NUTS: Network Updates in Real Time Systems

Saif Un Noor Prottoy, Damien Saucez, Walid Dabbous

▶ **To cite this version:**

Saif Un Noor Prottoy, Damien Saucez, Walid Dabbous. NUTS: Network Updates in Real Time Systems. ACM SOSR 2019 - Symposium on SDN Research, Apr 2019, San Jose, United States. hal-02427489

HAL Id: hal-02427489
https://hal.science/hal-02427489

Submitted on 7 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NUTS: Network Updates in Real Time Systems

Saif UN Noor Prottoy
Université Côte d'Azur, Inria, France

Damien Saucez
Université Côte d'Azur, Inria, France

Walid Dabbous
Université Côte d'Azur, Inria, France

ABSTRACT

Factories need to adapt their communication networks to versatile customer-driven markets. Software defined networking enables a programmatic approach that provides modularity, flexibility and paves the road for behavior certification. Previous works proposed rigorous programming languages and abstractions offering safety properties and verification in best-effort environments. In this work, we propose an approach to provide live update of network elements behavior while respecting real-time constraints. During the network updates, the traffic can be deviated to devices not involved in the desired upgrade ensuring that communication invariant and software requirements are always taken into account. We leverage Temporal NetKAT to write network wide programs and P4 annotations to give indications on the impact of the implementation on deterministic real-time communications passing through network appliances.

KEYWORDS

Network updates, real-time, SDN

1 IN A NUTSHELL

Software Define Networking (SDN) is a key enabler for the so-called Industry 4.0 because it provides flexibility and the possibility to formally reason on networks. However, even though substantial efforts have been provided to offer rigorous programming abstractions [1–3] or safety and verification properties [1, 4, 8] little has been done so far to understand how to support deterministic real-time communications when a network is dynamically programmable. The reason is that SDN is mostly used with best effort services in mind while the industry focuses on certified equipment and guaranteed services.

The problem Bringing the SDN concept to industrial networks is a vast problem. However, we have identified that a critical point to address is *how to support safe network updates of deterministic real-time communication SDN networks.*[1]

The challenge Substantial work has been accomplished to offer safe network updates (see Vissicchio et al. [8]) but the community has focused on updating policies and forwarding states and let aside the problem of software updates and device compatibility even though incremental deployment has been considered [6]. Two points that have not been studied are (*i*) the impact of *software updates* in the network implied by the network updates and (*ii*) the problem of *always respecting real-time constraints*. Usually, network devices in industrial networks implement the minimum set of features required for their tasks. As a consequence, each network update might result in software updates or even firmware upgrades. Orthogonally, in deterministic real-time communications systems the focus is on delivering data packets with respect to sequences, deadlines, and performance metrics (e.g., jitter). Hence, *network updates in industrial systems is a multi-dimensional problem because of network and software updates intertwining.*

The approach Re-purposing network elements because of network updates often means changing their software. Our approach is to consider the network as a whole instead of considering the part of the network that has to be updated: upon network updates, we can temporarily deviate traffic to other appliances. That is, with our approach during a network update, devices and communications that are not explicitly involved in the desired update can be updated as well. This freedom is necessary to always respect communication invariant.The drawback of this approach is that the number of transitions to move from an initial state to a final one can be large and can take considerable time (e.g., minutes).

2 THE CONSTITUTING ELEMENTS

An advantage of industrial systems over the internet/enterprise ones is that the expected behaviour of the communicating elements is known and is rather well formalized. However, by construction the constituting communication systems are segregated and interactions are impossible. If we move to a shared network substrate, the true segregation hypothesis is invalidated. We then (*i*) **need a declarative**

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SOSR '19, April 3–4, 2019, San Jose, CA, USA

© 2019 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-6710-3/19/04.
https://doi.org/10.1145/3314148.3318051

[1] This work was inspired by [8].

programming language to express global deterministic real-time communication policies. For that purpose, we leverage Temporal NetKAT [2]. Temporal NetKAT is programming language to define network wide programs. Temporal NetKAT has not been designed with real-time considerations in mind but the fact that it provides support for linear temporal logic makes it compatible with our objectives of formally defining deterministic real-time communication schemes, as long as all buffers in the network are bounded [7].

Network wide programs should be written independently of their actual implementation in the network, hence the choice of the declarative paradigm. However, the network technology impacts temporal performances of the element and thus the ability to respect real-time constraints. We thus (ii) **need to determine the temporal behavior for each network elements and software**. Because of their technical and operational constraints industrial network appliances seldom resort to operating systems and we can abstract them as programmable data planes. Under this assumption, we can leverage the annotation of the P4 language[2] provide formal ways to give indications on the impact of the implementation on deterministic real-time communications passing through them (e.g., queues, scheduler, processing latency).[3]

The description of the communications and the network elements behaviour is not sufficient to achieve our goal of making network updates in real time systems. In addition to them we **(iii) need to assess if an update is valid according to the different constraints (e.g., safety, liveness, real-time...) imposed by the network and its usage**. To determine whether an update is valid, we extract the requirements and constraints from the communication scheme with (i) and from the actual behaviour of network elements with (ii). This information is combined and we use network calculus [5] and invariant verification to determine if all constraints are respected before, during, and after an update.

To decide if a network update is valid, we (iv) **must determine and select the potential network updates**. To that aim, we can modify the generic network update algorithms proposed by Vissicchio et al. [8] to support our constraints.

Finally, when update sequences are determined, we (v) **need to deploy the changes in the network**. This part is out of the scope of this work but we don't foresee any particular conceptual difficulties.

3 SPROUTING UP

We are still at the early stage of bringing the SDN concept in deterministic real-time communication systems but we can already say that it will be adopted only if we can provide correct and efficient network updates. This requires to be able to precisely define the network workload and how it is expected to be treated. It also calls for new techniques to update network device software on-the-fly in absence of operating systems. Our proposition with NUTS is to define a provable *Network Updates mechanism for real-Time Systems*. With it, we consider the network as a whole and authorize any network element to be reprogrammed to temporarily take care of traffic while other elements are being updated, inasmuch as real-time constraints are not violated.

In this infinite realm, we will focus on how to provide a practical (i.e., fast, provable, and certifiable) method to be able to safely update networks. On the one hand, we will define a level of programming abstraction that allows one to indicate the temporal impact of their implementations on networks. On the other hand, we will develop a polynomial-time algorithm to determine an appropriate sequence of network updates, given the real-time constraints of the system.

[2]https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html#sec-annotations

[3]It doesn't mean that the element itself must support P4.

ACKNOWLEDGMENTS

This work was partially supported by the ANR DET4ALL Project (ANR-18-CE10-0002).

REFERENCES

[1] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. 2014. NetKAT: Semantic Foundations for Networks. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '14)*. ACM, 113–126. https://doi.org/10.1145/2535838.2535862

[2] Ryan Beckett, Michael Greenberg, and David Walker. 2016. Temporal NetKAT. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '16)*. ACM, 386–401. https://doi.org/10.1145/2908080.2908108

[3] Nate Foster, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. 2011. Frenetic: A Network Programming Language. In *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (ICFP '11)*. ACM, 279–291. https://doi.org/10.1145/2034773.2034812

[4] Arjun Guha, Mark Reitblatt, and Nate Foster. 2013. Machine-verified Network Controllers. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '13)*. ACM, 483–494. https://doi.org/10.1145/2491956.2462178

[5] Jean-Yves Le Boudec and Patrick Thiran. 2001. *Network calculus: a theory of deterministic queuing systems for the internet*. Vol. 2050. Springer Science & Business Media.

[6] Dan Levin, Marco Canini, Stefan Schmid, Fabian Schaffert, and Anja Feldmann. 2014. Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. USENIX Association, 333–345. https://www.usenix.org/conference/atc14/technical-sessions/presentation/levin

[7] A Prasad Sistla, Edmund M Clarke, Nissim Francez, and Albert R Meyer. 1984. Can message buffers be axiomatized in linear temporal logic? *Information and Control* 63, 1-2 (1984), 88–112.

[8] S. Vissicchio, L. Vanbever, L. Cittadini, G. G. Xie, and O. Bonaventure. 2017. Safe Update of Hybrid SDN Networks. *IEEE/ACM Transactions on Networking* 25, 3 (June 2017), 1649–1662. https://doi.org/10.1109/TNET.2016.2642586