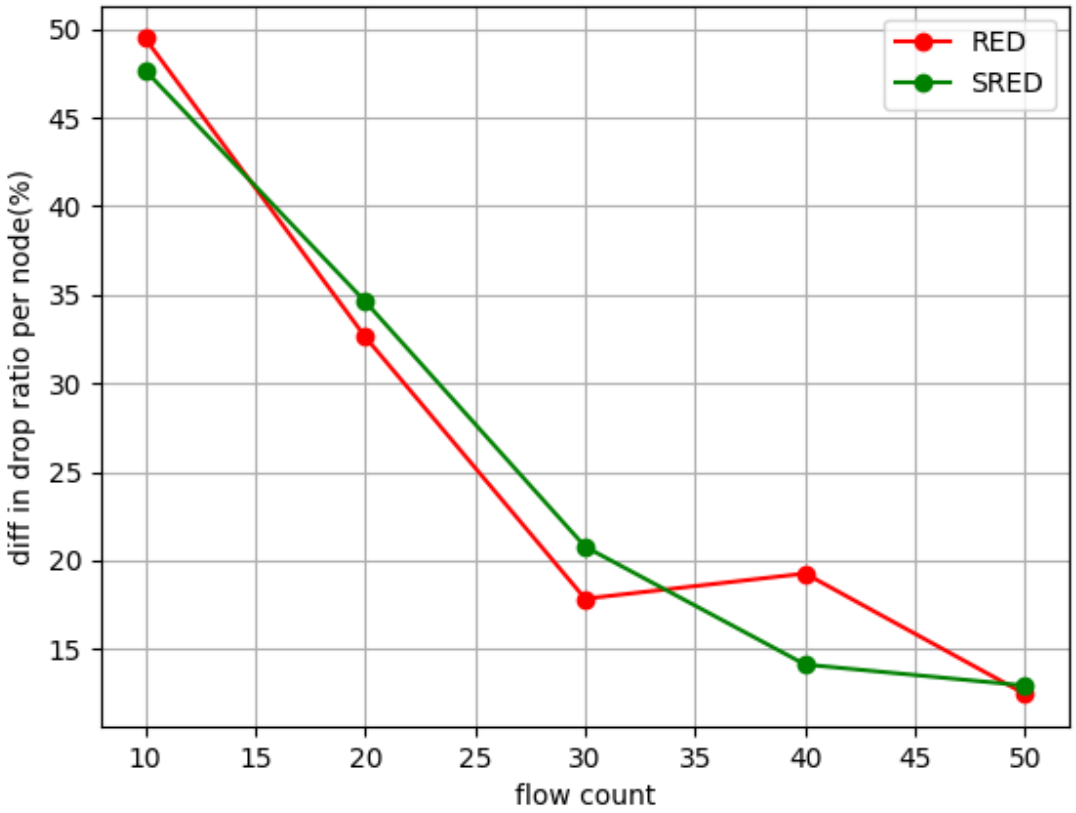
Vary Flow

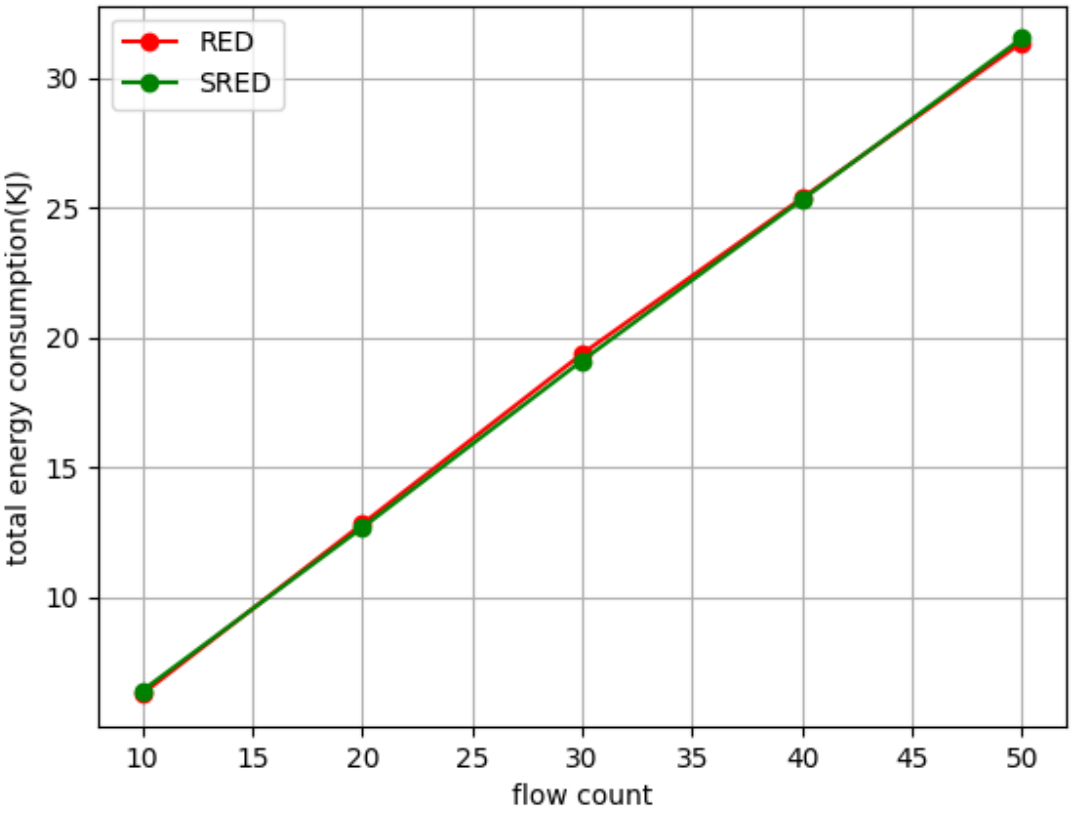

RED
SRED
throughput(kbps)
170
160
150
140
130
120
10
15
20
25
30
35
40
45
50
flow count


RED
SRED
0.275
0.250
0.225
0.200
0.175
0.150
0.125
average delay(s)
10
15
20
25
30
35
40
45
50
flow count

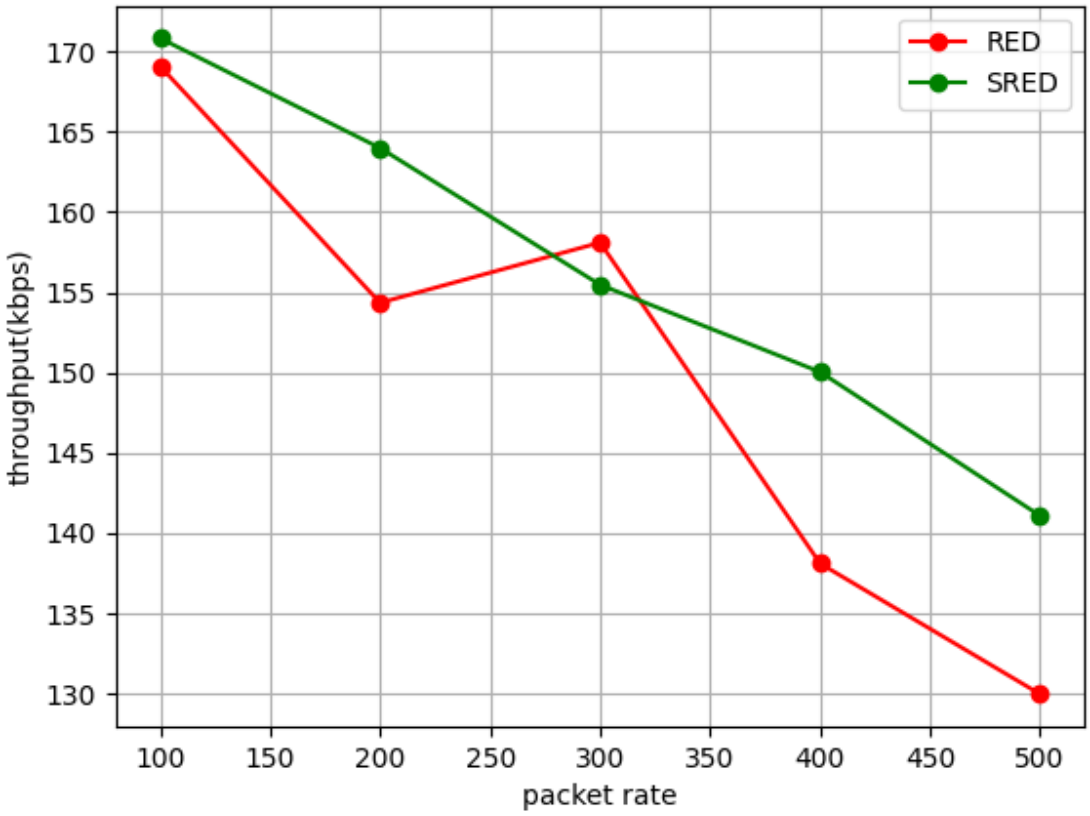

RED
SRED
delivery ratio(%)
10
8
6
4
2
10
15
20
25
30
35
40
45
50
flow count

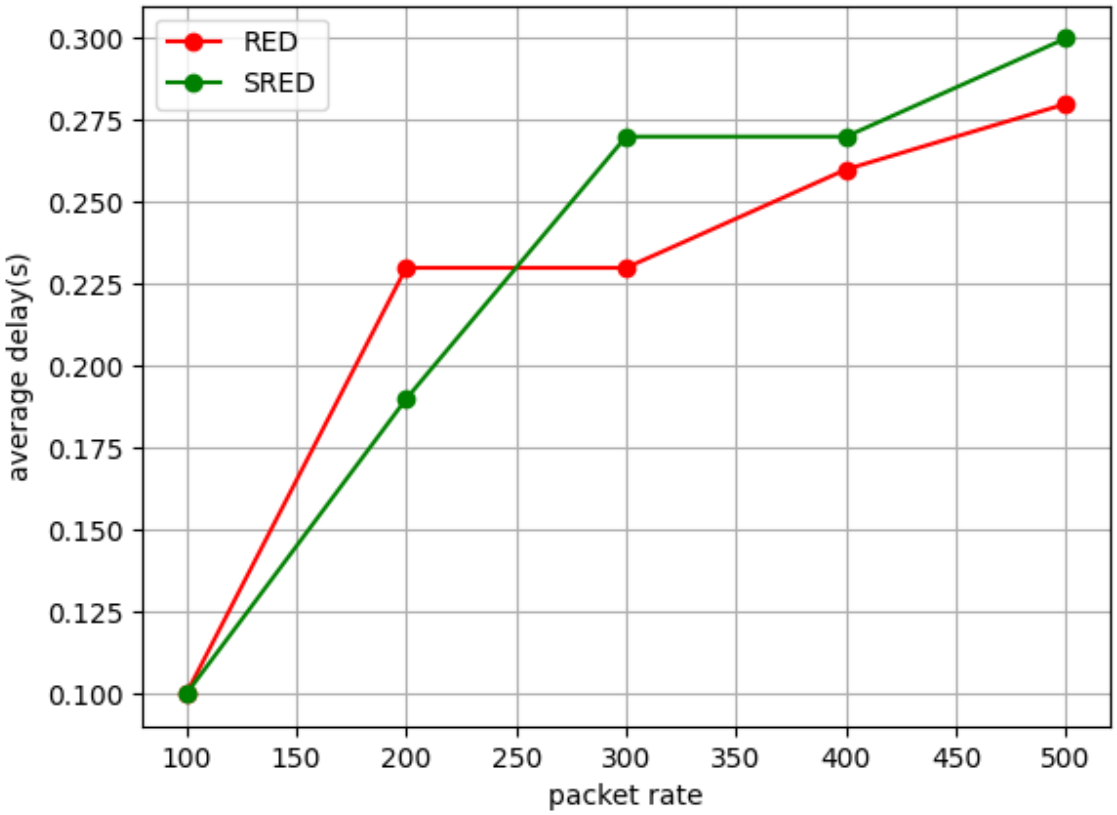

RED
SRED
drop ratio(%)
96
94
92
90
88
10
15
20
25
30
35
40
45
50
flow count

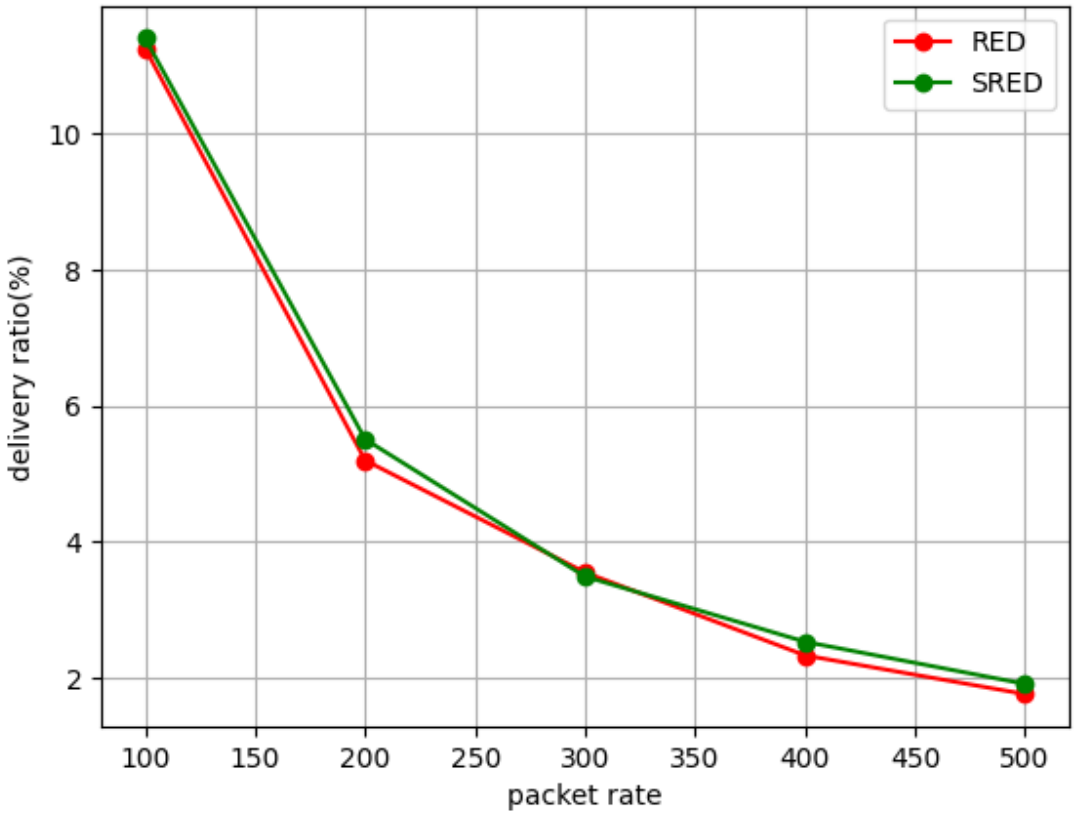

RED
SRED
diff in drop ratio per node(%)
flow count
50
45
40
35
30
25
20
15
10
15
20
25
30
35
40
45
50

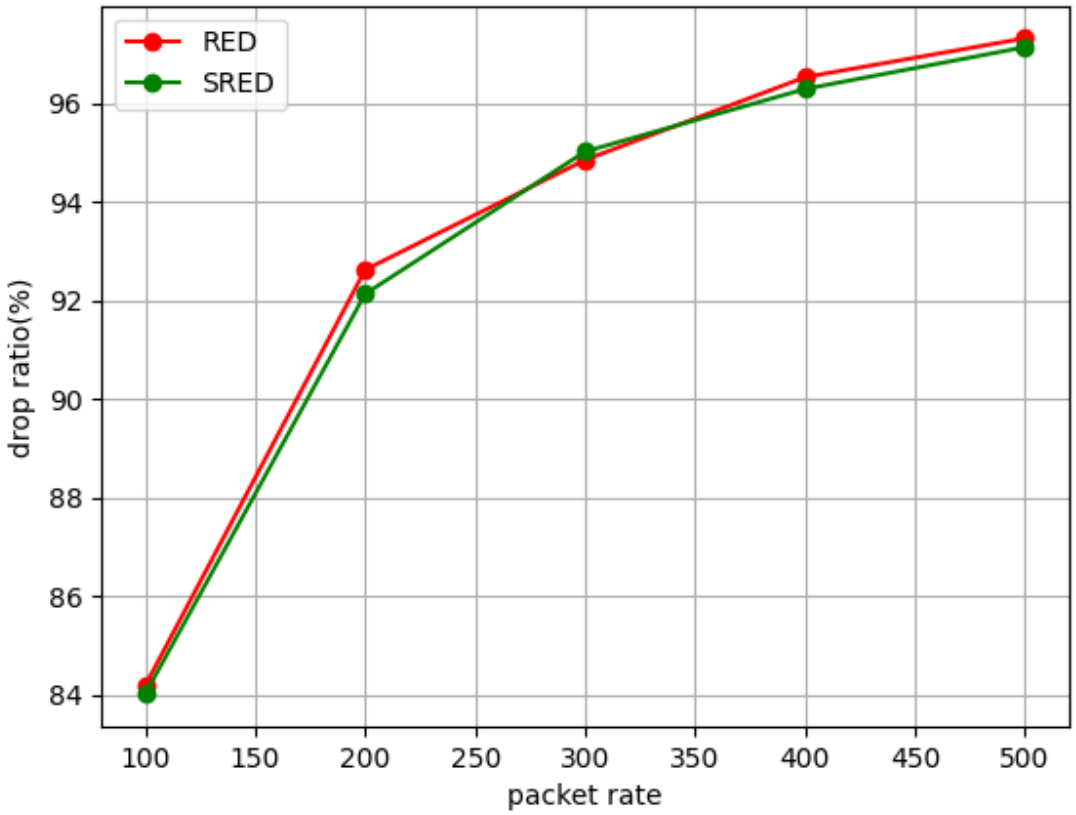

RED
SRED
total energy consumption(KJ)
flow count
10
15
20
25
30
35
40
45
50

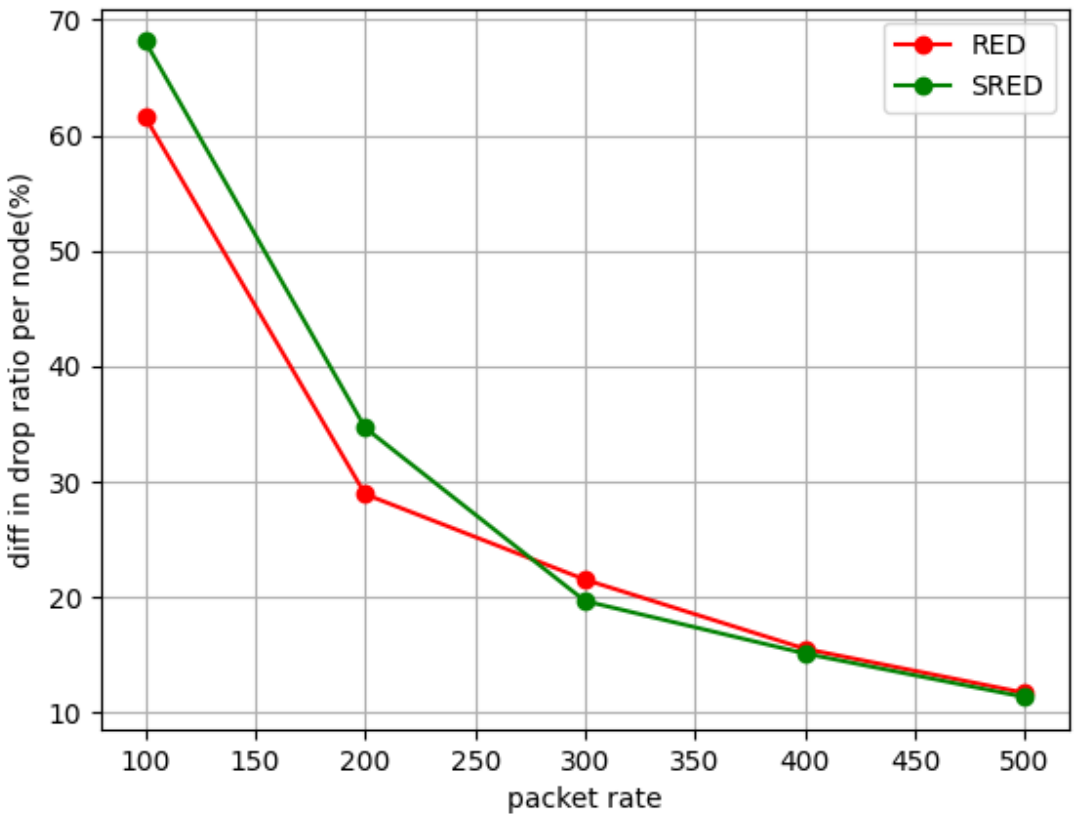
Vary Packet Rate

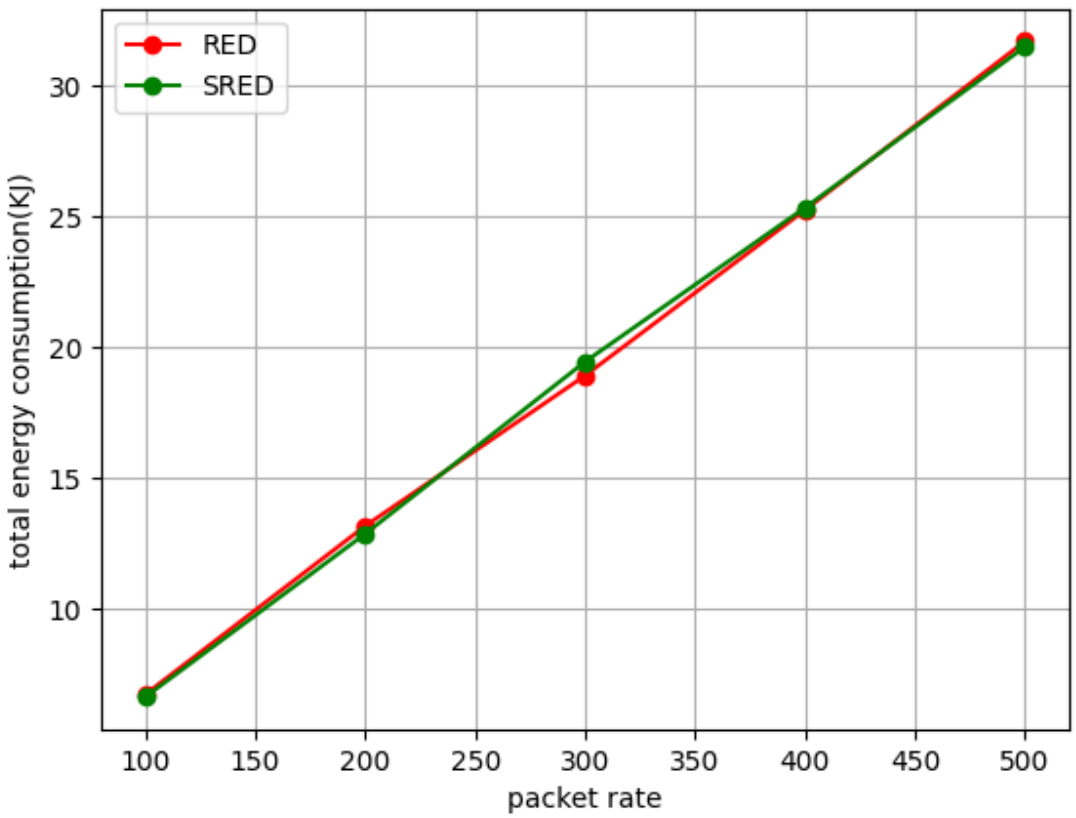



RED
SRED
0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.125
0.100
average delay(s)
100
150
200
250
300
350
400
450
500
packet rate

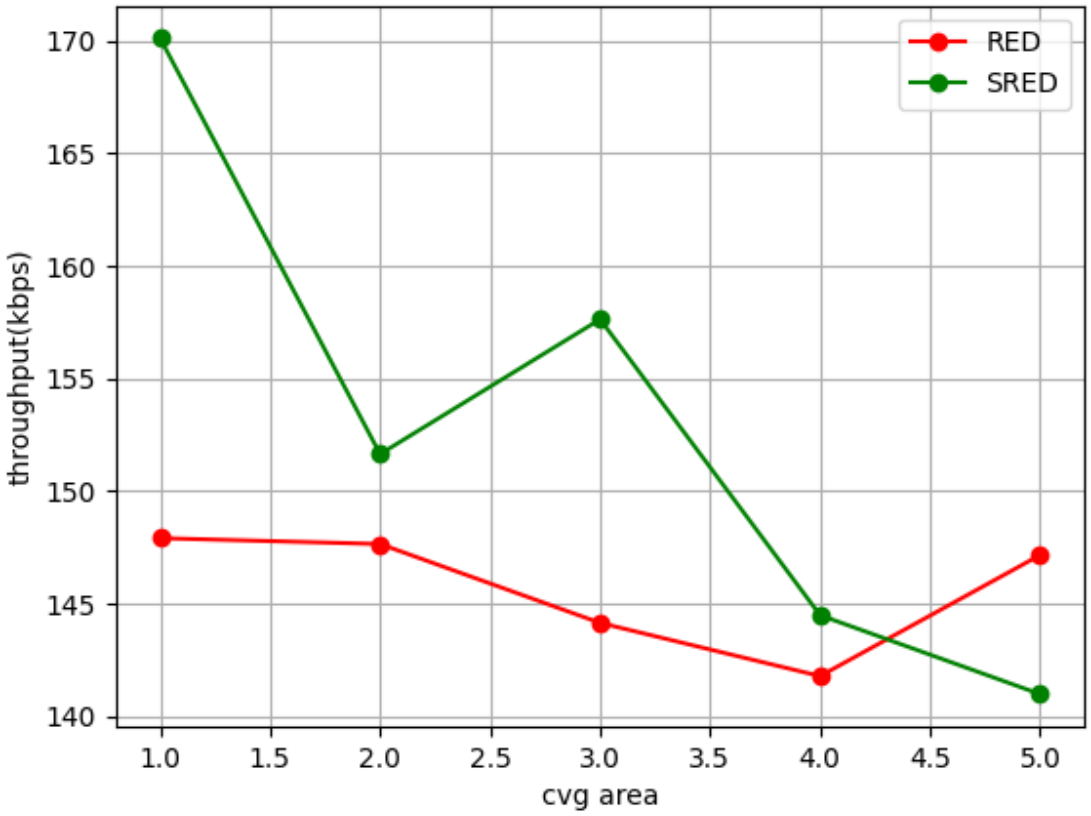

RED
SRED
delivery ratio(%)
10
8
6
4
2
100
150
200
250
300
350
400
450
500
packet rate

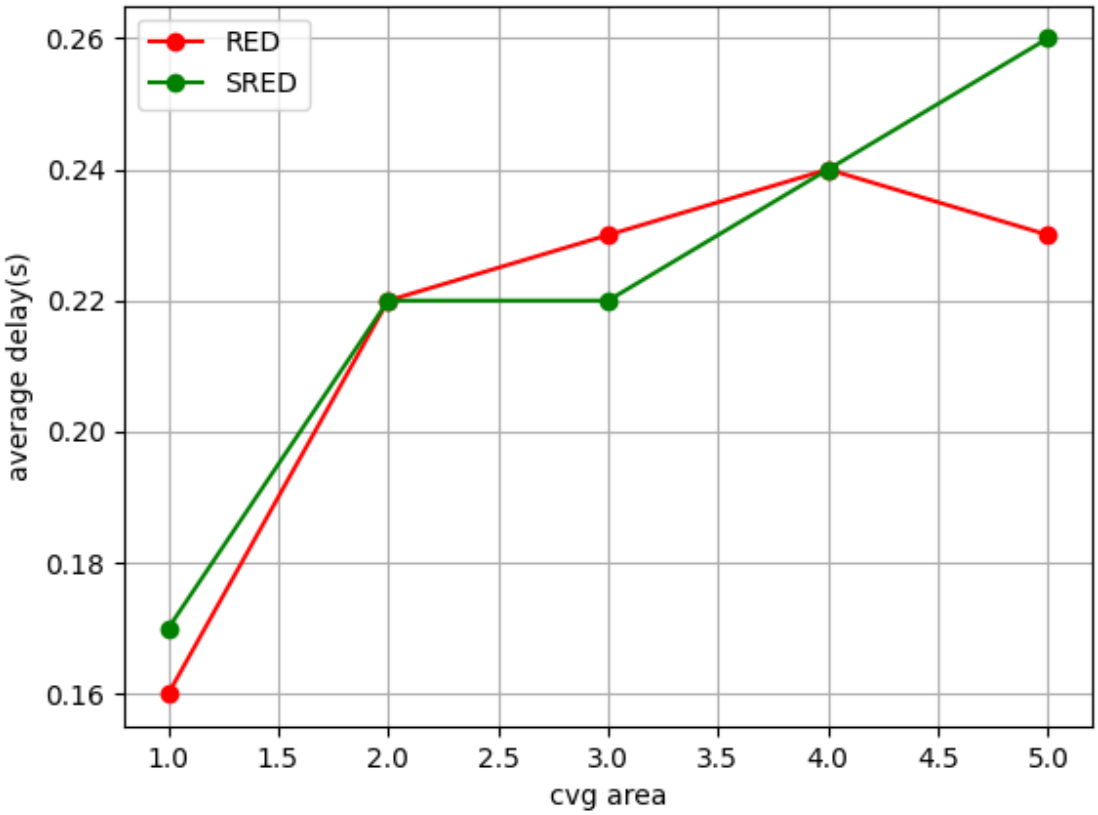

RED
SRED
drop ratio(%)
96
94
92
90
88
86
84
100
150
200
250
300
350
400
450
500
packet rate

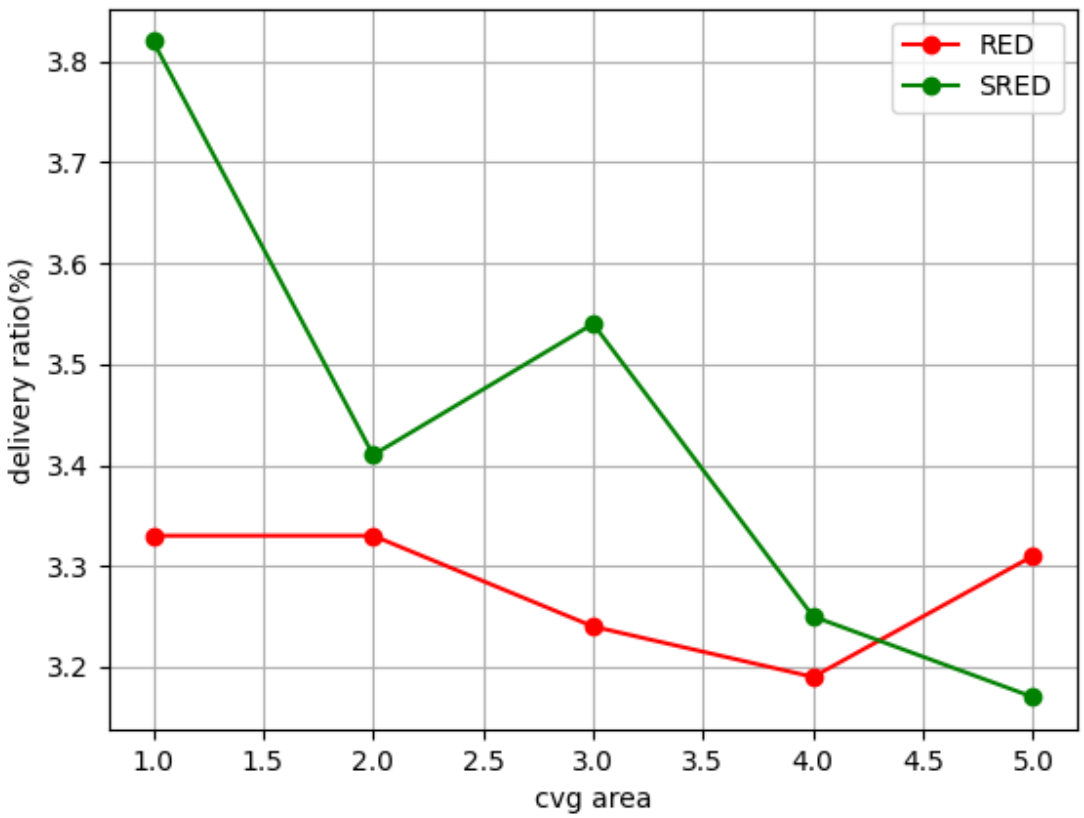

RED
SRED
diff in drop ratio per node(%)
packet rate
70
60
50
40
30
20
10
100
150
200
250
300
350
400
450
500

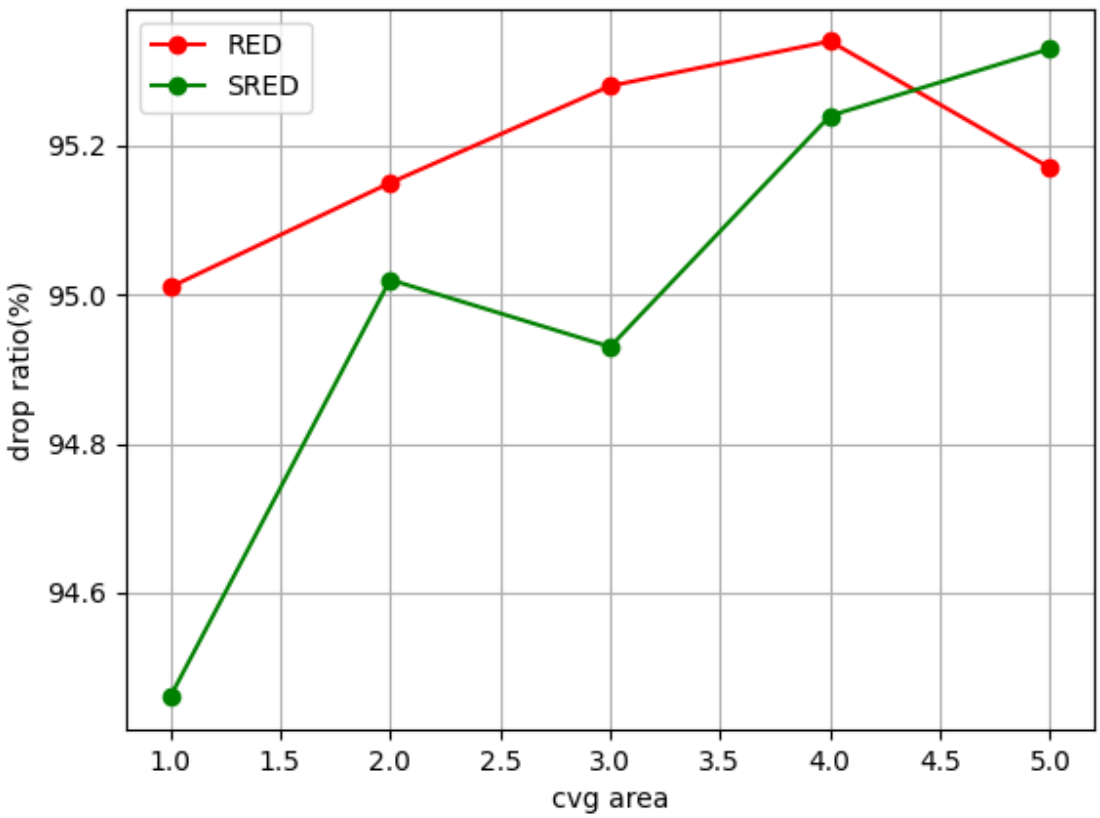

RED
SRED
total energy consumption(KJ)
30
25
20
15
10
100
150
200
250
300
350
400
450
500
packet rate

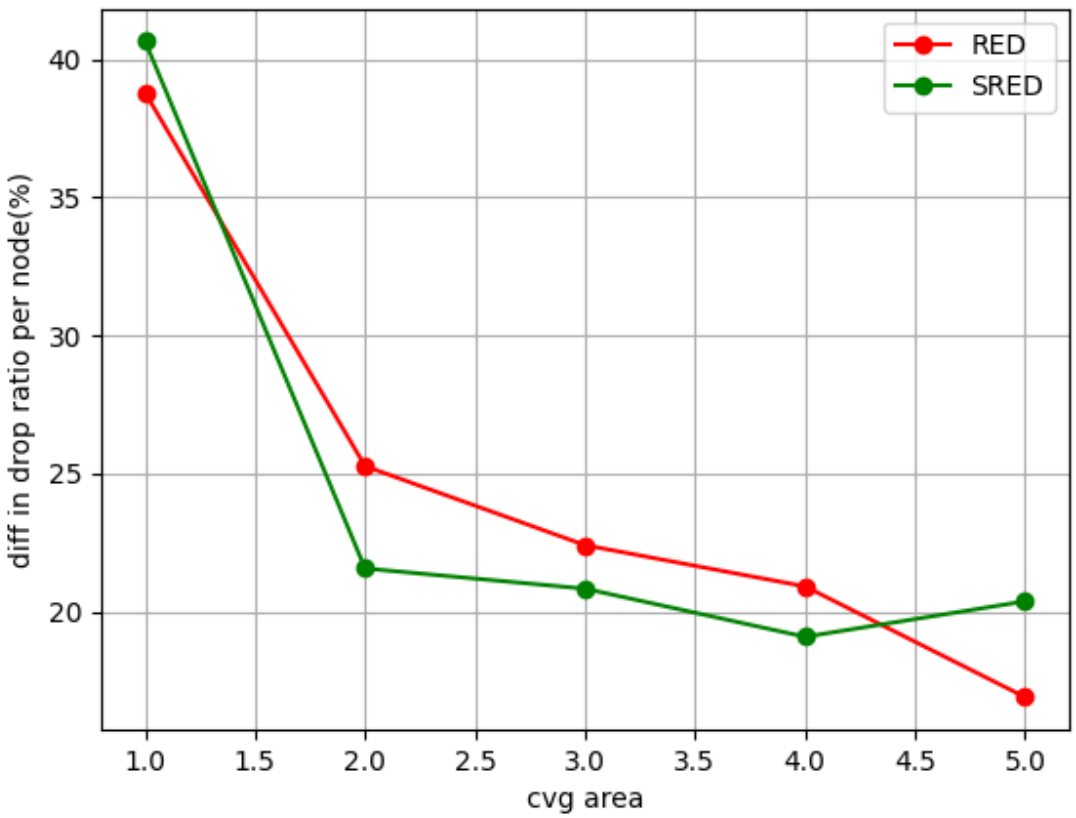
Vary Coverage Area

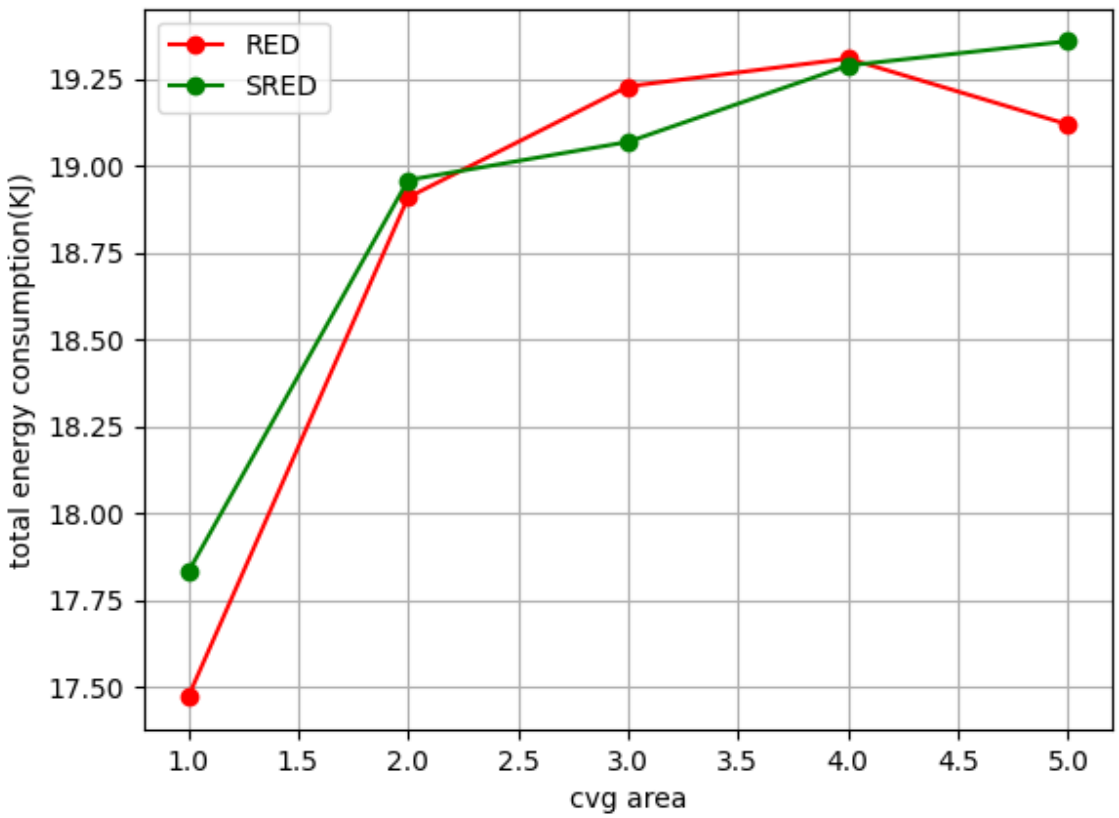



RED
SRED
0.26
0.24
0.22
0.20
0.18
0.16
average delay(s)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
cvg area


RED
SRED
delivery ratio(%)
3.8
3.7
3.6
3.5
3.4
3.3
3.2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
cvg area


RED
SRED
95.2
95.0
94.8
94.6
drop ratio(%)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
cvg area


RED
SRED
diff in drop ratio per node(%)
40
35
30
25
20
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
cvg area

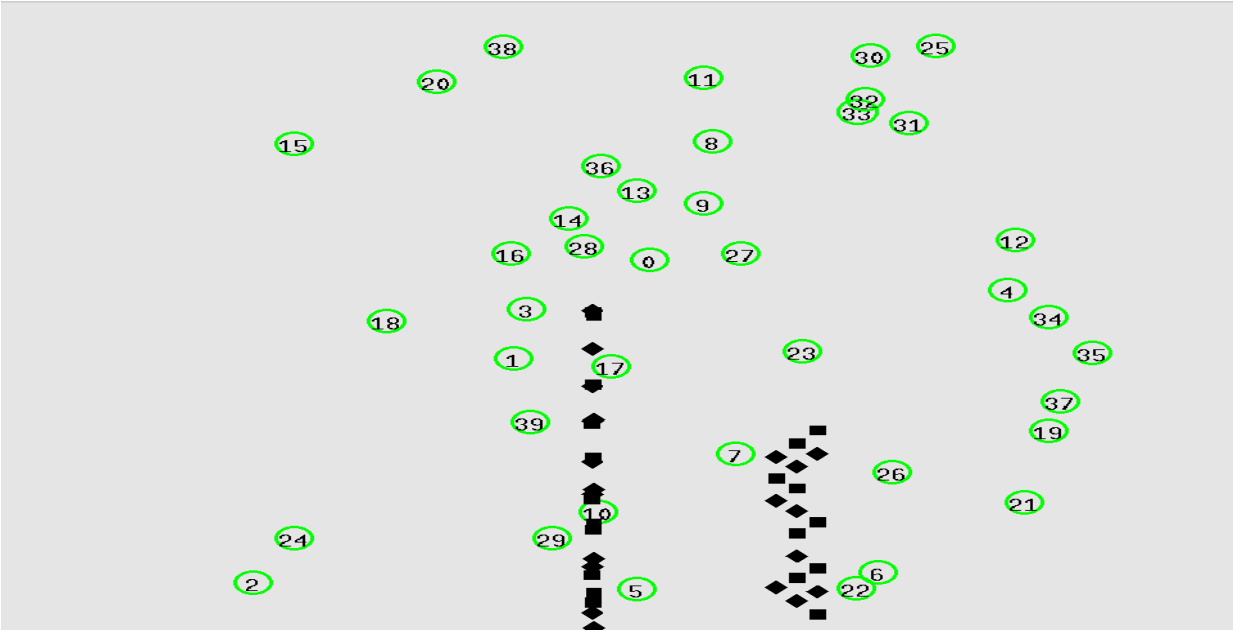

RED
SRED
19.25
19.00
18.75
18.50
18.25
18.00
17.75
17.50
total energy consumption(KJ)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
cvg area

TASK - 2

Topology

For task-2, the given network is **Wireless 802.15.4 (mobile)**. For a given number of nodes and flows, for each flow we randomly select two nodes and add a flow in them. So, they don't follow any specific topology.

For the transfer layer protocol we used UDP Protocol and for the application layer we used CBR Traffic. And in the network layer we used the DSDV Routing Protocol.



Parameters Under Variation

Baseline Parameters are -

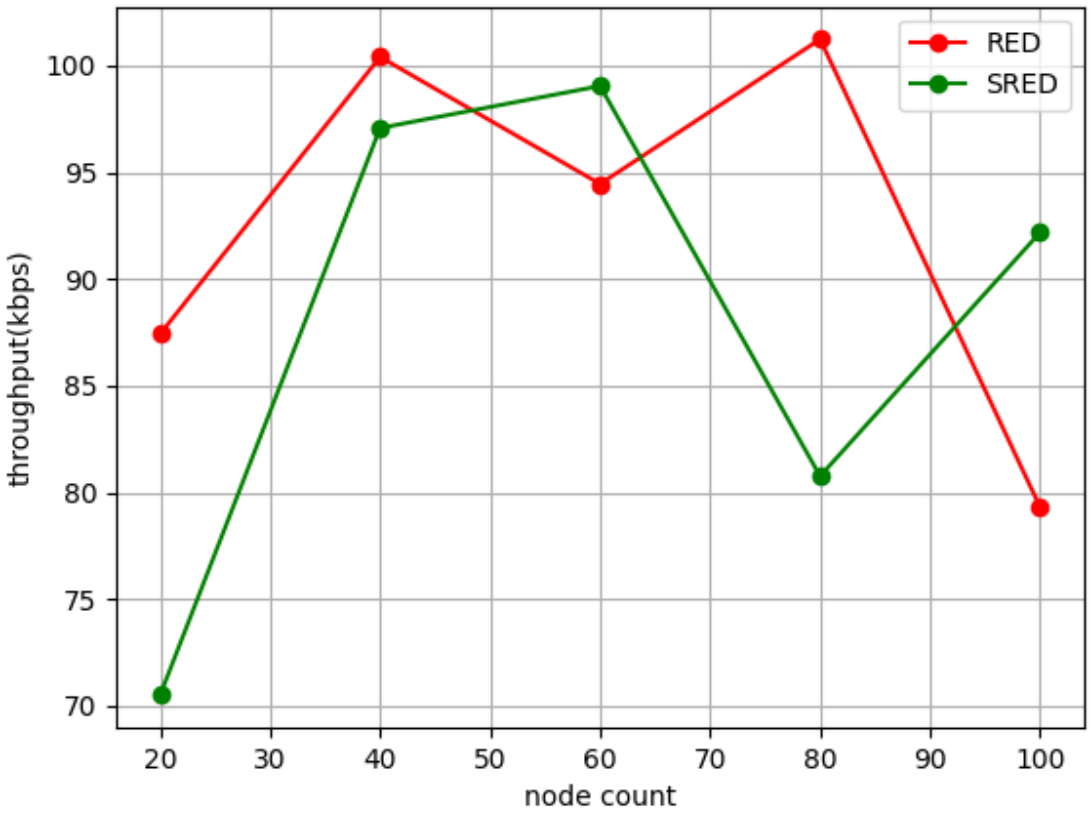
1. Number of Nodes : 40
2. Number of Flows : 30
3. Number of Packet per Second : 300
4. Node Speed (m/s) : 15

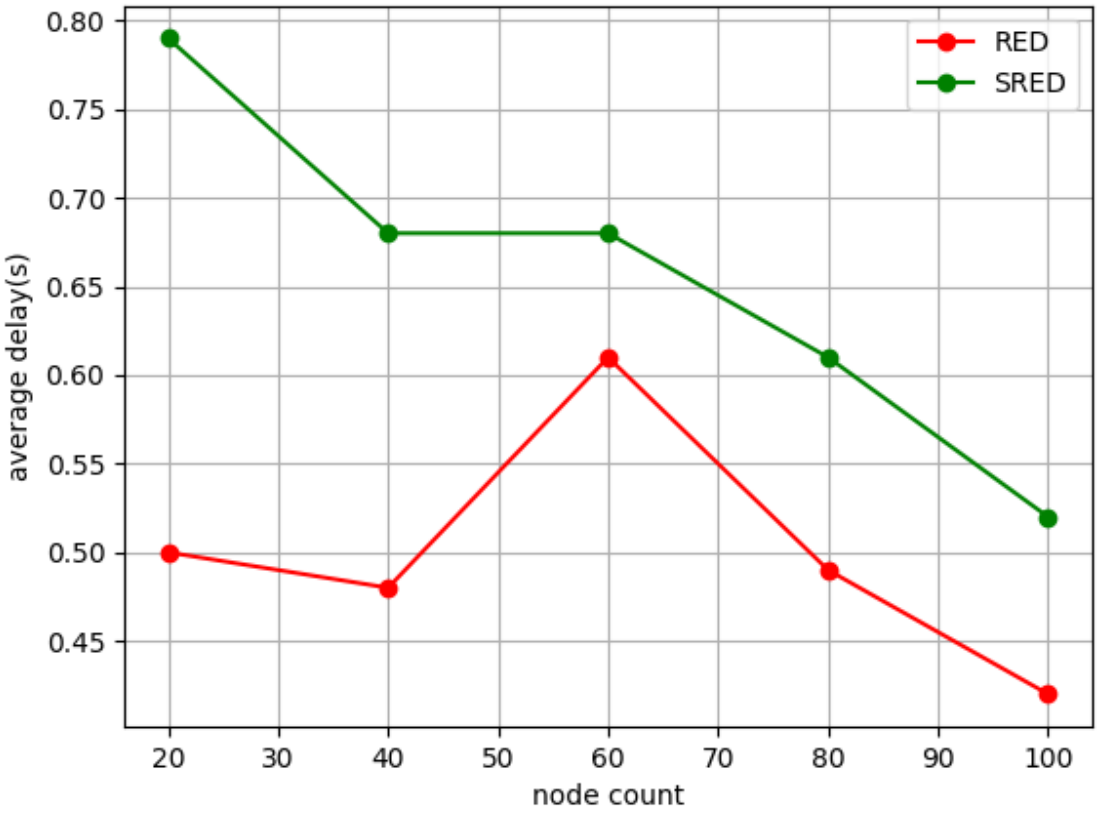
Varying Parameters -

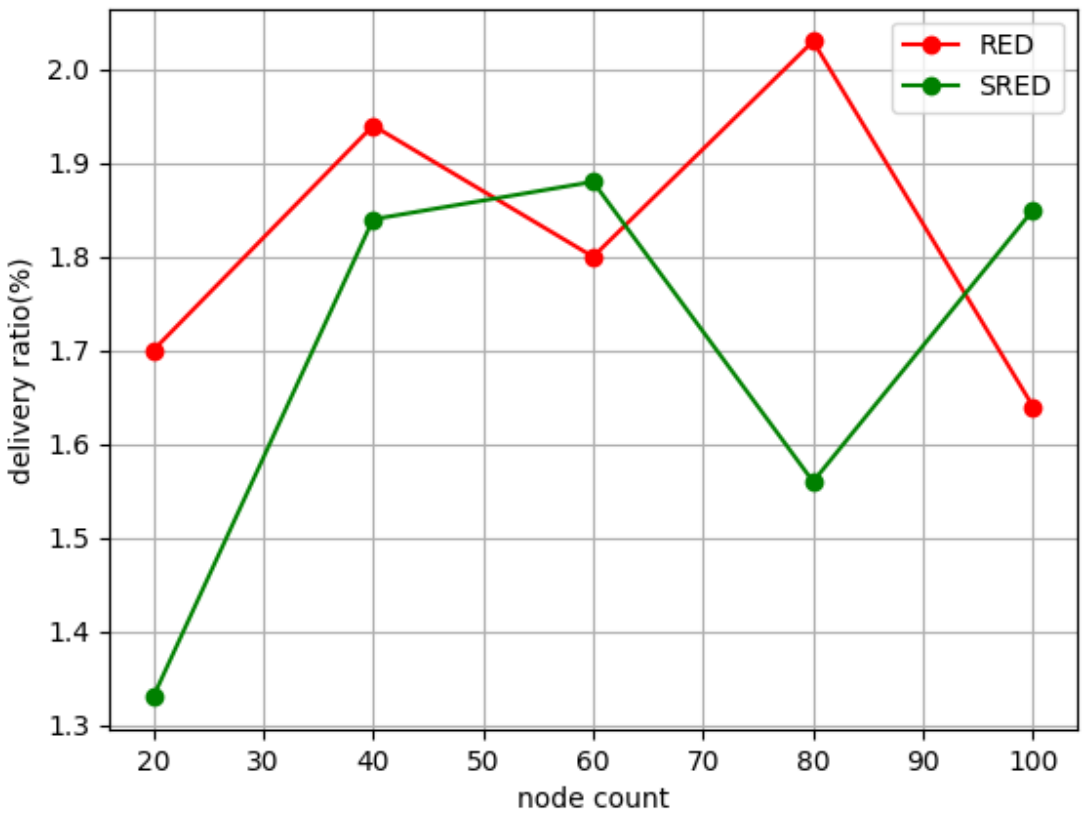
1. Number of Nodes : 20, 40, 60, 80, 100
2. Number of Flows : 10, 20, 30, 40, 50
3. Number of Packet per Second : 100, 200, 300, 400, 500
4. Node Speed (m/s) : 5, 10, 15, 20, 25

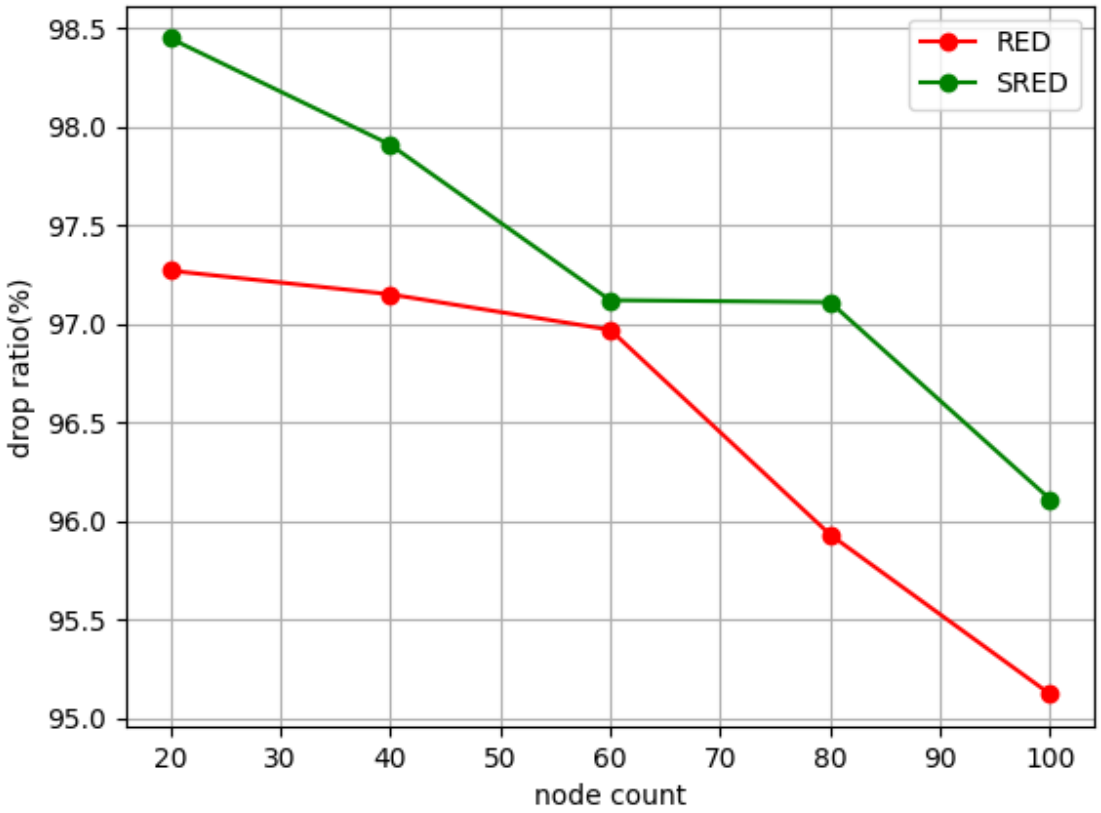
GRAPHS

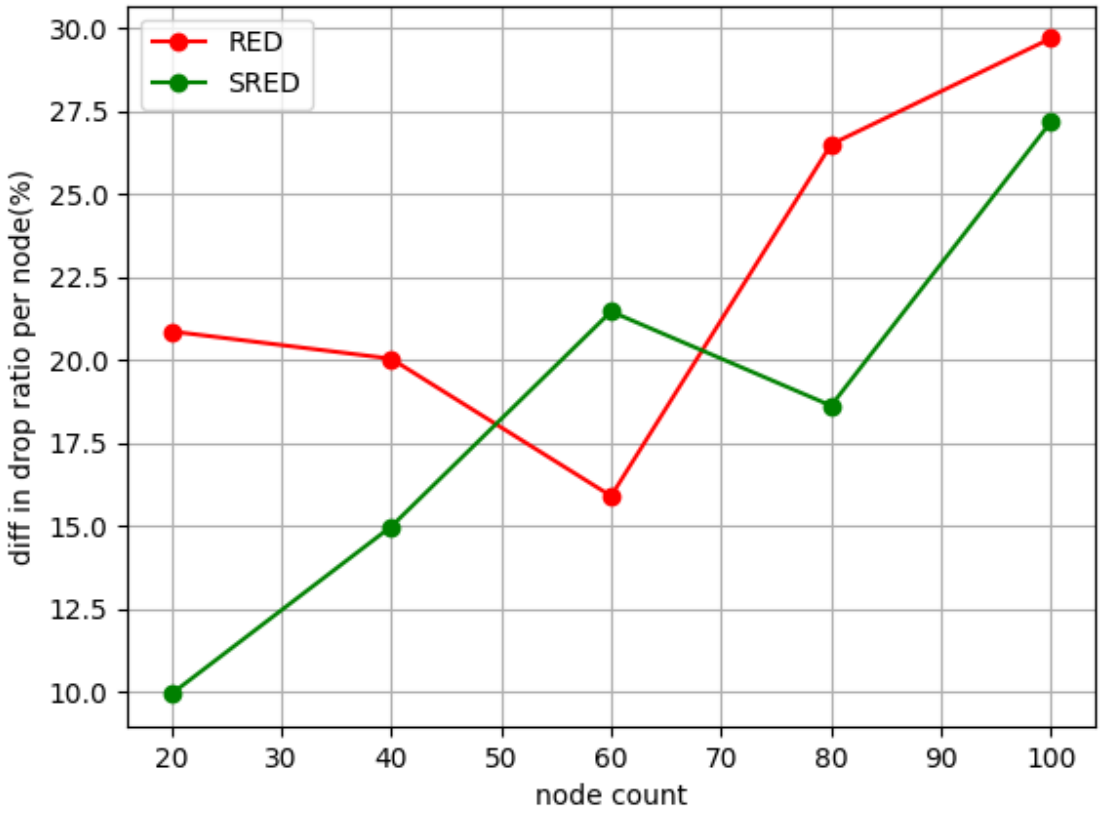
Vary Nodes

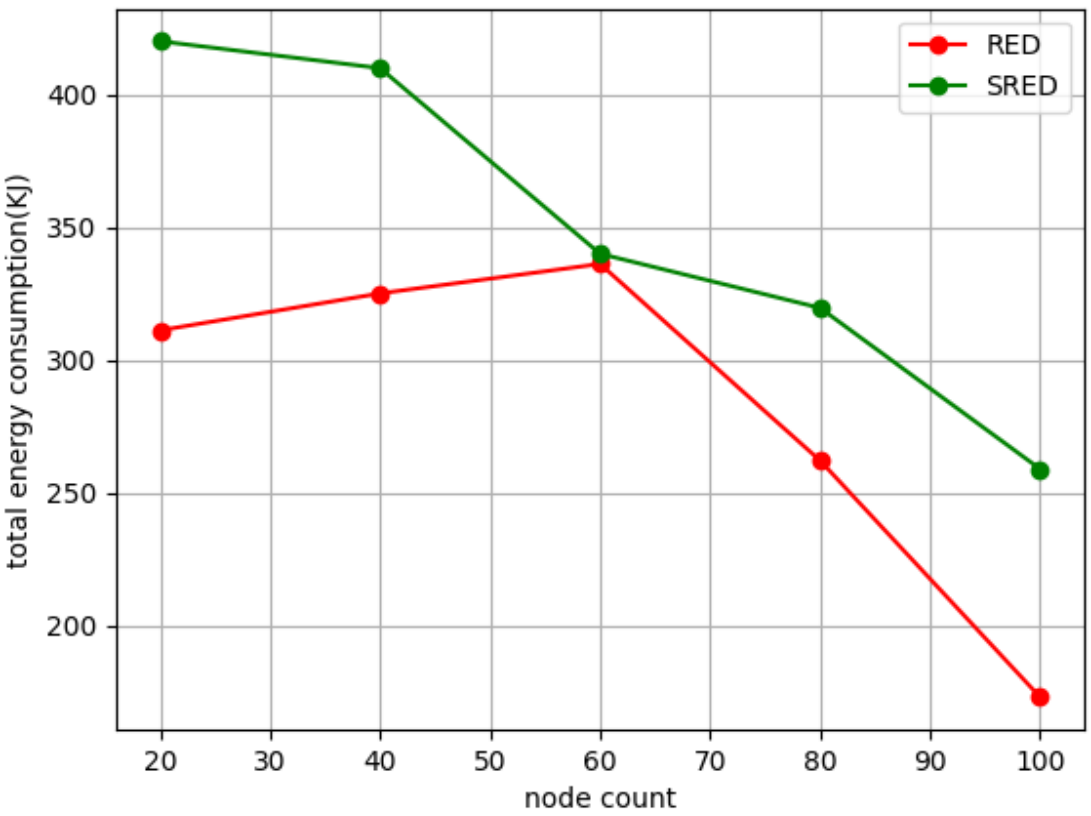



RED
SRED
average delay(s)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
20
30
40
50
60
70
80
90
100
node count

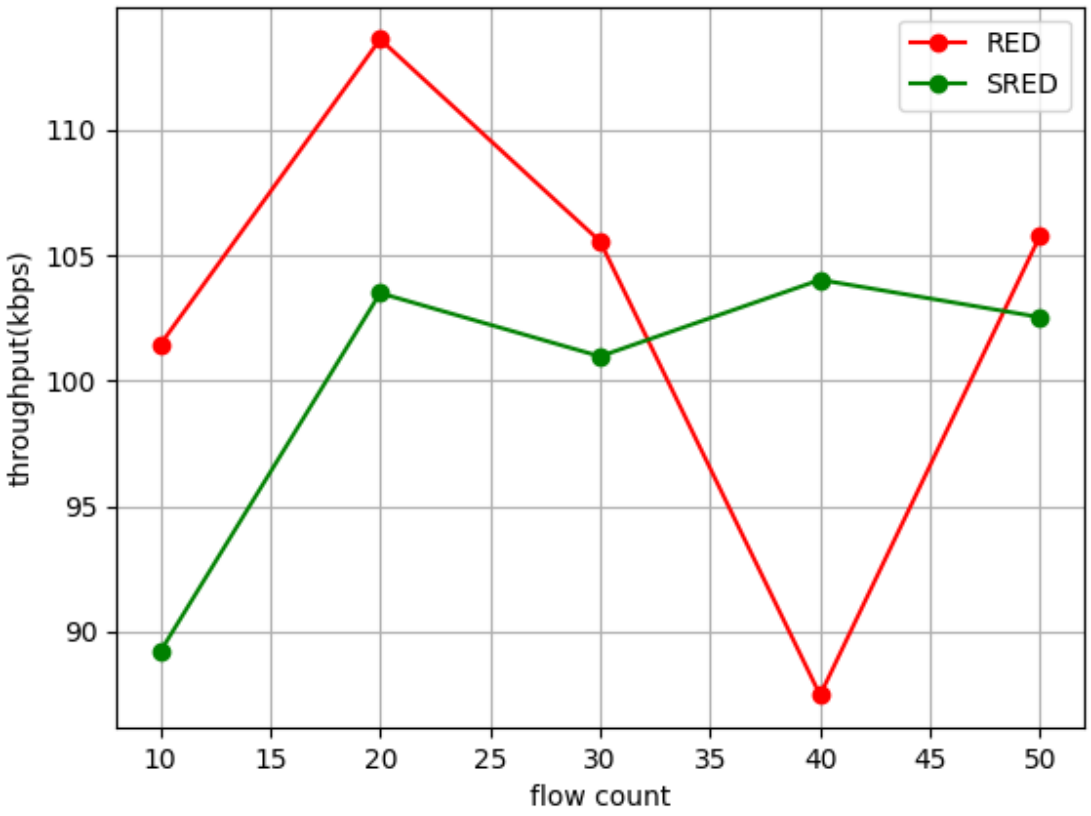

RED
SRED
delivery ratio(%)
2.0
1.9
1.8
1.7
1.6
1.5
1.4
1.3
20
30
40
50
60
70
80
90
100
node count

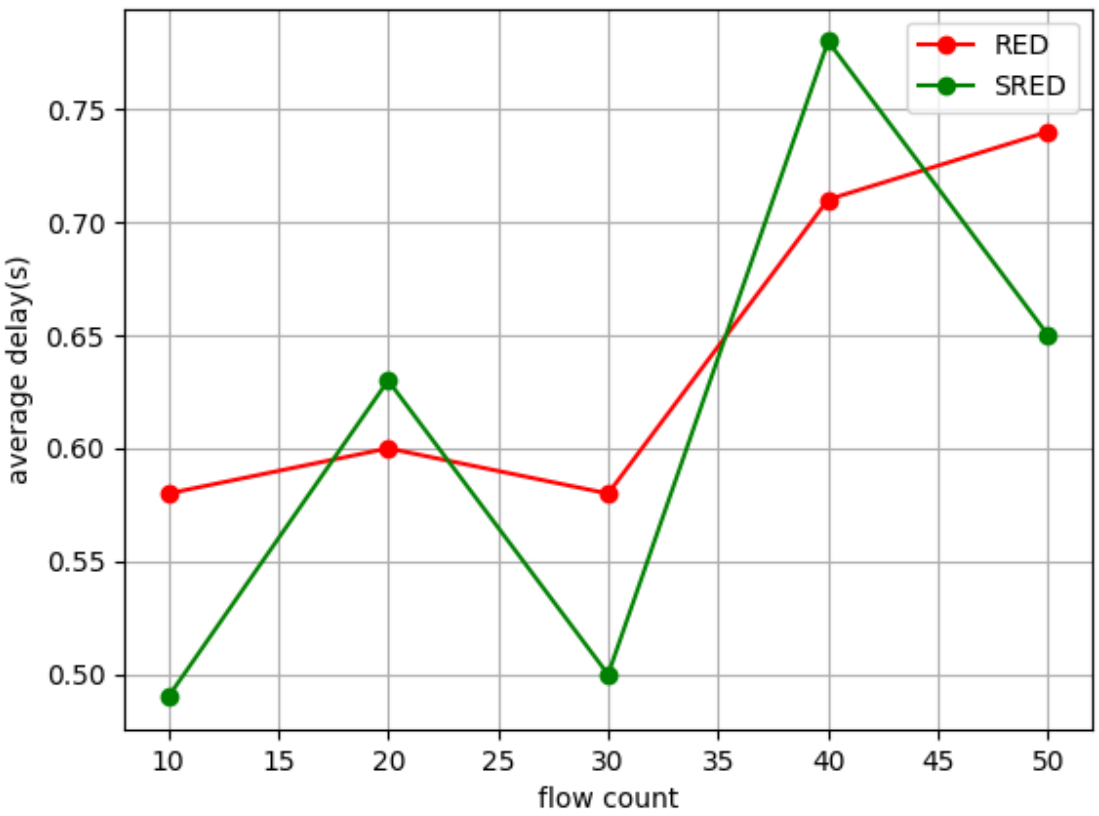

RED
SRED
98.5
98.0
97.5
97.0
96.5
96.0
95.5
95.0
drop ratio(%)
20
30
40
50
60
70
80
90
100
node count

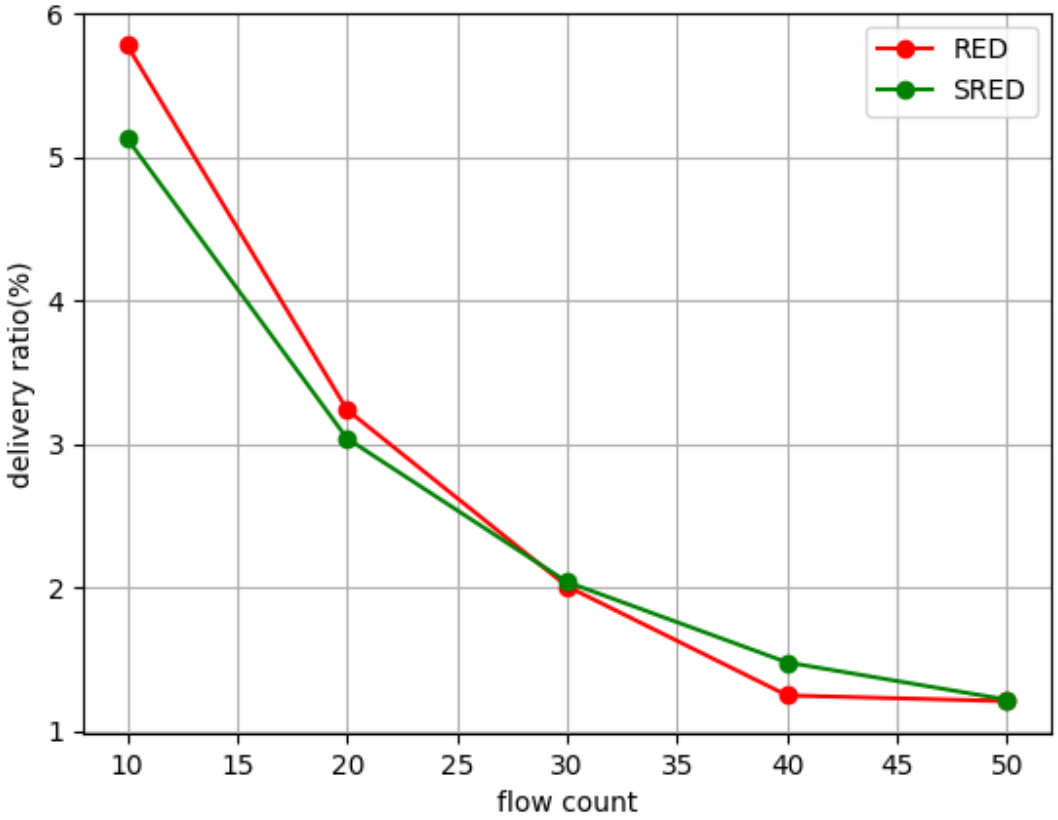

RED
SRED
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
diff in drop ratio per node(%)
20
30
40
50
60
70
80
90
100
node count

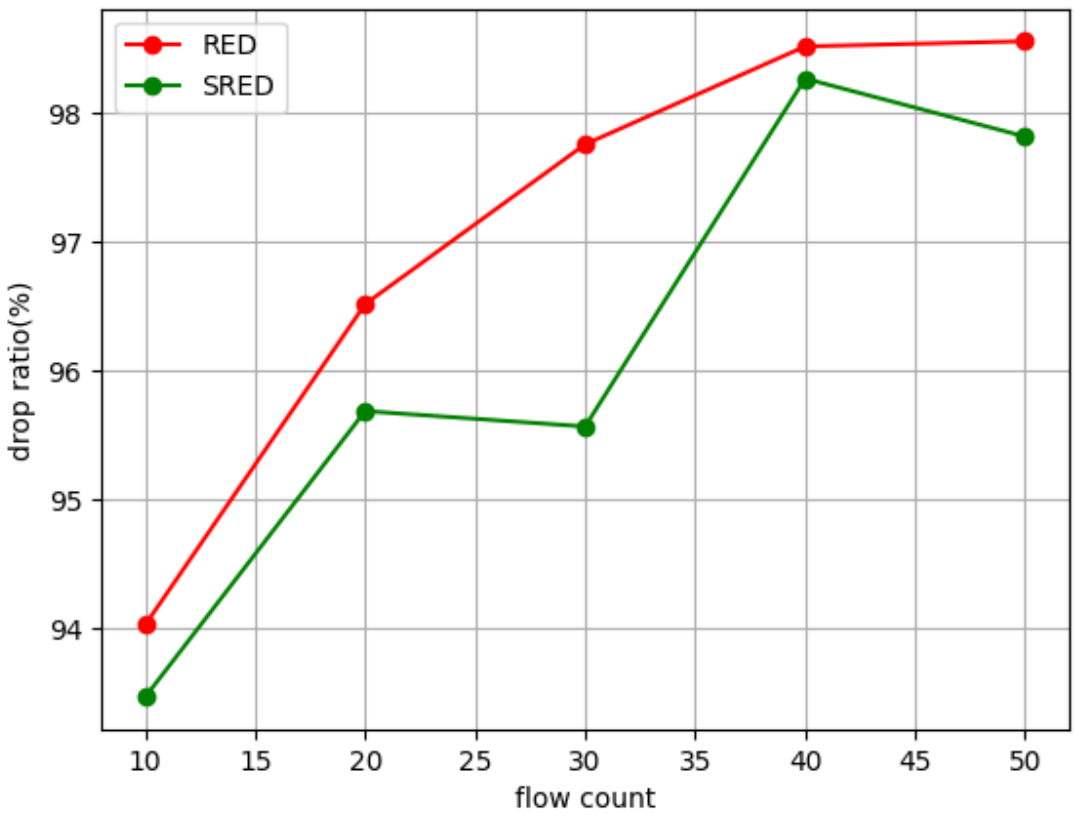

RED
SRED
total energy consumption(KJ)
400
350
300
250
200
20
30
40
50
60
70
80
90
100
node count

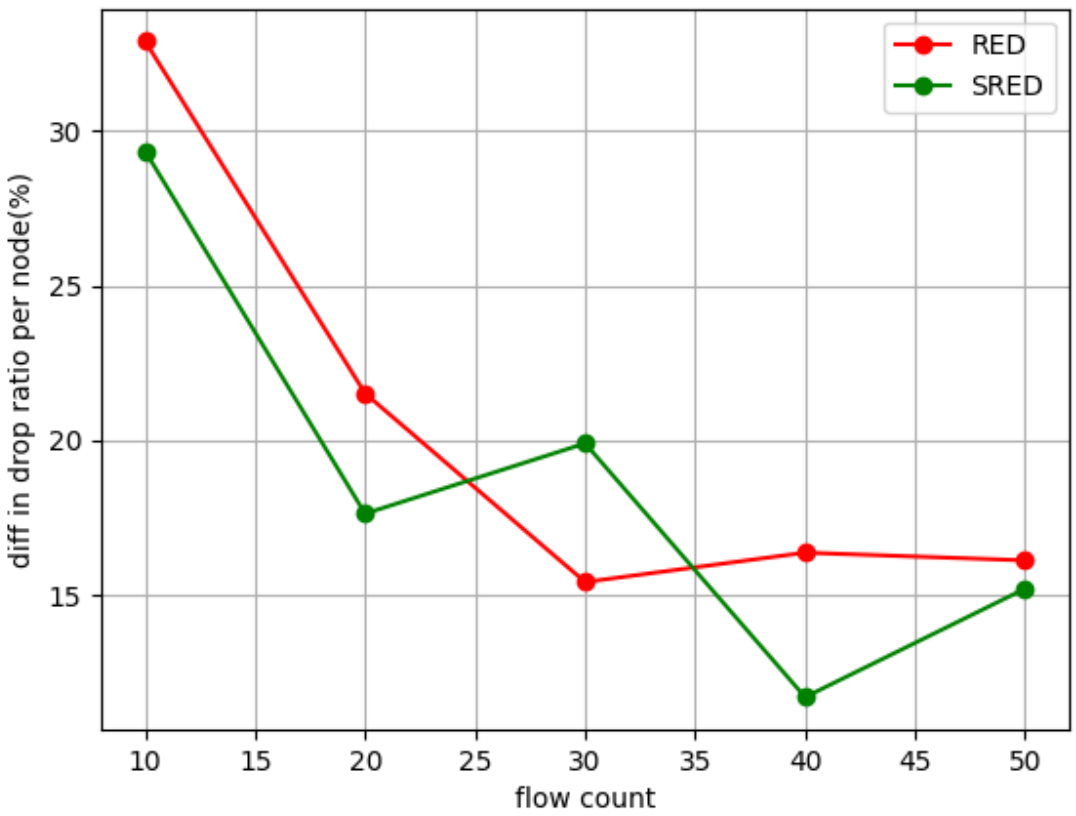
Vary Flow

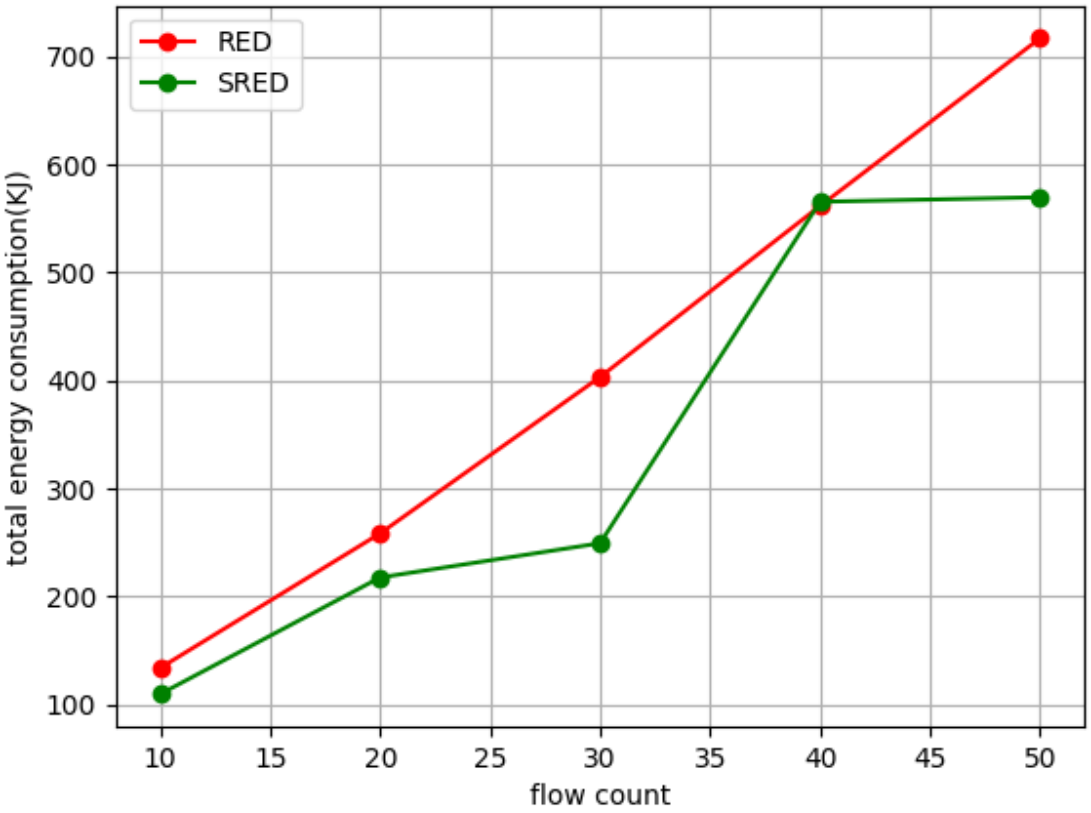

RED
SRED
throughput(kbps)
110
105
100
95
90
10
15
20
25
30
35
40
45
50
flow count


RED
SRED
average delay(s)
0.75
0.70
0.65
0.60
0.55
0.50
10
15
20
25
30
35
40
45
50
flow count

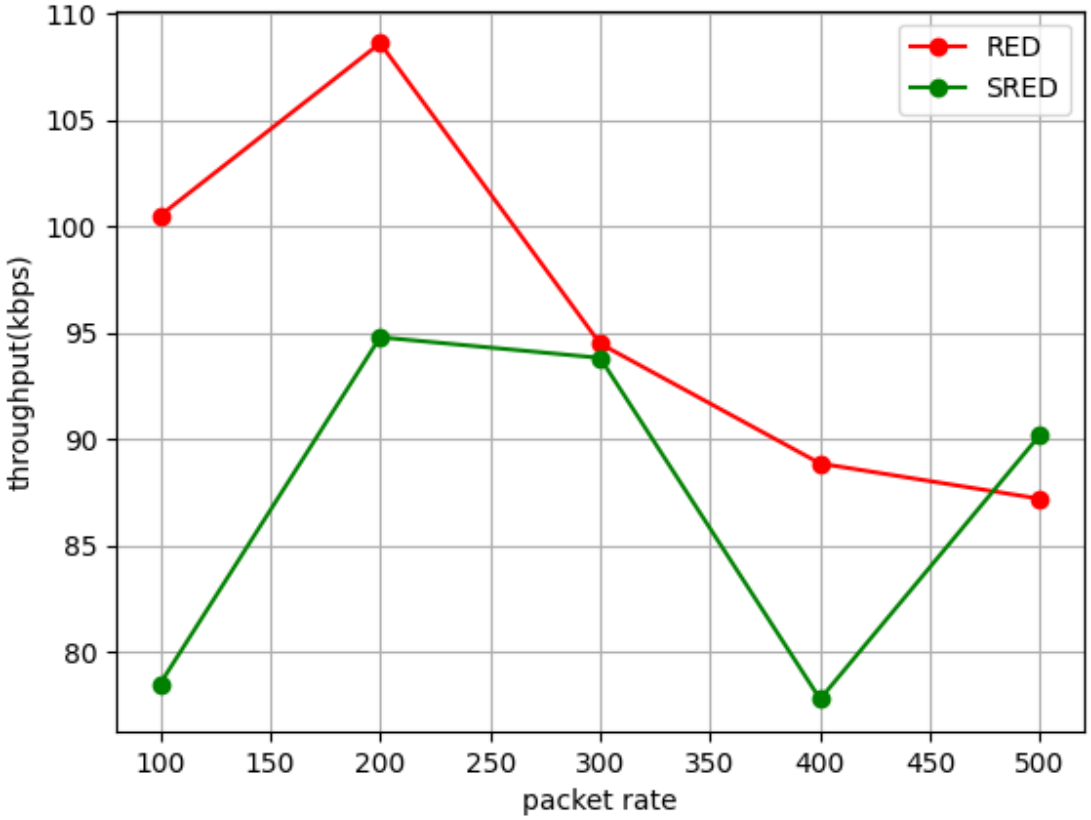

RED
SRED
delivery ratio(%)
flow count
6
5
4
3
2
1
10
15
20
25
30
35
40
45
50

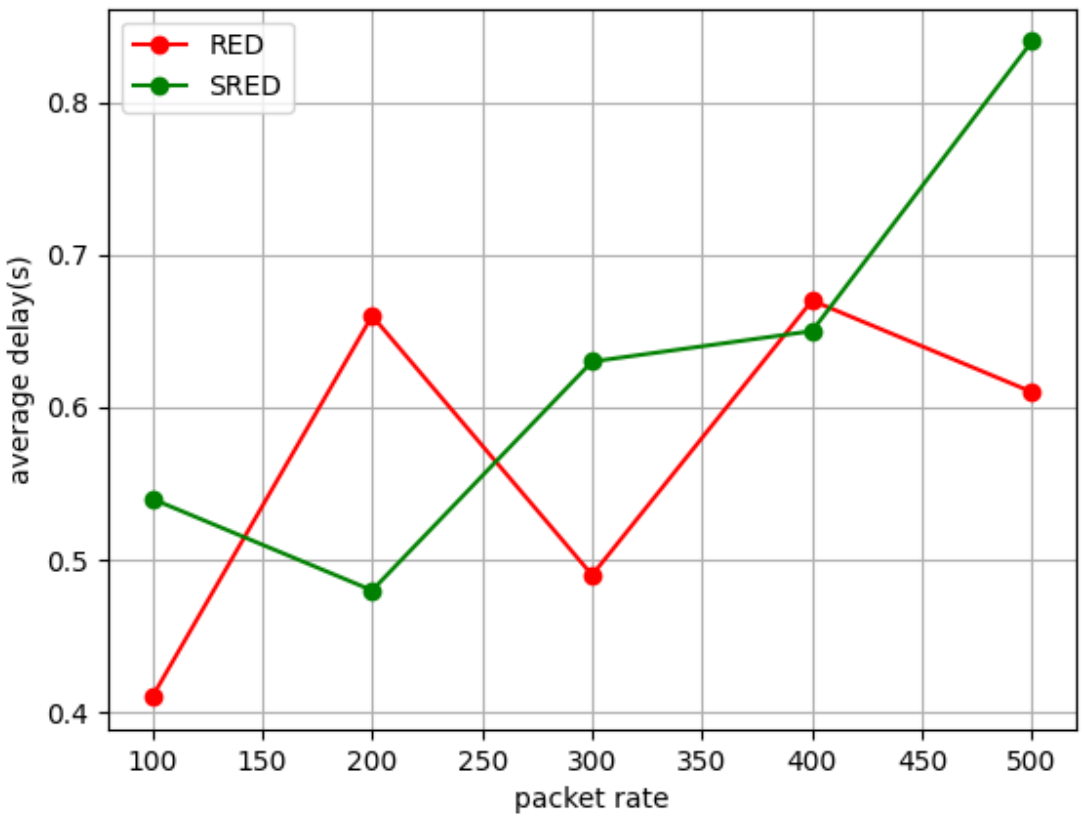

RED
SRED
drop ratio(%)
98
97
96
95
94
10
15
20
25
30
35
40
45
50
flow count

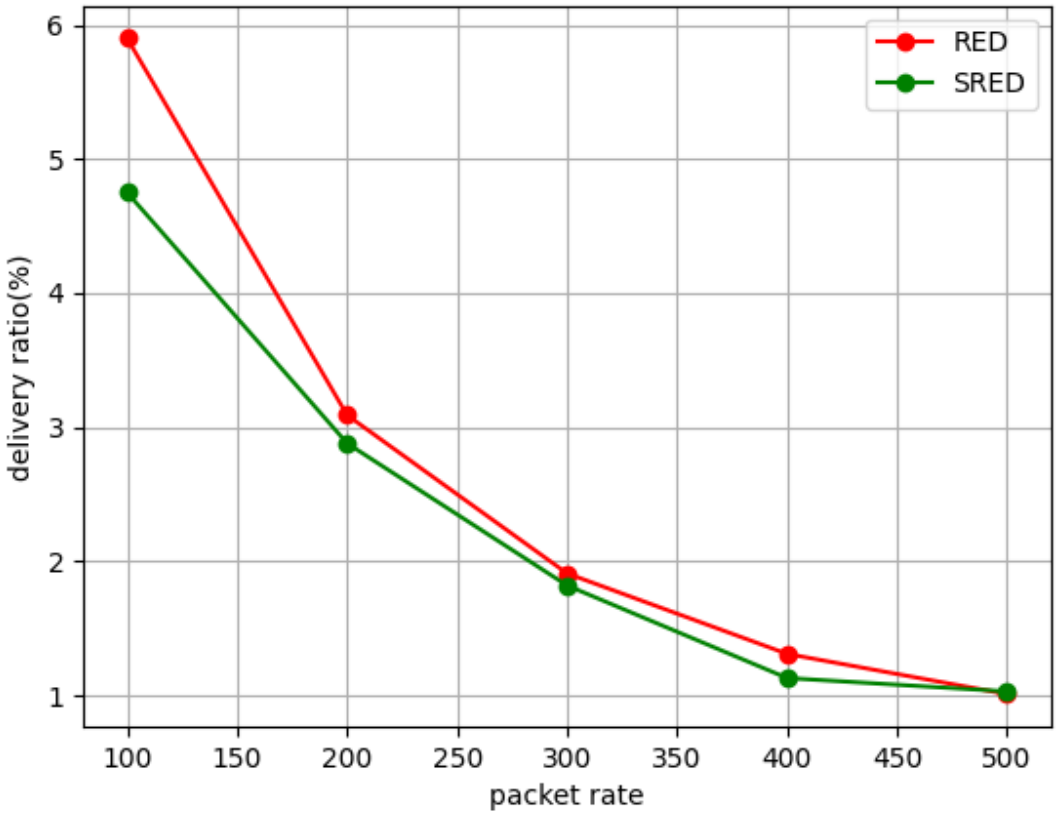

RED
SRED
diff in drop ratio per node(%)
30
25
20
15
10
15
20
25
30
35
40
45
50
flow count

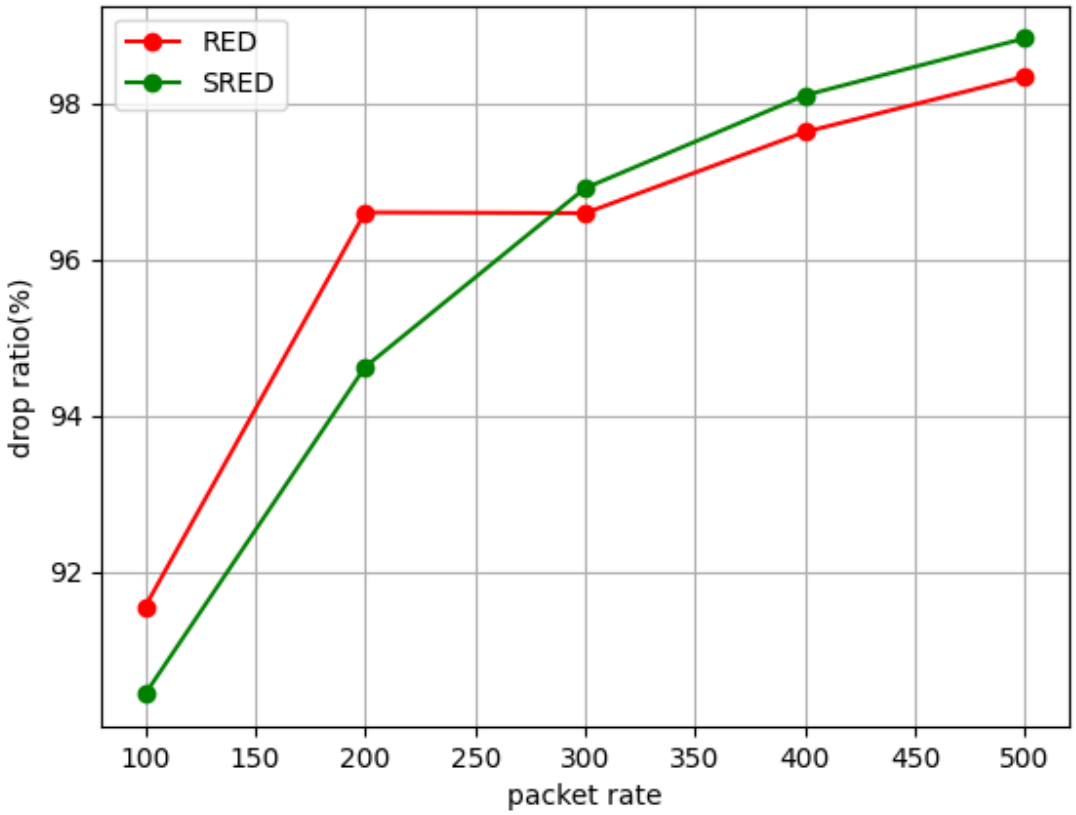

RED
SRED
total energy consumption(KJ)
flow count
100
200
300
400
500
600
700
10
15
20
25
30
35
40
45
50

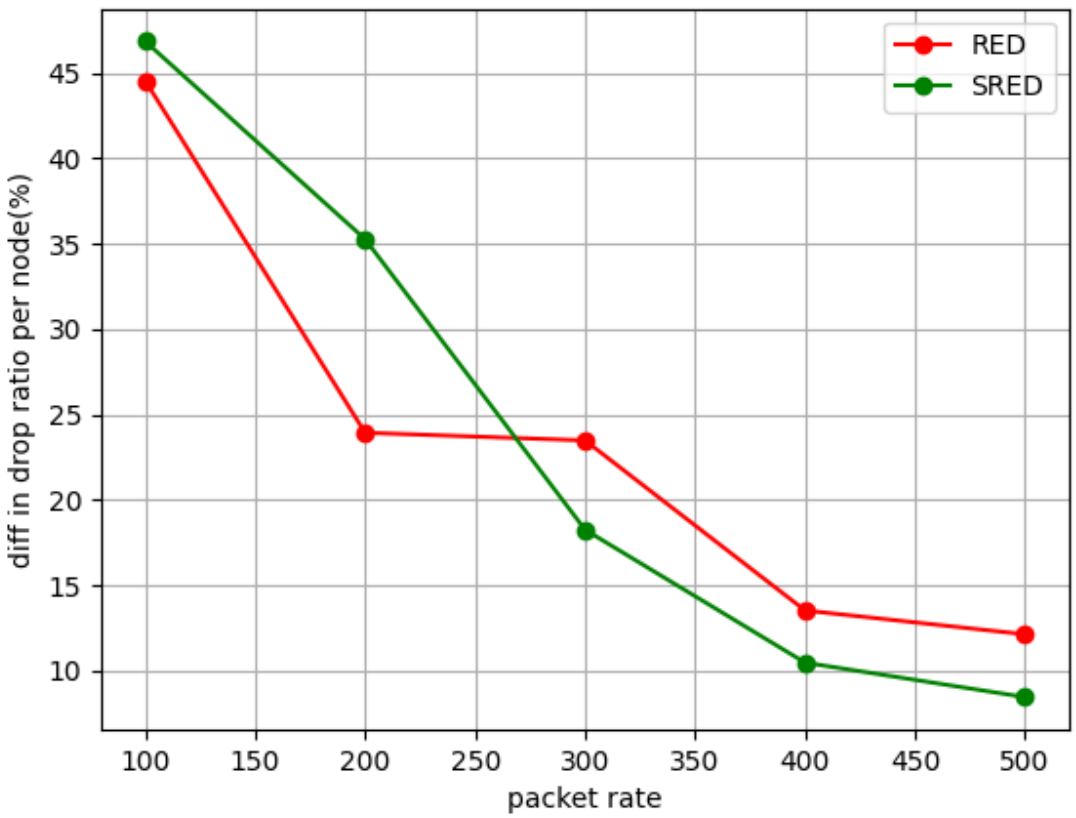
Vary Packet Rate (Per Second)

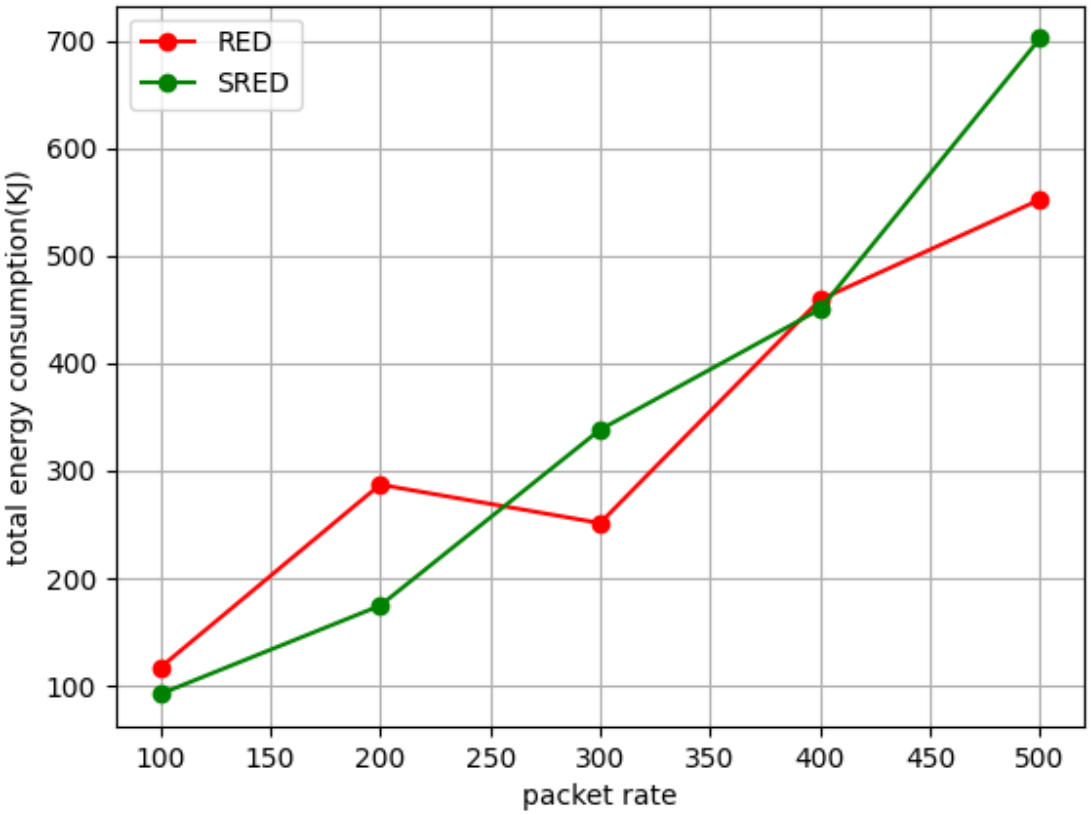



RED
SRED
0.8
0.7
0.6
0.5
0.4
average delay(s)
100
150
200
250
300
350
400
450
500
packet rate

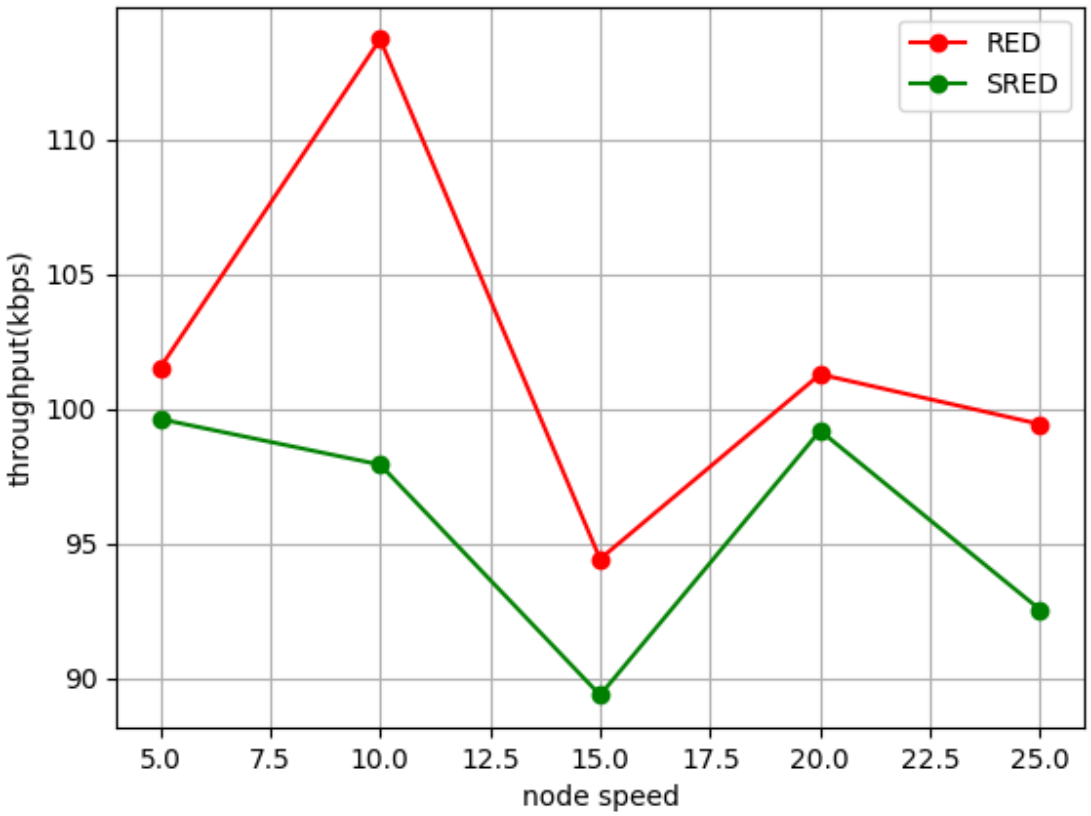

RED
SRED
delivery ratio(%)
packet rate
6
5
4
3
2
1
100
150
200
250
300
350
400
450
500

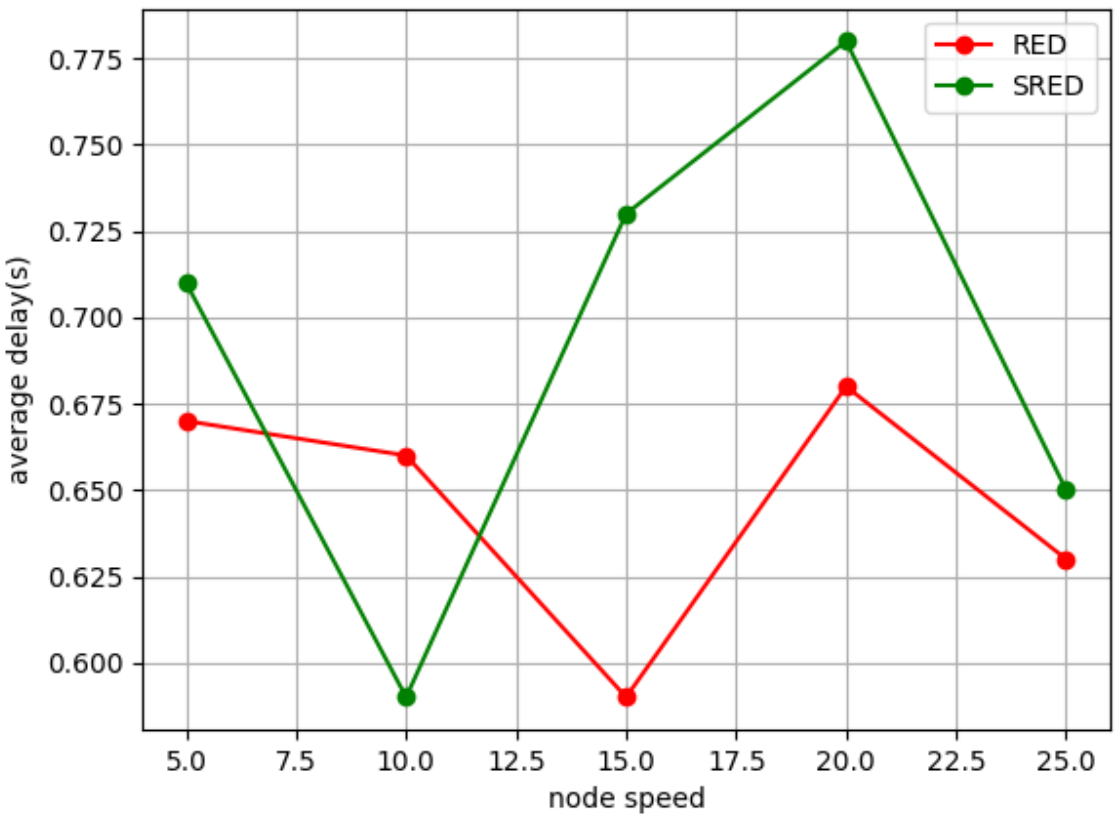

RED
SRED
98
96
94
92
drop ratio(%)
100
150
200
250
300
350
400
450
500
packet rate

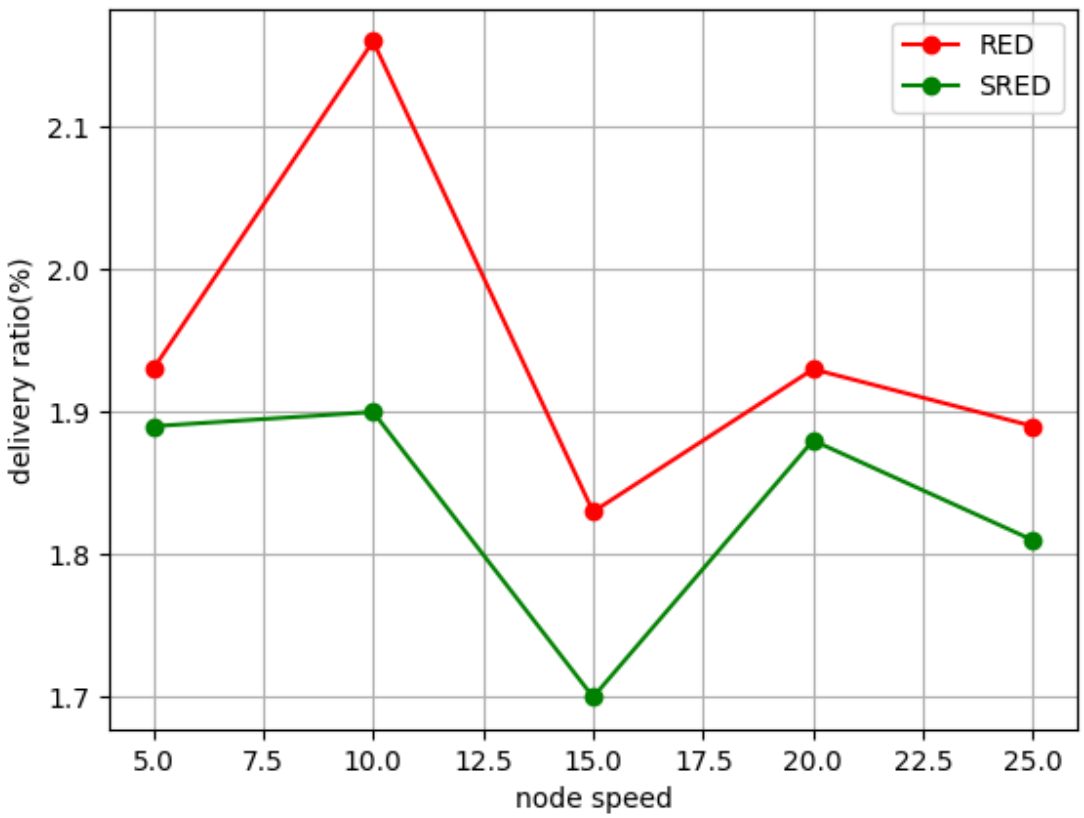

RED
SRED
diff in drop ratio per node(%)
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
500
packet rate

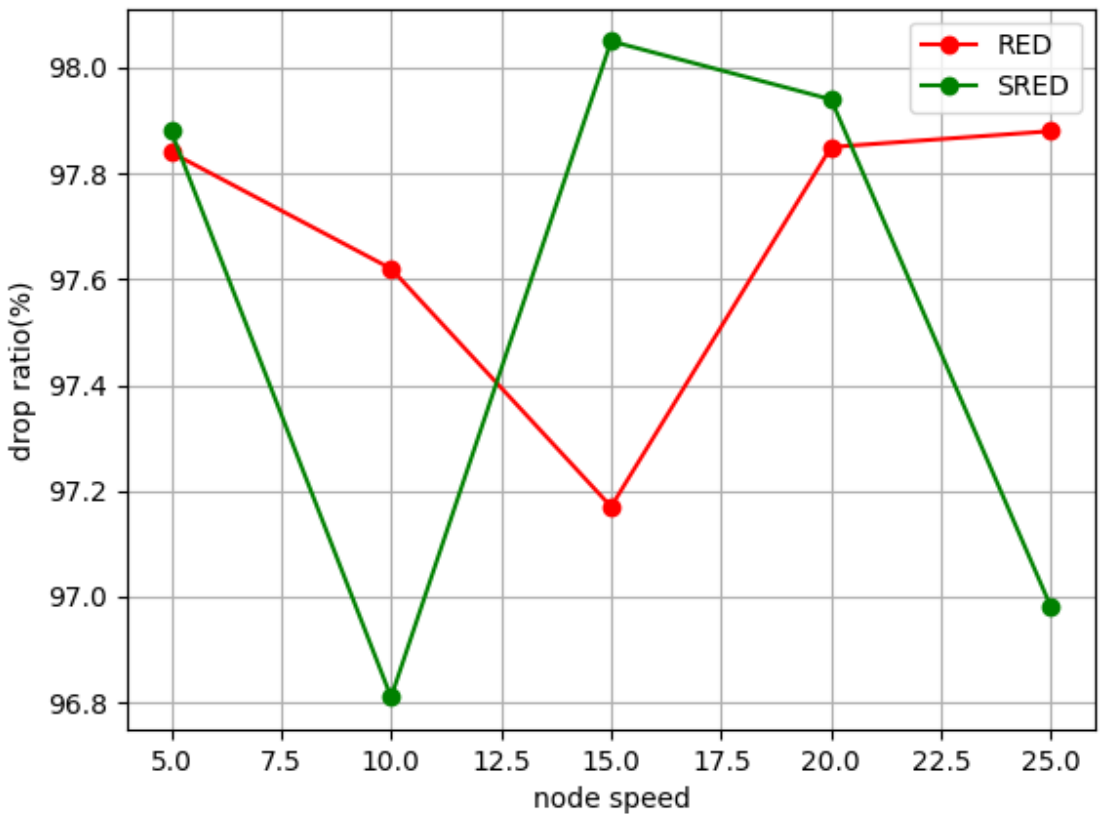

RED
SRED
total energy consumption(KJ)
packet rate
700
600
500
400
300
200
100
100
150
200
250
300
350
400
450
500

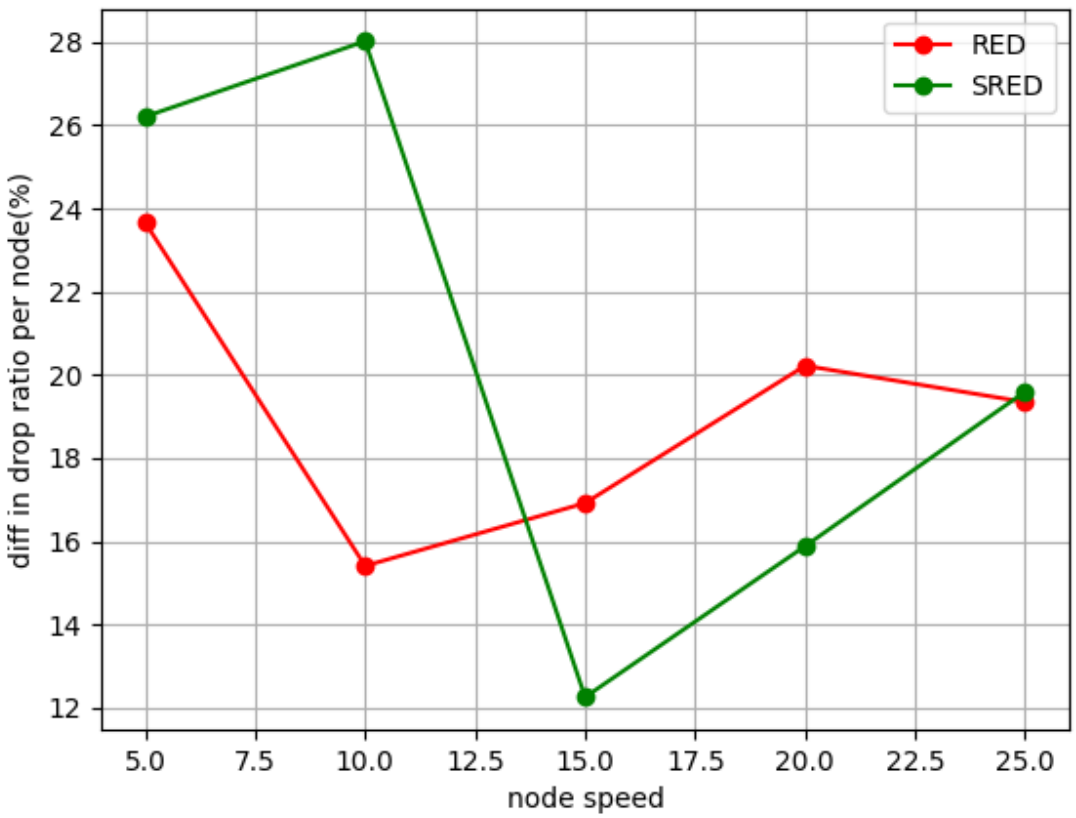
Vary Node Speed (Meter Per Second)

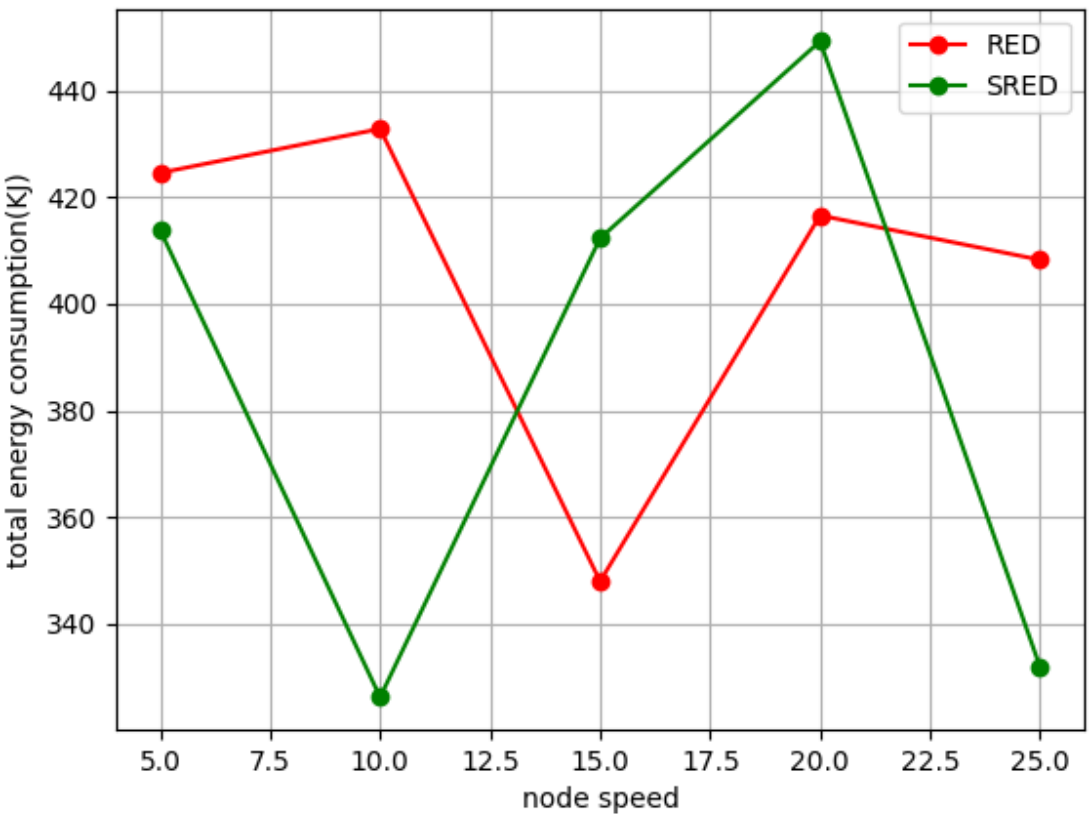



RED
SRED
0.775
0.750
0.725
0.700
0.675
0.650
0.625
0.600
average delay(s)
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
node speed


RED
SRED
delivery ratio(%)
2.1
2.0
1.9
1.8
1.7
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
node speed


RED
SRED
98.0
97.8
97.6
97.4
97.2
97.0
96.8
drop ratio(%)
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
node speed


RED
SRED
diff in drop ratio per node(%)
node speed
28
26
24
22
20
18
16
14
12
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0


RED
SRED
total energy consumption(KJ)
440
420
400
380
360
340
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
node speed

Summary Findings

1. For Most of the graphs SRED and RED follow the same shape, which is expected as SRED is a variant of RED.
2. If we compare RED with SRED, SRED has more fairness as packet dropping probability depends on the hit rate. If the hit occurs, it is more probable to drop the packet.
3. In the graph, we can also see that almost all the time SRED performs better in terms of fairness.
4. Most of the graphs of SRED are smoother than their RED counterparts. This is because SRED can stabilize the queue in different load levels by actively changing the probability function depending on the queue size.