Unidad 6: Nivel de Red

Internetworking (IP)

Al querer conectar redes de distinto tipo nos encontramos con dos problemas fundamentales:

- *heterogeneidad:* ambas redes pueden ser de distinto tipo, así como también las redes que se tiene que atravesar para llegar de una a otra.
- *escala:* cómo encontrar eficientemente un camino desde un punto a otro, y qué convención de direcciones usar

Al hacer esto se construye una red "lógica" entre redes. A diferencia de las redes switcheadas, que tenían limitaciones en cuando a cobertura y cantidad de hosts, se querrá que estas *internets* escalen arbitrariamente. Los nodos que conectan cada una de las subredes se llaman **routers**.

De acá en adelante se describe el funcionamiento de IP, la tecnología de internetworking detrás de Internet.

Modelo de servicio

El conjunto de servicios ofrecidos por la internetwork debe ser lo suficientemente laxo para que pueda implementarse sobre todos los tipos de redes que la componen y para un amplio abanico de casos de uso. Los protocolos montados sobre IP deben ser concientes de las garantías faltantes y compensarlas cuando lo requieran.

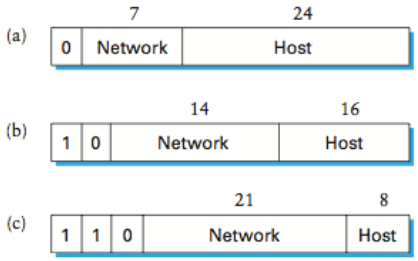
IP define tanto el **esquema de direccionamiento** como la forma de **enviar mensajes**, con las siguientes características:

- **datagramas**: no es orientado a conexión
- **best effort**: los paquetes pueden perderse, reordenarse, repetirse o llegar tarde

Direccionamiento global

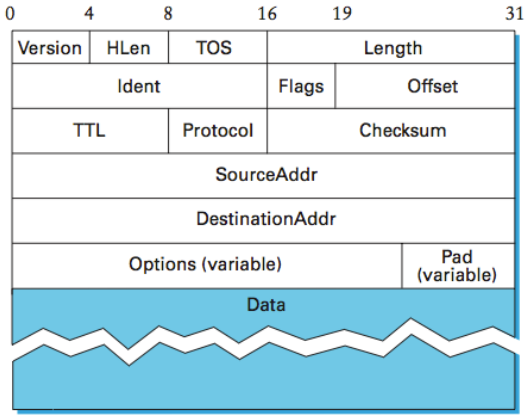
Dado que se quieren conectar redes que no tienen por qué conocerse inicialmente, es necesario asignar direcciones únicas a cada host. A su vez, es deseable que las direcciones tengan una estructura jerárquica que facilite el routeo (cosa que no tienen las direcciones Ethernet).

- Las direcciones IP tienen dos partes: una indica la red y otra identifica un host dentro de esa red.
- El espacio asignado a cada parte depende de la clase de dirección (ver figura).



Envío de datagramas

Formato de paquete



- Version: versión del protocolo utilizado
- HLen: longitud del header, variable cuando se incluyen opciones
- TOS: flags para modificar el envío
- Length: cantidad de bytes del datagrama completo
- Ident, Flags, Offset: datos sobre fragmentación (ver más adelante)
- TTL: cantidad de hops antes de ser descartado
- Protocol: demultiplexación de protocolos superiores (ejemplo: TCP y UDP)
- Checksum: control de integridad
- SourceAddr: utilizado para que el destinatario pueda o no aceptar el datagrama
- DestinationAddr: para permitir routeo por cada datagrama

Fragmentación y rearmado de paquetes

IP puede funcionar sobre redes con distintas limitaciones en tanto al tamaño máximo de paquetes. Definir un tamaño tan pequeño como para viajar por cualquier soporte es poco conveniente, sin contar que es imposible conocer ese tamaño a priori. Por esto, IP tiene un mecanismo para desarmar paquetes cuando no pueden viajar por una red, y rearmarlos cuando es necesario.

- Se llama **MTU** al máximo tamaño de datagrama IP que puede transmitirse en una red
- Cuando un router debe enviar un datagrama por una red que tiene un MTU menor al del datagrama, lo fragmenta
- El campo *ident* del header indica que dos fragmentos forman parte del mismo datagrama original
- El campo *offset* indica el número de fragmento

- El flag *M* indica que hay al menos un fragmento con mayor offset
- El rearmado se realiza en el destinatario del mensaje, no en cada router
- Si en el rearmado falta un fragmento, se descartan todos los de ese ident

Forwarding

- Los routers tienen varias interfaces, conectadas a distintas redes.
- Cada interfaz tiene una dirección distinta.
- Cuando un host o router debe enviar un datagrama se fija si pertenece a la misma red física (o una de ellas, en caso de routers).
- Si pertenece a la misma red, lo envía según corresponda para esa red.
- Si no, el datagrama debe reenviarse a un (otro) router (*next hop router*).
- El next hop se determina con una **tabla de routeo** (ver figura), y con un default en caso que no esté presente.

NetworkNum	NextHop
1	R3
2	R1
3	Interface 1
4	Interface 0

Observar cómo se utiliza agrupamiento jerárquico para reducir la cantidad de información requerida en cada router y mejorar la escalabilidad.

ARP (Address Resolution Protocol)

Caundo se tiene un datagrama que pertenece a la misma red, se envía utilizando el enlace de esa red. Para hacer esto, se debe conocer la dirección del host dentro del esquema de direccionamiento utilizado por la misma. Hace falta, entonces, un mecanismo para traducir direcciones IP a direcciones físicas dentro de una red.

ARP es un protocolo que permite construir dinámicamente las tablas utilizadas para estos mapeos. Funciona de la siguiente forma:

- El host que desea enviar el datagrama chequea si tiene un mapeo para la dirección IP destino
- Si no es así, realiza un broadcast con una *ARP query* que contiene la dirección IP a resolver, aparte de la dirección IP y física del emisor
- Si un host recibe la query y ya posee una entrada en su tabla para el emisor, la actualiza aunque no sea el destino de la query
- Si algún host tiene asignada la dirección IP de la query, responde informando su dirección física y aparte guarda un mapeo para conocer la dirección física del emisor (dado que es probable que pronto tenga que responder algún mensaje de nivel de aplicación).
- En caso de recibir respuesta, el emisor se agrega una nueva entrada a la tabla ARP.

DHCP (Dynamic Host Configuration Protocol)

A diferencia de las direcciones Ethernet (que están grabadas en ROM), las direcciones IP no pueden asignarse de antemano, ya que contienen información variable sobre la estructura de la red.

Aparte de su dirección, un host debe también conocer la dirección del router al cual debe delegar datagramas hacia afuera de la red.

DHCP es un protocolo para automatizar la configuración de los parámetros necesarios para operar en una red IP.

- Se basa en al menos un servidor DHCP
- El servidor puede utilizarse como un directorio centralizado configurado manualmente, pero también se puede encargar de asignar direcciones de un pool dinámicamente.
- Para descubrir el servidor, un host envía un mensaje *DHCPDISCOVER* a la ip 255.255.255.255 (broadcast)
- En caso de haber un servidor, responde al host con la información de configuración necesaria.

Otras consideraciones:

- Se podría querer tener un servidor DHCP que mantenga la información consistente para varias redes. En este caso se utiliza un **relay** en cada red, un host que reenvía los mensajes *DHCPDISCOVER* via unicast al servidor.
- Dado que no se puede depender de que los hosts liberen las direcciones, estas se asignan por un tiempo determinado a partir del cual el servidor puede volver a asignarlas (en caso de que no se hayan renovado).

ICMP (Internet Control Message Protocol)

Si bien IP no brinda grandes garantías sobre el envío de datagramas, viene acompañado del protocolo **ICMP**, que define mensajes para reportar los errores que puedan llegar a surgir (por ejemplo, en caso de que no se pueda alcanzar un host o haya fallado el ensamblado).

VPNs (Virtual Private Networks)

Uno podría querer brindar conectividad entre dos redes, manteniéndolas aisladas de otras. La forma más sencilla de lograr esto es utilizar cableados especiales. Sin embargo, también sería deseable crear circuitos virtuales aislados sobre una red existente. Esto permitiría, por ejemplo, reutilizar la gran infraestructura de Internet para crear redes privadas virtuales de escala global.

Para lograr esto sobre IP se crean **túneles IP**, que funcionan como un enlace punto a punto aunque encapsulen una cantidad arbitraria de redes intermedias.

- Un host en la red 1 envía normalmente un mensaje a una dirección IP interna de la red 2.
- El router de la red 1 (R1) sabe que debe enviar mensajes a direcciones de la red 2 a través del enlace virtual.
- Para enviar un mensaje a través del túnel, R1 encapsula el paquete en un nuevo datagrama IP con la dirección del router destino (R2) y lo envía.
- El datagrama viaja normalmente a través de la internet.
- Cuando R2 recibe el datagrama, ve que contiene su misma dirección, por lo cual inspecciona el contenido y encuentra otro datagrama IP con una dirección interna de su red.

Motivaciones para hacer eso:

- Enlaces seguros (complementado con encripción)
- Utilizar funcionalidad no presente a lo largo de toda la internet pero sí en los extremos (ejemplo: multicast)

- Enviar paquetes de protocolos no-IP sobre una red IP

Intradomain Routing

- **Routing** es el proceso que se lleva a cabo de forma distribuída a lo largo de una serie de redes en los routers para dotarlos de la información que requieren para dirigir paquetes.
- El objetivo final es contruir tablas como la siguiente:

Network Number	NextHop
10	171.69.245.10

- En caso de circuitos virtuales, debe realizarse sólamente en el inicio de la conexión.
- Para comunicación sin conexión (datagramas) el ruteo debe realizarse por cada mensaje.
- El problema consite en encontrar el camino mínimo en un grafo, donde puede cambiar la topología los costos de los ejes.
- Se busca un protocolo mediante el cuál los nodos puedan ir descubriendo dinámicamente estos caminos.

Se habla de **ruteo interno** cuando es dentro del mismo domino/sistema autónomo (dimensiones reducidas), y de **ruteo externo** cuando ocurre entre dominios (necesidad de escala). Hay dos clases principales de protocolos de ruteo interno: **Distance Vector** y **Link State**.

Distance Vector

Idea principal:

- En cada nodo se mantiene un vector con la distancia y siguiente paso hacia el resto de los nodos
- Inicialmente sólo se sabe el costo de llegar a los vecinos
- Los nodos vecinos se intercambian vectores repetidamente y actualizan los suyos en caso de encontrar algún camino mejor
- Ningún nodo tiene la información de toda la red, pero todos tienen información consistente sobre cuál es el mejor camino a tomar para cada caso.

Los mensajes se producen por dos razones:

- Períodicamente (sirve también como keep-alive)
- Si se detecta algún cambio en la topología o error en enlace

Count to infinity

Este algoritmo puede padecer del problema **count to infinity**: si un nodo -*A*- indica a otro -*B*- que conoce un camino, *B* no tiene forma de saber si él mismo forma parte de ese camino. Esto puede causar que, en determinadas condiciones, se tome mucho tiempo hasta darse cuenta que un nodo es inalcanzable. Ejemplo:

- *A*-*B*-*C*, todos los enlaces con el mismo peso
- Cae el enlace entre *A* y *B*
- *B* detecta que su enlace a *A* se cayó (por lo cuál ya no puede tomar el camino de longitud 1)
- Sin embargo, también recibe un camino de parte de *C* de distancia 2, y lo adopta sin saber que ese camino dependía de su enlace directo con *A*.
- En la siguiente actualización, *B* informará a *C* que su camino hacia *A* ahora tiene longitud 2, por lo cuál *C* ahora cree que está a distancia 3 de *A*.

Posibles soluciones:

- Limitar la cantidad máxima de hops para acotar la cuenta hasta infinito (podría traer problemas si la red crece)
- **Split horizon**: No se informa a al vecino las rutas que se aprendieron de él mismo (funciona sólo para loops entre dos nodos)
- Demorar las actualizaciones para evitar las condiciones de carrera (demora la convergencia)

Ejemplo: RIP

- **RIP** es un protocolo que implementa el algoritmo de distance vector de forma bastante directa.
- Los vectores indican direcciones desde el nodo hacia redes conocidas (no hacia routers)
- Asume que todos los enlaces tienen el mismo peso.
- Se utiliza un máximo de hops para evitar conteo al infinito. Esto limita la aplicación de RIP a internets chicas.

Link State

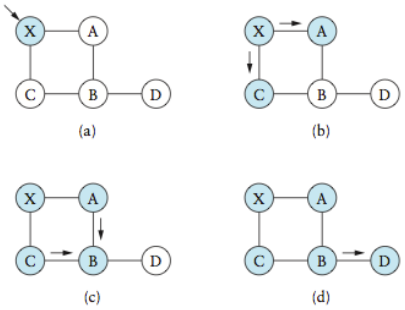
Los algoritmos de **link state** se basan en los iguiente:

- Cada nodo sabe llegar a sus vecinos, y disemina este conocimiento (*link state*) en toda la red
- Acumulando el *link state* del resto, todos los nodos pueden calcular rutas a los otros.

Observar que cada nodo transmite sólamente su *link state* y la reconstrucción de la topología se basa en acumular estos *link states*. En los algoritmos distance vector, por otra parte, cada nodo resumía todo el conocimiento que tenía y se iban propagando estos "resúmenes".

El mecanismo por el cual todos los nodos conocen el link state del resto se llama **Reliable Flooding**:

- Períodicamente o en caso de cambios de la topología, cada nodo envía a sus vecinos un paquete con su *link state*, número de secuencia y TTL.
- Cuando el vecino lo recibe, verifica mediante el número de secuencia si es más reciente de la que ya tiene.
- Si es así, reenvía a todas las interfaces (excepto aquella por la cual llegó el paquete) la nueva información.
- Para asegurarse de que eventualmente los paquetes viejos se descartan, se utiliza un TTL
- En caso de caída de un nodo, éste arranca con número de secuencia 0. En caso de que haya información vieja todavía sin caducar, eventualmente llegará al nodo y éste podrá aumentar su número de secuencia como sea necesario.



Aquí el término *reliable* significa que se tiene la última información y se descartan las versiones contradictorias. Una vez se tiene distrbuída la información de *link state*, cada nodo puede reconstruir toda la

topología por sí mismo y calcular los caminos mínimos. Esto se hace mediante una variación del algoritmo de Dijkstra (*forward search algorithm*).

- La idea principal es inicar con el nodo actual e ir explorando utilizando BFS el grafo actualizando la tabla de resultado cuando se encuentra un camino con menor costo que el ya conocido.
- A nivel implementativo, se utiliza una lista de *caminos confirmados* y otra de *caminos tentativos*.
- La nueva información se actualiza siempre en la lista de tentantivo.
- Al final de cada paso, se promueve a confirmado el dato de menor costo de los tentativos.

Estos algoritmos se estabilizan rápido y no generan demasiado tráfico, además de responder rápidamente a cambios en la topología o falla de nodos. Sin embargo, tienen la desventaja de que se debe almacenar una gran cantidad de informción en cada nodo (lo cuál atenta contra la escalabilidad de la red).

Áreas

- Para permitir escalar la dimensión dentro de un dominio, se puede subdibidir un dominio en **áreas**.
- La información de floodeo se mantiene dentro del área
- Hay siempre un área especial denominada *backbone* (dentro del dominio, distinto a SAs backbones de Internet)
- Cada área tiene un router frontera expuesto al área backbone.
- Los routers frontera se intercambian información sumarizada de cómo alcanzar redes de su área.
- Todo el tráfico pasa entonces por el backbone.

Ejemplo: OSPF

OSPF (Open Shortest Path First) es un protocolo de tipo link state que agrega lo siguiente:

- autenticación: para evitar que hosts mal configurados anuncien distancia cero a todos lados y se conviertan en un "agujero negro"
- jerarquización: permite partir un dominio en áreas, de modo que entre áreas la cantidad de información que viaja es menor. un router sólo debe saber cómo dirigir un paquete al área que corresponde.
- balanceo de carga: permite distribuir carga cuando se cuenta con varias rutas del mismo costo.

Además, permite priorizar ciertas rutas asignando pesos a los enlaces para determinado tipo de tráfico (utilizando un campo de type of service en los paquetes de configuración).

Tipos de mensajes:

- *Hello* (discovery / heartbeat)
- *Link state update*
- *Link state ack*
- *Database description*
- *Link state request*

Internet

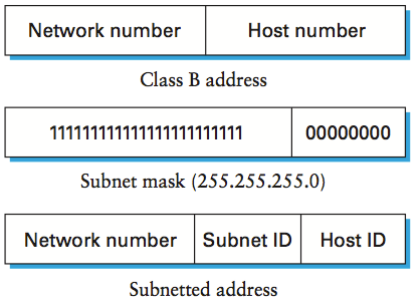
La implementación de una internet global trajo dos grandes problemas de escala: **routeo** (minimizar la cantidad de información que hace falta distribuir para poder routear) y **direccionamiento** (asegurarse de que no se consuma por completo el espacio de direcciones).

Subnetting

El esquema de tres clases de direcciones IP le asigna un número a cada red. Esto provoca un uso ineficiente del espacio de direcciones: se puede tener una red de 2 hosts que use una dirección clase C (~255 direcciones) o una red de poco más de 255 host que use una dirección de clase B (~64K direcciones). A su vez, asignar una gran cantidad de números de red complica el routeo, ya que implica almacenar y transmitir más información entre routers.

Una solución a esto es realizar **subnetting**, asignar todas las IP dentro de un número de red a varias redes físicas. Asumiendo que las redes están cerca geográficamente (para no perjudicar el routeo) este esquema permite aprovechar más el espacio de direcciones.

Para poder compartir una dirección de red entre varias redes físicas, todos los hosts se configuran con una **máscara de subred**. Esto agrega un nuevo nivel de jerarquía a la dirección IP. La porción de la dirección que escape a la máscara será la que identifique al host:



Cuando un host quiere enviar un paquete a otro, utiliza la máscara para ver si ambos pertenecen a la misma subred (es decir, si el "and" binario de ambas contra la máscara arroja el mismo resultado). Si es así, puede enviarse dentro de la subred. En caso contrario, se envía al router.

Notar que utilizando subnetting cambia el mecanismo de forwarding de los routers: ahora deben conocer cada subred y su máscara en vez de números individuales de red. Cuando llega un paquete, se busca la subred que concuerde con la aplicación de la máscara a la dirección de destino.

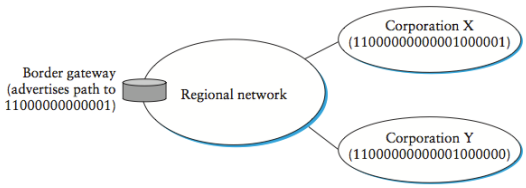
SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

Classless routing (CIDR - supernetting)

En vez de asignar una dirección de clase B para redes no tan grandes, se pueden asignar varias de clase C para aprovechar más eficientemente es espacio de direcciones. Sin embargo, implica mantener información de routeo para cada una de esas redes. Una alternativa para solucionar este problema sería otorgar varias redes tipo C con direcciones **consecutivas**. De esta forma, se puede direccionar la cantidad exacta deseada de equipos con una sóla entrada en cada tabla de routeo.

A este enfoque se lo denomina **supernetting**:

- En subnetting se divide un único número de red para que sea utilizado en varias redes físicas.
- En supernetting se agrupan varios números de red para direccionarlo de forma única.



De forma similar a como ocurría con subnetting, será necesario tener un sistema de routeo que utilize direcciones con prefijo de red variable.

Interdomain Routing

Un **sistema autónomo** (SA) es una unidad administrada independientemente del resto de Internet. Organizar la red en SAs permite:

- desacoplar la administración técnica de casa SA (permitiendo que dentro de cada uno se use un mecanismo de intradomain routing)
- reducir la cantidad de información necesaria para routear, a través de la organización jerárquica y el uso de rutas default.

En Internet se implementaron dos protocolos de interdomain routing:

- EGP (Exterior Gateway Protocol): no utilizado actualmente, limitado a topologías arbóreas con un único backbone.
- BGP (Border Gateway Protocol):

BGP

- Internet está orgizada de forma compleja con varios backbones desde los cuales se organizan ISP de distintos niveles
- BGP entiende a Internet como un grafo de SAs para no imponer una topología específica
- Se llama **tráfico local** al que se origina y termina dentro de un mismo SA
- Se llama **trafico en tránsito** al que sól pasa a través de un SA

Clasificación de SAs:

- *Stub AS*: tiene conexión a un sólo SA. Sólamente se encargará de tráfico local.
- *Multihomed AS*: tiene conexión a más de un SA, pero no lleva tráfico en tránsito.
- *Transit AS*: tiene conexión a más de un SA, y está diseñado para llevar tráfico loca y en tránsito (ejemplo: backbones)

Características:

- Debido a la complejidad del problema, no se busca necesariamente una ruta óptima, sino que alcanza con encontrar alguna.
- Cada SA tiene al menos un *speaker*.
- Los speakers de distintas SAs se intercambian información sobre qué puntos pueden alcanzar y cómo.
- A diferencia de los algoritmos de distance vector y link state, los SAs se intercambian rutas enteras (secuencias de SAs para llegar a una red).
- Se configuran políticas para elegir qué rutas conviene elegir y cuáles publicar.
- Tener rutas enteras también permite evitar ciclos.

Combinando BGP con intradomain routing

- BGP indica cómo hacen los speakers de los SAs para intercambiarse información de routeo
- Los speakers deben tener un mecanismo para que estos datos sean conocidos por los routers internos.
- En general se inyectan los datos utilizando el mecanismo de routing intradomain que corresponda, y configurando adecuadamente las rutas default.
- En casos como los backbones, se utiliza una variante "interna" de BGP (IBGP) para que cada router sepa cuál es el mejor punto de salida para una ruta, y routing intranetwork regular para que sepan cómo llegar a esos routers.

Routeo Multicast

- Multicast implementado en hardware para redes locales (ethernets, token rings)
- Protocolos para permitir multicast en redes globales
- Motivación: enviar un paquete a dirección multicast (IPs clase D) y que la red lo entregue a todos los host suscriptos

Link state multicast

- Se extiende link state normal agregando información sobre qué grupos tienen miembros
- Un host anuncia periódicamente a qué grupos pertenece
- Con esta información se computan los **multicast trees** (árboles con los caminos mínimos a cada host del grupo desde un nodo fuente)

Distance vector multicast

Mecanismo de **Reverse Path Broadcast**:

- Cada vez que llega un paquete desde una fuente S, forwardearlo a todos los puertos si el paquete llegó del shortest-path a S (esto evita ciclos y floodea la red)
- Para cada source, se designa un router parent para cada LAN (el que está más cercano), y sólo éste mete el paquete en la red (para evitar que entre por muchos lados)

Sobre el broadcast descrito anteriormente, se prunean las redes donde no hay miembros del grupo multicast. Esto se conoce como **Reverse Path Multicast**

- Se identifican los nodos hoja (redes con un único router)
- El router conectado a una hoja puede saber si no hay miembros (mediante los anuncios de los hosts), y evitar forwardearlo
- Cada router propaga hacia la internet cuáles son los grupos en los que está interesada la red
- Al final de la propagación, cada router sabe qué grupos hay con suscriptores en cada link

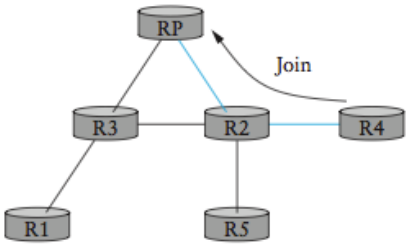
Dado el overhead de agregar esta información en los routers (los grupos activos) en la práctica, se utiliza broadcast hasta que se activa una dirección multicast y ahí se hace el pruneo. De todos modos, esto tiene problemas de escalabilidad.

Prototol Independent Multicast (PIM)

Los algoritmos anteriores no se comportan bien cuando son muy pocos los routers interesados en recibir información de ciertos grupos. **PIM** es un protocolo que tiene dos modos de funcionamiento, *sparse mode* y *dense mode*.

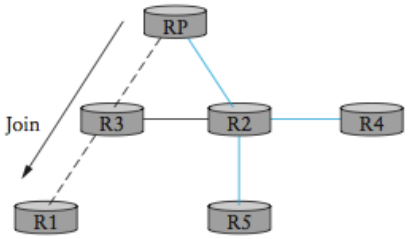
Características del sparse mode:

- Los nodos acuerdan entre sí designar a uno como **rendezvous point** (RP) de cada grupo.
- Se envían mensajes de *join* y de *prune* al RP utilizando unicast.
- A medida que los mensajes de join viajan hacia el RP, cada router anota en su tabla que cuando llegue un paquete para el grupo se debe forwardear por el puerto de donde provino el join (estado **<*, G>**).
- De esta forma, cada join agrega una rama al **shared tree**.
- Para enviar mensajes al grupo, se envía el mensaje mediante un túnel al RP (encapsulándolo en un datagrama unicast), y el mismo lo distribuye en el árbol.

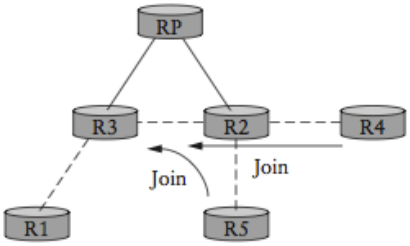


Optimizaciones:

- Si se observa gran cantidad de tráfico en un tunel hacia el RP, el RP puede enviar al sender un sender-specific join (agregando estado **<S, G>** a lo largo del camino), de modo que *para ese sender* el árbol comartido tendrá distinta raíz, y se evitará el overhead de encapsulamiento.



- Dado que el camino no es óptimo, cuando se observa mucho tráfico en una ruta mejorable se puede reemplazar todo el árbol compartido por un **source specific tree**. Para esto, cada router destino envía un source specific join al origien (agregando estado **<S,G>** en los routers del camino) de modo que para ese sender no hará falta pasar por la raíz.



Propiedades:

- se dice que es *protocol independant* porque puede utilizarse sobre cualquiera sea el mecanismo de ruteo unicast.
- el uso de shared trees aumenta la escalabilidad reduciendo la cantidad de estado que hay que tener en cada router (proporcional a la cantidad de grupos, y no al producto de cantidad de grupos por senders)
- cuando se requiere mayor eficiencia, se utilizan source specific trees (trade off entre escalabilidad y optimalidad)

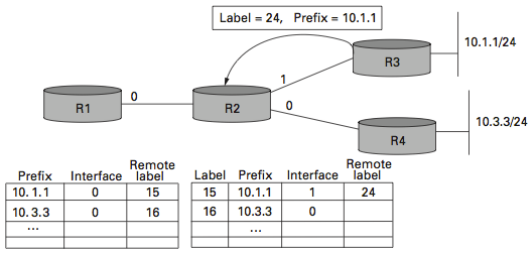
MPLS (Multiprotocol Label Switching)

MPLS es una tecnología que combina algunas características de los circuitos virtuales con la robustez y flexibilidad de los datagramas. A grandes rasgos, consiste en attachear *antes del header IP* una serie de tags a los datagramas IP, que pueden ser utilizados para tomar decisiones a lo largo del routeo. Hoy se usa para:

- Utilizar funcionalidades de IP en dispositivos que no soportan el forwardeo de datagramas IP tradicional.
- Para forzar el routeo a través de rutas explícitas que pueden no ser las normales definidas por routeo IP.
- Para proveer algunas funcionalidades de VPN

Uso: destination based routing

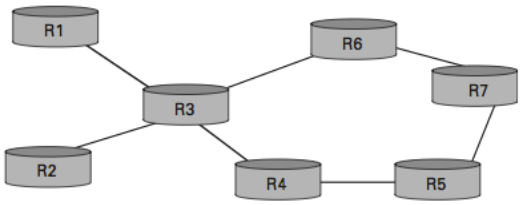
MPL define el protocolo **LDP** (Label Distribution Protocol) con con cuál un router comunica a sus adyacentes que desea recibir una etiqueta específica para paquetes de determinada red. De esta forma, se puede tener indexada la tabla de routeo por etiqueta. Así, al recibir un paquete no deberá hacer una búsqueda del mayor prefijo para conocer la ruta, sino que alcanzará por acceder a la entrada que matchee (de forma exacta) la etiqueta.



Este mecanismo permite implementer el mismo algoritmo de routeo (cualquiera sea) con algoritmos de forwarding más sencillos. Esto posibiltó hacer compatibles con IP switches de otros tipos de redes (ATM) sólo con actualizaciones de software.

Uso: routing explícito

Supongamos que en la imagen siguiente se quiere que el tráfico proveniente de R1 a R7 vaya por la rama superior, y en proveniente de R2 por la rama inferior. Usando routeo tradicional sería imposible, ya que el mismo no tiene en cuenta el origen de los paquetes.



Suponiendo que se tienen las tablas de routeo MPLS configuradas adecuadamente, se podría lograr haciendo que R1 y R2 agreguen distintas etiquetas. Sin embargo, la distribución de etiquetas descrita anteriormente (*LDP*) no permite configurarlas de este modo, ya que sirve para emular el routeo tradicional de IP.

MPLS describe otro protocolo, ***RSVP*** (Resource Reservation Protocol) para lograr este objetivo. Consiste en enviar mensajes de configuración a lo largo del camino deseado que crean las entradas requeridas para el forwarding.

Esto es útil para **traffic engineering**, donde se busca asegurar que habrá suficientes recursos a lo largo de la red. Aparte de definir los caminos a tomar para partes del tráfico, se pueden precalcular rutas que eviten ciertos nodos, y utilizarlas (agregando la etiqueta que corresponda) para evadir nodos caídos sin tener que esperar a que ejecuten nuevamente los algoritmos distribuídos de routeo (**fast rerouting**).

USO: VPNs y túneles

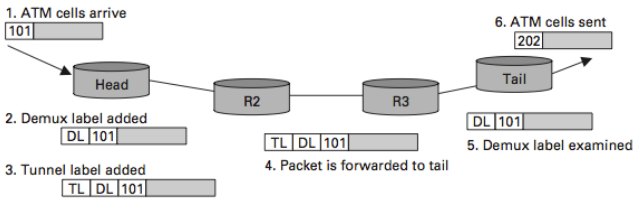
MPLS puede ser utilizado para construir túneles, que a su vez son un mecanismo útil par armar VPNs.

Layer 2 VPN

Una de las formas es armando una **layer 2 VPN**, que se basa en túneles que transmiten datos de capa 2 (frames ethernet o células ATM, por ejemplo) a través de una serie de routers. Notar que si el routeo se basa solamente en los labels, no hace falta que se transfiera contenido IP.

Ejemplo:

- Se quiere enviar tráfico no IP entre R1 y R2 (separados)
- R1 agrega un label de demultiplexación (para que R2 sepa cómo tratar el contenido)
- R1 agrega un label de routeo (como se explicó antes en destination based routing)
- La red transfiere el contenido utilizando los labels, sin necesidad de inspeccionar el contenido (¡que no es un datagrama IP!).
- R2 desencapsula el contenido: remueve el label de routeo y en base al label de demultiplexación sabe introducir el contenido en su red.



Observar que esta funcionalidad también puede lograrse usando túneles IP tradicionales, aunque MPLS tiene la ventaja de que ahorra ancho de banda debido a su header pequeño.

Layer 3 VPN

Una **layer 3 VPN** es otro tipo de VPN creada utilizando túneles MPLS. Es decir, routean utilizando labels que encapsulan los mensajes. La diferencia es que en este caso los mensajes encapsulados sí son datagramas IP.

La implementación es compleja, pero se utiliza para definir redes IP "vitualmente privadas". Es decir, redes IP basadas sobre internet donde los equipos están aislados y manejan su propio espacio de direcciones.