IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 31, 2021, accepted February 4, 2021, date of publication February 8, 2021, date of current version February 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3057690

DSF: A Distributed SDN Control Plane Framework for the East/West Interface

BASEM ALMADANI, ABDURRAHMAN BEG, AND ASHRAF MAHMOUD, (Member, IEEE)
Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Corresponding author: Abdurrahman Beg (g201703830@kfupm.edu.sa)

This work was supported by the Real-Time Embedded Systems Lab, Department of Computer Engineering, and King Fahd University of Petroleum and Minerals.

ABSTRACT The ever-increasing number of network-capable devices places a massive burden on modern networks. Communication infrastructure should provide quality-of-service essentials in terms of high-bandwidth capacity, scalability, resiliency, and security. Programmable networks are viewed as the prevailing method of encountering the challenges introduced by the accelerated expansion. The ability of software-defined networking (SDN) to separate the control plane from the data plane and enable the programmability of the network creates new ways to architect the network. The centralization of control logic introduces complexities in large-scale, distributed networks such as performance bottlenecks and reliability. Distributed SDN controllers have been proposed to overcome the performance concerns. The lack of a communication standard among distributed controllers, referred to as the East/West interface, presents a challenge in the adoption of SDN in large-scale, distributed networks. In this paper, we propose Distributed SDN control plane Framework (DSF) - a framework for the East/West interface for heterogeneous, distributed SDN controllers to synchronize topologies using a standardized data-centric real-time publish/subscribe paradigm known as the Data Distribution Service (DDS). Distributed control plane architectures are proposed using DSF: flat, hierarchical, and T-model. The DSF interface is implemented on multiple SDN control plane platforms to evaluate performance: Floodlight and Open Network Operating System (ONOS) controllers. Test cases with different configurations are designed for performance evaluation of the proposed interface in homogeneous and heterogeneous SDN control planes. In addition, a performance comparison is presented of DSF-based ONOS controllers versus Atomix-based ONOS cluster solutions.

INDEX TERMS SDN, distributed controllers, East/West interface, RTPS, ONOS.

I. INTRODUCTION

Present-day and future networks face an ever-increasing demand of providing fast and reliable interconnectivity among a growing number of network capable devices, estimated to reach 29.3 billion by the year 2023 [1]. Enterprise and Data Center Networks (DCNs) further the concern by necessitating Quality-of-Service (QoS) essentials in terms of high-bandwidth capacity, scalability, resiliency, and security. The concept of programmable networks is perceived as a method of revolutionizing communication operations and realizing the preceding objectives.

Software-defined networking (SDN) [2] is a programmable network concept designed to overcome the limitations of conventional networks such as configuration and management complexity, lack of scalability, and vendor lock-in. SDN proposes a programmable network control, decoupled from forwarding. The concept basis on two main characteristics: separation of the control and data planes and programmability of the control plane [3]. Control functions are transferred from data forwarding devices to a logically centralized network entity, the controller. The programmability of SDN is derived from the instructions given to the controller via application programming interfaces (APIs). Network-wide traffic forwarding decisions can be set by the controller simplifying network configuration, policy enforcement, and evolution [4].

Controllers are categorized into two classifications: centralized and distributed architectures. The initial kind of controller was physically-centralized and provided a simplified design choice. As networks expanded, the architecture became a focal point for performance bottlenecks and single

The associate editor coordinating the review of this manuscript and approving it for publication was Tawfik Al-Hadhrami.

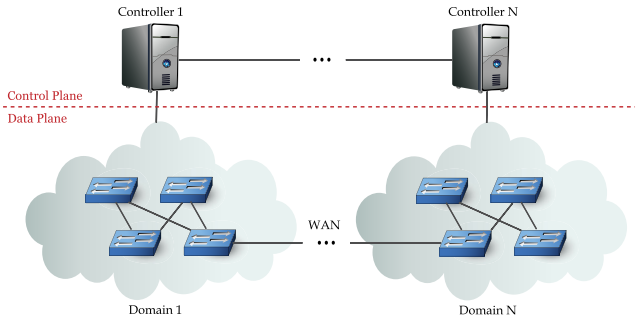


FIGURE 1. SDN divides network functions into control and data planes. Controllers represent the administrator platform of each data plane domain, synchronizing topologies to enable data packets to flow between data plane domains. In this illustration, N number of domains over WAN communication is displayed.

point of failures [5]. Large-scale network deployments, such as DCNs, containing tens of thousands of network elements overwhelmed the single physically-centralized controller [6]. NOX [7], a physically-centralized controller, serves 30K flow requests per second with a latency of 10 ms. Scalability and performance concerns of controllers took center stage of SDN adoptability [8], [9]. Consequently, concurrent physically-distributed controllers were introduced in the form of logically-centralized and logically-distributed control planes.

Logically-centralized control plane comprises several physically-distributed controllers cooperating to resolve a singular, consistent view of the network domain as illustrated by Fig. 1. The architecture requires tight synchronization between controllers. Logically-distributed control planes were designed to expand SDN to support large-scale, distributed networks where a logically-centralized controller governs a domain of forwarding elements within a large global domain network and participates in inter-domain communication with other controllers. The primary use of this category of controllers is in wide area networks (WANs) and globally distributed data centers. For example, the cloud providers and SD-WAN pioneers Microsoft [10] and Google [11] run services from data centers distributed around the world and require efficient inter-data center exchange over complex infrastructure and protocols (e.g., BGP and MPLS) that administer the traffic.

The issue of scalability in the control plane persists throughout the classifications of SDN due to the distributed structure. The method for communication among the distributed entities, known as the East/West interface, is an important problem discussed in the literature; highlighted in particular was the absence of a communication standard among inter-domain SDN entities [12]–[14].

In this paper, an adaptive framework for the East/West interface is proposed, referred to as Distributed SDN controller Framework (DSF). DSF uses a standardized data-centric Real-Time Publish/Subscribe (RTPS) model. Features of DSF compared to previous East/West techniques is presented. A series of experiments are conducted to analyze the performance of the proposed interface in differing SDN controller platforms in homogeneous and heterogeneous networks.

A. PROBLEM STATEMENT

Distributed SDN control planes require tight synchronization of network topology information to maintain a holistic view of the network, enabling route optimization of data packets from endpoint-to-endpoint in large-scale, distributed networks. Techniques of exchanging network state information in the literature vary with a wide range of protocols in the East/West interface. Some methods lack the capability of cross-platform coordination, while others in scalability. Further, previous works primarily demonstrate the network state synchronization in networks with homogeneously distributed control planes. This paper aims to outline and demonstrate the use of a standardized approach in the East/West interface by implementing an adaptive data-centric RTPS standard for large-scale, distributed SDN control planes in homogeneous and heterogeneous networks.

The remainder of the paper is organized as follows: Section II reviews related works. Section III presents the proposed DSF interface. Section IV details the communication model utilized in the proposed framework. Section V describes the implementation of the DSF interface in differing SDN controllers and the experiment test cases. The results of the experiments are discussed in Section VI. Finally, Section VII concludes the paper and describes possible future investigations.

II. RELATED WORKS

East/West interface solutions for distributed SDN controllers fall under two categories: logically-centralized or logically-distributed control planes. A summary of the primary techniques for both categories is provided in this section. Table 1 presents a comparison between previous East/West techniques and the DSF interface proposed in this article. Table 2 compares between publish/subscribe models used by previous East/West techniques and the data-centric RTPS solution utilized by the DSF interface.

A. LOGICALLY-CENTRALIZED CONTROL PLANE

HyperFlow [15] is a distributed event-based control plane for OpenFlow, providing a logically-centralized process implemented as an application for NOX [7], a physically-centralized controller. The application aims to synchronize controllers network-wide views using an event propagation system based on publish/subscribe paradigm. The system utilizes WheelFS [16], a distributed file system designed to enable wide-area storage for distributed applications. Controllers poll file directories, referred to as channels, for changes by subscribing to the data channel, control channel, and the controllers personal channel. The subscribed information consists of state changes in the network used to build and maintain the overall view of the network. A consistent network-wide view in all the controllers is a strict requirement

for HyperFlow. WheelFS is used for scaling distributed controllers. However, distributed file systems are broker-based systems becoming focal points for performance bottlenecks. The use of brokers increases the risk of network performance degradation in case of failure. Brokers in WheelFS are the distributed file storage nodes.

Another approach studied was the use of distributed systems principles as in Onix [17], which is a cluster of one or more physical servers providing one or more Onix instances. Onix function involves maintaining a distributed Network Information Base (NIB), which is a data structure storing network state information. NIB is the core of the Onix control and the basis for the distribution model. Control logic for network behavior is exposed to the network via Onix APIs and state information can be distributed using a transactional persistent database which is replicated to distribute state updates. The method has massive performance limitations suited only for slowly changing network state. Onix offers the use of memory-only distributed hash tables for dynamic networks with high-rates of state changes. Onix, like HyperFlow, is highly dependent on having a consistent, network-wide view in all the controllers. Onix is not suitable for dynamically changing, large-scale networks due to the limitations of replicating the transactional persistent database across all controllers in real-time.

The authors of [18] propose OpenDaylight (ODL), a cluster of cooperating controllers providing network resiliency, reliability, and scalability. ODL builds data structure trees using Yang modeling language and Model-Driven Service Abstraction Layer (MD-SAL). The MD-SAL has four primary tasks: route remote procedural calls between processes, a subscription-based mechanism for notification delivery, route and coordinate data reads, and manage mounts which are instances of MD-SAL. The Akka framework, specified in [19], is used to synchronize information between the cluster of controllers. ODL clustered controllers are limited due to the vast number of control/flow packets generated among larger clusters [20].

Similar to ODL, authors of [21] propose Open Network Operating System (ONOS), an open-source platform maintained by The Linux Foundation. ONOS is a cluster of cooperating distributed controllers designed for scalability, high performance, and high availability. Holistic topology is maintained by a distributed data structure administrated by elements in the cluster. ONOS uses RAFT [22] consensus algorithm to map each data forwarding entity to its master controller. The RAFT algorithm enables the replication of the distributed data structure, administrates over election process of a new leader if an element fails in the cluster, and restores the controller state post-repair. Clusters are managed by Atomix distributed system framework. Atomix is an event-driven reactive Java framework for coordinating distributed systems with fault tolerance [23]. ONOS suffers from challenges ODL and other cluster-based distributed controller systems face in terms of control/flow packets transmissions as clusters increase in size.

B. LOGICALLY-DISTRIBUTED CONTROL PLANE

In [25], the authors propose DISCO, a DIstributed SDN COntrol plane for multi-domain SDN networks. A distributed control channel is utilized by agents plugged into domain controllers to exchange inter-domain network-wide information. They tested the proposal in three use cases: inter-domain topology disruption, end-to-end priority service, and virtual machine migration. Advanced Message Queuing Protocol (AMQP) [27] is used as the East/West interface protocol for inter-domain controller communication. The client-server mode of AMQP is used, limiting scalability with network growth. Limitations of the AMQP publish/subscribe method is further detailed in Table 2.

Authors of [28] study logically-distributed controllers intended for large multi-domain networks. They propose an East/West interface called Communication Interface for Distributed Control plane (CIDC) for communication between controllers in a logically-distributed SDN control plane. The CIDC interface is used by each controller to synchronize its status and services with its distributed peers. Simulation results of various large networks show improved results in terms of delay, overheard, CPU, and memory performance. CIDC uses a custom event-driven protocol for the East/West interface and the paper lacks a detailed performance evaluation of the broker-based model. CIDC brokers enable topology synchronization: Consumer, Producer, DataUpdater, and DataCollector. Disadvantages of the broker-based models include lack of simultaneous read/write operations, limited scalability, and presents a risk of a single point of failure.

III. PROPOSED DISTRIBUTED SDN CONTROL PLANE FRAMEWORK - DSF

DSF is an adaptive framework for the East/West interface designed for heterogeneous, distributed control planes to synchronize topologies using a standardized communication protocol. The malleable and modular architecture of the DSF interface enables implementation in a diverse array of SDN controller platforms. The proposed framework employs a data-centric RTPS model known as the DDS standard [31] for the East/West interface.

An overview of the DSF framework in a heterogeneous, distributed SDN control plane with a flat model architecture is presented in Fig. 2. A description is provided of the three types of distributed control plane strategies enabled by the DSF interface: flat, hierarchical, and T-model. Control plane entities communicate within the network domain by implementing *DomainParticipants* hosted by DSF components. The framework is vendor-agnostic to the selection of the DDS implementation and adaptive to the programming language of the controller platform. To enable topology synchronization, adherence to a messaging protocol is required, referred to as *topics*. The figure further describes the channel used by controller participants to read and write samples of network state information topics, referred to as the data space.

TABLE 1. Comparison of features between previous East/West techniques and the proposed DSF interface.

Proposed Techniques	Controller Platform	Programming Language	Network State Dissemination Strategy			Coordination Strategy	Cross Platform Coordination	Data Storage Technique	East/West Protocol
			Flat	Hierarchical	Hybrid				
HyperFlow [15]	NOX [7]	C++	✓	-	-	Broker-based P2P	✗	WheelFS [16]	Publish/subscribe via distributed file system
ONIX [17]	ONIX	C++	✓	-	-	Cluster	✗	Distributed hash table	ZooKeeper API [24]
OpenDayLight [18]	OpenDayLight	Java	✓	-	-	Cluster	✗	In-memory MD-SAL DB	Akka [19], Raft [22]
ONOS [21]	ONOS	Java	✓	-	-	Cluster	✗	Distributed data structure	Raft
DISCO [25]	Floodlight* [26]	Java*	✓	-	-	Broker-based P2P	✓	Extended DB	Publish/subscribe via AMQP [27] in client-server mode
CIDC [28]	Floodlight*	Java*	✓	-	-	Broker-based P2P	✓	In-memory DB	Custom event-driven protocol
IRIS [29]	Floodlight	Java	-	✓	-	Leader-based	✗	MongoDB	Custom protocol
Kandoo [30]	Kandoo	C++	-	✓	-	Leader-based	✗	In-memory	Messaging channel
DSF	Any	Any†	✓	✓	✓	Broker-less P2P	✓	Platform-dependent	RTPS-DDS [31] standard

*Implemented custom modules are written in the language of the selected platform and requires complete code rewrite to specific programming languages to function in other platforms.
†Portability and interoperability features of DDS standard used by DSF enables multi-platform communication written in various programming languages including C, C++, C#, Java, Ada, Python, etc. Minimal interface coding is required to link DSF modules and internal modules of specific controller platforms.

TABLE 2. Comparison of publish/subscribe models in previous East/West techniques compared to RTPS-DDS used by the proposed DSF interface.

Protocol	Features	How it works	Applications	Strengths	Weaknesses
AMQP [27]	-Message-centric -Centralized multi-broker architecture: client and service protocols	Publish to an exchange agent and queues within the broker entity. AMQP brokers facilitate communication. Focuses on tracking all messages to ensure delivery.	Financial transaction-based systems, banking	-Reliability (TCP) -Interoperability -Scales to many queues -Easy to implement	-Not suitable for large-scale fan-out use cases. -Single points of failure and bottleneck risks at the queues/brokers. -Limited simultaneous read/write operations at queues/brokers.
WheelFS [16]	-Distributed file system designed for flexible wide-area storage.	Files and directories are spread across storage nodes. Files have a primary and two replicas. Clients cache files. Semantic cues applied to path names control behavior.	Data centers, distributed web cache, multi-site email service	-Easy and rapid prototyping of distributed applications -Control of performance, durability, and consistency -Scales to many file nodes	-Limited simultaneous read/write operation at distributed file nodes. -Bottlenecks at distributed file storage nodes.
DDS [31]	-Data-centric -Decentralized broker-less architecture: P2P protocol -Quality-of-service control -Performance oriented	Publish or write to a domain data space with specified topic structure and QoS properties. Peers subscribed to the specific structure and QoS properties read sample from the data space.	High-performance defense, industrial, and embedded applications	-Low latency -Scales to large number of simultaneous readers/writers -No single point of failure -Analogous to relational database management systems	-Requires UDP to avoid establishing large number of connection sessions in large-scale systems. -Not suitable for transaction-based systems.

A. DESIGN OBJECTIVES OF THE ARCHITECTURE

The following are the objectives of the framework architecture:

1) SCALABILITY

A primary objective of the framework is to present a scalable inter-domain SDN solution for data center and enterprise networks. In addition, scalability is a major concern for networks that function with strict delay constraints in a large cluster of nodes. A distributed solution has the capability of providing required performance objectives. The proposed framework should deliver a consistent view of the network to its participating controller entities to fulfill the aforementioned target. This is accomplished through the design of dissemination strategy of synchronization information in distributed control plane architectures detailed in section III-C.

2) HETEROGENEITY

The architecture should demonstrate the capability of vendor and platform agnosticism of the control plane entity, enabling a diverse portfolio of network participants. In addition, the framework should grant various implementations of the DDS standard to co-exist within this architecture.

3) RELIABILITY

The framework should withstand and overcome fail case scenarios, providing continued service in strenuous circumstances. Reliability and robustness are key

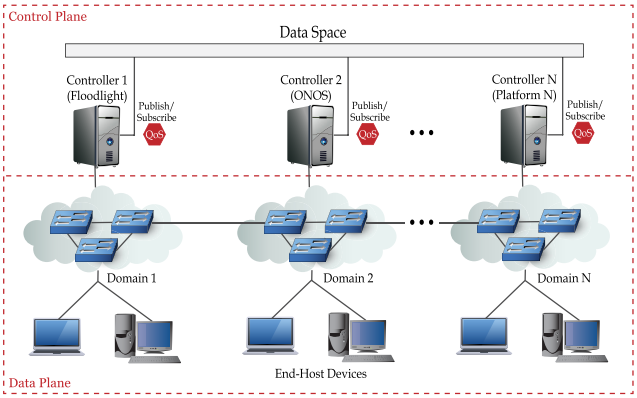


FIGURE 2. The proposed DSF interface utilizes a standardized data-centric real-time publish/subscribe (RTPS) protocol to communicate over a domain data space, enabling distributed SDN controllers to synchronize regardless of underlying platform differences.

non-functional requirements for data center and enterprise networks.

4) SECURITY

Security is key to the integrity of the framework. It should provide secure, encrypted end-to-end data exchange between authenticated participants. The DSF interface enables this capability through the DDS security plugins described in section IV-D.

5) TIMELINESS

Some networks function under strict timescales and require the architecture to provide QoS guarantees for controller-to-controller information exchange.

B. DESIGN OBJECTIVES OF THE EAST/WEST INTERFACE

The East/West interface is the communication protocol between inter-domain control plane entities.

1) MODULARITY

The framework describes a modular design for implementing the East/West interface. The primary benefit of modular design is the separation of tasks carried out by each component. This isolation of network function allows for the DSF modules to interact with specific core internal modules and complete prescribed tasks. The task of the framework modules remains consistent regardless of the implementation platform of the controller. The task description of the DSF modules is provided in section III-D. In addition, the modular design supports test cases to be carried out on individual components to confirm expected behavior during development.

2) PERFORMANCE

In Section V, the controller platforms implemented with DSF interface are presented. A distinguishing aspect of the controller selection is the multi-threading capacity. A controller monitors a vast number of links for changes within a local domain of data forwarding entities. The use of multiple threads to asynchronously monitor local and global link updates improves performance. The East/West interface should run concurrently to controller southbound administrative responsibilities to maintain a real-time view of the overall network topology.

C. DESIGNING NETWORK STATE DISSEMINATION STRATEGY

Early physically-centralized controllers managed the full holistic view of the network. Individual controllers in the distributed control plane maintain only a portion of the holistic view referred to as the local domain. To enable data forwarding between local domains and to present a logically-centralized control plane to the northbound applications, a holistic view must be maintained by at least one of the participating controllers. This requires controllers to exchange local domain network information with one another.

The strategy used to disseminate network state information among inter-domain distributed controllers is classified into two models: flat model and hierarchical model [32]. The proposed framework operates in both types of distributed control plane topologies. The framework additionally is capable of functioning in a hybrid combination of the two models for large-scale, geographically distributed networks, labeled by this study as the T-model.

1) FLAT MODEL

The DSF framework shown in Fig. 2 depicts a flat model, also referred to as a horizontal architecture. Distributed controllers in this topology behave as peer nodes. Nodes publish local network state changes into the domain data space with predefined topic structure and QoS attributes. The topic structure is detailed in section III-E and the QoS attributes in section IV-C. Nodes are subscribed to the network state information updates of all other peers. The updates are stored at individual nodes from which a global view of the administrative domain of the network is built and maintained. In the flat model, every node maintains its global view of the network.

2) HIERARCHICAL MODEL

Fig. 3 describes the proposed framework employed in a hierarchical control plane model, also referred to as a vertical architecture. Distributed controllers follow a chain of command in a tree-topology structure. Child nodes update governing parent nodes of their local state information through publish/subscribe topic samples applied with QoS attributes. The parent stores and combines the state information of the child domains, generating a holistic view. The parent node is a child node of a larger administrative domain entity and the combined view is then shared with those upper-tier entities, known as global controllers. A global controller is the root of the tree where the state information of local and administrative domains are collected. The global controller generates the global view for northbound network management applications.

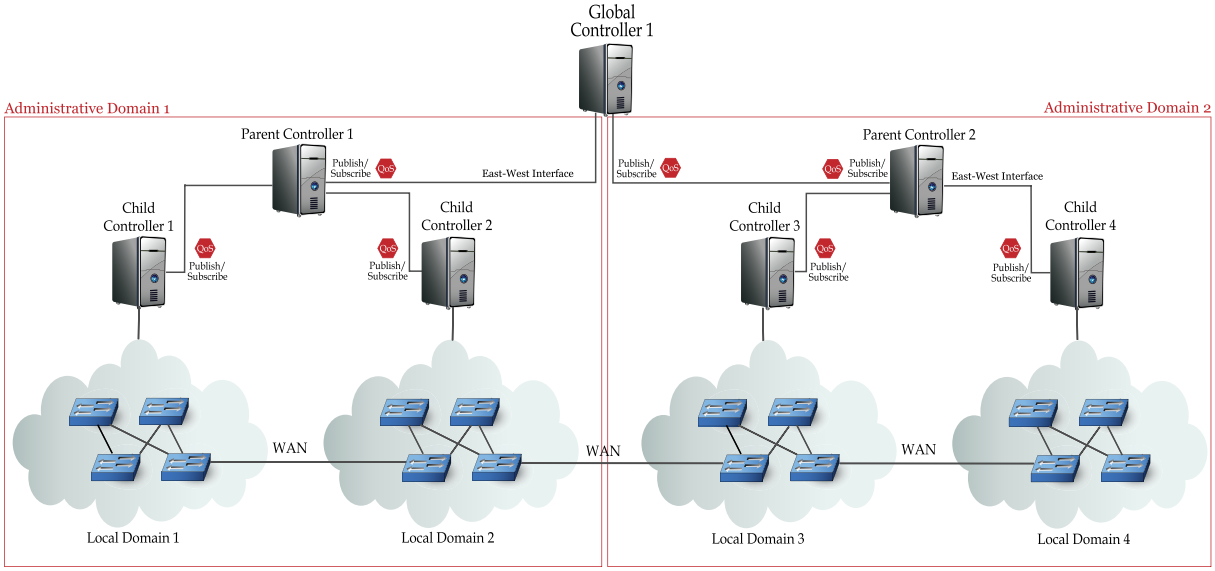


FIGURE 3. DSF framework in a hierarchically distributed controller architecture is illustrated. Parent controllers administrate over child controllers per respective administrative domain. Link updates flow vertically in a tree-topology structure between parent/child controllers. The global controller is the root of the tree maintaining a holistic view of the network for northbound applications managing flow control.

3) T-MODEL

This article proposes a hybrid architecture, referred to as the T-model, designed for large-scale distributed networks expanding multiple geographical locations. T-model uses the vertical distribution of state information for local hierarchical networks and horizontal distribution for global flat networks in large-scale global networks. The network state information moves through the hierarchy in a vertical fashion from child to parent controllers until it reaches the root of the geographic administrative domain. Once the holistic view of the respective domain is formed at the root, the domain view is then disseminated horizontally among peer nodes for each root global controller to attain and construct the holistic view. Fig. 4 presents an example of the described model using DSF interface as the East/West protocol between control plane entities.

D. CONTROLLER ARCHITECTURE AND MODULES

This section presents the modular controller components interacting with the DSF interface. The function of the interface modules remains identical in varying controller platforms, demonstrated between two controller platforms in the implementation section. Fig. 5 describes a controller architecture installed with DSF interface. Controller components are tasked with managing data plane forwarding entities via the southbound interface such as the OpenFlow protocol. DSF components interact with controller internal components to synchronize topologies due to link changes in the local and global domains.

1) PUBLISHER

The function of the Publisher is to monitor the Topology Manager module using the Interface component. A link update is generated when a topology event occurs at the Topology Manager. The sub-module *TopologyService* updates listener entities of the link change. The Publisher module receives this link update and forwards it to the relevant Data Writer entity within the DSF module. The entity then creates a sample of the update following the topic data structure assigned with QoS attributes. This sample or notification is then published into the data space of the specified domain.

2) SUBSCRIBER

The Subscriber module receives link update notifications from the global domain which matches its subscription of the topic data structure and QoS attributes. Data Reader participants within the Subscriber reads samples from the data space matching with the predefined subscriptions. The module then updates the Topology Manager of the received samples through the Interface DSF module.

3) TOPOLOGY MANAGER

This component administrates the local and global view of the network, generating link updates from local domain events and receiving updates from global domains through the Subscriber. The component maintains the topology instance of the network, providing access to network management applications through the northbound API. The implementation of this module is dependent on the architecture of the controller platform. The primary function of network topology maintenance remains consistent across platforms.

4) INTERFACE

The task of coordinating between DSF modules and core internal modules of the controller is performed by the Interface. Two primary sub-components of the Interface is the

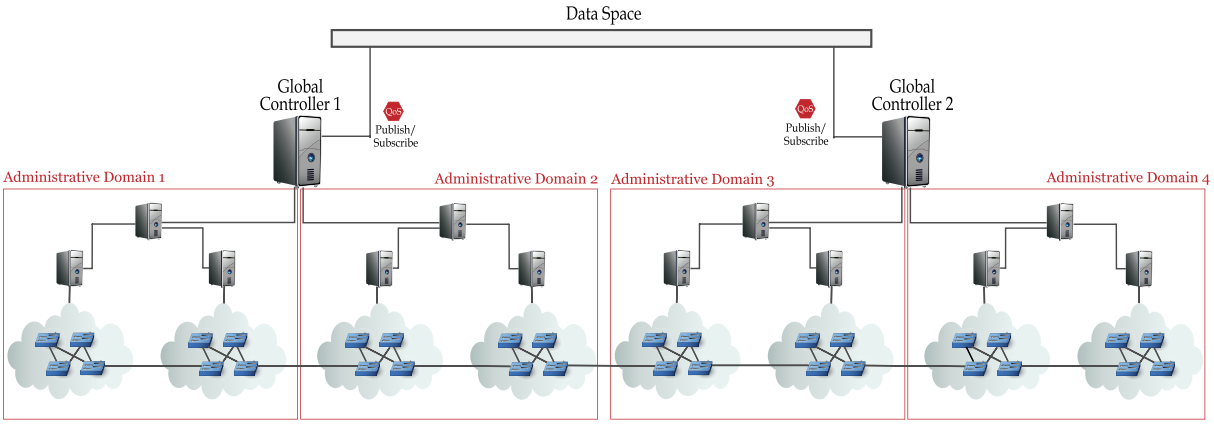


FIGURE 4. DSF in T-model distributed controller architecture. Hybrid of the hierarchical and the flat model for large and geographically distributed networks. Global controllers are the root of each tree-topology structure for multi-administrative local domains, sharing a holistic view of their tree with global nodes via the flat model.

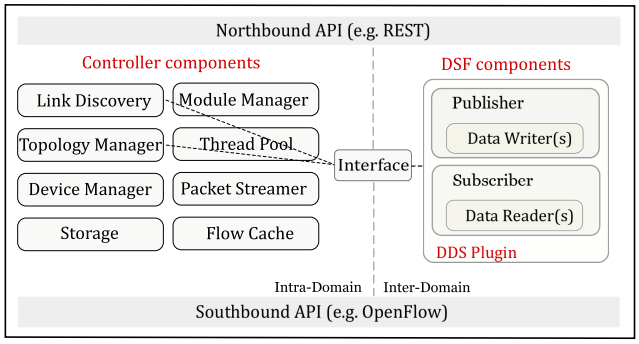


FIGURE 5. The controller architecture of the proposed DSF is presented. Shown on the left side is a subset of controller specific components, while the DSF framework modular components are presented on the right. The DDS Plugin module is the major component of DSF which encapsulates the Publisher and Subscriber sub-components, each containing one or more Data Writer/Data Reader entities respectively.

listener and the *service*. The listener is an interface implemented by the Topology Manager module. When a global link update event arrives at the Subscriber, the service subcomponent notifies all modules implementing the listener, thus notifying the Topology Manager of the update. DSF modules and core internal modules may directly interact without using an intermediary Interface module depending on the architecture of the controller.

E. LINK UPDATE MESSAGE PROTOCOL (LUMP)

LUMP is the messaging protocol in the DSF framework to communicate link updates among inter-domain controllers. The data structure is dependent upon the southbound protocol used by the implementation of the DSF interface. For example, OpenFlow [33] is a southbound communications protocol or an API that provides access to data plane or forwarding entities over the network. Published samples of OpenFlow-based controllers are structured with the following message format:

- *nodeID:* Represents controller ID. This information is used to distinguish between controllers participating in the domain for logging purposes.
- *operation:* Describes the operation of the link discovery update. The operation consists of the link status between data plane entities: link up, down, removed or updated.
- *srcMAC:* Provides the source MAC address.
- *srcPort:* Provides the source port number.
- *dstMAC:* Provides the destination MAC address.
- *dstPort:* Provides the destination port number.
- *latency:* Provides the latency of the uni-directional exchange between data plane entities.
- *type:* Provides the link type between data plane forwarding entities: internal, external, tunnel or invalid.

IV. DATA-CENTRIC REAL-TIME PUBLISH/SUBSCRIBE MODEL

An overview of the data-centric RTPS DDS standard [31], [34] is provided in this section describing core functionalities and components. QoS attributes provided by the standard and its capability of delivering reliable, secure, and time-sensitive data exchange is elaborated. The primary merit of using data-centric RTPS paradigm in the SDN control plane domain is the enabling of a large number of participants to communicate asynchronously and in real-time in contrast to client/server-based or broker-based models which typically incur additional processing delays, limit simultaneous read/write operations, and introduce a risk of single point of failures. In addition, secure end-to-end data connectivity is ensured by DDS through the security plugins enabling authentication, encryption, access control, and logging capabilities.

The data-centricity focus of DDS allows the system to decide what data to preserve and control how that data is shared. DDS implements a virtual store, known as the *Domain Global Data Space* as displayed in Fig. 6. To the individual network node, the global data space looks like native

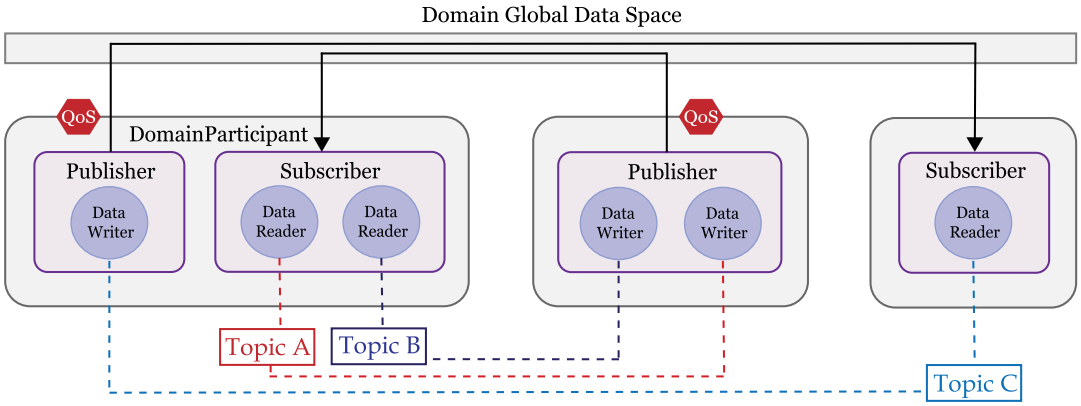


FIGURE 6. The RTPS-DDS communication architecture highlights the function of the domain data space. The domain data space enables peer-to-peer communication with predefined topic structures and QoS profiles. DomainParticipants include Publisher and Subscriber components with one or many Data Reader/Data Writer entities which read/write samples with QoS attributes respectively.

memory accessed via an API. The unit of data exchanged in the global data space is a sample of a specific topic, a data structure tied to a specific set of QoS attributes known as a QoS profile. This profile describes the data samples' non-functional requirements. DDS is unique in its monitoring of non-functional requirements of data it controls, such as reliability, time-constraints, and durability. Using QoS properties, DDS is capable of providing a consistent view of the shared data to all participating peers in the system.

An entity joining the domain data space implements a *DomainParticipant* consisting of a set of *Publisher* and *Subscriber* entities, each capable of containing one or more *Data Writers* and *Data Readers*. Data Writers and Data Readers publish and subscribe data samples with specific data structures and QoS profiles.

A. DESCRIPTION OF KEY ATTRIBUTES OF THE RTPS-DDS MODEL

Listed are the key factors that encompass the functional attributes of the model:

- *Publish/Subscribe paradigm:* Needed for discovery and management of new features, and for incoming/outgoing data among inter-domain controllers.
- *Lifecycle awareness:* DDS standard monitors the information status and activity for its application systems. For example, first and last sample updates for each topic instance.
- *Relational data modeling:* DDS mimics data management of Relational Database Management System (RDMS). Tailored requests are enabled by DDS through its reliance on structure-related topics in terms of time and content settings used by the filters.
- *Reliable multi-cast:* User Datagram Protocol (UDP) sockets are utilized for multi-cast. Enabling real-time streaming of data in contrast to the typical Transmission Control Protocol (TCP) sockets in the client/server approach.

B. ENTITIES WITHIN THE MODEL

The RTPS-DDS model contains the following entities:

- *Domain:* Data publishing/subscription is handled per-domain basis, and domains are virtually separated from each other. Participants of one domain cannot share information with participants in another domain.
- *Topic:* Topics are data structures that define the sample information to be exchanged. A Data Writer and Data Reader are connected indirectly through the shared topic samples. For example, a distributed set of sensors may publish information about a specific topic, e.g., temperature data updates.
- *Publisher:* A publisher is a set of Data Writers updating new samples of the topics that are subscribed to by Data Readers. It may filter samples generated and share only information about the topic should a predefined threshold be met. This QoS feature decreases the load on the network.
- *Subscriber:* The subscriber is a set of Data Readers receiving new samples of the data topics that it is interested in.
- *Discoverer:* Endpoint Discovery Phase (EDP) and Participant Discovery Phase (PDP) are discovery phases within DDS that enable new participants joining the domain to dynamically discover the other participants and their endpoints in the domain.

C. QUALITY-OF-SERVICE ATTRIBUTES IN DSF FRAMEWORK

The following QoS attributes of DDS are proposed for the DSF interface:

- *Reliability:* This attribute defines the level of reliability the Service provides. 'Reliable' value is opted for DSF due to the impact of each link update on the routing of the data packet.
- *Durability:* The attribute expresses if the data should persist after writing time. The value 'Transient',

'Transient-local', or 'Persistent' is suitable for instances of DSF application with dynamically joining SDN controllers per domain, otherwise, the 'Volatile' value is selected.

- *History:* An important QoS attribute for the framework when multiple new link updates are generated before the former ones are communicated to subscribers. This attribute assists in administrating a queue. 'Keep-all' value is selected for the framework for the importance of transmitting all intermediate link updates.
- *Resource limits:* Resource limits attribute defines the resources allocated to the Service for queuing samples on both Publisher and Subscriber entities.
- *Lifespan:* This attribute indicates to the Service the maximum time the sample is valid after being written into the network domain. This cleans up the network of previous, perhaps obsolete, link update samples in DSF.

D. SECURITY

Object Management Group (OMG), the body which formalized the DDS standard, provides a formal security specification for DDS [35] which describes the Security Model and Service Plugin Interface (SPI) architecture for DDS implementations, decoupling security aspects into a set of plugins:

- *Authentication:* The plugin ensures DDS entities, the SDN controllers, are authenticated with set credentials. The purpose is to avoid data contamination from unauthenticated participants.
- *Access Control:* The plugin monitors and administrates access control permissions for DDS topics, domains, etc. for authenticated participants.
- *Cryptography:* The plugin enables encryption, digest, message authentication codes, and key exchange (public/private keys). The aim is to preserve the integrity and confidentiality of the data.
- *Logging:* This plugin provides the capability to log all security events, inclusive of expected behavior and all violations of security policies and errors. The key aim of this plugin is to enable auditing to analyze methods to improve availability.

This set of security plugins enable different implementations of the proposed framework to define policies based on their own security objectives.

V. IMPLEMENTATION

The implementation aims at assessing the performance and adaptive capabilities of the proposed interface in multi-domain control plane environments. The DSF interface is implemented on Floodlight (FL) and Open Network Operating System (ONOS) controller platforms.

Experiment test cases were designed to measure performance metrics in different configurations. Test cases include performance evaluation of FL and ONOS controllers implemented with DSF interface, performance evaluation of DSF-based ONOS compared to Atomix-based ONOS cluster,

```
struct GPS_Position {
    int id;
    string operation;   //UpdateOperation
    string srcMAC;      //DatapathId
    short srcPort;      //OFPort
    string dstMAC;      //DatapathId
    short dstPort;      //OFPort
    string latency;     //U64
    string type;        //LinkType
};
```

List. 1. LUMP topic data structure converted from OpenFlow data types.

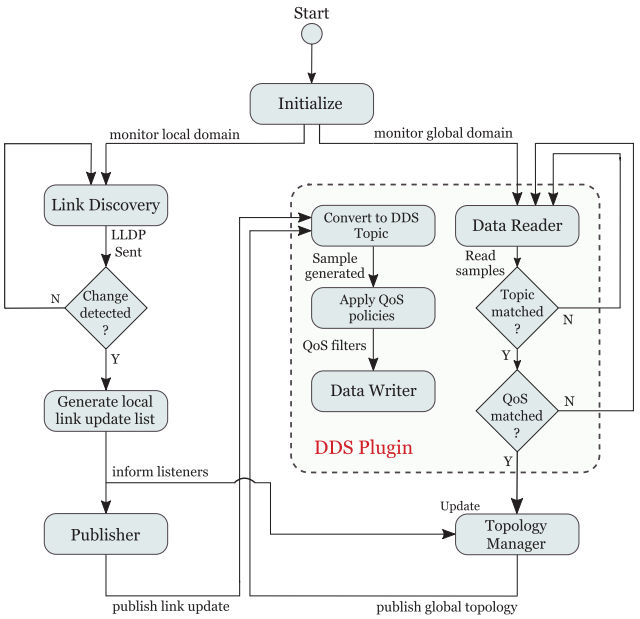


FIGURE 7. The DSF controller algorithm is described by the control flow logic of the system. Two concurrent monitoring states occur after the initialization stage from which local and global link updates are processed towards the Topology Manager to construct the holistic view of the network.

a case study of a multi-organization heterogeneous data center network, and Markov chain analysis of controller state transitions and traffic analysis of RTPS packet transmissions.

LUMP communication protocol is used for topology synchronization between the controllers. The interface definition language (IDL) data structure in C language of the protocol is shown in listing 1. The OpenFlow data types listed in the commented green text are not supported by DDS libraries used for this evaluation and required translation to primitive data types.

A. CONTROL FLOW LOGIC OF THE DSF INTERFACE

Fig. 7 describes the flow of control logic in DSF-based controller platforms. At the initial state, the controller periodically sends Link Layer Discovery Protocol (LLDP) packets encapsulated in a *Packet-out* message to its directly connected data plane forwarding entities (routers/switches). The forwarding entity receiving the packet reads the instructions on the Packet-out message to forward the packet to a specific neighboring peer. The neighbor then forwards the packet via

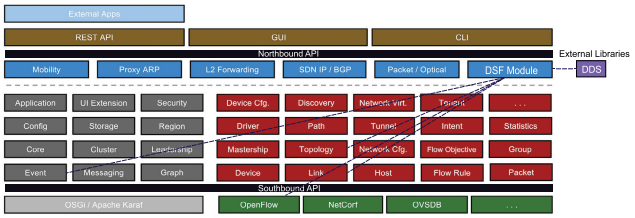


FIGURE 8. ONOS controller consists of a modular architecture [36] divided by network function. DSF module is inserted into the internal modules during run time as an application. Similarly, OpenFlow southbound API is installed in run time to connect to Mininet OpenFlow data plane entities. DSF module interfaces with internal modules to listen for link changes and update the topology module.

Packet-in message to the controller. The controller verifies the uni-directional link between the two peers with its records determining if it is an existing, new, updated, or deleted link in case of no reply. If a change has occurred, a link discovery update list is generated by the controller.

The link discovery update list is observed by two modules in the controller: the internal Topology Manager and DSF Publisher. The Topology Manager updates the topology instance of the controller and the Publisher forwards the update to the DDS Plugin module within the DSF framework. The DDS Plugin converts the update into a topic sample and writes it into the global data space through the Data Writer sub-module with predefined QoS attributes.

Simultaneously, the controller listens for new samples of link discovery updates in the global data space published by peer controllers via DSF module Subscriber. The Data Reader entity of the DDS Plugin module filters the received samples with the link discovery update topic structure and QoS attribute requirements. Once matched, the sample is forwarded to the Topology Manager to update the network topology instance, calculating new possible shortest routes.

B. CONTROLLER ARCHITECTURE OF FL AND ONOS WITH DSF INTERFACE

FL architecture in Fig. 5 describes DSF components coordinating with internal core components using the DSF module Interface. The Interface, as described previously, monitors internal modules for link updates and notifies the Publisher. The Publisher sends the link update as a DDS sample using LUMP topic structure to the global data space. The Subscriber receives link update samples from the data space, the Interface notifies the internal module Topology Manager of the global link update.

ONOS architecture with DSF interface is presented in Fig. 8. The DSF module interacts with internal modules using a topology listener and service, sub-components of the Interface. DSF module listens to *Topology* internal module for a link *Event* to occur. Event is an abstracted module that describes link events. Events compromise of *Link* module abstractions. DSF module extracts the link update information from the Link instance, converts the data to OpenFlow structure using OpenFlow API libraries, and creates a DDS sample in the LUMP message structure using DDS libraries. It then publishes the DDS sample using the Publisher sub-component within the DSF module.

TABLE 3. Physical host specification.

Item	Specification
CPU	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Operating System	Windows 10 Build 18363 (64-bit)
Main Memory	20 GB
Network Adapter	Qualcomm Atheros QCA9377

TABLE 4. Virtual machine specification.

Item	Specification(s)
No. of CPU(s)	2
Operating Systems	Ubuntu 16.04 & 18.04 (64-bit)
Main Memory	2-8 GB
Network Adapters	NAT & Host-only Adapter

TABLE 5. Software specification.

Item	Specification(s)
SDN Controllers	FL v1.2 [26] & ONOS v2.4.0 [21]
SDN Network Emulator	Mininet [37]
RTPS Implementation	RTI Connext DDS 6.0.1 [38]
Network Packet Analyzer	Wireshark [39]
Markov Analysis Tool	Java Application

C. PHYSICAL HOST, VIRTUAL MACHINE AND SOFTWARE SPECIFICATIONS

Table 3 lists the specifications of the physical host machine used for the emulation and Table 4 lists the specifications of the virtual machines (VMs) that host SDN controllers and the SDN emulator. Versions of Ubuntu operating systems for VMs differed due to SDN controller platform dependencies. Main memory assigned to VMs also varied based on the test case configurations elaborated in the experiment description.

Software specifications for the experiments and analysis of performance measurements are listed in Table 5. Two SDN controllers with DSF interface were emulated. Markov analysis tool is a Java application written in this project to automate the process of generating the probability state transition matrices from experiment logs to analyze SDN controller state transitions, described further in experiment descriptions.

D. PERFORMANCE METRICS

The following performance metrics are measured in the experiments:

- *Network Convergence Point (NCP)*: The time taken for the network to converge into one consistent, holistic topology after a link discovery update sample is published into the domain data space.
- *Topology Update Delay (TUD)*: The time taken for the link update packet to reach the peer controller to update the holistic view of the network.

- *RTPS Packet Transmissions (Packets/sec)*: The number of RTPS packets generated and transmitted per second.

E. MODELING NCP

Network convergence point is modeled in this section as a metric describing the time for a set of processes in a publish/subscribe technique to receive notifications. Publish/subscribe methods can be classified into two types: broker-based and broker-less models [40]. DSF architecture enables SDN controllers to communicate in a peer-to-peer fashion through the use of a data-centric RTPS paradigm, thus following a broker-less model with no intermediary entities. Broker-based models use intermediary entities to facilitate communication, such as queues in AMQP and file storage nodes in WheelFS publish/subscribe methods described in Table 2. In a distributed SDN environment, each control plane entity is a decoupled process. The decoupled processes communicate through a channel known as the domain data space. A distributed system consists of a set of processes as denoted by (1).

$$\prod = \{p_1, p_2, \ldots p_x\} \tag{1}$$

A subscription σ is a pair of process p applied with filter ϕ, where the p is subscribed to all other processes publishing samples through the specified filter is denoted by (2). The filter consists of both the topic structure and the QoS profile attached to the sample in DSF.

$$\sigma = (\phi, p) \text{ where } p \in \prod \tag{2}$$

There are four main operations in a publish/subscribe model: publish, subscribe, unsubscribe and notification [41]. A notification n matches the filter ϕ if each property of n matches all constraints detailed in ϕ. That is to say, both topic structure and QoS profile of n matches the filter. Equation (3) denotes the principle of notification n matching a subscription σ.

$$n \sqsubset \sigma.\phi \tag{3}$$

In the event of a publication occurring, the service disseminates the notification to interested subscribers which are processes whose subscriptions matches the notification as described by (3). The time it takes for notification n to reach an individual process p_i is T_{p_i}, we denote the delay as Δ_i. Suppose a set of subscribers $\{p_x, p_y, p_z\}$ are interested in n. We can denote the total delay for the dissemination of n to the set of interested subscribers as the maximum of those delays as seen in (4):

$$T_{ncp} = max\{\Delta_x, \Delta_y, \Delta_z\} \tag{4}$$

Therefore, by measuring the time taken to reach network convergence point (4), we evaluate the performance of DSF in the implementation. The objective is to measure the delay in the dissemination of link update packets within the network as the number of SDN controller nodes increases. Minimizing this delay is essential for controller entities to maintain a real-time, holistic view of the network. In reality, however, an additional processing delay is incurred while generating the holistic view. This processing delay is dependent upon the controller platform and an assumption is made that this delay is measured within Δ_i, the time for notification n to reach process p_i.

TABLE 6. Test Case Parameters - FL.

Parameter	Configuration
Virtual machines	3
Controllers per VM	1-6
Controllers	2-12
Switches per controller	2
Hosts per switch	1
No. of repetitions (n)	10

F. EXPERIMENT DESCRIPTION

The experiments are divided into three primary test cases: performance evaluation of DSF-based controller platforms in homogeneous networks, a case study of a multi-organization heterogeneous DCN, and Markov chain analysis of controller state transitions and traffic analysis of RTPS packet transmissions.

1) TEST CASE 1 - PERFORMANCE EVALUATION OF DSF-BASED CONTROLLER PLATFORMS IN HOMOGENEOUS NETWORKS

The primary objective of this test case is to evaluate the performance of the DSF interface implemented in FL and ONOS controllers in homogeneous networks. The secondary objective is to provide a performance comparison between DSF interface and industry-used techniques. The technique we compare against is Atomix distributed cluster management framework. Atomix-based ONOS cluster networks are compared against DSF-based ONOS networks. The performance evaluation is characterized by capturing measurements of network convergence point and topology update delay after a link update event. The number of controllers is incrementally increased per network configuration to the maximum capacity of the physical host machine.

Table 6 lists the parameters of the Floodlight-DSF homogeneous network experiments. A pair of VMs hosts the controllers while a third contains the Mininet SDN emulator. Mininet assigns each remote controller two data plane switches and a host device per switch, referred to as the local domain. Hosts are used for Internet Control Message Protocol (ICMP) ping tests between local domains administrated by peer control plane entities for verification of the network topology synchronization. Multiple controllers are hosted per VM to maximize the number of controllers emulated in the network with the physical host machine capabilities, reaching 12 FL controllers over two VMs. The number of controllers in the network configuration increased incrementally by one per VM. 10 repetitions are conducted per configuration to

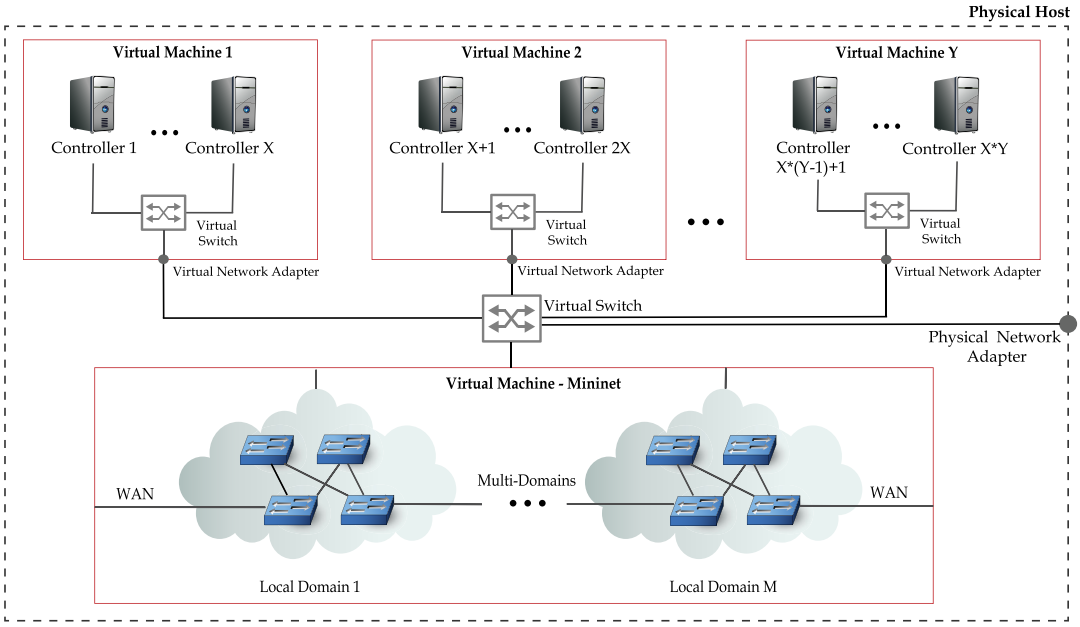


FIGURE 9. Experiment design on the physical host is illustrated. Each of the Y numbers of virtual machine instances runs an X number of controllers, leading to an X*Y number of total data plane domains on Mininet SDN emulator, a host is assigned to southbound forwarding switches. The experiments are conducted based on the test case parameters explained in the experiment description section.

TABLE 7. Test Case Parameters - ONOS.

Parameter	Configuration
Virtual machines	3-7
Controllers per VM	1
Controllers	2-6
Switches per controller	2
Hosts per switch	1
No. of repetitions (n)	10

estimate the variability of the results due to experimental error.

Table 7 lists the configuration for the ONOS-DSF homogeneous network experiments. Each ONOS controller requires 2 GB of main memory per VM. Mininet assigns each remote controller two data plane switches and a host device per switch. Mininet VM and six ONOS controllers VMs exhaust the main memory capacity of the physical host machine. Due to the relatively small-scale network emulation, a comprehensive scalability performance evaluation of the DSF interface is not performed in this work. Instead, we prioritize illustrating consistency of the interface in homogeneous networks and adaptive nature in heterogeneous networks described in the second test case.

To provide a comparison between the DSF interface and alternative methods of topology synchronization, we measure the performance of DSF-based ONOS networks and Atomix-based ONOS cluster networks. Atomix is a cluster management framework that uses brokers to synchronize controller states, referred to as agents. Three agents are configured for Atomix-based ONOS cluster networks. Both methods of ONOS inter-controller communications incrementally increase by one controller up to six emulations of each platform with Mininet VM emulating the network. This provides a baseline performance comparison of DSF interface, expected to synchronize networks in real-time, but compromise performance for consistency and flexibility of adapting to heterogeneous control plane platforms compared to homogeneous distributed systems.

Fig. 9 describes the design of the emulation environment for the various configurations of test cases. Each VM contains one or more sets of controllers, with each controller administrating over a local domain of data forwarding devices. The network is emulated by Mininet on a separate VM. A python script creates the network on Mininet VM, linking the hosts with their assigned switches and switches with their remote controllers via IP address and port number paired ID using TCP connection.

2) TEST CASE 2 – CASE STUDY - MULTI-ORGANIZATION HETEROGENEOUS DATA CENTER NETWORK

In this case study, a collaboration agreement between multiple organizations is assumed requiring data sharing capabilities between inter-domain data center networks, e.g., content delivery networks. Participating organizations utilize different control plane platforms for their specific data center SDN implementation. In this case study, we assume Floodlight and ONOS platforms. Fig. 10 describes the DCN with organizations cooperating with heterogeneous controller platforms. Data plane entities forward content between domains with

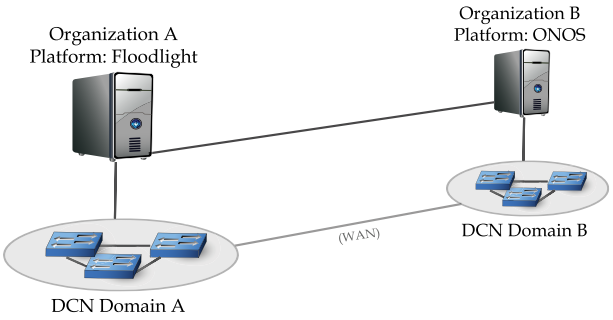


FIGURE 10. Multi-organization heterogeneous DCNs emulated with organizations using different SDN control plane platforms, ONOS and FL. DSF interface is implemented on each platform to enable topology synchronization and performance is measured as the number of controllers from each platform is increased.

flow configurations determined by the controller platforms synchronizing network topologies.

DSF interface is implemented in each controller platform to enable topology synchronization. The test case captures performance metrics as the number of SDN controllers per platform increases per organization. The experiment design consists of controllers hosted on separate VMs assigned with data forwarding and host devices by a Mininet VM. 10 repetitions of the experiments are conducted of the three network configurations: X = 1, 2, and 3, where X is the number of DSF implemented controllers per platform: FL and ONOS.

3) TEST CASE 3 - MARKOV CHAIN ANALYSIS OF STATE TRANSITIONS AND RTPS PACKET TRANSMISSIONS MEASUREMENT

This test case is designed to provide two primary insights in the performance analysis of the DSF interface: Markov chain of controller state transitions and RTPS packets transmission measurement.

The Markov chain analysis experiments measure the probability of state transitions in DSF-based controllers in network configurations with 4, 8, and 12 controllers. For each configuration, an experiment is conducted for 30 minutes to record state transitions. The log from the experiment is then processed by a custom Markov chain analysis tool written for this work to generate the probability transition matrix. The matrix describes the DSF-based controller state behavior during steady-state conditions in different network configurations. Link updates are not simulated from external sources but are generated in steady-state conditions as the cost of links in terms of latency changes and new optimal shortest routes are calculated.

RTPS packet transmissions are captured using Wireshark network packet analyzer tool for experiments containing 2, 4, and 8 DSF-based controllers. The objective is to analyze the number of RTPS packets transmitted over 120 seconds time period and during link update events. A further experiment is conducted to compare all packets transmitted versus RTPS transmissions in an 8-controller network configuration for traffic analysis.

VI. RESULTS AND DISCUSSION

The analysis of results is separated into the following sections: analysis of NCP, analysis of TUD, Markov chain analysis of DSF controller state transitions, analysis of RTPS packet transmissions, and performance evaluation of NCP.

A. ANALYSIS OF NETWORK CONVERGENCE POINT

Fig. 11 describes the time taken for networks to converge into a single, holistic topology after a link discovery update packet is published by a control plane entity. The y-axis describes the time elapsed till NCP for test cases: homogeneous DSF-based controller platforms of FL and ONOS networks in Fig. 11a and Fig. 11c respectively, homogeneous Atomix-based ONOS cluster networks in Fig. 11d, and heterogeneous data center networks compromising of both DSF-based platforms in the same configuration in Fig. 11b.

Fig. 11a shows FL-DSF networks converge within half-a-second to six-tenths of a second on average as the number of controllers per network configuration increases. The standard error of the mean (SEM) of the 10 repetitions is roughly equal between the different network configurations except for the network with two FL-DSF controllers which takes less than half-a-second to converge. This performance remains consistent in DSF implemented heterogeneous networks compromised of FL and ONOS controllers described by Fig. 11b. FL-DSF delays the time to reach convergence in all three DCN configurations. This observation is elaborated in the analysis of the topology update delay metric in the next subsection.

Observations of the ONOS-DSF networks in Fig. 11c clearly demonstrates that the performance of the DSF interface does not contribute to the delay in NCP significantly. DSF-based ONOS networks converge to a single, holistic topology within 35 milliseconds on average as the number of controllers increases. The SEM observed for the configurations on average account for 5 milliseconds or less. Furthermore, the delay to reach NCP in ONOS-DSF networks remains consistent as the number of controllers increases in the network. The consistency of the DSF interface measured in FL-DSF platforms attests to this observation.

Comparison of performance between DSF-based ONOS networks and Atomix-based ONOS cluster networks is shown by Fig. 11c and Fig. 11d respectively. We observe that an increase in average NCP occurs as the number of ONOS controllers increases in the Atomix-based cluster networks while DSF-based networks consistently converge within 23-35 milliseconds. In addition, the time to reach NCP measured in DSF-based ONOS networks is less than Atomix-based ONOS networks when the number of controllers in the configuration is four or more and the time to reach NCP is equal between both configurations at three controllers. Atomix-based networks observe a higher variance in time to NCP between experiments when the number of controllers in configurations is higher, this is also noted in the TUD

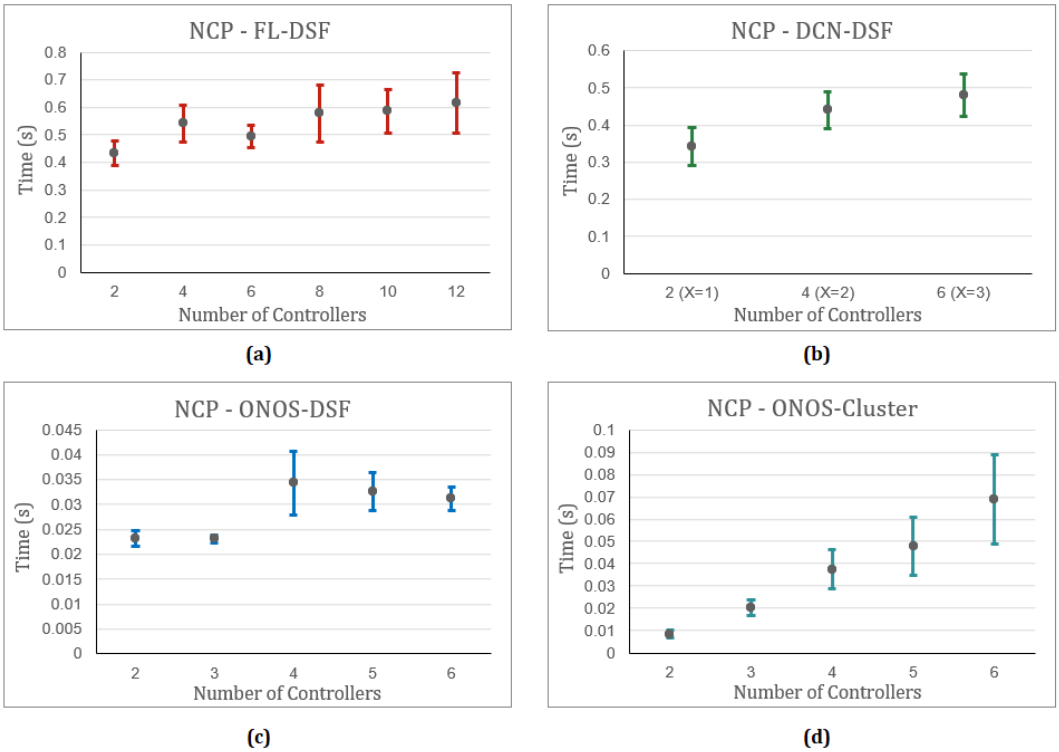


FIGURE 11. Results of 10 repetitions of time taken for networks to converge into a single, holistic view after a link discovery update event occurs is displayed. Networks configurations include a) FL-DSF, b) DCN-DSF, c) ONOS-DSF, and d) ONOS-cluster with differing numbers of controllers per configuration.

metric analysis. Cluster-based distributed systems incur a higher number of control packets to synchronize each entity as participants in the network increases. Atomix agents are the brokers in the cluster synchronizing ONOS controllers and maybe the performance bottleneck for larger cluster configurations.

The performance in terms of NCP depends on the SDN control plane platform as expressed by the results in Fig. 11. ONOS is a highly optimized, multi-threaded platform capable of running concurrent modules efficiently as compared to FL. DSF modules in ONOS react in real-time in ONOS regardless of controller state as opposed to FL which has a delayed response in DSF module activity. DSF interface performs consistently in both platforms as the number of controllers increases. The future scope of the study for DSF interface performance evaluation is performing a large-scale emulation of SDN controller platforms to observe the limits of the data-centric RTPS model.

B. ANALYSIS OF TOPOLOGY UPDATE DELAY

Fig. 12 describes the time taken for controller entities to receive and process a topology update sample. The y-axis describes the time delay till the topology update is received and processed at controllers in test case configurations: homogeneous DSF-based controller platforms of FL and ONOS networks in Fig. 12a and Fig. 12b respectively, homogeneous Atomix-based ONOS cluster networks in Fig. 12c, and heterogeneous data center networks compromising of both DSF-based platforms in Fig. 13.

In Fig. 12a (i), the average topology update delay from 10 repetitions is shown for each FL controller entity in a 4-controller configuration of the network. The mean delay to receive the link update is between a quarter of a second and half-a-second for each control plane entity. In 8 and 12 controller configurations shown by Fig. 12a (ii) and Fig. 12a (iii) respectively, the TUD remains consistent at approximately a quarter of a second. This measurement can be explained by the limitations of the FL controller platform. The optimization of the platform to run its modules concurrently and monitor changes in real-time effects the DSF interface. This is confirmed by the TUD measurements taken in DSF-based ONOS controllers presented in Fig. 12b.

The TUD at each ONOS controller implementing DSF interface in network configurations with 3, 4, and 6 controllers is described by Fig. 12b (i), (ii), and (iii) respectively. ONOS-DSF controllers in Fig. 12b (i) differed by a margin of 17 milliseconds in TUD over 10 repetitions, this may be attributed to the error margins in experimentation at these timescales. ONOS-DSF results in Fig. 12b (ii) if compared to FL-DSF in a similar configuration in Fig. 12a (i) shows that the ONOS-DSF platform performs significantly better in terms of TUD, indicating optimized concurrent function of modules compared to FL. Fig. 12b (iii) results show a highly

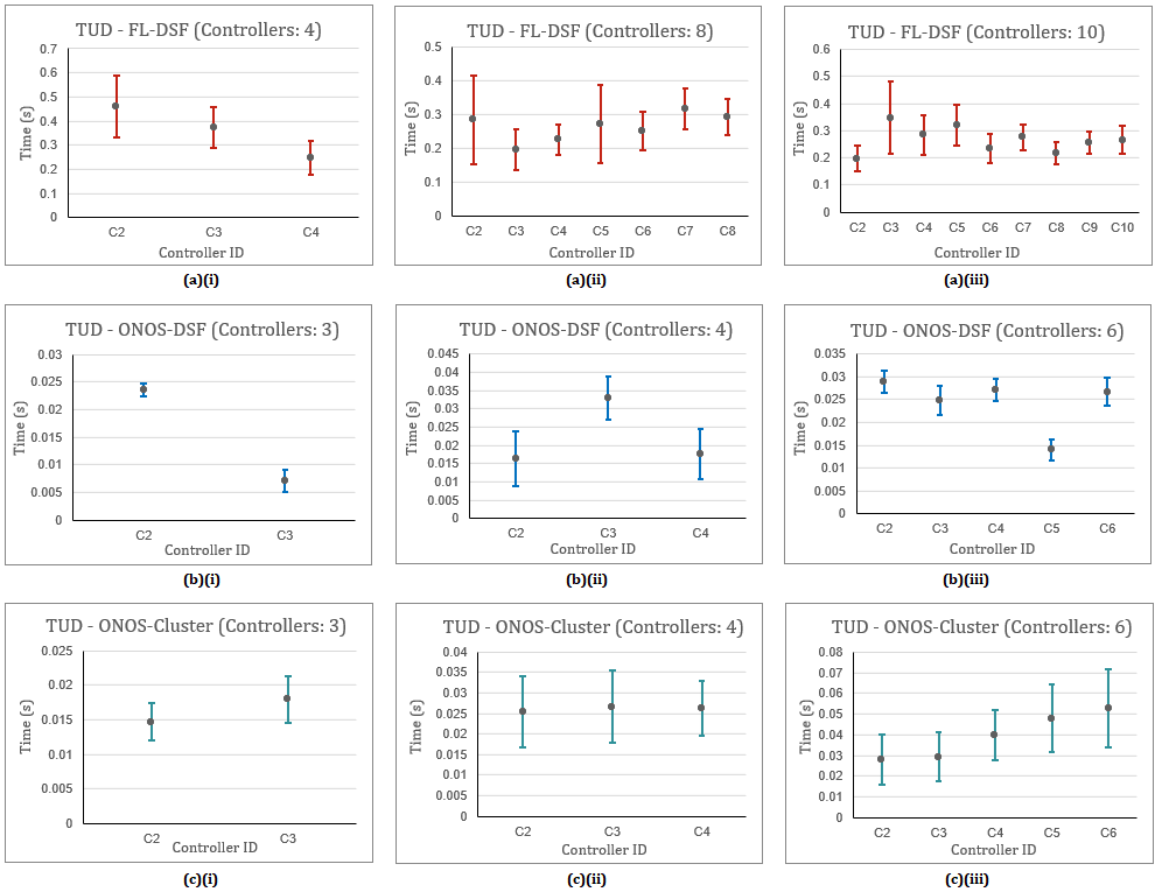


FIGURE 12. The topology update delay of the link update packet to be received and processed by homogeneous controller platforms in 10 repetitions for network configurations a) FL-DSF, b) ONOS-DSF, and c) ONOS-cluster with differing number of controllers per configuration.

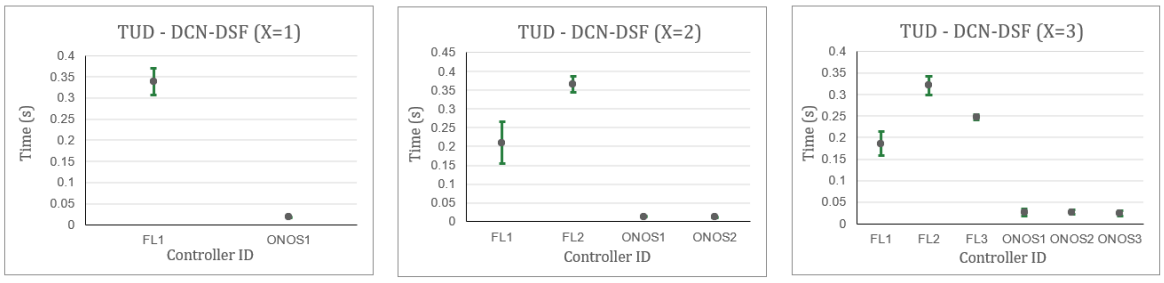


FIGURE 13. The topology update delay of the link update packet to be received and processed by heterogeneous controller platforms in a multi-organization data center network in 10 repetitions for configurations a) 2-controller, b) 4-controller, and c) 6-controller.

consistent network where 5 of 6 control plane entities receives the TUD between 25-30 milliseconds while the sixth processes the TUD even faster than its peers at 15 milliseconds over 10 repetitions. An overall consistent, real-time topology synchronization is observed in these configurations using the DSF interface.

The TUD at each ONOS controller administrated by Atomix-based cluster management brokers in network configurations with 3, 4, and 6 controllers is described by Fig. 12c (i), (ii), and (iii) respectively. In Fig. 12c (i), both controllers in the 2-controller cluster receive and processes the link update within 3 milliseconds of each other on average. A similar pattern is observed for the 4-controller configuration in Fig. 12c (ii). In configurations with a fewer number of controllers, it is noted that Atomix-based clusters synchronize faster with three Atomix agents. However, as the

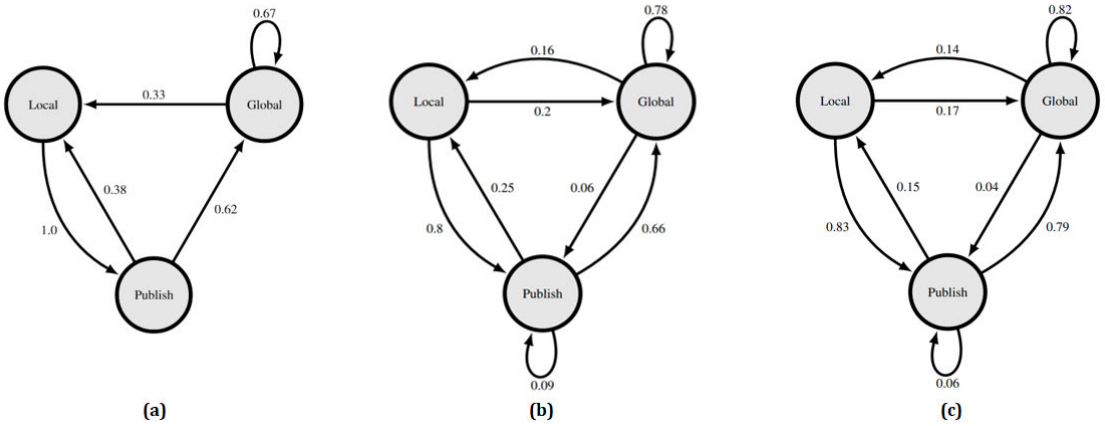


FIGURE 14. Markov chain analysis of DSF controller state transitions from patterns extracted from experiment logs in configurations a) 4-controller, b) 8-controller, and c) 12-controller.

entities within the cluster increases, some controllers receive and process the update with a larger delay as described by Fig. 12c (iii) where controllers with ID C4, C5, and C6 take over 40 milliseconds. In contrast, the proposed DSF interface implemented on ONOS controllers in the same configuration receives and processes the update within 30 milliseconds, shown by Fig. 12b (iii).

The TUD to receive and process link updates in heterogeneous DCN network configurations with 2, 4 and 6 controllers is described by Fig. 13a, Fig. 13b, and Fig. 13c respectively. The configurations include 1, 2, and 3 controllers from each DSF-based SDN platform. The analysis of NCP results implies that the DCN-DSF configuration topology synchronization time is dependent upon the FL controller processing time. The results in this figure confirm the behavior as ONOS controllers are able to process the link update within 30 milliseconds, while FL controllers take a significantly longer time, ranging between two-tenths and four-tenths of a second.

C. MARKOV CHAIN ANALYSIS OF DSF CONTROLLER STATE TRANSITIONS

Markov chain models of DSF-based controller state transitions in network configurations with 4, 8, and 12 controllers is described by Fig. 14a, Fig. 14b, and Fig. 14c respectively. An analysis of the experiment log showed a state transition pattern between three primary controller states: local, global, and publish.

The *local* state is one of two topology update states which occurs when the local link discovery protocol triggers a link update from a data plane entity administered by the controller. As described by the control flow diagram presented in Fig. 7, the next phase of the controller logic is to update topology followed by publishing the link update through DSF modules, this is known as the *publish* state. The transition probability is 1.0 due to each link update requiring a publish event, as shown by the 4-controller configuration Markov chain model in Fig. 14a. The second topology update state is known as the *global* state. This state is triggered by a subscription event from the DSF module, Subscriber, which listens for peer controller link updates. This is the most frequented state as observed by the transition probabilities due to individual controller listening to all link updates by peer controllers to maintain a consistent view of the network.

An interesting observation is made for configurations with a higher number of DSF controllers in Fig. 14b and Fig. 14c. The probability of state transition between local and publish state p is expected to equal 1.0. However, p is observed to be 0.8 and 0.83 respectively. A closer inspection of the Markov chain models provides an insight into the cause. From the local state, the only other state the controller transitions to is global at a probability of $1 - p$ or 0.2 and 0.17 respectively. The implication of this observation is that after a local update is discovered and before the update is published by the *Publisher* DSF module, the controller state transition is interrupted by receiving a global update from the DSF module *Subscriber*. Once the global update state is achieved, the controller transitions from global to publish state to complete the publication of the prior local update.

The transition probabilities between states were generated through a custom Java application written to automate the extraction of state information from the experiment log file, analyze state transition patterns, and generate the probability transition matrix. The following stages of processing occur in the analysis tool: extraction of the transition pattern from the log file, counting the number of unique states (C) from the transition pattern to create a double array of size C-by-C, the summation of the number of state-to-state transitions, and generating the probability transition matrix by dividing the summation C-by-C matrix by the total count of transitions from each state.

Fig. 16 displays the analysis output of the Markov chain tool from a log of an experiment with an 8-controller configuration. A total of 262 state transitions is observed in total

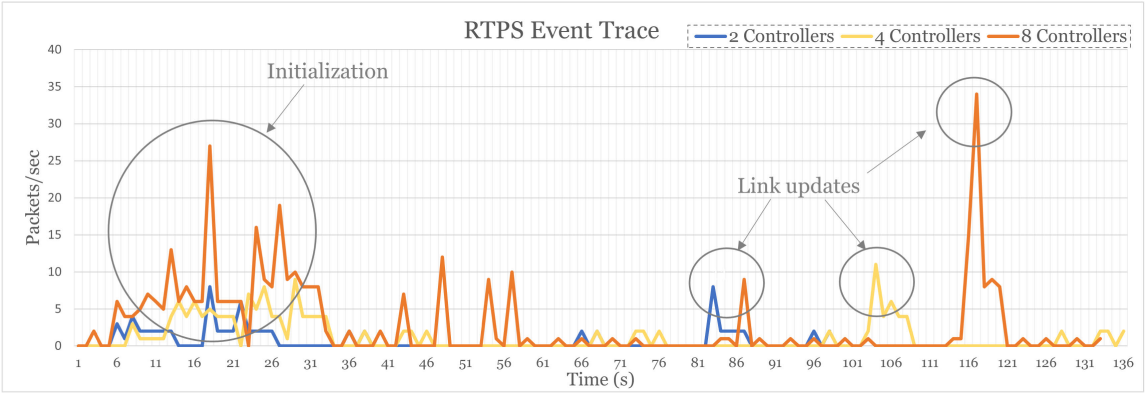


FIGURE 15. RTPS events trace versus time for the network configurations: 2, 4, and 8 controllers respectively. After initialization of the network, link update events occur as link changes are discovered by the LLDP process over a time period of 136 seconds.

```
<terminated> GenerateTransitionMatrix [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (1 Dec 2020, 11:05:45)
Pattern: [L, G, P, L, G, P, G, G, G, G, G, G, G, G, G, G, L, P, L, P, G, G, G, G, G, G, L, P, G, G, G, G,
G, L, G, P, G, G, G, G, G, L, P, L, P, G, G, L, P, G, G, G, G, L, P, G, G, G, G, G, G, G, G, G, G, G, L,
G, P, G, G, P, P, L, P, L, P, G, G, G, G, G, G, G, G, G, G, G, G, P, G, G, L, G, P, L, P, G, L, P, L,
P, G, P, P, P, G, G, G, G, G, L, P, G, G, G, G, G, L, P, L, P, G, L, P, G, L, P, L, P, L, P, G, G, L, P, G, G, G,
L, P, G, G, G, G, G, L, P, G, G, G, G, G, G, G, G, G, G, G, G, G, G, L, L, P, G, G, G, G, G, G, G, G,
G, G, G, G, L, G, P, L, G, P, P, G, G, G, G, G, G, L, P, L, P, G, L, P, G, L, P, L, P, G, G, G, G, G,
G, G, G, G, G, G, G, G, L, P, G, G, G, G, L, P, G, L, P, G, G, G, G, G, G, G, G, G, L, G, P, G, G, G,
G, G, G, G, L, P, G, G, G, G, G, G, G, G, G, G, G, G, L, P, G]
Size of Pattern: 262
Number of States: 3
States: [L, G, P]
Transition Count Matrix: [[0, 8, 32], [27, 136, 11], [12, 31, 4]]
Sum of rows: [40, 174, 47]
Probability Matrix: [[0.0, 0.2, 0.8], [0.15517241, 0.7816092, 0.06321839], [0.25531915, 0.65957445,
0.08510638]]
-------------------------------------------------------
---------- Transition Matrix ----------
-------------------------------------------------------
States  |       L       |       G       |       P       |
-------------------------------------------------------
L       |       0       |       0.2     |       0.8     |
-------------------------------------------------------
G       |       0.16    |       0.78    |       0.06    |
-------------------------------------------------------
P       |       0.26    |       0.66    |       0.09    |
-------------------------------------------------------
```

FIGURE 16. Markov chain analysis tool automates the process of extracting the probability transition matrix from the experiment log file. The figure illustrates the process extracted from an 8 controller configuration experiment conducted over 30 minutes resulting in 262 state transitions.

TABLE 8. Factors and their levels for DSF interface NCP study.

Factors \ Levels	1	2	3
Controller Platform	Floodlight-DSF	ONOS-DSF	
No. of Controllers	2	4	6

for this configuration. For 4 and 12 controller configurations, 122 and 389 state transitions are observed over a time period of 30 minutes in steady-state conditions.

D. ANALYSIS OF RTPS PACKET TRANSMISSIONS

Fig. 15 describes an RTPS packet transmissions trace of network configurations with 2, 4, and 8 controllers over a time period of 136 seconds captured by the Wireshark network packet analyzer tool. During the initialization of the network, all three configurations observe an increased volume of RTPS packets in the network caused by controllers exchanging their own initial set of data plane entity links. The network with the largest set of controllers reaches the highest packets/sec value at around 27 RTPS packets/second. The RTPS events after the initialization state reduce by a significant amount and stabilize by t = 34 seconds, periodically exchanging link updates due to link cost discoveries in terms of latency. When a link update is triggered during the experiment by simulating a link failure, RTPS messages in the network peak again before converging to the single, holistic view. Note in this experiment the link failure occurred at different times.

Fig. 17 displays the packet activity trace over time of all packets transmitted in the network compared to RTPS events in an 8-controller network configuration. RTPS events make up a small fraction of the packets transmitted during the initialization process of the network at t = 12 seconds. During a link update event, occurring at approximately t = 118 seconds, RTPS events make up the majority of packet activity in the network before returning to steady-state.

Fig. 18 presents the behavior of the network during a link update event at t = 3 seconds between three network configurations with 2, 4, and 8 controllers. Configurations with a higher number of controllers transmit more packets/sec in the network, approximately 34 and 11 RTPS packets/sec for configurations 8 and 4 controllers respectively compared to 8 RTPS packets/sec for configuration with 2 controllers. All three configurations stabilize by t = 8 seconds. Regardless of the network configuration, background RTPS control packet mechanisms continue after the link update is transmitted and have no influence on the network convergence time.

E. PERFORMANCE EVALUATION OF NCP

For a thorough performance evaluation of NCP via DSF interface, we opted to perform a *Two Factor Full Factorial Design with Repetitions* [42]. The factors that affect performance are the DSF-based SDN controller platforms and the number of SDN controllers. Each factor contains multiple levels. Table 8 shows the factors and their respective levels evaluated for the performance study.

y_{ijk} = Response (observation) in the kth replication of experiment with factors A=No. of Controllers and B=Controller Platforms at levels i=3, j=2, and r=10, respectively, where r is the number of repetitions.
μ = Mean response

TABLE 9. Analysis of variation (ANOVA) of network convergence point from a Two Factor Full Factorial Design with Repetitions.

Component	Sum of Squares	% Variation	Degree of Freedom	Mean Square	F-Comp.	F-Table
y	7.1506		54			
$\bar{y}...$	3.6384		1			
$y - \bar{y}...$	3.5122	100.0	53			
No. of Controllers	0.0324	0.9	2	0.0162	1.3	2.4
Controller Platform	2.8561	81.3	1	2.8561	227.6	2.8
Interactions	0.0214	0.6	2	0.0107	0.9	2.4
Errors	0.6023	17.1	48	0.0125		

$$S_e = 0.11$$

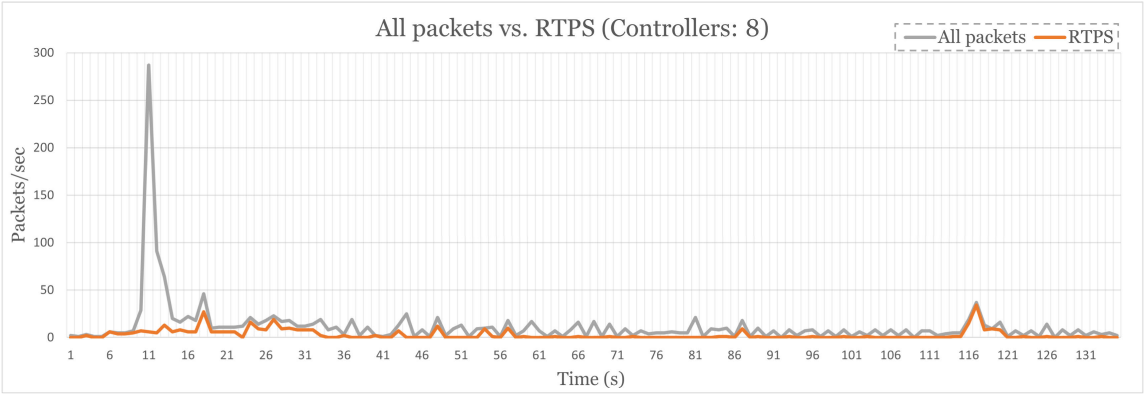


FIGURE 17. Packet activity trace over time of RTPS events compared to all packets in the network in an 8-controllers configuration. RTPS events count for a small percentage of all packets generated in the initialization process of the network. During a link update event, RTPS packets make up the majority of traffic before returning to the normal state.

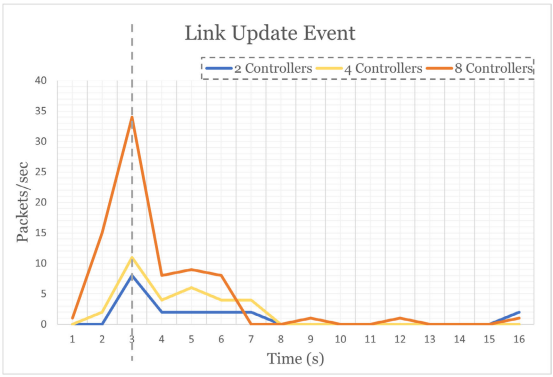


FIGURE 18. RTPS trace over time comparison between a differing number of controller configurations in the network. The comparison begins at the discovery of a link update by the LLDP process, describing the number of RTPS packets/sec generated to converge the network over a period of 16 seconds.

α_i = Effect of factor A at level i
β_j = Effect of factor B at level j
$\gamma_{AB_{ij}}$ = Interaction between factors A and B
$\gamma_{e_{ijk}}$ = Experimental error

Allocation of variation [43] for the aforementioned factorial design is described by the equation below:

$$\Sigma_{ijk} y_{ijk}^2 = abr\mu^2 + br\Sigma_j \alpha_j^2 + ar\Sigma_i \beta_i^2 + r\Sigma_{ij}\gamma_{ij}^2 + \Sigma_{ijk} e_{ijk}^2$$

The results of the ANOVA in Table 9 provides insight into the factors affecting the NCP in the network. The variation of NCP due to the number of controllers is 0.9% from 10 repetitions of the experiments in multiple configurations on both controller platforms. In contrast, the variation of NCP due to controller platforms is 81.3%, confirming the behavior observed in the analysis of performance metrics captured. Experimental errors account for 17.1% of the variation in NCP. DSF interface performance behavior is significantly dependent upon the controller platform it is implemented on.

Another insight from the ANOVA table is the interaction between the factors affecting NCP measurements. The interaction between the controller platform and the number of controllers accounts for 0.6% of the variation in NCP.

From these observations of variation, it can be deduced that the DSF interface performs consistently in the East/West domain as the number of controllers is scaled up despite the constraints of the controller platform.

VII. CONCLUSION & FUTURE WORKS

In this paper, the limitations of distributed SDN control plane were highlighted. The lack of a standardized communication method for East/West interfaces among distributed SDN controllers is a primary concern for the adoption in heterogeneous, multi-domain network environments. The paper proposed the use of a data-centric RTPS communication model among control plane entities to overcome the outlined limitations, referred to as DSF - a distributed SDN control

plane framework. In DSF, control plane entities, arranged in parent/child (vertical) or peer-to-peer (horizontal) manner exchanged link discovery updates in the proposed LUMP message format to synchronize the holistic topology maintained by participating entities, enabling routing of data plane packets over multiple domains. Performance evaluation of DSF interface implemented on Floodlight and ONOS controller platforms was conducted in homogeneous and heterogeneous networks. The experiment test cases consisted of controllers maintaining a single, holistic view of the network by exchanging link updates with an increasing number of control plane entities. The performance metrics collected and analyzed showed that the proposed East/West interface behaved consistently as the number of controllers increased and topology synchronization occurred in real-time, relative to the optimization of the controller platform selected.

Future research includes investigating a data-centric real-time publish/subscribe communication model to enable northbound applications to access the controllers to perform SDN control optimizations. Network optimization techniques such as artificial intelligence to gather, store and process long-term traffic flow patterns to create prediction models of future traffic for pre-emptive fast handovers, mobility, and more. In addition, investigating availability techniques in DSF-based control plane networks, creating new topic structures and QoS profiles for controller-to-controller polling. Furthermore, a comprehensive performance study of the quality-of-service attributes of the data-centric RTPS-DDS communication paradigm in the DSF interface needs to be studied in a large-scale emulation to identify the key factors affecting performance in a distributed, multi-domain environment.

ACKNOWLEDGMENT

The authors would like to thank the members of the Real-Time Embedded Systems Lab, Department of Computer Engineering, and King Fahd University of Petroleum and Minerals for their support.

REFERENCES

[1] "Cisco annual Internet report—Cisco annual Internet report (2018-2023) White Paper," Cisco Systems, Inc., San Jose, CA, USA, White Paper C11-741490-01. Accessed: Dec. 21, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] "Software-defined networking: The new norm for networks," Open Netw. Found., Palo Alto, CA, USA, White Paper, May 2013. Accessed: Dec. 23, 2020. [Online]. Available: https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/

[3] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, Jun. 2015.

[4] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[5] D. Kreutz, F. M. V. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2013, pp. 55–60.

[6] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013.

[7] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, Jul. 2008.

[8] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.

[9] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017.

[10] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, Aug. 2013, pp. 15–26.

[11] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Sep. 2013.

[12] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[13] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[14] N. Shah, P. Giaccone, D. B. Rawat, A. Rayes, and N. Zhao, "Solutions for adopting software defined network in practice," *Int. J. Commun. Syst.*, vol. 32, no. 17, p. e3990, Nov. 2019, doi: 10.1002/dac.3990.

[15] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, vol. 3, 2010, pp. 1–6.

[16] J. Stribling, Y. Sovran, I. Zhang, X. Pretzer, J. Li, M. F. Kaashoek, and R. T. Morris, "Flexible, wide-area storage for distributed systems with WheeLFS," in *Proc. NSDI*, 2009, pp. 43–58.

[17] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *Proc. OSDI*, vol. 10, 2010, pp. 1–6.

[18] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2014, pp. 1–6.

[19] *Akka: Build Concurrent, Distributed, and Resilient Message-Driven Applications for Java and Scala | Akka*. Accessed: Nov. 12, 2020. [Online]. Available: https://akka.io/

[20] D. Suh, S. Jang, S. Han, S. Pack, T. Kim, and J. Kwak, "On performance of OpenDaylight clustering," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Jun. 2016, pp. 407–410.

[21] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 1–6.

[22] E. Sakic and W. Kellerer, "Response time and availability study of RAFT consensus in distributed SDN control plane," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 1, pp. 304–318, Mar. 2018.

[23] *Atomix—A Reactive Java Framework for Building Fault-Tolerant Distributed Systems*. Accessed: Nov. 12, 2020. [Online]. Available: https://atomix.io/

[24] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-free coordination for Internet-scale systems," in *Proc. USENIX Annu. Tech. Conf.*, 2010, vol. 8, no. 9, pp. 1–14.

[25] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–4.

[26] *Floodlight Controller—Project Floodlight*. Accessed: Sep. 14, 2020. [Online]. Available: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview

[27] S. Vinoski, "Advanced message queuing protocol," *IEEE Internet Comput.*, vol. 10, no. 6, pp. 87–89, Nov. 2006.

[28] F. Benamrane, M. Ben Mamoun, and R. Benaini, "An east-west interface for distributed SDN control plane: Implementation and evaluation," *Comput. Electr. Eng.*, vol. 57, pp. 162–175, Jan. 2017.

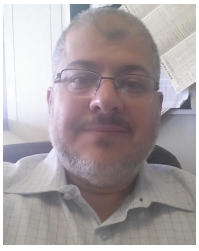
[29] B. Lee, S. H. Park, J. Shin, and S. Yang, "IRIS: The openflow-based recursive SDN controller," in *Proc. 16th Int. Conf. Adv. Commun. Technol.*, Feb. 2014, pp. 1227–1231.

[30] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 19–24.

[31] G. Pardo-Castellote, "OMG data-distribution service: Architectural overview," in *Proc. 23rd Int. Conf. Distrib. Comput. Syst. Workshops*, 2003, pp. 200–206.
[32] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, "Distributed SDN controller system: A survey on design choice," *Comput. Netw.*, vol. 121, pp. 100–111, Jul. 2017.
[33] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[34] *DDS Portal—Data Distribution Services*. Accessed: Nov. 16, 2020. [Online]. Available: https://www.dds-foundation.org/
[35] *About the DDS Security Specification Version 1.1*. Accessed: Nov. 16, 2020. [Online]. Available: https://www.omg.org/spec/DDS-SECURITY/
[36] *Overview (ONOS Java API (2.4.0))*. Accessed: Nov. 16, 2020. [Online]. Available: http://api.onosproject.org/2.4.0/apidocs/
[37] *Mininet Overview—Mininet*. Accessed: Oct. 12, 2020. [Online]. Available: http://mininet.org/overview/
[38] R.-T. Innovations. *Software System Integration With Connext DDS Professional | RTI*. Accessed: Oct. 12, 2020. [Online]. Available: https://www.rti.com/products/connext-dds-professional
[39] *Wireshark. Go Deep*. Accessed: Oct. 15, 2020. [Online]. Available: https://www.wireshark.org/
[40] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surveys*, vol. 35, no. 2, pp. 114–131, Jun. 2003.
[41] R. Baldoni, R. Beraldi, S. T. Piergiovanni, and A. Virgillito, "On the modelling of publish/subscribe communication systems," *Concurrency Comput., Pract. Exper.*, vol. 17, no. 12, pp. 1471–1495, Oct. 2005.
[42] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York, NY, USA: Wiley, 1991.
[43] E. R. Girden, *ANOVA: Repeated Measures*, no. 84. Newbury Park, CA, USA: Sage, 1992.

BASEM ALMADANI received the B.Sc. degree in computer engineering from KFUPM, in 1997, and the M.Sc. degree in industrial automation and the Ph.D. degree from the Institute for Automation, Montan University of Leoben (MUL), Austria, in 1999 and 2005, respectively. He joined the SABIC International Team to manage industrial automation project in Vienna, Austria. He joined KNAPP Systems Integration in 2001, Leoben, Austria, as a Logistics Automation Specialist. He was the Chairman of the Computer Engineering Department, KFUPM, from 2009 and 2014, where he is currently the Director of the Alfozan Academy, for Leaders Development in Non-Profit Sector Program. His areas of research interests include real-time systems integration, distributed systems, middleware, the IIoT, and IR 4.0.

ABDURRAHMAN BEG received the B.Sc. degree in computer science from Universiti Tenaga Nasional, Malaysia, in 2013, and the M.Sc. degree in computer networks from the King Fahd University of Petroleum and Minerals, Saudi Arabia, in 2020. His research interests include software-defined networking, mobile and wireless networks, machine learning, cybersecurity, and IoT.



ASHRAF MAHMOUD (Member, IEEE) received the B.Sc. degree in electrical and computer engineering from Kuwait University, in 1990, the M.Eng. degree in engineering physics (computer systems) from McMaster University, Hamilton, ON, Canada, in 1992, and the Ph.D. in systems and computer engineering from Carleton University, Ottawa, ON, Canada, in 1997. From 1997 to 2002, he was with the Nortel Networks Research and Development, where he focused on the development and evaluation of radio resource management algorithms for broadband and 3G networks. Since 2002, he has been with the Computer Engineering Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi. His research interests include performance evaluation and simulation techniques, mobile and wireless sensor networks, and the IoT.