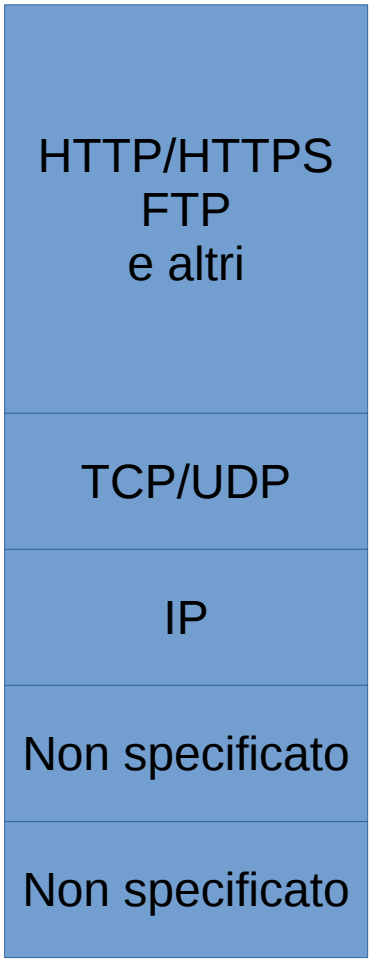
# Modello ISO/OSI

Modello ISO/OSI


Suite protocolli internet (detta TCP/IP)
in relazione con modello ISO/OSI



# Modello ISO/OSI

- Tra ogni livello è prevista un’interfaccia per passare dati dal livello sovrastante al livello sottostante e viceversa (es: il software che si occupa di gestire pacchetti ethernet scrive in una memoria interna all’adattatore di rete la frame ethernet da inviare)
- Protocolli di livello inferiore encapsulano dati provenienti da protocolli di livello superiore aggiungendo header e/o tail
  (es: ethernet che contiene IP che a sua volta contiene UDP)

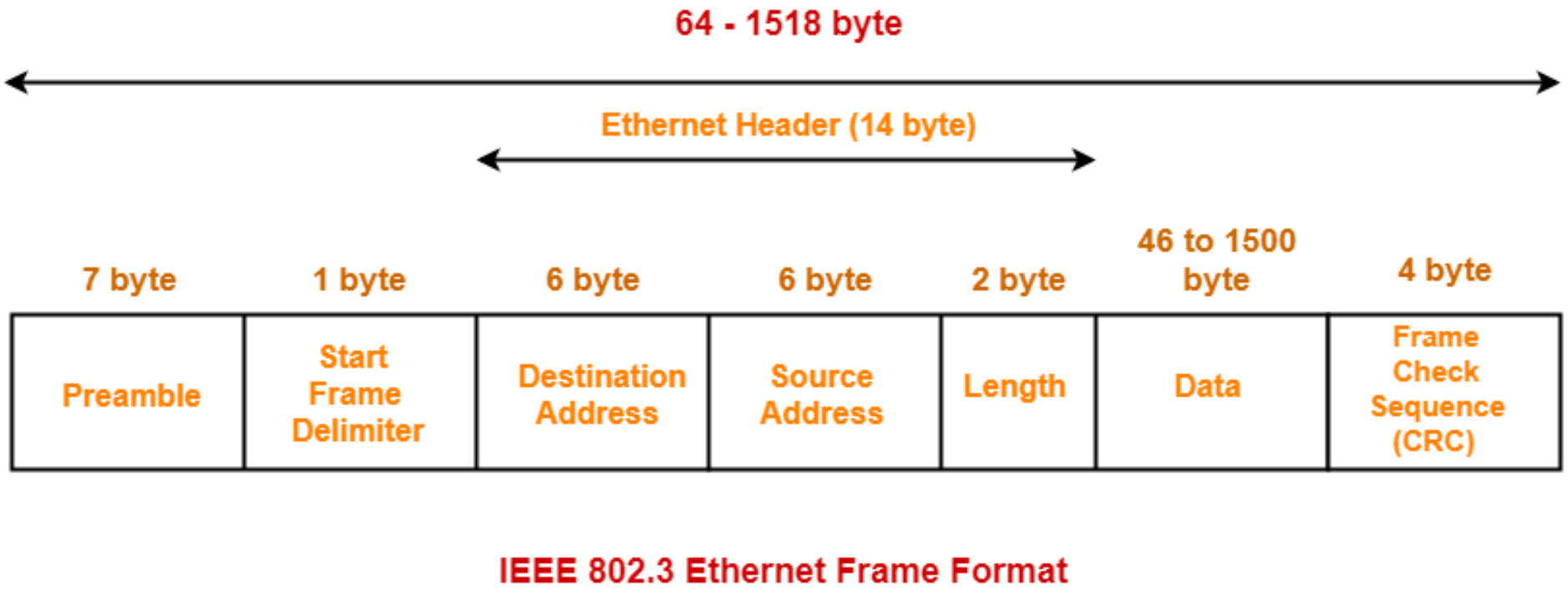
# Livello Fisico

- Scambia dati bit a bit.
- Generalmente non controlla errori di trasmissione
- Si occupa di tradurre in segnali fisici (tensione, onde radio, onde elettromagnetiche…) i dati che riceve da protocollo di livello sovrastante (data link)

# Livello Data Link

- Protocolli di questo livello ragionano a frame (sequenze finite di bit)
- Si occupano di mettere in comunicazione due interfacce di rete situate sulla medesima rete logica (es: collegate tramite ethernet)
- Protocolli di livello data link si occupano di riconoscere inizio e fine frame e di segnalarli al ricevente (mediante l'uso di sequenze di bit aggiuntive all'inizio e alla fine, rispettivamente preambolo e postambolo)
- Controllano presenza di errori mediante codice CRC. In caso di errori chiedono ritrasmissione del pacchetto
- Usano MAC address (a 48 bit) per identificare le interfacce
- Gli switch e i bridge lavorano a questo livello

# Esempio frame ethernet




IEEE 802.3 Ethernet Frame Format

# Livello Network (IP)

- Si occupa di collegare macchine appartenenti a reti distinte
- Usa indirizzi IP (es: IPv4 a 32 bit o IPv6 a 128 bit) per identificare una macchina in una rete
- I router lavorano a questo livello mantenendo una tabella di routing che mappa un indirizzo IP a un interfaccia su cui inviare un pacchetto rivolto a quell'indirizzo IP
- IP utilizza ARP (Address Resolution Protocol) per ottenere un indirizzo MAC dato un indirizzo IP in modo tale da sapere a chi inviare le frame di livello Data Link una volta consultata la routing table

# Formato IP CIDR (Classless Inter-Domain routing)

- Ogni indirizzo IP è formato da x bit che identificano il numero della rete
- Un indirizzo è espresso con la seguente notazione: aaa.aaa.aaa.aaa/x
- Es: 192.168.1.8/26 indica un indirizzo in cui i primi 26 bit costituiscono il numero della rete.
- Ogni rete ha quindi una maschera di sotto rete (subnet mask) corrispondente: la rete a cui appartiene l'indirizzo IP qui sopra ha la maschera 255.255.255.192

# Tabella di routing

192.168.0.0/16 via eth0

192.168.2.0/24 via eth1

default via 192.168.0.1

- Ogni record della tabella mantiene per ogni rete conosciuta il nome dell’interfaccia su cui spedire un pacchetto rivolto a quella rete
- se non si sa a chi spedire il pacchetto, verrà spedito a default (normalmente un altro router) che si occuperà di gestire l’instradamento

# Operazione svolte da router

1) Riceve pacchetto rivolto a 192.168.2.9
2) Prende ogni record nella tabella di routing ed effettua AND bit a bit tra maschera di sotto rete del record e indirizzo IP destinazione del pacchetto
3) Confronta il risultato dell' AND con l'indirizzo della rete presente in routing table
4) Terminati tutti i confronti scegli la rete che ha fatto match che ha la maschera più lunga se esite
5) Se non esiste controlla se esiste default ed invia a default
6) Se non esiste default non fare nulla, il pacchetto viene perso

# Operazione svolte da router

192.168.0.0/16 via eth0

192.168.2.0/24 via eth1

default via 192.168.0.1

1) Arriva pacchetto per 192.168.2.8
2) prendi prima subnet 255.255.0.0 e fai AND con 192.168.2.8 ottieni 192.168.0.0 che fa match con 192.168.0.0
3) prendi seconda subnet 255.255.255.0 e fai AND con 192.168.2.8 e ottieni 192.168.2.0 che fa match
4) routing table terminata, 192.168.2.0 viene scelto perché ha subnet più lunga, pacchetto viene instradato su eth1

# Livello Transport

- Il livello transport si occupa della comunicazione tra due processi
- Una volta che il pacchetto ha raggiunto la macchina destinazione (sia essa virtuale o fisica) grazie al protocollo IP, è necessario recapitarlo al processo corretto. I protocolli transport identificano un processo
- Fanno parte di questi protocolli TCP e UDP

# TCP

- TCP è un protocollo connection-oriented e cioè ad ogni comunicazione necessita l'apertura, il mantenimento e la chiusura di una connessione.
- In TCP ogni pacchetto viene inviato e attende un pacchetto di acknowledgment (ACK). Se l'ACK di un pacchetto non viene ricevuto entro un certo periodo, il pacchetto viene ritrasmesso. Questo garantisce l'arrivo dei pacchetti a destinazione

# UDP

- UDP è un protocollo connectionless e quindi non mantiene alcuna connessione
- Ogni pacchetto viene spedito indipendentemente dagli altri.
- UDP non garantisce l’arrivo dei pacchetti a destinazione
- Molto utile per applicazioni per le quali possiamo permetterci di perdere alcuni pacchetti (VoIP, VoD ecc…). Inutilizzabile per applicazioni che richiedono l’arrivo completo del messaggio (es: applicazioni basate su HTTP)

# HTTP

- HTTP è un protocollo connectionless di livello 5 e superiore
- Protocollo di tipo request-reply: un client effettua una richiesta e un server risponde con una risposta. Ogni richiesta-risposta è indipendente dalle altre (ecco perché HTTP è connectionless)
- HTTP definisce diversi metodi:
  - GET
  - POST
  - PUT
  - DELETE
  - OPTIONS

# HTTP

- Tutte le risposte HTTP hanno uno status code che ci da indicazioni sul tipo di risposta. Generalmente si ha che:
  - 200 <= status < 300: risposta positiva, tutto andato a buon fine
  - 300 <= status < 400: risorsa non più presente a quell' indirizzo, ma spostata su nuovo indirizzo
  - 400 <= status < 500: risorsa non trovata
  - status >= 500: errore interno del server

# HTTP

- HTTP definisce moltissimi header. I più importanti sono:
  - Content-Type: ci dice il MIME type della richiesta/risposta
  - Accept: permette di specificare al client quali MIME type si accettano per la risposta
  - cookie: contiene tutti i cookie inviati dal client al server
  - set-cookie: permette al server di impostare un cookie su client
  - location: permette al server di specificare un redirect al client

# HTTPS

- HTTPS aggiunge ad HTTP la cifratura e l’identificazione dei domini per permettere:
  - Integrità dei dati: nessuno può intercettare i pacchetti e manometterli
  - Confidenzialità dei dati: nessuno può intercettare i pacchetti e leggerli
  - Impedire a siti terzi di spacciarsi per altri siti

# HTTPS

- HTTPS usa una firma digitale per fornire un certificato che certifichi che un determinato sito sia effettivamente chi dice di essere
- La firma digitale viene creata utilizzando un algoritmo a chiave asimmetrica:
  - Una Certificate Authority (CA) è in grado di firmare un certificato utilizzando la sua chiave privata
  - Ogni client può verificare la firma utilizzando una chiave pubblica

# HTTPS

- HTTPS utilizza un algoritmo a chiave asimmetrica (generalmente TLS) per effettuare lo scambio della chiave simmetrica che verrà utilizzata sia da client che da server per decriptare e criptare i messaggi, garantendo la confidenzialità dei dati

# Passi logici di HTTPS

1) al momento dell'acquisto di un certificato per un dominio *xyz.com* si genera una chiave privata e una Certificate Signing Request (CSR). La CSR contiene oltre che al nome di dominio xyz.com anche la chiave pubblica generata a partire dalla chiave privata
2) Si sceglie una Certificate Authority e si invia alla CA il CSR
3) La CA richiede di verificare che siamo i veri proprietari di xyz.com generalmente tramite due modalità: invio mail a una mail @xyz.com o aggiunta di una stringa randomica come record TXT associato al dns di xyz.com

# Passi logici di HTTPS

1) Una volta verificato il dominio, la CA consegna il certificato
2) Un client inizia una conessione HTTPS con un HeloClient
3) Il server risponde con HeloServer contenente certificato
4) Il client verifica se tra le CA di cui si fida c'è quella che ha firmato il certificato presentato da xyz.com
5) Se c'è il client verifica che l'url nel certificato sia uguale all'url navigato
6) Se sono uguali il client genera una chiave e la cripta utilizzando la chiave pubblica presente nel certificato
7) Il server, unico possessore della chiave privata, decripta il messaggio e ottiene la chiave generata da client.
8) A questo punto tutti i messaggi successivi (richiesta e risposta) vengono criptati e decriptati usando questa chiave