Proseguiamo con la descrizione del protocollo DNS. L’RFC di questo tipo di protocollo è il numero 1035 (RFC 1035), o almeno è la versione attualmente in uso.

Abbiamo visto la struttura gerarchica dei server: Server Root, Top Level Domain, Server autoritativi (organizzabili su più livelli). Poi abbiamo introdotto un altro tipo di server detti “Server locali” e gli host che funzionano come client. Il Client fa un interrogazione al suo server locale, che interrogherà i server remoti, riceverà una risposta e la rimanderà al client.

Gli algoritmi studiati perché il server locale possa accedere alla gerarchia dei nomi e tradurre l’indirizzo IP sono due: l’algoritmo ricorsivo e l’algoritmo iterativo.

Nel caso dell’algoritmo ricorsivo l’idea è quella di relegare a qualcun altro la soluzione del nome rimanente: prendiamo come esempio il voler trovare l’indirizzo IP della stringa www.dibris.unige.it , per farlo l’host contatta il server locale. Se il server locale funziona con algoritmo ricorsivo esso inoltra la richiesta al root name server. Il root name server inizia scandendo il nome da destra a sinistra: individua la prima stringa fino al primo punto (it) e guarda se nella sua cartella c’è un top-level domain che si chiama “it”, se c’è inoltra la richiesta a quel top-level domain. Il top-level domain continua la scansione prendendo il secondo elemento da destra (unige) e guarda se nella sua cartella c’è un dominio autoritativo chiamato “unige”. Si prosegue allo stesso modo e il top level it manda la sua richiesta al dominio autoritativo unige. Questo guarda nella sua tabella e controlla se c’è un sottodominio autoritativo chiamato “dibris” e in tal caso inoltra la richiesta. Dibris si accorge di essere arrivato in fondo e l’unico modo per rispondere è che nel server autoritativo dibris esista la macchina “www”. Se è così questo invia indietro la risposta a unige, unige risponde a it, che la inoltra a root, che a sua volta la inoltra al Local DNS e così il DNS locale può rispondere al client.

Si ricorda che ciascun Top-Level domain conterrà tutti i domini autoritativi che sono stati attivati in quel domain e ciascuno dominio autoritativo conterrà tutti i sottodomini autoritativi attivati, la differenza è che mentre i Top Level Domain puntano ad altri SERVER (autoritativi), nei server Autoritativi sono contenuti gli indirizzi delle macchine che si possono raggiungere con quel dominio autoritativo.

Nel caso dell’algoritmo iterativo, invece, il client manda la richiesta al server locale (e fin qua è uguale), che guarda all’interno della sua tabella se è in grado di rispondere oppure no. Tipicamente non ne è in grado. Va in tal caso a vedere fino in fondo e nota che è necessario contattare il server root, quindi manda al client l’indirizzo del server root. Il client manda dunque al server root la richiesta, il root vede che il top-level domain è quello “it”, dunque manda come risposta l’indirizzo ip del top-level domain it. Quando il client manda un'altra richiesta a questo top level domain, questi risponde coll’indirizzo del server autoritativo unige. Il Client manda la stessa richiesta al server autoritativo unige, che quindi manderà indietro l’indirizzo ip del sotto-server autoritativo dibris. Il client manda la sua richiesta a dibris e qui otterrà come risposta l’indirizzo ip della macchina che gli interessa, presente nella cartella www.

In entrambi i casi il numero di datagrammi scambiati è esattamente uguale (cambiano solo mittente e destinatario). Tuttavia, nella versione iterativa i server fanno veramente poco lavoro (il local DNS Server non fa letteralmente un c\*\*\*o, il client potrebbe anche memorizzarselo l’indirizzo ip del server root), che viene tutto scaricato sul Client. Da questo punto di vista, almeno per il DNS conviene applicare l’algoritmo ricorsivo. D’altro canto, se tutti i client usassero un algoritmo completamente ricorsivo, i server sarebbero sempre occupati (perché prima di rispondere dovrebbero contattare tutti gli altri server del caso), mentre con gli algoritmi iterativi la risposta sarebbe immediata e potrebbe subito dopo dimenticarsi della richiesta (senza occupare memoria inutilmente).   
Per cui, in una configurazione con pochi server e tanti client, è molto più conveniente dal punto di vista dei server avere una soluzione di tipo iterativo. Per questa ragione, mentre per il DNS si usa l’algoritmo ricorsivo, per i server della gerarchia si usa l’algoritmo iterativo -> Il server locale DNS fa tutto il lavoro. Appena gli arriva la richiesta fa la richiesta al root server, poi ricevuto l’indirizzo del top-level domain fa la richiesta a questo, e poi allo stesso modo fa la richiesta al server autoritativo. Quando poi fa la domanda al sotto-server autoritativo e ottiene come risposta l’indirizzo della macchina www in “dibris” il server locale DNS può finalmente rispondere al client. Vengono quindi usati entrambi gli algoritmi.

Tutto questo si risolve in un carico di lavoro molto inferiore, soprattutto dal punto di vista della quantità di memoria necessaria.   
Il server locale è un punto di riferimento per un certo numero di client, ma che non sarà un numero gigantesco (il server dns potrebbe anche essere collegato solo a una piccola rete locale). Per questo motivo ha senso che lui funzioni in modalità ricorsiva (per facilitare la vita a quei “pochi” client).  
L’utilizzo dell’algoritmo ricorsivo nel server locale permette di ottimizzare ulteriormente le cose, perché gli permette di vedere la risposta finale (altrimenti con la modalità iterativa potrebbe soltanto mandare un messaggio), dunque la potrà memorizzare -> Il server DNS locale potrà fare uso di un meccanismo di caching e, andando a controllare il contenuto della sua cache, potrà evitare di dover andare a interpellare tutta la gerarchia di server ogni volta se arrivano più richieste per lo stesso dominio nella stessa rete locale (si pensi a quante traduzioni si risparmiano già solo memorizzando indirizzi come google.com).

Anche se i local DNS si chiamano così, nessuno ci impedisce di usare come DNS un server remoto, l’unico downside è che se tante persone si connettono a tale DNS remoto ci possono essere dei ritardi nella ricevuta delle risposte.

Nella cache dei DNS i dati possono rimanere qualche ora o qualche giorno, perché in caso di aggiornamento di un servizio (magari spostandolo da una macchina all’altra) o a seguito dell’aggiunta di nuove macchine questa è una tolleranza sopportabile per potervi accedere di nuovo usando lo stesso dominio (poiché nelle cash degli utenti rimarranno i vecchi indirizzi IP per delle ore se non fanno il reboot).

Il protocollo DNS è nato con una caratteristica all’origine che è quella di fidarsi delle informazioni disponibili all’interno della rete (e che tutte le macchine cooperassero). Ora questo non va più bene.

Il DNS è infatti un protocollo senza memoria: non si tiene conto dei messaggi che sono stati inviati dopo che si è mandata la richiesta. Questo dà la possibilità alle macchine che vogliono introdurre delle anomalie di farlo in maniera semplice. In particolare, una vulnerabilità è quella generata dal Cache Poisoning.   
Abbiamo visto che il DNS è un protocollo di tipo domanda e risposta a livello applicativo, che utilizza come trasporto il protocollo UDP (perché i Datagrammi che si inviano sono abbastanza piccoli, quindi così si può evitare di aprire una connessione coi server e ci si risparmia la three way handshake). Dunque, se a un client arriva una risposta questo assumerà di aver fatto una richiesta (perché che l’abbia fatto o meno se lo è già dimenticato, visto che non si tiene conto delle richieste già inviate), perciò la cache del DNS salverà la nuova associazione: così facendo è possibile avvelenare la cache del local DNS.

Per esempio, se un DNS server riceve la risposta “guarda che a [www.unige.it](http://www.unige.it)” corrisponde l’indirizzo xxx.xxx.xxx.xxx, questo salverà l’informazione nella cache e quando un client invierà richiesta di traduzione dell’indirizzo [www.unige.it](http://www.unige.it), il traffico del client verrà dirottato sull’indirizzo “avvelenato”. Se si fa una cosa del genere col sito di un social o ancora peggio di una banca, basterà emulare la schermata di accesso del sito (con un banale login e password) per rubare le credenziali dell’utente, che ignaro le inserirà pensando di essere sul sito reale.

--- Ci possono essere tanti indirizzi IP associati a un solo nome simbolico o tanti nomi simbolici associati allo stesso indirizzo IP ---

Per far funzionare questo cache-poisoning, poiché non c’è alcun sistema di autenticazione, basta preparare a mano il datagramma UDP in modo da far sembrare il mittente uno dei server della gerarchia.

Le versioni successive del protocollo DNS, comprese quelle attuali, hanno aggiunto dei meccanismi di protezione. Si ricorda infatti che i primi 16 bit dell’intestazione del datagramma DNS è un identificatore. Viene data l’indicazione di usare identificatori casuali e, anche se non viene memorizzata tutta la richiesta, viene memorizzato l’identificatore da 16 bit, in modo da ricevere risposte solo da chi usa lo stesso identificatore (se una macchina ha mandato tante richieste e non ha ancora ricevuto risposta, si tiene l’elenco degli identificatori usati per quelle richieste -> in questo modo anche se non si ricorda delle richieste inviate, la macchina sa quante ancora di esse sono senza risposta e per ogni risposta che riceve può controllare se è dovuta oppure no).  
Questo sistema è migliore del precedente, ma se l’attaccante è piazzato da uno dei server (il local DNS o la gerarchia) può vedere le richieste e memorizzare a sua volta gli identificatori. Quantomeno però se l’attaccante non è in uno di questi punti della rete può solo tirare a indovinare: il problema è che ci sono solo 65536 identificatori possibili, un numero di tentativi non difficile da raggiungere per un computer. Per sistemare questo problema, l’aggiornamento si è fatto sul protocollo UDP. Infatti, nel datagramma dell’UDP ci sono due porte: mentre la porta di destinazione è importante, la porta sorgente è una porta effimera (è anch’essa generata in modo random, viene ri-generato se la porta è già in uso ma questo è un dettaglio). Dunque, il numero che viene generato casualmente ed è usato come “sicurezza” è un numero su 32 bit che viene spezzato mettendone metà come identificatore e metà come porta effimera. A questo punto, con 2^32 opzioni possibili, se qualcuno anche solo provasse a mandare 2^32 messaggi in breve tempo sovraccaricherebbe la rete e renderebbe palese lo star lanciando un attacco (se sbaglia anche solo il numero di porta il protocollo UDP lo riconosce come un errore nel datagramma e lo scarta).

La versione attuale con le modifiche appena dette è un pochino meno facile da attaccare, questo sempre nell’ipotesi che l’attaccante non sia parte della normale rete utilizzata per il meccanismo DNS. Come fa un attaccante ad essere dentro al sistema? Ci sono vari modi, innanzitutto il provider è quello che è nella posizione migliore per fare una cosa del genere (però normalmente un provider non lo può fare legalmente e senza essere scoperto salvo sia dato il permesso da parte da un giudice). Il modo in cui un attaccante può entrare a far parte del sistema è registrando un proprio dominio autoritativo e mandando una risposta sbagliata di proposito (relativa ad altri indirizzi non presenti nella sua DNS), e avvelenando la cache, nel momento in cui qualcuno prova a connettersi al suo dominio.

Esiste anche la versione TCP del DNS, ma non è molto usata per via di tutti i ritardi causati dal three way handshake. È talvolta usata comunque la versione TCP quando i messaggi possono superare i 65 Kb (il limite dei datagrammi UDP). Un vantaggio del three way handshake sarebbe quello che evita lo spoofing degli indirizzi, ma la trasmissione delle informazioni diventerebbe poco efficiente.