

WNOS: An Optimization-based Wireless Network Operating System

Zhangyu Guan, Lorenzo Bertizzolo, Emrehan Demirors and Tommaso Melodia
Department of Electrical and Computer Engineering
Northeastern University, MA 02115, USA
{zguan,bertizzolo,edemirors,melodia}@ece.neu.edu

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.

CCS CONCEPTS

• **Networks** → *Network architectures* → *Network design principles*;

KEYWORDS

Software-defined networking, automated program generation, distributed control, wireless network optimization.

ACM Reference Format:

Zhangyu Guan, Lorenzo Bertizzolo, Emrehan Demirors and Tommaso Melodia. 2018. WNOS: An Optimization-based Wireless Network Operating System. In *Mobihoc'18: The Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, June 26–29, 2018, Los Angeles, CA, USA*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3209582.3209607>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Mobihoc'18, June 26–29, 2018, Los Angeles, CA, USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5770-8/18/06...\$15.00

<https://doi.org/10.1145/3209582.3209607>

1 INTRODUCTION

Most 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 [1–3]. 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 [1, 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*. 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 interface, e.g., OpenFlow [1], 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

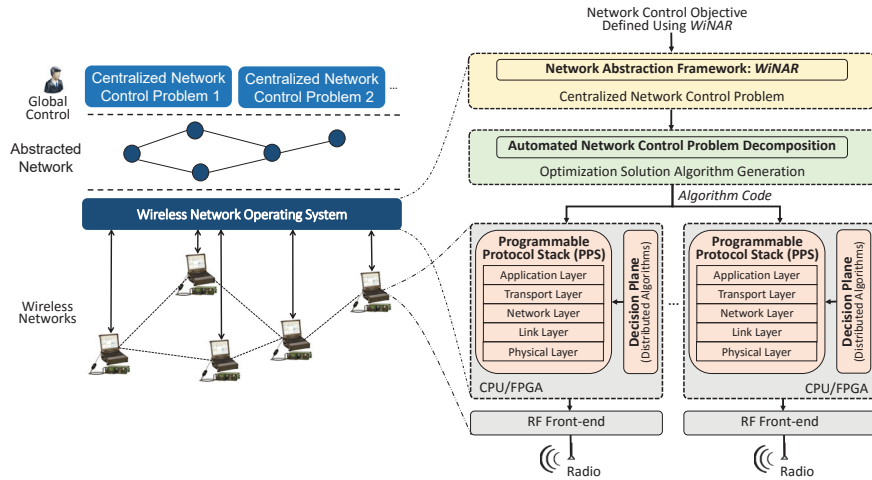


Figure 1: Architecture of the wireless network operating system.

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 decisions at all layers to maximize the network performance while keeping the network scalable, reliable, and easy to deploy [7–10]. 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 wireless network abstraction framework *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 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.

The remainder of the paper is organized as follows. In Section 2, we present the design architecture of WNOS, and then describe the network abstraction framework *WiNAR* in Section 3. We discuss the automated network control problem decomposition approach in Section 4, and present the prototyping and experimental evaluation of WNOS in Section 5. We discuss limitations and future work in Section 6, and finally we draw the main conclusions in Section 7.

2 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 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. 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 3.

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 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 [11, 12].

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 by

updating and passing a common set of optimization variables. See Section 4 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. This is executed in real time and 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. 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 [13, 14]. 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.

3 NETWORK ABSTRACTION: WINAR

The objective of the network abstraction 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.

Challenge 1: Pre-deployment network abstraction. Unlike traditional network abstraction and resource virtualization [15, 16], 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 or 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.

Challenge 2: 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, i.e., network representation, control interface and control problem definition.

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 3.1 (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} :

- $|\{\text{Network entities represented by } \mathcal{A}\}| = 1$ 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.¹

Definition 3.2 (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

- $|\{\text{Network entities represented by } \mathcal{V}\}| \geq 1$, 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 3.3*.

Definition 3.3 (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

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

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 3.3*, the inter-dependency $I(\cdot, \cdot)$ among different network elements can be characterized as a directed multigraph [17]. 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 two directed edges (hence a multigraph [17]): *Node* has an attribute *Neighbors of Node*, each member of which is a *Node*.

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: (i) *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*; (ii) *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*; (iii) *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*); and (iv) *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 (readers are referred to [18] for examples of network control problem definition). A network control problem comprises of the following four components:

- *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.

4 AUTOMATED 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 [11, 12, 19] 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. Next, we first provide a brief review of the decomposition theory based on which our automated decomposition approach is designed.

¹Here, *Link Capacity* and *Session Rate* refer to the network parameters rather than any specific values of the parameters that can be time varying.

4.1 Decomposition Approaches

Cross-layer Decomposition. In this paper we consider duality theory for cross-layer decomposition (while the automated decomposition approach in Section 4.2 is not limited to any specific decomposition theory). Consider a network control problem expressed as

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

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

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

where $L(x, \lambda)$ is called the Lagrangian of problem (2) [20]. 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

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

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

and then iteratively maximizing f_{sub_1} over control variables x with given λ and updating λ with the minimizer of f_{sub_2} .

Distributed Decomposition. 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, 8, 21], 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 $\tilde{U}_i(\mathbf{x}_i, \mathbf{x}_{-i})$ for each agent $i \in \mathcal{I}$, expressed as $\tilde{U}_i(\mathbf{x}_i, \mathbf{x}_{-i}) = \Theta_i(U(\mathbf{x})) + \Gamma_i(\mathbf{x})$, 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 $\tilde{U}_i(\mathbf{x}_i, \mathbf{x}_{-i})$ 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, 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 [12].

4.2 Automated Decomposition

A key step in cross-layer decomposition, as discussed in Section 4.1, 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 3.

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 λ_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*

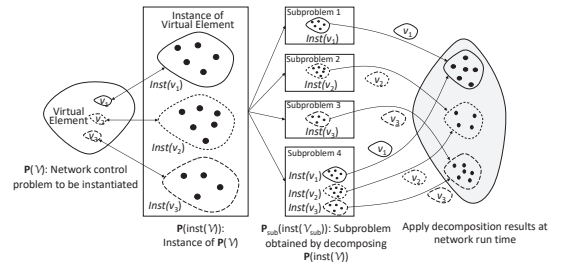


Figure 2: Basic principle of network control problem decomposition based on disciplined instantiation (DI).

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 certain rules (as discussed below), a specific instance of the user-defined abstract network control problem, such that the abstract problem can be decomposed by decomposing the specific instance and the obtained decomposition results can be applied to those control problems at network run time.

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 $P(\mathcal{V})$ as the network control problem to be instantiated, and let $P(inst(\mathcal{V}))$ represent the specific instantiated problem obtained by instantiating $P(\mathcal{V})$. Then, $P(inst(\mathcal{V}))$ can be decomposed into a set of subproblems $P_{sub}(inst(\mathcal{V}_{sub}))$ each involving only a subset \mathcal{V}_{sub} of the virtual elements with $\mathcal{V}_{sub} \subset \mathcal{V}$. For example, in Fig. 2, $P(inst(\mathcal{V}))$ 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 $P(inst(\mathcal{V}))$ are also applicable to the original problem $P(\mathcal{V})$ 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 3. 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 3 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^{glb} is first instantiated using a set of pre-determined number N^{glb} of elements, i.e., $|inst(v^{glb})| = N^{glb}$ with $inst(v^{glb})$ being the instance of the global virtual element v^{glb} and $|inst(v^{glb})|$ being the cardinality of $inst(v^{glb})$. The resulting instance $inst(v^{glb})$ will be used to serve as the mother set to generate instances for those local virtual elements v^{lcl} .

Then, each local virtual element v^{lcl} can be instantiated by randomly selecting a subset of members from the mother set $inst(v^{glb})$, i.e., the instance of the global virtual element v^{glb} . Denote the resulting subset instance as $inst(v^{lcl})$, then we have $|inst(v^{lcl})| = N^{lcl}$ and $inst(v^{lcl}) \subset inst(v^{glb})$, where N^{lcl} is the cardinality of instances for local virtual elements.

Rule 2: Ordered Uniqueness. With this rule, a unique instance will be generated for each local virtual element v^{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):

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

where the members of $inst(v^{lcl})$, i.e., the instance for local virtual element v^{lcl} , are first sorted, and then a unique id hid^{lcl} is calculated for the sorted instance $inst'(v^{lcl})$ 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. 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^{lcl} , with each member in the set representing a primitive element (see Section 3 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 $P_{sub}(inst(\mathcal{V}_{sub}))$ the members of each instance may be re-ordered by the mathematical manipulations decomposing the instantiated network control problem $P(inst(\mathcal{V}))$, e.g., forming and

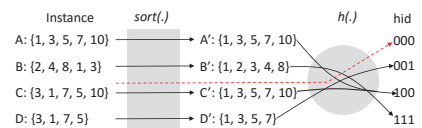


Figure 3: Illustration of hash mapping.

decomposing the dual function in (3), (4) and (5). In (6) function $\text{sort}(\cdot)$ 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 $\mathbf{P}_{\text{sub}}(\text{inst}(\mathcal{V}_{\text{sub}}))$. 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^{loc} is *identical* to any existing instances, i.e., they have the same id, another instance will be created v^{loc} 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.

Decomposition. 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 4.1. 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).

4.3 Toy Example of DI-based Decomposition

Consider the following cross-layer network control problem:

$$\begin{aligned} & \text{maximize} && \sum_{s \in \mathcal{S}} R_s \\ & \text{subject to} && \sum_{s \in \mathcal{S}_l} R_s \leq C_l(\mathbf{\Pi}), \forall l \in \mathcal{L} \end{aligned} \quad (7)$$

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 can be found in [18].

Instantiation. As defined in Section 4.2, \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 $N^{\text{glb}} = 3$ for global virtual elements \mathcal{S} and \mathcal{L} and $N^{\text{loc}} = 2$ 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

$$\begin{aligned} & \text{maximize} && R_1 + R_2 + R_3 \\ & \text{subject to} && R_1 + R_2 \leq C_1(\mathbf{\Pi}) \\ & && R_1 + R_3 \leq C_2(\mathbf{\Pi}) \\ & && R_2 + R_3 \leq C_3(\mathbf{\Pi}) \end{aligned} \quad (8)$$

Decomposition. Consider dual decomposition as discussed in Section 4.1, then the dual function of (8) can be written as

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

where λ_1, λ_2 and λ_3 are dual coefficients. By decomposing (9), problem (8) can be decomposed into two single-layer problems:

$$\begin{aligned} \text{Transport Layer} : & \begin{cases} \text{maximize } R_1 - \lambda_1 R_1 - \lambda_2 R_1, \text{ for } s = 1 \\ \text{maximize } R_2 - \lambda_1 R_2 - \lambda_3 R_2, \text{ for } s = 2 \\ \text{maximize } R_3 - \lambda_2 R_3 - \lambda_3 R_3, \text{ for } s = 3 \end{cases} \\ \text{Physical Layer} : & \text{maximize } \lambda_1 C_1(\mathbf{\Pi}) + \lambda_2 C_2(\mathbf{\Pi}) + \lambda_3 C_3(\mathbf{\Pi}), \end{aligned} \quad (11)$$

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 $\mathbf{\Pi}$.

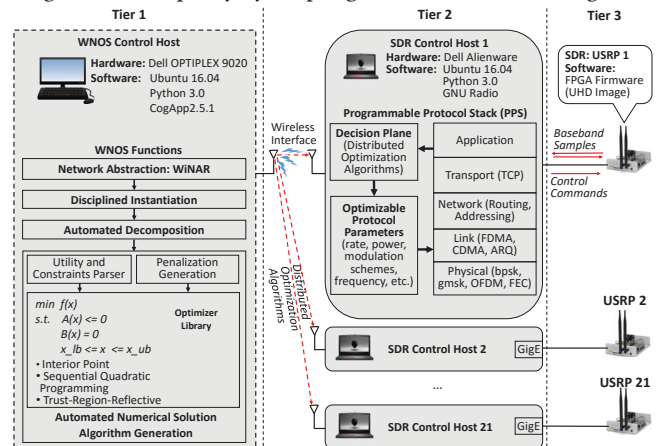


Figure 4: Prototyping diagram of WNOS.

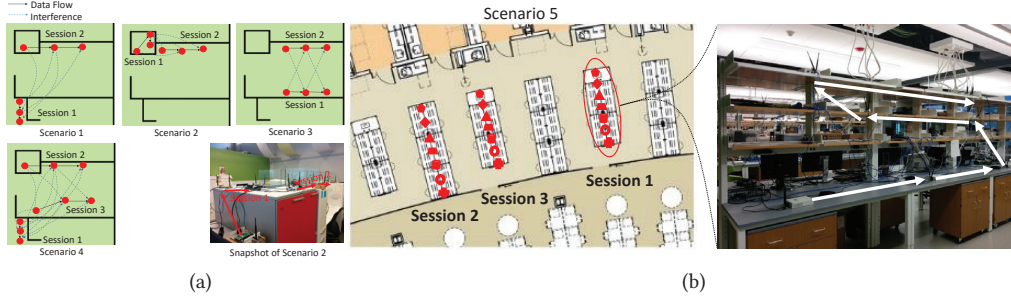


Figure 5: Experimental Scenarios: (a) Scenarios 1-4 and (b) Scenario 5.

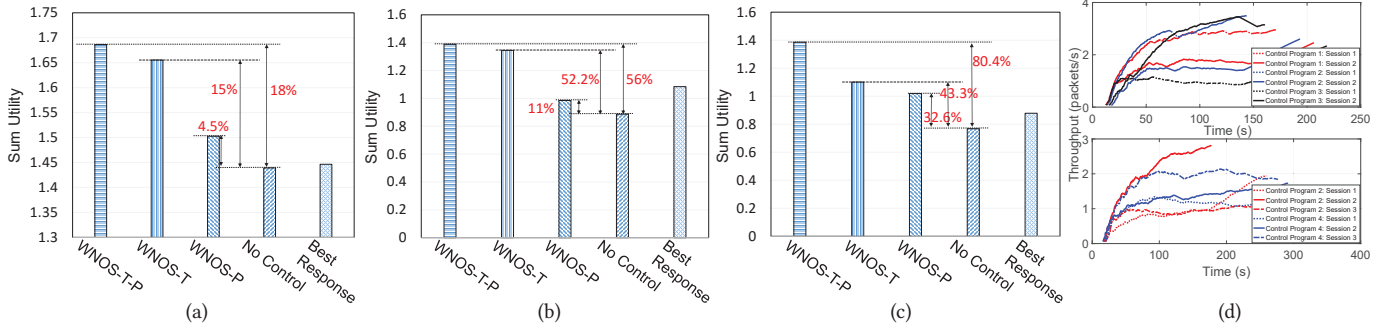


Figure 6: Average sum utility of scenarios (a) 1, (b) 2 and (c) 3; (d) Network behaviors with different control programs.

Application to Run-time. 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 identify 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.

5 EXPERIMENTAL EVALUATION

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. 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 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.

SDR Implementation. We test WNOS on the designed SDR testbed in five different networking scenarios. As shown in Fig. 5,

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 800 kHz. Reed-Solomon (RS) code is used for forward error correction (FEC) with coding rate ranging from 0.1 to 0.4 at a step of 0.1. 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: (i) **Effectiveness:** Through experiments in Scenarios 1-3, we show that WNOS-based network optimization outperforms non-optimal or purely locally optimal (greedy) network control; (ii) **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; and (iii) **Scalability:** In Scenario 5 we show the scalability of WNOS by deploying code over a large-scale network.

Effectiveness. Five schemes have been tested: (i) WNOS-T-P: transport and physical layers are jointly controlled 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) neither transport or physical layer are controlled by WNOS; and (v) Best Response: maximum rate and power are used at the transport and physical layers, respectively.

The average performance of the five schemes is reported in Figs. 6(a), (b) and (c) 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

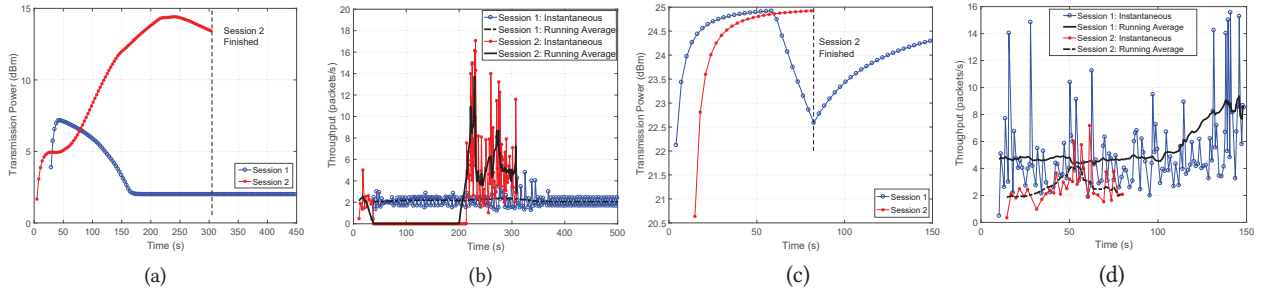


Figure 8: Experimental instance for (a) & (b) power minimization and (c) & (d) sum-log-rate maximization.

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 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. 6 (d, 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 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. 5) 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. 6 (d, top), where the throughput of session 1 has been effectively bounded. In another example, three sessions are deployed as in scenario 4 in Fig. 5. The normalized

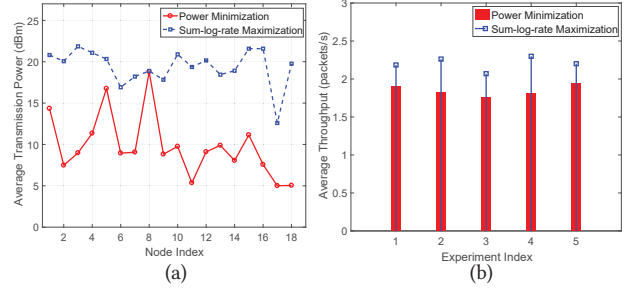


Figure 7: (a) Transmission power and (b) throughput resulting from sum-log-rate maximization and power minimization.

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. 6 (d, 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 considering two sharply different network control objectives: sum-log-rate maximization and power minimization. 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. 7(a), while the achievable throughput is reported in Fig. 7(b). Unsurprisingly, the two control objectives result in different network behaviors. With power minimization, the three sessions achieved approximately the target throughput (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 8 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. 8(b) that session 2's running average throughput decreases to zero during 20 – 200s because of low SINR. In response, 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 Fig. 8(d).

6 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.

Learning-based Automated Modeling. Currently 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 [22].

Multi-timescale Control. 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 5. 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.

Decomposition Approaches. Given user-defined high-level network control problems, mathematical optimization problems are constructed and then decomposed by WNOS. Currently only dual decomposition and decomposition by partial linearization (DPL) have been considered for cross-layer and distributed decompositions, respectively. We plan to incorporate other decomposition approaches, such as primal decomposition, hybrid dual and primal decompositions [19].

7 CONCLUSIONS

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 technologies 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 and incorporating heterogeneous decomposition approaches.

ACKNOWLEDGEMENTS

This work is based upon work supported in part by ONR grants 0014-16-1-2213 and N00014-17-1-2046, ARMY W911NF-17-1-0034, and NSF CNS-1618727.

REFERENCES

- [1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, March 2008.
- [2] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A Roadmap for Traffic Engineering in SDN-OpenFlow Networks," *Computer Network (Elsevier) Journal*, vol. 71, pp. 1–30, October 2014.
- [3] M. Abolhasan, J. Lipman, W. Ni, and B. Hagelstein, "Software-Defined Wireless Networking: Centralized, Distributed, or Hybrid?" *IEEE Network*, vol. 29, no. 4, pp. 32–38, July/August 2015.
- [4] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: A Programmable Wireless Dataplane," in *Proc. of HotSDN*, Helsinki, Finland, August 2012.
- [5] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software Defined Radio Access Network," in *Proc. of HotSDN*, Hong Kong, China, August 2013.
- [6] L. E. Li, Z. M. Mao, and J. Rexford, "CellSDN: Software-Defined Cellular Networks," *Technical Report*, 2012. Online Available. <ftp://ftp.cs.princeton.edu/techreports/2012/922.pdf>.
- [7] M. Chiang, "Balancing Transport and Physical Layers in Wireless Multihop Networks: Jointly Optimal Congestion Control and Power Control," *IEEE JSAC*, vol. 23, no. 1, pp. 104–116, January 2005.
- [8] Z. Guan, T. Melodia, D. Yuan, and D. Pados, "Distributed Resource Management for Cognitive Ad Hoc Networks with Cooperative Relays," *IEEE/ACM Trans. on Netw.*, vol. 24, no. 3, pp. 1675–1689, June 2016.
- [9] Z. Guan, T. Melodia, and G. Scutari, "To Transmit or Not to Transmit? Distributed Queueing Games for Infrastructureless Wireless Networks," *IEEE/ACM Trans. on Netw.*, vol. 24, no. 3, pp. 1675–1689, June 2016.
- [10] Z. Guan, N. Cen, T. Melodia, and S. Pudlewski, "Self-Organizing Flying Drones with Massive MIMO Networking," in *Proc. of Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, Capri, Italy, June 2018.
- [11] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures," *Proc. of IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.
- [12] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang, "Decomposition by Partial Linearization: Parallel Optimization of Multiuser Systems," *IEEE Trans. on Signal Process.*, vol. 63, no. 3, pp. 641–656, Feb. 2014.
- [13] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "Openradio: A Programmable Wireless Dataplane," in *Proc. of HotSDN*, Helsinki, Finland, August 2012.
- [14] E. Demirors, G. Sklivanitis, T. Melodia, and S. Batalama, "RcUBe: Real-Time Reconfigurable Radio Framework with Self-Optimization Capabilities," in *Proc. of IEEE SECON*, Seattle, WA, June 2015.
- [15] 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, First Quarter 2015.
- [16] A. Belbakkouche, M. M. Hasan, and A. Karmouch, "Resource Discovery and Allocation in Network Virtualization," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1114–1127, Fourth Quarter 2012.
- [17] V. K. Balakrishnan, *Graph Theory*. USA: McGraw-Hill, 1997.
- [18] http://www.ece.neu.edu/wineslab/wnos/wnos_arXiv2017.pdf.
- [19] D. P. Palomar and M. Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE JSAC*, vol. 24, no. 8, pp. 1439–1451, August 2006.
- [20] S. Boyd and L. Vandenberghe, *Convex Optimization*. USA: Cambridge University Press, March 2004.
- [21] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks," in *Proc. of IEEE INFOCOM*, Barcelona, Spain, April 2006.
- [22] A. C. Capelo, L. Ironi, and S. Tentoni, "Automated Mathematical Modeling from Experimental Data: An Application to Material Science," *IEEE Trans. on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 28, no. 3, pp. 356–370, Aug. 1998.