Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Telecommunications and Media Informatics

Technical Report

Research on Transport Protocol Without Congestion Control

Sándor Molnár

June, 2011

1 Summary

The research history of congestion control protocols unveiled that it is impossible to find an optimal protocol that meets all the challenges of the evolving Internet. In this technical report we advocate the idea of omitting congestion control (TCP) and propose a networking scenario where efficient erasure coding scheme plays the role of correcting packet loss and where fair queuing provides fair bandwidth sharing. Further, we address some protocol specific questions and we give a performance study of the solution and compare it with TCP-based scenarios. Our results demonstrate that it is possible to build a Future Internet without congestion control resulting in a solution which surpasses our current TCP-controlled Internet in terms of performance.

2 Introduction

Congestion control is one of the most important control mechanisms in today's Internet, which tries to avoid low utilization of network resources in cases when links are overloaded by transmitting end hosts. From the first efficient reaction to the phenomenon of congestion collapse, to its solutions to date, congestion control algorithms - mostly performed by the Transmission Control Protocol (TCP) - has been widely varied in order to fit the requirements in ever-changing network environments. From backbone to wireless networks, different conditions demand changes and improvements on the TCP, resulting in a high number of transport protocol variants. Moreover, the new challenges of next-generation networks (e.g., high speed communication or communication over different medias) have generated a fervent need to further develop the TCP. In recent years, several new proposals and modifications of the standard congestion control mechanism have been developed by different research groups all over the world, for example, HighSpeed TCP [1], Scalable TCP [2], BIC TCP [3], CUBIC TCP [4], FAST TCP [5], TCP Westwood [6], Compound TCP [7] and XCP [8], just to name a few of them. These new mechanisms address different aspects of future networks and applications, and improve the performance of the regular TCP. Concerning fairness performance as one of the most important features of congestion control protocols they can achieve very different results [9]. In spite of the fact that these new proposals were able to provide efficient solutions for some areas, they all fail to act as a *universal* mechanism in heterogeneous and changing network environments. It seems that neither the fundamental idea of

TCP nor its improved variants are capable of achieving a universally good solution in congestion control. This conclusion is supported by the fact that the development of new applications shows that they use their own congestion control mechanisms. However, in most of the cases these mechanisms are not TCP friendly so they cannot work together with the TCP efficiently [10].

Some ideas have been proposed and investigated where congestion control was not employed at all. One of these ideas was proposed by GENI (Global Environment for Network Innovations) advocating a *Future Internet without congestion control* [11]. The surprising idea is that every end host is allowed to send its data at maximal rate so that it can send as fast as it can whenever it has data to send. Naturally, if end point access rates do not cause congestion, then this idea yields the most efficient solution. However, when hosts send data at their maximal rates, excessive packet loss is likely to occur, mostly in a bursty manner. GENI proposed to use efficient erasure coding in order to deal with packet loss. This concept has several attractive advantages, like efficiency - as every network resource would always be fully utilized and any additional capacity would immediately be consumed, secondly simplicity - as relying on an efficient erasure coding could make packet loss inconsequential resulting in routers with highly reduced buffer sizes, and finally stability - as using maximal sending rates would result in predictable traffic patterns instead of a high extent of rate variation seen in TCP transmissions. These projected benefits especially favor optical networking, pointing toward all-optical networks where small buffer sizes would be highly required considering the difficulty of optical buffering via optical delays [12].

The idea of omitting congestion control is promising, but comprehensive research is needed to validate the concept, to work out implementation details and to show that the idea can be applied efficiently in practice. In the following, we review some of the related work. Raghavan and Snoeren investigated the benefits of a decongestion controlled network and proposed a possible solution based on a decongestion controller [13]. Bonald et al. investigated the behavior of networks in the absence of congestion control [14]. Their astounding result is that operating a network without congestion control does not lead to a general congestion collapse which is the common belief. López et al. applied a fountain based protocol and investigated the achieved performance by game theory [15]. They show that a Nash equilibrium can be obtained and that at this equilibrium the performance of the network is similar to the performance obtained when all hosts comply with TCP. Rateless codes seem to be good candidates to use as erasure coding scheme. For example, for a live unicast video streaming scenario a solution based on rateless codes

with experimental results is proposed in [16].

This technical report presents a scenario for Future Internet where congestion control is avoided. We propose a solution where packet loss is corrected by efficient rateless codes and fair queuing is used to provide faireness among flows. Moreover, a detailed performance analysis study with practical applications is presented. Further, some protocol specific questions are addressed and appropriate loss models are proposed and investigated. Finally, a performance comparison with TCP-based scenarios is also provided.

The novelty of this work is the performance comparison of networks with and without congestion control. As far as we know there is no similar publication on this issue with results presenting a performance comparison of a Future Internet with and without congestion control. We wanted to make the comparison relevant to the recent Internet and that is why we chose Compound TCP (TCP version of Windows Vista/Windows Server 2008/Windows 7) and CUBIC TCP (default TCP protocol of Linux kernel) since these are the TCP versions widely used in the Internet today. *The significance of our results is that they clearly show the high potential of this new idea and verify the hope that it is possible to build a more efficient Future Internet without congestion control.* Our results also indicate the areas where this idea can be utilized most. We consider this study as a first step to a detailed comprehensive performance comparison covering more scenarios, TCP versions, parameters, etc. to establish a strong foundation for this Internet development.

In this technical report we first give a brief overview of the Future Internet concept without congestion control in Section 3. Then, in Section 4 we give a brief overview of rateless code operation. In Section 5, basic theoretical loss models are investigated with emphasis on keeping QoS requirements assuming that network entities operate according to the new concept. In Section 6, the performance of the concept is investigated in presence of fair queuing. These results are used in Section 7 to compare network performance with and without congestion control in a single bottleneck link network. Some potential disadvantages of the new concept are mentioned in Section 8. Considering the potential drawbacks, detailed comparison results are presented in a multiple bottleneck link network in Section 9. Initial interoperability ideas are addressed in Section 10. Finally, Section 11 concludes the report.

3 A New Concept for Future Internet

In a new concept of Future Internet without congestion control, where maximal rate sending is employed by all entities in the network, excessive packet loss may easily occur due to the constant overload of network resources. The most important challenge of the concept is to define a mechanism which is able to cope with this inherent packet loss together with ensuring a scalable data communication. One possible approach is the use of *rateless coding*. As opposed to conventional block coding, where typically a k-digit information word is transformed into a n-digit codeword, a rateless code (or fountain code) produces a potentially infinite stream of code symbols from the original message of size k. Therefore, the rate of the rateless encoder k/n tends to 0 as n tends to infinity, which explains the denomination. The *Luby Transform* (LT) codes [17] were the first practical realization of universal rateless erasure codes which had the advantages of being near-optimal for every erasure channel and very efficient as data length grows, simultaneously. The next generation of practical fountain codes were *Raptor codes* [18] which are an extension of LT codes offering linear time encoding and decoding. Specifically, for a message length of k symbols and any real $\varepsilon > 0$ overhead parameter, a Raptor code generates a potentially infinite stream of encoded symbols from which any subset of size $\lceil (1+\varepsilon)k \rceil$ is sufficient to recover the original k symbols with high probability. In this way, every symbol, which is transmitted successfully is useful for the decoding process. Encoded symbols can be generated using $\mathcal{O}(\log(1/\varepsilon))$ operations and the original message symbols can be recovered from the collected ones with $\mathcal{O}(k \log(1/\varepsilon))$ operations. Encoding and decoding operations are very simple constituting either exclusive-or of several symbols or a simple copy of one symbol to another.

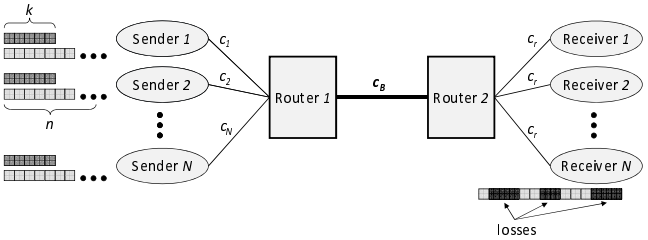

Figure 1: The network architecture with N sender-receiver pairs. Sender processes produce a potentially infinite stream of coded symbols from the original message of size k. Once $\lceil (1+\varepsilon)k \rceil$ coded symbols arrive, high probability decoding is possible. If decoding fails, the receiver should collect a bit more coded symbols before the next decoding attempt.

Figure 1 illustrates the architecture based on fountain coding on a simple dumbbell topology. Sender processes transmit their data at their maximal rates - which can be limited by either the application or the capacity of the access link - using a Raptor code realization. Raptor codes suit perfectly to the new concept of Future Internet without congestion control, as the code makes it possible to recover k message symbols from any subset of arriving code stream of size $\lceil(1+\varepsilon)k\rceil$. This property makes the data transfer fault tolerant to an arbitrary extent of loss experienced in the network, even if it is dynamically changing and occurs in heavy bursty manner. In the following, we give a more detailed description of rateless code operation and we also address some protocol specific questions.

4 Rateless Code Operation

In this section we give a brief overview of the LT and Raptor code operation and also present challenges and possible solutions to apply this coding schemes in a communication protocol over the Internet. As it was mentioned, the encoding and decoding process is quite simple, constituting only either exclusive-or of several symbols or a simple copy of one symbol to another. This encoding step is very simple, however, the mathematical construction which defines the symbols on which the exclusive-or operation is performed during an encoding step is not trivial. In order to produce an output symbol, the encoding process performs the following steps. First it generates a d number according to a well-defined probability distribution [17]. Secondly, d number of message symbols are selected according to a discrete uniform distribution on which the exclusive-or operation is performed. Note, that $d = 1$ is also a valid value, moreover it has utmost importance. The value $d = 1$ results in a simple copy of one input symbol into an output one.

Figure 2 illustrates the LT encoding and decoding process. At the decoding side, the so-called belief propagation algorithm reveals the message symbols out of the encoded stream by performing the following steps. One time connected symbols ($d = 1$) can be revealed instantly (x_4). Two time connected symbols which contain already revealed one time connected symbol can be revealed at the second step (x_3). This algorithm can be terminated if all message symbols are revealed or no further message symbols can be revealed. In the latter case the decoding process must wait for another segment of the encoded stream in the hope to be able to continue the belief propagation algorithm. As it can be seen, in this simple example, the loss of one encoded symbol (y_4) still not jeopardize the decoding process. The importance

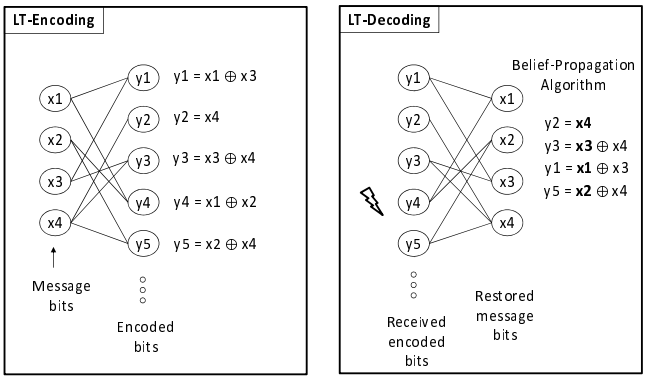

Figure 2: Illustration of the rateless encoding process and the so-called belief propagation algorithm on the decoding side. In this example one encoded symbol (y_4) is lost, still decoding can be performed.

of the basic probability distribution used to generate the d number is to mathematically guarantee the high probability decoding or equivalently the success of the belief propagation algorithm, once any subset of size $\lceil (1+\varepsilon)k \rceil$ encoded symbols arrive to the receiver.

Beside the simple operation of this rateless encoding and decoding process there is one very important detail which is to be solved yet. The belief propagation algorithm assumes, that the connectivity information is also available at each decoding step at the receiver side. The connectivity information reveals, which message symbols were used to generate a given encoded symbol. This is potentially a large overhead which should be avoided somehow. One possibility is to use large sized symbols (i.e., large packet granularity) where this redundant information containing the order of used message symbols is negligible related to the size of the actual packet. An other, smarter solution to the problem is to use pseudo-random generation of symbols according to a predefined random core. In this case the random core can be transferred to the receiver side at the beginning of the data transfer. Once the random core and probability distribution is common on both sides, only one sequence number is required to obtain the connectivity information of an encoded packet. By containing one sequence number the pseudo-random generator can produce connectivity information for any arriving symbol no matter how many symbols were lost before.

We propose a potential rateless-code based forward error correction solution with maximal rate sending. The sequence diagram of this solution is shown in Figure 3. First, the random core, the message size and the redundancy parameter are transferred to the receiver side. Once the confirmation of initializing data gets back to the source, maximal rate sending according to a proper rateless coding scheme

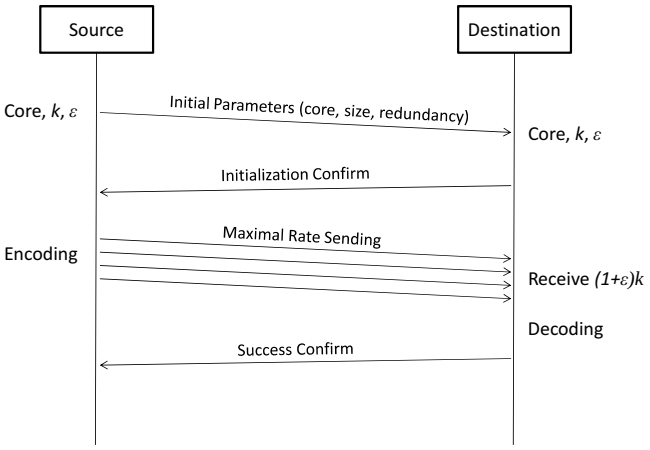

Figure 3: Sequence diagram of a potential rateless code-based protocol with maximal rate sending

can be performed. The destination entity waits for $\lceil (1+\varepsilon)k \rceil$ arriving encoded symbols, then using the belief propagation algorithm based on the common random core performs the decoding process. It is an interesting question when to start the belief propagation algorithm, i.e., it could be possible to speed up decoding process by partial decoding. However, if decoding fails after the collection of $\lceil (1+\varepsilon)k \rceil$ encoded symbols, all the destination host should do is to collect some more encoded symbols, and try to continue the belief propagation algorithm at the point it has stopped before. Once the decoding is successfully done, the receiver host should sign this success toward the source in order to stop the maximal rate data transfer.

Now, LT encoding and decoding are briefly reviewed. Based on this, Raptor codes are presented shortly in the following. In order to obtain linear time encoding and decoding complexity, the probability distribution that produces the connectivity between message and encoded symbols has changed. The connection graph becomes sparser resulting in linear complexity, however, the decoding error probability is also strongly effected.

Figure 4 depicts the decimal logarithm of the decoding error probability of a message symbol in the function of restored message size ratio. As it can be observed, at the end of the decoding process (when around 99% of the message is revealed), the error probability suddenly increases to 1. It means, that the whole message cannot be restored with probability one. To solve this problem, Raptor codes combine the advantages of rateless coding and block coding in a hybrid architecture.

Figure 5 illustrates how Raptor codes combine traditional block and rateless coding. First, a precoding is applied on the message symbols which is a properly chosen traditional block code, then LT coding

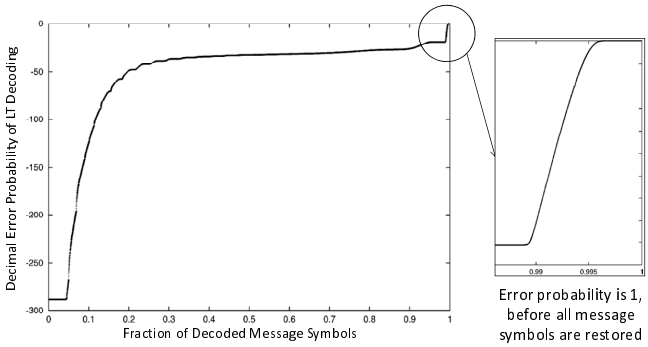

Figure 4: Decimal logarithm of decoding error probability in case of Raptor code in the function of the fraction of already decoded message symbols

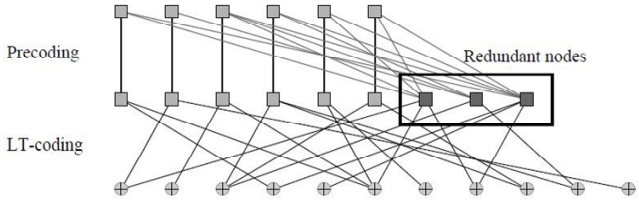

Figure 5: Layers of the Raptor coding scheme

scheme is applied on the result of the first layer producing a potentially infinite encoded stream. Figure 4 revealed how the decoding failure probability increases to one in case of the LT decoding when the connection graph is scattered. However, the ratio of the unrevealed message is quite small, with which the outer block code can easily cope with. This way, by combining the advantages of traditional block and rateless coding, Raptor codes become very effective erasure coding schemes offering linear time complexity.

In the following, successful transmission of given amount of packets via imperfect links with basic loss characteristics is investigated.

5 Basic Loss Models Focused on the Fountain Code Paradigm

Considering the fact, that every arriving packet is useful for the decoding process in a fountain coding scheme, we analyze the transmission task in basic loss environments and give tight bounds on the number of required packets to be sent, keeping specific reliability requirements, simultaneously.

5.1 Independent-Loss Channels

In this section a lossy channel with independently and identically distributed random loss (Bernoulli loss) characteristics is investigated with emphasis on keeping a predefined success criterion during data transfer. Consider the task that a sender process would like k data packets to be delivered to a receiver via an imperfect link, where each packet is dropped independently from the others with probability p and transmitted successfully with probability $1-p$. Let $X_{k,1-p}$ denote the number of packets required to be sent by the sender to fulfill this task. The random variable, $X_{k,1-p}$ follows the negative binomial distribution with mean $k/(1-p)$ and standard deviation $\sqrt{kp}/(1-p)$. The probability that $X_{k,1-p}$ equals to l can be expressed as follows:

$$P(X_{k,1-p} = l) = \binom{l-1}{k-1} p^{l-k}(1-p)^k,\ l \in \{k, k+1, ...\}$$

Note, that the above formula is not the usual definition of the negative binomial random variable, but it is equivalent to that, when the random variable denotes the number of required trials in order to successfully transfer k packets, in case of success probability $1-p$. In case of a specific success criterion for the transmission task, the minimal value of sent packets, n should be determined which satisfies the following reliability requirement:

$$P(X_{k,1-p} \leq n^*) \geq P_{sc}$$

The value of n^* determines the *minimal number of packets* required to be sent in order to effectively deliver k packets to the receiver with P_{sc} success probability (i.e., $1-10^{-6}$). In order to get a compact and easily computable formula for n^*, let us consider the negative binomial random variable, $X_{k,1-p}$ as the sum of k independent random variables (Y_1,Y_2,...,Y_k) following geometric distribution with parameter $1-p$, as each successful transmission can be described by a geometric process in the model. According to the central limit theorem, $X_{k,1-p}$ has an approximately normal distribution for sufficiently large values of k, with mean $k/(1-p)$ and standard deviation $\sqrt{kp}/(1-p)$.

$$\frac{\frac{Y_1+Y_2+...+Y_k}{k} - \frac{1}{1-p}}{\frac{\sqrt{p}}{\sqrt{k}(1-p)}} = \frac{Y_1 + Y_2 + ... + Y_k - \frac{k}{1-p}}{\frac{\sqrt{kp}}{1-p}} =$$

$$= \frac{X_{k,1-p} - \frac{k}{1-p}}{\frac{\sqrt{kp}}{1-p}} \approx N(0,1)$$

Consequently, for sufficiently large data transfers the success criterion can be formulated as follows:

$$P\left(\frac{X_{k,1-p} - \frac{k}{1-p}}{\frac{\sqrt{kp}}{1-p}} \leq \frac{n^* - \frac{k}{1-p}}{\frac{\sqrt{kp}}{1-p}}\right) \geq P_{sc}$$

Introducing $R = \Phi^{-1}(P_{sc})$ as a reliability factor according to the inverse of the standard normal distribution function, the minimal value of n is:

$$n^* = \left\lceil \frac{k}{1-p} + \frac{R\sqrt{kp}}{1-p} \right\rceil$$

In the above formula, the mean and the standard deviation of the negative binomial random variable $X_{k,1-p}$, in conjunction with the reliability factor R, determine a good approximation for the minimal number of required packets to be transmitted when k is sufficiently large.

Note, that this data transmission task in the context of Raptor codes would only scale parameter k of $X_{k,1-p}$ random variable to $k(\varepsilon) = \lceil (1+\varepsilon)k \rceil$ resulting in an other random variable following the negative binomial distribution with parameters $(k(\varepsilon), 1-p)$.

5.2 Burst-Loss Channels

Consider again the task, that k number of packets should be delivered by a sender process to a receiver via an imperfect link, but now let the packet loss be characterized by the Gilbert model [19], [20]. The Gilbert model has been commonly used to describe burst-loss channels, often found in the Internet. The aim is to give a good bound on minimal packet number (n^*), that is required to be sent in order to fulfill a predefined success criterion on transmitting k packets effectively.

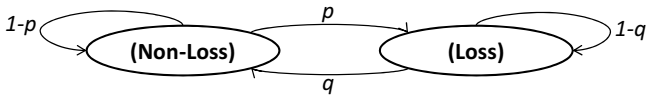

Figure 6: The Gilbert model

As Figure 6 shows, the Gilbert model has two states, where residence times in each state follow geometric distributions with parameters p and q. Let A_i denote the random variable describing the size of the i^{th} non-loss period and let X_i be the random size of the i^{th} loss period. With this notations $A_i \in G(p)$ and $X_i \in G(q)$. Let $X[k]$ denote the total amount of lost packets until k packets successfully arrive to the receiver. In order to give a proper bound on the number of packets required to be sent, the probability distribution of $X[k]$ should be determined. In the presence of a success criterion (P_{sc}) the aim is to find n^*, such that:

$$P(k + X[k] \leq n^*) \geq P_{sc}$$

Note, that $X[k]$ can be interpreted as the sum of k independently and identically distributed random

variables (L_1,L_2,...,L_k), where L_i denotes the random size of a loss period between the successful delivery of the $(i-1)^{th}$ and i^{th} packets. The probability distribution of L_i can be expressed as follows:

$$P(L_i = l) = \begin{cases} 1-p & : l = 0 \\ p(1-q)^{l-1}q & : l > 0 \end{cases}$$

The distribution of total loss $X[k]$ is then the convolution of L_1,...,L_r random variables, where L_i has mean p/q and standard deviation $\sqrt{p(2-p-q)}/q$. To cope with the problem of giving a good bound on n^* avoiding the calculation of the convolution of $\{L_i\}$ variables, the central limit theorem can be used, as for sufficiently large value of k we get that:

$$\frac{\frac{L_1+L_2+...+L_k}{k} - \frac{p}{q}}{\frac{\sqrt{p(2-p-q)}}{\sqrt{k}q}} = \frac{L_1 + L_2 + ... + L_k - \frac{kp}{q}}{\frac{\sqrt{kp(2-p-q)}}{q}} =$$

$$= \frac{X[k] - \frac{kp}{q}}{\frac{\sqrt{kp(2-p-q)}}{q}} \approx N(0,1)$$

Therefore, for sufficiently large value of k, the success criterion with success probability P_{sc} can be written as follows:

$$P\left(\frac{X[k] - \frac{kp}{q}}{\frac{\sqrt{kp(2-p-q)}}{q}} \leq \frac{n^{**} - \frac{kp}{q}}{\frac{\sqrt{kp(2-p-q)}}{q}}\right) \geq P_{sc}$$

Note, that according to the definition of $X[k]$, n^{**} denotes the necessary number of packets required to be sent on top of the k packets which must be delivered to the receiver. Therefore $n^* = n^{**} + k$ stands and introducing $R = \Phi^{-1}(P_{sc})$ we get the following:

$$\begin{aligned} n^{**} &= \left\lceil \frac{kp}{q} + \frac{R\sqrt{kp(2-p-q)}}{q} \right\rceil \\ n^* = n^{**} + k &= \left\lceil \frac{k(p+q)}{q} + \frac{R\sqrt{kp(2-p-q)}}{q} \right\rceil \end{aligned}$$

Considering the Gilbert model parameters in terms of burstyness, in case of $p < 1-q$ the model describes bursty loss, and scattered loss for $p > 1-q$. Note, that in case of $p = 1-q$ the Bernoulli model can be obtained as follows:

$$\begin{aligned} n^* &= \left\lceil \frac{k(p+q)}{q} + \frac{R\sqrt{kp(2-p-q)}}{q} \right\rceil |_{p=1-q} = \\ &= \left\lceil \frac{k}{1-p} + \frac{R\sqrt{kp}}{1-p} \right\rceil \end{aligned}$$

The value of n^* can be compared to the one obtained under Bernoulli loss. The case $p < 1-q$ results in higher and the case $p > 1-q$ results in lower values of required packets compared to the one obtained

when $p = 1-q$. The difference (n^*_{diff}) which is determined by the bias in expectation values and standard deviations can be expressed as follows:

$$n^*_{diff} = \left\lceil kp(\frac{1}{q} - \frac{1}{1-p}) + R(\frac{\sqrt{kp}}{1-p} - \frac{\sqrt{kp(2-p-q)}}{q}) \right\rceil$$

The results presented in this section give tight bounds on the number of required packets to be sent in case of two basic loss models. They also indicate that the proportion of the uncertainty expressed by the standard deviation relating to the expectation value of the random process, becomes insignificant for large message sizes. Specifically, with increasing message size k, both $(R\sqrt{kp}/(1-p))/(k/(1-p))$ and $(R\sqrt{kp(2-p-q)}/q)/((k(p+q))/q)$ tend to 0.

6 Performance of Data Transfer with Fountain Codes without Congestion Control

The great challenge of the Future Internet scenario without congestion control is to deal with the excessive and bursty packet loss induced by constant overload of network resources due to the maximal rate sending mechanism. Employing a proper fountain code, i.e., a Raptor code can cope with the inherent loss even in the case of heavy and dynamically changing burst loss characteristics. However, as far as fairness is concerned, a mechanism which guarantees equal bandwidth allocation among flows accessing overburdened links is highly desired. To solve the share allocation problem among flows sending at maximal rates, some kind of fair sharing allocation method must be used. In the followings, the architecture shown in Figure 1 is analyzed, where a fair queuing scheduler is responsible for the fair share rates at the bottleneck link. Specifically, consider the dumbbell environment with N different flows trying to access the network at maximal sending rates limited by their own access link capacities $(c_1, c_2, ..., c_N)$. In case of N competing flows the fair share capacity for the bottleneck link with capacity c_B is c_B/N for each flow. However, there could be flows with access rates lower than the fair share value. In this case, considering an ideal fair queuing mechanism, flows having higher access rates share equally the capacities which were not used by flows being below the fair share rate. The *actual shared rate* for flow i, s_i can be expressed

as follows:

$$s_i = \begin{cases} \frac{c_B}{N} & : \forall j,\ c_j \geq \frac{c_B}{N} \\ c_i & : c_i \leq \frac{c_B}{N} \\ \frac{c_B - \sum_{k=1}^{N} I_{\{c_k < \frac{c_B}{N}\}} c_k}{\sum_{k=1}^{N} I_{\{c_k \geq \frac{c_B}{N}\}}} & : \exists j,\ c_j < \frac{c_B}{N} \text{ and } c_i \geq \frac{c_B}{N} \end{cases}$$

The *average burst loss size* for flow i can be calculated by relating its original access rate to the shared rate it actually possesses on the bottleneck link.

$$b_i = \begin{cases} \max\{\frac{c_i}{s_i} - 1, 1\} & : c_i > s_i \\ 0 & : c_i \leq s_i \end{cases}$$

If the shared rate is equal to the access rate of the flow, i.e., the access rate of the flow is not higher than the fair share rate, then the demand of the flow can completely be fulfilled resulting in zero average burst loss length. On the other hand, when the access rate of the flow is higher than the shared rate, packet losses occur with an average burst loss length value of $\max\{\frac{c_i}{s_i} - 1, 1\}$, as on average only every $(\frac{c_i}{s_i})^{th}$ packet can get transmitted. Naturally, if loss actually occurs, then at least one packet is involved. From the ratio of the access rate and shared access rate, the *success ratio*, t_i can also be calculated as follows:

$$t_i = \begin{cases} \frac{1}{\frac{c_i}{s_i}} = \frac{s_i}{c_i} & : c_i > s_i \\ 1 & : c_i = s_i \end{cases}$$

The loss ratio for flow i, l_i is then $1 - t_i$. Now suppose the scenario, when N flows try to access the network with constant access rates $\{c_1,c_2,...,c_N\}$, using Raptor coding. The code parameters for the i^{th} flow are k_i - message block size in packets and ε_i - redundancy. The required number of packets to be sent (n_i) in order to receive $(1+\varepsilon_i)k_i$ packets at the receiver side can be expressed as:

$$n_i = \frac{(1+\varepsilon_i)k_i}{1-l_i} = \frac{(1+\varepsilon_i)k_i}{t_i}$$

The time required to send n_i packets with access rate c_i is then $T_i = n_i/c_i$. Eventually, considering the fact, that the useful amount of data is only the size of the original message k_i, a *goodput* definition, G_i is worth to be introduced which can be expressed as:

$$G_i = \frac{k_i}{T_i} = \frac{k_i}{\frac{n_i}{c_i}} = \frac{c_i k_i (1-l_i)}{(1+\varepsilon_i)k_i} = \frac{c_i t_i}{1+\varepsilon_i} = \frac{c_i \frac{s_i}{c_i}}{1+\varepsilon_i} = \frac{s_i}{1+\varepsilon_i}$$

Where by substituting the shared rate s_i, the following formula can be obtained for the goodput of flow i:

$$G_i = \begin{cases} \frac{c_B}{(1+\varepsilon_i)N} & : \forall j,\ c_j \geq \frac{c_B}{N} \\ \frac{c_i}{1+\varepsilon_i} & : c_i \leq \frac{c_B}{N} \\ \frac{c_B - \sum_{k=1}^{N} I_{\{c_k < \frac{c_B}{N}\}} c_k}{(1+\varepsilon_i) \sum_{k=1}^{N} I_{\{c_k \geq \frac{c_B}{N}\}}} & : \exists j,\ c_j < \frac{c_B}{N} \text{ and } c_i \geq \frac{c_B}{N} \end{cases}$$

These results indicate, that the link utilization - in terms of goodput - achieved by flow i on the bottleneck link is in inverse proportion to the redundancy, ε_i with gradient $1/(1+\varepsilon_i)$. Consequently, the transmission time (T_i) of a given amount of data (D_i) is directly proportional to the redundancy with the same gradient ($T_i = D_i/G_i$). It is important to note, that if the summarized access rate of flows exceeds the bottleneck link capacity, then the bottleneck link is always fully utilized, otherwise erasure coding should not be used in order to achieve maximal efficiency. However, the latter scenario is highly unlikely to occur, therefore our focus is on the former one, which is when the summarized access rate exceeds the bottleneck capacity significantly. Referring back to the section dealing with different random loss environments, the employment of a fair queuing mechanism results in a deterministic loss process for each flow trying to access the network, which depends only on the ratio of the access rates and the bottleneck link capacity. The fair queuing mechanism dissolves the uncertainty expressed by the standard deviation in a random loss environment by making the random loss process deterministic with predefined queuing rules, where each flow shares equally the bottleneck capacity. The value of n_i overlaps with the expectation value of a negative binomial process with parameters ($\lceil(1+\varepsilon_i)k_i\rceil$,$t_i$) in case of the Bernoulli loss model presented in Section 2. Furthermore, in case of fair queuing, the proportion of different transmission redundancies become independent of the transmission size. Specifically, we can define the *transmission redundancy* (ε_{tr}) as the number of sent packets on top of the required ones related to the required ones. For flow i the transmission redundancy can be expressed as:

$$\varepsilon_{tr}(i) = \frac{\frac{(1+\varepsilon_i)k}{t_i} - (1+\varepsilon_i)k}{(1+\varepsilon_i)k} = \frac{1}{t_i} - 1$$

Consequently, the ratio of two different transmission redundancies, i.e., between flow i and j, depends only on the relation of success ratios t_i and t_j, accordingly on the access rates and on the bottleneck capacity. Without proof we mention, that in case of the Bernoulli loss model, the ratio of transmission redundancies is dependent of the transmission size. This dependency stands due to the non-zero standard

deviation of the negative binomial process.

7 Performance Comparison with TCP

Previous sections revealed that the deterministic behavior of maximal rate sending with rateless coding and fair scheduling makes it possible to easily analyze system features under simple topologies like dumbbell. On the other hand, the behavior of current high speed transport protocols using closed loop congestion control mechanism can not be evaluated analytically so easily. In order to give an appropriate comparison between the cases with and without congestion control, current high speed TCP variants, Microsoft's Compound TCP and today's Linux kernel default CUBIC TCP were investigated through NS2 simulations. The network model used is similar to the one depicted in Figure 1. The delay of access links and the bottleneck link are both 10 ms. The TCP segment size is set to 1000 bytes. Access links are 1 Gbit links on the sender side ($c_1, c_2, ..., c_N$) and are considered limitless on the receiver side (c_r). The bottleneck link capacity is set to 10, 30, 50, 70 and 100% of the access link bandwidth in order to obtain different bottleneck environments ($c_B = \{100, 300, 500, 700, 1000\}$ [Mbit]). Buffer sizes at the bottleneck link are set in a logarithmically decreasing manner related to the bandwidth delay product (BDP) pointing toward very small buffer sizes. In this way, 1 BDP, 0.1 BDP and 0.01 BDP buffer sizes are set in the investigations. According to the rule-of-thumb for choosing router buffer size presented in [21], 1 BDP is considered as a large buffer scenario. In order to obtain a traffic environment where initial TCP rate variations are avoided and to reduce uncertainty of simulation results, 5 homogeneous TCP flows are used, and observed only after a specific transient time, i.e. 300 seconds, giving the traffics chance to share on the bottleneck link in a fair way.

7.1 Long-Scale Data Transfer

First, a longer scale TCP scenario is investigated, where 5 data transfers last for 1800 seconds after the 300-second-long transition interval and where our focus is on the averaged data amount which is successfully transmitted during this period. In the case without congestion control, transferred data size for flows is calculated according to the analytical consideration presented in the previous section ($D_i = G_i T_i$), assuming a hypothetical Raptor coding scheme. In order to compare performance the

so-called *equivalent redundancy* is introduced as follows:

$$\varepsilon_{equ} = \frac{T_i C_B}{D_i N} - 1$$

In the above equation, D_i is the successfully transferred data amount excluding retransmission data and T_i is the time required for the transmission in a TCP transfer. This metric makes connection between the cases with and without congestion control by indicating the redundancy at which the Raptor coding scheme achieves the same performance as the TCP. Note, that at specific $\varepsilon < \varepsilon_{equ}$ redundancies the rateless coding scheme outperforms the TCP as goodput increases with decreasing redundancy. Consequently, if the equivalent redundancy is above the practical limit of a rateless code, then a performance gain can be achieved by decreasing the redundancy up to the specific practical limit. Zero equivalent redundancy value would belong to a fair optimal case, where the rateless coding scheme performs the same as TCP only for zero redundancy, which is fair as every flow gets the same fair share capacity, and optimal because no redundant packets would be required for obtaining the original message from the coded stream. Clearly, this case can only be feasible when data transfers cause no congestion, therefore no erasure coding is required in the network. However, when rateless coding is employed, some redundancy is necessary for high probability decoding. Note, that a particular TCP flow among others could reach zero equivalent redundancy value at the expense of fairness, as it is assumed that the rateless coding scheme operates in presence of an ideal fair queuing mechanism, resulting in identical transferred data sizes or data transfer times. Therefore, if a TCP flow would get higher bandwidth in an unfair way, then transferred data size could be higher or transfer time could be shorter than the fair one.

Figure 7 shows the equivalent redundancy values for different TCP variants and logarithmically decreasing buffer sizes. Since an efficient Raptor code with $\varepsilon \approx 0.05$ can easily be applied [22], the $\varepsilon_{equ} > 0.05$ region shows that better performance can be achieved in some scenarios without TCP. The boundary of practically applicable redundancy is marked with a thickened line in the figure. It can be observed that large buffer scenarios (1 BDP) yield equivalent redundancy values below 0.05, which are under the practical value of Raptor code redundancy. In these cases TCP performs better on the long run. However, when buffer sizes are decreased to 0.1 BDP and further to 0.01 BDP, equivalent redundancy values begin to increase into the practical region of Raptor codes. The increment tends to raise with increasing bottleneck bandwidth and decreasing buffer size except in the case of Compound TCP with 500 and 700 Mbit/s bottleneck link capacity. Unfitness to the tendency in the latter cases is caused by TCP

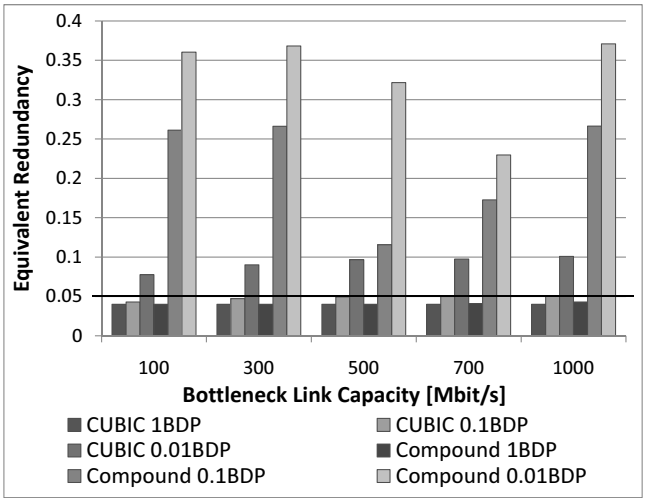

Figure 7: Equivalent redundancy values in case of different TCP variants with logarithmically decreasing buffer sizes in the long data transfer scenario (1800 sec). The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

unfairness, where experimental relative deviations of transferred data sizes are around 20%, indicating unfair bandwidth allocation. *The results clearly indicate the high potential of the solution without TCP in case of high speed and small buffer scenarios.* It is interesting to note, that Compound TCP suffers significantly at small buffer sizes.

In order to give a more conventional performance comparison, Figure 8 depicts actual performance increases by depicting the goodput increase gained from the use of a Raptor coding scheme with $\varepsilon = 0.05$ redundancy related to the TCP scenarios. For example, if the equivalent redundancy is 15% (0.15), then let G_1 be the goodput value achieved by the Raptor coding scheme with the 15% redundancy. Now, the rateless coding scheme performs the same as TCP with 15% redundancy. However, a practical Raptor code can be achieved with only 5% redundancy. Therefore, let G_2 denote the goodput value achieved with this practical redundancy limit. Then the actual rate increase can be calculated as $(G_2 - G_1)/G_1$. Consequently, equivalent redundancy values below the 5% redundancy limit map to negative rate increase values, whereas equivalent redundancy values above the 5% limit belong to a real performance increase.

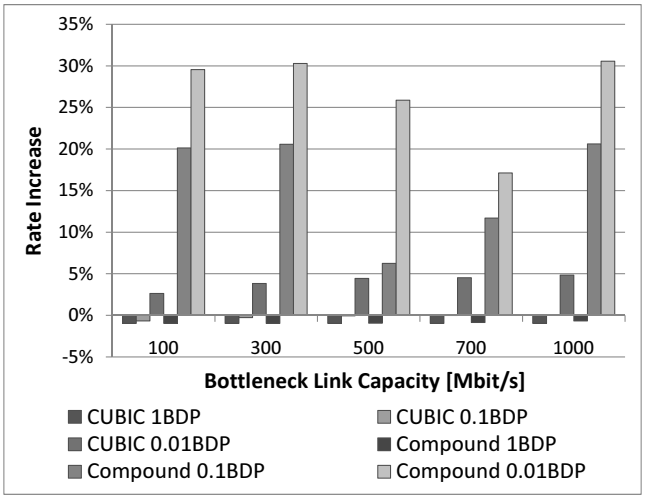

Figure 8: Performance gain (in terms of rate increase) in case of different TCP variants with logarithmically decreasing buffer sizes in the long data transfer scenario (1800 sec). The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

7.2 Small-Scale Data Transfer

As opposed to the long data transfer case where TCP rate variation is not significant on the long run, and where averaged results give a quasi-best case evaluation for the congestion controlled network, the second investigated scenario is a small data transfer case. Now 5 homogeneous TCP traffics try to transfer a web-traffic like objects (i.e., 1 MB) after a transition period of 300 seconds. Our focus is on the time when all transfers are finished. This time can be compared to the one obtained by the analytical consideration for the maximal rate sending scenario where congestion control is omitted from the network ($T_i = D_i/G_i$). In the latter case, download times are considered to be equal due to the assumed ideal fair queuing mechanism.

Figure 9 shows the equivalent redundancy values in the small data transfer case when the transferred data size (D_i) is fixed at 1 megabyte and the transmission time belongs to the longest download process. For comparability and perspicuity purposes redundancy values are truncated at 0.4. The small data transfer case results in higher equivalent values in general. Note, that values are always in the practical region of Raptor codes, above 0.05, even in the case of large buffer scenarios. *These results show that the new concept performs significantly better in all the investigated small-scale data transfer scenarios, which indicates the feasibility of the concept also in web-traffic like communication.*

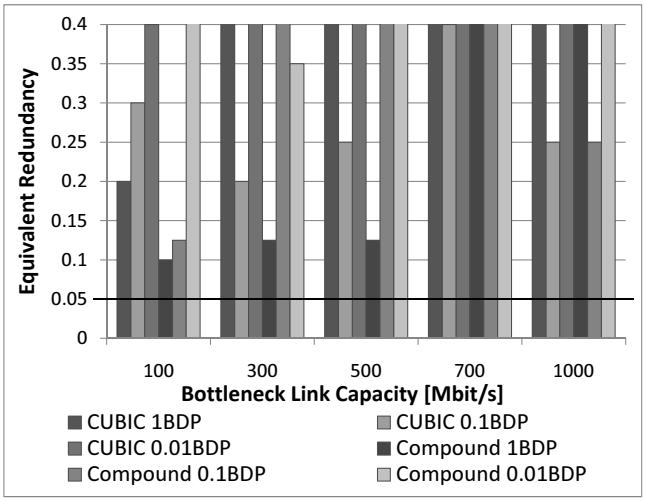

Figure 9: Equivalent redundancy values in case of different TCP variants with logarithmically decreasing buffer sizes in the small data transfer scenario (transfer of 1 MB). The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

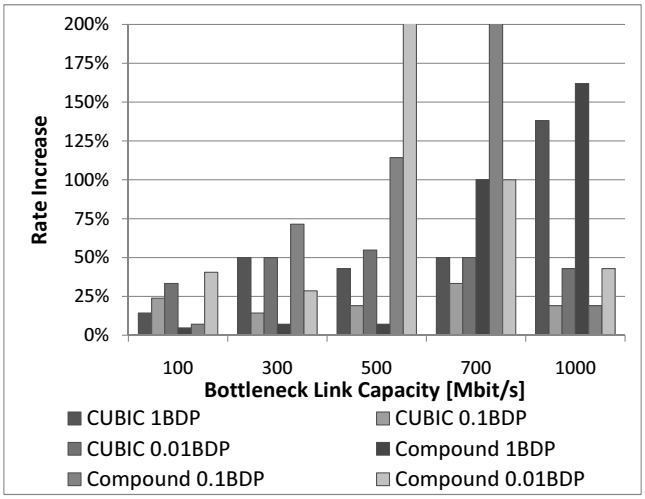

Figure 10: Performance increase in case of different TCP variants with logarithmically decreasing buffer sizes in the small data transfer scenario (transfer of 1 MB). The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

Figure 10 depicts the performance increase in the small-scale data transfer scenario. As equivalent redundancy values were always is the practical region of Raptor code, above the 5% limit, all rate increase values are positive, indicating significantly better performance in almost all investigated cases.

7.3 Medium-Scale Data Transfer

The third investigated scenario is the practical case of downloading several hundred megabytes (one-tenth of a DVD). The 300-second transition period is also used to avoid the potentially high extent of initial rate variation of TCP. Our focus is on the longest download time regarding 5 homogeneous TCP transfers, which is then compared to the one obtained by the analytical results for the maximal rate sending scenario when the data size (D_i) is fixed at 470 megabytes. In the latter case, the assumed ideal fair queuing makes the transmission times equal.

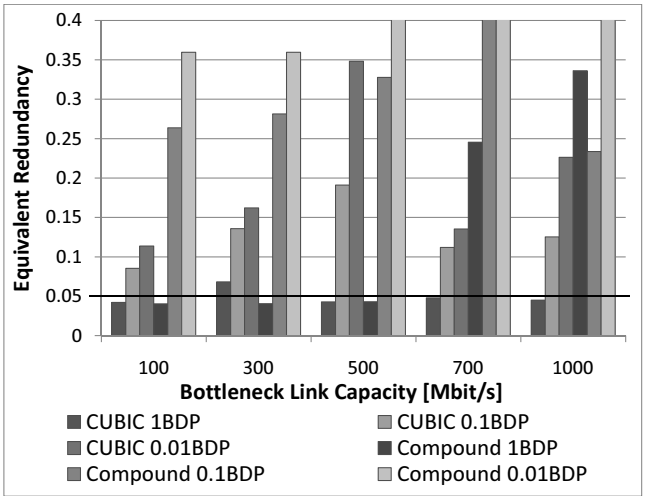

Figure 11: Equivalent redundancy values in the practical case of transferring several hundred megabytes (one-tenth of a DVD). The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

Figure 11 depicts the equivalent redundancy values which turn into the practical region of Raptor codes (above 0.05) for smaller buffer sizes for both TCP variants, but also show more variability relating them to the results represented by Figure 7. The high variability is due to the shorter scale TCP uncertainty caused by rate fluctuation. For perspicuity purposes equivalent redundancy values are truncated at 0.4, cutting down some results of Compound TCP at higher bandwidths, where it suffers significantly.

Figure 12 displays actual goodput increase values which turn into the negative region only for large buffer size scenarios.

The tendency observed in Figure 7 makes sense to examine the network for higher bandwidth and lower buffer size, pointing toward optical network environments. Figure 13 confirms the tendency in case

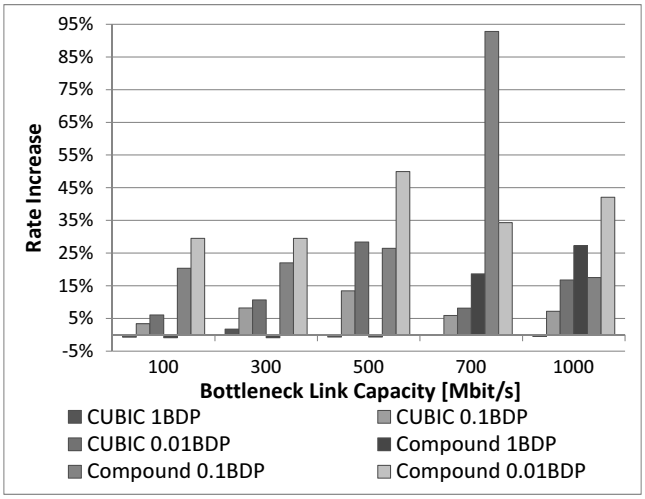

Figure 12: Rate increase values in the practical case of transferring several hundred megabytes (one-tenth of a DVD). The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

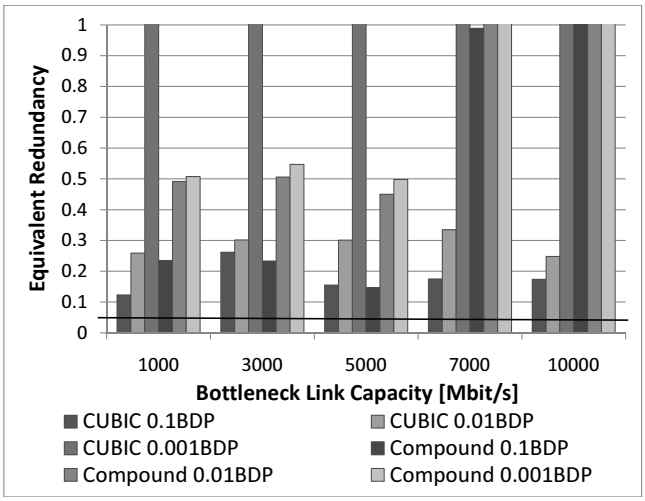

Figure 13: Equivalent redundancy values in the practical case of transferring several hundred megabytes obtained in a 10 Gbit network with logarithmically decreasing buffer sizes. The high buffer case is omitted, instead a smaller buffer size 0.001 BDP is introduced. The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 0.1BDP, CUBIC 0.01BDP, CUBIC 0.001BDP, Compound 0.1BDP, Compound 0.01BDP, Compound 0.001BDP.

of the transfer of several hundred megabytes and shows that equivalent redundancy values are higher than 0.1 in all small buffer scenarios. It also shows that CUBIC TCP suffers when buffer size is decreased significantly (0.001 BDP).

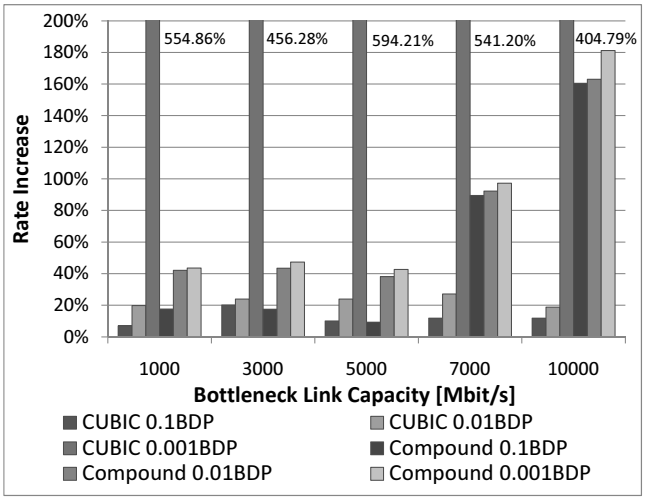

Figure 14: The performance gain achieved in the 10 Gbit network relating TCP goodput to the one obtained with a Raptor coding scheme using $\varepsilon = 0.05$ redundancy. The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 0.1BDP, CUBIC 0.01BDP, CUBIC 0.001BDP, Compound 0.1BDP, Compound 0.01BDP, Compound 0.001BDP.

Equivalent redundancy values are truncated at $\varepsilon_{equ} = 1$, but Figure 14 shows particular results depicting the goodput increase. In significantly decreased buffer size scenarios (0.001 BDP) performance gains are more than 400% for CUBIC TCP and are higher than 40% for Compound TCP with an increasing tendency in bandwidth. *These results clearly show that not only in small-scale transfers, but also in larger data transmissions, the new concept significantly outperforms today's congestion controlled network architectures at high speed and small buffer scenarios.*

8 Potential Disadvantages

Previous sections revealed, that maximal rate sending with rateless code-based forward error correction can reach and can even outperform TCP's performance in wide range of network scenarios. The potential advantages, foremost *simplicity* (network buffers with highly reduced capacity) but also *efficiency*, *stability*, *robustness* are appealing, especially thinking of all-optical networks where very small buffer sizes would be highly required. However, there are other cases where maximal rate sending is not feasible due to economical or technical reasons. The maximal rate sending method assumes that source hosts own their access link to the first network router, where they can freely choose sending rates with flat rate pricing. Unfortunately, it is not the case in wireless communication, i.e., in the radio access network (RAN)

where users compete for a shared medium. In this case, maximal rate sending in the access network would cause high extent of resource waste, that is why it is infeasible. Unfortunately, circumstances do not improve in the core network either. One could think of a method which change from a rate controlled data transfer to maximal rate sending at the border of the core network. However, it could only work if the border router would buffer data packets and it would employ the maximal rate sending with rateless coding on the buffered packets, as the source mobile host cannot transfer at maximal speed at all times. This buffering solution would jeopardize the use of delay-sensitive mobile applications.

The maximal rate sending mechanism could be feasible in a wireless broadcast scenario, where a whole group of users receives the same data transfer in the RAN. In this case, maximal rate sending can be used along the entire way, from the source to the mobile destination hosts in downlink direction. The great advantage of this scenario is, that it is no matter which packets are received by the different mobile hosts, once sufficient amount of encoded stream arrives to the mobiles, the transferred message can be restored with high probability. In this way, users experiencing different coverage and interference characteristics can use the different segments of the same encoded stream to obtain their data. Consequently, the mobile user with the worst radio access performance should wait for the longest time to retrieve useful information, however no retransmission is needed from the source during the whole data transfer.

In previous sections, it was assumed that an ideal fair queuing mechanism gives the fair share bandwidth for all flows competing for a bottleneck link. However, using blind, hop-by-hop techniques to provide fairness could easily result in poor overall performance. This serious problem arises in multiple bottleneck networks which is discussed in the next section.

9 Performance Comparison in Multiple Bottleneck Link Network

In the previous sections, a single bottleneck network was investigated, where the performance comparison based on the introduced equivalent redundancy definition revealed, that rateless code-based forward error correction can be feasible in wide range of data transfer scenarios, especially in small buffer cases. However, in a more complex network with multiple bottleneck links the end-to-end performance of individual fair-queuing mechanisms on the transmission route can be jeopardized.

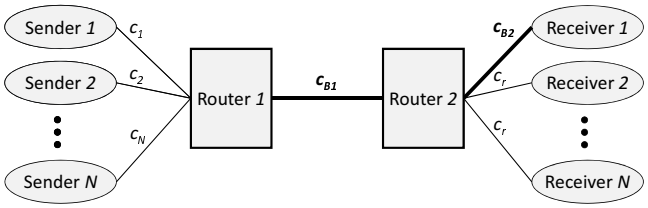

Figure 15: Simple network architecture with multiple bottlenecks (C_{B1}, C_{B2}). The end-to-end optimality of individual fair schedulers becomes questionable.

Consider the simple modification of the dumbbel topology, where an additional bottleneck link is added at the end of the route between the first source-destination pair. This simple network architecture with multiple bottleneck links (C_{B1}, C_{B2}) is depicted in Figure 15.

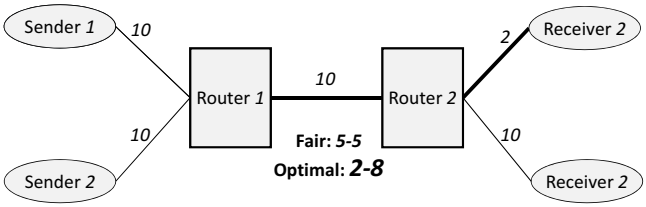

Figure 16: Simple network architecture example with multiple bottlenecks where end-to-end optimality is jeopardized by blind, hop-by-hop fair queuing

Now consider the case of two transferring flows ($N = 2$), with 10 Gbit access link capacities at sender side, $C_{B1} = 10$ Gbit, $C_{B2} = 2$ Gbit and 10 Gbit access link capacity for the second flow at the receiver side as shown in Figure 16. In this arrangement the fair share capacity at the first bottleneck link is 5-5 Gbit, however the first flow can only exploit 2 Gbit later on its route due to the second bottleneck link. Consequently, the end-to-end throughput is 7 Gbit which is not optimal. The optimal allocation would be 2-8 Gbit at the first bottleneck link resulting in an overall throughput of 10 Gbit. It can be seen, that in case of multiple bottleneck networks, using blind, hop-by-hop fair scheduling could easily result in not optimal throughput performance. On the other hand, regarding TCP's mechanism, no performance degradation is expected on the long run due to its end-to-end control mechanism which can adapt the sending rate to the tightest bottleneck between a given source-destination pair. Considering maximal rate sending with rateless code-based forward error correction, an information propagation mechanism would be required which would inform fair schedulers of estimated link capacities later on the routes in order to perform weighted fair queuing mechanism at early bottleneck links to provide more optimal

overall throughput. In the followings the performance of high speed TCP variants is compared to the rateless performance in the simple multiple bottleneck link network.

9.1 Long-Scale Data Transfer

The multiple bottleneck link network scenario is investigated similarly as the simple dumbbell topology considering CUBIC and Compound TCP, different central bottleneck link capacities (C_{B1}) and logarithmically decreasing buffer sizes. Access rate capacities at sender side are 1 Gbit link, whereas limitless at the receiver side. The additional bottleneck link (C_{B2}) is used with fixed 50 Mbit capacity.

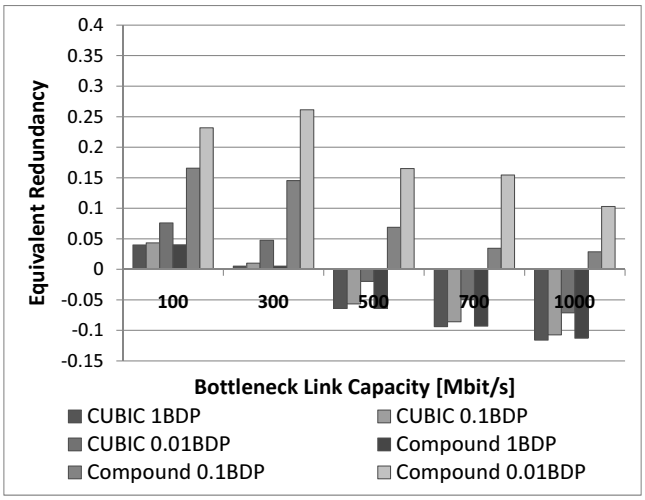

Figure 17: Equivalent redundancy values in case of different TCP variants with logarithmically decreasing buffer sizes in the long data transfer scenario (1800 sec) in a multiple bottleneck link network. The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

Figure 17 shows the equivalent redundancy values in the long-scale data transfer scenario in case of the multiple bottleneck link network. It can be seen, that equivalent redundancy values become lower, and they even turn to the negative region for higher central bottleneck link capacities. Negative equivalent redundancy values indicate that the rateless coding scheme performs a lot worse than a given TCP. It means, that the redundancy would have to be negative to reach the same performance as TCP. The decreasing tendency in equivalent redundancy in the function of the central bottleneck link capacity is due to the increasing gap between the fair share capacity and the fixed 50 Mbit additional bottleneck capacity. By increasing the central link capacity, more and more bandwidth is wasted due to the blind

hop-by-hop fair queuing mechanism. If the additional bottleneck information would propagate somehow to the fair scheduler of the central bottleneck link, then it could allocate only 50 Mbit for the first flow and the rest of the capacity for the other flow, which would result in a better overall performance than TCP. TCP's performance is not jeopardized on the long run, because it has an end-to-end congestion control mechanism which is able to take into account whole path characteristics regarding capacity allocation.

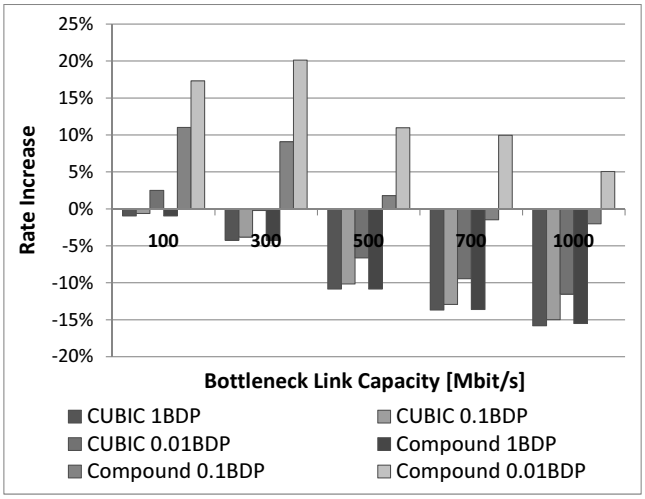

Figure 18: Rate increase values in case of different TCP variants with logarithmically decreasing buffer sizes in the long data transfer scenario (1800 sec) in a multiple bottleneck link network. The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

Figure 18 depicts actual rate increases which also turn to the negative region where equivalent redundancy values decrease below the 5% limit.

9.2 Medium and Small-Scale Data Transfer

In this section, performance comparisons in medium and small-scale data transfer cases are presented for the multiple bottleneck link network.

Figure 19 depicts equivalent redundancy values in the practical case of transferring several hundred megabytes in the multiple bottleneck link network. Surprisingly, these values are higher than the ones obtained in a single bottleneck link network. These results indicate that, current high speed TCP variants are not able to perform end-to-end control efficiently in this medium scale data transfer scenario.

Figure 20 shows actual rate increase values in case of the download of several hundred megabytes in

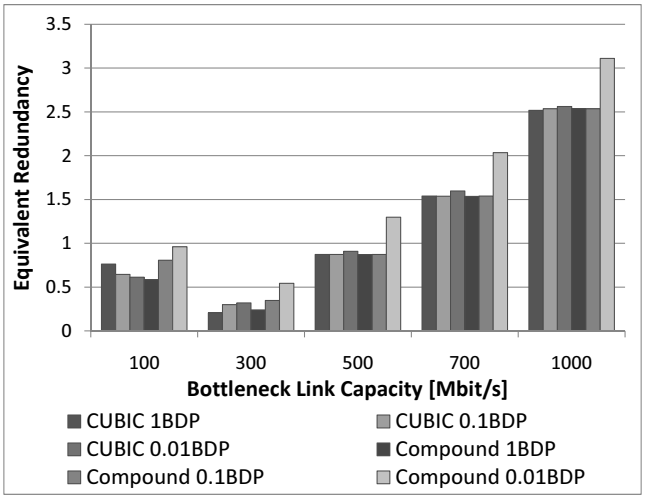

Figure 19: Equivalent redundancy values in the practical case of transferring several hundred megabytes (one-tenth of a DVD) in a multiple bottleneck link network. The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

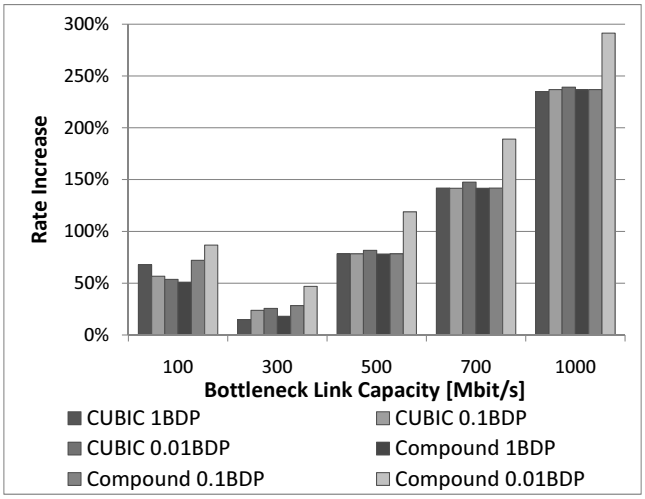

Figure 20: Rate increase values in the practical case of transferring several hundred megabytes (one-tenth of a DVD) in a multiple bottleneck link network. The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

the multiple bottleneck network. The growing tendency in rate increase indicates, that TCP is not able to determine the tightest bottleneck in such a short time.

Figure 21 displays equivalent redundancy values in the small-scale data transfer scenario in the multiple bottleneck link network. Now, it is not so surprising, that maximal rate sending outperforms TCP's

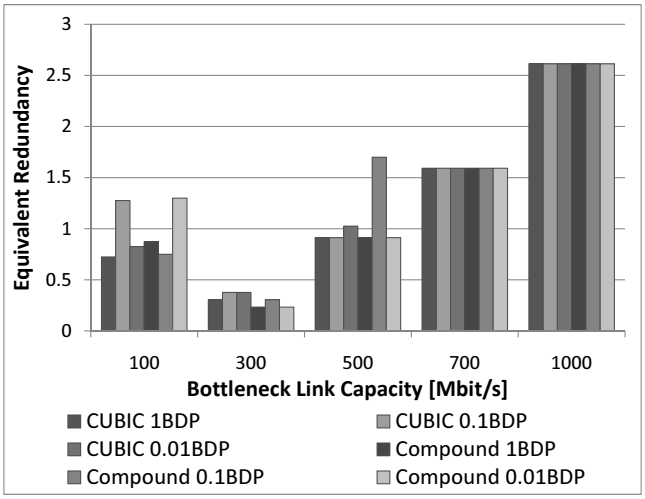

Figure 21: Equivalent redundancy values in case of different TCP variants with logarithmically decreasing buffer sizes in the small data transfer scenario (transfer of 1 MB) in a multiple bottleneck link network. The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

performance, regarding that these small-scale data transfers last for less than a second. During this small time TCP is not able to determine optimal per-flow rates, which also indicates the feasibility of the maximal rate sending method even in case of using non-optimal (blind, hop-by-hop) fair scheduling.

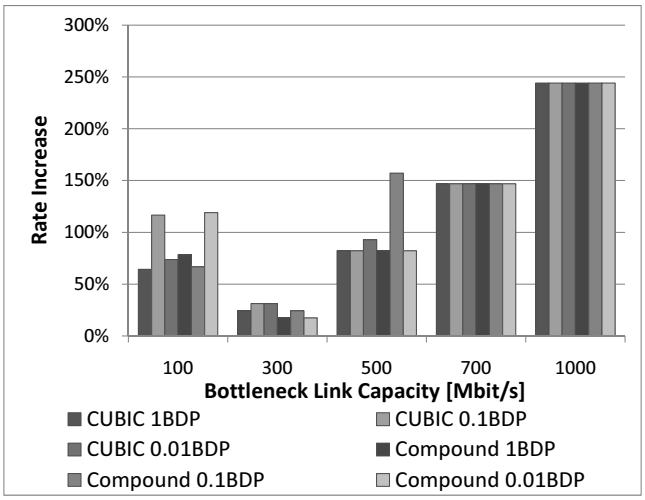

Figure 22: Rate increase values in case of different TCP variants with logarithmically decreasing buffer sizes in the small data transfer scenario (transfer of 1 MB) in a multiple bottleneck link network. The order of bars in the chart from left to right at each bottleneck capacity is the following: CUBIC 1BDP, CUBIC 0.1BDP, CUBIC 0.01BDP, Compound 1BDP, Compound 0.1BDP, Compound 0.01BDP.

Figure 22 depicts actual rate increase values, which shows a similar tendency to medium-scale data transfer, indicating growing performance gain with increasing central bottleneck link capacity.

10 Interoperability Questions

Using maximal rate sending with rateless code-based forward error correction instead of TCP's closed loop congestion control mechanism is a radically new and different idea. Considering the fact, that TCP is the most frequently used transport protocol in today's Internet architecture, the question of interoperability arises. It was presented, that maximal rate sending would require some fair queuing mechanism to prevent eager hosts to starve less active ones. Therefore, one can imagine an architecture where several traffic classes are in present (at least 2, for TCP and for maximal rate sending). In one traffic class with potentially lower priority this fair queuing mechanism would operate on flows sending at maximal rates, whereas the other class would belong to the TCP transmission layer.

11 Conclusion

In this technical report we have investigated a potential Future Internet scenario in the absence of congestion control. We briefly presented the basics of rateless codes including Raptor codes on which the new concept could be based on. We have given tight bounds on packets required to be sent using rateless coding scheme with large message sizes in case of basic loss models and predefined QoS requirements. We have shown that an ideal fair queuing scheduler makes the transmission process deterministic and the transmission redundancy independent of the message size. Presented comparison results have revealed that network architectures without congestion control can even outperform current high speed TCP protocols in case of large bandwidth and small buffer scenarios. Beside mentioning advantages of the new architecture some potential disadvantages were investigated, especially focusing on the non-ideal case of blind, hop-by-hop fair scheduling in a multiple bottleneck link network. We have also addressed some protocol specific questions and proposed some possible solutions pointing toward future research. However, the most important message of this technical report is, that the significant performance increase in very small buffer scenarios together with the projected benefits such as efficiency, simplicity and stability, makes the new concept promising toward the way to all-optical networking.

Bibliography

[1] S. Floyd, "HighSpeed TCP for Large Congestion Window", IETF RFC 3649, 2003.

[2] T. Kelly, "Scalable TCP: Improving Performance in Highspeed Wide Area Networks", ACM SIGCOMM Computer Communication Review, Vol. 33, No. 2, pp. 83-91, 2003.

[3] L. Xu, K. Harfoush and I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks", Proceedings of IEEE Infocom 2004, Hong Kong, China, pp. 2514- 2524, 2004.

[4] I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", Proceedings of Third International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2005), Lyon, France, 2005.

[5] C. Jin, D. X. Wei and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance", Proceedings of IEEE Infocom 2004, Hong Kong, China, pp. 2490-2501, 2004.

[6] R. Wang, K. Yamada, M. Y. Sanadidi and M. Gerla, "TCP with sender-side intelligence to handle dynamic, large, leaky pipes", IEEE Journal on Selected Areas in Communications, Vol. 23, No. 2, pp. 235-248, 2005.

[7] K. Tan, J. Song, Q. Zhang and M. Sridharan, "A Compound TCP Approach for High-speed and Long Distance Networks", Proceedings of IEEE Infocom 2006, Barcelona, Spain, 2006.

[8] D. Katabi, M. Handley and C. Rohrs, "Congestion control for high bandwidth-delay product networks", Proceedings of ACM SIGCOMM 2002, Pittsburgh, PA, USA, 2002.

[9] S. Molnár, B. Sonkoly and T. A. Trinh, "A Comprehensive TCP Fairness Analysis in High Speed Networks", Computer Communications, Elsevier, Volume 32, Issues 13-14, 17 August 2009, pp. 1460-1484.

[10] S. Floyd and K. Fall., "Promoting the use of end-to-end congestion control in the internet", IEEE/ACM Transactions on Networking, 7:458-472, 1999.

[11] David Clark, Scott Shenker and Aaron Falk, "GENI Research Plan", Version 4.5 of April 23, 2007.

[12] H. Park, E. F. Burmeister, S. Bjorlin, and J. E. Bowers, "40-gb/s optical buffer design and simulations", In Numerical Simulation of Optoelectronic Devices (NUSOD), 2004.

[13] Barath Raghavan and Alex Snoeren, "Decongestion Control", ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-V), November 2006.

[14] Thomas Bonald, Mathieu Feuillet and Alexandre Proutiere, "Is the "Law of the Jungle" Sustainable for the Internet?", The INFOCOM 2009, Rio de Janeiro, Brazil, 2009.

[15] Luis López, Antonio Fernández and Vicent Cholvi, "A game theoretic comparison of TCP and digital fountain based protocols", Computer Networks,Computer Networks, Vol. 51, Issue 12, pp. 3413-3426, 2007.

[16] Ahmad, S., Hamzaoui, R. and Al-Akaidi, M., "Robust live unicast video streaming with rateless codes", 16th International Packet Video Workshop, Lausanne, Nov. 2007.

[17] Michael Luby, "LT Codes", Proceedings of the 43rd Symposium on Foundations of Computer Science, pp. 271-280, 2002.

[18] Amin Shokrollahi, "Raptor Codes", IEEE Transactions on Information Theory, Vol. 52, No. 6, June 2006.

[19] Gerhard Haßlinger and Oliver Hohlfeld, "The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet", 14th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems, pp. 269-286, 2008.

[20] Wenyu Jiang and Henning Schulzrinne, "Comparison and Optimization of Packet Loss Repair Methods on VoIP Perceived Quality under Bursty Loss",Proceedings of the 12th International Workshop on Network and Operating System Support for Digital Audio and Video, 2002.

[21] G. Appenzeller, I. Keslassy and N. McKeown, "Sizing router buffers", In SIGCOMM '04, pp. 281-292, New York, NY, USA, 2004. ACM Press.

[22] D. MacKay, "Fountain codes", in The IEE Proc., vol. 152. IEE, Dec 2005, pp. 1062-1068