Corrección y comentarios a los trabajos finales en RFI

1) Autorización de prefijos EID (Leonardo Vidal, Federico Rodríguez-Teja)

En líneas generales el trabajo está muy bien, considerando claramente las limitantes de ser un trabajo para un curso de 3 semanas. Explican los principales problemas de seguridad en la actualidad y los analizan bajo una óptica realista. Las ideas de emplear DANE y TLSAs son razonables, aunque debatibles. Las principales debilidades de este trabajo están vinculadas justamente a la parte más compleja de LISP que es en escenarios de movilidad, especialmente con movilidad agresiva (e.g., smart vehicles). En estos casos el número de handovers implicando cambios de RLOCs e incluso de raomings (cambio a RLOCs remotos) puede ser alto en períodos de tiempo muy cortos, lo cual tendría implicaciones en los ReMAs. Cito a los autores del trabajo:

"En este caso podemos pensar en que el nodo móvil LISP (que incorpora una implementación liviana de xTR) pueda registrar objetos ReMA Mobile que sólo diferirían de los objetos ReMA en que la fecha de inicio en general será muy próxima a la del momento en que se consulte el mismo y la fecha de fin, no muy lejana."

En escenarios que pueden ser claves para LISP, como smart vehicles, el approach propuesto no funcionaría, ya que el cambio de RLOCs debe ser activo, no puede estar atado a un lease temporal.

Otros aspectos del trabajo:

El trabajo no explica en detalle como se resuelve el problema central de la movilidad. La DDT en sí está basada en RPKI, y además LISP-SEC asegura la autenticidad de la mensajería en los map request y map replies, por lo que el sistema de mapping en sí y la mensajería pueden considerarse razonablemente robustos en materia de seguridad. Los problemas más complejos surgen frente a la necesidad de tener que autorizar un update en un Map-Server, que en el caso móvil son los siguientes:

1) Cómo sabe un xTR que un EID alienígena es quien dice ser?
2) Cómo sabe el EID alienígena que ese xTR es quien dice ser?
Estos podrían resolverse mediante RPKI, pero implicaría gestión de criptografía (calve pública/privada) en móviles, lo cual hoy por hoy no está claro que sea deseable a menos que se utilicen mecanismos más livianos que los tradicionales, como por ejemplo elliptic curves.
3) Cómo puede ese xTR generar un update de forma autorizada en un Map-Server remoto? Lo de remoto se ilustra así: tomamos un avión y cambiamos de continente.
Mantenemos el mismo Map-Server o migramos? Si lo mantenemos, "bien", porque no rompemos la jerarquía DDT, pero "mal", porque ahora el proceso de hacer un update en mi home Map-Server implica autorizar entradas desde un ETR alienígena. Además, en la Fig. 1 varios de los mensajes 6, 7, 8 y 9 podrían tener que cruzar océanos en cada resolución. Si migramos, "bien", porque ahora el ETR y el Map-Server pueden formar parte del mismo dominio de trust y usar por ejemplo una clave compartida, pero "mal" porque rompemos la jerarquía DDT….. como se ve, el problema no es trivial.
4) Cómo se asegura el Map-Server que ese xTR desde el que le llega la solicitud de map-register y el EID para el que se quiere autorizar el update están efectivamente "juntos"? … es decir cómo se garantiza el binding?
5) Cómo evitamos MiM attacks?
6) Cómo evitamos ataques coordinados?

2) Optimización del Tráfico Broadcast (Fernando Rodríguez y Juan Vanerio)

Este trabajo lo conocía un poco de antemano dada la discusión que tuvimos antes de la entrega. En líneas generales y viendo ahora el resultado final, tengo poco claro que la solución propuesta sea buena y/o justifique la complejidad añadida. Pedir a una entidad aparte que gestione los broadcasts puede ser complejo y costoso. Como el fin es mantener incambiados los dispositivos finales, el switch debe pasar los broadcasts a la SDN app. La misma debe transformar cada broadcast en un unicast y dirigirlo a la aplicación o entidad que resolverá que hacer con ello. Como la entidad mantiene estado, puede determinar que el broadcast sólo debe fluir por ports concretos. Por ejemplo, en el caso de un ARP habrá un flujo unicast hasta arribar al último switch, el cual debe reconvertirlo a broadcast para no afectar el funcionamiento del dispositivo final.

Para que esto sea efectivo y sobre todo eficiente, la aplicación o entidad encargada de procesar y resolver el tránsito de unicasts[1] debería conocer no sólo la topología Layer 2 sino que probablemente también debería de tener la foto entera del spanning-tree en la red Layer 2. En esquemas de switching multi-vendor (es decir redes conformadas por switches de varias marcas distintas).....estos dos requerimientos más la necesidad de mantener estado de conectividad y ubicación son aspectos no triviales.

La pregunta es: vale la pena poner tal complejidad adicional? .... o la curva costo-beneficio arrojaría que una mejor estrategia es ampliar el parque mediante la compra de switches y/o gateways más baratos, segmentar adecuadamente y rutear entre segmentos?

En resumen, la aplicación de SDNs tiene sentido .... aunque no necesariamente tenga sentido la solución en sí misma. Para ello, hace falta un estudio a fondo en materia de costo-beneficio, lo cual evidentemente escapa el alcance de este curso.

Otros ítems:

En general el documento está plagado de faltas de ortografía, por ejemplo en el apéndice de NETBIOS. Se recomienda mayor cuidado al presentar documentación, porque si ustedes mismos no se preocupan por su trabajo, menos se interesará quien les lea.

[1] ...y en última instancia la conversión el o los broadcasts que toquen en el o los switches treminales.

3) Flexibilización de las Configuraciones (Fernando Rodríguez y Juan Vanerio)

Esta propuesta es más interesante que la anterior acerca del manejo de broadcasts. Ciertamente, la definición de una suerte de meta-lenguaje de configuración es algo más que deseable en networking y las SDNs apuntan a ofrecer gran potencial en esta dirección.

La propuesta está bien estructurada y se han tomado el trabajo de desarrollar con buen nivel de detalle un set de primitivas básico.

Los problemas en general con este tipo de propuestas son:

- La parte declarativa en sí (los ejemplos que muestran los autores son muy Cisco-like o TCL-like), pero entornos heterogéneos exigirán una semántica mucho más neutra, pero al mismo tiempo mucho más rica. Lo cual da lugar al segundo problema….
- … la definición de un modelo de datos con suficiente semántica como para abarcarlo todo…pero sin llegar a ser un monstruo intratable en sí. Por ejemplo, los objetivos de NETCONF son fácilmente justificables, pero sin YANG no prosperará.

En la actualidad hay varias compañías trabajando en esta línea.
Una opción es, configuro usando un meta-lenguaje declarativo y luego compilo/traduzco en un SDK específico para Cisco, para Huawei, etc. Otra opción es, configuro usando el mismo meta-lenguaje declarativo y luego utilizo una API y un protocolo estándar (como ser OpenFlow) para hacer el enforcement de las configuraciones en los routers. Si la compilación/traducción es en línea (por ej. declaro directamente desde consola) o utilizo una API y un standard protocol como OpenFlow => entonces sí tiene sentido la SDN. Por el contrario, si la compilación/traducción es offline, entonces no hace falta una SDN. Puede verse como un lenguaje de programación cualquiera que genere cada configuración particular como salida y luego se conecte al equipo y cargue la configuración.

Otro punto a considerar es el ejemplo de las ACLs. El ejemplo del ANTIVIRUS es gráfico pero no necesariamente trivial en casos más genéricos. El lugar exacto de donde insertar las líneas del bloque y la secuencia de las mismas para lograr el efecto de filtrado deseado puede variar de acuerdo a la semántica y al vendedor del equipo. Parecería que la forma de solucionar esto es manteniendo una meta-configuración común y pasarle la responsabilidad ya sea a la API o a cada proceso de compilación/traducción particular.

4) Filtrado de Capas Superiores en Equipos de Networking (Fernando Rodríguez y Juan Vanerio)

Desde el punto de vista de las SDNs, esta propuesta es probablemente la más indicada de las tres presentadas por Rodríguez y Vanerio. Las SDNs podrían ciertamente dar lugar a estrategias de inspección y filtrado mucho más elaboradas y ambiciosas que las provistas por las técnicas actuales.

El trabajo está bien motivado y ha sido razonablemente bien desarrollado. Tan sólo dos comentarios que vale la pena hacer:

1) Página 5, los autores mencionan:

"Los disectores deben estar compilados para poder ser utilizados, pero para facilitar el manejo, el router debe ofrecer la funcionalidad necesaria para compilar disectores a partir de sus códigos fuentes."

Esto no es ni correcto ni necesario. OpenFlow podría a futuro permitir hacer este tipo de cosas sin necesidad de compilación del lado del router. Todo el proceso de inspección y los nuevos disectores pueden estar embebidos enteramente en una lógica externa al router (el OpenFlow Controller) y luego mediante el protocolo OpenFlow y una interfaz estándar se le indica al router que hacer para cada flujo en particular.

2) Las SDNs son justamente "Software". Ahora bien, mantener el backplane de un switch en frecuencias de conmutación acordes con su capacidad física exige cosas como gran capacidad de procesado software en la SDN. A futuro, no tendrá sentido una solución "SDN" si el software (la app on top) se convierte en cuello de botella y nos baja notoriamente el throughput efectivo respecto a la capacidad del backplane. En este sentido, los disectores e inspecciones más elaboradas en capas superiores podrían comprometer el throughput efectivo del switch. Muy posiblemente, los dispositivos high-end vendrán con soporte (hardware) dedicado, so serán más bien HA-SDNs (Hardware Assisted SDNs). En otras palabras, es muy posible que aplicaciones mucho más ambiciosas de inspección como ser:

- lawful inspection
- parental control
- cybercrime
- ….

… terminen empleando técnicas similares a las mencionadas aquí, pero con la salvedad de ser asistidas mediante hardware dedicado para mantener tanto el throughput efectivo en el switch como contenedores virtuales optimizados y aislados per user/per VLAN/per flow….etc.

5) iPSP (Nicolás Antoniello, Emiliano Gutiérrez y José Restaino)

Lamentablemente esta propuesta está bastante mal, ya que hay varios conceptos que fueron captados erróneamente. Los autores mencionan:

"Se busca que el enrutamiento de datos hacia fuera del Sistema Autónomo (AS) se realice estrictamente de acuerdo a la decisión tomada por el ePSP"*

"Si los routers internos no tienen conocimiento del $ePSG_{ASN}$, puede que envíen el tráfico a un router de borde diferente al que hubiese elegido PSP. Esto resultará en que si se quiere tomar una decisión, de acuerdo con PSP, deberá de ser reenrutado dentro del AS."

Los errores radican en que el PSP no toma decisiones de enrutamiento. Es decir, el PSP no se utiliza para "enrutar" en línea. Tampoco se emplean los PSGs para enrutar paquetes. El PSP y los PSGs se utilizan para gestionar y controlar algunas funciones del enrutamiento de una forma simple, limpia y sin la necesidad de "tocar" al BGP. El que enruta en línea es siempre el BGP. La overlay PSP sirve para:

- Evitar IP prefix hijacking (e.g., integrando ROA inspection en PSP y filtrando updates BGP maliciosos).
- Evitar IP route hijacking (mediante el uso de PSGs validados en PSP y filtrando updates BGP).
- Evitar el path hunting en BGP cuando hay withdrawals (mediante updates driveados y gestionados por el PSP).
- Acelerar la convergencia en BGP (tomando control sobre el update de los grafos y evitando miles de mensajes inútiles en BGP).
- Reducir el churn (ya que al evitar miles de mensajes inútiles se baja claramente el churn).

Ahora bien, para que todo este control y gestión sea posible, se necesita mensajería a nivel de la overlay PSP. iPSP es la parte encargada de distribuir esa mensajería entre entidades overlay en un mismo dominio administrativo. Tanto el ePSP como el iPSP son para control, no para "enrutar" ni "decidir" acerca de routers de borde …. eso ya lo hace el BGP. En particular, no puede haber loops debido a decisiones de encaminamiento del PSP, simplemente porque el PSP no toma tales decisiones. El PSP hace inspección de updates BGP y toma decisiones de control, como ser: discard (por un ataque o al tomar control de la convergencia), pass unmodified (update légitimo y ok), etc.

Tampoco existe necesidad alguna de complicar el iPSP creando iPSGs. Qué ganamos con esto? Qué aportan los iPSGs? La idea detrás de un PSG es capturar las políticas de tránsito entre ASs a través de un tercero (el proveedor de tránsito). Para ello se usan grafos inspirados en los line graphs, donde la relación de grafo AS_i → AS_j → AS_k pasa mediante una transformación de lo anterior a un line graph del tipo (i,j) → (j,k). Pero adentro de un mismo AS no hay políticas de transito de tipo valley-free….so cuál es la necesidad y/o el incentivo para crear iPSGs para "modelar" las relaciones entre routers dentro de un AS? Los RRs? No se podría hacer más simple?

Los PSGs son line graphs modificados y pruneados por las policies. El objetivo final es que cada AS componga los grafos que recibe de sus vecinos y compute un único PSG que refleje una topología a nivel de inteconexión de ASs sujeta a las policies. La razón de ser del iPSP es sincronizar las entidades overlay para que mantengan una vista común y consistente. Hace falta mantener iPSGs para esto? No. Hace falta un iPSP? Sí, porque hace falta mensajería para distribuir los grafos recibidos mediante ePSP y componerlos.

Además, tampoco hace falta una instancia overlay por router.

La idea de usar RRs en iPSP tiene sentido, pero nuevamente no parece haber necesidad de tener que mantener grafos internos para ello…..alcanzaría con definir sesiones iPSP y las reglas de reflexión.

En líneas generales, se observa que los conceptos no fueron captados correctamente.

6) Multi-point BGP sessions for Traffic Engineering (Miguel Barreto)

El trabajo comienza bien, en particular en la necesidad de modificar o adaptar la capa de transporte para soportar sesiones multi-punto. En líneas generales, la propuesta de modificar el TCP es razonable pero mantener las sesiones sobre transporte UDP no. En rigor, habría que crear un protocolo de transporte acorde (sea adaptando el TCP o crenado uno nuevo).

La debilidad de este trabajo radica en los casos de uso. Si bien se ha captado correctamente el concepto de una sesión BGP multi-punto, su ámbito de aplicación y potencial no se ven reflejados en los casos de uso seleccionados. Por ejemplo:

Qué ventaja tienen los balanceos propuestos en las Fig. 6 y 7 frente a lo siguiente?

- Mantener en los anuncios iBGP en AS2 el next-hop (NH) externo (el del router de borde en AS 3)
- Redistribuir y hacer conocer ese NH externo en el IGP TE (por ejemplo en OSPF-TE) vía R2 y R3.
- Como en casi todos los operadores, utilizar túneles MPLS-TE entre R1-R2 y entre R1-R3 y luego balancear carga hacia *d* vía MPLS-TE.

Qué aportan las multi-point BGP sessions frente al esquema tradicional anterior?

En el caso de RRs: qué ventaja tienen las multi-point BGP sessions frente a soluciones alternativas como BGP add paths? …. Es decir simplemente extender BGP de forma que en lugar de anunciar sólo su mejor ruta, BGP pueda anunciar otras además de ésta.

Otros ítems:

La definición de route leak no es correcta y no es culpa del autor, hay una gran confusión en cuanto a lo que se denomina route leak, incluso entre expertos en el área. El autor menciona:

"El problema de Route Leak se produce cuando un AS anuncia a sus vecinos prefijos que no son suyos, es decir, que no les fueron asignados a él y que no son de sus Customers, sino que son de otro AS [nanog 49]."

En rigor lo anterior no es un leak, es IP prefix hijacking. Esto puede deberse o bien a un hurto (acción malintencionada) o bien a un error de configuración, pero en cualquier caso surge de una "mentira" (intencionada o no, pero mentira al fin). Es decir, debe forjarse un BGP update con información ilegítima. Un leak por el contrario surge cuando en una cadena de anuncios BGP legítimos, se envía un update a un AS rompiendo la cadena valley-free. Un leak no parte de una "mentira" ni de forjar mensajes BGP ilegítimos….surge de hacer leacking de rutas a través de enalces a los cuales esos updates nunca deberían de haber llegado. Por ejemplo, un AS anuncia rutas aprendidas desde un proveedor a sus otros proveedores. En este caso, el AS crea un valle, y para ello alcanza con hacer un leak, es decir no hay necesidad ni de mentir ni de forjar mensajes BGP inexistentes …. alcanza sólo con permitir el tránsito de los updates (alcanza con dejarlos pasar).

La confusión generada por la definición de route leaks más el desconocimiento acerca de ROA han hecho plantear el caso de uso en la sección 6.1. No hacen falta multi-point BGP sessions para esto, ROA ya resuelve el IP prefix hijacking y de una forma mucho más efectiva basada en RPKI. De hecho, ROA está siendo desplegado y testeado. El ejemplo que se plantea luego usando las Figs. 13 y 14 y esa especie de "broadcast" en las sesiones multi-punto a ver quien es dueño del prefijo X es innecesario bajo un esquema ROA. Además, para resultar efectivo el alcance del broadcast podría tener que llegar a ser a toda la Internet, además de tener que venir soportado por una infraestructura RPKI para evitar ataques mediante falsas respuestas a esos "boradcasts". La propuesta en sí es como una mala versión de ROA.

Otros errores:

En la sección 7, el autor menciona:

"La idea es que al llegar tráfico seteado con cierta comunidad (previamente acordada) dicho tráfico sea inyectado en la sesión p2mp correspondiente según el criterio utilizado por AS0."

El tráfico nunca será "inyectado" en la sesión p2mp. La sesión p2mp es a nivel de plano de control y es para intercambio de información de routing. El tráfico será inyectado y fluirá por los data paths que toque.

Quizás sea un problema de cómo se explican los casos de uso, pero tampoco queda claro el rol de las sesiones multi-punto como base para la diferenciación del tráfico y sobre todo, la ventaja de las sesiones multi-punto frente a otras alternativas más tradicionales como las usadas ahora mismo para CDNs. Por ejemplo, desde el punto de vista del "negocio", Google (AS2 en la Fig. 15) es un problema para empresas como Telefónica (AS0 en la Fig. 15). Pero desde el punto de vista "tecnológico", no queda clara la ventaja comparativa de las sesiones multi-punto frente al manejo y distribución de tráfico mediante las soluciones ya existentes.

En resumen, se ha captado bien el concepto de las sesiones BGP multi-punto, pero no se ha podido encontrar y desarrollar un caso de uso que realmente muestre el potencial que tienen.

7) Route Leaks: State of the Art and Proposed Solutions (Álvaro Valdés y Eduardo Cota)

Este es el mejor de los trabajos propuestos y con diferencia, no sólo por el nivel de detalle y rigurosidad en el análisis de los problemas abordados sino también por el intento de proponer soluciones a los mismos.

Nits:

- Pág. 5: *"Debo chequear que las rutas recibidas de peers o clientes vengan de color verde, y que todo el camino sea verde (de lo contrario estoy ante un route-leak)."*
    - De acuerdo a las explicaciones dadas en el texto las rutas recibidas de los peers ya deberían de venir amarillas, no verdes.
- Pag. 6: Tabla AV1?
- Faltan captions en las figuras.
- El ejemplo de partial transit en la Figs. 11 y 12 no está claro, está mal redactado y parece haber errores en la figura 11. Por ejemplo:
    - p) f = 0 => F = 11, no debería de ser f = 1 => F = 11?
      Y además parece haber leaks….la intención está clara pero el ejemplo en sí no.
- La Fig. 13 está mal. En celeste…lo que firma AS_R siempre debe hacer referencia al AS_D. El binding de "con quién estoy conectado" siempre debe coincidir con "a quién se lo envío". De esta forma la cadena es irrefutable. El caso en lila está bien, lo que firma AS_P contiene el binding AS_P <-> AS_R, pero los demás (AS_R y AS_D) están mal, ya que falta el binding con el AS next-hop en el pack firmado. Sin el binding en el pack firmado se pueden hacer ataques fácilmente. Lo que un AS firma siempre debe hacer referencia al next AS en el path explícitamente.
- Las propuestas de Overlay on Demand y Path on Demand no tienen sentido. De hecho, lo que tiene sentido es una estrategia similar a la Authenticated Overlay propuesta por lo autores, pero con ideas adicionales, de forma que se permita hacer un binding con los prefijos sin revelar las policies:
    - En línea con el ejemplo de Authenticated Overlay, la idea de construcción del PSG en la que se está trabajando ahora mismo está basada en forward signing en la overlay (similar a su uso en BGPSEC). Y es correcto el análisis de los autores acerca de la utilización de los enlaces "primarios". De hecho el proceso de construcción de la overlay es bastante más complejo de lo visto en el curso.
- Pág. 15: *"En definitiva estos son los fragmentos de información que termina recibiendo el AS 8 para poder validar el armado del camino hacia el AS 5 en su PSG (ver figura 12)."*
    - Fig. 12?
- Los ejemplos de T1, T2 y T3 en la página 15 no son claros.
- El tercer caso, el de la validación criptográfica sin BGPSEC ni PSGs no parce cuadrar o está mal explicado. Por una parte intenta solucionar el problema de route leaks y por otra se basa en atributos "no transitivos". Si yo AS Y recibo un anuncio del AS X, y quiero llevar a cabo acciones malintencionadas, un par de atributos no transitivos no pueden proteger al AS X de mis malas intenciones, porque al hacer mis anuncios a un tercero (AS Z) el histórico simplemente se pierde. Un atributo "no transitivo" me puede a mi AS Y ayudar a no generar misconfigurations tal vez, pero frente a ataques intencionados no.