

QoS: IntServ

Area de Ingeniería Telemática
<http://www.tlm.unavarra.es>

Grado en Ingeniería en Tecnologías de
Telecomunicación, 3º

Temas de teoría

1. Introducción
2. QoS
3. Encaminamiento dinámico en redes IP
4. Tecnologías móviles
5. Otros temas

Objetivos

- Conocer los mecanismos implementados por la arquitectura IntServ

Propuestas del IETF

- **DiffServ** (Differentiated Services)
 - Filosofía: priorización de tráfico
 - El usuario o un primer equipo de red marca los paquetes con un determinado nivel de prioridad
 - Los routers van agregando las demandas de los usuarios y propagándolas por el trayecto
 - Esto le da al usuario una confianza razonable de conseguir la QoS solicitada
- **IntServ** (Integrated Services)
 - Filosofía: reserva de recursos
 - Cada router del trayecto ha de tomar nota y efectuar la reserva solicitada
- Pueden coexistir

IntServ: Características

- RFC 1633 : “Integrated Services in the Internet Architecture : an Overview”
- Para cada flujo (puede ser agregado) reserva recursos en todo el camino
- Soporta control de admisión, llevada a cabo en cada salto
- Orientado a conexión
- Requiere un protocolo de señalización que soporten todos los routers
- No requiere modificar los protocolos existentes
- IntServ no ofrece mecanismos explícitos de CAC
- Clasificación por flujo empleando las direcciones IP, puertos y protocolo (5 valores)

IntServ: Servicios

Best Effort (análogo al *Default PHB DiffServ*)

Controlled load service

- RFC 2211 : “Specification of the Controlled-Load Network Element Service”
- Para aplicaciones elásticas con requisitos de ancho de banda
- Análogo a *PHB AF DiffServ*
- “*commitment ... to provide ... service closely equivalent to unloaded best-effort*”
- Prácticamente sin pérdidas
- No da garantías estrictas

Guaranteed service

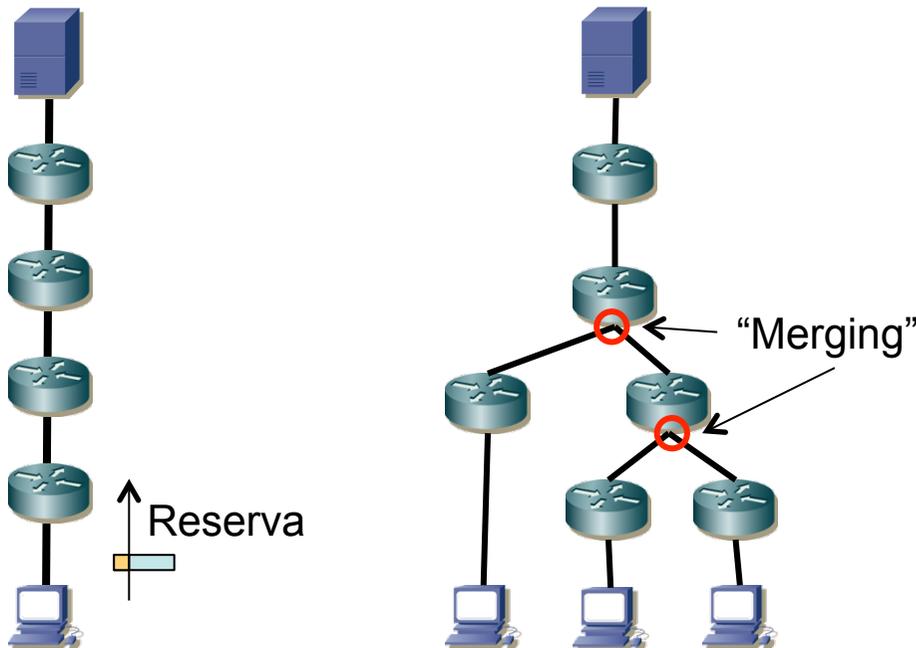
- RFC 2212 : “Specification of Guaranteed Quality of Service”
- Para aplicaciones inelásticas
- Análogo a *PHB EF DiffServ*
- “*provides firm (mathematically provable) bounds on end-to-end datagram queueing delays.*”
- Garantías de BW
- Retardo acotado
- Sin pérdidas en buffers

IntServ: *Signaling*

- Requisitos
 - Debe poderse usar en redes IP
 - Emplear tablas de rutas existentes
 - Reaccionar ante cambios de rutas
 - Soportar multicast
 - Flujos que se agregan en árbol
 - Pequeña sobrecarga
 - Pocos mensajes y pequeños
 - Modular y fácil de extender
- Múltiples protocolos o sistemas de gestión se podrían plantear para ofrecer este servicio
- Resultado:
 - RFC 2205 : “Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification”
 - Es una solución, otras serían posibles pero no se han dado
 - *Soft state (periodic updates)*
 - No sirve para calcular el camino (no es un *routing protocol*)

RSVP

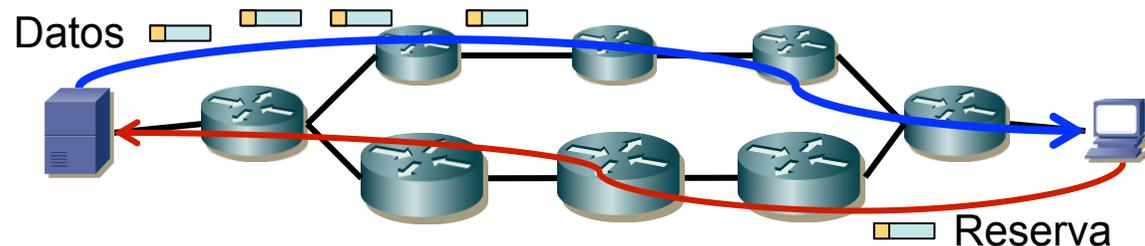
- Directamente sobre IP (protocolo 46)
- Reservas unidireccionales: bidireccional requiere dos reservas
- Soporte de multicast: puede agregar reservas en el sentido hacia la fuente del árbol de distribución multicast
- Las reservas las inicia el receptor del flujo lo cual ayuda en esa agregación al ascender en el árbol



Versión	Header Length	TOS	Longitud		
16-bit identifier			D	M	13-bit fragmentation offset
	F	F			
TTL	46	Header checksum			
Dirección IP origen					
Dirección IP destino					
[opciones]					
Mensaje RSVP					

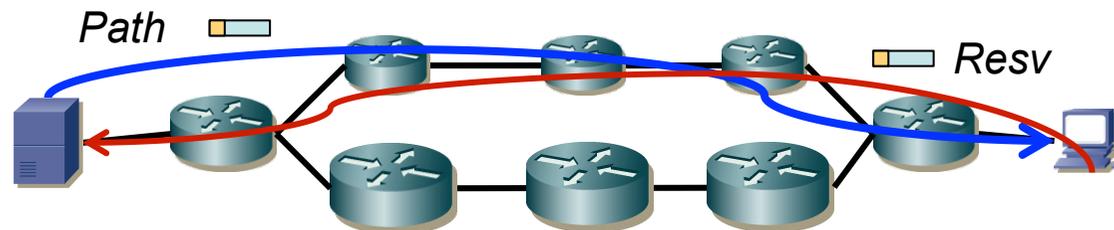
RSVP : Path

- RSVP no es un protocolo de encaminamiento
- No decide el camino que debe llevar el flujo
- Los datos seguirán el camino unicast/multicast decidido por otros protocolos
- El mensaje RSVP de reserva (*Resv*) seguiría así el camino indicado por las tablas de rutas
- Si las rutas son asimétricas los datos y la reserva no siguen el mismo camino: ¡ reserva hecha por donde no van los datos !
- Para resolverlo hay otro tipo de mensaje (*Path*) que envía la fuente



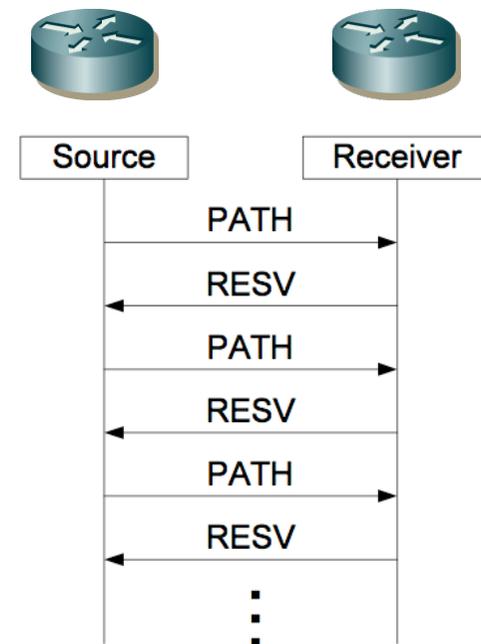
RSVP : Path

- El mensaje Path lo envía la fuente y sigue la ruta calculada por los protocolos de encaminamiento
- El mensaje Path sigue el mismo camino que seguirán los datos
- Este mensaje establece un estado en los routers que soportan RSVP en el camino
- El mensaje Resv emplea ese estado para seguir el camino inverso
- Si no hay recursos suficientes falla la reserva
- Falla incluso si existe otro camino que sí disponga de recursos
- Los mensajes *Path* y *Resv* se envían de forma periódica para mantener el estado instalado
- El estado es un “soft state” lo cual quiere decir que se elimina si no se refresca



RSVP: *states*

- Actualizaciones periódicas refrescan el estado
- Se libera al dejar de recibir actualizaciones
- Permite adaptarse ante cambios en la topología
- Permite adaptarse ante cambios en los miembros de un árbol de distribución multicast
- Alternativa (no soportada): Hard state
 - Se mantiene hasta liberarlo explícitamente
 - Requiere algoritmo ante errores



RSVP : Parametrización

- Una reserva viene dada por un “flow descriptor”
- Un “flow descriptor” está compuesto por un “filter spec” y un “flow spec”

filterspec (*Filter specification*)

- Determina qué paquetes forman el flujo
- Flujo identificado en base a IPs + puertos + protocolo
- Permite configurar el clasificador

flowspec (*Flow specification*)

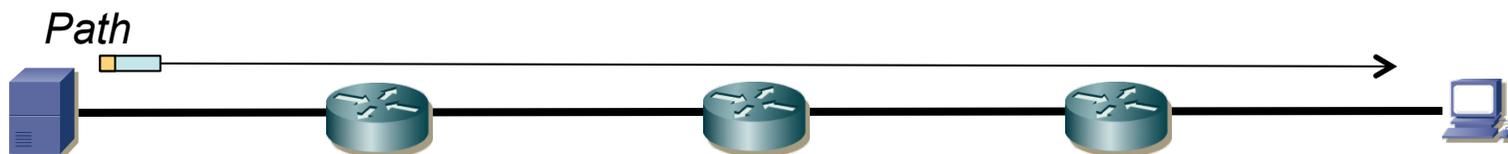
- Especifica la QoS deseada y permite configurar el planificador
- Emplea dos parámetros numéricos
- **TSpec** (*Traffic specification*)
 - Descripción del tráfico
 - Parámetros de un *token bucket* por el que pasa el tráfico
 - Mean rate, token bucket depth, max rate, max packet length
- **RSpec** (*Service Request specification*)
 - Requisitos de QoS impuestos a la red
 - BW, retardo, probabilidad de pérdida

RSVP e IntServ

- RFC 2210 : “The Use of RSVP with IETF Integrated Services”
- Se especifica el contenido de los parámetros de QoS que llevan los mensajes de RSVP para hacer reservas IntServ
- RSVP SENDER_TSPEC objects:
 - Viaja en mensaje *Path*
 - Sender TSpec : descripción del tráfico generada por el emisor
 - Emplea para ello parámetros de token bucket
- RSVP ADSPEC objects (opcional):
 - Viaja en mensaje *Path*
 - Contienen información modificada por los nodos de la red sobre la disponibilidad de servicio en el camino
- RSVP FLOWSPEC objects:
 - Viaje en mensaje *Resv*
 - Receiver TSpec : descripción del tráfico dada por receptor
 - Receiver RSpec : reserva solicitada por receptor
 - Puede cambiar en la red debido a *mergings*

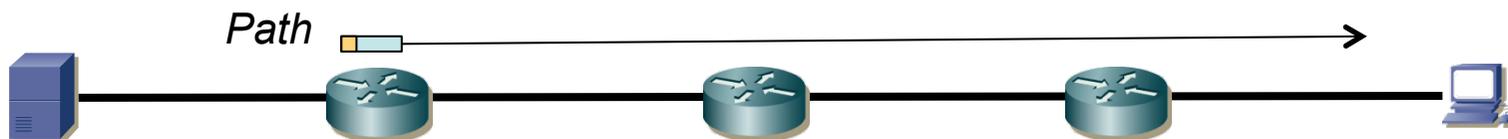
Ejemplo de reserva unicast

- La fuente de los datos envía mensaje *Path* conteniendo objetos:
 - Sender Template : describe los paquetes para poder clasificarlos (IPs, puertos)
 - SENDER_TSPEC : descripción del tráfico y por tanto recursos necesarios
 - ADSPEC
 - PHOP: *Previous Hop*, dirección IP suya
- Dirección IP origen de la fuente y destino del receptor
- (...)



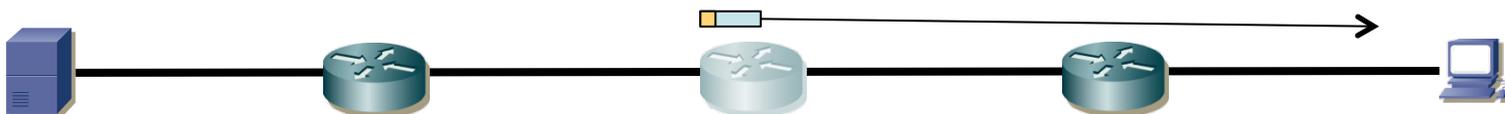
Ejemplo de reserva unicast

- *IP router Alert (RFC 2113)*
 - El paquete IP lleva la opción *IP Router Alert* (RFC 2113) para que los routers no simplemente conmuten el paquete
 - Hay que tener en cuenta que la IP destino es la del host final pero el paquete debe ser interceptado por el router
 - Esta opción permite optimizaciones en el procesado del router
- El router instala el estado que incluye la dirección IP del nodo upstream (PHOP)
- Modifica el ADSPEC según sus capacidades y recursos de QoS
- Reenvía en función de dirección IP destino del paquete IP (receptor final) y su tabla de rutas
- Objeto PHOP pasa a ser la dirección IP del interfaz por el que reenvía
- (...)



Ejemplo de reserva unicast

- *IP router Alert (RFC 2113)*
 - El paquete IP lleva la opción *IP Router Alert* (RFC 2113) para que los routers no simplemente conmuten el paquete
 - Hay que tener en cuenta que la IP destino es la del host final pero el paquete debe ser interceptado por el router
 - Esta opción permite optimizaciones en el procesado del router
- El router instala el estado que incluye la dirección IP del nodo upstream (PHOP)
- Modifica el ADSPEC según sus capacidades y recursos de QoS
- Reenvía en función de dirección IP destino del paquete IP (receptor final) y su tabla de rutas
- Objeto PHOP pasa a ser la dirección IP del interfaz por el que reenvía
- Si un router no soporta RSVP reenvía el paquete como un paquete IP normal y corriente
- (...)



Ejemplo de reserva unicast

- Los routers que soporten RSVP repiten el proceso:
 - El router instala el estado que incluye la dirección IP del nodo upstream (PHOP)
 - Modifica el ADSPEC según sus capacidades y recursos de QoS
 - Reenvía en función de dirección IP destino del paquete IP (receptor final) y su tabla de rutas
 - Objeto PHOP pasa a ser la dirección IP del interfaz por el que reenvía
- (...)



Ejemplo de reserva unicast

- Hasta llegar el paquete al que será el receptor del flujo de datos
- Ahora ya está instalado un *soft state* en todos los routers del camino que soporten RSVP
- Eso va a permitir al mensaje *Resv* seguir el camino inverso
- Aún no hay reserva hecha
- El `SENDER_TSPEC` no ha sido modificado y con eso la aplicación conoce las características del flujo
- El `ADSPEC` contiene la descripción de la QoS que podría ofrecer el camino
- La aplicación receptora podría emplear el `ADSPEC` para decidir el tipo de reserva a hacer (por ejemplo si va a poder pedir un vídeo SD o HD)
- (...)



Ejemplo de reserva unicast

- El receptor de los datos envía mensaje *Resv* conteniendo objetos:
 - FILTERSPEC : parámetros para clasificar a los paquetes de este flujo (srcIP+dstIP+srcPort+dstPort+protocol)
 - FLOWSPEC :
 - RECEIVER_TSPEC : descripción de la cantidad de tráfico para la que es la reserva (con parámetros de token bucket) y del tipo (GS, CL, BE)
 - RSPEC : existe solo para la clase Controlled Load e indica la reserva a hacer
- Dirección IP origen la del receptor de los datos, IP destino la del primer router RSVP en el camino (venía en el PHOP)
- (...)



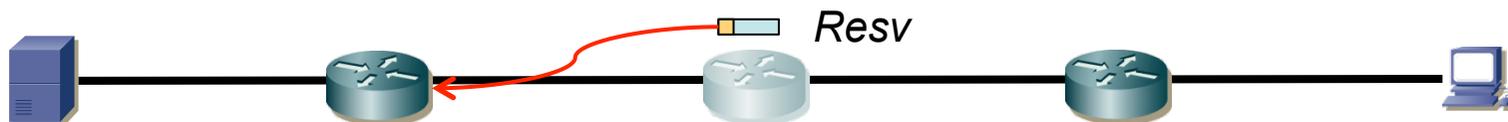
Ejemplo de reserva unicast

- Routers con soporte para RSVP aplican políticas para decidir si aceptan la reserva solicitada
- Se emplea control de admisión para decidir si hay recursos suficientes
- Si ambos tienen éxito:
 - Se instala clasificador basado en el FILTERSPEC
 - Se reservan recursos en el planificador del enlace por el que llega el *Resv* basados en el FLOWSPEC
- Puede tener que hacer *merging* con otras solicitudes y que se modifiquen los parámetros de la reserva upstream
- Si no hay errores envía el *Resv* upstream empleando la información de estado almacenada para que siga el camino inverso al del *Path*
- Dirección IP origen del router que envía, IP destino la del siguiente router RSVP upstream (valor que se guardó del PHOP)
- (...)



Ejemplo de reserva unicast

- Routers con soporte para RSVP aplican políticas para decidir si aceptan la reserva solicitada
- Se emplea control de admisión para decidir si hay recursos suficientes
- Si ambos tienen éxito:
 - Se instala clasificador basado en el FILTERSPEC
 - Se reservan recursos en el planificador del enlace por el que llega el *Resv* basados en el FLOWSPEC
- Puede tener que hacer *merging* con otras solicitudes y que se modifiquen los parámetros de la reserva upstream
- Si no hay errores envía el *Resv* upstream empleando la información de estado almacenada para que siga el camino inverso al del *Path*
- Dirección IP origen del router que envía, IP destino la del siguiente router RSVP upstream (valor que se guardó del PHOP)
- Los routers que no soportan RSVP reenvía el paquete IP con normalidad
- (...)



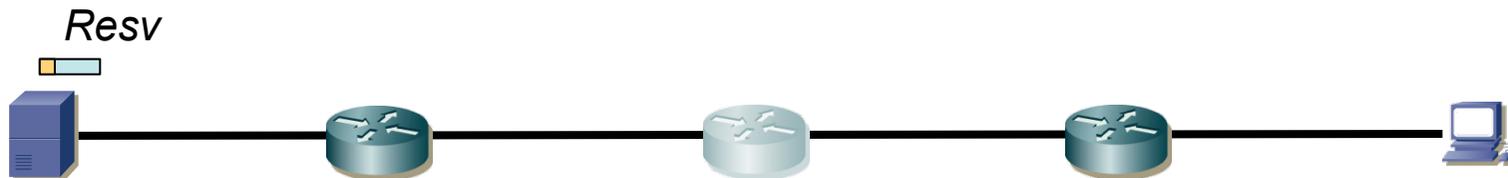
Ejemplo de reserva unicast

- Routers con soporte para RSVP aplican políticas para decidir si aceptan la reserva solicitada
- Se emplea control de admisión para decidir si hay recursos suficientes
- Si ambos tienen éxito:
 - Se instala clasificador basado en el FILTERSPEC
 - Se reservan recursos en el planificador del enlace por el que llega el *Resv* basados en el FLOWSPEC
- Puede tener que hacer *merging* con otras solicitudes y que se modifiquen los parámetros de la reserva upstream
- Si no hay errores envía el *Resv* upstream empleando la información de estado almacenada para que siga el camino inverso al del *Path*
- Dirección IP origen del router que envía, IP destino la del siguiente router RSVP upstream (valor que se guardó del PHOP)
- Los routers que no soportan RSVP reenvía el paquete IP con normalidad
- Los que sí lo soportan repiten el proceso de verificación y reserva (...)



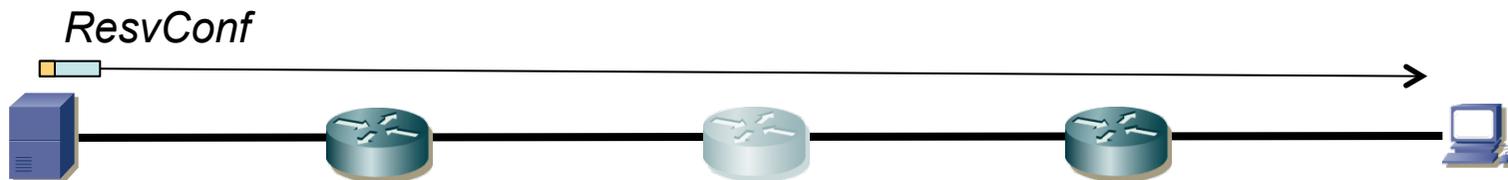
Ejemplo de reserva unicast

- La fuente recibe el *Resv* y conoce la reserva que se ha hecho
- (...)



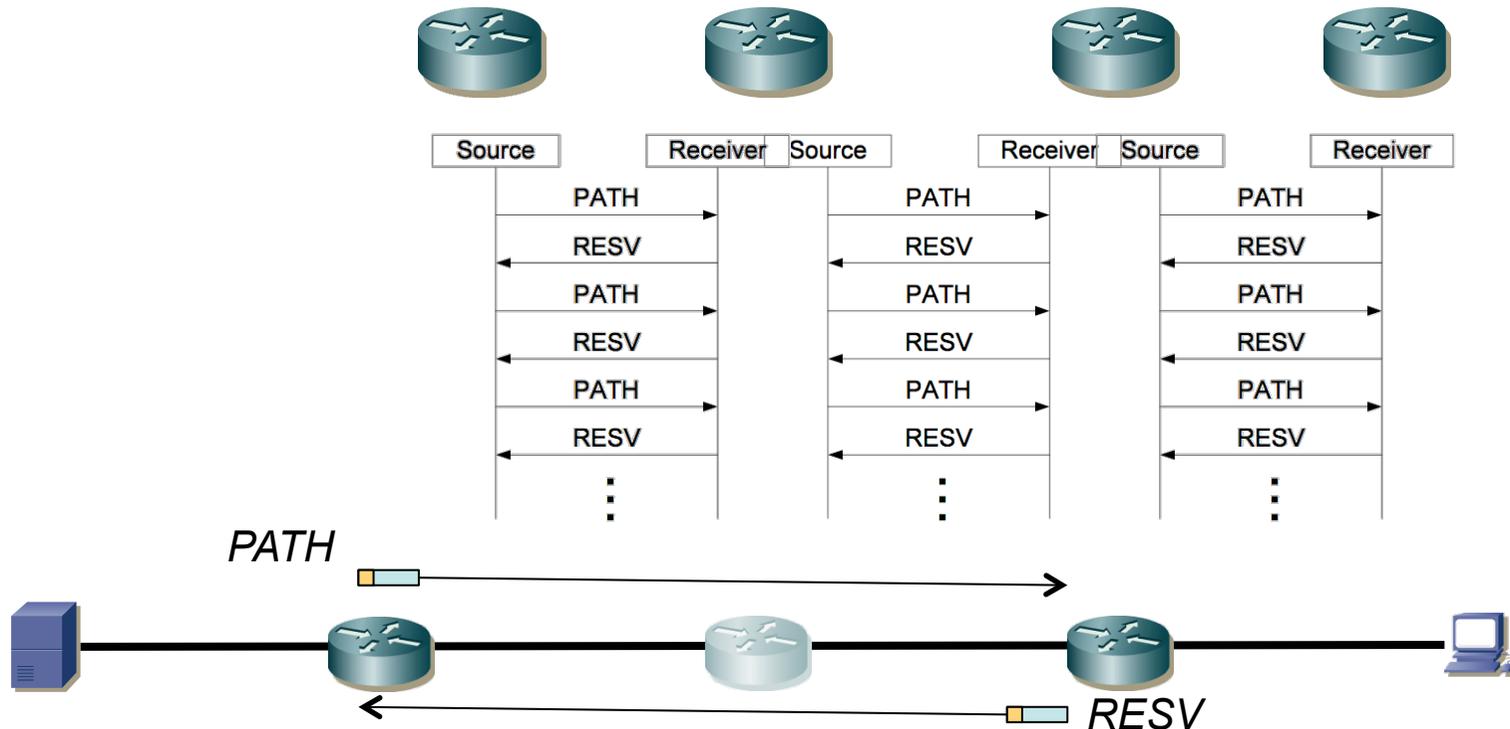
Ejemplo de reserva unicast

- La fuente recibe el *Resv* y conoce la reserva que se ha hecho
- El mensaje puede solicitar una confirmación (*ResvConf* hacia el receptor)
- (...)



Ejemplo de reserva unicast

- La fuente recibe el *Resv* y conoce la reserva que se ha hecho
- El mensaje puede solicitar una confirmación (*ResvConf* hacia el receptor)
- Para mantener el *soft state* se envían periódicamente de nuevo los mensajes *Path* y *Resv*
- Los envía cada nodo, no extremo a extremo
- (...)



Ejemplo de reserva unicast

- La fuente recibe el *Resv* y conoce la reserva que se ha hecho
- El mensaje puede solicitar una confirmación (*ResvConf* hacia el receptor)
- Para mantener el *soft state* se envían periódicamente de nuevo los mensajes *Path* y *Resv*
- Los envía cada nodo, no extremo a extremo
- Emisor o receptor pueden liberar la reserva enviando un *PathTear* o un *ResvTear* respectivamente

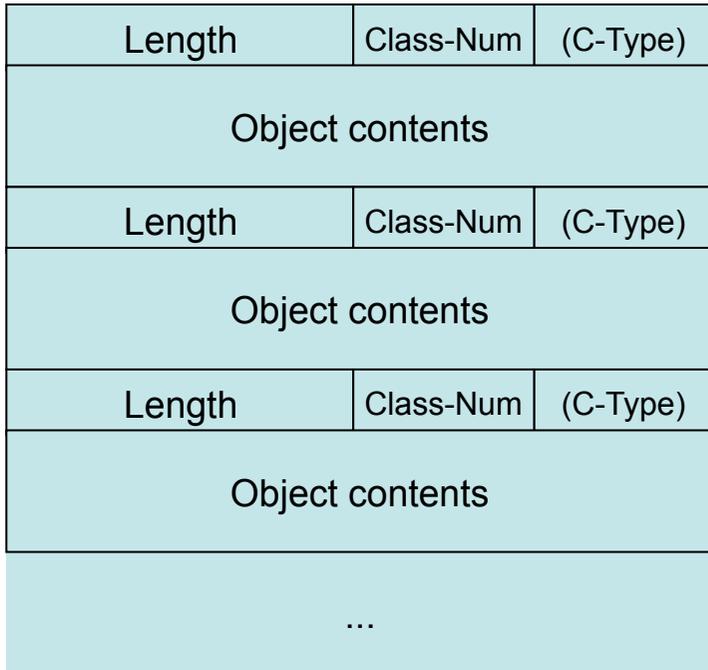


Uso de RSVP

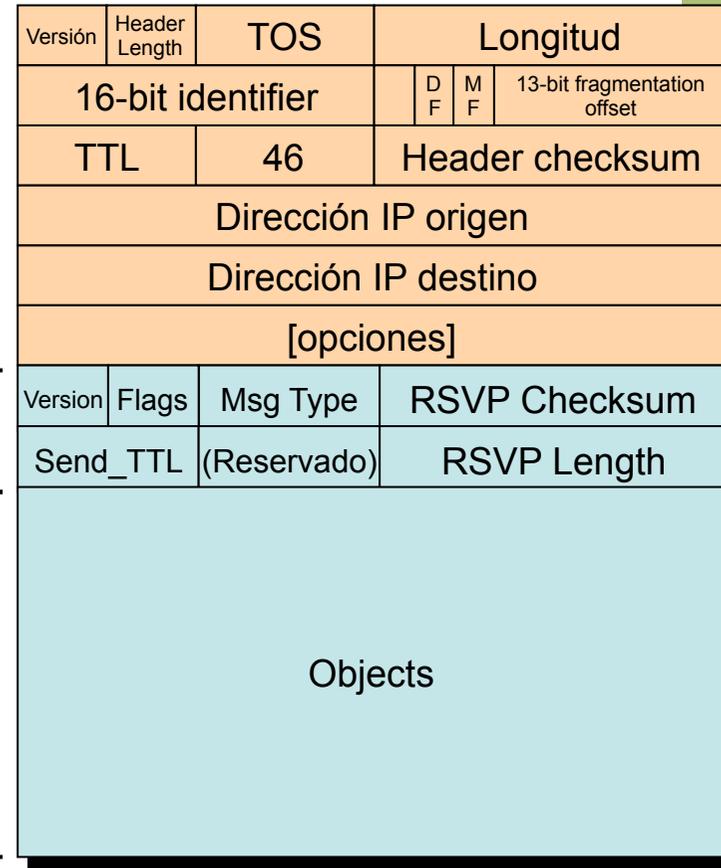
- Está bastante implementado
- Pero no está muy extendido su uso en redes IP
- Hay muchas dudas sobre la escalabilidad de la solución al requerir procesado por flujo
- Hay modificaciones que mejoran la escalabilidad reduciendo por ejemplo los mensajes de refresco
- También requiere intervención de la aplicación lo cual la vuelve más compleja
- API complicada

Paquete RSVP

- Msg Type: 1=Path, 2=Resv, 3=PathErr, 4=ResvErr, 5=PathTear, 6=ResvTear, 7=ResvConf
- Ejemplos: (...)



Common Header



Paquete RSVP

- Msg Type: 1=Path, 2=Resv, 3=PathErr, 4=ResvErr, 5=PathTear, 6=ResvTear, 7=ResvConf
- Ejemplos: (...)

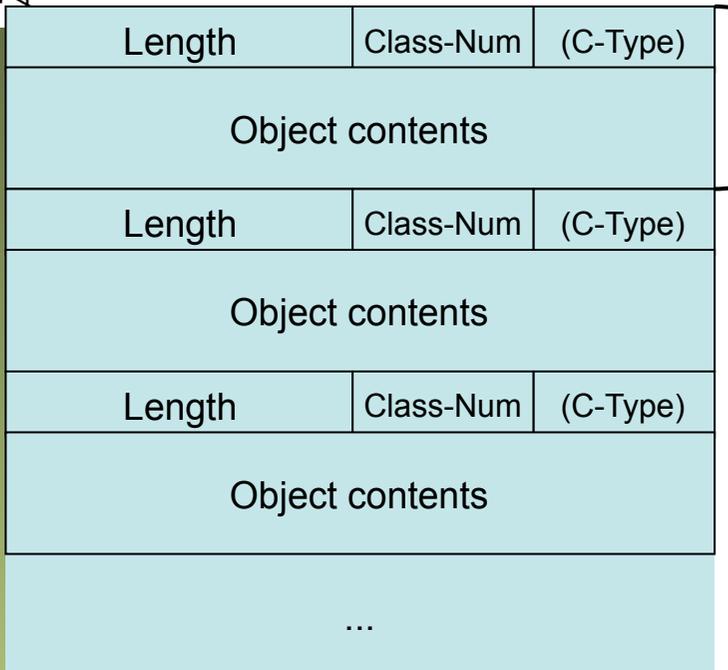
Versión	Header Length	TOS	Longitud		
16-bit identifier			D F	M F	13-bit fragmentation offset
TTL	46	Header checksum			
Dirección IP origen					
Dirección IP destino					
[opciones]					
Version	Flags	Msg Type	RSVP Checksum		
Send_TTL	(Reservado)	RSVP Length			

```

  ▸ Frame 1: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits)
  ▸ Ethernet II, Src: c0:01:12:08:00:00 (c0:01:12:08:00:00), Dst: c0:00:12:08:00:00 (c0:00:12:08:00:00)
  ▸ Internet Protocol Version 4, Src: 10.1.24.4 (10.1.24.4), Dst: 10.1.12.1 (10.1.12.1)
  ▾ Resource Reservation Protocol (RSVP): PATH Message. SESSION: IPv4, Destination 10.1.12.1, Protocol
    ▾ RSVP Header. PATH Message.
      RSVP Version: 1
      Flags: 00
      Message Type: PATH Message. (1)
      Message Checksum: 0x0a55 [correct]
      Sending TTL: 254
      Message length: 136
    ▾ SESSION: IPv4, Destination 10.1.12.1, Protocol 17, Port 16388.
    ▸ HOP: IPv4, 10.1.12.2
    ▸ TIME VALUES: 30000 ms
    ▸ SENDER TEMPLATE: IPv4, Sender 10.1.24.4, Port 16388.
    ▸ SENDER TSPEC: IntServ, Token Bucket, 6000 bytes/sec.
    ▸ ADSPEC
  
```

Paquete RSVP

- Msg Type: 1=Path, 2=Resv, 3=PathErr, 4=ResvErr, 5=PathTear, 6=ResvTear, 7=ResvConf
- Ejemplos: (...)



```

▶ Frame 6: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits)
▶ Ethernet II, Src: c0:01:12:08:00:00 (c0:01:12:08:00:00), Dst: c0:00:12:08:00:00
▶ Internet Protocol Version 4, Src: 10.1.24.4 (10.1.24.4), Dst: 10.1.12.1 (10.1.12.1)
▼ Resource Reservation Protocol (RSVP): PATH Message. SESSION: IPv4, Destination 10.1.12.1
  ▶ RSVP Header. PATH Message.
  ▶ SESSION: IPv4, Destination 10.1.12.1, Protocol 17, Port 16388.
  ▶ HOP: IPv4, 10.1.12.2
  ▶ TIME VALUES: 30000 ms
  ▶ SENDER TEMPLATE: IPv4, Sender 10.1.24.4, Port 16388.
  ▼ SENDER TSPEC: IntServ, Token Bucket, 6000 bytes/sec.
    Length: 36
    Object class: SENDER TSPEC object (12)
    C-type: 2 - Integrated Services
    Message format version: 0
    Data length: 7 words, not including header
    Service header: 1 - Traffic specification
    Length of service 1 data: 6 words, not including header
  ▼ Token Bucket TSpec: Rate=6000 Burst=6000 Peak=6000 m=0 M=2147483647
    Parameter 127 - Token bucket
    Parameter 127 flags: 0x00
    Parameter 127 data length: 5 words, not including header
    Token bucket rate: 6000
    Token bucket size: 6000
    Peak data rate: 6000
    Minimum policed unit [m]: 0
    Maximum packet size [M]: 2147483647
  ▶ ADSPEC
  
```

Paquete RSVP

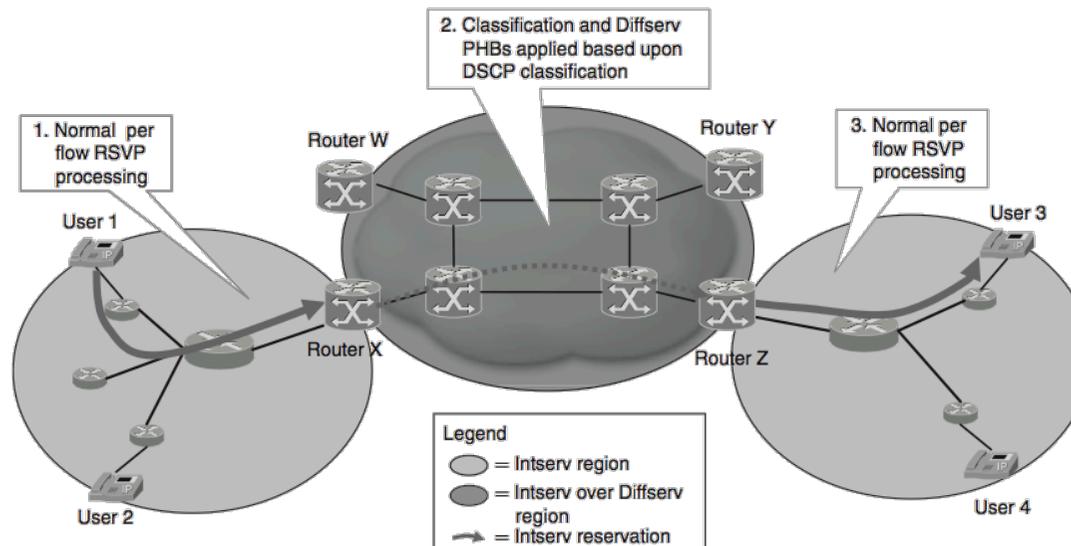
- Msg Type: 1=Path, 2=Resv, 3=PathErr, 4=ResvErr, 5=PathTear, 6=ResvTear, 7=ResvConf
- Ejemplos:

SENDER_TSPEC con parámetros de token bucket

```
▼ SENDER TSPEC: IntServ, Token Bucket, 6000 bytes/sec.  
  Length: 36  
  Object class: SENDER TSPEC object (12)  
  C-type: 2 - Integrated Services  
  Message format version: 0  
  Data length: 7 words, not including header  
  Service header: 1 - Traffic specification  
  Length of service 1 data: 6 words, not including header  
▼ Token Bucket TSpec: Rate=6000 Burst=6000 Peak=6000 m=0 M=2147483647  
  Parameter 127 - Token bucket  
  Parameter 127 flags: 0x00  
  Parameter 127 data length: 5 words, not including header  
  Token bucket rate: 6000  
  Token bucket size: 6000  
  Peak data rate: 6000  
  Minimum policed unit [m]: 0  
  Maximum packet size [M]: 2147483647
```

IntServ over DiffServ

- RFC 2998 : “A Framework for Integrated Services Operation over Diffserv Networks”
- Se plantea cómo interconectar dominios IntServ a través de dominios DiffServ
- Paquetes de una reserva GS deberían marcarse para emplear el PHB EF
- Paquetes de una reserva CL podrían marcarse para emplear un PHB AF
- ¿Control de admisión en la región DiffServ? Opciones:
 - a. Los routers del dominio DiffServ ignoran los mensajes RSVP: el dominio debe ser capaz de soportar todo el tráfico recibido
 - b. CAC en la frontera: router frontera ve los Resv y conoce los flujos que atraviesan el dominio DS; hace con eso CAC
 - c. CAC en cada salto del dominio DS: routers DS entienden RSVP como para hacer CAC aunque su plano de datos sigue siendo DS



RSVP-TE

- RFC 3209 : “RSVP-TE: Extensions to RSVP for LSP Tunnels”
- Muy extendido en combinación con MPLS/GMPLS
- RSVP-TE permite especificar el camino en vez de dejarlo en manos del IGP
- Establece MPLS LSPs, incluyendo el intercambio de etiquetas
- Permite reservas bidireccionales

Resumen

- Guaranteed service y Controlled load service
- RSVP: mensajes Path y Resv
- Resv camino inverso al Path mediante *soft-state*
- La reserva la hace el receptor
- Parámetros de token bucket