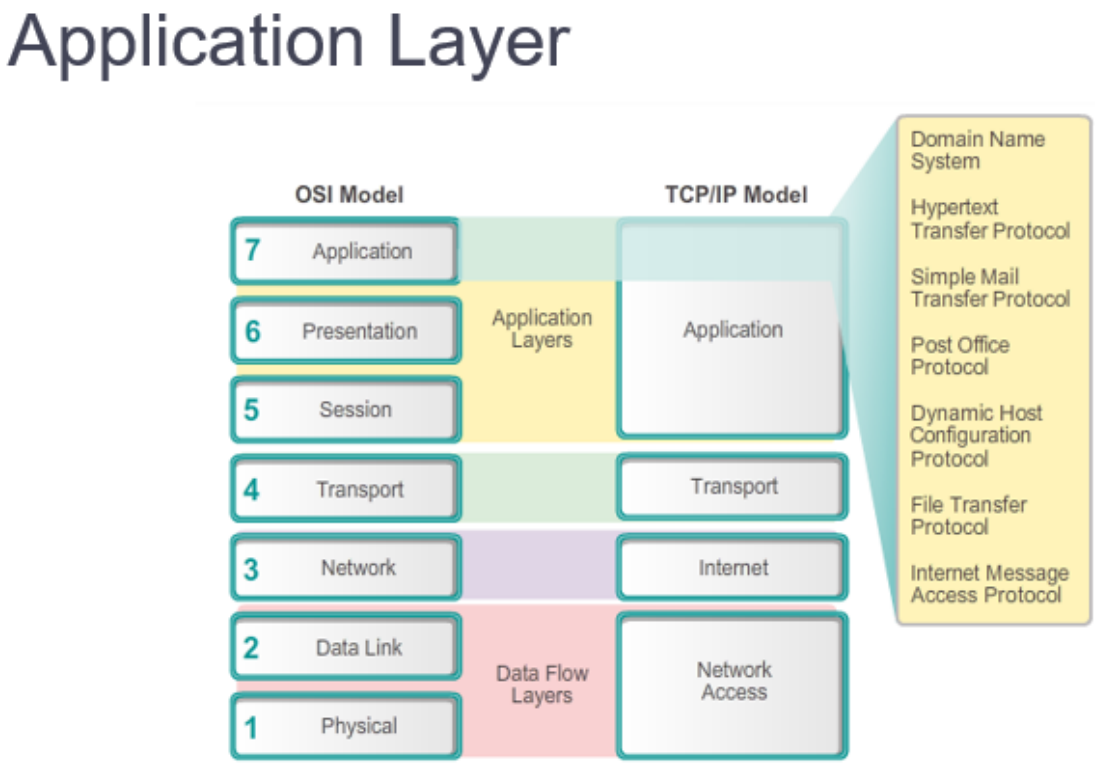
RDC

Data integrity → hash
origin authentication → public key
replay attack prevention → nonce
confidentiality → crittografia

Application Layer

Lo strato applicativo costituisce il livello più alto dei modelli OSI e TCP/IP.



Creare una network app significa scrivere un programma che funziona su differenti end systems e comunica mediante la rete senza la necessità di dover girare anche sui network-core devices (router, switches…).

Distinguiamo essenzialmente 2 tipi di architetture applicative

- Client-Server:
  dove i **server** sono host *always-on,* con IP STATICI, eventualmente scalati in

data centers, mentre i **client** sono host che comunicano necessariamente con i server per raggiungere altri client e possno avere IP DINAMICI

- P2P:
  Gli **end systems (peers)** che possono essere terminali o anche server comunicano direttamente senza passare da altri server, richiedendo o fornendo servizi anche in forma scalare quando più peers contribuiscono a soddisfare la domanda. Le connessioni sono intermittenti e gli ip sono dinamici.

Processi

Un processo è un programma che gira in un host. Più processi all'interno dello stesso host comunicano fra di loro tramite metodi definiti dall'OS, mentre fra differenti hosts la comunicazione avviene mediante scambi di messaggi.

Nell'architettura *client-server* i processi clienti iniziano la conversazione e i processi server aspettano di essere contattati. Nel *P2P* i peers sono allo stesso tempo client e server processes.

Lo scambio di messaggi fra processi avviene mediante l'uso di socket, ovvero delle porte che trasportano i messaggi in uscita negli strati inferiori e li elaborano in entrata. I processi devono avere degli identifier composti da IP_ADDRESS:PORT_NUMBER in modo da garantire la presenza di più processi (e socket) nello stesso host.

Servizi e protocolli di trasporto

Un applicazione può aver bisogno di servizi di trasporto differenti in base al suo uso:

- **data integrity:** per avere la sicurezza che un dato arrivi integro a destinazioni senza loss
- **timing:** per garantire un ritardo minimo nel trasporto.
- **throughput:** per garantire una quantità di banda minima necessaria
- **security:** per proteggere la lettura del contenuto da coloro che non sono ammessi a leggerlo

Essenzialmente distinguiamo 2 tipi di protocollo di trasporto:

TCP
+ è affidabile
+ ha funzioni di controllo di flusso e di congestione

UDP
+ throughput garantito
- tutto il resto

+ connected oriented
- no timing
- no throughput
- no sicurezza (si usa SSL)

E allora perchè si usa? si usa per quelle app che hanno solo bisogno di velocità di trasferimento senza badare troppo al resto come Streaming video, VoIP ecc.

HTTP

HyperText Transfer Protocol è il protocollo usato nelle applicazioni web basato sul modello client-server, utilizza il servizio TCP:

1. CLIENT inizializza una connessione TCP con SERVER su porta 80 (socket)
2. SERVER accetta connessione TCP dal CLIENT
3. Scambio di messaggi HTTP sotto forma di REQUEST RESPONSE tra browser (client) e web server
4. Si chiude la connessione TCP

Esistono due tipi di connessione HTTP:

Non persistent HTTP: Si trasferisce al massimo 1 oggetto alla volta durante una connessione TCP. Necessita di 2 RTT (Round time transfer) per ogni trasferimento.

Persistent HTTP: Si trasferiscono più informazioni in una singola connessione TCP e la connessione resta aperta dopo la response.

HTTP è stateless: i server non mantengono informazioni sui client che hanno fatto le request. Ma una tecnica per aggirare questo problema è con l'utilizzo di cookies.

I **cookies** sono utilizzati per autorizzazioni, carrelli della spesa, raccomandazioni e profilazioni, sessioni di login ecc.

I **proxy server** sono usati come intermediari fra client e server per vari scopi come il web caching, una tecnica che permette di salvare precaricamenti di informazioni dei server in modo da ridurre il carico di richieste ai server originari, aggiornando le stesse risorse nei proxy solo quando vengono modificate dai server originari.

Protocolli Mail

Un sistema di posta elettronica si compone di 4 elementi:

User agent: è il terminale che legge/scrive email

Mail server: è il server che contiene la posta in arrivo (mailbox) e quelle in uscita (message queue), comunica con altri mail server con il protocollo SMTP

SMTP (Simple Message Transfer Protocol): Usa il servizio TCP per il trasferimento di email tramite la porta 25.

Mail access protocols: Sono i protocolli tramite cui un User agent consente la ricezione delle mail e sono: POP (post office protocol), IMAP (Internet Message Access Protocol) e anche HTTP.

DNS

è il protocollo che si occupa di tradurre gli hostname di siti,host,mail,server in indirizzi ip a cui sono associati.

Sistema distribuito decentralizzato

Si tratta di un database distribuito decentralizzato il cui protocollo opera nello strato applicativo. Non è centralizzato perché sarebbe esposto a single point of failure, problemi di volume di traffico e problemi di gestione a distanza del db centralizzato.

Il database del DNS è approsimabile ad un grafo gerarchico dove, a partire dalla radice si cercano le associazioni host-ip prima in base al **Top-Level Domain servers** (.com .org .it …) o ad **authoritative DNS servers** (organizzazioni che possiedono proprio DNS servers) e poi in base all'host o all'ip dato in query.

Se non si riesce a risolvere il dominio vengono contattati i **root name servers** che si occupano di risolvere l associazione richiesta.

Modalità di risoluzione del DNS

Le query di ricerca passa per un **local DNS server** (ogni ISP ne ha uno) che potrebbe risolvere la richiesta se ha salvato in cache la stessa già fatta da qualcun altro.

- Iterativa: A partire dalla Root: Contatta il server, riceve come risposta il server da contattare finchè non trova il server che risolve l associazione
- Ricorsiva: Manda il nome nella root finchè in modo ricorsivo non viene risolta da qualcuno la query

P2P architecture

L’architettura Peer 2 Peer è un sistema basato sulla comunicazione diretta dei terminali (peers) senza la necessità di avere server sempre accesi. Viene usato principalmente per la distribuzione di file (BitTorrent) lo streaming e VoIP.

File distribution time

Per trasferire un file, oltre ai server, anche i client si comportano da server. A differenza della condivisione client-server il cui tempo di distribuzione cresce linearmente, nel p2p cresce asintoticamente (meglio).

Video streaming e Content Distrubtion Networks

Per gestire la distribuzione di massa di contenuti ad alto volume di contenuti si usano tecniche come DASH (Dynamic Adaptive Streaming over HTTP) che divide i contenuti video in frammenti codificati a differenti qualità ratio e che vengono “scelti” dai client in base alla loro banda disponibile, o facendo uso di server multipli che distribuiscono la stessa risorsa

Protocolli asincroni

Prima dei protocolli asincroni, per gestire le applicazioni che lavoravano in modo asincrono si usava una tecnica dell’http detta Polling che consisteva nel chiedere ciclicamente ad un server informazioni da far pervenire alle applicazioni o meno (sincronia ciclica).

AMQP

Advanced Message Queuing Protocol è il protocollo più diffuso per la realizzazione di protocolli asincroni. Si basa sullo smistamento di messaggi (eventi) in code in modo diretto o indiretto da parte di un broker.

Mentre in HTTP abbiamo chiamate REST, in AMQP abbiamo chiamate RPC (Remote Procedure Control). L’utilizzo di protocolli asincroni è sempre più importante e diffuso nell’IoT per l’efficienza, la rapidità e la leggerezza che offre per la comunicazione.

Uno dei broker più diffusi per il protocollo amqp è RabbitMQ

Websockets

Sono la trasposizione del concetto di socket a livello web (html5). Si tratta di un protocollo full duplex channel che permette la comunicazione diretta fra client e host remoto in modo rapido, a basso consumo di risorse e bypassando eventuali firewalls e proxies.

Si basano sempre sul protocollo HTTP (quindi con richiesta iniziale per stabilire una connessione) e instaura un collegamento con scambio di messaggi della dimensione del byte.

Sicurezza nella rete

Principi della network security:

- **confidenzialità:** Le *informazioni* sono disponibili solo per le persone intese ad usarle o vederle.
- **Integrità:** Le *informazioni* sono modificate in modo appropriato solo dalle persone autorizzate a farlo.
- **Disponibilità:** Le *app* e i *servizi* sono resi disponibili con performance accettabili.
- **Autenticità:** L'*identità* di una persona viene determinata prima di garantirne l'accesso se l'anonimato non è autorizzato.
- **Autorizzazione:** Le *persone* sono autorizzate o meno ad accedere alle app e ai servizi.
- **Non ripudio:** Le *persone* non possono eseguire azioni e poi negare di averlo fatto

Crittografia

È la soluzione scelta per garantire la confidenzialità delle informazioni condivise, e la verifica delle identità.

Ci sono vari tipi di attacchi a schemi di cifratura:

- Se non si conosce il testo in chiaro → si fa brute force delle chiavi di crittografia e statistiche
- Se si conosce il testo in chiaro → si determina l algoritmo di cifratura studiando la crittografia da mono a n-alfabetica.

- Se si può avere il testo cifrato → si decide di mandare del testo alternativo a partire da testo in chiaro scelto.

Crittografia simmetrica

Gli interlocutori condividono la stessa chiave di crittografia. Sono crittografie veloci perché fatte con operazioni elementari. Le chiavi di crittografia più conosciute:

- **DES (Data Encryption Standard):** ha chiavi a 56bit con blocchi in chiaro di 64bit. Non esiste una strategia nota (ad eccezione del brute force) in grado di romperlo, ma la chiave è calcolabile in meno di un giorno, infatti è in disuso.
- **AES (Advanced Encryption Standard):** è l'attuale chiave simmetrica utilizzata, processa dati a 128bit ed ha chiavi fino a 256bit. Gli attacchi brute force che impiegano 1 secondo a decifrare DES, impiegano 149 trilioni di anni per decifrare allo stesso modo AES.

Crittografia pubblica (asimmetrica)

Ad una chiave privata viene associata una pubblica tramite cui farsi riconoscere dagli altri. Queste chiavi devono essere verificate da autorità di certificazione. Questi algoritmi sono molto più pesanti e lenti di quelli simmetrici, per questo vengono usati solo per la verifica dell'identità e poi la simmetrica per lo scambio dei messaggi. Esempi di algoritmi più conosciuti:

- **RSA (Rivest, Shamir, Adelson algorithm):** si basa sulla creazione di una chiave pubblica e privata che usate l'una sull'altra e viceversa restituiscono lo stesso risultato (verifica dell'identità) ma alla base della sicurezza di tale algoritmo vi è il trovare il valore restituito dal prodotto di due fattori il cui calcolo è un problema Np Hard (dato n noto molto grande, =p*q trovare p e q è decisamente difficile).

Siano p e q due grandi numeri primi (1024 bits ciascuno) ed n = p*q scelti e e d secondo altri procedimenti si definiscono chiave pubblica quella costituita da (n,e) e chiave privata quella da (n,d).
Dato un messaggio m Priv(Pub(m)) = Pub(Priv(m)) = m.

Autenticazione

Il problema dell' autenticazione consiste nel riuscire a dimostrare la propria identità senza che un altra persona sia in grado di fingere di essere me stesso. Un metodo per certificarlo è con:

Firme digitali

Si tratta di una tecnica crittografica analoga alla firma scritta a mano. Si basa sul concetto di crittografia pubblica dove il possessore della chiave pubblica è un'autorità di certificazione che memorizza la coppia [chiave pubblica, persona fisica] e questo garantisce anche il non ripudio. Ma poiché il processo di crittografia pubblica è pesante, vengono usate hash function (md5 o SHA-I) per ridurre le informazioni a dimensioni predefinite in modo irreversibile.

SSL/TLS (trasport layer)

Secure Sockets Layer è uno protocollo di sicurezza che si inserisce fra App. layer e TCP/IP layer sviluppato da Netscape nel 1994 che offre confidenzialità, integrità e autenticazione. Oggi è stato aggiornato a TLS (transport Layer Secure). I passaggi del protocollo sono i seguenti:

- **handshake:** è il processo di autenticazione tramite certificati che avviene fra due dispositivi che consiste nella condivisione di una master secret key.
- **key derivation:** a partire dalla MS vengono generate 4 chiavi per la cifrazione e l'integrità (con Message Authentication Code) dei dati scambiati fra client e server
- **data records:** i dati vengono divisi in serie di records ognuno contenente un MAC il tipo di dato e il numero della sequenza spedita (o il finale)

IPsec (Network-layer)

Anche a livelli più bassi è possibile garantire la confidenzialità con le VPN e IPsec (praticamente quello che fa SSL ma nel network layer con IP ovvvero [*data integrity - hash], [origin authentication - public key], [replay attack prevention - nonce], [confidentiality - crittography].*

I protocolli che usa IPsec sono:

- **Authentication Header (AH):** per garantire autenticità e integrità
- **Encapsulation Security Protocol (ESP):** per garantire ANCHE la confidenzialità. (quindi meglio quest'ultima).

Firewall

è uno strumento che si occupa di isolare la rete interna di un'organizzazione dal'internet permettendo il passaggio o meno di pacchetti, in questo modo previene attacchi DDOS, accesso e modifica illegale ai dati. Esistono 3 tipi di firewall:

- **stateless packet filters:** la rete interna è connessa ad internet tramite firewall routers che filtrano packet-by-packet decidendo quale inoltrare e quale no sulla base di: IP address, TCP/UDP port numbers, ICMP message type, TCP SYN e ACK bits. Fa uso di Access Control Lists per applicare queste regole.

Policy	*Firewall Setting*
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

- **Stateful packet filtering:** il problema del tipo precedente è che è uno strumento pesante e può creare regole insensate. Questo tipo di firewall studia e prende decisioni sulla singola connessione TCP.
- **Application gateways:** filtrano i dati dei pacchetti a livello applicativo (tipo HTTP)

I firewalls e i gateways presentano delle limitazioni:

- **IP spoofing:** i router non possono garantire la veridicità della sorgente dei dati ricevuti
- Ogni applicazione necessita del suo gateway
- I Client Software devono sapere come contattare il gateway (ad esempio tramite proxy).
- Non sono quasi mai usati filtri per UDP
- Il livello di sicurezza inficia sulle prestazioni del sistema

QoS - Quality of Service

Multimedia networking applications

Gli aspetti qualitativi delle applicazioni multimediali **audio** si fondano essenzialmente su:

- campionamento al secondo del segnale analogico (normalmente 2f, f=20KHz)
- valore della quantizzazione del campionamento digitale (il valore digitale che può assumere in funzione di range di hertz).

Gli aspetti qualitativi delle applicazioni multimediali **video** invece sono:

- Sequenza di fotogrammi al secondo mostrati (fps)
- Tipo di codifica e compressione del video.

Tipi di applicazioni multimediali

Streaming stored

È il tipo di contenuto che viene visualizzato senza la necessità di averlo scaricato localmente ed è già esistente nei server di origine. Dal momento che la trasmissione del contenuto ha un ritardo variabile (jitter) l'obiettivo è quello di renderlo costante per il ricevitore, ad esempio tramite una coda di buffer che salva il contenuto in eccesso che arriva per poter garantire la fluidità. Dal momento che viene usato il protocollo UDP per inviare i dati e che, per definizione, non segue delle strade fissate per il trasporto, è necessario associare ai dati un **numero di sequenza** per garantire l'ordine di arrivo al ricevente e il **timestamp** per verificare che il timing sia coerente con la riproduzione del flusso. L'uso di HTTP/TCP per il trasferimento dei contenuti è ovviamente più lento (per la questione

Conversational - VoIP - RTP - SIP

La caratteristica più importante di cui ha bisogno il servizio **Voice-over-IP** per funzionare efficacemente è la latenza bassa (0-150-400ms al massimo). Per questo fra le varie tecniche adoperate vi sono:

- variazione dei dati trasmessi in base ai periodi di silenzio (64kbps durante il parlare ogni 20msec, e segmenti predefiniti ogni160ms per i periodi di silenzio).
- tolleranza di perdita di dati fra 1-10% con possibilità di ricalcolare il tempo di trasmissione dei successivi datagrammi (*playout delay*) e recuperare i pacchetti persi tramite logica ridondanza nell'invio dei pacchetti (*Forward Error Correction*: per ogni gruppo di n chunks è possibile recuperarli tramite XOR così da poter

mandare n+1 chunks e aumentare la banda) o tramite *riorganizzazione casuale* dell'invio dei pacchetti così da non perdere interi blocchi ma solo parti piccole di una frame (aumenta il ritardo di elaborazione).

Il **Real Time Protocol** è un protocollo end-to-end incapsulato dentro UDP che specifica la struttura dei paccchetti che trasportano dati audio e video definendone payload type, numero di sequenza e timestamp.
RTP non implementa alcun meccanismo per garantire il QoS ma fornisce solo gli strumenti per gestire sequenze e timestamps. Insieme ad esso si usa anche RTCP per analizzare e studiare il canale di trasmissione che per non inficiare sull'efficienza se ne riduce l'uso con l'aumentare dei partecipanti alla trasmissione. Si usa invece RTSP per stabilire e controllare il flusso dei dati trasmessi sul web.

Il **Session Initiation Protocol** è un protocollo predisposto per gestire le telefonate e le video conferenze su Internet. Implementa di fatto i meccanismi per inizializzare, gestire e concludere una chiamata, basati sul concetto di associare l'identità delle persone ai loro inidirizzi IP. Un'alternativa a SIP è il protocollo H.323 che gestisce sia il setup sia il trasposto.

Supporti di rete per il multimediale

Fino ad ora abbiamo visto tutte le tecniche e i protocolli utilizzati senza modificare gli apparati di rete. Ma è possibile andare ad operare direttamente su di essi per gestire al meglio il QoS.

Normalmente tutto il traffico di rete viene trattato **in egual modo** con il metodo Best Effort, ma può essere **gestito per "classi"** implementando meccanismi di tracciamento di pacchetti, scheduling e policing, impiegando una complessità media, o può essere **gestito per singola connessione** e implementare anche una politica di ammissione del flusso stesso, impiegando una complessità alta.

Traffico gestito per classi di servizio

Alla base di questo approccio vi è la catalogazione dei pacchetti in base al tipo mediante l'inserimento in testa di un codice identificativo del tipo (**Type of Service** bits) da parte degli End Systems o di Router adattati per lo scopo. Per la gestione della banda utilizzata invece si inseriscono opportune **market policy** agli estremi della rete ma con **allocazione dinamica** in modo da non sprecare spazi e avere il canale opportunamente occupato.

Tra le varie tecniche di scheduling per tipo di servizio abbiamo: *FIFO*, scheduling *prioritario*, *Round Robin* (manda i pacchetti solo se completi se disponibili), *Weighted Fair Queuing* (l'unione di scheduling prioritario e RR).

I meccanismi di policing si basano su 3 criteri: il *tasso medio* di pacchetti trasmessi per unità di tempo nel lungo periodo, il *tasso di picco,* il *massimo numero di burst* (pacchetti trasmessi consecutivamente). Si realizzano con i *token bucket* che si legano ai pacchetti trasmessi: la quantità di associazioni pacchetto-token sono determinati appunto secondo i criteri precedenti così da garantire la coerenza del traffico con le specifiche richieste.

Traffico gestito per singola connessione (DiffServ)

L'architettura Differentiated Services si basa sulla presenza di due tipi di router nella rete (in Autonomus System):

- **edge router:** assegnano profili pre negoziati di banda e di tasso di pacchetti trasferibile e si occupa di marcare i pacchetti per ogni flusso profilato con il metodo Per-hop Behavior (Per ogni hop router viene definito il marcatore e il profilo per determinare sempre la differenza di classe a cui appartiene). Per ogni profilo viene assegnata una classe che può essere: **expedited forwarding** se il tasso di partenza di un pacchetto è uguale o superiore ad una certa soglia o **assured forwarding** (suddiviso a sua volta in 4 classi) per i pacchetti sotto la soglia.
- **core router:** gestiscono il traffico per classi con scheduling basato sul marketing agli edge

Il problema principale di questo servizio è il fatto di disporre di risorse limitate da assegnare a tutti. Si introduce quindi una funzione di **call admission** che gestisce l approvazione dei flussi.

Peer 2 Peer

L'architettura Client-Server può essere realizzata in 2 modi diversi:

Centralizzata se la gestione dell'infrastruttura ha un unica governance e puo è essere a **singolo server** o **distribuito,** in particolare con quest'ultimo si hanno vantaggi in merito al bilanciamento del carico e ad evitare problemi Single-Point-of-Failure.

Decentralizzata se vi sono più entità che cooperano nella gestione dei **server** o, nel caso non vi sia distinzione fra client e server si definisce una rete **Peer 2 Peer.**

Peer

È l'entità che compone una rete P2P. Ogni Peer comunica direttamente con gli altri condividendo risorse senza limiti. Tra i vantaggi del P2P vi sono: distribuzione del potere computazionale per svolgere compiti, anonimità e privacy.

P2P file sharing

Una delle applicazioni più importanti e diffuse del p2p è il file sharing, in particolare esistono diversi schemi architetturali come: *Centralizzato, Decentralizzato, Gerarchico (ibrido), Incentivato, Strutturato.*

La **ricerca di un peer** si basa sulla presenza di *super nodi* in grado di sapere quali sono i peer attualmente connessi per la condivisione di una risorsa (centralizzato)

L'**accesso ad una risorsa** differisce dalle applicazioni C/S dove è possibile trovarle facilmente dal momento che è nota a priori l'identità dei server che la contengono. Nei sistemi p2p è necessario sempre passare per una fase di ricerca basata su una rete al di sopra di quella fisica (Overlay Network). In particolare osserviamo per ogni schema:

- Centralizzato (modello Napster): Un **server** memorizza sia i peer attivi, sia le risorse che possono condividere. PRO: efficiente CONTRO: single point of failure, non scalabile.
- Decentralizzato (modello Gnutella): la rete viene esplorata attraverso **query Flooding** (condividere con i vicini le proprie informazioni tipo passa parola) PRO: no point of failure, CONTRO: non efficiente. L'efficienza può migliorare introducendo un TTL (Time to Leave) o un numero massimo di nodi esplorabili per la ricerca.
- Gerarchico (modello Kazaa): è un evoluzione del modello decentralizzato che presenta dei **super nodi** (nodi che hanno banda e uptime più elevato) ai quali gli altri nodi si collegano e per conto dei quali vengono fatte le query serach.
- Incentivante (BitTorrent): introduce dei file statici metainfo, **torrent files,** che contiene vari informazioni come tracker del peer che lo sta offrendo e i peers che lo stanno attualmente scaricando e che contribuiscono di conseguenza a condividerlo. Questo gruppo di peers si definisce **swarm** ed è composto da peers che possiedono il file interamente (**seed)** e quelli che ancora non lo hanno interamente (**leechers).** PRO: incentivi, ricerca centralizzata. CONTRO: ricerca centralizzata.
- Strutturato (modelli Chord,CAN): Mentre il sistema C/S è server centrico, questo schema è content-centrico perchè ha come obiettivo quello di trovare ad ogni

costo la risorsa cercata. Gli indirizzi che caratterizzano i nodi riguardano anche il contenuto cercato (hash function dei file).
Nel modello **CAN** si associa ad ogni nodo e oggetto un id in uno spazio d-dimensionale diviso in sottospazi ogni volta che viene inserito un nuovo nodo e ogni file inserito in altre coordinate viene associato al nodo più vicino dello stesso spazio dimensionale in cui si trova.
Nel modello **Chord** si associano le coppie nodi oggetti su un anello con il criterio che ciascun nodo gestisce i dati che si trovano fra il nodo stesso e il nodo precedente.

Blockchain

La Blockchain è un libro mastro decentralizzato che registra le transazioni in un modo immutabile (hash function), verificabile (firma digitale) e permanente (link).

Alla base della realizzazione delle transazioni c'è il consenso. Si parla di transazioni *permissionless* quando chiunque può partecipare al consenso per l'approvazione della transazione, si parla di *permissioned* quando l'operazione è limitata ad un sottoinsieme di entità. La visibilità delle transazioni scritte può essere pubblica o privata.

Il concetto chiave del consenso è l'ordine degli eventi su cui si basa l'unicità delle transazioni. Vi sono svariati algoritmi che si occupano di gestire il consenso, tra i più importanti vi è il **Proof-of-Work.**

Trilemma della blockchain: Alla base della Blockchain ci sono 3 concetti chiave: Decentralizzazione, sicurezza e scalabilità. Non è mai possibile garantire tutti e tre i criteri contemporaneamente.

Transazioni

Definiamo una moneta elettronica come una catena di firme digitali. Ogni proprietario trasferisce la moneta al prossimo firmando digitalmente un hash della transazione precedente e la chiave pubblica del prossimo proprietario della moneta e infine aggiunge queste informazioni alla fine della moneta.

Le transazioni hanno dunque un *mittente* e un *destinatario.* L'integrità viene garantita con la **firma digitale (chiave pubblica)** senza necessità di Autorità di certificazione dal momento che non è normalmente necessario dimostrare l'identità delle entità ma solamente assicurare che non possa essere contraffatta la transazione. Questo implica pseudo-anonimia visto che ognuno può generare quante chiavi pubbliche

vuole, ma è sempre e comunque possibile tracciare e cercare di risalire all'identità reale della persona.

L'immutabilità e la permanenza dello storico delle transazioni si basa proprio sulla catena di blocchi di dati di transazioni che si genera secondo la definizione stessa di transazione. Dal momento che quindi ogni blocco registra sempre un link al blocco precedente, nel caso in cui uno di questi blocchi venga manipolato, si rompe tutta la catena di blocchi. La struttura dati che si viene a creare si definisce **merkle Tree** che non è altro che un grafo binario che può essere esplorato in tempo o(log(n)) garantendo così anche efficienza.

Consenso

Il problema del raggiungimento del consenso è esemplificabile nella teoria del *problema dei generali bizantini* che afferma sia necessario avere oltre il 2/3 dell'approvazione di una scelta per raggiungere il consenso. Nella blockchain abbiamo 3 figure chiave: i nodi (generali), un leader che propone proposte di blocchi (comandante), i followers (luogotenenti). Per prima cosa bisogna indicare il modo per eleggere un leader: in modo randomico, o con risoluzione di un puzzle (PoW).

Il **Proof-of-work** consiste essenzialmente nella risoluzione di un codice Nonce tramite forza bruta che, associato al blocco e all'informazione che esso contiene, restituisce un hash composto da un numero predefinito di "0" iniziali. Questo richiede dunque l'utilizzo di potenza computazionale per avere la possibilità di poter proporre a tutti gli altri il blocco "minato" come nuovo blocco della catena. Spesso le storie delle catene di blocchi si forkano, la scelta sulla catena singola da seguire ricade sempre sulla sottocatena forkata più lunga. Dal momento che PoW si basa su potenza computazionale, il suo principale CONTRO è la sostenibilità ambientale in termini di energia consumata e anche una perdita di concetto di decentralizzazione.

Smart Contract

È un programma eseguito in parallelo su una molteplicità di nodi della blockchain in modo asincrono che danno come risultato un cambio di stato della blockchain stessa per garantire l'attuazione del consenso per l'introduzione di un blocco.

Protocolli IoT

alla base della realizzazione di un progetto applicativo IoT vi sono 4 fattori da dover tenere in considerazione per farlo al meglio:

Costo della costruzione, durata della batteria, il raggio di copertura entro cui deve poter raggiungere un gateway, la banda massima che può raggiungere.

L’omonimo protocollo HTTP per IoT è **COAP (COnstrained Application Protocol),** un protocollo restful basato su UDP, facilmente mappabile da e verso HTTP. Il modello è asincrono, usa gli stessi metodi di CRUD (get post put delete), il funzionamento si basa sull’inclusione di identificativi all’interno dei messaggi con risposta opportuni ACK contenti quegli ID al fine di aggirare l’assenza di connessione garantita da TCP.

MQTT (Message Queue Telemetry Transport) è l’alternativa ad AMQP per l’IoT. Si tratta di un protocollo di messaggistica più leggero, con librerie piccole ed affidabile. Gli attori che vi troviamo sono Publisher, Subscriber, Broker che smistano messaggi costituiti oltre dal contenuto anche da topic di interesse per i subscriber