Abbiamo visto i tre protocolli principali utilizzati nel sistema di posta elettronica, l’SMTP, il POP e l’IMAP.

Il protocollo che vediamo adesso è un protocollo un po’ più complicato rispetto agli altri, parliamo del protocollo FTP (File Transfer Protocol), descritto nell’RFC 959. Il protocollo FTP è complicato perché usa DUE connessioni. FTP si basa sul protocollo TCP a livello di trasporto e i suoi server utilizzano la porta interna 21. Tuttavia, questa connessione da sola non basta, questa è infatti la connessione di controllo: una volta stabilita questa connessione il client deve autenticarsi sul server. Una volta autenticati c’è la possibilità di dare dei comandi, che è la stessa definita dall’applicazione TELNET (come per il protocollo SMTP): viene inviato uno stream di byte il cui bit più significativo è sempre 0 (perché la codifica è in ASCII su 7 bit). I comandi che si possono dare sono: listare i file sul server (dir, equivalente all’ls su shell), comandi equivalenti al cd per spostarsi tra directory e i due comandi principali, quello di upload e quello di download. Sulla connessione di controllo viaggiano solo i comandi e le relative risposte/acknowledgement (tipicamente la risposta “200” indica “tutto OK”, mentre i messaggi di errore hanno usualmente valori dal 500 in su).   
Le informazioni effettive, invece, vengono inviate attraverso la connessione data (la seconda connessione), questa viene attivata per ogni istruzione che richiede il passaggio di qualche informazione (anche solo per il dir, la lista dei file viene inviata sulla connessione dati). Questo permette di evitare l’estensione del protocollo (come, ad esempio, attraverso la codifica MIME in RADIX 64 per il protocollo SMTP), mantenendo l’efficienza della connessione, infatti, questa seconda connessione funziona a 8 bit normali e qualsiasi tipo di dato può essere inviato attraverso essa. Sono stati definiti due modi di funzionamento per il canale dei dati: la prima delle due è la modalità “attiva”, la seconda, introdotta successivamente, è la modalità “passiva”.

In caso di modalità attiva, il server predispone una seconda porta di comunicazione, che corrisponde al numero 20, tuttavia su questa seconda porta di comunicazione il server si comporta come se fosse un client (dopo la bind, anziché fare la listen fa la connect, mentre il client dovrà fare la listen e la accept). Il problema è che tipicamente un client deve usare porte alte (effimere), perché le porte da 0 a 1023 sono utilizzabili solo dall’amministratore di sistema, quindi, per dare la possibilità di connessione ad un client che non ha i privilegi di amministrazione si stabilisce che il client userà una porta maggiore o uguale a 1024. È quindi indispensabile che il client comunichi al Server il numero di porta su cui sta facendo la listen.

Il canale di controllo è più o meno permanente (resta attivo finché o il client o il server non decide di chiudere la connessione), la connessione data invece non è permanente, rimane attiva solo finché vengono trasferiti dei dati. Se il server deve inviare altri dati in un secondo momento, la connessione va riaperta. Essendoci tutto questo overhead per aprire la connessione (ci vuole un three way handshake ogni volta), il protocollo FTP conviene quando è necessario l’invio di file di grandi dimensioni.

Supponiamo di voler usare questa applicazione per fare l’upload di un file sul server:  
Si deve aprire la connessione, autenticarsi sul server e poi scegliere quale file inviare dalla macchina locale e su quale directory del server fare l’upload. Dunque, saranno necessari due diversi comandi cd: uno che funzioni in locale sul client per selezionare il file da inviare e uno che funzioni sul server remoto per scegliere la directory di destinazione. Per poter fare questa operazione, devo avere il permesso di lettura sul file locale e il permesso di scrittura sul file (in realtà directory) remoto.

La modalità attiva, oltre a complicare la comprensione del protocollo, ne complica anche la realizzazione, specialmente nel caso in cui il client e il server debbano attraversare un firewall: l’idea di avere un client che si deve comportare anche da server e un server che si deve comportare anche da client va contro i classici metodi di filtraggio dei firewall. Per questo motivo, quando si sono diffusi i firewall, è stata introdotta la modalità passiva della connessione dati per il protocollo FTP.

Nella modalità passiva, non viene più usata la porta 20 dal server, ma solo la 21, e quando c’è da aprire una connessione dati il server fa la bind su una porta effimera (alta) e poi fa la listen e la accept su tale porta. A questo punto, tramite la connessione di controllo, il server comunica al client la nuova porta che è stata aperta per la connessione dati. Il client così farà una connect su questa nuova porta per poter effettuare il trasferimento dei dati.   
Questo è il metodo attualmente più utilizzato per il trasferimento dei file attraverso il protocollo FTP.

Altri dettagli: oltre al funzionamento normale, che prevede l’autenticazione dell’utente sul server, è stata predisposta anche una modalità di funzionamento diversa, che è chiamata “FTP anonimo”. Questo FTP anonimo permette di accedere al server FTP a qualsiasi client, anche se questi non ha un account. Il funzionamento è uguale a quello del protocollo in versione normale (con uguale porta), solo che l’username assegnato al client è “Anonymous”: questo username accetta qualsiasi password. Poiché ci si fidava abbastanza degli utenti quando è stato introdotto il protocollo, si consigliava agli utenti di accedere in anonymous inserendo come password il proprio indirizzo email, tuttavia nessun controllo era effettuato (si poteva mettere l’indirizzo di qualcun altro o qualcosa a caso).   
Quello che cambiava con l’utente anonymous erano i permessi di accesso ai files. Storicamente, sono stati sviluppati due approcci diversi: il primo approccio consiste nel mettere il file system del server accessibile in sola lettura, permettendo quindi di effettuare solo le operazioni di download (questo metodo permetterebbe di rendere disponibile un file a tutti senza dar loro la possibilità di modificare il file system del server).  
L’altro approccio possibile era la possibilità di avere il file system sul server in modalità “sola scrittura”, questo permetterebbe solo l’upload di file sul server, senza poter effettuare il download. Il motivo per cui è stato sviluppato questo sistema è per il caso di un’organizzazione che sviluppa software che vuole ricevere bug reports (così facendo gli utenti possono segnalare i bug senza autenticarsi da nessuna parte). C’è il rischio che con questo metodo qualcuno saturi il file system, ma per quello basta introdurre un sistema automatico sulla macchina server per fare pulizia di file vecchi o di spam.

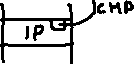
È sconsigliabile dare agli utenti anonimi accesso sia in lettura che in scrittura, perché qualcuno potrebbe uploaddare un malware e qualche altro utente potrebbe finire con scaricarlo, rendendo il server FTP una sorgente di diffusione di malware (e ciò è problematico perché nel caso degli utenti anonymous, la responsabilità delle loro azioni ricade sull’amministratore del server, mentre per gli utenti autenticati è diverso).

Nella versione definita dall’RFC 959 prevede che le password usate dagli utenti siano inviate in chiaro. Le versioni successive prevedono che sia utilizzato il TLS per cifrare la password dopo l’invio fino alla ricezione. La versione “sicura” del protocollo FTP in genere si chiama SFTP.

Lasciamo stare per il momento i protocolli a livello applicativo, e torniamo al livello 3, di network.

Parliamo del protocollo ICMP, descritto dall’RFC 792. Lo scopo del protocollo ICMP è di aiutare il protocollo IP, che è il protocollo principale a livello di rete (è un protocollo collaterale). In particolare, serve per fare controllo di funzionamento corretto della rete: può servire per fare debugging, per segnalare errori di trasmissione e per implementare la funzionalità del routing (cioè l’instradamento corretto dei pacchetti).

Tralasciamo il terzo aspetto e concentriamoci sui primi due. Il protocollo ICMP è in una situazione abbastanza anomala, è definito al livello 3, ma lì il protocollo principale è il protocollo IP, quindi l’ICMP si affianca all’IP, utilizzando le sue funzionalità per inviare datagrammi, ma non è comunque un protocollo di livello 4.



Il formato dei datagrammi è molto semplice, c’è un header che ha due campi principali, ciascuno da 8 bit, e questi due campi da 8 bit contengono due numeri: il primo dei due definisce un tipo (di datagramma ICMP) e il secondo contiene un codice (che differenzia all’interno del tipo). Vediamo alcuni dei tipi principali per fare debugging di rete e segnalazione degli errori: Echo Request e Echo Response.

L’idea è la seguente: un qualsiasi host (h1), può mandare un messaggio di tipo ICMP Echo Request a un qualsiasi altro host (h2) inviandolo come datagramma all’interno della rete, attraverso l’instradamento IP. Il protocollo è di livello 3, quindi l’header non viene aggiunto al posto di quello di livello 4.  
Quando l’host h2 riceve un datagramma di tipo Echo Request, questo risponde con un datagramma di tipo ICMP Echo Response (il cui contenuto è uguale a quello dell’invio, serve per fare il ping-pong). Questo permette ad h1 di misurare il round-trip-time. L’effetto di avere un meccanismo di comunicazione al livello 3 è che non è necessario specificare nessuna porta e che si è completamente indipendenti dalle applicazioni che girano sulle macchine (solo dal sistema operativo): in tal modo si bypassa completamente anche i protocolli TCP e UDP. Questo è il protocollo generalmente utilizzato per l’applicazione “ping” sui sistemi UNIX-like.

L’altro utilizzo per il protocollo ICMP è la segnalazione di errori. Tipicamente se ho un’applicazione che sta usando la rete si arriva a un certo punto in cui si può manifestare una situazione di errore durante l’invio di un datagramma: il protocollo IP non prevede alcuna segnalazione di errore, né tantomeno il protocollo UDP e i messaggi di tipo DATAGRAM: Il protocollo ICMP invece invia alla macchina inviante un datagramma di segnalazione di errore, che verrà gestito dal sistema operativo. C’è anche la possibilità che vada perso anche il datagramma di segnalazione di errore, ma più di questo non si può fare. Gli errori segnalati dall’ICMP non sono visibili ai livelli superiori, quindi, neanche il protocollo TCP vi può accedere (anche perché non sono affidabili), ognuno ha il suo timer ecc.

Un esempio tipico di errore: il TTL exceeded (il Time To Live presente nel protocollo IPv4 è arrivato a 0 prima di arrivare a destinazione).

Un modo di utilizzare queste segnalazioni di errore è mediante un’applicazione chiamata “traceroot”, anch’essa serve a fare un po’ di debugging della rete: essa serve ad avere un’idea di qual è la strada percorsa dai datagrammi per arrivare a destinazione. Il modo in cui fa ciò è inviando un datagramma che ha indirizzo sorgente e destinazione corretti, ma in cui il TTL è ridotto. In particolare, la prima volta che viene inviato un datagramma il TTL è impostato a 1 -> questo datagramma non arriverà mai a destinazione perché il primo router lo scarterà: però tale router invierà indietro un datagramma ICMP di tipo Time To Live Exceeded. Questo datagramma conterrà quindi come indirizzi, l’indirizzo del mittente e l’indirizzo del router che ha inviato il datagramma TTL Exceeded -> ciò permette di scoprire l’indirizzo IP del primo router. Al secondo tentativo il TTL è aumentato di 1 (quindi è ora uguale a 2) e si reinvia il datagramma. Questo datagramma passerà attraverso il primo router e raggiungerà il valore 0 a TTL sul secondo router: analogamente al primo tentativo, il secondo router invierà indietro un datagramma ICMP che rivelerà il suo indirizzo IP. La procedura si ripete, con TTL uguale a 3, poi uguale a 4 ecc.  
Quando si arriva all’ultimo router, per sapere che dopo non ce ne sono altri, si usa un trucco: traceroot usa il livello di trasporto UDP per inviare il messaggio (lavorando al livello 4) e inventandosi a caso un numero di porta di destinazione -> questo farà sì che nella maggior parte dei casi tale numero di porta non corrisponderà a nessun processo in attesa di messaggi UDP e ciò provocherà una segnalazione di errore ICMP di tipo “PORT Unreachable”; a questo punto traceroot sa che il messaggio è arrivato a destinazione.  
Tutto ciò permette, un po’ alla volta, al programma traceroot di collezionare gli indirizzi IP di tutti i router che il datagramma attraversa per arrivare a destinazione. Quando la procedura è completata (a traceroot ritorna un errore di tipo PORT Unreachable), questo può stampare l’elenco di indirizzi IP dei router che instradano il messaggio a destinazione.

Se per sfortuna si becca una porta su cui la macchina destinataria è in ascolto, non si riceve nessun messaggio di errore e in tal caso traceroot dovrà inviare un altro messaggio con una porta diversa.

Un altro problema è che col passare del tempo può cambiare la strada seguita dai router per l’instradamento del messaggio: quindi è importante per traceroot di inviare questa sequenza di Datagrammi a TTL incrementato abbastanza velocemente, e sperare che nel breve intervallo non cambi il percorso seguito dal messaggio.