Abstract—The applications in the critical infrastructure systems pose simultaneous resilience and performance requirements to the underlying computer network. To meet such requirements, the networks that use the store-and-forward paradigm poses stringent conditions on the redundancy in the network topology and results in problems that becoming computationally challenging to solve at scale. However, with the advent of programmable data-planes, it is now possible to use linear network coding (NC) at the intermediate network nodes to meet resilience requirements of the applications. To that end, we propose an architecture that realizes linear NC in programmable networks by decomposing the linear NC functions into the atomic coding primitives. We designed and implemented the primitives using the features offered by the P4 ecosystem. Using an empirical evaluation, we show that the theoretical gains promised by linear network coding can be realized with a per-packet processing cost. 

Index Terms—Resilience, Network Coding, Software Defined Networking

I. INTRODUCTION

The applications that constitute the critical infrastructure (e.g. smart power generation and distribution systems, oil refineries etc.) have a unique set of requirements regarding their underlying communication networks. For example, such applications require that their communication is seamlessly resilient against link or device failures. Furthermore, these applications also require a predictable end-to-end delay for data delivery in multicast settings [1] [2]. Such resilience and performance requirements cannot be simultaneously accomplished by mere over-provisioning of network resources such as topological redundancy or bandwidth.

Rather, in the packet store-and-forward paradigm, the resiliency is provided by routing the packets around a failure [3] [4]. However, such approaches requires solving combinatorial problems. Similarly, end-to-end delay requirements are met by solving resource allocation problems on a per-flow basis [5]. However, even a static resource allocation for flows that have such requirements is a known NP-complete problem [6]. Therefore, combining the performance and resilience requirements poses an intractable problem.

Such intractability is due to the atomic nature of a packet flow in the store-and-forward paradigm. This results in hard decisions. However, NC converts such hard decisions into one of many soft decisions by mixing packets at intermediate network devices using algebraic coding. In theory, NC promises to provide seamless resilience to failures for critical infrastructure applications over the store-and-forward paradigm [7] [8]. However, practical NC that achieves the promised theoretical gains has remained elusive until recently.

With the use of programmable data-planes [9], it is now possible to deploy network functions using a flexible data-plane architecture in production networks. Hence, we propose an architecture capable of meeting the requirements of the data streams generated by applications in critical infrastructure systems. Our contributions are:

- A library of atomic NC primitives and its use to construct linear network coding functions.
- Evaluation of the NC functions to show that the seamless resilience and multicast rate gains are obtained at a small per-packet processing cost of coding and decoding the packets in the data-plane.

II. DESIGN

A coding function realizes a linear code to improve resilience or throughput of a multicast data stream. Figure 1 shows two examples: A diversity code [10] that provides seamless resilience for a unicast stream and a code for enhancing receiver’s data rate of a multicast stream [7]. Each packet carries a coding header to coordinate the operations performed by the coding primitive and the packet’s payload.

A coding primitive is an atomic block of data-plane functionality. Each incident stream of packets on the switch is subject to one or more primitives. The coding functions orchestrate the exact sequence of coding primitives. We developed following coding primitives:

- **Splitting**: primitive splits a given packet stream into batches of packets for each stream.
- **Coding**: primitive generates packets by using the cloning and recirculation in tandem. The coded payload is computed over a batch of payloads held in registers.
- **Forwarding**: primitive performs unicast or multicast forwarding of a packet.
- **Gathering**: primitive collects a batch of ingress packets into the registers corresponding to their stream_id.
- **Decoding**: primitive also uses cloning and recirculation to generate packets. It takes the gathered packets, decodes them and forwards the resulting packets to the host.

III. EVALUATION

We performed experiments using a prototype of the library of primitives and mininet with BMv2 as P4 target. We measured the multicast rate gains obtained by using the code shown in Figure 1(a). The host on the left side multicasts a data stream to the two hosts on the right. We sent 1000 packets containing 4096 bytes of payload at various data send
rates with an exponentially distributed inter-packet time to match the rate. The link bandwidth was set to 0.01 Mbps. As shown in Figure 2, we found that the received rate drops for forwarding as the send rate is at 40% of the max-flow, whereas it does not drop for NC until 80% of the max-flow.

We measured the cost of cloning and recirculation when using the code shown in Figure 1(b). We varied the delay for link between \(S_1-S_4\) to create a delay differential (\(\delta\)) for packets arriving at \(S_5\). Figure 3 shows the per-packet processing time (\(\rho\)) over 1000 packets. We found that \(\rho\) for switches that forward the packets (at \(S_2, S_3, S_4\)) increases slightly with payload sizes. We found that \(\rho\) for coding (at \(S_1\)) and decoding (at \(S_5\)) is up to four and six times more than \(\rho\) to forward a packet. We found that for a lower \(\delta\), the \(\rho\) for decoding is higher than \(\rho\) for coding and vice versa. This is because, for a lower \(\delta\), the switch needs to decode the payloads, whereas for a high \(\delta\), the switch only forwards the packets.

**REFERENCES**

[1] D. E. Bakken, A. Bose, C. H. Hauser, D. E. Whitehead, and G. C. Zweigle, “Smart generation and transmission with coherent, real-time data,” Proceedings of the IEEE, vol. 99, no. 6, pp. 928–951, 2011.

[2] K. Leggett, R. Moxley, and D. Dolezilek, “Station device and network communications performance during system stress conditions,” in proceedings of the Protection, Automation and Control World Conference, Dublin, Ireland, 2010.

[3] T. Elhourani, A. Gopalan, and S. Ramasubramanian, “Ip fast rerouting for multi-link failures,” IEEE/ACM Transactions on Networking, vol. 24, no. 5, pp. 3014–3025, 2016.

[4] M. Chiesa, I. Nikolaevskiy, S. Mitroví, A. Panda, A. Gurtov, A. Maldý, M. Schapira, and S. Shenker, “The quest for resilient (static) forwarding tables,” in Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on. Ieee, 2016, pp. 1–9.

[5] R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, and R. B. Bobba, “End-to-end network delay guarantees for real-time systems using sdn,” in IEEE Conference Real-Time Systems Symposium (RTSS)(Accepted), 2017.

[6] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” IEEE Journal on selected areas in communications, vol. 14, no. 7, pp. 1228–1234, 1996.

[7] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, “Network information flow,” IEEE Transactions on information theory, vol. 46, no. 4, pp. 1204–1216, 2000.

[8] R. Koetter and M. Méard, “An algebraic approach to network coding,” IEEE/ACM Transactions on Networking (TON), vol. 11, no. 5, pp. 782–795, 2003.

[9] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese et al., “P4: Programming protocol-independent packet processors,” ACM SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 87–95, 2014.

[10] E. Ayanoglu, I. Chih-Lin, R. D. Gitlin, and J. E. Mazo, “Diversity coding for transparent self-healing and fault-tolerant communication networks,” IEEE Transactions on communications, vol. 41, no. 11, pp. 1677–1686, 1993.