



UNIVERSITÀ DI PISA  
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea in Ingegneria Informatica

**Valutazione del Consumo Energetico  
di HTTP/3 in Ambiente 5G**

PRIMO RELATORE:  
**Prof. ALESSIO VECCHIO**

CANDIDATO:  
**GIOVANNI LIGATO**

SECONDO RELATORE:  
**Ing. VALERIO LUCONI**

ANNO ACCADEMICO 2022/2023

---

*“So di non sapere”*

- SOCRATE -

# Indice

<b>1. Introduzione</b>	<b>1</b>
1.1. Fasi . . . . .	3
<b>2. Ambiente di Lavoro</b>	<b>4</b>
2.1. Hardware . . . . .	4
2.1.1. Setup . . . . .	4
2.2. Software . . . . .	6
2.2.1. Software per la Fase di Ristrutturazione . . . . .	6
2.2.2. Software per la Fase di Raccolta Dati . . . . .	6
2.2.3. Software per la Fase di Analisi . . . . .	12
<b>3. Scenario</b>	<b>13</b>
3.1. Qualche Accorgimento . . . . .	15
3.2. Dettagli della RAN . . . . .	15
3.2.1. Metodologia . . . . .	16
3.2.2. Quarta Generazione - LTE . . . . .	16
3.2.3. Quinta Generazione - NR . . . . .	16
3.3. Diverse Condizioni di Rete . . . . .	18
<b>4. Esperimenti</b>	<b>20</b>
4.1. Il File config.ini . . . . .	21
<b>5. Problemi e Soluzioni</b>	<b>23</b>
5.1. Inaffidabilità del Canale UART . . . . .	23
5.1.1. Soluzione (RDT 4.0) . . . . .	23
5.2. Una strana Eccezione . . . . .	27
5.2.1. Soluzione . . . . .	29
5.3. Un BUG dell'Ambiente Otii . . . . .	29
5.3.1. Soluzione . . . . .	30
5.3.2. Una Nuova Gestione delle Eccezioni . . . . .	30
<b>6. Risultati</b>	<b>32</b>
6.1. Analisi dei Risultati Preliminari . . . . .	32
6.2. Analisi dei Risultati Principali . . . . .	37
6.3. Analisi dei Risultati Aggiuntivi . . . . .	49
<b>7. Conclusioni</b>	<b>55</b>
<b>8. Bibliografia</b>	<b>56</b>

## 1. Introduzione

Una *Valutazione del Consumo Energetico di HTTP/3 in Ambiente 5G* ha lo scopo di misurare il consumo energetico dell'*ultima* versione dell'*HyperText Transfer Protocol*, utilizzando la *più recente* tecnologia di accesso mobile. Al fine di fornire un quadro più completo, in cui la valutazione in questione potrà assumere maggiore significato, vengono considerate nello studio anche le precedenti versioni degli elementi menzionati. Si considerano pertanto la versione 2 di HTTP e la *quarta* generazione delle tecnologie di accesso radio (i.e. 4G).

Relativamente alle versioni del protocollo HTTP, si è deciso di non includere nello studio anche la più longeva versione 1.1 perché nelle misurazioni fatte in [1] si osserva che le versioni 1.1 e 2 di HTTP hanno un comportamento analogo. Infatti, in riferimento al consumo energetico, i risultati di un'analisi ANCOVA (i.e. *analysis of covariance*), dove viene considerato il consumo energetico come variabile dipendente e la dimensione del payload di risposta come covariata, mostrano che i modelli lineari prodotti dall'analisi per le versioni 1.1 e 2 sono quasi sovrapposti. In effetti, la valutazione della significatività statistica delle differenze dei parametri dei due modelli ha avuto esito negativo, confermando l'affinità delle due versioni. La versione 3 del protocollo HTTP introduce modifiche interessanti, perché a differenza delle versioni precedenti che fanno uso del TCP, come protocollo di livello trasporto, questa utilizza il recente protocollo QUIC. QUIC, pur utilizzando UDP come base, ha lo scopo di raggiungere l'affidabilità del TCP con una latenza notevolmente ridotta. Gli algoritmi di controllo della congestione vengono spostati nello *spazio utente* (user space) in entrambi gli estremi della connessione, diversamente dal TCP in cui sono definiti nello *spazio del kernel* (kernel space). Poiché la maggior parte delle connessioni HTTP attuali fanno uso di TLS per lo scambio sicuro delle informazioni, QUIC include direttamente lo scambio delle chiavi e delle suite di cifratura supportate all'interno del processo di *handshaking* iniziale [2]. Quindi HTTP/3 è per definizione sempre crittografato e sicuro [3].

Per quanto riguarda le generazioni precedenti alla *quarta* delle tecnologie di accesso radio, queste non vengono considerate perché ormai superate, essendo la distribuzione del 4G in uno stato *abbastanza* avanzato. Per essere più precisi, in [4] viene mostrato che la percentuale della popolazione mondiale coperta dal 4G è passata dal 43% nel 2015 all'88% nel 2022. Il 5G, d'altra parte, ha raggiunto solo il 35% alla fine del 2022 [5]. Da queste semplici stime e, considerando che sono trascorsi *soltanto* 4 anni dalla prima distribuzione su larga scala del 5G (Corea del Sud, Aprile 2019 [6]), è chiaro che il 5G si trova ancora in uno stato prematuro. Se si considerano anche i diversi schemi di distribuzione del 5G si scopre che la strada da compiere per arrivare ai livelli di distribuzione del 4G è ancora più lunga di quanto le precedenti stime possano far trasparire. Difatti, il 5G può essere distribuito come:

- *Non-Standalone* (NSA);
- *Standalone* (SA).

Per ridurre il tempo di distribuzione sul mercato, nonché i costi associati, molti operatori iniziano con il distribuire il 5G NSA. Esiste però una enorme differenza tra le due configurazioni. In particolare, nel 5G NSA, la base station 5G (denominata gNB) è co-localizzata con una base station 4G (eNB). Le due stazioni condividono la stessa infrastruttura di rete 4G *Evolved Packet Core* (EPC). Sotto NSA, quindi, il 5G opera solamente al livello del piano dati (data plane) e si affida al 4G per le operazioni del piano di controllo (control plane). La configurazione SA, invece, è completamente indipendente dall'infrastruttura del 4G, e solamente con questo schema di distribuzione è possibile sfruttare appieno il potenziale del 5G. Infatti, sfruttando la *Core Network* propria del 5G, si riesce a garantire un netto miglioramento al *signaling*, ai messaggi di sincronizzazione e alla latenza per l'utente finale. In questo modo lo UE (*User Equipment*) 5G è in grado di sincronizzarsi più velocemente, risparmiando energia per merito della riduzione del tempo totale di *ON*. All'interno della percentuale relativa alla copertura del 5G (i.e. 35%) ricadono perciò anche le aree in cui è stato distribuito *soltanto* il 5G NSA. Da qui segue la precedente osservazione sullo stato alquanto primitivo della attuale distribuzione del 5G. Dunque, per arrivare ad una situazione in cui si riesca a sfruttare il vero potenziale del 5G bisognerà attendere ancora alcuni anni.

Altra caratteristica del 5G è quella di avere la possibilità di operare in un ampio spettro di frequenze. Secondo gli standard 3GPP, lo spettro di frequenze del 5G New Radio (NR) è composto da due bande:

- banda *sub-6 GHz*, anche chiamata *C-band*, comprende le frequenze che vanno da 0.45 GHz a 6 GHz;
- banda *millimeter-wave*, o *mmWave*, comprende invece frequenze più elevate contenute nell'intervallo [24.25, 52.6] GHz. Il mmWave offre una larghezza di banda ultra elevata, ma a causa della lunghezza d'onda molto ridotta è particolarmente sensibile a fattori come la mobilità e a ostacoli fisici, soffrendo dunque di forti attenuazioni e di *penetration loss*.

Le diverse bande di frequenza utilizzate dal 5G hanno pertanto differenti performance e anche diverse caratteristiche di propagazione del segnale. La scelta della banda da adoperare da parte degli operatori telefonici dipende dalle licenze dello spettro di cui dispongono e da considerazioni tecniche / di business. Per ulteriori approfondimenti si vedano [7, 8, 9].

Lo *scenario* in cui si andrà a valutare il consumo energetico riprende il pattern di comunicazione del tipo *machine-to-machine*. In una comunicazione di questo tipo un client IoT (*Internet of Things*) interagisce con un server remoto seguendo un approccio di *polling*. Il client dunque invia richieste HTTP periodiche per caricare / scaricare informazioni sul / dal server. Nella maggior parte dei casi il client IoT si trova però ad operare in ambienti dove l'accesso alle risorse non è illimitato. Una situazione abbastanza diffusa è quella di dispositivo IoT alimentato da batteria, che per definizione ha una capacità limitata, e vincolato al dover utilizzare una connessione mobile per comunicare con il server. Limitare lo studio ad uno scenario del genere non è per nulla riduttivo, in quanto

al 2022 il numero di dispositivi IoT connessi in tutto il mondo è di 13.14 miliardi. Secondo previsioni il numero dovrebbe più che raddoppiarsi entro la fine del decennio, raggiungendo i 29.42 miliardi nel 2030 [10]. Pertanto, se durante il processo di valutazione si riuscisse ad indentificare una configurazione particolarmente vantaggiosa per una data condizione di rete, allora si potrebbe ottenere un guadagno complessivo di energia non indifferente applicando la configurazione trovata tutte le volte in cui si presenti la specifica condizione di rete. Questo risparmio di energia, oltre ad avere un impatto ecologico, consentirebbe di aumentare la vita delle batterie dei dispositivi IoT. Il procedimento che verrà seguito nello studio è, di conseguenza, quello di approfittare delle due dimensioni di cui si ha il controllo (i.e. versione del protocollo HTTP e generazione della tecnologia di accesso radio) per valutare il consumo di un dispositivo IoT a fronte delle diverse condizioni di rete che possono presentarsi durante il suo normale funzionamento (e.g. parametri che influenzano lo stato della rete sono ad esempio latenza e banda).

### 1.1. Fasi

Le Fasi attraversate possono essere riassunte come segue:

1. Ristrutturazione del sistema di [1] per automatizzare il processo di raccolta dati: il controllo delle condizioni di rete viene incluso all'interno del codice che si occupa di eseguire i diversi Esperimenti.
2. Raccolta dei Dati: dopo aver scelto le configurazioni da testare il sistema eseguirà gli Esperimenti registrando il consumo energetico di ciascuno.
3. Analisi dei Risultati: i dati ottenuti durante la fase di raccolta vengono aggregati in grafici per studiare il rapporto che sussiste tra una particolare configurazione e il relativo consumo energetico.

## 2. Ambiente di Lavoro

Di seguito si riportano i dettagli dell'ambiente in cui è stato possibile svolgere lo studio.

### 2.1. Hardware

Dispositivi Hardware:

- Single Board Computer: Raspberry Pi 3B+ [11];
- Scheda di rete 5G: SIM8200EA-M2 5G HAT for Raspberry Pi [12];
- Alimentatore e Analizzatore per valutare il consumo energetico: Otii Arc Pro [13];
- Notebook di ausilio per controllare gli Esperimenti e per eseguire il Software di gestione dell’Otii Arc Pro.

#### 2.1.1. Setup

I dettagli relativi alla connessione fisica tra Otii Arc Pro e Raspberry Pi sono riportati nella seguente Figura.

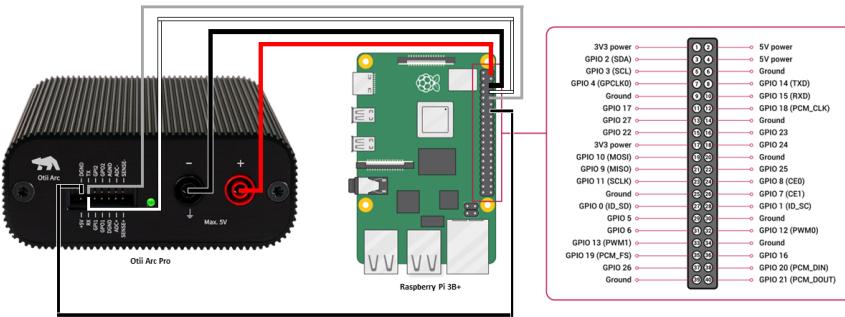


Figura 1: Schema di Connessione tra Otii Arc Pro e Raspberry Pi 3B+.

Oltre ad alimentare il Raspberry Pi (pin 2 e 6), i collegamenti mostrati in Figura permettono di instaurare un canale di comunicazione UART (*Universal Asynchronous Receiver-Transmitter*) per permettere lo scambio mi messaggi tra l’Otii Arc Pro e il Raspberry Pi.

Il Notebook in cui è in esecuzione il Software Otii è connesso all’Otii Arc Pro attraverso un collegamento USB.

Per alimentare il Raspberry Pi tramite l’Otii Arc Pro bisogna configurare il Software Otii. In particolare, dopo aver aperto il software ed aver creato un nuovo progetto Otii, dal menù laterale sinistro, sotto la voce *Control*, cliccare

su *Add* per aggiungere il dispositivo Arc. A questo punto, sempre dal menù laterale sinistro, impostare la *Main Current* ad *High range* e la tensione a 5V, perché questa è la tensione richiesta dal Raspberry Pi. In *OC protection* impostare un valore di corrente appropriato (e.g. maggiore di 2.5A e inferiore a 5A), cosicché se dovesse essere richiesta una corrente superiore al valore impostato, il software dell’Otii Arc provvederà ad interrompere l’erogazione di energia, limitando eventuali danni (impostazione di *Cut-off*). Cliccando sul simbolo di accensione, riportato in alto accanto alla tensione, il Raspberry Pi verrà alimentato utilizzando i parametri appena impostati.

L’installazione della scheda di rete 5G sul Raspberry Pi segue lo schema riportato in 2.



Figura 2: Installazione della SIM8200EA-M2 5G HAT sul Raspberry Pi 3B+ (Figura tratta da [12]).

Alla scheda 5G sono connesse poi le antenne ANT0, ANT1, ANT2, ANT3, e ANT4 (Figura 3). Si noti che, per gli esperimenti richiesti nell’ambito dello studio corrente, l’antenna GPS, ANT5 nella figura, non è necessaria. Per tale motivo non viene proprio collegata alla scheda.

La scheda SIM8200EA-M2 5G HAT ha bisogno, per poter funzionare correttamente, di essere connessa al Raspberry Pi attraverso un cavo USB. In più ha bisogno di essere alimentata tramite l’ingresso Micro-USB (si veda anche la Figura 4 dove vengono riportati i dettagli della scheda) con 5V e 3A. Per registrare l’energia richiesta dalla scheda 5G durante l’esecuzione degli Esperimenti, bisogna che anche questa venga alimentata attraverso l’Otii Arc Pro. Fortunatamente, la tensione richiesta dal Raspberry Pi e dalla scheda 5G è esattamente la stessa. Si è deciso perciò di utilizzare un cavo Micro-USB da connettere in parallelo all’alimentazione in uscita dall’Otii Arc e in ingresso al Raspberry Pi. A questo punto, considerando che un Raspberry PI 3B+ per poter avviarsi senza altri dispositivi connessi ha bisogno di 1.5A [14] e ricordando che la corrente assorbita dalla scheda 5G è di circa 3A, bisogna aumentare il limite della corrente nel software di gestione dell’Otii alla voce *OC protection* ad un minimo di 4.5A. Eseguite queste operazioni si è verificata la corretta accensione del Raspberry Pi e della scheda di rete.

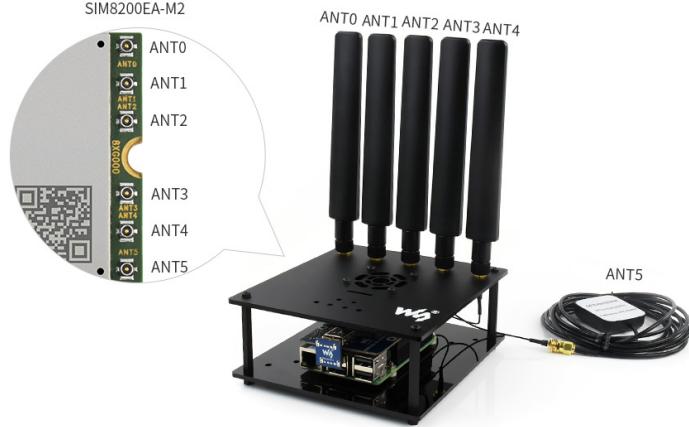


Figura 3: Connessione delle antenne esterne alla scheda SIM8200EA-M2 5G HAT (Figura tratta da [12]).

Si riporta in Figura 5 configurazione in cui la scheda di rete 5G viene alimentata tramite l’Otii Arc Pro. Si noti la presenza del led rosso sulla scheda ad indicare la corretta alimentazione.

Per connettere la scheda 5G alla rete mobile è necessario inserire una SIM (nel caso in esame una SIM *iliad*) nell’apposito slot e seguire la procedura riportata in [15].

## 2.2. Software

Diversi Software sono stati utilizzati durante le tre Fasi dello studio:

### 2.2.1. Software per la Fase di Ristrutturazione

Essendo l’attività principale di questa fase quella di scrittura del codice si è fatto uso principalmente di un IDE (*Integrated Development Environment*). In particolare l’ambiente di sviluppo che si è scelto è Visual Studio Code [16].

### 2.2.2. Software per la Fase di Raccolta Dati

Durante la fase di Raccolta Dati viene avviato il sistema per l’esecuzione automatica degli Esperimenti. Il codice del sistema è scritto in Python, per questo viene utilizzato sia sul Notebook sia sul Raspberry Pi l’interprete python3 [17].

Per l’invio delle richieste HTTP il Raspberry Pi fa uso dell’applicativo via riga di comando `curl` [18]. In particolare, vengono impiegate due differenti versioni per valutare l’impatto delle diverse implementazioni sul consumo energetico. Per le richieste in HTTP/2 e HTTP/3, `curl` si appoggia a librerie di terzi.

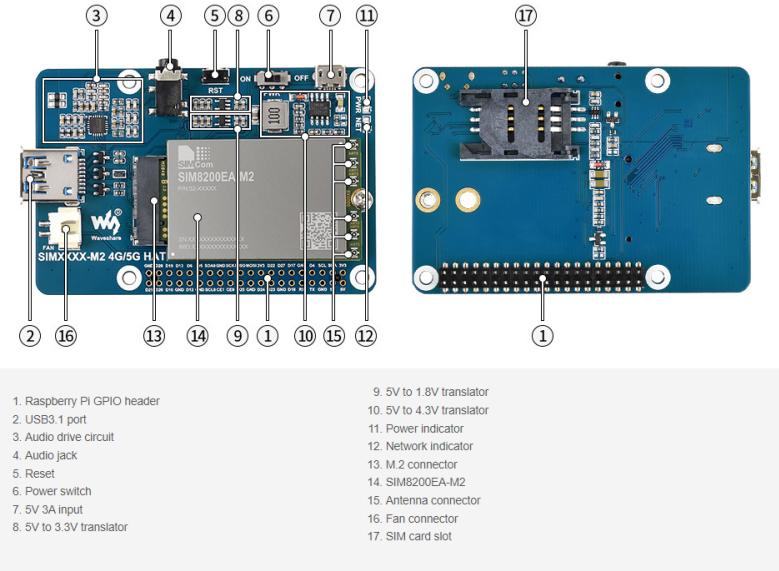


Figura 4: Scheda SIM8200EA-M2 5G HAT vista nel dettaglio (Figura tratta da [12]).

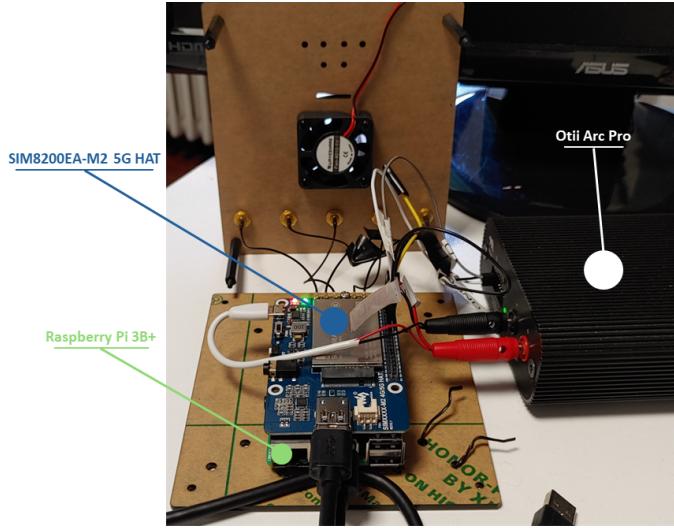


Figura 5: SIM8200EA-M2 5G HAT alimentata tramite l’Otii Arc Pro.

- **curl7.85.0-DEV:**
  - HTTP/2: nghttp2 (versione 1.36.0);
  - HTTP/3: nghttp3 (versione 0.6.0-DEV) + ngtcp2 (versione 0.7.0-DEV).
- **curl8.2.1-DEV** (ultima versione di `curl` al 25/07/2023):
  - HTTP/2: nghttp2 (versione 1.36.0);
  - HTTP/3: nghttp3 (versione 0.13.0) + ngtcp2 (versione 0.17.0).

Per mantenere la versione 7.85.0-DEV, la nuova versione di `curl` (i.e. la 8.2.1-DEV) è stata installata localmente all'interno di:

`/home/pi/ligato/NewCurl`

Vengono installate anche le componenti necessarie per l'utilizzo di HTTP/3 su `curl`: OpenSSL, nghttp3 e ngtcp2. La libreria nghttp2 necessaria per l'utilizzo di HTTP/2 su `curl` viene installata automaticamente all'atto dell'installazione di `curl`. Per l'installazione locale della nuova versione è stata seguita la procedura presentata in [19], facendo attenzione ad aggiungere il parametro `--prefix=/home/pi/ligato/NewCurl/curl` all'istruzione `./configure` di *Build curl*. In questo modo la nuova versione di `curl` è stata installata in:

`/home/pi/ligato/NewCurl/curl/bin/curl`

e per poterla utilizzare basta specificare il *path* completo.

Altro Software indispensabile in questa Fase è quello per registrare il consumo energetico durante l'esecuzione degli Esperimenti: il Software Otii (Otii App o equivalentemente la versione senza GUI avviabile tramite `otii_server.exe`). Questo, grazie allo scambio di messaggi attraverso il canale UART, consente la sincronizzazione delle tracce con l'evoluzione degli Esperimenti, in modo che si possa estrarre solamente il consumo derivante dalle singole richieste HTTP. Di questo Software sono state impiegate diverse versioni, perché ancora in continuo sviluppo. La prima versione che utilizzava il sistema era la 3.3.4, aggiornata subito nella 3.3.6. Per avere una visione dei miglioramenti introdotti da ciascuna si riportano le relative note di rilascio.

#### Release 3.3.6 (2023-07-05)

- Improved TCP-server performance and stability [Automation Toolbox]

#### Release 3.3.5 (2023-04-20)

- Offline licensing using Otii Ace (requires firmware 3.1.0) [Pro]
- Change SOC capacity by dragging in the discharge curve [Battery Toolbox]
- Added `arc_get_4wire` & `arc_set_4wire` to the TCP-server [Automation Toolbox]
- Calibrate all devices

- UX improvements
- Performance improvements
- Misc bug fixes

Release 3.3.4 (2023-03-30)

- 4-wire support [Pro]
- In-line measurement support on Ace [Pro]
- Improved battery estimator [Pro]
- TCP-server: Improved reliability [Automation Toobox]
- UX improvements
- Performance improvements
- Misc bug fixes

Dopo l'aggiornamento alla versione 3.3.6, il sistema non funzionava più in modo corretto. Nello specifico il programma lato Notebook si arrestava in modo anomalo. Confrontando le esecuzioni delle diverse versioni e debuggando il codice si è scoperto che nella nuova versione il Server Otii TCP restituiva un errore in caso di Progetto Otii non esistente, mentre nella precedente versione il comportamento era diverso. Infatti, la risposta ricevuta nella nuova versione a seguito della richiesta di apertura di un Progetto non esistente era:

```
{
    'type': 'error',
    'errorcode': 'Command timeout',
    'cmd': 'otii_open_project',
    'trans_id': '5',
    'data': {}
}
```

Nella versione precedente la risposta alla medesima richiesta era invece:

```
{
    'type': 'response',
    'cmd': 'otii_open_project',
    'trans_id': '5',
    'data': {
        'project_id': 1,
        'filename': 'C:\\\\Users\\\\Utente\\\\Desktop\\\\Ligato\\\\
                     Otii-Controller\\\\resultsUNIPI\\\\20230726_111145
                     \\\\otii\\\\traces_0'
    }
}
```

Ed il programma continuava l'esecuzione senza errori. Continuando l'analisi del codice si riesce ad identificare la funzione responsabile: `get_project` del file `manage_otii.py`, che si riporta di seguito.

```
1 def get_project(f, otii, force=True, progress=True):
2     projectName = os.getcwd()
```

```

3     splitted = f.split("/")
4     for elem in splitted:
5         projectName = os.path.join(projectName, elem)
6
7     print (f"Searching for {projectName}")
8     current_project = otii.get_active_project()
9     if current_project:
10        current_project.close(force)
11
12    try:
13        project = otii.open_project(projectName, force, progress)
14
15        assert (project)
16        print(f"Opening {projectName}")
17        return project
18    except Exception as e: #otii_exception.Otii_Exception as e:
19        error_msg = str(e)
20        print ("*****")
21        print (error_msg)
22
23        if (len(error_msg) > 0):
24            error_msg = error_msg.replace("'''", ' ')
25
26        e_json = json.loads(error_msg)
27
28        if (len (error_msg) == 0 or (len(error_msg) > 0 and
29            ↵ e_json["cmd"].strip() == "otii_open_project" and "
30            ↵ does not exist." in e_json["data"]["message"])):
31            #The target output does not exist.
32            #Close the current project and create a new one
33            current_project = otii.get_active_project()
34            if current_project:
35                current_project.close(force)
36
37            project = otii.create_project()
38            print(f"Created a new project {project}")
39
40        return project
41
42        print ("UnhandledException: " + error_msg)
43        sys.exit()

```

Si è notato, nella nuova versione, che venendo generata un'eccezione, il normale flusso di esecuzione del programma veniva alterato, andando ad eseguire il blocco `except`. All'interno di tale blocco però, il secondo `if` che compare ha una condizione che non viene soddisfatta dalla risposta fornita dal Server Otii

TCP. Infatti, il campo `data` fornito nella risposta non contiene alcun campo `message` con dentro la stringa `does not exist.`, essendo in questa nuova versione completamente vuoto (i.e. `'data': {}`). Si è provato anche a visitare la documentazione ufficiale delle API del Server Otii TCP relativamente al comando `otii_open_project` [20], ma non viene in alcun modo menzionata la situazione di Progetto non esistente. Per cui per comprendere il comportamento ed il formato delle risposte di tale comando in situazioni di errore si è dovuto procedere con l'analisi ed il *debugging* del codice in esecuzione.

Per risolvere il problema è bastato sistemare la condizione presente nel *terzo if* della `get_project()` di `manage_otii.py`. Condizione originale:

```
if (len(error_msg) == 0 or (len(error_msg) > 0 and
    ↵ e_json["cmd"].strip() == "otii_open_project" and "does not
    ↵ exist." in e_json["data"]["message"])):
```

Condizione dopo la modifica:

```
if (len(error_msg) == 0 or (len(error_msg) > 0 and
    ↵ e_json["cmd"].strip() == "otii_open_project")):
```

Questo perché, come già osservato, il campo `data` della risposta fornita dal Server Otii TCP in seguito alla richiesta di apertura di un Progetto Otii non esistente non contiene più il campo `message` o, perlomeno, in tali situazioni non viene più fornito alcun campo `message` con dentro `does not exist..`

L'ultima versione a cui è stato aggiornato il Software Otii è stata la 3.3.7. Le note di rilascio di questa versione sono:

```
Release 3.3.7 (2023-08-28)
- TCP-server settings now works in the dekstop application
    ↵ [Automation Toolbox]
- otii-server writes logs to otii_server.log [Automation
    ↵ Toolbox]
- Support Ace offline licenses in otii_server [Automation
    ↵ Toolbox]
- Fix timeout in project_close in TCP-server [Automation
    ↵ Toolbox]
- Fix timeout in project_save in TCP-server [Automation Toolbox]
- Fix connection issues in TCP-server [Automation Toolbox]
```

Si segnala che dopo l'aggiornamento a questa versione il Software Otii con GUI (i.e. App Otii), non funziona più in modo corretto. Per la corretta esecuzione del sistema, infatti, si è dovuto ricorrere all'utilizzo del Software Otii senza GUI, lanciando l'eseguibile `otii_server.exe`. Il problema è stato segnalato anche ai produttori del Software (Qoitech [21]), i quali lo hanno identificato e dovrebbero risolverlo con il rilascio di una nuova versione.

### 2.2.3. Software per la Fase di Analisi

Nella Fase di Analisi viene utilizzato largamente MATLAB per aggregare i dati ottenuti sotto forma di grafici in modo che siano di più facile e immediata comprensione. In particolare i grafici generati comprendono diversi *box-plot* ed anche modelli lineari ottenuti tramite *analisi ANCOVA*.

Un box-plot (o diagramma a scatola e baffi) è un grafico che viene utilizzato per visualizzare e confrontare gruppi di distribuzioni di dati. Questo rappresenta in modo compatto le principali statistiche dei dati:

- Linea centrale: rappresenta la mediana dei dati, ovvero il valore che divide il set di dati in due parti uguali, con il 50% dei dati al di sopra e il 50% al di sotto di essa.
- Box: rappresenta l'intervallo interquartile (IQR), che va dal primo quartile al terzo quartile. Il primo quartile è il valore che separa il 25% dei dati più bassi, mentre il terzo quartile separa il 25% dei dati più alti. La lunghezza del box rappresenta la dispersione dei dati all'interno dell'IQR.
- Baffi: si estendono dal box verso i valori minimi e massimi dei dati, ma non oltre un limite che di solito è 1.5 volte l'IQR.
- Punti outlier: i dati che si trovano al di fuori dei limiti dei baffi vengono rappresentati come un singolo punto. I valori da questi assunti sono particolarmente alti o bassi rispetto alla maggior parte dei dati e per questo vengono definiti valori anomali (i.e. outlier).

L'analisi della varianza con covariate (ANCOVA) è, invece, una tecnica statistica utilizzata per confrontare le medie di due o più gruppi, tenendo conto di una o più variabili continue aggiuntive, chiamate covariate. Le principali tipologie di variabili coinvolte in un'ANCOVA sono:

- Variabile dipendente: è la variabile che si sta cercando di analizzare e confrontare tra i gruppi.
- Variabile indipendente (o variabile di raggruppamento): suddivide i dati in gruppi o categorie distinte.
- Covariate (Covariate): sono le variabili continue aggiuntive che influenzano la variabile dipendente. Le covariate vengono utilizzate per ridurre il rumore statistico nei dati e per aumentare la precisione dell'analisi.

L'analisi ANCOVA permette poi di valutare se ci sono differenze statisticamente significative nella variabile dipendente tra i gruppi (variabile di raggruppamento), tenendo conto delle differenze nelle covariate. La probabilità che le differenze osservate tra i gruppi siano dovute al caso piuttosto che a effetti reali o significativi può essere controllata attraverso il livello di significatività, anche indicato come *p-value*. Nelle analisi ANCOVA svolte durante la Fase di Analisi dei Dati *p* è stato sempre scelto pari a 0.05.

### 3. Scenario

Per l'avvio del sistema è sufficiente seguire la procedura di seguito riportata.

1. Esecuzione di `Pi.py` lato Raspberry Pi;
2. Dopo che il Raspberry Pi visualizza la stringa “In Attesa di Messaggi (UART)” si può procedere con l'esecuzione di `serial.py` lato Notebook;
3. Inizializzazione del sistema eseguita in modo automatico Lato Notebook.

A questo punto il Notebook segnala al Raspberry Pi l'Avvio di uno specifico Esperimento, sulla base della configurazione da testare. Per descrivere in che modo procede l'esecuzione di un singolo Esperimento si farà riferimento ad una generica configurazione. Questo perché l'esecuzione dei diversi Esperimenti segue la stessa evoluzione. Nello specifico, l'esecuzione di un Esperimento segue lo schema riportato in Figura 6. Descrivendo la Figura, il Notebook segnala al Raspberry Pi l'inizio di un nuovo Esperimento e contestualmente invia i parametri della configurazione corrente (e.g. dimensione del Payload, versione del Protocollo HTTP, generazione della rete di accesso mobile, versione di curl). Si noti che la comunicazione tra Raspberry Pi e Notebook, in entrambi i versi, avviene utilizzando il canale UART. Il Raspberry Pi, a questo punto, invia al Notebook dei dettagli relativi all'Esperimento avviato, come l'URL della risorsa che andrà a scaricare e, prima di inviare la richiesta HTTP, avvisa il Notebook con un messaggio di INIZIO RICHIESTA HTTP. Adesso, utilizzando curl, il Raspberry Pi invierà una richiesta HTTP al Server per scaricare la risorsa relativa all'Esperimento in esecuzione. Il Server HTTP dunque, risponderà con la risorsa richiesta. Si noti che la comunicazione tra Raspberry Pi e Server non è simmetrica, perché la richiesta inviata dal Raspberry Pi non contiene alcun payload, mentre la risposta del Server avrà, in generale, payload di diversa dimensione. Subito dopo la ricezione della risorsa il Raspberry Pi segnala al Notebook la fine della richiesta HTTP ed anche il percorso in cui andrà a caricare le informazioni di *logging* sul Server Remoto. In seguito al caricamento di queste informazioni, il Raspberry Pi attenderà 25 secondi prima di segnalare al Notebook il completamento dell'Esperimento. Questa attesa, come si capirà meglio dal successivo paragrafo **3.2.3.**, consente al modulo radio del Raspberry Pi di tornare nello stato di *inattività* prima dell'inizio di un nuovo Esperimento. È necessario infatti misurare il consumo energetico della transizione dallo stato di *inattività* allo stato di *connessione* in ogni Esperimento. Con il risveglio il Raspberry Pi segnala la fine dell'Esperimento ed anche la possibilità di poterne iniziare uno nuovo. Il Notebook scarica dal Server quanto caricato dal Raspberry Pi e procede dunque con l'esecuzione di un nuovo Esperimento.

Il comportamento del sistema, dopo la la preliminare inizializzazione, è dunque ciclico. Il consumo energetico che viene registrato ad ogni Esperimento è relativo all'intervallo indicato in rosso nello schema. Si noti che grazie al canale UART e ai messaggi di INIZIO e FINE richiesta HTTP è possibile sincronizzare l'evoluzione dell'Esperimento con la traccia del consumo energetico registrata dal Software Otii in esecuzione sul Notebook.

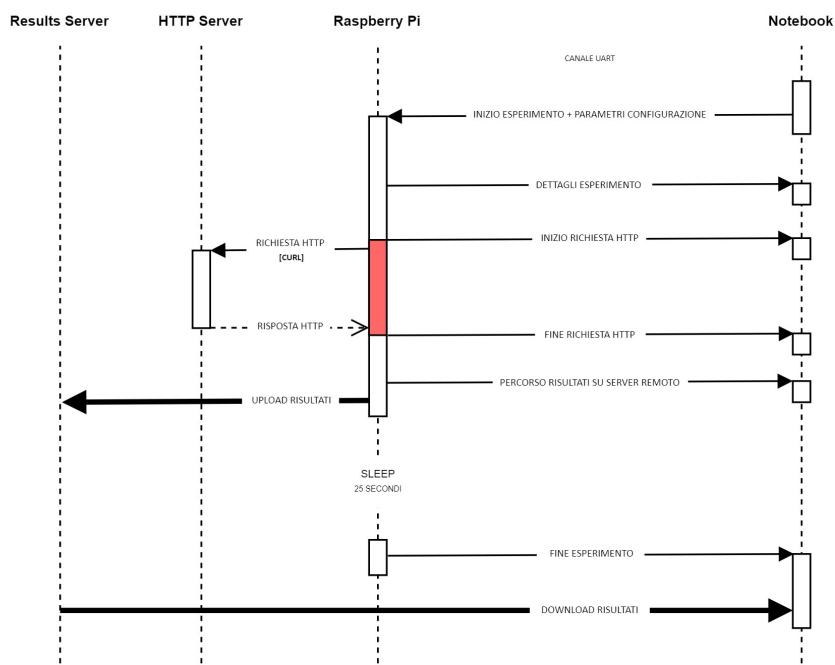


Figura 6: Schema relativo all’evoluzione di un Esperimento.

### 3.1. Qualche Accorgimento

Durante l'esecuzione degli Esperimenti bisogna disconnettere il Raspberry Pi da tutte le periferiche non necessarie per evitare che la valutazione del consumo energetico venga alterata. Nel caso in questione due sono i dispositivi da scollegare: il monitor e la tastiera. Questo viene fatto per evitare che tramite il connettore HDMI vi sia un assorbimento aggiuntivo di corrente da parte del Raspberry Pi. Relativamente alla tastiera, invece, vi sarebbe stata una dissipazione aggiuntiva di corrente per l'alimentazione della periferica. La valutazione da compiere, infatti, ha lo scopo di registrare *soltanto* il consumo della scheda di rete e del Raspberry Pi durante l'esecuzione di richieste HTTP. In questo modo i risultati che si otterranno saranno direttamente comparabili con l'universo dell'IoT. Difatti, gli unici elementi che compongono un dispositivo IoT sono in genere un Single-Board Computer ed una scheda di rete.

Per prevenire possibili meccanismi di caching delle risorse scaricate dal Raspberry Pi, `curl` viene eseguito con la seguente opzione aggiuntiva:

```
-H "Cache-Control: no-cache, no-store, private, max-age=0,  
    s-maxage=0"
```

Tale opzione aggiunge un *header* personalizzato alla richiesta HTTP dove, attraverso il campo `Cache-Control`, si indica che la risposta non dovrebbe essere memorizzata nella cache e nemmeno da intermediari. In più è stata prevista nel Server HTTP una risorsa diversa per ciascun Esperimento.

Importante è stato anche il luogo in cui posizionare il sistema, perché non in tutte le aree dove è stato effettuato lo studio vi era una copertura 5G. Per identificare una zona adeguata si sono seguiti i seguenti passi:

1. Ricerca sul Web di una mappa di copertura del 5G relativa all'operatore mobile *iload* [22].
2. Con l'aiuto di uno smartphone 5G (un Pixel 5 [23]) e sfruttando le informazioni fornite dalla mappa si è andati alla ricerca di una zona in cui lo smartphone riuscisse a connettersi alla rete 5G.
3. Dopo aver identificato un'area con tali caratteristiche si è trasferito tutto il sistema per verificare l'effettiva connessione al 5G.

Al punto 2 non si è direttamente fatto uso del sistema perché per poter funzionare correttamente ha bisogno di accedere in modo costante alla rete elettrica. Per questo motivo non è una configurazione che si può definire mobile. Le caratteristiche dell'area identificata sono quelle di essere vicina ad una finestra e di affacciarsi su una strada coperta dal 5G (secondo la mappa trovata).

### 3.2. Dettagli della RAN

Di seguito vengono presentati dei dettagli relativi alla *Radio Access Network* (RAN) di cui si fa uso nell'area identificata per l'esecuzione degli Esperimenti.

### 3.2.1. Metodologia

Per la visualizzazione delle informazioni relative alla rete di accesso radio vengono utilizzati dei comandi AT per comunicare con il modulo radio connesso al Raspberry Pi. In particolare i comandi usati a questo proposito sono:

- AT+CNMP: permette di visualizzare (AT+CNMP?) o impostare (AT+CNMP=X) la generazione della tecnologia radio preferita. I valori ammessi sono:

```
2 Automatic
13 GSM Only
14 WCDMA Only
38 LTE Only
71 NR5G
19 GSM+WCDMA Only
48 Any modes but LTE
39 GSM+WCDMA+LTE Only
51 GSM+LTE Only
54 WCDMA+LTE Only
55 WCDMA+LTE+NR5G
109 LTE+NR5G
100 UNKNOW
```

- AT+CPSI?: mostra varie informazioni sulla RAN a cui è connesso il modulo radio. Tra queste: stato di connessione del modulo radio, generazione della tecnologia radio, banda di frequenza, qualità del segnale ricevuto.

Maggiori dettagli su questi e su molti altri comandi AT possono essere trovati sul manuale della scheda di rete [24].

### 3.2.2. Quarta Generazione - LTE

Utilizzando le tecnologie di accesso radio di *quarta* generazione o LTE (*Long Term Evolution*) si hanno i seguenti dettagli: Frequenza Portante di 2.6 GHz appartenente alla Banda 7, dove si fa uso del FDD (*Frequency Division Duplex*) [25].

### 3.2.3. Quinta Generazione - NR

Relativamente alle tecnologie di accesso radio di *quinta* generazione o NR (*New Radio*) si ha:

- Architettura Non-Standalone (NSA).
- Frequenza Portante di 3.5 GHz appartenente alla Banda sub-6 GHz.
- Banda n78, dove la modalità di duplex è TDD (*Time Division Duplex*) [26].

Considerando quanto presentato nell'introduzione è chiaro che il 5G di cui si fa uso non è un 5G al pieno del suo potenziale. Al momento dello studio però *iliad* fornisce nella zona considerata solamente il 5G NSA, quindi la valutazione

del consumo energetico dovrà essere fatta adoperando questa modalità di distribuzione. Per tale motivo si procede con l'approfondire [8, 9] il modo in cui avviene la gestione energetica nel 5G NSA. Il comportamento che viene seguito è schematizzato dalla macchina a stati riportata in Figura 7. In questa sono presenti due stati:

- RRC\_IDLE: nessuna TX / RX;
- RRC\_CONNECTED: TX / RX in corso.

Si noti che gli stati RRC (*Radio Resource Control*) del 5G NSA e quelli del 4G sono esattamente gli stessi. Il funzionamento della macchina a stati può essere espresso come segue:

1. Inizializzazione dallo stato RRC\_IDLE.
2. Quando l'UE ha un pacchetto da trasmettere (TX), invia un RRC Connection Request alla stazione eNB (stazione base 4G) ed entra nello stato RRC\_CONNECTED dopo  $T_{LTE\_pro}$  (*LTE promotion delay*) millisecondi.
3. Nello stato RRC\_CONNECTED l'UE può trovarsi in una di due modalità:
  - connessione LTE;
  - connessione NR.
4. Se mentre si è nello stato RRC\_CONNECTED non vengono ricevuti più dati per un periodo di tempo pari a  $T_{inac}$  (*DRX inactivity timer*) millisecondi, il modulo radio tornerà nello stato RRC\_IDLE dopo un'ulteriore attesa di  $T_{tail}$  (*4G/5G traffic pattern tail cycle*) millisecondi.

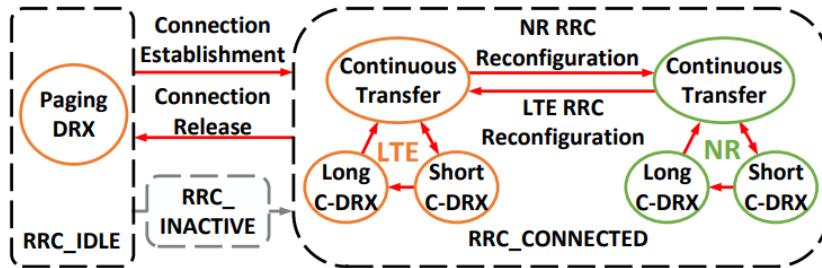


Figura 7: Macchina a Stati relativa alla gestione energetica del 5G NSA (Figura tratta da [9]).

Dentro la macchina a stati RRC il 5G adotta il meccanismo della discontinua ricezione (*Discontinuous Reception - DRX*) per il risparmio energetico. Secondo tale meccanismo l'UE rimane in *sleep mode* di default e si sveglia solamente per

ascoltare il canale per un piccolo intervallo di tempo,  $T_{on}$  (*On-duration timer*), all'inizio di ogni periodo  $T_{cycle}$ . Esistono tre cicli DRX:

- `paging` DRX;
- `short C-DRX`;
- `long C-DRX`.

La transizione dallo stato `RRC_CONNECTED` allo stato `RRC_IDLE` prevede il passaggio attraverso una fase definita di *long-tail*. Tale fase dura circa 10s nel 4G, mentre arriva a circa 20s nel 5G. Da qui l'attesa di 25 secondi che compare in Figura 6 (per tenere in considerazione eventuali ritardi aggiuntivi introdotti dalle particolari implementazioni vengono aggiunti 5 secondi). Questa durata notevole nel 5G, che porta ad uno spreco addizionale di energia, è dovuta all'architettura NSA. In particolare, per completare la transizione da `NR_RRC_CONNECTED` a `RRC_IDLE`, il modulo 5G deve prima passare attraverso la macchina a stati del 4G tramite una LTE RRC Reconfiguration. Questo processo è equivalente all'attivazione di un nuovo periodo di *tail*.

I ritardi per la promozione da `RRC_IDLE` a 4G e 5G sono in genere diversi, perché si hanno i seguenti cambiamenti di stato:

`LTE_RRC_IDLE` → `LTE_RRC_CONNECTED` → `NR_RRC_CONNECTED`

Nel 5G SA, invece, lo UE raggiunge direttamente `NR_RRC_CONNECTED`. Altra differenza che si ha in questa architettura è data dall'introduzione di un nuovo stato denominato `RRC_INACTIVE` (che per completezza è riportato anche nella Figura 7). Lo scopo principale di questo stato, simile ad uno stato a basso consumo, è quello di fornire un meccanismo efficiente per il modulo radio di *dormire*, salvando energia, e allo stesso tempo di permettere una veloce e leggera transizione allo stato `RRC_CONNECTED`, migliorando la latenza e riducendo il tempo di risveglio. Si veda a tale scopo la macchina a stati del 5G SA riportata in Figura 4.2.1-1 nelle specifiche NR (3GPP TS 38.331 version 16.3.1 Release 16) [27].

### 3.3. Diverse Condizioni di Rete

Per valutare l'impatto dello stato della rete sul consumo energetico vengono simulate diverse condizioni avverse modificando in modo controllato la banda a disposizione ed il ritardo di risposta del Server HTTP. A questo scopo si è utilizzato il comando Linux `tc`, acronimo di *traffic control*.

Per imporre un ritardo aggiuntivo *ed* una limitazione di banda ad una interfaccia si usa:

```
sudo tc qdisc add dev {interface} root netem delay {delay}ms rate
    ↳ {bandwidth}mbit
```

In questo modo si impone un ritardo aggiuntivo `delay` espresso in millisecondi (i.e. ms) ed una limitazione alla banda `bandwidth` espressa in Megabits al secondo (i.e. mbit).

Per ripristinare il ritardo e la banda originari è sufficiente eliminare la regola impostata:

```
sudo tc qdisc del dev {interface} root
```

Per ulteriori approfondimenti si consulti anche la sezione del manuale di `tc` relativa a `netem` [28].

## 4. Esperimenti

L'esecuzione dei diversi Esperimenti rappresenta la Fase di Raccolta Dati. Tale Fase può essere ulteriormente suddivisa in:

1. Preliminare;
2. Principale;
3. Aggiuntiva.

L'ordine di esecuzione delle diverse Sotto-Fasi di Raccolta Dati corrisponde con l'ordine con cui sono state presentate. Nello specifico, la Raccolta Dati Preliminare (1) serve a valutare l'impatto energetico delle diverse implementazioni di `curl`. I risultati di questa Fase serviranno anche a scegliere la versione di `curl` da utilizzare per la Raccolta Dati Principale. In 1 si fa uso solamente del 4G e la dimensione del payload di risposta è sempre pari ad 1MB. Questo perché si vuole isolare l'impatto delle diverse versioni di `curl`. In particolare vengono testate le versioni:

- 7.85.0-DEV che per brevità si indicherà spesso come 7.85.0 o anche come 7;
- 8.2.1-DEV che anche in questo caso per brevità si indicherà come 8.2.1 o semplicemente come 8.

Utilizzando queste due versioni si andrà a vedere in che modo cambia il comportamento delle richieste HTTP/2 e HTTP/3. Per semplicità anche la condizioni della rete non verranno alterate in questa prima Sotto-Fase.

La Raccolta Dati Principale (2) come lascia immaginare il nome è la più completa e durante questa, facendo uso della versione più efficiente di `curl` (determinata grazie alla raccolta di dati Preliminare), si andrà a valutare il consumo energetico delle diverse versioni di HTTP usando le reti 4G e 5G. La dimensione del payload di risposta sarà variabile andando ad assumere particolari valori, rappresentativi di diversi usi del protocollo HTTP: dimensione piccola (i.e. 128B), media (512KB) e media-grande (4MB). Anche le condizioni di rete verranno alterate, andando a valutare diversi scenari più o meno drastici.

La Raccolta Dati Aggiuntiva (3) inizialmente non era stata prevista ed è stata introdotta dopo aver riscontrato la necessità di voler valutare il comportamento energetico di 4G e 5G su payload di grandi dimensioni (64MB). Per ridurre il numero di configurazioni e conseguentemente il tempo necessario per la Raccolta dei Dati in relazione alle condizioni di rete viene alterato solamente il ritardo, non imponendo alcuna restrizione alla banda (anche perché grazie alla raccolta precedente si è notato che non influiva in modo particolare sul consumo energetico).

Per tutte e tre le Sotto-Fasi di Raccolta Dati vengono eseguiti, per una data configurazione, 30 Esperimenti.

## 4.1. Il File config.ini

Per modificare i parametri di configurazione del sistema in modo intuitivo e veloce è stato previsto da [1] il file `config.ini`, la cui struttura viene definita in `template_config.ini`. Le sezioni in esso presenti hanno il seguente significato:

- `[results_server]`: Dettagli del Server remoto in cui il Raspberry Pi carica i propri Risultati;
- `[http_server]`: Dettagli del Server HTTP da cui il Raspberry Pi scarica le diverse Risorse;
- `[otii_param]`: Dettagli relativi al Software Otii;
- `[exp_param]`: Dettagli sulle diverse configurazioni degli Esperimenti.

Tutti i Valori Ammessi per i campi di `[exp_param]` sono:

```
curl_versions = curl17.85.0, curl18.2.1
http_versions = http1.1, http2, http3
payload_dimensions = 128B, 512KB, 1MB, 4MB, 64MB
servers = caddyvm
radio_generations = 4G, 5G
bandwidths = 100%%, 25, 10
delays = 0, 20, 40, 60, 80
```

Le unità di misura dei valori specificati in `bandwidths` sono Megabits al secondo (Mbps). Il valore 100% rappresenta la situazione in cui non si impone alcuna limitazione sulla banda del Server HTTP. Si noti che viene anteposto un altro % al simbolo di % in modo che possa essere considerato. I valori del campo `delays` sono invece espressi in millisecondi (ms).

Nel seguito si riportano i valori dei campi di `[exp_param]` per le diverse Sotto-Fasi di Raccolta Dati.

- Raccolta Dati Preliminare:

```
curl_versions = curl17.85.0, curl18.2.1
http_versions = http2, http3
payload_dimensions = 1MB
servers = caddyvm
radio_generations = 4G
bandwidths = 100%
delays = 0
```

- Raccolta Dati Principale:

```
curl_versions = curl18.2.1
http_versions = http2, http3
payload_dimensions = 128B, 512KB, 4MB
servers = caddyvm
radio_generations = 4G, 5G
bandwidths = 100%%, 25, 10
```

```
delays = 0, 20, 40, 60, 80
```

- Raccolta Dati Aggiuntiva:

```
curl_versions = curl8.2.1
http_versions = http2, http3
payload_dimensions = 64MB
servers = caddyvm
radio_generations = 4G, 5G
bandwidths = 100%%
delays = 0, 20, 40, 60, 80
```

## 5. Problemi e Soluzioni

Di seguito si riportano i problemi riscontrati durante lo svolgimento dello studio e le relative soluzioni adottate.

### 5.1. Inaffidabilità del Canale UART

Raspberry Pi e Notebook comunicano tra di loro sfruttando il canale UART realizzato dall’Ottii Arc Pro. Tale canale, in seguito a diverse verifiche, si è rivelato essere inaffidabile in quanto possono verificarsi (anche abbastanza frequentemente) le seguenti situazioni di errore:

- introduzione di errori all’interno dei messaggi scambiati;
- perdita di messaggi;
- messaggi arrivati in ritardo, in seguito alla consegna di messaggi che temporalmente sarebbero dovuti arrivare dopo.

In Figura 8 sono riportati due esempi in cui si ha la corruzione di due messaggi, con l’introduzione di un carattere apparentemente casuale.

```
Received 4 new messages on the rx (UART) channel
Messaggio ricevuto su 'rx': {'value': 'EXP START', 'timestamp': 6.132}
Messaggio ricevuto su 'rx': {'value': 'REQ_START:0', 'timestamp': 6.142}
Messaggio ricevuto su 'rx': {'value': 'REQ_STOP:0', 'timestamp': 6.481}
Messaggio ricevuto su 'rx': {'value': 'UPLOAD PATH:ubuntu@131.114.73.3:/home/ubuntu/Ligato/risultatiRaspberry/2023-08-02/curl8.2.1/http2/2_caddy_vm_medium_00
1', 'timestamp': 6.426}
```

(a) Ricezione di un messaggio corrotto: il carattere ”O” evidenziato non è stato inviato dal Raspberry.

```
Messaggio ricevuto su 'rx': {'value': 'REQ_STOP:0', 'timestamp': 7.982}
Received 1 new messages on the rx (UART) channel
Messaggio ricevuto su 'rx': {'value': 'UPLOAD PATH:ubuntu@131.114.73.3:/home/ubuntu/Ligato/risultatiRaspberry/2023-08-02/curl8.2.1/http3/001_caddy_vm_noresou
rce/', 'timestamp': 8.048}
```

(b) Ricezione di un messaggio corrotto: il carattere ”p” evidenziato non è stato inviato dal Raspberry.

Figura 8: Statistiche sull’utilizzo della CPU per le versioni di curl considerate.

Facendo ulteriori indagini però si scopre che i caratteri (o talvolta porzioni di stringa) introdotti durante lo scambio sul canale non sono aleatori, ma funzione dei messaggi precedentemente inviati sul canale. Si nota anche che gli errori si verificano *più facilmente* quando il programma è stato avviato da *poco*, mentre tendono a verificarsi di *rado* quando il programma è stato avviato da *molto* tempo.

#### 5.1.1. Soluzione (RDT 4.0)

Non avendo trovato la causa (pur avendo dedicato molto tempo ed aver provato con diverse strade) che porta il canale UART ad essere inaffidabile, è stato definito un protocollo di Reliable Data Transfer su misura, per far sì che lo scambio di messaggi tra Raspberry Pi e Notebook avvenga senza errori. In particolare, traendo ispirazione dalla versione 3.0 del meccanismo di Reliable

Data Transfer del protocollo TCP presentata nel corso di *Reti Informatiche*, viene definita una nuova versione, la 4.0. I miglioramenti introdotti in questa nuova versione permettono di controllare uno scenario più complesso di quello che viene gestito nella precedente versione. In tale scenario non vi è più una distinzione netta tra *sender* (i.e. mittente) e *receiver* (i.e. destinatario), ma le due entità possono assumere il ruolo dell'uno o dell'altro a seconda dello stato in cui si trovano. Non vi saranno perciò più un *sender* ed un *receiver* che si scambiano dei messaggi ma, più in generale, due entità che prenderanno il nome di *sendeiver*, sincrasi dei termini *sender* e *receiver*. Questo perché nel corso della loro esecuzione potranno essere sia mittenti di messaggi che destinatari degli stessi. Nel RDT 4.0, i meccanismi che venivano impiegati nella precedente versione vengono mantenuti tutti. Oltre a quelli si vanno ad introdurre degli ulteriori accorgimenti per rendere possibile lo scenario descritto.

(i) Il primo consiste nel far uso di una singola variabile, all'interno di una entità, per mantenere il valore del numero di sequenza corrente, e di non utilizzare due variabili distinte, che in genere avranno una diversa evoluzione, da associate agli stati di *sender* e *receiver* in cui può trovarsi l'entità *sendeiver*. In questo modo l'evoluzione della variabile sarà indipendente dallo stato in cui si troverà il *sendeiver*. (ii) La condizione che permette ad un *sendeiver* nello stato di *sender* di proseguire con l'esecuzione dopo l'invio di un messaggio non è più solamente una, ovvero la ricezione di un ACK non corrotto con il numero di sequenza corrispondente al messaggio inviato. Nella RDT 4.0 vi è un'altra situazione che permette al *sendeiver* di capire che il messaggio inviato è stato ricevuto correttamente. Tale situazione si verifica quando il *sendeiver* riceve:

1. un messaggio non corrotto;
2. il messaggio ricevuto non è un ACK relativo al precedente numero di sequenza;
3. il messaggio ricevuto è relativo al prossimo numero di sequenza.

È chiaro che il verificarsi di una situazione del genere vuol dire che il messaggio inviato all'altra entità è arrivato in modo corretto, perché può essersi verificata solamente una sequenza di eventi comparabile con la seguente:

1. l'altra entità si trova nello stato di *receiver* e riceve in modo corretto il messaggio;
2. tale entità procede con l'invio di un messaggio di ACK relativo al numero di sequenza del messaggio ricevuto;
3. sempre questa entità, aggiorna il proprio numero di sequenza corrente e procede con l'esecuzione portandosi nello stato di *sender*;
4. il messaggio di ACK viene perso *oppure* viene corrotto dal canale inaffidabile;
5. l'entità che ha inviato il messaggio di ACK adesso invia un messaggio con il nuovo numero di sequenza;
6. il messaggio viene ricevuto in modo corretto dall'entità che ancora si trova nello stato di *sender*.

Prima di presentare la terza ed ultima accortezza del RDT 4.0, si riporta in

Figura 9 un diagramma di sequenza relativo allo scambio di messaggi che può avvenire durante l'avvio di un nuovo Esperimento. È importante precisare che tale diagramma di sequenza rappresenta una situazione possibile, che potrebbe verificarsi all'interno del sistema reale (i.e. si è realmente verificata). Lo scambio di messaggi presentato tiene in considerazione solamente i primi due accorgimenti introdotti dal RDT 4.0 e serve proprio a giustificare l'importanza della piccola modifica introdotta dal terzo accorgimento.

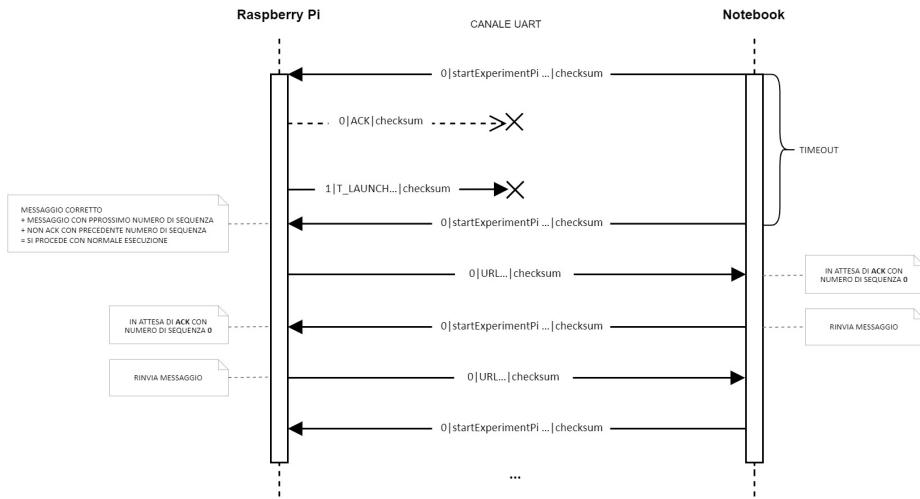


Figura 9: Diagramma di Sequenza: RDT 4.0 (versione incompleta) in funzione.

È chiaro che manca ancora qualcosa perché, con due sole perdite / corruzioni avvenute una di seguito all'altra, il meccanismo non funziona più e manda l'intero sistema in una situazione di stallo. (iii) L'ultimo dettaglio da sistemare farà in modo che la precedente situazione non si possa verificare. Bisogna estendere l'intervallo di valori ammessi per i numeri di sequenza di una unità. Dunque, il nuovo intervallo sarà: [0, 1, 2]. Andando a considerare la situazione illustrata nel precedente diagramma di sequenza, questa volta si avrà un comportamento diverso, come mostra la Figura 10.

Grazie a questa nuova versione del meccanismo di Reliable Data Transfer, si può dire che l'inaffidabilità del canale UART è stata *completamente* arginata. Infatti, durante la Fase di Sperimentazione principale, il sistema è stato in esecuzione, senza mai fermarsi, per un totale di circa 76 ore (i.e. più di 3 giorni), eseguendo complessivamente 5400 Esperimenti. Bisogna comunque menzionare anche il contributo della *nuova gestione delle eccezioni*, senza la quale il sistema non sarebbe mai stato in grado di raggiungere una tale stabilità.

#### 5.1.1.1. Meccanismi del RDT 3.0

Come detto, il RDT 4.0 nasce come estensione della versione 3.0 e per questo

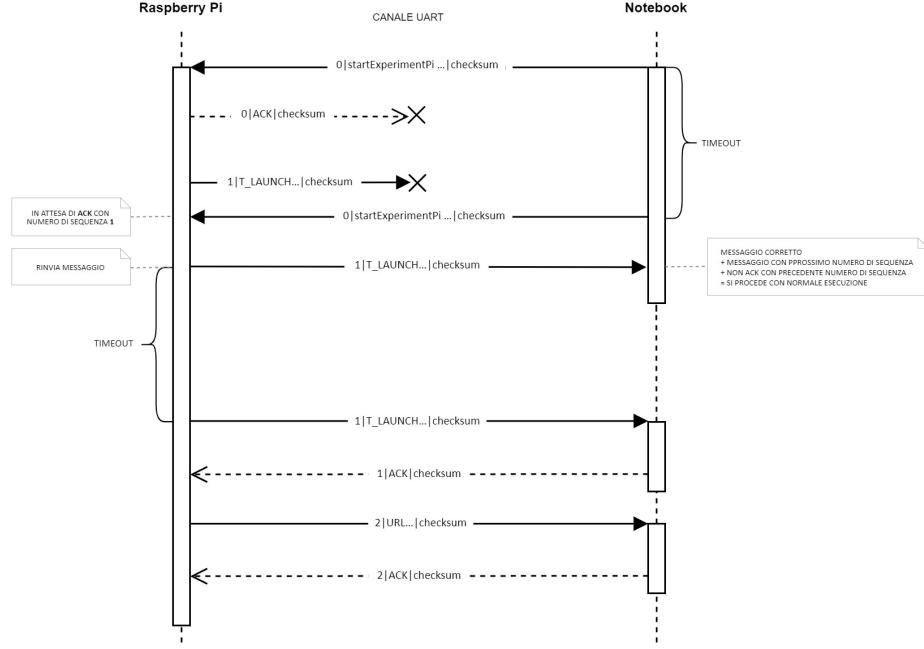


Figura 10: Diagramma di Sequenza: RDT 4.0 in funzione.

fa uso di tutti i meccanismi che vengono definiti in tale versione:

- **Error Detection:** in coda al messaggio viene appeso un checksum ottenuto calcolando l'MD5 del messaggio stesso.
- **Sequence Number:** ad ogni messaggio viene associato un numero di sequenza per identificare eventuali messaggi duplicati ricevuti dal destinatario. Tale numero di sequenza può essere un valore compreso nel seguente intervallo binario  $[0, 1]$ . Si fa uso di solamente due numeri di sequenza perché lo scambio di messaggi è sequenziale e non si procede con lo scambio di altri messaggi finché il messaggio da inviare non è stato correttamente inviato e confermato dal destinatario. Il numero di sequenza viene inserito in testa al messaggio testuale.

Considerando i due meccanismi presentati fino ad ora, i messaggi che verranno inviati attraverso il canale UART avranno una forma del genere:

`Sequence_Number | Message | Checksum`

Questo se il carattere separatore utilizzato corrisponde a `|`.

- **Feedback:** il destinatario della comunicazione invia al mittente un messaggio di ACK (i.e. di Acknowledgment) relativo all'ultimo messaggio correttamente ricevuto, specificando il numero di sequenza che il messaggio

conteneva. Se il messaggio appena ricevuto è arrivato danneggiato allora si riconosce come ultimo messaggio ricevuto correttamente il precedente. In questo modo il mittente è in grado di capire che il messaggio inviato è stato ricevuto danneggiato dal destinatario.

- **Retransmission:** la ritrasmissione di un messaggio si verifica nel caso in cui lo stesso non sia stato ricevuto in modo corretto dal destinatario. Anche in tutti i casi in cui non si riesce a discernere se il messaggio è stato ricevuto in modo corretto o meno (e.g. ACK danneggiato, ACK non ricevuto) si ha comunque una ritrammissione del messaggio. Il numero di sequenza associato al messaggio permetterà al destinatario di scartare eventuali messaggi arrivati ripetuti.
- **Timeout:** in caso di perdita di messaggi, o equivalentemente di ACK, lo scattare di un Timer opportunamente configurato permette al mittente il rinvio del messaggio corrente, evitando in questo modo un'attesa indefinita e una conseguente situazione di stallo.

## 5.2. Una strana Eccezione

Si verifica occasionalmente una Eccezione di `timed out` che causa l'arresto anomalo del sistema ed in particolare del programma in esecuzione sul Notebook. Si riporta di seguito l'output che viene visualizzato in queste situazioni.

```
Dettagli Configurazione Attuale:  
Dimensione Payload = noresource  
Struttura = caddy_vm  
Versione curl = curl8.2.1  
Versione HTTP = http3  
  
Unhandled Exception timed out.  
  
Closing the project...  
Unhandled Inner Exception timed out.  
Dettagli Configurazione Attuale:  
Dimensione Payload = noresource  
Struttura = caddy_vm  
Versione curl = curl17.85.0  
Versione HTTP = http3  
  
Error: last recording not retrieved...  
Error: last recording not retrieved...  
Error: last recording not retrieved...  
Traceback (most recent call last):  
  File  
    "c:\Users\Utente\Desktop\Ligato\Otii-Controller\serial.py",  
    line 117, in launch_experiment
```

```

        experiment.project.start_recording()
File "C:\Users\Utente\AppData\Local\Packages\
↳ PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\
↳ local-packages\Python310\site-packages\otii_tcp_client\
↳ project.py", line 129, in start_recording
    response = self.connection.send_and_receive(request)
File "C:\Users\Utente\AppData\Local\Packages\
↳ PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\
↳ local-packages\Python310\
↳ site-packages\otii_tcp_client\otii_connection.py", line 133,
↳ in send_and_receive
    data = self.receive_response(timeout,
↳ request["trans_id"])
File "C:\Users\Utente\AppData\Local\Packages\
↳ PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\
↳ local-packages\Python310\
↳ site-packages\otii_tcp_client\otii_connection.py", line 107,
↳ in receive_response
    raise Exception("Transaction id mismatch")
Exception: Transaction id mismatch

```

During handling of the above exception, another exception  
↳ occurred

Il problema è, però, che tale Eccezione non si verifica sempre in corrispondenza dello stesso punto, come dimostrato dall'output generato in seguito al ri-verificarsi della medesima Eccezione.

```

Received 1 new messages on the rx (UART) channel
Messaggio ricevuto su 'rx': {'value': 'OK', 'timestamp':
↳ 17.217}
Messaggio ricevuto in modo corretto:
↳ UPLOAD_PATH:ubuntu@131.114.73.3:/home/ubuntu/Ligato/
↳ risultatiRaspberry/2023-08-04/curl7.85.0/http2/
↳ 016_caddy_vm_noresource/
Unhandled Exception timed out.

Closing the project...
Initialize the environment...
Searching for
↳ C:\Users\Utente\Desktop\Ligato\Otii-Controller\Risultati\
↳ 2023-08-04_13.58.45\otii\Progetto_1
Opening
↳ C:\Users\Utente\Desktop\Ligato\Otii-Controller\Risultati\
↳ 2023-08-04_13.58.45\otii\Progetto_1

Arc device found.

```

### 5.2.1. Soluzione

Dopo l'aggiornamento del Software Otii alla versione 3.3.7 l'Eccezione non si è più presentata. Difatti, tra le note dell'aggiornamento si fa riferimento alla correzione di un `timeout` che si verificava in seguito alle invocazioni di `project_close` e `project_save` del Server Otii TCP. Si riportano per comodità anche qui le note di rilascio di questa versione:

Release 3.3.7 (2023-08-28)

- TCP-server settings now works in the dekstop application
  - ↳ [Automation Toolbox]
- otii-server writes logs to otii\_server.log [Automation Toolbox]
- Support Ace offline licenses in otii\_server [Automation Toolbox]
- Fix timeout in project\_close in TCP-server [Automation Toolbox]
- Fix timeout in project\_save in TCP-server [Automation Toolbox]
- Fix connection issues in TCP-server [Automation Toolbox]

## 5.3. Un BUG dell'Ambiente Otii

È stato rilevato un BUG all'interno dell'Ambiente Otii (nell'Otii APP ed anche in `otii_server.exe`). Per cercare di arginare l'impatto che tale BUG può avere sul sistema si è cercato in primis di caratterizzarlo. Una volta caratterizzato sarà facile riconoscerlo ed attuare le operazioni necessarie per riprendere l'esecuzione del sistema. Caratterizzazione del BUG:

- Registrazione Avviata senza alcun problema.
- Il Raspberry Pi riceve il messaggio di Avvio Esperimento (i.e. di `startExperimentPi`), dunque la comunicazione Notebook->RaspberryPi è possibile.
- L'ACK relativo al messaggio di Avvio Esperimento non arriva.
- Non arriva alcun messaggio inviato dal RaspberryPi e diretto verso il Notebook.
- Anche dall'Otii App, nel log UART, non vengono visualizzati i messaggi inviati dal Raspberry Pi (in tale log anche durante il normale funzionamento degli Esperimenti compaiono solo i messaggi `RaspberryPi->Notebook`).
- Dal Software Otii si nota che la Registrazione non sta andando avanti, come se si fosse bloccata appena è stata avviata.
- Il Notebook dunque non ricevendo il messaggio di ACK e dopo che il TIMER scatta rimanda il messaggio di Avvio Esperimento, che controllando sul Raspberry Pi arriva senza problemi.
- Il Raspberry Pi però ha già avviato l'Esperimento e sta inviando verso il Notebook il messaggio di inizio Esperimento (i.e. `T_LAUNCH`), per proseguire però ha bisogno che il Notebook invii il relativo messaggio di ACK.
- Il sistema è in stallo e lato PC si verificano tanti TIMEOUT uno di seguito

all’altro.

### 5.3.1. Soluzione

Sfruttando le informazioni derivanti dalla caratterizzazione del BUG, dopo aver identificato una tale situazione di blocco, si cerca di recuperare il sistema riavviando il Progetto ed anche l’Esperimento corrente. In particolare, ci si accorge che si è verificata la situazione descritta quando lato Notebook si verificano *molti* (i.e. 7 nel codice) TIMEOUT uno di seguito all’altro all’inizio di un nuovo Esperimento. A questo punto viene lanciata una Eccezione *user defined* denominata `BugSoftwareOtii`. Per gestire questa eccezione si chiude e si riapre il Progetto Otii corrente e si fa ripartire l’Esperimento con la stessa configurazione. In seguito a queste operazioni il Software Otii riprende a funzionare in modo corretto. A sbloccare la situazione in cui si trova il sistema subito dopo il riavvio dell’Esperimento ci pensa il codice del RDT4. Infatti, subito dopo il riavvio dell’Esperimento, che avviene nel programma del Notebook, quest’ultimo tenterà di rimandare al Raspberry Pi il messaggio di Avvio Esperimento. Il Raspberry Pi, però, è ancora rimasto nello stato di Esperimento avviato e attesa di ACK da parte del Notebook. Questo perché le operazioni di gestione Eccezione vengono eseguite solamente lato Notebook. A questo punto dunque, il Notebook tenterà di inviare il messaggio di Avvio Esperimento e il Raspberry Pi invierà al Notebook non il messaggio di ACK relativo al messaggio di Avvio Esperimento, ma il messaggio di Esperimento Avviato (perché trovandosi nello stato di mittente avrà associato anche lui un TIMER allo scattare del quale il messaggio verrà rinviauto) che ha dunque un numero di sequenza successivo rispetto a quello inviato dal Notebook. Grazie alle capacità del RDT4, il Notebook si renderà conto che il Raspberry Pi si trova già in uno stato successivo al messaggio che sta inviando e dunque può essere certo che il Raspberry Pi abbia ricevuto in modo corretto il messaggio di Avvio Esperimento. Il Notebook per questo può proseguire con la propria esecuzione. Il TIMER lato Raspberry Pi, scattando ancora una volta, permetterà il rinvio del messaggio di inizio Esperimento e questa volta il Notebook trovandosi nello stato corretto riuscirà ad inviare il messaggio di ACK. Il sistema dunque non è più in una situazione di stallo e può proseguire con la normale conduzione dell’Esperimento corrente. La situazione descritta è rappresentata nel diagramma di sequenza raffigurato in Figura 11.

### 5.3.2. Una Nuova Gestione delle Eccezioni

Per evitare che problemi del genere mandino in blocco l’intero sistema, è stata definita *una nuova gestione delle Eccezioni*. Infatti, il nuovo codice per la gestione delle Eccezioni presente lato Notebook (in `serial.py`) permette di ripristinare l’esecuzione del sistema in modo automatico anche in seguito alla verifica di situazioni di errore. Si è deciso di dedicare del tempo anche a questa parte del codice per permettere un’esecuzione del sistema del tutto autonoma, in modo che dopo l’avvio degli Esperimenti si potesse avere una raccolta di dati

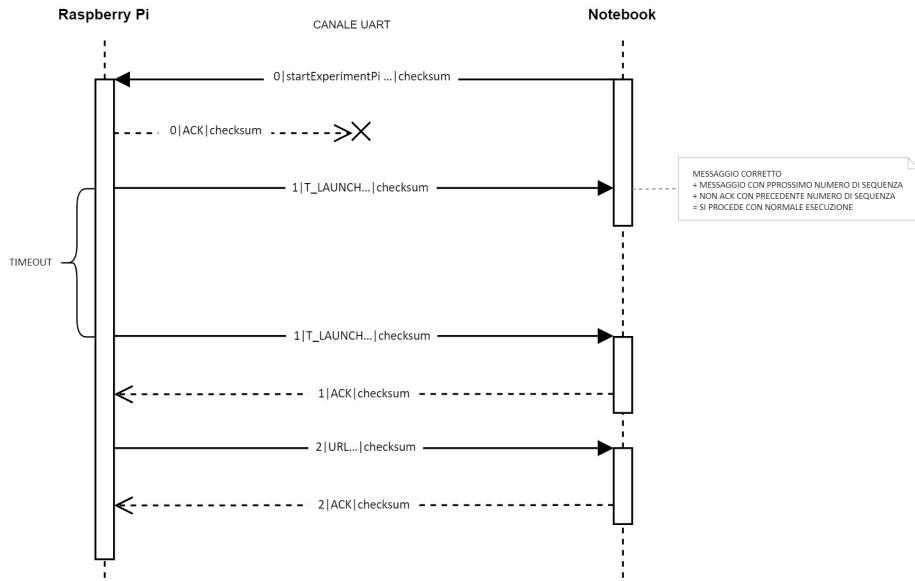


Figura 11: Diagramma di Sequenza: il RDT 4.0 permette di uscire dallo stallo che altrimenti si avrebbe dopo il ripristino del sistema a seguito del verificarsi del BUG nell'Ambiente Otii.

che durasse anche per più giorni di seguito, senza che vi fosse nemmeno un minimo intervento da parte dell'operatore.

## 6. Risultati

Avendo definito tre Sotto-Fasi di Raccolta Dati, è necessario che vi siano altrettante Sotto-Fasi di Analisi dei Risultati.

### 6.1. Analisi dei Risultati Preliminari

L'analisi di questi risultati precede temporalmente la Sotto-Fase di Raccolta Dati Principale. Questo perché la versione di curl che dovrà essere utilizzata in tali Esperimenti è funzione dei risultati che si analizzeranno nel seguito. Dunque, scopo principale di questa analisi è quello di valutare se esistano differenze in termini energetici tra le diverse versioni di curl e in tal caso di individuare la versione più efficiente.

Le configurazioni testate nella relativa Sotto-Fase di Raccolta Dati sono 4. Eseguendo 30 Esperimenti per ciascuna configurazione si arriva ad un totale di 120. Aggregando i dati registrati sotto forma di grafici si ottiene quanto segue.

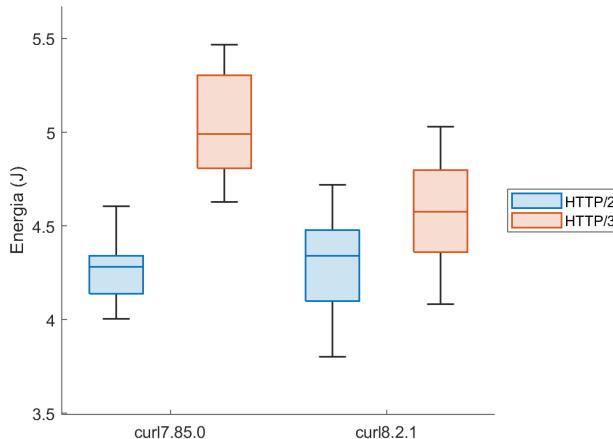


Figura 12: Consumo Energetico di diverse Versioni di curl.

In particolare, in Figura 12 viene riportato un box-plot dove sull'asse delle ordinate compare il consumo energetico espresso in Joule (J), mentre sull'asse delle ascisse si riportano le due versioni di curl sotto analisi. Per ciascuna versione viene riportato poi il consumo energetico di HTTP/2 (in blu) e di HTTP/3 (in arancione). Grazie a questo grafico è semplice osservare che il consumo di HTTP/2 non è particolarmente diverso nelle due versioni di curl. D'altra parte HTTP/3 ha un consumo inferiore quando eseguito su curl8.2.1. Le ragioni dietro queste osservazioni possono essere spiegate con quanto segue. Ricordando che HTTP/2 fa uso del TCP come protocollo di livello trasporto e considerando lo stato abbastanza inoltrato del TCP segue che tra versioni relativamente vicine di curl non vi saranno particolari differenze legate alla specifica implementazione. HTTP/3, invece, è talmente recente che viene definito ancora

come *sperimentale* da curl [3]. Anche QUIC, su cui si basa HTTP/3, non può essere paragonato a TCP se si considerano i tempi di vita di ciascuno. Per questo HTTP/3 risente così tanto della particolare implementazione ed è chiaro che è soggetto ancora ad ulteriori miglioramenti.

Oltre ai Dati sul consumo Energetico, durante la Fase di Raccolta Dati vengono registrate anche:

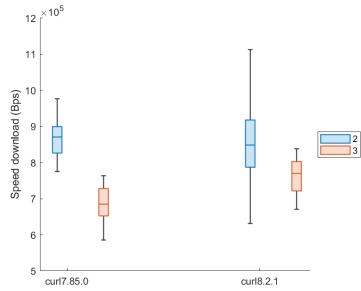
- statistiche sul tempo di utilizzo della CPU;
- informazioni generate da `curl` sulle richieste HTTP effettuate.

Per raccogliere le statistiche sull'utilizzo della CPU, [1] ha previsto la lettura del contenuto del file `/proc/stat` del Raspberry Pi con una frequenza di campionamento di 10 Hz (i.e. 10 letture al secondo). In particolare, per ogni Esperienza, si registra il tempo speso dalla CPU nello spazio *utente*, nello spazio *sistema* e nello stato di *idle*. In riferimento alle informazioni generate da `curl` si considerano:

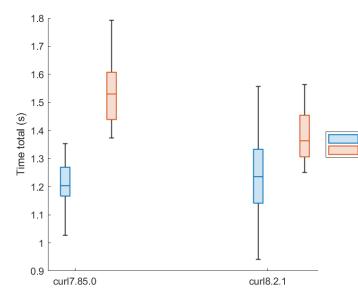
- Velocità di Download.
- Tempo Totale del trasferimento.
- Tempo di *Pretransfer*: intervallo di tempo che intercorre tra l'istante di inizio operazioni e l'istante prima che la richiesta HTTP venga inviata.
- Tempo di *Start Transfer*: intervallo di tempo che intercorre tra l'istante di inizio operazioni e l'arrivo del primo byte della risposta. Questo intervallo di tempo viene anche indicato come TTFB (Time To First Byte).

Considerando quest'ultime informazioni si disegnano i grafici di Figura 13. Per HTTP/2 non si osservano particolari differenze. Soffermandosi su HTTP/3 si vede da 13a che la velocità di download è leggermente aumentata nella versione 8.2.1 di `curl`. Di conseguenza il tempo totale del trasferimento è diminuito come si può verificare da 13b. Da quest'ultimo grafico si nota anche che il tempo totale di trasferimento di HTTP/3 rimane ancora maggiore di quello di HTTP/2. Questo è un ulteriore motivo che spiega perché il consumo energetico di HTTP/3 è maggiore di quello di HTTP/2 (Figura 12). Considerando 13c e 13d si vede che non vi sono particolari migliorie da commentare. Tra l'altro il tempo di pretransfer delle quattro configurazioni è molto simile. Lo stesso vale per il tempo di start transfer. Ricordando il significato dei due intervalli di tempo, è chiaro che il vantaggio introdotto da HTTP/2 su HTTP/3, relativamente al tempo totale del trasferimento, si avrà per forza nella ricezione dei byte della risposta successivi al primo.

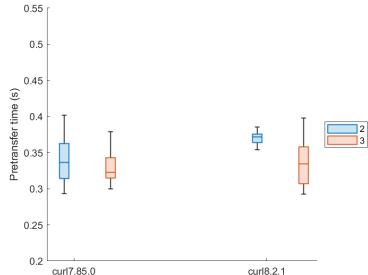
Per confermare le osservazioni fatte sul consumo energetico non resta altro che considerare le statistiche sull'utilizzo della CPU, che vengono rappresentate in Figura 14. Anche in questi grafici la situazione non cambia. Infatti, l'utilizzo della CPU di HTTP/2 nelle due versioni di `curl` è praticamente lo stesso. Se si considera HTTP/3 vi è una riduzione del tempo di utilizzo della CPU sia nello spazio sistema che nello spazio utente con un conseguente aumento del tempo trascorso nello stato di idle. HTTP/3 continua a spendere più tempo di



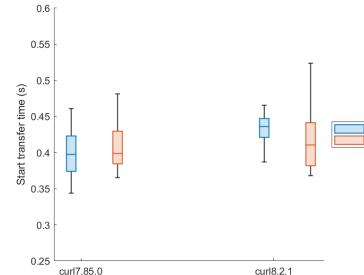
(a) Velocità di Download.



(b) Tempo Totale del trasferimento.



(c) Tempo di Pretransfer.



(d) Tempo di Start Transfer.

Figura 13: Dei grafici basati sulle informazioni generate da curl per l'analisi dei risultati preliminari.

HTTP/2 nello spazio utente perché QUIC è implementato nello spazio utente a differenza di TCP (come già osservato).

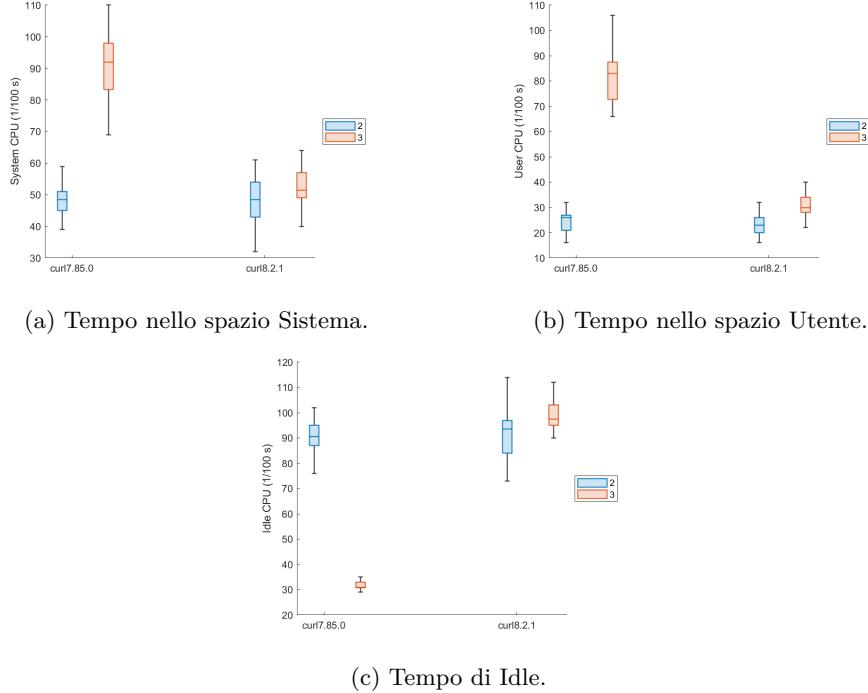


Figura 14: Statistiche sull'utilizzo della CPU per le versioni di curl considerate.

Se al posto di aggregare i dati relativi al consumo energetico in un box plot si preferisse visualizzarli in un formato tabellare si otterebbe la seguente la Tabella.

Versione HTTP	curl7.85.0	curl8.2.1
2	$4.283 \pm 0.204$	$4.342 \pm 0.272 (+1.4\%)$
3	$4.992 \pm 0.266 (+9.1\%)$	$4.577 \pm 0.270$

In questa si riporta per l'appunto il consumo energetico espresso in Joule (J) delle diverse versioni di curl per le versioni 2 e 3 di HTTP. I valori riportati rispettano il formato: mediana  $\pm$  deviazione standard. La percentuale riportata tra parentesi esprime il consumo energetico addizionale rispetto alla versione più efficiente di curl. Da qui si vede che si arriva addirittura ad un risparmio del 9.1% per HTTP/3 su curl8.2.1. Quanto detto in precedenza è dunque confermato e rafforzato dalla percentuale riportata.

È chiaro dunque che HTTP/3 è stato migliorato nella nuova versione di curl,

ma è anche chiaro che c'è ancora della strada da compiere per arrivare alle prestazioni ed ai consumi di HTTP/2. Il giorno in cui preferire *sempre* HTTP/3 rispetto ad HTTP/2 è perciò ancora lontano e non resta che analizzare in quali situazioni conviene utilizzare l'uno oppure l'altro. Si cercherà dunque di procedere in questa direzione durante la prossima Sotto-Fase di Analisi dei Risultati Principali. È scontato dire, arrivati a questo punto, che la versione di `curl` che verrà usata nella Sotto-Fase di Raccolta Dati Principale sarà la 8.2.1.

## 6.2. Analisi dei Risultati Principali

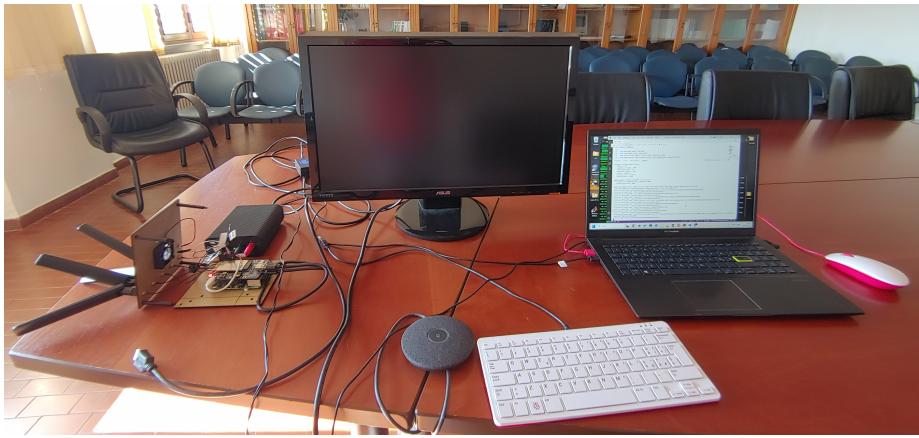


Figura 15: Postazione dove sono stati eseguiti gli Esperimenti della Sotto-Fase di Raccolta Dati Principale.

La Figura appena riportata rappresenta la postazione in cui sono stati eseguiti gli Esperimenti della Sotto-Fase di Raccolta Dati Principale. Come spiegato, questa era una delle poche aree coperte dalla rete 5G. Facendo attenzione si nota anche che il monitor e la tastiera solitamente connessi al Raspberry Pi sono stati scollegati durante l'esecuzione degli Esperimenti per le ragioni discusse in **3.1..** In Figura 16 è riportato a tale scopo un ingrandimento sul Raspberry Pi.

Le configurazioni testate nella relativa Sotto-Fase di Raccolta Dati in questo caso sono 180. Effettuando 30 Esperimenti per ciascuna configurazione si arriva ad un totale di 5400. Il tempo impiegato per l'esecuzione di tutti questi Esperimenti non è affatto trascurabile. Il sistema infatti è rimasto in esecuzione per un totale di più di 72 ore.

Dei 2700 Esperimenti che dovevano essere eseguiti adoperando la rete 5G solamente 1971 sono stati eseguiti utilizzando effettivamente tale generazione. I restanti, per motivi riconducibili a diversi fattori (e.g. fluttuazioni del segnale, particolari condizioni ambientali, carico della rete), hanno fatto uso del 4G. È chiaro che i dati relativi a questi Esperimenti non devono essere considerati per non alterare il processo di analisi. Per identificare tali situazioni si è fatto uso dell'output generato da alcuni comandi AT eseguiti subito dopo la ricezione della risposta HTTP. Questo output viene salvato nel file `AT_AFTER.log` e fa parte dei risultati che il Raspberry Pi carica alla fine di ogni Esperimento sul Server dei Risultati. In particolare, è il comando `AT+CPSI?` che permette di verificare la generazione delle tecnologie radio a cui il Raspberry è connesso. Infatti, se tra le informazioni visualizzate compare la stringa `NR5G_NSA` vuol dire che si sta facendo uso del 5G NSA. Altrimenti, se viene visualizzata solamente la stringa `LTE`, allora la rete che si sta usando è quella di quarta generazione (4G).

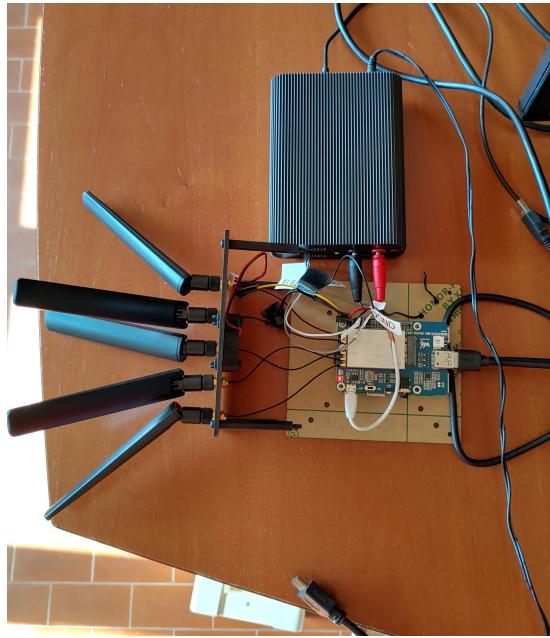


Figura 16: Un ingrandimento sul Raspberry Pi.

Considerando tutti gli Esperimenti 4G e quelli 5G che rimangono dopo l'operazione di filtraggio poc'anzi descritta, si passa alla generazione dei grafici. Il primo è quello presentato in Figura 17a, dove sull'asse delle ordinate viene riportato il consumo energetico espresso in Joule (J) e sull'asse delle ascisse compaiono i ritardi aggiuntivi che sono stati introdotti artificialmente sul Server HTTP. I gruppi rappresentati nel grafico sono del formato **Generazione Radio - Versione HTTP**, per cui il gruppo identificato come 4-3, ad esempio, sta ad indicare la situazione in cui viene inviata una richiesta HTTP/3 sulla rete 4G. I tre grafici presenti nella Figura fanno poi riferimento alle diverse dimensioni del payload di risposta, come specificato dal titolo di ciascuno. Per permettere una migliore lettura dei dati, i grafici di 17a vengono anche scomposti in 17b e 17c in cui viene fatto riferimento solamente ad HTTP/2 oppure ad HTTP/3, rispettivamente. Relativamente agli effetti della Limitazione della Banda, questi per il momento sono stati trascurati e verranno analizzati nei prossimi grafici. Guardando adesso ai grafici di Figura 17 in un'ottica di analisi si riesce ad osservare quanto segue.

- (i) In generale, il consumo del 5G sembra essere maggiore di quello del 4G. Tale differenza è tanto più evidente tanto è minore il payload e maggiore il ritardo. Per payload grandi e ritardi piccoli viceversa, il 5G sembra consumare leggermente meno del 4G. La ragione è molto probabilmente da ricondursi al fatto che la frequenza portante del 5G è maggiore del 4G, per cui il consumo energetico dovrebbe in genere essere maggiore nel

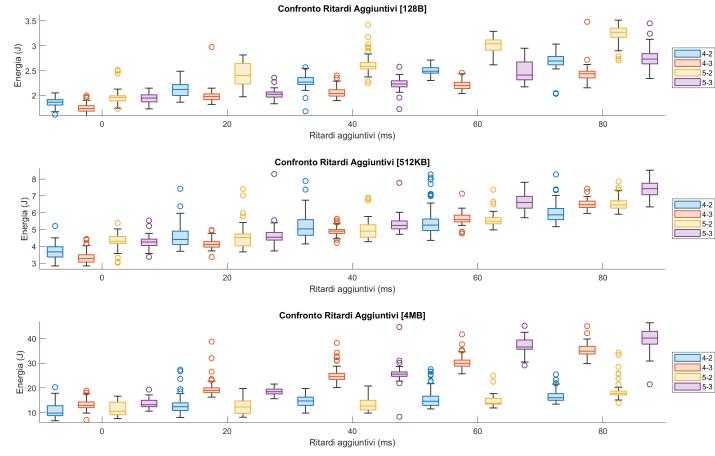
5G. Quello che accade è però che la banda nominale del 5G è maggiore di quella del 4G e per questo, per dimensioni del payload medio-grandi, si ha una maggiore utilizzazione del canale che permette di compensare il maggiore consumo energetico. Questo non accade invece con dimensioni piccole del payload, come si è anche osservato.

- (ii) Prendendo in considerazione adesso le due versioni di HTTP, si nota un comportamento interessante. Per osservare questo particolare si farà riferimento al caso del 4G, notando che nel caso del 5G il comportamento relativo è lo stesso e l'unica cosa che cambia è l'eventuale traslazione in verticale data dalle riflessioni in (i). Quello che si nota è che HTTP/3 per payload di piccole dimensioni (i.e. 128B) consuma meno di HTTP/2. Questo continua a rimanere vero e ad essere anche più evidente per ritardi crescenti. Per payload di medie dimensioni (i.e. 512KB) HTTP/3 rimane sotto HTTP/2 quando il ritardo è piccolo (i.e. minore di 40ms). All'aumentare del ritardo però HTTP/3 peggiora fino a superare HTTP/2. Infine, con payload di dimensioni medio-grandi (i.e. 4MB), HTTP/3 consuma sempre di più di HTTP/2 e per ritardi grandi questa differenza è davvero notevole. Per trovare un senso alle osservazioni fatte basta ricordare i protocolli di livello trasporto su cui si basano HTTP/2 e HTTP/3. Nello specifico, per payload di dimensioni piccole (128B) la leggerezza di QUIC (HTTP/3) permette di completare il trasferimento in minor tempo e consumando dunque meno energia rispetto al TCP (HTTP/2). Le operazioni preliminari di TCP, infatti, introducono dell'overhead che non viene compensato quando il payload è piccolo. In questo caso infatti i 128B della risposta possono essere inviati in un unico pacchetto. Con l'aumento delle dimensioni del payload di risposta, ed in particolare per il payload di 4MB, il consumo derivante dalle operazioni preliminari del TCP viene distribuito su un periodo più lungo e, grazie all'efficienza e alla stabilità del TCP, HTTP/2 consuma meno energia rispetto ad HTTP/3. Prima di poter preferire HTTP/3 ad HTTP/2 per payload di dimensioni medio-grandi è chiaro quindi che QUIC deve essere migliorato.

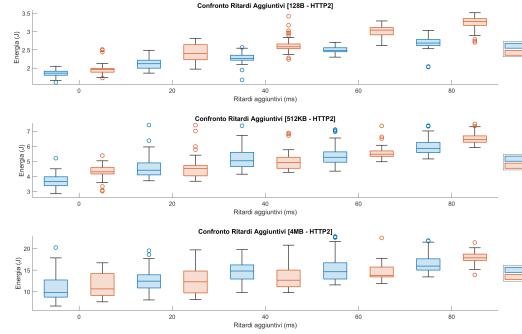
Si procede adesso con l'approfondire (i) e (ii). In quest'ottica vengono tracciati dei grafici dove vengono isolati i fenomeni osservati in modo da permettere un'analisi più attenta.

La Figura 18 fa riferimento a (i). In questa viene riportato il rapporto tra il consumo energetico del 5G e quello del 4G al variare del ritardo. Anche in questo caso sono presenti tre grafici, uno per ciascuna dimensione del payload di risposta. Da questi si nota che in generale il rapporto energetico è maggiore di 1 e dunque, come osservato anche in (i), il 5G consuma in genere più del 4G. Si nota però, dal grafico relativo al payload di 4MB, che esistono anche delle situazioni in cui il consumo del 4G è superiore a quello del 5G: HTTP/2 per ritardo aggiuntivo di 40ms e 60ms.

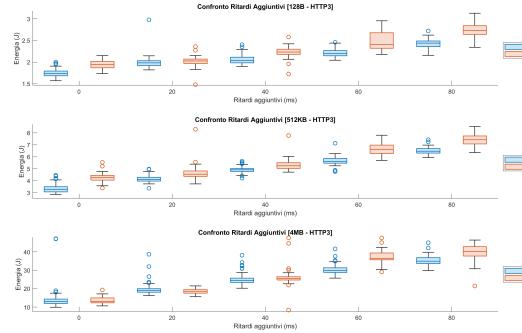
Per indagare su (ii) si traccia il grafico di Figura 19. Qui la condizione di rete presa in considerazione è quella ottimale, con nessuna limitazione di banda e



(a) Consumo Energetico delle coppie Generazione Radio - Versione HTTP.



(b) Consumo Energetico di HTTP/2 per 4G (4) e 5G (5).



(c) Consumo Energetico di HTTP/3 per 4G (4) e 5G (5).

Figura 17: Consumo Energetico a fronte di diversi Ritardi Aggiuntivi per payload di 128B, 512KB e 4MB.

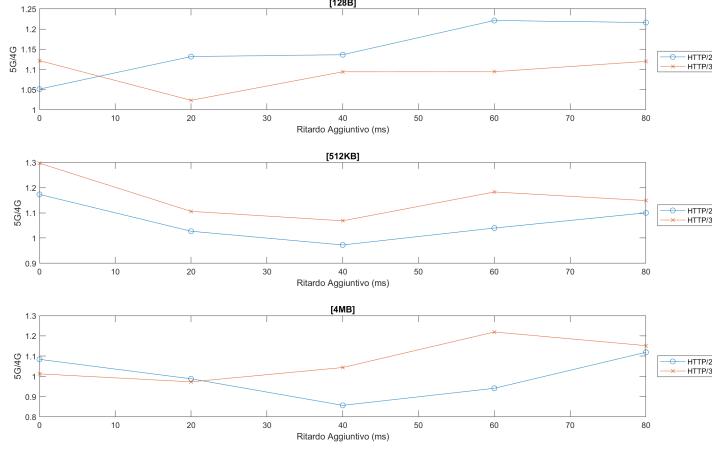


Figura 18: Rapporto tra il consumo energetico del 5G e quello del 4G al variare del ritardo aggiuntivo e per diverse dimensioni del payload di risposta.

nessun ritardo aggiuntivo introdotto dal Server HTTP. A supporto di (ii) si nota che quando il payload di risposta è piccolo HTTP/3 consuma meno di HTTP/2. Questo risparmio è più evidente nel 4G, a differenza di quanto accade nel 5G. La situazione rimane simile anche quando la dimensione arriva a 512KB. Lo stesso non si può dire nel caso di 4MB. Qui HTTP/3 supera nettamente il consumo di HTTP/2. Si nota anche che HTTP/3 su 4G è poco più dispendioso di HTTP/3 su 5G.

Considerando a questo punto la Limitazione di Banda, finora trascurata, si ottengono i grafici di Figura 20. In 20a e in 20b vengono analizzate due situazioni opposte: quella in cui non viene introdotto alcun ritardo aggiuntivo dal Server HTTP (i.e. 0ms) e quella dove si ha il ritardo massimo (i.e. 80ms). In tutti i grafici sull'asse delle ordinate viene riportato ancora il consumo energetico espresso in Joule (J), mentre sull'asse delle ascisse si riporta il valore, in Mbps, a cui viene limitata la banda sul Server HTTP. Il valore -1 sta ad indicare la situazione in cui non viene imposta alcuna limitazione di banda. Scopo dei grafici è quello di valutare se esista o meno una dipendenza del consumo energetico dai limiti alla banda considerati. Relativamente al 4G, osservando 20a e 20b, sembra che i tre limiti di banda considerati non introducano particolari effetti sul consumo energetico. L'unica situazione da osservare è quella riportata in 20a, configurazione [4G - 4MB - 0ms]. In questo caso, limitando la banda a 10 Mbps si osserva un consumo energetico insolito, che si discosta dai valori relativi al limite di 25 Mbps e dalla situazione di nessuna limitazione. Lo stesso non si può dire però per HTTP/3 e nemmeno nel caso di ritardo aggiuntivo di 80ms. Similmente guardando ai rispettivi grafici del 5G, nulla si nota se non

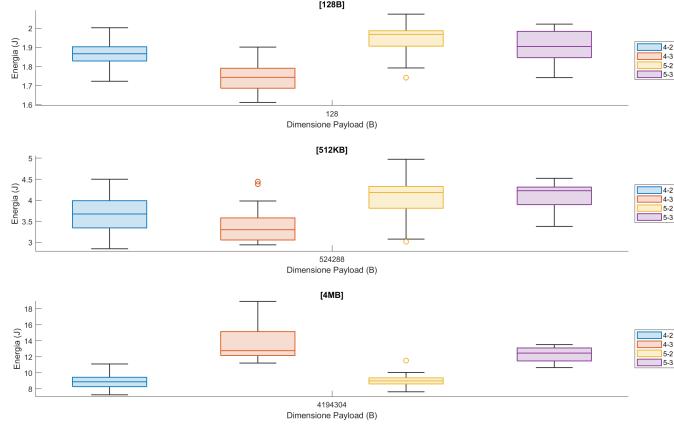
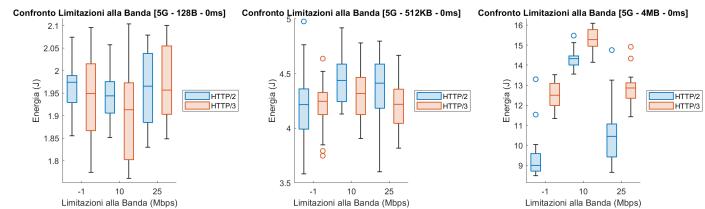
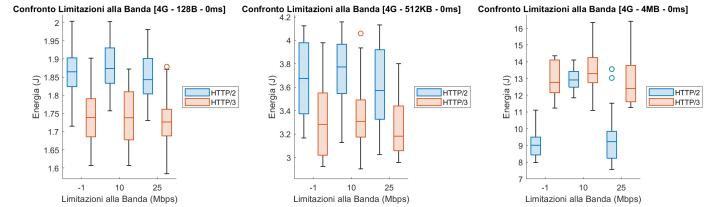


Figura 19: Consumo Energetico delle coppie Generazione Radio - Versione HTTP per payload di risposta di dimensioni pari a 128B, 512KB e 4MB.

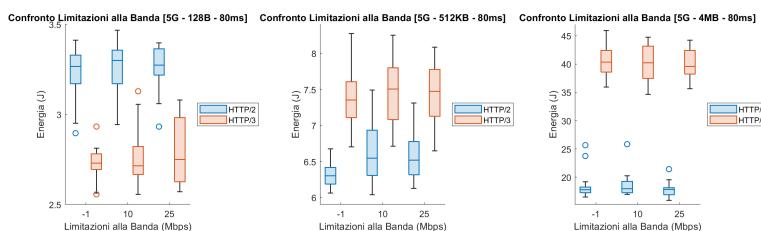
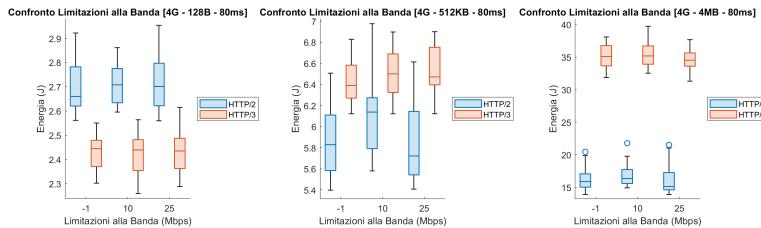
nella stessa configurazione, che questa volta sarà: [5G - 4MB - 0ms]. Come prima HTTP/2 si comporta diversamente solamente quando il limite alla banda è di 10 Mbps. Questa volta però anche HTTP/3 ha un comportamento diverso quando il limite è di 10 Mbps, e consuma di più rispetto agli altri due casi considerati. Oltre a questi piccoli particolari, si può dire che complessivamente, per le dimensioni dei payload considerate e per i limiti di banda imposti, limitare la banda del Server HTTP non ha effetti significativi sul consumo energetico.

Dopo aver valutato l'impatto dei diversi fattori sul consumo energetico si procede con l'effettuare un'analisi ANCOVA. In questa verrano esaminati solamente i fattori che più influenzano il consumo energetico delle richieste HTTP. In particolare, si sceglie come *variabile dipendente* l'energia (espressa in Joule), come *variabile di raggruppamento* la coppia Generazione Radio - Versione HTTP e infine come *covariata* il ritardo aggiuntivo (espresso in millisecondi). Quell'ultimo infatti, come si è visto, influenza l'energia indipendentemente dalla generazione radio / versione HTTP in uso. Per considerare poi le diverse dimensioni dei payload di risposta si è deciso di effettuare tre diverse analisi, una per ciascuna dimensione. Si riporta quanto ottenuto in Figura 21. Analizzando i tre grafici separatamente si riesce ad osservare quanto segue.

- Analisi ANCOVA per payload di piccole dimensioni (i.e. 128B), Figura 21a: dal grafico si osserva che la configurazione più dispendiosa in termini energetici è la 5-2 (i.e. HTTP/2 su 5G). A seguire vi sono, nell'ordine, 4-2 e 5-3 che hanno un comportamento simile. Infine la configurazione che consuma di meno è la 4-3. Esaminando la pendenza dei modelli tramite un test di significatività statistica si scopre che:
  - 5-2 ha una pendenza diversa da tutte le altre configurazioni;



(a) Ritardo Aggiuntivo a 0ms.



(b) Ritardo Aggiuntivo a 80ms.

Figura 20: Consumo Energetico in presenza di Limitazioni della Banda.

- 4-2 e 5-3 hanno una pendenza simile, e diversa dalle rimanenti configurazioni;
- 4-3 ha una pendenza diversa dalle restanti configurazioni.
- Analisi ANCOVA per payload di dimensioni medie (i.e. 512KB), Figura 21b: procedendo come al punto precedente, si osserva che la configurazione con il consumo più elevato è questa volta la 5-3. La configurazione 4-3, invece, ha un comportamento particolare, perché per ritardi piccoli è la configurazione con il consumo più basso. Con l'aumentare del ritardo però supera le configurazioni 5-2 e 4-2, ma rimane sempre al di sotto di 5-3. Ad avere un comportamento simile questa volta sono 5-2 e 4-2. I risultati del test di significatività sulla pendenza dei modelli questa volta sono:
  - 4-2 e 5-2 hanno, come anche osservato, pendenze simili, e diverse dalle altre due configurazioni;
  - anche le pendenze di 4-3 e 5-3 sono simili, ma diverse dalle restanti configurazioni.
- Analisi ANCOVA per payload di dimensioni medio-grandi (i.e. 4MB), Figura 21c: in questo caso si vede bene che 5-3 per ritardi maggiori di 20ms è la configurazione con il consumo più alto. Quando il ritardo aggiuntivo è pari a 0ms, però è 4-3 a consumare più energia. 4-2 e 5-2 sono le configurazioni con il consumo minore e questa volta hanno un comportamento quasi sovrapposto. Il test di significatività statistica, sempre in relazione alla pendenza dei modelli, afferma che:
  - 5-3 ha una pendenza diversa da tutte le altre configurazioni;
  - lo stesso vale per la pendenza di 4-3;
  - 4-2 e 5-2 hanno una pendenza simile, e diversa dalle altre due configurazioni.

L'analisi ANCOVA di Figura 21 è dunque in linea con (i) e (ii), e non fa altro che confermare le osservazioni fatte in precedenza.

Per avere un confronto numerico e per permettere il calcolo delle percentuali di consumo energetico addizionale che sussistono tra le diverse configurazioni si riporta la seguente Tabella. I valori in essa riportati (espressi in Joule) rappresentano le mediane dei consumi energetici misurati per le corrispondenti configurazioni. Le diverse limitazioni alla banda vengono trascurate perché ininfluenti rispetto al consumo energetico complessivo.

Configurazione	0ms	20ms	40ms	60ms	80ms
4G-HTTP/2-128B	1.864	2.124	2.272	2.488	2.691
4G-HTTP/3-128B	1.735	1.980	2.039	2.200	2.438
5G-HTTP/2-128B	1.959	2.404	2.582	3.038	3.272
5G-HTTP/3-128B	1.946	2.026	2.230	2.408	2.730
4G-HTTP/2-512KB	3.664	4.408	5.045	5.273	5.875
4G-HTTP/3-512KB	3.280	4.111	4.904	5.593	6.471
5G-HTTP/2-512KB	4.299	4.527	4.907	5.483	6.463

Configurazione	0ms	20ms	40ms	60ms	80ms
5G-HTTP/3-512KB	4.254	<b>4.546</b>	<b>5.240</b>	<b>6.615</b>	<b>7.432</b>
4G-HTTP/2-4MB	<b>9.821</b>	12.438	14.785	14.641	<b>15.961</b>
4G-HTTP/3-4MB	12.962	<b>19.035</b>	24.468	29.911	34.885
5G-HTTP/2-4MB	10.646	<b>12.287</b>	<b>12.665</b>	<b>13.777</b>	17.860
5G-HTTP/3-4MB	<b>13.121</b>	18.518	<b>25.536</b>	<b>36.466</b>	<b>40.177</b>

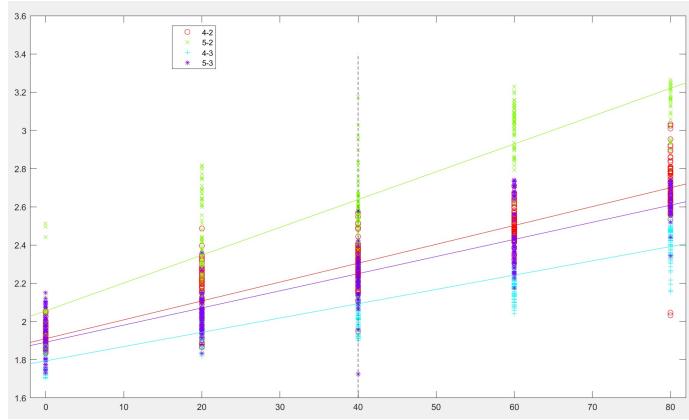
Come si vede le configurazioni sono state raggruppate in funzione della dimensione del payload di risposta. Vi sono per questo tre gruppi distinti associati a 128B, 512KB e 4MB. All'interno di ciascun gruppo e per ogni ritardo aggiuntivo (i.e. per ogni colonna di valori) viene indicato in *verde* il consumo energetico minore, mentre si indica in *rosso* il consumo energetico maggiore. Utilizzando questi valori si calcolano le percentuali di consumo energetico addizionale riportate nella seguente Tabella.

Dimensione	0ms	20ms	40ms	60ms	80ms
128B	+12.9%	+21.4%	+26.6%	+38.1%	+34.2%
512KB	+31.1%	+10.6%	+6.9%	+25.4%	+26.5%
4MB	+33.6%	+54.9%	+101.6%	+164.7%	+151.7%

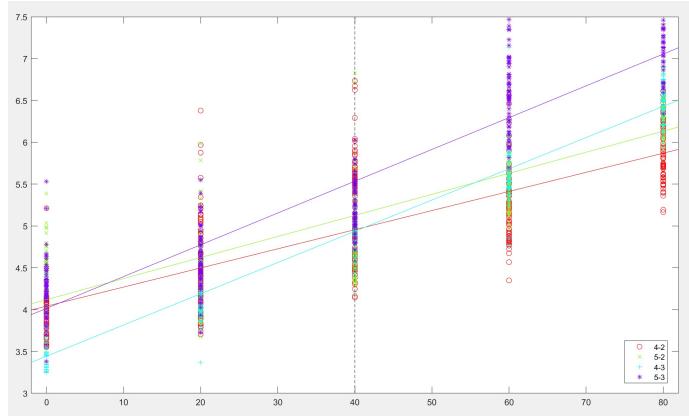
In riferimento al primo gruppo (128B), la configurazione più efficiente è sempre 4G-HTTP/3, mentre quella più dispendiosa è sempre 5G-HTTP/2. Il maggiore consumo addizionale che si ha in questo gruppo è del 38.1%, in corrispondenza di ritardo aggiuntivo pari a 60ms. Nel secondo gruppo (512KB) per 0ms si ha un consumo addizionale del 31.1% tra la configurazione 5G-HTTP/2 che è la più costosa e la 4G-HTTP/3 che è la meno costosa. Infine nel terzo gruppo (4MB), per 20ms, 40ms e 60ms, si nota che è una configurazione che fa uso del 5G ad essere la meno costosa: 5G-HTTP/2. Il consumo addizionale che si ha usando la 5G-HTTP/3 in corrispondenza dei 60ms è addirittura del 164.7%.

È chiaro che nemmeno i dati numerici sono in contrasto con (i) e (ii). Anzi, da (i) e da quanto osservato per il terzo gruppo sorge una curiosità. Cosa accade per dimensioni del payload di risposta ancora più grandi di 4MB? Nel particolare, il 5G riuscirà ad avere un consumo energetico *sempre* minore del 4G? Queste sono alcune delle domande che hanno spinto a svolgere la Sotto-Fase di Raccolta Dati Aggiuntiva e la conseguente Sotto-Fase di analisi.

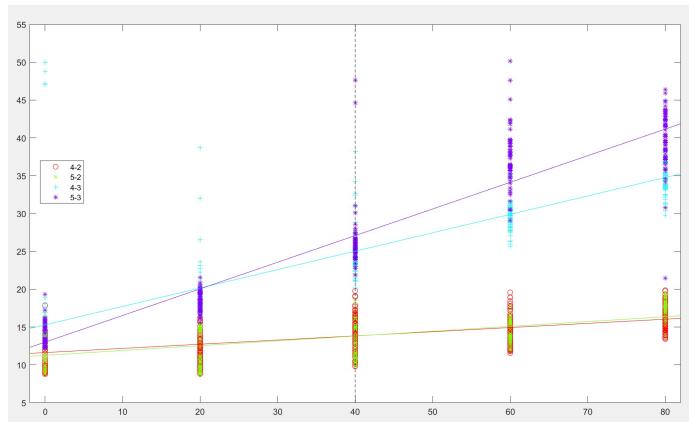
Prima però di rispondere a queste domande con la Sotto-Fase di Analisi Risultati Aggiuntivi si presentano i grafici relativi alle informazioni generate da `curl` (Figura 22) ed anche quelli relativi alle statistiche sull'utilizzo della CPU (Figura



(a) Dimensione del payload di risposta pari a 128B.



(b) Dimensione del payload di risposta pari a 512KB.



(c) Dimensione del payload di risposta pari a 4MB.

Figura 21: Modelli generati dall'analisi ANCOVA per l'analisi dei risultati principali.

23). Da questi non si nota nulla di particolare se non quanto osservato nei corrispondenti grafici della Sotto-Fase di Analisi dei Risultati Preliminari.

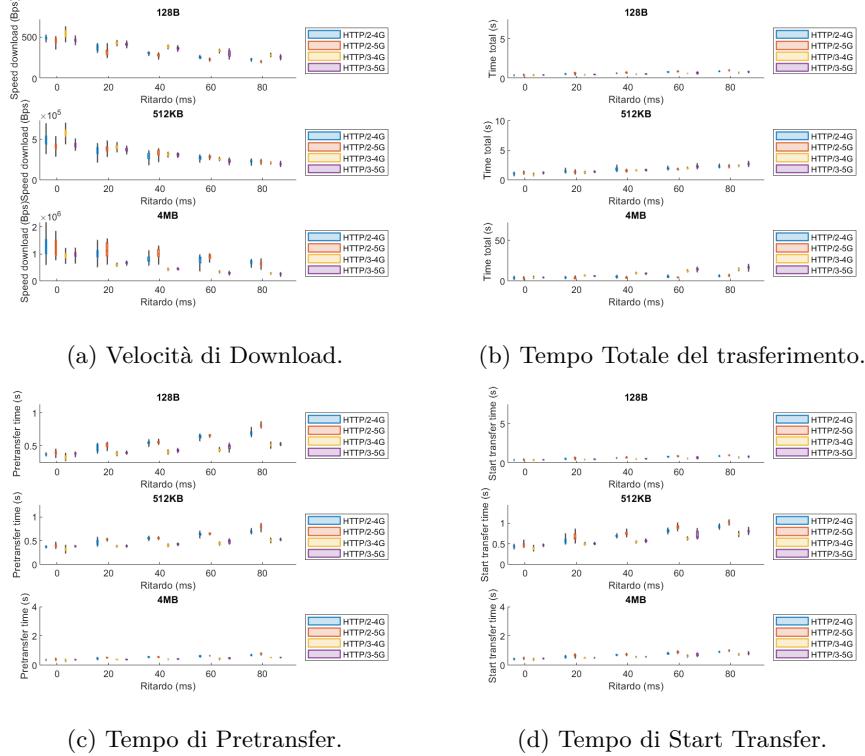


Figura 22: Dei grafici basati sulle informazioni fornite da curl per l'analisi dei risultati principali.

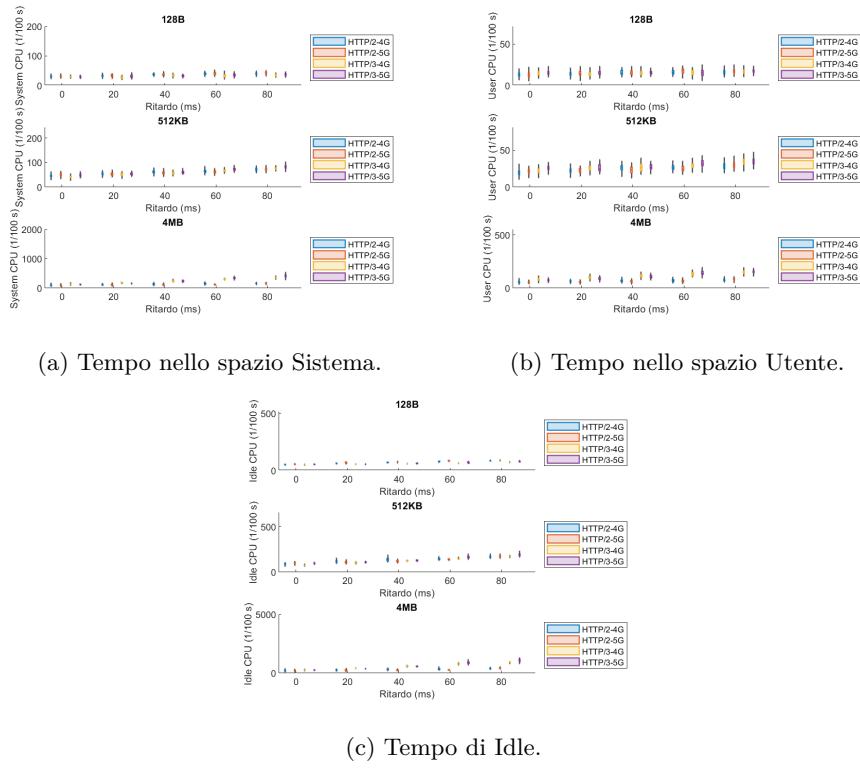


Figura 23: Statistiche sull'utilizzo della CPU per l'analisi dei risultati principali.

### 6.3. Analisi dei Risultati Aggiuntivi

Come già spiegato, questa Sotto-Fase non era inizialmente prevista ma è nata allo scopo di valutare se per payload di risposta di dimensione grande (i.e. 64MB) la maggiore disponibilità di banda del 5G permette di compensare il maggiore consumo energetico che è stato rilevato nei confronti con il 4G (derivante *probabilmente* dall'utilizzo di una frequenza portante maggiore).

Le configurazioni testate nella relativa Sotto-Fase di Raccolta Dati sono 20. Effettuando 30 Esperimenti per le configurazioni in 4G e 33 Esperimenti per le configurazioni in 5G si arriva ad un totale di 630. Il numero di Esperimenti delle configurazioni in 5G è stato aumentato a 33 per cercare di compensare il fatto che la connessione al 5G non è sempre possibile, come spiegato anche nella precedente Sotto-Fase di Analisi. Questa volta la Sotto-Fase di Raccolta Dati è stata più ostica di quella Principale. Anche dopo aver rieseguito gli Esperimenti più volte, nell'arco di 5 giorni, la situazione non sembrava migliorare. Questo serve a testimoniare come una sperimentazione a contatto con l'ambiente reale non sempre dà i risultati previsti nella teoria. Difatti, dei 330 Esperimenti previsti per il 5G, solo 15 sono stati eseguiti utilizzando effettivamente tale generazione radio, ed i restanti sono stati eseguiti sul 4G. I fattori che avevano influenzato la precedente Sotto-Fase di Raccolta Dati questa volta hanno avuto un'entità maggiore e non hanno permesso di raccogliere un numero significativo di dati. Per questo i grafici presentati in questa Sotto-Fase non hanno un alto grado di robustezza se comparati con quelli finora descritti, ma permettono comunque di fornire una stima del comportamento del 5G per payload di 64MB.

Il primo grafico a considerarsi è quello di Figura 24, equivalente di quello in Figura 17a per payload di risposta di 64MB. Come si osserva fin da subito è vero che i dati a disposizione del 5G sono davvero pochi. Questo però non impedisce di osservare il comportamento energetico del 5G. Contrariamente a quanto si credeva però, il consumo del 5G non è minore di quello del 4G, anzi è circa uguale se non anche maggiore.

Lo stesso si può vedere dall'analisi ANCOVA di Figura 25, che continua lo studio di Figura 21 per payload di dimensione pari a 64MB. Come ormai ci si aspetta, le configurazioni con HTTP/3 continuano a consumare più di quelle con HTTP/2. Relativamente al consumo del 5G, qui si vede bene quanto osservato per la Figura 24. Per avere una ulteriore conferma si esegue anche il test di significatività statistica sulla pendenza dei modelli:

- 5-3 e 4-3 presentano delle somiglianze, ma sono diversi dalle restanti configurazioni;
- anche 5-2 e 4-2 hanno una pendenza simile ma sono diversi dalle precedenti configurazioni.

Per indagare ulteriormente si effettua un'altra analisi ANCOVA, dove si sceglie ancora come *variabile dipendente* l'energia (espressa in Joule), come *variabile di raggruppamento* la coppia **Generazione Radio - Versione HTTP**, ma come

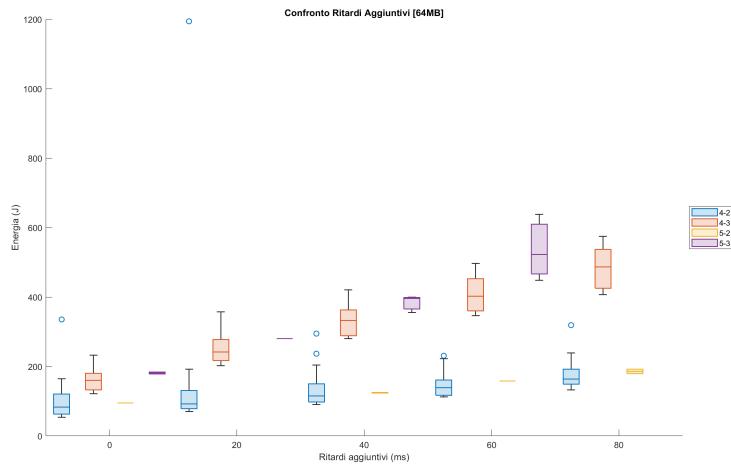


Figura 24: Consumo Energetico a fronte di diversi Ritardi Aggiuntivi per payload di 64MB.

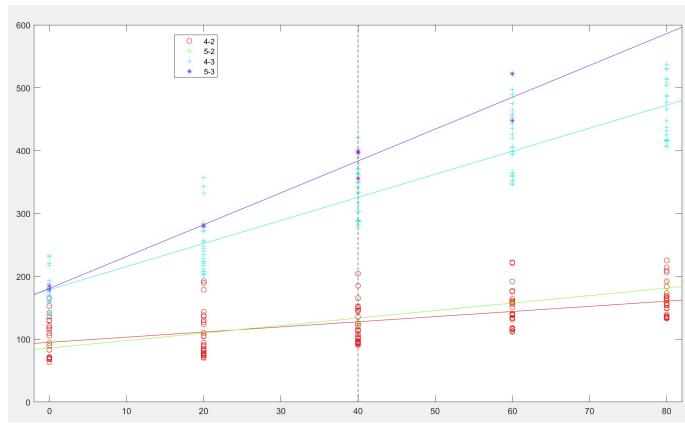


Figura 25: Modelli generati dall'analisi ANCOVA quando la dimensione del payload di risposta è pari a 64MB.

*covariata* questa volta si prende la dimensione dei payload di risposta (espressa in Byte). Relativamente al Ritardo Aggiuntivo, vengono fissati i due casi estremi di ritardo aggiuntivo pari a 0ms e ritardo aggiuntivo pari a 80ms. Per ciascuno dei due casi viene effettuata un'analisi ANCOVA separata (Figura 26). Sia da 26a che da 26b si vede che con l'aumentare della dimensione del payload le configurazioni che fanno uso del 5G consumano più di quelle sul 4G. Dunque sembra che la maggiore banda nominale del 5G non riesca a compensare il maggiore consumo. Il test di significatività statistica, sempre in riferimento alla pendenza, dà i seguenti risultati:

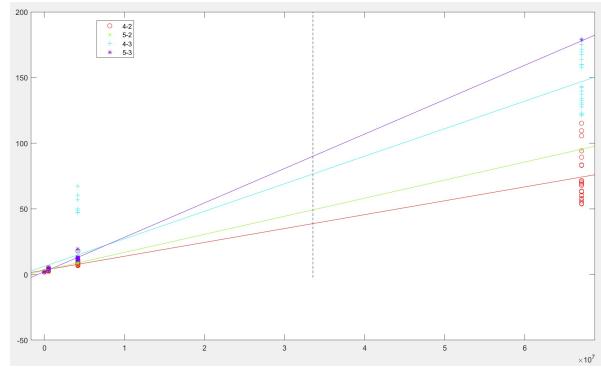
- Per l'analisi ANCOVA relativa al Ritardo Aggiuntivo pari a 0ms, di Figura 26a:
  - 5-3 ha una pendenza diversa da tutte le altre configurazioni;
  - 4-3 ha una pendenza diversa dalle restanti configurazioni;
  - 5-2 e 4-2 hanno una pendenza comparabile, ma diversa dalle rimanenti configurazioni.
- Per l'analisi ANCOVA relativa al Ritardo Aggiuntivo pari a 80ms, di Figura 26b:
  - 5-3 e 4-3 hanno una pendenza comparabile, ma diversa dalle restanti configurazioni;
  - 5-2 e 4-2 hanno una pendenza simile e diversa dalle altre configurazioni.

Da questi risultati si vede che quando il ritardo arriva ad 80ms, anche HTTP/3 su 4G (configurazione 4-3) peggiora a tal punto da essere comparabile con HTTP/3 su 5G (5-3).

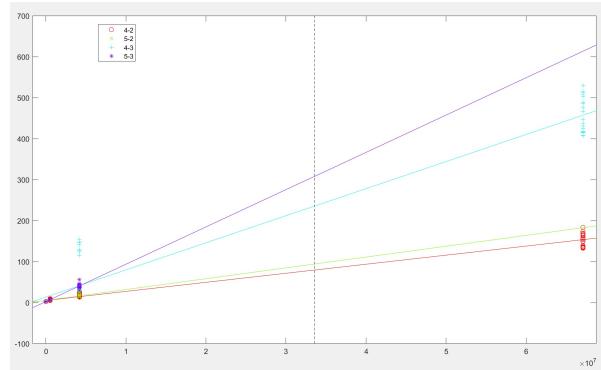
Presentando anche in questo caso i dati in formato tabellare si ottiene quanto segue. Come prima, i valori riportati nella Tabella (espressi in Joule) rappresentano le mediane dei consumi energetici misurati per le configurazioni corrispondenti. Nelle configurazioni in cui non si ha alcun dato a disposizione viene riportato NaN. Per ciascun ritardo aggiuntivo (i.e. per ogni colonna di valori) viene ancora indicato in *verde* il consumo energetico minore. Il consumo energetico maggiore non viene indicato perché dall'analisi ANCOVA di Figura 25 si vede che questo si ha per la configurazione 5-3, ma nella Tabella mancano dei dati per questa configurazione (in particolare quando il ritardo è 80ms).

Configurazione	0ms	20ms	40ms	60ms	80ms
4G-HTTP/2-64MB	83.267	92.597	114.781	139.269	163.706
4G-HTTP/3-64MB	159.697	241.938	332.973	401.934	487.118
5G-HTTP/2-64MB	94.769	NaN	124.709	157.977	185.606
5G-HTTP/3-64MB	181.420	279.950	396.160	522.107	NaN

Guardando ai grafici relativi alle informazioni fornite da curl (Figura 27), ed in particolare al grafico in 27b, si nota un ulteriore dettaglio che potrebbe essere di aiuto per capire le ragioni che stanno dietro al consumo elevato del 5G anche



(a) Ritardo Aggiuntivo pari a 0ms.



(b) Ritardo Aggiuntivo pari a 80ms.

Figura 26: Risultati dell'analisi ANCOVA dove come covariata viene scelta la dimensione del payload di risposta.

in presenza di payload di grosse dimensioni. Si nota che il Tempo Totale del trasferimento del 5G è paragonabile a quello del 4G. Questo è vero in entrambe le versioni di HTTP. Ma se la banda nominale (i.e. *bandwidth*) a disposizione del 5G è maggiore di quella del 4G, ed il tempo impiegato dal trasferimento sui due è lo stesso, allora è chiaro che il *throughput* effettivo dei due è al più uguale alla *bandwidth* del 4G. Vi è dunque una inutilizzazione della *bandwidth* del 5G che è da ricondursi, molto probabilmente, al particolare algoritmo di controllo della congestione di cui fanno uso TCP e QUIC sul Server HTTP. Difatti l'impatto degli algoritmi per il controllo della congestione è maggiore per payload di grosse dimensioni. Si decide quindi di esplorare brevemente questa strada. Per trovare che l'algoritmo per il controllo di congestione di cui fa uso il TCP è CUBIC è bastato eseguire il comando seguente sul Server HTTP.

```
sysctl net.ipv4.tcp_congestion_control
```

In riferimento a QUIC invece, non essendoci alcun comando del genere, si è visto da [29] che anche quest'ultimo, di *default*, fa uso di CUBIC. A questo punto riconsiderando [9] si osserva proprio che CUBIC, pur funzionando bene sul 4G e permettendo una utilizzazione della *bandwidth* del 64.4%, soffre di una estrema inutilizzazione della *bandwidth* nel 5G raggiungendo uno scarso 31.9%. La soluzione ivi proposta prevede di far uso di algoritmi per il controllo della congestione come BBR (*Bottleneck Bandwidth and Round trip-time*) meno sensibili alla perdita / ritardo dei pacchetti (CUBIC invece si basa proprio sulla rilevazione di questi eventi lato mittente). Sempre in [9] si vede infatti che grazie a BBR si riesce ad arrivare ad una utilizzazione della *bandwidth* nel caso del 5G dell' 82.5%. Il prossimo passo dello studio sarebbe quindi quello di includere nella valutazione energetica diversi algoritmi per il controllo della congestione tra cui, naturalmente, BBR.

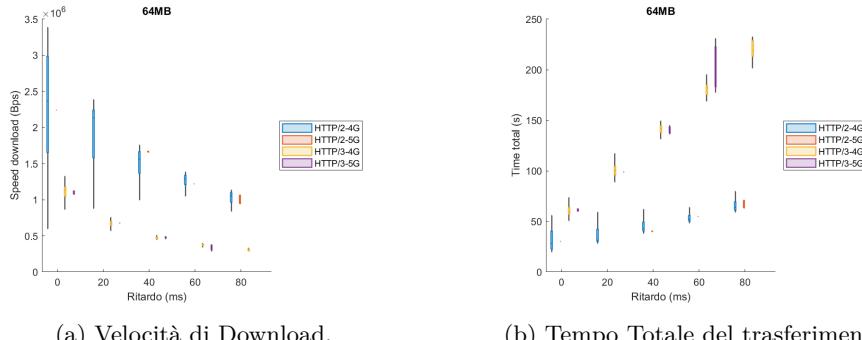


Figura 27: Dei grafici basati sulle informazioni fornite da curl per l'analisi dei risultati aggiuntivi.

Per completezza si riportano infine in Figura 28 i grafici relativi alle statistiche sull'utilizzo della CPU. Per questi, non notandosi nulla di particolare, si possono applicare le osservazioni fatte nella Sotto-Fase di Analisi dei Risultati Prelimi-

nari.

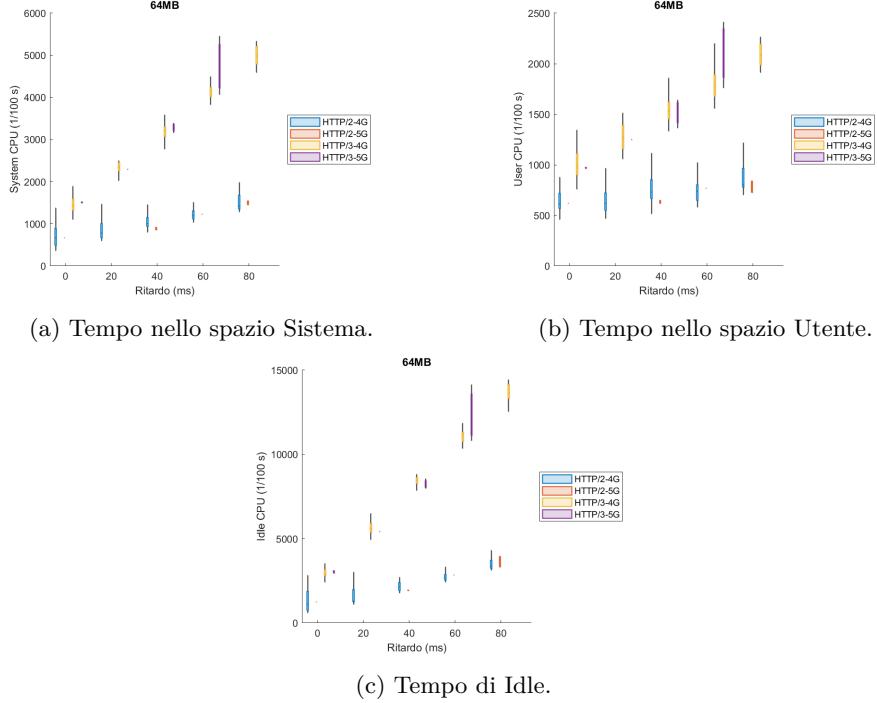


Figura 28: Statistiche sull'utilizzo della CPU per l'analisi dei risultati aggiuntivi.

## 7. Conclusioni

La valutazione oggetto dello studio ha permesso di capire che, nello scenario considerato (pattern di comunicazione del tipo *machine-to-machine*, rete 5G NSA, modifica artificiale delle condizioni di rete), vale quanto segue:

- HTTP/2 non è sempre più efficiente di HTTP/3 e non è vero nemmeno il viceversa. Nelle implementazioni attuali, il loro comportamento dipende fortemente dalla dimensione dei payload che devono trasportare. Difatti per:
  - payload di piccole dimensioni (128B negli Esperimenti): HTTP/3 funziona meglio e consuma meno di HTTP/2;
  - payload di medie dimensioni (512KB): HTTP/3 consuma meno di HTTP/2 solamente per condizioni di rete ottimali. Con il peggioramento del ritardo, ad esempio, la situazione si ribalta ed HTTP/2 ha un consumo minore di HTTP/3.
  - payload di dimensioni medio-grandi (4MB) / grandi (64MB): HTTP/2 è sempre più efficiente di HTTP/3.
- L'uso di una rete 5G (NSA) senza il *tuning* degli elementi coinvolti comporta, in generale, un consumo energetico aggiuntivo rispetto al precedente 4G, senza introdurre particolari miglioramenti in nessun aspetto. Ciò che funziona bene nel 4G non è detto funzioni bene anche nel 5G (come si è visto dall'analisi dei risultati). Bisogna considerare, infatti, che le nuove tecnologie del 5G avranno in generale requisiti diversi e potranno offrire i benefici promessi solo se utilizzate nel modo corretto.

È chiaro dunque che questo studio oltre a *Valutare il Consumo Energetico di HTTP/3 in Ambiente 5G* ha anche aperto la strada ad ulteriori approfondimenti, che potranno essere perseguiti nei successivi studi. Sarebbe interessante, ad esempio, *Valutare il Consumo Energetico* di:

- Architettura Standalone (SA) del 5G;
- Diversi Algoritmi per il controllo della congestione;
- Nuove implementazioni di QUIC, per scoprire se HTTP/3 riuscirà, un giorno, a superare HTTP/2 in ogni condizione.

## 8. Bibliografia

- [1] C. Caiazza, V. Luconi, A. Vecchio - “Energy Consumption of Different Versions of the HTTP Protocol in Edge/Cloud Environments”.
- [2] QUIC - <https://en.wikipedia.org/wiki/QUIC> - Data consultazione: Settembre 2023.
- [3] HTTP/3 - <https://everything.curl.dev/http/http3> - Data consultazione: Settembre 2023.
- [4] Mobile network coverage - <https://www.itu.int/itu-d/reports/statistics/2022/11/24/ff22-mobile-network-coverage/> - Data consultazione: Settembre 2023.
- [5] 5G network coverage outlook - <https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/network-coverage#:~:text=Globally%2C%205G%20mid%2Dband%20population,7%20percent%20to%2090%20percent> - Data consultazione: Settembre 2023.
- [6] 5G - <https://en.wikipedia.org/wiki/5G#:~:text=The%20first%20country%20to%20adopt,South%20Korea%2C%20in%20April%202019> - Data consultazione: Settembre 2023.
- [7] O. Shurdi, L. Ruci, A. Biberaj, G. Mesi (2021) - 5G Energy Efficiency Overview. European Scientific Journal, ESJ, 17(3), 315. - <https://doi.org/10.19044/esj.2021.v17n3p315> - Data consultazione: Agosto 2023.
- [8] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuwei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Z. Morley Mao, Feng Qian, Zhi-Li Zhang (2021) - A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications. In ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–28, 2021, Virtual Event, USA. ACM, New York, NY, USA, 16 pages. - <https://doi.org/10.1145/3452296.3472923> - Data consultazione: Agosto 2023.
- [9] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, Huadong Ma. (2020) - Understanding Operational 5G: A First Measurement Study on Its Coverage, Performance and Energy Consumption. In Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM '20), August 10–14, 2020, Virtual Event, NY, USA. ACM, New York, NY, USA, 16 pages. - <https://doi.org/10.1145/3387514.3405882> - Data consultazione: Agosto 2023.
- [10] Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030 - <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/> - Data consultazione: Settembre 2023.

- [11] Raspberry Pi 3 Model B+ - <https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/> - Data consultazione: Luglio 2023.
- [12] SIM8200EA-M2 5G HAT for Raspberry Pi, 5G/4G/3G Support, Snapdragon X55, Multi Mode Multi Band - <https://www.waveshare.com/sim8200ea-m2-5g-hat.htm> - Data consultazione: Luglio 2023.
- [13] Otii Arc Pro - <https://www.qoitech.com/otii-arc-pro/> - Data consultazione: Luglio 2023.
- [14] What's the minimum Amp to boot a Raspberry - <https://forums.raspberrypi.com/viewtopic.php?t=277944> - Data consultazione: Luglio 2023.
- [15] Wiki SIM8200EA-M2 5G HAT - [https://www.waveshare.com/wiki/SIM8200EA-M2\\_5G\\_HAT#Working\\_with\\_Raspberry\\_Pi](https://www.waveshare.com/wiki/SIM8200EA-M2_5G_HAT#Working_with_Raspberry_Pi) - Data consultazione: Luglio 2023.
- [16] Visual Studio Code - <https://code.visualstudio.com/> - Data consultazione: Settembre 2023.
- [17] Python - <https://www.python.org/downloads/> - Data consultazione: Settembre 2023.
- [18] Curl - <https://curl.se/> - Data consultazione: Luglio 2023.
- [19] Build with OpenSSL, Build with OpenSSL - <https://github.com/curl/curl/blob/master/docs/HTTP3.md#build-with-openssl> - Data consultazione: Luglio 2023.
- [20] Otii TCP API's documentation, Otii requests, otii\_open\_project - <https://www.qoitech.com/help/tcpserver/otii.html#otii-open-project> - Data consultazione: Luglio 2023.
- [21] Qoitech - <https://www.qoitech.com/> - Data consultazione: Agosto 2023.
- [22] Iliad Mobile 3G / 4G / 5G mappa di copertura, Italy - <https://www.nperf.com/it/map/IT/-/143640.Iliad-Mobile/signal/?ll=43.71936536922933&lg=10.390727519989015&zoom=16> - Data consultazione: Agosto 2023.
- [23] Google, Pixel 5 - [https://en.wikipedia.org/wiki/Pixel\\_5](https://en.wikipedia.org/wiki/Pixel_5) - Data consultazione: Settembre 2023.
- [24] SIM8200 Series\_AT Command Manual - [https://simcomturkiye.com/pdf/5G/SIM8200-M2\\_series/SIM8200%20Series\\_AT\\_Command\\_Manual\\_V1.01.pdf](https://simcomturkiye.com/pdf/5G/SIM8200-M2_series/SIM8200%20Series_AT_Command_Manual_V1.01.pdf) - Data consultazione: Agosto 2023.
- [25] Wikipedia, 4G in Italia - [https://it.wikipedia.org/wiki/4G\\_in\\_Italia](https://it.wikipedia.org/wiki/4G_in_Italia) - Data consultazione: Settembre 2023.
- [26] Wikipedia, Bande di frequenze 5G NR - [https://it.wikipedia.org/wiki/Bande\\_di\\_frequenze\\_5G\\_NR](https://it.wikipedia.org/wiki/Bande_di_frequenze_5G_NR) - Data consultazione: Settembre 2023.
- [27] 5G; NR; Radio Resource Control (RRC); Protocol specification (3GPP TS 38.331 version 16.3.1 Release 16) - [https://www.etsi.org/deliver/etsi\\_ts/13830](https://www.etsi.org/deliver/etsi_ts/13830)

0\_138399/138331/16.03.01\_60/ts\_138331v160301p.pdf - Data consultazione: Settembre 2023.

[28] Linux manual page, tc-netem(8) - <https://man7.org/linux/man-pages/man8/tc-netem.8.html> - Data consultazione: Agosto 2023.

[29] Zhang Z, Li S, Ge Y, Xiong G, Zhang Y, Xiong K. - PBQ-Enhanced QUIC: QUIC with Deep Reinforcement Learning Congestion Control Mechanism. Entropy (Basel). 2023 Feb 4;25(2):294. doi: 10.3390/e25020294. PMID: 36832660; PMCID: PMC9955954. - <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9955954/#:~:text=QUIC%20uses%20Cubic%20as%20the,receipt%2C%20packe,t%20loss%2C%20etc> - Data consultazione: Settembre 2023.

## **Ringraziamenti**

Al Primo Relatore, il Professore A. Vecchio, che mi ha concesso l'opportunità di continuare e condurre uno studio così rilevante, supportandomi e guidandomi fin dal primo giorno,

Alla mia Famiglia, mio Padre, mia Madre, mio Fratello e mia Sorella, ai quali devo tutti i traguardi raggiunti,

Ai miei Parenti, che sanno e che ci sono sempre stati,

Ai miei Amici, che mi sopportano,

A Tutti, che mi hanno permesso di essere quello che sono,

Rivolgo un immenso e sincero

Grazie.