QoS per varie applicazioni

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Soluzioni per il QoS

- Riguardo alla **reliability**, abbiamo già visto soluzioni come ***error control*** ed ***error correction***

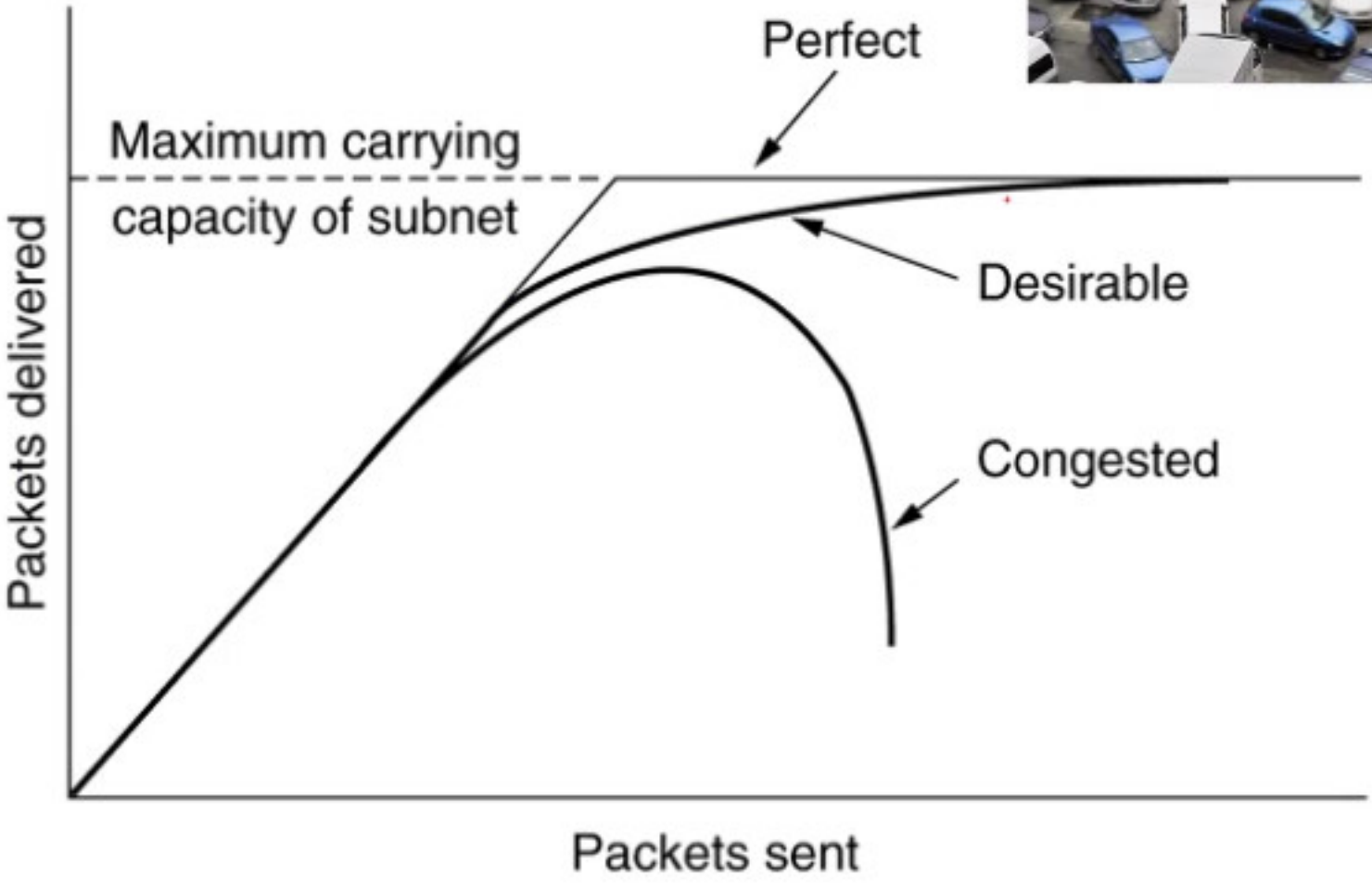
Soluzioni per il QoS

- Le altre misure (**bandwidth**, **delay**, **jitter**), derivano da alcune ***cause***, che devono essere risolte se si vuole mantenere una buona QoS

Causa: la congestione

- Una delle cause principali di problemi nel QoS è la ***congestione***, cioè quando la capacità della linea o di qualche stazione si satura
- E' importante dunque fare attenzione al ***congestion control*** per evitare i problemi di QoS

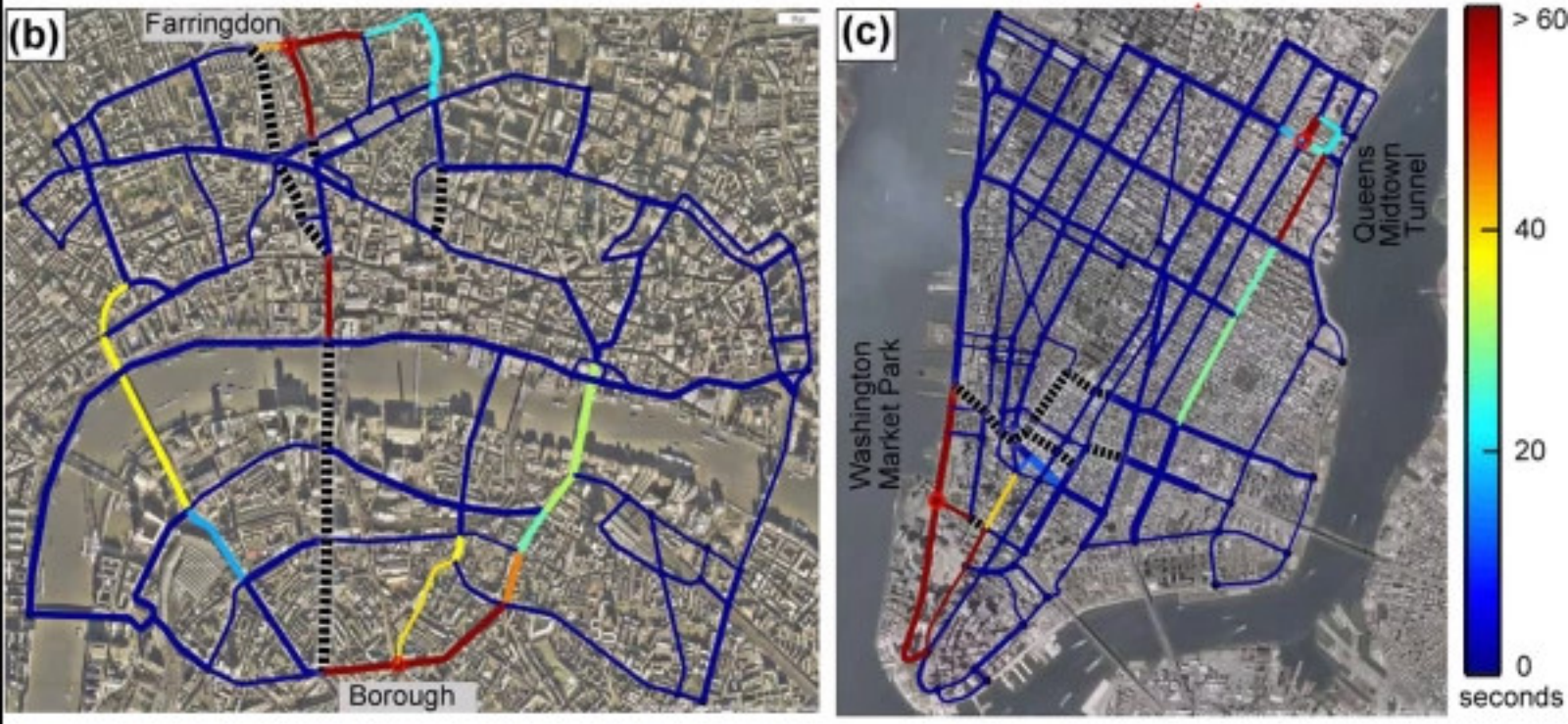
Congestione



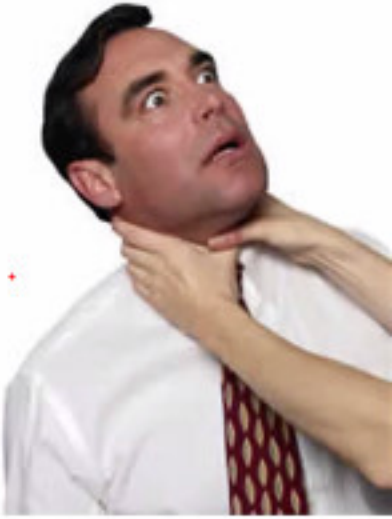
Problema molto complesso...

- Ad esempio il «Paradosso di Braess»:
- ***Aggiungere*** capacità ad una rete...
- ... può portare ad una ***diminuzione*** della performance (!!!)

Boston, Londra, New York etc...



La tecnica dei choke packet

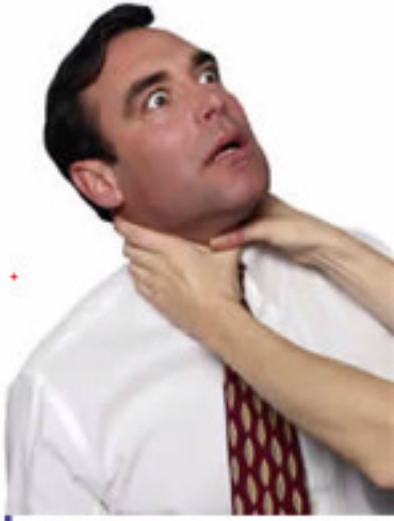


- Come nel flow control, alle volte il ricevente può segnalare che le cose non vanno bene, anche qui l'idea è che se un router si accorge che c'è congestione, può inviare un pacchetto speciale (***choke packet***) a chi sta inviando dati, dicendogli di rallentare

I choke packets

- Tipicamente dunque, un host ad esempio dimezza il suo data rate non appena riceve un choke packet
- Anche qui occorre fare attenzione alla gestione del choke:

Gestione del choke

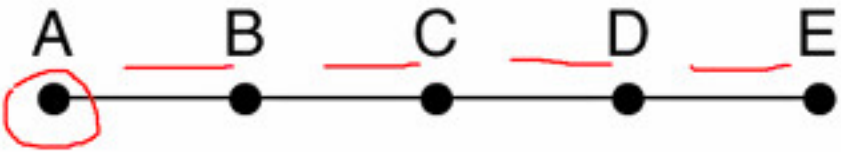


- Ad esempio, si usa il fading come modo per uscire dal choke:
- Se non riceviamo choke packets dopo un certo periodo di tempo, torniamo ad incrementare la nostra velocità di trasmissione

Gestione del choke

- C'è anche un altro problema più sottile: non l'uscita dal choke, ma l'entrata nel choke

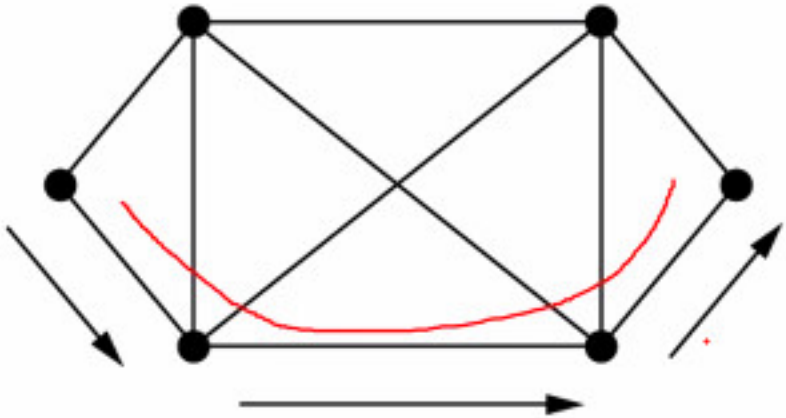
Esempio



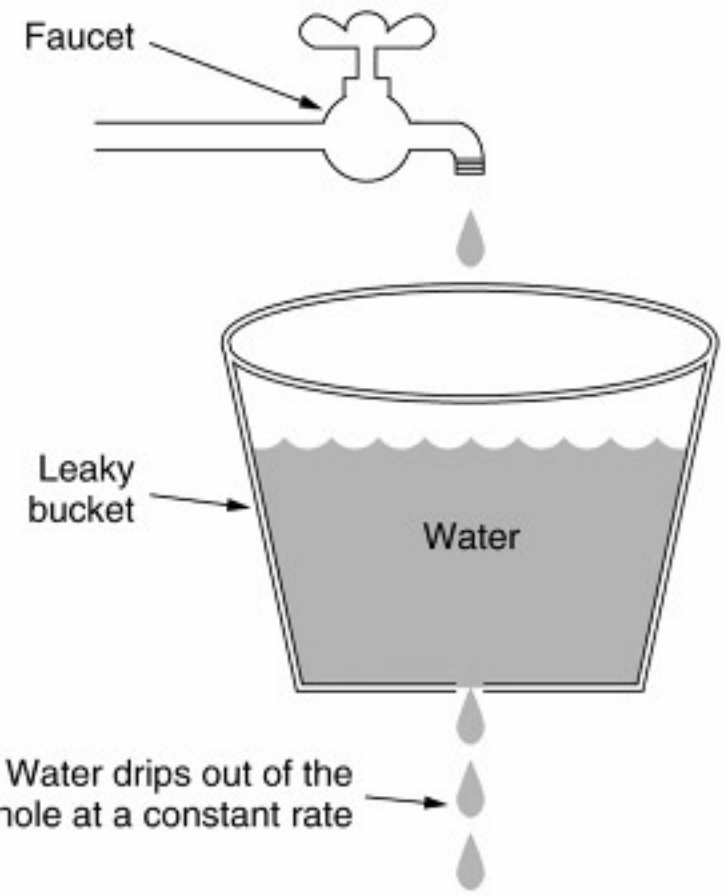
- C'e' congestione sulla linea tra A ed E
- → B, C e D inviano dei choke ad A
- A riceve il primo choke da B: -50% (0.5)
- Riceve il secondo da C: -50% (0.25)
- Riceve il terzo da D: -50% (0.125)

Soluzione:

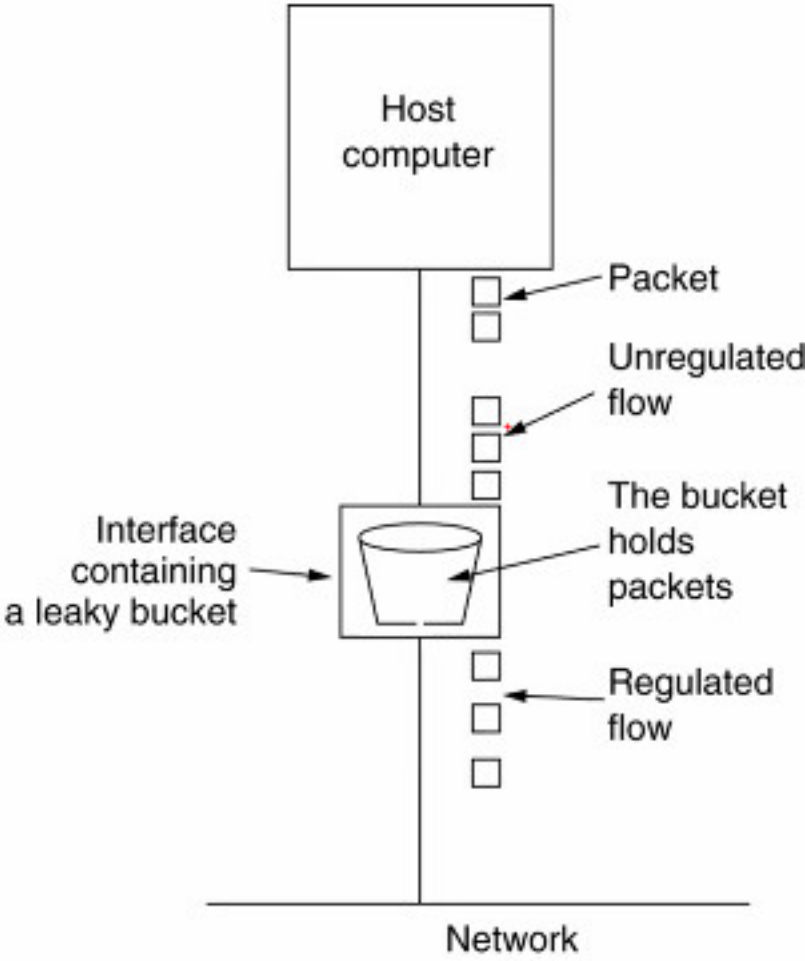
- Si fa fading alla rovescia, nel senso che non appena si riceve un choke, per un certo periodo di tempo si ***ignorano*** le altre richieste di choke che vengono dalla stessa destinazione



Leaky Bucket



(a)



(b)

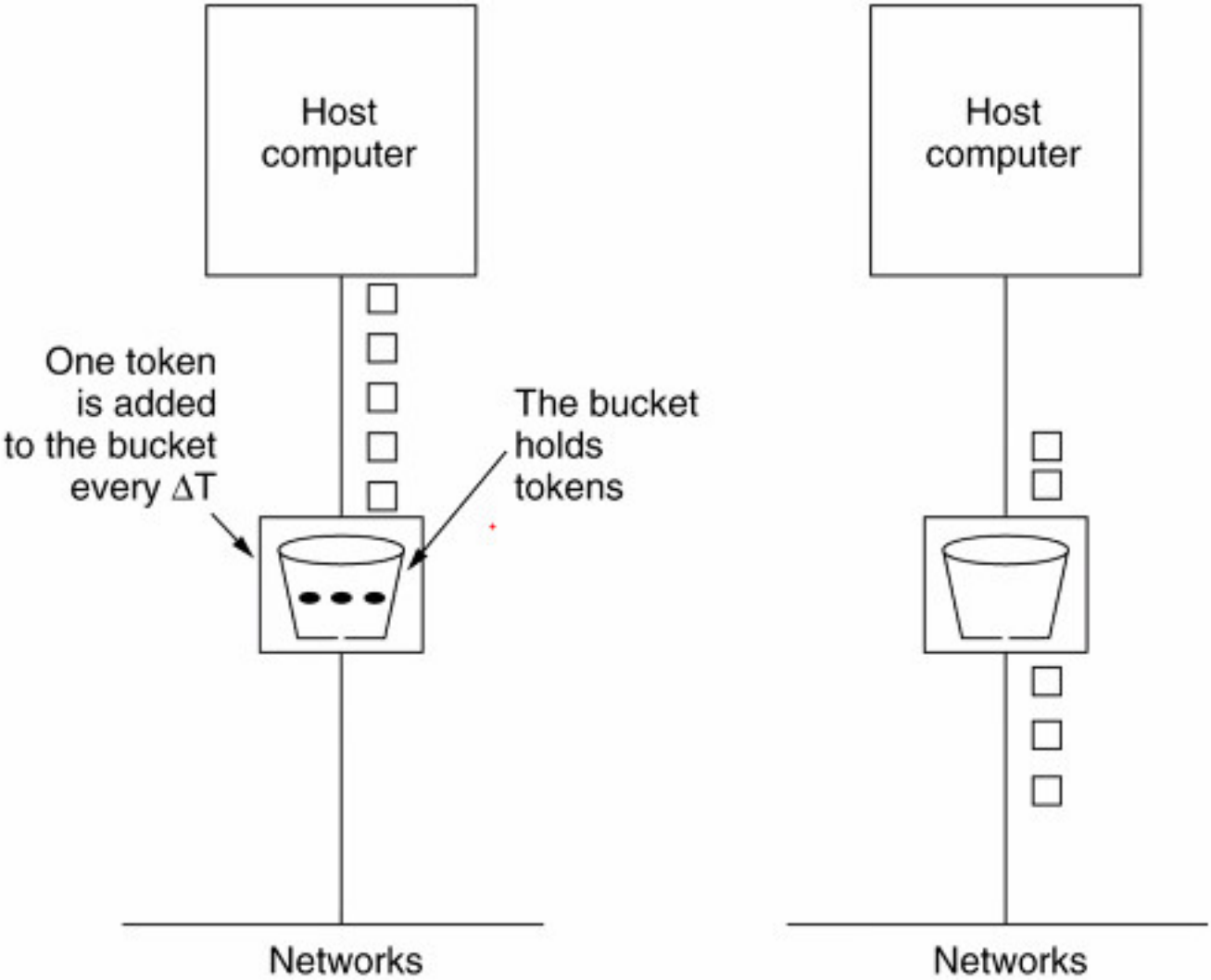
Oltre il leaky bucket

- Il leaky bucket ha il vantaggio/svantaggio che il max data rate è sempre costante
- Alle volte, ad esempio in presenza di traffico più sostenuto, converrebbe magari aumentare un po' il data rate

Il Token Bucket

- Il token bucket genera ogni certo intervallo di tempo un ***token***
- I pacchetti in arrivo possono uscire solo se "bruciano" un token disponibile
- Quindi, se il traffico per un certo periodo è lento, ma poi c'e' un burst, si riesce a gestirlo meglio consumando i token che si erano accumulati

Token Bucket



"TCP/IP" o "TCP" / "IP"?

- 1969: ARPANET
- Usava il Network Control Protocol (NCP)
- → ***inadatto***
- 1974: Transmission Control Protocol (TCP)
- 1978: Divisione TCP e IP

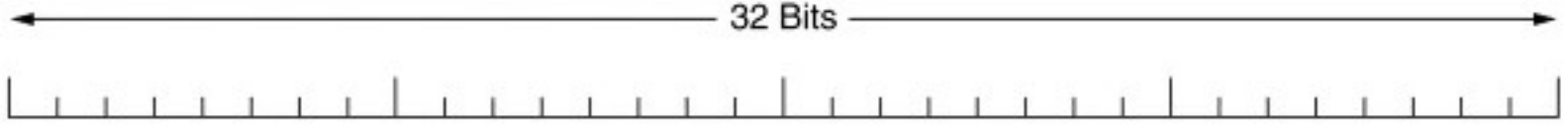
Ed ora...

Lo strato di rete di Internet

- Dopo aver visto varie tecniche per il routing ed il QoS, è arrivato il momento di vedere quale protocollo usa Internet per trasmettere dati a livello di rete
- **IP** = **I**nternet **P**rotocol
- Che avrete sentito senz'altro di solito dentro a "TCP/IP"...

L'header di IP

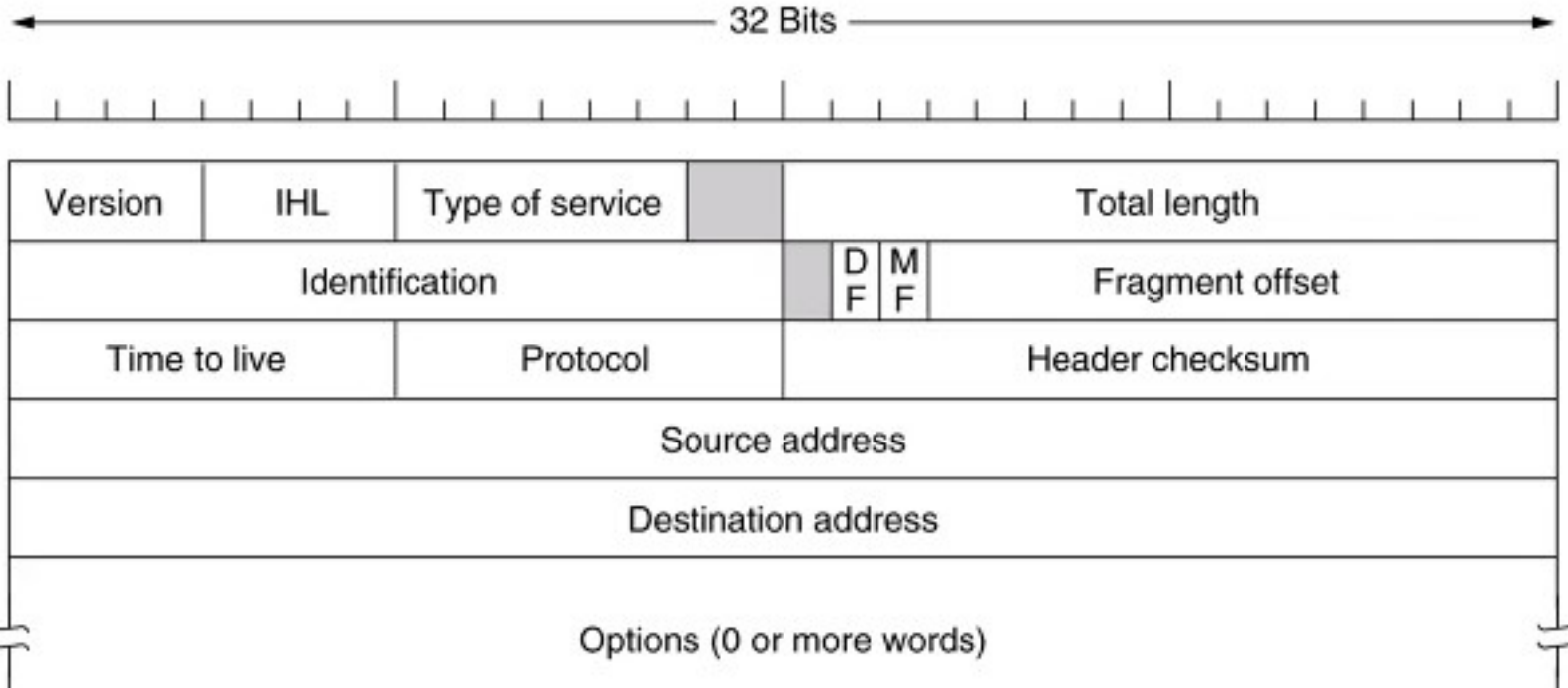
- Ha una parte fissa a 20 bytes, ed una a lunghezza variabile



Version	IHL	Type of service		Total length
Identification			DF MF	Fragment offset
Time to live		Protocol		Header checksum
Source address				
Destination address				
Options (0 or more words)				

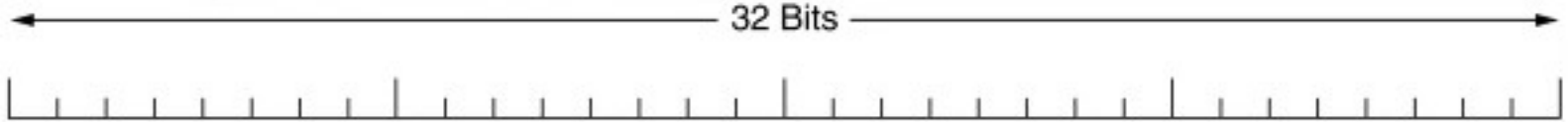
L'header di IP

- La prima parte è la versione di IP usata (permette quindi il ***versioning***)



L'header di IP

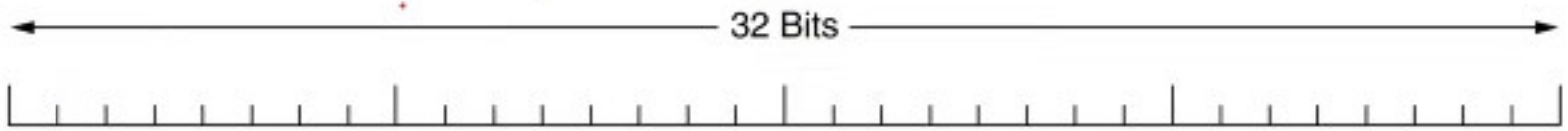
- La seconda è la lunghezza dell'header (**I**P **H**eader **L**enght)



Version	IHL	Type of service		Total length
Identification			DF MF	Fragment offset
Time to live		Protocol		Header checksum
Source address				
Destination address				
Options (0 or more words)				

L'header di IP

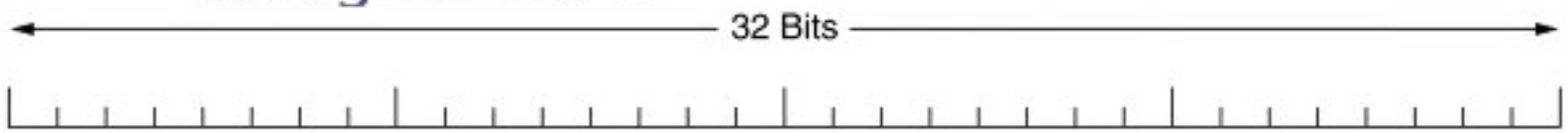
- La terza è il tipo di servizio (campo adatto per la selezione della QoS), anche se spesso ignorato



Version	IHL	Type of service		Total length		
Identification				DF	MF	Fragment offset
Time to live		Protocol		Header checksum		
Source address						
Destination address						

L'header di IP

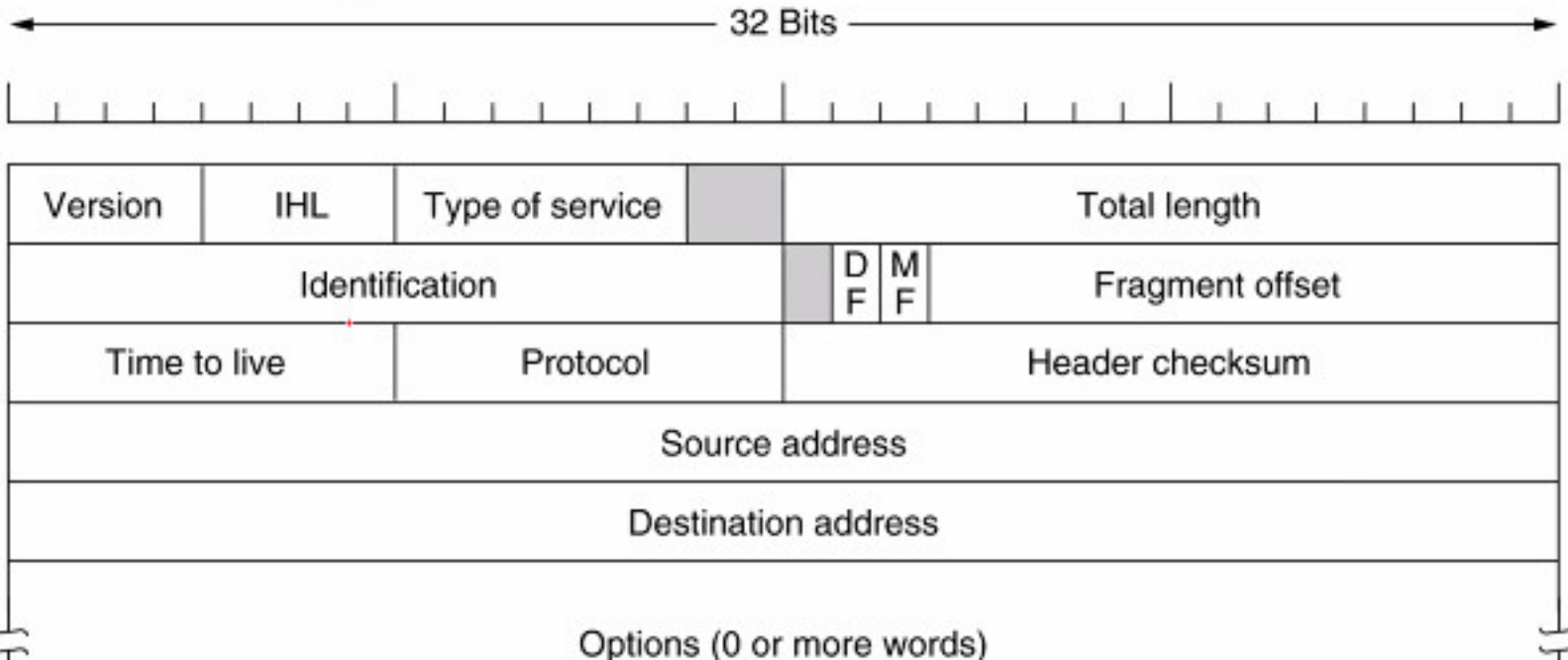
- C'è poi la lunghezza totale del datagramma IP



Version	IHL	Type of service		Total length	
Identification				DF MF	Fragment offset
Time to live		Protocol		Header checksum	
Source address					
Destination address					
Options (0 or more words)					

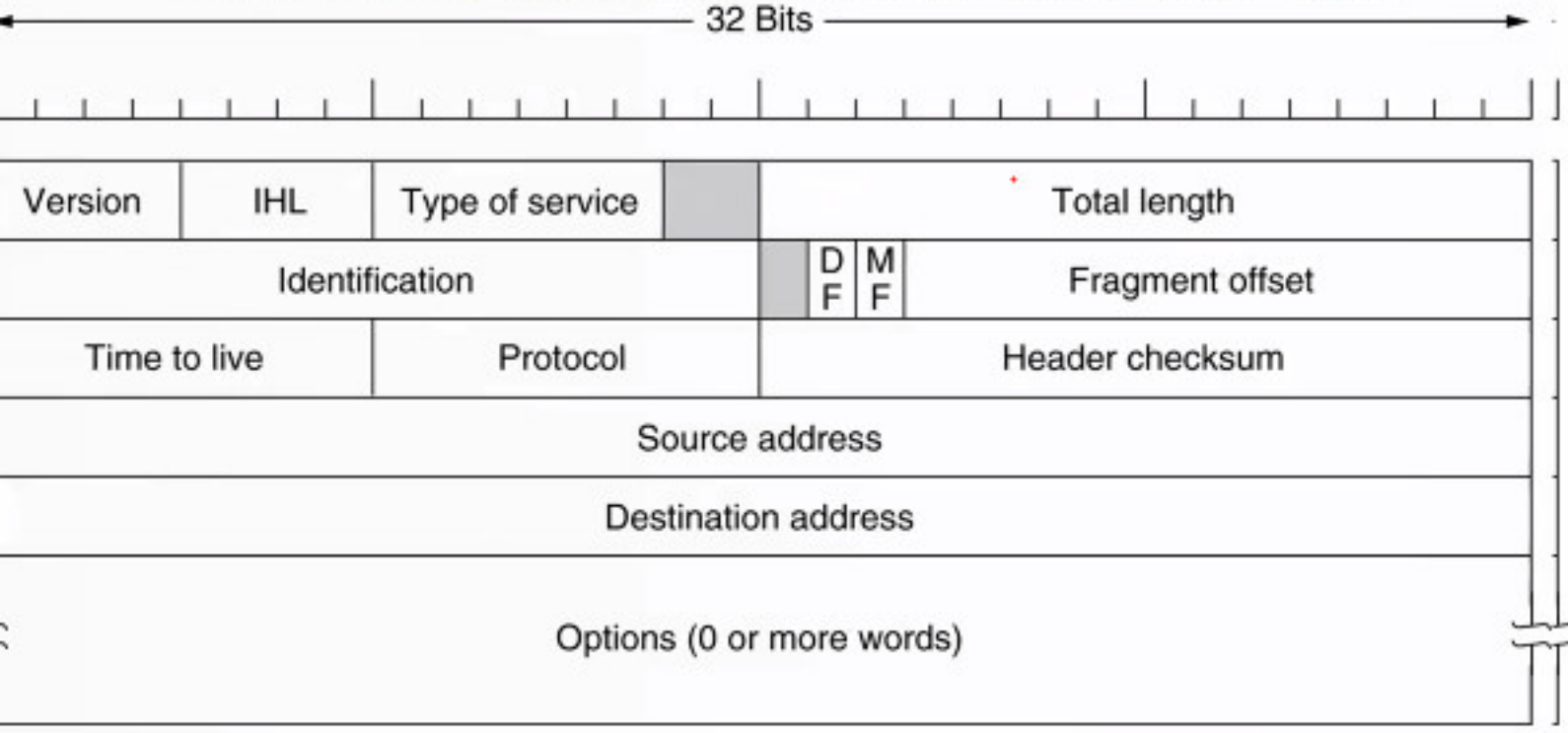
L'header di IP

- L'***identification*** serve a identificare se c'è stata frammentazione dei dati in più datagrammi



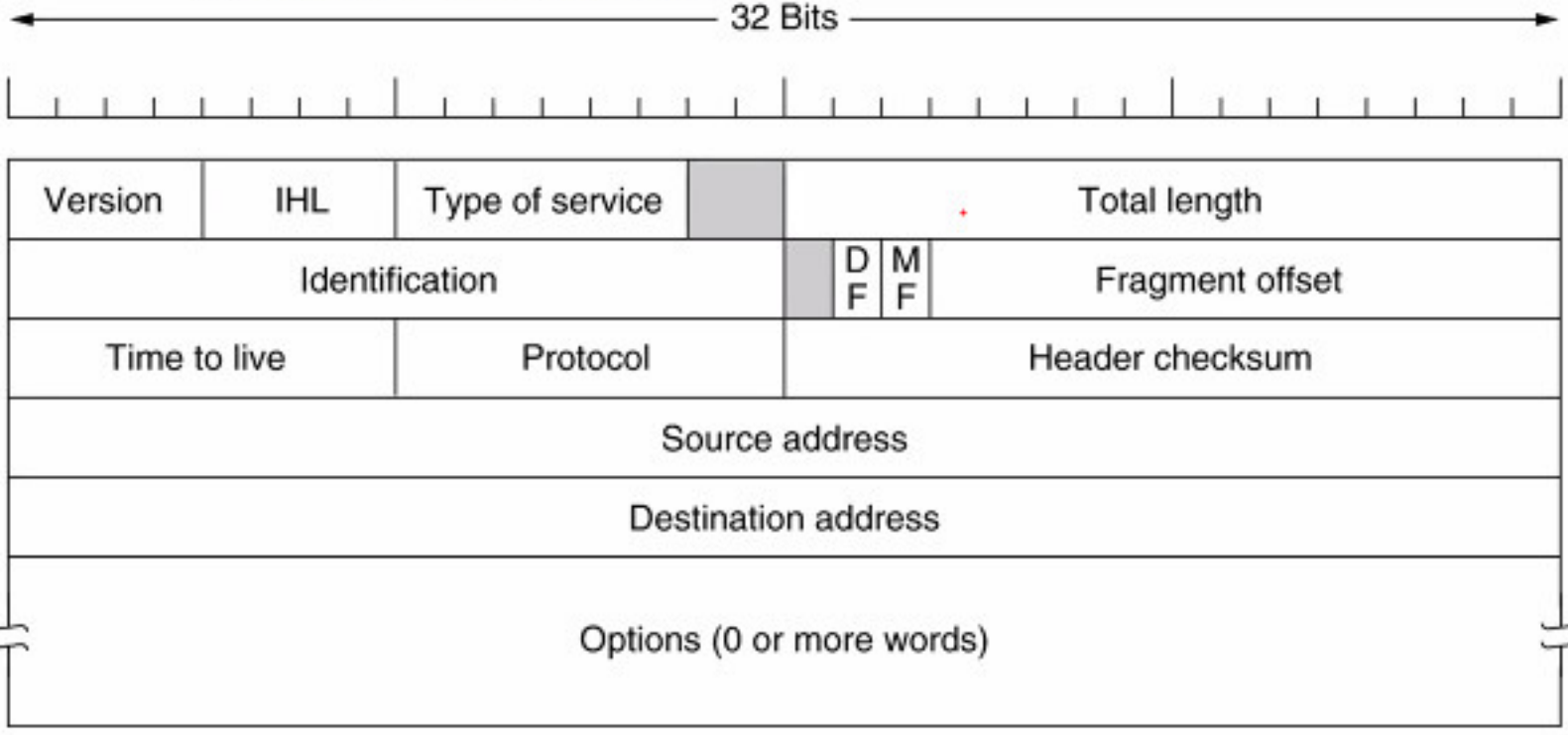
L'header di IP

- Il Campo DF invece (Don't Fragment) impone la non-frammentazione dei dati



L'header di IP

- Il campo **MF** (More Fragment) segnala l'ultimo frammento

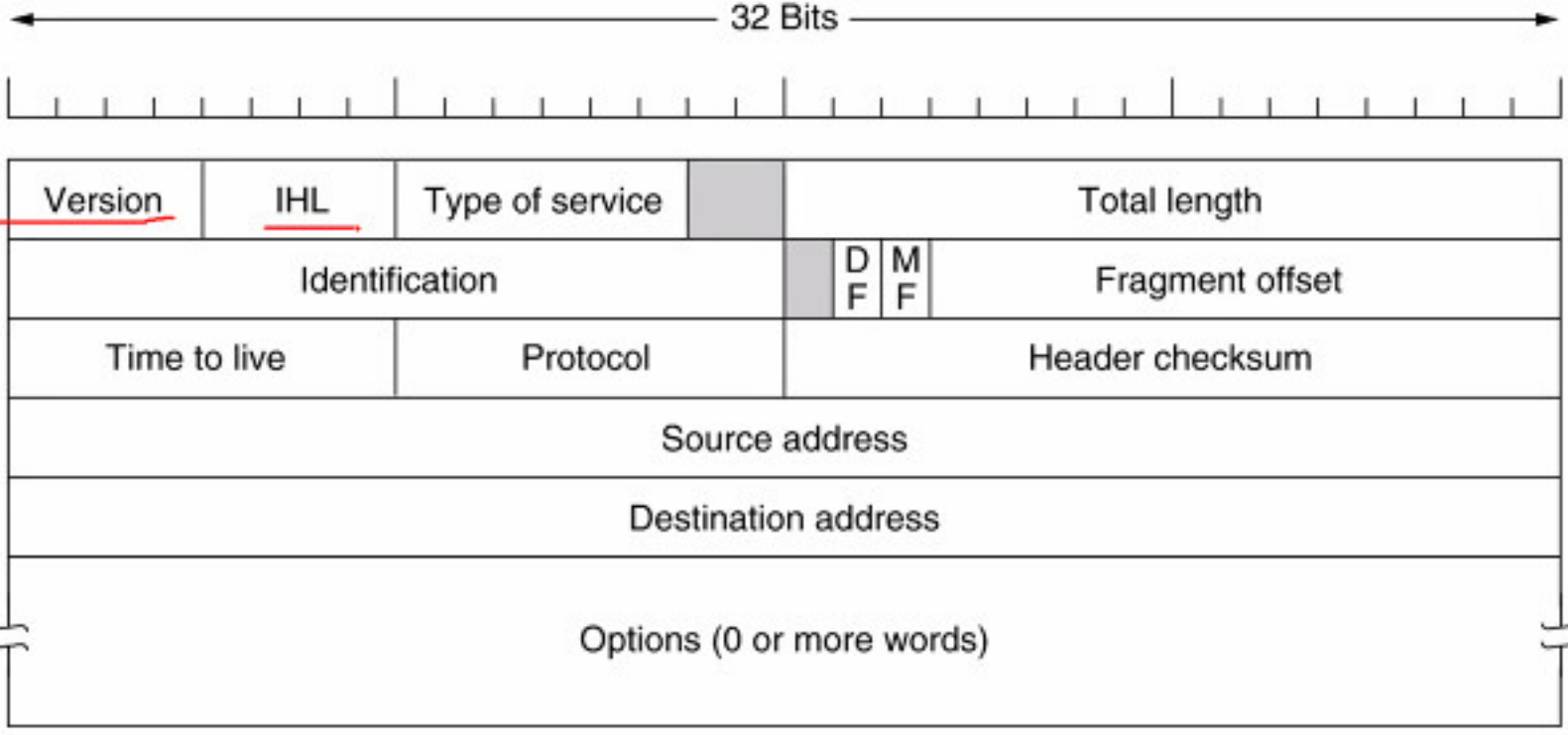


"Evil Bit"

- Standardizzato in IETF RFC3514

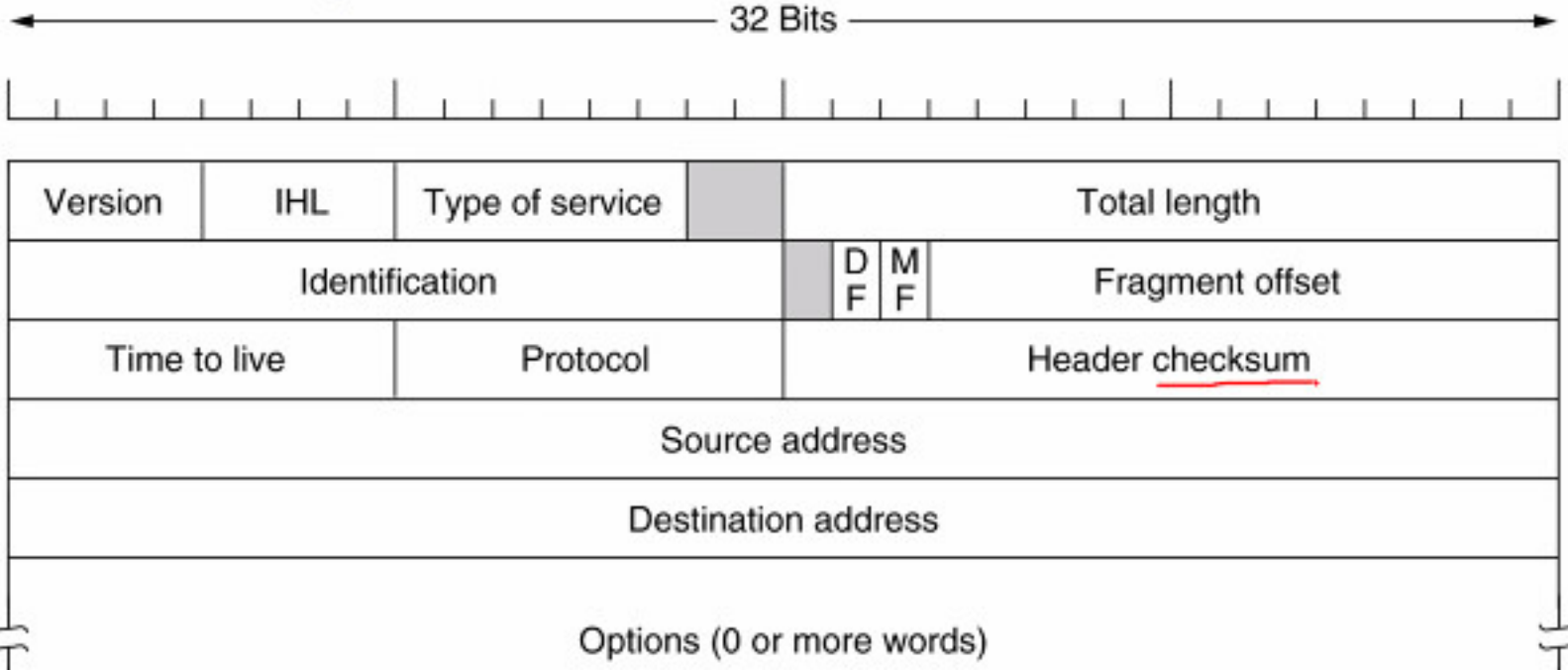
L'header di IP

- Il Time to Live (TTL) è l'età massima del pacchetto



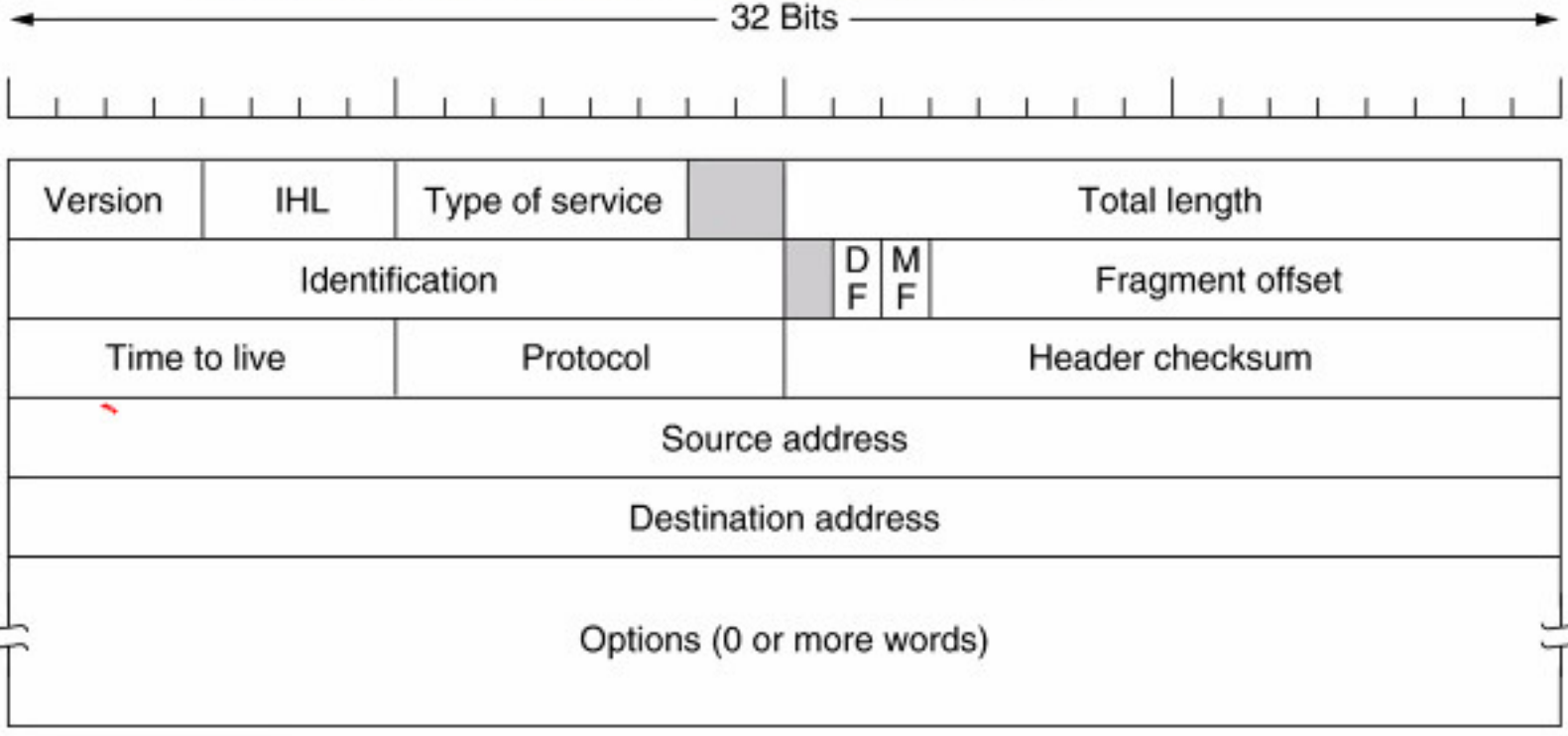
L'header di IP

- C'è poi un cheksum ***per l'header***, calcolato banalmente tramite somme in complemento a uno



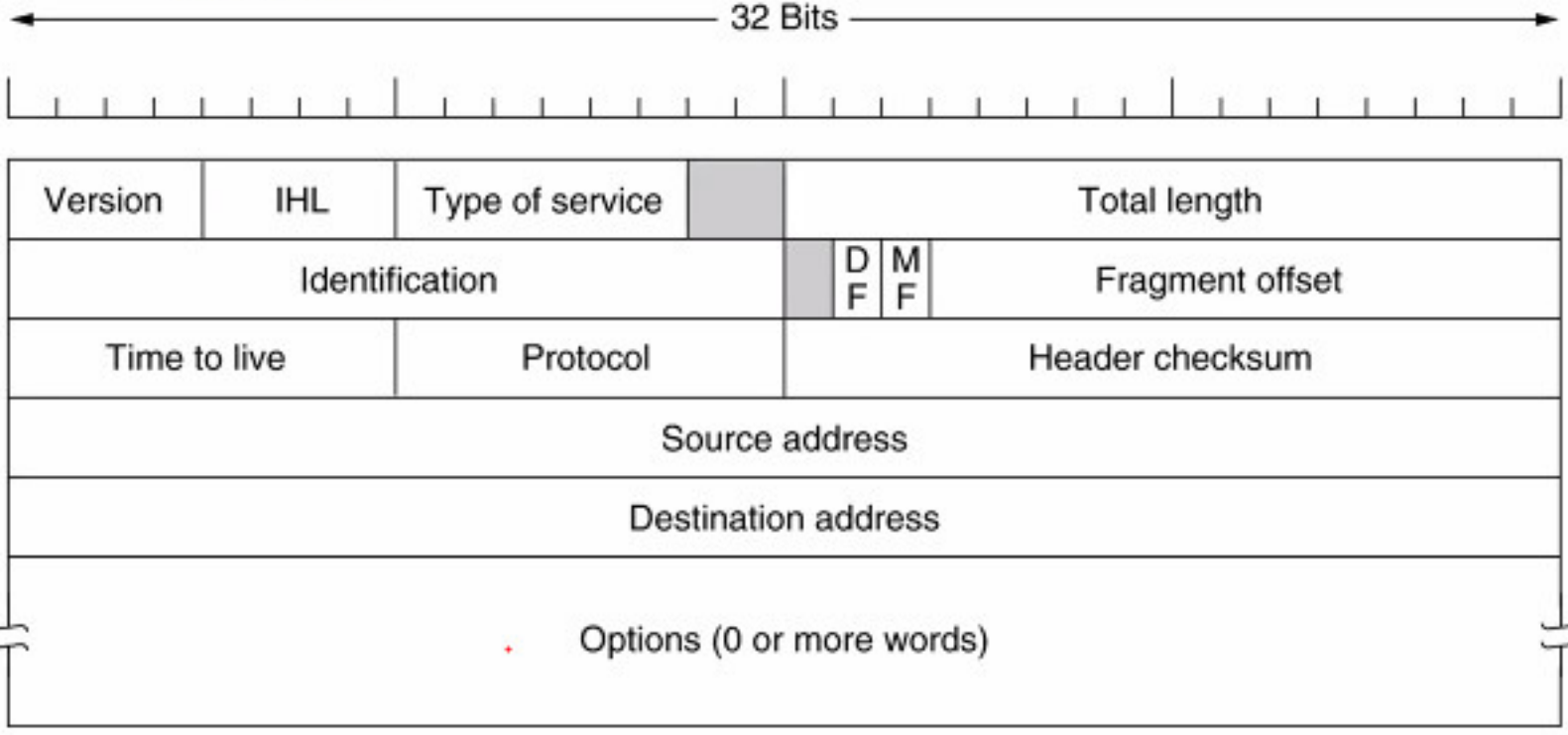
L'header di IP

- Poi, gli indirizzi IP di chi invia e di chi deve ricevere il datagramma



L'header di IP

- E infine, la parte variabile per eventuali opzioni



Analisi critica

- IP…. Lo strato fondamentale di Internet…

- Tutto qui?

DoD

Indietro alla storia…

- **1969**: ARPANET
- Usava il Network Control Protocol (NCP)
- ***Reliable, flow-controlled (!)***
- **1974**: Transmission Control Protocol (TCP) → ***ancora troppo per il DoD***
- **1978**: Divisione TCP e IP
- (**1983**: DoD impone il TCP/IP ☺)

DoD e l'uso militare

- ARPANET → Internet e… MILNET (!)

Design: no overload

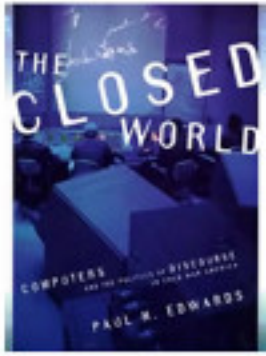
- La rete non deve essere sottoposta al pericolo di sovraccarichi
- → niente traccia in IP degli errori intermedi durante una comunicazione

Design: no overload (2)

- Ed inoltre, dunque, niente connessione persistente (→ IP "connection-less")
- E dunque, necessita' di ***ritrasmissione*** nel caso di failure

Modelli Open e Closed World

- Nonostante gli sforzi per rendere TCP/IP "robusto ad attacchi", il sistema e' stato pensato per il modello closed-world (tipo MILNET) → attacchi *esterni*
- Quando l'attacco e' *interno* (open-world model), ci sono molte vulnerabilita' (ad esempio, IP gestisce male la ***congestione del traffico***)



I limiti di estendibilità

- Uno dei più grossi problemi dell'informatica è l'estendibilità di un servizio

Esempi

"640k ought to be enough
for everybody"

Limiti di IP...?

- TTL (Time to Live): 255 hops massimi
- Può essere un problema se la rete è mal disegnata e non gerarchica
- Comunque, il router può intervenire e riaumentare il TTL (o, non diminuirlo), quindi non è un grosso problema

Limiti di IP (cont.)

- Gli indirizzi (source e destination): **32 bits**
- Questo si è rivelato invece essere un ***grandioso problema***, perché lo spazio degli indirizzi globale è fisso e non estendibile

Indirizzi IP

- Ovviamente, ci deve essere una ***autorità centralizzata*** che assegna gli indirizzi IP alle entità in Internet
- Potremmo assegnare gli indirizzi ***uno a uno***, ma in questo modo, i router dovrebbero mantenere delle tabelle mostruosamente grandi (!!!!)

Le classi principali di indirizzi IP

- **Classe A**: (7+24)
  128 reti possibili, con indirizzi di 24 bits (circa 16.7 milioni)
- **Classe B**: (14+16)
  16384 reti possibili, con indirizzi di 16 bits (65536)
- **Classe C**: (21+8)
  circa 2 milioni di reti, con indirizzi di 8 bits (256)

Notare…

- Le taglie possibili delle reti:
- **1 byte – 2 bytes – 3 bytes**

256 – 65536 – 16.7 milioni