Reprinted from the

Proceedings of the Linux Symposium

July 23rd–26th, 2008
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*

C. Craig Ross, *Linux Symposium*

Review Committee

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*

Dirk Hohndel, *Intel*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat, Inc.*
Matthew Wilson, *rPath*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
Gurhan Ozen, *Red Hat, Inc.*
Eugene Teo, *Red Hat, Inc.*
Kyle McMartin, *Red Hat, Inc.*
Jake Edge, *LWN.net*
Robyn Bergeron
Dave Boutcher, *IBM*
Mats Wichmann, *Intel*

Measuring DCCP for Linux against TCP and UDP
With Wireless Mobile Devices

Leandro Melo de Sales, Hyggo Oliveira, Angelo Perkusich
Embedded Systems and Pervasive Computing Lab
{leandro,hyggo,perkusic}@embedded.ufcg.edu.br

Arnaldo Carvalho de Melo
Red Hat, Inc.
acme@redhat.com

Abstract

Multimedia applications are very popular in the Internet. The use of UDP in most of them may result in network collapse due to the lack of congestion control. To solve this problem, a promising protocol is DCCP. DCCP[1] is a new protocol to deliver multimedia congestion-controlled unreliable datagrams.

This paper presents experimental results for DCCP in the Linux kernel while competing with TCP and UDP. DCCP behaves better than UDP, while it is fair with respect to TCP. The goal in this work is to help developers choose the proper protocol to use, as well as disseminate the DCCP Linux project. It was used with four Nokia N800s, three WLAN access points, and one router to emulate congestion. Some parameters were evaluated: throughput, loss/delay, and effects of handoffs performed by mobile hosts.

1 Introduction

With the rapid growth in popularity of wireless data services and the increasing demand for wireless connectivity, Wireless Local Area Networks (WLANs) have become more widespread and are making their way into commercial and public areas. They are available in almost everywhere including business, office and home deployments. WLANs based on the IEEE 802.11 standards enjoy high popularity due to setup simplicity, increased deployment flexibility, unlicensed frequency band, low cost and connectivity with minimal infrastructure changes. Lately, the need for Real Time (RT) multimedia services over WLANs have been dramatically increased, including Voice over IP (VoIP), audio/video (AV) streaming, Internet video conference, IPTV, entertainment and gaming, and so forth.

In this scenario, companies are adopting this technology to easily connect devices and offer new mobile services. The main reasons for this growth are:

1. the improvements on the quality of wireless transmissions;

2. the provision of security mechanisms to safely transmit application data, thus increasing the number of available services;

3. users can walk and still have their devices connected—this can contribute to the new era of mobile services;

4. efficient and seamless connection to a wireless network, reducing the time for network setup and the necessity of any kind of cable;

5. the increasing number of low-cost mobile devices—allowing home users to have access to the world of wireless Internet access; and

6. everytime/everywhere computing, enabling Internet access in public spaces.

Based on this visible growth, multimedia applications receive special attention due to the popularization of high-speed residential Internet access and wireless connections, considering also new standards such as IEEE 802.16 (WiMax). This enables network applications that transmit and receive multimedia contents through the Internet to become feasible once developers and industry invest money and software development efforts in this area. They are developing specialized multimedia applications based on technologies such as Voice over IP (e.g., Skype, GoogleTalk, Gizmo), Internet Radio (e.g., SHOUTcast, Rhapsody), online games (e.g., Half Life, World of Warcraft), video conferencing, and others; these have also become popular in the context

---

[1]Datagram Congestion Control Protocol

of mobile computing. These applications offer sophisticated solutions that can approximate a face-to-face dialog for people, although they can be physically separated by hundreds or thousands of kilometers in distance.

There are at least three reasons for the growth of the popularity on the usage of mobile multimedia applications. First, the availability of new development libraries for mobile multimedia applications focusing on data processing optimizations. Second, the availability of smaller mobile devices with higher processing power and data storage capacities. And third, the necessity of people to communicate considering cost and benefits. For instance, VoIP applications can, at least, halve the original costs of a voice call when compared to traditional means.

For these types of applications, non-functional requirements such as end-to-end delay (latency) and the variation of the delay (jitter) must be taken into account. Usually multimedia applications use TCP and UDP as their transport protocol, but they may present many drawbacks regarding these non-functional requirements, and hence decrease the quality of the multimedia content being transmitted. In order to deal with those types of requirements, IETF standardized the Datagram Congestion Control Protocol (DCCP) [4], which appears as an alternative to transport congestion controlled flows of multimedia data, mainly for those applications focusing on the Internet.

In this article we present the results of an experimental evaluation using TCP, UDP, and DCCP to transmit multimedia data over a test bed 802.11g wireless network, considering wireless mobile scenarios. In these scenarios, several parameters were evaluated, such as throughput, packet loss, jitter, and the execution of hand-off. Hand-off is a process of transferring a wireless connection in progress from one access point to another without interrupting the data transmission.

The remainder of this article is organized as follows: in Section 2, an overview of some characteristics available in the DCCP protocol is presented. In Section 3, the methods used to evaluate the experiments are explained. Results of the experiments are discussed in Section 4. Finally, we present conclusions and future works in Section 5.

2 Overview and Background

DCCP [4] was first introduced by Kohler *et al.* in July, 2001, at the IETF transport group. It provides specific features designed to fulfill the gap between TCP and UDP protocols for multimedia application requirements. It provides a connection-oriented transport layer for congestion-controlled but unreliable data transmission. In addition, DCCP provides a framework that enables addition of a new congestion control mechanism, which may be used and specified during the connection handshake, or even negotiated in an already established connection. DCCP also provides a mechanism to get connection statistics, which contain useful information about packet loss, a congestion control mechanism with Explicit Congestion Notification (ECN) support, and Path Maximum Transmission Unit (PMTU) discovery.

From TCP, DCCP implements the connection-oriented and congestion-controlled features, and from UDP, DCCP provides an unreliable data transmission. The main reasons to specify a connection-oriented protocol is to facilitate the implementation of congestion control algorithms and enable firewall traversal, a UDP limitation that motivated network researchers to specify the STUN [10] (Simple Traversal of UDP through NATs (Network Address Translation)). STUN is a mechanism that helps UDP applications to work over firewalled networks. An important feature of DCCP is the modular congestion control framework. The congestion control framework was designed to allow extending the congestion control mechanism, as well as to load and unload new congestion control algorithms based on the application requirements. All of these operations can be performed before the connection setup or during an already-established connection through the feature negotiation mechanism [4]. Each congestion control algorithm has an identifier called Congestion Control Identifier (CCID).

Considering motivations to design a new protocol, one of them is the way in which TCP provides congestion control and reliable data transfer. When loss of packets occurs, TCP decreases its transmission rate and increases the transmission rate again when it successfully sends data packets. To implement a reliable data transfer, when TCP losses packets, it retransmits them. In this case, new data generated by the application is queued until all lost packets have been sent. Because

of this way of implementing reliable data transfer, using TCP may lead to a high level of flow delay. As a consequence, the user may experience interruptions in the multimedia content being transmitted. In addition, the TCP congestion control mechanism limits the transmission rate for a given connection. This means that TCP is fair with respect to other TCP flows and can be fair with other congestion controlled flows, such as those transmitted by DCCP. These characteristics of TCP make it proper for those applications that require reliable data transfers, such as web browsers, instant messengers, e-mail, file sharing, and so forth.

On the other hand, UDP is a very simple protocol working on top of the best-effort IP protocol, implementing minimal functions to transport data from one computer to another. It provides a connectionless service and it does not care about data packets' delivery, nor about network congestion control. In addition, it does not provide packet reordering on the receiver end, if taking into account the original ordering of packets transmitted by the sender. Due to the lack of any type of congestion control, UDP may lead to a network congestion collapse, where TCP-based applications may also become unusable. Hence, a UDP application can send data as much as it can, but much of that data may be lost or discarded by the routers due to network congestion. Some examples of UDP applications are VoIP applications, video-conferencing, and Internet radio.

When developing multimedia applications using TCP as the transport protocol, end users may experience high streaming delays due to high packet retransmission rates caused by network congestion. On the other hand, the use of UDP may lead to a network collapse or bad streaming quality, since UDP does not provide any kind of congestion control. The new option is DCCP, which combines the good features of each protocol to provide better quality for multimedia data streaming, as well as to share network bandwidth with TCP.

2.1 DCCP Congestion Control Identifiers

Nowadays, DCCP provides two CCIDs already standardized: the TCP-Like Congestion Control (or CCID-2) [5] and the TCP-Friendly Rate Control (or CCID-3) [6]. The goal behind this feature is to provide a way to control the flow of packets according to the type of data being transmitted. A CCID may be used at any time of a DCCP connection, and it is possible to have one CCID running in one direction, and other in the opposite direction. The flexibility on the CCID usage is important because the transmitted multimedia flow may present different characteristics. For example, a VoIP flow is characterized by a burst of small packets—when one interlocutor says something—between periods of silence—when this interlocutor stops talking and waits the other peer to talk. Another example is the Video-on-Demand traffic characteristic, which is smoothly and generally based on a Constant Bit Rate (CBR).

Thus, considering different types of multimedia applications, DCCP designers defined the congestion control framework for supporting the addition of new congestion control algorithms, as well as the deletion of them regardless of the core of the protocol. In addition to the initial standardized CCIDs, the DCCP IETF is specifying the CCID-4 [7], which is a new congestion control algorithm for DCCP to be used by applications that transmit small packets of data in a short period, such as VoIP applications.

TCP-Like Congestion Control

CCID-2 [5] is based on window flow control and resembles TCP congestion control. When a host receives a DCCP packet, it sends an ACK back to the sender. After receiving that packet, the sender adjusts the window size and the expiration time. The CCID-2 algorithm is based on the AIMD [1] algorithm for window-based flow control. Similarly TCP, the window size used in the algorithm is given as the congestion window size (`cwsize`), which is equal to the maximum number of in-transit packets allowed in the network at any time.

The sending host itself adjusts `cwsize` through congestion estimation according to the sequence of the ACK packet received. In this way, the `cwsize` is increased by one packet in the following cases:

1. every acknowledged packet arrives in a slow-start phase, and

2. every window of data is acknowledged without lost packets in a congestion-avoidance phase.

On the other hand, the `cwsize` is halved when the sender can infer that loss of packets occurs due to duplicate acknowledgments, which is equivalent to TCP.

If an ACK packet does not arrive at the sender before the timeout timer expires (i.e., when an entire window of packets is lost), the sender sets `cwsize` to one. The CCID-2 is proper for applications that want to use as much bandwidth as possible and are able to adapt to sudden changes in the available bandwidth [2, 8].

TCP-Friendly Rate Control TFRC

The CCID-3 [5] implements a receiver-based congestion control algorithm where the sender is rate-limited by packets sent by the receiver with information such as receive rate, loss intervals, and the time packets are kept in queues before being acknowledged. This CCID is intended for applications that smoothly support rate changes. Since the changes are not abrupt, it responds more slowly than TCP or TCP-like congestion controls.

The transmission rate is changed by varying the number of packets sent and is not suitable for applications that prefer variation in the sending rate by changing the packet size. In the CCID-3 implementation, the sending rate is computed by analyzing the loss event rate based on a throughput equation named TFRC Equation [5]. It supports Explicit Congestion Notification (ECN) and, to verify whether the receiver reported an accurate loss event, it also reports the ECN Nonce Sum [5] for all packets reported as received.

2.2 Summary of TCP, UDP and DCCP features

According to Table 1, which shows a comparison between the features of TCP, UDP, and DCCP, one may observe that DCCP is different from TCP in four points that are highlighted in bold. The first of them is the size of the header of each packet, which varies depending on the value of the X field presented in the header. The X field represents the Extended Sequence Number. If it is equals 0, the length of the packet is 12 bytes; if X is equal to 1, the length of the packet is 16 bytes. The second item is conceptual: while TCP sends segments, DCCP sends datagrams. The third difference between TCP and DCCP is that DCCP does not guarantee the delivery of data transmitted, except when the data transmitted is related to a feature negotiation provided by DCCP. The last difference is that DCCP does not guarantee packet reordering, even though it uses a sequence number in the packet header.

3 Methods and Experiments

In this section we describe two scenarios used to perform the experiments using the DCCP protocol, presenting the parameters and methods adopted to obtain the data for each metric collected during the experiments, such as instantaneous throughput and latency. We use a statistical method based on the probability theory [3] to calculate how many times it is necessary to repeat a given experiment to obtain an acceptable confidence level for each collected metric. In this work, it was considered 95% for the confidence level. By using this mechanism, it is possible to compare each protocol in terms of its respective performance while competing with each other.

The network topology used to execute the experiments was an 802.11g wireless network composed of both computers and Internet Tablets, in this case, Nokia N800. The experiments also examined the execution of hand-offs, where the internet tablets performed hand-offs at the link level of the 802.11g wireless network during data transmission. After explaining the general considerations adopted in the experiments, the network topology is presented.

3.1 General Considerations

The DCCP implementation used to run the experiments is available in the Linux kernel version 2.6.25, which can be obtained from the DCCP development `git` tree. Because the version of the Linux kernel for the Internet Tablets was 2.6.21, we backported the DCCP implementation from Linux kernel version 2.6.25 to version 2.6.21. Therefore, all the devices used in the experiments had the same DCCP implementation.

To generate TCP, UDP, and DCCP data flows, IPerf was used; it provides statistical reports about the connection during data transmissions. This includes statistics about packets lost and received, jitter, and throughput for a given instant. We defined the scenarios of experiments, which mean specifying values for IPerf parameters and the devices to be used in each experiment considering confronts between two given protocols (TCP × UDP, TCP × DCCP, and UDP × DCCP). Experiments with packet sizes of 512 bytes and 1424 bytes were performed. In this case, the idea was to verify whether varying the packet size would produce any impact on the protocol performance, since varying the packet size

Table 1: Comparison of TCP, UDP and DCCP features

Feature	UDP	TCP	DCCP
Packet size	8 bytes	20 bytes	**12 or 16 bytes**
Transport layer packet entity	Datagram	Segment	**Datagram**
Port numbering	Yes	Yes	Yes
Error detection	Optional	Yes	Yes
Reliability: Error recovery by ARQ	No	Yes	**No**
Sequence numbering and reordering	No	Yes	Yes/**No**
Flow control	No	Yes	Yes
Congestion Control	No	Yes	Yes
ECN support	No	Yes	Yes

during the transmission may lead to fragmentation of packets in the IP layer. If this is the case, it may affect the multimedia data quality being transmitted. Varying the packet size during the multimedia data streaming is one of the well-known techniques adopted by multimedia applications to adapt the quality of the flow in response to network congestion.

Regarding congestion control algorithms for TCP and DCCP, Reno, Cubic, and Veno were used for TCP; and for DCCP, CCID-2 and CCID-3 were used. The device used in the experiments was the Nokia N800, with an ARM 330 *MHz* processor, 128 *MB* and a Texas Instruments wireless network interface.

3.2 Network Topology

The goal was to study the performance of TCP, UDP, and DCCP when running on resource-limited devices, considering processor and memory capacities. An important point was to analyze the behavior of the protocols studied when the user application performs hand-offs between two consecutive 802.11g wireless access points. In this case, two Internet Tablets working as clients performed hand-offs, while the other two worked as servers and did not perform hand-offs. The execution time for each of the experiments was 300 *s*, where the hand-offs were performed in 100 *s* and 200 *s*.

The network topology defined for this scenario is shown in Figure 1. In this case we used three Internet Tablets to transmit two UDP or DCCP flows—each flow in one Internet Tablet—and the third one was used to transmit one TCP flow. In practice, this means two multimedia flows using UDP or DCCP (audio and video) against a data-oriented application such as HTTP.

3.3 Parameters for the Experiments

Four parameters were considered for the experiments: protocol confront; packet size; congestion control algorithm; and the existence of hand-offs. As discussed before, to transmit data in any scenario of a given experiment, the protocols were combined between them two-by-two. In Table 2, the quantity of flows transmitted during the experiments for each protocol is shown, according to each confront. The goal to define these confronts is to analyze the fairness among the three protocols in terms of network bandwidth usage.

For each scenario, the packet size was varied. For each confront of the protocols, the experiments were performed with different packet sizes, either 512 *bytes* or 1424 *bytes*. The goal for variation is to analyze whether the transmitted packet size impacts the performance of the studied protocols. As discussed before, some applications perform adaptation in the quality of the flow being transmitted as a response to the network congestion. To perform this task, codecs that support such a feature are called VBR, which varies the bit rate for each generated packet—or for a small set of continuous packets—according to the multimedia content being transmitted, which dynamically changes the packet size during data transmission.

Varying the congestion control algorithm allows the performance analysis of each congestion control algorithm during data transmission. Besides, fairness with respect to the network bandwidth usage can be evaluated. Also, by varying this parameter, is possible to study the behavior of each protocol when network congestion occurs.

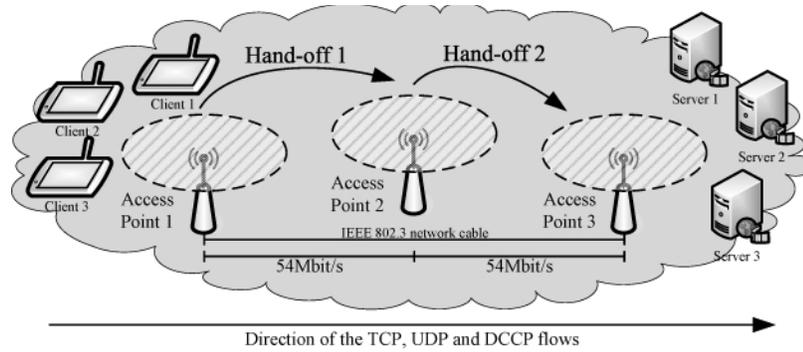The last parameter taken into account is the existence or

Figure 1: Network topology for the experiments.

#	Confronts	TCP flow	UDP flows	DCCP flows
1	**TCP × UDP**	1	2	0
2	**TCP × DCCP**	1	0	2
3	**UDP × DCCP**	0	2	1

Table 2: Number of flows used in each protocols confronts

not of hand-offs. Two of the four Internet Tablets performed hand-offs. It is known that during hand-off there are packet losses, but some congestion control algorithms mistake these losses as congestion, such as TCP Reno and the DCCP CCID-2. When a packet is lost, these algorithms assume congestion on the network and wrongly react by decreasing the allowed sending rate of the connection, but theses losses are temporary—they only occur during the hand-off. The goal in this case is to study the behavior of the congestion control algorithms in the existence of hand-off.

3.4 Collected Metrics and Derived Metrics

For all executed experiments, a TCP flow was first transmitted, and after 20 *s*, the other flows were started. By executing the experiments in this way, it was possible to evaluate how fair the protocols are with each other when new flows are introduced in the network. This enables analyzing whether any of them can impact the performance of the other—mainly whether DCCP and UDP impact the performance of TCP. In addition, the idea is to look for the most suitable TCP and DCCP congestion control algorithms to transmit multimedia data over the network.

To reach all of these goals, a set of metric values was collected for the flows transmitted during the experiments. The metrics were the throughput, packet loss, and latency. Considering these metrics, it is possible to obtain other two metrics: jitter and the rate of how many packets reached the receiver, which can be obtained from the quantity of transmitted packets. Through latency, it is possible to calculate jitter for a given instant; from throughput and the quantity of lost packets, it is possible to obtain the effective amount of data transmitted—how much data effectively reached the receiver.

3.5 Obtaining Throughput, Jitter and the Amount of Data Lost and Transmitted

The mean throughput and the amount of data transmitted for TCP was obtained through the average of the means in each repetition *r* of a given experiment. This is shown in Equations 1 and 2, where *n* is the total of repetitions.

$$\mu_{thoughput\_tcp} = \frac{\sum_{r=1}^{n} throughput\_mean_r}{n} \quad (1)$$

$$\mu_{load\_tcp} = \frac{\sum_{r=1}^{n} load\_mean_r}{n} \quad (2)$$

However, to obtain the means for the UDP and DCCP throughput and amount of data transmitted, the procedure was different. Considering that two UDP/DCCP flows have been transmitted—taken regardless—against a TCP flow, and considering also that the UDP/DCCP

flows started only $20\,s$ after the TCP flow started, it was necessary to define a mechanism that does not penalize both protocols in a confront. In this case the calculation of the means would not be an arithmetic average of the sum of the throughput and the amount of data transmitted by the two UDP/DCCP flows. Instead, it should be the mean throughput and the mean amount of data transmitted of each flow. This observation is represented in Equation 3, where each *thoughput_mean$_r$* of this equation can be obtained from Equation 4.

$$\mu_{partial\_throughput(udp/dccp)} = \frac{\sum_{r=1}^{n} thoughput\_mean_r}{n} \quad (3)$$

$$thoughput\_mean_r = \frac{\sum_{k=1}^{F} thoughput\_mean\_flow_k}{F} \quad (4)$$

Based on the same assumptions presented before, in Equation 4, the term *thoughput_mean_flow$_k$* is obtained through the arithmetic means of the throughput in each instant (per second) of the experiment. Therefore, the final value for the throughput for a given transmitted flow (connection) of UDP and DCCP can be obtained from Equation 5.

$$\mu_{final\_throughput(udp/dccp)} = \mu_{partial\_throughput(udp/dccp)} + S \times \left(\frac{\mu_{partial\_throughput(udp/dccp)}}{T}\right) \quad (5)$$

where F is the number of flows, $F_{UDP} = F_{DCCP} = 2$ for TCP $\times$ UDP/DCCP and $F_{DCCP} = 1$ for UDP $\times$ DCCP; S, is the await time to start the UDP or DCCP flows ($S = 20\,s$); and T, is the total time of the experiments ($T = 100\,s$ without hand-offs or $T = 300\,s$ with hand-off).

The means were normalized according to Equation 5, to avoid penalizing the protocols in the terms discussed before.

In a similar way the latency and effective amount of data transmitted can be obtained. Note that for UDP $\times$ DCCP confronts, the term F_{DCCP} is equal to 1. In these cases the throughput and amount of data transmitted are obtained through Equations 1 and 2, respectively.

Jitter

The calculation to obtain the mean *jitter* for a transmitted flow is very similar to the calculation of the mean throughput. The value for the jitter can be obtained through Equation 8 and it can be obtained as follows:

$$\mu_{partial\_jitter(udp/dccp)} = \frac{\sum_{r=1}^{n} jitter\_mean_r}{n} \quad (6)$$

and,

$$jitter\_mean_r = \frac{\sum_{k=1}^{F} \left(\frac{\sum_{k=1}^{QI} VA_k}{QI}\right)}{F} \quad (7)$$

then,

$$\mu_{final\_jitter(udp/dccp)} = \mu_{partial\_jitter(udp/dccp)} + S \times \left(\frac{\mu_{partial\_jitter(udp/dccp)}}{T}\right) \quad (8)$$

where: F is the number of flows used in the experiments, $F_{UDP} = F_{DCCP} = 2$ for TCP $\times$ UDP/DCCP and $F_{DCCP} = 1$ for UDP $\times$ DCCP, QI, is the quantity of intervals ($QI = T - 1$) for two consecutives read of collected data, VA, is the variation of the delay between packets of the same flow, for instance $time_1 = 10\,ms$ and $time_2 = 11\,ms$, $VA = 1\,ms$, T, is the total time of the experiments ($T = 100\,s$ without hand-off or $T = 300\,s$ with hand-off).

3.6 Statistic Methodology for the Final Calculation of the Collected Metrics

The results presented in this work—for instance, to determine what protocol performed better than the other in terms of bandwidth usage—were based on samples of data collected while performing the experiments. The methodology adopted was based on the concepts of confidence interval [3], considering $\rho = 95\,\%$ (confidence level) and therefore $\alpha = 5\,\%$ (significance level, or error).

Determining the Confidence Interval for $\rho = 95\%$

The principle for the confidence interval is based on the fact that it is impossible to determine a perfect mean μ for a infinite population of N samples, considering a finite number n of samples $\{x_1, ..., x_n\}$. However, it is possible to determine in a probabilistic way an interval where μ will belong to this interval, with probability equals to ρ, and that will be not in this interval with probability of α.

To determine the minimum value c_1 and the maximum value c_2 for this interval, called a confidence interval, it is considered the probability $1 - \alpha$, where the μ value will belong to this interval, for n repetitions of a certain executed experiment. The Equation 9 summarizes this consideration.

$$Probability\{c_1 \leq \mu \leq c_2\} = 1 - \alpha \qquad (9)$$

where (c_1, c_2) is the confidence interval; α is the significance level, expressed by a fraction and typically close to zero, for instance, 0.05 or 0.1; $(1 - \alpha)$ coefficient of confidence; and $\rho = 100 * (1 - \alpha)$, is the confidence level, traditionally expressed in percent and closer to 100 %; this work uses 95 %.

From the Central Limit Theorem[2] [3], if a set of samples $\{x_1, ..., x_n\}$ is independent, has a mean $\bar{x}$, and belongs to the same population N, with mean μ and standard deviation σ, then the average of the samples is in a normal distribution with $\bar{x} = \mu$ and standard deviation $\sigma/\sqrt{n}, \bar{x} \simeq N(\mu, \frac{\sigma}{\sqrt{n}})$.

Considering Relation 9 and the Central Limit Theorem, the confidence interval (c_1, c_2) for $\rho = 95\%$ and $\alpha = 0.05$ can be obtained as shown in Equation 10.

$$\left(\mu - z_{1-\alpha/2} \times \frac{s}{\sqrt{n}}, \mu + z_{1-\alpha/2} \times \frac{s}{\sqrt{n}}\right) \qquad (10)$$

where μ is the average for n repetition; $z_{1-\alpha/2}$ is equal to 1.96, this value determines 95 % of confidence level; n is equal to the number of repetitions; and s is the standard deviation of the means for n repetitions.

_____

[2]Central Limit Theorem: the sum of a large number of independent and identically-distributed random variables will be approximately normally distributed if the random variables have a finite variance.

Regarding the value for $z_{1-\alpha/2}$, also named quantile, is based on the Central Limit Theorem and since it is frequently used, it can be found in a table named *Quantile Unit of the Normal Distribution*. This table can be found in the reference [3], Table A.2 of Appendix A. Using the Relation 11, next it is explained how to determine the value 1.96 for the term $z_{1-\alpha/2}$.

$$z_{1-\alpha/2} = (1 - 0.05)/2 = 0.975 \qquad (11)$$

According to the table *Quantile Unit of the Normal Distribution* available in reference [3], the corresponding value for the result of the Equation 11 is 1.96, which is the value to be used as the variable z of the the Equation 10.

Therefore, based on the confidence interval of each average for each metric collected during the experiments (see Section 3.5), it is possible to perform comparisons with these values for the defined scenarios of experiments for 95 % of confidence with 5 % of error.

Determining the Value for n to obtain $\rho = 95\%$

The confidence level depends on the quantity of samples n collected for a certain metric of a given experiment. Thus, the higher the value of n is, the more precise the confidence level will be. However, to obtain big samples requires more effort and time. Therefore, it is important to define a value for n and avoid repeating a specific experiment unnecessarily, but maintaining the desired confidence level $\rho = 95\%$.

To start the process of the experiment performed in this work, each experiment was repeated 3 times ($n_{base} = 3$). For example, the initial throughput mean of a given trasmitted flow was obtained from the means obtained by running the experiment 3 times. This means that firstly we obtain a high value for the variance, which is used to determine the real value for n to obtain 95 % of confidence level.

Based on Equation 10, the confidence interval for a given value of n samples is defined by Equation 12.

$$\mu \pm z \times \frac{s}{\sqrt{n}} \qquad (12)$$

Thus, for the confidence level of $\rho = 95\,\%$ and $\alpha = 0.05$, the confidence interval is determined by Equation 13.

$$(\mu(1-0.05), \mu(1+0.05)) \tag{13}$$

Then, equating the confidence interval specified in Expression 13 with the confidence interval specified in Expression 12 (general), Equation 14 is obtained.

$$\mu \pm z \times \frac{s}{\sqrt{n}} = \mu(1 \pm 0.05) \tag{14}$$

Therefore, organizing the expression by isolating the variable n, each experiment was repeated n times determined in Equation 15, considering a confidence level $\rho = 95\,\%$, which implies in $z = 1.96$ (from Equation 11), and the 3 initial times of experiment repetition (n_{base}). For example, if the value for n is 12 for a given experiment, it was repeated $n = n - n_{base}$, which is equal to 9, and the three first means is also considered for the value of the final average of a given metric.

$$n = (\frac{1.96 \times s}{0.05 \times \mu})^2 \tag{15}$$

4 Results

Using the definitions and methods presented in Section 3, in this section the results and discussions about the experiments are presented according to methods discussed in Section 3.

The results are organized in two tables considering the packet size used in the transmission. The results for the evaluated metrics for transmissions using packets of size 512 bytes are presented in Table 3. The results for transmissions using packets of size 1424 bytes are presented in Table 4.

The values presented in these tables have a 95% confidence level with a 5% margin of error. The confidence interval is presented immediately below the value for the corresponding metric. For the UDP and DCCP protocols the confidence interval for the metric Transmitted / Lost corresponds to the effective load of transmitted data, that is, the subtraction of Transmitted−Lost. Also, consider that the values presented in the two tables correspond to the execution of the experiments following the process described in Section 3.1, considering that: the execution time is $300\,s$, the instants that

the hand-off was performed were at $100\,s$ and $200\,s$, the confront among protocols were TCP × UDP, TCP × DCCP, and UDP × DCCP. Also, the congestion control algorithms for TCP were: Reno, Cubic and Veno, and for DCCP: CCID-2 and CCID-3. The metrics analyzed were throughput, the amount of transmitted and lost data, and latency/jitter.

4.1 Discussions about the Experiments

The major point considered in the experiments is related to:

1. the impact of changing the data packet size on the performance of the protocol during transmissions;

2. the impact caused in terms of throughput and the amount of data transmitted and lost when performing hand-off during data transmission; and

3. the behavior of TCP, UDP, and DCCP in terms of fairness.

For the first item, there were no considerable changes in the behavior of the transmitted flow for all protocols taken regardless, mainly related to the metrics throughput and jitter. But it is possible to observe changes in the performance of TCP Reno, Cubic, and Veno algorithms for TCP × UDP and TCP × DCCP.

For the TCP × UDP test, the amount of transmitted data using the algorithms TCP Reno, Cubic, and Veno were not satisfactory if the result of the experiments is divided into two groups: one with experiments using a packet size of 512 bytes (Table 3) and another with experiments using packet size of 1424 bytes (Table 4). If the throughput of the TCP and UDP protocols is almost the same for the two groups of experiments (see lines 1, 2, and 3), we expect to observe that the bigger packet size is, bigger the amount of transmitted data should be, considering that there is no packet fragmentation in the network layer, since the MTU for the 802.11g connection is 1500 bytes.

On the other hand, there are no changes for TCP × DCCP, regardless of the algorithm used. Comparing the throughput values for the TCP × UDP and TCP × DCCP confronts presented in Tables 3 and 4, a balance among these values can be observed. For instance, in Tables 3 and 4 the mean throughput of TCP Reno

#	Confronts	Throughput (Kbits/s)	Transmitted / Lost (KBytes)	*Jitter* (ms)	n
1	**TCP Reno** × **UDP**	3359,12 (3202,15 − 3516,09)	123006,72 (117258,69 − 128754,75)	2,11 (1,72 − 2,21)	11
		2956,09 (2935,81 − 2976,36)	394825,25/293967,67 (100164,37 − 101550,79)	1,60 (1,57 − 1,64)	
2	**TCP Cubic** × **UDP**	3364,12 (3232,15 − 3496,09)	121855,28 (118258,44 − 125452,12)	7,51 (6,32 − 8,7)	5
		2919,76 (2893,96 − 2945,57)	419803,17/320187 (98735,37 − 100496,97)	1,71 (1,69 − 1,74)	
3	**TCP Veno** × **UDP**	3121,69 (3042,26 − 3201,13)	114322,77 (111412,94 − 117232,60)	4,2 (3,4 − 5,0)	7
		3002,75 (2994,15 − 3011,36)	378833,83/276384,92 (102150,85 − 102746,97)	1,5 (1,50 − 1,54)	
4	**TCP Reno** × **DCCP-2**	3042,90 (2951,57 − 3134,23)	111433,62 (108090,21 − 114777,03)	4,8 (4,36 − 5,24)	9
		2162,51 (2138,56 − 2186,47)	73688,67/287,51 (73401,16 − 74234,14)	5,4 (5,02 − 5,75)	
5	**TCP Cubic** × **DCCP-2**	3862,58 (3775,82 − 3949,34)	104830,49 (101653,65 − 108007,32)	3,32 (3,11 − 3,53)	9
		2119,10 (2109,87 − 2128,33)	72256,17/367,92 (71888,25 − 72215,14)	4,2 (4,02 − 4,48)	
6	**TCP Veno** × **DCCP-2**	2395,97 (2289,27 − 2502,67)	87744,27 (83836,15 − 91652,39)	7,81 (7,33 − 8,29)	20
		2899,25 (2810,27 − 2988,24)	64653,08/314,42 (61289,36 − 67387,96)	6,1 (5,50 − 6,64)	
7	**TCP Reno** × **DCCP-3**	3291,67 (3265,40 − 3317,94)	120549,10 (119587,17 − 121511,03)	3,6 (3,42 − 3,78)	6
		2851,59 (2841,67 − 2861,52)	97234/1181,92 (95725,77 − 96378,39)	0,85 (0,83 − 0,87)	
8	**TCP Cubic** × **DCCP-3**	3598,81 (3496,79 − 3700,84)	131790,21 (128052,30 − 135528,13)	4,13 (3,77 − 4,49)	8
		2665,55 (2571,62 − 2759,48)	90895/1533,67 (86127,94 − 92594,72)	1,18 (0,95 − 1,41)	
9	**TCP Veno** × **DCCP-3**	3734,55 (3634,92 − 3834,18)	136765,27 (133115,41 − 140415,13)	2,31 (2,15 − 2,47)	11
		2824,84 (2815,48 − 2834,20)	96381,08/1472,58 (92753,94 − 97063,06)	0,89 (0,85 − 0,93)	
10	**DCCP-2** × **UDP**	1792,20 (1749,55 − 1834,86)	65194,33/303,33 (62404,79 − 64891,03)	4,91 (4,75 − 5,08)	11
		2552,41 (2475,76 − 2629,06)	562465,08/475381 (84469,96 − 89698,20)	2,02 (1,87 − 2,18)	
11	**DCCP-3** × **UDP**	2519,84 (2461,98 − 2577,70)	91559,83/1696,83 (85041,99 − 94684,01)	1,01 (0,91 − 1,12)	13
		2898,34 (2841,75 − 2954,93)	427356,58/328470,17 (96902,29 − 100870,53)	1,82 (1,73 − 1,92)	

Table 3: Summary for the results of phase 1 for $\rho = 95\%$. Packet of size 512 *bytes*, execution of hand-off and considering the confronts between two protocols among the protocols TCP, UDP and DCCP.

was 3359.12 Kbits/s and 3162.41 Kbits/s, respectively. Moreover, if the throughput is almost the same, the bigger the packet size is, more data the flow should transmit. This happened only for the TCP × DCCP. In this case, TCP transmitted more data when the packet size was 1424 bytes for all the congestion control algorithms used. This was expected because DCCP also implements congestion control and it allows TCP flow

to transmit more data when increasing the packet size, considering that the maximum size for a packet is the MTU value minus the space occupied by the headers of protocols in the network, transport, and application layers.

Therefore, it is possible to conclude that in transmissions where the TCP and UDP protocols share the same communication channel, to increase the packet size

#	Confronts	Throughput (Kbits/s)	Transmitted / Lost (KBytes)	*Jitter* (ms)	n
1	**TCP Reno**	3162,41 (2968,61 − 3156,21)	112156,65 (108719,62 − 115593,67)	3,21 (3,09 − 3,33)	4
	× **UDP**	5773,26 (5493,16 − 6053,36)	730078,67/659256,67 (67701,48 − 73942,52)	2,37 (2,28 − 2,47)	
2	**TCP Cubic**	3201,33 (3063,57 − 3339,09)	80614,73 (75572,17 − 85657,29)	4,51 (4,42 − 4,60)	6
	× **UDP**	2575,34 (2544,25 − 2606,43)	1213656,83/1182062,33 (27053,38 − 36135,62)	3,17 (3,05 − 3,29)	
3	**TCP Veno**	3221,97 (3072,96 − 3370,99)	117998,49 (112542,56 − 123454,43)	5,12 (5,04 − 5,2)	8
	× **UDP**	2301,01 (2288,97 − 2313,06)	682801,67/605503,92 (73433,54 − 81161,96)	2,27 (2,20 − 2,35)	
4	**TCP Reno**	3776,63 (3717,94 − 3835,31)	166004,14 (163776,65 − 168231,63)	6,3 (6,21 − 6,34)	4
	× **DCCP-2**	3372,64 (3217,38 − 3527,90)	141343,33/375,92 (139722,68 − 142963,98)	6,68 (5,40 − 7,97)	
5	**TCP Cubic**	3969,39 (3901,75 − 4037,04)	172124,00 (169648,29 − 174599,72)	4,12 (4,03 − 4,21)	3
	× **DCCP-2**	3611,59 (3578,69 − 3644,49)	187001,58/1303,25 (182247,13 − 189149,53)	5,16 (4,90 − 5,43)	
6	**TCP Veno**	4561,88 (4218,12 − 4905,64)	180149,85 (178025,55 − 182274,15)	6,51 (5,81 − 7,21)	3
	× **DCCP-2**	2685,21 (2442,01 − 2928,41)	13600,533/1150 (133295,41 − 136415,25)	6,25 (5,47 − 7,02)	
7	**TCP Reno**	2735,82 (2576,05 − 2895,60)	100189,53 (94338,37 − 106040,69)	3,71 (3,28 − 4,14)	7
	× **DCCP-3**	3469,30 (3299,87 − 3638,73)	118268,52/5685,50 (111483,27 − 113682,77)	2,80 (2,56 − 3,05)	
8	**TCP Cubic**	2980,47 (2950,09 − 3010,85)	109147,54 (108034,15 − 110260,92)	4,1 (3,49 − 4,71)	5
	× **DCCP-3**	3482,36 (3319,60 − 3645,13)	118835,16/1549,83 (112236,44 − 122334,22)	2,63 (2,34 − 2,92)	
9	**TCP Veno**	2998,39 (2867,97 − 3128,80)	109806,54 (105029,84 − 114583,24)	2,33 (2,21 − 2,45)	6
	× **DCCP-3**	4831,47 (4576,48 − 5086,45)	184765,16/4835,08 (175018,62 − 180930,08)	1,69 (1,65 − 1,73)	
10	**DCCP-2**	3452,93 (3315,24 − 3590,62)	125611,37/482,50 (122601,76 − 127655,98)	7,81 (7,55 − 8,07)	11
	× **UDP**	5236,62 (4892,45 − 5580,79)	806485,67/742278,17 (62959,4 − 65455,6)	4,30 (3,70 − 4,90)	
11	**DCCP-3**	4053,65 (3904,76 − 4202,54)	147460,73/3415 (142043,27 − 146048,19)	1,84 (1,63 − 2,04)	13
	× **UDP**	5935,79 (5884,42 − 5987,15)	699595,58/626779,58 (71211 − 74421)	2,46 (2,42 − 2,51)	

Table 4: Summary for the results of phase 1 for $\rho = 95\%$. Packet of size 1424 *bytes*, execution of hand-off and considering the confronts between two protocols among the protocols TCP, UDP and DCCP.

from 512 bytes to 1424 bytes does not lead to performance improvement, even when considering the Veno congestion control algorithm. This suggests that TCP lost more data when packets with size 1424 bytes (without considering the packet retransmission mechanism implemented by TCP to provide reliability) were used. In this case, a future work can evaluate what the best packet size should be to minimize the amount of TCP

packets lost in this scenario. A discussion in this context is presented in [12].

Regarding the hand-off executions, in Figure 2 the progression of the transmission for TCP Cubic × UDP is depicted, which corresponds to line 2 of Table 3. In Figure 2(a), the mean throughput for the protocols TCP and UDP are presented. In Figure 2(b), the relation between the amount of transmitted and lost data for UDP in this

transmission is shown. The values for each point plotted in the graph shown in Figure 2 were calculated as an average for the values of each point for all repetitions of the experiment, in this case $n = 5$ (last column of the line 2 of Table 3).

It is important to observe that in Figure 2(a) the throughput for the TCP connection decreased due to hand-off, where packet loss occurred; that is reflected in the congestion control algorithms of TCP and DCCP. In the case of UDP, the throughput remained constant during all the transmission time. In Figure 2(b), it is possible to observe a high level of data loss when using UDP, mainly during hand-off. Around second 35, it is possible to observe another declining point for the TCP throughput. This fact can be explained by the introduction of the two UDP flows, where packet loss also happened.

In Figure 3, the transmission for TCP-Cubic $\times$ DCCP is presented, which corresponds to line 5 of Table 4. The values for each point were calculated in a similar way of previous ones, with $n = 9$. This procedure was also used for the other graphs in this section.

As it can be seen in Figure 3(a), the TCP throughput also decreased due to the hand-off, for the same reasons as in TCP $\times$ UDP. In the case of the DCCP protocol, there was a small drop in the throughput during the hand-off. Figure 3(b) shows the evolution for TCP $\times$ DCCP, and it is possible to observe that DCCP lost a small amount of data when compared to UDP in the confronts against to TCP. In addition, it can be seen that during the transmission, the DCCP and TCP protocols shared the channel in a fair way.

Regarding the fairness of the protocols in terms of network bandwidth usage, a congestion in the network caused by UDP was expected, but this did not happen. Therefore, it is possible to conclude that there is data contention in the source, in this case the Internet Tablets (N800 devices). As the processing power of such devices is limited, there is a throughput limitation of data processing and transmission, considering that the process (at the operating system level) of the IPerf application used less CPU clocks compared to a desktop computer, for instance. In Section 5 a discussion on this subject is presented, where a different behavior is observed: the wireless network presented a high level of congestion caused by the UDP protocol and in some cases avoiding TCP and DCCP protocols to transmit data.

It is also important to comment on two additional facts observed in the experiments.

- Sudden wireless disconnections of the Internet Tablet were observed. This is probably associated with the processing and management capacity of the applications executing in this device, particularly the wireless interface driver running in the device. In order to have a more elaborate explanation of this, a deeper study is suggested; the focus should be to analyze situations where the processor is overloaded, leading to a malfunction of the wireless interface driver. In addition, the study should examine whether the disconnections were caused by hand-offs;

- In addition to the weak performance of the UDP protocol for wireless data transmission in terms of packet loss, in all transmissions using the UDP protocol, it was observed that the protocol delivered out-of-order packets. The out-of-order data delivery also occurred with the DCCP protocol, but in smaller proportion compared to UDP. This proportion is equivalent to the packet loss with the DCCP protocol in the TCP $\times$ DCCP confronts—Tables 3 and 4, field Transmitted/Lost, lines 4 to 9 (inclusive).

5 Conclusion

In this article, an experimental evaluation of the DCCP protocol over a 802.11g testbed wireless network is presented. We presented an overview of DCCP, an explanation of how the experiments were performed, and explored the methods adopted to calculate each metric studied in this work. An important issue in this case is the use of a statistical method based on probability theory to achieve a 95% confidence level in all the values of the studied metrics. The results obtained by executing the experiments are presented. The experiments used only resource-limited devices.

Considering the results discussed in Section 4, some conclusions can be presented. First, the UDP protocol when used in resource limited devices is not capable of generating high network traffic resulting in congestion. This is due to data generation contention on the Internet Tablets.
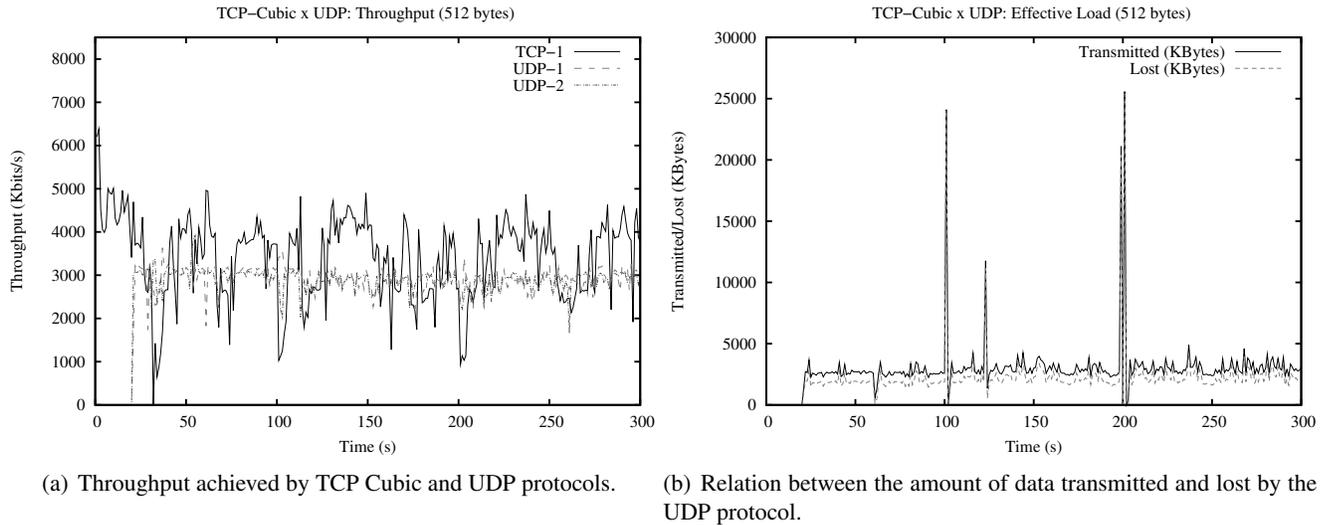
(a) Throughput achieved by TCP Cubic and UDP protocols.

(b) Relation between the amount of data transmitted and lost by the UDP protocol.

Figure 2: TCP × UDP: thoughput and effective amount of data for TCP-Cubic × UDP with 300 *s* of transmission, with packet size of 512 *bytes* and execution of *hand-offs* in 100 *s* and in 200 *s*.
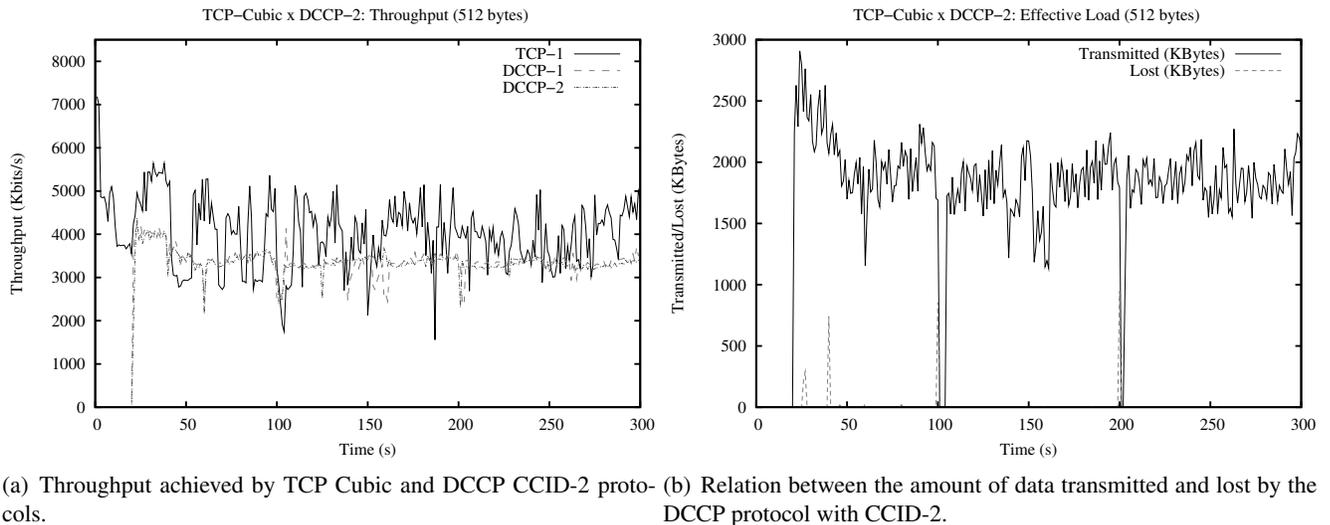

(a) Throughput achieved by TCP Cubic and DCCP CCID-2 proto-cols.

(b) Relation between the amount of data transmitted and lost by the DCCP protocol with CCID-2.

Figure 3: TCP × UDP: thoughput and effective amount of data of the TCP-Cubic × DCCP CCID-2 with 300 *s* of transmission, with packet size of 512 *bytes* and execution of *hand-offs* in 100 *s* and in 200 *s*.

It is important to point out that even though the UDP protocol was unable to cause network congestion, its use to transmit multimedia flows is not recommended, at least in the network topology used in this work. This recommendation is based on the observations that UDP lost a lot of packets when compared to DCCP, mostly in a network congestion period. This directly reflects in the multimedia quality being transmitted. In addition to the high level of packet loss, UDP interferes with the performance of other protocols that implement network congestion control, such as TCP and DCCP.

Unlike the discussion in [11], the hand-off execution during data transmission did not affect either the TCP or DCCP congestion control algorithms. In the results presented in this previous work, laptops were used, rather than resource-limited devices. There are two hypotheses to explain the non-effect of hand-offs using resource-limited devices: first, by using devices such as the N800, the hand-off occurs very fast and hence few packets are lost, if compared with hand-offs performed using computers (such as laptops) and considering that they are not manufactured with this type of service in mind, unlike the mobile devices. Thus, since the amount of the packet loss is small and considering that resource-limited devices are not capable of generating a big set of data in a short period (due to the short slice of time allocated to each application by the operating system), the small amount of data lost does not affect the congestion control algorithms. The second hypothesis completes the first one. Since N800-like devices are manufactured to work in wireless networks, the network driver is optimized for hand-off executions, unlike those available for the network interface of the laptops. For this point, it is necessary to conduct a more specific study to provide a more accurate conclusion.

Another important conclusion is that when the packet size for TCP was varied in data transmission against UDP flows, the results were not satisfactory. As observed in the results presented in Section 4, there is not a significant improvement in the amount of data transmitted when using 1424-byte (rather than 512-byte) packets. But for TCP × DCCP confronts, one may conclude that if the packet size is increased from 512 *bytes* to 1424 *bytes*, it is possible to improve the performance of both TCP and DCCP. Therefore, this procedure is encouraged. Although it was possible to observe this, it is necessary to run more experiments with packet size other than 1424 *bytes* and 512 *bytes*.

The current congestion control algorithms for DCCP performed worse than TCP when used to compete against UDP flows (except in the TCP Reno × UDP, where DCCP performed better than TCP in the DCCP CCID-3 × UDP), although DCCP seems to reach one of its goals: to be fair in respect to TCP. For this case DCCP performed very well, properly sharing the network bandwidth with TCP.

Supposing that the other part of the wireless and Internet traffic is TCP Veno, or TCP Cubic, the default congestion control algorithm for Linux, the UDP protocol must be fair in respect to TCP, since TCP Cubic and Veno performed very well in terms of the available network bandwidth. Until the end of this work, no references were found that explored possible congestion control algorithms for UDP, nor official comparative studies between TCP Cubic/Veno against UDP, since our work focused in the DCCP point of view. According to the results presented in this work, it is not recommended to use TCP Reno for data transmission mainly over wireless links and in the Internet. Moreover, based on the results presented in this work, for TCP transmissions it is recommended to use of TCP Cubic than TCP Veno, even though the official documentation for TCP Veno indicates that its main focus is on wireless networks. It is necessary to analyze the congestion control algorithms for DCCP in order to optimize them or provide new congestion control algorithms for it, equivalent to TCP Cubic and TCP Veno, preferentially.

The current work we are developing is a mVoIP application based on DCCP for mobile devices, focusing on the maemo™ platform [9].

6   Additional Authors and Acknowledgments

References

[1] J. Gao and N. S. V. Rao. TCP AIMD Dynamics over Internet Connections. In *IEEE Communication Latter*, pages 4–6, 1 2005.

[2] X. Gu, P. Di, and L. Wolf. Performance Evaluation of DCCP: A Focus on Smoothness and TCP-friendliness. In *Annals of Telecommunications Journal, Special Issue on Transport Protocols for Next Generation Networks*, volume 1, pages 191–206, 1 2005.

[3] Raj Jan. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, Inc, 1 edition, 3 1991.

[4] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP), 3 2006. http://www.ietf.org/rfc/rfc4340.txt. Last access on June 2008.

[5] Eddie Kohler, Mark Handley, and Sally Floyd. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control, 3 2006. http://www.ietf.org/rfc/rfc4341.txt. Last access on June 2008.

[6] Eddie Kohler, Mark Handley, and Sally Floyd. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC), 3 2006. http://www.ietf.org/rfc/rfc4342.txt. Last access on June 2008.

[7] Eddie Kohler, Mark Handley, and Sally Floyd. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 4: TCP-Friendly Rate Control for Small Packets, 6 2007. http://tools.ietf.org/wg/dccp/draft-ietf-dccp-ccid4. Last access on June 2008.

[8] P. Navaratnam, N. Akhtar, and R. Tafazolli. On the Performance of DCCP in Wireless Mesh Networks. In *Proceedings of the international workshop on Mobility management and wireless access*, volume 1, pages 144–147, 3 2006.

[9] Nokia Corporation. Maemo Platform, 2 2008. http://www.maemo.org/. Last access on June 2008.

[10] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs), 3 2003.

http://www.ietf.org/rfc/rfc3489.txt. Last access on June 2008.

[11] Leandro M. Sales, Hyggo O. Almeida, Angelo Perkusich, and Marcello Sales Jr. On the Performance of TCP, UDP and DCCP over 802.11g Networks. In *In Proceedings of the SAC 2008 23rd ACM Symposium on Applied Computing Fortaleza, CE*, pages 2074–2080, 1 2008.

[12] D. Wu, Song Ci, H. Sharif, and Yang Yang. Packet Size Optimization for Goodput Enhancement of Multi-Rate Wireless Networks. In *Consumer Communications and Networking Conference Proceedings*, pages 3575–3580, 3 2007.