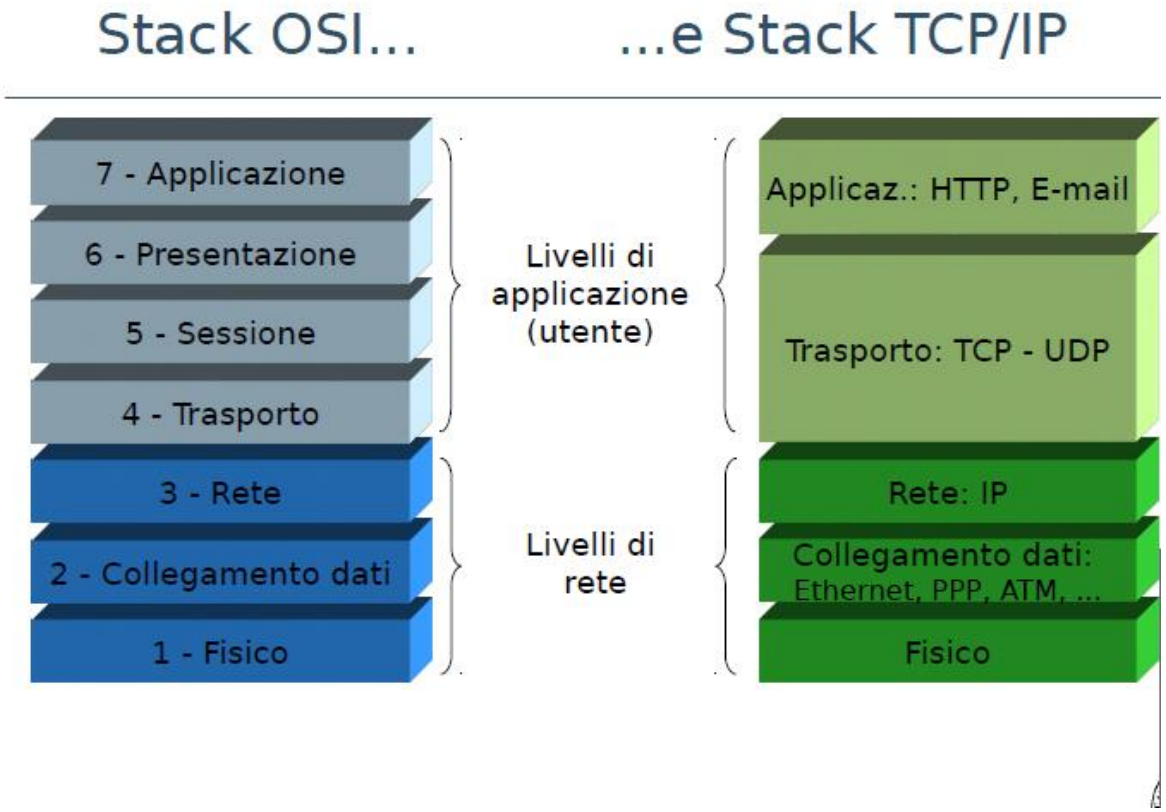
Ripasso dei concetti fondamentali di rete



Come avviene il trasporto dei segmenti dall'host sorgente all'host destinazione?

Il trasporto dei segmenti dall'host sorgente all'host destinazione avviene attraverso diversi passaggi:

1. **Incapsulamento dei segmenti:** Sul lato sorgente, i segmenti generati dal livello di trasporto (come ad esempio TCP o UDP) vengono incapsulati in datagrammi. Questo processo implica l'aggiunta di un header IP ai segmenti, creando così dei datagrammi IP.
2. **Trasmissione attraverso il livello di rete:** I datagrammi IP vengono quindi inviati nel livello di rete. Il livello di rete è presente in ogni end-host (come computer, server, dispositivi mobili, etc.) e in ogni apparato intermedio come i router.
3. **Esame dei datagrammi IP:** I router che fungono da apparati intermedi lungo il percorso verso la destinazione esaminano i campi dell'header presenti nei datagrammi IP. Questi campi includono informazioni come l'indirizzo IP di origine e di destinazione, il protocollo di trasporto utilizzato (TCP, UDP, etc.), e altre informazioni di routing necessarie per instradare i datagrammi al destinatario corretto.
4. **Consegna al livello di trasporto:** Una volta che i datagrammi IP raggiungono l'host destinazione, vengono estratti dal livello di rete e consegnati al livello di trasporto. Qui, l'header IP viene rimosso e i segmenti originali vengono estratti per essere consegnati all'applicazione o al protocollo di trasporto corretto (come TCP o UDP) sulla destinazione.

In sintesi, il trasporto dei segmenti avviene attraverso l'incapsulamento in datagrammi IP sul lato sorgente, la trasmissione attraverso il livello di rete che include l'esame dei datagrammi da parte dei router, e infine la consegna al livello di trasporto sul lato destinazione.

Quali sono alcune diffuse applicazioni di rete?

- Posta elettronica
- Web
- Messaggistica istantanea
- Autenticazione in un calcolatore remoto
- Condivisione di file P2P
- Giochi multiutente via rete
- Streaming di video-clip memorizzati
- Telefonia via Internet
- Videoconferenza in tempo reale
- Grid computing

Quali sono le architetture delle applicazioni di rete?

- Client-server
- Peer-to-peer (P2P)
- Architetture ibride (client-server e P2P)

Parlami dell'architettura client-server

L'architettura client-server è un modello di comunicazione e distribuzione dei servizi su una rete informatica. In questo modello, i ruoli principali sono svolti da due tipi di dispositivi: il client e il server.

Server:

- Il server è un computer o un'applicazione che fornisce servizi, risorse o funzionalità ad altri dispositivi, chiamati client.
- È progettato per essere sempre attivo e raggiungibile sulla rete, quindi di solito ha un indirizzo IP fisso.
- I server possono fornire una vasta gamma di servizi, come hosting web, archiviazione di file, gestione di database, posta elettronica, e molti altri.
- Le server farm sono gruppi di server fisici o virtuali collegati insieme per aumentare la capacità di erogazione dei servizi, migliorare la ridondanza e garantire l'affidabilità del sistema.

Client:

- Il client è un dispositivo che accede ai servizi o alle risorse offerti dal server.
- Può essere un computer, uno smartphone, un tablet o qualsiasi altro dispositivo connesso alla rete.
- I client possono comunicare con il server in qualsiasi momento per richiedere dati, inviare comandi o eseguire operazioni specifiche.

- Di solito, i client hanno indirizzi IP dinamici, assegnati temporaneamente dal server DHCP della rete a cui sono connessi.
- I client non comunicano direttamente con altri client sulla rete, ma solo con i server o con altri dispositivi intermedi, come router o switch.

Nel modello client-server, la comunicazione avviene tramite richieste e risposte: i client inviano richieste al server per ottenere risorse o servizi, e il server risponde fornendo i dati richiesti o eseguendo le operazioni richieste. Questo modello offre diversi vantaggi, tra cui la centralizzazione delle risorse, la gestione efficiente delle risorse di rete e la scalabilità del sistema. È ampiamente utilizzato in varie applicazioni, come il web hosting, i servizi cloud, la gestione dei database e molti altri.

Parlami dell'architettura Peer-to-peer (P2P)

L'architettura P2P (Peer-to-Peer) pura è un modello di rete decentralizzato in cui ogni dispositivo, chiamato peer, agisce sia da client che da server, consentendo la condivisione diretta di risorse e informazioni tra di essi. Ecco le caratteristiche principali:

1. **Assenza di un server centrale:** A differenza dell'architettura client-server, non esiste un server centrale sempre attivo che gestisce le comunicazioni tra i dispositivi. Invece, ogni peer ha lo stesso ruolo e la stessa autorità sugli altri.
2. **Comunicazione diretta tra i peer:** I peer comunicano direttamente tra loro senza passare attraverso un server intermedio. Questo permette una comunicazione più efficiente e riduce la dipendenza da un'infrastruttura centralizzata.
3. **Peer non sempre attivi e cambiamento degli indirizzi IP:** I peer non devono essere necessariamente sempre attivi. Possono entrare e uscire dalla rete in qualsiasi momento. Inoltre, possono cambiare gli indirizzi IP mentre si spostano da una rete all'altra, il che rende la gestione della connessione più dinamica ma anche più complessa.
4. **Facilmente scalabile:** Poiché ogni peer può contribuire alla rete fornendo risorse e servizi, l'architettura P2P è facilmente scalabile. Più peer si aggiungono alla rete, più risorse e capacità di elaborazione possono essere messe a disposizione per tutti gli utenti senza la necessità di investire in infrastrutture centralizzate costose.
5. **Difficile da gestire:** Sebbene l'architettura P2P offra vantaggi come la scalabilità e la decentralizzazione, può essere difficile da gestire a causa della sua natura distribuita. La mancanza di un'autorità centrale può rendere più complessa l'implementazione di funzionalità come la sicurezza, il controllo degli accessi e la gestione delle risorse. Inoltre, problemi come la ricerca efficiente delle risorse e la gestione dei conflitti possono essere più complessi da risolvere rispetto all'architettura client-server.

In sintesi, l'architettura P2P pura offre una modalità di comunicazione e condivisione delle risorse decentralizzata, dove i peer collaborano direttamente tra loro senza dipendere da un server centrale. Sebbene sia facilmente scalabile, può essere più complessa da gestire rispetto a un'architettura client-server tradizionale.

Parlami dell'architetture ibride (client-server e P2P)

Le architetture ibride che combinano elementi di client-server e P2P offrono una flessibilità e una capacità di adattamento che possono essere vantaggiose in diverse situazioni.

Skype: Skype è un'applicazione che utilizza un'architettura ibrida per fornire servizi di voice over IP (VoIP). Utilizza un'architettura P2P per le chiamate vocali effettive tra gli utenti. Questo significa che la trasmissione di dati vocali avviene direttamente tra i client senza passare attraverso un server centrale. Tuttavia, Skype utilizza anche un server centralizzato per la ricerca degli indirizzi della parte remota. Quando un utente desidera chiamare un altro utente, il suo client contatta un server centrale per ottenere l'indirizzo IP dell'utente di destinazione. Una volta ottenuto l'indirizzo IP, la comunicazione avviene direttamente tra i client.

Messaggistica istantanea: Le applicazioni di messaggistica istantanea spesso utilizzano un'architettura ibrida. La chat tra due utenti può avvenire direttamente tra i client, seguendo un modello P2P. In questo caso, i messaggi vengono scambiati direttamente tra i client senza passare attraverso un server centrale. Tuttavia, per individuare la presenza o la posizione degli utenti, queste applicazioni utilizzano un server centralizzato. Quando un utente si connette alla rete, il suo client registra il suo indirizzo IP sul server centrale. Quando un utente desidera contattare un amico, il suo client contatta il server centrale per ottenere l'indirizzo IP dell'amico.

Considerazioni generali:

1. **Scalabilità e decentralizzazione:** L'uso di elementi P2P consente di distribuire il carico di lavoro tra i peer, riducendo la dipendenza da un singolo server centrale e consentendo una maggiore scalabilità del sistema. Ciò significa che la rete può crescere in modo organico senza dover investire in server aggiuntivi costosi.
2. **Efficienza delle risorse:** L'architettura ibrida può sfruttare la potenza di calcolo e le risorse di archiviazione disponibili sui dispositivi dei peer, riducendo la necessità di risorse centralizzate. Questo può portare a una maggiore efficienza nell'utilizzo delle risorse complessive della rete.
3. **Gestione della larghezza di banda:** Utilizzando connessioni dirette tra i peer per lo scambio di dati, le architetture ibride possono ridurre il carico sulla rete centrale e migliorare le prestazioni complessive, specialmente in scenari di utilizzo intensivo della larghezza di banda come la condivisione di file.
4. **Affidabilità:** L'architettura ibrida può migliorare l'affidabilità del sistema distribuendo i punti di errore. Se un peer dovesse fallire, altri peer possono ancora comunicare tra loro senza dipendere da quel peer specifico.
5. **Gestione della sicurezza:** L'integrazione di elementi P2P e client-server può fornire un bilanciamento tra la privacy e la sicurezza dei dati. Ad esempio, mentre le comunicazioni P2P possono essere crittografate per garantire la privacy, la gestione dell'accesso e dell'autenticazione può essere centralizzata per garantire una maggiore sicurezza.
6. **Complessità della gestione:** Tuttavia, l'implementazione di architetture ibride può comportare una maggiore complessità nella gestione e nella manutenzione del sistema, poiché richiede la gestione sia dei componenti P2P che dei componenti client-server.

In sintesi, le architetture ibride offrono un approccio flessibile che può essere adattato alle esigenze specifiche di diversi scenari di utilizzo, combinando i vantaggi delle architetture client-server e P2P per offrire una soluzione efficiente, scalabile e affidabile.

Descrivi le principali caratteristiche di un protocollo a livello di applicazione, evidenziando le differenze tra protocolli di pubblico dominio e protocolli proprietari.

Un protocollo a livello di applicazione è un insieme di regole e convenzioni che definiscono come le applicazioni comunicano tra loro su una rete. Le principali caratteristiche di un protocollo a livello di applicazione includono:

1. **Tipi di messaggi scambiati:** I protocolli specificano i tipi di messaggi che possono essere scambiati tra le applicazioni. Questi messaggi possono includere richieste, risposte, conferme, aggiornamenti di stato e così via.
2. **Sintassi dei tipi di messaggio:** Definisce la struttura di un messaggio, inclusi i campi che contiene e il formato in cui sono rappresentati. Questi campi possono includere informazioni come l'indirizzo del mittente e del destinatario, i dati effettivi da trasmettere e i metadati associati al messaggio.
3. **Semantica dei campi:** Specifica il significato delle informazioni contenute nei campi del messaggio. Ad esempio, un campo "Data" può indicare la data e l'ora di invio del messaggio, mentre un campo "Tipo di richiesta" può indicare il tipo di azione richiesta dalla richiesta.
4. **Regole per determinare quando e come un processo invia e risponde ai messaggi:** Queste regole definiscono il comportamento delle applicazioni durante la comunicazione. Possono includere regole per la gestione degli errori, il timeout delle connessioni, la gestione dei conflitti e così via.

Ora, passando alle differenze tra protocolli di pubblico dominio e protocolli proprietari:

Protocolli di pubblico dominio:

- Sono definiti nelle RFC (Request for Comments), che sono documenti pubblicati dall'Internet Engineering Task Force (IETF) che descrivono gli standard, i protocolli e le tecnologie Internet.
- Consentono l'interoperabilità tra diverse piattaforme e sistemi, poiché sono standard aperti e documentati pubblicamente.
- Esempi includono HTTP (Hypertext Transfer Protocol), SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), che sono ampiamente utilizzati per comunicazioni web, email e trasferimento di file.

Protocolli proprietari:

- Sono sviluppati e mantenuti da organizzazioni o aziende specifiche e non sono pubblicamente documentati o standardizzati.
- Possono essere progettati per risolvere problemi specifici o per offrire funzionalità uniche non presenti nei protocolli di pubblico dominio.
- Esempi includono Skype, che offre servizi di messaggistica istantanea e chiamate vocali, e altri protocolli specifici di applicazioni o servizi di aziende private.

In sintesi, i protocolli di pubblico dominio sono standard aperti e documentati pubblicamente che favoriscono l'interoperabilità, mentre i protocolli proprietari sono sviluppati da singole entità e possono offrire funzionalità specifiche ma non sono accessibili a tutti e possono limitare l'interoperabilità con altre piattaforme e sistemi.

Quali sono i requisiti che un'applicazione può avere per il servizio di trasporto?

Il servizio di trasporto richiede requisiti diversi a seconda delle esigenze dell'applicazione. Alcuni dei principali requisiti includono:

1. **Affidabilità:** Alcune applicazioni, come quelle per il trasferimento di file o il telnet, richiedono un trasferimento dati affidabile al 100% senza perdite di pacchetti. Tuttavia, altre applicazioni, come la trasmissione audio, possono tollerare qualche perdita di pacchetti.
2. **Ritardi:** Applicazioni come la telefonia Internet o i giochi interattivi richiedono bassi ritardi o bassa variazione del ritardo per garantire un'esperienza utente realistica. D'altra parte, alcune applicazioni possono essere meno sensibili ai ritardi.
3. **Throughput:** Alcune applicazioni, come lo streaming video, richiedono una certa capacità del canale per essere efficaci, mentre altre, come il trasferimento di file, utilizzano semplicemente la capacità del canale disponibile.
4. **Sicurezza:** La sicurezza è un'altra considerazione importante. Le applicazioni possono richiedere cifratura, integrità dei dati e autenticazione per garantire la sicurezza delle comunicazioni.

Il ***throughput*** si riferisce alla quantità di dati che possono essere trasferiti attraverso una rete o un canale di comunicazione in un determinato periodo di tempo. In altre parole, rappresenta la velocità effettiva di trasmissione dei dati e viene misurato in bit al secondo (bps), kilobit al secondo (Kbps), megabit al secondo (Mbps) o gigabit al secondo (Gbps), a seconda della scala di misura. Un throughput più elevato indica una maggiore capacità del canale di trasporto e una maggiore efficienza nel trasferimento dei dati.

Requisiti del servizio di trasporto di alcune applicazioni comuni

Applicazione	Sensibilità alla perdita di dati	Throughput	Sensibilità al ritardo
Trasferimento file	Sì	Variabile	No
Posta elettronica	Sì	Variabile	No
Documenti Web	Sì	Variabile	No
Audio/video in tempo reale	No	Audio: da 5 Kbps a 1 Mbps Video: da 10 Kbps a 5 Mbps	Sì, centinaia di ms
Audio/video memorizzati	No	Come sopra	Sì, pochi secondi
Giochi interattivi	No	Fino a pochi Kbps	Sì, centinaia di ms
Messaggistica istantanea	Sì	Variabile	Intermedia

Quali sono le principali differenze nei servizi offerti dai protocolli di trasporto TCP e UDP?

Le principali differenze nei servizi offerti dai protocolli di trasporto TCP e UDP sono le seguenti:

TCP:

- TCP è orientato alla connessione, il che significa che richiede un setup tra i processi client e server prima di trasmettere i dati.
- Fornisce un trasporto affidabile fra i processi d'invio e di ricezione, garantendo che i dati siano consegnati senza errori e in ordine.
- Include un controllo di flusso, che impedisce al mittente di sovraccaricare il destinatario con troppi dati.
- Gestisce la congestione di rete limitando il tasso di invio quando la rete è sovraccaricata.
- Tuttavia, non offre controllo dei ritardi, garanzie su una capacità minima del canale o sicurezza.

UDP:

- UDP trasferisce dati inaffidabili tra i processi d'invio e di ricezione, senza garantire la consegna o l'ordine dei pacchetti.
- Non offre setup della connessione, affidabilità, controllo di flusso, controllo della congestione, controllo del ritardo, capacità del canale minima o sicurezza.

Perché esiste UDP?

UDP (User Datagram Protocol) esiste nonostante le sue limitazioni in quanto fornisce un servizio di trasporto più leggero e veloce rispetto a TCP (Transmission Control Protocol). UDP è utilizzato in situazioni in cui la velocità e la semplicità sono più importanti dell'affidabilità e del controllo. Ad esempio, applicazioni come videoconferenze, giochi online e trasmissioni in tempo reale possono beneficiare dell'uso di UDP, in quanto fornisce un trasferimento dati rapido e senza fronzoli, senza la necessità di stabilire e mantenere una connessione e senza il peso aggiuntivo di meccanismi di controllo e correzione degli errori. Sebbene possa comportare la perdita di alcuni pacchetti di dati, in certi contesti questo compromesso è accettabile per garantire prestazioni ottimali e tempi di risposta ridotti.

Applicazioni Internet: protocollo a livello applicazione e protocollo di trasporto

Applicazione	Protocollo a livello applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP [RFC 2821]	TCP
Accesso a terminali remoti	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Trasferimento file	FTP [RFC 959]	TCP
Multimedia in streaming	HTTP (es. YouTube) RTP [RFC 1889]	TCP o UDP
Telefonia Internet	SIP, RTP, proprietario (es. Skype)	Tipicamente UDP

Cosa è una socket e quali sono i suoi componenti principali?

Una socket è una rappresentazione astratta di un endpoint di comunicazione bidirezionale tra due processi su una rete.

E'una tupla che identifica il flusso tra due applicazioni su host anche molto distanti tra loro

I componenti principali di una socket includono l'indirizzo IP e il numero di porta di origine e di destinazione, che permettono ai dati di essere inviati e ricevuti tra le applicazioni.

Quali sono le caratteristiche principali del protocollo User Datagram Protocol (UDP)?

Il protocollo User Datagram Protocol (UDP) è un protocollo di trasporto connectionless non affidabile. Le sue caratteristiche principali includono:

1. Non gestisce connessioni, controllo di flusso, recupero degli errori, controllo della congestione o riordino dei pacchetti.
2. Svolge solo la funzione di indirizzamento delle applicazioni tramite l'utilizzo delle porte.
3. È compito dei livelli superiori, come le applicazioni, gestire aspetti come la verifica della perdita dei messaggi, la consegna in ordine e il controllo di flusso.
4. È utilizzato per il supporto di transazioni semplici tra applicativi e per le applicazioni multimediali che possono tollerare qualche perdita di dati o la mancanza di affidabilità nel trasferimento dei dati.

Quali sono i componenti del formato dei pacchetti UDP e quali sono le loro funzioni?

Il formato dei pacchetti UDP include i seguenti componenti:

1. Source Port e Destination Port [16 bit]: Questi campi identificano i processi sorgente e destinazione dei dati all'interno del datagramma UDP. Utilizzando i numeri di porta, UDP può instradare i dati agli appropriati processi sul mittente e sul destinatario.
2. Length [16 bit]: Questo campo indica la lunghezza totale del datagramma UDP, espressa in byte. Include sia la lunghezza dell'header UDP che dei dati contenuti nel pacchetto.
3. Checksum [16 bit]: Il checksum è un campo di controllo che aiuta a rilevare eventuali errori nel campo dati del datagramma UDP. Il mittente calcola un valore checksum basato sui dati trasmessi e lo include nel pacchetto. Il destinatario può quindi calcolare un checksum utilizzando i dati ricevuti e confrontarlo con il checksum ricevuto per verificare l'integrità del datagramma.

Quali sono i campi dell'header del protocollo TCP e quali sono le loro funzioni?

L'header del protocollo TCP include i seguenti campi:

1. ***Source Port e Destination Port***: Questi campi identificano rispettivamente la porta sorgente e la porta di destinazione.
2. ***Sequence Number***: Questo campo indica il numero di sequenza del primo byte del payload. (Viene utilizzato per numerare in modo univoco i byte di dati trasmessi e ricevuti, consentendo al destinatario di riordinarli correttamente)
3. ***Ack Number***: numero di sequenza del prossimo byte che si intende ricevere (ha validità se il segmento è un ACK)
4. ***Offset***: lunghezza dell'header TCP, in multipli di 32 bit
5. ***Reserved***: riservato per usi futuri
6. ***Window***: ampiezza della finestra di ricezione (comunicato dalla destinazione alla sorgente)
7. ***Checksum***: risultato di un calcolo che serve per sapere se il segmento corrente contiene errori nel campo dati
8. ***Urgent Pointer***: indica che il ricevente deve iniziare a leggere il campo dati a partire dal numero di byte specificato. Viene usato se si inviano comandi che danno inizio ad eventi asincroni "urgenti"
9. ***Flag [ogni flag è lunga 1 bit]***:

   – URG: vale uno se vi sono dati urgenti; in questo caso il campo urgent pointer ha senso
   – ACK: vale uno se il segmento è un ACK valido; in questo caso l'acknowledge number contiene un numero valido
   – PSH: vale uno quando il trasmettitore intende usare il comando di PUSH;
   – RST: reset; resetta la connessione senza un tear down esplicito
   – SYN: synchronize; usato durante il setup per comunicare i numeri di sequenza iniziale
   – FIN: usato per la chiusura esplicita di una connessione

10. ***Options & Padding***: [lunghezza variabile]: riempimento (fino a multipli di 32 bit) e campi opzionali come ad esempio durante il setup per comunicare il MSS
11. ***Data***: i dati provenienti dall'applicazione

Cos'è il "three-way handshake" nel protocollo TCP e come avviene l'instaurazione della connessione tra due end-system?

Il "three-way handshake" è la procedura utilizzata dal protocollo TCP per instaurare una connessione tra due end-system prima di iniziare il trasferimento dei dati. Essendo un protocollo connection-oriented, TCP richiede una connessione stabilita per garantire un trasferimento di dati affidabile. La procedura di "three-way handshake" si svolge in tre fasi:

1. **Invio del segmento SYN da parte della stazione richiedente (A):**
   - La stazione A invia un segmento di SYN (Synchronize) alla stazione destinataria (B).
   - Questo segmento include parametri come il numero di porta dell'applicazione cui si intende accedere e l'Initial Sequence Number (ISNA).
2. **Risposta con segmento SYN-ACK da parte della stazione ricevente (B):**
   - La stazione B risponde inviando un segmento che include sia un SYN (per sincronizzare) sia un ACK (Acknowledgment) per riconoscere il segmento SYN ricevuto da A.

    - I parametri specificati nel segmento di B includono l'Initial Sequence Number di B (ISNB) e l'ACK per il segmento SYN di A (ACK ISNA).
3. **Conferma con segmento ACK da parte della stazione richiedente (A):**
    - Infine, la stazione A conferma il segmento SYN di B inviando un segmento di ACK (Acknowledgment) alla stazione B.
    - Con questo segmento ACK, la stazione A riconosce la ricezione del segmento SYN di B, completando così il "three-way handshake".

Dopo questi tre passaggi, la connessione TCP è stabilita, e le due stazioni possono iniziare a trasferire i dati in modo affidabile.

Domanda:

Come avviene la terminazione di una connessione TCP?

Risposta:

Poiché la connessione è bidirezionale, la terminazione deve avvenire in entrambe le direzioni.

Procedura di Terminazione

1. **Invio del segmento FIN:**
    - La stazione che non ha più dati da trasmettere e decide di chiudere la connessione invia un segmento FIN (con il campo FIN impostato a 1 e il campo dati vuoto).
2. **Risposta con ACK:**
    - La stazione che riceve il segmento FIN invia un segmento di ACK e indica all'applicazione che la comunicazione è stata chiusa nella direzione entrante.

Half Close e Chiusura Completa

- **Half Close:**
    - Se la procedura di terminazione avviene solo in una direzione (half close), il trasferimento dei dati può continuare nell'altra direzione. Gli ACK non sono considerati come traffico originato, ma come risposta al traffico.
- **Chiusura Completa:**
    - Per chiudere completamente la connessione, la procedura di half close deve avvenire anche nell'altra direzione.

Domanda:

Come viene stimato il Round Trip Time (RTT) nel protocollo TCP e qual è l'importanza del parametro α nella stima?

Risposta:

Stima del Round Trip Time

Poiché il RTT (Round Trip Time) può variare significativamente in base alle condizioni della rete, il valore di RTT utilizzato per il calcolo del RTO (Retransmission Timeout) risulta essere una stima del valore medio del RTT sperimentato dai diversi segmenti, noto come Smoothed RTT (SRTT).

Parametro α

Il parametro α è regolabile e, a seconda dei valori assunti, rende il peso della misura di RTT istantaneo più o meno incisivo:

- **Se α → 1**:
  - Il SRTT stimato risulta abbastanza stabile e non viene influenzato da singoli segmenti che sperimentano RTT molto diversi.
- **Se α → 0**:
  - Il SRTT stimato dipende fortemente dalla misura puntuale dei singoli RTT istantanei.

Tipicamente, α è impostato a 0.9.

Formula di Stima del SRTT

La stima attuale del SRTT viene calcolata utilizzando la seguente formula:

SRTTattuale=(α×SRTTprecedente)+((1−α)×RTTistantaneo)\text{SRTT}_{\text{attuale}} = (\alpha \times \text{SRTT}_{\text{precedente}}) + ((1 - \alpha) \times \text{RTT}_{\text{istantaneo}})SRTTattuale=(α×SRTTprecedente)+((1−α)×RTTistantaneo)

- **RTT istantaneo**: Misura di RTT sull'ultimo segmento.
- **SRTT precedente**: Stima precedente del valore medio di RTT.
- **SRTT attuale**: Stima attuale del valore medio di RTT.
- **α**: Coefficiente di peso ($0 < \alpha < 1$).

Domanda:

Come viene stimato il Retransmission Time Out (RTO) nel protocollo TCP e quale processo viene utilizzato in caso di ritrasmissione?

Risposta:

Stima del Retransmission Time Out (RTO)

La sorgente attende fino a 2 volte il RTT medio (SRTT) prima di considerare un segmento perso e ritrasmetterlo.

Procedura di Ritrasmissione

1. **Ritrasmissione in caso di timeout:**
   - Se il RTO scade senza ricevere un ACK, probabilmente c'è congestione nella rete. Pertanto, è opportuno aumentare il RTO per quel segmento.

2. **Exponential Backoff:**
    - Il RTO per la ritrasmissione viene ricalcolato secondo un processo di exponential backoff: RTOritrasmissione=2×RTO\text{RTO}_{\text{ritrasmissione}} = 2 \times \text{RTO}RTOritrasmissione=2×RTO

Formula di Calcolo del RTO

RTO=β×SRTT\text{RTO} = \beta \times \text{SRTT}RTO=β×SRTT

- **β**: Fattore di varianza del ritardo, tipicamente impostato a 2.
- **SRTT**: Stima media del Round Trip Time.

Esempio di Procedura di Ritrasmissione

- **Stazione A invia il segmento X a stazione B.**
    - Attende l'ACK per il segmento X entro il tempo RTO.
- **Se non riceve l'ACK:**
    - Stazione A considera il segmento X perso e lo ritrasmette.
    - Ricalcola il RTO per la ritrasmissione utilizzando il processo di exponential backoff.

Scenario Dettagliato:

1. **Invio del segmento X:**
    - Stazione A → Segmento X → Stazione B
2. **Attesa dell'ACK:**
    - Stazione A attende l'ACK entro il tempo RTO.
3. **Ritrasmissione dopo timeout:**
    - Se l'ACK non arriva entro il RTO, Stazione A ritrasmette il segmento X e ricalcola il nuovo RTO: RTOritrasmissione=2×RTO\text{RTO}_{\text{ritrasmissione}} = 2 \times \text{RTO}RTOritrasmissione=2×RTO

Questo processo continua fino a quando non viene ricevuto un ACK per il segmento ritrasmesso, con il RTO che raddoppia ad ogni ritrasmissione senza successo per gestire efficacemente la congestione della rete

Domanda:

Quali sono le funzioni chiave del livello di rete e quale ruolo svolge l'indirizzamento in esse?

Risposta:

Funzioni chiave del livello di rete

Routing

- **Descrizione:** Determina il percorso che i pacchetti devono seguire dalla sorgente alla destinazione.
- **Ruolo degli algoritmi di routing:** Essenziali per determinare il percorso ottimale.
- **Analogia:** Simile al processo di pianificazione di un viaggio, dalla partenza all’arrivo.

Forwarding

- **Descrizione:** Trasferisce i pacchetti da una porta di input alla porta di output appropriata in un router.
- **Analogia:** Analogamente, durante un viaggio, è simile al passaggio attraverso un incrocio, dove si deve scegliere quale strada prendere.

Queste funzioni richiedono un componente fondamentale:

- **Indirizzamento (Addressing):** Identifica univocamente sorgenti e destinazioni all'interno di una rete, essenziale per l'effettivo routing e forwarding dei pacchetti.

Domanda:

Quali sono le funzionalità chiave nell'architettura di un router e quali piani sono coinvolti nel suo funzionamento?

Risposta:

Nell'architettura di un router, le funzionalità chiave includono:

- Eseguire gli algoritmi e i protocolli di routing come RIP, OSPF e BGP.
- Trasferire (commutare) datagrammi dalla porta di input alla porta di output.

Piani coinvolti nel Funzionamento del Router:

1. **Data Plane:**
    - Il Data Plane è responsabile del trasferimento effettivo dei dati. Gestisce il flusso di pacchetti dalla porta di ingresso alla porta di uscita, seguendo le regole stabilite dai protocolli di routing.
2. **Control Plane:**
    - Il Control Plane è responsabile dell'elaborazione delle decisioni di routing e della gestione degli algoritmi e dei protocolli di routing come RIP, OSPF e BGP. Si occupa di instradare i pacchetti nel modo più efficiente possibile attraverso la rete.

Domanda:

Quali sono le funzioni delle porte di input in un sistema di switching decentralizzato?

Risposta:

Le funzioni delle porte di input in un sistema di switching decentralizzato includono:

- **Determinazione della porta di output:** Utilizzando le tabelle di forwarding in memoria locale, le porte di input determinano la porta di output appropriata per un pacchetto in base alla sua destinazione.

- **Scopo:** Il principale obiettivo delle porte di input è quello di completare l'elaborazione dei pacchetti alla velocità di linea, garantendo che i pacchetti vengano inoltrati il più rapidamente possibile.
- **Gestione delle code:** Le porte di input gestiscono le code nel caso in cui i datagrammi arrivino alla porta di input con una velocità superiore alla velocità di uscita. In tal caso, i pacchetti vengono messi in coda fino a quando non possono essere inoltrati.

Queste funzioni consentono alle porte di input di gestire l'efficace instradamento dei pacchetti all'interno del sistema di switching decentralizzato, garantendo un'elaborazione efficiente e senza perdita di dati. Le porte di input operano a livello fisico e a livello di collegamento dati, ad esempio nel caso di Ethernet.

Domanda:

Quali sono le principali funzioni delle porte di output in un sistema di commutazione di pacchetti?

Risposta:

Le principali funzioni delle porte di output in un sistema di commutazione di pacchetti includono:

- **Gestione delle code (Queuing):** Le porte di output gestiscono le code nel caso in cui la velocità di arrivo dei datagrammi sia superiore alla velocità di uscita. In questo modo, i pacchetti vengono messi in coda nel buffer fino a quando possono essere trasmessi.
- **Scheduling:** Le porte di output utilizzano il scheduling per determinare l'ordine di trasmissione dei datagrammi in coda nel buffer. Questo processo di scheduling aiuta a garantire un uso efficiente delle risorse disponibili e a soddisfare le esigenze di QoS (Quality of Service) della rete. Protocolli di scheduling e metodi di encapsulation possono essere utilizzati per regolare l'ordine di trasmissione dei datagrammi.

Il formato dell'header del datagramma IP

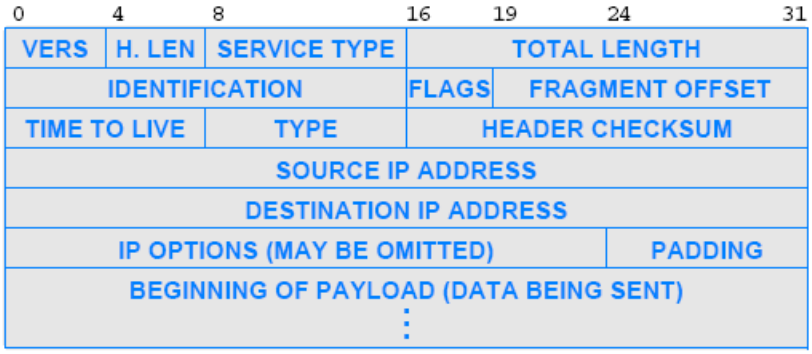


Figure 22.2 Fields in the IP version 4 datagram header.

Domanda:

Qual è la gerarchia degli indirizzi IP e quali sono le loro due parti concettuali?

Risposta:

Gerarchia degli Indirizzi IP

Gli indirizzi IP sono divisi concettualmente in due parti:

- **Prefisso (NetID):**
  - Identifica la rete fisica a cui l'host è connesso. Ogni rete in Internet viene assegnato un prefisso di rete univoco.
- **Suffisso (HostID):**
  - Identifica un host specifico all'interno di una rete. A ciascun host su una data rete viene assegnato un suffisso univoco, mentre il prefisso è uguale per tutti gli host nella stessa rete.

Lo schema degli indirizzi IP garantisce due proprietà fondamentali:

- A ciascun host viene assegnato un indirizzo unico.
- L'assegnazione dei numeri di rete (prefissi) viene coordinata globalmente. I suffissi vengono assegnati localmente, senza bisogno di coordinazione globale.

Domanda:

Quali sono le soluzioni proposte per superare le limitazioni dello schema classful degli indirizzi IP?

Risposta:

Subnet e Indirizzamento Classless

Subnetting e Indirizzamento Classless sono la soluzione alle limitazioni dello schema classful degli indirizzi IP.

Lo schema originale, con prefissi di lunghezza fissa, era limitante e portava a un sottoutilizzo degli indirizzi.

- Subnetting permette di dividere una rete in reti più piccole
-
- l'Indirizzamento Classless rende arbitraria la lunghezza dei prefissi/suffissi, offrendo maggiore flessibilità nell'allocazione degli indirizzi.

Domanda:

Cosa significa CIDR e quale è il suo ruolo nel contesto degli indirizzi IP?

Risposta:

Notazione CIDR

- **CIDR:** Acronimo di Classless Inter-Domain Routing.
- L'utilizzo di maschere per definire la dimensione del prefisso migliora l'efficienza, grazie alla velocità delle operazioni di AND bit a bit in hardware.
- Per semplificare la gestione, si adotta una notazione più diretta, dove si specifica direttamente la dimensione del prefisso.
- La notazione CIDR ha la forma "ddd.ddd.ddd.ddd/m", dove "ddd" rappresenta il valore decimale nella notazione decimale puntata e "m" indica il numero di bit del prefisso.
- Esempi di definizione di reti in CIDR includono "NetA: 128.10.0.0/16" e "NetB: 64.10.2.0/24".

Domanda:

Cos'è il routing?

Risposta:

Il routing è il processo attraverso il quale viene individuato il percorso ottimale da una sorgente a ogni destinazione all'interno di una rete.

Ad esempio, se un utente desidera connettersi a un server situato nell'Antartide dal proprio desktop, il routing determina il percorso da seguire.

Esso considera

- la lunghezza,
- la velocità
- la disponibilità del percorso

In caso di guasto di un link lungo il percorso, il routing gestisce la situazione, trovando alternative e mantenendo la connessione attiva.

Domanda:

Cosa gestisce un protocollo di routing e qual è il principale problema associato?

Risposta:

Un protocollo di routing gestisce una tabella di routing nei router. Questa tabella indica, per ogni destinazione, quale interfaccia di uscita utilizzare per inoltrare i pacchetti. Tuttavia, i nodi fanno scelte locali basate sulla topologia globale della rete, il che rappresenta il problema principale.

Domanda:

Qual è il problema chiave nel routing e come può essere affrontato?

Risposta:

Il problema chiave nel routing è determinare come effettuare decisioni locali corrette.

Questo implica che ogni router deve avere conoscenza di alcune informazioni sullo stato globale della rete. Tuttavia, lo stato globale della rete è intrinsecamente grande, dinamico e le informazioni ad esso relative sono di difficile reperibilità.

Per affrontare questo problema, un protocollo di routing deve essere in grado di riassumere le informazioni più importanti sulla rete, consentendo ai router di prendere decisioni locali accurate.

Domanda:

Quali sono le tipologie di algoritmi di routing e come funzionano?

Risposta:

Le tipologie di algoritmi di routing includono:

1. **Distance vector (DV):**
    - In questo tipo di algoritmo, periodicamente ogni nodo invia ai propri vicini il vettore delle distanze (distance vector, DV) verso tutte le sottoreti del mondo.
    - Ogni nodo basa le sue decisioni di routing sui vettori delle distanze ricevuti dai suoi vicini, scegliendo il percorso con la distanza minore verso ogni destinazione.
2. **Link state:**
    - In questo tipo di algoritmo, periodicamente ogni nodo invia a tutte le sottoreti del mondo lo stato dei collegamenti (link state) verso i propri vicini.
    - Ogni nodo utilizza queste informazioni per costruire una mappa completa della topologia di rete e calcolare il percorso più breve verso ogni destinazione utilizzando l'algoritmo di Dijkstra.(L'algoritmo di Dijkstra calcola il percorso più breve attraverso una rete utilizzando un approccio basato sulla ricerca dei nodi più vicini.)

Visibilita della rete del livello 2 e 3

livello 2: Visibilità limitata al singolo canale fisico

livello 3: Visibilità estesa a tutta la rete

Domanda:

Quali sono le principali funzioni del livello 2 all'interno di una rete?

Risposta:

Le principali funzioni svolte dal livello 2 includono:

1. **Framing:**
   - Questa funzione delimita i messaggi (frame) come gruppi di bit sul canale fisico, consentendo ai dispositivi di distinguere dove inizia e finisce ciascun frame.
2. **Rilevazione/gestione degli errori:**
   - Il livello 2 controlla se il frame contiene errori e, se necessario, gestisce il recupero degli stessi per garantire l'integrità dei dati trasmessi.
3. **Metodo di accesso al canale:**
   - Questo livello gestisce il metodo attraverso il quale i dispositivi accedono al mezzo trasmissivo condiviso, come ad esempio l'Ethernet che utilizza il protocollo CSMA/CD (Carrier Sense Multiple Access with Collision Detection).
4. **Controllo di flusso:**
   - Gestisce la velocità di trasmissione dei dati tra dispositivi, regolando il flusso di dati

Domanda:

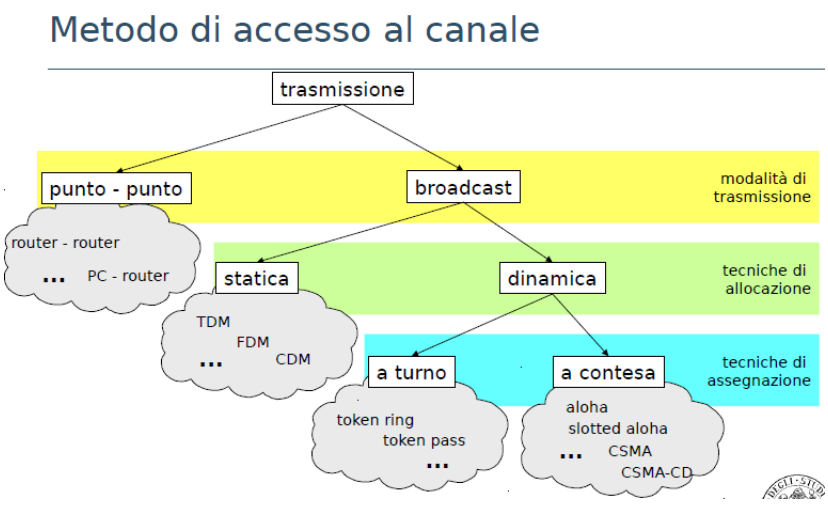
Qual è il problema fondamentale affrontato dal livello 2 durante la fase di framing?

Risposta:

Il problema fondamentale affrontato dal livello 2 durante la fase di framing è la delimitazione dei frame.

Poiché i pacchetti provenienti dal livello superiore (livello 3) e le relative trame a livello 2 possono avere lunghezze variabili e i sistemi non sono sincronizzati tra loro, diventa difficile distinguere dove inizia e finisce ciascun frame.

Pertanto si rende necessaria la funzionalità di framing che rende distinguibile un frame dall'altro attraverso l'utilizzo di opportuni codici all'inizio e alla fine del frame stesso.



Domanda: Quali sono i principali algoritmi di accesso multiplo al mezzo condiviso con contesa?

Risposta: I principali algoritmi sono:

- ALOHA
    - Pure ALOHA
    - Slotted ALOHA
- Carrier Sense Multiple Access Protocols
    - CSMA
    - CSMA/CD (con rilevazione della collisione – Collision Detection)
    - CSMA/CA (con tentativo di riduzione delle collisioni – Collision Avoidance)

Domanda: Che cos'è il Carrier Sense Multiple Access (CSMA) e come funziona?

Carrier Sense Multiple Access (CSMA) è un protocollo utilizzato nelle reti LAN, in cui le stazioni monitorano costantemente lo stato del canale di trasmissione per ridurre i ritardi. Prima di trasmettere, le stazioni "ascoltano" il canale per verificare se è libero. Se il canale è libero, la stazione trasmette immediatamente; se è occupato, possono essere adottate diverse strategie:

- Non-persistent: ritarda la trasmissione ad un nuovo istante casuale.
- Persistent: trasmette appena il canale si libera.

In caso di collisione, simile ad ALOHA, la stazione attende un tempo casuale prima di ritentare la trasmissione.

Scrittura di applicazioni di rete mediante interfaccia socket

● L'host (in inglese "colui che ospita") è un dispositivo come PC, supercomputer, tablet, smartphone, smartwatch, semaforo intelligente, ecc.

- Sempre identificato da un indirizzo IP a cui può essere associato, opzionalmente, un nome Internet.

● Processo: programma in esecuzione sull'host che trasmette o riceve pacchetti verso o da altri processi su altri host attraverso la rete.

- Identificato da un numero di porta nell'intervallo 0..65535.

● Applicazione: collaborazione tra un insieme di processi distribuiti sulla rete per svolgere attività utili per l'utente (es. Web, chat, e-mail, telemedicina).

Esempi di applicazioni che utilizzano la rete:

● Il Web è l'applicazione.

● Chrome, Firefox, Edge e Safari sono i processi in esecuzione sul mio PC, tablet o smartphone che fungono da host.

● Apache e NGINX sono i processi in esecuzione sulla macchina remota, anch'essa un host.

● Telegram è l'applicazione.

● L'app Telegram è il processo che gira sul mio smartphone, che funge da host.

● Il server Telegram è il processo in esecuzione sulla macchina remota, anch'essa un host.

La finestra dei comandi del PC anche chiamata: terminale, shell, console.

Per scoprire l'indirizzo IP degli host:

● Prendiamo l'esempio del Web e consideriamo il sito www.wikipedia.org.

● L'indirizzo IP del mio host si può trovare digitando un comando da terminale (finestra CMD in Windows):

- In Linux e Mac OS: `ifconfig -a`
- In Windows: `ipconfig /all`

● L'host del server di Wikipedia ha il nome Internet www.wikipedia.org e l'indirizzo IP si può trovare digitando il seguente comando da terminale (finestra CMD in Windows):

- `nslookup www.wikipedia.org`

Modalità di trasmissione in Internet

● Trasmettere un bit o un byte alla volta non è efficiente.

● Una sequenza di byte è chiamata con vari sinonimi:

- Pacchetto
- Protocol Data Unit (PDU)
- Datagramma

Applicazioni orientate al datagram

- Ogni pacchetto scambiato tra gli host è indipendente dai precedenti e successivi
- Le perdite di pacchetti non vengono tenute in considerazione

Se tra i pacchetti trasmessi c'è una relazione→ sono parte di un messaggio più grande (ad es. un'immagine)

Il sistema operativo deve introdurre nei pacchetti un numero di sequenza e rilevare eventuali pacchetti persi per poterli ritrasmettere

Vantaggi:
- L'utente scrive/legge su un archivio remoto con la stessa naturalezza di quando scrive/legge su un archivio locale come se la rete in mezzo non ci fosse

Svantaggi:
- Gli host trasmettitore e ricevitore devono "lavorare" di più dentro il sistema operativo
- Può nascere un maggior ritardo di trasmissione in caso di ritrasmissione di pacchetti persi

Schemi di applicazioni che usano la rete

- Le applicazioni di rete sono insiemi di processi su host diversi che si scambiano messaggi attraverso la rete
- Esistono degli schemi base che regolano lo scambio di messaggi:

  - Client/server

  - Publisher/Subscriber (Pub/Sub)