Fairness Evaluation of Pipeline Coded and non Coded TCP Flows

Paul-Louis AGENEAU[1], Nadia BOUKHATEM[1], Mario GERLA[2]

[1]Computer Science & Network Dept., Telecom ParisTech/LTCI, Paris, France
{ageneau,boukhatem}@telecom-paristech.fr
[2]Computer Science Dept., UCLA, Los Angeles, CA 90095, USA
gerla@cs.ucla.edu

***Abstract*—Intra-flow network coding was proposed in recent years as a way of enhancing TCP performance over wireless networks. Transmission reliability is improved by sending redundant coded packets instead of retransmitting data packets. In this paper, we discuss the complex issue of TCP interaction with network coding and identify how network coding hides indistinctively link and congestion losses from TCP. Thus, some congestion losses do not trigger TCP congestion window reduction mechanisms. We then focus on fairness issues between coded and non-coded flows. As coded flows are less sensitive than non-coded flows to congestion losses, it prevents them from being as good as non-coded ones at reacting to congestion, making them greedier. We compare the performance of competing Pipeline-coded and non-coded flows in a bottleneck topology. In order to evaluate the fairness of coded flows, we introduce a new fairness index, given that coded flows normally perform better on lossy links without necessarily impacting non-coded flows. Our results show that unfairness exists, but it does interestingly not impact highly, because even with relatively high redundancy factors, non-coded flows do not starve. It shows that congestion losses are correlated enough to enable TCP over coding to react to the signal.**

INTRODUCTION

The Transmission Control Protocol (TCP) performs poorly when used over lossy links, like wireless links, since TCP interprets packet losses as congestion signals [12]. Recently, new approaches using network coding have emerged to deal with losses in wireless networks, like Pipeline Coding [2] or TCP/NC [10]. As a form of erasure code, they aim at masking link losses from TCP by adding redundancy and thus allowing its performance to be improved.

This research is partially funded by the Direction Générale pour l'Armement (French Government Defense procurement agency).

We believe that one of the big challenges when implementing intra-flow network coding consists in minimizing the impact it can have on legacy non-coded traffic; this is a prerequisite for incremental deployment. Since TCP is by far the most widely used transport protocol on the Internet, we focus on the case of TCP interaction with network coding.

As adding a network coding layer tends by design to conceal network characteristics from TCP, unwanted side effects can appear. TCP uses the well-known AIMD congestion control algorithm which, in most implementations, uses packet losses as a congestion signal. However, intra-flow network coding, like pipeline coding [2] [10], is a good way to mask link losses but potentially also congestion losses from the upper layer, and thus making TCP flows over network coding greedier.

In this article, we highlight the potential fairness issue caused by coded flows insensibility to losses and we compare the performance of competing Pipeline-coded and non-coded TCP flows in a bottleneck topology. We introduce a specific index to precisely evalutate the fairness of coded flows, given that they normally perform better on lossy links without necessarily impacting non-coded ones.

I. RELATED WORK

When using intra-flow network coding, for instance Batch coding, the source node sends random linear combinations for each batch of outgoing packets. To compensate losses, more combinations than original packets, *i.e.* degrees of freedom, are sent, according to a constant or adaptive redundancy factor. At destination, Gauss-Jordan elimination is performed and all packets can be recovered if enough independant linear combinations have been received. Depending of the coding implementation, intermediary nodes may recode packets or just forward them.

Pipeline coding [2] is an enhancement of Batch coding. A slightly enhanced version of it is part of ComboCoding [1], an approach that combines the benefits of intra-flow coding with inter-flow coding. The main advantage of Pipeline over Batch coding is its reduced coding delay. Data from the upper layer is sent immediately in the current generation whereas with batch coding it has to be buffered and to wait for the current batch to be completed. When the coding window grows too large, a new empty one for a new generation is created. This technique reduces delays, as they do not depend on block or generation size, thus enabling the use of TCP on top of coding without improperly triggering timeouts. Therefore, it is a more adapted approach for real-time interactive or multimedia sessions.

Since TCP is the most used transport protocol, there are numerous attemps to specifically integrate it with network coding. TCP/NC [10] introduces a layer between IP and TCP that implements Pipeline-like network coding over IP and tricks TCP mechanisms to produce desired results. The source sends linear combinations of all packets in the congestion window, in a pipeline coding way, and the receiver does not actually acknowledge decoded packets but only degrees of freedom in the form of *seen* packets. TCP/NC translate random losses as longer RTT, therefore the losses are masked and the lossy behaviour of the link appears to both ends as a virtual queueing delay, thus interacting well with TCP Vegas. It has been demonstrated TCP/NC is able to react properly to congestion because congestion losses are not independant but correlated, as shown in its model [6]. However, this study only considers two TCP/NC flows in competetion, and not TCP/NC competing with regular TCP flows.

These protocols have been proven fair when two coded flows of the same type are competing, but the possible fairness issue when competing with non-coded flows, caused by coded flows being less sensitive to losses, has been more or less left aside.

The underlying issue here is the complex problem of distinguishing link losses and congestion losses. Different approaches have been attempted to solve the problem in the case of non-coded flows. An estimation of RTT can be directly used like in Non-Congestion Packet Loss Detection (NCPLD) [9], with a threshold to assume congestion loss. TCP NewReno-LP [7] and Veno [3] and TCP Vegas's mechanism to estimate queue size, and assume congestion when queue size is high enough. Packet jitter also carries information about congestion state. Jitter-based TCP (JTCP) [11] computes the average of the inter arrival jitter during one RTT, which is proportional to the queueing speed. When 3 duplicate ACKs are received, if the estimated queueing speed is less than the sending speed, then congestion is assumed.

For coded flows, an enhancement of TCP/NC [8] has been proposed to weaken the redundancy when congestion is probable using a loss differentiation scheme based on the Vegas algorithm. CTCP [5] takes a different approach by introducing a whole new coded transport protocol, with its own new coding-aware block-based congestion control based on RTT: the higher the RTT, the more the coding window is decreased when a loss occur. This radical approach comes at the cost of a difficult deployment, as it aims at replacing TCP.

In this paper, we focus on the fairness issue as a first step of our study related to TCP and network coding interaction.

II. Sensitivity to losses and unfairness

A. Link and congestion losses

Because losses are indistinctively hidden from TCP by pipeline coding, chances are some part of the congestion losses do not actually trigger TCP window reducing mechanism. This situation get worse when the code redundancy factor increases. Verifying this claim is pretty simple. After setting up a simple bottleneck topolgy with 4 nodes and no link losses (figure 1) in the *ns-3* network simulator, we monitor packet drops on queue overflow, which correspond to congestion losses, and we compare the window evolution between TCP NewReno without coding and TCP NewReno over coding in an extreme situation, with generation size $g = 64$ and redundancy ratio $R = 1.25$. In our implementation, all intermediary nodes drop redundant packets and recode innovative ones with redundancy R, so on each link the actual code redundancy is R, independently of the loss rate on preceding links.

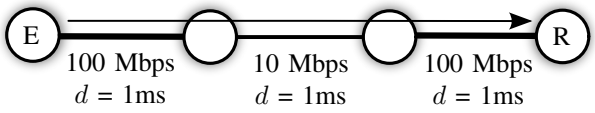

Fig. 1. Simulated bottleneck topology

On figure 2 are represented the evolutions of congestion windows, and vertical bars mark the times when a packet is dropped because of queue overflow, *i.e.* when a congestion loss occurs. The observable behaviour of TCP congestion control mechanisms is radically different between the coded case and the non-coded case. Whereas TCP without coding only need one congestion loss to detect the congestion (by receiving duplicated ACKs) and to react by dividing its window, it needs

dozens of them to react over coding, as losses are masked from TCP, therefore congestion detection needs an entire coding generation to be lost, and it can happen only when more than $(R-1)g = 16$ packets are lost. For this reason, overcoded flows are slower to react to congestion than non-coded ones.

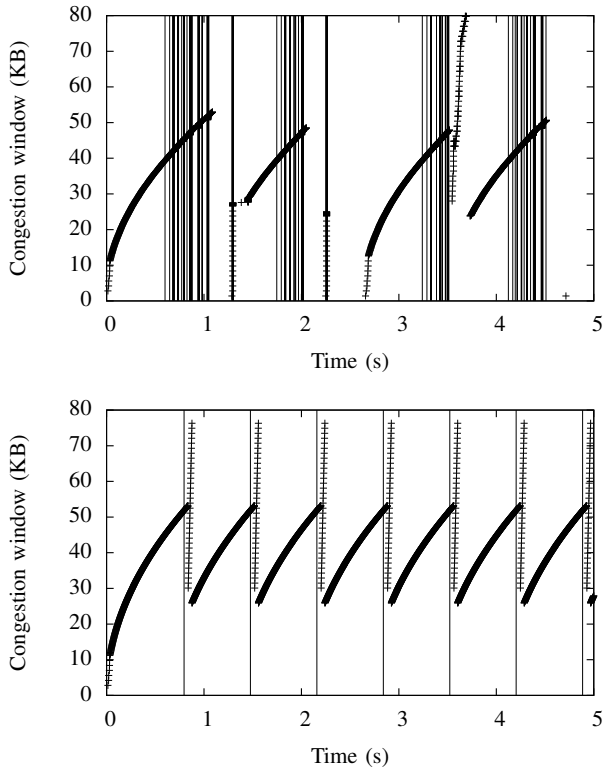

Fig. 2. Congestion window with (top) and without (bottom) coding

One issue is that some overcoding is required, as redundancy should not be set as low as $R_{\min} = \frac{1}{1-p}$ (where p is the average measured loss rate) in order to compensate the losses on average, some more redundancy is necessary to avoid losing generations that statistically encounter more losses or to accommodate a slight decrease in link quality. For instance, ComboCoding's authors use in their redundancy adaptation algorithm a ratio $R = 1.4 + \frac{1}{1-p}$ [1]. A higher redundancy value can lead to coded TCP not reacting well to congestion.

B. Case of competing flows

Let's consider two flows competing for the bandwidth on a network. The first one is TCP running over pipeline coding, and the second one is a not coded TCP flow. Two different factors hinder the non-coded flow and prevent it from using its whole share of available throughput :

- Link losses are hidden from the coded flow but not from the non-coded one. Threrefore, the congestion control of the second flow interprets them as congestion losses : random losses on a link also result in duplicated ACKs or timeouts, as a result the TCP sender adjusts its window size as if a congestion was present on the network. TCP over coding does not suffer from those losses, so it keeps a larger part of the available throughput.
- Some congestion losses are hidden from the coded flow, but none of them are hidden from the non-coded one, so the second flow is more sensitive to actual congestion in the network and reacts faster, leading to a larger part of the throughput for the first flow.

The first mechanism is desired, as it is a simple result of TCP working better over coding, but the second one is clearly undesired. We want to measure which part from the better performance of a coded flow comes from its lower sensibility to link loss and which part comes from higher sensibility of concurrent flows to congestion.

We simulate a simple cross topology (figure 7) with the simulator *ns-3*.

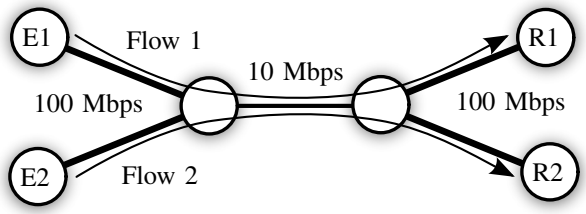

Fig. 3. Simulated bottleneck topology. All links have 1 ms delay and uniform loss with probabilty p.

While keeping an uniform loss probability $p = 1\%$, we run data transfer across two TCP flows, one with coding and one without. Note throughout this article, the loss probability p is not expressed at a physical level, but after retransmissions on the MAC layer, so p is the actual average loss rate experienced by TCP without coding. The average throughput over 10 minutes is recorded for each of the two flows in situations with a different coding ratio R for pipeline coding. The generation size is constant $n = 16$. Note we measure the throughput at application level, so measured values represent in reality the goodput.

Results show the performance with coding is actually worse than without, when R is very low this can be explained simply : when the flow has not enough redundancy it tends to lose entire generations, causing TCP to timeout rather than detecting a loss by receiving

duplicated ACKs. The difference increase between flows as R increases, the coded flow taking gradually the largest part of the available bandwidth. However, there seems to be a limit and the coded flow does not take it all, even with relatively high coding ratios like $R = 1.5$. It means the correlation of congestion is sufficent to make the coded flow react at some point.

Yet the behaviour does not depend on generation size, as shown on figure 5. Whereas generation size is a key parameter, since it should be set higher enough to not be sensitive to statistically localised losses but low enough to reduce then computational overhead, it is not relevant to flow fairness.

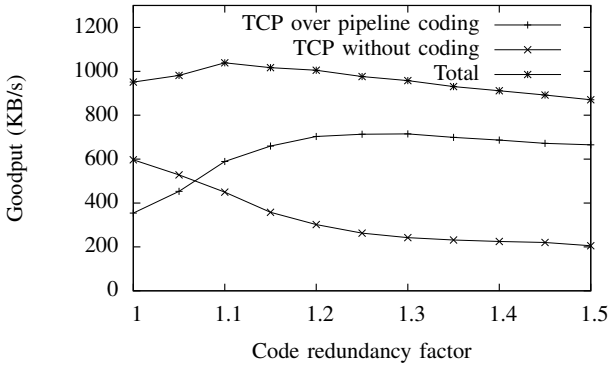

Fig. 4. Goodput comparison between concurrent flows, one with coding at different coding redundancy factors and one without ($p = 1\%$)

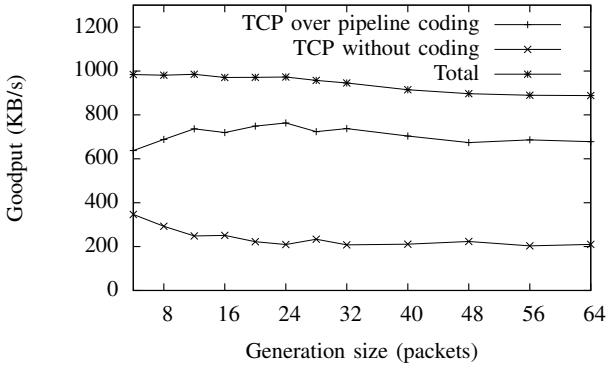

Fig. 5. Goodput comparison between concurrent flows, one with coding at different generation sizes and one without ($R = 1.25$, $p = 1\%$)

Running the two flows over different loss rates (figure 6) shows without surprise the throughput for the non-coded flow decreases when losses increase whereas the one of the coded flow reach a maximum as the other flow's throughput decreases. We can see that at a low loss rate, the coded flow leaves some room to the other one. However, it is not clear if part of its throughput is taken at the expense of the non-coded one.

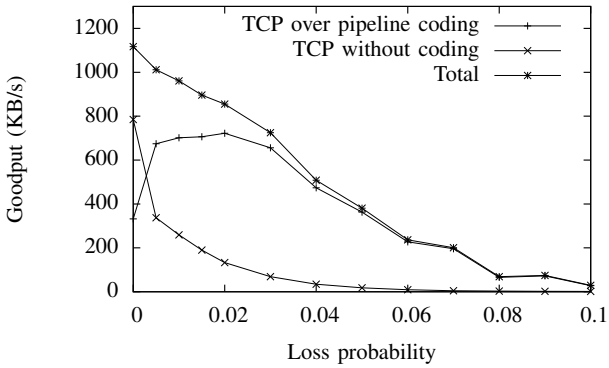

Fig. 6. Goodput comparison between concurrent flows at different loss rates ($g = 16$, $R = 1.25$)

III. Measuring fairness

Our goal is to obtain a measure that is sensitive to the throughput that coded flows unfairly take at the expense of not coded ones, but not sensitive to coded flows performing better without impacting not coded ones. Figure 7 shows graphically in a simple case with two flows the part of throughput we consider as unfairly taken : in the second case, with two non-coded flows, the whole capacity is not used because links are too lossy, whereas in the first case, the coded flow may get more bandwidth by using the whole capacity, but what is taken from the first non-coded flow is considered unfair.

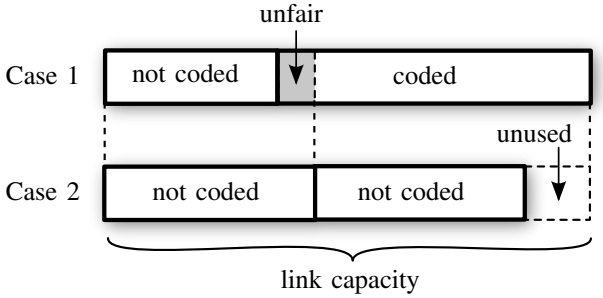

Fig. 7. Comparison of capacity utilization on a lossy link between a first case with a non-coded and a coded flow, and a second case with two non-coded flows

To solve this issue, we introduce a simple modified fairness index taking into account the better performance of coded flows over non-coded ones.

A. Jain's fairness index

Let's consider n flows, x_i being the throughput of the ith flow. Jain's fairness index J [4] rates the fairness of this allocation with a value between $\frac{1}{n}$ (worst case) and 1 (best case, all users receive the same allocation).

$$J(x_1, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

The best case for J corresponds to a uniform allocation, but a uniform allocation is not neccessarily optimal with coded and non-coded flows. The index reflects the actual throughput difference and does not take into account the potentially better performance of coded flows. Therefore, we need to define an ideal *fair allocation* taking into account that coded flows can perform better without impacting non-coded ones.

B. Fair allocation with coded and non-coded flows

For a given set of flows, we define a *fair allocation* as an allocation where :

- The allocation for the subset of coded flows is fair according to Jain's index.
- The allocation for the subset of non-coded flows is fair according to Jain's index.
- No non-coded flow can get less throughput than it would get if coded flows were replaced with non-coded ones.

Intuitively, a fair allocation as defined previously is not necessarily fair according to Jain's index, as a coded flow can get more throughput than a non-coded one, but in a fair allocation, no coded flow can get more throughput at the expense of non-coded ones.

C. Formalization

Let A be an allocation with n flows competing on a lossy path, the first k are coded and the others $n-k$ are not coded ($k > 0$). x_i is the throughput the flow i get in the allocation A.
Let A' be an allocation with the same characteristics but where all coded flows are replaced with non-coded ones. x'_i is the throughput the flow i get in the allocation A'

The capacity c is used in the allocation A. We assume here that the presence of at least one coded flow means more bandwidth is used compared to the allocation A', as coded flows should perform better on lossy links.

$$c = \sum_{i=1}^{n} x_i \geqslant \sum_{i=1}^{n} x'_i$$

Let y_i be the throughput the flow i should get in a fair allocation according to III-B.

A non-coded flow can get less throughput than a coded flow, but it remains fair if every non-coded one gets the same absolute throughput as it would if coded flows were replaced with non-coded flows. It means $\forall i \in \{k+1, ..., n\}\ y_i = x'_i$.

Because of the fairness of TCP congestion control, as we consider identical network charasteristics for every flow (RTT, losses...), we get :

$$J(x'_1, ..., x'_n) = 1$$
$$\forall i \in \{1, ..., n\}\ x'_i = x'$$

So the share of every non-coded flow in the fair allocation should be:

$$\forall i \in \{k+1, ..., n\}\ y_i = x'_i = x'$$

Coded flows should get the same share of the fair allocation :

$$\forall i \in \{1, ..., k\}\ y_i = y$$

The whole available capacity should be used :

$$\sum_{i=1}^{n} y_i = ky + (n-k)x' = c$$

So the share of every coded flow in the fair allocation should be:

$$\forall i \in \{1, ..., k\}\ y_i = y = \frac{1}{k}(c - (n-k)x')$$

In particular if $n = 2$ and $k = 1$, it simply means the coded flow should use the whole bandwith that would remain if it was not coded:

$$y_1 = c - x'$$
$$y_2 = x'$$

D. Modified fairness index

We have seen in the previous paragraph that, in a fair allocation, a coded flow should get $\frac{1}{k}(c-(n-k)x')$ when a non-coded flow should get x', x' being the average throughput a flow would get in the same situation if coded flows where not coded. Let's define λ the ratio between the throughput of a non-coded flow and the one of a coded flow in a fair allocation :

$$\lambda = \frac{kx'}{c - (n-k)x'}$$

If we multiply the throughputs $y_1, ..., y_k$ of the coded flows by a factor λ, Jain's index for the fair allocation becomes 1 :

$$J(\lambda y_1, ..., \lambda y_k, y_{k+1}, ..., y_n) = 1$$

So we introduce a modified fairness index J' taking this correction into account, *i.e.* the throughputs $x_1, ..., x_k$ are multiplied by λ :

$$J'(x_1, ..., x_n) = \frac{(\lambda \sum_{i=1}^{k} x_i + \sum_{i=k+1}^{n} x_i)^2}{n(\lambda^2 \sum_{i=1}^{k} x_i^2 + \sum_{i=k+1}^{n} x_i^2)}$$

This modified fairness index has the same properties as Jain's fairness index, but the best case corresponds to a fair allocation as defined in III-B and not a uniform allocation. It rates the fairness with a value between $\frac{1}{n}$ (worst case) and 1 (best case, the allocation is fair).

In particular if $n = 2$ and $k = 1$:

$$\lambda = \frac{x'}{c - x'}$$

$$J'(x_1, x_2) = \frac{(\lambda x_1 + x_2)^2}{2(\lambda x_1)^2 + 2x_2^2}$$

Example : let's consider a coded flow competing with a non-coded flow on a lossy link. The measured throughputs are $x_1 = 7$ Mbps for the coded flow and $x_2 = 3$ Mbps for the non-coded flow, so $c = 10$ Mbps. A similar experiment but with 2 identical non-coded flows give us $x'_1 = 4$ Mbps and $x'_2 = 4$ Mbps, so $x' = 4$ Mbps and $\lambda = \frac{2}{3}$. We eventually get $J'(x_1, x_2) \simeq 0.95$ (nearly fair) whereas $J(x_1, x_2) \simeq 0.86$.

Applying this new index to the previously measured throughputs gives us a more precise idea of the actual flow fairness. Note computing this index also requires running non-coded TCP flows in the same configuration, as their average throughput is required to compute λ.

Figure 8 highlights the allocation is actually fairer in reality than what Jain's index shows, as the coded flow takes a part of the resource the non-coded flow could not take anyway. As a side effect, the maximum fairness is displaced to higher redundancy (here roughly from 1.05 to 1.1). In the case of higher loss rate (figure 9), the two indices give rather different results. This is caused by Jain's index not taking into account the very poor performance of TCP at important loss rates, whereas the modified index does. We can see that at some point, when losses are too important, the non-coded flow performs so badly that it is actually fair for the coded flow to take an overwhelming part of the throughput.

When R approaches higher values, fairness does not drop and stays relatively high around 0.85. This is a good sign, as it indicates even with too much redundancy, coded TCP flows do not starve non-coded ones. It means congestion losses are correlated enough to trigger a generation loss and TCP congestion control algorithm.

It is interesting to note that the maximum fairness is achieved when $R = 1.1$, and it corresponds to the session giving maximum cumulated throughput. This is logical in the sense that fairness is achieved when the two flows give simultaneously the best possible performance. A maximum fairness is achieved for $R = 1.25$ near $p = 0.02$ for similar reasons.

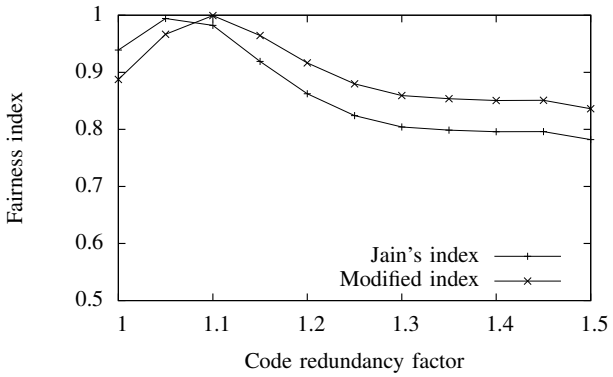

Fig. 8. Jain's index at different coding factors for a coded and a non-coded flow ($p = 1\%$)

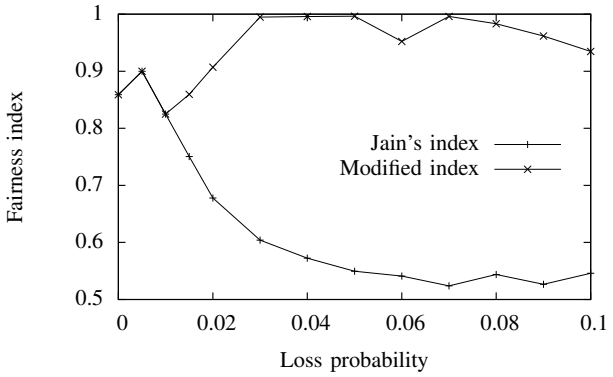

Fig. 9. Jain's index at different loss rates for a coded ($g = 16$, $R = 1.25$) and a non-coded flow

CONCLUSION

In this paper, we highlighted the fairness problem when Pipeline coding is implemented underneath TCP. This issue arises because concealing link losses can interfere with TCP congestion control by also masking congestion losses. We indroduced a simple specific index to more precisely evalutate the fairness of coded flows by taking into account their better performance on lossy links. Our results shows that unfairness exists but its impact is relatively limited, because even with

high redundancy factors a coded flow does not starve a non-coded one, indicating that congestion losses are correlated enough to cause the coding to fail and TCP to react to the congestion.

This investigation is carried on by considering the effect of redundancy adaptation algorithms on fairness.

REFERENCES

[1] Chien-Chia Chen, Clifford Chen, Soon Y. Oh, Joon-Sang Park, Mario Gerla, and M.Y. Sanadidi. ComboCoding: Combined intra-/inter-flow network coding for TCP over disruptive MANETs. *Journal of Advanced Research*, 2(3):241 – 252, 2011.

[2] Chien-Chia Chen, Soon Y. Oh, Phillip Tao, Mario Gerla, and M. Y. Sanadidi. Pipeline network coding for multicast streams. In *Proceedings of the International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2010)*, Seattle, USA, April 2010.

[3] C. P. Fu and S. C. Liew. TCP Veno: TCP enhancement for transmission over wireless access networks. pages 216–227, February 2003.

[4] Rajendra K. Jain, Dah-Ming W. Chiu, and William R. Hawe. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. Technical report, Digital Equipment Corporation, September 1984.

[5] MinJi Kim, Jason Cloud, Ali ParandehGheibi, Leonardo Urbina, Kerim Fouli, Douglas J. Leith, and Muriel Médard. Network coded TCP (CTCP). *CoRR*, abs/1212.2291, 2012.

[6] Minji Kim, Muriel Médard, and João Barros. Modeling network coded TCP throughput: a simple model and its validation. In *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '11, pages 131–140, ICST, Brussels, Belgium, Belgium, 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[7] Fabio Martignon and Luigi Fratta. Loss differentiation schemes for TCP over wireless networks. In *Proceedings of the Third international conference on Quality of Service in Multiservice IP Networks*, QoS-IP'05, pages 586–599, Berlin, Heidelberg, 2005. Springer-Verlag.

[8] Hamlet Medina Ruiz, Michel Kieffer, and Béatrice Pesquet-Popescu. An adaptive redundancy scheme for tcp with network coding. In *IEEE International Symposium on Network Coding (NETCOD)*, United States, 2012.

[9] N. K G Samaraweera. Non-congestion packet loss detection for TCP error recovery using wireless links. *Communications, IEE Proceedings*, 146(4):222–230, 1999.

[10] J.K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros. Network coding meets TCP. In *INFOCOM 2009, IEEE*, pages 280–288, 2009.

[11] E. H.-K. Wu and Mei-Zhen Chen. JTCP: jitter-based TCP for heterogeneous wireless networks. *IEEE J.Sel. A. Commun.*, 22(4):757–766, September 2006.

[12] G. Xylomenos, G.C. Polyzos, P. Mahonen, and M. Saaranen. Tcp performance issues over wireless links. *Communications Magazine, IEEE*, 39(4):52–58, 2001.