SAPIENZA
UNIVERSITÀ DI ROMA

Generazione e visualizzazione grafica di traffico di reti

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Laurea Triennale in Informatica

Francesco Pannozzo
Matricola 699427

Relatore
Prof. Daniele De Sensi

Anno Accademico 2023/2024

Tesi non ancora discussa

Generazione e visualizzazione grafica di traffico di reti
Relazione di tirocinio. Sapienza Università di Roma

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: francesco.pannozzo@libero.it

Dedicato alla
mia famiglia

Sommario

Questa relazione descrive il lavoro di tirocinio interno svolto presso l'università La Sapienza, concretizzato nella realizzazione di un progetto volto a realizzare un software per poter visualizzare in forma grafica l'andamento del traffico di una rete. Il progetto ha come obiettivo di mostrare il traffico di rete al variare del tempo e ciò viene raggiunto tramite grafiche e animazioni generate programmaticamente. L'idea dell'ambito di tirocinio nasce dalla volontà di sperimentare una realizzazione front-end tramite la libreria Manim, un motore di animazioni per video matematici esplicativi..

Indice

Capitolo 1

Introduzione

Nel mondo le reti informatiche sono oramai un concetto ben istanziato nella colletività, la loro presenza è soverchiante e si dirama nei più disparati settori. Basti pensare già alle reti PAN (Personal Area Network) le quali connettono dispositivi personali entro pochi metri e che ognuno di noi usa abitualmente nella propria casa, alle reti LAN (Local Area Network), anch'esse presenti nelle nostre case così come in uffici o edifici scolastici, le reti dei datacenter fino a giungere alla rete globale internet, la quale è creatrice a sua volta di paradigmi come può essere l'internet of things. Le reti informatiche sono impiegate nei più vari settori come l'istruzione, in cui le reti sono cruciali nelle scuole e nelle università per avere accesso a risorse educative o sfruttare l'e-learning, i servizi pubblici governativi e sanitari, nel settore ludico e multimediale come il gioco online e l'attuale streaming di contenuti multimediali: insomma, le reti informatiche sono di fatto una presenza piena e diffusissima ed è estremamente difficile riuscire a immaginare il mondo come lo vediamo oggi senza questa tecnologia. Con l'aumentare delle funzionalità legate alle reti, così come i dispositivi collegati a esse, capire cosa succede al loro interno, come si muovono i dati, è quindi di cruciale importanza, tramite l'analisi dei dati che vi fruiscono è possibile fare diagnostica, per quanto riguarda un discorso di monitoraggio, ma anche è possibile applicare le analisi in un ambito didattico e accademico. Capire cosa sta succedendo in una rete in modo immediato e visivo è lo scopo di questo progetto, il quale punta a mostrare, in modo grafico, l'andamento del traffico di una rete.

1.1 Ambito del tirocinio

Il progetto fa parte del percorso di tirocinio interno intrapreso presso l'Università La Sapienza di Roma. L'argomento su cui verte il progetto è la realizzazione di un visualizzatore grafico dell'andamento del traffico di una rete, basato su animazioni programmatiche. Il tool permette di visualizzare gli switch rappresentanti i vari endpoints e i link che li collegano i quali vengono colorati tramite animazioni nel tempo in base al traffico di rete precedentemente analizzato. Nel tool è presente anche una parte generativa di traffico di rete, una creazione di traffico fittizia di vitale importanza ai fini di testing. Il progetto è suddiviso quindi in tre parti; una parte che si occupa della generazione di una configurazione network e relativo traffico

di rete, una parte che analizza il traffico di rete calcolandone le medie percentuali di intervalli di tempo e aggiornate ripetutamente e infine una parte che visualizza il traffico analizzato dall'analizzatore producendo una rappresentazione video.

1.2 Motivazioni

L'idea di sviluppare un visualizzatore grafico di traffico di rete è nata, in sede di proposta, dal Professore Daniele De Sensi, relatore del tirocinio, e dalla mia volontà di sviluppare un'applicazione avente il front-end come focus dell'esperienza. Nel mio personale corso di studi presso il Dipartimento di Informatica non ho avuto modo di studiare e approfondire un discorso legato al front-end, per cui la volontà di intraprendere questo percorso nasce in primis da un forte interesse verso questo aspetto dell'informatica e in secondo luogo per un completamento di formazione professionale personale.

1.3 Stato dell'arte

L'esigenza di analisi di reti informatiche ha portato alla luce svariati tool che permettono appunto di analizzare cosa avviene in una rete, di studiarne i dati statistici e di visualizzare graficamente determinati scenari. L'universita americana Johns Hopkins[7] ha stilato una lista di software per la visualizzazione e analisi di reti[8]:

- **Gephi[3]:** Gephi è il software leader di visualizzazione ed esplorazione per tutti i tipi di grafici e reti ed è open source. Le sue caratteristiche includono l'analisi esplorativa dei dati mediante manipolazioni di reti in tempo reale, analisi dei collegamenti per rivelare le strutture sottostanti delle associazioni tra oggetti, analisi dei social network per la creazione di connettori di dati sociali per mappare le organizzazioni della comunità e le reti di piccoli mondi, analisi della rete biologica per rappresentazione di modelli di dati biologici ed esportazione e creazione poster erp promuovere e divulgare il lavoro scientifico con mappe stampabili di alta qualità.

- **Cytoscape[2]:** è una piattaforma software open source per visualizzare reti complesse e integrarle con qualsiasi tipo di dati. Consiste in una piattaforma per visualizzare reti di interazioni molecolari e percorsi biologici, potendo integrare queste reti con annotazioni, profili di espressione genica e altri dati. Originariamente progettato per la ricerca biologica, ora è una piattaforma generale per l'analisi e la visualizzazione di reti complesse.

- **GraphVis[4]:** è un software di visualizzazione di grafici open source. Caratteristiche, I programmi di layout Graphviz accettano descrizioni di grafici in un semplice linguaggio di testo e creano diagrammi in formati utili, come immagini e SVG per pagine web; PDF o Postscript per l'inclusione in altri documenti; o visualizzare in un browser grafico interattivo. Graphviz ha molte funzionalità utili per diagrammi concreti, come opzioni per colori, caratteri, layout di nodi tabulari, stili di linea, collegamenti ipertestuali e forme personalizzate.

- **igraph[5]:** è una collezione di librerie per creare, manipolare grafici e analizzare ponendo l'enfasi nell'efficienza, portabilità e facilità d'uso. Igraph è open source e gratuito e può essere programmato in R, Python, Mathematica e C/C++

- **UCINET6[15]:** è un pacchetto software per l'analisi dei dati dei social network. UCINET viene fornito con il tool di visualizzazione di rete NetDraw. Può leggere e scrivere una moltitudine di file di testo diversamente formattati, nonché file Excel. I metodi di analisi dei social network includono misure di centralità, identificazione di sottogruppi, analisi di ruolo, teoria dei grafi elementari e analisi statistica basata sulla permutazione. Inoltre, il pacchetto dispone di potenti routine di analisi delle matrici, come l'algebra delle matrici e la statistica multivariata.

- **SocNetV[13]:**è un'applicazione software gratuita multipiattaforma per l'analisi e la visualizzazione dei social network. Tra le caratteristiche principali troviamo il poter disegnare i social network, caricare i campi da un file supportato (GraphML, GraphViz, Adjacency, EdgeList, GML, Pajek, UCINET, ecc.), personalizzare attori e collegamenti tramite sistema punta e clicca, analizzare le proprietà dei grafici e dei social network, produrre report HTML e incorporare layout di visualizzazione di rete

- **Pajek[9]:** è un software per la visualizzazione e l'analisi delle reti. La sua forza risiede nel poter analizzare reti complesse potendo arrivare fino a un miliardo di vertici. L'analisi e la visualizzazione vengono eseguite utilizzando sei tipi di dati: rete (grafico), partizione, vettore, cluster (sottoinsieme di vertici), permutazione (riordinamento dei vertici, proprietà ordinali); e gerarchia (struttura generale ad albero sui vertici).

1.4 Contributi

Da un punto di vista grafico e quindi di visualizzazione, la maggior parte degli strumenti sopra elencati, permette una certa forma di personalizzazione nella disposizione dei nodi, sia automaticamente attraverso algoritmi di layout sia manualmente, permettendo agli utenti di spostare i nodi per ottimizzare la visualizzazione o per enfatizzare certi aspetti della rete. Tuttavia, la possibilità di avere animazioni dinamiche che mostrino l'andamento del traffico nel tempo, mostrando la variazione del colore in base alla quantità dello stesso, risulta essere una caratteristica meno comune nei software di analisi di rete, nello specifico:

- Gephi: Non supporta nativamente animazioni dinamiche basate su traffico in tempo reale. Tuttavia, la sua flessibilità e la capacità di aggiungere plugin potrebbero permettere implementazioni personalizzate.

- Cytoscape: Anche se fortemente orientato all'analisi statica, plugin o estensioni potrebbero aggiungere capacità simili.

- GraphVis (Graphviz): Principalmente orientato verso la visualizzazione statica; non supporta direttamente animazioni dinamiche dei link basate sul traffico.

- igraph: Come libreria di analisi, non è orientato verso la visualizzazione in tempo reale o animazioni dei link basate su traffico nel suo utilizzo standard.
- UCINET (con NetDraw): Focalizzato sull'analisi statica di reti sociali; non supporta animazioni dinamiche in base al traffico.
- SocNetV: Orientato all'analisi statica e alla visualizzazione; non è progettato per visualizzare animazioni dinamiche basate sul traffico.
- Pajek: Simile agli altri, è più un tool per l'analisi statica e la visualizzazione di grandi reti, senza un supporto diretto per animazioni dei link basate su traffico.

In questo contesto, l'inserimento di una caratteristica che permetta di fare quanto premesso come base del progetto di tirocinio, risulta particolarmente indicata nel contribuire a fornire una soluzione visiva come strumento aggiuntivo di analisi di una rete, di debugging e anche come strumento didattico. La possibilità di avere un riscontro visivo istantaneo di cosa avviene nel tempo in una rete, a livello di traffico, può dare immediato feedback nel caso ci fosse un problema di congestione in un punto nevralgico, oppure mostrare parti di rete libere dove poter studiare un reindirizzamento dello stesso, volto a ottimizzare le prestazioni. A livello didattico ciò si potrebbe mostrare per presentazioni così come per didattica tramite banalmente spiegazioni. Insomma i benefici derivanti da una rappresentazione del generale sono evidenti e ciò può essere di grosso aiuto nell'analisi così anche solo come semplice rappresentazione del traffico di rete, nonchè uno strumento complementare a quanto già presente in circolazione.

1.5 Base di partenza del progetto

Il progetto è partito da zero, si basa sullo sviluppo totalmente nuovo dell'applicazione ed è stato tutto idealizzato e pianificato in sede di proposta. Come approfondirò in seguito, nella sezione dedicata alla tecnologia impiegata, il progetto non è l'unica cosa a essere partita da zero, poichè il linguaggio scelto per sviluppare l'applicazione è Python[11], linguaggio non incluso nel mio personale percorso di studi e che ho dovuto necessariamente studiare da zero per poter affrontare il percorso di tirocinio.

Capitolo 2

Contesto di sviluppo: le tecnologie impiegate

Il progetto, a livello di tecnologia impiegata, pone le fondamenta su tre aspetti che andremo a elencare di seguito, descrivendo le varie motivazioni che hanno spinto a sceglierli.

2.1 Python

Uno dei primi aspetti di cui si è tenuto conto è stata la scelta del linguaggio di programmazione che, come accennato precedentemente, è Python. Ci sono diversi validi motivi per cui puntare su questa tecnologia; in primis è materia di insegnamento alla facoltà di Informatica de La Sapienza, ciò ha quindi una forte valenza accedemica, in secondo luogo risulta essere il linguaggio più usato al mondo, ad affermarlo è l' Institute of Electrical and Electronics Engineers (IEEE)[6][1] un'associazione internazionale di scienziati professionisti con l'obiettivo della promozione delle scienze tecnologiche. Il linguaggio ha molte caratteristiche ottime, come una sintassi semplice e leggibile che lo rende facile da imparare e semplice da usare per gli sviluppatori esperti accorciando di gran lunga i tempi di sviluppo, una grande versatilità per poter essere usato in ambiti diversi come l'intelligenza artificiale, il web development, data analysys e molto altro, un ampio supporto delle librerie, due delle quali usate proprio nel progetto (di cui ne parlerò a breve), una grande comunità in cui trovare facilmente risorse, tutorial e supporto, una interoperabilità che permette un'ottima integrazione con altre tecnologie e altri linguaggi, orientato agli oggetti volto a facilitare la gestione del codice e migliora il riuso, scalabile e di facile integrazizone.

2.2 La libreria Manim

Il secondo aspetto risiede nella scelta della libreria Manim[14], che viene definita come "Animation engine for explanatory math videos". Lo scopo di Manim è quindi quello di animare concetti tecnici legati alla matematica e si affida alla semplicità di Python per generare animazioni in modo programmatico. Manim può produrre anche immagini e gif, ma è nella produzione di video che splende, in questo modo è possibile progettare animazioni e renderle visibili in movimento, coprendo figure

algebriche, grafici cartesiani, grafi e molto altro[14]. La libreria offre molta liberà sui risultati che si vogliono ottenere, può renderizzare singole immagini, gif e da molte opzioni per quanto riguarda l'output video, fornendo la possibilità di renderizzarli nei formati più comuni con la possibilità di personalizzazione del framerate:

- 480p (SD): 854 x 480
- 720p (HD): 1280 x 720
- 1080p (HD): 1920 x 1080
- 1440p (2K): 2560 x 1440
- 2160p (4K): 3840 x 2160

Manim viene impiegato principalmente per presentazioni che implichino aspetti matematici, la sfida del progetto è stata quella di cercare di sfruttare le potenzialità della libreria e renderle al servizio di uno strumento di analisi sul traffico di reti, una sfida vinta come potremo vedere in seguito. Il terzo aspetto tecnologico riguarda l'aspetto di gestione dei dati. Un visualizzatore grafico di traffico di reti ha bisogno principalmente di due insiemi di informazioni importanti; uno riguarda tutte le informazioni che riguardano il come è costruita la rate, parliamo quindi degli endpoint quali sono gli switch e conseguenti informazioni annesse, pensiamo ad esempio all'indirizzo di rete, un nome identificativo e così via, ma parliamo anche dei link che collegano i vari endpoints, con la necessità di tenere traccia delle loro capacità trasmissive, la tipologia di rete, se ha una struttura a grafo completo, mesh, torus o disposizione libera e molte altre informazioni che discuteremo in seguito.

2.3 I formati Json e Yaml

L'altro insieme di informazioni deriva dal traffico vero e proprio, rendendo quindi necessario un sistema di mantenimento dei dati legati ai pacchetti trasmessi. L'analisi di questi due insiemi di informazioni ha portato alla valutazione di tre sistemi per la strutturazione di dati; json[10], yaml[16] e csv[12]. Dopo attenta anlisi si è deciso di adottare il formato json per strutturare e memorizzare i dati legati al traffico, con la possibilità di scegliere anche il formato yaml, mentre per i dati relativi alla descrizione della rete il compito è stato affidato esclusivamente a yaml, csv è stato scartato. Perchè csv è stato scartato? Sebbene csv rappresenti una valida alternativa tenendo conto di aspetti prestazionali, essendo un formato molto veloce da analizzare, da leggere e scrivere, tuttavia lo diventa meno quando c'è bisogno di strutturare maggiormente i dati con strutture più complesse dalla semplice forma tabellare, tipicamente usate nei database. L'idea di scartarla, sia per strutturare i dati della rete che per quelli del traffico, deriva principalmente dai seguenti motivi:

- **Leggibilità:** uno dei primi intenti del progetto era di rendere l'applicazione il più leggibile possibile, questo perchè si è fortemente voluto attribuirne anche scopi di debugging e didattici, laddove avere una certa leggibilità è più che ragionevole.

- **Strutture complesse:** sicuramente il motivo più importante. Con csv non è possibile rappresentare strutture complesse, parliamo ad esempio di oggetti all'interno di altri oggetti. Sebbene per come sia ora strutturato il progetto una rappresentazione cvs è ancora possibile, ciò potrebbe non esserlo in futuro nell'ottica di espansione del progetto

Per esplicare meglio il concetto di struttura complessa non fattibile è possibile focalizzarsi sul seguente esempio. Segue la rappresentazione di un pacchetto di rete così come viene utilizzato nel progetto, in questo caso una lista contenente un solo elemento:

```
[
    {
        "A": 1,
        "B": 2,
        "t": "2024-03-22 12:30:00",
        "d": 1518
    }
]
```

fig. 2.1. pacchetto di rete rappresentato in json

Dove A e B sono gli endpoints interessati, t è il timestamp di creazione pacchetto e d la dimesione del payload in bytes(). In csv questa struttura è rappresentabile come segue:

```
A,B,t,d
2,1,2024-03-22 12:30:00,1518
2,1,2024-03-22 12:30:00,1518
```

fig. 2.2. pacchetti di rete rappresentati in csv

Tuttavia se si dovesse rendere questa struttura più complessa, avremmo problemi a realizzarla in csv, basterebbe l'aggiunta di un campo che a sua volta necessita di informazione strutturata, come ad esempio inserire le informazione dell'header:

```
    [
        {
          "A": 1,
          "B": 2,
          "t": "2024-03-22 12:30:00",
          "d": 4000,
          "ip_header": {
            "version": 4,
            "ihl": 5,
            "type_of_service": 0,
            "total_length": 300,
            "identification": 98765,
            "flags": {
              "reserved_bit": false,
```

```
            "dont_fragment": true,
            "more_fragments": false
          },
          "fragment_offset": 0,
          "time_to_live": 64,
          "protocol": 6,
          "header_checksum": "2B5A",
          "source_address": "192.168.1.1",
          "destination_address": "192.168.1.2"
        }
      }
    ]
```

fig. 2.3. pacchetto di rete maggiormente strutturate in json

In questa struttura abbiamo ben due oggetti nidificati, ip header e flags. Questo tipo di struttura è difficilmente replicabile in csv che si presta maggiormente per strutture piatte mentre json e yaml permettono molta più libertà di strutturazione. La scelta del formato json per rappresentare i dati del traffico di rete è quindi legata alla possibilità di espandere la rappresentazione con strutture più complesse in ottica di espansioni future del progetto, ma è indubbiamente legata alle prestazioni che questo formato riesce a dare. Json è un formato ampiamente supportato e la sua semplicità permette di ottenere risultati ottimi per le operazioni di parsing, lettura e scrittura per strutture aventi grandi quantità di dati. Sebbene l'applicazione supporti sia json che yaml per quanto riguarda il memorizzare i dati legati al traffico, la scelta preferenziale ricade su json. Questo perchè, nonostante json produca files di dimensioni maggiori rispetto a yaml, la schiacciante velocità di elaborazione di json rende la creazione di un file di dimensioni maggiore ampiamente giustificabile, considerando soprattuto la natura del progetto in cui non vi è una criticità d'uso nella dimensione dei files. Come vedremo in seguito, i test eseguiti sui tempi di lettura e scrittura rispettivamente della stessa struttura ricreata sia in json che in yaml, sapranno ben dimostrare quanto appena affermato. Infinie giungiamo ai motivi per cui si è scelto invece esclusivamente yaml per rappresentare i dati relativi alle informazioni che descrivono la rete. Le motivazioni consistono sempre nella volontà di garantire un formato che si presti a espansioni future e quindi che possano richiede strutturazioni complesse, ma soprattutto, in questo caso specifico, nell'alta leggiblità che yaml offre. Generalmente i dati che servono a descrivere una rete non sono mai paragonabili a quelli necessari per registrare tutto il traffico, si voleva dare quindi uno strumento molto chiaro da leggere per fare in modo che la configurazione di rete fosse sempre molto chiara, intuitiva e di facile accesso, sia in scrittura che in lettura. Per dare una stima di grandezza, il file di configurazione network che produce il generatore di pacchetti, basato su 50 switch collegati come un grafo mesh pesa appena 9 KB. Seguirà la descrizione approfondita del progetto, principalmente suddiviso in tre parti; la prima parte riguarda la generazione di traffico di rete fittizio, la seconda è l'analizzatore dei dati di configurazione di rete e relativo traffico mentre la terza parte riguarda il visualizzatore grafico vero e proprio.

Capitolo 3

Generatore di configurazione e traffico di rete

Il progetto è composto da tre anime, una di esse è un generatore di traffico di rete e relativa configurazione. Lo scopo del generatore è da ricercare principalmente in motivazioni di testing, ma può essere usato anche per didattica per scopi esemplicativi. La configurazione di rete consiste nella produzione di una struttura dati che descriva le caratteristiche della rete, la quale verrà memorizzata in un file dal nome network e avente estensione yaml, mentre per quanto riguarda il traffico, crea una struttura che consiste in una lista di pacchetti. Lo script permette di eseguire quanto descritto mettendo a disposizione due modalità, auto e user. Entrambe le modalità produrranno due files, network.yaml conterà le caratteristiche della rete e il file packets.json (o packets.yaml a seconda della scelta) conterrà il traffico vero e proprio di tutti i pacchetti generati dalla simulazione. La modalità auto chiede all'utente il numero di switch, la capacità dei link e la tipologia del grafo con il quale rappresentare la rete (completo, mesh, torus) e imposterà in modo del tutto automatico la disposizione degli switch in base alla scelta del grafo effettuata. Per poter operare, in entrambe le modalità, lo script ha bisogno di leggere il file sim_setup.yaml, il quale contiene le informazioni necessarie per la configurazione. Il file di setup è impostato come segue:

```
averageDelta: 1000
updateDelta: 100
startSimTime: 2024-03-22 12:30:00
simTime: 5
packetSize: 4000
colorblind: "no"
dotsSize: "fixed"
trafficVariation: random
packetsFile: json
```

fig. 3.1. esempio di setup file

Andremo ora a spiegare il significato dietro ogni voce:

- **averageDelta:** rappresenta l'intervallo temporale in millisecondi delle medie percentuali di traffico da calcolare nell'analizzatore di traffico.

- **updateDelta:** rappresenta ogni quanti millisecondi si deve aggiornare la media averageDelta
- **startSimTime:** è il datetime dell'inizio della generazione nel formato YY:MM:DD HH:MM:SS, inoltre è possibile specificare anche un eventuale tempo avente millisecondi da specificare come ad esempio 12:30:00.500
- **simTime:** è la durata della simulazione in secondi
- **packetSize:** è la dimensione in bytes di un pacchetto, i pacchetti nella simulazione avranno questa dimensione
- **colorblind:** è una stringa "yes" o "no" che abilita se posta su "yes" una visualizzazione compatibile per persone daltoniche di cui parleremo successivamente quando discuteremo della parte relativa al visualizzatore grafico.
- **trafficVariation:** è la variazione di traffico per secondo che si desidera impostare; può essere il valore "random" oppure uno dei seguenti [5, 10, 20, 25, 50] e di conseguenza determinerà la variazione percentuale di traffico che avviene ogni secondo di simulazione. Il valore "random" sceglie casualmente una percentuale ogni secondo con un valore che va da 0 a 100
- **packetsFile:** specifica quale tecnologia si vuole utilizzare per il file packets, si può scegliere tra json e yaml

La generazione di pacchetti è calcolata sulla base della capacità dei link fornita e su un valore casuale di percentuale di traffico che varia ogni secondo, per esempio avendo 6 links su 3 secondi di simulazione e il parametro "trafficVariation" settato a random potremmo avere delle assegnazioni di percentuali di traffico come le seguenti:

```
endpoints: [1, 2], sim second: 0, trafficPerc: 65
endpoints: [1, 3], sim second: 0, trafficPerc: 9
endpoints: [2, 4], sim second: 0, trafficPerc: 23
endpoints: [2, 5], sim second: 0, trafficPerc: 67
endpoints: [3, 6], sim second: 0, trafficPerc: 7
endpoints: [3, 7], sim second: 0, trafficPerc: 87
endpoints: [1, 2], sim second: 1, trafficPerc: 86
endpoints: [1, 3], sim second: 1, trafficPerc: 26
endpoints: [2, 4], sim second: 1, trafficPerc: 13
endpoints: [2, 5], sim second: 1, trafficPerc: 13
endpoints: [3, 6], sim second: 1, trafficPerc: 39
endpoints: [3, 7], sim second: 1, trafficPerc: 65
endpoints: [1, 2], sim second: 2, trafficPerc: 31
endpoints: [1, 3], sim second: 2, trafficPerc: 54
endpoints: [2, 4], sim second: 2, trafficPerc: 11
endpoints: [2, 5], sim second: 2, trafficPerc: 20
endpoints: [3, 6], sim second: 2, trafficPerc: 17
endpoints: [3, 7], sim second: 2, trafficPerc: 46
```

fig. 3.2. esempio di variazione traffico casuale

Una volta lanciato il generatore avremo le seguenti possibilità di scelte:

- **auto:** la modalità automatica, provvederà a disporre i nodi (switch) di rete in modo del tutto automatico. Dopo aver scelto la modalità auto si dovranno inserire i seguenti parametri via prompt:
  - numero di switch
  - capacità dei link
  - tipologia della rappresentazione grafica della rete (grafo completo, mesh, torus)
- **user:** una modalità in cui l'utente può personalizzare la configurazione di rete

La modalità user necessita di un file "custom_graph.yaml" con i parametri necessari a descrivere la rete del quale si vuole analizzare il traffico. Con questa modalità l'utente ha completa libertà nel personalizzare la rete ed è tenuto quindi a descriverne ogni suo aspetto. Il custom_graph.yaml prevede la struttura seguente, con possibili varianti:

```
---
data:
  graphType: mesh
  coordinates:
    - [1, 0, 2, 0]
    - [3, 0, 4, 5]
    - [6, 7, 8, 9]
  switches:
    1:
      ip: "123.123.123.0"
      switchName: Anthem
    2:
      ip: "123.123.123.1"
      switchName: Beta
    3:
      ip: "123.123.123.2"
      switchName: Cyber
    4:
      ip: "123.123.123.3"
      switchName: Dafne
    5:
      ip: "123.123.123.4"
      switchName: Eclipse
    6:
      ip: "123.123.123.5"
      switchName: Fox
    7:
      ip: "123.123.123.6"
      switchName: Gea
```

```
      8:
        ip: "123.123.123.7"
        switchName: H20
      9:
        ip: "123.123.123.8"
        switchName: Italy
    links:
      - { linkCap: 10,  endpoints: [1, 3] }
      - { linkCap: 100, endpoints: [1, 6] }
      - { linkCap: 10,  endpoints: [3, 6] }
      - { linkCap: 10,  endpoints: [4, 8] }
      - { linkCap: 10,  endpoints: [5, 9] }
      - { linkCap: 100, endpoints: [3, 5] }
      - { linkCap: 10,  endpoints: [6, 9] }
      - { linkCap: 100, endpoints: [4, 5] }
      - { linkCap: 10,  endpoints: [6, 7] }
      - { linkCap: 10,  endpoints: [7, 8] }
      - { linkCap: 100, endpoints: [8, 9] }
      - { linkCap: 10,  endpoints: [2, 4] }
      - { linkCap: 10,  endpoints: [2, 8] }
    phases:
      2024-01-01 00:00:01: "phase1"
      2024-01-01 00:00:02: "phase2"
```

fig. 3.3. esempio di custom file per la configurazione

Descriviamo i vari parametri:

- **graphType:** identifica la tipologia del grafo da rappresentare, sono disponibili tre opzioni:
  - **mesh:** l'algoritmo individua in modo automatico gli archi (i link) che collegano i nodi (gli switch) adiacenti tra loro presenti nella matrice coordinates
  - **torus:** esegue lo stessa procedura usate per mesh e in addizione collega tra loro i nodi che si trovano alle estremità della matrice
  - **graph:** è la modalità più libera, collega gli switch tramite i link forniti dall'utente, indipendentemente da dove vengono collocati
- **coordinates:** rappresenta le coordinate dei vari switch, i quali vanno rappresentati con un id numerico che va da 1 a 1000. C'è una precisa motivazione di design per questa scelta ed è legata a una rappresentazione grafica ottimale del visualizzatore grafico. Gli zeri invece rappresentano uno spazio vuoto in cui non è presente uno switch.
- **links:** rappresenta i link della rete i quali possono essere specificati con i campi:
  - **linkCap:** esprime la capacità del link in Mbps

  - **endpoints:** esprime gli endpoints collegati al link

- **phases:** rappresenta le fasi temporali che accompagnano la durata dell'attività di rete, sono identificate tramite timestamp che ha come valore la descrizione della fase che parte dal timestamp stesso.

Come si può notare dalla figura 3.4, gli switch sono identificati tramite un valore numerico, questa è una precisa scelta di design e mira a mantenere il concetto di leggibilità sempre presente. In questo caso la scelta deriva dal semplificare e rendere immediatamente chiaro il campo coordinates; così facendo si rende intuitivo il posizionamento degli switch nello spazio (rappresentato da una matrice quadrata), e l'occhio identifica immediatamente la rappresentazione della disposizione. Qualora la capacità dei link sia uguale per tutti è possibile insererire il campo linkCap per poter così evitare di avere lo stesso valore nei vari link rappresentati nel campo links, potendo rappresentare solo gli endpoints:

```
data:
  graphType: torus
  coordinates:
    - [1, 0, 2, 0]
    - [3, 0, 4, 5]
    - [6, 7, 8, 9]
  linkCap: 10
  switches:
    1:
      ip: "123.123.123.0"
      switchName: A
    2:
      ip: "123.123.123.1"
      switchName: B
    3:
      ip: "123.123.123.2"
      switchName: C
    4:
      ip: "123.123.123.3"
      switchName: D
    5:
      ip: "123.123.123.4"
      switchName: E
    6:
      ip: "123.123.123.5"
      switchName: F
    7:
      ip: "123.123.123.6"
      switchName: G
    8:
      ip: "123.123.123.7"
      switchName: H
```

```
    9:
      ip: "123.123.123.8"
      switchName: I
  links:
    - [1, 3]
    - [1, 6]
    - [3, 6]
    - [4, 8]
    - [5, 9]
    - [3, 5]
    - [6, 9]
    - [4, 5]
    - [6, 7]
    - [7, 8]
    - [8, 9]
    - [2, 4]
    - [2, 8]
  phases:
    2024-01-01 00:00:01: "phase1"
    2024-01-01 00:00:02: "phase2"
```

fig. 3.4. esempio di custom file con capacità uguali a tutti i link

Un'altra possibilità è quella di lasciare che il programma ricavi in automatico i link, in questo caso basterà specificare solo linkCap che sarà uguale per tutti i link:

```
    ---
data:
  graphType: mesh
  coordinates:
    - [1, 2, 3]
    - [4, 5, 6]
    - [7, 8, 9]
  linkCap: 10
  switches:
    1:
      ip: "123.123.123.0"
      switchName: Anthem
    2:
      ip: "123.123.123.1"
      switchName: Beta
    3:
      ip: "123.123.123.2"
      switchName: Cyber
    4:
      ip: "123.123.123.3"
      switchName: Dafne
    5:
```

```
      ip: "123.123.123.4"
      switchName: Eclipse
    6:
      ip: "123.123.123.5"
      switchName: Fox
    7:
      ip: "123.123.123.6"
      switchName: Gea
    8:
      ip: "123.123.123.7"
      switchName: H20
    9:
      ip: "123.123.123.8"
      switchName: Italy
  phases:
    2024-01-01 00:00:01: "phase1"
    2024-01-01 00:00:02: "phase2"
```

fig. 3.5. esempio di custom graph file con link di eguale capacità

Come si può notare, questo tipo di approccio concede una certa libertà di azione, dando la possibilità di prestarsi anche a soluzioni ibride di grafi canonici. Possiamo avere un esempio di qaunto descritto nell'opzione torus. Il tipo di grafo torus generalmente è associato ad un grafo in cui ogni nodo ha quattro nodi adiacenti come mostrato in figura:

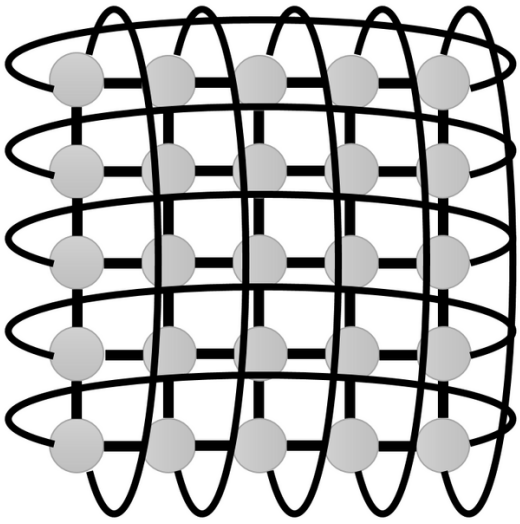

Figura 3.1. esempio di torus graph

Tuttavia, le opzioni messe a disposizione dall'applicazione permettono una certa libertà di movimento, per esempio avendo una configurazione come quella riportata in figura 3.4, una volta analizzati i dati tramite l'analizzatore e dati come imput al visualizzatore, potremmo ottenere un risultato in cui i nodi nella matrice che abbiano un valore pari a zero, non vengano collegati:

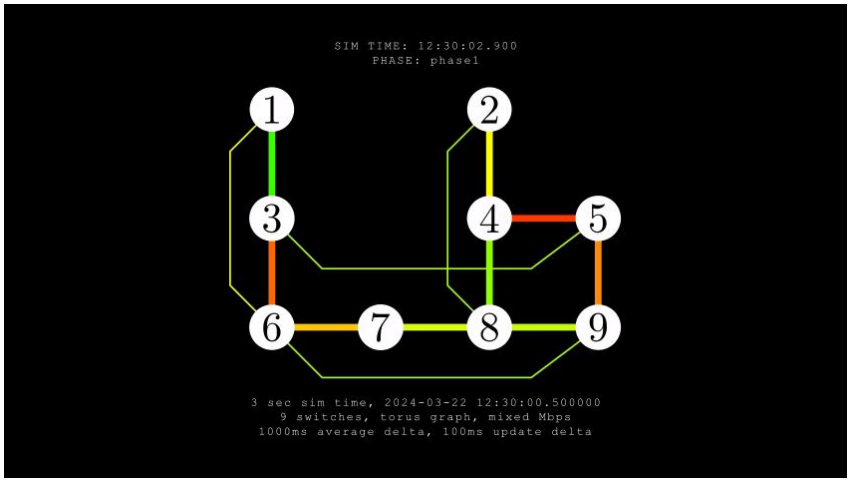

Figura 3.2. risultato grafico del visualizzatore per un custom graph file

Perfetto ora continuiamo

Capitolo 4

Test

Per ottenere i lati di un rettangolo che abbia proporzioni 16 : 9 partendo da un quadrato di lato n, dobbiamo innanzitutto considerare che l'area del quadrato è data da $A = n^2$. Vogliamo che il rettangolo abbia la stessa area del quadrato ma rispetti le proporzioni 16 : 9.

Denotiamo con l la lunghezza e con h l'altezza del rettangolo. La condizione di proporzione si può esprimere come

$$\frac{l}{h} = \frac{16}{9}.$$

Dato che l'area del rettangolo deve essere uguale a quella del quadrato, abbiamo che

$$l \cdot h = n^2.$$

Utilizzando la proporzione, possiamo esprimere l in termini di h come

$$l = \frac{16}{9}h.$$

Sostituendo questa espressione nell'equazione dell'area, otteniamo

$$\frac{16}{9}h \cdot h = n^2,$$

che si semplifica in

$$\frac{16}{9}h^2 = n^2.$$

Da qui, isoliamo h ottenendo

$$h^2 = \frac{9}{16}n^2 \quad \Longrightarrow \quad h = n \cdot \frac{3}{4}.$$

Risostituendo il valore di h nell'espressione di l, abbiamo

$$l = \frac{16}{9} \cdot n \cdot \frac{3}{4} = n \cdot \frac{4}{3}.$$

Quindi, per un quadrato di lato n, per ottenere i lati di un rettangolo che mantenga la stessa area (n^2) con proporzioni 16 : 9, l'altezza h del rettangolo sarà $n \cdot \frac{3}{4}$ e la lunghezza l sarà $n \cdot \frac{4}{3}$.

Quindi il tutto funziona poichè è sempre vero quanto segue:

$$(l + 1)(m + 1) > ml \tag{4.1}$$

4.1 Sotto capitolo test

Ecco un esempio di codice YAML:

```
- coordinates:
    - [1, 2]
    - [3, 0]
```

E ora un esempio di codice Python:

```
links = {}
# Extracting links data
for content in networkData[CONST.NETWORK["LINKS"]]:
    links[frozenset({content["endpoints"][CONST.EP_A], content["endpoi
        #"linkID": link,
        "capacity": content["capacity"],
        "trafficDT":0,
        "trafficUDT":0,
        "updateDeltaTraffic": [],
        "traffic": []
        }
```

La complessità temporale dell'algoritmo è $O(m + n)$.

$$O(m + n) \tag{4.2}$$

.. .. ..

As you can see in the figure 3.2, the function grows near 0. Also, in the page 16 is the same example.

Bibliografia

[1] Stephen Cass Author. *The Top Programming Languages 2023*. https://spectrum.ieee.org/the-top-programming-languages-2023. Accessed: 2024-04-12. 2023.

[2] Cytoscape Consortium. *Cytoscape: An Open Source Platform for Complex Network Analysis and Visualization*. https://cytoscape.org. Accessed: 2024-04-12. 2024.

[3] Gephi. *Gephi: The Open Graph Viz Platform*. https://gephi.org. Accessed: 2024-04-12. 2024.

[4] Graphviz. *Graphviz - Graph Visualization Software*. https://graphviz.org. Accessed: 2024-04-12. 2024.

[5] igraph. *igraph - The Network Analysis Package*. https://igraph.org. Accessed: 2024-04-12. 2024.

[6] Institute of Electrical and Electronics Engineers. *IEEE - Advancing Technology for Humanity*. https://www.ieee.org. Accessed: 2024-04-12. 2024.

[7] Johns Hopkins University. *Homepage of Johns Hopkins University*. https://www.jhu.edu/. Accessed: 2024-04-12. 2024.

[8] Johns Hopkins University Libraries. *Network Data Visualization Guide*. https://guides.library.jhu.edu/datavisualization/network. Accessed: 2024-04-12. 2024.

[9] Pajek. *Pajek - Program for Large Network Analysis*. http://mrvar.fdv.uni-lj.si/pajek/. Accessed: 2024-04-12. 2024.

[10] Jon Postel. *Internet Protocol*. RFC 791. Accessed: 2024-04-12. RFC Editor, 1981. URL: https://www.rfc-editor.org/rfc/rfc791#page-11.

[11] Python Software Foundation. *Python Programming Language – Official Website*. https://www.python.org. Accessed: 2024-04-12. 2024.

[12] Y. Shafranovich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. RFC 4180. Accessed: 2024-04-12. RFC Editor, 2005. URL: https://www.rfc-editor.org/rfc/rfc4180.html#page-2.

[13] SocNetV. *Social Network Visualizer (SocNetV)*. https://socnetv.org. Accessed: 2024-04-12. 2024.

[14] The Manim Community Developers. *Manim Documentation: Quickstart Guide*. Accessed: 2024-03-25. 2024. URL: https://docs.manim.community/en/stable/index.html.

[15] UCINET. *UCINET Software*. `https://sites.google.com/site/ucinetsoftware/`. Accessed: 2024-04-12. 2024.

[16] YAML Core Team. *YAML Ain't Markup Language (YAML™) Version 1.2*. `https://www.rfc-editor.org/info/rfc9512`. Accessed: 2024-04-12. 2009.