# Analisi delle prestazioni della Rete Wi-Fi del Campus X di Roma Tor Vergata

## Introduzione

Il presente documento si riferisce allo studio effettuato per lo sviluppo del progetto del corso Performance Modelling of Computer Systems and Networks (9 CFU), tenuto dalla prof.ssa Vittoria De Nitto Personè al 1° Anno del Corso di Laurea Magistrale in Ingegneria Informatica. Il progetto è stato sviluppato da un team composto da 3 persone: Gabriele La Delfa, Gabriele Tummolo e Domenico Verde. L’idea alla base della scelta di questo progetto era quella di generare un qualcosa di utile per il mondo reale, proponendo ed analizzando anche diverse soluzioni migliorative per il problema analizzato.

## Traccia del Progetto

****

Blocco 1

Edificio 4D

Access Points (APs)

Il Campus X di Roma Tor Vergata è il più grande centro di residenze universitarie in Italia; esso sorge in un’area adiacente alla maggior parte delle Macroaree dell’Università ed è attualmente sede del CLA. Al suo interno si svolgono regolarmente diversi eventi riguardanti la comunità universitaria, tra cui anche le lezioni di alcuni corsi di laurea. Esso comprende 5 Blocchi di Edifici (nominati rispettivamente Blocco 1, …, Blocco 5) ciascuno dei quali comprende a sua volta 3 o 4 palazzi (nominati con le lettere A,B,C,D – ad esempio, ci si riferisce ad un palazzo con la sigla 4D). Ogni palazzo è costituito da 3 o 4 piani (a cui vi si aggiunge il piano terra). L’accesso ad internet per i residenti nella struttura è garantito attraverso una rete wireless che comprende 4 Access Point (AP) per piano, posizionati agli angoli dei piani di ogni edificio. Ciascun utente quindi si connette all’ AP più vicino utilizzando un dispositivo Wi-Fi e può navigare sul web (non vi sono limiti sul numero di dispositivi che ogni utente può connettere alla rete). Ciascun Access Point è collegato mediante un cavo Ethernet (RJ45) ad uno Switch a 48 porte posto al piano terra di ogni edificio. Lo switch provvede a fondere i flussi provenienti dai diversi piani del palazzo e ad inoltrare correttamente il traffico in entrata ed in uscita nella giusta direzione.

## Obiettivi dello Studio

Il Primo Step consiste nel definire attentamente gli obiettivi dello studio. Tenuto conto di quanto concordato durante il corso e considerato quanto riportato nei file messi a disposizione dal docente, contenenti diversi esempi di progetti assegnati negli anni passati e delle linee guida di tutto ciò che il progetto deve contenere, si è deciso di definire come obiettivi dello studio i seguenti:

1. Simulare le prestazioni della rete in un edificio tramite un approccio di tipo Next-Event Simulation ed effettuare un’Analisi del Transiente e dello Stato Stazionario;

2. Valutare le prestazioni della rete:

2.1. Studiare come varia il tempo medio di risposta E(Ts) di un utente al variare dell’intensità del traffico in ingresso;

2.2. Identificare eventuali Bottlenecks;

3. Proporre delle possibili soluzioni alternative finalizzate al miglioramento delle prestazioni della rete.

## Un Modello Concettuale

Una volta definiti gli obiettivi dello studio, è necessario procedere costruendo dapprima un modello concettuale. Da una attenta analisi della traccia, il modello concettuale a reti di code proposto è il seguente:



Chiaramente, ciascun edificio è composto da 5 piani sostanzialmente identici fra loro, pertanto è possibile semplificare il problema e studiare un solo piano dell’edificio ed estendere, per simmetria, i risultati anche agli altri piani. Come evidenziato nella traccia, ciascun utente invia pacchetti all’ AP più vicino, il quale provvede ad inoltrare il pacchetto verso lo switch, che è unico in ogni edificio. Ciascun AP e lo switch, quindi, possono avere 2 stati: Libero (Idle – se non sta trasmettendo alcun pacchetto) ed Occupato (Busy – se un pacchetto è in fase di trasmissione). Infine, supponiamo per semplicità che ciascun server abbia infiniti posti in coda.

## Un Modello di Specifica

In questa fase, invece, provvederemo a convertire il modello concettuale creato in precedenza in modello di specifica, aggiungendo informazioni di natura statistica. Innanzitutto, cercheremo di capire che distribuzioni e quali valori assumere per quanto riguarda i tempi di servizio e di inter-arrivo. Non avendo misure del traffico in ingresso alla rete, supporremo che esso abbia distribuzione esponenziale e pertanto assumeremo che il processo di arrivo dei pacchetti nel sistema sia un processo di Poisson. Pensiamo che tale ipotesi rappresenti un giusto compromesso tra semplicità ed accuratezza del modello. Studieremo, inoltre, il comportamento del sistema al variare del traffico totale in ingresso al sistema .

Inoltre, abbiamo ritenuto ragionevole assumere che, data la particolare simmetria del problema, in ciascun piano ci sia la stessa quantità di traffico:

Per quanto riguarda i tempi di servizio, in base a quanto riportato nel link 3 osserviamo come sia ragionevole assumere come dimensione media di un pacchetto (per ragioni di test):

= 512 bytes (Medium Packet Size on Internet)

Inoltre, come visto diverse volte durante il corso e confermato da quanto riportato nel link 4, utilizzare la distribuzione Bounded Pareto può essere un ottimo modo per modellare la distribuzione del tempo di servizio dei serventi nelle reti a pacchetto. In effetti, è noto anche da altri corsi universitari che un pacchetto Ethernet non ha dimensione indefinita, bensì ha una dimensione minima bytes ed una dimensione massima bytes. La distribuzione Bounded Pareto è una distribuzione heavy tailed e presenta 3 parametri: un parametro di shape (utile a modellare la variabilità della distribuzione) a e due parametri che indicano rispettivamente i valori minimo e massimo assumibili. Pertanto, utilizzeremo come distribuzione dei tempi di servizio una distribuzione bounded pareto e studieremo il comportamento del sistema sia nel caso di alta variabilità del tempo di servizio (), sia nel caso di bassa variabilità ().

Per quanto riguarda i valori delle capacità dei componenti della rete, sono stati ricavati direttamente dai datasheet ufficiali (riportati nella cartella Datasheets):

* APs UniFi UAP-AC-PRO [Max data-rate 1300 Mbps @ 2.4 GHz e 5 GHz using 3x3 MIMO]:
* Switch Cisco MS250-48-HW [Switching Capacity 176 Gbps]

Inoltre, al fine di ottenere una corretta verifica e validazione del modello creato, si è deciso di creare anche un’ulteriore versione di esso avente sia tempi di servizio che i tempi di inter-arrivo con distribuzione esponenziale. In tal caso l’intero sistema diventa così una rete di Jackson e pertanto è analizzabile anche teoricamente come una rete di code separabile. In quest’ultimo caso i valori calcolati risultano:

* APs UniFi UAP-AC-PRO [Max data-rate 1300 Mbps @ 2.4 GHz e 5 GHz using 3x3 MIMO]:
* Switch Cisco MS250-48-HW [Switching Capacity 176 Gbps]

Infine, abbiamo effettuato un ulteriore raffinamento del modello concettuale, ottenendo il seguente:

Chiaramente, se ogni edificio possiede 5 piani ciascun piano (per simmetria) ha 1/5 del traffico totale. L’analisi svolta verrà fatta per piano. Tuttavia è necessario tenere conto di tale traffico nello switch.

## Il Modello Computazionale

Si è scelto di utilizzare come linguaggio di programmazione il linguaggio C, per avere maggiore efficienza e per continuità con quanto visto durante il corso. Si è deciso di dotare il programma di diverse strutture quali clock, sum e statistiche di output. File nsssn.c. Descrizioni delle funzioni.

## V&V – Verifica e Validazione

Al fine di creare un modello che emuli realmente e con estrema accuratezza il sistema in oggetto, seguendo quanto detto durante il corso ed il libro di testo, si è deciso di seguire opportunamente le fasi di verifica e validazione del modello. La fase di verifica (are we building the product right?), si è svolta sin dall’inizio dello sviluppo del progetto utilizzando un compilatore appropriato (verifica formale - sintassi). Inoltre, è stata svolta un’attenta attività di debugging al fine di individuare errori nel codice. Infine, sono stati progettati diversi test per le funzioni create per verificare la correttezza delle funzioni implementate. Essendo la verifica un’attività di software engineering, è stato effettuato un testing estensivo delle funzioni.

Per quanto riguarda l’attività validazione, invece, si è agito dapprima considerando il modello esponenziale. Essendo tale modello una rete di Jackson, è possibile calcolare anche i valori teorici del modello tramite la teoria delle reti di code e la formula di KP. Per questo motivo si

Mp/s (Millions of Packets per Second)

Mp/s (Millions of Packets per Second)

## Analisi Transiente

Per effettuare tale analisi è stato utilizzato il meccanismo delle repliche, in particolare sono stati eseguiti 100 e 1000 run.

Il flusso di arrivo (Lambda) preso in esame è uguale a 10, mentre per quanto riguarda la variabilità del sistema sono stati considerati due valori di ALPHA (0,5 e 1,5) per analizzare il comportamento del sistema sia con un alta variabilità del tempo di servizio, che con una bassa variabilità del tempo di servizio.

È stato simulato il comportamento del sistema rispettivamente per:

* 1024 Jobs ≈ 2 minuti
* 2048 Jobs ≈ 4 minuti
* 4096 Jobs ≈ 7 minuti
* 8192 Jobs ≈ 14 minuti
* 16384 Jobs ≈ 28 minuti
* 32768 Jobs ≈ 55 minuti
* 65536 Jobs ≈ 2 ore
* 131072 Jobs ≈ 4 ore

Per effettuare tale analisi è stato eseguito lo stesso codice utilizzato per il simulatore, dove però nella condizione di arresto è stato omesso il controllo dei jobs rimanenti in coda (funzione empty\_queues). Tale funzione prende nome di *transient()* ed ha lo scopo di restituire il tempo medio di attesa degli utenti. Nella funzione principale viene iterata la funzione *transient()* in base al numero di repliche che si vuole considerare, ottenendo così la totalità delle repliche detta ensemble.

L’indipendenza tra le repliche è garantita considerando lo stato finale *del generatore* di ogni replica come stato iniziale della replica successiva, quindi per ogni replica vengono azzerate le statistiche, ma non viene cambiato lo stato del generatore.

Dopodiché i risultati sono stati salvati in file testuali dove vengono riportati i tempi medi di attesa degli utenti per ogni replica. Infinite tramite il programma estimate.c è stato calcolato l’intervallo di confidenza per ogni file testuale, in modo da poter riportare i risultati sia in forma grafica che in forma numerica.

Si può notare come l’intervallo di confidenza diminuisca all’aumentare del numero di repliche, questo perché si ha una maggiore consistenza nei risultati aumentando il numero di repliche (*pag 370 libro di testo – questo testo verrà eliminato a fine revisione della relazione*)

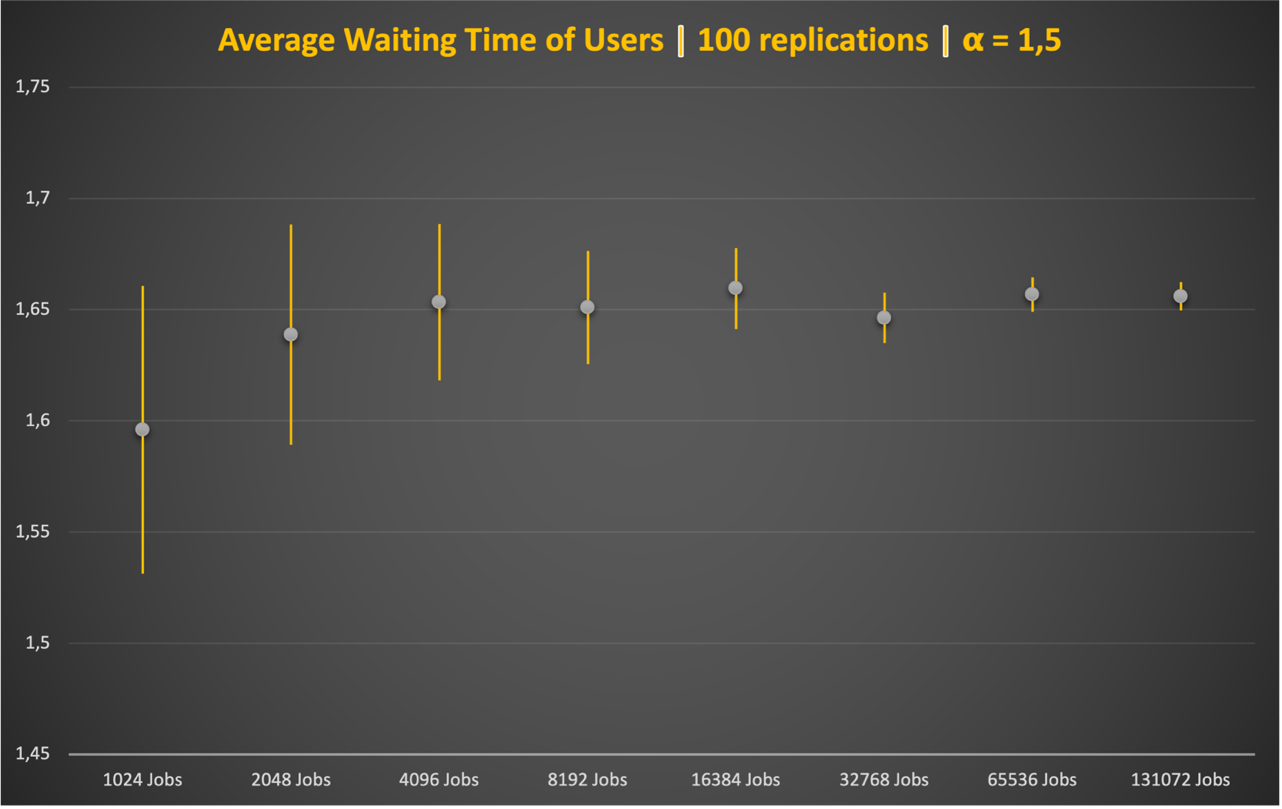
Nei grafici sottostanti vengono riportati nell’asse verticale il tempo medio di attesa degli utenti, mentre nell’asse orizzontale si hanno le etichette facenti riferimento ai jobs presi in considerazione per ogni replica.

## 100 Repliche

Nel primo grafico si ha un valore di Alfa = 0,5, mentre nel secondo grafico si ha un valore di Alfa = 1,5

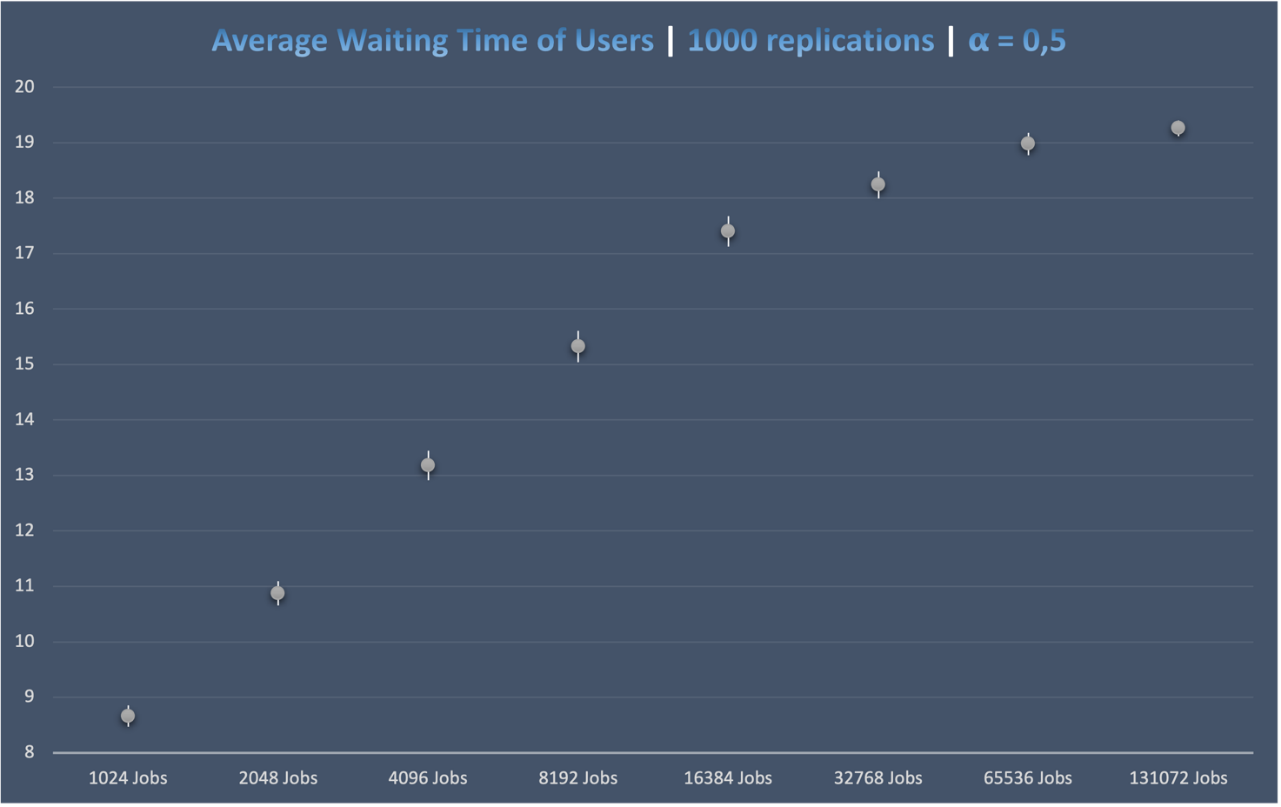
Immagine che contiene testo, elettronico, screenshot, schermo

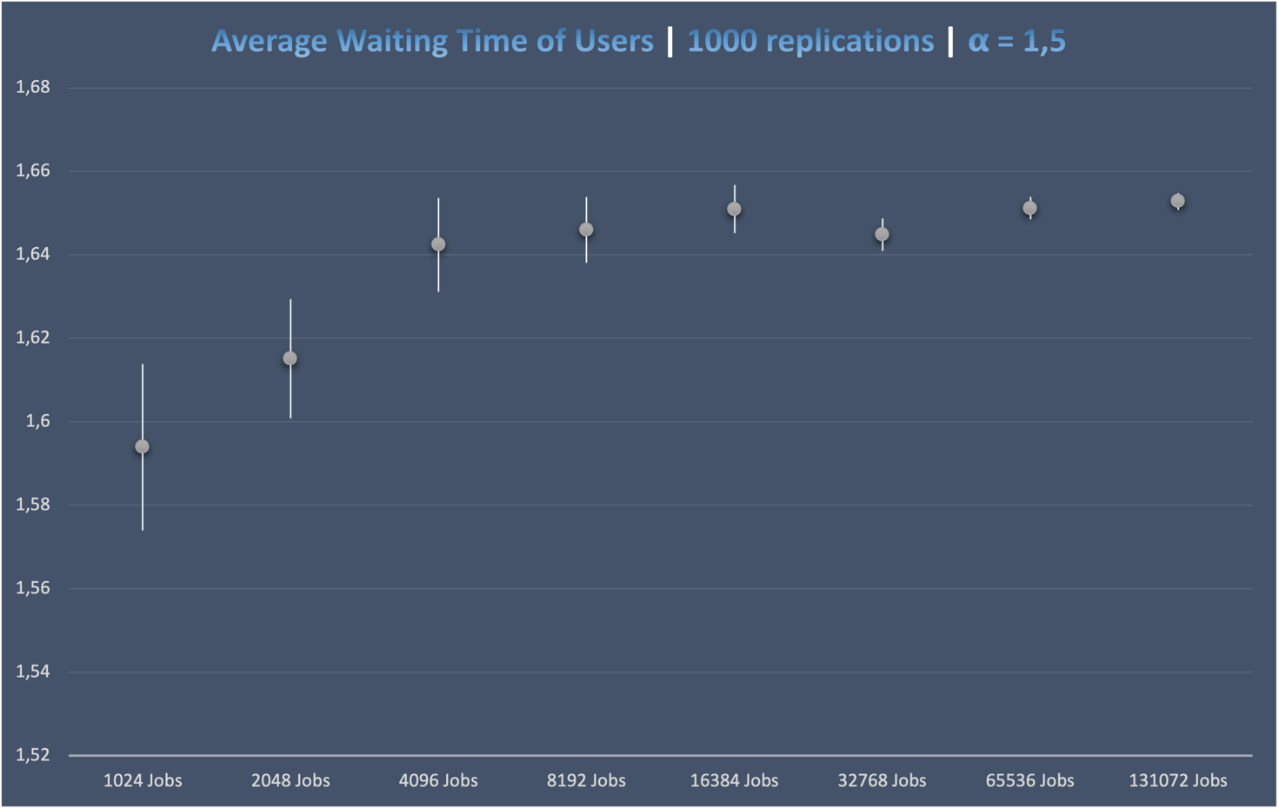
Descrizione generata automaticamente



## 1000 Repliche

Nel primo grafico si ha un valore di Alfa = 0,5, mentre nel secondo grafico si ha un valore di Alfa = 1,5





Notare come la variabilità influenza di brutto il tempo di risposta.

## Analisi Stazionaria

Qui va la parte dell’analisi dello stato stazionario (Batch Means)

## Identificazione dei Bottlenecks

L’identificazione dei bottlenecks (o colli di bottiglia) avviene osservando se all’aumentare del traffico in ingresso al sistema λ esistono alcuni dei devices nella rete la cui utilizzazione tende ad 1 (ovvero al 100%). In tal caso, il valore del traffico per cui questo fenomeno avviene è detto “di saturazione” e si indica con , ed il devices si dice che “è un collo di bottiglia” nel sistema. Ovviamente, se tutti i devices della rete raggiungono un’utilizzazione pari a 1 contemporaneamente, allora la rete è perfettamente bilanciata e non esistono colli di bottiglia nel sistema. Nel nostro caso, aumentando l’intensità del traffico in ingresso fino al valore (misurato in pacchetti al microsecondo), osserviamo che tutti gli Access Points raggiungono un’utilizzazione che è circa del 100%:

Quindi, si può concludere dicendo che gli APs costituiscono dei colli di bottiglia e pertanto, terremo conto di questo risultato nella progettazione dell’algoritmo migliorativo per il sistema. Inoltre, è bene osservare che la variabilità nella distribuzione dei tempi di servizio influenza fortemente il valore del traffico di saturazione. Tale variabilità è modellata attraverso il parametro α della distribuzione BoundedPareto: valori di indicano un’alta variabilità, mentre valori di indicano una bassa variabilità. Infatti, eseguendo diverse volte il simulatore con un valore di α = 1.5 (bassa variabilità):

Osserviamo facilmente che, in caso di bassa variabilità della distribuzione dei tempi di servizio ed a parità di traffico in ingresso, l’utilizzazione dei devices si riduce fortemente. In altre parole, meno è variabile la distribuzione dei tempi di servizio e più performante è il comportamento della rete (ovvero maggiore è il traffico che la rete riesce a sopportare). Tuttavia, tale variabilità non influenza lo studio fatto per l’identificazione dei bottlenecks, che continuano a rimanere gli APs in ogni caso. I valori mostrati nei grafici precedenti sono riportati rispettivamente nei files bottleneck-high.txt e bottlenecks-low.txt e, per completezza, vengono riportati in forma tabellare di seguito.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lambda** | **AP1** | **AP2** | **AP3** | **AP4** | **Switch** |
| 06 | 0.539459 | 0.545725 | 0.549087 | 0.535034 | 0.078725 |
| 07 | 0.645568 | 0.640247 | 0.640872 | 0.624962 | 0.091742 |
| 08 | 0.718426 | 0.722649 | 0.740033 | 0.713626 | 0.104981 |
| 09 | 0.794464 | 0.830568 | 0.811054 | 0.800644 | 0.117331 |
| 10 | 0.896684 | 0.894283 | 0.928219 | 0.890877 | 0.129918 |
| 11 | 0.995716 | 0.982219 | 0.974416 | 0.977301 | 0.141422 |
| 12 | 0.999290 | 0.994507 | 0.974642 | 0.974117 | 0.143312 |
| 13 | 0.991379 | 0.999841 | 0.996060 | 0.989148 | 0.143483 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Lambda** | **AP1** | **AP2** | **AP3** | **AP4** | **Switch** |
| 13 | 0.581249 | 0.598913 | 0.582157 | 0.577501 | 0.084874 |
| 14 | 0.643155 | 0.630188 | 0.632899 | 0.636000 | 0.090829 |
| 15 | 0.669480 | 0.668723 | 0.685700 | 0.675952 | 0.096928 |
| 16 | 0.710267 | 0.733011 | 0.727268 | 0.713491 | 0.104383 |
| 17 | 0.769204 | 0.772666 | 0.771969 | 0.746125 | 0.110765 |
| 18 | 0.808799 | 0.814836 | 0.813391 | 0.816793 | 0.116930 |
| 19 | 0.853394 | 0.849017 | 0.853363 | 0.852466 | 0.123373 |
| 20 | 0.889390 | 0.892075 | 0.911906 | 0.888710 | 0.129954 |
| 21 | 0.944546 | 0.943441 | 0.945751 | 0.946383 | 0.136795 |
| 22 | 0.982787 | 0.976771 | 0.978850 | 0.991418 | 0.142026 |
| 23 | 0.987263 | 0.992586 | 0.997780 | 0.999659 | 0.144607 |

## Alcune Possibili Soluzioni

### Sostituzione degli AP

Considerato lo studio effettuato nel paragrafo precedente, ne è risultato che per eliminare i colli di bottiglia è necessario effettuare delle operazioni di Load Balancing, ovvero di bilanciamento del carico dovuto al traffico in ingresso. Una prima operazione, molto semplice da attuare, consiste nel sostituire gli AP con altri di capacità maggiore: in tal modo ne risulterebbe sicuramente un incremento delle prestazioni; tuttavia, l’acquisto di nuovi apparecchi ha un costo (di diverse decine di euro cadauno) e sostituire tutti gli AP di ogni piano di ogni edificio potrebbe arrivare a costare anche migliaia di euro (considerati anche i costi per la manodopera).

### Incrementare il numero degli AP

Un altro modo per effettuare load balancing potrebbe essere quello di incrementare il numero di access points presenti in ogni piano: in questo modo, il traffico viene ripartito in più AP e pertanto si osserverà una diminuzione dell’utilizzazione di ogni dispositivo e quindi un incremento delle prestazioni; tuttavia, anche in questo caso vale il discorso fatto in precedenza, ovvero: l’acquisto di nuovi apparecchi ha un costo e potrebbe risultare anche elevato. Per evitare ciò, si potrebbe pensare di fornire ogni abitazione di una presa Ethernet e comunicare a ciascun utente in fase contrattuale che l’acquisto dell’AP è a carico dell’inquilino. In questo modo, si riducono le spese derivanti dagli acquisti degli AP e si massimizza il numero di AP stessi in ogni piano (Upper Bound: 1 AP per abitazione). Tuttavia, anche questa soluzione è sconsigliata poiché l’intera rete andrebbe riprogettata per dotare ogni abitazione di una presa ethernet (quindi ulteriori costi di progetto, materiali e manodopera).

### Meccanismo QoS per il Controllo dei Pacchetti

La casa madre degli Access Points Ubiquity, tramite il controller implementato nel firmware delle proprie apparecchiature, mette a disposizione una funzionalità chiamata “Smart Queue”, attraverso la quale è possibile personalizzare la configurazione dei dispositivi. Grazie a questa tecnologia è quindi anche impostare dei limiti sulla lunghezza della coda in cui vengono memorizzati i pacchetti ancora non trasmessi. In questo modo, scartando alcuni pacchetti, è possibile migliorare le performance in caso di congestione del sistema. Tale soluzione rappresenta una soluzione a costo-zero, nel senso che non è necessario acquistare nuovi dispositivi o modificare la rete esistente ma consiste solamente in una piccola modifica della configurazione degli AP, ed è pertanto quella che consigliamo.

## Analisi dell’Algoritmo Migliorativo Proposto

Per convincere della validità della soluzione proposta, seguiremo gli stessi passi dell’algoritmo utilizzato per modellare la rete al fine di creare un modello che rappresenti fedelmente la rete nel caso di code limitate. . Il sistema così facendo diventa un sistema con perdita modellabile utilizzando una rete di code (identico al precedente ma con code limitate).

ed analizzeremo il suo comportamento e le differenze col modello precedente nel caso di code con 10 posti disponibili (il numero 10 è stato scelto poiché rappresenta il numero medio di job in coda con utilizzazione dell’80% circa).

Per quanto riguarda il modello concettuale e di specifica, in sostanza vale lo stesso discorso fatto in precedenza per valori, distribuzioni e variabili di stato. L’unica differenza, appunto, rimangono la lunghezza delle code degli AP che assumeremo di dimensione 10. Per quanto riguarda il modello computazionale, anch’esso rimane sostanzialmente identico al precedente, con l’aggiunta di una porzione di codice che permette di scartare i jobs se la coda è piena.

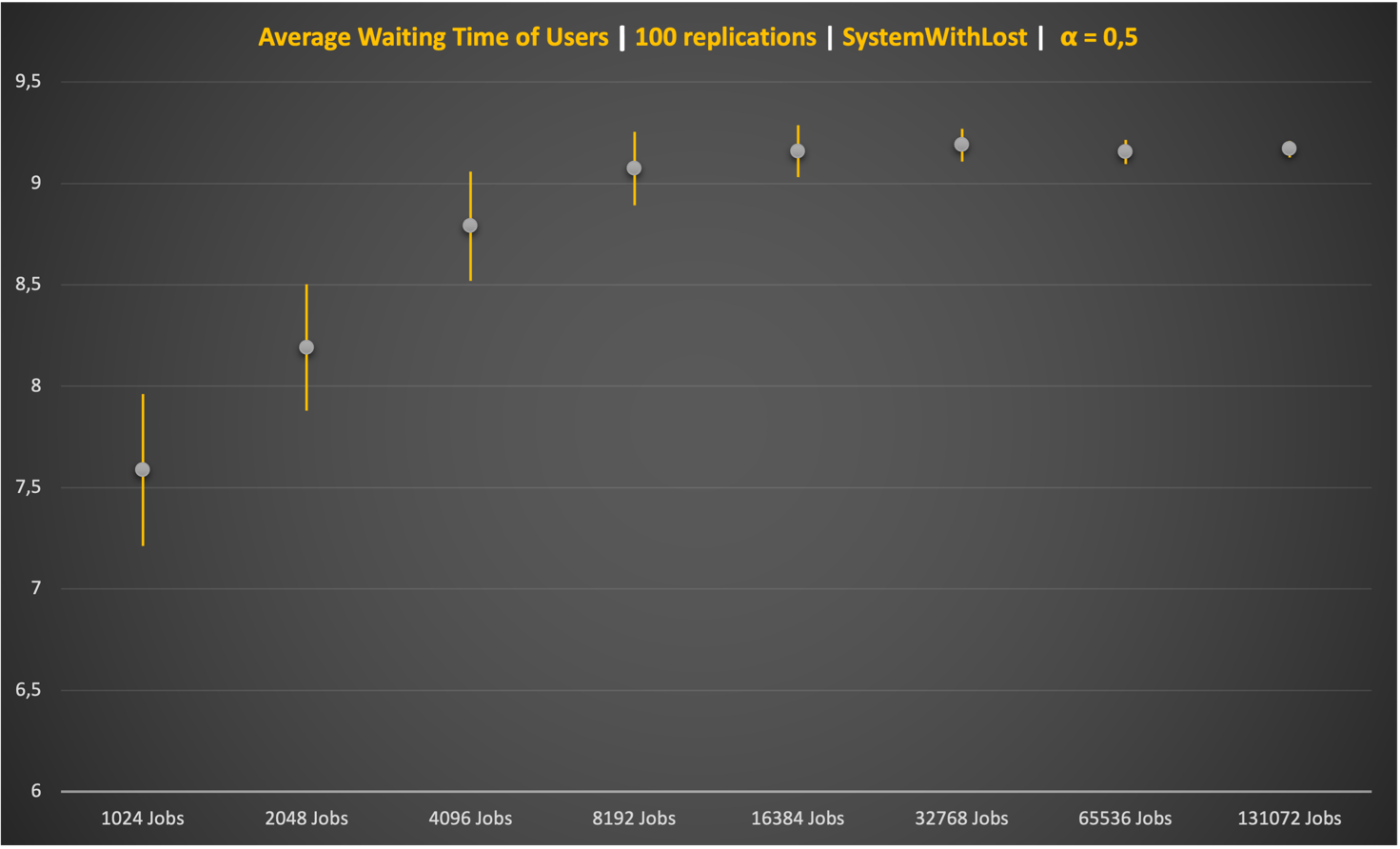
Per quanto riguarda le fasi di verifica e validazione, invece, …

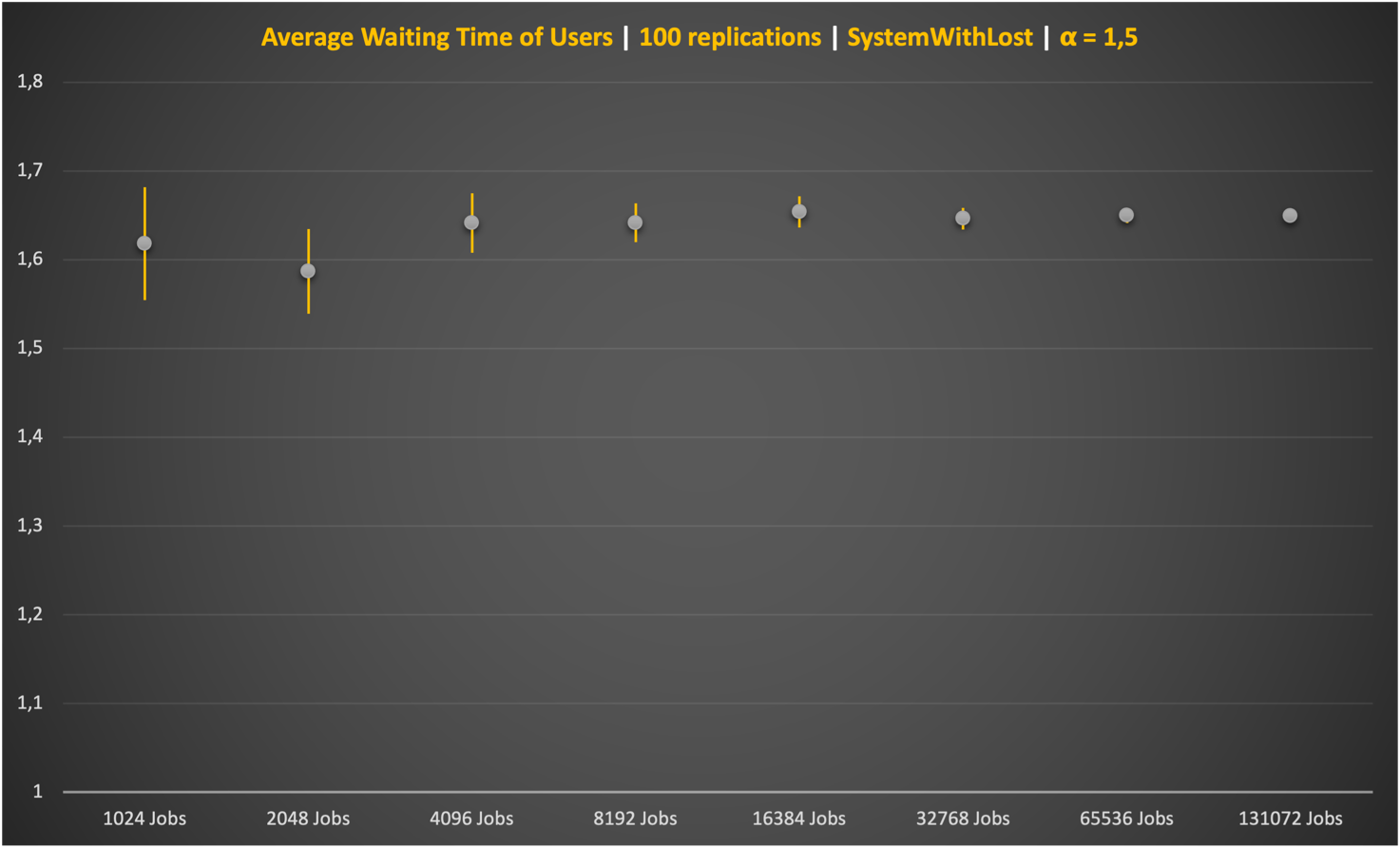
Di seguito, infine riportiamo analisi transiente e stazionaria.

Per quanto riguarda l’algoritmo migliorativo è stato considerato un meccanismo di controllo degli accessi, bloccando l’accesso ai nuovi jobs qualora il numero dei jobs in coda nei singoli AP superi la capacità massima prestabilita, dove nel caso preso in esame è uguale a 10. Si può notare dai grafici sottostanti come questo meccanismo ha garantito un notevole miglioramento delle prestazioni.

Come nel caso precedente vengono riportati nell’asse verticale il tempo medio di attesa degli utenti, mentre nell’asse orizzontale si hanno le etichette facenti riferimento ai jobs presi in considerazione per ogni replica.

## 100 Repliche

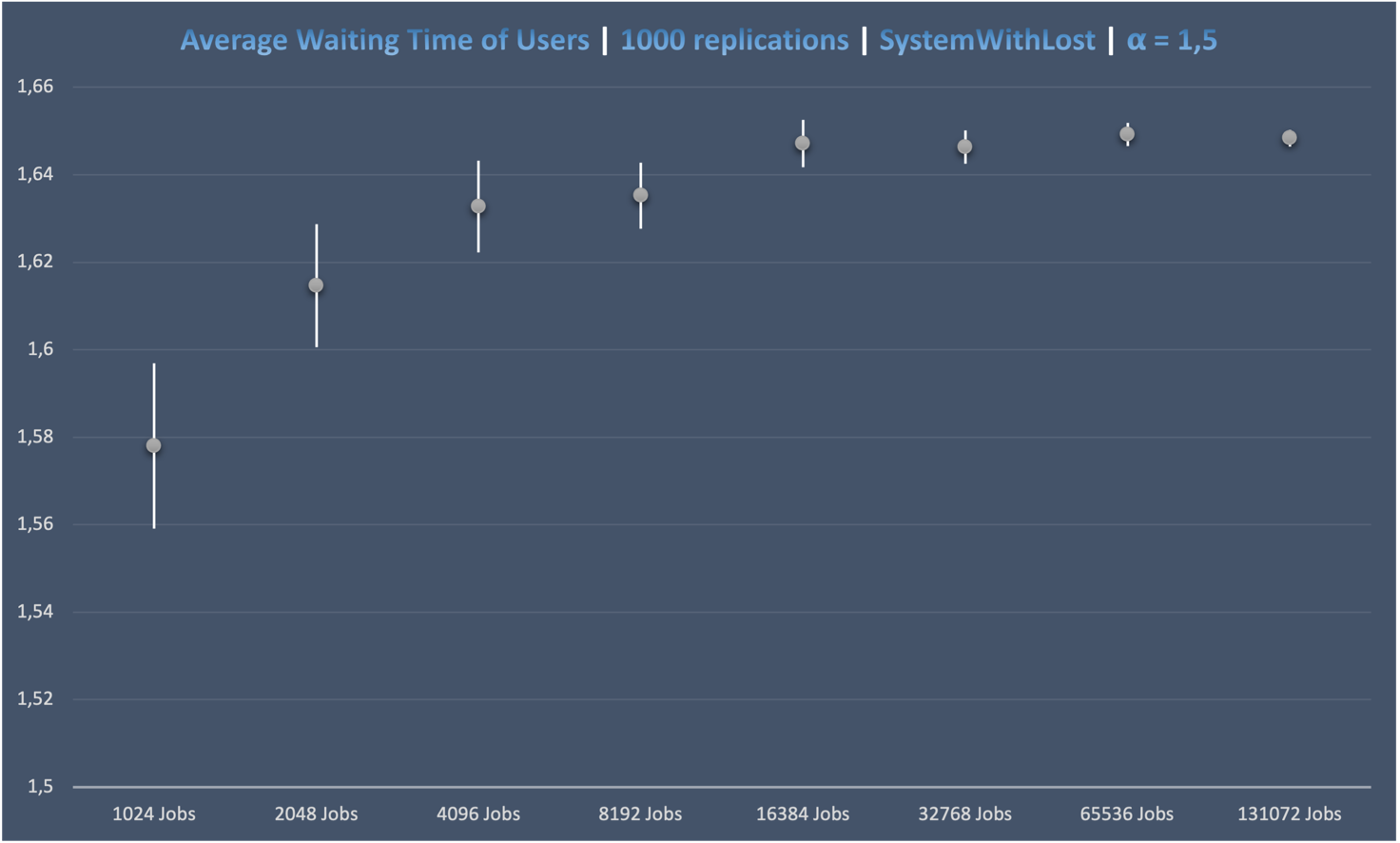
Nel primo grafico si ha un valore di Alfa = 0,5, mentre nel secondo grafico si ha un valore di Alfa = 1,5



## 1000 Repliche

Nel primo grafico si ha un valore di Alfa = 0,5, mentre nel secondo grafico si ha un valore di Alfa = 1,5

Immagine che contiene testo, screenshot, elettronico, computer

Descrizione generata automaticamente

Per quanto riguarda l’analisi dell’output della fase transiente, di seguito vengono riportate le tabelle facenti riferimento ai grafici precedenti:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 REPLICHE - Alfa (0,5) | | | | | | | | |
|  |
|  |
|  | **1024 Jobs** | **2048 Jobs** | **4096** Jobs | **8192 Jobs** | **16384 Jobs** | **32768 Jobs** | **65536 Jobs** | **131072 Jobs** |  |
| MAX | 9,499722 | 11,488959 | 14,183519 | 16,569229 | 18,756561 | 19,251341 | 19,486887 | 19,600163 |  |
| MIN | 8,237987 | 10,283406 | 12,571957 | 14,678723 | 16,757246 | 17,723984 | 18,276679 | 18,773449 |  |
| VALORE | 8,868855 | 10,886183 | 13,377738 | 15,623976 | 17,756903 | 18,487663 | 18,881783 | 19,186806 |  |
| (+/-) | 0,630868 | 0,602777 | 0,805781 | 0,945253 | 0,999657 | 0,763679 | 0,605104 | 0,413357 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 REPLICHE - Alfa (1,5) | | | | | | | | |
|  |
|  |
|  | **1024 Jobs** | **2048 Jobs** | **4096 Jobs** | **8192 Jobs** | **16384 Jobs** | **32768 Jobs** | **65536 Jobs** | **131072 Jobs** |  |
| MAX | 1,660643 | 1,68827 | 1,688574 | 1,676501 | 1,67781 | 1,657733 | 1,664645 | 1,662382 |  |
| MIN | 1,531311 | 1,589293 | 1,618291 | 1,625602 | 1,641219 | 1,634924 | 1,648926 | 1,649653 |  |
| VALORE | 1,595977 | 1,638782 | 1,653432 | 1,651052 | 1,659515 | 1,646329 | 1,656785 | 1,656018 |  |
| (+/-) | 0,064666 | 0,049489 | 0,035141 | 0,02545 | 0,018296 | 0,011405 | 0,007859 | 0,006365 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 REPLICHE - SystemWithLost - Alfa (0,5) | | | | | | | | |
|  |
|  |
|  | **1024 Jobs** | **2048 Jobs** | **4096 Jobs** | **8192 Jobs** | **16384 Jobs** | **32768 Jobs** | **65536 Jobs** | **131072 Jobs** |  |
| MAX | 7,962667 | 8,502488 | 9,059739 | 9,255147 | 9,286185 | 9,270251 | 9,215343 | 9,209633 |  |
| MIN | 7,212711 | 7,879854 | 8,521188 | 8,892077 | 9,030971 | 9,108657 | 9,096068 | 9,129041 |  |
| VALORE | 7,587689 | 8,191171 | 8,790463 | 9,073612 | 9,158578 | 9,189454 | 9,155706 | 9,169337 |  |
| (+/-) | 0,374978 | 0,311317 | 0,269275 | 0,181535 | 0,127607 | 0,080797 | 0,059638 | 0,040296 |  |
| Jobs Refused | 6,04 | 14,75 | 34,37 | 73,66 | 151,72 | 301,46 | 605,29 | 1219,51 |  |
| % Refused | ***0.57%*** | ***0.7025%*** | ***0.8383%*** | ***0.8988%*** | ***0.9261%*** | ***0.9195%*** | ***0.922%*** | ***0.9293%*** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 REPLICHE - SystemWithLost - Alfa (1,5) | | | | | | | | |
|  |
|  |
|  | **1024 Jobs** | **2048 Jobs** | **4096 Jobs** | **8192 Jobs** | **16384 Jobs** | **32768 Jobs** | **65536 Jobs** | **131072 Jobs** |  |
| MAX | 1,681703 | 1,634683 | 1,674599 | 1,663311 | 1,671275 | 1,658509 | 1,658587 | 1,654867 |  |
| MIN | 1,554235 | 1,538835 | 1,607924 | 1,619494 | 1,636187 | 1,633827 | 1,641262 | 1,643324 |  |
| VALORE | 1,617969 | 1,586759 | 1,641261 | 1,641402 | 1,653731 | 1,646168 | 1,649925 | 1,649096 |  |
| (+/-) | 0,063734 | 0,047924 | 0,033337 | 0,021908 | 0,017544 | 0,012341 | 0,008663 | 0,005772 |  |
| Jobs Refused | 0,04 | 0,08 | 0,19 | 0,3 | 0,61 | 1,22 | 2,37 | 4,96 |  |
| % Refused | ***0.0038%*** | ***0.0039%*** | ***0.0045%*** | ***0.0034%*** | ***0.0041%*** | ***0.0034%*** | ***0.0025%*** | ***0.0029%*** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 REPLICHE - Alfa (0,5) | | | | | | | | |
|  |
|  |
|  | **1024 Jobs** | **2048 Jobs** | **4096** Jobs | **8192 Jobs** | **16384 Jobs** | **32768 Jobs** | **65536 Jobs** | **131072 Jobs** |  |
| MAX | 8,85061 | 11,090299 | 13,443291 | 15,60604 | 17,673309 | 18,484241 | 19,182256 | 19,407009 |  |
| MIN | 8,468253 | 10,660149 | 12,915534 | 15,039237 | 17,129982 | 17,997315 | 18,77938 | 19,1159 |  |
| VALORE | 8,659431 | 10,875224 | 13,179412 | 15,322638 | 17,401645 | 18,240778 | 18,980818 | 19,261454 |  |
| (+/-) | 0,191178 | 0,215075 | 0,263878 | 0,283401 | 0,271663 | 0,243463 | 0,201438 | 0,145554 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 REPLICHE - Alfa (1,5) | | | | | | | | |
|  |
|  |
|  | **1024 Jobs** | **2048 Jobs** | **4096** Jobs | **8192 Jobs** | **16384 Jobs** | **32768 Jobs** | **65536 Jobs** | **131072 Jobs** |  |
| MAX | 1,613881 | 1,62941 | 1,653655 | 1,653889 | 1,656686 | 1,648778 | 1,65386 | 1,654749 |  |
| MIN | 1,573964 | 1,600839 | 1,631179 | 1,638058 | 1,645151 | 1,640931 | 1,64849 | 1,650838 |  |
| VALORE | 1,593923 | 1,615124 | 1,642417 | 1,645974 | 1,650919 | 1,644854 | 1,651175 | 1,652794 |  |
| (+/-) | 0,019959 | 0,014285 | 0,011238 | 0,007916 | 0,005768 | 0,003923 | 0,002685 | 0,001956 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 REPLICHE - SystemWithLost - Jobs - Alfa (0,5) | | | | | | | | |
|  |
|  |
|  | **1024 Jobs** | **2048 Jobs** | **4096** Jobs | **8192 Jobs** | **16384 Jobs** | **32768 Jobs** | **65536 Jobs** | **131072 Jobs** |  |
| MAX | 7,757685 | 8,425836 | 8,840278 | 9,077061 | 9,117388 | 9,16854 | 9,177188 | 9,191624 |  |
| MIN | 7,511645 | 8,236784 | 8,68842 | 8,970101 | 9,042759 | 9,11604 | 9,139531 | 9,165748 |  |
| VALORE | 7,634665 | 8,33131 | 8,764349 | 9,023581 | 9,080073 | 9,14229 | 9,158359 | 9,178686 |  |
| (+/-) | 0,12302 | 0,094526 | 0,075929 | 0,05348 | 0,037314 | 0,02625 | 0,018828 | 0,012938 |  |
| Jobs Refused | 5,655 | 15,15 | 33,989 | 72,408 | 147,537 | 300,499 | 604,031 | 1218,46 |  |
| % Refused | ***0,53548%*** | ***0,71921%*** | ***0,82764%*** | ***0,88289%*** | ***0,89955%*** | ***0,91600%*** | ***0,92074%*** | ***0,92849%*** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 REPLICHE - SystemWithLost - Jobs - Alfa (1,5) | | | | | | | | |
|  |
|  |
|  | **1024 Jobs** | **2048 Jobs** | **4096** Jobs | **8192 Jobs** | **16384 Jobs** | **32768 Jobs** | **65536 Jobs** | **131072 Jobs** |  |
| MAX | 1,5969 | 1,628746 | 1,643273 | 1,642786 | 1,652603 | 1,650122 | 1,651811 | 1,650244 |  |
| MIN | 1,559095 | 1,600573 | 1,622304 | 1,62767 | 1,641697 | 1,642559 | 1,646594 | 1,646475 |  |
| VALORE | 1,577997 | 1,614659 | 1,632789 | 1,635228 | 1,64715 | 1,646341 | 1,649203 | 1,648359 |  |
| (+/-) | 0,018902 | 0,014086 | 0,010485 | 0,007558 | 0,005453 | 0,003782 | 0,002609 | 0,001884 |  |
| Jobs Refused | 0,024 | 0,065 | 0,139 | 0,315 | 0,616 | 1,302 | 2,517 | 5,113 |  |
| % Refused | ***0,00225%*** | ***0,00317%*** | ***0,00328%*** | ***0,0036%*** | ***0,00399%*** | ***0,00392%*** | ***0,00306%*** | ***0,00305%*** |  |

Da tale analisi si può notare come la percentuale dei Job respinti si trova al di sotto del 1% in tutti i casi considerati, ma il tempo medio di attesa dell’utente è diminuito notevolmente

Conclusioni e Sviluppi Futuri

ione

Bibliografia e Link Utili

1. Capacity Calculus (MIMO): <https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/Wireless_Throughput_Calculations_and_Limitations>
2. Switching Capacity: <https://forum.huawei.com/enterprise/en/forwarding-performance-and-switching-capacity/thread/570609-861>
3. Packet Best Size: <https://www.networkworld.com/article/2300175/picking-the--best--packet-size.html>
4. Pareto Distribution: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1045.7&rep=rep1&type=pdf>
5. Smart Queue: <https://community.ui.com/questions/Smart-queue-Limit-setting/3930a03e-b7a3-4ac0-85fe-1bc8f042b9c6>