A network can run at its ideal bandwidth thanks to the congestion mechanism, which results in little delay and great throughput. The most important factor is for users to regulate their transmission rates, or when to send more and when to send less to avoid overtaxing the network. To arrive at the best option, this research investigates several reduction strategies. Traditional congestion models demonstrate that when the load is low, the throughput keeps up with us, but as the demand rises, the throughput grows exponentially until the load reaches a certain level at which it is so great that throughput drops, and this is what is meant by congestion. With light loads, the reaction time is pretty quick. As the load grows, it does so linearly until it reaches the cliff point, where the reaction time climbs dramatically. Utilizing the window approach to regulate the number of packets sent via the network is the current congestion avoidance technique. Limiting the user's packet transmission speeds is an alternative strategy. Dynamic control, which is the foundation of the congestion control mechanism, might be used in both techniques. We employ the Binary feedback technique, which demonstrates that when the congestion bit is set to 1, it indicates an overflow, and when it is set to 0, it indicates an underflow. As a result, users will adjust their transmitting rate appropriately. The packet header's Qos includes this bit. We assume that for every user, the feedback and control loop is synchronous. According to the diagram, each user's load is totaled and then compared to a threshold; if either value is more or lower than the threshold, feedback is delivered. There are several approaches to managing congestion, but to compare them, we employ metrics like fairness, distributedness, efficiency, and convergence. Efficiency is the state in which the entire load generated by the users is equal to the maximum load that the network can support. Fairness refers to the fairness point, which states that all users are consuming the bandwidth equally or transmitting an equal number of loads. A centralized system necessitates that everyone is fully aware of the state, yet doing so is extremely difficult and expensive. We presume that only the hosts of the data are aware of its underflow or overflow conditions. The time needed to attain equilibrium is referred to as convergence. The trajectory for 2 users in the network is shown in figure 5 of the study when additive increase and multiplicative reduction are used, which is the best method for dynamically regulating the transmitting rate. The fundamental tenet of AIMD is that when a load is underflowing, it will raise that load by a constant amount, but when a load is overflowing, it will reduce that load rapidly, leading to the construction of a zigzag line that eventually converges in the middle and gives us the optimal outcome. Since scaling an algorithm depending on the underlying software and hardware would result in a more sophisticated algorithm, the fundamental premise we have assumed is that an algorithm is independent of the underlying software and hardware. Rounding off these numbers might result in significant mistakes since the window size is another limitation that we take into account. Additionally, factors like the minimum hardware needed and the number of necessary multiplications might influence the implementation. We found that when congestion management is used, we can use the network more effectively. To ensure that fairness stays the same or improves, we must lower the burden more multiplicatively. Fairness must be increased, and optimization must demand these adjustments. The additive rise and multiplicative decline approaches were ultimately selected.