INDICE

[**SPECIFICHE TECNICHE:** 2](#_Toc86082811)

[**Computer**: 2](#_Toc86082812)

[**Sistema Operativo:** 2](#_Toc86082813)

[**Librerie:** 2](#_Toc86082814)

[**ESEMPIO DI FUNZIONAMENTO:** 3](#_Toc86082815)

[**Un esempio di list:** 3](#_Toc86082816)

[**Un esempio di get:** 3](#_Toc86082817)

[**Un esempio di put:** 3](#_Toc86082818)

[**TESTING** 4](#_Toc86082819)

[**Operazione Get:** 4](#_Toc86082820)

[**Ritrasmissioni**: 5](#_Toc86082821)

[**Operazione put:** 7](#_Toc86082822)

[**Variazioni della dimensione della finestra:** 7](#_Toc86082823)

[**Time-out fisso:** 9](#_Toc86082824)

# **SPECIFICHE TECNICHE:**

*Le specifiche tecniche presenti al momento del test. I vari test possono variare su macchine, condizioni e/o metodi differenti dalle seguenti elencate*

## **Computer**:

HP Pavilion Gaming Laptop 15-ec0021nl

-Microprocessore: AMD Ryzen™ 7 3750H with Radeon™ RX Vega 10 Graphics (2.3 GHz base clock, up to 4 GHz max boost clock, 6 MB cache, 4 cores),

-Ram: 8GB DDR4-2400 SDRAM

-Disco rigido: 512GB SSD M.2

## **Sistema Operativo:**

MX Linux 19.3 on virtualbox.

## **Librerie:**

time.h

# **ESEMPIO DI FUNZIONAMENTO:**

*La comunicazione tra client e server è caratterizzata da tre comandi, list, get e put, in seguito viene riportato un breve esempio per ognuno di questi tre casi.*

## **Un esempio di list:**

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamenteNel caso l’utente desidera conoscere quali sono i file presenti sul server, non necessiterà di grandi sforzi per scoprirlo, bisognerà soltanto invocare il comando list “./client list” e stamperà a schermo tutti i file (cartelle e file nascosti **esclusi**)

## **Un esempio di get:**

Immagine che contiene testo, arancia, scuro, vicino

Descrizione generata automaticamenteFacendo la get sul server ovvero “./client get *nomefile*” es: . Il server elaborerà la richiesta, e invierà (se disponibile, altrimenti stamperà un errore) il file verrà inviato con successo.



## **Un esempio di put:**

Immagine che contiene testo, arancia, scuro, vicino

Descrizione generata automaticamenteFacendo la put, similmente ai precedenti, si scriverà “./client put *nomefile*” e in seguito si riceverà, la conferma o possibili errori.

# **TESTING**

*Testing eseguito sui vari comandi in condizioni diverse, come file di diverse dimensioni e/o probabilità di errore differente o dimensioni diverse di finestra e time-out dinamico o fisso.*

**Un po’ di grafici:**

Tutti i risultati usati per la creazione dei grafici, sono stati fatti dopo una serie accurata di test, e presa soltanto la media di questi casi, il numero di casi considerati oscilla tra i 15 e 20. Sono stati accuratamente rimossi rari picchi anomali che avrebbero reso meno chiara, leggibile e comprensibile la visione dei grafici. I grafici delle operazioni trattano **soltanto** le operazioni, non tutto il processo di invio, come un hand shake o ritrasmissioni di ques’ultimo, ma solo la durata dell’operazione, non di tutto il processo. Inoltre, si tengono conto, nel caso delle ritrasmissioni, solo quelle legate a chi deve ricevere il file. Ovvero ritrasmissioni soltanto del server nel caso della put, e del client nel caso della get.

## **Operazione Get:**

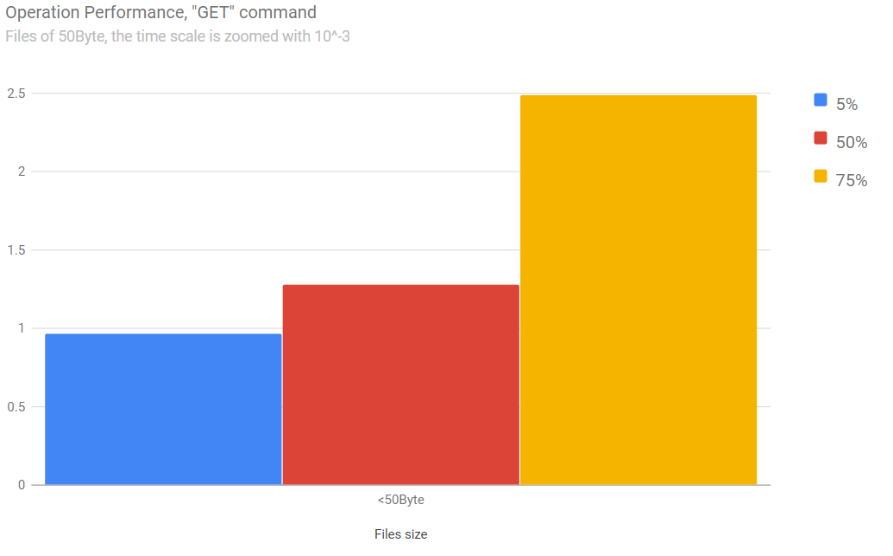
La prima operazione che analizziamo è quella della “get”, l’operazione è chiamata del client come mostrato nell’[esempio della get](#_Un_esempio_di) sopra riportata. Nella Figura 1 possiamo osservare la differenza nei tempi in base alla probabilità di errore fatta, ovvero di 5%, 50% e 75%. Si può notare che diventa molto più marcata per file di dimensioni maggiori, di fatti nei file di piccola-media grandezza, ovvero nell’ordine massimo di pochi KB i tempi non variano in modo troppo esagerato. Per vedere meglio i tempi, si può analizzare la , dove per comodità di lettura c’è stato anche un incremento della scala di riferimento, per un indice di 103. I tempi possono essere simili o comunque molto ravvicinati, diversa la situazione riguardante i file di dimensione considerevoli, come nel caso di 10MB, dove la durata totale dell’invio è vicina al secondo, dovuta proprio a un continuo perdere dati. La perdita di dati comporta ovviamente una ritrasmissione del pacchetto inviato in modo errato. Essendoci una probabilità di perdita fissa (5%, 50% 75%) la perdita di un pacchetto crea si un ritardo come è ovvio, però porta anche una ritrasmissione che nel caso peggiore al 75% fallirà di nuovo, quindi ulteriore ritardo cioè il invio del nuovo pacchetto e tempo per ricevere l’ack.

Figura 1, mostra il comportamento temporale dei pacchetti, in base alla probabilità (5%,50% e 75%) di perdere un pacchetto ed in base alla dimensione del file (50Byte, 1MB, 10MB)

Figura 2, un ingrandimento sul comportamento temporale, ingrandito di 103, dei file di 50Byte.

### **Ritrasmissioni**:

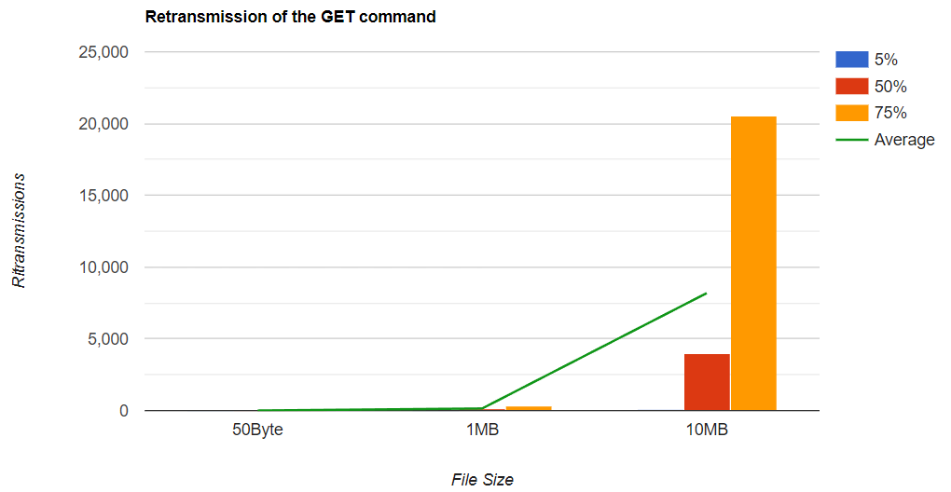
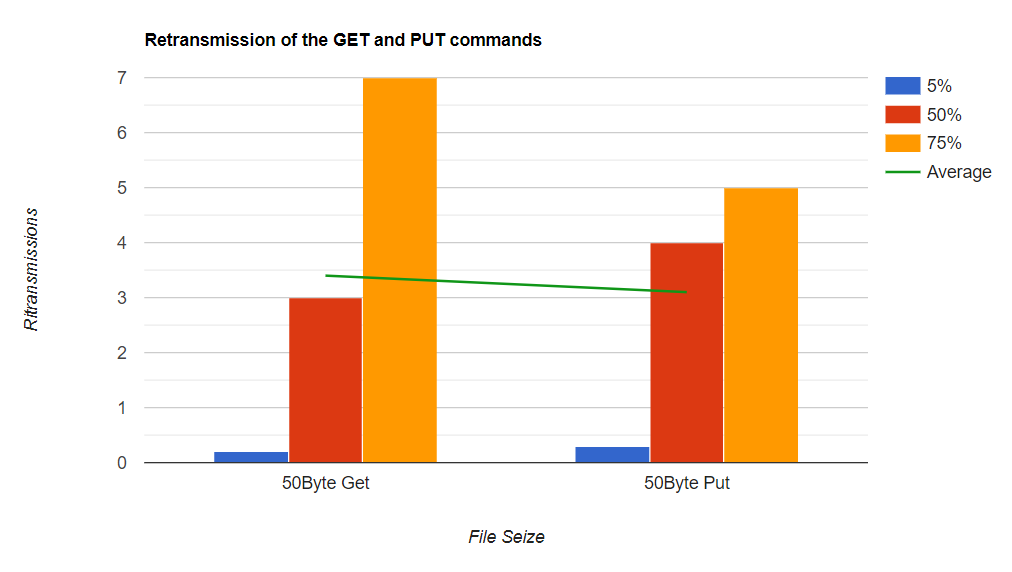
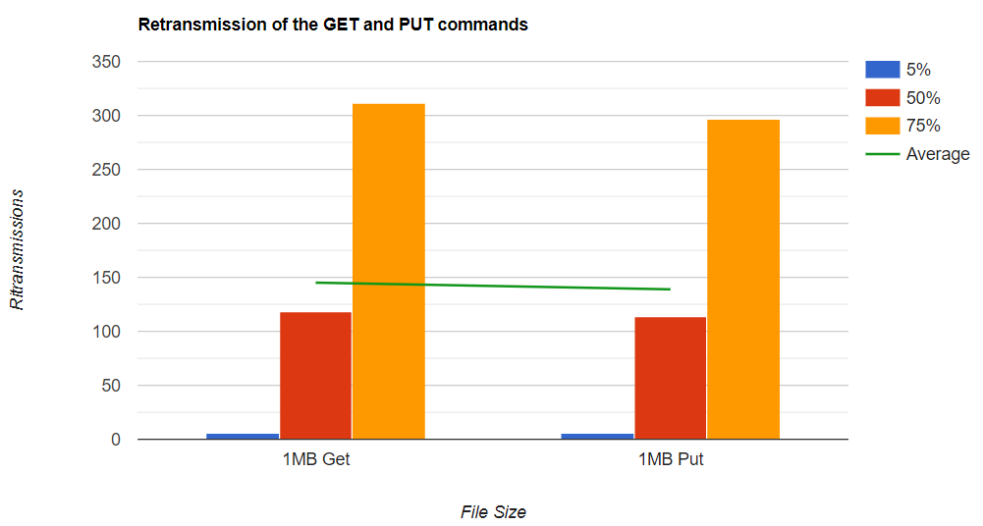
Per osservare meglio questo comportamento, questa crescita irregolare, osserviamo il grafico in 3, il quale mostra una varianza tra le probabilità, ovvero quanto varia in maniera percentuale il numero di ritrasmissioni che vengono ricevute rispetto al precedente tasso di perdita considerato. Cioè per esempio di quanto aumentano in percentuale, il numero di ritrasmissioni del pacchetto di un file da 1MB dove ha il tasso di perdita del 50% ha una percentuale di ritrasmissione del 225% maggiore rispetto ad un tasso di perdita del 5%. La prima parte ci mostra l’aumento percentuale di ritrasmissioni con una percentuale di perdita compresa tra lo 0% ed il 5%, poi tra il 5% ed il 50%, poi tra il 50% ed il 75%. La cosa interessante che si può notare è ovviamente come addirittura mediamente in percentuale c’è un incremento minore nel file da 10MB a confronto con quello da 1MB, si nota come i file da 1MB il numero di ritrasmissioni, nel caso di probabilità di errore pari al 5% e quello pari al 50%, ha un incremento percentuale di ritrasmissioni decisamente maggiore. Non va confusa la probabilità con il numero di ritrasmissioni, che ovviamente sono maggiori come si vede dalla figura 4. Il grafico della figura che a primo impatto sembra dubbio e non chiaro, ma il suo essere così estremo, è la prova di cui parlavamo, cioè se percentualmente aumenta in maniera nettamente più elevata il file da 1MB, quello da 10MB ha un numero di ritrasmissioni decisamente più elevato. Questo ci fa capire che i file di grandi dimensioni, se trasmessi con una connessione instabile anche leggermente, porterà dei ritardi maggiori, il che è logico, ma per dare un’occhiata leggermente più accurata potremmo fare un focus sulla prima parte, dove possiamo metterli a confronto con la put. Cominciando così ad introdurre anche quest’altro comando. Osserviamo nella Figura 5, come casualmente il numero delle ritrasmissioni tra il 50% delle probabilità di perdita dell’operazione put ed il 50% dell’operazione get è diverso, però mediamente sono praticamente identiche, come ci si aspettava, stessa cosa la possiamo notare anche nella figura 6, dove sono ancora più simili, dove la media è praticamente identica, questo è un buon segno e premia una costruzione neutra delle 2 operazioni, ovvero entrambe hanno lo stesso comportamento, e quindi stesse prestazioni.

Figura 5, numero di ritrasmissioni delle operazioni put e get, per file da 50Byte.

Figura 4, numero di ritrasmissioni dell’operazione GET.

Figura 3, grafico che mostra quanto percentualmente varia la quantità di ritrasmissioni per i file di varie dimensioni.

Figura 6, numero di ritrasmissioni delle operazioni put e get, per file da 1MB

## **Operazione put:**

Continuiamo ad analizzare la put, sempre tenendo presente cosa fatto fino ad ora, ovvero la get, per avere un confronto e una visualizzazione a noi già famigliare.

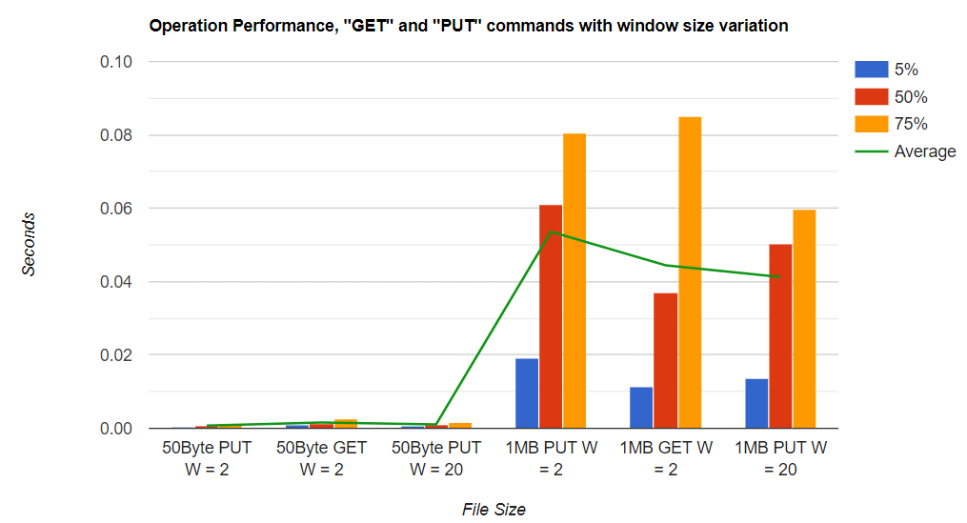
Come ci aspettavamo, osservando la figura 7, i tempi di trasmissione sono molto simili, soprattutto mediamente, essendo state fatte anche un tempo distante, ci sono vari fattori che hanno influito, e fatto variare magari più di quanto possa essere intuitivo, restando ovviamente su quella media e quei picchi tutto sommato. Infatti si nota che la get sembra essere più lenta nel caso di file di 50Byte e più veloce nel caso di 1MB. Anche se presa un vario numero di test, ci fa capire quanto sia vario e quanto può influenzare anche un cambio di ambiente che può sembrare esterno al problema, oppure anche un piccolo aggiornamento di sistema.

Figura 7, ci mostra a confronto i tempi di invio tra put e get

### **Variazioni della dimensione della finestra:**

Un’altra cosa da dover prendere in considerazione è la variazione della dimensione della finestra, un esempio lo possiamo vedere nella Figura 8. Ovviamente si nota che sarà più rapido soprattutto per i file di grandi dimensioni, la finestra aiuta molto l’invio, per i file di grandi dimensioni. Si nota la differenza tra i 2 put principalmente, essendo stati fatti i testi in condizioni simili, sono decisamente più facili e giusti da comparare.

Figura 8, il confronto temporale tra get e put con finestra dimensione = 2 e put con finestra di dimensione = 20.

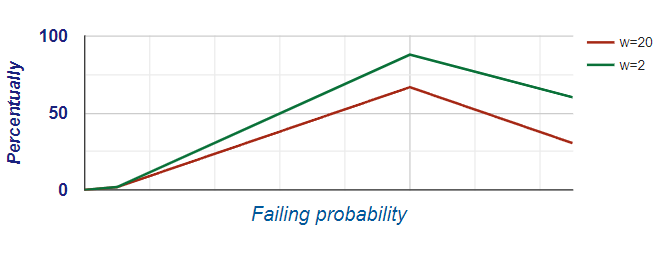
Osserviamo la Figura 9, che rappresenta la probabilità di ritrasmissione di un file (sulle y) in base alla probabilità (sulle x) al variare della finestra di trasmissione. Si nota che con una finestra diversa, ma file e probabilità di perdita identica, una finestra, tende a fare meno ritrasmissioni. Il motivo non è del tutto chiaro, poiché il numero di pacchetti da inviare rimane invariato, solo la grandezza della finestra viene ingrandita, possiamo però ipotizzare che sia ha meno probabilità di errore se inviate in determinati momenti, e meno in altre. Questo comportamento non è molto estraneo o lontano dalla verità, per questo lo possiamo considerare accettabile e utile all’analisi.

Figura 9, variazione della percentuale di ritrasmissione dei file, in base alla grandezza della finestra.

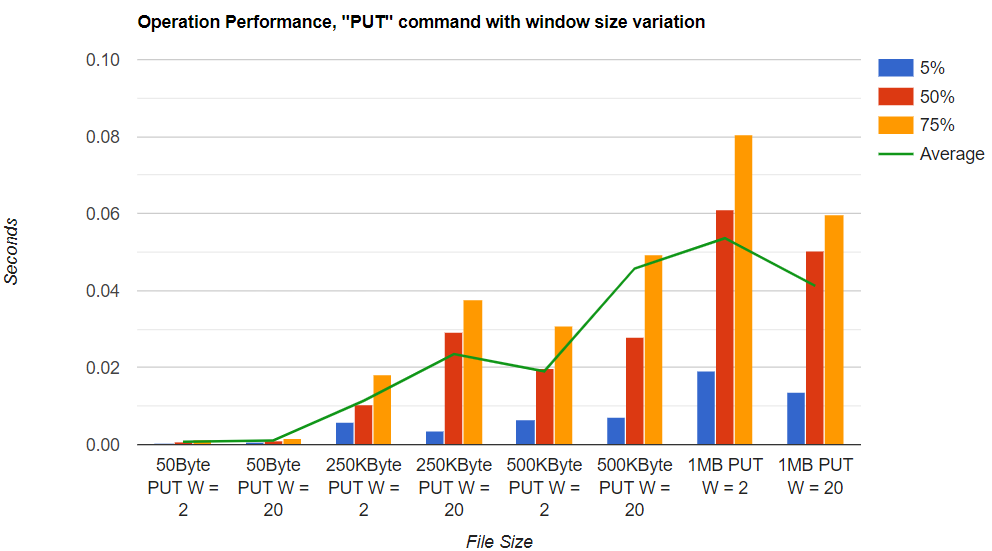
Possiamo però tornare a dei grafici molto più facili da leggere e comprendere di quest’ultimo, ovvero vediamo l’incremento temporale della nostra cara put, però in condizioni diverse di finestra, e di file con dimensioni diverse.

Figura 10, rappresenta il grafico delle variazioni temporali, al cambiare della finestra e della probabilità di fallimento dell’invio dei pacchetti.

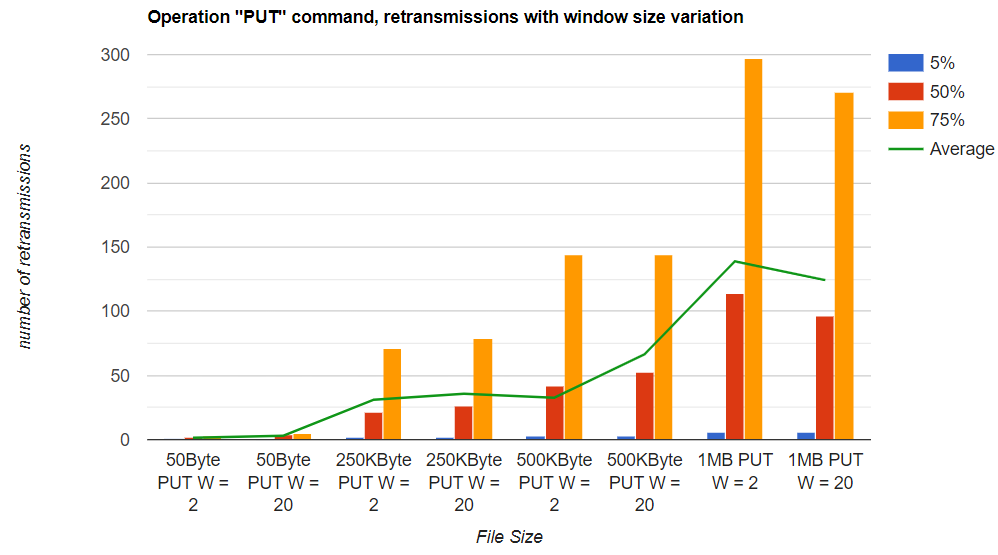
Cominciamo con l’analisi della Figura 10, si nota già da subito una grande differenza tra i file di stessa dimensione, stessa probabilità di errore, ma finestra diversa. Infatti, si nota che con la finestra da 20, il tempo è maggiore di invio per i file di piccola dimensione, d’altro canto per i file di grandi dimensioni, inizia ad avere un comportamento inverso, ovvero il tempo di trasmissione di file di grandi dimensioni con una finestra più ampia, sono più bassi e quindi convenevoli. Con una probabilità di errore alta, i tempi sono sempre elevati, però si notano dei miglioramenti dei tempi. Una parentesi si può aprire sul caso comprendente la finestra di dimensione 20 ed il file da 250KB. Poiché ha un comportamento anomalo, ovvero, è normalmente più alto, del suo simile con finestra di grandezza 2 come accennato in precedenza, e non sorprendentemente maggiore di quello da 500KB con finestra = 2. La cosa strana è che con una probabilità del 5% riesce a raggiungere dei tempi inferiori a quelli del suo predecessore, il che è un mistero, probabile che sia stato dato da un caso multiplo di eventi favorevoli.

Figura 11, numero di ritrasmissioni fatte da dalla put con varie dimensioni delle finestre.

Altra piccola osservazione da fare è sulla figura 11, consideriamo ora solo le ritrasmissioni della put, al variare della dimensione della finestra, ovviamente un discorso analogo lo possiamo fare per la get, essendo operazioni simili, ma inverse. Notiamo come le ritrasmissioni, per i file dimensioni ridotte siano decisamente maggiori nel caso di finestra di 20, però nel caso di file di dimensioni maggiori, i ruoli si scambiano e diventa più efficace avere una finestra maggiore. Questo è un comportamento prevedibile ed aspettato, poiché si comporta meglio per file di dimensioni maggiori, una finestra di dimensioni maggiori.

### **Time-out fisso:**

Figura 12, i tempi al variare della probabilità di errore e del time-out da statico a dinamico

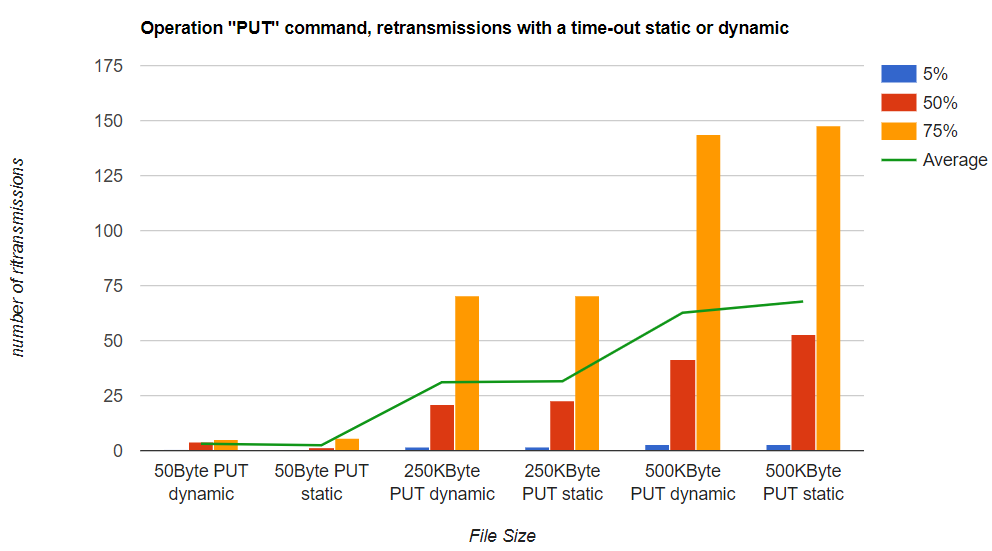
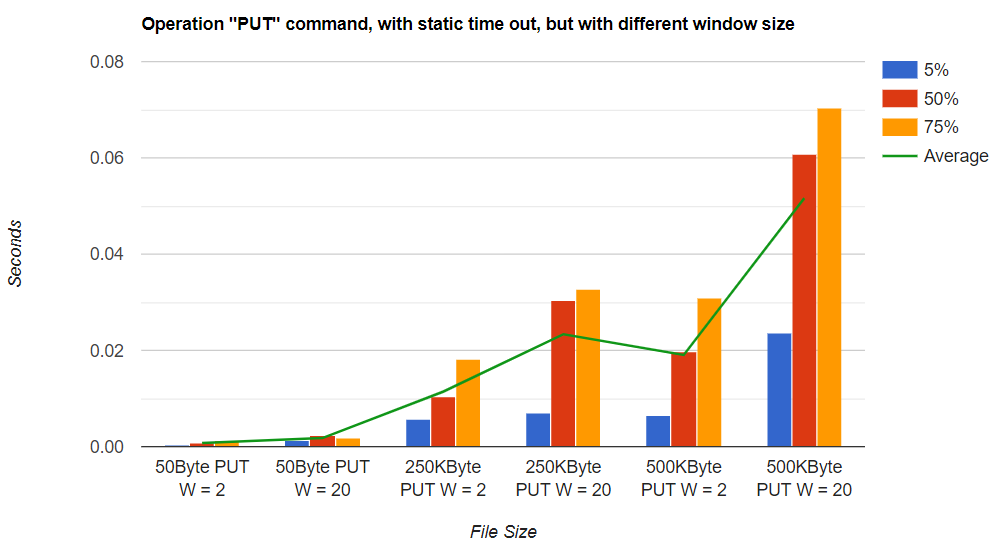
Ora vediamo un’altra variabile che può essere un’interessante casistica da studiare. Cioè quella del time-out, che finora anche se non scritto in modo diretto, si è parlato di un time-out dinamico! Le osservazioni da fare sono subito chiare guardando il grafico in Figura 12, ovviamente, un time-out sempre fisso porta ad avere dei tempi diversi, ma il comportamento è quello che ci aspettavamo, ovvero con un time-out dinamico i tempi sono minori se guardiamo i file di grandi dimensioni, altrimenti i tempi sono molto simili, poiché il time out non ha tempo per adattarsi. Altro comportamento che ci aspettavamo è quello di vedere i tempi più alti anche quando c’è meno errore. Infatti, come si nota in modo chiaro nel caso del file da 500KB della Figura 12, anche se pochi errori i tempi sono molto elevati, poiché la correzione del time-out non avviene mai e quindi anche se ci sono pochi errori, non diminuisce mai e quindi anche se si possono ritrasmettere file, non lo farà immediatamente, aspetta lo scadere del timer.

Figura 13, rappresento il numero di ritrasmissioni di pacchetti al variare del tipo di time-out, dinamico e statico.

Figura 14, rappresento la differenza temporale tra la finestra di grandezza 2 e di grandezza 20, con time-out fisso.

Un’altra possibile cosa prevedibile, la vediamo nella Figura 13, che ci mostra che mediamente il numero di ritrasmissioni sono proprio le stesse, poiché non influenza in nessun tipo di modo con la probabilità di fallimento nell’invio dei file. Si vedono leggere oscillazioni medie, ma sono date da causalità non da motivi veri e propri. Nella Figura 14, vediamo con poca sorpresa che i tempi con w = 20, sono maggiori di w = 2.