EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Toward network function virtualization for cognitive wireless mesh networks: a TCP case study

Wooseong Kim

Abstract

Nowadays, cognitive radio (CR) is considered the key technology to deploy wireless nodes that are able to adapt their transmission parameters based on the characteristics of the environment. In our previous works, we leveraged the reconfigurability properties offered by the CR technology to implement Cognitive Wireless Mesh Networks (CMNs) operating on the 2.4-GHz industrial, scientific, and medical (ISM) band. *Urban-X*, a new architecture for Multi-radio Cognitive Mesh Networks, self-configures and self-adapts to the dynamic interference conditions experienced on the Wi-Fi channels in dense urban environments. *Urban-X* estimates traffic load of existing users outside the mesh in each channel and utilizes less occupied channels opportunistically. In such dynamic environment, a transport layer protocol can suffer from variation in delay and available bandwidth due to spectrum sensing and mobility. Several TCP solutions for the CWMNs have been proposed but they could not be employed for implementation due to complexity and inefficiency. Instead, we adopt network function virtualization (NFV) technique which allows to instantiate virtualized network functions (VNFs) dynamically on demands. For this, we propose overall architecture of NFV framework for the CMNs and apply TCP accelerators (ACC) VNFs to improve TCP throughput as a case study. Simulation results show TCP-ACC VNF can achieve a notable performance enhancement compared to previous TCPs for the CMNs.

Keywords: Cognitive radio; Network function virtualization; TCP

1 Introduction

The proliferation of electronic devices equipped with Wi-Fi transceivers has favored the rapid deployment of large-scale wireless mesh networks (WMNs) based on Wi-Fi backbone in several highly dense urban areas of the world [1, 2], which is considered a cheaper alternative to the traditional cabling solutions for the Internet access and Internet of thing (IoT). However, the performance of WMNs is still not competitive since the mesh nodes in highly dense urban environments must share the 2.4-GHz industrial, scientific, and medical (ISM) channels with several external devices, primary nodes (PNs), that cannot be governed by the mesh nodes and possibly interfere with the WMNs transmissions. Cognitive radio (CR) researches have mainly investigated the potential of reconfigurable radio technology for enabling an opportunistic spectrum access paradigm [3, 4], in which a cognitive device exploits the portions of the spectrum left vacant and less interfered multi-hop routes by the licensed users. While this solution can effectively improve the spectrum utilization of wireless communication, it also poses formidable challenges for its implementation, e.g., the legacy aspects for the utilization of licensed spectrum and the protection of the licensed users.

We proposed the utilization of CR technology for the deployment of self-configuring and self-adapting WMNs in the 2.4-GHz ISM band, which is *Urban-X* [5], a cognitive mesh network (CMN) in highly dense urban area. Here, we underline that the utilization of CR technology on the 2.4-GHz ISM band does not pose the legacy issues of the conventional CR technology, since a mesh node is not expected to protect a PN and leave the frequency once a PN is detected. This greatly simplifies the hardware design of the cognitive mesh nodes as accurate PN detection is not required. Instead, simplified spectrum sensing

Correspondence: wooseong@gachon.ac.kr
Department of Computer Engineering, Gachon University, Seongnam street, Seongnam, Korea

can be used in order to infer the load on the different channels in the ISM band and utilize this information for, e.g., channel assignment.

Reconfigurability in a mesh node can be realized by software on top of the simple hardware. In our *Urban-X* testbed that uses general purpose processor (GPP)-based personal computers [6], most of the wireless communication functions including even physical modulation, a media access control (MAC) protocol, channel sensing, and switch for cognitive radio are implemented by software on top of Window operating systems. Owing to the increasing general computing power, thus mesh nodes can be easily reconfigured or updated over the air by remote software management technologies. For example, if a new MAC protocol needs to be replaced, the protocol software is downloaded over the air and installed in each mesh node.

Recent software-defined network (SDN) and network function virtualization (NFV) concepts help the above procedure more smoothly being done. NFV adopts cloud computing technologies for creating or renewing network functions easily on top of virtual machine. In the above example, a new software package including the new MAC protocol is downloaded and then a new computing instance is created to load the new package. After the new mesh node instance works successfully, old mesh node instance with an old MAC protocol is removed. This SDN and NFV change network architecture and operation mechanism in WMNs. Granelli et al. [7, 8] depicts an overall picture of SDN and NFV use for the WMNs. In short, the SDN can set up forward tables of multi-hop nodes along shortest path in the WMNs using switch or router control protocols, such as OpenFlow [9]. And NFV instantiates or removes virtualized network functions (VNFs) dynamically using cloud computing technologies such as virtual machine, cloud protocol stack, etc. For example, [10] the proposed “CloudMAC” that virtualizes a MAC protocol of the WMNs and controls a path from the software MAC to a physical radio interface dynamically to reduce latency and save power consumption using the OpenFlow.

In this paper, we propose a new NFV and SDN architecture for a cognitive wireless mesh network, the *Urban-X*, in which various VNFs can be realized on top of the reconfigurable network devices for the cognitive radio such as firewall, packet filter, content cache, etc. And we implement a TCP accelerator VNF that accelerates TCP throughput by splitting a TCP connection for a case study of NFV and SDN in the CMNs, because TCP performance in the *Urban-X* is very limited even if it is one of most important protocols carrying popular application protocols like HTTP. From simulation, we confirm that our NFV-based approach improves TCP performance in the CMNs.

The rest of the paper is organized as follows. Section 2 introduces concept of the *Urban-X* and architecture of a cognitive mesh node, and Section 3 shows reconfigurable hardware platforms for the *Urban-X*. Section 4 describes SDN and NFV architecture for the *Urban-X*, and Section 5 presents a TCP case study of SDN and NFV techniques in the *Urban-X*. Related works and conclusions follow in Sections 6 and 7.

2 The *Urban-X* architecture

CR constitutes one of the most investigated solutions to deploy flexible and high capacity wireless networks. A CR device is defined as a wireless node that is able to adapt its transmission parameters based on the characteristics of the environment and on the requests of the applications [3].

In our previous works, we proposed the utilization of CR technology in the ISM bands for WMNs, *Urban-X*, and CMNs [5]. In the following, we discuss drawbacks and benefits of this approach, and we review details of the *Urban-X*.

Benefits While several recent studies demonstrate the presence of unused portions of the spectrum in the licensed bands, the practical utilization of these frequencies by CR devices poses several challenges in terms of spectrum access regulations and of CR devices and network deployment. At present, the existing IEEE standards (e.g., IEEE 802.22 wireless regional area networks (WRANs)) enable the opportunistic usage on a limited set of frequencies and introduce strict constraints on the protection of the licensed users. On the opposite, cognitive WMNs on the license-exempt bands do not need to implement specific spectrum protection policies. This is because interference is common in those frequencies and mesh routers (MRs) and PNs can coexist on the same channels. Therefore, protection mechanisms need not be implemented, and a MR is not forced to vacate a channel already occupied by a PN. Also, as we will see later, the standard Wi-Fi MAC layer does not necessarily need to be modified. This greatly simplifies the hardware design of the cognitive mesh node and also facilitates the WMN deployment over realistic scenarios.

Drawbacks The 2.4-GHz ISM channels might experience high congestion in dense urban areas, due to the number of transceivers and networks operating on the Wi-Fi channels. As a result, CR technology might not provide an effective network capacity increase as when utilizing the licensed band. However, the spectrum reconfigurability can still improve the performance of WMNs, by allowing the WMN to adapt its operations to the current PN activities and, thus, mitigating the mutual interference between PNs and MRs.

Urban-X is cross-layer protocol framework for cognitive multi-radio mesh networks. The main goal of the *Urban-X* is to adapt the WMN operations to the current channel utilization conditions and to favor the co-existence of different networks operating over the shared ISM bands. To this aim, *Urban-X* borrows the principles of "spectrum reconfigurability" and "spectrum sensing" from classical CR devices, in order to estimate the amount of external interference on each Wi-Fi channel and thus, the capacity left for the mesh nodes. Based on this information, *Urban-X* adapts the network operations through three layers of self-configuration: (1) frequency adaptation (i.e., dynamically decide the channel to be used on each MR in a distributed way), (2) path adaptation (i.e., dynamically re-configure the routing process based on the actual network conditions), and (3) load adaptation (i.e., dynamically decide the channel/flow to be served at each MR).

Figure 1 shows a practical deployment of the *Urban-X* network over an urban scenario. *Urban-X* is composed of three types of nodes: the (1) Internet mesh gateways (IMGs), the (2) mesh clients (MCs) that correspond to the user devices, and the (3) MRs that form a static backbone between the MCs and the IMGs and forward the MCs' traffic toward the closest IMG or the MG where the destination MC is attached to.

The MRs are wireless access point (APs) with routing and cognitive capabilities equipped with three standard Wi-Fi radio interfaces (i.e., $R1$, $R2$, and $R3$). Each radio interface is tuned to one of the Wi-Fi channels in the ISM band. The $R1$ and $R2$ interfaces are used to receive or transmit data packets produced by applications, while the $R3$ interface is tuned to a common control channel (CCH) for exchanging control messages required for network management, routing, and channel (re-)configuration. More specifically, the $R1$ interface is used to receive ingoing traffic from other mesh nodes and is tuned to a semi-dynamic channel that is decided based on the PN activity and on the mesh internal network traffic load. The $R2$ interface is used to transmit ongoing traffic toward other mesh nodes and thus, dynamically switches among the available channels with a predefined

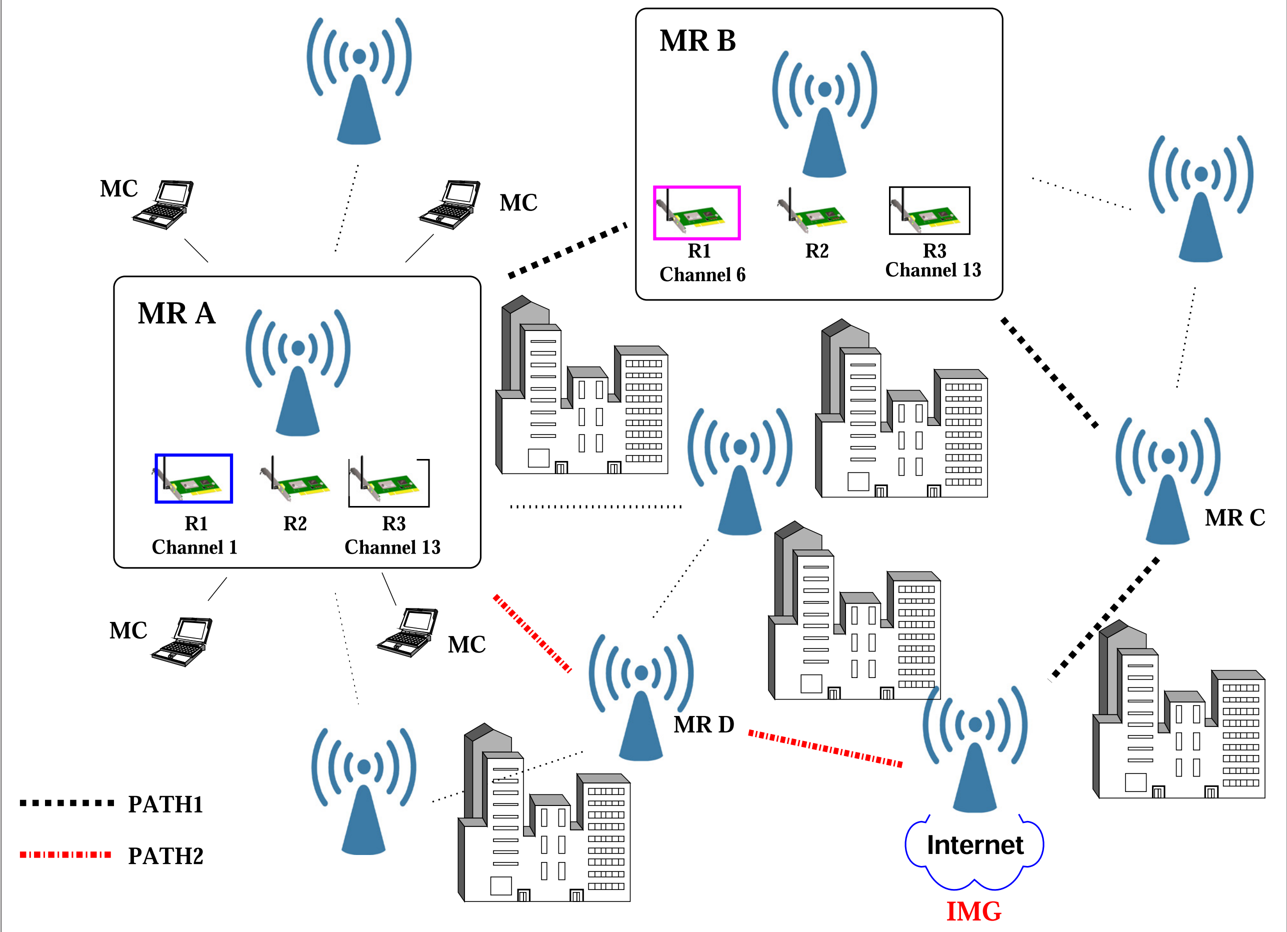


Fig. 1 The *Urban-X* network deployment

switching interval (e.g., 40 ms), which can adapt based on the traffic load for forwarding. The wireless communication among MRs is implemented as follows. When a MR, e.g., MR A of Fig. 1, needs to forward data to another MR, e.g., MR B of Fig. 1, then A must tune its $R2$ interface to the channel used by the $R1$ interface of node B (that is channel 6 considering the example of Fig. 1). Through the utilization of two radios for sending and receiving, *Urban-X* can preserve the network connectivity, while expanding the network capacity by allocating interfering MRs on different channels.

3 *Urban-X* implementation

Cognitive MR architecture in the *Urban-X* is depicted in Fig. 2. Through the *spectrum sensing block*, each MR gathers information about the PNs activity and estimates the capacity left for the WMNs operations on each Wi-Fi channel. The channel information is continuously updated and shared among the layers of the protocol stack through the implementation of a cross-layer *repository*. Based on the amount of PNs interference on each channel and on the traffic requests from the nodes of the WMNs, each MR dynamically decides the channel to be used on its fixed interface $R2$, through the *channel decision block*. The *mesh forwarding block* is responsible for building and updating the routing tables of the MRs. In *Urban-X*, the routing algorithm implements a multi-path approach, i.e., multiple (possibly overlapping) paths can be discovered and kept for each pair of source-destination traffic flows, as shown in Fig. 1. When a MR must forward data toward a destination d (where d can be a MC or the closest IMG), the *path decision block* dynamically selects the optimal next-hop node toward d, based on the performance of each path. We highlight that the performance of each link/channel might dynamically change as a consequence of a variation of PNs activity and/or of the internal WMN traffic. These changes are handled by the *Urban-X* architecture through a configuration adaptation performed by the *channel decision block* and/or from the *path decision block*. Moreover, since each MR uses a single transceiver to transmit on multiple channels (i.e., $R2$), *Urban-X* includes a *channel scheduler block* to determine at any time the channel served by the switchable interface $R2$. The *channel scheduler block* is between the MAC and routing layer and maintains a separate queue for each Wi-Fi channel. Once the next-hop node for d is decided (e.g., MR B), the data packets are inserted into the queue corresponding to the channel used by the receiving interface $R1$ of the MR B (e.g., channel s). The message will be transmitted once the interface $R2$ of node A is tuned

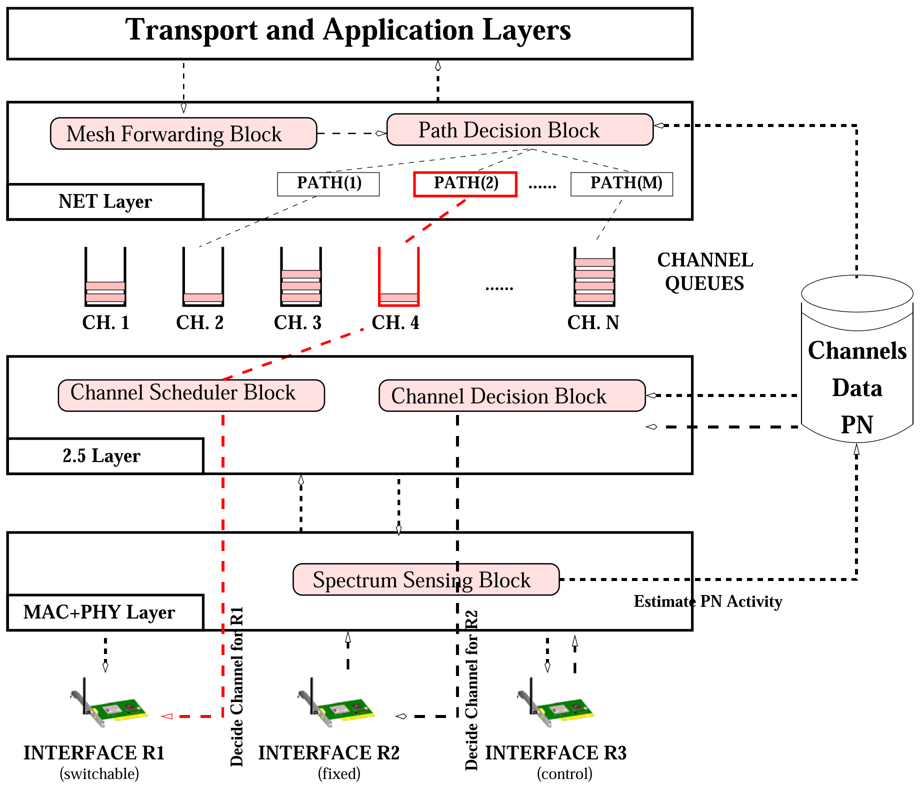


Fig. 2 The *Urban-X* protocol stack

to channel s. The *channel scheduler block* implements the channel scheduler algorithm on the $R2$ interface, based on the traffic load of each channel. In the following, we briefly present the main characteristics of the Block components of the *Urban-X* architecture. Technical details are out of the scope of this paper but can be found in [5] and [11].

Spectrum sensing block In *Urban-X*, each MR node senses the spectrum periodically through an energy-detector scheme and estimates the workloads of PN traffics (ω) on each channel. Let $\omega(s,i)$ be the workload of channel s, estimated by MR i. In our experiments, we compute $w(s,i)$ as:

$$\omega(s,i) = \frac{\text{Number of busy samples on}\, s}{\text{Total number of samples monitored on}\, s} \tag{1}$$

where a sensing sample is classified as "busy" if the average power perceived on s is higher than an energy threshold for channel clear assessment (CCA) of Wi-Fi. Since the spectrum conditions might dynamically change over time, each MR performs periodic spectrum sensing on the current channels used by the $R1$ and $R2$ interfaces with a fixed frequency and for a duration of T_s sensing time intervals. Moreover, cooperative techniques are introduced to reduce the overhead of spectrum scanning, i.e., the MRs share the workload information by broadcasting these values on the control interface $R3$.

Channel decision block Based on the sensing information, each MR node decides the channel for its receiving interface $R1$, by choosing among the available channels of the ISM band. Currently, most of the commercial APs are manually configured and are usually set to one of the three orthogonal channels provided by the IEEE 802.11 technology. Some APs implement automatic channel selection schemes that choose the Wi-Fi channel providing the highest signal-to-noise ratio (SNR). However, both these channel selection techniques do not take into account the effective amount of utilization of each channel. In our approach, we attempt to allocate each MR to the channel that provides the highest capacity needed to serve the current traffic request of the node. To this aim, we compute the remaining capacity (RC) of each channel, defined as the portion of bandwidth left from the PNs for mesh usage. Each MR i computes the RC of each channel s as follows:

$$\mathrm{RC}(s,i) = (1 - w(s,i)) \cdot C \tag{2}$$

where C is the maximum capacity of a channel that can be offered by the version of the IEEE 802.11 standard in use. Based on the values of the $\mathrm{RC}(s,i)$, the *channel decision block* implements a distributed algorithm to select a channel for the receiving interface by taking into account the available capacity based on mesh external traffic (RC) and factoring in the amount of traffic required to forward mesh internal traffic toward the destination MRs or the IMGs. Once the channel is selected, a control message is sent on $R3$ to inform neighboring MRs which then update their repository information accordingly [5].

Mesh forwarding block This block is responsible for establishing the path between the current MR (i.e., MR i) and any possible node of the *Urban-X* network. The routing algorithm implements an enhanced version of the popular AODV protocol, with multi-path capabilities. During network setup, each MR creates an adaptive forwarding mesh structure, with multiple next-hop candidates toward the target IMG/MR. The different paths can have links or nodes in common and can be used sequentially or concurrently from the MR based on the path and link conditions, through the path scheduler scheme described below.

Path decision block In case a MR i has to forward a message toward a destination d and has multiple routing entries, the *path decision block* decides the next-hop node to be used to transmit a message. To this aim, *Urban-X* utilizes a novel routing metric that estimates the quality of each path, considering both the path length and its reliability in terms of PN interference [11]. Once the next-hop node is determined (i.e., MR j), the message is inserted into the queue corresponding to the fixed channel used by the $R2$ interface of MR j and transmitted according to the scheduler implemented by the channel scheduler block.

Channel scheduler block Instead of using a single queue for ongoing packets, in *Urban-X*, each MR keeps a separate packet queue for each Wi-Fi channel. The queue k contains the packets to be transmitted on channel k using the switchable interface $R1$. Every T_{switch} second, the *channel scheduler block* decides the channel queue to be served and tunes the interface $R1$ accordingly. Once the channel is tuned in by the switching radio, all packets are sent on the channel using the standard 802.11 MAC layer operations. This greatly reduces complexity and allows to implement *Urban-X* on off-the-shelf hardware.

Several hardware platforms have facilitated the implementation of CR technology, which allows upper layers like applications to modify or update easily features for network or link layer protocols and even physical layer functions. From this, channel sensing and switching period and channel selection algorithm of the *Urban-X* can be dynamically changed according to user traffic or external PN interference. In what follows, we introduce several candidate platforms for the *Urban-X*. Among them, Microsoft Software Radio Academic Kit (SORA) is used for implementing the MR of the *Urban-X*.

The Universal Software Radio Peripheral (USRP) board is a commercial hardware platform which is highly reconfigurable and frequently used to implement various aspects of CR architectures such as spectrum sensing and co-existence [12] or integrated MAC/physical layer (PHY) processing [13]. USRP provides the motherboard which hosts an field-programmable gate array (FPGA), analog/digital conversion, and the interface to the host. Specific radio front-ends can be used to implement a software-based radio solution for different frequency bands. The signal processing is done on the host computer using the open-source GNURadio software, which supports both C++ programming for real-time critical tasks and Python for high-level implementation. However, supported RF bandwidth is quite limited and the performance may be suboptimal due to its internal block structure leading to delays in the processing.

Wireless Open-Access Research Platform (WARP) [14] is a scalable and programmable wireless platform composed of custom hardware, platform support packages, open-access repository, and several research applications. An FPGA allows to implement complex PHY processing tasks such as MIMO. In addition, a PowerPC processor supports C-based applications providing an interface to the FPGA running the PHY layer code. Interestingly, tools are available which expose the WARP hardware to MATLAB thus integrating signal processing code implemented in a high-level language with the hardware in a fast way. Nevertheless, the FPGA programming for the PHY layer makes it quite difficult to implement.

SORA [15] is a radio platform which is fully programmable on commodity PC architectures. Therefore, wireless protocol stacks such as IEEE 802.11a/b/g can be implemented (including baseband processing) on standard off-the-shelf PCs. One or more dedicated processor cores can be reserved for signal and protocol processing. SORA uses a radio control board in order to transfer high-fidelity digital waveform samples from the hardware RF front-end board to PC memory via Peripheral Component Interconnect Express (PCIe) bus for processing supporting up to 16.7 Gbps ($\times 8$ mode) throughput at below microsecond latency. In the host PC, the software architecture uses extensive lookup operations together with single-instruction multiple-data processor extensions to speed up waveform processing, which even can be implemented over multiple cores. In [6], we implemented a new MAC protocol for the *Urban-X* MR using the SORA PCIe boards and Dell PCs, in which we defined several additional MAC states for performing spectrum sensing to estimate channel workload and selecting a least occupied channel and configuring it for the *R*2.

OPENAIRINTERFACE.ORG [16] is an open-source hardware and software initiative for experimental radio experimentation. It features software written in C-code for physical and link layer functionalities that can be used to implement cellular and mesh network topologies. The software runs under a real-time application interface (RTAI) as an extension of the Linux operating system. In addition, a hardware platform is available based on FPGA, which can use the same baseband processing resources for multiple radio standards.

Coral [17] provides a framework for MAC and PHY layer control using a standard multi-radio Wi-Fi router. Due to its modular design, it can be used to implement typical CR features such as channel assignment, spectrum sensing, or controlling radio access in time space and frequency. However, fine grained co-existence mechanisms required by typical CR approaches such as vacating spectrum immediately once PNs are detected might be difficult to implement due to the limited capabilities of the hardware platform.

4 Network function virtualization in *Urban-X*

NFV technique enables conventional hardware-based network equipment to be implemented as software on top of bare metal servers, e.g., server farm, which can reduce cost of network deployment and management effort. In addition, the network function can be dynamically created or removed just in time when triggered by applications based on policies or rules, which saves time for network deployment compared to the hardware equipment and also computing resources. This easy-to-deploy virtualized network equipment helps establishing various service chains flexibly.

Recently, several standard organizations such as ITU, European Telecommunications Standards Institute (ETSI) NFV Industry Specification Group (ISG) [18], and Open Networking Foundation (ONF) [19] are developing NFV concept and its system framework. Open Platform for NFV Project (OPNFV) [20] releases an open-source platform following such standard like ETSI NFV architecture [21]. The NFV architecture mainly consists of an operation system layer like operations support system (OSS), VNF, virtual machine (VM), and a management layer as shown in Fig. 3.

Once any network function is demanded from the OSS layer, the corresponding VNF is instantly created on top of a VM-based compute node like conventional cloud computing technique. Then, VNF manager (VNFM) manages life cycle (i.e., create or delete VNF instances and scale up/down or in/out computing resources) of the VNFs cooperating with virtualized infrastructure manager (VIM) that allocates hardware resources (i.e., computing, storage, and networking resources) for each VNF. In contrast to conventional cloud computing, network virtualization is an integrated service that links flexibly multiple VNFs for a specific service. This service chaining

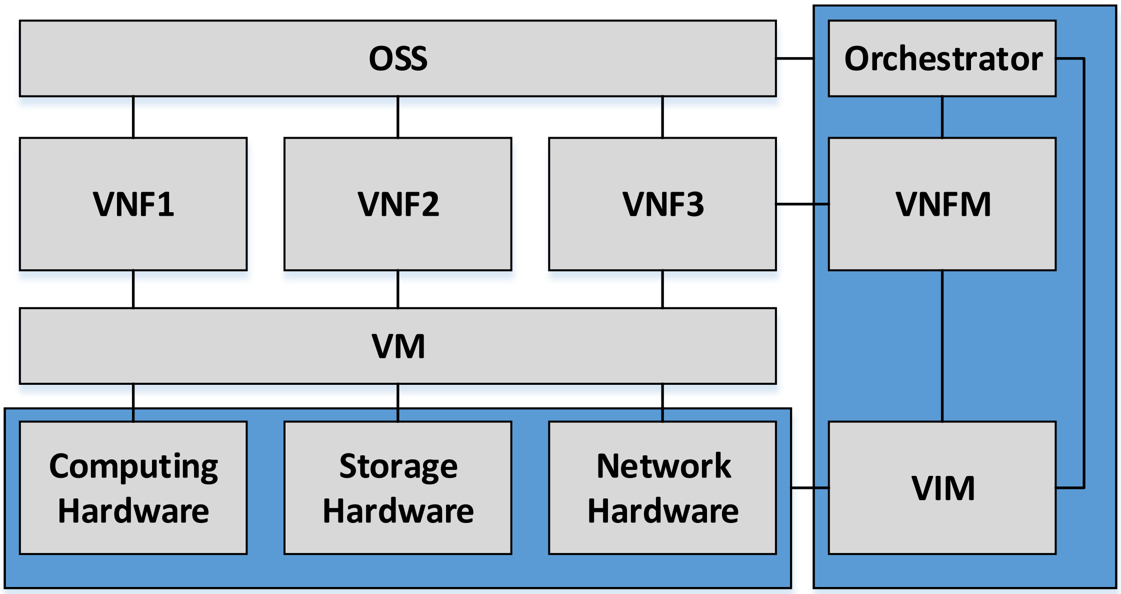


Fig. 3 System architecture of network function virtualization

is orchestrated by the management layer, in which overlay network for the VNFs is established using network control protocols like OpenFlow [9].

Figure 4 shows a new *Urban-X* architecture combined with SDN and NFV technologies. The VIM has a network controller to establish a multi-hop route for end-to-end communication. Here, the network controller uses OpenFlow (OF) that is one of the most popular control protocols, which manipulates a forward or a routing table of each mesh node along the route. In the figure, supposed that a MC near the node *A* sends packets to another MC near the node *F*, the path can be set as the bold line: *A - C - D - G*. For this, an OpenFlow client at each mesh node reports link status information that is actually a RC value of Eq. 2 to the OpenFlow controller periodically. Based on this information, the controller calculates cumulated expected transmission time (CETT) of possible routes for the flow and then creates or updates forwarding tables (FT) of the mesh nodes along a minimum CETT route. An incoming interface to the node *A* from the MC is mapped with an interface to the node *C* for the flow identification (ID) (e.g., addresses, port number, etc) in the FT. Then, all incoming packets with the flow ID are forwarded automatically to the interface toward the node *C*. The control packets for mesh node configuration between VIM and mesh nodes like OpenFlow messages are exchanged through the CCH of *R3* in Fig. 2. The VIM/OF controller is basically a logical entity that can be included in one of the mesh nodes in the figure. In case the VIM/OF cannot reach directly each mesh node, it can send or receive the control messages by multi-hop routing.

According to the service chain, various VNFs can be created in the mesh nodes such as deep packet inspection (DPI), firewall, authentication server, QoS-based packet queueing, etc. Among them, TCP accelerator (ACC) is a very attractive VNF for the *Urban-X* because there is a limitation to provide good TCP throughput to the MC due to complexity of CMNs. Detailed reason will be explored in Section 5. Also, most of the recent applications including even video streaming use TCP connections. Accelerating TCP throughput using NFV techniques is therefore applied to our *Urban-X* architecture as can be seen in Fig. 4. VIM instantiates VNFs of TCP-ACC at each mesh node once the OpenFlow controller decides a TCP flow path within the *Urban-X*. In detail, VIM informs the OpenFlow controller to send control messages to the OF client in order to instantiate the TCP-ACC on each mesh node when the path information is given to the VIM. As a result, the node *A*, *C*, *D* and *G* create TCP-ACCs over the VMs and snoop the TCP flow with a flow ID to accelerate.

This TCP-ACC or proxy as the VNF is interesting also to researchers in wired or wireless cellular communication areas. Thus, our CMN model with TCP-ACC can be considered for WiMAX or LTE mesh networks and IEEE 802.22-based mesh networks even though evaluation is performed using Wi-Fi connectivity between the mesh nodes.

Our TCP acceleration approach breaks TCP end-to-end design philosophy and creates multiple short TCP

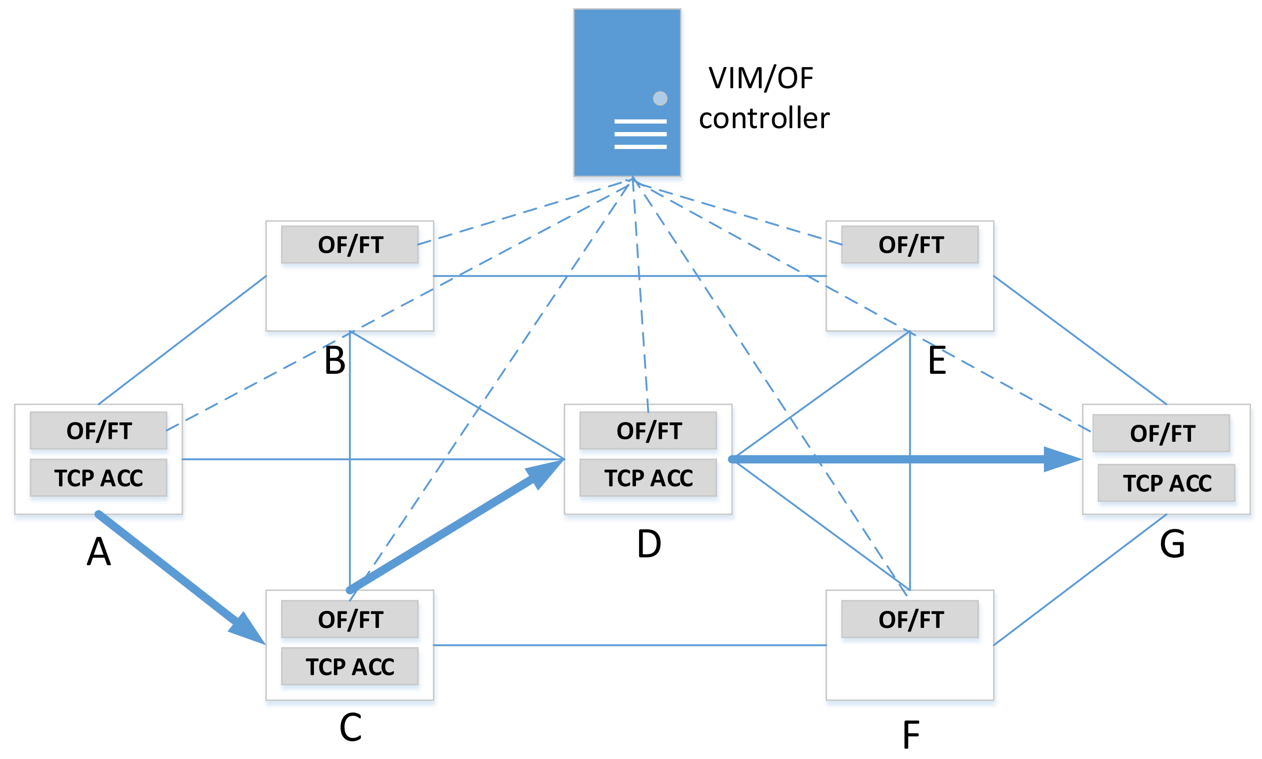


Fig. 4 Example of NFV in *Urban-X*

connections as many as number of hops in the *Urban-X*. Split TCP connections by the TCP-ACCs can lead to higher overall throughput by reducing the round-trip time. In addition, the TCP-ACC can relieve the rest of the path from having to carry end-to-end retransmissions as well. Previously, multiple TCP proxies along the route for the wireless ad hoc networks were considered in order to improve throughput in [22–24]. The TCP split scheme was originally designed to separate the different mediums of wire and wireless in wireless cellular networks. For the multi-hop ad hoc networks (MANETs), Kopparty et al. [22] divided an original long route into a sequence of short TCP connections by locating TCP proxies every three nodes along the original route to improve the throughput of end-to-end TCP connection. Ouyang et al. [25] proposed more efficient TCP proxy placement based on link status information since processing packets at the higher layer by proxies causes overhead. For this, cross-layer approach between network and transport layers is considered to find optimal nodes to place the TCP proxy. In detail, source routing protocols collect queued packets and retransmission counts of links along the route and select TCP proxy nodes. Then, a sender piggybacks a TCP proxy node list on TCP SYN, and each proxy node responses SYN-ACK to establish segmented TCP connection.

5 A case study: TCP in *Urban-X*

In the *Urban-X*, TCP performance may be adversely affected due to not only multi-hop transmission in ad hoc networks but also channel sensing and channel mobility for cognitive radio. Several studies have been undertaken to optimize TCP for the WMNs in order to enhance network performance solving problems caused by node mobility and route breaks. For example, explicit notification messages provided by the network layer [26, 27] help distinguishing a temporary disconnection due to deep fading or mobility. However, such approaches do not consider the unique properties of cognitive radio networks. For instance, periodic channel sensing and spectrum mobility and transmission channel switching can cause significant additional delay and jitter. Round-trip time (RTT) variation because of them degrades eventually TCP performance by distorting adversely bandwidth estimation and retransmission timeout (RTO), which requires a rethinking of transport layer protocol design.

In [28, 29], we evaluated performance of various TCP versions such as TCP Reno, New Reno [30], Vegas [31], and TCP-Sack [32] in the *Urban-X* with varying PN workload, sensing duration and heterogeneity of channel bandwidth and channel switching. According to our previous works [28, 29], short sensing duration does not always lead to improve TCP performance. Shorter sensing time can lead to ineffective estimation of external traffic and increased interference and packet loss. Not only the packet loss due to fluctuation in wireless channel quality and external PN interference but varying delay from channel sensing and switching affects TCP performance in the *Urban-X*, since the end nodes at the transport layer has limited information about reasons for the loss or delay. Here, we discuss details of three major challenges for the TCP operations in the *Urban-X*, especially for the congestion control mechanism which may considerably reduce achievable throughput.

- Unpredictable external interference from PNs: Due to co-existing PNs creating interference, CMN transmissions may face heavy packet loss. Such packet loss can lead to increased RTT due to MAC layer retransmits. This may result in low TCP throughput due to frequently triggered slow start. The channel assignment in *Urban-X* has been designed exactly to minimize the probability of external interference by selecting the channel least impacted by PN traffic.
- Spectrum sensing: For the channel assignment, CMNs estimate PN workload based on spectrum sensing. However, during such, cooperative sensing nodes are not able to transmit, potentially leading to TCP RTO. There is a trade-off between estimation accuracy and sensing overhead. To reduce the total sensing time, CMNs can exchange sensed channel workload information. Such collaboration among CMNs can decrease the required sensing window, which eventually may improve TCP throughput.
- Channel mobility/switching: In *Urban-X*, nodes adjust the receive channel *R1* dynamically and inform neighbors with Hello messages transmitted on the CCH. During this switch over, CMN links become disconnected, leading to potential packet loss. Likewise, nodes switch the transmitting radio channel (*R2*) to send packets to different neighbors. Once the *R2* radio switches to the next channel, it stays there for a predefined switching interval. Packets arriving to the receiving interface are queued at the correct channel queue for the next hop. If a node serves many neighbors, there may be a potential large switching delay until a given specific channel is served again. Such large delay can be detrimental to TCP performance as the number of intermediate forwarding nodes in the path increases. This is because channel switching jitter distorts RTT estimates and negatively affects RTO and throughput.

5.1 TCP performance in *Urban-X*

In the following, we evaluate the performance of TCP Reno over the multi-hop *Urban-X* that is extended from ns-2 simulator for the cognitive radio-based ad hoc networks as can be seen in Fig. 2.

Every simulation was repeated 50 times, and a data point denotes the average of those simulation runs. We send a single TCP flow over the chain topology (Fig. 5(a)) from node 0 to node 4. In the Fig. 5(a), 11 PNs are present on all 11 available channels at the same position with the same traffic load but different traffic pattern. Maximum data rate of each channel is 2 Mbps. Unless otherwise stated, the following parameters are used: a node periodically senses the spectrum for 70 ms, followed by 1-s transmission time. During the transmission time, the node switches to a channel with non-empty queue and stays there for a switching interval of 50 ms. The channel switching overhead, i.e., the time for the hardware to reconfigure the card once the channel switch is triggered, is assumed to be 1 ms.

First, we evaluate throughput of the TCP Reno in the *Urban-X* under different PN traffic patterns by varying the idle and busy duration. As shown in Fig. 6a, TCP throughput is highest when PNs have long idle and short busy duration. This allows TCP to effectively ramp up its congestion window. Rapid ON/OFF patterns of PNs lead to low TCP throughput as TCP cannot increase its window size fast enough. At fixed idle duration (e.g., 0.5 s), throughput increases slightly when busy duration decreases because the smaller the busy duration, the more time for TCP to transmit.

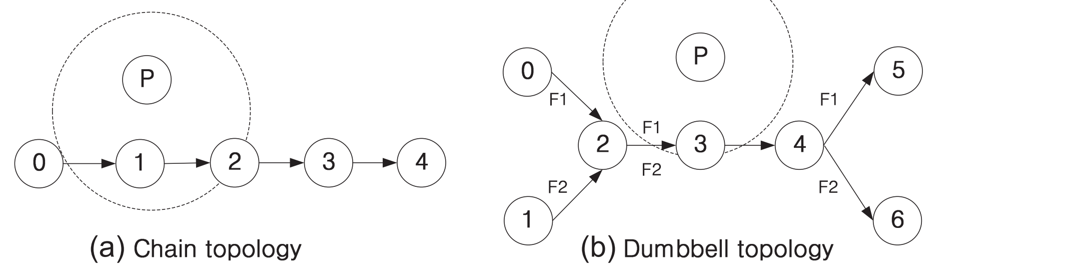
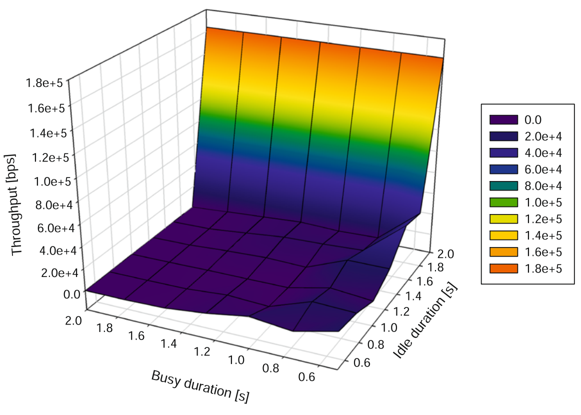
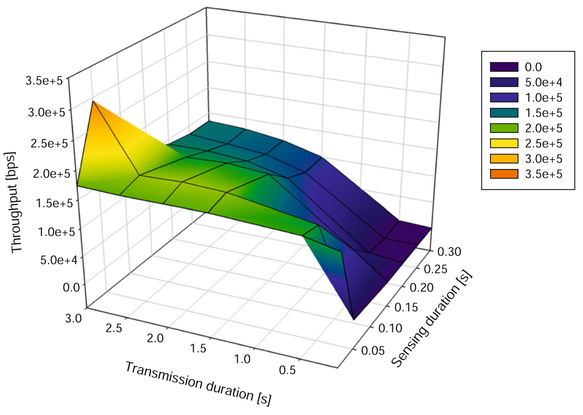


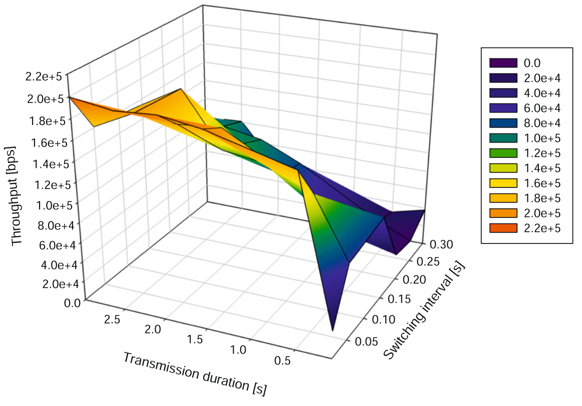
Fig. 5 The *Urban-X* evaluation topology



(a) Varying PN pattern



(b) Varying sensing pattern



(c) Varying switching pattern

Fig. 6 TCP Reno throughput

As a consequence, TCP throughput decreases drastically as the workload of PN traffic increases as the available capacity for the mesh decreases. Interestingly, the TCP throughput decreases more than the reduction of available channel capacity RC due to PN traffic load. This is the additional negative effect on TCP of packet loss caused by PN interference.

Figure 6b shows the impact of the sensing pattern under fixed PN traffic pattern ($T_{\text{idle}} = 2$ s; $T_{\text{busy}} = 0.5$ s) on achievable TCP throughput. Sensing is conducted after transmission duration finishes. TCP throughput increases obviously with transmission duration. The higher this value, the less overhead due to sensing and the longer time TCP has to increase its congestion window. Throughput with varying sensing period is almost comparable except cases of long transmission duration. When the sensing duration is larger than 200 ms, however, throughput again is penalized as packets need to wait too long in the buffers.

Switching interval is an important parameter to consider as it directly impacts the delay jitter and the achievable TCP throughput. Once the transmitting interface *R2* switches to a given channel with non-empty queue, it stays on that channel for a predefined switching interval (awaiting for more packets to arrive to the queue) in order to minimize switching overhead. A small switching interval leads to smaller jitter and is sensitive to channel mobility of neighbor nodes, but less time on a given channel.

In Fig. 6c, as the switching interval decreases, TCP throughput generally increases unless a very small switching interval and transmission duration are used, which reduces throughput due to overhead. Also, longer transmission duration increases TCP throughput. However, there is a constraint on the maximum transmission duration to measure timely channel occupancy from primary users.

TCP performance comparison regarding the external PN interference is shown in Fig. 7. Figure 7a shows the throughput achievable for TCP Reno, NewReno, Vegas, and SACK. Throughput of TCP is significantly higher for *Urban-X* (i.e., CMNs) compared to WMNs that are unaware of interference from PNs across all TCP versions, because the *Urban-X* selects the least interfered channel having highest available capacity and least probability for collisions due to PN traffic.

In Fig. 7(d), the average CWND size also shows that the sending data rate for TCP in *Urban-X* is higher than for standard WMNs. As a conclusion, the spectrum aware channel assignment helps effectively to reduce external interference due to PN traffic, which in turns leads to lower packet loss rate and significantly higher TCP throughput. The additional price to pay for the spectrum sensing is well utilized in higher throughput.

5.2 TCP accelerators on cognitive mesh nodes

In previous simulation, the *Urban-X* shows that cognitive approach in the ISM bands helps increasing TCP throughput by reducing packet loss due to external PN interference. Additional simulation results in Table 1 show that TCP performance in the *Urban-X* is still affected by transmission durations and channel switching interval seriously because both delay accumulates hop by hop

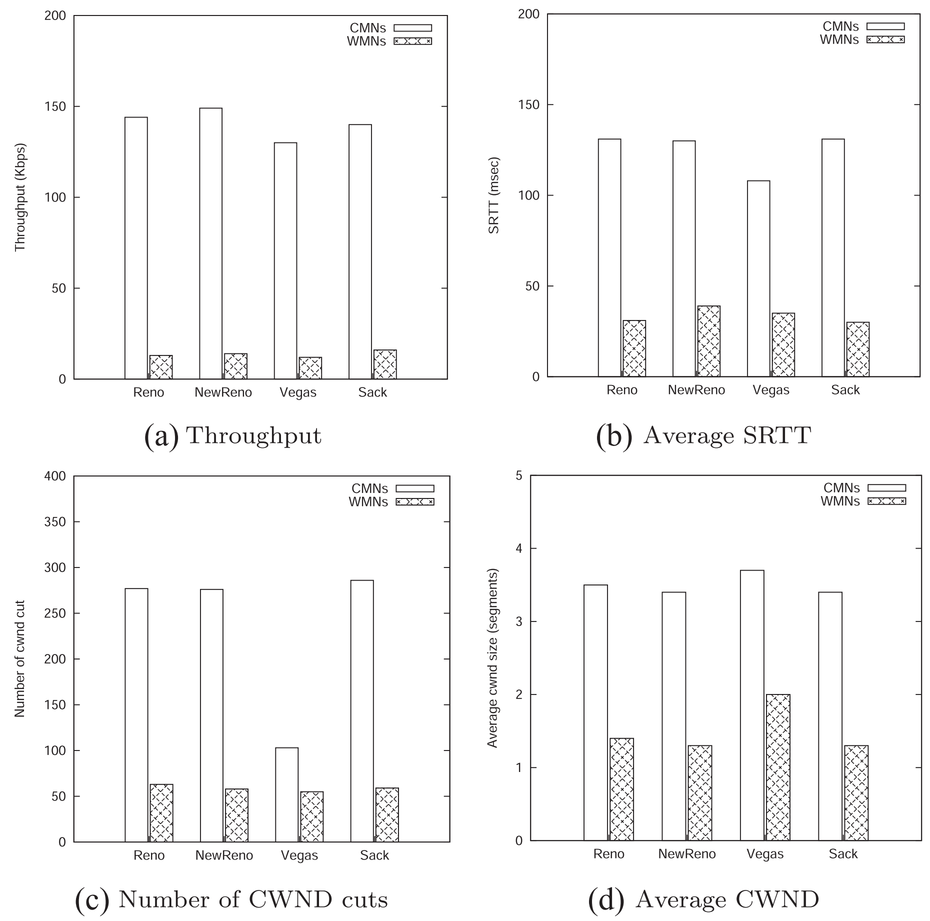
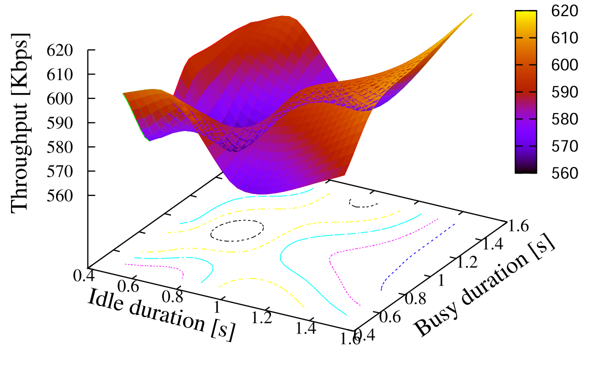


Fig. 7 TCP performance comparison with or without cognitive radio

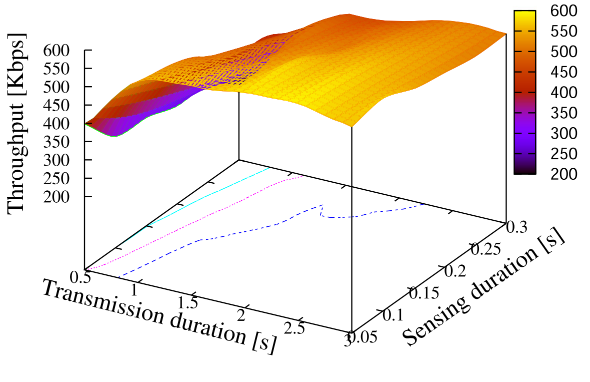
leading to significant end-to-end delay variation. The TCP throughput is measured for different path lengths (fixing the sender at node 0 and varying the receiver from nodes 1 to 4 and configuring the switching interval as 40 ms, spectrum sensing duration as 10 ms, transmission duration as 1 s). When path length increases to four hops, throughput decreases much more severely and the packet drop rate increases to around 5 %. We repeated the same simulation using User Datagram Protocol (UDP). The UDP throughput was 1.3 Mbps for a four-hop path even although the delivery ratio was low (around 60 %). This is because UDP's traffic pattern is unidirectional. In contrast, TCP data packets are acknowledged leading to a two-way traffic pattern. Accordingly, the chain topology leads to frequent channel switches for the intermediate nodes in Fig. 5a as they need to switch between transmissions toward their upstream and downstream neighbor nodes to send TCP data and TCP-ACK. Such channel switch delay leads to large end-to-end delay variations affecting RTO and leading to frequent TCP timeouts. As the number of hops increases, channel switching mainly limits TCP throughput. As a conclusion, TCP should be re-built to cope with those effects in the *Urban-X*.

We demonstrate the TCP Accelerator (TCP-ACC) in the *Urban-X* under same simulation condition as previous one in Fig. 8. Figure 8a depicts TCP throughput under varying PN traffic with different idle and short busy duration, which forms an irregular concave shape. This graph does not show clear relationship between TCP throughput and PN traffic patterns compared to the result in Fig. 6a, but throughput is relatively higher in short busy and long idle duration. In contrast to the previous TCP Reno case, packet loss affection that happened near node 1 in Fig. 5 is not propagated to the remaining TCP path in the TCP-ACC case. To say, such partial PN interference in the sub-TCP path does not direct overall throughput toward a consistent way.

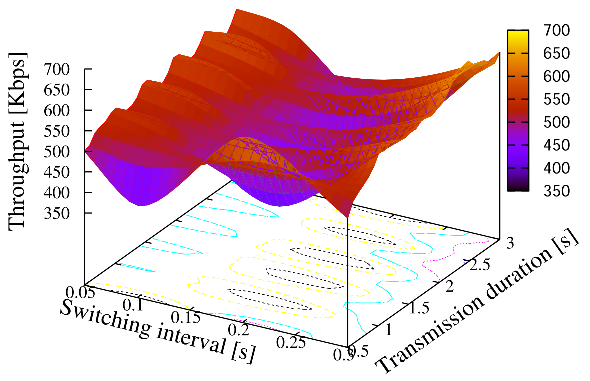
Figure 8b shows TCP throughput with a different sensing pattern; PN traffic pattern is configured as T_{idle} = 2 s and T_{busy} = 2 s). TCP throughput increases monotonically as the transmission duration increases similar with



(a) Varying PN pattern



(b) Varying sensing pattern



(c) Varying switching pattern

Fig. 8 TCP throughput with NFV-based accelerators

the TCP Reno. However, TCP throughput is almost comparable with varying sensing duration in TCP-ACC while throughput gets improved in case the transmission duration is long in the TCP Reno. From this, we can confirm again the TCP throughput is not much affected by subset of TCP path that is interfered by PN traffic. Packet loss from the PN traffic can be recovered fast owing to the short TCP path. However, the overhead of sensing duration degrades TCP throughput slightly in any transmission duration.

In the TCP-ACC, throughput is achieved high either in the most frequent or least frequent switching case as shown in Fig. 8c. In the TCP Reno, less frequent switching leads to decrease throughput since a node cannot switch fast transmission channels between a forward and a reverse node for data or acknowledgement which

Table 1 TCP Reno throughput with number of hops

Number of hops	1	2	3	4
Throughput (Kbps)	1413	886	291	141
Packet drop rate	0	0	0.05	0.05

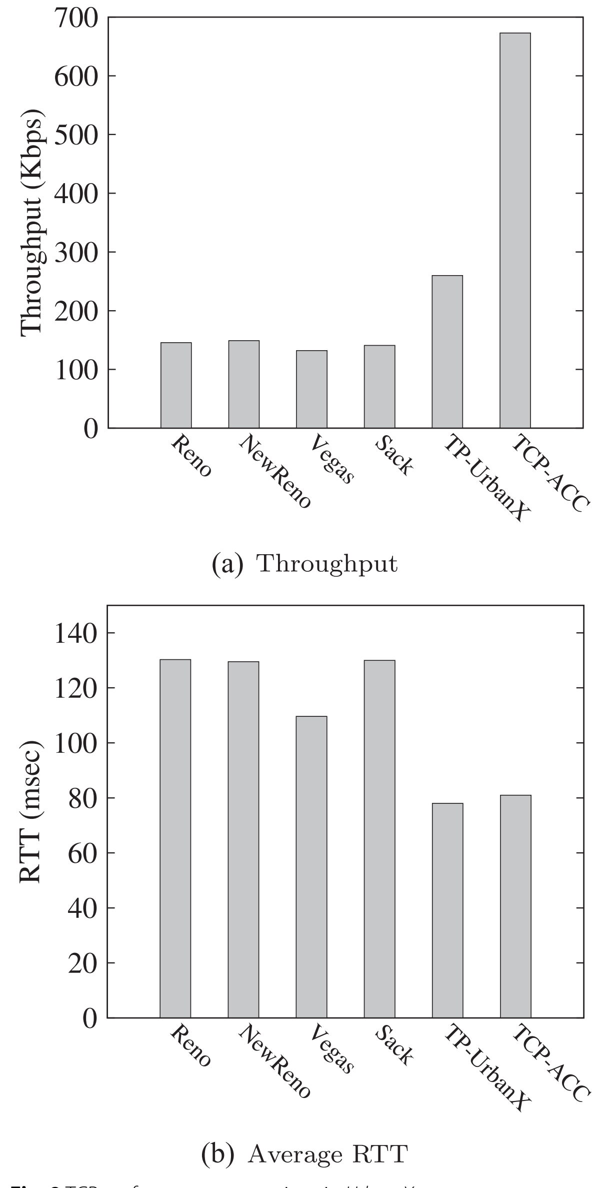


Fig. 9 TCP performance comparison in *Urban-X*

causes long RTT or RTO. However, the TCP-ACC does not suffer from such long RTT due to short sub-TCP path. Although long switching interval does not degrade throughput with the long RTT, it can increase packet buffering in intermediate TCP proxy nodes, which may not be a preferable situation for real-time applications.

Table 2 shows throughput of the TCP-ACC and RTTs of the TCP subset. The RTT is measured at each node, which is a time gap between packet arrival time at the node and acknowledged time from a next node. TCP throughput decreases according to the number of hops since the backlog increases at each node due to packets arrived out of order or loss due to PN interference or channel

Table 2 TCP-ACC throughput with varying TCP length

Number of hops	1	2	3	4
Throughput (Kbps)	1408	1104	692	689
Number of paths	0–1	1–2	2–3	3–4
RTT (ms)	7.74	13.3	34.7	25.3

noise. Coincidentally, the RTT increases drastically from the node 3, in which the throughput also decreases. There could be better algorithm with asymmetric TCP, for example, the last and intermediate TCP sinks conduct different operation. However, in this paper, all same TCP-ACCs are supposed to be deployed in mesh nodes by NFV techniques, which is simple and manageable. To investigate proper number of TCP-ACCs along the end-to-end route, we deploy TCP-ACCs every two hops in the same chain topology. Throughput is 360.68 Kbps at the node 4 that is almost a half of one in Table 2; processing delay in the TCP-ACC is less considerable compared to delay from channel sensing and switching. Processing overhead for the TCP-ACC will be measured in our future work since some nodes of CMNs can have very limited hardware platforms.

Figure 9 shows comparison of average TCP throughput and RTT in a chain topology with PN traffic (T_{idle} = 2 s and T_{busy} = 2 s) with two TCP variants, *TP-UrbanX* and TP-CRAHN for CMNs for performance comparison. The *TP-UrbanX* outperforms than other previous TCP versions by about 50 %, which increases or decreases congestion window adaptively based on the feedback information. In [33], TP-CRAHN also improves throughput by 20$\sim$30 % of conventional TCP throughput according to PN traffic load. Compared to the two cognitive TCPs, the TCP-ACC achieves impressive performance, more than four times of conventional end-to-end TCP variants and more than double of TCPs specialized for the CMNs. The RTTs of conventional end-to-end TCP variants are similar with each other while RTT of the *TP-UrbanX* is reduced about 30 % due to adaptive transmission. Basically, the RTT value is tightly coupled to TCP throughput. Even though end-to-end RTT of the TCP-ACC is close to the *TP-UrbanX*, it should be divided into multiple RTTs of sub-TCPs. Unfortunately, the RTT of each sub-TCP connection can be different as shown in Table 2. In such case, TCP throughput is depending on the longest RTT value. From this observation, we can consider to place TCP accelerators first near mesh nodes that suffer from PN interference because the RTT of sub-TCP path can be longer than others due to packet loss if we have to limit the number of TCP-ACCs in the *Urban-X*.

Figure 10 shows a simulation result from two flows in a dumbbell topology illustrated in Fig. 5b. Flow *F1* and *F2* use the same TCP scheme but different destination nodes

5 and 6. The aggregated throughput of all TCP versions was only as much as throughput of a single flow in Fig. 7 because bottleneck links to share. Although the shared links are not fully utilized by two flows, channel switching in nodes 2 and 4 limits aggregated throughput. For example, the node 2 should switch three channels for the nodes 0, 1 and 3 with the defined interval. From this effect, fairness between two flows was mostly achieved as shown in Fig. 10 when both flows start at a same time; throughput of a single flow *F1* is almost a half of the aggregated throughput of *F1* and *F2*.

6 Related works

Many researches about cognitive radio technology have been conducted for the last decade. The cognitive radio improves capacity of ad hoc networks by scavenging white spaces in spectrum and avoiding internal interference [34]. In [5, 11], CMNs are exploited in unlicensed bands. They propose a multi-hop routing protocol and a forwarding scheduling algorithm for the CMNs. Youssef et al. [4] shows overall comparison among multi-hop routing protocols proposed for cognitive multi-hop ad hoc networks.

A few studies have investigated TCP performance under the cognitive radio-based networks. Slingerland et al. [35] analyze TCP performance in dynamic spectrum access (DSA) scenarios using a single-hop cognitive radio access network such as considered by, e.g., 802.22 WRAN. Sarkar et al. [36] also propose TCP Everglades (TCPE) for the single-hop cognitive radio networks. Authors assumed spectrum sensing duration is longer than RTO and extended TCP Westwood [37] with several rules of a knowledge-based module that decides TCP operation.

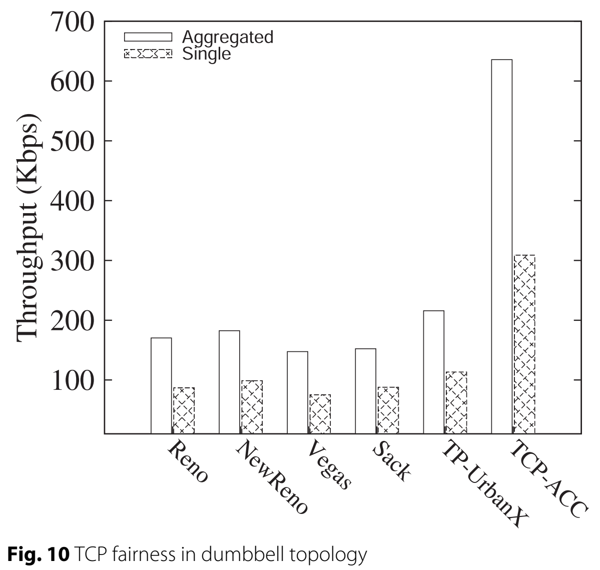


Fig. 10 TCP fairness in dumbbell topology

Chowdhury et al. [33] propose a new transport layer protocol, TP-CRAHN, for cognitive ad hoc networks. In TP-CRAHN, additional TCP states are defined for cognitive operation such as channel sensing and switching, and state transition is triggered by explicit notification from nodes along the TCP path. Kim et al. [38] propose *TP-UrbanX*, a new transport control protocol for cognitive wireless mesh. *TP-UrbanX* uses feedback and TCP acknowledgement information to detect network congestion considering the intrinsic properties of the environment such as channel switching, spectrum sensing, or external interference. It integrates reinforcement-based learning techniques in order to adapt its operation over time to changing network conditions.

Recently, software defined networking (SDN) technology is considered for controlling resources and steering traffic in wireless networks. Liu et al. [39] addressed the multi-flow update problem in the SDN and proposed a polynomial time heuristic algorithm. Yang et al. [8] shows comparative analysis among proposed SDN solutions with research challenges for the SDN-based wireless networks. Granelli et al. [7] describe architecture of cellular and Wi-Fi wireless networks changed by the SDN and introduce standardization activity. And virtualized wireless networks by the SDN are converged with user services effectively [40].

Network function virtualization (NFV) has started by demands from wireless network operators who want to reduce CAPEX and OPEX. ETSI NFV industry specific group is now investigating NFV structure and interoperability between vendors or other standard organization [18]. In [10], software Wi-Fi APs are virtualized by separating logical entities, e.g., MAC from an entire Wi-Fi module. Liang et al. [41] suggests a NFV model for 5G wireless networks for information centric networking. Gember-Jacobson et al. [42] proposes OpenNF, a control model to support service level agreement and networking monitoring in order to optimize performance by coordinating packet processing and forwarding.

7 Conclusions

In this paper, we introduce an architecture of network virtualization for cognitive wireless mesh networks, which is based on distributed computing different to conventional NFV in data center-based cloud computing environment. In this model, each mesh node owns a reconfigurable platform to manage VNF entities that are instantiated dynamically by a central controller. A cognitive wireless mesh network, *Urban-X* that is attractive to provide Internet connectivity to many field devices for IoT without infrastructure can be built on top of the reconfigurable NFV platform to combat interference from various wireless devices in ISM bands. Especially, TCP performance in the *Urban-X* is limited due to complicated node operation for interference avoidance. Compared to previous works that requires modification of TCP, TCP splitting is much simpler than end-to-end TCP optimization schemes in terms of implementation and management. The reconfigurable NFV platform and increasing computing power allow a mesh node to create or remove dynamically a TCP proxy for acceleration without severe overhead to deal with every packet in the TCP layer. From simulation results, we can confirm that NFV-based TCP accelerators can improve TCP throughput by more than four times of conventional TCP variants in the *Urban-X*. As future works, we will implement VNFs including TCP-ACC on our testbed of CMNs using open NFV protocol stacks and measure processing overhead and latency to instantiate or delete VNFs.

Competing interests
The author declares that he has no competing interests.

Acknowledgements
This research was supported by the Gachon University research fund GCU-2015-0044 of 2015.

Received: 20 July 2015 Accepted: 14 September 2015
Published online: 06 October 2015

References

1. Muniwireless project. Http://www.muniwireless.com/
2. Chaska wireless solutions. Http://www.chaska.net/
3. IF Akyildiz, W-Y Lee, MC Vuran, S Mohanty, Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. Comput. Netw. **50**(13), 2127–2159 (2006)
4. M Youssef, M Ibrahim, M Abdelatif, L Chen, AV Vasilakos, Routing metrics of cognitive radio networks: a survey. Commun. Surv. Tutorials IEEE. **16**(1), 92–109 (2014)
5. W Kim, A Kassler, MD Felice, M Gerla, in *Proc. of IFIP Wireless Days*. Urban-X: Towards distributed channel assignment in cognitive multi-radio mesh networks, (2010)
6. J Bajaj, W Kim, S Oh, M Gerla, in *IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2011*. Cognitive radio implementation in ISM bands with microsoft SORA (Toronto, ON, Canada, 2011), pp. 531–535. doi:10.1109/PIMRC.2011.6140018. http://dx.doi.org/10.1109/PIMRC.2011.6140018
7. F Granelli, AA Gebremariam, M Usman, F Cugini, V Stamati, M Alitska, P Chatzimisios, Software defined and virtualized wireless access in future wireless networks: scenarios and standards. IEEE Commun. Mag. **53**(6), 26–34 (2015). doi:10.1109/MCOM.2015.7120042
8. M Yang, Y Li, D Jin, L Zeng, X Wu, AV Vasilakos, Software-defined and virtualized future mobile and wireless networks: a survey. Mob. Netw. Appl. **20**(1), 4–18 (2014)
9. N McKeown, T Anderson, H Balakrishnan, G Parulkar, L Peterson, J Rexford, S Shenker, J Turner, Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
10. P Dely, J Vestin, A Kassler, Cloudmac—using openflow to process 802.11 MAC frames in the cloud. Prax. Informationsverarbeitung Kommun. **36**(1), 53 (2013). doi:10.1515/pik-2012-0068
11. W Kim, A Kassler, MD Felice, M Gerla, in *Proc. of IEEE IPCCC*. Cognitive multi-radio mesh networks on ism bands: A cross-layer architecture, (2010)
12. Z Yan, Z Ma, H Cao, G Li, W Wang, in *Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference On*. Spectrum sensing, access and coexistence testbed for cognitive radio using usrp (IEEE, 2008), pp. 270–274
13. R Dhar, G George, A Malani, P Steenkiste, in *Networking Technologies for Software Defined Radio Networks, 2006. SDR'06.1 St IEEE Workshop On*. Supporting integrated mac and phy software development for the usrp sdr (IEEE, 2006), pp. 68–77
14. P Murphy, A Sabharwal, B Aazhang, in *Signal Processing Conference, 2006 14th European*. Design of WARP: a wireless open-access research platform (IEEE, 2006), pp. 1–5
15. K Tan, H Liu, J Zhang, Y Zhang, J Fang, GM Voelker, SORA: high-performance software radio using general-purpose multi-core processors. Commun. ACM. **54**(1), 99–107 (2011)
16. F Kaltenberger, R Ghaffar, R Knopp, H Anouar, C Bonnet, Design and implementation of a single-frequency mesh network using openairinterface. EURASIP J. Wirel. Commun. Netw. **2010**, 19 (2010)
17. J Sydor, in *Wireless Communication Systems (ISWCS), 2010 7th International Symposium On*. Coral: A wifi based cognitive radio development platform (IEEE, 2010), pp. 1022–1025
18. ETSI NFV ISG. Http://www.etsi.org/technologies-clusters/technologies/nfv
19. ONF. Https://www.opennetworking.org/about/onf-overview
20. OPNFV. Https://www.opnfv.org/
21. GS NFV 002 Network Functions Virtualisation (NFV); Architectural Framework. NFV ISG
22. S Kopparty, S Krishnamurthy, M Faloutous, S Tripathi, in *Proc. of IEEE GLOBECOM*. Split TCP for mobile ad hoc networks, (2002)
23. J Liu, S Singh, Atcp: TCP for mobile ad hoc networks. IEEE J. Sel. Areas Commun. **19**(7), 1300–1315 (2001)
24. A Bakre, BR Badrinath, in *Distributed Computing Systems, 1995., Proceedings of the 15th International Conference On*. I-TCP: indirect TCP for mobile hosts, (1995), pp. 136–143. doi:10.1109/ICDCS.1995.500012
25. T Ouyang, S Jin, M Rabinovich, in *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference On*. Dynamic tcp proxies: Coping with disadvantaged hosts in manets (IEEE, 2009), pp. 530–536
26. K Chandran, S Raghunathan, S Venkatesan, R Prakash, in *Proc. of IEEE ICDCS*. A feedback-based scheme for improving TCP performance in ad-hoc wireless networks, (1998)
27. J Monks, P Sinha, V Bharghavan, in *Proc. of Mobile and Multimedia Communications*. Limitations of TCP-ELFN for ad hoc networks, (2000)
28. MD Felice, KR Chowdhury, W Kim, AJ Kassler, L Bononi, End-to-end protocols for cognitive radio ad hoc networks: an evaluation study. Perform. Eval. **68**(9), 859–875 (2011). doi:10.1016/j.peva.2010.11.005
29. W Kim, AJ Kassler, M Gerla, in *CogART 2011, International Conference on Cognitive Radio and Advanced Spectrum Management*. TCP performance in cognitive multi-radio mesh networks (Barcelona, Spain, 2011), p. 44. doi:10.1145/2093256.2093300. http://doi.acm.org/10.1145/2093256.2093300
30. S Floyd, T Henderson, in *Internet Engeneering Task Force, Request for Comments (Experimental) 2582*. The new Reno modification to TCP fast recovery algorithm, (1999)
31. L Brakmo, L Peterson, TCP Vegas: end to end congestion avoidance on a global internet. IEEE J. Sel. Areas Commun. **13**(8), 1465–1480 (1995)
32. M Mathis, J Mahadavi, S Floyd, A Romanow, in *Internet Engeneering Task Force, Request for Comments (Experimental) 2018*. TCP selective acknowledgment options, (2006)
33. KR Chowdhury, MD Felice, IF Akyildiz, in *Proc. of IEEE INFOCOM*. TP-CRAHN: a transport protocol for cognitive radio ad-hoc networks, (2009)
34. IF Akyildiz, W-Y Lee, KR Chowdhury, Crahns: Cognitive radio ad hoc networks. Ad Hoc Netw. **7**(5), 810–836 (2009). doi:10.1016/j.adhoc.2009.01.001
35. AMR Slingerland, P Pawelczak, R Venkatesha Prasad, A Lo, R Hekmat, in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium On*. Performance of transport control protocol over dynamic spectrum access links, (2007), pp. 486–495. doi:10.1109/DYSPAN.2007.71
36. D Sarkar, H Narayan, in *Proc. of IEEE INFOCOM CWCN*. Transport layer protocols for cognitive networks, (2010)
37. S Mascolo, C Casetti, M Gerla, SS Lee, M Sanadid, in *UCLA CSD Technical Report*. TCP Westwood: congestion control with faster recovery, (2000)
38. W Kim, M Gerla, AJ Kassler, MD Felice, in *12th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WOWMOM 2011*. TP-UrbanX - A new transport protocol for cognitive multi-radio mesh networks (Lucca, Italy, 2011), pp. 1–3.

doi:10.1109/WoWMoM.2011.5986210. http://dx.doi.org/10.1109/WoWMoM.2011.5986210

39. Y Liu, Y Li, Y Wang, AV Vasilakos, J Yuan, in *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*. Achieving efficient and fast update for multiple flows in software-defined networks (ACM, 2014), pp. 77–82
40. Q Duan, Y Yan, AV Vasilakos, A survey on service-oriented network virtualization toward convergence of networking and cloud computing. IEEE Trans. Netw. Serv. Manag. **9**(4), 373–392 (2012)
41. C Liang, FR Yu, X Zhang, Information-centric network function virtualization over 5g mobile wireless networks. IEEE Netw. **29**(3), 68–74 (2015). doi:10.1109/MNET.2015.7113228
42. A Gember-Jacobson, R Viswanathan, C Prakash, R Grandl, J Khalid, S Das, A Akella, in *Proceedings of the 2014 ACM Conference on SIGCOMM*. Opennf: Enabling innovation in network function control (ACM, 2014), pp. 163–174