Software defined network based architecture to improve security in a swarm of drones

Christophe Guerber, Nicolas Larrieu, Mickaël Royer

▶ **To cite this version:**

Christophe Guerber, Nicolas Larrieu, Mickaël Royer. Software defined network based architecture to improve security in a swarm of drones. ICUAS'19, International Conference on Unmanned Aircraft Systems, Jun 2019, Atlanta, United States. art. no. 8797834, pp. 51-60., 10.1109/ICUAS.2019.8797834 . hal-02149924

HAL Id: hal-02149924

https://enac.hal.science/hal-02149924

Submitted on 6 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Software defined network based architecture to improve security in a swarm of drones

Christophe GUERBER
ENAC, Université de Toulouse, France
christophe.guerber@enac.fr

Nicolas LARRIEU
ENAC, Université de Toulouse, France
nicolas.larrieu@enac.fr

Mickaël ROYER
ENAC, Université de Toulouse, France
mickael.royer@enac.fr

***Abstract*—With the trend of developing more and more applications for Unmanned Aerial Vehicles (UAV), several research projects have considered new missions where single UAVs are replaced by swarms of drones. Although today regulations do not take into account such scenarios, implementation of an efficient security policy appears mandatory before operating a swarm of drones in open spaces.**

Consequently, this paper introduces an architecture for providing security features through the use of software defined network (SDN) technologies. To validate our approach, we compare the routing performance of our architecture with a classical solution based on the AODV routing protocol and the use of iptables rules. The results confirm the suitability of a SDN solution in this context. Finally, we present how it may be used to improve network security for a swarm of cooperative drones.

***Index Terms*—UAV, UAV Ad hoc NETwork UAANET, AODV, SDN, OpenFlow, swarm of drones, multi-agent system, security architecture, routing protocol**

I. Introduction

The expansion of Unmanned Aerial Vehicle (UAV) usage has drastically reduced costs involved in possessing and operating drones. New missions have emerged that implied not one, but several more or less autonomous UAVs collaborating in a specific mission [1]. Types of missions may vary from monitoring an area, to exploring inaccessible locations, to Search And Rescue or providing wireless network coverage without any ground infrastructure. These missions based on swarms of cooperating drones require additional network capabilities, both for ground to UAV and UAV to UAV communications.

Although we only consider civilian (as opposed to military) drones and missions, security has to be taken into account when deploying these UAV systems. First, we must guarantee that the system is secure and is operated in a safe way. Secondly, we have to protect our system against any attempt of denial of service, so that UAVs may successfully accomplish their mission.

Given the diversity of the types of missions mentioned above and the induced communication schemes (from very centralized ground-UAV to more diverse UAV-UAV dialogues), security measures will have to adapt from one mission to another, if not from time to time during the mission. As such, security mechanisms providing adaptability and the possibility of reprogramming would be desirable.

Ad hoc networks does not rely on any pre-existing infrastructure or access points as each drone and the ground control station (GCS) participate in both routing discovery and forwarding of data. Current implementations of ad hoc networks make use of traditional distributed algorithms for routing. Each node has a more or less parcellar knowledge of the surrounding nodes and takes routing decisions solely on a local basis, limiting the horizon of any security solution to local countermeasures. Nowadays, many solutions exist in wired networks to provide security mechanisms. Some recent approaches use SDN-based technologies [2] [3]. They make use of the centralized nature of the SDN architecture to aggregate a general view of the data flows, trying to detect and react to security threats. The success of SDN, however, has its roots in its high adaptability and flexibility in response to changes in data flows.

In this paper, we present a new architecture for unmanned aerial ad hoc network (UAANET) that can provide innovative security features and capabilities. Moreover, the use of SDN will allow a higher adaptability of the system and provide easier reprogramming.

The remaining of this paper is organized as follows. In Section II, we review some research relating to ad hoc networks, Software Defined Networks (SDN), security and monitoring. In Section III, we propose and discuss four alternative architectures providing the functionalities required for improving security in a swarm of drones, and we select two architectures to be compared. Section IV compares the two architectures from a performance point of view and provides some insight into how the proposed architecture may be used to improve the UAANET security. Finally, in Section VI we discuss our conclusions and make suggestions for future research.

II. Related work

A. Threats in a swarm of drones

[4] provides an in depth survey of several routing protocols usable in UAANET, providing a taxonomy of these protocols. It also shows different threats and security challenges inherent to ad hoc networks: wormhole and blackhole attacks.

Blackhole attack consists of corrupting route discovery in the routing protocol so as to place itself on the path of as many routes as possible. Once done, the attacking node may drop all or the majority of data packets and thus significantly degrade routing performances.

On the other hand, wormhole attack consists of two colluding attacks literally creating a worm hole within the ad hoc network by corrupting the route discovery process in order

to capture sensitive data. Both techniques imply forging false routing packets.

Like any other network, ad hoc networks may experience other attacks like Distributed Denial of service which are not specific to this kind of network, though the size of the network may be a limiting factor.

Finally, swarms of drones may also be attacked using techniques that do not specifically aim the network. Battery depletion, wireless channel jamming are efficient attacks that should be considered.

Though, [4] shows that most of the ad hoc routing protocols were not designed with security in mind, over the years, several proposals including security features were proposed. Most of them are dedicated to one specific threat or technique. None addresses application level attacks as this requires acquiring data at a system level.

B. Distributed monitoring tasks

Intrusion or anomaly detection heavily rely on monitoring the system under scrutiny. Statistics on different quantities has to be shared or reported to a central location in order to be classified.

[5] presents challenges of monitoring in Mobile Ad hoc Networks (MANET) and proposes a classification of the different solutions.

The basic approach uses Simple Network Management Protocol (SNMP) which requires the supervisor to request data from each node (polling technique) and is therefore very expensive in terms of bandwidth. Sometimes the nodes are provided with a monitoring policy so that they send the data without a request from the supervisor. Other techniques aggregate local data using clustering before transmitting them in a more efficient way. Finally, some solutions just use the data that are already available in routing protocols, thus do not require additional protocols for the collection of data.

[5] concludes by comparing the additional load induced by all the monitoring tasks in terms of CPU and network traffic:

- Clustering-based approaches diminish the number of monitoring messages using aggregation but require a costly cluster maintenance process;
- Policy based approaches give more autonomy to nodes and improve delays;
- Routing protocol based monitoring is efficient, due to the very nature of those protocols, similar to a monitoring task.

C. Software Defined Networks

SDN has become one of the main fields of research in networks a few years ago for wired networks. In SDN networks [6], control plane (where the network control resides) and data plane (where the user applications live) are separately considered. On the control plane, the nodes maintain a connection with the SDN controller through the control network. A typical SDN control network is a loop-less Ethernet network, while the data plane may have any topology with nodes behaving both as switches or as routers. The controller supports SDN applications providing instructions about the behavior of each node. SDN controllers thus have two interfaces: the southbound interface with the SDN switches, and the northbound interface with all the applications that will take decisions about anything happening inside the network (forwarding, filtering, etc.). The dialog between the SDN switch and the SDN controller may use the OpenFlow protocol [7]. One of the functions of a controller is usually to detect the topology of the data network using a discovery protocol, the most commonly used being Link Layer Discovery Protocol (LLDP) [8].

Previous research has worked on bringing wireless network to SDN, especially in vehicular ad hoc networks [9]. However, as vehicles are limited to the road network, Vehicular Adhoc Networks (VANET) are characterized by a heavy ground based fixed infrastructure where vehicle are mobiles. [10] proposes a hybrid architecture using SDN, but also depends on some ground infrastructure to cover a specific area. Of course, in the context of a swarm of flying drones, especially for missions we are considering, we do not have a heavy infrastructure.

If the initial implementations of SDN used very simple switches with as few functions as possible, some research trends tend to add some smarter functionalities inside the switch. After having added group tables allowing fast fail over behavior and QoS management in the lastest versions of the OpenFlow protocol, BEBA project suggested adding stateful behaviors [11]. Stateful behaviors allow a controller to specify how nodes behaviour dynamically adapt to time-varying flows and traffic behavior. They showed that though smarter switches are less generic, SDN can accommodate this scenario, improving the programmability of the network.

D. Attack detection and SDN

In the above described network security context, SDN architecture presents interesting features. It defines a central system where data can be gathered in order to identify system wide behaviors. Moreover, SDN being responsible for the routing, it naturally gathers information about the flows of data inside the network.

[12] provides a method to infer link utilization in a flow-based network like SDN, thus requiring no additional monitoring traffic. It captures and analyzes control messages. Accuracy depends on the amount of control messages and a good balance between this passive estimate and active monitoring has to be determined in the case of specific flow policies (e.g. proactive flows, long lasting flows, large timeouts for flows).

There are several research projects dealing with attack detections, both on the data plane and, in the case of an SDN based network, on the controller itself. The type of data used for anomaly detection is often bound to IP addresses, transport layer and transport ports. However, other types of data, like the flow graph have also been successfully used here. Current detection techniques may involve either mathematical properties and/or some machine learning algorithms.

[13] and [14] both address TCP SYN flood attacks, the objective of which is to consume resources from the target

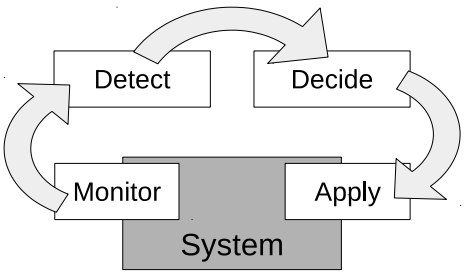

Figure 1. Detection and response process

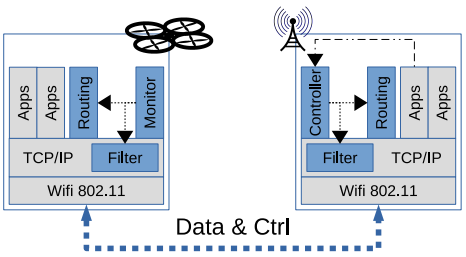

Figure 2. AODV architecture with iptables

up to a point where it will become unresponsive for any new request. They take advantage of the central position of the controller to identify malicious behaviors using counters and thresholding.

[15] uses traffic dispersion graph analysis to detect anomalous patterns of flows like in DDoS. The proposed approach is based on static and dynamic graph metrics and an algorithm performing graph comparison. Using traffic databases like CAIDA[1], they generate patterns of malicious behaviors to be compared with real time traffic. They successfully identify several DDoS attacks in traces from POSTECH university[2].

III. Design of a SDN-based security architecture

In our approach of swarms of cooperating UAVs, the system under scrutiny is the network part that allows several user applications to exchange data. We consider both cooperation protocols like multi-agent system network i.e. UAV to UAV communications (e.g. market based protocols) and mission related traffic like sensors i.e. UAV to ground control station or command and control (C2). In addition to actually provide routing inside the swarm of drones, we are studying how to improve security within this network. This includes ”outsiders” trying to impair the capability of the network to provide its service, but also the risk of ”insiders” having the same unwanted behavior, e.g. after having been hacked.

A. Security architecture characteristics

A typical detection and response process like the one that we are studying here, is composed of the following building blocks (see Figure 1):

- One set of measures of the system under scrutiny;
- Some specialized algorithm to detect undesired behavior;
- One decision process to identify an appropriate response;
- A specific pattern to apply the identified response.

This paper focusses on the first and last steps of the above process: monitor and apply. Our intent is to define an efficient way to monitor the network activity within the swarm of drones, and to be able to deploy a response to a hypothetical attack against the swarm and its mission.

When considering appropriate security policy and countermeasures, it is usually based on the followings (although there may be others):

- Protection against physical access;
- Providing sets of rules of authorized transmissions and behaviors;
- Authentication and integrity checking through credential exchanges;
- Hiding critical information from the outside world as far as practically possible;
- Isolation and segmentation between different elements of the system which limit contagion and allow definition of the above mentioned legitimate rules.

Protection against physical access in a wireless network is of course illusory. Hiding data from the outside world also makes wireless a special and challenging case: the antenna of any station in the coverage of a node’s radio receives all the transmissions of that node.

The first step of our work was to define architectures providing some security features that are not limited to supporting an intrusion detection and response process. In addition, it must provide comparable performances (in terms of availability and delays) with current routing protocols for ad hoc network. Adaptability and programmability are also features that we should take into account. We have defined the following four architectures, going from today available routing protocols to innovative proposals.

B. Set of security architecture designs

1) *AODV [16] routing with iptables filtering:*
This architecture uses an ad hoc routing protocol. We chose AODV mainly because it was shown that it provides better performances for UAANET [17]. Here is a description of the operation of this architecture with a short description on how to implement required functions for securing the network.
AODV nodes announce themselves to the surrounding nodes using Hello packets. Each node monitors the neighboring nodes by receiving Hello packets and activating timers so as to detect their absence.
A node having to send a packet to destination D for which it has no fresh route, will issue a Route Request (RREQ) to the surrounding nodes. RREQ are retransmitted by surrounding nodes if they have no route to the requested destination D. The originator may control the depth of retransmission through the TTL value in the RREQ by a so called expanding ring search technique.

[1]http://www.caida.org/
[2]http://postech.ac.kr/

Whenever a node has a route to the destination D, either because it is the destination or because it has currently a fresh route to it, the node responds with a Route Reply (RREP) to the originator. This is made possible as the nodes receiving a RREQ will cache a route back to the originator. When a node detects the loss of neighbor node that is part of an active route, a route error (RERR) packet is sent in order to inform the other nodes of the loss of connectivity. These routes must then be re-discovered.

Routes are maintained a short period of time if no activity is detected for each active route (3 seconds). For each packet received or sent using a route, keepalive timers for the forward and reverse routes are restarted.

AODV does not provide monitoring or supervision capabilities by itself. We will have to choose a dedicated protocol like SNMP, or modify AODV for this purpose, or define a new specific protocol, in order to collect surveillance data. Some data may be collected through AODV. Regardless of the technique that is employed, this will require specific transmissions and fine tuning to balance increased traffic load and detection accuracy. Policy based monitoring may also be used here.

AODV does not provide filtering capability neither. But filtering capabilities are usually available on the operating system (OS) like in Linux's iptables. So we could implement some specific protocol or adapt an existing one in order to deploy iptable (or equivalent) rules to the different nodes.

Outside threat may be managed through AODV security improvements, for which a public key infrastructure[3] (PKI) is to be defined.

Figure 2 shows a typical AODV architecture along with additions to be brought so as to provide monitoring and security actions. As presented in Section III, we need some distributed monitoring function and a way to deploy the identified response, e.g. in the form of iptables rules.

2) *SDN with wide range network as control network:*

As stated in previous section, SDN provides some interesting features for the kind of problem that we are facing here.

In this architecture, the loop less control network is provided by a wide range network N. This architecture is thus very close to a typical SDN network. Network N provides connectivity between the controller and every node in the coverage, while a second network M with a shorter range provides the data network.

As an SDN based architecture, flow entries may be considered as a sort of authorization for a packet to go through the network. SDN flows will take the role of the iptables of the previous architecture.

[3]Public key cryptography requires the ability to verify the identity of a peer station. Reliable verification of the identity is based on certificates management. The Public Key Infrastructure is a system for the creation, the storage and the redistribution of those certificates.

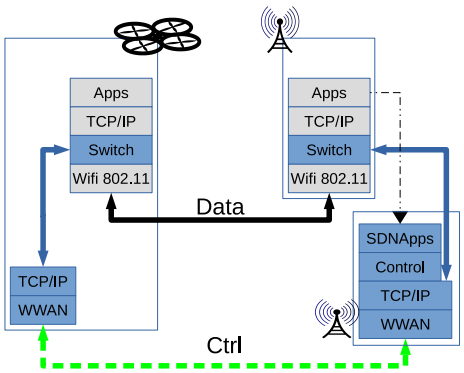

Figure 3. SDN architecture

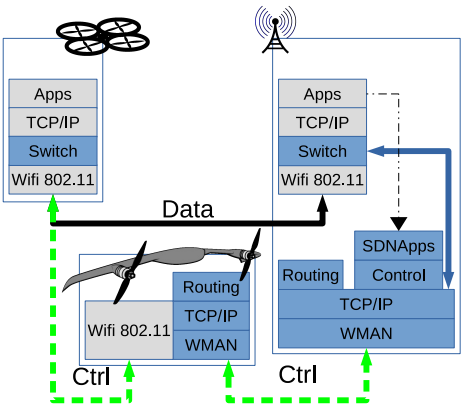

Figure 4. Hierarchical SDN architecture

In addition, SDN protocols allow for collecting statistics on the flows and switches in the network. It is also possible to implement new statistics that may be gathered through SDN protocols.

Outside threat is managed through secured connection and the security features of the network M if any.

Figure 3 shows this new architecture. SDN applications will rule the behavior of our network, gather statistics trying identifying and detecting attacks. Decided responses will be sent back to the SDN switches of the swarm.

3) *Hierarchical SDN using multiple networks:*

As wide range network may not be available everywhere, we consider here a hybrid and hierarchical network using a medium range network N to some intermediary UAVs acting as relays/routers between the ground station and the rest of the swarm of drones. This may also allow defining several controllers or delegating parts of the responsibility to intermediary UAVs (see Figure 4).

We keep the same requirement and additions as in the previous case, but do not rely on a wide range network anymore.

4) *SDN with AODV routed control plane:*

As the use of a control dedicated network would limit the possible scenarios for the previous architecture, we

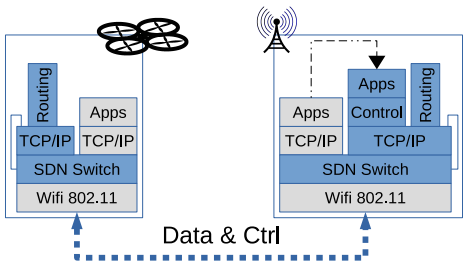

Figure 5. In-band AODV SDN architecture

propose to use OpenFlow in-band wise: control plane and data plane use the same network. Of course, we then need a routing protocol to allow each node to establish southbound interface connection with the controller, as shown in Figure 5. We have chosen AODV for routing purposes on the control plane, while SDN will provide forwarding on the data plane, both using the same ad hoc network. Indeed, routing in the control plane will be limited to UAV-to-GCS and GCS-to-UAV routes. There is no need for a more complex routing protocol.

In addition, network topology discovery will not be based on some active protocols like LLDP as it would be in a typical SDN network [18]. This would require regular sending of packets from the controller to a sending switch and from the receiving switch back to the controller. So as to limit the amount of data sent for this purpose, we propose to base our knowledge about the topology on the neighborhood information from AODV protocol. A node will thus inform the controller about its known neighborhood at the moment it establishes the OpenFlow connection, and will also inform him about any update: new neighbor appearing or current neighbor disappearing. OpenFlow allows experimental addition to the protocol almost without impact on the switch and controller, except the parsing, the processing and the generation code for the new interactions.

We thus define four interactions using experimental extension to OpenFlow:

- AODV_STATUS_REQUEST,
- AODV_STATUS_RESPONSE,
- AODV_ADD_NEIGHBOR and
- AODV_DELETE_NEIGHBOR.

The first two are used on switch connection establishment, the last two to update controller's knowledge about the neighborhood of the node.

Each node (UAV) contains both a router and a host. The SDN switch will behave as a router and the payload applications as host applications. Routers (i.e. AODV, control part of SDN) and hosts live on two separate IP networks which allow for an easy identification of flows. As in previous architectures, SDN will provide a standard set of statistics and a flow graph that may be used for detection. In addition, new specialized statistics may be defined.

SDN switch and controller are connected through a secured transport connection. Credentials management is eased by the definition of a controller node.

C. Security architecture design selection

As our intent is to cover medium to small UAVs, we have to consider the aforementioned architectures in a constrained context in terms of dimension and battery consumption. Especially, providing two wireless technologies on a small hardware platform will impose physical constraints for installing antennas while guaranteeing correct coupling for both technologies to operate without interferences. In addition, long range radios usually require more battery power to operate than shorter range ones. As such, we will not consider those architectures requiring a second network onboard the UAVs, especially if they are long range. Moreover, wide coverage wireless networks would limit the scenarios where such a coverage actually is available. We will thus put the 2nd architecture aside.

Although hierarchical network seems to provide some advantage over the wide range network through the usage of relays to the second wireless network, it will require the operator to possess and operate a mixed fleet of UAVs with dedicated missions for the "heavier" UAVs and will limit the type of mission and operator.

As a consequence, we will focus on the first (AODV routing with iptables filtering) and the last (SDN with AODV routed control plane) architectures.

IV. Security architecture designs' performance evaluation

A. Implementation details

This section will describe implementation choices and details of the architecture principles described above: AODV for routing OpenFlow on the control plane and a SDN managed data plane.

AODV will consider a route as invalid if it is not used for a few seconds. According to Section 6.2 of [16], the lifetime of an active route may be either determined by the control packet i.e. the RREP, or initialized to ACTIVE_ROUTE_TIMEOUT value of which is 3 seconds. Instead of increasing the lifetime, we chose to guarantee that some traffic will be generated between each UAV and the controller. This may be done either through OpenFlow Hello packets, or through TCP keep alive. As Hello packets are also used in OpenFlow to detect loss of communication, we preferred TCP keep alive over the OpenFlow option.

In a typical SDN based network, loss of connectivity between two switches will be entirely managed by the controller. Controller's topology discovery function will identify the loss of connectivity and react by changing the flow entries inside the different switches to repair the flow of data within the network [18]. As described in architecture description in Section III-B, instead of LLDP, we propose to use AODV knowledge about its direct neighborhood. More precisely, the topology changes when a neighbor disappears will be announced to

the controller through the above mentioned AODV delete interaction. New neighbors are announced through an add interaction.

Topology changes must be taken into account as soon as possible. We have to avoid waiting for the controller to receive and react to the topology change events. Especially, a node detecting a missing neighbor shall not continue sending data to this neighbor. Deletions should be taken into account immediately to avoid sending data to a node that we know is too far away. This means that the AODV daemon will have to automatically delete the flow entries that concerns a node for which the Hello timer has expired.

In OpenFlow 1.3, flow deletion command only allows flow selection based on their match fields. Thus, it is not possible to delete a flow entry based solely on the destination of the flow: destination is determined through the actions of a flow entry (port and MAC address). However, OpenFlow provides a cookie field in each flow entry that can be used for selection of flow entries to be deleted. When the controller creates an outgoing or a forwarding flow entry, the value of the cookie represents the next hop on the route. When a node detects the loss of connectivity with one of its neighbor, it will generate a delete command for all the flow entries having the required cookie value.

B. Evaluation testbed implementation

The choice of the different building block of our evaluation testbed took into account the following guidelines:

- We intended to be able in the short future to make some field testing, so pure simulation software was not an option,
- We needed to modify easily the different software, so limit the amount of code dealing with the kernel,
- Field proved implementation as far as practically feasible.

For the above listed reasons, we took AODV-UU implementation [19] that provides an efficient and tested implementation of AODV and we updated it in order to work with recent version of Linux kernel.

The SDN switch, is derived from the OFSoftSwitch13 [20] that proved to be easily modifiable. This user mode SDN switch provides an almost complete 1.3 OpenFlow connection and has been used in recent research projects like BEBA.

Finally, the controller part will be based on Ryu SDN framework [21]. It supports OpenFlow 1.3, is easily modifiable as it is Python based and thus allows using a lot of libraries including graphs, math and machine learning.

We implemented some specific OpenFlow interactions to convey AODV neighborhood discovery from the switches to the controller and included AODV daemon in the OpenFlow dialog.

We deployed a testbed using several PCs under Linux to act as nodes (UAVs) in our network. One of these PCs will be the ground control station. The wireless network has not been simulated but replaced by a wired network. Although this limits the representativity of the physical layer, it allows for an easy control on the visibility between the nodes and thus gives a full freedom when it comes to defining a topology or a dynamic scenario. Of course, delay measurement are not considered in absolute value but only relative to the other architecture.

The SDN controller hosts a very simple and basic SDN application that provides the shortest path for each received PacketIn. Proactive routing where the controller may install flows beforehand of any network traffic and reactive repair where active flows are repaired immediately on topology modification, were not implemented. No backup routes were added. The controller tries to do as good as AODV and may shorten the path like AODV.

Finally, a simulation management station will follow the simulation scenario, starting the different nodes, replaying network traffic using a capture file and downloading log files, captures and node states for offline processing. In addition, nodes are NTP synchronized to help post-processing and log correlation. Time synchronization allows a less than 1ms accuracy between two nodes (relative synchronization rather than absolute time).

C. Evaluation scenarios

After having tested static scenarios to validate our implementation, we developed a realistic dynamic scenario. We will thus focus on the latter as it represents our goal scenario: a swarm of mobile drones during a realistic mission.

So as to describe the scenario, we will first present the mobility scenario and show how it represents a realistic operational scenario. Then we will describe the network traffic used during the tests.

Mobility scenario

The mission that we consider aims at observing a remote location using onboard cameras. The scenario contains 3 steps that will be played in round trip: starting on the ground, transiting to the mission location, and finally on site for the mission. These three steps are depicted in Figure 6. On the ground (1), all the UAVs have connectivity with each other and with the ground station. During the transit to the mission location (2), some UAVs may lose connectivity with the ground station. Once the UAVs arrive on site for the mission (3) they are considered as being as far as possible from each other, given the radio range provided by the wireless network. Once the mission has ended, drones come back to the ground station through the exact same steps.

In the above referred figure, plain lines represent radio connectivity i.e. the two stations are in the radio range of each other. As in a wired IP network, packets from a node to another node with which it has radio connectivity will be sent directly. Packets from a node to another node with which it has no radio connectivity are sent to a third node (the next hop) which will then forward the packet closer to the destination. Thus, routes in a dynamic topology will have to adapt to changes in the most efficient way.

The scenario does not consider the event of a complete loss of connectivity for any of the UAVs. It slowly changes the topology. As explained here below, we will focus on data

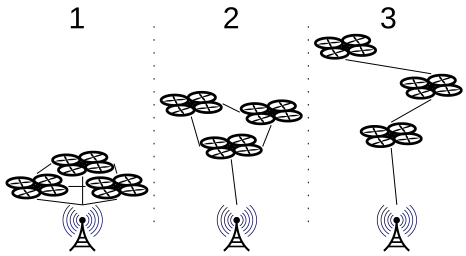

Figure 6. Dynamic scenario topology evolution

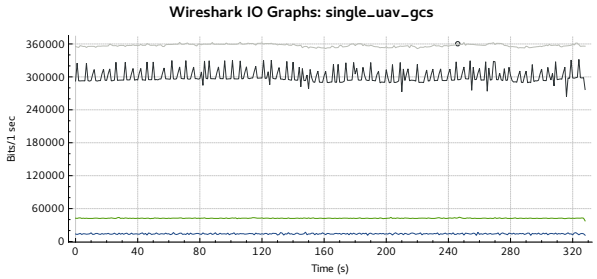

Figure 7. Traffic characterization (from the lowest to highest: telemetry, C2, video, moving average of total data)

exchanges taking place between the GCS and the UAV which will be the farthest from the GCS in order to evaluate the worst case.

In order to ease the following description, we will number UAVs using the distance from the GCS, when the swarm is in the mission configuration (3). UAV1 is the only to have connectivity with the GCS. UAV2 has connectivity with UAV1 and UAV3, thus UAV3 only has connectivity with UAV2.

As far as GCS and UAV3 nodes are concerned, our scenario will generate four mobility events. This is slightly more than strictly necessary given the mobility scenario that we described here above. At the beginning, UAV3 and GCS are in visibility of each other. The first loss of connectivity event happens when the UAV3 flies too far away from the GCS. The second is when this same UAV3 will be too far away from UAV1.

On the way back, we generate an "artificial" loss of connectivity, considering that UAV3 and UAV2 will fly far away from each other. The last "artificial" loss of connectivity is supposed to take place when all the UAV are back to the station. Here, UAV3 will lose connectivity with UAV2. These artificial mobility events have been added on purpose, to stimulate the architecture and protocols.

Network traffic scenario

We decided to simulate the network traffic by replaying real network traffic. Thus, the data are representative by themselves of a UAV based video capture mission. Although it may not be representatiave of all kinds of possible mission, it is representative of a majority of the missions where a primary stream flows from the UAVs to the ground control station.

We made the capture between the GCS and a real UAV that was broadcasting the video stream of its camera. We used the open-source software Paparazzi[4] for autopilot and ground station, generating C2 and telemetry traffic. The capture contains several flows of data: a 10 images per second UDP/RTP video stream compliant with [22], telemetry and other traffic sent from the UAV to the ground control station (GCS) and commands sent from the GCS to the UAV. Figure 7 shows the different streams of data between the UAV and the GCS. It shows that video stream is about 300kbps, telemetry is about 14kbps and C2 is about 42kbps. Most of the transmissions are small UDP packets, while video transmissions are composed of 1,500 bytes packets. In total, UAV and GCS generate roughly 356kbps of throughput including the protocol overhead.

In terms of packets per second, the GCS generates about 95 tiny (less than 27 bytes of application data) packets per second while the UAV generates 45 packets per second.

In the simulation scenario, we will play this exact same capture for each UAV of our topology. The ground station will thus receive the video stream of the three UAVs, along with the three telemetry data and will send three times the command stream.

Although it might be anticipated that a swarm of drones may not need a command for each individual drone, we argue that this still makes our simulation scenario representative of a today realistic scenario.

D. Evaluation results

The results presented here below show a comparison between the performances of the two studied architectures: pure AODV architecture and In-band SDN with AODV routing. We conducted several measurements on the performance of the proposed architecture in a static topology, especially for validation.

Simulations have been made on the testbed described in Section IV-B, using the operational scenario presented in the previous section. Measurements were made offline by post-processing the capture files obtained from different locations in our network.

The comparison focuses on the performances of each architecture in a realistic and thus dynamic scenario. Due to the implementation choices, maximum throughput proved to be difficult to measure with any degree of reliability. We will not consider any security benefit yet. The use of SDN in a security context is discussed in Section V.

Route acquisition time measurements would require a dedicated scenario to artificially generate numerous route acquisition events and will consequently not be considered in our scenario. In addition, route acquisition time very much depends on the kind of route to establish (e.g. from GCS to UAV vs. UAV to UAV) in both architectures. It would be difficult to compare in an operational context as we cannot characterize these parameters.

As the starting point of this simulation is a dynamic scenario with several UDP based streams, we will measure the

[4]http://www.paparazziuav.org/

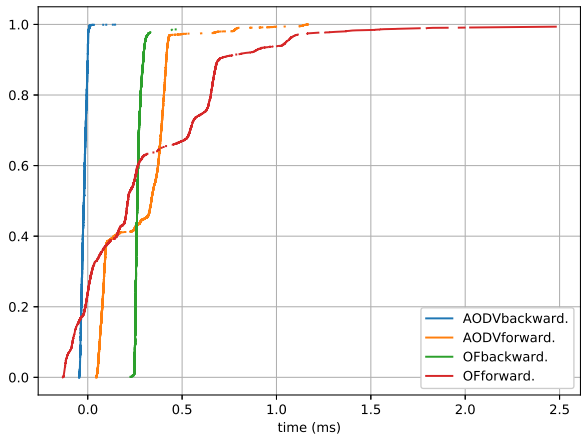

Figure 8. CDF of delay distribution

Table I
LOSS OF COMMUNICATION DURING MOBILITY

		AODV		SDN	
direction	**Event**	**duration**	**packets**	**duration**	**packets**
GCS to UAV	1	1.012s	97	1.431s	136
(95pps)	2	1.149s	110	1.964s	188
	3	0s	0	1.169s	112
	4	0s	0	1.460s	141
UAV to GCS	1	1.008s	52	1.396s	73
(48pps)	2	0.449s	25	1.943s	96
	3	1.159s	52	1.104s	60
	4	0s	0	0s	0

Table II
SIZE OF CAPTURED DATA

	AODV	SDN
Throughput	366 kbps	374 kbps

performance difference for end-to-end delay in the worst case, and the loss ratio due to the dynamic aspect of the topology. The former will show the cost of providing additional services compared to the pure AODV performance and whether or not our architecture is usable in an operational context. The latter will show how well our architecture deals with dynamic topology compared to a routing protocol dedicated to dynamic network like ad hoc nets.

In addition, we will compare the amount of data generated by both architectures. This will provide information on the overhead of the SDN solution in terms of bandwidth consumption.

Figure 8 shows the delay distribution for our realistic scenario in the case of pure AODV routing. AODV forwarding uses kernel's highly optimized standard IP protocol forwarding functions and forwarding delays are very low. A few values indicates negative delay, which means that we are below the time synchronization and time measurement accuracy. In AODV, losses are mainly due to neighborhood detection time. AODV takes between 1 and 2 seconds to detect the absence of AODV Hellos. During this time, AODV sends continuously packets using the obsolete route. As soon as one loss has been detected, a new RREQ is issued and any new transmission is queued, until the reception of an RREP which allows sending all the queued packets. Given the size of our network, this takes usually a few tens of ms.

Figure 8 shows the measured delay distribution using the AODV/SDN architecture. Synchronization between the PCs may have changed compared to the AODV measures. The performance difference with AODV is within a millisecond although about 10 percent of the packets experience slightly longer delays. These delays are mainly due to the controller until it has installed all the necessary flow entries.

Our SDN architecture is based on AODV and relies on this protocol for neighborhood loss detection. As a consequence, we do not lower detection time. Once detected, the controller will be informed as soon as the TCP OpenFlow connection is re-established. New packets will be sent to the controller using PacketIns through the TCP connection.

Due to the reliability of the TCP connection, new PacketIns might be delayed if ever an upstream TCP segment has been lost. OpenFlow suggests to use auxiliary connectionless transport but neither Ryu nor the SDN switch of our choice supports auxiliary connections.

In addition, the SDN switch of our choice is highly customizable but is known to offer average performances compared to other less customizable ones. Although we included part of the improvements developed by the BEBA project in the OFSS13 [23], the performance of our switch appeared to be lower than production SDN switches like Open vSwitch [24]. Especially, packet processing (matching and flow entry processing) is done in user space and is thus more sensitive to CPU usage variations.

Finally, controller performances have a strong impact on the performances of the architecture on the very first packets of a stream. Measurements of the controller and application part show delays that may be up to 15ms.

All these considerations explain why SDN architecture may experience some increased delays from time to time. However, packet forwarding is still low on our testbed (i.e. without any consideration of the physical layer) and compares with AODV.

Concerning mobility management, results are shown in table I. The table shows, for each of the four events, and both directions, the duration of the loss of communication and the number of packets that have been lost during the simulation. This happens two times for GCS to UAV traffic. When UAV3 flies closer to the GCS, it becomes successively neighbor with UAV2 and then GCS while both are on the data path from GCS to UAV3. These two events are thus automatically shorten by AODV and the loss of a link on the old path is then transparent. On the reverse path, it is only when UAV3 is back in the neighborhood of the GCS that AODV shortens the path.

As shown, AODV is also more efficient when it deals with mobility. Especially, routing is updated as soon as the

destination becomes a neighbor in order to use the shortest path. This does not require a new RREQ/RREP exchange. This explains why mobility events may not be noticeable in some cases.

On a contrary, our basic SDN application does not try to optimize routing on mobility events. This explains why SDN architecture suffers of more losses of communication. This limitation could easily be circumvented by using a smarter SDN application.

As shown in Table II, the flexibility brought by SDN comes with some overhead in bandwidth consumption. The difference between the amount of data exchanged between both nodes is increased by slightly more than 2 percent in SDN compared to pure AODV. As a conclusion, the experimentations described here show comparable performances in both architectures in terms of end to end delay. Once the route or flow is determined, forwarding is not significantly slower in SDN: difference between SDN and AODV is below 1ms.

In the SDN architecture, the use of AODV both as routing on the control plane and as source of mobility management through the generation of OpenFlow events allows a good behavior of the controller. Controller is not responsible for the topology discovery anymore and receives updates about the topology in a timely manner. Especially, as OpenFlow connections are kept active, a spanning tree like routing is put in place and allows for a node to move without having to reestablish the route down to the destination. Except maybe in very erratic topology changes, any neighbor will already have a route to the destination and new routes are detected very efficiently. However, if no route can be established with the controller, the drone is separated from the rest of the swarm and no application data exchange can take place.

V. How SDN can improve UAANET communication security?

The challenges for improving the security in an ad hoc network inside a swarm of drones are manifold and we do not pretend to solve all of them. However, the proposed architecture exposes interesting features that may become foundations or building blocks for a security policy for the intended scenarios.

Figure 9 illustrates the discussion below. A swarm of drones with mission data exchanges are shown on the left part of the figure. The implemention elements of our architecture are depicted on the right part. Thin dashed lines show on which hardware the software or device is hosted. Especially, the mission control software, the SDN controller and a SDN switch are colocated on the same host.

First of all, OpenFlow is supposed to use a secured transport connection (i.e. Transport Layer Security TLS). Configuration of a swarm of drones with this technique thus implies that each node has the credentials of the controller node, and the controller, the credentials of all the nodes. We do not need to let all the drones know each other. But the controller is now a trustful node and may become a source for credentials and may serve as a building block for a PKI.

Securing AODV seems of primary importance in our architecture, as it is the prerequisite for switch-to-controller dialog. Especially, it is sensible to some routing attacks, like wormhole and grey/black hole attacks. Proposed solutions for securing AODV do not study how to implement a PKI in a UAANET. As shown here above, the proposed architecture may provide this features through the controller. It may either serve nodes with credentials when requested, or verify credentials on behalf of the nodes, or even push all new credential as the nodes authenticate through the TLS secured OpenFlow connection.

Additionally, we may note that the new architecture changes the nature of the hijacked traffic by wormhole techniques. It is still to be proven if the SDN applications as defined here are made insensitive to AODV wormhole attacks or not. If it is not, wormhole may still capture traffic but will be limited to control data. Though these data are of high criticality, they will be protected by TLS and their real time nature.

Then, payload applications may be isolated from the more sensible routing parts through Linux namespace techniques and control traffic will be filtered to only allow OpenFlow and AODV to traverse. By doing this we lower the attack surface of the control part and limit the contagion from the payload applications to the control part.

Finally, each node will request flows from the controller and will report some statistics to the controller. OpenFlow provides some common standard statistics but also allows definition of new statistics. Hence the controller gathers effortlessly a lot of network information (neighborhood, network and flow graphs, flow statistics, etc.). In addition, common UAV applications and payload applications will provide additional information that will be easily accessible to the SDN controller (geographical position, battery charge, etc.). Convergence of all these data at the controller location gives us an opportunity to apply machine learning methods with the intent to detect and identify several attacks on the swarm of drones, and produce a protection/prevention strategy. Our SDN based architecture then comes handy to apply such a strategy from a network routing point of view. In addition, it may allow a global representation of the routing situation to a human supervisor.

Actions that our new architecture will allow include both restricting payload application network activity by filtering source and/or destination of part or each of the flows, and some limitations on the control plane. Limiting network traffic on the data plane allows stopping or limiting the impact of e.g. DDoS attacks. Filtering on the control plane will allow limiting the possibility of the unsafe applications to act on the control plane e.g. by generating huge amount of unknown flows, thus producing a lot of Packet-In events on the control plane, or prohibiting a node to send an AODV route reply.

It may be argued that the centralized nature of SDN may expose the central node to specific attacks like PacketIn flooding. Obviously, SDN applications in our architecture will have to take this threat into account. However, the ground control station actually is a central node even without SDN. Though our architecture increases the threat level on that node,

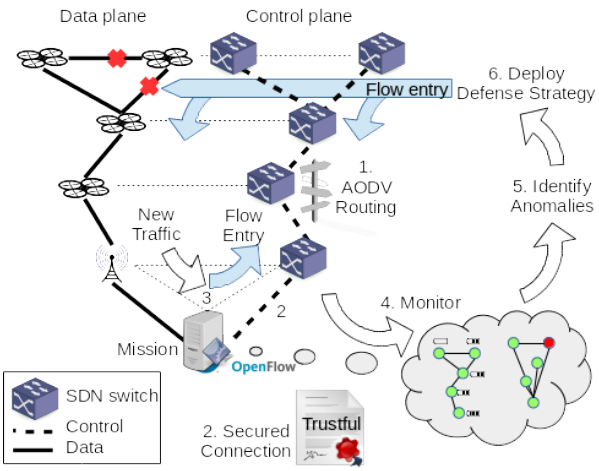

Figure 9. Security use case for SDN

it does not create a new risk, and security measures to be enforced will have to be outside of the network domain.

VI. Conclusion and Future Work

There are few existing solutions to improve security within swarms of drones and they are usually able to respond only to a given type of attack. In this paper, we have put forward different architectures capable of improving security and selected two that suit the most constrained drones: the first a traditional ad hoc routing protocol to which we would add iptables based filtering, the second a novel in-band SDN based architecture. The former is a well known approach for ad hoc networks. The latter introduces SDN to the UAANET providing all the flexibility that SDN can bring to wired networks. It can be deployed on constrained platforms due to its simplicity, and the use of this architecture spans several network domains.

To test both architectures, we designed a testbed consisting of several drones and a control station capable of running operational scenarios, which applies both architectures to real systems. We defined a testing scenario representative of an operational mission including mobility, and real C2 and mission data. Our evaluation was that they have equivalent performances in terms of delays and losses during mobility, providing the required performances at the cost of some bandwidth.

We propose several uses of our new architecture in the network security domain: to enforce security rules in response to an attack, to take advantage of the secured connection with the ground station and to use SDN as a basis for supervising the activity within the network.

We plan to continue improving the current AODV/SDN implementation and especially the SDN application. So far implementation has focused on the performance evaluation, we intend to evaluate security application of the SDN architecture: use of SDN flow tables for filtering, supervision and data gathering.

We also intend to implement our AODV/SDN architecture on a group of UAVs for field evaluations.

References

[1] I. Bekmezci, O. K. Sahingoz, and S. Tamal, "Flying Ad-Hoc Networks (FANETs): A survey," *Ad Hoc Networks*, vol. 11, no. 3, p. 1254–1270, 2013.

[2] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research trends in security and ddos in sdn," *Security and Communication Networks*, vol. 9, no. 18, pp. 6386–6411, 2017.

[3] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, Jun 2018.

[4] J.-A. Maxa, M.-S. Ben Mahmoud, and N. Larrieu, "Survey on UAANET Routing Protocols and Network Security Challenges," *Ad Hoc & Sensor Wireless Networks*, Mar. 2017. [Online]. Available: https://hal-enac.archives-ouvertes.fr/hal-01465993

[5] N. Battat, H. Seba, and H. Kheddouci, "Monitoring in mobile ad hoc networks: A survey," *Computer Networks*, vol. 69, pp. 82 – 100, 2014.

[6] *SDN Architecture*, Open Networking Foundation, Jun. 2014.

[7] *OpenFlow®Switch Specification Ver 1.3.5*, Open Networking Foundation, Apr. 2015.

[8] "Station and Media Access Control Connectivity Discovery," IEEE 802.1AB, Institute of Electrical and Electronics Engineers, Sep. 2009.

[9] I. Ku, Y. Lu, M. Gerla, F. Ongaro, R. L. Gomes, and E. Cerqueira, "Towards software-defined vanet: Architecture and services," 2014.

[10] M. Abolhasan, J. Lipman, W. Ni, and B. Hagelstein, "Software-defined wireless networking: Centralized, distributed, or hybrid?" vol. 29, 07 2015.

[11] G. Bianchi and A. Capone. (2014) From dumb to smarter switches in software defined networks: an overview of data plane evolution.

[12] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "Flowsense: Monitoring network utilization with zero measurement cost," in *Passive and active measurement (PAM)*, 2013, pp. 31–41.

[13] S. Fichera, L. Galluccio, S. C. Grancagnolo, G. Morabito, and S. Palazzo, "OPERETTA: An OPEnflow-based REmedy to mitigate TCP SYNFLOOD Attacks against web servers," *Computer Networks*, vol. 92, pp. 89 – 100, 2015.

[14] R. J. R. Mohammadi and M. Conti, "SLICOTS: An SDN-Based Lightweight Countermeasure for TCP SYN Flooding Attacks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 487–497, June 2017.

[15] D. Q. Le, T. Jeong, H. E. Roman, and J. W. Hong, "Traffic dispersion graph based anomaly detection," in *Proceedings of the 2011 Symposium on Information and Communication Technology, SoICT 2011, Hanoi, Viet Nam, October 13-14, 2011*, 2011, pp. 36–41.

[16] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, Jul. 2003. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3561.txt

[17] J.-A. Maxa, G. Roudière, and N. Larrieu, "Emulation-Based Performance Evaluation of Routing Protocols for Uaanets," in *Nets4Aircraft 2015*, ser. Nets4Cars/Nets4Trains/Nets4Aircraft 2015, vol. LNCS, no. 9066. Sousse, Tunisia: Springer, May 2015, pp. 227–240.

[18] S. Khan, A. Gani, A. W. A. Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 303–324, Firstquarter 2017.

[19] AODV-UU. [Online]. Available: https://github.com/erimatnor/aodv-uu

[20] Basic OpenFlow Software Switch (BOFUSS). [Online]. Available: https://wiki.sdn.ieee.org/display/sdn/OFSS13

[21] Ryu component-based software. Ryu SDN Framework Community. [Online]. Available: https://osrg.github.io/ryu/

[22] L. Berc, W. Fenner, R. Frederick, S. McCanne, and P. Stewart, "Rtp payload format for jpeg-compressed video," RFC 2435, Oct. 1998. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2435.txt

[23] N. Bonelli, G. Procissi, D. Sanvito, and R. Bifulco, "The acceleration of ofsoftswitch," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 1–6.

[24] Open vSwitch (OvS). The Linux Foundation. [Online]. Available: http://www.openvswitch.org/