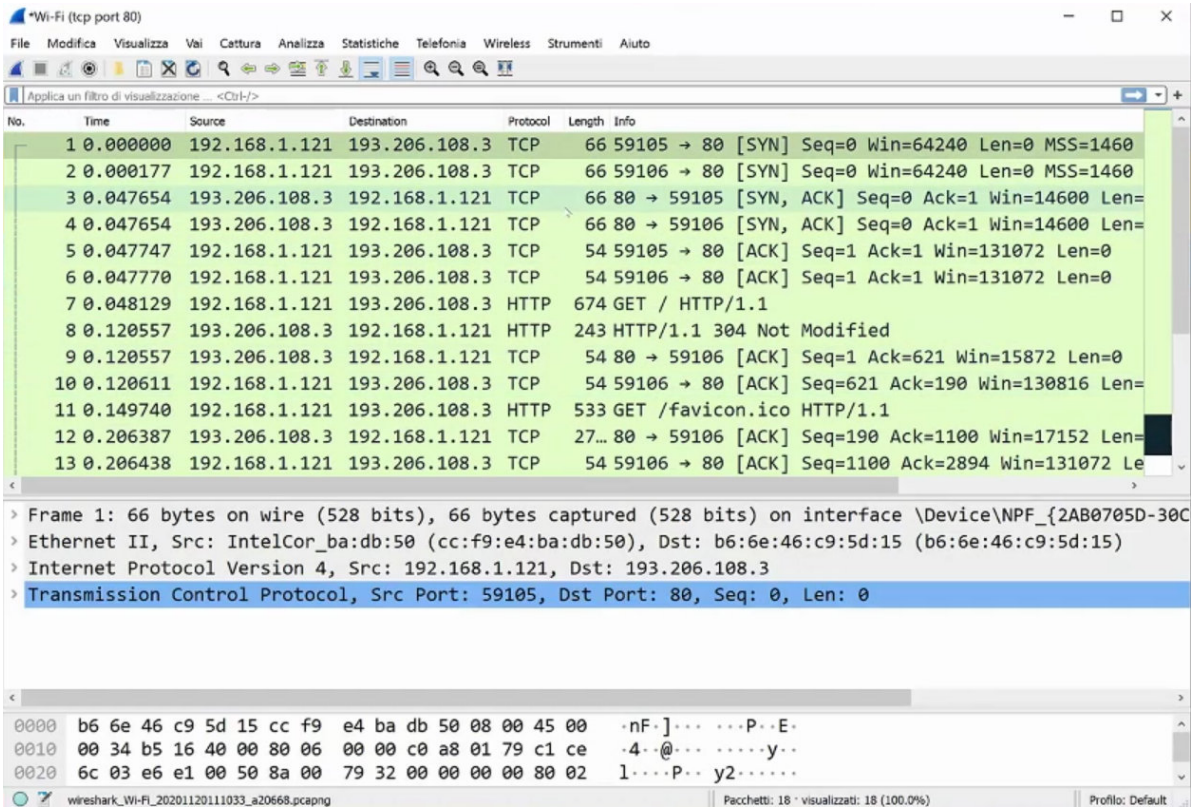
Visualizzazione di pacchetti scambiati con Wireshark

Utilizziamo il sito www.old-unisannio.it (non più online) per visualizzare i pacchetti; questo perchè il sito nella sua vecchia versione ascolta sulla singola porta 80.

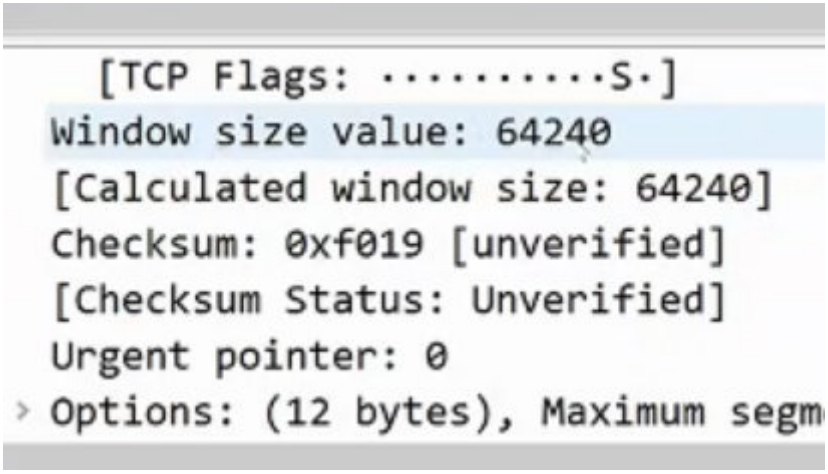
Se avviamo wireshark in ascolto sulla rete wifi, con il filtro `tcp port 80`, otteniamo la cattura dei pacchetti:



Segmento SYN

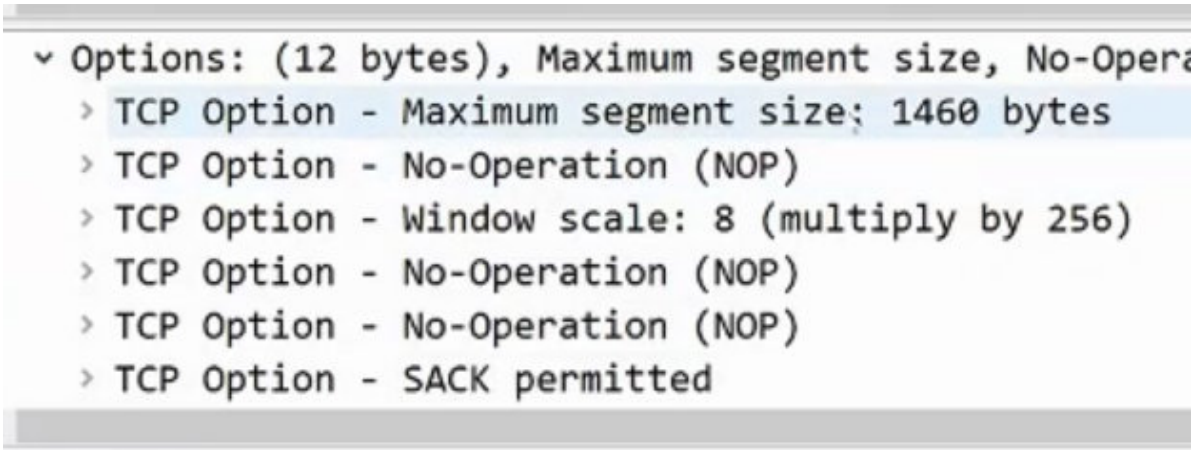
L'attivazione della connessinoe è data dai segmenti SYN e ACK: infatti il segmento SYN inviato dalla porta 59105, numero di porta aperto dal browser verso il numero di porta 80 verso la macchina remota. E' un **segmento di controllo** inviato dal client verso il server con un numero di sequenza posto a 0.

Tramite wireshark è possibile anche vedere la grandezza della finestra



Questo significa che il mittente può inviare fino a 64240 byte, corrispondente alla memoria disponibile nel buffer del ricevente. Non è però detto che inviando più di 64240 byte quelli in eccesso vengano scartati; questo perchè nel frattempo il buffer in ricezione potrebbe liberarsi.

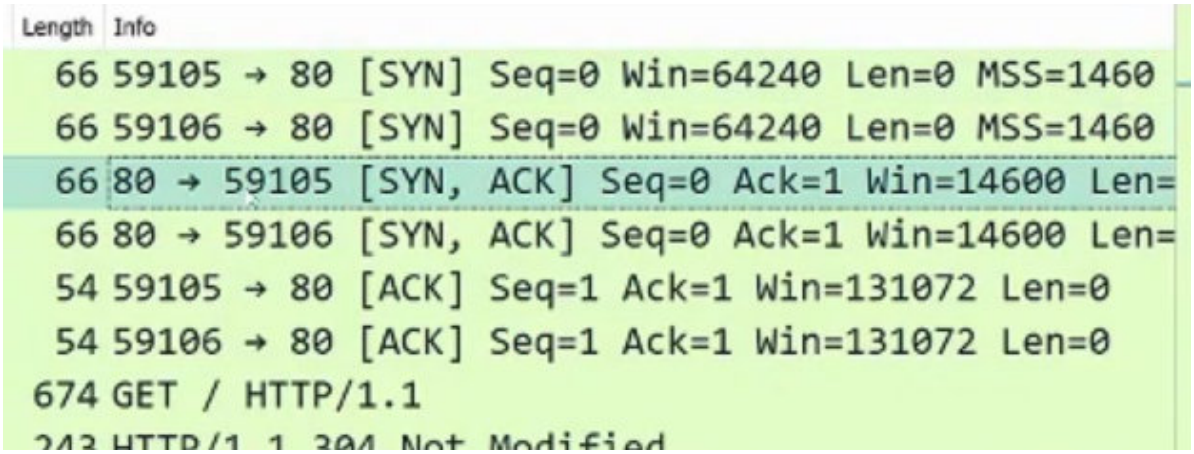
Possiamo inoltre vedere come siano presenti delle opzioni all'interno dell'intestazione:



da queste informazioni estrapoliamo la grandezza massima di un segmento, ovvero 1460 bytes; questo ci dice che le nostre informazioni verranno incapsulate all'interno di segmenti con un'area dati al più di 1460 byte.
Tra le opzioni troviamo anche SACK (selective acknowledgement), opzione che ci permette di abilitare la modalità di riscontro selettivo come variante del protocollo TCP.

Segmento SYN ACK

Possiamo osservare anche il segmento SYN ACK:



Questo segmento viene inviato dal server verso il client per accettare la richiesta di connessione e promuovere la connessione client-server.

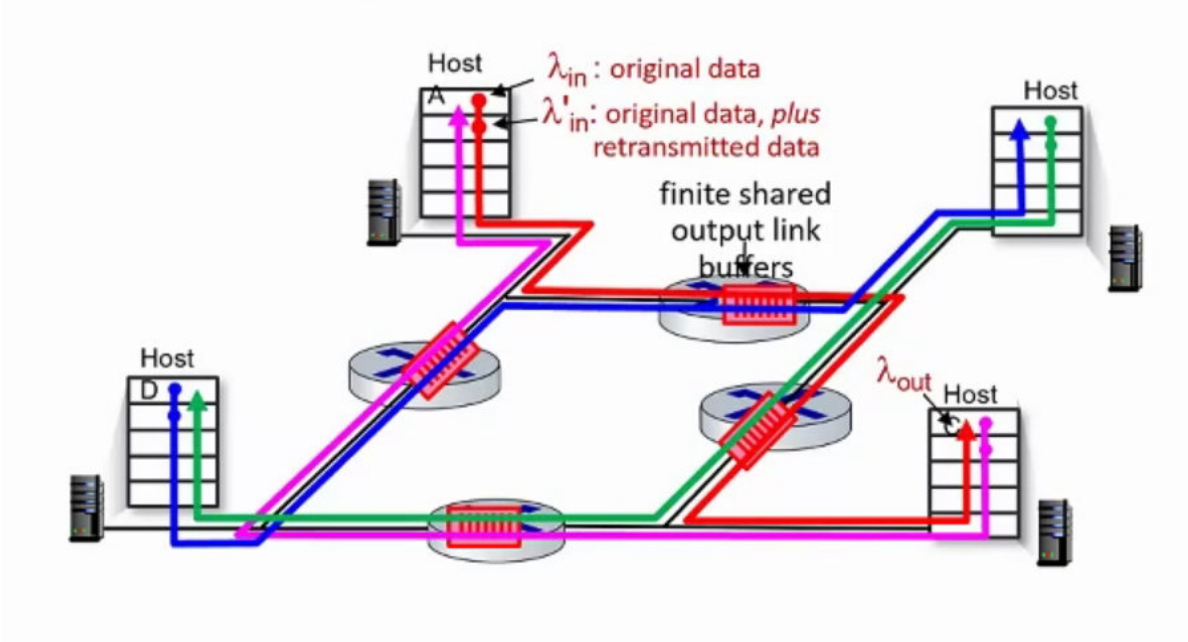
Infine si chiude la 3-Way Handshake con il segmento di riscontro:

```
 66 59105 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
 66 59106 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
 66 80 → 59105 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=
 66 80 → 59106 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=
 54 59105 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
 54 59106 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
674 GET / HTTP/1.1
243 HTTP/1.1 304 Not Modified
```

Notiamo anche che il browser apre due connesisoni: una dalla porta 59105 ed una da 59106; i browser moderni attivano due connessioni per migliorare l'efficienza. Inoltre, se un'ulteriore connessione servirà successivamente, la connessione è già pronta all'uso.

Cause delle congestioni: scenario 3

Si manifesta un nuovo effetto di congestione nel momento in cui la rete diventa più complessa:



In questo scenario abbiamo 4 router condivisi da diversi canali logici. E' un caso particolare in cui ogni router condivide due canali logici e se la larghezza di banda su questi link è R, ogni canale logico avrà a disposizione **al più R/2** in una situazione in cui i canali si suddividono in maniera equa la larghezza di banda. Il problema è che se consideriamo il primo router (in alto al centro), esso è il primo router che viene attraversato dal canale che va da A a C, ma è il secondo che viene attraversato dal canale da D a B.

Questo router sicuramente con carichi alti inizierà a prediligere il traffico che proviene da A, perchè man mano che aumenta il carico i pacchetti che provengono da D potrebbero venire persi dall'altro router. Lo stesso discorso vale per gli altri routers.

Di conseguenza, capiamo che i router, sotto condizioni di alta saturazione della rete, prediligono i pacchetti che arrivano dagli host "più vicini", o meglio che attraversano il minor numero di altri router (oltre se stesso).

Quando ci accorgiamo che le risorse di rete vanno verso la saturazione dobbiamo abbassare λin.

Come capire quando la rete è in saturazione?

Il problema è di natura diversa rispetto a quello affrontato nel controllo del flusso; in quel caso il ricevente ha pieno controllo del buffer; di conseguenza lo spazio residuo può essere notificato al client in maniera esatta, ed il server può usare quell'informazione per regolare il flusso.

Quando si opera a livello di rete, i buffer sono condivisi, e quindi la cosa è più complessa.

Prima soluzione

Questa soluzione prevede **l'inferenza** (*Inferenza statistica, procedimento di generalizzazione dei risultati ottenuti attraverso una rilevazione parziale per campioni all'intera popolazione da cui è stato estratto il campione.*) dello stato di funzionamento della rete dal livello di trasporto.

Se il livello di trasporto osserva dei timeout, interpreta quei timeout come un **rallentamento da parte della rete**, e quindi può utilizzare quell'informazione per regolare l'attività del mittente.

Il TCP usa questa soluzione.

Seconda soluzione

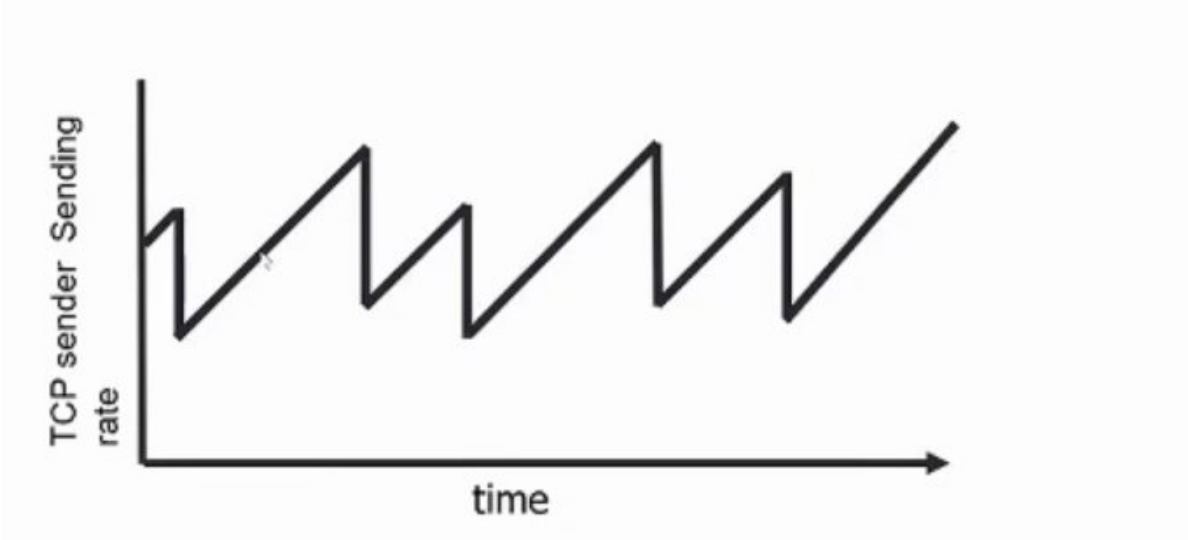
Un'altra soluzione è quella di far partecipare la rete: in questo caso i router partecipano in modo attivo al controllo della congestione, perchè **notificano in maniera diretta** che la rete è in condizione di congestione. Il TCP prevede anche questa modalità di funzionamento in combinazione con il protocollo IP.

Questa tecnica viene chiamata **TCP ECN, ovvero Excplicit Congestion Notification**. Il ricevente invia al mittente un segmento con il bit **ECN** posto ad 1, che indica che il ricevente ha ricevuto i dati, ma sono stati segnalati come dati che hanno attraversato una rete satura.

Controllo della congestione in TCP senza ECN

Poichè dobbiamo regolare l'attività del mittente, abbiamo come necessità la stessa necessità riscontrata nel controllo del flusso: ridimensionare continuamente la finestra di ritrasmissione. Nel caso del controllo del flusso abbiamo una variabil che viene costantemente aggiornata **con i dati trasportati dai segmenti (campo rcvwindow)**.

Nel caso della gestione della congestione non abbiamo un campo analogo (flag), ma l'approccio su cui si basa il TCP è il **probing**, ovvero verificare sempre lo stato della rete cercando di aumentare in modo progressivo il throughput finchè non si verifica una situazione di anomalia (ad esempio timeout).

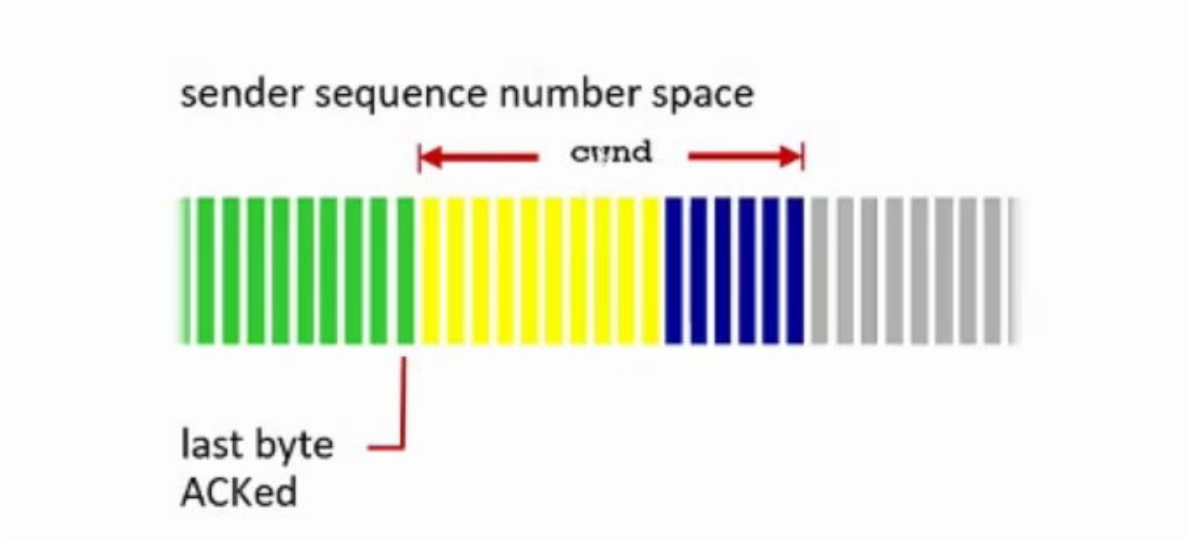


Il rate di trasmissione del mittente inizia con valore basso, e cresce fino al verificarsi di un'anomalia che segnala la presenza di una congestione.

Ogni volta che viene inviato un segmento e viene ricevuto un riscontro, la finestra di trasmissione usata dal TCP viene incrementata **di una unità** per ogni riscontro ricevuto. Quando viene rilevato un problema di trasmissione, la finestra viene **istantaneamente dimezzata**.

Il comportamento complessivo viene chiamato **AIMD - Additive Increase Multiplactive Decrease**.

Esempio



Viene usata una variabile **cwnd** i cui valori non sono notificati **da chi riceve**, ma la variabile viene gestita dal mittente; il valore viene incrementato o decrementato sulla base di eventi che il mittente osserva. Questa variabile ci dice quanti segmenti può spedire senza riceverne il riscontro. Con questa finestra il mittente regola il suo throughput.

Il funzionamento è analogo a quello del controllo del flusso, ma la variabile viene modificata in modo diverso.

Non possiamo avere una differenza **tra l'ultimo byte spedito** e **l'ultimo byte riscontrato** che sia **maggiore della variabile cwnd**: `LastByteSent - LastByteAcked <= cwnd`

Calcoliamo il throughput

Possiamo calcolare il throughput tramite la formula: `TCP rate = cwnd / RTT` bytes/sec

TCP slow start - controllo della congestione

Il TCP usa un algoritmo che si suddivide in due fasi:

1. slow start
2. congestion avoidance

> in realtà è la seconda fase che controlla la congestione, la prima fase è di probing.

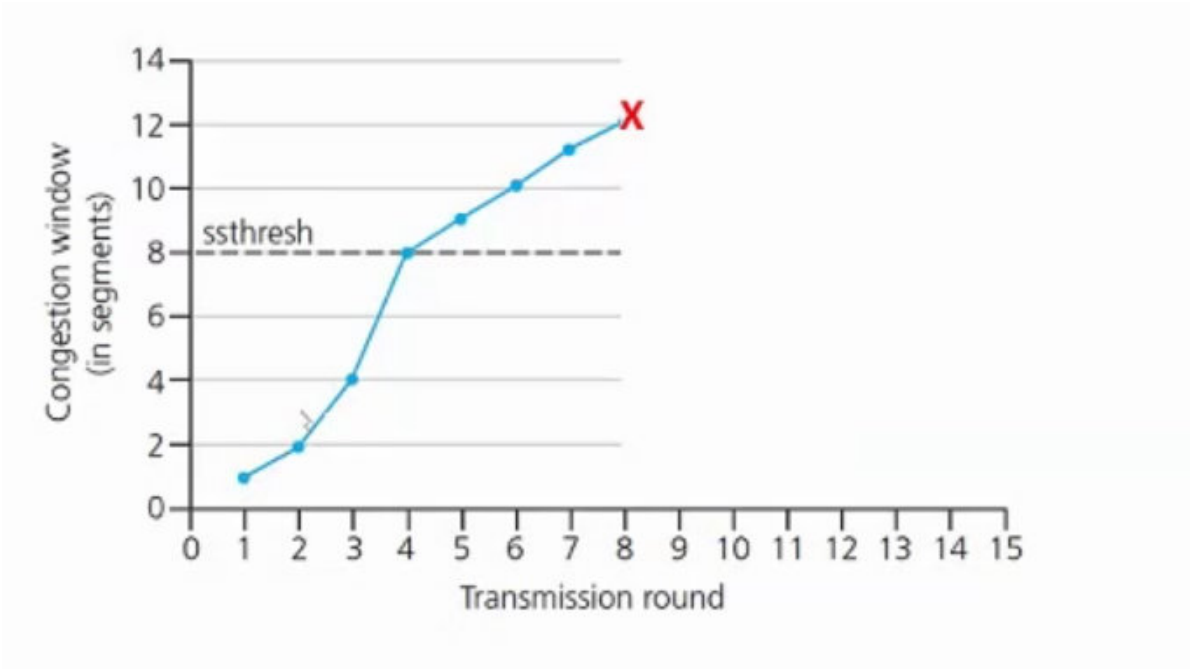
Slow start

Quando il client invia il messaggio di richiesta HTTP, non potrà inviare più di un segmento, essendo la finestra di congestione inizializzata a **1 MSS**, ovvero un solo segmento di dimensione massima al massimo segmento concesso per quel link.

Di conseguenza deve attendere un ACK per spedire un messaggio di richiesta successiva. Se il messaggio è grande il messaggio viene incapsulato in più segmenti, ma inizialmente **se ne può inviare solo uno**. Dopo aver ricevuto il primo riscontro possiamo spedire 2 segmenti; ricevuto il riscontro dei 2 segmenti è possibile spedire 4 segmenti, e quando saranno ricevuti i 4 riscontri la finestra passerà ad 8, e così via.

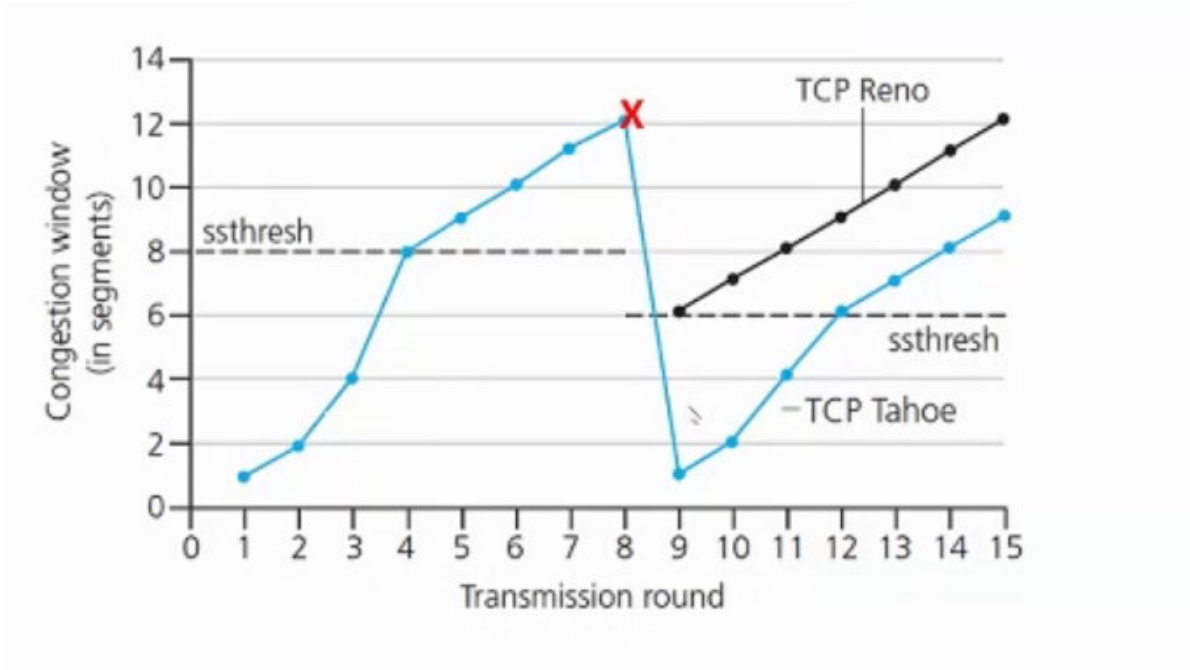
Questo comportamento di slow start ad un certo punto si arresta, ed inizia la fase successiva:

Congestion avoidance



Possiamo chiaramente vedere le due fasi in questa immagine: inizialmente lo sviluppo sembra esponenziale, ma dopo una certa soglia la crescita è lineare, fino ad arrivare alla perdita di un pacchetto. Per ogni gruppo di segmenti spediti, la finestra di congestione viene incrementata di **una singola unità totale**, e non di **una per ogni riscontro**.

Nonappena si presenta un'anomalia, il TCP cambia il valore della finestra di congestione:

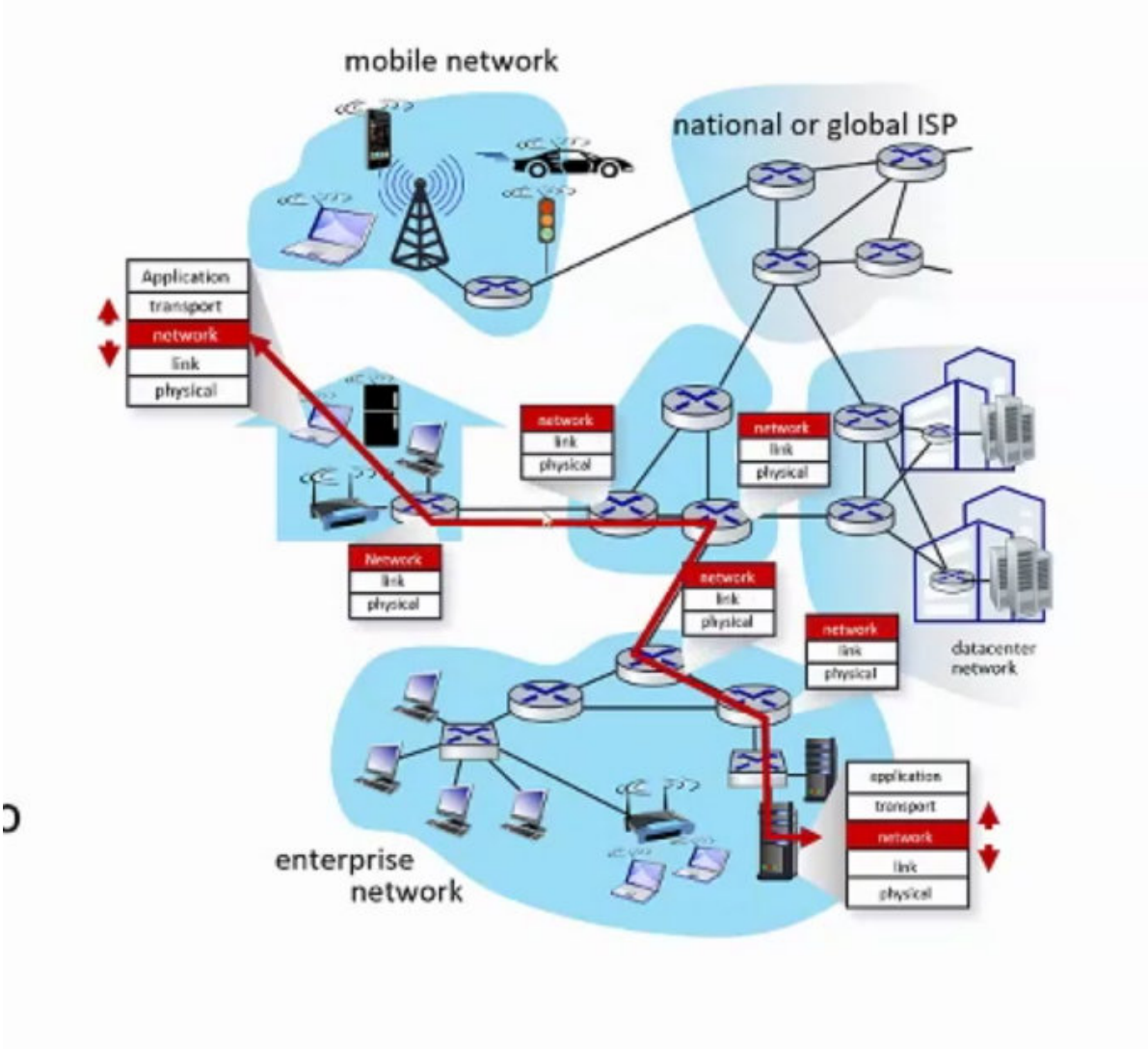


Quando si verifica questa anomalia, il valore di soglia che determina il valore di inizio della fase lineare viene posto alla **metà del valore corrente** al momento del verificarsi dell'anomalia. Ad esempio il valore al momento dell'anomalia è 12, la soglia viene posta a 6.

Livello di rete

🏁 1:14

Il compito del livello rete è quello di veicolare i segmenti che vengono prodotti a livello di trasporto: Il mittente incapsula i segmenti in un datagram del livello rete, ed il livello rete fa in modo che il datagram attraverso i router arrivi alla macchina di destinazione ed il protocollo di livello rete estrae il segmento per consegnarlo al livello di trasporto.



Ogni router che viene attraversato partecipa all'inoltro dei datagram grazie alle informazioni di controllo all'interno dei datagram stessi, che indicano dove quel particolare datagram deve essere recapitato. Il router deve essere in grado a quale dei link di uscita deve affidare il datagram ricevuto.

Da un lato è necessario che nel datagram debba essere presente l'indirizzo IP dell'host di destinazione, dall'alta sul router devono essere presenti delle informazioni utili affinchè il router possa capire dove instradare il datagram.

Il livello di rete ha due funzioni importanti:

- **Forwarding - inoltro** spostare il datagram da un link di ingresso verso un link di uscita (tramite le informazioni di controllo sul datagram e tramite le informazioni presenti sul router).
- **Routing - instradamento** costruzione del percorso che consente ad un host di inviare un datagram fino all'host di destinazione.

Per ciascun deve essere noto il percorso che un datagram deve seguire; la costruzione del percorso viene affidata agli **algoritmi di instradamento**.

Analogia con il mondo reale

Un'analogia per **l'inoltro** potrebbe essere quando siamo in auto e dobbiamo raggiungere una destinazione: **l'inoltro** è la scelta di una strada ad un incrocio, che viene fatta sulla base di una **preventiva pianificazione del percorso**, che sta a rappresentare il **routing - instradamento**.

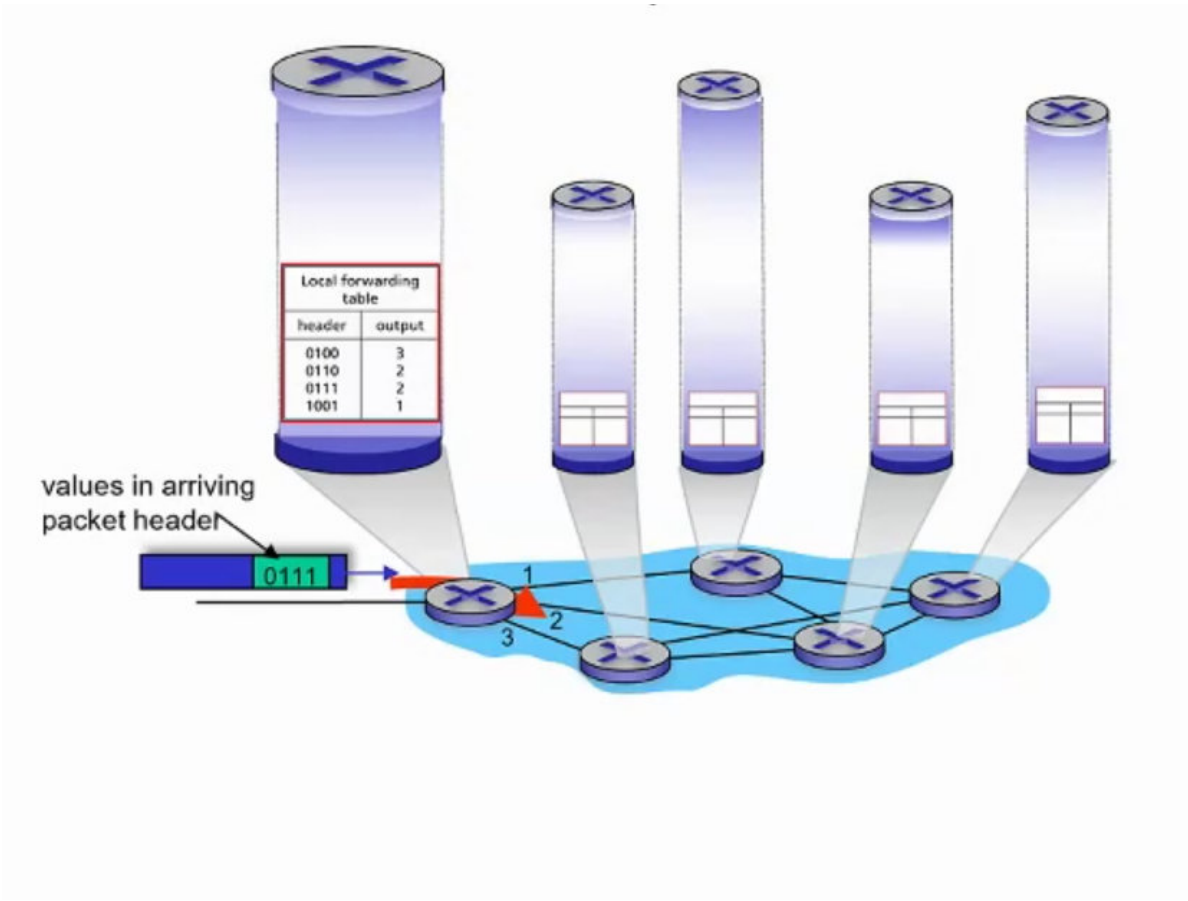
Data plane - Control plane

Le funzionalità che riguardano l'inoltro sono funzionalità locali ad ogni router, e sono assegnate a ciò che viene chiamato **Data Plane - Piano di dati**. Questa distinzione ha particolarmente piede con le moderne architetture di rete. Quindi il data plane si occupa dell'inoltro e deve farsi carico di analizzare l'intestazione del pacchetto in entrata per capire quale data link selezionare per l'inoltro.

Il **Control plane** abbiamo la logica a livello di rete per la costruzione mediante algoritmi dei percorsi di cui abbiamo appena parlato; la costruzione di un percorso può avvenire in modi diversi:

- **Approccio decentralizzato** i router stessi erano delegati alla costruzione dei percorsi, comunicando tra di loro.
- **Impiego delle SDN - Software defined networking**, che è la tecnica impiegata nelle architetture moderne.

Pre-router control plane - Approccio vecchio

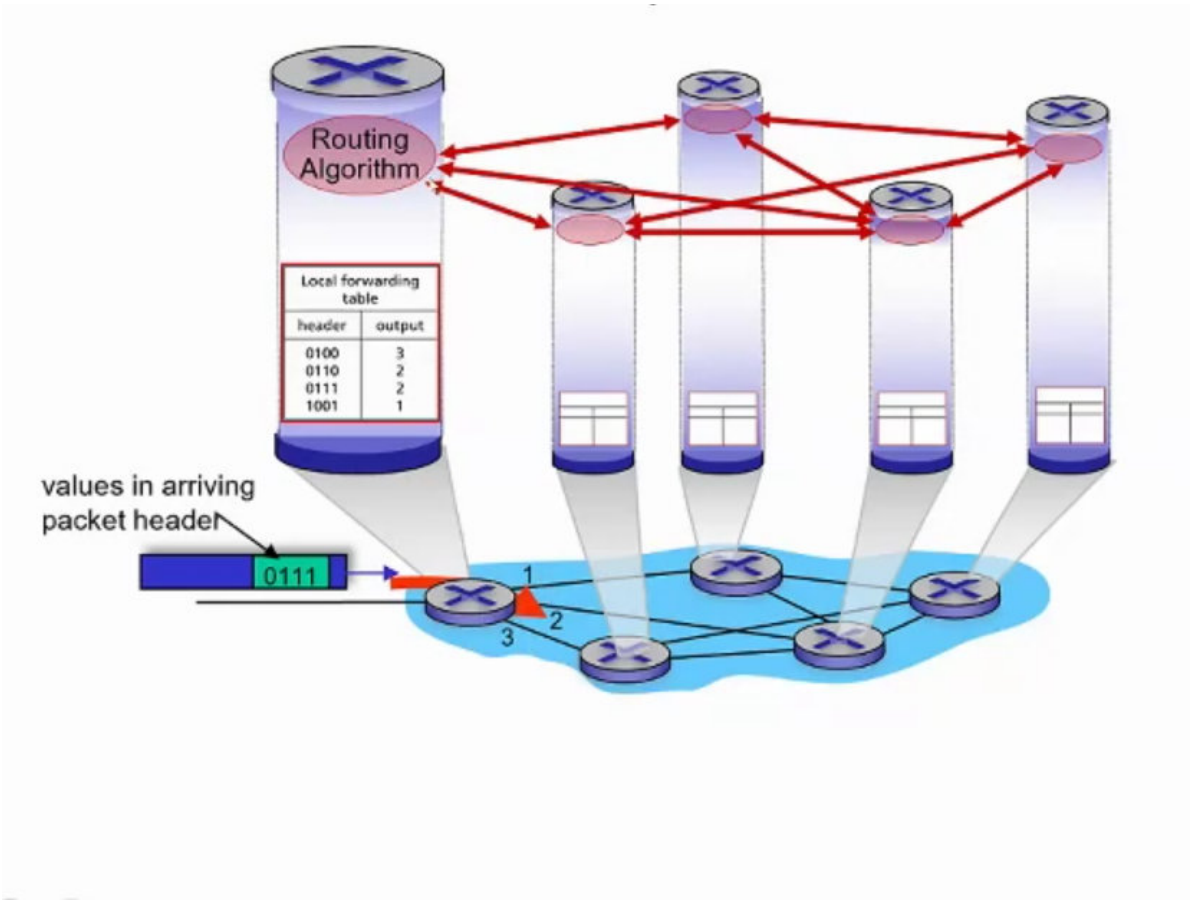


Il data plane prevede l'utilizzo di **tabelle di inoltro** che indicano al router qual è il link di uscita da utilizzare. E' quindi una tabella informativa per permettere al router di smistare il traffico in uscita.

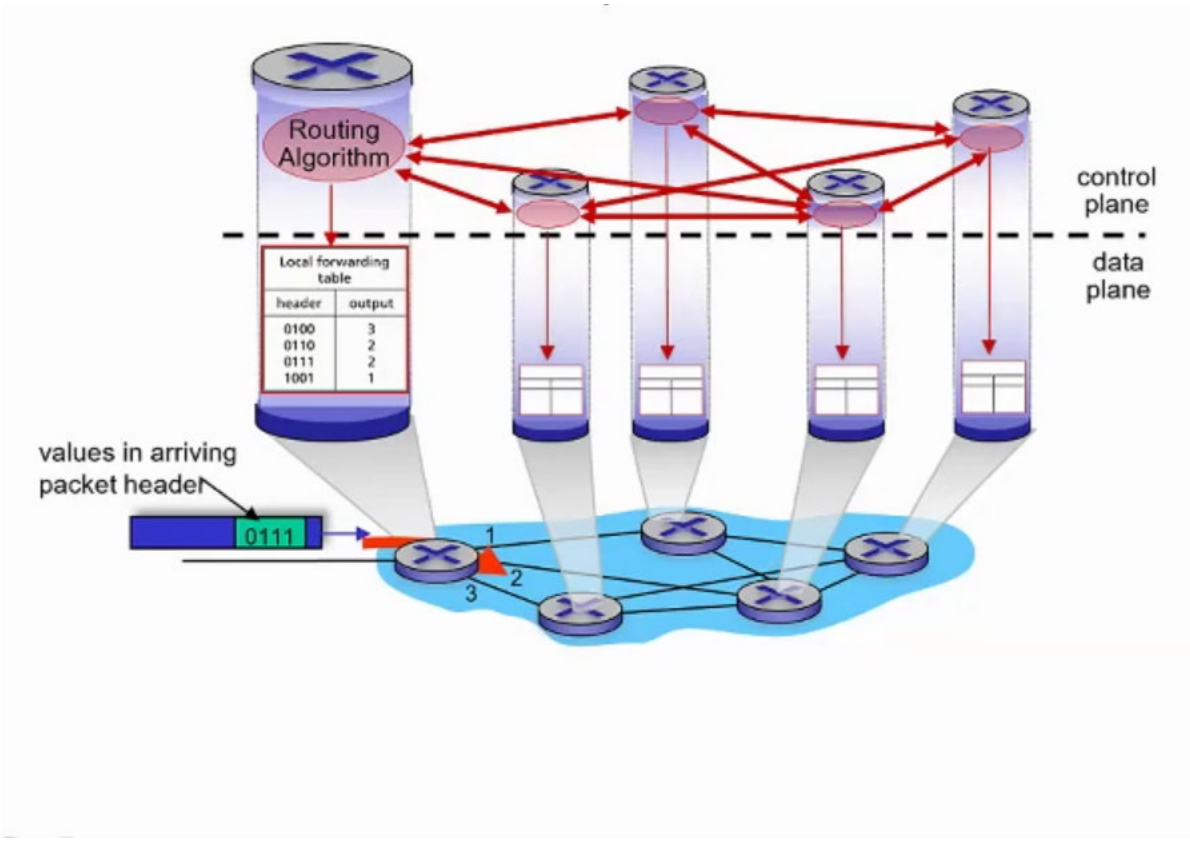
Chi costruisce queste tabelle?

La costruzione di questa tabella deve essere fatta in modo tale che per raggiungere una destinazione il router deve essere in grado di capire esattamente quale link di uscita usare, altrimenti si potrebbe selezionare un link errato.

I router dialogano per capier quali sono i percorsi che devono essere seguiti per raggiungere le diverse destinazioni:



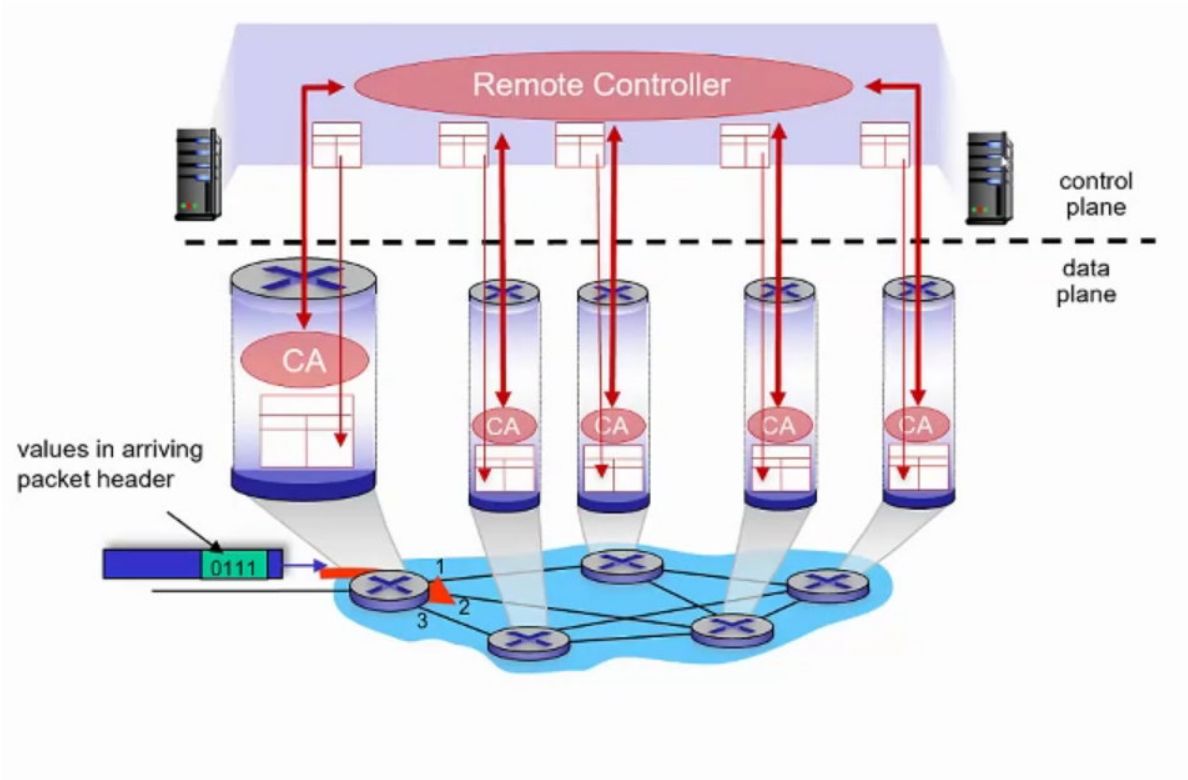
Gli algoritmi tipo che vengono eseguiti sono quelli sul percorso minimo dei grafi. Queste informazioni raccolte dai router permettono di elaborare i dati e costruire le tabelle. Di conseguenza i router producono ed aggiornano costantemente le tabelle che permettono di far funzionare le comunicazioni.



Vediamo quindi la distinzione tra **control plane e data plane**.

Approccio SDN - Software defined networking - approccio nuovo

In questo approccio abbiamo le tabelle di instradamento come nel caso precedente, ma il control plane è **completamente delocalizzato**; viene infatti eseguito da macchine remote, tipicamente all'interno di un data center:



Anche in questo caso vi è una continua comunicazione tra i router ed i server, in modo da avere le tabelle quanto più aggiornate possibile. Gli algoritmi che prima venivano eseguiti all'interno dei singoli router, ora sono spostati all'interno di server, di cui si sfrutta l'enormemente maggiore potenza di calcolo.

Network service model - modelli di servizio

Utilizzando il control plane e meccanismi di forwarding possiamo dare vita a diversi servizi: il servizio potrebbe avere un impatto sul singolo pacchetto garantendone la consegna (affidabilità), oppure si può garantire non solo che il pacchetto arrivi a destinazione, ma anche in un certo tempo (ritardo).

In riferimento ad un **flusso di pacchetti** la rete portebbe garantire la consegna ordinata, o una banda minima.

Internet non consente di prenotare alcunchè sulla banda, quindi la banda condivisa non gestisce perdite, non prevede la possibilità di rispedizione, non prevede meccanismi di riordinamento ne di gestione Timing.

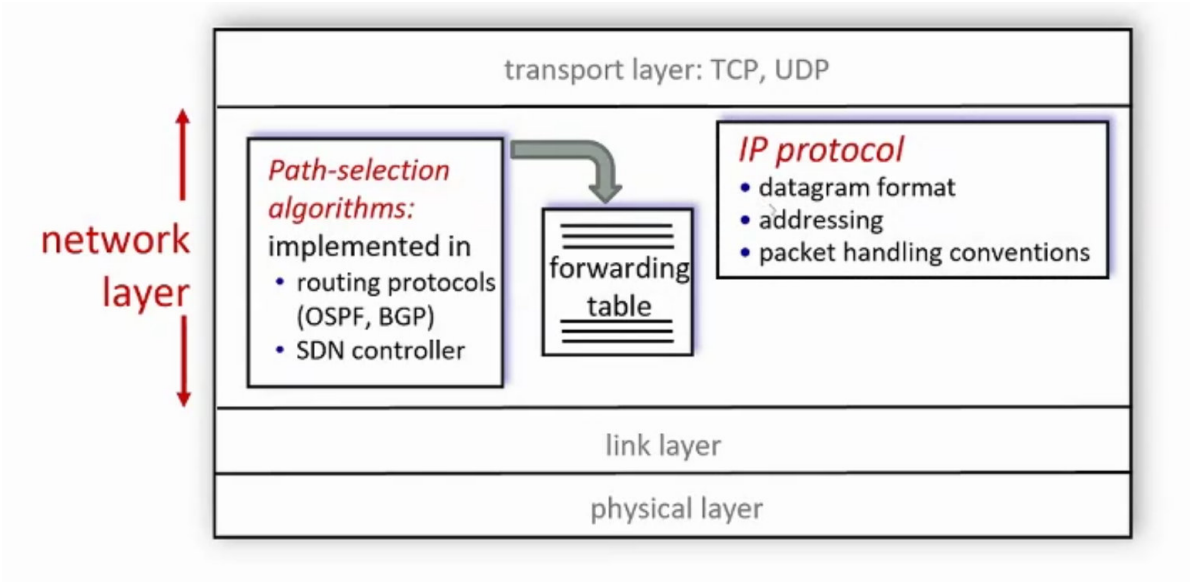
Di conseguenza, diciamo che internet offre un servizio di tipo **best effort**, ovvero che non fa nulla. Non garantisce l'affidabilità della trasmissione, non ha meccanismi di timing ne di ordinamento dei pacchetti. **A livello di rete dobbiamo aspettarci la capacità dei pacchetti verso la destinazione.**

Questo servizio è comunque utile, perchè per certi versi è un servizio primitivo, quindi è molto efficiente ma anche flessibile: da la possibilità di costruire meccanismi più complicati ai livelli superiori: ad esempio in TCP aggiungiamo molte funzionalità ed affidabilità.

Grazie alla possibilità di sopperire alle mancanze della rete internet è possibile avere ritardi contenuti: possiamo distribuire i server e cache in prossimità degli utenti, ecc.

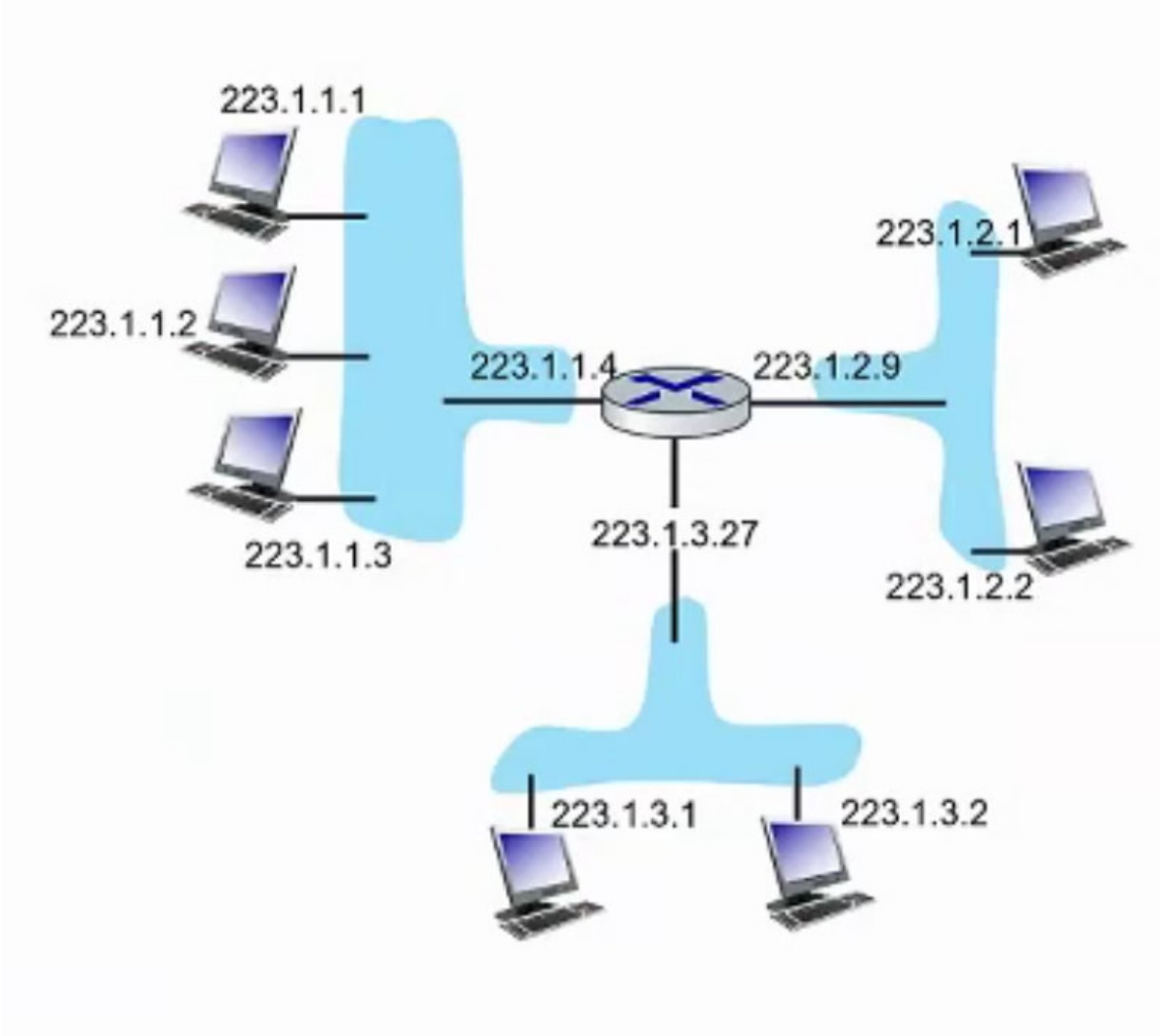
Network Layer: internet nel dettaglio

Da un lato abbiamo gli algoritmi per la selezione dei percorsi di cui abbiamo parlato prima:



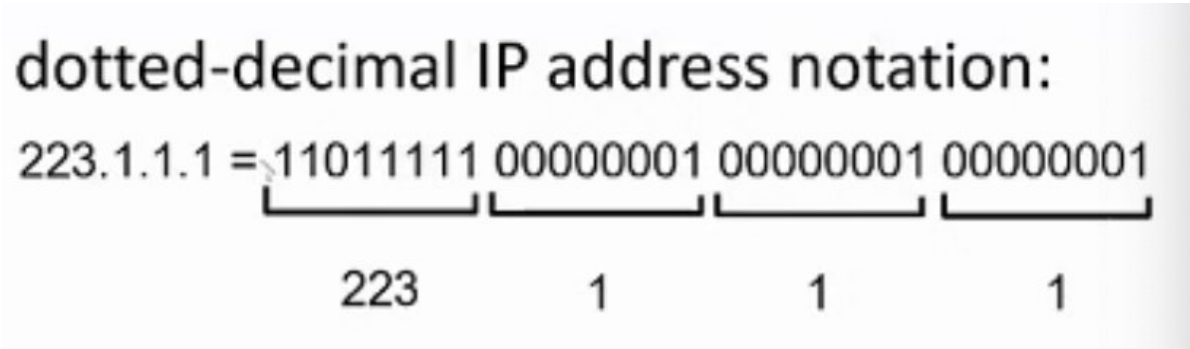
che sono usati per costruire le tabelle di inoltro che a loro volta sono usate per l'inoltro di datagram IP; il protocollo IP definisce il formato dei datagram.

IPv4 addressing: introduzione



L'indirizzamento è fondamentale per poter identificare le macchine e poi instradare i pacchetti fino alla destinazione. Ci soffermiamo sulla versione 4 del protocollo IP: IP ha subito un'evoluzione importante, verso la versione 6 che permette di avere un maggior numero di indirizzi.

Gli indirizzi IP sono indirizzi che assegnamo alle interfacce di rete di una macchina; in versione 4 sono numeri interi da 32 bit, che rappresentiamo in uno specifico modo:



> notazione decimale a punti

In questa notazione i 32 bit sono suddivisi in **4 byte** ed ognuno di questi byte è rappresentato in decimale.

Gli indirizzi ip (precisiamo) sono assegnati **alle interfacce di rete, e non alla macchina**; infatti un host può avere diverse interfacce di rete.

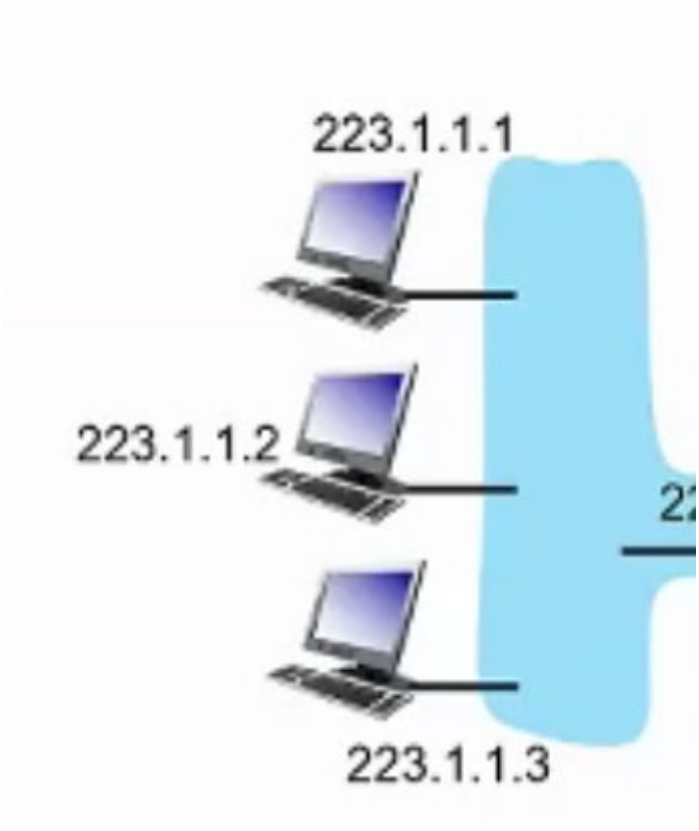
Come sono connesse le interfacce di rete? Nel caso di un collegamento wired, le interfacce sono collegate ad un dispositivo 2 dell'OSI: lo **switch**. Lo switch è poi collegato al router. Lo switch consente alle macchine che sono sulla stessa rete di dialogare tra di loro, ma se queste vogliono dialogare su macchine presenti su altre reti, interviene il router.

Potremmo avere anche bisogno di collegare delle macchine wireless ad internet (o in lan), ed il dispositivo che ci permette di farlo è un **base station wifi**.

L'oggetto che tipicamente chiamiamo **modem, contiene tutti questi componenti citati** : base station wifi, switch, router (ci consente di creare una sottorete locale con indirizzi privati), ed infine il **modem** vero e proprio.

Sottorete IP

E' un'insieme di interfacce di rete che possono dialogare direttamente, senza necessità di attraversare un router:

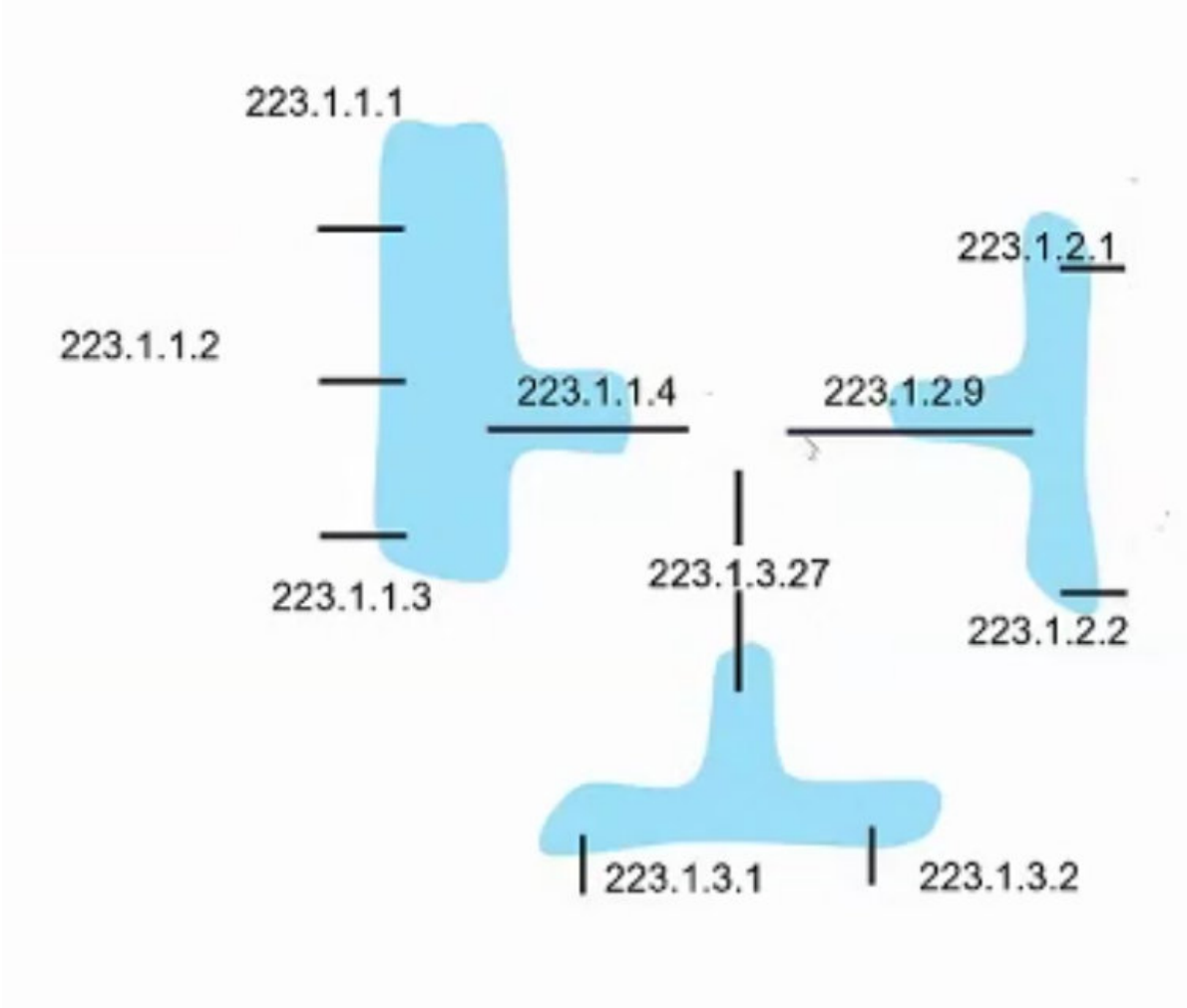


Gli indirizzi IP sono inoltre divisi in due parti:

- Sottorete
- Suffisso di host

Nella sottorete che vediamo in esempio, la parte di rete è la parte comune dell'indirizzo che troviamo in tutti gli indirizzi: `223.1.1`, la parte che varia è l'ultimo byte. Di conseguenza i dispositivi aventi una parte comune, rappresentano non solo una **sottorete**, ma sono anche caratterizzati da una **parte comune dell'indirizzo IP**.

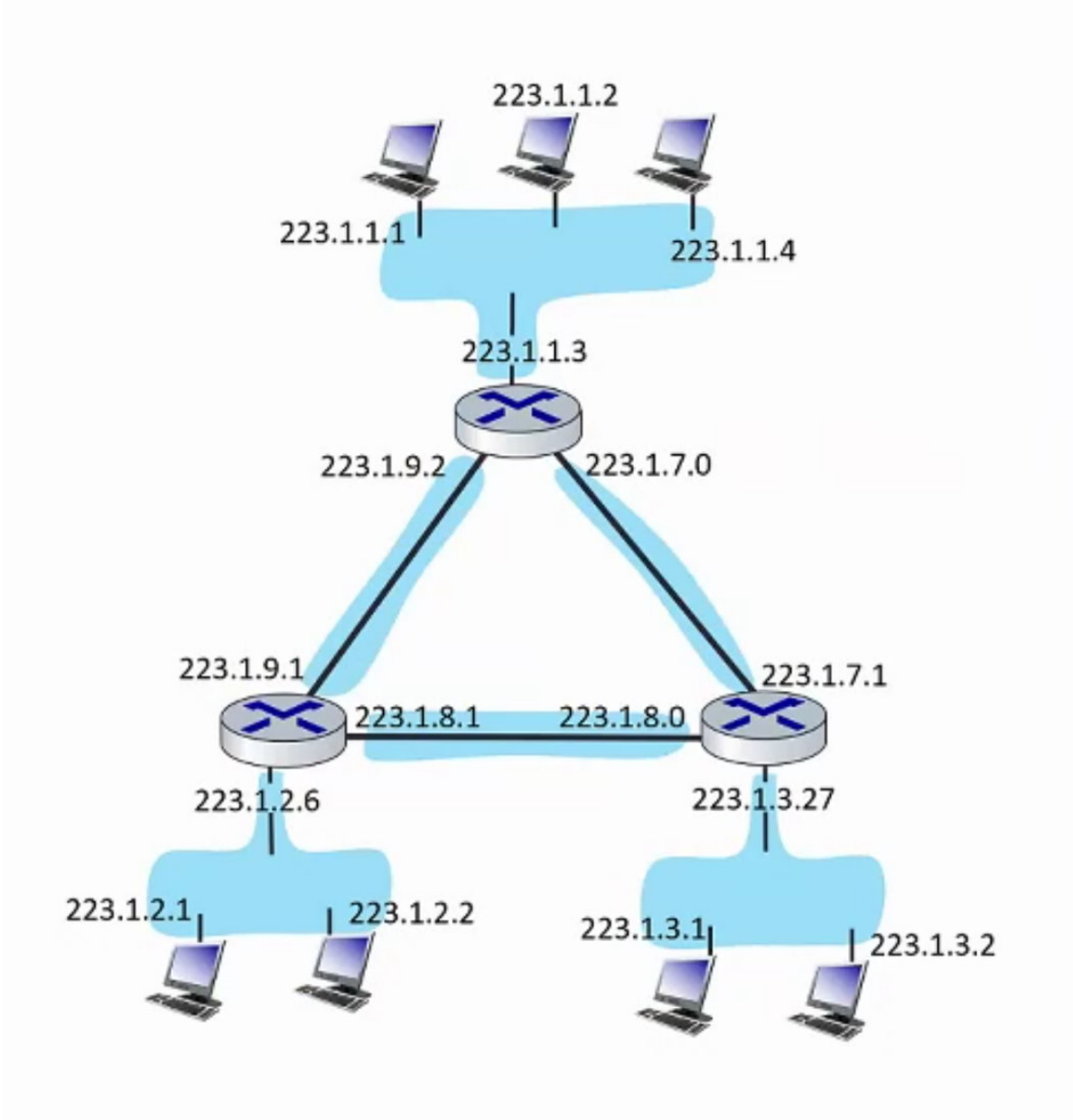
Come facciamo a capire quali sono le sottoreti in una rete complessa? E' sufficiente rimuovere gli host e routers:



Capiamo quindi (graficamente) dove sono presenti le reti locali (in blu).

Quante sono le sottoreti IP in questo esempio?

https://tva1.sinaimg.cn/large/008i3skNgy1gz4awaysr7j30tc0uqdhs.jpg



> L'indirizzo 223.1.7.0 è usato impropriamente perchè l'indice 0 viene usato per indicare l'indirizzo complessivo di rete.

Ci basta ignorare tutti i dispositivi e contare le aree in blu per capire immediatamente che le sottoreti dell'esempio sono **6**; anche le reti di collegamento (tra due sottoreti) sono delle vere e proprie sottoreti!

Fine lezione 25