Reti di Telecomunicazioni
Internet Protocol versione 6

Introduzione

E' facile rendersi conto del motivo per cui il **protocollo IP**, nella sua versione attuale (**IPv4**) ha ormai i giorni contati. Fino a poco tempo fa, Internet è stata usata per lo più dalle università, dall'industria ad alta tecnologia e dal governo americano (in particolare dal Dipartimento della Difesa, che ne è stato il promotore, con l'introduzione della rete Arpanet). Con l'esplosione dell'interesse per Internet, avvenuta dalla metà degli anni novanta e forse anche prima, è altamente probabile che la "rete delle reti" venga usata, a breve, da un insieme di persone estremamente più grande e soprattutto con esigenze diverse. Per esempio, uno degli usi più innovativi che si prevede venga fatto di questo protocollo è nelle **reti wireless**, nelle quali cioè gli utenti avranno a disposizione terminali (computer portatili, palmari, addirittura cellulari) senza filo con i quali tenersi in contatto con la propria postazione base. Appare evidente, perciò, che IP deve evolversi e diventare più flessibile.

Sin dal 1990, l'**IETF** cominciò a lavorare sulla nuova versione di IP, ponendosi una serie di obbiettivi:

- supportare miliardi di host con un sufficiente margine, il che significa sostanzialmente avere a disposizione uno **spazio di indirizzamento** che sia destinato a non esaurirsi praticamente mai come invece sta per avvenire con IPv4;

- ridurre la dimensione delle **tabelle di routing** all'interno dei router della rete;
- semplificare il protocollo, al fine essenzialmente di ridurre il **carico di lavoro** dei router e quindi ridurre anche i **tempi di latenza** dei pacchetti nei router stessi;
- fornire una maggiore **sicurezza** (problematiche di *autenticazione* e *privacy*) rispetto ad IPv4;
- prestare più attenzione al **tipo di servizio**, in particolare ai *servizi real-time* come la voce (Voice over IP, VoIP) o la videoconferenza;
- semplificare il **multicasting**;
- rendere possibile il trasferimento fisico di un host senza modificare il suo indirizzo (**Mobile IP**);
- permettere al protocollo di evolversi nel futuro;
- permettere comunque ai protocolli vecchio e nuovo di coesistere per anni (problematiche di **backward compatibility**).

Sulla scorta di questi obbiettivi, l'IETF fece una richiesta di proposte e discussione in **RFC 1550**. Ci furono ben 21 risposte, anche se alcune non proprio complete. Nel dicembre 1992, il campo delle proposte valide venne ristretto a 7: alcune di queste prevedevano solo piccole modifiche su IPv4, mentre altre volevano rimpiazzarlo con un protocollo completamente nuovo.

Nel 1993, le tre migliori proposte furono pubblicate sulla rivista *IEEE Network*. Dopo una serie di passaggi, fu scelta una versione modificata e combinata delle proposte di *Deering* e *Francis*: inizialmente fu usato il nome *SIPP* (Simple Internet Protocol Plus), ma rapidamente si passò ad **IPv6** ([1]). I documenti di riferimento sono gli **RFC 1883-1887**.

[1] La denominazione IPv5 era già in uso per un protocollo sperimentale in tempo reale, che poi è stato però abbandonato.

Principali caratteristiche di IPv6

IPv6 soddisfa abbastanza bene gli obbiettivi alla base della sua nascita. Esso mantiene le caratteristiche positive di IP, scarta (o, secondo alcuni, "nasconde") quelle negative e ne inserisce di nuove ove necessario. In generale, IPv6 non è compatibile con IPv4, ma è compatibile con tutti gli altri protocolli delle reti TCP/IP, come TCP,UDP, ICMP, IGMP, BGP, DNS e così via. In alcuni casi, bisogna apportare comunque qualche modifica a tali protocolli perché possano appoggiarsi ad IPv6: il motivo è essenzialmente nell'uso di indirizzi più lunghi rispetto ad IPv4.

IPv6 usa infatti **indirizzi** lunghi **16 byte**, al contrario dei 4 di IPv4, il che risolve completamente il problema dello spazio di indirizzamento praticamente illimitato.

Un altro vantaggio è nel fatto che l' **header IPv6** risulta adesso semplificato: esso contiene solo 7 campi (supponendo che gli *indirizzi mittente* e *destinatario* costituiscano un unico campo) contro i 13 di IPv4. Questo snellimento consente ai router di elaborare i pacchetti più velocemente.

Un ulteriore pregio è nel supporto alle **opzioni**:

- in primo luogo, alcuni campi, che in IPv4 erano obbligatori, adesso diventano opzionali;
- in secondo luogo, i router hanno la possibilità di saltare più velocemente le opzioni che non li riguardano.

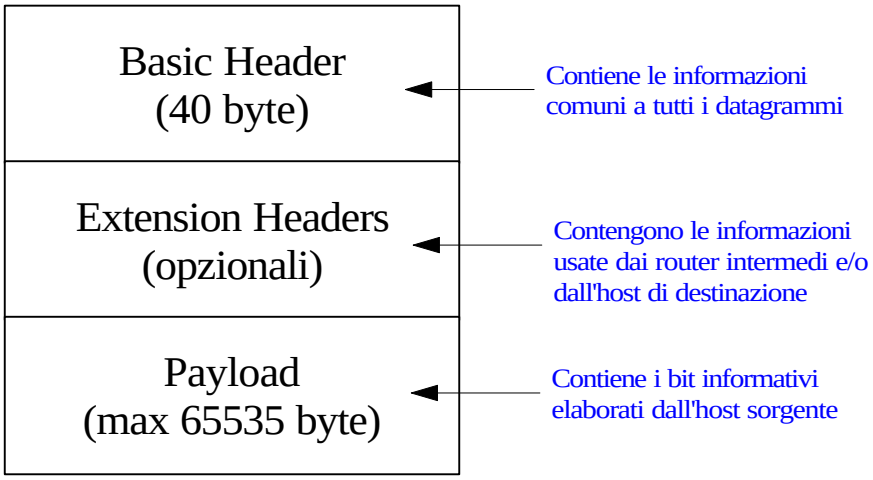
Entrambi questi aspetti rientrano ancora nella necessità di alleggerire il carico dei router.

Anche l'aspetto della sicurezza è stato migliorato, dato che proprio l'**autenticazione** e la **privacy** sono due caratteristiche chiave di IPv6.

Infine, è stata prestata una maggiore attenzione, rispetto al passato, ai tipi di servizi che IP deve supportare. La versione 4 prevede solo un campo ad 8 bit (*type of service*) dedicato a questo problema, mentre invece IPv6 è molto meglio strutturato da questo punto di vista.

Struttura del datagramma IPv6

La figura seguente propone il formato generale secondo cui è strutturato un **pacchetto IPv6**:



Formato generale del datagramma IPv6

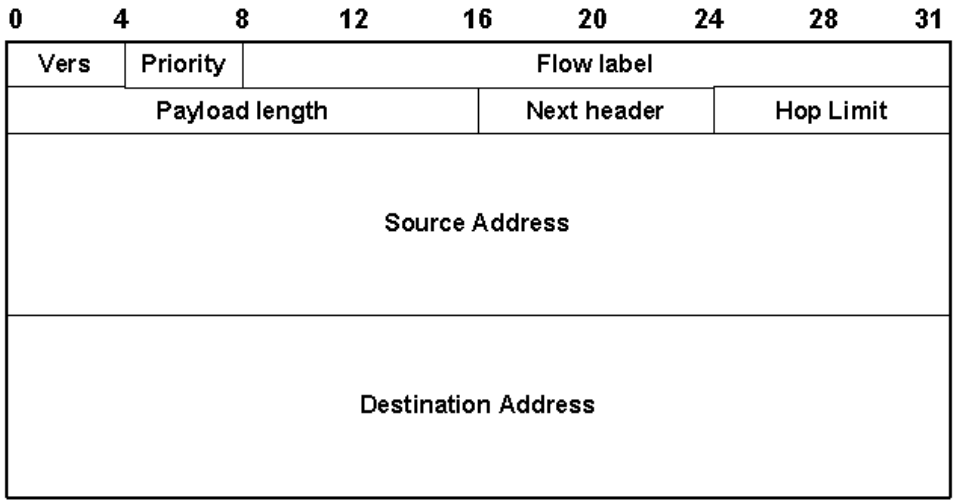
Si individuano dunque fondamentalmente 3 parti: la prima, denominata **Basic header**, consiste nell' header vero e proprio del pacchetto; la seconda, denominata **Extension Headers**, è la parte riservata alle opzioni; la terza, **Payload**, accoglie invece i dati.

Il **Basic header IPv6** è lungo **40 byte** (contro i 24 di IPv4), di cui 32 sono usati per gli indirizzi IPv6 (contro gli 8 di IPv4) e i rimanenti 8 byte sono usati da 6 campi addizionali (mentre invece in IPv4 ci sono 16 byte rimanenti, adibiti a 12 campi addizionali).

Tutti i campi per realizzare le molte nuove funzionalità aggiuntive sono stati inseriti nei cosiddetti **Extension Header**, che sono presenti solo se la funzionalità è effettivamente richiesta. In questo modo, la maggior parte dei pacchetti transita molto velocemente tramite i router che incontra lungo il suo cammino e solo i pacchetti che hanno richieste particolari hanno un trattamento più sofisticato (che prevede appunto l'analisi degli *extension header*). In ogni caso, molti *extension header* riguardano funzionalità "end to end" e quindi non devono essere processati dai router, ma solo dai nodi mittente e destinatario (un tipico caso è l'extension header per la crittografia).

Basic Header

La figura seguente descrive la struttura del **basic header** (da 40 byte) di un pacchetto IPv6:



IPv6 Datagram Basic Header

Analizziamo le funzionalità dei vari campi.

Il campo **Version** (4 bit) contiene semplicemente la versione del protocollo: esso può valere solo 4 (IPv4) oppure 6 (IPv6). Durante il periodo di transizione dall'una all'altra versione, che si prevede duri un decennio, i router dovranno essere in grado di esaminare il contenuto di questo campo per capire che tipo di pacchetto stanno esaminando. In effetti, però, dato che questo controllo comporta un certo tempo di elaborazione ed invece la tendenza è a ridurre tale tempo, non è escluso che il controllo non venga fatto: in questo caso, per capire il tipo di pacchetto, si dovrà ricorrere ad esaminare qualche campo dell' header di livello inferiore (denominato *Host-to-Network* nel modello generale di riferimento TCP/IP); una volta capito se è IPv4 o IPv6, il pacchetto sarà direttamente passato all'entità di livello rete competente ([2]).

Il campo **Priority** (4 bit) stabilisce la priorità del datagramma, al fine sostanzialmente di distinguere le sorgenti che sono sotto *controllo di flusso* da quelle che invece non lo sono. E' dunque un primo mezzo usato per gestire in modo differente dei flussi che abbiano caratteristiche di servizio diverse.

[2] E' ovvio che questo procedimento è una aperta violazione al principio progettuale per cui ogni livello dovrebbe ignorare il significato dei bit che riceve da quello superiore. Tuttavia, nell'ottica di una velocizzazione delle elaborazioni dei router, sembra comunque una strada perseguibile.

Sono definite, in proposito, due **classi di priorità**:

- livelli 0 – 7 (**Congestion Controlled Traffic**): sono riservati a quelle trasmissione che possono rallentare nel caso di congestioni:
- livelli 8 – 15 (**Noncongestion Controlled Traffic**): sono riservati al traffico in tempo reale (ad esempio audio e video), in cui il tasso di spedizione è costante, il ritardo ammesso è piccolo ma viene tollerata la perdita saltuaria di pacchetti.

Questa distinzione permette ai router di gestire meglio i pacchetti in caso di congestione.

La relazione di priorità ha valore solo all'interno di una classe (pacchetti con numeri più bassi sono meno importanti di quelli con numeri più alti), mentre invece non è definita nessuna relazione di priorità tra datagrammi appartenenti a classi diverse.

Per quanto riguarda i livelli di priorità per la classe *Congestion Controlled Traffic*, sono i seguenti:

- 0 : no-specific priority
- 1 : background traffic (news)
- 2 : unattended data transfer (e-mail)
- 3 : reserved
- 4 : attended bulk traffic (FTP)
- 5 : reserved
- 6 : interactive traffic (Telnet)
- 7 : control traffic (e.g. routing protocols and network management)

Il campo **Flow label** (24 bit) è ancora sperimentale ed ha vari scopi:

- serve ad identificare, insieme al campo *source address*, un particolare flusso di datagrammi emessi da una sorgente;
- consente di ridurre i tempi di elaborazione dei datagrammi nei router di rete;

- consente di instradare i datagrammi in hardware mediante consultazione di *tabelle di cache* ([3]) evitando l'esecuzione degli algoritmi di instradamento
- può servire per *procedure di riservazione di risorse* per traffico con qualità di servizio garantita (*protocollo RSVP*).

In generale, quindi, un valore di Flow label diverso da zero indica un **flusso** con particolari esigenze, che quindi richiedono un trattamento ad hoc presso i vari router. Questi ultimi, leggendo nelle proprie tabelle il tipo di servizio richiesto da quel valore del Flow Label, si comportano di conseguenza. Si tratta dunque di un ulteriore mezzo per la suddivisione dei comportamenti da riservare a flussi con caratteristiche diverse.

Ogni flusso è identificato da un proprio **identificatore** oltre che dagli indirizzi sorgente e destinatario, il che significa, per esempio, che possono esserci più di un flusso contemporaneo tra ciascuna coppia di stazioni terminali. Si prevede che gli identificatori dei flussi vengano scelti in modo casuale e non sequenziale, al fine di semplificare ai router la procedura di *hashing*.

Il campo **Payload Length** (16 bit) indica la lunghezza in byte del datagramma IP escluso il basic header: esso quindi dice quanti byte ci sono dopo il basic header. Si tratta dunque di qualcosa di diverso del campo *Total Length* di IPv4, nel quale veniva incluso anche l' header nel conteggio dei byte. Normalmente, la lunghezza massima del payload è 65535 byte. E' anche possibile l'uso dell'opzione "jumbo payload" (contenuta nell'*extension header* denominato **hop-by-hop options**), di cui parleremo più avanti.

Il campo **Next Header** (8 bit) identifica quali *extension header* seguono il basic header nel datagramma. Ci sono una serie di possibilità attualmente standardizzate:

46: Resource Reservation Protocol
50: Encapsulating Security Payload
51: Authentication Header
58: Internet Control Message Protocol (ICMP)
60: Destination Options Header
0: Hop-by-hop options header
4: Internet Protocol (IP)
6: Transmission Control Protocol (TCP)

[3] Il life-time delle tabelle di cache è stato fissato in 6 secondi

17: User Datagram Protocol (UDP)
43: Routing Header
44: Fragment Header

In questo elenco si notano anche i termini TCP e UDP: il motivo è che il campo *next header* può indicare a quale gestore di protocollo di trasporto (ad esempio TCP o UDP) debba essere passato il pacchetto.

Nel campo **Hop Limit** (8 bit) l' host sorgente indica il numero massimo di tratte di rete (e cioè di salti, *hop*) che il datagramma può attraversare. Ogni router decrementa di una unità tale campo. Se il contatore si azzera prima che la destinazione sia raggiunta, il datagramma viene scartato e quindi "scompare" dalla rete. In questo modo, si evitano gli effetti di eventuali *loop* in rete e si possono inoltre effettuare delle ricerche di host in rete a distanza prefissata.

Infine, i campi **Source Address** e **Destination Address** (entrambi da 128 bit) indicano gli indirizzi IP degli host sorgente e di destinazione.

Extension headers (preamboli di estensione)

Come già detto, il motivo per cui il *basic header IPv6* risulta particolarmente semplificato è proprio per il fatto di aver introdotto eventuali *header opzionali* che possano affiancarsi ad esso qualora sia necessario. Questi header contengono dunque informazioni addizionali, destinate direttamente alla destinazione oppure ai sistemi intermedi (essenzialmente ai router), codificate in modo efficiente.

Sono attualmente definiti sei **Extension header**, ognuno dei quali è identificato da un valore del campo *Next Header*. Sono tutti opzionali, ma, se presenti, devono apparire subito dopo il *basic header* e nell'ordine qui di seguito riportato:

- **Hop-by_Hop options**: informazioni varie per i router;
- **Routing**: percorso completo o parziale da far seguire al pacchetto;
- **Fragmentation**: gestione dei datagrammi frammentati;
- **Authentication**: verifica dell'identità del mittente;
- **Encrypted security payload**: informazioni sul contenuto codificato;
- **Destination options**: informazioni addizionali per la destinazione.

Alcuni di questi header hanno un formato fisso, mentre altri contengono un numero variabile di campi, a loro volta di lunghezza variabile.

Hop-by-Hop options

Questo header contiene opzioni per ogni sistema intermedio sul percorso del datagramma, per cui è analizzato da tutti i router.

Ogni opzione è codificata con una tripla di valori:

- **Type** (8 bit): indica il tipo di opzione;
- **Length** (8 bit): indica la lunghezza del campo value (da 0 a 255 byte);
- **Value** (da 0 a 255 byte): trasporta il valore dell'opzione e alcune indicazioni per il router utili per l'elaborazione dell'opzione.

La codifica del tipo di opzione è stata scelta in modo che i primi due bit del campo *type* indichino la "reazione" che un router deve avere nel caso non riconosca l'opzione:

Type	Action
00xxxxxx	Ignora l'opzione ed elabora ugualmente il datagramma
01xxxxxx	Scarta il datagramma
10xxxxxx	Scarta il datagramma ed invia un messaggio di ICMP alla sorgente
11xxxxxx	Scarta il datagramma ed invia un messaggio ICMP alla sorgente solo se la destinazione non è multicast

Il terzo bit del campo *type* stabilisce se il campo *value* può essere modificato:

Type	Action
xx0xxxxx	Il campo value non deve essere modificato
xx1xxxxx	Il campo value può essere modificato

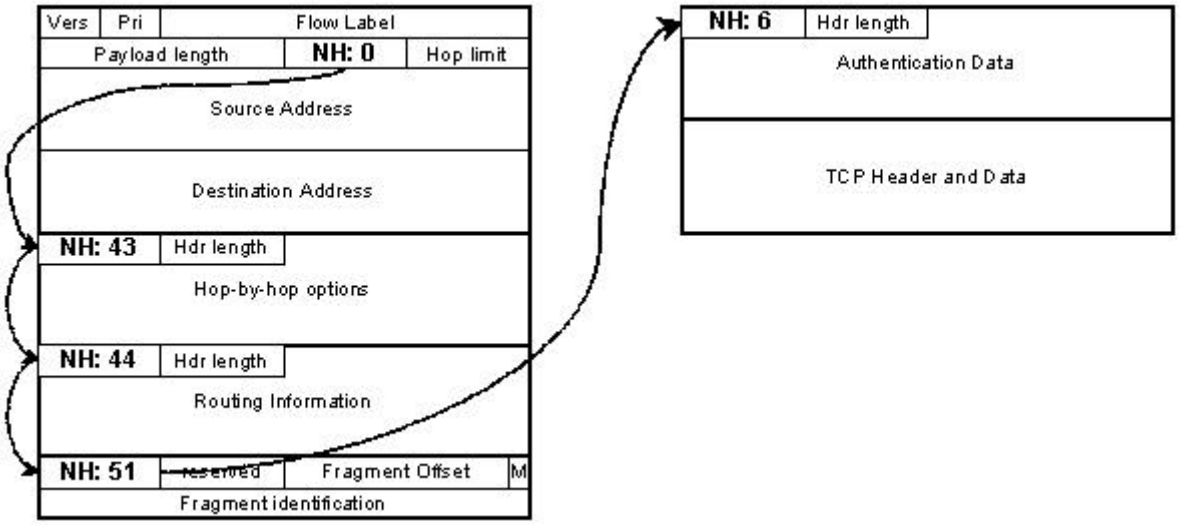
Attualmente, è stata definita un'unica opzione, corrispondente al supporto per pacchetti più lunghi di 65535 byte (i cosiddetti **jumbogram**): il valore del campo Type vale in questo caso 194.

Il formato di questo header è il seguente:

prossimo header	0	194	4
Lunghezza del contenuto del jumbogram			

Come tutti gli extension header, anche questo inizia con un byte che indica il tipo di extension header che viene dopo, secondo la numerazione prima introdotta. Segue un ulteriore byte contenente la lunghezza (in byte) dell' *header hop-by-hop*, escludendo da tale lunghezza i primi 8 che sono obbligatori: in questo caso, essendo proprio 8 la lunghezza complessiva, il valore riportato è 0, come si vede in figura. I successivi due byte indicano che questa opzione definisce la grandezza di un pacchetto (codice 194) come un numero a 4 byte (codice 4 nel quarto byte). Infine, gli ultimi 4 byte restituiscono la dimensione del payload del jumbogram: non sono permesse dimensioni inferiori a 65535 e, nel caso questa regola venisse violata, il primo router scarta il pacchetto e spedisce indietro un messaggio di errore secondo il protocollo **ICMP**.

Tanto per fare un esempio di come i vari extension header possono essere "accodati" al basic header, si consideri la figura seguente, nella quale sono presenti, oltre al basic header, quattro extension header:



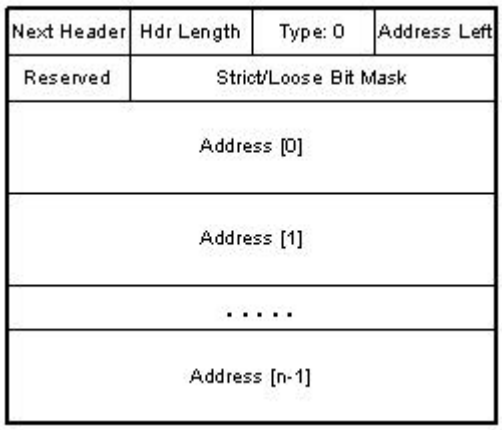
In "cima" troviamo evidentemente il basic header, con i suoi 7 campi. In particolare, il campo next header (NH) riporta il valore 0, il che indica, in base a quanto visto prima, che un *hop-by-hop header* segue il basic header. A sua volta, l'hop-by-hop header (lungo 8 byte) indica che è seguito da un *routing header* (NH=43), a sua volta seguito da un *fragment header* (NH=44), a sua volta seguito da

un *authentication header* (NH=51). Quest'ultimo, essendo l'ultimo header del pacchetto, riporta nel campo NH l'identificativo del processo di livello di trasporto al quale il pacchetto deve essere passato: in questo caso, il valore 6 indica un processo TCP.

Routing header

Questo extension header fornisce indicazioni per l'instradamento del datagramma, forzando l'uso di un particolare **cammino**: esso cioè indica, tramite i rispettivi indirizzi, uno o più router che devono essere necessariamente attraversati. In particolare, sono possibili sia il **routing rigido** (per cui il pacchetto deve passare per tutti e soli i router specificati) sia il **routing flessibile** (per cui il pacchetto deve passare per i router selezionati e, eventualmente, anche per altri).

Il formato di questo header è il seguente:

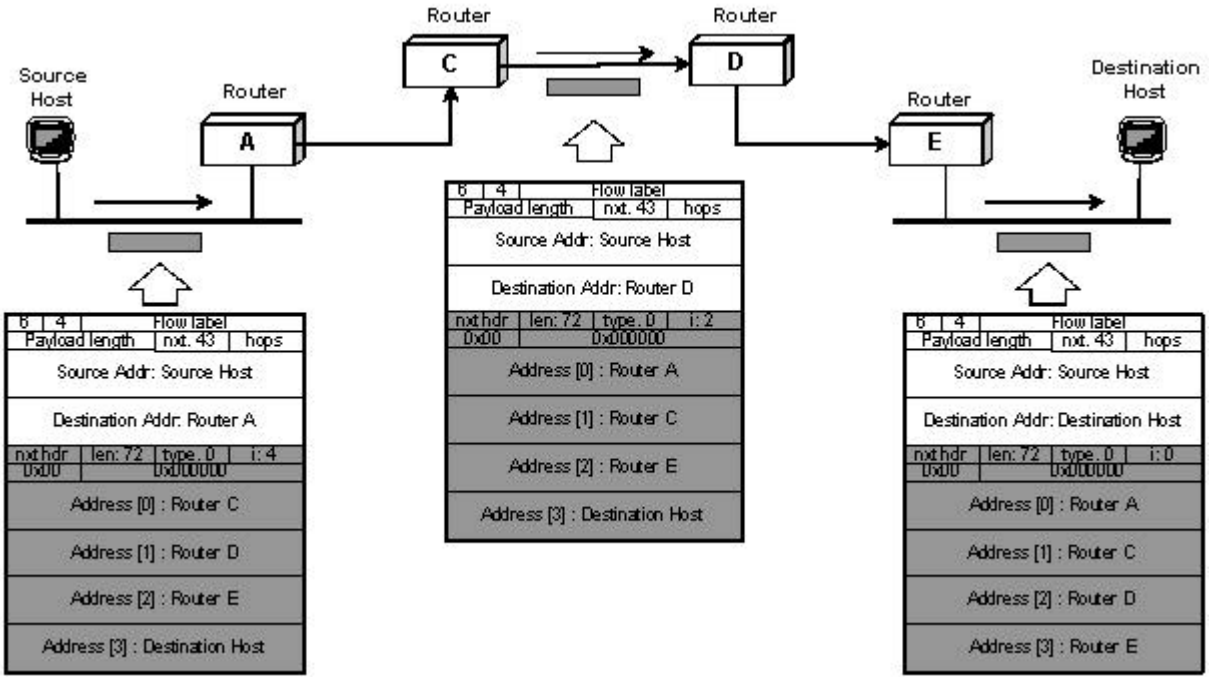


Il primo campo è il solito **Next Header** da un byte. Seguono il campo **Header Length**, che quantifica la lunghezza, in byte, dell' header (sempre ad eccezione dei primi 8 byte obbligatori), ed il campo **Type**, quest'ultimo riportante il tipo di routing header (normalmente 0). Da notare che il contenuto del campo *Length* consente di dedurre univocamente quanti indirizzi (*address*) sono riportati nella lista riportata alla fine dell' header: ad esempio, un valore di 72 byte, considerando che ogni indirizzo IPv6 è di 16 byte, corrisponde a 72/16=4 indirizzi, cioè a 4 router.

Il campo **Address left** indica il numero di indirizzi che devono essere ancora elaborati: di conseguenza, ogni router decrementa tale campo prima di inviare il pacchetto al router successivo.

C'è poi un campo da un byte non ancora utilizzato, seguito dal campo **Strict/Loose bit mask**: i bit di questa maschera esso indicano, per ogni indirizzo della lista successiva (comprendente da 1 a 24 indirizzi), le modalità di instradamento; valore 0 indica *instradamento rigido*, mentre valore 1 significa possibile *instradamento flessibile*. Quindi, per ognuno dei router riportati nella lista degli indirizzi, il corrispondente bit del campo *Strict/Loose bit mask* dice se bisogna raggiungerlo immediatamente a partire dal router precedente oppure è possibile passare prima, qualora sia necessario, per altri router intermedi.

Facciamo un semplice esempio di funzionamento di questo tipo di header, con riferimento alla figura seguente:



L' *host sorgente* emette dunque un pacchetto, specificando, nel campo *destination address* del basic header, un particolare router (router A). Viene ovviamente specificato anche l'indirizzo della destinazione, come ultimo della lista. Il router A, in base al valore del campo *Address Left* (che adesso vale 4), deduce qual è l'indirizzo del prossimo router cui inviare il pacchetto (router C) nonché se sia obbligatorio o meno l'invio diretto a quel router. In questo esempio, pur non essendo obbligatorio il passaggio al router C, il router A dispone di una linea verso di esso e quindi gli invia il pacchetto; prima dell'invio, però, il router diminuisce di uno il campo *Address Left* ed inoltre modifica il campo *destination address* (inserendo l'indirizzo del router C) nonché l'elenco degli indirizzi dei router riportati nel routing header: quest'ultima modifica viene fatta in modo che, quando il

pacchetto giunge fino a destinazione, nel routing header venga riportata la sequenza esatta dei router (solo quelli obbligatori) attraversati, in modo che la destinazione conosca perfettamente il percorso seguito dal pacchetto.

Gli altri router si comportano evidentemente in modo analogo.

Fragment header

Questo extension header serve alla segmentazione e ricostruzione dei datagrammi (qualora siano troppo grandi), usando lo stesso meccanismo di IPv4. Tutti i frammenti, tranne l'ultimo, hanno lunghezza multipla di 8 bytes.

Il formato di questo header è il seguente:

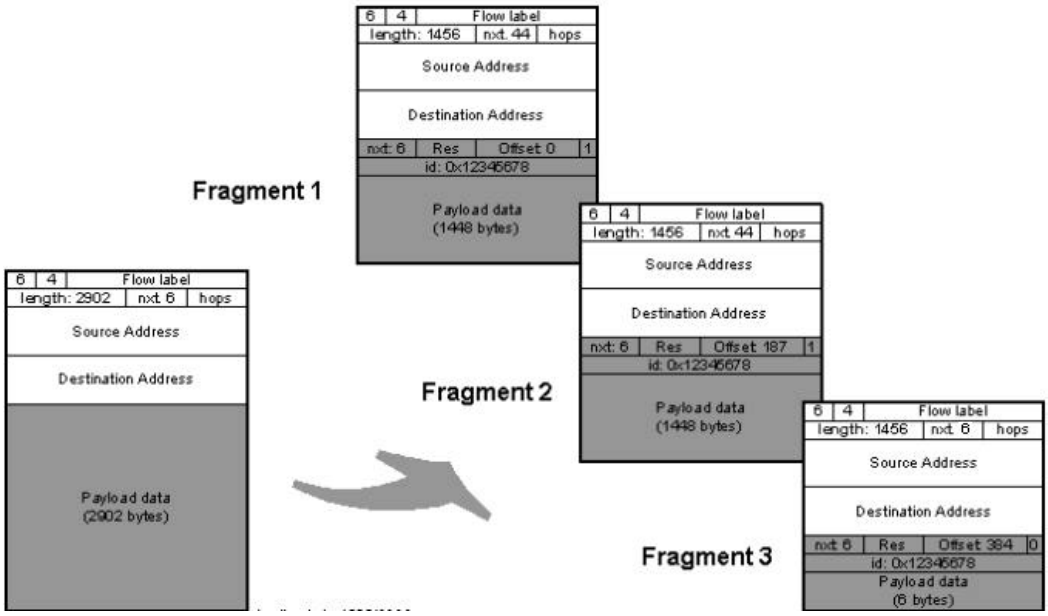
Il campo **Fragment Offset** specifica la posizione (in unità di 8 bytes) del primo byte del frammento nel datagramma originale.

Il **bit M** è posto a 1 in tutti i frammenti del datagramma tranne l'ultimo, in cui è posto ad 0. Serve dunque a segnalare la fine del datagramma frammentato.

Il campo **Fragment identification** identifica il datagramma a cui il frammento appartiene: si tenga infatti conto che i vari frammenti viaggiano in modo indipendente nella rete, per cui diventa importante, nella destinazione, poterli riordinare nel modo corretto.

In IPv6, diversamente da IPv4, solo l' host sorgente può frammentare un pacchetto, mentre invece i router sul cammino non possono farlo (al fine di alleggerire il loro lavoro). Se un router dovesse incontrare un pacchetto troppo grande, lo scarta e spedisce un pacchetto ICMP alla sorgente, che quindi può provvedere a segmentare il pacchetto originale in pezzi più piccoli e provare ancora.

Un esempio di frammentazione di pacchetto IPv6 tramite l'uso del Fragment header è riportato nella figura seguente:

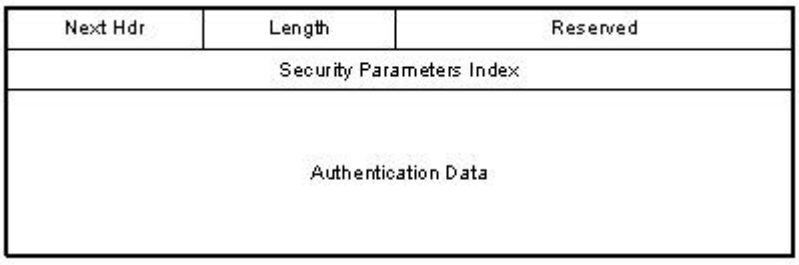


Come si vede, il campo *next header* del pacchetto originale contiene il valore 6, identificativo del processo TCP cui consegnare il pacchetto. Al contrario, il campo next header dei primi 2 frammenti contiene il valore 44, mentre l'ultimo contiene nuovamente il valore 6.

Authentication header

Questo header assicura che il datagramma sia autentico e cioè che non sia stato alterato durante il transito in rete e sia stato emesso effettivamente dal mittente indicato nel datagramma. IPv4 non fornisce invece questa garanzia.

Il formato è il seguente:



Destination options

Questo header serve ad indicare informazioni aggiuntive (**opzioni**) sul destinatario. Se le *destination options* sono solo per l'utente finale, questo extension header è l'ultimo. Se sono invece dirette ad un router intermedio, tale opzione è usata in unione con l'opzione *routing header* e precede quest'ultima.

Si possono inserire due destination options per distinguere le informazioni dirette ai router intermedi da quelle dirette all'utente finale.

Per quanto riguarda la codifica delle opzioni, valgono esattamente le stesse regole usate per l'extension header *Hop-by-Hop Options*.

Autore: **Sandro Petrizzelli**
e-mail: sandry@iol.it
sito personale: http://users.iol.it/sandry