F.A.Q. RETI DI CALCOLATORI T

Corso di Laurea in Ingegneria Informatica

A. A. 2015-2016

CELSO ALESSANDRO DAVIDE
Università degli Studi di Bologna

Modelli

1. **Descrivere il modello C/S sincrono non bloccante e individuare i ruoli e le entità che entrano in gioco.**
   Il modello C/S sincrono, non bloccante ha due entità: il client sequenziale, colui che chiede il servizio ed un server sequenziale o parallelo, che risponde. Il modello a default è sincrono, bloccante asimmetrico. Sincrono perché si prevede che ci sia una risposta da parte del server al client che ha fatto richiesta e non bloccante perché il cliente non resta in attesa di una risposta ma, anzi, può inviare più richieste a diversi server.

2. **Definisci precisamente le caratteristiche di un rapporto C/S in un sistema distribuito nelle sue caratteristiche di sincronicità, blocco e dinamicità.**
   Il rapporto tra C/S può presentare diverse caratteristiche di sincronicità, si dice sincrono quando il server da sempre una risposta al cliente, nel caso questa non sia prevista allora definiamo il rapporto come asincrono. È bloccante quando il processo cliente si blocca in attesa di una risposta da parte del server, si blocca fino ad un determinato timeout stabilito in fase di scrittura del codice, viceversa sarà non bloccante quando il cliente non si aspetta una risposta da parte del server ma delega un suo agente, cioè un proxy, da tramite per questa risposta, nel modello C/S non bloccante il cliente può tranquillamente inviare richieste a diversi server. Il binding, nei sistemi distribuiti, tra C/S è sempre dinamico, in quanto non si può sapere a priori chi richiede il servizio.

3. **Definisci il modello C/S a default**
   Il modello C/S a default è sincrono, bloccante, asimmetrico, dinamico e con due entità: client (sequenziale) e server (sequenziale o parallelo). Sincrono perché si prevede sempre una risposta da parte del server al cliente, bloccante perché il cliente aspetta una risposta da parte del server, asimmetrico perché il cliente conosce il server ma questo a priori non conosce il client finché questo non fa una richiesta e infine dinamico perché il binding tra le entità viene fatto soltanto ad avvenuta richiesta da parte del client.
   Poiché bloccante il client fa una richiesta e aspetta, nel caso questa non arrivi all'interno di un intervallo di tempo prestabilito (timeout) viene scatenata una eccezione oppure si potrebbe provvedere a fare un'altra richiesta ad un altro server.

4. **Definisci i possibili modi di interazione tra cliente e servitore.**
   Le interazioni tra client e server possono essere di due tipi:
   - Con connessione: si stabilisce uno stream, cioè un canale di comunicazione tra cliente e servitore ciò garantisce che i messaggi vengano scambiati rispettando un certo ordine, senza perdita di informazione, ordinata e senza copie. Il protocollo usato è il TCP, affidabile ma con un alto costo.
   - Senza connessione: non garantisce l'ordine dei pacchetti e neanche la ritrasmissione di quelli persi, perciò garantisce una affidabilità minore rispetto all'interazione con connessione, tuttavia è molto rapido, non è presente latenza (spesso dovuta a riordino e ritrasmissione). Il protocollo usato è UDP.
   -

5. **Definisci i possibili modi di interazioni del cliente con il server**
   Sono possibili due diverse modalità di comunicazione da parte del client con il server:
   - PULL: modello in cui il cliente esegue le richieste impostando un timeout molto piccolo, in modo che possa effettuare nuovamente la richiesta e ricevere la risposta da parte del server immediatamente, poiché questo modello prevede che questo abbia tenuto in memoria la richiesta inizialmente fatta. Progetto client complesso ma server semplice.
   - PUSH: modello in cui il cliente sia non bloccante, quando il server ha gestito ed elaborato la richiesta, etichettata con un id del client, sarà lo stesso server a inviare la risposta.
     Progetto client semplice ma server complesso.

6. **Definisci e confronta i modelli C/S e PUB/SUB (eventi a scambio di messaggi).**
   - C/S: due entità (client e server), sincrono, bloccante, il cliente conosce il server solo alla richiesta, le entità in gioco sono in accoppiamento forte e devono essere presenti entrambi affinché ci sia interazione.
   - PUB/SUB tre entità (SUB(scribe), GESTORE, PUB(blish)), il sub si registra ad un gestore, il pub genera gli eventi e li consegna al gestore, il gestore riceve gli eventi e ne fa un push ai sui già registrati, a differenza del modello C/S c'è un accoppiamento molto debole, non c'è bisogno della copresenza delle parti e permette comunicazioni multicast, ed è molti a molti.

7. **Definisci il modello C/S asincrono.**
   Il modello C/S oltre che a default (sincrono bloccante) può anche essere asincrono, cioè quando il cliente non deve aspettarsi una risposta da parte del server, ovvero quando il sistema è PUSH, PULL o PUB/SUB (per la trattazione estesa guarda su).

8. **Descrivi cos'è un sistema di nomi.**
   Il cliente deve riferire al servitore tramite il suo "nome" e questo può essere:
   - Trasparente: nomi indipendenti dalla locazione fisica del servizio, quando un client utilizza tale nome si aspetta di poter raggiungere tutti i servitori che realizzano quel servizio. BIND DINAMICO (cioè fatto al momento del bisogno)
   - Non trasparente: nomi che dipendono dalla locazione fisica del servizio. BIND STATICO (viene fatto a priori dell'esecuzione)

   In sistemi concentrati si preferisce un sistema statico, mentre nel distribuito quello dinamico e quindi viene fatto uso di un sistema affinché il cliente conosca il servitore.

9. **Descrivi cos'è e quali siano le funzioni del servizio DNS**
   Al fine di conoscenze reciproche tra entità è necessario reperire il nome dei servitori nel cliente. Il DNS (Domain Name System) è un insieme di gestori di tabella di nomi logici e indirizzi IP, con l'obiettivo di attuare corrispondenze tra nomi logici (HOST) e fisici (IP)
   Il DNS introduce nomi logici in una gerarchia ad albero con a capo una radice senza alcun nome da cui vengono individuati i domini di primo, secondo, terzo, quarto… livello.
   I domini sono logici e non collegati alla rete fisica, ogni dominio ha un name resolver e un domain name server, il client fa richiesta al name resolver che consulta la cache per cercare una eventuale risposta già presente, in caso questa non sia già presente provvede a far richiesta al name server che provvederà a passare la risposta

10. **Quali tipo di query DNS possono essere fatte? Che cos'è il DNS resolver? Che entità utilizza il DNS per risolvere le query e in che modo?**
    Ci sono due tipi di query che possono essere fatte: ricorsiva (prevede che il resolver fornisca la risposta, chiedendo ovviamente in modo ricorsivo ad altri server) oppure iterativa (prevede che il resolver fornisca la risposta o il miglior suggerimento come riferimento al miglior DNS server).
    Un DNS resolver è una entità che fornisce la risposta ad una richiesta DNS, questa risposta può arrivare o perché già presente in cache oppure perché trovata attraverso una richiesta al name server.

11. **Directory e organizzazione X.500**
    X.500 è uno standard definito da OSI, nato per superiore il vincolo del DNS, tale standard è una directory ovvero un sistema di nomi in grado di catalogare qualunque informazione nel sistema di nomi con caratteristiche di qualità. Presenta una struttura ad albero il ché facilità le query complesse e la ricerca su tutta la struttura.
    Per accedere alla directory serve un DUA (directory user agent) utlizzato dagli utenti per effettuare una query, la directory necessita di una serie di agenti, cioè delle macchine adibite alla memorizzazione di party della directory che vengono consultate per ottenere il risultato dell'interrogazione.

12. **Descrivere le possibili semantiche di interazione C/S: may-be, at-least-once, ecc**
    La semantica may-be non garantisce affidabilità siccome il messaggio può arrivare o meno, un esempio è il Best-effort in IP, UDP. Nella semantica at-least-once il messaggio può arrivare più volte a causa delle possibili ritrasmissioni non controllate, può essere utile per operazioni idempotenti, cioè operazioni ripetute aventi sempre lo stesso risultato. Invece nella semantica at-most-once, il cliente e il servitore lavorano cercando di garantire affidabilità e in questo modo il messaggio arriva al più una volta, quindi è una semantica adatta a qualunque tipo di operazione, questo avviene grazie al mantenimento dello stato da parte del server.

Socket in Java

---

1. **Descrivere le classi e primitive usate per una comunicazione UDP in Java, distinguendole tra cliente e servitore.**
   Le classi usate da cliente e servitore sono le stesse:
   - DatagramSocket, crea un end-point di comunicazione tra due thread senza stabilire una connessione, creata la socket viene fatto lo scambio di messaggi. Su DatagramSocket vengono invocati i metodi: send(DatagramPacket p) e receive(DatagramPacket p) che sono, rispettivamente, usati per l'invio e la ricezione di un singolo datagramma.

2. **Descrivere le classi e primitive usate per una comunicazione TCP in Java, distinguendole tra cliente e servitore.**
   Distinguiamo la classe ServerSocket per il servitore e Socket per il cliente, presenti all'interno del package java.net.
   - ServerSocket, crea una socket di ascolto sulla porta del server public ServerSocket(int port) e permette di accodare e accettare (con la primitiva accept()) le richieste di connessione provenienti da diversi clienti. Per l'invio e la ricezione di pacchetti vengono usate DataOutputStream() e DataInputStream().
     Nel costruttore è possibile anche aggiungere un int count come lunghezza di coda.
   - Socket, crea una socket stream di comunicazione mediante TCP con il servitore, il costruttore prende in ingresso indirizzo IP e porta. Dopo aver qualificato le risorse di una connessione socket stream, l'invio e la ricezione dello stream avviene tramite: *public InputStream getInputStream()* e *public OutputStream getOutputStream().*Attraverso gli stream di java si possono inviare/ricevere solo bytes, questi arrivano ordinati e non duplicati e al più una volta secondo la semantica AT-MOST-ONCE, quindi con una certa affidabilità, ma senza nessun controllo in caso di malfunzionamento.

3. **Multicast socket in Java**
   ??????????????

4. **Descrivere le primitive usate in fase di chiusura delle socket in Java.**
   In Java, nelle socket stream, trattandosi di socket connesse è necessario chiudere eventuali connessioni rimaste aperte al fine di risparmiare risorse, ciò viene vato tramite il metodo synchronized close() throws SocketException, il quale chiude l'oggetto socket precedentemente creato e disconnette il lciente dal servitore.

Socket in C

1. **Elenca le funzioni usate per creare una datagram socket in C e dire cosa esprime il loro valore di ritorno.**
   Le datagram socket in C prevedono il supporto di alcune primitive per la loro creazione e sono:
   - int socket() che consente la creazione delle socket e restituisce un intero maggiore di 0 in caso di successo, altrimenti in caso di errore un valore minore di 0.
   - int bind() crea il legame tra indirizzi e numero di porta, restituisce un valore intero maggiore 0 in caso di successo, minore di zero in caso di errore.
   - int read() e int write() sono usate, rispettivamente, per la ricezione e l'invio di messaggio, restituiscono un intero che altro non è che il numero di byte trasmessi.
   - void close() è la funzione che chiude la comunicazione tra cliente e servitore.

2. **Socket TCP in C, nominare le primitive usate per l'apertura di una connessione socket TCP.**
   Le primitive usate per creare una socket stream di tipo TCP in C sono:
   - socket() crea l'entità l'entità logica di una connessione, usata sia per il cliente che per il servitore e restituisce un socket descritor in caso di successo o -1 se fallisce.
   - bind() collega la socket locale a un indirizzo e al sistema di nomi, opzionale per il cliente, obbligatoria per il servitore che ne fa uso per essere raggiunto dal cliente che gli fa una richiesta.
   - connect() primitiva di comunicazione sincrona usata dal cliente, termina quando la richiesta è stata accodata o in caso di errore, restituisce il file descritor del risultato in caso di successo, un valore negativo in caso di errore.
   - listen() primitiva senza attesa usata dal server per accordare le possibili richieste, restituisce il file descriptor in caso di successo, un valore negativo in caso di errore.
   - accept() primitiva con attesa usata dal server usata per stabilire la reale connessione accettando e servendo le richieste in coda, restituisce una nuova socket connessa al client in caso di successo, un valore negativo in caso di errore.
   - read()/write() o send()/recv() lettura e scrittura di byte sulla socket connessa, restituisce il numero di byte inviato o ricevuto.
   - close() e shutdown() funzioni usate per la chiusura della connessione.

3. **Primitiva select() in C, indicarne l'uso, la sintassi, la firma, il valore di ritorno e gli argomenti.**
   La realizzazione delle socket in C fa sì che si utilizzino delle primitive che a volte hanno un comportamento sincrono bloccante, ciò, ovviamente, può penalizzare il funzionamento di un sistema generale perché il processo potrebbe restare bloccato in attesa di richieste che non arrivano, tale gestione viene risolta con l'introduzione della primitiva select(), tale funzione consente di creare un server concorrente monoprocesso che possa accettare servizi molteplici.
   La funzione select si presenta così:

   ```
   int select(size_t ndfs, int* readfds, int* writefds, int* exceptfds, struct timeval* timeout)
   ```

   Dove il valore di ritorno è un intero e indica il numero di eventi occorsi e quali per mezzo di maschere. I parametri di ingresso, invece sono:
   - ndfs: numero massimo di eventi attesi.
   - readfds, writefds, exceptfds: le tre maschere che rappresentano dei file descriptor (o socket) su cui si attendono eventi.
   - timeout: valore di timeout che se settato a 0 rende la select() una funzione sincrona bloccante.

Java RMI

1. **Parla dello stub e dello skeleton in Java RMI.**

Lo stub è un proxy locale dal lato client che riceve invocazioni destinate all'oggetto remoto, le quali, vengono serializzate e inviate allo skeleton, proxy lato server, entità remota che riceve il riferimento all'oggetto remoto la quale deserializza e invoca il registry su cui sono registrati i servizi remoti, ottiene la risposta, la serializza e la invia allo stub che, a sua volta, la deserializza e restituisce il risultato al client.

2. **Definire i livelli di un'architettura RMI e le entità coinvolte.**

RMI utilizza il pattern proxy, l'oggetto remoto non viene riferito direttamente ma attraverso una infrastruttura a più livelli. Le entità in gioco sono: cliente, servitore e registry, cliente e servitore sono sincroni bloccanti, perché interagiscono mediante dei proxy (stub e skeleton).
I proxy vengono chiamati stub (lato client) su cui vengono effettuate le invocazioni destinate all'oggetto remoto e skeleton (lato server) che riceve le invocazioni fatte sullo stub e le realizza effettuando chiamate sul server.
Il RRL si occupa di gestire i riferimenti agli oggetti remoti e astrae delle connessioni stream, il livello di trasporto usato è sempre con connessione (TCP).
Il RMI Registry è il sistema di nomi che consente al server di pubblicare un servizio, i clienti per ottenere e raggiungere il riferimento remoto fanno una richiesta al RMI Registry attraverso il proxy stub.

3. **Cos'è il RMI Registry? A cosa serve? Come si interagisce? È obbligatorio un riferimento al registry?**

Il RMI Registry è un servizio di nomi.
Per attivare il RMI Registry bisogna utilizzare il programma rmiregistry di Sun, si interagisce con esso tramite funzioni come la bind e rebind invocabili solo da dove è in esecuzione il registry.
Il registry è in vero un server RMI, lo si può anche creare all'interno del codice server mediante createRegistry().
Il RMI Registry viene usato anche per ovviare al problema del bootraping (un client in esecuzione su un macchina ha bisogno di localizzare un server a cui vuole connettersi, che è in esecuzione su un'altra macchina.)

4. **Cos'è il Remote Reference Layer RRL?**

Risiede nella Java Virtual Machine di cliente e servitore in una architettura RMI, è responsabile della gestione dei riferimenti agli oggetti remoti, dei parametri e della astrazioni delle connessioni stream. Quando trasferisce le varaibili per riferimento trasferisce l'intero grafo delle relazioni di dipendenza.

5. **Come avviene il passaggio dei parametri nei metodi remoti in RMI?**

In RMI il passaggio di parametri avviene:

- Per valore se di tipo primitivo.
- Per valore (deep copy) se oggetti serializabili.
- Per riferimento remoto se oggetti remoti.

Oggetti serializzabili: Oggetti la cui locazione non è rilevante per lo stato. Sono passati per valore: ne viene serializzata l'istanza che sarà deserializzata a destinazione per crearne una copia locale.
Oggetti remoti: Oggetti la cui funzione è strettamente legata alla località in cui eseguono (server). Sono passati per riferimento: ne viene serializzato lo stub, creato automaticamente dal proxy (stub o skeleton) su cui viene fatta la chiamata in cui compaiono come parametri.

RPC, Implementazione e sistemi di nomi

1. **Definisci il modello architetturale alla base delle RPC**

Il modello architetturale che sta alla base delle RPC è costituito

- XDR (External Data Rapresentation) e una conversione di dati simile all'interfaccia RMI Server. Per dichiarare dati cotanti e strutture utili per la dichiarazione delle procedure remote.
- RPCGEN (Remote procedure call generator) strumento con una funzione simile ad un compilatore, malgrado non possa essere considerato tale, usato per convertire i dati di un file XDR (con estensione .x) in file stub per cliente e servitore.
- Port Mapper: servizio di nomi simile al rmiregistry, utile per reperire i servizi quando dal lato client vengono invocati. Il port mapper è un unico processo che gestisce le tabelle chiamate port map dove vengono memorizzate le informazioni, riguardate i servizi registrati attraverso la registerpc() e la svc_run().
- Network File System: File system di Sun.

Le entità fondamentali sono tre: cliente, servitor e portmapper (sistema di nomi per le procedure remote). Il cliente crea un gestore di trasporto, una struttura dati usata per tenere traccia e comunicare con i server. Anche il server crea un gestore di trasporto per mantenere il collegamento con i potenziali clienti e con il servizio corrente.
Il port mapper invece si occupa di fornire ai clienti il numero di porta a cui risponde la procedura da chiamare (inviando numero e versione di programma). Su ogni porta possono risiedere più procedure.

2. **Descrivi le entità coinvolte in RPC e le operazioni effettuate distinguendo tra cliente e servitore.**

Le entità coinvolte in una implementazione RPC sono stub lato cliente, stub lato server e port mapper.

- Lato client: consulta il portmapper per ottenere la porta su cui invocare la procedura remota, insieme ad argomenti invoca la procedura e passa i dati al gestore di trasporto (stub) che si occupa di realizzare il supporto RPC e inviare il messaggio.
- Lato server: il gestoredi trasporto (stub) riceve il messaggio, legge gli argomenti, chiama la procedura e passa i risultati nuovamente allo stub che realizzerà il supporto e invierà una risposta al client.

3. **Confronta i sistemi di nomi RMIRegistry e Portmapper**

Sono entrambi SISTEMI DI NOMI fondamentali per il funzionamento delle architetture, rispettivamente, RMI e RPC. Consentono entrambi di far registrare dei servitori che forniscono servizi e consentono ai clienti di trovarli.
La differenza sostanziale sta nel fatto che con il registryRMI il cliente ottiene effettivamente un riferimento remoto all'oggetto che risiede sul server, su cui chiamare la procedura e agire direttamente, mentre con il portmapper i clienti ricevono la porta su cui fare la chiamata e l'azione di per sé viene fatta dal server e non dal cliente che richiede il servizio.

4. **Descrivi il portmapper, quali sono le attività e ciò che privilegia.**

Il portmapper è un processo demone che gestisce la tabella port_map presente in ogni nodo RPC. Servizio di nomi (server RPC) che mantiene una tabella di corrispondenze tra numero di programma, versione e porta su cui insiste la procedura. Il port mapper registra i servizi sul nodo e permette di inserire o eliminare un servizio, recuperare la porta da una corrispondenza astratta per la procedura remota e fornisce supporto all'esecuzione remota.
Per l'interrogazione abilita due gestori di trasporto propri, uno TCP e uno UDP, sulla porta 111. A default utilizza solo UDP.

5. **Come funziona il portmapper?**

Il portmapper è un processo demone che gestisce le tabelle su ogni nodo allocando dinamicamente dei servizi e associando due porte alla tripla <programma, versione, protocollo> una per socket TCP e una per socket UDP.
Il portmapper registra servizi e offre procedure per per: inserire servizi, eliminare servizi, lista delle corrispondenze, corrispondenza tra associazione astratta e porta.

6. Semantica RPC Sun

L'implementazione RPC di Sun ha una semantica del servizio UDP del tipo at-least-once, con server sequenziale e cliente sincrono bloccante. L'implementazione di Sun include XDR, RPCGEN, PORTMAPPER e NFS. A default si fanno diverse ritrasmissioni dopo un breve intervalo di timeout.

7. RPC, descrivi le responsabilità dei vari componenti e cosa fanno (chi genera, che funzioni hanno)

I componenti di RPC sono:

- File XDR: file con estensione .x dove vengono rappresentati i dati, è costituito da due parti, la prima dichiarazione dei dati utilizzati nella parte di dichiarazione della procedura che saranno individuate da un numero di programma, versione e procedura.

CLIENTE: il cliente ha un corpo semplice, viene usato per la chiamata della procedura remota passando i parametri in ingresso al gestore di trasporto.

SERVITORE: non ha main, poiché la procedura remota viene invocata nello stub, vengono dichiarate eventuali strutture dati e implementati i servizi a cui vengono fatte le invocazioni. Gli argomenti di ingresso e di uscita sono passati per riferimento.

STUB: sia da lato client e server sono generati mediante rpcgen che dal semplice file .x genera gli stub sia per il cliente che per il servitore. Nello stub del server vengono invocate le procedure remote definite nel codice del server.

Servizi applicativi UNIX

1. **Descrivi FTP**

FTP (File Transfer Protocol) è uno dei protocolli di accesso e trasferimento di file che ci consente di lavorare su un file system remoto. Dopo aver stabilito una connessione TCP/IP possiamo autenticarci con user e pwd e lavorare sul file system come se ci trovassimo in locale (usando delle operazioni di get, put, ls, cd, i.e.)
L'implementazione del protocollo si basa su server paralleli, quindi prevede un approccio concorrenziale da parte di più client e un solo server. In pratica si hanno almeno due collegamenti per ogni cliente, una connessione di controllo che checka che il server sia sulla porta di default (21), il server fa una listen e il client una connect, quindi svolge un ruolo attivo. Oltre a questa è presente una seconda connessione data: la porta su cui si aggancia sul server è la 20, qui il ruolo attivo viene svolto dal server che fa la connect. Fatto ciò è possibile, per il client svolgere le sue funzioni di get e put per chiedere il trasferimento di file.

2. **Descrivi SMTP**

Simple mail transfer protocol, è un protocollo standard usato per il trasferimento di mail (sincrono non bloccante). I ruoi di send e receiver possono essere invertiti poiché vi è la facoltà di poter rispondere ad un messaggio in ingresso nel senso opposto. Esistono altri protocolli come POP e IMAP.

3. **Descrivi il protocollo NNTP**

Network News Transfer Protocol è un protocollo usato per l'invio di news. Si basa su TCP porta 119.

4. **Descrivi Telnet**

Virtual Terminal Protocol. Protocollo per sistemi TCP/IP con modello cliente servitore sincrono, in cui il cliente stabilisce una connessione TCP con il server, questo accetta le richieste di connessione e aspetta eventuali richieste. Il server gestisce il servizio e un figlio ad ogni sessione mediante la creazione di un processo demone.

5. **Descrivi NVT**

NVT (Network Virtual Trminal), di cui un esempio sono telnet o ftp, protocollo con connessione TCP/IP, si utilizza un terminale virtuale standard nella rete così da poter effettuare conversioni da terminale locale a standard virtuale.

6. **Descrivi RLOGIN**

RLogin è un servizio di login remoto su un'altra macchina UNIX. Si accede alla home directory e in caso di inesistenza si accede alla radice della macchina. Usa TCP/IP e lavora un carattere alla volta, più semplice di telnet ma limitato e non ottimizzato.

7. **Descrivi Usenet News.**

Usenet News è un servizio di condivisione di news e scambio messaggi basato su protocollo NNTP, lavora su TCP, un client news locale mantiene le news in modo che gli altri clienti possano leggerle. Il client si coordina con il server per ottenere queste informazioni.

OSI

1. **Descrivere brevemente cosa sia OSI**

OSI è uno standard id comunicazione tra sistemi aperti, definisce delle specifiche di protocollo ma non alcuna applicazione diretta. Suddiviso in livelli così da poter suddividere anche i compiti in modo che ci sia trasparenza e astrazione. Ogni livello ha il compito di comunicare con il pari.
OSI ha una struttura in descrizione verticale in cui si descrive il protocollo e una descrizione orizzontale in cui si specifica il protocollo stesso, quindi è la parte in cui si realizza il servizio.
In un livello ogni elemento attivo è detto entità e tra due entità vicine è presente una interfaccia logica chiamata SAP contenente alcune API.
In OSI è presente la SDU (Service Data Unit) presente per ogni sessione e tramite la quale si richiedono i servizi alla S-SAP. La PDU (Protocol Data Unit) invece è il protocollo con cui si mandano i messaggi tra i pari.

2. **Primitive OSI**

Connect per la connessione, Data per l'invio di dati, Disconnect per la disconnessione. Inoltre altre forme primitive usate sono la Request per richiedere un servizio e response e confirm.

3. **Livello di rete**

Si occupa della realizzazione del routing tra le diverse reti e i suoi compiti sono di indirizzamento, controllo flusso e congestione.

4. **Livello di trasporto**

Si occupa del trasporto dei dati e può spezzare e ricomporre i dati. Il trasporto può avvenire unendo o decomponendo i flussi di trasporto. Il livello di trasporto può essere visto come parte di separazione tra livello di comunicazione e applicativo. (Un esempio è TCP)

5. **Livello di sessione**

Si occupa del supporto al dialogo tra le entità e offre servizi per aperture e chiusura di connessione, gestione delle interazioni, eccezioni e sincronizzazione.

6. **Livello di presentazione**

Si occupa di offrire i servizi per trasformare la codifica dei dati ricevuti e da mandare tramite compressione e crittografia. Se tra le parti c'è completa uniformità allora non si effettuano trasformazioni.

7. **Livello di applicazione**

Livello che si interfaccia con l'utente finale e che ha come obiettivo quello di creare astrazione e nascondere le complessità dei livelli sottostanti, questo tramite servizi indipendenti dal sistema. OSI lavora ad oggetti con unicità di nomi tramite X.500 che altor non è che un servizio di directory che suddivide i nomi in attributi rendendoli facili da trovare tramite ricerche sia globali che locali.

Reti e Routing

1. **Descrivi cosa sia il protocollo IP**

IP (Internet Protocol) è il protocollo utilizzato a livello 3, ovvero quello di Network, è un protocollo a pacchetto senza connessione, quindi si basa su una semantica a bassa affidabilità best-effort. I pacchetti che non raggiungono la destinazione saranno scartati e non avremo alcun controllo di flusso, per questo motivo a tale scopo si fa uso di protocollo a livello superiore (trasporto) con maggiore affidabilità come TCP. A livello network vengono usati IP address per assegnare un piano di indirizzamento e quindi un percorso di raggiungibilità.
Gli indirizzi IP logici si commutano in indirizzi MAC fisici attraverso ARP.

2. **Descrivi in linea generale il routing**

Il router è il mezzo che si occupa di risolvere il problema dell'interconnessione tra mittente e ricevente mediante proprio il routing. Questo deve essere tollerante ai guasti, semplice e ottimale. I routing può essere locale (decisioni a bassa visibilità) o globale (con visuale su tabelle con tutte le possibilità di connessioni).
Può anche essere statico (con cammini di propagazione fissi) o dinamico (quando i cammini variano).
Infine anche adattativo (se sfrutta dinamicamente risorse che si liberano) o non adattativo altrimenti.

3. **Quali sono le famiglie di protocolli di routing più usate e cosa le distingue tra loro e da altre strategie?**

Le due famiglie principali sono Distance Vector e Link State, entrambe basate su tabelle e dinamiche con overhead. Nella prima ogni gateway la tabella mantiene la sola distanza in passi e il primo passo in uscita per il routing, le tabelle qui sono minime. Avviene una propagazione locale delle tabelle ad ogni vicino che e in caso di cambiamento ci possono essere dei problemi di convergenza tra le tabelle. Nel Link State, ogni nodo mantiene tutto il grafo, così da poter spedire messaggi diversi su cammini diversi e si tende a limitare la propagazione delle informazioni. Entrambi sono poco scalabili per via dell'essere globali e non locali. DV tutti i messaggi seguono gli stessi cammini, diversamente dal LS che può avere più cammini, controlla i nodi vicini e rileva i guasti.

4. **Descrivere la parte finale del protocollo TPC ovvero la disconnessione e metterla in relazione con le funzioni socket.**

In tcp la fase finale avviene tramite una chiusura graceful o dolce, in cui si chiude definitivamente solo un verso senza perdere messaggi. La chiusura avviene in 4 fasi, A invia il segmento Fin dopo l'invio dei dati, TCP manda da A a B l'ack e al termine del traffico applicativo B invia ad A il segmento Fin per confermare la disponibilità di chiudere, A riceve il segmento e avviene la chiusura delle connessione. I dati in uscita terminano in modo controllato, come avviene tramite la funzione shutdown() che chiude un lato della socket in modo dolce, invece la close() chiude sia in uscita che in entrata , distruggendo subito la socket.

Internet

1. **Descrivi cosa sia lo switching**

È la tecnica di instradamento che permette di dedicare le risorse a più richieste in tempi diversi, quindi rendendo le risorse condivise. Lo switching può avvenire tramite canali dinamici oppure usando datagrammi senza connessioni. Esistono vari tipi di switching, come il Circuit Switching, una tecnica pessimista e statica in cui si mantiene un canale per un flusso di dati atteso, impegnando sempre le risorse.
Invece nel lo Switching a datagrammi, si usano datagrammi che possono essere mandati tra nodi vicini in modo indipendente e senza canali o circuiti, ma senza controlli o QoS. Ci può essere anche switching a messaggi o pacchetti, dove si usano messaggi di lunghezza diversa o pacchetti di dimensione fissata e quindi uguale per tutti.

2. **Descrivi cosa sia il bridge**

Apparato che collega e separa 2 o più reti a livello datalink. Ci possono essere vari tipi, bridge multiporta, in cui si collegano più segmenti di reti diverse o bridge trasparenti che lavorano in modo invisibile all'utente. Il bridge ha una fase di learning in cui controlla le esigenze di comunicazione per adeguarsi ed effettuare i filtri. I bridge usano anche l'algoritmo spanning tree con cui si scambiano messaggi per trovare costi minori e creare un albero unico.

3. **Algoritmi di Routing**

SHORTEST PATH FIRST: ogni nodo crea un grafo completo dell'interconnessione e il traffico va verso il cammino più corto. DISTANCE VECTOR e LINK STATE
Algoritmo Multipath, in cui si indentificano più percorsi per uno stesso destinatario.
BACKWARD LEARNING, in cui ogni messaggio contiene la distanza dal mittente.
PATATA BOLLENTE: il traffico va verso la coda più vuota
RANDOM: a caso

4. **Descrivi il protocollo IPv4**

Prescrive il modo in cui si implementa l'instradamento dei datagrammi ip, che sono l'unità base di informazione su internet, incapsulati in frame.

5. **ARP**

Protocollo per la ricerca dell'indirizzo fisico di un nodo a partire dall'indirizzo IP. ARP ha un ruolo attivo, dove inizialmente si consulta la memoria cache e in caso negativo si comunica con ARP BROADCAST, invece quello passivo risponde alle richieste di altri. Molto usato.

6. **RARP**

Protocollo che ricerca l'indirizzo ip per i nodi che conoscono solo l'indirizzo MAC. Lavora tramite rete fisica e ritrasmette tramite server RARP. Deprecato.

7. **DHCP**

Protocollo per l'attribuzione dinamica di indirizzi IP, con risparmio rispetto a IP statici. Si basa su due ruoli, quello del client e server, con protocollo di offerta tipo asta a più fasi, con iniziativa del cliente. Alla scadenza dell'offerta, la si rilascia. Utile per la gestione di molti host di un'organizzazione.

8. **NAT**

Protocollo per la traslazione di indirizzi intranet privati in indirizzi ip globali. Tramite router Nat che traslano mantenendo tabelle. Nat a porte: si usa 1 solo indirizzo esterno per gli interni così da impegnare meno risorse.

9. **ICMP**

Protocollo di gestione e controllo su IP per migliorarne la scarsa qualità best effort, inviando messaggi di controllo ed errore alla sorgente del messaggio. Questo tramite messaggi ICMP imbustati in datagrammi ip, in cui è contenuto l'head icmp e i primi 64bit del datagramma che ha causato il problema.

10. ARQ e CONTINUOUS REQUEST

Servono per avere più affidabilità e asincronia nella comunicazione. ARQ ritrasmette pacchetti lavorando tramite ACK che confermano la ricezione. Continuous request è usato da tcp, in cui si mandano messaggi in modo ripetuto fino a saturare la finestra di buffer->vedi foglio (18).

11. HEADER TCP

Formato da 5 parole da 4 byte e riporta alcuni CODE BYTE, ack come conferma messaggio, push per invio immediato, syn per stabilire connessione, fin per terminarla, rst per reset connessione. IN TCP ack cumulativo.

12. IDL

Linguaggi per la descrizione delle operazioni remote e generazione degli stub, in più definisce in modo astratto i dati. Un esempio è XDR.

13. X.500

Insieme di standard di nomi, articolato e completo, organizzato in albero logico detto directory. In x.500 i nomi vengono visti come attributi così da rendere facile la ricerca tramite query. Si usano filtri per le ricerche per la loro capacità espressiva con cui si possono usare condizioni logiche ed espressioni. Per la ricerca si fa bind con la directory e poi si delega ad agenti. Le directory mantengono informazioni su risorse eterogenee evitando duplicazioni, diversamente dai database, gli oggetti sono indipendenti tra loro e c'è più ottimizzazione siccome ci sono tante letture e poche scritture.

14. TCP-IP, come si definisce la dimensione della finestra di trasferimento? come si risponde a una presunta congestione?

La dimensione della finestra viene decisa dal ricevente siccome è esso che deve allocare memoria per ricevere i dati, una volta scelta, il mittente l'accetta così da poter inviare segmenti fino a saturare la finestra. In caso di congestione il mittente dimezza la dimensione della finestra di invio e raddoppia il timeout. Al termine della congestione ritorna con un transitorio con finestra piccola, chiamata anche slow start. TCP gestisce la congestione tramite un ulteriore finestra chiamata di congestione e tramite lo slow start in cui nella finestra del mittente si parte da una situazione iniziale dolce/fredda per arrivare ad una calda..