Annex to the IFIP Networking 2018 Proceedings

Demonstration and Poster Session

Annex to ISBN 978-3-903176-08-9 © 2018 IFIP

Demo: Chaining of Segment Routing aware and unaware Service Functions

Ahmed Abdelsalam
Gran Sasso Science Institute

***Abstract*—Segment Routing (SR) is a source routing paradigm that can benefit from both MPLS and IPv6 data planes to steer traffic through a set of nodes. It provides a simple and scalable way to support Service Function Chaining (SFC). In this demo, we propose an NFV architecture based on SR and implemented in Linux environment. It allows chaining of both SR-aware and SR-unaware Service Functions (SFs). In order to include SR-unaware SFs into SR SFC, we use our SR proxy implementation: srext, a Linux kernel module that handles the processing of SR information in behalf of the SR-unaware SFs. As SR-aware SFs, we use two of our implementation; SERA and SR-aware snort. SERA is a SEgment Routing Aware Firewall, which extends the Linux iptables firewall, and capable of applying the iptables rules to the inner packet of SR encapsulated traffic. SR-aware snort is an extended version of snort that can apply snort rules directly to inner packet of SR encapsulated traffic. We show the interoperability between SR-aware and SR-unaware SFs by including both of them within the same SFC.**

***Index Terms*—Service Function Chaining, Network Function Virtualization, Segment Routing, Linux networking**

I. Introduction

Telecommunication networks infrastructures are evolving at a rate rarely seen since the transformation from analog to digital [1]. Network functions virtualization (NFV) offers an agile way to design and deploy networking service [2]. In an NFV infrastructure, network functions are decoupled from proprietary hardware appliances and moved to virtual servers so they can run in software modules called Virtual Network functions (VNFs), which are sometimes referred to as Service Functions (SFs). This dramatically reduces both capital expenditures (CAPEX) and operating expenses (OPEX) [3]. A set of these VNFs (SFs), which can be arbitrarily located in a distributed virtualization infrastructure, are often required to deliver an end-to-end service, hence Service Function Chaining (SFC) comes into play.

SFC denotes the process of forwarding packets through the sequence of SFs [4]. It requires a steering mechanism to force packets to go through SFs. These steering mechanisms often require inserting a new header into packets, which carries the path information. Network Service Header (NSH) and IPv6 Segment Routing header (SRH) are two examples of those headers. NSH has been proposed by the IETF SFC Working Group to support the encapsulation of packets with a header that specifies the sequence of SFs to be crossed [5]. Using NSH requires creating a state (per each NFV chain) in the network fabric, which doesn't make it the preferred solution in the recent era of networking, where everything is going towards stateless and simplicity. On the contrary, using SRH for SFC doesn't have the need for those state information.

In this demo, we consider the use of the Segment Routing (SR) architecture to support SFC. SR is a new network architecture that leverages the source routing paradigm [6]. It allows to steer packets through an ordered list of nodes, which are refereed to as segments. SR can be instantiated over both MPLS (SR-MPLS) and IPv6 (SRv6) data planes. SRv6 defines a new IPv6 Routing type, named SRH. It allows including a list of segments in the IPv6 packet header [7]. Each A segment is encoded as an IPv6 address and represents function to be called at a specific location in the network. SR enables SFC in a simple and scalable manner, by associating each SF with a segment. Such segments are combined together in a segment list to achieve SFC.

II. SR-aware vs SR-unaware SFs

SFs can be categorized into two types, depending on their ability to properly process SR encapsulated packets. These are respectively named SR-aware and SR-unaware SFs [8]. An SR-aware SF is able to correctly process SR-encapsulated packets it receives, which imply being able to process the original packet despite the fact that it has been encapsulated within a SR packet, but also being able to process the SRH. On the contrary, An SR-unaware SF is not able to correctly process the SR-encapsulated it receives. It may either drop the traffic or take erroneous decisions due to the unrecognized SR information. In order to include SR-unware SFs in an SR SC policy, it is thus required to remove the SR information as well as any other encapsulation header before the SF receives the packet, or to alter it in such a way that the SF can correctly process the packet. SR proxy is an entity, separate from the service, that performs these modifications and handle the SR processing on behalf of a SR-unaware service. Srext [9] is a Linux kernel module providing advanced SR functions. It supports different SR proxy behaviours detailed in [8].

III. SR/SFC Testbed

In order to showcase the SFC of SR-aware and SR-unaware SFs, we built the testbed shown in Figure 1, which consists of six nodes (R1-R6), implemented as Linux VMs and represent our SR domain. This SR domain is used to connect two branches (BR1 and BR2) of an enterprise to an external network (Ext). All nodes, except R4, support SRv6. Nodes R1 and R6 respectively represent the ingress and egress nodes, while nodes R2, R3 and R5 are used as NFV nodes of our

SRv6 based SFC scenario. Node R4 is a normal IPv6 Linux router. Service Functions (F1-F3), Branches (BR1 and BR2), and external network (Ext) are deployed as Linux network namespaces. F1 is an SR-aware Linux iptables firewall, F2 is SR-unaware snort, and F3 is an extended SR-aware version of snort. Nodes R1, and R4-R6 are running kernel 4.14 and have iproute2 v4.14 installed. Node R2 is running compiled Linux kernel 4.15-rc2 with SRv6 enabled and SERA firewall included [10]. Node R3 has the srext [9] kernel module installed. The links between any two nodes Rx and Ry are assigned IPv6 addresses in the form `fc00:xy::x/64` and `fc00:xy::y/64`. For example, the two interfaces of the link between R1 and R2 are assigned the addresses `fc00:12::1/64` and `fc00:12::2/64`. Each node owns an IPv6 prefix to be used for SRv6 local SID allocation, which is in the form `fc00:n::/64`, where `n` represents the node number. As an example, R2 owns the IPv6 prefix `fc00:2::/64`. SFs are instantiated on an SR SID of form `fc00:n::fk:/112` where `n` represents the node hosting the SF and `k` is the SF number. For example, F1 which is running in node R2 is given the prefix `fc00:2::f1:/112`. BR1, BR2, and Ext are respectively assigned the IPv6 prefixes `fc00:b1::/64`, `fc00:b2::/64`, and `fc00:e::/64`.

IV. SR/SFC POLICIES

The testbed in Figure 1 supports two different path, with different bandwidth and security guarantees, towards *Ext*. Path p_1 ($R_1 \rightarrow R_4 \rightarrow R_5 \rightarrow R_6$) provides high bandwidth. Path p_2 ($R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_6$) has lower bandwidth, but more security guarantees. Going through p_1 implies crossing F1 and F2. The same way, going through p_2 implies crossing F3. BR1 and BR2 have different traffic requirements; BR2 traffic is very delay-sensitive, while BR1 traffic is highly confidential, but less delay sensitive. We exploit p_1 and p_2 to satisfy those traffic requirement. BR1 traffic is steered through p_1, and BR2 traffic is steered through p_2. At the ingress node (R1), we configured two different SR SFC policies (CP1 and CP2) that steer traffic through p_1 and p_1. Policy Based Routing (PBR) is used to classify traffic coming form BR1 and BR2, which respectively go through CP1 and CP2.

V. DEPLOYMENT AND TESTING

We built our testbed by using *VirtualBox* [11] as hypervisor and *Vagrant* [12] as VM manager. This makes it easy to replicate the demo on any commodity hardware. Scripts required to deploy the demo are open source and can be found at [13]. To verify the deployment of the demo, we use *iperf* [14] to generate traffic from BR1 and BR2. BR1 traffic should cross both F1 and F2. F1 is configured with iptables rules, which are applied by SERA firewall directly to inner packet of received SR traffic. F2 is an SR-unaware snort, which can't correctly processes SR packets. We used srext to remove SR encapsulation from packets before being handed to F2. The removed SR encapsulation is re-added again to packets after being processed. F3 is the only SF crossed by BR2 traffic. It's an SR-aware snort that can apply configured snort rules directly to inner packet. To make sure that BR1 and BR2 traffic follows the exact path in both upstream and downstream, we configure two SR SFC policies on the egress node (R6) for the reverse path. This guarantees that SFs get the traffic in both directions.

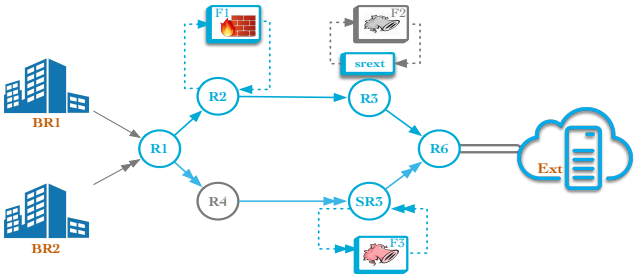


Fig. 1: Testbed for SR/SFC demo

VI. CONCLUSIONS

In this demo, we introduce a Linux NFV infrastructure that support SFC of both SR-aware and SR-unaware SFs. We used our SR proxy implementation (srext) to include those SR-unaware SFs in an SR SFC. As SR-aware SFs, we provided two implementations of SR-aware SFs. SR-aware and SR-unaware SFs have been included in the same SFC to show their inter-operability. We provided an open source implementation for the SR/SFC testbed been used.

REFERENCES

[1] B. Thekkedath, *Network Functions Virtualization For Dummies*. Wiley, 2016.

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[3] R. Mijumbi et al., "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, 2015.

[4] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," Internet Requests for Comments, RFC Editor, RFC 7665, October 2015.

[5] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)," Internet-Draft, November 2017. [Online]. Available: http://tools.ietf.org/html/draft-ietf-sfc-nsh

[6] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The Segment Routing Architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[7] S. Previdi (ed.) et al., "IPv6 Segment Routing Header (SRH)," Internet-Draft, September 2016. [Online]. Available: http://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-02

[8] F. Clad *et al.*, "Segment Routing for Service Chaining," Internet-Draft, October 2017. [Online]. Available: https://tools.ietf.org/html/draft-clad-spring-segment-routing-service-chaining-00

[9] "srext - a Linux kernel module implementing SRv6 Network Programming model," Web site. [Online]. Available: https://github.com/netgroup/SRv6-net-prog/

[10] "SERA - SEgment Routing Aware Firewall," Web site. [Online]. Available: https://github.com/SRouting/SERA

[11] "VirtualBox home page," Web site. [Online]. Available: http://www.virtualbox.org/

[12] "Vagrant home page," Web site. [Online]. Available: http://www.vagrantup.com/

[13] "SRv6 SFC demo," Web site. [Online]. Available: https://github.com/SRouting/sr-sfc-demo

[14] "iPerf - The ultimate speed test tool for TCP, UDP and SCTP," Web site. [Online]. Available: http://iperf.fr

Demo: Intent-Based 5G IoT Application Slice Energy Monitoring

*+Fred AKLAMANU, *Sabine RANDRIAMASY, +Eric RENAULT, *Imran LATIF, *Abdelkrim HEBBAR, *Alberto CONTE, *Bilal AL_JAMMAL, *Warda HAMDAOUI
*Nokia Bell Labs, Nozay, France
firstname.lastname@nokia-bell-labs.com

+Institut Mines-Télécom / Télécom SudParis, Samovar CNRS, France
firstname.lastname@telecom-sudparis.eu

***Abstract*—Current Telco Network Service Provisioning requires proficient expertise on Infrastructure equipment. Moreover the process is tedious and erroneous making Network Service lifecycle a daunting task for Network Operators (NOs) and 3rd Party Network Tenants. This paper, proposes an Over-The-Top Intent Based Network Framework for Network Slice Energy Monitoring and Provisioning. The aim is to automate the task of network slicing through a declarative approach known as Intents while hiding network complexity enabling NOs monitor Network Slice energy consumption. We provide an experimental validation of the Intent Framework with a scenario of a 5G IoT Application Network Slice energy monitoring.**

I. Introduction

The next generation mobile network, 5G is becoming a reality. The backbone or foundation of 5G is viewed as Software Defined Networking [1] and Network Function (SDN) Virtualisation (NFV) [2]. These technologies provide potential cost cutting .i.e. Capital Expenditure (CAPEX), Operational Expenditure (OPEX) and they will help Network Operators (NOs) maintain elastic networks to meet growing network demands.

5G will potentially provide a window for NOs to extend their infrastructure services to Mobile Virtual Network Tenants (MVNO) and Vertical Markets such as Over-The-Top Application Providers. The services envisioned by NOs to potential tenants will include physical or virtual shared end-to-end network resources known as a Network Slices. This encompasses network resources from Cloud Radio Access Network (CRAN) [3] through to Evolve Packet Core Network (EPC) [4] which traverses a transport network. Both NOs and Network Tenants will want an idea of how much energy these network slices consume. This will help NOs to bill their clients based on network slice energy consumption and other factors.

The current approach to realise network services by NOs requires proficient expertise on infrasturucture equipments and moreover the process is tedious and erroneous. They exert a manual method to deliver the network services to their clients i.e. MVNOs and vertical markets. Such an approach will not suffice on 5G mobile networks due to a 30 million expected connected devices. Intent-Based Networking (IBN) [5] aims to ease such challenge of manual network service provisioning through network automation. Networks Tenants will only have to specify their Intents i.e. "WHAT" network service while an Intent-Based Management Framework handles the automation process of realising the Intent, that is, the "HOW". An Intent in this paper refers to an Over-The-Top (OTT) Network Application Slice.

We propose an OTT Intent-Based Network Framework which provides a simplified and non-technical interface for MVNOs and vertical markets to request for OTT Network Application Slice without knowledge of the physical insfrastructure details such as configuration, topologies and protocols. The framework enables energy monitoring of such Network Application Slices.

II. Proposed Over-The-Top Intent Based Networking (IBN) Framework

The proposed IBN Framework provides network tenants and NOs with a simple service platform. The framework speeds up service request placement and provisioning, provides a feedback for network service feasibility and guarantees the platform reliability. The modules of the OTT IBN platform on top of a Cloud-Over-The-Top Application Slicing Platform (COASP) are shown in Fig. 1.

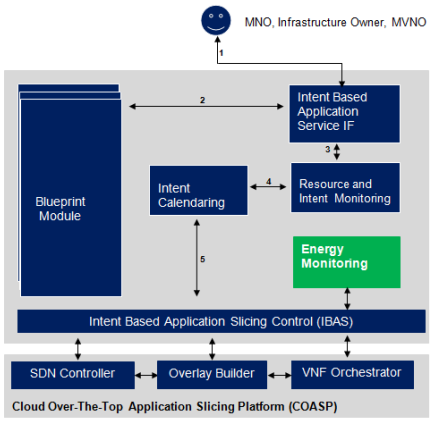


Fig. 1. OTT IBN Framework

We provide details on the energy monitoring module for VNFs on the OTT IBN Framework. Details of other

sub-modules of the framework are skipped here and provided in our related work on Intent-Based Real Time 5G Cloud Service Provisioning. We employ the term micro-services for VNFs. These micro-services are cloud native Docker containers [6] which provide numerous advantages over traditional virtual machine. They are lighter and setup time is in order of few seconds compared to minutes for virtual machines.

A. Energy Monitoring Module

This module is responsible for microservices energy consumption monitoring. The module comprises a Software Defined Network middleware power metering, PowerAPI [7]. This tool enables the power measurements of micro-services which will be important for NOs about possible VNF placement decisions and furthermore identify high energy consuming ones. The energy data will be potentially useful in the future for data analysis and learning purposes.

III. DEMONSTRATION

The demonstration involves the setup of the physical insfrastructure. This is made up of two Nokia Airframes Front-End Unit (FEU) and Edge Cloud (EC) and a laptop for Central Cloud (CC) interconnected by two switches. The table I shows the physical platform specifications.

	Front End Unit	Edge Cloud	Central Cloud
OS	Ubuntu 16.04	Ubuntu 16.04	Ubuntu 16.04
RAM (GB)	128	128	16
CPU Cores	24	24	8

TABLE I
CLOUD PLATFORMS

The next phase is the virtual network setup, deployment of a VNF Orchestrator, distributed key-value store, ETCD [8] and ONOS [9], an SDN controller as a docker container on the Central Cloud (CC). This phase also involves the installation of Open Virtual Switches (OVS) [10] on FEU and EC. ONOS is responsible for the management of the OVS to establish connectivity for the Virtual Network Infrastructure (VNI).

Phase two involves deployment of 5G helper VNFs, these are VNFs necessary for the deployment of 5G Network Application Slices. 3 VNFs and 5 VNFs are deployed on FEU and EC respectively by a VNF Orchestrator.

The above steps ensure that the VNI is properly set up to receive network service requests. The Network Tenant provides the type of service (Intent), which is a 5G IoT Application Slice from a GUI interface. Energy monitoring option is activated for End-to-End Network Slice monitoring. End-to-End Network Slice monitoring comprise energy monitoring of all micro-services on the VNI. The tenant request is transmitted to the Intent Engine for feasibility of service deployment. In the absence of any problem, a 5G IoT VNF is deployed as well as PowerAPI docker containers for individual microservices monitoring on the VNI. The individual consumption of the VNFs are summed and stored in a real-time database, influxDB. The total network slice energy consumption is displayed on a dashboard as shown in Fig. 2.

Fig. 2. 5G IoT Application Slice Energy Consumption

IV. CONCLUSION AND FUTURE WORKS

Our proposed OTT Intent Based Networking Framework simplifies network slice energy monitoring through automation. The NO does not need to manually setup different configuration for OTT Application Network Slice components for energy monitoring.

V. ACKNOWLEDGMENT

The power metering virtualization was done within the studies of the Celtic-Plus project SooGREEN [11].

REFERENCES

[1] M.-K. Shin, K.-H. Nam, and H.-J. Kim, "Software-defined networking (sdn): A reference architecture and open apis," in *ICT Convergence (ICTC), 2012 International Conference on*. IEEE, 2012, pp. 360–361.

[2] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," vol. 51, no. 11. IEEE, 2013, pp. 24–31.

[3] e. Imran Latif, "Cloud ran architecture for smart cities," in *The 1st American University in The Emirates International Research Conference (AUEIRC)*. Springer, 2017.

[4] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc)," vol. 28, no. 6. IEEE, 2014, pp. 18–26.

[5] SDxCentral, "Intent: Dont tell me what to do! (tell me what you want," February 2015. [Online]. Available: https://www.sdxcentral.com/articles/contributed/network-intent-summit-perspective-david-lenrow/2015/02/

[6] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," vol. 2014, no. 239. Belltown Media, 2014, p. 2.

[7] A. Bourdon, A. Noureddine, R. Rouvoy, and L. Seinturier, "Powerapi: A software library to monitor the energy consumed at the process-level," vol. 2013, no. 92, 2013.

[8] Core OS, "A distributed, reliable key-value store for the most critical data of a distributed system." [Online]. Available: https://coreos.com/etcd/

[9] ON.Lab, "ONOS intent framework," May 2016. [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework

[10] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar *et al.*, "The design and implementation of open vswitch." in *NSDI*, 2015, pp. 117–130.

[11] SooGREEN, "Service-oriented optimization of green mobile networks" (soogreen), celtic-plus project, partially funded by the french directorate general for enterprise (dge) and the scientific and technological research council of turkey (tubitak)," 2018.

Demo: Enabling In-Network Processing utilizing Nearby Device-to-Device Communication

The An Binh Nguyen, Christian Klos
Björn Richerzhagen, Ralf Steinmetz
KOM, TU Darmstadt, Germany
{firstname.lastname}@kom.tu-darmstadt.de

Christian Meurisch
TK, TU Darmstadt, Germany
meurisch@tk.tu-darmstadt.de

Patrick Lampe
University Marburg, Germany
lampep@mathematik.uni-marburg.de

***Abstract*—In disaster situations, relief work can be enhanced and facilitated by acquiring and processing distributed information. However, the communication and computation infrastructures might be impaired or inaccessible in emergency response scenarios. Consequently, approaches for coordination, and resource utilization are still challenging. To this end, we proposed the concept of an *adaptive task-oriented message template* (ATMT), that bundles the control information and the payload data, required to process and extract information, into a single message. Thus, an ATMT enables distributed in-network processing of complex tasks, allows to leverage the idle resources of mobile devices of the first responders. In this paper, we demonstrate the use of the ATMT concept in an example of face detection, which can be used to offer *Person Finder* similar services in an emergency ad hoc network. We utilize Google Nearby peer-to-peer networking API, standard and available on Android-based devices, to realize the handover of an ATMT message between mobile devices. This successful integration underpins the prospective adoption of the ATMT concept.**

I. Introduction

Acquiring and processing distributed information are crucial for disaster situations. Today, several services designed to enhance relief work, and to offer information relevant for emergency situations, such as Google's Person Finder [1] or Facebook's Crisis Reponse [2], are available. However, these services require stable Internet-based communication, which might not be possible in disaster situations. To maintain communication in emergency situations, mobile hand-held devices such as smart phones can be used to create an opportunistic ad hoc network [1], which allows mobile devices to share data and exchange information through device-to-device communication. Combining with the built-in sensors, and the computing resource available on mobile devices, it is possible to provide emergency relief services on top of mobile opportunistic ad hoc network. Hereby, approaches to enable distributed coordination, and efficient resource utilization are necessary. For this purpose, we proposed and designed a message template, called *adaptive task-oriented message template* (ATMT) in [2].

An ATMT message describes a complex task, the operations and the payload data required to complete the defined complex task; which makes each ATMT message a self-encapsulated message. Hence, the ATMT message template is able to support distributed processing, leveraging idle resource of the participating mobile devices. Additionally, since an ATMT message is self-encapsulated, each participating device can make an autonomous decision based on its available capability, and its available resources. Overall, the ATMT message provides a basis to create complex services on an opportunistic network, utilizing mobile devices.

To showcase the advantages, and the applicability of the ATMT concept in practice, we provide a demonstration, that (i) uses ATMT message template to implement a face detection technique, which requires multiple-processing stages, specially designed for mobile devices [3], and (ii) utilizes the Google Nearby Networking API [4] for enabling handover of an ATMT message directly between devices.

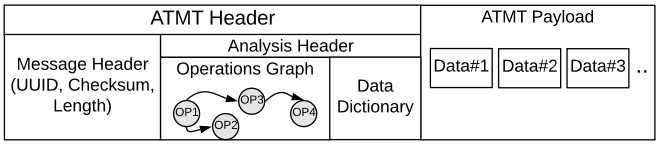


Fig. 1: Structure of the adaptive task-oriented message template (ATMT) as proposed in [2]

II. Adaptive Task-oriented Message Template

The design and construction of the ATMT message template are illustrated in Fig. 1. The goal in designing ATMT message template is to allow a user to define a processing goal, and the steps/operations required to reach this goal. These information are captured in an *operations graph*, modeled by a *directed acyclic graph* within the *ATMT header*. A device can check on the processing status of the *operations graph*, by reading only the *checksum* field, without reading the whole content of the ATMT message. This allows a device to make a fast decision on whether to merge, to drop, or to handover an ATMT task. To incorporate the payload data required for the operations, considering the resource constraints of the devices in an opportunistic mobile ad hoc network, we devise the *ATMT payload* to contain pieces of data, which can be compressed by different encoding methods to allow for more flexibility. Each piece of data is mapped to an operation

[1] https://google.org/personfinder
[2] https://www.facebook.com/about/crisisresponse/

in the *operations graph* through the *data dictionary* in the ATMT header. To organize the coordination among mobile devices, we conceive four roles, which can be assumed by the participating devices according to their available capabilities. These roles are *sensor node* to obtain data, *delegator node* with domain knowledge to set up and adapt the *operations graph* if necessary, *operator node* which executes one or more operations from the operations graph based on its available resource, and *forwarder node* which receives, store and forward an ATMT message. Distributed processing of a complex task is realized by simply passing ATMT messages. Each device, receiving ATMT messages, will act accordingly to its role, adjust the content of ATMT messages w.r.t. the current processing state, and handover the adjusted ATMT message to the other.

III. Integration of ATMT with Google Nearby

Support to setup and to provide device-to-device communication using mobile devices such as smart phones is still restricted. To enable WiFi ad hoc communication between Android devices, these are required to be rooted. Even though WiFi direct allows for direct communication, but it requires a master-slave model, which makes transmission of a message through multi-hops difficult [1]. Recently, Google enables and provides *Nearby* [4], a peer-to-peer networking API that allows for discovery, connection and data exchange with devices in the close vicinity. We use *Nearby Connections* to let each device advertise its role/service. Thus, each participating device is able to look for the next role/service, that it requires. For instance, a *sensor node*, after acquiring data, needs to look for a *delegator node*, so that the *delegator node* can setup and include the corresponding *operation graphs* into the ATMT message. Similarly, a *delegator node* searches for *operator nodes* to execute the operations on the obtained data. If an *operator node* cannot complete the task described in ATMT message alone, this *operator node* will look for further *operator nodes* which possess the capabilities, e.g., special hardware, special algorithms/libraries, to take over the upcoming operations. When an *operator node* notices the completion of the ATMT task, it can look for *forwarder nodes* to transport the final result to a predefined destination. In this manner, a multi-stage processing is realized through multi-hops device-to-device communication.

IV. Scenario and Demonstration

We use the ATMT concept to implement the face detection technique proposed in [3], which can be used to support *person finder* service in an emergency ad hoc network. The face detection technique as introduced in [3] relies on multiple stages pipeline to detect multiple faces within an image; these are (1) preprocessing to handle parameters, (2) a fast face detection to determine important regions within the image, and (3) a validation phase using *dlib* library to detect and validate the faces within the image.

Despite the fact, that the face detection technique in [3] is designed considering the resource and energy constraint of a mobile device, this technique can be further enhanced using the ATMT concept. The processing pipeline as described can be distributed to multiple devices; consequently, each device has to process only one phase of the pipeline. The advantages for the utilization of the ATMT concept in this scenario are (i) further reducing the resources consumption of the participating devices, since now each device has to process only one phase of the pipeline, and (ii) being able to leverage heterogeneous capabilities, for instance, in the described scenario, the *dlib* library might not available on all devices.

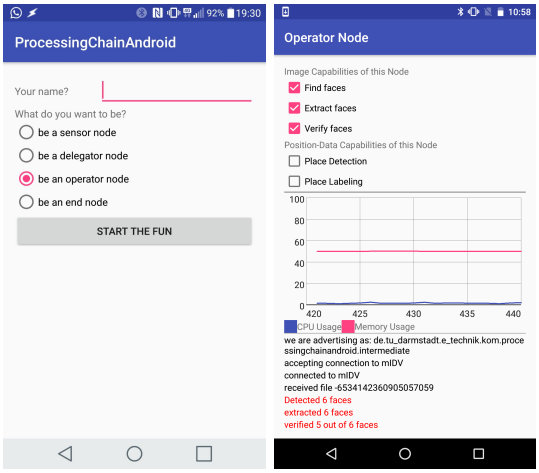


Fig. 2: User interface of the Android mobile devices used in the demonstration

In the demonstration, a user can use a *sensor* device to capture and send a picture containing multiple faces to other devices for further processing. Next, the device chosen as a *delegator node* should receive the picture, adds the *operations graph* for the face detection technique accordingly, and forwards the constructed ATMT message to the *operator devices*. On each *operator* device, the user can choose which operations should be available, and can observe through log messages, how the devices handle ATMT messages, and how the *Nearby Connections* are used (cf. Fig. 2). At the end of the processing pipeline, the results of the face detection can be seen in the chosen end-node device, showing the cropped images of different detected faces. A short video, showing the demonstration as described, can be found at the following link http://bit.ly/2GGenHZ

References

[1] S. Trifunovic, M. Kurant, K. A. Hummel, and F. Legendre, "Wlan-opp: Ad-hoc-less opportunistic networking on smartphones," *Ad Hoc Networks*, vol. 25, 2015.

[2] T. A. B. Nguyen, C. Meurisch, S. Niemczyk, D. Böhnstedt, K. Geihs, M. Mühlhäuser, and R. Steinmetz, "Adaptive Task-oriented Message Template for In-Network Processing," in *NetSys'17*. IEEE, 2017.

[3] P. Lampe, L. Baumgärtner, R. Steinmetz, and B. Freisleben, "Smartface: Efficient face detection on smartphones for wireless on-demand emergency networks," in *IEEE ICT*, 2017.

[4] "Google Nearby." [Online]. Available: https://developers.google.com/nearby/

DEMO: Controlling software router resource sharing by fair packet dropping

Vamsi Addanki, Leonardo Linguaglossa, Jim Roberts, Dario Rossi
Telecom ParisTech, Paris

***Abstract*—We demonstrate a practical way to achieve multi-resource sharing in a software router, where both bandwidth and CPU resources may be bottlenecks. Our main idea (published in a same-titled paper in this year IFIP Networking conference [1]), is to realize per-flow max-min fair sharing of these resources by wisely taking drop decisions according to the state of a shadow system. We implement our FairDrop proposed algorithm in Vector Packet Processor (VPP), a novel high-speed software router architecture. We demonstrate FairDrop is capable of fairly sharing CPU cycles among flows with heterogeneous computing workload, at 10Gbps on a single core.**

I. Introduction

Controlling how bandwidth is shared between concurrent flows is a classical issue in networking, and the advantages of imposing fairness have been repeatedly discussed since Nagle's pioneering work [2]. More recently, the blending of networking and computing raise new challenges [3] in terms of resource contention and sharing – however, simple mechanisms that are capable of handling heterogeneous resources have yet to appear. In emerging high-speed software routers, flow throughput may additionally be impeded by network capacity limitations as well as other resources, such as the amount of available CPU cycles to process packets of any given flow: in this case, it would be desirable in this case to impose per-flow fair throughput expressed in cycle/s[4].

As in[4], we advocate that flexible dropping algorithms are an attractive solution to control resource sharing, be it cycles of a multi-core CPU or network bandwidth. We implement a simple and practical algorithm, which we refer to as FairDrop (FD), that realizes max-min fair flow rates while retaining the network interface card (NIC) and server code optimizations that are necessary to keep up with line speeds of 10 Gbps on a single CPU core. These optimizations notably require packets to be batched for both I/O and processing making implementation of classical scheduling algorithms like DRR [5] problematic if not impossible, as argued in [3].

Our proposal is then to realize fairness via a *shadow system.* Briefly, suppose packets are handled simultaneous by two service systems, one the actual buffer management system implemented in the router (e.g., a DPDK circular ring), the other a shadow system implementing a more sophisticated scheduler (e.g., per-flow FQ). Packets that are dropped in one system are also dropped by the other so that both systems yield exactly the same rate over the lifetime of a flow. The shadow system in our proposal is virtual and makes dropping decisions based on a measure of per-flow virtual queue occupancy. This measure is depleted between packet arrivals, at a rate that varies depending on the number of active flows, and incremented by packet length on the arrival of every batch. In particular, if the shadow system implements per-flow head-of-line processor sharing, the long-term flow rates will be max-min fair.

We implement the above proposal in Vector Packet Processor (VPP), an software router released as open source in the context of the FD.io Linux foundation project. For a detailed explanation of our FairDrop (FD) algorithm we refer the interested reader to a same-titled paper in this year IFIP Networking conference [1]. In this extended abstract we instead describe the experimental environment and scenarios that we will demonstrate, contrasting results achieved under simple buffer management policies (such as FIFO or NIC ring buffers). More information about the project, as well as our implementation, is available at [6].

II. FairDrop implementation and demonstration

In a software router, a CPU core becomes a bottleneck when flows emit packets too fast yielding a compute load greater than the CPU capacity, leading to packet drops. High-speed software routers are intrinsically flow-aware: flow-awareness is facilitated by NICs implementing receive side scaling (RSS), that hashes the 5-tuple and maps packets to distinct virtual queues, mainly for the purpose of load balancing over multiple CPU cores. Individual threads of packet processing applications are bound to a CPU core and, using kernel-bypass stacks such as DPDK, threads consume independent streams of packets, each from a different RSS queue. Additionally, high-speed software routers and their NICs generally deal with packets in *batches* rather than individually, which reduces interrupt pressure and that is a necessary optimization for line-speed packet processing. Software routers typically polls for available packets in the NIC circular buffer, grabbing and processing the whole batch before the next poll. FairDrop operates over packet batches at the router ingress.

We demonstrate FairDrop with a scenario where N flows share a C=10Gbps link and are processed by a single CPU core clocked at 2.6GHz. Particularly, flows have equal input rate C/N but different treatment cost. For the sake of simplicity, in the demonstration we consider only two flow classes: the majority of the flows belong to the light-weight class C_L (e.g., Ethernet switching or IPv4 forwarding), whereas few flows belong to a heavy-weight treatment class C_H (e.g., IPsec or stateful L4 operation). In particular, we select functions whose $C_H/C_L \approx$10 so that a single packet of an heavy-weight flow

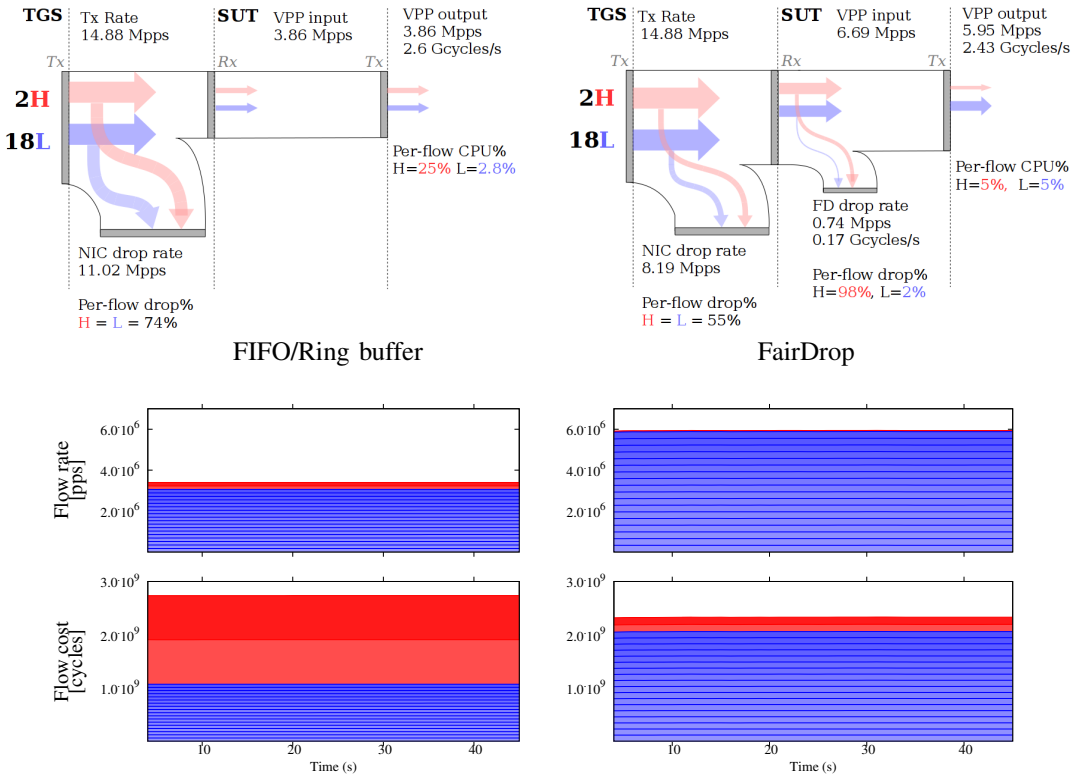


Fig. 1. Illustration of Classical (left column) vs FairDrop (right column) operations. Top part reports sankey diagrams of the rate at the Traffic Generator (TGS) and at the System Under Test (SUT). Lower part depicts the time evolution of the flow rate (in Mpps, middle) and the flow ccost (in cycles, bottom).

requires as many CPU cycles as about 10 packets of light-weight flows. We additionally fix $N_H = 2$ and $N_L = 18$ so that out of the total $N = 20$ flows, the N_H flows of class C_H requires as many processing cycles as the N_L flows of class C_L. Needless to say, 64B packets are sent to the maximum rate of 14.88Mpps, so that not all flows can be processed with the CPU budget.

We represent experimental results of the demo with the visual layout of Fig.1, where plots in the left column represent the case of traditional buffer management, and plots in the right column report the FairDrop case. In particular, the top plots report a sankey visualization of the experiments, whereas the bottom plots report the individual flow rate (in packets per second) and the individual flow cost (in cycles per second). The two heavy-weight flows are represented in red, and the 18 light-weight flows in blue.

In the traditional case, since the CPU budget is not enough to process packet of all flows, about 74% of packets are lost at the NIC before entering the VPP router. Given that flows have equal rates, there is no loss differentiation at the NIC, so that only about 3.86Mpps exit the VPP router, consuming the 2.6Gcycles/sec budget of our CPU. Notice that each flow have equal rate, but that a single heavy-weight flow alone consumes 25% of the CPU budget.

Conversely, the FairDrop mechanism preferentially drops packets of the heavy-weight flows to reinstate fairness (at a rate approximately 10 times higher). Dropping decisions have a cost (i.e., the packets need to be fetched from the NIC, the queue in the shadow system is updated, etc.) and FairDrop consumes 0.17Gcycles/sec. The net result of fair dropping decisions, more light-weight packets are processed in the router: this increases the overall throughput at 5.95Mpps (top right plot), reducing the drops at the NIC buffer, and reinstates per-flow fairness in terms of the number of cycles (bottom right plot).

The demonstration will allow to interact with the VPP router configuration (e.g., FairDrop vs classical ring management) and altering the scenario parameters (e.g., number of flows, relative cost, etc.) to contrast the key performance indicators under both approaches.

Acknowledgments

This work was funded by NewNet@Paris, Cisco's Chair "Networks for the Future" at Telecom ParisTech.

References

[1] V. Addanki, L. Linguaglossa, J. Roberts, and D. Rossi, "Controlling software router resource sharing by fair packet dropping," in *IFIP Networking*, 2018.
[2] J. Nagle, "On packet switches with infinite storage," RFC 970, 1985.
[3] K. To, D. Firestone, G. Varghese, and J. Padhye, "Measurement based fair queuing for allocating bandwidth to virtual machines," in *ACM HotMiddlebox*, 2016.
[4] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker, "Approximate fairness through differential dropping," *ACM SIGCOMM Comput. Commun. Rev.*
[5] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," *SIGCOMM Comput. Commun. Rev.*
[6] https://newnet.telecom-paristech.fr/index.php/fairdrop/.

Poster: Impact of Prioritized Network Coding on Sensor Data Collection in Smart Factories

Marie Schaeffer, Roman Naumann, Stefan Dietzel, and Björn Scheuermann
Humboldt-Universität zu Berlin, Germany
Email: {marie.schaeffer, roman.naumann, stefan.dietzel}@hu-berlin.de, scheuermann@informatik.hu-berlin.de

***Abstract*—Utilizing information from the production process is integral to smart factory concepts. In our example use-case, plastic industry, sensor information from the injection molding process helps to detect defective parts and provides automated guidance for process set-up. An enabler for such applications is a means to wirelessly collect machines' sensor information in harsh factory environments, and network coding has been proposed as a tool to implement suitable network protocols. Using pre-recorded sensor data from actual injection processes, we study the impact of network coding on the latency of sensor data collection. In particular, we show how network coding with prioritization helps to reduce delays until information becomes usable.**

I. Introduction

Many smart factory use cases strive to automate previously manual tasks via the utilization of highly detailed process information. In our example use-case, plastic injection molding, molten plastic is injected with high pressure and temperature into a form, termed the "mold." As the plastic cools down, the final product hardens out and is finally ejected from the mold. Here, relevant process information includes material pressure and temperature measured within the mold. Such information, in combination with machine learning techniques, allows the automated detection of a variety of product defects before they can reach the customer [1], [2].

In order to leverage process information, it has to be collected quickly from machines throughout the factory. A centralized server then acts upon results and, for example, issues alarms to operators should the process become unstable. Wireless transmission of sensor information is preferable, because it avoids expensive retrofitting of factories. Wireless transmission, however, can be difficult due to the harsh factory environment with metal obstruction and widespread factory areas that necessitate multi-hop capabilities.

Using network coding in our use case can improve the throughput, simplify routing decisions, and add robustness against packet loss. But using random linear network coding (RLNC) to transmit sensor information may result in intolerable delays due to the "all-or-nothing" property. This property states that it is highly unlikely that the server can decode parts of the sensor information before a sufficient number of linear combinations for, in our case, a complete injection cycle are received. A number of prioritized network coding schemes have been proposed to allow early decoding of a subset of a generation's information.

We study the impact of two prioritized network coding techniques – hierarchical network coding (HNC) [3] and iNsPECt [4] – on delays in sensor data collection. As a third mechanism, regular RLNC [5] serves as a baseline for our comparison.

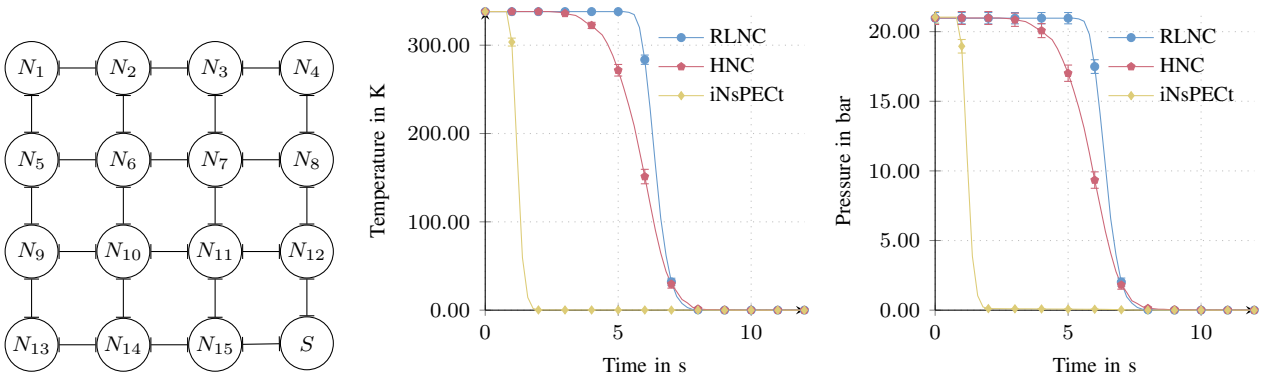
II. Encoding and Transmission Schemes

Using regular RLNC as an example, we explain how network coding in general can be applied to our sensor data collection use case. We then briefly introduce the two prioritized network coding mechanisms used in our comparison.

RLNC splits information into generations of data messages. In our case, a generation is one production cycle's worth of sensor information from a single sensor. Each message is a set of sensor samples and consists of several symbols over a finite field. We employ the common finite field $\mathbb{F}_{2^8}$, as it combines efficient byte alignment with sufficient protection from linear dependency. Each machine generates linear combinations of one generation's messages using *random* coefficients. Each machine then continually broadcasts these linear combinations until all neighbors can decode the current generation. Subsequently, the next generation is sent.

To apply prioritized network coding techniques to our industrial use case, we pre-process sensor information such that it can be divided into different priority layers, as described in [6], We apply discrete cosine transform (DCT) to each production cycle's sensor information and divide its output into blocks of coefficients. Blocks with low-frequency coefficients provide an early preview of a complete sensor cycle, whereas blocks with high-frequency coefficients incrementally increase precision to enable more demanding detection techniques. We again use one injection cycle as a generation, but we use blocks of coefficients as the prioritized network coding mechanisms' prioritization layers. To generate a linear combination associated with a given priority layer, the prioritized network coding (PNC) codes combine only messages of equal-or-lower layers. In our case, this concept translates to only lower-or-equal frequencies of the DCT-provided spectrum of sensor information. As the prioritized layers form a linear subspace in the decoding matrix, they can generally be decoded earlier and, therefore, reduce delays in data processing.

In our evaluation, we study the impact of HNC [3], a PNC protocol, on the delay after which information is usable by the central server. We also study the impact of layer selection, a central aspect of PNC protocols, on decoding delay. To that extent, we compare HNC, which selects priority layers at random, with iNsPECt [4], which employs limited knowledge



(a) Grid topology with one sink.

(b) Temperature error.

(c) Pressure error.

Fig. 1. Topology and average sensor error over time.

on neighboring network nodes' decoding states to determine ideal layers.

III. Factory Network Model and Evaluation

We use a wireless network model in which nodes broadcast their messages. Our topology, given in Figure 1a, represents a typical factory layout with rows of machines in a regular grid and node distances of 30 m. The fifteen nodes, N_1 to N_{15}, represent the machines where the sensor data is measured. One sink node, S, is the factory's central server system. We consider a single sensor for each machine. More sensors in machines bring a constant factor for the amount of required transmissions, analogous to a higher sample rate.

We evaluate using the discrete event network simulator ns-3 (version 3.25) with YANS Wifi model, 802.11g MAC, and 2.4 GHz PHY using log-distance propagation loss model ($\gamma = 3.0$, which is in line with a range modern factory environments [7]) combined with Rayleigh fast fading. We use real, pre-recorded sensor information from the injection molding process. Our sensor information stems from a 25 s long production cycle that was sampled at 500 Hz rate. Each measured sample is a 4 B floating-point number. We split frequency components into five priority layers with a generation size of 53 frequency components to limit each data message's size to 1008 B. For the PNC-iNsPECt variant, we set the data-feedback ratio to $1 : 2$. Each sample shown in the following is the average over five simulation runs of 200 s simulated duration each, using different sub-streams of ns-3's PRNG. Error bars depict 95% confidence intervals (assuming normal distribution), but might not be visible if the error is negligible. During each run, several production cycles are transmitted to the sink.

Figures 1b and 1c show the simulation results for temperature error over time and pressure error over time. The time measurement starts with the first message being transmitted, which explains the initially very high average error that results from production cycles without any frequency components decodable at the server. Generally, it can be seen that the preview provided by the PNC scheme iNsPECt quickly gains precision and is virtually indistinguishable from the original sensor information much earlier than RLNC can provide any information. HNC also gains precision more quickly on average than RLNC. The overhead of the HNC scheme, however, results in RLNC providing the full picture before HNC can lower the remaining error below 1 K or 1 bar. In contrast, PNC achieves such a low average error approximately four times as fast as RLNC for both temperature and pressure readings. The maximum time until each production cycle was available with full precision was 8.40 s with our baseline RLNC. As a result of the principal message overhead imposed by PNC schemes, iNsPECt and HNC required up to 9.20 s and 17.80 s, respectively, until the preview reached full precision. Especially with iNsPECt, however, the error is extremely low during the time after which RLNC finished transmission.

IV. Conclusion

We studied the impact of prioritized network coding for smart factory use-cases using real sensor information from plastic industry. Our results suggest that iNsPECt provides significant benefits over non-prioritized RLNC, whereas HNC can only provide a coarse preview before RLNC provides the full picture.

References

[1] B. Ozcelik and T. Erzurumlu, "Comparison of the warpage optimization in the plastic injection molding using ANOVA, neural network model and genetic algorithm," Feb. 1, 2006.

[2] H. Oktem, T. Erzurumlu, and I. Uzman, "Application of Taguchi optimization technique in determining plastic injection molding process parameters for a thin-shell part," 2007.

[3] K. Nguyen, T. Nguyen, and S. c Cheung, "Peer-to-peer streaming with hierarchical network coding," in *2007 IEEE International Conference on Multimedia and Expo*, Jul. 2007.

[4] M. Schaeffer, R. Naumann, S. Dietzel, *et al.*, "Hierarchical Layer Selection with Low Overhead in Prioritized Network Coding," in *2018 IFIP Networking Conference (IFIP Networking)*, 2018.

[5] T. Ho, R. Koetter, M. Medard, *et al.*, "The benefits of coding over routing in a randomized setting," 2003.

[6] R. Naumann, S. Dietzel, and B. Scheuermann, "INFLATE: Incremental wireless transmission for sensor information in industrial environments," in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, Dec. 2015.

[7] S. Phaiboon, "Space Diversity Path Loss in a Modern Factory at frequency of 2.4 GHz," *WSEAS Transactions on Communications*, 2014.

Poster: Design and Evaluation of a Time Efficient Vertical Handoff Algorithm between LTE-A and IEEE 802.11ad Wireless Networks

Sina Rafati Niya, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
Email: [rafati | stiller@ifi.uzh.ch]

***Abstract*—LTE-A and IEEE 802.11ad are two of the networks that can have complementary roles in 5G networking. Thus, this paper proposes a time efficient, predictive, and dynamic Handoff (HO) algorithm between these two protocols. The algorithm presented measures the Signal to Interference Plus Noise Ratio (SINR), the Reference Signal Received Quality (RSRQ), velocity, and the Time of Stay (ToS) for pedestrian mobile users and calculates the best Time to Trigger (TTT) value accordingly. One of the main advances of this algorithm is that the TTT value is calculated dynamically regarding user's velocity and Handoff Failure Ratio (HoFR). Comparisons with other algorithms in this area determine that the algorithm proposed gains the least HoFR for these two protocols by avoiding unnecessary handoffs.**

I. Introduction

One of the major research areas in 5G is the offloading process. With offloading, user traffic traverses the local wireless AP instead of utilizing a continuous connection to cellular networks even in indoor areas. In case of using LTE-A in 5G, a proper network for offloading LTE-A communications with up and downlink data rates of higher than 1 Gbps needs to support the same data rates, otherwise, users will not be willing to switch to the local networks. One of the wireless technologies to support high data rates and being deployed in indoor areas as a replacement of traditional WiFi networks is the IEEE 802.11ad standard. This protocol is known as WiGig because of the Gigabit scale data rates it supports. WiGig provides almost 7 Gbps for downlink and almost 3 Gbps for uplink [2].

Switching the user's network from a home (already connected) network, to a new target network (one of the possible networks to be switched to) and vice versa is known as Handoff (HO) or Handover. In this work, a new time-efficient HO algorithm is designed to provide specifically a seamless connection and offloading between LTE-A and WiGig networks as two of the networks might be used broadly in 5G for high data rates . Results of a comparison with other HO algorithms reveal (cf. Section IV) that the algorithm proposed is capable of reducing Handoff Failure Rates (HoFR) in a predictive fashion. Also, the Time to Trigger (TTT) is updated frequently based on measures defined, and the HO process follows a cross-layer algorithm to increase the time efficiency.

II. Simulation Scenarios, Parameters and Evaluation

Simulation of the scenarios done in Matlab. The focus of this work laid on two scenarios: (1) HO process of a user connected to the LTE-A cell and moves toward the WiGig network as presented in Figure 1. (2) HO process of a user connected to the WiGig network and moves toward the LTE-A cell as presented in Figure 2. Parameter used in simulations are listed in Table 1.

TABLE I
Simulation Parameters

Simulation Parameters	Symbol	Value
eNB Number	N_{LTE}	1
WG-AP Number	N_{WG}	1
Simulation Duration	T_{sim}	1000 s
RSRQ Threshold	$RSRQ_{th}$	19.5 dBm
SINR Threshold	$SINR_{th}$	25 dB
LTE-A eNB Transmission Power	P_{LTE}	30 dBm
WG Transmission Power	P_{WG}	10 dBm
LTE-A Bandwidth	B_{LTE}	100 MHz
WG Bandwidth	B_{WG}	2160 MHz
LTE-A Antenna Height	h_{eNB}	40 m
WG Antenna Height	h_{WG}	1.5 m
Mobility Model	———	Gauss-Markov and LPP
Initial TTT	TTT_i	0.1 s
LTE Frequency	f_{c-LTE}	2100 MHz
WG Frequency	f_{c-WG}	60 GHz
LTE Radius	LTE_r	Whole Simulation Area
User Velocity	V_u	$[0-5]m/s$
Data Transfer Direction	———	Downlink
Number of LTE users	N_{U-LTE}	(1-50)
Number of WG users	N_{U-WG}	(1-50)

The HO algorithm proposed is memory-and-time efficient, and managed in a cross-layer fashion. Number of unnecessary HO and HoFR are managed by TTT value. Being a proactive algorithm, users' mobility, including their next location, speed, and angle of movement, are estimated using the Gauss-Markov mobility model, which is updated and readjusted by the accurate data received from Location Positioning Protocol (LPP). This update increases the precision of the Gauss-Markov model for upcoming estimations.

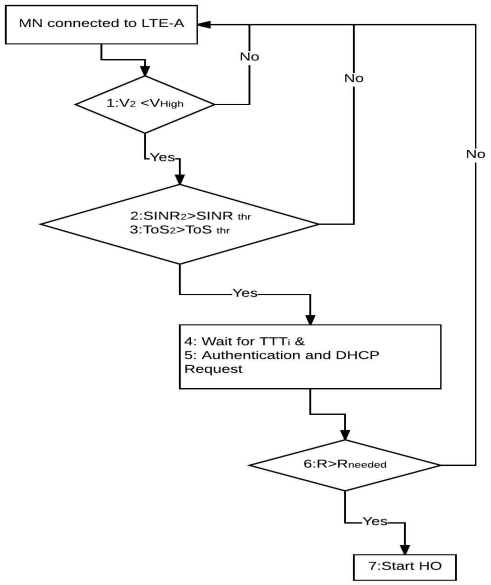


Fig. 1. HO Process: Moving From LTE-A To WiGig

To be computationally efficient, this algorithm does not continuously gather information from target and host networks, instead time intervals are set dynamically according to the amount of TTT to gather information from networks. The algorithm reacts to HoFR increase by adjusting the TTT value. Besides lowering the computational complexity, a binary search is used in TTT calculations to provide faster converges than with a linear method by the order of O(log(n)), which leads a very practical application.

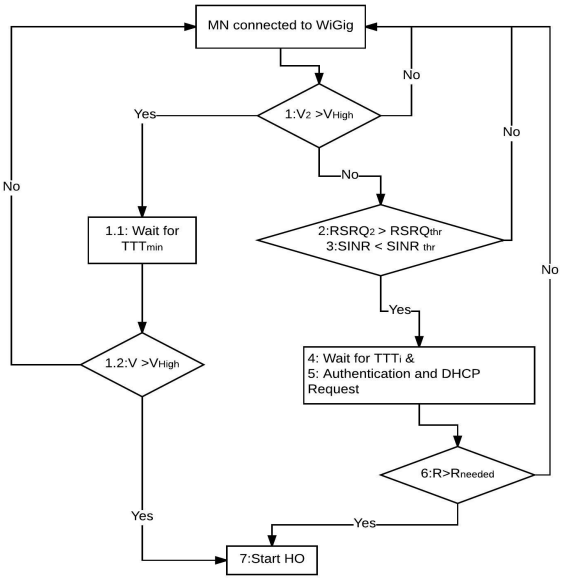


Fig. 2. HO Process: Moving From WiGig to LTE-A

To avoid continuous monitoring of various parameters, this algorithm specifies a high priority to users' velocity. For that reason, checking other variables such as SINR, RSRQ, and TTT is done only, if the user's velocity is in a specific range. Being sensitive to the user's velocity, this new algorithm performs better in comparison to other HO algorithms, especially within the decision making phase for high user velocities.

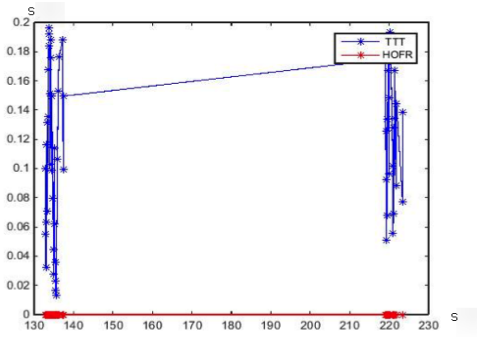


Fig. 3. TTT Variation Using Method of [3] With User Velocity Larger than 3 m/s.

In this algorithm, base (home) and target networks and User Equipment (UE), work together to reduce the process load on each side and calculate the parameters used in decision-making phase. finally, UE decides to start or deny the HO process based on her velocity, current TTT, Reference Signal Received Quality (RSRQ), and Signal to Interference Plus Noise Ratio (SINR). Also, this model is not memory bounded and with only few bytes of memory the user's exact location can be estimated.

Finally, as presented in Figure 3, the proposed algorithm manages to handle the HO process while keeping the HoFR rate close to 0. This is done with keeping the TTT amount in less than 0.2s which will end in a seamless connection. Comparisons with similar HO algorithms [1] revealed that other algorithms cannot be used in scenarios between LTE-A and WiGig networks with the goals of time-efficiency and low data loss during a HO process. High (or constantly increasing) TTT values of other algorithms make them inappropriate to be used for managing the HO processes for these scenarios and in most cases HoFR could not be controlled or reduced by employing the methods used by them.

REFERENCES

[1] S. R. Niya, B. Stiller, "Design and Evaluation of a Time Efficient Vertical Handoff Algorithm between LTE-A and IEEE 802.11ad Wireless Networks," IFI Tecnical Report No.2018.03, Zürich, Switzerland, Tech. Rep., April 2018. [Online]. Available: https://files.ifi.uzh.ch/CSG/staff/Rafati/Handoff-IFI-2018.03.pdf

[2] H. Peng, K. Moriwaki, Y. Suegara, "Macro-Controlled Beam Database-Based Beamforming Protocol for LTE-WiGig Aggregation in Millimeter-Wave Heterogeneous Networks," in *IEEE 83rd Vehicular Technology Conference (VTC Spring)*, May 2016, pp. 1–6.

[3] J. Xu, Y. Zhao, X. Zhu, "Mobility Model Based Handover Algorithm in LTE-Advanced," in *10th International Conference on Natural Computation (ICNC)*, Aug 2014, pp. 230–234.

Poster: WebMaDa 2.0 - Automated Handling of User Requests

Corinna Schmitt, Dominik Bünzli, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
[schmitt|stiller]@ifi.uzh.ch, dominik.buenzli@uzh.ch

***Abstract*—Today users want to monitor their networks remotely and adjust privileges immediately. Addressing the first request is not a big problem anymore, because many applications offer such solutions by default (e.g., via a special app to be installed or a browser-based solution). The immediate privilege handling is the challenge nowadays, because usually a global administrator in the background needs to be included in the workflow. This is required, because he is the only person who has the full overview of the tool the network is included. In general this is a nice idea, but introduces delays to the privilege management depending of the number of networks linked to the system, in total. WebMaDa 2.0 overcomes this bottleneck by introducing an automated request handling solution to a web-based framework for monitoring sensor networks remotely as well as supporting privacy and immediate handling of privilege requests.**

***Keywords*- WebMaDa, automation**

I. Introduction

Today, many different devices are connected with each other building small networks that are part of the Internet of Things (IoT). Such networks are designed for individual solutions specialized for a specific purpose (e.g., environmental monitoring, health monitoring). Devices used show heterogeneity concerning hardware and software and are linked to a specialized solution allowing analysis and visualization of data collected. This itself is nothing really new within the IoT community. But the requests of users and network owners changed over time towards (1) mobility support, (2) ownership and controlling of data, as well as (3) updating granted privileges immediately.

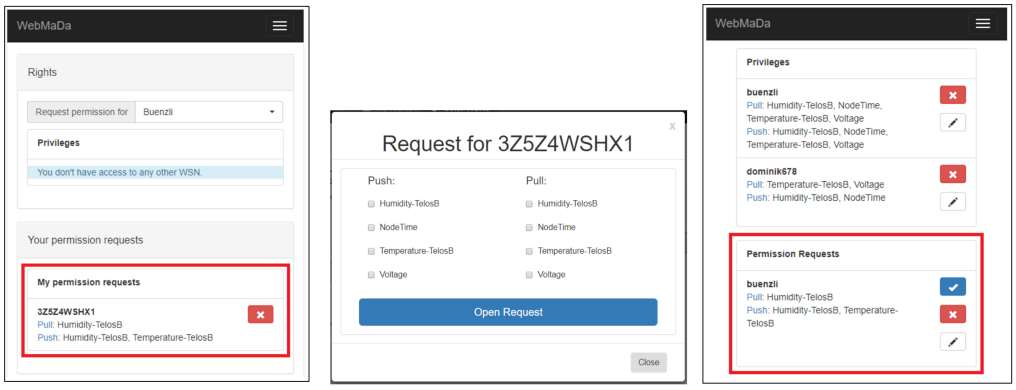
Many specific solutions are in place addressing the mobility request installing a special application on the mobile device. In general this is a good solution, but these solutions usually have special requirements to the operating system of the device and can exhaust the device quickly when running. The later can be overcome by integrating energy saving solutions, but still the applications require much memory of the device. To overcome this, web-based solutions are thought of beeing most suitable, because they only require Internet access and a browser installation on the device. Fortunately, both can be considered to be available by default on mobile devices. Furthermore, the code base only has to be updated in one place, thus reducing the cost for maintenance. The urge for control and ownership of the collected data is manifesting itself more and more in the minds of users. This is due to increased media coverage of data abuse caused by data leaks and the possibility of having data analyzed and visualized by third-party providers. Together with this situation comes the users' request to update granted privileges to manage access to the data collected. This is challenging, because access granted to applications can hardly be revoked or updated immediately if at all. Thus, the call for solutions supporting data and access control immediately arise. The aforementioned three issues (1)-(3) are addressed by WebMaDa, a Web-based Management and Data Handling Framework for sensor networks. The development started in 2014 with a basic support of mobile access to owned sensor networks allowing visualization of collected data in a flexible and hardware independent manner [2]. In 2016 WebMaDa received an update addressing the general request of fine-grained access management and pulling data in emergency cases [3]. The drawback was that each request (e.g., create networks, access to foreign networks, to view or pull data) required interaction of a global administrator introducing delay into the system. This drawback has now been solved in WebMaDa 2.0 [1] by automating the request handling within the system allowing immediate handling without the involvement of a global administrator. At the same time, the request for privacy and controlling data access is respected as every action that affects access rights is logged in the database.

In Section II, the main design decisions taken are presented leading towards the implemented WebMaDa 2.0 solution. Section III summarizes the new features of WebMaDa 2.0 highlighting the benefits and practical issues, as well as giving a hint to future improvements.

II. Design and Implementation

In order to handle any request received immediately an automated solution is required. This solution must support (1) user creation, (2) access request to foreign networks, and (3) password reset. Furthermore, for addressing privacy and controlling of the data (4) transparency must be assured by including a detailed logging system into the infrastructure.

Addressing the first three requests an automated mailing solution was integrated into WebMaDa 2.0. If a new user wants to use WebMaDa he needs to register by filling out the registration form. By submitting the form, the user creates an invitation request that is stored in the database. At the



(a) Requestor view (b) For for requesting access to data (c) Network owner's view of pending request

Figure 1: Graphical views during data access request

same time, the administrators receive a notification that a new request has been created. If the request is accepted by the administrators the user will receive an invitation which can be used to complete the registration. Otherwise, a message will be sent informing that the request has been rejected. After this, the user is able to create new networks or request access to foreign networks when not yet having permission as shown in Figure 1a. The latter is done by creating a permission request filling out a form (cf. Figure 1b). The form is received by the backend. Here a mapping between the selected network by the unique WSN Identifier (ID) and the stored owner address is performed resulting in mailing the request to him. The owner receives a mail with the request and a personalized link to handle it within WebMaDa (cf. Figure 1c). The owner can now grant the access, update the request or deny it. In return, the requester receives the result andl a log entry is created in the backend's database addressing the transparency request. Same procedure is followed if after time the network owner updates granted privileges. In case a registered WebMaDa user looses the password, a request can be placed via the corresponding form. The filled in data of the form is then compared to the logged entries in the database. If the check fails, no action is performed as not to provide a single bit of information wheter a user exists or not. Otherwise, the user receives a link to reset the password. In order to ensure transparency, the updating of a password also triggers the creation of a log entry.

III. Summary, Practical Issues, and Benefits

WebMaDa 2.0 supports the original functionality developed in 2014 and 2016. This is extended by an automated mailing solution to handle incoming requests immediately and, thus, reducing delays in the system each time an administrator interaction was required in earlier versions.
The designed and implemented solution is user-friendly due to its intuitive design in the graphical environment including easy understandable instruction to conclude the workflow (e.g., request data access, register new user). All steps are following a global process starting with a form that need to be filled out with respective information required, checkup with stored information if applicable, and updating database with new information (e.g., new user information, new networks, granted/updated/revoked privileges). In order to address the general privacy request of users the administrator is only involved when new users are registered or an existing WebMaDa user should become administrator of WebMaDa for the case the original administrator needs a representative. Addressing the transparency concerns of users, any changes are logged within the database with required information (e.g., timestamp, what was done and by whom). All this logging information can only be accessed by the network owner or the global administrator.
Looking from a practical perspective all user requests are addressed within WebMaDa without having drawbacks on performance of WebMaDa assuming several networks hosted at the same time. Due to the fact that WebMaDa 2.0 is still web-based, no new special requirements to the mobile device exist and the solution is still hardware and software independent.
Further developments are conceivable with regard to session timeout similar to banking systems, two-way authentication besides mailing using SMS, and further flexibility in visualizing data collected.

References

[1] D. Buenzli, "Efficient and User-friendly Handling of Access Requests in WebMaDa," Bachelor Thesis, Communication Systems Group, Department of Informatics, University of Zurich, Zurich, Switzerland, Jan. 2018.

[2] M. Keller, "Design and Implementation of a Mobile App to Access and Manage Wireless Sensor Networks," Master Thesis, Communication Systems Group, Department of Informatics, University of Zurich, Zurich, Switzerland, Nov. 2014.

[3] C. Schmitt, C. Anliker, and B. Stiller, "Pull Support for IoT Applications Using Mobile Access Framework WebMaDa," in *IEEE 3rd World Forum on Internet of Things (WF-IoT)*. New York, NY, USA: IEEE, Dec. 2016, pp. 377–382.