RDSNET: A proposal for control architecture for software defined MANETs

Sergio Mora[1], Jhon Vera[2]
[1]Associate professor, Research Department, Universidad ECCI
Cl. 51 #19-12, Bogotá, Colombia
smoram@ecci.edu.co
[2]Associate professor, Research Department, Universidad ECCI
Cl. 51 #19-12, Bogotá, Colombia
jverav@ecci.edu.co

Abstract—**Software-defined networks (SDN) have quickly gained importance in telecommunications. This paper has examined the performance of SDN's techniques used in wireless networks, specifically in ad hoc mobile networks (MANET). The architecture of SDN control specially designed for MANETs has been proposed under the name RDSNET. RDSNET provides a solution to the design and deployment of the controllers in these networks. Several simulation scenarios were built in OMNET ++ to decide whether the design of a distributed or centralized controller was the most suitable for implementation. The results showed that the RDSNET distributed configuration yielded better performance. It was also shown that, despite the additional overload introduced by synchronizations, the performance was similar to a conventional MANET without SDN for the message success rate. However, the latency with RDSNET does not improve over SDN networks. Thus, though flexibility is the advantage of SDN for MANETS, latency is the main disadvantage.**

Keyword - Software defined networking, Control, MANET

I. INTRODUCTION

Software-defined networks are a new paradigm of network architecture. They have aroused great interest in the telecommunications community lately. The SDN permit better control and administration, rapid reconfiguration, and dynamic traffic. They also have the potential to evolve and facilitate innovation in networks [1]. These characteristics make them interesting to companies and service providers enabling them to build highly scalable, flexible and adaptive networks that cater for the different needs of the industrial sector.

Research carried out with SDN so far has focused mainly on wired networks and wireless networks [2-3].However, little attention has been paid to the use SDN in ad-hoc mobile network (MANET). There are potential benefits in the use of SDN in the MANETs, where the characteristics of the SDN could be immensely useful. Such benefits must be weighed against the cost of wireless channel solutions needed to replace the conventional wired SDN implementation.

Most works related to wireless SDN in cellular networks,however, their use has been extended to several types of networks, for example, the authors of [4] identified use cases for SDN in cellular networks and proposed interesting extensions to SDN wired that overcome some inherent limitations to this type of network. For wireless sensor networks (WSN), SDN architectures such as those introduced in [5] have been presented, although in this work they do not present simulations or test results that allow the design to be validated. Recently, the Open Networking Foundation (ONF) has initiated a working group to find use cases for SDN in wireless networks [6].

A key feature of the SDN is the decoupling of the control and data planes.The controller is central to this type of network since it has maintaining and configuring the network, ensuring that the general aims of the network and the administration of it are being met. The intelligence of the SDN resides mainly in the controllers whereas the routers focus on the transmission of data. Currently, most wired SDN use the OpenFlow protocol at first proposed in [7]. At first thiswas implemented with a single controller scheme, until [8] it was determined that multiple controllers improved the performance until a limit of operation of the net was reached. The most commonly used controller in SDN is NOX and its Python-based POX successor [9], both run with OpenFlow. The scalability and location of the SDN drivers are a still open topic in wireless networks [10, 11].

This paper seeks to analyze the challenges of using SDN in MANET networks and proposes RDSNET, a driver architecture specially designed for MANETs' operationalconstraints and requirements, such as frequent route changes and mobility. The performance of RDSNET was evaluated using simulations in OMNET ++ in two scenarios (i) a centralized controller design, (ii) a distributed system of controllers.

II. Architecture Design

RDSNET proposes a control architecture for MANET that provides a solution to one of the main technical challenges of wireless SDN: The design and deployment of controllers. The goal of the architecture consisted of reaching a balance between a reasonable use of bandwidth, message latency and precision in the flow tables. The RDSNET architecture is shown in Fig. 1.

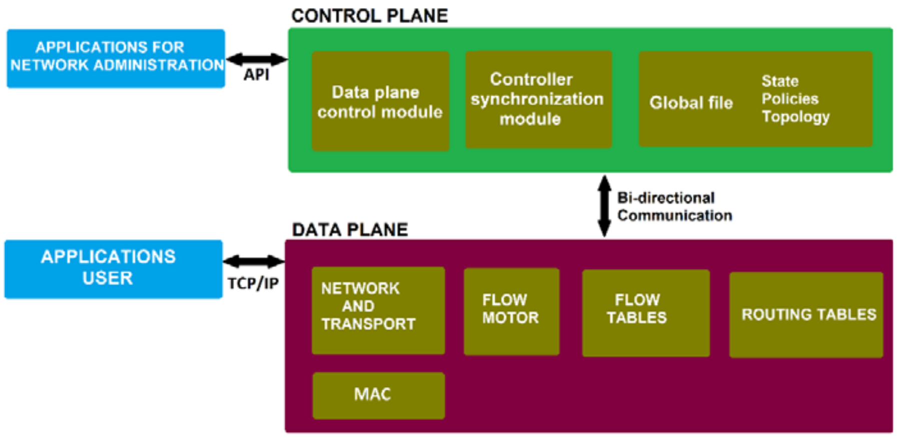

Fig. 1. Architecture Design RDSNET

The control plane is made up of the data plane control module, the synchronization module and the global status file. The data plane control module monitors the communications between the controller and the data in the nodes caring for the input flow and collecting information about the operation status. The synchronization module is responsible for the uniform operation between controllers when they work in a distributed fashion. The global status file has all the information obtained from the driver synchronization process.

The data plane consists of the flow engine, the flow tables, a typical network protocol, and the routing tables. The flow motor is the main module in the data plane and is responsible for modifying the flow tables following the instructions of the controllers, and it also periodically sends the status of the node to the controller and makes the corresponding requests when the applications need to send data to destinations that do not exist in the flow tables. The communication between the control and data planes is possible thanks to a communication protocol designed as part of the architecture. This is shown as PROTOCOL SD-DEN in Fig. 2, the latter also indicates the general operations between the controller and the communication nodes.

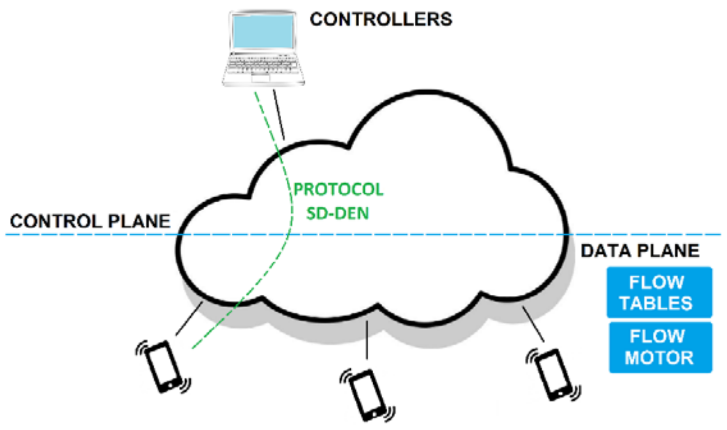

Fig. 2. Operations SD-MANET

A. Flows in RDSNET

In RDSNET, the nodes send messages from flow tables instead of the conventional routing tables. Protocols and routing tables continue to be used to keep up the routes between the nodes. The flows are superimposed on these routes to create preferred routes, allowing the controllers to create and alter flows from the needs of the administration and their own metrics.

Traditionally, distributed routing algorithms, such as AODV and OLSR, converge to a better routing path for a source-target pair, for which they use several types of metrics. These algorithms always choose the same path for the same source-destination pair, although other multiple possible routes are present in the network. This mechanism of rigid routing can cause bottlenecks against which, network administrators have certain limitations in issuing actions that solve these problems.

In RDSNET, flows overlap existing routes as in Fig. 3. Controllers can define any path between a source-destination pair as a flow, as much as flows are handled separately they can serve a different type of data. This opens up new ways to route network traffic, providing different paths for critical time messages, avoiding the routing of high priority packets through nodes in certain geographic areas, spreading traffic to achieve load balancing or even the deliberate creation of flows that maximize the gains derived from techniques such as network coding. Instead of the rigidity of the traditional route selection using RDS concept will allow great flexibility in the networks.

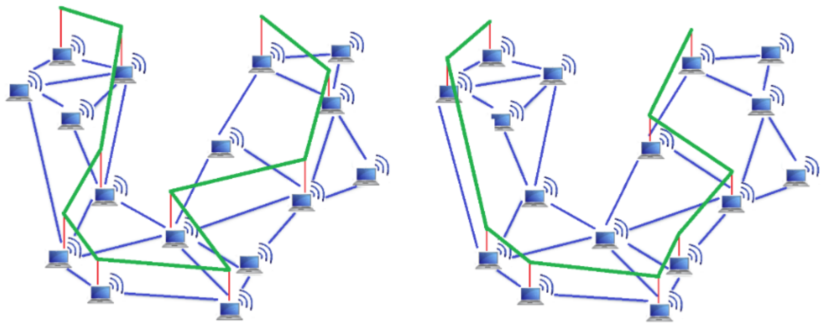

Fig. 3. Different flows for the same source-destination pair.

RDSNET uses some novel methods to keep overloads and latency as low as possible. One of these methods is to allow the controllers to insert flows within each node at the start of the network ensuring flows to all nodes in the same subnet. Nodes can start sending messages to other nodes within their subnets without incurring delays. This contrasts with traditional SDN designs, where nodes start with empty flow tables and only request new flows when they have to send information to destinations that are not in their tables; this results in the message being delayed significantly at the source until the controller handles the request. Initially, only the flows that are inserted are in the nodes that are part of the same subnet, since shipments are often made between nearby nodes and not the nodes furthest from the network. It is noted that this type of design concentrates the majority of overloads at the start-up of the network and so releases valuable resources during data transmission operations.

B. *Communication protocol for control and data planes*

The RDSNET protocol was designed to be lightweight and to include only the surcharges necessary for the system to work.

The main functionality of the protocol can be divided into the functions of the controller and the functions of the nodes. Controller protocols handle insertions and flow changes remotely, state requests to the nodes and synchronization between the controllers. Data plane protocols add functions such as pairing the flow table, requesting new flows and sending packets.

The main challenge with this part of the design is to ensure reliability and consistency when the controller distributes the flow inputs to several nodes concurrently. This is a continuing challenge in MANET since it is possible that a node is located so far away that it may never receive a flow insertion message despite repeated attempts.

C. *Controller selection*

A possible mechanism for selecting the controller for RDSNET is to dynamically generate where the nodes can choose the controller at execution time. The advantage of the dynamic selection of the controller is the ability to adopt another controller if the first one were to fail. However, after considering the benefits, the extra complexities and the typical operation of the MANETs, it was decided that such a dynamic mechanism is not necessary for RDSNET.

RDSNET instead uses a simpler static selection, where all controllers are predefined before the network is configured. There are three main reasons that make static selection a practical solution: First, in many MANETs, the network administrator is a person operating a computer. This person (or his delegate) is the only one authorized to make changes to the network, so he can easily predefine the controller. Second, not all nodes need the correct configuration to assume the role of the controller. Third, a dynamic selection algorithm requires control messages to be issued on the network, while the static choice does not generate any traffic.

D. Synchronization between controllers

When RDSNET operates in the distributed mode, the controllers have the additional task of synchronizing information amongst themselves, to ensure that all controllers can determine the overall state of the network. Since the RDSNET controllers are pre-configured instead of dynamically selected, synchronization is less complex, as each controller knows the identity and address ofother controllers.

The controllers will join a multicast group and periodically exchange their network states, such as flow tables, route tables, policies, and the status of the nodes. When correctly synchronized, a given controller will know the global state of the network, helping the network administrator, or the management algorithm, to decide on creating flows to nodes of other subnets.

An open topic here is how to ensure the consistency of the global state in all controllers. RDSNET partially handles this, by sending synchronization messages reliably and eliminating outdated records from drivers that do not respond after a timeout.

E. Handling controller errors

There are two main failure scenarios in handling RDSNET. First, the nodes have to detect and handle the faults of their direct controller. The controllers periodically send small packets of warnings to all the nodes they control. The nodes will treat a controller as failed if they do not receive messages from the controller for three consecutive periods of the timeout, after which time they will pass to the next controller in hierarchy order determined by the pre-configured list of controllers. Delayed receive messages from the original controller are detected by address return.

Second, the controllers have to handle the failures of other controllers. When one fails, the remaining controllers will have to be aware of this so that they can eliminate the corresponding entries of the device that has failed in its global state file. This is done to ensure the consistency of the global state seen by the controllers. RDSNET accomplishes this by using the periodic synchronization messages exchanged between the nodes. When controllers do not receive a synchronization message from a controller for three consecutive wait times, they eliminate failed controller entries from the global state.

III. SIMULATION RESULTS

The performance of the RDSNET architecture was analyzed through a series of network simulations using the OMNET ++ simulation environment. Particularly, the simulation was performed to compare the performance of a centralized controller design with a distributed one.

The functions of the RDSNET controller were modeled in the network created in OMNET ++ taking into account the routing and MAC protocols. In this RDSNET simulation, no management algorithm was implemented, since creating a management application for SDN went beyond this work. When a controller sends a flow insertion message to a node, the size of the message can be simulated accurately and the packet will be sent through the network. The flow inserted in the table simply reflects the information in the routing table,but there is no management application to perform the traffic engineering or the flow alteration in the simulation. This did not affect the interpretation of the results, since the performance of the controller design was compared and not the efficiency of the management algorithm.

A. *Configuration of the simulation and scenarios*

The simulations were performed with an ad hoc mobile network consisting of 32 nodes over 802.11g in the ad hoc mode with CSMA / CA at 1Mb / s, and the propagation loss established at $\gamma = 2$. The nodes were separated into 3 smaller subnets, each with 10 nodes and a gateway, to emulate the usual network topologies. The routing algorithm used was BATMAN [12], and the nodes used the IPv4 addressing scheme for simplicity.

A screenshot of the topology of the simulation can be seen in Fig. 4. The three computers on the stage host the controllers - all three will be active for the distributed case, whilst only the one on network 1 will be active for the centralized case. Each subnet is configured to use a separate frequency, so that messages need to pass through the gateways to pass between the subnets.

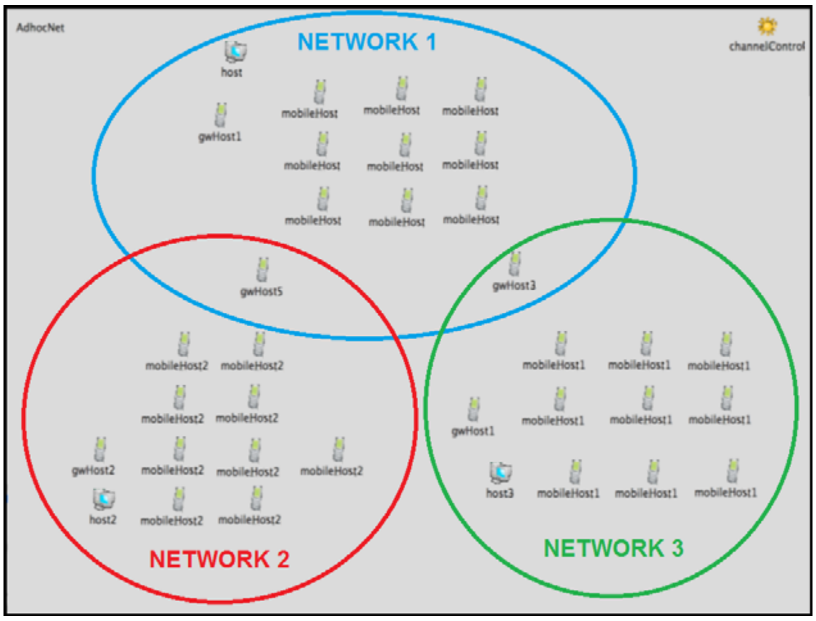

Fig. 4. Network topology used in the simulation

The six main scenarios for the simulation are shown in Table I. All simulations were performed with 20 minutes using 6 different sizes and frequency of delivery. A scenario with static nodes was at first simulated to provide a reference result set. In addition, the same scenario was simulated, but with the gateways moving, to test the performance of the designs when the subnets are disconnected from each other. The gateways are configured to move away at a speed of 40 km / h, in such a pattern that would result in the loss of connection to the network 1. Scenarios 3 and 6 were simulated using a conventional MANET without RDSNET but with all other configurations unchanged. One of the main concerns in using SDN in wireless networks is that performance may degrade in relation to the current generation due to additional costs in terms of communications between controllers and the nodes. These scenarios were implemented to assess the performance differences of RDSNET in relation to networks without the benefit of SDN. The evaluation metrics are in Table II.

TABLE I. List of scenarios for the simulation

Scenarios	Description
1	Static nodes with RDSNET central controller
2	Static nodes with three distributed RDSNET controllers, one for each subnet
3	Static nodes without RDSNET (to make comparisons)
4	Mobile nodes with a central RDSNET controller
5	Mobile nodes with three distributed RDSNET controllers, one per subnet
6	Mobile nodes without RDSNET

TABLE II. Metrics used for the evaluation of results

Metric	Target
Number of overloads related to traffic control in SDN	Evaluate which design has the highest number of overloads
Success rate and latency of control messages	Evaluate whether the choice of a controller design impacts on a different performance of the control messages
Number of application messages received	Evaluate the effect of each design on the completion rate of application messages
Latencies of application messages	Evaluate the effect on message latency

B. *Results for scenarios with static nodes*

The first set of results presented are those obtained for static scenarios 1 and 3. Fig. 5 shows the number of messages received per node for the centralized control design, distributed controller, in addition to the conventional case without SDN.

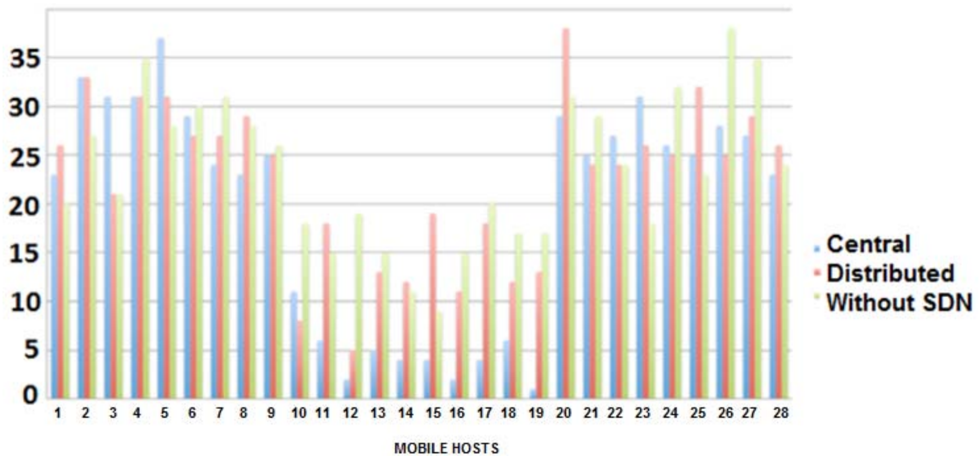

Fig. 5. Messages received by node in static scenarios.

It can be seen that the number of messages received per node in the distributed design is generally higher than with the centralized design. The total number of messages received in the distributed controller network was 15.9% higher than with the centralized controller network. It was found that the main reason for this was that in the centralized design all nodes must send requests for new flows to the only controller in the network. This resulted in greater congestion and collisions in network 1, causing more packet loss request flow. Fig. 6 clearly shows that the rate of completing flow requests is only 55% in the centralized design.
In contrast, there are several controllers in the distributed design, which allows the nodes of each subnet to communicate only with their own controller, avoiding overloading the network. The nodes in the distributed design has flow tables that are more up todate and can achieve a higher message success rate.

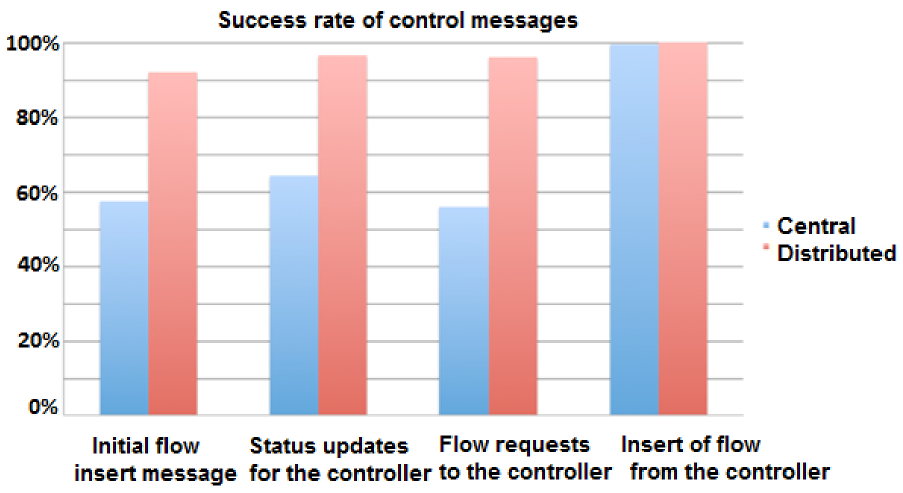

Fig. 6. Success rates of control messages in each design

This phenomenon can also be seen in Table III, which shows that the application data messages sent by the nodes in a centralized control network is smaller than in the distributed case.

Table III. Summary of results for static scenarios

	Centralized	Distributed	Difference
Number of messages sent	826	1175	42.3%

The performance of a conventional network without SDN is considered for comparison with the distributed case. The network without SDN does not have an independent control plane and the packets are routed exclusively through decisions governed by the routing tables of the nodes. In this manner fewer operating costs are generated. There is no need for communication with the controllers;however there are no benefits of flexibility offered by SDN.

The result showed that the distributed controller performance was similar to the case without SDN; despite the additional operating costs and the required communications between the node and the controller. The distributed network RDSNET received only 4.4% fewer messages; this shows that a wireless SDN application does not have as much performance degradation as was previously thought. Since management algorithms and reliable transport were not included in the simulations of this work. With an adequate management application and reliable transport the difference of 4.4% can easily be reduced.
From the results of the simulation, the average latency of the message was also studied. This latency in the centralized controller network was 80.6% higher than in the distributed controller network. Fig. 7 shows that a node can experience much higher latency using the centralized design. Outside exceptional cases, the average of the latency for the centralized case is still 24.59% higher.
The poor performance of message latency in the centralized design can be attributed mainly to a single controller, creating congestion and collisions, which occur more often in the network 1.

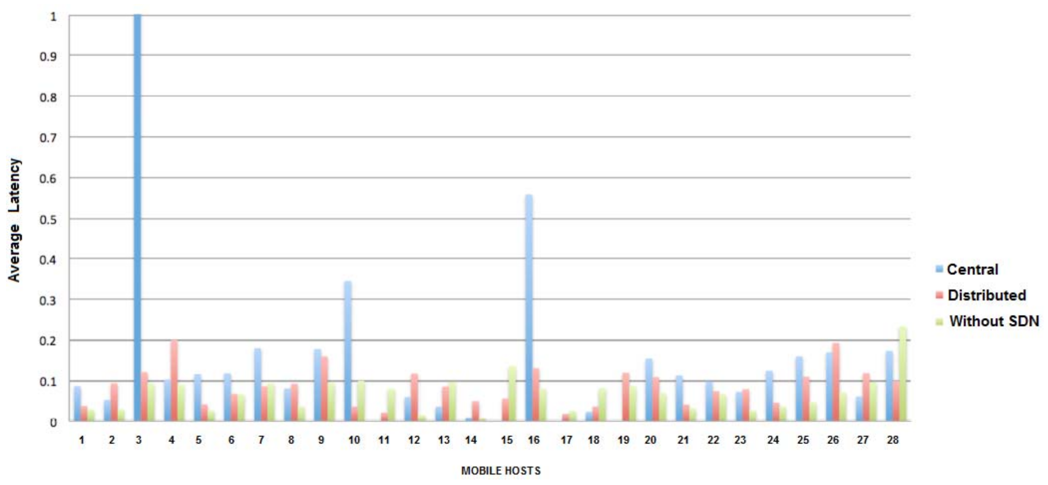

Fig. 7. Average message latency per node for static scenarios.

For the several controllers cases the nodes in distributed controller networks can achieve better latency performance. Since there is a controller in each network, the messages need to travel fewer jumps to reach a controller, hence shortening the time needed for exchanges. Bottlenecks are alleviated by controller distribution. This means that the queues of delays that appear in the different trajectories are much shorter compared to the central design. The success rates of the flow request messages are higher and the nodes do not need to send many requests before receiving a response. All these factors contribute to the nodes of the distributed controller network experiencing a lower message latency.
When compared to a network without SDN, the distributed design returned a lower performance. It was found that the latency of messages in the network without SDN is 20.36% better than the distributed RDSNET design. This can be seen as the main performance tradeoff: flexibility versus latency. It must be noted that his result does exclude the benefits that can be introduced through a management algorithm.

The operating expenses introduced by each controller designs, and the performance of the control messages, were evaluated. The total operating costs generated by the designs throughout the scenario are shown in Fig. 8. It can be seen that in the distributed controller it has about a 12.5% higher operation cost. This is due mainly to the additional controller synchronization messages that are required and to a greater number of flow insertion messages.

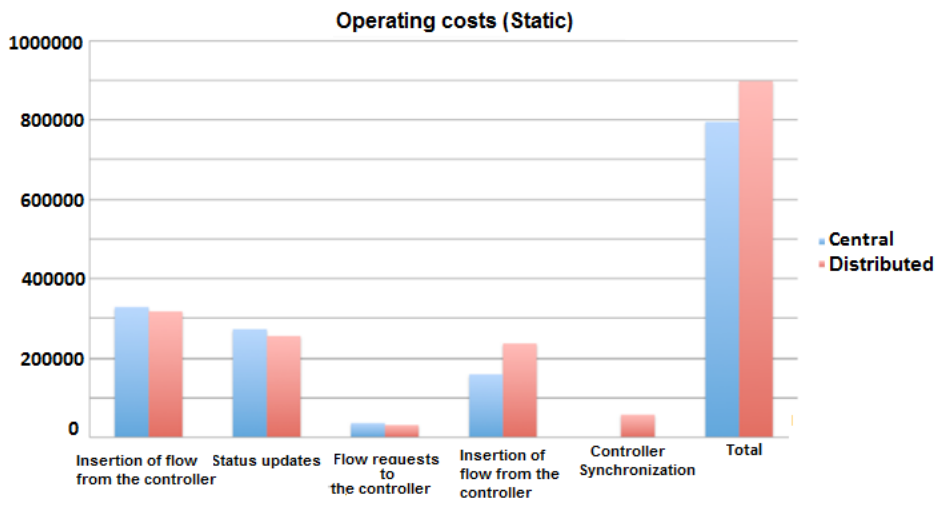

Fig. 8. Additional overloads due to RDSNET in each design.

Although the number of overloads that appear in the distributed design is greater, the number of control messages received successfully is also much higher, and the latencies are much lower. Fig. 6 can be used to explain the poor completion rates of application messages in the central design (Table IV). Flow requests and initial flow insertion are lost more often and nodes are unable to send many messages due to the lack of availability of flow entries in their tables. It is also in Fig. 9 that the latency of the flow requests and the flow inserts are longer for the centralized design. The time from which an application generates a message until in which the destination receives the message can be much longer for messages that require new flows since the request and response flow process is much longer. This is because the application of latency messages is greater than in the distributed design.

TABLE IV. Summary of results for the test with static scenarios

	Number of messages received	**Average Latency**
Centralized vs Distributed	Distributed 15.9% more	Distributed 80.6% faster
Distributed vs Without SDN	Without SDN 4.5% more	Without SDN 20.36% faster

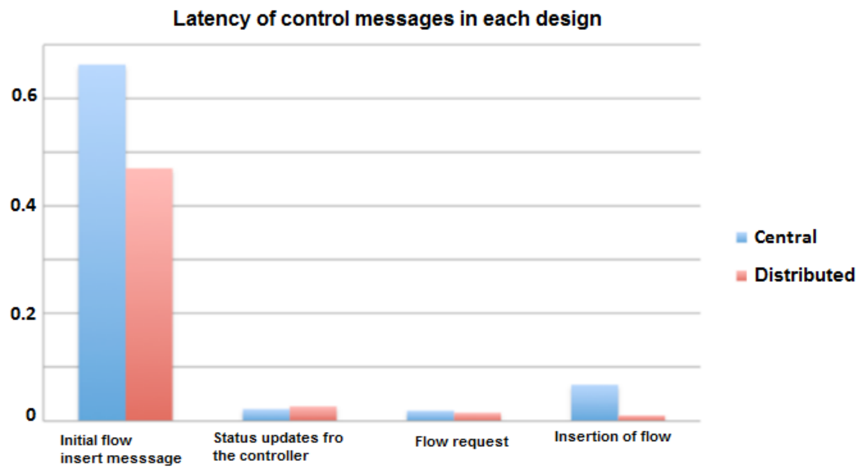

Fig. 9. The latency of control messages

It was observed that the highest latency in both designs occurred in the first messages of flow insertion. These messages were sent during the start up phase of the network by the controller(s) to the nodes under its control; for a central controllerthis means all the nodes in the network and for distributed controllers it only includes the nodes within its subnet.

C. *Results for scenarios with mobile nodes*

Tests similar to the previous were carried out using mobile nodes, but in scenarios with mobile nodes. The goal of these tests was to look at performance when mobility causes networks to disconnect. The simplest way to achieve this was to configure the gateways to move out of the networks. In the following tests, the topology used was the equal to that shown in Fig. 4. During the simulation of the scenarios, the nodes of the gateways begin to move away from network 2 and 3 at a speed of 40 km/h until outside the coverage range. The messages used were the same as those of the static case.

From Fig. 10, it is seen that the number of messages received by the nodes in the design of the distributed controller is greater than in the central design. The distributed case received 171.5% more messages than the centralized one. This enormous disparity can easily be explained by recalling that in the central case there was one controller in the entire network. When networks 2 and 3 are separated from network 1, their nodes cannot reach the controller. Consequently, not only were they unable to send messages between networks, they were not even able to send messages within the network once existing flows expired.

Nodes in the distributed design were still able to work partially, even after the gateways moved out of range. This is because the nodes could still reach their assigned controllers in the subnet, and this allowed them to successfully request new flow requests.

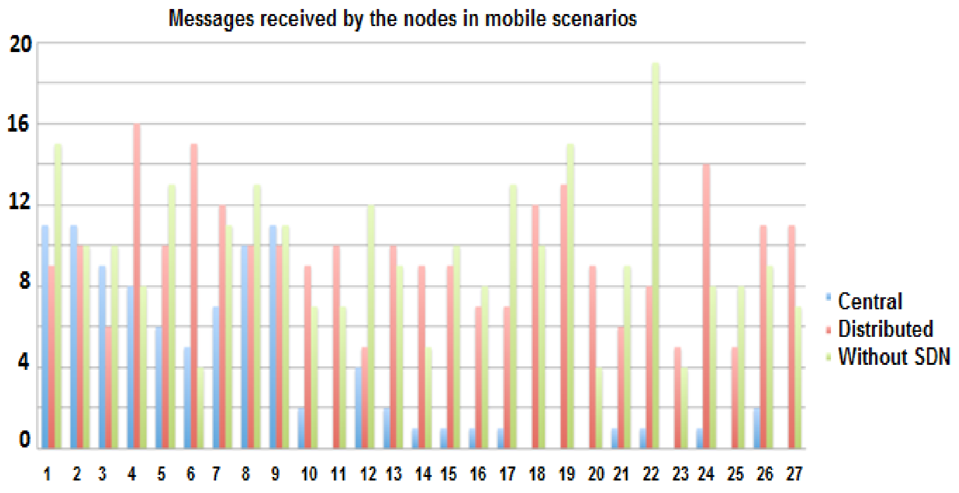

Fig. 10. Messages received by nodes for mobile scenarios

Compared to the network without SDN, the distributed RDSNET network had almost the same number of messages received successfully. The difference was only 0.38%. One explanation could be that after dismantling networks, all messages between networks would fail and only messages within the network had any chance of success. Here, the network without SDN would not have a significant advantage (in terms of successful messages) over an RDSNET, resulting in similar performance. However, the network latency advantage without SDN still exists and is explained below.

Message latency can only be calculated for messages that arrived successfully at their destination. Messages that were lost or never sent were not taken into account in calculating the average latency of messages on the network. The latency estimate for the centralized controller in this scenario was very low probably due to the small numbers that were received, as shown in Fig. 11. The distributed design received almost 3 times more successful messages.The average latency was higher possibly due to a greater number of messages that were taken into account for the calculation so providing a better expectation of the average. Due to the large difference in the number of messages received between centralized and distributed designs, it is unreasonable to compare average latency message with incompatible statistical estimators.

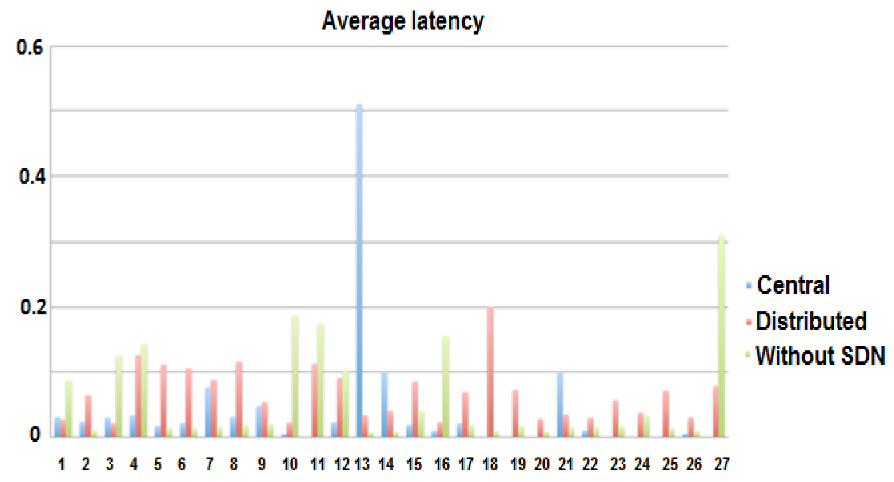

Fig. 11. Average latency by nodes in mobile scenarios

The latency performance of the distributed design RDSNET with respect to the network without SDN was compared. The network without SDN recorded a latency that was 12.5% lower than the RDSNET. This number is less than the 20% seen in the static case and shows that the latency overloads produced by the flow of node-controller requests caused fewer delays in communication patterns that were predominant within the network, ie, since most of the messages are destined for the nodes within the same subnet, the latency due to the flow of requests is lower, since most of the flows within the network must already be inserted from the start of the network.

The operating costs generated in the distributed design are higher than in the centralized design, as seen in Fig. 12. This was mainly due to many flow inserts and the controller synchronization messages sent. In the centralized case, the controller was able to receive fewer flow request messages and this resulted in a fewer insert messages sent.

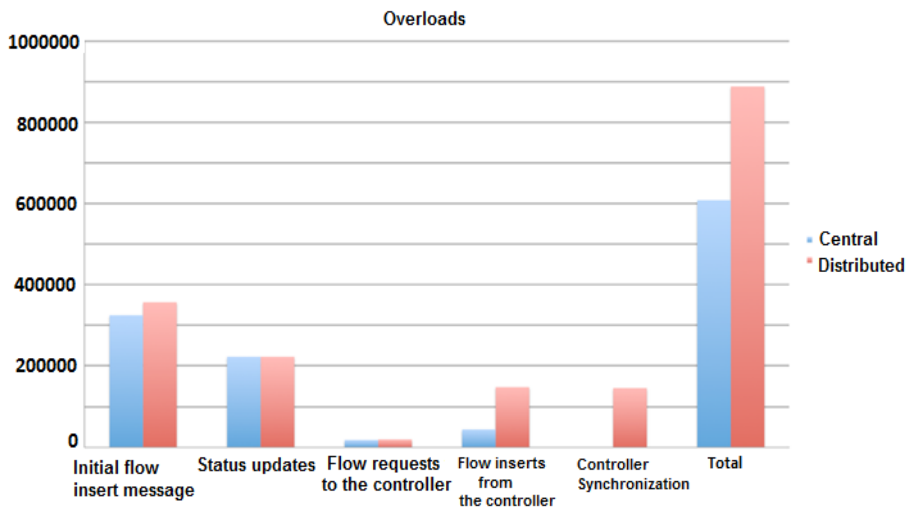

Fig. 12. Additional overloads due to RDSNET for each design

As in the static case, although the operating costs incurred were greater for the distributed design, the success rates of the control messages also exceeded the centralized design.

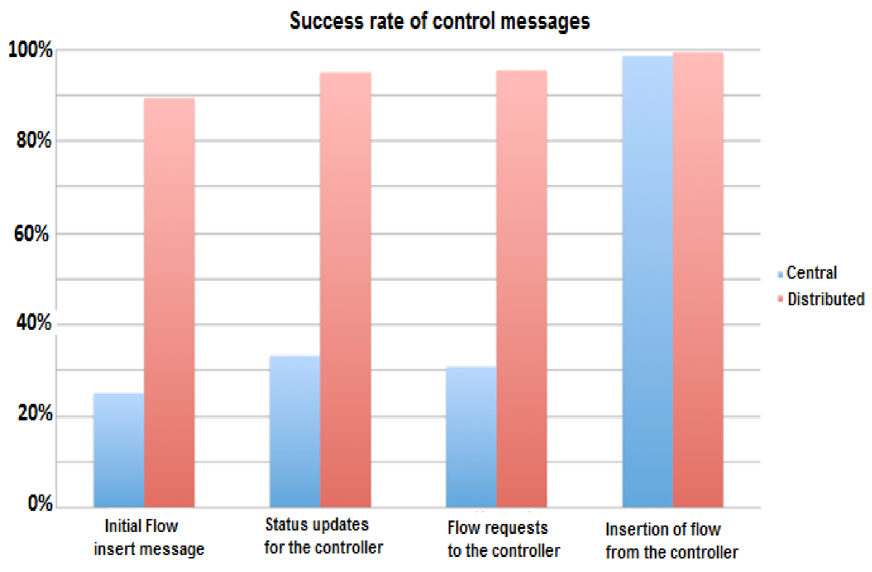

Fig. 13. Success rates of control messages for each mobile scenario

Fig. 13 clearly shows that the success rates of flow requests and status updates that were about 3 times better than in the centralized case. The poor performance of the central controller design was due to the disconnection of many nodes from the controller, which caused most of the control messages to be sent, but never received. An exception was the flow insertion messages, which had high completion rates, even in the centralized design. This was because the controller was inserting the flows in the flow tables of the nodes within its own subnet (network 1).

IV. Conclusion

In this work, the viability of using SDN techniques in wireless networks, especially in MANET was examined. A SDN controller architecture designed specifically for MANET called RDSNET was proposed. RDSNET solvesto one of the main technical challenges in wireless SDN, the design and implementation of controllers. The proposed architecture sought to balance between reasonable bandwidth usage, message latency and the accuracy of the flow tables.

The RDSNET simulations were performed with OMNET ++ to test its operation and to determine whether a centralized or distributed controller design was best. The simulations showed that the distributed design clearly exceeded the centralized design. When the performance of a distributed design was compared against a network without SDN, the message success rate was close the distributed design receiving at most 4.5% fewer messages. However, its performance was lower in respect of latency. The network without SDN had a message latency 20% lower. This performance disadvantage must be weighed against the flexibility that SDN brings to the MANETs.

Attractive use cases were found for the use SDN in MANETs, such as the introduction of dynamic traffic engineering and network management. There remainmany challenges that must be overcome before achieving a robust system. This work has shown that the performance differences are close between MANETs enabled with SDN and conventional networks without SDN.

Acknowledgment

This work is the result of a research funded by Universidad ECCI through the internal call Nº 1 of2017-2018.

References

[1] J. Moura, D. Hutchison, Review and analysis of networking challenges in cloud computing, Journal of Network and Computer Applications 60 (2016) 113 – 129.
[2] A. Hakiri et al., Software-Defined Networking: Challenges and research opportunities for Future Internet, Comput. Netw. (2014).
[3] I. Ku, Y. Lu and M. Gerla, "Software-Defined Mobile Cloud: Architecture, services and use cases," 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), Nicosia, 2014, pp. 1-6.
[4] K. Pentikousis, Y. Wang and W. Hu, "Mobileflow: Toward software-defined mobile networks," in IEEE Communications Magazine, vol. 51, no. 7, pp. 44- 53, July 2013.
[5] T. Luo, H. P. Tan and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," in IEEE Communications Letters, vol. 16, no. 11, pp. 1896-1899, November 2012.
[6] Open Networking Foundation ONF (Mar 2016). URL https://www.opennetworking.org/about/onf-overview
[7] N. McKeown et al. OpenFlow: Enabling Innovation in Campus Networks. In Sigcomm Computer Communications Review, Vol 38, No 2, April 2008.
[8] G. Yao, J. Bi, Y. Li and L. Guo, "On the Capacitated Controller Placement Problem in Software Defined Networks," in IEEE Communications Letters, vol. 18, no. 8, pp. 1339-1342, Aug. 2014.
[9] Nox (Jul 2016). URL http://www.noxrepo.org/

[10] W. Xia, Y. Wen, C. H. Foh, D. Niyato and H. Xie, "A Survey on Software- defined Networking," in IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, Firstquarter 2015. doi: 10.1109/COMST.2014.2330903

[11] C. Salgado, S. Mora and D. Giral, “Collaborative Algorithm for the Spectrum Allocation in Distributed Cognitive Networks,” in International Journal of Engineering and Technology, 2016, vol. 18, pp. 2288-2299. doi: 10.21817/ijet/2016/v8i5/160805091

[12] A. Klein, L. Braun and F. Oehlmann, "Performance study of the Better Approach to Mobile Adhoc Networking (B.A.T.M.A.N.) protocol in the context of asymmetric links," 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), San Francisco, CA, 2012, pp. 1-7. doi: 10.1109/WoWMoM.2012.6263783

AUTHOR PROFILE

Sergio Mora received his Bachelor's degree in Electronics Engineering from Universidad Nacional, Bogotá, Colombia in 2009 and his M.Sc. degree in Electronics and Computers Science from Universidad de los Andes, Colombia in 2013. Since 2014 he joined the Electronics Department of Universidad ECCI as a researcher and associate professor. He was a member of Grupo de Investigación y Desarrollo Tecnológico Aplicado INDETECA. His research interests in engineering include energy resource management in data centers, smart grid, and wireless systems.

Jhon Vera received his Bachelor's degree in Electronics Engineering from Universidad Santiago de Cali, Colombia in 2002 and his M.Sc. degree in Electronics and Computers Science from Pontificia Universidad Javeriana, Colombia in 2013. Since 2014 he joined the Electronics Department of Universidad ECCI as a researcher and associate professor. He was a member of Grupo de Investigación y Desarrollo Tecnológico Aplicado INDETECA. Jhon Vera has publications in internationally recognized conferences, such as the 4th Circuits and Systems (CWCAS), XII International Conference on Electrical Engineering, Computing Science and Automatic Control (2015); and IEEE ARGENCON 2016. His research interests in engineering include energy resource management in, smart grids, and power electronics.