pandas を用いた ping 通信遅延分布の分析

岡本真拓

作成年月日：June 14, 2025

1　調査内容

ping などの RTT を測定するツールを用いて、通信遅延分布がどうなっているか調べ、グラフ化する。但し、統計的な正当性を担保するため、ping は高精度なものを利用し、テストは複数箇所行う。

2　設定条件

Linux の ping コマンドを利用した。まず外部のサイト、今回は慶應義塾大学の hp を目的地としてパケットを送信したが、情報が全く届かなかった。実際に利用した Linux には以下のような画面が表示された。

```
$ ping -c 3 www.keio.ac.jp
PING e100342.dscb.akamaiedge.net (23.219.170.25) 56(84) bytes of data.

--- e100342.dscb.akamaiedge.net ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2035ms
  traceroute www.keio.ac.jp
traceroute to www.keio.ac.jp (23.219.170.25), 30 hops max, 60 byte packets
 1  192.168.16.252 (192.168.16.252)  0.367 ms  0.438 ms  0.301 ms
 2  icsintsvgw.ics.es.osaka-u.ac.jp (133.1.240.254)  0.761 ms  0.923 ms  1.074 ms
 3  icsintgw.ics.es.osaka-u.ac.jp (133.1.240.81)  0.177 ms  0.316 ms  0.245 ms
 4  * * *
 5  * * *
```

そこで、traceroute コマンドで 3 つの経由地を確認し、確実に届くのでそれら全てに 1 つずつパケットを送信し、送信時間検出と平均、分散などの統計情報を取得することにした。十分なデータを確保するため、リクエスト数は 100 とした。

Table 1: 条件

条件	数値
テスト数	3
リクエスト数	100

3　遅延分布グラフ

まずデータを csv ファイルに取り込んだ後、pandas を使った python プログラムで分析とグラフ化を行った。以下のようなプログラムである。

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# CSV ファイルの読み込み
# ここに csv ファイル名を入れる
df = pd.read_csv('ping_results_complete.csv')

# time_ms 列の平均と分散の計算
mean_time = df['time_ms'].mean()
var_time = df['time_ms'].var()

print(f"平均応答時間: {mean_time:.3f} ms")
print(f"応答時間の分散: {var_time:.6f} ms^2")

# グラフの描画
plt.figure(figsize=(10, 5))
plt.plot(df['icmp_seq'], df['time_ms'], marker='o', linestyle='-', label='Ping time')
plt.axhline(mean_time, color='red', linestyle='--', label=f'Mean: {mean_time:.3f} ms')
plt.title('Ping 応答時間 (icmp_seq vs time_ms)')
plt.xlabel('icmp_seq')
plt.ylabel('time (ms)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

このプログラムによってグラフ化した際、初めに、以下のようなグラフが検出された。

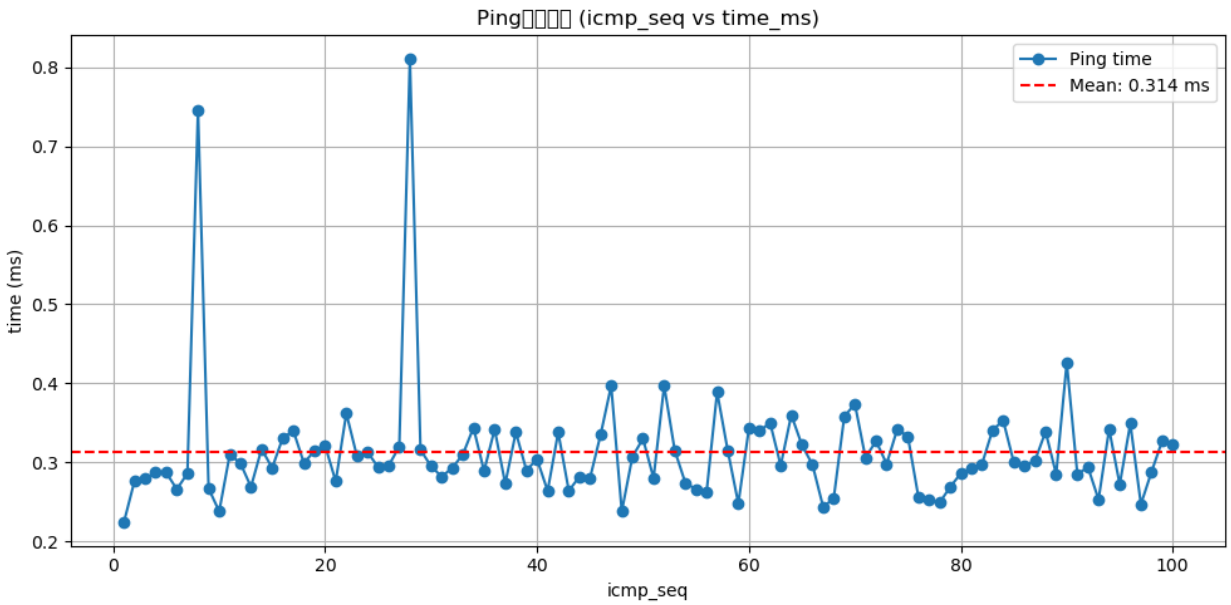

Figure 1: icsintgw のグラフ

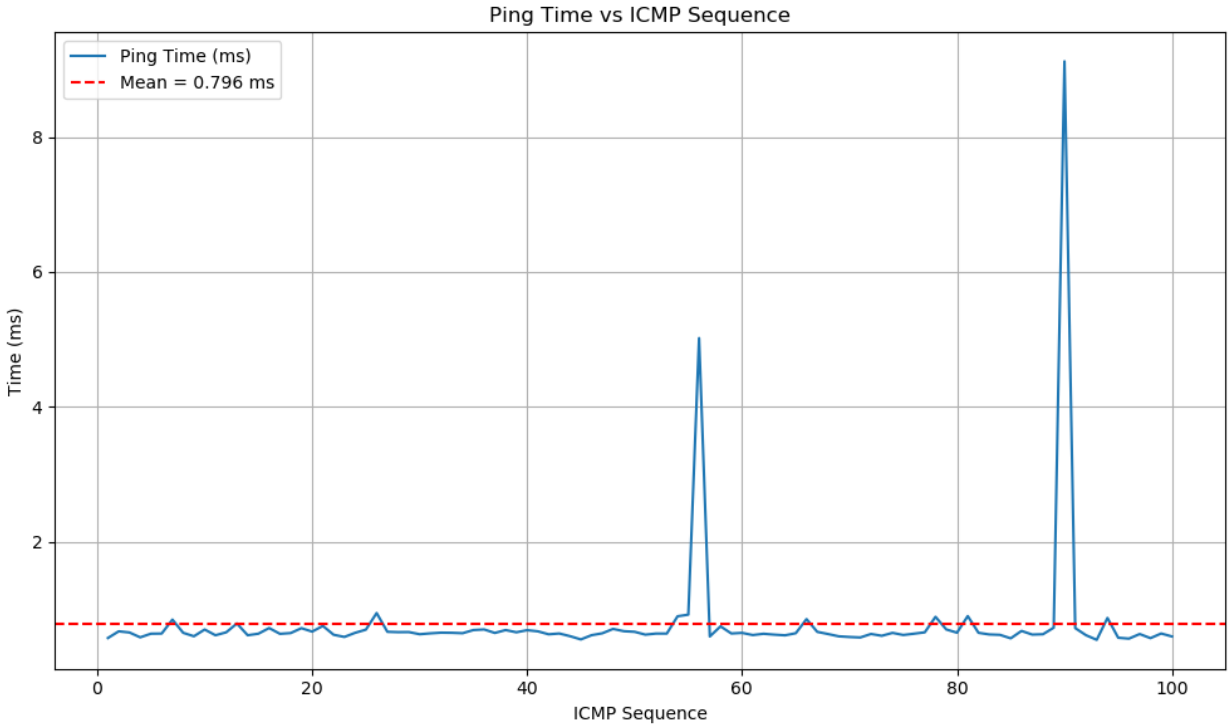

Figure 2: icsintsvgw のグラフ

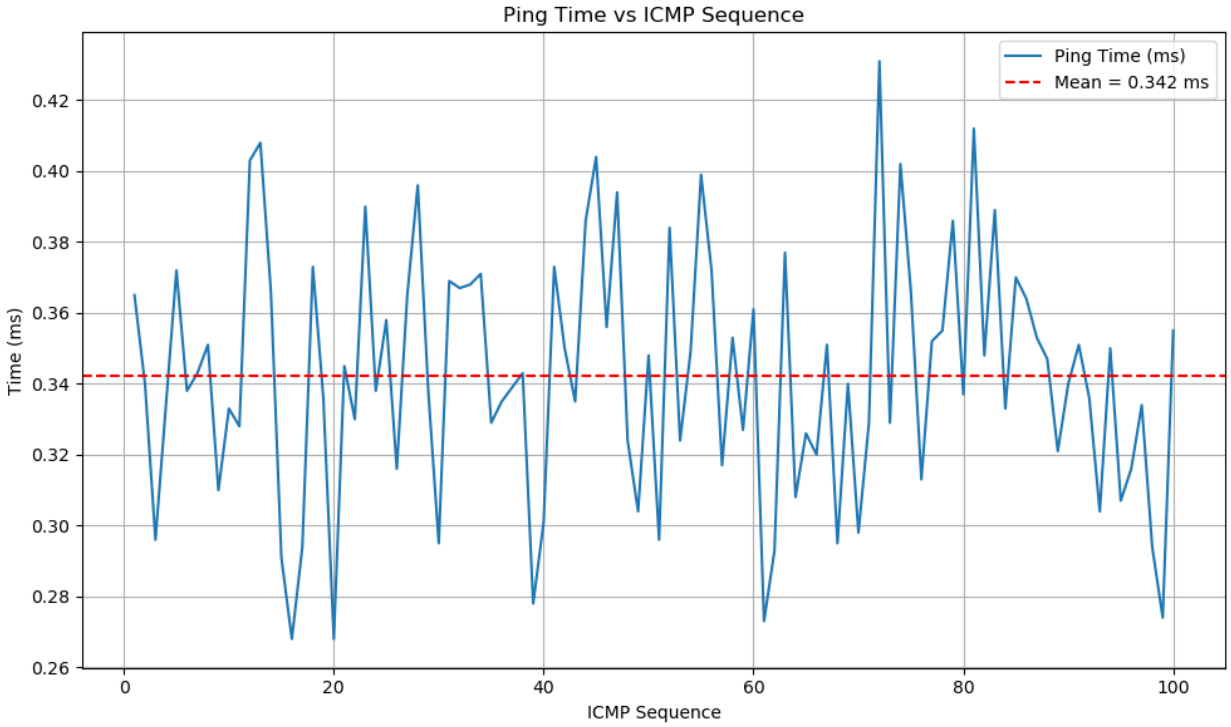

Figure 3: 192.168.16.252 のグラフ

しかし、上 2 つのグラフでは大きく外れた値があったので、外れ値を除外したグラフを新たに生成した。またこのグラフを生成するために python の pandas を利用した。以下のようなプログラムコードである。

```
import re
import pandas as pd
import matplotlib.pyplot as plt

# ping 結果のテキストを貼り付け
# ここに terminal の生データをコピペする
ping_output = """(
)"""

# icmp_seq と time を抽出する
pattern = r"icmp_seq=(\d+).*?time=([\d\.]+) ms"
matches = re.findall(pattern, ping_output)

# データフレーム化
data = pd.DataFrame(matches, columns=["icmp_seq", "time"])
data["icmp_seq"] = data["icmp_seq"].astype(int)
data["time"] = data["time"].astype(float)

# CSV に保存
data.to_csv("ping_data.csv", index=False)
print("ping_data.csv に保存しました。")

# 外れ値（2 秒以上）を除外
df = pd.read_csv("ping_data.csv")
# filtered_df = df[df["time"] < 5]
filtered_df = df

# 平均と分散の計算
# 外れ値除去後の平均と分散
mean_time_filtered = filtered_df["time"].mean()
var_time_filtered = filtered_df["time"].var()

print(f"外れ値除去後の平均応答時間: {mean_time_filtered:.3f} ms")
print(f"外れ値除去後の分散: {var_time_filtered:.6f}")

# グラフ描画
plt.figure(figsize=(10, 6))
plt.plot(filtered_df["icmp_seq"], filtered_df["time"], label="Ping Time (ms)")
plt.axhline(y=mean_time_filtered, color='r', linestyle='--', label=f"Mean = {mean_time_filtered:.3f} ms'
plt.title("Ping Time vs ICMP Sequence")
plt.xlabel("ICMP Sequence")
plt.ylabel("Time (ms)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("ping_graph.png")
plt.show()
```

このプログラムによりグラフ化した際、外れ値を除いた以下のようなグラフが生成された。外れ値はそれぞれ 0.5 秒、2 秒と定義した。

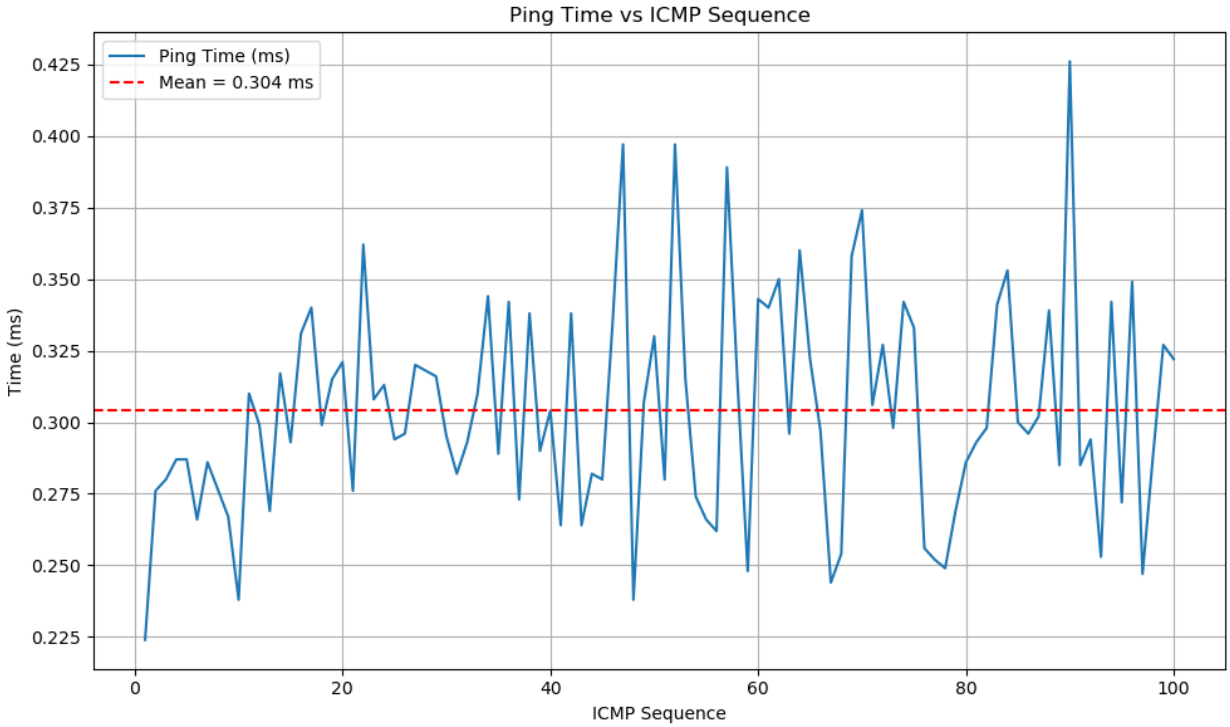

Figure 4: icsintgw のグラフ

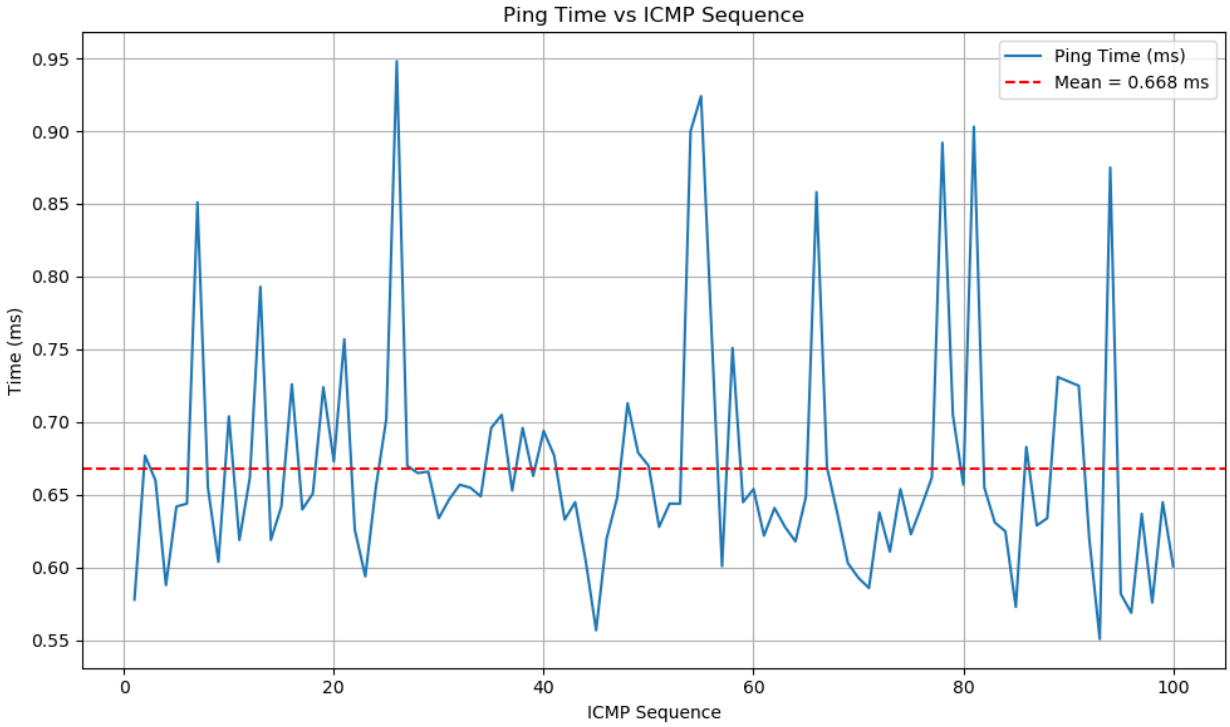

Figure 5: icsintsvgw のグラフ

グラフの分析結果、以下のような数値が得られた。外れ値は除外したデータを採用している。

Table 2: icsintgw 分析結果

条件	時間 (us)
平均値	304
分散	6

Table 3: icsintsvgw 分析結果

条件	時間 (us)
平均値	668
分散	6

Table 4: 192.168.16.252 分析結果

条件	時間 (us)
平均値	342
分散	1

4　考察、感想

折れ線グラフにする際に外れ値があるとグラフが視覚的に非常に分かりずらくなる上、特に分散の値が大幅に変わるため、規定値で除外することの大切さが理解できた。上 2 つのデータは、外れ値を除外してもなお分散が比較的大きかった。これについて私は、IP アドレスがそのまま書いてあるかドメイン名で書いてあるかの違いによっておこるものであると洞察した。さらにこれについて、DNS サーバーでの IP アドレスの翻訳時間の影響が大きく、セキュリティ上調査が必要なパケットが www のドメイン名で始まるものの方が多いから起こるものではないかと予想した。これを特定するために、追加で実験分析を行うことにした。

5　追加実験分析の手法

全く同じ宛先に IP アドレスの数値と、ホスト名等の組合せの文字列の 2 通りで、リクエスト数を 300 と 1000 に増やして、最大値と標準偏差を中心に比較する。これにより、宛先の書き方による外れ値の多さの相関を抽出することが出来る。

6　追加分析のグラフ

以下の画像は平均と分散の値である。

```
mah-okmt@exp004:~$ /usr/bin/python3 /home/exp/mah-okmt/ping_changed.py
ping_data.csv に保存しました。
外れ値除去後の平均応答時間: 0.294 ms
外れ値除去後の分散: 0.047594
mah-okmt@exp004:~$ /usr/bin/python3 /home/exp/mah-okmt/ping_changed.py
ping_data.csv に保存しました。
外れ値除去後の平均応答時間: 0.286 ms
外れ値除去後の分散: 0.003937
mah-okmt@exp004:~$ /usr/bin/python3 /home/exp/mah-okmt/ping_changed.py
ping_data.csv に保存しました。
外れ値除去後の平均応答時間: 0.286 ms
外れ値除去後の分散: 0.003937
mah-okmt@exp004:~$ /usr/bin/python3 /home/exp/mah-okmt/ping_changed.py
ping_data.csv に保存しました。
外れ値除去後の平均応答時間: 0.301 ms
外れ値除去後の分散: 0.010605
mah-okmt@exp004:~$ /usr/bin/python3 /home/exp/mah-okmt/ping_changed.py
ping_data.csv に保存しました。
外れ値除去後の平均応答時間: 0.281 ms
外れ値除去後の分散: 0.011034
mah-okmt@exp004:~$
```

Figure 6: 平均と分散の値

まずはリクエストが 300 回の時のグラフである。

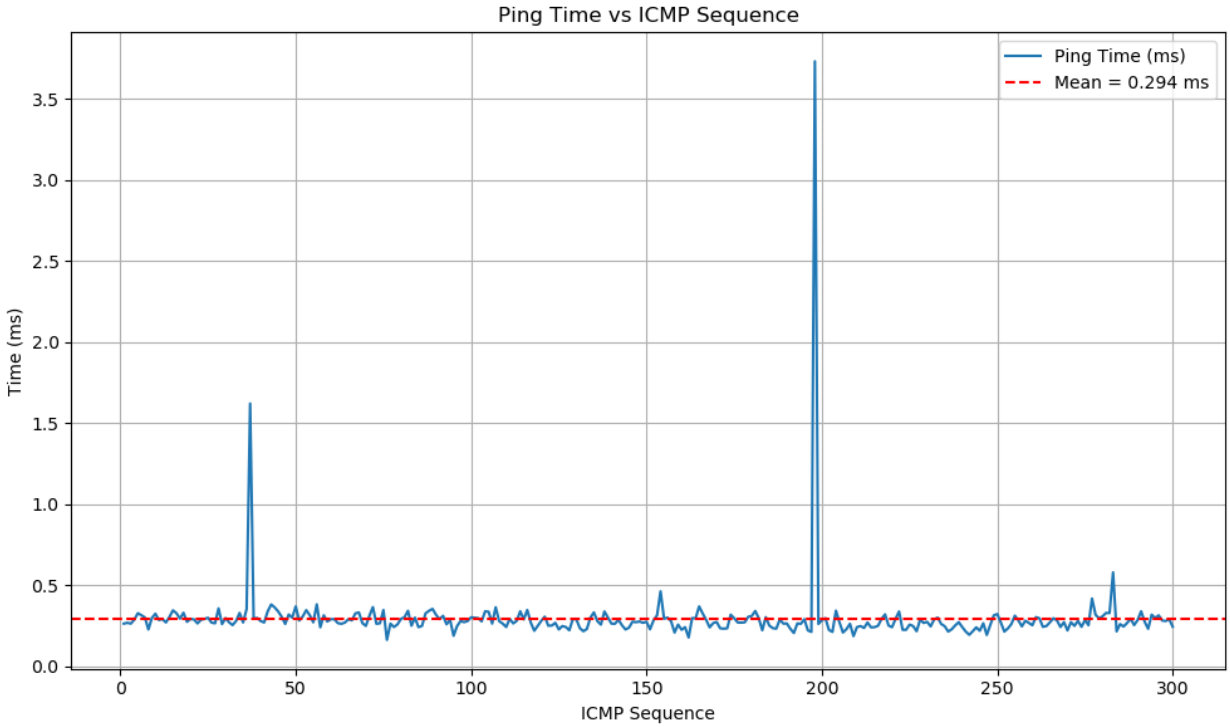

Figure 7: リクエスト 300、文字列で宛先を書いた場合のグラフ

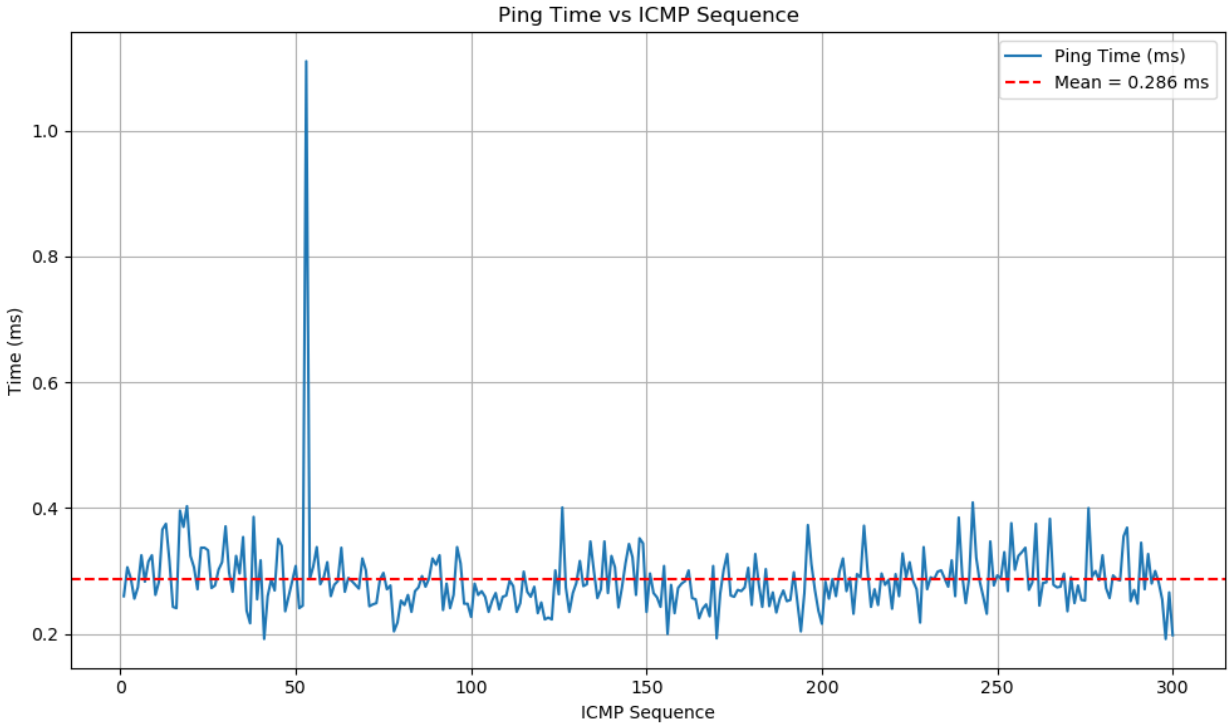

Figure 8: リクエスト 300、IP アドレスで直接参照する場合のグラフ

次にリクエストが 1000 回の時のグラフである。

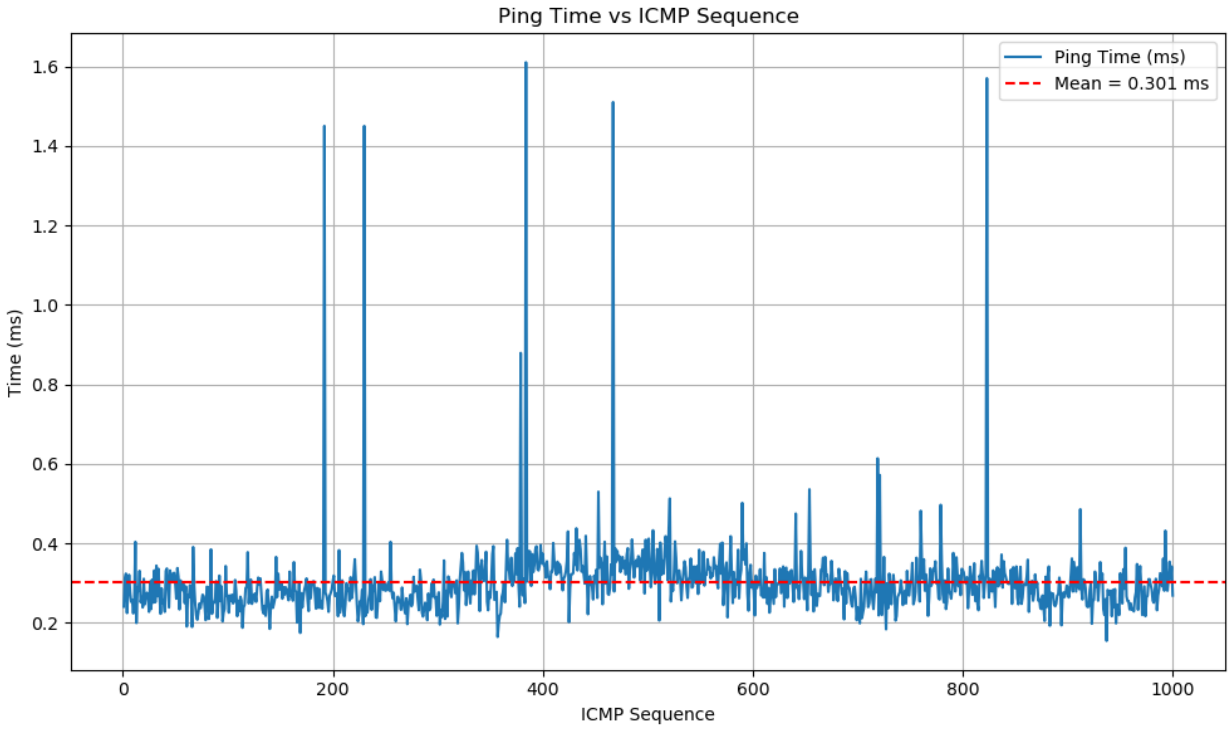

Figure 9: リクエスト 1000、文字列で宛先を書いた場合のグラフ

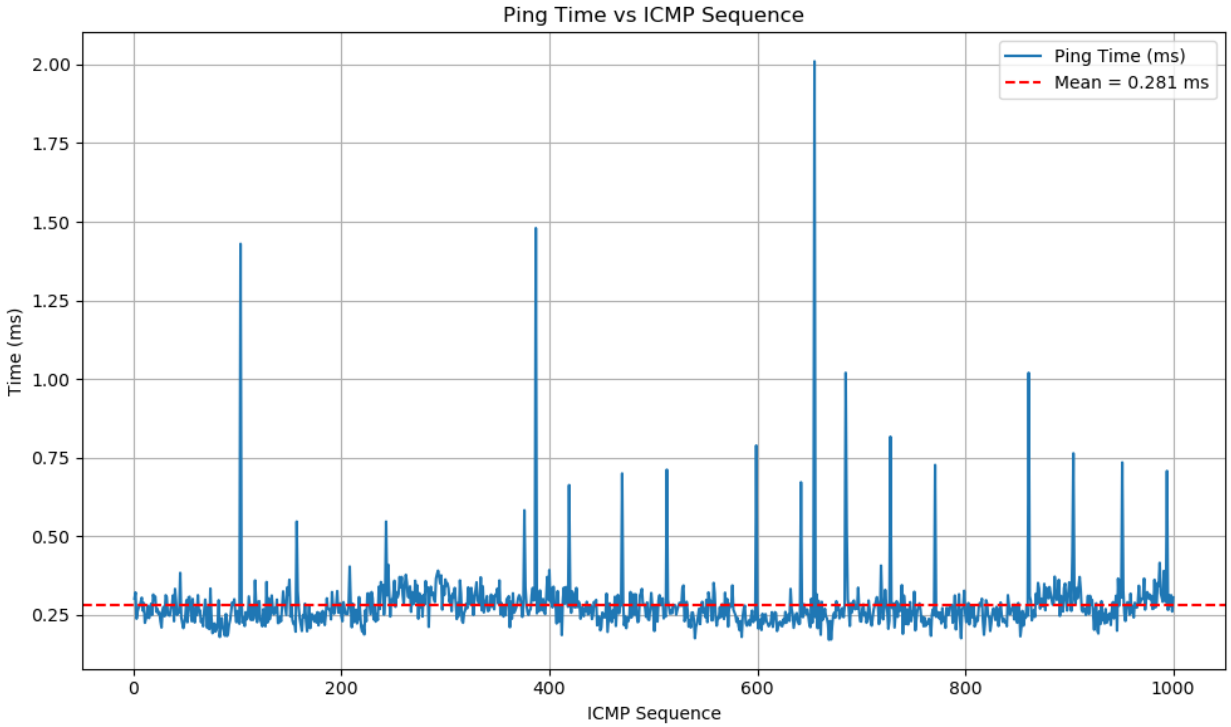

Figure 10: リクエスト 1000、IP アドレスで直接参照する場合のグラフ

精密に相関を調べる為にはもっと多くのデータが必要だが、時間の制約があり最低限のデータを取得した。文字列で宛先を書いた場合、IP アドレスで直接参照した場合に比べて、300 の時は分散、平均値、最大値ともに前者が大きい。一方で 1000 の場合は平均値は大きいものの分散はあまり変わらなかった。これにより、中程度の情報量の場合は、IP アドレスで通信した方が文字列で通信するよりも速く通信できることがわかった。しかし、ドメイン名で宛先を書いた場合、文字列ハッシュ値分析などによってセキュリティ調査に時間をかけている可能性が高いので、高速性を求める通信なのか安全性を求める通信なのかにより使い分けることが最適な使い方だと結論付けた。多くの情報を通信している場合、直近で審査済みの情報が多いという理由でパケット審査速度が早まっているのではないかと考察した。（この考察はあくまで自身の考察であり、事実と異なる可能性があります。）