Software-Defined Networking in wireless ad hoc scenarios: Objectives and control architectures

M. Fogli, C. Giannelli and C. Stefanelli

Journal of Network and Computer Applications, 2022

Cite this as

M. Fogli, C. Giannelli and C. Stefanelli, "Software-Defined Networking in wireless ad hoc scenarios: Objectives and control architectures," in *Journal of Network and Computer Applications*, vol. 203, July 2022, Art. no. 103387, doi: 10.1016/j.jnca.2022.103387.

Software-Defined Networking in Wireless Ad hoc Scenarios: Objectives and Control Architectures

Mattia Fogli[a], Carlo Giannelli[b,*], Cesare Stefanelli[a]

[a]*Department of Engineering, University of Ferrara, Italy*
[b]*Department of Mathematics and Computer Science, University of Ferrara, Italy*

Abstract

While SDN has traditionally targeted fixed and wired environments with hundreds of nodes, recently its main principles have also been followed in wireless ad hoc scenarios. However, node mobility and heterogeneity characterizing such dynamic scenarios prevent the adoption of purely centralized control solutions, pushing for more articulated and distributed architectures. We provide a thorough analysis of state-of-the-art literature related to the adoption of SDN in wireless ad hoc scenarios, ranging from WSNs with mostly fixed sensors and WMNs with mobile clients but stable routers to MANETs with every node freely moving on the field, VANETs with vehicles speeding through the network road, and FANETs with aerial vehicles flying in every direction. For each identified wireless ad hoc scenario, we present the rationale behind the adoption of SDN, recognizing that the objective of the SDN controller is far beyond mere routing management in such challenging scenarios. Furthermore, we detail the proposed control architectures, showing how the peculiar characteristics (and related goals) of a given scenario influence the architecture design. Finally, we point out hybrid control architectures as a notable trend in SDN solutions targeting wireless ad hoc scenarios.

Keywords: Software-Defined Networking, Control architectures, Wireless ad hoc scenarios, WSN, WMN, MANET, VANET, FANET

1. Introduction

Software-Defined Networking (SDN) has been traditionally adopted in fixed and wired scenarios composed of several nodes, such as datacenters and enterprises. In fact, its centralized approach based on a controller gathering topology information and computing routing tables for routers and switches perfectly fits scenarios requiring to control and manage nodes in a coordinated manner, with the primary goal of maximizing the flexibility of infrastructures. Furthermore, SDN has been successfully deployed in wide area networks. For example, B4 [1] is a private wide area network connecting Google's data centers geographically distributed all over the world that leverages SDN principles for improving link utilization.

However, recent research efforts have demonstrated how SDN principles can also be applied without relying on a purely centralized approach. Typically, centralized controllers are replicated to increase scalability and resiliency, resulting in distributed solutions that consist of multiple controllers interacting with each other, e.g., in a flat or hierarchical structure. In other words, while first proposals had presented strictly centralized control architectures in sharp contrast with traditional networking approaches, later work recognized that architectural solutions with some degree of distributed control could provide valuable benefits. For instance, hierarchical solutions with multiple sub-controllers each one administrating a network partition allow to increase the scalability of SDN in huge networks.

Distributed control architectures have been already widely adopted in environments pioneering SDN, from datacenters and enterprises to cellular core networks and the Internet of Things. Most recent contributions have focused on how the design of control architectures impact on SDN capabilities and performance. By relaxing the canonical centralized approach and partially giving back control features to routers and switches, recent literature has demonstrated that it is possible to achieve

*Corresponding Author
Email addresses: mattia.fogli@unife.it (Mattia Fogli), carlo.giannelli@unife.it (Carlo Giannelli), cesare.stefanelli@unife.it (Cesare Stefanelli)

a suitable trade-off between fault-tolerance (characterizing traditional distributed solutions to networking) and optimal decision-making (enabled by logically centralized solutions).

Distributed control architectures represent a paramount step forward in spreading SDN to other scenarios, particularly wireless ad hoc ones. In fact, such scenarios are typically characterized by a high degree of node mobility with wireless routers and/or mobile nodes that generate short-living connections in an impromptu manner. This prevents the adoption of purely centralized solutions.

1.1. Motivations

The peculiarities of wireless ad hoc scenarios have pushed for articulated control architectures provided with tailor-made capabilities. In contrast to enterprises and data centers (where the proposed solutions strictly adhere to the canonical SDN principles), the proposals targeting wireless ad hoc scenarios are highly heterogeneous. Such heterogeneity makes it difficult to analyze and compare the state-of-the-art literature systematically. The lack of a systematic and comprehensive review prevents identifying the most distinctive features and recurrent design patterns of control architectures for software-defined wireless ad hoc networks.

Several surveys have analyzed the current literature, but only regarding a single wireless ad hoc scenario. In particular, [2, 3] survey SDN in Wireless Sensor Networks (WSNs); [4] focuses on SDN in Wireless Mesh Networks (WMNs); [5, 6, 7, 8] study SDN in Vehicular Ad Hoc Networks (VANETs); and [9] reviews SDN and network function virtualization in Unmanned Aerial Vehicle (UAV)-assisted systems. Furthermore, other surveys have investigated SDN control architectures. In particular, [10, 11, 12] provide extensive surveys concerning distributed SDN control architectures, but not specifically considering wireless ad hoc scenarios. Lastly, [13] provides a comprehensive analysis and performance evaluation study on both general-purpose and specialized controllers. More specifically, such specialized controllers target the Internet of Things, blockchain networks, WSNs, and VANETs.

Few surveys have explored proposals across multiple wireless ad hoc scenarios. However, such surveys have only considered a subset of the possible scenarios. In this regard, [14] reviews proposals bringing SDN in Wireless Cellular Networks (WCNs), WSNs, WMNs, and wireless local area networks. Similarly, [15] is a survey focusing on SDN in WCNs, WSNs, WMNs, and wireless home networks. Let us stress that [14, 15] do not fully cover the spectrum of wireless ad hoc scenarios and do not provide an in-depth analysis of the proposed control architectures.

1.2. Contributions

In comparison with prior surveys, we originally present a systematic and comprehensive review of SDN control architectures specifically designed for WSNs, WMNs, Mobile Ad Hoc Networks (MANETs), VANETs, and Flying Ad Hoc Networks (FANETs). To the best of our knowledge, this survey is the first one covering such a wide range of wireless ad hoc scenarios in the context of SDN. In fact, by recognizing the growing importance and relevance of software-defined wireless ad hoc networks (and since prior surveys lack comprehensive coverage), we provide a thorough analysis of the state-of-the-art literature related to the adoption of SDN in such scenarios. This survey ranges from WSNs with mostly fixed sensors and WMNs with mobile clients but stable routers to MANETs with every node freely moving on the field, VANETs with vehicles speeding through the network road, and FANETs with aerial vehicles flying in every direction.

By delving into finer details, this survey:

- Points out the main goals pursued by the adoption of SDN in wireless ad hoc scenarios, recognizing that the objective of the SDN controller is far beyond mere routing management;
- Lays out the control architectures proposed in the state-of-the-art literature, showing how the peculiar characteristics (and related goals) of a given scenario influence the architecture design;
- Discusses hybrid control architectures as a notable trend in software-defined wireless ad hoc networks.

The remainder of the survey is structured as follows. First, Section 2.1 provides a brief history of SDN, Section 2.2 describes the typical layering model representation, and Section 3 lays out most spread control architectures. The main goal is to share a consolidated terminology related to SDN before presenting its adoption in the various scenarios. Then, Section 4 introduces the target wireless ad hoc scenarios and, for each of them, details most representative proposals and related control architectures. Finally, Section 5 discusses the role of hybridization in software-defined wireless ad hoc networks and Section 6 provides conclusive remarks. To improve the readability of the survey, Table 1 lists the most recurring acronyms.

Table 1: List of the most recurring acronyms.

Acronym	Definition
BS	Base Station
CP	Control Plane
EI	Eastbound Interface
FANET	Flying Ad Hoc Network
FE	Forwarding Element
FP	Forwarding Plane
LC	Local Controller
MANET	Mobile Ad Hoc Network
MP	Management Plane
ND	Network Device
NI	Northbound Interface
OBU	On-Board Unit
RC	Root Controller
RSU	Roadside Unit
SDFN	Software-Defined Flying Network
SDMAN	Software-Defined Mobile Ad Hoc Network
SDN	Software-Defined Networking
SDVN	Software-Defined Vehicular Network
SDWMN	Software-Defined Wireless Mesh Network
SDWSN	Software-Defined Wireless Sensor Network
SI	Southbound Interface
UAV	Unmanned Aerial Vehicle
VANET	Vehicular Ad Hoc Network
WI	Westbound Interface
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network

2. SDN Background

2.1. History

In the 1990s, active networking emerged as a set of strategies to overcome relevant network issues such as network ossification, due to vertically-integrated purpose-built Network Devices (NDs) consisting of tightly coupled hardware and proprietary software, and proliferation of middleboxes (e.g., firewalls, network address translators, and load balancers). Active networks aimed to go beyond traditional packet networks by envisioning packets carrying software executed by routers/switches for manipulating packets' content.

According to [16], the metaphor of active networking was pursued through two distinct approaches, either programmable switches or capsules. The former relies on an out-of-band channel for management and an in-band channel for data transfer. Users inject programs into routers/switches through the out-of-band channel, and, in turn, routers/switches inspect data packet headers and deliver packets to programs accordingly. The latter conceives every packet (or capsule) as a piece of software. Such a so-called capsule, along its path, may pass through several routers/switches, which execute its content in an isolated environment. Both programmable switches and capsules enable custom in-network processing. On the one hand, active networks offered several significant intellectual contributions towards programmable networks. On the other hand, as [17] points out, since the active networking movement missed the "killer" application justifying such a network redesign, active networks did not see widespread deployment.

If compared with active networks, subsequent research efforts adopted a more pragmatic strategy. Instead of proposing radical paradigm shifts, e.g., programmable switches and capsules, the networking research community focused on decoupling the Forwarding Plane (FP) and Control Plane (CP), moving the control logic outside NDs, i.e., outside the core network. The FP forwards packets according to a data structure called Forwarding Information Base (FIB), while the CP controls the FP by updating the FIB through a device-specific interface. Developed at Stanford in the mid-2000s, OpenFlow [18] is the most significant implementation of an open interface between the CP and FP. The research task group behind OpenFlow was interested in enabling researchers to perform experiments in their everyday campus networks, given that researchers had no practical ways to test their ideas in real-world scenarios. Exploiting a common set of features shared by Ethernet routers and switches across different network equipment vendors, OpenFlow provides a programmatic interface for manipulating the FIB. OpenFlow-enabled routers/switches (henceforth defined as OpenFlow switches) associate an action to each flow entry and establish a secure channel with a remote controller. Similar to a traditional operating system, such a controller abstracts the underlying resources, i.e., NDs, and offers network-wide abstractions to network applications, which exploit such abstractions as building blocks to perform high-level network control and management tasks.

2.2. Definition and Layering

While there is no agreement on a standard architecture design [19, 20], the SDN paradigm mainly leads to a 3-tier network architecture (see Figure 1).

First, the control logic is removed from NDs, which become mere Forwarding Elements (FEs) that make flow-based (rather than destination-based) forwarding decisions, where a flow is defined by rules matching a set of packet header values. Accordingly, the FP (see Figure 1-down), also called "data plane", consists of wireless or wired interconnected NDs, available both in hardware (e.g., Arista 7150 Series [21], NoviSwitch 2000 Series, and Centec V350 Series [22]) and software (e.g., Open vSwitch [23], BOFUSS [24]). To mitigate the network infrastructure heterogeneity, the FP exposes

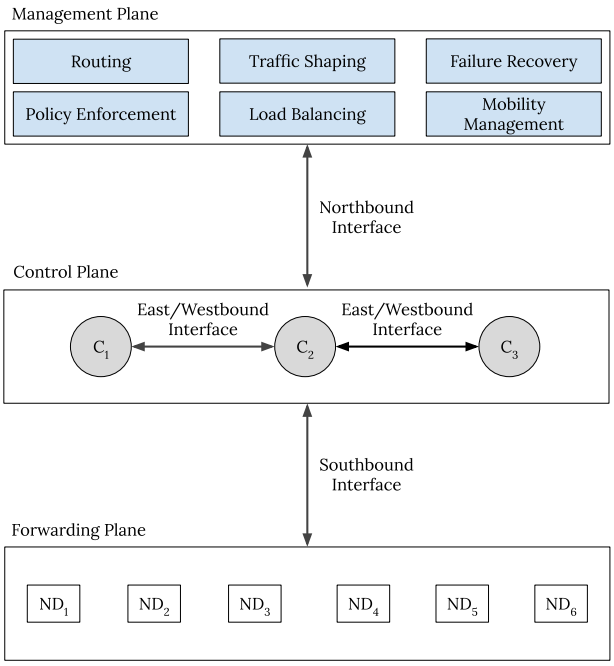


Figure 1: SDN layering and fundamental abstractions.

its resources through abstraction models, such as OpenFlow Switch Specification [25], Forwarding and Control Element Separation (ForCES) [26], YANG [27], and Management Information Base for Simple Network Management Protocol [28].

Then, the control logic is moved in an out-of-network software entity, i.e., the SDN controller in charge of dictating the behavior of the entire network infrastructure. Thus, the CP (see Figure 1-center) abstracts the underlying network infrastructure and, based on its network-wide view, provides high-level abstractions to network applications. In particular, the CP is a logically (but not necessarily physically) centralized entity that comprises one or more SDN controllers.

Finally, software applications running on top of such a controller can program the whole network. Consequently, the Management Plane (MP) (see Figure 1-up) consists of network applications that dictate the network behavior, performing sophisticated tasks such as routing, load balancing, policy enforcement, mobility management, traffic shaping, and failure recovery, among others. In turn, the CP is responsible for translating such high-level requests coming from the MP in low-level instructions for the FP. Summing up, network applications dictate ”what to do”, SDN controllers define ”how to do”, and NDs behave accordingly.

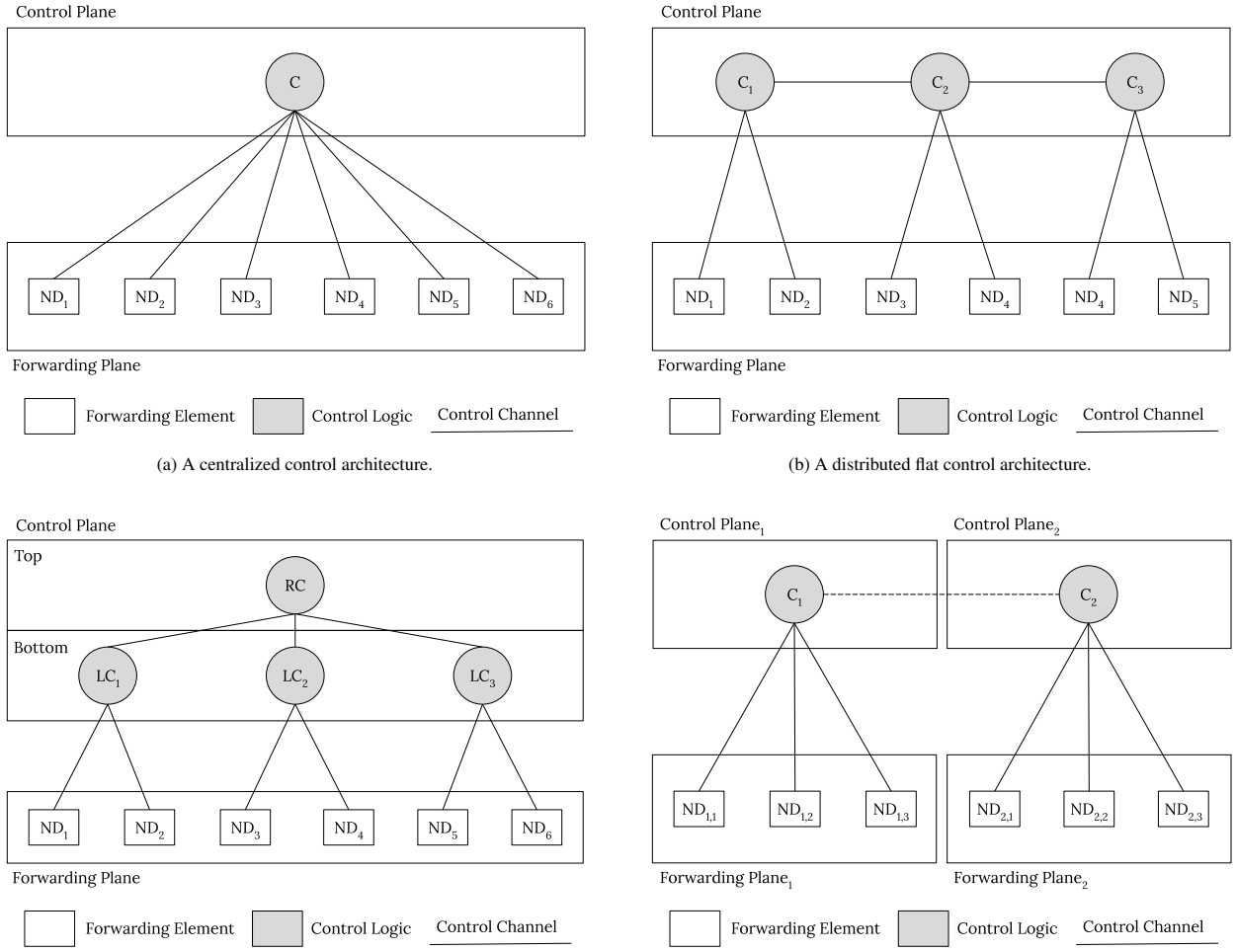
In addition, Southbound Interfaces (SIs) connect FP and CP by providing low-level communication mechanisms to the SDN controller so that it can remotely manage NDs in a vendor-agnostic fashion. We distinguish control-oriented SIs from configuration-oriented ones, where the former support the SDN controller in controlling traffic flows, while the latter in configuring NDs. While OpenFlow is the most widely adopted control-oriented SI, alternatives are ForCES [29], Protocol-Oblivious Forwarding [30], OpenState [31], and OpFlex [32]. Instead, examples of configuration-oriented SIs are ForCES [29], Network Configuration Protocol [33], Simple Network Management Protocol [34], and Open vSwitch Database Management Protocol [35].

Eastbound Interfaces (EIs) enable controller-to-controller communications, whereas Westbound Interfaces (WIs) link the SDN domain with traditional networks [36]. Examples of EIs are SDNi [37], Apache ZooKeeper [38], Advanced Message Queuing Protocol [39], and WheelFS [40]. Instead, examples of WIs are Border Gateway Protocol [41] and Path Computation Element Communication Protocol [42].

Northbound Interfaces (NIs) connect CP and MP by providing high-level services to network applications, easing network programmability. In contrast to SIs, where OpenFlow represents a de facto standard, a widely accepted NI is still missing. Therefore, the portability of network applications across heterogeneous SDN controllers is heavily limited. According to [19, 11], NIs range from ad hoc Application Program Interfaces (APIs), RESTful APIs, and APIs based on programming languages (e.g., Frenetic [43], Nettle [44], Procera [45], and Pyretic [46]).

3. SDN Control Architectures

By advocating for a clear FP and CP decoupling, SDN moves control logic from NDs to controllers. This leads to a logically centralized CP responsible for monitoring, controlling, and configuring the FP. We categorize control architectures, i.e., how controllers interact one each other, in centralized, distributed, and federated (plus hybrid as a cross-category attribute). In addition, we further specialize distributed control architectures in flat (each controller acts as a peer) and hierarchical (controllers are hierarchically organized). The rest of this section discusses design patterns (see Figure 2 and Figure 3) of such control architectures and provide notable examples of how well-known SDN controllers fit the taxonomy we propose.



(a) A centralized control architecture.

(b) A distributed flat control architecture.

(c) A distributed hierarchical control architecture.

(d) A federated control architecture.

Figure 2: Control architectures for software-defined networks.

3.1. Centralized

As Figure 2a shows, a centralized control architecture consists of a single controller responsible for every ND. Although a single controller represents the optimal design choice in terms of simplicity, it intrinsically implies scalability and resiliency issues [47]. Regarding scalability, the incoming requests may overwhelm the controller as the network grows. In fact, when a packet goes through a ND without matching any flow rule, the ND sends a flow request to the controller, which replies with the generated flow rule. Therefore, centralized control architectures, where a single controller takes over the flow setup process for the whole network, may easily result in bottlenecks, additional delays, and thus limited network scalability. In this regard, a workaround is the adoption of proactive strategies, which inject flow rules into NDs in a proactive fashion, mitigating the controller load consequently. About resiliency, a single controller represents a Single Point of Failure (SPoF), potentially breaking the network when NDs cannot serve incoming flows due to the controller's unreachability. Hybrid strategies (see Section 3.4) may overcome such lack of resiliency by (temporarily) falling back to traditional routing protocols on NDs that cannot reach the controller.

Examples of controllers based on centralized control architectures are NOX [48], Maestro [49, 50], Beacon [51], NOX-MT [52], Floodlight [53], POX [54], Ryu [55], POF Controller [56], Meridian [57], and Rosemary [58].

3.2. Distributed

To address the scalability and resiliency issues raised by centralized control architectures, the SDN commu-

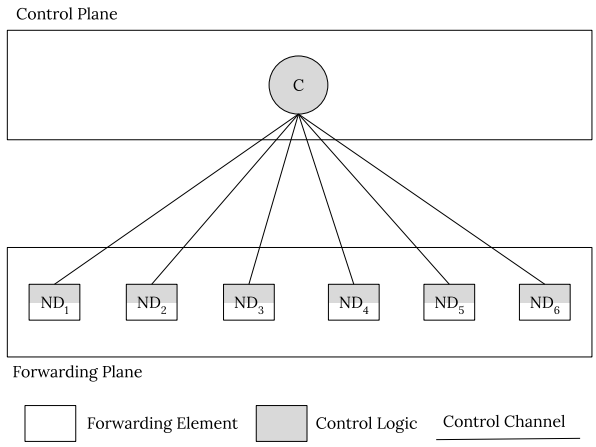


Figure 3: A hybrid centralized control architecture.

nity started proposing distributed control architectures where multiple controllers interact one each other to manage the network infrastructure. On the one hand, multiple controllers can improve overall performance (by fairly sharing the network load), resiliency (by avoiding a SPoF), and scalability (by on-demand deploying additional controllers if necessary). On the other hand, such controllers must accomplish coordination and synchronization tasks to maintain a global and (eventually) consistent network-wide view [59, 60, 61]. In addition, distributed architectures make extremely challenging the controller placement problem [62], which consists in optimizing the number of deployed controllers and their placement in the network topology.

According to the taxonomy, we further distinguish distributed control architectures in flat and hierarchical. As Figure 2b shows, flat control architectures involve multiple controllers managing non-overlapping subsets of NDs. Note that controllers based on flat architectures interact with each other to share a complete knowledge of the whole network, while each controller directly manages only a subset of NDs. One of the first published proposals designing a flat control architecture is HyperFlow [63], which proposes multiple controllers sharing the same consistent network-wide view, i.e., the state is fully available in each controller. HyperFlow is implemented as an application running on top of NOX. Other examples of controllers based on flat control architectures are Onix [64], ONOS [65], and OpenDayLight [66].

As Figure 2c shows, hierarchical control architectures assume a multi-layer CP. For example, Kandoo [67] proposes a two-layer hierarchical CP, where the bottom layer consists of multiple Local Controllers (LCs) that do not communicate with each other, while the top one comprises a single Root Controller (RC). LCs hold local-domain network views and handle frequent and small (mice) flows, while the RC takes over infrequent and huge (elephant) flows by taking advantage of its network-wide view. Given that mice flows are more than elephant ones, LCs substantially reduce the burden of the RC. Other examples of controllers based on hierarchical control architectures are Orion [68], B4 [1], and Espresso [69].

In conclusion, it is worth stressing the main difference between flat and hierarchical control architectures. In flat control architectures, all controllers share the same network-wide view ensuring optimal decision-making but leading to non-trivial issues in guaranteeing state consistency. In hierarchical control architectures, LCs, whose visibility is limited to a local network domain, provide regular updates to RCs, which aggregate such information. Therefore, only RCs (i.e., a subset of controllers) hold such a network-wide view, whereas LCs may make suboptimal decisions.

3.3. Federated

As Figure 2d shows, a federated control architecture assumes loosely coupled controllers (managing non-overlapping network domains) that do not share a global network-wide view. Instead, each controller shares a summary of its own state (i.e., typically a high-level abstraction of its local network view) or requests specific information to remote controllers for enabling end-to-end services. Since controllers do not share the whole picture of what they know, federated control architectures naturally fit inter-organizational scenarios, in contrast to distributed architectures that best suit intra-organizational ones. Note that the absence of a global network-wide view reduces complexity since there are no consistency strategies in place, decreases control network traffic because controllers do not share their whole state, and improves privacy thanks to the sovereignty of controllers over information sharing. An example of federated control architecture is DISCO [70].

3.4. Hybrid

The peculiarity of hybrid control architectures is to maintain a certain degree of control logic within NDs, as Figure 3 shows. Therefore, such hybrid solutions do not strictly follow the SDN canonical approach, which advocates for NDs acting as mere FEs. In the taxonomy we propose, ”hybrid” is an orthogonal attribute to centralized, flat, hierarchical, and federated.

As mentioned in Section 3.1, a typical scenario involving hybrid control architectures is when NDs reenable their local control logic (e.g., by falling back to traditional routing protocols) to face emergency circumstances (e.g., the controller's unreachability). More radical hybrid approaches take advantage of hybridization not only when unexpected events occur but on a regular basis, e.g., the controller computes forwarding weights and disseminates them to NDs that take into account such weights but make (semi-)autonomous decisions. Another example is the case of in-network processing, where NDs locally process network traffic by dropping, delaying, or aggregating packets.

Let us note that hybridization positively impacts not only on resiliency, e.g., NDs keep operating successfully despite controller failures, but also on scalability, e.g., in-network processing allows reducing the packets traversing the network.

4. Wireless Ad hoc Scenarios

Recently, several state-of-the-art solutions proposed the adoption of SDN in wireless ad hoc scenarios. This section lays out the most representative proposals, pivoting the presentation around control architectures and their objectives. In this regard, Table 2 presents the wireless ad hoc scenarios surveyed in this work, i.e., Wireless Sensor Networks (WSNs), Wireless Mesh Networks (WMNs), Mobile Ad Hoc Networks (MANETs), Vehicular Ad Hoc Networks (VANETs), and Flying Ad Hoc Networks (FANETs), each described through seven defining characteristics:

1. Node velocity, i.e., how fast nodes move, ranging from extremely low (e.g., sensors that are primarily static) to extremely high (e.g., aerial vehicles);
2. Node mobility, i.e., movement patterns followed by nodes, depicting whether nodes move unpredictably, constrained by the environment (e.g., road network), or according to pre-planned routes (e.g., flight plans);
3. Node energy, i.e., how much energy is available onboard, ranging from heavily limited (e.g., sensors) to abundant (e.g., vehicles);
4. Network infrastructure, i.e., whether the scenario is infrastructure-less or based on pre-placed communication infrastructure;
5. Network interoperability, i.e., if there are heterogeneous interconnected networks that can cooperate;
6. Medium congestion, i.e., whether or not the scenario typically suffers from high node density potentially congesting the wireless medium (e.g., thousands of sensors deployed in a limited area or vehicles in dense urban areas);
7. Safety applications, i.e., if delay-sensitive applications requiring reliable and low-latency communications to guarantee safety characterize the scenario (e.g., applications seeking to prevent road accidents).

Additionally, we detect the key concern (marked with an asterisk) for each considered scenario.

Table 3, 4, 5, 6, and 7 list the proposals included in this survey, specifying for each one the control architecture (i.e., centralized, flat, hierarchical, or federated), whether such an architecture is hybrid (according to the definition we provide in Section 3.4), the ready-made controller on which it relies on (if any), and its main distinctive features.

The rest of this section presents the scenarios mentioned above sorted according to the typical node velocity degree, starting from WSNs (given that sensor nodes are primarily static) and ending with FANETs (characterized by the highest degree of spatial mobility). We build up the analysis of the solutions targeting a particular scenario as follows:

- First, we provide a comprehensive description of the target scenario;
- Then, we explain the rationale behind the adoption of the SDN paradigm in such a scenario, considering not only routing aspects but also other scenario-specific peculiarities;
- Next, we systematically describe the proposals targeting such a scenario, arranging such proposals according to their control architectures, i.e., centralized, flat, hierarchical, and federated;
- Finally, we provide suggestions for researchers according to the distinctive features and design patterns proposed in the surveyed literature.

We believe that such an investigation discloses relevant patterns in designing future control architectures for software-defined wireless ad hoc networks.

4.1. Wireless Sensor Networks (WSNs)

A WSN consists of sensor nodes provided with sensing, processing, and communication capabilities. Once deployed in the target environment, sensor nodes start collecting data about the surrounding. Usually, sensor nodes are densely deployed within or nearby the phenomenon of interest. By exploiting the onboard processor, sensor nodes can partially process the gathered data. Then, sensor nodes act both as data sources and routers, cooperating to route data to a sink, i.e., the node in charge of collecting every data. Communica-

Table 2: Defining characteristics of wireless ad hoc scenarios.

Characteristic	WSN	WMN	MANET	VANET	FANET
Node velocity	Extremely low	Medium	Medium	High	Extremely low/high (*)
Node mobility	Primarily static	Unpredictable	Unpredictable	Constrained by road network	Pre-planned
Node energy	Heavily limited (*)	Limited	Limited	Abundant	Limited/Abundant
Network infrastructure	Infrastructure-less	Infrastructure-less	Infrastructure-less (*)	Infrastructure-based	Infrastructure-less
Network interoperability	No	Yes (*)	No	Yes	No
Medium congestion	Yes	No	No	Yes	No
Safety applications	No	No	No	Yes (*)	Yes

(*) Key concern

tions between sensor nodes are typically wireless, ranging among radio, infrared, and optical communication links. Table 2 (second column) states the defining characteristics of WSNs.

Since WSNs are infrastructure-less, sensor nodes must provide self-organizing capabilities for delivering packets to the sink. When the network topology changes, sensor nodes must quickly react and adapt themselves to the new conditions. Such adaptation involves battery-consuming tasks, such as packet rerouting and network reorganization. However, sensor nodes are resource-constrained devices, heavily limited in terms of battery lifespan and computational capabilities. In addition, since sensors are prone to failures, glsplwsn must be fault-tolerant, i.e., they must keep working smoothly despite the failure of one or more sensors. Many factors might cause sensor failures, such as particular environmental conditions that (temporarily) damage one or more sensor nodes. The number of sensor nodes deployed in the field might be dramatically high, even in the order of thousands. Such a tremendous number of sensors raises issues regarding scalability, topology maintenance, and medium congestion. Lastly, sensors might not present a globally unique identifier.

4.1.1. SDWSN: Objectives and Control Architectures

A Software-Defined Wireless Sensor Network (SDWSN) leverages the SDN paradigm into a WSN [2]. The adoption of SDN principles may mitigate the main issues affecting WSNs. In SDWSNs, the most energy-intensive tasks are moved from sensor nodes to the controller. As a result, sensors consume less energy with an enhancement in terms of overall network lifespan. For example, sensor nodes do not run distributed algorithms for making routing decisions, but the controller instructs them on packet forwarding. Therefore, the burden of routing only regards the controller that runs on commodity server technology, where no energy constraints are in place. Also, network management and configuration of traditional WSNs may become tedious, especially depending on the environment in which sensors are deployed. Instead, SDWSNs promote the programmability of the underlying network infrastructure by transferring the control logic from sensor nodes to the controller. In such networks, sensors become mere FEs, which can be remotely configured through the APIs provided by the controller. Table 3 sums up the surveyed proposals concerning SDWSNs.

[71] proposes a *hybrid centralized* control architecture providing a single unified paradigm to program how sensor nodes behave, simplify policy enforcement, and supply developers with a standard set of abstractions for building applications. Typically, sensor nodes are vendor-specific devices whose software and hardware are tightly coupled. Such a design choice makes WSNs dramatically inflexible. In fact, a WSN usually consists of several sensor nodes provided with different (proprietary) vendor-specific software. [71] distinguishes three different categories of nodes, i.e., a controller node, which acts as a traditional controller, intermediate nodes, which receive instructions from the controller and hold the flow tables accordingly, and leaf nodes, which behave as classic sensor nodes. Specifically, leaf nodes gather information about the surroundings and send sensed data to intermediate nodes. Beyond mere forwarding, intermediate nodes also perform additional tasks. For example, the controller can instruct an intermediate node to forward packets only if a specific field value is greater than a given threshold. Additionally, intermediate nodes can collect data from different sensor nodes and then apply a particular aggregation function, e.g., averaging. Therefore, such an architecture substantially reduces network traffic by only transmitting relevant information. In fact, by discarding or aggregating messages in the early stages, the overall number of sent messages is substantially reduced. Also, sensor nodes do not have to allocate computational resources for making routing decisions with a consequent saving of energy.

[72] proposes a *hybrid centralized* control architecture to lower power consumption and improve bandwidth utilization in WSNs. In fact, sensor nodes traditionally run distributed routing algorithms that lead to suboptimal routing decisions (given that no one knows

Table 3: Control architectures for SDWSNs.

Proposal	Architecture	Hybrid	Based on	Distinctive Feature
[71]	Centralized	Yes	-	In-network processing
[72]	Centralized	Yes	-	Fallback routing mechanism
[73]	Flat	No	-	Control traffic reduction and locality-based controller placement
[74]	Flat	Yes	-	Control traffic reduction, duty cycle support, and in-network processing
[75]	Hierarchical	No	[74]	Fragmentation-based controller placement

the whole network topology), unbalanced network load, and slow recovery from failures. In the SDN-based solution proposed in [72], sensor nodes run both an OpenFlow agent (to interact with the controller) and an Ad hoc On-Demand Distance Vector (AODV) daemon [76]. Such a design choice implies a higher computational resource consumption as compared to solutions in which the control logic is completely removed from NDs. However, the AODV daemon locally running on sensor nodes makes the entire network more resilient to failures. In particular, the AODV daemon activity is limited to the following two circumstances. First, immediately after the deployment phase, sensor nodes do not know how to reach the controller (routing tables are still empty). In this circumstance, sensor nodes exploit the AODV routing protocol to establish a connection with the controller. Second, when the controller becomes unreachable, sensor nodes rely on the AODV routing protocol for delivering packets successfully without controller instructions.

[73] proposes a *flat* control architecture to address the following four challenges. First, sensor nodes impose an in-band control, in contrast to the out-of-band control typically adopted in wired networks. In fact, sensor nodes are usually provided with only a radio to either transmit or receive in a single frequency at a time. Second, WSNs are affected by an intrinsic higher latency than wired networks. Such an architecture decreases the latency by placing multiple controllers closer to data sources (locality principle [77]). Third, control messages should fit in one link-layer frame that is substantially smaller than in wired networks, e.g., 127 bytes according to the IEEE 802.15.4 standard. Fourth, sensor nodes are resource-constrained devices, especially in terms of energy. To leverage SDN in WSNs, [73] targets TinyOS compatible platforms. At the network startup, sensor nodes rely on the Collection Tree Protocol [78] to discover how to reach the controller. In the case of multiple controllers, sensor nodes join the nearest one. Since both control and data traffic pass through the same channel, [73] distinguishes network packets accordingly. Once properly classified, packets may be dropped or forwarded.

[74] proposes a *hybrid flat* control architecture reducing the amount of control traffic, i.e., messages exchanged between the controller(s) and sensor nodes, as well as make sensor nodes programmable as finite state machines. Such a solution supports energy efficiency by enabling duty cycle (i.e., a mechanism that allows to turn off sensor nodes periodically) [79] and data aggregation [80]. Compared to traditional OpenFlow-like solutions, the programmability of sensor nodes as finite state machines enables the enforcement of a broader policy range. For example, it is possible to define a policy dropping packets from a specific source if the temperature value in packets coming from a different source is lower than a given threshold. Thus, sensor nodes need to maintain a state [31], whereas OpenFlow-like solutions are usually stateless.

[75] proposes a *hierarchical* control architecture providing a reliable, scalable, and efficient control system for SDWSN. By exploiting the concept of fragmentation, sensor nodes are clustered in segments, where a segment consists of several sensor nodes and a sink. Each segment is managed by a dedicated LC deployed nearby its own segment to reduce the distance from the sink. A global controller oversees the whole network, while LCs only know their own segment. In particular, a LC communicates with sensor nodes under its responsibility through the sink. Given that LCs are close to their segment, the control system benefits from lower latency and better responsiveness. LCs update the global controller, which is the only control entity holding a network-wide view. The two-level control architecture, along with the fragmentation technique, brings several advantages in terms of resiliency. In fact, if the global controller becomes unreachable, LCs can keep making decisions regarding the segment on which they are in charge. However, decisions requiring global network knowledge might cause temporal disruptions. Also, each LC is completely independent of other controllers. In fact, there is no exchange of information among LCs with a consequent reduction of the overhead network costs.

4.1.2. SDWSN: Lessons Learned

The key concern about WSNs is to lower energy consumption because sensor nodes are resource-constrained devices, deeply limited in power availability. Therefore, computational-intensive tasks burden such devices, dramatically shortening their lifespans.

Adopting SDN in WSNs allows moving complexity from sensor nodes to the controller, which dictates the network behavior. Such a network control logic centralization naturally fits with the purpose of lengthening sensor lifespans. In fact, the controller can compute optimal routing decisions by taking advantage of its centralized network-wide view. Thus, sensor nodes do not locally execute a distributed routing algorithm (which may lead to suboptimal decisions and is a resource-intensive task for such heavily limited devices) but put into action the instructions received by the controller. Therefore, SDWSNs extend sensor lifespans by relieving sensor nodes from routing decision-making and optimally routing traffic among such nodes. Moreover, SDN also improves the programmability of sensor nodes, which are traditionally vendor-specific and tedious to configure.

As mentioned above, the key concern of researchers in WSNs is to lower energy consumption. In this regard and according to the state-of-the-art analysis we provided, researchers should be aware of three distinctive features (see Table 3): in-network processing, duty cycles, and control traffic reduction. Specifically, [71, 74] provide in-network processing, i.e., sensor nodes can use onboard control logic to process packet payloads (e.g., by averaging consecutive temperature values or dropping values as of little application interest) as they traverse the network. Therefore, the overall network traffic decreases and, accordingly, the energy consumed by sensor nodes for forwarding. [74] also supports duty cycles, i.e., the ability to turn off sensor nodes' radio interfaces. Lastly, [73, 74] focus on control traffic reduction to mitigate the network overhead introduced by the adoption of SDN.

Other distinctive features proposed by researchers, but not primarily related to the key concern, regard controller placement and fallback routing. In particular, [73] relies on the locality principle to place controllers optimally to reduce latency. Similarly, [75] provides a fragmentation-based controller placement to improve responsiveness and resiliency. Finally, [72] introduces a fallback routing mechanism to enable communications when the controller becomes unreachable.

Let us note that we classify SDWSN solutions supporting in-network processing as *hybrid*. In fact, such solutions consist of sensor nodes that keep control logic to process packet payloads based on an aggregation function. Therefore, sensor nodes are not mere FEs as in traditional SDN. Especially in SDWSNs, in-network processing is a critical feature to improve overall performance and energy efficiency.

4.2. Wireless Mesh Networks (WMNs)

A WMN typically consists of mesh routers and mobile mesh clients [81]. A mesh router may be equipped with multiple wireless network interfaces (possibly) supporting different wireless access technologies. Consequently, a mesh router can bridge different types of networks, such as WCNs, WSNs, and the Internet. Both mesh routers and mesh clients route packets, but mesh clients are usually provided with a single wireless network interface. The WMN backbone consists of multiple self-organizing routers forming a mesh. Typically, mesh routers communicate with each other using long-range communications and with clients using a distinct wireless interface. Mesh clients sharing the same wireless interface can interact with each other in a peer-to-peer fashion. Table 2 (third column) states the defining characteristics of WMNs.

Usually, WMNs consists of multiple mesh routers forming the backbone network and several types of mesh clients organized in subnetworks, where mesh routers enable interoperability between such a wide range of heterogeneous wireless access technologies. The minimal degree of node mobility characterizing mesh routers makes WMNs more robust than traditional wireless ad hoc networks. Moreover, mesh routers are not affected by power consumption constraints. However, mesh clients, which are energy-constrained devices, move unpredictably.

4.2.1. SDWMN: Objectives and Control Architectures

A Software-Defined Wireless Mesh Network (SDWMN) is a WMN that adopts the SDN paradigm. As pointed out by [4], WMNs benefit of SDN as follows. First, WMNs can integrate several and heterogeneous wireless networks. Accordingly, the controller can manage the required gateways and handle network traffic flows across backbone-integrated subnetworks. Through its network-wide view, the controller can route traffic optimally and manage node mobility efficiently. Second, although mesh routers are relatively stable, mesh clients may affect the network topology quite frequently, joining and leaving subnetworks in the mesh as they move in the field. In such cases, the controller must redirect network traffic flows accordingly. Table 4 sums up the surveyed proposals concerning SDWMNs.

Table 4: Control architectures for SDWMNs.

Proposal	Architecture	Hybrid	Based on	Distinctive Feature
[82]	Centralized	No	-	SDR module and traffic orchestration algorithms
[83]	Centralized	Yes	POX	Fallback routing mechanism
[84]	Flat	No	Ryu	SD-OLSR protocol and controller/switch-initiated handoff
[85]	Flat	No	POX	LLDP-based QoS monitoring
[86]	Federated	Yes	[87]	Opportunistic controller selection and dynamic controller election

[82] proposes a *centralized* control architecture taking advantage of the SDN paradigm in WMNs while considering the traffic characteristics typically affecting wireless networking scenarios. In particular, [82] designs three spectrum allocation and scheduling algorithms based on a weighted throughput optimization problem to optimally leverage control and data traffic in WMNs. The controller monitors the whole network, makes routing decisions, schedules both data and control traffic, and configures spectrum resources. As in traditional SDN-based solutions, mesh routers become mere FEs that apply the flow rules pushed by the controller. In addition, mesh routers are equipped with a software-defined radio module [88] for tuning radio transceivers within their own radio spectrum. The controller interacts with such tuning modules residing in the mesh routers to push configuration regarding the radio frequency.

[83] proposes a *hybrid centralized* control architecture to provide a robust control framework coping with emergency circumstances, such as network partitions and controller failures. Such a solution designs an in-band control network managed by Optimized Link State Routing (OLSR) daemons [89] running locally on mesh routers. In particular, OLSR daemons learn the control network topology and set up control rules accordingly. Also, emergency-specific rules come into action when the controller becomes unreachable. In particular, mesh routers periodically check the controller status through liveness probes. If failures occur, mesh routers activate a sort of emergency mode, substantially entrusting the whole network traffic routing to OLSR daemons. As soon as the controller becomes reachable again, mesh routers promptly leave the so-called emergency mode by forcing packet-in data traffic flows to pass through the controller (that is responsible for appropriately routing such flows) and by limiting OLSR daemons to only manage the control traffic.

[84] proposes a *flat* control architecture providing a self-configuration framework tailored to WMNs. Specifically, [84] presents two distinct schemes to enable node self-configuration: controller-initiated handoff and switch-initiated handoff. The former defines the following three controller roles toward a switch: master, i.e., the controller can query and modify the state of the switch, slave, i.e., the controller can query but not modify the state of the switch, and equal, i.e., same as the master role. Note that only a single controller can play the master role towards a switch, but multiple controllers can play an equal role towards the same switch (equal is also the role assigned by default when a switch connects to a controller). Assuming that such controllers hold a global network-wide view, assigned roles may be changed based on the controller selection algorithm running on the CP side. The latter defines a procedure that is initiated on the switch side. In particular, switches obtain information via the SD-OLSR protocol. Such a protocol (an extension of the OLSR protocol [90]) supports resource discovery of SDN-specific resources under in-band control by helping nodes in sharing information and configuring themselves according to the ongoing changes affecting the network. Then, switches connect to the nearest controller with an equal role. Even when multiple controllers are available, switches can connect to a single controller at a time (differently from the controller-initiated handoff).

[85] proposes a *flat* control architecture providing Link Layer Discovery Protocol (LLDP)-based topology discovery and Quality of Service (QoS) monitoring. Such a solution targets wireless multi-hop networks, especially focusing on WMNs characterized by multiple geographical-dislocated or administrative domains. Each controller controls a distinct network domain and communicates in a single-hop fashion with the mesh routers under its responsibility. Interactions among different controllers (i.e., inter-domain communications) take place via the Advanced Message Queuing Protocol [39]. Specifically, such controllers publish/subscribe on different topics to exchange information and build a network-wide view. Therefore, the topology discovery scheme consists of two phases. First, controllers build up-to-date views of the network domain under their responsibility. Then, they aggregate neighboring domains' topology information. Additionally, such a

topology discovery scheme also allows collecting QoS information. In fact, [85] enriches LLDP messages with four optional fields with QoS properties (i.e., bandwidth, packet loss, delay, and jitter) subsequently propagated through the above-mentioned topology discovery scheme.

[86] proposes a *hybrid federated* control architecture based on a middleware that copes with QoS management in spontaneous networks, i.e., WMNs where mesh clients not only create their own multi-hop networks but also become part of the network infrastructure itself. In traditional WMNs, infrastructure network administrators take care of configuring mesh routers but mesh clients typically work in a best-effort manner. This prevents network-wide coordination. On the contrary, [86] envisions a networking scenario in which both mesh routers and clients synergically share information regarding the resources made available by easing to ensure the demanded QoS requirements. In such a scenario, a WMN consists of multiple so-called islands (i.e., subsets of nodes residing at a small distance and sharing common interests), each managed by a single controller. Controllers autonomously manage the traffic related to their own island, while interact with each other in a federated manner to make QoS management decisions about inter-island traffic flows. WMN nodes periodically check the availability of their controller and can migrate to another one that represents a better alternative. Note that each WMN node running the proposed middleware can potentially become a controller. A node takes the controller role within an island based on a node election protocol that considers some key factors, such as computing capabilities, energy, and incentive mechanisms. In particular, WMN nodes can dynamically cooperate by electing a new controller under some circumstances, e.g., in case the current one runs out of battery.

4.2.2. SDWMN: Lessons Learned

The key concern about WMNs is to enable interoperability between multiple and (potentially) heterogeneous networks forming the mesh. Specifically, mesh routers are responsible for allowing communications between the backbone-integrated networks and supporting the mobility of mesh clients across them.

Adopting SDN in WMNs allows enforcing fine-grained policies between the backbone-integrated networks participating in the mesh. Note that such networks may belong to different administrative domains. As a result, each domain may vary in QoS, security, and privacy requirements. In some cases, such requirements may even be conflicting. In SDWMNs, since the controller holds a network-wide view, it can program the network behavior to enforce policies (and mediate conflicts) according to the requirements demanded by the domains involved in the mesh. Furthermore, the controller can manage traffic flows traversing different subnetworks by dynamically configuring subnet gateways along their paths. Lastly, the controller can support seamless communications of mesh clients while moving across different domains.

As mentioned above, the key concern of researchers about WMNs is to enable interoperability. In this regard and according to the state-of-the-art analysis we provided, researchers should focus on the distinctive features (see Table 4) for acquiring granular control over network configuration to deal with heterogeneous wireless technologies and multiple administrative domains. Accordingly, [85] proposes LLDP-based QoS monitoring primarily intended for WMNs characterized by multiple geographical-dislocated o administrative domains. [82] designs three traffic orchestration algorithms to use wireless spectrum efficiently. To do so, [82] jointly combines SDN and software-defined radio to tune transceivers within the allocated radio spectrum. [84] provides two handoff schemes and the SD-OLSR protocol to help SDWMNs configure themselves according to the ever-changing network conditions. Lastly, [86] proposes dynamic controller election and opportunistic controller selection to cope with mobile nodes while ensuring QoS requirements.

Another distinctive feature proposed by researchers, but not primarily related to the key concern, is fallback routing. Specifically, [83] proposes a fallback routing mechanism based on OLSR to cope with emergency circumstances, such as network partitions and controller failures.

Let us note that since WMNs typically have multiple domains, a controller per domain may be a logical choice for researchers. This leads to distributed (i.e., flat or hierarchical) or federated control architectures. As stated in Section 3.3, the former naturally fit intra-organizational scenarios, whereas the latter inter-organizational ones.

4.3. Mobile Ad Hoc Networks (MANETs)

A MANET consists of autonomous mobile nodes communicating through wireless links without relying on a pre-placed network infrastructure or centralized administration [91]. Because of the absence of pre-existing network infrastructure, nodes must self-provide routing capabilities. By acting as independent routers, nodes cooperate to deliver packets in a multi-hop fashion. Multi-hop routing enables nodes to receive packets from senders located beyond their transmission range.

When a sender and a receiver cannot directly communicate, intermediate nodes operate as relays and carry out packets between the source and the destination. Consequently, MANETs can be defined as infrastructure-less multi-hop wireless ad hoc networks. Table 2 (fourth column) states the defining characteristics of MANETs.

MANET nodes are free to move everywhere anytime, resulting in frequent and unpredictable changes in network topology. Accordingly, network partitions cannot be considered extraordinary events, and, as the network topology changes, nodes must adapt their routing decisions accordingly. Such a dynamic nature makes extremely challenging the design of routing protocols that cope with the ever-changing network conditions efficiently. Finally, MANET nodes are battery-powered devices provided with heterogeneous hardware characterized by variable capabilities, ranging from constrained devices with limited computational and memory capabilities to relatively more powerful devices such as laptops.

Disaster relief operations represent a typical deployment scenario for MANETs, where emergency personnel cannot rely on the pre-existing network infrastructure, which may be partially or totally destroyed. Therefore, enabling communications between rescuers is of paramount importance because information sharing improves the situational awareness of rescuers and their effectiveness in the field.

4.3.1. SDMAN: Objectives and Control Architectures

With the term Software-Defined Mobile Ad hoc Networks (SDMANs) we indicate MANETs that take advantage of SDN. Traditionally, MANETs rely on purely distributed approaches for supporting multi-hop routing. Each node builds its own local network view by exploiting information coming from its neighbors. Therefore, the decision-making process performed by each node is affected by a lack of global network-wide view, which may lead to suboptimal decisions. The highly distributed nature of MANETs may also affect the enforcement of potentially competing network policies. For example, two distinct nodes may target the same large-bandwidth path for satisfying specific QoS requirements. However, if competing nodes select the same large-bandwidth path (the optimal decision based on their local view), they may make an overall suboptimal decision if the same large-bandwidth path does not support the demanded QoS requirements for both nodes at the same time.

By relying on a global network-wide view, the SDN controller can optimally manage the ever-changing network resource demand. However, such a logic centralization does not naturally fit with purely distributed contexts. Therefore, the adoption of SDN principles in MANETs must carefully consider the following [96]. First, MANET nodes may continuously join and leave the network. On the one hand, the controller must adapt its decisions to the dynamic network environment promptly. On the other hand, the controller must collect data about the network status in a lightweight manner (without saturating the whole network bandwidth). Second, the controller must be flexible, i.e., traffic engineering and policy enforcement must be intended as never-ending processes, where the controller continuously (re)allocates the resources based on current state and requirements. Table 5 sums up the surveyed proposals concerning SDMANs.

[92] proposes a *hybrid centralized* control architecture realizing an SDN-augmented MANET swarm node platform. Such swarm nodes participate in two-tier networking, where the underlay is MANET-based while the overlay is SDN-based. The MANET-based underlay takes advantage of the MANET Swarm Networking Protocol to assign the proper role (i.e., peer, node, and head) to nodes participating in a so-called swarm. As soon as a node starts the networking, it becomes a peer. Then, following a head selection algorithm (based on some key metrics, such as the battery level), a node takes the head role while the others take the node role. Note that the MANET-based underlay is purely distributed, characterized by self-organized nodes. In the SDN-based overlay, the head node enables controller capabilities. Since the head node is not known a priori, each node is provided with the adequate software packages to potentially become a controller. Once the SDN-based overlay is ready, the head node (i.e., the controller) starts monitoring the network. The topology discovery relies on an extension of LLDP (proposed by [92]) enhanced with custom metrics that take into account the ad hoc nature of the underlying networking layer (e.g., bandwidth, delay, and battery). [92] also proposes an OpenFlow extension supporting the control of physical actions. Specifically, such an extension allows controlling swarm nodes' actions about mobility, surveillance, and monitoring.

[93] proposes a *hybrid centralized* architecture exploiting stateful extensions of SDN protocols [31, 97] and leveraging local agents capable of making autonomous decisions when specific circumstances occur. Such agents run locally on NDs by disseminating to each other messages concerning link-state information, basically mimicking how fully distributed routing protocols work. Notwithstanding, the message dissemination relies on a mechanism preventing flood-

Table 5: Control architectures for SDMANs.

Proposal	Architecture	Hybrid	Based on	Distinctive Feature
[92]	Centralized	Yes	Floodlight	Dynamic controller election and LLDP-based QoS monitoring
[93]	Centralized	Yes	POX	Sub-based link-state dissemination and autonomous decision-making at FP
[94]	Flat	Yes	ONOS	SR-based traffic scheduling
[95]	Hierarchical	Yes	-	Fallback routing mechanism and control functions in code fragments

ing, where local agents only receive link-state information from neighbor nodes by subscription. Based on the (un)availability of routing paths, local agents can unilaterally change their behavior by applying so-called backup forwarding rules instead of the flow rules installed by the controller. The transition permitting the dynamic selection between controller-provided flow rules and backup forwarding rules is enabled by a stateful extension of OpenFlow. Through such an extension, NDs can dynamically select which rule to apply depending on the current network state, e.g., available routing paths and state of the links. However, whenever possible NDs should behave as dictated by the controller to take full advantage of its network-wide view.

[94] proposes a *hybrid flat* architecture addressing the reliability problem without simply recurring to redundancy. In fact, the unpredictable nature of MANETs frequently causes connectivity issues, preventing stable communication channels between controllers and NDs. The adaptive controller placement (i.e., dynamic controller deployment to ensure living connections between the CP and FP) mitigates the before-mentioned connectivity issues. However, it does not substantially solve the reliability problem that would theoretically require a controller per ND to face the worst-case scenario. Moreover, such adaptivity intrinsically leads to supplementary network costs, potentially delving into performance degradation. Accordingly, [94] suggests a scheme where NDs do not rely on flow rules installed by the controllers to make routing decisions. Instead, controllers periodically broadcast lists of segment labels (i.e., intermediate nodes that packets should traverse) to the NDs, similar to Segment Routing (SR) [98]. Then, based on a distributed routing protocol running locally, NDs make autonomous decisions on routing packets from one segment to the next one in the list. Nonetheless, NDs can neglect the guidelines (i.e., the lists of segment labels) provided by the controllers if the distributed routing protocol does not find the target node and controllers are not reachable to update the provided guidelines.

[95] proposes a *hybrid hierarchical* control architecture supporting modern military communications systems during coalition tactical operations, which usually take place in areas where military personnel cannot rely on pre-existing network infrastructure. Such operations may involve multiple actors, even belonging to different nations, by demanding a particular kind of mission data dissemination based on the "need-to-know" principle. Such a principle enables information circulation but prevents sensitive information sharing beyond the necessary. [95] designs a two-tier hierarchy where LCs monitor their respective domain by regularly updating the RC, which aggregates a network-wide view and dictates operation policies. In turn, LCs (primarily working as bridges between RC and NDs) disseminate such policies towards the NDs. However, LCs can take complete control over their domain when specific circumstances occur, e.g., the RC becomes unreachable or applications impose real-time requirements. Benefiting from control logic centralization but coping with ever-changing network conditions, the FP consists of NDs that make autonomous decisions if necessary. Specifically, [95] explores two distinct approaches. The former suggests backing up the centralized SDN control with a legacy MANET protocol, which takes over the routing decision-making when NDs cannot connect with the controller. The latter envisions the ability to precompute and push code fragments to NDs that locally execute them. Note that the first approach takes advantage of traditional distributed routing algorithms only when emergency circumstances occur, while the second one implies delegating part of the control logic to NDs consistently.

4.3.2. SDMAN: Lessons Learned

The key concern about MANETs is the absence of pre-placed network infrastructure. Therefore, MANET nodes must enable communications in a self-organized fashion. Note that such nodes may range from resource-constrained to high-performance devices. Such a degree of heterogeneity, along with frequent but unpredictable node movements, make configuration and routing exceptionally challenging.

At first glance, the centralized nature of SDN seems to clash with the inner essence of MANETs, which traditionally privilege fully distributed approaches. However, (partially) moving control logic from NDs to a

controller may be a crucial turning point in MANETs, which historically lack adequate network flexibility and programmability. In SDMANs, the controller can support efficient and flexible routing (intended as an ongoing process continuously adapting to the ever-changing network conditions), enforce sophisticated network policies, and reconfigure NDs dynamically.

As mentioned above, the key concern of researchers about MANETs is to cope with the absence of pre-placed network infrastructure. In this regard and according to the state-of-the-art analysis we provided, researchers should prioritize the distinctive features (see Table 5) for improving network resiliency. Specifically, [92] proposes a dynamic controller election for a swarm platform. Since the controller is not known a priori, each node participating in the swarm is provided with the capabilities (i.e., software packages) to become a controller. To avoid the controller-per-node approach, [94] suggests a scheme where the CP periodically broadcasts lists of segment labels, which represent guidelines indicating the intermediate nodes packets should traverse. However, NDs make autonomous decisions (and can neglect such guidelines if necessary) on routing packets from one segment to the next. Similarly, [93] provides NDs with local agents that disseminate among each other link-state information based on a publish/subscribe mechanism to build backup forwarding rules. Such NDs can dynamically select controller-provided or backup forwarding rules depending on the current network state. Lastly, [95] investigates the possibility of providing NDs with a distributed routing protocol as a fallback routing mechanism. Alternatively, [95] also considers the possibility to equip NDs with a common software execution platform for running code fragments pre-computed and pushed by the CP.

Let us note that [92, 93, 94, 95] consistently propose hybrid control architectures to improve network resiliency. Therefore, SDMANs demand to fade the clear distinction between the CP and FP imposed by canonical SDN. This design choice denotes the necessary trade-off between control logic centralization (SDN) and distributed routing decision-making (MANETs). Since the network topology changes continuously, network disruptions occur unpredictably. Accordingly, researchers have to carefully consider that NDs should keep the ability to make autonomous decisions, such as routing decision-making or even controller (s)election, if necessary.

4.4. Vehicular Ad Hoc Networks (VANETs)

A VANET enables Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications through a wireless medium. In the context of VANETs, the infrastructure consists of fixed roadside equipment, typically referred to as Roadside Units (RSUs). RSUs are provided with NDs (usually) based on IEEE 802.11p to enable ad hoc networking with vehicles. Moreover, RSUs may be equipped with additional NDs for connecting to the Internet or with other RSUs. When a vehicle cannot reach other vehicles because out of its transmission range, multiple RSUs dislocated on the road act as relays to allow communications. On-Board Units (OBUs), mounted on vehicles, are equipped with a ND based on 802.11p to enable ad hoc networking with other vehicles and RSUs. Also, the OBU may include NDs that support other radio technologies, e.g., WCNs. WCNs successfully support non-safety applications, providing information/entertainment services to passengers and pursuing traffic efficiency. However, such networks cannot still cope with safety applications [99], which require minimal latency to improve road safety and avoid accidents. Key enablers for such applications are direct V2V and V2I communications. Table 2 (fifth column) states the defining characteristics of VANETs.

VANETs present the following peculiarities [100]. First, since vehicles are limited by road topology, roadside equipment, and other vehicles' movement, VANETs are not characterized by fully random node mobility. However, the speed of vehicles may be very high and thus the network topology may change rapidly. Second, the network size may grow significantly, especially in dense urban areas. Third, vehicles do not suffer from power constraints and may be easily equipped with adequate sensors and computational resources. Given the limited bandwidth available in VANETs, the lack of a central coordinator makes ensuring fair bandwidth utilization a challenging task. Especially in dense urban areas, channel congestion may frequently occur by limiting the possibility for vehicles to transmit information. Therefore, vehicles must wait until the medium is available again, delving into increased latency, a crucial factor for safety applications.

Another critical challenge is the design of efficient routing protocols, which must consider several factors, such as vehicle density increasing dramatically in dense urban areas, buildings causing signal interference, and vehicles moving at volatile velocity (e.g., highways vs. urban roads). Furthermore, such protocols should provide high performance in terms of communication delay and bandwidth consumption. Lastly, the network topology is highly dynamic, resulting in frequent network partitions. However, information dissemination must keep working in a timely fashion, preserving the

ability to avoid or eventually cope with emergency circumstances.

Since VANETs consist of autonomous entities sharing and consuming information for making decisions that may directly impact people's lives, VANETs also raise concerns in terms of privacy and security. For example, a vehicle might deliberately share misleading information to take advantage of how other vehicles respond to that information. On the one hand, the receiver wants to rely on trustworthy information. On the other hand, the receiver's needs may conflict with the sender's privacy policy.

4.4.1. SDVN: Objectives and Control Architectures

The term Software-Defined Vehicular Network (SDVN) indicates the integration of the SDN paradigm into VANETs [8]. Specifically, VANETs may benefit from SDN along several dimensions [102]. First, the controller can improve the quality of routing decisions, e.g., by dynamically load-balancing network traffic to mitigate traffic congestion. Second, the controller can compute optimal paths and coordinate the access to the wireless medium by exploiting its network-wide view. For instance, the controller can reserve a channel for emergency services when road accidents occur. Generally, the controller should continuously adapt the allocation of underlying network resources to meet the delay-constrained requirements demanded by safety applications. Lastly, the controller can change the power transmission of NDs mounted on vehicles to enhance their transmission range, which may be a good strategy in scenarios characterized by a low density of vehicles. Given that some of the following proposals also consider ground stations provided with cellular-based technologies as part of the available network infrastructure (in addition to RSUs), we use the term Base Stations (BSs) to refer to such network equipment. Table 6 sums up the surveyed proposals concerning SDVNs.

[101] proposes a *centralized* control architecture addressing the heterogeneity of the network infrastructure and the tremendous degree of mobility characterizing VANETs. The FP consists of an overlay network where vehicles, RSUs, and BSs become mere FEs. In the CP, the controller collects status updates from underlying NDs and maintains a network-wide view. However, the highly dynamic nature of VANETs may frequently undermine the controller capability of gathering information about the FP. Although RSUs and BSs are usually reachable through reliable connections, vehicles not directly connected to a RSU may become completely unreachable. Therefore, [101] proposes a trajectory prediction system to estimate the positions of vehicles, which aims at lowering both network overhead and complexity. In fact, continuously tracking vehicle location in a real-time fashion may consume a significant portion of the available bandwidth. Also, temporarily disconnected vehicles would require some so-called fallback mechanisms. Without a trajectory prediction system, disconnected vehicles should roll back to purely decentralized ad hoc communications. Hence, the control logic would not be completely removed from vehicles that should implement specific hardware/software (imposing higher degree of complexity) to adapt how they behave according to the circumstances.

[102] proposes a *hybrid centralized* control architecture where NDs are equipped with local agents being able to take control when specific circumstances occur. For example, if a vehicle cannot reach the controller anymore, the local agent running on such a vehicle turns on a fallback mechanism reverting to a traditional ad hoc routing protocol. Such a local agent can also play a role when the controller and NDs are fully connected. In fact, [102] proposes a particular control mode where the controller only defines policies without directly controlling how the underlying NDs handle each flow request. Then, according to the controller policies (which define the general network behavior), NDs make local-level decisions autonomously.

[103] proposes a *hybrid centralized* control architecture to address the routing scalability challenge in wireless distributed networks (i.e., MANETs, WSNs, and VANETs). We present this work here because its effectiveness has been experimentally evaluated only for vehicular environments. In particular, [103] designs a solution in which the controller processes state-link information in a centralized manner. Then, the pre-processed information is delivered to underlying NDs. Based on the information processed by the controller, NDs make their own routing decisions by using distributed algorithms. Therefore, NDs do not need to directly exchange state-link information among them, with the notable benefit of reducing network traffic. Also, especially in large networks, routing computation may become a remarkably computational-intensive task. Instead, [103] partially moves computational complexity on the controller, which provides the key information for enabling routing decision-making on NDs, resulting in a solution that can scale as the network grows. However, scalability comes at the cost of potentially making suboptimal routing decisions.

[104] proposes a *flat* control architecture realizing high-reliability and low-latency vehicle-to-everything communications by joining VANETs and WCNs. While VANETs represent the logical choice for V2V commu-

Table 6: Control architectures for SDVNs.

Proposal	Architecture	Hybrid	Based on	Distinctive Feature
[101]	Centralized	No	POX	Adaptive routing protocol switching and trajectory prediction system
[102]	Centralized	Yes	-	Fallback routing mechanism
[103]	Centralized	Yes	-	Centralized link-state processing but distributed routing decision-making
[104]	Flat	No	-	MEC integration
[105]	Hierarchical	Yes	-	Dynamic controller election and efficient link failure recovery
[106]	Hierarchical	Yes	-	Proactive flow rule installation and fallback routing mechanism

nications, they cannot ensure adequate network performance when RSUs are scarcely deployed (or even inexistent, especially true in rural areas). Instead, WCNs benefit from better coverage, and therefore can potentially bridge the gap. However, packets should travel from vehicles to the controller (running on a remote cloud) and back again to vehicles. As a result, it would be impossible to guarantee the low-latency requirements demanded by safety applications. Therefore, [104] proposes equipping BSs with Multi-access Edge Computing (MEC) technology. By placing computing and storage resources at the edge of the cellular infrastructure, a controller can be directly deployed on a BS. Accordingly, multiple controllers physically distributed (but sharing a common global network view) on different BSs can manage distinct subnetworks. In such a scenario, finding the optimal controller placement and reducing controller synchronization messages become key challenges.

[105] proposes a *hybrid hierarchical* control architecture efficiently managing circumstances characterized by a loss of connectivity with the controller without falling back to traditional ad hoc routing protocols. In particular, [105] designs a two-layer hierarchical CP consisting of a RC in the upper layer and a LC for each SDN domain in the bottom one. SDN domains are defined through a clustering technique [107], and LCs are located at cluster heads. The RC is responsible for dictating the network behavior by communicating its decisions to LCs. However, the RC represents a SPoF. In fact, network partitions may isolate the RC by undermining the whole network performance. When the RC becomes unreachable, LCs can act independently within their domain. As soon as the connection is reestablished, LCs re-start operating according to the instructions received by the RC.

[106] proposes a *hybrid hierarchical* control architecture taking advantage of SDN while minimizing the additional delay introduced by the SDN architecture itself. Specifically, such an architecture seeks to reduce path computation and end-to-end delay by only recurring to VANET-specific technologies (i.e., no usage of additional radio access technologies such as WCNs) and to provide flexibility and programmability to the system. The CP consists of two hierarchical levels. The upper layer comprises a RC running on the Internet, and the lower layer consists of multiple LCs placed on the roadside equipment level. In fact, placing LCs on RSUs lowers the end-to-end delay considerably. However, it would not be feasible to place a LC for each RSU. First, the number of LCs would increase dramatically. Second, vehicles would frequently pass from one LC to another by increasing the handover management cost. Third, the LC would take advantage of a minimal amount of knowledge, the only one related to a single RSU. Therefore, the optimal option is to place a sufficient number of controllers so that each of them can benefit from an adequate level of local knowledge. In other words, the objective is to place a certain number of LCs in several RSUs by maximizing the contact duration between vehicles and LCs but without exceeding the minimum route setup delay of conventional VANETs. In the case of disconnected vehicles, [106] proposes a fallback mechanism where such vehicles revert to traditional ad hoc routing protocols.

4.4.2. SDVN: Lessons Learned

The key concern about VANETs is to ensure the low-latency requirements demanded by safety applications, which seek to enhance road safety by taking advantage of the information circulation enabled by V2V and V2I communications.

Adopting SDN in VANETs allows bridging the gap between low-level traffic management and high-level requirements of safety applications. In SDVNs, the controller can enforce policies to guarantee low-latency communications for safety applications, e.g., by delaying, steering, or dropping non-mission-critical traffic flows under specific circumstances. However, researchers should carefully consider that supporting safety applications is extremely challenging due to sudden changes in network topology, heavy traffic zones dramatically increasing the network size and potentially resulting in medium congestion, buildings causing sig-

nal interference, volatile vehicle velocity, scarceness of RSUs in rural areas, as well as security and privacy concerns.

As mentioned above, the key concern of researchers about VANETs is to enable safety applications. In this regard and according to the state-of-the-art analysis we provided, researchers should be aware of the distinctive feature (see Table 6) for providing highly-resilient, low-latency communications. In fact, network resiliency is paramount to prevent circumstances where vehicles cannot communicate because they do not receive up-to-date flow rules from the controller(s). In the worst-case scenario, such circumstances may even lead to road accidents, highly undesirable as hazarding people's lives. Accordingly, [102, 106] adopt fallback routing mechanisms that revert communications to traditional ad hoc routing protocols to address a lack of connectivity with the CP. In addition, [106] minimizes the delays due to SDN itself by optimally placing LCs on a subset of RSUs and instantiating flow rules proactively. Similarly, [104] deploys controllers on MEC-integrated BSs, which benefit from better coverage than RSUs. To avoid fallback routing mechanisms, [101] provides a trajectory prediction system that estimates the positions of vehicles, while [105] proposes a dynamic controller election to recover from link failures efficiently.

Another distinctive feature proposed by researchers, but not primarily related to the key concern, is about the routing scalability challenge in wireless ad hoc scenarios. In particular, [103] proposes centralized link-state processing but distributed routing decision-making, taking advantage of hybridization systematically, not just as a backup plan.

Let us note that hierarchical control architectures seem to be the logical design choice for SDVNs. Such a design pattern provides good scalability (useful in urban areas) and low-latency communications (a critical aspect for safety applications) enabled by LCs on RSUs or BSs.

4.5. Flying Ad Hoc Networks (FANETs)

A FANET is a form of MANET in which multiple UAVs communicate with each other through an ad hoc network [108]. Note that only multi-UAV systems where UAV-to-UAV (U2U) communications take place in an ad hoc fashion can be defined as FANETs. On the contrary, some multi-UAV systems only support U2U communications passing through the infrastructure, i.e., each UAV always communicates with a BS or satellite, but never directly one another. Table 2 (sixth column) states the defining characteristics of FANETs.

According to [108], FANETs differ from MANETs, VANETs, and WSNs for the following reasons. First, FANETs are characterized by a different degree of node mobility. In contrast to VANETs, where vehicles move through the road network, UAVs fly in the sky. Although UAVs may follow predefined flight plans, they can move in any direction at any time in response to external factors, e.g., environmental conditions. Another difference is related to node velocity. In fact, UAVs can be still, e.g., airships and quadcopters, but can also move much faster than terrestrial entities such as pedestrians and even vehicles. As a consequence of the (potential) higher degree of mobility, the network topology characterizing FANETs may change more frequently than in MANETs and VANETs. Similar to WSNs, FANETs can sense the environment. In fact, UAVs can be equipped with sensors for gathering information about the environment. However, UAVs do not suffer from energy constraints as much as sensor nodes. Also, the node density in FANETs is much lower than in WSNs. Lastly, U2U communications usually cover longer distances than typical point-to-point communications characterizing MANETs and VANETs.

4.5.1. SDFN: Objectives and Control Architectures

With the term Software-Defined Flying Network (SDFN), we indicate FANETs that take advantage of SDN. Traditionally, FANETs suffer from complex network management as well as poor network performance partly resulting from the purely distributed network control, which struggles with the ever-changing network topology, the distinctive mobility degree characterizing UAVs, and the extreme scenarios where UAVs typically operate, among others. Instead, SDFNs enable a logically centralized CP that makes decisions based on a network-wide view, potentially providing far better efficiency. In addition, SDFN provides flexibility to the FP, which (theoretically) limits its scope of work to apply the instructions pushed by the CP, and programmability, which results from the SDN layering. Finally, SDFNs offer greater network survivability since the logically centralized CP can make routing decision-making optimally, coping with UAV and link failures as well as minimizing energy consumption. Given that some of the following proposals also consider ground stations as part of the available network infrastructure, we use the term BSs to refer to such network equipment. Table 7 sums up the surveyed proposals concerning SDFNs.

[109] proposes a *centralized* control architecture providing resilient multi-path routing in FANETs. The versatile nature of UAVs (which can potentially be easily and rapidly deployed almost everywhere) makes

Table 7: Control architectures for SDFNs.

Proposal	Architecture	Hybrid	Based on	Distinctive Feature
[109]	Centralized	No	OpenDaylight	Resilient multi-path routing
[110]	Centralized	Yes	-	Dynamic controller election and QoS-based multi-path routing
[111]	Flat	No	-	Blockchain-based CP and dynamic controller selection
[112]	Hierarchical	No	-	Reliability prediction model and ant-colony-based routing
[113]	Hierarchical	No	-	SR-based traffic scheduling

FANETs remarkably useful to enable communications in scenarios where the information exchange is essential but extremely challenging, such as disaster relief missions, tactical operations, and environmental exploration. Consequently, environmental obstructions, weather conditions, and adversarial attacks may frequently affect communications in such scenarios. In the network architecture depicted by [109], UAVs, acting as OpenFlow switches, are provided with GPS units and various radio access technologies. In addition to information about network status, UAVs also send their 3D positions to the controller. According to the network-wide view provided by the controller and on a spatial diversity metric designed by the authors, a centralized routing algorithm (an enhance version of the modified Dijkstra one with a vertex-splitting method [114]) computes disjoint multi-path routes to prevent end-to-end connection disruptions. Note that two routes are disjoint if they do not share any intermediate UAV. Given that a malicious UAV may jam multiple UAVs and therefore multiple routes, the routing algorithm also considers the physical distance between UAVs belonging to different routes by preventing cases in which the same malicious UAV disrupts multiple routes simultaneously.

[110] proposes a *hybrid centralized* control architecture based on SDN and MQTT [115] to support battlefield UAV swarms, which will play a critical role in future battlefield networking. Such an architecture imposes that each UAV involved in the swarm presents the same capabilities. According to a leader election mechanism (similar to [116]), a UAV becomes the master and the others become slaves. However, a UAV can adaptively activate or deactivate some capabilities, switching from master to slave (or vice versa) to fit the ever-changing network conditions. Additionally, [110] provides a QoS-based multi-path routing framework working as follows. First, it computes the transmission cost of each link, where the cost depends on several factors, such as network state, distance, energy consumption, resource reservation, and security policies. Then, UAVs classify transmission services based on the available communication technologies. Specifically, type-A, type-B, and type-C services communicate through delay-tolerant networking, satellite, and radio modules, respectively. However, UAVs lacking delay-tolerant networking or satellite modules reclassify type-A and type-B services as pseudo-type-C ones, adjust their QoS accordingly, and route them towards devices providing such modules. Lastly, a routing algorithm computes multiple disjoint paths for type-C and pseudo-type-C services. If no paths are available, then the controller seeks to negotiate lower QoS requirements. If possible, pseudo-type-C services should be recovered to type-A or type-B services, depending on their initial status.

[111] proposes a *flat* control architecture supporting a blockchain-based CP to promote flexibility, programmability, survivability, and security in FANETs. Advantages in terms of flexibility and programmability directly come from the adoption of the SDN paradigm itself. Instead, improvements regarding survivability and security depend on specific design choices. The CP consists of multiple physically distributed (but logically centralized) controllers running on different BSs. Such a redundant strategy inhibits a SPoF, improving network survivability. UAVs opportunistically select the controller instance to interact with, lowering transmission delay and enhancing communication network efficiency. In addition, UAVs are provided with the public keys of the trustworthy controllers beforehand, avoiding that such UAVs communicate with malicious controllers. The controllers share and synchronize information in a blockchain-based fashion, adopting the Practical Byzantine Fault Tolerance consensus algorithm [117]. Additionally, the routing scheduling and the flow rule calculation logic are stored in the blockchain as smart contracts by ensuring the consistency of the scheduling algorithm. Since the blockchain enables controllers to hold the same consistent network-wide view, it does not matter which BS controls which UAV. In fact, controllers make equivalent decisions since based on the same information.

[112] proposes a *hierarchical* control architecture providing traffic-differentiated routing supported by a transmission reliability prediction system. The FP con-

sists of several UAV clusters, which represent distinct logical domains, each managed by a dedicated controller. Such controllers form the bottom layer of the two-tier hierarchy. Instead, the top layer includes a single so-called collaborative controller responsible for coordinating the cluster-level controllers. Note that controllers do not run on BSs nor UAVs, but on upper stationary airships. Cluster-level controllers use the traffic-differentiated routing algorithm to guarantee delay and reliability requirements claimed by the heterogeneous range of applications provided by UAVs. Such an algorithm periodically updates routing paths by minimizing the average delay affecting delay-sensitive applications and maximizing reliability for applications demanding data integrity. Specifically, the routing problem is solved using an ant-colony-based heuristic function. Logically dividing nodes demanding delay-sensitive from those requiring reliability, such a heuristic function seeks to find a possible merged spanning tree according to the introduced optimization problem. To this purpose, the traffic-differentiated routing algorithm also relies on a transmission reliability prediction system that provides estimations about link availability and node forwarding ability based on historical knowledge.

[113] proposes a *hierarchical* control architecture integrating SDN and airborne backbone networking to support Network Centric Warfare (NCW), which considers the success of tactical operations heavily based on the capacity to exploit information. At the same time, the dynamic, chaotic, and complex nature of the military context makes information circulation during NCW-based operations extremely challenging. However, recent advances in airborne technology and SDN may jointly pave the way for building an airborne backbone network satisfying tactical communication demands while providing flexibility, openness, interoperability, and evolvability. In particular, [113] designs a CP consisting of a two-tier hierarchy. At the top level, multiple controllers back up each other and jointly dictate the network behavior in a logically centralized manner. Such top-level controllers are physically deployed only on aircraft with strong survivability to keep them safe. At the bottom level, multiple controllers (one for each aircraft) work as proxies between the top-level controllers and the FP. In addition, such bottom-level controllers are provided with independent network control logic, taking control when the top-level controllers become unreachable or inefficient. The FP consists of software-defined communication systems (i.e., programmable NDs equipped with heterogeneous radios), which simultaneously enable communications between heterogeneous networks (e.g., airborne tactical subnetworks, navy networks, and army networks). Each aircraft is provided with a so-called software-defined communication system. Moreover, [113] provides a SR-based traffic scheduling algorithm specifically designed for the control architecture described earlier that aims at improving reliability as well as the real-time performances related to network traffic forwarding.

4.5.2. SDFN: Lessons Learned

The key concern of FANETs is to cope with a unique kind of mobility, where aircraft reach tremendous velocities, move along the three spatial dimensions, and change direction unpredictably. Furthermore, FANETs typically support communications in challenging scenarios ranging from tactical to disaster relief operations, where environmental conditions and adversarial attacks may threaten the network survivability.

Adopting SDN in FANETs allows improving network flexibility, programmability, and, above all, survivability, going beyond the traditional distributed network control, which is not sufficiently adequate to fulfill the increasingly demanding requirements that FANETs deal with nowadays.

As mentioned above, the key concern of researchers about FANETs is to cope with extremely high node mobility. In this regard and according to the state-of-the-art analysis we provided, researchers should focus on the distinctive features (see Table 7) for improving network resiliency. Accordingly, [109] designs a resilient multi-path routing algorithm based on a spatial diversity metric. Similarly, [113] provides an SR-based traffic scheduling scheme to improve reliability by balancing the network traffic across multiple paths. Additionally, [112] introduces a transmission reliability prediction system to support a traffic-differentiated routing algorithm specifically designed for ensuring delay and reliability requirements. Lastly, [110, 111] propose dynamic controller election and dynamic controller selection, respectively.

Other distinctive features proposed by researchers, but not primarily related to the key concern, regard QoS and security. In particular, [110] provides QoS-based multi-path routing based on several metrics (e.g., energy consumption, network state, distance, among others) and the communication technologies available in situ. [111] proposes a blockchain-based CP where routing scheduling and flow rule calculation logic are smart contracts within the blockchain.

Let us note that the highly dynamic nature of FANETs makes the controller placement extremely problematic. Notably, some works [109, 111] place the

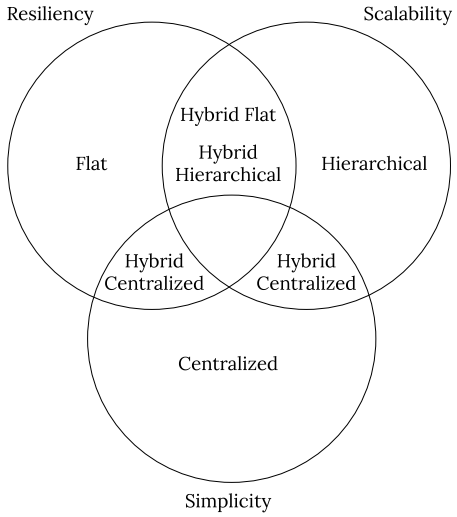


Figure 4: The correlation between design patterns and complexity.

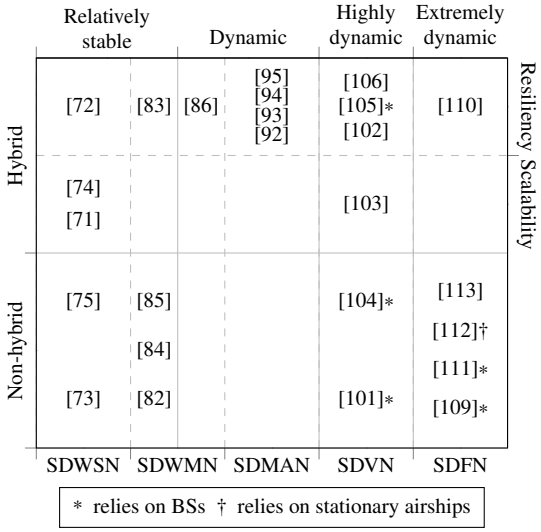


Figure 5: The correlation between mobility and hybridization.

controller(s) on BSs, while others [112, 113, 110] on aircraft without relying on any form of terrestrial support.

5. The Role of Hybridization

Based on the in-depth analysis of the state-of-the-art literature presented in Section 4, we identify hybridization as a clear trend in designing SDN control architectures for wireless ad hoc scenarios.

Figure 4 schematically depicts how centralized, flat, and hierarchical control architectures fit resiliency, scalability, and simplicity requirements. The figure omits federated control architectures because we consider them as compositions of different architectures, where each federated domain may implement a distinct design pattern. Let us remark that hybridization is a cross-category attribute, and thus it can be applied to each control architecture. For example, centralized control architectures best fit simplicity with little resiliency and scalabilty. However, by adopting an hybrid strategy, centralized control architectures can improve resiliency (e.g., fallback routing mechanisms) as well as scalability (e.g., in-network processing) at the cost of simplicity. Similar considerations can be done for hierarchical and flat architectures: hybrid strategies provide better resiliency or scalability, while increasing complexity.

Table 8 lists the proposals presented in this survey by outlining the role of hybridization in designing control architectures for SDWSNs, SDWMNs, SDMANs, SDVNs, and SDFNs. For the sake of clarity, Figure 5 depicts a slice of Table 8 graphically, i.e., the role of hybridization in correlation with the mobility degree.

In particular, the figure shows that wireless ad hoc scenarios involving relatively stable NDs, such as SDWSNs and SDWMNs, do not recur to hybridization extensively. In this regard, it is worth remarking that the FP of SDWSNs includes primarily static sensor nodes. Moreover, the FP of SDWMNs usually consists of only mesh routers (i.e., static elements) [82, 83, 84, 85], while mesh clients (i.e., dynamic elements) are included rarely [86]. Since relatively stable network topologies usually prevent the need to maintain control logic outside the controller(s), the role of hybridization is rather marginal. A notable exception regards SDWSNs, where [71, 74] exploit hybrid architectures to enable in-network processing for improving scalability.

As the mobility degree grows, so does the role of hybridization. In fact, SDMANs systematically take advantage of hybridization. Since MANETs are infrastructure-less multi-hop wireless ad hoc networks where nodes move unpredictably, resiliency is critical. Accordingly, [92, 93, 94, 95] (i.e., the full spectrum of the proposals surveyed for this scenario) implement hybridization strategies for improving resiliency. Note that even flat control architectures recur to hybridization in SDMANs, notwithstanding such architectures are the most resilient by nature (see Figure 4).

Although SDVNs are characterized by a higher mobility degree than SDMANs (dynamic vs. highly dynamic), the role of hybridization is not as much as pervasive in the former in comparison with the latter. However, it is worth noting that several propos-

Table 8: The role of hybridization in software-defined wireless ad hoc networks.

SDN	Proposal	Architecture	Network Devices	Network Topology	Hybridization Strategy	Objective
SDWSN	[71]	Centralized	Sensors	Relatively stable	In-network processing	Scalability
	[72]	Centralized	Sensors	Relatively stable	Fallback routing mechanism	Resiliency
	[73]	Flat	Sensors	Relatively stable	-	-
	[74]	Flat	Sensors	Relatively stable	In-network processing	Scalability
	[75]	Hierarchical	Sensors	Relatively stable	-	-
SDWMN	[82]	Centralized	Mesh routers	Relatively stable	-	-
	[83]	Centralized	Mesh routers	Relatively stable	Fallback routing mechanism	Resiliency
	[84]	Flat	Mesh routers	Relatively stable	-	-
	[85]	Flat	Mesh routers	Relatively stable	-	-
	[86]	Federated	Mesh routers and clients	Dynamic	Dynamic controller election	Resiliency
SDMAN	[92]	Centralized	Mobile nodes	Dynamic	Dynamic controller election	Resiliency
	[93]	Centralized	Mobile nodes	Dynamic	Backup forwarding rules	Resiliency
	[94]	Flat	Mobile nodes	Dynamic	Distributed routing protocol	Resiliency
	[95]	Hierarchical	Mobile nodes	Dynamic	Fallback routing mechanism	Resiliency
SDVN	[103]	Centralized	Vehicles	Highly dynamic	Distributed routing decision-making	Scalability
	[102]	Centralized	Vehicles and RSUs	Highly dynamic	Fallback routing mechanism	Resiliency
	[106]	Hierarchical	Vehicles and RSUs	Highly dynamic	Fallback routing mechanism	Resiliency
	[101]	Centralized	Vehicles, RSUs, and BSs	Highly dynamic	-	-
	[104]	Flat	Vehicles, RSUs, and BSs	Highly dynamic	-	-
	[105]	Hierarchical	Vehicles, RSUs, and BSs	Highly dynamic	Dynamic controller election	Resiliency
SDFN	[109]	Centralized	UAVs	Extremely dynamic	-	-
	[110]	Centralized	UAVs	Extremely dynamic	Dynamic controller election	Resiliency
	[111]	Flat	UAVs	Extremely dynamic	-	-
	[112]	Hierarchical	UAVs	Extremely dynamic	-	-
	[113]	Hierarchical	Aircraft	Extremely dynamic	-	-

als for SDVNs include BSs as part of the FP to benefit from a better coverage of fixed network infrastructure [101, 104, 105]. SDVNs mainly exploit hybridization to ensure information circulation among vehicles when the CP is unreachable, keeping the ability to cope with emergency circumstances (e.g., road accidents) timely.

Lastly, the role of hybridization seems negligible in SDFNs at first glance. However, it is worth noting that [109, 111] place the CP on BSs, which are not formally part of FANETs. Similarly, [112] places the CP on stationary airships, which, as BSs, are stable. This explains why the proposals for SDFNs quantitatively recur to hybridization less than the others, although SDFNs have the highest mobility degree.

6. Conclusion

This survey explores the adoption of SDN control architectures in wireless ad hoc scenarios characterized, among others, by different mobility degrees. Such original state-of-the-art analysis allows us to disclose relevant trends in designing control architectures targeting such scenarios. In particular, we observe a pervasive role of hybridization to enable SDN in dynamic and mobile environments while leveraging a certain degree of on-board control logic in NDs.

As future research directions, we believe next-generation proposals will deeply integrate machine learning techniques (e.g., reinforcement learning and federated learning) to predict the ever-changing network conditions and dictate an appropriate network behavior accordingly. In this regard, some solutions included in this survey already take advantage of prediction systems to cope with highly dynamic environments [101, 112]. In addition, we also believe that novel hybrid CP-FP interplay will pave the way to effectively enable SDN even in extremely dynamic scenarios where (partially) logically centralized solutions are still not widely accepted. In conclusion, we envision SDN as a no-way-back trend, even for wireless ad hoc networks, bridging the gap between high-level application objectives and low-level traffic management.

References

[1] 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, A. Vahdat, B4: Experience with a globally-deployed software defined wan, in: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM '13, Association for Computing Machinery, 2013, p. 3–14. doi:10.1145/2486001.2486019.

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

[3] H. Mostafaei, M. Menth, Software-defined wireless sensor networks: A survey, Journal of Network and Computer Applications 119 (2018) 42–56.

[4] M. Rademacher, K. Jonas, F. Siebertz, A. Rzyska, M. Schlebusch, M. Kessel, Software-defined wireless mesh networking: Current status and challenges, Computer journal 60 (10) (2017) 1520–1535.

[5] M. Chahal, S. Harit, K. K. Mishra, A. K. Sangaiah, Z. Zheng, A survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases, Sustainable Cities and Society 35 (2017) 830–840.

[6] O. S. Al-Heety, Z. Zakaria, M. Ismail, M. M. Shakir, S. Alani, H. Alsariera, A comprehensive survey: Benefits, services, recent works, challenges, security, and use cases for sdn-vanet, IEEE Access 8 (2020) 91028–91047.

[7] W. Ben Jaballah, M. Conti, C. Lal, Security and design requirements for software-defined vanets, Computer Networks 169 (2020).

[8] M. M. Islam, M. T. R. Khan, M. M. Saad, D. Kim, Software-defined vehicular network (sdvn): A survey on architecture and routing, Journal of Systems Architecture 114 (2021).

[9] O. Sami Oubbati, M. Atiquzzaman, T. Ahamed Ahanger, A. Ibrahim, Softwarization of uav networks: A survey of applications and future trends, IEEE Access 8 (2020) 98073–98125.

[10] Y. E. Oktian, S. Lee, H. Lee, J. Lam, Distributed sdn controller system: A survey on design choice, Computer Networks 121 (2017) 100–111.

[11] F. Bannour, S. Souihi, A. Mellouk, Distributed sdn control: Survey, taxonomy, and challenges, IEEE Communications Surveys and Tutorials 20 (1) (2018) 333–354.

[12] Y. Zhang, L. Cui, W. Wang, Y. Zhang, A survey on software defined networking with multiple controllers, Journal of Network and Computer Applications 103 (2018) 101–118.

[13] L. Zhu, M. M. Karim, K. Sharif, C. Xu, F. Li, X. Du, M. Guizani, Sdn controllers: A comprehensive analysis and performance evaluation study, ACM Computing Surveys 53 (6) (2021).

[14] N. A. Jagadeesan, B. Krishnamachari, Software-defined networking paradigms in wireless networks: A survey, ACM Computing Surveys 47 (2) (2014).

[15] I. T. Haque, N. Abu-Ghazaleh, Wireless software defined networking: A survey and taxonomy, IEEE Communications Surveys and Tutorials 18 (4) (2016) 2713–2737.

[16] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, G. J. Minden, A survey of active network research, IEEE Communications Magazine 35 (1) (1997) 80–86.

[17] N. Feamster, J. Rexford, E. Zegura, The road to sdn: An intellectual history of programmable networks, Computer Communication Review 44 (2) (2014) 87–98.

[18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: Enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74. doi:10.1145/1355734.1355746.

[19] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proceedings of the IEEE 103 (1) (2015) 14–76.

[20] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, O. Koufopavlou, Software-defined networking (sdn): Layers and architecture terminology, RFC 7426 (2015).

[21] Arista 7150 series, `https://www.arista.com/en/products/7150-series`, [Online; accessed 20-July-2021].

[22] Centec v350 series, `https://www.centecnetworks.com/solution/9`, [Online; accessed 20-July-2021].

[23] Open vswitch, `https://github.com/openvswitch/ovs`, [Online; accessed 24-March-2021].

[24] E. L. Fernandes, E. Rojas, J. Alvarez-Horcajo, Z. L. Kis, D. Sanvito, N. Bonelli, C. Cascone, C. E. Rothenberg, The road to bofuss: The basic openflow userspace software switch, Journal of Network and Computer Applications (2020) 102685.

[25] O. N. Foundation, Openflow switch specification, version 1.5.0 (2014).

[26] J. Halpern, J. H. Salim, et al., Forwarding and control element separation (forces) forwarding element model, RFC 5812 (2010).

[27] M. Bjorklund, et al., Yang-a data modeling language for the network configuration protocol (netconf), RFC 6020 (2010).

[28] R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, Management information base (mib) for the simple network management protocol (snmp), RFC 3418 (2002).

[29] A. Doria, J. H. Salim, R. Haas, H. M. Khosravi, W. Wang, L. Dong, R. Gopal, J. M. Halpern, Forwarding and control element separation (forces) protocol specification., RFC 5810 (2010).

[30] H. Song, Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 127–132. doi:10.1145/2491185.2491190.

[31] G. Bianchi, M. Bonola, A. Capone, C. Cascone, Openstate: Programming platform-independent stateful openflow applications inside the switch, SIGCOMM Comput. Commun. Rev. 44 (2) (2014) 44–51. doi:10.1145/2602204.2602211.

[32] M. Smith, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, N. Weidenbacher, Opflex control protocol, 2016, Tech. rep., IETF draft, work in progress.

[33] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, Network configuration protocol (netconf), RFC 6241 (2011).

[34] D. Harrington, R. Presuhn, B. Wijnen, An architecture for describing simple network management protocol (snmp) management frameworks, RFC 3411 (2002).

[35] B. Pfaff, B. Davie, The open vswitch database management protocol, RFC 7047 (2013).

[36] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, Y. Wang, A comprehensive survey of interface protocols for software defined networks, Journal of Network and Computer Applications 156 (2020).

[37] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, R. Sidi, Sdni: A message exchange protocol for software defined networks (sdns) across multiple domains, Tech. rep. (2012).

[38] Apache zookeeper, `https://github.com/apache/zookeeper`, [Online; accessed 19-July-2021].

[39] Advanced message queuing protocol, `https://www.amqp.org`, [Online; accessed 19-July-2021].

[40] J. Stribling, Y. Sovran, I. Zhang, X. Pretzer, J. Li, M. F. Kaashoek, R. Morris, Flexible, wide-area storage for distributed systems with wheelfs, NSDI'09 (2009) 43–58.

[41] Q. Vohra, E. Chen, Bgp support for four-octet autonomous system (as) number space, RFC 6793 (2012).

[42] J. P. Vasseur, J. L. Le Roux, et al., Path computation element (pce) communication protocol (pcep), RFC 5440 (2009).

[43] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, D. Walker, Frenetic: A network programming language, ACM SIGPLAN Notices 46 (9) (2011) 279–291.

[44] A. Voellmy, P. Hudak, Nettle: Taking the sting out of programming network routers, Vol. 6539 LNCS of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2011.

[45] A. Voellmy, H. Kim, N. Feamster, Procera: A language for high-level reactive network control, in: HotSDN'12 - Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks, 2012, pp. 43–48.

[46] C. Monsanto, J. Reich, N. Foster, J. Rexford, D. Walker, Composing software-defined networks, in: Proceedings of the 10th USENIX Symposium on Networked Systems Design and Im-

plementation, NSDI 2013, 2013, pp. 1–13.
[47] S. H. Yeganeh, A. Tootoonchian, Y. Ganjali, On scalability of software-defined networking, IEEE Communications Magazine 51 (2) (2013) 136–141. doi:10.1109/MCOM.2013.6461198.
[48] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker, Nox: Towards an operating system for networks, SIGCOMM Comput. Commun. Rev. 38 (3) (2008) 105–110. doi:10.1145/1384609.1384625.
[49] Z. Cai, A. L. Cox, F. Dinu, T. Ng, J. Zheng, The preliminary design and implementation of the maestro network control platform, Tech. rep. (2008).
[50] Z. Cai, A. L. Cox, T. Ng, Maestro: A system for scalable openflow control, Tech. rep. (2010).
[51] D. Erickson, The beacon openflow controller, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, Association for Computing Machinery, 2013, p. 13–18. doi:10.1145/2491185.2491189.
[52] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, R. Sherwood, On controller performance in software-defined networks, in: 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE 2012, 2012.
[53] Floodlight sdn openflow controller, `https://github.com/floodlight/floodlight`, [Online; accessed 04-March-2021].
[54] The pox network software platform, `https://github.com/noxrepo/pox`, [Online; accessed 04-March-2021].
[55] Ryu component-based software defined networking framework, `https://github.com/faucetsdn/ryu`, [Online; accessed 09-March-2021].
[56] S. Li, D. Hu, W. Fang, S. Ma, C. Chen, H. Huang, Z. Zhu, Protocol oblivious forwarding (pof): Software-defined networking with enhanced programmability, IEEE Network 31 (2) (2017) 58–66. doi:10.1109/MNET.2017.1600030NM.
[57] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, G. Wang, Meridian: an sdn platform for cloud network services, IEEE Communications Magazine 51 (2) (2013) 120–127. doi:10.1109/MCOM.2013.6461196.
[58] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, B. B. Kang, Rosemary: A robust, secure, and high-performance network operating system, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 78–89. doi:10.1145/2660267.2660353.
[59] E. A. Brewer, Towards robust distributed systems, PODC '00, Association for Computing Machinery, New York, NY, USA, 2000.
[60] S. Gilbert, N. Lynch, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, SIGACT News 33 (2) (2002) 51–59. doi:10.1145/564585.564601.
[61] A. Panda, C. Scott, A. Ghodsi, T. Koponen, S. Shenker, Cap for networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 91–96. doi:10.1145/2491185.2491186.
[62] B. Heller, R. Sherwood, N. McKeown, The controller placement problem, SIGCOMM Comput. Commun. Rev. 42 (4) (2012) 473–478. doi:10.1145/2377677.2377767.
[63] A. Tootoonchian, Y. Ganjali, Hyperflow: A distributed control plane for openflow, in: 2010 Internet Network Management Workshop / Workshop on Research on Enterprise Networking, INM/WREN 2010, 2010.
[64] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, S. Shenker, Onix: A distributed control platform for large-scale production networks, in: Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010, 2010, pp. 351–364.
[65] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar, Onos: Towards an open, distributed sdn os, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN '14, Association for Computing Machinery, 2014, p. 1–6. doi:10.1145/2620728.2620744.
[66] The opendaylight project, `https://github.com/opendaylight`, [Online; accessed 10-March-2021].
[67] S. Hassas Yeganeh, Y. Ganjali, Kandoo: A framework for efficient and scalable offloading of control applications, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12, Association for Computing Machinery, 2012, p. 19–24. doi:10.1145/2342441.2342446.
[68] Y. Fu, J. Bi, K. Gao, Z. Chen, J. Wu, B. Hao, Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks, in: Proceedings - International Conference on Network Protocols, ICNP, 2014, pp. 569–576.
[69] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, V. Lin, C. Rice, B. Rogan, A. Singh, B. Tanaka, M. Verma, P. Sood, M. Tariq, M. Tierney, D. Trumic, V. Valancius, C. Ying, M. Kallahalla, B. Koley, A. Vahdat, Taking the edge off with espresso: Scale, reliability and programmability for global internet peering, SIGCOMM '17, Association for Computing Machinery, 2017, p. 432–445. doi:10.1145/3098822.3098854.
[70] K. Phemius, M. Bouet, J. Leguay, Disco: Distributed multi-domain sdn controllers, in: IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World, 2014.
[71] R. Friedman, D. Sainz, An architecture for sdn based sensor networks, in: ACM International Conference Proceeding Series, 2017.
[72] A. S. Yuan, H. . Fang, Q. Wu, Openflow based hybrid routing in wireless sensor networks, in: IEEE ISSNIP 2014 - 2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Conference Proceedings, 2014.
[73] B. Trevizan De Oliveira, L. Batista Gabriel, C. Borges Margi, Tinysdn: Enabling multiple controllers for software-defined wireless sensor networks, IEEE Latin America Transactions 13 (11) (2015) 3690–3696.
[74] L. Galluccio, S. Milardo, G. Morabito, S. Palazzo, Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks, in: Proceedings - IEEE INFOCOM, Vol. 26, 2015, pp. 513–521.
[75] H. I. Kobo, A. M. Abu-Mahfouz, G. P. Hancke, Fragmentation-based distributed control system for software-defined wireless sensor networks, IEEE Transactions on Industrial Informatics 15 (2) (2019) 901–910.
[76] Aodv implementation provided by uppsala university (aodv-uu), `https://sourceforge.net/projects/aodvuu/`, [Online; accessed 23-August-2021].
[77] S. Schmid, J. Suomela, Exploiting locality in distributed sdn control, in: HotSDN 2013 - Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 121–126.
[78] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, Collection tree protocol, in: Proceedings of the 7th ACM Confer-

ence on Embedded Networked Sensor Systems, SenSys 2009, 2009, pp. 1–14.
[79] W. Ye, J. Heidemann, D. Estrin, An energy-efficient mac protocol for wireless sensor networks, in: Proceedings - IEEE INFOCOM, Vol. 3, 2002, pp. 1567–1576.
[80] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann, Impact of network density on data aggregation in wireless sensor networks, in: Proceedings - International Conference on Distributed Computing Systems, 2002, pp. 457–458.
[81] I. F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: A survey, Computer Networks 47 (4) (2005) 445–487.
[82] H. Huang, P. Li, S. Guo, W. Zhuang, Software-defined wireless mesh networks: Architecture and traffic orchestration, IEEE Network 29 (4) (2015) 24–30.
[83] A. Detti, C. Pisa, S. Salsano, N. Blefari-Melazzi, Wireless mesh software defined networks (wmsdn), in: International Conference on Wireless and Mobile Computing, Networking and Communications, 2013, pp. 89–95.
[84] S. Babu, P. V. Mithun, B. S. Manoj, A novel framework for resource discovery and self-configuration in software defined wireless mesh networks, IEEE Transactions on Network and Service Management 17 (1) (2020) 132–146.
[85] H. Elzain, Y. Wu, Software defined wireless mesh network flat distribution control plane, Future Internet 11 (8) (2019).
[86] P. Bellavista, A. Dolci, C. Giannelli, D. D. Padalino Montenero, Sdn-based traffic management middleware for spontaneous wmns, Journal of Network and Systems Management 28 (4) (2020) 1575–1609.
[87] Real ad-hoc multi-hop peer-to-peer (ramp) project, `https://github.com/DSG-UniFE/ramp`, [Online; accessed 23-June-2021].
[88] Y. T. Hou, Y. Shi, H. D. Sherali, Optimal spectrum sharing for multi-hop software defined radio networks, in: Proceedings - IEEE INFOCOM, 2007, pp. 1–9.
[89] Olsr daemon (olsrd), `http://www.olsr.org/mediawiki/index.php/Olsr_Daemon`, [Online; accessed 23-August-2021].
[90] T. H. Clausen, P. Jacquet, Optimized link state routing protocol (olsr), RFC 3626 (Oct. 2003).
URL `https://rfc-editor.org/rfc/rfc3626.txt`
[91] I. Chlamtac, M. Conti, J. J. . Liu, Mobile ad hoc networking: Imperatives and challenges, Ad Hoc Networks 1 (1) (2003) 13–64.
[92] X. Chen, T. Wu, G. Sun, H. Yu, Software-defined manet swarm for mobile monitoring in hydropower plants, IEEE Access 7 (2019) 152243–152257.
[93] K. Poularakis, Q. Qin, E. M. Nahum, M. Rio, L. Tassiulas, Flexible sdn control in tactical ad hoc networks, Ad Hoc Networks 85 (2019) 71–80.
[94] K. Poularakis, Q. Qin, K. M. Marcus, K. S. Chan, K. K. Leung, L. Tassiulas, Hybrid sdn control in mobile ad hoc networks, in: Proceedings - 2019 IEEE International Conference on Smart Computing, SMARTCOMP 2019, 2019, pp. 110–114.
[95] K. Poularakis, G. Iosifidis, L. Tassiulas, Sdn-enabled tactical ad hoc networks: Extending programmable control to the edge, IEEE Communications Magazine 56 (7) (2018) 132–138.
[96] P. Bellavista, A. Dolci, C. Giannelli, Manet-oriented sdn: Motivations, challenges, and a solution prototype, in: 19th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2018, 2018.
[97] M. Moshref, A. Bhargava, A. Gupta, M. Yu, R. Govindan, Flow-level state transition as a new switch primitive for sdn, in: HotSDN 2014 - Proceedings of the ACM SIGCOMM 2014 Workshop on Hot Topics in Software Defined Networking, 2014, pp. 61–66.
[98] Z. N. Abdullah, I. Ahmad, I. Hussain, Segment routing in software defined networks: A survey, IEEE Communications Surveys and Tutorials 21 (1) (2019) 464–486.
[99] H. Hartenstein, L. P. Laberteaux, A tutorial survey on vehicular ad hoc networks, IEEE Communications Magazine 46 (6) (2008) 164–171. doi:10.1109/MCOM.2008.4539481.
[100] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, H. Zedan, A comprehensive survey on vehicular ad hoc network, Journal of Network and Computer Applications 37 (1) (2014) 380–392.
[101] Z. He, J. Cao, X. Liu, Sdvn: Enabling rapid network innovation for heterogeneous vehicular communication, IEEE Network 30 (4) (2016) 10–15.
[102] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, E. Cerqueira, Towards software-defined vanet: Architecture and services, in: 2014 13th Annual Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET 2014, 2014, pp. 103–110.
[103] M. Abolhasan, J. Lipman, W. Ni, B. Hagelstein, Software-defined wireless networking: Centralized, distributed, or hybrid?, IEEE Network 29 (4) (2015) 32–38.
[104] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, M. Qiu, A scalable and quick-response software defined vehicular network assisted by mobile edge computing, IEEE Communications Magazine 55 (7) (2017) 94–100.
[105] S. Correia, A. Boukerche, R. I. Meneguette, An architecture for hierarchical software-defined vehicular networks, IEEE Communications Magazine 55 (7) (2017) 80–86.
[106] K. L. K. Sudheera, M. Ma, G. G. M. N. Ali, P. H. J. Chong, Delay efficient software defined networking based architecture for vehicular networks, in: 2016 IEEE International Conference on Communication Systems, ICCS 2016, 2017.
[107] Z. Y. Rawashdeh, S. M. Mahmud, A novel algorithm to form stable clusters in vehicular ad hoc networks on highways, Eurasip Journal on Wireless Communications and Networking 2012 (2012).
[108] I. Bekmezci, O. K. Sahingoz, S. Temel, Flying ad-hoc networks (fanets): A survey, Ad Hoc Networks 11 (3) (2013) 1254–1270.
[109] G. Secinti, P. B. Darian, B. Canberk, K. R. Chowdhury, Sdns in the sky: Robust end-to-end connectivity for aerial vehicular networks, IEEE Communications Magazine 56 (1) (2018) 16–21.
[110] F. Xiong, A. Li, H. Wang, L. Tang, An sdn-mqtt based communication system for battlefield uav swarms, IEEE Communications Magazine 57 (8) (2019) 41–47.
[111] N. Hu, Z. Tian, Y. Sun, L. Yin, B. Zhao, X. Du, N. Guizani, Building agile and resilient uav networks based on sdn and blockchain, IEEE Network 35 (1) (2021) 57–63.
[112] W. Qi, Q. Song, X. Kong, L. Guo, A traffic-differentiated routing algorithm in flying ad hoc sensor networks with sdn cluster controllers, Journal of the Franklin Institute 356 (2) (2019) 766–790.
[113] K. Chen, S. Zhao, N. Lv, W. Gao, X. Wang, X. Zou, Segment routing based traffic scheduling for the software-defined airborne backbone network, IEEE Access 7 (2019) 106162–106178.
[114] R. Bhandari, Survivable networks: algorithms for diverse routing, Springer Science & Business Media, 1999.
[115] Message queuing telemetry transport (mqtt) protocol, `https://mqtt.org/`, [Online; accessed 24-August-2021].
[116] I. Zacarias, L. P. Gaspary, A. Kohl, R. Q. A. Fernandes, J. M. Stocchero, E. P. De Freitas, Combining software-defined and delay-tolerant approaches in last-mile tactical edge networking, IEEE Communications Magazine 55 (10) (2017) 22–29.
[117] M. Castro, B. Liskov, Practical byzantine fault tolerance and proactive recovery, ACM Transactions on Computer Systems

20 (4) (2002) 398–461.