Pioneer in 2004 : Improving TCP performance by adding awareness of wireless link errors

Author: Simon Leung
Supervisor: Tim Moors

Abstract
In this thesis, we attempt to show the benefits of using a cross layer feedback approach to improve TCP performance in Wireless Situation. We will also implement and evaluate the performance of an algorithm that I designed. This algorithm is a cross layer feedback approach that makes use of the signal to noise ratio obtainable from the network card in order to differentiate between wireless errors and congestion.

Background
TCP
TCP performance in wireless situation is known to be poor. It is originally designed only for wired network and assumes that any loss is due to congestion. Upon any loss, the congestion avoidance algorithm is used to back off and increase the retransmission timer such that it will not overload the network. However, it is different in wireless situation in that wireless errors are more likely to occur than congestion. The inability to differentiate between wireless errors and congestion leads to the poor performance of TCP in wireless situations.

Wireless Extension
Wireless extension was developed by Jean Tourrilhes and is an extension to the current Linux networking interface. It provides a wireless API that allows users to configure and access information about wireless LAN cards in a standard and uniform way. This enable us to access information such as signal strength and noise power in any network card that provide supports for Wireless Extension.

SNR VS BER
SNR can not be directly associated with BER. It depends on the speed the network card is currently operating in. (i.e., the modulation the card is currently using. Higher modulation requires a higher SNR in order to get the same BER as the one using a lower modulation, e.g., QPSK VS BPSK)

What's new ?
My Algorithm attempts to differentiate between Wireless error and Congestion using the SNR obtained from the Network Card. It works by storing a few SNR due to consecutive TCP retransmission and doing a comparison to track the change of SNR. Assuming the network speed stays the same, if • SNR Increases, then the BER will decrease. So, if a retransmission occurs, it is more likely congestion rather than transmission error (Assuming the SNR is above a threshold required for basic transmission).

Therefore, we will make a decision of "Congestion" • SNR decreases, then the BER will increase. So, if a retransmission occurs, it is more likely transmission error and we will make a decision of "Wireless Error" •SNR fluctuates, then we use a slower back off rate. The "decision" made above will continue to be true for that particular TCP connection until either transmission is successful or the maximum retry limit is reached, which will reset the "decision".

How we did it
To implement my algorithm, I had to modify the Linux kernel. I added in a few structures in the tcp_opt structure for storing SNR, slower back off value and the decision made. I also added in a function in the TCP timer file to communicate with the network card through wireless extension and obtain the SNR from it. This function is then called each time a retransmission occurs to determine the cause of the retransmission and act accordingly.

What was hard?
The main challenge is to come up with a solution that identifies correctly between wireless errors and congestions, requires the least modifications and doesn’t affect the original semantics of TCP. Furthermore, we want to conduct experiment which reflects reality and this can not be shown in simulation tools like ns-2. This requires an understanding of how wireless errors are generated and how to control the experimental variables such that the results obtained are valid. Experimental control is a difficult task in real life. Also, this thesis requires a lot of code tracing which can sometimes be hard as some people don’t write comments for their functions.

How good is it?
Performance testing was carried out on both the Fedora Core 2 TCP implementation and my TCP implementation using the program called TTCP. We transfer 10000 TTCP packets (equivalent to 56620 TCP packets) from the Laptop to another computer through the Access Point under a certain environment. The result is shown in the following table:

<table>
<thead>
<tr>
<th></th>
<th>Average Number of Retransmission</th>
<th>Improvement over FEDORA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Congestion (Bad Channel)</td>
<td>20 – 21</td>
<td>17.02%</td>
</tr>
<tr>
<td>Congestion (Worst Channel)</td>
<td>35 – 36</td>
<td>27.51%</td>
</tr>
<tr>
<td>No Congestion (Bad Channel)</td>
<td>20 – 21</td>
<td>15.47%</td>
</tr>
<tr>
<td>Congestion (Good Channel)</td>
<td>0</td>
<td>No effect</td>
</tr>
<tr>
<td>No Congestion (Good Channel)</td>
<td>0</td>
<td>No effect</td>
</tr>
</tbody>
</table>

What is lost?
Currently, differentiating between wireless error and congestion requires 4 retransmissions. During these 4 retransmissions, a slower back off rate is being used. This might be a problem if the network is very congested. Also, the implemented version only works well (identify wireless error and congestion correctly) if the transmission speed stays the same, but a proposed version rectifies this problem.

Conclusion
My Algorithm has been successfully implemented in Linux TCP and the performance is shown to have an improvement over the Fedora Core 2 implementation. In the future, we should implement the proposed fix so that the algorithm works transmission that changes. After that, we should compare with some TCP implementations of other proposals such as ATCP and FreezTCP to see how it compares with them.