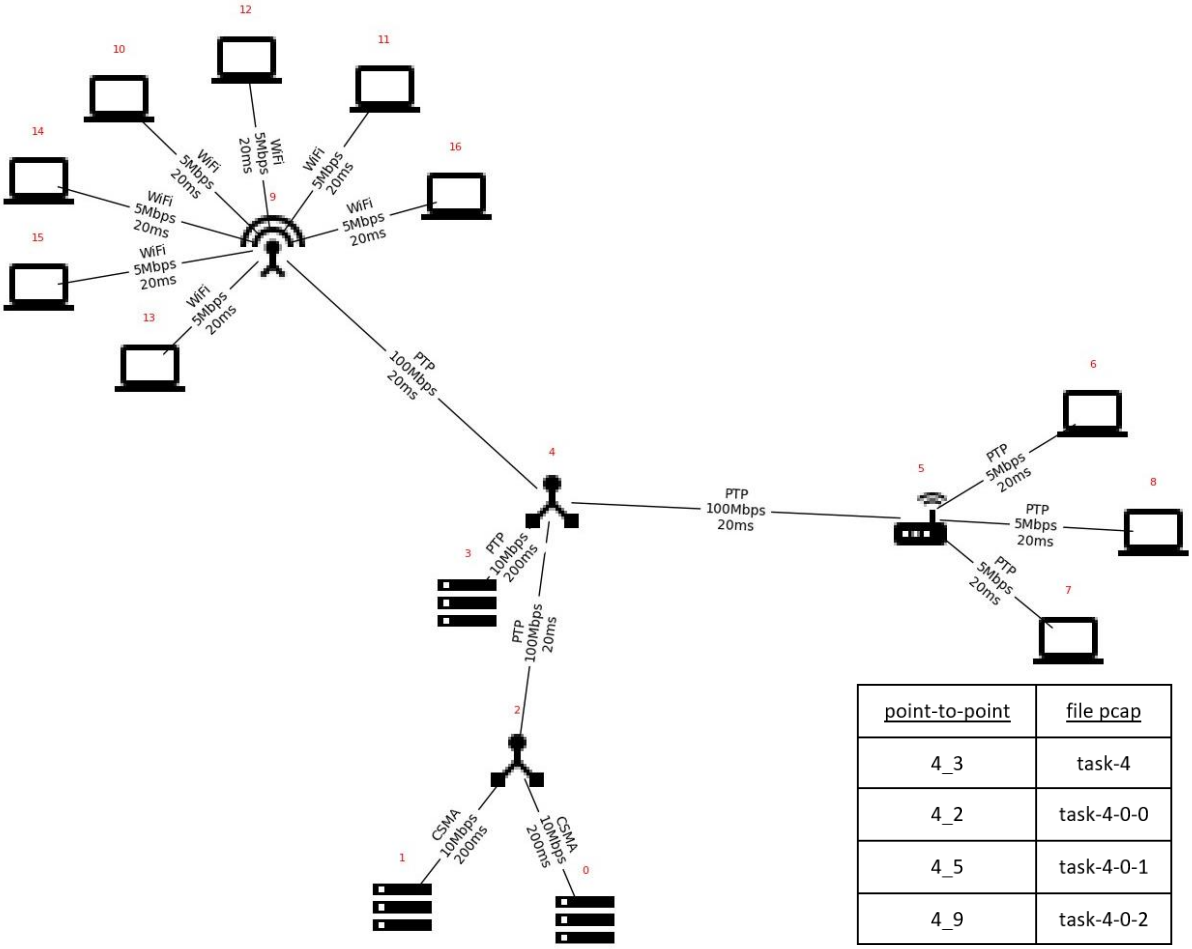
Nome referente: Benedetta Fiorillo	Matricola: 1710531	Corso: Telecomunicazioni



point-to-point	file pcap
4_3	task-4
4_2	task-4-0-0
4_5	task-4-0-1
4_9	task-4-0-2

```
TCP burst traffic of 1117 B for each packet starting at 0.25 s
        Sender: Node 14 Receiver: Server 0
TCP burst traffic of 1369 B for each packet starting at 3.80 s
        Sender: Node 8  Receiver: Server 1
TCP burst traffic of 1745 B for each packet starting at 3.97 s
        Sender: Node 10 Receiver: Server 0
Wi-Fi operating in Infrastructure mode, AP is stationary, Stations nodes move with
random walk mobility pattern over a bounded square of 30 meters for each side
UDP Echo Application with Client 6 and Server 3
        Size of packet: 1724 Bytes      Periodicity: 20ms       MaxPackets: 250
```

Note
È stata aggiunta una tabella per la corrispondenza tra interfacce e file pcap per il nodo 4, per evitare eventuali ambiguità.

Creazione della rete
Dopo aver analizzato lo schema fornito, sono stati creati i nodi e, sfruttando le topologie di interesse, i link per collegarli tra di loro.
Per quanto riguarda la rete wireless lo standard utilizzato (e richiesto) è ***802.11g***.
Tramite la classe ***InternetStackHelper*** è stata installata la stack internet in tutti i nodi; inoltre, tramite la classe ***Ipv4AddressHelper***, sono stati assegnati i seguenti indirizzi IP:

Star	191.168.0.0/29	4+2 indirizzi → 3 bit richiesti
WiFi	10.1.1.0/28	8+2 indirizzi → 4 bit richiesti
Sottorete_router5	Da 10.1.2.0/30 a 10.1.2.11/30	3 P2P
Sottorete_router2	Da 10.1.3.0/30 a 10.1.3.7/30	2 Ethernet CSMA
Sottorete_router3	195.130.67.0/30	1 P2P

Nella scelta degli indirizzi l'obiettivo è stato quello di minimizzare le relative maschere quanto più possibile. Infine, per lo strato applicativo, sono state sviluppate tre ***On-Off*** Application con l'utilizzo del protocollo TCP e una ***UDP Echo*** Application seguendo i requisiti forniti sopracitati, ossia: dimensione e numero massimo dei pacchetti, sender e receiver, tempo d'inizio ed eventuale periodicità. Per forzare l'uso del meccanismo RTS/CTS è stata scelta come soglia il valore 500, poiché l'MSS di default è pari a 536 Bytes.
Complessivamente la simulazione ha durata pari a 15 secondi.

Topologie e ritardi di rete

1) Le varie topologie che compongono la rete sono le seguenti:

- ***Stella*** con Router4 come hub e 4 spokes (RouterWiFi9, Router5, Router2, Server3)
- Due ***alberi***: il primo con Router5 come radice e 3 host (Laptop 6, Laptop7, Laptop8) e il secondo con Router2 come radice e due host (Server0, Server1). Gli host sono collegati alle rispettive radici tramite link point-to-point e link Ethernet CSMA
- ***Point-to-point*** tra il Router4 e il Server3
- ***Rete wireless*** con Infrastructure Mode, AP stazionario e host a mobilità random.

2) La ricostruzione del percorso è stata eseguita usando i file pcap generati senza il meccanismo RTS/CTS; inoltre, non è stato ricostruito il flusso dell'Echo Application come richiesto dalle specifiche.

- ***On/Off*** Application tra sender 14 e receiver 0
  file pcap: task-9, task-4-0-2
  filtri: `tcp && ip.src == 10.1.1.5` per tracciare i pacchetti dal nodo 14 al nodo 0;
  `tcp && ip.src == 10.1.3.6 && ip.dst == 10.1.1.5` per tracciare i pacchetti ACK dal nodo 0 al nodo 14
- ***On/Off*** Application tra sender 8 e receiver 1
  file pcap: task-5, task-4-0-0
  filtri: `tcp && ip.src == 10.1.2.10` per tracciare i pacchetti dal nodo 8 al nodo 1;
  `tcp && ip.src == 10.1.3.2` per tracciare i pacchetti ACK dal nodo 1 al nodo 8
- ***On/Off*** Application tra sender 10 e receiver 0
  file pcap: task-9, task-4-0-2
  filtri: `tcp && ip.src == 10.1.1.1` per tracciare i pacchetti dal nodo 10 al nodo 0;
  `tcp && ip.src == 10.1.3.6 && ip.dst == 10.1.1.1` per tracciare gli ACK dal nodo 0 al nodo 10.

3) I ***bottlenecks*** che si hanno nella rete sono in corrispondenza dei link con il bit rate più basso, ovvero i link con capacità 5 Mbps e 10 Mbps, a seconda delle comunicazioni che si vogliono eseguire. Supponendo che un laptop collegato alla rete WiFi voglia comunicare con un laptop nella sottorete del router 5, il bottleneck è 5 Mbps. Lo stesso si avrebbe se un qualunque laptop volesse comunicare con i server che si trovano nelle due reti CSMA. Se questi ultimi stabiliscono una comunicazione che non sia verso i laptop presenti, allora il bottleneck sarà di 10Mbps.
Possibili contromisure e soluzioni al fine di evitare bottlenecks sono:

- Aumentare il bit rate del link
- Diminuire la distanza fisica tra nodi così da ridurre il tempo di propagazione.

4) Il ***throughput*** medio del flusso di stato di trasporto per t=2.0s è pari a:
throughput 4.0.0 → $46248\ byte\ /\ 2.0\ s\ =\ 23124\ byte/s \sim 23{,}12\ KB/s$
throughput 4.0.1 → $0\ byte\ /\ 2.0\ s\ =\ 0\ byte/s \sim 0\ KB/s$
throughput 4.0.2 → $46248\ byte\ /\ 2.0\ s\ =\ 23124\ byte/s \sim 23{,}12\ KB/s$

È stato calcolato applicando il filtro `tcp && !udp && frame.time_epoch <= 2.0` e considerando le interfacce relative ai point-to-point della stella con nodo 4 come hub centrale, quindi i relativi file pcap. Poiché nell'interfaccia relativa ai nodi 3_4 vi è solo il passaggio di segmenti UDP, non è stata considerata nell'analisi. I dati sono stati estratti aggiungendo una colonna "cumulative bytes", in modo da ottenere la somma dei byte ottenuti nei primi 2.0s.

5) Il ***throughput*** medio del flusso di stato di trasporto per t=5.0s è pari a:
throughput 4.0.0 → $153028\ byte\ /\ 5.0\ s\ =\ 30605{,}6\ byte/s \sim 30{,}60\ KB/s$
throughput 4.0.1 → $1701\ byte\ /\ 5.0\ s\ =\ 340{,}2\ bytes/s \sim 0{,}34\ KB/s$
throughput 4.0.2 → $151327\ byte\ /\ 5.0\ s\ =\ 30265{,}4\ bytes/s \sim 30{,}27\ KB/s$

È stato calcolato con la stessa metodologia riportata nel punto 4, modificando il filtro in `tcp && !udp && frame.time_epoch <= 5.0`. La differenza tra i risultati ottenuti per tempo 2.0s e 5.0s differiscono a causa dei tempi di accensione delle On/Off Application: come si evince dal grafico TCP Steam Graph relativo al throughput,

mentre nei primi due secondi solo una di queste genera un burst, nei restanti tre secondi anche le altre due generano traffico ed inoltre si osserva un altro burst da parte della Application già attiva.

Rete Wireless (modalità Infrastructure nello strato WiFi)

1) L'***Access Point*** è il nodo di accesso alla rete WiFi, invia periodicamente un frame "faro" chiamato ***beacon*** che ha il compito di segnalare la presenza dell'AP a tutti i nodi stazione, inoltre salva il traffico in un buffer, in attesa che essi si sveglino.
In questo modo, i nodi stazione possono disattivarsi quando non c'è nulla da trasmettere, effettuare polling verso di lui una volta svegliati e in seguito tornare in modalità sleep.
Applicando il filtro `wlan.fc.type_subtype == 0x0008` nel file task-9.pcap, relativo al modem WiFi, si può osservare che il primo invio del beacon file avviene pochi istanti dopo l'inizio della simulazione.
Analizzandolo nel dettaglio si può notare che viene inviato in modalità broadcast (destination address) e comunica le informazioni dell'AP (source address) e della rete (capabilities information); ad esempio è riportato che l'AP è il trasmittente, cioè che il WiFi opera in modalità Infrastructure.

2) Le parti della rete dove possono avvenire ***collisioni*** sono la rete WiFi e le due reti CSMA, cioè le tre reti ad accesso condiviso; infatti, gli altri collegamenti sono di tipo point-to-point, dove la trasmissione è dedicata.
Applicando il filtro `tcp.analysis.retransmission` e controllando il flag `Retry` all'interno del campo Frame control (IEEE 802.11) per i segmenti che passano nella rete WiFi, si può notare che nella simulazione non ci sono collisioni. Inoltre, anche da un'analisi per verifica diretta non risultano ritrasmissioni. Scenario ben differente da ciò che ci aspetterebbe da una situazione reale, soprattutto nella rete WiFi, dove non c'è alcun meccanismo di Collision Avoidance. Probabilmente, grazie anche alla dimensione dell'MSS di default usata da ns3, pari a 536 Byte, che suddivide i segmenti TCP in segmenti più piccoli, si diminuisce la probabilità di collisione.

3) Dopo aver utilizzato la funzione SetFill, la dimensione del pacchetto UDP scende da 1724 a 141 Byte; infatti, il pacchetto viene ridimensionato come previsto dalla funzione stessa. Quindi la quantità utile di informazioni scambiate è 282 Byte.
Ad ogni pacchetto bisogna aggiungere solo la dimensione dell'header UDP di 30 Byte; infatti, nei canali attraversati dal flusso dei pacchetti non vi sono costi legati al meccanismo RTS/CTS o costi di associazione relativi alla rete WiFi. I pacchetti viaggiano solo in reti point-to-point; quindi, i file che rappresentano i tre link attraversati (pcap task-5, task-4-0-1, task-4) non contengono al loro interno ARP request o ARP reply. Non è presente quindi neanche il costo di attraversamento relativo ad eventuali reti MAC.
L'***overhead*** complessivo della Echo Application è quindi pari a 342 Byte.

4) Forzando l'uso del ***meccanismo RTS/CTS*** per i pacchetti di strato applicativo si aggiunge al flusso dei dati nella rete due tipi di frame di controllo `request to send` e `clear to send` che permettono di diminuire la probabilità di collisioni. Come specificato nel punto 2, non ci sono state collisioni, si ha perciò soltanto un aumento di traffico.

5) Come specificato nel punto 3, i pacchetti della Echo Application attraversano solo collegamenti point-to-point scollegati dalla rete WiFi; quindi, forzando il meccanismo RTS/CTS non cambia nulla relativamente all'***overhead*** complessivo.