Abbiamo visto l’algoritmo locale (distance vector) per la costruzione delle forwarding table, che funziona tramite protocollo ICMP. L’algoritmo funziona attivandosi quando si aggiunge un nuovo router alla rete, questo informerà i router a lui vicini, che informeranno lui su chi sono i loro vicini, e i cui vicini faranno la stessa cosa di livello in livello. Finché si parla dei router vicini gli indirizzi ip nella tabella di forwarding possono essere memorizzati in maniera completa, e man mano che ci si allontana si può ridurre il numero di bit significativi negli indirizzi ip salvati (secondo l’algoritmo di Bellman-Ford).

Il fatto che nessun router sappia con precisione la strada completa che il messaggio deve seguire (salvo magari quelli più vicini alla destinazione), non bisogna escludere la possibilità che con questo tipo di algoritmo si vadano a formare dei cicli chiusi (questo è il motivo per cui esiste il TTL). La qualità del risultato che si riesce a raggiungere dipende da quanto si diffondono le informazioni: in un certo senso il fatto che ci sia un aggiornamento costante (che causa la riscrittura delle tabelle ogni volta che viene aggiunto un router o che ne viene rimosso uno/ uno si guasta) migliora la tolleranza ai guasti della rete.

Possiamo passare a vedere brevemente un algoritmo diverso, l’algoritmo Link State (l’algoritmo globale). L’approccio è l’esatto opposto rispetto al distance vector: qui non c’è scambio di informazioni tra vicini ma tra tutti i router della rete, usando un’applicazione che viene chiamata broadcast. Sebbene anche in questo algoritmo si stabilisca subito quali sono i vicini immediati, questa informazione viene resa nota a tutti i router della rete; tutti i router, quindi, sapranno com’è fatta la nuova rete. Tutti si costruiscono un grafo dei router che costituiscono la rete con i loro canali di comunicazione e poi utilizzano l’algoritmo di Dijakstra per calcolare i minimi cammini della rete (= l’albero minimo che connette tutti i router della rete a partire da quello su cui viene chiamato l’algoritmo). Trovando l’albero minimo, fatto di un sottoinsieme dei canali di comunicazione, siamo sicuri che non ci siano cicli, e basta quindi seguire l’unico cammino reso possibile dall’albero quando si esegue l’instradamento. Questo algoritmo viene eseguito da tutti i router della rete, che si memorizzeranno solo le informazioni relative a loro.   
Le caratteristiche positive dell’algoritmo Link State sono che non ci possono essere cicli chiusi, inoltre tutti i router più o meno sanno quale sarà l’instradamento dei pacchetti (perché tutti i router eseguono il calcolo con dijakstra), inoltre grazie all’algoritmo di navigazione dei grafi utilizzato, tutti i percorsi di instradamento saranno “i migliori” dal punto di vista del numero di hop necessari.  
Gli svantaggi sono che il broadcast causa un grosso traffico nella rete, inoltre se c’è un guasto da qualche parte bisogna ricalcolare l’intero Spanning (?) Tree. Ma soprattutto, c’è da tenere conto che l’algoritmo di dijakstra ha una complessità polinomiale rispetto al numero dei nodi, quindi se si applica a tutti i router della rete internet (che sono miliardi), il tempo di calcolo per trovare un albero sarà parecchio lungo.  
Cosa succede se un router dà delle informazioni sbagliate durante lo scambio di informazioni ICMP verso i suoi vicini? Tipicamente questo ha un effetto negativo sulla capacità di instradamento, nel caso dell’algoritmo Link State potrei avere delle parti di rete che non funzionano (o che funzionano male) e altre parti che non sono influenzate. A essere influenzate negativamente saranno i router nelle immediate vicinanze di quello “guasto”.  
Nel caso dell’algoritmo distance vector l’effetto sarà peggiore: questo perché le informazioni sbagliate si diffonderanno un po’ per volta: l’errore potrà quindi raggiungere anche parti molto distanti della rete (immagina se una macchina a Genova pensasse di essere connessa direttamente col server di Google, questa informazione si può diffondere a livello globale e tutto il traffico diretto verso google finirà col dirigersi verso la macchina a Genova, causando la perdita di quasi tutti i pacchetti -> uno potrebbe farlo intenzionalmente come attacco informatico).

Nessuno dei due algoritmi può essere considerato migliore dell’altro in assoluto: l’idea che è stata messa in pratica è quella di usare a volte l’algoritmo distance vector e a volte il link state, ogni service provider può scegliere quale dei due usare (nel caso del link state il broadcast non viene fatto all’interno di tutta la rete, ma solo all’interno dei router interni al sistema del service provider). In questo modo da un lato si riduce la dimensione della sotto rete su cui si sta agendo, dall’altra, avendo la possibilità di scegliere quale algoritmo usare, ciascuno può scegliere quello che preferisce.

Ogni organizzazione si gestisce i router interni alla propria organizzazione e poi può eventualmente scambiare datagrammi con quelli di un’altra organizzazione (nei cosiddetti “punti di presenza”, tramite i router di frontiera).

Per gestire la problematica dello scambio di datagrammi tra router di diverse organizzazioni, viene introdotto un altro protocollo chiamato OSPF, che permette di tradurre da un sistema all’altro (da distance-vector a link-state e viceversa).

Capito come funziona l’instradamento, vediamo qualche dettaglio in più sul protocollo IP, sempre per la versione 4. Riprendiamo più in dettaglio l’header:

Il primo campo contiene alla prima riga il numero di versione (4 o 6), poi la lunghezza dell’header, il tipo di servizio e poi 16 bit di lunghezza del datagramma (i datagrammi possono essere lunghi al massimo 2^16 byte).

Alla seconda riga abbiamo 16 bit di identificatore, 3 bit di flag e altri 13 di offset di frammentazione. A cosa servono questi tre parametri? Si ricorda che il livello inferiore ad IP è il DATA LINK, e, se usassimo il protocollo ethernet a quel livello, questo dovrebbe avere una definizione tale che gli permette di mandare l’equivalente di datagrammi ma la cui lunghezza massima è di 1500 byte. Quindi come facciamo a mandare datagrammi più lunghi di 1500 bytes (per esempio 4000) se non ce lo permette il data link? La soluzione è frammentare. Si suddivide il datagramma in pezzi minori di 1500 byte (nel nostro esempio potremmo frammentarlo in quattro parti da 1000 bytes), questi pezzi avranno lo stesso identificatore nel campo ID (perché fanno parte dello stesso datagramma), ma cambieranno nel campo dell’offset di frammentazione (il primo pezzo avrà 0, il secondo 1000, il terzo 2000 e il terzo 3000). L’offset serve per ricostruire il datagramma originario una volta arrivato a destinazione, i flag indicano se c’è la frammentazione e in tal caso anche quando termina.

Il modo che è stato scelto per gestire la frammentazione della versione 4 di IP è che chi invia il messaggio si preoccupa di effettuare la frammentazione per quanto riguarda il primo hop (i datagrammi devono essere inviabili verso il primo router). Ad ogni passo, i router successivi che fanno l’instradamento devono verificare che la dimensione del datagramma completo sia corretta, anche per poter inviare il messaggio verso il router successivo.

Il numero che indica la dimensione massima dei messaggi inviabili col livello 2 è chiamato MTU; nel caso dell’ethernet, che è quello usato da quasi tutti, è 1500. Anni fa è stato proposta una versione espansa di ethernet chiamata Jumbo Frames con un MTU a 9000 bytes. Se si mischiassero router con diversi livelli due i router dovrebbero a un certo punto cambiare il tipo di frammentazione.  
Infatti è responsabilità dei router verificare che il datagramma sia già frammentato correttamente per l’MTU del cavo di uscita (la frammentazione, se serve, la possono fare tutti i router, ma ciò non si applica alla deframmentazione). A rimettere insieme i pezzi sarà la macchina del destinatario.

Quando si frammenta un pacchetto bisogna rielaborarlo, pertanto serve una certa potenza di calcolo e una certa quantità di memoria -> bisogna ripetere le informazioni presenti nell’header, cambiando soltanto i flag e l’offset di frammentazione.

Il carico di lavoro del destinatario (Hd) consiste nel ricevere tutti i frammenti e ricompattare il messaggio prima di darlo al livello di protocollo superiore (quindi il datagramma UDP va ricompattato prima di passarlo al livello UDP); quindi, si riempie un buffer di memoria con i pezzi che arrivano man mano e, se arrivano tutti, si può ricompattare il datagramma. Si noti che mentre si aspetta che arrivino tutti i pezzi, il buffer incompleto occupa memoria: il motivo per cui si affida la deframmentazione al destinatario è quindi di evitare di riempire ulteriormente la memoria dei router intermedi (andando a sovraccaricare la rete).

Il lato negativo è che basta perdere uno dei frammenti perché l’intero pacchetto sia da buttare.

Il fragmentation offset è analogo al sequence number nel protocollo TCP.

Alla riga successiva dell’header (la terza) IPv4 abbiamo il TTL (Time To Live), il Next Level Protocol, che è un’indicazione se al livello 4 c’è un protocollo TCP o UDP (o altro, ma in genere no) e il CheckSum (che in caso di frammentazione viene ricalcolato).

La quarta e la quinta riga corrispondono agli indirizzi sorgente e destinazione (che occupano 32 bit ciascuno). Poi abbiamo le eventuali Opzioni (facoltative) e poi abbiamo il payload, da interpretare sulla base del next-level-protocol.

A questo punto possiamo cominciare a discutere la versione 6 del protocollo IP.

L’header inizia allo stesso modo, con il numero di versione (che in questo caso è 6). Quello che segue viene chiamato “classe di traffico” e grossomodo ha lo stesso significato della type of service, poi c’è una porzione chiamata “etichetta di flusso” (flow label), che per il momento tralasciamo.

La cosa che manca nella versione 6 è la lunghezza dell’header, perché l’header è di lunghezza fissa (non ci sono parti opzionali).

Nella seconda riga da 32 bit, abbiamo i primi 16 che mi dicono qual è la lunghezza del payload (analogo alla lunghezza del datagramma della versione 4 ma c’è sottratta la lunghezza dell’header, che ha lunghezza costante). Poi abbiamo un’indicazione che è “Next Header”, che corrisponde al Next-Level-Protocol (ci dice cosa troviamo al livello 4, al momento nel payload del livello 3). Gli ultimi bit della seconda riga sono l’equivalente del TTL, e sono chiamati HOP LIMIT (chiarificando che non usa più la parola “time” perché ambigua visto che il numero rappresentato è una quantità di salti).

Poi ci sono l’indirizzo sorgente e l’indirizzo destinazione entrambi su 128 bit (quindi su 4 righe ciascuno). Dopo gli indirizzi, c’è il payload, la cui lunghezza è, come già detto, specificata nell’header.

Se si fa un confronto tra versione 4 e versione 6 si nota che nella versione 6 manca il checksum, per evitare di doverlo ricalcolare ogni volta ad ogni hop (ci si è resi conto che avere un checksum che deve essere ricalcolato da ogni router non conviene): i controlli di integrità sono fatti anche al livello di data link, quindi, non succede niente (oltretutto il controllo di integrità dell’Ethernet, il CRC 32, è molto più affidabile del checksum che usa solo un campo da 16 bit).   
Inoltre, e soprattutto, nell’header IPv6 manca tutta la parte di frammentazione (la seconda riga dell’IPv4): questo perché si è stabilito che fare la frammentazione sia una responsabilità interamente del mittente (e quindi sta a lui frammentare il messaggio in modo che vada bene per tutti gli MTU dei collegamenti della rete). Infatti, già il mittente sa che la maggior parte degli MTU è 1500 (per via della diffusione dell’ethernet); inoltre il protocollo ICMP relativo all’IPv6 contiene una segnalazione di errore in caso di frammentazione mancata (che contiene anche l’MTU del canale di comunicazione). Così, al mittente basta frammentare il datagramma e, se gli arriva la frammentazione mancata, deve solo frammentarlo di più e reinviarlo.   
La frammentazione e la deframmentazione sono responsabilità dei protocolli superiori a quello IP. L’header IPv6 ha lunghezza fissa di 320 bit. Per il resto, a parte la differenza nella lunghezza degli indirizzi (che permette un range di rappresentazione MOLTO più ampio), le proprietà sono le stesse tra IPv4 e IPv6 (a parte qualche cambio di nome).