Implementation and Performance Comparison of Rate Adaptation Algorithms in WLANs

IEEE 802.11 networks are wireless local area networks. In most of modern IEEE 802.11 networks the physical layer supports multiple transmission rates. We can use this facility for increasing the throughput in various channel and environment conditions. For example:

1. If the channel conditions are good for transmission, also the number of nodes in the environment is less, then we can maximize the throughput by transmitting at higher rate.
2. On the other hand if the channel is having disturbances, or the distance from AP(Access Point) is too high, or there are too many collisions between packets which are transmitted from different nodes, then higher rate will cause high error in received packets. So in such case lower rate should be preferred in order the maximize number of successful transmissions.

Therefore efficient algorithms are required for determining the rate in varying conditions. Many algorithms have been proposed for rate adaptation in IEEE 802.11 networks. We have selected two of such algorithms and compared their performance in practical kind of scenario.

1. **Auto Rate Fallback (ARF) Algorithm:**

This is a simple approach to control the link rate. The decision of increasing or decreasing the rate is based on number of consecutively successfully transmitted packets or number of failed packets. If 10 consecutive packets are successfully transmitted (or 10 consecutively ack packets are received) then it attempts to increase the rate to next level. If the packet fails to be transmitted successfully, immediately after increasing the rate, it decreases the rate back to previous level. This algorithm is widely adopted because of being simple.The drawback of this algorithms is that is fails to identify difference between the reasons of collisions. Another drawback is that it attempts to increase the rate after every ten successful transmission, even if current rate is the most convenient.

2. **Adaptive Auto Rate Fallback (AARF) Algorithm:**

AARF is a modified version of ARF. The decision of increasing or decreasing the rate is again based on number of consecutively successfully transmitted packets, but this threshold number is not constant (as it was 10 in ARF). Indeed, if the first packet after increasing rate fails to transmit, then we double the threshold number, i.e. the next attempt to increase the rate will happen after 20 successful transmissions. Again if the first packet fails after attempt to increase the rate, if again increases the threshold. There is a maximum limit on this threshold (160 in our implementation). If the attempt to increase the rate succeeds, the threshold is reset to basic value (10 in this case). This algorithm gives better throughput in multi-user environment, because it tries to stabilize at most suitable rate.

We have implemented these two algorithms in a network simulator NS-2.35. NS2 is a discrete event simulator mainly for research in networking. It provides support for simulation of routing, TCP and various multicast protocols for wired and wireless networks. NS2 is written in C++ and TCL as scripting language. It also supports graphical representation of network topologies and transmission of data. We have used NS-2.35 for our project. The implementation of algorithms was done in C++ in the simulator. The generation of wireless scenario for testing of algorithms and evaluation of throughput of the algorithms was done in TCL script.

For evaluation of algorithms, we generated simple topology consisting of one access point (AP) and eight other nodes, which try to transmit data to the AP. Both algorithms were run on the same topology and the throughput of the algorithms was calculated and plotted against time.

We see that when a some new nodes are added the ARF gives better throughput for first one second, because of its tendency to maintain high rates. But as the throughput gets stabilized after 1 second the ARF has much lower performance than the AARF because of its tendency to always prompt to increase rate, even if current rate is more convenient. So, in these cases, the advance learning of AARF algorithm proves to be useful in maintaining the throughput.

The stable throughput of ARF algorithm was observed to be 4.4 MBps, while that of AARF was 6.25 MBps. Hence an improvement of 42% was observed in the throughput in highly congested kind of environment.

There are mainly three observations:

1. The algorithm must be robust enough to get stabilized at an intermediate rate, which is the remotely most convenient. As was seen in case of ARF, it always tries to increase the rate, which actually decreases the throughput.
2. Ability of the algorithm to learn about the environment dynamically makes an algorithm better in maximizing the throughput. Learning about the channel conditions and reacting in better way each time is a good direction for new researchers to work towards.
3. Some more modern algorithms support advance learning e.g. differentiating the reasons of failure (collisions or channel conditions), service differentiation etc. Maintaining the Quality of Service (QoS) with service differentiation is the field, where the results of such projects are extremely useful.

Mentor: Dr. Venkata Ramana Badarla

Team:
Dinesh Kumar Maurya (B14BS005)
Vijay Kumar Paliwal (B14CS040)

Date: 25.04.2016