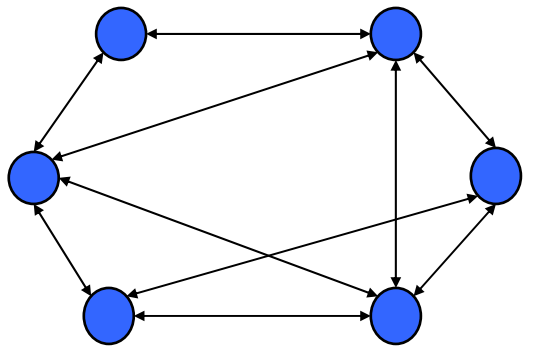
Contesto: Peer to Peer

- *Un'architettura di rete P2P è caratterizzata da:*
  - *Connessioni dirette tra i suoi componenti.*
  - *Tutti i nodi sono entità paritarie (**peer**).*
  - *Risorse di calcolo, contenuti, applicazioni sono fornite ad ogni peer in modo distribuito.*



Non esiste nessun elemento di gestione centrale	
Vantaggi	**Svantaggi**
Scalabilità Robustezza Dinamicità	Non vi è coordinazione

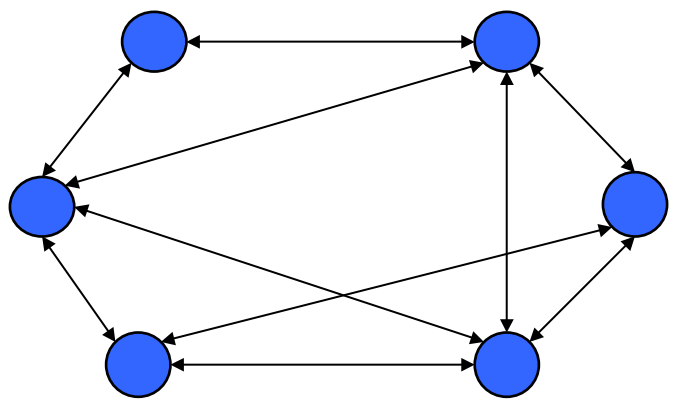
- *Esempi di protocolli P2P sono:*

Napster, Kazaa, BitTorrent, Skype, Chord, Can

Il 60-80% del traffico in Internet è generato da applicativi P2P

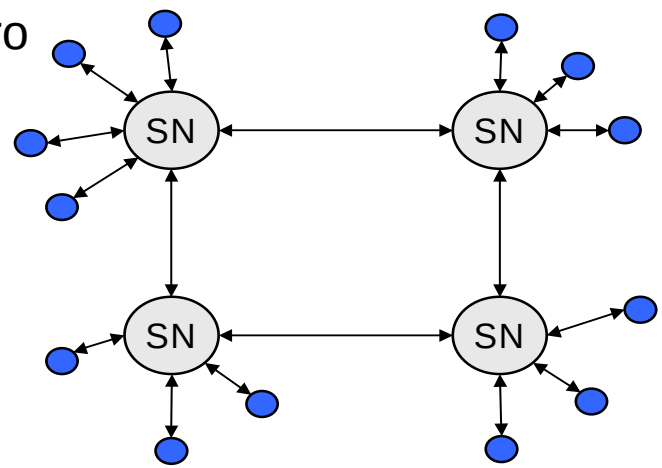
Architetture per-to-peer (1/3)

- P2P puro
  - Tutti i nodi della rete ricoprono lo stesso ruolo e svolgono le stesse funzioni, agendo sia da client che da server per le comunicazioni che vengono instaurate.
  - La rete è completamente decentralizzata e non esiste alcun elemento centrale che coordini le attività dei peer.
  - Risultano essere reti molto robuste non presentando nessun single point of failure e difficilmente controllabili .
  - Il loro principale svantaggio sta nella fase di ricerca che, avvenendo in maniera del tutto decentralizzata, risulta essere lenta e non sempre efficace.
  - Esempio: Gnutella



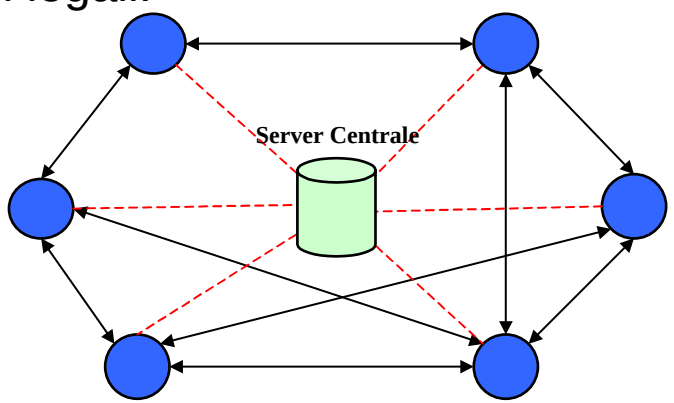
Architetture per-to-peer (2/3)

- P2P parzialmente centralizzato
  - Alcuni nodi (*supernodi*) assumono un ruolo di maggior importanza rispetto agli altri, svolgendo in particolare funzioni di search e location distribuito per gli altri peer.
  - I meccanismi secondo i quali i supernodi vengono scelti per questi "compiti speciali" variano da sistema a sistema, ma si basano quasi sempre sulla capacità di banda e/o di elaborazione degli host.
  - Selezione non statica.
  - I supernodi non costituiscono single point of failure
  - Riduzione dei tempi di ricerca rispetto al P2P puro
  - Esempi: BitTorrent, Kazaa ed Edonkey

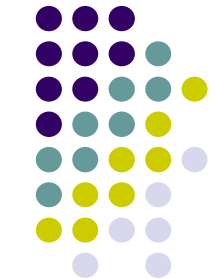


Architetture per-to-peer (3/3)

- P2P ibrido
  - Esiste un nodo centrale che agevola le operazioni di interazione tra i peer occupandosi delle operazioni di ricerca di utenti e file.
  - A seguito di una richiesta da parte di un utente, il server crea una lista di contenuti controllando i file degli utenti connessi in quel momento alla rete.
  - L'utente può scegliere uno o più contenuti e stabilire una connessione diretta con gli host per avviare le operazioni di download.
  - Grazie alla presenza di un indice centrale, le ricerche avvengono in modo rapido ed efficace.
  - Il server centrale si presenta come un single point of failure ed è un obiettivo visibile per possibili attacchi legali.
  - Esempio: WinMX

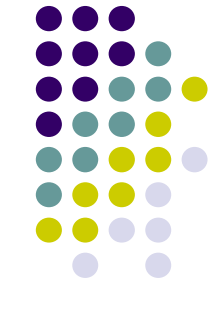


Meccanismi di search and location (1/4)



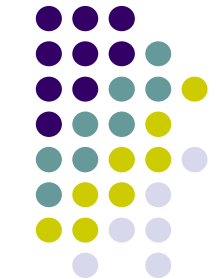
- Flooding broadcast
  - Utilizzato nelle reti P2P pure.
  - Quando un peer richiede una risorsa invia un messaggio in broadcast a tutti i peer vicini. Se i vicini non posseggono la risorsa, il messaggio di richiesta è inoltrato in broadcast ai peer vicini dei vicini.
  - Quando una risorsa viene trovata, il peer che la possiede invia un messaggio al peer che ha dato origine alla richiesta indicando come può essere raggiunto per dare inizio alla connessione P2P per la fase di trasferimento.
  - Utilizzo del campo TTL
  - L'occupazione di banda della rete cresce all'aumentare del numero di peer.
  - Meccanismo molto semplice da realizzare.

Meccanismi di search and location (2/4)



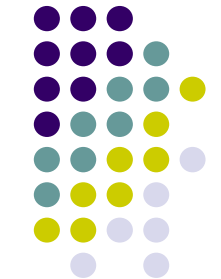
- Selective forwarding
  - Utilizzato nelle reti P2P parzialmente centralizzate.
  - I pacchetti di richiesta vengono inoltrati solo verso i supernodi della rete, che si occupano di effettuare la ricerca della risorsa richiesta.
  - Ogni utente invia la propria lista di file ad un supernodo. Ogni supernodo conosce quindi quali file sono posseduti dai nodi agganciati ad esso.
  - I supernodi periodicamente si scambiano tali liste.
  - La ricerca avviene all'interno della "rete dei supernodi", con tecniche simili comunque a quelle del flooding broadcast, ma con i messaggi che vengono scambiati solo tra gli stessi supernodi.
  - Riduzione dell'occupazione complessiva di banda della rete.

Meccanismi di search and location (3/4)



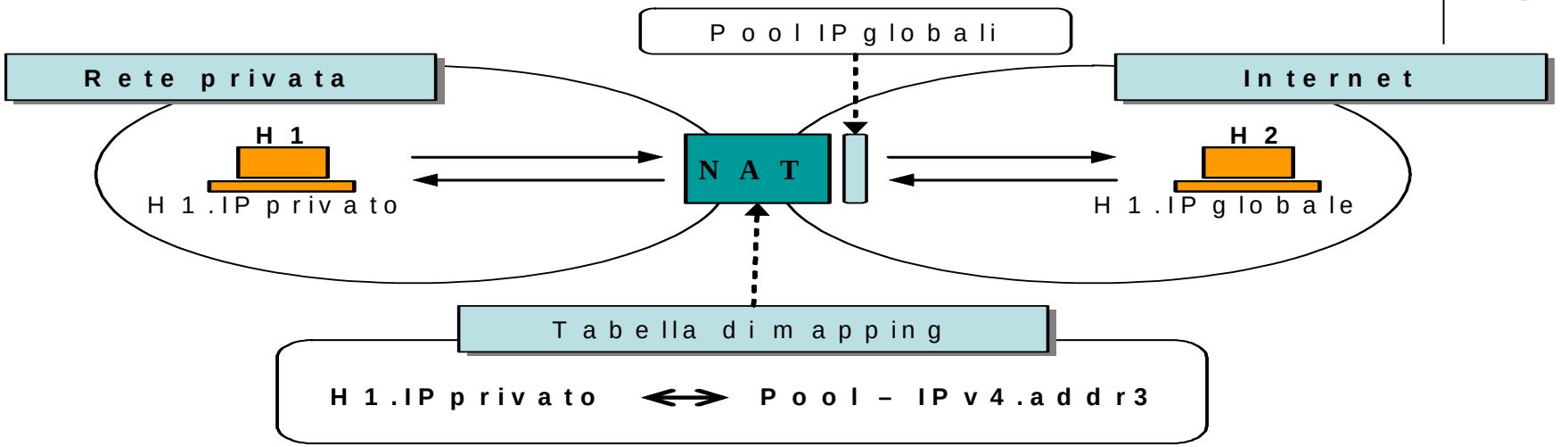
- Indici e repository centralizzati
  - Utilizzato nelle reti P2P ibride.
  - Il nodo centrale viene utilizzato come server per memorizzare un indice di tutti i peer e delle loro risorse.
  - Ogni utente invia la propria lista di file a tale nodo centrale.
  - Ogni richiesta viene inoltrata al server, che effettua una ricerca all'interno dell'indice.
  - Se la risorsa è disponibile, invia un messaggio al peer su dove può trovare il file richiesto (indirizzo IP di un altro utente che condivide il dato cercato).
  - L'utente può quindi connettersi direttamente all'IP che gli è stato fornito per ottenere la risorsa.
  - Meccanismo che consente elevate prestazioni in fase di ricerca, ma poco robusto.

Meccanismi di search and location (4/4)



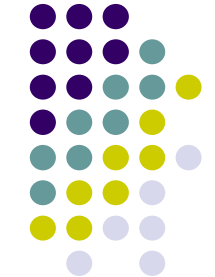
- Decentralized Hash Table (DHT)
  - Utilizzate nelle reti P2P di seconda generazione (*structured networks*).
  - Ogni file (o puntatore ad esso) è posizionato in una specifica locazione.
  - La posizione di una risorsa all'interno del sistema è stabilita applicando una funzione *f()*, comune a tutti i nodi, ad un identificativo della risorsa stessa.
  - Ad ogni file e ad ogni nodo è associato un identificativo unico.
  - Gli identificativi vengono creati effettuando un'operazione di *hashing* sul nome del file e sull'indirizzo IP del nodo.
  - L'identificativo del file viene detto "chiave".
  - Ogni nodo del sistema è responsabile di un insieme di file (o chiavi).
  - I collegamenti tra identificativi dei file e rispettive locazioni vengono realizzati sotto forma di tabelle di routing distribuite, le **DHT** in modo tale da poter inoltrare in modo efficiente ogni richiesta verso il nodo che possiede il file desiderato.
  - L'insieme di tutti i nodi della rete realizza la DHT.

Network Address Translation



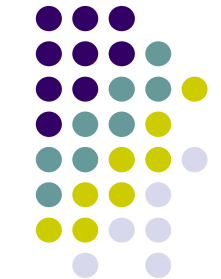
- Svantaggi
  - Comportamento non prevedibile di alcune applicazioni.
  - Aumento del carico computazionale con conseguenti limiti nelle prestazioni.
  - Il nodo che implementa la funzionalità di NAT diventa un single-point-of-failure.
  - Perdita dell'indipendenza tra indirizzamento IP e DNS.
  - Impossibilità di utilizzare i meccanismi di sicurezza e mobilità attualmente considerati (e.g. IPsec).
  - Maggior complessità nella diagnosi dei problemi di rete.

Classificazione dei server NAT



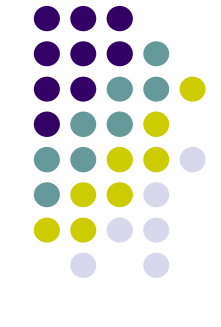
- Full Cone
  - E’ il meno restrittivo.
  - Una volta che l’host locale ha dato inizio ad una comunicazione verso l’esterno, permette a qualsiasi host di comunicare con l’host dietro il NAT tramite il suo indirizzo di trasporto pubblico (coppia indirizzo IP/numero di porto).
- Restricted Cone
  - Accetta tutti i pacchetti provenienti da un indirizzo IP verso il quale l’host locale ha precedentemente inviato pacchetti.
- Port Restricted Cone
  - Accetta tutti i pacchetti che hanno come porto sorgente lo stesso porto verso il quale l’host locale ha precedentemente inviato pacchetti.
- Symmetric
  - Accetta in ingresso solo pacchetti provenienti da indirizzo IP e porto, associati alla destinazione di una precedente comunicazione iniziata dall’host locale. Risulta essere, quindi, la tipologia più restrittiva.

NAT discovery: STUN (1/5)



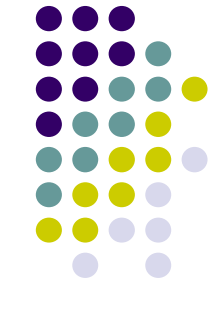
- STUN (*Simple Traversal of UDP through NATs*) è un semplice protocollo che permette alle applicazioni di determinare la presenza e il tipo di NAT dietro il quale si trova un host, esaminando lo scambio di informazioni tra l'host stesso (su cui si attiva un client STUN) ed un server STUN sul lato pubblico del NAT.
- Il server STUN deve poter inviare e/o ricevere pacchetti con due indirizzi IP e due porti oppure il meccanismo deve coinvolgere due diversi server.
- Il protocollo si basa su una semplice sequenza di domande-risposte con lo scambio di pacchetti UDP tra client e server.

NAT discovery: STUN (2/5)



1. Il client invia un pacchetto UDP al server in cui chiede di conoscere qual è il proprio indirizzo di trasporto pubblico. Se non riceve nessuna risposta il client conclude di essere dietro un NAT (e/o Firewall) che blocca i pacchetti UDP in uscita e in ingresso.
2. Se il server riceve il pacchetto inviato dal client, gli risponde inviando un pacchetto UDP con le informazioni su indirizzo IP e porto dal quale ha ricevuto la richiesta (indirizzo di trasporto pubblico del client).
3. Se l'indirizzo di trasporto pubblico, contenuto nel pacchetto di risposta del server, e l'indirizzo di trasporto privato sono gli stessi, il client può concludere che non vi sono NAT tra lui e il server.
4. Se l'indirizzo di trasporto pubblico, contenuto nel pacchetto di risposta del server, e l'indirizzo di trasporto privato differiscono, il client può concludere di essere dietro un NAT.

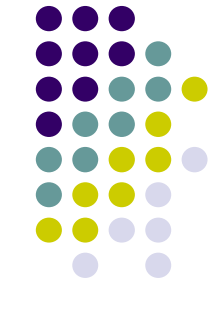
NAT discovery: STUN (3/5)



Per stabilire di che NAT si tratti:

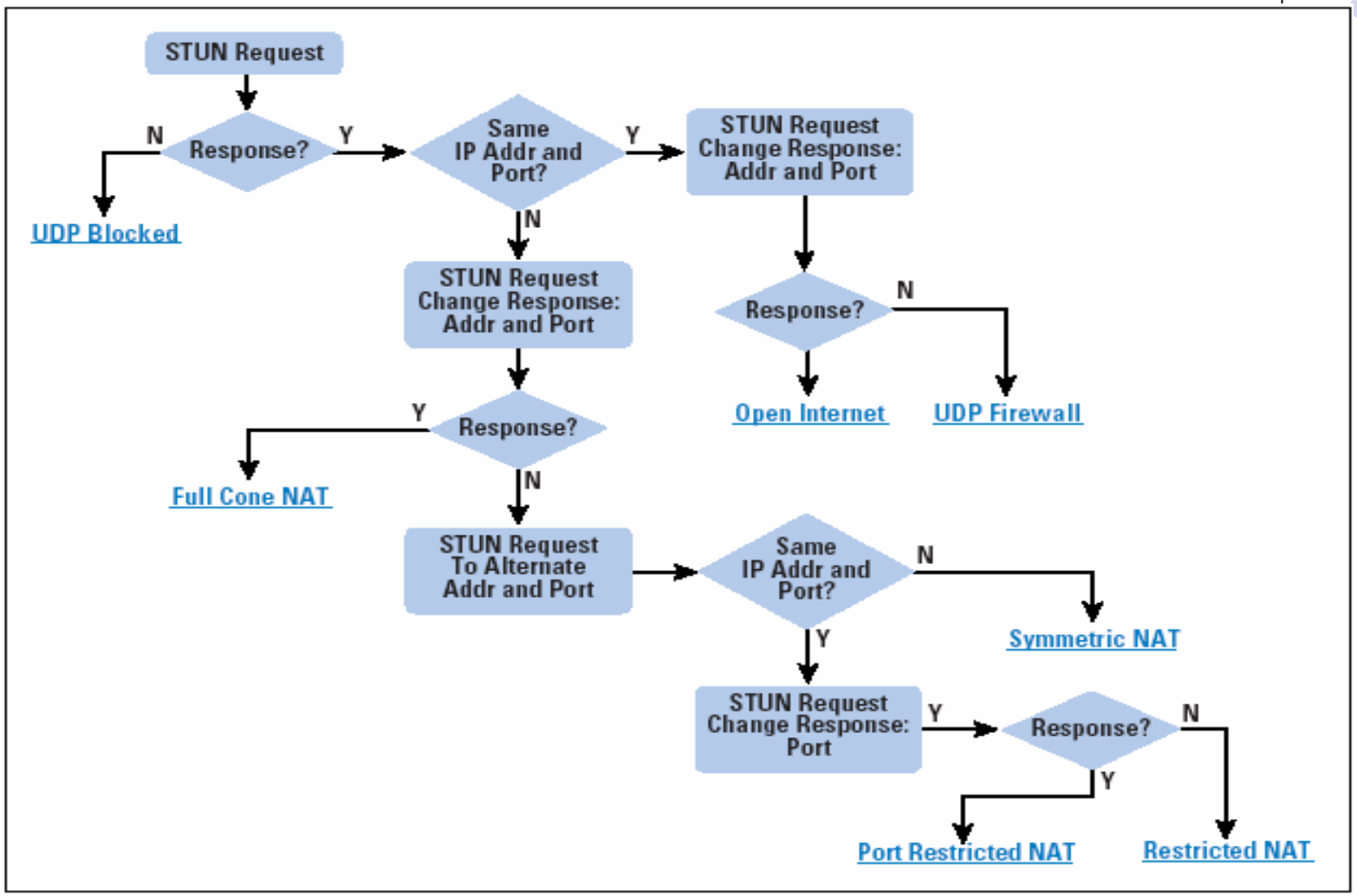
1. Il client invia un pacchetto UDP al server in cui chiede ad esso di rispondere usando indirizzo IP e porto alternativi. Il server risponde inviando un pacchetto UDP al client dal suo indirizzo di trasporto alternativo.
2. Se il client riceve una risposta, conclude che il NAT è di tipo full-cone.
3. Se il client non riceve nessuna risposta, invia un pacchetto UDP verso l'indirizzo di trasporto alternativo del server in cui chiede di conoscere quale indirizzo di trasporto pubblico utilizza per questa nuova comunicazione. Il server risponde inviando al client un pacchetto UDP con le informazioni su indirizzo IP e porto dal quale ha ricevuto la richiesta.

NAT discovery: STUN (4/5)



5. Se l’indirizzo di trasporto pubblico, contenuto nel pacchetto di risposta, è diverso dall’indirizzo pubblico utilizzato nella prima comunicazione con il server, il client può concludere che il NAT è di tipo simmetrico.
6. Se l’indirizzo di trasporto pubblico utilizzato dal client nelle due comunicazioni con il server è lo stesso il client conclude che il NAT è di tipo restricted.
7. Il client invia allora un pacchetto UDP al server in cui gli chiede di rispondere utilizzando lo stesso indirizzo IP, ma cambiando il porto. Il server risponde inviando un pacchetto UDP al client dal porto alternativo.
8. Se il client riceve una risposta conclude che il NAT è di tipo restricted cone. Se nessuna risposta viene invece ricevuta il client può concludere di essere dietro un NAT di tipo port restricted cone.

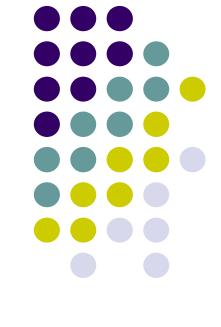
NAT discovery: STUN (5/5)



NAT discovery: TURN (1/2)

- Il protocollo TURN (*Traversal Using Relay NAT*) permette ad un qualsiasi host dietro un NAT o un Firewall di ricevere dati in ingresso su connessioni TCP o UDP.
- Permette la comunicazione tra un host locale ed un peer esterno.
- E' un semplice protocollo di tipo client-server, identico a STUN per quel che riguarda le operazioni più generali.
- Può lavorare sia con UDP che con TCP.
- Provvede ad un meccanismo di mutua autenticazione tra client e server prima della fase di domande-risposte.
- Un client TURN fornisce al server il proprio indirizzo di trasporto privato e chiede al server il suo indirizzo di trasporto pubblico.
- Il server memorizza l'indirizzo di trasporto privato del client dal quale arriva la richiesta e gli restituisce il suo indirizzo pubblico.

NAT discovery: TURN (2/2)



- Il server TURN fa poi in modo che tutti i pacchetti inviati da peer esterni verso l'indirizzo pubblico del client siano in realtà diretti verso se stesso. Quando riceve questi pacchetti (UDP o TCP), il server li inoltra verso il client.
- Quando il client invia pacchetti verso il peer, il server li riceve e li inoltra. Il server TURN lavora quindi da relay per il traffico.
- Il protocollo TURN risulta essere molto pesante sia in termini di carico elaborativo per il server che lo supporta che in termini di traffico.