1. NETWORK TOPOLOGIES UNDER SIMULATION

Two wireless networks were simulated in this assignment - 802.11 static and 802.15.4 mobile.

By default, 50 nodes were kept in the simulations. Node numbers, total flow and velocity (in case of mobile networks) were varied as necessary in both the topologies.

The *nam* animations of the networks are shown below:

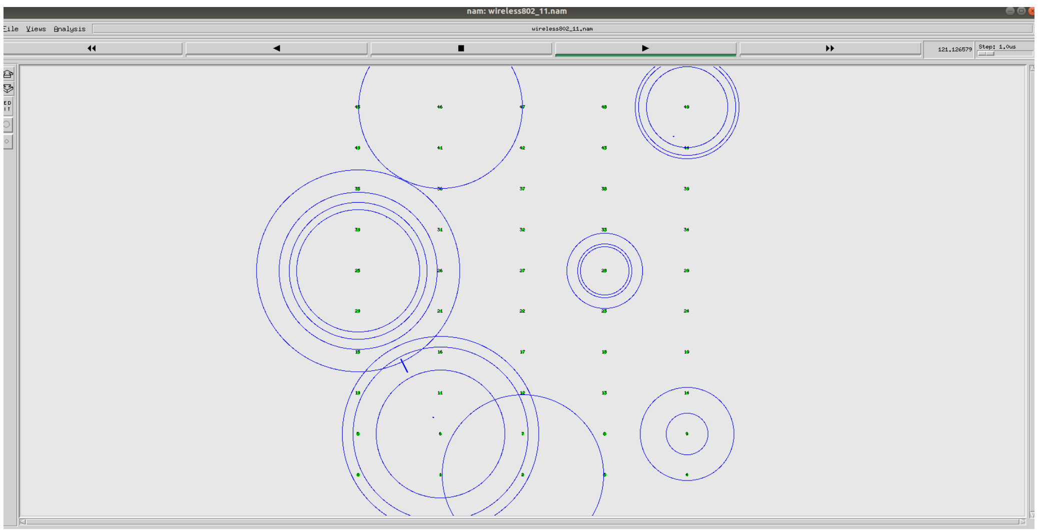

Fig: Wireless 802.11 static topology showing 50 nodes.

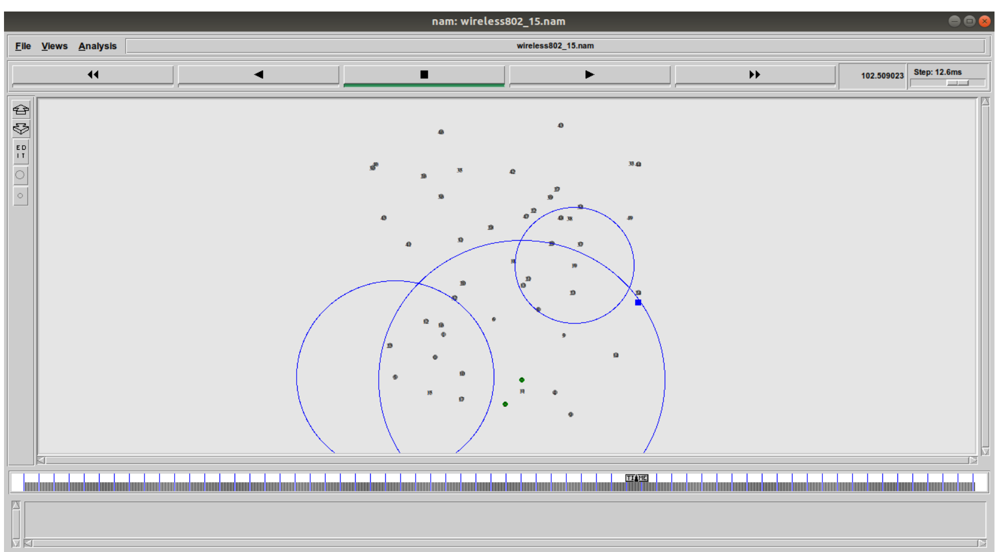

Fig: Wireless 802.15.4 mobile topology with 50 nodes.

2. <u>Parameters under variation</u>

The number of nodes, total flow and the number of packets per second were varied in both the networks under simulation. In addition, the coverage area was varied in the static network, by changing the coverage distance of each router. In the mobile network the velocity was varied as well.

Each of these parameters were varied over five different values. For each value, five performance metrics were calculated – total energy, throughput, end-to-end delay, delivery ratio and drop ratio. Corresponding graphs showing how these performance metrics change as the values of the parameters change were plotted using *gnuplot.*

The graphs have been shown in Section 3 below.

3. <u>Modifications made in the simulator</u>

Two modifications were made in the simulator, with the expected outcome being an improvement in the measured performance metrics.

a. **An improved mechanism for calculating the round-trip time (RTT) for TCP**: The smoothed RTT (srtt) and variance of RTT (rtt_var) are normally calculated using the two constants alpha and beta, with alpha set to a value of 0.125 and beta to 0.25.

   The equations used to calculate srtt and rtt_var are as follows:

   i) srtt = (α * new_rtt) + ((1 - α) * old_srtt)

   ii)rtt_var = (β * |new_rtt – new_srtt|) + (1 - β) * old_rtt_var

   While these give a sufficiently good value of the srtt and rtt_var, it is an obvious intuition that changing these values depending on the rate of change of the rtt would bring an improvement in their measurements. This intuition was followed in the assignment.

   The rate of change of rtt (k) was calculated for each packet transmission, according to the following formula:

   k_{n+1} = (new_rtt – old_srtt)/old_srtt

   This was then added to alpha and subtracted from beta to get their new values.

   $\alpha_{n+1} = \alpha_n + k_{n+1}$

$$\beta_{n+1} = \beta_n + k_{n+1}$$

The srtt and rtt_var were then calculated using these new values. The RTO was then calculated using the modified values of srtt and rtt_var.

The changed were brought in the function *rtt_update(tao)* in the file *tcp.cc.* A code snippet showing the portion of the code where the changes have been brought is shown below:

```
/* This has been modified to use the tahoe code. */
void TcpAgent::rtt_update(double tao)
{
        double now = Scheduler::instance().clock();
        if (ts_option_)
                t_rtt_ = int(tao /tcp_tick_ + 0.5); //t_rtt_ = current rtt_; rtt_n+1
        else {
                double sendtime = now - tao;
                sendtime += boot_time_;
                double tickoff = fmod(sendtime, tcp_tick_);
                t_rtt_ = int((tao + tickoff) / tcp_tick_); //t_rtt_ = current rtt_; rtt_n+1
        }
        if (t_rtt_ < 1)
                t_rtt_ = 1;

        if (t_srtt_ != 0) {
                t_rtt_ = t_rtt_ >> T_SRTT_BITS;
                t_rttvar_ = t_rttvar_ >> T_RTTVAR_BITS;
                double k_ = (t_rtt_ - t_srtt_)/t_srtt_;
                if(k_ < 0) k_ = -k_;
                double alpha_ = 0.125 * (1 + k_);
                double beta_ = 0.25 * (1 - k_);

                t_srtt_ = ((1 - alpha_) * t_srtt_) + (alpha_ * t_rtt_);
                if(t_srtt_ <= 0) t_srtt_ = 1;

                int delt = t_srtt_- t_rtt_;
                if(delt < 0) delt = -delt;
                t_rttvar_ = ((1 - beta_)*t_rttvar_) + (beta_ * delt);

                t_rtt_ = t_rtt_ << T_SRTT_BITS;
                t_rttvar_ = t_rttvar_ << T_RTTVAR_BITS;

        } else {
                t_srtt_ = t_rtt_ << T_SRTT_BITS;                // srtt = rtt
                t_rttvar_ = t_rtt_ << (T_RTTVAR_BITS-1);        // rttvar = rtt / 2
        }

        t_rtxcur_ = (((t_rttvar_ << (rttvar_exp_ + (T_SRTT_BITS - T_RTTVAR_BITS))) +
                t_srtt_)  >> T_SRTT_BITS ) * tcp_tick_;

        return;
}
```

b. **Congestion Control Modification:**

We have modified congestion control in two ways. We added two cases in TcpAgent::opencwnd() function.

1. Constant window size: We added a case to support constant cwnd size which can be varied by setting a new variable constant_cwnd_size_ in tcl script.
2. Accommodating the fluctuation of RTTs of the network path.

We followed the paper titled “IMPLEMENTATION OF NEW TCP CONGESTION CONTROL MECHANISM OVER LONG TERM EVOLUTION ADVANCED NETWORKS” by Ghassan A. Abed, Mahamod Ismail and Kasmiran Jumari and added another case in TcpAgent::opencwnd() function.

The sender TCP updates its congestion window size in the congestion avoidance phase according to this equation:

cwnd = cwnd + (f / cwnd)

This mechanism updates f for the above equation every time that TCP sender receives a new ACK packet:

f = f / (cwnd * pow (cwnd, k_parameter))

Code snippet:

```
//modification
case 9:
        //constant window size
        cwnd_ = constant_cwnd_size_;
        //printf("%d----------OOOO\n\n", int(cwnd_));
        break;

case 10:
        //from paper
        f = wnd_const_ * ssthresh_ / (cwnd_ * pow(cwnd_, k_parameter_));
        cwnd_ = cwnd_ + (f/cwnd_);
        break;
```

4. **<u>Results with graphs:</u>**

a.i) The results obtained with the original, unmodified simulator are given below:

a.i.a. **802.11 static network:**

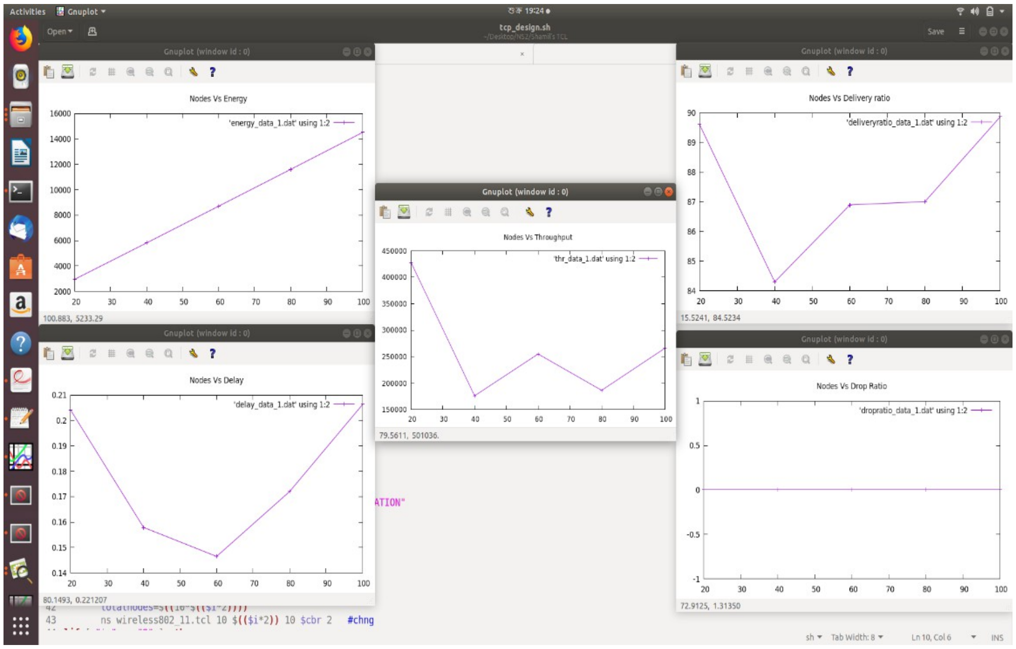

Fig: Variation of the performance metrics with the number of nodes

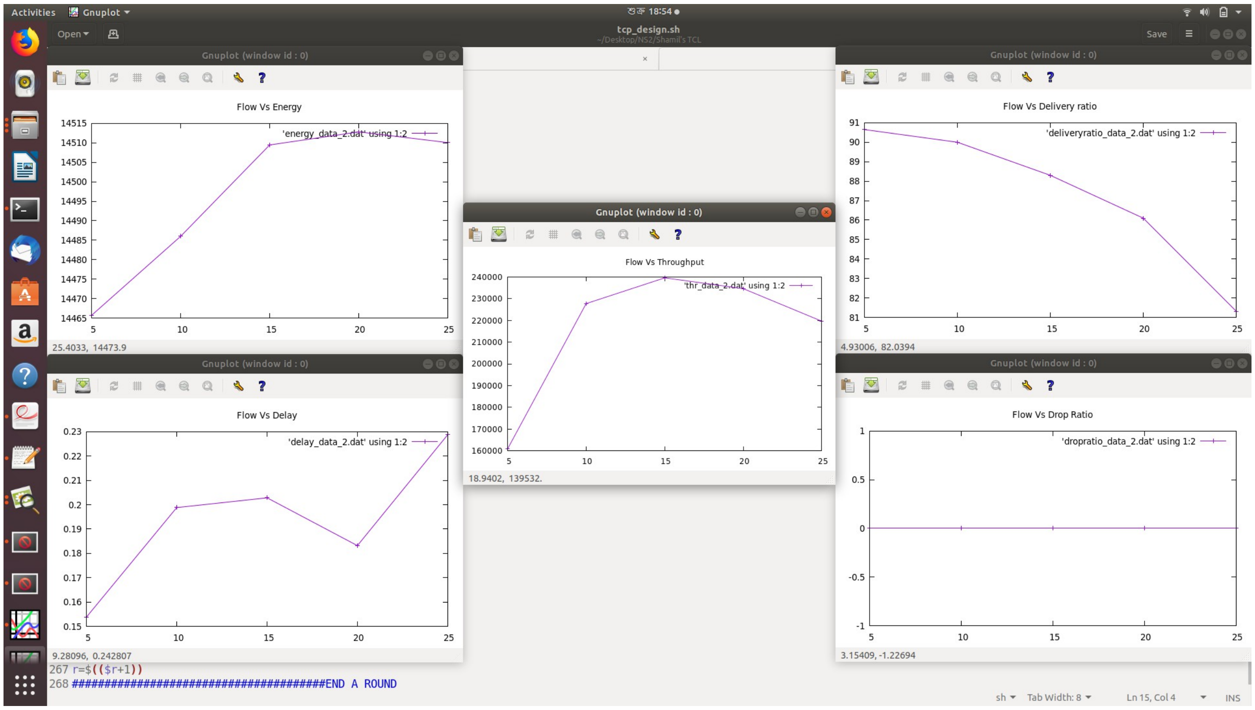

Fig: Variation of the performance metrics with the total flow

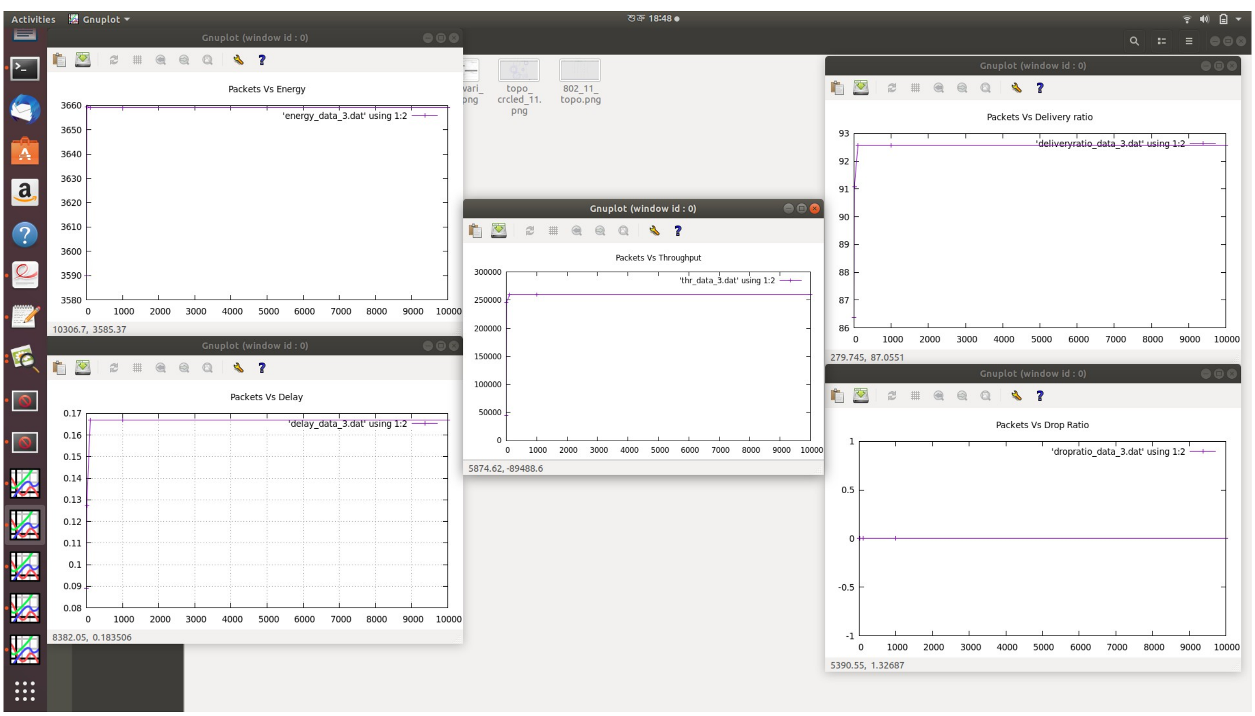

Fig: Variation of the performance metrics with packet rate

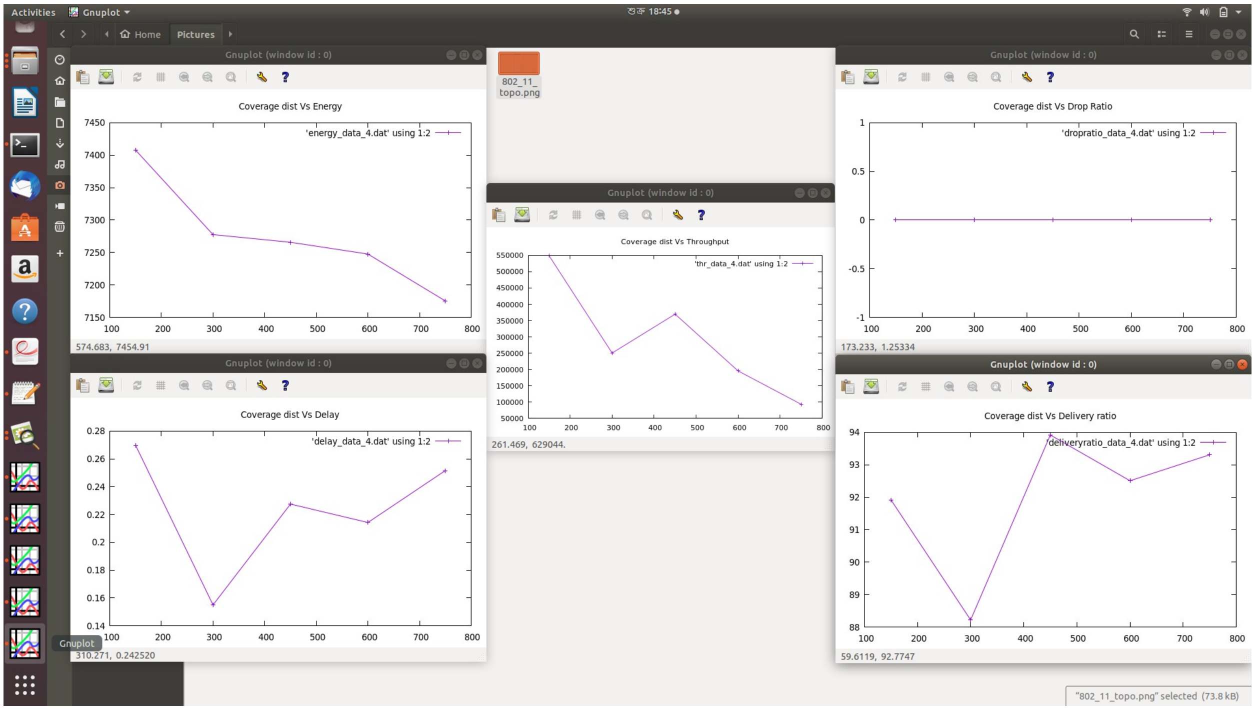

Fig: Variation of the performance metrics with coverage distance

b. **802.15.4 mobile network:**

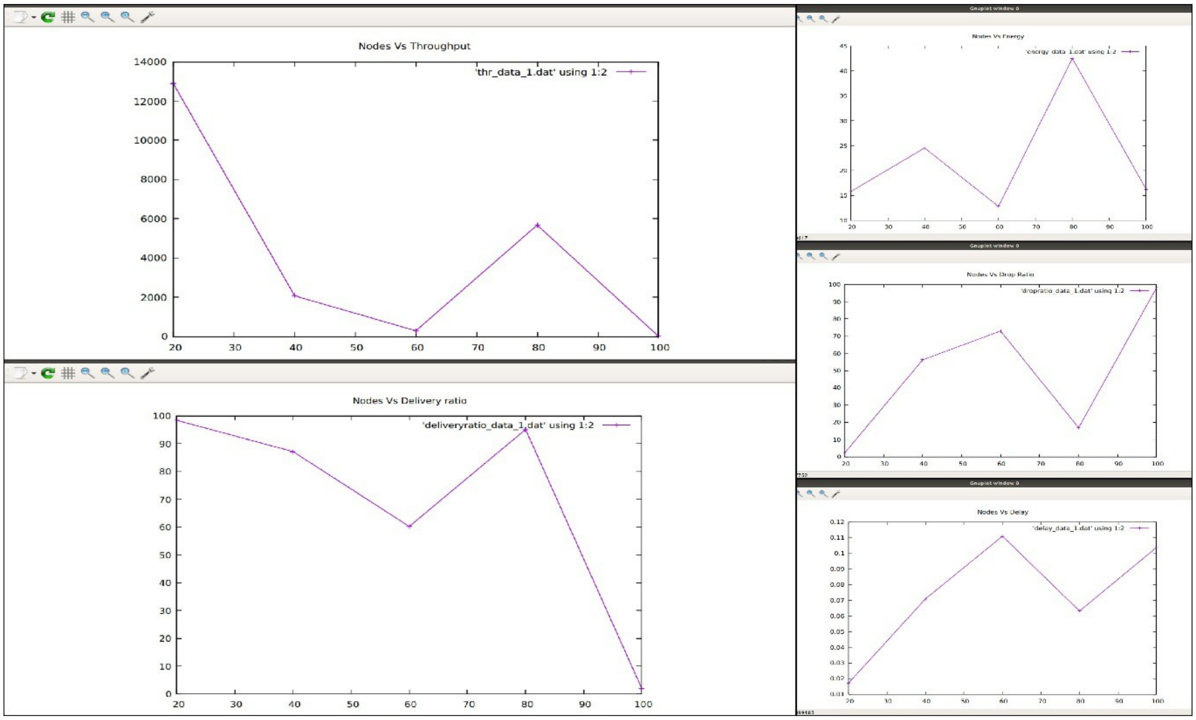

Fig: Variation of the performance metrics with the number of nodes

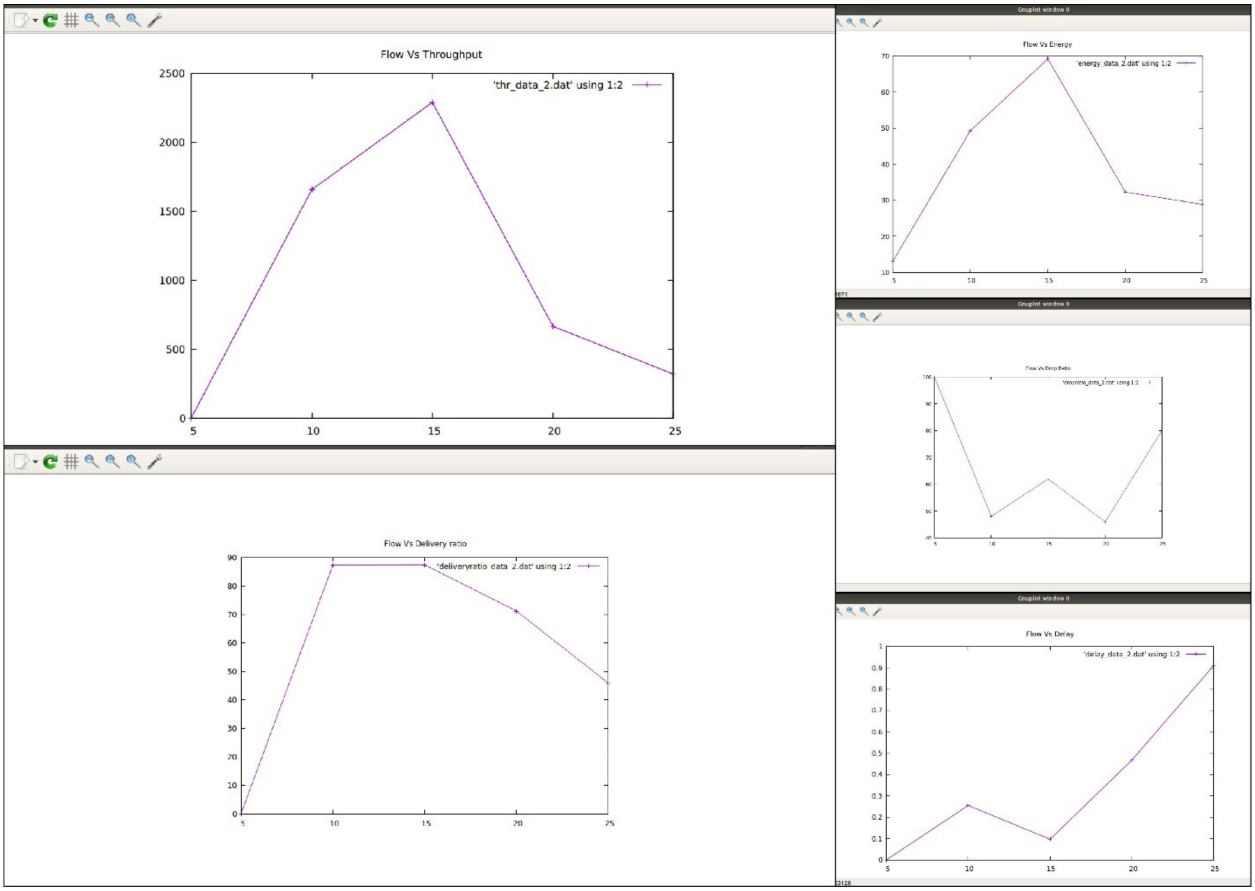

Fig: Variation of the performance metrics with the total flow

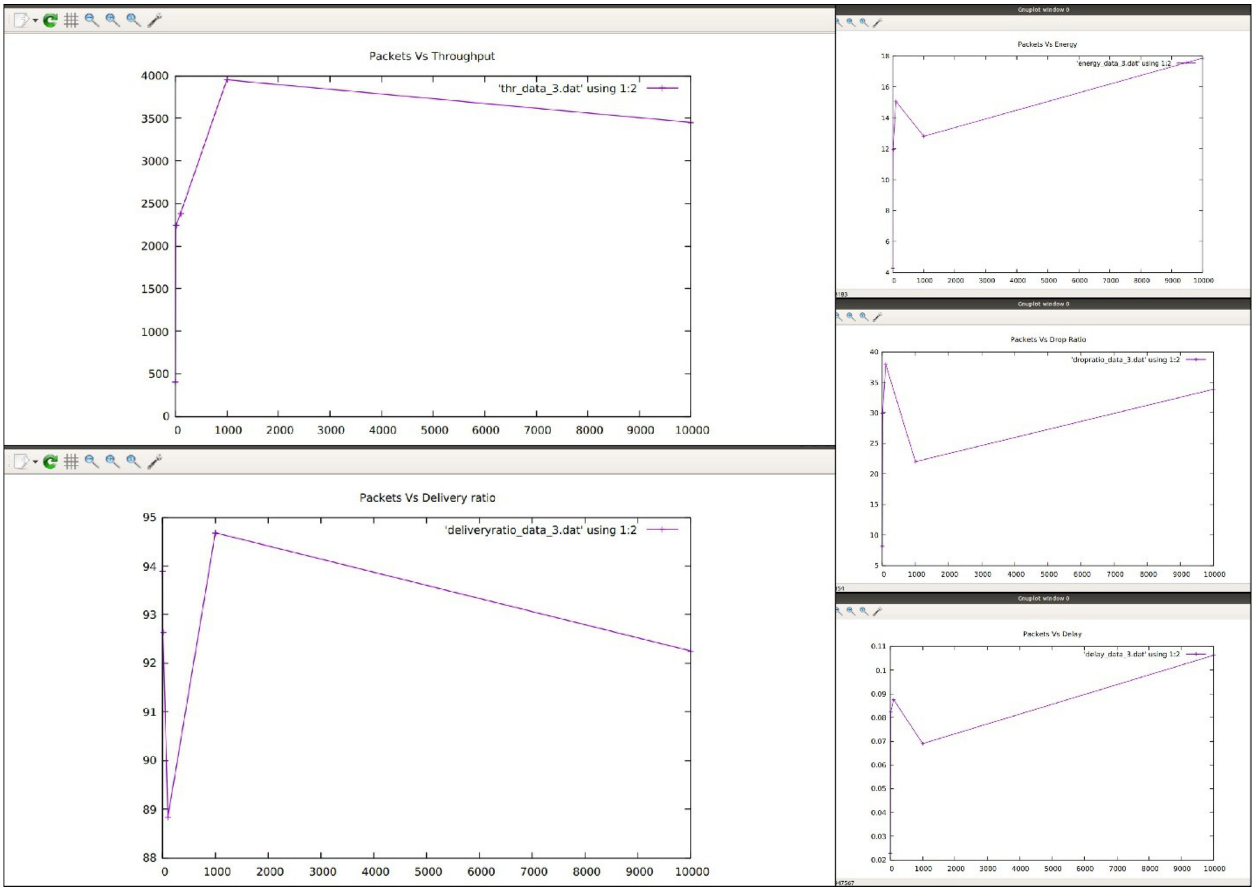

Fig: Variation of the performance metrics with packet rate

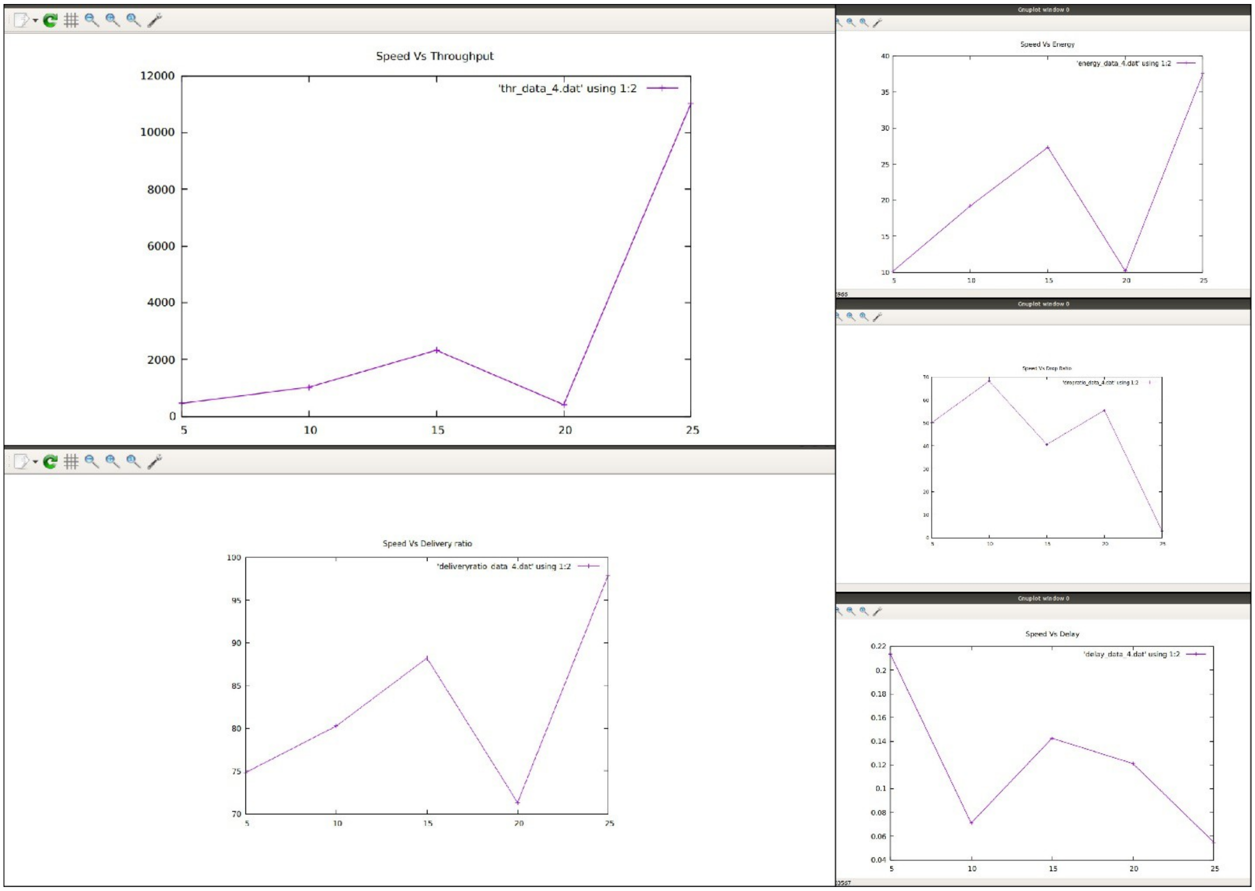

Fig: Variation of the performance metrics with velocity

ii) The results obtained with the modifications mentioned in the previous section are given below:

b.i.a. **802.11 static network:**

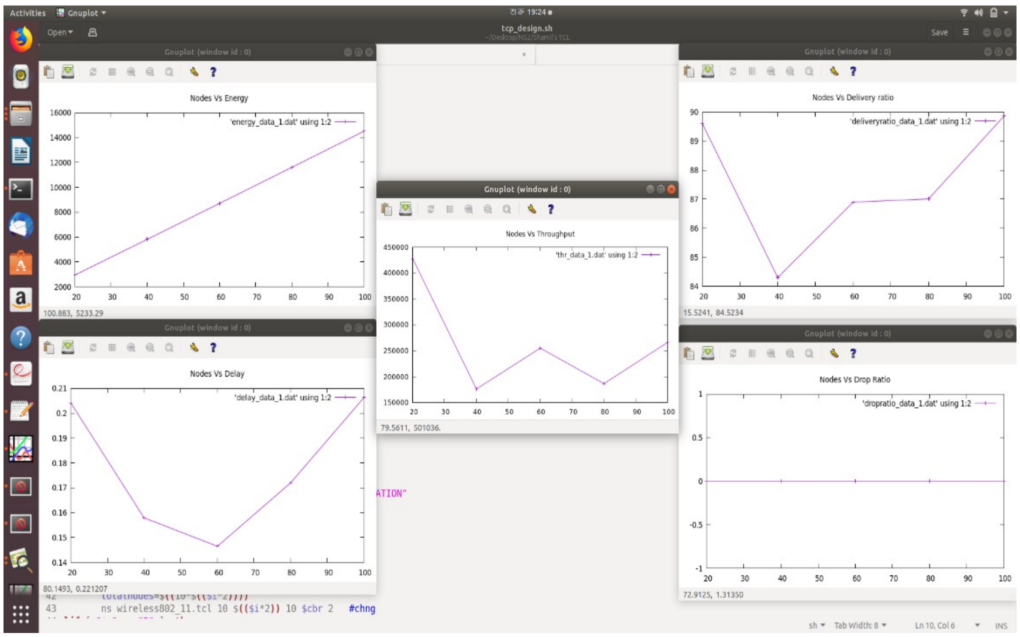

Fig: Variation of the performance metrics with the number of nodes

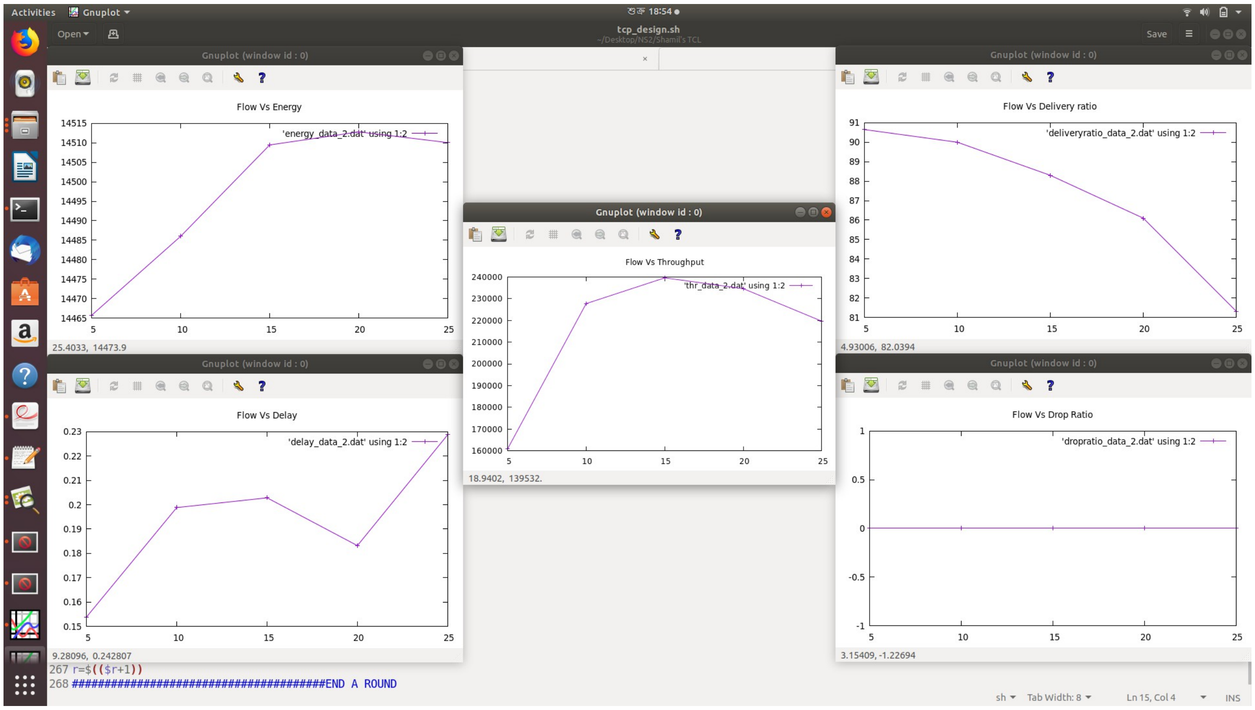

Fig: Variation of the performance metrics with the total flow

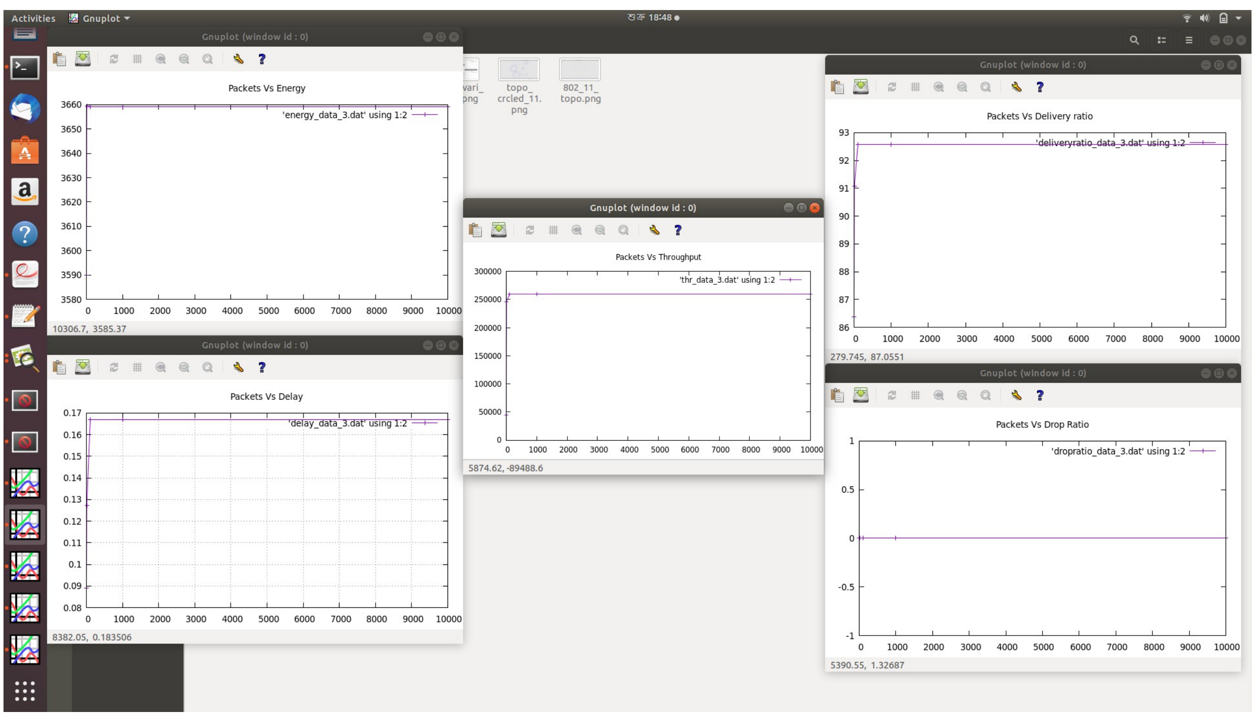

Fig: Variation of the performance metrics with packet rate

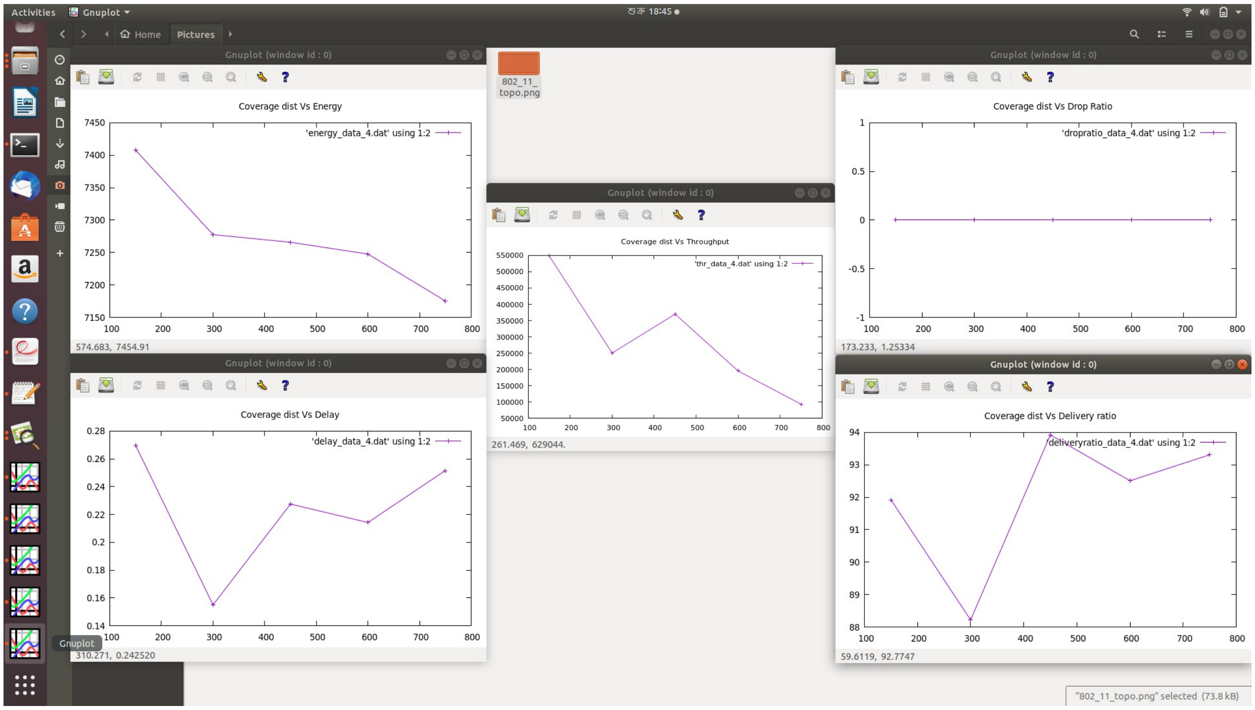

Fig: Variation of the performance metrics with coverage distance

c. **802.15.4 mobile network:**

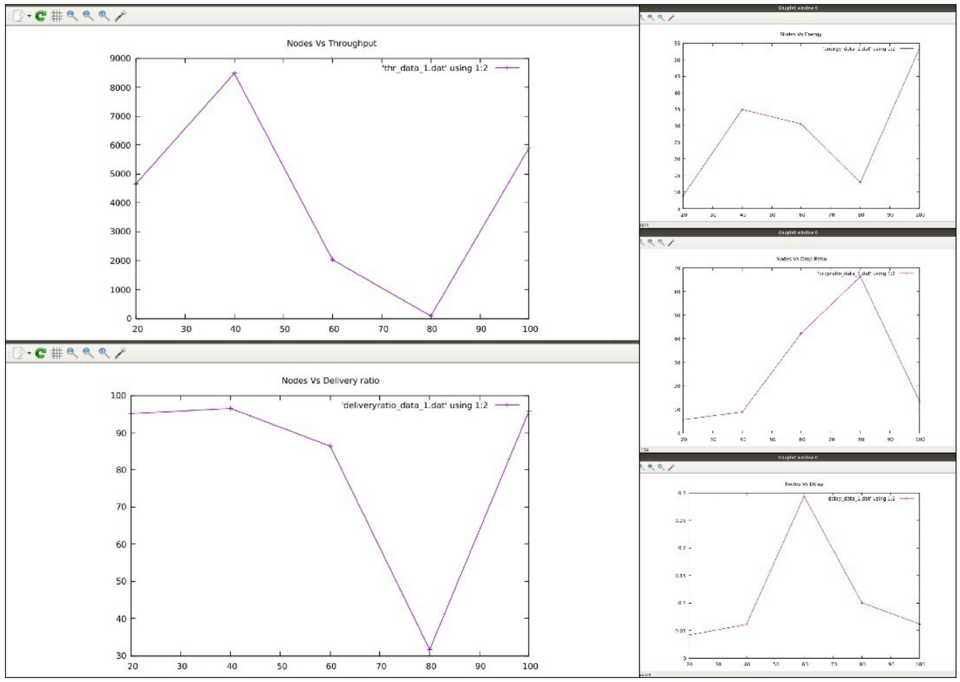

Fig: Variation of the performance metrics with the number of nodes

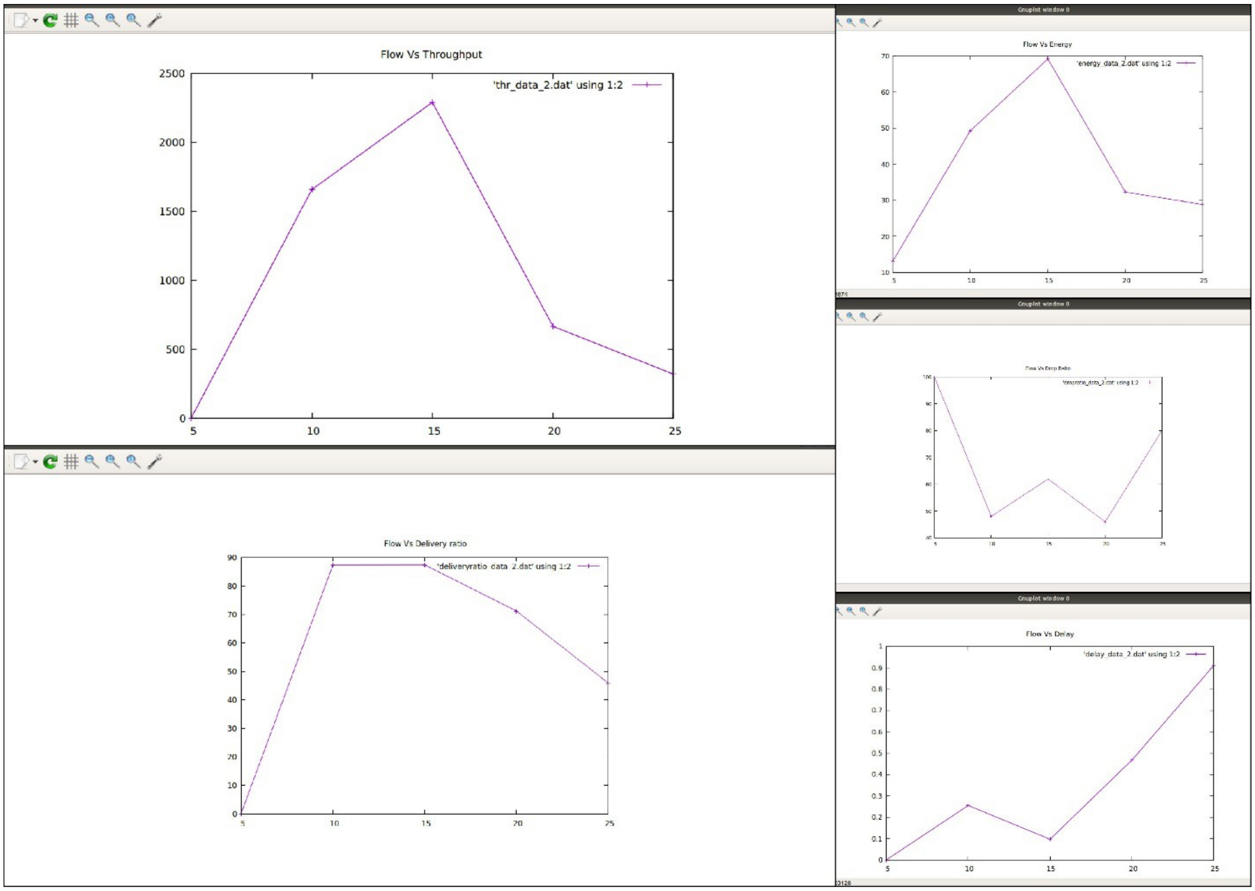

Fig: Variation of the performance metrics with the total flow

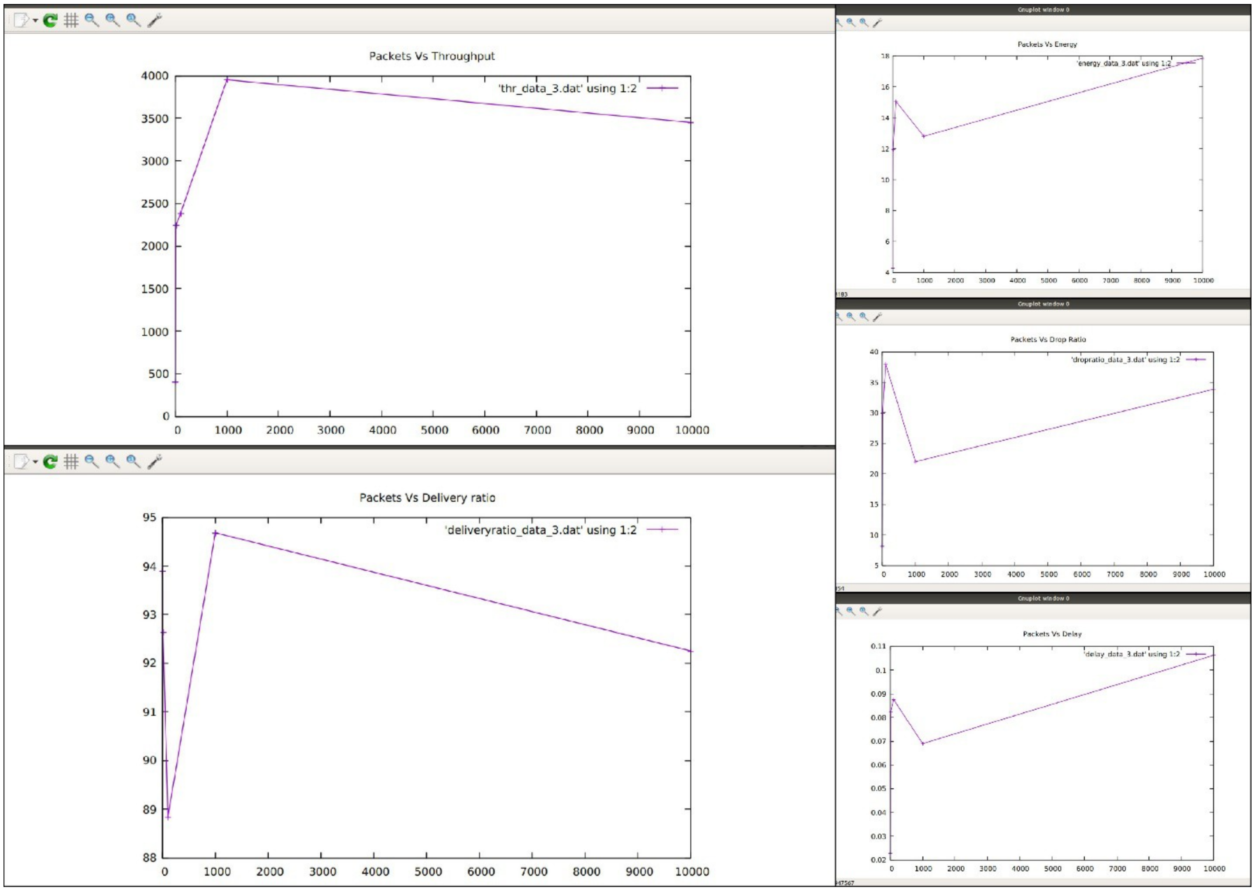

Fig: Variation of the performance metrics with packet rate

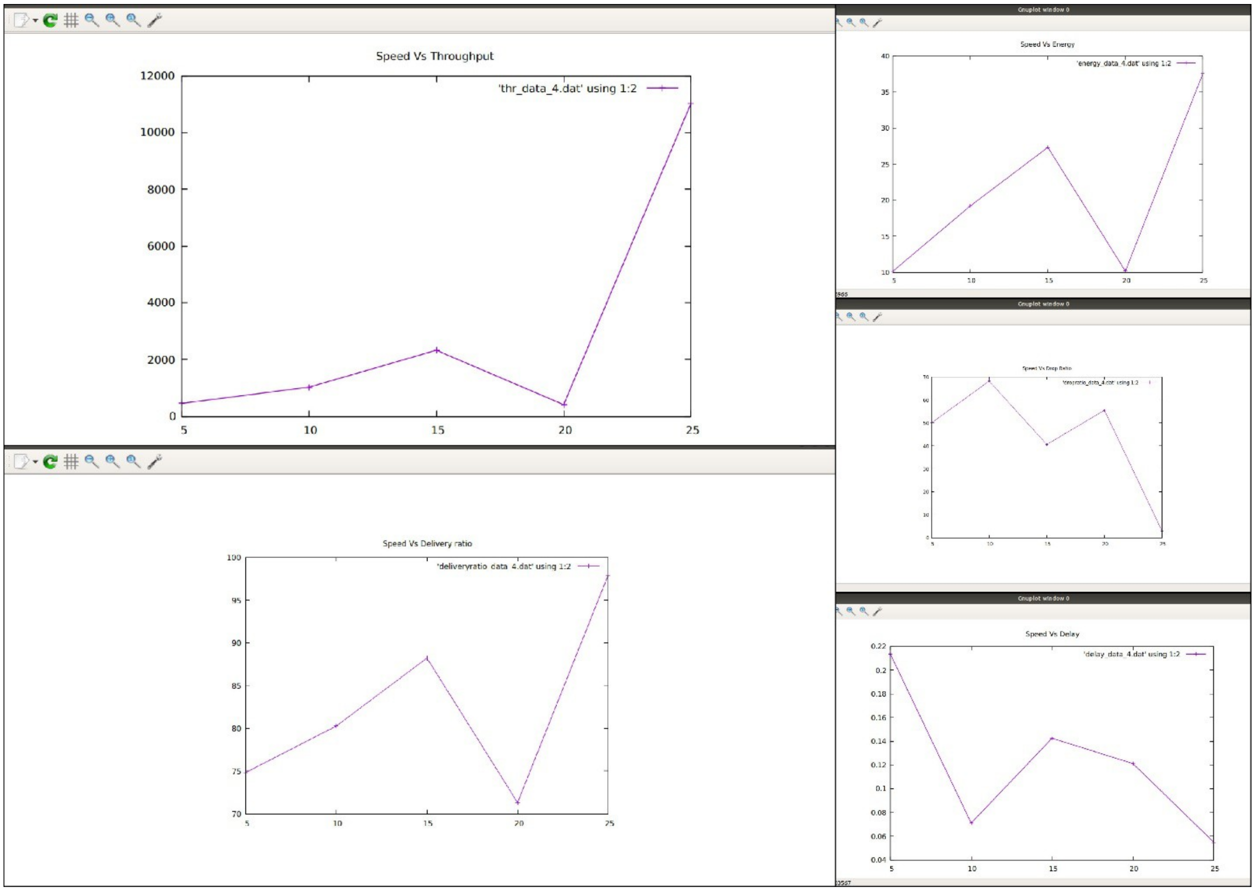

Fig: Variation of the performance metrics with velocity

5. **Bonus Tasks:**

The bonus tasks that were implemented in this assignment are:

a) **A satellite network was simulated.**

An .awk file was written to analyze the events occurring within the network. The throughput, delivery ratio and drop ratio were calculated and displayed.

The number of nodes (satellites) were also varied, and the changes in throughput, delivery ratio and drop ratio were plotted in graphs.

b) **Throughputs of each of the nodes (per-node throughput) were calculated and the results displayed.**

c) **Mean and current jitter values were calculated as an extra performance metric.**