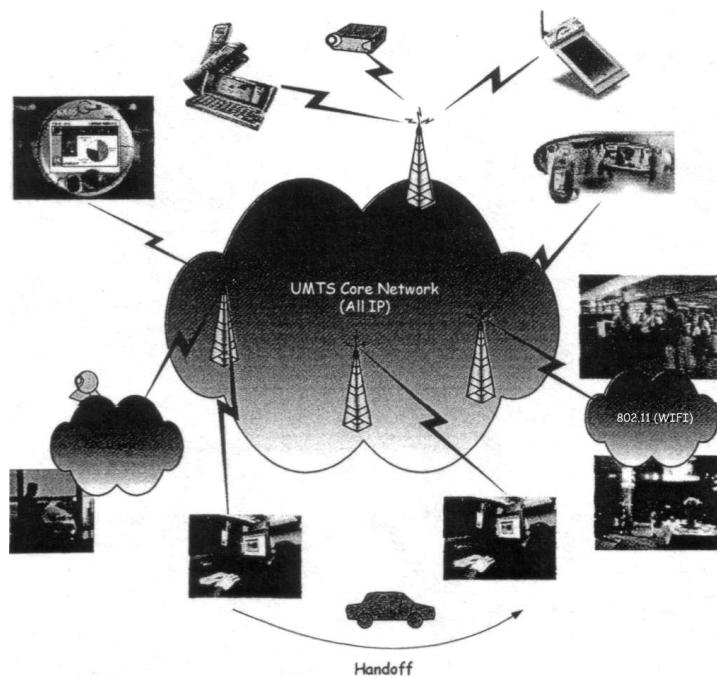


# **Label Switching Course Book**

**versione 1.0**  
**3 giugno 2009**



## **Architetture Avanzate di Networking & Sistemi Wireless**

**Facoltà di Ingegneria Informatica dell'Università di Pisa  
a/a 2008/09**



# Introduzione

Il presente testo è una trascrizione in forma di dispensa delle slides ufficiali del corso di Architetture Avanzate di Networking & Sistemi Wireless. Tale corso è tenuto dal professor L. Lenzini ed è uno degli insegnamenti previsti dalla laurea di Ingegneria Informatica dell'Università di Pisa. Le slides trascritte sono quelle dell'anno accademico 2008/09, è possibile che non siano le più recenti. La dispensa è interamente in inglese, come le slides da cui è stata tratta. La forma di dispensa ha un numero di pagine che è circa un terzo della forma in slides. Stampare questa dispensa (invece di stampare le slides) assicura così un minor spreco di carta, un minore ingombro e una minore fatica nello sfogliare le pagine. Avere gli argomenti esposti in una forma più compatta può anche agevolare lo studio e aiutare la memorizzazione. Inoltre, la dispensa può essere anche letta in formato elettronico senza bisogno di stampe. In alcuni punti l'autore ha aggiunto delle frasi, prese dalle lezioni del prof. Lenzini. Tali frasi sono evidenziate in nero. La dispensa è fornita in formato PDF. Per correzioni, ampliamenti ed aggiornamenti è possibile scaricare il formato ODT (OpenOffice.Org) dal circuito eDonkey cercando "Label Switching Course Book".

Lista delle slide trascritte nella presente dispensa:

1. Introduction & IP QoS
2. Scheduling Algorithms for QoS Guarantees
3. IntServ
4. DiffServ
5. MPLS Architecture
6. Traffic Engineering
7. Virtual Private Networks
8. 802.11e (in formato elettronico)
9. WiMAX (in formato elettronico)
10. Long Term Evolution (in formato elettronico)
11. Radio Propagation Basics
12. Mesh Networks
13. Channel Assignment Problem (CLICA Algorithm)
14. Routing for MANETs (AODV Protocol)
15. Simulation with NS-2

# Modalità di Esame

Modalità di svolgimento:

- Progetto o prova scritta (a scelta)
- Prova orale

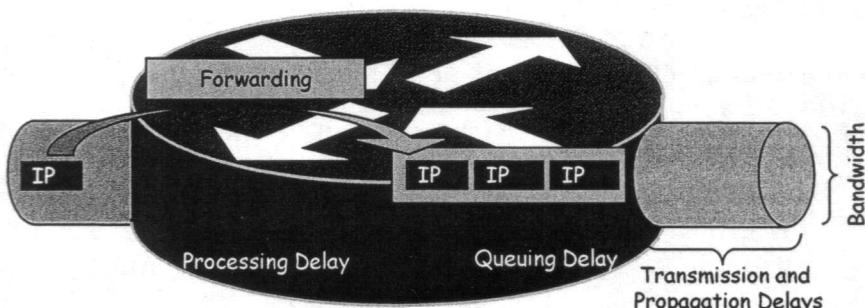
Il progetto viene implementato da un gruppo di studenti sotto la guida di un supervisore. Normalmente consiste nella valutazione delle prestazioni di un protocollo tramite NS2 anche se, con il presente a.a. (2008/09), altre alternative saranno possibili. Durante la prova orale gli studenti del gruppo illustrano i risultati del progetto mediante una presentazione PPT. Il file PPT ed il codice debbono essere consegnati al docente almeno una settimana prima della prova orale. Il progetto rimane valido per la durata dell'anno accademico 2008/2009. L'esame (progetto & prova orale) deve essere sostenuto da tutto il gruppo. Se almeno uno studente del gruppo intende svolgere una tesi sul networking, Sarà cura del docente scegliere un progetto che sia in traccia con la finalità della tesi. In questo modo esiste una continuità tra attività di progetto ed eventuale attività di tesi con considerevole risparmio di tempo da parte del tesista.

## 1 The Internet With Quality of Service (IP QoS)

### 1.1 Why IP QoS?

For many years, packet-switched networks have offered the promise of supporting multimedia applications, that is, those that combine *audio*, *video*, and *data*. Obviously, once digitized, audio and video information become like any other form of data - a stream of bits to be transmitted. One obstacle to the fulfillment of this promise has been the need for higher-bandwidth links. Recently, however, improvements in coding have reduced the bandwidth needs of audio and video applications, and link speeds have increased. There is more to transmitting audio and video over a network than just providing sufficient bandwidth, however. Participants in a telephone conversation expect to be able to converse in such a way that one person can respond to the other and be heard almost immediately. Thus, for audio and video applications the *timeliness* of delivery can be very important. We refer to applications that are sensitive to the timeliness of data as *real-time applications*. Voice and video applications tend to be the canonical examples, but there are others such as *industrial control* - you would like a command sent to a robot arm to reach it before the arm crashes into something. Even *file transfer* applications can have timeliness constraints, such as a requirement that a database update complete overnight before the business that needs the data resumes on the next day. The distinguishing characteristic of real-time applications is that they need some sort of assurance from the network that data is likely to arrive on time. Whereas a non-real-time application can use an end-to-end retransmission strategy to make sure that data arrives correctly, such a strategy cannot provide timeliness.

Processing/Queuing/Propagation Delays:



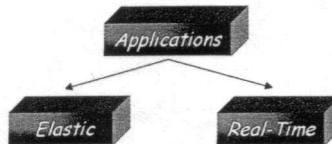
- *Processing Delay* is the time it takes for a router to take the packet from an input interface and put it into the output queue of the output interface. It depends on various factors (CPU speed, CPU utilization, IP switching mode, router architecture, configured features on both input and output interface, ...).
- *Queuing Delay* is the time a packets resides in the output queue of a router. It depends on the number and sizes of packets already in the queue and on the bandwidth of the output link. It also depends on the queuing mechanism.
- *Transmission and Propagation Delays* are the times needed to transmit and "propagate" a packet to the adjacent router/host respectively. They depend on the bandwidth and

propagation delay of the output link.

Timely arrival must be provided by the network itself (the routers), not just at the network edges (the hosts). We therefore conclude that the best-effort model, in which the network tries to deliver your data but makes no promises and leaves the cleanup operation to the edges, is not sufficient for real-time applications. What we need is a new service model (or service paradigm), in which applications that need higher assurances can ask the network for them. The network may then respond by providing the assurance, or perhaps by saying that it cannot promise anything better at the moment. Such a service model is a superset of the current model: applications that are happy with best-effort service should be able to use the new service model since their requirements are just less stringent. This implies that the network will treat some packets differently from others - something that is not done in the best-effort model. *Definition:* A network that can provide these different levels of service is said to support *Quality of Service (QoS)*. This leads to a service model with not just one class (best-effort), but with several service classes, each available to meet the needs of some set of applications. So, IP with QoS allows applications to request and receive predictable service levels in terms of (for instance) rate, e2e delay, e2e delay variations (jitter), etc.

## 1.2 Elastic Applications

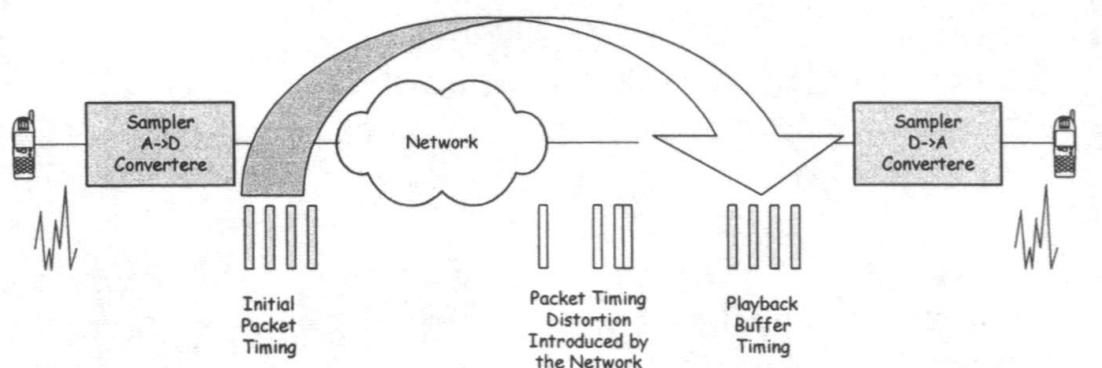
Before looking at the various protocols and mechanisms that may be used to provide QoS to applications, we should try to understand what the needs of those applications are. To begin, we can divide applications into two types: *non-real-time (elastic, traditional data) applications*, and *real-time applications*.



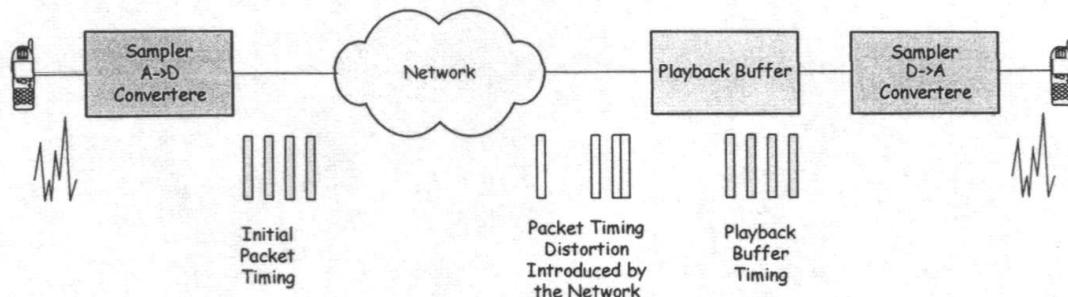
Another term for this non-real-time class of applications is *elastic*, since they are able to stretch gracefully in the face of increased delay. Non-real-time applications are sometimes called traditional data applications, since they have traditionally been the major applications found on data networks. All of these applications can work without guarantees of timely delivery of data. Such applications do not particularly suffer if their traffic is treated unreliably; best-effort service is acceptable as long as some network resources are still available. These applications will operate over a wide range of data rates, delay bounds, and reliability conditions. Any application that expects the user to wait after clicking a button or typing a command when communicating over the Internet is a likely candidate for an elastic application. They will simply adapt to the networks characteristics and slow down or speed up accordingly. Applications such as *e-mail*, *WWW browsers*, *file transport*, and *Internet news* are some concrete examples of elastic applications. Applications built on top of TCP can generally be described as elastic.

## 1.3 Real-Time Applications/Voice Service

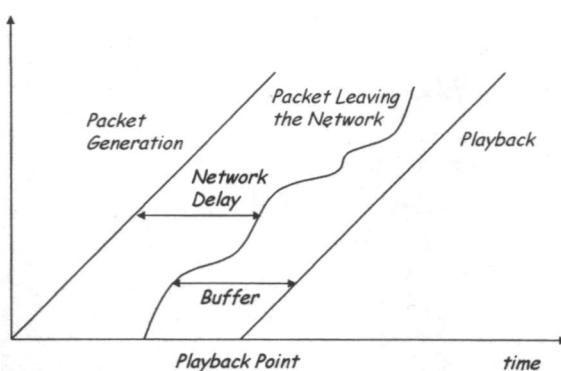
As a concrete example of a real-time (RT) application, consider an audio application. Data is generated by collecting samples from a microphone and digitizing them using an analog-to-digital (A->D) converter. The digital samples are placed in packets, which are transmitted across the network and received at the other end. At the receiving host, the data must be played back at some appropriate rate. For example, if the voice samples were collected at a rate of one per 125  $\mu$ sec, they should be played back at the same rate. Thus, we can think of each sample as having a particular *playback time*: the point in time at which it is needed in the receiving host. In the voice example, each sample has a playback time that is 125  $\mu$ sec later than the preceding sample.



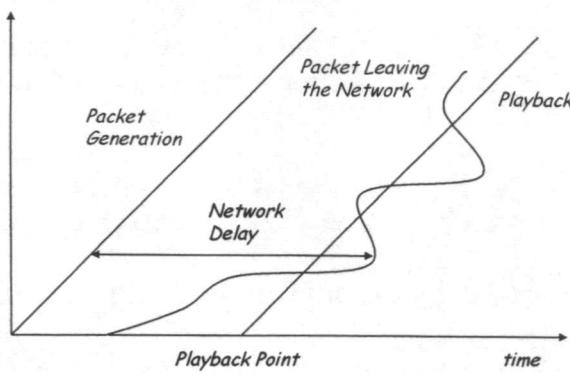
If data arrives after its appropriate playback time, either because it was delayed in the network or because it was dropped and subsequently retransmitted, it is essentially useless. It is the complete worthlessness of late data that characterizes real-time applications.



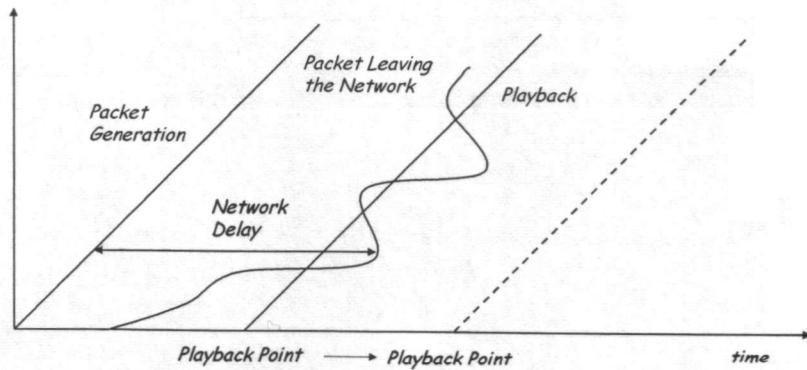
In elastic applications, it might be nice if data turns up on time, but we can still use it when it does not. One way to make our voice application work would be to make sure that all samples take exactly the same amount of time to traverse the network. Since samples are injected at a rate of one per 125  $\mu$ sec, they will appear at the receiver at the same rate, ready to be played back. However, it is generally difficult to guarantee that all data traversing a packet-switched network will experience exactly the same delay. Packets encounter queues in switches or routers and the lengths of these queues vary with time, meaning that the delays tend to vary with time and, as a consequence, are potentially different for each packet in the audio stream. The way to deal with this at the receiver end is to buffer up some amount of data in reserve, thereby always providing a store of packets waiting to be played back at the right time. If a packet is delayed a short time, it goes in the buffer until its playback time arrives. If it gets delayed a long time, then it will not need to be stored for very long in the receiver's buffer before being played back. Thus, we have effectively added a constant offset to the playback time of all packets as a form of insurance. We call this offset the *playback point*. The only time we run into trouble is if packets get delayed in the network for such a long time that they arrive after their playback time, causing the playback buffer to be drained.



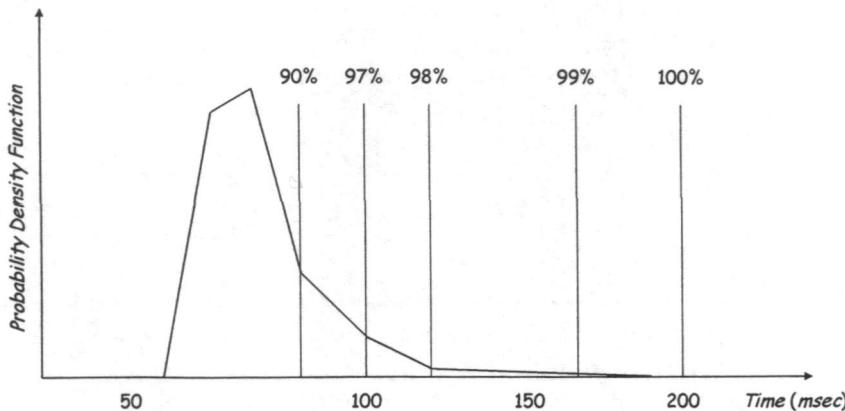
The left-hand diagonal line shows packets being generated at a steady rate. The wavy line shows when the packets arrive, some variable amount of time after they were sent, depending on what they encountered in the network. The right-hand diagonal line shows the packets being played back at a steady rate, after sitting in the playback buffer for some period of time. As long as the playback line is far enough to the right in time, the variation in network delay is never noticed by the application. However, when the playback line stays on the left, then some packets will begin to arrive too late to be useful.



For audio application, there are limits to how far we can delay playing back data.



It is hard to carry on a conversation if the time between when you speak and when your listener hears you is more than 300 ms. Thus, what we want from the network in this case is a guarantee that all our data will arrive within 300 ms. If data arrives early, we buffer it until its correct playback time. If it arrives late, we have no use for it and must discard it. To get a better appreciation of how variable network delay can be, see the figure below which shows the *probability density function* of one-way delay measured over a certain path across the Internet over the course of one particular day.



As denoted by the cumulative percentages given across the top of the graph, 97% of the packets in this case had a latency of 100 ms or less. This means that if our example audio application were to set the playback point at 100 ms, then on average, 3 out of every 100 packets would arrive too late to be of any use. One important thing to notice about this graph is that the tail of the curve - how far it extends to the right - is very long. We would have to set the playback point at over 200 ms to ensure that all packets arrived in time. The audio quality depends on the sample rate and sample resolution, and its bit rate requirement is:

$$\text{Audio bit rate} = (\text{bits/sample})(\text{samples/sec})$$

The main difference between speech and audio applications is that audio requires more quantization levels sampled at higher rates. If the audio signal is vocal and instrumental music instead of speech, then it has a frequency range of 10 to 20,000 Hz. In current compact disc (CD) technology, audio is stored using 16-bit PCM coding of 44.1- kHz sampled signals giving a bit rate of 700 kbps. In the case of stereo music, this bit rate is doubled to 1.4 Mbps.

## 1.4 Real-Time Applications/Image Distribution Center

Consider a black-and-white X-ray to be transmitted from a medical file to a radiologist. The vertical and horizontal axes of a single image are each broken into 2000 lines, or levels, to ensure the high resolution required of the image to be read. This is referred to as 2000 x 2000 resolution, giving rise to 4,000,000 picture elements (pixels) per image. If 8-bit gray scale quantization per pixel is used (this provides  $2^8 = 256$  levels of intensity), and a picture is to be transmitted in less than 1 sec, the bandwidth required turns out to be more than 32 Mbps.

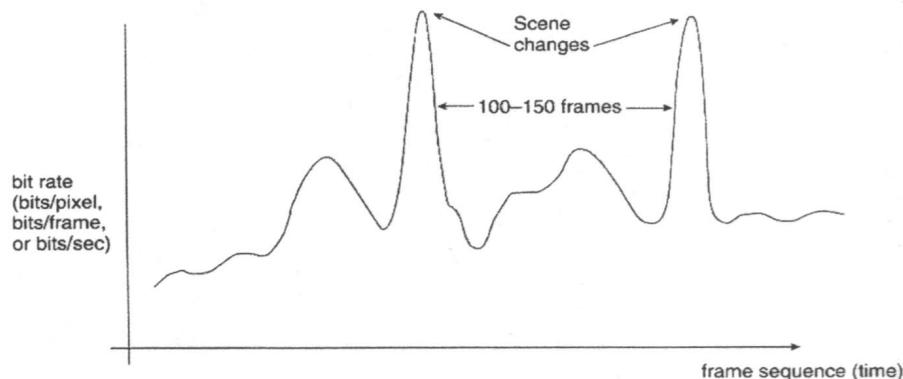
$$\frac{(4 \times 10^6) \times 8}{1} = 32 \text{ Mbps}$$

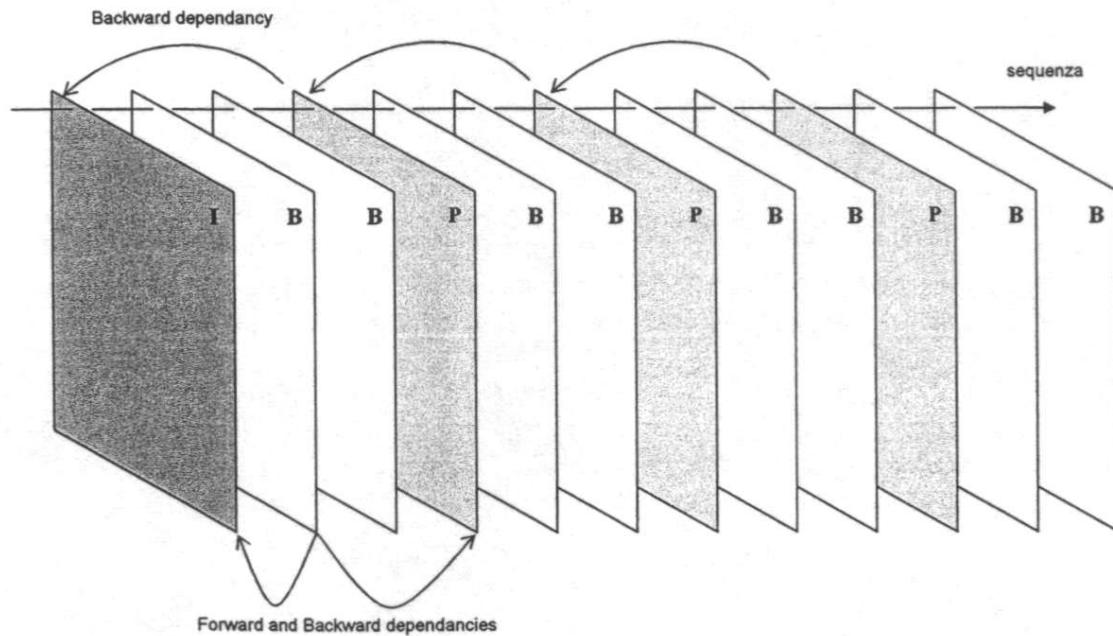
If a three - color picture with the same specifications is to be transmitted, 96 Mbps capacity is required. Compression could, however, be used to reduce this number considerably.

## 1.5 Real-Time Applications/Video Service

TV sets in the United States currently provide a resolution per frame, or picture, of approximately 500 x 500, or 250,000 pixels. Consider colour television transmitting 30 frames/sec. This translates to 7.5 million pixels/sec. If an 8-bit (256 levels of amplitude) grey scale is used, this gives rise to a rate of 60 Mbps. For three colours to be transmitted, this means 180 Mbps. Compression, using a variety of coding techniques, can reduce these values considerably. With 0.5 bit/pixel transmission attained, on the average, through the use of compression, for example, this reduces the transmission rate for a video signal to 3.75 Mbits/sec for monochrome video or 11.25 Mbits/sec for color signals. High-definition TV, under development, would require much higher bandwidths, however. A 1000 x 1000 frame, with 24 bits/pixel used, requires 720 Mbps of capacity. Recent gains in compression techniques have brought this number down considerably to an expected value of the order of 50 Mbits/sec or less. For this reason, B-ISDN rates have initially been chosen as 155.52 Mbps and 622 Mbps (two of the SONET rates), with the possibility of eventually reaching 2.4 Gbps and above. An ATM-based 155.52 Mbps UNI would thus be handling (155.52/[53x8]) 367,000 cells/sec in either direction across the interface. At 622 Mbps, this represents 1.467x10<sup>6</sup> cells/sec! Much of the traffic would be expected to be video or images. MPEG-2 coding is one of several coding standards being developed by the Moving Pictures Experts Group (MPEG), a working group of ISO and IEC. MPEG-2 involves higher-quality coding, including that for HDTV.

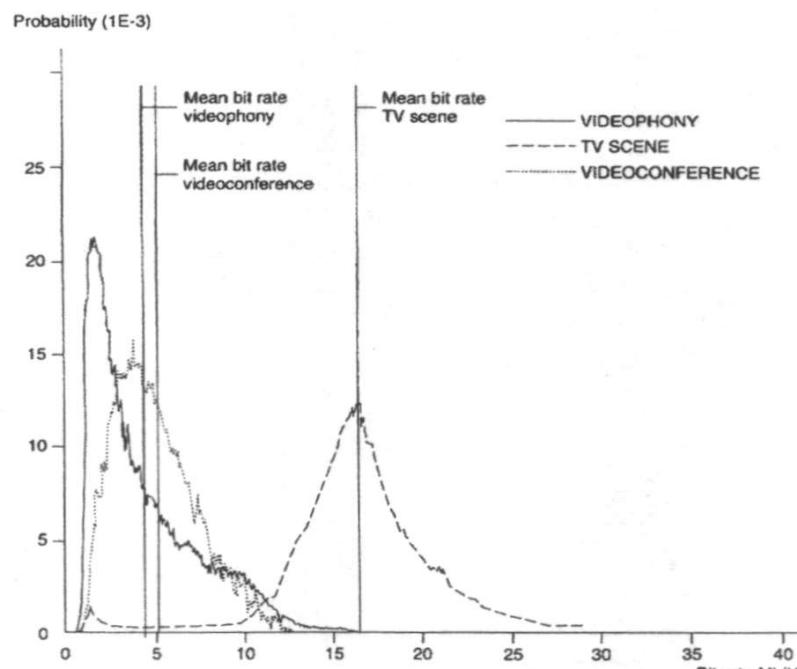
Typical compressed video bit rate:



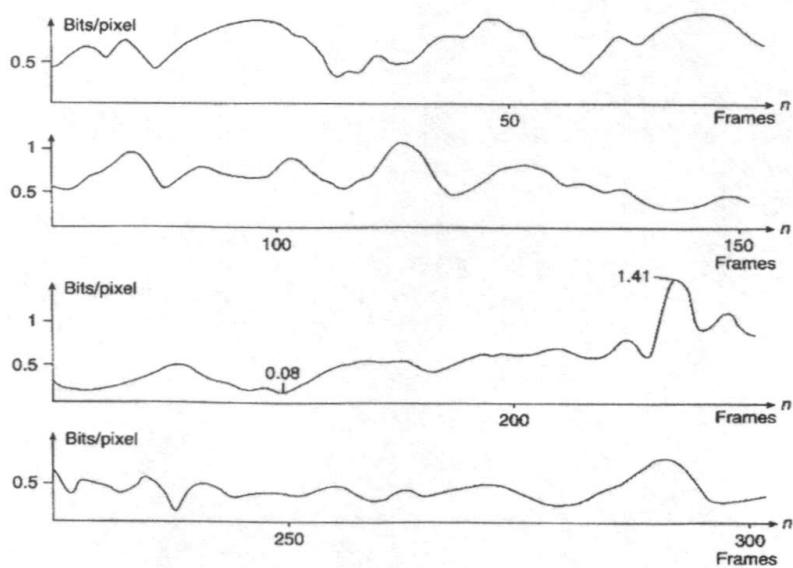


*Schematizzazione delle dipendenze esistenti tra i diversi tipi di quadro in un GoP*

Bit rate pdf:

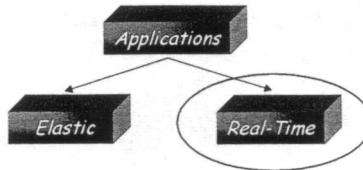


Head of a talking person:

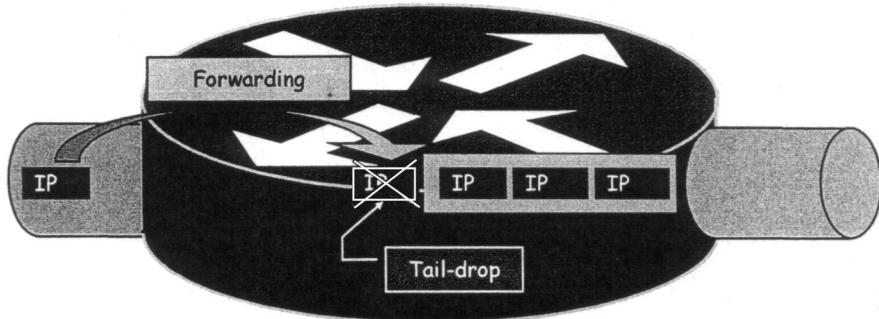


## 1.6 Taxonomy of Real-Time Applications

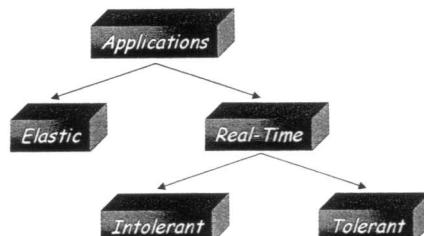
Now that we have a concrete idea of how real-time applications work, we can look at some different classes of applications, which serve to motivate the service model described below.



The first characteristic by which we can categorize real-time applications is their tolerance of loss of data, where loss might occur for several reasons, e.g. tail-drops, transmission errors, hardware problems. Tail-drops occur when the output queue is full. These are the most common drops which happen when a link is congested.

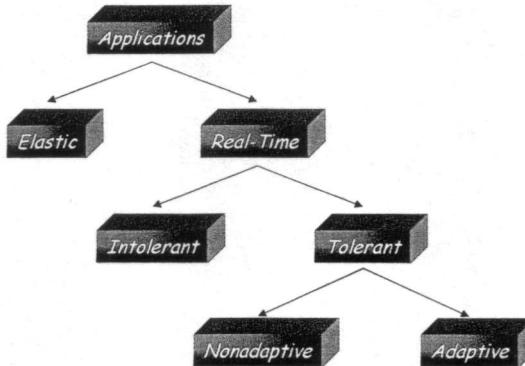


There are also many other types of drops (input queue drop, ignore, overrun, etc.), which are not as common and which may require a hardware upgrade. These drops are usually a result of router congestion. One lost audio sample can be interpolated from the surrounding samples with relatively little effect on the perceived audio quality. It is only as more and more samples are lost that quality declines to the point that the speech becomes incomprehensible. However, a robot control program is likely to be an example of a real-time application that cannot tolerate loss - losing the packet that contains the command instructing the robot arm to stop is unacceptable. Thus, we can categorize real-time applications as *tolerant* or *intolerant* depending on whether they can tolerate occasional loss.

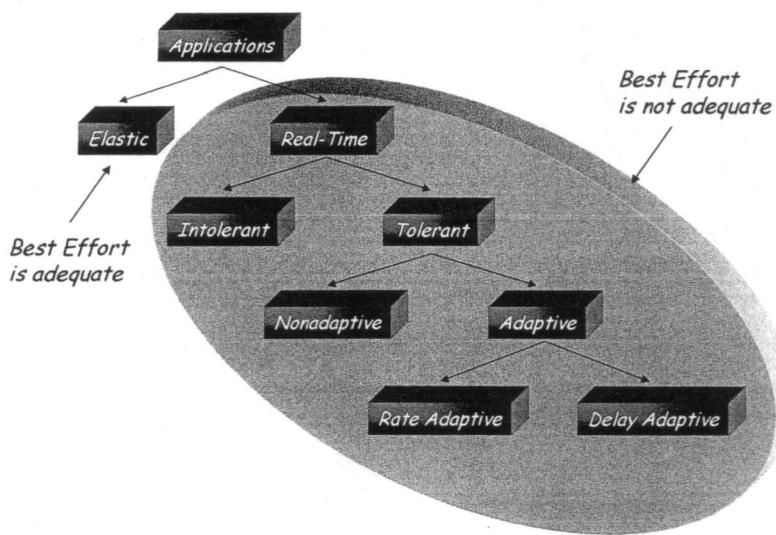


Note that many real-time applications are more tolerant of occasional loss than non-real-time applications. For example, compare our audio application to FTP, where the uncorrected loss of

one bit might render a file completely useless. A second way to characterize real-time tolerant applications is by their *adaptability*. For example, an audio application might be able to adapt to the amount of delay that packets experience as they traverse the network. If we notice that packets are almost always arriving within 200 ms of being sent, then we can set our playback point accordingly, buffering any packets that arrive in less than 200 ms. Suppose that we subsequently observe that all packets are arriving within 100 ms of being sent. If we moved up our playback point to 100 ms, then the users of the application would probably perceive an improvement.

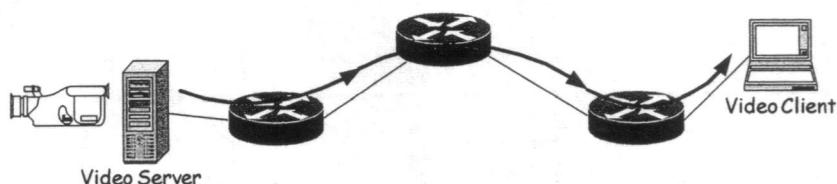


Observe that if we set our playback point on the assumption that all packets will arrive within 100 ms and then find that some packets are arriving slightly late, we will have to drop them, whereas we would not have had to drop them if we had left the playback point at 200 ms. Thus, we should advance the playback point only when it provides a perceptible advantage and only when we have some evidence that the number of late packets will be acceptably small. We may do this because of observed recent history or because of some assurance from the network. We call applications that can adjust their playback point *delay adaptive applications*. Another class of adaptive applications are *rate adaptive*. For example, many video coding algorithms can trade off bit rate versus quality. Thus, if we find that the network can support a certain bandwidth, we can set our coding parameters accordingly. If more bandwidth becomes available later, we can change parameters to increase the quality.



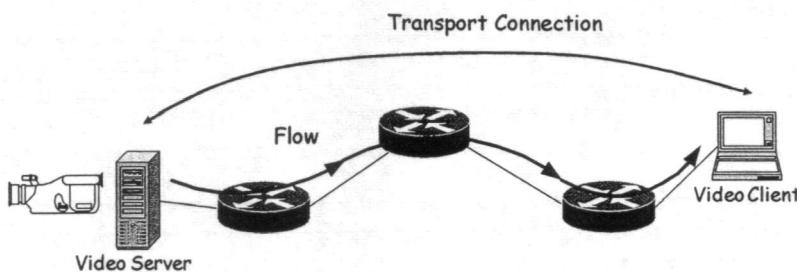
## 1.7 Flow

Before discussing the QoS IP architectures, we need to consider the notion of *flow*. A flow can be defined as a distinguishable stream of related packets travelling from the same source (sender) to one or more destinations (receivers) that have the same QoS requirements. The stream of packets results from a single user activity which might be a video conference, a telephone conversation, etc.

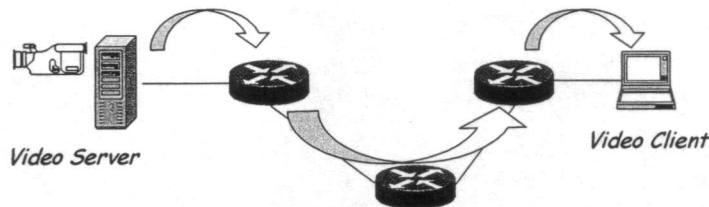


From the abstraction standpoint a flow is essentially the same as a transport connection,

however a flow is visible to the routers inside the network, whereas a transport connection is an end-to-end abstraction.



For a video conferencing system which needs a minimum bandwidth of 128 kbps and a maximum packet delay of 100 ms to assure a continuous video display, such a QoS should be reserved for this flow. In the *unicast* case, there will only be one flow, from a single source to a single destination.



A unicast flow is inherently homogeneous from the QoS standpoint. The above figure shows a sample unicast flow where packets belonging to the flow originate at the video server, travel through three routers and arrive at the video client.

## 1.8 Approaches to QoS Support

Considering this rich space of application requirements, what we need is a richer service model that meets the needs of any application. This leads us to a service model with not just one class (best effort), but with several classes, each available to meet the needs of some set of applications. The various approaches that have been developed to provide a range of qualities of service can be divided into two broad categories:

- *fine-grained* approaches, which provide QoS to individual flows;
- *coarse-grained* approaches, which provide QoS to aggregated traffic flows.

In the fine-grained category we find *Integrated Services*, a QoS architecture developed first by the ITU-T (A TM) and then by the IETF (IntServ). Although this idea is not new it is still a significant challenge to integrate resource reservation into the existing Internet architecture. In the coarse-grained category lies *Differentiated Services* (DiffServ), standardized by the IETF.

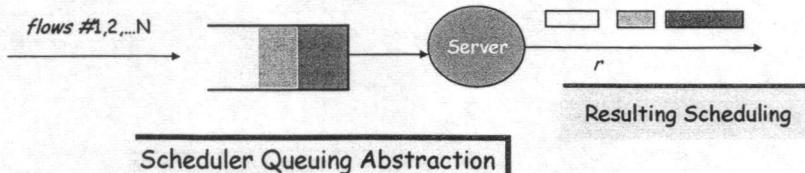
Notes:

- ITU-T: International Telecommunications Union - Telecommunication Standardization Sector.
- IETF: Internet Engineering Task Force.

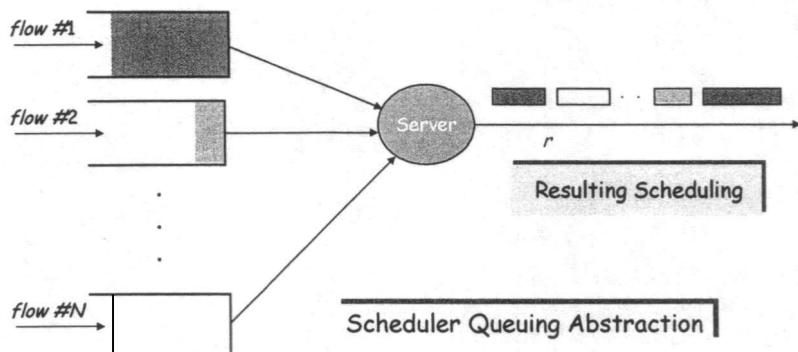
## 2 Packet Scheduling Algorithms for QoS

### 2.1 Introduction

In a packet-switched network (such as the Internet) offering a best-effort service, the performance of each flow can degrade significantly when the network is overloaded. Packets from different flows will interact with each other at each router; without proper control, these interactions may adversely affect the network performance experienced by applications.

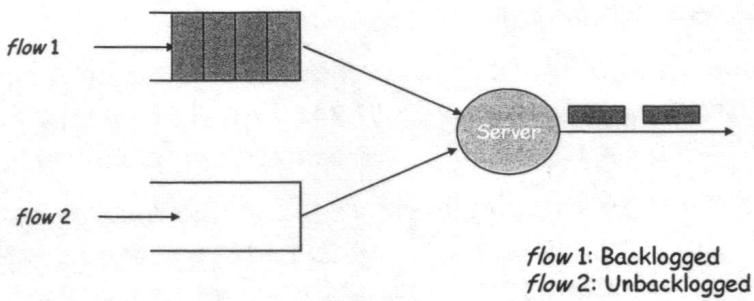


The scheduling algorithms at the routers control the order in which packets are serviced, and thus determine how packets from different flows interact with each other.



Specifically, the function of a scheduling algorithm is to select, for each outgoing link of the router, the packet to be transmitted next from the available packets belonging to the flows sharing the output link. We use flow to denote the entity based on which scheduler discriminates. In the literature:

- The term *switch* is used in the context of ATM networks, while *router* is used more often in an Internet environment. In the following, we will call switching elements as *router*.
- The terms *flows*, *sessions*, *connections* are used interchangeably. In the following, we will use the term *flow*.
- The terms *service discipline*, *service policy*, *scheduling discipline*, *scheduling policy* and *scheduling algorithm* are used interchangeably. In the following, we will use the term *scheduling algorithm*.
- A flow is *backlogged (active)* at time  $t$  if a positive amount of that flow's traffic is queued at time  $t$  and *idle (inactive)* otherwise.



### 2.2 Networking vs Hard Real-Time

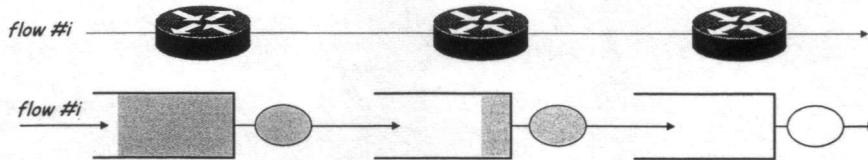
Although scheduling algorithms and associated performance problems have been widely studied in the contexts of hard real-time systems, results from these studies are not directly applicable in the context of providing guaranteed performance service in packet-switching networks. Analyses of hard real-time systems usually assume a single server environment, periodic jobs, and the job delay bounded by its period. However:

1. The network traffic is *bursty*, and (in general) the delay constraint for each individual connection is independent of its bandwidth requirement.

2. Bounds on end-to-end performance need to be guaranteed in a networking environment, where traffic dynamics are far more complex than in a single server environment.
3. In addition to the challenge of providing end-to-end per-flow performance guarantees to heterogeneous and bursty traffic, scheduling algorithms must be simple so that they can be implemented at very high speeds.

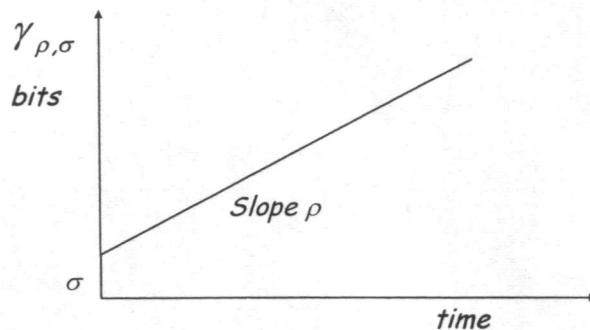
### 2.3 Network Model

We consider a network with arbitrary topology of links and routers. Links are assumed to have bounded delay (link latency). Packets destined for different output links do not interfere with each other, and queuing occurs only at the output ports of the router. The network supports variable-size packets. With the above assumptions, a flow in such a network can be modeled as traversing a number of queuing servers, with each server modeling the output link of a router.



### 2.4 Traffic Models

Although there is a general consensus within the research community on the (super) set of parameters to characterize performance requirements, there is no agreement on which traffic model or which set of traffic parameters should be adopted. In the traditional queuing theory literature, most models are based on stochastic processes. Among the more popular ones are the *Poisson* model for data, *on-off* model for voice sources and more sophisticated *Markovian* models for video sources. In general, these models are either too simple to characterize the important properties of the source or too complex for tractable analysis. Recently, several new models are proposed to bound the traffic rather than characterize the process exactly. The most popular model is the  $(\sigma, \rho)$  one. A traffic stream satisfies the  $(\sigma, \rho)$  model if during any interval of length  $t$ , the number of bits in that interval is less than  $\sigma + \rho t$ .



In the  $(\sigma, \rho)$  model,  $\sigma$  and  $\rho$  can be viewed as the *maximum burst size* and the *long term bounding rate* of the source respectively.

$$\gamma_{\rho, \sigma}(t) = \begin{cases} \sigma + \rho t & t > 0 \\ 0 & t \leq 0 \end{cases}$$

(Affine Function)

In the  $(\sigma, \rho)$  model, the exact traffic pattern for a flow is unknown, the only requirement is that the volume of the traffic be bounded.

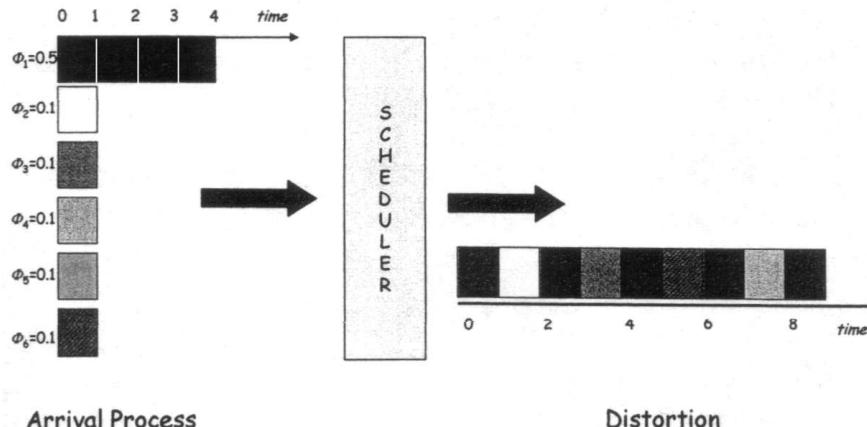
### 2.5 Desirable Properties

In order to be useful in practice, a scheduling algorithm is required to exhibit a number of properties:

- isolation of flows
- end-to-end delay guarantees for individual flows
- high utilization of the output link bandwidth
- fairness
- simplicity of implementation

- scalability

**Isolation of Flows.** The algorithm must isolate a flow from the undesirable effects of other (possibly misbehaving) flows. Ideally, a scheduling algorithm should provide each flow with the availability of a (virtually) dedicated link of capacity equal to its minimum guaranteed rate, so that the performance of a flow is not affected at all by the presence of other flows. However, in a packet environment, the presence of competing flows introduces some distortion with respect to the ideal case.



Although some distortion in the traffic profile due to packet atomicity is unavoidable in case of competition, it is important to keep it as small as possible. Specifically, a scheduling algorithm should prevent possibly misbehaving flows from stealing resources from other flows, thus jeopardizing their performance guarantees.

**End-to-End Delay Guarantees.** The scheduling algorithm must provide end-to-end delay guarantees for individual flows without severely under-utilizing the network resources. The delay behavior of an ideal scheduling algorithm also includes the following attributes:

- insensitivity to traffic patterns of other flows,
- delay bounds that are independent of the number of flows sharing the outgoing link,
- ability to control the delay bound of a flow by controlling only parameters of the flow, such as its bandwidth reservation.

Note that the above three attributes are related.

**Utilization.** The scheduling algorithm must utilize the link bandwidth efficiently. This implies that the scheduling algorithm must attempt to achieve the maximal gain from the statistical multiplexing.

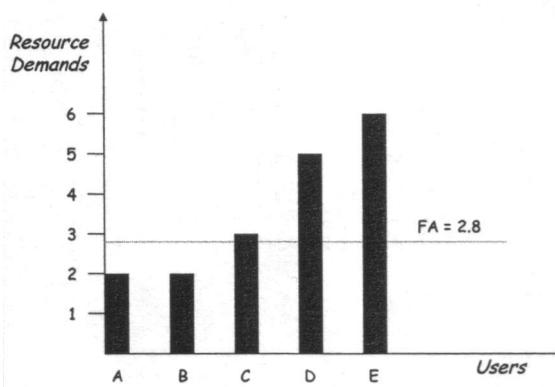
**Fairness.** The available link bandwidth must be divided among the flows sharing the link in a fair manner. Two algorithms with the same maximum delay guarantee may have significantly different fairness characteristics. An unfair scheduling algorithm may offer widely different service rates to two flows with the same reserved rate over short intervals. While there is no common accepted method for estimating the fairness of a scheduling algorithm, it is easy to define fairness in an informal manner. The system should always serve flows proportional to their reservations and distribute the unused bandwidth left behind by idle flows proportionally among the active ones. Flows should not be penalized for excess bandwidth they received while other flows were idle.

## 2.6 Max-Min Fair-Share Allocation

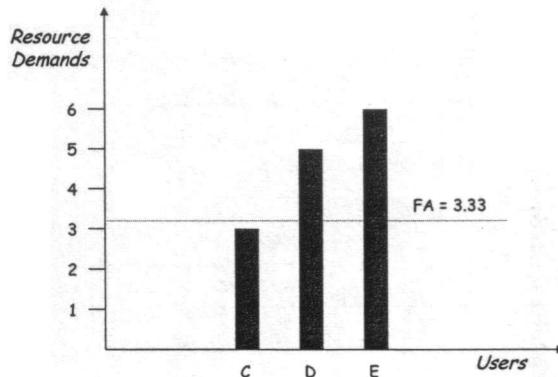
A widely accepted fair-share allocation scheme is called the *max-min fair-share* scheme. Various users' demands for a resource usually differ. So it is possible to classify users in the order of their increasing demand for a resource. The max-min fair-share allocation is defined as follows:

- resources are allocated in order of increasing demand;
- no user gets a resource share larger than its demand;
- users with unsatisfied demands get an equal share of the resource.

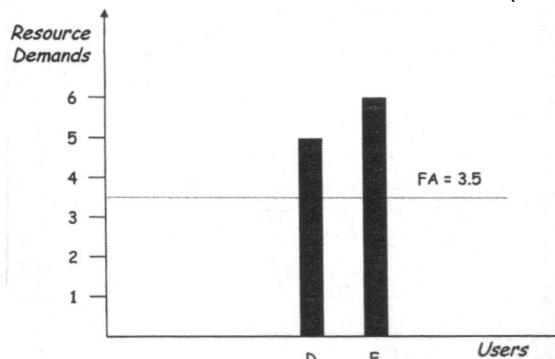
A resource has a capacity of 14, servicing five users, A, B, C, D, and E, with demands 2, 2, 3, 5, and 6, respectively.



Initially, the source with the smallest demand is given a resource equal to the resource capacity divided by the total number of users. Thus, User A and User B are given a resource of  $14/5 = 2.8$ . Since Users A and B actually need only 2, the unused excess, 1.6 (0.8 each from Users A and B), is distributed evenly among the other three users, i.e. Users C, D, and E each get a resource of  $2.8 + (1.6/3) = 3.33$ .



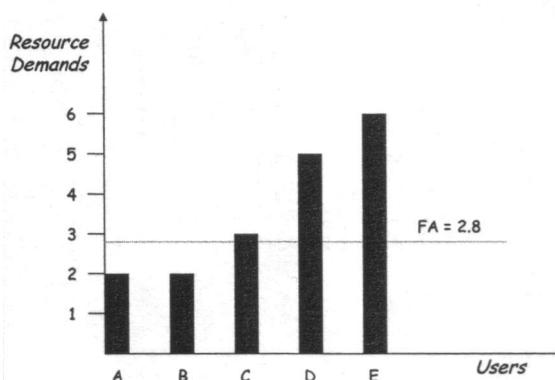
Now, User C, i.e. the user with the next-smaller demand is serviced. The resource allocated to User C is 0.33 units in excess of its demand for 3. This unused excess is distributed evenly between Users D and E so that each now has a resource of  $3.33 + (0.33/2) = 3.5$ .



We can calculate fair allocation as follows:

$$\text{Fair Allocation} = \frac{(\text{resource capacity} - \text{resource capacity already allocated to users})}{\text{number of users who still need resource allocation}}$$

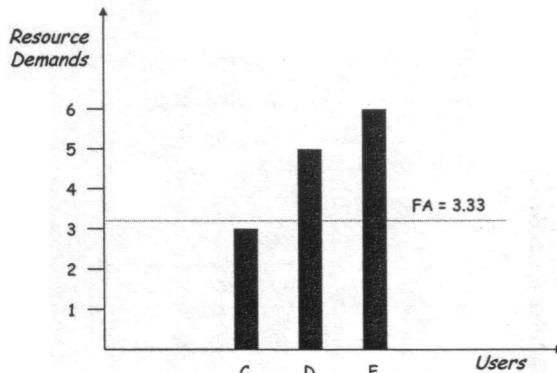
In presence of greedy users each user is given a resource equal to the resource capacity divided by the total number of users. In Step 1 the demands of Users A and B are fully allocated because their resource requests fall within the fair allocation.



$$\text{Step 1: Fair Allocation (FA)} = (14-0)/5 = 2.8$$

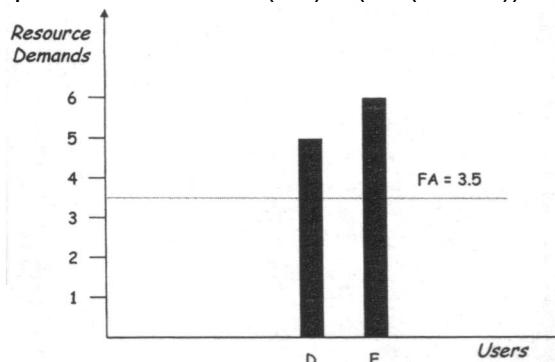
In this step, the demands of C, D, and E exceed fair allocation of 2.8 and, hence, cannot be allocated. In the next step, the unused excess bandwidth of A and B's fair allocation is equally distributed among the three remaining users, C, D, and E. In Step 2 the demand of User C is fully allocated because its resource request falls within the fair allocation.

$$\text{Step 2: Fair Allocation (FA)} = (14-(2+2))/3 = 3.3$$



In this step, the demands of D and E exceed fair allocation of 3.33 and, hence, cannot be allocated. In the next step, the unused excess bandwidth of C's fair allocation is equally distributed between the two remaining users, D and E. In Step 3 the fair allocation of 3.5 falls below the requests of both Users D and E, which are each allocated 3.5 and have unsatisfied demands of 1.5 and 2.5, respectively.

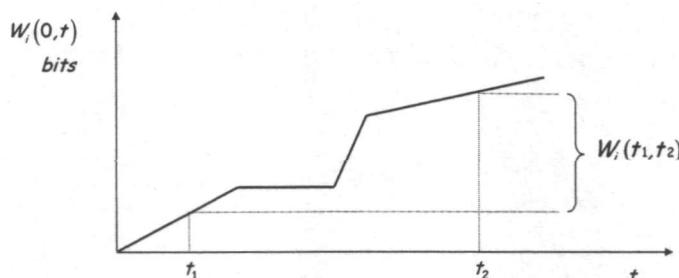
$$\text{Step 3: Fair Allocation (FA)} = (14-(2+2+3))/2 = 3.5$$



This scheme allocates resources according to the max-min fair-share scheme. Note that all users with unsatisfied demands (beyond what is their max-min fair share) get equal allocation. So, you can see that this scheme is referred to as max-min fair-share allocation because it maximizes the minimum share of a user whose demand is not fully satisfied. There is an extension to the max-min fair-share allocation scheme in which each user is assigned a weight. Such a scheme is referred to as *weighted max-min fair-share allocation*, in which a user's fair share is proportional to its assigned weight.

## 2.7 Fairness

Following Golestani's work, we define the *fairness* parameter of a scheduling algorithm as the maximum difference between the normalized service received by two backlogged connections over an interval in which both are continuously backlogged.  $W_i^S(t_1, t_2)$  denotes the service received by flow i (measured in units of traffic, e.g. bit) during the time interval  $(t_1, t_2]$  under the scheduling discipline S.



**Definition 1 (by Golestani):** Under any given scheduling discipline S, the relative fairness measured with respect to a pair of flows (i,j) which are both continuously backlogged over time interval  $(t_1, t_2]$ , denoted by  $RF_{(i,j)}(t_1, t_2)$ , is defined as:

$$RF_{(i,j)}(t_1, t_2) = \left| \frac{W_i^S(t_1, t_2)}{r_i} - \frac{W_j^S(t_1, t_2)}{r_j} \right|$$

where  $r_i$  is the service rate allocated to flow  $i$ .

$RF_{(i,j)}(t_1, t_2)$  is often referred to as *Service Fairness Index* and denoted by *SFI*.

**Definition 2:** The relative fairness with respect to a flow  $i$  over time interval  $(t_1, t_2]$ , denoted by  $RF_i(t_1, t_2)$ , is defined as:

$$RF_i(t_1, t_2) = \max_{\forall j} RF_{(i,j)}(t_1, t_2)$$

**Definition 3:** The relative fairness over time interval  $(t_1, t_2]$ , denoted by  $RF(t_1, t_2)$ , is defined as:

$$RF(t_1, t_2) = \max_{\forall i} RF_i(t_1, t_2)$$

**Definition 4:** The relative fairness bound, denoted by  $RFB$ , is defined as:

$$RFB = \max_{\forall (t_1, t_2]} RF(t_1, t_2)$$

Note that, according to definition 1, some amount of short-term unfairness between flows is inevitable in any packet-level scheduler, since each packet must be serviced exclusively. Golestani showed that the Service Fairness Index of a packet-level scheduler is lower-bounded by:

$$\frac{\left(\frac{L_i}{r_i} + \frac{L_j}{r_j}\right)}{2}$$

where  $L_i$  and  $L_j$  are the maximum packet sizes of flows  $i$  and  $j$ , respectively, and  $r_i$  and  $r_j$  are their allocated (service) rates. In other words, it is not possible to have an arbitrary small short-term unfairness in a packet scheduling algorithm, because of the atomicity of the packet transmission. The unfairness can be maintained small by good design. For example, the *WFQ* algorithm achieves a Service Fairness Index of  $O(\max(L_i/r_i))$  close to the theoretical bound. If RFB depends on  $t_2-t_1$ , the RFB is unbounded and this implies that the algorithm exhibits *long-term unfairness*, i.e. offers a persistently unfair treatment to couple of flows. This can be either because one of the flows is not serviced at all for arbitrary long period of times (i.e. it is starved), or because one of the flows is granted persistently more bandwidth than the other. A simple example of an unfair scheduler (which is also starvation-prone) is the static priority scheduler. On the other hand, algorithms for which RFB does not depend on  $t_2-t_1$  - and therefore have a bounded RFB - are commonly called *fair*. In that case, the value of the RFB is often referred to as the *short-term unfairness*. Obviously, the smaller the RFB is, the more accurate the algorithm is in sharing the capacity of the output link among the competing flows.

## 2.8 Simplicity of Implementation

In order to be feasible in high speed networks, scheduling algorithms must be simple, i.e. they must be able to take a decision within a reasonable time, in order not to represent a bottleneck. In an ATM network, the available time for completing a scheduling decision is very short. At synchronous optical network (SONET) OC-3 speeds the transmission time of a cell is less than 3  $\mu s$ . For higher speeds the available time is even less. This forces a hardware implementation. In packet networks with larger packet sizes and/or lower speeds, a software implementation may be adequate, but scheduling decisions must still be made at the rate of packet arrivals. A commonly used measure for the feasibility of a scheduling algorithm at high transmission speed is the (*worst-case*) *per-packet complexity*, i.e. the number of operations that it needs to perform for selecting a packet in a worst-case scenario. Rather than as an absolute number of operations, the per-packet complexity is often expressed as a function of the number of scheduled flows  $N$ . We talk about  $O(N)$  algorithms meaning that the number of operations that are needed in order to transmit a packet in the worst case are linearly increasing with the number of flows. Conversely, we will say that an algorithm is  $O(1)$  if that number of operations does not depend on the number of flows. The weaker the dependency from the number of flow is, the more scalable the algorithm is: an  $O(\log N)$  algorithm is likely to work faster than an  $O(N)$  one as the number of flows grows large. The per-packet complexity of an algorithm is often driven by the complexity of one or more critical subtasks. As an example, sorted-priority algorithms timestamp packets on arrival (customarily according to some time function), and sort them by increasing timestamp order. The first subtask has a

complexity which depends on the timestamping function; the second subtask, i.e. that of managing a sorted list, has an intrinsic complexity which is  $O(\log N)$ . Therefore, sorted priority algorithms are at least  $O(\log N)$  algorithms. On the other hand, neither timestamping nor sorting is necessary in a round robin scheduler: thus, schedulers in the round robin class might achieve lower per-packet complexity, down to  $O(1)$  in some cases. Note that having  $O(1)$  complexity is by no means a general characteristic of round robin schedulers: *Carry-Over Round Robin* is in fact an  $O(N)$  algorithm.

## 2.9 Scalability

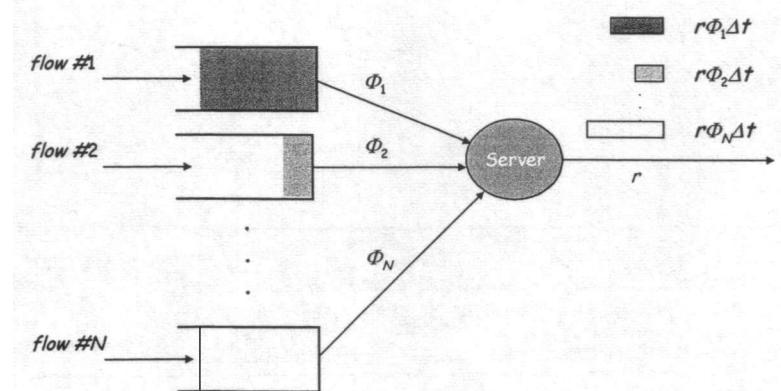
The algorithm must perform well in routers with a large number of flows, as well as over a wide range of link speeds.

## 2.10 First Road Map for Scheduling Algorithms

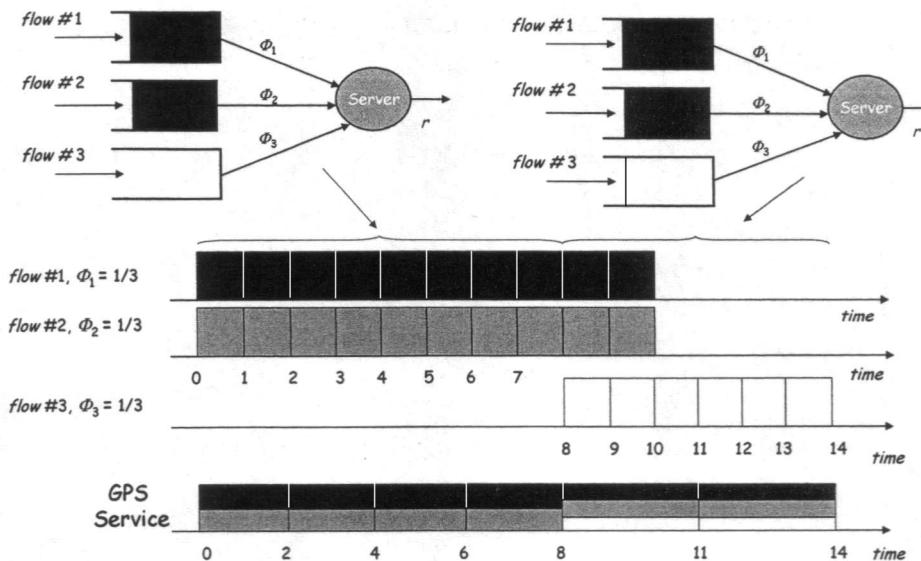
Scheduling algorithms for packet networks are generally divided into two broad categories: *work conserving* and *non-work conserving*. An algorithm is said to be *work conserving* if the server cannot refuse to start the transmission of a queued packet when the output link is idle. On the contrary, a *non-work conserving* algorithm might choose not to start the transmission of a packet even if the output link is idle. As a general rule, non-work conserving algorithms associate an *eligibility time* to a packet, either implicitly or explicitly. Only eligible packets compete for transmission on the output link, and it is therefore possible that, though there are packets queued, no packet is eligible and therefore the server is idle. We will consider work-conserving scheduling algorithms. The existing work conserving scheduling algorithms are commonly classified as *sorted-priority* or *frame-based*. *Sorted-priority* algorithms associate a timestamp to each queued packet and transmit packets by increasing timestamp order. *Frame-based* algorithms divide time into frames, and select which packet to transmit on a per-frame basis.

## 2.11 Generalized Processor Sharing (GPS)

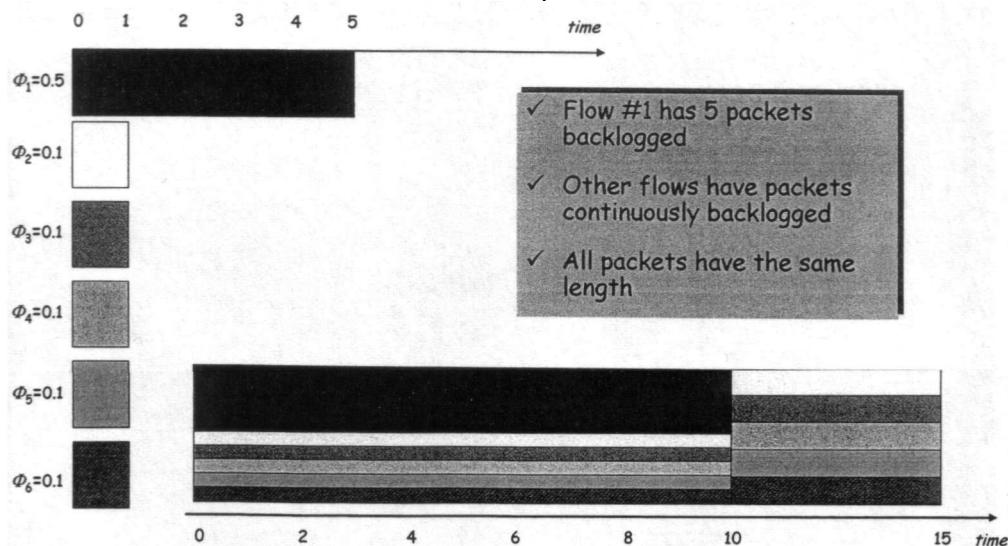
A *Generalized Processor Sharing (GPS)* is an ideal scheduling discipline. GPS multiplexing is defined with respect to a fluid model, where packets are considered to be infinitely divisible and the server can serve multiple flows simultaneously. Assume that a set of  $N$  flows  $[1, 2, \dots, N]$  share a link of capacity  $r$ . Associate to flow  $i$  a weight  $\Phi_i$  (real positive number) that determines the relative bandwidth share (resource) that flow  $i$  should receive.



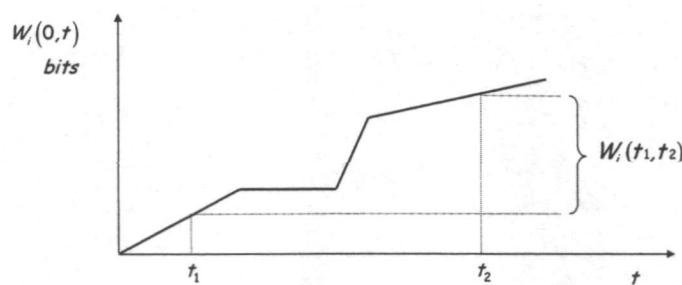
GPS Example1:



GPS Example 2:

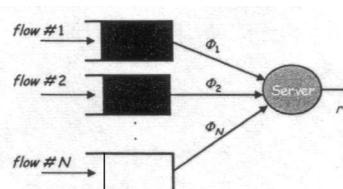


$W_i(t_1, t_2)$  indicates the amount of flow  $i$  traffic (measured in units of traffic, e.g. bit) served in an interval  $(t_1, t_2]$ .



GPS is defined as one for which:

$$\frac{W_i(t_1, t_2)}{W_j(t_1, t_2)} \geq \frac{\phi_i}{\phi_j} \quad j = 1, 2, \dots, N$$



for any flow  $i$  that is continuously backlogged in the interval  $(t_1, t_2]$ . If both flows  $i$  and  $j$  are backlogged in  $(t_1, t_2]$ , then:

$$\frac{W_i(t_1, t_2)}{W_j(t_1, t_2)} = \frac{\phi_i}{\phi_j}$$

From the previous result it follows that:

$$RFB_{i,j} = \max_{t_2 > t_1} \left| \frac{W_i(t_1, t_2)}{\phi_i} - \frac{W_j(t_1, t_2)}{\phi_j} \right| = 0$$

$$RFB = \max_{i,j} (RFB_{i,j}) = 0$$

i.e., GPS is a fair scheduling algorithm.

Summing over all flows j:

$$W_i(t_1, t_2) \sum_{j=1}^N \phi_j \geq \phi_i \sum_{j=1}^N W_j(t_1, t_2)$$

Since GPS is work conserving and since flow i is continuously backlogged in the interval  $(t_1, t_2]$ :

$$\sum_{j=1}^N W_j(t_1, t_2) = r(t_2 - t_1)$$

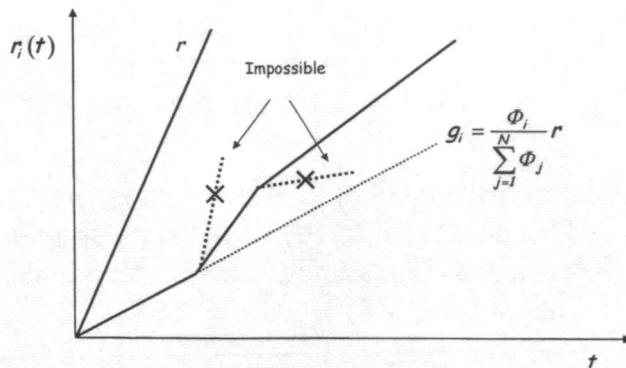
Therefore, in the time interval  $(t_1, t_2]$ :

$$W_i(t_1, t_2) \geq g_i(t_2 - t_1)$$

where:

$$g_i \stackrel{\text{def}}{=} \frac{\phi_i}{\sum_{j=1}^N \phi_j} r$$

Therefore, to flow i is guaranteed a minimum rate of  $g_i$ .



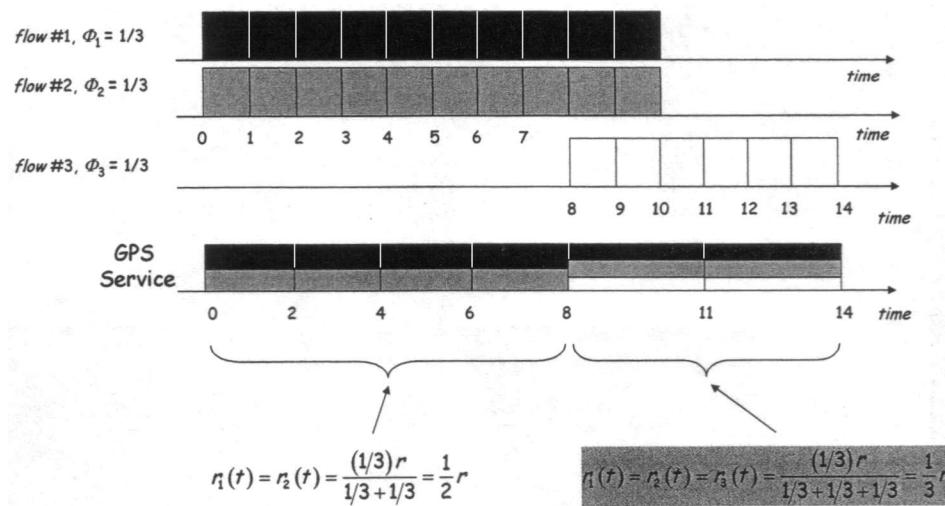
GPS serves each backlogged flow at every instant with a minimum rate. The excess bandwidth available from flows not using their reservations is distributed among all the backlogged (i.e. busy) flows  $B(t)$  at time  $t$  in proportion to their individual weights. Define  $r_i(t)$  to be the flow i rate at time  $t$  (*instantaneous rate*):

$$r_i(t) = \frac{\phi_i}{\sum_{j \in B(t)} \phi_j} r \geq \frac{\phi_i}{\sum_{j=1}^N \phi_j} r = g_i \Rightarrow r_i(t) \geq g_i$$

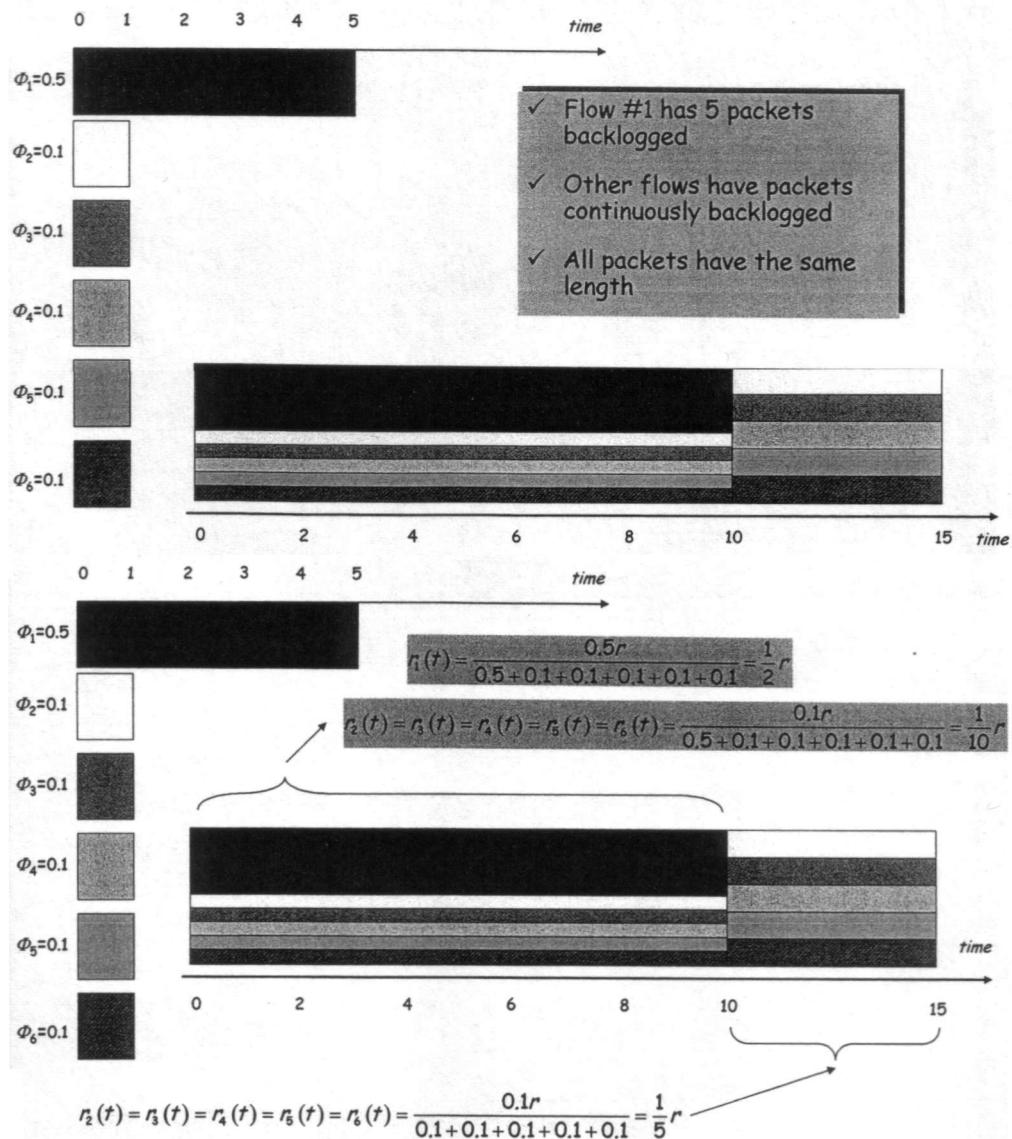
If the number of backlogged flows remains constant during a time interval  $(t, t + \Delta t]$ , then flow i is entitled to transmit  $r_i(t)\Delta t$  traffic units (e.g. bits) in that time interval. When the number of backlogged flows varies over time, the amount of traffic units (e.g. bits) that flow i should transmit in an ideal fluid-flow system while being active during a time interval  $(t_0, t_1]$  is:

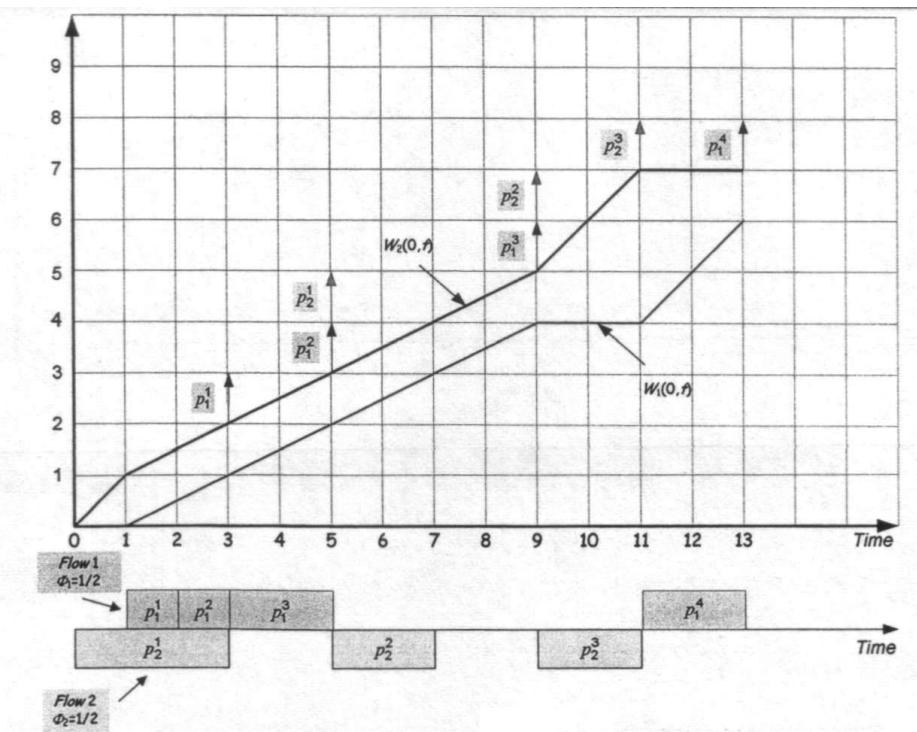
$$W_i(t_0, t_1) = \int_{t_0}^{t_1} r_i(t) dt$$

GPS Example 1:



### GPS Example 2:





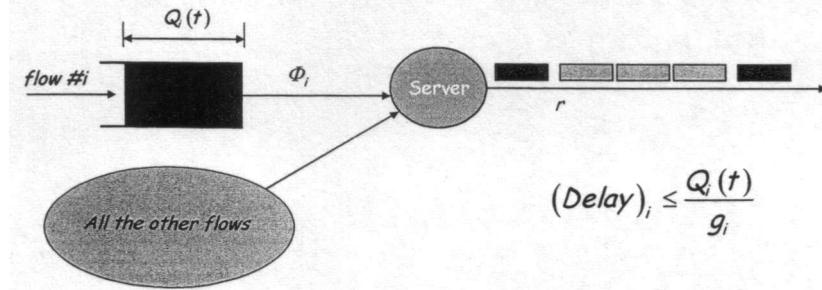
## 2.12 GPS Properties

**First.** Define  $r_i$  to be the service rate allocated to flow  $i$ . Then, as long as

$$r_i \leq g_i = \frac{\phi_i}{\sum_{j=1}^N \phi_j} r$$

the flow can be guaranteed a rate  $r_i$  independent of the demands of the other flows. In addition to this rate guarantee, a flow  $i$  backlogged will always be cleared at a rate  $r_i(t) \geq g_i$ .

**Second.** The delay of an arriving flow  $i$  bit can be bounded as a function of the flow  $i$  queue length, independent of the queues and arrivals of the other flows (take into consideration that to flow  $i$  is guaranteed a rate). Schemes such as FCFS and LCFS do not have this property.



**Third.** By varying the  $\Phi$ 's we have the flexibility of treating the flows in a variety of different ways:

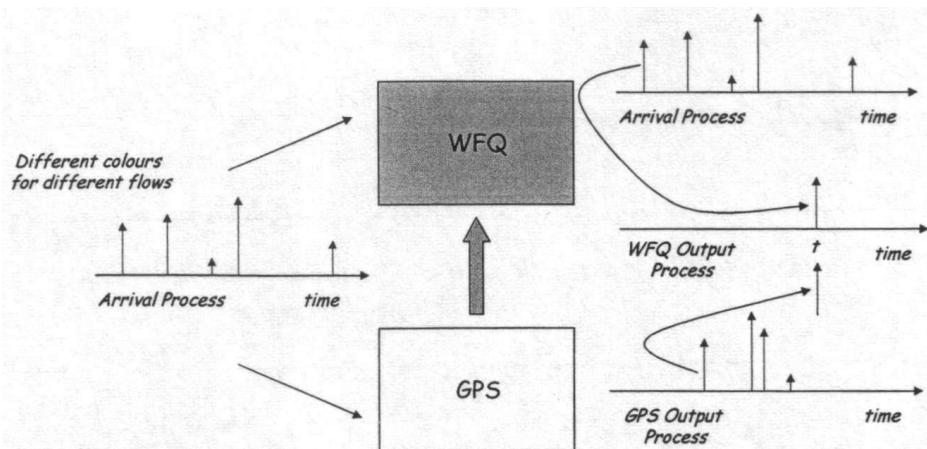
1. When all  $\Phi$ 's are equal, the system reduces to *uniform processor sharing*.
2. As long as the combined rate of the flows is less than  $r$ , any assignment of positive  $\Phi$ 's yields a stable system.
3. A delay-insensitive flow  $i$  can be assigned  $g_i$  much less than its average rate, thus allowing for better treatment of the other flows.

**Fourth.** It is possible to make worst-case network queuing delay guarantees when the sources are constrained by token buckets. GPS is particularly attractive for flows sending real-time traffic such as voice and video.

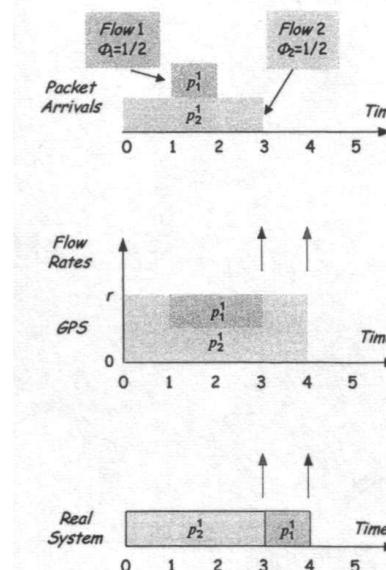
## 2.13 Approximating GPS with WFQ

GPS is defined in an idealized fluid model: it assumes that the server can serve multiple flows simultaneously and that the traffic is infinitely divisible. Real systems are packet systems: one queue is served at any given time and packet transmission is not preempted. The goal is to define

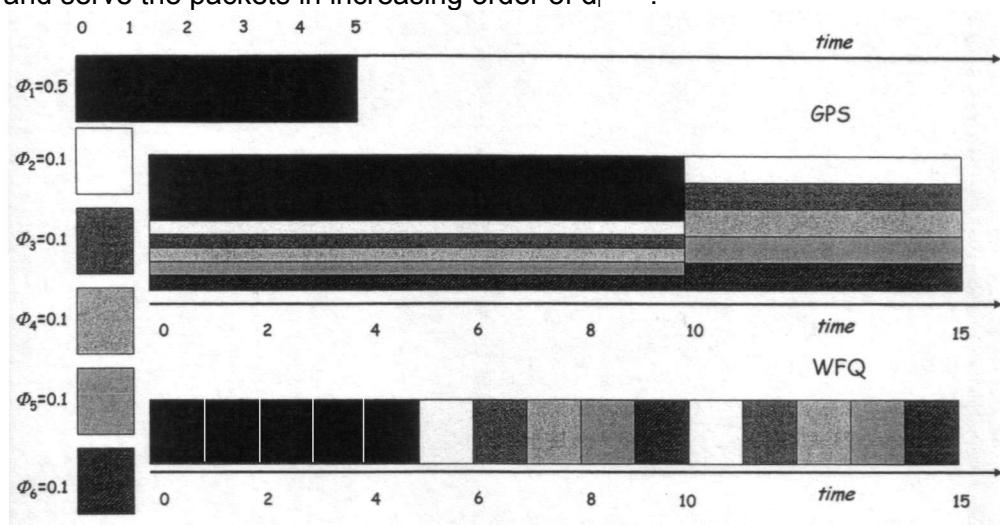
packet algorithms approximating the fluid system that maintain most of the important GPS properties.

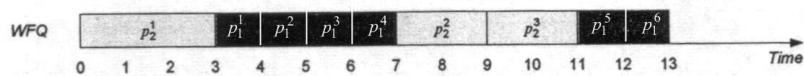
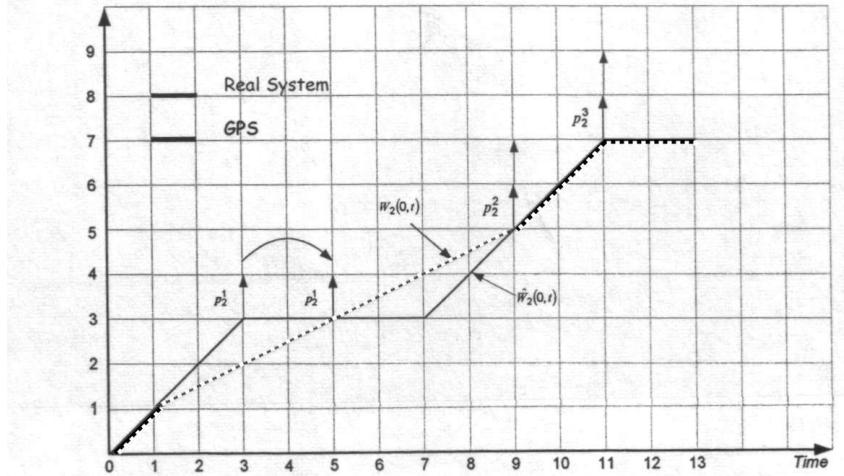
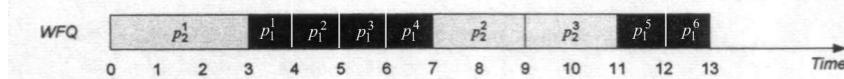
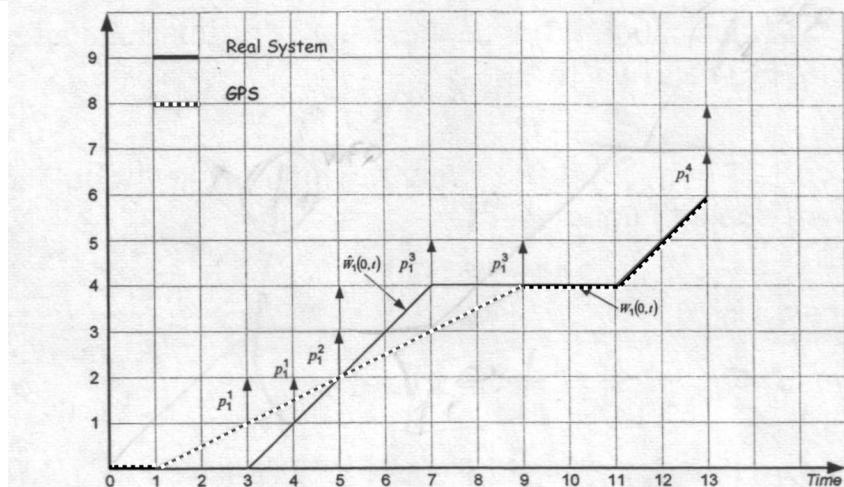
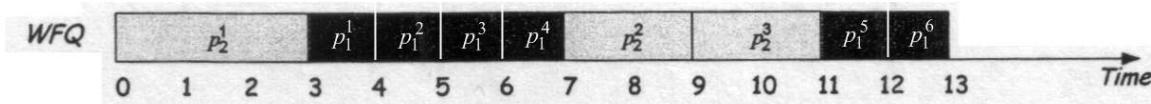
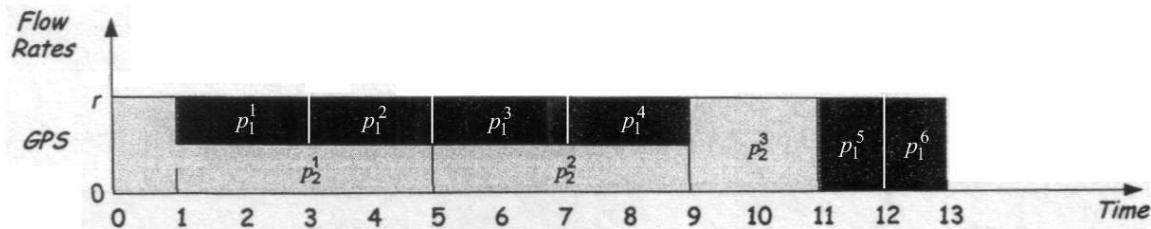
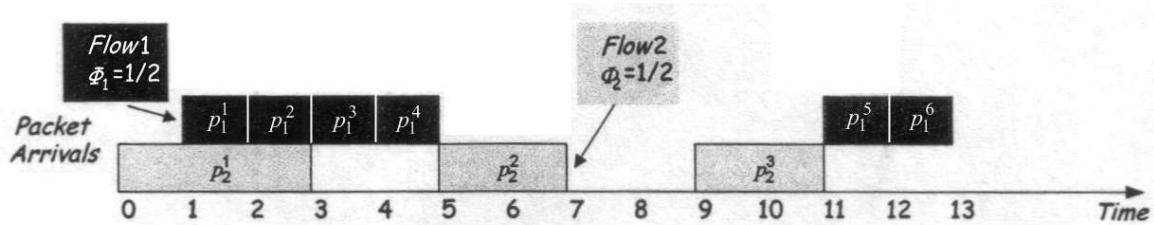


When the server (i.e. router) is ready to transmit the next packet at time  $t$ , it picks, among all the packets queued in the system at  $t$ , the first packet that would complete service in the corresponding GPS system if no additional packets were to arrive after time  $t$ . This scheduling algorithm is called *Packet-by-Packet GPS (PGPS)* or *Weighted Fair Queueing (WFQ)*. The scheduling algorithm is called *Fair Queueing (FQ)* when all the weights coincide. More formally, let  $d_i^{k,\text{GPS}}$  be the time at which packet  $k$  belonging to flow  $i$  will depart (finish service) under GPS. A good approximation of GPS would be a work-conserving scheme that serves packets in increasing order of  $d_i^{k,\text{GPS}}$ . Assume that the server becomes free at time  $t=0$ :



The next packet to depart under GPS may not have arrived at time  $t$  and, since the server has no knowledge of when this packet will arrive, there is no way for the server to be both work conserving and serve the packets in increasing order of  $d_i^{k,\text{GPS}}$ .



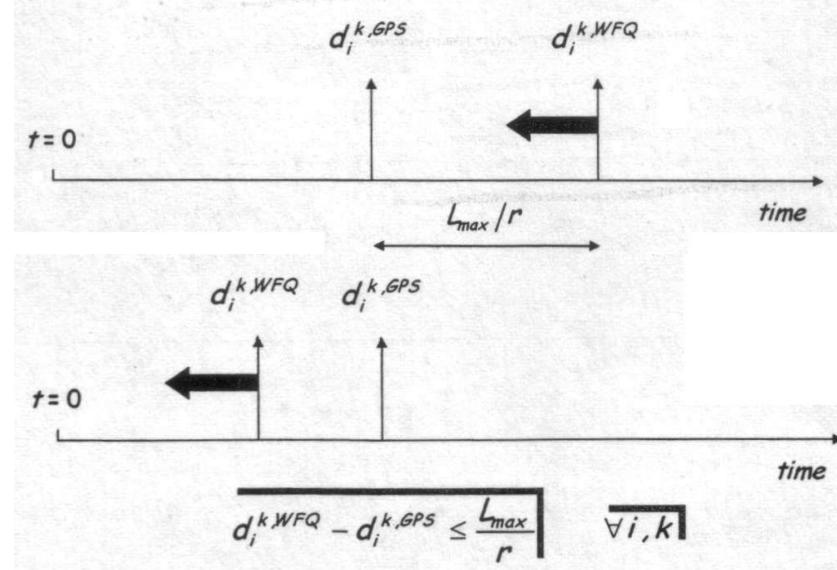


*Question:* how much later packets may depart the system under WFQ relative to GPS? *Answer:* Let  $d_i^{k,WFQ}$  be the time at which packet k belonging to flow i departs under WFQ.

*Parekh Theorem:*

$$d_i^{k,WFQ} - d_i^{k, GPS} \leq \frac{L_{max}}{r} \quad \forall i, k$$

where  $L_{max}$  is the maximum packet length among all flows and r is the transmission rate of the server. What Parekh has proven is that WFQ cannot fall behind GPS with respect to the service given to a flow by one maximum-size packet.



## 2.14 Time Complexity of WFQ

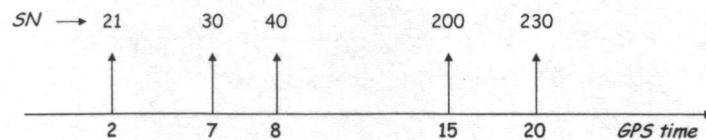
As already said, WFQ sorts packets in the same order in which they would leave under GPS. Therefore, in order to implement them, a reference GPS scheduler must be emulated. However, given a time instant t, it is not possible to compute a leaving time of a packet under GPS in the future, since the actual rate at which GPS services a flow depends on the set  $B(t)$  which in turn might change in response to future packet arrivals. Therefore, in order to compute the leaving order of packets under GPS, every time a flow becomes idle or backlogged, all backlogged flows should have their instantaneous service rates recomputed and the leaving time of their head-of-line packet should be updated accordingly. This implies computing  $O(N)$  quantities  $O(N)$  times per packet transmission in a worst case, i.e.  $O(N^2)$  per-packet complexity. Such a cost is clearly unfeasible in a high-speed network if more than a bunch of flows are scheduled.

## 2.15 SN Computation-Based WFQ

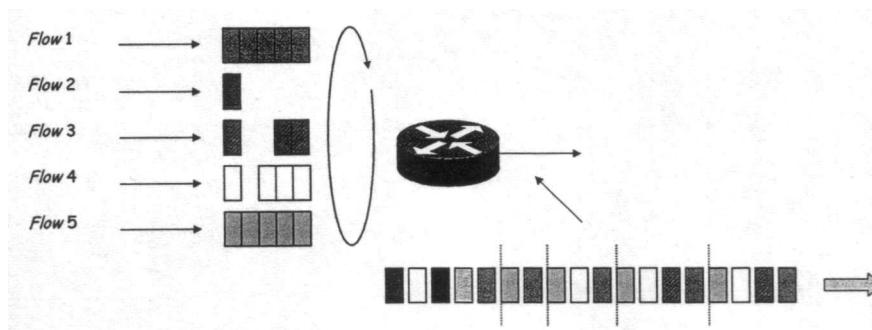
Techniques of approximating fluid GPS select packet that will finish first in GPS assuming that there are no future arrivals.

1. Implementation based on *Sequence Number* (SN) computation assigns sequence number to each packet upon arrival. Packets served in increasing order of sequence numbers.
2. Implementation based on *Virtual Time* computation assigns virtual finish time to each packet upon arrival. Packets served in increasing order of virtual times.

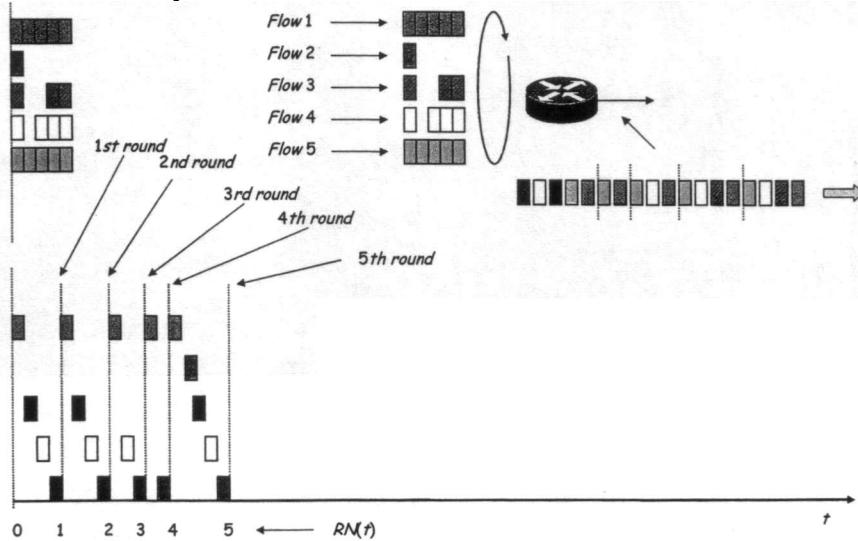
FQ simulates GPS by computing a sequence number for each arriving packet. The assigned *Sequence Numbers* (SN) are essentially service tags, which define the relative order in which the packets are to be serviced. The service order of packets using sequence number computation emulates the service order of a GPS scheduler.



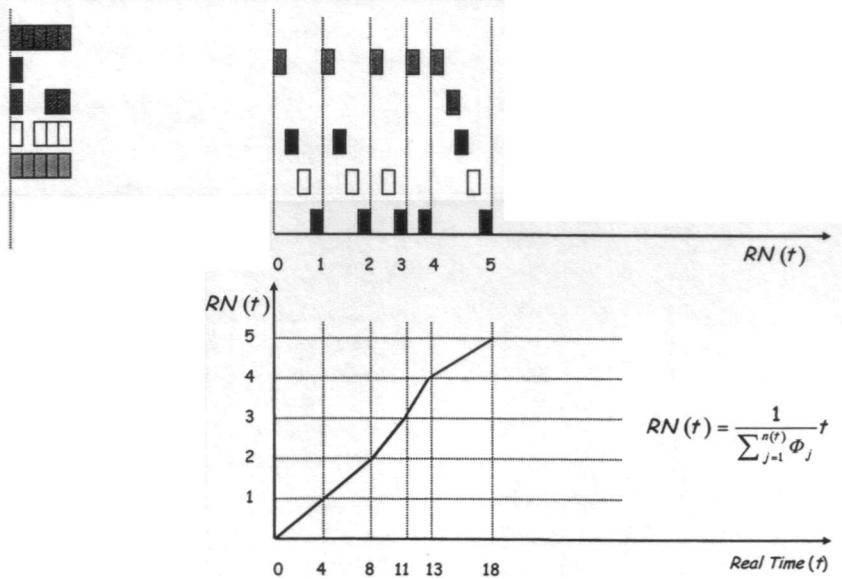
To understand how FQ simulates GPS let's look at the GPS fluid flow system as if it were a byte-by-byte round-robin; that is, the router transmits one byte from flow 1, then one byte from flow 2, and so on (all  $\Phi$ 's are assumed to be equal to 1).



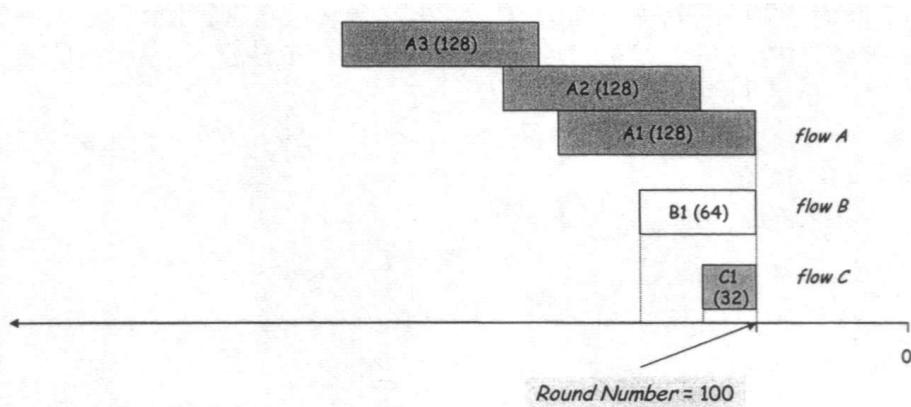
Consider the behavior of a single flow (pink flow that we call flow  $i$ ) and imagine a virtual clock that ticks once each time one byte is transmitted from all of the active flows.



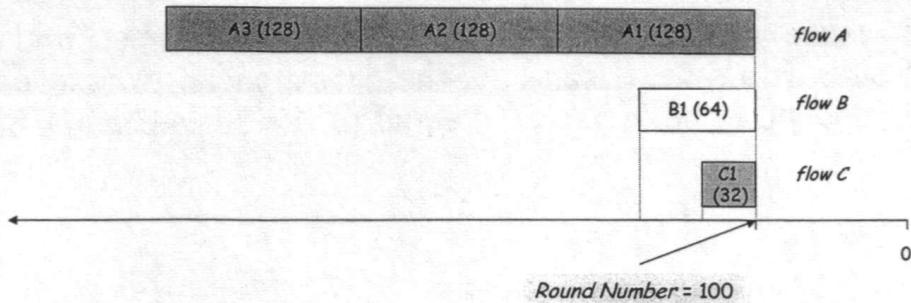
Denote by  $RN(t)$  (where  $RN$  stands for *Round Number*), the *Virtual Time* measured by the virtual clock at a given (real) time  $t$ .  $RN(t)$  indicates the number of rounds (or clock ticks) of service a byte-by-byte round-robin scheduler has completed at a given real time  $t$  (we assume that it takes one real time unit to transmit one byte).  $RN(t)$  changes slope anytime the number of active flows  $n(t)$  changes; i.e. any time a flow becomes backlogged or unbacklogged. The SN of a packet is the round in which the last byte of the packet is transmitted.



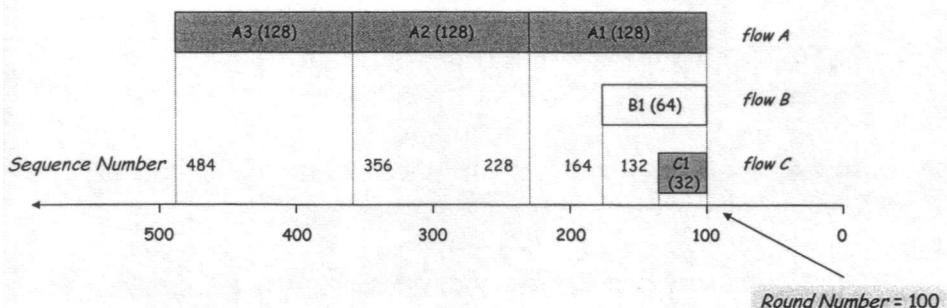
To illustrate how GPS is simulated by FQ, consider three flows, A, B, and C, with packet sizes 128, 64, and 32 bytes, respectively. Assume that all  $\phi$ 's are equal to 1. Packets A1, B1, and C1 arrive on inactive flows.



Packets A2, will be served after packet A1 is served and packet A3 will be served after packet A2 is served. The SN of a packet is the round in which the last byte of the packet is transmitted.



Assuming service by a byte-by-byte round-robin scheduler, an entire 128-byte packet is sent when the scheduler completes 128 rounds of service since the packet arrived. If the round number at the time Packet A1 arrived is 100, the entire packet is transmitted when the round number becomes  $100 + 128 = 228$ . The SN of a packet is the round in which the last byte of the packet is transmitted. Therefore, the sequence number of a packet for an inactive flow is calculated by adding the round number and the packet size in bytes. Because, in reality, a scheduler transmits a packet and not 1 byte at a time, it services the entire packet whenever the round number becomes equal to the sequence number. When Packet A2 arrives, the flow is already active with A1 in the queue, waiting for service, with a sequence number of 228, thus the sequence number of Packet A2 is  $228 + 128 = 356$ , because it needs to be transmitted after A1. Hence, the SN of a packet arriving on an active flow is the highest sequence number of the packet in the flow queue, plus its packet size in bytes. Packet A3 gets a sequence number of  $356 + 128 = 484$ . Because Packets B1 and C1 arrive on an inactive flow, their sequence numbers are 164 (that is,  $100 + 64$ ) and 132 (that is,  $100 + 32$ ), respectively.

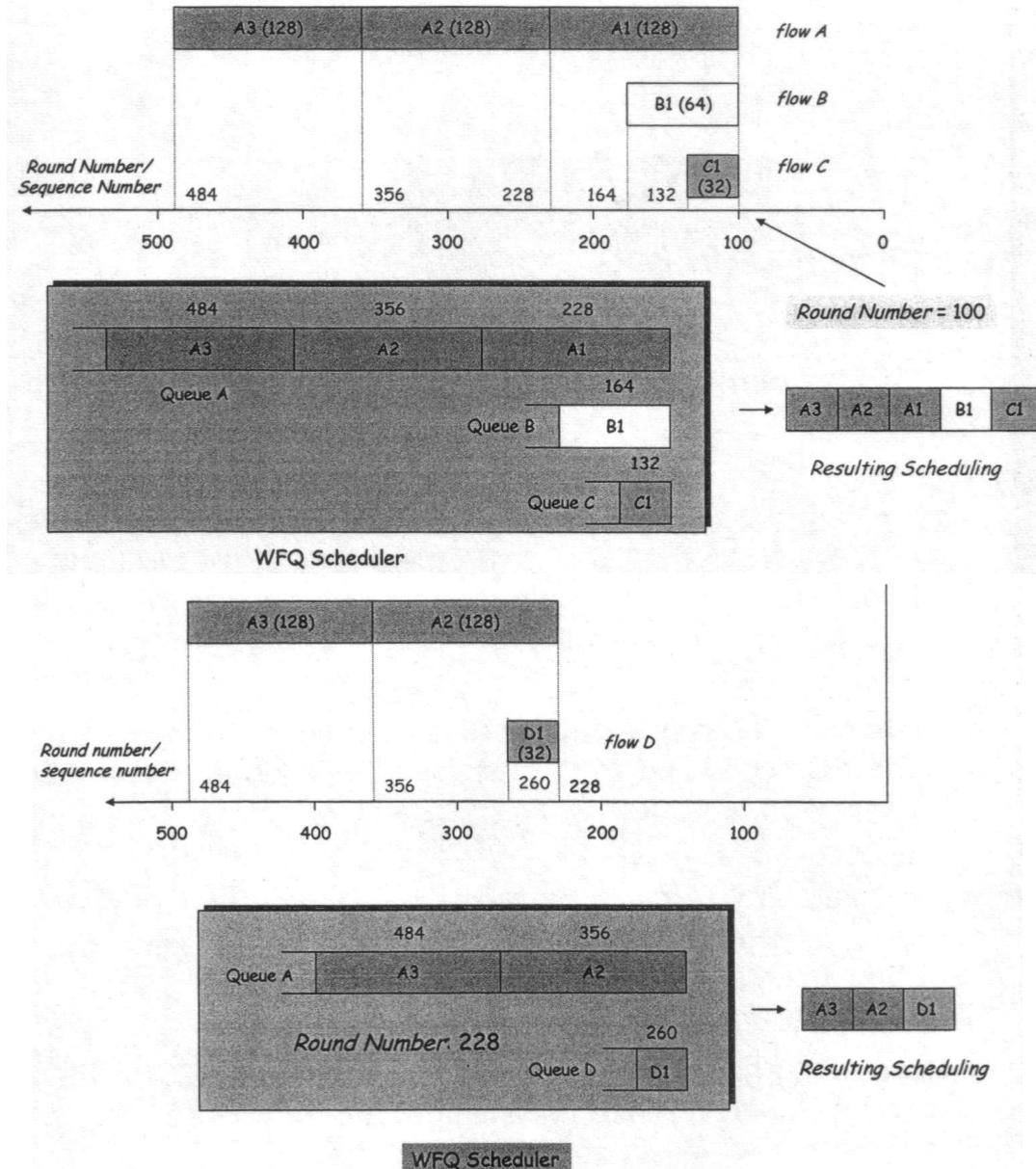


Sequence Number (SN) assignment for a packet is summarized based on whether it arrives on an active or an inactive flow as follows:

- Packet arrives on an inactive flow:  
SN= Size of the packet in bytes + the RN at the time the packet arrived.
- Packet arrives on an active flow:  
SN= Size of the packet in bytes + the highest SN of the packet already in the flow queue.

A GPS scheduler will have completed scheduling the entire Packet A1 in the 228th round. The sequence number denotes the relative order in which the packets are served by the scheduler. The FQ scheduler serves the packets in the following order: C1, B1, A1, A2, and A3. Round numbers are used only for calculating sequence numbers if the arriving packet belongs to an inactive flow. Otherwise, the sequence number is based on the highest sequence number of a packet in that flow awaiting service. If Packet A4 arrives at any time before A3 is serviced, it has a sequence number of  $484 + 128 = 612$ . Note that the round number is updated every time a packet is scheduled for

transmission to equal the sequence number of the packet being transmitted. So if Packet D1 of size 32 bytes, belonging to a new flow, arrives when A1 is being transmitted, the round number is 228 and the sequence number of D1 is 260 (228 + 32). Because D1 has a lower sequence number than A2 and A3, it is scheduled for transmission before A2 and A3.



$$p_i^k = k_{th} \text{ packet belonging to flow } i$$

$$a_i^k = \text{arrival time of } p_i^k$$

FQ Algorithm:

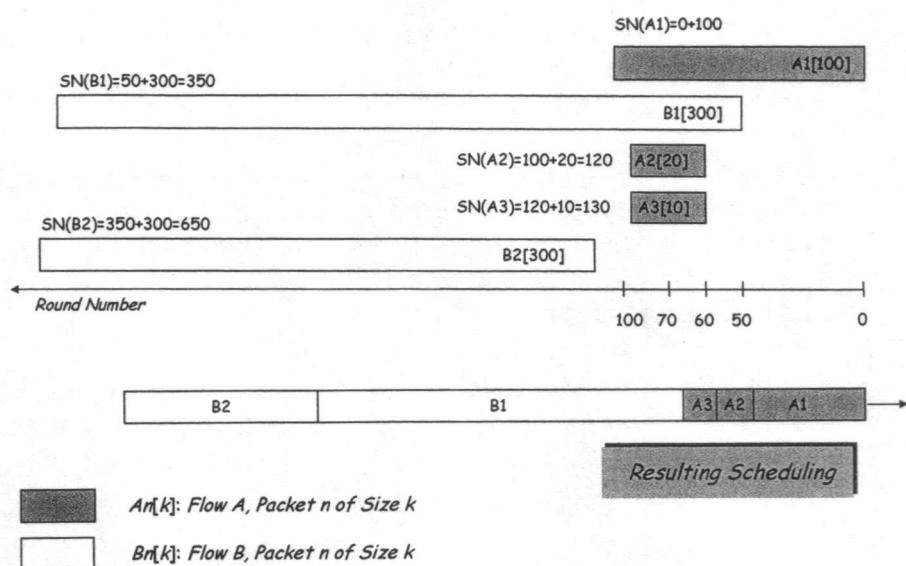
If flow  $i$  is active

$$\text{then: } SN(p_i^k) = SN(p_i^{k-1}) + \text{Size}(p_i^k)$$

$$\text{otherwise: } SN(p_i^k) = SN(a_i^k) + \text{Size}(p_i^k)$$

That is:

$$SN(p_i^k) = \max[SN(p_i^{k-1}), RN(a_i^k)] + \text{Size}(p_i^k)$$



In general different weights are assigned to each flow (queue). A weight logically specifies how many bytes to transmit each time the router services that queue, which effectively controls the percentage of the link's bandwidth that flow will get. Simple FQ gives each queue a weight of 1, which means that logically only 1 byte is transmitted from each queue each time around. With WFQ, however, one queue might have a weight of 2, a second queue might have a weight of 1, and a third queue might have a weight of 3. Assuming that each queue always contains a packet waiting to be transmitted, the first flow will get one-third of the available bandwidth, the second will get one-sixth of the available bandwidth, and the third will get one-half of the available bandwidth. While we have described WFQ in terms of flows, note that it could be implemented on classes of traffic, where classes are defined in some other way than the simple flows introduced at the start of this chapter. For example, we could use the Type of Service (TOS) bits in the IP header to identify classes, and allocate a queue and a weight to each class. This is exactly what is proposed as part of the Differentiate Services architecture. Note that a router performing WFQ must learn what weights to assign to each queue from somewhere, either by manual configuration or by some sort of signalling from the sources.

WFQ Algorithm:  
If flow i is active

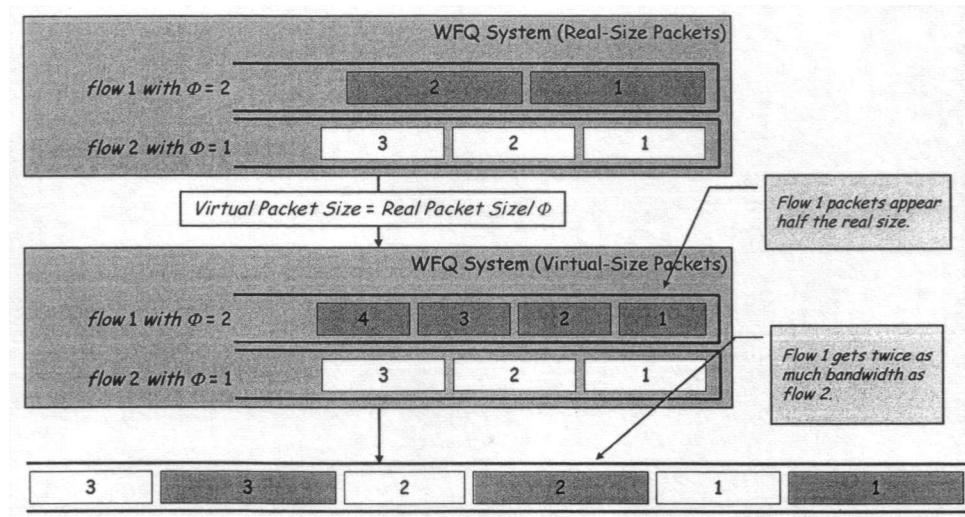
$$\text{then: } SN(p_i^k) = SN(p_i^{k-1}) + \frac{\text{Size}(p_i^k)}{\phi_i}$$

$$\text{otherwise: } SN(p_i^k) = SN(a_i^k) + \frac{\text{Size}(p_i^k)}{\phi_i}$$

That is:

$$SN(p_i^k) = \max[SN(p_i^{k-1}), RN(a_i^k)] + \frac{\text{Size}(p_i^k)}{\phi_i}$$

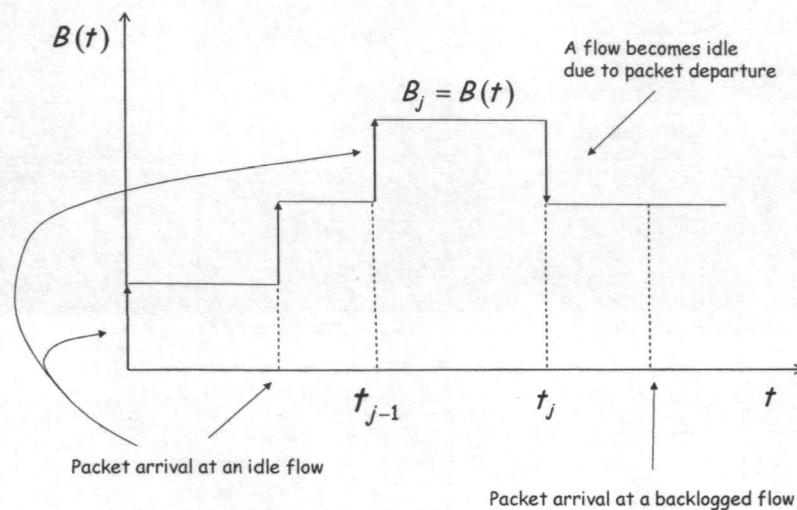
Resulting Scheduling:



## 2.16 Virtual Time Definition<sup>1</sup>

Denote event each arrival and departure from the GPS server. Let  $t_j$  the time at which the  $j^{\text{th}}$  event occurs (simultaneous events are ordered arbitrarily). Let the time of the first arrival in a busy period be denoted as  $t_1=0$ . For each  $j = 2, 3, \dots$  the set of flows that are busy in the interval  $(t_{j-1}, t_j]$  is fixed, and we denote this as:

$$B_j = B(t), \forall t \in (t_{j-1}, t_j]$$



Consider any busy period, and let the time it begins be time zero.

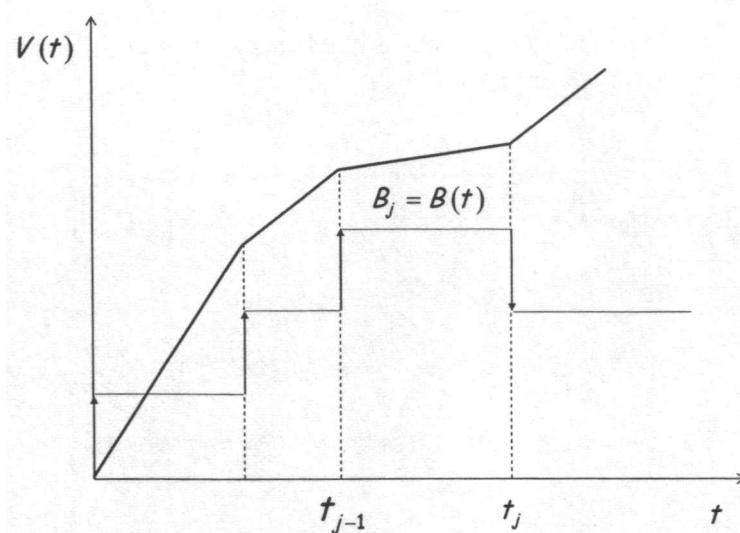
$$V(0)=0$$

$$V(t_{j-1} + \tau) = V(t_{j-1}) + \frac{r\tau}{\sum_{i \in B_j(\tau)} \phi_i} \quad \forall \tau \leq t_j - t_{j-1}$$

$$\Rightarrow \frac{\partial V(t_{j-1} + \tau)}{\partial \tau} = \frac{r}{\sum_{i \in B_j(\tau)} \phi_i}$$

$$r_i(\tau) = \frac{r \phi_i}{\sum_{j \in B(\tau)} \phi_j} = \phi_i \frac{\partial V(t_{j-1} + \tau)}{\partial \tau} \quad \forall \tau \leq t_j - t_{j-1}$$

<sup>1</sup> Gli ultimi passaggi algebrici del paragrafo contengono inesattezze nel testo originale. Qui sono state corrette.



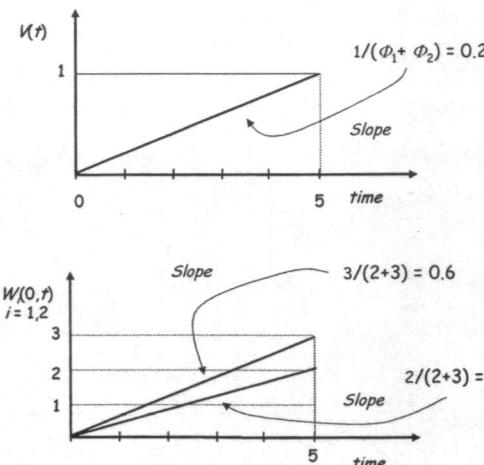
Therefore, the system virtual time at time  $t$  can be expressed:

$$V(t) = r \int_0^t \frac{d\tau}{\sum_{j \in B(\tau)} \phi_j}$$

The virtual time increases at a rate inverse proportional to the sum of the weights of all backlogged flows. When the competition increases the virtual time slows down, while when the competition decreases it accelerates. The amount of traffic units that a backlogged flow  $i$  should transmit in the interval  $(t_0, t_1]$  is:

$$W_i(t_0, t_1) = \int_{t_0}^{t_1} r_i(t) dt = \phi_i r \int_{t_0}^{t_1} \frac{dt}{\sum_{j \in B(t)} \phi_j} = \phi_i [V(t_1) - V(t_0)]$$

Intuitively, the flow of the virtual time changes to "accommodate" all backlogged flows in one virtual time unit. That is, the size of a virtual time unit is modified such that in the corresponding fluid-flow system each backlogged flow  $i$  receives  $\Phi_i$  real time units during one virtual time unit. Consider two flows with weights  $\Phi_1 = 2$ , and  $\Phi_2 = 3$ .



Then the rate at which the virtual time increases is  $1/(\Phi_1 + \Phi_2) = 0.2$ . Therefore, if  $r = 1$ , a virtual time unit equals five real-time units:

$$V(t) - V(0) = 1 = \int_0^t 0.2 d\tau = 0.2t \Rightarrow t = 5$$

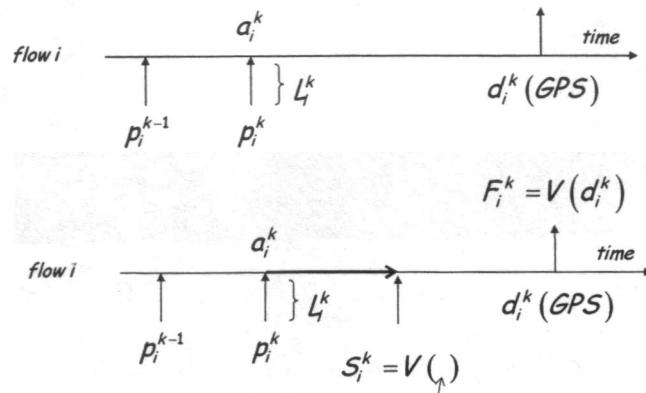
Thus, in each virtual time unit the two flows should receive  $\Phi_1 = 2$ , and  $\Phi_2 = 3$  real time units.

## 2.17 Lemma

Consider a busy period beginning at  $t = 0$ , a flow  $i$  and a sequence of packets of  $i$  which arrive during this busy period. Definitions:

- $p_i^k = k^{\text{th}}$  packet of the sequence

- $a_i^k$  = arrival time of  $p_i^k$
- $d_i^k$  = the time  $p_i^k$  finishes service in the GPS system
- $L_i^k$  = length of  $p_i^k$



- $S_i^k$  = virtual time at which  $p_i^k$  begins service (also called virtual start time)
- $F_i^k$  = virtual time at which  $p_i^k$  completes service, i.e.  $f_i^k = V(d_i^k)$   
(also called virtual finish time)

**Lemma 1:** The virtual finishing times  $F_i^k$ ,  $k=1,2,3,\dots$  associated to packets of each flow  $i$  inside  $B(t)$ , satisfy the following relationship:

$$\begin{aligned} S_i^k &= \max\{F_i^{k-1}, V(a_i^k)\} & k=1,2,\dots \\ F_i^k &= S_i^k + \frac{L_i^k}{\phi_i} & i \in B(t) \end{aligned}$$

where:  $F_i^0 = 0$ ,  $i \in B(t)$

Note the comparison between **Sequence Number** computation and **Virtual Time** computation:

SN:

$$SN(p_i^k) = \max[SN(p_i^{k-1}), RN(a_i^k)] + \frac{\text{Size}(p_i^k)}{\phi_i}$$

VT:

$$F_i^k = \max\{F_i^{k-1}, V(a_i^k)\} + \frac{L_i^k}{\phi_i}$$

## 2.18 Proof

Let:

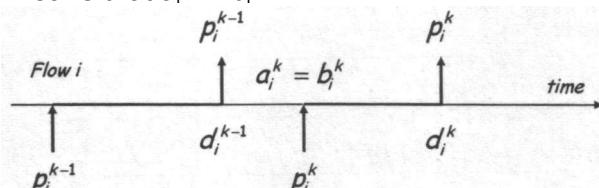
$$b_i^k = \max\{a_i^k, d_i^{k-1}\} \quad k=1,2,3,\dots$$

where, for consistency:

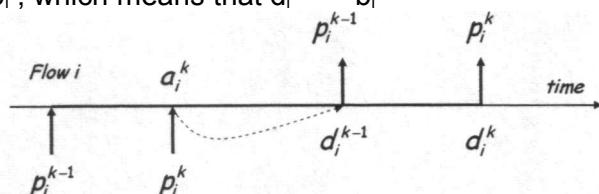
$$d_i^0 = 0$$

Service of packet  $p_i^k$  cannot start before  $b_i^k$  since, either

- $p_i^k$  arrives at  $b_i^k$ , which means that  $a_i^k = b_i^k$



- $p_{i-1}^{k-1}$  is in service until  $b_i^k$ , which means that  $d_{i-1}^{k-1} = b_i^k$



Moreover, all of the previous packets of flow i are completely served by  $b_i^k$ .

2 La dimostrazione contiene alcune inesattezze nel testo originale. Qui sono state corrette.

$$W_i(b_i^k, d_i^k) = \int_{b_i^k}^{d_i^k} r_i(t) dt = \phi_i r \int_{b_i^k}^{d_i^k} \frac{dt}{\sum_{j \in B(t)} \phi_j} = \phi_i [V(d_i^k) - V(b_i^k)]$$

Since flow  $i$  is constantly backlogged during  $(b_i^k, d_i^k]$ :

$$L_i^k = W_i(b_i^k, d_i^k)$$

and therefore:

$$\frac{L_i^k}{\phi_i} = [V(d_i^k) - V(b_i^k)]$$

Let's consider again:

$$b_i^k = \max \{a_i^k, d_i^{k-1}\} \quad k=1,2,3,\dots$$

Since, during the busy period,  $V(t)$  is monotonically increasing, it follows:

$$V(b_i^k) = \max \{V(a_i^k), V(d_i^{k-1})\} \quad k=1,2,3,\dots$$

Thus, by assuming  $r=1$ :

$$V(d_i^k) = \frac{L_i^k}{\phi_i} + \max \{V(a_i^k), V(d_i^{k-1})\} \quad k=1,2,3,\dots$$

Since, by definition:

$$F_i^k = V(d_i^k) \quad k=1,2,3,\dots$$

it follows:

$$F_i^k = \frac{L_i^k}{\phi_i} + \max \{V(a_i^k), F_i^{k-1}\} \quad k=1,2,3,\dots$$

Now, because:

$$S_i^k = \max \{F_i^{k-1}, V(a_i^k)\} \quad F_i^0 = 0 \quad \forall i \in B(t) \quad k=1,2,3,\dots$$

The lemma is proved.

## 2.19 Virtual Time Implementation

Order-preserving property of GPS guarantees that WFQ can be implemented by the "Smallest virtual Finish time First" (SFF) policy. Need to keep per flow instead of per packet virtual start, finish time only. System virtual time is used to reset a flow's virtual start time when a flow becomes backlogged again after being idle. The best known implementation method for the WFQ scheme follows from Lemma 1. Since  $V(t)$  is an increasing function of time during a busy period, it would be identical to order the packets in terms of the corresponding finishing times  $d_i^k$  or to order them in terms of the corresponding virtual finishing times  $F_i^k = V(d_i^k)$ . Hence, in the WFQ system, each time the server becomes free, that packet is picked up for transmission which has the smallest virtual finishing time in the GPS system, among the packets present in the queue. Lemma 1 provides an iterative algorithm for computing virtual finishing time of a packet in terms of the:

- packet length,
- virtual finishing time of the previous packet of the same flow, and
- system's virtual time when the packet arrives.

$$\begin{aligned} S_i^k &= \max \{F_i^{k-1}, V(a_i^k)\} & k=1,2,\dots \\ F_i^k &= S_i^k + \frac{L_i^k}{\phi_i} & i \in B(t) \end{aligned}$$

where:  $F_i^0 = 0, i \in B(t)$

Therefore, the WFQ scheme may be implemented by stamping each packet, upon arrival into the queue, with a service tag equal to the corresponding virtual finishing time, and then serving the packets from the queue in increasing order of the associated service tags. Thus, the implementation of the WFQ scheme requires the evaluation (in real-time) of the virtual time  $V(t)$  of the GPS system.  $V(t)$  is a piecewise linear function with its slope at any point of time  $t$  inversely proportional to the sum of the weight of flows in the set  $B(t)$ . Whenever some flow  $k$  becomes backlogged or ceases to be backlogged in the GPS system, the slope of  $V(t)$  changes, constituting a breakpoint in its piecewise linear form. Therefore, evaluation of  $V(t)$  is conceptually simple; it requires keeping track of the sets  $B(t)$  and its evolution in time. However, from a practical standpoint, the computational complexity associated with the evaluation of  $V(t)$  depends on the frequency of breakpoints in  $V(t)$ , i.e., the frequency of transitions in and out of the set  $B(t)$ . Unfortunately, while such transitions can be rather infrequent on the average, occasionally, a large

number of them could happen during a single packet transmission time. The reason for this peculiar phenomenon is that in the GPS system, where  $V(t)$  is to be determined, packets are not served one after another; instead one packet from every backlogged flow is in service, simultaneously. It is therefore possible that many packets finish service almost simultaneously, but not at exactly the same time. With each packet finishing service, the corresponding flow could become absent, should there be no other packet from that flow in the queue, thereby forming a new breakpoint in  $V(t)$ . We conclude that, in general, the number of breakpoints in  $V(t)$  in an arbitrarily short period of time, can approach the total number of flows ( $N$ ) set up on the transmission link. To state this requirement more accurately, in the WFQ system, let a packet be picked up for service at  $t_1$  and finish service at  $t_2$ . At  $t_2$ , in order to select the next packet for service, the virtual finishing time of packets arrived during  $(t_1, t_2]$  must be known. Therefore, evaluation of  $V(t)$  for the interval  $(t_1, t_2]$  must be completed by  $t_2$ . According to the previous arguments, the number of breakpoints in  $V(t)$  during  $(t_1, t_2]$  can be as high as the total number of flows crossing the system (switch, router, etc.). Therefore, the time complexity of WFQ is  $O(N)$  because of the overhead in keeping track of  $B(t)$ , where  $N$  is the maximum number of backlogged flows in the system.

## 2.20 Fairness of a WFQ Scheduler

For a WFQ scheduler:

$$RFB^{(WFQ)} = \max_{i,j} \left[ \max \left( C_j + \frac{L_{\max}}{\phi_i} + \frac{L_j}{\phi_j}, C_i + \frac{L_{\max}}{\phi_j} + \frac{L_i}{\phi_i} \right) \right]$$

where  $C_i$  is the maximum normalized service that a flow may receive in a WFQ server in excess of that in the GPS server, given by:

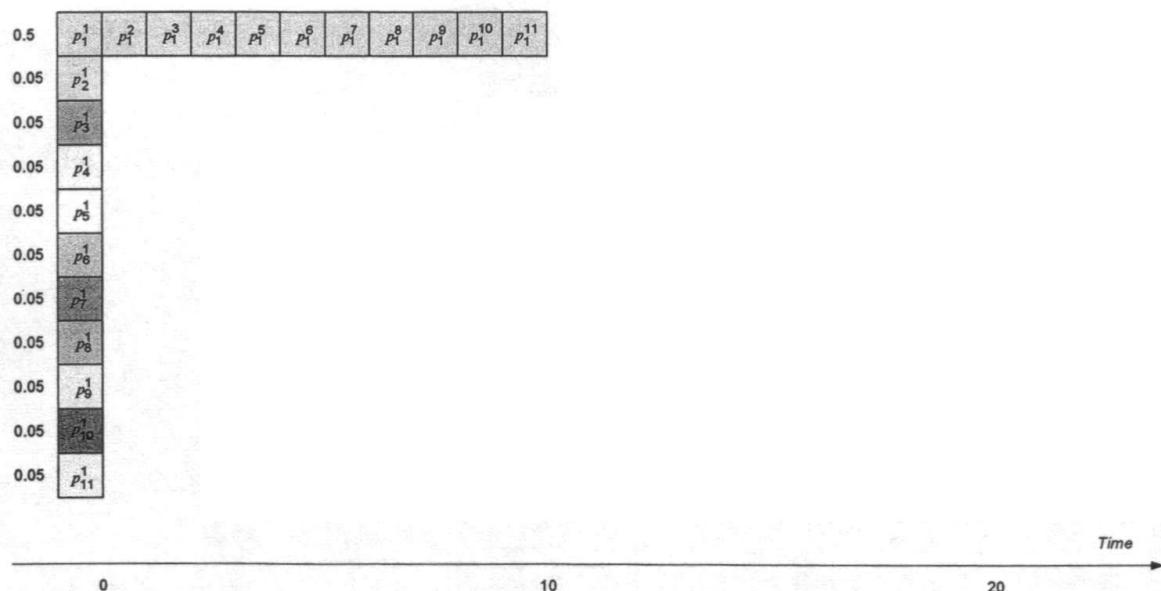
$$C_i = \min \left[ (N-1) \frac{L_{\max}}{\phi_i}, \max_{i \leq n \leq N} \left( \frac{L_n}{\phi_n} \right) \right]$$

It can be proved that the above bound is tight.

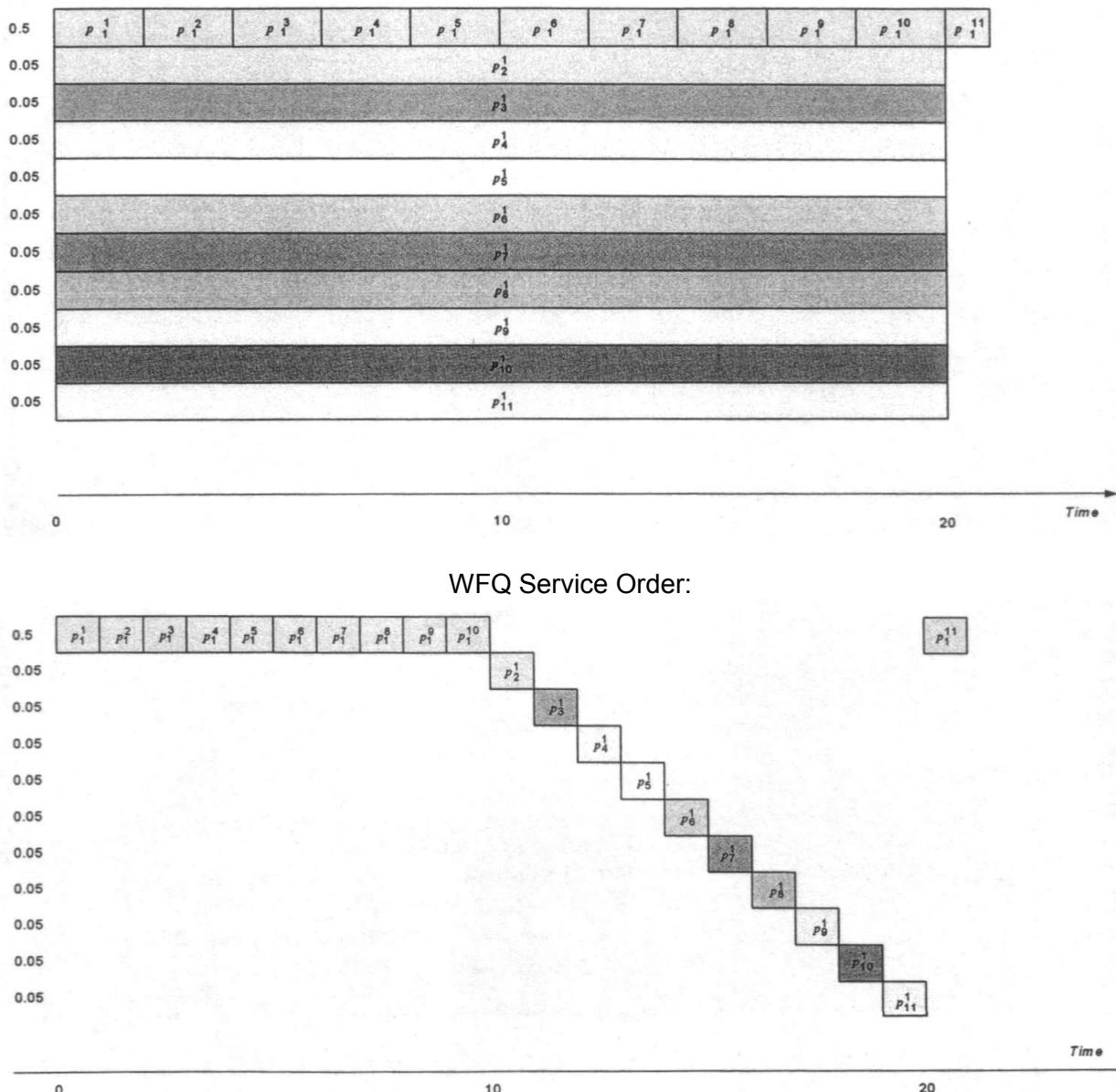
## 2.21 Problem With WFQ

Packets can leave much earlier in a WFQ system than in a GPS system, which means that WFQ can be far ahead of GPS in terms of the number of bits served for a flow.

$$d_i^{k,WFQ} - d_i^{k,GPS} \leq \frac{L_{\max}}{r} \quad \forall i, k$$



GPS Service Order:

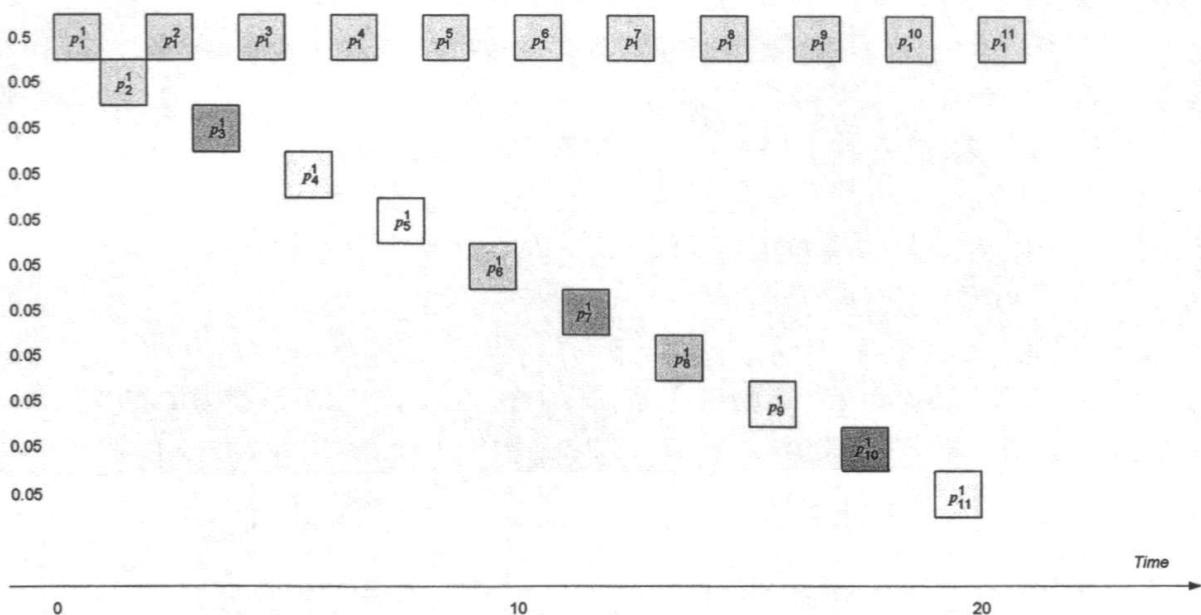


This cycle of bursting 10 packets and going silent for 10 packets times can continue indefinitely. With more flows, the length of the period between bursting and silence can be larger. Such oscillation is undesirable for flow and congestion control in data communication networks. Within the framework of feedback-based congestion control, a data source has to balance between two considerations: on the one hand, it wants to send data to the network as fast as possible, on the other hand, it does not want to send data so fast that causes network congestion.

## 2.22 Approximating GPS with WF<sup>2</sup>Q

To minimize the difference between a packet system and the fluid GPS system, another class of scheduling algorithms called *shaper-schedulers* has been proposed. A packet is *eligible* if its virtual start time is no greater than the current system virtual time. With these algorithms, when the server is picking the next packet to transmit, it chooses, among all the eligible packets, the one with the smallest timestamp. This is called the eligibility test or *Smallest Eligible virtual Finish time First* (SEFF) policy. *Worst-case Fair Weighted Fair Queuing* or WF<sup>2</sup>Q, is one such example. Recall that in a WFQ system, when the server chooses the next packet for transmission at time  $\tau$ , it selects, among all the packets that are backlogged at  $\tau$ , the first packet that would complete service in the corresponding GPS system. In a WF<sup>2</sup>Q system, when the next packet is chosen for service at time  $\tau$ , rather than selecting it from among all the packets at the server as in WFQ, the server only considers the set of packets that have started (and possibly finished) receiving service in the corresponding GPS system at time  $\tau$ , and selects the packet among them that would complete service first in the corresponding GPS system.

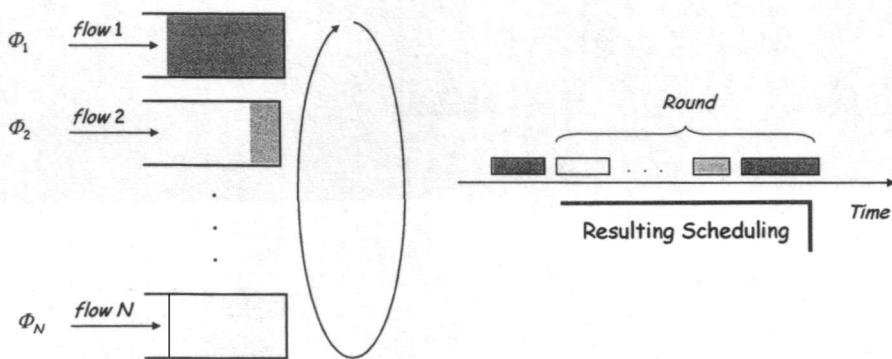
WF<sup>2</sup>Q Service Order:



Even in the case when flow 1 is sending back-to-back packets, its output from the  $\text{WF}^2\text{Q}$  system is rather smooth, as opposed to the bursty output under a WFQ system.

## 2.23 Deficit Round Robin

Deficit Round Robin (DRR) is a variation of Weighted Round Robin (WRR) designed for providing weighted fair queuing to flows with variable length packets at a low complexity. Rather than sorting packets according to some function of time, as in the WFQ scheduling algorithm, a DRR scheduler services flows in a fixed order, and each flow is allowed to transmit a bounded amount of traffic every time it is serviced. The time interval between two consecutive visits of the same flow is called a round:



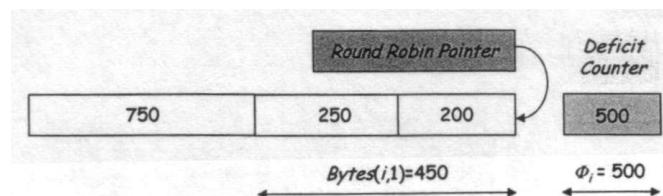
Each flow  $i$  is characterized by:

- a quantum of  $\Phi_i$  traffic units (e.g. bit, bytes, ATM cells, etc.), which measures the amount of traffic units that flow  $i$  should ideally transmit during a round;
- a deficit variable  $DC_i$ .

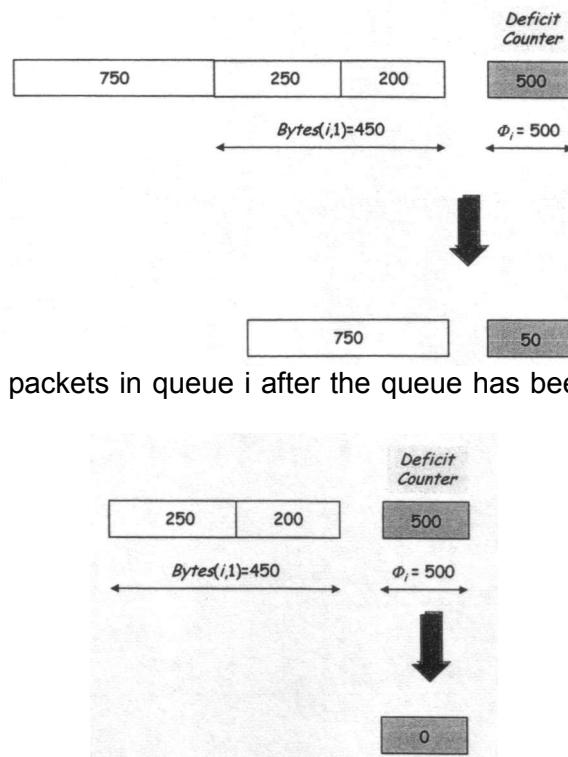
$\Phi_i$  plays the same role as a weight in the WFQ algorithm. Packets coming in on different flows are stored in different queues. Denote by  $\text{Bytes}(i,k)$  the number of bytes sent out for flow  $i$  (or queue  $i$ ) in round  $k$ .

DRR Steps:

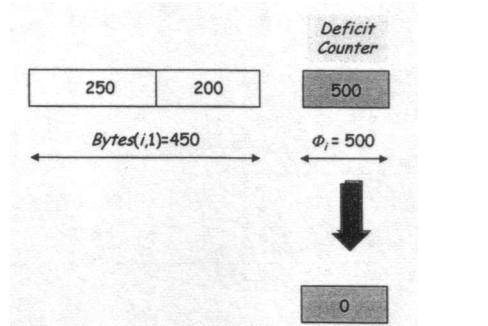
1.  $DC_i$  is initialized to 0.
2. Each queue  $i$  is allowed to send out packets in the first round subject to the restriction that  $\text{Bytes}(i,1) \leq \Phi_i$ .



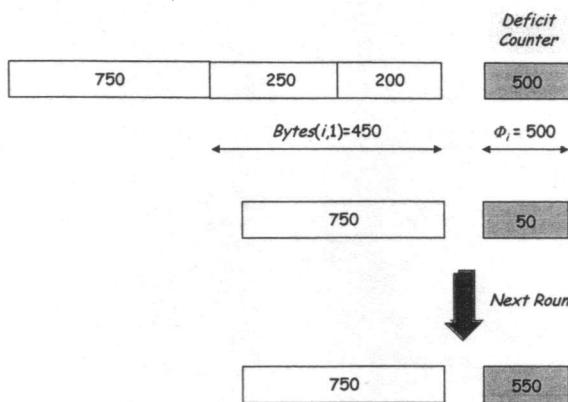
3. If there are more packets in queue  $i$  after the queue has been serviced, the remaining amount  $[\Phi_i - \text{Bytes}(i,k)]$  is stored in the state variable  $DC_i$ .



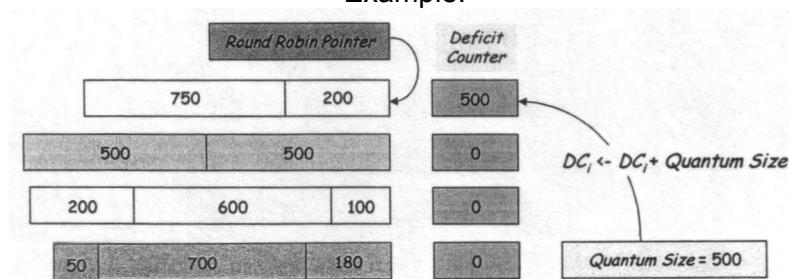
4. If there are no more packets in queue  $i$  after the queue has been serviced,  $DC_i$  is reset to zero.

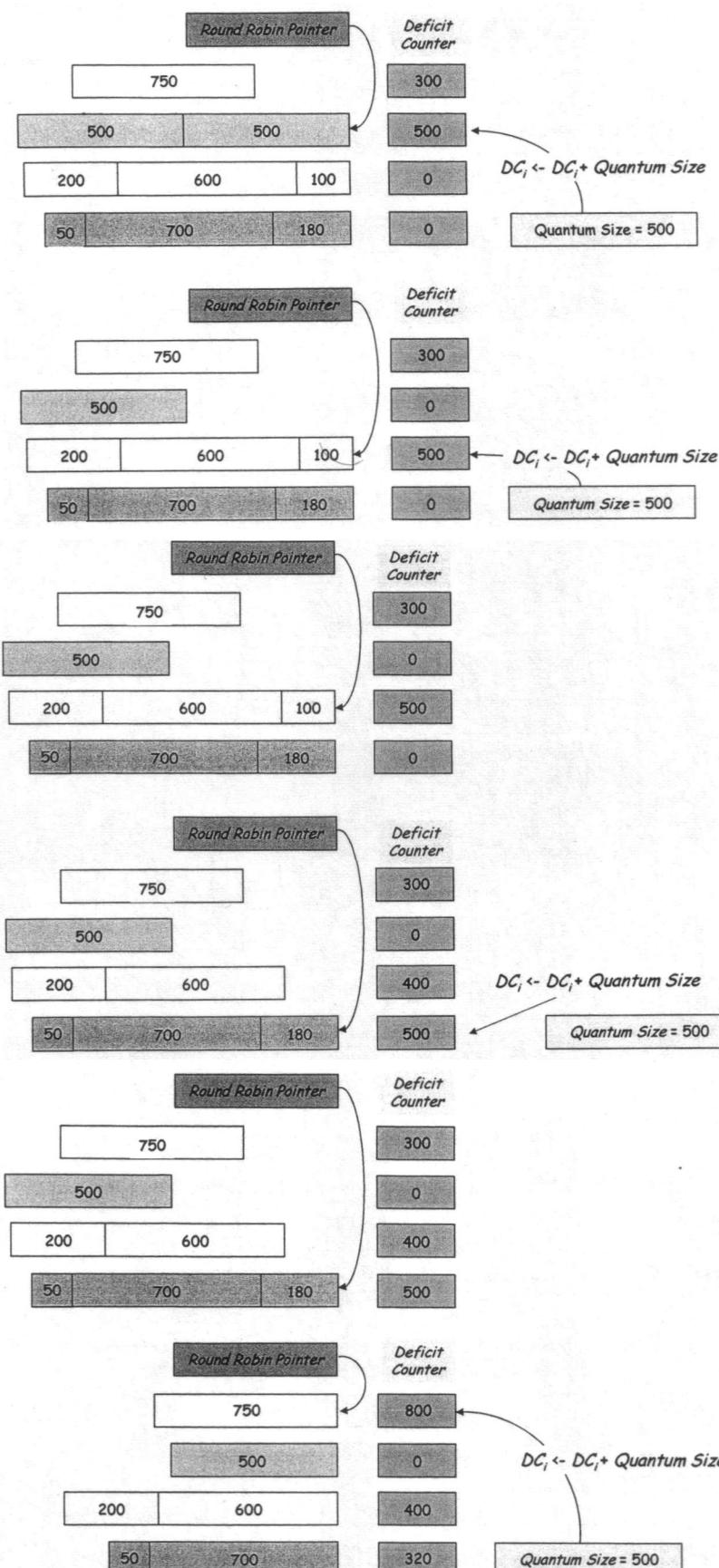


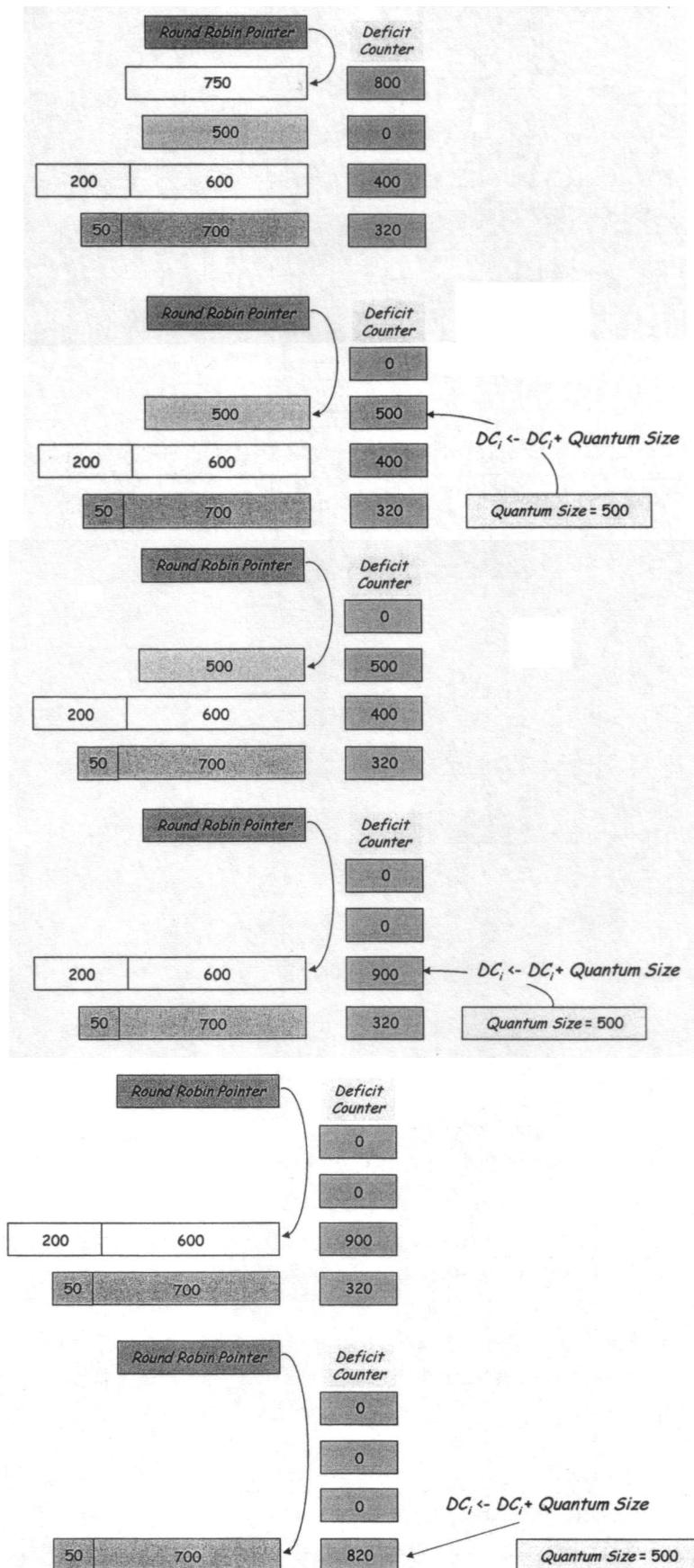
5. In subsequent rounds, the amount of bytes this flow is entitled to transmit is the sum of  $DC_i$  of the previous round added to  $\Phi_i$ , i.e.  $\Phi_i + DC_i$ .

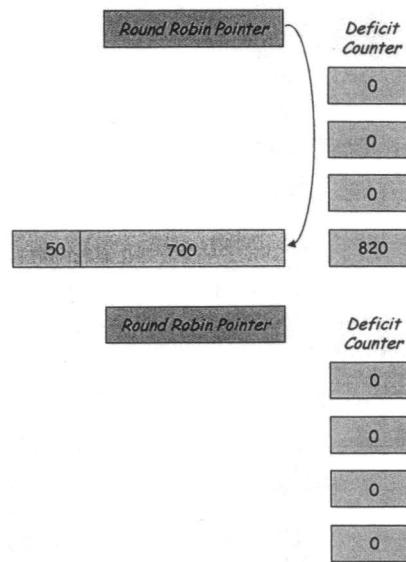


Example:









To avoid examining empty queues, DRR keeps an auxiliary list *ActiveList* that is a list of indices of queues that contain at least one packet. Whenever a packet arrives to a previously empty queue  $i$ ,  $i$  is added to the end of *ActiveList*. Whenever index  $i$  is at the head of *ActiveList*, the algorithm services up to  $\Phi_i + DC_i$  worth of bytes from queue  $i$ . If at the end of this service opportunity, queue  $i$  still has packets to send, the index  $i$  is moved to the end of *ActiveList*; otherwise,  $DC_i$  is set to zero and index  $i$  is removed from *ActiveList*.

## 2.24 Deficit Round Robin Results

Let  $L_{i,\max}$  be the maximum length of a packet for flow  $i$  (measured in bits). The following inequality has been proved to hold right after a flow has been serviced during a round:

$$0 \leq DC_i < L_{i,\max}$$

which means that the deficit counter does not grow indefinitely. This ensures that the amount of service that a flow misses due to packet atomicity is always bounded.

- Fairness:

$$RFB_{i,j} = F \left( 1 + \frac{L_{i,\max}}{\phi_i} + \frac{L_{j,\max}}{\phi_j} \right)$$

$$RFB = 3F$$

$$\text{where: } F = \sum_{i=1}^N \phi_i$$

- Rate:

$$g_i = \frac{\phi_i}{F} r$$

- Complexity:

DRR has  $O(1)$  complexity if the following condition holds:

$$L_{i,\max} \leq \phi_i \quad \forall i$$

If some flows violate this condition, it is possible that a flow which is selected for transmission (i.e. dequeued from the head of the list), though backlogged, cannot transmit a packet during the current round. Nevertheless, its deficit variable must be updated and the flow must be queued back at the tail of the list. In a worst case, this can happen as many consecutive times as the number of flows violating the inequality. Therefore, if the above condition does not hold, the number of operations needed to transmit a packet in the worst case can be as large as  $O(N)$ .

# **3 Integrated Services Model (*IntServ*)**

## **3.1 Introduction**

The *Integrated Services* (*IntServ* for short) architecture is a new architecture for resource allocation to meet the requirements of real-time applications. It was developed by the IETF around 1995-97. The basic approach is per-flow resource reservation. The goal of Integrated Services is to preserve the datagram model of IP-based networks and at the same time support resource reservation for real-time applications. The IETF (*IntServ*) working group developed specifications of a number of service classes (models, paradigms) designed to meet the needs of some of the application types described above.

## **3.2 Service Classes**

Given the broad categories of applications and associated requirements discussed previously, the question arises of how many service classes should be provided by *IntServ* to support the above applications. There are currently two service classes defined for *IntServ* besides Best-Effort: Guaranteed QoS Service (RFC 2212) and Controlled Load Service (RFC 2211). Both these service classes need resource reservations. Clearly, the above two service classes are subset of all the service classes that might be provided.

The *Guaranteed QoS Service Class* assures that every packet of flow will arrive within a guaranteed delivery time, i.e. every packet of a flow will arrive at the receiver not later than a certain time (maximum delay time) after it was transmitted by its source. The *Guaranteed QoS Service class* is designed for real-time intolerant applications and provides firm bounds (mathematically provable) on end-to-end delays. It does not control the minimal or average delay of packets and does not attempt to minimize jitter.

The *Controlled Load Service Class* is designed for tolerant and adaptive applications and was motivated by the observation that existing applications of this type run quite well on networks that are not heavily loaded. The audio application, for example, can adjust its playback point as network delay varies, and produces reasonable audio quality as long as loss rates remain on the order of 10% or less. The aim of the *Controlled Load Service* is to emulate a lightly loaded network for those applications that request this service class, even though the network as a whole may in fact be heavily loaded. The *Controlled Load Service class* offers only one service level, i.e. it approximates best-effort service over lightly loaded networks. This means that applications that make QoS reservations using *Controlled Load Services* are provided with service closely equivalent to the service provided to uncontrolled (best-effort) traffic under lightly loaded conditions. In this context, lightly loaded conditions means that a very high percentage of transmitted packets will be successfully delivered to the destination, and the transit delay for a very high percentage of the delivered packets will not greatly exceed the minimum transit delay. Each router in a network that accepts requests for *Controlled Load Service* must ensure that adequate bandwidth and packet processing resources are available to handle QoS reservation requests. This can be realized by means of admission control. Before a router accepts a new QoS reservation, it must consider all important resources, such as link bandwidth, router or switch port buffer space and computational capacity of the packet forwarding. The *Controlled Load Service* does not make use of specific target values for QoS parameters, such as bandwidth, delay, or loss. Applications that use *Controlled Load Service* must guard against small amounts of packet loss and packet delays. Thus, *Controlled Load Service* is designed for applications that can tolerate reasonable amount of packet loss and delay, such as audio and video conferencing applications.

## **3.3 Overview of Mechanism**

Now that we have augmented the best-effort service with two new service classes, the next question is how we design a network that provides these service classes to applications. To provide the new service classes to applications *IntServ* introduces the following concepts:

- FlowSpec,
- Admission Control,
- Resource Reservation,
- Packet Classifier,
- Packet Scheduler.

First, whereas with best-effort service we can just tell the network where we want our packets to go and leave it at that, a real-time service involves telling the network something more about the service class we require. We may ask the underlying network:

- qualitative informations such as “I want to use a Controlled Load Service” or
- quantitative informations such as “I need a maximum delay of 100 ms”.

In addition to describing the class of service we want, we need to tell the network something about what we are going to inject into it, i.e. the traffic profile, since a low-bandwidth application is going to require fewer network resources than a high-bandwidth application. The set of informations (Class of Service & traffic profile) that we provide to the network is referred as a FlowSpec.

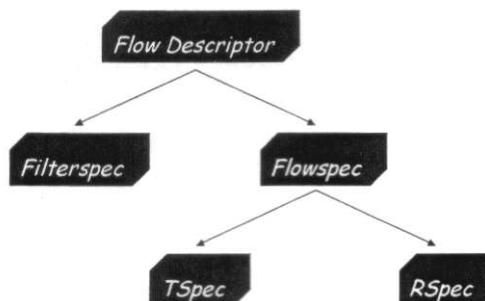
Second, when we ask the network to provide us with a flow with a specific QoS, the network needs to check if it has the necessary resources to provide it. For example, if 10 applications ask for a flow of 2Mbps rate, and they all share a link with 10Mbps capacity, the network will have to say “no” to some of them. The process of deciding when to say “no” is called Admission Control.

Third, we need a mechanism by which the users of the network and the components of the network itself exchange information such as requests for service, FlowSpecs, and Admission Control decisions. This is called “signalling” in the ATM world, but since this word has several meanings, we refer to this process as “Resource Reservation”.

Fourth, when flows and their requirements have been described, and admission control decisions have been made, the network routers need to meet the requirements of the flows. A key part of meeting these requirements is managing the way packets are queued and scheduled for transmission in the router. This last mechanism is Packet Scheduling.

### 3.4 Flow Descriptor

In the IntServ model each flow is assigned a Flow Descriptor. The Flow Descriptor defines the traffic and QoS characteristics for a specific flow. In the IntServ specifications, the Flow Descriptor consists of a Filter Specification (FilterSpec) and a Flow Specification (FlowSpec).



### 3.5 FlowSpec

To make a reservation, an application must characterize the traffic that it will inject into the network and specify the service requirements for the flow. In IntServ these are described in a so called Flow Specification or FlowSpec for short. The Flow Specification is in essence a service contract that specifies the traffic that the source will send and the resources and services the network promises to commit. The FlowSpec contains a set of parameters that are called the invocation information. It is possible to classify the invocation information into groups: Traffic Specification (TSpec) and Service Request Specification (RSpec). The TSpec describes the traffic characteristics (traffic profile) of a flow by means of a number of parameters. The following parameters are common:

- peak rate,
- average rate,
- burst size.

**Peak rate:** the highest rate at which a source can generate traffic. The peak rate is limited by the speed of the hardware devices. For example, we cannot generate packets faster than 10Mbps over a 10Mbps Ethernet. In some cases traffic is deliberately shaped to reduce the peak rate from the source; the peak rate can be calculated from the packet size and the spacing between consecutive packets.

**Average rate:** the average transmission rate over a time interval. The average rate can be calculated in many ways, and the results can be quite different. It is important to know the exact method and the time interval used in the calculation. Typically the average rate is calculated with a moving time window.

**Burst size:** the maximum amount of data that can be injected into the network at the peak rate. The burst size reflects the “burstiness” of the traffic source.

RSpec specifies the QoS the application wants to request for a specific flows. The following common parameters have been widely used to describe QoS requirements:

- minimum rate,
- delay,
- delay jitter,
- packet loss rate.

In the IntServ implementation, the information from TSpec and RSpec is used in the packet scheduler.

**Minimum rate:** the minimum amount of bandwidth required by an application flow. The time interval for measuring rate should also be specified since different measurement intervals, may yield different results. Bandwidth allocation is guaranteed by the packet-scheduling algorithms. The WFQ class of scheduling algorithms is capable of providing minimum bandwidth guarantees over very small time intervals.

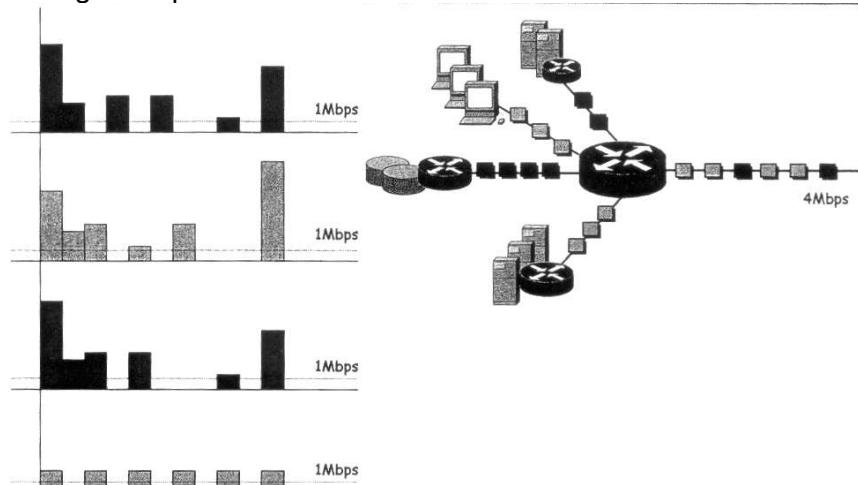
**Delay:** the delay requirement can be specified as the average delay or worst-case (maximum) delay.

**Delay jitter:** a delay-jitter requirement specifies the maximum difference between the largest and smallest delays that packets experience. In any case, the delay jitter should not be larger than the maximum delay.

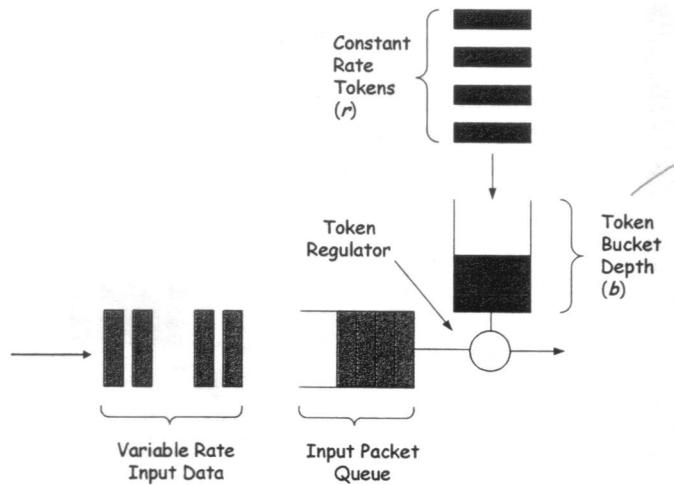
**Loss rate:** loss rate is the ratio of lost packets and total packets transmitted. Packet losses in the Internet are often caused by congestion, and such losses can be prevented by allocating sufficient bandwidth and buffers for traffic flows.

### 3.6 Token Bucket

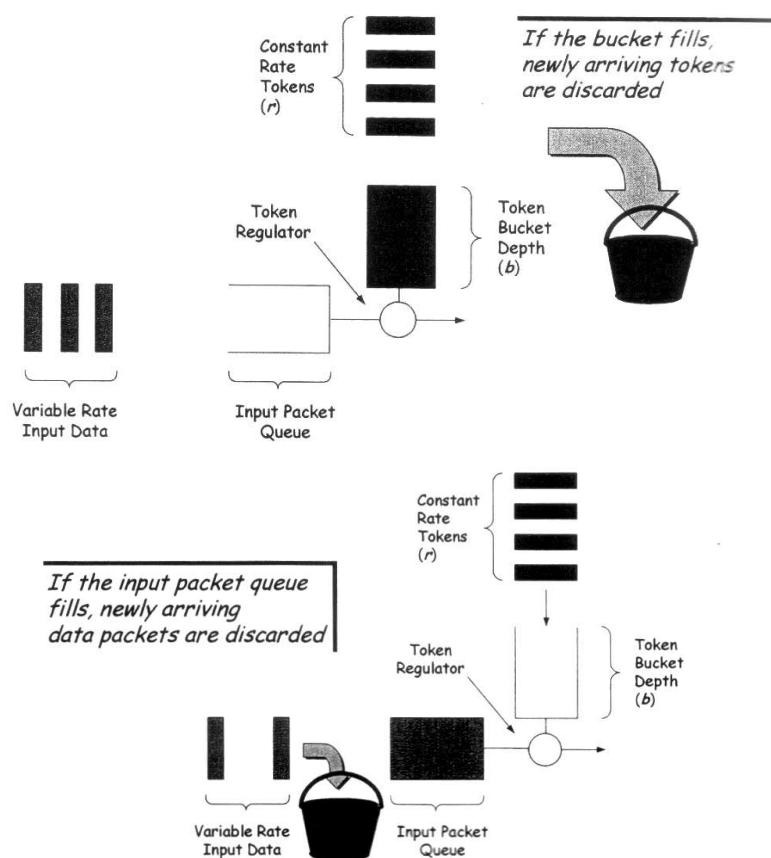
For most applications, the bandwidth requirement is not constant; it is something that varies constantly. A video application, for example, will generally generate more bits per second when the scene is changing rapidly than when it is still. Just knowing the long-term average bandwidth is not enough, as the following example illustrates.



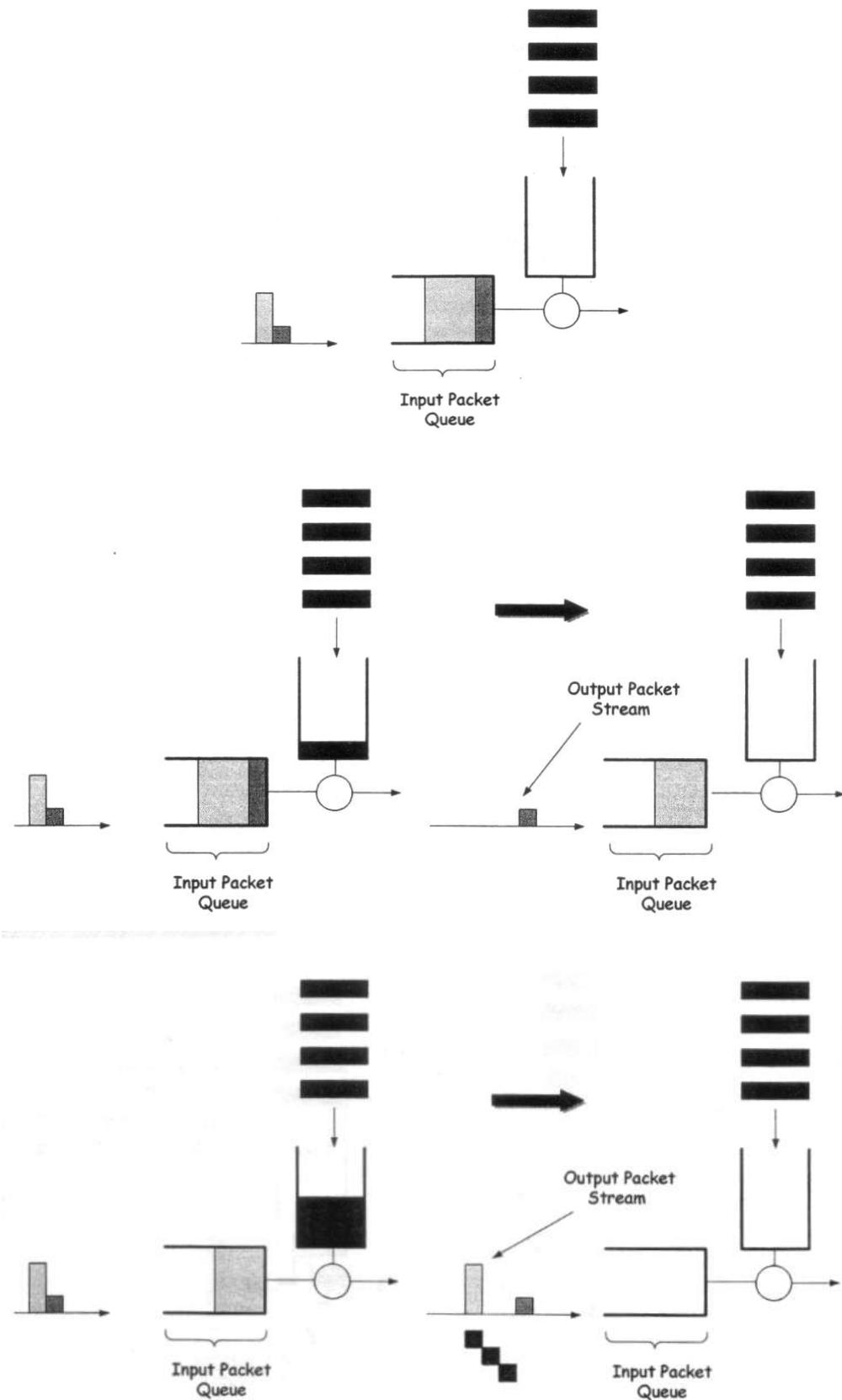
Suppose that we have 4 flows that arrive at a router on separate input ports and that all leave on the same 4Mbps link. Assume that over some suitably long interval each flow can be expected to send no more than 1Mbps. You might think this presents no problem. However, if these are available bit rate applications, such as compressed video, then they will occasionally send more than their average rates. If enough sources send at above their average rates, then the total rate at which data arrives at the switch will be greater than 4Mbps. This excess data will be queued before it can be sent on the link and the longer this condition persists, the longer the queue will get. Packets might have to be dropped, and even if it doesn't come to that, data sitting in the queue is being delayed. If packets are delayed long enough, the service that was requested will not be provided. Thus we need to know something about how the bandwidth of our sources varies with time. The purpose of the token bucket is to describe the flow of traffic as it enters and travels through the network. Given such a traffic specification, it is then simple to talk about the amount of network resources a flow will consume and how it can be serviced. The token bucket is a variation of the leaky bucket.



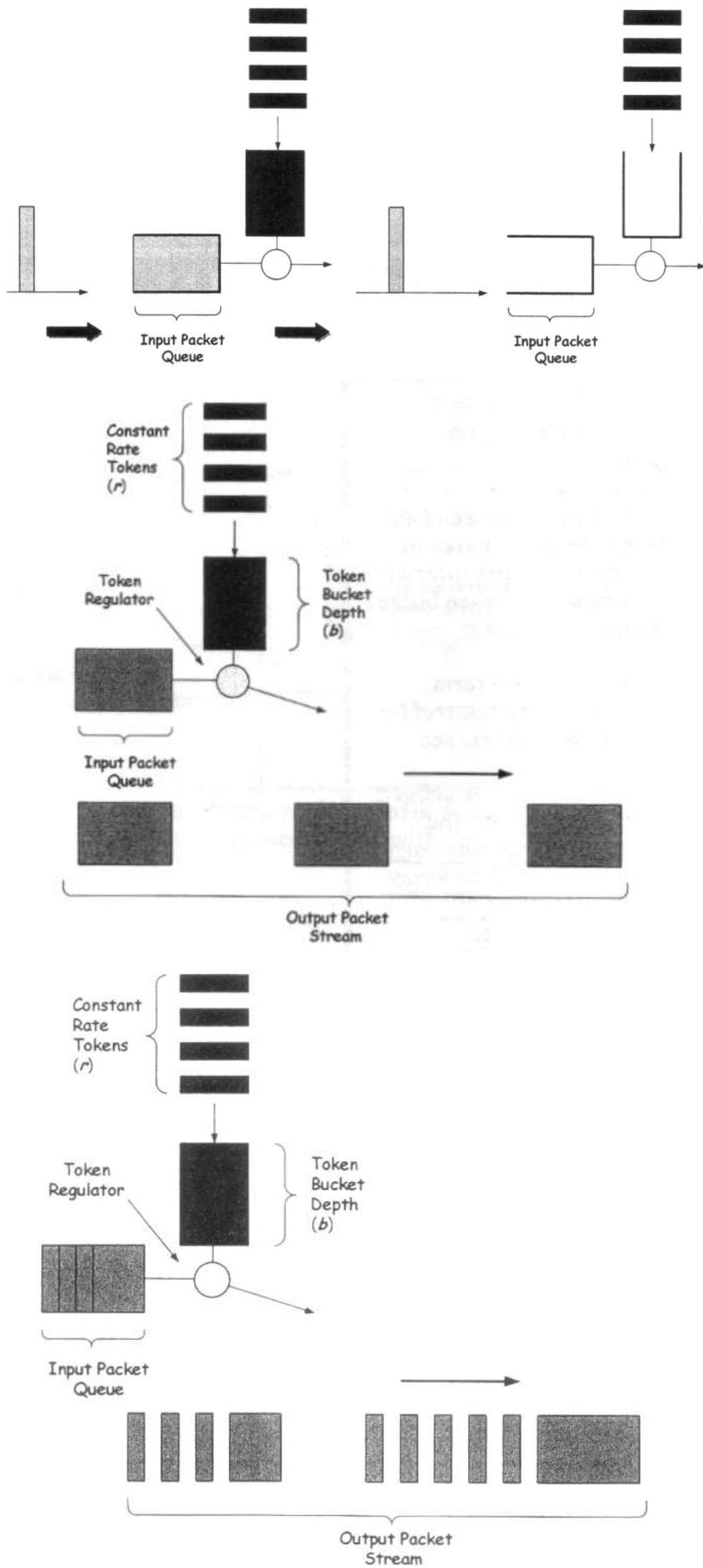
The token bucket is described by two parameters: token rate  $r$  and bucket depth  $b$ . The bucket has a capacity of  $b$  bytes and tokens are placed in the bucket at  $r$  bytes/sec. When a packet arrives it is placed in the buffer.



Assume that each token corresponds to one byte of data. Suppose a packet of size  $d$  bytes is at the front of the buffer, to transmit the packet, the regulator must remove  $d$  tokens from the bucket. If the bucket does not contain a sufficient number of tokens ( $d$ ), the packet must wait for  $d$  tokens to accumulate in the bucket before it can be transmitted.

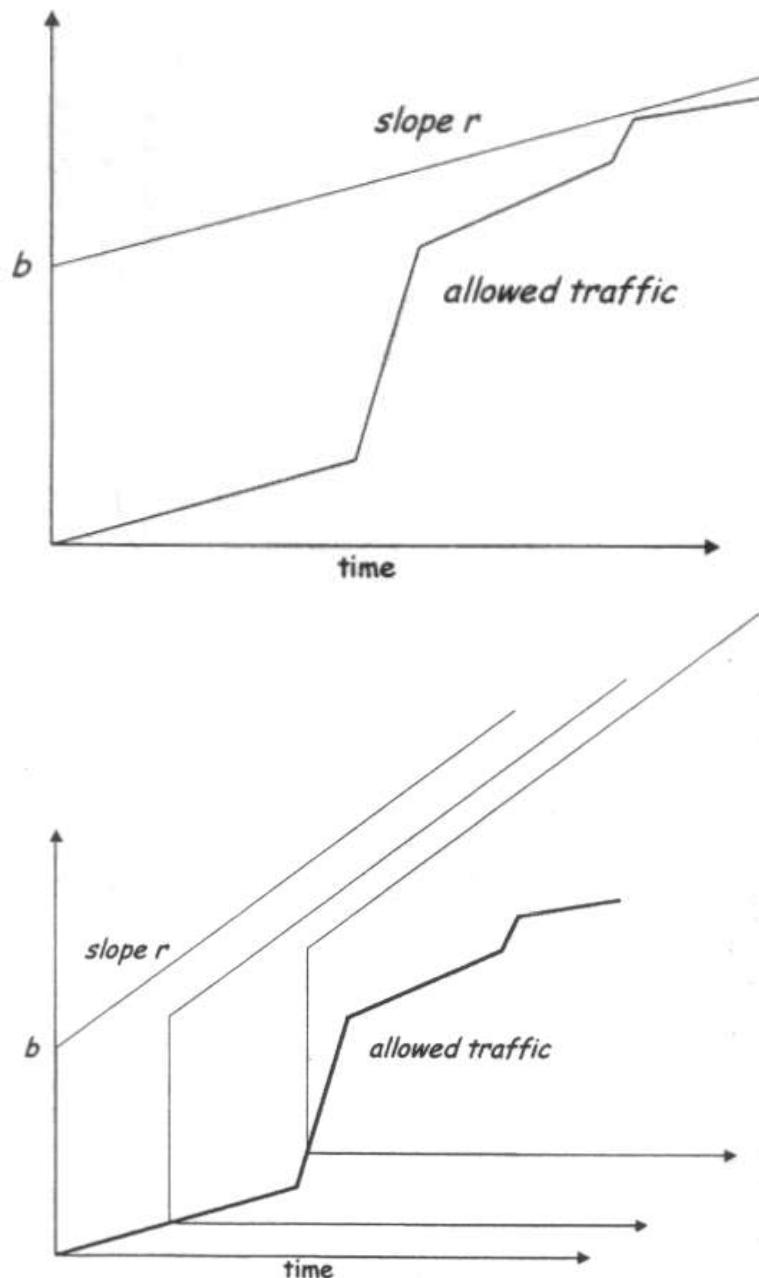


If the bucket is full and a burst of packets arrives into the buffer on the left, a burst of packets (equal to  $b$  bytes in size when added together) will leave the regulator immediately with no gaps in between.

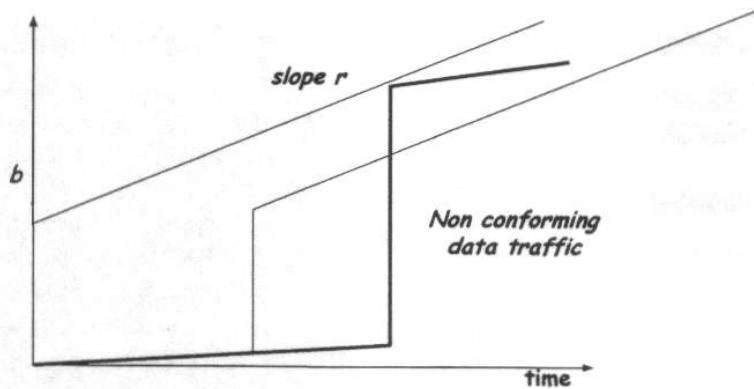


The token bucket has a number of interesting properties:

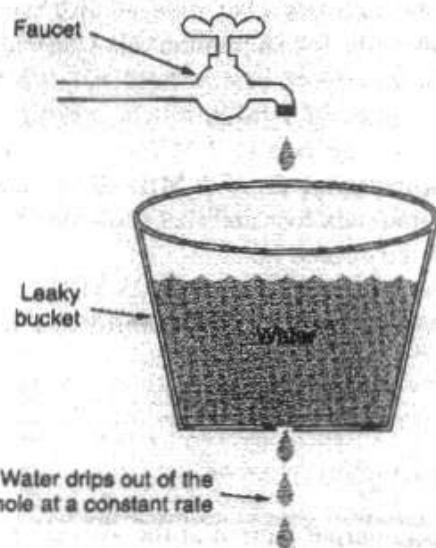
1. First, it will never send out more than  $b+r \cdot t$  bytes of data over any interval  $t$  for a flow with token bucket parameters  $b$  and  $r$ .
2. Second, the long-term average rate of the traffic flow will be  $r$  bytes/sec.
3. Third, the source is allowed to send bursts into the network, but the maximum burst size cannot be larger than the depth of the bucket, i.e.  $b$  bytes.



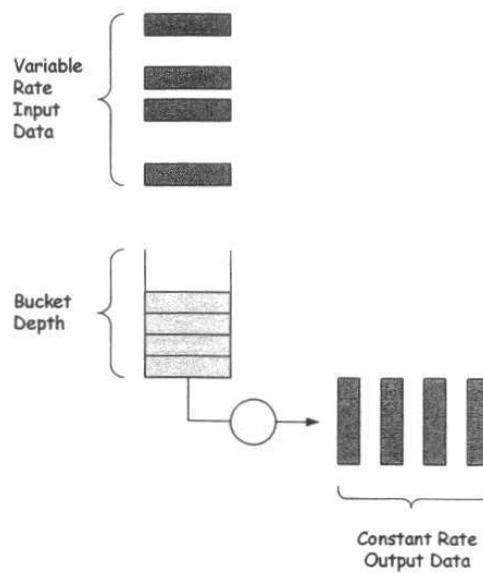
Example of data traffic profile which does not conform to a token bucket with depth  $b$ :



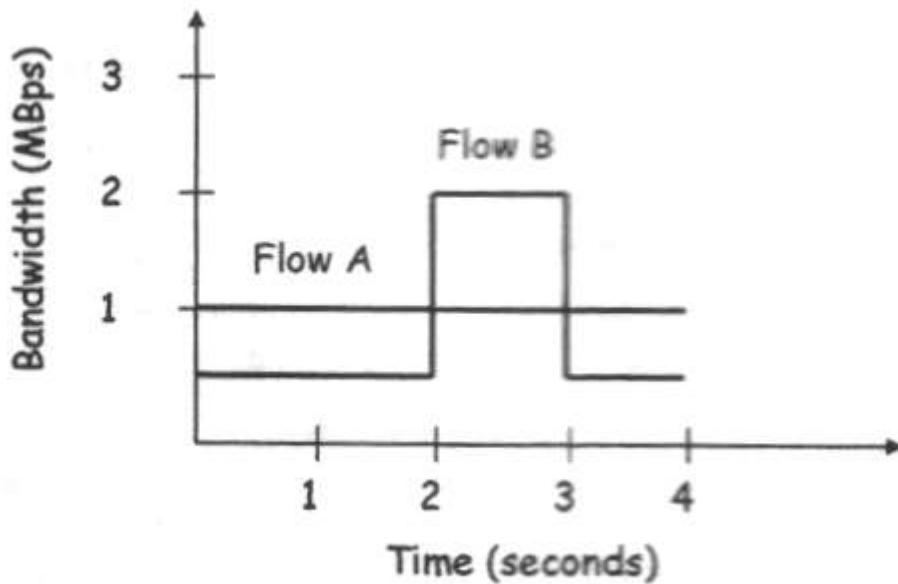
The leaky bucket algorithm:



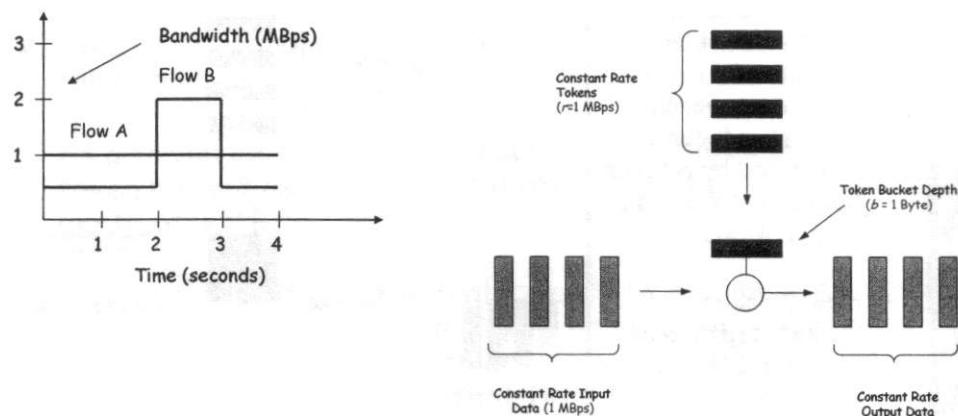
Imagine a bucket with a small hole in the bottom as illustrated in the previous figure. No matter the rate at which water enters the bucket, the outflow is at a constant rate,  $r$ , when there is any water in the bucket and zero when the bucket is empty. Also, once the bucket is full, any additional water entering it spills over the sides and is lost (i.e., does not appear in the output stream under the hole). An important difference between leaky bucket and token bucket is that the former uses the bucket for managing the regulator rather than controlling the flow's arrival of data.



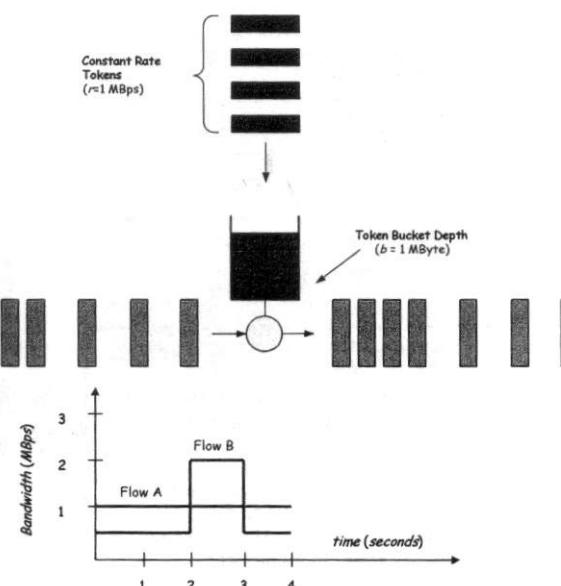
The following example illustrates how a token bucket can be used to characterize a flow's bandwidth requirements. For simplicity, assume that each flow can send data as individual bytes, rather than as packets.



Flow A generates data at a steady rate of 1Mbps, so it can be described by a token bucket with a rate  $r$  of 1Mbps and a bucket depth  $b$  of 1 byte. This means that it receives tokens at a rate of 1Mbps but that it cannot store more than 1 token – it spends them immediately.

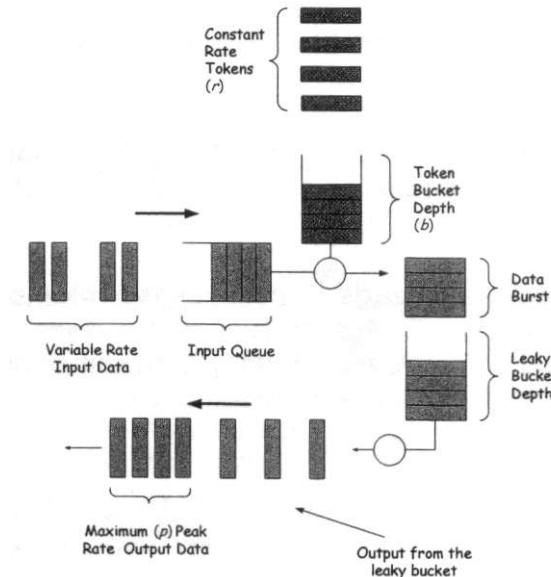


Flow B also sends at a rate that averages out to 1Mbps over the long term, but does so by sending at 0.5Mbps for 2 seconds and then at 2Mbps for 1 second. Since the token bucket rate  $r$  is a long-term average rate, flow B can be described by a token bucket with a rate  $r$  of 1Mbps. Flow B needs a bucket depth  $b$  of at least 1MB, so that it can store up tokens while it sends at less than 1Mbps to be used when it sends at 2Mbps.

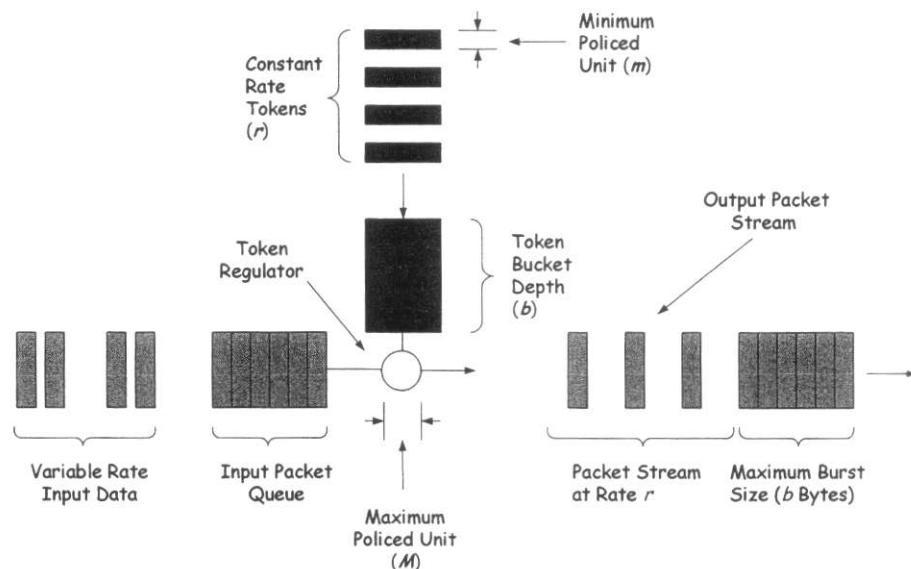


For the first 2 seconds flow  $B$  receives tokens at a rate of 1Mbps but spends them at only 0.5Mbps, so it can save up  $2 \times 0.5 = 1\text{MB}$  of tokens, which it then spends in the third second (along with the new tokens that continue to accrue in that seconds) to send data at 2Mbps. At the end of the third second, having spent the excess tokens, it starts to save them up again by sending at 0.5Mbps again. Note that a single flow can be described by many different token buckets. As a trivial example, flow  $A$  could be described by the same token bucket as flow  $B$ , with rate of 1Mbps and a bucket depth of 1MB. The fact that it never actually needs to accumulate tokens does not make that an inaccurate description, but it does mean that we have failed to convey some useful information to the network – the fact that flow  $A$  is actually very consistent in its bandwidth needs. In general, it is good to be as explicit about the bandwidth needs of an application as possible, to avoid over allocation of resources in the network.

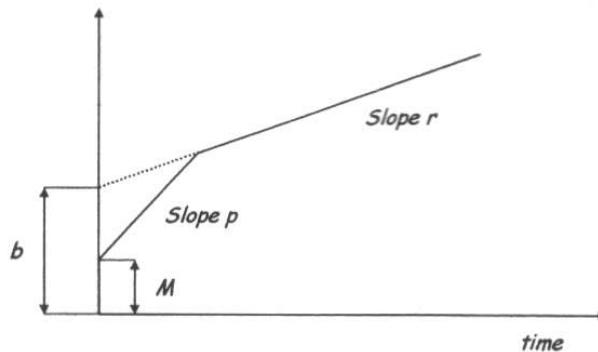
The only problem with the token bucket is how the burst of packets is introduced into the network. Suppose that a flow is idle for some time and the bucket fills completely. When the flow is active again, it can send  $b$  bytes of data at the link transmission rate without stopping. This can potentially deprive other flows from access to network resources for the duration of the burst. Moreover, such a behavior may require a large amount of buffering at network devices along the path. To avoid such a problem, many researchers have proposed using a token bucket model that combines the basic token bucket scheme with a leaky bucket. The output form the regulator in the token bucket is not transmitted, but, instead, is places into a leaky bucket of size  $b$  and the leaky bucket regulator transmits data out of that bucket at a peak rate of  $p$  bytes/sec. The peak rate  $p$  must be much larger than  $r$ , so that a burst of data can still be transmitted, while not completely starving other traffic.



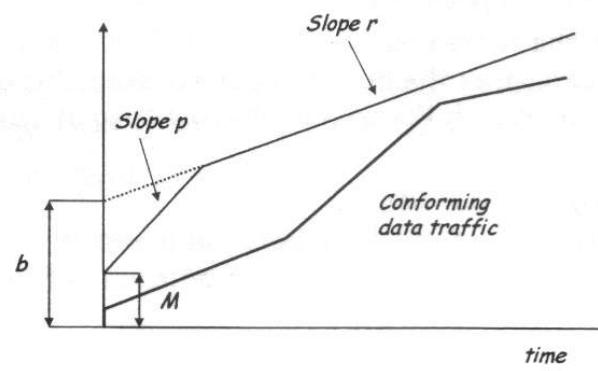
The combined scheme permits bursty traffic to be sent, limited to a burst of  $b$  bytes, with a long-term average rate of  $r$  and a peak rate of no more than  $p$ . A traffic flow transmitted using such a scheme is thus characterized by three parameters, namely,  $\langle b, r, p \rangle$ . Under the IntServ model, the sender TSpec describes a traffic flow using the parameters  $\langle b, r, p, m, M \rangle$ . The first three parameters ( $b, r, p$ ) are the token bucket parameters as previously described. In addition,  $M$  specifies the maximum datagram size in the flow and  $m$  specifies the minimum policed unit. The parameter  $m$  is used by network nodes to estimate the correct amount of bandwidth to be allocated to the flow.



Note that the application can specify only the minimum size of packet it generates in terms of the data or payload within the packet. Additional headers get tacked on the data to create the packet that gets transmitted over a particular link layer technology. So the network nodes use the value of  $m$  to compute the maximum bandwidth overhead needed to carry a flow's packet over a specific link layer technology. This computation is based on the ratio of  $m$  to the link layer header size [RFC2210]. The maximum size  $M$  is needed by nodes to determine whether or not they can service such a packet, and the sender can choose  $M$  based on the Maximum Transmission Unit (MTU) supported over the path.



For any period  $T$ , the amount of data sent cannot exceed  $M + \min(pT, b + rT - M)$ .



It can be proved how, with a token bucket shaping algorithm, a guarantee on the minimum output rate is equivalent to a guarantee on the maximum queuing delay. This property of token bucket is used by IntServ to give guarantees on end-to-end delay to flows.

### 3.7 TSpec

The parameters used by end devices (i.e. hosts) to request QoS guarantees from the network is the *TOKEN\_BUCKET\_TSPEC* parameter. The *TOKEN\_BUCKET\_TSPEC* parameter is actually

comprised of multiple values that characterize the data flow in detail. These are:

- the token rate ( $r$ ),
- the bucket depth ( $b$ ),
- the peak rate ( $p$ ),
- the minimum policed unit ( $m$ )
- the maximum packet size ( $M$ )

**Token rate ( $r$ )** (bytes/second): the rate at which tokens arrive at the token bucket. It describes the average data rate of the flow.

**Bucket depth ( $b$ )** (bytes): the size of the token bucket. It roughly estimates the buffer size that should be allocated to the flow's data to absorb the unavoidable bursting of data.

**Peak rate ( $p$ )** (bytes/second): the maximum rate at which packets can transmit. it specifies the maximum short-term data rate of the flow.

**Minimum policed unit ( $m$ )** (bytes): describes the minimum datagram size, taking into account all but the link layer headers. any packet with a size smaller than  $m$  will be counted  $m$  bytes.

**Maximum packet size ( $M$ )** (bytes): represents the largest packet size that can be accepted.

### 3.8 RSpec

RSpec describes the service requirements with two parameters:

- **Service rate ( $R$ )** (bytes/second): the required service rate, which must be no less than the token bucket parameter  $r$  in TSpec. Recall that, with a token bucket algorithm, to a minimum rate is automatically associated a maximum queuing delay.
- **Slack term ( $S$ )** (microseconds). the extra amount of delay that a node may add that still meets the end-to-end delay requirements.

QoS reservations using Controlled Load Service need to provide a TSpec. An RSpec is not necessary, because Controlled Load Service does not guarantee any fixed bandwidth or minimum packet delays. Controlled Load Service provides QoS control only for packets that respect the token bucket rule. In the Guaranteed QoS Service model, TSpec and RSpec are both necessary to set up a flow reservation.

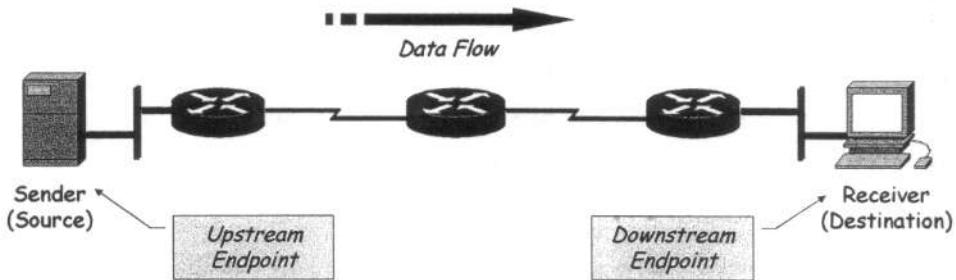
### 3.9 RSVP Overview

While connection oriented networks have always needed some sort of setup protocol to establish the necessary virtual circuit state in the switches, connectionless networks like the Internet have had no such protocols. We have already seen that we need to provide a lot more information to the network when we want a real-time service from it. While there have been a number of setup protocols (signalling protocols) proposed for the Internet the one which most current attention is focused is RSVP (Resources reSerVation Protocol). RSVP provides a general means for an application to communicate its QoS requirements to an Integrated Services Internet infrastructure. RSVP is not a data transport protocol, but a control protocol that signals QoS requests on behalf of a data flow. RSVP allows reservations for both the one to one unicast and the many to many multicast data flows.

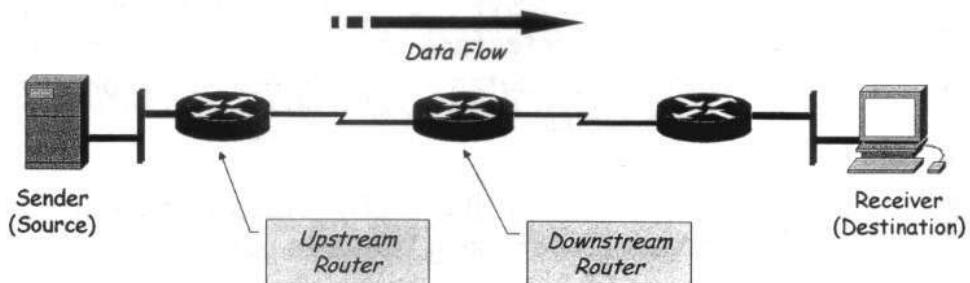
RSVP is used:

- by the sender to describe the attributes of its data flow, and then
- by the receiver to request a specific level of QoS of the data flow.

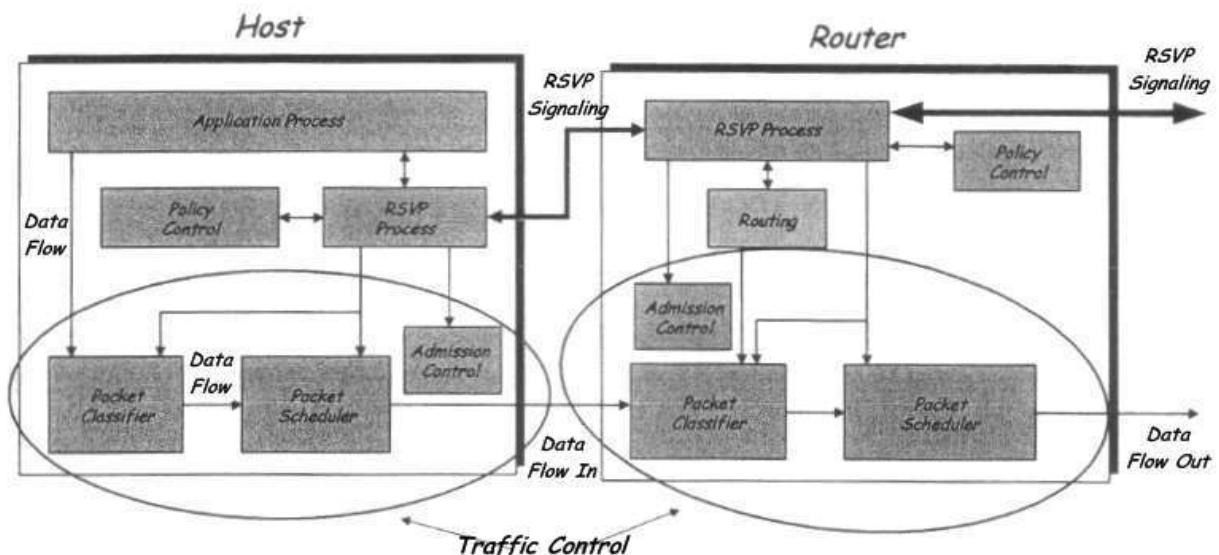
Thus, for RSVP, characterization of the flow is the sender's responsibility, while the receiver specifies its particular service requirements. When resources are reserved for a flow, they are effectively committed to servicing the specified data flow above all other competing network traffic. RSVP establishes a reservation for a simplex flow traveling from its source to its destination. Thus, RSVP inherently distinguishes between the source and the destination endpoints of data flow. The receiver (or destination) of a data flow is considered the downstream endpoint, while the sender (or source) is considered the upstream endpoint.



Likewise, routers along the data path receiving data from the perspective of the data flow are considered downstream routers. Routers forwarding data from the perspective of the data flow are considered upstream routers.



RSVP messages are transmitted directly on top of the IP protocol. RSVP has its own Protocol ID value, specified in the basic IP header (RSVP = 46). Thus, it does not depend on any other protocols such as TCP for reliable delivery, or UDP for application layer interaction. In other words RSVP messages are transmitted in the payload of IP packets. RSVP is not a routing protocol, nor is it part of the routing architecture of the routers involved in packet forwarding. Furthermore, RSVP is not dependent on a particular routing protocol or architecture. The only requirement RSVP has from the underlying routing algorithm is a simple interface, to be used in determining the next downstream router of a potential flow from the router's routing tables. This information enable the RSVP process to determine the next downstream, router along the data path.

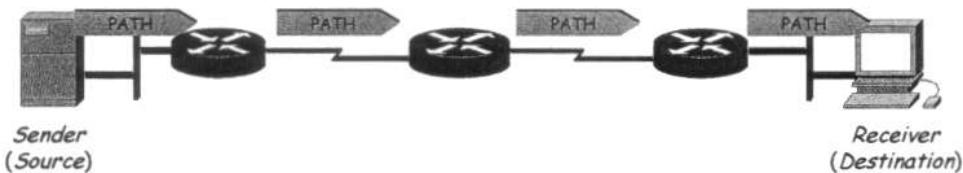


Thus, RSVP relies on the functionality of the underlying routing architecture to discover or pin down a route between the source and destination of data flow. The protocol's messaging follows this route, allowing it to communicate QoS information about a flow to all relevant routers along the data path. Routes can and do change from time to time. When routes change, the data packets will simply follow a new path, and so will the corresponding RSVP messages.

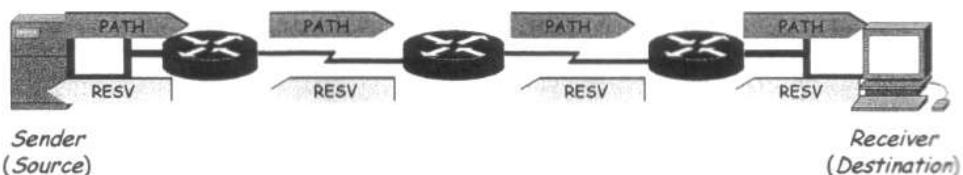
### 3.10 RSVP Base Mechanism

In the simplex data flow scenario, the source application knows the characteristics of the traffic it is capable of sending. This information is encoded in parameters within an RSVP message and is transmitted from the source to the destination. Thus, the destinations, as well as every RSVP-aware router along the path, are aware of the sender's traffic generating capabilities. The RSVP message responsible for communicating the capabilities of the source of a data flow to all

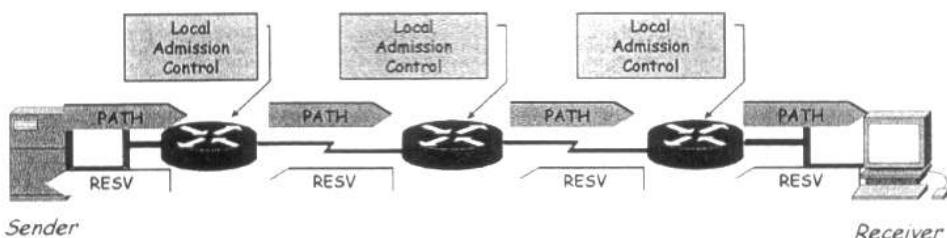
downstream routers and the eventual destination(s) is called the PATH message.



Once the path state has been established, a reservation may be issued (RESV message). This reservation request must then follow, hop-by-hop, the path laid down by the corresponding PATH message. No router will commit its resources to service a data flow until a reservation request has been issued for the flow and accepted by the router along the data path. The RSVP reservation request contains the receiver's QoS specification for the data flow, i.e. the destination of the data flow determines what QoS the flow will actually receive. This determination may be dependent on the application's capabilities, the application requirements and other administrative considerations. Different receivers may specify different QoS requirements for the same data flow. The RSVP message that communicates the reservation request is called the RESV message. Like the PATH message, the RESV message is communicated to all routers along the data path. In the case of the RESV, however, the message is sent hop-by-hop upstream from the destination to the source.

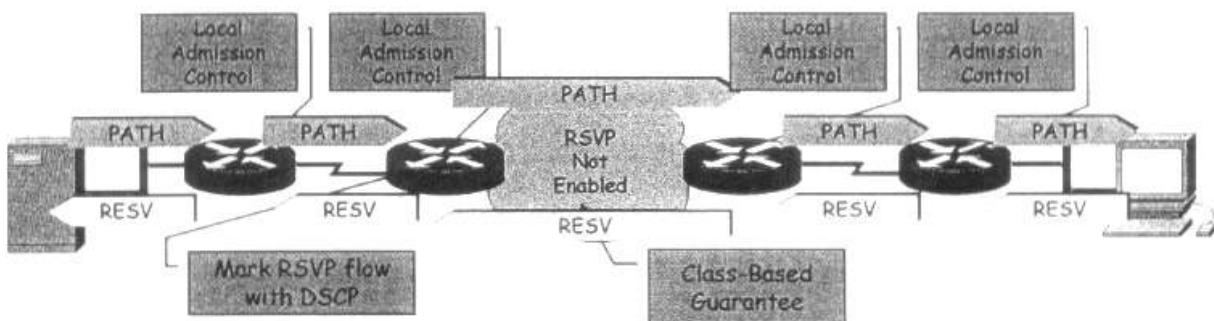


This mechanism avoids the possibility that the reverse path may follow an entirely different route through the network. Through the hop-by-hop reservation setup mechanism, all the RSVP routers along the original path will become aware of the QoS reservation request. Each router along the path may then individually decide to accept and commit the reservation, modifying appropriate parameters in the RESV message before sending it upstream, or refuse the request altogether due to capacity or administrative constraints. All routers have to be enabled for RSVP. Each router determines whether it has enough resources via a local admission control function.



When part of the network may not support RSVP:

- mark RSVP flows with a class of service marker (e.g. IP Precedence or DSCP);
- make sure that the core provides guarantees to the RSVP class.

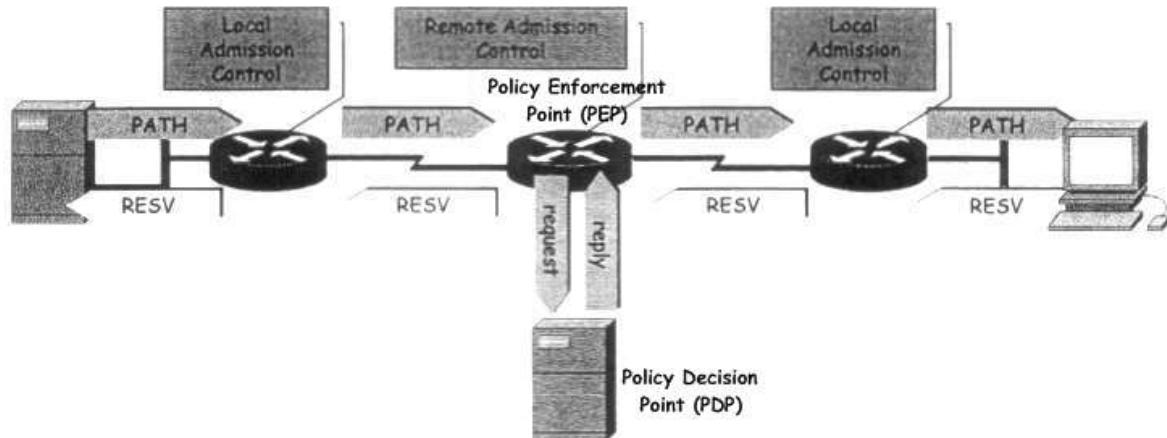


### 3.11 Common Open Policy Service

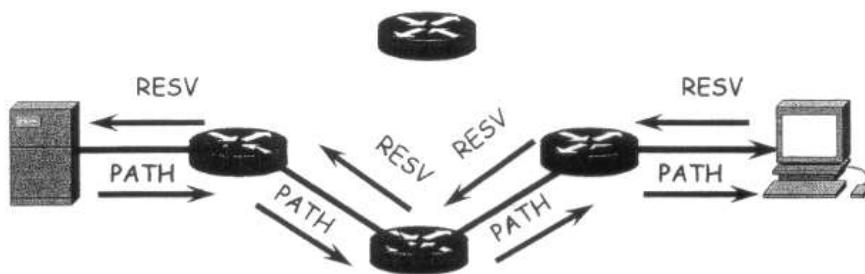
Network devices can use the *Common Open Policy Service* (COPS) to outsource RSVP requests to a remote *Policy Decision Point* (PDP). When a *Policy Enforcement Point* (PEP)

receives an RSVP message, it should notify the PDP via a COPS Request message. The type of request depends on whether the message just arrived, is about to allocate resources from the device, or is being forwarded out of the device. COPS will return the appropriate decision corresponding to the request. The decision may instruct the Policy Enforcement Point (PEP) to accept or reject an arriving RSVP message, allocate resources for a reservation request, or either forward or drop an outgoing message, depending on the request. COPS allows a more centralized approach to building RSVP-enabled networks (more scalable) and provides additional control over who can reserve what. The COPS protocol runs over TCP for reliable message transfer.

RSVP messages flow:



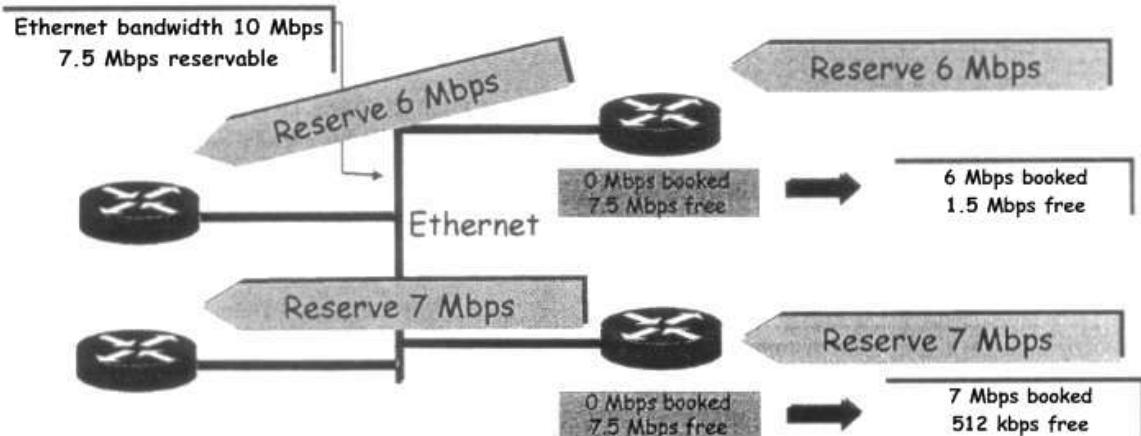
Receiver-Issued Reservations:



### 3.12 Subnet Bandwidth Management (SBM)

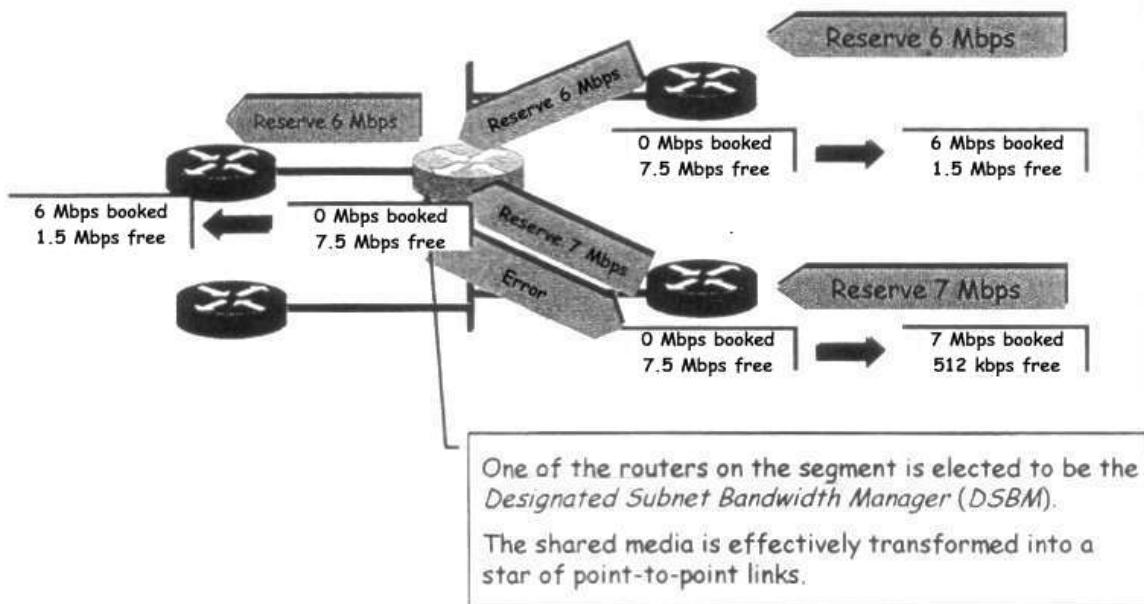
RSVP on shared media can result in oversubscription. SBM is an add-on to RSVP on shared media to prevent it.

Without SBM:



Both routers are within the 75 percent reservable limit. Total reserved bandwidth is 13Mbps (above Ethernet bandwidth). Ethernet should be treated carefully because it is impossible to achieve 100 percent use (collision can occur, depending on implementation).

With SBM:



### 3.13 Soft State Design

The Internet is inherently an unreliable packet switching network. This fundamental constraint must be a consideration in the design of any protocol excepted to operate over the Internet. RSVP is no exception. Still, RSVP is a protocol used to communicate QoS requests through a number of different components along data path. These requests essentially establish state within participating devices along the path. This state is necessary because the routers must be aware of the QoS requirements specified in the request and must allocate corresponding resources, if they are available, for a data flow. When a host no longer requires quality of service from the network, the corresponding RSVP state installed on the routers along the path must be removed. In this way, the resources allocated to service the QoS requirements of the flow can be made available to other data flows. Both PATH and RESV messages install state on routers along a data path. Within the Internet, depending on the routing protocols employed, routes can change relatively frequently. When routes change, the path data takes from its source to its destination will also change. As RSVP needs to install state within all RSVP-aware routers along the data path, it is also necessary that the routers along the new path be updated with the correct path and reservation states corresponding to the rerouted flow. Similarly, routers no longer participating along the path due to a route change must be able to delete the inapplicable path and reservation states so that resources may be made available for the other QoS requests. The RSVP implements the soft state model to perform state management. If an installed state is soft, it is a temporary state. Under this model, state must simply be continuously refreshed or it will be automatically removed. Thus, with respect to RSVP, the PATH and RESV messages must be periodically retransmitted to keep a reservation state active along the data path. This technique is powerful in that it solves the problems associated with lost packets and route changes. To illustrate the RSVP soft state model in action, first consider the state initiated by PATH message which follows the route determined by the routing tables of the forwarding routers along the data path. Route changes are typically incurred due to link or router failures along the data path or the introduction of new routes, both of which are relatively rare situations. The first path state is installed on all RSVP-aware routers between the source and destination. However this path state will not remain permanently. A timeout interval will be specified in the PATH message that limits the period the path state will be remembered by the participating router. This timeout period is reset each time a PATH message for the same data flow is received. Thus, the path state can be persisted by periodically sending PATH messages to refresh the state. As long as the route remains unchanged and the source is still advertising the traffic characteristics of its data flow by sending PATH messages, the state will remain installed on the routers along the data path. The RSVP reservation state is maintained in a similar manner to the path state. The RESV message installs state upstream hop-by-hop in all RSVP-aware routers comprising the data path. The RSVP message installs state upstream hop-by-hop in all RSVP-aware routers comprising the data path. Like the path state, the reservation state will simply timeout if it is not refreshed by periodic RESV messages. Thus, if messages are lost or routes change, the reservation state will be purged automatically. To compensate for lost refresh messages, the timeout period is typically three times the interval between RSVP message

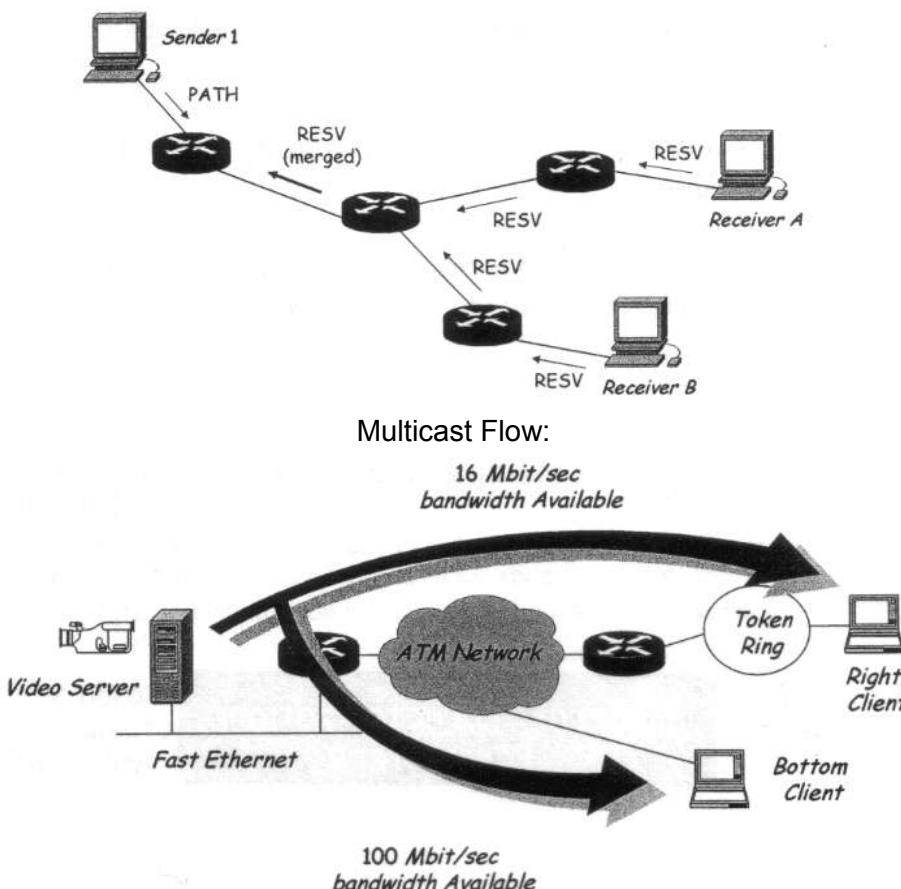
refreshes. Provided that the routers along the path give priority to the RSVP control messages, this redundancy should prove sufficient for keeping state stable along the path.

### 3.14 RSVP Characteristics

RSVP differs substantially from conventional signalling protocols for connection-oriented networks:

- Soft state in contrast to the hard state (found in connection-oriented networks).
- Multicast flows in contrast to unicast flows.
- Receiver-oriented in contrast to sender-oriented resource reservation request.

The soft state and receiver-oriented nature of RSVP give it a number of properties. One property is that it is very straightforward to increase or decrease the level of resource allocation provided to a receiver. Since each receiver periodically sends RESV (refresh) messages to keep the soft state in place, it is easy to send a new reservation that asks for a new level of resources. In the event of a host crash, resources allocated by that host to a flow will naturally time out and be released. Heterogeneous multicasting flows are much simpler to support when the receivers reserve resources.



The bottom client uses the full 100Mbps, while the right client makes do with 16Mbps. If the sender were the system making the reservation, it would have to know the characteristics of all possible receivers, and structure its reservation accordingly. Each receiver need only understand its own capabilities and requirements. Receiver-based registrations also easily support a dynamic environment. New receivers can join the flow, and existing ones can drop out, all without bothering the sender.

### 3.15 General Characterization Parameters

There are a number of general characterization parameters that need to be understood by all IntServ routers. These standardized parameters have a common and consistent interpretation that allows the discovery and specification of QoS end-to-end over heterogeneous network. The NON\_IS\_HOP parameter is used to indicate that there are neither routers along the data path that are legacies and therefore not aware of the Integrated Services' specifications, or routers that simply do not support the requested service. The parameter basically takes the form of a flag that set to indicate the presence of such devices along the path of a data flow. Another useful parameter for the characterization of QoS-aware routers along a data path is the determination of the number of

hops that are IntServ-aware. The NUMBER\_OF\_IS\_HOPS parameter is a simple counter that is incremented by every router along the path of a flow that supports the IntServ architecture. This parameter can range from 1 to 255. The path through the network that the data will follow may be comprised of many different routing devices. The device with the least amount of bandwidth available determines the bandwidth available along the entire path. The AVAILABLE\_PATH\_BANDWIDTH parameter specifies the minimum available bandwidth along the data path. The parameter expresses bandwidth in terms of bytes per second available along the composite data path. The IntServ MINIMUM\_PATH\_LATENCY parameter describes the cumulative smallest latency that would be incurred due to all the devices participating along the path. To calculate this parameter, each router along the path must first determine the smallest possible latency that a packet traveling through it might experience. This delay is basically a factor of the speed of light propagation delay. No packet within the flow can expect to experience a delay less than this cumulative delay. MINIMUM\_PATH\_LATENCY expresses the absolute lower bound on the overall path latency acceptable. The PATH\_MTU parameter represents the maximum transmission unit (i.e. maximum allowed packet size) that can be supported over the data path. Basically, the router along the path that supports the smallest MTU size will dictate the value of this parameter. The PATH\_MTU is not simply a factor of the maximum packet size that can be physically supported by all the routers along the path, rather this parameter has an impact on the QoS associated to the data flow. In fact, the packets must be small enough that the multiple packets can be efficiently multiplexed without creating excessive delay. Large packets will take longer to transmit, and, thus, potentially produce excessive jitter in smaller packets. Packet size is extremely important in effectively supporting QoS. The parameter used by end devices (i.e. Hosts) to request QoS guarantees from the network is the TOKEN\_BUCKET\_TSPEC parameter. The TOKEN\_BUCKET\_TSPEC parameter is actually comprised of multiple values that characterize the data flow in detail. These are:

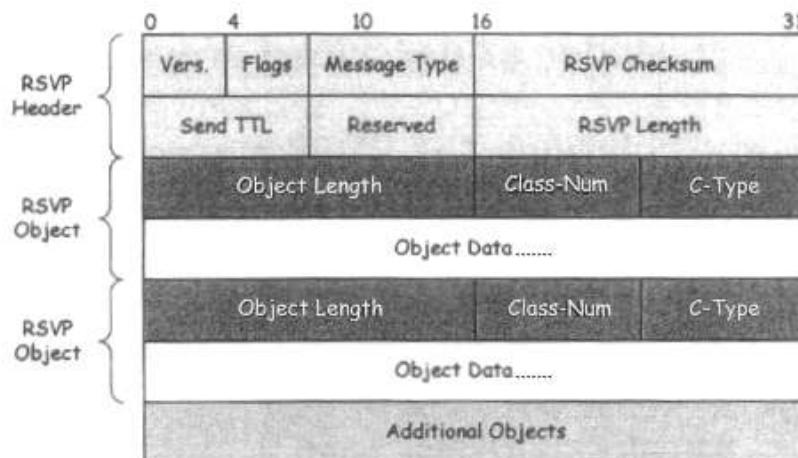
- the token rate ( $r$ )
- the token bucket ( $b$ )
- the peak rate ( $p$ )
- the minimum policed unit ( $m$ )
- the maximum packet size ( $M$ )

### 3.16 RSVP Messages

There are seven RSVP messages:

- PATH
- RESV
- PATH Error
- RESV Error
- PATH Tear
- RESV Tear
- Confirmation

Each of these messages is comprised of the RSVP message header, followed by a set of objects. The objects contain the information necessary for describing the message in question. An RSVP message basically contains a header that identifies the RSVP Version, various Flags, the Message Type, and the Length of the message. The header also has a Checksum field for verifying that the message is not corrupted, and a sender TTL field useful for determining the number of RSVP-aware hops. The current RSVP version is 1.



The one-byte type of message field is used to determine if the RSVP message is a PATH or RESV, PATH or RESV Error, PATH or RESV Tear, or RESV Confirmation message. The two byte Length field is used to determine the size of the entire RSVP message in bytes. The size of an RSVP message is fundamentally limited to 64Kbytes due to the packet size restrictions imposed by the underlying IP protocol. Further RSVP message size limitations due to the underlying MTU size of the path may also be necessary.

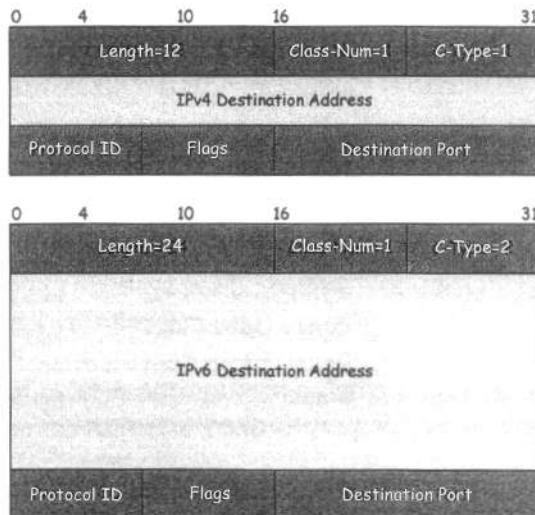
RSVP is fundamentally an extensible protocol. It achieves this extensibility through the use of type/length/value format objects. An RSVP object is simply a variable-sized chunk of data that is given a type for purposes of identification. The RSVP object header consists of a two-byte Length field followed by a one-byte Class Number and one-byte Class Type field. The Length field simply determines the size of the object in bytes including the object header length. The Class Number and Class Type effectively categorize and identify the structures contained in the data portion of the object. The Class Number can be thought of as a general characterization of the object data while the Class Type represents the specific data format. RSVP message itself then begins with the RSVP message header as described previously. Following the RSVP message header are the individual objects that comprise the data describing the RSVP message. This generic format is useful for defining all the RSVP messages. The appropriate data objects will simply be included in their corresponding RSVP message. Fourteen classes of objects are currently defined in the RSVP specification. These objects comprise the attributes that constitute the RSVP messages. Each class of object in RSVP can contain multiple types that further specify the format of the encapsulated data.

#### List of classes of objects:

- Session Classifier
- RSVP Hop Classifier
- Integrity Classifier
- Time Values Classifier
- Error Specification Classifier
- Scope Classifier
- Style Classifier
- Flow Specification Classifier
- Filter Specification Classifier
- Sender Template Classifier
- Sender TSpec Classifier
- AdSpec Classifier
- Policy Data Classifier
- Reservation Confirmation Classifier

### 3.17 Session Class

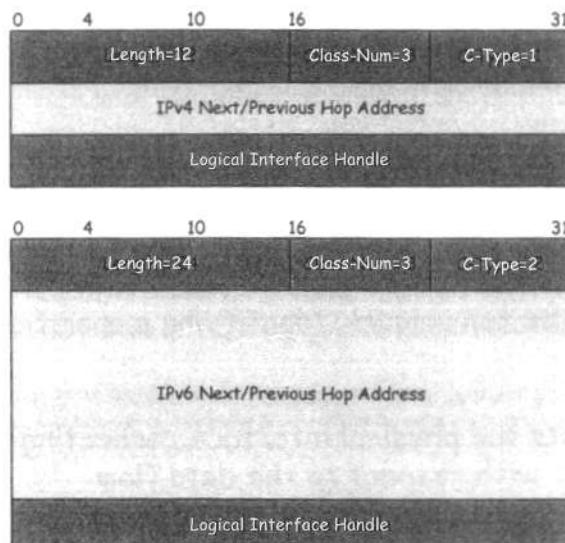
The *Session Class of objects* specifies the destination IP address (unicast or multicast), protocol ID, and destination port of a data flow. The class number for this object is 1. There are two types of session objects currently defined – the IPv4 class type and the IPv6 class type – which support the IPv4 and IPv6 style of addressing, respectively.



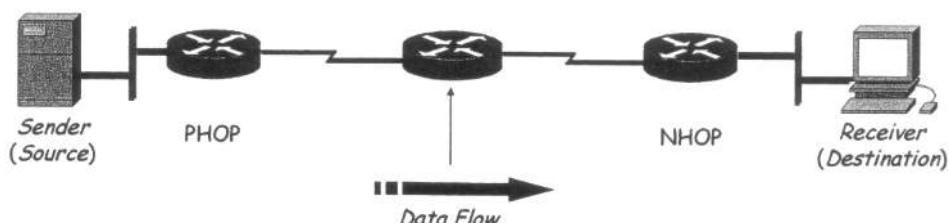
The Session object is one of the objects used to identify a flow. In order to identify a flow, its protocol type, destination address, and destination port must all be known. The Session object simply provides this information to accurately describe the destination of a particular flow. It is required attribute of all RSVP messages, as they all are used to relay QoS information about a particular flow.

### 3.18 RSVP Hop Class

The *RSVP Hop Class of objects* is used to identify neighboring RSVP-aware routers along the data path. A router's neighboring RSVP hop is the closest RSVP router upstream or downstream along the data path.



A router's next hop (NHOP) is considered the next neighboring router downstream from the perspective of a data flow. A router's previous hop (PHOP) is considered the previous neighboring router upstream from the perspective of a data flow. The Hop Class object is basically comprised of an IP address and a logical interface handle.



The two class types defined for the RSVP Hop Class provide address formats compliant with the IPv4 and IPv6 addressing, respectively. The logical interface handle attribute is represented in both object types. This parameter is useful for uniquely identifying a specific interface on a device. This number represents the physical interface connecting the neighboring router with respect to the data flow.

### 3.19 Time Values Class

The *Time Values Class* of objects is used to provide timing information about RSVP messaging. Since RSVP is a soft state protocol, messages must be periodically refreshed. It is useful to control this interval between message refreshes (for routers where many flows are handled, too many refreshes can lead to degradation in performance).

0	4	10	16	31
Length = 8		Class-Num = 5	C-Type = 1	
Refresh Period R				

### 3.20 Flow Specification Class

The *Flow Specification Class* defines the IntServ-compliant characteristics of a data flow. This class is used in RESV messages to define the receiver's requirements for the data flow. Controlled Load and Guaranteed Service are characterized by two different Flow Specification objects.

Controlled Load, Flow Specification object:

0	4	10	16	31		
Length = 36		Class-Num = 9	C-Type = 2			
Version Number	Reserved	IS Length				
Service Number	Reserved	Service Data Length				
ParamID=127	Parameter Flags	Parameter Data Length				
Token Rate						
Token Bucket Size						
Peak Data Rate						
Minimum Policed Unit						
Maximum Packet Size						

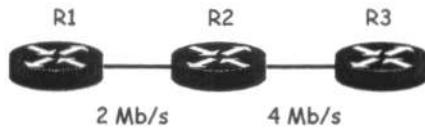
Guaranteed Service, Flow Specification object:

0	4	10	16	31		
Length = 48		Class-Num = 9	C-Type = 2			
Version Number	Reserved	IS Length				
Service Number	Reserved	Service Data Length				
ParamID=127	Parameter Flags	Parameter Data Length				
Token Rate						
Token Bucket Size						
Peak Data Rate						
Minimum Policed Unit						
Maximum Packet Size						
ParamID=130	Parameter Flags	Parameter Data Length				
Rate						
Slack Term						

### 3.21 Slack Term

Recall that in the delay calculation for Guaranteed Service, the bandwidth reserved for a

particular flow is identical at all hops along path. For example, suppose that there are two links between the sender and the receiver and that to meet the delay bound, we need 2.5Mbps. Now, the amount of bandwidth available on the first link is 2Mbps and on the second is 10Mbps. This reservation will fail since the first link does not have sufficient bandwidth. However, one possible solution is to reserve different amounts of bandwidth on these two hops: increase the bandwidth reservation on the second hop and reduce the bandwidth reservation on the first hop.

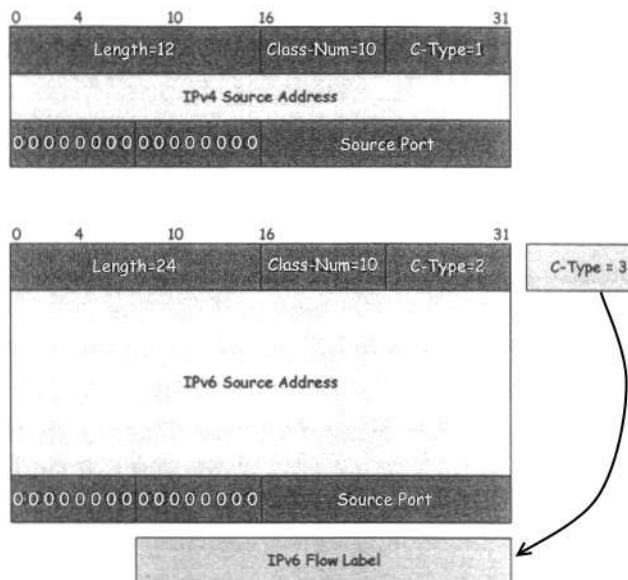


By doing so, we can reduce the delay in the second link to compensate for the increased delay in the first link. As long as we can meet the overall delay requirement, the reservation can go through. The *Slack Term* signifies the difference between the desired delay and the actual delay obtained with the current bandwidth reservation. In other words, the Slack Term is the credit in delay (the amount of delay that can be increased without violating the overall end-to-end delay) in the previous hops that may be used by other hops.

### 3.22 Filter Specification Class

The *Filter Specification Class* is used by RESV messages to identify a particular source of a data flow. It is most useful in the multicast case where multiple sources are contributing to the same multicast session. This object is typically used in conjunction with the Flow Specification object to describe the desired QoS a particular data flow should receive. Note: Please compare the Filter Specification Class with the Session Class which describes the destination of a particular flow. The Filter Specification Class object is comprised of the source's IP address and additional demultiplexing information. Three types of this object are defined: The first type is composed of an IPv4 format address and a field for specifying the source application port. The other two types are comprised of an IPv6 format address field with a source application port in one case, and an IPv6 flow label in the other.

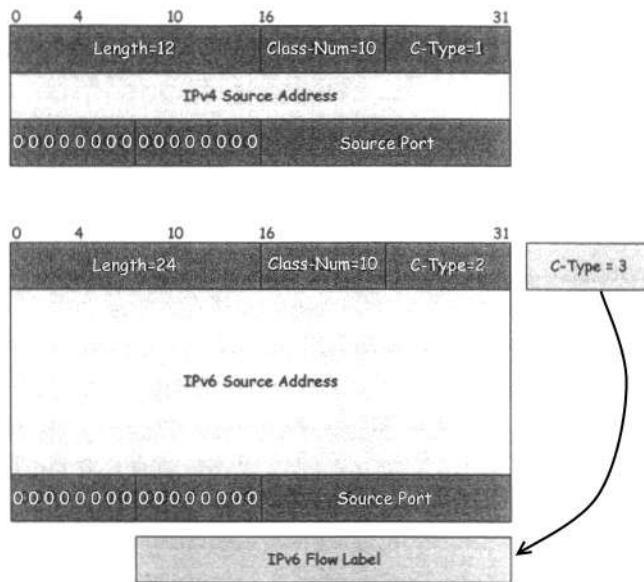
RSVP Filter Specification object:



### 3.23 Sender Template Class

This object is required to uniquely identify a flow's source in PATH messages. It effectively identifies the sender host and originating application for the PATH message. Three types of this object are defined: the first type is composed of an IPv4 format address and a field for specifying the source application port. The other two types are comprised of an IPv6 format address field with a source application port in one case, and an IPv6 flow label in the other.

RSVP Sender Template object:



### 3.24 Sender TSpec Class

The *Sender TSpec Class object* is used by the source of a data flow to specify the traffic characteristics of its data flow. It is a required object in PATH messages, since it is used to describe the data flow generated by a source. The attributes for this object include:

- the token rate
- token size
- peak rate
- minimum policed unit, and
- maximum packet size.

RSVP Sender TSpec object:

0	4	10	16	31		
Length = 36				Class-Num = 12		
C-Type = 2		IS Length				
Version Number	Reserved	IS Length				
Service Number	Reserved	Device Data Length				
ParamID=127	Parameter Flags	Parameter Data Length				
Token Rate						
Token Bucket Size						
Peak Data Rate						
Minimum Policed Unit						
Maximum Packet Size						

### 3.25 Policy Data Class

The *Policy Data Class object* encapsulates information useful for making policy decisions with respect to an RSVP message. Information in this object may be used to authenticate a user, provide credit card billing information, or provide other useful policy-related data. This object is optional in the PATH, RESV, PATH Error, and RESV Error messages.

### 3.26 AdSpec Class

Although the Sender TSpec sufficiently describes the traffic characteristics of the data flow as sent by the source, it does not contain information about the characteristics of the routers along the data path. We have already seen a number of IntServ parameters that describe such characteristics. These parameters include:

- NON\_IS\_HOP
- NUMBER\_IS\_HOPS
- AVAILABLE\_PATH\_BANDWIDTH
- MINIMUM\_PATH\_LATENCY
- PATH\_MTU

The *AdSpec object* is contained in the PATH message and is manipulated hop-by-hop along the data path. This information is used by the receiving application to determine the available service types, the minimum capacity of the data path, and other characteristics of the data path. This information, in turn, is used to construct an appropriate reservation for the data flow. The format of the AdSpec object is divided into three functional blocks.

1. The first block is used to represent the general characterization parameters of the flow.
2. The second is used to specify the availability of the Controlled Load Service; and
3. The third is used to advertise the Guaranteed Service and describe its parameters.

The *General Characterization Parameters Block* carries values related to the following parameters described above:

- NUMBER\_OF\_HOPS
- AVAILABLE\_PATH\_BANDWIDTH
- MINIMUM\_PATH\_LATENCY
- PATH\_MTU

The *Controlled Load Service Block* typically has no values besides a header. Typically, the Controlled Load Block is included only to advertise to the destination that Controlled Load is an acceptable service type for the flow.

The *Guaranteed Service Block* provides delay bound information about the data path. Routers along the data path support Guaranteed Services will modify the bound information to describe their local worst case delay characteristics. Any router that does not support Guaranteed Service and therefore cannot update the delay bounds values must set the Service Not Supported bit in the Guaranteed Service block's header. The Guaranteed Service Block contains information about two delay terms: *C* and *D*. *C* and *D* depend upon the scheduler. For a WFQ scheduler:

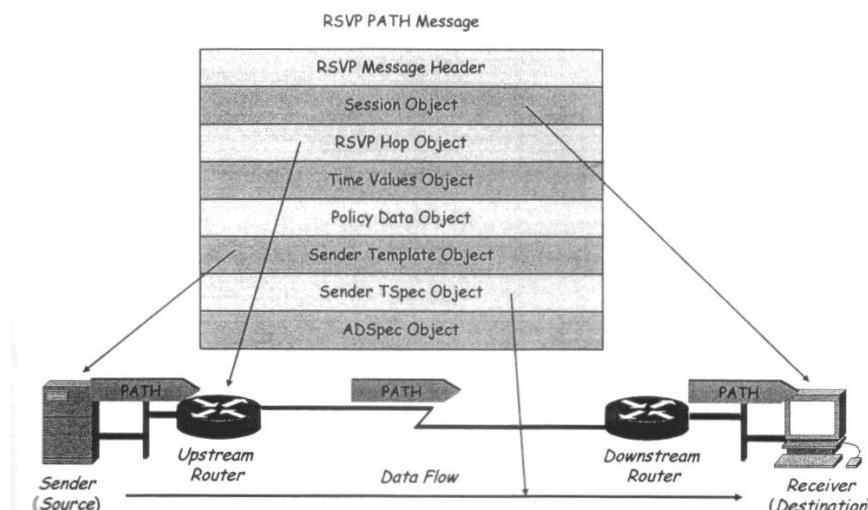
$$C_n = M \text{ (maximum packet size for a flow } n) \text{ and } D_n = \frac{L_{max}}{v_n} = \frac{MTU}{Link\_Capacity}$$

These terms are modified hop-by-hop along the data path. Each device will add its local delay characteristics to the received *C* and *D* terms to produce a total end-to-end delay bound for the entire data path, *C<sub>tot</sub>* and *D<sub>tot</sub>*.

$$C_{tot} = \sum_{n=1}^N C_n \quad D_{tot} = \sum_{n=1}^N D_n$$

### 3.27 The PATH Message

The two RSVP messages involved with path setup and resource reservations are the PATH and RESV messages respectively. The *PATH message* is used to discover the path of a data flow and bind this route for use by RSVP. It is generated by the data flow's source and is eventually received by the data flow's destination, or destinations in the multicast case. The PATH message allows the RSVP routers along the data path to discover their RSVP-aware neighbors, advertise their capabilities, and *install state* relevant to the data flow.

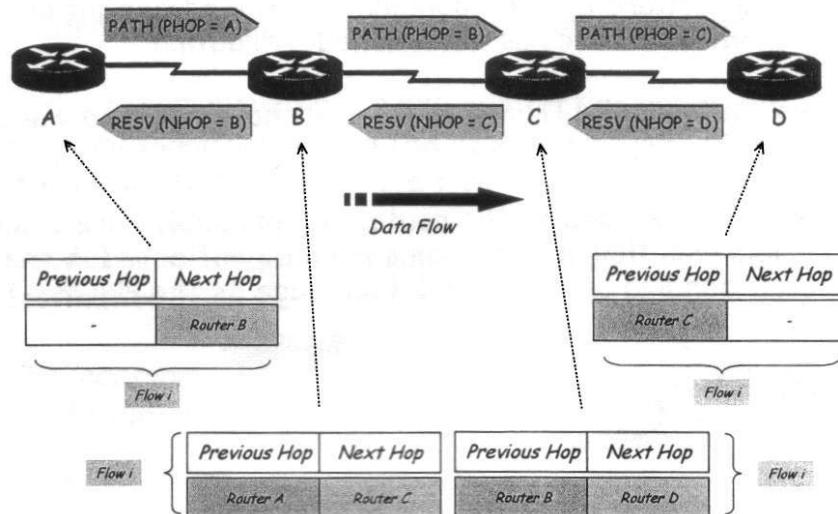


The PATH message is addressed from the source to a destination at the IP header level. The destination address may be unicast or multicast. PATH messages are addressed this way so they will be routed appropriately by RSVP-aware and RSVP-unaware devices alike. A PATH message also specifies Protocol ID=46 in the IP header. RSVP-aware devices recognize Protocol ID=46 as the RSVP protocol. Intercepted messages will then be delivered to the RSVP process on the device. If an RSVP message received by a router is a PATH message, the RSVP process interprets the message and will establish a path state for the message if one doesn't already exist.

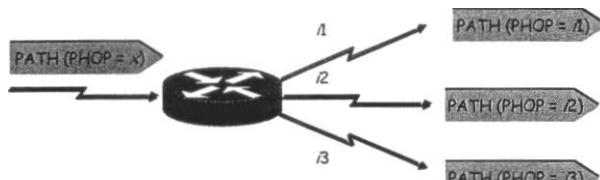
The PATH message contains:

- a *Session object* describing the destination address and port information.
- a *Sender Template* and *Sender TSpec object* identifies the source (sender host and originating application) and the traffic characteristics of the data flow.
- an *RSVP Hop object* that identifies the interface of the previous RSVP-aware router along the data path (PHOP) that generated the PATH message (this is effectively the neighboring upstream RSVP-aware router).

The Session, PHOP, Sender Template and Sender traffic information uniquely identify the path state for a flow.



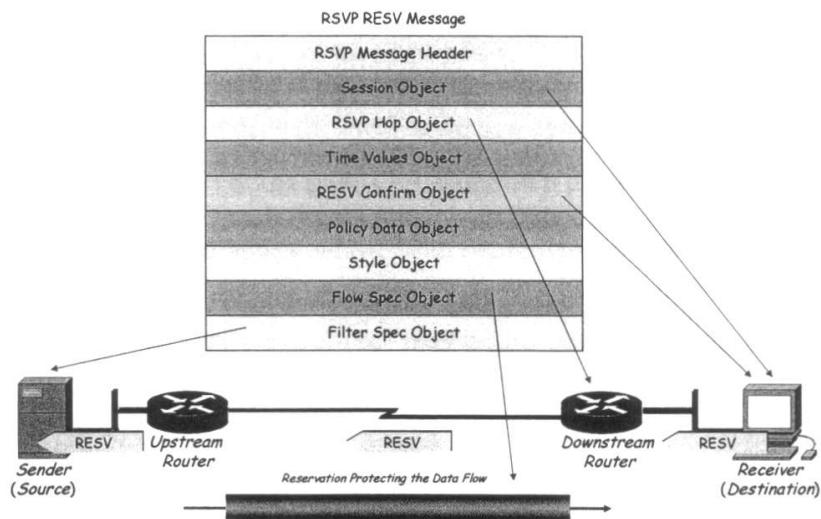
The Time Values object is included in the PATH as well, and describes the refresh interval of the PHOP. Based on this refresh timer, a device can determine when to timeout the associated RSVP path state. An AdSpec object, if included, is used to describe the kind of services and resources available on the participating upstream routers. Finally, a Policy Data object may be included in the PATH specifying any policy information relative to the source or PHOP of the data flow. Once an RSVP path state is installed on a router, the router should forward a corresponding PATH message out all relevant interfaces as determined by routing. Such outgoing PATH messages must have updated values for the HOP. As each interface on the router will probably have a unique IP address, this information should be reflected in the corresponding outgoing PATH message as the PHOP.



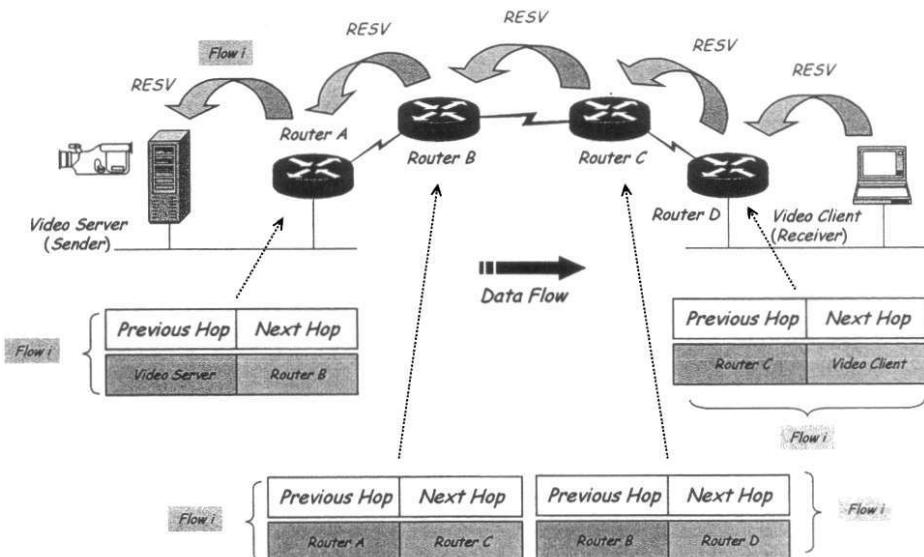
The installed path state must be refreshed periodically in accordance with the value specified within the Time Values object. Simply sending a refresh PATH message to all applicable downstream RSVP devices accomplishes this task of keeping the path state alive in the neighboring downstream device.

### 3.28 The RESV Message

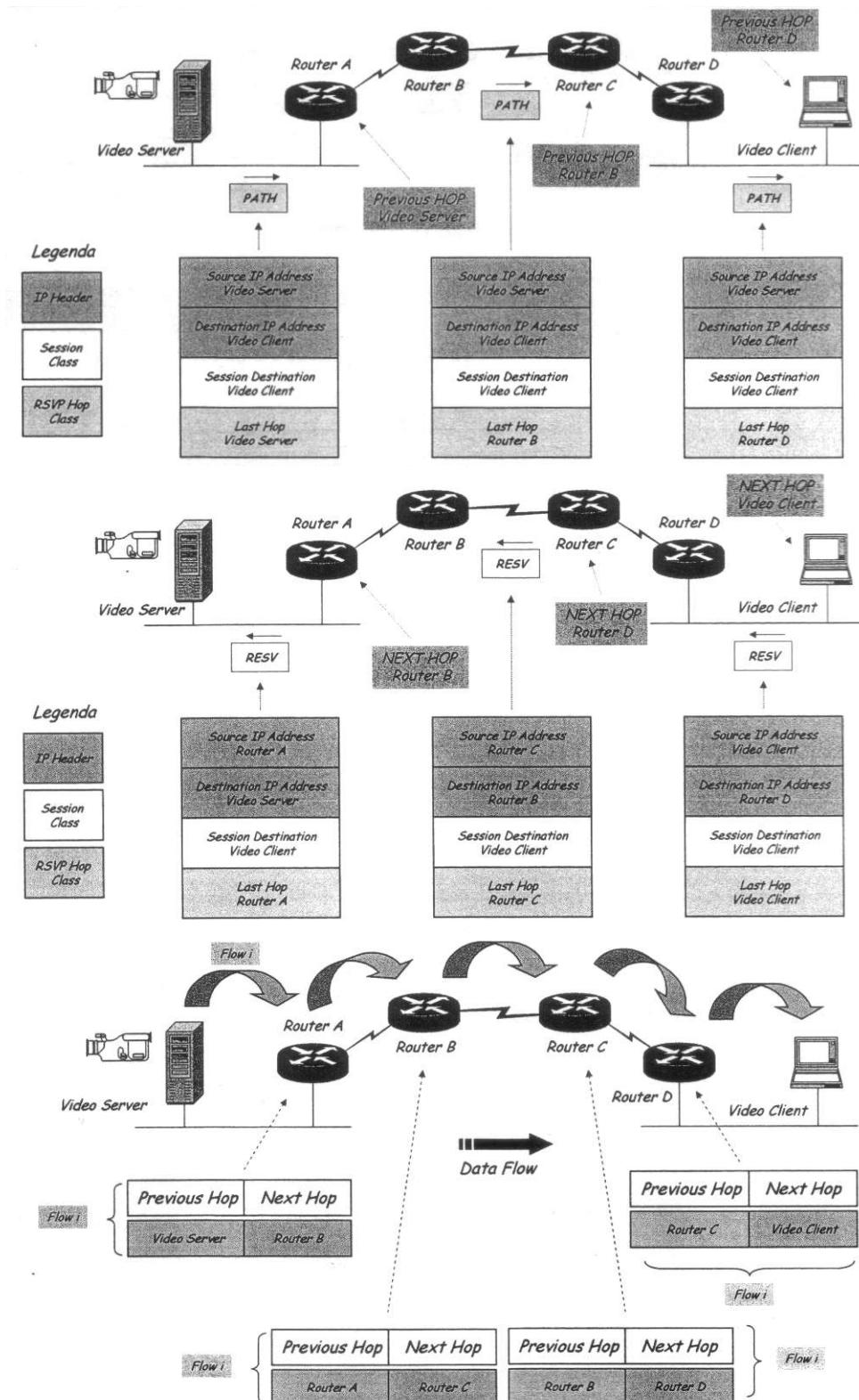
Once a path state is installed along the data path, a destination may opt to make a reservation request. This is accomplished when the destination generates an RESV message specifying the requested QoS characteristics for the data flow. The RESV message is transmitted hop-by-hop from the destination back to the data source. The message simply follows the reverse of the path established by the PATH messages.



The RESV messages are addressed from the sending downstream hop to the previous upstream hop. The IP header source address field simply contains the address of the router that generated the RESV message and the destination address contains the unicast address of the PHOP to which the RESV is destined. The PHOP information is obtained from the corresponding path state installed on the sending router for the same flow. When a PHOP receives a RESV message it will be interpreted by the router's RSVP process.



This process will check to see if there is at least one valid path state installed for the same session. If a valid path state exists, the reservation will be processed and, if admission control succeeds, a reservation state will be installed. This implies that resources will be applied to service the corresponding flow. The RESV message contains the Session object that identifies the destination address of the data flow. An RSVP Hop object is included that describes the downstream device interface that produced the RESV message (NHOP). A Time Values object describes the refresh interval of the NHOP. Optionally, a Reservation Confirmation object may be included if the destination wishes to have the reservation explicitly confirmed. Additionally, a Policy Data object may be included to describe policy information relevant to the destination or NHOP. Once a reservation state is successfully installed on a forwarding device, the device must prepare outgoing RESV messages for this state for all applicable outgoing interfaces. The HOP object should be set to the forwarding interface out which the RESV is sent. The HOP object is interpreted by the upstream RSVP-aware device to determine what NHOP generated the message. As with the PATH message, the Time Values object should reflect the refresh interval of the downstream RSVP-aware device.



### 3.29 RSVP Components

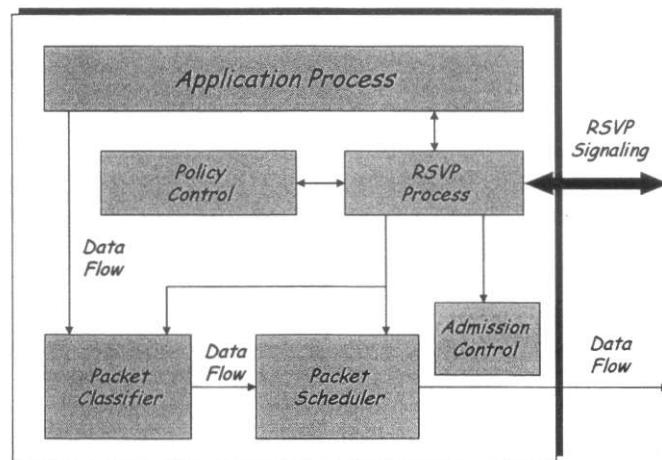
There are several components necessary to properly support RSVP on a device. The exact components vary somewhat depending on whether the implementation is located on an end host or a forwarding device such as a switch or a router. Besides a process required to handle the RSVP messaging, a method is needed:

- to classify and control traffic, and
- to provide both policy and capacity admission control, as well as
- to interact with the underlying routing architecture.

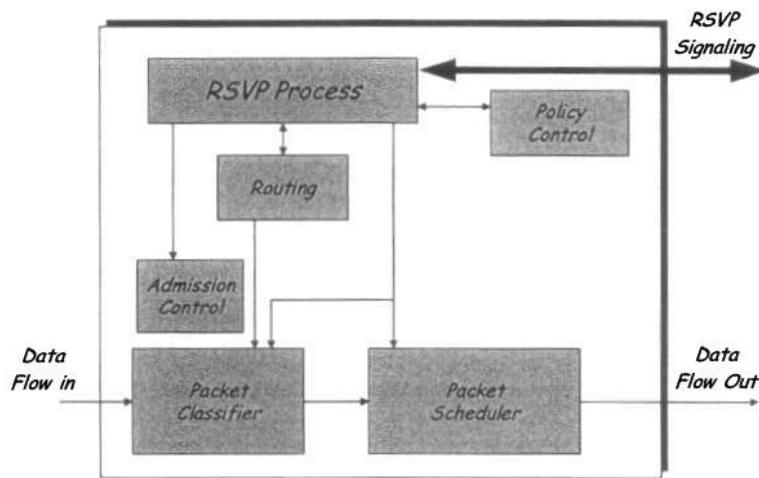
In RSVP, the end hosts must be able to signal their QoS requests to the network infrastructure. The network, in turn, must be able to either successfully provision the QoS request or signal an error condition. This process involves every RSVP-aware device along the data path. Each device must successfully allocate resources on all of its relevant forwarding interfaces or notify the

requesting host of any failure.

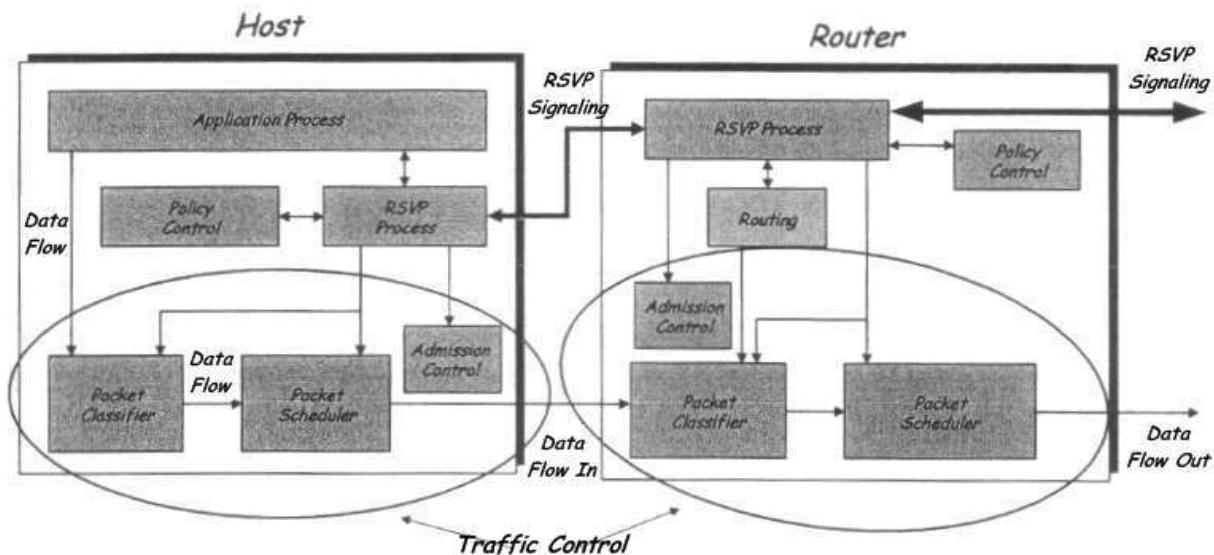
RSVP Components on a host:



RSVP Components on a router:



Interaction between host and router:



### 3.30 Application Process

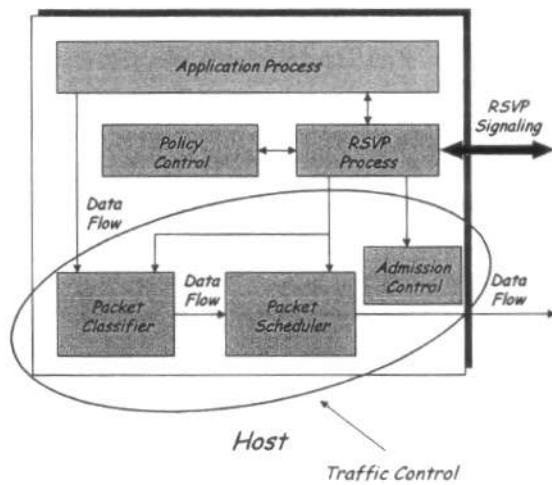
An *application* is the process running on a host that is the ultimate source or destination of a data flow. It is the application that actually generates a data flow and knows the characteristics of its traffic. Additionally, the application that receives a data flow knows the desired properties of the flow. Examples of applications range from interactive, videoconferencing applications that have specific requirements for network bandwidth and delay, to streaming or presentation applications

that simply have bandwidth requirements.

If an application is sending traffic to be protected, the application should have the ability to cause a PATH message specifying the characteristics of the traffic to be sent. If an application is receiving a flow and desires the flow to receive protection from the network infrastructure, the application should be able to create an RESV message specifying the QoS it wishes to receive. If the application is to act as both the source and the destination of a data flow, both PATH and RESV messages will have to be issued on its behalf. In this case, the PATH will describe the characteristics of the traffic the application will generate, while the RESV will request the QoS desired for any incoming traffic from applications on other hosts.

### 3.31 RSVP Process

The purpose of the RSVP process running on a host is to handle the specifics of RSVP signaling. This process must be able to interact with applications requiring QoS support from the network. It must have access to the network interface on the host and be able to deliver raw RSVP messages to the network.



The RSVP process keeps all states relevant to the establishment and maintenance of path and reservation state. The application will typically inform the RSVP process of its requirements and characteristics through a basic Application Programming Interface (API). Based on information provided by the application through such an interface, the RSVP process will know whether to establish a path or reserve state or to remove an existing state. The RSVP process will then continue to refresh the path or reserve state on behalf of the application.

### 3.32 Traffic Control

On the host, *traffic control* is comprised of three components:

- admission control,
- packet classification, and
- packet scheduling.

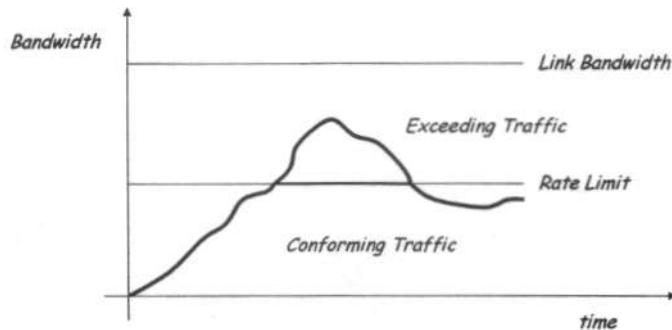
These components are responsible for classifying packets that are to be protected, keeping the allocated resources within acceptable limits, and scheduling packets for transmission consistent with their corresponding QoS specification.

### 3.33 Admission Control

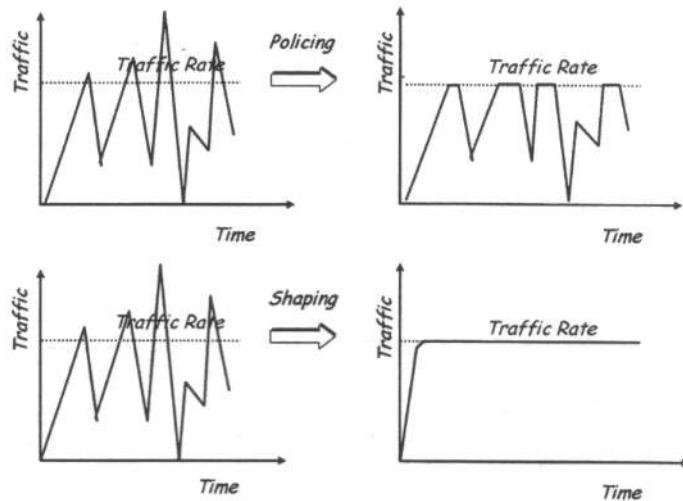
The *admission control* (which must be consistent with the service model) contains the decision algorithm that a host system and a router use to determine if there are enough resources to accept the requested QoS for a new flow. If there are not enough free resources, accepting a new flow would impact earlier guarantees and the new flow must be rejected. Admission control is invoked at the (source) host system and at each router along a reservation path, to make a local accept/reject decision at the time a host requests a real-time service. When a resource request is torn down or removed due to a soft state timeout, the corresponding allocated resources will be made available for use by future resource requests.

### 3.34 Policing

*Admission Control* should not be confused with *Policing*. The former is a per-flow decision to admit a new flow or not. The latter is a function applied on a per-packet basis to make sure that a flow conforms to the TSpec that was used to make the reservation. If a flow does not conform to its TSpec - for example, because it is sending twice as many bytes per second as it said it would - then it is likely to interfere with the service provided to other flows, and some corrective action must be taken. Routers use the token bucket model to keep track of packet arrival rate. The token bucket model is used whenever a new packet is processed. The return value is conform or exceeds.



There are several options, the obvious one being to drop offending packets. However, another option would be to check if the packets really are interfering with the service of other flows. If they are not interfering, the packets could be sent on after being marked with a tag that says, in effect, “*This is a nonconforming packet. Drop it first if you need to drop any packets*”.

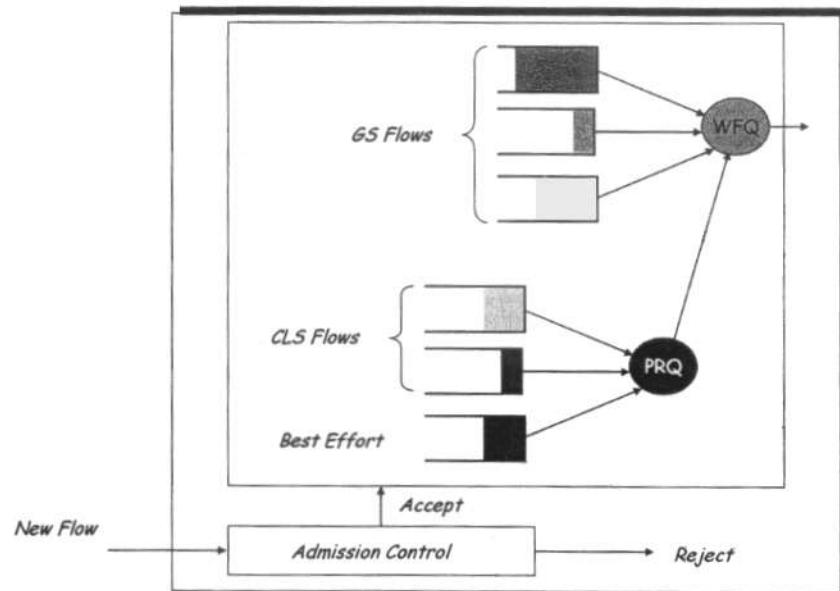


### 3.35 Packet Classifier

Once a RESV message for a flow has been accepted by admission control, a filter identifying the flow's traffic will be installed in the packet classifier. The header information to which the filter is typically compared includes the source and destination IP addresses, protocol ID, and the source and destination ports specified in the packet. Together, these five fields are called the five-tuple. RSVP may further use the FlowLabel field in the IPv6. For example, an outgoing packet may be classified into the Controlled Load classes, or it may be part of a Guaranteed QoS flow that needs to be handled separately from all other Guaranteed QoS flows.

### 3.36 Packet Scheduler

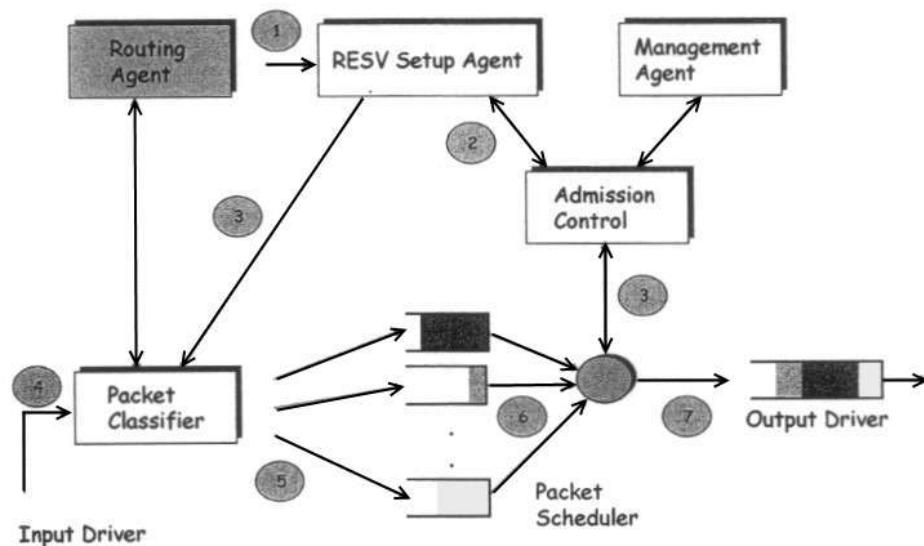
Once a packet has been classified as belonging to a QoS flow, the packet scheduler will guarantee that its QoS requirements are satisfied. In the case of Guaranteed Service (GS), a WFQ scheduler may be used while for Controlled Load Service (CLS), simpler schedulers can provide the requested QoS. The Controlled Load traffic is treated as a single, aggregated flow (as far as the scheduling mechanism is concerned), with the weight for that flow being set based on the total amount of traffic admitted in the Controlled Load class.



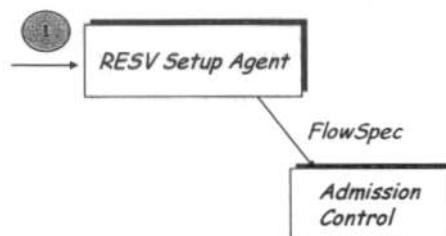
### 3.37 Policy Control

This is the component that is responsible for deciding who gets to make reservations and who doesn't. It is responsible for enforcing administratively imposed constraints determining the usage for RSVP.

### 3.38 Traffic Control Overview

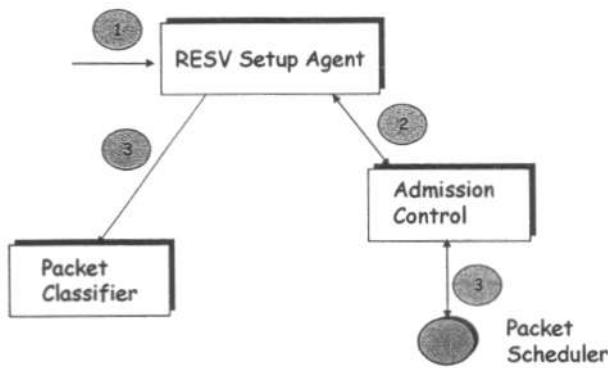


Step 1:



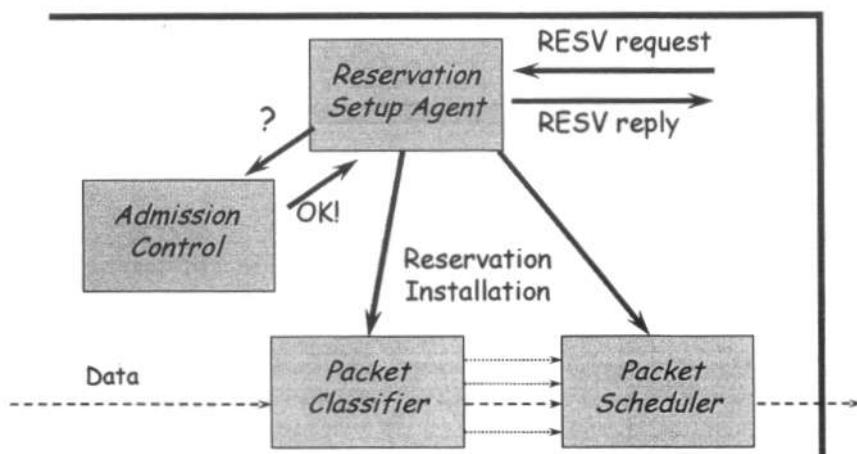
1. An RESV message arrives at the RESV Setup Agent in the router. After the RESV message is processed to verify its correctness and to extract its contents, the RESV Setup Agent invokes the Admission Control Module. The RESV Setup Agent passes the FlowSpec to the Admission Control Module.

Steps 2, 3:

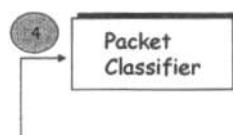


2. The Admission Control Module uses its admission control algorithm and information on available resources to determine whether or not to accept the RESV request. If the request is accepted, the Admission Control Module returns its decision to the RESV Setup Agent.
3. After successful admission the RESV Setup Agent passes the packet filtering information (five-tuple) to the Packet Classifier that describes how to identify the packets in the admitted flow. The Packet Classifier maintains an internal database of such filtering information for several flows. Similarly, the Admission Control Module also contacts the Packet Scheduler and passes it information such as the FlowSpec parameters (e.g., token bucket parameters) so that the scheduler can schedule the packets in the flow appropriately.

#### Overview of the Reservation Setup Procedure:

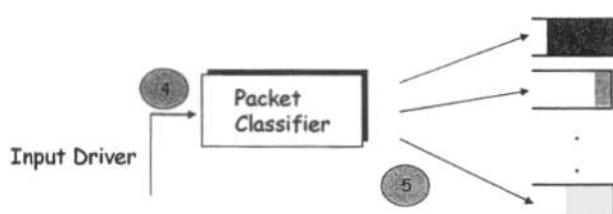


#### Step 4:



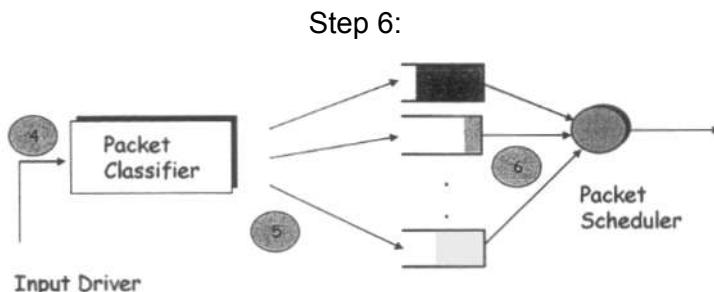
4. As the traffic arrives at a router, the input driver receives each packet and passes the packet to the Packet Classifier.

#### Step 5:

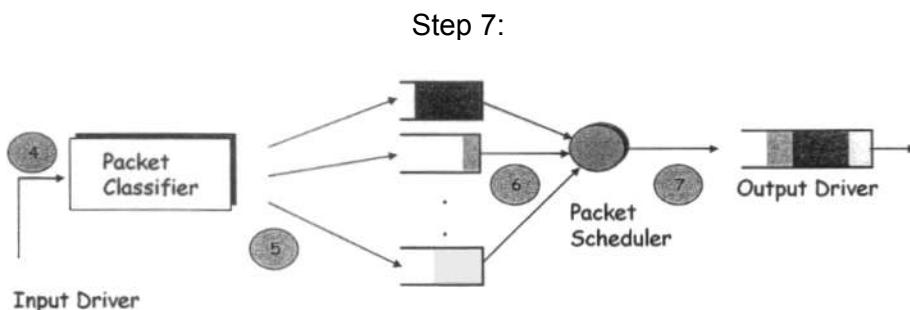


5. The Packet Classifier searches its filtering database to identify the flow to which the packet belongs and then enqueues the packet in an appropriate queue. Depending on the scheduling mechanisms used, the scheduler may use a separate queue for each flow, or flows may be aggregated into a single class or queue for scheduling purposes. In some

implementations, the arriving packet may first be passed to the packet scheduler and the scheduler may invoke the classifier as part of its processing.



- Given one or more input queues, the Packet Scheduler uses its scheduling algorithm to pick a packet from one of the queues and schedules it for transmission over the outgoing link.



- When the time for packet transmission arrives according to the schedule, the scheduler passes the packet to the output driver for transmission. More than one packet may be queued to be transmitted at the output link, depending on the speed and nature of the transmission link.

### 3.39 RSVP versus ATM

ATM Service Specification:	RSVP Flow Specification:
Average Bit Rate (SCR)	Average Bit Rate ( $r$ )
Peak Rate (PCR)	Peak Rate ( $p$ )
Emission Burst (MBS)	Burst/Bucket Size ( $b$ )
UBR/ABR	Best Effort
CBR or rt-VBR	Guaranteed
nrt-VBR or ABR (with min cell rate)	Controlled Load

### 3.40 Scalability Issues

While the IntServ architecture and RSVP represented a significant enhancement of the best-effort service model of IP, many Internet Service Providers (ISPs) felt that it was not the right model for them to deploy. The reason for this reticence relates to one of the fundamental design goals of IP: scalability. In general, the term scalability is used to indicate how quickly some measure of resource usage increases as a network gets larger. For example, in a large IP network such as an Internet Service Provider backbone, we are likely to be concerned with how large the routing tables are, since they consume memory in routers, processor power (to update), and perhaps link bandwidth (when update information is transmitted). For example, it is desirable for routing tables to grow more slowly than the number of users attached to the network. Making reservations for individual flows is clearly a bad idea for scalability. We might reasonably expect that each user will make reservations at some average rate, so the number of reservations that might be made across a large network is likely to grow about as fast as the number of users of the network. To understand the severity of this problem, suppose that every flow on an OC-48 (2.5Gbps) link represents a 64Kbps audio stream. The number of such flows is:

$$\frac{2.5 \cdot 10^9}{64 \cdot 10^3} = 39000$$

Each of those flows needs some amount of state that needs to be stored in memory and refreshed periodically. The router needs to classify, police, and queue each of those flows. Admission control decisions need to be made every time such a flow requests a reservation. Other approaches that do not require so much per-flow state have been developed.

### **3.41 Note on RSVP**

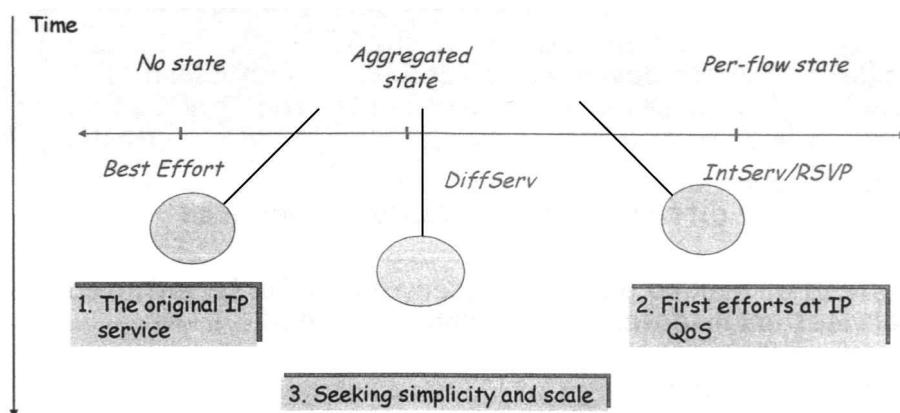
One of the most widely held beliefs about RSVP is that it suffers from certain scalability problems. In fact, this characterization is not strictly accurate. RSVP was originally designed to support resource reservations for individual application flows - microflows -and this is a task with inherent scalability challenges. Any protocol that attempted to make reservations at this level of granularity would face similar issues. Therefore, it is much more accurate to say that microflow reservations scale badly than to say the same for RSVP. This distinction is particularly important when we consider that RSVP is not required to make reservations only for individual microflows, but can also make reservations for highly aggregated traffic. In fact, RSVP with extensions is also used by MPLS Traffic Engineering to establish MPLS/TE tunnels (initiated by a router). In 1998, the DiffServ WG was formed under IETF. DiffServ provides traffic differentiation by classifying traffic into a few classes, with relative service priority among the traffic classes. Unlike the IntServ model, an application using differentiated services does not explicitly signal the router before sending data. RSVP and WFQ are supported since '95. RSVP signaling for VoIP calls is supported on all VoIP platforms. Cisco IOS supports hop-by-hop and pass-through RSVP. RSVP-to-DSCP (Diffserv Code Point) mapping (RSVP proxy) is supported in 12.1T.

# 4 Differentiated Services Model (DiffServ)

## 4.1 Introduction

The *Differentiated Services* (*DiffServ* for short) architecture was developed as an alternative resource allocation scheme for service providers' networks. By mid-1997 service providers felt that *Integrated Services* were not ready for large-scale deployment, and at the same time the need for an enhanced service model had become more urgent. The Internet community started to look for a simpler and more scalable approach to offer a better than best-effort service. After a great deal of discussion, the IETF formed a new working group to develop a framework and standards for allocating different levels of services in the Internet. The new approach, called *Differentiated Services*, is significantly different from *Integrated Services*. Instead of making per-flow reservations, *Differentiated Services* architecture uses a combination of edge policing, provisioning, and traffic prioritization to achieve service differentiation.

## 4.2 The IP QoS Pendulum



The best-effort and the IntServ models represent two extremes of the resource allocation spectrum: the best-effort model works on a per-packet basis, whereas the IntServ model deals with individual flows. The DiffServ model lies somewhere in between these two extremes: it tries to take one small step further from the best-effort model to offer a “better than best-effort” service.

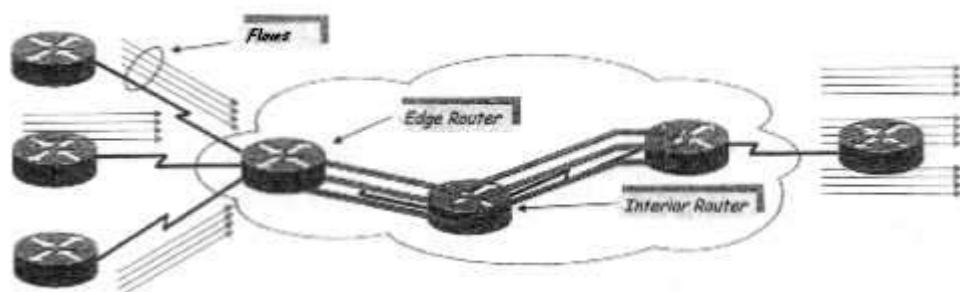
## 4.3 Integrated Services/RSVP Model

As we have already seen, there are some difficulties associated with the IntServ model. One problem is scalability: per-flow resource reservation using RSVP implies the need for a router to process resource reservations and to maintain per-flow state for each flow passing through the router. Per-flow reservation processing at a backbone router can thus incur a potentially significant overhead in large networks. The other problem is flexibility. The IntServ framework provides for the Controlled Load and Guaranteed QoS Service classes. This set of service classes does not allow for more qualitative or relative definitions of service distinctions, e.g. “Service class A will receive preferred forwarding treatment over service class B”. These more qualitative definitions might better fit our intuitive notion of service distinction, i.e. first class versus coach class in air travel, or platinum versus gold versus standard credit cards.

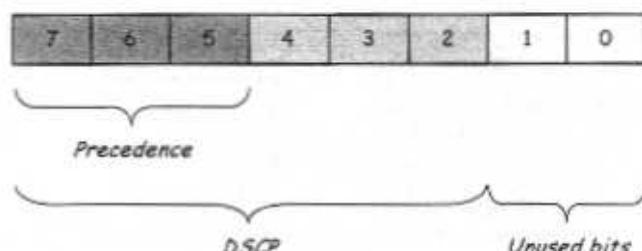
## 4.4 Differentiated Services



With Differentiated Services (DiffServ or DS for short), traffic is partitioned into a small number of groups called *forwarding classes*. Each forwarding class represent a predefined forwarding treatment in terms, for instance, of drop priority and bandwidth allocation. To distinguish the data packets from different customers in DS-capable network devices, the IP packets are modified in a specific field called the *DS field*. The forwarding class is directly encoded into the packet header. The DS field uses the space of the former Type of Service (ToS) octet in the IPv4 IP header and the Traffic Class octet in the IPv6 header.

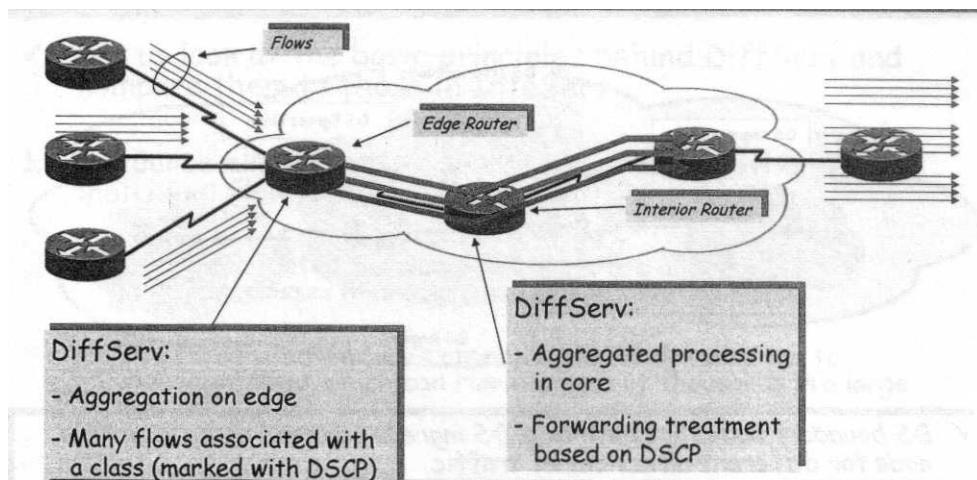


The six-bit *Differentiated Service Code Point* (DSCP) subfield determines the forwarding class and therefore the forwarding treatment that the packet will receive within the network. The other two-bit subfield is currently unused.



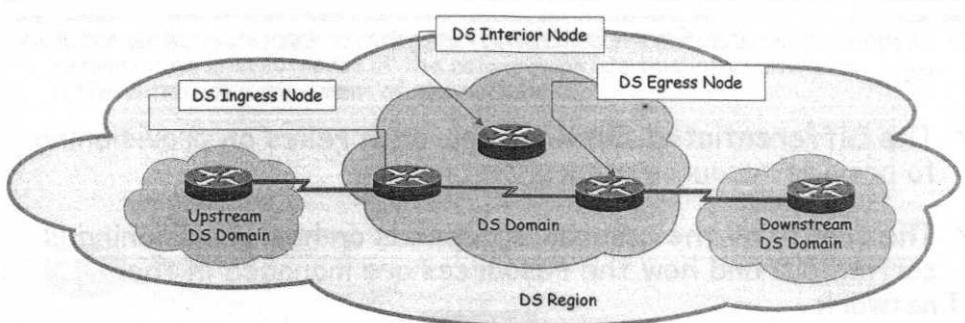
*Former ToS Byte = New DS Field*

After packets are marked with their forwarding classes at the edge of the network, the interior nodes of the network use this information to differentiate the treatment of the packets. The forwarding classes may indicate drop priority or resource priority. For example, when a link is congested, the network will drop packets with the highest drop priority first. Differentiated Services do not require resource reservation setup. The DiffServ approach relies on *provisioning* to provide resource assurance. The quality of assurance depends on how provisioning is carried out and how the resources are managed in the network. Because the dynamic nature of traffic flows, precise provisioning is difficult. Thus it generally is more difficult to provide deterministic guarantees through provisioning rather than reservation.



DiffServ scalability comes from aggregation of traffic on edge and processing of aggregate only in core.

## 4.5 Topological Terminology



A DS domain consists of DS boundary nodes and DS interior nodes. Boundary nodes (or edge nodes) interconnect the DS domain to other DS or non-DS-capable domains, whilst interior nodes (or core nodes) only connect to other DS interior or boundary nodes within the same DS domain. Both DS boundary and interior nodes must be able to apply the appropriate forwarding treatment to packets based on the DSCP. DS boundary nodes act both as a DS ingress node and as a DS egress node for different directions of traffic. Traffic enters a DS domain at an ingress node and leaves at a DS egress node. A Differentiated Services Region (DS region) is a set of one or more contiguous DS domains. DS regions are capable of supporting differentiated services along paths which span the domains within the region.

## 4.6 DiffServ Architecture – RFC 2475

Let us look at the basic principles behind DiffServ and compare them to those in IntServ.

1. *Resource allocation to aggregated traffic.* In Differentiated Services, resources are allocated to classes that represent aggregated traffic. The Integrated Services approach allocates resources to individual flows, which can run into tens of thousands in a large network.
2. *Traffic classifying and marking on the edge and class-based forwarding in the core.* In Differentiated Services, only boundary nodes at the edge of the network classify traffic and mark packets. Once the packets are marked, the interior nodes use the forwarding classes encoded in the packet header (i.e. the DSCP value) to determine the treatment of the packets. Integrated Services, in contrast, requires all nodes to perform packet classification to identify packets from reserved flows and schedule them with per-flow queuing.
3. *Define forwarding behaviors not end-to-end services.* Each forwarding class represents a forwarding treatment rather than a service. The service, however, can be constructed by combining forwarding classes and admission control (see SLA – Service Level Agreement). The IntServ model takes the opposite approach: it defines services; for example, the Guaranteed QoS Service and the Controlled Load Service.
4. *Guarantees by provisioning.* DiffServ provides resource assurance through provisioning rather than per-flow reservation. This means that applications using DiffServ do not need to setup QoS reservations for specific flows. By allocating resources to forwarding classes and controlling the amount of traffic for these classes, DiffServ creates different levels of

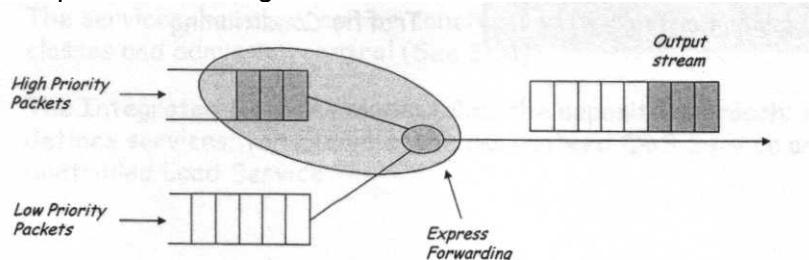
services but not absolute bandwidth guarantees or delay bounds for individual flows.

## 4.7 IntServ versus DiffServ

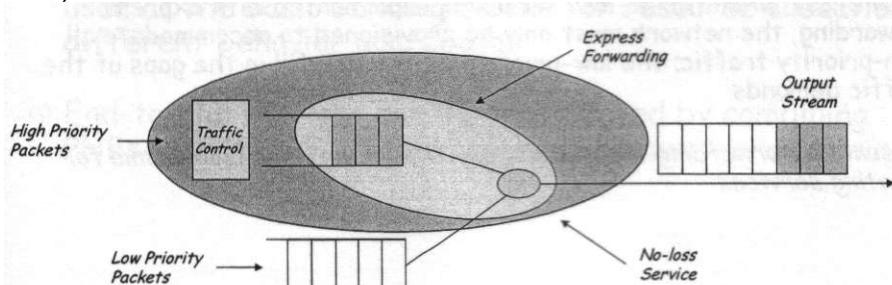
IntServ	DiffServ
<ul style="list-style-type: none"> <li>• Flow based</li> <li>• Per flow state</li> <li>• RSVP signalling</li> <li>• Classification at every hop</li> <li>• Main issue is scalability</li> <li>• Admission control</li> </ul>	<ul style="list-style-type: none"> <li>• Flow-aggregate based</li> <li>• Per aggregate state</li> <li>• No signalling (provisioning)</li> <li>• Classification at the edge</li> <li>• Scalability is not an issue</li> <li>• Traffic conditioning</li> </ul>

## 4.8 Service and Forwarding Treatment

Service and *forwarding treatment* (or *treatment* for short) are two important concepts in the DiffServ architecture. The difference between them is quite subtle, and people often confuse them. In the DiffServ context, forwarding treatment refers to the externally observable behavior of a scheduling algorithm that is implemented in a node. In contrast, service is defined by the overall performance that a customer's traffic receives. Services and forwarding treatments are not the same but are closely related, let's illustrate the difference with an example. Priority queuing is a scheduling algorithm that allows high-priority packets to get serviced (i.e. transmitted) before packets with lower priority. Let us call this forwarding treatment for the packets with higher priority "express forwarding". Express forwarding is a treatment, not a service.



This forwarding treatment, however, can be used to construct services. For example, suppose that we define a service called "no-loss service", which guarantees no packet losses for the customers. This service can be implemented with the express forwarding treatment by assigning high priority to the packets of the customers using the service. However, the express forwarding alone is not sufficient to guarantee the no-loss service. If too many high-priority packets arrive at a link, they will also be dropped. The no-loss service must be constructed by limiting (see traffic conditioning later on) the amount of traffic in the no-loss service.



A service and a forwarding treatment do not necessarily have a one-to-one mapping. Typically there are multiple ways of implementing a service with different forwarding treatments. For example, it is possible, at least in theory, to implement the no-loss service with First Come First Serve queuing. To do that, all traffic into the network must be limited to a very small percentage of the link bandwidth so that the network is always lightly loaded. This is, of course, much more difficult to implement than the approach we described using express forwarding. Thus services constructed with different forwarding treatments may result in different provisioning. Traffic conditioning, and efficiency in terms of bandwidth utilization. For the no-loss service implemented with FCFS, the network must be provisioned so that in the worst case resources are sufficient to meet all traffic demands. In contrast, when the no-loss service is implemented with express forwarding, the network must only be provisioned to accommodate all high-priority traffic; the low-priority traffic can fill in the gaps of the traffic demands. To sum up, forwarding treatments are the

underlying mechanisms for creating services.

## 4.9 Per-Hop Behavior

Per-Hop Behavior (PHB) is defined as

*"a description of the externally observable forwarding behavior of a Differentiated Services node applied to a particular behavior aggregate".*

A Behavior Aggregate (BA) is a collection of packets with the same DSCP crossing a link in a particular direction. Different BAs will then receive different forwarding treatments within the core network. RFC 2475 uses the term “behavior aggregate” rather than “class of traffic”. PHBs are implemented in DS nodes by means of some buffer management and packet scheduling algorithms. PHBs may describe the forwarding treatments in either relative or absolute terms. In DiffServ, PHBs are used as the basic building blocks for resource allocation to different behavior aggregates. End-to-end services can be constructed by combining PHBs with traffic conditioning and network provisioning. While a PHB defines differences in behavior among behavior aggregates, it does not mandate any particular scheduling algorithm for achieving these behaviors. As long as the externally observable performance criteria are met, any scheduling algorithm and any buffer/bandwidth allocation policy can be used. For example, a PHB would not require that a particular packet scheduling algorithm, for example, a Priority Scheduling versus a WFQ versus a FIFO, be used to achieve a particular behavior.

From RFC 2475:

*"This architecture achieves scalability by implementing (complex) classification and conditioning functions only at network boundary nodes, and by applying per-hop behaviors to aggregates of traffic which have been appropriately marked using the DS field in the IPv4 or IPv6 headers. Per-application flow or per-customer forwarding state need not be maintained within the core of the network."*

## 4.10 Per-Hop Aggregate Examples

1. A simple PHB is one that guarantees that a given Behavior Aggregate receives at least X% of the outgoing link bandwidth.
2. Another PHB might specify that one Behavior Aggregate always receive strict priority over another Behavior Aggregate – that is, if a high priority packet and a low-priority packet are present in a router’s queue at the same time, the high-priority packet will always leave first.
3. A slightly more complex PHB would guarantee a Behavior Aggregate a minimal bandwidth allocation of X% of a link, with proportional fair sharing of any excess link capacity.

## 4.11 Per-Hop Behavior Group

Some PHBs are defined with closely related behaviors and are referred to as *PHB groups*. For example, a set of PHBs that specify the same basic queuing and scheduling behavior but indicate different drop probabilities to the queue manager. A single PHB may be viewed as a special case of a PHB group.

## 4.12 Services

A service describes the overall treatment of a customer’s traffic within a DS domain or end-to-end. Services are what is visible to customers, whereas PHBs are hidden inside the network elements (i.e. routers). Services can be defined in either quantitative or qualitative terms.

- Quantitative services specify the parameters in absolute terms; for example, a minimum bandwidth of 0.30Mbps.
- Qualitative services typically use relative terms, such as lower delay or higher-loss probability.

Creating a service requires that many components work together: mapping of traffic to specific PHBs, traffic conditioning at the boundary nodes, network provisioning, and PHB-based forwarding in the interior of the network. Thus, a service requires an agreement to be negotiated between the customer (which may be a user organization or another DS domain) and the Internet service provider (ISP). The service provider must assure that the traffic of a customer gets the contracted QoS. Therefore, the service provider network administration must set up the appropriate service policies and measure the network performance to guarantee the agreed traffic performance. In the

DiffServ architecture there two levels of agreements:

- *Service Level Agreement (SLA)*: a contract between a customer and a service provider that specifies the service a customer requires (i.e. a SLA is the formalization of the QoS in a contract between customer and service provider);
- *Traffic Conditioning Agreement (TCA)*: defines the rules used to realize the service, such as metering, marking and discard.

SLAs should be:

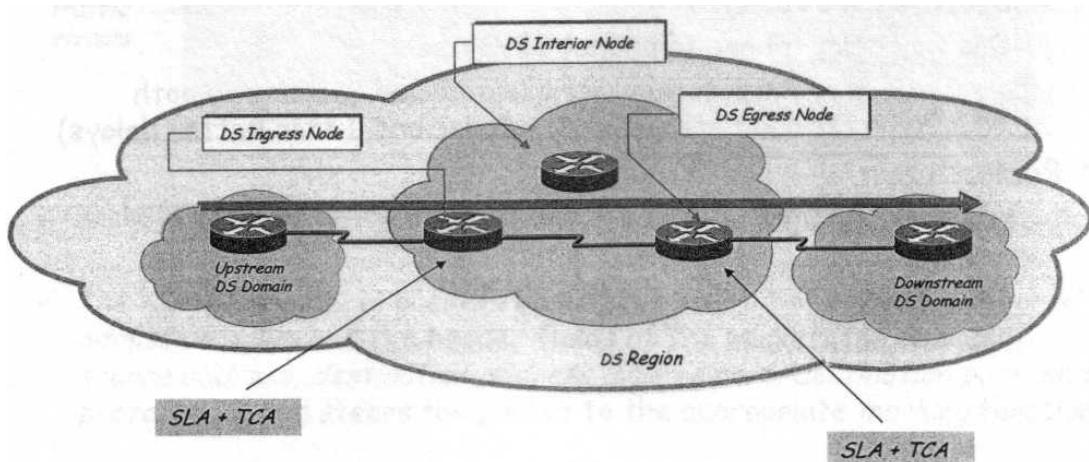
- simple to understand,
- standardized,
- simple to measure & report,
- the same for ALL customers (no customized SLAs per customer).

Network SLA metrics are:

- General metrics
  - Availability
  - Mean time to restore (MTTR)
- QoS metrics
  - Network delay
  - Packet loss
  - Network delay variation (jitter) (example for voice services)

VoIP-SLA	Total Delay	Propagation Delay	CODEC (G.729)	Access Budget	Core Budget
Excellent	100	25	40	25	10
Good	150	25	40	25	60
Moderate	250	25	40	25	160

The TCA contains information on how metering, marking, discarding and shaping of packets must be done in the traffic conditioner to fulfill the SLA. Specifically, the TCS information must be available in all boundary components of a DS network to guarantee that packets passing through different DS domains receives the same service in each domain.



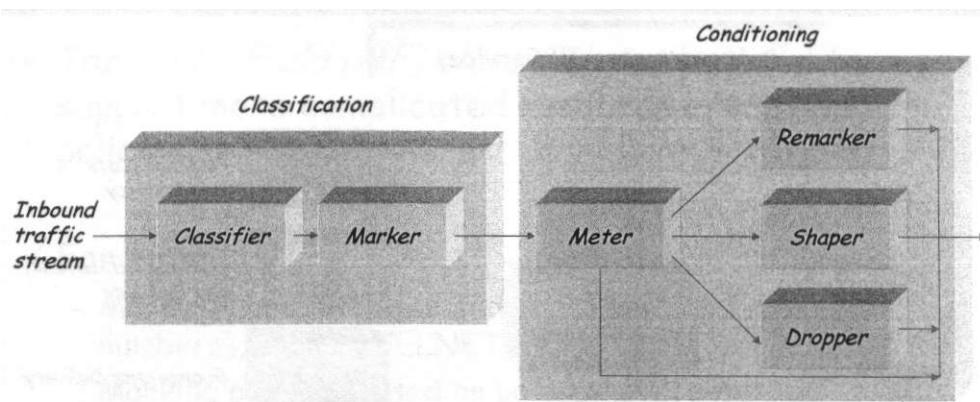
What SLAs do we have today? “Internet Services”:

- Backbone availability: typically 99.9%, starting to see more and more 100%;
- Packet loss: typically lower than 1% loss, starting to see lower than 0.1% loss;
- Round trip delay.

What are the requirements for quality voice?

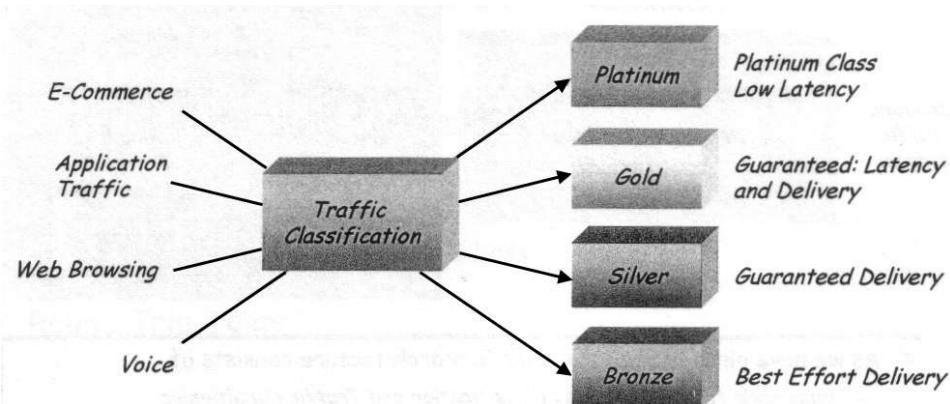
- Delay/jitter: specified in ITU G.114, one way delay (phone to phone) < 150msec (sum of propagation, serialization and queuing on both access and core network + codec and jitter buffer delays)
- Packet loss: lower than 0.25%.

## 4.13 Architectural Model



As we have already seen the DiffServ architecture consists of edge node functions (packet classification and traffic conditioning) and core node functions (forwarding).

#### 4.14 Classifier



Packets arriving to the edge router are first classified. The classifier selects packets based on a match between a configured admission policy and the header fields of the packet (for example, source address, destination address, source port, destination port, and protocol ID) and steers the packet to the appropriate marking function.

The DS model specifies two types of packet classifiers:

- Behavior Aggregate (BA) classifier, which classifies packets based on their DSCP value only. BA classifiers are generally used when the DSCP has been set (also referred to as marked) before the packet reaches the classifiers.
- Multi-Field (MF) classifier, which can classify packets based on the DSCP field as well as on any other IP header field, for example, the source address, destination address, source port, destination port, and protocol ID.

The multi-field (MF) classifier can be used to support more complicated resource allocation policies from customers. For example:

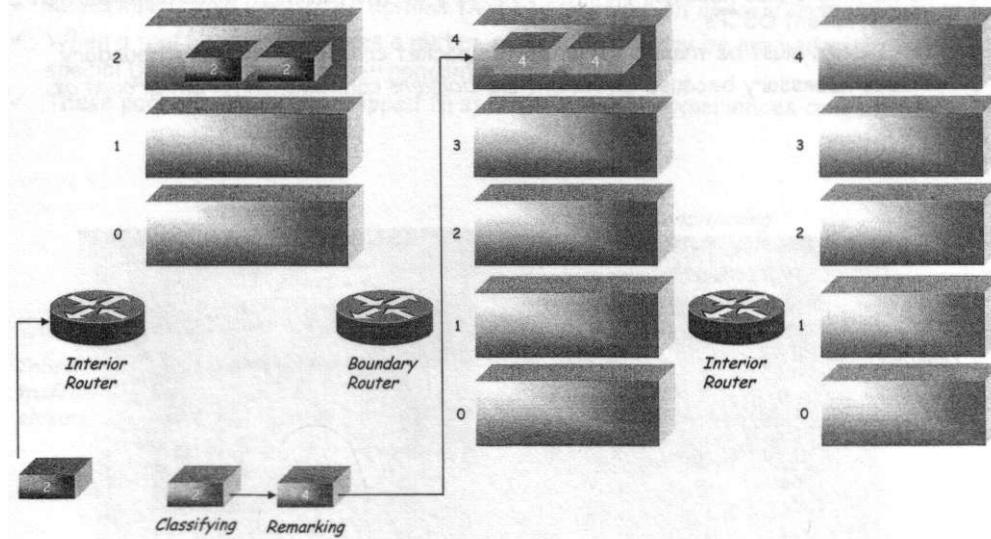
- marking packets based on the application types (port numbers), such as TELNET or FTP;
- marking packets based on source and destination addresses or network prefixes, such as CEO's traffic or some mission-critical servers;
- marking packets based on the 5-tuple that determines an application flow, such as a video stream.

A question not addressed by DiffServ is how the classifier obtains the rules for such classification. This could be manually – that is, the network administrator could load a table of source addresses that are to be marked in a given way into the edge routers – or under the control of some yet-to-be-specified signalling protocol.

#### 4.15 Marking & Remarking

Markers set the DS field of a packet to a particular DSCP, adding the marked packet to the Forwarding Class. Markers may act on unmarked packets or remark previously marked packets. Remarking is also one of the actions that can be taken on non-conformant packets. When traffic stream passes a meter, some packets may be marked with a special DSCP to indicate their non-conformance. These packets should be dropped first if the network experiences congestion. Since

packets may pass many different domains, packets that have been previously marked may be remarked. When a packet stream violates traffic profiles at any administrative boundary, packets may be remarked to a different DSCP. Remark is also necessary at the boundary of two administrative domains that use different and incompatible DSCPs. The DSCPs must be translated when the packet crosses the domain boundary. This is necessary because different DS domains can have different groups of PHBs. For example, in the first domain a packet traverses, all routers have three queues with different queue priorities (0-2) where packets with a PHB value of 2 are routed with the highest priority. But in the next domain the packet travels through, all routers have five different queues and all packets with the PHB value of five are routed with the highest priority.



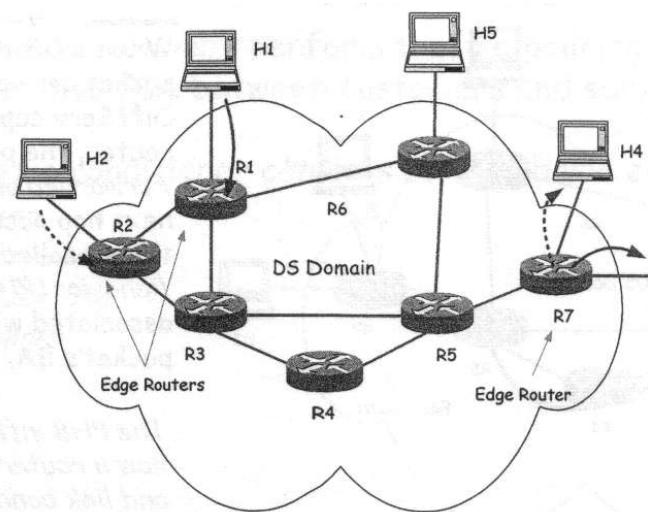
The packet that was forwarded in the first domain with high priority has only medium priority in the second domain and this may violate the Service Level Agreement contract between customer and service provider. Therefore, the traffic conditioner in the boundary router that connects the two domains must assure that the PHB value is remarked from two to five if the packet travels from the first to the second domain. It is also possible that several DS domains within a DS region may adopt a common service provisioning policy and may support a common set of PHB groups and code point mappings, thus eliminating the need for traffic conditioning between those DS domains.

Question: *who marks the packets?* Here are two answers to this question:

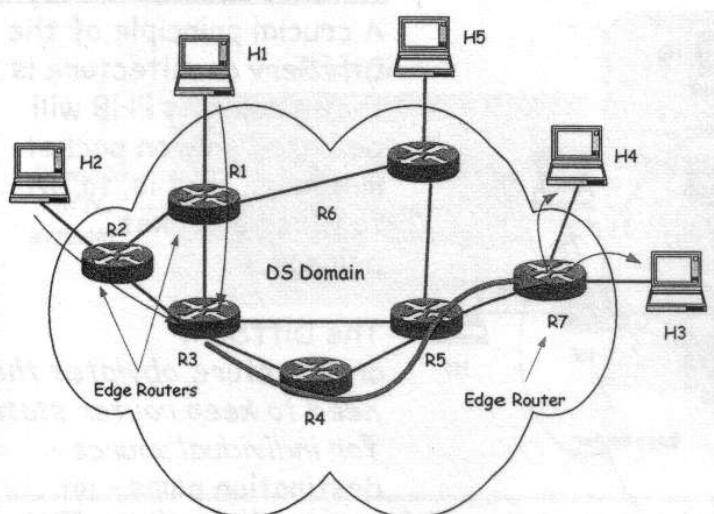
- *First answer:* the host marks the packets with their desired DSCP, indicating the service quality desired.
- *Second answer:* no assumption is made about the capability of the host to mark packets – the DS ingress node rewrites the DS packet mark unconditionally, to correspond to the traffic classification and the associated profile meter directive dictated by the network's admission policy.

#### 4.16 Example of Marking

All packets coming from a certain set of source IP addresses (for example, those IP addresses that have paid for an expansive priority service within their ISP or DS domain) could be marked on entry to the DS domain, and then receive a specific forwarding service (for example, a higher-priority forwarding) at all subsequent DiffServ-capable routers.



Example – packets being sent from H1 to H3 might be marked at R1, while packets being sent from H2 to H4 might be marked at R2.



If packets being sent from H1 to H3 receive the same marking as packets being sent from H2 to H4, then the network (core) routers treat these packets as an aggregate, without distinguishing whether the packets originated at H1 or H2. For example, R3 would not distinguish between packets from H1 and H2 when forwarding these packets to R4. When a DS-marked packet arrives at a DiffServ capable router, the packet is forwarded onto its next hop according to so-called per-hop behavior associated with that packet's BA. The PHB influences how a router's buffers and link bandwidth are shared among the competing BAs. A crucial principle of the DiffServ architecture is that a router's PHB will be based only on packet markings, that is, to BA to which a packet belongs. The DiffServ architecture obviates the need to keep router state for individual source – destination pairs – an important consideration in meeting the scalability requirement.

#### 4.17 Traffic Conditioners

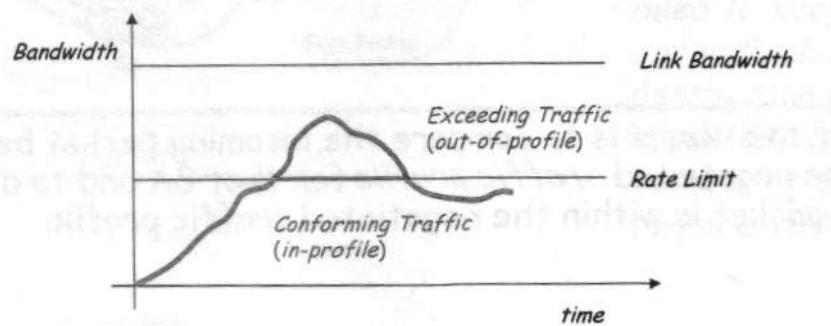
Traffic conditioners perform traffic-policing functions to enforce the TCS between customers and service providers. A traffic conditioner consists of four elements: meter, (re)marker, shaper and dropper.

#### 4.18 Meter

The role of the meter is to compare the incoming packet belonging to a BA with the negotiated traffic profile for that BA and to determine whether a packet is within the negotiated traffic profile. Traffic profiles can be described as specific properties of a traffic stream that allow packets to be classified as either in-profile or out-profile by the meter. For example, a specific profile at an ingress node may be based on a token-bucket implementation and may classify packets with a specific DSCP against a specific rate and burst:

$$\text{DSCP} = X, \text{use token bucket } (r, b)$$

This simple example of a profile indicates that all packets received with a DSCP of X be measured against a token-bucket meter with a rate of  $r$  and a burst-size of  $b$ . Packets that exceed these rate and burst parameters would be considered out-of-profile, and packets that conform to them would be considered in-profile. Since traffic profiles are typically described in terms of token bucket parameters, most meters are implemented as token buckets.

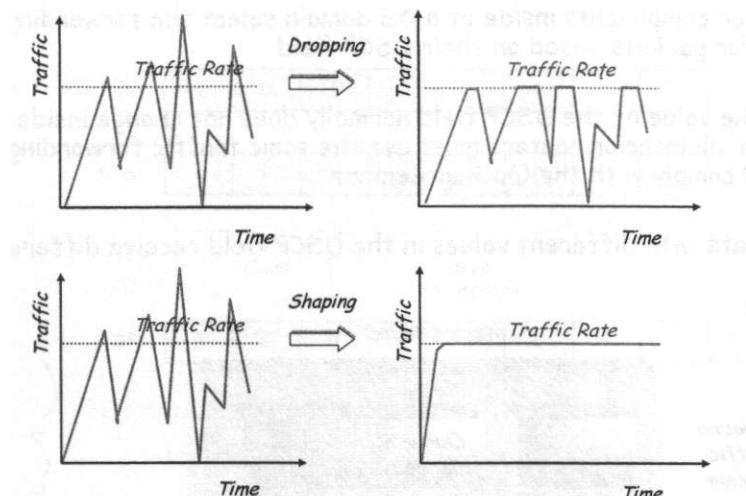


Additional rules need to be added to the example to indicate the action to take regarding the disposition of out-of-profile packets. The actions that a traffic conditioner may include shaping, marking and dropping, e.g., out-of-profile packets can be re-marked to a lower relative priority.

#### 4.19 Shaper

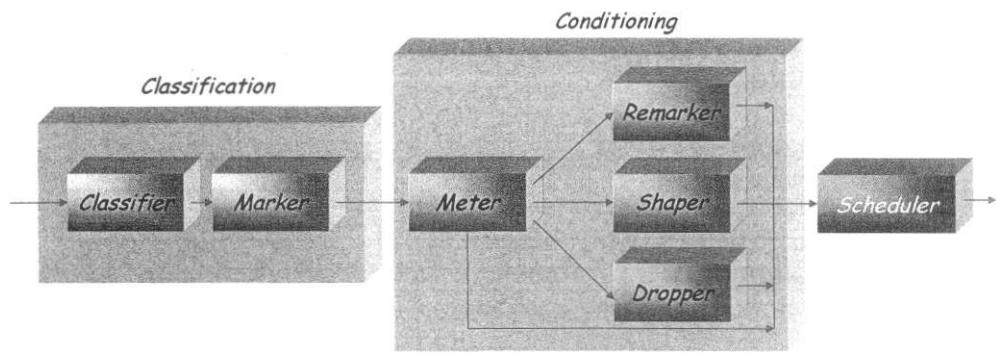
Shapers delay the non-conformant packets in order to bring the stream into compliance with the agreed-on traffic profile. The difference between a shaper and a marker is that a marker simply marks a packet and lets it into the network whereas a shaper prevents the packet from entering the network until the stream conforms to the traffic profile. Shaping may also needed at a boundary node to a different domain. Traffic profiles tend to change when packets traverse the network. The egress node may need to shape the outgoing traffic stream so that it conforms to the appropriate traffic profile of the next domain.

#### 4.20 Dropper



Dropping is another action that may be applied to out-of-profile packets. Compared with shaping, which must buffer packets temporarily, dropping is much easier to implement. Since a shaper has a finite-size buffer, it also drops packets when the buffer overflows.

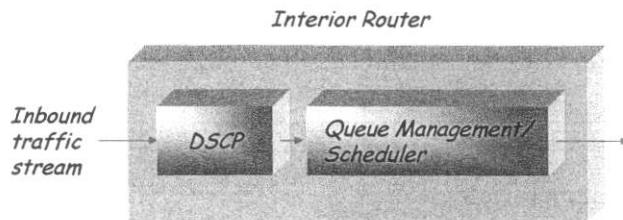
#### 4.21 Scheduler



Possible scheduling algorithms: WFQ, DRR, etc.

## 4.22 DS Interior Components

The interior components inside of a DS domain select the forwarding behavior for packets based on their DSCP field. Because the value of the DSCP field normally does not change inside of a DS domain, all interior routers must use the same traffic forwarding policies to comply with the QoS agreement. Data packets with different values in the DSCP field receive different QoSs.

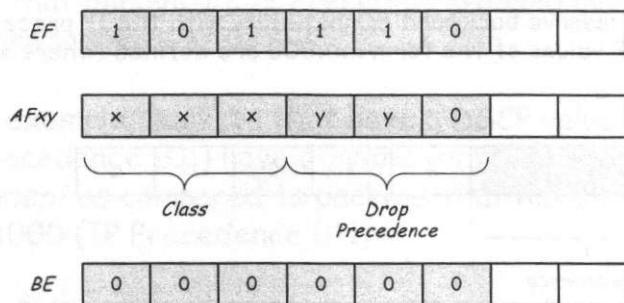


## 4.23 The Standard PHBs

To date, four standard PHB implementations of DiffServ are available:

- Default PHB,
- Class-Selector PHB,
- Expedited Forwarding (EF) PHB (RFC 3246) and
- Assured Forwarding (AF) PHB (RFC 2597).

PHB recommended codepoints:

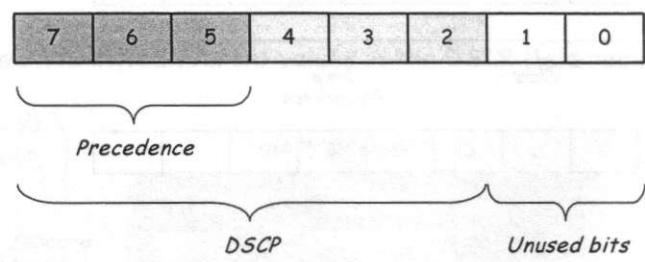


## 4.24 Default PHB

The Default PHB results in a standard best-effort delivery of IP packets. Packets marked with DSCP value of 000000 get the traditional best-effort service from a DS-compliant node. Furthermore, if a packet arrives at a DS-compliant node and its DSCP value is not mapped to any of the other PHBs, it is mapped to the default PHB.

## 4.25 Class-Selector PHB

Many current implementations of IP QoS use IP Precedence due to its simplicity and ease of implementation. In order to preserve backward compatibility with the IP Precedence scheme, DSCP values of the form xxx00 are defined (where x equals 0 or 1).

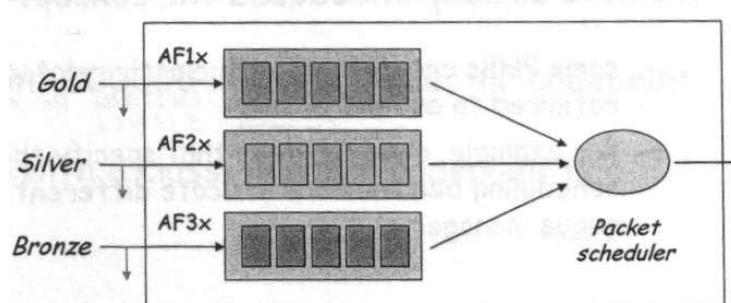


*Former ToS Byte = New DS Field*

Such codepoints are called Class-Selector codepoints. The default codepoint 0000000 is a Class-Selector codepoint. The PHB associated with a Class-Selector codepoint is a Class-Selector PHB. These PHBs retain the same forwarding behavior as nodes that implement IP Precedence-based classification and forwarding. As an example, packets that have a DSCP value of 101000 (IP Precedence 101) have a preferential forwarding treatment as compared to packets that have a DSCP value of 011000 (IP Precedence 011). These PHBs ensure that DS-compliant nodes can coexist with IP Precedence-aware nodes.

## 4.26 Assured Forwarding PHB group

Defined in RFC 2597, *Assured Forwarding* is actually a PHB group for edge-to-edge services specified in terms of relative bandwidth availability and multi-tiered packet drop characteristics. Whereas Expedited Forwarding supports services with “hard” bandwidth and jitter characteristics, the AF group allows more flexible and dynamic sharing of network resources – supporting the “soft” bandwidth and loss guarantees appropriate for bursty traffic.

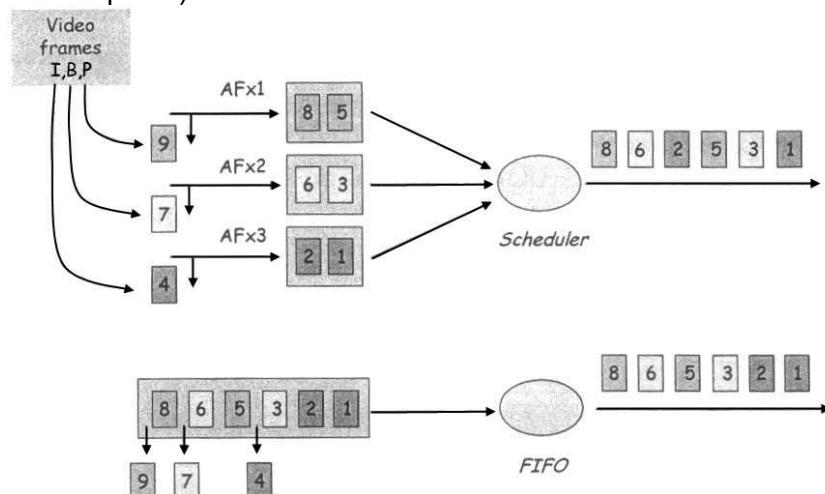


Two distinct classification contexts are encoded within the DSCP – a packet’s service class and its drop precedence. The service class provides context with which to select an appropriate queue (and, hence, an appropriate bandwidth share from the scheduler). The drop precedence provides contexts to weight RED-like behavior expected from the queue manager – making it more or less aggressive depending on whether a packet’s drop precedence is high or low. The basic idea behind AF came from the RIO (RED with In and Out) scheme, one of the early proposals that influenced the DiffServ work. In the RIO framework a service profile specifies the expected capacity for the user. The boundary nodes monitor the traffic flows and tag the packets as being in or out their profiles. During congestion the packets tagged as out will be dropped first. In essence, the in/out bit indicates drop priorities. The service providers should provision their networks to meet the expected capacities for all in-profile packets and allow out-of-profile packets only when excessive bandwidth is available. The AF standard extended the basic in-or-out marking in RIO into a structure of four forwarding classes and, within each forwarding class, three drop precedences. Each forwarding class is allocated a minimum amount of buffers and bandwidth. Customers can subscribe to the services built with AF forwarding classes and their packets will be marked with the appropriate AF DSCPs. When backlogged packets from an AF Class exceed a specified threshold, packets with the highest drop precedence are dropped first, then those with lower drop precedence. Drop priorities in AF are specific to the class: comparing two drop precedences in two different AF classes may not always be meaningful. For example, when a DS node starts to drop the packets with the highest drop precedence in one AF class, the packets in other AF classes may not experience any packet dropping at all. By varying the amount of resources allocated to each class, an ISP can provide different levels of performance to the different AF classes.

Drop precedence	Class #1	Class #2	Class #3	Class #4
Low drop precedence	AF11 001 010	AF21 010 010	AF31 011 010	AF41 100 010

Medium drop precedence	AF12 001 100	AF22 010 100	AF32 011 100	AF42 100 100
High drop precedence	AF13 001 110	AF23 010 110	AF33 011 110	AF43 100 110

Packets in one AF class must be forwarded independently from packets in another AF class, i.e., a DS node must not aggregate two or more AF classes together. A DS node must allocate a configurable, minimum amount of forwarding resources (buffer space and bandwidth) to each implemented AF class. An AF class may also be configurable to receive more forwarding resources than the minimum when excess resources are available either from other AF classes or from other PHBs. The only difference between, say, AF11, AF12, and AF13 is the drop precedence. A DS node must not reorder AF packets of the same microflow when they belong to the same AF class regardless of their drop precedence. Typically, this means that all packets of the same AF class (i.e. packets marked AFx1, AFx2 and AFx3, where x is a class) go into a common queue (the AFx queue).



There are no quantifiable timing requirements (delay or delay variation) associated with the forwarding of AF packets. The AF PHB could be used as a building block to provide different levels of service to the end systems, for example, gold, silver, and bronze classes of service. But what would be required to do so? If gold service is indeed going to be “better” (and presumably more expensive) than silver service, then the ISP must ensure that gold packets receive lower delay and/or loss than silver packets. Recall, however, that a minimum amount of bandwidth and buffering are to be allocated to each class. Problem: what would happen if gold service was allocated x percent of a link’s bandwidth, but the traffic intensity of gold packets was 100 times higher than that of silver packets? Very likely silver packets would receive better performance than the gold packets! Clearly, when creating a service out of PHB, more than just the PHB itself will come into play. In this example, the dimensioning of resources – determining to what extent resources will be allocated to each class of service – must be done hand-in-hand with knowledge about the traffic demands of the various classes of traffic.

## 4.27 AF PHB Implementation Guideline

The AF PHB group can be implemented as a bandwidth partition between classes and drop precedences within a class. The bandwidth partition is specified in terms of minimum bandwidth, and so many different implementations are available. One common approach is to use variants of WFQ and assign weights according to minimum bandwidth requirements. Such implementations also distribute excessive bandwidth proportional to backlogged classes. Let us look at an example. Suppose that we have four AF classes with a minimum bandwidth of 2, 4, 8, and 16Mbits/sec, respectively. In order to meet the bandwidth guarantees, each link needs at least 30Mbits/sec capacity available to the AF traffic. With WFQ-like implementations, the four AF classes can be mapped to four queues with weights of 1, 2, 4 and 8, respectively. To achieve these results take into consideration that:

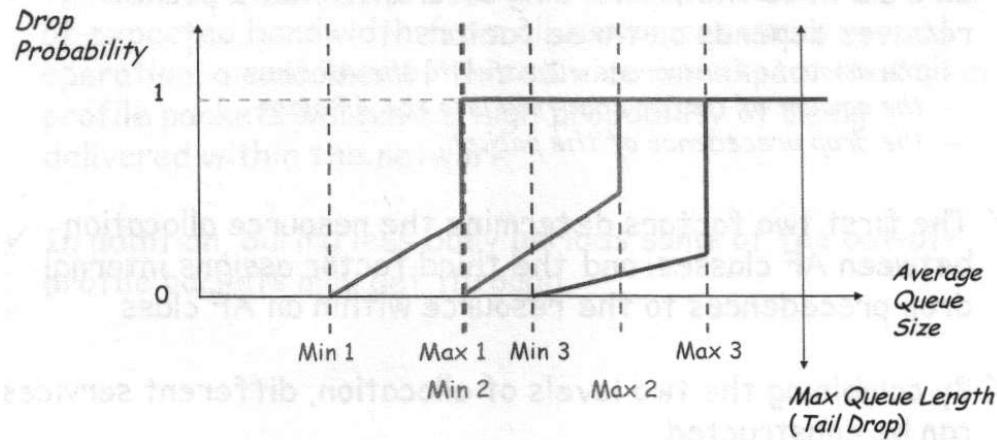
$$g_i = \frac{\Phi_i}{\sum_{j=1}^N \Phi_j} r = \frac{\Phi_i}{15} 30$$

The AF specification allows many dropping implementations:

- First idea: Random Early Discard (RED).

- Permanent control of the average level of packet in queue.
- Avoid congestion by packet dropping before queue is filled-up.
- Uses a random dropping.
- Second idea: Weighted RED (WRED).
  - Differentiated Services.
  - Different threshold per IP Precedence/DSCP.

Example: WRED implementation of three importance levels:



WRED uses different RED profile for each weight. Each profile is identified by:

- Minimum threshold,
- Maximum threshold,
- Maximum drop probability.

Weight can depend upon:

- IP Precedence (8 profiles)
- DSCP (64 profiles)

WRED drops “less important” packets more aggressively than “more important” packets.

## 4.28 Example Services

In a DS node the forwarding assurance that a packet receives depends on three factors:

- the amount of resources allocated to the AF class,
- the amount of traffic admitted into the AF class,
- the drop precedence of the packet.

The first two factors determine the resource allocation between AF classes, and the third factor assigns internal drop precedences to the resource within an AF class. By combining the two levels of allocation, different services can be constructed. The AF PHB group could be used to provide simple services based on drop precedences, such as the expected bandwidth service, as outlined in RIO. A customer for the expected bandwidth service is assigned a traffic profile. The traffic profile specifies the amount of bandwidth the service provider is prepared to accept base on the subscription fee. At the boundary nodes of the network, a customer's traffic stream is measured against its traffic profile and marked as in profile and out of profile. When the network is provisioned to accomodate all traffic of expected bandwidth from all customers under normal operation, a customer of this service can expect that all in-profile packets will have a high probability of being delivered within the network. In addition, during less busy periods some of the out-of-profile packets may get through. Another possible use of AF is to construct several services with different provisioning levels. For example, we can create a service with three service classes: bronze, silver and gold, with different subscription fees. The three service classes can be mapped to three AF forwarding classes with different bandwidth allocation. The gold service class has the best provisioning in terms of the ratio between the amount of traffic allowed to this class and the amount of bandwidth reserved for it. Thus under normal operation, the load for the gold service class is always lower than that for the silver and bronze services. Packets within each class may be further separated by giving them different drop procedures.

## 4.29 Expedited Forwarding PHB

The Expedited Forwarding (EF) PHB is intended to provide a building block for *low delay, low*

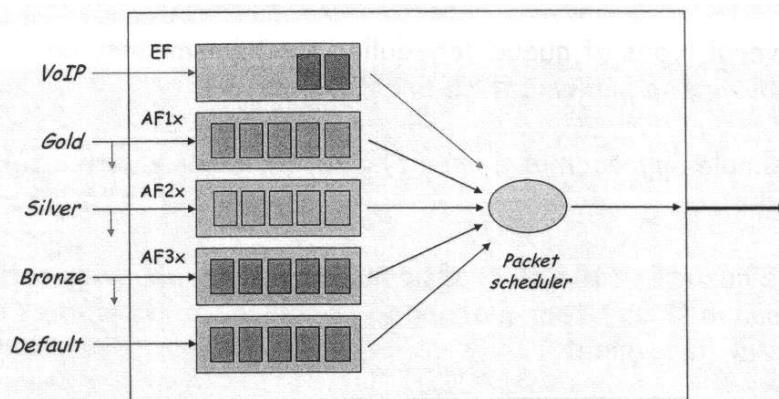
*jitter* and *low loss* services by ensuring that the departure rate of the aggregate's packet from a DS node must equal or exceed a configurable rate.

### DSCP value: <101110>

(Note: “jitter” is defined as the variation between maximum and minimum delay.)

Such a service appears to the endpoints like a point-to-point connection or a Virtual Leased Line. This service has also been described as Premium Service. We have already seen that the dominant causes of delay in packets networks are fixed propagation delays (e.g. those arising from speed-of-light delays) on wide area links and queuing delays in switches and routers. Since propagation delays depends upon the topology, delay and jitter are minimized when queuing delays are minimized. Thus, the intent of the EF PHB is to provide a PHB in which the related packets usually encounter short or empty queues. Furthermore, if queues remain short relative to the buffer space available, packet loss is also kept to a minimum. To ensure that queues encountered by EF packets are usually short, it is necessary to ensure that the service rate of EF packets on a given output interface exceeds their arrival rate at that interface over long and short time intervals, independent of the load of other (non-EF) traffic. This specification defines a PHB in which EF packets are guaranteed to receive service at or above a configured rate and provides a means to quantify:

- the accuracy with which service rate is delivered over any time interval;
- the maximum delay and jitter that a packet may experience under bounded operating conditions.

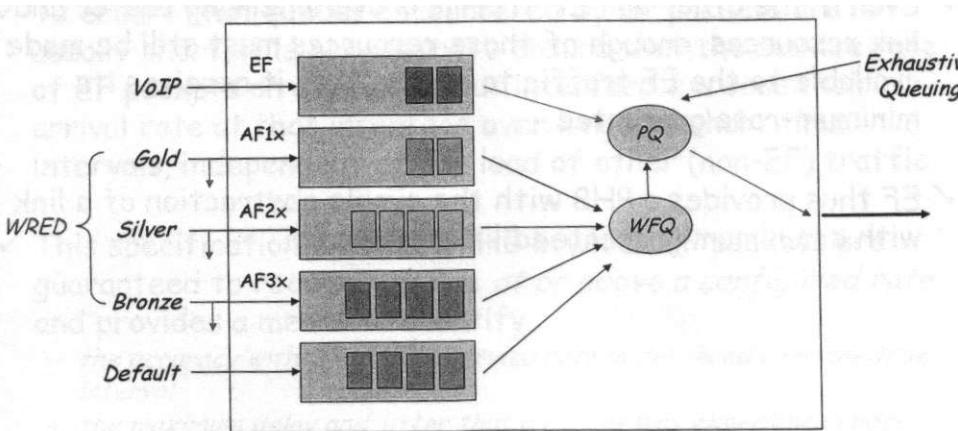


Intuitively, the definition of EF is simple: the rate at which EF traffic is served at a given output interface should be at least the configured rate  $R$ , over a suitably defined interval, independent of the offered load of non-EF traffic to that interface. Even if the other non-EF traffic is overwhelming router and link resources, enough of those resources must still be made available to EF traffic to ensure that it receives its minimum guaranteed link bandwidth.

### 4.30 EF PHB Implementation Guideline

Several types of queue scheduling mechanism may be employed to implement EF PHB. A simple approach is a priority-queuing scheme with a token bucket. The queue for the EF traffic must be the highest-priority queue in the system in order to preserve the properties of the EF treatment. The token bucket is there to limit the total amount of EF traffic so that other traffic will not be starved by bursts of EF traffic.

### 4.31 Putting All Together



### 4.32 Which Applications Have Which QoS Requirements?

	Throughput	Delay	Loss	Jitter
Interactive (e.g. Telnet)	Low	Low	Low	Not important
Batch (e.g., FTP)	High	Not important	Low	Not important
Fragile (e.g., SNA)	Low	Low	None	Not important
Voice	Low	Low and predictable	Low	Low
Video	High	Low and predictable	Low	Low

Enterprise networks are typically focused on providing QoS to applications.

### 4.33 Which Services Can Be Implemented in a Network?

	Throughput	Delay	Loss	Jitter
Gold	Guaranteed	Low	Low	Low
Silver	Guaranteed	No guarantee	No guarantee	No guarantee
Bronze	Guaranteed limited	No guarantee	No guarantee	No guarantee
Best effort	No guarantee	No guarantee	No guarantee	No guarantee

Service provider networks typically offer services based on source and destination addresses.

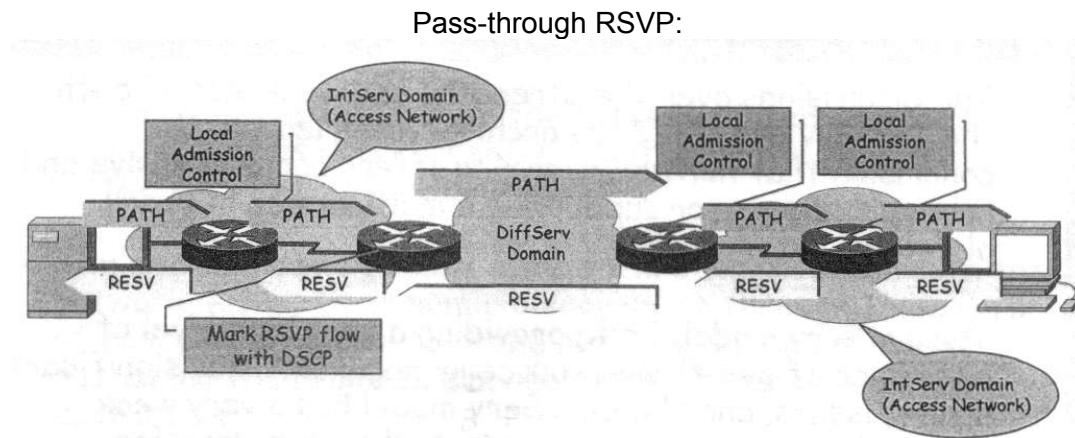
### 4.34 The Trouble with DiffServ

As currently formulated, DiffServ is strong on simplicity and weak on guarantees. The missing element within the DiffServ architecture as it is currently specified is the reproduction of the end-to-end service model. The end application needs some form of negotiation with the network, either by dynamic discovery or through some form of policy negotiation, that the network will be capable of sustaining the required service profile demanded by the application. Invariably this will not be as strict a negotiation as required by the IntServ architecture, but nevertheless it makes little sense for the application to proceed with a premium service profile if the network is incapable of meeting the associated service demands. Clearly, much remains to be done to lift DiffServ out from a carefully managed localized service platform for multi-service environments within large service networks. This does not imply that DiffServ is so immature that it is not ready for use in today's service networks. The elements of admission classification, profilers, packet marking, and various queue management systems that can support resource allocation are a reality in network equipment today.

### 4.35 Combining IntServ and DiffServ

The discussion over the strengths and weaknesses of both IntServ and DiffServ architectures lead to the conclusion that neither approach offers comprehensive and robust solutions for supporting a multi-service network platform. The IntServ model, while providing a very high level of assurance of per-flow resource management, has significant scaling issue, and DiffServ model has very weak approach to resource management, though possessing significant scaling properties. The realization that neither model is sufficient has resulted in a proposal to combine the two models, using IntServ control mechanism at the edge of the network and DiffServ within the network core. The proposal has been further motivated by the observation that some form of end-to-end QoS negotiation is required to support the needs of certain applications, such as video-on-demand and various time- or quality-critical applications. The proposed model is conceived as a

collection of IntServ-capable networks as the edge feeder networks and a DiffServ-capable network as the common service core system.



Part of the network may not support RSVP. Mark RSVP flows with a class of service marker (e.g., IP Precedence or DSCP). Make sure that the core provides guarantees to the RSVP class. It is proposed that applications use RSVP to request that their flows be admitted into the network. If a request is accepted, it would imply that there is a resource reservation within the IntServ-capable components of the network, and that the service requirements have been mapped into a compatible aggregate service class within the DiffServ-capable network. The DiffServ core must be capable of carrying the RSVP messages across the DiffServ network, so that further resource reservation is possible within the IntServ network upon egress from the DiffServ environment. The approach is that the DiffServ network will use Multi-Field (MF) admission classifiers, where the MF classification is based on the IntServ flow specification. The service specification of the IntServ resource reservation is mapped to an aggregate DiffServ behavior aggregate, and the MF admission filter will mark all such packets into the associated aggregate service class. It is also possible for the host to condition the traffic in advance of transmission into the network, and to mark the packets within the flow with the associated commonly defined DSCP values, so that the admission filter can operate via a behavior aggregate admission classifier. The end-to-end QoS model requires that any admission failure within the IntServ network be communicated to the IntServ network, and from there to the end application via RSVP. This allows the application to take corrective action to avoid failure of the service itself. If the service agreement between the DiffServ network is statically provisioned, then this static information can be loaded into the IntServ boundary systems, and IntServ can manage the allocation of the available DiffServ behavior aggregate resources. If the service agreement is dynamically variable, some form of signalling is required between the two networks to pass this resource availability information across to the IntServ environment. The mechanism of an end-to-end QoS required in such a model are envisioned as follows:

1. The sending host generates an RSVP PATH message, describing the intended traffic profile of the sending application.
2. The PATH message is carried across the IntServ feeder network, invoking standard RSVP PATH message processing and path state to be installed within the network elements.
3. The PATH message is carried transparently through the DiffServ transit network, and passed to the terminating IntServ stub network.
4. The PATH message is carried across the IntServ terminating network, again invoking standard RSVP PATH message processing.
5. The receiver host generates an RSVP RESV response back to the sender.
6. This RSVP message is passed back through the terminating IntServ network, invoking reservation processing.
7. At the interface to the DiffServ network, a DiffServ admission control system (DACS) is invoked, and requested to compare the current DiffServ resource availability against the requested service profile. If admitted, the DACS would record this reservation and the associated DSCP in the RESV message, then pass the RESV message back through the DiffServ transit network.
8. The RESV message is passed through the IntServ feeder network to the host, invoking reservation processing.
9. The host is then permitted to commence the traffic flows; it can set the DS field in the flow packets to the DSCP value, which maps to the relevant DiffServ service type specified in

the RSVP message.

# 5 MultiProtocol Label Switching (MPLS)

## 5.1 Integration of IP Over ATM

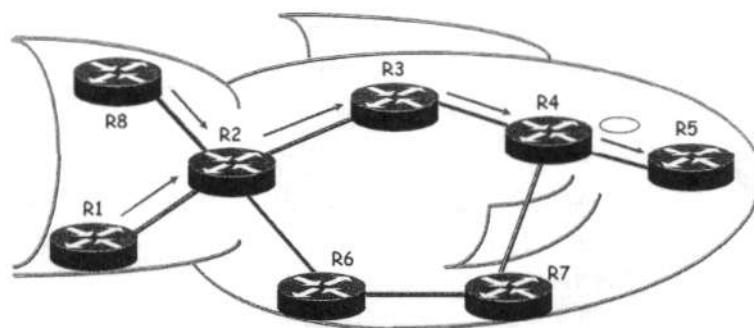
Many factors led to the development of MPLS, among which the integration of IP Over ATM, and the need of extending routing functionality. The problem of mapping IP onto ATM was taken up by a number of standards bodies, primarily the ATM Forum and the IETF. The complexity of the problem can be appreciated to some extent just by counting the number of different working groups that have tackled aspects of this mapping problem:

- *IP over ATM (IPATM)* defined encapsulations for IP datagrams when carried inside ATM adaptation layer PDUs and an address resolution protocol (ATMARP) for mapping IP addresses to ATM addresses, which was later extended to handle multicast.
- *IP over Large Public Data Networks (IPLPDN)* and later *Routing Over Large Clouds (ROC)* defined the Next Hop Resolution Protocol (NHRP) to enable widely separated hosts and routers to establish direct virtual circuits across an ATM network.
- *LAN Emulation (LANE)* defined procedures to make an ATM network appear to behave more like a multiaccess LAN.
- *Multiprotocol Over ATM (MPOA)* combined and extended the work of many of the other groups to support multiple network layer protocols (as opposed to just IP)
- And others...

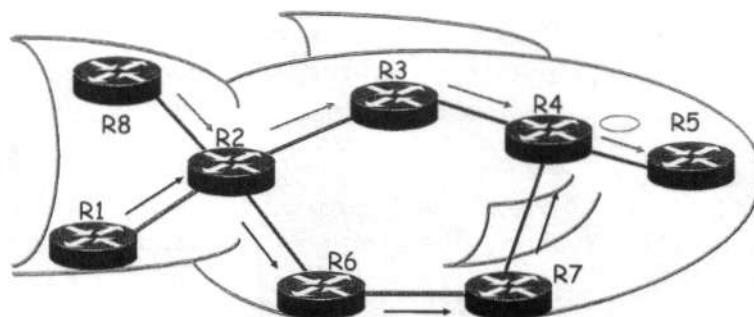
All the complexity derives from the fact that the Internet protocols and the ATM protocols were developed with no regard for each other. This situation induced a lot of people to wonder if ATM switches could be used with a different set of protocols than those defined by the ATM Forum and the ITU - a set of protocols that are more consistent with the Internet architecture. In order to avoid the overlay model and the complexity it brings, various technologies were designed: Cell Switching Router or CSR by Toshiba, IP Switching by Ipsilon, Tag Switching by CISCO, and ARIS by IBM. Instead of having two different protocol architectures with different addressing, routing protocols, resource allocation schemes, and so on, all of the proposals enable IP control protocols to run directly on ATM hardware. The ATM switches still forward packets using label swapping, but the mechanisms by which they set up the forwarding tables and allocate resources are all driven by IP control protocols (OSPF, RSVP, etc.). From a control point of view, the ATM switches effectively become IP routers, thus removing the need to map between IP and ATM control protocols.

## 5.2 Extending Routing Functionality

All routing today in IP networks is destination-based; that is, the decision about where to forward a packet is made based only on its destination address. In principle, other fields in the IP header (e.g., source address, Type of Service bits) could be used when deciding where to forward a packet. As a packet of a connectionless network layer protocol travels from one router to the next, each router makes an independent forwarding decision for that packet. The process of implementing traditional IP routing also is called hop-by-hop destination-based unicast routing.



Example: router R2 forwards packets using only the IP destination address. When a packet arrives at R2, the forwarding decision is not affected by any factor other than destination address. Problem: Suppose we want router R2 to implement the policy: "Packets arriving from R8 that are going to router R5 should go via router R3, while all other packets destined for R5 should go via router R6".



A forwarding mechanism that looks only at destination clearly cannot implement this policy.

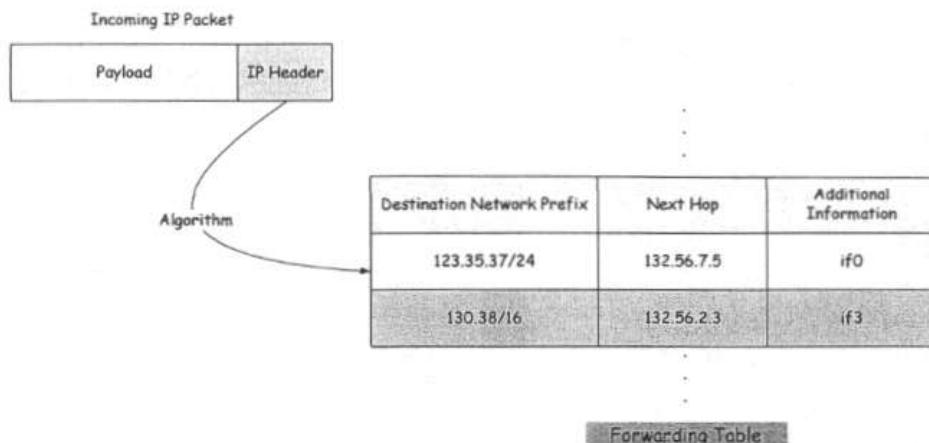
### 5.3 The MPLS Working Group

Without a standard group there would be a proliferation of incompatible label switching techniques. An effort to charter a working group began, and a charter was successfully accepted by IETF in 1997. The name *MultiProtocol Label Switching (MPLS)* was adopted primarily because the names IP Switching and Tag Switching were each associated with the products of a single company, and a vendor neutral term was required. MPLS is a new forwarding mechanism in which packets are forwarded based on labels. Labels may correspond to IP destination networks (equal to traditional IP forwarding). Labels can also correspond to other parameters (QoS, source address, etc.).

### 5.4 MPLS Components

Traditional IP (or network layer) routing functions can be partitioned into two separate functional components: *Control* and *Forwarding*. The *Control Component* is responsible for the construction and maintenance of the forwarding table. Consists of one or more routing protocols (e.g. OSPF, BGP) that provide exchange of routing information among routers, as well as the procedure (or algorithms, e.g. Dijkstra) that a router uses to convert this information into a forwarding table.

The *Forwarding Component* is responsible for the actual forwarding of packets from input to output across a switch or router. To forward a packet the forwarding component uses two sources of information: a forwarding table maintained by a router and some information carried in the packet header itself. The algorithm specifies the information from the packet header that a router uses to find a particular entry in its forwarding table, and the exact procedure that the router uses for finding the entry.



Forwarding can be of unicast packets, unicast packets with Types of Service (ToS), or multicast packets.

	Unicast Forwarding	Unicast Forwarding with Type of Service	Multicast Forwarding
Information from a packet header that a router uses to find a particular entry in the forwarding table	IP destination address	IP destination address & Type of Service (ToS) value	IP source and destination addresses & incoming interface
Forwarding Algorithm	Longest match on destination address	Longest match on destination address & exact match on Type of Service	Longest match on destination address & exact match on source address, and incoming interface

## 5.5 Forwarding Equivalent Classes (FECs)

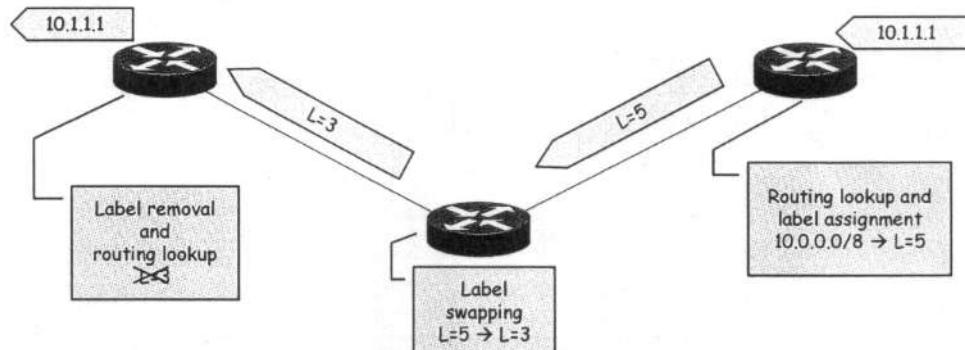
Choosing the next hop can be conceptually regarded as the composition of two functions: the first one partitions the entire set of possible packets into finite number of disjoint subsets which (in MPLS) are called *Forwarding Equivalence Classes (FECs)*; the second one maps each subset (FEC) to a next hop. Therefore, the construction of a forwarding table by the control component could be modeled as constructing a set of FECs and the next hop for each of these FECs. Examples of FECs are:

- a set of unicast packets whose IP destination address matches a particular IP address prefix;
- a set of multicast packets with the same source and destination IP addresses;
- a set of multicast packets with the same source and destination IP addresses and the same incoming interface;
- a set of unicast packets whose IP destination addresses match a particular IP address prefix and whose Type of Service (ToS) bits are the same;
- a set of unicast packets whose IP destination addresses match a particular IP address prefix and have the same destination TCP port number.

## 5.6 Conventional IP Routing Reformulated

As the packet traverses the network, each router examines the content of the IP header and assigns the packet itself to an FEC (costly operation from the processing standpoint). From the forwarding decision standpoint, packets which get mapped into the same FEC are indistinguishable even if these packets may differ from each other with respect to the information they carry in the IP header. Example: an FEC could include all the packets whose IP destination address matches a particular address prefix (although they may have a different IP source address). MPLS is able to forward packets on a given path without re-evaluating the FEC at each router. The FEC evaluation will be done only once when the packet enters the network, where the FEC value will be mapped into a *label*. A *label* is a short, fixed length, and locally significant identifier which is used to identify a FEC. The label will be inserted into the packet to avoid any further evaluation. At each router along the path, the label will be swapped and the labeled packet sent to the next hop.

Example:



Only edge routers must perform a routing lookup. Core routers switch packets based on simple label lookups, and swap labels.

## 5.7 MPLS Architecture

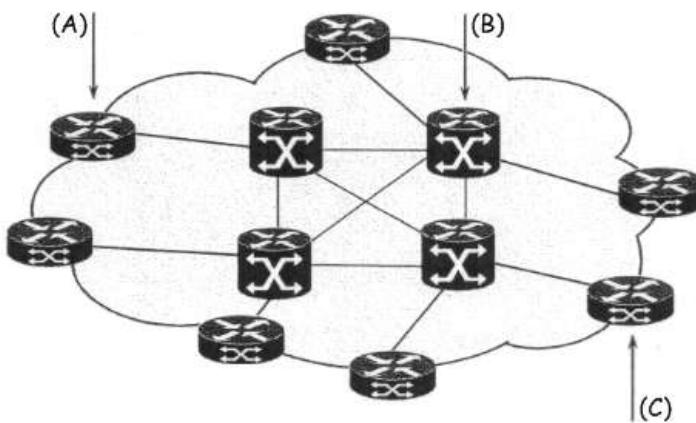
MPLS has two major components:

- *Control Component (Plane)*: exchanges Layer 3 routing information and labels.
- *Data Component (Plane)*: forwards packets based on labels.

Control Plane contains protocols to exchange routing information, such as OSPF, to exchange labels, such as Label Distribution Protocol (LDP) and Resource reSerVation Protocol (RSVP) and to maintain contents of the label-switching table. Data Plane has a simple forwarding engine.

## 5.8 The Forwarding Component

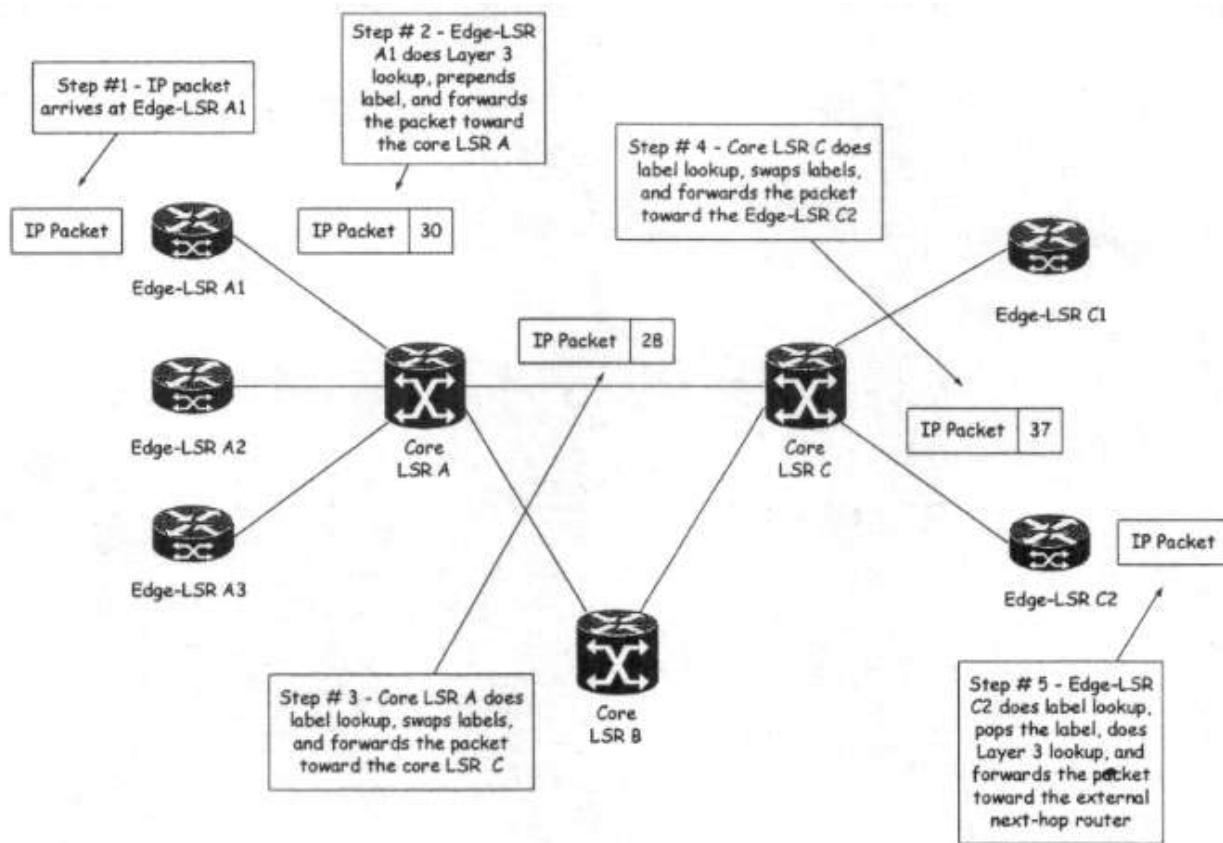
To make a forwarding decision on a packet, the MPLS forwarding component (also called the Data Plane) uses a Label carried in the packet, and a Forwarding Table maintained by each router.



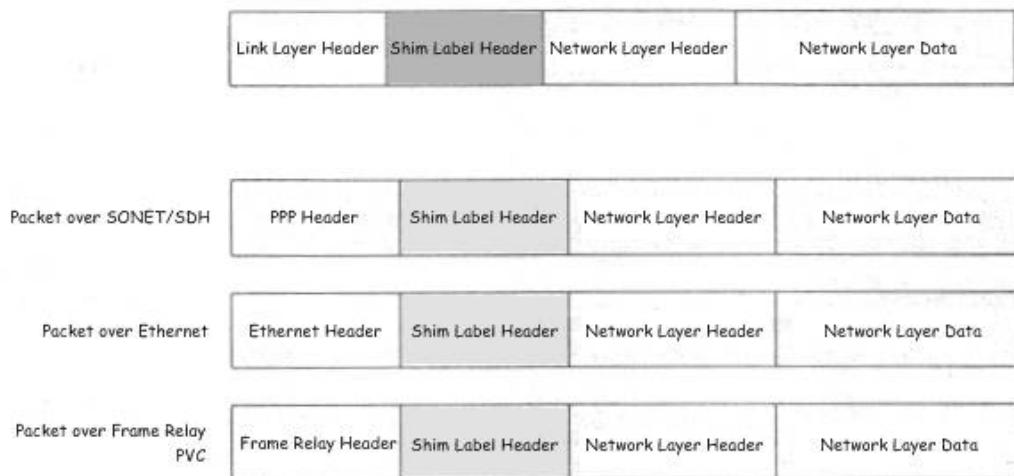
(A) The Ingress Edge-LSR (Label Switching Routers or LSRs) receives an IP packet, classifies the packet into a forward equivalence class (FEC) and performs Label Imposition (also called Push action) which is the act of prepending a label, or a stack of labels, to the IP packet. Note: In MPLS, the assignment of a particular packet to a particular FEC is done just once, when the packet enters the network.

(B) When Core-Label Switching Routers (Core-LSRs) receive this labeled packet there is no more analysis of the packet header, but the forwarding consists of using the label as an index to a table specifying the next hop, the new label and the encapsulation over the output link to reach the next router.

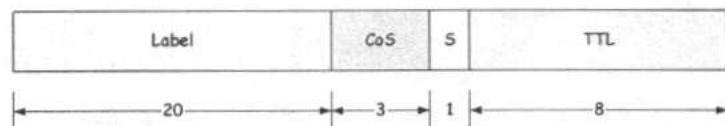
(C) When the Egress-Edge-LSR for this particular FEC receives the labeled packet, it performs Label Disposition (also called Pop action) which is the act of removing the last label from a packet at the egress point before it is forwarded (by means of a traditional Layer 3 lookup on the resulting IP packet) to a neighbor router outside the MPLS domain.



With ATM the label could be carried in VPI/VCI fields of the ATM Cell Header. For those link layer technologies that cannot accommodate labels in the link layer header (e.g., Ethernet, FDDI, Token Ring, point-to-point links) MPLS uses a *Shim Label Header*.



## 5.9 MPLS Label Format



✓ <i>Label</i>	MPLS Label
✓ <i>CoS</i>	Class of Service
✓ <i>S</i>	Bottom of Stack
✓ <i>TTL</i>	Time To Live

- **Label:** carries the actual value of the MPLS label.
- **CoS:** affects the queuing and discard algorithms applied to the packet as it is transmitted through the network.

- Label Stack: supports a hierarchical label stack (see below).
- TTL: the TTL field is similar to the time-to-live field carried in the IP header. The MPLS router only processes the TTL field in the top entry of the label stack. The IP TTL field contains the value of the IPv4 TTL field or the value of the IPv6 Hop Limit field - whichever is applicable.

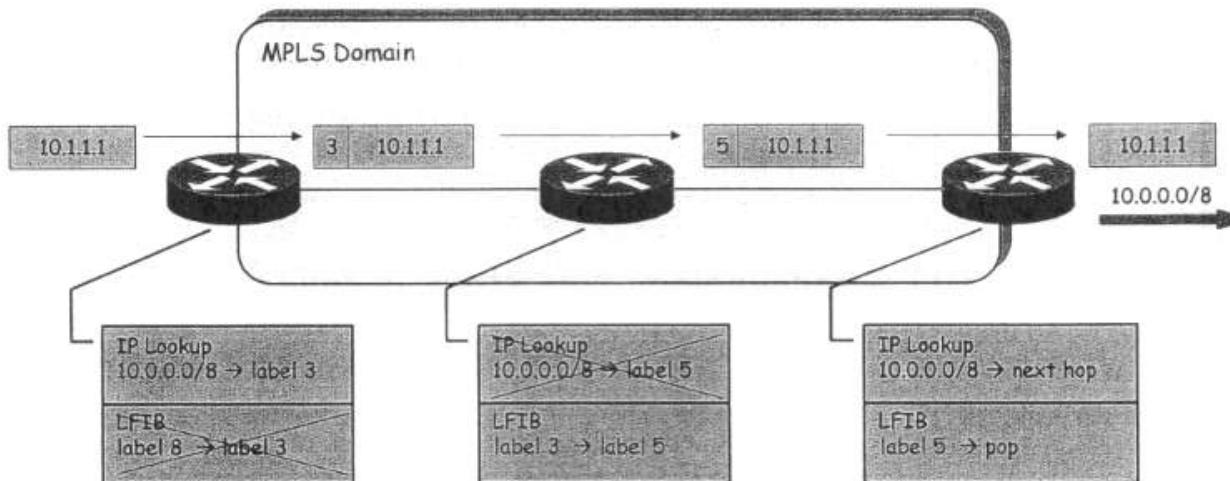
Note: ATM MPLS nodes carry labels in the VCI or VPI/VCI field of the ATM header. The CoS, Stack, and TTL fields are not supported. However, QoS and loop-detection features are still available and can be implemented using ATM mechanisms.

Support of the shim header requires that the sending router have a way to indicate to the receiving router that the frame contains a shim header. This is facilitated differently in various technologies. Ethernet uses ethertype values 0x8847 and 0x8848 to indicate the presence of a shim header. Ethertype value 0x8847 is used to indicate that a frame is carrying an MPLS unicast packet, and ethertype value 0x8848 is used to indicate that a frame is carrying an MPLS multicast packet. Token Ring and FDDI also use the type values as part of the SNAP header. PPP uses a modified Network Control Program (NCP) known as MPLS control protocol (MPLSCP) and marks all packets containing a shim header with 0x8281 in the PPP protocol field. Frame Relay uses the SNAP Network Layer Protocol ID (NLPID) and SNAP header marked with type value 0x8847 in order to indicate frames carrying shim headers.

## 5.10 Frame-mode & Cell-mode Operations

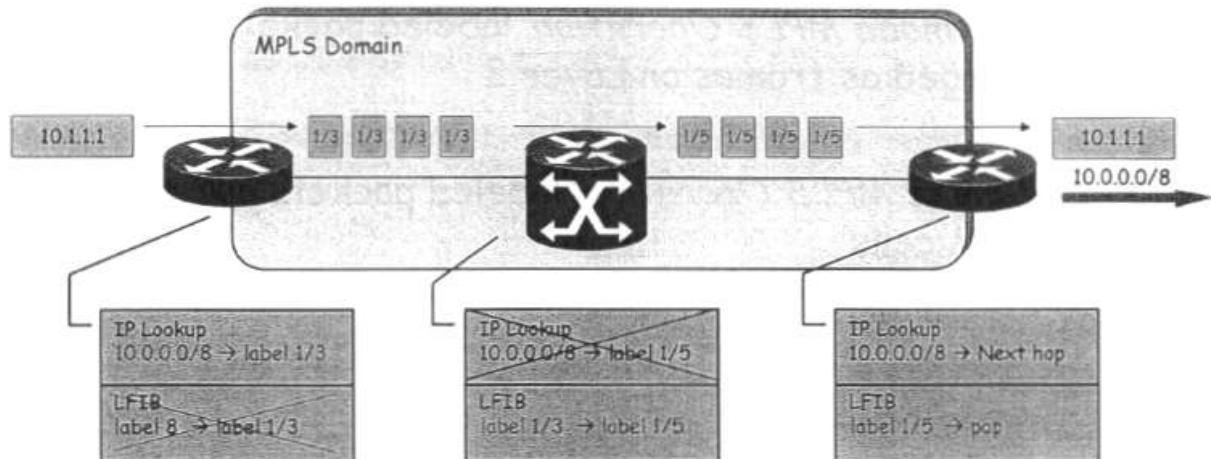
- *Frame-mode MPLS Operation*: labeled packets are exchanged as frames on Layer 2.
- *Cell-mode MPLS Operation*: labeled packets are transported as ATM cells.

Frame-mode:



On ingress a label is assigned and imposed by the IP routing process. LSRs in the core swap labels based on the contents of the label forwarding table. On egress the label is removed and a routing lookup is used to forward the packet.

Cell-mode:

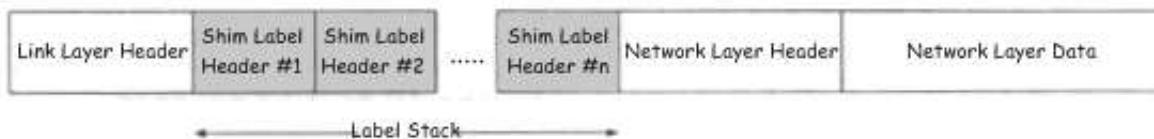


Labels (VPI/VCI) are imposed during the IP lookup process on ingress ATM edge LSRs. Packets are segmented into cells. ATM LSRs in the core swap labels based on the contents of the ATM

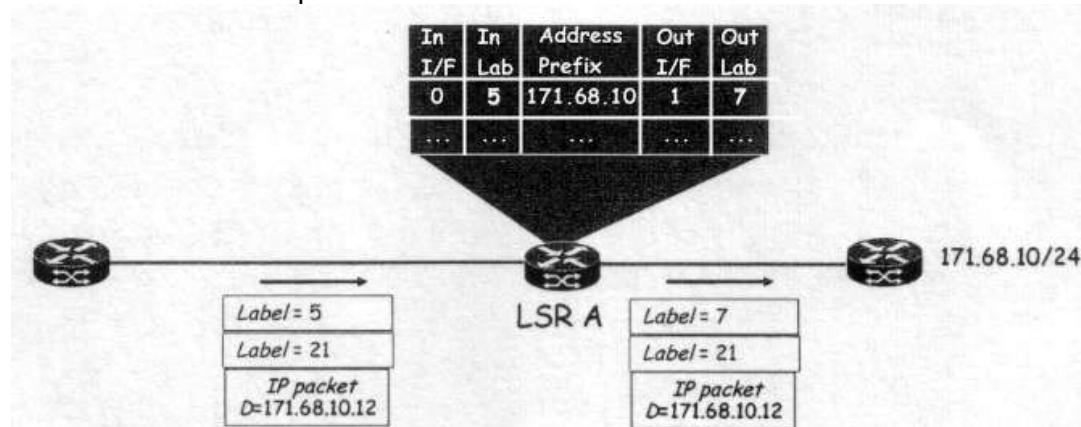
switching table. ATM LSRs cannot forward IP packets. On egress ATM edge LSRs the labels are removed (cells are reassembled into packets) and a routing lookup is used to forward packets.

## 5.11 Label Stack

MPLS implements a general model in which a labeled packet may carry a number of labels, organized as a last-in, first-out stack. In the MPLS terminology this stack is called *Label Stack*.



MPLS supports a hierarchical label stack. The *stack bit* is set to 1 to indicate the bottom of the stack. All other stack bits are set to 0. In packet-based MPLS, the top of the stack appears right after the link layer header, and the bottom of the label stack appears right before the network layer header. Packet forwarding is accomplished using the label values of the label on the top of the stack. Unicast IP routing does not use stacked labels, but MPLS VPNs and traffic engineering utilize stacked labels for their operation.



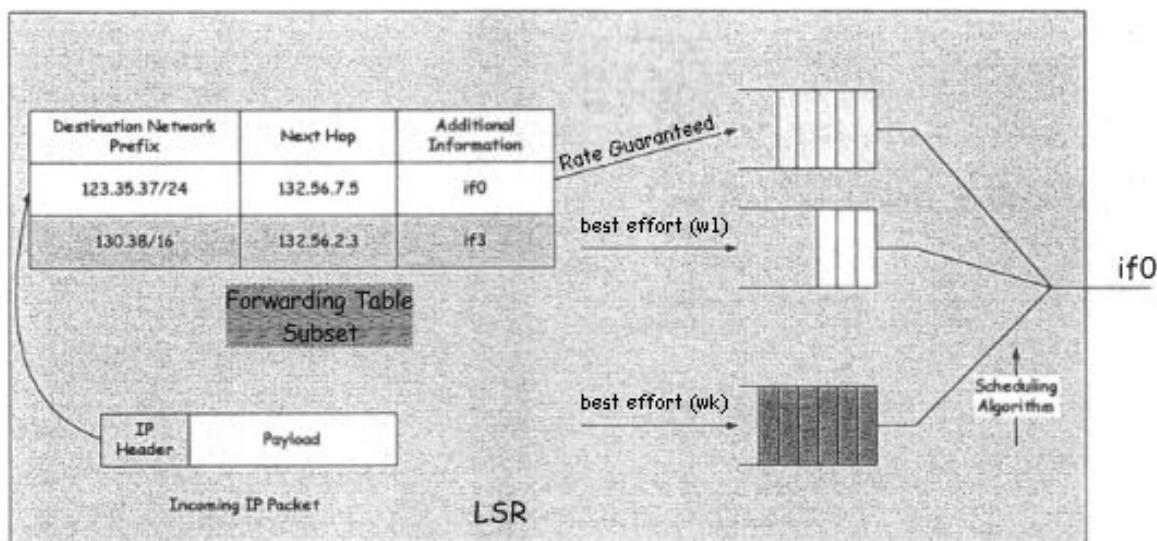
LSR A forwards the labeled packet based on the label at the top of the label stack.

## 5.12 Label Forwarding Information Base (LFIB)

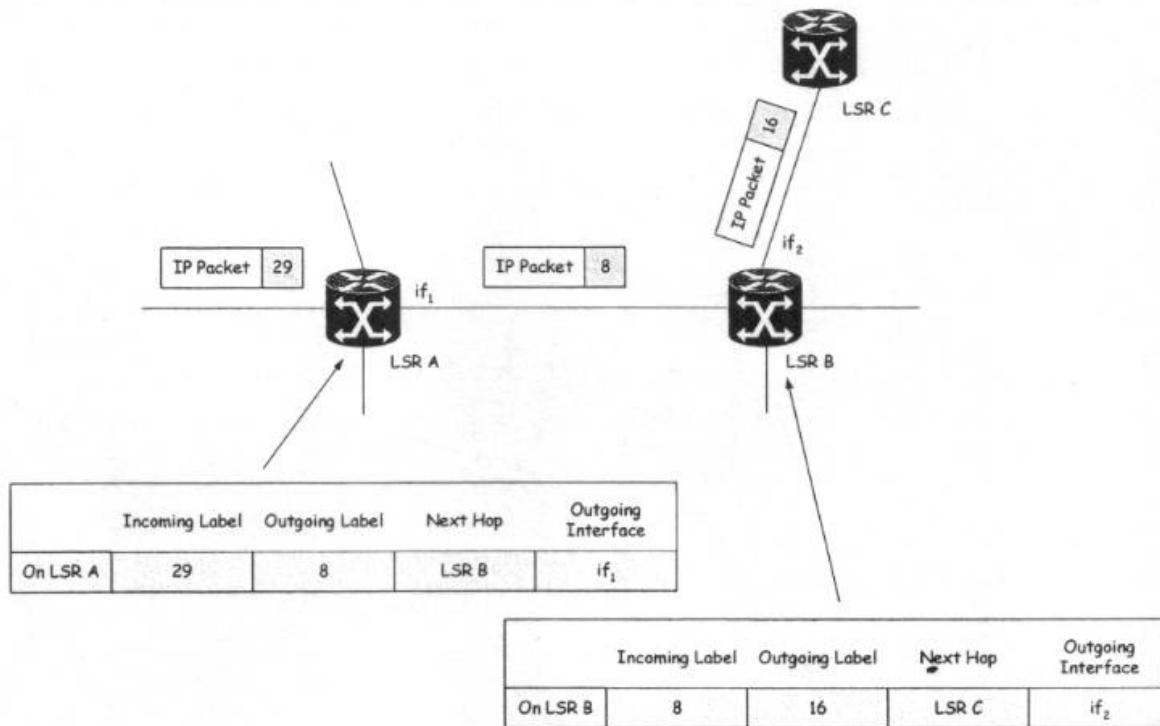
*Label Forwarding Information Base (LFIB)* is used during the actual forwarding of packets and holds only labels that are in use currently by the forwarding component of MPLS. Consists of a sequence of entries structured as follows:

Incoming Label	First Subentry	Second Subentry
Incoming Label	Outgoing Label Outgoing Interface Next Hop Address	Outgoing Label Outgoing Interface Next Hop Address

Different subentries within an individual entry may have either the same or different outgoing labels. There may be more than one subentry in order to handle multicast forwarding, where a packet that arrives on one interface would need to be sent out on multiple outgoing interfaces. An entry in the forwarding table may also include the information related to what resources the packet may use, such as a particular outgoing queue that the packet should be placed on.



LS Forwarding Example:



### 5.13 Conventional Routing vs Label Switching

In the conventional IP routing architecture, different functionality provided by the control component requires multiple forwarding algorithms in the forwarding component.

	Unicast Forwarding	Unicast Forwarding with Type of Service	Multicast Forwarding
Information from a packet header that a router uses to find a particular entry in the forwarding table	IP destination address	IP destination address & Type of Service (ToS) value	IP source and destination addresses & incoming interface
Forwarding Algorithm	Longest match on destination address	Longest match on destination address & exact match on Type of Service	Longest match on destination address & exact match on source address, and incoming interface

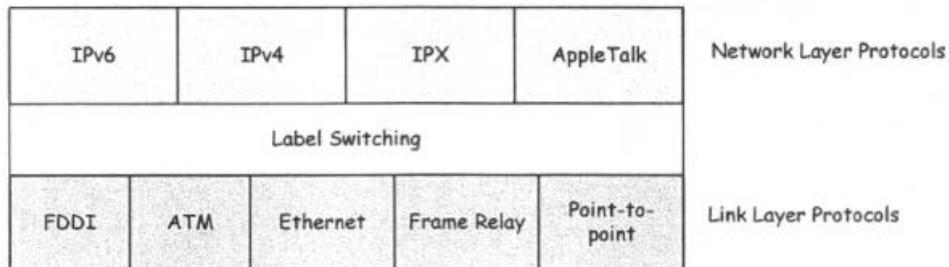
In MPLS there is just one forwarding algorithm in the forwarding component - the algorithm

based on label swapping.

	Unicast Forwarding	Unicast Forwarding with Type of Service	Multicast Forwarding
Forwarding Algorithm	Common Forwarding (Label Swapping)		

## 5.14 MultiProtocol

The forwarding component is not specific to a particular network layer (for example, the same forwarding component could be used when doing label switching with IP as well as when doing label switching with IPX). This makes label switching suitable as a multiprotocol solution with respect to the network layer protocols.



Label switching is also capable of operating over virtually any link layer protocol. This makes label switching a multiprotocol solution with respect to the link layer protocols. These properties of label switching explain the name given to the IETF working group that is working to standardize this technology: MultiProtocol Label Switching (MPLS).

## 5.15 MPLS Key Properties

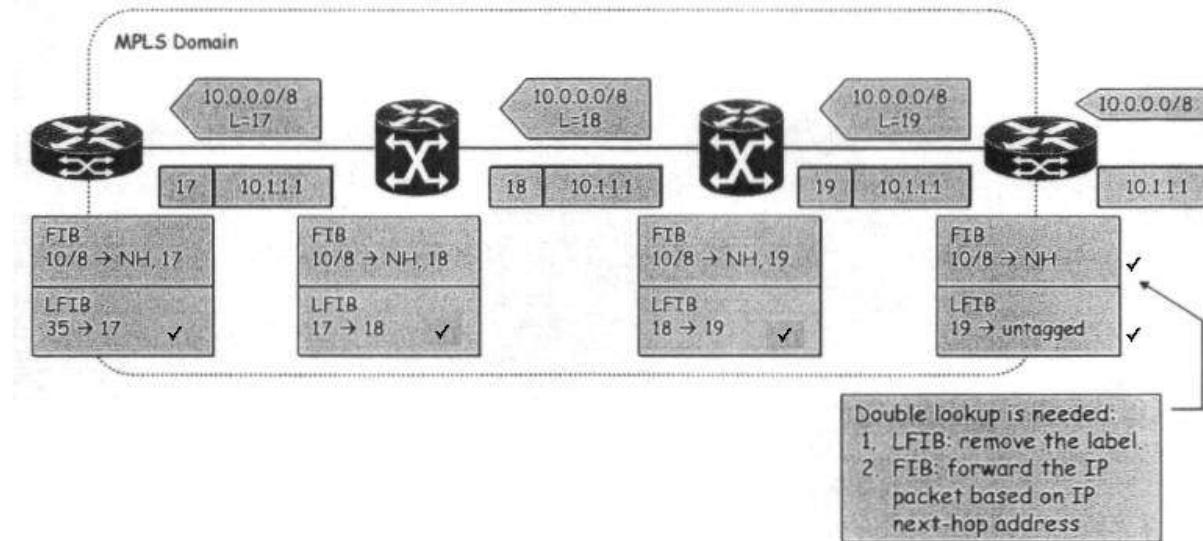
Key properties of the label switching forwarding component are:

- It uses a single forwarding algorithm based on label swapping.
- It doesn't place any constraints on the forwarding granularity that could be associated with a label.
- It can support multiple network layer protocols as well as multiple link layer protocols.

## 5.16 Penultimate Hop Popping

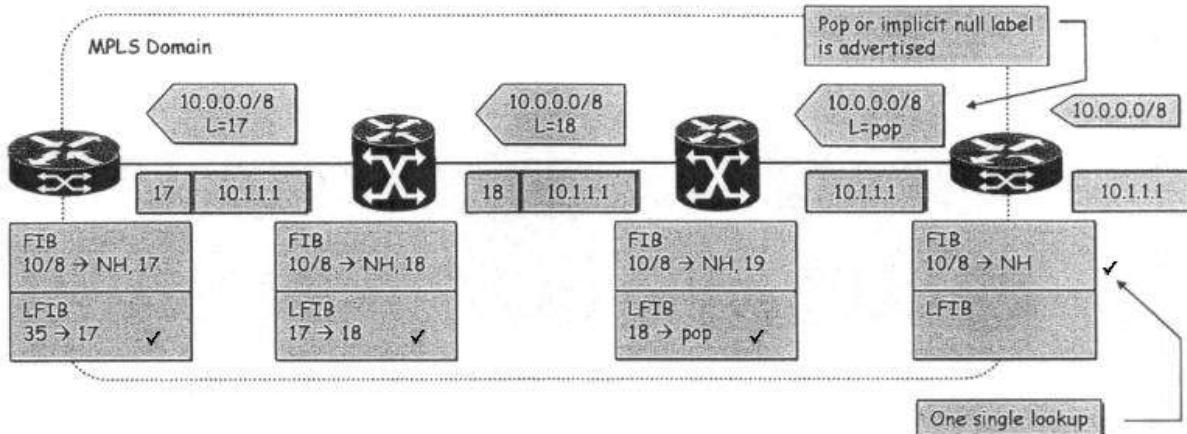
An egress Edge-LSR in an MPLS network might have to perform two lookups on a packet received from an MPLS neighbor and destined for a subnet outside the MPLS domain. It must inspect the label in the label stack header, and it must perform the label lookup just to realize that the label has to be popped and the underlying IP packet inspected. An additional Layer 3 lookup must be performed on the IP packet before it can be forwarded to its final destination. The double lookup in the Edge-LSR might reduce the performance of that node. Furthermore, in environments where MPLS and IP switching is realized in hardware, the fact that a double lookup might need to be performed can increase the complexity of the hardware implementation significantly. To address both issues, *Penultimate Hop Popping (PHP)* was introduced into the MPLS architecture. With penultimate hop popping, the Edge-LSR can request a label pop operation from its upstream neighbors. Penultimate hop popping is requested through LDP by using a special label value (3) that also is called the implicit-null value, i.e. the pop or implicit-null label uses value 3 when being advertised to a neighbor. When the egress LSR requests penultimate hop popping for an IP prefix, the local LIB entry in the egress LSR and the remote LIB entry in the upstream LSRs indicate the imp-null value and the LFIB entry in the penultimate LSR indicates a label pop operation.

Double Lookup Scenario:



Double lookup is not an optimal way of forwarding labeled packets. A label can be removed one hop earlier.

#### Penultimate Hop Popping:

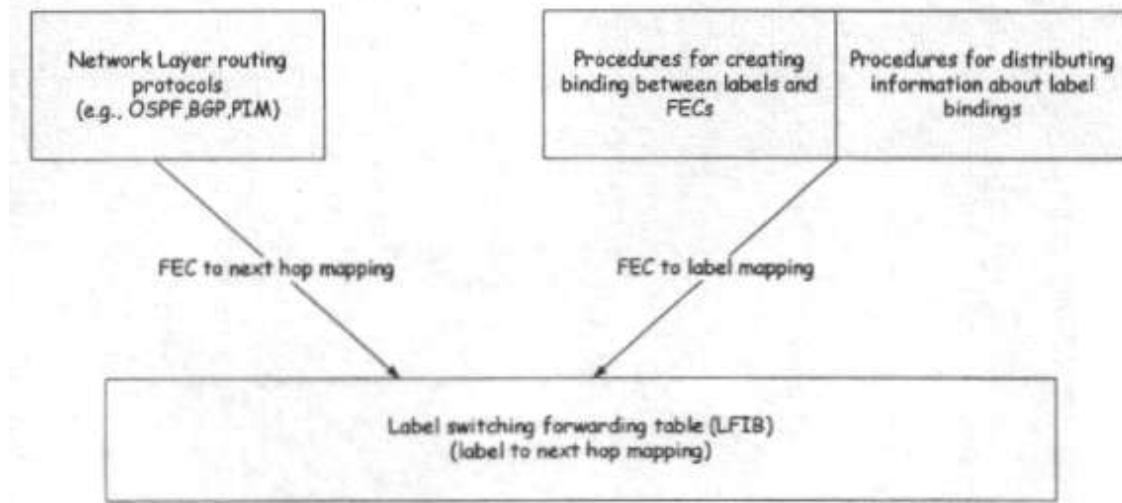


A label is removed on the router before the last hop within an MPLS domain. Penultimate hop popping optimizes MPLS performance (one less LFIB lookup). PHP does not work on ATM (VPI/VCI cannot be removed).

### 5.17 The Control Component

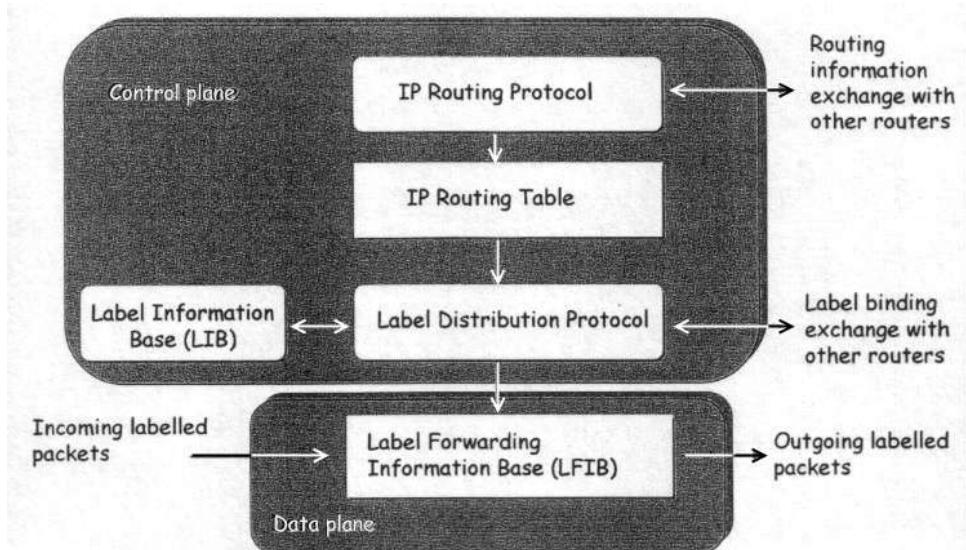
The Control Component (also called the Control Plane) of MPLS includes procedures by which an LSR can:

1. create bindings between labels and FECs;
2. inform other LSRs of the bindings it creates;
3. utilize both (1) and (2) to construct and maintain the forwarding table used by the label switching component.

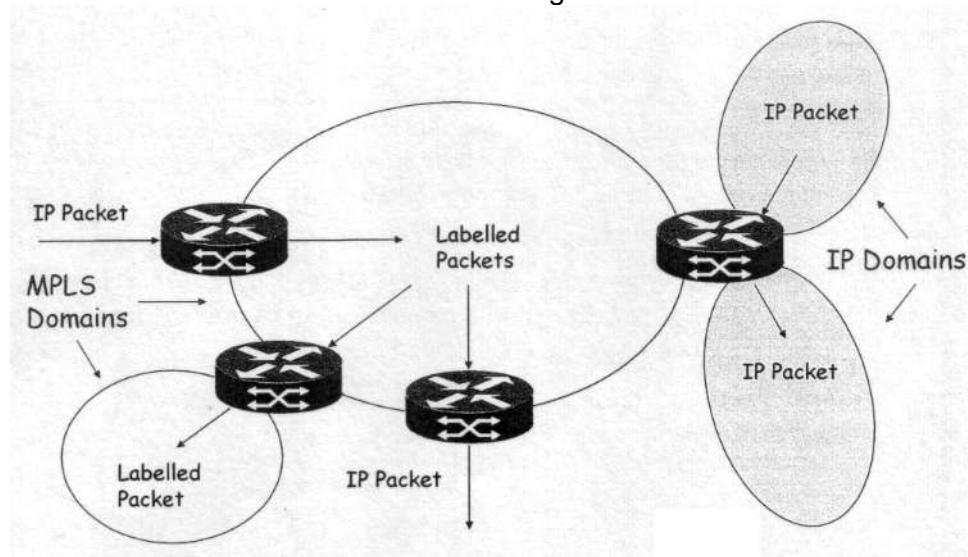


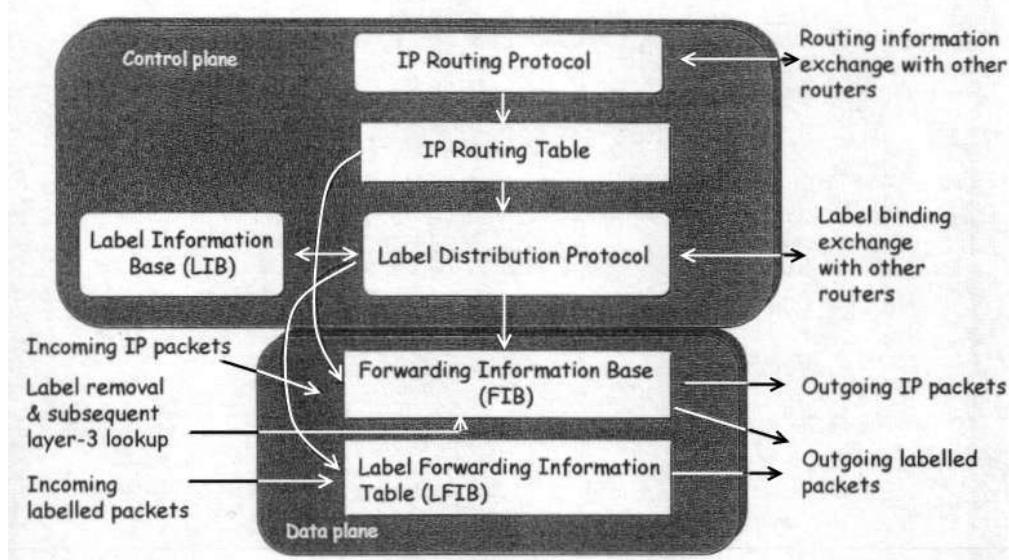
Furthermore, the MPLS control component includes all the routing protocols (e.g., OSPF, BGP, and so forth) used by the control component of the conventional IP routing architecture.

Architecture of LSRs:



Architecture of Edge-LSRs:





## 5.18 LIB and LFIB

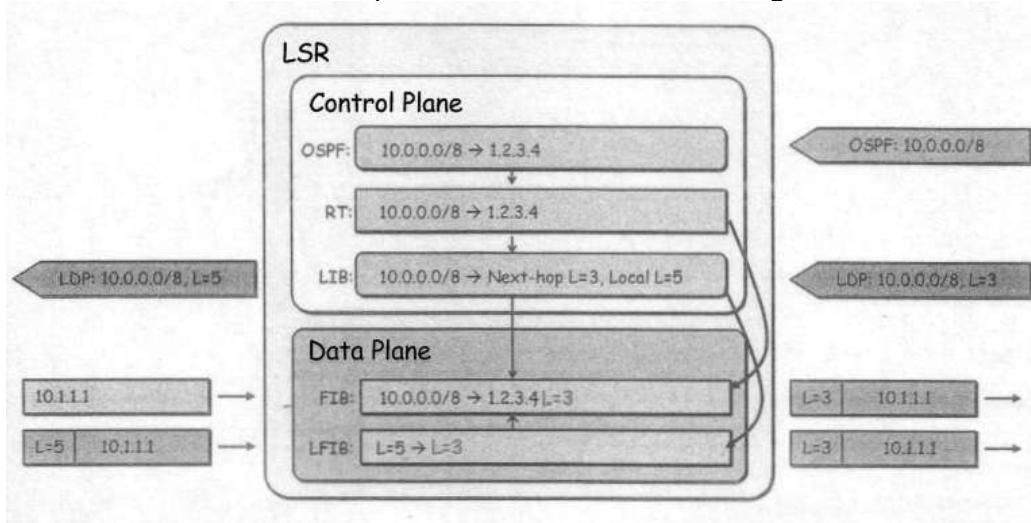
Each LSR keeps two tables, which hold information that is relevant to the MPLS forwarding component.

*Label Information Base (LIB)* holds all labels assigned by this LSR and the mappings of these labels to labels received from any neighbors (these label mappings are distributed through the use of Label Distribution Protocols).

*Label Forwarding Information Base (LFIB)* is used during the actual forwarding of packets and holds only labels that are in use currently by the forwarding component of MPLS.

Incoming Label	First Subentry	Second Subentry
Incoming Label	Outgoing Label Outgoing Interface Next Hop Address	Outgoing Label Outgoing Interface Next Hop Address

An Example of MPLS Unicast IP Routing:

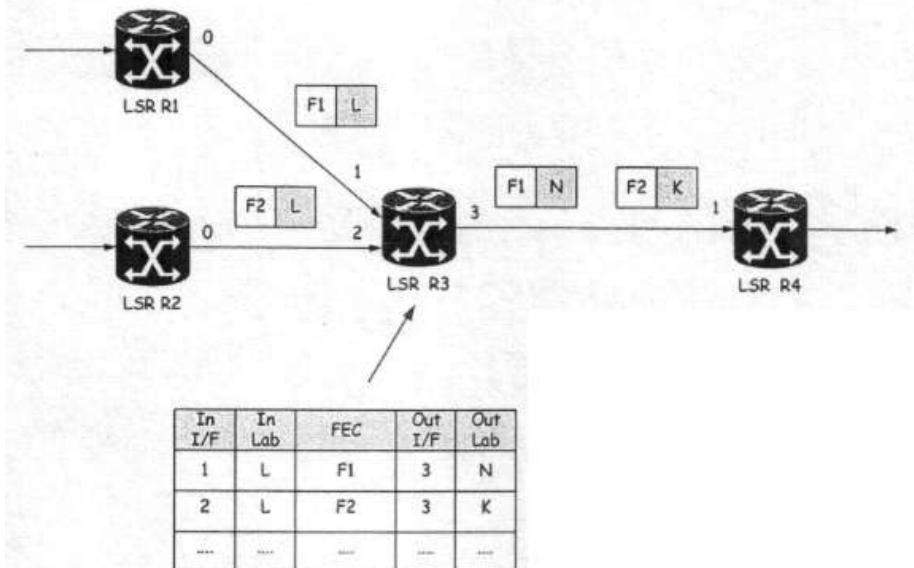


## 5.19 Free Labels

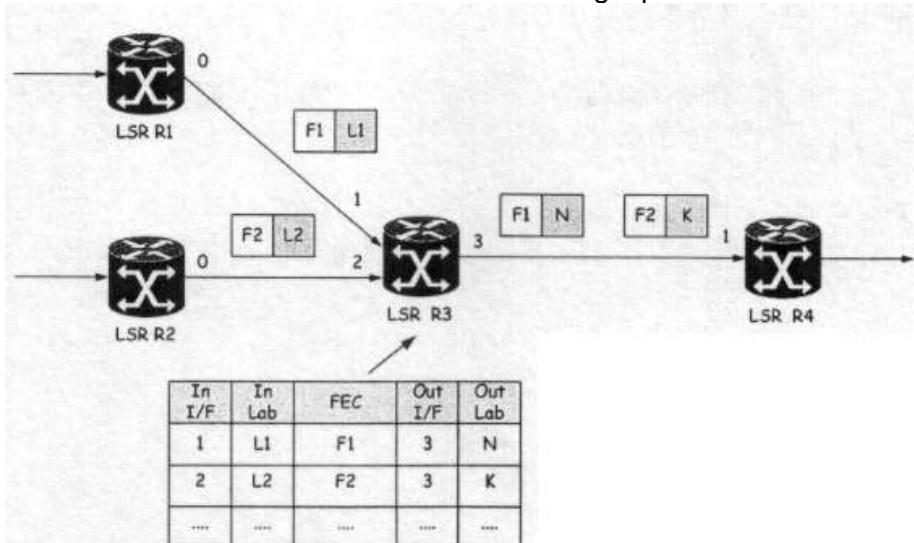
An LSR maintains a pool of free labels (i.e. labels with no bindings). When the LSR is first initialized, the pool contains all possible labels that the LSR can use for local binding. When the router creates a new local binding, the router takes a label from the pool. When the router destroys a previously created binding, the router returns the label associated with that binding to the pool.

## 5.20 Label Space

Labels can be unique on a per interface basis (*Per Interface Label Space*). The LSR maintains pools of labels on a per-interface basis.



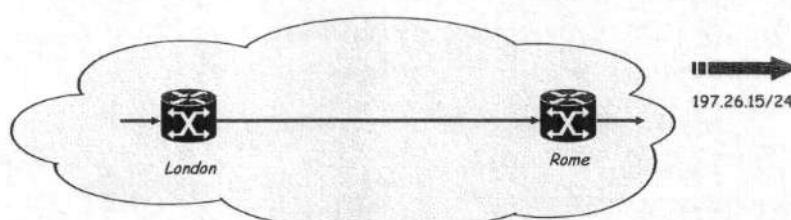
If  $F1=F2$  (i.e. FEC1=FEC2), two outgoing flows may or may not be aggregated. Labels can be unique on a per LSR basis instead. The LSR maintains a single pool of labels.



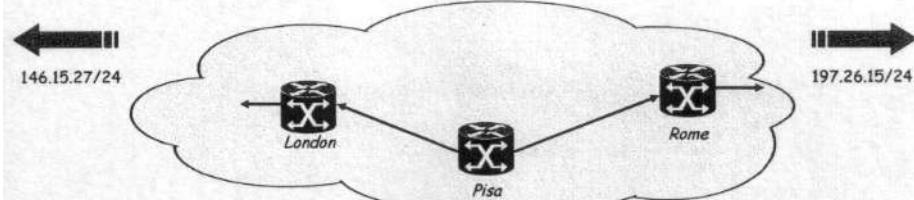
Two different labels (L1 & L2) must be used.

## 5.21 Label Assignment and Allocation

Consider packets flowing from London to Rome.



London is the upstream LSR and Rome is the downstream LSR for FEC 197.26.15/24.

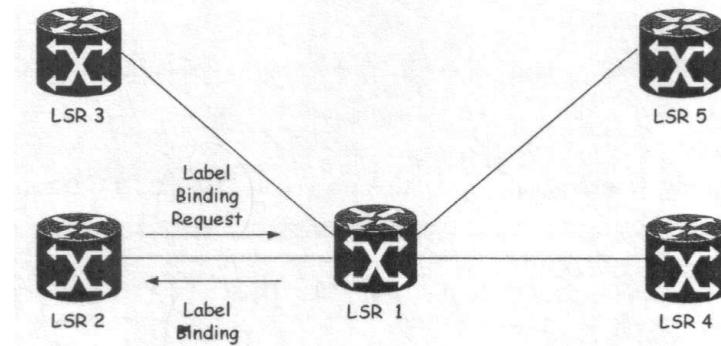


Rome router is downstream neighbor of Pisa for FEC 197.26.15/24. London is downstream neighbor of Pisa for FEC 146.15.27/24.

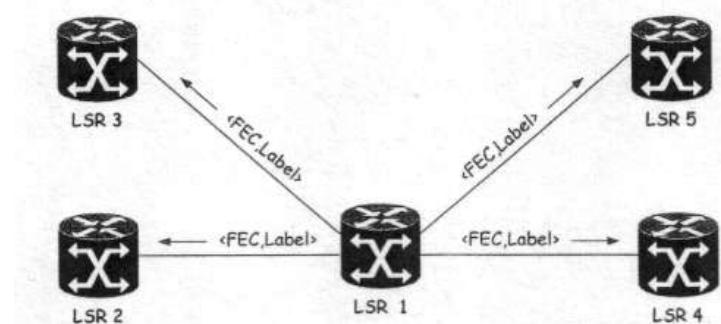
In the MPLS architecture labels are *downstream-assigned*. Label Allocation to a FEC can be:

- *Downstream-on-Demand*
- *Unsolicited Downstream*

Labels are allocated to a FEC by the downstream LSR (LSR 1 in the figure) (*downstream-on-demand*) but only on demand from the upstream LSR (LSR 2 in the figure) and for a specific interface. The label distribution will be done only for the requesting LSR (LSR 1 in the figure).



The MPLS architecture also allows an LSR (LSR 1 in the figure) to distribute bindings to LSRs that have not explicitly requested them (*unsolicited downstream*). The label will be distributed to all adjacent neighbors.



## 5.22 Label Retention Modes

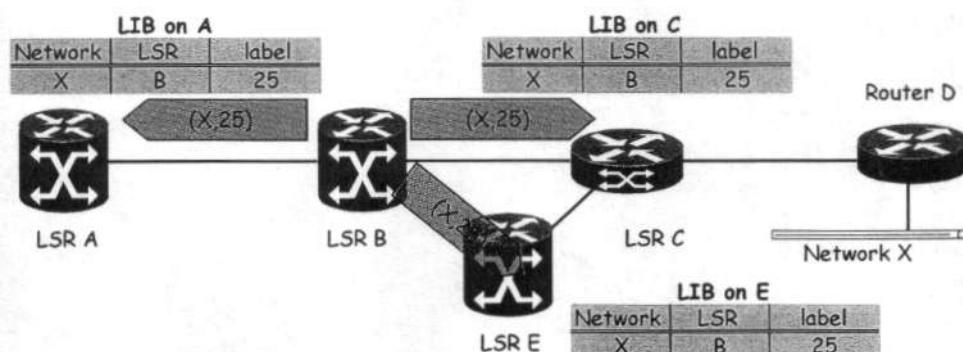
With the unsolicited downstream distribution, LSR X (X=2,3,4,5) may receive a label binding for a particular FEC from the LSR 1 which is not the next hop for that FEC. ASR X has the choice of whether to keep track of such binding, or whether to discard it:

- *Liberal Label Retention Mode*
- *Conservative Label Retention Mode*

*Liberal Label Retention Mode*: If LSR X (X=2,3,4,5) keeps track of such binding, then it may immediately begin using the binding if LSR 1 eventually becomes its next hop for the FEC in question.

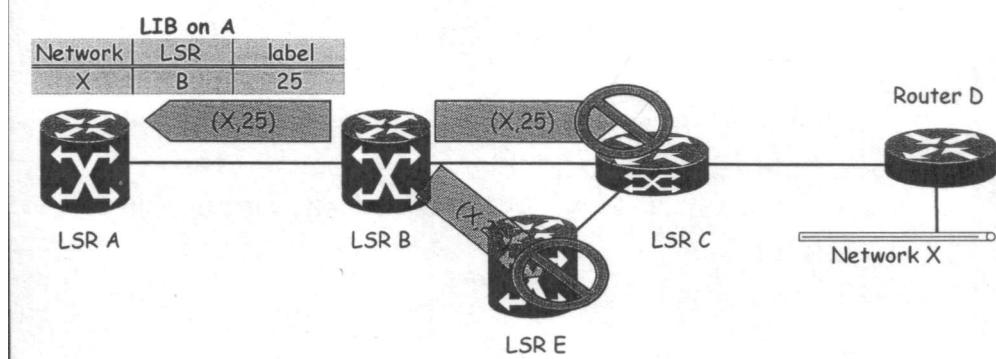
*Conservative Label Retention Mode*: If LSR X (X=2,3,4,5) discards such bindings, then if LSR 1 later becomes the next hop, the binding will have to be reacquired by a downstream-on-demand

Liberal Retention Mode:



Every LSR stores the received label in its LIB, even when the label is not received from a next hop LSR.

## Conservative Retention Mode:

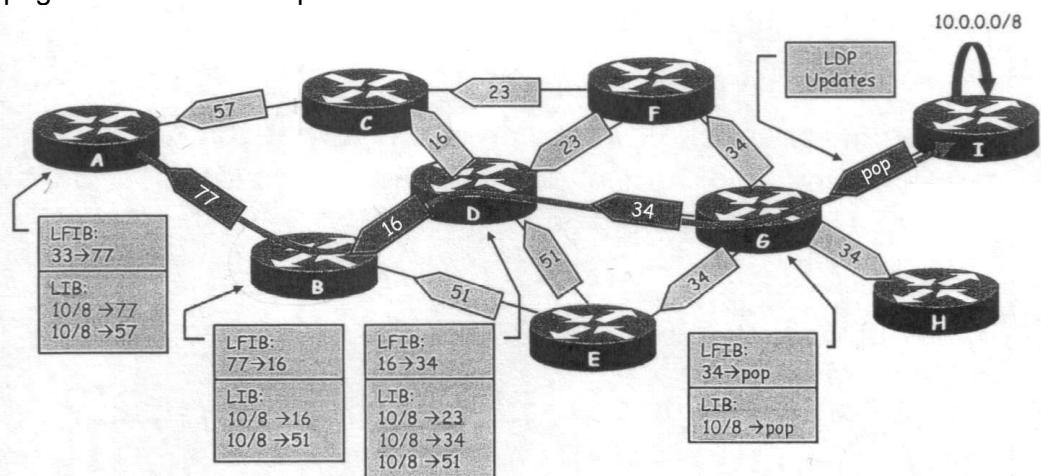


An LSR stores only the labels received from next hop LSRs; all other labels are ignored.

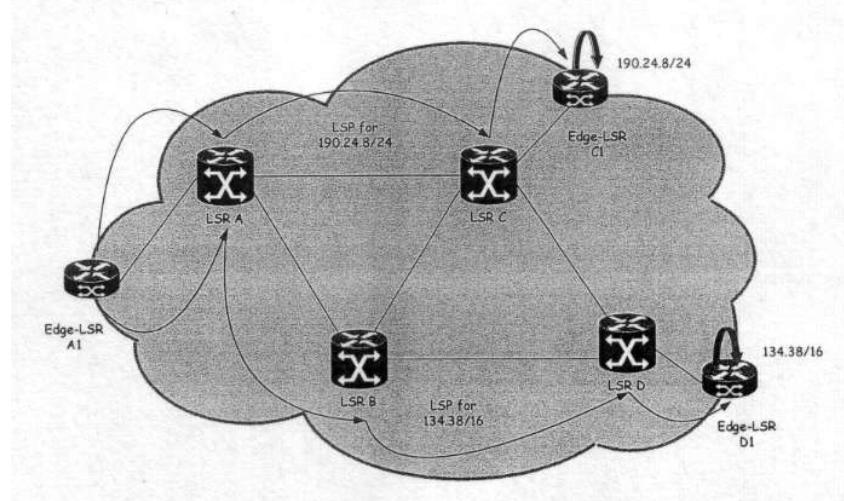
The chief advantage of liberal retention mode is quicker response to changes in routing. Its main drawback is waste of labels which can be particularly important in devices that store label forwarding information in hardware, notably ATM-LSRs. It is common to see conservative label retention used on ATM-LSRs. Standard Parameter Sets in Cisco IOS Platform MPLS Implementation (for Routers with packet interfaces) are: *Per-Platform (LSR) Label Space, Unsolicited Distribution, Liberal Label Retention, Independent Control*.

### 5.23 Label Switched Paths

LDP propagates labels to set up the LSP.



The *Label Switched Path (LSP)* can be considered the path over a set of LSRs that packets belonging to a certain FEC travel in order to reach their destination. In destination-based routing, FECs correspond to address prefixes which are distributed via a (dynamic) routing protocol (e.g. OSPF).



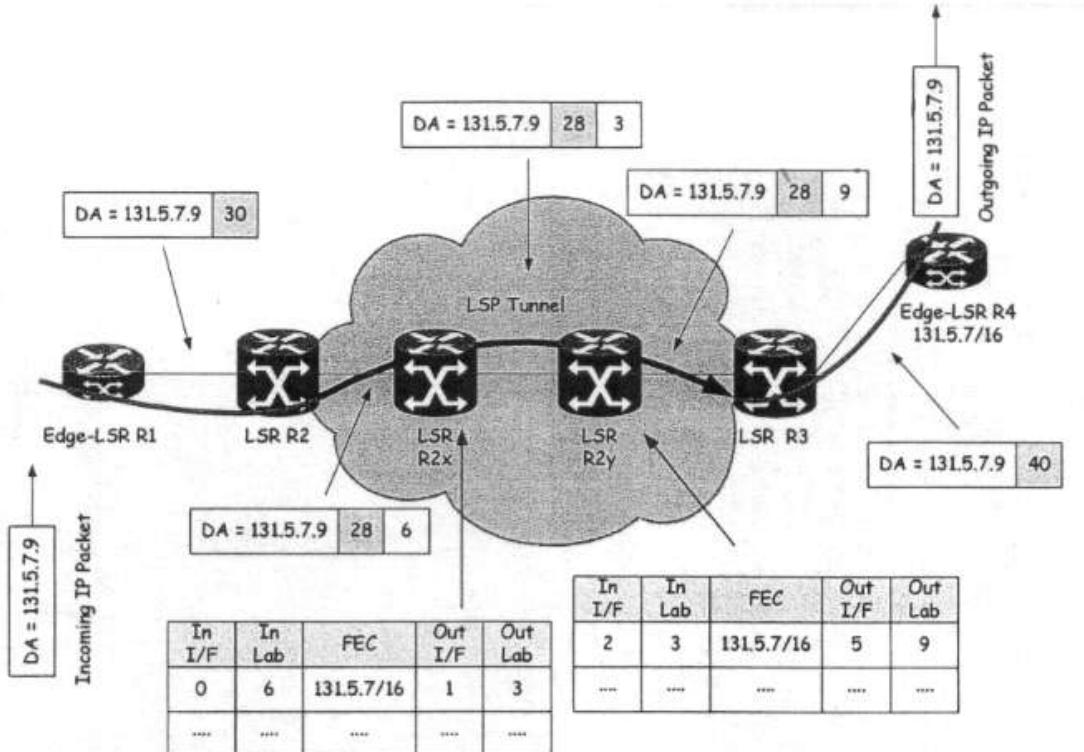
Label Switching with Label Stack:



### 5.24 Example 1

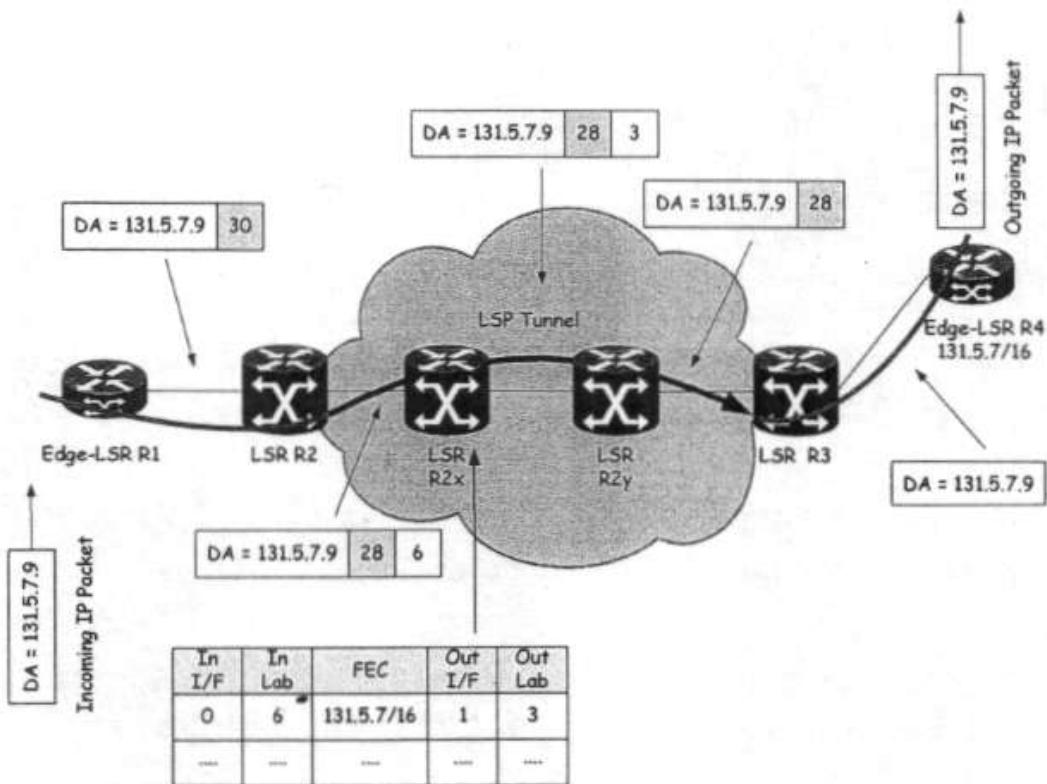
Consider the LSP {R1,R2,R3,R4}. Let us suppose that R1 receives unlabeled packet P, and pushes on its label stack the label to cause it to follow this path. Let us further suppose that R2 and R3 are not directly connected, but are neighbors by virtue of being the endpoints of an LSP tunnel. So the actual sequence of LSRs traversed by P is {R1,R2,R2x,R2y,R3,R4}.

Without Penultimate Hop Popping:



When P travels from R1 to R2, it will have a label stack of depth 1 (30). R2, switching on the label, determines that P must enter the tunnel. R2 first replaces the Incoming Label with a label that is meaningful to R3 (28). Then it pushes on a new label and this level 2 label (6) has a value which is meaningful to R2x. Switching is done on the level 2 label by R2x (swaps label from 6 to 3) and R2y (swaps label from 3 to 9).

With Penultimate Hop Popping:



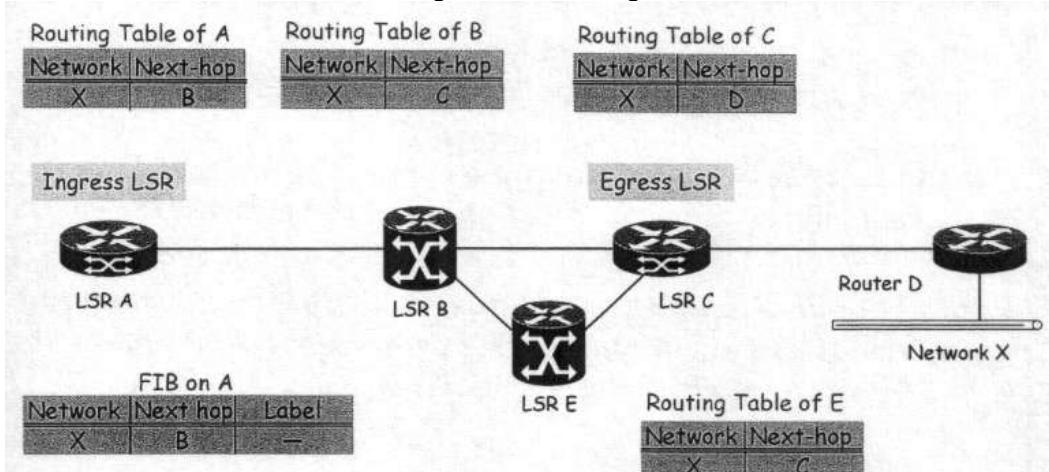
R<sub>2y</sub>, which is the penultimate hop in the R<sub>2</sub>-R<sub>3</sub> tunnel, it pops the label stack before forwarding the packet to R<sub>3</sub>. When R<sub>3</sub> sees packet P, P has only a level 1 label, having now exited the tunnel. Since R<sub>3</sub> is the penultimate hop in P's level 1 LSP, it pops the label stack, and R<sub>4</sub> receives P unlabeled. LSPs are unidirectional, which means that a packet could take a different return path (LSP) on its way back. An LSP is set up according to a connection-oriented paradigm, i.e. the path is set up prior to any traffic flow. For the FECs corresponding to address prefixes which are distributed via dynamic routing protocols, the set up of the LSPs can be performed in one of two ways: *Independent Control* or *Ordered Control*.

## 5.25 Example 2

Here is an example of *Independent Control*, *Unsolicited Downstream*, *Liberal Label Retention Mode*. The LSP set up in a packet-mode MPLS environment for unicast packets follows these steps:

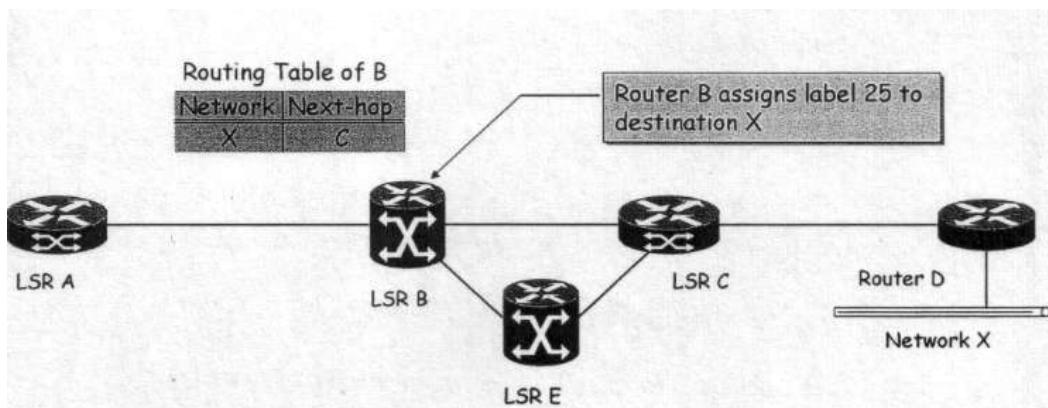
- IP routing protocols build the IP routing table
- Each LSR assigns a label to every destination in the IP routing table independently
- LSRs announce their assigned labels to all other LSRs
- Every LSR builds its LIB, LFIB, and FIB data structures based on received labels

Building the IP Routing Table:



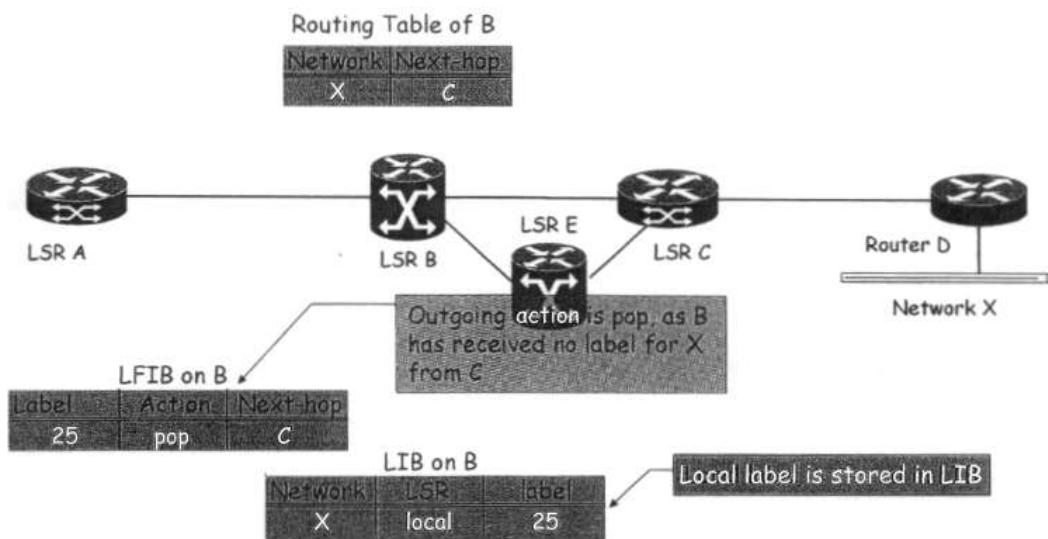
IP routing protocols are used to build IP routing tables on all LSRs. FIBs are built based on IP routing tables with no labeling information.

Allocating Labels:



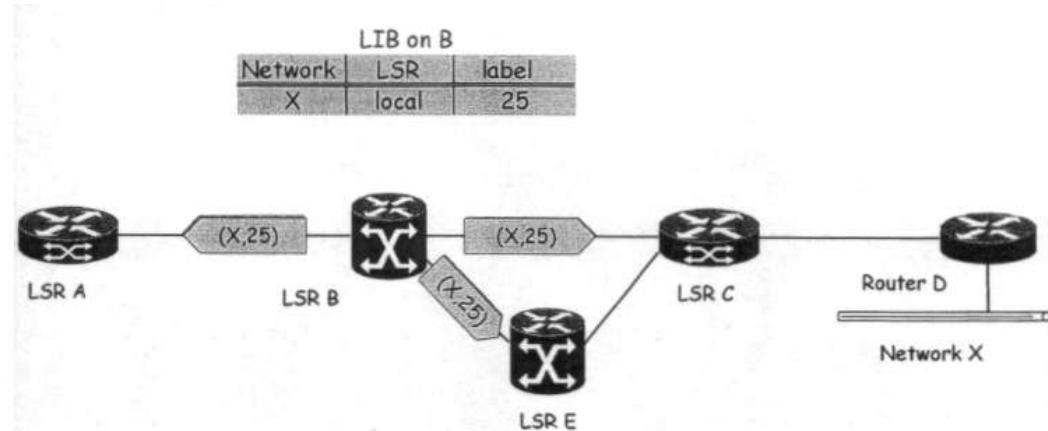
Every LSR allocates a label for every destination in the IP routing table. Labels have local significance. Label allocations are asynchronous.

#### LIB and LFIB Setup:



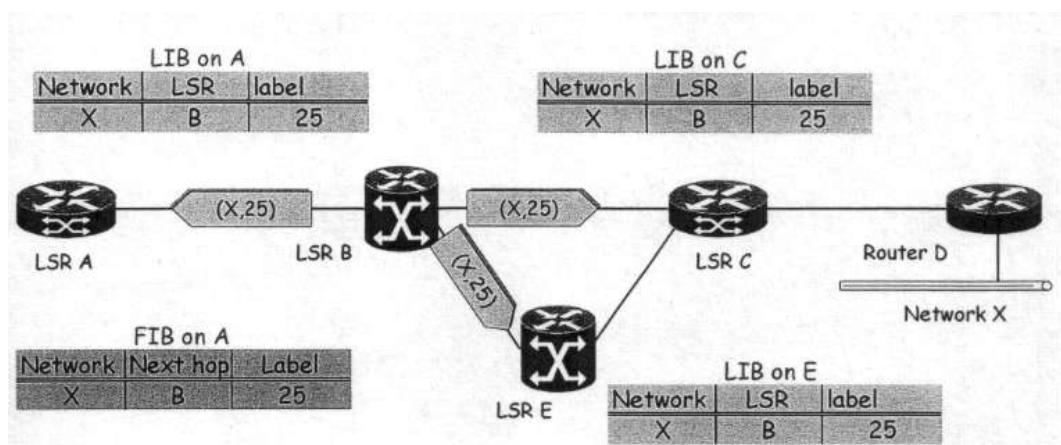
LIB and LFIB structures have to be initialized on the LSR allocating the label.

#### Label Distribution:



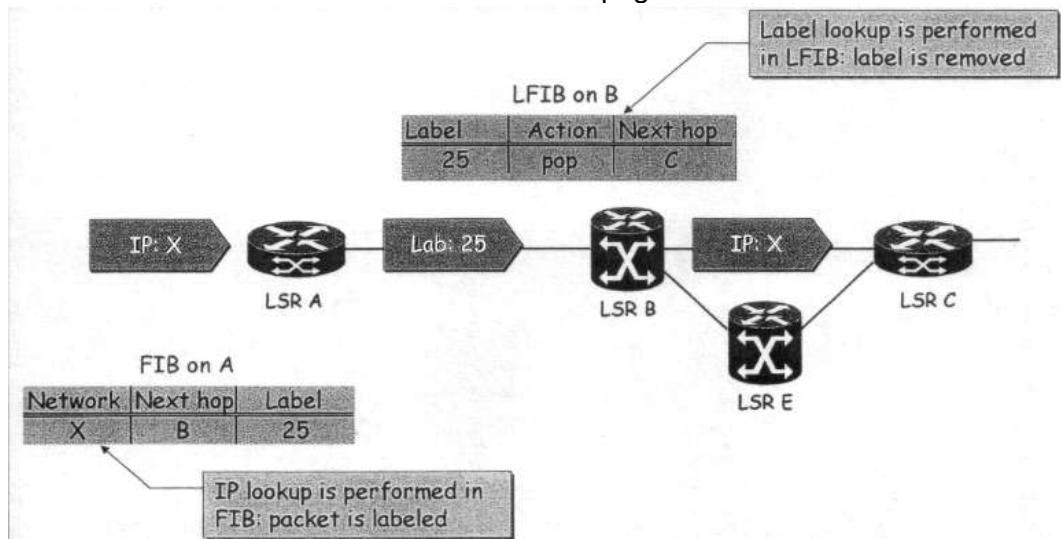
The allocated label is advertised to all neighbor LSRs, regardless of whether the neighbors are upstream or downstream LSRs for the destination.

#### Receiving Label Advertising:



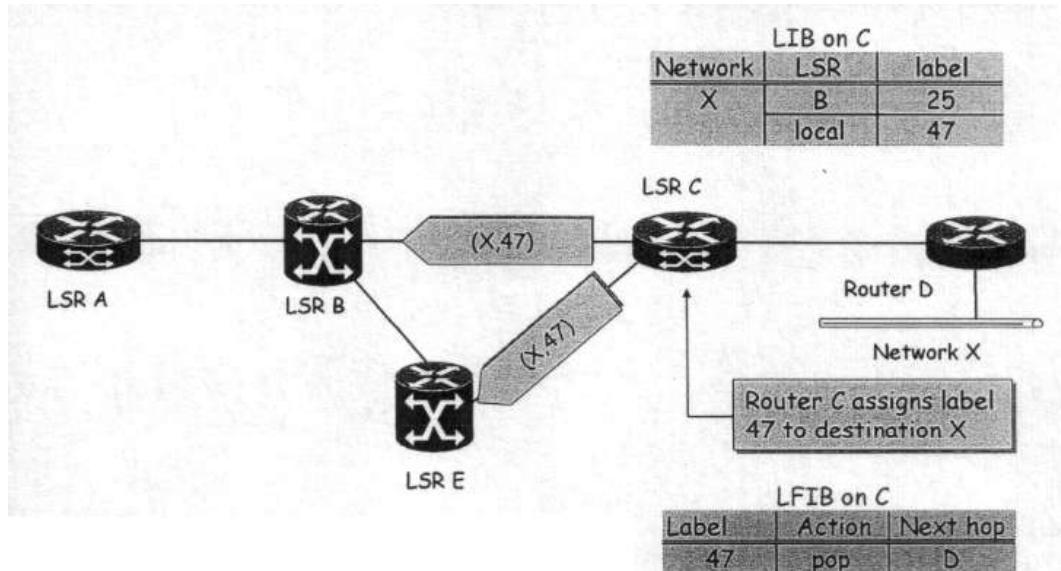
Every LSR stores the received label in its LIB. Edge LSRs that receive the label from their next hop also store the label information in the FIB.

#### Interim Packet Propagation:



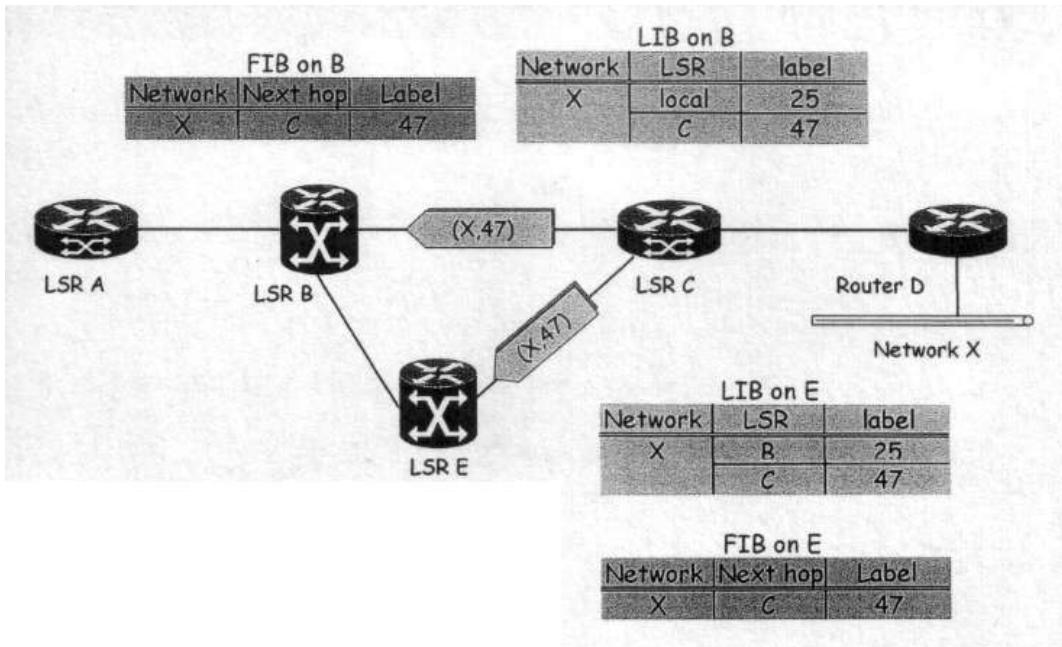
Forwarded IP packets are labeled only on the path segments where the labels have already been assigned.

#### Further Label Allocation:



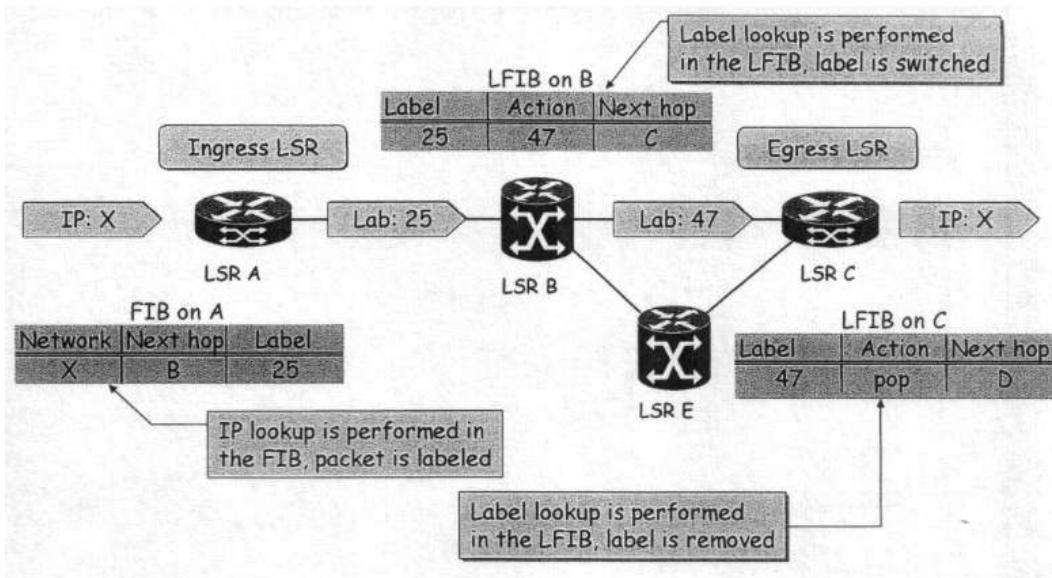
Every LSR will eventually assign a label for every destination.

#### Receiving Label Advertisement:



LSRs B and E store received information in their LIBs and populate their FIBs.

Packet Propagation in MPLS:



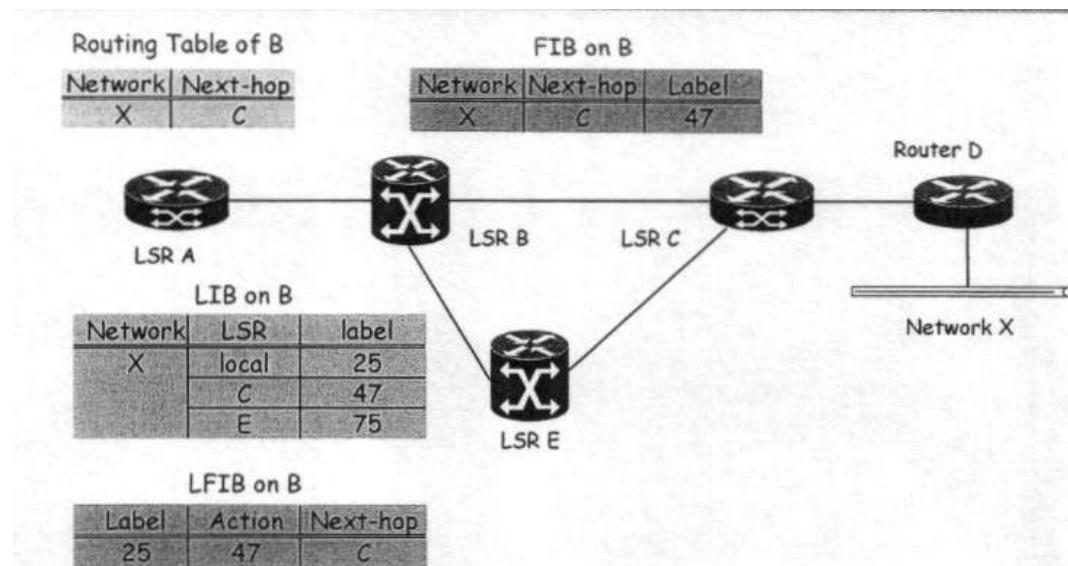
## 5.26 Summary of Packet Allocation and Distribution

Packet-mode MPLS label allocation and distribution follows these rules:

- every LSR assigns a label for every destination in the IP routing table
- labels are assigned once per LSR (per-platform)
- every LSR advertises its label assignments to all neighbors
- every LSR stores all advertised labels in the LIB
- labels received from next hop LSRs are used to populate label information in the FIB and the outgoing label in the LFIB

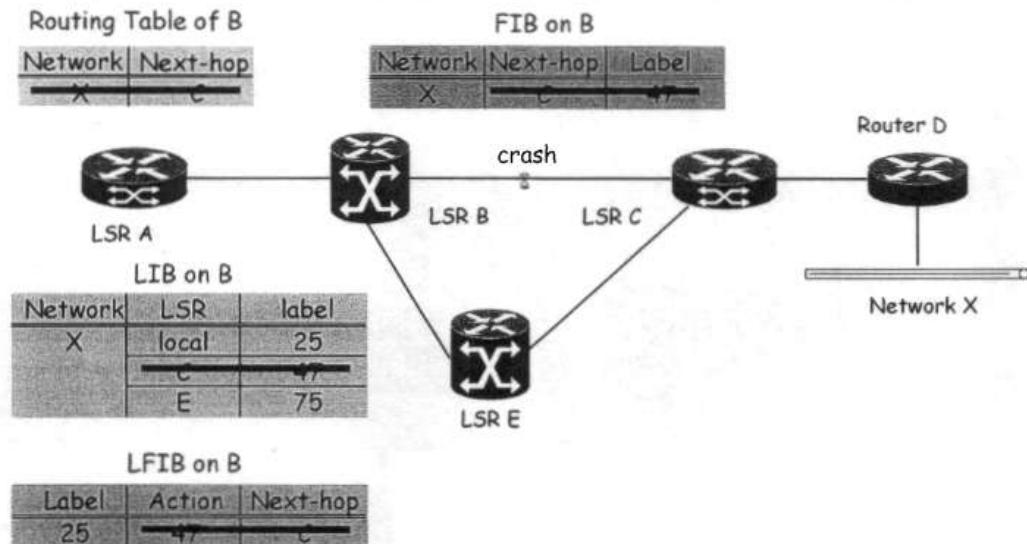
## 5.27 Link Failure

Steady State Description:



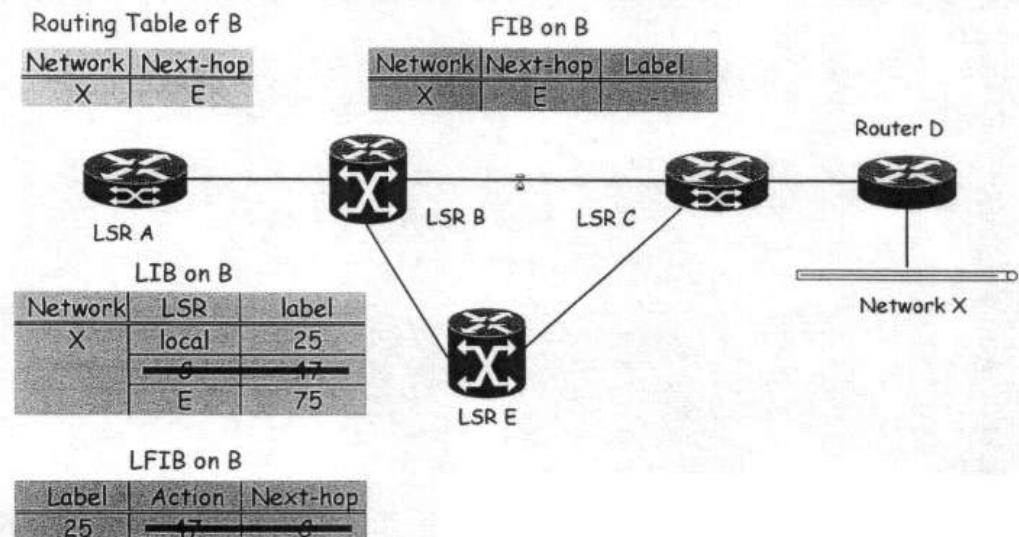
After the LSRs have exchanged the labels, LIB, LFIB and FIB data structures are completely populated.

#### Link Failure Actions:

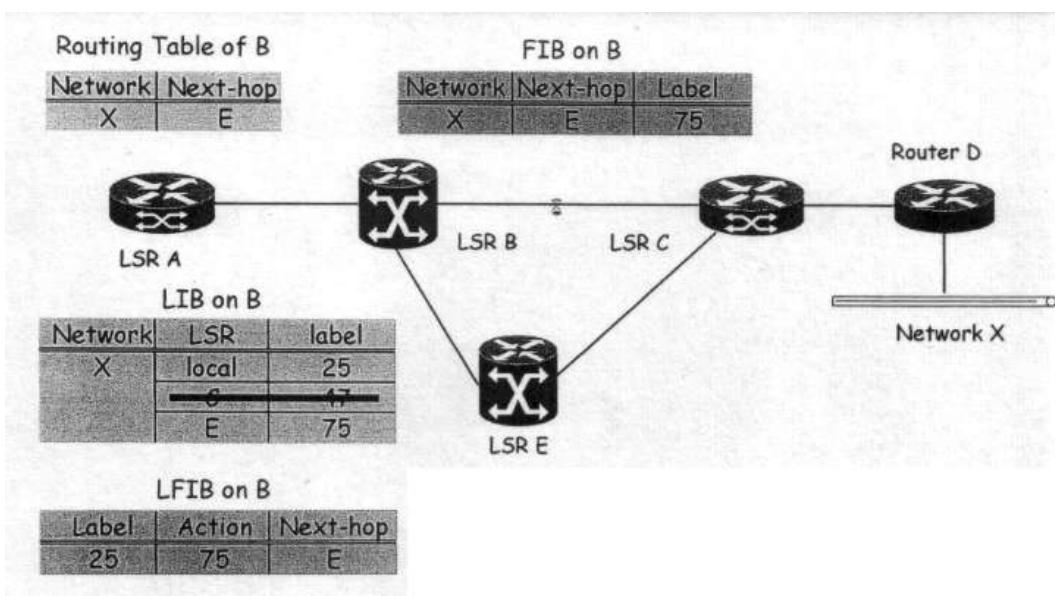


Routing protocol neighbors and LDP neighbors are lost after a link failure. Entries are removed from various data structures.

#### Routing Protocol Convergence:

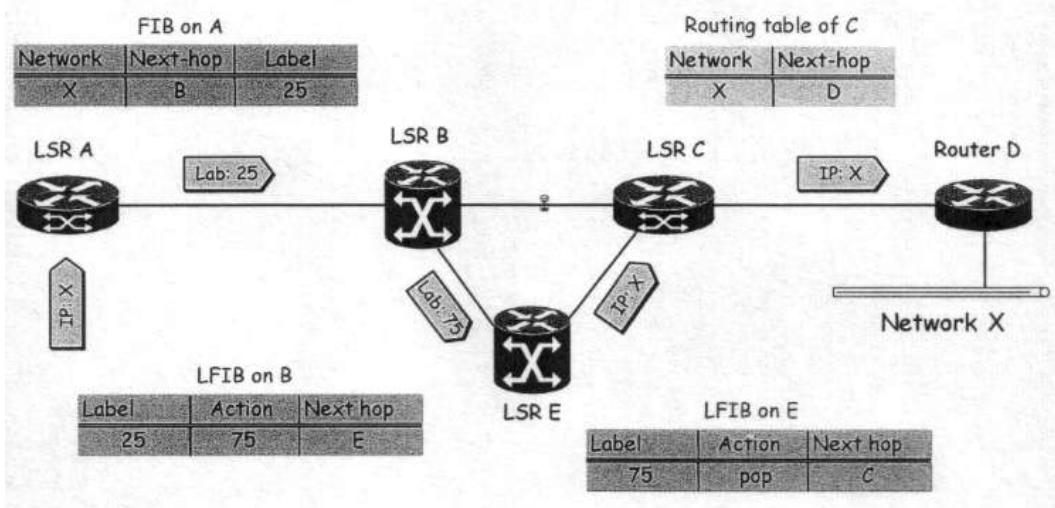


Routing protocols rebuild the IP routing table and the IP forwarding table.

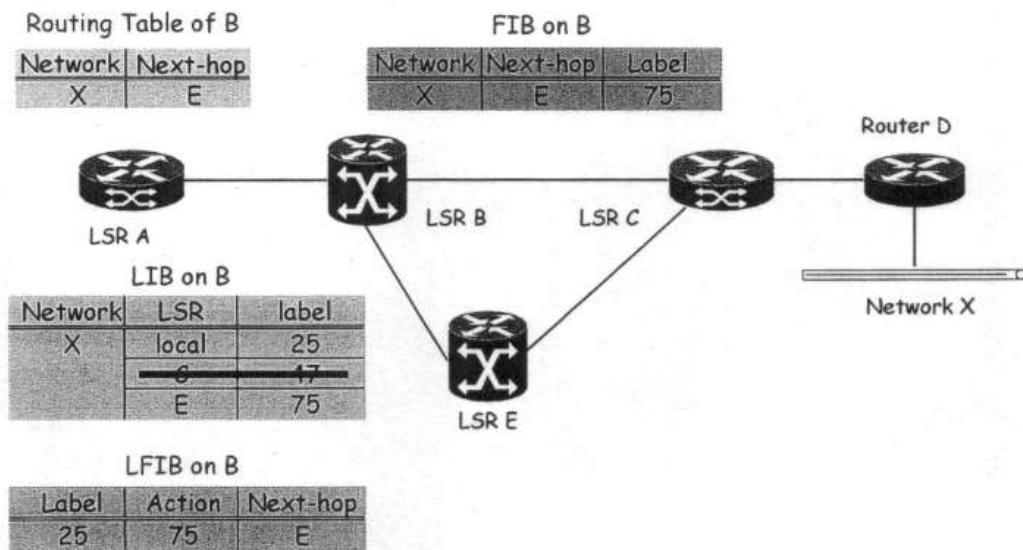


The LFIB and labeling information in the FIB are rebuilt immediately after the routing protocol convergence, based on labels stored in the LIB.

#### MPLS Convergence:

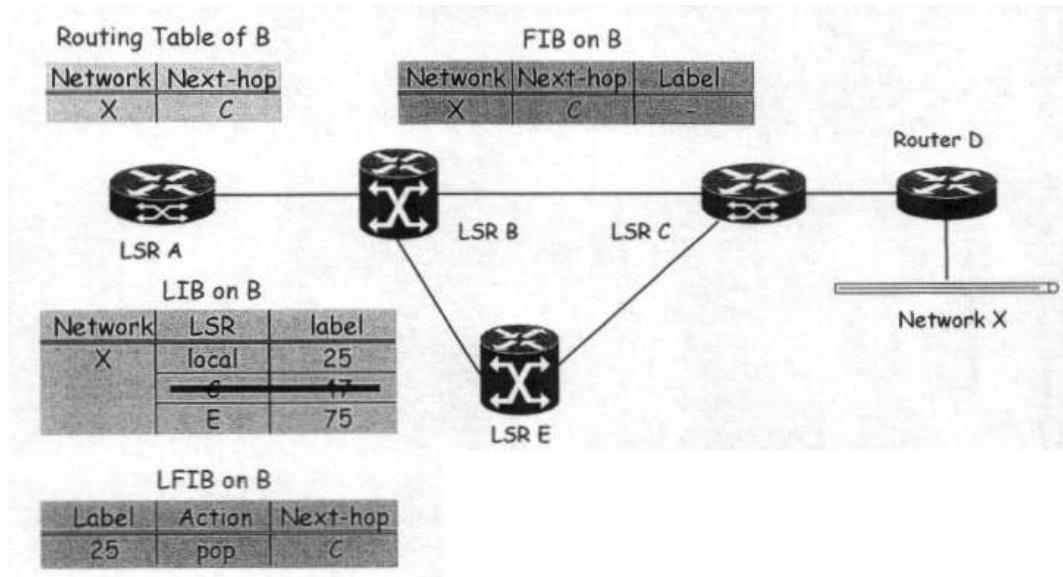


#### Link Recovery Actions:



Routing protocol neighbors are discovered after link recovery.

#### IP Routing Convergence After Link Recovery:

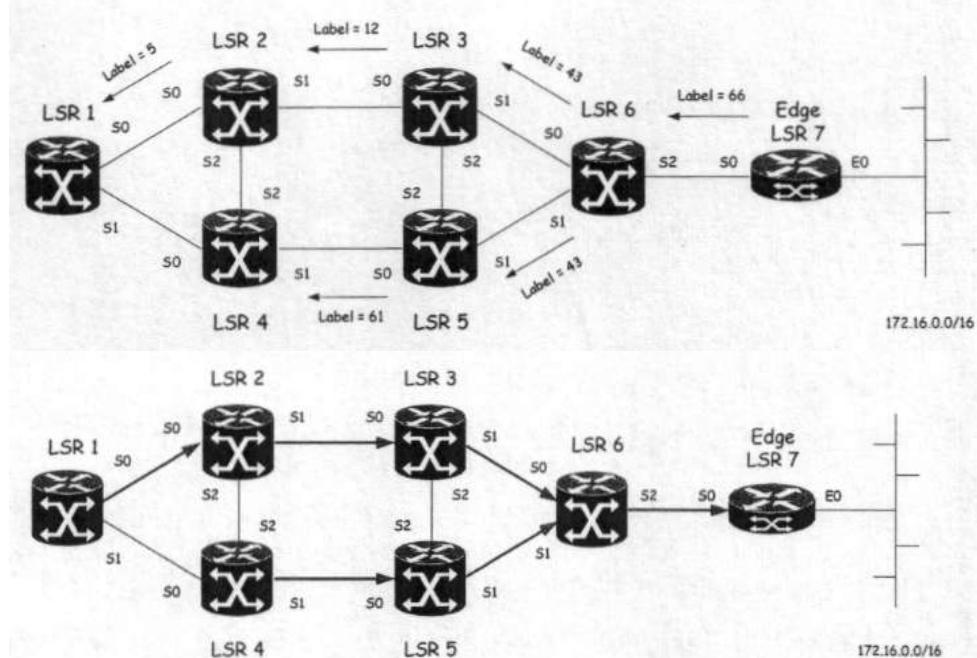


IP routing protocols rebuild the IP routing table. The FIB and the LFIB are also rebuilt, but the label information might be lacking.

## 5.28 Ordered LSP Control

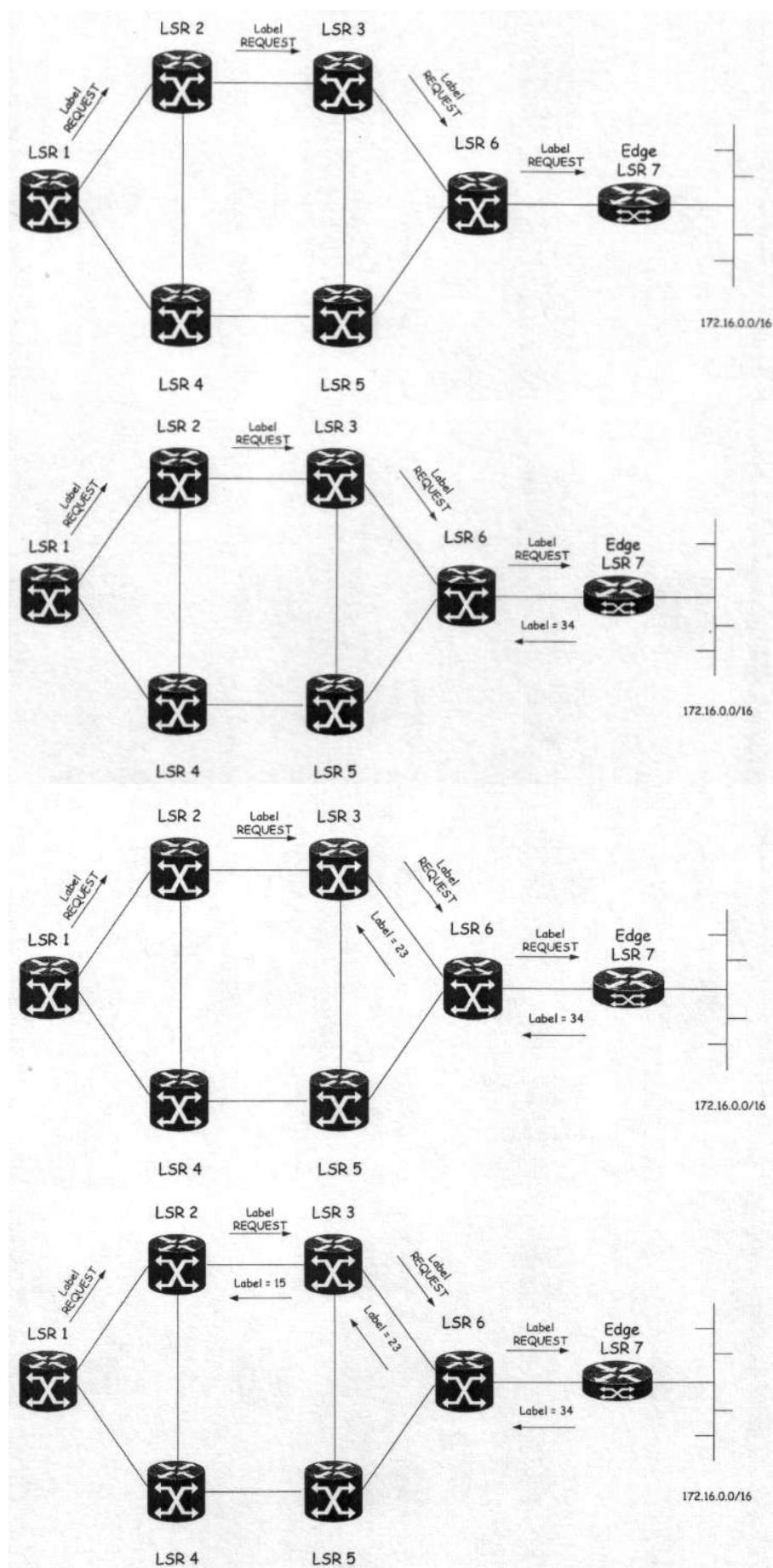
Ordered LSP Control can be with *Unsolicited Downstream* or with *Downstream-on-Demand*. Ordered LSP Control with *Unsolicited Downstream* is initiated by the egress. An LSR knows that it is an egress for a given FEC if its next hop for that FEC is not an LSR. The egress LSR assigns a label to the FEC and advertises the assignment to its neighbor LSRs. Any neighbor that believes the egress LSR is the next hop for that FEC would then proceed to assign a label for that FEC and advertise the assignment to its neighbors, and so on. The assignment of labels would proceed in an orderly fashion from egress to ingress.

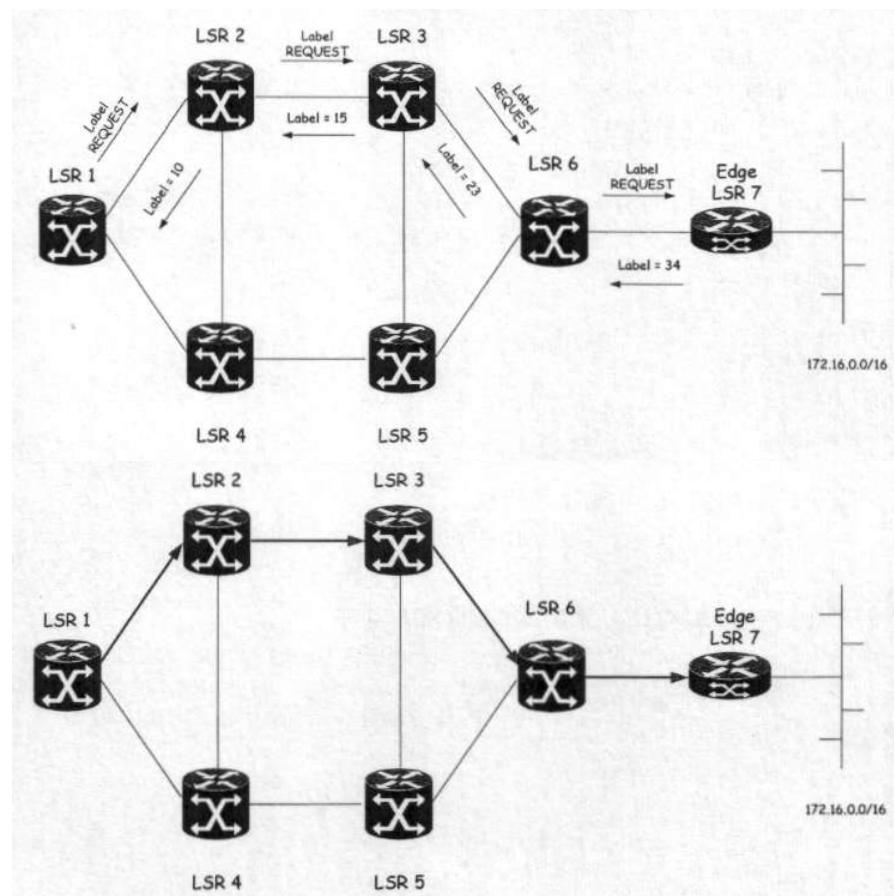
Example:



Ordered LSP Control with *Downstream-on-Demand* is initiated by the ingress. In Ordered LSP Control, an LSR only binds a label to a particular FEC if it is the egress LSR for that FEC, or if it has already received a label binding for that FEC from its next hop (for that FEC).

Example:

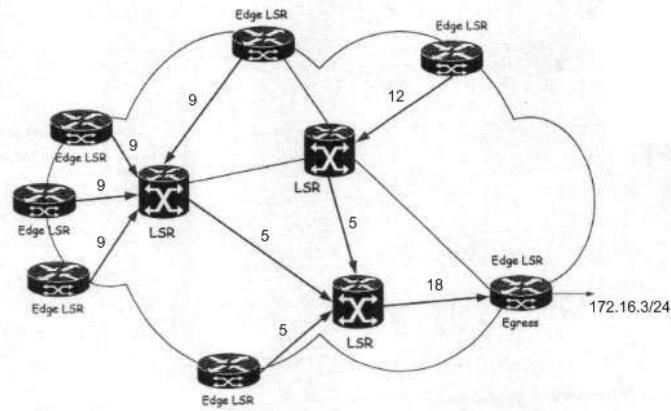




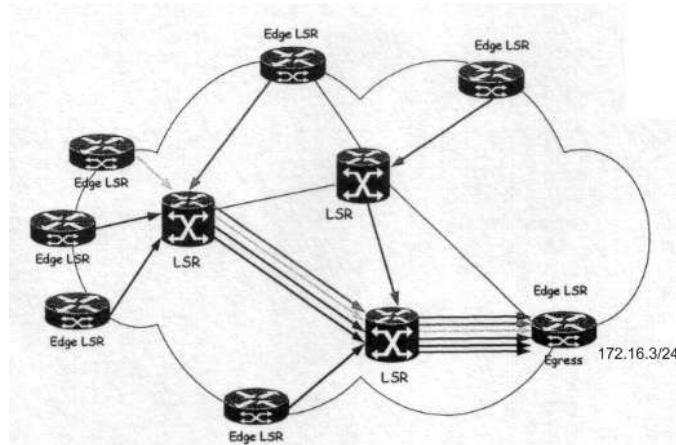
**Questions:** What are the respective merits of independent and ordered control? Are both required? The MPLS architecture eventually settled on allowing both, because each offers its own set of advantages and drawbacks. Ordered approach merit is that it helps to provide loop prevention capabilities. Its disadvantages is an increase in the amount of time it takes to set up an LSP since the ordered approach requires that bindings propagate across an entire region of LSRs before a label switched path is established.

## 5.29 Aggregation

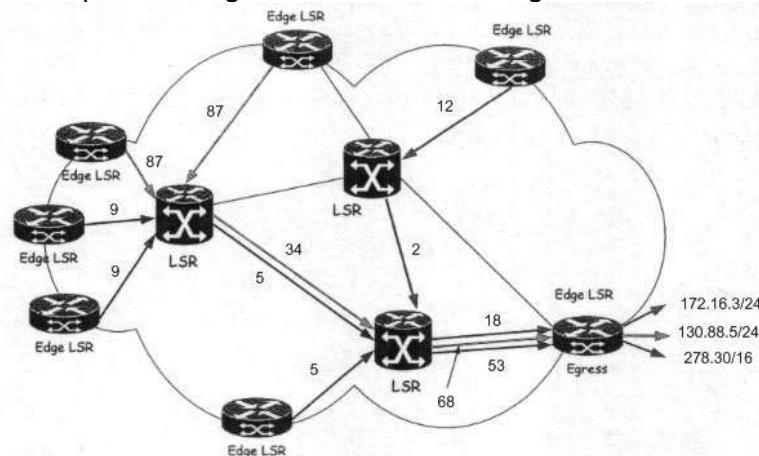
With Ordered LSP Control with Unsolicited Downstream, we end up with the following situation for the FEC 172.16.3/24.



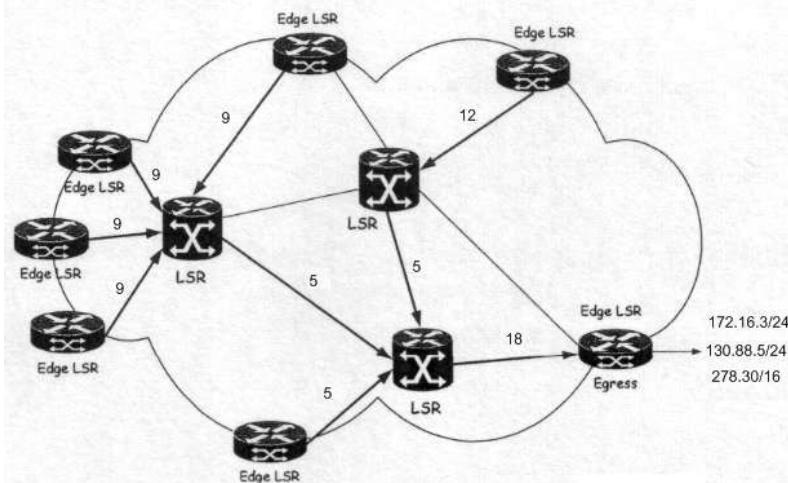
The set of LSPs are forming a reverse tree whose root is the LSP egress. This is also called a multipoint-to-point tree. If we had used Ordered LSP Control with Downstream-on-demand, the situation would have required more labels (in the example of the figure, the egress LSR would have allocated 6 labels instead of 1).



A set of distinct address prefixes might all have the same egress node.



One way of partitioning traffic into FECs is to create a separate FEC for each address prefix which appears in the routing table. Aggregation is the procedure of binding a single label to a union of FECs which is itself a FEC (within some domain), and of applying that label to all traffic in the union. The MPLS architecture allows aggregation which reduces the number of labels which are needed and may also reduce the amount of label distribution control traffic needed. It is possible to aggregate all the previous reverse trees having their roots at the egress LSR into a single reverse tree in such a way that every ingress router will have a single label to use for all the packets which exit the MPLS domain by the egress LSR. This is the ultimate possible aggregation.



With aggregation, within the MPLS domain, the union of those FECs is itself a FEC. This creates a choice: should a distinct label be bound to each component FEC or should a single label be bound to the union, and that label applied to all traffic in the union?

Given a set of FECs which are aggregatable into a single FEC, it is possible to

- aggregate them into a single FEC;
- aggregate them into a set of FECs or
- not aggregate them at all

Thus we can speak of the granularity of aggregation, with (a) being the coarsest granularity and (c) being the finest. Coarse forwarding granularity is essential for making the overall system scalable which however results fairly inflexible, as it wouldn't allow differentiation among different

types of traffic. Example: coarse forwarding would not allow different forwarding or resource reservations for traffic that belongs to different applications. A routing system that is both scalable and functionally rich would require the system to support a wide spectrum of forwarding granularities, as well as the ability to flexibly intermix and combine different forwarding granularities.

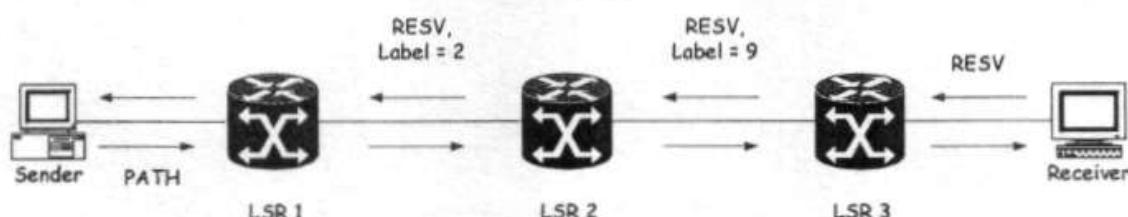
### 5.30 Distributing Label Binding Information

A *Label Distribution Protocol* (LDP - RFC 3036) is a set of procedures and messages by which one LSR informs another of the label/FEC bindings it has made. Two LSRs which use a label distribution protocol to exchange label/FEC binding information are known as label distribution peers with respect to the binding information they exchange. If two LSRs are label distribution peers, we will speak of there being a label distribution adjacency between them. The LDP also encompasses any negotiations in which two label distribution peers need to engage in order to learn of each other's MPLS capabilities. The MPLS architecture does not assume that there is only a single LDP. Existing protocols are being extended so that label distribution can be piggybacked on top of them (e.g. MPLS-RSVP, MPLS-BGP) and new protocols are being defined for the explicit purpose of distributing labels (e.g. MPLS-LDP).

Using RSVP as LDP:

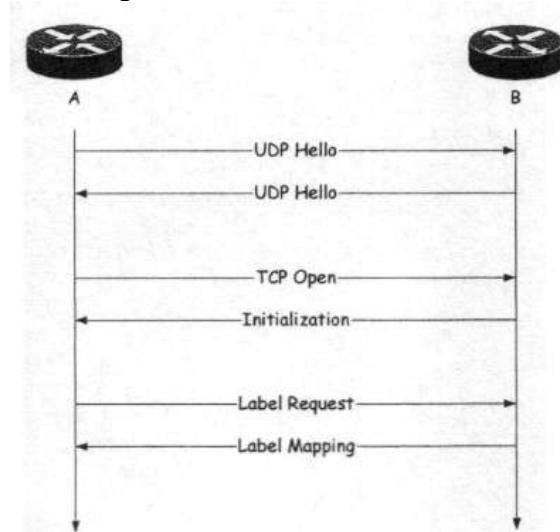


PATH messages flow from sender to receiver and RESV messages follow the reverse path



Label distributed in RESV messages

LDP defines four classes of messages.



- *DISCOVERY* messages (UDP Hello) to announce and maintain the presence of an LSR in a network.
- *ADJACENCY* messages (TCP Open/Initialization) to establish, maintain and terminate sessions between LSR peers.

- *LABEL ADVERTISEMENT messages* (Label Request/Label Mapping), which deal with label binding advertisements, requests, withdrawal, and release.
- *NOTIFICATION messages*, used to provide advisory information and to signal error information.

LDP has four major functions:

1. Neighbor Discovery
2. Session Establishment and Maintenance
3. Label Advertisement
4. Notification

### **5.31 Neighbor Discovery**

Like other network protocols, LDP has the concept of neighbors. LDP uses UDP/TCP ports 646 for discovery. LDP runs over TCP to provide reliable delivery of messages. (with the exception of HELLO messages which are carried by UDP). LDP is designed to be easily extensible, using messages specified as collections of TLV (Type, Length, Value) encoded objects. There is the need for reliability: if a label binding or a request for a binding is not successfully delivered, then traffic cannot be label switched and will have to be handled by the control processor or dropped. The order of message delivery is important: a binding advertisement followed by a withdrawal of that binding, for example, will have a very different effect if the messages are received in the reverse order. The issue was whether to use TCP to provide reliable, in-order delivery or to build that functionality into LDP itself.

### **5.32 Session Establishment and Maintenance**

After discovering potential LDP neighbors, LDP session establishment can begin. LDP session establishment is a two-step procedure: determine who plays the active role and who plays the passive role in this establishment and initialize the session parameters. Active or passive roles are determined by comparing the transport address (almost always the LSR-ID of the transmitter) in the HELLO packet. If the receiver determines that he is the one who should be active, he initiates a TCP session. If not, he waits for the sender to initiate it. After the TCP session is established, the LSRs negotiate the session parameters through LDP initialization messages. These include:

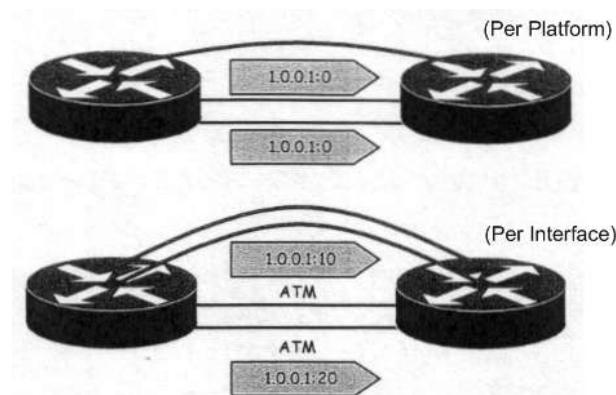
- version
- label distribution method (unsolicited downstream or downstream-on-demand)
- timer values (the hold-down time for the TCP connection)
- VPI/VCI ranges for label-controlled ATM, and so on

After it is established, the session is maintained by sending periodic UDP discovery HELLOS and KEEPALIVE messages within the TCP session.

### **5.33 Label Space Negotiation**

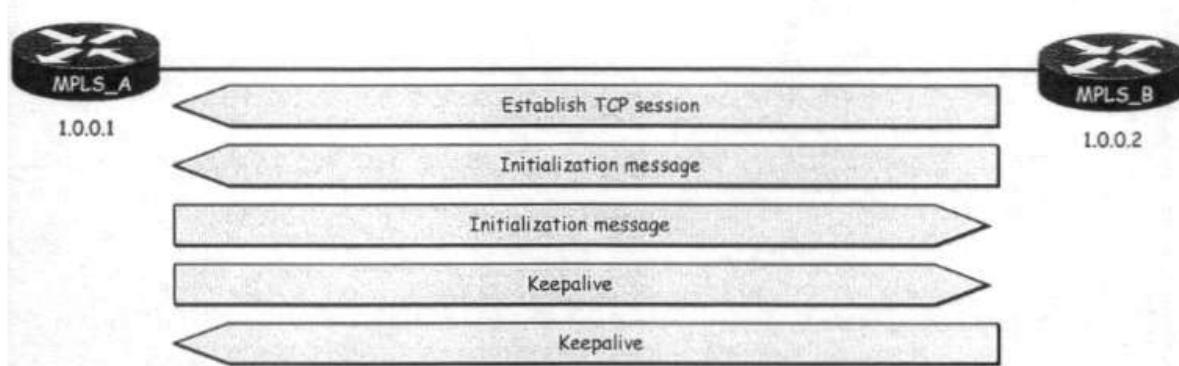
The number of sessions needed between LSRs depends on the label space. LSRs establish one LDP session per label space. For frame-mode interfaces, even though there are multiple links between two LSRs, only one LDP session is established, because for frame-mode operation, a per-platform label space is used. Per-platform label space is announced by setting the label space ID to zero (for example, LDP ID=1.0.0.1:0). When ATM interfaces are used, the LSR has to maintain a per-interface label space for each of them. RFC 3036 requires the use of one neighbor relationship per interface when using per-interface label space. Therefore, a combination of frame-mode and cell-mode, or multiple cell-mode links, results in multiple LDP sessions.

Label Space Negotiation Example:



One LDP session is established for each announced LDP identifier (router ID + label space). The number of LDP sessions is determined by the number of different label spaces.

#### LDP Space Negotiation:



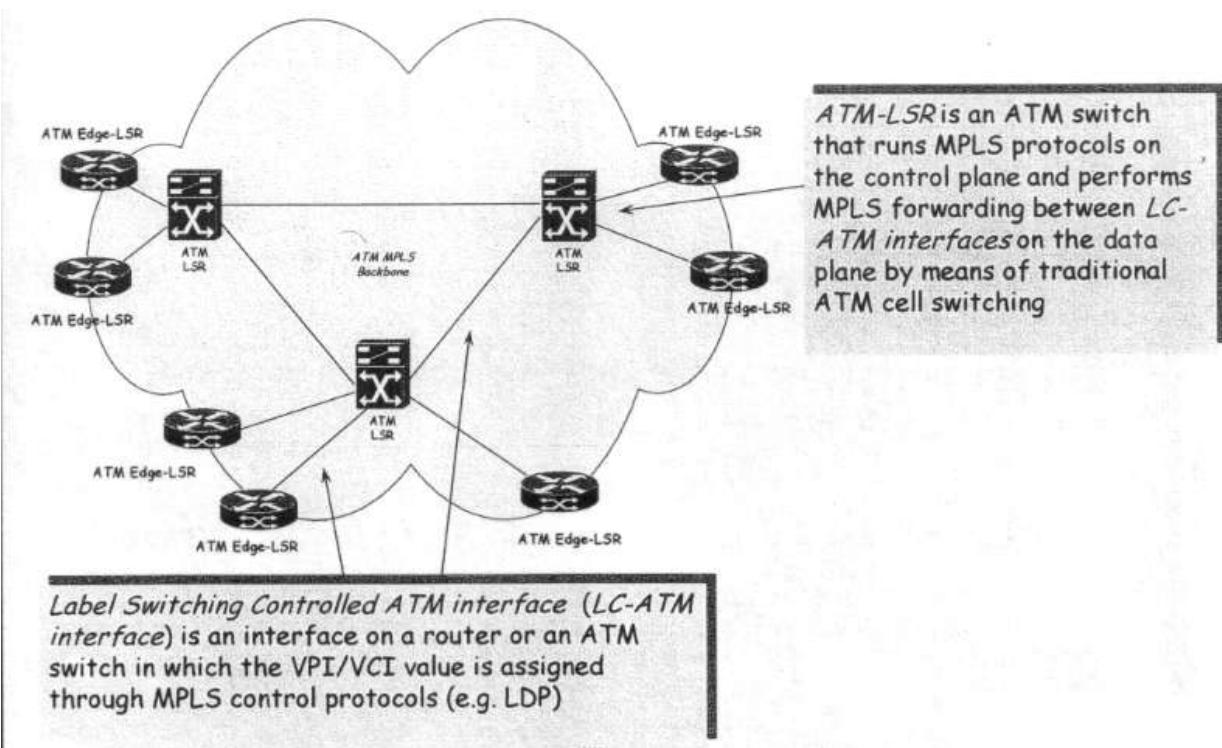
Peers first exchange initialization messages. The session is ready to exchange label mappings after receiving the first keepalive. LDP has both HELLO messages and KEEPALIVE messages. Both are necessary. As mentioned earlier, LDP forms only one neighbor relationship if there are multiple frame-mode links between two LSRs. HELLO messages are sent on each link via UDP multicast and are used to ensure that an LSR knows which interfaces a neighbor is over. KEEPALIVE are sent between established neighbors on a TCP connection, and which interface the keepalives come in on isn't tracked. HELLO messages are per-link keepalives, whereas KEEPALIVE messages are per-neighbor keepalives.

### 5.34 Label Advertisement

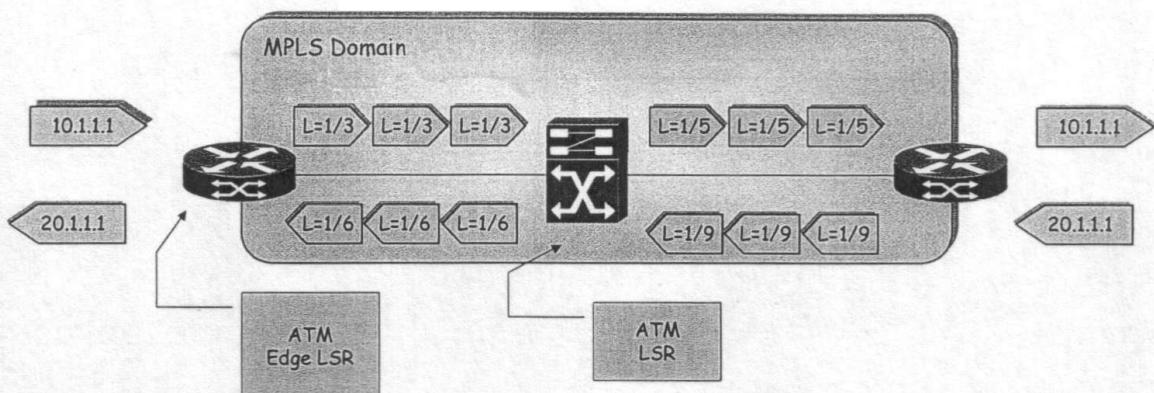
As soon as LSRs have established an LDP neighbor relationship, they begin to advertise labels to each other. Seven different LDP message types are involved in label advertisement, among which *Label Request* and *Label Mapping*. Label Request is used only with downstream-on-demand (DoD) label distribution mode and allows an LSR to explicitly request, from its next hop for a particular FEC, a label binding for that FEC. Label Mapping advertises the binding between FECs and labels.

### 5.35 Cell-mode MPLS Operations

#### Cell-mode MPLS Components:



ATM Label Switch Router:

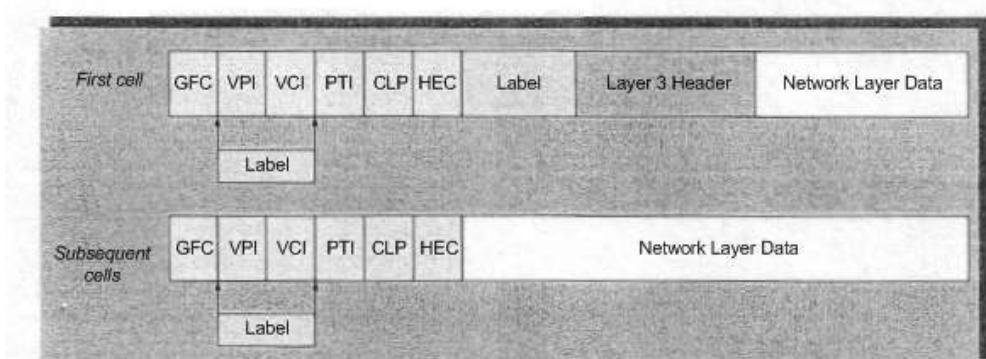


An ATM LSR can only forward cells. An ATM Edge LSR segments packets into cells and forwards them into an MPLS ATM domain, or reassembles cells into packets and forwards them out of an MPLS ATM domain. An MPLS label is encoded as the Virtual Path Identifier/Virtual Channel Identifier (VPI/VCI) value in cell-mode MPLS environments. Each VPI/VCI combination represents a virtual circuit in ATM. The number of virtual circuits supported by router and switch hardware is severely limited. Conclusion: labels in cell-mode MPLS are a scarce resource.

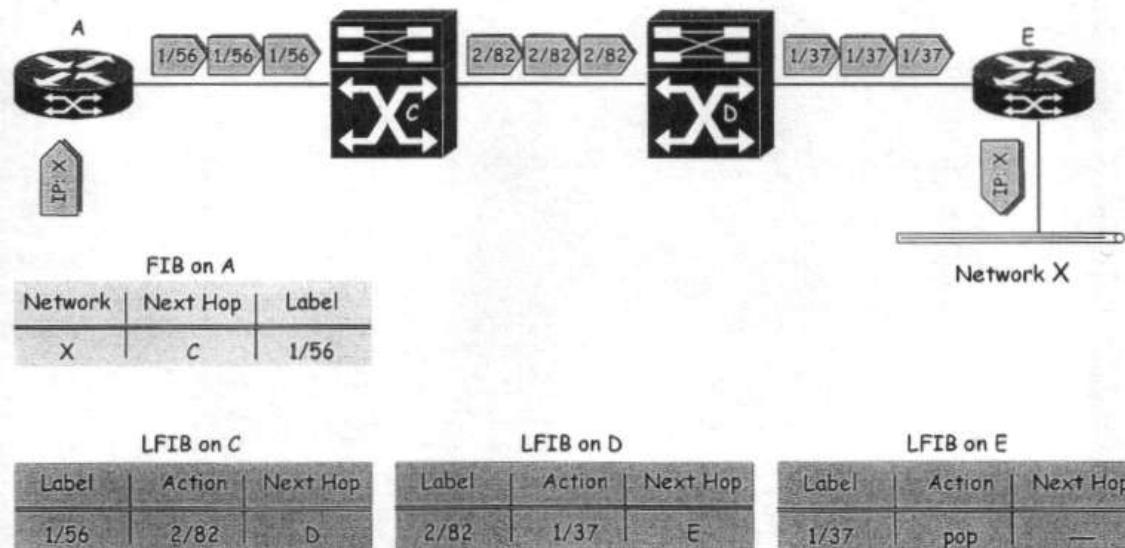
### 5.36 ATM-based MPLS Components

In order to run MPLS, the top label of the label stack is translated into a VPI/VCI value. The label allocation and distribution procedures are modified so that the ATM-LSR looks up the VPI/VCI label value and determines the outgoing interface and outgoing label (*Forwarding Component*).

ATM MPLS Encapsulation Technique (Peer Model):



Example:



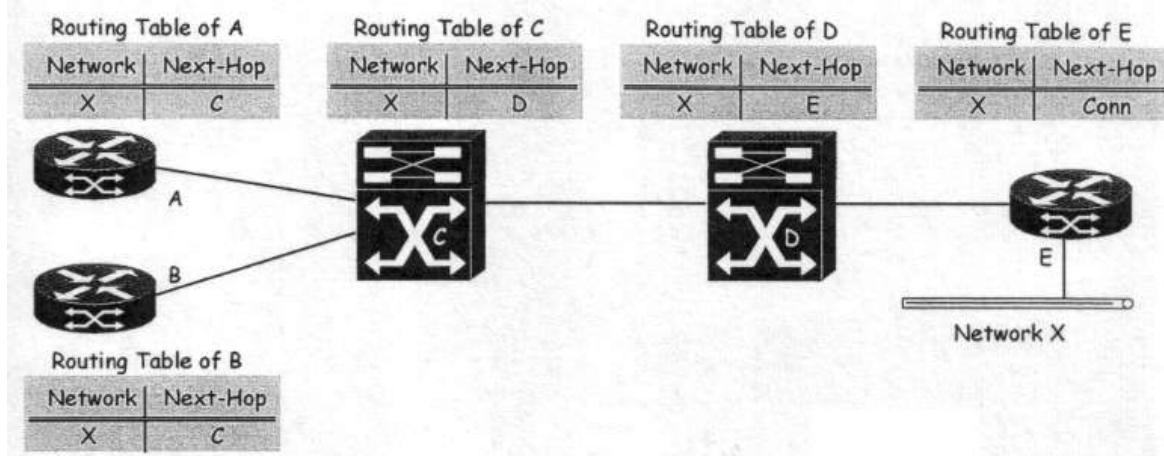
An unlabeled IP packet with a given destination address arrives at (ATM Edge) A. A looks into its FIB and matches the destination with prefix X and a label value of (VPI/VCI) 1/56. The ATM Edge A sends an ATM Adaptation Layer 5 (AAL5) packet as a sequence of cells on VPI/VCI=1/56 to the ATM-LSR C. The ATM-LSR C looks into its LFIB and performs a switching operation by switching cells coming on VPI/VCI=1/56 to VPI/VCI=2/82, .... The ATM Edge E reassembles cells coming on VPI/VCI=1/37 into the original unlabeled IP packet. The *Control Component* of MPLS consists of link-state IP routing protocols, such as the Open Shortest Path First (OSPF), running in conjunction with MPLS label allocation and distribution (e.g. LDP) procedures. Standard Parameter Sets in Cisco IOS Platform MPLS Implementation are:

- for Routers with packet interfaces: *Per-Platform (LSR) Label Space, Unsolicited Distribution, Liberal Label Retention, Independent Control.*
- For ATM switches: *Per-Interface Label Space, On-Demand Distribution, Conservative Label Retention, Ordered Control.*

### 5.37 LSP Set Up Example

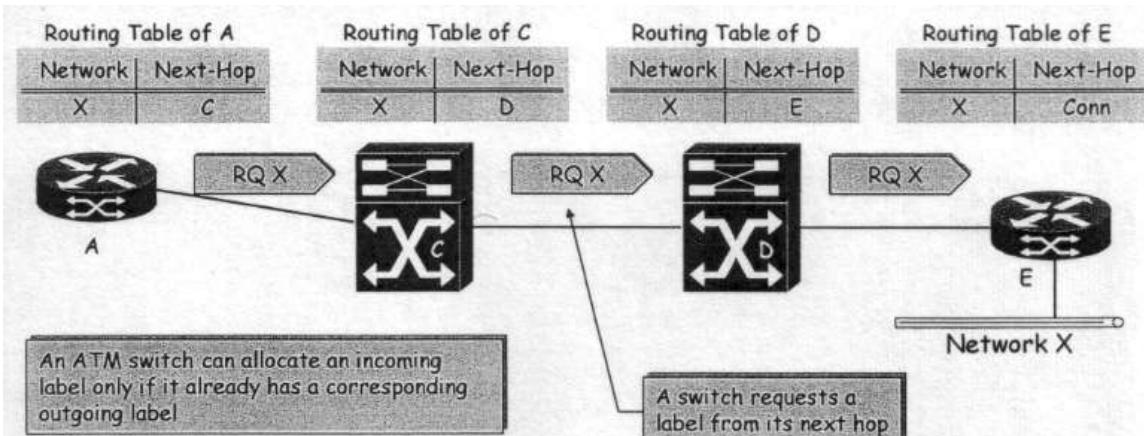
(Ordered Mode Downstream-on-Demand Label Allocation)

Building the IP Routing Table:



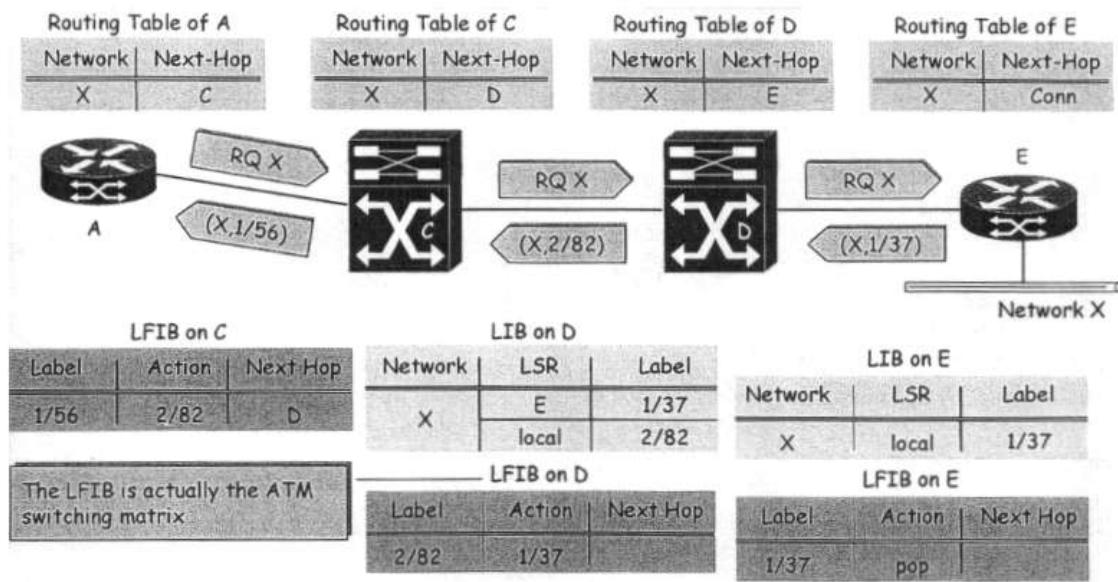
IP routing protocols are used to build IP routing tables on all LSRs. The routing tables are built as if the ATM switches were regular routers.

Ordered Control - Requesting a Label:



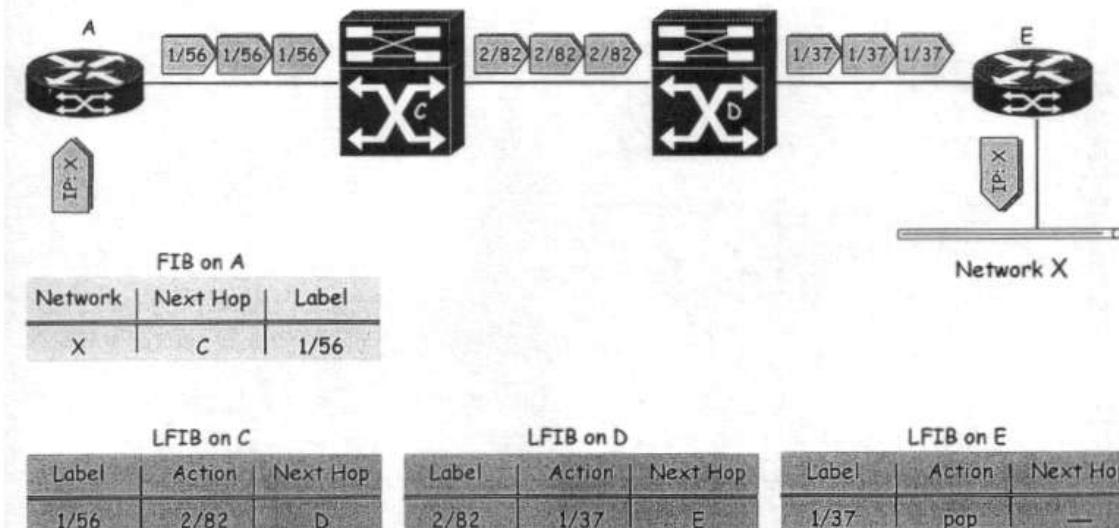
Labels have to be explicitly requested over LC-ATM interfaces. A router requests a label for every destination in the routing table with the next hop reachable over an LC-ATM interface.

#### Ordered Control - Allocation a Label:

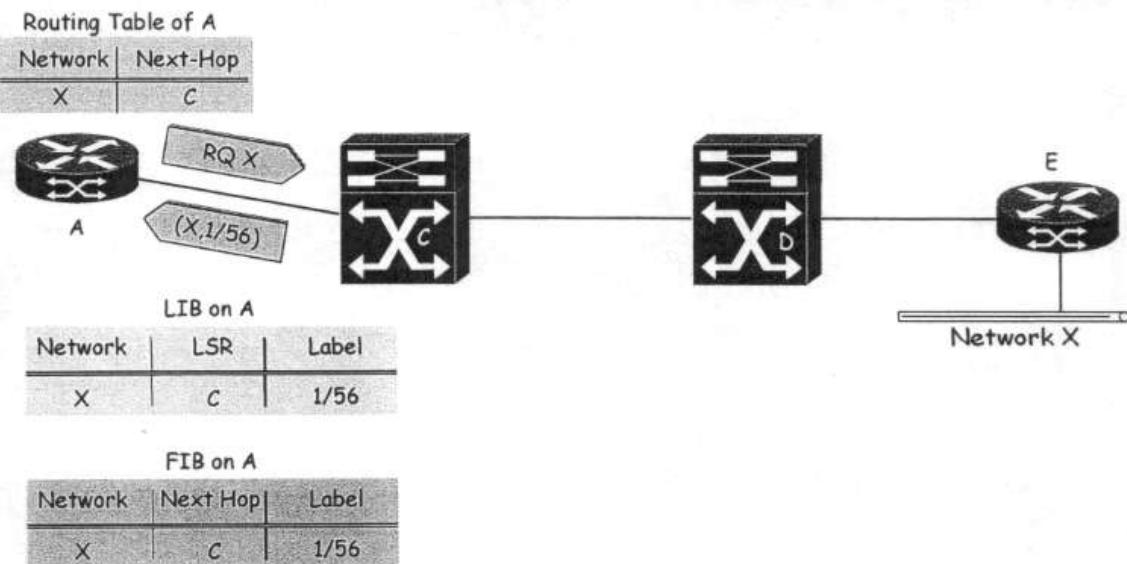


An ATM LSR can allocate an incoming label after receiving an outgoing label. It replies with the allocated label to the incoming request. The egress ATM edge LSR allocates a label and replies to the request.

#### Data Transfer:

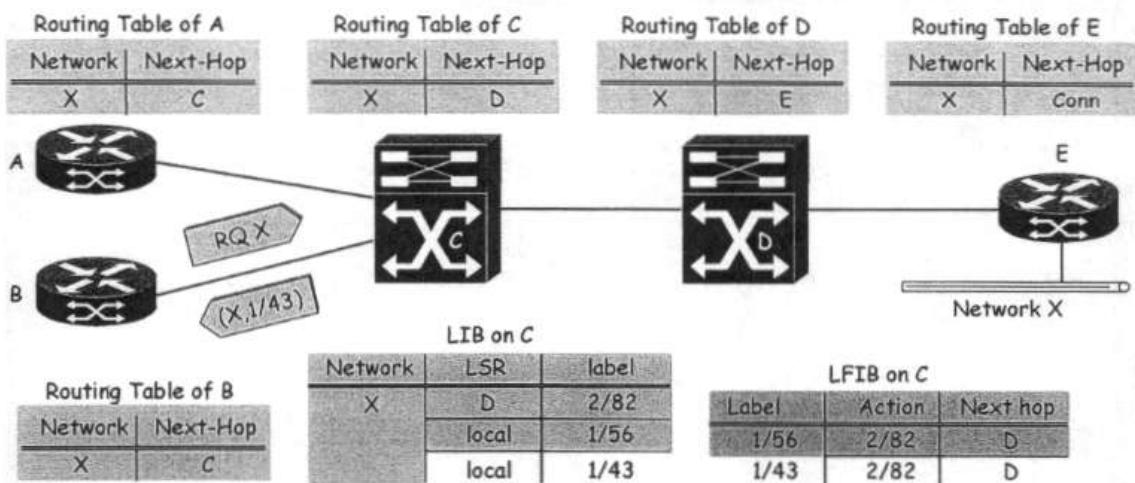


#### Processing Label Allocation Reply:



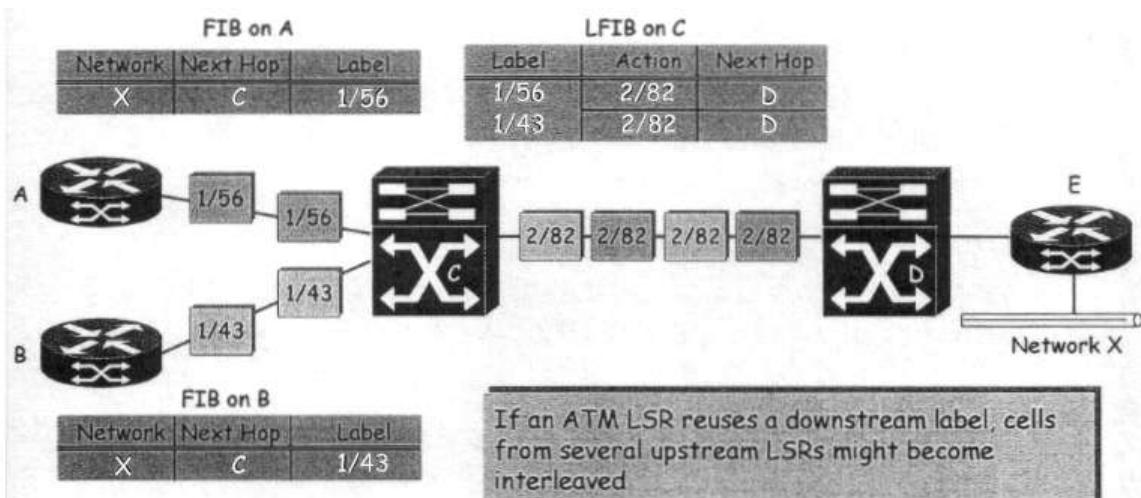
The ingress ATM edge LSR requesting a label inserts the received label in its LIB, FIB, and (optionally) LFIB. In independent mode the LSR that initiated the request receives the binding information, creates an entry in its LFIB, and sets the outgoing label in the entry to the value received from the next hop. The next hop ATM LSR repeats the process, sending a binding request to its next hop. The process continues until all label bindings along the path are allocated.

Allocation Requests from Additional Upstream LSRs:



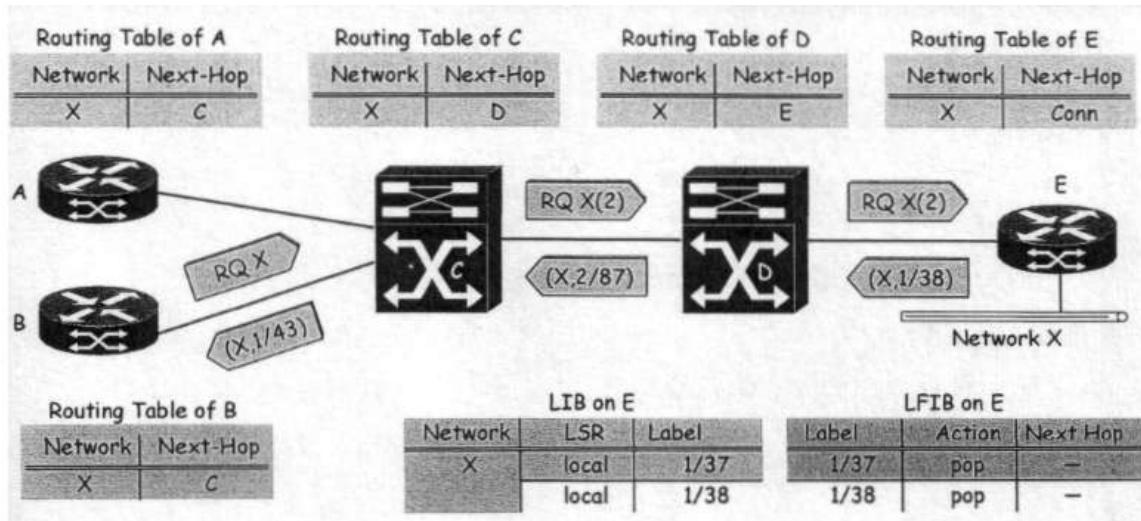
Every upstream LSR will request a label for downstream destinations from an ATM LSR. The ATM LSR could reuse an already allocated downstream label for the second upstream label.

### 5.38 Cell Interleave Issue



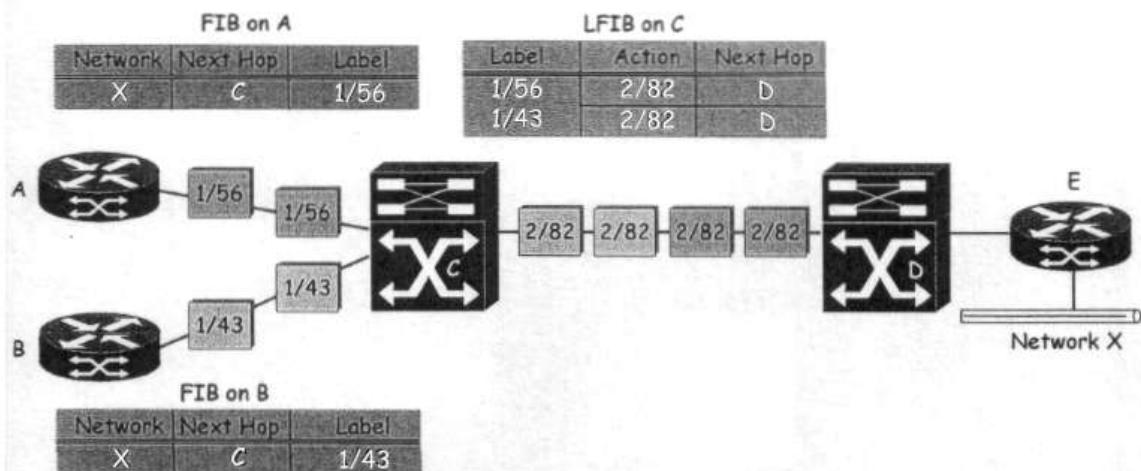
Solution 1: downstream label for every upstream request. Solution 2: prevent cell interleave by blocking incoming cells until a whole frame is collected.

Additional Label Allocation (Solution 1):



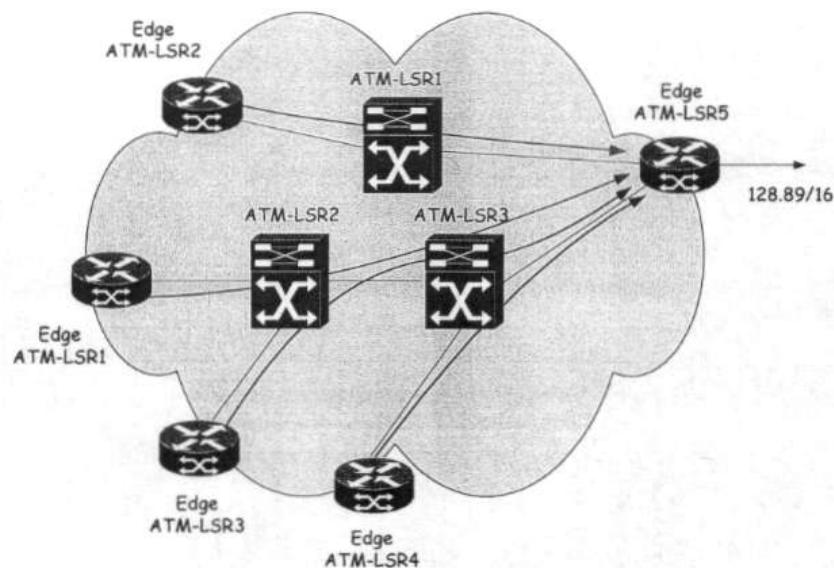
The ATM LSR requests a new label from downstream LSRs for every upstream request. The ATM egress router has to allocate a unique label for every ATM ingress router for every destination.

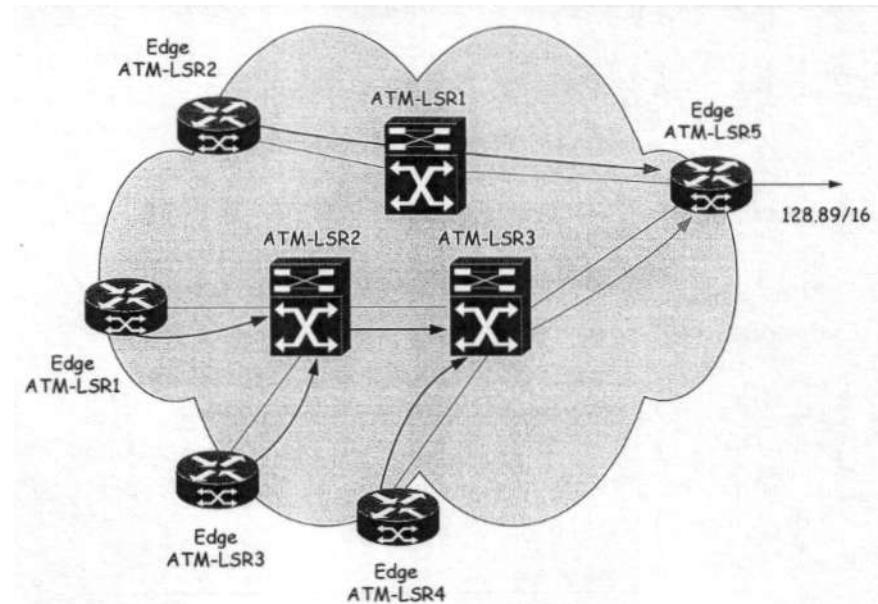
#### Virtual Circuit Merge (Solution 2):



VC merge is a solution in which incoming cells are blocked until the last cell in a frame arrives. All buffered cells are then forwarded to the next hop ATM LSR. Benefit of VC merge is that the Merging ATM LSR can reuse the same downstream label for multiple upstream LSRs. Its drawbacks are that buffering requirements increase on the ATM LSR and that jitter and delay across the ATM network increase.

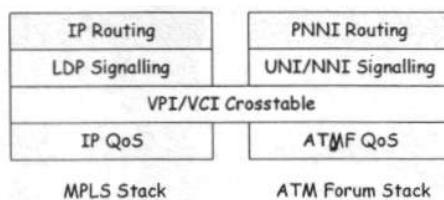
#### Example of No VC Merge:





## **5.39 Control-plane in an ATM Switch**

With the deployment of MPLS in the ATM-LSRs, the central processor of an ATM switch must support MPLS signaling in addition to the traditional ATM Forum signaling protocols. The two sets of protocols run transparently side by side. This mode of operation is sometimes also called the *ships-in-the-night* approach.



*Ships in the Night* will likely be used as a transitioning mechanism as networks migrate their ATM control planes to MPLS. Networks will initially preserve ATM for carrying time sensitive data traffic such as voice and video, and for connecting to non-MPLS enabled nodes, while concurrently running MPLS to carry data. As MPLS Quality of Service features evolve, it is conceivable that there will no longer be a need for separate ATM flows and therefore networks will only carry MPLS label-based traffic.

## **5.40 MPLS Loop Detection and Prevention**

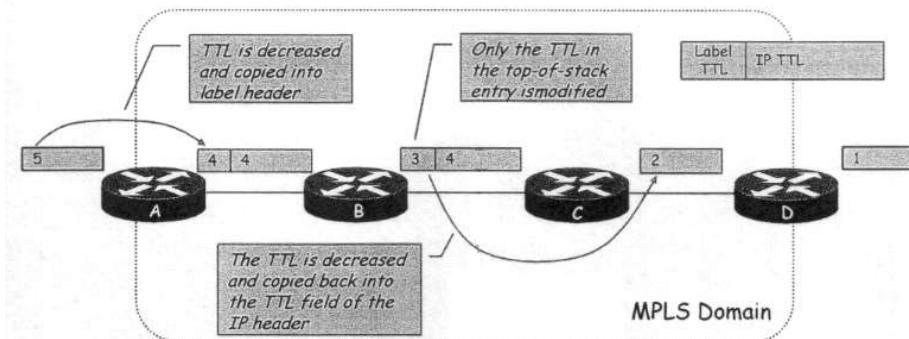
The LDP makes use of information gathered by the Layer 3 routing protocols and therefore is susceptible to routing loops unless the Layer 3 protocol itself can prevent loops from occurring. An important issue to consider when deploying the MPLS architecture is its capability to detect and prevent forwarding loops within the topology. A forwarding loop in an IP network is the process by which a router forwards a packet down the incorrect path (as far as its neighbor is concerned) to a particular destination based on the information contained in its routing table. This can happen during a convergence transition when dynamic routing protocols are used, or through the misconfiguration of the routers so that one router points to another router that is not actually the correct next hop for a particular destination. In terms of the MPLS architecture, you must consider both the control plane and the data plane, and how loop prevention is deployed in both a Frame-mode and Cell-mode backbone. You also must understand how each can detect, and deal with, forwarding loops.

## ***5.41 Loop Detection and Prevention in Frame-mode MPLS***

Labels are assigned to particular FECs using independent control mode when running MPLS across a Frame-mode implementation. When you use this mode, labels are assigned to FECs based on whether the FEC exists within the routing table of the LSR. Using these label assignments, you can establish Label Switched Paths (LSPs) across the MPLS network. Building on this knowledge, you can understand how each LSR can detect, and prevent, forwarding loops. In a standard IP-routed network, forwarding loops can be detected by examining the TTL field of an

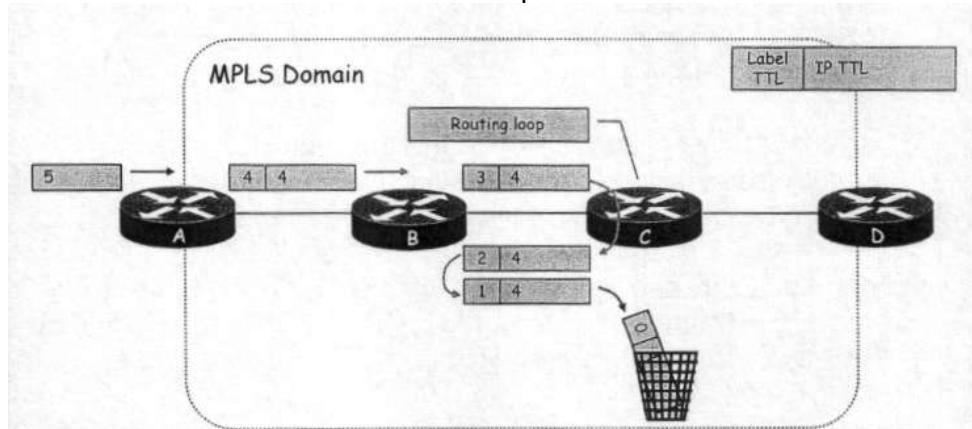
incoming IP packet. Using this field, each router in the packet's path decrements its value by 1; if the field reaches 0, the packet is dropped and the forwarding loop is broken. This same mechanism is used within the *data plane* of a Frame-mode implementation of MPLS (TTL functionality in the label header is equivalent to TTL in the IP headers). Each LSR along a particular LSP decrements the TTL field of the MPLS header whenever it forwards an incoming MPLS frame, and drops any packets that reach a 0 TTL.

Normal TTL Operation:



On ingress, TTL is copied from IP header to label header. On egress, TTL is copied from label header to IP header.

Data Plane Loop Detection:



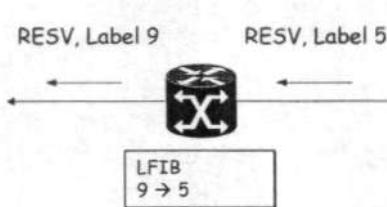
Labeled packets are dropped when the TTL is decremented to zero. The detection of forwarding loops is obviously a very necessary function. However, it also is necessary that the LSR be capable of preventing these forwarding loops before they occur. This prevention activity must be achieved within the *control plane* because this is where LSPs are created. In a standard IP-routed network, the prevention of forwarding loops is the job of the interior routing protocol. Because each LSR in a Frame-mode implementation of MPLS uses these same routing protocols to populate its routing table, the information that is used to form the LSPs within the network is the same as with a standard IP-routed network. For this reason, a Frame-mode implementation of MPLS relies on the routing protocols to make sure the information contained in the routing table of the LSR is loop-free, in exactly the same way as a standard IP-routed network.

## 5.42 MPLS Support of RSVP (IntServ)

In the following we examine in detail how RSVP is supported in an MPLS network. Specifically we focus only on the role of RSVP in supporting QoS; its role in traffic engineering is discussed later on. The first goal in adding RSVP support to MPLS is to enable LSRs - which classify packets by examining labels, not IP headers - to recognize packets that belong to flows for which reservations have been made. In other words, we need to create and distribute bindings between flows and labels, for those flows that have RSVP reservations. We can think of the set of packets for which an RSVP reservation has been made as just another instance of an FEC. It turns out to be rather easy to bind labels to reserved flows in RSVP, at least in the unicast case. A new RSVP object is defined: the *LABEL object* and is carried inside an RSVP RESV message. MPLS defines also a new RSVP object that may be carried in a PATH message: the *LABEL REQUEST object*. When an LSR wants to send a RESV message for a new RSVP flow, the LSR:

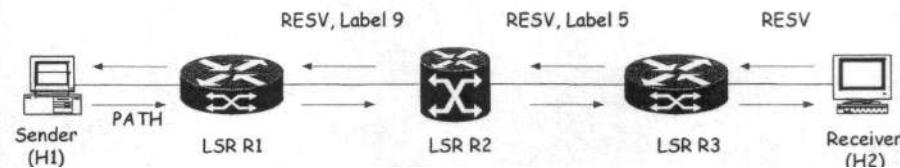
1. allocates a label from its pool of free labels;
2. creates an entry in its LFIB with the incoming label set to the allocated label, and
3. sends out the RESV message containing this label in the LABEL object.

Since RESV messages flow from receiver to sender, this is a form of downstream label allocation.



Upon receipt of a RESV that contains a LABEL object, an LSR populates its LFIB with this label as the outgoing label. It then allocates a new label to use as the incoming label and inserts that in the RESV message before sending it upstream. It should be clear that, as RESV messages propagate upstream, an LSP is established all along the path. Also note that, since the labels are provided in the RESV messages, each LSR can easily associate the appropriate QoS resources with the LSP.

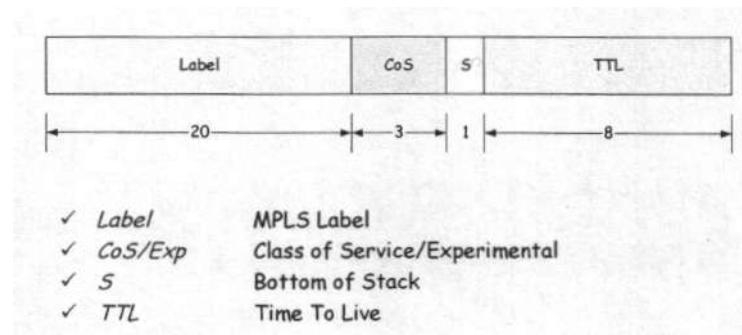
Label Distributed in RESV Messages:



We assume that the hosts do not participate in label distribution (although there is nothing to prevent them from doing so). LSR R3 allocates a label (5) for this reservation and advertises it upstream to its neighbor R2. R2 allocates a label (9) for the same reservation and advertises it to R1. There is now an LSP for this reserved flow from R1 to R3. When packets matching this reservation (i.e., packets sent from H1 to H2 with the appropriate source and destination port numbers and transport protocol number) arrive at R1, R1 classifies them using the IP and transport layer header information and performs the appropriate QoS actions for this reservation, such as policing and scheduling the packets in the output queue. In other words, it does all the normal functions of an Integrated Services router running RSVP. In addition, R1 prepends a label header to the packets and inserts the value 9 as the outgoing label before forwarding the packets to R2. When R2 receives packets bearing the label 9, it is able to look up that label in its LFIB and find all the QoS related state that tells it how to police the flow, queue the packet, and so on (i.e., it does not of course need to examine the IP or transport layer headers). It then replaces the label on the packet with the outgoing entry from its LFIB - 5 - and forwards the packet. Observe that, since the creation of label bindings is driven by RSVP messages, binding is control-driven, just as in other MPLS environments. Also note that this is an example of piggybacking the label binding information on top of an existing protocol, and it does not require a separate protocol such as LDP. One interesting consequence of setting up an LSP for a flow with an RSVP reservation is that only the first router in the LSP - R1 in the example above - needs to be concerned with which packets belong to the reserved flow. This enables RSVP to be applied in MPLS environments in a way that is not possible in conventional IP networks. Conventionally, RSVP reservations could be made only for individual "microflows", that is, flows identified by the five header fields described above. However, it is quite possible to configure R1 to select packets based on a wide range of criteria. For example, R1 could take all packets destined for a particular prefix and put them on the LSP. Thus, rather than having one LSP for each microflow, a single LSP can provide a QoS guarantee for a large aggregate of traffic. An obvious application of this capability would be to provide a guaranteed bandwidth "pipe" from one site of a large company to another site, emulating a leased line between those sites. This capability is also useful for traffic engineering purposes, where large aggregates of traffic need to be sent along LSPs with sufficient bandwidth to carry the traffic.

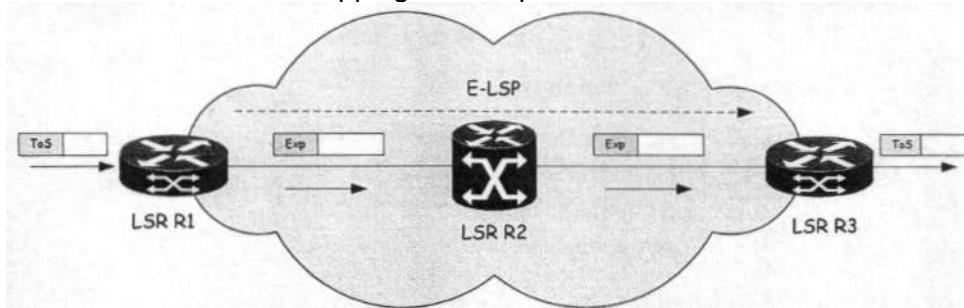
### 5.43 MPLS Support of DiffServ

MPLS LSRs do not examine the contents of the IP header and the value of its DSCP field as required by DiffServ. It follows that the appropriate PHB must be determined from the label value. The PHB can be inferred from the *Exp field* (*E-LSP*) or from the *Label* (*L-LSP*). Note that *Exp* stands for *Experimental*.



## 5.44 MPLS E-LSP

The MPLS shim header has a 3-bit field called Exp which was originally defined to support marking of packets for DiffServ. The Exp field is only 3 bits long, whereas the DiffServ field is 6 bits. Therefore, in a network that supports fewer than eight PHBs, the Exp field is sufficient. Just as a conventional DiffServ router maintains a mapping from the possible values of DSCP to the PHBs it supports, an LSR can maintain a mapping from Exp values to PHBs.

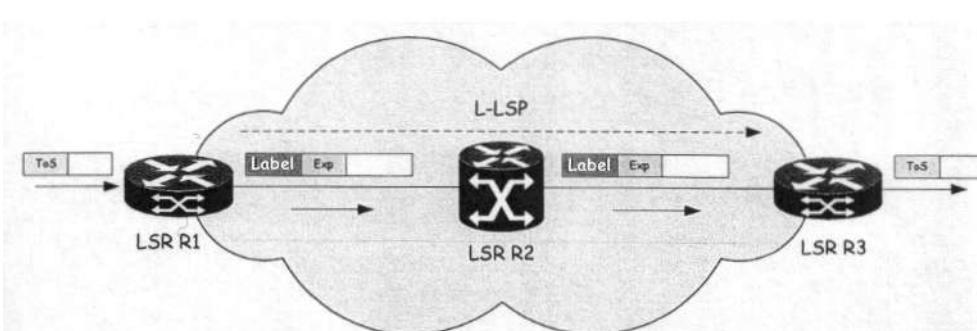


When the Exp field is used to determine a PHB:

- no additional signaling is required;
- any of the label distribution mechanisms (e.g, LDP) can be used without modification to assign labels;
- the label tells an LSR where to forward a packet, and the Exp bits tell it what PHB to treat the packet with.

An LSP of this type is referred to as an *E-LSP*: E stands for Exp, meaning that the PHB is inferred from the Exp bits. Just as it is necessary to configure a DiffServ router so that it correctly maps DSCPs to the appropriate PHBs, it will be necessary to configure an LSR to map the Exp values to different PHBs. For example, in the simple case of a network supporting only two classes of packets (best effort and premium), we could use the Exp value 000 (binary) to mark the best effort packets, and the Exp value 001 (binary) to mark the premium packets. LSRs would then be configured to provide default behavior to the 000 packets and to put packets marked 001 into the queue that implements the EF PHB.

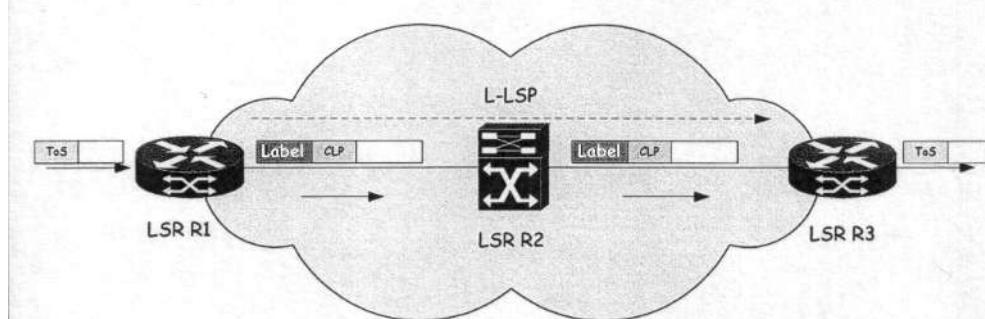
## 5.45 MPLS L-LSP



If more than 8 PHB are needed in the MPLS network, the Exp field alone is not sufficient to indicate the PHB. The obvious choice is to use the label to convey the PHB. In this case we will have *L-LSPs*: L stands for label, since in this case the PHB is inferred from the label. To convey information about PHBs inside labels, the label distribution mechanisms need to be enhanced. It is needed a way that a label can be bound to both an FEC and a PHB (or, perhaps, a set of PHBs).

## 5.46 AF PHB Comments

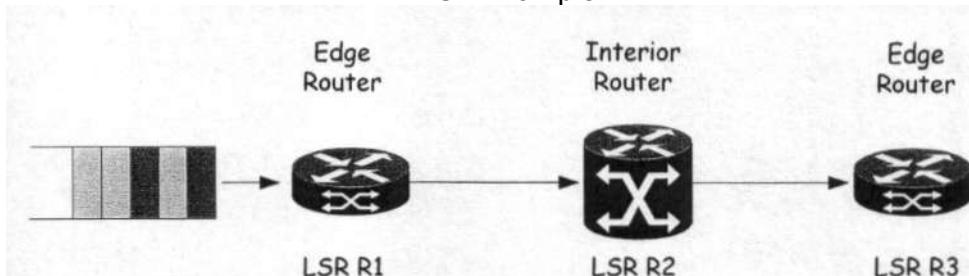
AF specification requires that packets belonging to a single microflow must not be misordered if they differ only in drop preference. In order to meet the ordering constraints for AF traffic, it will typically be necessary to send packets of the same AF class (e.g., AF11, AF12, and AF13 packets) on a common LSP. In the case of DiffServ AF, packets sharing a common PHB can be aggregated into a FEC, which can be assigned to an LSP. This is known as a *PHB scheduling class*.



In the MPLS context, packets of a common PHB scheduling class must travel on the same LSP. The drop preference are encoded in the Exp bits of the shim header. However, in case of ATM, the drop preferences are encoded in the CLP bit. Any existing LDP can be extended to specify which PHB (or PHB scheduling class) is to be bound to the advertised label. In LDP, the messages that request and advertise bindings of prefixes to labels are now extended to include a PHB or PHB scheduling class, i.e. <prefix, PHB> pairs. In most cases an LSP established in this way will carry packets of a single PHB, in which case the Exp bits or CLP bit are unused. For AF traffic, an LSP will carry packets of the same AF class but of a different drop preference, with the Exp bits or CLP bit indicating the drop preference.

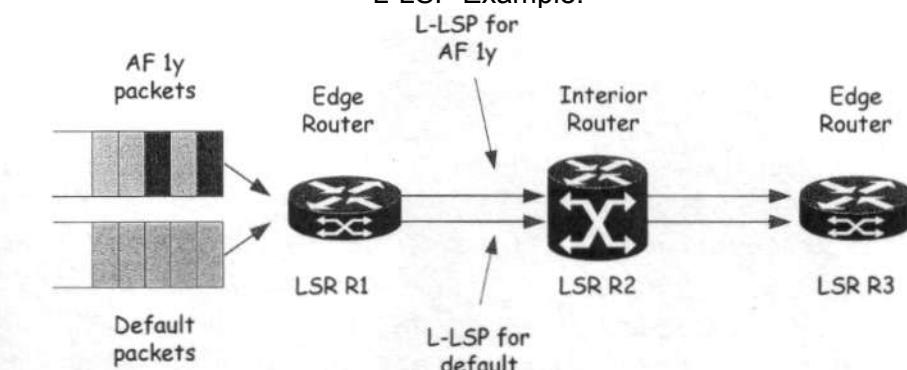
## 5.47 E-LSP and L-LSP Examples

E-LSP Example:



A single E-LSP is in use and may carry packets requiring up to eight different PHBs. The PHB to be applied to any packet is determined by examining the Exp bits.

L-LSP Example:



The lower L-LSP carries only default packets. LSRs R2 and R3 know (since they were told at LSP establishment time) that packets that arrive on the lower LSP must be default packets and should be queued accordingly. The upper L-LSP carries packets that may be AF11, AF12, or AF13. LSRs R2 and R3 use the label to determine that the packets are AF1y packets, and they examine the Exp bits to find the drop preference, that is, the value of y. In these examples, we assume that LSR R1 is the label edge router, that is, it is the router at the start of the LSPs. Thus, the job of getting packets onto the correct LSP, and of setting the Exp bits correctly, falls to R1.

## 5.48 Edge LSR Functions (for DiffServ)

Problem: How does R1 know exactly what to do? As would be the case in a conventional DiffServ network, we assume that R1 has undergone some configuration.

1. LSR R1 can be responsible for applying the local policy that establishes which packets receive which PHB, or
2. LSR R1 may receive IP packets whose DSCP has already been set at some prior hop.

In either case it can be assumed that, before putting a packet onto an LSP, R1 knows the PHB of the packet. The details of setting the Exp bits depend on whether a packet is to travel on an E-LSP or an L-LSP. If it is an E-LSP, LSR R1 must have a configured mapping of PHBs to Exp values, which must be consistent with the reverse mapping (Exp to PHB) in LSR R2 and LSR R3. For an L-LSP, the Exp bits are only relevant for AF PHBs, and the mapping is statically defined: the drop preference is carried in the low order bits of Exp. One point we need to stress is that it is possible to use both types of LSPs in a single network, even on a single link, provided that the link uses the shim header and thus has Exp bits to support E-LSPs. Because L-LSPs require the signaling of a PHB at LSP setup, any LSP that is set up without explicit signaling of a PHB is assumed to be an E-LSP by default.

E-LSPs	L-LSPs
PHB is determined from Exp bits	PHB is determined from label or from label plus Exp/CLP bits
No additional signaling is required	PHB or PHB scheduling group is signaled at LSP setup (in LDP, RSVP, etc.)
Exp → PHB mapping is configured	Label → PHB mapping is signaled Exp/CLP → PHB mapping is well known (used only for AF)
Shim header is required: E-LSP are therefore not possible on ATM links	Shim header is used: therefore, L-LSPs are suitable for ATM links.
Up to eight PHBs per LSP	One PHB per LSP except for AF: one PHB scheduling group (2-3 PHBs) per LSP for AF

# 6 Internet Traffic Engineering

## 6.1 Introduction

In the next two lectures we face the issue of how service providers can efficiently provision and manage their network resources to accommodate their resource commitments. In this very competitive business, service providers must balance two conflicting goals:

- on the one hand, they must meet the customer's expectation of guaranteed and differentiated services;
- on the other, they must manage network resources well to reduce the cost of provisioning the services.

Many of these issues are addressed by techniques collectively called *Traffic Engineering* (*TE* for short).

## 6.2 Optimization Objectives

Traffic engineering aims to improve network performance through optimization of resource utilization in the network. One important issue that we need to address before we go further is the objective of the optimization. The optimization objective tends to vary depending on the specific problem that service providers try to solve. However common objectives include:

- minimizing congestion and packet losses in the network;
- improving link utilization;
- minimizing the end-to-end delay experienced by packets;
- increasing the number of customers with the current assets.

Congestion is one of the most difficult problems that service providers face in their networks. When a network is congested, the perceived performance suffers as queuing delay and packet losses increase. Therefore one useful objective would be to minimize congestion.

Two main factors are responsible for network congestion:

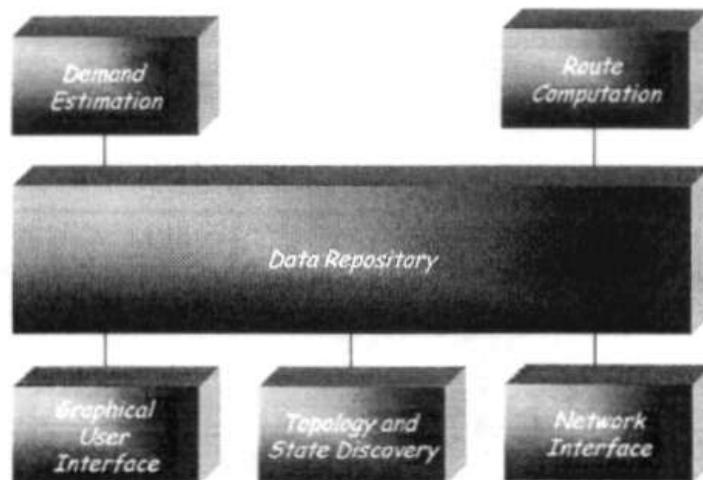
1. Inadequate network resources are often the reason for congestion in many networks. The traffic demands simply far exceed what networks can accommodate. In this case all network resources are highly utilized.
2. Congestion, however, can also be caused by *unbalanced traffic distribution*. When the traffic is distributed unevenly across the network, some links in the network are overloaded while other links are underutilized.

Although the problem of inadequate network resources must be solved with new capacity or by reducing and controlling the demands, unbalanced traffic distribution can be addressed through better management of the resources in the network. With carefully arranged traffic trunks, service providers can spread traffic across the network to avoid hot spots in parts of the network. When we translate this into a mathematical formulation, the objective is in essence to *minimize the maximum of link utilization in a network*. Intuitively the hot spots are the points with the highest link utilization. Reducing the link utilization at these points balances the traffic distribution across the network. This optimization objective of minimizing the maximum of link utilization has a number of desirable features.

- First, minimizing the maximum of link utilization can at the same time reduce the total delay experienced by the packets.
- Second, it can also be shown that the total losses are minimized.
- Third, this optimization objective moves traffic away from congested hot spots to less utilized parts of the network, so that the distribution of traffic tends to be balanced.
- Fourth, it also leaves more space for future traffic growth.

When the maximum of link utilization is minimized, the percentage of the residual bandwidth on links (*unused* or *available bandwidth*) is also maximized. The growth in traffic therefore is more likely to be accommodated and can be accepted without requiring the rearrangement of connections.

## 6.3 Building Blocks of a Traffic Engineering System



The block diagram of a TE System includes six main components:

- Topology and State Discovery
- Traffic Demand Estimation
- Route Computation
- Graphical User Interface
- Network Interface
- Data Repository

**Data Repository:** The Data Repository module provides a central database and persistent storage for all shared data objects, such as the:

- network topology,
- link state,
- traffic demands,
- routes, and
- policies.

**Topology and State Discovery:** In a TE system, any changes in the network topology and link state must be monitored. Dynamic information such as residual (available) bandwidth or link utilization may be collected through network management systems such as SNMP traps and polling.

**Traffic Demand Estimation:** A TE system relies on reasonably accurate information about the traffic demands of users. In some scenarios, for example, a VPN service, the traffic demands may be specified explicitly as part of the service agreement between the service providers and the customers. In other cases the demands may have to be estimated based on traffic measurement. Many types of statistics are available from the networks, such as traffic loads on links. However, traffic engineering typically uses aggregated traffic statistics. *Example: for route optimization it is necessary to know the traffic load between any pair of ingress and egress nodes.*

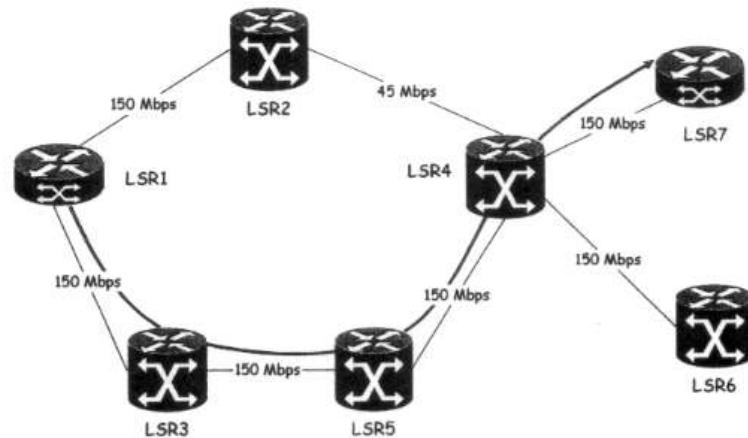
**Route Computation:** Central to a TE system is the route computation engine, which calculates routes based on traffic demands. In current IP routing, the shortest-path computation is used to select routes based on the costs assigned to links. For traffic engineering, however, the route selection must be subject to multiple complex constraints. Example: select a path between router A and router B such that the rate of the path is no less than 50 Mbps and routers C, D and E should not be crossed. For this reason, routing for traffic engineering is often referred to as constraint-based routing. Constraint-based routing can be performed in two different modes: *off-line* or *on-line*. In the *off-line mode*, route computation is performed for all routes periodically with current information, and the network cuts over to the new routes during maintenance periods. This approach tends to yield optimal results since all routes are systematically reoptimized after changes. In the *on-line mode*, route computation is performed in an incremental fashion as each new traffic demand arrives. When a new demand is requested, the route computation module calculates the optimal route for the new demand only, without changing the routes for existing demands. The benefits of on-demand route computation and minimum impacts on existing traffic, however, often outweigh the loss in efficiency. The two modes may be combined at different time scales. For example, the routes for new demands can be placed incrementally, and after some time interval complete reoptimization is performed for all demands during less busy periods.

**Network Interface:** Once the traffic-engineering system has worked out the optimal routes for the demands, it has to configure the network elements in the network accordingly. A traffic-engineering

system can interface a number of options with the network elements. When a Web server is embedded in the controller of the network elements, a traffic-engineering system can configure network elements via a Web-based user interface; thus this approach is highly flexible. However, such an interface is vendor-specific, and so it will not work between different vendors. An alternative, standard-based approach is to use SNMP for configuration of network elements.

## 6.4 Constraint-Based Routing

Constraint-based routing has two basic elements: *Route Optimization*, and *Route Placement*. *Route Optimization* is responsible for selecting routes for traffic demands subject to a given set of constraints. Once the routes are decided, *Route Placement* implements these routes in the network so that the traffic flows will follow them. A topic related to constraint-based routing is QoS routing. Several schemes have been proposed for finding a path under multiple constraints; for example, bandwidth and delay. These algorithms, however, are typically "greedy" in the sense that they try to find a path that meets a particular request without considering the networkwide impacts. In contrast, the objective of traffic engineering is the optimization of overall network performance. Consider a network that is represented by a graph  $(V, E)$ , where  $V$  is the set of nodes and  $E$  is the set of links that interconnect these nodes. Associated with each link is a collection of attributes.



For each pair of nodes in the graph, there is a set of constraints that have to be satisfied by the path from the first node in the pair to the second one. This set of constraints is expressed in terms of the attributes of the links and is usually known only by the node at the head end of the path (the first node in the pair). The goal of constraint-based routing is to compute a path from one given node to another, such that the path:

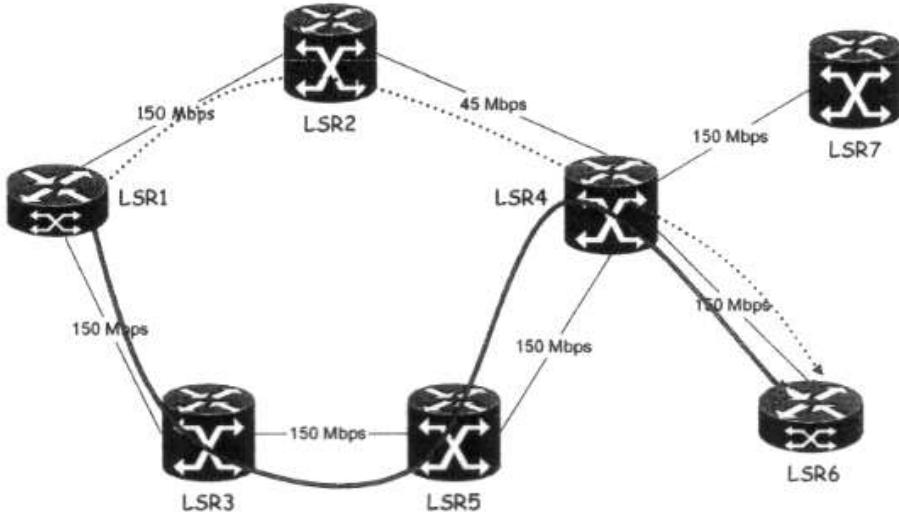
- doesn't violate the constraints and
- is optimal with respect to some scalar metric.

Once the path is computed, constraint-based routing is responsible for establishing and maintaining forwarding state along such a path (*Route Placement*).

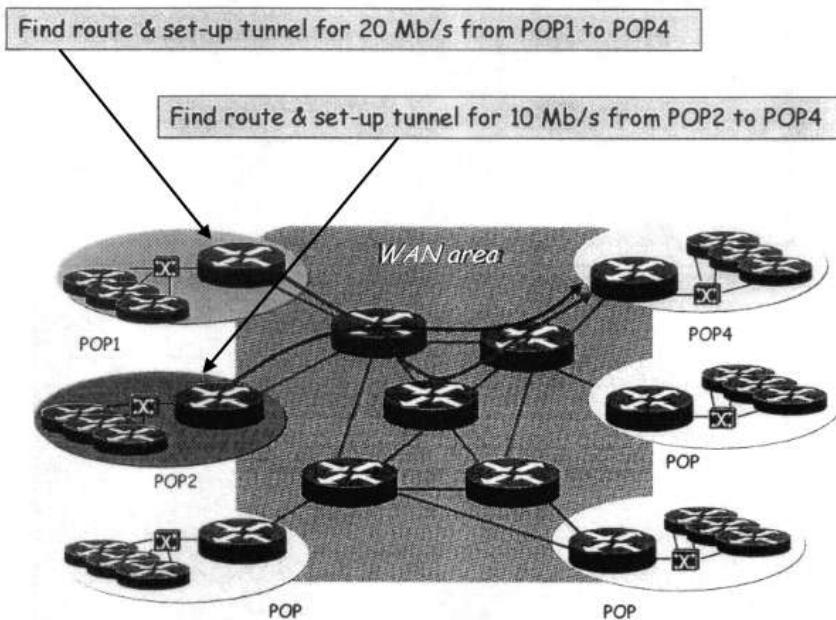
The difference between conventional IP routing and constraint-based routing: Plain IP routing algorithms aim to find a path that optimizes a certain scalar metric (e.g., minimizes the number of hops), while constraint-based routing algorithms set out to find a path that *optimizes a certain scalar metric and at the same time does not violate a set of constraints*.

## 6.5 Types of Constraint

*Example 1 - Performance Constraint.* The ability to find a path with certain *minimum available bandwidth*. In this case the constraint imposed on the routing algorithm is that the path computed by the algorithm must have at least that amount of available bandwidth on all the links along the path, and the link attribute we use is the link available bandwidth. Find a path from LSR1 to LSR6 that has minimum administrative distance and at least 100 Mb/sec available bandwidth. That is, the constraint we need to satisfy is the *constraint on available bandwidth*.

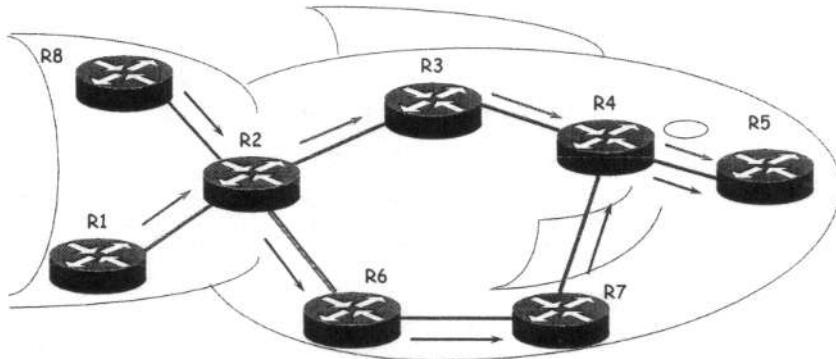


The resulting constrained shortest path from LSR1 to LSR6 is (LSR1, LSR3, LSR5, LSR4, LSR6), which is different from the path that would be computed by plain SPF, which would be (LSR1, LSR2, LSR4, LSR6). Note that different paths within a given network may have different bandwidth constraints associated with them. For example, for one pair of nodes the path from the first node in the pair to the second one may require one value of the minimum available bandwidth, while for another pair of nodes the path from the first node in the pair to the second one may require a different value of the minimum available bandwidth.



**Example 2a - Administrative Constraint.** A network administrator may want to exclude certain traffic from traversing certain links in a network, where such links would be identified by a particular link attribute. In this case the constraint imposed on the routing algorithm is that *the path for that traffic must not traverse through any of the links that have to be excluded*.

**Example 2b - Administrative Constraint.** The network administrator may want to require certain traffic to traverse only certain links in a network, where again the links would be identified by a particular link attribute. In this case the constraint imposed on the routing algorithm is that *the path for that traffic must traverse only these certain links*.



Just like with performance constraints, with administrative constraints different paths may have different administrative constraints associated with them. For example, for one pair of nodes the path from the first node in the pair to the second one may have one set of links to be excluded, while for another pair of nodes the path from the first node in the pair to the second one may require a different set of links to be excluded.

*Example 3 - Mixed Constraints.* Constraint-based routing may also include a combination of performance and administrative constraints, not just one or the other. For example, constraint-based routing should be able to find a path that has certain available bandwidth and at the same time excludes certain links.

## 6.6 Constraint-Based Routing versus Plain IP Routing

Constraint-based routing can't be supported by plain IP routing for the following reasons.

*First Reason:* Constraint-based routing requires *route (path)* calculation at the source since different sources may have different constraints for a path to the same destination, and the constraints associated with a particular source router are known only to that router, but not to any other router in a network. With plain IP routing, a route (path) is computed in a distributed fashion by every router in a network and this computation does not take into account constraints of different sources. In fact, plain IP routing can't take into account constraints of different sources, as these constraints are local to the individual sources and are not distributed throughout a network.

*Second Reason:* When a path is determined by the source, forwarding along such a path can't be provided using the destination-based forwarding paradigm, which is the paradigm that plain IP routing uses. Some sort of explicit (or source) routing capabilities are required, because different sources may compute different paths to the same destination; therefore, the destination by itself is insufficient to determine how to forward packets.

*Third Reason:* Since the path computation at the source needs to take into account the information about attributes associated with the individual links in a network, there should be some way of distributing such information throughout the network. Plain IP routing doesn't provide this; conventional IP Link-State Routing Protocols (e.g., OSPF) distribute only the information about the state (up/down) of individual links and the administrative metric assigned to each link, and Distance Vector Routing Protocols (e.g., RIP) distribute only the address of the next hop and the distance.

While constraint-based routing can't be supported by plain IP routing, it doesn't mean that the plain IP routing can't be augmented with additional capabilities to support such functionality; in fact it can. Moreover, by augmenting plain IP routing, we can build a single routing system where it is possible to mix and match plain IP routing with constraint-based routing. For example, with such a system some traffic could be routed via plain IP routing, while some other traffic could be routed via constraint-based routing. The most important property of a routing system that combines both plain IP routing and constraint-based routing is that such a system must provide diversity of routing for applications that need it, while at the same time providing good scaling via routing information aggregation whenever possible.

## 6.7 Mechanisms

Key mechanisms needed to support constraint-based routing are:

*First Mechanism:* The ability to compute a path at the source, and to compute it in such a way that the computation can take into account not just some scalar metric that is used as an optimization criteria but also a set of constraints that should not be violated. This requires the source to have all the information needed to compute such a path which can be partitioned into the information available locally at the source, and information that the source has to obtain from other

routers in a network. The information available at the source is the information about the constraints of various paths originated by the source. The information that the source has to obtain from other routers in a network includes the information about the network topology as well as the information about various attributes associated with the links in a network.

Since potentially any node in a network may originate traffic that has to be routed via constrained-based routing, this information has to be available to every node in a network. So:

**Second Mechanism:** The ability to distribute the information about network topology and attributes associated with links throughout the network.

Once we compute the path, we also need a way to support forwarding along such a path. So:

**Third Mechanism:** Support of explicit routing.

Establishing a route for a particular type of traffic may require reservation of resources along the route which may alter the value of the attributes associated with individual links in a network. For example, if available bandwidth is one of the link constraints that we are trying to satisfy for some type of traffic, establishing a route for that traffic requires bandwidth reservation along the route, which, in turn, changes the amount of available bandwidth on the links along the route. So:

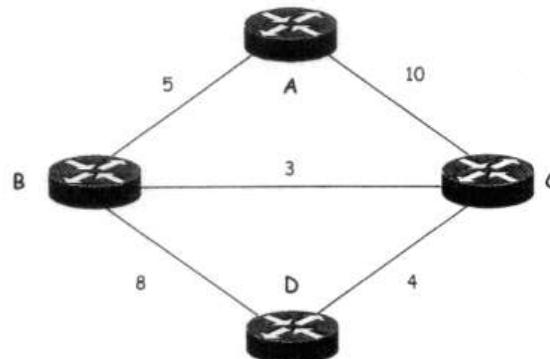
**Fourth Mechanism:** Is one by which network resources can be reserved and link attributes can be modified as the result of certain traffic taking certain routes.

## 6.8 Constrained Shortest Path First (CSPF)

The algorithm used for path computation requires the router that performs path computation to have information about all the links in a network. That, in turn, imposes a restriction on the type of routing protocols that we can use to support constraint-based routing - we have to use Link-State Routing Protocols (e.g. SPF), as Distance Vector Routing Protocols (e.g. RIP) are not capable of meeting the requirement. To compute a path that doesn't violate constraints, all we need to do is *to modify the SPF algorithm in such a way that it will be able to take into account the constraints*. We'll refer to such an algorithm as *Constrained Shortest Path First (CSPF)*.

## 6.9 How SPF Works

In a link-state routing protocol, each router knows about all other routers in a network and the links that connect these routers. In OSPF, this information is encoded as Link-State Advertisements (LSAs). As soon as a router knows about all other routers and links, it runs the Dijkstra Shortest Path First algorithm to determine the shortest path between the calculating router and all other routers in the network. Because all routers run the same calculation on the same data, every router has the same picture of the network, and packets are routed consistently at every hop. Understanding SPF is fundamental to understanding CSPF, which is based on the basic Dijkstra SPF algorithm. The following example walks through what happens when Router A runs SPF and generates its routing table.



After each router has flooded its information to the network, all the routers know about all the other routers and the links between them. So the *Link State Database* on every router looks like:

Router	(neighbor,cost) Pairs
A	(B,5) (C,10)
B	(A,5) (C,3) (D,8)
C	(A,10) (B,3) (D,4)
D	(B,8) (C,4)

So what does Router A do with this information? In SPF calculation, each router maintains two lists:

- A list of nodes that are known to be on the shortest path to a destination. This list is called the PATH list or sometimes the PATHS list.
- A list of next hops that might or might not be on the shortest path to a destination. This list is called the TENTative or TENT list.

Each list is a table of (router,distance,next-hop) triplets from the perspective of the calculating router. Question: "why a triplet?" Answer:

1. You need to know which router you're trying to get to in PATH and TENT lists, so the first part of the triplet is the name of the router in question (In reality, this is the router ID (RID) rather than the textual name, but for this example, we'll use the router name).
2. The next hop is the router you go through to get to the node in question.
3. The third item in the triplet is the distance, which is the cost to get to the node in question.

Each router runs the following algorithm for computing the shortest path to each node:

Step 1:

- Put "self" on the PATH list with a distance of 0 and a next hop of self.
- The router running the SPF refers to itself as either "self" or the root node, because this node is the root of the shortest-path tree.

Step 2:

- Take the node just placed on the PATH list, and call it the PATH node.
- Look at the PATH node's list of neighbors.
- Add each neighbor in this list to the TENT list with a next hop of the PATH node, unless the neighbor is already in the TENT or PATH list with a lower cost.
- Call the node just added to the TENT list the TENT node.
- Set the cost to reach the TENT node equal to the cost to get from the root node to the PATH node plus the cost to get from the PATH node to the TENT node.
- If the node just added to a TENT list already exists in the TENT list, but with a higher cost, replace the higher-cost node with the node currently under consideration.

Step 3:

- Find the neighbor in the TENT list with the lowest cost, add that neighbor to the PATH list, and repeat Step 2.
- If the TENT list is empty, stop.

Consider the process that Router A goes through to build its routing table.

Step 1:

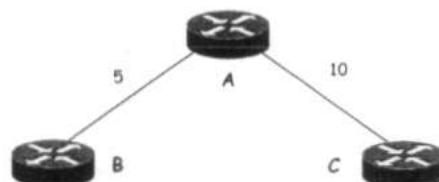
PATH List	TENT List
(A,0,0)	(empty)



Step 2:

(B,5,B) and (C,10,C) get added to the TENT list.

PATH List	TENT List
(A,0,0)	(B,5,B) (C,10,C)



Step 3:

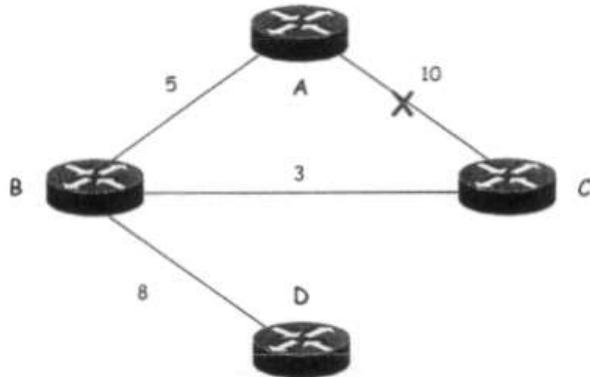
(B,5,B) is moved to the PATH list, because that's the shortest path to B. Because (C,10,C) is the

only other neighbor of Router A, and the cost to get to C is greater than the cost to get to B, it's impossible to ever have a path to B that has a lower cost than what's already known.

PATH List	TENT List
(A,0,0)	(C,10,C)
(B,5,B)	

#### **Step 4:**

Router B's neighbors are examined. Router B has a link to C with a cost of 3 and a link to D with a cost of 8. Router C, with a cost of 5 (to get from "self" to B) + 3 (to get from B to C) = 8 (the total cost from A to C via B) and a next hop of B, is added to the TENT list, as is Router D, with a cost of 5 (the cost to get from the root node to B) + 8 (the cost to get from B to D) = 13 and a next hop of B. Because the path to C with a cost of 8 through B is lower than the path to C with a cost of 10 through C, the path to C with a cost of 10 is removed from the TENT list.



PATH List	TENT List
(A,0,0)	(C,10,C)
(B,5,B)	(C,8,B) (D,13,B)

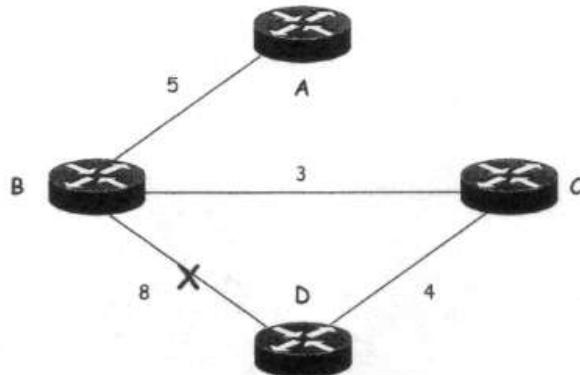
## **Step 5:**

The path to C through (C,8,B) is moved from the TENT list to the PATH list.

PATH List	TENT List
(A,0,0)	
(B,5,B)	
(C,8,B)	(D,13,B)

### **Step 6:**

The path to D through B (which is really B->C->D) with a cost of 12 replaces the path to D through B->D with a cost of 13.



PATH List	TENT List
(A,0,0) (B,5,B) (C,8,B)	(D,13,B) (D,12,B)

**Step 7:**

The path to D is moved to the PATH list.

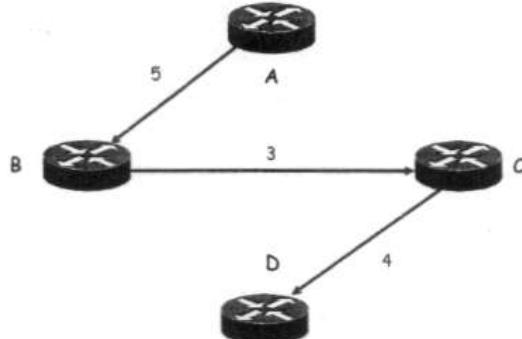
PATH List	TENT List
(A,0,0) (B,5,B) (C,8,B) (D,12,B)	

**Step 8:**

The TENT list is empty because D has no neighbors that are not already on the PATH list, so you stop. At this point, the PATH list becomes As routing table, which looks like the following table:

Router	Cost	Next Hop
A	0	Self
B	5	B (directly connected)
C	8	B
D	12	B

Router A's view of the network after running the SPF algorithm:



As you can see, traffic from Router A never crosses the links from A to C or the links from B to D.

## 6.10 How CSPF Works

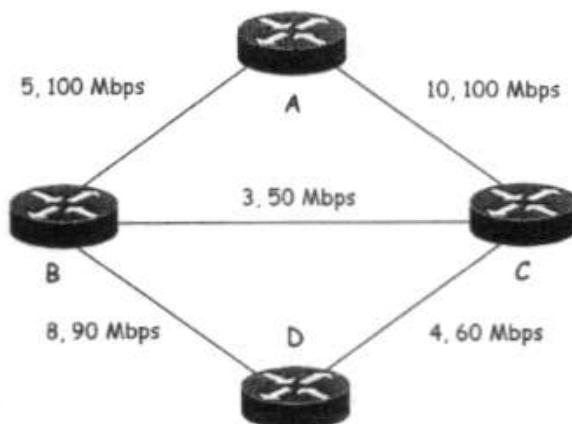
The process that generates a path for a TE tunnel to take is different from the regular SPF

process, but not much. There are two major differences between regular SPF done by routing protocols, and CSPF, run by MPLS Traffic Engineering.

*First:* The path determination process is not designed to find the best route to all routers - only to the tunnel endpoint. This makes the SPF algorithm slightly different: you stop as soon as the node you're trying to get to is on the PATH list, rather than trying to calculate the shortest path to all nodes.

*Second:* There's now more than the cost at each node. In addition to the cost for a link between two neighbors, there are also attributes (e.g. available bandwidth).

So how does CSPF work? To help you better understand this concept, let's run through a couple of examples. The first one is similar to the example we developed for SPF, but each link is advertising its available bandwidth. For simplicity, each item reported in the PATH/TENT lists is characterized by (neighbor, cost, next hop, and available bandwidth).



Router A wants to build a path to Router D with an available bandwidth of 60 Mbps. Each link lists its cost (metric) and its available bandwidth. Without taking the available bandwidth into account, Router A's best path to Router D is A->B->C->D, with a total cost of 12. But the path A->B->C->D doesn't have 60 Mbps of available bandwidth. CSPF needs to calculate the shortest path that has 60 Mbps of available bandwidth. The steps for the CSPF algorithm are as follows:

#### Step 1:

- Put "self" on the PATH list with a distance of 0 and a next hop of self.
- Set the available bandwidth to N/A.

#### Step 2:

- Take the node just placed on the PATH list, and call it the PATH node.
- Look at that node's neighbor list.
- Add each neighbor in this list to the TENT list with a next hop of the PATH node, unless the neighbor is already in the TENT or PATH list with a lower cost.
- Do not add this path to the TENT list unless it meets all configured constraints for the desired path - available bandwidth and affinity.
- If the node just added to the TENT list already exists in the list, but with a higher cost or lower minimum bandwidth, replace the higher-cost path with the path currently under consideration.

#### Step 3:

- Find the neighbor in the TENT list with the lowest cost, add that neighbor to the PATH list, and repeat Step 2.
- If the TENT list is empty or the node that is the tail of the path is on the PATH list, stop.

Given that the rules are similar, here's the SPF calculation for a path from A to D that needs 60 Mbps. Instead of (node, cost, next hop), the PATH and TENT lists contain information encoded as (node, cost, next hop, minimum bandwidth). This example follows the same general format as the previous SPF example but some of the basic steps are compressed into one (for example, moving a node from TENT to PATH and putting its neighbors on the TENT list).

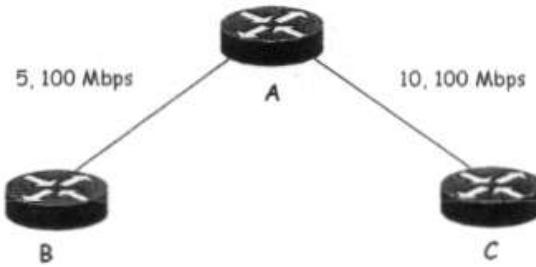
#### Step 1:

PATH List	TENT List
(A, 0, self, N/A)	(empty)



**Step 2:**

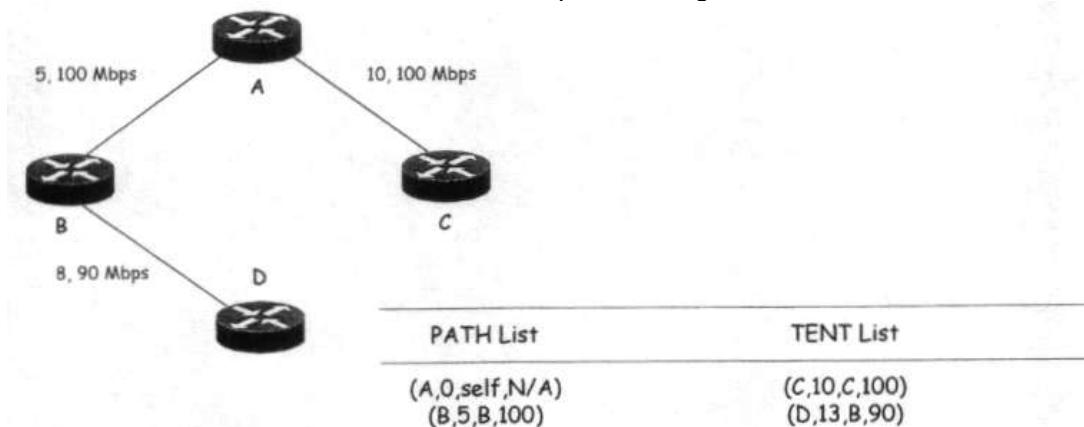
Put Router A's neighbors in the TENT list.



PATH List	TENT List
(A,0,self,N/A)	(B,5,B,100) (C,10,C,100)

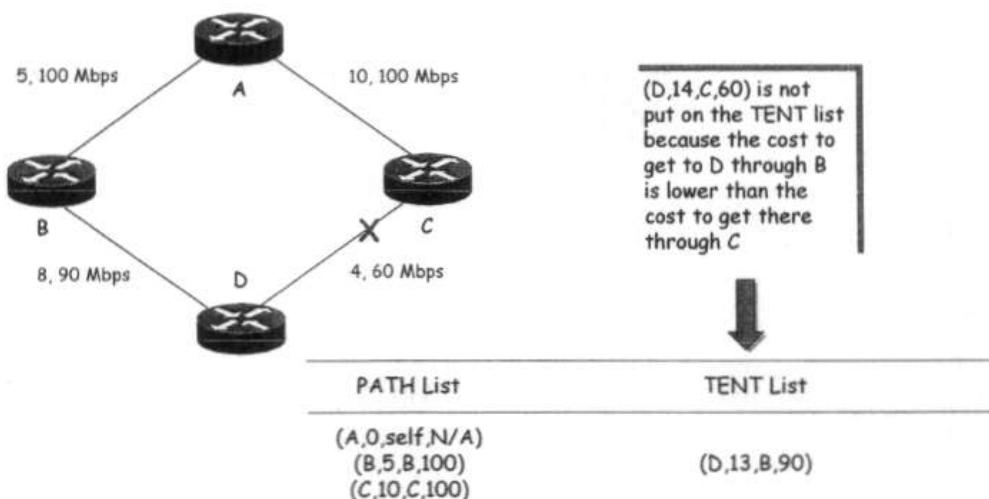
**Step 3:**

Move B from the TENT list to the PATH list, and put B's neighbors in the TENT list.



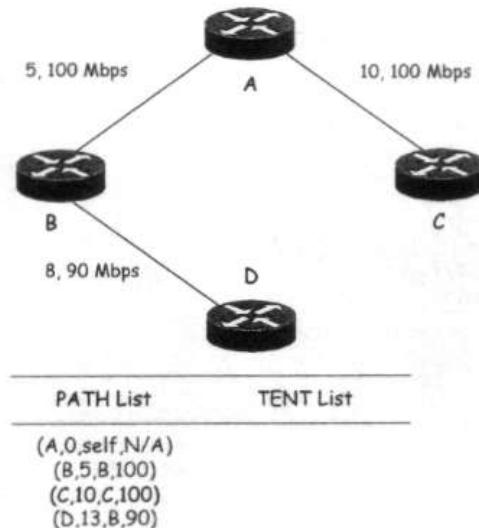
**Step 4:**

Put B's neighbors in the TENT list, and take C from TENT and put it in PATH.



**Step 5:**

Take D off the TENT list. At this point, the best possible path to D is in the PATH list, so you're done. The fact that the TENT list happens to be empty is an artifact of the network topology; D happened to be the last node you encountered in the SPF. If you had reached the best path to D and there were still more nodes in the TENT list, you'd have stopped anyway.



In reality, the path calculation is more complex than what is covered here CSPF has to keep track of all the nodes in the path, not just the next hop.

## 6.11 RSVP Extensions

We already know how to establish label forwarding state with RSVP by using the RSVP Label Object. What we still miss is the ability to establish this state along an explicit route. Plain RSVP, augmented by the Label Object, can establish MPLS forwarding state (an LSP) only along the path computed by plain IP routing. This is because with plain RSVP the path taken by the RSVP PATH message is controlled by the destination-based forwarding paradigm, and the path taken by the PATH message determines the path taken by the LSP. When a router has to forward the PATH message, the router uses a combination of its forwarding table constructed by such protocols as IS-IS, OSPF, RIP, or BGP and the destination address in the IP packet that carries the message to determine the next hop router. What is missing is the ability to "steer" the PATH message along a particular explicit route. To accomplish this RSVP is augmented by a new object, the *Explicit Route Object (ERO)*. This object is carried in the PATH message and contains the explicit route that the message has to take. Forwarding of such a message by a router is determined not by the destination address in the IP header of the packet that carries the message, but rather by the content of the ERO carried in the message. The ERO consists of a sequence of subobjects, i.e. an ordered sequence of *hops*, where the sequence specifies an explicit route. Each hop is represented by an *abstract node*. Each abstract node identifies a group of one or more routers, whose internal topology is opaque to the router that computes the explicit route. One example of an abstract node would be a collection of routers that belong to a particular autonomous system. Another example of an abstract node would be a collection of routers whose addresses fall within a particular address prefix. From the encoding point of view, the ERO is a sequence of <type,length,value> triples, where each triple depicts a particular abstract node. There are three different types of abstract nodes defined:

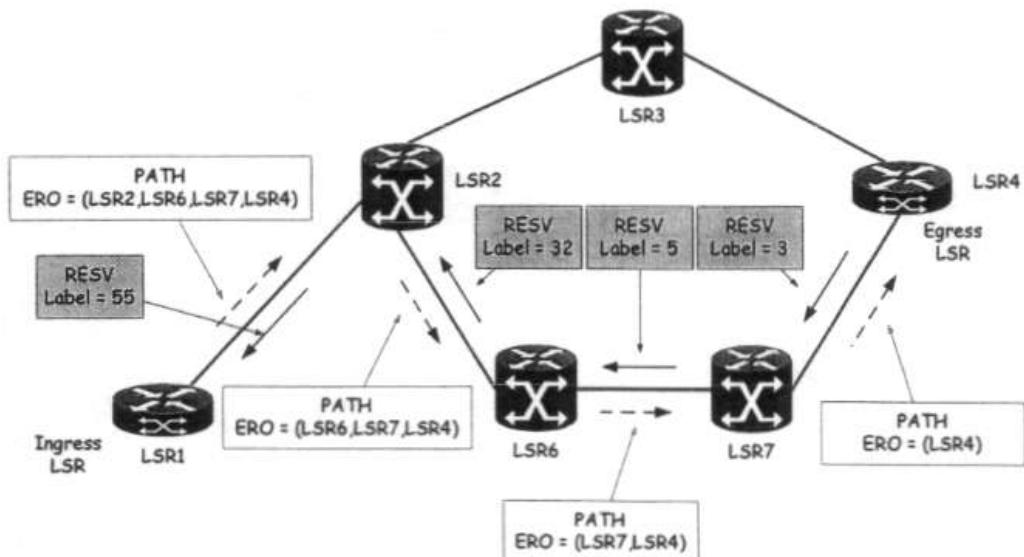
- IPv4 address,
- IPv6 address,
- Autonomous System.

Cisco IOS software currently supports only IPv4 addresses in the ERO (to be verified).

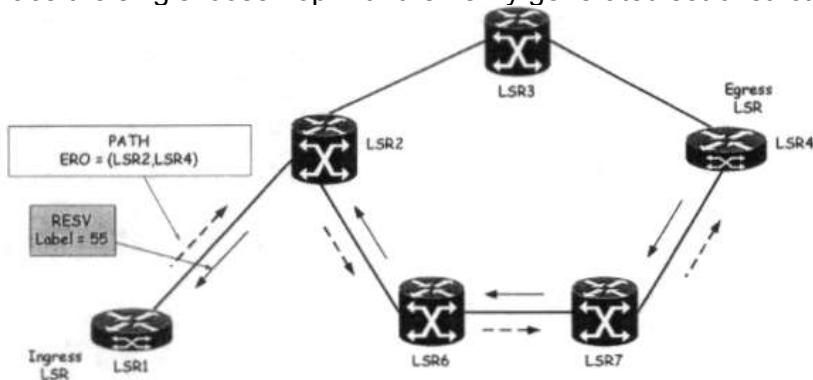
ERO Structure:

L	Type (7 bits)	Length (8 bits)	IPv4 address (first 16 bits)		
	IPv4 address (last 16 bits)	Prefix Length (8 bits)	Reserved (8 bits)		

Each subobject can be either a *strict hop* or a *loose hop*. When a router processes a strict hop, the IPv4 address in the subobject must be directly connected to the router doing the processing.



One possible use of loose hops is when the router that computes an explicit route doesn't have sufficient topology information to compute the path in terms of strict hops, for example, when the route has to cross multiple OSPF areas. If a router processes an ERO subobject with a loose hop, it is the responsibility of that router to generate a set of strict hops to get this PATH message to the destination and replace the single loose hop with the newly generated set of strict hops.



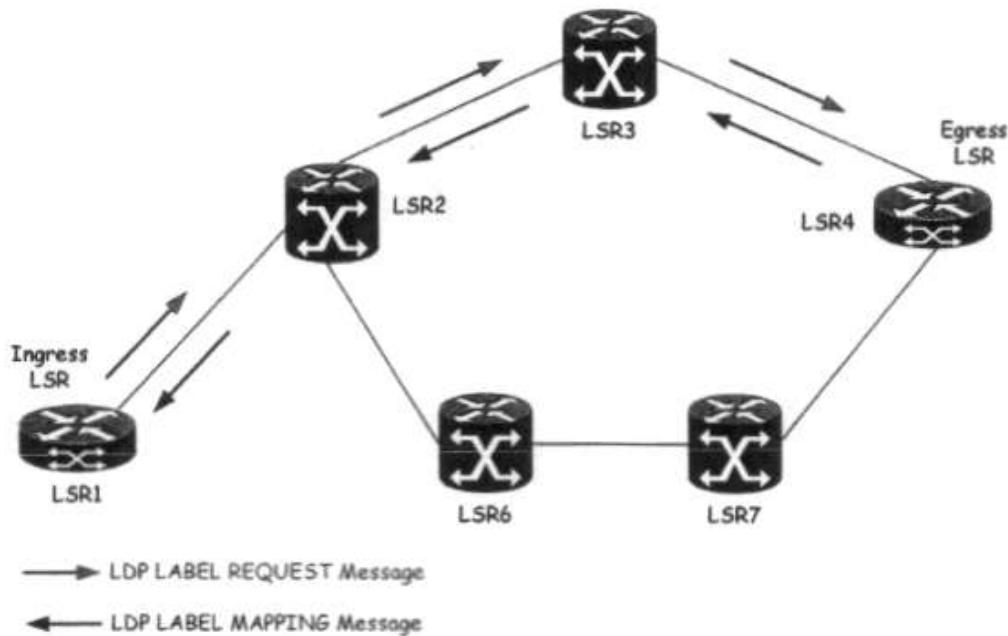
Assume that LSR1 wants to establish MPLS forwarding state along the explicit route (LSR2, LSR6, LSR7, LSR4). LSR1 constructs an ERO that contains a sequence of four abstract nodes, LSR2, LSR6, LSR7, and LSR4. Each abstract node is represented by an IP address prefix of length 32 (which is nothing but a plain IP address) which could depict either one particular interface on an LSR (and in this case it is the IP address associated with that interface) or the whole LSR (in this case, the IP address associated with the loopback address of the LSR). LSR1 then constructs an RSVP PATH message and includes in this message the ERO as well as the LABEL\_REQUEST object. LSR1 examines the first abstract node in the ERO, LSR2, finds the link that connects LSR1 to LSR2, and sends the PATH message over that link. When LSR2 receives the PATH message, LSR2 finds itself as the first abstract node in the ERO. LSR2 then looks at the next abstract node, LSR6, and finds the link that connects it to LSR6. LSR2 then modifies the ERO by removing the abstract node associated with LSR2 from the ERO and sends the PATH message that contains the modified ERO and the LABEL\_REQUEST object to LSR6 (The modified ERO at this point contains just LSR6, LSR7, and LSR4). Handling of the PATH message (including handling of the ERO and the LABEL\_REQUEST object) by LSR6 is similar to how the message is handled by LSR2. The ERO in the PATH message that LSR6 sends to LSR7 contains just LSR7 and LSR4. Likewise, handling of the PATH message (including handling of the ERO and the LABEL\_REQUEST objects) by LSR7 is similar to how the message is handled by LSR6. The ERO in the PATH message that LSR7 sends to LSR4 contains just LSR4. When the PATH message arrives at LSR4, LSR4 finds, by examining the ERO, that it is the last abstract node in the ERO. Given that, LSR4 creates a RESV message including, among other things, the LABEL object and sends this message to LSR7. LSR7, when it receives the message, uses the label carried in the LABEL object to populate its label forwarding table. Following this, LSR7 sends a RESV message including, among other things, the LABEL object to LSR6. Handling of this message by LSR6 is similar to the way the message is handled by LSR7. LSR6 sends the RESV message which includes the LABEL object to LSR2. Handling of this message by LSR2 is similar to the way the message is handled by LSR7. Finally, the message is received by LSR1. At this point the LSP for the explicit route is established. The procedures for handling the ERO are in addition to, not

instead of, the procedures for handling RSVP without MPLS. That is, when RSVP is used to establish an LSP, handling of RSVP on the LSRs along the LSP is defined by a combination of plain RSVP procedures and MPLS-specific procedures. Therefore, establishing an LSP along an explicit route may involve reserving resources (e.g., bandwidth) along the route, which is handled by standard RSVP procedures. Note that in this case the full set of QoS parameters (e.g. peak bandwidth, sustained rate, burst size) defined for Integrated Services may be used. LSP setup may also involve adjusting the amount of available resources (e.g., available bandwidth) on all the links along the route and this is handled by standard RSVP procedures.

## 6.12 CR-LDP

CR-LDP is an alternative to RSVP to establish MPLS forwarding state along an explicit route. We already know how to establish label forwarding state between adjacent LSRs with LDP. What is missing in LDP is the ability to *establish label forwarding state* on all the LSPs along an explicit route and the *ability to reserve resources* along the route. CR-LDP contains a set of extensions to LDP that provides these missing pieces. To establish label forwarding state along an explicit route, CR-LDP introduces a new object, called Explicit Route (ER). The structure of this object, as well as its handling by LSRs, is almost identical to the ERO. This ER object is carried as a <type,length,value> triple in the LDP LABEL REQUEST message. Label binding for explicit routes is established by using the downstream-on-demand label advertisement, which, in turn, uses the LDP LABEL REQUEST message. To support explicit routing, the LDP LABEL REQUEST message also includes the ER object.

Assume that LSR1 wants to establish MPLS forwarding state along the explicit route (LSR2, LSR3, LSR4).



LSR1 constructs the ER object that contains a sequence of three abstract nodes, LSR2, LSR3, and LSR4. Each node is represented by a plain IP address. Such an address could depict either one particular interface on an LSR (in this case, the IP address associated with that interface) or the whole LSR (the IP address associated with the loopback address of the LSR). LSR1 then constructs an LDP LABEL REQUEST message and includes in this message the ER object. Once the message is constructed, LSR1 examines the first abstract node in the ER object, LSR2, finds the link that connects it to LSR2, and sends the LABEL REQUEST message over that link. When LSR2 receives the LABEL REQUEST message, LSR2 finds itself as the first abstract node in the ER object carried by the message. LSR2 then looks at the next abstract node, LSR3, and finds the link that connects it to LSR3. LSR2 then modifies the ER object by removing the abstract node associated with LSR2 from it and sends the LABEL REQUEST message that contains the modified ER object to LSR3 (the ER object contains just LSR3 and LSR4). Handling of the LABEL REQUEST message (including handling of the ER object) by LSR3 is similar to how the message is handled by LSR2. When the message arrives at LSR4, LSR4 finds that it is the last abstract node in the ER object. Therefore, LSR4 creates a LABEL MAPPING message which includes, among other things, the LABEL object, and sends it to LSR3. When it receives the message, LSR3 uses the label carried in the message to populate its label forwarding table. Then LSR3 sends a LABEL MAPPING message which includes, among other things, the LABEL object, to LSR2.

Handling of this message by LSR2 is similar to the way the message is handled by LSR3. Finally, the message is received by LSR1, and the LSP for the explicit route is established.

To carry information about resources that need to be reserved along a route, CR-LDP uses the *Traffic Parameter* object. At present, the CR-LDP specifications define seven traffic parameters:

- Peak Data Rate (PDR),
- Peak Burst Size (PBS),
- Committed Data Rate (CDR),
- Committed Burst Size (CBS),
- Excess Burst Size (EBS),
- Frequency,
- Weight.

Peak data rate and peak burst size together define a token bucket, which characterizes the maximum rate of traffic that is expected to be sent down to this LSP. Similarly, committed data rate and committed burst size define a token bucket characterizing the average rate at which traffic is expected to be sent on this LSP. Excess burst size defines another token bucket that can be used to characterize the amount by which bursts may exceed the committed burst size. It is expected that all of these parameters might be used at LSRs along an LSP to determine how to allocate resources to an LSP as it is established. They may also be used to perform policing actions at the entrance to an LSP; for example, packets that do not conform to the token bucket defined by MR and PBS might be dropped at that point.

## **6.13 Distribution of Network Attributes Informations**

CSPF assumes that a node that performs CSPF has information not just about the state (up/down) of all the links in a network but also about link attributes, like available bandwidth, which allow the node to determine whether a particular link violates the constraints associated with a particular path that the node has to compute. Since each node in a network could, at least in principle, perform CSPF, we need to have a way to provide each node in a network with such information. One way to do this is to utilize *flooding mechanisms* used by link-state protocols such as OSPF. These flooding mechanisms distribute information about the state of a particular link (link up/down state) to every node in a network (to be more precise, to every node within a single OSPF area). To accomplish such a distribution, we can just piggyback the information on top of the link-state information. With OSPF this information is carried in the Opaque Link-State Advertisement (Opaque LSA). The information is encoded as a collection of type-length-value objects, where each object carries information related to a particular attribute. Establishing a route for a particular set of traffic may require resource reservation along the route and therefore may alter attributes of the links along that route. When the attribute information of a link changes, it is important to inform all the nodes in a network about such a change. To accomplish this, when the attribute information of a given link changes, the node to which the link is connected floods this information (using the reliable flooding protocol) throughout the network. If every change in the attribute information of a link resulted in flooding this information throughout the network, it could result in an excessive amount of routing updates to carry the information, which in turn could overload the control component of the routers involved in flooding. To avoid this, a set of thresholds may be used. For example, when the attribute is the available bandwidth, then a change in the available bandwidth of a link would trigger flooding of this information only if the available bandwidth crosses a certain threshold. Also note that OSPF already has built-in mechanisms that impose an upper bound on how frequently a router can originate flooding of the information associated with one of the router's links. These mechanisms apply as well when flooding is triggered by changes in the link attribute information.

## **6.14 Comparison of CR-LDP and RSVP**

Similarities between CR-LDP and RSVP:

1. The Explicit Route Objects that they use are extremely similar.
2. Both protocols use ordered LSP setup proceeding from the head to the tail along the explicit route, returning to the head as the labels are assigned.
3. Both protocols include some QoS information in the signaling messages to enable resource allocation and LSP establishment to take place atomically.
4. Both protocols are extensions of previously defined protocols:
  - RSVP has been extended for traffic engineering by adding label assignment and explicit

route specification to a Resource Reservation Protocol.

- CR-LDP extends LDP by adding explicit route specification and resource allocation.

It is hard to make meaningful claims as to which set of extensions are more reasonable.

#### Differences between CR-LDP and RSVP:

We discuss the technical differences between the two protocols in terms of *scalability* and *signaling mechanisms*.

##### 1. Scalability:

When RSVP was first proposed as a standard for QoS signaling, there were concerns about its scalability for deployment in large networks. These concerns centered on the fact that RSVP makes reservations (and installs some state) for individual "microflows," that is, for streams of data corresponding, typically, to a single application running on a pair of hosts (or a group of hosts using multicast). The number of microflows passing through a single router in a large IP network could easily be in the millions, and there could be serious consequences in terms of memory and performance if a single router had to hold millions of pieces of reservation state. Unfortunately, these scalability concerns are often summarized by the simplistic statement "RSVP doesn't scale". To say that something doesn't scale is to say that it cannot reasonably be deployed in large networks such as ISP backbones. In fact, the aspect of RSVP that does not scale well is the making of reservations for individual application microflows. When RSVP is used for explicit LSP setup, no such reservations (i.e. at the microflow level) are made. The dominant factor in the scalability of an LSP setup protocol is the number of LSPs established, which is clearly independent of the protocol. Thus, the scalability argument really has no place in a comparison of RSVP and CR-LDP. The one legitimate scaling concern that may need to be examined in the context of constraint-based routing and LSP setup is *the effect of soft state on scalability*.

##### 2. Signaling Mechanisms

RSVP, as originally defined, uses a "soft state" approach to signaling. Such an approach displays robustness in the face of failures because unrefreshed state simply times out. However, RSVP's soft state also has two drawbacks:

- it requires refreshing in the steady state, which consumes bandwidth and processing resources;
- it lacks a reliable delivery mechanism.

CR-LDP, by contrast, runs on top of TCP, since it is an extension to LDP and therefore it has no refresh overhead and obtains reliability from TCP. Running on top of TCP does present some drawbacks, however:

- TCP's congestion avoidance may throttle back the transfer of information between LSRs.
- When there is only one TCP connection between two LSRs, TCP forces FIFO delivery of messages. Thus, there is no opportunity for a critical message to be delivered ahead of a less critical one that was sent first. In addition, when a packet is dropped, all messages sent behind that packet are delayed until the lost packet is successfully retransmitted.
- There is a moderate amount of overhead involved in establishing an adjacency between two LSRs. They must go through TCP's three-way handshake before initiating an LDP session.

This last point results in an advantage for RSVP, which does not require connection establishment before label distribution can occur. We can say that RSVP has "*lightweight adjacencies*", which enable new neighbor relationships to be established quickly as needed. An application of this capability is in the area of *fast reroute*.

#### One Protocol or Two?

A final point of comparison between RSVP and CR-LDP is the fact that CR-LDP can be viewed as an extension to LDP, whereas RSVP is a separate protocol and therefore you have to run more protocols when using RSVP, assuming it would not be running for any other reason. These facts are obvious, but it is not at all obvious whether it is better to have more or fewer protocols. For example, IP, ICMP, and ARP are three separate protocols that are required to send IP packets over a network; it seems unlikely that networks would run better or be easier to administer if these three protocols were somehow integrated into one. Thus, we can say no more about this argument than to point out that it has been made and that it is not an important distinction between the two approaches.

## 6.15 Application to Traffic Engineering

### Problem Description:

It has been observed that the efficiency and cost structure of the service provider business is affected by how efficiently a service provider utilizes its infrastructure and, specifically, the available bandwidth. More efficient use of bandwidth resources means that the provider should be able to avoid a situation in which some parts of its network are congested while other parts are underutilized. A cost savings that results in more efficient use of bandwidth resources helps to reduce the overall cost of operations. This, in turn, helps service providers to gain advantage over their competitors and this advantage becomes increasingly important as the service provider market gets more and more competitive. One way to solve the traffic engineering problem is by using routing, as routing determines the paths taken by the traffic that a provider has to carry and thus controls how much traffic traverses through each link in a network. Specifically, a service provider should be able to construct routes in such a way as to avoid causing some parts of the provider's network to be *overutilized* while other parts are *underutilized*.

### Solving Traffic Engineering with MPLS Constraint-Based Routing:

Any solution to the traffic engineering problem must do two things:

- establish routes that are optimal with respect to a certain scalar metric, and
- take into account the available bandwidth on individual links.

These are the capabilities that can be provided by MPLS constraint-based routing, where the *constraint* used is *available bandwidth*.

A *traffic trunk* is a collection of individual TCP, or UDP microflows, that share two common properties:

- all microflows are forwarded *along the same common path*, and
- all microflows share the same *Class of Service*.

It is important to keep in mind that the common path shared by the microflows of a traffic trunk is just a path within a single service provider, not an end-to-end path. For the purpose of traffic engineering, we use MPLS constraint-based routing to route traffic trunks, not individual microflows, within a single service provider. Routing at the granularity of traffic trunks has better scaling properties (with respect to the amount of forwarding state and the volume of control traffic) than routing at the granularity of individual microflows. As the traffic increases, the amount of traffic per individual trunk increases; but that doesn't result in an increase of the number of trunks, which means there is no increase in the forwarding state or the control traffic needed to establish and maintain this state. In contrast, if we were to apply constraint-based routing at the granularity of individual microflows, then the amount of forwarding state, as well as the control traffic needed to establish and maintain this state, would grow with the growth of the traffic that a provider has to carry. Using the concept of traffic trunks, the traffic engineering problem reduces to two subproblems:

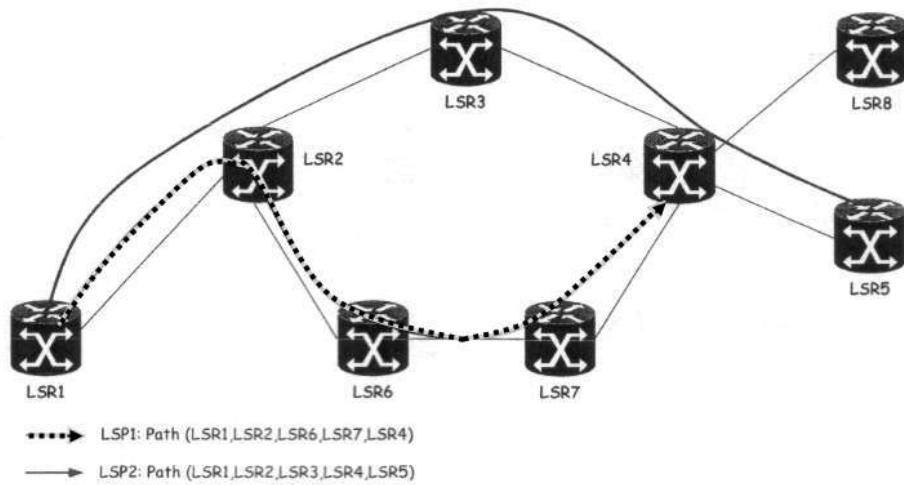
- Problem of representing the traffic that has to be carried by a service provider as a collection of traffic trunks with specific bandwidth requirements.
- Problem of using MPLS constraint-based routing to route these trunks within a service provider.

For the purpose of traffic engineering, it may be useful to support not just bandwidth but administrative constraints as well, i.e. in addition to the bandwidth constraints, it may be useful to allow a service provider to restrict the set of links that could be used to route a particular traffic trunk.

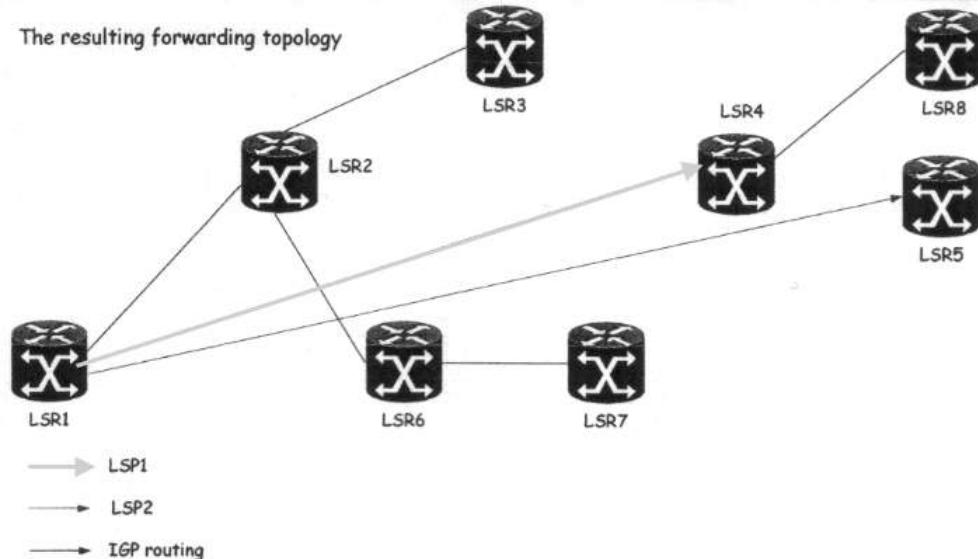
### Forwarding Packets onto an LSP:

Routing determined by MPLS constraint-based routing can be *intermixed* with routing determined by plain IP. This is important, as a service provider may decide to apply traffic engineering to only a portion of the traffic the provider has to carry. The SPF algorithm can be modified to achieve the above objective. With plain SPF, a router computes the shortest path tree rooted at the router. This shortest path tree is used to determine the outgoing interface, as well as the next hop, that the router has to use when forwarding packets. The SPF algorithm can be modified as follows: when a new router is added to the shortest path tree, the algorithm checks whether the newly added router is a tail end of an LSP whose head end is the router, and if so the algorithm uses the LSP as a "*logical*" outgoing interface.

Assume that constraint-based routing establishes two LSPs, LSP1 and LSP2, each with its own "logical" interface.



As LSR1 starts to build its shortest path tree, it follows the standard SIT/ procedures until it adds LSR4 to the tree. At this point LSR1 notices that there is an LSP, LSP1, that starts on LSR1 and ends on LSR4. Therefore, the outgoing (logical) interface for destinations reachable via LSR4 would be LSP1. As SPF continues, it adds LSR8 to its shortest path tree. Since LSR8 is reachable via LSR4, and since the outgoing interface for LSR4 is LSP1, the outgoing interface for all the destinations reachable via LSR8 is LSP1. Finally, SPF adds LSR5 to its shortest path tree. At this point LSR1 notices that there is an LSP, LSP2, that starts at LSR1 and ends at LSR5. Therefore, the outgoing (logical) interface for the destinations reachable via LSR5 would be LSP2.



All the traffic with destinations reachable via LSR5 and beyond will be forwarded along the path taken by LSP2. All the traffic with destinations reachable via LSR4 and beyond will be forwarded along the path taken by LSP1. The rest of the traffic will be forwarded via the paths computed by normal IGP procedures.

## 6.16 Application to Fast Rerouting

### Rerouting Upon Failure:

When a link goes down, it is important to *reroute* all the trunks that were routed over the link. Since the path taken by a trunk is determined by the LSR at the head end of the trunk, rerouting has to be performed by the head end of the trunk as well. MPLS constraint-based routing can be used to "locally" repair a failed link, without involving the head ends of all the LSPs traversing that link in rerouting those LSPs.

### Routing Convergence with Plain IP Routing:

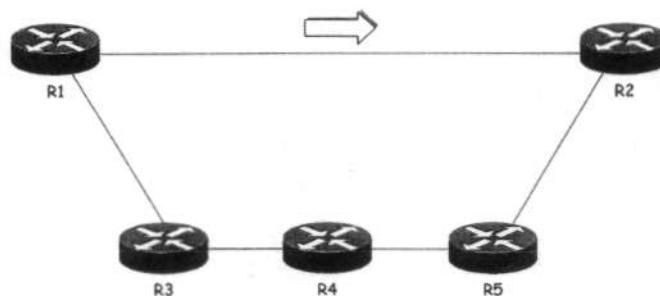
*Routing Transient* is a term used to refer to episodes in a network where routing information is changing. These episodes most commonly occur as a result of failures of links or routers or both. At such times, the routing information stored at different routers may be temporarily inconsistent, which may result in formation of temporary (transient) forwarding loops. With plain IP routing, the longer the routing transient lasts, the more packets are likely to be discarded. With plain IP routing

the duration of routing transients due to a link or node failure, and thus the amount of packet loss, depends on the following factors:

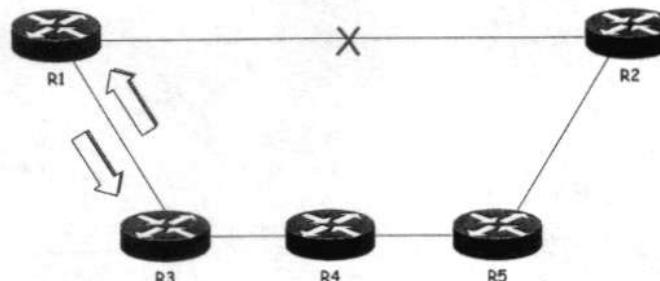
- the time it takes for a router adjacent to the failed link (or node) to detect the failure,
- the time it takes to distribute this information among all the routers,
- the time it takes for the routers to recompute their forwarding tables.

To reduce the time it takes to *detect link failure*, we can use link layer-specific mechanisms. For example, with SONET it is possible to detect link failure in less than 10 ms. However, the time it takes to distribute the information about the failure and recompute the forwarding tables within a single OSPF routing domain is on the order of seconds. Thus within a network composed of such a domain, a link or a node failure, even if detected instantaneously, may cause packet losses that could last on the order of seconds. To reduce packet losses during routing transients, you might attempt to reroute the traffic around the failed link or node along some alternative path. This approach doesn't always work.

*Example:* When the link between R1 and R2 fails, if R1 somehow knows that there is an alternative path to R2 via R3, R4, and R5 with R3 as the next hop, then as soon as R1 detects the link failure, R1 could immediately start forwarding all the traffic destined for R2 to R3, with the hope that this traffic would be delivered to R2.



With link-state protocols (e.g., OSPF), R1 could certainly find out that there is an alternative path via R3, R4, and R5. However, when R1 starts to forward the traffic to R3, until R3 receives the information about the link failure and updates its forwarding table, R3 would forward this traffic back to R1, thinking that R1 is only one hop from R2.

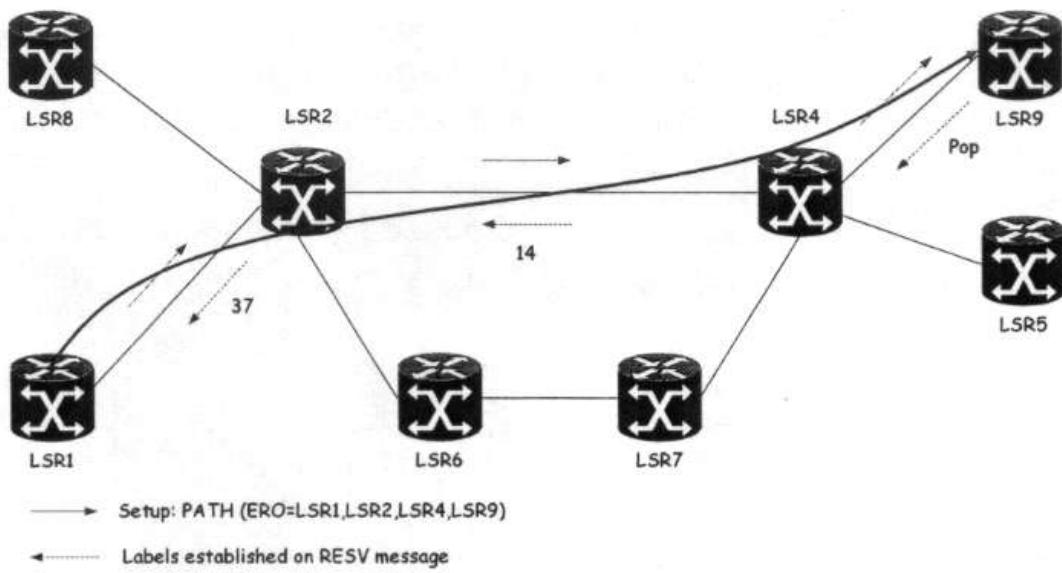


Thus, for some period of time, there would be a (*transient*) forwarding loop, which would result in packet losses. This illustrates that, with plain IP routing, forwarding along an alternative path doesn't eliminate the need for routing convergence among the routers and thus doesn't help to reduce packet losses due to link or node failures. To summarize, the fundamental limitation on how much we could reduce packet losses during routing transients is imposed by the need to distribute the information about a link or node failure among a set of routers that then must process this information and converge to a new set of routes. This limitation, in turn, is caused by the fundamental nature of plain IP routing-independent, hop-by-hop, destination-based forwarding.

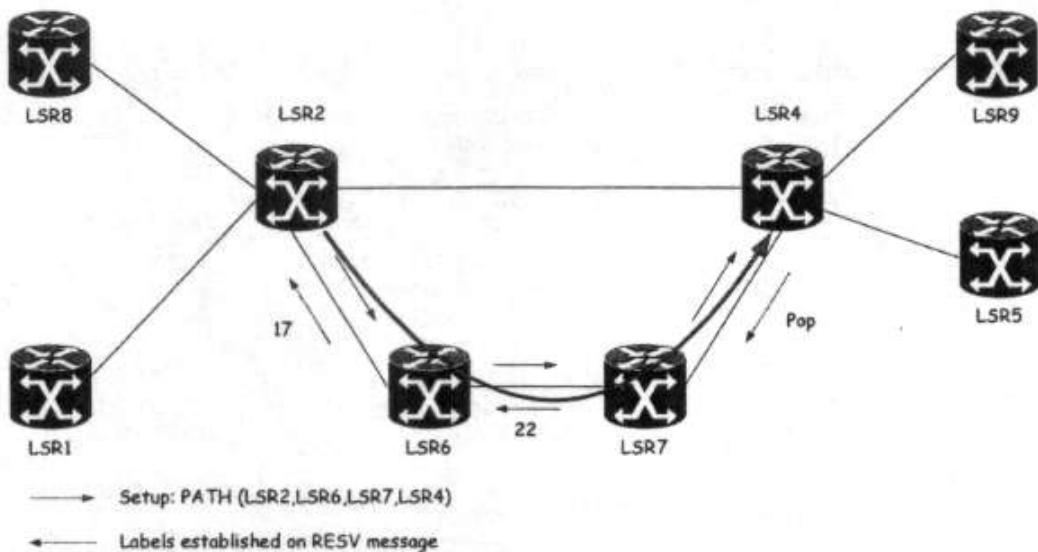
#### Fast Reroute with Constraint-Based Routing:

To handle link failures, MPLS constraint-based routing can be used to construct a "*protection*" LSP around a link. When the link fails, the LSR attached to the failed link uses the label stacking capability of MPLS to "nest" all the LSPs that used to go over the failed link into the protection LSP. The protection LSPs are unidirectional, just like any other LSP.

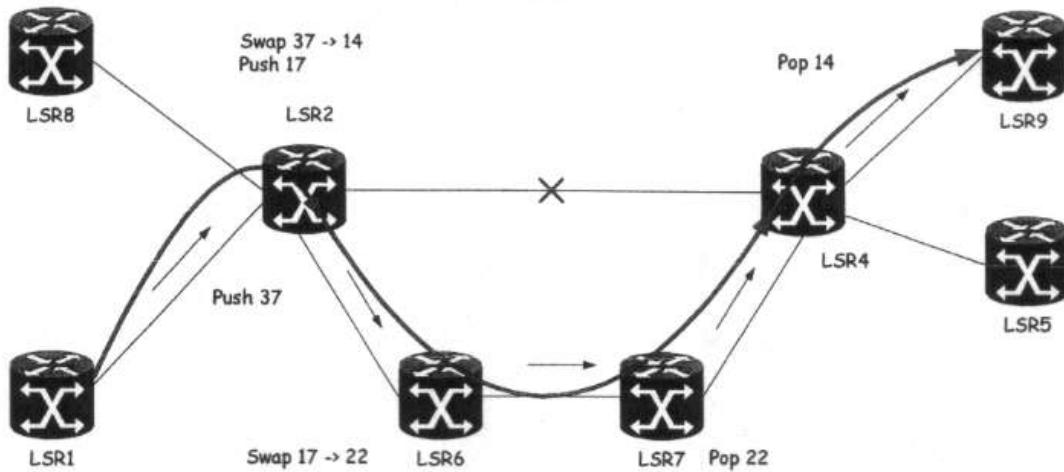
Assume that there is an LSP from LSR1 to LSR9 that is routed over LSR2 and LSR4.



At LSR2 the label forwarding table entry for that LSP has 37 as the incoming label, 14 as the outgoing label, and the interface that connects LSR2 to LSR4 as the outgoing interface. When LSR2 receives a packet routed along that LSP (the packet arrives at LSR2 with label 37), LSR2 replaces label 37 with label 14 and sends the packet over the interface that connects LSR2 to LSR4. To handle failure of the link between LSR2 and LSR4, we construct a protection LSP around that link. This LSP starts and ends at the LSRs connected to the link and is routed along an explicit route (LSR2, LSR6, LSR7, L5R4). To forward a packet along that LSP, LSR2 has to place label 17 on the packet and send the packet on the interface that connects LSR2 to LSR6.



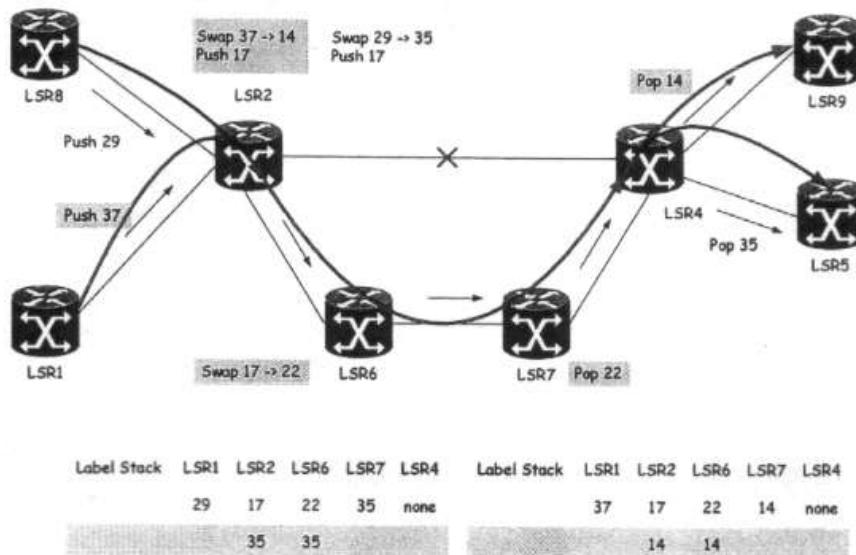
When LSR2 detects that the link (LSR2, LSR4) has gone down, LSR2 changes the label forwarding table entry associated with the LSP as follows.



Label Stack	LSR1	LSR2	LSR6	LSR7	LSR4
	37	17	22	14	none
		14		14	

In addition to swapping label 37 with label 14, the entry now indicates that label 17 has to be pushed onto the label stack, and the outgoing interface is changed from the interface that connects LSR2 to LSR4 to the one that connects LSR2 to LSR6. Thus, when LSR2 receives a packet with label 37, LSR2 replaces label 37 with label 14 (just as it did before the link failure), pushes one more label, label 17, onto the label stack carried by the packet, and sends the packet over the interface that connects LSR2 to LSR6 (rather than sending the packet over the interface that connects LSR2 to LSR4). The label that LSR2 pushes onto the label stack is the label associated with the protection LSP, the LSP from LSR2 to LSR4. When the packet arrives at LSR4, it carries exactly the same label, label 14, that it carried when the link (LSR2, LSR4) was up. From that point on, the fact that the link went down has no impact on the forwarding of the packet to the rest of the LSP.

The mechanism we described above may result in suboptimal forwarding. However, this potentially suboptimal forwarding is of short duration. When a link between LSR2 and LSR4 failed, the information about this failure will be distributed (via OSPF) to all the LSRs, including LSR1. Once LSR1 gets this information, it can use constraint-based routing to compute a new route and then establish label forwarding state along the newly computed route. A single protection LSP could be used to fast-reroute not one, but multiple LSPs since MPLS provides the ability to nest LSPs via the label stacking mechanism.



In the example, if we assume that there is also an LSP from LSR8 to LSR5 that is (in a steady state) routed over LSR2 and LSR4, then the same protection LSP could be used to fast-reroute both the LSP from LSR1 to LSR9 and the LSP from LSR8 to LSR5. The ability to use a single protection LSP to fast-reroute multiple LSPs makes this approach scalable, as the number of protection LSPs (and therefore resources needed to establish and maintain them) is independent

of the number of LSPs that require fast reroute capabilities, but depends solely on the number of links you want to protect.

*Question:* what are the essential differences between the plain IP approach and the MPLS approach such that the former doesn't work, while the latter does? The two fundamental reasons why fast rerouting based on MPLS constraint - based routing works are:

- Path computation is not distributed,
- Forwarding is not destination-based.

Using the example above, with MPLS constraint-based routing, the protection path for the link (LSR2, LSR4) is determined (and controlled) solely by LSR2. In contrast, with plain IP routing, forwarding along the alternate path is controlled not just by LSR2 but by all other LSRs along that path. This example can be generalized to cover not just link, but router failure as well. The first essential difference between providing fast reroute in the presence of link failure and in the presence of router failure is that in the former the protection LSP starts/ends at the LSRs of a given link that are one hop away from each other, while in the latter the protection LSP starts/ends at the LSRs that are two hops away from each other. The second essential difference is that in the former case existing link layer mechanism(s) can be used to detect link failure, while in the latter case we need to develop a new mechanism (such as a hello protocol) to detect router failure.

## 6.17 Application to QoS

Provision of QoS guarantees along LSPs selected by constraint-based routing.

### Relationship Between QoS and Routing:

QoS is often viewed as being decoupled from routing, especially in the IP networking community. The design approach of RSVP, for example, was based on the idea that RSVP would make resource allocation requests along whatever paths routing picked as the "best". A similar approach prevails in the current definition of Differentiated Services, where resource allocation decisions are made independently of routing. In reality there is a close relationship between routing and QoS, and something is lost when we ignore this relationship.

*Example:* suppose there are two alternative paths from node A to node B and that routing decides one of these paths is the best. Now suppose we start to make resource allocation requests for traffic flowing from A to B along the best path and that at some point, that path will run out of resources. As long as routing is separated from QoS, it will continue to favor this path and attempts to reserve resources from A to B will fail, in spite of the fact that resources are available on the alternate path. The only way out of this situation is:

- for routing to have some knowledge of resources, and
- for QoS requests to be made over paths that are appropriate for those requests.

One way to think about the relationship between QoS and routing is as follows:

- In order to provide a QoS guarantee along a path between two nodes, we need to allocate some resources (e.g., buffer space, link bandwidth) along that path.
- Queuing is the set of mechanisms that provides control over local resources at each node on the path, while routing, by selecting the path, provides control over whose (local) resources are used.

### Guaranteed Bandwidth LSPs:

We saw how RSVP could easily be extended to distribute labels as part of the resource reservation process, thus allowing LSPs to be established with reserved resources. While this provides some advantages over the conventional use of RSVP to reserve resources, it does not address the relationship between routing and QoS. In the absence of some extra help from routing, RSVP PATH messages follow the shortest (lowest-cost) path toward their destination, and thus an LSP established in this way will also lie on the shortest path. The extensions to RSVP described before (see *ERO*) can be used when establishing an LSP with reserved resources which is called "*guaranteed bandwidth LSP*". Example: suppose that we wish to establish a reservation along a path from node A to node B, both nodes being in the same routing area. Before sending an RSVP PATH message from A to B, node A consults the link-state database and selects a path from A to B that meets the following constraint: it must have enough available bandwidth on every link to support the desired reservation. Upon obtaining such a path, A inserts an Explicit Route Object (ERO) into the PATH message, thus ensuring that the LSP will be established along the selected path. Note that this process does not guarantee that a reservation attempt will succeed, even if the constraint-based routing system is able to find a path from A to B meeting the stated constraint. For example, each node along the path needs to have enough buffer space to accommodate

bursts for the reserved traffic flow, and a lack of buffer space at one node might cause admission control to fail even on a path that has enough bandwidth. Of course, you might avoid such problems by using both buffer space and bandwidth as constraints when selecting the path. How is this process different from the application of constraint-based routing to traffic engineering? The pivotal difference is that traffic engineering does not require all the queuing machinery that is required to deliver QoS guarantees. When we establish a traffic engineering trunk between two points, it is not necessary to police the offered load entering the LSP or to dedicate any queuing resources to the trunk. To provide QoS guarantees on an LSP, however, does require:

- policing at the LSP ingress, as well as
- appropriate scheduling of link resources and
- allocation of buffer space at routers along the path.

Thus, from a signaling point of view, establishing a traffic engineering trunk and setting up a guaranteed bandwidth LSP are similar processes; the difference lies in how we allocate resources and treat traffic at routers along the LSP.

# 7 Virtual Private Networks (VPN)

## 7.1 References

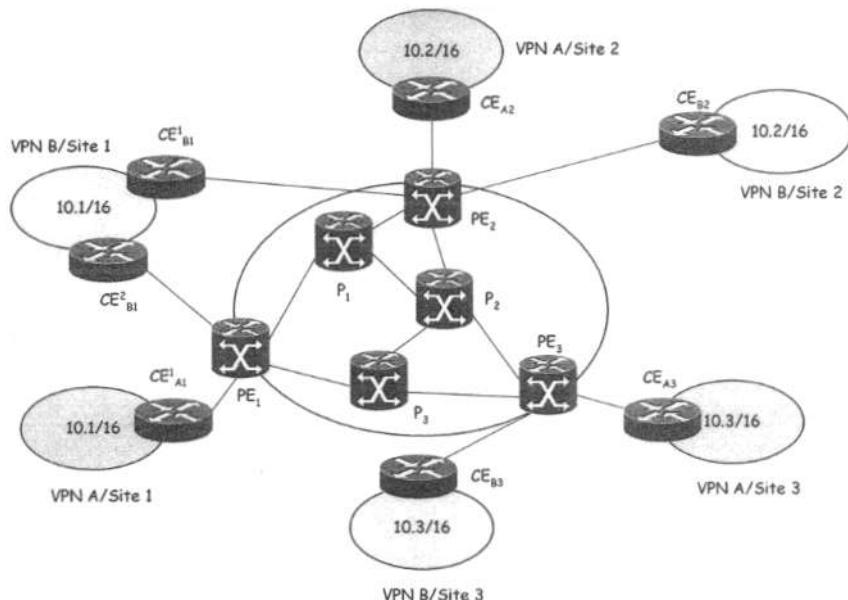
- “MPLS Technology and Applications” by Bruce Davie & Yakov Rekhter, *Morgan Kaufmann Publishers*.
- “MPLS and VPN Architectures” by Ivan Pepelnjak & Jim Guichard, *Ciscopress.com*.

## 7.2 What is a VPN?

Consider a company that has a set of geographically dispersed sites. To interconnect computers at these sites, the company needs a network. The network is *private* in the sense that:

- it is expected to be used by only that company, and
- the routing and addressing plan within the network is completely independent of the routing and addressing plans of all other networks sharing the same infrastructure.

The network is *virtual* in the sense that the facilities used to build and operate such a network may not be dedicated just to that company, but could be shared with other companies that want to have their own VPNs. The facilities needed to build such a network are provided, at least partially, by a third party, known as a *VPN service provider*. The company that uses the network is known as a *VPN customer*.



Thus, a VPN is defined loosely as a network in which customers connectivity among multiple sites is deployed on a shared infrastructure with the same access or security policies as a private network. More formally, a VPN is defined by:

*“a set of administrative policies that control both connectivity and Quality of Service among sites”*

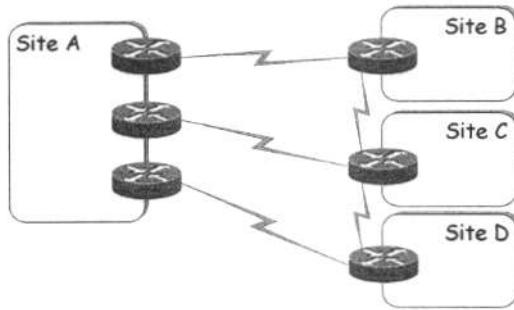
It is the VPN customer who sets up the policies that define that customer's VPN. With respect to who implements these administrative policies, depending on the technology used, there could be a range of choices, i.e. it should be possible:

- to completely confine the implementation of these policies to the VPN service provider, so that the VPN customer could completely outsource the VPN service to the service provider;
- to distribute the implementation of these policies between the VPN service provider and the customer.

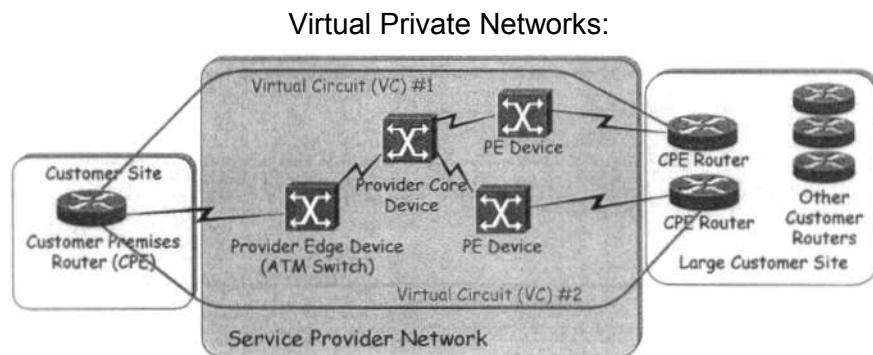
## 7.3 VPN Technology

Traditional router-based networks connect customer sites through routers connected via dedicated point-to-point links.

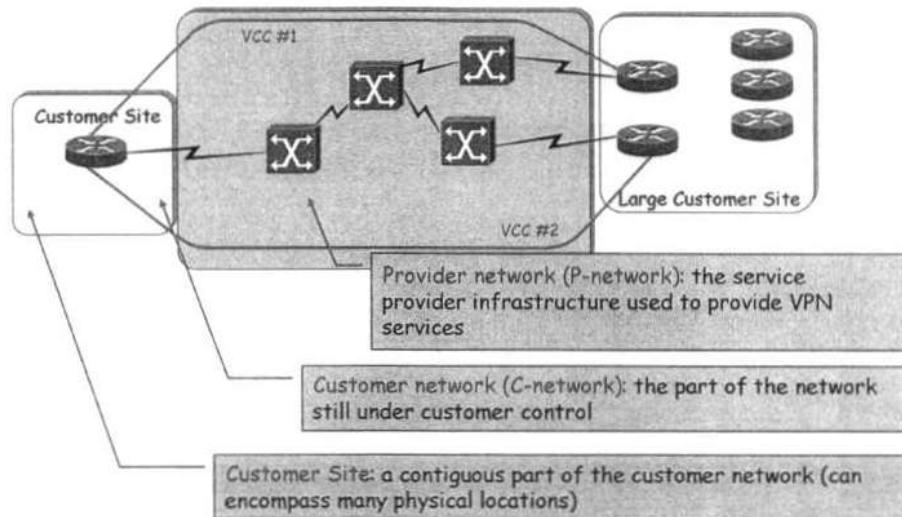
Traditional router-based networks:

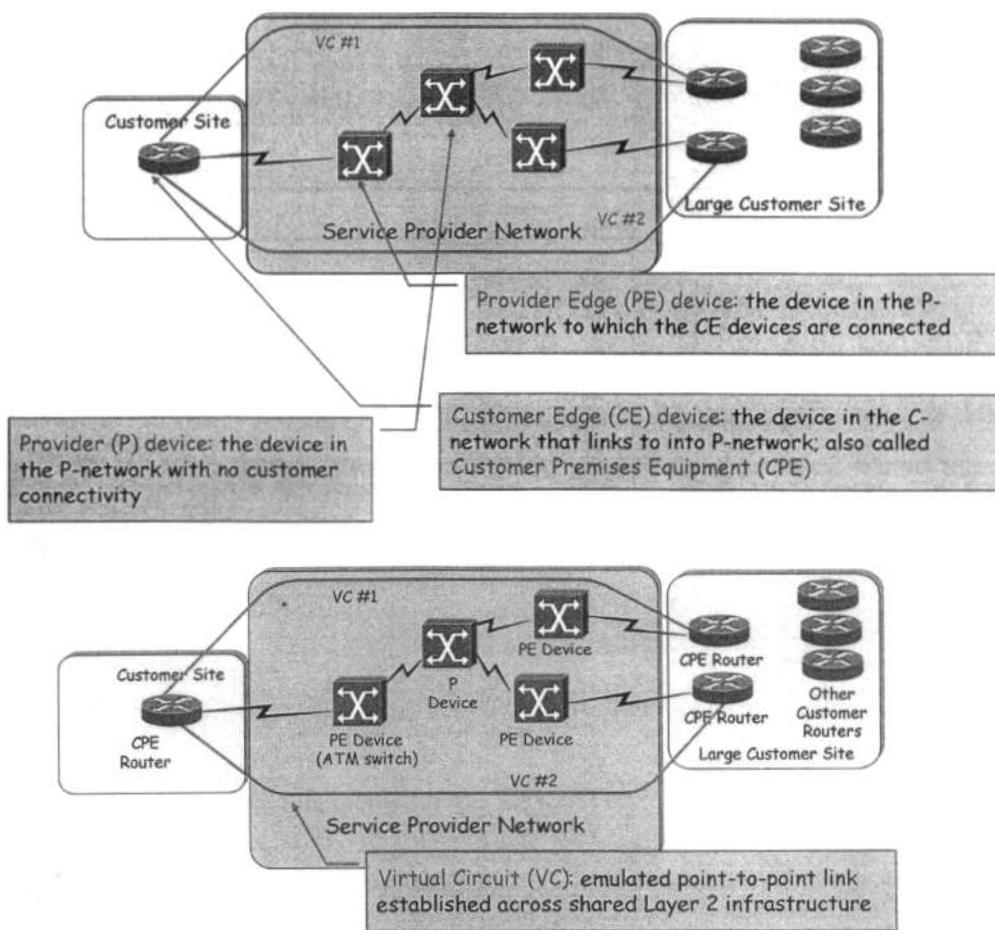


Virtual Private Networks (VPNs) replace dedicated point-to-point links with emulated point-to-point links sharing common infrastructure. Customers use VPNs primarily to reduce their operational costs.



## 7.4 VPN Terminology





A *Permanent Virtual Circuit (PVC)* is established through out-of-band means (network management) and is always active. A *Switched Virtual Circuit (SVC)* is established through CE-PE signaling on demand from the CE device.

## 7.5 VPN Implementation Technologies

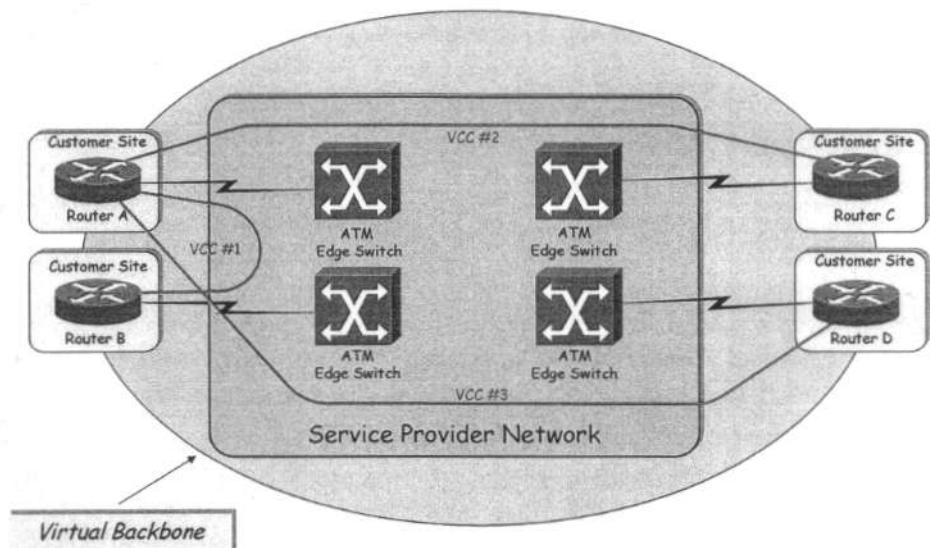
VPN services can be offered based on two major paradigms:

- *Overlay VPNs*, in which the service provider provides virtual point-to-point links between customer sites;
- *Peer-to-Peer VPNs*, in which the service provider participates in the customer routing.

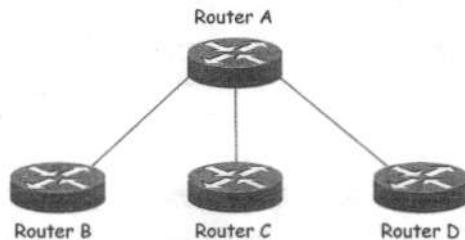
## 7.6 Overlay VPN

With the *Overlay Model* each site has a router that is connected via point-to-point links to routers in other sites. A site may have one or more such routers, which may be connected to all other sites or to only a subset of all other sites. The technology used to offer point-to-point links could be leased lines, Frame Relay circuits, or ATM circuits. The network that consists of these point-to-point links and the routers attached to these links is called “*virtual backbone*” (It is this virtual backbone that provides connectivity among sites).

Sample Overlay VPN Network (with ATM):



Layer 3 Routing in Overlay VPN Implementation

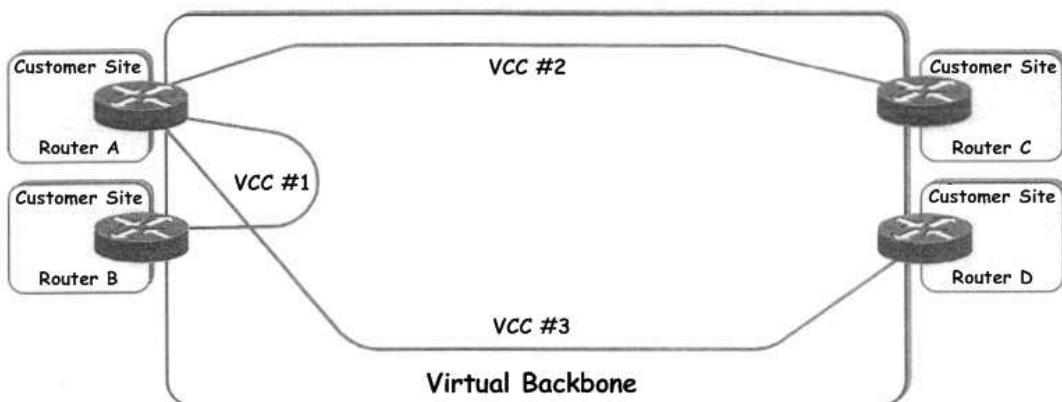


Service provider infrastructure appears as point-to-point links to customer routes. Routing protocols run directly between customer routers. Service provider does not see customer routers and is responsible only for providing point-to-point transport of customer data.

## 7.7 Overlay Model Problems

*First problem:* it comes from the need for a VPN customer to design and operate its own virtual backbone and this implies a certain minimum amount of expertise in IP routing, which is not that common. As a result, not too many companies could use a VPN service that requires each company to design and operate its own virtual backbone.

First problem:



*Second problem:* in an attempt to solve the first problem, VPN service providers offer what is known as a “*managed router*” service, whereby the service provider designs and operates a virtual backbone for each of its VPN customers. However, while solving one problem, this approach introduces another, as it requires a service provider to design and operate a virtual backbone for each of its VPN customers. This requirement makes it difficult for the service provider to offer services to a large number of customers (consider a service provider that has 100.000 VPN customers and therefore has to design and operate 100.000 different virtual backbones, one for each customer). This requirement also makes the service rather costly to the service provider, as

designing and operating a virtual backbone is a labor-intensive job. Inability to support a large number of customers is just one of the problems with the overlay model. So the managed router service doesn't really solve the problem.

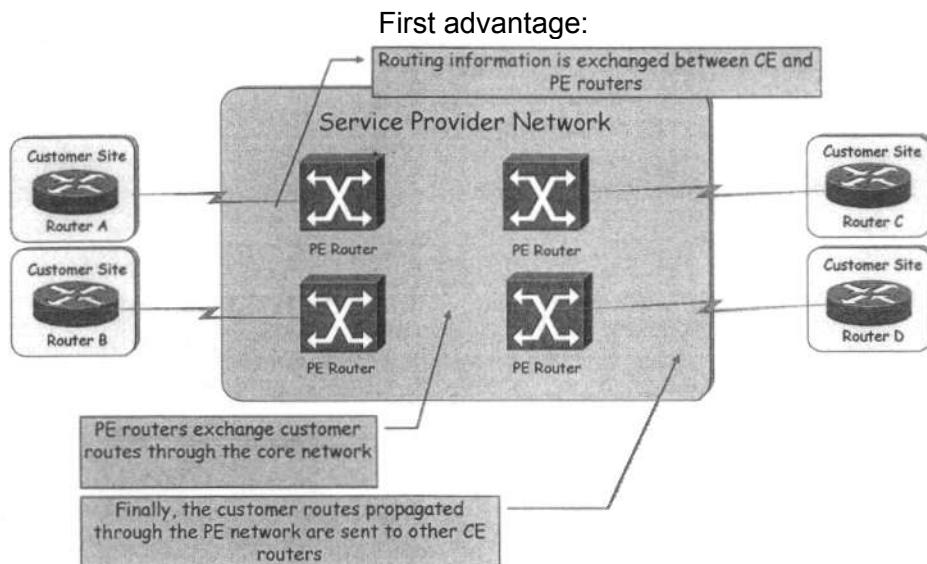
*Third problem:* it comes from customers who have a relatively large number of sites (100 and more) and require *complete mesh connectivity* among these sites. For these customers a backbone router within a given site would require a routing peering with the backbone router in every other site. So, for a VPN of  $N$  sites, a backbone router would need  $N-1$  routing peerings. The problem with that number of peerings is precisely the same problem as running IP over ATM with the overlay model.

*Fourth problem:* the amount of configuration changes required when adding a new site to an existing VPN. For a VPN that requires full mesh connectivity among its sites, adding a new site involves changes to the configuration on all of the existing sites, as each one would need an additional point-to-point connection to the new site and an additional routing peering with the router in the new site.

## 7.8 Peer-to-Peer VPN

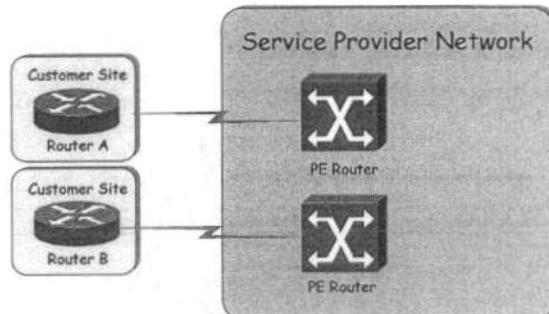
The *peer-to-peer VPN model* was introduced a few years ago to alleviate the drawbacks of the overlay VPN model. In the peer-to-peer model, the PE device is a router (PE router) that directly exchanges routing information with the CE router. The peer-to-peer model provides a number of advantages over the traditional overlay model.

*First advantage:* routing (from the customer's perspective) becomes exceedingly simple, as the customer router exchanges routing information with only one (or a few) PE router, whereas in the overlay VPN network, the number of neighbor routers can grow to a large number.



*Second advantage:* The addition of a new site is simpler because the service provider provisions only an additional site and changes the configuration on the attached PE router. On the other hand, under the overlay VPN model, the service provider must provision a whole set of VCs leading from that site to other sites of the customer VPN.

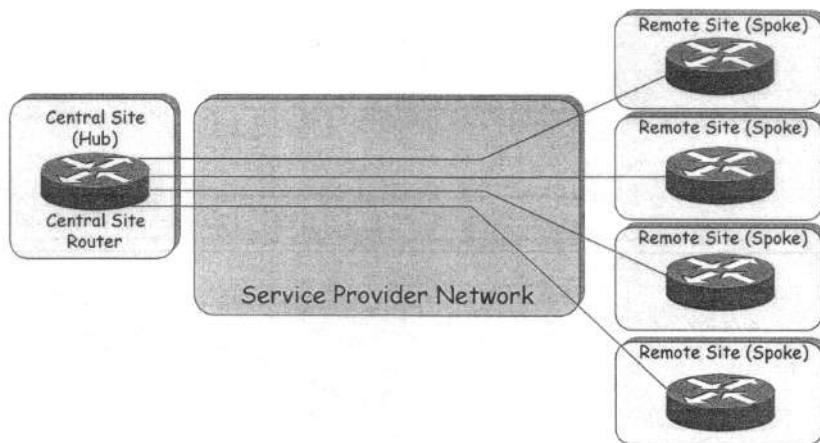
Second advantage:



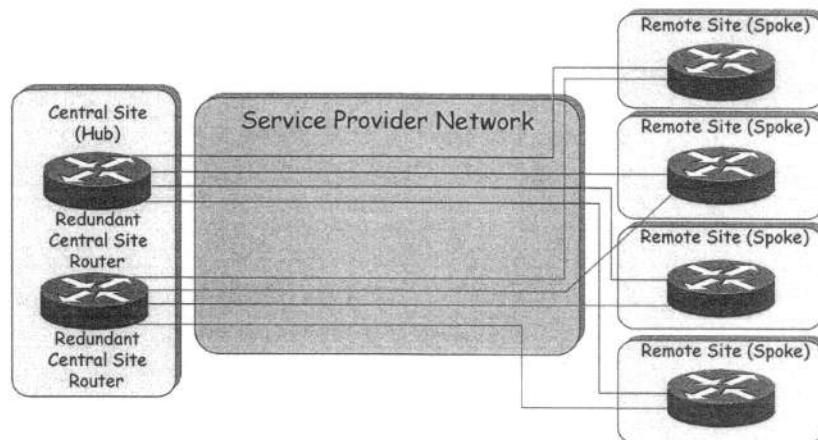
## 7.9 VPN Topology Categorization

The VPN topology required by an organization should be dictated by the business problems the organization is trying to solve. However, several well-known topologies appear so often that they deserve to be discussed here. The VPN topologies discussed here are those topologies influenced by the overlay VPN model, which include *hub-and-spoke* topology, *partial* or *full-mesh* topology. At one end of the spectrum is complete mesh connectivity among all the sites, while at the other end of the spectrum is hub-and-spoke connectivity, where connectivity among the sites labeled as “spokes” occurs only through the site labeled as the “hub”. Yet another example of intersite connectivity is when the sites are partitioned into two or more sets; in this case connectivity among sites within each set is a complete mesh, whereas connectivity between sites in different sets is indirect, only through a particular site.

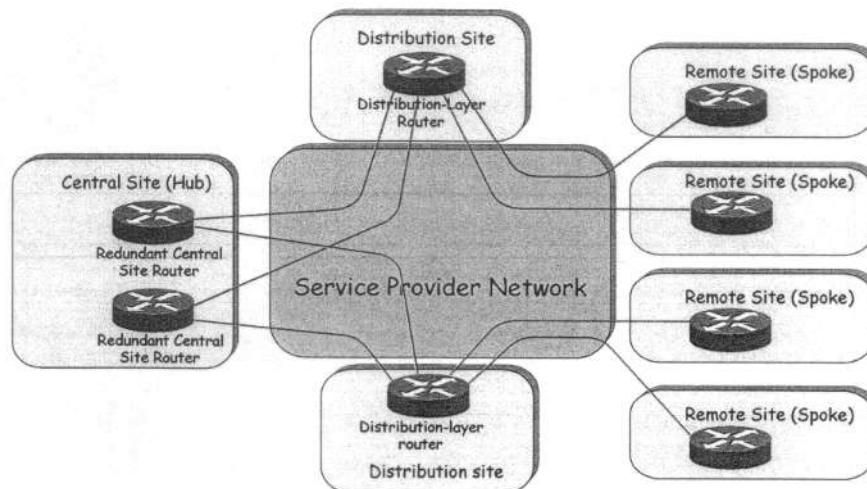
Hub-and-Spoke Topology:



Redundant Hub-and-Spoke Topology:

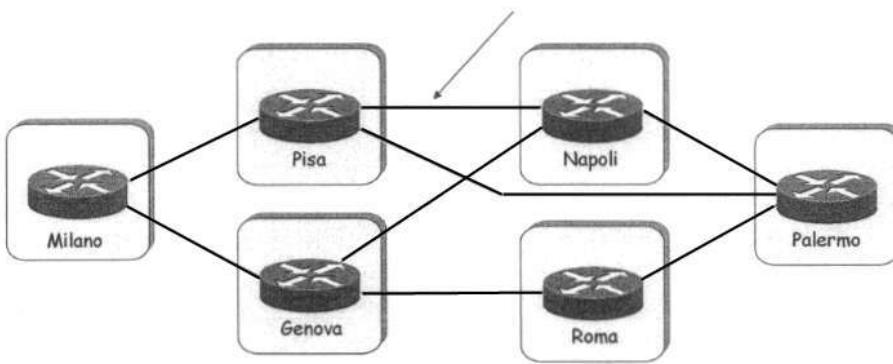


Multilevel Hub-and-Spoke Topology:



Partial Mesh Topology:

Virtual Circuits (ATM, Frame Relay, etc.)



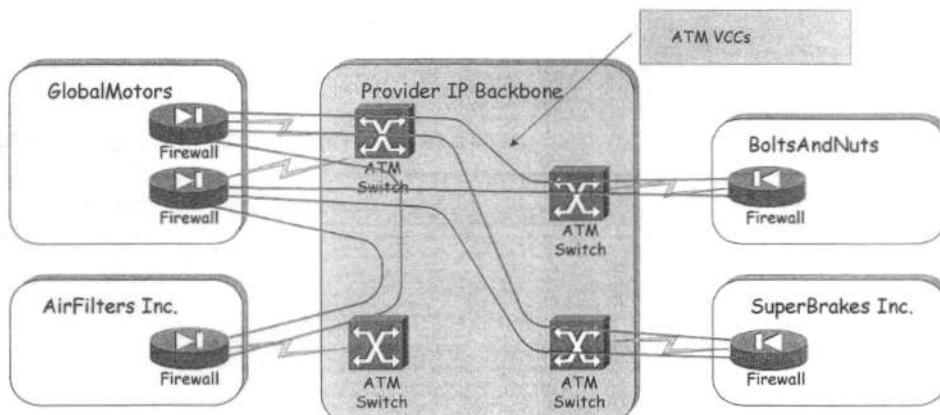
## 7.10 VPN Business Categorization

In addition to VPNs that span sites that belong to the same company, there are more and more cases where VPNs are also used to interconnect sites in different companies. The common name for the former is an *intranet* and for the latter is an *extranet*. Thus, VPNs can be categorized on the business needs they fulfill:

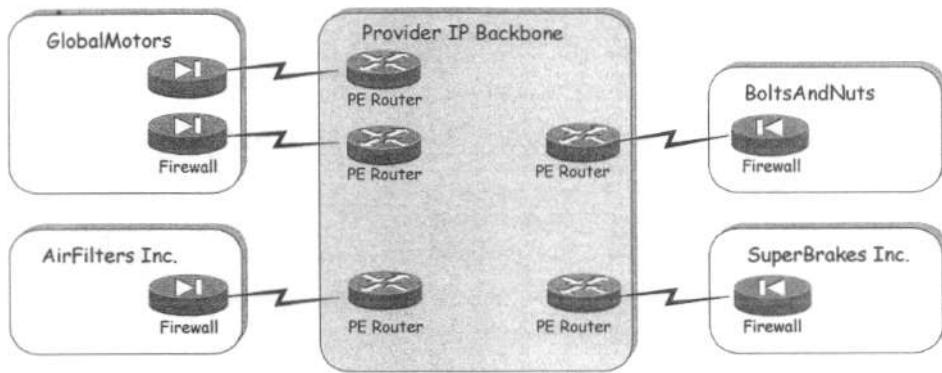
- *Intranet VPN* - connects sites within an organization;
- *Extranet VPN* - connects different organizations in a secure way.

The distinction between the two cases is in who sets up the policies that define the VPN - in the case of an intranet it is a single company (or administrative entity) while in the case of an extranet it is a group of companies.

Extranet VPN – Overlay VPN Implementation:



## Extranet VPN – Peer-to-Peer VPN Implementation

**7.11 Additional VPN Features**

The above VPN definition allows a given site to be part of more than one VPN. For example, a particular site may be in a VPN associated with an intranet and at the same time, it could also be in a different VPN associated with an extranet (in that sense VPNs could overlap). *It is not assumed that a given VPN has to be confined to a single VPN service provider - facilities needed to build and operate a VPN could be provided by a group of VPN service providers.*

**7.12 Peer-to-Peer VPN Model**

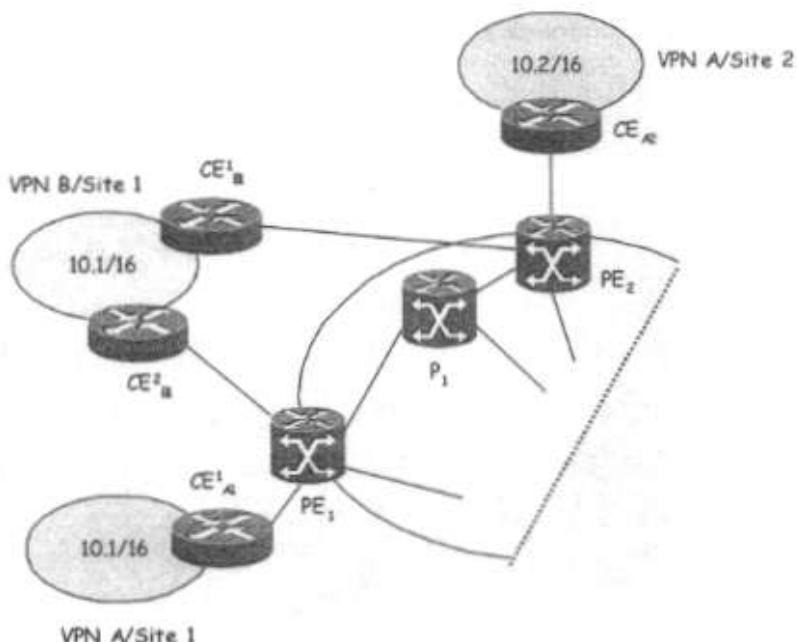
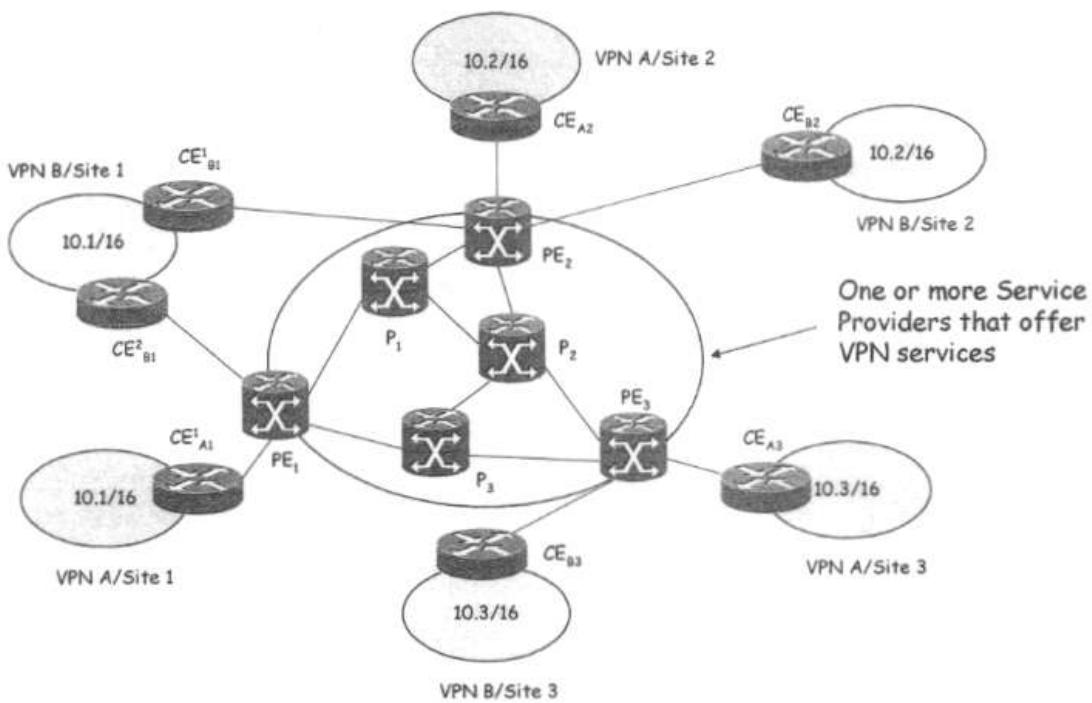
This model aims at enabling VPN service providers:

- to support very large-scale VPN service offerings (thousands to millions of VPNs per service provider), where most of the customers have little or no IP routing expertise;
- to support a diverse population of customers, where some VPNs would consist of just a few sites while others would have hundreds or even thousands of sites per VPN;
- to keep the cost of offering the VPN service low.

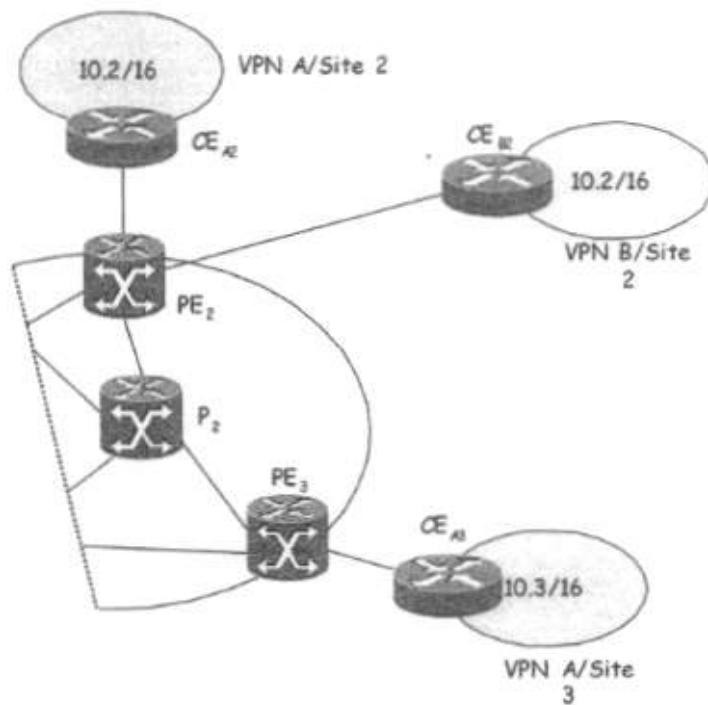
**7.13 MPLS/VPN**

With the introduction of MPLS, it became possible to construct a technology that combines the benefits of an overlay VPN (such as security and isolation among customers) with the benefits that a peer-to-peer VPN implementation brings. In the following we will focus on the *MPLS/VPN architecture*. To support connectivity requirements of a VPN, the MPLS/VPN architecture supports the concept of sites. A VPN is essentially a collection of sites sharing common routing information, where a site may belong to one or more than one VPN if it holds routes from separate VPNs. A VPN in the MPLS/VPN architecture can therefore be pictured as a community of interest or a closed user group, which is dictated by the routing visibility that the site will have.

MPLS/VPN Example 1:



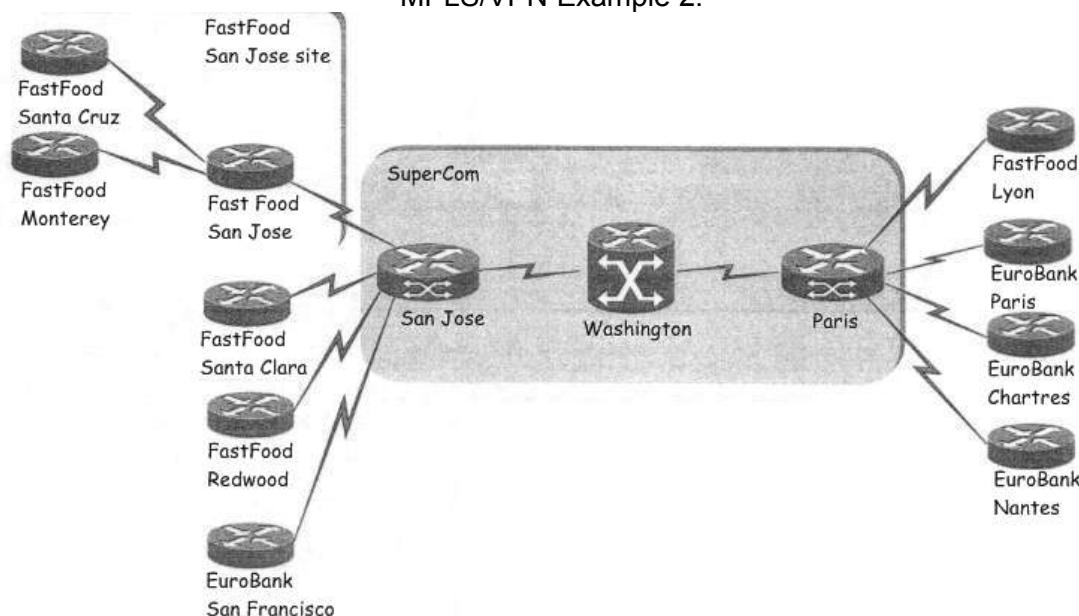
A site may have one or more *Customer Edge (CE)* routers. Site 1 in VPN B has two CE routers,  $CE^1_{B1}$ , and  $CE^2_{B1}$ , while Site 1 in VPN A has only one CE router,  $CE_{A1}$ . There is a set of destinations reachable within each site. The set of destinations reachable within Site 2 of VPN A is covered by an IP address prefix 10.2/16. On the service provider side, a CE router is connected to a *Provider Edge (PE)* router but a single PE router may be connected to sites that belong to the same or different VPNs; moreover, these sites may use the same IP addresses for destinations within the sites (*overlapping addresses*). A site may be connected to more than one PE router. Site 1 of VPN B is connected to  $PE_1$ , and also connected to  $PE_2$ .



$PE_2$  is connected to sites that belong to VPN A (Site 2) and VPN B (Site 2). Both Site 2 in VPN A and Site 2 in VPN B use the same block of addresses, 10.2/16, for destinations internal to these sites (overlapping addresses). The third type of router in this figure is the Provider ( $P_2$ ) router, which does not connect to customer sites.

The reason for calling this a "peer" model is that, from the routing point of view, the service provider network acts as a peer of the customer networks, since the customer routers peer directly with provider routers. This is in contrast to the overlay model, where customer routers peer only with other customer routers over layer 2 links or IP tunnels provided by the service provider.

MPLS/VPN Example 2:



MPLS/VPN Example 2, address space of FastFood and EuroBank:

Company	Site	Subnet
FastFood	San Jose	192.12.2.0/24
	Santa Clara	10.1.1.0/24
	Redwood	10.1.2.0/24
	Santa Cruz	10.1.3.0/24
	Monterey	10.1.4.0/24
	Lyon	10.2.1.0/24
EuroBank	Paris	196.7.25.0/24
	Chartres	10.2.1.0/24
	Nantes	10.2.2.0/24
	San Francisco	10.1.1.0/24

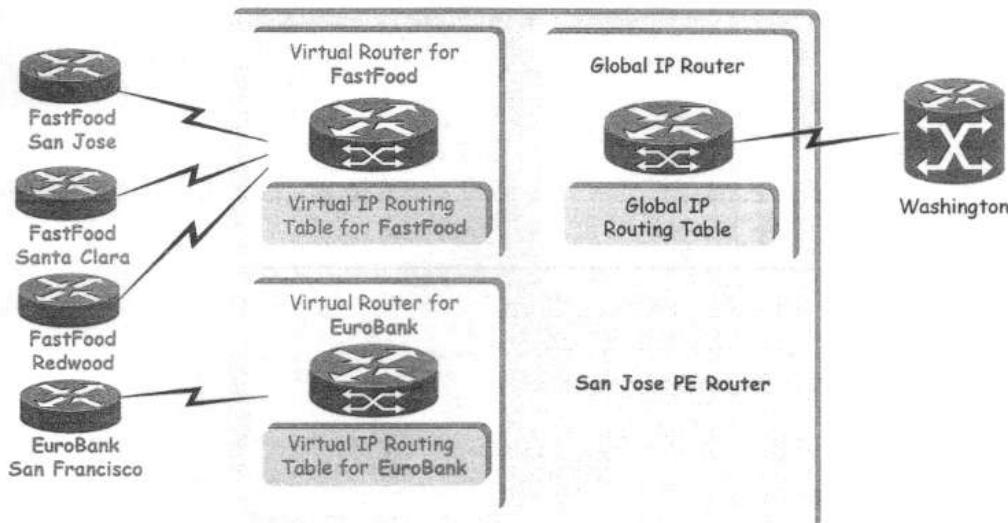
## 7.14 Multiple VRFs

The MPLS/VPN solution is built around several key concepts:

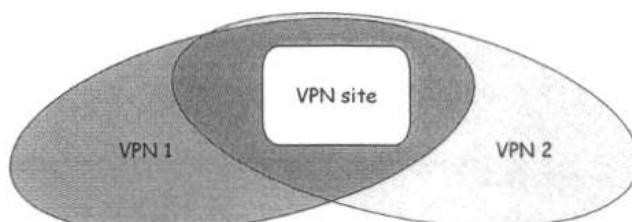
- multiple forwarding tables;
- use of a new type of addresses, VPN-IP addresses;
- MPLS forwarding capabilities.

The overlapping addresses, usually resulting from usage of private IP addresses in customer networks, are one of the major obstacles to successful deployment of peer-to-peer VPN implementation. The MPLS/VPN architecture provides an elegant solution to the dilemma: each VPN has its own Routing and Forwarding table in the PE router, so any customer or site that belongs to that VPN is provided access only to the set of routes contained within that table. Any PE router in an MPLS/VPN network thus contains a number of per-VPN routing tables and a global routing table that is used to reach other routers in the provider network as well as external globally reachable destinations (for example, the rest of the Internet). Effectively, a number of virtual routers are created in a single physical router, as displayed in the next slide for the case of San Jose router of SuperCom network.

Virtual routers created in a PE router:



The concept of virtual routers allows the customers to use either global or private IP address space in each VPN. Each customer site belongs to a particular VPN, so the only requirement is that the address space be unique within that VPN. Uniqueness of addresses is not required among VPNs except where two VPNs that share the same private address space want to communicate.



The combination of the VPN IP routing table and associated VPN IP forwarding table is called *VPN Routing and Forwarding (VRF)* instance. In the SuperCom case, The San Jose PE router contains one VRF per customer and a global table used to forward non-VPN IP packets and to

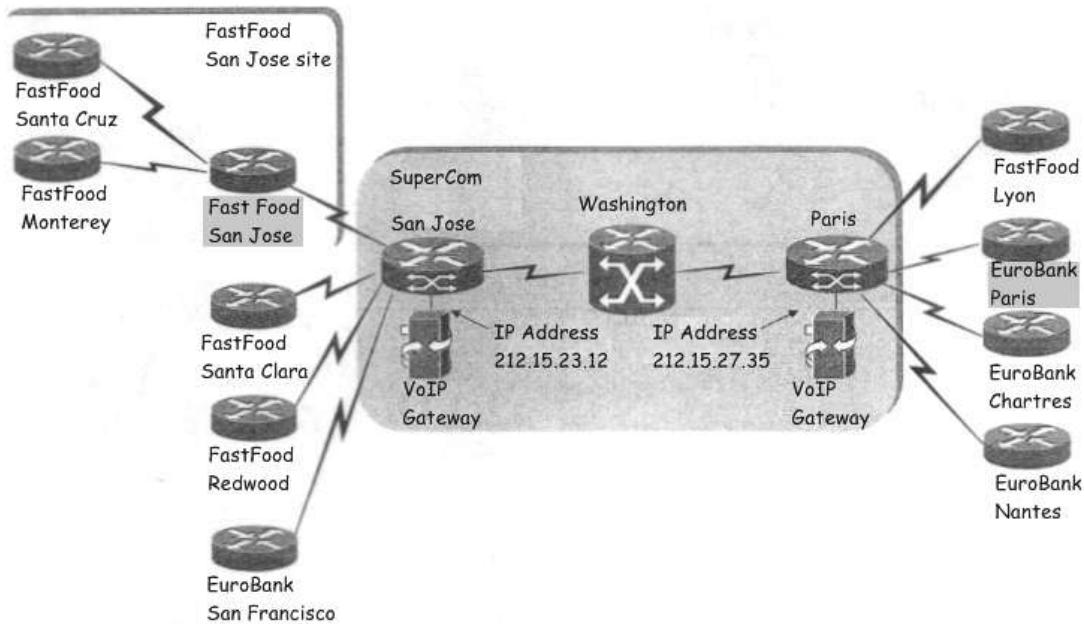
route VPN packets between PE-routers. In this example a VPN is associated with a single VRF in the San Jose PE router.

In an MPLS environment, the only minor difference between routing table and forwarding table is the fact that the IP forwarding table also contains MPLS encapsulation information. A major difference between the two tables arises in cases where an IP route refers to a next hop that is not directly connected. The *routing table* will contain the next-hop information, but not the outgoing interface or the IP address of the downstream router. The *forwarding table* will contain all the information needed to forward the packet toward the destination, i.e., the outgoing interface or the IP address of the downstream router.

## 7.15 Overlapping VPNs

In practice the situation might become more complex and require more than one VRF per VPN customer. Example: SuperCom wants to extend its service offering with a *Voice over IP* (VoIP) service with gateways to the public voice network located in San Jose and Paris. The VoIP gateways were placed in a separate VPN to enhance the security of the newly created service.

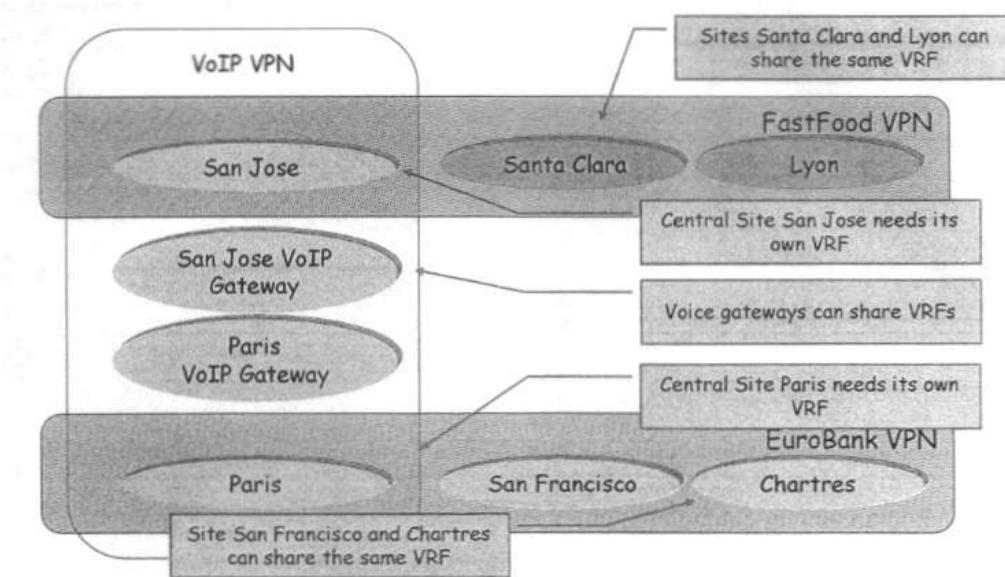
MPLS/VPN Example 3:



Both EuroBank and FastFood decided to use the service, but only from their central sites - the branch offices have no need for international voice connectivity. This requirement leads to the problem that the central sites of both organizations need to be in two VPNs:

- the corporate VPN to reach their remote sites, and
- the VoIP VPN to reach the VoIP gateways.

In this example, five different VRF tables are needed to support three VPNs. There is no one-to-one relationship between the number of VRFs and the number of VPNs.



To illustrate the requirements for multiple virtual routing tables, consider a VoIP service with three VPNs (FastFood, EuroBank, and a VoIP VPN). Five VRFs are needed to implement this service:

1. All sites of FastFood (apart from the central site) can share the same VRF because they belong to a single VPN.
2. The same for all sites of customer EuroBank (apart from the central site).
3. The VoIP gateways participate only in the VoIP VPN and can belong to a single VRF.
4. Central site FastFood has unique connectivity requirements - it has to see sites of FastFood and sites in the VoIP VPN and consequently requires a dedicated VRF.
5. Likewise, central site EuroBank requires a dedicated VRF.

*The VRF concept must support the concept of sites that can reside in more than one VPN.* For example, the central site of FastFood and EuroBank cannot use the same VRF as all other FastFood or EuroBank sites connected to the same PE router. The central site of EuroBank, for example, needs to access the VoIP gateways, so the routes toward these gateways must be in the VRF for that site, whereas the same routes will not be in the Chartres' site VRF. Therefore, the MPLS/VPN architecture unbundles the concept of VRF from the concept of VPN. The VRF is simply a collection of routes that should be available to a particular site (or set of sites) connected to a PE router. These routes can belong to more than one VPN. The relationship between the VPNs, sites, and VRFs can be summarized in the following rule, which should be used as the basis for any VRF definition in an MPLS/VPN network:

*"All sites that*

- *share the same routing information (usually this means that they belong to the same set of VPNs),*
  - *are allowed to communicate directly with each other,*
  - *are connected to the same PE router*
- can be placed in a common VRF"*

VRFs in the PE Routers in the SuperCom Network:

PE router	VRF	Sites in the VRF	VRF belongs to VPN
San Jose	FastFood Central	FastFood San Jose site	FastFood, VoIP
	FastFood	FastFood Santa Clara site	FastFood
		FastFood Redwood site	
	EuroBank	EuroBank San Francisco site	EuroBank
Paris	VoIP	San Jose VoIP Gateway	VoIP
	FastFood	FastFood Lyon site	FastFood
	EuroBank Central	EuroBank Paris site	EuroBank, VoIP
	EuroBank	EuroBank Chartres site	EuroBank
		EuroBank Nantes site	
	VoIP	Paris VoIP Gateway	VoIP

**Question:** if there is no one-to-one mapping between VPN and VRF, how does the router know which routes need to be inserted into which VRF?

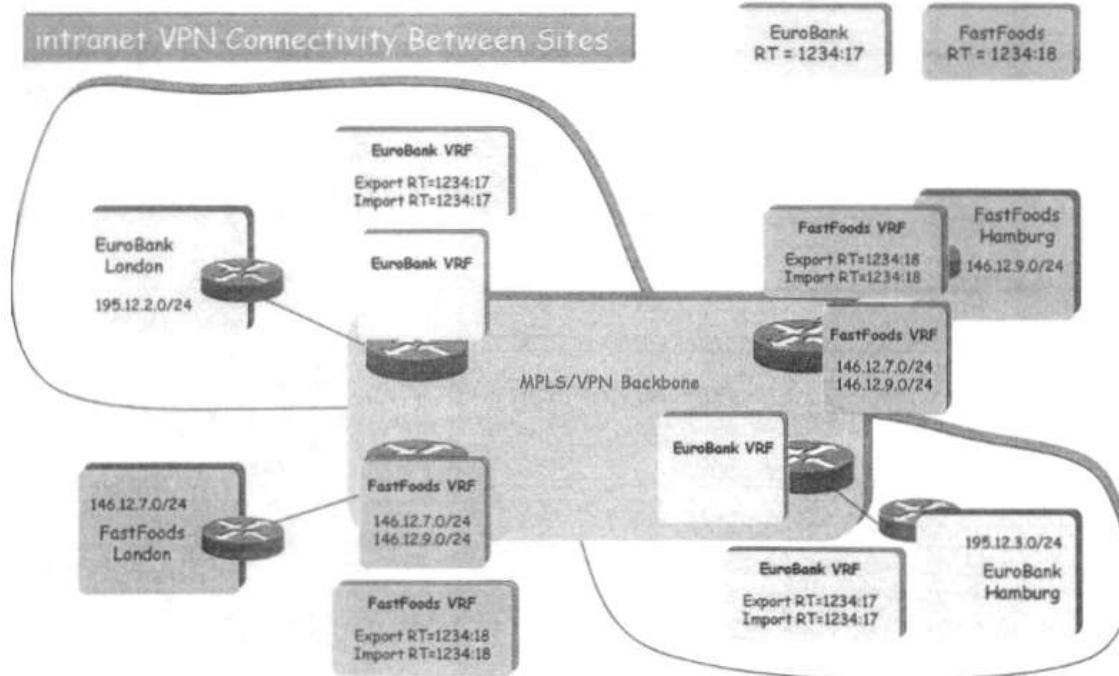
**Answer:** this problem is solved by the introduction of another concept in the MPLS/VPN architecture: the *Route Target (RT)*. Every VPN route is tagged with one or more RTs when it is exported from a VRF (to be offered to other VRFs). You can also associate a set of RTs with a VRF, and all routes tagged with at least one of those route targets will be inserted (i.e. imported) into the VRF. Thus the configuration of each VRF requires the addition of import and export policies for the VRF to use. These policies are used to populate routes into the VRF and to advertise routes out of the VRF. The route target dictates the policies used by the VRF. The route target must be configured to specify the routes (which contain this specific route target value) that are imported into the VRF, and the route target that is added to the routes that are exported from the VRF.

The SuperCom network contains three VPNs and thus requires three RTs. The association between RTs and VRFs in the SuperCom network is outlined in the following table:

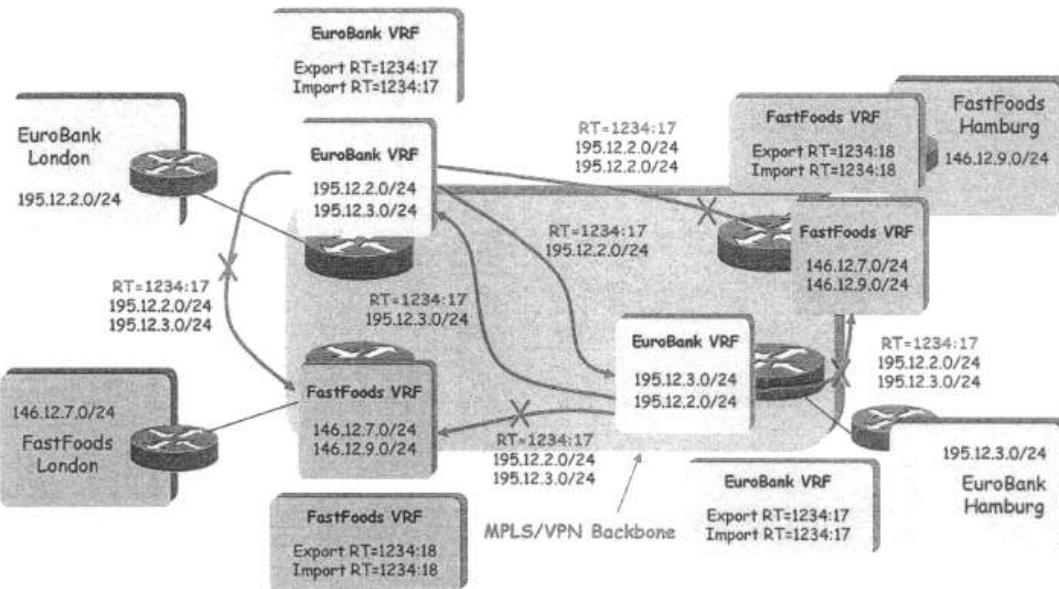
PE router	VRF	Sites in the VRF	Route target attached to exported routes	Import route targets
San Jose	FastFood Central	FastFood San Jose site	FastFood, VoIP	FastFood, VoIP
	FastFood	FastFood Santa Clara site	FastFood	FastFood
		FastFood Redwood site		
	EuroBank	EuroBank San Francisco site	EuroBank	EuroBank
	VoIP	San Jose VoIP Gateway	VoIP	VoIP
Paris	FastFood	FastFood Lyon site	FastFood	FastFood
	EuroBank Central	EuroBank Paris site	EuroBank, VoIP	EuroBank, VoIP
	EuroBank	EuroBank Chartres site	EuroBank	EuroBank
		EuroBank Nantes site		
	VoIP	Paris VoIP Gateway	VoIP	VoIP

You might assume that the RTs attached to routes exported from a VRF always match the set of import RTs of a VRF. Although that is certainly true in simpler VPN topologies, there are widespread VPN topologies (for example, central services VPN) in which this assumption is not true.

Example 1:

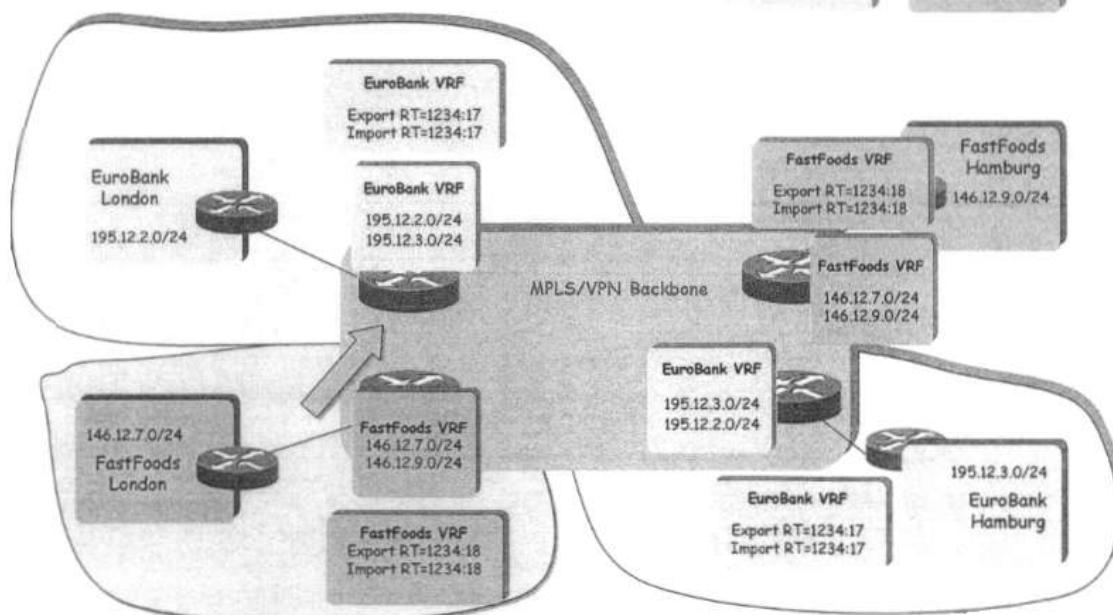


## intranet VPN Connectivity Between Sites

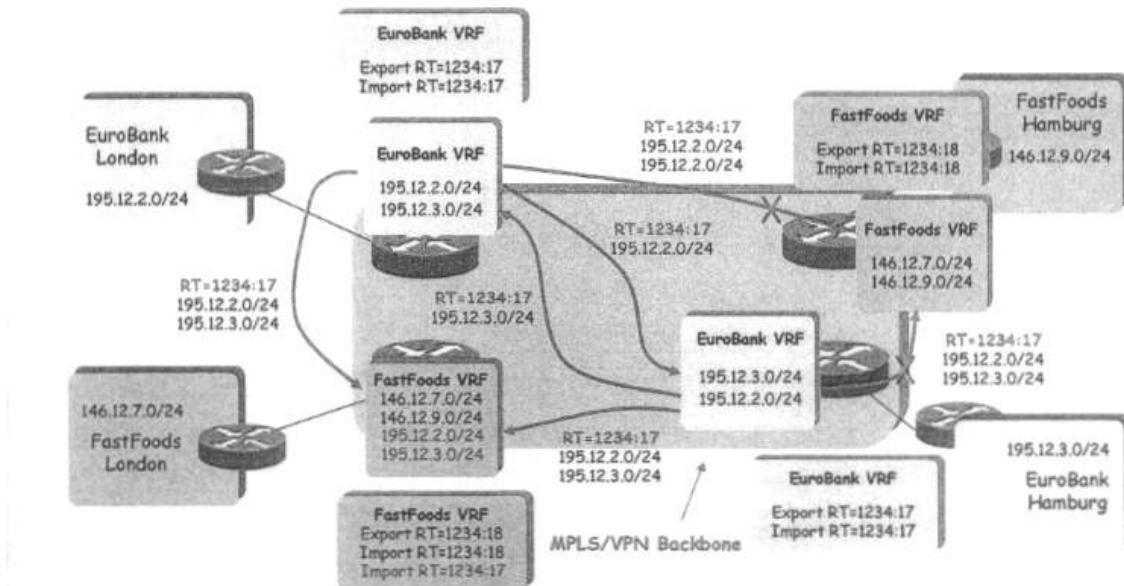
EuroBank  
RT = 1234:17FastFoods  
RT = 1234:18

## Example 2:

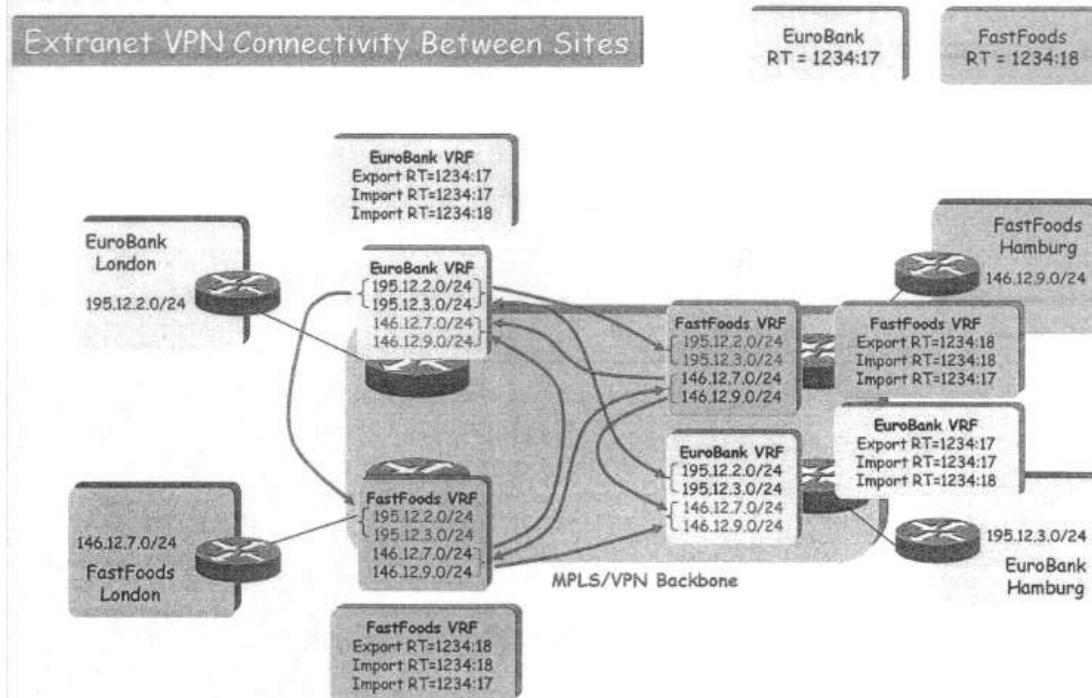
## Extranet VPN Connectivity Between Sites

EuroBank  
RT = 1234:17FastFoods  
RT = 1234:18

## Extranet VPN Connectivity Between Sites

EuroBank  
RT = 1234:17FastFoods  
RT = 1234:18

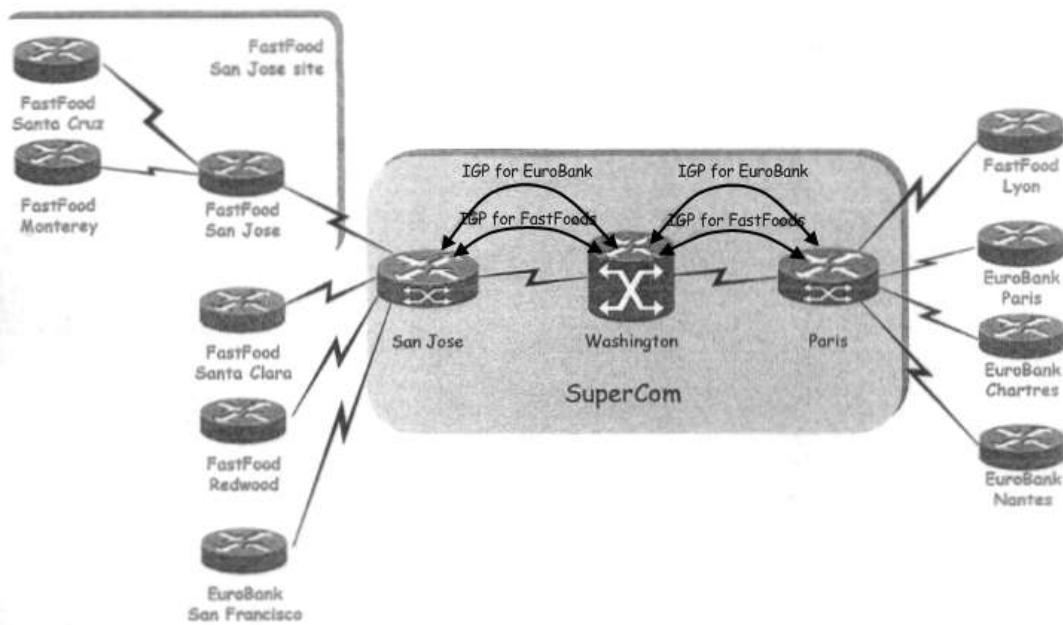
Example 3:

**7.16 Propagation of VPN Routing Informations**

We have previously explained MPLS/VPN architecture from a single PE router standpoint. Two issues have yet to be addressed:

- How will the PE routers exchange information about VPN customers and VPN routes between themselves?
- How will the PE routers forward packets originated in customer VPNs?

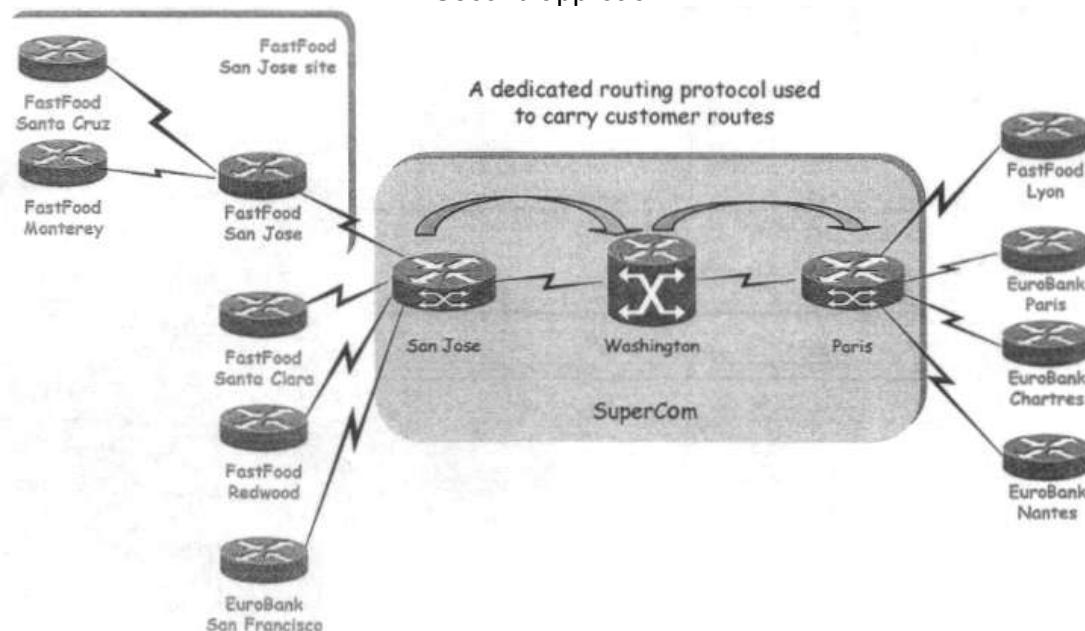
First approach:



**Question:** How will PE routers exchange customer routing information?

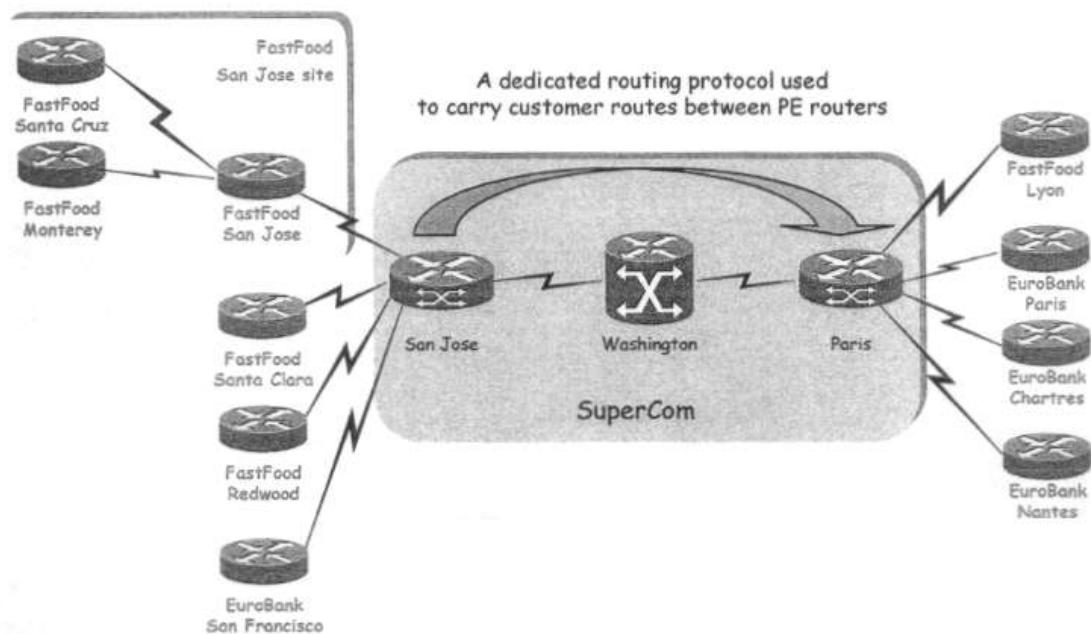
**Answer 1:** Run a dedicated Interior Gateway Protocol (IGP) for each customer (i.e. VPN) across the Provider-Network. Wrong answer: the solution does not scale, P routers carry all customer routes. Because you cannot run more than one copy of OSPF over the same link, you would have to configure per-VPN subinterfaces on the link between San Jose (or Paris) and Washington, resulting in an extremely complex network. Suppose that SuperCom network would have to support more than 100 VPN customers connected to the San Jose and Paris PE routers with OSPF as the routing protocol. The PE routers in the SuperCom network would run more than 100 independent copies of OSPF routing process (if that were technically possible), with each copy sending hello packets and periodic refreshments over the network.

**Second approach:**



**Answer 2:** Run a single routing protocol that will carry all customer routes inside the provider backbone. Better answer, but still not good enough: P routers carry all customer routes.

**Third approach:**



**Answer 3:** Run a single routing protocol that will carry all customer routes *between PE routers*. Use MPLS labels to exchange packets between PE routers. **The best answer:** P routers do not carry customer routes. The solution is scalable.

**Question:** Which protocol can be used to carry customer routes between PE routers?

**Answer:** The number of customer routes can be very large. BGP is the only routing protocol that can scale to a very large number of routes. **Conclusion:** BGP is used to exchange customer routes directly between PE routers.

## 7.17 BGP Considerations

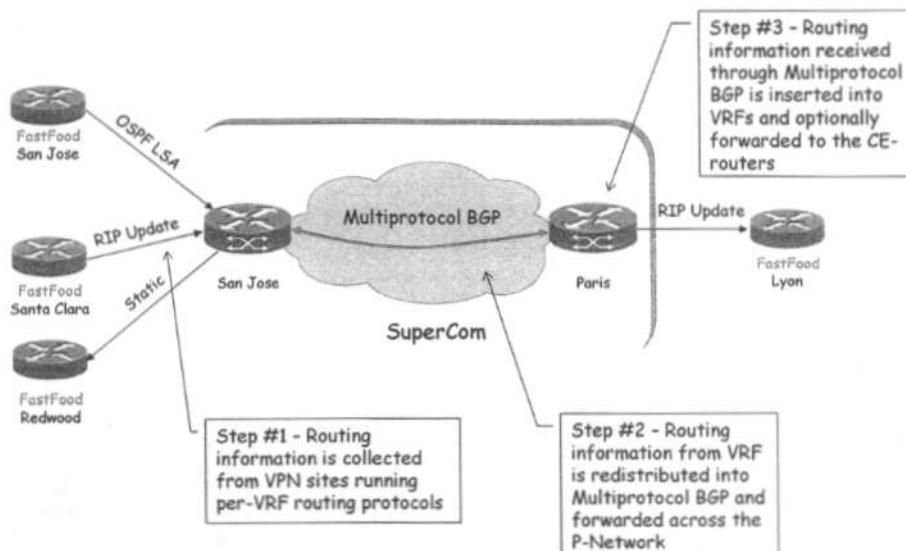
Reasons for choosing BGP as the routing protocol used to transport VPN routes are:

- The number of VPN routes in a network can become very large and BGP is the only routing protocol that can support a very large number of routes.
- BGP, EIGRP, and IS-IS are the only routing protocols that are multiprotocol by design (all of them can carry routing information for a number of different address families).

However:

- IS-IS and EIGRP do not scale to the same number of routes as BGP.
- BGP is also designed to exchange information between routers that are not directly connected. This BGP feature supports keeping VPN routing information out of the provider core routers (P-routers).
- BGP can carry any information attached to a route as an optional BGP attribute and more than that it is possible to define additional attributes that will be transparently forwarded by any BGP router that does not understand them.

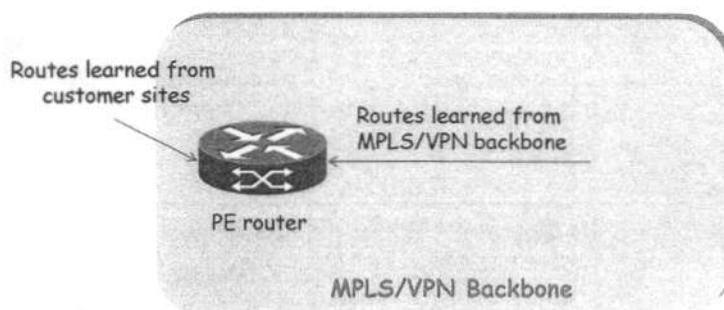
Recall that if an optional attribute is not recognized by the BGP implementation, that implementation looks for a *transitive flag* to see whether it is set for that particular attribute. If the flag is set, which indicates that the attribute is *transitive*, the BGP implementation should accept the attribute and pass it along to other BGP speakers. In the next figure there is a static route configured on the San Jose PE-router and the default route configured on the Redwood router. The routing protocols used in FastFood VPN are OSPF (San Jose) and RIP (Santa Clara).



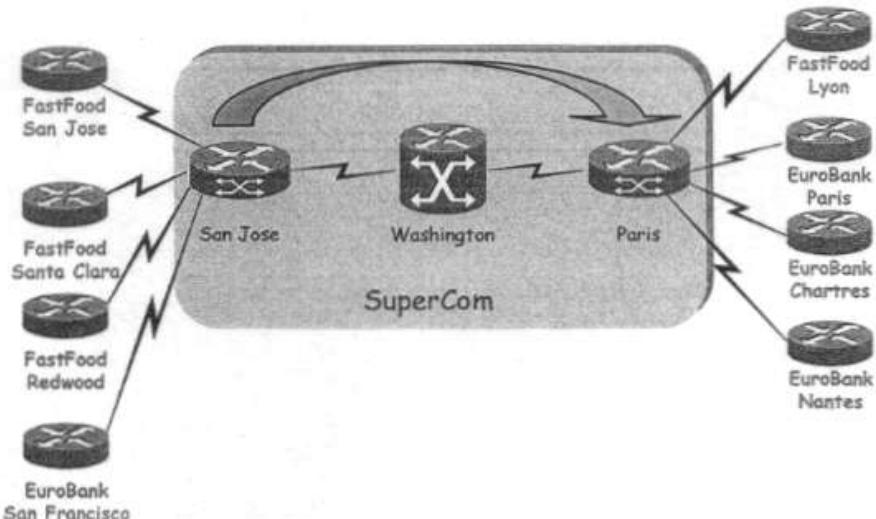
A BGP session between PE routers belonging to the same MPLS/VPN domain is consequently called a *MultiProtocol BGP (MP-iBGP)* session, where “*i*” stands for “*interior*”.

### 7.18 BGP Extended Community Attribute

We have seen already that each PE router learns routes from across the MPLS/VPN backbone, and from attached customer sites.

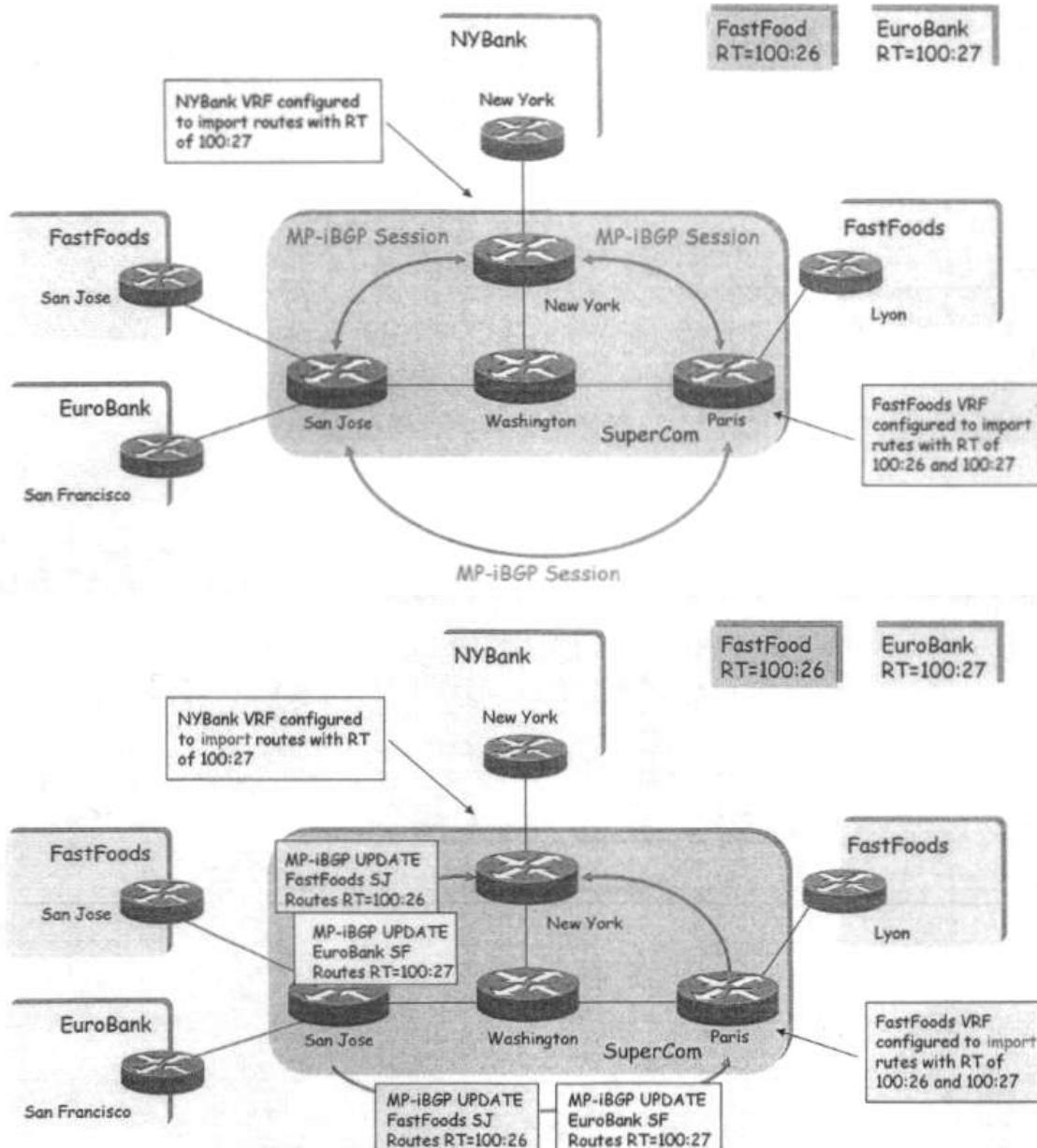


Any routes learned from customers are advertised across the MPLS/VPN backbone through the use of MP-iBGP, and any routes learned via MP-iBGP are placed into the VRFs of interested parties.



To achieve this, each PE router needs information that tells it how to process any routes it receives. This information not only tells the PE router into which VRF the routes should be imported but also what information it should append to the route when advertising to other PE routers. We have already introduced the concept of a route target and briefly explained how this entity determines which VRF, and thus which VPN sites, should receive the route. Although the RT provides the mechanisms to identify which VRFs should receive the routes, it does not provide a facility that can prevent routing loops. These loops could occur if routes learned from a site are advertised back to that site. To prevent this, the *Site of Origin (SOO)*, another concept in the

MPLS/VPN architecture, identifies *which site originated the route*. The *BGP Extended Community attribute* applies both of these concepts (SOO and RT) to the MPLS/VPN architecture. This attribute is described in *draft-ramachandra-bgp-ext-communities*, which you can find on the IETF Web site at “[www.ietf.org](http://www.ietf.org)”. *Draft-ramachandra-bgp-ext-communities* defines a new transitive optional BGP attribute, which contains a set of *Extended Communities* that define the site from where the VPN-IPv4 address was learned (the route origin) and the set of routers to which the route should be exported (the route target). The extended community is attached to a BGP route the same way standard BGP communities or any other BGP attributes are. The MP-iBGP UPDATE propagates the extended community along with other BGP attributes between PE routers. Careful definition of the route target extended community values provides the flexibility to provision many different VPN topologies. One such topology, which uses the MPLS/VPN backbone from the case study, helps to show the basic operation of the route target.



The previous figure shows that the SuperCom San Jose PE router exports routes for the FastFoods VPN with a route target of 100:26 and for the EuroBank VPN with a route target of 100:27. The NYBank VRF on the SuperCom New York router imports routes with a route target of 100:27. This means that it contains only routes for the EuroBank San Francisco site. The FastFoods VRF on the SuperCom Paris router imports routes with a route target of 100:26 and 100:27. This means that it contains routes for the FastFoods San Jose site and for the EuroBank San Francisco site. The example also shows how one site within a VPN can hold routes that are not contained in another site of the same VPN. The Paris PE router imports routes with a route target of 100:26 and 100:27 and therefore the FastFoods Lyon site has routes that belong to the EuroBank San Francisco site and the FastFoods San Jose site. However, neither the EuroBank or FastFoods VRFs on the San Jose PE router are configured to import both route target values.

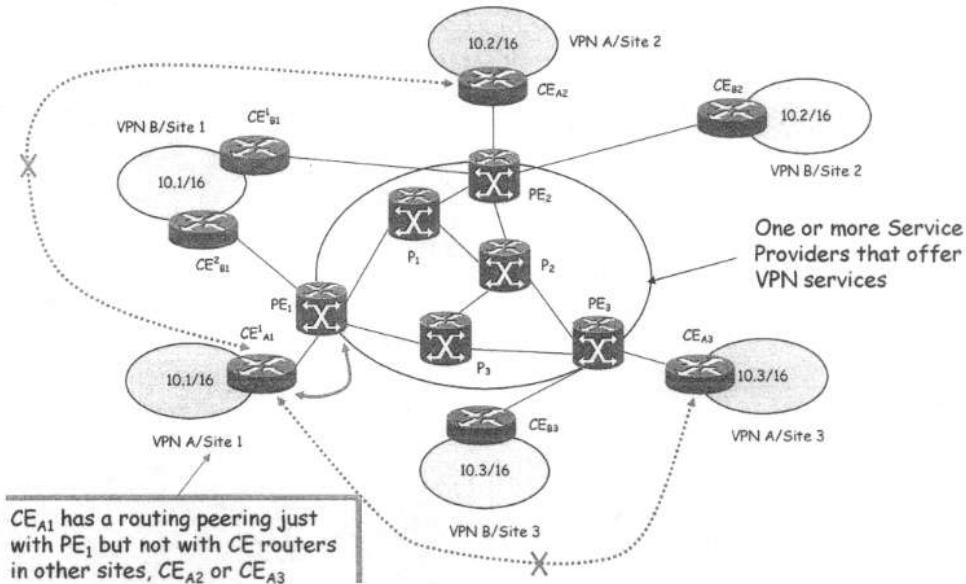
They contain only routes that belong to their particular VPN because the Paris PE router does not re-advertise routes from within its VRF that it learned via iBGP (standard iBGP rules).

*Consideration:* a VPN customer doesn't have to be involved in understanding the above concepts (Multiple VRFs, Route Target, BGP Extended Community, etc.) which, in turn, helps to accomplish the goal of being able to offer VPN services to customers who don't have significant expertise in IP routing.

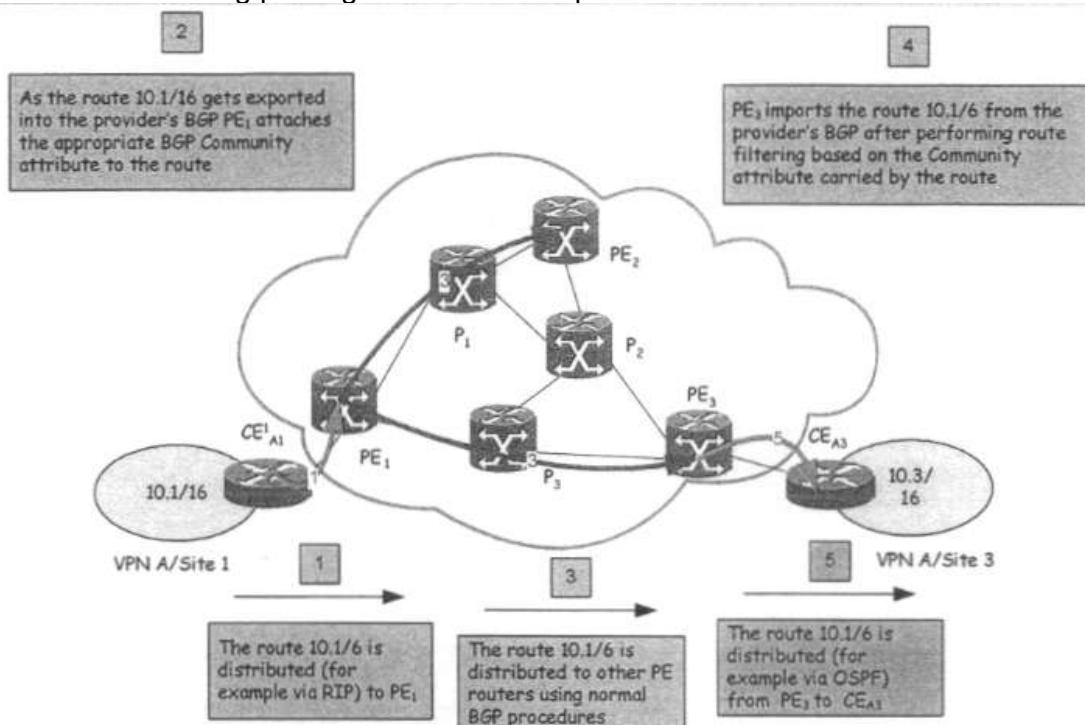
## 7.19 MPLS/VPN Characteristics

Characteristics of the mechanism used by BGP/MPLS VPN to control intersite connectivity:

*First* - within a given VPN, a CE router maintains routing peering just with its directly connected PE router, but not with CE routers in other sites of that VPN (this is a basic attribute of the peer model).

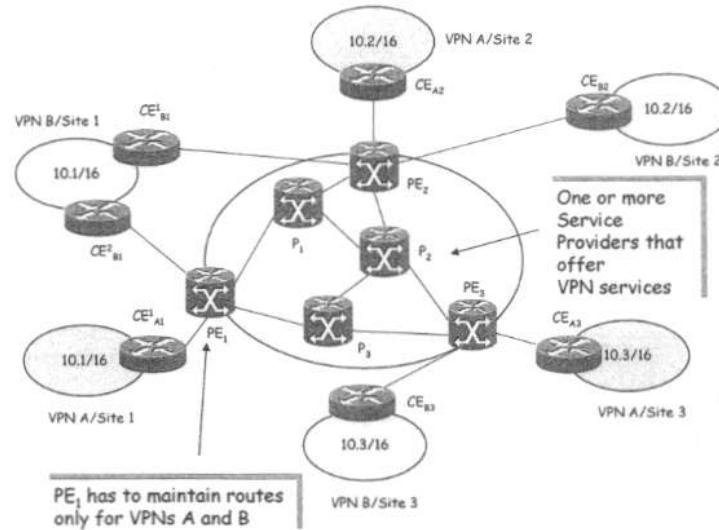


With the peer model, the number of routing peers that a CE router has to maintain is constant and independent of the total number of sites within a VPN (that facilitates support of large VPNs with hundreds and thousands of sites per VPN, because the amount of routing peering that a CE route has to perform is independent of the total number of sites within a VPN). With the overlay model, the need to have a complete mesh routing peering among all the sites means that the amount of routing peering grows with the number of sites. *Implication:* the scalability of the overlay model in the area of routing peering is inferior to the peer model.



*Second* - to add a new site to an existing VPN, all a service provider has to do is appropriately configure the PE router that will be connected to the new site. With the peer model the amount of configuration changes needed to handle addition/deletion of sites within a given VPN is constant and independent of the number of sites of that VPN. With the overlay model, the number of such configuration changes is proportional to the number of sites (as adding a new site requires addition of a new Frame Relay or ATM virtual circuit or a new IP tunnel to all other sites). *Implication*: the scalability of the overlay model in the area of configuration management is inferior to the BGP/MPLS VPN approach.

*Third* - a PE router has to maintain only the routes for the VPNs whose sites are directly connected to that PE router (i.e., whose sites have CE routers connected to that PE router).



## 7.20 BGP Extended Community Attribute Format

Now that we know how the *BGP extended community attribute* is used in the MPLS/VPN architecture, we can learn how this attribute is structured. Each of the extended community attributes has a defined community type code of 16 and is encoded as an 8-octet value. The first two octets define the *attribute type*, and the next six octets hold the *value of the attribute*.

<2 octets>	<6 octets>
------------	------------

Types 0 through 0xFFFF inclusive are assignable by the *Internet Assigned Numbers Authority (IANA)*, and types 0x8000 through 0xFFFF inclusive are vendor-specific. The *Route Target* extended community has a type code of 0x0002 or 0x0102, and the *SOO* extended community has a type code of 0x0003 or 0x0103. The structure of the value field (and the way the value field is displayed) depends on the high order byte of the type field.

General formats:

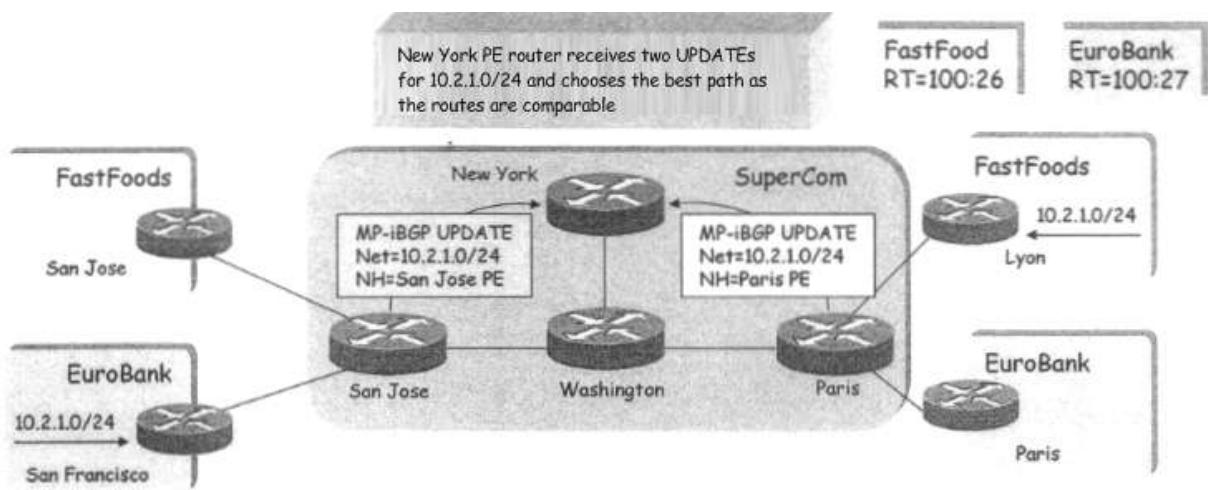
<16 bit type>	<16 bit ASN> <32 bit number>
---------------	------------------------------

<16 bit type>	<A.B.C.D>:<16 bit number>
---------------	---------------------------

Since each Extended Community embeds the AS number (just like a plain Community) which are globally unique, it follows that each service provider can control local assignment on its own, and that the global uniqueness of such assignments is maintained. A route can carry both the standard BGP Communities attribute as defined in RFC1997 and the extended BGP Communities attribute as defined in *draft-ramachandra-bgp-ext-communities*.

## 7.21 VPN-IP Addresses

One issue has yet to be addressed: *How will the PE routers forward packets originated in customer VPNs?* BGP assumes that IP addresses are unique. That assumption is clearly incorrect in the VPN service provider environment, where the same block of IP addresses (e.g. private addresses as defined in RFC 1918) could be simultaneously used by multiple VPN customers.



In this scenario, the PE router chooses the best route between the two routes received based on the standard BGP decision process. This means that a mechanism is needed so that MP-iBGP does not consider identical (thus comparable) routes as belonging to different VPNs, even if these routes carry the same IPv4 Network Layer Reachability Information (NLRI). To use BGP-based mechanisms, we need to figure out how to use BGP in an environment where IP addresses are no longer unique. *Solution:* map non-unique addresses into unique addresses. *How?* By creating a new type of address, called **VPN-IP (VPNv4) address**, and making sure that these addresses are unique. A VPN-IP address is constructed by concatenating a fixed-length field, called a *Route Distinguisher (RD)*, with a plain IP address.

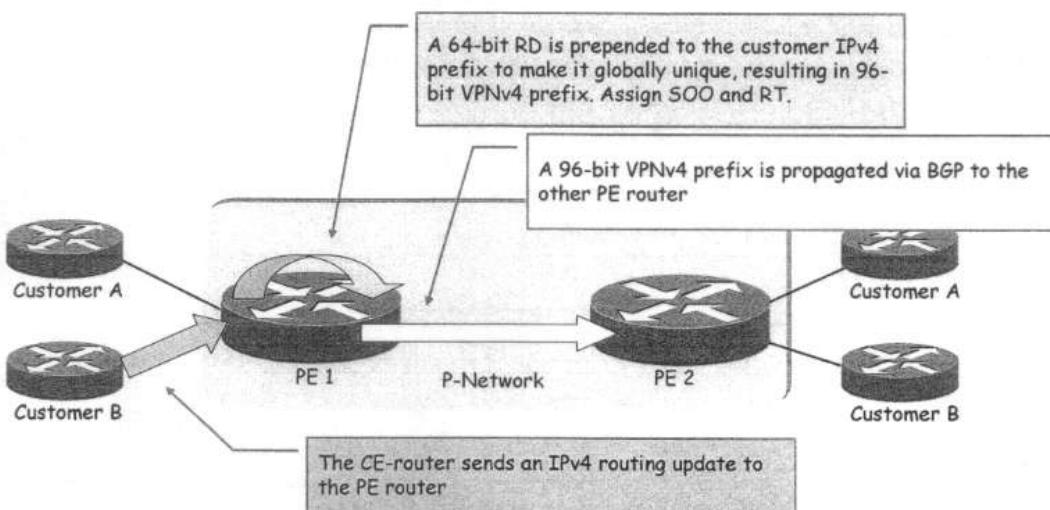
A *Route Distinguisher (RD)* consists of three fields (64 bits):

- Type (2 octets),
- Autonomous System (AS) Number (2 octets), and
- Assigned Number (4 octets).

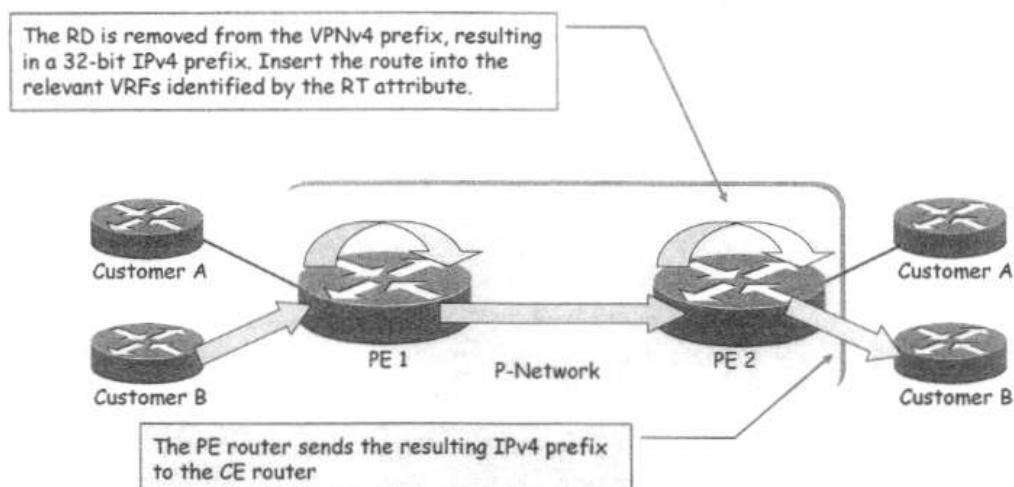
Route Distinguisher (64 bits)	IPv4 address (32 bits)
e.g., 100:34:192.10.3.1	

The *AS Number* field contains the Autonomous System Number of the VPN service provider. *AS Number* format is recommended. The assignment of the *Assigned Number* field is controlled by each VPN service provider on its own. A service provider assigns just one value to a given VPN served by the provider but in some cases it may assign more than one value. Since no two VPNs within a given service provider share the same *Assigned Number*, it follows that no two VPNs share the same *Route Distinguisher*. Since IP addresses are assumed to be unique within a VPN, and since no two VPNs share the same RD, it follows that VPN-IP addresses are globally unique. From the BGP point of view, handling routes for VPN-IP addresses is no different than handling routes for plain IP addresses. The structure of VPN-IP addresses, as well as the structure of the *Route Distinguisher* component of VPN-IP addresses, is totally opaque to BGP - when BGP compares two VPN-IP address prefixes, it ignores the structure. BGP does not need new mechanisms. For example, such mechanisms as *BGP Communities*, *route filtering based on the communities*, *use of BGP Route Reflectors* or *BGP Refresh*, and so on, are as applicable to routes for VPN-IP addresses as to routes for plain IP addresses. VPNv4 addresses are exchanged only between PE routers; they are never used between CE routers (i.e., VPNv4 addresses are not visible to the customer). BGP between PE routers must therefore support exchange of traditional IPv4 prefixes as well as exchange of VPNv4 prefixes. Use of VPN-IP addresses is completely confined to a VPN service provider - a VPN customer (and, specifically the customer's equipment) is unaware of VPN-IP addresses. *Conversion between VPN-IP addresses and IP addresses happens at the PE routers*. For each directly connected VPN, a PE router (as part of the setup of a VPN site) is configured with a *Route Distinguisher*. The RD has no special meaning or role in MPLS/VPN architecture - its only function is to make overlapping IPv4 addresses globally unique. Simple VPN topologies require only one RD per customer, raising the possibility that the RD could serve as RT. This design, however, would not allow implementation of more complex VPN topologies, as when a customer site belongs to multiple VPNs.

## 7.22 RD Usage in an MPLS/VPN

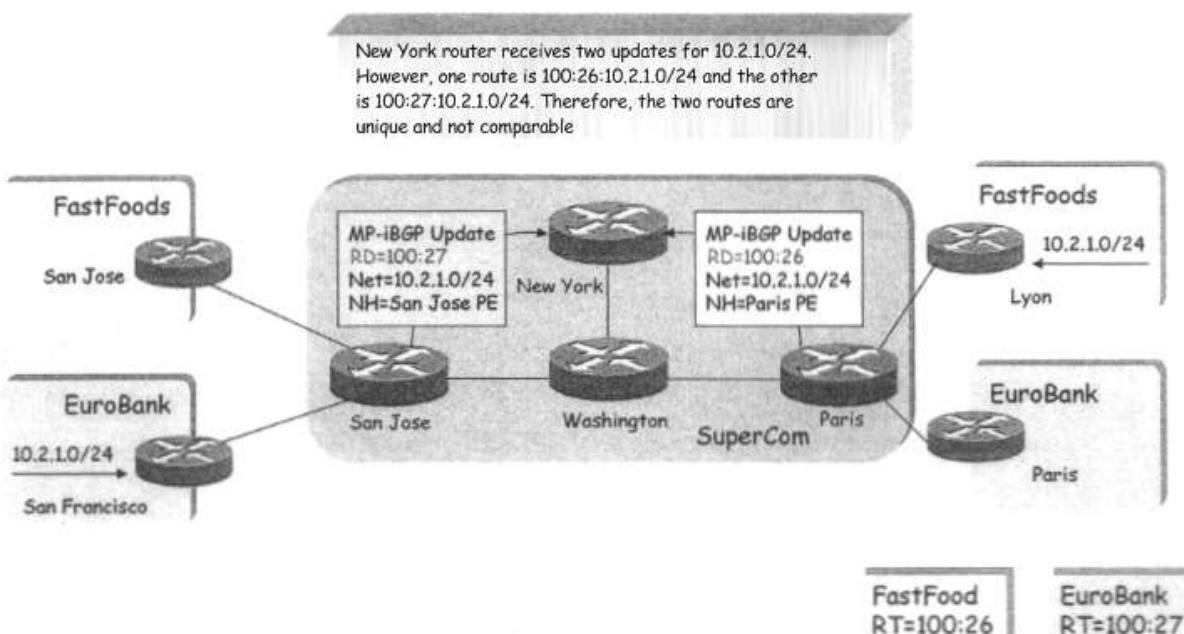


When the PE router receives a route from a directly attached CE router, the PE router identifies the VPN that the CE router belongs to and, before exporting this route into the provider's BGP, converts the reachability information of this route from plain IP to VPN-IP by using the Route Distinguisher that is configured for that VPN.



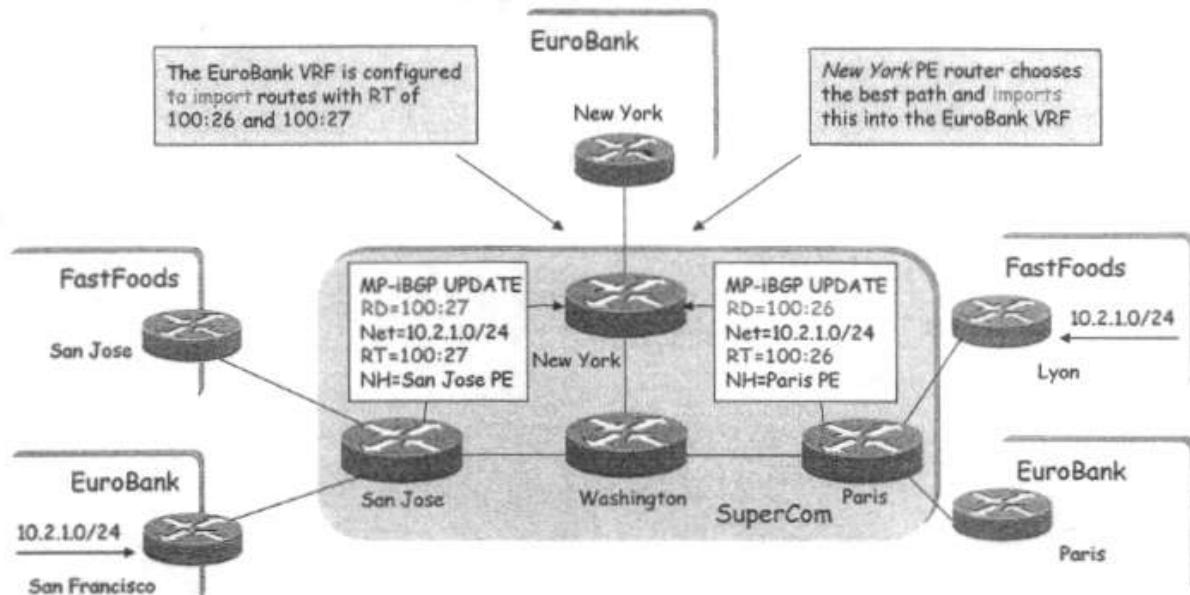
Likewise, when a PE router has to import routes from the provider's BGP, the PE router converts the reachability information of these routes from VPN-IP addresses to plain IP addresses.

#### PE Router Comparison of BGP Routes:



The RD mechanism allows VPN customers to use the same private addressing scheme. However, the RD mechanism does not solve the problem of multiple customers within the same VPN using the same addressing scheme within their sites.

## 7.23 An Example



The New York PE router receives an MP-iBGP update for subnet 10.2.1.0/24 from two separate VPNs: the EuroBank and FastFoods VPNs. The EuroBank VPN is configured to import any routes that contain the RTs of 100:26 or 100:27; i.e. EuroBank imports any routes from members of the EuroBank or FastFoods VPNs as they export their routes using these route targets. The New York PE router compares the two routes to determine which one to import into the EuroBank VRF; depending on which one is chosen, connectivity to the other VPN site is lost. For example, if the New York PE router determines that the MP-iBGP update for 10.2.1.0/24 received from the Paris PE router is the best path, then connectivity from the EuroBank New York site to destinations within subnet 10.2.1.0/24 in the EuroBank San Francisco site are lost. For this reason, the design of the MPLS/VPN architecture was restricted to limit the use of overlapping address ranges to VPNs that do not communicate with each other across the MPLS backbone if they share the same set of addresses within their sites. If full connectivity between VPNs is required, the address ranges should be unique, or *Network Address Translation (NAT)* could be deployed.

Follows a comparison between the roles played by Route Distinguishers and Route Targets. There are two separate problems to be solved:

- the first problem is how to deal with globally non-unique addresses;
- the second problem we need to solve is how to constrain connectivity.

To solve the first problem, we introduce a new type of address, the VPN-IP address, and use Route Distinguishers to make such addresses globally unique. The Route Distinguisher is used to disambiguate IP addresses, it's not used to constrain connectivity, as it is not used for route filtering. To solve the second problem we use filtering based on Route Targets. The Route Target is not used to disambiguate IP addresses. While the same RD can't be used by more than one VPN, a given VPN may use multiple Route Distinguishers (e.g., multiprovider operations). While the same RT can't be used by more than one VPN, a given VPN may use multiple RTs (e.g., hub-and-spoke VPN). *RT numbering need not follow RD numbering.*

### RD Summary:

1. RD has VPN-local significance.
2. All routes that are part of the same VPN can use the same RD.
3. No duplicate IP addresses allowed within the same VPN.
4. Sites belonging to the same VPN may have to use different RDs when these sites also belong to other different VPNs.
5. With central services or hub and spoke topology, all client or spoke sites must use different RDs.

## 7.24 Enhanced BGP Decision Process for VPN-IPv4 Prefixes

The RT BGP extended community and the RD control the VPN-IPv4 route selection process. This process occurs after routes are learned from other PE routers across MP-iBGP sessions but before these routes are imported into any VRFs. The first step of the BGP decision process is to group all relevant routes so they can be compared. Before the PE router can select routes, it has to know which VPN routes exist and which of these routes should be comparable with each other by the BGP selection process. When the PE router is provisioned for VPN service, each VRF is

configured with statements that tell the PE router which routes should be imported into the VRF. You already know that the route target BGP extended community controls this import process. Armed with this information, the PE router does the following:

- takes all routes with the same RT as any of the import statements within the VRF;
- considers all routes that have the same RD as the one assigned to the VRF being processed;
- creates new BGP paths with a RD that is equal to the RD configured for the VRF that is being processed.

All the routes are now comparable and, at this point, the BGP selection process is executed.

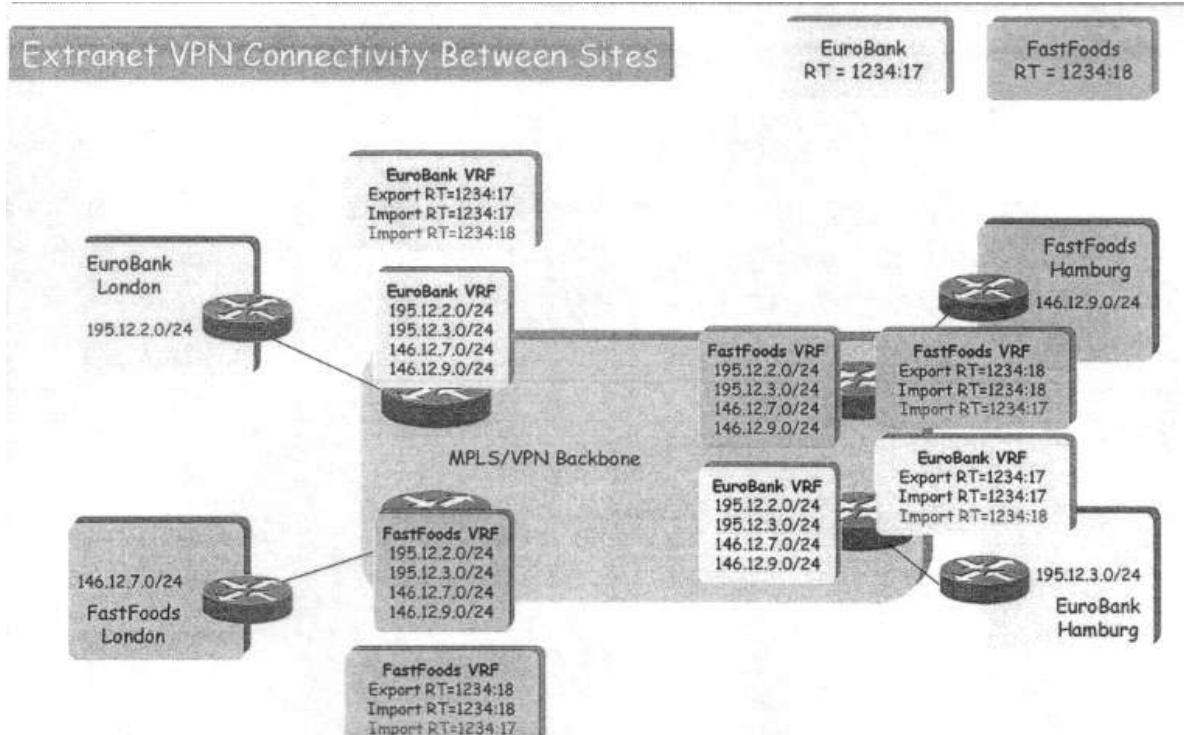
## 7.25 Impact of Complex VPN Topologies on Virtual Routing Tables

A single virtual routing table (VRFs) can be used only for sites with identical connectivity requirements. Complex VPN topologies therefore require more than one virtual routing table per VPN. Because each virtual routing table requires a distinctive RD, the number of RDs in an MPLS VPN network increases with the introduction of overlapping VPNs. Moreover, the simple association between RD and VPN that was true for simple VPNs is also gone.

## 7.26 Intranet and Extranet Integration

Numerous topologies can be deployed using the MPLS/VPN architecture. There are too many scenarios to be able to provide examples for each and every one; instead, we present some of the more widely used topologies. Extranet support within the context of the MPLS/VPN architecture simply involves the import of routes from one VRF into a different VRF that services another VPN site. This is controlled through the use of the RT and import statements within the VRF that is associated with the particular VPN site. In the following example (Example 1), we can see that two separate organizations are capable of communicating directly across the MPLS/VPN backbone as they import each other's routes into their relevant VRFs.

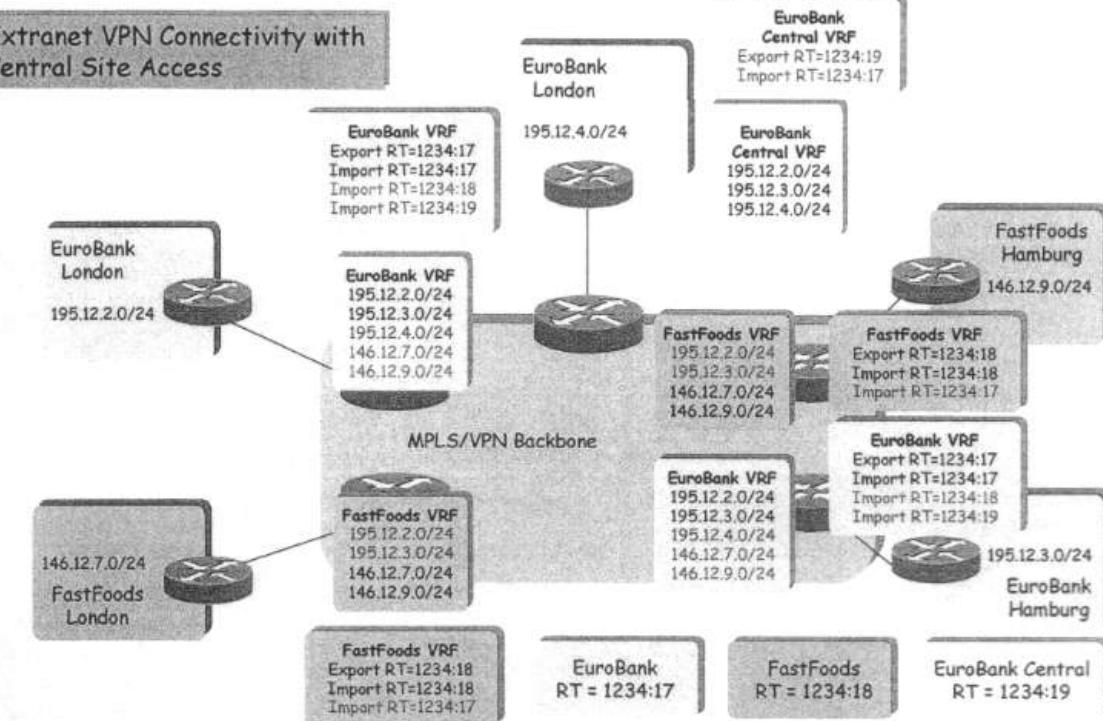
Example 1 – Extranet VPN connectivity between sites:



If we examine the EuroBank London site, for example, we can see that it has its routes advertised with a route target value of 1234:17. All other PE routers that have EuroBank sites (or FastFoods sites) attached are configured to import any routes that contain a route target with a value of 1234:17 into the EuroBank and FastFoods VRFs. These PE routers are also configured to import any routes that contain a route target value of 1234:18, which corresponds to the FastFoods VPN, into both of these VRFs. Because all the FastFoods site routes are advertised across the MPLS/VPN backbone with a route target value of 1234:18, these routes are also imported into the EuroBank and FastFoods VRFs. This means that all EuroBank and FastFoods sites have both

EuroBank and FastFoods routes within their local routing table and, therefore, have connectivity across organizations. This type of topology can be enhanced further through manipulation of the route target as shown in Example 2.

Example 2 – Extranet VPN connectivity with Central Site Access:



In Example 2, the EuroBank VPN customer has added a central site to the topology with which only EuroBank sites should be capable of communicating. The Example 2 shows that the EuroBank central site exports its routes with a route target value of 1234:19. Routes that contain this RT value are imported into the EuroBank VRF on all PE routers, but not into the FastFoods VRF because this has not been configured to import routes with this particular route target value. This means that only the EuroBank sites are capable of accessing the central site, but they are also still capable of accessing the FastFoods sites. However, the FastFoods sites are not capable of accessing the EuroBank central site, even though they import routes from the EuroBank VPN.

It should be noted that with this type of connectivity, IP address space between the two different VPN customers should be unique. In our example, if the FastFoods sites were to use the same address space as the EuroBank central site (or any other EuroBank site), and vice versa, connectivity would be broken because sites would receive the same set of routes from two different locations.

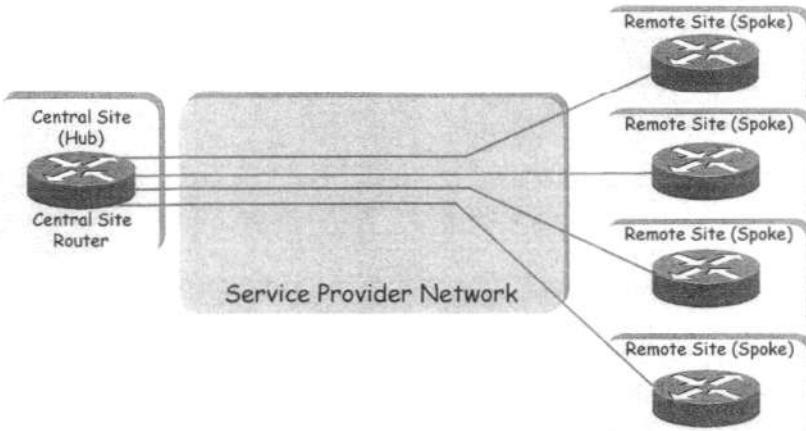
## 7.27 VPN Hub-and-Spoke Topology<sup>3</sup>

In certain circumstances, it may be desirable to use a *hub-and-spoke topology* so that all spoke sites send all their traffic toward a central site location. This may be because:

- certain central site services for a particular VPN, such as Internet access, firewalls, server farms, and so on, are housed within the hub site;
- this particular VPN customer requires that all connectivity between its sites be via the central site.

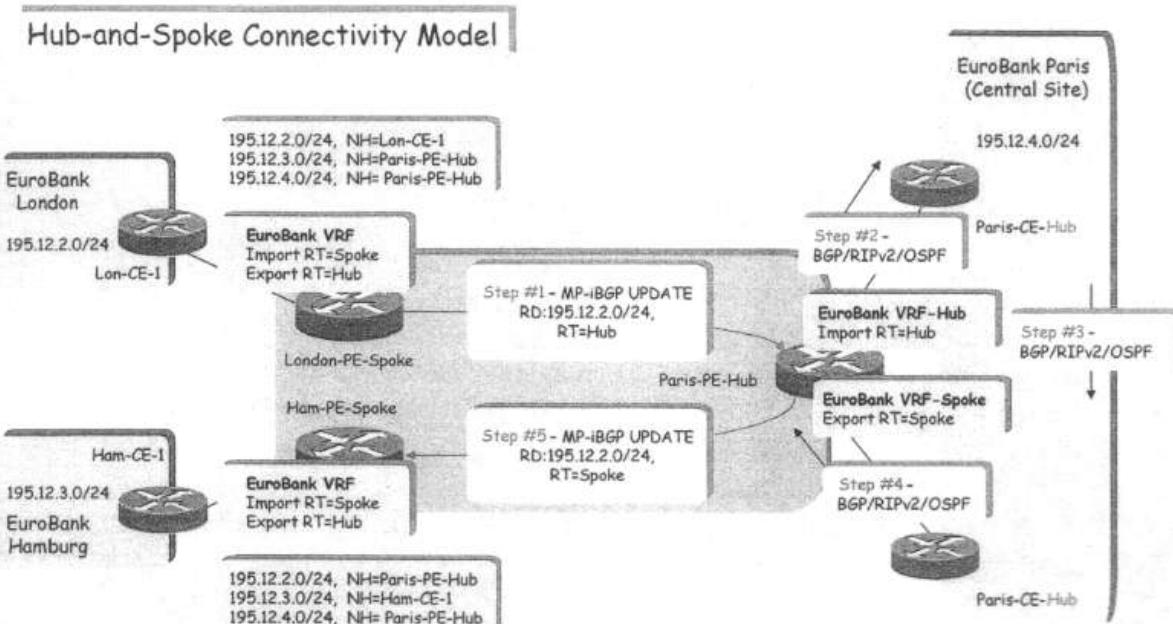
In either case, it is necessary to define a *hub site* that contains *full routing knowledge* of all *spoke sites* that belong to the same VPN.

<sup>3</sup> La parte finale del paragrafo è confusa e poco comprensibile. Viene qui riportato il testo originale senza modifiche.



All traffic from the spoke sites, destined either for the central site services or for intersite connectivity, will flow via the central hub site. With this type of topology, the spoke sites export their routes to the hub site, and then the hub site re-exports the spoke site routes through a second interface (either physical or logical) using a different route target so that other spoke sites can import the routes. This causes the hub site to become a transit point for interspoke connectivity. An example (Example 3) of the Hub-and-Spoke topology is shown in the following figure.

Example 3 – Hub-and-Spoke connectivity model



Example 3 shows two spoke sites, EuroBank London and EuroBank Hamburg, that do not have direct connectivity but that can reach each other by sending traffic via the EuroBank Paris central site. For simplicity, the relevant RTs have been given names rather than numerical values. We can see in Example 3 that the EuroBank London site advertises its local prefixes, an example of which is 195.12.2.0/24, and these are placed into the EuroBank VRF on the Lon-PE-Spoke PE router. To achieve the desired connectivity, the following steps can be taken:

1. The Lon-PE-Spoke PE router exports all the routes from the EuroBank VRF using a route target value of Hub.
2. The hub PE router (Paris-PE-Hub) is configured to import the Hub route target into one of its VRFs (defined as VRF-Hub). This VRF interface attaches to the EuroBank central site and forwards all routes that are learned from the spoke sites to the CE-router (Paris-CE-Hub), which is located within the central site.
3. The Paris-CE-Hub CE-router advertises these routes across the central site.
4. The routes are eventually advertised back to the Paris-PE-Hub PE router across a separate interface.
5. The Paris-PE-Hub PE router then advertises the routes back into the MPLS/VPN backbone with a route target value of Spoke.
6. Each of the spoke PE routers (Lon-PE-Spoke and Ham-PE-Spoke) is configured to import any routes with a route target value of Spoke into the EuroBank VRF. This means that the NEXT\_HOP address for all the spoke sites, as seen by other spoke sites, is via the central

site Paris-PE-Hub PE router

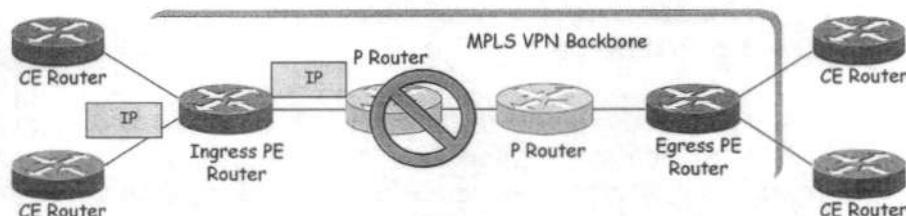
The simplest way to configure this type of connectivity would be to assign the same RD to each spoke site VRF because the spoke sites do not exchange routing information directly with each other and do not import each other's RTs. This is possible when the spoke sites are not connected to the same PE router as the hub site because of the automatic VPN-IPv4 route filtering feature which filters any routes that contain a route target that is not configured to be imported into any VRFs on the PE router.

In the case shown in Example 3 of a distributed hub-and-spoke model everything works fine because the spoke PE routers are not configured to import the route target with a value of Spoke<sup>4</sup>. However, if the hub-and-spoke sites are connected to the same PE router, then it is essential to assign a different RD per spoke site VRF; otherwise, the automatic route filtering feature will not filter the routes with a route target value of Spoke, and essential routing information could potentially be lost. The loss of routing information can be explained through the MPLS/VPN architecture and also through how the BGP selection process is executed within an MPLS/VPN environment. Architecturally, two or more VRFs may use the same RD if these VRFs are used by sites belonging to the same VPN and share exactly the same routing information. This is not the case with the hub-and-spoke topology because the hub site imports different routes from the spoke sites. The BGP selection process applies to all routes that must be imported into the same VRF, plus all routes that have the same RD as this VRF. When the selection process is finished, only the best routes are effectively imported. In the case of the hub-and-spoke topology, if the same RD is used for each spoke site, this could result in a "best" route that is not imported into the VRF. This is because, if all spoke VRFs have the same RD, then each PE router will include in the selection process not only the routes coming from the hub site, but also the routes coming from the other spoke sites (which are not configured to be imported). Automatic route filtering will prevent this if the topology is distributed, but it will not help if the hub-and-spoke sites are connected through the same PE router. It is also worth noting that if the same RD is used on every PE router for a particular VPN, then the topology is somewhat restricted because it requires that all spokes connected to a particular PE router for a particular VPN use the same VRF. This is because each VRF on a PE router must have a unique route distinguisher. This means that all spokes for a particular VPN would need to share the exact same routing information.

## 7.28 VPN Packet Forwarding Across an MPLS/VPN Backbone

**Question:** How will the PE routers forward the VPN packets across the MPLS VPN backbone?

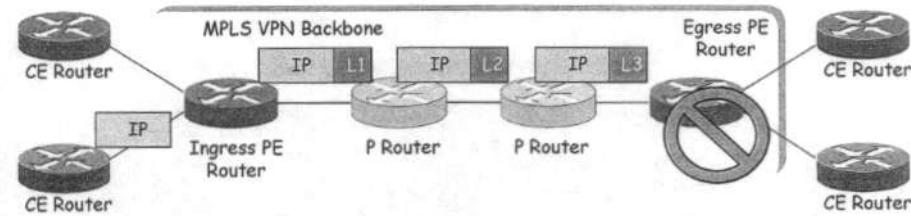
**Answer 1:** They will forward pure IP packets. Wrong answer: P routers do not have VPN routes; the packet is dropped on IP lookup.



**Idea:** How about using MPLS for packet propagation across the backbone? To provide forwarding of IP packets along the routes expressed in terms of VPN-IP addresses, the MPLS technology is used. **Why?** Because MPLS decouples the information used for packet forwarding (*the label*) from the information carried in the IP header. By using MPLS we can bind LSPs to VPN-IP routes and then forward IP packets along these routes. From the MPLS point of view the PE router is an edge LSR.

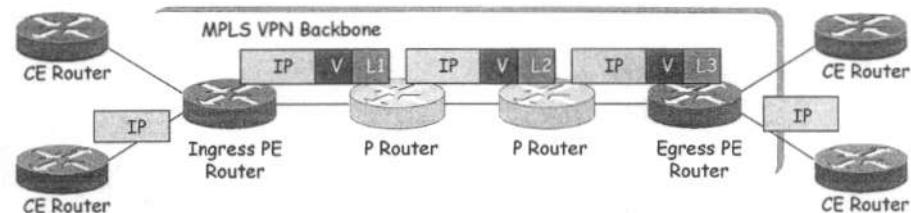
**Answer 2:** They will label the VPN packets with an LDP-label for the egress PE router and forward the labeled packets across the MPLS backbone. Better answer: the P routers perform the label switching and the packet reaches the egress PE router. However, the egress PE router does not know which VRF to use for packet switching, so the packet is dropped.

4 In realtà i PE-Spoke Router sono configurati ad importare le rotte con RT "Spoke". Viene qui riportato il testo originale senza modifiche.

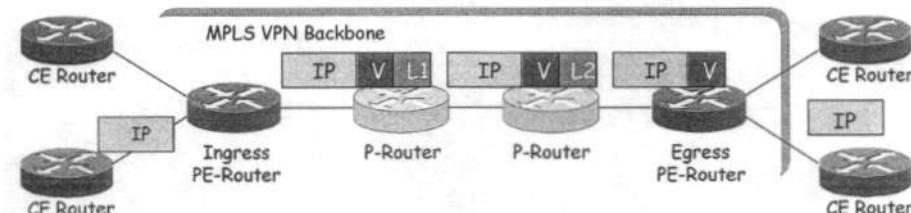


Idea: How about using a label stack?

Answer 3: They will label the VPN packets with a label stack, using the LDP label for the egress PE router as the top label and the VPN label assigned by the egress PE router as the second label in the stack. Correct answer: the P routers perform label switching and the packet reaches the egress PE router. The egress PE router performs a lookup on the VPN label and forwards the packet toward the CE router.



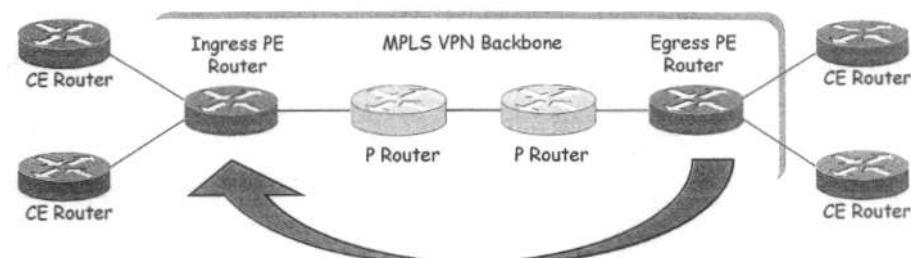
Penultimate hop popping on the LDP label can be performed on the last P router. The egress PE router performs label lookup only on the VPN label, resulting in faster and simpler label lookup. IP lookup is performed only once - in the ingress PE router.



## 7.29 VPN Label Propagation

Question: How will the ingress PE router get the second label in the label stack from the egress PE router?

Answer: Labels are propagated in MP-iBGP VPNv4 routing updates.



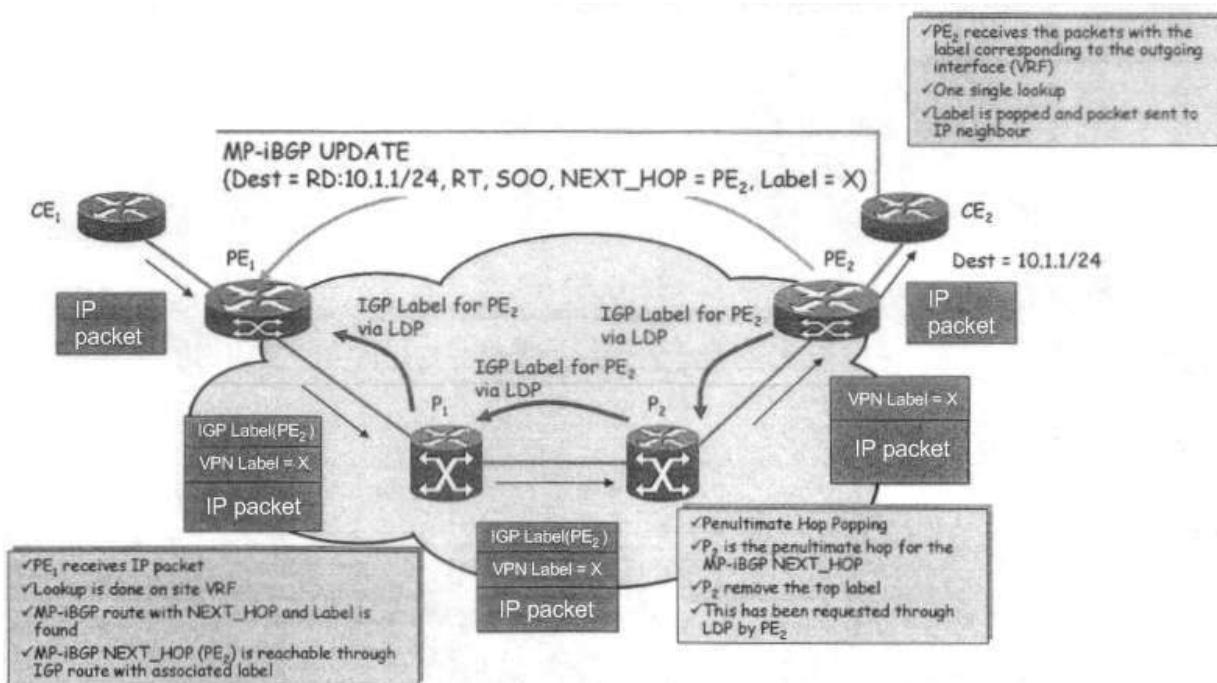
Step #1: A VPN label is assigned to every VPN route by the egress PE router.

Step #2: The VPN label is advertised to all other PE routers in an MP-iBGP UPDATE.

Step #3: A label stack is built in VFR table.

## 7.30 MPLS as a Forwarding Mechanism

The next example shows two sites within a particular VPN, where each site is represented just by its CE router ( $CE_1$  and  $CE_2$ ). Both  $PE_1$  and  $PE_2$  are configured with the appropriate Route Distinguisher to be used for that VPN, as well as with the appropriate Route Target Extended Community to be used when exporting routes into the provider's BGP and when importing routes from the provider's BGP. On  $PE_1$ , the interface that connects  $PE_1$  to  $CE_1$  is associated (at provisioning time) with the forwarding table of that VPN.



When PE<sub>2</sub> receives from CE<sub>2</sub> a route with reachability information 10.1.1/24; PE<sub>2</sub>:

- converts the reachability information of that route from plain IP to VPN-IP addresses,
- attaches the route target and SOO extended community attributes, and
- exports this route into the provider's BGP.

The NEXT\_HOP BGP attribute of this route is set to the address of PE<sub>2</sub> (usually, this address is the loopback address of the router). In addition to all the conventional BGP information, the route also carries a label that is associated with that VPN-IP route and this information is distributed to PE<sub>1</sub> using MP-iBGP (as shown by the upper line). When PE<sub>1</sub> receives the route, PE<sub>1</sub> converts the route from VPN-IP to IP and uses it to populate the forwarding table associated with the VPN. In addition, there is an LSP from PE<sub>1</sub> to PE<sub>2</sub>, which is associated with a route to PE<sub>2</sub> and is established and maintained by either LDP or MPLS traffic engineering. The route distributed via MP-iBGP carries, as the NEXT\_HOP attribute, the address of PE<sub>2</sub>, and that the route to that address is provided via the provider's intradomain routing. So it is the address of PE<sub>2</sub> (carried in the NEXT\_HOP attribute) that provides coupling between the provider's internal routing (e.g., routing to PE<sub>2</sub>) and the VPN routes (e.g., routing to 10.1.1/24). At this point the VPN forwarding table on PE<sub>1</sub> contains a route for 10.1.1/24 and a label stack where the inner label is the label that PE<sub>1</sub> receives via BGP and the outer label is the LDP-label associated with a route to PE<sub>2</sub>. Assume that CE<sub>1</sub> sends a packet with destination address 10.1.1.1. When the packet arrives at PE<sub>1</sub>, PE<sub>1</sub> determines the appropriate VRF and then performs the lookup in that table. As a result of the lookup, PE<sub>1</sub> attaches two labels to the packet and sends the packet to P<sub>1</sub>. P<sub>1</sub>, in turn, uses the outer label when making its forwarding decision and forwards the packet to P<sub>2</sub>. Since P<sub>2</sub> is the penultimate hop with respect to the LSP associated with a route to PE<sub>2</sub>, P<sub>2</sub> pops the outer label before sending the packet to PE<sub>2</sub>. When PE<sub>2</sub> receives the packet, it uses the label carried by the packet (the label that PE<sub>2</sub> distributed to PE<sub>1</sub> via BGP) to make its forwarding decision. PE<sub>2</sub> strips the label and sends the packet to CE<sub>2</sub>.

### 7.31 Scalability

To appreciate the scalability gain provided by the MPLS hierarchy of routing knowledge, consider the example of a service provider network that consists of 200 routers (both PE and P routers), which supports 10,000 VPNs, with each VPN having on average 100 routes. Without the use of the MPLS hierarchy of routing knowledge each P router would need to maintain  $10,000 \times 100 = 1,000,000$  routes. With the MPLS hierarchy of routing knowledge, each P router would need to maintain only 200 routes. The amount of routing peering that a CE router has to maintain is constant and therefore independent of the total number of sites within a VPN, thus allowing support of VPNs with sites numbering in the hundreds or thousands. The amount of configuration changes needed when adding or deleting a site is constant and therefore is independent of the total number of sites within a VPN.

Scaling properties with respect to handling routing information:

- *First*, by employing the MPLS hierarchy of routing knowledge, P-routers don't maintain any VPN routing information - the VPN routing information is maintained only on the PE routers.
- *Second*, a PE router has to maintain only the routing information for the VPNs whose sites are directly connected to that PE router - it doesn't have to maintain routes for all the VPNs supported by a service provider. If the volume of VPN routing information on a particular PE router gets too high for that router, we can add a new PE router and move some of the VPNs from the old router to the new one.

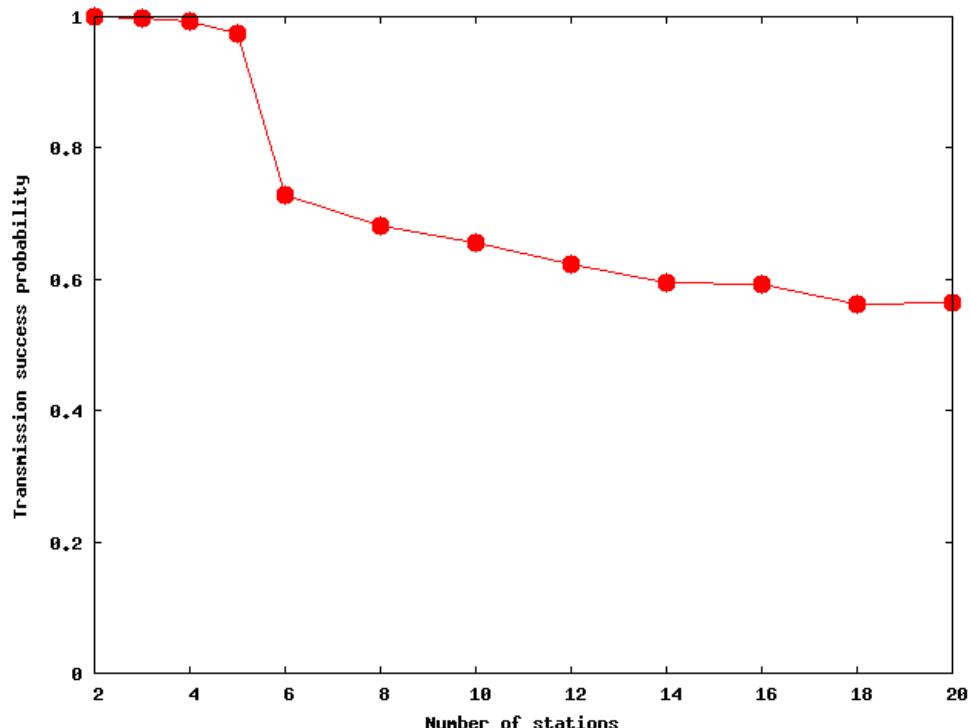
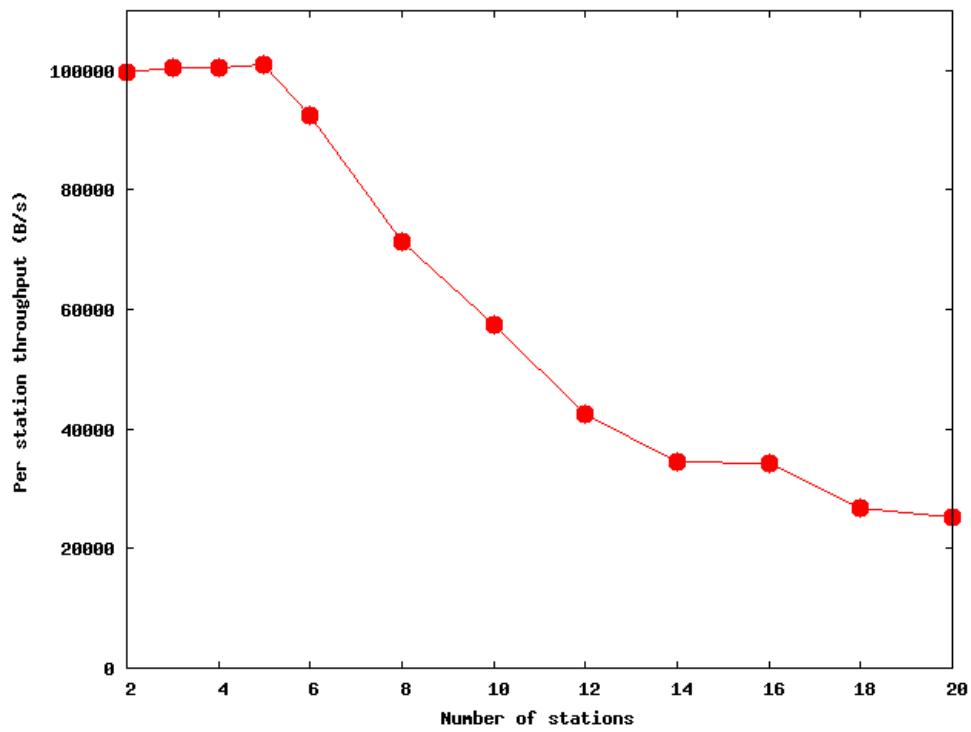
BGP Route Reflectors are useful to eliminate the requirement of a complete mesh of BGP sessions among all the PE routers - with BGP Route Reflectors, a PE router would need to maintain BGP sessions with just a few Route Reflectors. To avoid a situation where a particular Route Reflector would be required to handle routing information for all the VPNs supported by a provider, we partition Route Reflectors among VPNs supported by the provider. As a result, if the volume of VPN routing information maintained by a particular set of Route Reflectors gets too high, we can add a new set of Route Reflectors and move some of the VPNs from the old to the new set. Note that there is no single component within a service provider network that is required to maintain all the routing information for all the VPNs supported by the provider. Consequently, the VPN routing capacity of the service provider network isn't bound by the capacity of any individual component, which results in virtually unlimited routing scalability.

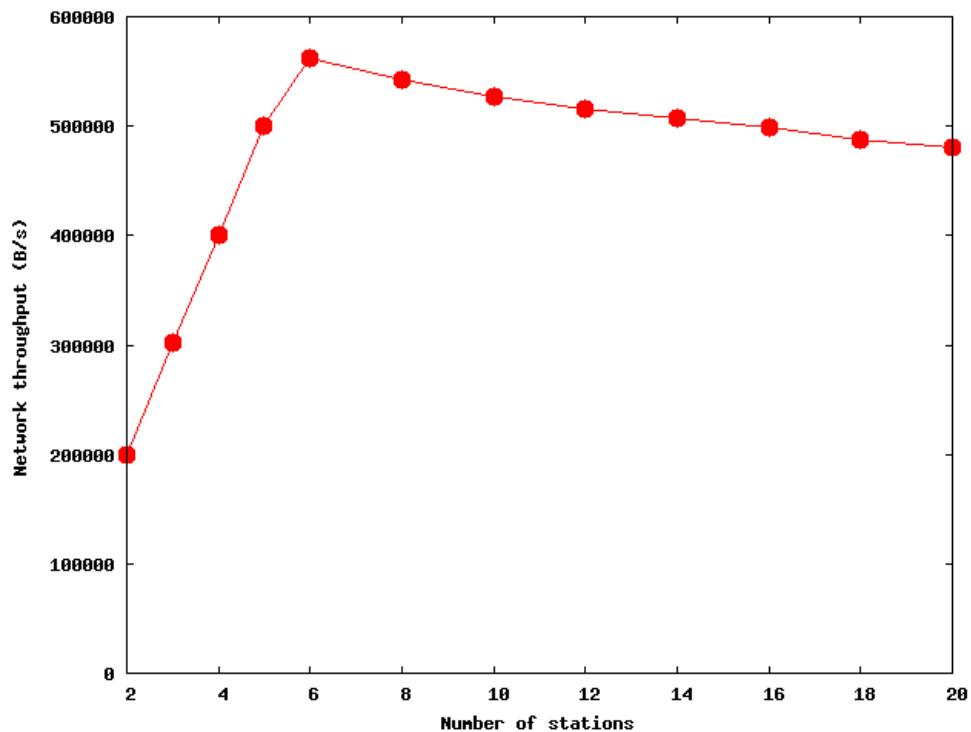
## 8 802.11e

### 8.1 QoS Limitations of 802.11 DCF

The *DCF* and *PCF* medium access functions do not include support for Quality of Service (QoS). As a matter of fact, they are unable to provide applications with QoS guarantees, both differentiated and parameterized. Such limitations are inherent of the access functions. In DCF medium access function there is *no admission control*: the performance significantly degrades when too many stations access the medium at the same time, due to several retransmissions, which consume the medium. Also, there is *medium capture effect*, due to the fact that when a station eventually wins the contention, it resets the contention window, thus gaining a higher chance of winning the contention again. Consider an IEEE 802.11a WLAN with a number of stations increasing from 2 to 20. Each station has a flow of constant size packets (1000 B) generated at fixed intervals. The offered load is 100 kB/s. The physical transmission rate is 6 Mb/s.

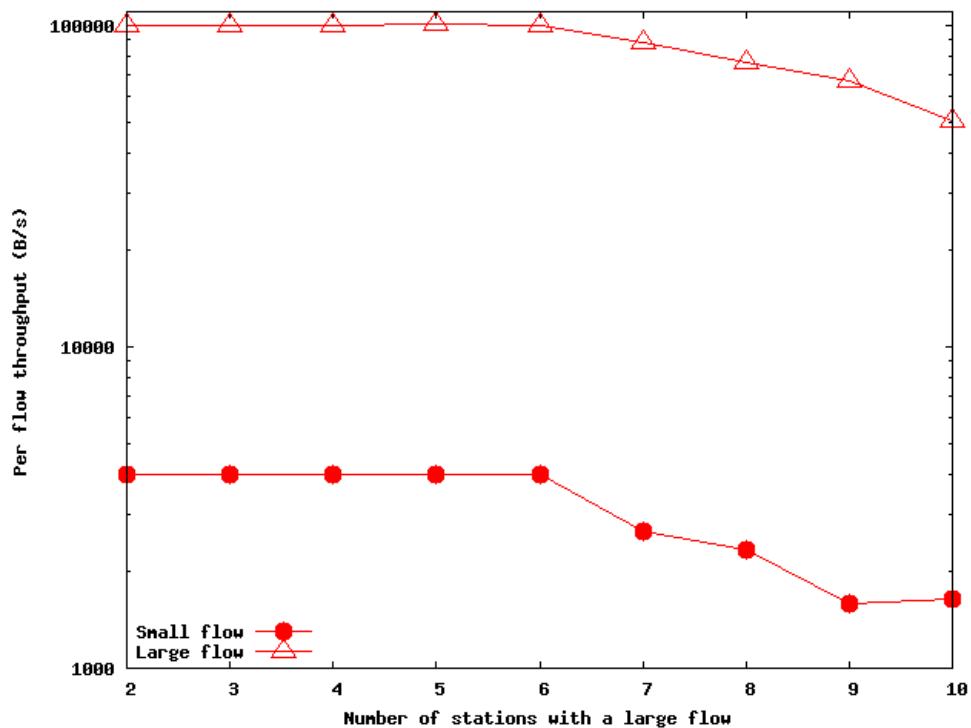
QoS Limitations of DCF:

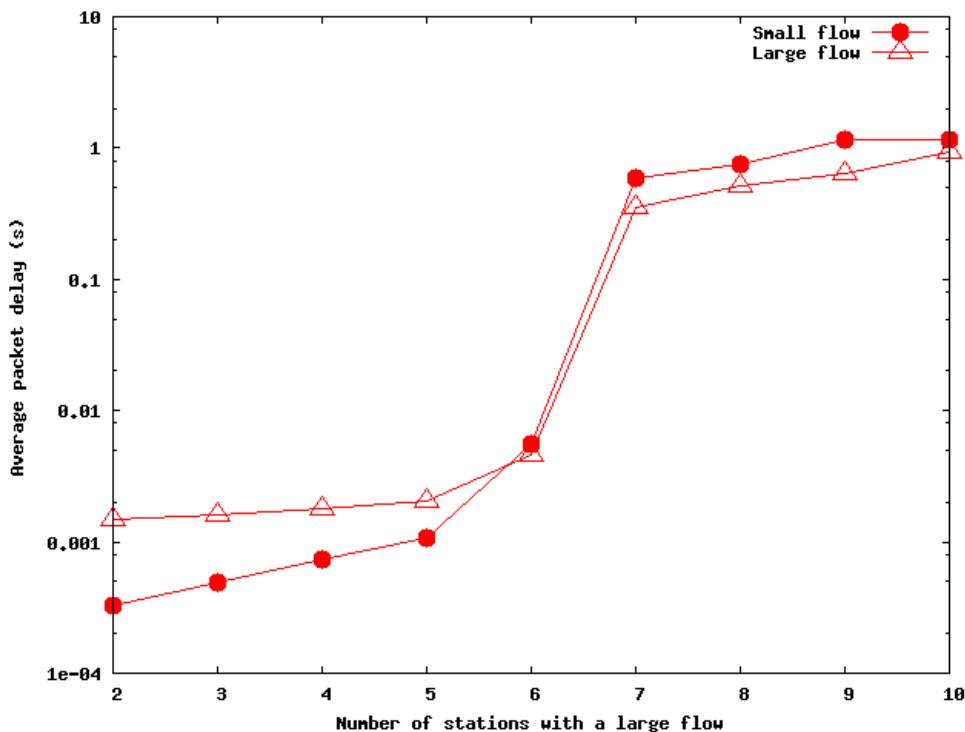




When the number of stations is smaller than or equal to 5, for each station the carried load is equal to the offered load, which entails zero packet loss and very small delays. When the number of stations becomes greater than 5, the carried load begins dropping, which causes an increasing performance degradation. Admission control is needed to solve this issue.

In DCF there is *no isolation*: more aggressive stations try to access the medium more often than stations with sporadic traffic, hence they obtain more resources. Furthermore, DCF is known to achieve *time fairness*: over long time scales, all the stations backlogged transmit the same number of frames in the unit of time. This way, stations with small packets or employing high transmission rates are penalized. Consider an IEEE 802.11a WLAN with a number of stations increasing from 2 to 11. One station has a “small” flow of constant size packets (40 B) generates at fixed intervals. The offered load of a “small” flow is 4 kB/s. The other stations have a “large” flow of constant size packets (1000 B) generated at fixed intervals. The offered load of a “large” flow is 100 kB/s. The physical transmission rate is 6 Mb/s.



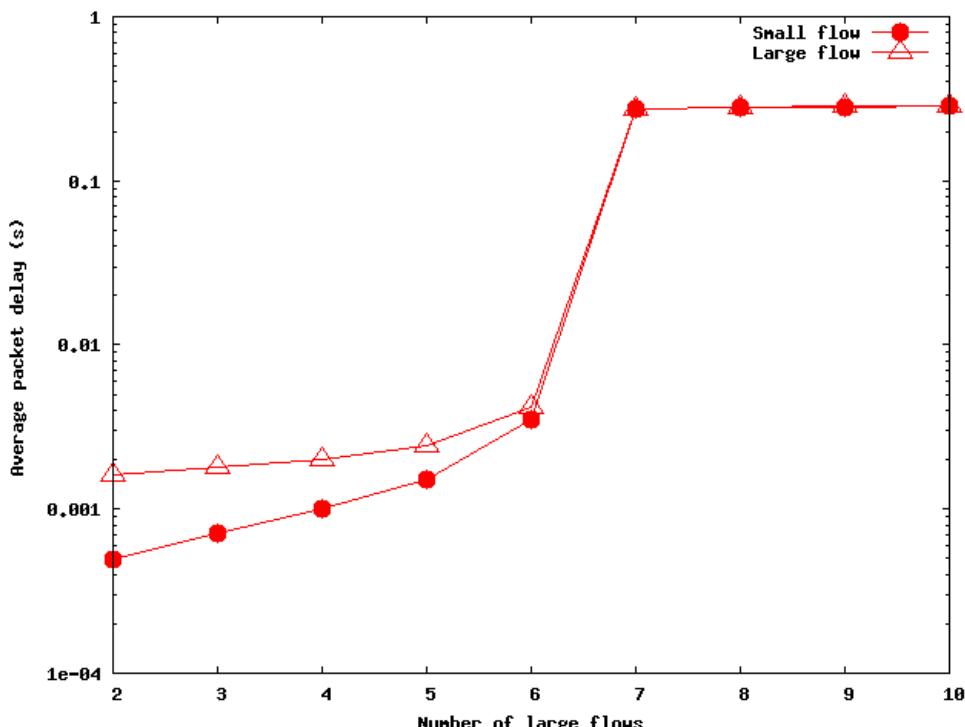


Even though the station with the “small” flow requires significantly less resources than the “large” flows (the offered load being 1/25), the former experiences the same performance degradation as the latter, in terms of both throughput and packet delay. A mechanism to differentiate small vs. large flows is needed to solve this issue.

There is *intra-station unfairness*: if a station has more traffic flows with different QoS requirements, resource sharing may be unfair. For instance, consider an IEEE 802.11 user using VoIP while downloading her e-mails. VoIP traffic is much more precious than e-mail, because the call quality highly depends on delays and losses of speech samples. However, since e-mail uses TCP, it is likely that it will get most of the available bandwidth, due to traffic multiplexing into the same buffer of the IEEE 802.11 device. Consider an IEEE 802.11a WLAN with two stations, one of which is not generating data. The active stations has:

- a “small” flow of constant size packets (40 B) generated at fixed intervals. The offered load of a “small” flow is 4 kB/s,
- a variable number of “large” flows of constant size packets (1000 B) generated at fixed intervals, increasing from 2 to 10. The offered load of a “large” flow is 100 kB/s.

The physical transmission rate is 6 Mb/s.

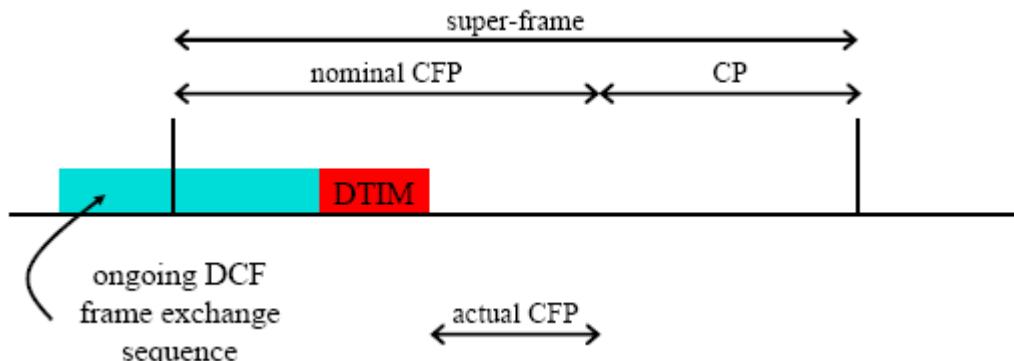


An IEEE 802.11 station has a single buffer to keep packets from all its traffic flows. When the

number of “large” flows increases beyond 6, the network is not able anymore to completely dispatch the offered load. Therefore, both flows experience packet loss, due to buffer overflow, and long packet delays, due to queueing. A mechanism to isolate within the same station flows with different characteristics is needed to solve this issue.

## 8.2 QoS Limitations of 802.11 PCF

There is *no QoS parameters negotiation*: all the flows that are associated to an AP via PCF are treated the same. However, different application may have very different traffic characteristics (e.g. packet size and generation interval) and QoS requirements (e.g. delay and bandwidth). The limitation above makes it impossible to have fine-tuned admission control implemented in the AP. The beginning of the CF period can be delayed due to stations using DCF, which makes the CF period variable.

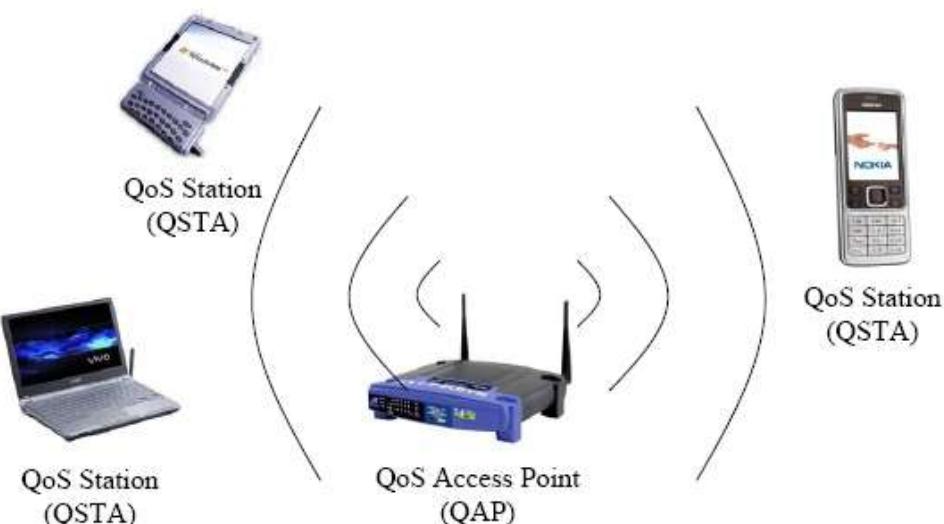


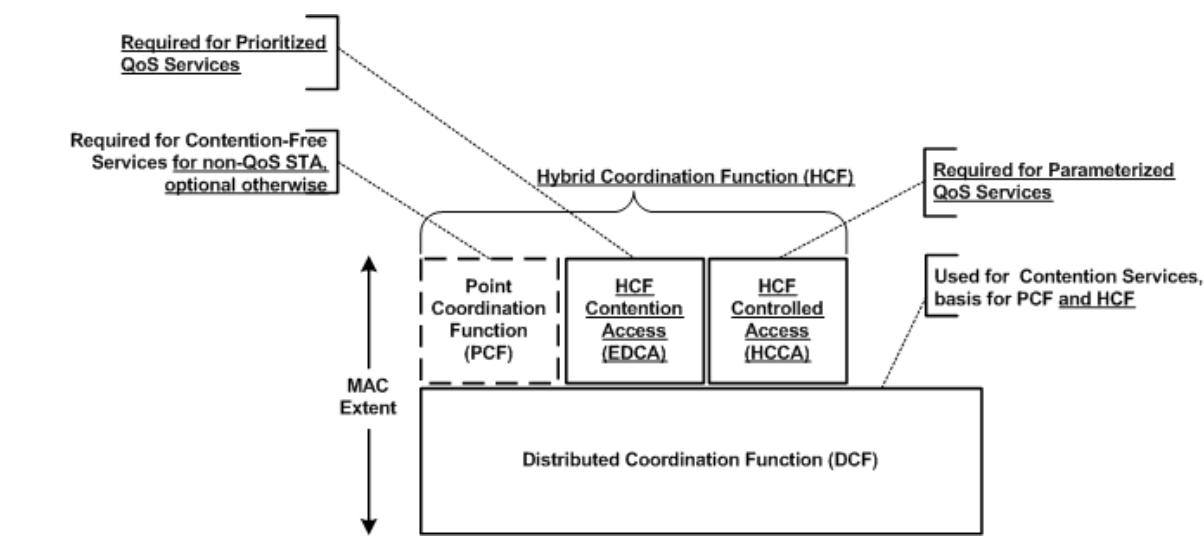
Resource allocation is not flexible and inefficient: one poll for each station, which is entitled to transmit exactly one MSDU. If a station is polled too often, resources are wasted due to unnecessary opportunities to access the medium during the CFP. If a station is polled too infrequently, it is not able to transmit its backlog completely during the CFP. What is worse, the polling period depends on the number of stations using PCF and on how much traffic they have!

## 8.3 IEEE 802.11e MAC

The purpose of Task Group E is to enhance the current IEEE 802.11 MAC to expand support for applications with Quality of Service requirements, and in the capabilities and efficiency of the protocol. The amendment to the IEEE 802.11 standard which includes QoS support was published on 2005 as IEEE 802.11e, and is now included in the IEEE 802.11-2007 document. The Wi-Fi Alliance proposed a sub-set of the IEEE 802.11e enhancements as part of the Wi-Fi Multimedia (WMM) interoperability certification.

QoS Basic Service Set (QBSS):





IEEE 802.11e-specific medium access functions:

- Enhanced Distributed Channel Access (EDCA),
- HCF Controlled Channel Access (HCCA) [optional].

IEEE 802.11e optimized mechanisms:

- Direct Link Setup (DLS) [optional],
- Block Acknowledgment (BA) [optional].

## 8.4 Differentiated vs Parametrized QoS Guarantees

*Differentiated QoS guarantees* means that the applications that are allowed to access the network are given different treatment, depending on their *priority*. An application that is given a higher priority will receive a better service than another application with a lower priority. However, no absolute guarantees are provided. Therefore, it can happen that the QoS experienced by the users of both applications is not satisfactory.

*Parameterized QoS guarantees* means that any application that is given the right to access the network is guaranteed a minimum level of service, which is specified by means of a set of QoS parameters. This is true regardless of the other established applications or the network load. In other words, absolute guarantees are provided. While *admission control* can be optional with differentiated QoS, it is clearly compulsory with parameterized QoS. In wireline networks, the QoS experienced by an application mostly depends on the amount of resources that are provisioned along the path from the source to the destination. It is thus possible to provide fine-grained QoS guarantees in terms of, e.g., bandwidth and delay. On the other hand, we have to keep in mind that in wireless network the performance is severely affected by the current wireless channel conditions. Strictly speaking it is thus not possible to actually provide absolute QoS guarantees of any kind.

## 8.5 Transmission Opportunity (TXOP)

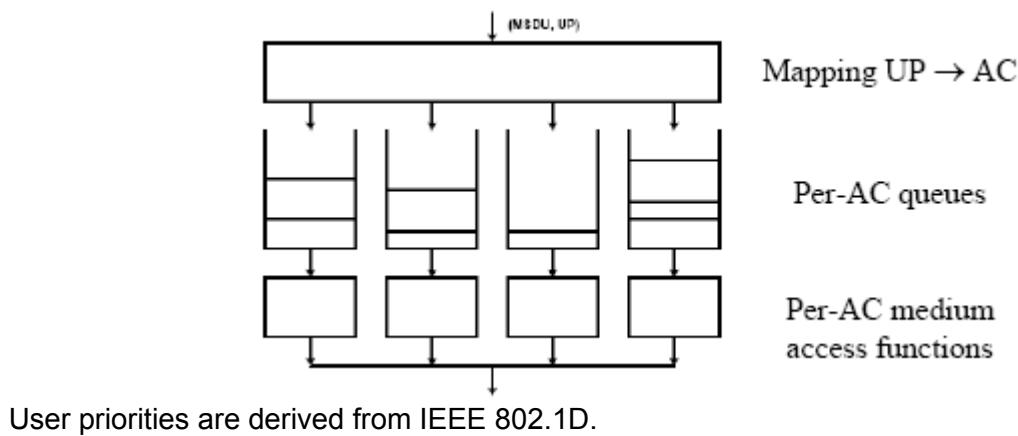
The basic unit of allocation of the right to transmit onto the wireless channel is the *transmission opportunity (TXOP)*. A TXOP is acquired by a QSTA by:

- either winning an EDCA contention (EDCA TXOP);
- or, receiving a QoS (+)CF-Poll frame (HCCA TXOP).

A TXOP is characterized by a *limit duration*, which is advertised by the QAP implicitly (EDCA) or explicitly (HCCA).

## 8.6 Enhanced Distributed Channel Access

EDCA is tailored to provide prioritized QoS guarantees, with isolation of traffic flows with different *user priorities (UPs)*, which are mapped to different *access categories (ACs)*.



User Priority (UP)	Access Category (AC)	Designation (Informative)
1	AC_BK	Background
2	AC_BK	Background
0	AC_BE	Best Effort
3	AC_BE	Best Effort
4	AC_VI	Video
5	AC_VI	Video
6	AC_VO	Voice
7	AC_VO	Voice

Increasing priority ↓

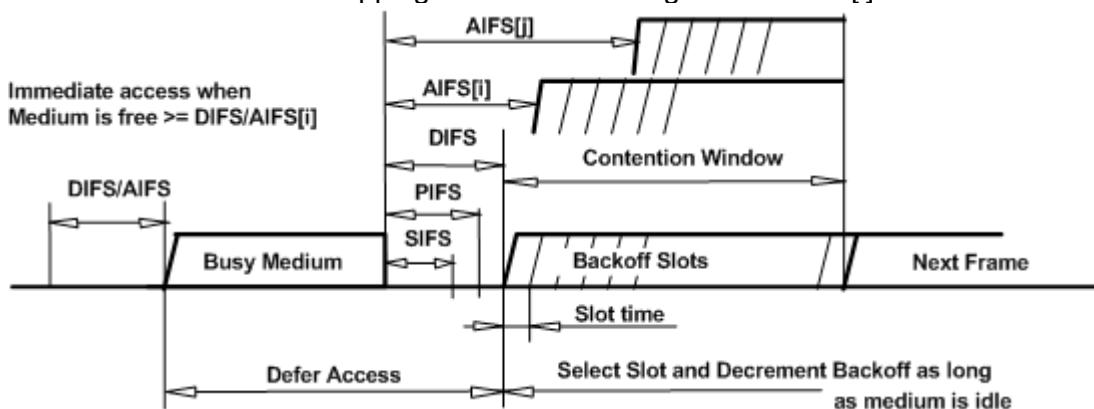
Each AC  $i$  is characterized by four parameters:

1. AIFSN[i]: Arbitration Inter-Frame Space Number,
2. CWmin[i]: Minimum Congestion Window,
3. CWmax[i]: Maximum Congestion Window,
4. TXOPlimit[i]: TXOP limit duration.

AIFSN[i] is used to derive the Arbitration Inter-Frame Space:

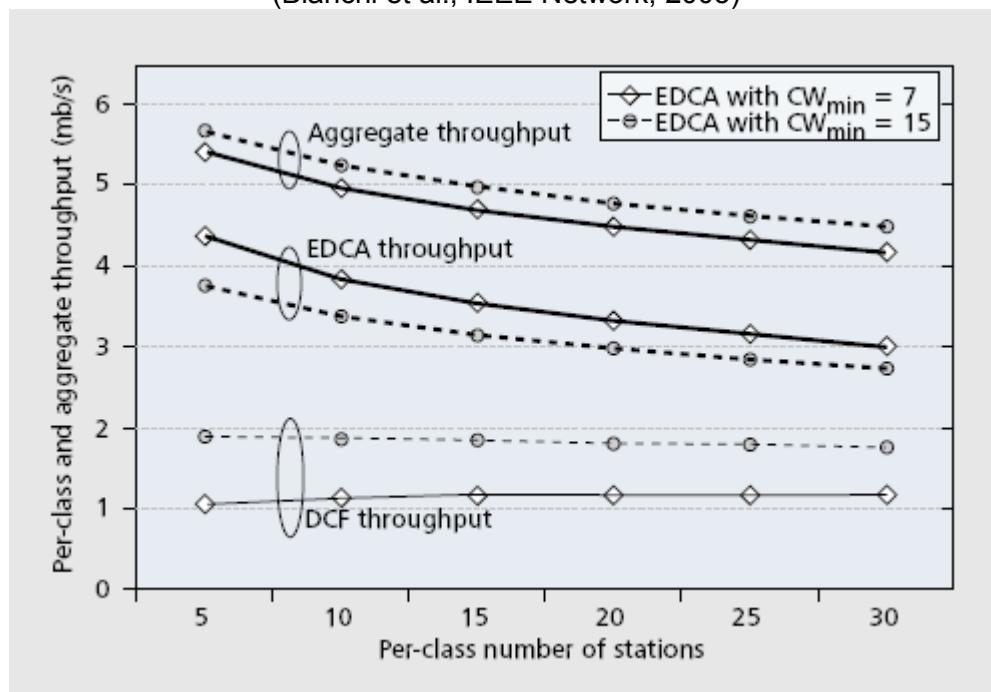
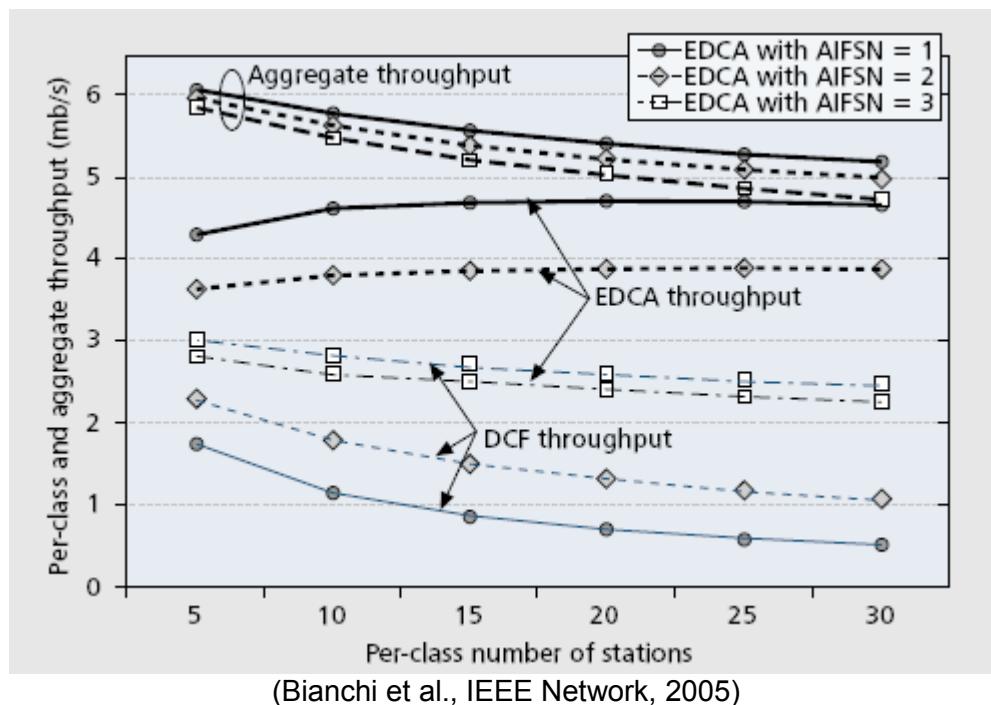
$$AIFS[i] = SIFS + AIFSN[i] \times aSlotTime$$

where  $aSlotTime$  depends on the physical layer employed. ACs with a smaller AIFSN[i] have a number of back-off slots non-overlapping with those with a greater AIFSN[i].

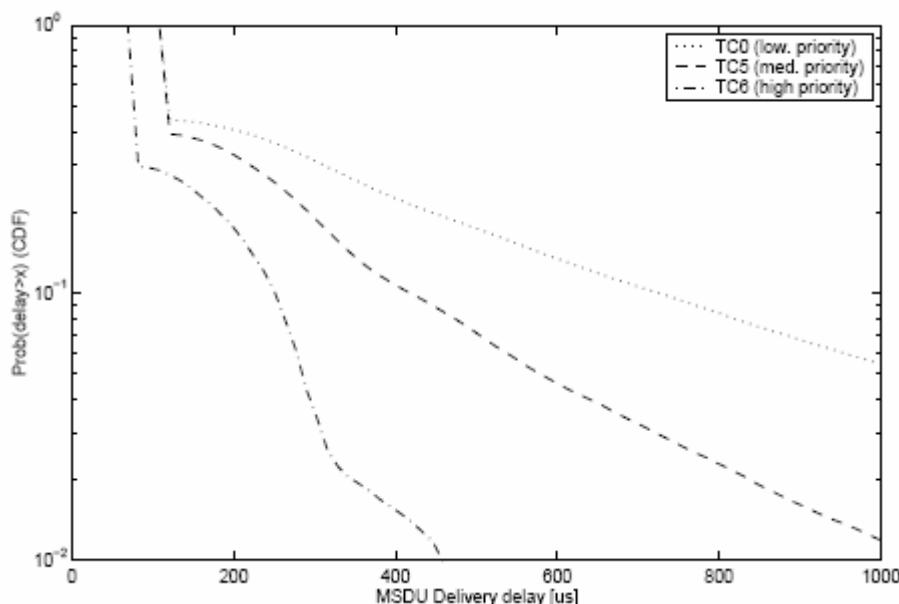


ACs with a smaller CWmin[i] have a higher chance of transmitting after while backing off, thus they are more aggressive when trying to win access to the medium. On the other hand, the CWmax[i] differentiating parameter has been shown not to affect significantly the performance. In fact, it only bounds the maximum size of the contention window, which is however hardly reached in practice because it is likely that the pending MAC frame will be dropped before that (and the CW reset to the initial value).

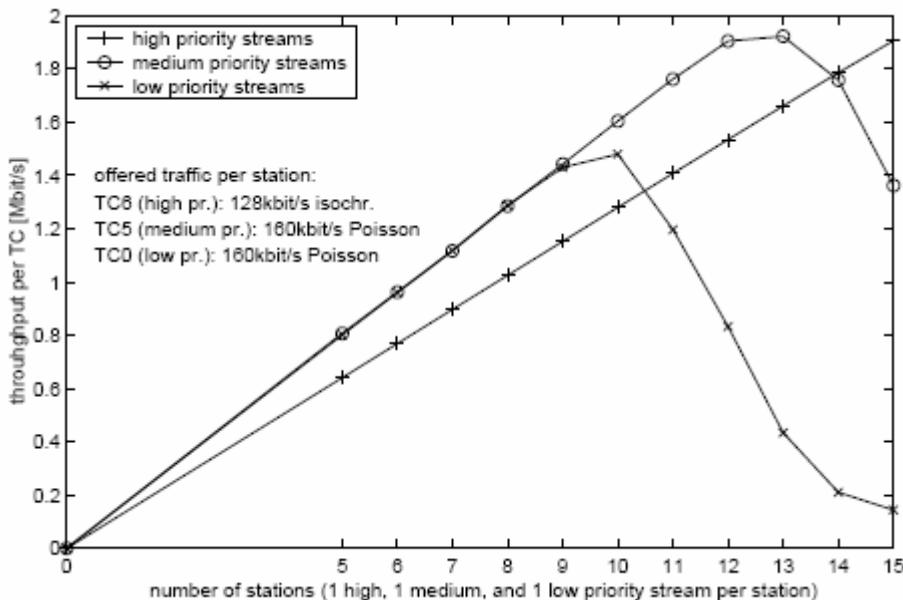
(Bianchi et al., IEEE Network, 2005)



(Mangold et al., European Wireless, 2002)



(Mangold et al., European Wireless, 2002)



Finally, since the right to transmit is granted based on the TXOP limit duration, the channel is shared fairly, in time, among traffic flows of the same AC. This is true regardless of the size of packets of the traffic flow and the physical transmission rate employed. In an infrastructure QBSS, the EDCA parameters are advertised regularly by the QAP in beacon frames.

Default set of EDCA parameters (ad-hoc QBSSs):

AC	CWmin	CWmax	AIFSN	TXOP limit		
				For PHYs defined in Clause 15 and Clause 18	For PHYs defined in Clause 17 and Clause 19	Other PHYs
AC_BK	aCWmin	aCWmax	7	0	0	0
AC_BE	aCWmin	aCWmax	3	0	0	0
AC_VI	(aCWmin+1)/2 - 1	aCWmin	2	6.016 ms	3.008 ms	0
AC_VO	(aCWmin+1)/4 - 1	(aCWmin+1)/2 - 1	2	3.264 ms	1.504 ms	0

aCWmin and aCWmax depend on the physical layer employed.

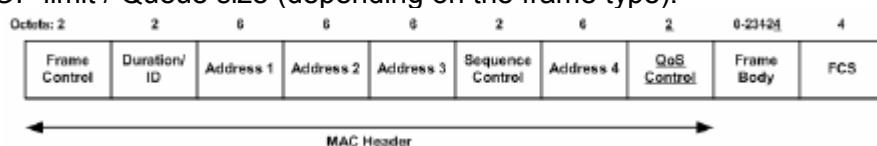
## 8.7 HCF Controlled Channel Access

HCCA is tailored to provide parameterized QoS guarantees, with isolation of micro-flows. Admission control is compulsory. Each QSTA may have up to eight established *Traffic Streams* (TSs) identified by:

- a TS identifier (TID), from 0 to 7,
- a set of traffic specifications (TSPEC),
- a set of traffic classification rules (TCLAS), which is used by the QSTA to classify the packets coming from the upper layer into the correct traffic streams.

Each MAC frame transmitted in a QBSS contains an additional QoS Control field:

- 3-bit TID,
- End-Of-Service-Period (EOSP) bit,
- 2-bit ack policy: normal ack, no ack, no explicit ack, block ack,
- 1 reserved bit,
- 8-bit TXOP limit / Queue size (depending on the frame type).

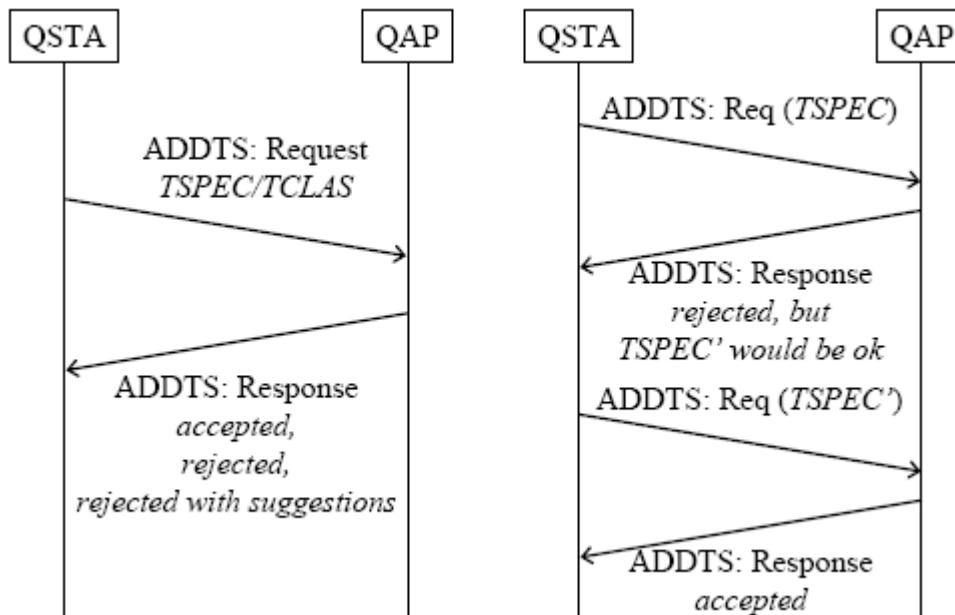


TSPEC parameters include:

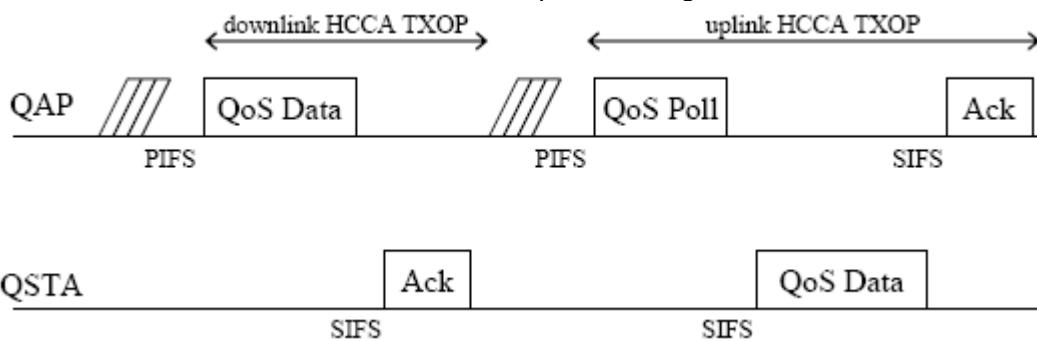
- Traffic characterization (dual token-bucket model)
  - Mean Data Rate
  - Peak Data Rate

- Maximum Burst size
- Traffic requirements
  - Maximum and Nominal MSDU size
  - Minimum PHY rate
  - Minimum Data Rate (not including overheads)
  - Delay Bound
  - ...

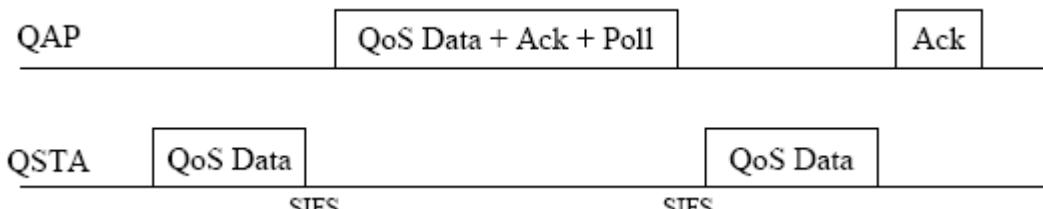
*Traffic characterization* is optional and can be used by the HC to schedule HCCA TXOPs efficiently, based on the specific characteristics of the traffic flows. *Traffic requirements* define the service that the QSTAs are requesting. If the HC accepts the TS requesting admission, it is then committed to provide the TS with a service which complies with the parameters specified under controlled channel conditions. The QAP cannot guarantee that no channel errors will occur!



The HC, which is located within the QAP, is able to provide parameterized service by granting contention-free HCCA TXOPs, both downlink and uplink, during either the CP and the CFP.



The QAP grabs the medium by waiting PIFS so that ongoing frame exchange sequences are not interrupted, and it has higher priority than any DCF or EDCA transmission. QoS frames (data, poll, and ack) can be combined together to achieve better medium usage.

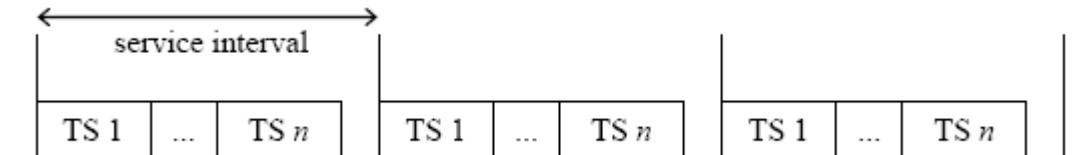


The QAP specifies the maximum duration of the uplink HCCA TXOP into the QoS Control field of the MAC header. The QSTA can never exceed this limit, but is allowed to release the medium even though part of the TXOP is left unused. If the QSTA does not have data to transmit for the polled TS, it responds to the poll from the QAP with a QoS Null frame, which does not carry user data. QoS Null messages can be considered as protocol overhead, and should be avoided whenever possible. To do so, QSTAs may piggyback the buffer occupancy (in bytes) of a given TS on any outgoing MAC frame of that TS. However, the IEEE 802.11e standard does not specify how

the QAP should exploit this piece of information, which is an optional feature of HCCA. Similarly, the IEEE 802.11e standard does not specify how the QAP should:

- use the traffic specifications provided by the QSTAs,
- enforce scheduling of HCCA TXOPs so that the traffic requirements are met,
- implement the Call Admission Control function,
- poll QSTAs so as to optimize resource usage.

However, it does define a *sample HCCA scheduling* function for informational purposes. The sample HCCA scheduler defines a TDM-like sequence, so that each TS is granted a fixed amount of capacity every period, the latter being the same for all admitted TSs. The grant duration of each TS is determined so that packets of Nominal MSDU Size can be transmitted at the Mean Data Rate at the Minimum PHY Rate.



TSPEC parameters of TS i are:

- Mean Data Rate (MDR[i], in b/s),
- Nominal MSDU Size (NMS[i], in B),
- Delay Bound (DB[i], in s).

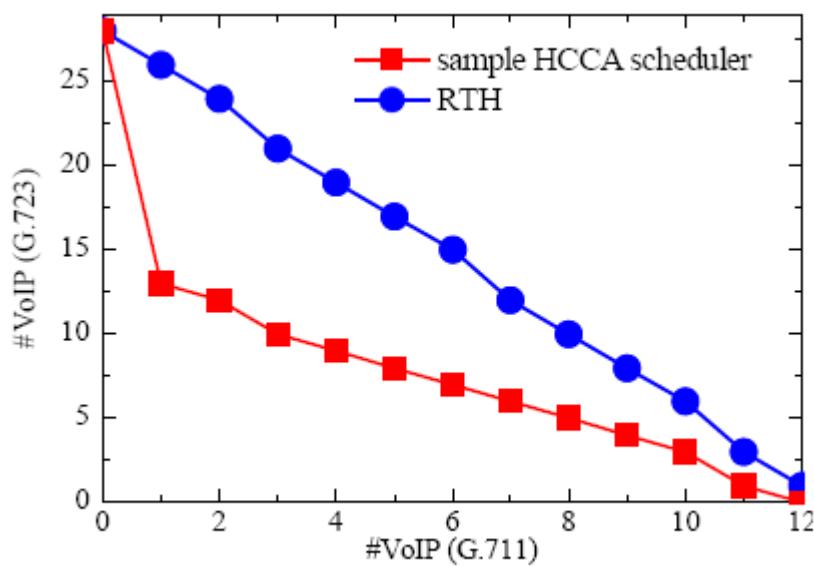
Service Interval (SI) is the minimum DB[j], for any TS j. TXOP[i] is equal to the time to transmit all the MAC frames arriving during SI, provided that the application generates packets of size NMS[i] at rate MDR[i], including any protocol overhead such as acknowledgments, polls, and inter-frame spaces. A new TS k can be admitted if the following is true:

$$\frac{TXOP_k}{SI'} + \sum_{i=1}^{k-1} \frac{TXOP'_i}{SI'} \leq 1$$

Where SI' is the new SI duration, should TS k be admitted; and TXOP'\_i are the new TXOP values, should TS k be admitted. While the sample HCCA scheduler fits very well constant bit-rate (CBR) traffic streams with the same delay bound values, it cannot deal efficiently variable bit-rate (VBR) flows because:

1. if the Mean Data Rate is selected equal to the average traffic rate, then packets belonging to large bursts may experience huge delay;
2. if the Mean Data Rate is selected equal to the peak traffic rate, then admission control is overly pessimistic.

Additionally, it cannot deal efficiently CBR flows with different delay bound values, because the queue length indication is ignored, thus inactive TSs can be unnecessarily polled, hence resources are wasted.

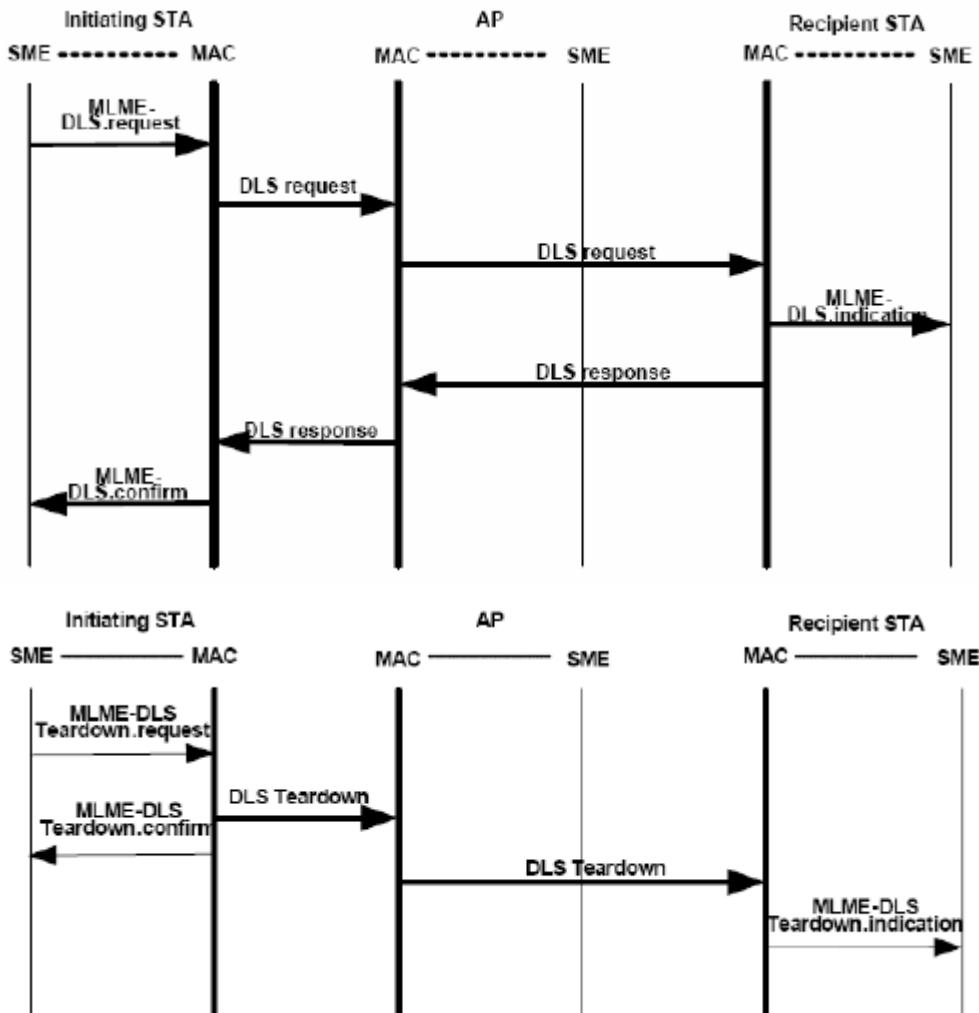


Cicconetti et al., Computer Networks, 2007

## 8.8 Direct Link Setup

In an IQBSS, all communications between QSTAs must flow through the QAP. However, this

may unnecessarily consume resources, especially if the two QSTAs are close to one another. In such situations, the DLS, which is an optional mechanism of the IEEE 802.11e, allows the medium to be used more efficiently, by establishing a *direct link* between the QSTAs.



## 8.9 Block Acknowledgment

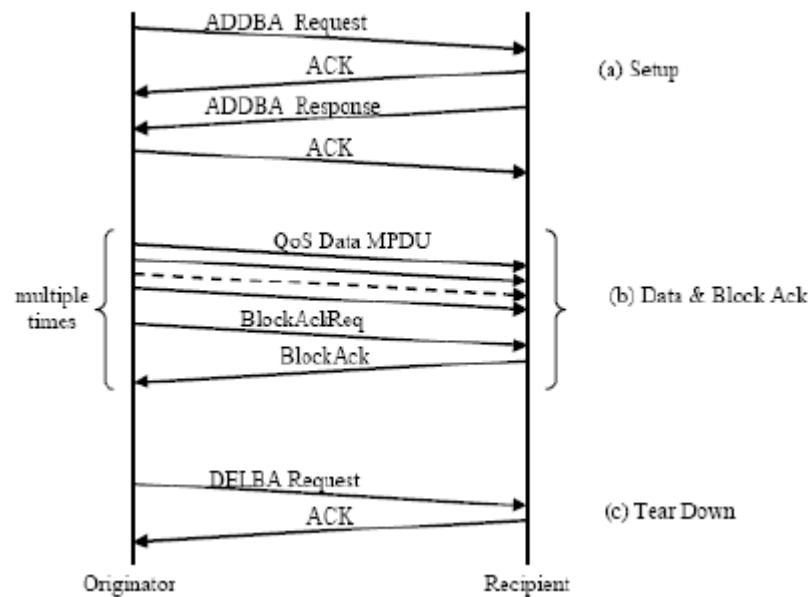
**Problem:** acknowledgments consume a significant portion of the frame exchange sequence, especially with small packets. Ratio between the ack duration and the total frame exchange sequence, with 40 bytes payload (e.g. TCP segment with no data or typical VoIP frame) are:

- IEEE 802.11a @ 6 Mb/s: 0.25
- IEEE 802.11a @ 54 Mb/s: 0.43
- IEEE 802.11b @ 1 Mb/s: 0.27
- IEEE 802.11b @ 11 Mb/s: 0.80

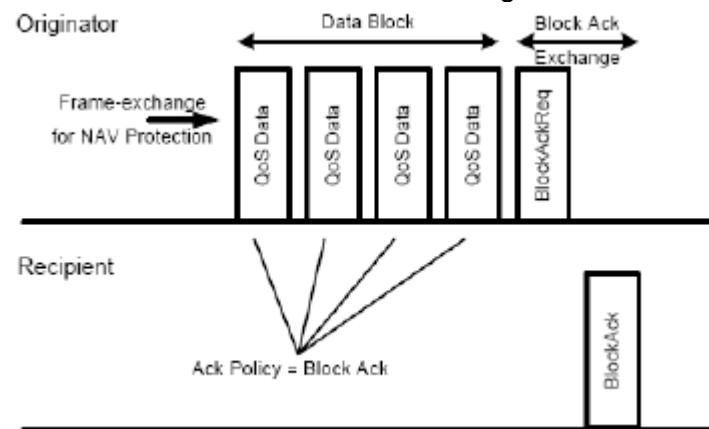
**Solution 1:** removing the acknowledgment and rely on upper layer functions for error recovery, if needed.

**Solution 2:** employing *block acknowledgment*, which is optional with IEEE 802.11e:

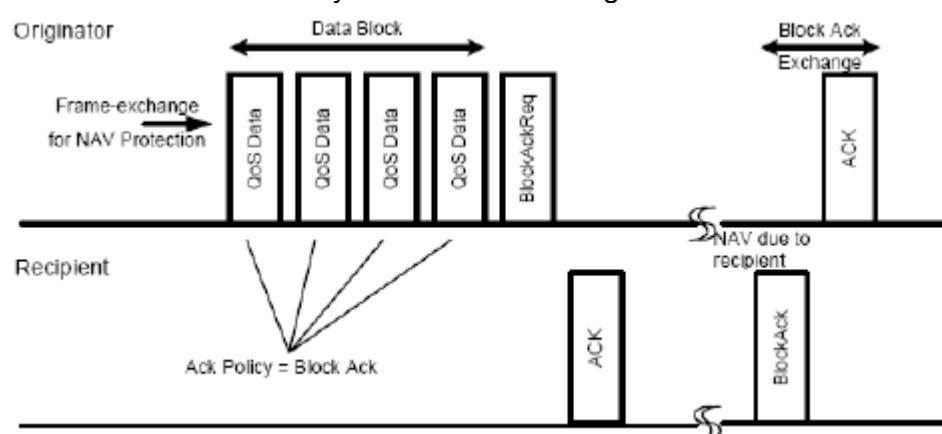
- *immediate ack*: a cumulative acknowledgment is sent by the receiver QSTA immediately after data transmission from the sender QSTA;
- *delayed ack*: a cumulative acknowledgment is sent separately so as to give enough time to the receiver QSTA for decoding and reassembling the MSDUs.



#### Immediate Block Acknowledgement:



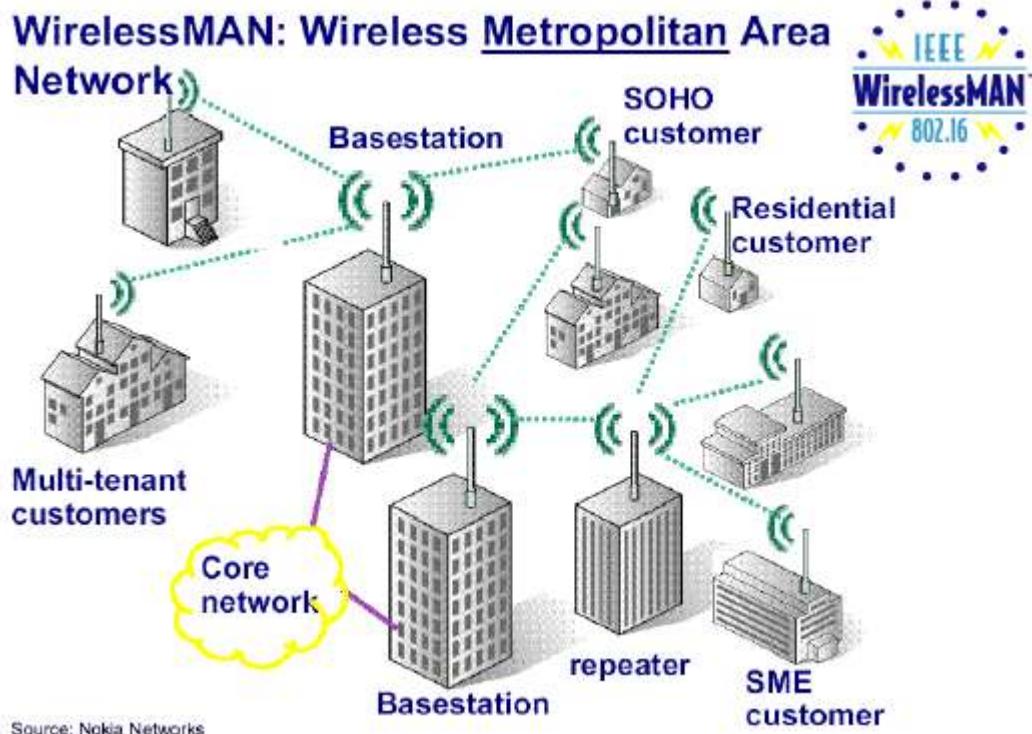
#### Delayed block acknowledgment:



# 9 WiMAX (IEEE 802.16)

## 9.1 Introduction

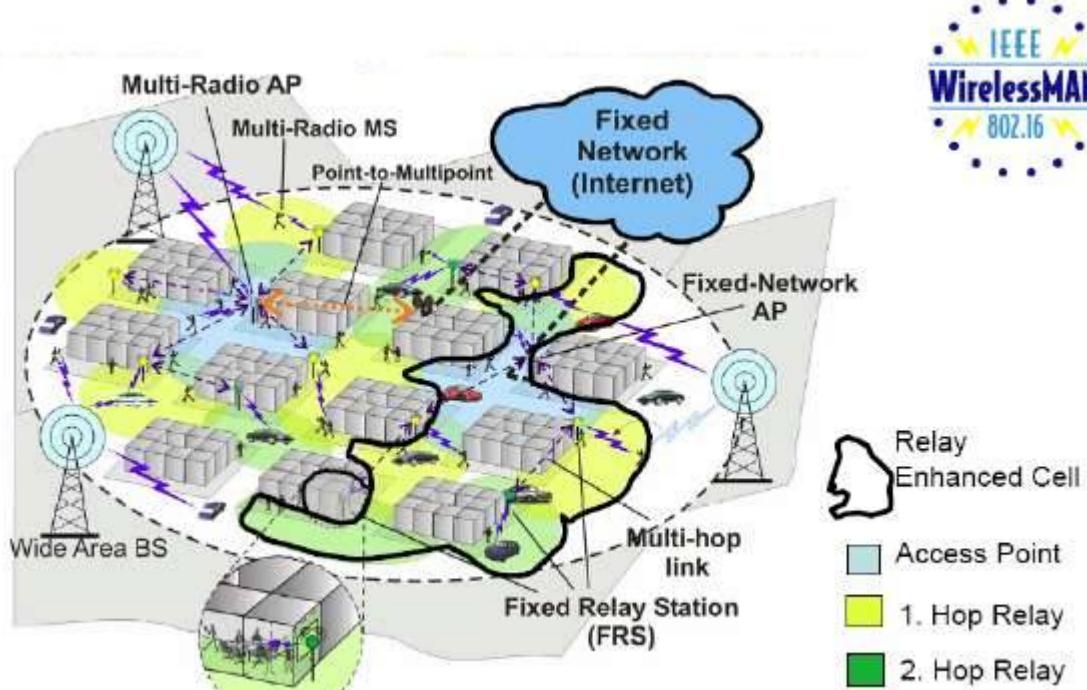
The original motivation (back to Y'00):



Original motivations for 802.16 were:

- A standard air interface for *fixed* Broadband Wireless Access (BWA):
  - to address the "first-mile/last-mile" connection in wireless metropolitan area networks,
  - alternative to wireline broadband access
  - interoperable multi-vendor BWA products.
- The "... standard specifies the air interface of fixed broadband wireless access systems *supporting multimedia services*."
- Multiple services, with *different QoS*, simultaneously!

The current driver is user mobility: *Mobile BWA*, as opposite to *Fixed BWA*.

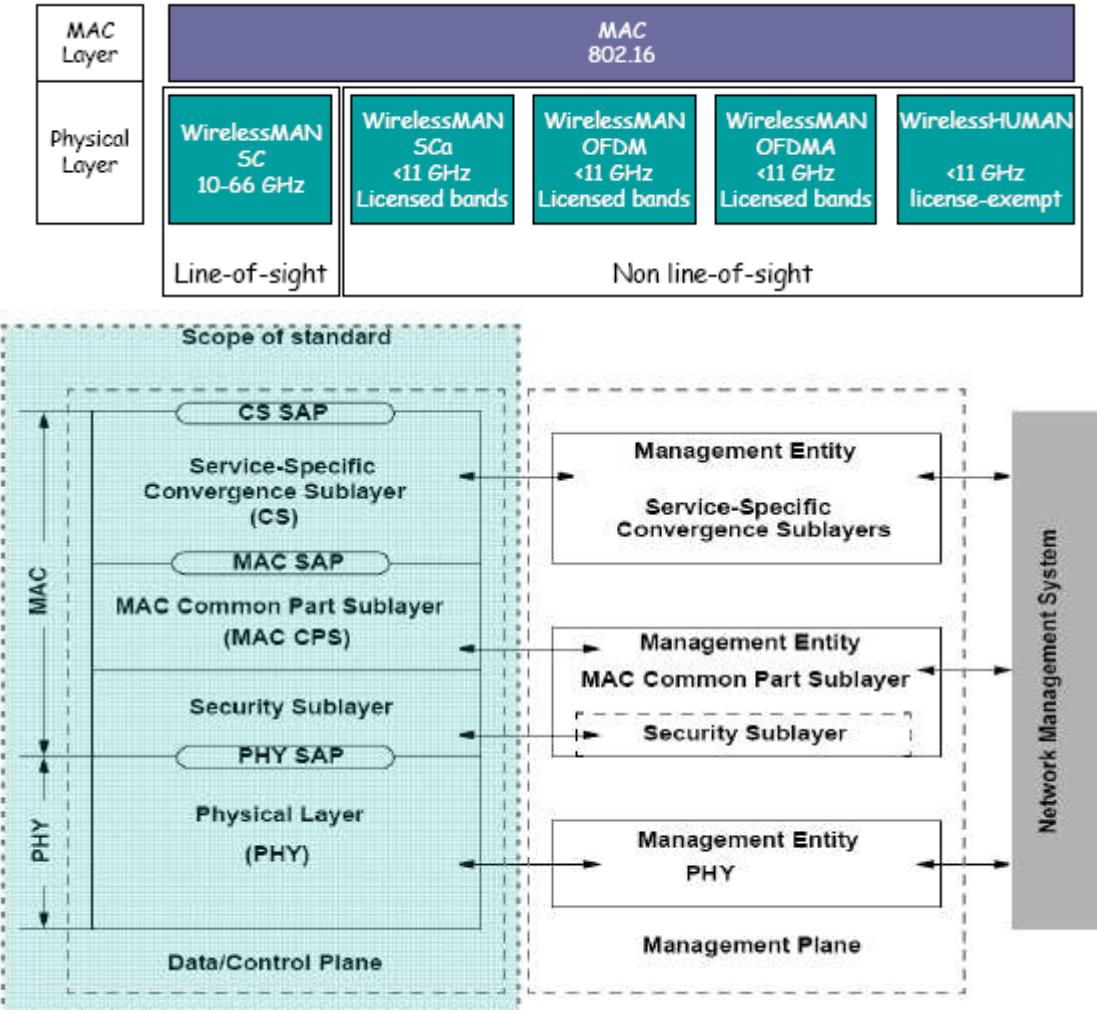


Source: Welke, Pabst, Schultz, A Mobile Broadband System based on Fixed Wireless Routers. Proc. ICCT 2003 Beijing, China 2003

IEEE 802.16 standards (publicly available at <http://standards.ieee.org/getieee802/802.16.html>):

- IEEE Std 802.16™-2004\* (Oct. 1st, 2004) last comprehensive revision (893 pp.):
  - Air Interface for Fixed Broadband Wireless Access Systems
  - Reference topology: Point-To-Multipoint (PMP) (mandatory) and Mesh (optional).
- IEEE Std 802.16e™-2005\* (Feb 28th, 2006) amendment to IEEE Std 802.16™-2004 (864 pp.):
  - Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands.

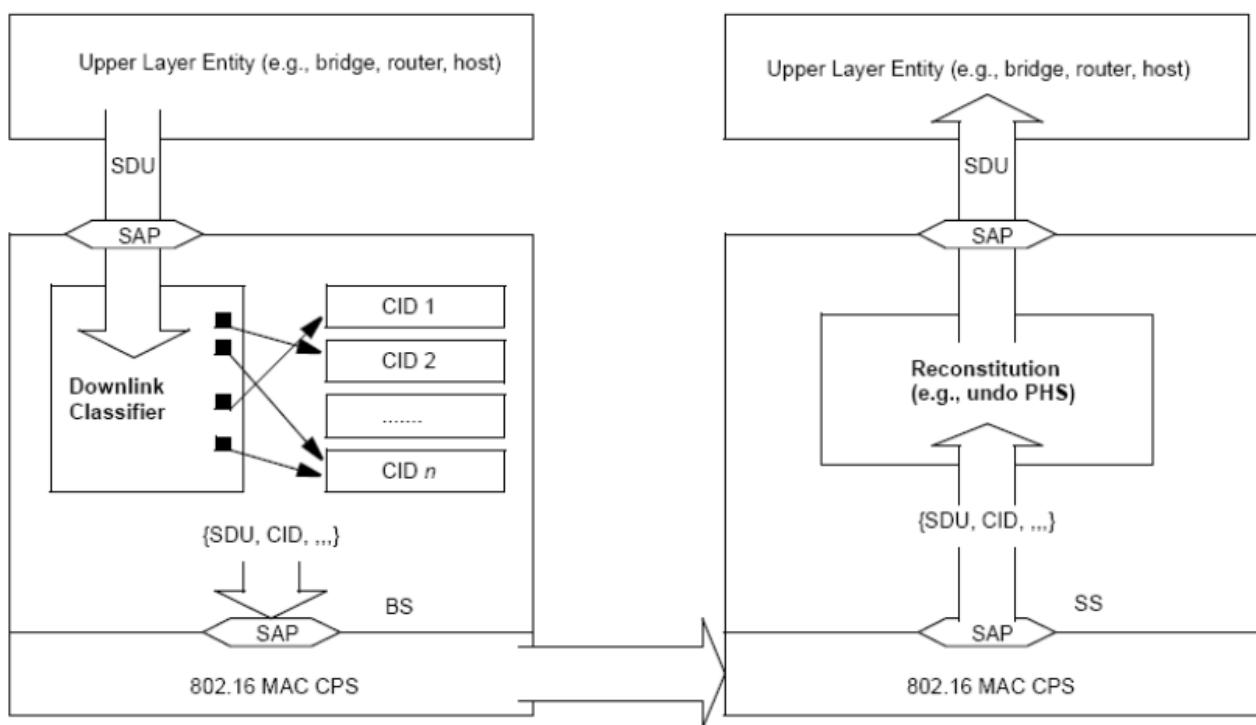
Reference model and scope:



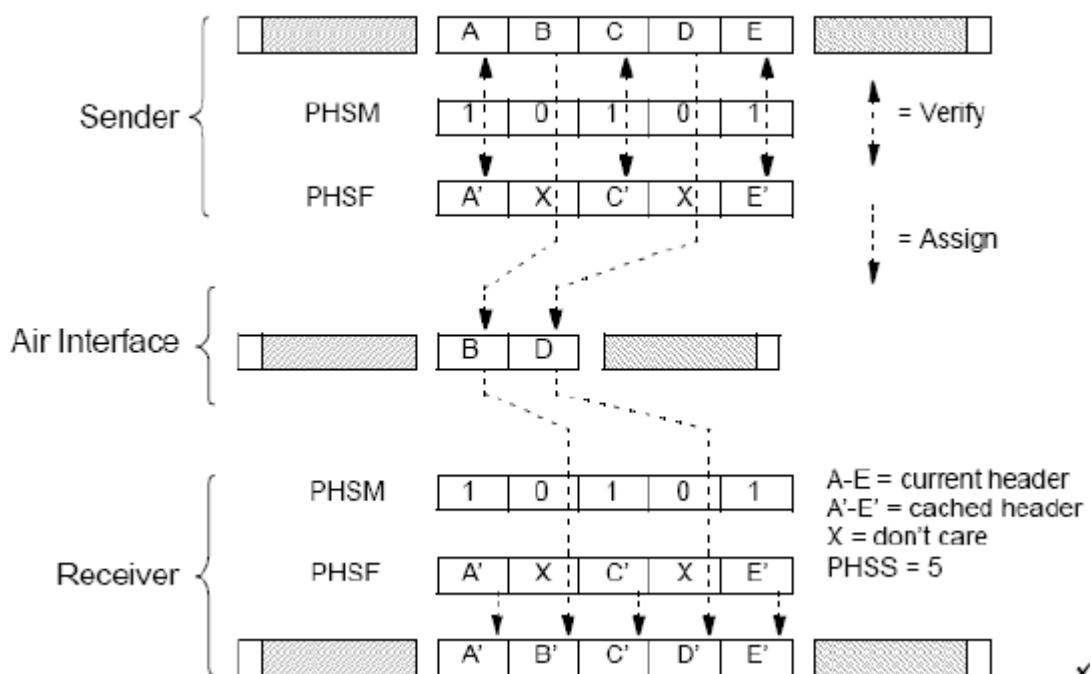
## 9.2 Service-Specific Convergence Sublayer

The Service-Specific Convergence Sublayer (CS) provides any transformation or mapping of external network data, received through the CS service access point (SAP), into MAC SDUs received by the MAC Common Part Sublayer (CPS) through the MAC SAP. It classifies external network service data units (SDUs) and map them to the proper MAC service flow identifier (SFID) and connection identifier (CID). It performs, if provisioned, *Payload Header Suppression (PHS)* and delivers CS PDUs to the appropriate MAC SAP. Multiple CS specifications are provided for interfacing with various protocols: cell-based (ATM) and packet-based (e.g. IP, PPP, Ethernet).

CS – Classification and CID Mapping:



CS – Payload Header Suppression:



CS performs selective suppression in the MAC SDU of header fields that remain static within a higher-layer session (e.g. IP addresses), while enabling transmission of fields that change from packet to packet (e.g. IP Total Length). Implementation of Payload Header Suppression (PHS) capability is optional!

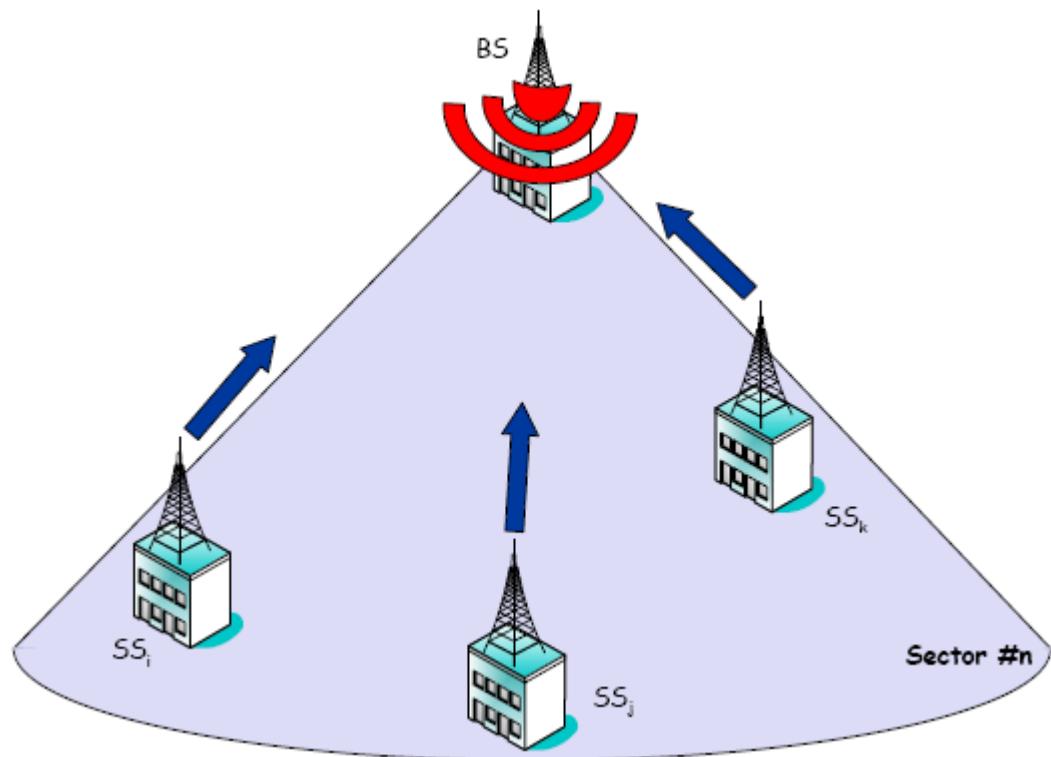
### 9.3 MAC Common Part Sublayer

The MAC CPS provides the core functionalities of:

- Air interface access: modes point to multipoint or mesh (optional);
- Connection establishment and maintenance;
- Bandwidth allocation.

**Quality of Service (QoS)** is applied to the transmission and scheduling of data over the PHY.

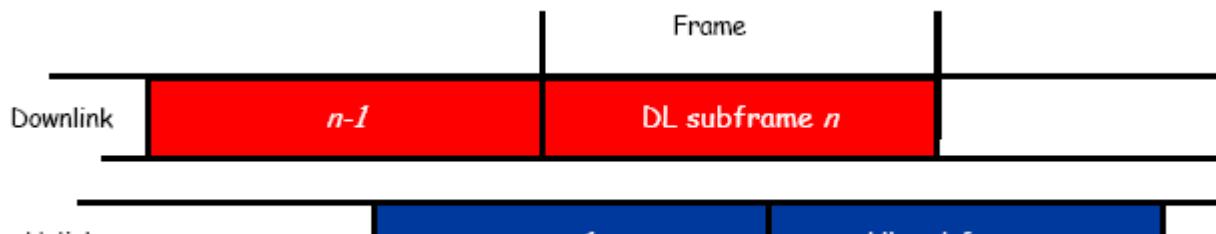
Point-to-multipoint (PMP):



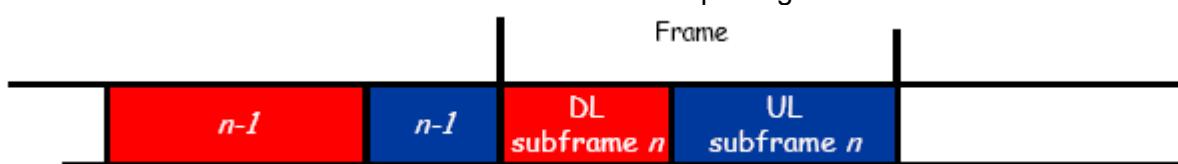
Communication from BS to SSs (Downlink) is broadcast; receiver is addressed indirectly through the connection ID. Communication from SS to BS (Uplink) has a polling-based access, centrally coordinated by the BS; contention access is also allowed, but at controlled time intervals.

#### 9.4 Frame-Based Duplexing

FDD – Frequency Division Duplexing:

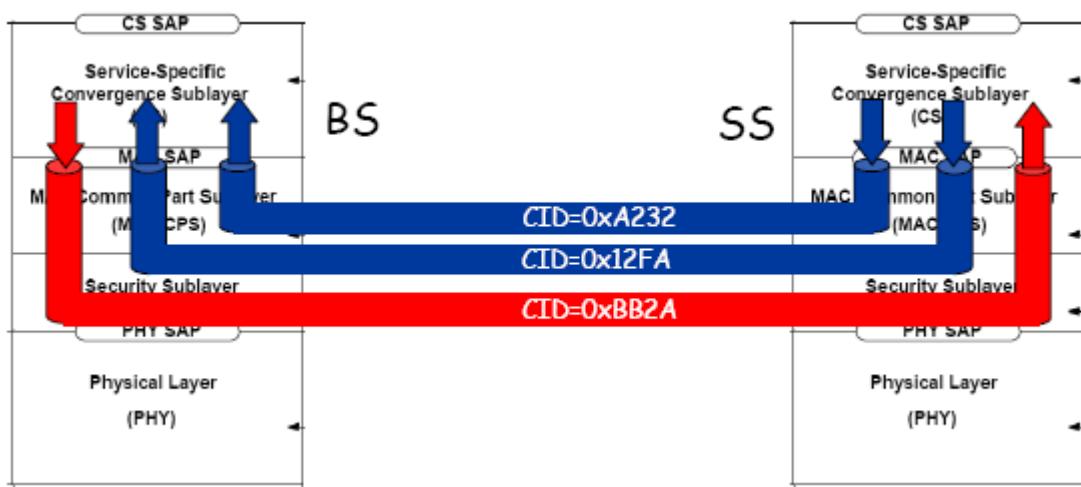


TDD – Time Division Duplexing:



#### 9.5 MAC connections

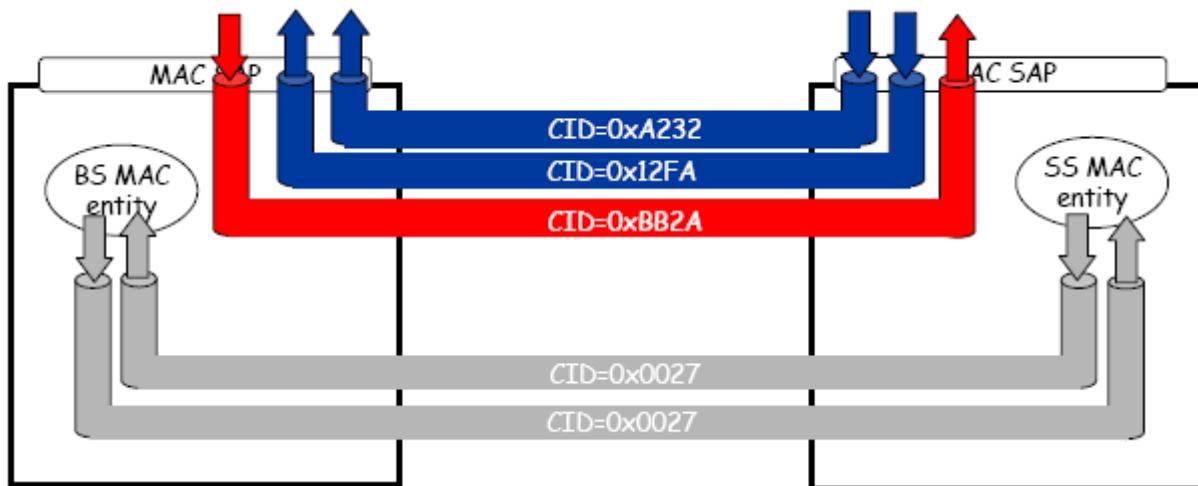
MAC is *connection-oriented*: it receives data from the various CSs, through the MAC SAP, classified to particular MAC connections. Connections are unidirectional (DL or UL). Each connection is characterized by a 16-bit *Connection Identifier (CID)*. The CID address space is common (i.e., shared) between UL and DL.



Connections are of two types:

- *Management*, to transport messages exchanged by BS and SS MAC entities in order to implement specific MAC functionalities (e.g. connection establishment).
- *Transport*, to carry MAC SDUs received from the CS (bearer service).

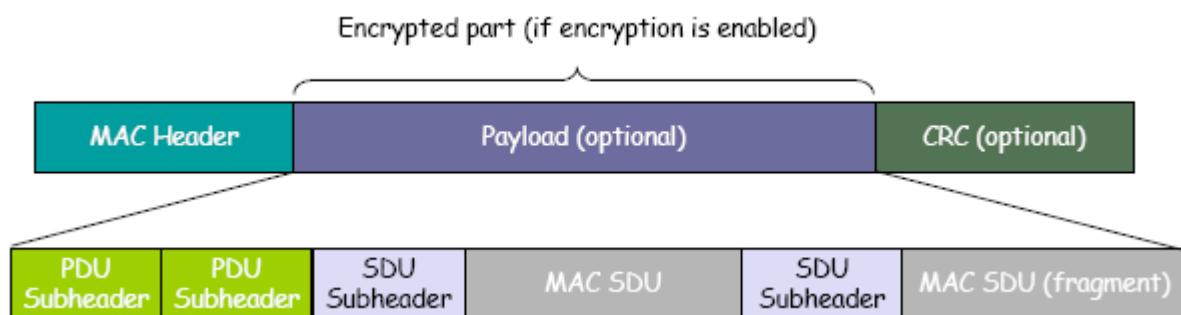
At SS initialization, two pairs of management connections (UL and DL) are established between the SS and the BS. The *basic connection* is used by the BS MAC and SS MAC to exchange short, time-urgent MAC management messages. The *primary management connection* is used by the BS MAC and SS MAC to exchange longer, more delay-tolerant MAC management messages.

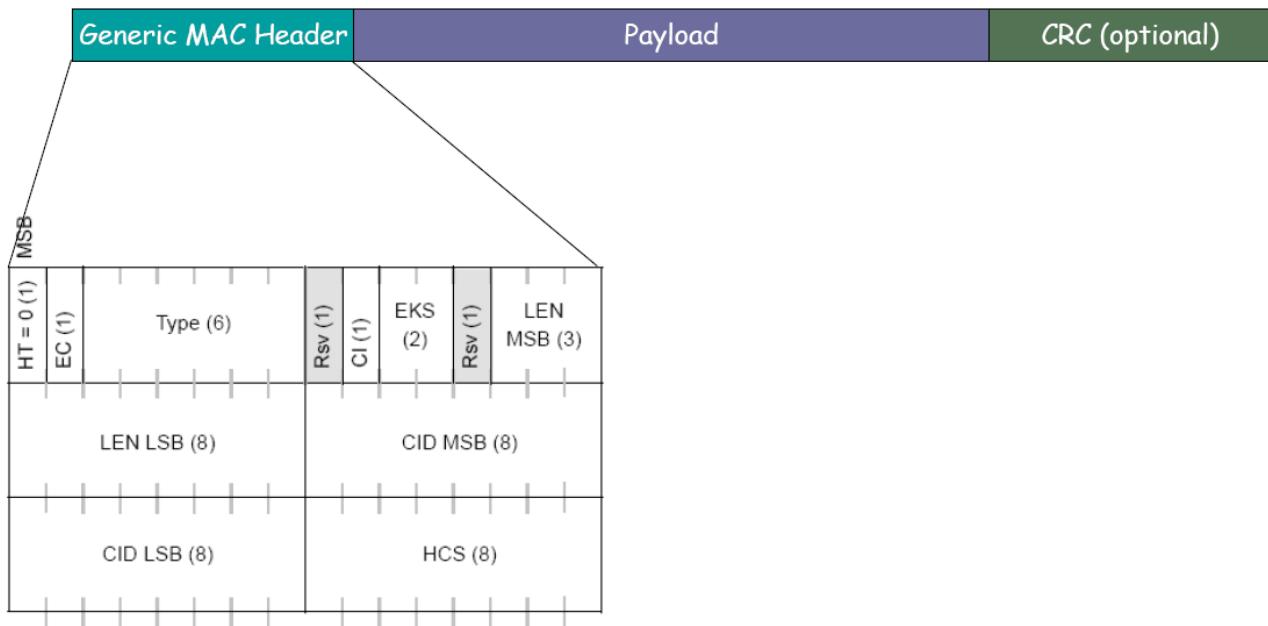


Basic CID is 0x0027, the same for both directions (only exception).

## 9.6 MAC Protocol Data Unit (MPDU) Format

Each PDU has a fixed length header (6 bytes). If payload is present, it may include additional subheaders: *PDU subheaders*, one per PDU, immediately after the MAC header; and *SDU subheaders*, one per SDU included in the payload (packing).



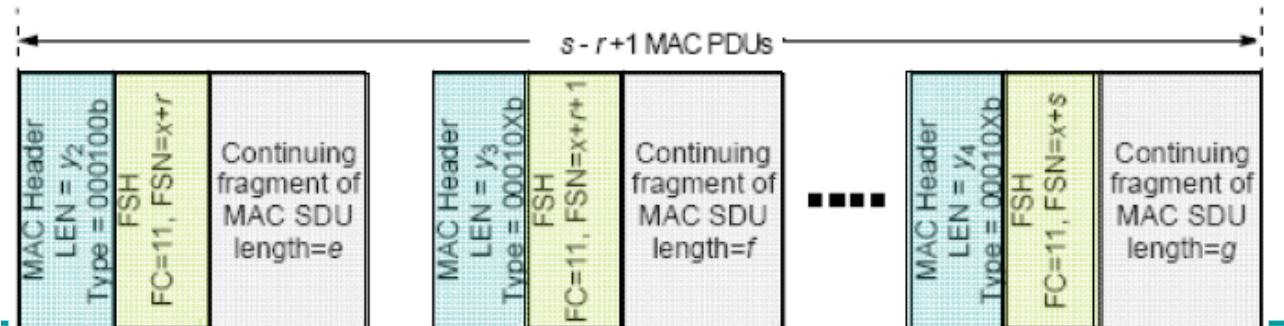


- HT (Header Type) = 0
- EC - Encryption Control
- EKS – Encryption Key sequence
- HCS – Header Check Sequence
- LEN – PDU length in bytes
- CI – CRC indicator
- CID – Connection Identifier
- Type – determines the specific format of this PDU. Each bit is a flag indicating the presence of a corresponding subheader.

## 9.7 Fragmentation

*Fragmentation* is the process by which a MAC SDU is divided into multiple MAC PDUs. This process is undertaken to allow efficient use of available bandwidth relative to the QoS requirements of a connection's service flow. If an MPDU transports an MSDU fragment, then it must include a *Fragmentation subheader*:

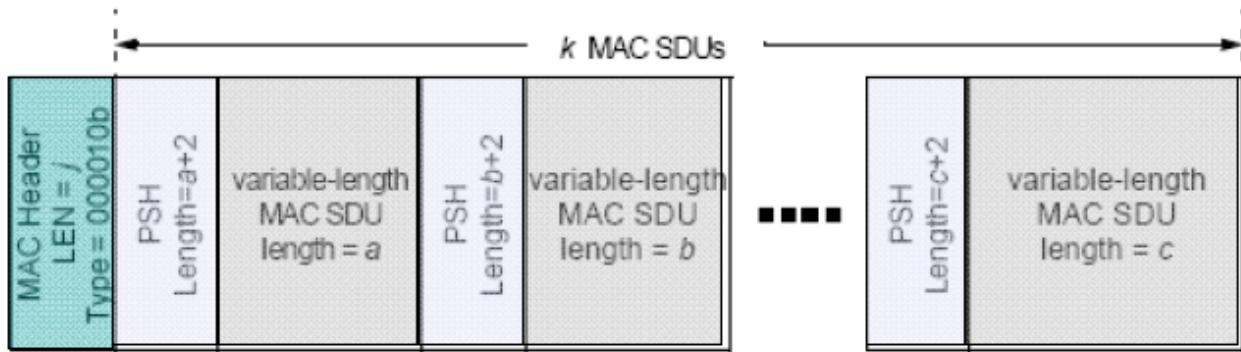
- FC – fragmentation state (first fragment, continuing fragment, last fragment, no fragment);
- Sequence number (optional).



## 9.8 Packing

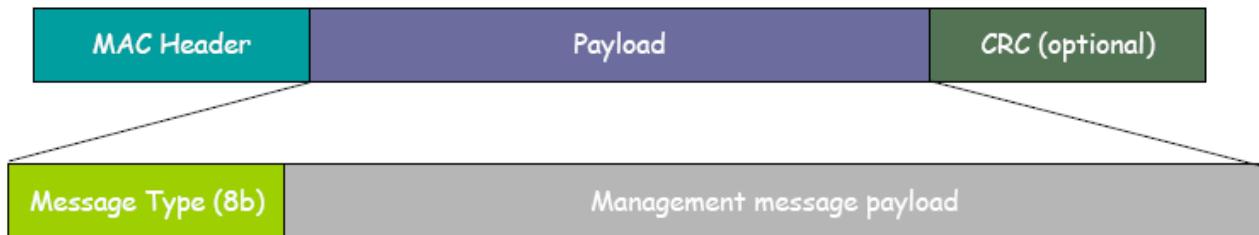
*Packing* is the process by which multiple MAC SDUs are packed into one MAC PDU. If an MPDU transports multiple MSDUs, each MSDU must be preceded by a *Packing subheader*:

- FC – fragmentation state (first fragment, continuing fragment, last fragment, no fragment);
- Sequence number (optional);
- Length.



## 9.9 MAC Management Messages

MAC management messages are exchanged as payload over management connections.



They are used for channel configuration and link management, bandwidth allocation, connection setup and maintenance, network startup, etc.

## 9.10 ARQ

ARQ mechanism is available at the MAC layer. Implementation is optional (and not allowed with WirelessMAN SC). ARQ is enabled on a per-connection basis. For ARQ-enabled connections, each MAC SDU is logically partitioned into blocks whose length is specified by the connection parameter ARQ\_BLOCK\_SIZE. Mechanisms:

- Selective ACK with maps
- Cumulative ACK
- Cumulative with Selective ACK
- Cumulative ACK with Block Sequence Ack

The ARQ feedback information can be sent as a standalone MAC management message on the appropriate basic management connection, or piggybacked on an existing connection (not necessarily the one for which feedback is provided).

## 9.11 Quality of Service (QoS)

MAC is QoS-oriented. Each transport connection is implicitly associated to a specific *scheduling service*. Scheduling services represent the data handling mechanisms supported by the MAC scheduler for data transport on a connection. A scheduling service is determined by a set of QoS parameters that quantify aspects of its behavior. The standard keeps flexibility at maximum, by not specifying any set of Classes of Service! Available QoS parameters are:

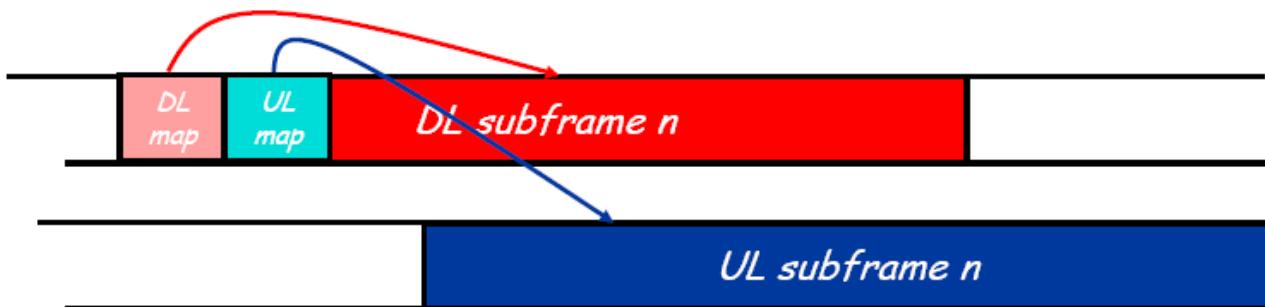
- Minimum Reserved Traffic Rate
- Maximum Sustained Traffic Rate
- Maximum Latency
- Tolerated Jitter
- Traffic Priority
- ...

Well-known scheduling services can be implemented by specifying a specific (sub)set of QoS parameters. For each transport connection, QoS parameters values for the subset specified by the associated scheduling service are negotiated at connection setup. Example of QoS parameters providing scheduling service to support real-time constant bit-rate data streams are Tolerated jitter, SDU size, Maximum Sustained Traffic Rate, Maximum Latency, etc... Example of QoS parameters providing scheduling service to support delay-tolerant variable-rate data streams are Minimum Reserved Traffic Rate, Maximum Sustained Traffic Rate, Traffic Priority, etc... QoS provisioning is

supported by:

- centralized bandwidth allocation,
- packet classification and scheduling and
- specific bandwidth request/grant mechanisms for uplink connections.

Centralized Bandwidth Allocation:

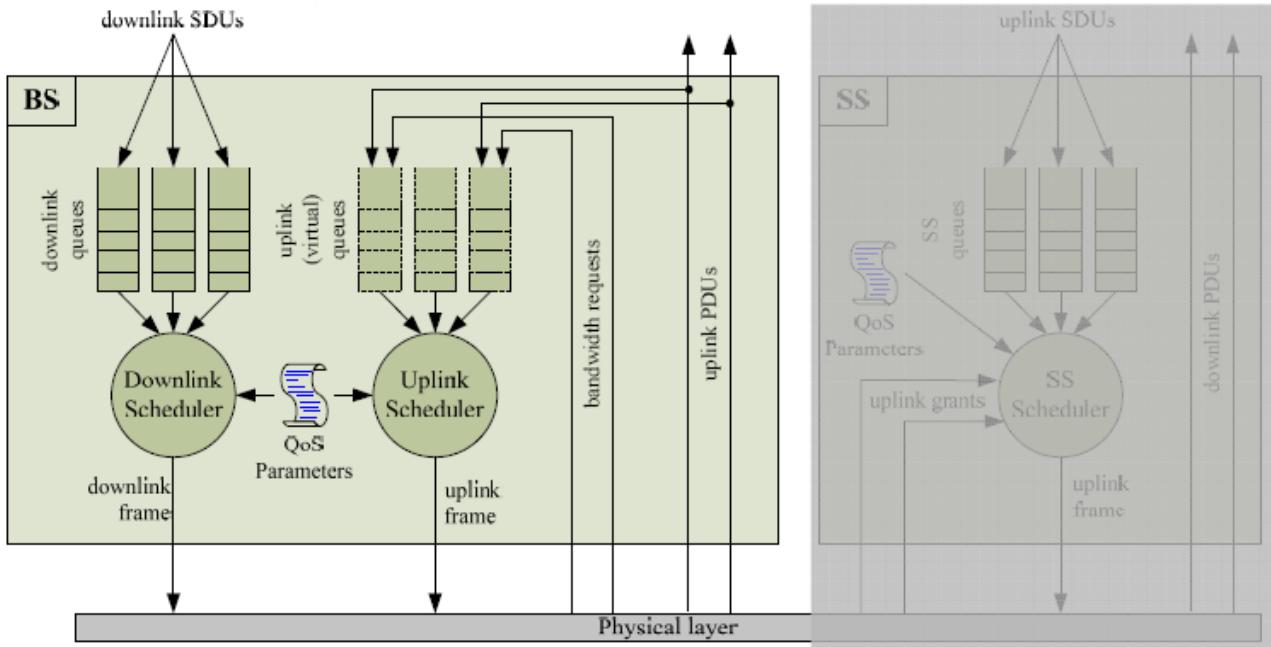


At the beginning of each downlink (sub-)frame, the BS transmits two dedicated MAC management messages:

- *DL-map*: provides information about data that is going to be transmitted in the current DL subframe
- *UL-map*: provides detailed information about which SSs are allowed to transmit (contention-free), when, and for how long, in the corresponding UL subframe. Additionally, may set dedicated portions of the subframe for contention-based access.

Map message formats are (strongly) PHY-specific. Packet classification takes place at the Convergence Sublayer. It's a mapping from higher layers packet fields (e.g destination IP address or TOS field) to a CID, i.e. a corresponding scheduling service. Scheduling takes place at the BS MAC layer. Downlink scheduling selects which packets to transmit in the next downlink subframe (all packet queues are at the BS). Uplink scheduling selects which SSs are provided a grant to transmit in the next uplink subframe (packet queues are distributed among SSs and queue state is implicitly known or explicitly obtained through bandwidth requests).

MAC Scheduling (algorithms are not standardized!):



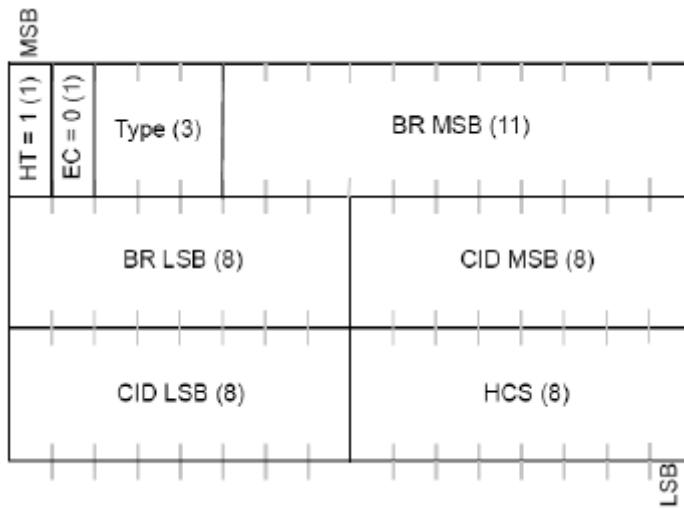
(Image: C. Cicconetti, C. Eklund, L. Lenzini, and E. Mingozzi, “Quality of Service Support in IEEE 802.16 Networks”, IEEE Network, Vol. 20, No. 2, pp. 50-55, March/April 2006)

## 9.12 Bandwidth Request/Grant Mechanisms

SSs are allowed to ask for bandwidth on a connection basis. Bandwidth requests are specified in terms of bytes!, not including any PHY overhead. Several mechanisms are implemented to accomplish this task.

1. *Implicit bandwidth requests*: No request messages during the connection lifetime. Bandwidth needs (typically, a fixed amount of bytes on a periodic basis) are negotiated at connection setup.

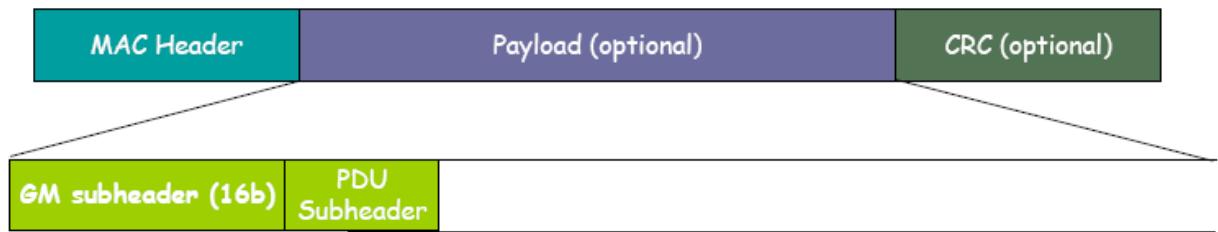
2. Bandwidth Request (BR) messages:



- HT (Header Type) = 1;
- BR – Bandwidth Request (up to 512 KB within a single message);
- Type Incremental: adds to previous requests, Type Aggregate: resets previous requests.

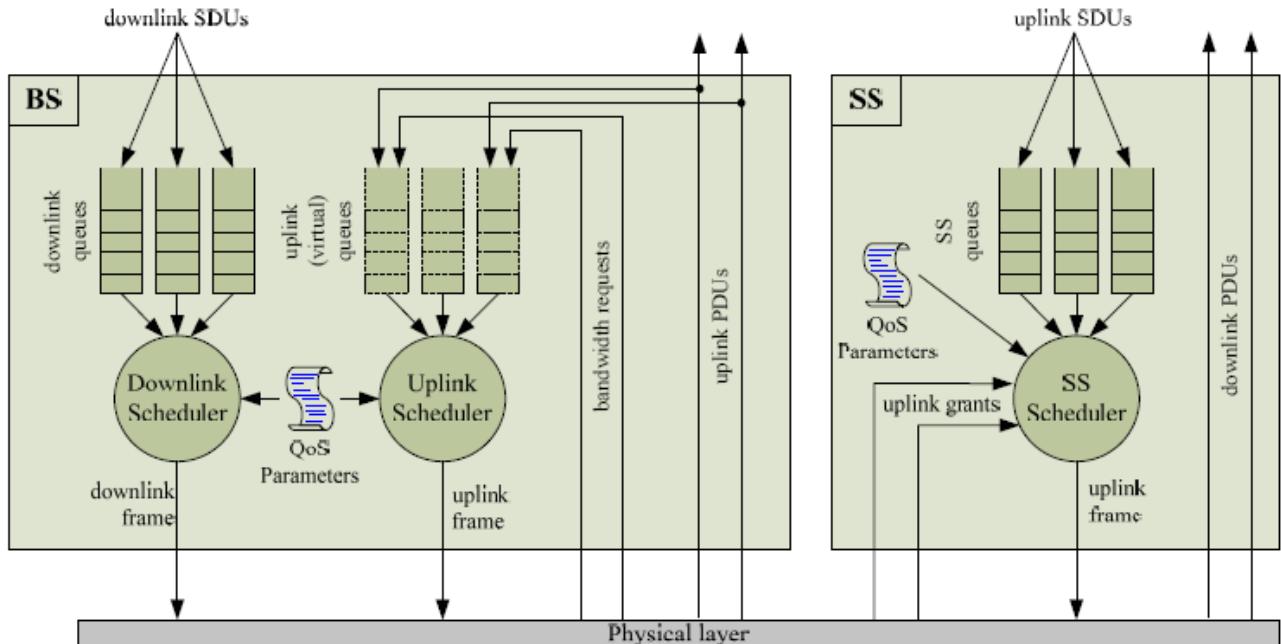
BR messages are only allowed on uplink connections. Can be transmitted *Contention-basis* (contention request opportunities) or *Contention-free* (unicast request opportunities).

3. *Piggybacked requests*: The bandwidth request is transported in a dedicated Grant Management (GM) subheader. The request is incremental, for the CID of the carrying MPDU, up to 64 KB.



Bandwidth grants are addressed to SSs (identified by the respective Basic CID), not to individual CIDs. When an SS receives a bandwidth grant, it is nondeterministic which request is being honored. It follows that a scheduling functionality is also needed in the SS MAC.

MAC Scheduling (algorithms are not standardized!):



(Image: C. Cicconetti, C. Eklund, L. Lenzini, and E. Mingozzi, “Quality of Service Support in IEEE 802.16 Networks”, IEEE Network, Vol. 20, No. 2, pp. 50-55, March/April 2006)

## 9.13 UL Request/Grant Scheduling Types

Five request/grant UL scheduling types are specified:

- Unsolicited Grant Service (UGS)
- Real-Time Polling Service (rtPS)
- Extended rtPS (ertPS)
- Non-Real-Time Polling Service (nrtPS)
- Best Effort Service

For each uplink connection, the corresponding scheduling type is negotiated at connection setup.

*Unsolicited Grant Service (UGS)* is for real-time service flows with fixed size packets generated at periodic intervals (e.g. T1, VoIP without silence suppression). QoS parameters:

- Maximum Sustained Traffic Rate (mandatory)
- Maximum Latency (mandatory)
- Tolerated Jitter (mandatory)
- Unsolicited Grant Interval (mandatory)
- SDU size (optional)

The BS provides bandwidth grants to the SS at periodic intervals. The size of these grants shall be sufficient to hold the fixed-length data associated with the service flow, based upon the Maximum Sustained Traffic Rate. The SS is prohibited from using any contention request opportunities for this connection.

*Real-Time Polling Service (rtPS)* is for real-time service flows with variable sized packets generated at periodic intervals (e.g MPEG). QoS parameters:

- Minimum Reserved Traffic Rate (mandatory)
- Maximum Sustained Traffic Rate (mandatory)
- Maximum Latency (mandatory)
- Unsolicited Polling Interval parameter (optional)

The BS provides bandwidth grants at periodic intervals to poll for transmission of bandwidth requests (unicast request opportunities). The SS is prohibited from using any contention request opportunities for this connection.

*Extended Real-Time Polling Service (ertPS)* combines the efficiency of both UGS and rtPS, for, e.g., real-time service flows, fixed sized packets on-off generated at periodic intervals (VoIP with silence suppression). The BS provides bandwidth grants to the SS at periodic intervals. The size of these grants, by default, is set, as with UGS, based upon the Maximum Sustained Traffic Rate. The SS (MS) may request changing the size by means of, e.g., the GM subheader. If the size is set to zero, the BS provides unicast request opportunities.

*Non-Real-Time Polling Service (nrtPS)* is for delay-tolerant data streams with variable sized packets and a required minimum data rate (e.g. FTP). QoS parameters:

- Minimum Reserved Traffic Rate (mandatory)
- Maximum Sustained Traffic Rate (mandatory)
- Traffic Priority (mandatory)

The BS provides bandwidth grants at regular intervals (not necessarily periodic), to poll for transmission of bandwidth requests, in order to assure request opportunities even during network congestion. The SS can also use contention requests and piggybacking.

*Best Effort (BE) Service* is for data streams for which no minimum service level is required. The BS does not guarantee to provide any unicast poll. The SS can use any available means to transmit bandwidth requests (contention, unicast polls, piggybacking).

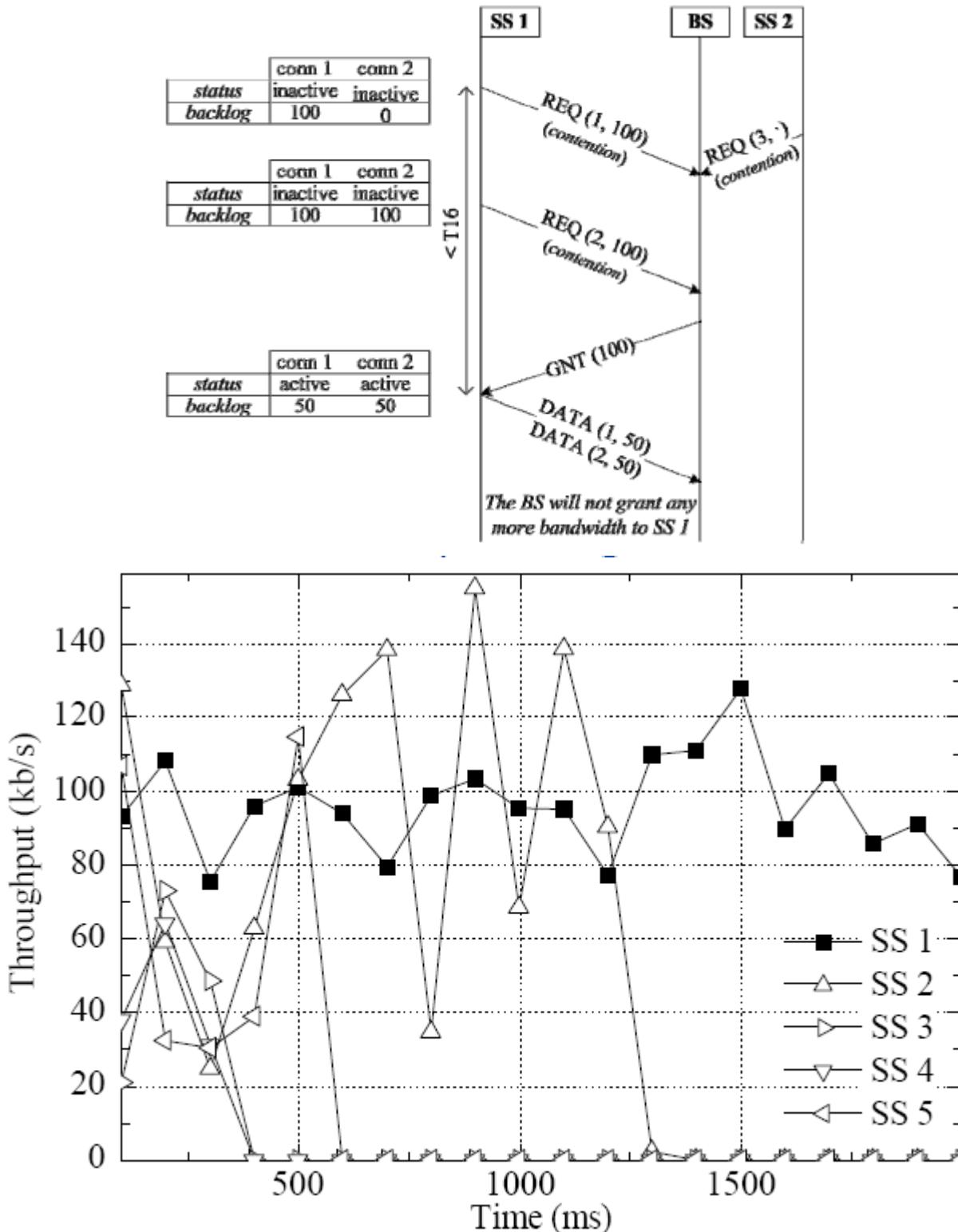
Summary of Scheduling Types:

Scheduling type	Piggyback request	Bandwidth stealing	Polling
UGS	Not allowed	Not allowed	PM bit is used to request a unicast poll for bandwidth needs of non-UGS connections
rtPS	Allowed	Allowed	Scheduling only allows unicast polling
nrtPS	Allowed	Allowed	Scheduling may restrict a service flow to unicast polling via the transmission/request policy
BE	Allowed	Allowed	All forms of polling allowed

## 9.14 Deadlock of the Request/Grant Mechanism

All pending bandwidth requests are implicitly acknowledged by the reception of the next

bandwidth grant, i.e., there is no explicit acknowledgement. But BR messages can get lost.



(Image: C. Cicconetti, L. Lenzini, and E. Mingozzi, "Avoiding Service Deadlock in IEEE 802.16 MAC through Bandwidth Request Reiteration", work in progress)

## 9.15PHY Layer

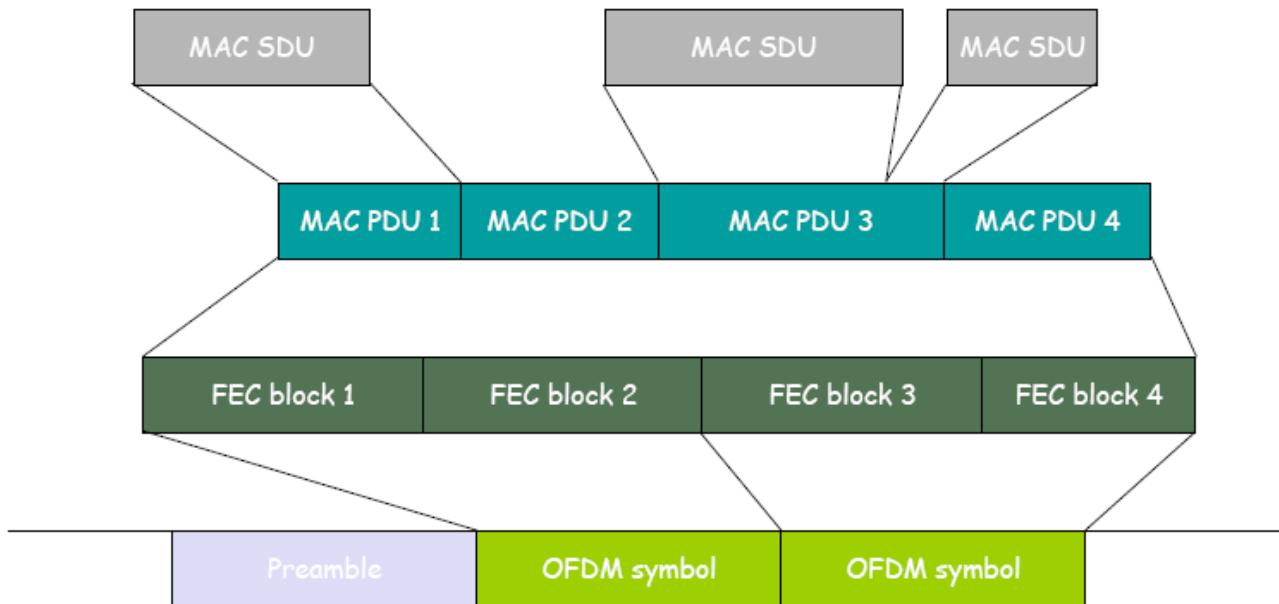
There are four PHY layer specifications:

1. The *WirelessMAN-SC PHY* specification is based on single-carrier technology and targeted for LOS operation in the 10–66 GHz frequency band.
2. The *WirelessMAN-SCa PHY* is based on single-carrier technology and designed for NLOS operation in frequency bands below 11 GHz.
3. The *WirelessMAN-OFDM PHY* is based on OFDM modulation and designed for NLOS operation in the frequency bands below 11 GHz.
4. The *WirelessMAN-OFDMA PHY*, based on OFDM modulation, is designed for NLOS operation in the frequency bands below 11 GHz.

802.16 WirelessMAN-OFDM has:

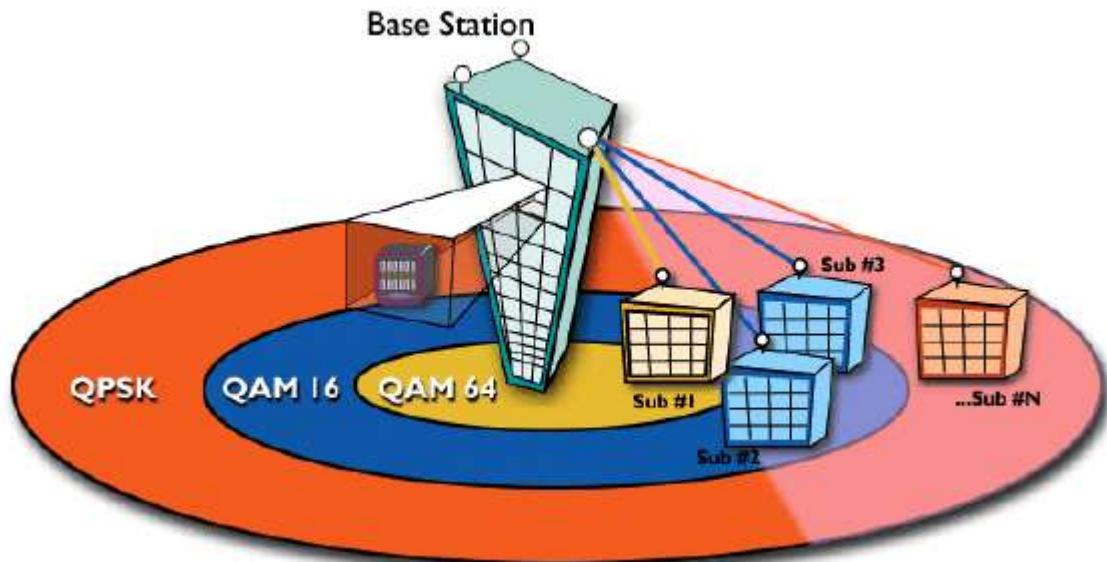
- Burst transmissions;
- Adaptive burst profiles in uplink and downlink;
- Both (alternative) duplexing schemes supported: Time-Division Duplexing (TDD) and Frequency-Division Duplexing (FDD);
- Support for both full and half duplex stations.

Multiple MPDUs are concatenated into PHY bursts (PHY burst transmission).

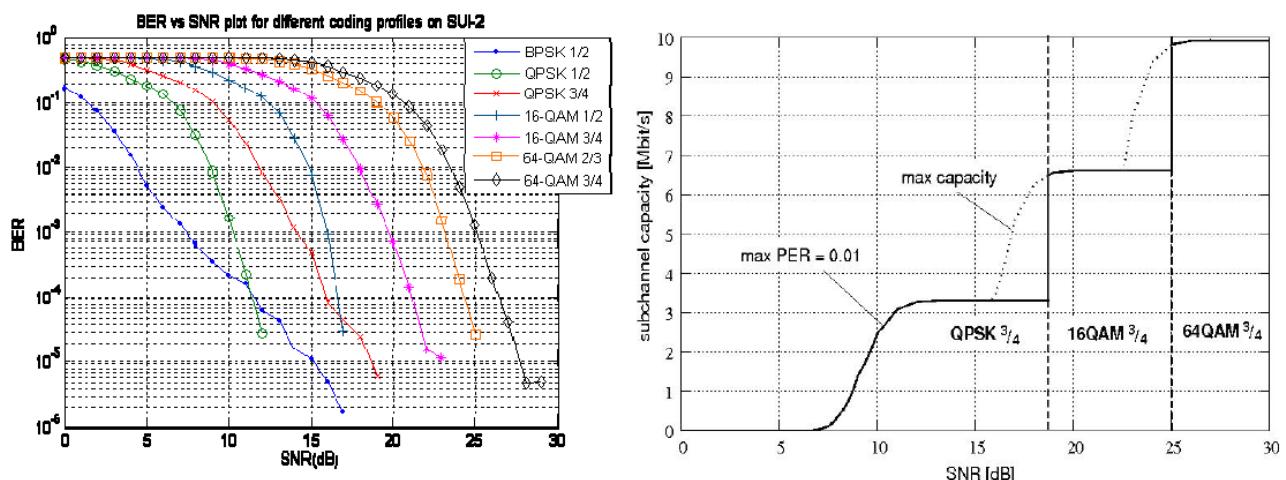


## 9.16 Adaptive PHY

Downlink and Uplink Burst Profiles have different available modulation and coding schemes identified by an Interval Usage Code (DIUC and UIUC).

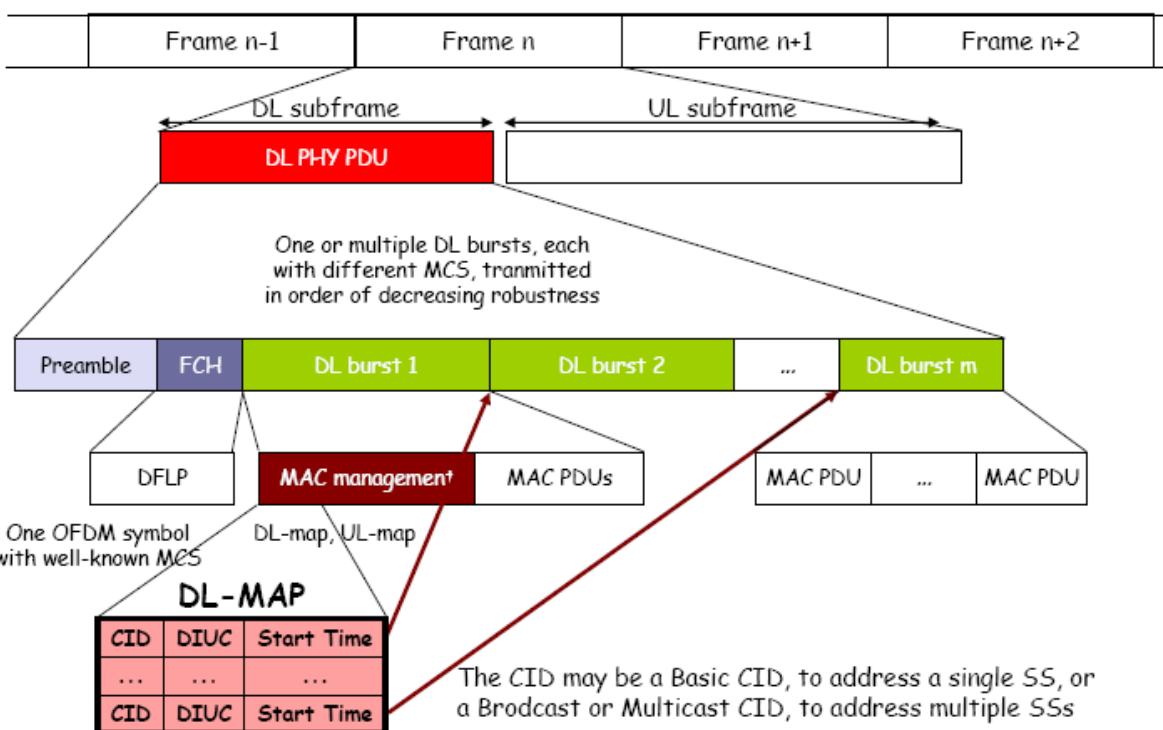


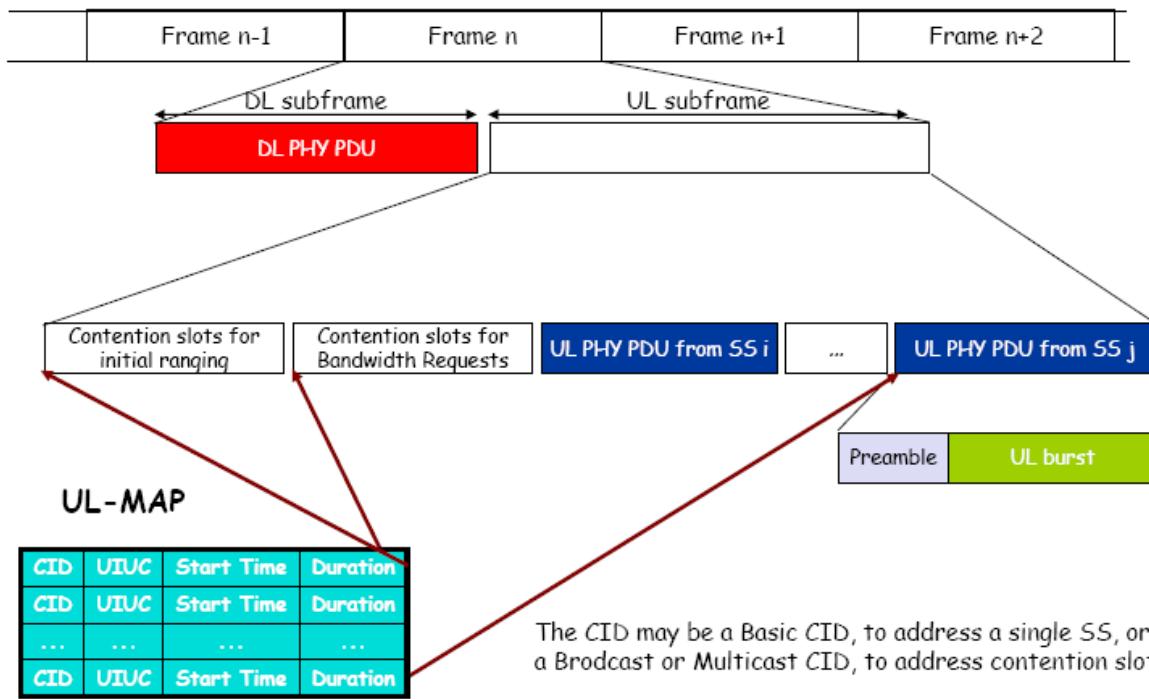
Burst profiles are dynamically selected for downlink and uplink burst transmissions depending on (estimated) link quality per SS (adaptive modulation and coding).



## 9.17 TDD and FDD Frame Structures

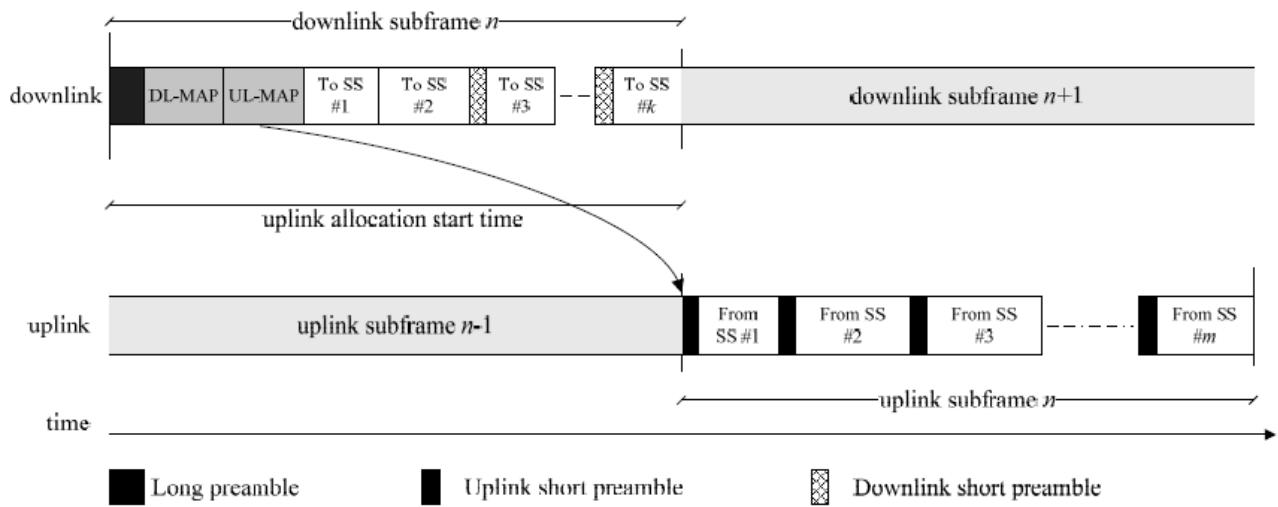
TDD Frame Structure:



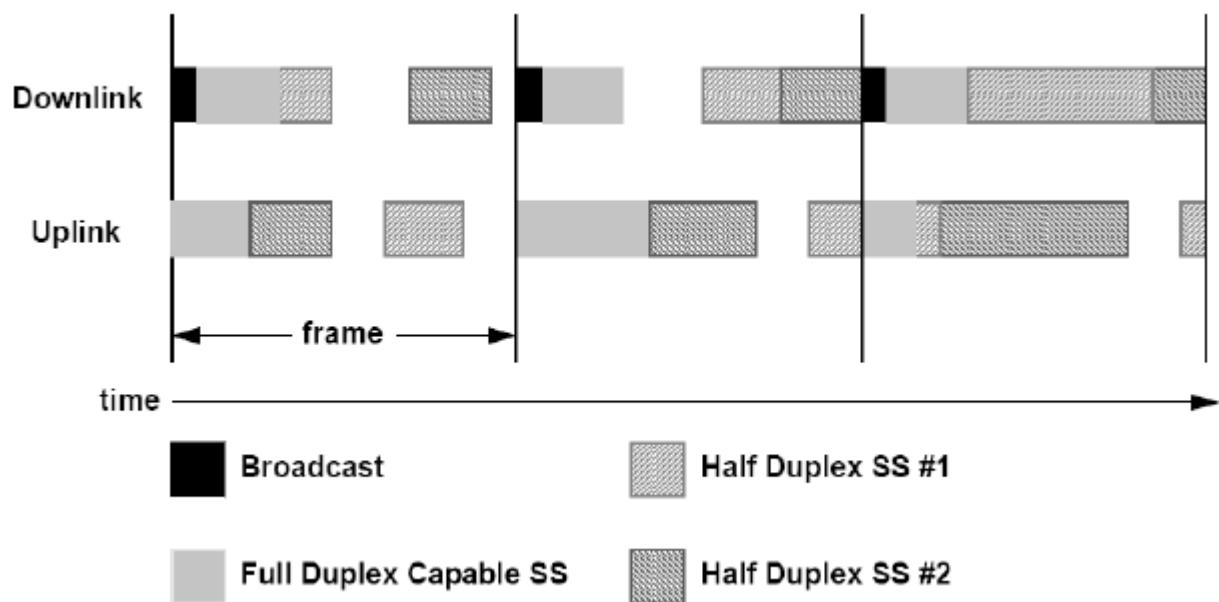


The *Initial Ranging* opportunities are to determine network delay and to request power or profile changes. Collisions may occur in this interval. In Request Contention opportunities SSs request bandwidth. Collisions may occur in this interval as well. SSs transmit data bursts in the intervals granted by the BS (Bandwidth grants per SS). Note that bandwidth requests are in bytes but bandwidth grants are in OFDM symbols (i.e., time). Do you see any issue in provisioning QoS?

FDD Frame Structure:



Support for half-duplex stations in FDD:

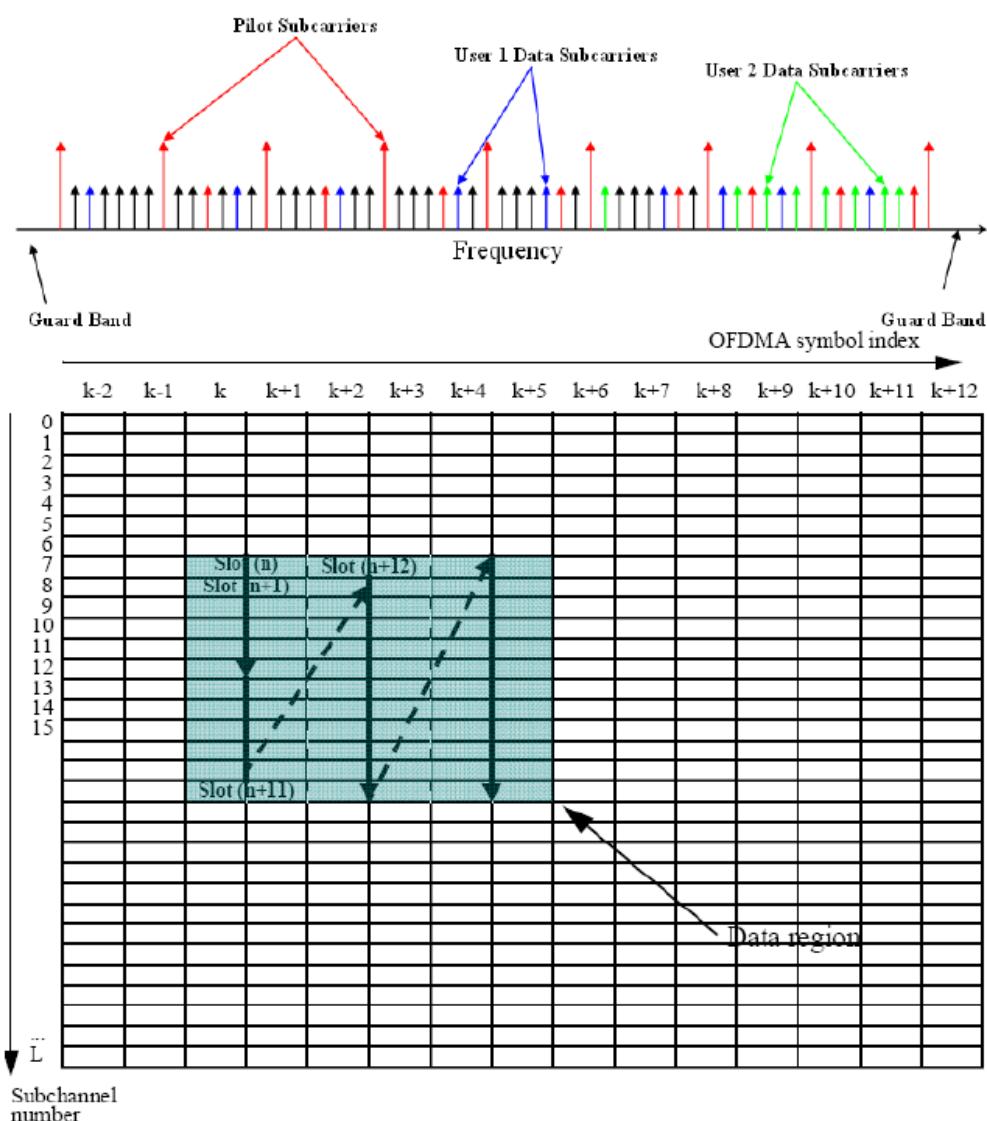


### 9.18 802.16 WirelessMAN-OFDMA

Characteristics:

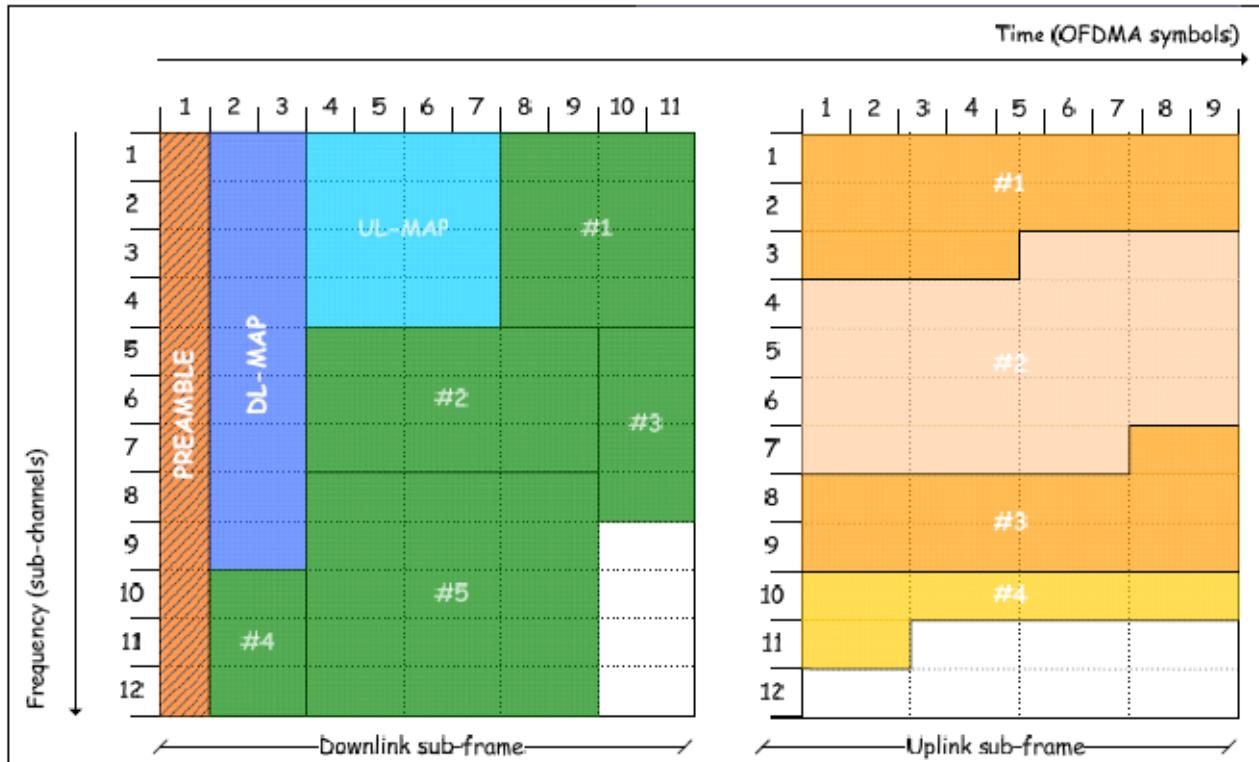
- Mobility management functions (Power management, Handoff);
- Enhanced data protection mechanisms (Hybrid ARQ (H-ARQ));
- Frequency selective scheduling, to exploit multi-user diversity.

OFDMA Modulation:



Orthogonal Frequency Division Multiple Access (OFDMA) is a multi-user version of OFDM. Multiple access is achieved in OFDMA by assigning subsets of subcarriers (named subchannels) to individual users as shown in the figure below. Multi-user diversity may be exploited by dynamically assigning different subchannels to different MSs.

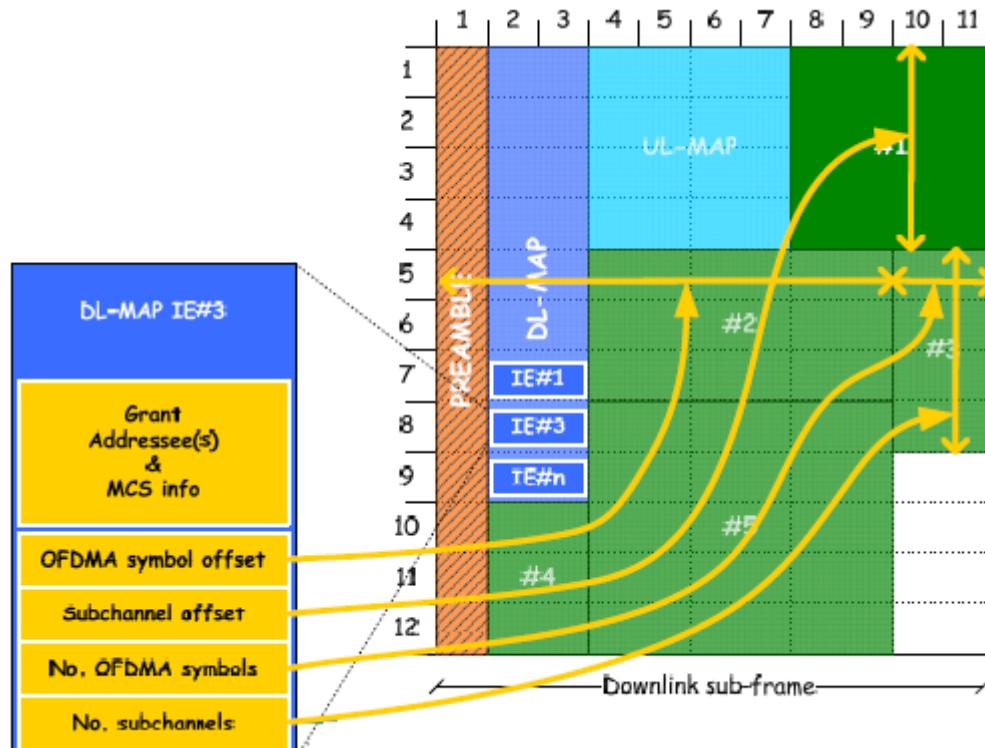
Frame Structure (Increased complexity at the MAC layer!):



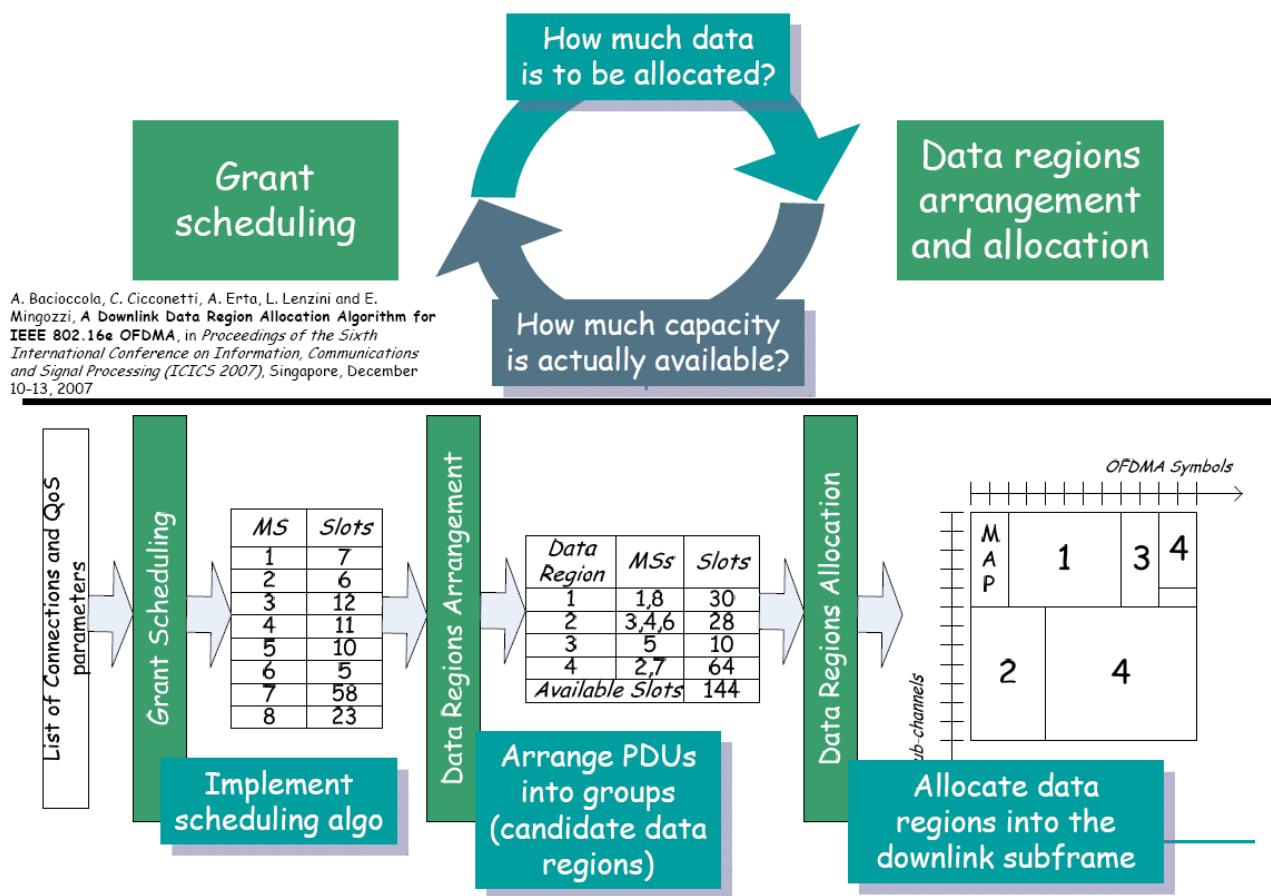
Scheduling constraints are:

- QoS requirements of connections (common);
- In-band signalling (common): maps are transmitted at the beginning of the downlink subframe;
- Data regions allocation in the downlink subframe (OFDMA specific).

Resource allocation is a key issue. Data regions arrangement and allocation are done in a way that size, shape and position must be set so that they do not overlap in time and frequency.

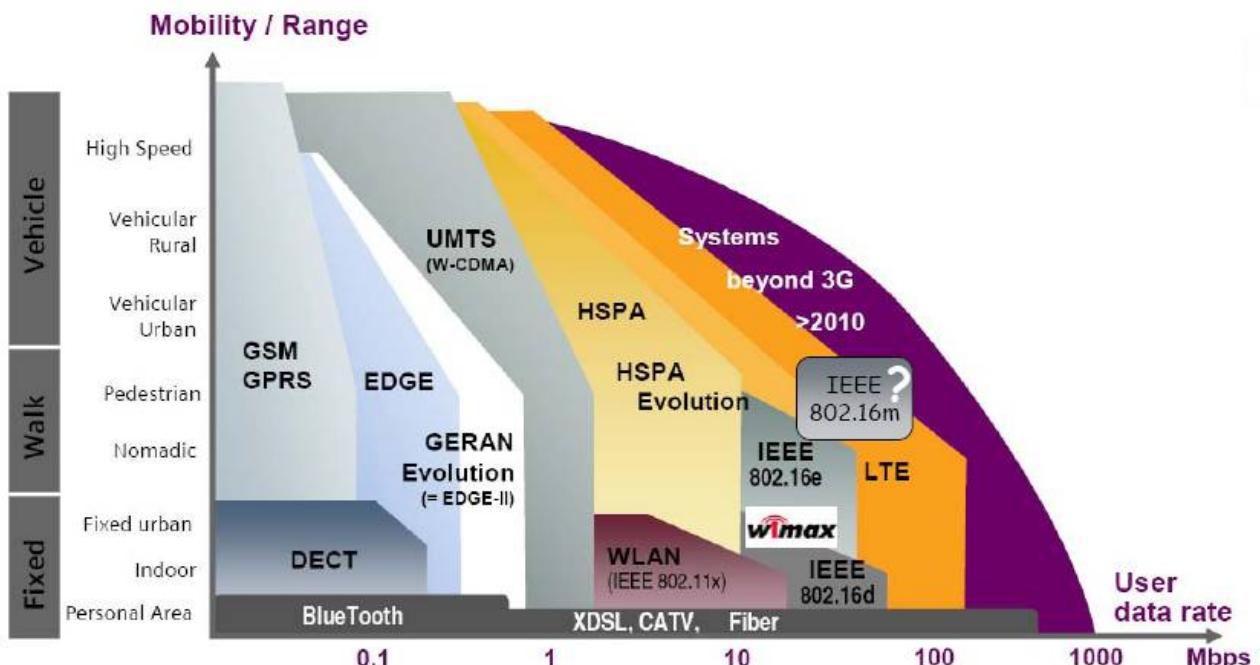


It's a two-dimensional bin packing optimization problem. It's NP-HARD! How to solve for interdependence?



## 9.19 Conclusions and Outlook

WiMAX (IEEE802.16d/e) covers fixed wireless and nomadic access, the e-Standard extends towards (limited) mobility. HSPA Evolution and LTE target at high data rates combined with high subscriber mobility.



Source: Nokia Siemens Networks 2007

## 9.20 References

- <http://www.wirelessman.org>
- <http://www.wimaxforum.org>
- <http://standards.ieee.org/getieee802/>
- *IEEE Std 802.16™-2004* (Revision of IEEE Std 802.16-2001) - IEEE Standard for Local

and metropolitan area networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems.

- *IEEE Std 802.16e™-2005* and *IEEE Std 802.16™-2004/Cor1-2005* (Amendment and Corrigendum to IEEE Std 802.16-2004) - IEEE Standard for Local and metropolitan area networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems. Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1.

# 10 Long Term Evolution (LTE)

## 10.1 References

- LTE application note (Rhode & Swartz)  
“[http://www2.rohde-schwarz.com/file/1MA111\\_2E.pdf](http://www2.rohde-schwarz.com/file/1MA111_2E.pdf)”
- 3GPP website “[www.3gpp.org](http://www.3gpp.org)”
- Computer Networking Group – University Of Pisa “<http://cng1.iet.unipi.it>”
- My Email “matteo.andreozzi@iet.unipi.it”

## 10.2 UMTS Introduction

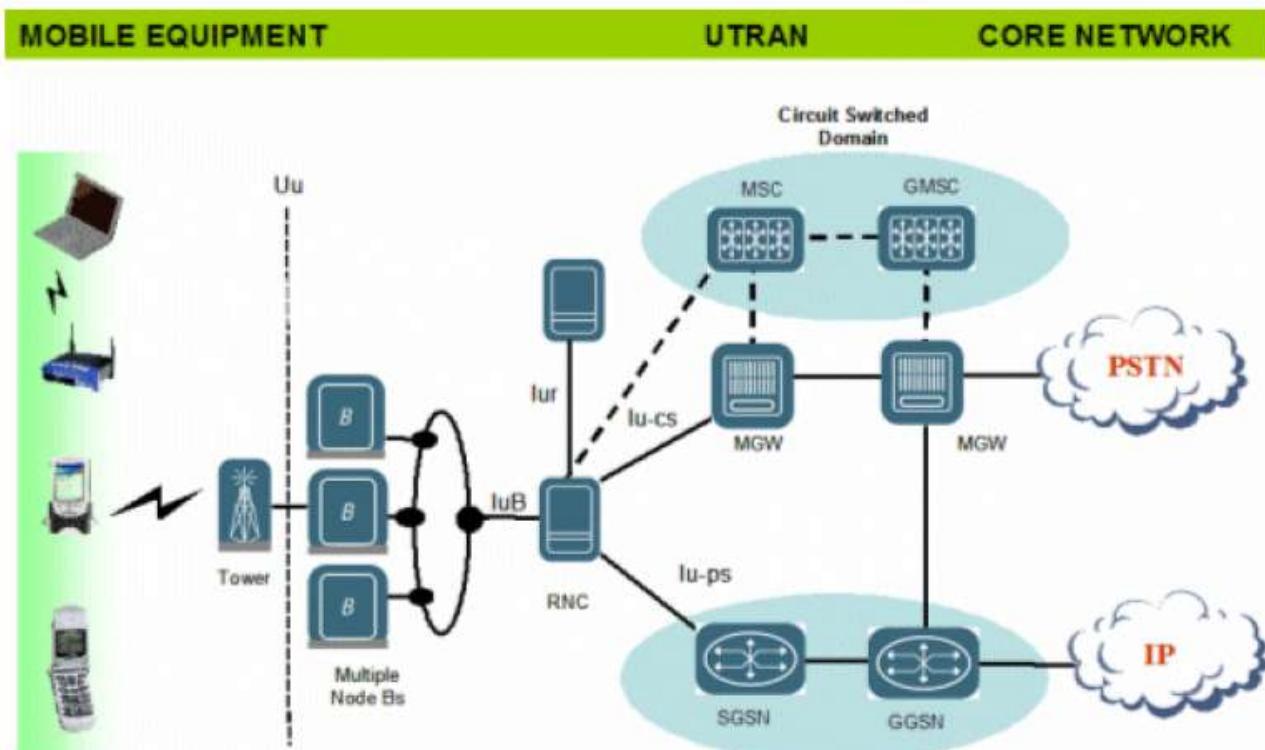
UMTS offers telephony services (like speech or SMS) and bearer services, which provide the capability for information transfer between access points (packet data transfer). Offered data rate targets are:

- 144 kbit/s satellite and rural outdoor,
- 384 kbit/s urban outdoor,
- 2048 kbit/s indoor and low range outdoor.

UMTS has different QoS parameters for maximum transfer delay, delay variation and bit error rate. UMTS has different QoS classes for four types of traffic:

- Conversational class (voice, video telephony, video gaming),
- Streaming class (multimedia, video on demand, webcast),
- Interactive class (web browsing, network gaming, database access),
- Background class (email, SMS, downloading).

UMTS Architecture:



- **MSC:** The *Mobile Switching Center (MSC)* switch, including the Visitor Location Register (VLR), is a switch that serves the Mobile Equipment (ME) in its current location for Circuit Switched (CS) services.
- **GMSC:** The *Gateway MSC (GMSC)* switch serves the UMTS network at the point where it is connected to the external CS network.
- **MGW:** The MSC and GMSC handle control functionality, but user data goes through the *Media Gateway (MGW)*, which performs the actual switching for user data and network interworking processing.
- **SGSN:** The *Serving GPRS Support Node (SGSN)* covers functions similar to the MSC for

packet data, including VLR type functionality.

- **GGSN:** The *Gateway GPRS Support Node (GGSN)* connects the Packet-Switched (PS) core network to other networks such as the Internet.
- **Node B:** A *3G Base station (Node B)* handles radio channels, including the multiplexing/demultiplexing of user voice and data information.
- **RNC:** The *Radio Network Controller (RNC)* is responsible for controlling and managing the multiple base stations (Node Bs) including the utilization of radio network services.

A UMTS network consist of three interacting domains; *Core Network (CN)*, *UMTS Terrestrial Radio Access Network (UTRAN)* and *User Equipment (UE)*. The UMTS Terrestrial Radio Access Network (UTRAN) includes the Radio Network Controller (RNC), the 3G Base stations (Node Bs) and the air interface (Tower) to the UE. The main function of the core network is to provide switching, routing and transit for user traffic. Core network also contains the databases and network management functions. The basic Core Network architecture for UMTS is based on GSM network with GPRS. Wide band CDMA technology was selected to for UTRAN air interface. UMTS WCDMA is a Direct Sequence CDMA system where user data is multiplied with quasi-random bits derived from WCDMA Spreading codes. In UMTS, in addition to channelization, Codes are used for synchronization and scrambling. WCDMA has two basic modes of operation: Frequency Division Duplex (FDD) and Time Division Duplex (TDD).

### 10.3 LTE Introduction

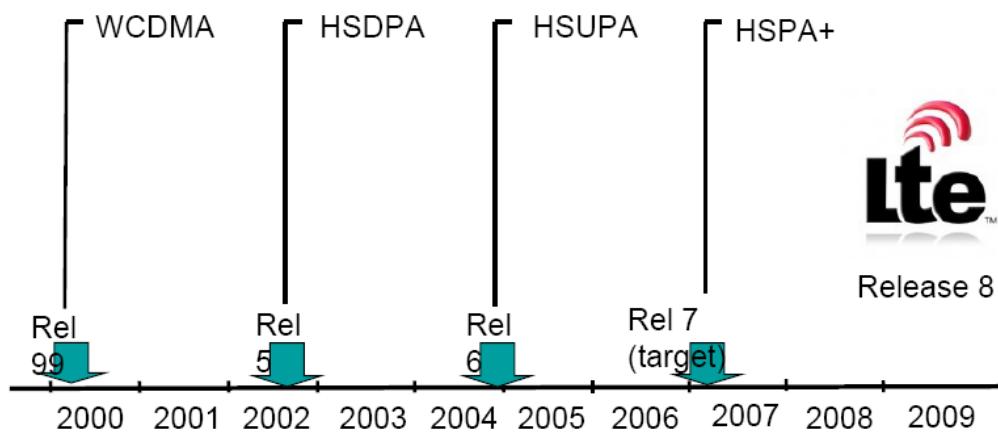


A GLOBAL INITIATIVE

3GPP stands for *3rd Generation Partnership Project*. It is a partnership of 6 regional SDOs. These SDOs take 3GPP specifications and transpose them to regional standards. ITU references the regional standards.



3GPP Release Timeline:



*High Speed Downlink Packet Access (HSDPA)* increases data rate and capacity for downlink packet data. *High Speed Uplink Packet Access (HSUPA)* will boost uplink performance in UMTS networks. UMTS networks worldwide are being upgraded to releases 5 and 6. The combination of HSDPA and HSUPA is often referred to as *HSPA*. *HSPA+* will bring significant enhancements in 3GPP release 7. Features of HSPA+ are:

- downlink MIMO (Multiple Input Multiple Output),
- higher order modulation for uplink and downlink,
- improvements of layer 2 protocols (MAC protocols),
- continuous packet connectivity.

LTE is focused on:

- enhancement of the Universal Terrestrial Radio Access (UTRA),
- optimization of the UTRAN architecture.

With HSPA (downlink and uplink), UTRA will remain highly competitive for several years. LTE project aims to ensure the continued competitiveness of the 3GPP technologies for the future. In order to ensure the competitiveness of UMTS for the next 10 years and beyond, concepts for *UMTS Long Term Evolution (LTE)* have been investigated. LTE/EUTRA will then form part of 3GPP release 8 core specifications. LTE plan is endorsed by 3GPP Project Co-ordination Group. LTE work plan was created in September 2006. Initial studies and work-plan creation was almost completed in June 2006. Completion is foreseen in 2008. LTE objectives are:

- Improvements in spectral efficiency
- Improvement in latency, capacity, throughput
- Simplification of the radio network
- Simplification of the core network
- Optimization for IP traffic and services
- Simplified support and handover to non-3GPP access technologies

## 10.4 Main Requirements

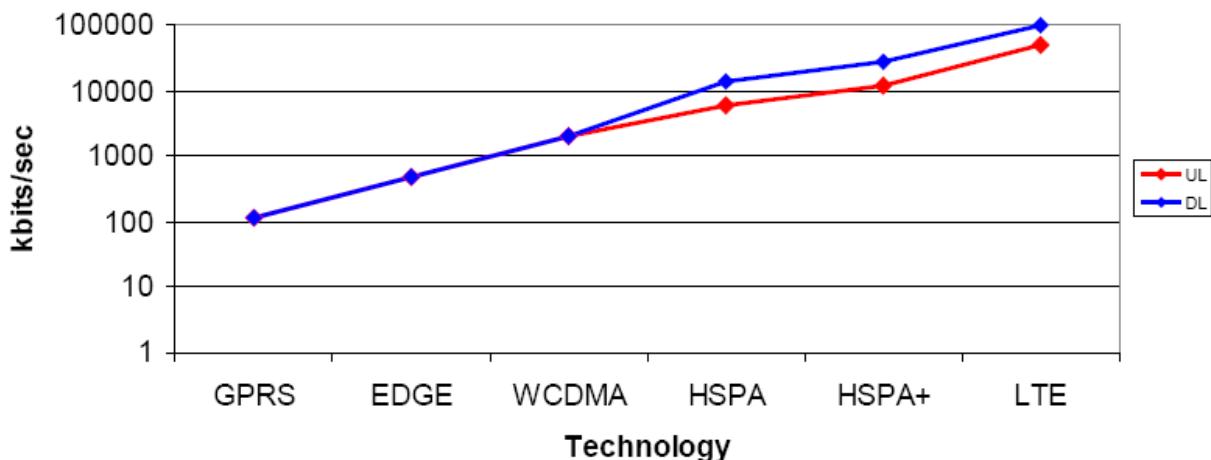
The LTE concept has the potential to fulfill both the system capacity and user throughput targets. Evaluated uplink peak data rate is a bit smaller than the requirements, however, it is expected that the peak data rate can be increased by some optimizations, e.g. higher TTI values and/or by reducing the amount of control signalling information. It was confirmed that the requirements of Control Plane and User Plane latency can be satisfied. Fulfilments without any issues are identified for requirements on deployment scenarios, spectrum flexibility, interworking, mobility, E-UTRAN architecture and RRM. The main requirements are:

- *Data Rate*: Peak data rates target 100 Mbps (downlink) and 50 Mbps (uplink) for 20 MHz spectrum allocation, assuming 2 receive antennas and 1 transmit antenna at the terminal.
- *Throughput*: Target for downlink average user throughput per MHz is 3-4 times better than release 6. Target for uplink average user throughput per MHz is 2-3 times better than release 6.
- *Latency*: The one-way transit time for a packet traveling from UE to EnB and vice versa shall be less than 5 ms.
- *Spectrum Efficiency*: Downlink target is 3-4 times better than release 6. Uplink target is 2-3 times better than release 6.
- *Bandwidth*: Bandwidths of 5, 10, 15, 20 MHz shall be supported. Also bandwidths smaller than 5 MHz shall be supported for more flexibility.

- *Interworking*: Interworking with existing UTRAN/GERAN systems and non-3GPP systems shall be ensured. Interruption time for handover between E-UTRAN and UTRAN/GERAN shall be less than 300 ms for real time services and less than 500 ms for non real time services.
- *Mobility*: The system should be optimized for low mobile speed (0-15km/h), but higher mobile speeds shall be supported as well including high speed train environment as special case.
- *Spectrum allocation*: Operation in paired (Frequency Division Duplex / FDD mode) and unpaired spectrum (Time Division Duplex / TDD mode) is possible.
- *Co-existence*: Co-existence in the same geographical area and co-location with GERAN/UTRAN shall be ensured. Also, coexistence between operators in adjacent bands as well as cross-border coexistence is a requirement.
- *Quality of Service*: End-to-end Quality of Service (QoS) shall be supported.

Evolution of 3GPP Radio Rates:

### Peak Network Data Rates



- Reduced cost per bit: improve spectrum efficiency (e.g. 2-4 x Rel6) and reduce cost of backhaul (transmission in UTRAN).
- Increased service provisioning – more services at lower cost with better user experience.
- Focus on delivery of services utilising “IP”.
- Reduce setup time and round trip time.
- Increase the support of QoS for the various types of services (e.g. Voice over IP).
- Increase “cell edge bit rate” whilst maintaining same site locations as deployed today.
- Increase peak bit rate (e.g. above 100Mbps DL and above 50Mbps UL).
- Flexibility of use of existing and new frequency bands.
- Allow to deploy in wider and smaller bandwidths than 5 MHz (e.g. ranging from 1.25 to 20MHz).
- Allow variable duplex technology within bands as well as between bands.
- Non-contiguous spectrum allocations to one UE should not be precluded.

Other requirements are:

- Need to consider UTRAN Evolution and UTRA Evolution at the same time aiming at simplifying the current architecture.
- Shall provide open interfaces to support Multi-vendor deployments.
- Consider robustness – no single point of failure.
- Support multi Radio Access Technologies (RAT) with resources controlled from the network.
- Support of seamless mobility to legacy systems as well as to other emerging systems including inter RAT Handovers and Service based RAT Selection.
- Maintain appropriate level of security.

## 10.5 Key Features

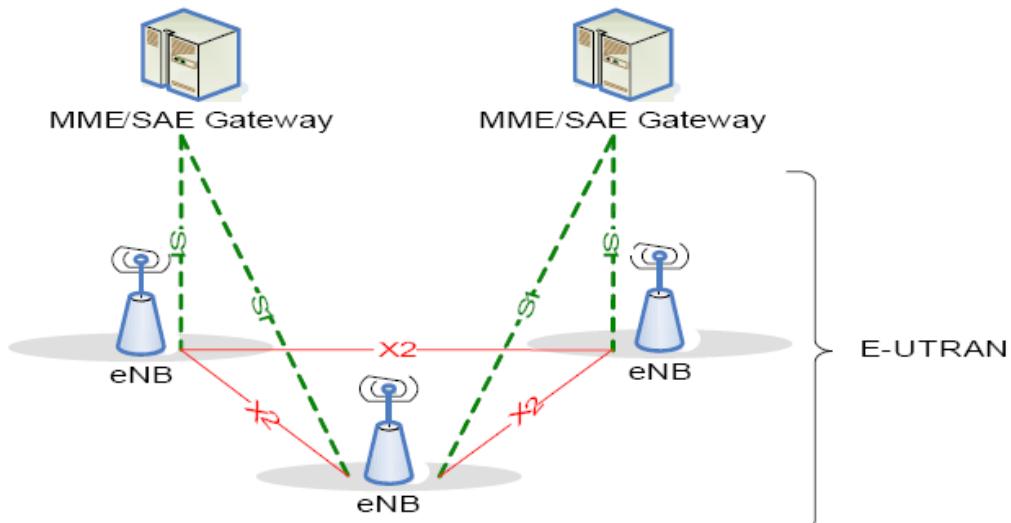
2 main issues have been investigated: the *physical layer* and the *access network internal architecture*. Physical layer is based on SC-FDMA for the Uplink and OFDMA for the Downlink.

OFDMA offers improved spectral efficiency, capacity etc. SC-FDMA is technically similar to OFDMA but is better suited for uplink from hand-held devices (battery power considerations). Two modes FDD and TDD (with similar framing + an option for TD SCDMA framing also) are considered. User Equipment has to support both. *Spectrum flexibility*: E-UTRA operates in 1.25, 1.6, 2.5, 5, 10, 15 and 20 MHz allocations... hence allowing different possibilities for re-farming already in use spectrum (uplink and downlink, paired and unpaired). *Co-existence*: with GERAN/3G on adjacent channels, with other operators on adjacent channels, with overlapping or adjacent spectrum at country borders. It's possible the Handover with UTRAN and GERAN, and the Handover with non 3GPP Technologies (CDMA 2000, WiFi, WiMAX). *Access Network consideration*: for the access network it was agreed to get rid of the RNC which minimized the number of nodes. The access network is simplified and reduced to only the Base Station called eNode B. LTE targets are:

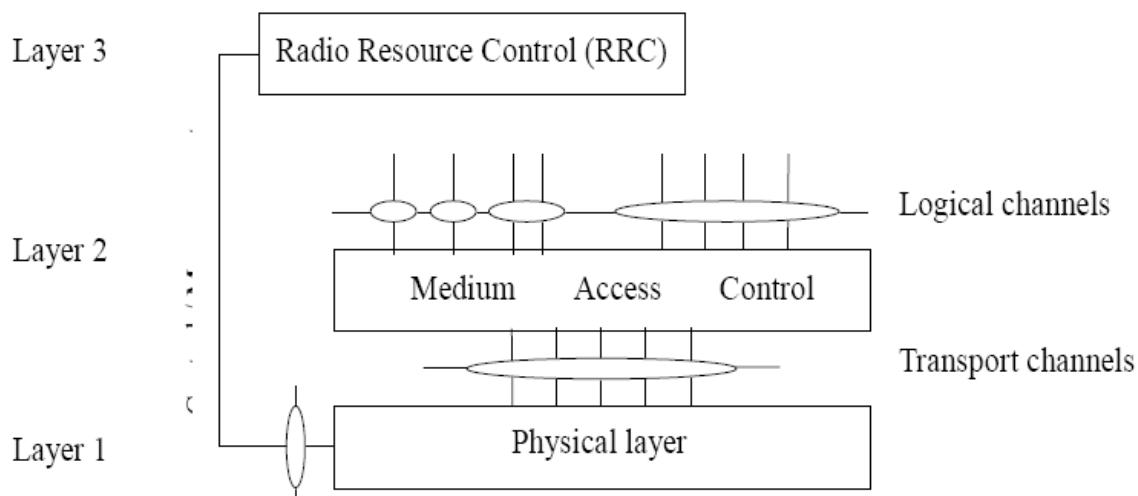
- Significantly increased peak data rates
- Increased cell edge bitrates
- Improved spectrum efficiency
- Improved latency
- Scalable bandwidth
- Acceptable system and terminal complexity, cost and power consumption
- Compatibility with earlier releases and with other systems
- Optimized for low mobile speed but supporting high mobile speed

## 10.6 Physical Layer

Architecture Overview:

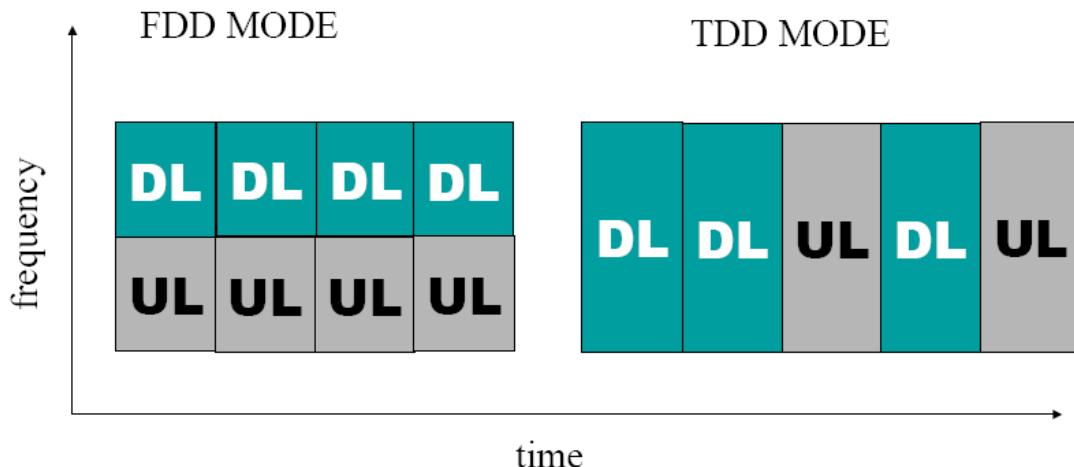


Layers Overview:

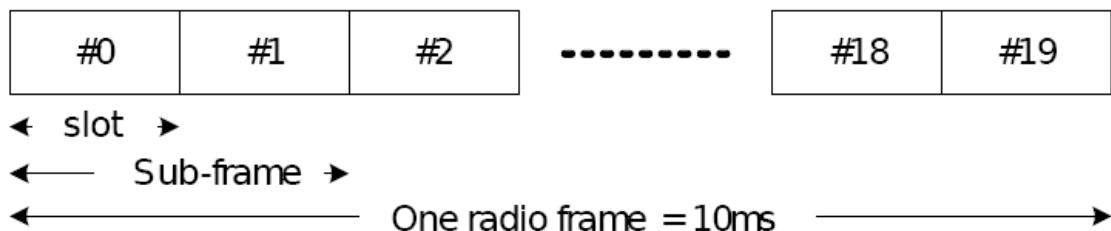


The Layer 1 is defined in a bandwidth agnostic way, allowing the LTE Layer 1 to adapt to various spectrum allocations. The generic radio frame for FDD and TDD has a duration of 10ms and consists of 20 slots with a slot duration of 0.5ms. Two adjacent slots form one sub-frame of length

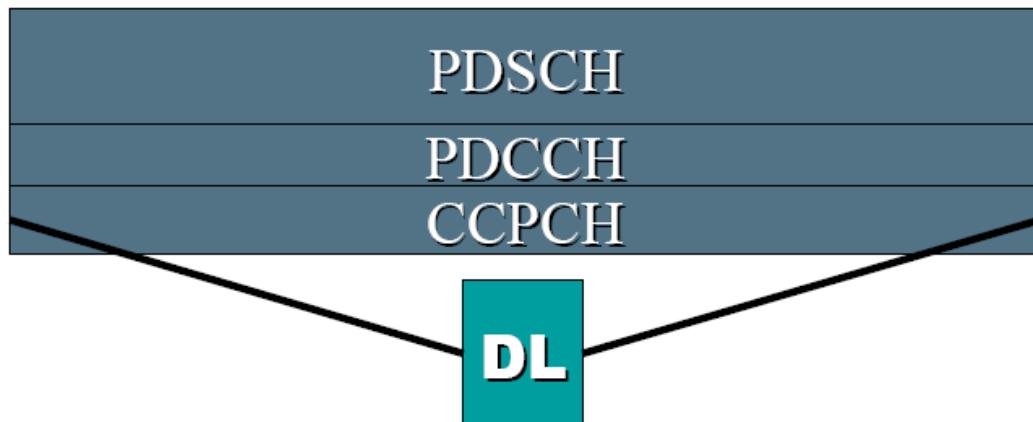
1ms.



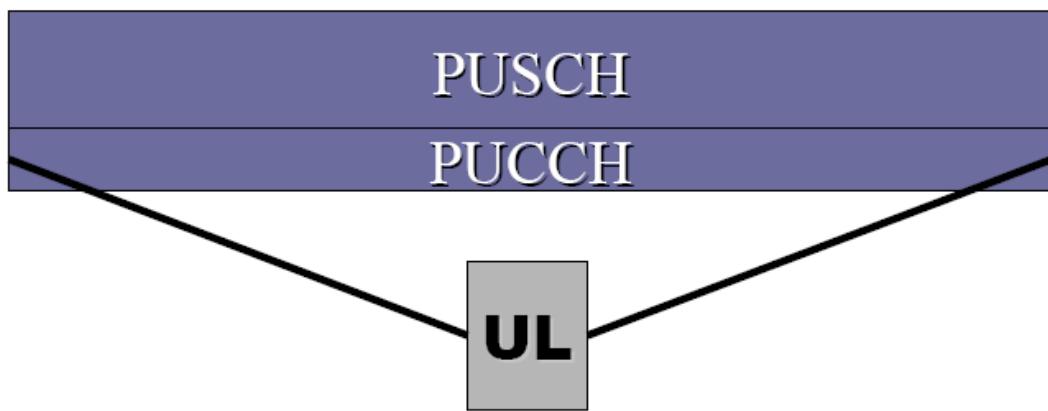
Each 10 ms radio frame is divided into ten equally sized sub-frames. Each sub-frame consists of two equally sized slots. Each sub-frame can be assigned for either downlink or uplink transmission.



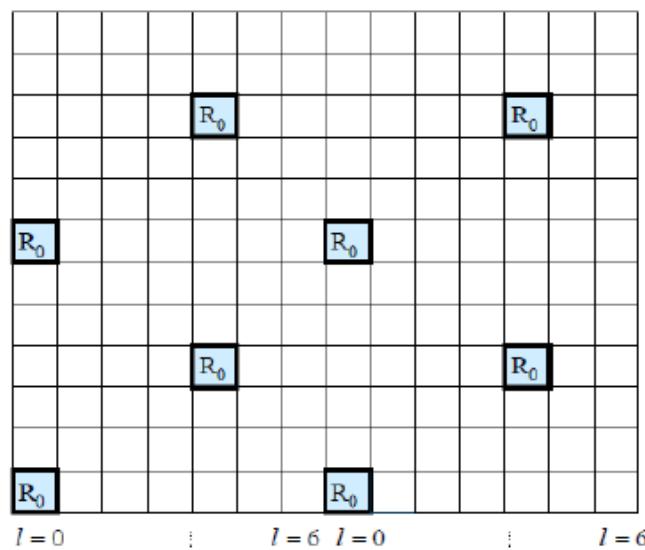
The physical channels defined in the downlink are the Physical Downlink Shared Channel (PDSCH), the Physical Downlink Control Channel (PDCCH) and the Common Control Physical Channel (CCPCH). The actual downlink communication flows through PDSCH.



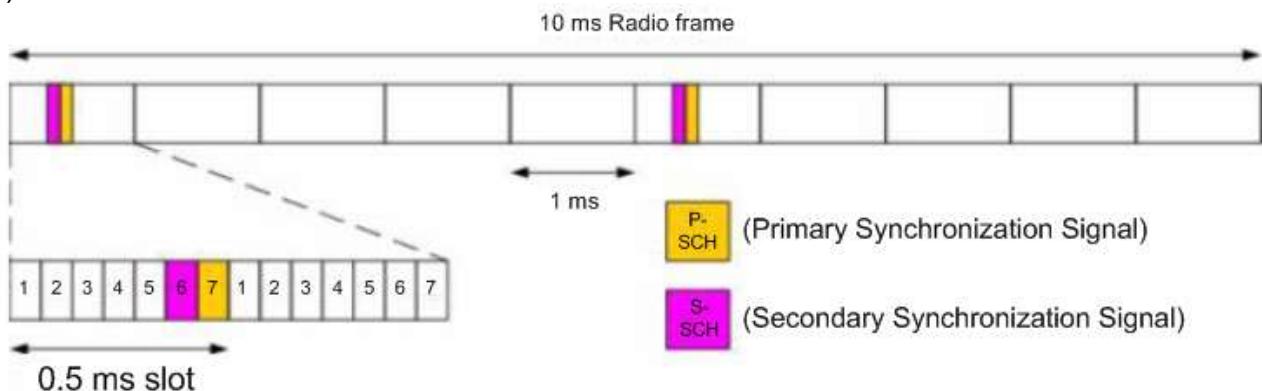
The physical channels defined in the uplink are the Physical Uplink Shared Channel (PUSCH) and the Physical Uplink Control Channel (PUCCH). The actual uplink communication flows through PUSCH.



OFDM reference signals sequence carries the cell identity.

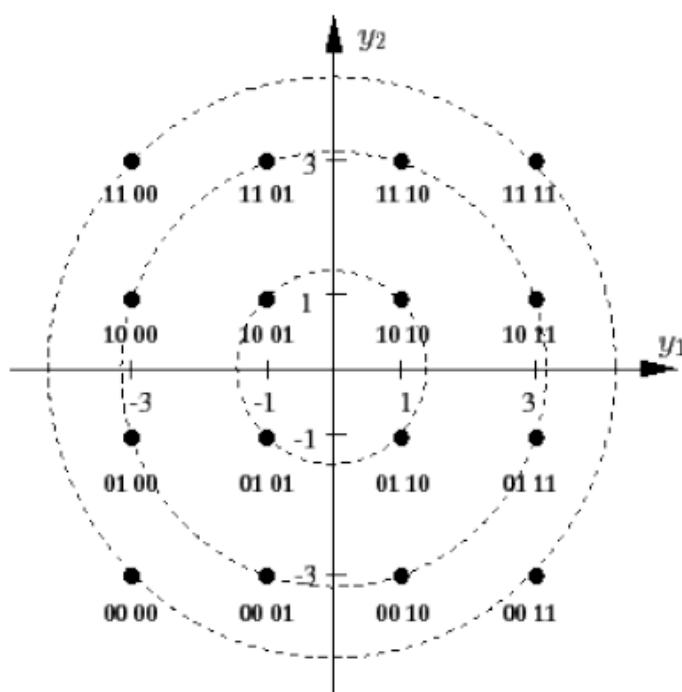


*OFDM primary and secondary synchronization signals* are needed during cell search. The synchronization acquisition and the cell group identifier are obtained from different SCH signals. They are transmitted on the 72 centre sub-carriers within the same predefined slots (twice per 10 ms) on different resource elements.



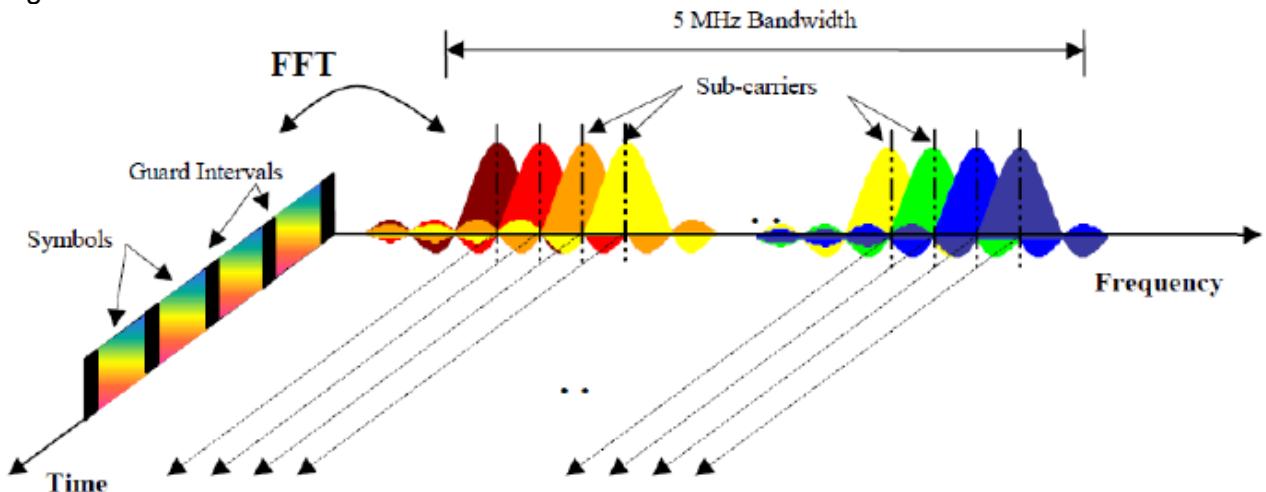
The modulation schemes supported are (both DL and UL) QPSK, 16QAM and 64QAM. The Broadcast channel use only QPSK. Modulation schemes are decided by the eB-node on every scheduling decision, basing on periodic CQI (Channel Quality Indicator) reports by the UEs. An eB-node makes a scheduling decision every 1ms.

16QAM Constellation:

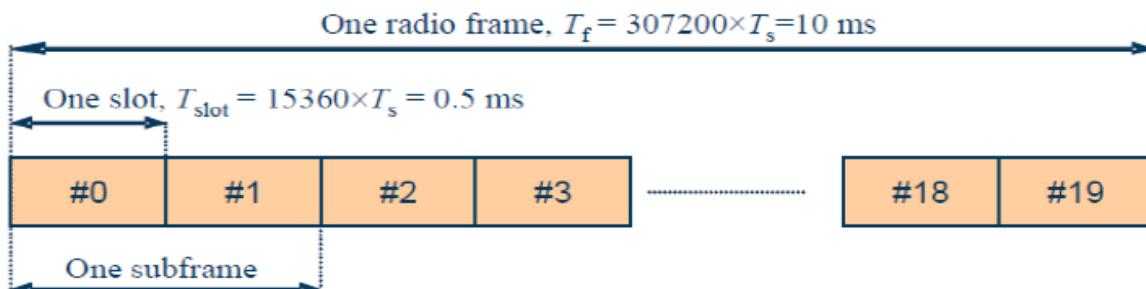


## 10.7 OFDMA (Downlink)

The downlink transmission scheme for E-UTRA FDD and TDD modes is based on conventional OFDM. The available spectrum is divided into multiple carriers, called sub-carriers, which are orthogonal to each other.

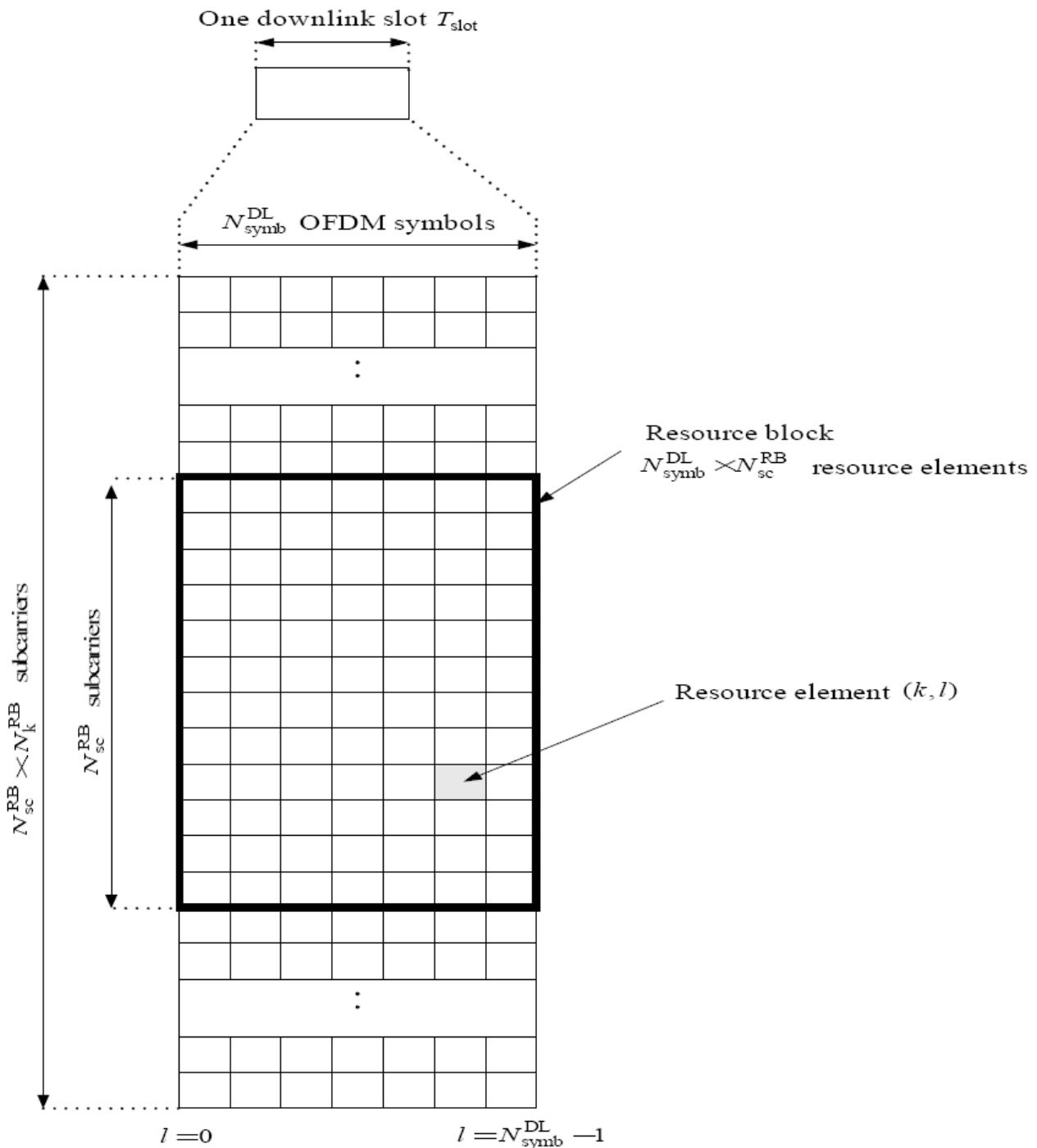


OFDM is used as well in WLAN, WiMAX and broadcast technologies like Digital Video Broadcasting (DVB). OFDM has several benefits including its robustness against multipath fading and its efficient receiver architecture. In contrast to an OFDM transmission scheme, *OFDMA* allows the access of multiple users on the available bandwidth. Each user is assigned a specific time-frequency resource, i.e. for each transmission time interval of 1 ms, a new scheduling decision is taken regarding which users are assigned to which time/frequency resources during this transmission time interval. A generic frame structure is defined for both E-UTRA FDD and TDD modes. Additionally, an alternative frame structure is defined for the TDD mode only.



For the generic LTE/E-UTRA frame structure, the 10 ms radio frame is divided into 20 equally sized slots of 0.5 ms. A subframe consists of two consecutive slots, so one radio frame contains 10 sub-frames. From the perspective of the MAC layer, each slot is viewed as a contiguous list of Resource Blocks (RBs). In downlink, a subset of the resource elements in a RB can be dedicated to control signalling:

- transport format, resource allocation, and Hybrid-ARQ information related to DL-SCH;
- uplink scheduling grant;
- if H-ARQ is used, ACK/NACK in response to uplink transmission.



The quantity  $N^{\text{RB}}$  depends on the downlink transmission bandwidth. The number of OFDM symbols in a slot depends on the cyclic prefix length and sub-carrier spacing. Number of bits in a Resource Element depends on chosen modulation. The available downlink bandwidth consists of DL  $N^{\text{BW}}$  subcarriers with a spacing of 15 kHz. DL  $N^{\text{BW}}$  can vary in order to allow for scalable bandwidth operation up to 20 MHz. One downlink slot consists of DL  $N_{\text{symb}}^{\text{DL}}$  OFDM symbols. The generic frame structure with normal cyclic prefix length contains DL  $N_{\text{symb}}^{\text{DL}}=7$  symbols. Data is allocated to the UEs in terms of *resource blocks*. A physical resource block consists of 12 (24) consecutive subcarriers in the frequency domain for the Nf=15 kHz (Nf=7.5 kHz) case. In the time domain, a physical resource block consists of DL  $N_{\text{symb}}^{\text{DL}}$  consecutive OFDM symbols.

## 10.8 SC-FDMA (Uplink)

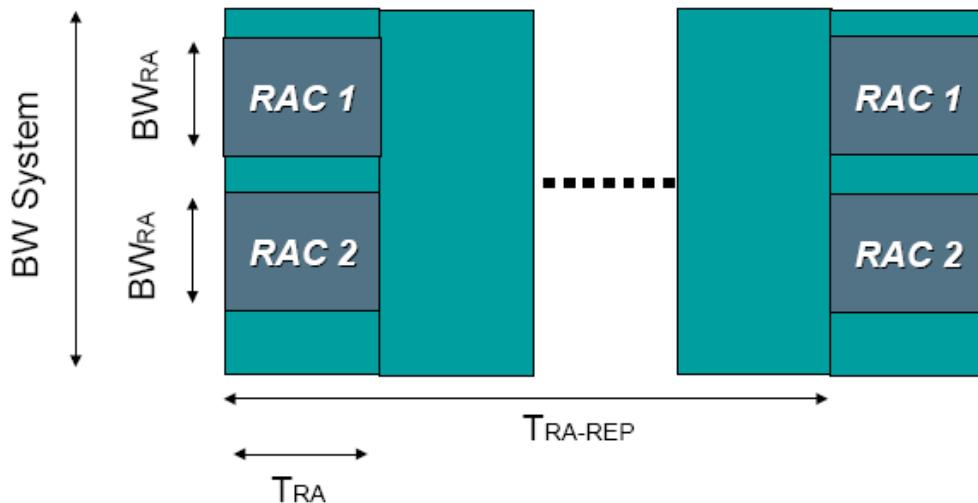
The LTE uplink transmission scheme for FDD and TDD mode is based on **SC-FDMA** (Single Carrier Frequency Division Multiple Access). In SC-FDMA, all the assigned RBs have to be on the same set of contiguous sub-carriers. This is not true on OFDMA scheme. However, this set of sub-carriers can change from a scheduling decision to the next. SC-FDMA signals have better Peak to Average Power Ratio properties compared to an OFDMA signal. SC-FDMA signal processing has some similarities with OFDMA signal processing, so parameterization of downlink and uplink can be harmonized. The E-UTRA uplink structure is similar to the downlink. An uplink radio frame

consists of 20 slots of 0.5 ms each, and 1 subframe consists of 2 slots. Each slot carries UL  $N_{\text{symb}}$  SC-FDMA symbols, where UL  $N_{\text{symb}}=7$ . Symbol number 3 (i.e. the 4th symbol in a slot) carries the reference signal for channel demodulation. In uplink, data is allocated in multiples of one resource block. Uplink resource block size in the frequency domain is 12 subcarriers, the same as in downlink. The uplink transmission time interval is 1 ms (same as downlink).

## 10.9 RAC Procedure

For E-UTRA, the following uplink physical layer procedure is especially important.

NON Synchronized Random Access:



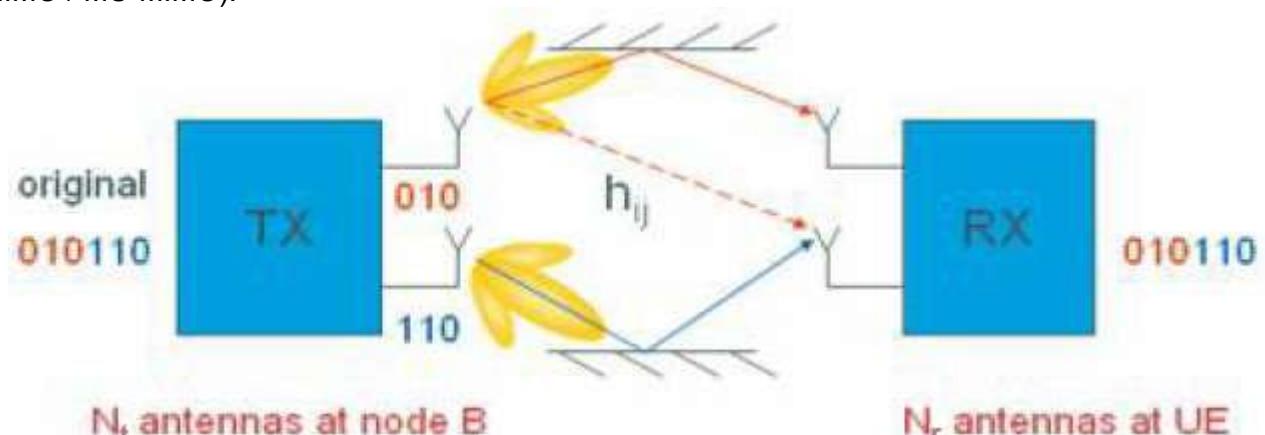
The random access may be used to:

- request initial access,
- as part of handover,
- when transiting from idle to connected,
- to reestablish uplink synchronization.

The random access procedure uses open loop power control with power ramping similar to WCDMA. After sending the preamble on a selected random access channel, the UE waits for the random access response message. If no response is detected then another random access channel is selected and a request is sent again.

## 10.10 MIMO

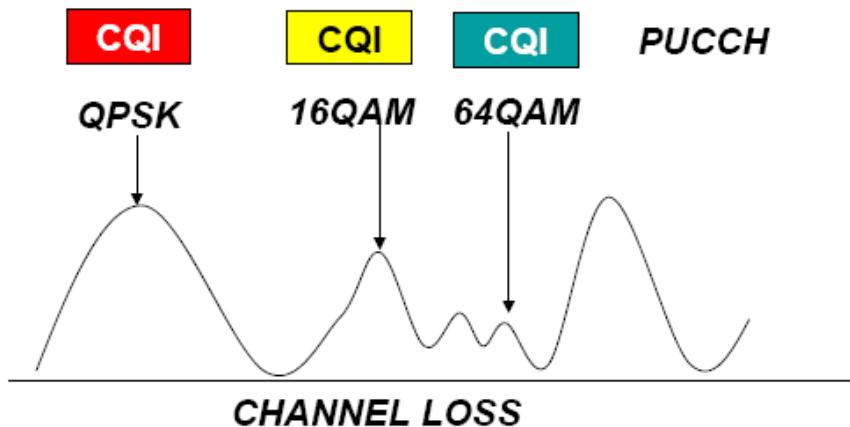
MIMO refers to the use of multiple antennas at transmitter and receiver side. In both uplink and downlink, only one transport block is generated per transmission time interval in the non-MIMO case. For the LTE downlink, a 2x2 configuration for MIMO is assumed as baseline configuration. Configurations with 4 antennas are also being considered. *Spatial multiplexing* allows to transmit different streams of data simultaneously on the same downlink resource block(s). Data streams can belong to one single user (single user MIMO / SU-MIMO) or to different users (multi user MIMO / MU-MIMO).



MU-MIMO can be used in uplink. Multiple user terminals may transmit simultaneously on the same resource block. This is called *Spatial Domain Multiple Access (SDMA)*. The scheme requires only one transmit antenna at UE side.

## 10.11 Link Adaptation

*Link adaptation* is already known from HSPA as Adaptive Modulation and Coding. Also in E-UTRA, modulation and coding for the data channel is not fixed, but it is adapted according to radio link quality. For this purpose, the UE regularly reports Channel Quality Indications (CQI) to the eNodeB.



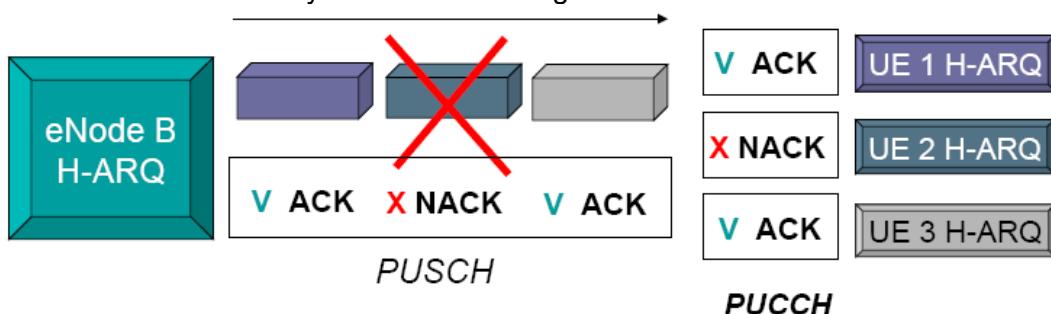
As uplink link adaptation methods can be used:

- transmission power control,
- adaptive modulation and channel coding rate,
- adaptive transmission bandwidth.

## 10.12 HARQ (Hybrid Automatic Repeat reQuest)

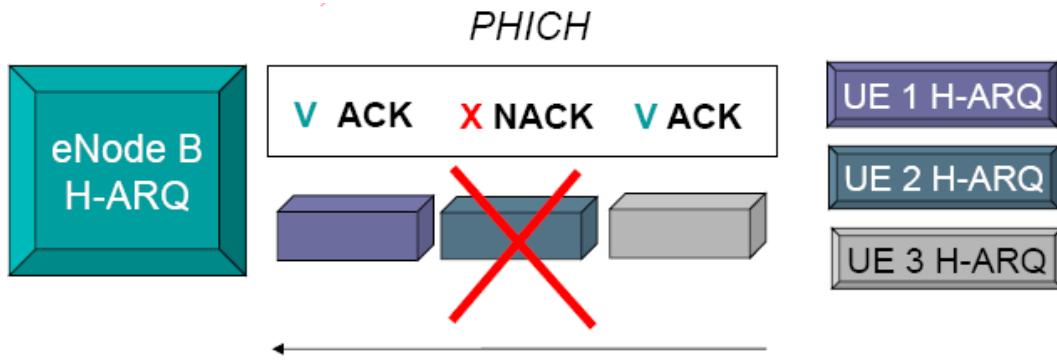
Hybrid ARQ is also known from HSDPA/HSUPA. It is a retransmission protocol. The eNodeB/UE can request retransmissions of incorrectly received data packets. The HARQ functionality ensures delivery between peer entities at layer 1. HARQ is “hybrid” in the sense that it can request retransmission of incorrectly delivered data but can also reconstruct the original data from two damaged retransmissions without requesting a further retransmission. The HARQ is an N channel stop-and-wait protocol with asynchronous downlink retransmissions and synchronous uplink retransmissions. Downlink HARQ characteristics are:

- Asynchronous adaptive HARQ: “asynchronous” means “window-based ACK management”;
- Uplink ACK/NACKs in response to downlink (re)transmissions are sent on PUCCH or PUSCH;
- PDCCH signals the HARQ process number and if it is a transmission or retransmission;
- Retransmissions are always scheduled through PDCCH.



Uplink HARQ characteristics are:

- Synchronous HARQ: “synchronous” means “stop-and-wait ACK management”;
- Maximum number of retransmissions configured per UE (as opposed to per radio bearer);
- Downlink ACK/NACKs in response to uplink (re)transmissions are sent on PHICH (Physical Hybrid Automatic Repeat Request Indicator Channel).



### 10.13 Layer 2

The eNode B hosts the following functions:

- Radio Bearer Control,
- Radio Admission Control,
- Connection Mobility Control,
- Dynamic allocation of resources to UEs in both uplink and downlink (scheduling).

### 10.14 Scheduling

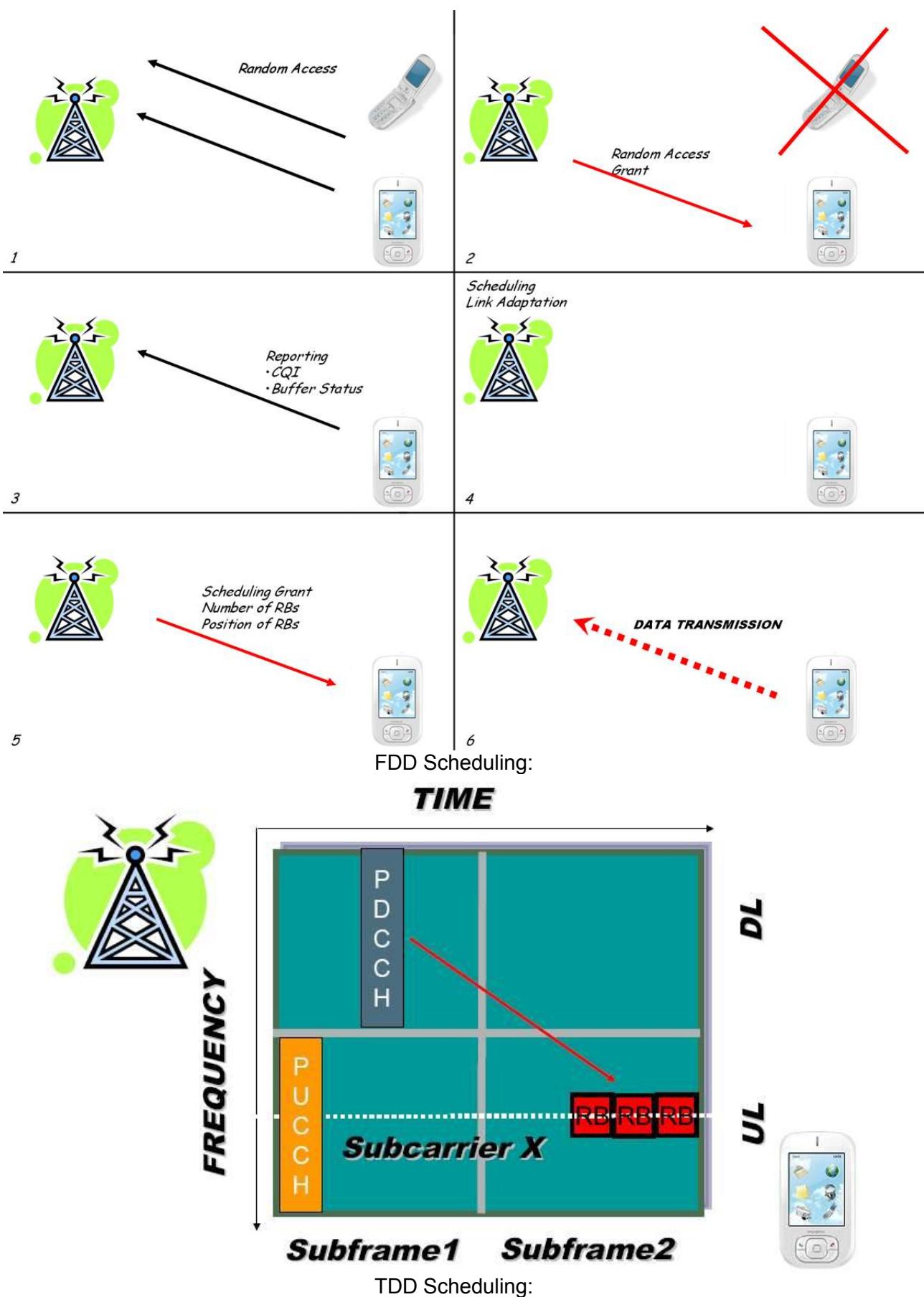
Scheduling is done in the base station (eNode B). The downlink control channel PDCCH informs the users about their allocated time/frequency resources and the transmission formats to use. The scheduler evaluates different types of information: Quality of Service parameters (measurements from the UE), UE capabilities, buffer status. Downlink scheduling is based on the following factors:

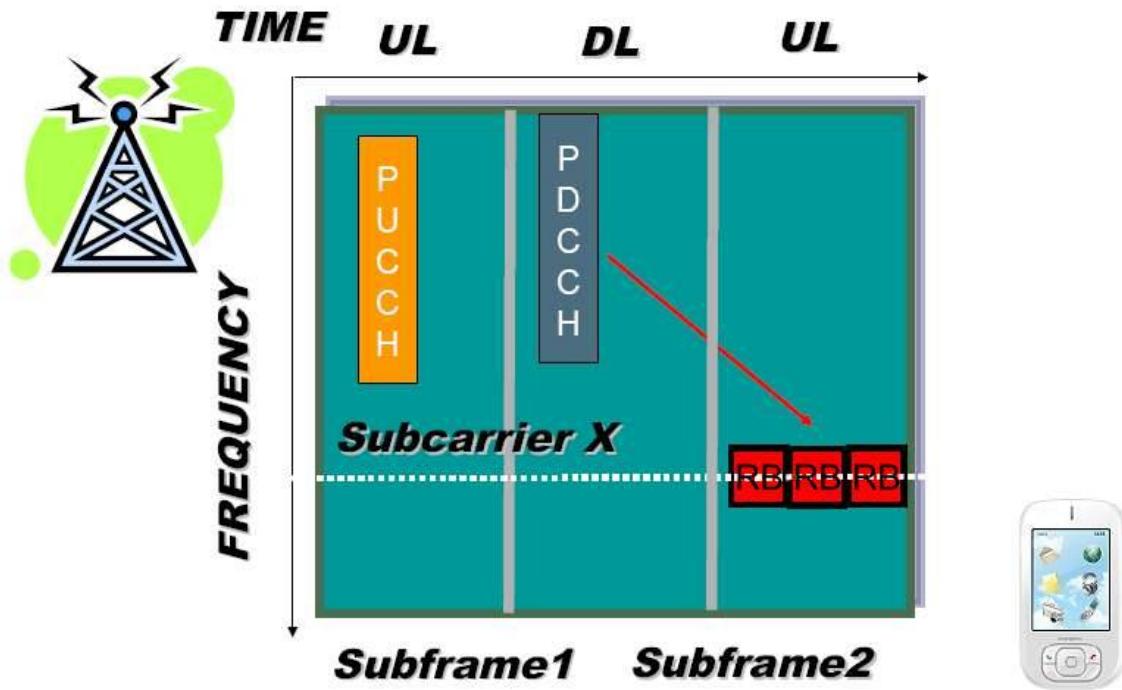
- the current status of the control messages waiting transmission, including, e.g., power control indications and ACK/NACK reports for H-ARQ transmission;
- the current status of the buffers of downlink radio bearers, possibly including H-ARQ retransmissions;
- the QoS parameters of downlink radio bearers;
- measurements from the PHY layer.

Uplink scheduling is based on the following factors:

- the virtual backlog of uplink radio bearers, as acquired by means of the Buffer Status Reports (BSRs) conveyed by the UEs in the previous frames;
- the QoS parameters of uplink radio bearers;
- measurements from the PHY layer of the UEs, as conveyed by CQI (Channel Quality Indicator) messages.
- only “per UE” grants are used to grant the right to transmit (i.e. there are no “per UE per RB” grants).

Upload Scheduling Example:

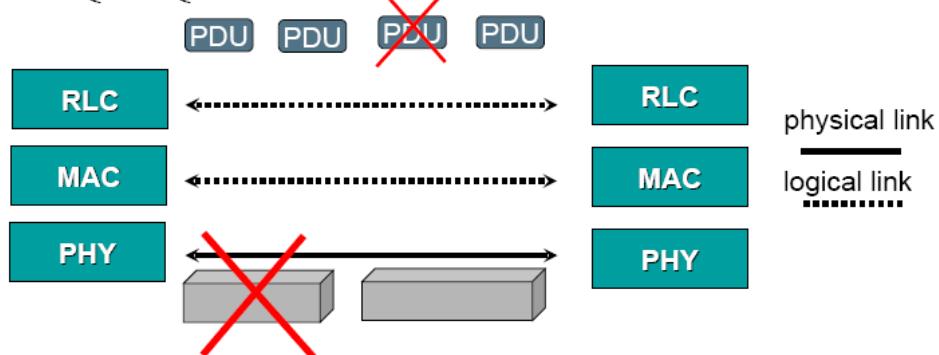




### 10.15 ARQ

E-UTRAN provides ARQ and HARQ functionalities. The ARQ functionality provides error correction by retransmissions in acknowledged mode at layer 2. ARQ retransmissions are based on RLC status reports and HARQ/ARQ interaction. This means that layer-2 ARQ doesn't request retransmission while layer-1 HARQ is doing the same thing.

HARQ/ARQ interaction.



### 10.16 Service Classes

- Conversation



- Real-time interactive



Broadcast Yourself™

- Non-real-time streaming
- Best-effort



QoS parameters are:

- Guaranteed Bit Rate (GBR) or non-GBR;
- [GBR only] Prioritized Bit Rate (PBR) and priority;
- Maximum Bit Rate (MBR).

## 10.17 Core Network

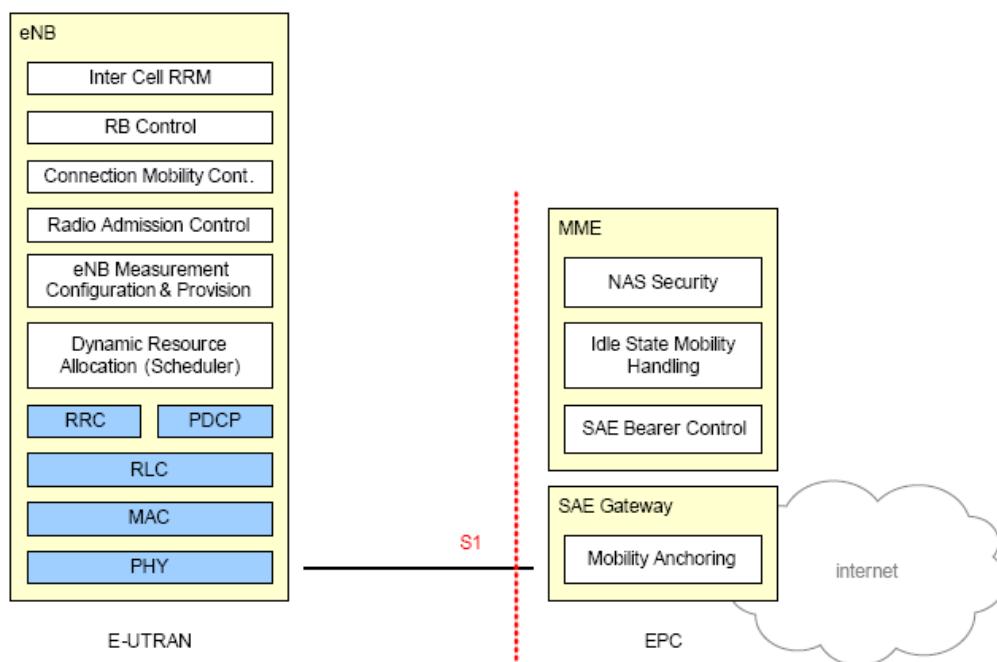
Interface towards the Core network has two interfaces:

- S1 for the Control plane
- X1 for the User plane (new)

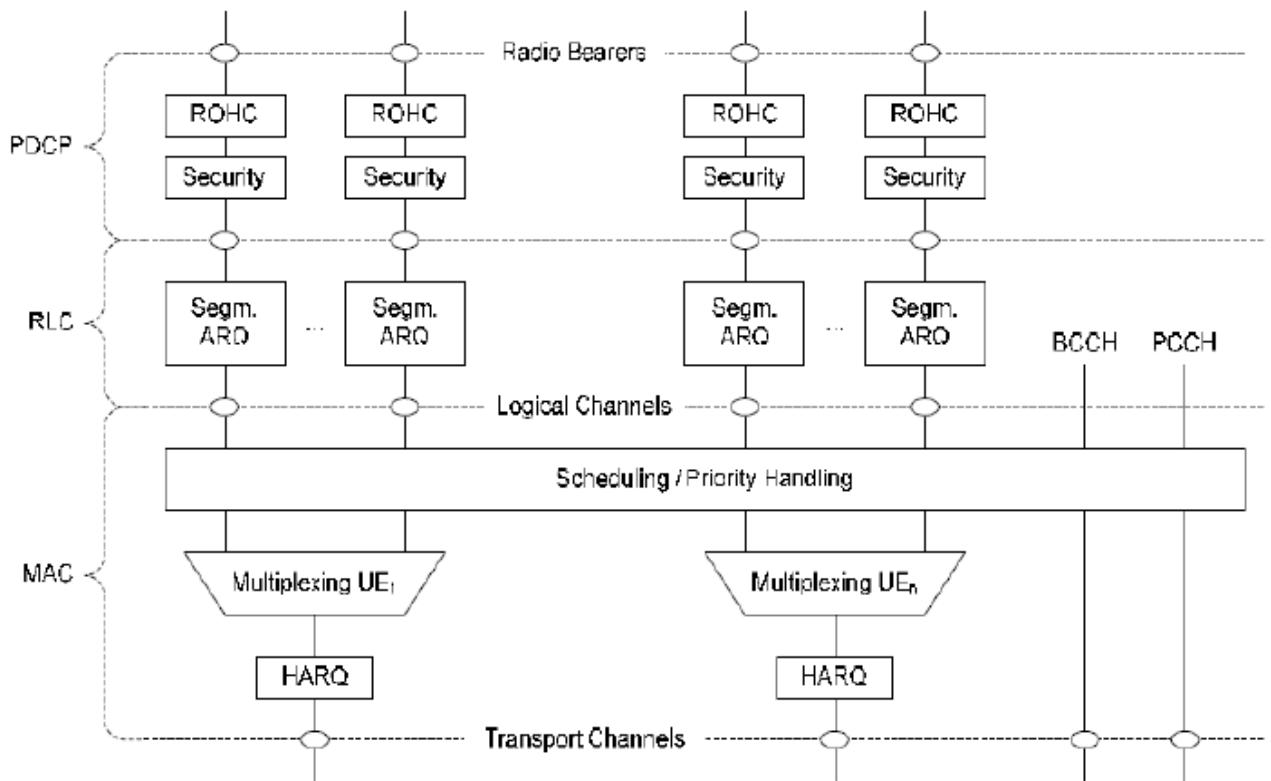
and an additional interface in between eNode Bs:

- X2, including both Control and User plane.

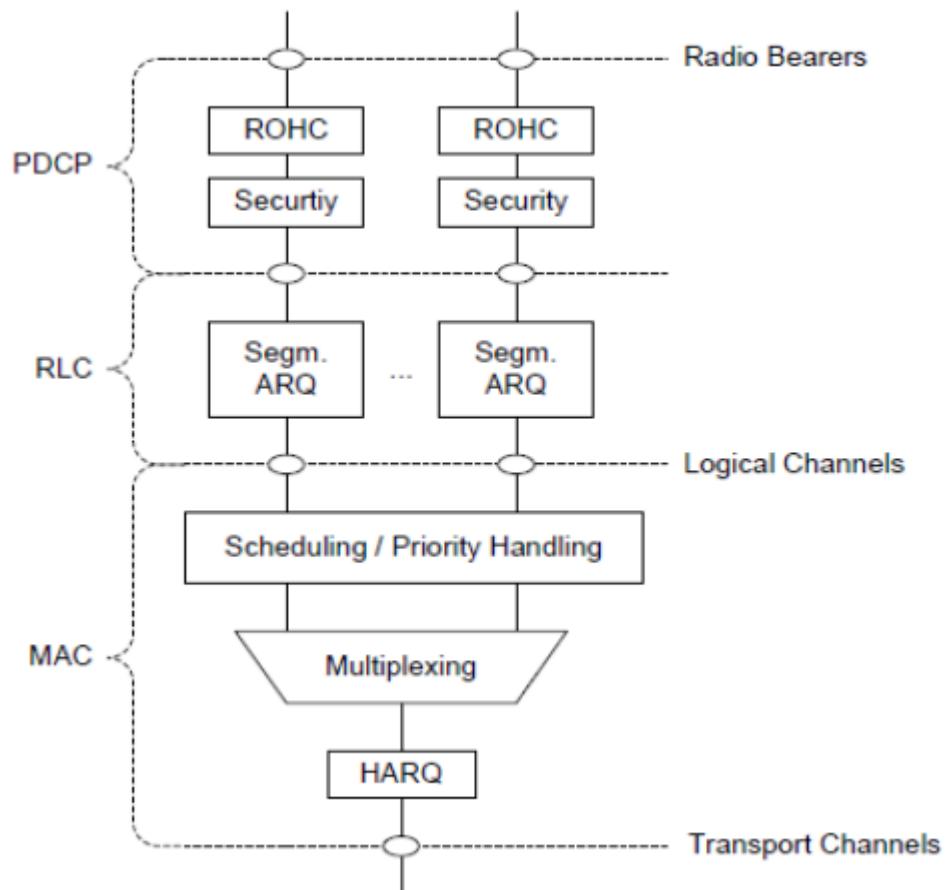
Interface towards the Core network:



## 10.18 eNodeB

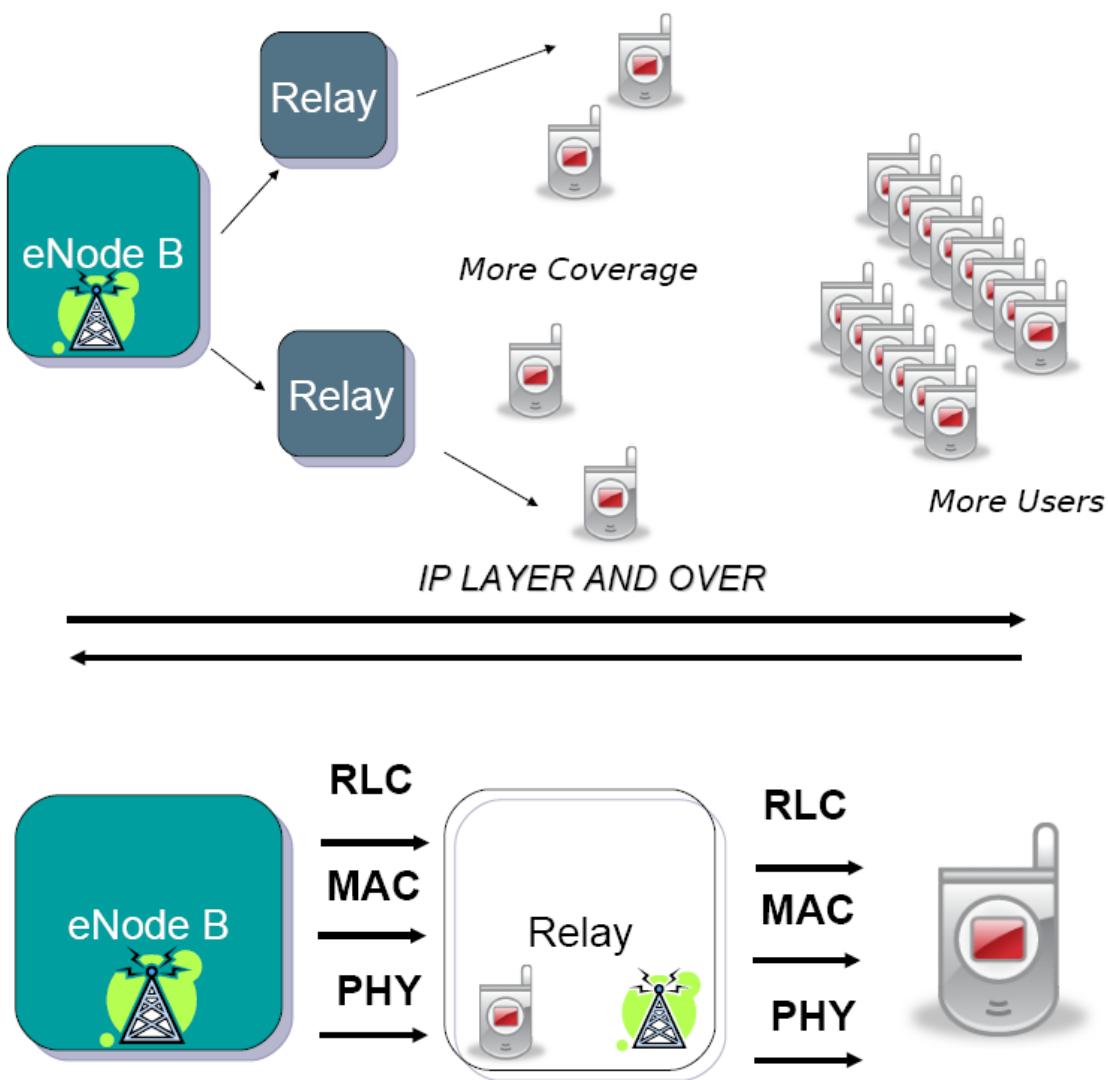


### 10.19 Mobile Terminal



### 10.20 Beyond LTE

LTE Advanced and the Radio Relay Concept:



## 10.21 Conclusions

LTE exists to ensure competitiveness of 3GPP systems for the next 10 years and beyond. It assures optimization of the network for IP traffic and its expected growth. It has performance improvements:

- reduced latency,
- higher user data rates,
- improved system capacity and coverage, and reduced overall cost for the operator,
- potential network and traffic cost reduction.

There is flexible accommodation and deployment of existing and new access technologies with mobility by a common IP-based network. Targets are:

- Instantaneous downlink peak data rate of 100Mbit/s in a 20MHz downlink spectrum (i.e. 5 bit/s/Hz).
- Instantaneous uplink peak data rate of 50Mbit/s in a 20MHz uplink spectrum (i.e. 2.5 bit/s/Hz).

The Enhanced UTRAN (E-UTRAN) will:

- be optimized for mobile speeds 0 to 15 km/h,
- support, with high performance, speeds between 15 and 120 km/h,
- maintain mobility at speeds between 120 and 350 km/h and even up to 500 km/h depending on frequency band,
- support voice and real-time services over entire speed range with quality at least as good as UTRAN.

# 11 Wireless Mesh Networks

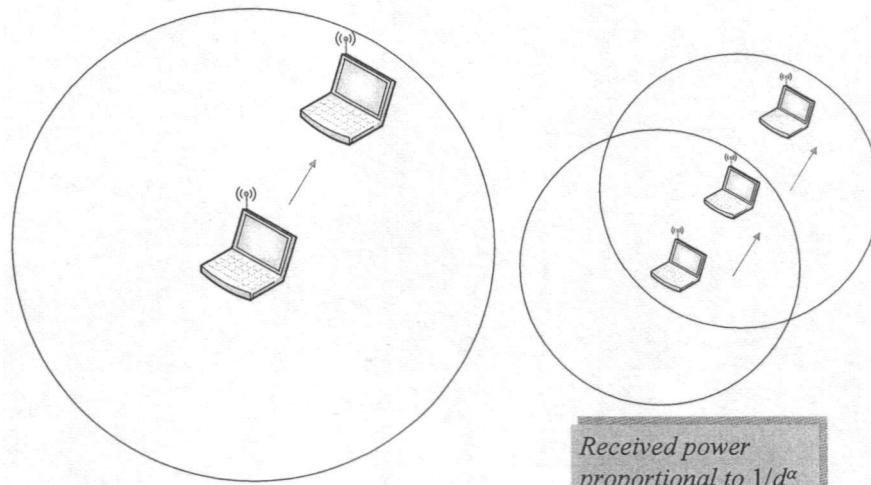
## 11.1 Why Wireless Mesh Networks?

The concept of wireless *multihop* networks dates back to 1970s: *DARPA Packet Radio NETwork (PRNET)*. Development languished in 1980s partially due to the lack of low cost CPU and memory for ad hoc routing. It was rekindled since about 1995. Enabling technologies were:

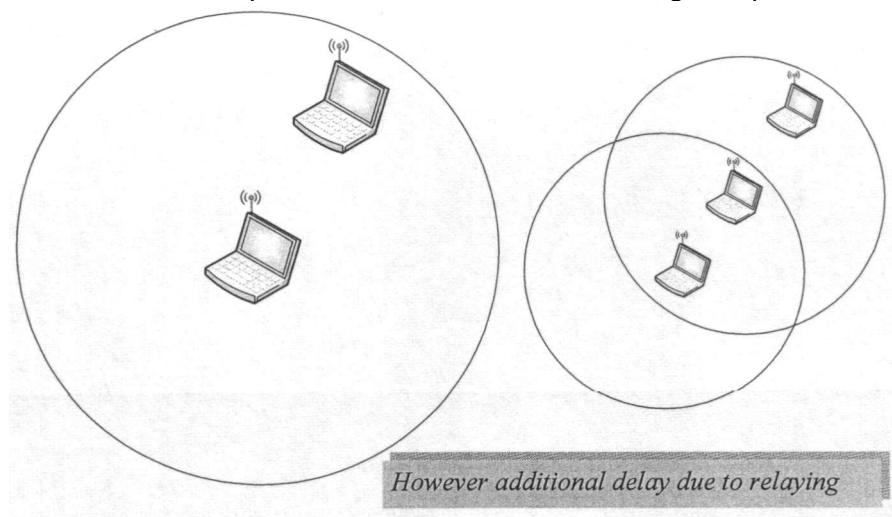
- miniaturization,
- battery technology,
- smart antennas,
- user terminal evolution,
- new frequency bands,
- etc.

Battery technology has lagged behind processor/memory/radio technology. More power was needed for longer range. The fundamental physical limitation is *path loss*: received power proportional to  $1/d^\alpha$ . The solution is use short range radios and multiple hops for communication.

Multihop saves power over single hop:



Multihop reduces interference over single hop:



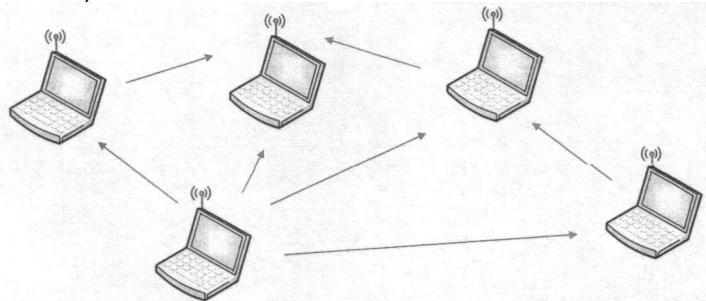
## 11.2 History of Wireless Multihop Networks

- Mid 1990s: IETF MANET (Mobile Ad Hoc NETworks) working group was formed. Driving force was IEEE WLAN Standard 802.11 being developed.
- Late 1990s: new focus on low-power micro sensor networks. Driving force was understanding of ad hoc networks, availability of inexpensive low-power radios, microcontrollers, sensors.
- Early 2000s: interest in *Mesh Networks*. Driving force was availability of low-cost laptop/palmtop with IEEE WLAN Standard 802.11 interface. Need for ubiquitous broadband

connectivity.

### 11.3 Various Types of Multihop Wireless Networks

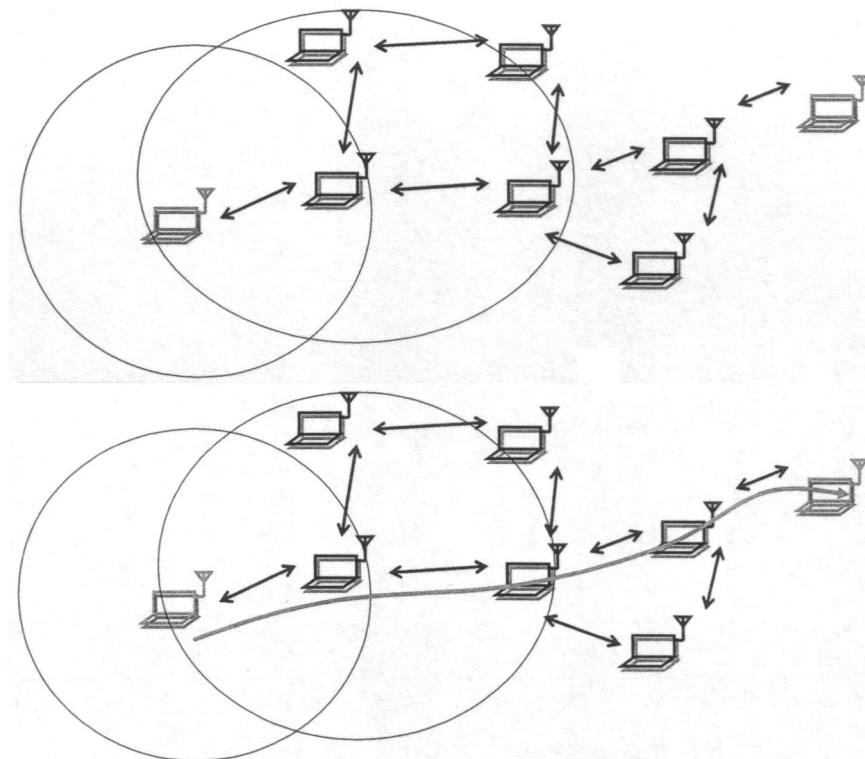
**Mobile (multihop) Ad hoc NETworks (MANETs)** are collections of mobile nodes connected together over a wireless medium. These nodes can freely and dynamically self-organize into arbitrary and temporary ad hoc network topologies, allowing people and devices to seamlessly internetwork in areas with no preexisting communication infrastructure (e.g., disaster recovery and battlefield environments).

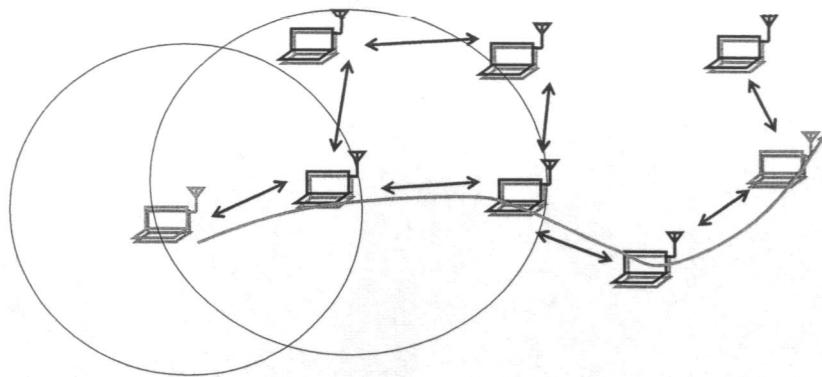


- **Ad Hoc Networks:** hosts are also routers and could be mobile.
- **Sensor Networks:** ad hoc network where hosts are actually sensor devices and are not usually mobile.
- **Mesh Networks:** hosts and routers are different entities. Usually hosts are mobile, routers are not.

### 11.4 Wireless Network in Ad Hoc Configuration

- No cellular infrastructure
- Multi-hop wireless links
- Data must be routed via intermediate nodes
- Routing must be dynamic because of mobility





## 11.5 Why Ad Hoc Networks?

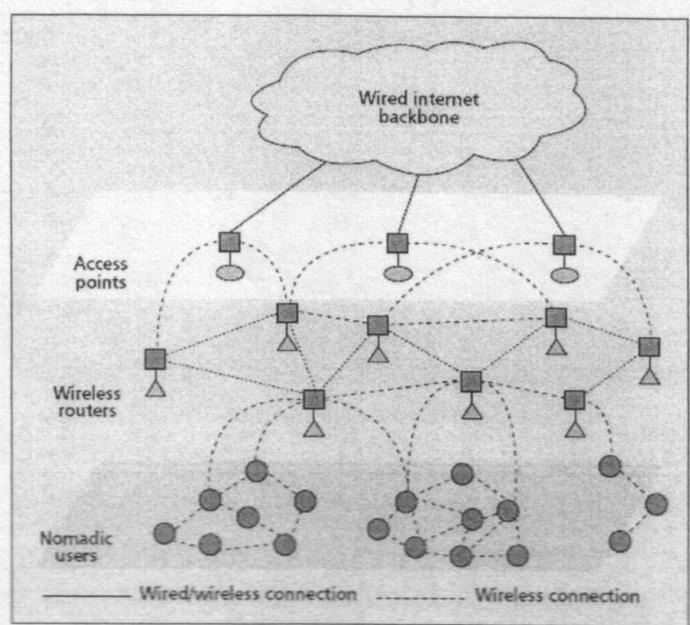
**Self-supporting:** Does not need infrastructure support. No service provider. No compatibility issues with the infrastructure-based protocols etc. On-the-fly networking. **Decreased dependence on infrastructure:** infrastructure may be destroyed in a war zone, or after earthquake. Infrastructure may not be practical for short-range. There are many applications like:

- **Military environments:** Soldiers, tanks, planes, robots. Initial work on packet radio networks (PRNET) funded by US. DARPA in the eighties.
- **Emergency operations/explorations/sensing:** Search-and-rescue operations, mobile robot-based explorations in remote regions.

## 11.6 What is a Mesh Network?

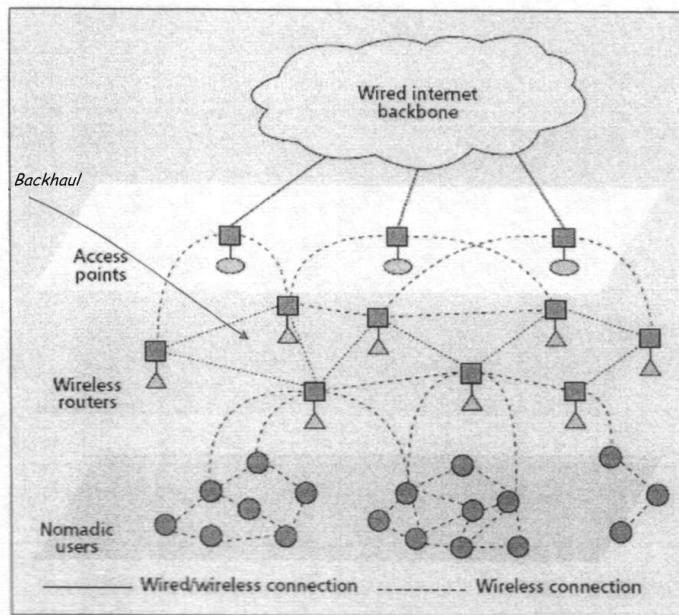
Wireless Mesh Networks (WMNs) have been envisioned as the economically viable networking paradigm to build up broadband and large-scale wireless multihop networks. In the following we elaborate on this vision to identify the unique and distinct characteristics of this new network architecture.

Several flavors of WMN architectures have been conceived by both industry and academia. However, core building blocks and distinct features may easily be identified in wireless mesh architecture. A wireless mesh network is a *fully wireless* network that employs multihop communications to forward traffic to and from wired Internet entry points. Different from *flat ad hoc networks*, a WMN introduces a hierarchy in the network architecture. The architecture includes dedicated nodes (called wireless mesh routers) communicating among each other and providing wireless transport services to data traveling from users to either other users or access points (also called wireless mesh routers with gateways). *Access points* are special wireless routers with a high-bandwidth wired connection to the Internet backbone.

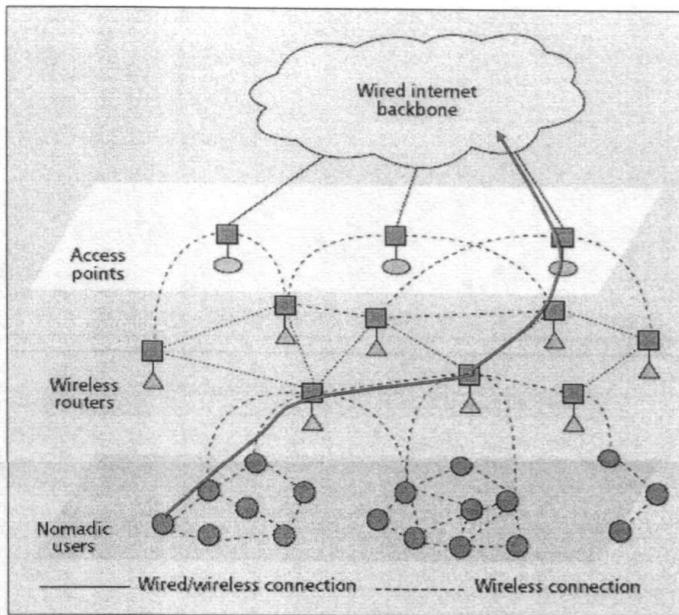


The network of wireless mesh routers forms a wireless backbone which provides multihop connectivity between nomadic users and wired gateways. The meshing among wireless routers and access points creates a wireless *backhaul* communication system, which provides

each mobile user with a low-cost, high-bandwidth, and seamless multihop interconnection service with a limited number of Internet entry points and with other wireless mobile users. Roughly and generally speaking, backhaul is used to indicate the service of forwarding traffic from the originator node to an access point from which it can be distributed over an external network.



Specifically in WMNs, the traffic is originated in the users' devices, traverses the wireless mesh backbone, and is distributed over the Internet network.



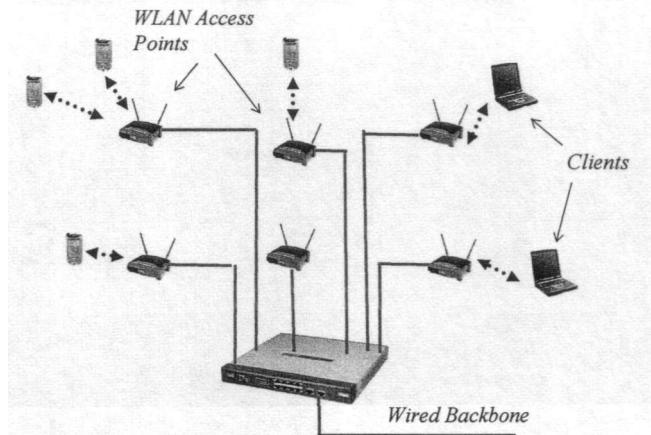
Thus, the mesh network architecture addresses the emerging market requirements for building wireless networks that are highly scalable and cost effective, offering a solution for the easy deployment of high-speed ubiquitous wireless Internet. We further elaborate on the major noticeable benefits of wireless mesh networks that provide substantial arguments in favor of the above claim. The following is not necessarily an exhaustive list of all the possible benefits, but represents an extensive discussion on the motivations behind the mesh networking vision.

## 11.7 Reduction of Installation Costs

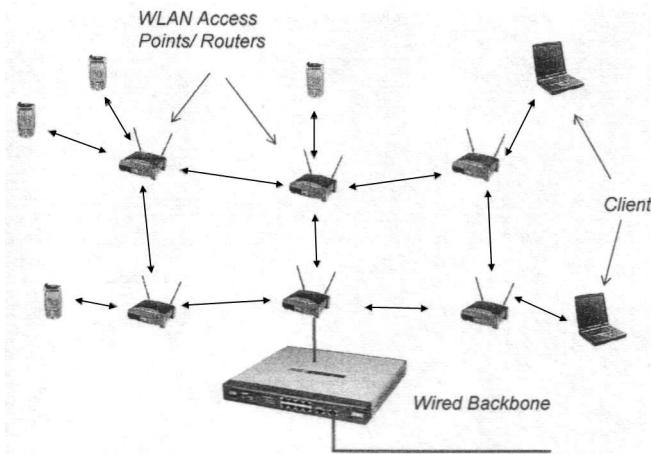
Currently, one of the major efforts to provide wireless Internet beyond the boundaries of indoor WLANs is through the deployment of Wi-Fi hot spots. To ensure almost ubiquitous coverage in a metroscale area, it is necessary to deploy a large number of access points due to the limited distance covered by the 802.11 signal. The downside of this solution is an unacceptable increase in the infrastructure costs because a cabled connection to the wired backbone is needed for every access point. Installing the necessary cabling infrastructure not only slows down hot spot implementation, but also significantly increases installation costs. As a consequence, the hot spot architecture is costly, unscalable, and slow to deploy. On the

other hand, building a mesh wireless backbone enormously reduces the infrastructural costs because the mesh network needs only a few points of connection to the wired backbone.

Wireless paradox: WLAN Access Points are typically wired:



Get rid of the wires from wireless LAN:



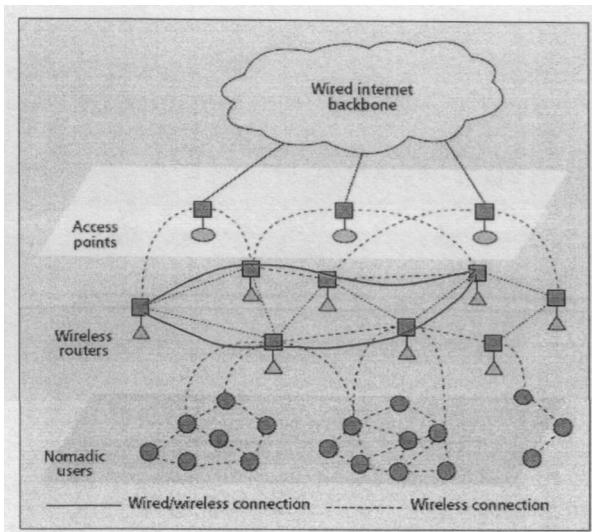
Access Points operate as "wireless routers". Wireless routers form a backbone network.

## 11.8 Large Scale Deployment

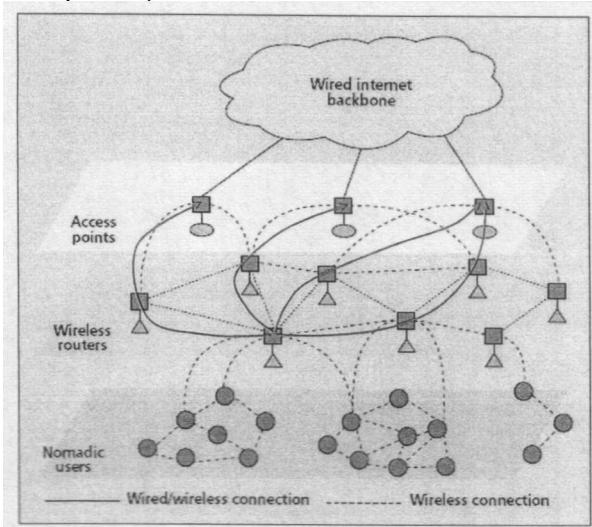
Multihop communications offers long distance communications via hopping through intermediate nodes. Since intermediate links are short, these transmissions could be at high data rates, resulting in increased throughput compared to direct communications. Moreover, the wireless backbone can take advantage of non-mobile powered wireless routers to implement more sophisticated and resource-demanding transmission techniques than those implemented in user devices. Consequently, the wireless backbone can realize a high degree of spatial reuse and wireless links covering longer distance at higher speed than conventional WLAN technologies.

## 11.9 Reliability

The wireless backbone provides redundant paths between each pair of endpoints, significantly increasing communications reliability, eliminating single points of failure and potential bottleneck links within the mesh.



Network resilience and robustness against potential problems (e.g., node failures, and path failures due to temporary obstacles or external radio interference) is also ensured by the existence of multiple possible access points toward the wired Internet, and alternative routes to these destinations (i.e. access points).

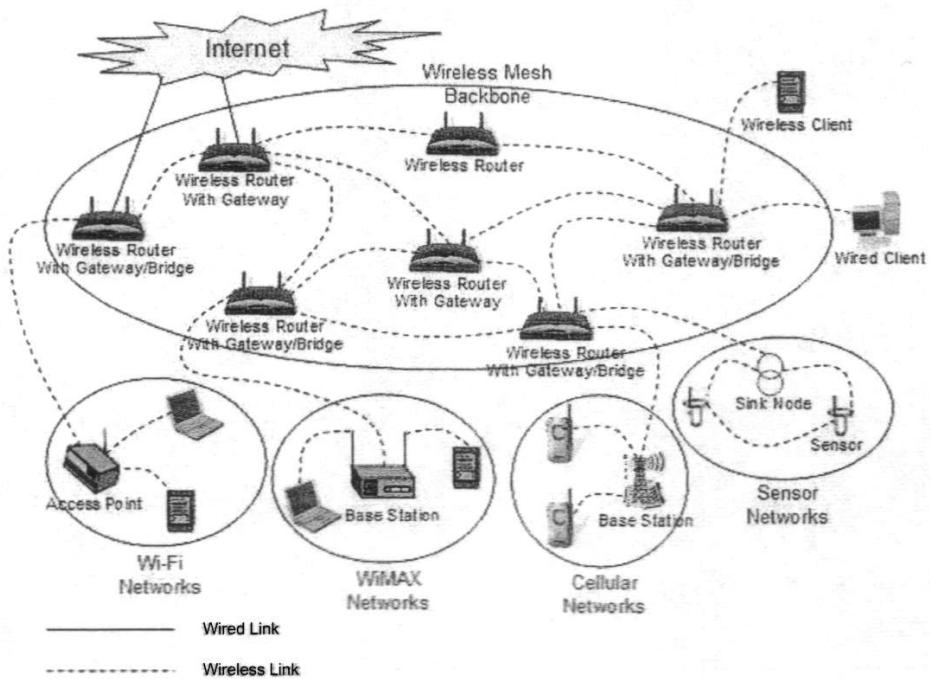
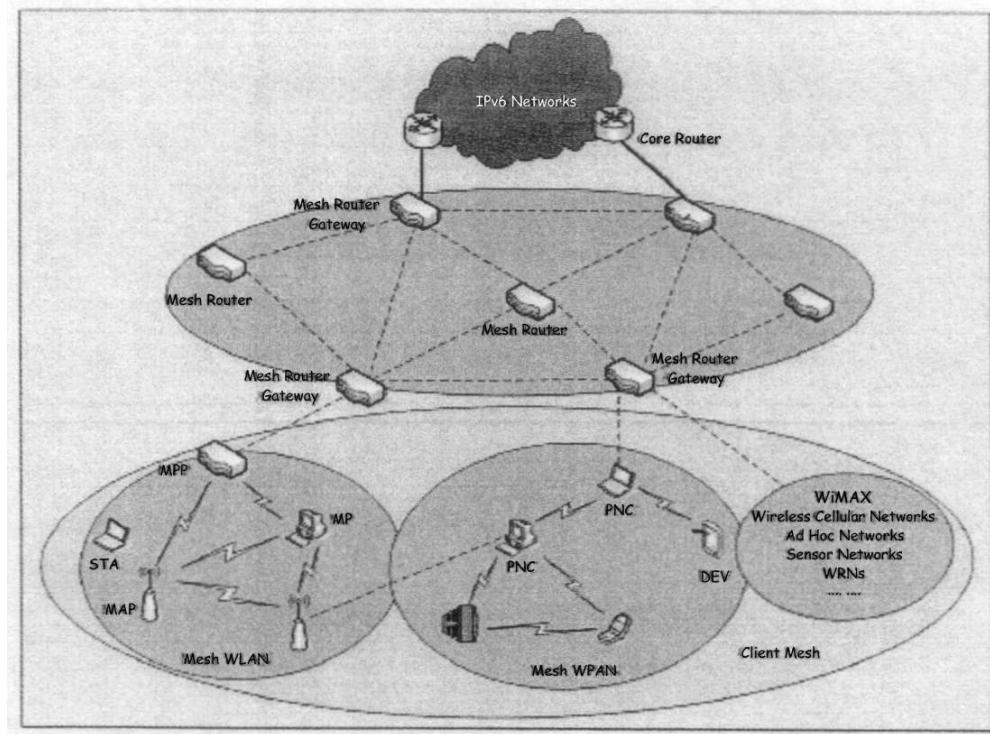


## 11.10 Self-Management

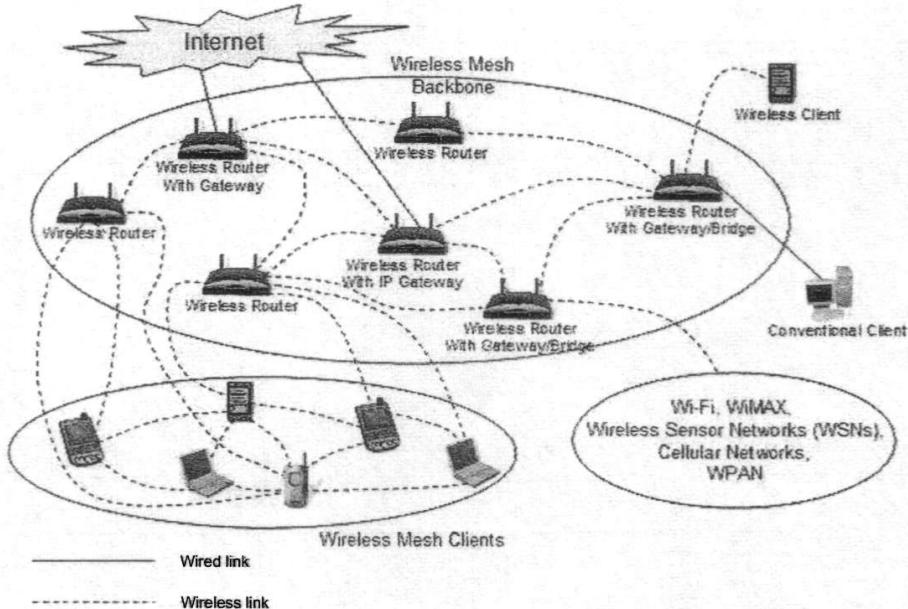
The adoption of peer-to-peer networking to build a wireless distribution system provides all the advantages of ad hoc networking, such as *self-configuration* and *self-healing*. Network setup is automatic and transparent to users. For instance, when adding additional nodes in the mesh, these nodes use their meshing functionalities to automatically discover all possible wireless routers and determine the optimal paths to the wired network. In addition, the existing wireless routers reorganize, taking into account the new available routes. The network can easily be expanded, because the network self-reconfigures to assimilate the new elements.

## 11.11 WMN Architectures

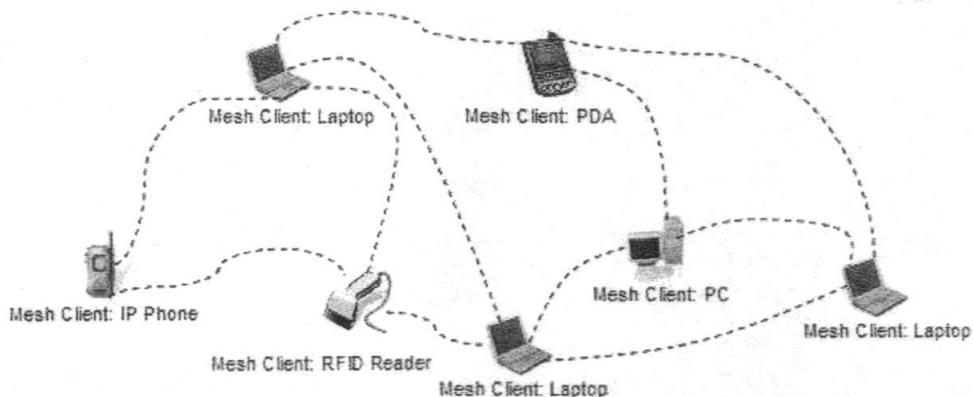
WMN Infrastructure Architecture:



WMN Hybrid Architecture:

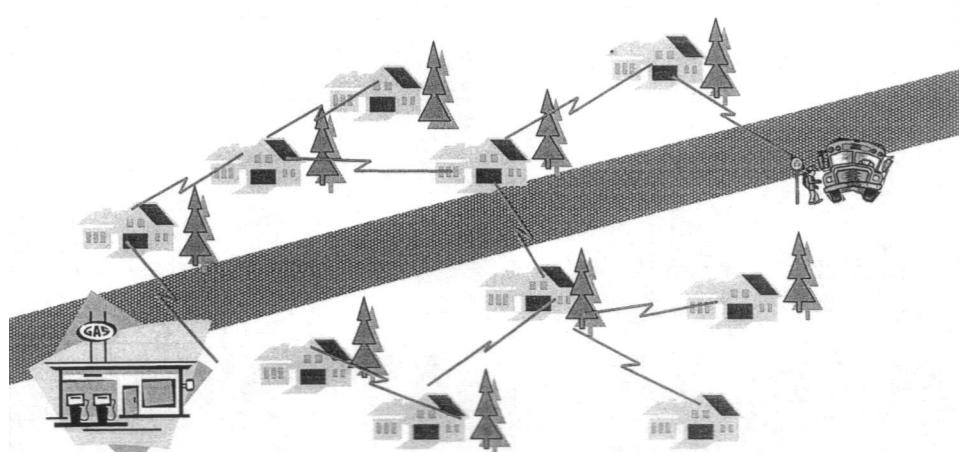


WMN Client Architecture:



## 11.12 Community Mesh Network

- Grass-roots wireless network for communities.
- Share Internet connections via gateway.
- Peer-to-peer neighborhood applications.
- Serious opportunities in developing countries, rural areas.

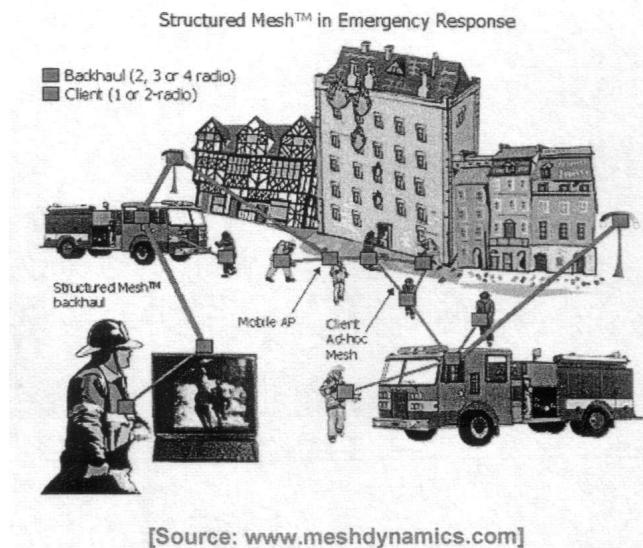


There are many service models for outdoor mesh:

- Private ISP (paid service)
- City/county/municipality efforts
- Grassroots community efforts
- May be shared infrastructure for multiple uses: Internet access, government, public

safety, law enforcement, education, community peer-to-peer.

### Public Safety:



### Metro-Scale Mesh Network:

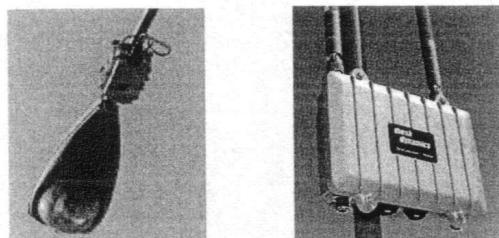


Photo Credit:  
Mesh Dynamics

[Source: <http://muniwireless.com>]

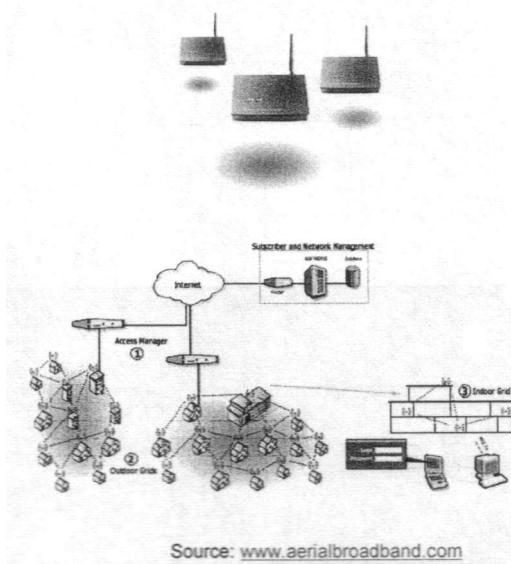
- ✓ Covers an entire metropolitan area

## 11.13 Companies

- Aerial Broadband
- BelAir Networks
- Firetide
- Intel
- Kyon
- LamTech (ex. Radiant)
- Locust World
- Mesh Dynamics
- Microsoft
- Motorola (ex. Mesh Networks)
- Nokia Rooftop
- Nortel Networks
- Packet Hop
- Ricochet Networks
- SkyPilot Networks
- Strix Systems
- Telabria
- Tropos Networks

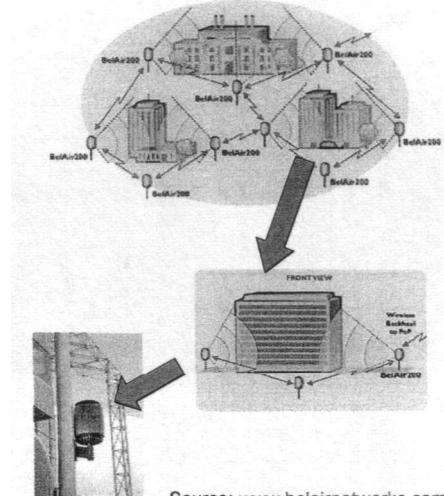
### Aerial Broadband

- Tiny start-up in RTP, NC, USA in 2002
- Closed its doors shortly after its start
- Application: broadband Internet access to apartment complexes
- Features: 802.11b-compatible product, Zero configuration, Layer 2 "routing".



### BelAir Networks

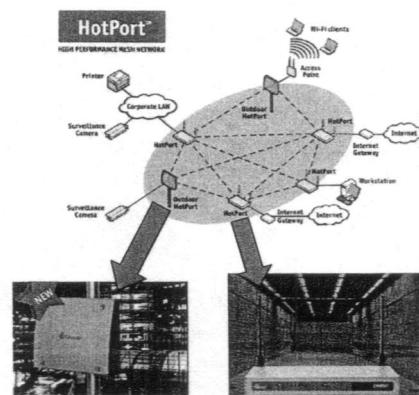
- Based in Ontario, Canada
- Application: 802.11b coverage of large zones
- Features: Three radios on each wireless router; dynamically mapped on, 8 fixed directional antennas, Dynamic Tx power and data rate control, Routing based on PHY feedback, congestion, latency, Load balancing features.



Source: [www.belairnetworks.com](http://www.belairnetworks.com)

### Firetide

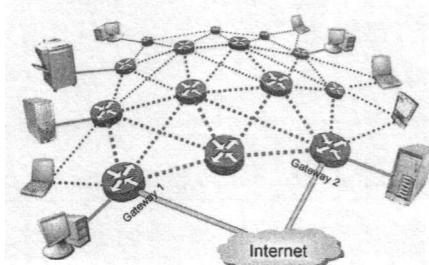
- Based in Hawaii and Silicon Valley, USA
- Application: Layer 2 connectivity (indoor and outdoor)
- Features: Proprietary routing protocol - 2.4GHz and 5GHz products, AES, WEP security, Variable Tx Power, Management software.



Source: [www.firetide.com](http://www.firetide.com)

### Intel

- Expressed interest in WMNs (since 2002).
- Research in: Low power - related with their wireless sensor networks activities at Intel Research Berkeley Lab. Traffic balancing.
- Together with Cisco active in 802.11s standardization process



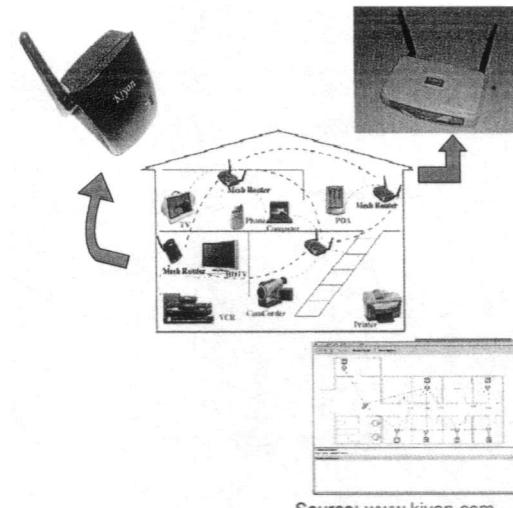
### Kyon

- Based in La Jolla, CA, USA

- Applications: extended 802.11 indoor

coverage

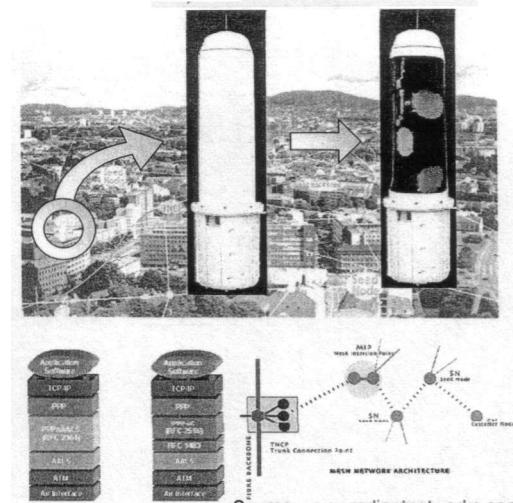
- Features: Products based on 802.11a/b/g, Custom routing (WARP), Management software.



Source: [www.kiyon.com](http://www.kiyon.com)

### LamTech (ex. Radiant Networks)

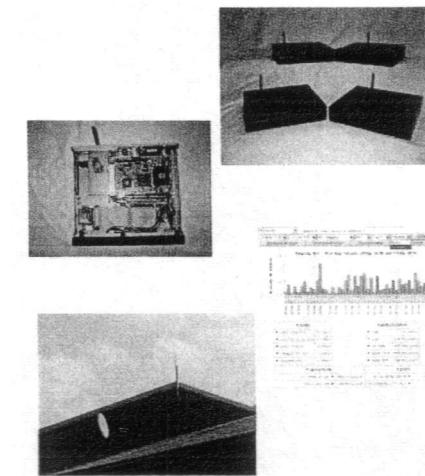
- UK-based company
- Purchased by LamTech in 2004
- Applications: broadband Internet access
- MESHWORK™: ATM switch in wireless router, 90 Mbps, Directional links, 4 mobile directional antennas, QoS - CBR & VBR-NR.



Source: [www.radiantnetworks.com](http://www.radiantnetworks.com)

### Locust World

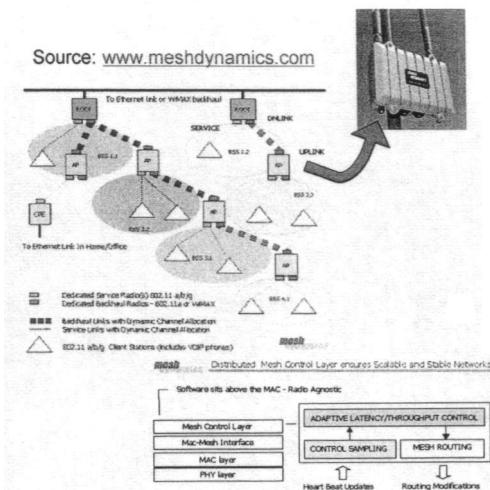
- Based in UK
- Application: community networks
- Features: Free, open source software, Off-the-shelf hardware + open source software, Monitoring software, Several deployments around the world.



Source: [www.locustworld.com](http://www.locustworld.com)

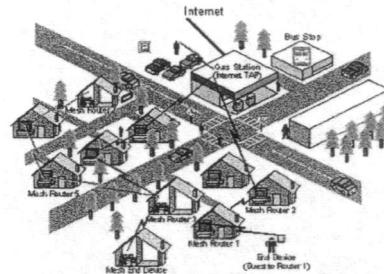
### Mesh Dynamics

- Based on Santa Clara, CA, USA
- Application: 802.11 coverage (indoor, outdoor, citiwide), VoIP, video
- Features: 802.11a/b/g compatible, Multiple radios options (1-4), Dynamic channel selection, Dynamic tree topology, Management software, Radio agnostic control layer.



### Microsoft

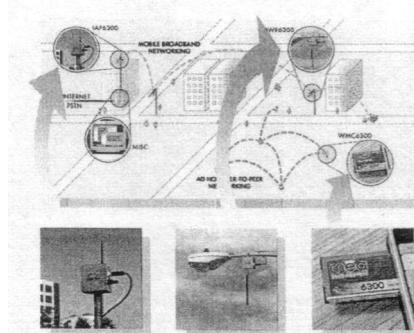
- Application: community networks
- Software: Routing and Link quality in "Mesh Connectivity Layer" (MCL).
- Routing based on DSR (named LQSR)
- Transparent to lower and higher layers
- Binaries for Windows XP available at [research.microsoft.com/mesh/](http://research.microsoft.com/mesh/)



Source: [research.microsoft.com/mesh/](http://research.microsoft.com/mesh/)

### Motorola - ex. MeshNetworks

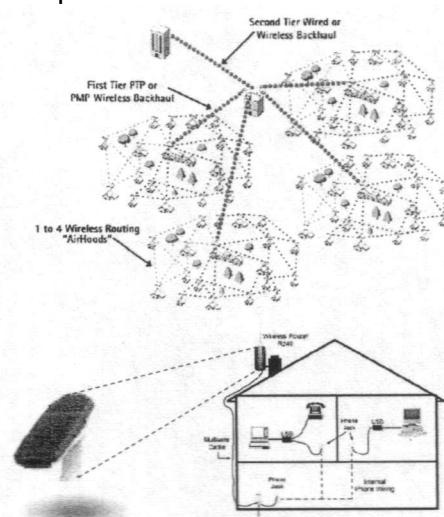
- Based in Orlando, FL, USA
- Acquired by Motorola in Nov. 2004
- Application: mobile broadband Internet access
- Features: Support for high speed mobile users, Proprietary routing protocol, Adaptive transmission protocol, Proprietary QDMA radio, Proprietary multichannel MAC, Proprietary geolocation feature, Support for voice applications, Local testbeds.



Source: [www.meshnetworks.com](http://www.meshnetworks.com)  
(now [www.motorola.com](http://www.motorola.com))

### Nokia Rooftop

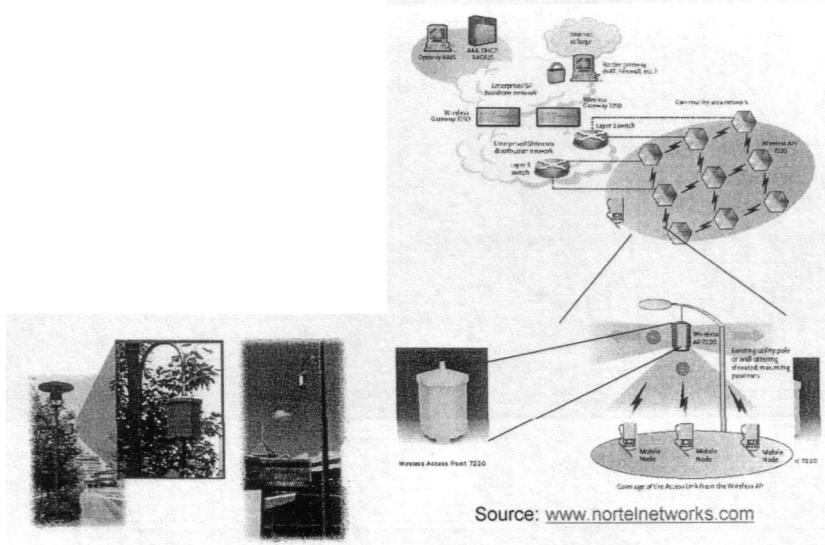
- Acquisition of Rooftop Comm.
- Discontinued in 2003
- Application: broadband Internet access
- Features: Proprietary radio, Proprietary multi-channel MAC, Variable TX Power, Management and monitoring tools.



Source: [www.rooftop.com](http://www.rooftop.com)

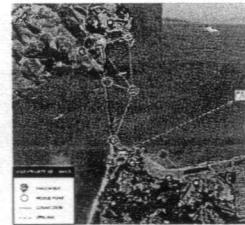
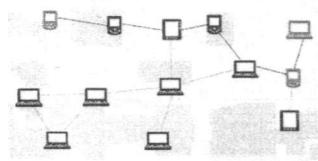
### Nortel Networks

- Applications: extended WLAN coverage
- Features: 802.11a backhaul, 802.11b for users, Management software.



### Packet Hop

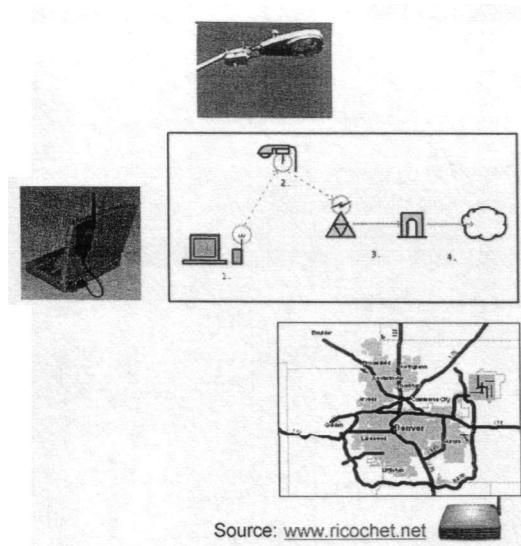
- Based in Belmont, CA, USA
- Application: emergency response
- Product: software for mesh networking
- Features: Works on 802.11a/b/g based hardware platforms, Security, Management software, Deployed testbed near Golden Gate Bridge in Feb. 2004.



Source: [www.packethop.com](http://www.packethop.com)

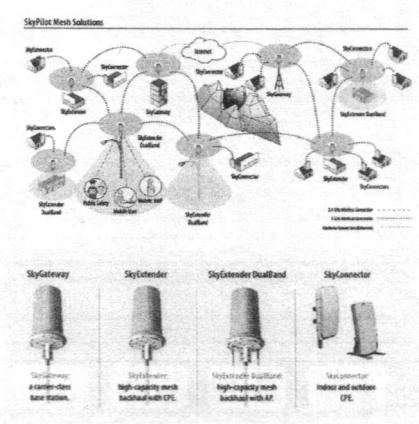
### Ricochet Networks

- Based in Denver, CO, USA
- Application: Internet access Features: Mobile user support - 2 hop architecture, 900 MHz user - pole top, 2.4GHz pole top – WAP, Sell both hardware and service in Denver and San Diego, Speed: "up to 4 times the dial-up speed".



### SkyPilot Networks

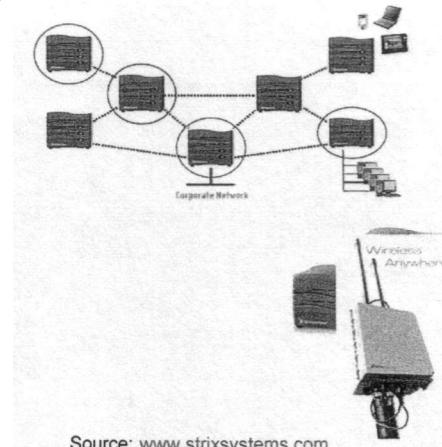
- Based in Santa Clara, CA, USA
- Application: broadband Internet access
- Features: High power radio + 8 directional antennas, Proprietary routing (based on link quality and hop count), Dynamic bandwidth scheduling (decides who transmits when), Management software, Dual band (2.4GHz for users, 5GHz for backhaul).



Source: [www.skypilot.com](http://www.skypilot.com)

### Strix Systems

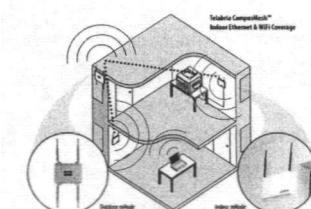
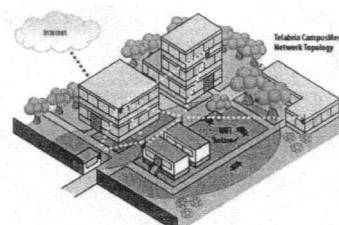
- Based in Calabasas, CA, USA
- Application: indoor and outdoor WLAN coverage, temporary networks
- Features: Compatible with 802.11a/b/g, Supports multiple (up to 6) radios, Management software, Soon to come testbeds



Source: [www.strixsystems.com](http://www.strixsystems.com)

### Telabria

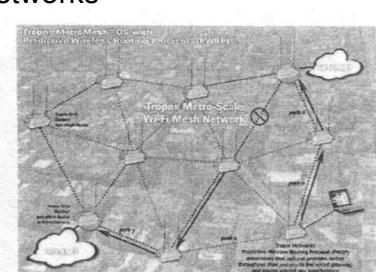
- Based in Kent, UK
- Application: WLAN coverage (campus/city);
- Features: 802.11 compatibility, Compatible indoor/outdoor products, Dual radio 802.11a/(b,g) (one for router-router, one for router-user traffic)



Source: [www.telabria.com](http://www.telabria.com)

### Tropos Networks

- Based in Sunnyvale, CA, USA
- Ex - FHP wireless
- Applications: citywide 802.11b/g coverage
- Features: Proprietary routing optimizing throughput, Support for client mobility, Security, Management software, Indoor/outdoor products, 150 customers installed their products



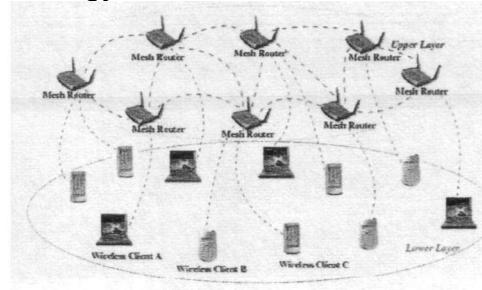
Source: [www.tropos.com](http://www.tropos.com)

### University Testbeds

- Georgia Tech – BWN-Mesh
- MIT – Roofnet
- Rutgers WinLab – Orbit
- SUNY Stonybrook – Hyacinth
- University of Utah – Emulab

### Georgia Institute of Technology BWN-Mesh

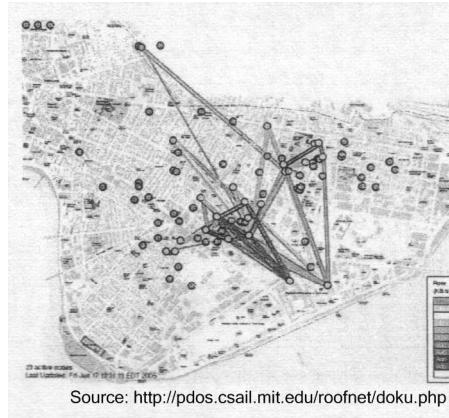
- 15 IEEE 802.11b/g nodes
- Flexible configuration and topology
- Can evaluate routing and transport protocols for WMNs.
- Integrated with the existing wireless sensor network testbed



Source: <http://users.ece.gatech.edu/~ismailhk/mesh/work.html>

### MIT Roofnet

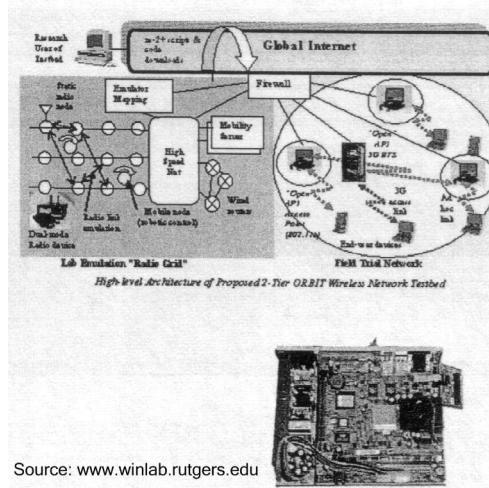
- Experimental testbed
- 40 nodes at the present
- Real users (volunteers)
- Focus on link layer measurements and routing protocols
- Open source software runs on Intersil Prism 2.5 or Atheros AR521X based hardware



Source: <http://pdos.csail.mit.edu/roofnet/doku.php>

### Rutgers Winlab ORBIT

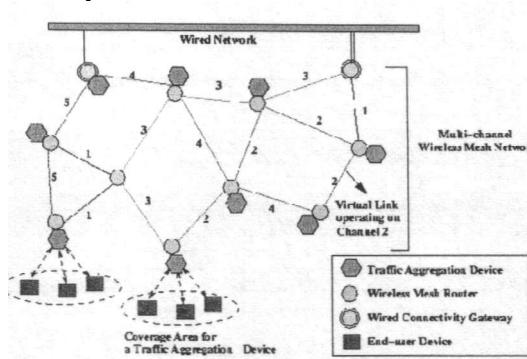
- Collaborative NSF project (Rutgers, Columbia, Princeton, Lucent Bell Labs, Thomson and IBM Research)
- Start date: September 2003
- Emulator/field trial wireless system
- 400 nodes radio grid supporting 802.11x
- Software downloaded for MAC, routing, etc.
- Outdoor field trial



Source: [www.winlab.rutgers.edu](http://www.winlab.rutgers.edu)

### SUNY Stonybrook Hyacinth

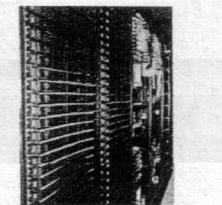
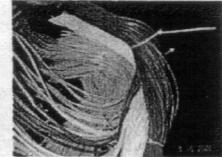
- Multichannel testbed based on stock PCs with two 802.11a NICs.
- Research focus on interface channel assignment and routing protocol.



Source: <http://www.ecsl.cs.sunysb.edu/multichannel/>

## University of Utah Emulab

- Three experimental environments:  
*simulated, emulated* - hundreds of PCs (168 PCs in racks), Several with wireless NICs (802.11 a/b/g) - and  
*wide-area network* - 50-60 nodes geographically distributed across approximately 30 sites.
- Smaller brothers at University of Kentucky and Georgia Tech.



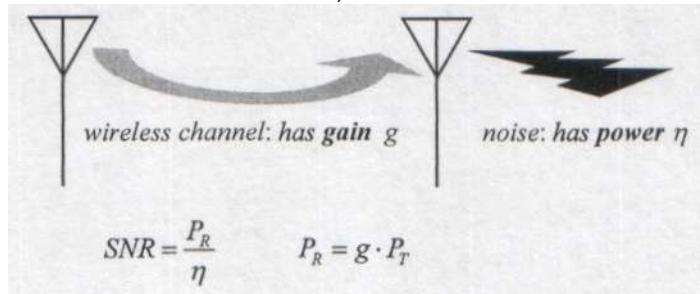
Source: [www.emulab.net](http://www.emulab.net)

# 12 Radio Propagation Basics

## 12.1 Radio Environment

In practical channels, transmitted signals in addition of being contaminated by additive white Gaussian noise, they undergo attenuation and delay. In wireless channels, because of multipath, the attenuation can vary with time and with the relative distance between the transmitter and the receiver. The BER performance of a digital communication system depends on the received SNR (Signal power to Noise power Ratio).

Transmitter: Uses Power  $P_T$ , Receiver: Receives Power  $P_R$ :



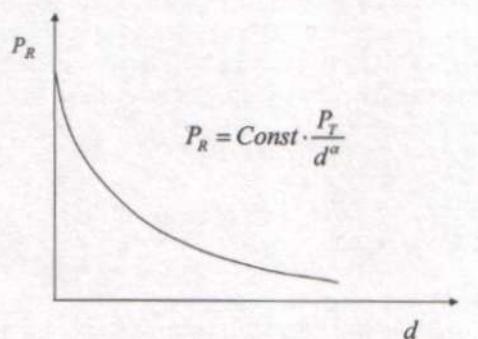
We are interested in the received signal power after the signal has passed through the channel, hence we need to know  $g$ . The link gain  $g$  is determined by three major phenomena: *Path Loss*, *Slow Fading (Shadowing)*, *Fast (Multipath) Fading*.

- *Path Loss*: as long as the signal propagates, its energy is spread on a wider area (the surface of a sphere in free space).
- *Slow Fading (Shadowing)*: additional signal intensity variations due to the presence of big obstacles (buildings, hills, trees).
- *Fast (Multipath) Fading*: reflections and scattering over smaller objects in relative motion with the receiver.

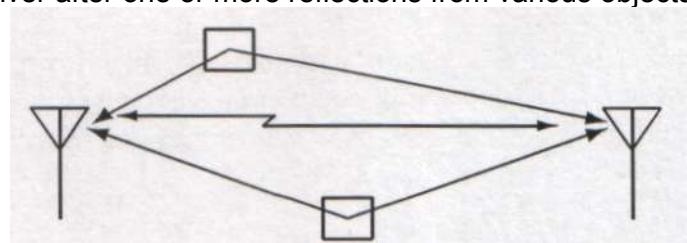
Assume that the signal power transmitted by the transmitter is  $P_T$ . The signal power  $P_R$  received at the receiver at a large distance  $d$  from the transmitter drops with distance following an inverse-power law:

$$P_R = Const \cdot \frac{P_T}{d^\alpha}$$

Where  $\alpha$  (*path loss exponent*) is a constant which is usually between 2 and 5 depending on the propagation environment (specifically,  $\alpha = 2$  in free space,  $\alpha \approx 4$  if ground reflection is present). The constant factor (*Const*) governing the above proportionality is a function of parameters not of direct concern to us, such as antenna parameters, transmit carrier frequency, etc.



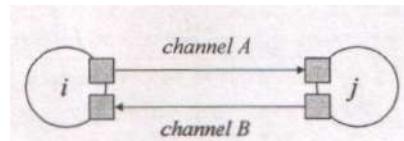
Radio waves are reflected and scattered by the objects they impinge on. Hence, unless a very narrow antenna beam is used, the receiver's antenna receives the transmitted signal along several paths. There is often a *direct*, or *line-of-sight*, path, and there are several paths along which the signal reaches the receiver after one or more reflections from various objects.



Fading is caused by different components of the transmitted signal reaching the receiver via paths of different lengths, and combining either constructively or destructively. Individual signals differ in phases as they travel different lengths; they also differ in amplitudes as they undergo different path losses. *Slow (Shadow) Fading* characterize relatively slower variations due to big obstacles. It comprises signal variations which stay constant for a time interval comparable with the propagation delay. *Fast (Multipath) Fading* is associated with movements (transmitter, receiver, other objects) which cause *Doppler shift* in frequency. It describes sudden signal strength variations which occur over a faster time-scale than the propagation delay. Both terms are generally modeled by proper statistical distributions. If reflections are present, the path loss exponent  $\alpha$  may need to be increased. Also, the power profile is no longer smooth but fluctuates. In fact, received power may be increased or decreased, according to the reflected rays being superimposed either constructively or destructively. Net effect is time varying signal strength.

## ***12.2 Transceiver Capability***

The *radio (NIC or transceiver)* capability is basically half-duplex, i.e., a radio can not listen on the same channel on which it is transmitting at the same time, or the transmitted power signal will jam any packet reception. Full-duplex operation is possible at each node, i.e., a node  $j$  can be receiving from or transmitting to a neighbour  $i$  on channel A, while transmitting to or receiving from neighbour  $i$  on channel B, where  $A \neq B$ .

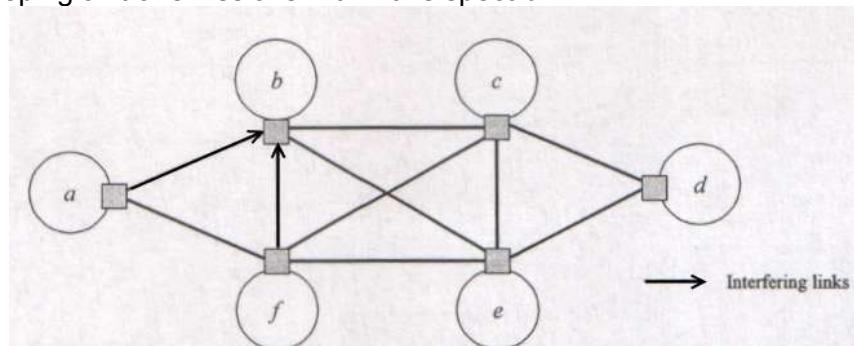


According to this approach, full-duplex capability is obtained at the price of additional resource. Since the number of radios (i.e., NICs or channels) per node is limited, the allocation of two radios for establishing a full-duplex communication between two neighbours may decrease the overall network connectivity, which in general is an undesirable effect.

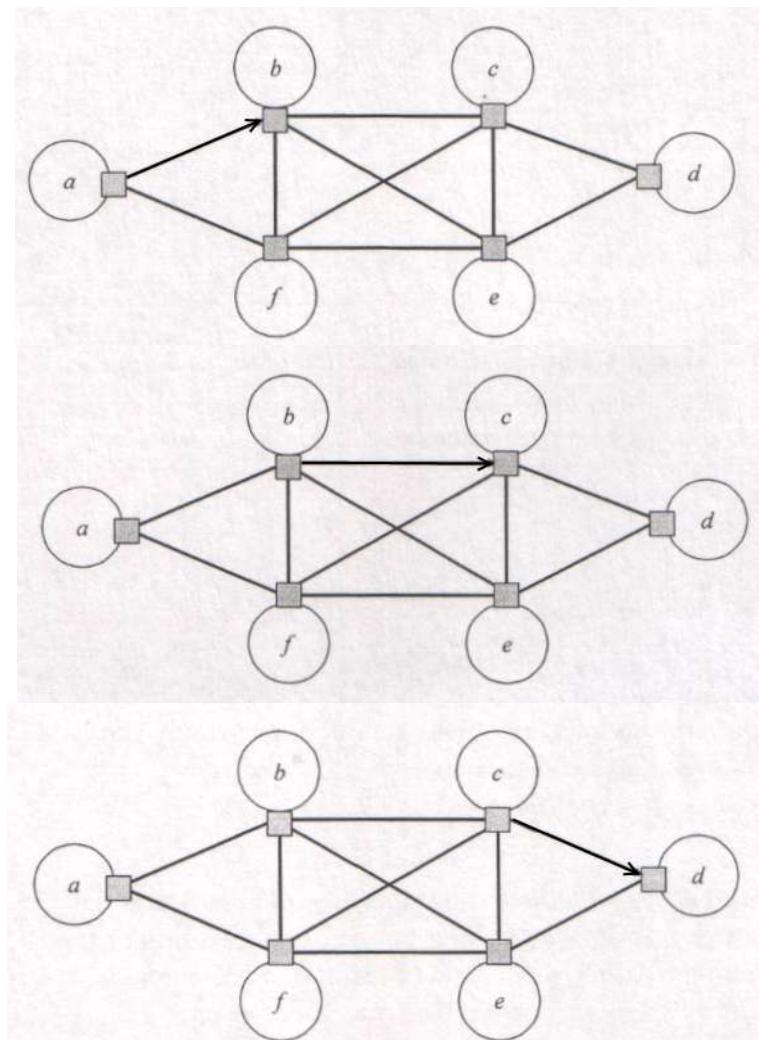
**REMARK:** In the multiple radios (i.e., channels) case any radio is still utilized in a half-duplex fashion, i.e., no simultaneous transmission and reception is still possible on the same channel. For these reasons, we impose that the activation of links should satisfy the constraint of not activating more than one operation (i.e., either a transmission or a reception), for each radio.

### **12.3 Interference Models**

A multihop wireless network would comprise several transmitters and receivers sharing an assigned portion of the radio spectrum. We would like to carry as much traffic in this network as possible while keeping all transmissions within this spectrum.



An inefficient way would be to permit exactly one transmitter and receiver pair communicate at one time. Obviously the attempt should be to let several transmitter and receiver pairs communicate simultaneously.



We assume that at most one signal can be decoded by the receiver. Any other transmission the receiver is able to listen to can only be regarded as *interference*. The presence of interference at the receiver does not necessarily mean that the packet can not be correctly decoded. It will be shown that the interference model comes into play at this point. To represent interference two models are widely used:

- The protocol interference model (*protocol model*);
- The physical interference model (*physical model*).

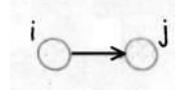
In the *protocol model*, any superposition of signals with enough power (we see later on the exact meaning of this statement) at the receiver will result in a collision, i.e., no packet can be received correctly. In the *physical model*, when there is a superposition of signals at the receiver the strongest received signal can still be successfully decoded. Regardless of the interference model, the maximum number of possible simultaneous successful receptions is one. A similar situation happens for the transmitter. Multiple simultaneous (i.e. time overlapping) transmissions of packets from the same transmitter are not possible. Conclusions:

- at the transmitter simultaneous packets transmissions are not allowed,
- at the receiver simultaneous packets receptions are not allowed.

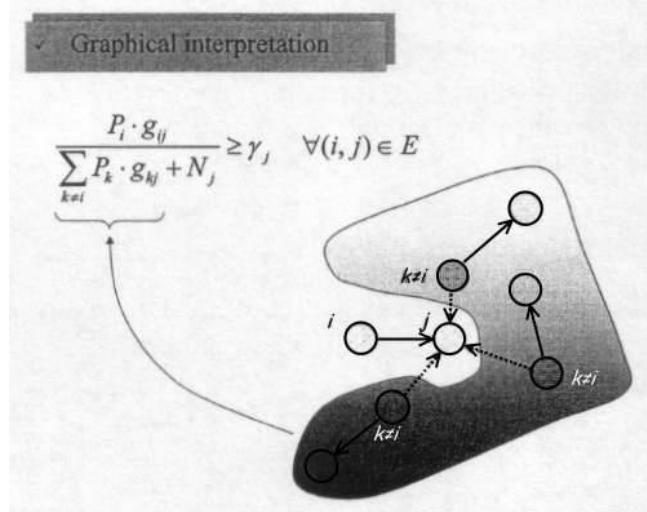
There are several variations of the protocol and physical interference models. These variations take into consideration transmission aspects such as directional antennas, capture effect (when modelled with a threshold) and many others. However, in the following lectures, we will only focus on *path loss model*. An overview on interference models can be found in: A. Iyer, C. Rosenberg, and A. Karnik, “*What is the right model for wireless channel interference?*” in Proc. Qshine, no. 2, Waterloo, ON, Canada, 2006, invited paper.

## 12.4 Physical Interference Model

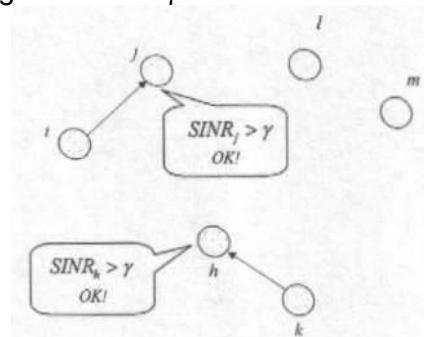
The basic assumption of this model is that a packet is correctly received by a node with probability 1 if the *Signal to Interference-and-Noise Ratio* (*SINR*) is above a given threshold. A way to formalize this:

$$\frac{P_i \cdot g_{ij}}{\sum_{k \neq i} P_k \cdot g_{kj} + N_j} \geq \gamma_j \quad \forall (i, j) \in E$$


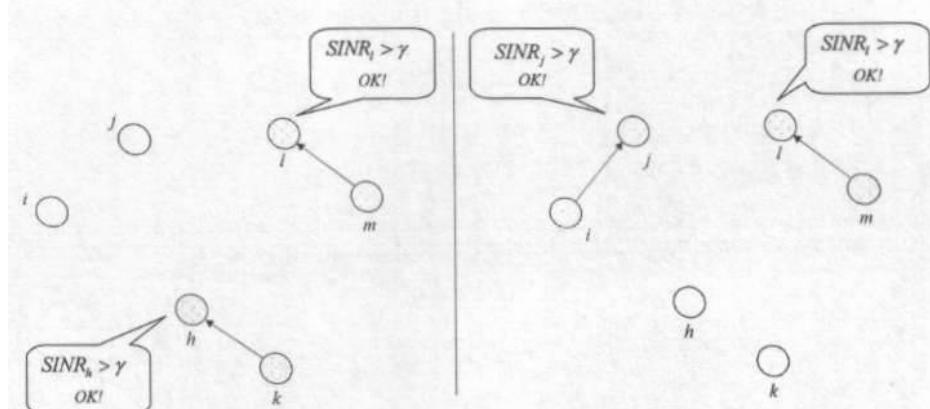
Where  $(i, j)$  is the link of interest, the index  $k$  in the lower sum denotes a possible interferer ( $i$  is in fact excluded from the sum, as it is the useful transmitter),  $P_i$  is the power emitted by node  $i$ ,  $g_{ij}$  is the path gain from  $i$  to  $j$  and  $N_j$  is the noise at the receiver node  $j$ . The value  $\gamma_i$ , which defines the SINR threshold, can be in general a different value for every node  $j$ .



The idea of treating interference within the same channel (i.e., co-channel or intra-channel interference) as noise, along with the geographical attenuation of power, can be used to deliberately let several transmitter and receiver pairs simultaneously use the same radio spectrum as long as they are sufficiently separated in space. Called *spatial reuse*, this is an important way of optimizing the utilization of a radio spectrum. At each receiver there would be some target value of SINR,  $\gamma$ , and hence there may be a need for power control of the transmitters to achieve the desired SINR targets. The physical interference model is a cumulative interference approach because at each receiver the whole interference produced by all the other interfering nodes is taken into account. According to this model, even though a set of links may not interfere singularly with any given link, they could still collectively cause a collision. In order to clarify this concept, we consider a transmission in which link  $(i,j)$  and link  $(k,h)$  can be activated simultaneously, because the SINR at node  $h$  and node  $j$  is greater than  $\gamma$ .

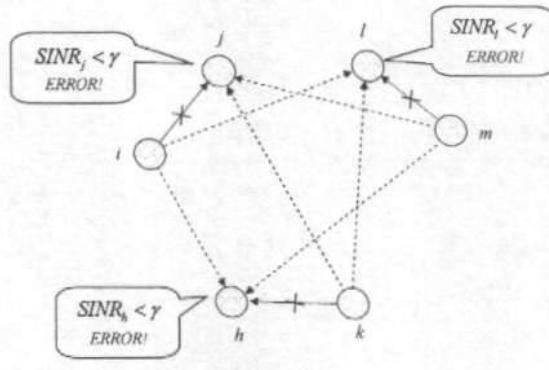


Similarly, as shown in both figures below, the link  $(m,l)$  can be activated simultaneously with link  $(k,h)$  and  $(i,j)$  respectively.



However, if we try to activate all three links together we experiment the destruction of all

transmissions, as shown in the following figure.

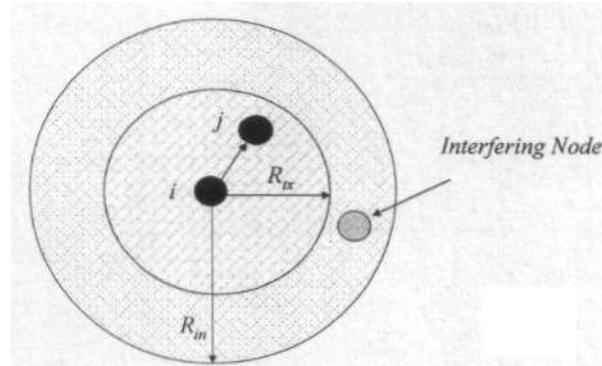


## 12.5 Protocol Interference Model

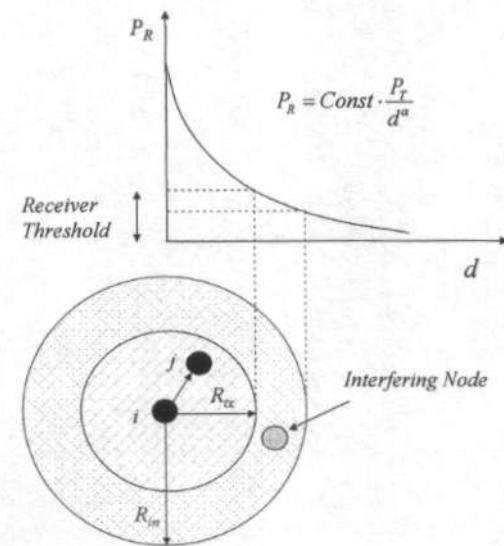
The protocol interference model (or protocol model for short), in its original version, was designed to describe interference in an IEEE 802.11 environment. The protocol model is an approximation of the physical model, based on geometrical assumptions only. The protocol model defines co-channel interference. The main advantage of an interference description through the protocol model is its conceptual simplicity, and the ease of mathematically formalizing the resulting interference conditions. For the sake of simplicity, and without loss of generality, we suppose the nodes disposed on a two-dimensional plane. Consider the transmission from node i to node j. The protocol model is specified by two areas around the transmitter i:

- the *transmission area* with radius  $R_{in}$  called transmission radius/range,
- the *interference area* with radius  $R_{tx}$  called interference radius/range.

The *interference range* is always greater than or equal to the transmission range (i.e.  $R_{in} \geq R_{tx}$ ). The *transmission range* delimits the area into which the transmission by node i is correctly decoded by a node j positioned in that area provided no other node located within the interference area transmits at the same time.

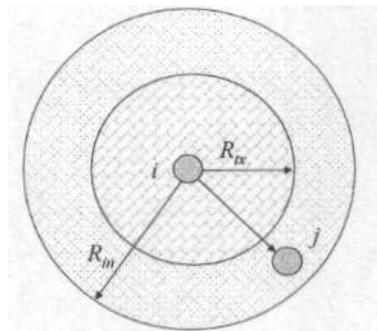


*Receive Threshold* is defined as the minimum received power needed for successful reception of packets.

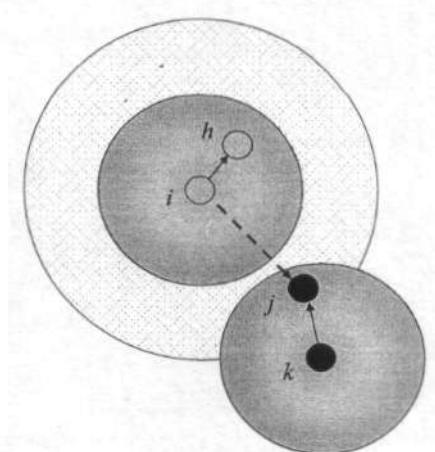


In the region of space outside the transmission range but inside the interfering range, receiver j cannot decode the transmission by node i because the channel gain attenuates too much the

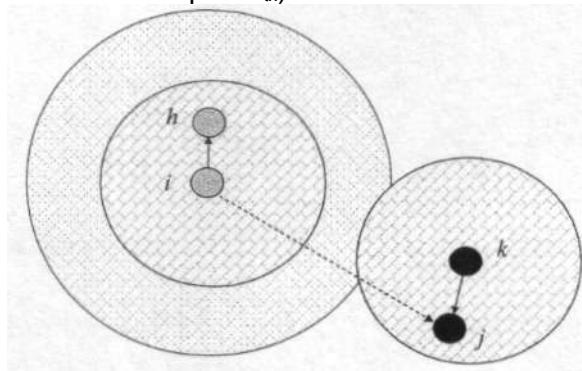
transmit power (we recall that the attenuation caused by the channel gain increases with the distance from the transmitter).



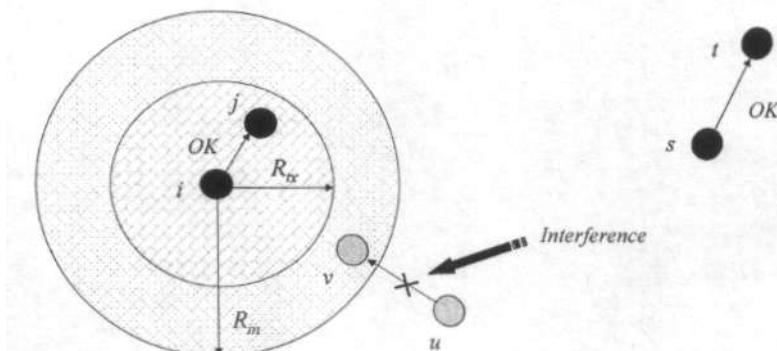
However the transmission power of  $i$  (transmitting to  $h$ ) is strong enough to interfere with an ongoing potential successful transmission from node  $k$  to node  $j$  where the distance between  $k$  and  $j$  is less than or equal  $R_{tx}$ .



Outside the interference range the transmission power of  $i$  is considered to be too low to cause interference at node  $j$  of an ongoing successful transmission from node  $k$  to node  $j$  (i.e. the distance between  $k$  and  $j$  is less than or equal  $R_{tx}$ ).



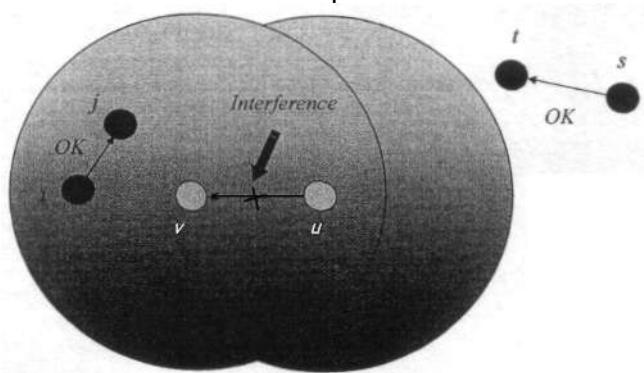
Example 1:



The link  $(u,v)$  can not be activated, because the receiver  $v$  is in the interference range of  $i$ , instead the activation of link  $(s,t)$  does not suffer from interference by link  $(i,j)$ . The destruction/corruption of a message due to interference can happen at the receiver and not at the transmitter. Thus, the problem of identifying interference at a receiver can be more clearly described considering the interference area centred on the receiver. In fact, in such a case, the interference range represents the area of “*interference sensibility*” of the receiving node, meaning

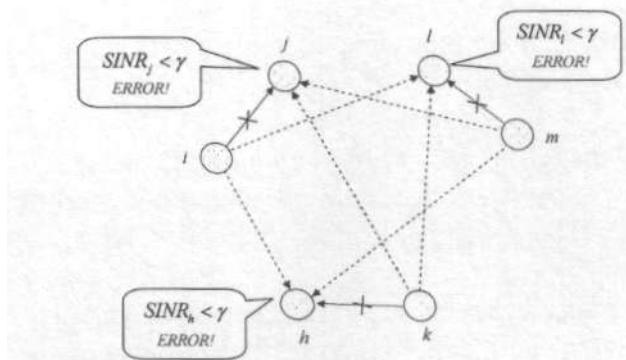
that the transmission at that receiver can be destroyed by the one generated by any other interfering node present in the interference range.

Example 2:



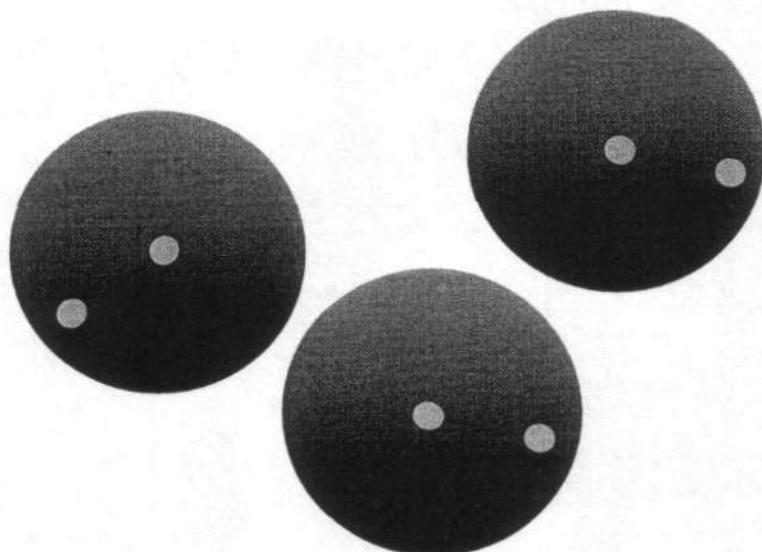
The reception at node v is not successful because node i transmits and it is into the interference range of v. At the same time j is not into the interference range of u. Once we have explained how protocol model works, we can underline the most important difference between the protocol model and the physical model. As said above, the physical interference model considers the interference produced by all the links cumulatively. Instead, the protocol model leads to binary conflict relationships among links. More precisely, each pair of links is considered at a time, according to the geometrical considerations said above. For example, in the following example we recall the network previously analyzed. Suppose that each pair of links belonging to the set  $\{(i, j), (u, v), (s, t)\}$  can successfully transmit both according to the physical model and according to the protocol model. The difference between those two interference models can be shown in case all the three links of the above set are simultaneously activated. In such a case, the protocol model considers each pair of nodes at time, and then, supposed that each pair of nodes can successfully transmits, the three links can surely be simultaneously activated according to the protocol model. On the contrary, according to the physical model, the three links can not be simultaneously activated, due to the cumulative interference generated by each link.

Physical Interference Model:



Physical Interference Model

Protocol Interference Model:



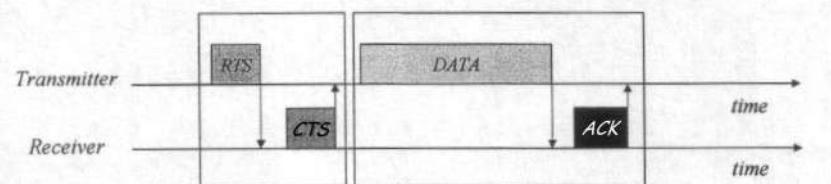
Two protocol models have already been specified and deeply studied in the literature. They are actually used to model interference in two widely known standard technologies, i.e. IEEE 802.11 and IEEE 802.16. For this reason we refer to these two protocol models as *11protocol model* and *16protocol model*.

## 12.6 11Protocol Model

**NOTATION:** Let's indicate with  $d(i,j)$  the Euclidean distance between node  $i$  and  $j$ .

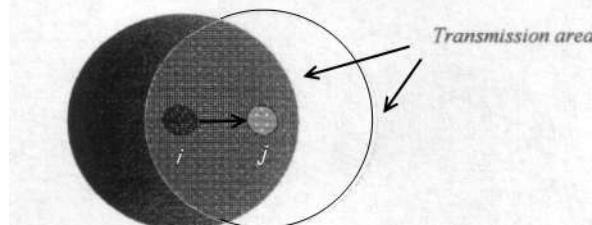
$$d(i,j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

where  $(x_i, y_i)$  and  $(x_j, y_j)$  are the coordinates of nodes  $i$  and  $j$ . The *11protocol model* requires both the transmitter and the receiver to be free of interference. The reason for requiring the absence of interferers in both receiver's and transmitter's interference range of both interfering transmitters and receivers is in that the IEEE 802.11 standard forces the receiver to acknowledge RTS and DATA packets with CTS and ACK, respectively. In other words, due to the four way handshake, a receiver is also a transmitter, therefore it can cause disturbance to others. Similarly, the transmitter needs to perform reception (i.e., to receive CTS and ACK), for which it has to be collision-free.

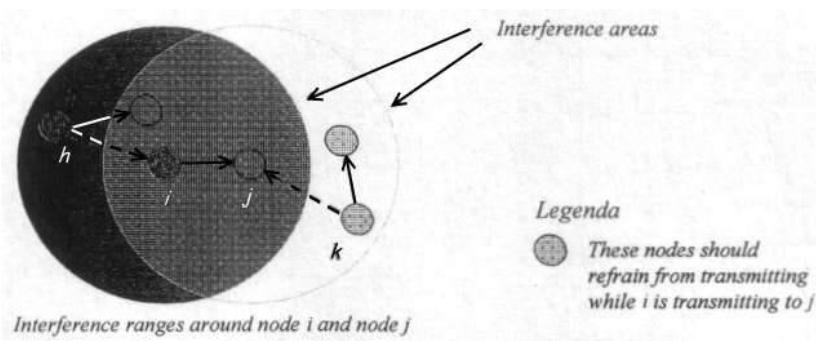


A transmission from node  $i$  to node  $j$  is successful provided if:

- Nodes  $i$  and  $j$  are located in the transmission range of  $j$  or  $i$  respectively, i.e.  $d(i,j) \leq R_{tx}$ .



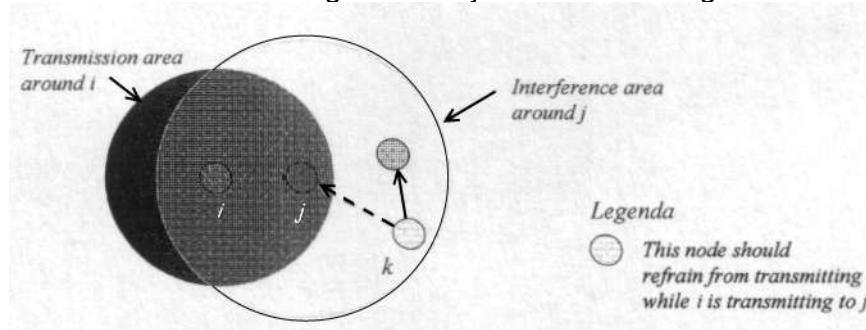
- No other node (e.g.  $k$ ), located in the interfering range around  $j$ , transmits at the same time.
- No other node (e.g.  $h$ ), located within the interfering range around  $i$ , transmits at the same time.



## 12.7 16Protocol Model

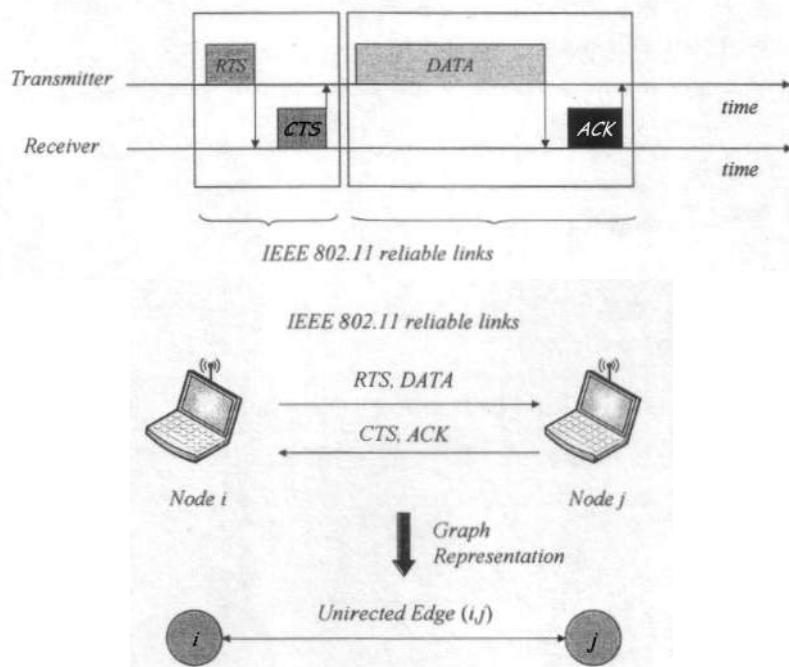
In the 16protocol model a transmission from node  $i$  to node  $j$  is successful if both of the following conditions are satisfied:

1. Nodes  $j$  is located in the transmission range of node  $i$ , i.e.  $d(i,j) \leq R_{tx}$ .
2. Any node  $k$  in the interference range of node  $j$  is not transmitting.



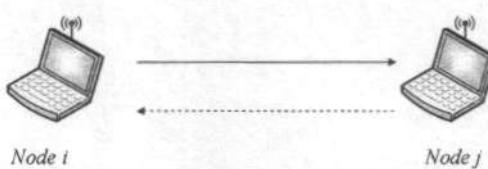
## 12.8 WMN as a Graph

We consider a WMN with stationary wireless routers. We assume further that each wireless router is equipped with one NIC. Given a WMN, we first derive a connectivity graph  $G = (N, E)$  where vertices of  $N$  correspond to the set of WMN wireless routers and the edges of  $E$  corresponds to the set of wireless links between WMN wireless routers. There is a direct link  $(i, j)$  from node  $i$  to node  $j$  if  $d(i, j) \leq R_{tx}$  and  $i \neq j$ , i.e. nodes  $i$  and  $j$  are distinct and  $i$  is in the transmission range of  $j$  (also implying  $j$  is in the transmission range of  $i$ ).  $G$  can be an undirected graph or a directed graph, it depends on the WMN technology it models.  $G$  is an *undirected graph* when it models the IEEE 802.11 technology since the IEEE 802.11 standard is supposed to work on entirely reliable links only and therefore edges in  $E$  needs to be bidirectional.

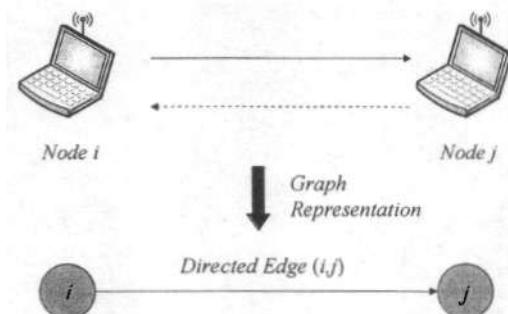


$G$  is a *directed graph* when it models the IEEE 802.16 technology since the IEEE 802.16

standard is not supposed to work on reliable links. Due to environmental limitations and also different power levels, it is even possible that two nodes  $i$  and  $j$  belonging to  $N$  are linked only one-way, i.e.,  $(i,j) \in E$  but  $(j,i) \notin E$ .



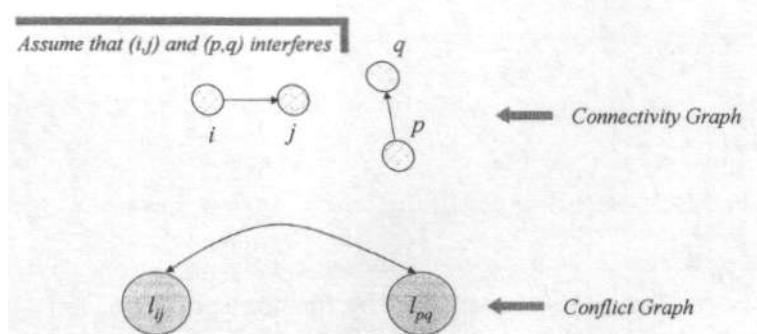
The communication link where a sender node  $i \in N$  transmits to a receiver  $j \in N$  is represented by an element  $e \in E$  equal to the ordered pair  $(i,j)$ . The inclusion of this link in  $E$  actually happens only if node  $j$  can receive a transmission from node  $i$  in the absence of any other interference source. Under the above assumptions, the connectivity between mesh routers can be modeled using a directed graph referred to as connectivity graph.



To quantify the capacity of the link we make use of variables  $r_{ij}$ , called *link rates* and collected into a matrix  $R = (r_{ij})$ . Rate  $r_{ij}$  can be regarded as the amount of bits which can be transmitted over the link represented by edge  $(i,j)$  on a given time unit. When required by physical specifications, we will also consider a parameter  $g_{ij}$  corresponding to the wireless link gain over  $(i,j)$ . A matrix  $G = (g_{ij})$  can be introduced collecting the  $g_{ij}$  variables for all edges.

## 12.9 Conflict Graph

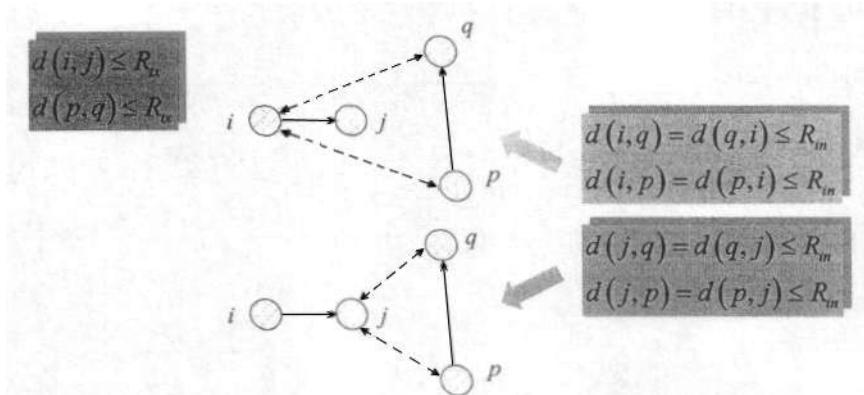
Because of the broadcast nature of the wireless medium, the success of a transmission is greatly influenced by the amount of multiple access interference. To incorporate the wireless interference into several problem formulations related to WMNs (e.g. optimal throughput computation, channel assignment) the concept of conflict graph was introduced. *NOTE:* In the following we use the terms *node* and *link* in reference to the connectivity graph and reserve the terms *vertex* and *edge* for the conflict graph. Corresponding to every link  $(i,j)$  in the connectivity graph between nodes  $i$  and  $j$ , the conflict graph ( $G_{ij}$ ) contains a vertex denoted by  $I_{ij}$ . We place an edge between two nodes (say,  $I_{ij}$  and  $I_{pq}$ ) in the conflict graph if the corresponding links  $(i,j)$  and  $(p,q)$  in the connectivity graph interfere; the weight of the edge indicates the extent of interference between those links. An edge is placed between two vertices in the conflict graph if the corresponding links in the connectivity graph interfere.



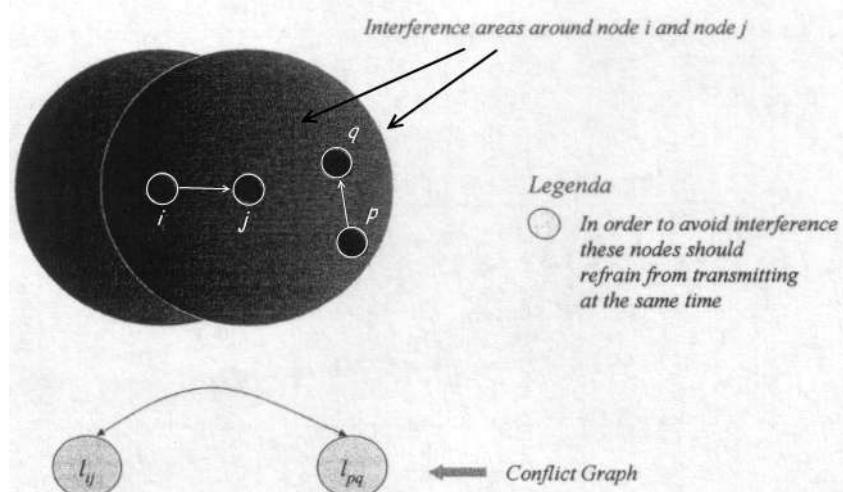
Assuming that all nodes in the WMN have the same interference range, in the 16protocol model the transmission from  $i$  to  $j$  is successful only if no other node located within distance  $R_{in}$  (i.e., interference range) from  $j$  transmits at the same time as  $i$ . Additionally, in the case of IEEE 802.11, if the *RTS/CTS* (*Request to Send/Clear to Send*) mode is used, then also no other node within distance  $R_{in}$ , from  $i$  should be transmitting at the same time (11protocol model). Stated differently the 11protocol model requires both the transmitter and the receiver to be free of interference. Therefore the *existence and extent* of interference between a pair of links in the network is

determined by an interference model. In the base case, all edges of the conflict graph have unit weight. However, the conflict graph can be further refined in such a way that the weight value can be different from one. In the following description we assume that all nodes in the WMN have the same transmission and interference ranges.

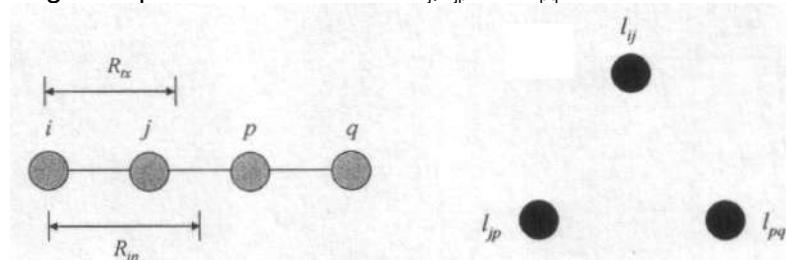
## 12.10 Conflict Graph/11Protocol Model



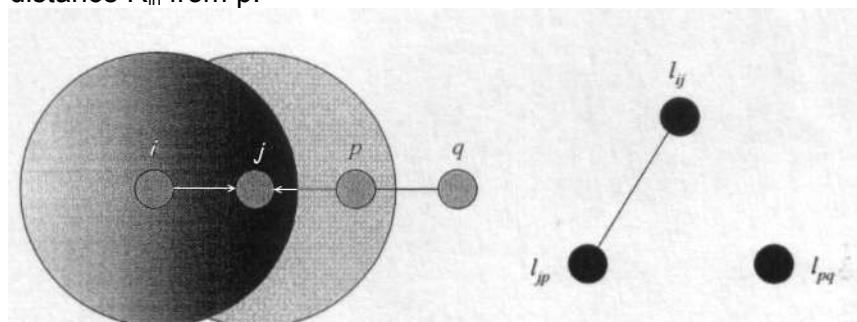
Nodes  $i, j$  and  $p, q$  positioned in their transmission range interfere if any of the sentences on the right of the previous image is true.



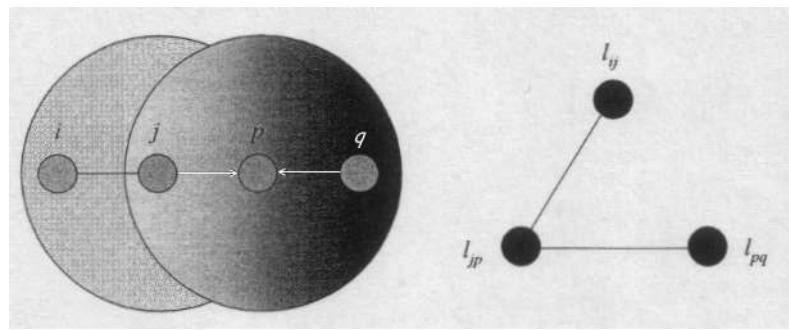
*Example #1:* The left figure represents the connectivity graph of a linear WMN with nodes  $i, j, p$ , and  $q$  while the right figure represents the vertices  $l_{ij}, l_{jp}$  and  $l_{pq}$  of the related conflict graph.



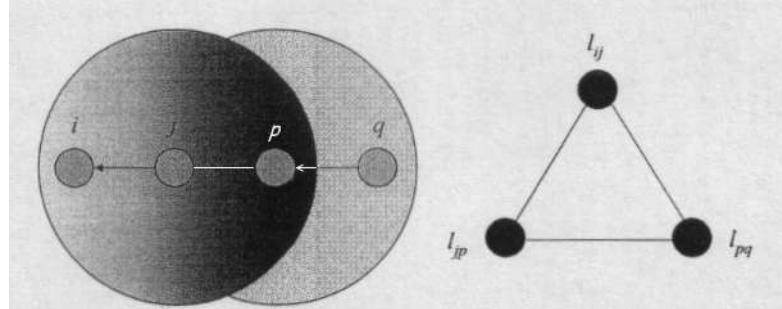
The conflict graph associated to the connectivity graph contains an edge between  $l_{ij}$  and  $l_{jp}$  since  $j$  is located within distance  $R_{in}$  from  $p$ .



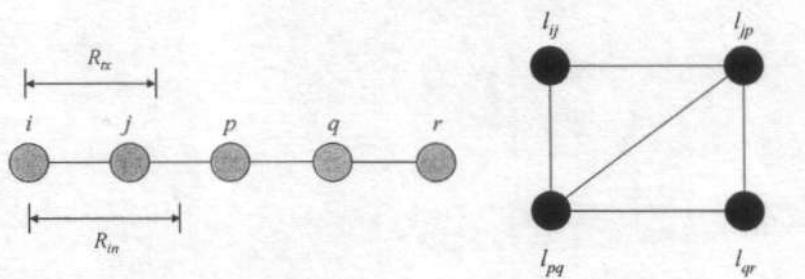
Furthermore the conflict graph contains an edge between  $l_{jp}$  and  $l_{pq}$  since  $p$  is located within distance  $R_{in}$  from  $j$  and  $q$ .



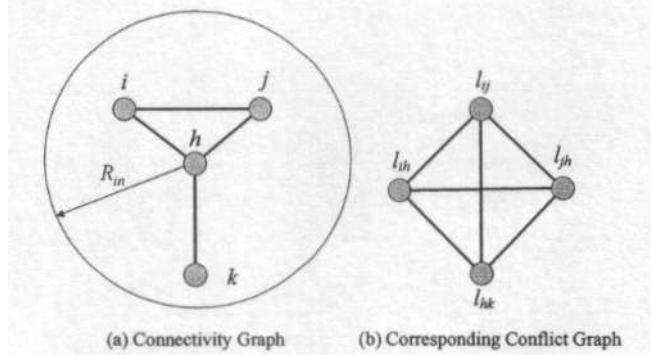
Finally the conflict graph contains an edge between  $l_{ij}$  and  $l_{pq}$  since p is located within the same interference range centred at j and p is at distance  $R_{tx}$  (i.e. less than  $R_{in}$ ) from q. If j transmits to i, then p can hear this transmission and thus if q transmits to p there is an interference between links (i,j) and (p,q).



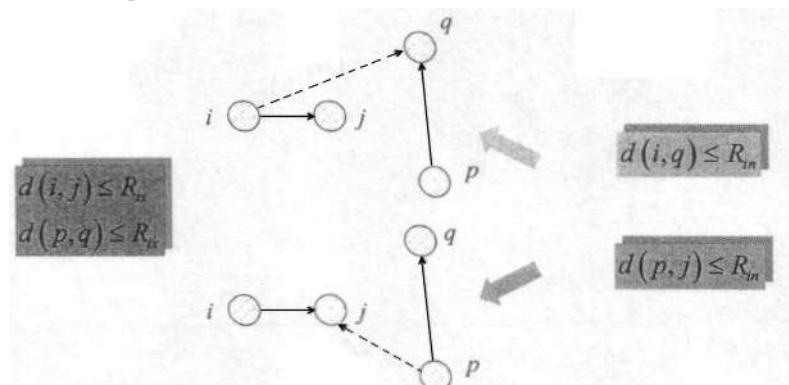
*Example #2:* In this example we add node r to the chain.



*Example #3:* In this example the interference range centered at node h includes all the other nodes (i.e. i, j, and k) and therefore the conflict graph for the 11protocol model is a *clique*.

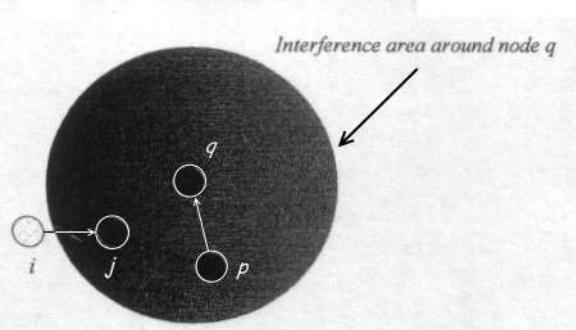


## 12.11 Conflict Graph/16Protocol Model

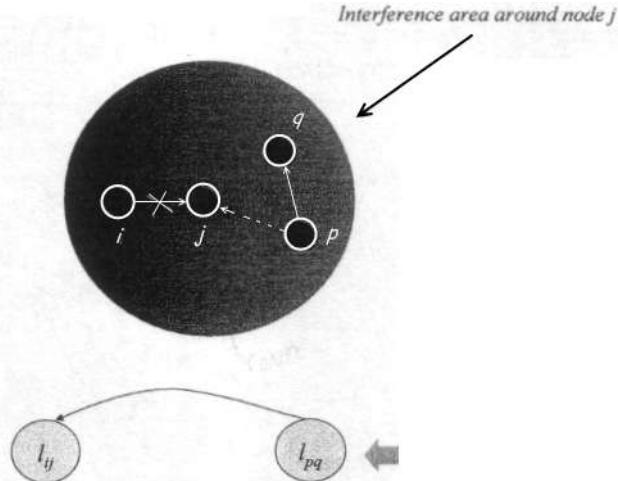


Nodes i,j and p,q positioned in their transmission area interfere if any of the sentences on the right of the previous image is true. Transmission from p to q is successful since i is outside the

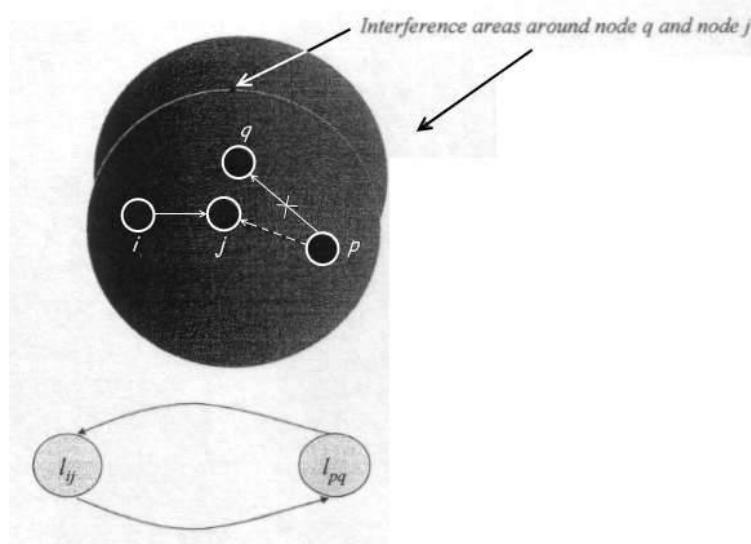
interference range of q.



There is an interference at j caused by a transmission from p to q which results in a corrupted transmission from i to j. In the conflict graph, the activation of link (p,q) generates interference at link (i,j).



Assume that i and p are located within the interference area of q and j respectively. Interferences at j and q caused by transmissions from p to q and i to j respectively. This result in corrupted transmissions from i to j and from p to q. In the conflict graph, activation of link (p,q) generates interference at link (i,j) which in turn causes interference at link (p,q).



## 12.12 Conflict Graph/Physical Model

Unlike in the protocol model, conflicts in the physical model are not binary. Rather, the interference gradually increases as more neighboring nodes transmit, and becomes intolerable when the noise level reaches a threshold. This gradual increase in interference suggests that the conflict graph should be a *weighted graph*, where the weight of a directed edge from vertices  $l_{pq}$  to vertices  $l_{ij}$ , (denoted by  $w_{ij}^{pq}$ ) indicates what fraction of the maximum permissible noise at the receiving node j (for link  $l_{ij}$  to still be operational) is contributed by activity on link  $l_{pq}$  (i.e. node p's transmission to node q). The edges of the conflict graph are directed and in general  $w_{ij}^{pq} \neq w_{pq}^{ij}$ . For further details on the physical model, we refer the reader to [K. Jain, J. Padhye, V.N. Padmanabhan, L. Qiu, "Impact of Interference on Multi-hop Wireless Network Performance" Proc

IEEE/ACM MobiCom, pp 66-80], 2003.

### 12.13 01Protocol Model

According to this protocol model at most one edge can be activated at any given time throughout the whole network. In other words, either exactly one edge is active, or no edge is active at all. Due to this property, we refer to this version as the *01protocol model*. Even though it is quite oversimplified, it can be useful as a theoretical term of comparison. In fact, the 01protocol model is clearly the worst possible case of interference condition, where space diversity can not be exploited to obtain transmission parallelism. For example, very dense topologies (i.e. if the connectivity graph is a clique) fall down to the 01protocol model. However this is due to the half-duplex constraint rather than to the interference constraints. We emphasize that these two constraints should not be confused.

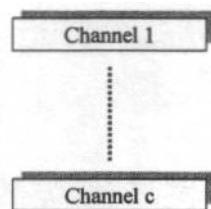
# 13 Channel Assignment and Link Scheduling Strategies

## 13.1 IEEE 802.11 Channels

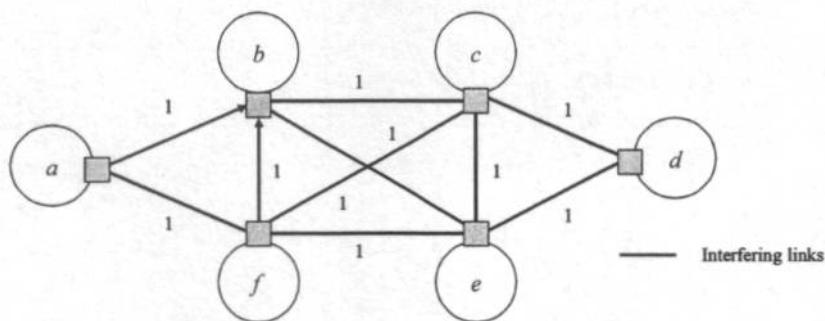
Deploying mesh networks as a simple multihop extension of 802.11 wireless LANs can lead to significant performance problems due to inefficient utilization of available spectrum, in turn degrading end-user throughput. Access points in 802.11 LANs typically have a single half-duplex radio and the MAC protocol in the 802.11 standard is designed to use a single channel. However, there are 3 non-overlapping channels with IEEE 802.11b standards in 2.4 GHz band, and 12 non-overlapping channels with original IEEE 802.11a standard in 5 GHz band.

## 13.2 Radios and Channels

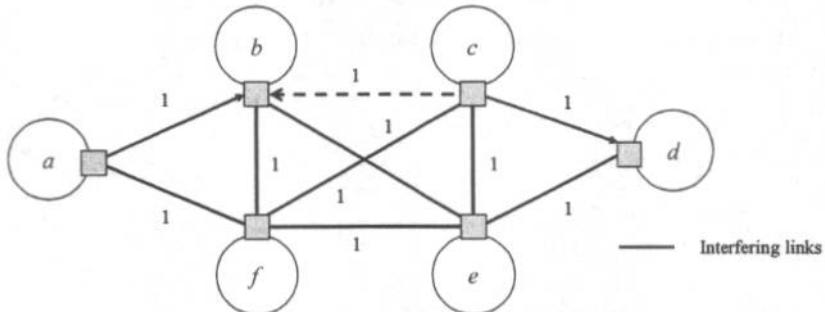
We use the term *channel* to refer to a part of the frequency spectrum with some specified bandwidth. There are  $c$  channels, and we assume that every node is equipped with  $m$  radios (interfaces, NICs), where  $1 \leq m \leq c$ .



We assume that a radio is capable of transmitting or receiving data on any one channel at a given time. We use the notation  $(m, c)$ -network to refer to a network with  $m$  radios per node, and  $c$  channels.



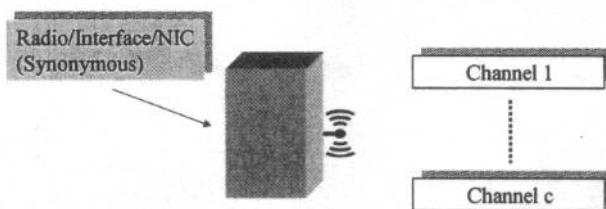
The use of one half-duplex IEEE 802.11 radio per node (#1 in the figure) for multihop wireless communication in a WMN forces all nodes to use the same channel (*single-radio single-channel* mode/architecture) to maintain connectivity. As a result, this architecture poorly utilizes available spectrum and suffers from the well-known capacity scaling problem with single-channel multihop wireless networks, arising from the need to share the same channel among neighboring transmissions even those at neighboring hops of a multihop path.



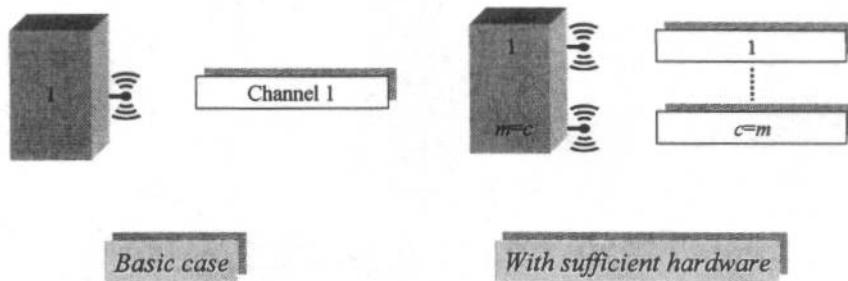
Using multiple 802.11 channels with a single radio per node is possible, but this approach (commonly termed *single-radio multi-channels*) has certain drawbacks. A common theme across these single-radio solutions is for each node to dynamically switch between channels, while coordinating with neighboring nodes to ensure communication over a common channel for some period. Such coordination, however, requires tight time synchronization among nodes.

A radio with multiple channels can only use one channel at a time. Switching between channels

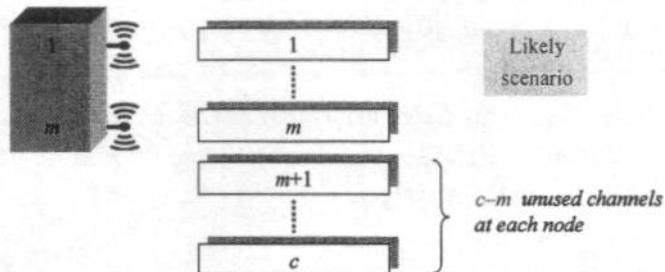
may incur delay.



In the following we focus on a wireless mesh network architecture with multiple radios (or NICs) per node (commonly termed *multi-radios multi-channels*) for effective use of given spectrum. This architecture can not only leverage inexpensive commodity 802.11 hardware, but also overcome deficiencies of single radio solutions.



In real mesh networks the following condition generally holds:  $m < c$ .



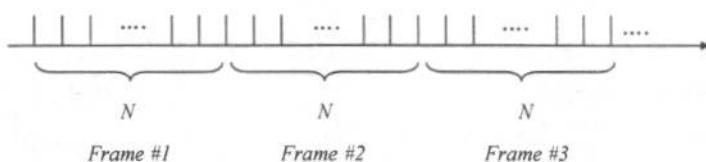
Such multi-channel multi-radio networks pose an interesting set of resource allocation problems, including:

- *channel-assignment*: what are the set of channels the NICs belonging to a node should operate on? (MAC Layer problem);
- *routing*: how to select paths that minimize interference and increase throughput? (Network Layer problem).

These two problems are inter-related with each other, and thus form a challenging cross-layer control problem across the MAC layer and the network layer. Using multiple channels and multiple radios can alleviate but not eliminate the interference. To achieve robust and collision free communication, there are two alternatives. One is to utilize a random access MAC layer scheme, the other is to carefully construct a Transmission Schedule (or Link Schedule) in the framework of a slotted system.

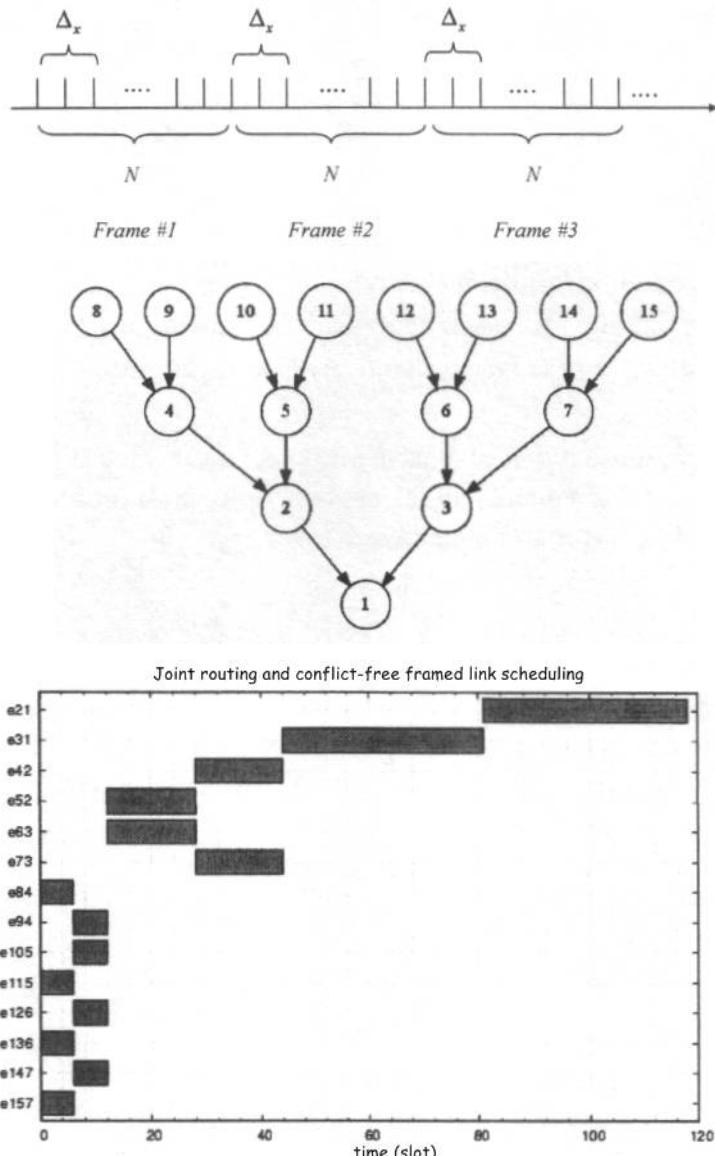
### 13.3 Link Scheduling

*Link Scheduling* is commonly used in the framework of a system where each node (mesh router) is equipped with a single time slotted channel where transmission slots of a fixed duration  $T_s$  are grouped into a frame of  $N$  slots, which is periodically repeated every  $N T_s$  time units.



Each slot is assigned to sets of non-interfering links through *conflict free link scheduling*, so as to prevent the wireless links from interfering with each other. Stated differently, during every slot, a subset of links may be activated for transmission with the assurance that no conflicts occur at the

intended receivers. Therefore, a link  $L_x$  which is activated for  $\Delta_x$  slots in a frame can be characterized by means of a long-term minimum guaranteed rate equal to  $R_x = C_x \Delta_x / N$ ,  $C_x$  being its capacity.



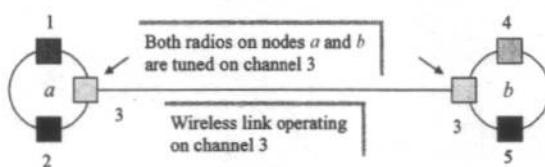
Such Link Scheduling poses an interesting set of resource allocation problems, including:

- *link scheduling*: when should each link be activated?
- *routing*: how to select paths that minimize for example  $N$ ?

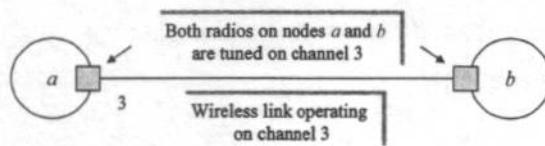
Again, these two problems are inter-related with each other, and thus form a challenging cross-layer control problem across the MAC layer and the network layer.

### 13.4 Channel Assignment

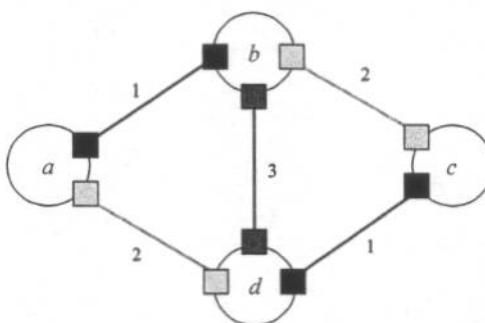
With multiple radios (i.e., multi-radios) operating on different channels (i.e. multi-channels), two nodes can communicate only if each of them has an interface assigned to a common channel. When two neighbouring nodes can communicate they can establish a *logical link*.



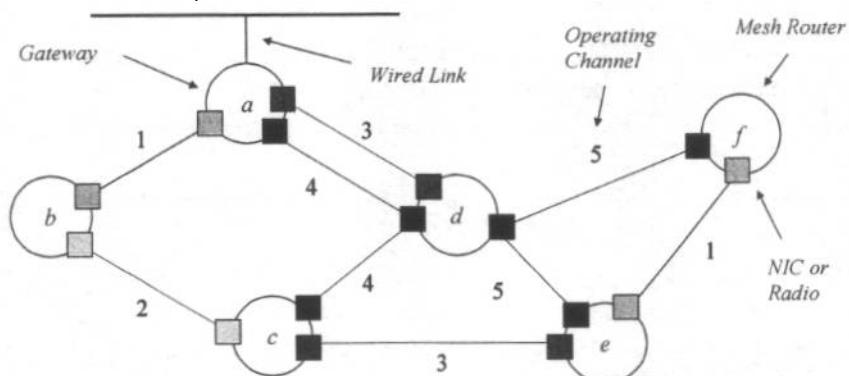
In the figure below nodes  $a$  and  $b$  has established a logical link ( $a,b$ ) using channel number 3.



A logical link is *active* if it is being used for packet transmission. A logical link is *idle* if it is not active. A logical (or network) topology is comprised of the sets of nodes and logical links. The use of multiple radios per node allows simultaneous transmission and reception on different radios tuned to different (frequency) channels, which can substantially improve multihop throughput.



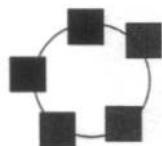
In practice we may realistically consider 3 channels per radio, 3 radios for node, and 50 nodes, which gives 150 radios vs 3 channels. The total number of radios is much higher than the number of available channels. As a consequence many links between mesh routers of a WMN will be operating on the same set of channels (e.g. router a operates on channels 1, 3, and 4 while node a operates on channel 1, 3, and 5).



Therefore the interference problem still persists when, for example:

- routers a and c transmit at the same time to router d on channel 4,
- router a transmits to router d on channel 3 and, at the same time, router c transmits to router e on the same channel.

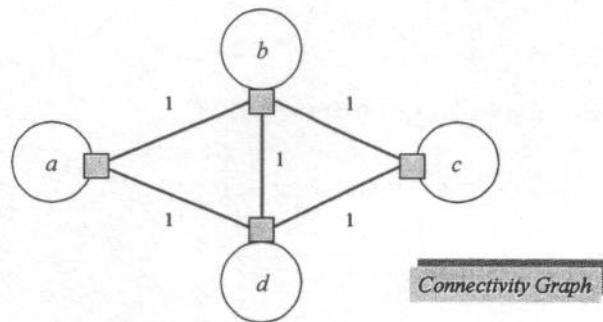
The interference among transmissions on these channels can dramatically decrease their utilization (e.g., due to contention among the nodes, as in the IEEE 802.11 protocol). Therefore the main factor for minimizing the effect of interference is the efficient reuse of channels. A key issue to be addressed in a multi-radio multi-channel WMN architecture is the *channel assignment problem*, that involves assigning (mapping or binding) each radio to a channel in such a way that efficient utilization of available channels can be achieved. The assignment of channels to radios induces the *network topology graph*. Specifically, a *network topology graph*  $T = (N, E)$  is defined as an undirected graph where  $N$  represents the set of nodes, and  $E$  represents the set of logical links in the graph.  $\forall i, j \in N, (i, j) \in E$  if one of the  $i$ 's radios and one of the  $j$ 's radios share a common channel and they are within the transmission range of each other. Moreover, if the multiple radios at  $i$  and  $j$  share  $n$  common channels, there are  $n$  logical links  $(i, j) \in E$ , where  $n$  is a positive integer. The connectivity graph is independent of the channel assignment, but the network topology graph (network topology for short), on the contrary, is determined only after the channel assignment is complete. This difference arises from the constraint that, in a multi-radio multi-channel environment, two neighbor radios must share a common channel to communicate. Due to this constraint, altogether two kinds of discrepancies can occur between the connectivity graph and the network topology graph. First, a link in the connectivity graph may be absent in the network topology if the nodes at the end points of this link do not have any radios assigned to the same channel. Second, it may have several corresponding links in the network topology if the nodes at the end points have more than one radio each with common channels. Generally speaking, the goodness of a channel assignment rests on two factors: *connectivity*, and *interference* (dependent on load). Assigning many radios to a few channels can provide richer connectivity, but has the undesirable effect of increasing interference among transmissions on those channels. For example, it's unreasonable to assign 2 channels to 5 radios installed on the same node since some radios operate on the same channel.



Radios operating on the same channel clearly interfere when activated at the same time. Thus, the channel assignment has to balance between minimizing interference (on any given channel) and maintaining sufficient connectivity. In other words, there exists a trade-off between maximizing connectivity and minimizing interference. In this sense, the channel assignment in a multihop wireless network such as a WMN can be viewed as a topology control problem. Unlike a wired network, links in a wireless network can be tuned or configured. The tunable parameters in a wireless environment can be:

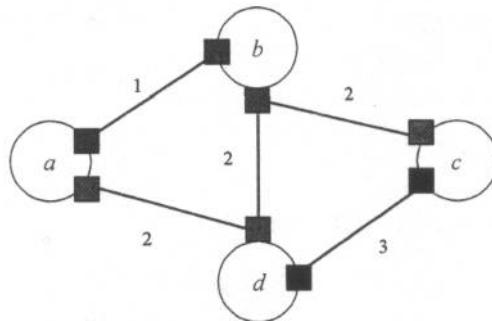
- channel frequency,
- transmission power,
- bit rate,
- directional transmission (using directional antennas).

In a broad sense, topology control exploits these parameters in order to obtain a desired network topology of the network. This can be one of the roles of the channel assignment in WMNs in addition to maximizing connectivity and minimizing interference. To show the relationship between connectivity and interference we will analyze the following WMN architectures: single-radio single-channel and multi-radio multi-channel. A single-radio single-channel architecture limits the whole network to operate in one single channel.



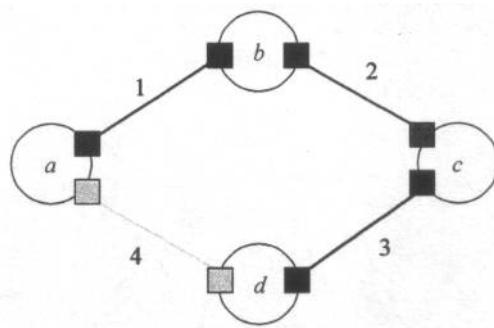
In this scenario, a link is placed between two nodes if they are inside their respective transmission ranges (c is outside the transmission range of a and vice versa). This is the maximum achievable network connectivity since a single common channel is shared between all the nodes. In the example shown in the next figure for the multi-radio multi-channel architecture we assume that there are four non-overlapping channels available for communication and that every node is equipped with two radios (NICs). Furthermore we discuss two scenarios:

- Scenario 1: the network connectivity is maximized (same as single-radio single-channel connectivity), and
- Scenario 2: the interference is minimized (efficient use of the available channels).



*Scenario 1: Maximum Connectivity*

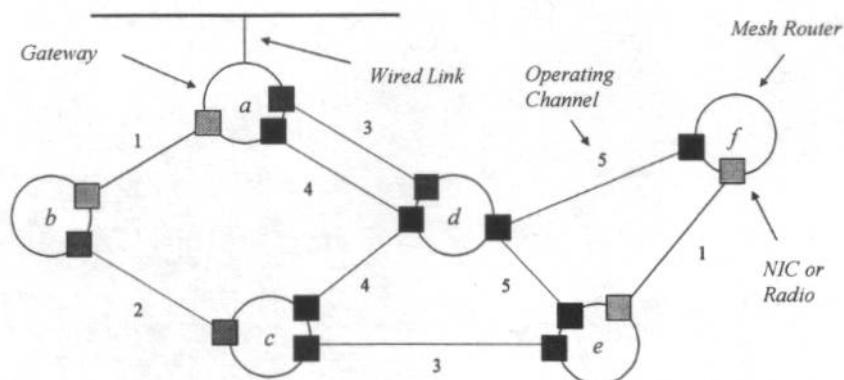
The assignment of channels to the radios results in maximum network connectivity. A logical link exists between every pair of neighbors. However, not all the logical links can be active simultaneously because of possible interference (logical link 2).



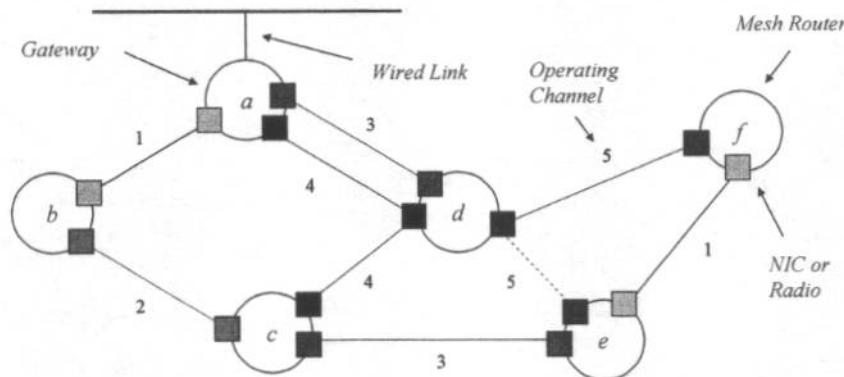
Scenario 2: Minimum Interference

The figure shows how interference could be completely eliminated and all logical links can be simultaneously active. The compromise here is that there is no common logical link between neighbors b and d. For a packet to reach d from b it takes two hops instead of one as it happens when there is a direct logical link between b and d.

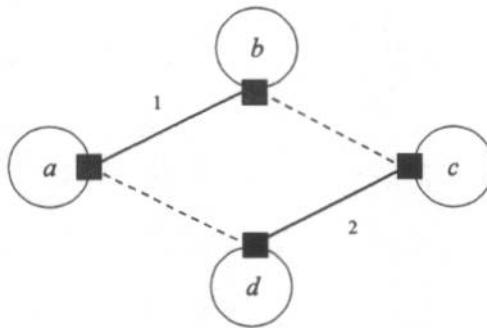
The next slide figures a more complex example of a WMN with six wireless mesh nodes where logical links and channels have been established. Each node has at maximum three radios. Five channels are being used in the network.



Both logical links (a,b) and (f,e) are assigned to channel number 1. In this case, interference will occur if these logical links are close to each other. If these logical links interfere they cannot be active simultaneously. In practice, as we said before, the number of available radios is limited. Providing up to three/four radios is considered reasonable. A small number of radios implies that some logical links in a node may need to share a radio to transmit and receive data packets. Logical links (d,f) and (d,e) share a radio on router d. Thus, they are required to operate over the same frequency channel (i.e., channel number 5). When two logical links in a router share a radio, they cannot be active simultaneously. It significantly reduces their effective capacity. The effective link capacity can be increased by removing some of the logical links from the network topology. For example, by removing logical link (d,e), logical link (d,f) can be attached to an exclusive (not shared) radio on router d.



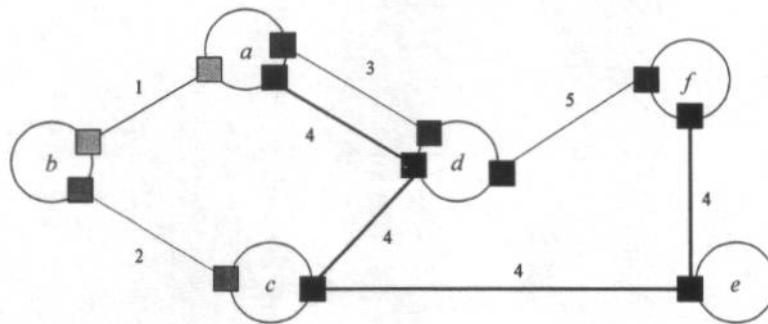
However, this may increase the number of hops through some of the routing paths (e.g., from router e to the gateway it takes 3 hops instead of 2). In certain cases, the network topology may not even be connected. Assume one radio per node and two channels (1 and 2).



If a logical link on channel 1 is established between nodes a and b and another logical link is established on channel 2 between nodes c and d, the resulting network topology is disconnected since nodes a and b cannot communicate with nodes c and d.

### 13.5 Ripple Effect

A key problem in the design of channel assignment for multi-radio multi-channel WMNs is the channel dependency among the logical links that share a common radio. Consider the WMN shown in figure where five non-overlapping channels are available.



Notice that logical links (a,d), (d,c), (c,e) and (e,f) all share channel 4 and therefore, if anyone of the nodes a, d, c, e, and f decides to reassign the channel (common) to these logical links, then the rest of the logical links have to change their channel assignment which produces a *ripple effect*.

### 13.6 Problem Formulation

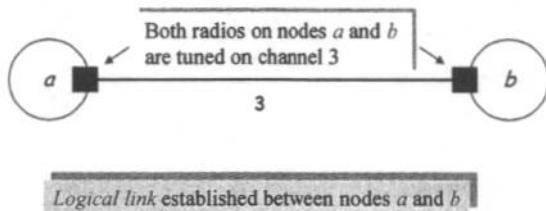
Thus, the channel assignment goal is to facilitate effective utilization of available channels. This is equivalent to reducing the interference on any given channel. With the above view, channel assignment becomes an optimization problem, where some interference measure defined over the whole network according to a given interference model is optimized, with the constraint that some notion of connectivity is preserved. The problem formulation of channel assignment requires as inputs:

- The interference model to represent the interference (protocol models vs. physical model).
- The measure of interference that serves as the objective function; we may choose to minimize the maximum interference (i.e., link conflict weight) at any link or, alternatively, we can also minimize the average interference over all links.
- The constraints.

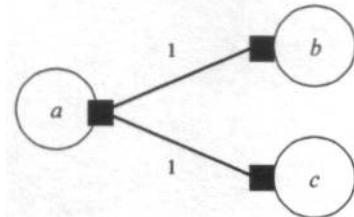
### 13.7 Constraints

Here is a list of main constraints:

1. The total number of channels is fixed. Suppose that there are K distinct channels denoted by  $\{1, 2, \dots, K\}$ .
2. The number of distinct channels that can be assigned to a node is limited by the number of its radios. If  $R_i$  denote the number of radios at node i, then  $1 \leq R_i \leq K$ . Different nodes may have different number of radios.
3. Two nodes, that share a logical link expected to carry certain amount of traffic, should be bound to a common channel.



4. The sum of the expected traffic loads on the logical links that share the same channel and that interfere with each other should not exceed the channel's raw capacity. The sum of the offered loads on the two logical links shown in the figure below and sharing channel 1 should not exceed the raw capacity of that channel.



Channel assignment algorithms can be partitioned into the following two classes: *load independent* and *load aware*. In the following slides we will focus on *CLICA* (*load independent*).

### 13.8 Load Independent Problem Formulation

The load independent channel assignment problem requires as inputs:

- the 11protocol model to represent the interference,
- the conflict graph with the associated *link conflict weights*, which is simply the sum of weights of the edges incident to the vertex in the conflict graph corresponding to a network link,
- the measure of interference that serves as the objective function (for example minimize the maximum interference or link conflict weight at any link),
- the connectivity constraints.

One reasonable choice for this constraint is to require that all links in the connectivity graph are still "preserved" in the network topology after the channel assignment is complete). This ensures that the shortest path length (in number of hops) between any two network nodes does not increase because the network topology becomes different from the original connectivity graph due to channel assignment.

Given:

- Connectivity Graph  $G = (V, E)$
- $K$  distinct channels
- $R_i$  radios at node  $i$
- 11protocol model
- non-negative integer  $M$

the above optimization problem can be formulated as follows: "is there a connectivity-preserving assignment of channels to radios such that the maximum link conflict weight in the resultant network topology graph  $\leq M$ ?"

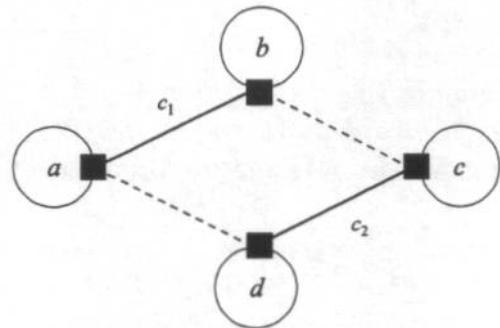
### 13.9 Analysis of Complexity

Most of the traditional channel assignment problems for wireless networks are known to be difficult and have close relationship with *graph coloring* problems. Even though at first sight, there does not seem to be any connection between these problems and the above channel assignment problem, a closer look does reveal that our problem is in fact a generalized version of a well-known graph edge coloring problem. Unlike these other problems which either seek proper coloring (i.e. incident edges have different colors) or conflict-free channel assignment, we attempt to minimize a measure of conflict. Except for the extreme and trivial cases where there is one radio per node or as many radios as number of channels, the optimal solution is intractable as it can be proved that the optimization problem belongs to the class of NP-complete problems.

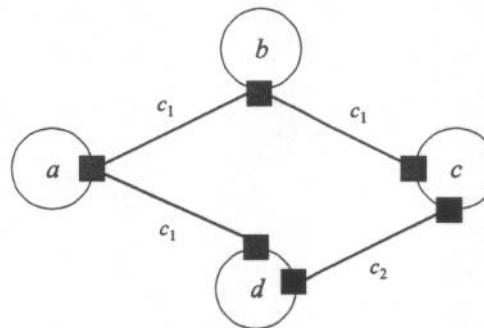
### 13.10 CLICA

In the following, we describe a polynomial-time heuristic called *Connected Low Interference Channel Assignment* or *CLICA* for assigning channels to radios. The CLICA algorithm is proposed in: Mahesh K. Marina and Samir R. Das, “*A Topology Control Approach to Channel Assignment in Multi-RadioWireless Mesh Networks*”, The 2nd International Conference on Broadband Networks, 2005, pp. 381 – 390. Based on the previous discussion about the close relationship between channel assignment and graph coloring, from this point on we sometimes use the term color in place of channel for ease of exposition.

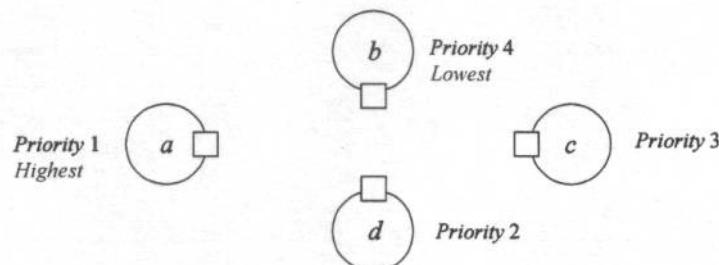
Before we present the CLICA algorithm, let us first look at the pitfall of arbitrarily coloring radios and links using a simple example that we have already analyzed, i.e. a 4 node connectivity graph with one radio per node. Suppose further we are given 2 colors:  $c_1$  and  $c_2$ . If we first color the link (a,b) with  $c_1$  (by assigning channel  $c_1$  to the radios at a and b) and later color the link (c,d) with  $c_2$  in a similar fashion, then we end up with a partitioned network where nodes a and b are disconnected from nodes c and d.



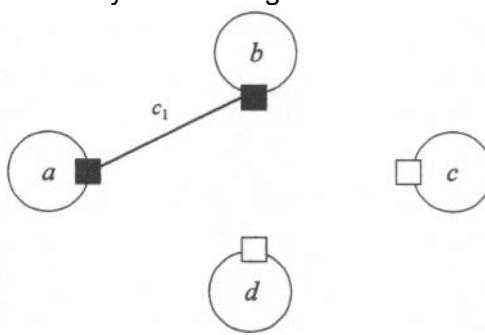
This simple example shows that a coloring decision constrains the flexibility for future coloring decisions if we want to preserve the network connectivity. Specifically, if we want to preserve the network connectivity, nodes c and d are precluded from using color  $c_2$  because of an earlier choice to use color  $c_1$  for link (a,b). However, adding more radio interfaces to nodes will provide more flexibility in coloring.



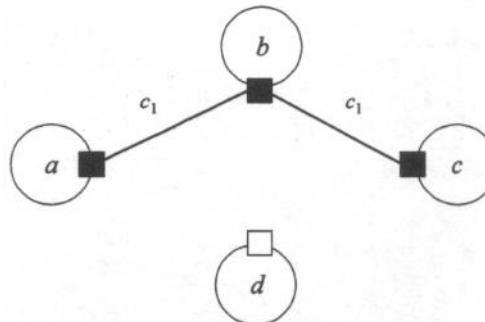
The central idea in the CLICA algorithm is to use that *degree of flexibility* as a guide in determining the order of future coloring decisions. Specifically, each node is associated with a priority, and coloring decisions are made on a node-by-node basis in the order of this priority. The set of coloring decisions at a node  $i$  include choosing colors for radios at  $i$  and its adjacent nodes in order to color all links incident to  $i$  in the connectivity graph. At the beginning of the algorithm, each node is given a priority based on some criterion. These priorities determine the default order for making coloring decisions. However, the algorithm, in the midst of its execution, may override that order by setting priority of a subset of nodes to a value greater than the maximum priority (over all nodes) to reflect the lack of flexibility for coloring radios at nodes in that subset. This characteristic of the CLICA algorithm to alter a node's priority during the course of its execution makes it an *adaptive priority algorithm*. Going back to the previous example, suppose that initial order of priorities is a, d, c and b.



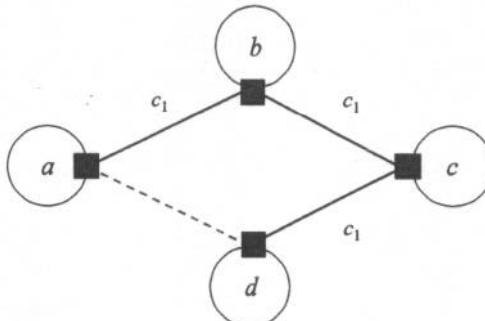
So, CLICA starts at a to color its incident links. Suppose it chooses  $c_1$  to color the link (a,b). As a result, both a and b lose further flexibility in choosing colors for their other incident links.



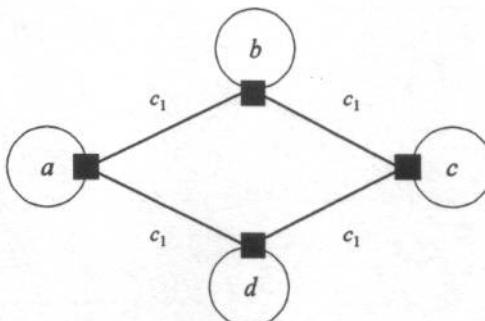
As a consequence, CLICA additionally bumps b's priority to the highest because b hasn't free NICs. Moreover, it recursively starts coloring at b to retain links on other paths connecting a and b (only one path in this example: b-c-d-a), which results in node b reusing color  $c_1$  for link (b,c).



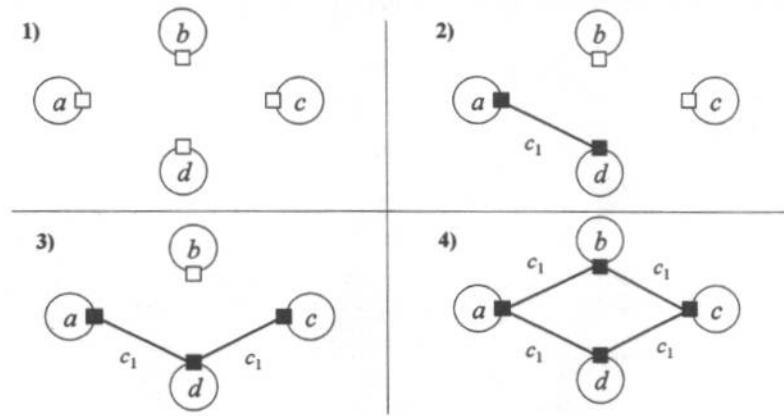
Same procedure as above (i.e., priority increase followed by recursive color reuse) repeats itself at node c forcing link (c,d) to use  $c_1$ , which in turn increases the priority of d.



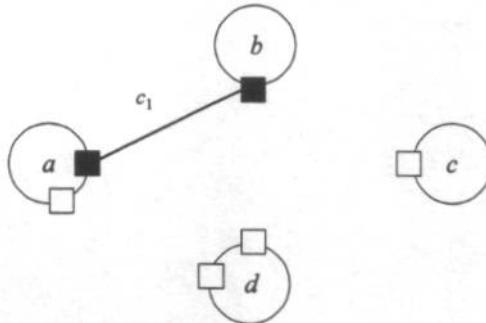
At d, since there is already a common color ( $c_1$ ) with node a, the link (a,d) is colored with  $c_1$ . At this point, CLICA comes out of recursion and terminates.



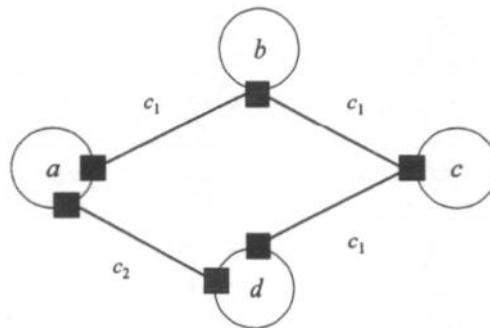
Suppose that initial order of priorities is a, d, c and b. If a starts to color link (a,d) the initial order of priorities doesn't change.



Now suppose that nodes a and d have two radios and the algorithm starts like before at a by coloring link (a,b) with  $c_1$ .



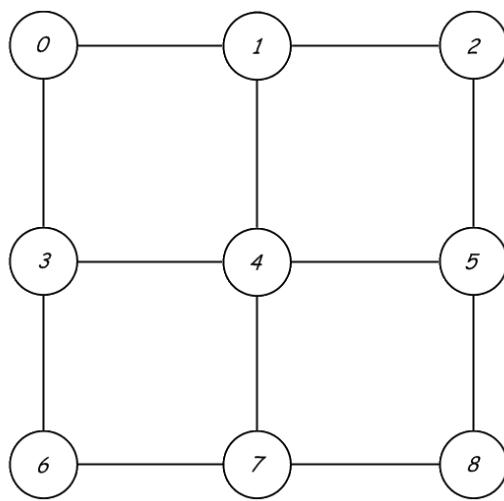
Even in this case, the algorithm goes through recursion to color b and c ahead of d; however unlike in previous case the algorithm colors the link (a,d) with  $c_2$  by using the additional radios.



Note that CLICA is naturally recursive and follows a chain of the least flexible nodes to maintain network connectivity. Also note that it is a one-pass algorithm in that coloring decisions once made are not reversed later in the algorithm execution. Each coloring decision is made in a greedy fashion: node i, when faced with a decision to pick a color for an incident link (i,j), makes a locally optimal choice from among the feasible set of colors: the color that minimizes the maximum link conflict weight over all links that can interfere with the link (i,j) including itself (*GreedyMax*).

### 13.11 CLICA – An Example

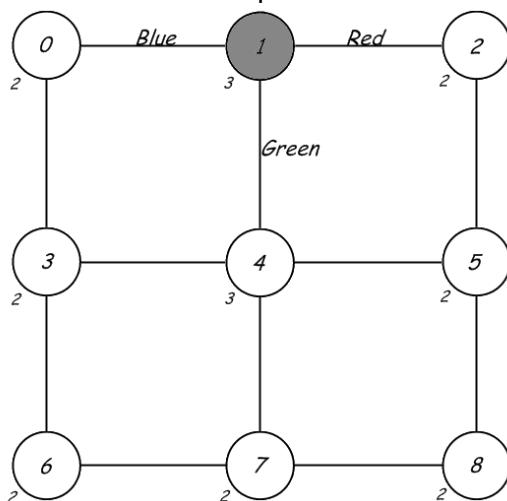
The network consists of 9 nodes, placed in a 3x3 grid.



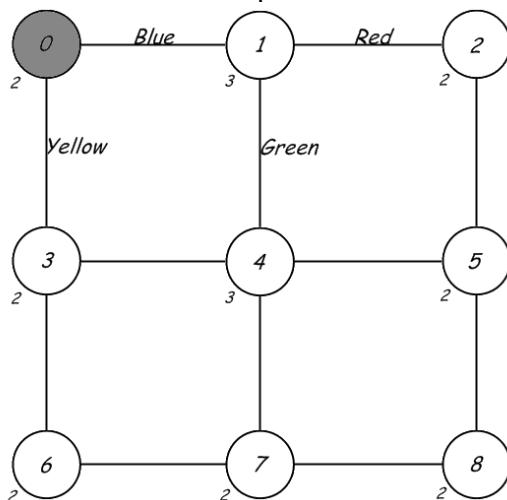
We make no claims that this topology is representative of typical wireless mesh networks. We have deliberately chosen a small, simple topology, to facilitate detailed discussion of the results. We start with several assumptions. We assume that the range of each node is one unit, i.e., just enough to reach its lateral neighbours, but not the diagonal ones. We also assume that the interference range is equal to the transmission range. We assume 11Protocol Model to model interference. The resulting conflict graph for this scenario is shown in matrix form. The number to the left represents the number of radios for each node.

- *Rank:* 1, 0, 2, 4, 5, 3, 8, 7, 6.
- *5 Channels (Colors):* Blue, Red, Green, Yellow, Pink.

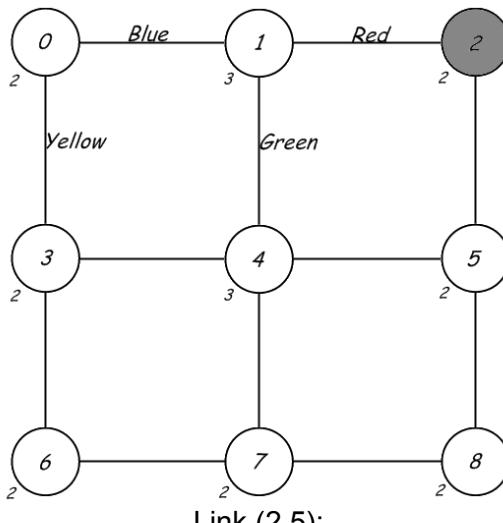
Step 1:



Step 2:



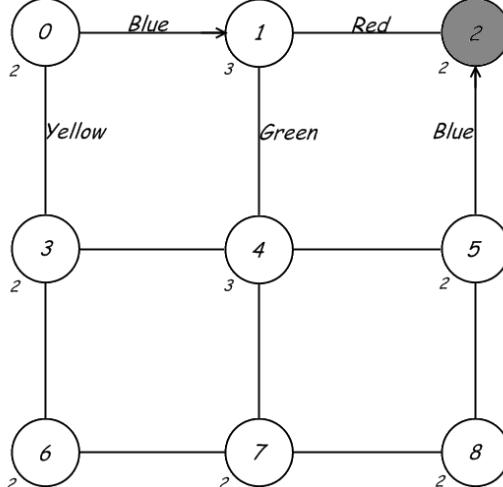
Step 3:



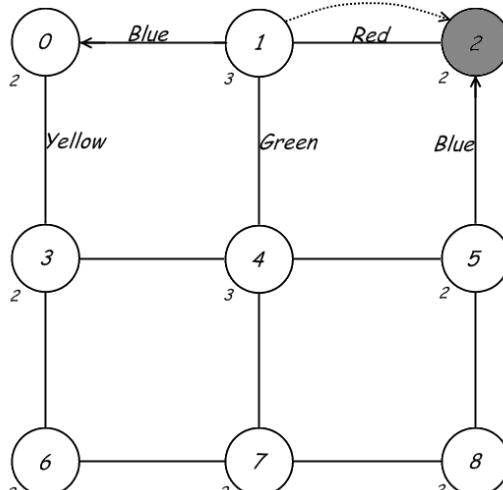
Link (2,5):

Blue	1
Red	X (color already assigned to a radio at node 2)
Green	2
Yellow	0
Pink	0

Regarding link (2,5), let's explain in detail the reason there is 2 instead of 1 in the conflict on the green channel in comparison to the value of 1 of conflict on the blue channel. Let's look first at link (0,1). Because node 2 is two hops away from node 0 and at one hop from node 1, while node 0 is sending to node 1, node 5 can also be transmitting to node 2 simultaneously without any conflict.

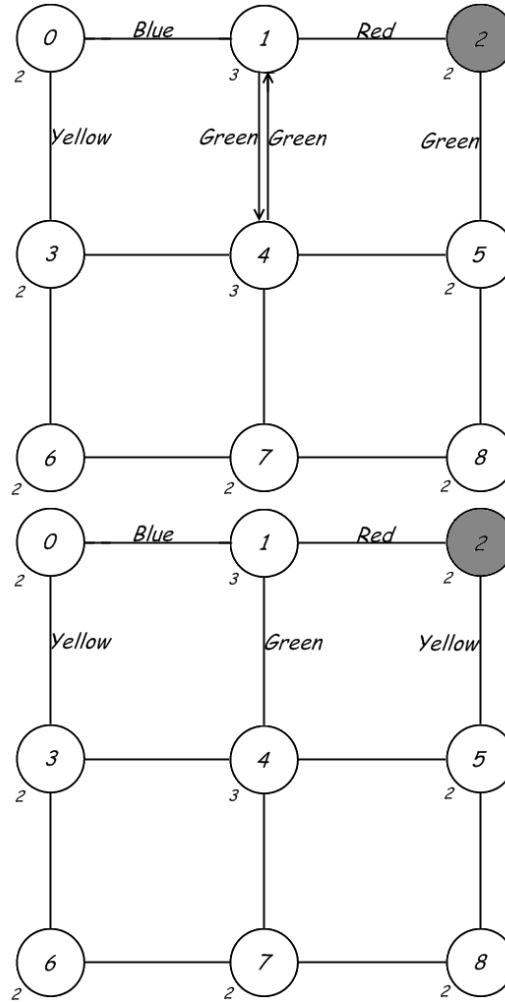


However, the conflict cannot be avoided when node 1 is sending to node 0. Thus 1 is the amount of conflict between (0,1) and (2,5) on the blue channel.

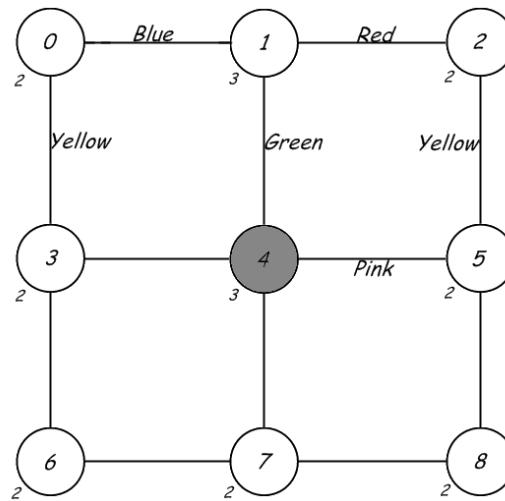


Let's now look at link (1,4). In both cases when node 1 is sending to node 4 and vice versa, link

(2,5) will always have to compete with link (1,4) on the green channel. So giving the value of 2 just means that, there will be more conflict if (1,4) and (2,5) share the green channel than if (0,1) and (2,5) share the blue channel. The same explanation applies at (Step 8) where the conflict on the blue link is also 2 following the same line of reasoning.

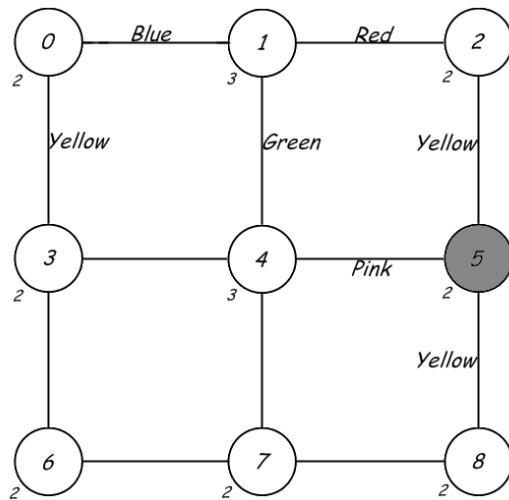


Step 4:



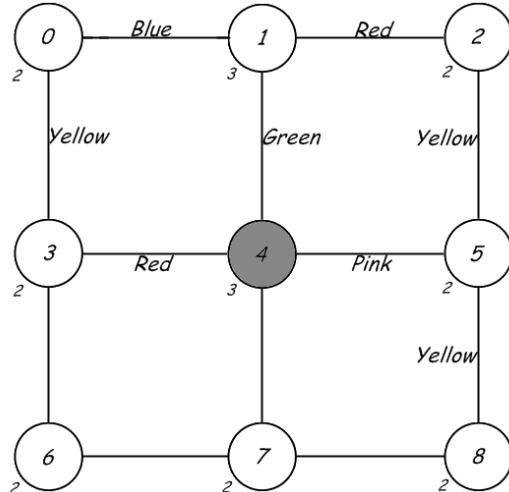
Step 5:  
Link (5,8) (All radios at 5 already assigned):

Blue	X
Red	X
Green	X
Yellow	1
Pink	1



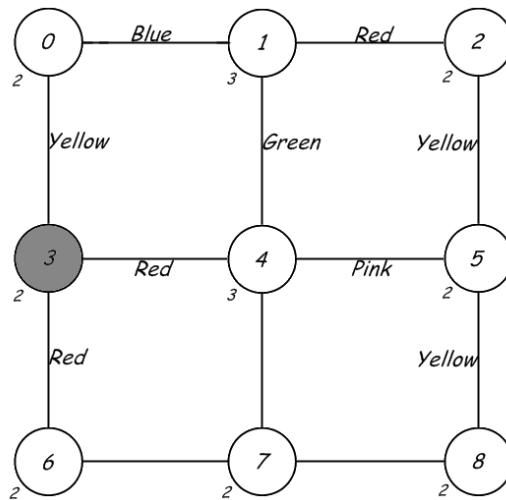
Step 6:  
Link (4,3):

Blue	2
Red	1
Green	X (color already assigned to a radio at node 4)
Yellow	X (color already assigned to a radio at node 3)
Pink	X (color already assigned to a radio at node 4)



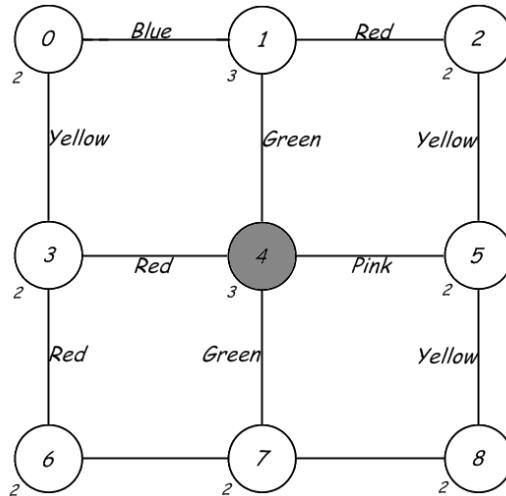
Step 7:  
Link (3,6) (All radios at 3 already assigned):

Blue	X
Red	1
Green	X
Yellow	1
Pink	X



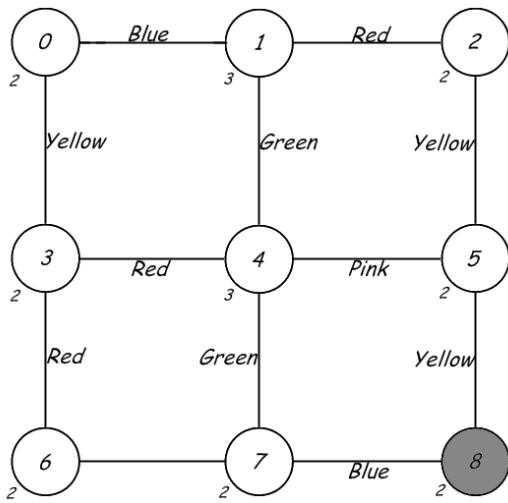
Step 8:  
Link (4,7) (All radios at 4 already assigned):

Blue	X
Red	3
Green	1
Yellow	X
Pink	1



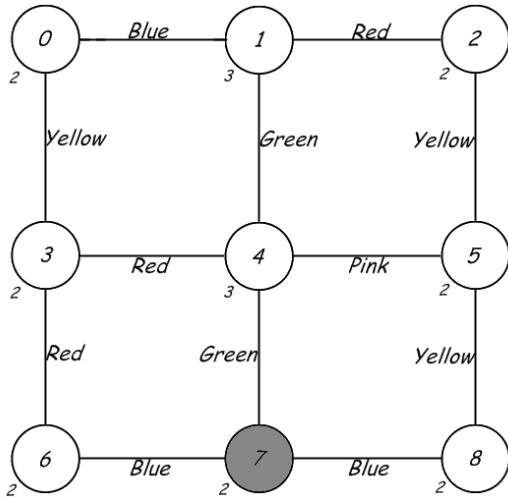
Step 9:  
Link (8,7):

Blue	0
Red	2
Green	2
Yellow	2
Pink	1



Step 10:  
Link (7,6) (All radios at 7 already assigned):

Blue	1
Red	X
Green	2
Yellow	X
Pink	X



# 14 Ad-hoc On-demand Distance Vector (AODV)

## 14.1 Overview of AODV

This text contains citations from C. E. Perkins and E. M. Royer, "Ad-Hoc On Demand Distance Vector Routing" Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), pages 90-100, 1999.

*Ad-hoc On-demand Distance Vector (AODV)* is a *reactive (or on-demand) protocol* for routing on *Mobile Ad-hoc NETworks (MANET)* and it deals with route table management. The AODV routing protocol is designed for mobile ad hoc networks with populations of tens to thousands of mobile nodes. AODV can handle low, moderate, and relatively high mobility rates, as well as a variety of data traffic levels. It has been designed to reduce the dissemination of control traffic and eliminate overhead on data traffic, in order to improve scalability and performance. AODV retains the desirable feature of DSR that routes are maintained only between nodes which need to communicate. AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that DATA packets do not have to contain routes. No caches are used: there is only one route per destination in the routing table. It only maintains the freshest route, if multiple possibilities are available. Route table information must be kept even for short-lived routes, such as are created to temporarily store reverse paths towards nodes originating *RREQs*. AODV uses the following fields with each route table entry:

- Destination IP address,
- Destination Sequence Number,
- Hop Count to Destination,
- Next Hop,
- List of Precursors,
- Lifetime (i.e. expiration time for this route table entry).

When a source node wishes to send a DATA packet to some destination node, the former node checks its routing table to determine if it has a current route (or path) to the destination. If yes, forwards the DATA packet to Next Hop node; if no, it initiates a Route Discovery Process by broadcasting a *Route REQuest (RREQ)* packet to its neighbors. The Route Discovery Process requires three message types:

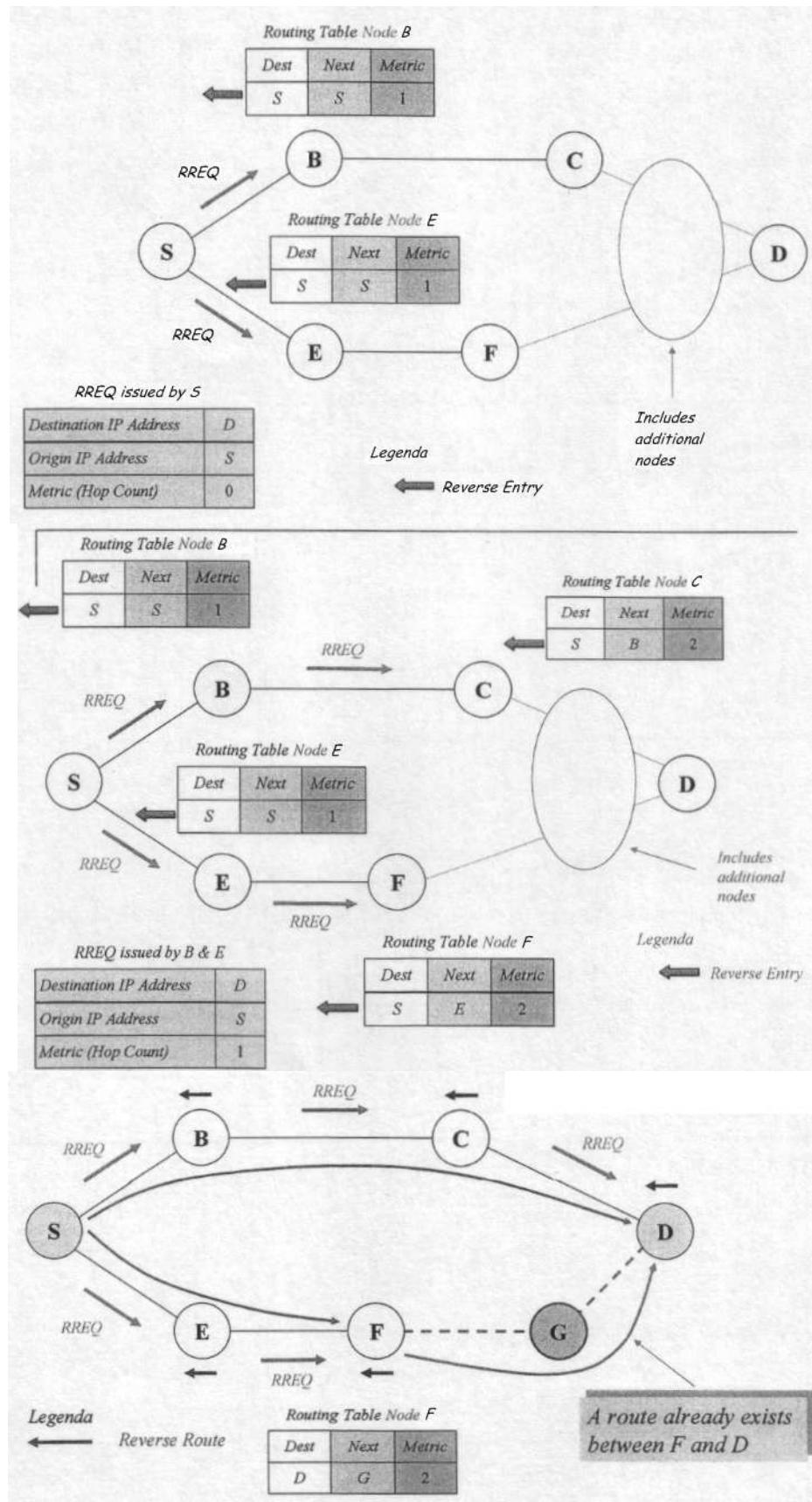
- *Route Request (RREQ)*,
- *Route Reply (RREP)*,
- *Route Error (RERR)*.

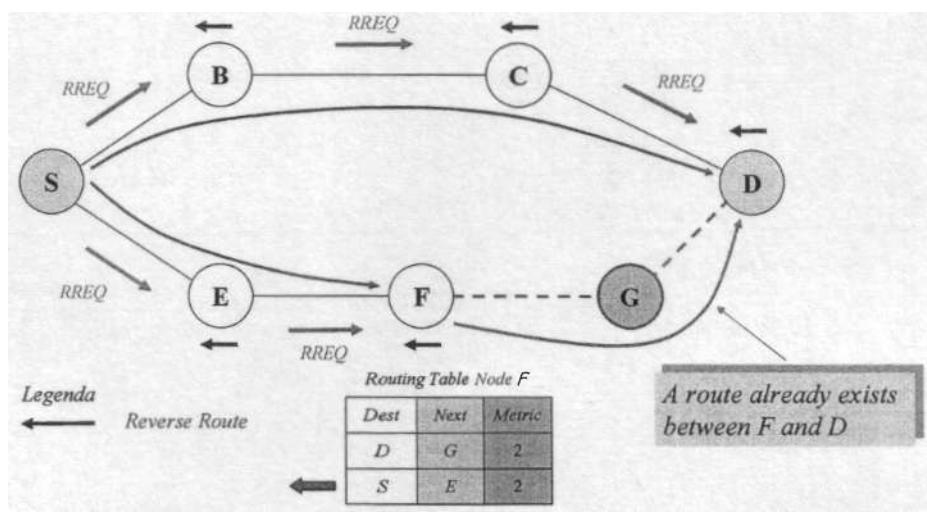
All AODV messages are sent to port 654 using UDP and normal IP header processing applies. For instance, the requesting node is expected to use its IP address as the originator IP address for the messages. For broadcast messages, the IP broadcast address (255.255.255.255) is used.

## 14.2 Route Request (RREQ)

As long as the endpoints of a communication connection have valid routes to each other, AODV does not play any role. When a route to a new destination is needed, the node uses a RREQ to find a route to the destination. A route can be established when the RREQ reaches either the destination itself or an intermediate node with a "*fresh enough*" route to the destination. A "*fresh enough*" route is a valid route entry for the destination whose associated (i.e. destination) sequence number is at least as great as that contained in the RREQ. Each node receiving the RREQ caches a route back to the source (reverse route) of the request.

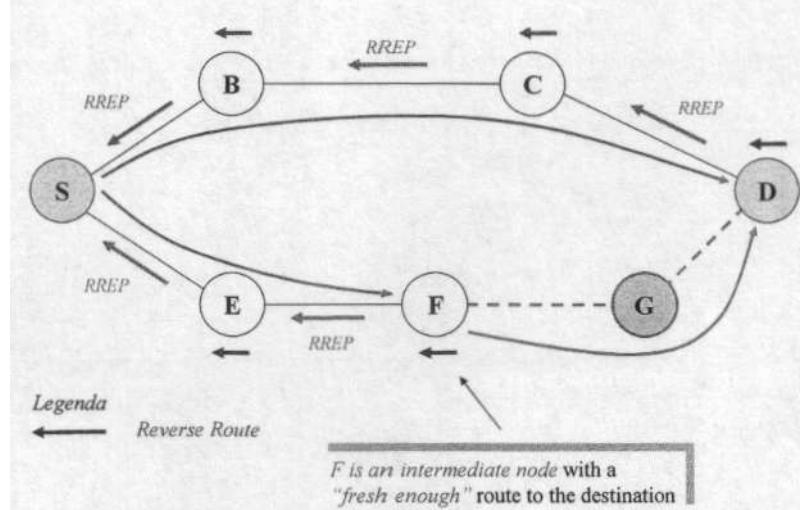
Assume that S needs to establish a route to D:





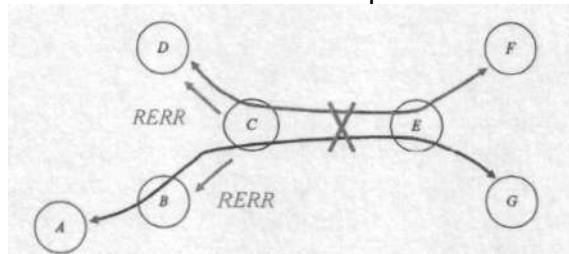
### 14.3 Route Reply (RREP)

The route is made available by unicasting a *RREP* back to the source of the *RREQ*. Since each node receiving the *RREQ* caches a route back to the source of the request, the *RREP* can be unicast back from the destination along a path to that source, or likewise from any intermediate node that is able to satisfy the request.



### 14.4 Route Error (RERR)

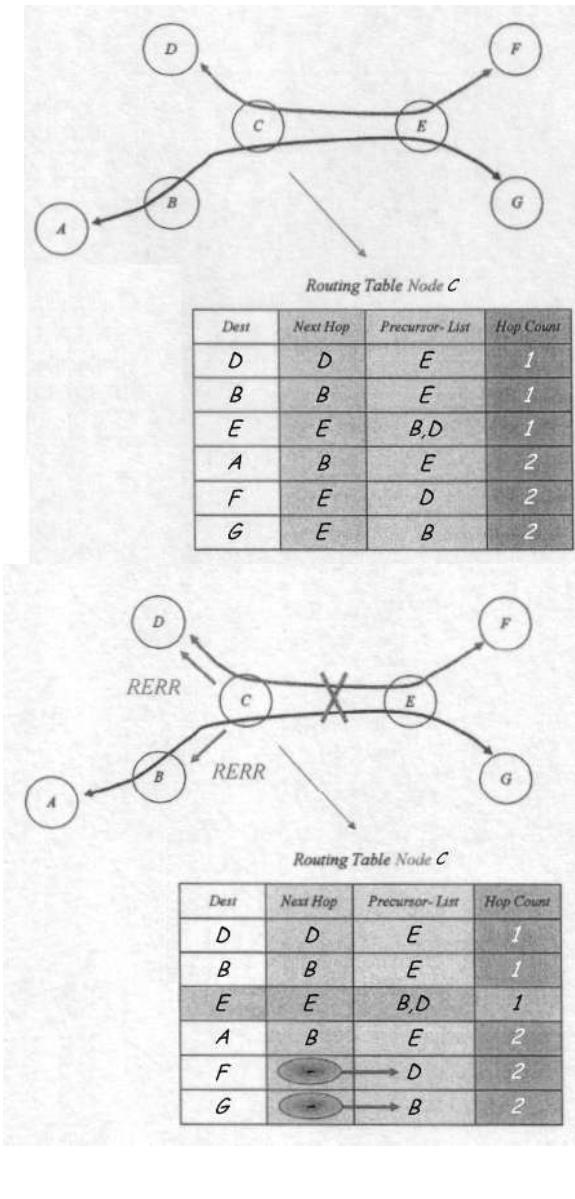
Nodes monitor the link status of next hops in active routes. When a link break in an “active route” is detected, a *RERR* message is used to notify other nodes that the loss of that link has occurred. NOTE: An active route is a route towards a destination that has a routing table entry that is marked as valid. Only active routes can be used to forward data packets.



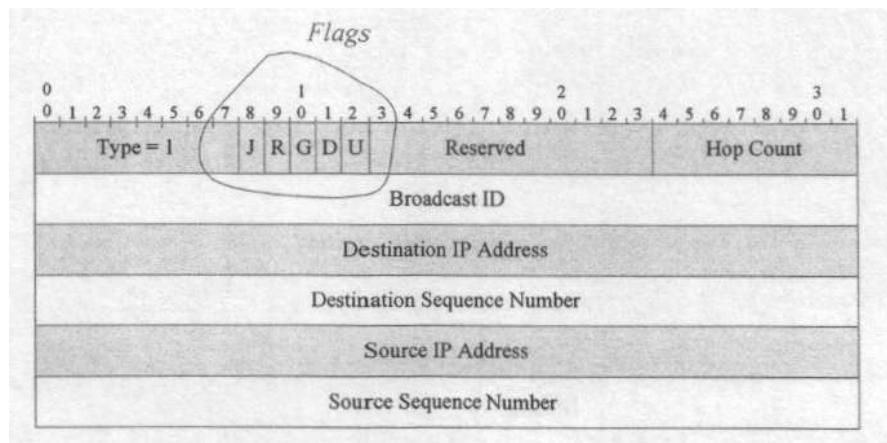
The *RERR* message indicates which destinations are now unreachable due to the loss of the link. In order to enable this reporting mechanism, each node keeps a “precursor list”, containing the IP address for each its neighbors that are likely to use it as a next hop towards each destination. The information in the precursor lists is most easily acquired during the processing for generation of a *RREP* message, which by definition has to be sent to a node in a precursor list.

Precursor List:

(Two communication sessions exist through node C: Node D → Node F, and Node A → Node G)

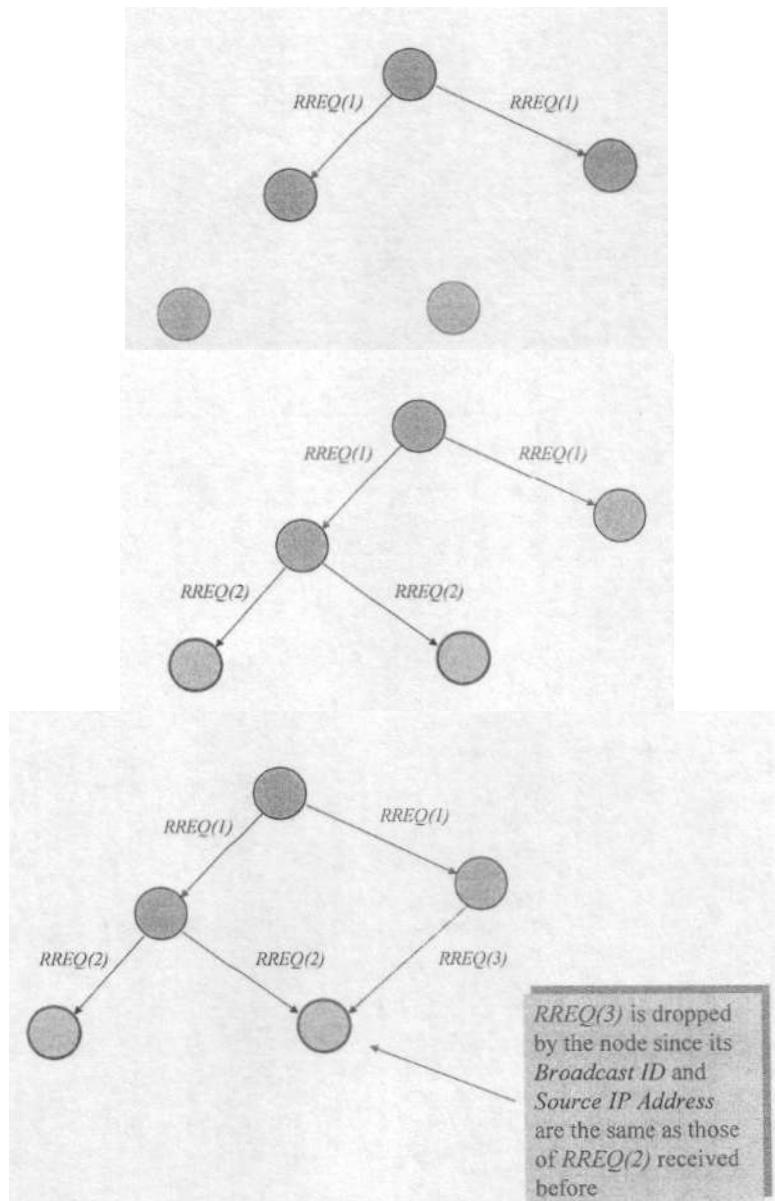


## 14.5 RREQ Format



- *Hop Count* contains the number of hops from the Source IP Address to the node handling the request.
- *Broadcast ID Number* gets incremented each time a source node issues a new RREQ. Broadcast ID and Source IP Address form a unique identifier for the RREQ.

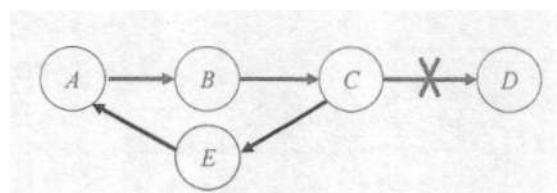
Each neighbor either satisfies the RREQ by sending a route reply RREP back to the source (see below) or rebroadcasts the RREQ to its own neighbors after increasing the Hop Count. Due to flooding, a node may receive multiple copies of the same RREQ from various neighbors. When an intermediate node receives a RREQ, if it has already received a RREQ with the same Broadcast ID and Source IP Address it drops the redundant RREQ and does not rebroadcast it.



- *Destination IP Address* contains the IP address of destination for which a route is desired.
- *Source IP Address* contains the IP address of the node which originated the Route REQuest (RREQ) packet.
- *Destination Sequence Number* contains the latest sequence number received in the past by the originator for any route towards the destination.
- *Source Sequence Number* contains the current sequence number to be used in the route entry pointing towards the originator of the route request.

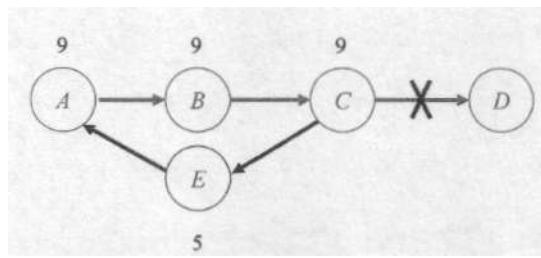
## 14.6 Why Sequence Number in AODV?

Assume that A does not know about failure of link C-D because route error (RERR) sent by C is lost.

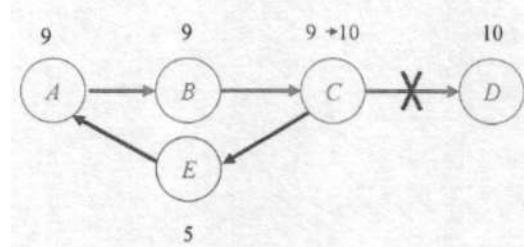


Now C performs a route discovery for D. Node A receives the RREQ (say, via path C-E-A). Node A will reply since A knows a route to D via node B. Results in a loop (for instance, C-E-A-B-C). How using Sequence Numbers can avoid loops?

All numbers shown in the figure are Destination Sequence Number:



Link failure increments the destination sequence number at C (now is 10). If C needs a route to D, RREQ carries the current destination sequence number (10). A does not reply as its own destination sequence number is less than 10.

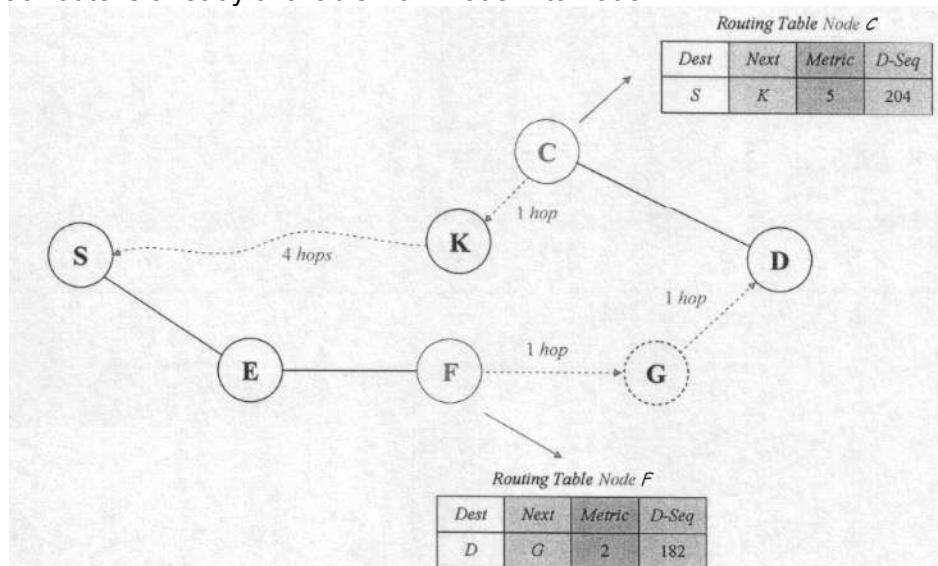


## 14.7 Generating RREQs

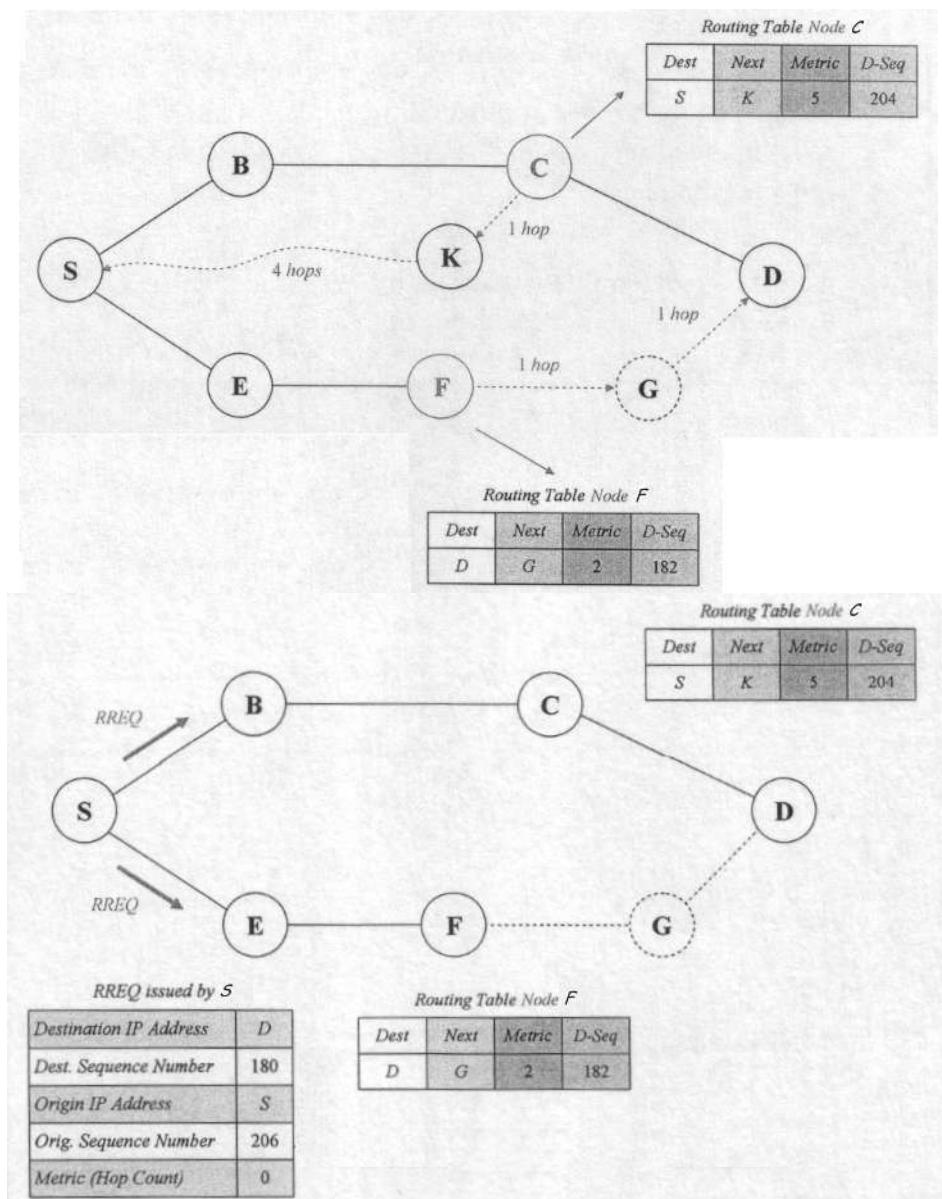
As we said before a node broadcast a RREQ when it determines that it needs a route to a destination and does not have one available. The Destination Sequence Number field in the RREQ message is the last known Destination Sequence Number for this destination. The Originator Sequence Number in the RREQ message is the node's own sequence number, which is incremented prior to insertion in a RREQ. The RREQ ID field is incremented by one from the last RREQ ID used by the current node. Each node maintains only one RREQ ID. The Hop Count field is set to zero.

## 14.8 Example 1

Assume that a route is already available from node F to node D.



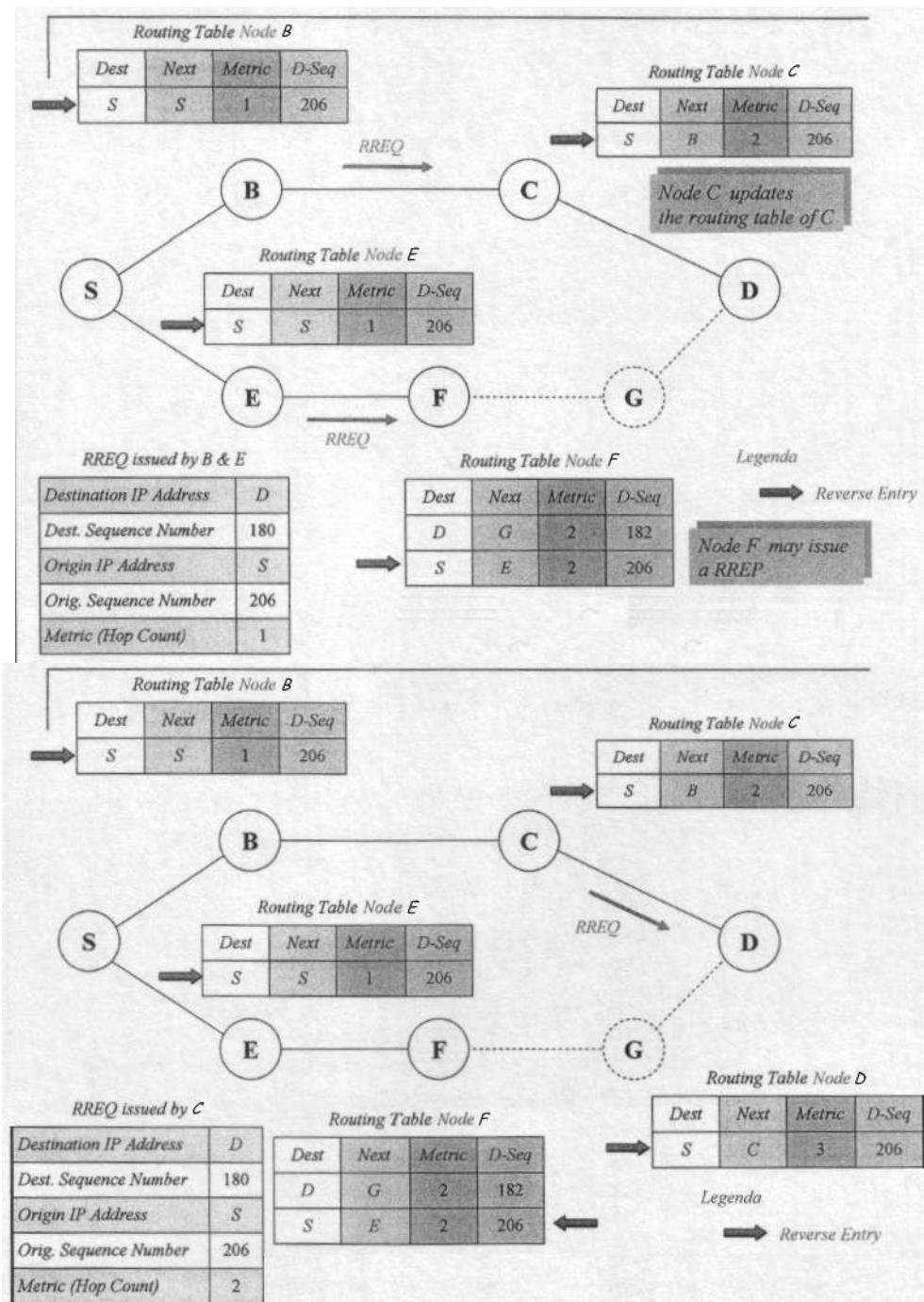
Assume further that node B becomes active when there is already a route from node C to node S.



## 14.9 Processing & Forwarding RREQs

When a node receives a RREQ, it *first* creates or updates a route to the previous hop (Reverse Route) without a valid sequence number, *then* it checks to determine whether it has received a RREQ with the same Originator IP Address and RREQ ID. If such a RREQ has been received, the node silently discards the newly received RREQ. For RREQs that are not discarded by the intermediate node, it first increments the Hop Count value in the RREQ by one to account for the new hop through the intermediate node, then it searches for a reverse route to the Originator IP Address. If need be, the route is created, or updated using the Originator Sequence Number from the RREQ in its routing table. When the reverse route is created or updated, the following actions on the route are also carried out:

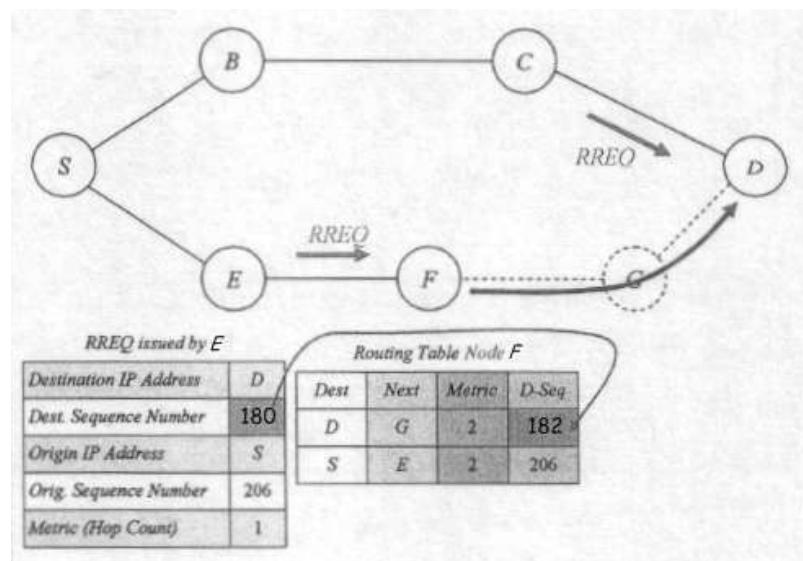
1. The Originator Sequence Number from the RREQ is compared to the corresponding Destination Sequence Number in the route table entry and copied if either:
  - it's greater than the existing value there, or
  - the Sequence Numbers are equal, but the Hop Count as specified by the RREQ is now smaller than the existing hop count in the routing table.
2. The next hop in the routing table becomes the node from which the RREQ was received (it is obtained from the source IP address in the IP header and is often not equal to the Originator IP Address field in the RREQ message).
3. The hop count is copied from the Hop Count in the RREQ message.



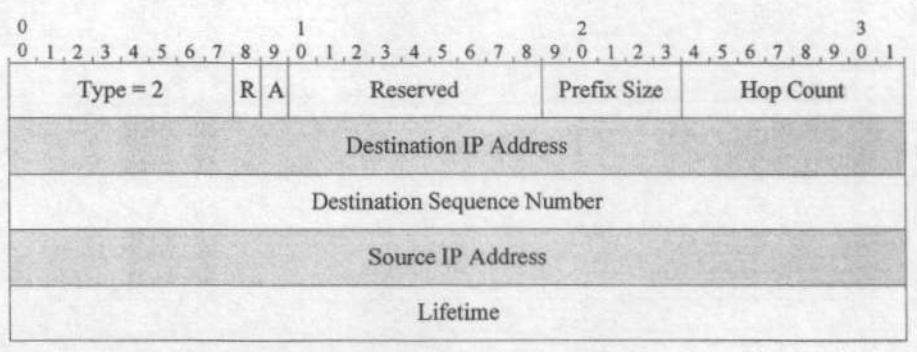
## 14.10 Generating RREPs

A node generates a RREP if either:

- it is itself the destination, or
- it has an active route to the destination and the destination sequence number in the node's existing route table entry for the destination is greater than or equal to the Destination Sequence Number of the RREQ.

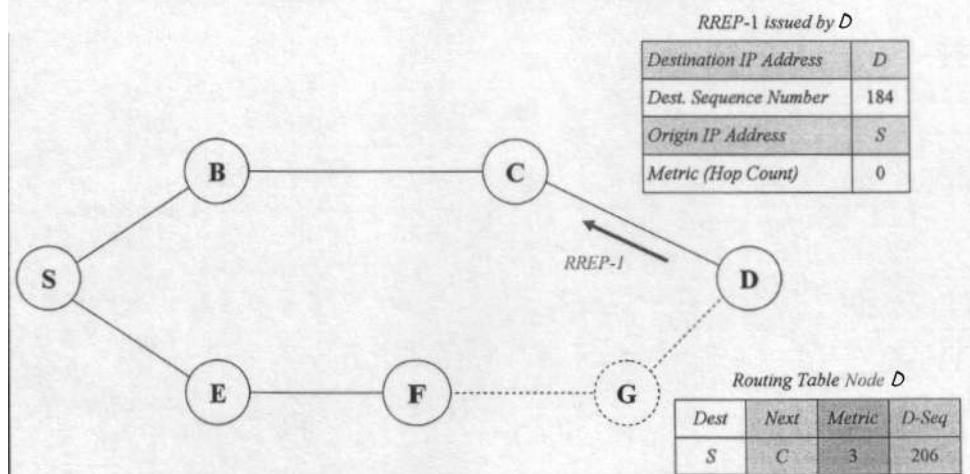


RREP Format:

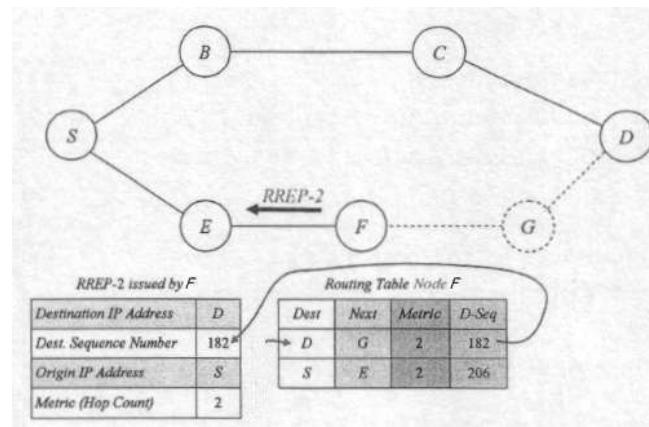


- *Lifetime* contains the time for which nodes receiving the RREP consider the route to be valid.

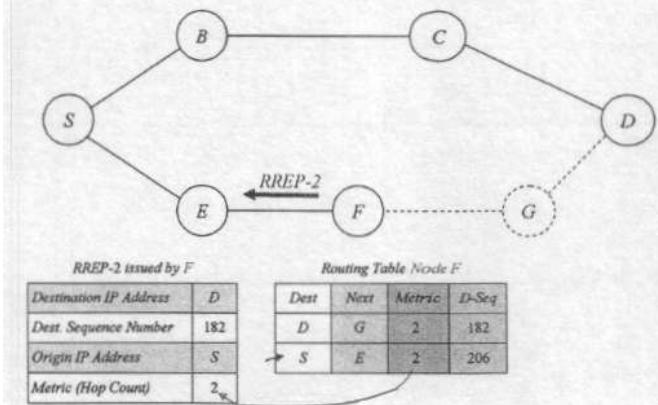
When generating a RREP message, a node copies the Destination IP Address and the Originator Sequence Number from the RREQ message into the corresponding fields in the RREP message. Processing is slightly different, depending on whether the node is itself the requested destination, or instead if it is an intermediate node with a fresh enough route to the destination. Once created, the RREP is unicast to the next hop toward the originator of the RREQ, as indicated by the route table entry for that originator. As the RREP is forwarded back towards the node which originated the RREQ message, the Hop Count field is incremented by one at each hop. Thus, when the RREP reaches the originator, the Hop Count represents the distance, in hops, of the destination from the originator. If the generating node is the destination itself it *MUST* update its own sequence number to the maximum of its current sequence number and the destination sequence number in the RREQ packet. The destination node places its (perhaps newly incremented) sequence number into the Destination Sequence Number field of the RREP, and enters the value zero in the Hop Count field of the RREP.



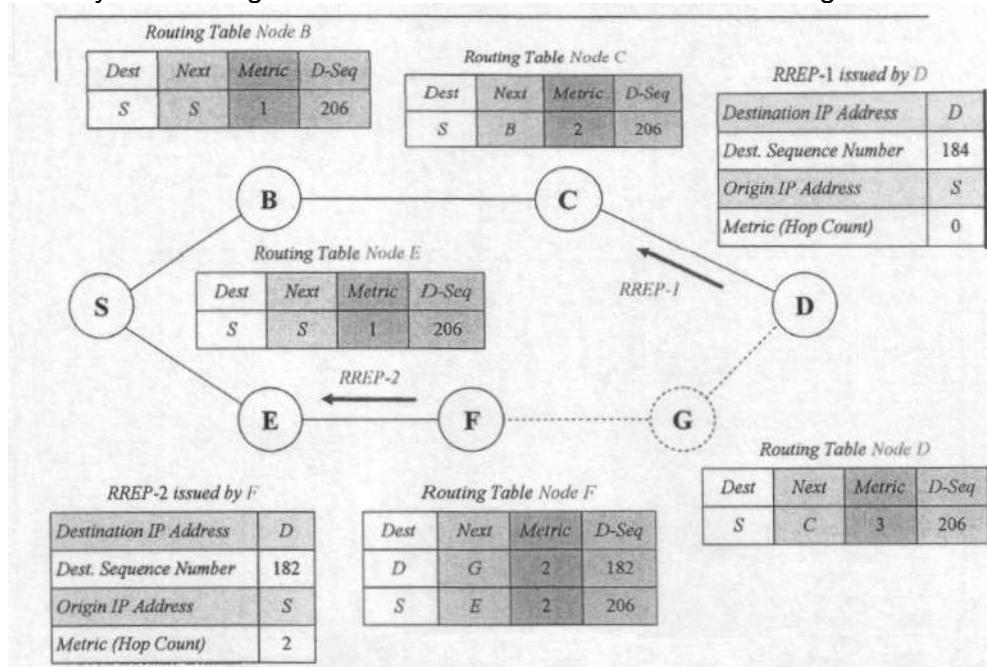
If the node generating the RREP is an intermediate node along the path from the originator to the destination, it copies its known sequence number for the destination into the Destination Sequence Number field in the RREP message.

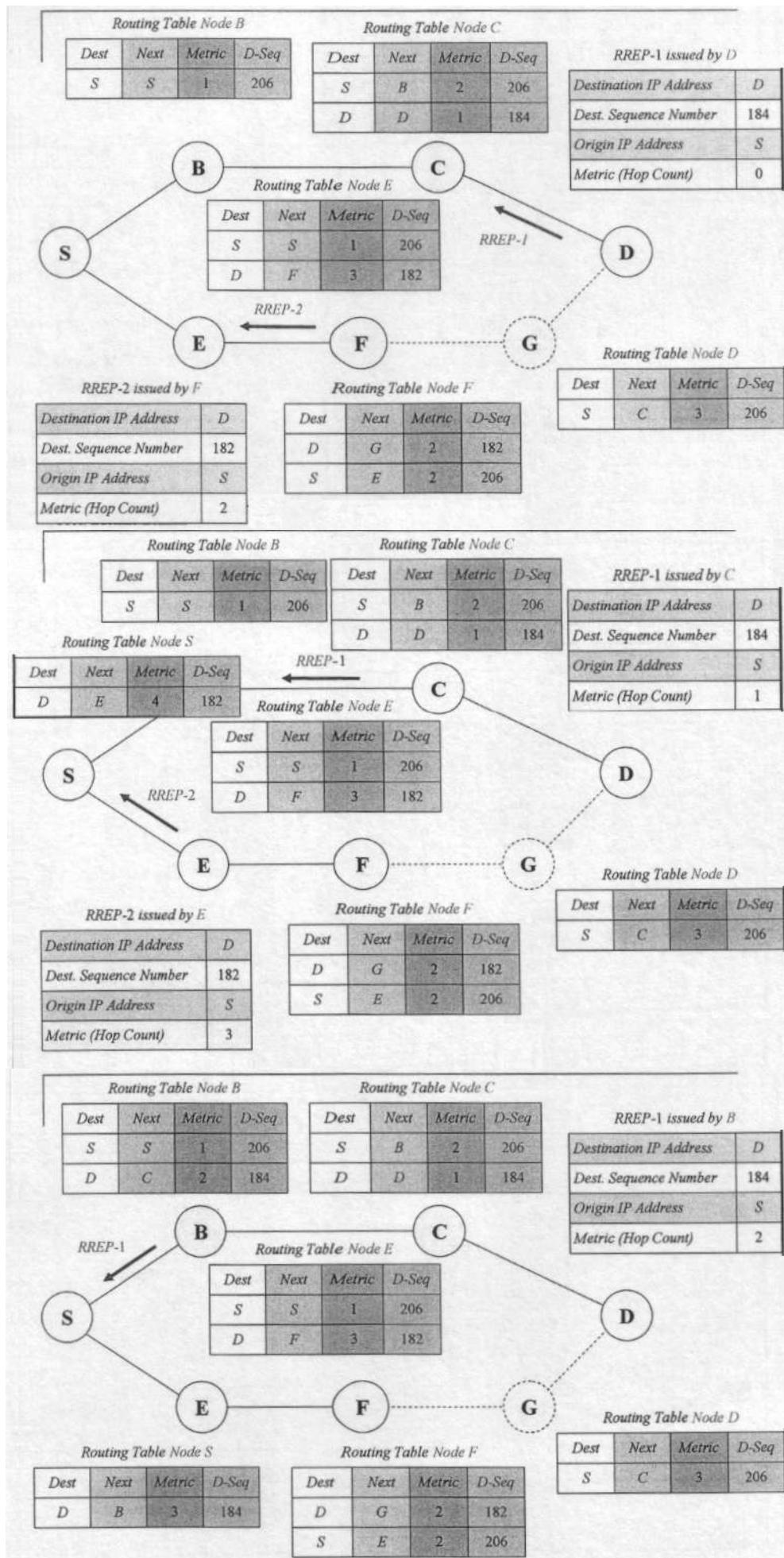


The intermediate node (F) updates the forward route entry by placing the last hop node (E) (from which it received the RREQ, as indicated by the source IP address field in the IP header) into the precursor list for the forward route entry - i.e., the entry for the Destination IP Address. The intermediate node places its distance in hops from the destination (indicated by the hop count in the routing table) Count field in the RREP.



The intermediate node also updates its route table entry for the node originating the RREQ (E) by placing the next hop node (G) towards the destination in the precursor list for the reverse route entry - i.e., the entry for the Originator IP Address field of the RREQ message data.





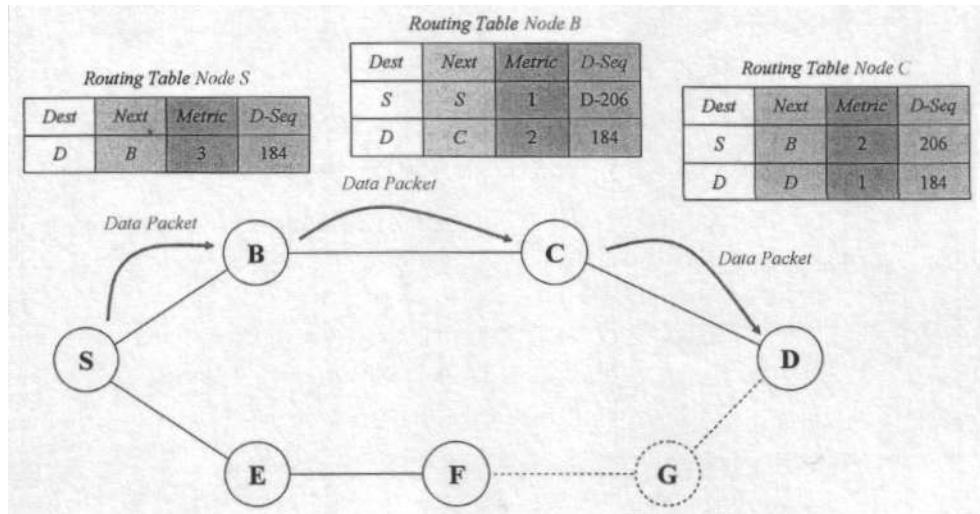
## 14.11 RREP Propagation

A node receiving an RREP propagates the first RREP for a given source node towards that source. If it receives further RREPs it updates its routing information and propagates the RREP only if the RREP contains either:

- a greater Destination Sequence Number than the previous RREP, or
- the same Destination Sequence Number with a smaller Hop Count.

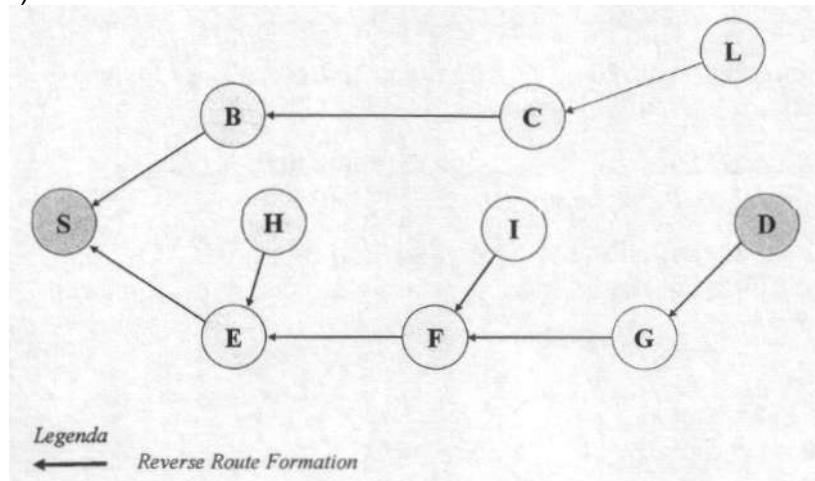
Otherwise it suppresses all other RREP it receives. This decreases the number of RREPs propagating towards the source while also ensuring the most up-to-date and quickest routing information. The source node can begin data transmission as soon as the first RREP is received and can later update its routing information if it learns of a better route.

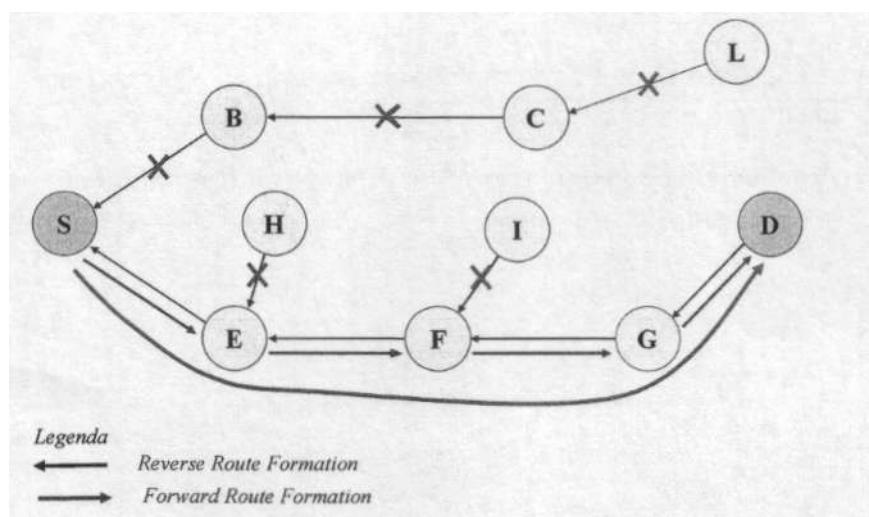
## 14.12 AODV Data Delivery



## 14.13 AODV Timeouts

A routing table entry maintaining a reverse route is purged after a *timeout* interval. Timeout should be long enough to allow RREP to come back. A routing table entry maintaining a forward route is purged if not used for an *active\_route\_timeout* interval. If no data is being sent using a particular routing table entry, that entry will be deleted from the routing table (even if the route may actually still be valid).





#### 14.14 Link Failure Detection

Neighboring nodes periodically exchange *Hello* or *Keep-Alive* messages. Absence of any *Hello* message for some time is used as an indication of link failure. Alternatively, failure to receive several link-layer ACK for a transmitted packet may be used as an indication of link failure. A node *MAY* offer connectivity information by broadcasting local *Hello* messages. A node *SHOULD* only use *Hello* messages if it is part of an active route.

#### 14.15 Summary of AODV

- No source routing.
- Based on routing tables.
- Use of sequence numbers to prevent loops.
- At most one route per destination maintained at each node:
  - only the freshest one is maintained (via destination seq. no.),
  - stale route problem is less severe,
  - after link break, all routes using the failed link are erased.
- Unused routes expire even if valid.

# 15 Network Simulation

## 15.1 Performance Evaluation

Example: does that new scheduling algorithm (or TCP congestion control algorithm) perform better than the previous ones?

- Analytical methods: set up an equation-based model of the (interesting aspects of the) system; solve it and get the results.
- Simulative methods: replicate the (interesting aspects of the) system in software; give it an input; measure its output.

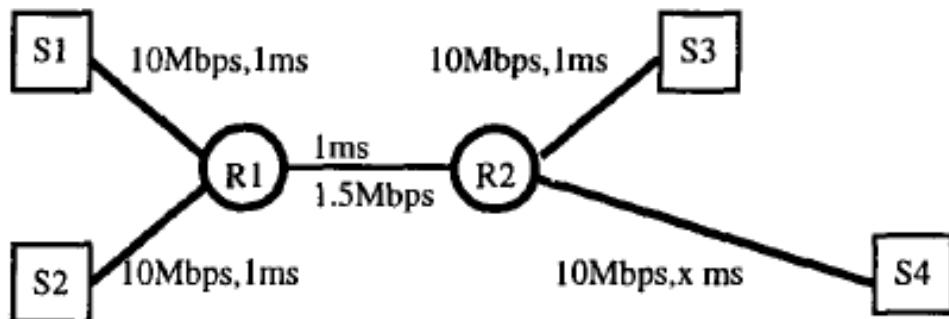
## 15.2 The Simulation Workflow

1. Define the *objectives*. For instance, assessing the effectiveness of an end-to-end congestion control algorithm for TCP.
2. Define the *performance indices*, which depend on our objectives. For instance, we would like to show that our congestion control algorithm protocol is better than existing ones, like Reno or Vegas. In our case, “better” means “it achieves a higher end-to-end throughput”.
3. Define the *factors*, i.e. the degrees of freedom of the system model, which might affect the performance. The factors can change depending on the accuracy of the system model implemented in the simulator. Some of the factors correspond to something that can actually be modified in the real life, while some others do not. In our example, the congestion window size can be adapted, but the number of nodes in the network or the RTT cannot.
4. Select the *simulation tool*. Straightforward factors affecting the choice are:
  - cost of the software;
  - expertise with a specific simulator (or class thereof);
  - simulator capabilities and/or amount of expected modifications needed to run simulations that fulfill our objectives;
  - analysis tools available for the simulator.
5. According to our needs, and depending on the simulator capabilities, it might be necessary to *implement or modify parts of the simulator*. Any coding should follow the best practices of programming. In particular, careful debugging should be performed. If the implementation is flawed, it can be very difficult (and frustrating) to distinguish between configuration errors and simulator bugs!
6. *Calibrate the simulator*, i.e. tune the simulation parameters consistently with the objectives. In our example, the bandwidth and latency of the links between nodes should be set according to our target application scenario: 5 minute is a sensible RTT only if you plan to talk with somebody on Mars... Some configuration parameters in common with all network simulations are:
  - the simulation duration;
  - the warm-up period duration (discussed later).
7. *Plan the simulation experiments*, which consists of two steps:
  - select the range of values of interest for each factor;
  - remove those factors whose impact on the defined performance indices is negligible.
 Difficult in general, usually done empirically. There are some known techniques, like  $2^k r!$  Analysis.
8. *Run simulations*. This step requires a variable time, mostly depending on the complexity of the simulations and the accuracy required. The hardware used to host the simulations has a role, too. For large simulation campaigns, it can be useful to run simulations in parallel over multiple machines:
  - Parallel Discrete-Event Simulation (PDES);
  - Multiple Replications In Parallel (MRIP).
9. *Collect and organize output data*. Very important for large simulation campaigns. It is especially useful to record the exact conditions under which the data have been obtained. With poor storage organization, those data are destined to become unrecoverable, because the complexity of distinguishing which data belonged to which simulations might become comparable to that of running the simulations again.

10. *Analyze results.* In this step the performance indices have to be extracted from the raw data and visualized, in combination with the possible values of the relevant factors. 2D and 3D graphs, tables, animations, etc. can be used to this purpose.
11. *Verify the correctness of the results obtained.* To this aim, experimental/simulation data obtained by others or analytical results can be used. If the simulation tool is publicly available and the results can be made available to the scientific community at large, then another option is peer-reviewing the results.

### 15.3 The Simulation Workflow - Example

TCP Reno vs. Vegas: Simulation scenario.



We show the throughput obtained with both Reno and Vegas (and their ratio) with varying buffer size of the switches connecting nodes.

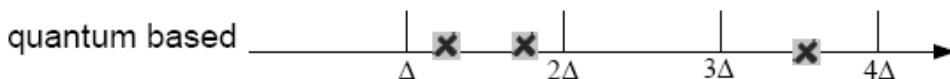
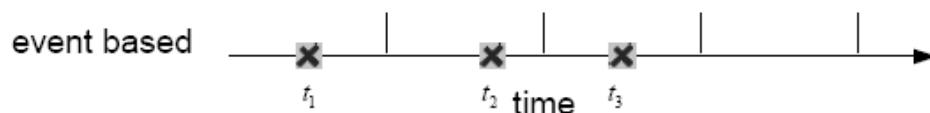
Buffer Size	ACK of Reno	ACK of Vegas	Reno/Vegas
4	13,010	24,308	0.535
7	16,434	20,903	0.786
10	22,091	15,365	1.438
15	25,397	12,051	2.107
25	30,798	6,621	4.652
50	34,443	2,936	11.73

### 15.4 Events

A simulation run takes some time to execute. The simulator runs while *real time* flows. During the simulation, the system evolves in *simulated time*. Simulated time can be *discrete* or *continuous*. There is – in general – no constant relationship between the flow of simulated time and the flow of real time.

*Event:* everything that changes the state of a system. E.g., the arrival/departure of a packet at a node.

*Event based vs. quantum based* simulators:



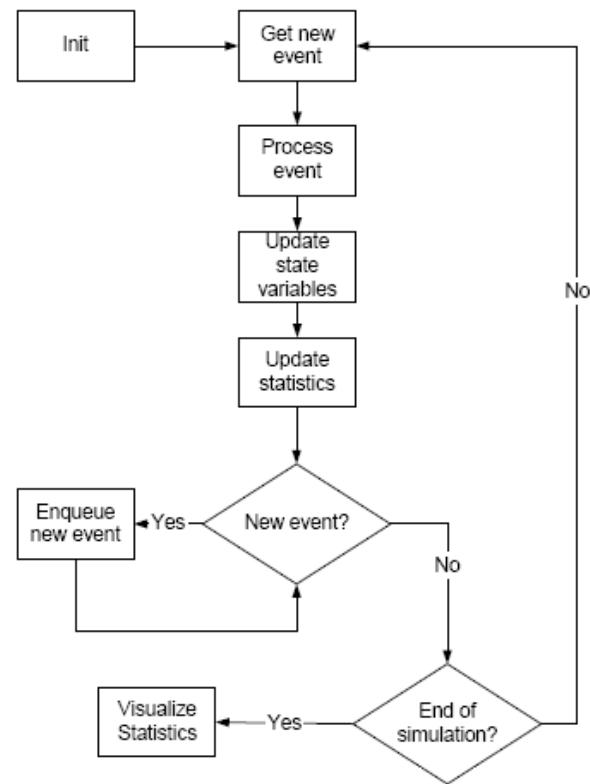
An event is some kind of data structure, containing a *firing time* field. An *event queue* keeps a list of the events, sorted by firing time, supporting the following operations:

- extraction of the nearest future event;

- ordered insertion of a future event.

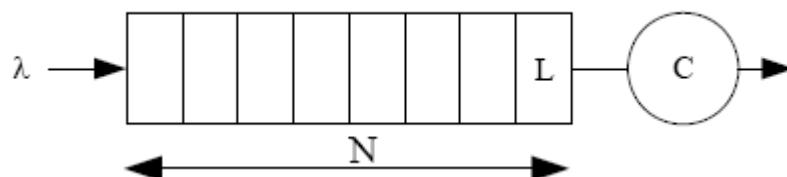
Components of a simulator are:

- Data Structures**
  - State Variables
  - Clock
  - Event Queue
  - Statistic gathering
- Functions**
  - Initialization
  - Event Scheduler
  - Event Handlers
  - Statistic computation and visualization



### 15.5A Simple Example

Problem Definition:



You want to know the average delay of a source in a single queue system:

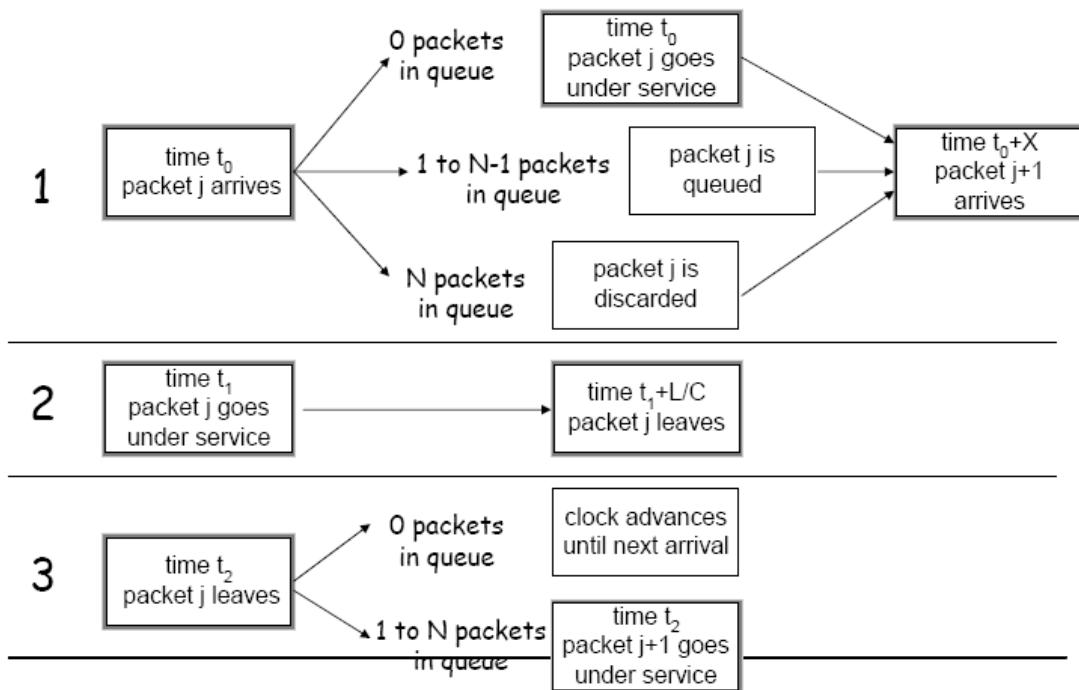
- packets length is constant and equal to  $L$ ;
- link bandwidth is  $C$ ;
- the queue contains at most  $N$  packets;
- the processing time of the scheduler is null;
- interarrival time of packets is an exponential variable,  $E[.] = 1/\lambda$ ;

You build up a simulator of the system.

Events in your system are:

1. A packet arrives at the queue.
2. A packet goes under service (starts transmission).
3. A packet leaves the system.
4. (implicit) simulation ends.

Event Handling:



We measure the following quantity:

$$D = \frac{\sum_{i=1..m} Delay_i}{m}$$

where:

$$Delay_i = \text{Departure\_time}_i - \text{Arrival\_time}_i$$

In order to compute it you can:

- Write down all the events on a trace file; then parse it off-line and make the computations.  
Or:
- Associate an arrival time variable to each queued packet, compute the delay of the packet at departure time and maintain an ongoing average on line. Then, when the simulation ends, print the result.

Trace File - For each dispatched event there is a line in the log file:

```

time 0: arrival of packet 1
time 0: packet 1 goes under service
time 7: arrival of packet 2
time 10: packet 1 leaves
time 10: packet 2 goes under service
time 12: arrival of packet 3
time 16: arrival of packet 4
time 20: packet 2 leaves
time 20: packet 3 goes under service
time 30: packet 3 leaves
time 30: packet 4 goes under service
time 40: packet 4 leaves
time 50: arrival of packet 5
time 50: packet 5 goes under service
time 60: packet 5 leaves

```

$$D = \frac{((10-0)+(20-7)+(30-12)+(40-16)+(60-50))}{5} = 13$$

## 15.6 Metrics

There are different types of metrics, depending on how you analyze the data sampled during the simulation.

Rate: given N samples  $\{x_1, x_2, \dots, x_N\}$  the value of a *rate metric* is:

$$y = \frac{\sum_i x_i}{\tau}$$

where  $\tau$  is the duration of the simulation.

Example:  $x_i$  is the size of the  $i$ -th packet (in bytes) served by a router,  $y$  is the throughput of the router, in bytes/s.

Discrete r. v.: given  $N$  samples  $\{x_1, x_2, \dots, x_N\}$  the value of a *discrete r. v. metric* is:

$$y = \frac{\sum_i x_i}{N}$$

Note that  $y$  does not depend on the time when the samples were collected.

Example:  $x_i$  is the delay of the  $i$ -th packet received by a host for an application,  $y$  is the average delay experienced by the application.

Continuous r. v.: Given  $N$  samples  $\{(x_1, T_1), (x_2, T_2), \dots, (x_N, T_N)\}$  the value of a *continuous r. v. metric* is:

$$y = \frac{\sum_{i=1}^{N-1} x_i \cdot (T_{i+1} - T_i)}{\sum_{i=1}^{N-1} (T_{i+1} - T_i)} = \frac{\sum_{i=1}^{N-1} x_i \cdot (T_{i+1} - T_i)}{T_N - T_1}$$

where  $T_i$  is the time when sample  $x_i$  is collected.

Example:  $x_i$  is the size of the congestion window of a TCP connection changed at time  $T_i$ ,  $y$  is its average size.

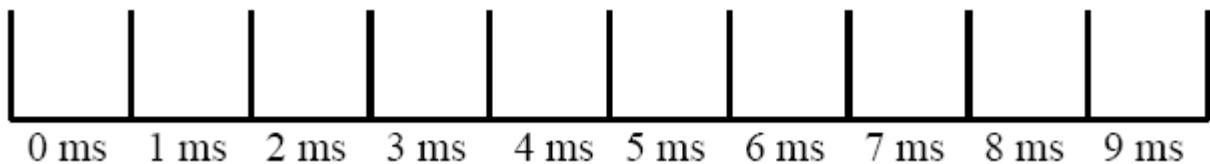
Distribution: There are cases when you are interested in estimating the distribution, instead of the average value of a metric. This is useful if you want to answer questions like:

- what is the probability that the occupancy of my buffer grows over 100 kB?
- what is the 99th percentile of delay? (which means “what is the largest delay obtained, provided that I am not interested in those delays that only happened in 1% of the cases?”)

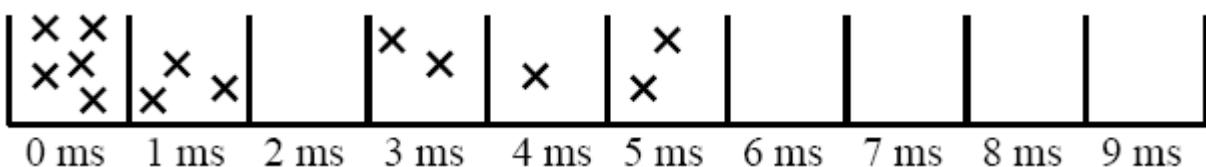
Estimating distributions is a very complex task, unless you can tolerate a few approximations (we can).

Decide which is the range of interest of your metric. For instance, if you are interested in estimating the delay of MAC frames in an Ethernet network, then the range [0 ms, 10 ms] should be safe enough. Then decide what is the granularity that you can accept. Assume we select 1 ms.

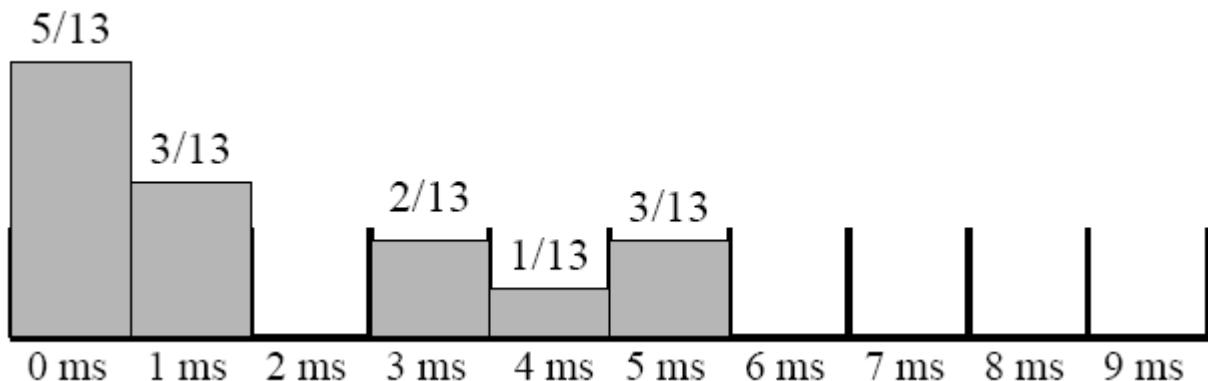
Divide your range of interest in equal size bins:



Whenever a sample is collected during simulation, add 1 to the corresponding bin. Keep trace of samples overflowing the range, if any:



At the end of the simulation, we can then derive the *probability mass function (pmf, below)*, or the *cumulative distribution function (cdf)* or quantiles, if needed:



## 15.7 Implementation of an Event Queue

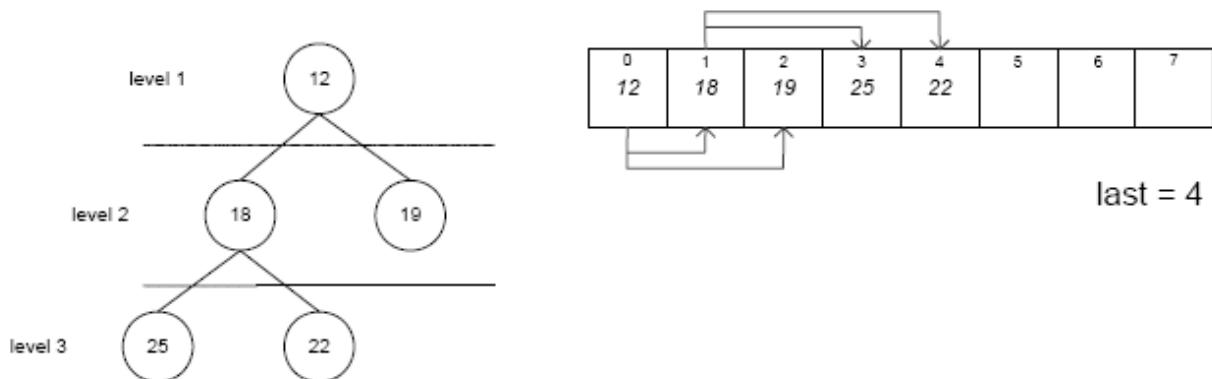
Data structure commonly employed are:

- Min heap tree
- Splay tree
- Calendar queue

**Min heap tree:** A *min heap tree* is a binary tree, in which:

- the content of a parent node is less than or equal to the content of both its children, and
- each level is filled from left to right. Level  $n+1$  cannot be populated unless level  $n$  is completely full. Thus, depth is  $\sim \log_2(\text{nodes})$ .

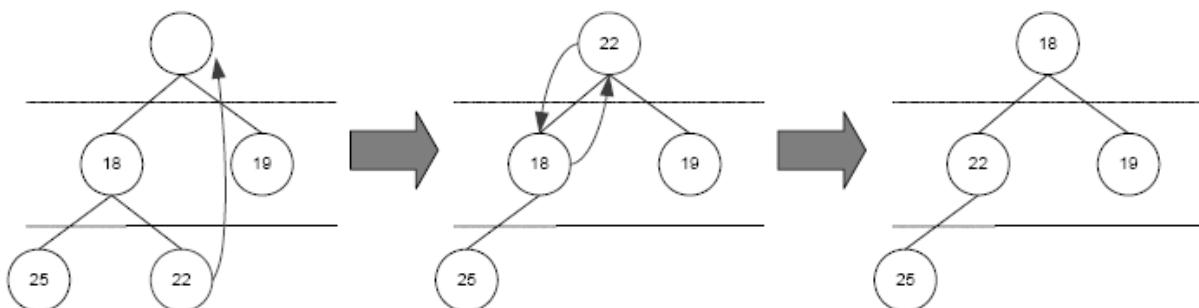
It can be implemented with an array and a counter:



Node  $j$ 's children are at  $2j+1$ ,  $2j+2$ . Node  $j$ 's parent is at  $\text{floor}((j-1)/2)$ .

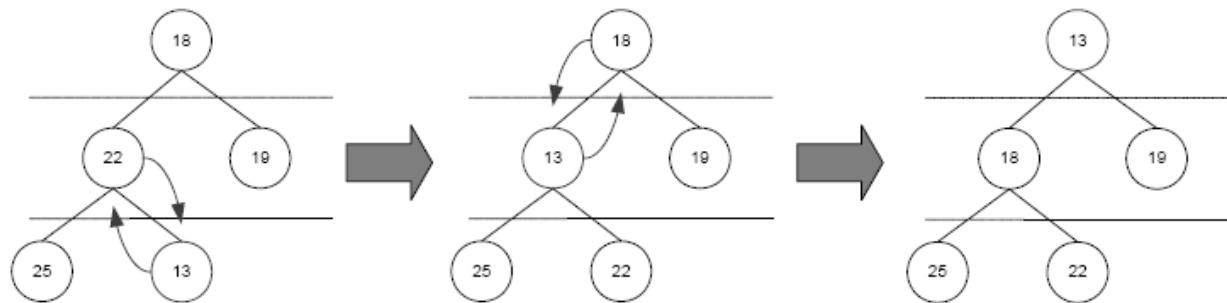
To extract the min element:

- Extract root node (min element)
- “Reheapification”: recover the min heap property
  - Select the “last” node as the new root
  - Swap it with the min between the two child nodes of the root if greater
  - Recursively delve into the tree, and do the same with each sub-tree

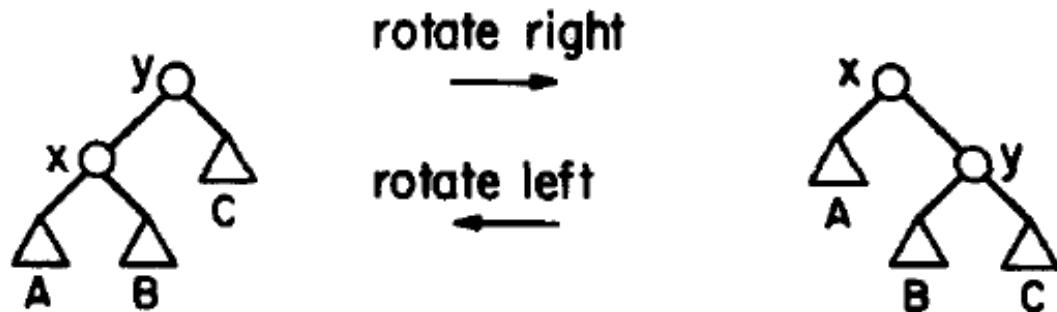


To insert a new element:

- Insert at the first free place (last level, left to right)
- Reheapification:
  - Compare (and, in case, swap) the newly inserted node and its parent
  - Go on until the root

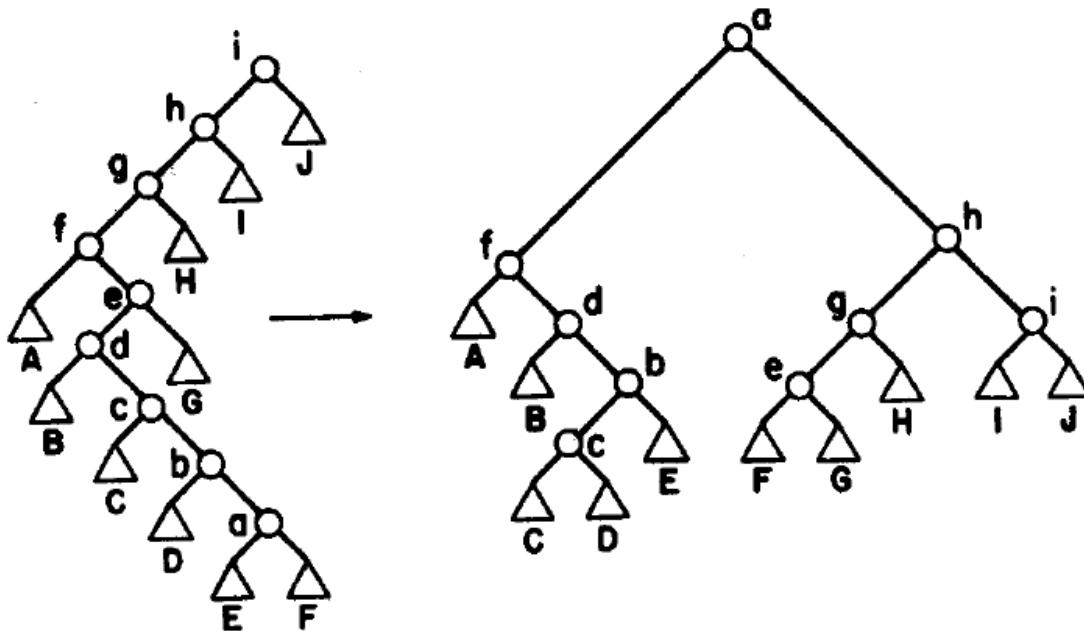


**Splay tree:** A *splay tree* is a binary search tree (not balanced). It's a self-adjusting data structure, based on *splaying*: rotating pairs of nodes whenever the tree is accessed in a heuristic manner.



Splay trees have known performance bounds.

Example of root splaying:

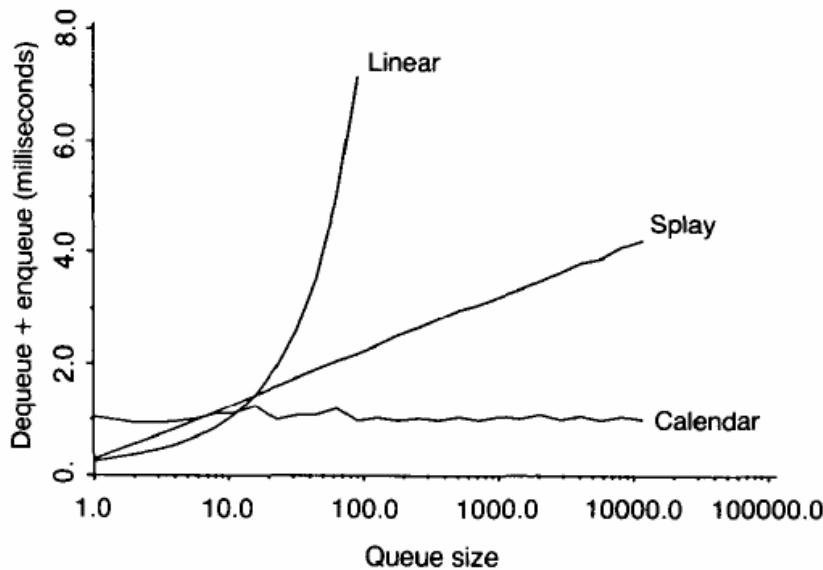


**Calendar queue:** A *calendar queue* is an array of sorted lists. Each entry is called a *bucket*. All buckets have equal *width*. The event expiring at time  $t$  is placed into the bucket ( $(t/\text{width}) \% \text{n\_buckets}$ ). The events into each bucket are sorted in time. You can see a bucket as a page of a calendar.

On average, you only need to sort a small subset of events ( $<< N$ ). It might take up to  $O(K)$  operations to find the next event to be scheduled in a worst case. Inefficient if the width and  $n_{\text{buckets}}$  are chosen poorly. The calendar should be dynamically adjusted, for instance:

- Increase the calendar when the number of events is twice the number of buckets.
- Shrink the calendar when the number of events is half the number of buckets.

R. Brown. Calendar Queues: A Fast  $O(1)$  Priority Queue Implementation for the Simulation Event Set Problem. 1988:



## 15.8 Random Number Generation

- *Trace-driven* simulation: Input is taken from a trace obtained by measuring the real system (or something quite close to it).
- *Self-driven* simulation: Input is made of “artificially generated” traffic. In order to generate artificial traffic we need *random numbers*.

*Problem:* I need a random variable with a given distribution (e.g. exponential, normal, etc.) in order to simulate the packet interarrival times.

*Solution:* Generate a “random” variable uniformly distributed within [0,1) (random number generation). Transform that variable in order to meet the desired criteria (random variate generation).

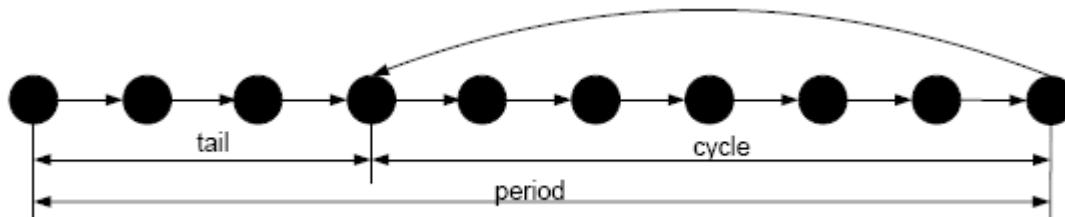
In the most general case:

$$x_n = f(x_{n-1}, x_{n-2}, \dots)$$

- $f$  is a deterministic function.
- $x_0$  is called *initial seed* of a sequence.
- Given  $f$  and  $x_0$ , you always obtain the same set of numbers.

*Pseudo-random numbers* “look like” random numbers. The same sequence can be repeated if needed.

How to choose a good  $f$ :



- Efficiently computable
- The period should be large
- Successive values should be independent and uniformly distributed

Beware of common folklore: “A complex set of operations leads to random results”.

Linear-Congruential Generators (LCGs):

$$x_n = (a \cdot x_{n-1} + b) \bmod m$$

where  $0 < m, a < m, b < m, x_0 < m$

Random numbers appearing to be IID U(0,1) are:

$$\frac{x_0}{m}, \frac{x_1}{m}, \dots, \frac{x_P}{m}$$

The period  $P$  is at most  $m$ , but can be smaller depending on the other parameters. If the period is  $m$ , the LCG is said to have *full period*.

It is important to choose a good seed for the generator. “Good” depends on the algorithm used for the generation and on the parameters chosen. Generally advisable to avoid:

- zero,

- even numbers,
- unpredictable numbers, such as current clock value or PID.

If you need different independent streams, use seeds that you know to be far away in the period.

Problems with LCG are:

1. Every  $x_i$  can be deterministically computed as:

$$x_i = \left( a^i x_0 + \frac{b(a^i - 1)}{a-1} \right) \bmod m$$

2. The  $x_i$ 's can only take rational values 0,  $1/m$ ,  $2/m$ , ...,  $P/m$ . In general, they take only a fraction of those values, depending on the parameters: no possibility of getting a value of  $0.8/m$ , which should however occur with some non-zero probability.

Some other generators are:

- quadratic congruential generators:

$$x_i = (ax_{i-1}^2 + b_{i-1} + c) \bmod m$$

- multiplicative recursive generators:

$$x_i = (a_1 x_{i-1} + a_2 x_{i-2} + \dots + a_q x_{i-q}) \bmod m$$

- Tausworthe generators, which operate directly on bits. In this case then next bit is selected as random, that is the modulus  $m$  is 2. Very large periods can be obtained with these generators, which are related to cryptographic methods.

## 15.9 Random Variate Generation

After we have obtained a stream of random numbers, we can sample values from random distributions. Unfortunately, there is not a general approach that can be followed for any distribution. Some methods are *inverse transform* or *convolution*.

**Inverse transform:** Suppose I need a sample  $x$  from a r.v. with a given distribution  $F(x)$  that is continuous and strictly increasing. Let  $F^{-1}$  denote the inverse of function  $F$ . Then the method of the inverse transform is:

1. Generate  $U \sim U(0, 1)$
2. Return  $X = F^{-1}(U)$

The method can be used also to generate discrete r.v.'s.

For instance, suppose we want samples from an exponentially distributed r.v. with mean  $1/\lambda$ , i.e. the probability density function is:

$$f(x) = \lambda \cdot e^{-\lambda \cdot x}$$

Assume we have a r.v.  $y$  uniformly distributed within  $[0,1]$ . First, we compute the CDF:

$$F(x) = 1 - e^{-\lambda \cdot x}$$

Then, we invert the formula and obtain:

$$x = -\frac{1}{\lambda} \ln(1 - y)$$

One obvious practical problem with the method of the inverse transform is that sometimes we are not able to write a formula for  $F^{-1}$ . For instance, the Normal distribution  $N(\mu, \sigma^2)$  has the following probability density but no closed form for the probability distribution:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

**Convolution:** If the r.v.  $X$  can be represented as a sum of other r.v.'s, then the method of convolution can be used. Assume:  $X = Y_1 + Y_2 + \dots + Y_m$ . Then:

1. Generate  $Y_1, Y_2, \dots, Y_m$ ,
2. Return  $X = Y_1 + Y_2 + \dots + Y_m$ .

**Normal distribution:** (*Central limit theorem*) The sample mean of a sufficiently large number of IID r.v.'s with finite mean and variance is approximately normally distributed. In principle, you can thus use the convolution method to draw samples from a Normal distribution by summing any random variables (e.g. uniform r.v.'s). A more efficient algorithm to generate pairs of samples from a Normal distribution is the Box-Muller method:

1. Generate  $U_1$  and  $U_2$  from IID  $U(0,1)$
2. The following samples are  $N(0,1)$ -distributed:

$$X'_1 = \sqrt{-2\ln U_1} \cos(2\pi i U_2)$$

$$X'_2 = \sqrt{-2\ln U_1} \sin(2\pi i U_2)$$

3. Then  $X_i$  is  $N(\mu, \sigma^2)$ -distributed:

$$X_i = \mu + \sigma X'_i, \quad i=1,2$$

## 15.10 Steady-State Analysis

Experimental evidence show that systems often reach a *steady state* after some *transient* (*warm-up*). Usually, steady-state properties of a system are investigated by simulation. To estimate the length of the transient:

- have some knowledge of the system (e.g., time constants, etc.), or
- compute a moving average of some target steady state quantity, and assume that the system is in the steady state when the moving average stabilizes.

Start tracing the target quantity after the transient has expired.

*Question:* How long should a simulation be? Long enough to collect a statistically meaningful number of

events (in the steady state). Example: Suppose you simulate a link with a given error model, and you want to estimate the packet loss probability. From some a-priori knowledge, you expect the packet loss probability to be  $\sim O(10^{-4})$ . You need to simulate the transmission of more than  $10^4$  packets in order to obtain statistically meaningful results.

## 15.11 Reliability of the Output

Even though we remove the initial bias, still the dynamics of the simulated system depends on the initial conditions selected, e.g. the initial seeds for random number generators. To obtain statistically confident output, we thus have to obtain several *observations* of the phenomenon that we are investigating (= many samples of the same metric for the scenario scenario). Such observations can be seen as samples from IID r.v.'s.

- *Independent replications:* Perform N replication of the simulation with independent initial conditions. Need to discard N initial transients.
- *Batch means:* Perform one “long” simulation, and slice it into N intervals. Assume each interval is independent of the others (which is seldom true). Only discard one transient.

Some systems have “regeneration states”, i.e. states starting from which the future behavior of the system is independent from its past history. In the example, a regeneration state occurs whenever the system empties. Regeneration states can be used to slice up intervals (similar to the batch method). Problem: regeneration states may be too rare (and thus simulations will be overly long), or too frequent (too few events for each interval). You need a considerable insight of your system.

Now, we have n IID samples for random variable D. The *sample mean* is an unbiased estimator of the mean of D, (the latter is  $\mu = E[D]$ ).

$$\bar{D}(n) = \frac{\sum_{i=1}^n D_i}{n}$$

Unfortunately, we have no idea “how close” this value is to  $\mu$ .

## 15.12 Confidence Intervals

I cannot say for sure that the average delay of the system is  $\bar{D}(n)$  (that would require an infinite number n of IID samples). However, I can estimate that the average delay  $\mu$  is within

$$[\bar{D}(n) - d, \bar{D}(n) + d]$$

with probability  $(1-\alpha)\%$ . The above interval is called  $(1-\alpha)\%$  *confidence interval* for the average delay.

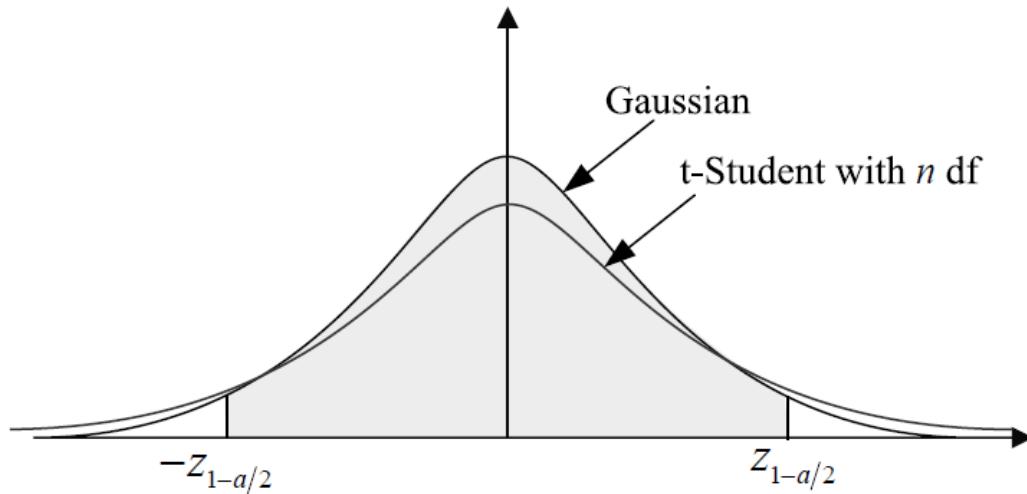
Let  $S(n)$  be the *sample variance* of D:

$$S(n) = \frac{\sum_{i=1}^n [D_i - \bar{D}(n)]^2}{n-1}$$

The confidence interval is:

$$\left[ \bar{D}(n) - t_{n-1,1-\alpha/2} \sqrt{\frac{S(n)}{n}}, \bar{D}(n) + t_{n-1,1-\alpha/2} \sqrt{\frac{S(n)}{n}} \right]$$

where  $t_{n-1,1-\alpha/2}$  come from the *t-Student distribution* function, whose values can be found in tables.



### 15.13 The Network Simulator

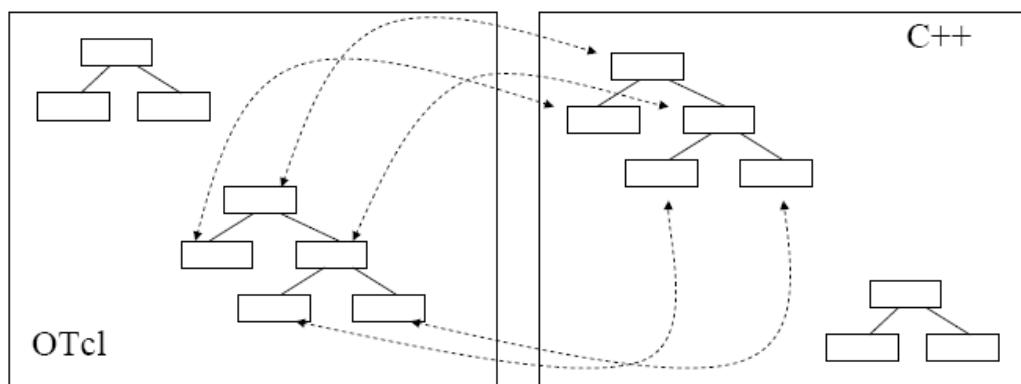
*Ns* is a discrete event simulator targeted at networking research. *Ns* provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Free project started in 1989: <http://www.isi.edu/nsnam/ns/>. *Ns* is employed in both research and production environments. *Ns* can be built and used under Linux or Cygwin or Mac OS X.

*Ns* is written in two languages:

- C++
  - Faster (compiled)
  - Greater flexibility
  - Support for specialization of network elements
- OTcl (*Object-oriented Tcl*)
  - Slower (interpreted)
  - Quick setting up of simulation scenarios
  - Configuration easier to manage

### 15.14 Double Class Hierarchy

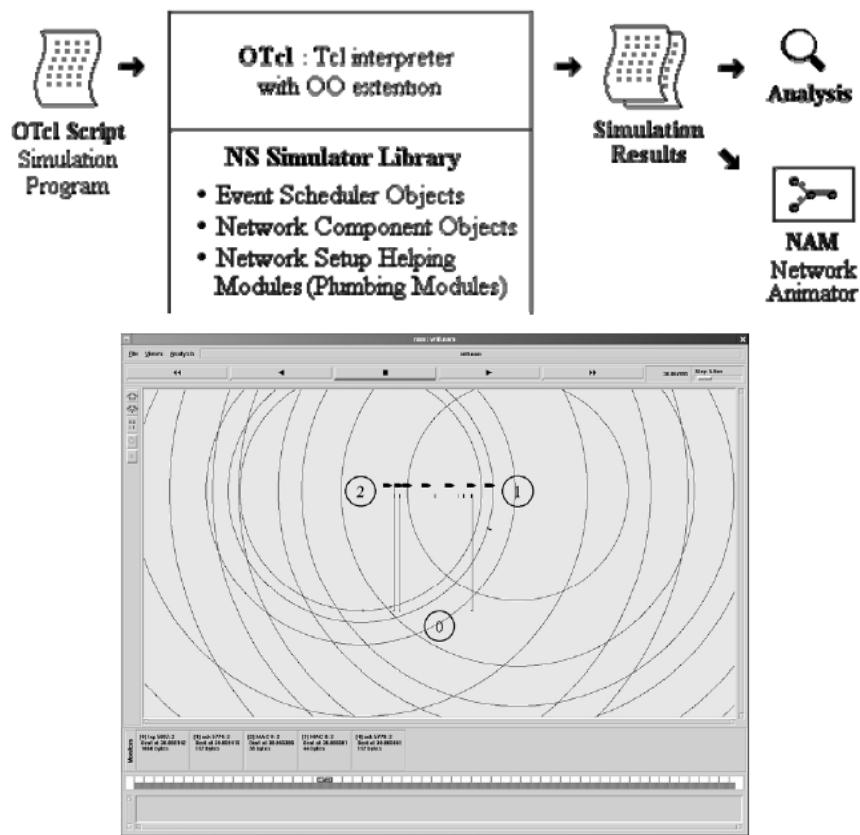
OTcl is used for simulation control and setup of network elements. Each class in C++ may represent a network element (e.g. a link, a node, a traffic generator). Most C++ classes have an OTcl counterpart. C++ variables that need be controlled from the user are made available through OTcl (*binding*). A C++ class may define *commands*, through which its methods are invoked from OTcl.



### 15.15 Simulating with NS

Ns-2 users interact with the simulator only by means of OTcl scripts. Ns-2 interprets an OTcl

script, executes the simulation (doing most of the processing in C++), and returns results under the form of a *trace file*. Traces can be analyzed by using ad-hoc instruments (e.g. gawk, perl, home-made C++ code) and plotted. A *network animator* (*NAM*) can be used for off-line animation of a simulation run.



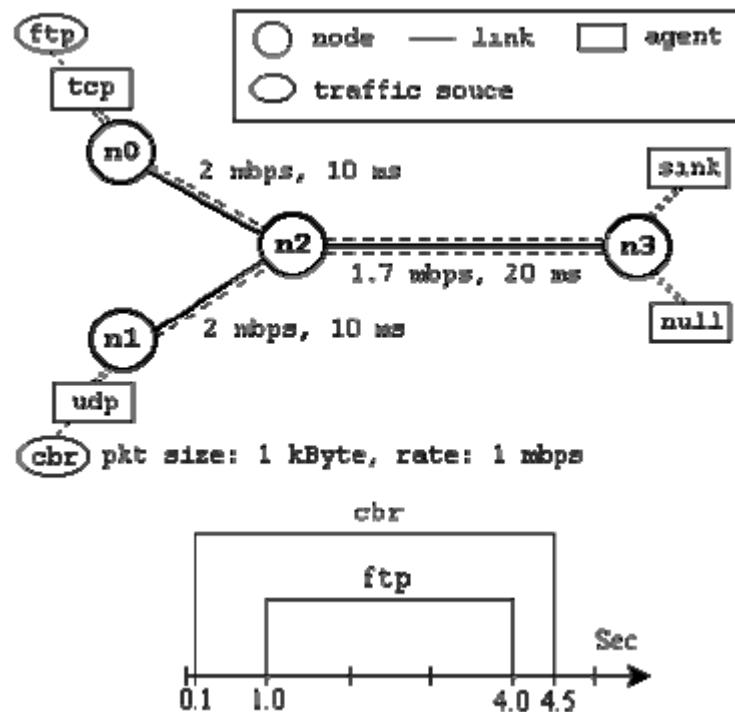
OTcl is a complex language, with lots of built-in functions:

```
# Writing a procedure called "test"
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {($k < 10)} {incr k} {
        if {$k < 5} {
            puts "k < 5, pow = [expr pow($d, $k)]"
        } else {
            puts "k >= 5, mod = [expr $d % $k]"
        }
    }
}

# Calling the 'test' procedure created above
test
```

```
k < 5, pow = 1.0
k < 5, pow = 1120.0
k < 5, pow = 1254400.0
k < 5, pow = 1404928000.0
k < 5, pow = 1573519360000.0
k >= 5, mod = 0
k >= 5, mod = 4
k >= 5, mod = 0
k >= 5, mod = 0
k >= 5, mod = 4
```

## 15.16 A First Simulation Scenario



```

#Create a simulator object
set ns [new Simulator]
#Open the trace file
set nf [open out.txt w]
$ns trace-all $nf

#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Exit
    exit 0
}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
#Setup a UDP connection
$ns attach-agent $n1 $udp set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
$cbr attach-agent $udp
$cbr set packetSize_ 1000
$cbr set rate_ 1Mb

#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run

```

## 15.17 Output Trace

The output of the simulation is stored in *out.txt* as a sequence of 12-field event records, ordered by firing time:

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id
-------	------	-----------	---------	----------	----------	-------	-----	----------	----------	---------	--------

```
r : receive (at to_node)
+ : enqueue (at queue)
- : dequeue (at queue)
d : drop (at queue)

src_addr : node.port (3.0)
dst_addr : node.port (0.0)

r 1.3556 3 2 ack 40 ----- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ----- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ----- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ----- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ----- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ----- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ----- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ----- 2 1.0 3.1 157 207
```

Writing down trace files on the disk might be the true speed bottleneck in ns simulations. Instead of tracing all the events in the system, one might trace only events on single links:

```
set nf [open out_PTTSD_overload.txt w]
set nj [$ns node]
set nk [$ns node]
$ns duplex-link $nj $nk 1Mb 10ms PTTSD
$ns trace-queue $nj $nk $nf
```

There are other ways for limiting the amount of data written on the disk.

The easiest way to parse an output trace is to write *gawk* (*awk*) programs. *gawk* programs are a sequence of lines of the type.

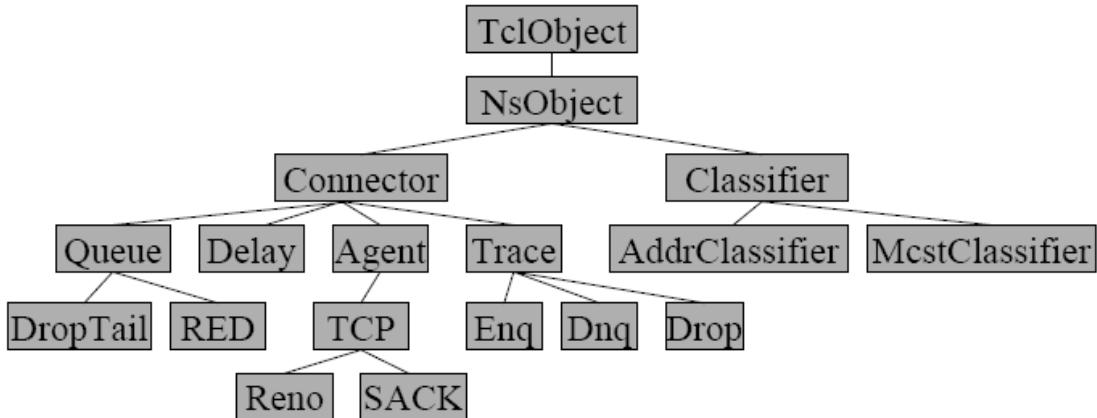
```
rule {action;}
gawk '$1=="+" && $8=="$1' {start[$12]=$2;}
      $1=="r" && $2>'$2' && start[$12]>'$2' && $8=='$1' {print $2-start[$12];}'
```

delay flowid start\_time < out.txt > delays.txt

It is utterly easy to build a library of common parsing routines to be performed on ns traces.

## 15.18 Common Objects

Class Hierarchy (Partial View):

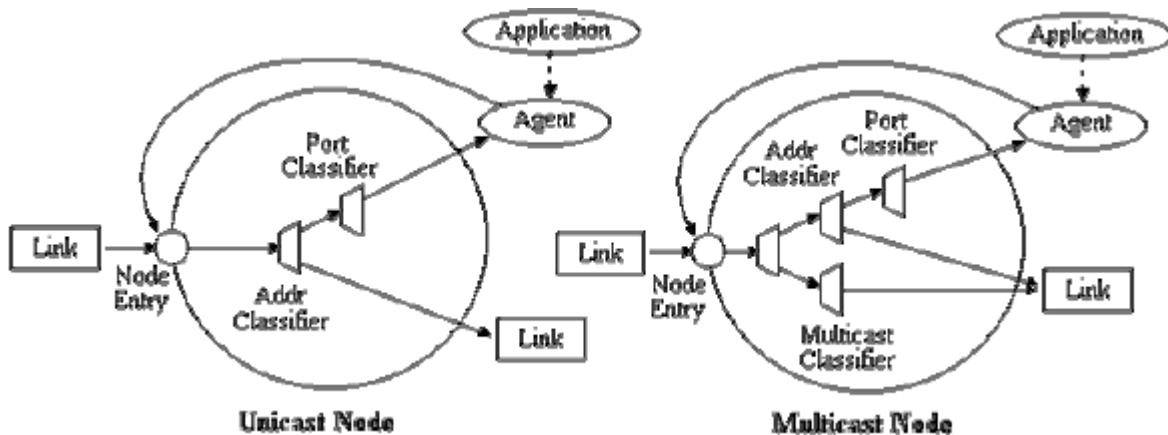


**Nodes:** A *node* is a compound object, which includes:

- An address classifier, performing routing functions
- A port classifier
- Agents attached to the node

```
set nj [$ns node]
```

```
set nk [$ns node]
```

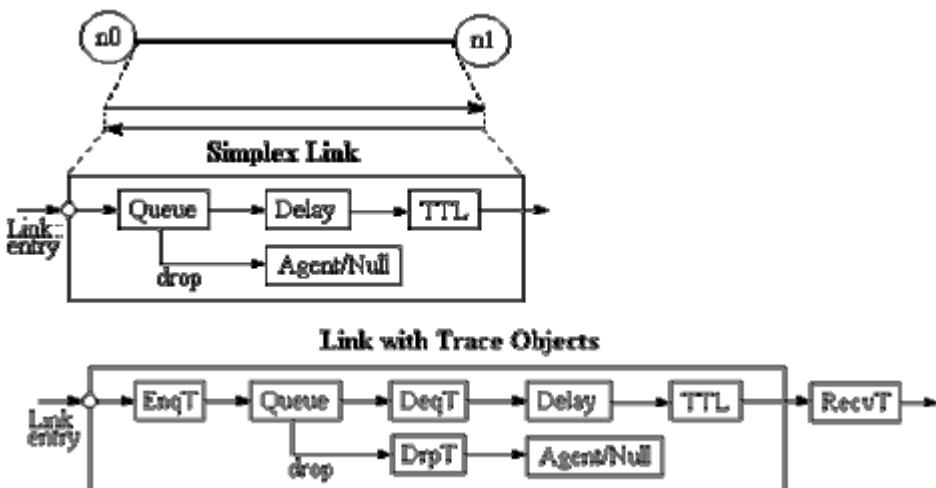


**Links:** A *link* is a compound object including:

- A scheduler (called a queue in ns)
- A delay, emulating the transmission and propagation delay
- A TTL module, which decreases the TTL field

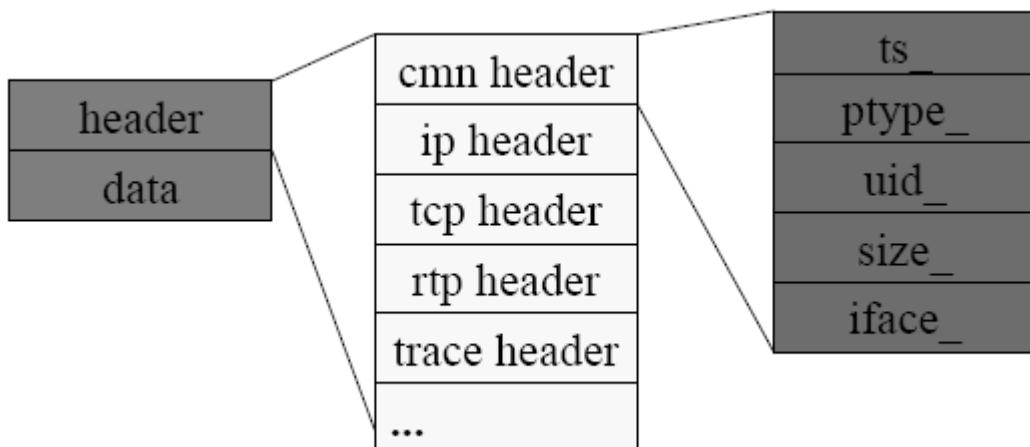
```
$ns duplex-link $nj $nk 1Mb 10ms PTTSD
```

```
$ns trace-queue $nj $nk $nf
```



**Packets:** Packets are data structures containing:

- A common header containing data displayed in the trace file
- Protocol-specific headers
- Typically, no data

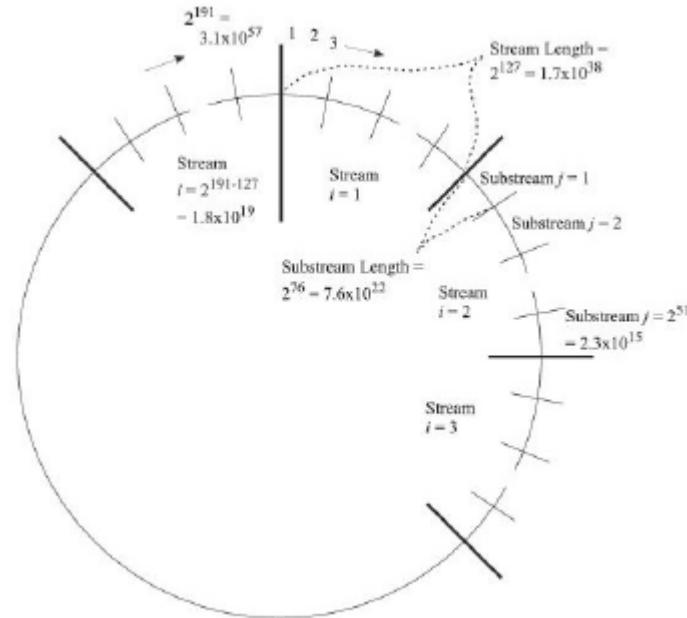


**Agents:** Agents usually implement protocols entities. They send and receive packets of a given

type. Usually, they act on behalf of an application (e.g., a traffic generator), which actually requires packet transmissions and processes received packets. They have two important methods: *Send* and *Receive*.

## 15.19 Random Number Generation on Ns

The *RNG* class implements the combined multiple recursive generator *MRG32k3a* proposed by L'Ecuyer (replacing the minimal standard multiplicative linear congruential generator by Park and Miller). The *MRG32k3a* generator has many independent streams of random numbers, each of which consists of many substreams. Each substream has a very large period.



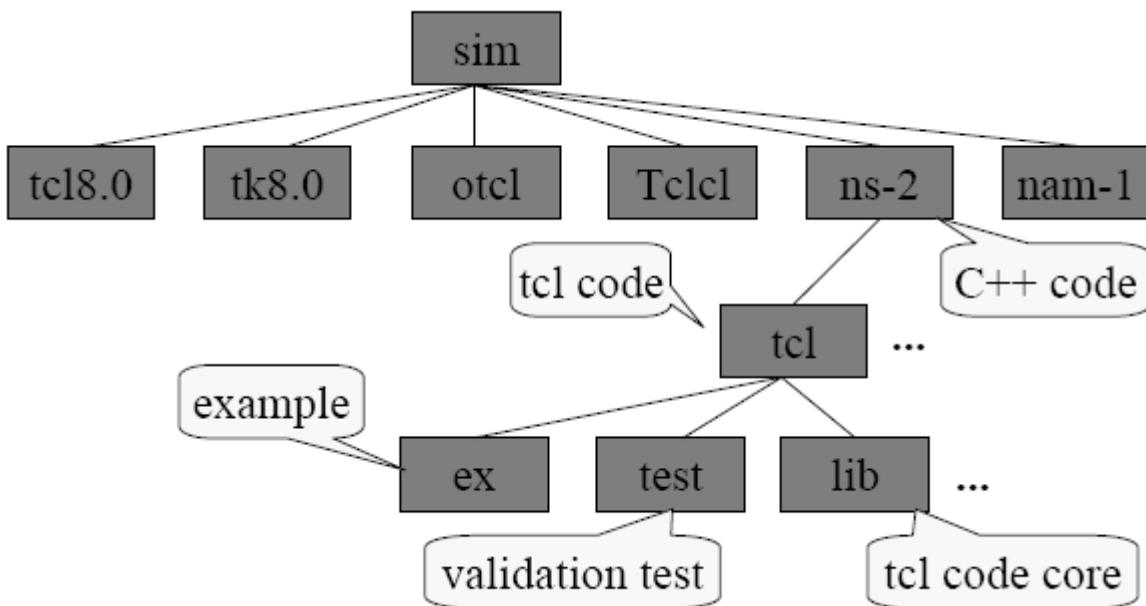
## 15.20 Configuring and Modifying ns-2

What commands and bound variables are available through OTcl?

- You look it up in the manual;
- You search the C++ code.

Default values for all bound variables must be stored in *ns/tcl/lib/ns-default.tcl*.

Where is what in the ns-2 directory tree:



To modify ns-2:

- Compile ns-2 from scratch (*./configure && make && make depend*);
- After having modified the existing code: *make*.

- After having added some new code: add object files to Makefile.in, recompile from scratch. If you have to add new functionalities to existing objects, consider using *inheritance*.

## 15.21 Comprehensive ns2measure example

*Problem:* we want to study the behavior of TCP/Reno on a single link, with no packet loss, with varying:

- *sender buffer size*: { 10, 20, 50, 100, 200 }
- *link rate*: { 1 Mb/s, 2 Mb/s, 3 Mb/s, 4 Mb/s, 5 Mb/s }
- *link latency*: { 0.1 ms, 1 ms, 10 ms, 100 ms }

The metric of interest is the *throughput*. The following steps are needed to carry out the task:

1. Setup the OTcl ns-2 scenario in a file *tcp.tcl*;
2. Select the simulation duration and warm-up period;
3. Decide how many replications per simulation;
4. Setup a script that runs the batch of simulation and calls the *recover* utility to gather data from the simulation output;
5. Extract results and produce plots.

```
rate="1000000 2000000 3000000 4000000 5000000"
delay="0.0001 0.001 0.01 0.1"
queue="10 20 50 100 200"
for (( i = 1 ; i <= 10 ; i++ )) ; do
    for r in $rate ; do
        for d in $delay ; do
            for q in $queue ; do
                savefile= savefile/save-$r-$d-$q
                ./ns tcp.tcl -rate $r -delay $d -queue $q -run $i -out $$
                cat $$ >> $savefile
                recover -d -s output/ -n $r,$d,$q $savefile
            done
        done
    done
done
```

The shell commands above produce:

- a set of files *savefile/save-\**, one for each set of independent replications of a simulation, containing the simulation output data in binary format;
- a set of files *output/\*.dat*, one for each metric, containing the average measures and confidence intervals in ASCII format.

A few lines from *output/e2e\_tpt.dat*:

```
1000000,0.0001,100,1,125008,0
1000000,0.0001,200,1,110289,23.9565
1000000,0.001,10,1,125008,0
1000000,0.001,20,1,125008,0
1000000,0.001,50,1,125008,0
1000000,0.001,100,1,125001,15.6832
```

What is the throughput vs. queue size with 3 Mb/s link rate and 10 ms link latency?

```
% extract.pl 3000000 0.1 X 1 < output/e2e_tpt.dat
10 249302 47.0496
20 177694 172.515
50 363099 0
100 370032 0
200 324189 76.8317
```

What is the throughput vs. link rate with buffer size equal to 100 packets and 10 ms link latency?

```
% extract.pl X 0.01 100 1 < output/e2e_tpt.dat
1000000 125008 0
2000000 249995 23.9565
3000000 374969 20.9109
4000000 499977 20.9109
5000000 624971 0
```

What is the throughput vs. link latency with 2 Mb/s link rate and buffer size equal to 100

packets?

```
% extract.pl 2000000 X 100 1 < output/e2e_tpt.dat
0.0001 249967 23.9565
0.001 250016 0
0.01 249995 23.9565
0.1 231851 52.2773
```

## 15.22 References

- A. M. Law and W. D. Kelton. “*Simulation Modeling and Analysis*”. McGraw-Hill, New York, third edition, 2000.
- Examples/tutorials:  
<http://nile.wpi.edu/NS>  
<http://www.isi.edu/nsnam/ns/tutorial/index.html>
- Ns-2 manual:  
<http://www.isi.edu/nsnam/ns/ns-documentation.html>
- How to install ns under cygwin:  
<http://www.isi.edu/nsnam/ns/ns-cygwin.html>
- Our ns-2 software:  
<http://info.iet.unipi.it/~cng/software.htm>

## English-Italian Glossary

Il glossario contiene la traduzione di alcuni termini (tecnicici e non) presenti nel testo. L'obiettivo non è solo la comprensione del testo ma anche l'esposizione con proprietà di linguaggio, dato che l'esame orale viene tenuto in lingua italiana.

English	Italiano
To bind (pp “Bound”)	Legare (pp “Legato”)
Burst	Scoppio
Charter	Carta (documento)
Compound (object)	(Oggetto) composto
Comprised (of)	Costituito (di)
To conceive	Concepire
To constrain (the connectivity)	Vincolare (la connettività)
Constraint	Vincolo
To cut over (to a route)	Cambiare direzione (verso una rotta)
To delve (into a tree)	Frugare (dentro un albero)
To digitize (data)	Digitalizzare (dati)
To encompass (a feature)	Includere (una funzionalità)
Extent (of an interference)	Portata (di un'interferenza)
Flavor (of WMN)	Sapore (cioè tipo) (di WMN)
Forwarding treatment	Trattamento di inoltro
From scratch	Da zero
To impinge on (an object)	Incidere (su un oggetto)
To internetwork	Formare una inter-rete
To jeopardize (performance guarantees)	Mettere in pericolo (garanzie sulla prestazione)
Large scale deployment	Utilizzo su larga scala
Leaky Bucket	Secchio che perde
Loss (of data)	Perdita (di dati)
Multi-tiered	Multi-livello
To nest	Annidare
To outweigh	Aver maggior peso di
To pin down (a route)	Fissare (una rotta)
Policy	Politica
To process (a message)	Elaborare (un messaggio)
Provisioning	Mettere a disposizione
Reservation	Prenotazione
(Network) resilience	Resistenza (di una rete)
Scattering	Diffusione
Seamlessly	Senza intoppi
Self-healingness	Che si ripara da solo
Slope	Pendenza
To span (a domain)	Attraversare (un dominio)

Stale (route problem)	(Problema delle rotte) obsolete
To steer (a packet)	Guidare (un pacchetto)
Straightforward	Chiaro
To tear down (a resource request) (pp “Torn down”)	Revocare (una richiesta di una risorsa) (pp “Revocato”)
To throttle back (a transfer of information)	Strozzare (un trasferimento di informazioni)
Token Bucket	Secchio a gettoni
Traffic conditioning	Condizionamento del traffico
Ubiquitous (wireless network)	(rete senza fili) onnipresente
To umbundle (a concept from another concept)	Disaccoppiare (un concetto da un altro concetto)
Utterly (easy)	Assolutamente (facile)
Worthlessness	Inutilità
To yield (results)	Produrre (dei risultati)

## **General Index**

1 - The Internet With Quality of Service (IP QoS).....	4
2 - Packet Scheduling Algorithms for QoS.....	13
3 - Integrated Services Model (IntServ).....	42
4 - Differentiated Services Model (DiffServ).....	76
5 - MultiProtocol Label Switching (MPLS).....	94
6 - Internet Traffic Engineering.....	134
7 - Virtual Private Networks (VPN).....	157
8 - 802.11e.....	189
9 - WiMAX (IEEE 802.16).....	201
10 - Long Term Evolution (LTE).....	219
11 - Wireless Mesh Networks.....	236
12 - Radio Propagation Basics.....	252
13 - Channel Assignment and Link Scheduling Strategies.....	266
14 - Ad-hoc On-demand Distance Vector (AODV).....	283
15 - Network Simulation.....	296