# WNOS: Enabling Principled Software-Defined Wireless Networking

Zhangyu Guan, *Senior Member, IEEE*, Lorenzo Bertizzolo, *Student Member, ACM*,
Emrecan Demirors, *Member, IEEE*, and Tommaso Melodia, *Fellow, IEEE, Senior Member, ACM*

*Abstract*— This article investigates the basic design principles for a new *Wireless Network Operating System (WNOS), a radically different approach to software-defined networking (SDN) for infrastructure-less wireless networks.* Departing from well-understood approaches inspired by OpenFlow, WNOS provides the network designer with an abstraction hiding (i) the lower-level details of the wireless protocol stack and (ii) the distributed nature of the network operations. Based on this abstract representation, the WNOS takes network control programs written on a centralized, high-level view of the network and automatically generates distributed cross-layer control programs based on distributed optimization theory that are executed by each individual node on an abstract representation of the radio hardware. We first discuss the main architectural principles of WNOS. Then, we discuss a new approach to automatically generate solution algorithms for each of the resulting subproblems in an automated fashion. Finally, we illustrate a prototype implementation of WNOS on software-defined radio devices and test its effectiveness by considering specific cross-layer control problems. Experimental results indicate that, based on the automatically generated distributed control programs, WNOS achieves 18%, 56% and 80.4% utility gain in networks with low, medium and high levels of interference; maybe more importantly, we illustrate how the global network behavior can be controlled by modifying a few lines of code on a centralized abstraction.

*Index Terms*— Software-defined networking, distributed optimization, automated control program generation, wireless ad hoc networks.

## I. INTRODUCTION

**M**OST existing wireless networks are inherently hardware-based and rely on closed and *inflexible architectures* that delay adoption of new wireless networking technologies. Moreover, it is very challenging to control large-scale networks of heterogeneous devices with diverse

capabilities and hardware. Quite the opposite, software-defined radios provide a vast degree of flexibility. At the same time, software radios today lack appropriate abstractions to enable prototyping of complex networking applications able to leverage the cross-layer interactions that characterize wireless operations. To use an analogy from computer systems, trying to build a complex networked application on software radios is today as hard as trying to build a complex piece of enterprise software by writing bare-metal code in a low-level programming language.

There has been no lack of efforts trying to define new networking abstractions in recent years. The notion of software defined networking (SDN) has been introduced to simplify network control and to make it easier to introduce and deploy new applications and services as compared to classical hardware-dependent approaches [2], [3]. The main ideas are (i) to separate the data plane from the control plane (an idea that in different form was already pervasive in the cellular industry); and more importantly (ii) to "control" the network behavior through a centralized programmatic network abstraction. This simplifies the definition of new network control functionalities, which are now defined based on an *abstract* and *centralized* representation of the network.

So far, most SDN work has concentrated on "softwarization" of routing for commercial infrastructure-based wired networks, with some recent work addressing wireless networks [2], [4]–[6]. However, *applications of software-defined networking concepts to infrastructure-less wireless networks (i.e., tactical ad hoc networks, mesh, sensor networks, D2D, IoT) are substantially unexplored.*[1] This is not without a reason. Essentially, distributed control problems in wireless networks are complex and hard to separate into basic, isolated functionalities (i.e., layers in traditional networking architectures). Typical control problems in wireless networks involve making resource allocation decisions at multiple layers of the network protocol stack that are inherently and tightly coupled because of the shared wireless radio transmission medium; conversely, in software-defined commercial wired networks one can concentrate on routing at the network layer in isolation. Moreover, in most current instantiations of this idea, SDN is realized by (i) removing control decisions from the hardware, e.g., switches, (ii) by enabling hardware (e.g., switches, routers) to be remotely programmed through an open and standardized

---

[1]We will discuss a few exceptions in Section VIII: Related Work.

interface, e.g., OpenFlow [2], and (iii) by relying on a centralized network controller to define the behavior and operation of the network forwarding infrastructure. This unavoidably requires a high-speed backhaul infrastructure to connect the edge nodes with the centralized network controller, which is typically not available in wireless networks where network nodes need to make distributed, optimal, cross-layer control decisions at all layers to maximize the network performance while keeping the network scalable, reliable, and easy to deploy [7]. Clearly, these problems, which are specific to wireless, cannot be solved with existing SDN approaches.

**New Approach to Wireless SDN.** For these reasons, in this paper we propose and explore *a new approach to software-defined networking for wireless networks*. At the core, we attempt to answer the following question: is it possible to *automatically generate distributed wireless network control programs* that are *defined* based on a centralized abstraction of the network that hides low-level implementation details; and in this way *bridge the gap between software defined networking and distributed network optimization/control?* Can we, in this way, keep the benefits of distributed network control (where decisions are taken close to the network/channel/interference state without the need for collecting information at a centralized decision making point); and at the same time be able to define the network behavior based on a centralized abstraction? Can we, by answering these questions, develop a *principled approach* to software-defined wireless networking based on cross-layer optimization theory? We attempt to provide a preliminary answer to these compelling questions by studying the core building principles of a *Wireless Network Operating System (WNOS)*. Similar to a computer operating system, which provides the programmer with an abstraction of the underlying machine that hides the lower level hardware operations (e.g., its parallel nature in multi-core systems) and exposes only critical functionalities, WNOS provides the network designer with an abstraction hiding the lower-level details of the network operations. Maybe more importantly, WNOS hides the details of the *distributed implementation* of the network control operations, and provides the network designer with a centralized view abstracting the network functionalities at a high level. Based on this abstract representation, WNOS takes centralized network control programs written on a centralized, high-level view of the network and automatically generates distributed cross-layer control programs based on distributed optimization theory that are executed by each individual node on an abstract representation of the radio hardware. This paper takes a decisive step in this direction and claims the following contributions:

- *WNOS Architecture Design*. We propose an architecture for WNOS by defining three key components: network abstraction, automated network control problem decomposition, and programmable protocol stack.
- *Network Abstraction*. We propose a new w̲i̲reless n̲etwork a̲bstraction f̲ramework *WiNAR* - inspired by the language of network utility maximization (NUM), based on which network designers can characterize diverse desired network behaviors before actual deployment.

- *Automated Decomposition*. We propose the notion of *disciplined instantiation*, based on which user-defined abstract centralized network control problems can be decomposed into a set of distributed subproblems in an automated fashion. Distributed control programs regulate the behavior of each involved node at network run time to obtain the desired centralized behavior in the presence of time-varying local conditions (including channel, traffic, etc.).
- *WNOS Prototyping and Testbed Evaluation*. We outline the design of a WNOS prototype that implements the proposed network abstraction and automated decomposition and solution algorithm generation approach, as well as a newly designed general purpose programmable protocol stack (PPS) that covers all protocol layers. Based on the PPS, a multi-hop wireless ad hoc network testbed is developed using software-defined radios to provide a proof of concept of the WNOS.

Unlike traditional SDN, which relies on centralized or logically centralized control (unsuitable for infrastructure-less wireless networks) and manual design of network control programs, we propose to *define control behaviors* on a centralized network abstraction, while *executing the behaviors* through automatically generated distributed control programs based on local network state information only. Hence, the user-defined centralized cross-layer network control objective can be achieved with no need to distribute network state at all layers of the protocol stack across the global network, which is obviously undesirable. We envision that the resulting WNOS will contribute to bridging the gap between centralized/distributed optimization techniques and software-defined networking - distributed control is not based on design-by-intuition principles, but on rigorous mathematical models based on nonlinear optimization theory.

The remainder of the paper is organized as follows. In Section II, we present the design architecture of WNOS, and then describe the network abstraction framework WiNAR in Section III. We discuss the automated network control problem decomposition approach in Section IV, and present the prototyping and experimental evaluation of WNOS in Sections V and VI, respectively. We discuss limitations and future work in Section VII and review related work in Section VIII. Finally, we draw the main conclusions in Section IX.

## II. WNOS ARCHITECTURE

The architecture of the proposed wireless network operating system (WNOS) is illustrated in Fig. 1. At a high level, the WNOS comprises three key components: network abstraction, network control problem decomposition, and programmable protocol stack (PPS).

**Network Abstraction**. This is the interface through which the network designer can define the network control problem to achieve certain application-specific objectives. Two core functionalities are provided by this component, that is, *network behavior characterization* and *centralized network control problem definition*. WNOS provides the designer with a rich set of network abstraction APIs through which the designer can characterize at a high-level the desired network behavior.
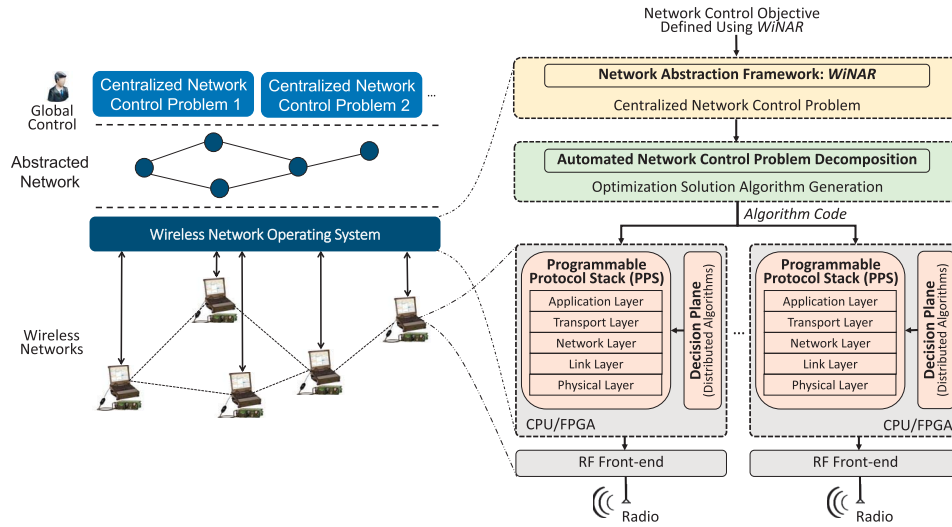
Fig. 1.    Architecture of the wireless network operating system.

Through the API, the designer can define various network control objectives, such as throughput maximization, energy efficiency maximization, delay minimization, or their combinations; can impose different constraints on the underlying physical network, such as the maximum transmission power of each node, available spectrum bandwidth, maximum end-to-end delay, among others. Importantly, to define a network control problem, the designer does not have to consider all implementation details of the networking protocol stack. That is, the designer can select different templates of network protocols, which are programmable with parameters that can be optimized in real time, such as deterministic scheduling vs stochastic scheduling, proactive routing vs reactive routing vs hybrid routing, delay-based vs packet-loss-based congestion control, among others.

It is worth pointing out that the network designer does not need to control protocol parameters manually. Instead, the parameters are optimized by WNOS through automatically generated distributed algorithms. These control objectives, network constraints, and selected protocol templates together serve as the input of the network control problem definition. Then, given a network control problem defined at a high-level, a mathematical representation of the underlying centralized network utility maximization problem is constructed by parsing the network abstraction functions. Details of the network abstraction design will be discussed in Section III.

**Network Control Problem Decomposition**. The resulting centralized network control problem, which characterizes the behavior of the wireless network, is then decomposed into a set of distributed sub-problems, each characterizing the local behavior, e.g., a single session (a flow from one node to another) or a single node. To this end, WNOS first determines a decomposition approach based on the mathematical structure of the network control problem, including whether the problem involves one or multiple sessions, what protocol layers are to be optimized, if the problem is convex or not, among others. Different decomposition approaches can lead to

different structures of the resulting distributed control program with various convergence properties, communication overhead, and achievable network performance [8]–[11].

Through vertical decomposition, a centralized network control problem can be decomposed into subproblems each involving a single or subset of protocol layers, while through horizontal decomposition each of the resulting subproblems involves local functionalities of a single session or node device. Different decomposition approaches can be jointly and iteratively applied if the centralized network control problem involves multiple concurrent sessions and cross-layer optimization of multiple protocol layers. For each of the resulting subproblems, a numerical solution algorithm (e.g., interior-point method) is then selected to solve the problem. Different distributed solution algorithms interact with each other dynamically based on local network information at network run time, by updating and passing a common set of optimization variables. It is worth mentioning that WNOS needs to regenerate the distributed solution algorithms only when the network control objective changes. See Section IV-B for details of the decomposition approach.

**Programmable Protocol Stack (PPS)**. For each of the resulting distributed network control problems, a numerical solution algorithm is selected to solve the optimization problem. The obtained optimization results are used to configure the control parameters of a PPS on each local network device to adapt to the dynamic networking environments in real time. The PPS provides abstractions and building blocks necessary to prototype complex cross-layer protocols based on a high level, abstract representation of the software radio platform without hiding, and instead while retaining control of, implementation details at all layers of the protocol stack and while maintaining platform independence [4], [12]. The control interface between the PPS and the distributed solution algorithms is defined so that (i) the solution algorithm can retrieve network status information from the register plane of the PPS, such as noise and interference power level, queue

status, available spectrum band, among others, and then use the retrieved information as input parameters of the distributed optimization problems; and (ii) based on the optimized solutions, the programmable protocol stack is able to configure in an on-line fashion the parameters of the adopted protocol stack via its decision engine in the decision plane, e.g., update the modulation scheme based on the optimized transmission power hence SINR, configure the TCP window size based on the optimized application-layer rate injected into the network.

## III. NETWORK ABSTRACTION: WiNAR

The objective of the network abstraction component WiNAR is to provide network designers with an interface to characterize network behaviors at a high and centralized level. This goal is however not easy to accomplish because of the following main challenges:

- *Pre-deployment network abstraction.* Unlike traditional network abstraction and resource virtualization [13], where the objective is to abstract or virtualize networks at one or two protocol layers at run time with fixed network topology and known global network information, in our case run-time information is not available in the design phase. For example, the available links that can be used by a session, the neighbors of a node (i.e., those devices that can hear from the node), the interferers of a node, among others are not known a-priori. Therefore, the challenge is to abstract the wireless network before actual deployment by taking run-time uncertainties at all protocol layers into consideration, including time-varying wireless channels, interference coupling among nodes, network topology and traffic load variations, among others.

- *Multi-role network element.* A physical network entity may serve in different roles in the network. For example, a node can be the source or destination of a session, the transmitter, relay or receiver of a link, the neighbor of other nodes, a head of a cluster, a member of the whole network, among others. The network abstraction needs to allow designers to characterize network element behaviors with respect to heterogeneous roles while controlling the same physical network entity.

To address these challenges, elements in WNOS are represented following a three-fold abstraction. At the core of the network abstraction there is a *network representation layer*, which bridges the outer *network control interface layer* and inner *network modeling layer*. Through the network control interface layer, the designer defines the network control objective at a high level, and a mathematical representation of the defined centralized network control problem is then constructed based on the network modeling layer.

**Network Representation**. The network abstraction represents different network entities as two categories of network elements, i.e., *Primitive Element* and *Virtual Element*, defined as follows.

*Definition 1 (Primitive Element):* A primitive element is a network element that represents an individual determined network entity. Two criteria need to be satisfied for each primitive element $\mathcal{A}$:

- $|\{$Network entities represented by $\mathcal{A}\}| = 1$ with $|\cdot|$ being the cardinality of a set, i.e., there exists a one-to-one mapping between any primitive element and a physical network entity.
- For any time instants $t_1 \neq t_2$, $\mathcal{A}(t_1) = \mathcal{A}(t_2)$ always holds, i.e., the one-to-one mapping does not change with time.

Examples of primitive elements include $Node$, $Link$, $Session$, $Link\ Capacity$ and $Session\ Rate$, among others.[2]

*Definition 2 (Virtual Element):* A virtual element represents an undetermined set of network entities, i.e., cannot be mapped to a deterministic set of primitive elements other than at runtime A virtual element $\mathcal{V}$ satisfies

- $|\{$Network entities represented by $\mathcal{V}\}| \geq 1$, i.e., each virtual element is mapped to physical network entities in a one-to-many manner.
- $\mathcal{V} = \mathcal{V}(t)$, i.e., the set of network entities represented by each virtual element is a function of the network run time $t$.

Examples of virtual element include $Neighbors\ of\ Node$ (the set of neighbors of a node), $Links\ of\ Session$ (the set of links used by a session), $Sessions\ of\ Link$ (the set of sessions sharing the same link), among others. The members of a virtual element are primitive elements, e.g., each member of virtual element $Links\ of\ Session$ is a primitive element $Link$.

Then, a wireless network can be characterized using a set of primitive and virtual network elements as well as the cross-dependency among the elements, which is formalized in *Definition* 3.

*Definition 3 (Network):* With primitive elements $\mathcal{A}_m, \mathcal{A}_{m'}$ and virtual elements $\mathcal{V}_n, \mathcal{V}_{n'}$, a network $Net$ can be represented as

$$Net = \{\mathcal{A}_m, \mathcal{V}_n, I(\mathcal{A}_m, \mathcal{A}_{m'}), I(\mathcal{V}_n, \mathcal{V}_{n'}), I(\mathcal{A}_m, \mathcal{V}_n)$$
$$m, m' \in \mathcal{M}_A, m \neq m', n, n' \in \mathcal{N}_V, n \neq n'\} \quad (1)$$

where $\mathcal{M}_A$ and $\mathcal{N}_V$ are the sets of primitive and virtual network elements, respectively, and $I(\mathcal{A}_m, \mathcal{A}_{m'})$, $I(\mathcal{V}_n, \mathcal{V}_{n'}), I(\mathcal{A}_m, \mathcal{V}_n)$ represent the inter-dependencies between primitive elements $\mathcal{A}_m$ and $\mathcal{A}_{m'}$, between virtual elements $\mathcal{V}_n$ and $\mathcal{V}_{n'}$, between primitive element $\mathcal{A}_m$ and virtual element $\mathcal{V}_n$, respectively.

In *Definition* 3, the inter-dependency $I(\cdot, \cdot)$ among different network elements can be characterized as a directed multigraph [14]. Each vertex of the graph represents a network element, and the relationship between two coupled vertices are characterized using one or multiple directed edges connecting the two vertices. All directed edges together characterize the cross-dependency relationship among the network elements. For example, primitive element $Link$ is the holder of another primitive element $Capacity$ (i.e., $Link$ has attribute $Capacity$). Similarly, $Link$ is an attribute of primitive element $Node$ and is a member of virtual element $Links\ of\ Session$. The mutual relationship between primitive element $Node$ and virtual element $Neighbors\ of\ Node$ are characterized using

---

[2]Here, $Link\ Capacity$ and $Session\ Rate$ refer to the network parameters rather than any specific values of the parameters that can be time varying.

two directed edges (hence a multigraph [14]): $Node$ has an attribute $Neighbors\ of\ Node$, each member of which is a $Node$.

- *Has Attribute* characterizes parent-child relationships between network elements, e.g., parent element $Link$ has child elements $Link\ Capacity$ (lnkcap) and $Link\ Power$ (lnkpwr) as its attributes.
- *Each Member is* characterizes set-individual relationships between virtual and primitive elements, e.g., each member of $Links\ of\ Session$ (lnkses) is a $Link$ (lnk).
- *Is Function of* defines the mathematical model of an element based on other elements, e.g., element $Link\ SINR$ (lnksinr) is a function of $Link\ Power$ (lnkpwr).

**Network Control Interfaces**. Based on the network element representation, network control interfaces can then be designed. Based on these, network designers are allowed to characterize network behaviors. Four categories of operations have been defined:

- *Read*: Extract network information from a single or a group of network elements, e.g., extract the set of links used by a session from the attributes of $Node$.
- *Set*: Configure parameters for a single or a group of network elements, e.g., set $Maximum\ Power$ (i.e., maxpwr), which is also an attribute of element $Node$.
- *Compose*: Construct a new expression by mathematically manipulating network parameters obtained through $Read$ operations. For example, add together the power of all links originated from the same node, i.e., sum $Link\ Power$ (lnkpwr) over $Links\ of\ Node$ (lnknd).
- *Compare*: Define network constraints by comparing two expressions obtained using *Compose* operations.

**Centralized Network Control Problem**. Finally, centralized network control problems can be defined based on the network control interfaces. A network control problem comprises of four components: network setting, control variables, network utility and network constraints.

- *Network Setting* can be configured by setting network parameters using *Set* operations and extracted from network elements using *Read* operations. Configurable network parameters include network architecture (single- or multi-hop, flat or clustered topology), spectrum access preferences (scheduled or statistical access), routing preferences (single- or multi-path routing), among others.
- *Control Variables* can be defined by setting (i.e., *Set* operation) network parameters as optimization variables, including transmission power, frequency bandwidth, transmission time, source rate, channel access probability, among others.
- *Network Utility* can be defined by binding (i.e., *Compose* operation) one or multiple expressions with mathematical operations like $+$, $-$, $\times$, $\div$ and mathematical functions like $\log$, $\sqrt{(\cdot)}$ and their combinations.
- *Network Constraints* can be defined by comparing two expressions using *Compare* operations.

Examples of network control problem definition based on the developed abstraction APIs will be given in Section VI.

Given the high-level characterization of network behaviors, the underlying mathematical models of the problem can then be constructed by extracting the mathematical models of each network element using the *Read* operation. The resulting network utility maximization problem is a centralized cross-layer network optimization problem. Our goal is to generate, *in an automated fashion*, distributed control programs that can be executed at individual network devices, which is accomplished by another main component of WNOS, i.e., *Network Control Problem Decomposition* as described in Section IV.

## IV. AUTOMATED NETWORK CONTROL PROBLEM DECOMPOSITION

So far, there is no existing unified decomposition theory that can be used to decompose arbitrary network control problems. Depending on whether we need to decompose coupled network constraints, or coupled radio resource variables; and depending on the decomposition order, a cross-layer network control problem can be theoretically decomposed based on dual decomposition, primal decomposition, indirect decomposition and their combinations. Please refer to [8], [9] and references therein for a tutorial and survey of existing decomposition theories and their applications. In this paper, as one of our major contributions, we propose an automated network control problem decomposition approach based on decomposition of nonlinear optimization problems.

The core objective of the decomposition is two-fold:

- *Cross-layer Decomposition*: Decouple the coupling among multiple protocol layers, resulting in subproblems each involving functionalities handled at a single protocol layer;
- *Distributed Control Decomposition*: Decouple the coupling in radio resource allocation among different network devices, resulting in subproblems that can be solved at each device in a distributed fashion.

Next, we first provide a brief review of cross-layer distributed decomposition theory based on which our automated decomposition approach is designed.

### A. Decomposition Approaches

In this paper we consider duality theory for cross-layer decomposition (while the automated decomposition approach in Section IV-B is not limited to any specific decomposition theory). Consider a network control problem expressed as

$$\underset{\boldsymbol{x}}{\text{maximize}} \sum_{i \in \mathcal{I}} f_i(x_i),$$
$$\text{subject to}: \sum_{i \in \mathcal{J}_i} g_i(x_i) \leq c_j, \quad \forall j \in \mathcal{J} \qquad (2)$$

with $\boldsymbol{x} = (x_i)_{i \in \mathcal{I}}$ being the control vector. The dual function can be constructed by incorporating the constraints into utility in (2) by introducing Lagrangian variables $\boldsymbol{\lambda} = (\lambda_j)_{j \in \mathcal{J}}$,

$$\text{maximize } L(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{i \in \mathcal{I}} f_i(x_i) - \sum_{j \in \mathcal{J}} \lambda_j \left( c_j - \sum_{i \in \mathcal{J}_i} g_i(x_i) \right) \tag{3}$$

where $L(\boldsymbol{x}, \boldsymbol{\lambda})$ is called the Lagrangian of problem (2) [15]. Then, the original problem (2) can be solved in the dual

domain by minimizing (3), i.e., minimizing the maximum of the Lagrangian. This can be accomplished by decomposing (3) into subproblems

$$f_{\text{sub\_1}} = \underset{\boldsymbol{x}}{\text{maximize}} \sum_{i \in \mathcal{I}} f_i(x_i) + \sum_{j \in \mathcal{J}} \lambda_j \left( \sum_{i \in \mathcal{J}_i} g_i(x_i) \right), \quad (4)$$

$$f_{\text{sub\_2}} = \underset{\boldsymbol{\lambda}}{\text{minimize}} \; f_{\text{sub\_1}} - \sum_{j \in \mathcal{J}} \lambda_j c_j, \quad (5)$$

and then iteratively maximizing $f_{\text{sub\_1}}$ over control variables $\boldsymbol{x}$ with given $\boldsymbol{\lambda}$ and updating $\boldsymbol{\lambda}$ with the minimizer of $f_{\text{sub\_2}}$.

The outcome of cross-layer decomposition is a set of network control subproblems each corresponding to a single protocol layer, e.g., capacity maximization at the physical layer, delay minimization through routing at the network layer, among others. The objective of distributed decomposition is to further decompose each of the resulting single-layer subproblems into a set of local network control problems that can be solved distributively at each single network entity based on local network information.

In the existing literature, this goal has been accomplished by designing distributed network control algorithms manually for specific network scenarios and control objectives [7], [16], which however requires deep expertise in distributed optimization. Next, we present a theoretical framework based on which distributed control programs can be designed for *arbitrary* user-defined network control problems.

The core design principle is to decompose a coupled multi-agent network control problem into a set of single-agent subproblems, where each agent optimizes a penalized version of their own utility. Consider a multi-agent network control problem with the objective of maximizing $U(\mathbf{x}) \triangleq \sum_{i \in \mathcal{I}} U_i(\mathbf{x}_i, \mathbf{x}_{-i})$, where $U_i$ is the utility function of agent $i \in \mathcal{I}$, $\mathbf{x} = (\mathbf{x}_i, \mathbf{x}_{-i})$ with $\mathbf{x}_i$ and $\mathbf{x}_{-i}$ representing the strategy of agent $i$ and the strategy all other agents in $\mathcal{I}/i$. Then, the key of distributed decomposition is to construct a penalized individual utility $\widetilde{U}_i(\mathbf{x}_i, \mathbf{x}_{-i})$ for each agent $i \in \mathcal{I}$, expressed as $\widetilde{U}_i(\mathbf{x}_i, \mathbf{x}_{-i}) = \Theta_i(U(\mathbf{x})) + \Gamma_i(\mathbf{x})$, where $\Theta_i(U(\mathbf{x}))$ is the individual item of $U(\mathbf{x})$ associated to agent $i \in \mathcal{I}$, $\Gamma_i(\mathbf{x})$ is the penalization item for agent $i$. Below are three special cases of $\widetilde{U}_i(\mathbf{x}_i, \mathbf{x}_{-i})$ while both individual and penalization items can be customized by network designers to achieve a trade-off between communication overhead and social optimality of the resulting distributed control programs.

- Case 1: $\Theta_i(U(\mathbf{x})) = f_i(\mathbf{x}_i, \mathbf{x}_{-i})$, $\Gamma_i(\mathbf{x}_i, \mathbf{x}_{-i}) = 0$, i.e., *best response without penalization*. In this case, the agents optimize their own original utility $U(\mathbf{x}_i, \mathbf{x}_{-i})$ by computing the best response to the strategies of all other competing agents (i.e., $\mathbf{x}_{-i}$) with zero signaling exchanges.
- Case 2: $\Theta_i(U(\mathbf{x})) = \nabla_{\mathbf{x}_i} U_i(\mathbf{x}^0)(\mathbf{x}_i - \mathbf{x}_i^0)$, $\Gamma_i(\mathbf{x}) = \sum_{j \in \mathcal{I}/i} \nabla_{\mathbf{x}_i} U_j(\mathbf{x}^0)\,(\mathbf{x}_i - \mathbf{x}_i^0)$, with $\mathbf{x}_i^0$ and $\mathbf{x}^0$ being the current strategy of agent $i$ and of all agents. This will result in distributed gradient algorithm [17], where partial cooperation is allowed among the agents by exchanging appropriate signaling messages.

- Case 3: $\Theta_i(U(\mathbf{x})) = U_i(\mathbf{x}_i, \mathbf{x}_{-i}^0)$, $\Gamma_i(\mathbf{x}_i, \mathbf{x}_{-i}) = \sum_{j \in \mathcal{I}/i} \nabla_{\mathbf{x}_i} f_j(\mathbf{x}^0)$, which leads to decomposition by partial linearization (DPL), a newly established decomposition result [9].

It is worth pointing out that, the main contribution of this paper is not to propose a specific new decomposition theory. Instead, our objective is to accomplish network control problem decomposition based on existing theories in an automated fashion [8], [18]–[20]. Next, we describe how this can be accomplished by taking cross-layer decomposition as an example in Section IV-B.

### B. Automated Decomposition

A key step in cross-layer decomposition, as discussed in Section IV-A, is to form a dual function for the original user-defined network control problem by absorbing constraints into the utility. Here, an underlying assumption is that the original problem (2) must have a determined set of constraints, i.e., sets $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{J}_i, \forall i \in \mathcal{I}$ in (2) must be known. This poses significant challenges to automated network control problem decomposition at design phase, because the sets associated to the network elements are not determined *other than at run time*, i.e., they are virtual elements as defined in Section III.

Take virtual element *nbrnd* as an example, i.e., the set of $Neighbors\ of\ Node$. The neighbors of a node may change from time to time because of movement of nodes, joining of new nodes or leaving of dead nodes. Similarly, the set of links along an end-to-end path, the set of sessions sharing the same link and the set of all active links in the network, among others, are also time varying with no predetermined sets. That is to say, a network control problem defined at a high and abstract level may result in many instances of problems with different sets in the constraints and hence different dual variables $\lambda_j$ in the resulting dual function (3). Therefore, *a centralized user-defined network control problem cannot be decomposed by decomposing an arbitrary specific instance of the problem*.

As a core contribution of this work, next we present a new methodology to enable network control problem decomposition in an automated fashion at design phase with no need to know run time network information. At the core, we ask the following question: *For a user-defined centralized abstract network control problem, are there any special set of instances of the problem such that decomposing any problems in the special set decomposes all possible instances? If yes, what is the right approach to obtain such problem instances?* We answer these questions by proposing the notion of *disciplined instantiation (DI)*.

**Disciplined Instantiation**. In a nutshell, the DI technique generates at design time, following a set of predefined rules (as discussed below), a special instance of the user-defined abstract network control problem, such that the abstract problem can be decomposed by decomposing the special instance and the obtained decomposition results can be applied to different arbitrary instances of the control problem at network run time. Notice that DI does not guarantee optimality of the generated distributed algorithms. Instead, the objective of DI is
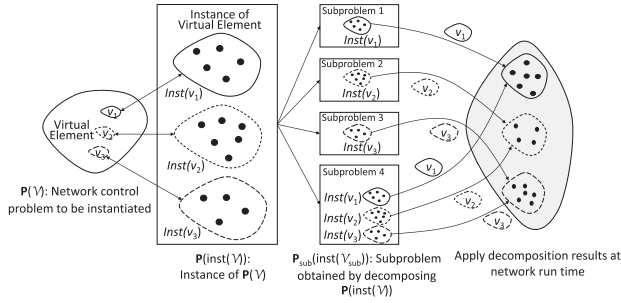
Fig. 2. Basic principle of network control problem decomposition based on disciplined instantiation (DI).

to automate the decomposition of user defined network control problems based on given decomposition rules.

In Fig. 2 we illustrate the basic principle of the DI-based decomposition approach by considering a network control problem that involves three virtual elements $v_1$, $v_2$ and $v_3$, which, e.g., can be $Neighbors \ of \ Node$ for nodes 1, 2 and 3, respectively. Let $inst(v_i)$ represent the instance of virtual element $v_i$, denote $\mathcal{V} = \{v_1, v_2, v_3\}$ as the set of all the three virtual elements and further denote the set of instances for all $v_i \in \mathcal{V}$ as $inst(\mathcal{V})$. Then, the objective of DI is to create a unique instance for each virtual element $v_i \in \mathcal{V}$ such that there exists a one-to-one mapping between $\mathcal{V}$ and $inst(\mathcal{V})$.

Denote $\mathbf{P}(\mathcal{V})$ as the network control problem to be instantiated, and let $\mathbf{P}(inst(\mathcal{V}))$ represent the specific instantiated problem obtained by instantiating $\mathbf{P}(\mathcal{V})$. Then, $\mathbf{P}(inst(\mathcal{V}))$ can be decomposed into a set of subproblems $\mathbf{P}_{\text{sub}}(inst(\mathcal{V}_{\text{sub}}))$ each involving only a subset $\mathcal{V}_{\text{sub}}$ of the virtual elements with $\mathcal{V}_{\text{sub}} \subset \mathcal{V}$. For example, in Fig. 2, $\mathbf{P}(inst(\mathcal{V}))$ has been decomposed into four subproblems, with the first subproblem involving only virtual element $v_1$, the second involving only $v_2$, the third involving only $v_3$ while the fourth involves all three virtual elements. Because of the one-to-one mapping between each virtual network element $v_i$ and its instance $inst(v_i)$, the decomposition results obtained by decomposing $\mathbf{P}(inst(\mathcal{V}))$ are also applicable to the original problem $\mathbf{P}(\mathcal{V})$ represented in virtual elements and hence its various specific instances at network run time.

In the above procedure, the key is to guarantee one-to-one mapping between each virtual element $v_i$ and its instance $inst(v_i)$. This cannot be achieved by generating arbitrarily disjoint instances for different virtual elements $v_i$ because active network elements assume multiple roles as described in Section III. For example, a physical link needs to be involved in the instances of virtual element *"Links of Session"* for all the sessions sharing the link. In the following, we first describe the two rules following which instances are generated in WNOS, i.e., *equal cardinality* and *ordered uniqueness*, and then discuss why the two rules are needed for DI. Before this, we first identify two categories of virtual elements, i.e., *global* and *local* virtual elements. Please refer to Section III for the definition of virtual element.

- A *global virtual element* is a virtual element whose set of physical network entities have the same entity type (e.g., node, or link) and spans over the entire network,

e.g., element *netnd* represents $Nodes \ in \ Network$, the set of all users $\mathcal{I}$ in (2)-(4).

- Differently, a *local virtual element* comprises a subset of physical network entities of the network, and hence is a subset of the corresponding *global virtual element*. For example, local virtual element nbrnd (i.e., $Neighbors \ of \ Node$) is a subset of global virtual element $Nodes \ in \ Network$; as another example, in (2)-(4), since $\mathcal{J}_i$ is a subset of $\mathcal{J}$, i.e., $\mathcal{J}_i \subset \mathcal{J}$, $\mathcal{J}_i$ is a local virtual element while $\mathcal{J}$ is a global virtual element.

*Rule 1: Equal Cardinality*. This rule requires that all the instances for the same type of local virtual elements, e.g., $Neighbors \ of \ Node$, must have the same cardinality, i.e., the same number of members. Instances that satisfy this requirement are called *peer* instances.

In WNOS, this is achieved by *peer random sampling*, a technique that can be used to generate *peer* instances. Specifically, given a user-defined network control problem, the global virtual element denoted as $v^{\text{glb}}$ is first instantiated using a set of pre-determined number $N^{\text{glb}}$ of elements, i.e., $|inst(v^{\text{glb}})| = N^{\text{glb}}$ with $inst(v^{\text{glb}})$ being the instance of the global virtual element $v^{\text{glb}}$ and $|inst(v^{\text{glb}})|$ being the cardinality of $inst(v^{\text{glb}})$. The resulting instance $inst(v^{\text{glb}})$ will be used to serve as the mother set to generate instances for those local virtual elements $v^{\text{lcl}}$.

Then, each local virtual element $v^{\text{lcl}}$ can be instantiated by randomly selecting a subset of members from the mother set $inst(v^{\text{glb}})$, i.e., the instance of the global virtual element $v^{\text{glb}}$. Denote the resulting subset instance as $inst(v^{\text{lcl}})$, then we have $|inst(v^{\text{lcl}})| = N^{\text{lcl}}$ and $inst(v^{\text{lcl}}) \subset inst(v^{\text{glb}})$, where $N^{\text{lcl}}$ is the cardinality of instances for local virtual elements. Specific examples will be discussed in Sections IV-C and VI-A for generating instances based on this rule.

*Rule 2: Ordered Uniqueness*. With this rule, a unique instance will be generated for each local virtual element $v^{\text{lcl}}$. This means that no two subsets generated by peer random sampling will be *identical*.

In WNOS, this is accomplished by *hash checking*, as in (6):

$$inst(v^{\text{lcl}}) \xrightarrow{sort(\cdot)} inst'(v^{\text{lcl}}) \xrightarrow{h(\cdot)} hid^{\text{lcl}}, \qquad (6)$$

where the members of $inst(v^{\text{lcl}})$, i.e., the instance for local virtual element $v^{\text{lcl}}$, are first sorted, and then a unique id $hid^{\text{lcl}}$ is calculated for the sorted instance $inst'(v^{\text{lcl}})$ using a hash function $h(\cdot)$. A hash function is a function that can be used to map an arbitrary-size data (instances in our case) to a fixed-size id [21]. In WNOS, hash function is used to enable fast uniqueness checking by generating an id for each of the generated instances.

**Rationale for The Rules.** The above two rules together guarantee that there exist a one-to-one mapping between the local virtual elements and their instances. As discussed above, this is the key to guarantee that the decomposition results obtained based on DI are also applicable at network run time. To show how the one-to-one mapping can be achieved following the two rules, we take Fig. 3 as an example where A, B, C and D represent four specific instances of local virtual element $v^{\text{lcl}}$, with each member in the set representing
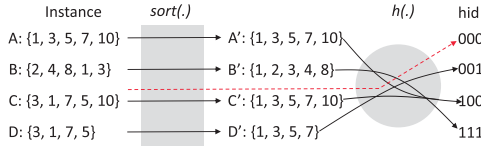
Fig. 3.   Illustration of hash mapping.

a primitive element (see Section III for the definition), e.g., an individual node. Denote A′, B′, C′ and D′ as the sets resulting from sorting the members of A, B, C and D, respectively.

It can be seen that set A is mapped to a three-digit id 100 while B is mapped to 111. Instance C is also mapped to 100 since its sorted instance C′ is identical to A′. Note that in Fig. 2, in each instantiated sub-problem $\mathbf{P}_{\mathrm{sub}}(inst(\mathcal{V}_{\mathrm{sub}}))$ the members of each instance may be re-ordered by the mathematical manipulations decomposing the instantiated network control problem $\mathbf{P}(inst(\mathcal{V}))$, e.g., forming and decomposing the dual function in (3), (4) and (5). In (6) function $sort(\cdot)$ guarantees that the same instances are always mapped to the same id regardless of the order of its members; otherwise, instance C will be mapped to a different id 000 as the red dashed arrow indicates in Fig. 3.

Moreover, in Fig. 3 instance D is mapped to an id different from that of A and C. This implies that an instance A and its subset instance D cannot be used at the same time for disciplined instantiation (DI); otherwise, it will be hard to separate them if they appear in the same instantiated network control sub-problems $\mathbf{P}_{\mathrm{sub}}(inst(\mathcal{V}_{\mathrm{sub}}))$. In DI, this is prevented by keeping all local instances *peer*, i.e., it holds true for all local virtual elements that no instance is a proper subset of any other instances. If hash checking finds that a new instance for local virtual element $v^{\mathrm{lcl}}$ is *identical* to any existing instances, i.e., they have the same id, another instance will be created $v^{\mathrm{lcl}}$ by peer random sampling.

Following the above two rules, a unique specific instance can be obtained for each of the virtual local elements, while there exists a one-to-one mapping relationship between the local virtual elements and their instances. Thus, decomposing the original network control problem can be equivalently achieved by decomposing the corresponding specific instantiated problem, which is machine understandable and can be automatically conducted.

Finally, with instantiated network elements in $\mathcal{V}$, the dual function (3) of the network control problem $\mathbf{P}(\mathcal{V})$ can be obtained and then decomposed as described in Section IV-A. To enable automated decomposition, the resulting dual function is represented using a tree. The whole dual function $\mathbf{P}$ is represented using the root node, which can be represented as the sum of a leaf node and an intermediate node, which can be further represented in a similar way. In this way, the decomposition of a network control problem (the dual function if dual decomposition is used) can be conducted in an automated fashion by traveling through all leaf nodes of the tree. The output of automated decomposition is a set of distributed subproblems each involving a single protocol layer and single network node. For each subproblem, a solution algorithm will be automatically generated and the resulting

optimized network protocol parameters will be used to control the programmable protocol stack (PPS), which will be discussed in Section V: WNOS Prototyping.

### C. Toy Example of DI-Based Decomposition

Consider the following cross-layer network control problem:

$$\text{maximize} \sum_{s \in \mathcal{S}} R_s$$
$$\text{subject to} : \sum_{s \in \mathcal{S}_l} R_s \leq C_l(\mathbf{\Pi}), \ \forall l \in \mathcal{L} \qquad (7)$$

where the objective is to maximize the sum of rate $R_s$ of all flows $s \in \mathcal{S}$ at the transport layer; subject to the constraints that, for each link $l \in \mathcal{L}$, the aggregate rate of all the flows in $\mathcal{S}_l$, i.e., the set of links sharing link $l$, cannot exceed the capacity of the link $C_l(\mathbf{\Pi})$ achievable with transmission strategies $\mathbf{\Pi}$ at the physical layer; by jointly controlling $R_s$ and $\mathbf{\Pi}$. Next, we show how the problem can be decomposed into two single-layer control problem through DI-based decomposition, while more examples of the DI-based decomposition that consider different network problems will be discussed in Section VI.

As defined in Section IV-B, $\mathcal{S}$ (i.e., the set of all flows) and $\mathcal{L}$ (i.e., the set of all links) are global virtual elements while $\mathcal{S}_l \subset \mathcal{S}$ is a local virtual element. In favor of easy illustration, consider cardinality $N^{\mathrm{glb}} = 3$ for global virtual elements $\mathcal{S}$ and $\mathcal{L}$ and $N^{\mathrm{lcl}} = 2$ for local virtual elements $\mathcal{S}_l$, $\forall l \in \mathcal{L}$. Then, the global virtual elements $\mathcal{S}$ and $\mathcal{L}$ can be instantiated as $\mathcal{S} = \{1, 2, 3\}$ (i.e., the network has in total three flows) and $\mathcal{L} = \{1, 2, 3\}$ (i.e., the network has in total three links). The instance of $\mathcal{S}$ will then be used as the mother set for instantiating local virtual elements $\mathcal{S}_l$, $\forall l \in \mathcal{L}$, as follows.

First, according to rule 1, i.e., equal cardinality, all $\mathcal{S}_l$ must have the same number of members. According to rule 2, no two or more $\mathcal{S}_l$ will be the same in the sense of ordered uniqueness. If local virtual element $\mathcal{S}_l$, i.e., the set of sessions sharing link $l$, is instantiated to $\{1, 2\}$ and $\{2, 1\}$ for links $l = 1$ and $l = 2$, respectively, the resulting two instances will have the same set of ordered members, which violates rule 2 and hence are not allowed in DI. An example instantiation that meets the two rules, which can be generated by a combination of peer randomly sampling and hash checking as discussed earlier in this section, is $\mathcal{S}_1 = \{1, 2\}$, $\mathcal{S}_2 = \{1, 3\}$ and $\mathcal{S}_3 = \{2, 3\}$. Let $\mathcal{L}_s$ represent the set of links used by flow $s$. Then, according to the instances for $\mathcal{S}_l$, the instances for $\mathcal{L}_s \subset \mathcal{L}$ can be given as $\mathcal{L}_1 = \{1, 2\}$, $\mathcal{L}_2 = \{1, 3\}$ and $\mathcal{L}_3 = \{2, 3\}$. As a result, problem (7) can be instantiated as

$$\text{maximize} \ R_1 + R_2 + R_3$$
$$\text{subject to} : R_1 + R_2 \leq C_1(\mathbf{\Pi})$$
$$R_1 + R_3 \leq C_2(\mathbf{\Pi})$$
$$R_2 + R_3 \leq C_3(\mathbf{\Pi}). \qquad (8)$$

Consider dual decomposition as discussed in Section IV-A, then the dual function of (8) can be written as

$$\text{maximize} \ R_1 + R_2 + R_3 + \lambda_1[C_1(\mathbf{\Pi}) - R_1 - R_2]$$
$$+ \lambda_2[C_2(\mathbf{\Pi}) - R_1 - R_3] + \lambda_3[C_3(\mathbf{\Pi}) - R_2 - R_3], \quad (9)$$

where $\lambda_1, \lambda_2$ and $\lambda_3$ are dual coefficients. By decomposing (9), problem (8) can be decomposed into two single-layer problems:

Transport Layer :

$$\begin{cases} \text{maximize } R_1 - \lambda_1 \ R_1 - \lambda_2 \ R_1, \text{ for } s = 1 \\ \text{maximize } R_2 - \lambda_1 \ R_2 - \lambda_3 \ R_2, \text{ for } s = 2 \\ \text{maximize } R_3 - \lambda_2 \ R_3 - \lambda_3 \ R_3, \text{ for } s = 3 \end{cases} \quad (10)$$

Physical Layer :

$$\text{maximize } \lambda_1 C_1(\mathbf{\Pi}) + \lambda_2 C_2(\mathbf{\Pi}) + \lambda_3 C_3(\mathbf{\Pi}), \quad (11)$$

where, at the transport layer, each flow $s \in \{1, 2, 3\}$ maximizes its own utility by adjusting its transmission rate $R_s$ with given dual coefficients; while the physical-layer subproblem maximizes a weighted-sum-capacity by adapting the transmission strategies $\mathbf{\Pi}$, i.e., the transmission power of individual nodes.

Now we show how the the decomposition results can be applied at network run time by taking the transport-layer subproblem for $s = 1$ as an example while the same principles can also be applied to other subproblems. For $s = 1$, the utility of the subproblem can be rewritten as $R_1 - (\lambda_1 + \lambda_2)R_1$. Then, to determine the dual coefficients of $R_1$ at run time, we only need to identity the local virtual element corresponding to instance $\{\lambda_1, \lambda_2\}$, which is virtual element $\mathcal{L}_1$ according to the instantiation results of $\mathcal{L}_s$. This means that, at run time, the dual coefficients for flow $s$ should be collected, e.g., at the source node of flow $s$, from those links used by the flow.

## V. WNOS Prototyping

So far, we have described the basic design principles of network abstraction and automated network control problem decomposition. To validate the proposed new ideas, we prototyped WNOS over a testbed with software defined radios. This is however not easy because of several challenges: (i) with WNOS, one should be able to deploy a large scale network by creating only a single piece of code to define the network control objective in a centralized manner. Since different SDR front-ends are controlled by different hosts, it is challenging to distribute and synchronize the generated code among the hosts; and (ii) there is no existing programmable protocol stack (PPS) that supports cross-layer control with optimizable protocol parameters at each layer. To address these challenges, next we first discuss the prototyping approach and then describe the newly designed PPS.

### A. Prototyping Approach

A proof of concept of WNOS has been deployed over a network with 21 USRP software radios. The prototyping diagram is illustrated in Fig. 4, which follows a hierarchical architecture with three tiers, i.e., WNOS control host, SDR control host and SDR front-end.

At the top tier of the hierarchical architecture is the WNOS control host, based on which one can specify the network control objective using the provided network abstract framework WiNAR. The output of this tier is a set of automatically generated distributed solution algorithms, which will be sent
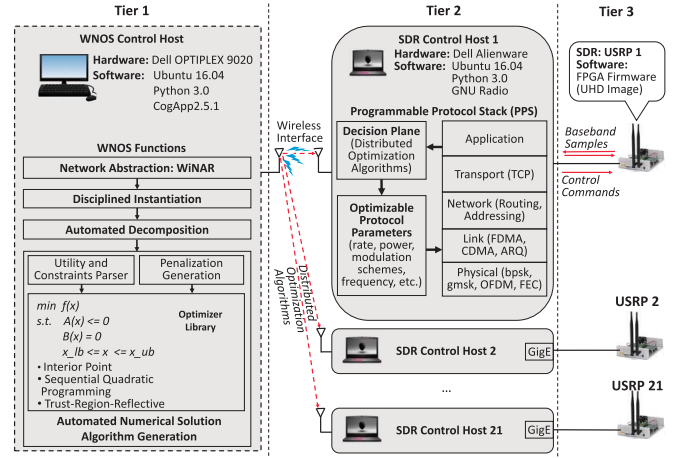


Fig. 4. Prototyping diagram of WNOS.

to each of the SDR control hosts. At the second tier, the programmable protocol stack (PPS) is installed on each of the SDR control hosts. The distributed optimization algorithms received from the WNOS control host are stored at the decision plane of the PPS. At run time, the PPS will be compiled to generate operational code to control the SDR front-ends of the third tier. Finally, each of the SDR front-ends (i.e., USRP) receives the baseband samples from its control host via Gigabit Ethernet (GigE) interface and then sends them over the air with transmission parameters dynamically specified in the control commands from the SDR control hosts.

The primary benefit of prototyping WNOS based on an hierarchical architecture is to enable scalable network deployment. Specifically, the tier-1 WNOS control host is connected to all tier-2 SDR control hosts via wireless interfaces (which is Wi-Fi in current prototype), through which the generated distributed algorithms can be automatically *pushed* to and installed at each of the SDR control hosts. Hence, one needs to create a single piece of code only in order to control all the 21 USRPs.

On the WNOS control host, which is a Dell OPTIPLEX 9020 desktop running Ubuntu 16.04, four key WNOS functions have been implemented using a combination of Python 3.0 and CogApp 2.5.1, including the wireless network abstraction framework WiNAR, disciplined instantiation, automated decomposition as well as automated numerical solution algorithm generation (refer to Sections III and IV for the techniques). We base our development on Python to take advantage of its high programming efficiency and high-level expressiveness and the flexible, open-source programming interfaces to GNU Radio for controlling USRPs. CogApp is an open-source software written in Python for template programming [22], a programming technique based on which the automated numerical solution algorithm generation has been implemented in the current prototype.

### B. Programmable Protocol Stack

As shown in Fig. 4, the programmable protocol stack (PPS) is installed on each of the five SDR control hosts, which

are Dell Alienware running Ubuntu 16.04. The PPS has been developed in Python on top of GNU Radio to provide seamless controls of USRPs based on WNOS. To this end, a decision plane has been designed to install those distributed optimization algorithms generated by the WNOS control host and then *pushed* to the SDR control hosts.

The developed PPS covers all the protocol layers. Based on the protocol stack, a multi-hop wireless ad hoc network testbed has been established using software-defined radio devices to verify the effectiveness of the designed wireless network operating system (WNOS).

The application layer opens end-to-end sessions for transferring custom data such as files, binary blobs, as well as random generated data, among others. A session can be established between any two network entities and multiple sessions can be established at the same time. Programmable parameters include the number of sessions and the number of hops in each session, as well as the desired behavior of each session, e.g., maximum/minimum rate, power budget of the nodes, among others.

The transport layer implements segmentation, flow control, congestion control as well as addressing. This layer supports end-to-end, connection-oriented and reliable data transfer. To accomplish this, a *Go-Back-N* sliding window protocol is implemented for flow control and congestion control, and transport layer acknowledgments are used to estimate the end-to-end Round Trip Time (RTT), which serves as an estimate of network congestion. Programmable parameters at this layer include transmission rate, sliding window size and packet size, among others.

The network layer implements host addressing and identification, as well as packet routing. The network layer is not only agnostic to data structures at the transport layer, but it also does not distinguish between operations of the various transport layer protocols. Routing strategies can be programmed at this layer.

At datalink layer the core functionalities include fragmentation/defragmentation, encapsulation, network to physical address translation, padding, reliable point-to-point frame delivery, Logical Link Control (LLC) and Medium Access Control (MAC) among others. In particular, the reliable frame delivery employs an hybrid LLC's *Stop and Wait* ARQ protocol and Forward Error Correction (FEC) mechanism (*Reed-Solomon* coding), such that frames are padded with FEC code and retransmissions are performed when the link is too noisy. The FEC is dynamic, reprogrammable, and can automatically adapt to the wireless link conditions at fine granularity, by increasing or decreasing the channel coding rate based on the observed packet error rate. Programmable parameters at this layer include channel coding rate, maximum retransmission times, and target residual link-level packet error rate, among others.

Finally, the physical layer features both CDMA and OFDM access schemes, yet with a wide set of modulation schemes supported, including Binary phase-shift keying (BPSK), Quadrature phase-shift keying (QPSK), Gaussian Minimum Shift Keying (GMSK) among others. Programmable parameters at the physical layer include modulation

```
# Network setting
net = new_ntwk(adhoc)
net.add_node()
net.add_sess()
net.set_protocol(CDMA)
net.set_protocol(TCP_VEGAS)

# Define network utility
net.make_var('wos_x', [netses, sesrate], [all, None])
expr = mkexpr('sum(wos_x)', 'wos_x')
net.set_utlt(expr)

# Define network constraints
net.make_var('wos_y', [netlnk, lnkses, sesrate], [every, all, None])
net.make_var('wos_z', [netlnk, lnkcap], [every, None])
cstr = mkexpr('sum(wos_y) <= wos_z', 'wos_y', 'wos_z')
net.add_cstr(cstr)

# Network control problem decomposition
net.dcmp('dual', 'dpl')
```

User-defined network utility

Network constraints

Dual-based cross-layer decomposition
Distributed decomposition using DPL

Fig. 5.　WNOS definition of network control problem based on WiNAR.

schemes, transmission power, and receiver gain, among others.

## VI. Experimental Evaluation

We evaluate the effectiveness, flexibility as well as scalability of the proposed WNOS by conducting experiments on the developed WNOS prototype, which is a testbed on large-scale USRP testbed with 21 nodes. Next, we first demonstrate in Section VI-A the automated network control problem decomposition by considering specific network, and then show the experimental evaluation results in VI-B.

### A. Automated Network Control Problem Decomposition

We showcase how WNOS works by taking the network control problem in [16] as an example. The objective of the network control problem, referred to as JOCP in [16], is to maximize the sum utility of a set of concurrent sessions by jointly optimizing the transmission rate of each session at the transport layer and controlling the transmission power of each transmitter at the physical layer. The underlying mathematical model of the network control problem is given as

$$\text{maximize} \sum_{s \in \mathcal{S}} U_s(x_s)$$
$$\text{subject to}: \sum_{s:l \in L(s)} x_s \leq c_l(\mathbf{P}), \ \forall l \in \mathcal{L}$$
$$\mathbf{x}, \ \mathbf{P} \succeq 0 \quad (12)$$

where $\mathbf{x} \triangleq (x_s)$, with $x_s$ representing the transmission rate of session $s \in \mathcal{S}$, $\mathbf{P} \triangleq (P_n)$ is the transmission power profile of all the involved network nodes, $c_l(\mathbf{P})$ is the achievable capacity of link $l \in \mathcal{L}$ on path $L(s)$, and $U_s(x_s)$ is the achievable utility of session $s$. Readers are referred to [16] for details of the network control problem.

**WNOS Representation**. Based on the network abstraction interface provided by WNOS, i.e., *WiNAR*, network control problem JOCP can then be defined in a high-level centralized abstract fashion as shown in Fig. 5, where network utility in (12) is defined as the sum rate of all sessions,

Fig. 6. Dual function of the instantiated centralized network control problem.

| Link | Session Instances |
|---|---|
| 0 | 3, 4 , 6, 7, 8, 14, 15, 17, 18, 19 |
| 1 | 0, 2, 3, 6, 8, 10, 11, 12, 16, 17 |
| 2 | 0, 5, 6, 7, 11, 13, 14, 15, 18, 19 |
| 3 | 0, 1, 4 , 6, 10, 11, 13, 16, 17, 19 |
| 4 | 0, 3, 4 , 7, 8, 12, 13, 14, 18, 19 |
| 5 | 1, 3, 6, 10, 11, 12, 13, 15, 18, 19 |
| 6 | 0, 1, 3, 5, 6, 7, 11, 14, 15, 16 |
| 7 | 1, 3, 4 , 6, 12, 14, 15, 16, 17, 19 |
| 8 | 1, 2, 5, 6, 7, 8, 9, 10, 12, 14 |
| 9 | 0, 2, 3, 4 , 5, 12, 13, 15, 16, 17 |
| 10 | 0, 1, 4 , 6, 7, 8, 9, 11, 16, 19 |
| 11 | 4 , 5, 6, 7, 14, 15, 16, 17, 18, 19 |
| 12 | 2, 3, 4 , 6, 7, 8, 12, 15, 17, 18 |
| 13 | 4 , 6, 8, 13, 14, 15, 16, 17, 18, 19 |
| 14 | 0, 1, 3, 4 , 5, 6, 8, 12, 16, 17 |
| 15 | 2, 3, 6, 10, 11, 12, 13, 15, 16, 19 |
| 16 | 1, 2, 3, 5, 6, 8, 9, 12, 15, 16 |
| 17 | 1, 2, 7, 8, 12, 13, 14, 16, 18, 19 |
| 18 | 0, 1, 3, 4 , 6, 7, 12, 13, 18, 19 |
| 19 | 0, 2, 4 , 5, 8, 12, 14, 15, 16, 17 |

i.e., $U_s(x_s) = x_s$ for each session $s \in \mathcal{S}$ in (12). In the high-level definition in Fig. 5, there are three virtual network elements, i.e., netses, netlnk and lnkses, representing the set of all sessions, the set of all links and the set of links used by a session, respectively. The former two are *global* elements representing the set of all sessions $\mathcal{S}$ and the set of all links $\mathcal{L}$ in the network, respectively. The third virtual element lnkses, i.e., $\{s : l \in L(s)\}$ in (12) represents the set of sessions sharing the same link, and hence is a *local* element associated with each link instance of global virtual element netlnk.

For instantiation of global virtual elements, the cardinality of the set of instances is by default set to 20, while it is set to 10 for local virtual element instantiation. Based on this, WNOS can generate up to 184756 unique instances for each abstract network element, which is sufficient to decompose moderate-size network control problems with up to hundreds of constraints.

Table I shows the instantiation result of global virtual element *Links in Network* and local virtual element *Sessions of Link*, where for each link instance a unique set of sessions sharing the link was constructed based on *peer sampling* and *hash checking* as described in Section IV-B. The set of links used by a session instance can then be instantiated accordingly, e.g., $\{0, 3, 4, 7, 9, 10, 11, 12, 13, 14, 18, 19\}$ for Session 4 as underlined in Table I.

**Problem Decomposition**. Consider dual-based cross-layer decomposition (refer to Section IV-A for the decomposition theory) as specified in the high-level abstract network control problem definition in Fig. 5. The resulting dual representation of the user-defined centralized network control (12) is given in Fig. 6, where the network constraints of (12) (constraints component) have been absorbed into the utility function (utility component), by introducing dual coefficients lbd_id with id being the index of the link instance to which each dual coefficient is associated. Our objective is to decompose the initial user-defined centralized network control problem by decomposing the corresponding dual representation into a set of sub-problems each involving a single network element, e.g., a single session 4 as shown in Fig. 6.

To this end, the dual representation is further represented as a three-level tree of sub-expressions, where level 0 is the initial dual expression in Fig. 6, level 1 comprises sub-expressions of sum operation in the initial representation, while each expression at level 1 can be further represented as a multiplication of two sub-expressions at level 2. Then, to decompose the user-defined central network control problem, we only need

to walk over all level-1 elements of the tree and determine to which subproblem each of the elements should be categorized. For cross-layer decomposition, this can be accomplished as follows:

- For each level-1 sub-expression, extract the protocol layer information of the network element involved in the sub-expression using *Read* operations defined in Section III.
- Categorize the sub-expression into the sub-problem of the corresponding protocol layer.

The decomposition will result in a set of subproblems each involving only a single protocol layer. Similarly, each of the resulting sub-problems can be further decomposed into sub-problems each involving a single network element, e.g., node, session, so that they can be solved in a distributed fashion and result in distributed control actions.

*Mapping From Instantiation to Abstract Domain*: Recall in Section IV-B that our objective is to construct a set of instantiations of the user-defined high-level abstract network control so that decomposing any of the problem instantiations decomposes the abstract problem. As described in Section IV-B, this is guaranteed by the one-to-one mapping between virtual network elements and their instantiations obtained through peer-sampling and hash checking. Next, we show the one-to-one mapping taking the following instantiated transport-layer subproblem as an example:

$$sesrate\_04 - sesrate\_04 * (lbd\_09 + lbd\_11$$
$$+ lbd\_12 + lbd\_13 + lbd\_14 + lbd\_18 + lbd\_19$$
$$+ lbd\_00 + lbd\_03 + lbd\_04 + lbd\_07 + lbd\_10). \quad (13)$$

We can see that (13) is the subproblem by categorizing those Session 4 components in Fig 5. In (13), dual coefficients *lbd*

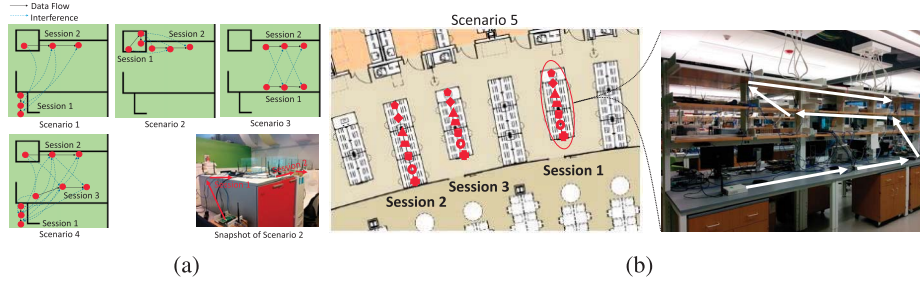(a)                                                                (b)

Fig. 7.   Experimental Scenarios: (a) Scenarios 1-4 and (b) Scenario 5.

are parameters that will be received by the source node of session 4 from links with indexes {09, 11, 12, 13, 14, 18, 19, 00, 03, 04, 07, 10}, which is namely the instantiation set for local virtual element *Links of Session* for Session 4, as shown in Table I. Hence, (13) can be further represented for all sessions, which corresponds to the global virtual network element *Sessions of Network*,

$$sesrate - sesrate * sum(\mathbf{lbd}) \tag{14}$$

where **lbd** represents the vector of dual parameters received by each source node of the session at network run time.

### B. Software-Defined Radio Implementation

We test WNOS on the designed SDR testbed in five different networking scenarios. As shown in Fig. 7, Scenarios 1-3 deploy six nodes and two traffic sessions; while Scenario 4 considers nine nodes and three traffic sessions, with each session spanning over two hops. In Scenario 5, three sessions are deployed over 21 nodes, with six hops for each session. Six spectrum bands in the ISM bands are shared by the 21 USRPs, with bandwidth of 200 kHz for each spectrum band. At each USRP, the data bits are first modulated using GMSK and then sampled at sampling rate of configured 800 kHz. Reed-solomon (RS) code is used for forward error coding (FEC) with coding rate ranging from 0.1 to 0.4 at a step of 0.1. In Fig. 7(a) the data flow is indicated using solid arrows, and the interference is indicated using dashed arrows. There is interference only when two nodes share the same spectrum band. The code to repeat experiments is available on website: http://www.ece.neu.edu/wineslab/WNOS.php.

Through the experiments, we seek to demonstrate the following properties:

- *Effectiveness.* Through experiments in Scenarios 1-3, we show that WNOS-based network optimization outperforms non-optimal or purely locally optimal (greedy) network control;
- *Flexibility.* Through experiments in Scenarios 4 and 5, we showcase the flexibility of WNOS in modifying the global network behavior by changing control objectives and constraints.
- *Scalability.* In Scenario 5 we show the scalability of WNOS by deploying code over a large-scale network.

**Effectiveness**. We show the effectiveness of WNOS on the developed SDR testbed. At the physical layer, two spectrum bands are used, 1.3 GHz and 2.0 GHz. If two transmitters

(either source or relay) are tuned to the same spectrum band, their transmissions will interfere as shown in Fig. 7(a). The control objective is to maximize the sum utility of the two sessions (referred to as Control Program 1, which can be specified by *expr = mkexpr('sum(log(wos_x))', 'wos_x')* as in (12)) by jointly controlling the transmission rate at the transport layer and the transmission power at the physical layer. For each session, the utility is defined as the logarithm of the achievable end-to-end throughput, and considers only those successfully delivered packets.

Five schemes have been tested: (i) WNOS-T-P: transport and physical layers are jointly controlled using the optimization algorithms automatically generated by WNOS; (ii) WNOS-T: only the transport layer rate is controlled by WNOS; (iii) WNOS-P: only the physical layer power is controlled by WNOS; (iv) No Control: neither transport or physical layer are controlled by WNOS, and the rate and power are selected randomly; and (v) Best Response: maximum rate and power are used at the transport and physical layers, respectively. In all schemes, the initial operating points (i.e., rate and power) are randomly generated. Power control is implemented by controlling the transmit gain (which takes value from 0 to 30 dB) of the FPGA of USRP N210s.

We first validate WNOS in an environment with tight coupling of different sessions via interference. Fig. 8(a) reports the achievable end-to-end throughput vs time for the two sessions (in terms of packets/s) in network scenario 2. The packet length is set to 2048 bits. We observe that the throughput of the two sessions converges to 1.6 and 2.3 packets/s when they are active simultaneously, i.e., in the interference-limited region in Fig. 8(a). After session 2 is done transmitting all of its packets (3000 packets), session 1 operates in the interference-free region and its throughput starts to increase significantly.

The average performance of the five schemes is reported in Figs. 8(b), (c) and (d) for network scenarios 1, 2 and 3, respectively. As discussed above, the three network scenarios have been designed to present different levels of interference between the two sessions. The sum utility achievable by the best response scheme is a good indicator since with this scheme each node always transmits at the maximum power, i.e., 30 dB transmit gain is used for USRP N210. For example, with the least amount of interference in scenario 1, best response achieves the highest sum utility of 1.44 compared to 0.89 in network scenario 3. From the three figures, it can be seen that, compared with no control, considerable performance
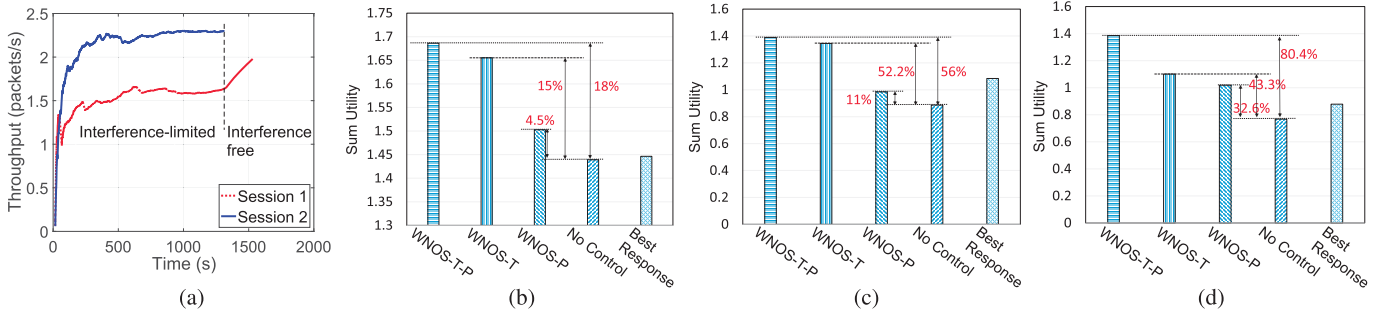
Fig. 8. (a) End-to-end throughputs of sessions 1 and 2; Average sum utility of scenarios (b) 1, (c) 2 and (d) 3.
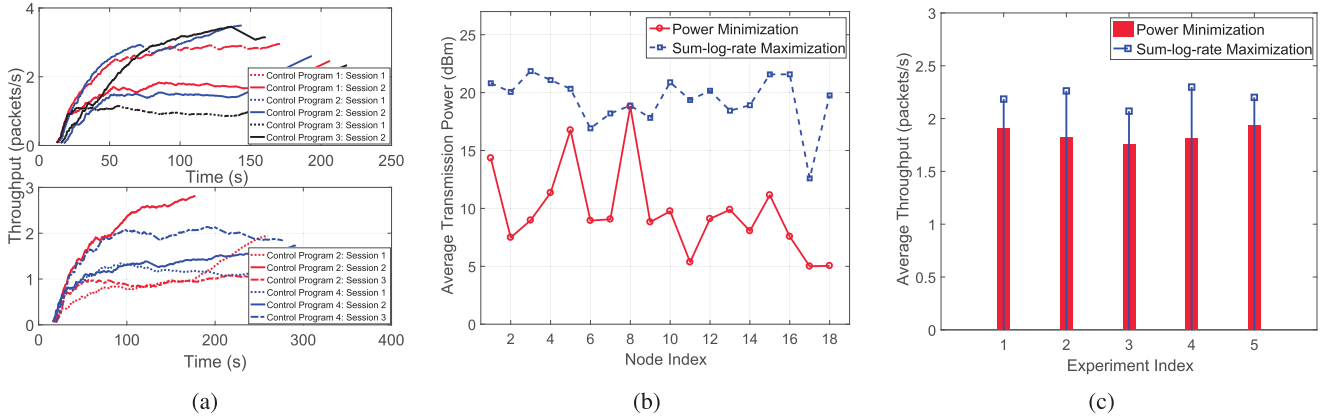


Fig. 9. (a) Network behaviors with different control programs; (b) Transmission power and (c) throughput resulting from two different control objectives: sum-log-rate maximization and power minimization.

gain can be achieved by the WNOS-T-P, i.e., with transport and physical layers jointly controlled, and this gain increases as the interference level increases. Once more, we would like to emphasize that this is obtained by writing *only a few lines of high level code on a centralized abstraction*; while the behavior is obtained through automatically generated distributed control programs. Specifically, up to 80.4% utility gain can be achieved in network scenario 3, which has the highest interference. In the case of no cross-layer control, i.e., only one protocol layer is optimized, WNOS still achieves significant utility gain, which varies from 4.5% to 52.2% in the tested instances.

**Modifying Network Behavior**. In the following experiments, we showcase WNOS's capability of modifying the global network behavior by changing a few lines of code. To achieve different desired network behaviors, one only needs to change the centralized and abstract control objective or modify the constraints while WNOS generates the corresponding distributed control programs automatically. For example, if the control objective is to maximize the sum throughput (i.e., maximize $\sum x$) of all sessions instead of sum log throughput (i.e., maximize $\sum \log(x)$) as in Control Problem 1 (Control Program 2), this can be accomplished by rewriting one line of code only: *expr = mkexpr('sum(wos_x)', 'wos_x')*. As shown in Fig. 9 (a, top), compared with Control Program 1 (i.e., maximizing sum-log-throughput), Control Program 2 obtains higher sum throughput (4.92 vs 4.66 in $\mathrm{packets/s}$) by increasing the throughput of session 1 while decreasing the throughput of

session 2, in this way, as expected, trading throughput for fairness. This is because it is easier for session 1 (see scenario 1 in Fig. 7) to achieve higher throughput than session 2 since session 1 has shorter links.

Furthermore, if the network user needs to limit the maximum transmit power of the first session (Control Program 3), this can be accomplished simply by defining a new constraint using the following two lines of code:

*nt.make_var('wos_z', [ntses, seslnk, lkpwr], [1, all, None])*

*nt.add_cstr('wos_z < 5', 'wos_z')*

where the first line of code defines link power as a variable while the second line specifies the upper bound constraint. The resulting session behaviors are shown in Fig. 9 (a, top), where the throughput of session 1 has been effectively bounded. In another example, three sessions are deployed as in scenario 4 in Fig. 7. The normalized transmission power of sessions 2 and 3 are programmed to be smaller than 6 and greater than 20, respectively (Control Program 4). It can be seen in Fig. 9 (a, bottom) that, compared with Control Program 2, the throughput of sessions 2 and 3 can be successfully changed with the new control program. As shown above, all this needs only two new lines of code to characterize the behavior of session 3.

**Flexibility and Scalability**. We further test the flexibility and scalability of WNOS in changing control programs on a large-scale SDR testbed of 21 USRPs (i.e., Scenario 5) and by
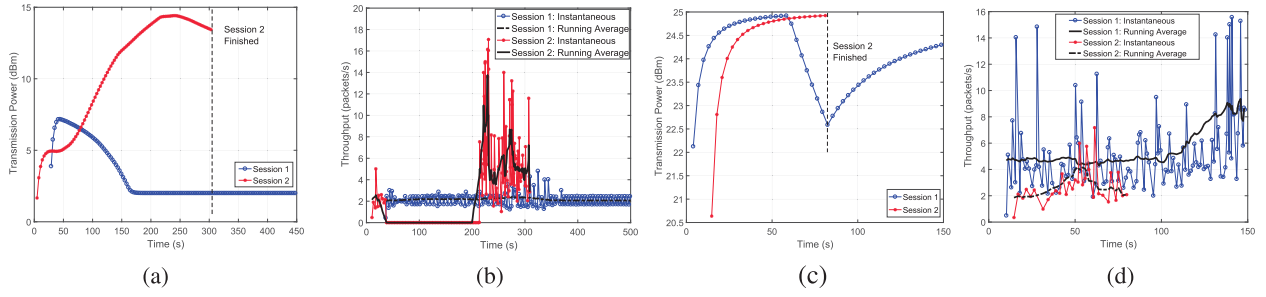
Fig. 10. Instance of (a) transmission power (source node) and (b) throughput resulting from power minimization; Instance of (c) transmission power (source node) and (d) throughput resulting from sum-log-rate maximization.
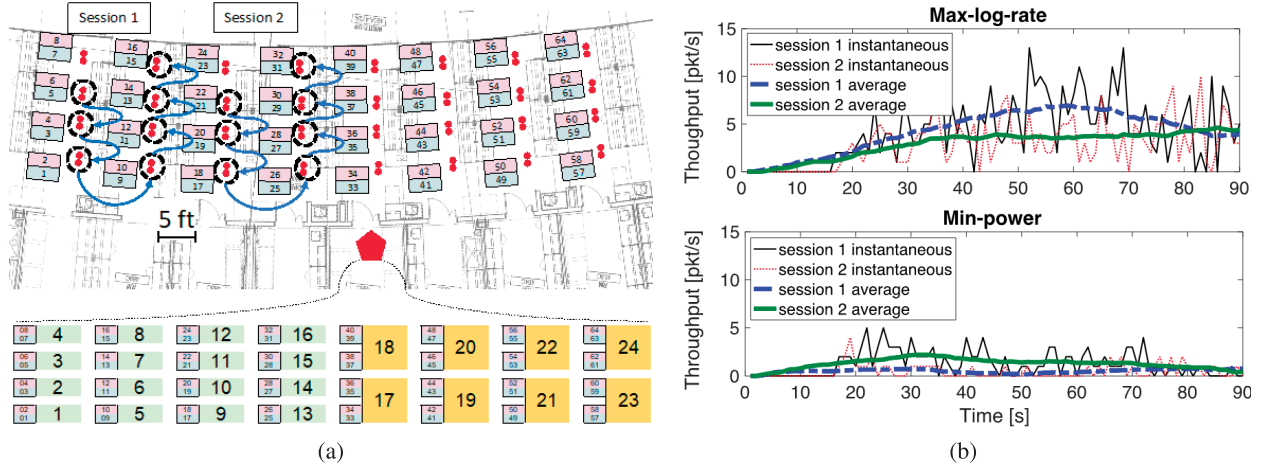


Fig. 11. Experimental evaluation of WNOS Instantiation on ARENA. (a) WNOS deployment on ARENA; (b) Experimental Results.

considering two sharply different network control objectives: i) sum-log-rate maximization and ii) power minimization under minimum rate constraints. Again, changing the network control behaviors based on WNOS requires modifying a couple of lines of code only. The *WiNAR* code for defining the power minimization control objective is as follows:

*nt.make_var('wos_x', [ntlk, lkpwr], [all, None]);*

*expr = mkexpr('sum(wos_x)', 'wos_x'),*

where the first line states the transmission power of all the active links in the network as control variables, while the second line defines the *sum of the transmission power* as the utility function to be minimized.

The measured average transmission power of the source and intermediate nodes are plotted in Fig. 9(b), while the achievable throughput is reported in Fig. 9(c). Unsurprisingly, the two control objectives result in different network behaviors. With power minimization, the three sessions achieved approximately the target throughput ( $\text{packets/s}$ ) with much lower average power than sum-log-rate maximization; while the latter achieves higher throughput at the cost of higher power consumption.

Figure 10 provides a closer look at the contrasting network behaviors resulting from the two control objectives, respectively, by plotting the interactions between sessions 1 and 2 in terms of transmission power and the corresponding

instantaneous throughputs. It can be seen from Fig. 10(b) that session 2's running average throughput decreases to zero during $20 - 200s$ because of low SINR. In response, as shown in Fig. 10(a), session 2 increases its transmission power while session 1 decreases until session 2 recovers at around 200s. After session 2 is finished, session 1 keeps its current transmission power, which is sufficient to achieve the target throughput. Very differently, in the case of sum-log-rate maximization, after session 2 is done, session 1 increases its transmission power to maximize the throughput, as shown in Figs. 10(c) and (d).

Finally, we further verify the flexibility of WNOS by implementing a 14-node WNOS prototype on a new experimentation platform called Arena [52]. Arena is a wireless testing platform at Northeastern University based on a grid of antennas mounted on the ceiling of a large office-space environment. As shown in Fig. 11(a), in this experiment two source nodes intend to deliver data to two destinations through 12 relay nodes, in a wireless multi-hop fashion. As source, relay, and destination nodes, we use SDRs 3-11, SDRs 1-2-5-6-7-9-10-13-14-15, and SDRs 8-16, respectively. We employ WNOS to dictate two different network behaviors, namely max-rate and min-power. The network performance for the two traffic sessions under the two different control problems is shown in Fig. 11(b). It can be seen that different network behaviors have been achieved on the new experimentation platform, by defining the network behaviors with WNOS by

modifying only several lines of the code. This further verifies the flexibility and scalability of WNOS.

## VII. LIMITATIONS AND FUTURE WORK

We believe that our work on WNOS provides the first proof of concept of the ability to create a principled optimization-based wireless network operating system, where the desired global network behavior is defined on a centralized high-level abstraction of the network and obtained through automatically generated distributed cross-layer control programs. We acknowledge several limitations, which will be addressed in future work.

First, WNOS generates cross-layer distributed control programs by decomposing high-level defined network control objective problems, and hence users don't have to deal with tedious details of lower-layer protocols and distributed optimization theory. The decomposition requires the WNOS to specify mathematical models for network protocols at all layers. We are working to standardize the interface of WNOS and plan to make the source code of WNOS available so that new protocols and mathematical models can be easily incorporated into the existing programmable protocol stack (PPS). While the current version of the PPS is designed for software defined radios, we also plan to develop versions of the PPS designed to operate on legacy wireless interface cards (e.g., WiFi). Last, we also plan to extend WNOS to build mathematical models for user-defined network control problems by online learning and automated modeling [23].

Second, network protocols at different layers operate at different time scales, which can be up to orders-of-magnitude different. In the current WNOS implementation, static time scales have been considered, e.g., 30 times larger for transport-layer rate adaptation than physical-layer power control in the testbed evaluation in Section VI. In the future, we will work to let WNOS determine time scales automatically for different protocols based on the user-defined high-level network control objective, including convergence and delay requirements, network size, as well as the underlying transmission medium.

Third, given user-defined high-level network control problems, mathematical optimization problems are constructed and then decomposed by WNOS. Currently, WNOS considers dual decomposition and decomposition by partial linearization (DPL) for cross-layer and distributed decompositions, respectively. We plan to incorporate other optimization and decomposition approaches in the future generation of WNOS, such as primal decomposition, hybrid dual and primal decompositions [24], parallel and distributed successive convex approximation [25], distributed stochastic optimization [26], distributed optimization and statistical learning [27]. Additionally, WNOS in its current version considers devices cooperating within a network utility maximization framework, and in future work non-cooperative devices will be modeled in WNOS through non-cooperative game theory notions [28].

Finally, the objective of WNOS is to generate distributed cross-layer optimized network control programs in an automated way based on available optimization and decomposition theories. A promising research direction is to evaluate the optimality of the resulting programs by comparing them with those methods hand-designed for specific network control problems through extensive testbed experiments.

## VIII. RELATED WORK

Software-defined networking has shown great potential to enhance the performance of wireless access networks, e.g., improving network resource utilization efficiency, simplifying network management, reducing operating costs, and promoting innovation and evolution [4]–[6], [29]. For example, in [4] Bansal *et al.* proposed OpenRadio, which provides a programmable wireless network data plane to enable users/controllers to upgrade and optimize the network in a software-defined fashion. Gudipati *et al.* presented SoftRAN [5] to redesign the radio access layer of LTE cellular networks. In [6], Li *et al.* presented CellSDN to simplify the design and management of wireless cellular networks while enabling new applications. Chen *et al.* propose a software-defined networking based flow scheduling system for integrated LTE-WiFi network. Readers are referred to [30]–[32] and references therein for excellent surveys of this field.

Compared to SDN-based cellular networks, enabling SDN in distributed wireless networks, e.g., multi-hop ad hoc networks and vehicular networks, is significantly more challenging because of the absence of a centralized control entity. Research efforts in this field include [3], [33]–[49]. For example, in [3] the authors propose new software-defined network architecture that can eliminate the need of multi-hop flooding for route discovery in large scale wireless distributed networks (WDNs). In [33], Zhu *et al.* proposed an SDN-based routing scheme for Vehicular Ad Hoc Network (VANET) where a central controller collects network information from switches and computes the optimal routing strategies. Palazzo *et al.* propose SDN-WISE [34], [36] to provide a stateful programmable protocol stack for wireless sensor networks (WSNs). Cheng *et al.* focus on distributed control plane in software-defined networks [40] and study how to migrate switches to achieve a balanced control plane. In [41], the authors propose LayBack, a unifying Software Defined Network (SDN) orchestrator for sharing backhaul network resources across different operators and wireless platforms. The authors of [48] propose a Levenberg-Marquardt algorithm for traffic load minimization in software defined wireless sensor networks. Please refer to [50], [51] and references therein for a good survey of the main results this field. Different from these existing work, where the network control algorithms and protocols are still hand-designed, our objective is to study the basic principles for designing WNOS, an optimization-based wireless network operating system with automated generation of distributed and cross-layer optimized network control programs.

## IX. CONCLUSION

We discussed the basic building principles of the *Wireless Network Operating System (WNOS)*. WNOS provides network designers with an abstraction hiding the lower-level details of the network operations. Based on this abstract representation, WNOS takes centralized network control programs written on a centralized, high-level view of the network and automatically

generates distributed cross-layer control programs based on distributed optimization theory that are executed by each individual node on an abstract representation of the radio hardware. We presented the design architecture of WNOS, discussed the techniques to enable automated decomposition of user-defined centralized network control problems. We have also prototyped WNOS and evaluated its effectiveness using testbed results. Future research directions will include automated modeling, multi-timescale control, incorporating heterogeneous decomposition approaches, and thorough experimental comparison between automatically generated network control programs and those hand-engineered ones.

## REFERENCES

[1] Z. Guan, L. Bertizzolo, E. Demirors, and T. Melodia, "WNOS: An optimization-based wireless network operating system," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Los Angeles, CA, USA, Jun. 2018, pp. 241–250.

[2] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[3] M. Abolhasan, J. Lipman, W. Ni, and B. Hagelstein, "Software-defined wireless networking: Centralized, distributed, or hybrid?" *IEEE Netw.*, vol. 29, no. 4, pp. 32–38, Jul. 2015.

[4] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: A programmable wireless dataplane," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, Helsinki, Finland, Aug. 2012, pp. 109–114.

[5] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software defined radio access network," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, Hong Kong, Aug. 2013, pp. 25–30.

[6] L. E. Li, Z. M. Mao, and J. Rexford. (2012). *CellSDN: Software-Defined Cellular Networks*. [Online]. Available: ftp://ftp.cs.princeton.edu/techreports/2012/922.pdf

[7] Z. Guan, T. Melodia, D. Yuan, and D. A. Pados, "Distributed resource management for cognitive ad hoc networks with cooperative relays," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1675–1689, Jun. 2016.

[8] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[9] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang, "Decomposition by partial linearization: Parallel optimization of multi-agent systems," *IEEE Trans. Signal Process.*, vol. 62, no. 3, pp. 641–656, Feb. 2014.

[10] R. Gupta, L. Vandenberghe, and M. Gerla, "Centralized network utility maximization over aggregate flows," in *Proc. 14th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, Tempe, AZ, USA, May 2016, pp. 1–8.

[11] N. Karakoc, A. Scaglione, and A. Nedic, "Multi-layer decomposition of optimal resource sharing problems," in *Proc. IEEE Conf. Decis. Control (CDC)*, Miami Beach, FL, USA, Dec. 2018, pp. 178–183.

[12] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli, "Wireless MAC processors: Programming MAC protocols on commodity hardware," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 1269–1277.

[13] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, 1st Quart., 2015.

[14] V. K. Balakrishnan, *Graph Theory*. New York, NY, USA: McGraw-Hill, 1997.

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, Mar. 2004.

[16] M. Chiang, "Balancing transport and physical layers in wireless multihop networks: Jointly optimal congestion control and power control," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 104–116, Jan. 2005.

[17] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA, USA: Athena Scientific, 1997.

[18] B. Johansson, P. Soldati, and M. Johansson, "Mathematical decomposition techniques for distributed cross-layer optimization of data networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1535–1547, Aug. 2006.

[19] N. Karakoc, A. Scaglione, A. Nedic, and M. Reisslein, "Multi-layer decomposition of network utility maximization problems," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2077–2091, Oct. 2020.

[20] N. Karakoc and A. Scaglione, "Federated network utility maximization," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Austin, TX, USA, Mar. 2020, pp. 1–6.

[21] J. L. Carter and M. N. Wegman, "Universal classes of hash functions," in *Proc. ACM Symp. Theory Comput.*, Boulder, CO, USA, May 1977, pp. 106–112.

[22] *CogApp 2.5.1*. Accessed: Dec. 2018. [Online]. Available: https://pypi.python.org/pypi/cogapp

[23] A. C. Capelo, L. Ironi, and S. Tentoni, "Automated mathematical modeling from experimental data: An application to material science," *IEEE Trans. Syst., Man Cybern., C, Appl. Rev.*, vol. 28, no. 3, pp. 356–370, Aug. 1998.

[24] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[25] G. Scutari and Y. Sun, "Parallel and distributed successive convex approximation methods for big-data optimization," in *Multi-agent Optimization* (Lecture Notes in Mathematics). New York, NY, USA: Springer-Verlag, Jan. 2018.

[26] S. A. Alghunaim and A. H. Sayed, "Distributed coupled multiagent stochastic optimization," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 175–190, Jan. 2020.

[27] S. Boyd, N. Parikh, E. Chu, B. Peleto, and J. Eckstein, *Distributed Optimization and StatisticalLearning Via the Alternating DirectionMethod of Multipliers*. Boston, MA, USA: NOW, 2010.

[28] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory* (Classics in Applied Mathematics). Philadelphia, PA, USA: SIAM, 1999.

[29] K. Chen, J. Liu, J. Martin, K.-C. Wang, and H. Hu, "Improving integrated LTE-WiFi network performance with SDN based flow scheduling," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Hangzhou, China, Jul. 2018, pp. 1–9.

[30] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2713–2737, 4th Quart., 2016.

[31] K. Sood, S. Yu, and Y. Xiang, "Software-defined wireless networking opportunities and challenges for Internet-of-Things: A review," *IEEE Internet Things J.*, vol. 3, no. 4, pp. 453–463, Aug. 2016.

[32] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3259–3306, 4th Quart., 2018.

[33] M. Zhu, J. Cao, D. Pang, Z. He, and M. Xu, "SDN-based routing for efficient message propagation in VANET," in *Wireless Algorithms, Systems, and Applications*, K. Xu and H. Zhu, Eds. New York, NY, USA: Springer, 2015, pp. 788–797.

[34] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Proc. IEEE INFOCOM*, Hong Kong, Apr. 2015, pp. 513–521.

[35] M. Alqallaf and B. Wang, "Software defined collaborative secure ad hoc wireless networks," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, Atlanta, GA, USA, Jun. 2015, pp. 196–203.

[36] P. D. Dio, S. Faraci, L. Galluccio, and S. Milardo, "Exploiting state information to support QoS in software-defined WSNs," in *Proc. Med-Hoc-Net*, Vilanova i la Geltru, Spain, Jun. 2016, pp. 1–7.

[37] T. Miyazaki *et al.*, "A software defined wireless sensor network," in *Proc. ICNC*, Honolulu, HI, USA, Feb. 2014, pp. 847–852.

[38] D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu, and Y. Xiang, "Energy minimization in multi-task software-defined sensor networks," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3128–3139, Nov. 2015.

[39] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in *Proc. ACM Conf. Special Interest Group Data Commun.*, London, U.K., Aug. 2015, pp. 43–56.

[40] G. Cheng, H. Chen, Z. Wang, and S. Chen, "DHA: Distributed decisions on the switch migration toward a scalable SDN control plane," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, Toulouse, France, May 2015, pp. 1–9.

[41] L. Ferrari, N. Karakoc, A. Scaglione, M. Reisslein, and A. Thyagaturu, "Layered cooperative resource sharing at a wireless SDN backhaul," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[42] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Fragmentation-based distributed control system for software-defined wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 901–910, Feb. 2019.

[43] K. S. Kalupahana Liyanage, M. Ma, and P. H. Joo Chong, "Controller placement optimization in hierarchical distributed software defined vehicular networks," *Comput. Netw.*, vol. 135, pp. 226–239, Apr. 2018.

[44] G. Li, S. Guo, Y. Yang, and Y. Yang, "Traffic load minimization in software defined wireless sensor networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1370–1378, Jun. 2018.

[45] A.-C. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SD-WISE: A software-defined WIreless SEnsor network," *Comput. Netw.*, vol. 159, pp. 84–95, Aug. 2019.

[46] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Trans. Emerg. Topics Comput.*, early access, Feb. 14, 2019, doi: 10.1109/TETC.2019.2899407.

[47] C. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, "Deep Q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5871–5883, Jun. 2019.

[48] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[49] X. Huang, R. Yu, J. Kang, Z. Xia, and Y. Zhang, "Software defined networking for energy harvesting Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1389–1399, Jun. 2018.

[50] M. A. Hassan, Q.-T. Vien, and M. Aiash, "Software defined networking for wireless sensor networks: A survey," *Adv. Wireless Commun. Netw.*, vol. 3, pp. 10–22, May 2017.

[51] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 333–354, 1st Quart., 2018.

[52] L. Bertizzolo *et al.*, "Arena: A 64-antenna SDR-based ceiling grid testing platform for sub-6 GHz 5G-and-Beyond radio spectrum research," *Comput. Netw.*, vol. 181, pp. 1–17, 2020.

**Zhangyu Guan** (Senior Member, IEEE) received the Ph.D. degree in communication and information systems from Shandong University, China, in 2010. He is currently an Assistant Professor with the Department of Electrical Engineering, State University of New York at Buffalo, where he also directs the Wireless Intelligent Networking and Security (WINGS) Laboratory, with research interests in network design automation, new spectrum technologies, and wireless network security. He has served as the TPC Chair for IEEE INFOCOM Workshop on Wireless Communications and Networking in Extreme Environments (WCNEE) 2020, the Student Travel Grants Chair for IEEE Sensor, Mesh, and Ad Hoc Communications and Networks (SECON) from 2019 to 2020, and the Information System (EDAS) Chair for IEEE Consumer Communications Networking Conference (CCNC) since 2021. He has served as an Area Editor for *Journal of Computer Networks* (Elsevier) since July 2019. He has also served as a TPC Member for IEEE INFOCOM from 2016 to 2021, IEEE GLOBECOM from 2015 to 2021, IEEE IPCCC from 2015 to 2020, and among others.

**Lorenzo Bertizzolo** received the B.S. and M.S. degrees (Hons.) in computer and communication networks engineering from the Politecnico di Torino, Italy, in 2014 and 2015, respectively. He is currently pursuing the Ph.D. degree in computer engineering with Northeastern University.

He is also a Research Assistant with the Institute for the Wireless Internet of Things, Northeastern University. His research interests include 5G, software-defined networking for wireless networks, distributed optimization, and unmanned aerial networks. He is also a Collaborator of the AT&T Labs Research, working on the integration of Unmanned Aerial System into the next generations' cellular networks. He is a Student Member of the ACM.

**Emrecan Demirors** (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Northeastern University in 2017. He is currently a Research Assistant Professor with the Department of Electrical and Computer Engineering, Northeastern University, where he is also a member of the Institute for the Wireless Internet of Things. His current research interests include the Internet of Things and 5G networks, underwater communications and networks, Internet of Medical Things, unmanned aerial, and underwater vehicle networking. He is an Associate Editor of IEEE ACCESS. He organized the IEEE International Workshop on Wireless Communications and Networking in Extreme Environments from 2017 to 2020. He has also been serving as a TPC Member for IEEE WCNC since 2018, IEEE PIMRC since 2020, and among others. He is the Co-Founder and the Research and Development Director with Bionet Sonar.

**Tommaso Melodia** (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology in 2007. He is currently the William Lincoln Smith Professor with the Department of Electrical and Computer Engineering, Northeastern University. He is also the Director of the Institute for the Wireless Internet of Things, and the Director of Research of the PAWR Project Office, a public-private partnership that is developing four city-scale platforms for advanced wireless research in the USA. His research interests include modeling, optimization, and experimental evaluation of wireless networked systems, with applications to 5G networks and the Internet of Things, software-defined networking, and body area networks. His research work was supported mostly by the U.S. federal agencies, including the National Science Foundation, the Air Force Research Laboratory, the Office of Naval Research, the Army Research Laboratory, and DARPA. He is the Editor-in-Chief of *Computer Networks*, and a former Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON MULTIMEDIA, and among others. He is a Senior Member of the ACM.