Implementación de un Prototipo de una Red Definida por Software (SDN) Empleando una Solución Basada en Hardware

Juan Carlos Chico; David Mejía; Iván Bernal

Ingeniería Eléctrica y Electrónica, Escuela Politécnica Nacional, Quito - Ecuador

***Resumen*--La arquitectura de las redes convencionales ha sido útil hasta la actualidad, pero en los últimos años han surgido nuevos requerimientos que este tipo de arquitectura no puede satisfacer eficazmente, por lo que se han desarrollado alternativas como las SDN (Software Defined Networks). En las SDN se persigue la separación entre el plano de control y el de datos, concentrando la inteligencia de la red en servidores en los que se ejecutan los denominados controladores, para así definir de forma flexible el comportamiento de la red. Este artículo presenta la simulación de una SDN en Mininet, usando una topología similar al prototipo implementado posteriormente de manera física y que se emplea como una infraestructura de prueba (*testbed*). En el prototipo físico, para el plano de datos, se emplean conmutadores a los cuales se les modifica su firmware para soportar el protocolo OpenFlow. En el plano de control, los controladores definen y transmiten reglas de flujos a los conmutadores por medio de OpenFlow, las que determinan el tratamiento que dan los conmutadores a cada flujo de tráfico que circula por ellos. En el proyecto se usan cuatro controladores: NOX, POX, Beacon y Floodlight; para cada uno se desarrolla un componente de software, en el lenguaje de programación determinado por el controlador, como Python y Java, para definir la funcionalidad de la red, lo que permite visualizar a la red, en su conjunto, como una plataforma programable. Finalmente, se presentan los resultados obtenidos al ejecutar los componentes desarrollados en los controladores y aplicar las reglas instaladas en los conmutadores sobre los flujos de datos.**

Términos para indexación—SDN, Mininet, Openflow, Plano de control, Plano de datos, controladores.

I. INTRODUCCIÓN

En los últimos años, Internet ha sufrido varios cambios e innovaciones para satisfacer la creciente demanda de aplicaciones de sus usuarios. Por ejemplo, se han incrementado los servicios de comercio electrónico, las redes sociales, los servicios móviles, los servicios de virtualización y *cloud computing*, lo que ha generado la necesidad de reestructurar las arquitecturas de red tradicionales [5].

Las arquitecturas de red tradicionales presentan limitaciones frente a estos nuevos requerimientos; así, la limitada capacidad de adaptación a nuevas tecnologías, la poca escalabilidad de las redes y el ineficiente uso de políticas de control de acceso muchas veces crean huecos de seguridad susceptibles a ataques. Esto ha impulsado una búsqueda de varias alternativas para reemplazar las redes tradicionales. Una de esas alternativas, proveniente de la Open Networking Foundation (ONF), plantea el uso de Redes Definidas por Software (SDN) para satisfacer los requerimientos actuales de los usuarios de Internet [5], [8].

Las SDN constituyen una arquitectura de red cuyo objetivo fundamental es desacoplar el plano de control del plano de datos, lo que facilita un mayor control y nivel de gestión sobre los equipos de conectividad, garantizando al administrador de la red un control centralizado. El término control centralizado de la red tiene una connotación netamente lógica y, por tanto, la administración de la red se puede centrar en uno o varios servidores controladores, es decir, un control distribuido de manera física; incluso pueden existir servidores de respaldo en caso de una falla de un determinado servidor [5], [7], [8].

Como consecuencia de un control centralizado y directo, se pueden optimizar las políticas de seguridad de una determinada entidad o empresa, reaccionar

rápidamente a cambios en los requerimientos de la red o a una adaptación de nuevas tecnologías y conseguir redes altamente escalables.

La arquitectura SDN consta de tres grupos de dispositivos:

- Uno o varios servidores controladores, en los cuales se concentra y centraliza toda la inteligencia de la red. Estos dispositivos se encargan de definir las reglas para la comunicación en la red, basados en la información suministrada por el administrador de la red. Sobre este servidor se ejecuta el software para controlar la red. En la actualidad, existen varias alternativas de software controlador cuya diferencia fundamental radica en el lenguaje de programación que usa dicho software para la definición de las reglas de flujo. Así, se tienen alternativas como: NOX, que es el controlador original de OpenFlow [5] y que tiene dos versiones que usan los lenguajes de programación Python y C++; POX, escrito en Python; Beacon, basado en Java; y Floodlight, escrito en Java, entre otros [8].
- Dispositivos de conectividad que en este caso toman la denominación de conmutadores (*switch*), pero que, sin embargo, pueden realizar varias funciones: conmutación, enrutamiento, control de acceso, entre otras [8].
- Hosts que utilizarán la red para comunicarse entre ellos. Por host se entiende una computadora personal, un servidor u otro dispositivo que utilice la red SDN.

En el plano de control, la comunicación entre el o los servidores controladores y los conmutadores es posible mediante el protocolo OpenFlow, el cual permite establecer la comunicación inicial entre estos dispositivos, para posteriormente enviar la información relacionada a las reglas de flujo que se transmitirán desde la entidad controladora hacia los conmutadores.

En la Fig. 1 se puede observar las partes constitutivas de la arquitectura SDN y el rol que cumple el protocolo OpenFlow en esta arquitectura [5].

El encaminamiento de los flujos se basa en una comparación realizada en el conmutador entre los datos de un flujo entrante y las estructuras de comparación definidas por el servidor controlador y enviadas al conmutador para ser añadidas como reglas de flujo. Una estructura de comparación consta de varios parámetros tales como: puertos físicos de entrada, direcciones IP o puertos TCP/UDP de origen o destino, tipos de paquetes (TCP, IP, ARP), entre otros.

Una regla de flujo consta de: contadores que definen cuanto tiempo debe permanecer la regla en la tabla de flujos de un conmutador y cuál es el tiempo de inactividad de un flujo tras el cual su regla se elimina; prioridad, que define que regla de flujo debe considerarse antes o después que las otras; la estructura de comparación que se explicó anteriormente y las acciones que se deben realizar para el flujo, como por ejemplo, redireccionarlo por un puerto de salida específico, enviarlo a todos los puertos mediante un proceso de inundación, ponerlo en cola, reenviarlo al servidor controlador, etc. [5], [8].

Una vez introducido el marco teórico referencial de las SDN, se procede a describir la implementación de un prototipo de SDN, compuesta por un servidor controlador, dos conmutadores con soporte para OpenFlow y varios hosts. Esto se realizará tanto a nivel de simulación como de implementación física, y para ello se usarán las siguientes alternativas de software controlador: NOX, POX, Beacon y Floodlight.

El artículo se ha organizado de la siguiente manera: en primer lugar se presentan los elementos que conforman la red a simularse e implementarse físicamente y su diagrama de topología; posteriormente, se presenta tanto el desarrollo de la simulación así como de la implementación del prototipo, incluyendo una descripción de los componentes programados para los diferentes controladores para definir reglas que determinen la funcionalidad de la red; finalmente se presentan los resultados obtenidos y las correspondientes conclusiones.

II. TOPOLOGÍA DEL PROTOTIPO

A. Dispositivos utilizados

Para la simulación del prototipo se han utilizado los siguientes elementos:

- Un computador portátil Dell Inspiron 5520 con procesador Intel Core i5 @2.5 GHz, sistema operativo Windows 7 de 64 bits y memoria RAM de 8 GB.

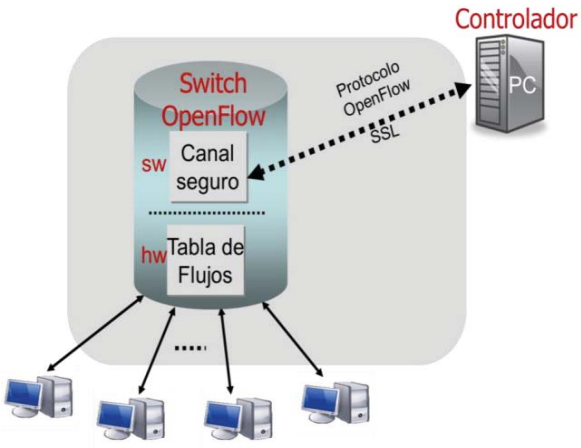

Fig. 1 Arquitectura SDN [5]

- Los requisitos de software para este computador son: un gestor de máquinas virtuales (VMware

Workstation, VirtualBox u otro); un sistema de ventanas X (para Windows se sugiere usar Xming [11]); y, un cliente de Secure Shell (SSH).

- La máquina virtual de Mininet, que viene con OpenFlow y NOX instalados, la cual se puede descargar de la página virtual de Mininet [3]. Esta máquina no incluye interfaz gráfica y para facilitar el trabajo con ella se emplea Xming.

Para la implementación del prototipo se han usado los siguientes elementos:

- Un servidor controlador, cuyo hardware posee las siguientes características: un procesador Intel Core i7 @3.4 GHz, memoria RAM de 8GB, disco duro de 500 GB y el sistema operativo VMware vSphere Hypervisor (ESXi) 5.1, sobre el cual se montarán las máquinas virtuales para la instalación de cada software controlador.
- Sobre el servidor controlador se crearon tres máquinas virtuales, cada una de ellas con el sistema operativo Ubuntu 12.04 LTS y ejecutando un controlador diferente. Se creará una máquina adicional para el manejo de la herramienta *dpctl*, que se explicará en la sección de análisis, con el mismo sistema operativo.
- Dos Access Point (AP) Linksys WRT54GL, con las siguientes características: CPU Broadcom BCM5352 @200MHz. Memoria RAM de 16MB y memoria flash de 4MB. Se utilizaron estos AP simplemente porque su firmware puede actualizarse con la versión de OpenWRT con soporte para OpenFlow [6], sin el cual no se podría implementar el prototipo SDN.
- Dos hosts con las mismas especificaciones de hardware del servidor controlador. En cada host se instalarán tres máquinas virtuales que harán las veces de hosts y se intercomunicarán usando la red SDN.
- Los enlaces físicos entre los dispositivos se realizarán utilizando cables UTP categoría 5e.

B. *Diagrama de topología*

En la Fig. 2 se muestra el diagrama de topología de la simulación y en la Fig. 3 el de la implementación. Las dos topologías son prácticamente las mismas, salvo pequeñas diferencias en los hosts; en la simulación se usaron hosts virtuales mientras que en la implementación se usaron hosts físicos con máquinas virtuales en ellos.

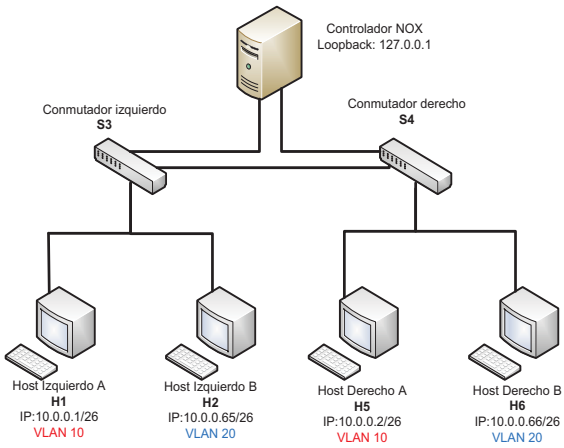

Fig. 2 Diagrama de topología para la simulación

III. SIMULACIÓN E IMPLEMENTACIÓN

A. *Simulación de la red*

Se desarrolló un script en Python para generar por programa la topología a usarse durante la simulación. Se usó como base el archivo topo-2sw-2host.py que se encuentra bajo el directorio ~/mininet/custom, contenido en la máquina virtual Mininet, considerando que provee una base similar a la topología de la Fig. 2. En este archivo se definen los conmutadores, hosts y el controlador que conforman la red así como los enlaces entre ellos [3].

Para probar el funcionamiento de la SDN creada con Mininet usando el controlador NOX, el cual, por defecto, genera las reglas que se transmitirán a los conmutadores para que actúen como conmutadores comunes de capa dos, se emplea la sentencia presentada en el Código 1.

Una forma manual de transmitir y añadir las reglas en el conmutador es usar la herramienta *dpctl*, a través de una línea de comandos de Windows o Linux con una sintaxis básica. Además, *dptcl* también permite realizar consultas de las reglas que se encuentran instaladas en los conmutadores.

```
$sudo mn –custom <archivo de creación de topología> -topo <nombre de la topología>
```

Código 1. Sentencia para ejecutar el simulador Mininet

Una vez entendido el proceso de transmisión de las reglas de flujos con la ayuda de *dpctl* se procede a programar un componente personalizado para ser usado con el controlador NOX; con lo que se consigue que en lugar de enviar reglas que reflejen el comportamiento de un conmutador regular, se envíen reglas definidas por el administrador, de acuerdo a la funcionalidad deseada.

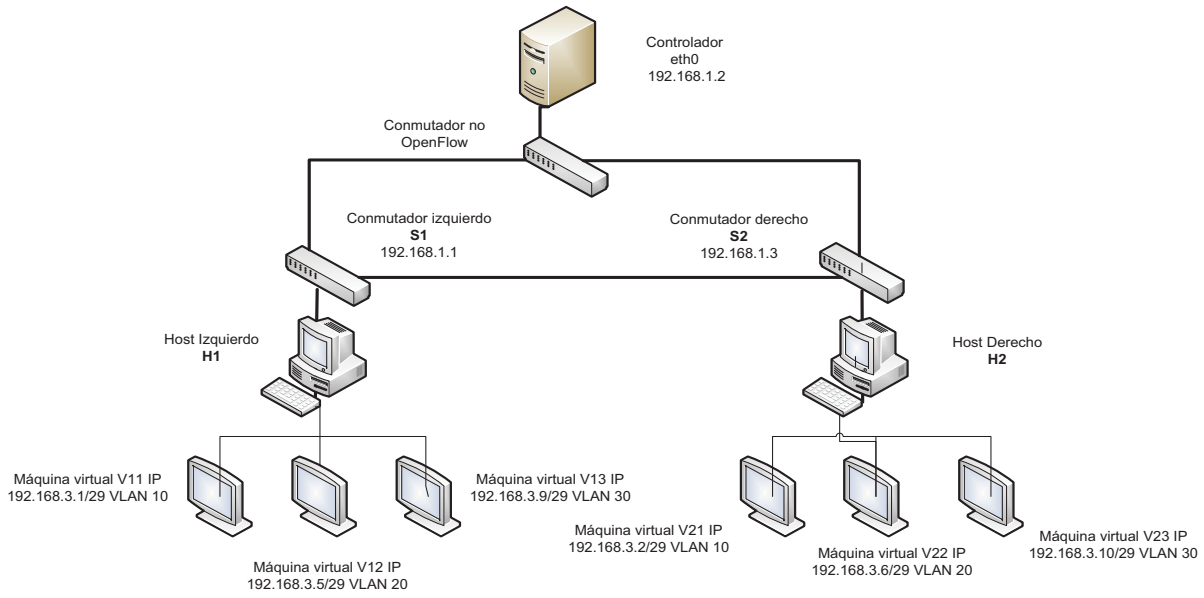

Fig. 3 Diagrama de topología para la implementación

En primer lugar se debe identificar qué lenguaje de programación usa el controlador y qué reglas se pretenden definir. En el caso de la simulación en Mininet con el controlador NOX, la programación se realiza en Python, aunque existe otra versión que usa C++ [4].

El archivo del componente programado debe tener una extensión .py y es recomendable que esté localizado bajo el directorio ~/noxcore/build/src. El componente desarrollado lee los parámetros que conformarán las reglas de flujos de un archivo de configuración y los transmite a los conmutadores.

Para la simulación se generó tráfico ICMP (Internet Control Message Protocol) desde los hosts creados con Mininet usando el comando *ping*, lo cual inicialmente genera tráfico ARP (Address Resolution Protocol). En el componente desarrollado a ejecutarse en el controlador se programaron las reglas para los flujos tanto para ICMP (en este caso se lo realizó mediante un archivo de configuración) así como para ARP (que se lo definió directamente en el componente). Cabe mencionar que Mininet permite que los usuarios programen aplicaciones que generen otro tipo de tráfico desde los hosts.

En el caso de ARP, se permite el intercambio de mensajes solo entre los hosts H1 y H5 (Fig. 2). Las solicitudes ARP son fundamentales para poder realizar la traslación entre direcciones lógicas capa tres del modelo OSI y direcciones físicas de capa dos. Para ello, se deben definir dos flujos, uno de ida y otro de vuelta.

Los atributos que se deben tomar en cuenta para la definición de la regla para el flujo de ida son:

1. Tipo de paquete encapsulado en capa 2: ARP (identificado por el número hexadecimal 0x0806).
2. Puerto físico de entrada: 1 (definido en el *script* para la generación de la topología).
3. Acciones: se debe redireccionar el tráfico por el puerto físico 3.

El flujo de retorno ARP tendría las mismas características, salvo que el puerto de entrada se intercambia con el puerto de salida definido en las acciones a tomarse.

```
#código para el conmutador S3 cuyo identificador
(dpid) #es 3, valor que se otorgó en el script para la
creación de #la topología
if (dpid == 3 ):
 idle_timeout =
openflow.OFP_FLOW_PERMANENT
 hard_timeout =
openflow.OFP_FLOW_PERMANENT
 attrs = {}
 attrs[core.DL_TYPE] = ethernet.ARP_TYPE
 attrs[core.IN_PORT] = 1
 actions = [[openflow.OFPAT_OUTPUT, [0, 3]]]
 self.install_datapath_flow(dpid, attrs, idle_timeout,
```

Código 2. Regla de flujo ARP para NOX

A manera de ejemplo, en el Código 2 se puede observar cómo se introduce la regla de flujo ARP, definida en Python, para el camino de ida en el componente desarrollado para el controlador NOX. En primer lugar se identifica a que conmutador enviar la regla de flujo (S3), posteriormente se definen contadores que definen el tiempo durante el cual la regla debe mantenerse en la tabla de flujos del conmutador S3 y se crea un diccionario denominado *attrs* con los atributos de la regla de flujo, en este caso el tipo de paquete capa dos y el puerto físico por el que los paquetes del flujo entrarán al conmutador. Posteriormente se indica que acciones realizar, en este caso reenviar los paquetes por el puerto 3 y finalmente se envía e instala la regla en el conmutador.

Para el caso de ICMP, se permite el intercambio de mensajes exclusivamente entre los hosts H1 y H5 (Fig. 2). Para ello se deben definir dos flujos, uno de ida y otro de vuelta. Los atributos que se deben tomar en cuenta para el flujo de ida son:

1. Tipo de paquete encapsulado en capa 2: IP (identificado por el número hexadecimal 0x800).
2. Puerto físico de entrada: 1 (definido en el *script* para la generación de la topología).
3. Dirección IP origen: 10.0.0.1
4. Dirección IP destino: 10.0.0.2
5. Protocolo transportado sobre la capa tres: ICMP (identificado por el número 1).
6. Acciones: se debe redireccionar el tráfico por el puerto físico 3.

De igual manera, para el flujo de retorno ICMP se intercambia el puerto de entrada con el puerto de salida. Como se puede observar, estas reglas de flujo para ICMP toman características adicionales a las de ARP. La complejidad de la regla de flujo depende de qué acciones desee implementar el administrador de la red.

Una vez finalizada la programación del componente, se lo ejecuta en conjunto con Mininet para la simulación de la SDN. Los resultados obtenidos se presentarán y analizarán en la sección cuatro.

B. Implementación del prototipo

Para la implementación del prototipo, en primer lugar se habilita los dispositivos Linksys WRT54GL con OpenWRT con soporte OpenFlow para que envíen las peticiones al servidor controlador, cuya dirección IP en el prototipo es 192.168.1.2 (Fig. 3): además, el puerto en el que el controlador escucha las peticiones OpenFlow de los conmutadores es el 6633. Para especificar la dirección IP y el puerto del controlador, se modifica el archivo openflow, ubicado bajo el directorio /etc/config en los conmutadores.

Posteriormente, se instalan los controladores POX [9], Beacon [1] y Floodlight [2] en el servidor controlador. Las reglas definidas en cada controlador para la topología de la Fig. 3 son exactamente las mismas. Estas reglas se detallan a continuación:

Para el caso de ARP, se permite el intercambio de mensajes entre todos los hosts que componen la red, con reglas de flujo similares a las presentadas en la simulación, salvo los parámetros como los puertos de entrada y salida. En el Código 3 se observa el equivalente a la definición de un flujo ARP presentado en el Código 2 para el controlador Floodlight y en el Código 4 se puede observar la regla de flujo ARP para Beacon.

Para el controlador POX, la definición es similar al controlador NOX. Para Beacon (Código 4), las reglas se definen en Java, para lo que en primer lugar se debe crear un objeto que represente el flujo, agregar los contadores que se incluyeron en NOX, crear un objeto que maneje los atributos de la regla de flujo y definir el parámetro *wildcards*, que permite incluir o ignorar ciertos elementos de la estructura de comparación; posteriormente se crea un objeto con las acciones a realizarse y finalmente se envía la regla a los conmutadores. Para Floodlight (Código 3), se usa el comando *curl*, que permite enviar mensajes conformados por URL (Uniform Resource Locator) a un componente de Floodlight denominado Static Flow Entry Pusher. Estos mensajes incluyen parámetros como el identificador del conmutador, los parámetros de la regla de flujo, las acciones a realizarse y la dirección del controlador. El componente toma la información proporcionada y envía la regla de flujo al conmutador especificado.

Para el caso de ICMP, se permite el intercambio de mensajes exclusivamente entre los hosts V11 y V21 (Fig. 3), de manera similar a lo que se realizó en la simulación, salvo por variaciones en parámetros de las reglas.

Para el caso de HTTP (*HyperText Transfer Protocol*), se permite el intercambio de mensajes exclusivamente entre los hosts V11 y V21. El host V11 hace las veces de servidor HTTP, por lo que el host V21 enviará peticiones hacia el servidor HTTP y recibirá respuestas, las cuales procesará y presentará en el navegador web. El servidor manejará un puerto capa cuatro bien conocido, el 80 y el cliente usará un puerto aleatorio, por lo que la regla de flujo deberá reflejar esta información.

```
curl -d '{"switch": "00:00:00:23:20:b2:8c:28",
"name":"flow-mod-arp1", "ingress-port":"2", "ether-
type":"0x0806", "actions":"output=3"}'
http://192.168.1.2:8080/wm/staticflowentrypusher/
```

Código 3. Regla de flujo ARP para Floodlight

```
OFFlowMod fm = new OFFlowMod();
fm.setCommand(OFFlowMod.OFPFC_ADD);
fm.setIdleTimeout((short) 60);
OFMatch match =
OFMatch.load(pi.getPacketData(), pi.getInPort());
match.setInputPort((short) 2);
 match.setDataLayerType((short) 0x0806);
 match.setWildcards(1581102);
 short outPort = 3;
fm.setMatch(match);
OFAction action = new OFActionOutput(outPort);
fm.setActions(Collections.singletonList((OFAction)
action));
sw.getOutputStream().write(fm);
```

Código 4. Regla de flujo ARP para Beacon

Los atributos que se deben tomar en cuenta para el flujo de ida son:

1. Tipo de paquete encapsulado en capa 2: IP (identificado por el número hexadecimal 0x0800).
2. Puerto físico de entrada: 2 (puerto del conmutador).
3. Dirección IP de origen: 192.168.3.1.
4. Dirección IP de destino: 192.168.3.2.
5. Tipo de segmento encapsulado en capa 3: TCP (identificado por el número 6).
6. Puerto origen: 80 (HTTP).
7. Puerto destino: No se especifica ninguno, al ser un puerto aleatorio.
8. Acciones: redireccionar el tráfico por el puerto físico 3.

Para el camino de retorno, se intercambian los parámetros puerto físico origen y destino, dirección origen y destino y puerto capa cuatro origen y destino.

Para el caso de Telnet, se permite el intercambio de tráfico entre los hosts V12 y V22. El host V22 hará les veces de servidor y el host V12 las de cliente. En esta comunicación, el cliente Telnet inicia una conexión mediante una terminal virtual con el servidor, lo que le permite manejar al servidor de manera remota. El servidor recibe peticiones en un puerto bien conocido (23), mientras que el cliente usa un puerto aleatorio. Los atributos que se deben tomar en cuenta para el flujo de ida son:

1. Tipo de paquete encapsulado en capa 2: IP (identificado por el número hexadecimal 0x0800).
2. Puerto físico de entrada: 2 (puerto del conmutador).
3. Dirección IP de origen: 192.168.3.5.
4. Dirección IP de destino: 192.168.3.6.
5. Tipo de segmento encapsulado en capa 3: TCP (identificado por el número 6).
6. Puerto origen: No se especifica ninguno, al ser un puerto aleatorio.
7. Puerto destino: 23 (Telnet)
8. Acciones: redireccionar el tráfico por el puerto físico 3.

El camino de retorno se crea de la misma manera que en la regla de flujo anterior.

Para el caso de video *streaming,* se permite el intercambio de mensajes entre los hosts V13 y V23. El host V23 hará las veces de servidor y el host V13 las de cliente. En esta comunicación, el cliente envía peticiones al servidor para que le entregue el contenido del *streaming* de video, que será reproducido en el cliente. El servidor transmitirá el video, en este caso mediante el protocolo HTTP encapsulado sobre TCP. En el caso de *streaming* de video se puede usar también comunicación sobre el protocolo de transporte UDP (User Datagram Protocol). El servidor recibe peticiones en el puerto número 8081, que es un puerto asignado cuando se inicia el *streaming* de video, mientras que el cliente usa un puerto aleatorio. Los atributos que se deben tomar en cuenta para el flujo de ida son:

1. Tipo de paquete encapsulado en capa 2: IP (identificado por el número hexadecimal 0x0800).
2. Puerto físico de entrada: 2 (puerto del conmutador).
3. Dirección IP de origen: 192.168.3.9.
4. Dirección IP de destino: 192.168.3.10.
5. Tipo de segmento encapsulado en capa 3: TCP (identificado por el número 6).
6. Puerto origen: No se especifica ninguno, al ser un puerto aleatorio.
7. Puerto destino: 8081 (Asignado al servidor de video *streaming*)
8. Acciones: redireccionar el tráfico por el puerto físico 3.

El camino de retorno se crea de la misma manera que en la regla de flujo anterior.

Estas son las reglas de flujo que han sido definidas para comprobar las características de las SDN y el funcionamiento de las reglas de flujo. En la siguiente sección, se analizarán los resultados obtenidos.

IV. PRUEBAS Y RESULTADOS

Para la simulación del prototipo, la primera prueba realizada para verificar si los flujos se transmitieron a los conmutadores es usar el comando *dpctl dump-flows*, que se encarga de desplegar información detallada de cada flujo. Como ejemplo, la Fig. 4 indica las características representativas de un flujo como: prioridad del flujo (*priority*), contadores (*idle_timeout y hard_timeout*) y la estructura de comparación (flujo de tipo IP, con la dirección de origen 10.0.0.1 y destino 10.0.0.2) y las acciones a realizarse con el flujo, que en este caso indican una redirección hacia el puerto físico número tres [3], [8].

A continuación se realiza una prueba de conectividad básica con el siguiente escenario: el host H1 envía mensajes ICMP al host H2. El primer intercambio de mensajes ICMP se demora hasta que se hayan establecido los flujos ARP e ICMP y se haya obtenido la dirección física de H2 a partir de su dirección lógica con ARP. Esto porque algunos paquetes asociados al comando *ping* (paquetes ICMP y ARP iniciales) deben enviarse al controlador, considerando que en un inicio no existe una regla específica en los conmutadores. Los mensajes consiguientes se envían y reciben en menor tiempo, una vez que los conmutadores cuenten con las reglas instaladas en sus tablas de flujos, como se indica en la Fig. 5.

Una vez probada la conectividad entre los hosts H1 y H5, se prueba la conectividad entre todos los hosts de la red. Para ello, en la línea de comandos de Mininet, se puede utilizar el comando *pingall*, que básicamente enviará mensajes ICMP a todos los dispositivos y esperará sus respuestas. Como se indica en la Fig. 6, solo es posible la comunicación entre los hosts H1 y H5.

Como siguiente prueba, en la implementación física, se inicia la captura de los paquetes que atraviesan la red con el software Wireshark, para identificar los campos que componen el protocolo OpenFlow. El primer mensaje que se intercambia en el establecimiento de una conexión OpenFlow se denomina *Hello*; el mensaje *Features Request* se usa cuando se solicita la información de las reglas de flujo existentes en el conmutador con *dpctl* y la respuesta a este se denomina *Features Reply*. En la Figura 7 se indica en detalle el contenido de un paquete *Features Reply*, en donde se pueden observar las características del conmutador [8].

Cuando se envía una regla de flujo se usa un mensaje *Flow Mod,* que contiene las estructuras de comparación, contadores y acciones a realizarse, como se observa en la Fig. 8.

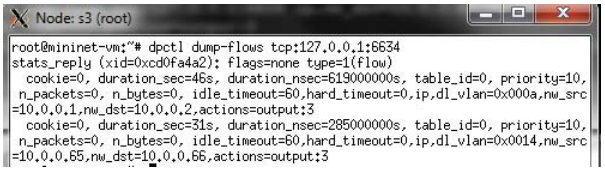

Fig. 4 Resultado de la ejecución del comando *dpctl* dump-flows

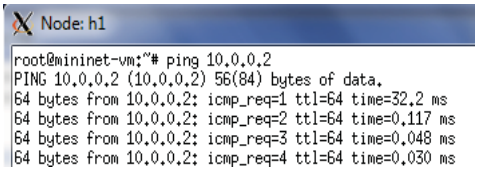

Fig. 5 Mensajes ICMP transmitidos entre H1 y H5

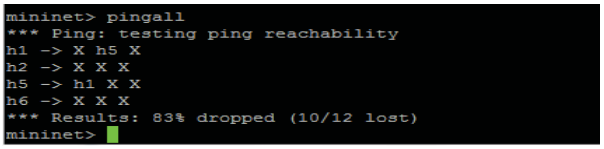

Fig. 6 Prueba de conectividad entre los hosts que componen la simulación

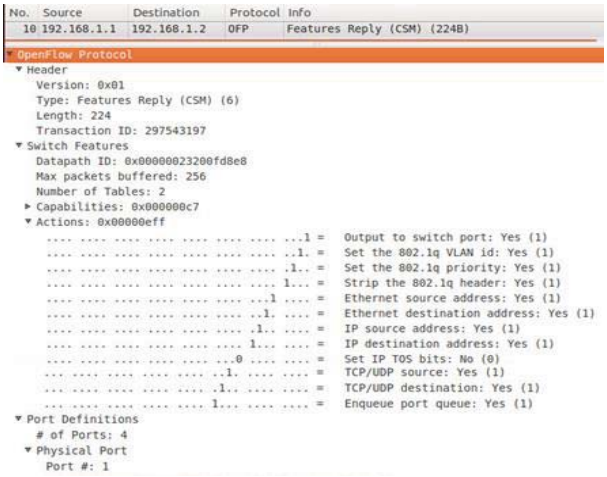

Fig. 7 Detalle de la captura de un paquete *Features Reply*

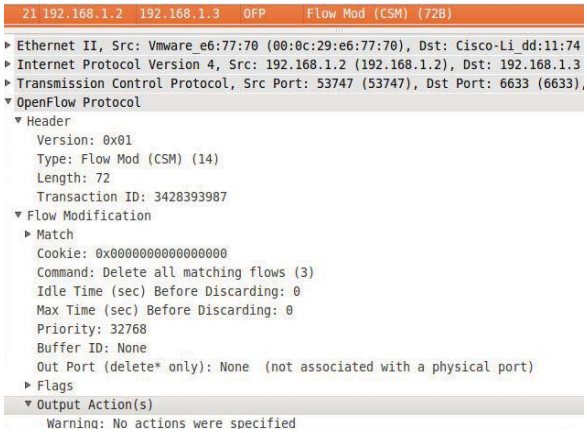

Fig. 8 Detalle de la captura de un paquete *Flow Mod*

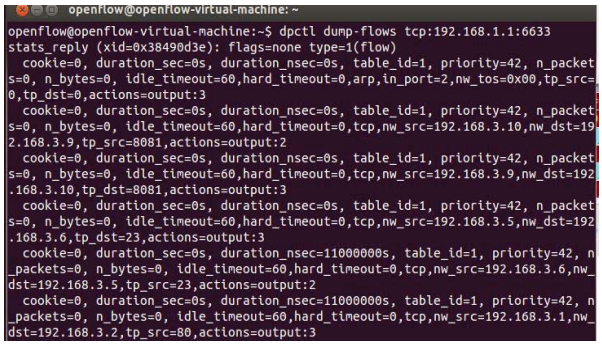

Fig. 9 Verificación de la instalación de flujos en S1

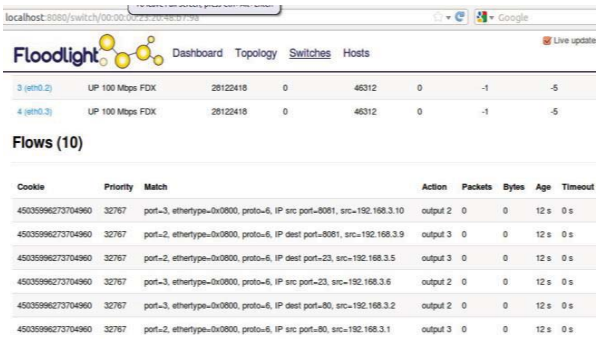

Fig. 10 Visualización de los flujos instalados en S1 desde el controlador Floodlight

Como última prueba, se verifican que todas las reglas de flujo definidas en cada controlador, por separado, estén instaladas en los conmutadores, para lo cual se puede usar el comando *dpctl*.

En la Fig. 9 se muestra el resultado de la ejecución del comando *dpctl dump-flows* en el conmutador S1, en el que se detallan los flujos que se han agregado al mismo de manera resumida, para uno de los controladores utilizados.

Algunos controladores tienen interfaces web que permiten visualizar de una manera más amigable la información de los flujos, así como las características del controlador y de los conmutadores. Así, la interfaz de Beacon permite visualizar los nodos que conforman la topología, y en Floodlight se puede visualizar la información de los flujos y de los conmutadores de la red, como se muestra en la Fig. 10.

La principal utilidad de una interfaz web es la de proporcionar información de forma más ordenada. Existen interfaces que permiten crear un flujo, como es el caso del controlador SNAC [10], sin embargo SNAC no ha sido adaptado a los sistemas operativos lanzados recientemente y, por lo tanto, no se considera un controlador actual, a diferencia de los cuatro controladores que se implementaron en este proyecto.

Cada controlador tiene características específicas en cuanto a su uso. Por ejemplo, en POX, las reglas se debieron programar en Python y se tuvo que importar librerias en las que se definen las estructuras de comparación y los paquetes que se van a enviar a los conmutadores. NOX, al ser una línea de desarrollo paralela a POX, posee prácticamente las mismas caracterísitcas.

En el caso de Beacon, se recomienda usar un entorno de desarrollo integrado, como por ejemplo Eclipse o Net Beans, para la programación de componentes personalizados; debido a que en estos entornos se incluyen ayudas para los atributos y métodos que posee un determinado objeto.

Como Beacon se basa en Java, la programación del componente debió ser escrita en este lenguaje, por lo que se crearon objetos para el manejo de conexiones y estructuras de comparación. En este caso también se tienen disponibles librerías que estructuran los paquetes que deben ser enviados a los conmutadores y contienen los métodos usados en los objetos creados para las estructuras de comparación.

En el caso de Floodlight, en lugar de programar las reglas en código, se puede usar el comando *curl* para el envío de reglas de flujo a un componente conocido como Static Entry Flow Pusher, que se encarga de interpretar la regla enviada y a su vez transmitirla a los conmutadores.

En este caso no se deben crear objetos ni estucturas de comparación, sino mediante el comando *curl,* establecer un URL con todos los datos requeridos para su envío a los conmutadores. Sin embargo, es buena idea crear un *script* que realice este proceso automáticamente y así permitir que el administrador pueda definir reglas en un archivo de configuración sin necesidad de conocer los detalles de la estructura del comando *curl.*

De lo realizado con todos los controladores, se puede concluir que la programación de los cuatro componentes sigue una lógica similar, sin embargo, las diferencias radican en el cambio de los nombres de ciertos métodos o variables y la manera como se llama a los archivos de configuración, siempre considerando las diferencias de los lenguajes de programación empleados.

En la programación de los componentes se puede observar que el administrador tiene el control completo sobre todo el tráfico que pasa por la red y lo puede encaminar de acuerdo a sus necesidades. Un conmutador OpenFlow, haciendo una analogía con las redes tradicionales, puede ser un conmutador, un enrutador y un firewall a la vez, lo que representa una ventaja tanto de rendimiento, así como económica y de espacio, al tener todo en un mismo equipo.

Las reglas que se definieron pueden modificarse para cualquier necesidad que tenga el administrador. Se puede bloquear hosts con determinadas direcciones,

proporcionar permisos a segmentos de una compañía, permitir el acceso desde Internet únicamente a los servidores web de una compañía, mientras se permite el acceso de los usuarios de la red de la compañía a todos los recursos, etc.

Fig. 11. Prueba de aplicación: video *streaming*

En la Fig. 11 se puede observar la recepción de video streaming en el cliente V13 generado en el servidor de streaming V23 (Fig. 3).

Las reglas definidas son netamente experimentales y el administrador se encuentra en libertad de definir cualquier regla que desee para moldear la red según sus requerimientos. Por ejemplo, se puede prohibir el paso de mensajes ICMP hacia un servidor web con el fin de disminuír la posibilidad de un ataque de denegación de servicio.

V. CONCLUSIONES

La arquitectura SDN tiene como principio fundamental la separación de planos de control y de datos y se basa en el protocolo OpenFlow para la comunicación entre el o los servidores controladores y los conmutadores.

Se implementó un prototipo con conmutadores físicos habilitados, totalmente funcional, con el cual se pudo disponer de una infraestructura de prueba (*testbed*) sobre la cual se evidenciaron los beneficios de una SDN y, en particular, la capacidad que se brinda al administrador de modificar o programar la red según sus necesidades. En este prototipo de SDN se emplearon varios controladores y se desarrollaron componentes para cada uno de ellos empleando el lenguaje de programación predefinido; estos componentes definieron reglas para flujos de paquetes determinados y se alteraron las reglas de una manera rápida para adaptarse a los cambios que surjan en la red, mediante la modificación de archivos de configuración.

Durante las pruebas del prototipo se pudo constatar esta aseveración que se realizó en el marco teórico. Por un lado, se establecieron conexiones entre los conmutadores S1 y S2 y el controlador que solo manejan tráfico de administración, encapsulado en el protocolo OpenFlow (Figura 3). A este tipo de tráfico corresponden los mensajes de modificación de flujos, de establecimiento del enlace OpenFlow, etc. Por otro lado se establecieron enlaces entre los hosts (máquinas virtuales de H1 y H2) y los conmutadores, que solo manejan los datos de las aplicaciones. De esta manera se puede comprobar que los enlaces de datos no se ocupan para la gestión de la red y por tanto, tienen una mayor capacidad de canal para datos, lo que supone una gran ventaja en redes altamente congestionadas.

Existen varias alternativas de plataformas de software que permiten la implementación de un servidor controlador, entre las que se seleccionaron cuatro; en primer lugar está NOX, para el cual se implementó un componente en Python y se lo empleó en conjunto con Mininet para simular una SDN y se comprobó que su funcionamiento, y en particular el del componente, era adecuado según las reglas de flujo definidas. Este componente se puede adaptar para implementaciones físicas, por ejemplo, para tener una mayor cobertura de filtrado, ya que en escenarios reales se puede usar una mayor gama de tipos de tráfico que la simulación básica no permite, por estar limitada a mensajes ICMP, a menos que se recurra a programar aplicaciones que generen otro tipo de tráfico dentro del ambiente de Mininet.

En segundo lugar se encuentra POX, que es una plataforma muy versátil y que es recomendada para el desarrollo de componentes personalizados, como es el caso del componente que se creó para el prototipo. La función del componente es la de leer parámetros de configuración de un archivo y enviar reglas basadas en estos parámetros a los conmutadores mediante mensajes de modificación de flujo. Además, se pudo constatar que este software es el que más documentación tiene en Internet y por ello su estudio y comprensión puede tomar menos tiempo que los otros controladores. Finalmente, el lenguaje Python es relativamente sencillo de entender y aprender, por lo que se puede manejar POX en un tiempo razonable leyendo la documentación disponible.

En tercer lugar se encuentra Beacon, que presenta una mayor dificultad al momento de programar que su similar POX porque tiene estructuras más complejas y métodos que se deben analizar cuidadosamente. Por ejemplo, en la implementación del prototipo con el controlador Beacon se programaron ciertos parámetros adicionales que no fueron necesarios en POX, así como por ejemplo los *wildcards*, que se encargan de definir qué parámetros se toman en cuenta y que parámetros se ignoran al realizar una comparación entre un paquete de un flujo entrante y una regla de flujo (por ejemplo ignorar puerto origen de la comunicación entre un host A y un host B, pero si tener en cuenta el puerto destino), por lo que es necesario definir este parámetro para cada tipo de

regla que se pretenda crear, y si no se define este parámetro, las reglas pueden llegar a ser inconsistentes. Esto sin embargo, puede ser beneficioso para el administrador, ya que le permite tener un mayor control sobre la estructura de comparación para las reglas de flujos; por ejemplo, el usuario puede definir parámetros y transmitirlos del controlador al equipo, pero no tomarlos en cuenta al momento de la comparación, simplemente modificando el valor de los *wildcards*.

Finalmente, se encuentra Floodlight, que tiene una gran aceptación a nivel comercial debido a las facilidades que ofrece. Además, cuenta con una interfaz web bastante amigable, a través de la cual se pueden definir reglas de manera mucho más sencilla que el resto de controladores mediante el comando *curl*.

Durante la implementación del prototipo de la SDN se modificó el firmware de los conmutadores Linksys WRT54GL, ya que fueron adaptados para habilitarlos y que funcionen con el protocolo OpenFlow. La actualización del firmware es un proceso que requiere especial atención porque existen ocasiones en las que la imagen se corrompe, por lo que se debe regresar a un modo de CLI básico mediante el cual se puede descargar una nueva imagen y realizar la actualización del firmware. Sin embargo, muchas veces esto no es posible, por lo que se debe desarmar el equipo y usar un cable JTAG para realizar este proceso. Estos equipos no son diseñados para soportar este protocolo, solo han sido adaptados, por lo que el rendimiento frente a un equipo diseñado para este propósito es mucho menor y las funcionalidades son diferentes.

Por último, cabe resaltar la utilidad de la herramienta de simulación Mininet, ya que el componente desarrollado para la simulación puede migrarse a un controlador físico. La herramienta es muy útil cuando no se tiene una infraestructura de pruebas, aunque no es recomendada para verificar los tiempos de respuesta del controlador, porque al no existir retardos ocasionados por conexiones físicas, los tiempos de transmisión no se ajustan a la realidad.

REFERENCIAS

[1] Beacon Stanford. https://openflow.stanford.edu/display/Beacon/Configuration (Consultado el 12 de Junio de 2013)

[2] Floodlight. http://www.projectfloodlight.org (Consultado el 8 de Marzo de 2013)

[3] Mininet. http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet (Consultado el 23 de Noviembre de 2012)

[4] NOX. http://www.noxrepo.org/nox (Consultado el 9 de Noviembre de 2012)

[5] OpenFlow. http://www.openflow.org (Consultado el 6 de Noviembre de 2012)

[6] OpenFlow OpenWRT. http://www.openflow.org/wk/index.php/OpenFlow_1.0_for_OpenWRT (Consultado el 20 de Mayo de 2013)

[7] OpenFlow Stanford. https://openflow.stanford.edu (Consultado el 14 de Noviembre de 2012)

[8] ONF, "Software-Defined Networking: The New Norm for Networks", USA, Abril 2012. OpenFlow White Paper. http://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf (Consultado el 6 de Noviembre de 2012)

[9] POX. http://www.noxrepo.org/pox (Consultado el 15 de Noviembre de 2012)

[10] SNAC http://archive.openflow.org/wp/snac/ (Consultado el 6 de Noviembre de 2012)

[11] Xming. http://xming-x-server.uptodown.com (Consultado el 6 de Septiembre de 2013)

Juan Carlos Chico Jiménez, nacido en Quito, Ecuador el 22 de Julio de 1989. Realizó sus estudios secundarios en el Liceo José Ortega y Gasset, en donde obtuvo el título en Bachiller en Ciencias. En el año 2012 egresó de la carrera de Ingeniería Electrónica y Redes de Información en la Escuela Politécnica Nacional. Sus áreas de interés actuales son: Redes Definidas por Software, redes de computadoras, *cloud computing* y administración de negocios informáticos.

Raúl David Mejía Navarrete. Obtuvo el título de Ingeniero en Electrónica y Redes de Información, en la Escuela Politécnica Nacional en Quito – Ecuador, en 2005. Obtuvo el título de Magister en Multimedia y Comunicaciones en la Universidad Carlos III de Madrid en España en 2011. Actualmente se desempeña como Profesor Auxiliar a Tiempo Completo en el Departamento de Electrónica, Telecomunicaciones y Redes de Información.

Iván Marcelo Bernal Carrillo, graduado del Instituto Nacional Mejía. Obtuvo el título de Ingeniero en Electrónica y Telecomunicaciones en la Escuela Politécnica Nacional en 1992. Obtuvo los títulos de M.Sc. (1997) y Ph.D. (2002) en Computer Engineering en Syracuse University, NY, USA. Ha realizado cursos especializados en varios países europeos, latinoamericanos, Estados Unidos y en Corea del Sur. Realiza tareas de investigación, docencia y gestión en la Escuela Politécnica Nacional, en el Departamento de Electrónica, Telecomunicaciones y Redes de Información desde 1991. Sus áreas de interés actuales son las comunicaciones inalámbricas, TV digital, diseño y síntesis de hardware, computación distribuida y redes de computadoras.

Instrumentación para Nanotecnología

RESUMEN:

El avance en la Investigación y el Desarrollo de la Nanotecnología es posible gracias al desarrollo de instrumentos de medida y prueba a niveles nanométricos, para satisfacer necesidades científicas y educativas. Con soluciones de prueba para nanofabricación, nanoestampación, ciencias de la vida, química, electroquímica, bioquímica, chips, semiconductores, fluorescencia, agricultura, nanomecánica, etc.

Se presenta el Sistema de Microscopía con Barrido de Microonda, especialmente útil para prueba de semiconductores y su caracterización con la más alta resolución espacial y eléctrica y la mejor sensibilidad y rango dinámico de la industria, para mediciones de impedancia compleja (resistencia y reactancia), capacitancia y densidad de dopante calibradas, en estudios de difusión y de topografía de semiconductores a escala nanométrica.

ABSTRACT:

Advances in Research and Development of Nanotechnology is possible by the development of test and measurement instruments for nanometric levels to meet scientific and educational needs. With solutions for nanofabrication, nanoembossing, life sciences, chemistry, electrochemistry, biochemistry, chipsets, semiconductors, fluorescence, agriculture, and nanomechanical testing, etc.

The Scanning Microwave Microscopy System is introduced, it is especially useful for semiconductor testing and characterization, with the highest spatial and electrical resolution, and the best sensitivity and dynamic range of its kind, allowing measurements of complex impedance (resistance and reactance), calibrated capacitance and calibrated dopant density in diffusion studies and topography of semiconductor at nanometric scale.

INTRODUCCIÓN:

Sin la instrumentación utilizada para caracterizar materiales no existiría la Nanotecnología como la conocemos actualmente.

La espectrometría de masas de Hewlett Packard permitió identificar, a mediados de los 80, varias estructuras (esférica, elipsoidal y de tubo), altamente simétricas, formadas solo con átomos de carbono y de dimensiones nanométricas (1 nm).

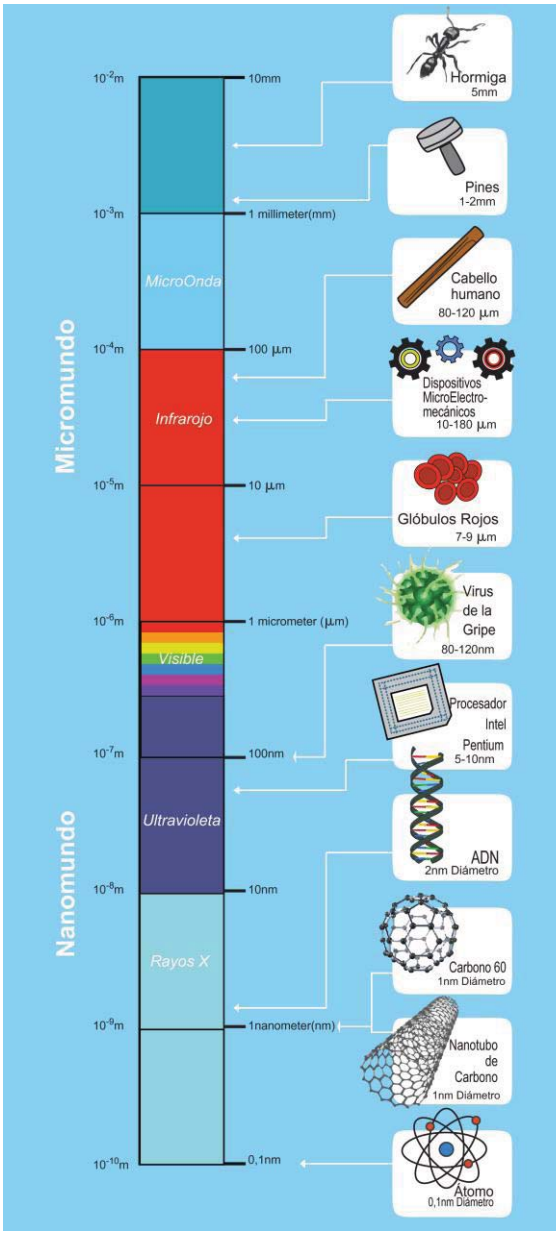

Figura 1: Micro y Nano Escalas[1]

Por ejemplo la estructura C_{60} que consta de 60 átomos de carbón organizados en pentágonos y hexágonos, similares a una pelota de fútbol, fue bautizada "buckyball" (ver figura 1)[1].

Los investigadores de nanotecnología tienen la capacidad de visualizar y trabajar con la materia a nanoescala con ayuda de los instrumentos apropiados.

INSTRUMENTACIÓN NANOMÉTRICA:

El primer Microscopio de Fuerza Atómica **AFM** Atomic Force Microscope) se construyó en 1986. Junto a otros Microscopios con Sonda de Barrido **SPM** (Scanning Probe Microscopy) se convierten en instrumentos fundamentales para investigación, facilitando las mediciones y manipulaciones en la escala nanométrica.

Fig. 2. KeySight-Agilent[2] SPM/AFM 5420.

Cada vez más universidades incorporan en sus laboratorios el microscopio de fuerza atómica AFM. Los alumnos se familiarizan con imágenes tridimensionales de nanotubos o del ADN. Los investigadores observan y pueden manipular los materiales a niveles moleculares.

La investigación de nanotecnología demanda más equipos de vanguardia; la industria requiere instrumentos fáciles de usar y robustos para el trabajo de metrología y control de calidad en la fabricación; los educadores buscan herramientas para capacitar a los estudiantes y satisfacer la demanda de trabajo en nanotecnología.

A finales del 2008 aparecen AFMs capaces de manejar las obleas de semiconductores, con escaneado rápido, control posicional más preciso y con muy alta resolución para pequeñas muestras.

La velocidad es un factor limitante para el AFM. Para escanear más rápido, los sujetadores de las sondas deben moverse a intervalos menores a 10 micras, mucho menor que los AFM típicos.

Se han desarrollado complejos acoplamientos entre los AFMs y los microscopios ópticos, así como otras técnicas avanzadas, junto con mejoras en los métodos de detección de barrido SPM, como la microscopía de fuerza piezo-respuesta para el estudio de materiales ferro- y piezo-eléctricos.

KeySight-Agilent[3] (que mantiene el ADN de la instrumentación Hewlett Packard) desarrolló el modelo 5420 de AFM (ver fig. 2) y está diseñado con la electrónica de ruido más bajo y mejor óptica visual. Con un diseño fácil de usar para análisis de materiales, polímeros y caracterización general de superficies, a ser usado en el laboratorio o en el aula. Los profesores los utilizan para presentar a los estudiantes las técnicas de AFM a través de un plan de estudios de pregrado y muestras que KeySight-Agilent proporciona.

El AFM 5420 ofrece capacidades avanzadas para caracterización eléctrica. La técnica desarrollada, conocida como Modo Eléctrico de un solo paso ESPM (Electrical single-pass mode) permite su utilización para la nueva electrónica de bajo ruido que requiere de la mayor resolución posible, con microscopía de fuerza Kelvin (KFM), microscopía de fuerza eléctrica (EFM) y microscopía de fuerza piezo-eléctrica (PFM). El modo de barrido de microondas con microscopía SMM (Scanning Microwave Microscopy) permite conseguir una muy alta sensibilidad eléctrica, calibrada y con caracterización espacial.

Para hacer asequible el AFM 5420 se omiten los excesivos controles ambientales y térmicos, así como otras características avanzadas de este instrumento, ya que serían subutilizados en el aula o en un pequeño laboratorio de universidad, así pueden trabajar en condiciones ambientales y no requieren una complicada preparación de la muestra. En algunas universidades hay planes de estudios que introducen a los estudiantes en química, biología y física en la nanoescala.

Los microscopios de fuerza atómica AFMs se han adaptado a las técnicas de nanofabricación, y a las tecnologías de nanoestampación, como:

fotolitografía, litografía por haz de electrones, y litografía por nanoimpresión, que son utilizadas por varias empresas que han emprendido en esta nueva industria.

La litografía por haz de electrones junto con la microscopía electrónica relacionadas, compiten con las empresas ópticas. Los microscopios electrónicos permiten una alta resolución, pero su funcionamiento requiere una preparación más compleja de las muestras.

La electrónica de barrido y de transmisión para los microscopios y los instrumentos de iones y de haz de electrones (o de doble haz) se incorpora a la investigación electrónica, ciencias de la vida, y mercados industriales. Se puede apreciar una dinámica muy grande en las nuevas áreas de la nanotecnología, que se extiende cada vez más hacia economías pequeñas.

Con la microscopía óptica se descubrió que con un rayo láser especialmente configurado y bien enfocado, al tratar de detectar nanopartículas en fondos complejos, se las podía ver en un líquido, bajo un microscopio óptico. Igual que se ven partículas de polvo en un rayo de sol en una habitación oscura. Se las ve en realidad, pero no la imagen de las partículas. No podemos ver qué forma tienen, ni cómo se ven, o de qué están compuestas, pero dependiendo de la forma en que se mueven podemos apreciar su tamaño. Con videos de las partículas en movimiento, la distribución de tamaños se podía calcular a partir de su movimiento browniano.

Hay sistemas integrados por un microscopio, un módulo de iluminación óptico basado en láser, una cámara, una PC y un software para el seguimiento y el análisis de nanopartículas. Algunas empresas farmacéuticas y químicas han utilizado estos sistemas como herramientas de investigación y desarrollo, pero no para el control de calidad.

Si consideramos los AFMs de alta resolución o los microscopios electrónicos como muy sofisticados para el trabajo, que en la práctica no lo son, el método que más se le acercaba era el de la dispersión de luz dinámica, que era más caro. Este método era rápido y reproducible. Tenía un buen comportamiento si la muestra no estaba agregada o mono dispersada. El software de seguimiento de partículas brindaba información sobre muestras heterogéneas en menos de un minuto. Para ser capaz de ver partículas, el software de seguimiento y análisis sólo funcionaba en un intervalo de concentración definido. No permitía detectar partículas que son demasiado pequeñas o no dispersaban la luz lo suficiente debido a su índice de refracción. Para materiales con bajo índice de refracción, el tamaño mínimo detectable era de 20 a 40 nm. Sin el software de seguimiento y análisis eran visibles con el mismo microscopio sólo partículas mayores a 500 nm.

Los fabricantes de instrumentos ven el cambio a corto y mediano plazo. La nanotecnología se mueve desde el laboratorio de investigación a la industrialización. Aumenta la demanda de equipos no sólo para la caracterización en los procesos de fabricación, sino para hacer frente a la salud, la seguridad y el medio ambiente.

El enfoque a la nanoescala requiere de un cambio en el diseño de los instrumentos. Construir una herramienta para la investigación científica, muy sofisticada, no es lo mismo que adecuarla para el ingeniero de proceso, el ingeniero de producción, el técnico en química o bioquímica. Puede tratarse de un Microscopio de Fuerza Atómica AFM o de un Microscopio con Sonda de Exploración STM o de un Analizador de Partículas, que la forma en que el usuario interactuará con él será muy diferente. Es la visión de KeySight-Agilent[3].

Evolución de Hewlett Packard - Instrumentación

- ***1939**: **HP** inicia como una empresa de Prueba y Mediciones (el ADN de HP).*
- ***01NOV1999**, **AGILENT Technologies** inicia su operación como empresa independiente, con 2 divisiones:*
  1. *Electronic Test & Measurement*
  2. *Life Sciences & Chemical Analysis*

  Agilent encarna el compromiso histórico con la innovación y la contribución, integridad absoluta, trabajo en equipo, confianza y respeto por el individuo.
- ***01NOV2013**, se crea **KEYSIGHT Technologies** para operar a partir de 01NOV2014, enfocada en: Electronic Test & Measurement. Incluirá toda la cartera de productos de medición electrónica.*

  *Ron Nersesian, quien ha sido presidente y director general de operaciones de Agilent, ha sido nombrado director general y presidente de KeySight Technologies **KT**.*

  "Keysight Technologies" refleja el espíritu de nuestra organización innovadora, intuitiva y con visión de futuro", manifiesta Ron Nersesian.

Microscopía con Barrido de Microonda SMM [4]
(Scanning Microwave Microscopy System)

Figura 4

AFM, Scanner con SMM, Nose Cone y PNA[4]

El sistema SMM de KeySight-Agilent N9416S es el único que permite tener dos modos de operación: escanear con microondas y microscopía, combinar las capacidades de mediciones eléctricas de un analizador vectorial de redes VNA (Vector Network Analyzer) y con la resolución espacial de un AFM. El modo de SMM supera todas las técnicas de microscopía de barrido de capacitancia basado en los AFM tradicionales, ofreciendo mucha mayor versatilidad de aplicación, capacidad de adquirir los resultados cuantitativos, con la sensibilidad y rango dinámico más altos de la industria, junto a una resolución espacial y eléctrica excepcionales, permitiendo mediciones de impedancia compleja (resistencia y reactancia), capacitancia calibrada, densidad de dopante calibrado, en estudios de difusión y de topografía a escala nanométrica.

Es muy útil para la prueba de semiconductores. Su caracterización funciona en todos los semiconductores: Si, Ge, III-V (por ejemplo GaAs, InAs, GaN) y II-VI (por ejemplo, CdTe, ZnSe).

Funciona a frecuencias múltiples y variables.

El software de modelado EMPro permite realizar simulaciones electromagnéticas para apoyar la interpretación de los experimentos de escaneo de microondas con microscopía. Complementa el análisis de SMM de forma eficiente y permite investigaciones más detalladas en relación con la geometría 3D de la muestra, el diámetro y el ángulo del eje de la sonda del AFM, así como la frecuencia de medición. El modo de imagen avanzada del SMM con las curvas de retracción de la sonda sobre la muestra también se puede modelar y la muestra puede ser investigada con más detalle con el uso de este software.

El sistema de medición está formado por:

1. Un analizador vectorial de redes para microondas de muy alta precisión PNA-X (Precision Network Analyzer) con opciones configurables para el rango de frecuencias desde 300 kHz hasta 1.1 THz.
2. Un Microscopio AFM de barrido en X y Y; y, Z (para lazo cerrado).
3. Un Cable de precisión para Microonda y un cono de prueba con elementos de calibración.

La operación y rango del sistema depende de las características del material bajo prueba y de las opciones del PNA-X.

Principio de operación:

- El analizador de redes PNA envía una señal RF incidente a la sonda a través de un diplexer.
- La señal RF es reflejada por el material bajo prueba a través de la sonda y medida por el analizador.
- La Magnitud y la Fase de la relación entre la señal incidente y la reflejada es calculada por el PNA.
- Se aplica un modelo para medir las propiedades eléctricas del material bajo prueba.
- El microscopio AFM explora y mueve la sonda a lugares específicos para hacer un nuevo punto de medición.

[1] Nanotechnology Measurements Operations, KeySight-Agilent Latin America Region presentation (David Yee)
[2] 5420 SPM/AFM: KeySight-Agilent (5990-3848EN Rev.B)
[3] www.keysight.com - www.agilent.com
[4] N9416S Scanning Microwave Microscopy System
[5] Advanced Nanomeasurement solutions (5989-6406EN Rev.A)