Diseño y Operación de Redes Telemáticas

Tema 3. Algoritmos y Protocolos de Nivel de Red

Ramón Agüero Calvo

Departamento de Ingeniería de Comunicaciones

Índice

1 Encaminamiento

2 Routers: scheduling

3 Software Defined Networking y OpenFlow

4 Information-Centric Networking

Índice

1 Encaminamiento

2 Routers: scheduling

3 Software Defined Networking y OpenFlow

4 Information-Centric Networking

Encaminamiento: introducción

- El encaminamiento se puede clasificar en función del número de destinos de la información
  - **Unicast.** Para comunicaciones entre dos nodos
  - **Broadcast.** Comunicaciones que se envían a todos los nodos de la red.
  - **Multicast.** En este caso, la información se dirige a un grupo de destinos.
- Conceptos teóricos
  - **Teoría de grafos.** Herramienta matemática sobre la que se sustentan los problemas de encaminamiento (y otros)
  - **Modelo del nodo de comunicaciones.** Se quiere encontrar la ruta con mejores prestaciones, se necesita conocer el retardo que introducen los nodos y enlaces de la red

Encaminamiento: teoría de grafos

- La teoría de grafos $\mathcal{G}(N, V)$ se utiliza para resolver los problemas subyacentes al encaminamiento (y otros) sobre redes
- En la mayoría de los casos se trata de resolver un problema de optimización
- Encaminamiento: el objetivo es encontrar la ruta de coste mínimo entre un origen y el resto de nodos de la red

$$\textbf{min} \quad \sum_{\forall(i,j)\in E} c_{i,j} x_{i,j}$$

$$\textbf{s.t.} \quad \sum_{\forall(i,k)\in E} x_{i,k} - \sum_{\forall(k,i) in E} x_{k,i} = \begin{cases} -1 & i = N - \{S\} \\ N-1 & i = S \end{cases}$$

$$x_{i,j} = \{0, 1\}$$

Encaminamiento: teoría de grafos

- El problema anterior es lineal y existen algoritmos eficientes que los resuelven
- Existen otros problemas, conocidos como $\mathcal{NP}-$completos, para los que no se han entrado algoritmos eficientes
  - En estos casos, se suelen utilizar técnicas heurísticas para solucionarlos
  - Uno de los problemas más conocidos es el del viajante: *Travelling Salesman Problem* (TSP)
- Otra limitación de la formulación anterior es que el coste de enviar una unidad de flujo por un enlace es constante
- Los retardos de los nodos de comunicaciones dependen, en realidad, de su carga

Cálculo del retardo: modelo MM1

- Para modelar el retardo de un paquete al atravesar un nodo de comunicaciones se utiliza la teoría del teletráfico
- Se define ρ como el tráfico (o grado de ocupación) de un nodo, calculado como el producto de la tasa de llegadas λ y el tiempo de servicio T_S
- En el modelo más sencillo se asume que los paquetes llegan según un proceso de *Poisson* (tiempo entre llegadas exponencial negativo) y que el tiempo de servicio es también exponencial negativo

$$T_T = \frac{T_S}{1-\rho} \qquad T_Q = T_T - T_S = \frac{T_S \cdot \rho}{1-\rho}$$

Cálculo del retardo: modelo MG1

- La principal limitación del anterior modelo es que asume que el tiempo de servicio es exponencial negativo, el modelo MG1 se puede utilizar para obtener valores más precisos

$$T_Q = \frac{T_S \cdot \rho}{1-\rho} \frac{1 + C(T_S)^2}{2}$$

$C(T_S)$ es el coeficiente de dispersión del tiempo de servicio:
$C(T_S) = \frac{\sigma_{T_S}}{T_S}$

- El tiempo total sería la suma del tiempo de espera más el tiempo de servicio

Encaminamiento unicast

- El encaminamiento unicast se basa en encontrar una ruta entre un origen y un destino
- Los dos algoritmos que más relevancia tienen en este ámbito son *Bellman-Ford* y *Dijkstra*
- A partir de ellos aparecen los dos esquemas de encaminamiento unicast más importantes

Distance-Vector

- Basado en *Bellman-Ford*
- Convergencia lenta
- Protocolo RIP

Link-State

- Basado en *Dijkstra*
- Mayor sobrecarga
- Protocolo OSPF

Encaminamiento unicast

- La estrategia *link-state* ha ido suplantando paulatinamente a *distance-vector*
- Sin embargo, ninguna de ellas es lo suficientemente escalable para las redes actuales
- Se establece una jerarquía en la red, que se estructura en sistemas autónomos (*Autonomous Systems, AS*)
- Un AS es una red que 'pertenece' (es gestionada) por una organización
- Dentro del AS se utilizan los esquemas de encaminamiento anteriores (principalmente *link-state*)
- Para habilitar las comunicaciones entre diferentes AS se emplea el protocolo *Border Gateway Protocol* (BGP)

Encaminamiento unicast: BGP

- Retos
  - Escalabilidad
  - Privacidad: para evitar 'mostrar' la topología interna y los acuerdos con otros AS
  - Políticas: el coste de los enlaces no tiene una interpretación homogénea, se necesita controlar los puntos por los que atraviesan el tráfico
- BGP no puede basarse únicamente en el *shortest-path*, ya que implica la definición de un coste común, incompatible con relaciones de negocio
- ¿Cuál es el punto de partida adecuado?
  - Link-state: se necesita la topología completa de la red: grandes necesidades de almacenamiento y sobrecarga
  - Distance-vector: se oculta la topología, tomando decisiones en base al *siguiente salto*

Encaminamiento unicast: BGP

- El funcionamiento básico de BGP se basa en un esquema *distance-vector*, teniendo que superar las siguientes limitaciones
  - No se puede plantear estrictamente como un problema de minimización (el tipo de distancia no es homogéneo)
  - Tiene que ser capaz de superar los problemas de convergencia
- Para ello se pasa del *distance-vector* al *path-vector*, en el que se anuncia el camino completo, en lugar de la métrica
- Se anuncian los prefijos *Classless InterDomain Routing* (CIDR) y se implementan políticas de *Path Selection* y *Path Export*

Encaminamiento broadcast

- N-way unicast: utilizar N veces un algoritmo unicast
- Flooding no controlado
- Flooding controlado a través de números de secuencia, empleado por *Gnutella*
- Reverse Path Forwarding (RPF)
  - Los paquetes solo se reenvían si llegan a través de la interfaz utilizada por la ruta de menor coste al origen
- Búsqueda del *minimum spanning tree*

Encaminamiento multicast

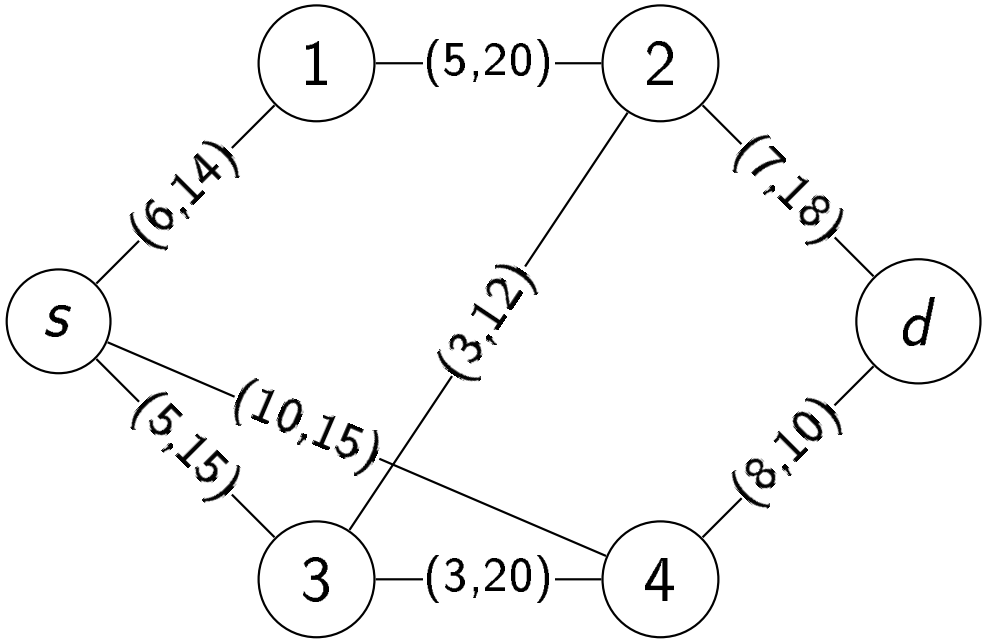
- Un reto que aparece en este tipo de encaminamiento es el de la identificación de los nodos que tienen que recibir los paquetes y su direccionamiento
  - ¿Cómo se forma el grupo multicast?
  - ¿Cómo se actualiza?
- En arquitecturas TCP/IP se emplea el protocolo *Internet Group Management Protocol* (IGMP)
- El objetivo es establecer un árbol que cubra todos los routers a los que se conecten nodos del grupo multicast
  - Un único árbol para todas las fuentes, a partir de un nodo central: **Group-Shared Tree**, similar a un spanning-tree
  - Un árbol para cada fuente, utilizando una versión modificada del RPF con mensajes de *pruning*

Constraint-Based Routing

- La selección del camino o ruta se hace en base a *constraints* o políticas de QoS (retardo, ancho de banda)
- La tecnología *Multi-Protocol Label Switching* (MPLS), que se basa en el encaminamiento en base a etiquetas, ha sido uno de los catalizadores del *Traffic Engineering*
  - Permite cambiar la operación del protocolo de encaminamiento: control de admisión, scheduling
- El *Constraint-Based Routing* no reserva capacidades, solo establece rutas
- En función de los criterios de las políticas (retardo, fiabilidad, capacidad) y de si se pretende satisfacer un umbral u optimizar el criterio, aparecen numerosas alternativas, con diferentes niveles de complejidad

Constraint-Based Routing

- Ejemplo CBR; cada enlace tiene una tupla (*coste*,*retardo*)



  - **min** *coste*, *retardo* ≤ 50: $\boxed{s \rightarrow 3 \rightarrow 2 \rightarrow d}$ **(15, 45)**
  - *coste* ≤ 18, *retardo* ≤ 40: $\boxed{s \rightarrow 4 \rightarrow d}$ **(18, 25)**

Índice

1 Encaminamiento

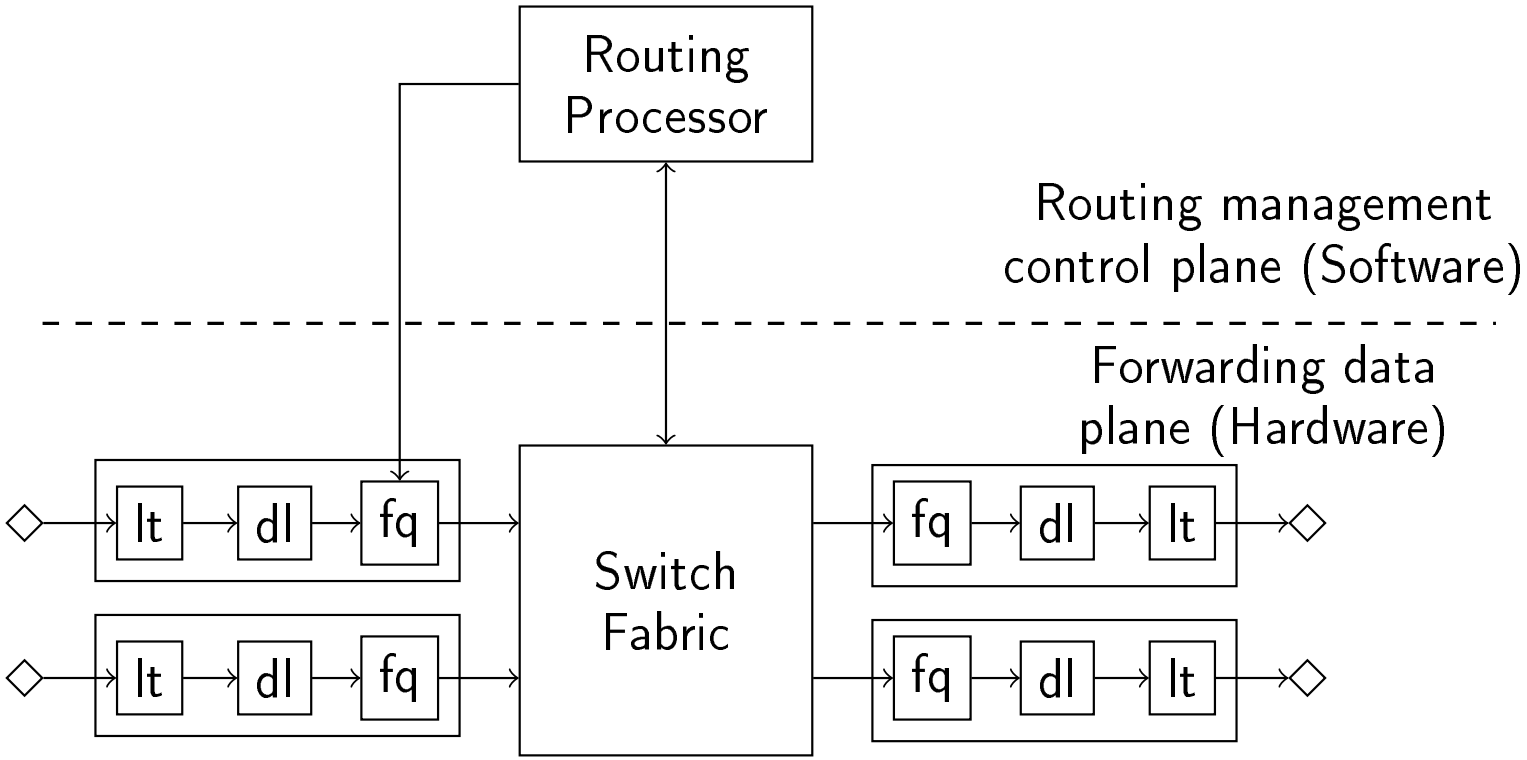
2 Routers: scheduling

3 Software Defined Networking y OpenFlow

4 Information-Centric Networking

Estructura: puertos de entrada/salida

- Estructura genérica de un router



Estructura: puertos de entrada/salida

- Estructura (inversa en cada caso)
  - Terminación de línea
  - Procesado del enlace de datos
  - Búsqueda, forwarding y buffer
- La espera en un router suele estar asociada a los puertos de salida, pero si la conmutación es lenta se podrían producir esperas en los puertos de entrada: bloqueo *Head-Of-Line* (HOL)
- El tamaño de los buffers es finito, por lo que en situaciones (saturación) puede ser necesario tirar paquetes

Estructura: switching y routing

Switching

- Es la parte más importante del router, se encarga de conectar los puertos de entrada y salida
  - Memoria: como los dispositivos IO de sistemas operativos tradicionales, mediante el uso de interrupciones
  - Bus compartido
  - Crossbar, para evitar el problema de la limitación en *ancho de banda* de la anterior alternativa

Routing

- Cuenta con los protocolos y las tablas de encaminamiento
- Implementado en *software*

Gestión de los puertos de salida

- Planificación (scheduling)
  - FCFS (*First-Come-First-Served*)
  - WFQ (*Weight-Fair-Queueing*), que permite distribuir los recursos de manera *justa* en base a los flujos extremo a extremo
- Eliminación de paquetes (drop)
  - *Drop-Tail*, descarta el último paquete recibido
  - Gestión dinámica de las colas (*Active Queue Management*): se descartan paquetes antes de que se llegue a saturar el router: *Random Early Detection* (RED)
- Dimensionado de la capacidad del buffer
  - Producto entre la capacidad del enlace y el RTT ($C \cdot RTT$)
  - Con N flujos TCP activos, $\frac{C \cdot RTT}{\sqrt{N}}$

Fair Queuing

- Una planificación *FCFS* puede ser muy perjudicial, especialmente si hay flujos heterogéneos en la red
  - *Trenes de paquetes* con aplicaciones con gran demanda, que pueden acaparar toda la capacidad
- Solución inicial: *Round-Robin* entre flujos, con una cola por flujo
  - Dependencia con el tamaño de los paquetes
- Max-Min Fairness
  - Se satisfacen las peticiones de los flujos que requieran menos de su *cuota justa*
  - Se reparten los recursos que sobran entre el resto de flujos

Fair Queuing

- Algoritmo Max-Min Fairness: MaxMin-Fairness(C, N, r_i)
  1. Calcular $\frac{C}{N}$
  2. while haya flujos en los que $r_i < \frac{C}{N}$

$$C = C - \sum_{\forall i | r_i < \frac{C}{N}} r_i \quad ; \quad N = N - \sum_{\forall i | r_i < \frac{C}{N}} 1$$

  3. $f = \frac{C}{N}$

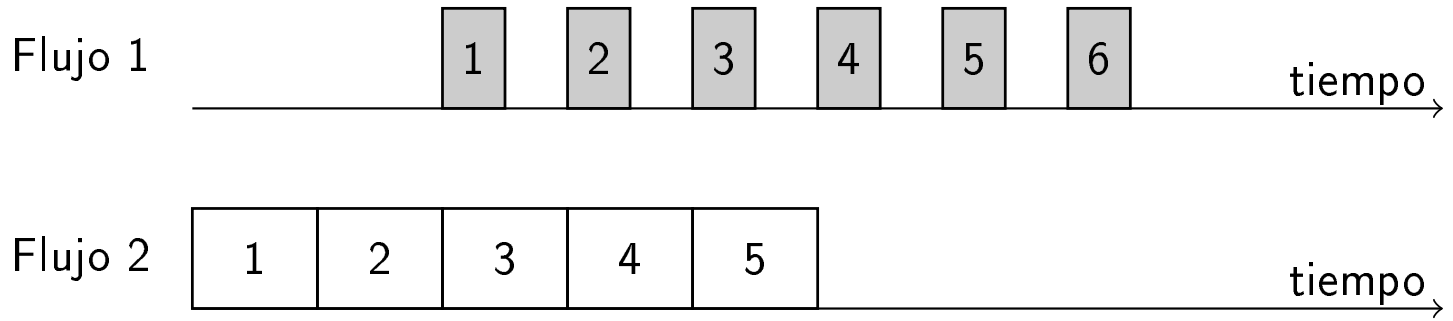
  – Ejemplo: $C = 10$, $r = \{8, 6, 2\}$; OUT $= \{4, 4, 2\}$

- **Weighted Fair Queuing.** Se asignan pesos diferentes a cada flujo: w_i, con los que se calcularía la tasa ofrecida

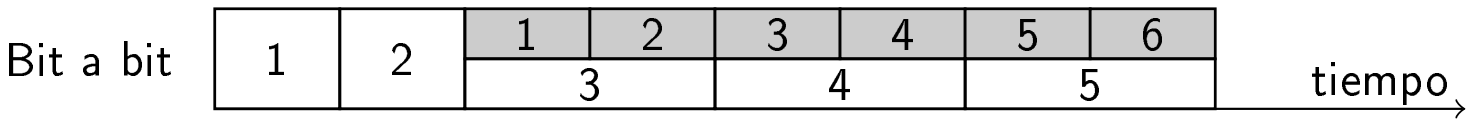
$$c_i = \frac{w_i \cdot C}{\sum_{\forall k} w_k}$$

Fair Queuing

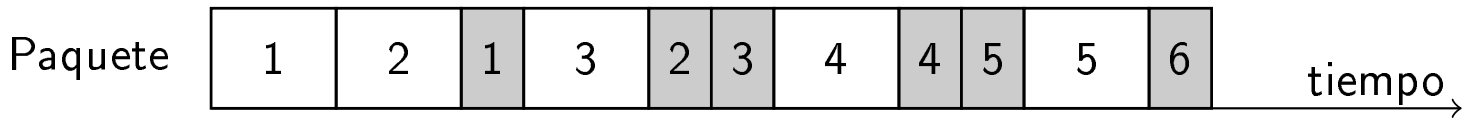
- Ejemplo con dos flujos



- Modelo de fluido, en los flujos se 'atienden' bit a bit
  - Se asume el uso de un *Generalized Processor Sharing* (GPS)



- Utilizando el tiempo en el que acaba la transmisión de los paquetes



Random Early Detection

- En lugar de esperar hasta la saturación, se descartan los paquetes con cierta probabilidad $\mathbb{P}_{\text{drop}}$ cuando la longitud de la cola supere cierto umbral
- Cálculo de la longitud media (δ) con una EWMA (*Exponentially Weighted Moving Average*) a partir de la ocupación *instantánea* del buffer a la llegada de un paquete q

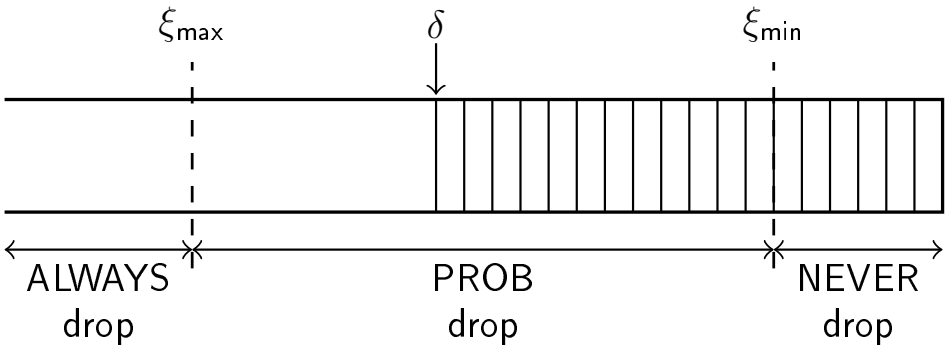
$$\delta_t = (1 - \omega) \cdot \delta_{t-1} + \omega \cdot q$$

- Tras la llegada de un paquete...
  - Si $\delta < \xi_{\text{min}}$, el paquete se encola
  - Si $\delta > \xi_{\text{max}}$, el paquete se descarta
  - Si $\xi_{\text{min}} < \delta < \xi_{\text{max}}$, el paquete se descarta con cierta probabilidad $\mathbb{P}_{\text{drop}}$

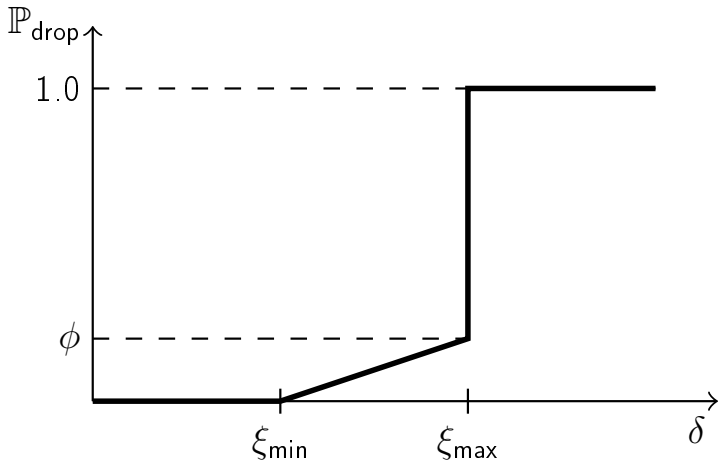
$$\mathbb{P}_{\text{drop}} = \phi \cdot \frac{\delta - \xi_{\text{min}}}{\xi_{\text{max}} - \xi_{\text{min}}}$$

Random Early Detection

- Representación gráfica del RED



- Probabilidad de descartar un paquete entrante en función de δ



Índice

1 Encaminamiento

2 Routers: scheduling

3 Software Defined Networking y OpenFlow

4 Information-Centric Networking

Software Defined Networking: motivación

Visión tradicional de las redes

- Equipos propietarios: variedad de dispositivos (switch, router, middleboxes); modos de configuración diferentes
- El software interno es cerrado
- Las soluciones/arquitecturas de gestión tradicionales (SNMP) no son suficientes

Inconvenientes

- Ralentización de la innovación
- Incremento de la complejidad
- Mayores CAPEX y OPEX

Software Defined Networking: introducción

- El Sotware Defined Networking (SDN) busca cambiar la manera en la que se diseñan y se gestionan las redes
  - Separa los planos de control y datos
  - Fomenta el uso de un único programa software que controla varios elementos del data plane, a través de una API (*Application Programming Interface*) común
- Se puede decir que realmente revisita ideas del pasado, la separación de los planos de datos y control ya lo realizaba la PSTN hace más de 20 años
- La aparición del protocolo *OpenFlow* en 2008 puede verse como un catalizador de esta propuesta
- El principal promotor de *OpenFlow* expone los principales elementos del SDN en `http://video.mit.edu/watch/tr10-software-defined-networking-541/`

Software Defined Networking: evolución

- La filosofía del SDN no surge realmente en los últimos años, sino que se han ido realizando propuestas que han sentado sus bases
- Se distinguen tres fases[§] en la evolución al SDN
  - Active Networks [1995-2000], en las que se introducen las funciones programables
  - Separación de los planos de control y datos [2001-2007], desarrollo de interfaces abiertas entre los planos de datos y control
  - OpenFlow y Sistemas Operativos de Red [2007-2010], primer ejemplo de despliegue masivo de estas tecnologías

[§] *Nick Feamster, Jennifer Rexford y Ellen Zegura. "The Road to SDN: An Intellectual History of Programmable Networks". En: SIGCOMM Comput. Commun. Rev. 44.2 (abr. de 2014), págs. 87-98*

Active Networking

- Esta primera etapa coincide con la eclosión de la Internet
- Las aplicaciones que tradicionalmente se habían venido usando (email/ftp) no son suficientes
- Dos alternativas para analizar las diferentes propuestas
  - Pruebas en entorno de laboratorio (pequeña escala)
  - Simulación de topologías más complejas
- Cuando las propuestas reciben interés y financiación se inicia el proceso de estandarización, que es lento
- La alternativa es la programación de los nodos de red
  - Encapsulando el código en paquete de datos, lo que recibe el nombre de **Active Networking**
    Se puede destacar la propuesta contemporánea del Cognitive Radio, que presenta varios aspectos comunes
  - Paquetes fuera de banda

Active Networking

Tecnology Push

- Reducción de los costes de computación
- Aparición de lenguajes de programación avanzados
- Virtualización de máquinas, lo que proporcionaba una mayor protección
- Financiación

Use Pull

- Dificultad y rigidez a la hora de desplegar nuevos servicios: **Osificación**
- Aparición de multitud de nuevos *middleboxes*, necesidad de un control más genérico

Active Networking

Principales contribuciones

- Funciones programables para favorecer la innovación
- Virtualización de red, envío de paquetes a aplicaciones en función de las cabeceras
- Arquitectura unificada para la orquestación de *middleboxes*

Separación de los planos de datos y control

- Alrededor del año 2000, se produce un incremento notable del volumen de tráfico
- Aparece la necesidad de ofrecer una mayor fiabilidad y rendimiento
- Los operadores se plantean estrategias de gestión mejoradas, como el control de las rutas (*traffic engineering*)
- Los protocolos de encaminamiento tradicionales no eran adecuados, pero se trabaja con un objetivo menos ambicioso, proponiendo soluciones pragmáticas y de plazo corto
- Uno de las principales limitaciones era la estrecha integración de los planos de datos y control en los routers y switches

Separación de los planos de datos y control

Technology Push

- Funciones de lógica del forwarding de paquetes implementadas en hardware
- Incremento del tamaño y alcance de las redes, demanda de mayor fiabilidad y nuevos servicios
- Mejora de las plataformas de computación

Separación de los planos de datos y control

Use Pull

- Se centran en las necesidades de las redes y los operadores, no en el usuario final
- Programabilidad del plano de control de la red en global (no de dispositivos individuales)
- Selección de rutas basadas en carga, control sobre los flujos de tráfico, rechazar tráfico sospechoso

Separación de los planos de datos y control

Principales contribuciones

- Control centralizado de la red
- Interfaz abierta entre los planos de datos y control
- Gestión distribuida, pasando del despliegue de controladores de *backup* a arquitecturas en las que cada controlador se 'encargaba' de una parte de la red

OpenFlow y Sistemas Operativos de Red

- Interés de la comunidad científica en la experimentación de red a gran escala
  - En Estados Unidos: GENI (proyecto NSF) y Clean Slate Program (Standford), para campus de universidades
  - En Europa: EU FIRE
- Dos visiones en el camino hacia el SDN: redes completamente programables y despliegue/desarrollo más pragmático
- *OpenFlow* se sitúa como una solución que balancea las dos visiones
  - Habilita un número mayor de funciones que los controladores tradicionales
  - Se basa en los switch/routers existentes, lo que le proporciona un gran pragmatismo

OpenFlow y Sistemas Operativos de Red

- A la aparición de la interfaz *OpenFlow* le siguen rápidamente el desarrollo de controladores, como NOX, lo que permite la creación de múltiples aplicaciones de control
- Un switch *OpenFlow* consta de una tabla de reglas de gestión de paquetes
  - *Patrón*, en función de de los bits de la cabecera del paquete
  - *Acción* a realizar para el paquete
  - *Contadores*
  - *Prioridad*

OpenFlow y Sistemas Operativos de Red

Technology Push

- Los fabricantes de chipsets deciden abrir sus interfaces para modificar las tareas de *forwarding*
- Aprovechando la aparición de dichos componentes, aparecen compañías que se dedican a desarrollar conmutadores
- *OpenFlow* se basa inicialmente en las funcionalidades de los dispositivos existentes, con lo que se favorece su rápida adopción (actualización del firmware)

OpenFlow y Sistemas Operativos de Red

Use Pull

- Surge en redes de campus universitarios (Standford), en los que los investigadores pretendían experimentar soluciones *clean-slate* sobre plataformas reales
- Posteriormente se empieza a expander a otros ámbitos, como los datacenters, donde la necesidad de gestionar grandes volúmenes de trafico
  - Es más rentable desarrollar controladores genéricos que adquirir hardware propietario

OpenFlow y Sistemas Operativos de Red

Principales contribuciones

- Generalización de funciones y dispositivos de red
  - *OpenFlow* permite establecer reglas con hasta 13 cabeceras de paquetes
  - Se generaliza la instalación de reglas, pudiendo llevarse a cabo en función de la aplicación
  - Carece de soporte a funciones del plano de datos avanzadas, por lo que no es posible implementar cualquier función de middlebox

OpenFlow y Sistemas Operativos de Red

Principales contribuciones

- Sistema Operativo de Red, que separa la operación de red en tres capas
  - Un plano de datos con una interfaz abierta
  - Una capa de gestión del estado de la red, manteniéndolo consistente
  - Una lógica de control, que toma acciones en función del estado de la red
- Técnicas de gestión distribuidas
  - Para favorecer la escalabilidad y fiabilidad es necesario tener varias entidades que acten como un único controlador centralizado

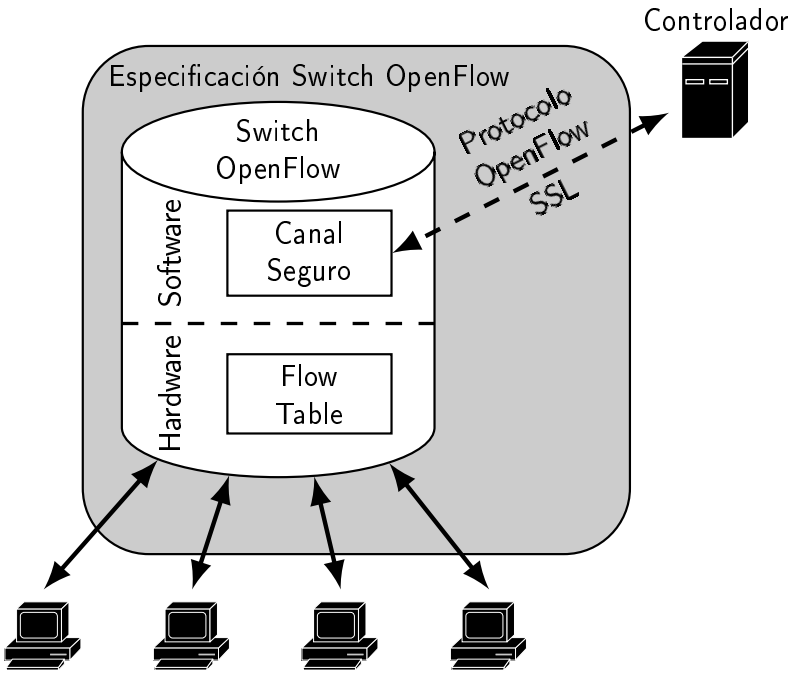
El Switch OpenFlow

- La idea es aprovechar el conjunto de funciones comunes que existen en varios switches y routers
- *OpenFlow* proporciona un protocolo abierto que permite modificar la tabla de flujos en diferentes dispositivos de red, permitiendo aislar diferentes tipos de tráfico (investigación) en la red

El Switch OpenFlow

Estructura de un Switch OpenFlow

- Una tabla de flujos, con una acción asociada a cada entrada
- Un canal seguro que conecta el switch con el controlador
- El protocolo *OpenFlow*, que emplea el controlador para comunicarse con el switch



El Switch OpenFlow

Acciones básicas

- Reenviar los paquetes de un flujo a un puerto de salida determinado
- Encapsular y reenviar los paquetes al controlador
  - Se suele hacer con el primer paquete, para establecer una nueva entrada en la tabla
  - Podría emplearse para enviar todos los paquetes al controlador para ser procesados
- Descartar los paquetes del flujo
- Procesar el paquete según el funcionamiento tradicional del switch

Índice

1 Encaminamiento

2 Routers: scheduling

3 Software Defined Networking y OpenFlow

4 Information-Centric Networking

Content Delivery Networking

Comunicación Cliente/Servidor tradicional

- Alta sobrecarga en el servidor
- Pobre rendimiento: elevados retardos y bajo throughput
- Un único punto de fallo
- Escasa protección frente a ataques *Denial-of-Service* (DoS)
- Vulnerabilidad frente a situaciones *flashcrowd/slashdot effect*

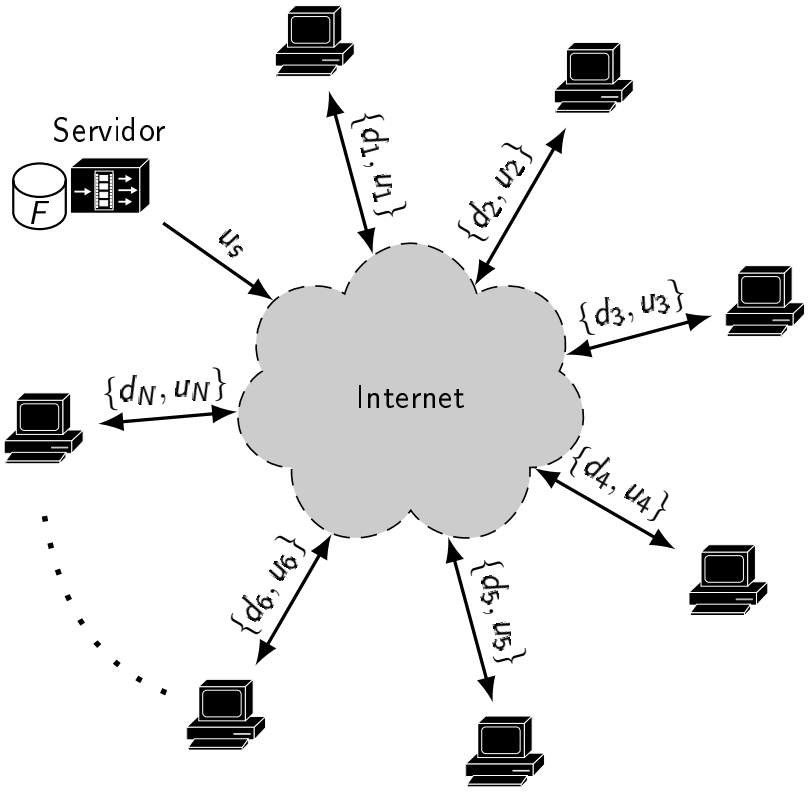
Content Delivery Networking

Posibles soluciones

- Acercar el servidor web al usuario
  - Menor retardo y mayor throughput
  - El dimensionado de recursos no es tan relevante (balanceo)
  - Se 'soportan' picos de carga
  - ¿Cómo seleccionar el servidor?
- Acercar la aplicación
  - La base de datos se convierte en el cuello de botella

Peer-to-peer

- Se tiene un servidor y se pretende transmitir un fichero a N usuarios
- Se supone que las velocidades de subida y bajada son u_s y d_s para el servidor y u_i, d_i para cada uno de los clientes
- Solución tradicional
  - El servidor manda el fichero *individualmente* a cada cliente
- Solución p2p
  - Cada cliente redistribuye una parte del fichero



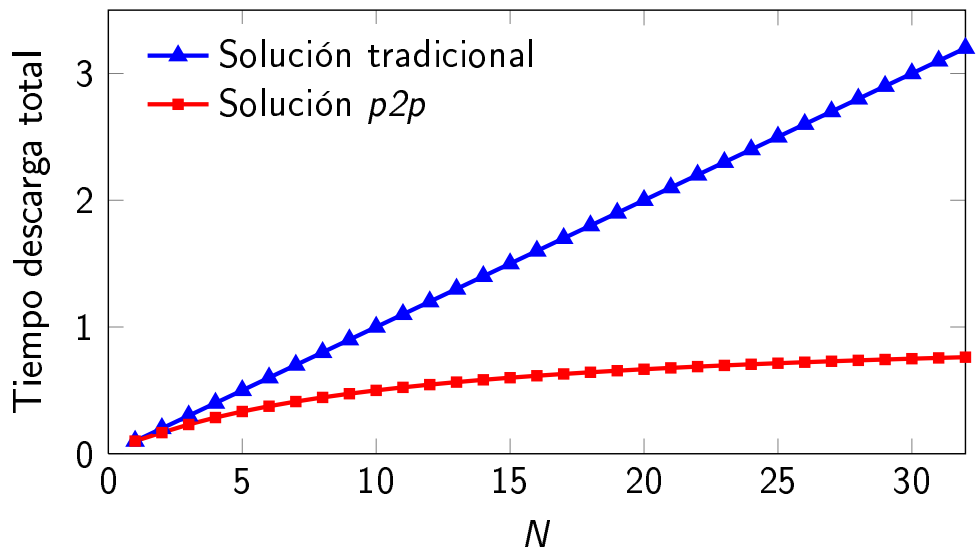
Peer-to-peer

- Retardo solución tradicional

$$D_{\text{trad}} \geq \max\left\{\frac{F \cdot N}{u_s}, \frac{F}{\min_i d_i}\right\}$$

- Retardo p2p

$$D_{\text{p2p}} \geq \max\left\{\frac{F}{u_s}, \frac{F}{\min_i d_i}, \frac{F \cdot N}{u_s + \sum_{\forall i} u_i}\right\}$$



Peer-to-peer

- Las soluciones p2p se emplean en el nivel de aplicación (aplicaciones torrent, por ejemplo)
- Establecen una arquitectura completa y proporcionan ciertos niveles de seguridad
  - Se establece una red *overlay*
- Se basan en el uso de *Distributed Hash Tables* (DHTs)
  - La llave de la base de datos es el contenido y el valor el localizador (dirección IP) de los nodos
  - A la hora de asignar llaves se tiene en cuenta un identificador del nodo, para que la búsqueda sea más eficiente

Information-Centric Networking

- Una nueva filosofía, enmarcada en lo que se conoce como el Future Internet, para superar las limitaciones del *host-centric* para distribuir contenido
- Se basa en los *Named Data Objects* (NDOs): los receptores solicitan NDOs y los transmisores los 'publican'
- Presenta una arquitectura más apropiada para acceder a y distribuir contenido
- Principales ejemplos
  - Content-Centric Networking (CCN), propuesto por Van-Jacobson, que se basa en OSPF
  - Network of Information (NetInf), iniciativa europea (proyectos 4WARD y SAIL)

Information-Centric Networking: arquitectura

- In-network storage: caching
- Comunicaciones múltiples: réplicas del contenido (acceso simultáneo) a las que acceden los clientes
- Se desacoplan los transmisores de los receptores
- Distribución de contenido fiable: soporta desconexiones puntuales (*Delay-Tolerant Networking*, DTN) y situaciones *flash-crowd*
- Se adapta mejor a escenarios con movilidad y multi-homing (dispositivos con varias tecnologías)

Information-Centric Networking: NDOs

- Puede tratarse de cualquier elemento de información: página web, documento, película, música, etc
- Es independiente de la localización, método de almacenamiento y el transporte (o aplicación que lo requiere)
- Las copias existentes (cache) son indistinguibles
- Diferentes niveles de granularidad: un NDO puede ser una parte de un objeto o el objeto completo
- Se pueden distinguir varias representaciones de un mismo objeto
  - Uso de versiones
  - Diferentes codificaciones (mayor o menor calidad)
- Integración de diferentes meta-datos: autor, fecha, etc

Information-Centric Networking

Naming y seguridad

- Se requieren nombres unívocos
- Es obligatorio establecer mecanismos de verificación del nexo entre el nombre y el contenido
- Alternativas de naming
  - Jerárquica: similar a las url, lo que favorece el encaminamiento
  - Plana: es auto-verificable, sin que sea necesaria una infraestructura de seguridad pública (PKI), el nombre se establece con una función *hash* del contenido

Information-Centric Networking

Interfaz con aplicaciones: API

- Los transmisores *publican* la disponibilidad de un NDO
- Los clientes se *suscriben* a NDOs y son avisados cuando estén disponibles

Encaminamiento

- Se utiliza un *Name Resolution System* (NRS), que realiza un *binding* del contenido a los localizadores (direcciones IP) de las máquinas donde se guarda (1 o varias)
- El cliente acude al NRS, para después comunicarse con las fuentes (podría utilizar encaminamiento IP); finalmente, la(s) fuente(s) manda(n) el contenido al cliente