Throughput vs. Delay in Lossy Wireless Mesh Networks with Random Linear Network Coding

Hundebøll, Martin; Pahlevani, Peyman; Roetter, Daniel Enrique Lucani; Fitzek, Frank Hanns Paul

Published in:
European Wireless 2014; 20th European Wireless Conference; Proceedings of

Publication date:
2014

Document Version
Peer reviewed version

Throughput vs. Delay in Lossy Wireless Mesh Networks with Random Linear Network Coding

Martin Hundebøll, Peyman Pahlevani, Daniel E. Lucani, Frank H.P. Fitzek
Department of Electronic Systems
Aalborg University, Denmark
Email: {mhu,pep,del,ff}@es.aau.dk

***Abstract*—This work proposes a new protocol applying on–the–fly random linear network coding in wireless mesh networks. The protocol provides increased reliability, low delay, and high throughput to the upper layers, while being oblivious to their specific requirements. This seemingly conflicting goals are achieved by design, using an on–the–fly network coding strategy. Our protocol also exploits relay nodes to increase the overall performance of individual links. Since our protocol naturally masks random packet losses from the upper layers, it makes it particularly suitable for enhancing TCP's performance in wireless mesh networks, where packet losses are typically interpreted as a sign of congestion by TCP's congestion control algorithms, thus crippling TCP's throughput. To investigate the gains and downsides of our protocol, we implement it as a configurable proof-of-concept application, which is deployed and evaluated in a real test bed with Raspberry Pi devices. We show that order of magnitude gains in throughput over plain TCP are possible with moderate losses and up to two fold improvement in per packet delay in our results.**

I. Introduction

Enhancing performance of wireless mesh networks has been an important research area due to the potential to extend coverage, adapt to changes in topology by using a distributed mechanism, and the reduced infrastructure required to support it. However, it has also brought a series of challenges to guarantee reliability, high throughput, and low delay for packets routed through the system. This is particularly important if we consider that *de facto* standards, such as TCP, perform badly on wireless networks due to the presence of random packet losses. The reason is that TCP will interpret these as a sign of congestion and thus back off [1]–[3]. This back off mechanism is quite aggressive, rendering TCP almost unusable for moderate to high end-to-end packet losses, i.e, above 10%. Several approaches have been proposed to improve the reliability of the lossy links, e.g., [4]–[7].

In recent years, network coding has been proven to be pivotal in addressing the performance problems of TCP in networks with random packet losses [8]–[10]. Network coding focuses on transmitting linear combinations of data packets that can be recoded at intermediate nodes instead of simply forwarding data packets. In particular, Random Linear Network Coding (RLNC), provides a distributed and efficient approach to create coded packets by choosing the coefficients of the linear combinations of blocks of packets uniformly at random from the elements of a finite field. From a practical perspective, one of the primary benefits of network coding is its rate less property and its ability to provide reliability with only little overhead from signaling and coordination, thus reducing the costs of reliability. In [11], it is proposed to integrate RLNC in the TCP stack using an online coding strategy, and the approach is evaluated with respect to throughput in [12]. However, these solutions have been tailored specifically to TCP and not compatible with other approaches. One of our goals is to provide a reliable layer that can be used by a variety of protocols, including TCP.

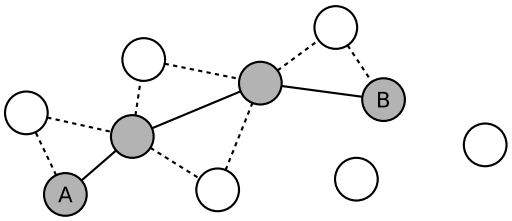

Fig. 1. A first step towards multipath routing is segment-wise multipath, where each hop benefits from multiple links.

Another key goal of our work is to provide increased performance even in the presence of highly lossy links along the communication path. To achieve this, we are inspired by the work in [13], which proposed a scheme for utilizing overhearing in off-path nodes (called helpers) in a mesh network to improve performance by reducing the number of transmissions in the system. Figure 1 illustrates how a multi-hop path in a wireless mesh network can include neighbors as part of the path, thus forming a *path corridor*, composed of small three-node multipath elements. RLNC removes the need for packet scheduling, as the forwarding node (the encoder) transmits linear combinations of packets, from which the helper node can create new independent combinations and help the next-hop (the decoder) to decode the received packets. The work in [13] presents how link quality estimates can be used to estimate how many packets the encoder and helper should transmit, and when the helper should step in.

In the following work we enhance the ideas presented in [13] by (i) adding a minimum of signalling between the nodes to handle a continuing flow of packets, (ii) leveraging an on-the-fly coding strategy to maintain low delay for packet transmissions, and (iii) providing and implementation and demonstration in real systems. Our protocol is described in detail in Section II, while the implementation of our proof-of-concept in Raspberry devices and setup are described in Section III and Section IV, respectively. We provide extensive

measurement results transporting both UDP and TCP traffic over our protocol and comparing it with plain implementations of UDP and TCP under a variety of network conditions in Section V. Our results show that order of magnitude gains over plain TCP are possible even for moderate packet losses and without trying to adapt specifically to TCP behavior as previous work focused on. Finally, we conclude on the results in Section VI.

II. Protocol Description

The Fast, Reliable, And Network Coded (FRANC) protocol is composed from different building blocks. The raw encoding, recoding, and decoding is handled by the Kodo framework [14], the budgets used by the encoder and helper nodes are derived from the PlayNCool algorithms [13], while the simple managing of coding blocks is the primary task for FRANC.

A. Budgets

The core element of FRANC is the utilization of the PlayNCool algorithms from [13]. The algorithms estimate, based on the error probabilities, the number of encoded symbols to transmit in order for the decoder to receive enough symbols to decode. By including a helper node as illustrated in Figure 2 in the system, the transmissions needed to decode a block can be reduced, depending on the given error probabilities.

The error probabilities needed to derive the estimate, can be extracted from the data-link layer. Implementations, such as the Minstrel rate control system, or the B.A.T.M.A.N. and OLSR mesh routing protocols, all maintain estimates of the error probabilities towards neighbors in the network.

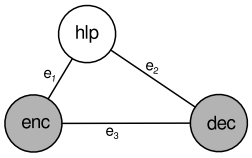

Fig. 2. The PayNCool budgets are based on three link metrics (e_1, e_2, e_3) describing the error probability of each link.

B. Parameters

FRANC is tuned by a set of parameters, with *field size*, *block size*, *symbol size*, *encoder timeout*, and *budget overshoot* being the most significant ones. We explain these parameters in the following:

1) Field Size: For low computational complexity when encoding/decoding symbols, binary field size should be chosen. A field size of 2^8 reduces the probability of generating linear dependent encoded symbols, at the cost of more overhead and higher computational complexity when encoding/decoding.

2) Block Size: The block size influences the length of the Kodo header, the computational complexity for encoding/decoding, and the overall cost of protocol signaling. It should be selected as large as possible with respect to the available MTU on the wireless link.

3) Symbol Size: The symbol size, together with the block size, determines the resulting packet size, which includes room for the encoded symbol, the Kodo header, and the FRANC header as illustrated in Figure 3.

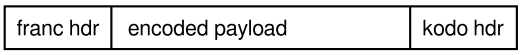

Fig. 3. Structure of encoded packets in FRANC. First a FRANC header identifying the ENC type descibed below, followed by the encoded symbol generated with Kodo, and finally the Kodo header which contains the encoding coefficients.

4) Encoder Timeout: This controls the time the encoder should wait for an acknowledgement from the decoder before sending additional encoded packets, similar to the SIFS in the IEEE 802.11 standards [15].

5) Budget Overshoot: The overshoot parameter adds a fixed redundancy ratio to the encoder budgets, to reduce the impact of variance in the estimated error probabilities. A high overshoot ratio (e.g. 1.10), reduces the number of encoder timeouts at the cost of more linear dependent packets.

C. Packet Types

One of the valuable characteristics of RLNC is the little need for organization between nodes in wireless networks, and FRANC benefits from this by using only two packet types and not more than three states.

The primary packet type is ENC, which is sent by the encoder/recoder, and carries encoded symbols generated by Kodo. After receiving enough encoded packets to either partially or fully decode a block, the decoder acknowledges this by sending an ACK packet. These contain feedback from the decoder including current rank and a bitmap of decoded symbols.

The helper node overhears both ENC and ACK packets from the encoder and decoder, respectively, and sends recoded ENC packets according to the behaviour described by PlayNCool in [13]. Every packet sent with FRANC has a header with a type ID and a block ID.

D. Protocol States

The encoder resides in either of the three states illustrated in Figure 4: ENC, WAIT, and IDLE. When in the former state, the encoder increases its budget for every packet added to the block, and spends it again by sending encoded packets, after which the state is changed to WAIT. In this, the encoder waits for the configured timeout, and if no ACK packet is received, the budget is incremented and spent, similar to when a symbol is added in the ENC state.

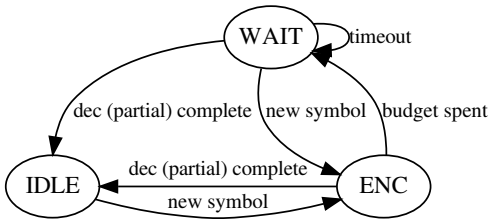

Fig. 4. Possible states and transitions of the encoder.

When receiving an ACK packet, the encoder compares the included rank with number of added symbols. If the decoder rank is less than the number of added symbols, nothing is

changed. If the rank is equal, and the number of added symbols is less than the block size, the state changes to IDLE. When the decoder rank equals the block size, the encoder increments the block ID by one, and the state changes to IDLE. When being in IDLE, and new symbols are added to the encoder, the state changes to ENC.

The decoder is simply adding packets as they arrive and sends ACKs whenever partially or fully decoded. In case of the latter, the block count is incremented by one.

The helper can be in one of the two states illustrated in Figure 5: IDLE or ENC. When in IDLE, the helper is adding overheard symbols to its recoder, but not sending recoded packets. When the number of added symbols exceeds the threshold determined by PlayNCool, the state changes to ENC, and the budget is increased and spent for every added symbol. Whenever the maximum budget is spent, or a packet with an new block ID is received, the state changes to IDLE.

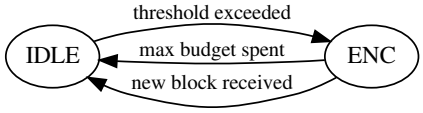

Fig. 5. Possible states and transitions of the helper.

III. Implementation

The implementation of FRANC aims at providing a proof-of-concept reference, as well as the means to evaluate and improve the protocol in the test setup described Section IV. It is built with the Netmix Networking Layers [16], and uses the Kodo network coding software library from [14].

To allow the use of Kodo and to support maximum flexibility, the implementation is built as a user space application using raw Ethernet sockets on the wireless links. To avoid the overhead of generating and processing data on the same device as the encoding and decoding takes place, the implementation uses a client application on a separate device to generate and deliver data. This setup is illustrated in Figure 6.

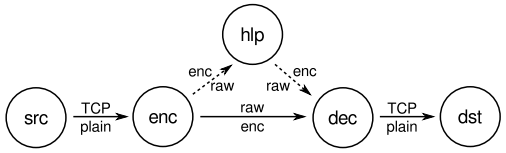

Fig. 6. The flow of data from the generating device, through the encoder, (and possibly helper), decoder, to the destination device, where it is delivered.

When testing unreliable communication, i.e. no feedback or reliability mechanism at network or application layer, a similar user space application is used to introduce the losses. As illustrated in Figure 7, this simply reads data from the client source and forwards it as raw Ethernet frames to the next-hop, from where it is forwarded to the destination client and delivered.

Both the encoding/decoding and the forwarding applications support two-way traffic flows. They can be configured to introduce synthetic packet losses with different probabilities

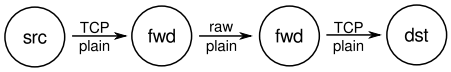

Fig. 7. The flow of data from the generating source device, through the forwarding nodes, to the delivering destination device.

for different devices. The client applications on the source and destination devices uses a virtual network interface to support both TCP/IP and UDP/IP traffic from any regular user space application.

IV. Test Setup

The test setup consists of a laptop to generate/deliver data, and three Raspberry Pi (RPi) devices equipped with USB Wifi dongles as described in [17]. Tests are performed by running the `nuttcp` network benchmarking tool [18] in UDP or TCP mode, depending on the test case. Both the `nuttcp` server and client is executed on the laptop, and `iptables` is used to force packets to be delivered to the virtual interfaces instead of the loopback interface. This approach, as illustrated in Figure 8, eliminates the need for clock synchronization when measuring end-to-end delays, since the sender and receiver are referencing the same clock.

Due to the computational overhead of doing encoding/decoding in user space, and the limited CPU resources on the RPi devices, the maximum rate of the wireless network is limited to 12 Mbps, making the tests bound by link speed instead of CPU resources.

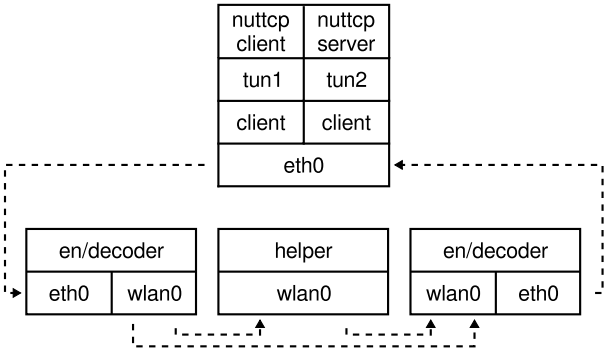

Fig. 8. Simplified network/software stacks for each device in the test setup.

The data flow in Figure 8 originates from the `nuttcp` client and enters the virtual `tun1` interface, where it is read by the left side client application. Data from the virtual interface is tunneled through a TCP connection to the encoder/decoder application on the left side RPi, where it is added to the block in the encoder, and transmitted as encoded symbols in raw Ethernet packets to the right side RPi. Once decoded, the data is again tunneled to the right side client application on the laptop, where it is written to the virtual `tun2` interface and thus received by the `nuttcp` server. The wireless interface on the helper device uses promiscuous mode to overhear packets from the to encoding/decoding devices, which enables the use of unicast packets with rate adaptation and link level acknowledgements. TCP acknowledgements from the `nuttcp` server travels the reverse direction.

A. Tests

All tests are conducted with first an unreliable link, then with FRANC without a helper, and finally with FRANC with a helping node. To minimize the impact of changes in the surrounding wireless networks, each test runs for 20 seconds and is repeated ten times in an interleaved fashion for every parameter configuration. The following values are selected for the static parameters:

- Field Size: Binary
- Block Size: 150 symbols
- Symbols Size: 1450 bytes
- Timeout: 20 ms
- Overshoot: 5%

Three kinds of tests are conducted:

1) UDP throughput measurements for varying e_3: The error probability e_3 is increased in steps of 10 percentage points, while the two other error probabilities (e_1, e_2) grows accordingly by 5 percentage points. The offered end-to-end load on the system is 10 Mbps, which is enough to congest the wireless link.

2) UDP rates for changing e_1 and e_2: To evaluate the performance of FRANC with different values of e_1 and e_2, UDP tests are conducted with e_3 varying from 10% to 90% in steps of 10. Each sweep is tested for three sets of (e_1,e_2): (10%, 50%), (30%,3%), and (50%,10%).

3) TCP delays and throughputs for varying e_3: While UDP traffic is suitable for measuring raw performance, TCP traffic is the most common type of traffic, and should thus be tested as well. To evaluate the performance of FRANC and TCP flows, tests are conducted for varying errors between 0% and 20%. Errors probabilities above 20% makes TCP practically unusable. `cubic` is used as congestion avoidance algorithm, as this is the default on linux systems.

V. Results

The following results show an initial picture of the potential of FRANC with respect to the performance of plain UDP flows and the more complex TCP flows.

A. UDP Throughput and Transmissions

To evaluate the cost of introducing reliability for the wireless link, the throughput is measured for FRANC vs. a simple UDP flow, where dropped packets are ignored.

Since the unreliable link does no attempt to recover lost packets, it gives the upper bound of the link. On average, the throughput of FRANC without a helper is 83% of the unreliable link. With the helper enabled, the performance is 105% of the unreliable link, with a clear trend of improvement as e_3 increases.

The number of transmission used by FRANC to successfully decode one block is shown in Figure 10. The results show that a for increasing error probabilities on e_3, the helper reduces the total number of transmissions, which increases the throughput as seen in Figure 9, and is beneficial for other nodes in the network, who can use the freed airtime.

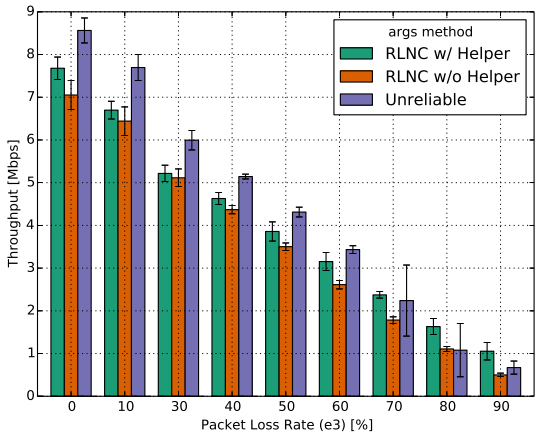

Fig. 9. Throughput as measured on a single hop wireless link with varying synthetic losses.

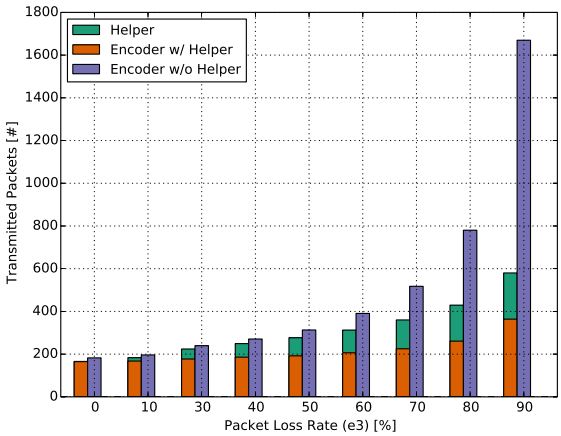

Fig. 10. Number of transmissions per block from encoder needed to successfully decode.

B. Helper Placement

The influence of the helper node is evaluated by changing the error probabilities for e_1 and e_2. Figure 11 shows the UDP throughput for three sets of e_1 and e_2 for a range of e_3 error probabilities, as well as the corresponding throughput for FRANC without a helper and the unreliable link.

As expected, the throughput of FRANC without a helper is on average 82% of the unreliable throughput. With the helper enabled, there is a clear improvement as e_3 increases. The best helper performance is seen with symmetric e_1 and e_2, due to the slightly better combined link quality (49% vs. 45%), and the higher number of encoder timeouts, which is shown in Figure 12. The increased number of timeouts in the asymmetric cases, suggests that the PayNCool budgets have room for improvement in the corner cases.

C. TCP Throughput and Delay

With zero losses, FRANC performs similar to the unreliable link, when the link is not congested. Figure 13 shows that FRANC is suffering of high delays and limited throughput when the offered load exceeds 5000 kbps. This is due to a severe case of bufferbloat in the chain of nodes, where the encoder device is reading from the client application

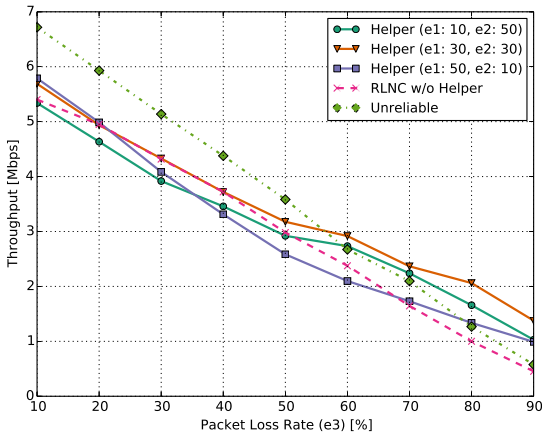

Fig. 11. Throughput for varying error probabilities for e_3 and three different sets of (e_1, e_2).

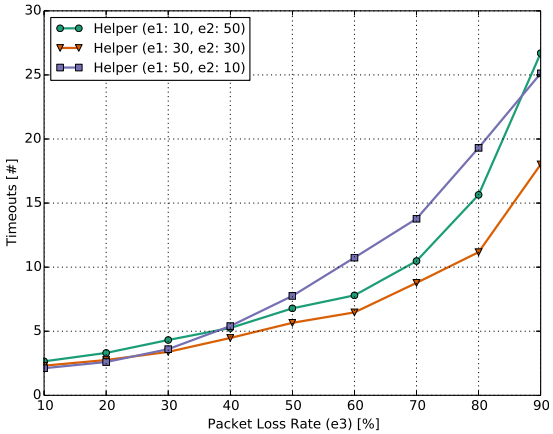

Fig. 12. Timeouts per block for varying error probabilities for e_3 and three sets sets of (e_1, e_2).

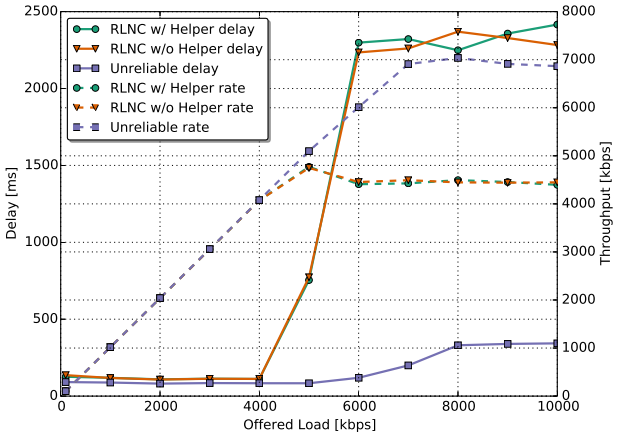

Fig. 13. Delay and throughput for a TCP connection on a wireless link with no error probability.

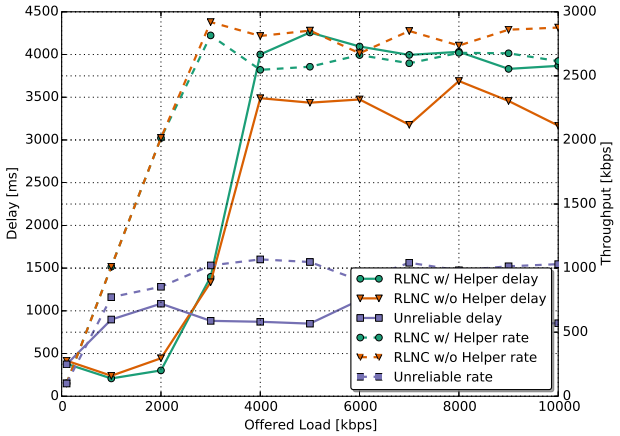

Fig. 14. Delay and throughput for a TCP connection on a wireless link with 10% error probability.

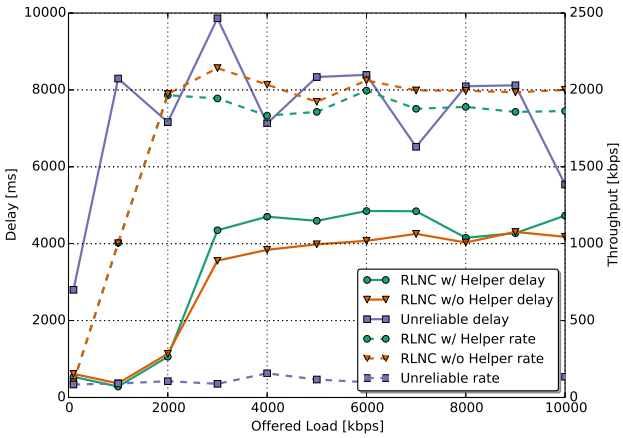

Fig. 15. Delay and throughput for a TCP connection on a wireless link with 20% error probability.

connection in bursts whenever af block is acknowledged. For the unreliable case, the throughput is limited to 7000 kbps, which is close the maximum UDP throughput.

When increasing the error probability of e_3 to 10%, TCP without FRANC misinterprets this as congestion, and backs off unnecessarily. This is seen in Figure 14, where the maximum TCP throughput drops to ∼1000 kbps in the unreliable case, while it is increased to >2500 kbps when enabling FRANC. Delays increase by a factor of two for both the unreliable and the FRANC cases, again showing the effects of bufferbloat when congesting FRANC. The high delays for FRANC can be reduced by changing to delay-aware congestion control algorithms such as TCP Vegas. Initial tests on this already show significant improvements.

For the case of 20% loss on e_3 in Figure 15, TCP throughput drops to less than 200 kbps if running without FRANC, while FRANC is able to maintain a throughput of ∼2000 kbps, with delays decreased by an order of magnitude when compared to the unreliable case. When looking at the delay per bit, as plotted for 10% loss in Figure 16, it is clear that FRANC performs best in the range just below the congested state.

The results from TCP tests show no gain when enabling the helper node, which was expected from the UDP results above. This is explained by the TCP ACKs travelling the inverse direction being limited by fair nature of the MAC. When enabling the helper, three nodes are competing for access, as opposed to only two nodes without the helper.

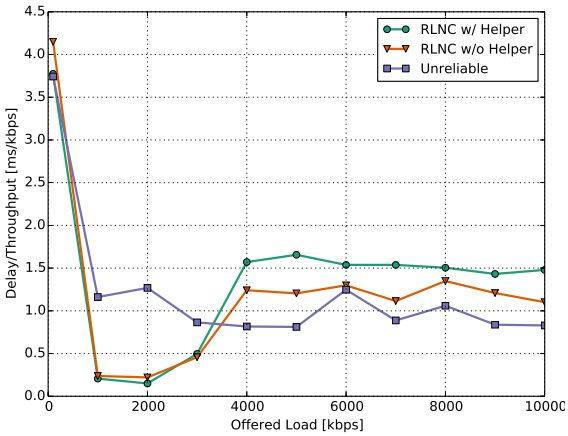

Fig. 16. Delay over throughput for a TCP connection on a wireless link with 10% error probability.

VI. Conclusion

When TCP connections travel on wireless mesh networks with lossy links, the congestion control algorithms suffer from random packet losses. In cases of packet loss rates of more than 5-10%, the performance of TCP degrades and the connection becomes practically useless.

To improve TCP performance in these scenarios, we proposed the Fast Reliable Network Coded (FRANC) protocol, which is an enhancement of the PlayNCool scheme proposed in [13]. However, FRANC is not designed purely for TCP and can support other flow types seamlessly. The protocol utilizes an on-the-fly version of Random Linear Network Coding to let forwarding nodes transmit linear combinations of packets that are decoded at the receiver side. The on-the-fly encoder and decoders allow FRANC to provide both good throughput and delay performance. Furthermore, the use of linear combinations enables overhearing nodes to inject recoded combinations in the link, which reduces the total number of transmissions needed to successfully decode at the receiver side. The ability to recode with random coefficients requires no coordination between nodes, which in turn reduces the needed signalling to a bare minimum and simplifies the overall logic. Thus, FRANC uses only two packet types: encoded packets and acknowledgement packets.

With a proof-of-concept implementation of FRANC, we evaluate its performance with respect to both throughput and delay. Our results show that the increased delay as a result of random packet loss is significant, and we show the FRANC is able to reduce the delay several fold, while maintaining a high throughput. Our measurements show that the throughput can be an order of magnitude higher than plain TCP for moderate packet losses, e.g., 20%.

The proof-of-concept implementation of FRANC suffers from the bufferbloat syndrom, which causes high delays when the connection is congested. Initial tests with alternative congestion control algorithms show that these effects can be reduced dramatically, and further investigation will be carried out for our future work.

The reduced number of transmissions with a helper node brings benefits to the surrounding network, which can utilize the freed airtime to transmit. This effect is expected to grow as more hops are added to the connection. This, together with the benefit of doing recoding in the forwarding nodes, shall be addressed in future work.

References

[1] M. Gerla, K. Tang, and R. Bagrodia, "TCP performance in wireless multi-hop networks," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, Feb 1999, pp. 41–50.

[2] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *INFOCOM 2003. Twenty-second annual joint conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3. IEEE, pp. 1744–1753.

[3] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 275–288, 2002.

[4] S. Rangwala, A. Jindal, K.-Y. Jang, K. Psounis, and R. Govindan, "Understanding Congestion Control in Multi-Hop Wireless Mesh Networks," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*. ACM, 2008, pp. 291–302.

[5] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks," in *MobiCom*, vol. 95. Citeseer, 1995, pp. 2–11.

[6] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks," *Personal Communications, IEEE*, vol. 8, no. 1, pp. 34–39, 2001.

[7] D. Kim, C.-K. Toh, and Y. Choi, "TCP-BuS: Improving TCP performance in wireless ad hoc networks," in *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 1707–1713.

[8] Y. Huang, M. Ghaderi, D. Towsley, and W. Gong, "TCP Performance in Coded Wireless Mesh Networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08. 5th Annual IEEE Communications Society Conference on*, June 2008, pp. 179–187.

[9] P. Samuel David and A. Kumar, "Network coding for TCP throughput enhancement over a multi-hop wireless network," in *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*. IEEE, 2008, pp. 224–233.

[10] L. Scalia, F. Soldo, and M. Gerla, "PiggyCode: a MAC layer network coding scheme to improve TCP performance over wireless networks," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*. IEEE, 2007, pp. 3672–3677.

[11] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros, "Network coding meets TCP," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 280–288.

[12] J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets TCP: Theory and implementation," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 490–512, 2011.

[13] H. Khamfroush, P. Pahlevani, D. E. Lucani, M. Hundebøll, and F. Fitzek, "On the Coded Packet Relay Network in the Presence of Neighbors: Benefits of Speaking in a Crowded Room," *IEEE International Conference on Communications*, 2014.

[14] M. Pedersen, J. Heide, and F. Fitzek, "Kodo: An Open and Research Oriented Network Coding Library," *Lecture Notes in Computer Science*, vol. 6827, pp. 145–152, 2011.

[15] "IEEE Standard 802.11n-2009: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," http://standards.ieee.org/findstds/standard/802.11n-2009.html, 2009, online; accessed 04-feb-2014.

[16] "netmix - A C++ Software Library for Network Application," http://github.com/hundeboll/netmix, online; accessed 05-feb-2014.

[17] A. Paramanathan, P. Pahlevani, S. Thorsteinsson, M. Hundebøll, D. Roetter, and F. Fitzek, "Sharing the Pi: Testbed Description and Performance Evaluation of Network Coding on the Raspberry Pi," *IEEE VTC Vehicular Technology Conference. Proceedings*, 2014.

[18] "nuttcp - Network Benchmarking Tool," http://nuttcp.net, online; accessed 05-feb-2014.