Domande RETI

Descrivere il protocollo TLS (nella versione 1.2). Nella descrizione, seguire la traccia data dalle domande:

- **Quali sono vantaggi e svantaggi di applicare la crittografia ai vari layer dell'IP stack?**
- **Quali sono le proprietà dell'handshake protocol?**
- **A cosa servono i pacchetti ChangeCipherSpec e Finished?**

Transport Layer Security è un protocollo che serve ad introdurre servizi di sicurezza nello stack TCP/IP, che inizialmente non ne prevedeva alcuno. Ne esistono diverse versioni, la più supportata è la 1.2, l'ultima e più sicura è la 1.3. TLS utilizza crittografia a chiave simmetrica per cifrare e autenticare i dati, e crittografia a chiave pubblica per inizare una sessione e generare le chiavi simmetriche di sessione. Si appoggia al sistema dei certificati digitali per garantire l'autenticazione dell'origine dei dati. TLS si può applicare a molti contesti, ma quello più comune è verificare che un sito che viene navigato con HTTP sia effettivamente quello che corrisponde al dominio richiesto.

La scelta fatta con TLS è quella di inserire la crittografia sopra lo strato di trasporto (TCP). Questa scelta è stata considerata il miglior compromesso tra applicabilità e sicurezza, ma non è l'unica possibile. A livello datalink infatti alcuni standard di comunicazione introducono servizi di sicurezza (ad esempio, Wi-Fi) ma questo non è un servizio end-to-end, ma solo usabile nella rete locale. D'altra parte, cifrare tutto il pacchetto IP incluso il suo header non permetterebbe ai router di instradare i pacchetti, e può quindi essere fatto solo su reti private. In fine, cifrare tutto l'header TCP renderebbe impossibile usare tutte le middlebox presenti su Internet (ad esempio, i NAT). L'applicazione di TLS sopra a TCP ha permesso la sua rapida diffusione ed oggi, la stragrande maggioranza del traffico HTTP è cifrato.

L'handshake protocol serve a utilizzare una qualche forma di negoziazione delle credenziali per generare chiavi di sessione, e dipende normalmente dall'utilizzo di certificati validi. L'handshake è una fase molto critica perchè avviene prima della generazione delle credenziali, e quindi in una fase in cui un'attaccante potrebbe modificare i messaggi lungo il loro cammino (Eve, che sarà un MiTM). Deve quindi garantire che:

1) che l'identità di almeno uno dei peer (Alice e Bob) sia verificabile attraverso credenziali. Nell'applicazione in HTTP Alice sarà responsabile del dominio alice.com e dovrà possedere un certificato rilasciato da una CA conosciuta per "alice.com".
2) la chiave simmetrica negoziata che verrà utilizzata con algoritmi a chiave simmetrica non sia rivelata ad Eve
3) che la negoziazione termini senza che Eve possa interferire con essa, ovvero che sia garantita integrità dei dati scambiati durante l'handshake

L'handshake per ottenere questi risultati introduce però due scambi di pacchetti in più (2 RTT) rispetto ad una connessione TCP senza TLS (nel caso di TLS 1.3, questo viene ridotto ad un solo RTT).

ChangeCipherSpec è il messaggio che indica che la negoziazione è terminata e che dal prossimo messaggio si cominceranno ad utilizzare le chiavi negoziate, e che quindi il prossimo messaggio sarà il primo ad essere cifrato ed autenticato (normalmente con AES, e con un HMAC basato su una funzione hash sicura). Il messaggio successivo è il messaggio Finish, che contiene un hash di tutti i dati scambiati fino a quel momento. Essendo autenticato, permette di verificare che tutti i messaggi dell'handshake (che per definizione non possono essere autenticati) non siano stati modificati durante il loro cammino.

Descrivere le chiamate più rilevanti che sono necessarie per aprire un TCP socket nello standard POSIX (BSD socket). Non è necessario enumerare in dettaglio tutti I parametri delle call, ma è necessario descriverne il funzionamento generale. Usare la seguente traccia:

- **descrivere le chiamate più importanti per un server**
- **le differenze con un client**

Un BSD socket è un'interfaccia standard per l'apertura di un socket, che può essere UDP o TCP, ci concentreremo sul caso di TCP. Un server che voglia aprire una socket dovrà passare attraverso le seguenti chiamate:

1) socket(): questa chiamata istanzia il socket stesso e prende come parametri principali il tipo di socket (TCP/UDP)

2) bind(): questa chiamata prende in input il socket e i dati su cui associarlo, ovvero una porta ed eventualmente un indirizzo IP del server. La porta dovrà essere libera da altri socket aperti.

3) listen(): questa chiamata instanzia effettivamente il server che a quel punto è pronto a ricevere connessioni in ingresso.

4) accept(): questa chiamata è bloccante, e viene interrotta quando effettivamente un client si collega al socket

5) una volta che il socket è aperto e la connessione TCP è in corso, i dati verranno letti e inviati nella socket con delle read() e write(), o altre chiamate analoghe.

6) close() per terminare il socket.

La differenza fondamentale tra un server ed un client è che il client non farà la chiamata bind() nè la listen() ma farà direttamente una connect(), che prende come parametri l'indirizzo e la porta del server a cui collegarsi.

Come funziona il MAC di 802.11, nel caso del problema del nodo nascosto ("hidden node"). Nella descrizione, seguire la traccia data dalle domande:

- **quale problema introduce il nodo nascosto?**
- **cosa cambia nel DCF nel caso del nodo nascosto?**
- **quali sono i vantaggi e gli svantaggi di questi cambiamenti e come si mitigano gli svantaggi?**

Per descrivere il problema del nodo nascosto bisogna accennare il funzionamento di DCF in una situazione generale. DCF è un protcollo di accesso (livello datalink) usato per reti Wi-Fi (IEEE 802.11) che si basa su Carrier-Sensing, Multiple Access/Collision Avoidance (CSMA/CA) per evitare collisioni, ovvero la situazione in cui un ricevitore riceve contemporaneamente due frame, che vengono entrambi scartati. In CSMA/CA un terminale A prima di inviare un frame al terminale B ascolta sul canale per identificare se sia o meno occupato (Carrier-Sensing), dopodichè aspetta un tempo predeterminato (timer DIFS) e se il canale rimane libero, invia il frame. Dopo aver trasmesso, attende la ricezione di un frame di ACK da B per confermare l'avvenuta corretta ricezione. Se il terminale A comincia a ricevere un frame generato da un terzo terminale C prima della fine di DIFS, va in uno stato di back-off, per un tempo pari alla durate del campo DURATION del frame, più un tempo casuale. In ciascun frame è presente il campo DURATION, che contiene un numero che esprime il tempo necessario per completare l'invio del frame e per ricevere l'ACK di risposta. Il terminale A quindi, legge il campo duration e sa per quanto tempo il canale sarà occupato da C.

L'assunzione che sta alla base di questo MAC è che nel momento in cui C invia il frame, A sia in grado di ricevere questo frame, anche nel caso non sia destinato ad A stesso. Se questa condizione non avviene, allora A non si accorgerà che è in corso una trasmissione, e non andrà nello stato di back-off, generando una collisione. È la situazione tipica in cui A e C non possono comunicare tra di loro (ad esempio, c'è una spessa parete tra i due) ma entrambi possono comunicare con l'access point (AP) della rete (nel nostro esempio, potrebbe essere il terminale B). Quando questo avviene, B continuerà a ricevere frame sovrapposti, subendo collisioni continue.

CSMA/CA affronta questo problema introducendo due frame di controllo RTS (request to send) e CTS (clear to send). Prima di inviare un frame A invierà un frame RTS verso l'AP. RTS è un frame molto breve ma che contiene il campo DURATION relativo al frame di dati che A vuole inviare a B. Se la ricezione da parte di B è corretta (non c'è una collisione) allora B risponderà con un CTS che viene inviato in broadcast e contiene lo stesso DURATION del RTS. Anche se A e C non possono comunicare tra loro, entrambi devono poter ricevere frame dall'AP, quindi C riceverà il CTS, che specifica il tempo nel quale C non può trasmettere.

RTS/CTS riesce a mitigare il problema del nodo nascosto, ma riduce l'efficienza del MAC, perchè introduce un nuovo scambio di 2 frame prima che A possa iniziare a trasmettere il suo frame di dati. Inoltre questo meccanismo non garantisce l'assenza di collisioni, perchè l'RTS stesso può collidere con un altro RTS. Se A deve inviare un frame molto piccolo (ad esempio un frame che trasporta un ACK TCP) la probabilità di collisione potrebbe essere molto simile a quella di un frame RTS, quindi potrebbe essere più conveniente inviare il frame senza RTS/CTS ed attendere l'ACK. Se l'ACK non arriva, si ripeterà l'invio. RTS/CTS vengono quindi utilizzati in due casi: quando A si accorge di collisioni ripetute (mancanza di ricezione di ACK) pure se il canale è libero per la maggior parte del tempo, ma soprattutto, solo se il frame da inviare supera una soglia di attivazione degli RTS/CTS, che non è specificata nello standard ma è definita dai singoli driver delle schede wireless.

Descrivere il funzionamento di DNS. Nella descrizione, seguire la traccia data dalle domande: come sono organizzati i nomi di dominio? Come avviene una risoluzione di un nome? Quali sono i vantaggi di usare un server DNS locale?

Un dominio è un nome associato ad un indirizzo IP su Internet. Si usano perchè per gli umani è più comodo ricordare nomi piuttosto che indirizzi IP, una volta effettuata la traduzione dal primo al secondo, il protocollo IP non usa mai nomi di dominio, ma solo indirizzi. È necessario quindi un protocollo che permetta ad un client che vuole ottenere una risorsa (ad esempio identificata da un URL, tipo www.alice.com) di convertire questo nome in un indirizzo IP, questo protocollo è il Domain Name System (DNS).

Un dominio è una sequenza di parole separate da punti, che costituiscono un'architettura gerarchica ad albero, con gerarchia che aumenta da sinistra a destra. L'ultimo elemento (a destra) di un dominio è quello che si chiama un top level domain (TLD), nell'esempio precedente ".com". L'architettura è gerarchica ma decentralizzata, con server DNS che sono responsabili per intere "zone", ovvero tutti i sottomini sotto un certo

livello della gerarchia. Alcuni server DNS sono responsabili per il dominio "root", e vengono interrogati per conoscere i server DNS di livello subito inferiore. I root server sono gesisti da un'entità unica (ICANN) che però delega ad altre entità (i "registrars") la registrazione di nuovi domini sotto dei TLD prestabiliti. In pratica, se Alice vuole registrare www.alice.com dovrà chiedere ad un registrar di affidarle la zona *.alice.com, che lei gestirà con un suo server DNS. A quel punto il suo server sarà "authoritative" per quella zona, e Alice potrà aggiungere vari sottodomini (ad es. www.alice.com, mail.alice.com ecc...).

La risoluzione di un dominio avviene in modo ricorsivo. Ciascuna rete ha un suo server DNS locale che i terminali usano per risolvere i nomi. Normalmente (in IPv4) i terminali apprendono quale sia l'indirizzo IP del server DNS locale attraverso il protocollo DHCP all'ingresso del terminale nella rete. Il terminale che vuole risolvere www.alice.com chiederà al DNS locale di fare una risoluzione ricorsiva, ed il DNS locale inizierà la ricorsione per conto del terminale. Chiederà prima ad uno dei root server quale server è "authoritative" per ".com", e poi si rivolgerà a quel server per sapere quale server è authoritative per "alice.com". In fine interrogherà il server DNS di Alice che farà la risoluzione di www.alice.com. Il DNS locale salverà in una propria cache questa informazione, e l'informazione di chi è authoritative per "alice.com".

Il vantaggio di un sistema come quello descritto è che se un secondo terminale nella stessa rete deve fare la stessa risoluzione, oppure una risoluzione per la stessa zona (ad es. mail.alice.com) il server DNS della rete locale può rispondere con i dati presenti in cache. Nel primo caso risponderà con l'IP corretto, nel secondo caso farà la risoluzione di "mail.alice.com" ma chiedendo direttamente al server DNS authoritative per "alice.com", e quindi saltando dei passaggi che fanno risparmiare tempo.

Spiegare il funzionamento congiunto di SPF e DKIM. Nella descrizione seguire la traccia data dalle seguenti domande:

- **Quale problema risolve SPF? riportare un esempio di record TXT che usa la sintassi di SPF**
- **Quale problema risolve DKIM?**
- **Perché c'é bisogno di entrambi i protocolli?**

Ad alto livello, SPF e DKIM sono due protocolli che vengono utilizzati per impedire agli spammer di inviare messaggi e-mail fingendo di essere qualcun'altro. SPF (Sender Policy Framework) risolve il problema dell'autenticazione dell'indirizzo IP di una connessione SMTP che tenta di inviare un e-mail con un determinato campo "MAIL From". Quando il server A riceve una connessione SMTP dal server B, che usa nel "MAIL From" il dominio example.com fa una risoluzione del dominio attraverso record DNS TXT. Il proprietario di example.com aggiunge tra i record uno nel formato che segue:

v=spf1 ip4: 1.2.3.4 1 ip4: 4.3.2.0/24 include:spf.google.com -all

In questo esempio: - ip4:1.2.3.4 autorizza un IP specifico ad aprire connessioni. - ip4: 4.3.2.1/24 autorizza tutta la subnet ad aprire connessioni - include:spf.google.com include anche le regole impostate dal dominio spf.google.com. - -all vieta qualsiasi altro IP. DKIM (DomainKeys Identified Mail) autentica l'integrità dei contenuti delle e-mail (come intestazioni e corpo del messaggio) firmandoli digitalmente. Risolve il problema della modifica fraudolenta del messaggio durante il transito, o della creazione di un email falsa, pur proveniente da un IP verificato con SPF (nota, SPF potrebbe non imporre una regola usando "-all"). L'header DKIM contiene il dominio di origine (che normalmente corrisponde al campo "From" della email), ed il server inviante firma alcuni campi selezionati utilizzando una chiave privata. Il server ricevente verifica la firma utilizzando la chiave pubblica ottenuta con una risoluzione DNS per i record TXT nel DNS del dominio mittente. I due protocolli svolgono ruoli diversi: SPF garantisce che il server SMTP che invia l'e-mail sia autorizzato da una policy pubblicata dagli amministratori dal dominio, ma non protegge i campi visibili al destinatario, come From. DKIM, invece, verifica che i contenuti della mail non siano stati alterati, ma da solo non impedisce a un mittente fraudolento di inviare e-mail non autorizzate usando un "MAIL From:" arbitrario. L'uso congiunto di SPF e DKIM crea una difesa più completa: SPF autentica il server di origine, DKIM protegge l'integrità dei messaggi. Inoltre, insieme a DMARC, consentono al dominio mittente di stabilire politiche chiare su come trattare e-mail non conformi, in modo che il destinatario agisca con una policy prestabilita e non arbitraria.

Riportare le equazioni di Shannon e di Nyquist, spiegare il loro utilizzo e la loro differenza. Seguire la seguente traccia:

- **Introdurre l'equazione di Shannon, spiegare i suoi termini ed il suo significato. Specificare le unitá di misura dei termini e del risultato.**
- **Introdurre l'equazione di Nyquist, spiegare i suoi termini ed il suo significato. Specificare le unitá di misura dei termini e del risultato.**
- **Fare un esempio numerico in cui si usano le due equazioni per stimare il throughput massimo di un collegamento.**

Le equazioni di Shannon e di Nyquist forniscono la capacità massima di trasmissione dei dati in un canale, ma con assunzioni differenti: il primo fornisce dei limiti ideali, il secondo dei limiti che tengono conto della presenza di rumore sul canale.

L'equazione di Shannon, stabilisce il limite massimo della capacità C di un canale in cui é presente rumore. È espressa come:

C=Blog_2(1+S/N) Dove: C è la capacità del canale, misurata in bit per secondo (bit/s). B rappresenta la larghezza di banda del canale, espressa in Hertz (Hz). S/N è il rapporto segnale-rumore (Signal-to-Noise Ratio, o SNR), un valore senza unità che rappresenta il rapporto tra la potenza del

segnale e quella del rumore. L'equazione dice che la capacità di un canale dipende sia dalla larghezza di banda sia dalla qualità del segnale rispetto al rumore. La capacità cresce con S/N, ma l'aumento è logaritmico: un raddoppio del SNR non raddoppia la capacità. Questo valore rappresenta il massimo teorico raggiungibile, indipendentemente dalla tecnologia utilizzata. L'equazione di Nyquist si applica ai canali ideali, ossia privi di rumore, e descrive la capacità massima di trasmissione in funzione della larghezza di banda e del numero di livelli discreti M utilizzati per codificare i dati. L'equazione è: C=2Blog_2(M) dove: C è la capacità del canale in bit/s. B è la larghezza di banda in Hz. M rappresenta il numero di livelli distinti utilizzati per codificare i bit in un simbolo. Ad esempio, se M=2, ogni simbolo rappresenta un bit; se M=4, ogni simbolo rappresenta due bit, ecc... La capacitá cresce logaritmicamente con M, ma nella pratica, usare un numero maggiore di livelli richiede ricevitori più precisi per distinguere tra i simboli, che diventa sempre piú difficile in presenza di rumore. L'equazione peró non tiene conto del rumore, assumendo che il ricevitore possa sempre distinguere i simboli. Entrambe le equazioni sono lineari con la larghezza di banda, che é un fattore determinante per aumentare la capacitá del canale. Immaginiamo un canale con: Larghezza di banda B= 2 MHz. Numero di livelli M=16 (ogni simbolo rappresenta 4 bit). Rapporto segnale-rumore S/N=31 Con Nyquist calcoliamo la capacità massima ideale: C=2Blog_2(M)=224 = 16 Mb/s Con Shannon, considerando il rumore, la capacità reale è: C=Blog_2(1+S/N)=2log_2(1+31) = 25=10 Mb/s

Il rumore limita la capacità reale del canale rispetto al valore ideale di Nyquist. Se il trasmettitore invia dati con un bit-rate piú alto da quello dato dall'equazione di Shannon, i dati (mediamente) conterranno errori.

Descrivi il meccanismo alla base del NATP. Cerca di rispondere alle seguenti domande:

A che serve il NAT, perchè è stato introdotto, quale problema risolve?

Come funziona? riporta un esempio di una tabella NAT di un router/firewall

Menzione e spiega almeno un problema che il NAT introduce rispetto all'architettura di Internet pensata in origine

Il Port-base Nettwork address Translation è una tecnica che serve a poter utilizzare delle classi di inidirizzi IP privati in modo da non saturare il numero di indirizzi IPv4 pubblici
Ciascun terminale collegato ad Internet dovrebbe avere un indirizzo IP unico, ma il numero di indirizzi IPv4 è limitato a 232 che non è sufficiente per i terminali al momento in uso. Una soluzione di lungo termine prevede di utilizzare una nuova versione di IP (v6) con indirizzi a 128 bit. Una soluzione temporanea consiste nel poter riutilizzare alcune classi di indirizzi. Si definiscono quindi 4 classi di indirizzi private, che sono la 10/8, la 172.16/20, la 192.168/16 e la 100.64/10 che sono classi private non routabili. Possono essere usate all'interno di una subnet ma un router di bordo non può instradare pacchetti che siano destinati a questi indirizzi. Ciascuna rete privata potrà accedere ad Internet attraverso il suo router di bordo, che avrà un indirizzo IP pubblico che maschera la rete privata. In questo modo esistono molte (milioni) di reti private in Internet i cui terminali riutilizzano gli stessi indirizzi, e possiamo risparmiare sull'uso degli indirizzi IP pubblici.
Quando un terminale dietro ad un NAT invia verso Internet un pacchetto IP, questo contiene negli header un indirizzo sorgente di una rete privata (sia 192.168.1.2) ed un indirizzo destinazione pubblico (sia 8.8.8.8). Un NAT è un router o un firewall che ha due interfacce di rete, una con un indirizzo nella rete privata (sia 192.168.1.1) e uno pubblico (sia 1.2.3.4). Il NAT modifica il pacchetto IP in uscita dalla rete, sostituendo l'IP sorgente con quello pubblico del NAT, come segue: S: 192.168.1.2; D: 8.8.8.8 -> S: 1.2.3.4; D: 8.8.8.8. Nel momento in cui il server esterno riceve il pacchetto, risponderà (se previsto) inviando un pacchetto con IP destinazione 1.2.3.4. Il NAT riceverà il pacchetto e farà la traduzione inversa: S: 8.8.8.8; D: 1.2.3.4 -> S: 8.8.8.8; D: 192.168.1.2 Per tenere traccia delle conversioni da fare il NAT mantiene una tabella NAT, che contiene IP e porte del flusso (S_IP, D_IP, S_P, D_P) sia in ingresso che in uscita. Nell'esempio precedente si avrebbe una entry del tipo:
192.168.1.2; 8.8.8.8; 12345; 80 -> 1.2.3.4; 8.8.8.8; 12345; 80 Se due flussi fossero indicizzati dalla stessa qudrupletta, il NAT potrebbe cambiare la porta di uscita, rendendo i flussi distinguibili, come segue: 192.168.1.2; 8.8.8.8; 12345; 80 -> 1.2.3.4; 8.8.8.8; 12345; 80
192.168.1.3; 8.8.8.8; 12345; 80 -> 1.2.3.4; 8.8.8.8; 12346; 80 In questo caso una seconda macchina (192.168.1.3) nella subnet apre una connessione verso 8.8.8.8 usando le stesse porte, e id il NAT cambia la porta di uscita nella sua tabella a 12346.

NAT presenta molti problemi, di cui se ne citano solo due. Il primo è che le macchine dietro un NAT non possono essere raggiunte dalle macchine fuori dal NAT. Questo è de-facto, un modo per mascherare macchine e non esporle ad attacchi dall'esterno, ma limita alcune applicazioni, ad esempio, la possibilità di ospitare un server raggiungibile dall'esterno. La seconda è che il NAT deve comportarsi come un firewall, ricomporre i frammenti (che invece andrebbe fatto end-to-end) e ispezionare il traffico.

Spiega quali protocolli vengono utilizzati per implementare l'architettura necessaria per inviare e ricevere posta elettronica. Copri i seguenti punti:

Quali sono i due principali protocolli coinvolti? fai un esempio di consegna da un MUA ad un altro di un email verso bob@bob.com da alice@alice.com

Quale è l'interazione tra questi protocolli ed il protocollo DNS?

Descrivi brevemente una tecnica che serve a ridurre il problema dello spam

La posta elettronica si basa su almeno due protocolli principali SMTP e POP3/IMAP. Il primo è un protocollo che si usa per consegnare la posta elettronica verso una mailbox conenuta in un server, i secondi due sono invece necessari all'utente per recuperare la mail una volta che è stata consegnata al suo server di posta. Alice configura il proprio MUA per usare il server SMTP di uscita dalla propria rete, verso cui fa un'autenticazione. Il server in uscita le permette di fare relay verso qualsiasi altro dominio. Il MUA si collega al server SMTP e consegna l'email per bob@bob.com, che non è un utente locale. Il server locale dovrà scoprire quale è il server responsabile del dominio bob.com, si collegherà a quel server SMTP e consegnerà l'email. In questo caso non esiste una fase di autenticazione, quindi la consegna può essere fatta solo per e-mail con il dominio bob.com. A quel punto l'email è arrivata nella mailbox di Bob, il quale può usare POP3 o IMAP per contattare il server, ricevere la lista delle nuove e-mail e scaricarle in locale. Il server di uscita di Alice non sa quale sia il server SMTP che gestisce le e-mail per il dominio bob.com. Aarà una risoluzione di dominio per bob.com per il record MX, che serve a ricevere il dominio (o l'indirizzo IP) del server SMTP di bob.com. Idealmente, chiunque può collegarsi ad un server SMTP e consegnare una e-mail, fingendo di essere esso stesso un server SMTP. Una tecnica elementare per evitarlo è quella di fare una risoluzione inversa dell'IP da cui il server riceve la connessione. Questo impone che chi si comporta da client SMTP (senza fare un'autenticazione) debba avere un fully-qualified domain name, ovvero che is presenti nel messaggio HELO con un dominio valido, e che facendo una risoluzione inversa del suo IP, si ottenga un dominio valido. Questo rende più difficile per un computer senza un indirizzo IP pubblico di comportarsi come un MTU verso un server SMTP

LE DOMANDE E RISPOSTE QUI SOTTO NON SONO ESATTAMENTE LE STESSE DEL PROF, MA MOLTO SIMILI (sono “a memoria” da esami passati)

Descrivere il funzionamento dell'infrastruttura a chiave pubblica (PKI), inclusa la creazione e verifica dei certificati, l'organizzazione gerarchica delle Autorità di Certificazione (CA) e il ruolo delle Liste di Revoca dei Certificati (CRL).

Come funziona l'interfaccia a chiave pubblica e la creazione/verifica dei certificati?

Qual è l'organizzazione gerarchica delle Autorità di Certificazione (CA)?

A cosa servono le Liste di Revoca dei Certificati (CRL)?

L'infrastruttura a chiave pubblica (PKI) è un sistema che si basa su una parte fidata, chiamata Autorità di Certificazione (CA), per associare in modo sicuro una chiave pubblica a un'identità. Questo approccio riduce il numero di entità di cui un utente deve fidarsi, concentrando la fiducia su un numero limitato di CA note. Il funzionamento dell'interfaccia a chiave pubblica e il processo di creazione e verifica dei certificati inizia quando un'entità, ad esempio Alice, genera la sua coppia di chiavi, una pubblica (Pub_A) e una privata (Priv_A). Successivamente, Alice crea una Certificate Signing Request (CSR), un file in un formato specifico che contiene le sue informazioni identificative e la sua chiave pubblica (Pub_A), firmando questa richiesta con la sua chiave privata (Priv_A). Alice invia la CSR alla CA, che verifica l'identità di Alice (spesso richiedendo documenti) e la firma della CSR utilizzando Pub_A, confermando così che Alice è la proprietaria di Priv_A. La CA risponde con un certificato valido, che è essenzialmente la CSR firmata digitalmente dalla chiave privata della CA (Priv_CA). I certificati seguono uno standard, tipicamente X.509, e contengono campi importanti come un numero di identificazione della CA, un numero di serie del certificato, un periodo di validità, i dati che identificano il proprietario della chiave (es. nome, indirizzo IP, dominio) e la chiave pubblica stessa, oltre alla firma della CA.Quando un'altra entità, ad esempio Bob, riceve il certificato di Alice, lo verifica attraverso diversi passaggi:

1. Controlla che il nome riportato nel certificato corrisponda al nome di Alice.

2. Verifica che il certificato sia valido per il periodo corrente.

3. Si assicura che la CA sia un'autorità fidata e conosciuta, consultando un database di CA attendibili (presente, ad esempio, nel suo browser) per le quali possiede già la chiave pubblica

4. Decifra la firma della CA presente nel certificato utilizzando la chiave pubblica della CA (Pub_CA), ottenendo un digest (riassunto) D del certificato

.

5. Calcola autonomamente un hash del certificato (D'), escludendo il campo della firma

6. Se D è uguale a D', il certificato è considerato valido e Bob può fidarsi che Pub_A appartenga ad Alice

L'organizzazione gerarchica delle Autorità di Certificazione (CA) è alla base della catena di fiducia che la PKI implementa. Al vertice di questa gerarchia si trovano le Root CA, come ad esempio Microsoft o Apple, le cui chiavi pubbliche (sotto forma di certificati auto-firmati) sono pre-memorizzate e considerate attendibili da tutti i browser e sistemi operativi. Al di sotto delle Root CA possono esserci altre CA intermedie, a volte chiamate Regional CA o CA secondarie, che vengono certificate a loro volta dalle Root CA o da altre CA intermedie superiori. Quando un server web, ad esempio www.alice.com, deve presentare il proprio certificato a un client, esso deve fornire l'intera catena di certificati, partendo dal proprio e risalendo fino alla Root CA. Il browser del client esamina e verifica ogni certificato nella catena, partendo dalla Root CA, per accertarsi dell'autenticità di tutti gli anelli della catena fino al certificato finale. Infine, il ruolo delle Liste di Revoca dei Certificati (CRL) è cruciale per la gestione della sicurezza. Un certificato, una volta emesso, potrebbe dover essere invalidato prima della sua data di scadenza naturale. Le motivazioni per la revoca possono includere la perdita di un dispositivo contenente una chiave privata, la compromissione di una chiave privata da parte di un ex-dipendente, o la violazione di un server della CA stessa. Le CRL sono liste di certificati che sono stati revocati dalla CA che li ha emessi. Ogni CRL è firmata dalla chiave della stessa CA e pubblicata da essa. I browser dovrebbero aggiornare periodicamente le CRL da tutte le CA che riconoscono per verificare lo stato di un certificato. Tuttavia, non esiste un accordo universale o un protocollo standardizzato per la migliore metodologia di aggiornamento delle CRL.

Spiegare i principio di funzionamento dello Spanning Tree Protocol. Seguire la seguente traccia:

1) quale problema si risolve con STP?

2) descrivere nel dettaglio lo scambio dei frame necessari per la formazione dello STP

3) Riportare le condizioni di confronto dei BPDU che si usano per prendere decisioni in SPT

Lo Spanning Tree Protocol (STP) è un protocollo fondamentale nel livello data link per le reti Ethernet, progettato per risolvere problemi specifici derivanti dalla topologia di rete. Il problema che si risolve con STP è la formazione di switching loop (anelli di commutazione) nelle reti Ethernet. Nelle reti con più switch interconnessi, in assenza di un meccanismo per prevenire i loop, un frame inviato in broadcast può viaggiare all'infinito, saturando la capacità dei collegamenti e sovraccaricando le CPU degli switch, fino a bloccare completamente la rete. Questo accade perché Ethernet non prevede un campo Time-to-Live (TTL) nei suoi frame, a differenza del livello di rete. La presenza di anelli, sebbene problematica, è però importante per creare reti ridondanti che resistono ai guasti. STP permette di avere questa ridondanza senza incorrere nei problemi dei loop. La formazione dello STP avviene attraverso uno scambio di frame specifici e un processo distribuito:
1. Identificazione degli switch: Ogni switch in una rete che supporta STP possiede un identificatore unico di 64 bit, composto da 16 bit configurabili dall'amministratore e 48 bit derivati dall'indirizzo MAC del produttore
2. Selezione della Root Bridge (Root CA): L'STP costruisce un albero di copertura (spanning tree) in cui la Root Bridge è lo switch con l'ID più basso. L'amministratore può influenzare questa scelta configurando i primi 16 bit dell'ID per posizionare la Root Bridge in una posizione strategica, ad esempio vicino a un router di bordo per motivi di prestazioni.
3. Scambio di BPDU (Bridge Protocol Data Unit): Ogni switch invia periodicamente (normalmente 10 volte al secondo) dei frame speciali chiamati BPDU su tutte le sue porte. Inizialmente, ogni switch si considera la Root Bridge e invia BPDU con il proprio ID come `Rootid` e un costo pari a zero. I BPDU sono inviati a un indirizzo multicast specifico e vengono processati solo dagli switch vicini, senza essere inoltrati.
4. Processo di convergenza: Quando uno switch (es. Switch A) riceve un BPDU da un vicino (es. Switch B), calcola un costo del percorso verso la root (`c = Cost + CAB`, dove `CAB` è il costo del collegamento tra A e B) e un vettore di priorità per quella porta. Se questo vettore indica un percorso migliore verso una Root Bridge con un ID più basso o un costo inferiore rispetto a quanto conosciuto, lo Switch A aggiorna la sua conoscenza e identifica la nuova Root Bridge, aggiornando di conseguenza i BPDU che invierà.

Stati delle porte: Durante la convergenza, le porte degli switch assumono tre stati principali: Root Port Su ogni switch non-root, è la porta che offre il percorso con il costo più basso verso la Root Bridge; Designated Port: Su ogni segmento di rete, è la porta dello switch che offre il percorso con il costo più basso verso la Root Bridge; Blocked Port: Le porte che non sono né Root né Designated vengono bloccate per prevenire i loop. Le porte bloccate non inoltrano il traffico dati, ma continuano a ricevere ed elaborare i BPDU per monitorare i cambiamenti nella topologia.

Rilevamento dei cambiamenti di topologia: Gli switch mantengono un "age" (età) per l'ultimo BPDU ricevuto. Se questo valore supera un massimo predefinito, lo switch rileva un cambiamento di topologia (es. un guasto di un vicino) e avvia nuovamente il protocollo STP, bloccando tutte le porte e cercando una nuova Root Bridge.

Le condizioni di confronto dei BPDU sono basate su un ordinamento lessicografico. Quando due BPDU vengono confrontati, la decisione su quale sia "migliore" viene presa seguendo questi criteri in ordine di priorità: ID della Root Bridge (`Rootid`): Il BPDU con il `Rootid` numericamente più basso è considerato migliore. Questo è il criterio principale per la selezione della Root Bridge e per determinare i percorsi più brevi verso di essa. Costo del percorso verso la Root (`Cost`): Se i `Rootid` sono uguali, si confronta il costo del percorso per raggiungere la Root Bridge. Il BPDU con il costo minore è preferito. ID del Sender (`Senderid`): Se sia il `Rootid` che il `Cost` sono uguali, si confronta l'ID dello switch che ha inviato il BPDU (`Senderid`). Il BPDU proveniente dallo switch con l'ID più basso è considerato migliore. Numero di porta (`p`): Se tutti i criteri precedenti sono in parità, si utilizza il numero di porta da cui è stato inviato il BPDU (`p`) per risolvere ulteriormente i conflitti.

Attraverso questi meccanismi, STP garantisce che la rete abbia sempre un unico percorso attivo tra due punti qualsiasi, eliminando i loop e fornendo al contempo un percorso di backup in caso di guasti.

Spiegare come il traffico di rete funziona nel 802.11MAC 802.11 nel problema di accessi multipli alla rete, come funziona DCF senza RTS/CTS. Cosa serve il tempo di backoff

Nelle reti Wi-Fi, pur essendo l'organizzazione logica a stella, la comunicazione fisica avviene come su un bus. Questo significa che tutte le stazioni condividono lo stesso mezzo trasmissivo e un ricevitore può gestire un solo segnale alla volta. Se più stazioni tentano di trasmettere contemporaneamente, si verifica una collisione, con conseguente perdita di tutti i frame coinvolti. Il compito del livello MAC (Media Access Control) è proprio quello di coordinare l'accesso a questo mezzo condiviso per minimizzare le collisioni e ottimizzare l'efficienza della rete. Il protocollo DCF è la modalità di accesso al mezzo più diffusa in 802.11 e implementa un meccanismo di accesso casuale noto come CSMA/CA. Senza l'utilizzo dei frame RTS/CTS (Request To Send/Clear To Send), il DCF opera nel seguente modo:
Quando un terminale intende trasmettere un frame, innanzitutto ascolta il canale (carrier sensing) per verificare se è già occupato. Se il canale è rilevato come libero, la stazione attende per un intervallo di tempo chiamato DIFS. Se il canale rimane inattivo per tutta la durata del DIFS, la stazione procede con la trasmissione del suo frame. Il terminale ricevente, una volta ricevuto correttamente il frame e verificato il checksum, attende un breve intervallo SIFS (Short-IFS) e invia un frame di ACK (Acknowledgement) per confermare la corretta ricezione. Se il mittente non riceve l'ACK entro un certo tempo, assume che la trasmissione non sia andata a buon fine, probabilmente a causa di una collisione.

Per facilitare la gestione del canale, ogni frame include nel suo header un campo chiamato NAV (Network Allocation Vector). Questo campo specifica la durata totale necessaria per completare l'invio del pacchetto e ricevere il relativo ACK. Tutte le stazioni che si trovano nel raggio di ricezione e leggono l'header del frame (anche se non sono i destinatari diretti) aggiornano il loro NAV, sapendo così di non poter trasmettere per quella durata, "riservando" di fatto il canale. Nonostante il carrier sensing e il NAV, le collisioni possono ancora verificarsi. Un esempio si ha quando due stazioni vicine ricevono quasi simultaneamente un frame dal sistema operativo. Entrambe potrebbero rilevare il canale libero dopo il DIFS e iniziare a trasmettere, causando una collisione sul ricevitore. Questo problema è particolarmente rilevante nelle LAN wireless, dove la vicinanza fisica dei nodi può portare a una sincronizzazione involontaria delle trasmissioni. Il NAV, infatti, può accentuare questo problema, sincronizzando le stazioni che terminano di attendere il canale dopo lo stesso periodo. Per mitigare queste collisioni "sincronizzate", il DCF introduce un tempo di backoff casuale. Quando il canale torna inattivo, ogni stazione che ha un frame da trasmettere non trasmette immediatamente, ma attende per un intervallo di tempo casuale estratto da un intervallo [0, CW] (Contention Window). Poiché questo tempo è casuale, sarà diverso per ogni stazione. Se il timer di una stazione scade per prima, questa inizia a trasmettere. Le altre stazioni, ricevendo il frame, rileveranno il canale come occupato e congelano il proprio timer di backoff, riprendendolo solo quando il canale tornerà libero. Questo meccanismo riduce notevolmente la probabilità di collisioni, sebbene non la elimini del tutto, specialmente se la differenza tra i timer casuali è inferiore al tempo di propagazione del segnale. La probabilità di collisione diminuisce all'aumentare della dimensione della finestra CW. In caso di trasmissione fallita (mancanza di ACK), il mittente riprova l'invio, ma prima di farlo, implementa una strategia di Binary Exponential Backoff. Questo significa che la dimensione della finestra di contesa (CW) viene aumentata, raddoppiando a ogni tentativo fallito secondo la regola `CWi = 2 × CWi-1`, fino a un valore massimo `CWmax`. Questo rende meno probabile che i timer casuali di più stazioni siano simili in situazioni di congestione. Dopo una trasmissione riuscita, il CW viene resettato al valore minimo `CWmin`. Il backoff si attiva sia quando una stazione deve trasmettere ma il canale è occupato, sia dopo una trasmissione non riuscita. In sintesi, mentre i timer casuali riducono le collisioni, possono anche portare a momenti in cui il canale rimane inattivo, anche sotto carico, riducendo l'efficienza complessiva della rete e sprecando risorse.

Congestion window, cos'è, che relazioni ha con le altre finestre. Mild/severe congestion e come cambia la cwnd di conseguenza

Nelle reti TCP, la finestra di congestione (cwnd) è un parametro fondamentale introdotto per gestire il problema della **congestione di rete**, che si verifica quando i router intermedi sono sovraccarichi e iniziano a scartare i pacchetti. A differenza della finestra di ricezione (rwnd), che è controllata dal ricevitore per prevenire il suo sovraccarico, e della finestra di invio (swnd), che è una variabile interna del mittente, la cwnd è una variabile che il mittente utilizza per limitare la quantità di dati non ancora confermati che possono essere inviati sulla rete in un dato momento, con l'obiettivo di evitare o mitigare la congestione. La inestra di invio effettiva del mittente, ovvero la quantità massima di dati che il mittente può avere in transito senza aver ricevuto una conferma, è determinata dal minimo tra la propria finestra di invio (`swnd`), la finestra di ricezione annunciata dal destinatario (`rwnd`) e, appunto, la finestra di congestione (`cwnd`). Questo garantisce che il mittente non sovraccarichi né il destinatario (rispettando `rwnd`) né la rete stessa (rispettando `cwnd`). Inizialmente, all'apertura di una connessione TCP, il mittente non conosce le condizioni della rete e, per prevenire un'eccessiva immissione di traffico che potrebbe causare congestione, adotta l'algoritmo di Slow Start. La `cwnd` viene inizializzata a un valore molto piccolo, tradizionalmente un MSS (Maximum Segment Size), sebbene le implementazioni moderne la impostino spesso a 10 volte l'MSS (ad esempio, 14600B se l'MSS è 1460B). Durante la fase di Slow Start, per ogni RTT (Round-Trip Time) in cui tutti i segmenti inviati vengono correttamente confermati, la `cwnd` raddoppia esponenzialmente. Questo permette una crescita rapida ma controllata della `cwnd` fino a quando non si raggiunge una soglia, chiamata `sstrash` (slow-start threshold), anch'essa inizializzata a un valore relativamente piccolo (come l'MSS). Le variazioni della `cwnd` sono strettamente legate alla rilevazione di eventi di congestione, che vengono interpretati dal mittente come la perdita di pacchetti. Il TCP distingue tra due tipi principali di eventi di congestione:
1 (Mild Congestion: Questo evento si verifica quando il mittente riceve tre ACK duplicati per lo stesso segmento. La ricezione di ACK duplicati

indica che un segmento intermedio è andato perso, ma i successivi segmenti sono arrivati al destinatario. Quando ciò accade, la `cwnd` viene ridotta: La `sstrash` viene impostata alla metà del valore corrente di `cwnd`; La `cwnd` viene impostata al nuovo valore di `sstrash`. Questo è un meccanismo più "gentile" per ridurre la velocità di trasmissione, suggerendo che la rete è congestionata ma non in modo catastrofico.
2. Severe Congestion: Questo evento si verifica quando il timer di ritrasmissione scade. La scadenza del timer indica che un segmento (o più segmenti) è andato perso e che l'ACK corrispondente non è stato ricevuto entro il tempo previsto. Questo è un segnale più forte di grave congestione di rete. Quando ciò accade: La `sstrash` viene impostata alla metà del valore corrente di `cwnd`; la `cwnd` viene resettata al valore iniziale (`cwnd0`), riavviando di fatto la fase di Slow Start. Questo approccio è più aggressivo, portando a una drastica riduzione della velocità di trasmissione per permettere alla rete di recuperare dalla congestione. Questi meccanismi di controllo della congestione, inizialmente proposti da Van Jacobson e poi evoluti in numerose varianti (come TCP Reno, CUBIC o BBR), sono cruciali per garantire che il TCP possa operare efficientemente su una vasta gamma di condizioni di rete, adattando dinamicamente la propria velocità di invio e bilanciando la necessità di massimizzare il throughput con quella di prevenire il collasso della rete. L'obiettivo finale è quello di mantenere la rete il più possibile "occupata" con dati utili, riducendo al minimo gli sprechi dovuti a collisioni o scarti di pacchetti.

pseudocodice protocollo DV nei router, cos è il count to infinity e la differenza fra l’algoritmo di routing basato sul distance vector “base” e quello usato nel protocollo BGP

per l'invio del DV

```
Ogni N secondi:

  v = Vector()

  per ogni destinazione d nella Routing Table R:

    # Aggiungi la destinazione d al vettore

    v.add(Pair(d, R[d].costo))

  per ogni interfaccia i:

    # Invia il vettore v su questa interfaccia

    send(v, i)
```

per la ricezione dei DV

```
Funzione received(V, l): # V: vettore ricevuto, l: link da cui è stato ricevuto

  per ogni destinazione d in V:

    se d non è in R: # Nuova rotta

      R[d].link = l

      R[d].costo = V[d].costo + l.costo

      R[d].time = now()

    altrimenti: # Rotta esistente, devo aggiornare?

      se ((V[d].costo + l.costo) < R[d].costo) oppure (R[d].link == l):

        R[d].link = l

        R[d].costo = V[d].costo + l.costo

        R[d].time = now()
```

Il routing Distance Vector (DV) è un algoritmo distribuito utilizzato dai router per costruire le proprie tabelle di routing. Ogni router mantiene una tabella di routing che, per ogni destinazione conosciuta, indica il link di uscita e il costo per raggiungere tale destinazione. Il problema del count-to-infinity è una limitazione intrinseca degli algoritmi di routing Distance Vector che si manifesta in reti con loop e in presenza di perdita di pacchetti o ritardi nella propagazione delle informazioni. Si verifica quando un link fallisce, e i router coinvolti aggiornano erroneamente le loro tabelle di routing basandosi su informazioni obsolete. Supponiamo che il router D abbia una rotta per la destinazione E attraverso il router A. Se il link tra D ed E si rompe, D dovrebbe settare il costo per E a infinito. Tuttavia, se prima che questa informazione si propaghi, A invia il suo DV a D, e A ha ancora una rotta per E (che in realtà passa attraverso D), D potrebbe aggiornare la sua tabella credendo di poter raggiungere E attraverso A, aumentando il costo. Questo crea un loop: D crede di raggiungere E via A, e A crede di raggiungere E via D (o un altro percorso che alla fine punta a D). I router continueranno ad aggiornare le loro distanze per quella destinazione, aumentandole gradualmente (il "conteggio")

fino a raggiungere il valore massimo di "infinito". Questo spreca risorse di rete e causa periodi di routing non funzionale. Anche con meccanismi come gli aggiornamenti "triggered" (invio immediato del DV in caso di cambiamento di costo di una rotta) o lo Split-Horizon con Poison Reverse (un router non annuncia una rotta a un vicino se quel vicino è il next-hop per quella rotta, o lo annuncia con costo infinito), il problema non è completamente risolto. Questi meccanismi possono mitigare il problema in loop di due router, ma non in loop più grandi. Ritardare la convergenza (tramite timer di hold-down) può aiutare nella pratica ma rende la rete estremamente lenta ad adattarsi ai cambiamenti. La causa principale è che i messaggi DV contengono solo la distanza e non l'intero percorso, quindi un router non può sapere se la rotta offerta da un vicino passa in realtà attraverso se stesso. Pur utilizzando i principi del routing Distance Vector, BGP presenta tre differenze chiave:
1. Non esporta solo la distanza, ma l'intero percorso di Autonomous Systems (AS) per raggiungere una destinazione. Questo significa che prima di annunciare un prefisso, un AS verifica che non sia già presente nel percorso, prevenendo la creazione di loop di qualsiasi lunghezza. Questa è una soluzione fondamentale al problema del count-to-infinity
2. BGP non invia regolarmente aggiornamenti. Invia aggiornamenti solo quando qualcosa cambia nella rete o quando un vicino chiede esplicitamente un aggiornamento. Questo riduce l'overhead di controllo
3. I messaggi BGP UPDATE contengono informazioni solo su specifici prefissi: se una rotta è nuova, se uno dei suoi attributi (come l'AS Path) è cambiato, o se la rotta è diventata irraggiungibile e deve essere ritirata (withdrawal message). Questo rende gli aggiornamenti più efficienti e mirati.

algoritmo di nagle, pseudocodice e spiegare quando è efficiente nel nel trasmettere e quando invece non lo è

L'algoritmo di Nagle è una strategia implementata nel TCP per migliorare l'efficienza della rete, in particolare per evitare l'invio di molti segmenti TCP di piccole dimensioni, che comporterebbe un notevole spreco di capacità dovuto agli header del protocollo. Un segmento TCP, infatti, ha almeno 40 byte di header (20 IP + 20 TCP), quindi inviare un solo byte di dati per segmento significherebbe sprecare 40/41 della capacità per i soli header. L'algoritmo di Nagle cerca un compromesso tra l'invio immediato dei dati (necessario in alcune applicazioni come una shell remota) e l'attesa di una quantità sufficiente di dati per riempire un segmento completo.Ecco il suo pseudocodice:

```
if len(data) >= MSS and rcv.wnd >= MSS:

           send one MSS - sized segment

else:

           if there are unacknowledged data:

                   place data in buffer until acknowledgment has been received

         else:

                  send one TCP segment containing at most rcv.wnd data
```

Le righe 1-2 dello pseudocodice indicano che, se la quantità di dati da inviare è maggiore o uguale all'MSS (Maximum Segment Size) e la finestra di ricezione del destinatario (rcv.wnd) permette l'invio di un segmento di dimensioni MSS, il mittente invia immediatamente un segmento di dimensioni MSS. Questo approccio è pensato per i trasferimenti di massa (bulk transfers): se ci sono molti segmenti grandi da inviare, verranno inviati senza ritardi. Se il ricevitore vuole rallentare, può farlo inviando un pacchetto ACK con una finestra di ricezione (rcv.wnd) più piccola. Se i dati da inviare sono meno dell'MSS o la finestra di ricezione è piccola (riga 3), l'algoritmo prende decisioni più "sagge". Se ci sono dati non ancora confermati in transito (righe 4-5), il mittente attende che arrivi una conferma, mettendo in coda i nuovi dati nel buffer. Questo permette al buffer di riempirsi. Se, invece, non ci sono dati non confermati (righe 6-7), il mittente può inviare un piccolo segmento, ma solo una volta per RTT (Round Trip Time).
L'algoritmo è molto efficiente quando le applicazioni generano un throughput elevato, in quanto tende a inviare segmenti grandi, avvicinandosi il più possibile all'MSS. Questo massimizza il rapporto dati utili/header, riducendo lo spreco di capacità. Le misurazioni su Internet hanno osservato che la maggior parte dei pacchetti sono lunghi 1460B (che è l'MSS standard per Ethernet, considerando i 40B di header IP e TCP) o 0B (per i pacchetti ACK TCP). Inoltre, evitando l'invio di un gran numero di piccoli pacchetti, l'algoritmo di Nagle contribuisce a ridurre il carico sulla rete, mitigando potenzialmente la congestione.
Invece, ènadatto per il traffico in tempo reale: L'attesa di un ACK o il riempimento del buffer introduce un ritardo. Questo rende il TCP con Nagle non adatto per applicazioni che richiedono bassa latenza, come le shell remote interattive o il traffico in tempo reale (ad esempio, il gaming online o le videochiamate). In questi scenari, anche l'invio di un singolo byte (come una lettera digitata in una shell remota) deve essere quasi istantaneo. Per questo motivo, l'algoritmo di Nagle viene tipicamente disabilitato per tali applicazioni.Inoltre, L'algoritmo di Nagle può interagire in modo inefficiente con la strategia degli ACK ritardati. Se il mittente aspetta un ACK per inviare ulteriori dati (come nel caso di piccoli segmenti), e il ricevitore sta ritardando l'invio dell'ACK, ciò può causare ulteriori ritardi nella trasmissione.
In sintesi, l'algoritmo di Nagle è un'ottimizzazione fondamentale per l'efficienza della rete TCP, soprattutto per il trasferimento di file di grandi dimensioni, ma richiede di essere disabilitato per le applicazioni che privilegiano la bassa latenza rispetto al massimo throughput.

HTML, codice client-side e server-side, CSS

Le pagine web, che costituiscono il World Wide Web, sono una combinazione di diversi componenti, tra cui linguaggi di markup come HTML, codici eseguiti lato client e lato server, e fogli di stile come CSS. L'HTML è un linguaggio di markup utilizzato per creare pagine web e strutturare il contenuto. Inizialmente standardizzato dall'IETF e successivamente dal consorzio W3C, l'ultima versione stabile è HTML 5.0. Un documento HTML è composto da un header e un body, inclusi nei rispettivi tag. Il testo all'interno della pagina può essere formattato utilizzando vari tag, che possono anche fare riferimento a oggetti esterni come immagini o fogli di stile, i quali vengono caricati separatamente dal browser quando la pagina viene renderizzata. [metti qalche esempio di tag se vuoi].

Il codice client-side è un tipo di programmazione per pagine dinamiche in cui il codice è incorporato nella pagina HTML, inviato al browser e quindi eseguito dal browser stesso. Il linguaggio client-side più popolare è Javascript. Questo approccio è utilizzato per creare azioni dinamiche che, in alcuni casi, potrebbero non richiedere nemmeno l'uso della connessione di rete. Un esempio è l'ordinamento di elementi in una pagina (ad esempio, prodotti in un e-commerce per prezzo o data) direttamente nel browser dell'utente, senza la necessità di interrogare nuovamente il sito web. Questo modello si distingue dal codice lato server dove l'utente non dovrebbe accedere al codice sorgente, in quanto il codice viene eseguito dal server e l'HTML generato viene consegnato al browser.

Il codice server-side rappresenta un altro modo per creare pagine HTML con contenuti dinamici. In questo caso, il codice viene eseguito sul server e restituisce un file HTML al browser. Questo è noto come Common Gateway Interface (CGI). Linguaggi come PHP sono ampiamente utilizzati per la programmazione server-side. Il codice PHP può essere incorporato all'interno della pagina HTML stessa o essere direttamente referenziato dall'URL (ad esempio, un file .php). Il codice lato server può utilizzare le variabili passate come query nell'URL per generare contenuti personalizzati. Il codice server-side è tipicamente utilizzato per interrogare database e selezionare le informazioni corrette, senza che l'utente abbia accesso diretto al database.

Il CSS è uno standard che permette di regolare graficamente il contenuto di una pagina web, fornendo un disaccoppiamento tra i dati (struttura HTML) e lo stile (aspetto visivo). Questo significa che è possibile definire lo stile (ad esempio, il colore del testo) in un file separato, che viene poi caricato dal browser attraverso un tag `<link>` quando la pagina viene servita. Questo approccio è più efficiente rispetto alla modifica manuale dello stile per ogni singolo elemento HTML, specialmente quando si vuole applicare lo stesso stile a molti elementi. Un esempio è definire che tutti i paragrafi `<p>` debbano avere un certo colore o dimensione di font.

Oggi, molti siti web caricano i loro componenti da una moltitudine di fonti, inclusi librerie Javascript, fogli di stile CSS (come Twitter Bootstrap o FontAwesome) e immagini da siti esterni. Un sito web moderno può quindi combinare tutte queste tecnologie: codice server-side per interrogare un database e preparare i dati, codice client-side per azioni dinamiche nel browser e fogli di stile CSS per definire l'aspetto estetico della pagina.

Descrivere le tecniche di accesso al mezzo basate su TDMA (Time-Division Multiple Access) e S-ALOHA (Slotted ALOHA). Nella descrizione, seguire la traccia data dalle domande:

• Quali sono le caratteristiche principali del TDMA e quali i suoi vantaggi e svantaggi in termini di efficienza e latenza?

• Come funziona l'S-ALOHA e quali sono i concetti chiave della sua operazione, inclusi gli stati dei terminali?

• Quali sono le differenze nelle loro prestazioni sotto carico elevato e basso?

Le tecniche di accesso al mezzo sono fondamentali per coordinare la trasmissione in reti dove più terminali condividono lo stesso canale fisico, come nelle reti bus o wireless, al fine di evitare collisioni. Il TDMA e l'S-ALOHA rappresentano due approcci distinti a questo problema, uno basato sull'allocazione deterministica e l'altro sull'accesso casuale. Il TDMA (Time-Division Multiple Access) è una tecnica di accesso al mezzo in cui il tempo viene diviso in slot di dimensione fissa, e ogni slot viene assegnato a un terminale specifico in modo ciclico. Se una rete ha `m` terminali, il tempo viene suddiviso in `m` slot, e il terminale `i` può trasmettere solo nel suo slot assegnato.
Vantaggi: Quando tutti i terminali hanno sempre dati da trasmettere (situazione di saturazione), il TDMA raggiunge la massima efficienza, poiché ogni slot è occupato e non ci sono collisioni. È facile da implementare e garantisce che ogni utente abbia accesso a una porzione delle risorse del canale, anche in condizioni di sovraccarico. Questo è particolarmente utile in sistemi in cui si paga per il servizio (come le reti cellulari o satellitari) o in piconet (come il Bluetooth), dove la quantità di dati scambiati è solitamente limitata. Il TDMA non subisce mai un "congestion collapse" (collasso da congestione), ovvero un degrado improvviso delle prestazioni all'aumentare del carico, poiché ogni terminale ha una finestra di trasmissione garantita.
Svantaggi: Se un terminale ha un frame da inviare, potrebbe dover aspettare un ciclo completo di `m` slot prima di poter trasmettere, anche se gli altri nodi non hanno nulla da inviare. Questo introduce un ritardo medio non nullo e rende il TDMA inefficiente quando la rete è scarica, sprecando risorse. Inoltre, la sua natura di allocazione fissa lo rende meno flessibile rispetto ai protocolli ad accesso casuale.

L' S-ALOHA (Slotted ALOHA) è una tecnica di accesso casuale, antenato del CSMA/CA utilizzato nel Wi-Fi, in cui i terminali tentano di trasmettere e, in caso di collisione, ritrasmettono il frame. A differenza del TDMA, ogni terminale può trasmettere in qualsiasi slot.

1. Quando un nodo deve inviare un frame, attende l'inizio dello slot successivo e trasmette.
2. L'S-ALOHA ammette una probabilità di collisione.
3. Si assume che i terminali siano sincronizzati con l'inizio degli slot e che le trasmissioni siano più brevi di uno slot.
4. In caso di collisione (due o più trasmissioni contemporanee), la ricezione fallisce, e i terminali coinvolti rilevano immediatamente l'errore (o in un secondo momento, tramite la mancata ricezione di un ACK in uno slot dedicato).

* Stato "normale" e "backlogged": Un terminale in stato normale invia un frame con una probabilità `qa` (dipendente dal traffico in ingresso). Se si verifica una collisione, il terminale entra nello stato "backlogged".
* Ritrasmissione:Un terminale backlogged ritrasmette i frame precedenti con una probabilità `qb`, maggiore di `qa`. Un terminale torna allo stato normale solo dopo aver trasmesso con successo.
* Scarto di nuovi frame: Se un terminale backlogged riceve un nuovo frame da inviare, lo scarta.
* Carico di sistema (`G`): Definito come il numero medio di frame che devono essere trasmessi per slot da tutti i terminali.

Le differenze nelle prestazioni tra TDMA e S-ALOHA sono evidenti a seconda del carico di rete.
TDMA raggiunge la massima efficienza a carico elevato/saturazione, dove ogni slot viene utilizzato. A basso carico**, l'efficienza è ridotta perché molti slot rimangono vuoti, aumentando la latenza media.
A basso carico, l'S-ALOHA può essere più veloce, in quanto i terminali possono trasmettere non appena hanno dati, senza aspettare il proprio turno fisso. Tuttavia, all'aumentare del carico (`G`), la probabilità di successo delle trasmissioni (`PS`) aumenta fino a un massimo di `1/e` (circa 36,8%) quando `G=1` (saturazione), per poi diminuire drasticamente. Questo fenomeno è noto come "congestion collapse", dove l'aumento del carico porta a un'impennata delle collisioni e un crollo delle prestazioni effettive. L'S-ALOHA è intrinsecamente instabile.

Nel contesto dell'algoritmo di routing "Hot Potato", descrivere come il meccanismo di backward learning contribuisce alla costruzione delle tabelle di forwarding dei router. In particolare, spiegare come i router apprendono la raggiungibilità dei nodi sorgente attraverso l'osservazione del traffico, quali sono i vantaggi di questo approccio per la risposta ai pacchetti e quali le sue limitazioni intrinseche.

L'algoritmo "Hot Potato" è una tecnica di routing distribuito molto semplice, concepita per la costruzione dinamica delle tabelle di inoltro (forwarding tables) in reti, assumendo idealmente una topologia ad albero. Il suo funzionamento si basa sull'azione immediata di un router: quando riceve un pacchetto destinato a un nodo di cui non ha una voce nella sua tabella di inoltro, il router lo "diffonde" su tutte le sue interfacce, eccetto quella da cui il pacchetto è giunto, agendo come se volesse liberarsi rapidamente di una "patata bollente". In questo processo, ogni router implementa implicitamente un meccanismo di backward learning. Questo significa che, ogni volta che un router riceve un pacchetto su una specifica interfaccia (o "porta" nel contesto di uno switch), registra l'indirizzo sorgente di quel pacchetto e l'interfaccia da cui è stato ricevuto nella sua tabella di inoltro. Per esempio, se un host A invia un pacchetto a un host B, e il router R1 è il primo a ricevere questo pacchetto, R1 apprenderà che l'host A è raggiungibile tramite l'interfaccia da cui ha ricevuto il pacchetto, aggiornando di conseguenza la sua tabella di inoltro. Questo principio è analogo al modo in cui uno switch Ethernet apprende la posizione dei terminali nella sua rete. Il vantaggio principale di questo approccio è la sua semplicità e la capacità di costruire tabelle di routing in modo distribuito e auto-organizzante, senza la necessità di specifici pacchetti di controllo di routing o di una configurazione manuale estesa. Grazie a questo meccanismo, se un nodo B deve rispondere a un pacchetto originato da A, i router sul percorso di ritorno avranno già appreso, osservando il flusso di traffico iniziale da A, come raggiungere A. Pertanto, il pacchetto di risposta da B ad A non richiederà un ulteriore "broadcast" da parte dei router intermedi, ma verrà instradato direttamente lungo il percorso appreso, rendendo il processo di risposta efficiente una volta che il percorso di andata è stato "imparato". Tuttavia, le limitazioni intrinseche del backward learning nell'algoritmo Hot Potato sono significative. La più critica è la sua intrinseca instabilità in reti con topologie che contengono cicli, ovvero per qualsiasi struttura che non sia un albero. In una rete non ad albero, un pacchetto broadcast, inviato da un router che non conosce la destinazione, può essere ricevuto più volte dallo stesso router, che lo ritrasmetterà, creando un loop di inoltro. Questi loop possono causare la propagazione infinita dei pacchetti, saturando la capacità dei collegamenti e sovraccaricando la CPU dei router. Poiché i frame Ethernet non includono un campo Time-To-Live (TTL), a differenza dei pacchetti IP, i frame possono viaggiare indefinitamente in questi cicli. Pertanto, sebbene il backward learning sia un concetto efficiente per l'apprendimento delle sorgenti, la sua applicazione nel routing Hot Potato è funzionale solo in reti molto piccole e con topologie specifiche. Per reti più complesse, ridondanti e stabili, sono necessari algoritmi e protocolli di routing più sofisticati, come lo Spanning Tree Protocol (STP) per gli switch o i protocolli di routing a stato di link (Link State) o a vettore di distanza (Distance Vector) per i router a livello di rete

descrivere il ruolo dei Link State Packets (LSP). In particolare, spiegare la loro struttura e contenuto, come avviene la loro distribuzione all'interno della rete per la costruzione delle tabelle di inoltro, quali sono i vantaggi di questo approccio per la rapida reazione ai cambiamenti di topologia e quali le sue principali limitazioni in termini di scalabilità e overhead.

Nel campo dei protocolli di routing, il Link State (LS) si distingue per il suo approccio alla costruzione delle tabelle di inoltro, dove ogni router mira ad ottenere una visione completa e dettagliata della topologia di tutta la rete. Questo obiettivo viene raggiunto principalmente attraverso lo scambio e la propagazione di messaggi specifici noti come Link State Packets (LSP). Un LSP può essere inteso come una vera e propria "fotografia" dello stato dei collegamenti diretti di un router che lo genera. La sua struttura è ben definita e include diverse informazioni essenziali. Innanzitutto, contiene l'identificativo, o indirizzo, del router che ha creato e inviato l'LSP. Vi è poi un campo dedicato all'età (LSP.age) che indica la durata di validità residua dell'LSP stesso, un meccanismo utile per gestire l'obsolescenza delle informazioni. Un elemento cruciale

per la coerenza del sistema è il numero di sequenza (LSP.seq), un valore che viene incrementato ogni volta che il router emette una nuova versione del suo LSP, permettendo così agli altri router di distinguere le informazioni più aggiornate da quelle vecchie o duplicate. Infine, e forse il dato più significativo, l'LSP elenca tutti i link diretti del router, specificando per ciascuno l'identificativo del router vicino e il costo associato a quel collegamento. La diffusione di questi LSP all'interno della rete avviene tramite un meccanismo robusto e diffuso chiamato flooding (inondazione). Quando un router genera un nuovo LSP, che può essere innescato da un cambiamento nello stato di uno dei suoi link (ad esempio, un guasto o un ripristino) o da un aggiornamento periodico, lo invia su tutte le sue interfacce, con l'unica eccezione dell'interfaccia da cui, eventualmente, l'LSP stesso è stato ricevuto. Ogni router che riceve un LSP esegue un controllo: se si tratta di un LSP nuovo o di una versione più recente (con un numero di sequenza più alto) rispetto a quello che ha già nella sua Link State Database (LSDB), allora lo memorizza nella propria LSDB e lo ritrasmette a tutti i suoi vicini, escludendo ancora una volta la porta di ricezione. Se invece l'LSP ricevuto è obsoleto o già presente, viene semplicemente scartato. Questo processo capillare assicura che, in un periodo relativamente breve, ogni router nella rete detenga una copia aggiornata di tutti gli LSP generati da ogni altro router, consentendogli di costruire una mappa completa e coerente della topologia di rete. Con questa mappa, ogni router può poi utilizzare algoritmi come Dijkstra per calcolare i percorsi più brevi verso qualsiasi destinazione. I vantaggi di questo modello sono notevoli. Innanzitutto, la conoscenza globale della rete permette ai router di prendere decisioni di routing più informate e potenzialmente più efficienti, non solo basate sul percorso più breve ma anche su metriche più complesse come la capacità dei link. La convergenza della rete è generalmente rapida in seguito a un cambiamento di topologia; le informazioni sui guasti dei link, ad esempio, si propagano rapidamente attraverso il flooding degli LSP, portando a un ricalcolo e aggiornamento celere delle tabelle di routing. I messaggi HELLO, utilizzati per rilevare i vicini diretti, possono essere inviati con alta frequenza senza sovraccaricare la rete, dato che non vengono propagati oltre il link diretto, il che accelera la rilevazione di guasti sui collegamenti. Tuttavia, il routing Link State presenta anche alcune limitazioni importanti che ne condizionano l'applicabilità, specialmente in reti di grandi dimensioni come l'intera Internet. Il primo svantaggio è il costo computazionale elevato L'algoritmo di Dijkstra, sebbene efficiente per la sua classe, deve essere eseguito da ogni router su una rappresentazione completa della rete, e in una rete con migliaia o milioni di router, questo richiede una notevole potenza di calcolo e memoria per mantenere la LSDB e ricalcolare i percorsi. Inoltre, il meccanismo di flooding degli LSP genera un overhead di traffico di controllo sulla rete. Ogni minimo cambiamento nello stato di un link locale viene propagato a tutti i router della rete, forzandoli a ricalcolare le proprie tabelle, il che può risultare uno spreco di risorse in una rete molto dinamica e vasta. In fase transitoria, durante il flooding degli LSP, soprattutto in reti con topologie complesse che includono cicli, possono verificarsi temporaneamente instabilità come loop di routing o "buchi neri" (blackhole) fino a quando la rete non converge. Infine, l'affidabilità della propagazione degli LSP è fondamentale; se alcuni messaggi si perdono durante il flooding e non ci sono meccanismi per garantire la loro ricezione o una rigenerazione periodica, la LSDB di alcuni router potrebbe non essere coerente, impedendo una corretta convergenza. In sintesi, mentre il routing Link State, con i suoi LSP, offre una visione dettagliata della rete e una rapida reazione ai cambiamenti, la sua complessità computazionale e l'overhead generato lo rendono più adatto a domini di routing di dimensioni moderate, piuttosto che all'intera infrastruttura globale di Internet.