International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-5, Issue-5)

Research Article | **May 2016**

OpenFlow Controllers Performance Evaluation

[1]FAID Souad, [2]Mohamed MOUGHIT, [3]Noureddine IDBOUFKER
[1] Laboratory of Computer Networks, Mobility and Modeling IR2M Faculty of Science and Technology, University Hassan First Settat, Morocco
[2] Laboratory of Computer Networks, Mobility and Modeling IR2M National School of Applied Sciences, University Hassan First Settat, Morocco
[3] Ecole Nationale des Sciences Appliquées, Université Cadi Ayyad Marrakech, Morocco

Abstract—

Software-defined networking (SDN) is an approach to make networks programmable and used through virtualization. In traditional networks, the overall control of the network is ensured by logic centralized control function. With SDN, network flows are controlled to an overall level of abstraction, usually, but not always, using the OpenFlow protocol. In this paper, I presented an overview about the SDN technology as well as its purpose and how it can deal with all kind of problems encountered in current networks. As part of SDN architecture, a presentation of the OpenFlow Protocol is inevitable, thus, I explained its big role in the interaction between SDN's Layers and why OpenFlow is the most suited for this role, OpenFlow remains the most compatible solution with SDN architecture. To have a clear idea about how OpenFlow works in an SDN Network, i treated the OpenFlowcontrollers(NOXMT, POX, BEACON) and made a comparison between them which led me to the conclusion that all the current OpenFlow controllers have almost the same gap, is that they cannotperfom throughput & latency at the sametime, my contribution in science researches will start from this point and try to fill this gap in order to improve networks performance.

Keywords— SDN, Open Flow, NOXMT, POX, BEACON.

I. INTRODUCTION

The definition of SDN that is currently emerging focuses a little less on decoupling and more onthe ability to provide programming interfaces in the network equipment, whether there is a separation between control and transmission or not. One reason for this change is the fact that manufacturers have recently announced that in the SDN technology, they will provide APIs in multiple platforms. Seen this way, the SDN enables companies to replace a manual interface in equipment with a programming interface that can help automate tasks[1].

- Data Plan:Any activity involving data packets results sent by the end user.
- Management plan: All activities related to the monitoring of networks.
- Control plan: Activities needed to perform data plan activity.

II. SDN TECHNOLOGY PRINCIPLE

We define three levels of abstractions for SDN.

- Transmission: it must enable the deployment of any behavior or service desired by a network.
- Distribution: aims to protect SDN applications of the whims of the distributed state, by centralizing the control at an SDN controller, running an operating system known as Network Operating System (NOS).
- Specification: The latter abstraction must allow an application to express the desired behavior of the network[2].

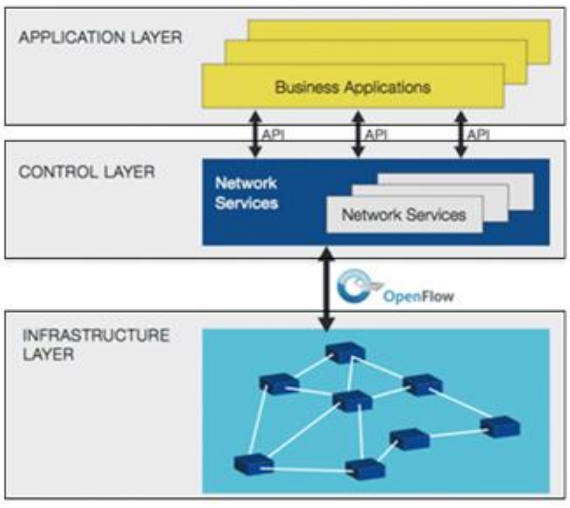

Fig. 1. SDN Layers.

- Application Layer: Network applications express behaviors to deploy and implement, these behaviors are then translated into commands to be installed in the equipmentsof the data plan.
- APIs:interfaces that allow communication between layers of the SDN model.
- The SDN Control layer**:** provides a centralized view of the global network and sends commands to all network equipments.
- The Infrastructure Layer**:** is composed of a set of network equipments (Switches, routers and intermediate equipments). The main difference is that the intelligence is removed from these devices and centralized at the controller[3].

A. The difference between SDN and traditional networks.

The difference between SDN and traditional networks is noted below:

- Control and transmission logic decoupling: Decoupling between the control of network and the actual transmission is not a new concept.
  The definition of SDN that is currently emerging focuses a little less on decoupling and more on the ability to provide programming interfaces in the network equipment, whether there is a separation between control and transmission or not.
- A programming interface: SDN delivers programming interfaces in the network equipment, the SDN enables companies to replace a manual interface in the network equipment with a programming interface that can enable the automation of tasks[4].

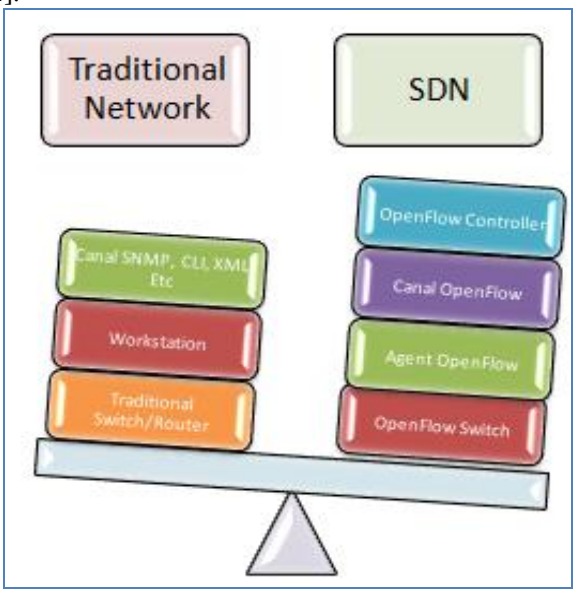

Fig.2. Difference between SDN and traditional.

B. Software defined technology domains

- In a large corporate network, deploying a single controller to manage all network devices prove undesirable. A more likely scenario is that the operator of a large corporate network or carrier dividesthe entire network in many domains of SDN donot overlap. Within each SDN domain, its controller may define policies specific to the informationfieldimporting from devices, aggregation, and exporting to external entities. These policies must be secured by the factthat other domain controllers do not know the existence of suchpolicies for a given SDN domain[5].
- SDNi is a protocol for interfacing between SDN domains. One way to implement SDNi is to use the BGP and SIP extensionsonSCTP protocols to ensure better coordination between the SDN domains.

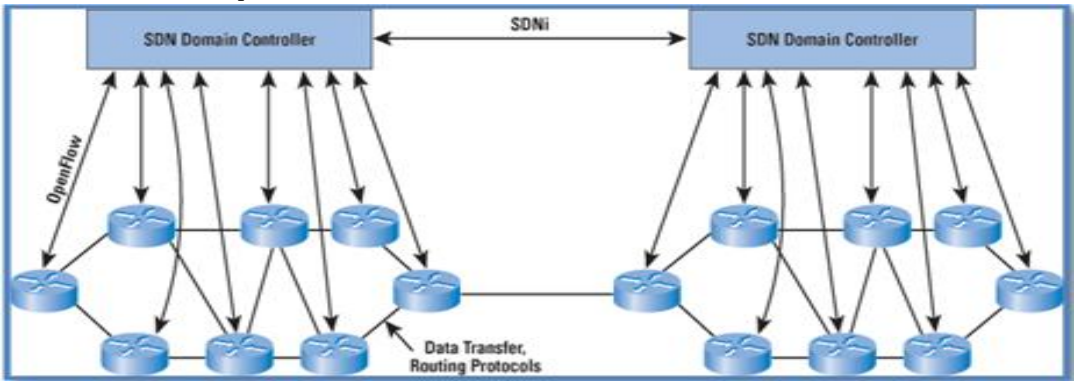

Fig. 3.The architecture of a SDN Domain.

C. Software defined network control platforms

The choice between a distributed or centralized control platforms depends on the network size where to implement SDN. The figure below illustrates the two types of architectures.

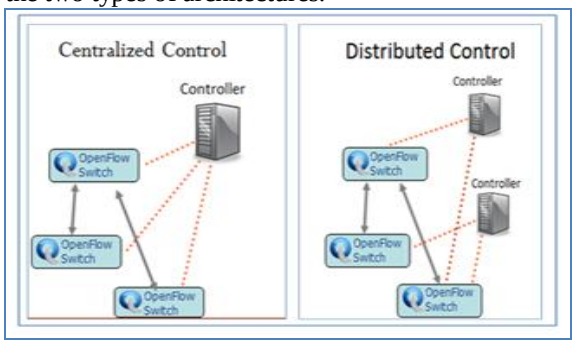

Fig. 4. Centralized/Distributed architectures.

III. THE ARCHITECTURE OF OPENFLOW PROTOCOL

The architecture starts from the reference of a typical Ethernet switch or router. It consists of:

- A data plane that receives packets on the input ports.
- A control plane performs several high-level functions such as running dynamic protocols (BGP, OSPF, STP, etc.), supporting a management interface (SNMP, CLI, SOAP, etc.) [6].

A. OpenFlow functioning

In order to perform the functioning of OpenFlow, the following elements are necessary:

- Controller :is an OpenFlow server running on the control plane.
- Forwarder: is an OpenFlow device running on the forwarding plane.
- OpenFlow flow table :is a forwarding table independent of service type. A flow table contains match fields and associated actions. A forwarder matches packets against a specific field in a flow table and performs a specific action associated with the field on matching packets.Each packet that enters a switch passes through one or more flow tables[7].

Each flow table contains entries consisting of six components:

- Field matching: Used to select the packets that match the values in the fields.
- Priority: relative priority of table entries.
- Counters:used for matching . it equals Packet + Error Bits.
- Instructions:Measures in case of correspondence.
- Timeouts: the maximum inactivity time of a flow.
- Cookie :Can be used by the controller to filter flow statistics, changes in flows and the removal of flows; not used during packet processing.
- Header Fields: Allows packets matching.
- Actions: Actions to apply to the corresponding packets.

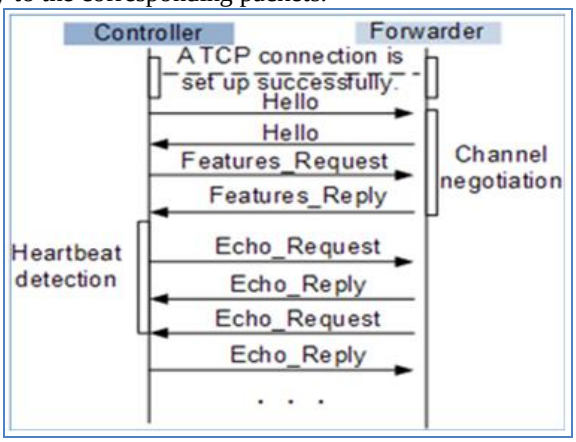

Fig. 5. Flowchart for establishing an OpenFlow channel.

- The controller and forwarder establish a TCP connection.
- The controller and forwarder exchange Hello packets to negotiate parameters.
- After negotiation is successful, the controller sends a Features_Request packet to query attributes on the forwarder.
- Upon receipt of the Features_Request packet, the forwarder sends the controller a Features_Reply packet carrying the forwarder's attributes, such as the supported flow table format and buffer size.

- After receiving the Features_Reply packet, the controller succeeds in establishing an OpenFlow channel with the forwarder.
- The two entities exchange Echo packets (heartbeat packets) to monitor the status of each other. One end periodically sends an Echo_Request packet to the other end. Upon receipt of this packet, the other end responds with an Echo_Reply packet. If one end fails to receive an Echo_Reply packet after sending a specified number of Echo_Request packets, it considers the other end faulty and terminates the connection to the other end[8].

B. Reporting OpenFlow Port information

The forwarder can report its port information to controller along an OpenFlow channel.In Figure 6,After an OpenFlow channel is established between a controller and a forwarder, the controller queries port information on the forwarder.The controller sends a Multipart_Request message to query the forwarder's port information.

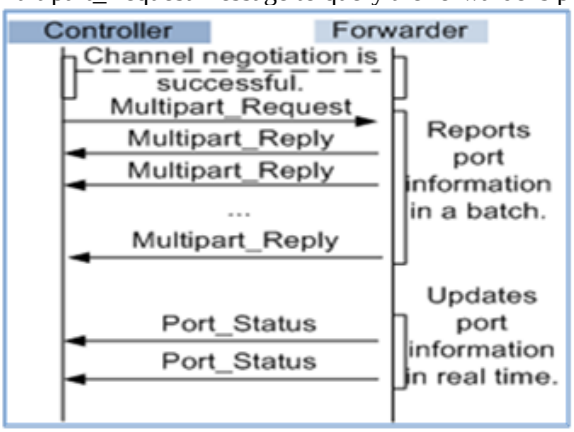

Fig. 6. Flowchart for reporting port information from a forwarder to a controller.

C. Delivering an OpenFlow Flow Table

OpenFlow defines actions of a forwarding device in a flow table independent of service type. Entries in the flow table are delivered by a controller to a forwarder along an OpenFlow channel. In figure 7, a controller sends a Flow_Mod message carrying flow table information to a forwarder. The Flow_Mod packet primarily carries basic flow table information (such as the table ID and priority),a match attribute set, and an instruction set. The match attribute set contains information (such as MAC and IP addresses) used to match packets against entries in the flow table. Matching packets are processed using a specific instruction set. The instruction set contains various instructions, such as modifying packet attributes and forwarding packets through a specific outbound interface[9].

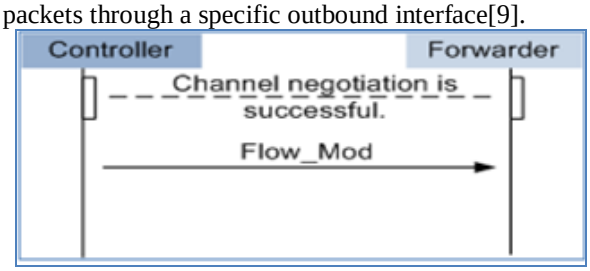

Fig. 7. Flowchartfor delivering a flow table.

D. OpenFlowTable Delivery Process

The controller sends private flow table information in a private extended Experimenter packet to the forwarder. In addition, the controller supports the private smoothing process, involving the start, data smoothing, and end phases. The private flow table contains VXVLAN-related FEC entry information[10].

E. OpenFlow Packet Transparent Transmission

Usage Scenario: The Controller and a forwarder transparently exchange packets to meet service requirements. For example, VXLAN-related protocol packets, such as ARP and ICMP packets, are exchanged between the controller and forwarder along an OpenFlow channel in some service scenarios:

-The forwarder receives a flow table delivered by the controller and processes packets based on behaviors defined in the flow table. The forwarder sends packets to the controller in the following scenarios:

If the forwarder has no flow table to guide packet forwarding, it sends the packets to the controller based on a configured or default policy. This usage scenario is not supported.

If the forwarder has a flow table delivered by the controller, it sends packets that match the flow table to the controller.

-The controller sends packets to the forwarder in the following scenarios:

Upon receipt of packets sent by the forwarder, the controller forwards the packets based on a service policy.

The controller can also generate packets and send them to a forwarder.

F. Messages Formats

OpenFlow protocol has three types of messages to communicate between the controller and the OpenFlow switch over a secure channel or over a TCP channel as shown in Fig.5. They are classified according to the initiator of the message into controller to switch messages, asynchronous (switch to controller) messages, and symmetric messages. The controller to switch messages are used to assert its control upon the switch, reading the switch status, and modifying the switch states which includes editing the switch flow tables.OpenFlow protocol defines Packet-in and Packet-out messages: Packet-in: sent by a forwarder to a controller & Packet-out: sent by a controller to a forwarder[11].

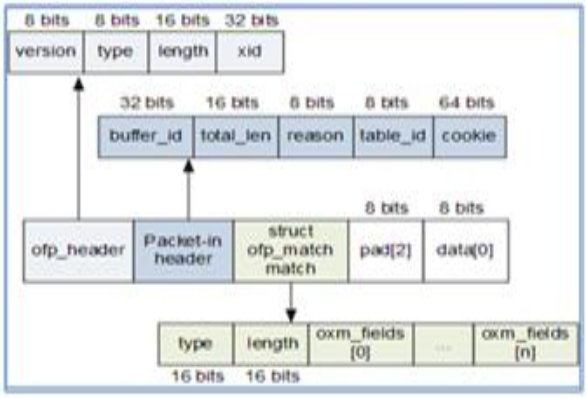

Fig. 8. Packet-in message format.

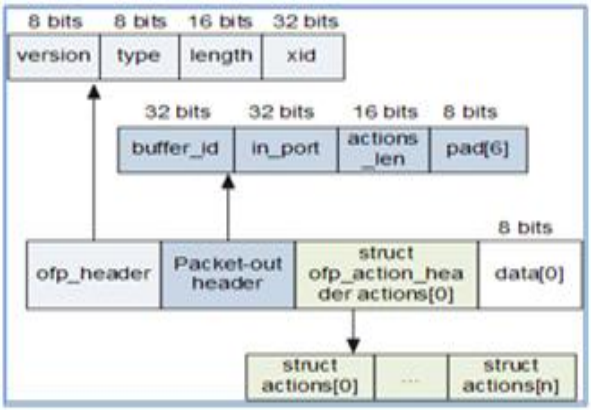

Fig. 9. Packet-out message format.

Packet-in:Thecontroller delivers a flow table to a forwarder. Upon receipt of the flow table, the forwarder matches a packet against the flow table. Then the forwarder encapsulates the packet in a Packet-in message, and sends the message to the controller through an OpenFlow channel. Upon receipt of the Packet-in message, the controller processes it to meet service requirements.

Packet-out:The controller encapsulates a packet into a Packet-out message and sends the message to a forwarder through an OpenFlow channel. The forwarder then obtains information carried in the Packet-out message and forwards the packet[12].

IV. OPENFLOW CONTROLLERS COMPARISON

Mininet is a network emulation platform that supports rapid development in SDN using OpenFlow protocol. It is the most popular platform used by researchers due to its simplicity, availability and flexibility. The Mininet open-source network simulator is designed to support research and education in the field of SDN systems. It creates a simulated network that runs real software on the components of the network so it can be used to interactively test networking software. In the following test drive, we will use Mininet to simulate and test some SDN scenarios and evaluate Mininet as network simulator[13].

Evaluation Scenario

To evaluate the OpenFlow controller's performance, we have to built the following scenarios:

- A virtualized network using POX,NOXMT,BEACON,
- FLOUDLIGHT.
- 16 switchs.
- 30 hosts.

In each scenario, a host generates traffic to cross the entire network topology simulating a production network.

To evaluate the controller performance, Cbench should be used .CBench is a performance measurement tool designed for benchmarking OpenFlow controllers. The benchmarking measurement is the amount of flows persecond that can be processed by the controller.

Cbench emulates a configurable number of OpenFlow switches that all communicate with a single OpenFlow controller.Each emulated switch sends a configurable number of new flow messages to the OpenFlow controller, waits for the appropriate flow setup responses, and records the difference in time between request and response.

Cbench supports two modes of operation which are Latency&Throughput[14].

Evaluation Procedure

To define the experiment, initially it is necessary to specify the hosts and network that will be used. The OpenFlow controller has the responsibility to define the best path toconnect all hosts.

To evaluate the controller performance into the test topology the Cbench program was included, part of theOpenFlow suite that creates and sends a large amount of OpenFlow messages to the controller in order to test its performance[15].

Results

The performance test results are shown in the following graphs. They show the performance of each controller in how many OpenFlow messages could be processed.Figures12 &13 show the performance of each OpenFlow controller in term of throughput and latency. We can see that every controller, independently of architecture and programming language, has better performance in the throughput or latency approach.

Beacon Controller is the best when it comes to throughput, from the graph above we can see that it can carry the highest rate which is 7929.39, on the other hand it has high latency compared to other controllers, where POX has the lowest value.

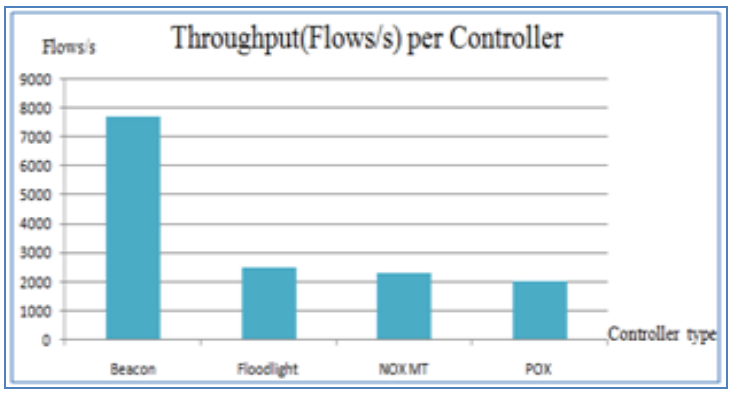

Fig. 10. Throughput per controller.

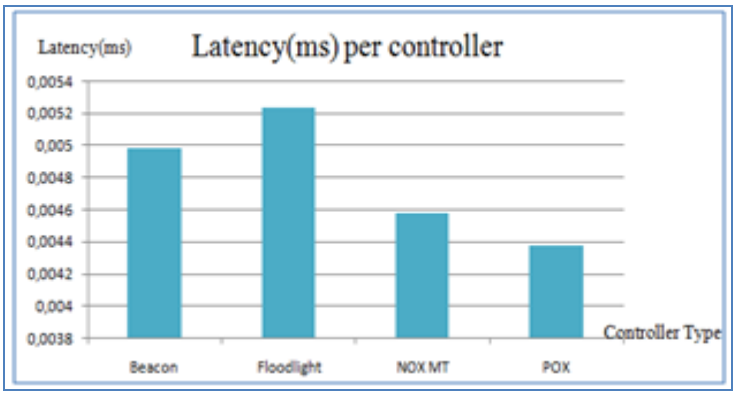

Fig. 11. Latency per controller.

As seen in the graph below, the throughput cannot exceed 8013.98 flows/ms, no matter how many switches we add to the networks.

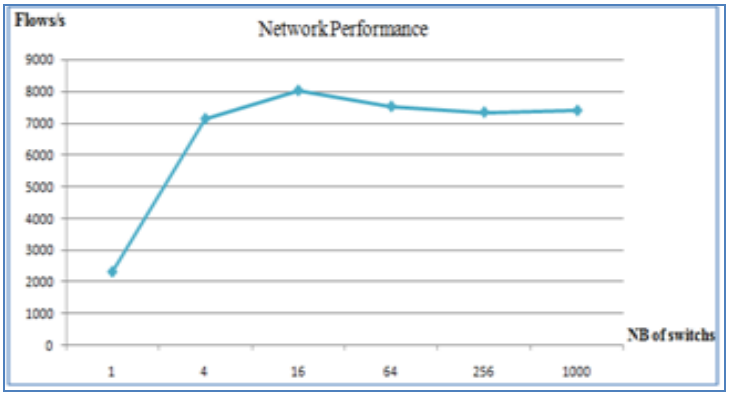

Fig. 12. Network performance per size.

V. CONCLUSION

It was a difficult and complicated project, since it's a new technology,on the other hand it was very interesting because it's the technology for future networks and rich of information. Initially it has been necessary to state the context of networks virtualization. To achieve this objective, a number of problems must be solved. In this article, we presented a discussion on the coexistence of SDN & Open flow to improve network performance in the mininet environment.

Given its low latency, POX is partially the best among OpenFlow controllers, the reason why my future researches will be focused on POX in order to solve its throughput limitation.

REFERENCES

[1] S. Sesia, I. Toufik, and M. Baker, LTE, The UMTS Long Term Evolution:From Theory to Practice, ser. Wiley InterScience. Wiley, 2009.
[2] L. Verma, M. Fakharzadeh, and S. Choi, "WiFi on steroids: 802.11AC and 802.11AD," Wireless Communications, IEEE, vol. 20, no. 6, pp. 30–35, December 2013.
[3] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: software definedradio access network," in Proc. of ACM HotSDN, 2013.
[4] X. Jin, L. Li, L. Vanbever, and J. Rexford, "SoftCell: scalable and flexible cellular core network architecture," in Proc. of ACM CoNEXT,2013.
[5] ONF, "SDN Architecture Overview vl.O, Dec 20 13.Updated White paper on "Network Functions Virtualisation".
[6] A. Basta et ai., "A Virtual SDN-Enabled LTE EPC Architecture: A Case Study for S-/P-Gateways Functions," Proc. 20 13 IEEE SDN for Future Networks and Services, Trento, Italy, Nov. 20 13.
[7] G.O.Young,"Syntheticstructureofindustrialplastics,"in*Plastics*, 2nded., vol. 3, J. Peters, Ed. New York: McGraw-Hill,1964,pp.15–64.
[8] W.-K.Chen,*LinearNetworksandSystems*.Belmont, CA:Wadsworth, 1993, pp. 123–135.
[9] J. U. Duncombe, "Infrared navigation—Part I: An assessment of feasibility," *IEEE Trans. Electron Devices*, vol. ED-11, no. 1, pp. 34–39, Jan. 1959.
[10] E. P. Wigner, "Theory of traveling-wave optical laser,"*Phys. Rev.*, vol. 134, pp. A635–A646, Dec. 1965.
[11] E. H. Miller, "A note on reflector arrays," *IEEE Trans.AntennasPropagat.*, to be published.

