Providing Adaptive QoS in Wireless Networks by Traffic Shaping

Tomas Lennvall, Gerhard Fohler
Deptartement of Computer Engineering
Mälardalen University, Västerås, Sweden
{tomas.lennvall,gerhard.fohler}@mdh.se

1 Overview

Our goal is to stream video over a wireless network by dynamically adapting the size of the stream according to the fluctuating available bandwidth in the network.

In this paper we propose an architecture that provides QoS by dynamically adapting the transmission rate of nodes to match the currently available bandwidth of a wireless network. The architecture prioritizes real-time traffic over non real-time traffic when transmitting packets. The traffic adaptation serves the purpose of minimizing the network congestion occurring due to high load.

The architecture consists of a *bandwidth predictor* that first uses a simple probe-packet technique to predict the available bandwidth of the network. Then, exponential averaging is used to predict the future available bandwidth based on the current measurement and the history of previous predictions.

This predicted bandwidth is then fed into the *traffic shaping* part of the architecture, which adjusts the transmission rate of the node accordingly.

As presented in [4, 2], the basic idea of applying traffic smoothing for network transmission is to smooth a bursty stream of data into a constant stream of data by using the leaky bucket algorithm. The rationale is that by smoothing out bursts the transmission is evenly spread out over time in order to reduce the probability of congestion and collisions on the network. It is also a way to control the rate of the transmitted traffic generated by each node.

As motivation for the work in this paper we use the two application scenarios described below.

The first application is a streaming server capable of dynamically switching between a small number of differently sized versions [5] of the same stream. Synchronization of the streams is based on GOPs, i.e. a switching between streams always occurs at a GOP start. The server switches the transmission between these streams according to the currently available bandwidth predicted by our method.

The second application is also a streaming server but it uses a web camera to capture pictures which are then encoded into a video stream with the possibility to control the stream size by adjusting a quality parameter of the encoding. The quality parameter is determined based on the bandwidth prediction we perform. This stream is then transmitted onto the network in a way so that it fits the available bandwidth.

2 Architecture

In this paper we assume all nodes are connected using the *Infrastructure* connection, i.e all nodes are connected to an *Access Point (AP)*. Furthermore we assume that there is only one AP in the system, hence no roaming between different APs can occur even though nodes are mobile. We assume that nodes are always within transmission and reception range of the AP.

The basic idea is that each node is assigned a proportional share of the total available bandwidth of the wireless network and each node stays within its limit. Here we assume that when the available bandwidth fluctuates, the proportional share of each node also fluctuates. We consider the node containing a streaming server most important, and therefore we assign it a larger share of the available bandwidth than the other nodes. Note that, because of the previously mentioned fluctuation of available bandwidth, the assigned share for each node will also fluctuate over time. This implies that the traffic generated by a node must be dynamically adapted to conform to the currently available bandwidth share for that node.

On each node the bandwidth assigned to that node is shared between real-time and non real-time traffic. Because we consider the video stream to be real-time, we assign a higher share of the allocated bandwidth to real-time traffic in order to prioritize the transmission of the video stream.

An overview of the architecture is presented in figure 1.

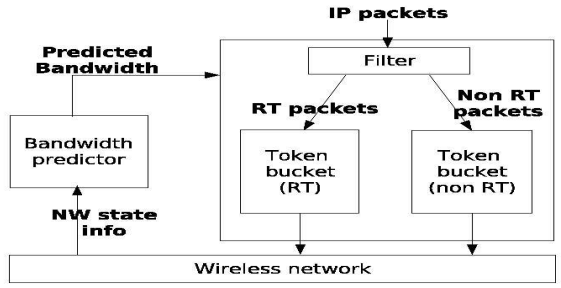

Figure 1. Architecture of bandwidth prediction and traffic shaping

In order to properly adjust the transmission rate, the first part of our architecture, the *bandwidth predictor*, predicts the future bandwidth based on the current available bandwidth and previous bandwidth predictions.

The bandwidth prediction is then fed into the second part of our architecture, the traffic shaper. It shapes the traffic according to the available network bandwidth. By separating the handling of real-time and non real-time traffic by using two token buckets (see Figure 1), we prioritize real-time traffic by giving it a larger share of the bandwidth assigned to the node.

Available bandwidth is predicted between one sender and one receiver, i.e if there are multiple receivers, available bandwidth must be predicted for each, because the results can be different depending of the fluctuations of the wireless network. In this paper we assume that each sender has only one receiver present.

3 Bandwidth Prediction

Our bandwidth prediction uses a well known *packet-pair* probing technique detailed in [3], which is a way to determine the end-to-end characteristics of a network path mainly for Internet, which we apply to wireless networks. The network state can be determined by measuring the time delay of the probe packets.

In order to dynamically react to the varying bandwidth of the network, we have to periodically repeat the bandwidth prediction. Here, we repeat the probing and prediction with a period of 1 second.

In packet pair probing, the sender transmits two probe packets (of identical length), back to back, to a receiver. The receiver measures the delay between the probe packets and returns this information to the sender. This delay gives an indication of the current network load, a high delay indicates a high network load and vice versa. Formula 1 shows our simple calculation for the measured bandwidth.

$$BWT = L/\Delta_T \tag{1}$$

Where BWT is the resulting bandwidth, L is the probe packet length, and Δ_T is the measured delay between the probe packets $(T_2 - T_1)$.

In order to predict the future available bandwidth we use *exponential averaging*, which is a technique used to examine and average a sequence values along a time series which enables us to make a prediction based on previous predictions as well as the current network load.

Formula 2 shows how we predict the bandwidth:

$$P_k = \alpha BWT_k + (1 - \alpha)P_{k-1} \tag{2}$$

Where P_k is the future prediction, P_{k-1} is the previous prediction, BWT_k is the current bandwidth measurement, and α is a constant used to determine how important the history vs. the current measurement is in the prediction.

We implemented the bandwidth predictor as a user level program running on Linux running the 2.6 kernel. The program uses the *libpcap* [6] library to transmit and receive the probe packets directly to the network card (bypassing the higher layers). On the sender side, the program transmits the two probe packets back to back and then waits for a return packet containing the time difference measured at the receiver. The receiver waits for the two probe packets, takes time stamps when they arrive, and finally calculates the difference which is then transmitted back to the sender.

4 Traffic Shaping

The *Traffic Shaper* shapes the outbound traffic according to the portion of available bandwidth assigned to the node. It also prioritizes real-time over non real-time traffic for transmission.

Since we want to shape the outgoing traffic at a bit level we insert the traffic shaper as close to the network card as possible. At this position all outbound packets can be caught and queues before being processed by the traffic shaping mechanism.

The traffic shaper takes the predicted bandwidth as an input parameter and adjusts it's output rate accordingly. It also ensures that a node does not use more bandwidth than its allowed share. Figure 1 shows that

the traffic shaper architecture actually contains two *Token Buckets (TB)*, one for real-time and the other for non real-time traffic. Then we prioritize real-time packet transmissions by assigning a higher rate to the real-time TB .

We implemented the traffic shaper using the QoS features built into the Linux 2.6 kernel. The shaper architecture uses the *hierarchical token bucket* (HTB) and consists of a *u32* filter and two HTBs (real-time and non real-time) as shown in the traffic shaping part of figure 1.

The u32 filter allows us to distinguish between real-time and non real-time packets and send them to the corresponding HTB. When using HTBs a maximum transmission rate is set for the whole architecture (both HTBs). Transmissions rates are also set for both of the HTBs (with the sum equal to the maximum rate). The HTB implementation in the Linux kernel allows bandwidth unused by an HTB to be lent to another HTB, allowing for effective use of the bandwidth.

5 Status and Ongoing Work

We have implemented the architecture described above and currently we focus mainly on bandwidth prediction. First results from our implementation are encouraging.

We are currently investigating how to further enhance our bandwidth prediction by looking at more network parameters to be used as input [1].

Furthermore, we are investigating stability issues and the control aspect of the stream adaptation and the fluctuations of bandwidth on wireless networks.

References

[1] F. Carone and R. Guerra. Available Bandwidth Measurement on Wireless Networks. Technical report, Departement of Computer Engineering, 2004.

[2] A. Carpenzano, R.Carponetto, L. LoBello, and O. Mirabella. Fuzzy Traffic Smoothing: An Approach for Real-Time Comunication over Ethernet Networks. In *IEEE International Workshop on Factory Communication Systems*, Västerås, Sweden, August 2002.

[3] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Conference of Communications Architecture and Protocols*, Zürich, Switzerland, September 1991.

[4] S.-K. Kweon, K. Shin, and G. Workman. Achieving Real-Time Communication over Ethernet with Adaptive Traffic Smoothing. In *IEEE Real-Time Technology and Applications Symposium*, Washington DC, USA, May-June 2000.

[5] L. Rizvanovic and G. Fohler. The MATRIX: A QoS Framework for Streaming in Heterogeneous Systems. In *RTMM - International Workshop on Real-Time for Multimedia*, Catania, Sicily, Italy, July 2004.

[6] tcpdump/libpcap, http://www.tcpdump.org/.