Terza puntata relativa al laboratorio ping pong. Vogliamo iniziare la fase di sperimentazione. Per questo useremo una serie di script bash. Lo scopo è aggiungere gli script mancanti alla cartella scripts. A questi script è possibile passare dei parametri. L’idea è la seguente: noi vogliamo costruire un diagramma per punti che rappresenti il comportamento della rete: lanciando una volta il ping\_pong otteniamo un punto sul diagramma che vogliamo realizzare. L’automatizzazione che vogliamo ottenere è di lanciare uno script e ottenere tutti i punti del diagramma. Lo script dovrà costruire un makefile tale che basterà usare un make per ottenere un grafico. Per ottenere il diagramma useremo l’applicazione gnuplot. Gnuplot cerca delle colonne in un file, ne usa una per le X e l’altra per le Y. Se in un file ci sono più di due colonne, bisogna specificare quali colonne usare.

Gnuplot permette tanti tipi di rappresentazione, per esempio il grafico a punti, o l’istogramma… un’altra cosa che è possibile fare è quella di definire delle funzioni, dando una formula matematica che ne rappresenta l’andamento (in tal modo sarà gnuplot a calcolare i punti per determinare il grafico).

Dovremo effettuare il calcolo secondo il modello banda latenza.

D(n) = L + n / B. D è il ritardo, n è la dimensione del messaggio. L è la latenza e B è la banda: potremo quindi confrontare sul grafico l’andamento di questa curva teorica con quello del grafico disegnato dai valori reali.

Alcuni dettagli: dai programmi tcp\_ping e udp\_ping vengono estratti sia il valore medio che il valore mediano, quindi, gli script dovranno catturare dimensione del messaggio, valor medio e valore mediano. Questi dati si possono salvare in un file di testo (e formattarlo con Dimensione del messaggio, valore medio e valore mediano per ogni riga). Questo file può essere dato in input a gnuplot (e ciò permetterà di ottenere due grafici con la prima e la terza colonna e la prima e la seconda). È lasciata a noi la decisione di studiare o il valore medio o il valore mediano.

L’output prodotto da tcp\_ting e udp\_ping viene espresso sotto forma di throughput, dove il throughput è uguale al numero di byte inviati per secondo. Per risalire alla formula del modello banda-latenza bisogna risalire al delay dal throughput. Tutte le trasformazioni necessarie saranno da fare attraverso script bash. Una difficoltà che incontreremo è che la shell ci permette di definire variabili, ma ci sono limitazioni sul loro tipo (la variabile principale è la stringa di caratteri, poi in più ci sono i numeri interi, ma non c’è la possibilità di rappresentare floating point). Poiché non si possono salvare variabili in floating point conviene piuttosto chiamare qualche altro programma che possa fare i calcoli con quel tipo di dato. In ambiente UNIX tipicamente ci sarà l’applicazione bc a questo scopo (è un’applicazione che fa da calcolatrice testuale). Una caratteristica interessante di bc è quella di avere risultati con precisione arbitraria (dando un parametro in input che indichi la precisione): la precisione di default è 0, quindi va impostata.

Alla fine, c’è da scrivere una relazione. Deve contenere tutte quelle cose che non sono desumibili dal codice sorgente.

Cambiamo argomento. Abbiamo iniziato a vedere il protocollo ausiliario ICMP. Abbiamo visto che i tre obiettivi principali sono di effettuare il debugging della rete (con Echo request e Echo reply = ping), il test dell’instradamento (attraverso il traceroot, che usa le segnalazioni TTL exceeded e PORT unreachable), segnalazione di eventuali altri errori (come HOST unreachable, …) e l’implemenzione degli alogritmi di instradamento.

L’ICMP supporta l’implementazione del routing. Iniziamo quindi a parlare di questi algoritmi di instradamento. Siamo sempre al livello di Network (3) e riprendiamo l’analisi di IPv4; si ricorda che in IPv4 vengono usati degli indirizzi da 32 bit, che possono concettualmente essere divisi in due parti, subnet e numero di host. L’idea dell’instradamento è la seguente: io parto da una rete locale (LAN sorgente) e voglio arrivare ad un host, che tipicamente farà parte di una LAN diversa (LAN destinazione). Nella LAN sorgente ci sarà un router che si connette alla rete (quindi ad altri router) e poi al router della LAN di destinazione. Non c’è un limita al numero di router che possono costituire la rete -> ci potrebbero essere tanti percorsi alternativi per arrivare a una stessa destinazione. Per instradare il messaggio non è necessario prendere tutti e 32 i bit, è sufficiente prendere la parte di subnet, in quanto quella è la parte comune a tutti gli indirizzi della rete locale di destinazione: di fatto ai router che fanno l’instradamento non interessa distinguere i singoli host, sarà al router che collega la rete locale di destinazione a internet a dover trovare l’host giusto nella sua rete. Il routing IP funziona quindi solo sulla rete geografica (e cioè sulla rete internet) -> da rete locale a rete locale, non da host a host. L’ultimo pezzo del trasporto (da router LAN a host) viene fatto in modo particolare, sfruttando il livello inferiore del protocollo, cioè il DATA LINK (2) attraverso l’indirizzo MAC.

Le esigenze che abbiamo quando vogliamo instradare un datagramma è di recapitare il messaggio il più velocemente possibile (quindi se per trovare la strada più corta ci vuole tanto tempo non conviene): l’algoritmo di instradamento deve essere il più veloce possibile. Il modo più veloce in assoluto è sapere già a priori su quale router trasmettere il datagramma per farlo arrivare alla subnet di destinazione. Quindi, i router si basano su delle routing table, che vengono calcolate in anticipo (da un algoritmo preciso ma che prenderebbe troppo tempo se calcolato al momento), così quando arriva un datagramma il router può usare queste “Tabelle di Forwarding” per ritrasmettere subito il messaggio. Queste tabelle sono composte da delle coppie indirizzo-porta di uscita (il router avrà delle porte di uscita che indicano le direzioni in cui trasmettere il messaggio): la tabella sancisce che se arriva un messaggio indirizzato a x indirizzo, il messaggio va instradato da y porta di uscita per fare il prossimo HOP. Ogni router ovviamente dovrà avere la propria tabella di forwarding per rendere questo possibile. Quanto possono essere lunghe queste tabelle (si noti che non si sa quant’è la subnet mask della LAN di destinazione finchè non si è arrivati)? Si utilizza solo un sottoinsieme di bit, sulla base delle esigenze del router (perché non si conosce la subnet mask) -> le tabelle di forwarding vengono costruite associando un preambolo a ciascuna porta di uscita. Per esempio, nel caso del penultimo router, quello che è collegato a quello della LAN, ha senso usare tutti (o quasi) i bit dell’indirizzo come preambolo e, man mano che ci si allontana dalla destinazione -e si va indietro nel percorso- si riduce il numero di bit di preambolo (il primo router potrebbe tener conto solo dei primi 2 bit -> la soluzione non è perfettamente ottimale, ma è veloce): in questo modo si limitano le dimensioni della tabella di forwarding. Basta che ogni router contenga indirizzi di preambolo completi/quasi completi solo per le reti locali che ha a un hop di distanza, e indirizzi di preambolo via via più generici a seconda di quanto è lontano dalle reti che iniziano con quel preambolo.

Come fanno i routers a costruirsi queste tabelle di forwarding? Si sfrutta il protocollo ICMP, che permette ai routers di scambiarsi informazioni tramite datagrammi di questo tipo di protocollo. Lo scambio di questi datagrammi permette ai router di conoscere in che punto della rete essi si trovano e in modo che possano costruire le proprie tabelle di forwarding.

Di algoritmi per costruire la strada più breve ne esistono due famiglie principali: una delle due sfrutta le informazioni di tipo globale, l’altra sfrutta informazioni di tipo locale.

Vediamo prima gli algoritmi di tipo locale: supponiamo che io voglia aggiungere un nuovo router aumentare le prestazioni della rete. Questo router dovrà connettersi ad altri router (supponiamo di dargli 4 canali fisici di comunicazione, con altrettanti router); nel momento in cui lo accendiamo per la prima volta il router non ha nessuna tabella di forwarding, dunque non può instradare nulla. Usando l’ICMP il nuovo router si presenta ai suoi vicini e a seguito del suo messaggio i suoi vicini si rendono conto della sua presenza e del suo indirizzo IP (ipotizziamo che sia 1111). I vicini del nuovo router aggiungono informazioni alle loro tabelle di forwarding, aggiungendo l’indirizzo IP completo della nuova macchina nella tabella: adesso il nuovo router può ricevere datagrammi, ma non li può instradare. I vicini del nuovo router, però, diffondono il nuovo aggiornamento (che comprende la presenza del nuovo router) a tutti i loro vicini, in modo che aggiornino le tabelle anche loro. Questo scambio di informazioni fa sì che il nuovo router si renda conto di quali sono i suoi vicini (e potrebbe iniziare a instradare i messaggi indirizzati verso essi), tuttavia l’algoritmo non si ferma qui; dopo aver costruito la mappa dei router a distanza 1 il nuovo router può inviare altri router ICMP per capire quali sono i router a distanza 2, poi ripetere la procedura per quelli a distanza 3 e così via, riducendo man mano la quantità di bit di rappresentazione degli indirizzi preambolo. Quando il router ha un’idea generale dei percorsi di instradamento che può far prendere ai datagrammi, può preferire un percorso rispetto ad un altro sulla base della banda o della latenza.

Vantaggi e svantaggi dell’algoritmo locale. Questa tecnica funziona ragionevolmente bene su delle reti di grandi dimensioni, perché permette di creare tabelle abbastanza piccole basandomi sullo scambio di datagrammi tra pochi router adiacenti; lo svantaggio è che non è detto che si arrivi a una soluzione ottima (gli instradamenti realizzati con tabelle di questo tipo potrebbero non essere i più brevi possibili).