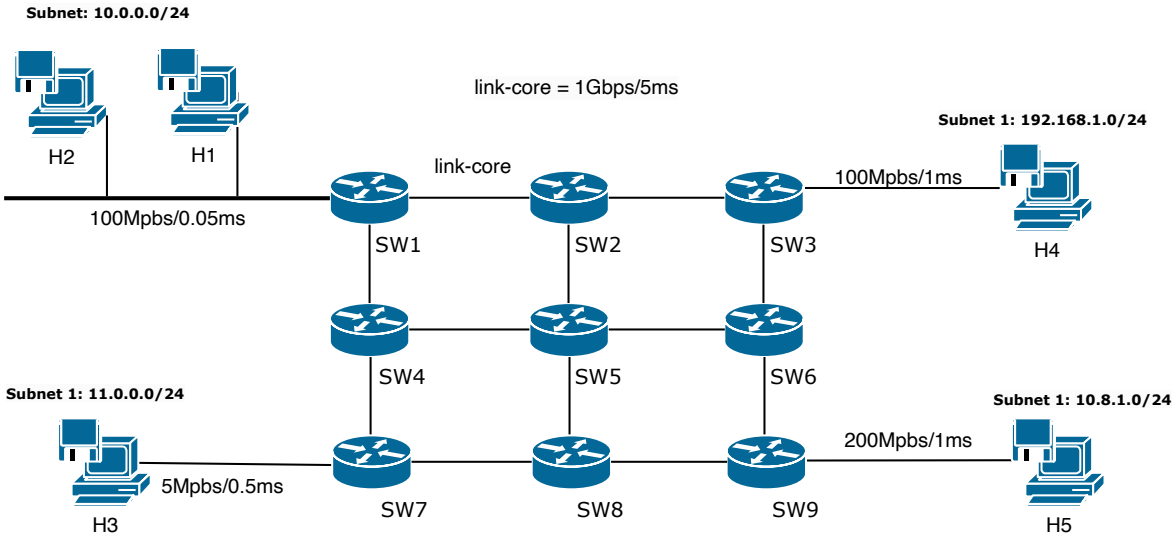
Reti di Calcolatori A.A. 2025-2026

Progetto 10

Progettare ed implementare un sistema di comunicazione che emuli una rete composta da switch, router e host. In particolare, si faccia uso di **Mininet** [1] per la realizzazione della rete. La topologia di rete richiesta è composta da n. 9 nodi che operano come switch OpenFlow e n. 5 host. I nodi di rete sono collegati tra loro come mostrato in figura:



In figura sono inoltre indicate le capacità dei link e i ritardi di propagazione. I link della griglia di switch sono uguali e configurati con C=1Gpbs e d=5ms.

Si consiglia di utilizzare un controller SDN per configurare gli indirizzi di rete le tabelle di flusso per ciascuno switch. (esempio: Ryu-Manger)[2] . È possibile configurare i nodi host usando REST [3] .

Occorre sviluppare il progetto rispettando le seguenti specifiche:

1. Tutti gli host devono essere raggiungibili fra loro,
2. Gli indirizzamenti IP devono rispettare le sottoreti indicate nello schema.
3. La rete deve prevedere uno schema di adattamento dinamico dei percorsi di instradamento degli switch preesnti nella rete core. In particolare, il peso dei link è da considerare dinamico in funzione della quantità di byte che transitano nella rete. Come noto dalla teoria (e.g. path-oscillation) la scelta del percorso ottimo mediante l'algoritmo dijkstra dinamico può innescare instabilità. Si chiede di esplorare il problema implementando un controller che assegni i percorsi ai nodi su diversi approcci, ad esempio esplorando l'algoritmo A* come estensione di Dijkstra per mitigare tali criticità.
4. I link emulati con Mininet devono avere Rate e Ritardo di propagazione come indicato nello schema.

5. Sul nodo H1, avviare un server http usando il framework Flask[4] che implementi un servizio di API REST per l'esecuzione di esperimenti di prestazioni sulla rete.
6. Ogni nodo della rete deve poter raggiungere il servizio su H1 ed inviare i comandi.
7. I comandi forniti nella API consistono nell'attivazione di traffico mediante tool "iperf". In particolare, è necessario che siano definite almeno i seguenti endopoint:
    a. /start_iperf, che avvia il traffico iperf e prende come input: *IP_DEST*, *L4_proto*, *src_rate*
    b. /stop_iperf, che interrompe ogni traffico iperf istanziato sul client.
8. Il destinatario deve essere validato mediante il campo *src_rate*.
9. Il traffico tra client e server che può essere di tipo sia TCP che UDP validando il campo *L4_proto*.
10. Il data-rate di invio dei pacchetti può essere modificato impostando il campo *src_rate*
11. È necessario che in fase di inizializzazione della rete, tutti i nodi devono avviare un server iperf, per tale ragione si suggerisce di avviare i server in fase di avvio della topologia.
12. I risultati degli esperimenti svolti devono essere salvati per ciascun host su file di log separati. Per il salvataggio su file di log sui server si suggerisce di utilizzare il seguente formato di avvio: **`iperf -s -y C >> h2_log.csv &`**
    Usare il formato già fornito da iperf così strutturato:
    20241128113057,10.1.1.1,5001,192.168.1.2,56178,4,0.0-12.0,1441792,957762
13. Considerare due scenari implementativi: Dijkstra e A*. Per entrambi gli scenari di routing dinamico occorre effettuare dei log dei percorsi ottimi in assenza di traffico i in presenza di traffico. Nello specifico si chiede di analizzare i seguenti scenari:
    a. H2 attiva un flusso UDP al massimo del data-rate supportato verso H1
    b. H2 e H3 attivano flusso UDP al massimo del data-rate supportato verso H1
    c. H2, H3 e H4 attivano flusso UDP al massimo del data-rate supportato verso H1
    d. H2, H3, H4 e H5 attivano flusso UDP al massimo del data-rate suppotato verso H1

    Per ciascuno esperimento devono essere loggati i valori di throughput come descritto nel pungo 12. Confrontare gli scenari a-d sia con Dijkstra sia con A*. Analizzare eventuali differenze in termini di prestazioni. Concettualmente ci si aspetta che al variare del flusso l'algoritmo di routing debba inoltrare flussi evitando link troppo con elevato traffico. A titolo di esempio H3 raggiunge H1 tramite SW7, SW4 e SW1 mentre H5 raggiunge H1 tramite SW9, SW6, SW5, SW2, SW1.
14. Rappresentare graficamente i percorsi ottimi dinamici con un intervallo di aggiornamento (es T=5sec). Rapprsentare graficamente i throughput ottenuti con i due algoritmi di instradamento.

Output previsto:

- **Relazione tecnica di progetto** dettagliata che includa la descrizione dell'analisi progettuale, la descrizione e lo schema logico architetturale del sistema, l'analisi e la validazione delle specifiche richieste, le metodologie di raccolta dati e risultati ottenuti. Non includere il codice nella relazione, esso deve essere consegnato su file separati.
- **Codice sorgente del progetto** (script, eventuali file di configurazione, eventuali file di libreria, eventuali log, eventuali dipendenze).

Modalità di consegna dei progetti:

- Invio e-mail all'indirizzo fabrizio.giuliano@unipa.it, si suggerisce di specificare nell'oggetto "PROGETTO RETI DI CALCOLATORI ID-PROGETTO/COGNOME", allegando:
  - PDF della relazione
  - ZIP file contenente i codici sorgente.
- Nel caso di tesine di gruppo, tutti i componenti del gruppo devono essere inclusi tra i destinatari della email.

Link di riferimento:

[1] https://mininet.org/
[2] https://ryu.readthedocs.io/en/latest/
[3] https://osrg.github.io/ryu-book/en/html/rest_router.html
[4] https://flask.palletsprojects.com/en/stable/
[5] https://github.com/Rad6/SDN-Routing-Ryu.git