Network Coded Software Defined Networking: Enabling 5G Transmission and Storage Networks

Jonas Hansen, Daniel E. Lucani, Jeppe Krigslund, Muriel Médard, and Frank H. P. Fitzek

ABSTRACT

Software defined networking has garnered large attention due to its potential to virtualize services in the Internet, introducing flexibility in the buffering, scheduling, processing, and routing of data in network routers. SDN breaks the deadlock that has kept Internet network protocols stagnant for decades, while applications and physical links have evolved. This article advocates for the use of SDN to bring about 5G network services by incorporating network coding (NC) functionalities. The latter constitutes a major leap forward compared to the state-of-the-art store and forward Internet paradigm. The inherent flexibility of both SDN and NC provides fertile ground to envision more efficient, robust, and secure networking designs, which may also incorporate content caching and storage, all of which are key challenges of the upcoming 5G networks. This article not only proposes the fundamentals of this intersection, but also supports it with key use cases and a thorough performance evaluation on an implementation that integrated the Kodo library (NC) into OpenFlow (SDN). Our results on single-hop, multihop, and multi-path scenarios show that gains of 3× to 11× are attainable over standard TCP and multi-path TCP.

INTRODUCTION

Communication and networking systems have been structured in a series of layers to ease design and, in principle, allow for novel technologies and services to be incorporated with minimal effort. Although this has been true for higher layers dealing with applications and services (e.g., incorporating multimedia content, social networking, and cloud applications) and lower layers dealing with access technologies (e.g., evolving from early mobile cellular technologies to 4G), the intermediate network layer control protocols have been stagnant for several decades. In fact, this deadlock is due in part to the fact that multiple technologies depend on the current state of the network to operate [1] and in part to the challenges of configuring and managing networks based on network element operations instead of the network services to be provided. The downside to the current network design and protocols is that they were conceived with a fairly narrow set of goals, which is now limiting the effectiveness and feasibility of the more complex and resource demanding applications of the upcoming 5G networks. For example, the lack of a holistic view of the network can translate into reduced efficiency and higher congestion in specific paths or the inability to use multi-path techniques to provide ultra-reliable services. This structure also limits the adoption of verifiably optimal multicast techniques, such as network coding (NC), at the core of the network.

As a way to deal with some of the inefficiencies derived from these drawbacks, networking theory and the Internet are currently facing two large, game-changing trends: software defined networking (SDN) and NC. Each provides a disruptive concept to enable more efficient and flexible networking, and both have experienced a proliferation of academic and industrial applications. The goal of this article is to bridge the two concepts through fundamental analysis and understanding of their joint potential as well as testing their combination with a real implementation. The combination of both ideas goes beyond marrying two buzz words. SDN offers fertile ground to implement NC ideas and make them widely and readily available, as well as to help guide network coding research in terms of realistic restrictions, such as complexity, memory, and in-network routers, as well as routers capabilities (e.g., changing data operations on the fly). In other words, SDN is the key enabler for the timely implementation and deployment of NC, initially as a virtual function next to each communication node and in the future as an integrated part of the SDN system.

This article provides key examples to demonstrate the potential of NC's recoding capabilities and ability to mask losses and stabilize lossy links, even without altering the end-to-end use

Jonas Hansen and Daniel E. Lucani are with Aalborg University. Daniel Lucani is also with Chocolate Cloud ApS.

Jeppe Krigslund is with Steinwurf ApS.

Muriel Médard is with the Massachusetts Institute of Technology.

Frank H. P. Fitzek is with TU Dresden.

This work was financed in part by the Green Mobile Cloud project granted by the Danish Council for Independent Research (Grant No. 10-081621).

0163-6804/15/25.00 © 2015 IEEE

of TCP. This provides a simple but critical stepping stone to improve current systems and open the path to more complex protocols natively centered around network coding, such as coded TCP (CTCP) [2]. We show that our simple approach provides three-fold to 11-fold gains over standard TCP and multi-path TCP [3].

SDN and NC: The Key to 5G Services?

More than providing an evolution in network technologies, 5G networks are envisioned as a revolution. It is not just about increased data rates with a new radio access technology, but rather a large expansion of the network's goals to provide traditional services (e.g., voice, user data), as well as radically new services: machine-to-machine (M2M) communications with support for massive numbers of devices, millisecond latency for communications, cloud and caching services, high reliability, and energy efficiency. This vision requires a system that is able to judiciously allocate resources, treats storage and transport of data as a single process, and exploits the meshed nature of communication networks to guarantee these requirements. This article argues that a combination of SDN and NC is the key to addressing these challenges.

SDN allows for a network to perform flexible resource allocation (e.g., buffer management, dynamic routing, exploitation of multiple paths) beyond a single layer in the network stack (Fig. 1). To achieve this, SDN virtualizes services in the network by separating the data transmission and control of the network. In initial designs, the control plane was achieved with additional controller devices to allow it to evolve rapidly, while maintaining simple and cost-effective switching elements.[1]

Recent work has envisioned more refined and distributed control mechanisms and security aspects of SDN [4]. The de facto standard[2] for the time being for SDN is provided by OpenFlow, a commercially usable platform [1]. To date, OpenFlow supports a series of functionalities. First, it allows the control plane access to:

- The type of connection used (e.g., fiber optic, copper wire) along with negotiated connection speed
- Hardware description and available capabilities, and various statistics to each individual switch, specifically:
  - –Port input/output in both packets and bytes
  - –Data flow input/output in both packets and bytes
  - –Packets dropped in both input and output queues of each port
  - –Number of collisions and collision errors detected

Second, the control plane may instruct the data path to:

- Install data flow entries on switches and routers
- Modify existing flow entries on switches and routers
- Request and set features and configurations of switches and routers

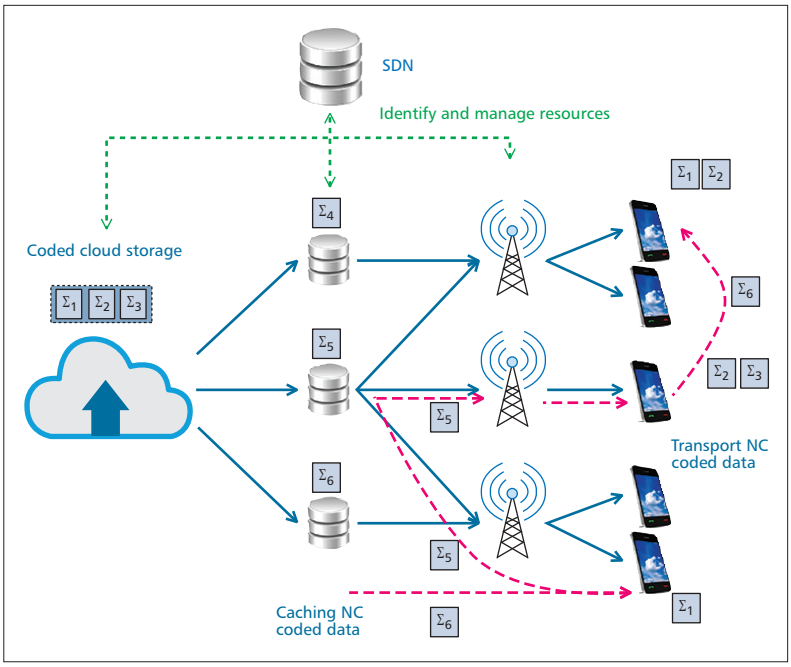

Figure 1. 5G network with a network coded software defined network. Seamless identification of data source, cache management, device-to-device, or machine-to-machine communications, multi-path support, and multi-source exploitation. The SDN controller has an overall perspective of the network including data sources, edge caches, and base stations, and is able to identify and manage network resources, including computational ones for network coding. The SDN controller can identify coded fragments of files in caches as well as in peer devices for orchestrating the transmission of data to new devices interested in a specific content. Bold lines represent connections, while red dashed lines represent data flows, and green dashed lines represent control plane flows.

- Request update on input/output statistics for individual switches and routers

Finally, the SDN data plane can inform the control plane about:

- Unidentified packet headers (and thus request a new flow entry)
- Removed data flows
- Modified port status
- Errors in the data path

These functionalities allow a centralized control plane, shown in Fig. 1, to identify coding-capable devices, link characteristics (round-trip delay, packet loss rates), and topology to determine paths to be used and where/how much coding needs to be incorporated. The SDN controller could also identify coded fragments of popular files in caches as well as in peer devices for orchestrating the transmission of data to new devices.

On the other hand, NC breaks with the store-and-forward paradigm used in today's networks by encouraging intermediate nodes in the network to recode incoming data packets using algebraic operations over finite fields (e.g., a middle device in Fig. 1 recoding and sending a new coded packet to the upper devices). This contrasts with standard end-to-end erasure correcting codes (e.g., LT and Raptor codes), and allows the network to generate redundancy where it is needed instead of injecting it from the source. In a way, NC proposes a store-code-forward paradigm to networking. Random linear network coding (RLNC) provides a distributed,

[1] https://irtf.org/sdnrg

[2] There are other mechanisms to configure switches or new SDN software projects. For example, OpenDayLight (http://www.opendaylight.org/) is a recent and exciting project with a first software release in February 2014. However, OpenFlow was released in 2011 and is already supported in commercial switches.

The comprehensive view of network conditions that is available through SDN can be pivotal to deploy and manage NC configurations and recoding potential within the network as well as identifying storage locations to bridge users and/or devices to their data.

asymptotically optimal approach to employ NC. RLNC is based on choosing random coefficients to create linear combinations of incoming packets. The reason behind these gains comes from the fact that:

- The network itself does not need to transport packets without modification, but rather a linear combination of the original data packets, thus providing a richer set of options and actions available to the network.
- The receivers do not need to track individual packets, but instead focus on accumulating enough independent linear combinations in order to recover the original packets.

Although the gains have been shown in a variety of scenarios, and implementations have confirmed NC's potential in practice, NC's incorporation in standards and wide deployment has been limited with some exceptions; for example, CATWOMAN [5] is currently deployed as part of the Linux Kernel.[3] Part of the limitation lies in the difficulty of retrofitting routers and switches in the network with NC functionalities. However, enabling even a limited number of such devices with NC (e.g., new 5G equipment) can have a large impact on performance if we are able to identify and exploit them. SDN can ease this process and other functionalities. Finally, the presence of a high-performance network coding library (kodo [6]) would ease the deployment of a multiplicity of NC strategies. Although the base functionalities are simple — encode, recode, and decode data packets — NC supports a variety of code designs and configurations, from the classical block-by-block RLNC to online NC, which essentially allows the encoder and recoders to use a moving window for deciding which packets to include in the next linear combination. Kodo [6] also supports systematic NC codes, sparse RLNC, perpetual NC, and fulcrum NC [7], providing a wide range of configurations for deployment in networks.

Current State of Affairs

Both technologies are currently discussed within the Internet Research Task Force (IRTF) on NC[4] and SDN.[5] However, the communities actively involved in SDN and NC have had little if any overlap in the past, which limits mutual understanding of the challenges and capabilities of each, and limits the opportunities to combine the two in a meaningful way.

Where Can SDN and NC Help?

The combination of SDN and NC brings forth an interesting potential for the management and operation of 5G networks. In particular, the comprehensive view of network conditions that is available through SDN can be pivotal to deploy and manage NC configurations and recoding potential within the network as well as identifying storage locations to bridge users and/or devices to their data. The following benefits are possible by this combination:

Exploitation of multiple communication paths: NC is particularly well suited to exploiting multiple communication interfaces and routes [8], which can then exploit SDN's ability to recognize multiple communication paths between source and destination. This is key in 5G networks to comply with reliability requirements as well as appropriate management of heterogeneous interfaces, such as millimeter-wave (mmWave) for increased speed, and another technology for continuous connectivity or for M2M purposes as in Fig. 1.

Management of data storage and caching: SDN's ability to virtualize and/or identify caching and storage nodes in the network are key to exploiting NC to enhance the impact and reduce the storage cost of caching/storage by relying on linearly coded packets instead of replication of the original data per storage location/device. NC also provides a single code for both storage and data transmission, which is key to treating data as a single holistic process, as seen in Fig. 1. This management can also include the preemptive caching of data of a user as it moves through the network using location information. The goal is to guarantee low latency for access to the user's data.

Adaptation of redundancy based on link quality: SDN provides simple mechanisms to identify the link quality, including packet losses, for the transmission routes used for a flow. This capability is particularly relevant for using NC's recoding capabilities to generate the right level of redundancy per link, instead of introducing end-to-end redundancy to compensate for packet losses, which is an inherently inefficient strategy.

Assessment of system load and complexity allocation: SDN is useful for identifying whether a device can commit resources to recoding and how many, since the control plane is able to access information about the hardware of each switch. This may allow us to choose the NC parameters to meet the current network demands. This is particularly relevant with novel NC schemes that provide fluid allocation of complexity, such as fulcrum network codes [7], by performing linear combinations using different finite fields end to end and at different nodes in the network. The choice of the finite field has a direct impact on the computational effort required by a given node. This added flexibility is key to dealing with energy efficiency in 5G networks, not only for the infrastructure but for connected machines (e.g., sensors, actuators) and end-user devices. Additionally, an SDN controller could provide a simple trade-off between computation (reduced coding load) and communication (increased communication load) in the network by deciding whether and where to code various flows depending on load statistics obtained by SDN, while utilizing the flexible SDN flow management to accommodate such changing conditions. In a sense, each switch can be configured to code (or not) the incoming flows, as illustrated in Fig. 2.

Implementation and Testbed

In order to advocate for the integration of NC into SDN, we include a set of simple network topologies that show the benefits of this merging of technologies. The simplicity of the topologies further emphasizes the usability of the merged technologies as large-scale and more complex network scenarios can be broken into a collection of these simple topologies. That is, the benefits found in the simple topologies are readily applica-

[3] http://lwn.net/Articles/549477/

[4] https://irtf.org/nwcrg

[5] https://irtf.org/sdnrg

ble in complex scenarios. The network topologies utilized for this each represent a scenario where NC provides a potential benefit, but where an implementation conflicts with the boundaries set up by the structure of the network protocol stack. Simple network scenarios include single-hop, multihop, and multi-path. Each of the scenarios are further elucidated below.

Coding Devices

A functional software defined network is necessary in order to confirm the capabilities of the developed coding in such a network. However, network equipment capable of SDN (e.g., OpenFlow or similar protocols) only provide limited possibilities in terms of modification and configuration of capabilities. We propose a flexible approach to evaluate the potential of the combination of NC with SDN without initiating full integration of NC into SDN software. This is a crucial step not only to experiment with different strategies and schemes but to gain insight as to the key and most promising elements that could be included into OpenFlow or similar projects as well as motivating switch manufacturers to support new coding features.

For this reason, a virtual network environment has been set up using Open vSwitch,[6] an open source virtual switch supporting the OpenFlow protocol [9]. A coding device in the virtual network setup is based on the developed coding software. Instead of integrating the coding software directly into the Open vSwitch devices, the coding is deployed on virtual machines. This is a development decision based on both a limited timeframe for development, and the fact that a virtual machine can be substituted with a real device without changes to the code in case a real network scenario should be deployed. The possibilities of integrating the coding software on existing network equipment, such as a switch, is very limited. A (virtual) machine with the coding software deployed acts as a coding device on the virtual network, and can be used for either encoding, decoding, or recoding. The coding software distinguishes between packets that need to be coded, recoded, or decoded, and packets that should be ignored in terms of coding.

The virtual machine utilizes virtual network interfaces that ensures communication toward the host operating system (OS). That is, within the virtual machine, this device acts as a normal network interface connected to a network, while the host OS has the responsibility of handling traffic to and from this interface. The virtual interface can be included in virtual network scenarios specified by the host OS. Two of these network interfaces are included in each virtual machine. One is to be included in the virtual software defined network scenarios, and one is utilized for direct communication with the host OS. The latter eases the setup and configuration through host-side scripts without "polluting" the investigated network scenario. This relationship between host OS, virtual machine, and coding software is illustrated in Fig. 2.

Deploying the coding in virtual machines naturally curbs the obtainable performance compared to dedicated integration directly in the virtual switch, say, due to the introduction of additional delay for directing traffic to the virtual machine, processing, and sending back to the switch. However, we argue that a virtual implementation is an equal advocate for NC to be integrated into a software defined network. The virtual setup shows that only a limited amount of coding-related instructions is needed to create a beneficial coding approach that can be deployed on future SDN-capable equipment.

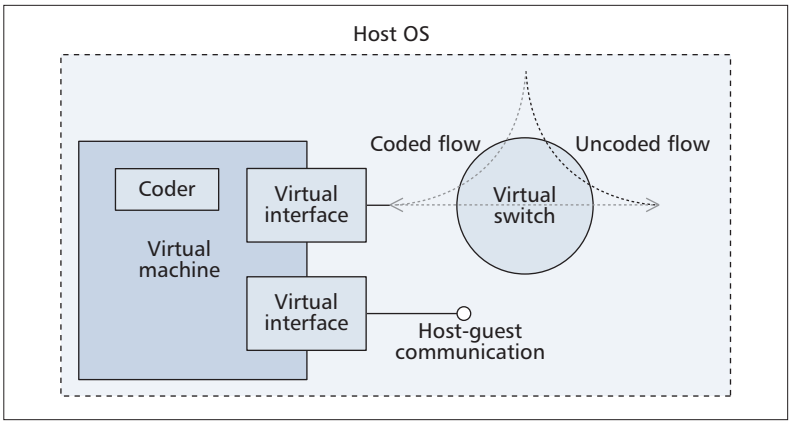

Figure 2. Integrating network coding into an Open vSwitch, showing the relation between coding software, virtual machine, and host OS. A data flow (the grey dashed line) is redirected through the coding device before it is forwarded onto the network. The flows that do not require coding follow the black dashed line straight through the switch.

Intermediate network nodes in the constructed virtual network setup consist of an Open vSwitch along with an adjacent coding device. Data flows to be coded are then redirected from the switch through the coding device and back to the switch, which then forwards the coded packets appropriately. The coded data flows are specified in the coding devices using statistics data obtainable from the switch using the capabilities of the OpenFlow protocol, where key supported parameters for implementing efficient network coding are the packet loss statistics and topology information. Although packet loss statistics could be obtained by other means, such as using the Simple Network Management Protocol (SNMP), the SDN framework will allow for further system configuration and control in future developments, including route selection and complexity management for NC functionalities. From the perspective of the SDN controller, the controller would select the use of the coding switch for a specific flow by routing the data to the virtual machine and from the virtual machine back to the router for transmission to the next hop.

In general, this combination of switch and coding device imitates the behavior of a network-coding capable network node. This could be deployed either as an overlay, for example, computation on top of existing (but limited) SDN-capable switches, or within the switch itself. Figure 2 provides a simple illustration of how the two components, switch and coding device, cooperate to create coded flows within a network.

Coding Software

The software implementation[7] uses Kodo [6], which is a C++11 network coding library capable of random linear network coding (RLNC).

[6] http://www.openvswitch.org

[7] https://github.com/14gr1010/software

The coding scheme is used as forward error correction (FEC), that is, neither positive nor negative acknowledgement (ACK/NACK) is used to ensure delivery of every packet. The encoder uses a systematic code [10] where all the original packets are transmitted uncoded (but with a header added by Kodo and zero-padded to the RLNC symbol size) the first time. Throughout the performance measurements we use a generation size of 10 symbols, Galois field in use is GF (2^8), and the symbol size is set to 1356 bytes. Some of the benefits of a systematic code is lower decoding complexity and delay [10]. Lower decoding delay is achievable since all uncoded packets can be forwarded directly, and having uncoded packets reduces the decoding complexity of Gaussian elimination, which is used to decode the RLNC generation. Kodo supports RLNC on the fly where packets can be added to the encoder as they arrive. Similarly, packets can be extracted from the decoder as soon as they are decoded, without the need for decoding the entire generation first. One of the benefits of RLNC on the fly is that coded packets can be transmitted before the entire generation is fed to the encoder (i.e., adding redundancy on the fly). The combination of systematic and on-the-fly coding allows the encoder to transmit uncoded and coded packets with minimal delay. However, using on-the-fly coding also reduces the decoding probability compared to a traditional block code. Since redundancy packets can only aid decoding the symbols that were added to the encoder at the time of their creation, they may become useless later. For example, a redundancy packet created at the beginning of a generation cannot be used to recover a lost symbol in the last part of a generation. But without on-the-fly coding, slow or infrequent communication is at best problematic; for example, in a ping scenario it would be inconvenient to wait for an entire generation to be filled before sending any packets.

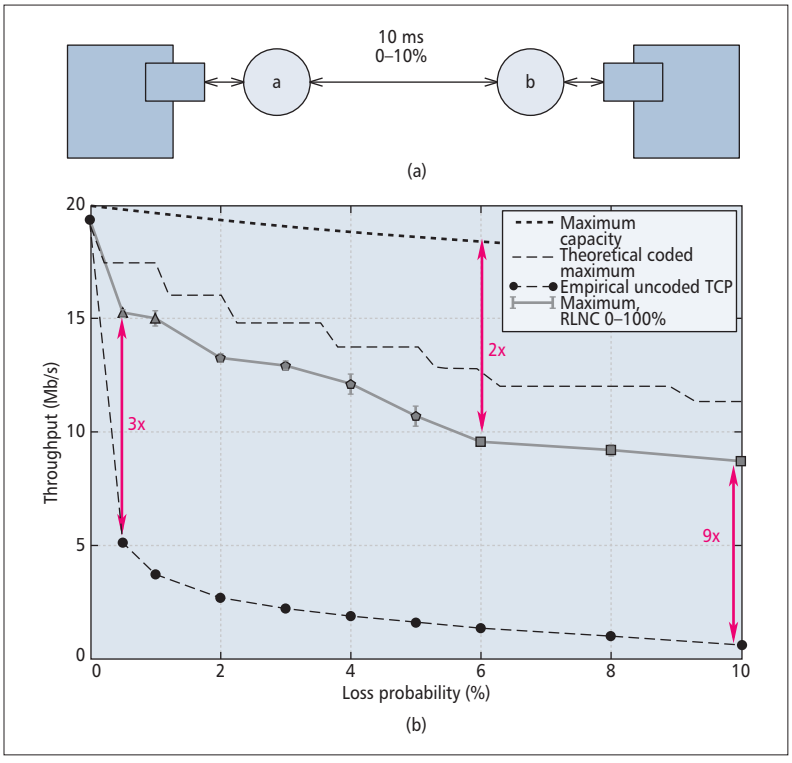

Figure 3. Single-hop network comparing performance of TCP using network coding for erasure protection, TCP without coding, as well as the theoretical maximum of our simple coded approach and the theoretical capacity of the channel (without protocol effects). SDN is used to detect the link quality and adapt the redundancy introduced by the system.

Measurement Results

The developed software and the virtual network environment in which the software is deployed indicate that integration of NC as a functionality of a software defined network is indeed possible. However, this alone does not show that this combination of technologies is actually beneficial. In order to show that the proposed integration of network coding is plausible, a series of performance measurements has been carried out using the standard tool for network performance measurement, *iperf*. As a secondary result, this should also show that the coding approach can be applied without breaking functionality with the conventional TCP/IP network protocol stack.

We focus on links with a mean loss rate within 0 and 10 percent. Note that typical WiFi links' mean loss rates have been reported between 0.1 and 0.5 percent and as high as 4 percent depending on hardware and drivers [11] and higher in long-range links [12]. The mean packet loss rate for 3G, 3.5G, and 4G networks has been found to be in the range of 0.18–0.66 percent, 0.05–0.14 percent, and 0.03–0.30 percent, respectively [13]. The following results show that significant performance gains can be found even with loss rates below 0.5 percent using our techniques.

Single-Hop

The single-hop scenario consists of two virtual nodes, each connected to individual virtual switches. These switches are then connected with a virtual Ethernet connection on which delay and loss are introduced. This simple topology along with specified delay and packet loss is depicted in Fig. 3a. This scenario is representative of networks where only the destination and the source are capable of coding. Alternatively, the network itself could provide such functionality transparently for end devices in order to provide protection against lossy parts of the network (e.g., a satellite link).

By isolating the coding approach to a single link between coders, the consequence of the data recovery process within the developed coding approach is revealed. Despite the potential ability to recover every single erasure that may occur on the link, the performance of the transported TCP communication may not resemble error-free TCP communication. This is due to the inevitable delay of the error recovery phase, from when a lost packet should have been received to the point where it is successfully decoded. The amount of additional interference in terms of delay and jitter is reduced to a bare minimum in this single-hop scenario. The channel conditions on the investigated link are then adjusted to illustrate the tolerance of both the TCP communication and the deployed coding. Increasing packet loss on the link reveals the

robustness added by the coding. The bandwidth of an uncoded TCP connection is compared to TCP connections carried by the deployed coding approach using systematic on-the-fly RLNC. The achievable throughput for both uncoded and coded data flows is stated in Fig. 3b. This is compared to the channel capacity and the theoretical maximum throughput for a coded flow, found using a model for TCP throughput [14] modified to accommodate the utilized coding approach.

From the performance of the coded data flow a gain of 3*x* is obtained already at 0.5 percent packet loss. This performance boost increases up to 9*x* at 10 percent packet loss probability. Furthermore, the obtained performance of the coded flow follows a trend similar to the theoretical maximum coded throughput, showing coherence between theory and practice.

Finally, note that the latency (i.e., the number of packets in flight given the transmission rate and round trip delay) and the packet loss rate are the key factors to determine the optimal choice of redundancy to be added for coding. Thus, it is possible to develop a network coding control algorithm that can optimize for the current situation in the network.

It is important to note that the use of a more advanced and integrated coding technique (e.g., Coded TCP [14]) would require the optimal amount of redundancy for the specific network. Thus, coding would not introduce unnecessary redundancy and provide much better end-to-end service. In fact, the use of SDN would address the main limitation of Coded TCP, that is, the estimation of packet losses through the network not attributed to congestion, thus having the potential to improve its performance with respect to current practical demonstrations.

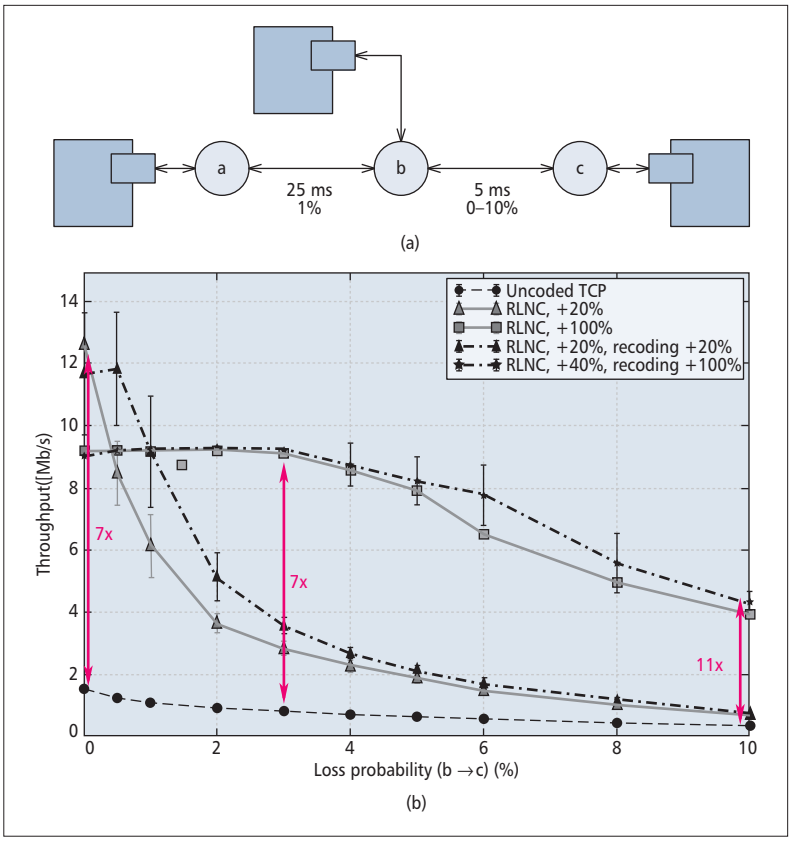

Figure 4. Multihop network comparing a) uncoded TCP, RLNC without recoding; RLNC with recoding at the intermediate node. Recoding even after a moderate loss channel (1 percent loss rate) can provide some benefits in end throughput.

Multihop

In order to illustrate the benefits of recoding, we utilize a multihop setup where both links experience different loss probability and link delay. This network setup is presented in Fig. 4a. This particular setup is representative of Internet service provider (ISP) networks, where typically the last hop is a wireless and lossy link. Our assumption is that a switch at the last mile provides the necessary recoding for protecting against losses. Thus, studying a single recoder has a large implication in more complex topologies, where we abstract other hops within the network as they are error-free for all practical purposes.

This network scenario is also representative of mesh-like network structures, such as dedicated sensor networks, mobile ad hoc networks, and vehicular communication networks. While uncommon in consumer oriented networks, such networks are expected to gain popularity in the future with the growth of the Internet of Things (IoT). The setup illustrates the necessity of intermediate coding (recoding) when all links are prone to erasures. While prior research efforts have already drawn this conclusion, the setup tests the validity of this with a simple feedback-less coding approach using SDN to gain knowledge of the channel conditions and structure of the network. Additionally, recoding may introduce channel irregularities such as packet reordering and additional delay and jitter. By running TCP on top of a recoded data flow, these recoding issues are tested in practice.

Figure 4b reveals the strength of recoding. Apart from up to 11*x* gains over uncoded TCP, recoding also reduces congestion on the first link $a \rightarrow b$ and introduces higher achievable throughput compared to the end-to-end RLNC coding approaches.

Similar to the single-hop case, network characteristics such as latency packet loss rate at each link determine the optimal network coding mechanism at the encoding and each of the recoding nodes. Thus, developing an NC control algorithm ran at the control plane will allow optimal operation of the system.

Multi-Path

The final investigated network scenario is a multi-path setup, where multiple data paths span out between nodes. The conventional methods for communication in such scenarios is choosing the best of the available paths. This is naturally the simplest approach, and while the chosen data path provides adequate capacity, it is probably also the best approach. However, some communication scenarios may benefit from utilizing several of the available data paths. Multi-Path TCP (MPTCP) has been developed for such scenarios, but suffers from similar behavior toward packet loss and link delay as that of conventional TCP. Using a combination of network

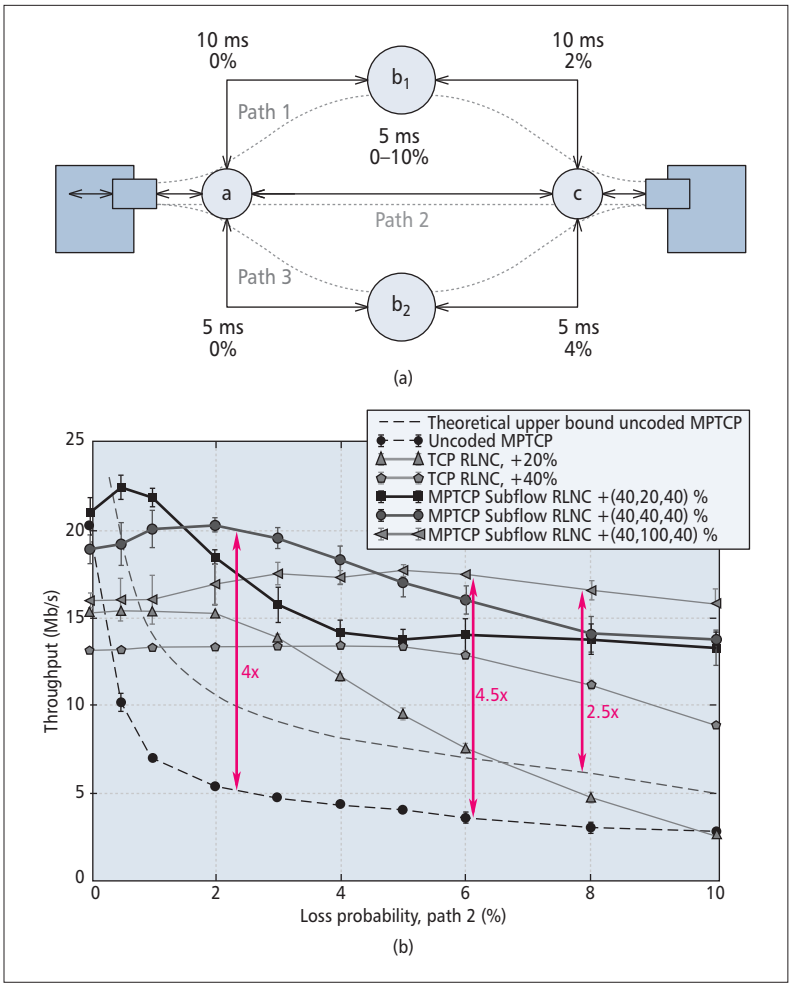

Figure 5. Multi–path network comparing the use of standard MPTCP, individual TCP flows with RLNC loss protection, and MPTCP with protection of RLNC on individual paths. The plot shows that the theoretical maximum for MPTCP without coding can be outperformed more than two-fold by the use of coding, while the gain is over four-fold when comparing to a real MPTCP implementation: a) network structure; b) multi-path network performance.

coding and SDN to accommodate packet loss on each individual path may provide similar benefits for MPTCP as for conventional TCP in the single-hop and recoding scenarios.

SDN is used here not only for link quality statistics and discovery of coding nodes, but for topology discovery. In ISP networks, this can be representative for devices using separate technologies (e.g., 4G, WiFi) for accessing different networks but with a common destination (e.g., a server). In a sense, we consider that each of those networks can be represented by a simple model of delay and loss. Once the paths and their characteristics have been identified, the coding is set up to provide protection against losses in individual paths, allowing MPTCP to handle congestion control. This is not the most integrated solution, but it does allow the network to code and improve performance without changing the behavior of MPTCP.

Figure 5a illustrates the multi-path scenario investigated. This consists of one direct single-hop path (path 2) and two indirect paths with an additional hop. The two multi-hop paths, $\mathbf{a} \rightarrow \mathbf{b}_1 \rightarrow \mathbf{c}$ and $\mathbf{a} \rightarrow \mathbf{b}_2 \rightarrow \mathbf{c}$, are denoted path 1 and path 3, respectively.

In a multi-path scenario, the achievable throughput of uncoded MPTCP is compared to that of a coded approach, where RLNC is utilized to protect each MPTCP subflow individually. The resulting performance is illustrated in Fig. 5b. This also includes a theoretical upper bound for the performance of MPTCP [15] along with a coded conventional TCP flow, carried over path 2. The various configurations of Coded MPTCP (top three curves) correspond to different levels of redundancy introduced in the direct link between **a** and **c**, that is, 20, 40, and 100 percent, while the redundancy of the two other paths is kept fixed at 40 percent.

Due to the high sensitivity toward packet loss and link delay, even the theoretical upper bound for MPTCP indicates poor performance in the multi-path network, and even the single-path coding approach outperforms MPTCP even though only a third of the total capacity is avaiable to this approach. The multi-path coding provides a performance increase of up to 2.5x over the theoretical MPTCP upper bound and 4.5x over the obtained MPTCP performance.

Conclusions

This article advocates the integration of network coding as part of software defined networking as a key to operate 5G networks more efficiently, with higher resiliency, providing higher throughput, and allowing control of data location to enable low-latency services. Furthermore, we show that the essential software packages from each concept, OpenFlow and Kodo, are already available and can be integrated to provide the required functionalities to current and future networks.

Using three basic topologies, we demonstrate not only this integration of concepts but also that simple coding strategies enable us to outperform standard TCP and multi-path TCP without modifying the underlying end-to-end transport protocols as a first step to understanding their potential. To achieve this, we exploit recoding at intermediate nodes in the network and show that gains of 3x to 11x are attainable.

Outlook

In order to validate the measurement results using real SDN-capable switches and high-end desktops, we recently built a testbed with 16 programmable high-performance network nodes and one real SDN-enabled 48-port switch. These nodes can be configured in a variety of scenarios and topologies for measurement and demonstration purposes. Each node consists of one NetFPGA with a 10 Gb/s PCI-Express programmable network interface (netfpga.org), which is highly configurable and has an active research community. The FPGA solution is needed to allow for fast switching and routing decisions, and high-end processors are needed to get the network coding speeds to satisfy the 10 Gb/s links. This equipment shall be at the core of the design and testing of 5G algorithms and network protocols.

Acknowledgment

This work was partially financed by the Green Mobile Cloud project (Grant No. DFF — 0602-01372B) granted by the Danish Council for Independent Research and by the VELUX Visiting Professor Programme 2013-2014 granted by the VELUX Foundation.

References

[1] N. McKeown *et al.*, "Openflow: Enabling Innovation in Campus Networks," *SIGCOMM Comp. Commun. Rev.*, vol. 38, no. 2, Mar. 2008, pp. 69–74, http://doi.acm.org/10.1145/1355734.1355746

[2] M. Kim *et al.*, "Network Coded TCP (CTCP)," *Computing Research Repository,* vol. abs/1212.2291, 2012.

[3] O. Bonaventure, M. Handley, and C. Raiciu, "An Overview of Multipath TCP," USENIX login; vol. 37, no. 5, Oct. 2012.

[4] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards Secure and Dependable Software-Defined Networks," *Proc. 2nd ACM SIGCOMM Wksp. Hot Topics in Software Defined Networking*, 2013, pp. 55–60, http://doi.acm.org/10.1145/2491185.2491199

[5] M. Hundebøll *et al.*, "Catwoman: Implementation and Performance Evaluation of IEEE 802.11 based Multi-Hop Networks Using Network Coding," *2012 IEEE VTC-Fall*, Sept 2012, pp. 1–5.

[6] M. V. Pedersen, J. Heide, and F. Fitzek, "Kodo: An Open and Research Oriented Network Coding Library," *LNCS*, vol. 6827, 2011, pp. 145–52.

[7] D. E. Lucani *et al.*, "Fulcrum Network Codes: A Code for Fluid Allocation of Complexity," *CoRR*, vol. abs/1404.6620, 2014.

[8] A. Moreira and D. Lucani, "On Coding for Asymmetric Wireless Interfaces," *2012 Int'l. Symp. Network Coding*, June 2012, pp. 149–54.

[9] S. Vaughan-Nichols, "OpenFlow: The Next Generation of the Network?," *Computer*, vol. 44, no. 8, 2011, pp. 13–15.

[10] J. Heide *et al.*, "Network Coding for Mobile Devices — Systematic Binary Random Rateless Codes," *IEEE ICC Wksps.*, 2009, June 2009, pp. 1–6.

[11] D. C. Salyers, A. D. Striegel, and C. Poellabauer, "Wireless Reliability: Rethinking 802.11 Packet Loss," *2008 Int'l. Symp. A World of Wireless, Mobile and Multimedia Networks*, June 2008, pp. 1–4.

[12] A. Sheth *et al.*, "Packet Loss Characterization in WiFi-based Long Distance Networks," *26th IEEE INFOCOM*, May 2007, pp. 312–20.

[13] Y.-C. Chen *et al.*, "Characterizing 4G and 3G Networks: Supporting Mobility with Multi-Path TCP," Dept. Comp. Sci., UMass Amherst, tech. rep. UM-CS-2012-022, 2012.

[14] M. Kim, M. Médard, and J. a. Barros, "Modeling Network Coded TCP Throughput: A Simple Model and Its Validation," *Proc. 5th Int'l. ICST Conf. Performance Evaluation Methodologies and Tools*, Brussels, Belgium, 2011, pp. 131–40, http://dl.acm.org/citation.cfm?id=2151688.2151704

[15] J. Cloud *et al.*, "Multi-Path TCP with Network Coding for Mobile Devices in heterogeneous networks," *2013 IEEE VTC-Fall*, Sept. 2013, pp. 1–5.

Biographies

Jeppe Krigslund (jepkri@es.aau.dk) is a software developer at Steinwurf ApS working on network coding protocols for wireless video multicast. He was a student in the Elite Masters Programme in Wireless Communication at Aalborg University (AAU), Denmark, where he also received his B.S. degree in electrical engineering in 2012. His research interests revolve around wireless communications and multimedia transmission with work including a mix of wireless mesh networks, network coding, video streaming, and cooperative protocol design.

Jonas Hansen (jh@es.aau.dk) is an industrial Ph.D. student at Bang & Olufsen and AAU working on network coding code design and wireless protocols for audio signals. He was a student in the Elite Masters Programme in Wireless Communication at AAU, where he also received his B.S. degree in electrical engineering in 2012. His research interests are wireless communications and multimedia transmission with an emphasis on low-latency traffic and applications.

Daniel E. Lucani (del@es.aau.dk) is an associate professor in the Department of Electronic Systems, AAU. He was an assistant professor at the University of Porto from 2010 to 2012 before joining AAU. He received his B.S. and M.S. degrees in Electronics Engineering from Universidad Simón Bolívar, Venezuela, in 2005 and 2006, respectively, and his Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology in 2010. His research focuses on communications, network theory, and network coding theory and applications.

Frank H. P. Fitzek (frank.fitzek@tu-dresden.de) is the coordinator of the 5G Lab Germany and a professor at Technische Universitat Dresden He received his diploma (Dipl.-Ing.) degree in electrical engineering from RWTH-Aachen, Germany, in 1997, and his Ph.D. (Dr.-Ing.) in electrical engineering from the Technical University Berlin, Germany in 2002. He has received numerous awards, including the NOKIA Champion Award five times, the NOKIA Achievement Award (2008), the Danish SAPERE AUDE research grant (2010), and the Vodafone Innovation prize (2012). His research focuses on wireless and mobile networks, mobile phone programming, network coding, cross-layer and energy-efficient protocol design, and cooperative networking.

Muriel Medard [F] (medard@mit.edu) is the Cecil H. Green Professor in the Electrical Engineering and Computer Science Department at MIT, and leads the Network Coding and Reliably Communications Group at the Research Laboratory for Electronics at MIT. She is the Editor-in-Chief of the IEEE Journal on Selected Areas in Communications. She was President of the IEEE Information Theory Society in 2012. She received the 2009 IEEE Communication Society and Information Theory Society Joint Paper Award, the 2009 William R. Bennett Prize in the Field of Communications Networking, and the 2002 IEEE Leon K. Kirchmayer Prize Paper Award. She is among the most highly cited researchers in her field. As a result, she was named one of the World's Most Influential Scientific Minds in 2014 by Thomson Reuters.

The FPGA solution is needed to allow for fast switching and routing decisions, and high-end processors are needed to get the network coding speeds to satisfy the 10 Gb/s links. This equipment shall be at the core of the design and testing of 5G algorithms and network protocols.