## Introduzione

Questo file è pensato come una grossa raccolta di Domanda - Risposta tratte dai Temi d’Esame di Impianti Informatici del prof. Baroni Pietro.

Formato:

* Le domande vanno scritte come intestazione 3
* Le risposte vanno scritte come testo normale
* Riportare sempre qua sotto i dati del TE da cui avete tratto

| Data | Domande Tratte |
| --- | --- |
| 2016-2017 (dal suo sito) |  |
| 2010-2011 | Solo pertinenti al programma |
| 2012-2013 |  |
| 2014-2015 |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# TE

2016-2017

1) Descrivere sinteticamente le quattro principali classi di sistemi di elaborazione individuate dalla tassonomia di Flynn. Descrivere quindi la suddivisione in sottoclassi della classe MIMD e descrivere in dettaglio un aspetto specifico che caratterizza il funzionamento dei sistemi

appartenenti ad una di tali sottoclassi, a scelta dello studente.

* Uma numa, problema cache

2) Anche facendo riferimento all'architettura astratta di un impianto informatico, descrivere a livello

generale i diversi livelli di garanzia sull'effettiva trasmissione che possono essere forniti da un

meccanismo di scambio di messaggi, discutendone vantaggi e svantaggi. Fare quindi un esempio

concreto di un sistema, protocollo o applicazione che utilizza lo scambio di messaggi,

descrivendone brevemente il funzionamento e commentando il livello di garanzia adottato in tale contesto.

* 5 modi
* RPC

3) Un amico vi dice "In ambito automobilistico le vetture con le maggiori prestazioni, come quelle

da corsa, non sono le più affidabili ed hanno problemi tecnici piuttosto frequenti rispetto all'auto

che io uso tutti i giorni. Si può fare un discorso del genere anche per i calcolatori ed i sistemi

informatici in generale? Mi verrebbe naturale pensare che i calcolatori ad altissime prestazioni

possano essere più delicati e più soggetti a guasti rispetto al mio PC di casa, perché i loro

componenti sono più stressati. E' proprio così? Altrimenti mi sapresti dire che tecniche si possono

usare per aumentare l'affidabilità di questi sistemi? Magari però i sistemi informatici sono più

deterministici delle automobili ed esistono tecniche per prevedere i guasti e anticiparli in modo che

in pratica non si verifichino mai. Se fosse così, avremmo dei sistemi che funzionano praticamente in

eterno, non come le automobili. Tu che ne dici?" Cosa gli rispondereste?

1. Alte prestazioni = non affidabile? Non è vero un cazzo, ci sono supercomputer con sistemi di fault tolerance e cazzo ne so
2. Alte prestazioni = più guasti, più stress.
3. Tecniche per affidabilità -> FT, componenti secondari, backup e repliche dati, tecniche di recupero dati lock o sessione, distribuire un’app/processo su piu nodi,
4. Si può prevedere alla buona, organizzare un piano di manutenzione o sostituzione dei pezzi, ma il guasto è una probabilità e non ho mai la certezza che qualcosa duri in eterno o che qualcosa non si rompa tutto in contemporanea vanificando gli sforzi di fault tolerance

4) Discutere a livello generale la differenza tra protocolli stateful e stateless nell'ambito delle

architetture client-server discutendone vantaggi e svantaggi. Fare quindi un esempio concreto di

protocollo stateful ed uno di protocollo stateless, descrivendone il funzionamento, evidenziandone

le caratteristiche specifiche legate alla presenza o assenza di stato, e commentando le motivazioni

che hanno portato all'adozione dell'approccio stateful o stateless nei due casi.

* Stateful: ha memoria dello stato, conosce il passato, lega le richieste e risposte
* Stateless: non ha memoria di nulla, ogni richiesta è eseguita a sè stante anche se viene dallo stesso client;
* Vantaggi stateful: messaggi piu brevi perchè ho lo stato che memorizza certe cose, possibile non rifare handshake, alcune app NECESSITANO di stato, permette meccanismi di locking sulle risorse, aumentano le prestazioni e meglio gestisce perdita di pacchetti
* Svantaggi: consuma memoria (orfani), problema degli ID, overhead connessione, se server crasha perdo stati (problema di recuperarli o gestire inconsistenze)
* Esempio con TCP e UDP
* Esempio con NFS v1 (stateless) e NFS v4 (stateful)
* Esempio con FW di livello trasporto (stateless vs stateful)

5) Definire in termini generali il problema dell'elezione del coordinatore, spiegare quali meccanismi

possono essere usati per svolgere tale elezione e fare esempi concreti di situazioni nelle quali

l'elezione si rende necessaria.

* Mutua Esclusione in contesto distribuito con coordinatore -> elezione con algoritmo bullo o algoritmo ad anello.
* Necessaria l’elezione per la mutua esclusione con 1 coordinatore oppure quando va perso il token e serve qualcuno che lo rigenera

6) Il seguente testo, tratto dal sito di un'azienda che fornisce soluzioni di sicurezza, descrive i

vantaggi di una soluzione innovativa per la protezione di web server contro attacchi DDoS rispetto

ad altre soluzioni disponibili sul mercato.

*A good conventional defence against DDoS attacks usually involves blocking a wide range of IP*

*addresses.*

*The main problem with this method, however, is that significant segments of the network*

*components such as public Wi-Fi networks, business centers, neighbourhoods and even small towns*

*could be set to offline mode to safeguard against further DDoS attacks.*

*A superior security company is usually regarded by the amount of the IP addresses it is able to*

*detect and block during a potential attack.*

*Solution:*

*Only malicious automated queries are blocked.*

*Legitimate users from IP address of “intruder” do not end up being affected. Even during an active*

*DDoS attack, basic business operations should function without any interruptions.*

*Ultimately, customers may continue using the service and the service provider will maintain its*

*visitor frequency and database.*

Si richiede di discutere e commentare il testo facendo preferibilmente riferimento alle seguenti

domande (con libertà di aggiungere ulteriori considerazioni personali):

1) Cosa sono gli attacchi DDoS e perché per difendersi vengono bloccati numerosi indirizzi IP?

Distributed Denial of Service, rete di macchine vulnerabili controllate per inviare un fottio di connessioni/messaggi per sovraccaricare/crashare la vittima. Per difendersi perchè le macchine attacanti sono nella stessa rete e io blocco tutta la rete (tutti gli indirizzi IP)

2) Perché questo metodo di difesa potrebbe escludere anche degli utenti legittimi e non solo gli

attaccanti?

Già risposto.

3) Cosa è ragionevole ipotizzare sul funzionamento della soluzione innovativa stante il fatto che

riesce a distinguere le richieste legittime da quelle ostili a parità di indirizzo IP di provenienza?

….

4) Spiegare la differenza tra protocollo HTTP e protocollo HTTPS. Quindi, anche tenendo conto del

punto 3, commentare a livello generale le possibili differenze tra tecniche di difesa per attacchi

DDoS condotti usando HTTP e tecniche di difesa per attacchi DDoS condotti usando HTTPS

2010-2011

1. Definire in termini generali il problema della consistenza delle cache in un sistema di elaborazione, fare alcuni esempi di contesti concreti nei quali il problema si pone e, per almeno uno di tali contesti, descrivere in dettaglio come tale problema viene risolto.

* Nasce in contesto multiprocessore con memoria condivisa, quando più processori copiano lo stesso dato in cache, nasce il problema della coerenza cioè fare in modo che quel dato non esista diverso in più cache ma sia sempre appunto coerente.
* Non garantire la consistenza significa rischiare prestazioni scarse, o alterare la logica dei programmi.
* In generale esistono le politiche di write through o write back per mantenere le informazioni delle repliche aggiornate.
* Uma ha snooper cache -> legge quello che passa sul bus e si accorge se qualcuno sta cercando di accedere/modificare dati nella cache -> gestisce quindi l’interazione tra due processori quando c’è la modifica di copie di dati in cache tramite due modalità
* Write Update: si aggiorna il dato nella propria cache leggendo la copia modificata
* Write Invalidate: chi deve scrivere lo fa e lo segna come modified, tutte le altre copie diventano invalide
* La write invalidate viene realizzato pensando che ogni blocco di cache ha associato un automa (che tiene traccia dello stato del blocco) a stati finiti implementato in hardware. Lo stato dell’automa evolve sulla base di azioni effettuate dal processore sul blocco, o azioni di altri sul blocco “snoopate” dal bus.E’ lo snooper che si occupa di aggiornare il bit di stato (non la cpu)
* Dopo un evento si effettua una transizione di stato e se serve anche una comunicazione sul bus
* Definire automa MESI se si vuole (M: modified, dato modificato e le altre copie invalide, E: exclusive, non esistono copie in altre cache, S: shared, esiste in almeno 2 cache, valido ovunque, I: invalid, il dato è stato modificato da qualcun altro nella sua cache)

1. Definire i principali parametri di valutazione nel progetto di una rete ad alte prestazioni considerando sia gli aspetti topologici sia quelli tecnologici e fare esempi di applicazione di tali parametri a casi concreti.

* Nella progettazione di una rete ad alte prestazioni ci sono due cose fondamentali: la topologia e la tecnologia di routing.
* Valutazione topologia:
* Numero link per nodo
* Numero link totale
* Diametro (max distanza tra due nodi)
* Distanza media (media distanza tra due nodi)
* Bisection bandwidth (nelle topologie regolari esiste un asse di simmetria, Bis BW indica quanti link sono tagliati da questo asse - indica il massimo parallelismo)
* Dimensionality: numero di percorsi alternativi tra due nodi qualsiasi (da un’idea di FT)
* Topologia poco scalabile a stella: un nodo ha un link, ci sono n nodi e n link, diamtro = 2, distanza media = 2, BIS BW non ha senso perchè dipende tutto da quanto lo switch centrale è rapido nel commutare pacchetti, dim=1 (no fault tolerance, peggio ancora se si spacca lo switch)
* Topologia logaritmica albero binario: dist massima cresce con il logaritmo in base due del nomero di nodi, ha nodi di commutazione e nodi di calcolo, ha due sottovarianti, una per risparmiare switch (li si vuole piu semplici ed efficienti possibile) oppure una per aumentare la dimensionality (e aumentare la FT)

1. Definire in termini generali il problema del bilanciamento del carico e fare esempi concreti di come esso può essere affrontato, considerando esplicitamente se ritenuto opportuno, diverse tipologie di carico e diverse architetture di elaborazione di riferimento.

* Il problema del bilanciamento del carico si pone in un contesto distribuito, dove ci sono reti di calcolatori che devono occuparsi di richieste provenienti dall’esterno. Esiste anche traffico interno alla rete ma quello è considerato un problema di scheduling e assegnazione del carico di cui si occupa il sistema operativo dell’impianto.
* Esistono tre modalità per distribuire il carico: 1. Si permette all’utente di scegliere a quale nodo inviare la richiesta (terribile non fatelo mai) 2. Si sfrutta un rotating dns (dns che sta davanti logicamente al sistema e ogni volta che riceve una richiesta di traduzione del nome simbolico dell’impianto, fornisce un IP diverso in base a quale server assegnare la richiesta; non particolarmente efficiente ma FT) 3. Si sfrutta un load balancer
* Il load balancer è un componente dedicato che sta logicamente davanti alla rete, ha un IP pubblico ed è il punto di ingresso di tutte le richieste. Rischia bottleneck e SPOF.
* Ci sono 4 modalità di configurare un LB, in 2 categorie diverse: bidirezionale e unidirezionale.
* Bidir: gateway o reverse proxy
* Unidir: tunnel o direct routing

1. Definire i concetti di file system di rete e distribuito, descrivere le proprietà che li caratterizzano e discutere anche facendo riferimento ad esempi concreti, alcune delle principali scelte progettuali relative alla loro realizzazione.

* Il NFS è tradizionale ma arricchito di accesso a file remoti. Quello distribuito (DFS) è concepito per virtualizzare le risorse di rete con visione unitaria, è caratterizzato da una marcata separazione tra gestione dello spazio dei nomi e della localizzazione dei file.
* Ci sono 5 proprietà di cui possono godere i filesystem, se hanno almeno il primo sono NFS, e più altre ne hanno più diventano DFS.
* **Access transparency**: non ci sono differenze nell’accesso ai file remoti o locali perché sono virtualmente sullo stesso sistema, stesse primitive usate.
* **Location transparency**: l’utente non vede differenza tra file locali e file remoti.
* **Login transparency**: le prime due assieme, “potersi connettere con delle credenziali da più macchine ma vedere sempre la stessa roba”
* **Location independence**: se da una macchina ho una certa visibilità di una risorsa/file, quando questa viene fisicamente spostata, io continuo a vederlo nello stesso punto.
* **Replication transparency**: la gestione delle repliche è completamente trasparente all’utente, il sistema sceglie in autonomia a quale/i accedere.
* **Concurrency transparency**: il FS fornisce meccanismi automatici per la gestione degli accessi concorrenti, chiaramente se vogliamo scrivere in due, ci sarà qualche messaggio o warning che avvisa l’utente ma più di così apposto
* Le scelte progettuali sono:

1. **Metodi di accesso** (remoto, download/upload, caching)

2. **Semantica dei file** (unix o session o file immutabili)

3. **Gestione molteplicità lettori scrittori** (single, multiple/remote, migrate, replicate)

* Un esempio è NFS, un protocollo di condivisione di file e risorse in rete nato stateless divenuto stateful. C’è FTP per il trasferimento e condivisione dei file. L’SMB è un protocollo per la condivisione di file, stateful.

1. Definire il problema dell’autenticazione tramite un canale insicuro, discutere a livello generale le principali tecniche alternative che possono essere utilizzate per la sua soluzione, confrontandone vantaggi e svantaggi e descrivere in dettaglio un esempio di applicazione di una delle tecniche considerate.

* Quando due controparti vogliono comunicare, può esserci la necessità di confermare le loro identità, voglio la garanzia che il mittente dichiarato coincida con il mittente reale. Di base si potrebbe pensare che sia sufficiente verificare che il mittente abbia criptato usando una chiave simmetrica nota solo a me e lui, o in contesto asimmetrico, inviando qualcosa criptato con la sua pubblica e sfidandolo a decifrarlo.  
  E’ proprio quest’idea della sfida che pone il meccanismo base per l’autenticazione: serve per verificare che l’interlocutore sia in grado di usare in quel momento la chiave (simm o asimm) che si sa lui possiede. Devo dimostrare che sono io a possedere la chiave adesso, che non è stata rubata e che è SOLO mia.
* L’idea di challenge response è (leggiti il cazzo di file) esempio a chiave simmetrica + asimmetrica

Vantaggio: Molto sicuro

Svantaggi:Rischio Man in the Middle + C-R, siccome usa funzioni matematiche complesse, è molto pesante

* Comunque alcune implementazioni sono vulnerabili nonostante si usino sfide matematicamente forti. -> spiega DH
* Un altro strumento è la firma digitale

Vantaggi:Molto semplice (fiducia pregressa)

Svantaggi:Problema di pubblicare la chiave pubblica oppure ottenere i certificati di autenticità

2012-2013

1. Un amico vi dice “A volte mi chiedo quali siano le differenze tra un sistema monoprocessore e un sistema multiprocessore. Sono diversi nell’HW o anche nel SW? Si tratta solo di mettere insieme più pezzi o c’è qualche complicazione più seria? Mi diresti quali caratteristiche o componenti in più o diversi deve avere un sistema multiprocessore? Mi faresti una descrizione dettagliata di una di queste caratteristiche o componenti?” Cosa gli rispondereste?

* Uma (snooper cache) e numa (directory) sistemi multiprocessore -> complicazione? Protocolli per gestire la coerenza della cache

1. Elencare e descrivere alcune delle alternative di progetto da considerare nella realizzazione di un’architettura C-S e, facendo riferimento specifico a tali alternative, fare esempi di scelte diverse adottate in sistemi concreti, spiegandone le motivazioni.

* Tier: suddivisione logica applicativa -> fat client, thin client (POP e IMAP)
* Middleware
* Molteplicità utenti
* Protocollo stateful o stateless
* Implementazione: singola macchina, tante macchine (DNS a struttura gerarchica multimachine)
* Sincronia messaggi (NFS con sincronia nei messaggi per file locking)

1. Definire i concetti generali di failover e failback, elencarne le diverse tipologie, e descrivere le tecniche necessarie alla realizzazione di uno specifico meccanismo di failover/failback.

* Failover (c’è un guasto, si fa la sostituzione) Failback (ripristino del componente kaboom)
* Probabilmente la sua tassonomia di merda?
* Meccanismo Failover in cluster -> nodi o lb (argomento: monitoraggio cluster)

1. Elencare alcuni dei diversi significati con i quali viene utilizzato il termine directory in contesti diversi nell’ambito degli impianti informatici, evidenziandone gli aspetti comuni e le differenze. Descrivere quindi in dettaglio la struttura e l’utilizzo della directory in uno dei contesti considerati.

* Directory NUMA
* Directory Service -> un componente di filesystem di rete/distribuito, gestisce lo spazio dei nomi dei file/cartelle

1. Fare alcuni esempi di protocolli, a diversi livelli architetturali, che includono funzionalità crittografiche e/o autenticazione forte. Descrivere quindi in dettaglio la struttura ed il funzionamento di uno di essi.

* File protocolli sicuri: esempi sono TLS/SSL, SSH, IPSEC
* Scegline uno

1. Definire i diversi livelli di RAID, descrivendone la struttura e le principali caratteristiche. Fare quindi esempi di come ad alcuni di tali livelli possano essere applicate valutazioni quantitative di affidabilità.

* RAID 0: no ridondanza, solo uso di tanti dischi (virtualizza n dischi in 1)
* RAID 1: ridondanza pura, uno dei due dischi è un clone, testine sincronizzate, resiste a un guasto, SCRITTURE STABILI
* RAID 2: come uno ma testine sync, resiste a un guasto, stripe piccole, codifica hamming K utili H ridondanza, letture rapide scritture no
* RAID 3: K dischi con 1 di parità testine sync e scritture penalizzate per la modifica del bit di parità
* RAID 4: K utili 1 di parità, stripe sono multipli di blocchi, scritture penalizzate ma adatto a qualsiasi tipo di carico => disco di parità è bottleneck perchè le letture sono parallele ma le scritture sono in serie
* RAID 5: blocco di parità distribuito, letture veloci e scritture parallele fino a K+½ adatto a ogni carico
* RAID 6: 2 dischi di parità distribuiti, resiste fino a 2 guasti contemporanei tramite erasure coding
* RAID DI RAID

2014-2015

1. Definire il concetto di SPOF e fare alcuni esempi di possibili SPOF in contesti diversi. Spiegare quindi a livello generale quali soluzioni tecniche si possono adottare per evitare l’esistenza di SPOF. Descrivere quindi un esempio specifico di applicazione di una di tali soluzioni, discutendo anche, se possibile, il conseguente miglioramente ottenuto in termini di probabilità di guasto.
2. Definire a livello generale la differenza tra architetture UMA e NUMA e discutere le conseguenze che derivano da tale differenza, con particolare riferimento al problema della consistenza della cache.
3. Spiegare i possibili vantaggi e svantaggi derivanti dall'adozione di un’architettura gerarchica a più livelli per la fornitura di un servizio e fare due esempi concreti di protocolli che si basano su un’architettura gerarchica, descrivendone il funzionamento.
4. Definire il concetto generale di cluster e discutere in base a quali caratteristiche possono essere definite diverse tipologie di cluster. Introdurre quindi una tecnica specifica utilizzata nell’ambito dei cluster e spiegarne il funzionamento.
5. Definire il concetto di firewall, spiegare come possono essere classificati i firewall da diversi punti di vista e fare esempi della loro possibile collocazione nell’ambito dell’architettura di un impianto informatico, motivando le diverse soluzioni esistenti.
6. Una sbrodolata.

TE 2011/2012

1. Nell’ambito della tass. flynn , definire il concetto di architettura multiprocessore, elencandone e caratterizzarne le diverse sottoclassi e descriverne in dettaglio un aspetto specifico del funzionamento di una di tali sottoclassi.

* Hanno stra tanto parallelismo.
* MIMD: Multiple Instructions Multiple Data ovvero più operazioni su più dati quindi anche lavoro di parallelismo e multi-thread quindi istruzioni diverse di thread diverse su dati diversi -> sottoclasse shared memory (UMA, NUMA e COMA in base al tipo di accesso alla memoria)
* UMA: tempo uniforme di accesso per qualsiasi processore a qualsiasi blocco di memoria.
* NUMA: ogni processore ha la sua memoria personale e il suo bus.
* Gabriele sceglie UMA. Ha lo snooper che controlla quando si vuole accedere a qualcosa nella cache. QUANTO CAZZO NE SO *sbatte le mani sul tavolo* no era la matita
* Aspetti del funzionamento: gestione coerenza cache, sincronia comunicazione, gestione molteplicità processi (parallelizzazione e scheduling-assegnazione carico ai core, gestione località cache)

1. Ci sono 3 alternative architetturali per la realizzazione di un sistema di storage high-end.

* C’è un modello che descrive queste architetture e quelle principali sono caratterizzate dal modo in cui virtualizzano la risorsa: DAS a livello dispositivo, NAS a livello file e SAN a livello rete

**DAS** -> Direct Attached Storage, le unità di storage sono collegate tramite un link agli host che visualizzano tramite sw con LVM se si tratta di un array di dischi oppure adeguato sw per visualizzare i blocchi come risorsa di storage.

**SAN** -> Storage Area Network, prevede una rete locale specializzata per l’accesso allo periferiche di storage, in questo modo è facile condividere i dati tra le reti senza bisogno di server per la gestione di IO, elevata scalabilità. Purtroppo costa

**NAS** -> Network Attached Storage, le unità sono collegate alla rete locale e non una specializzata. I client vi accedono tramite FS remoto, è meno costosa ma ovviamente il traffico di trasferimento file si mischia con quello della rete portando più facilmente a sovraccarichi.

1. Definire concetto di middleware elencare tipologie e descrivere in dettaglio la struttura e funzionamento di un esempio concreto.

* Middleware -> strato sw che sta nel punto di divisione della logica C-S. Serve per far comunicare le due controparti. Un’application logic che maschera le differenze fra HW e SW fra nodi. Fornisce un’interfaccia per le API che scambiano richieste e risposte
* Tipologie? Noi vediamo l’application middleware, che sta tra presentazione e application level.
* Message Oriented:
* Remote Procedure Calling: fa delle chiamate a procedure remote che stanno su una macchina esterna (server) e vengono eseguite lì. I risultati della procedura sono poi spawnati nello stack della macchina chiamante -> descrivere un po’ come funzionano gli stub
* Object Request Broker: si usano nei contesti di programmazione ad oggetti, si richiedono procedure che richiedono oggetti.
* P2P: peer to peer, sono tutti client e server, architettura che lavora coi socket.
* Infrastruttura RPC: ha quattro componenti 1. Interfacciamento a livello di chiamata (esiste un linguaggio per definire le chiamate middleware che lasciano l’implementazione al caso specifico) 2. Localizzazione (di solito RPC non si occupa di questo comunque alcuni offrono servizi di DNS, si da per scontato che i client conoscano il server a cui si parla) 3. Invocazione (generazione della stub dinamica vs statica in base alle specifiche IDL) 4. Comunicazione (stub hanno compito di marshallare, parlare un NEUTRO tra client e server)

1. Definire il problema del distributed snapshot e descrivere un algoritmo distribuito nella soluzione.

* Problema di definire un quadro globale a partire da informazioni locali. Nel contesto distribuito ci sono tanti nodi, tanti “locali”, bisogna capire se esiste un singolo che si fa il carico di disegnare questo quadro oppure se optare per una soluzione dove tutti “collaborano”: questo è il caso della distributed snapshot. Algoritmo chandy lamport -> un nodo inizia la registrazione dello snapshot, salvando i dati locali e mandando un marker su tutti i suoi canali in uscita. Un nodo generico marca tutti i canali su cui riceve un marker come VUOTO e ad ogni marker salva lo stato locale. Quando tutti i suoi canali sono VUOTI allora ha finito e invia al nodo iniziatore lo stato salvato. Alla fine il nodo iniziatore riceve lo stato locale di tutti i nodi, li mette assieme e ottiene una vista globale del sistema.
* I problemi nel fare questa snapshot sono i messaggi pendenti, il fatto che delle soluzioni non distribuite potrebbero non riuscire a tenere conto della parte dinamica, potrebbero generare inconsistenze e non riuscire a fare le foto insieme. Chandy lamport è intelligente 👍

1. Esempi protocolli e servizi destinati a supportare i compiti di amministrazione di un impianti informatico. Descrivere uno in dettaglio.

* DHCP -> permette di assegnare a ogni client di una rete di ottenere la configurazione per utilizzare la rete. Indirizzo ip, subnet mask, default gateway, dns server. Modalità automatica (assegnazione permanente da un pool) manuale (da una tabella un admin setta MAC <-> IP) dynamic (di passaggio, assegnazione temporanea, leasing, di un indirizzo IP)
* SNMP -> protocollo universale di amministrazione, vede i dispositivi come insiemi di variabili, sovrascrivibili e impostabili dall’amministratore tramite le sue stazioni di controllo. Un server SNMP si occupa di manipolare le variabili secondo i comandi admin. Ogni tipo/categoria di dispositivo ha un set di variabili associato, quindi server che abbiano anche una specie di ID o comunque un identificatore per distinguere le varie istanze.  
  Ci sono dispositivi che non parlano SNMP e quindi esisterà un SNMP agent che si frappone tra client e server e parla una lingua neutra per permettere un uso trasparente da parte dell’amministratore.
* NIS -> permette di configurare gruppi di macchine, c’è un NIS server depositario di file di configurazione e tutte le macchine di un certo gruppo accederanno a questi file per copiare la configurazione. Il server è spof e bottleneck potenzialmente. Alcune info comunque è meglio salvarle su file locali e non remoti.
* DIR SERVICE -> meccanismo/struttura per gestire e fornire informazioni strutturate gerarchicamente, informazioni lette più spesso che scritte o con esigenza di scalabilità. Esiste una struttura Directory Information Tree che contiene entries che associano nome cartella con nome cartella radice e con i suoi figli. Questa struttura è interrogata dai DSA, server del protocollo LDAP. Il protocollo LDAP va ad interrogare i DAS per farsi fornire entries DIT oppure informazioni su sottoalberi. Come il DNS implementa modalità di funzionamento iterativo o ricorsivo. DAS era costruito per iso osi e quindi LDAP sta sul tcp per mantenere l’architettura fino ad oggi. LDAP praticamente ti fa browsare i file.