In [121…

```
%pip install pandas matplotlib seaborn scipy statsmodels scikit-learn
```

```
Requirement already satisfied: pandas in e:\documentos\ufrj\10 periodo\probest\tr
abalho\projeto-probest-parte-1\.venv\lib\site-packages (2.3.3)
Requirement already satisfied: matplotlib in e:\documentos\ufrj\10 periodo\probes
t\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (3.10.7)
Requirement already satisfied: seaborn in e:\documentos\ufrj\10 periodo\probest\t
rabalho\projeto-probest-parte-1\.venv\lib\site-packages (0.13.2)
Requirement already satisfied: scipy in e:\documentos\ufrj\10 periodo\probest\tra
balho\projeto-probest-parte-1\.venv\lib\site-packages (1.16.2)
Requirement already satisfied: statsmodels in e:\documentos\ufrj\10 periodo\probe
st\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (0.14.5)
Requirement already satisfied: scikit-learn in e:\documentos\ufrj\10 periodo\prob
est\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (1.7.2)
Requirement already satisfied: numpy>=1.26.0 in e:\documentos\ufrj\10 periodo\pro
best\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from pandas) (2.3.
4)
Requirement already satisfied: python-dateutil>=2.8.2 in e:\documentos\ufrj\10 pe
riodo\probest\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from pand
as) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in e:\documentos\ufrj\10 periodo\prob
est\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from pandas) (2025.
2)
Requirement already satisfied: tzdata>=2022.7 in e:\documentos\ufrj\10 periodo\pr
obest\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from pandas) (202
5.2)
Requirement already satisfied: contourpy>=1.0.1 in e:\documentos\ufrj\10 periodo
\probest\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from matplotli
b) (1.3.3)
Requirement already satisfied: cycler>=0.10 in e:\documentos\ufrj\10 periodo\prob
est\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from matplotlib)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in e:\documentos\ufrj\10 periodo
\probest\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from matplotli
b) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in e:\documentos\ufrj\10 periodo
\probest\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from matplotli
b) (1.4.9)
Requirement already satisfied: packaging>=20.0 in e:\documentos\ufrj\10 periodo\p
robest\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from matplotlib)
(25.0)
Requirement already satisfied: pillow>=8 in e:\documentos\ufrj\10 periodo\probest
\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from matplotlib) (12.
0.0)
Requirement already satisfied: pyparsing>=3 in e:\documentos\ufrj\10 periodo\prob
est\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from matplotlib)
(3.2.5)
Requirement already satisfied: patsy>=0.5.6 in e:\documentos\ufrj\10 periodo\prob
est\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from statsmodels)
(1.0.2)
Requirement already satisfied: joblib>=1.2.0 in e:\documentos\ufrj\10 periodo\pro
best\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from scikit-learn)
(1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in e:\documentos\ufrj\10 peri
odo\probest\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from scikit
-learn) (3.6.0)
Requirement already satisfied: six>=1.5 in e:\documentos\ufrj\10 periodo\probest
\trabalho\projeto-probest-parte-1\.venv\lib\site-packages (from python-dateutil>=
2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

Vitor Taranto DRE: 121063585

Projeto Probest - Parte 1

Fase 1: Análise Exploratória de Dados (EDA)

In [36]:
```
import pandas as pd
import numpy as np

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 1000)
pd.options.display.float_format = '{:.6f}'.format
```

1. Carga e Verificação dos Dados

Carregamento do arquivo `ndt_tests_corrigido.csv` e verificação inicial da estrutura dos dados (colunas e primeiras linhas).

In [37]:
```
file_path = 'ndt_tests_corrigido.csv'

try:
    df = pd.read_csv(file_path)
    print(f"Dataset carregado: {file_path}")
    print(f"Formato: {df.shape}")

    print("\nColunas disponíveis:")
    print(df.columns.tolist())

    print("\nVerificando dados (primeiras linhas):")
    print(df.head())

except FileNotFoundError:
    print(f"ERRO: Arquivo não encontrado em '{file_path}'")
except Exception as e:
    print(f"Ocorreu um erro: {e}")
```

```
Dataset carregado: ndt_tests_corrigido.csv
Formato: (7087, 8)

Colunas disponíveis:
['timestamp', 'download_throughput_bps', 'rtt_download_sec', 'upload_throughput_b
ps', 'rtt_upload_sec', 'packet_loss_percent', 'client', 'server']

Verificando dados (primeiras linhas):
                      timestamp  download_throughput_bps  rtt_download_sec  uplo
ad_throughput_bps  rtt_upload_sec  packet_loss_percent    client    server
0  2025-08-09 15:28:02.000 +0000          82236567.995454          0.231862
69732172.059986        0.247727             0.000000  client12  server06
1  2025-08-09 15:30:11.000 +0000         902731479.238230          0.012000
852177960.399178        0.005423             0.008226  client01  server07
2  2025-08-10 04:27:43.000 +0000         591065502.098614          0.014000
281218794.113733        0.014544             5.954284  client13  server07
3  2025-08-09 22:45:07.000 +0000         672113910.536765          0.011000
113539960.296302        0.010482             0.261528  client12  server07
4  2025-08-10 04:49:21.000 +0000         812208745.286963          0.009000
685790474.925742        0.009368             1.381646  client03  server03
```

2. Pré-processamento: Normalização da Fração de Perda

A coluna `packet_loss_percent` está em formato percentual (0-100). Para a modelagem (especialmente a Beta-Binomial), é necessário convertê-la para uma fração (proporção de 0 a 1). A célula seguinte cria a nova coluna `packet_loss_fraction`.

In [135…

```python
# Criar a coluna de fração de perda dividindo por 100
if 'packet_loss_percent' in df.columns:
    df['packet_loss_fraction'] = df['packet_loss_percent'] / 100
    print("Coluna 'packet_loss_fraction' criada.")

    # Verificar a nova coluna
    print(df[['packet_loss_percent', 'packet_loss_fraction']].head())
else:
    print("AVISO: Coluna 'packet_loss_percent' não encontrada.")
```

```
Coluna 'packet_loss_fraction' criada.
   packet_loss_percent  packet_loss_fraction
0             0.000000              0.000000
1             0.008226              0.000082
2             5.954284              0.059543
3             0.261528              0.002615
4             1.381646              0.013816
```

2. Pré-processamento: Normalização da Fração de Perda

A coluna `packet_loss_percent` está em formato percentual (0-100). Para a modelagem (especialmente a Beta-Binomial), é necessário convertê-la para uma fração (proporção de 0 a 1). A célula seguinte cria a nova coluna `packet_loss_fraction`.

In [39]:

```python
variables_to_analyze = [
    'download_throughput_bps',
```

```
    'upload_throughput_bps',
    'rtt_download_sec',
    'rtt_upload_sec',
    'packet_loss_fraction'
]

group_by_client_col = 'client'
group_by_server_col = 'server'

def q_0_9(x):
    return x.quantile(0.9)

def q_0_99(x):
    return x.quantile(0.99)

stats_to_calculate = ['mean', 'median', 'var', 'std', q_0_9, q_0_99]

print("Variáveis e estatísticas (corrigidas) definidas.")
```

```
Variáveis e estatísticas (corrigidas) definidas.
```

4. Análise Agrupada por Cliente

Cálculo das estatísticas descritivas para cada variável, agrupadas por `client`.

In [40]:
```
try:
    stats_by_client = df.groupby(group_by_client_col)[variables_to_analyze].agg(

    stats_by_client.rename(columns={'std': 'std_dev'}, level=1, inplace=True)

    print("--- Estatísticas por Cliente ---")
    print(stats_by_client)

except KeyError as e:
    print(f"ERRO: Chave não encontrada. Verifique os nomes das colunas. Detalhe:
except Exception as e:
    print(f"Ocorreu um erro: {e}")
```

```
--- Estatísticas por Cliente ---
         download_throughput_bps
upload_throughput_bps
rtt_download_sec                                                rtt_upload_sec
packet_loss_fraction
                              mean           median                       var
std_dev            q_0_9           q_0_99                   mean           median
var          std_dev            q_0_9           q_0_99             mean   median
var  std_dev    q_0_9   q_0_99           mean   median      var  std_dev    q_0_9
q_0_99                 mean   median      var  std_dev    q_0_9   q_0_99
client
client01        650329090.178619 892829694.920871 119496947928815232.000000 34568
3305.828927 914699457.161784 931600136.961252      714049706.694203 878876045.232
304  82982288096875040.000000 288066464.721035 911211386.162695 917368382.556616
0.044897 0.011000 0.002887 0.053727 0.120452 0.125607       0.031305 0.006363 0.0
02170 0.046584 0.119980 0.122391             0.005241 0.000055 0.000279 0.016689
0.007665 0.084549
client02        639309508.102630 824316656.031722 105923125087020336.000000 32545
8330.799843 911271977.779224 927931763.099398      561951628.703508 629699678.586
210  45068517857858344.000000 212293471.067431 757271287.732368 887298800.140516
0.039058 0.010000 0.002748 0.052420 0.118634 0.125138       0.029352 0.004243 0.0
02319 0.048154 0.117383 0.125348             0.007814 0.000655 0.000353 0.018798
0.018138 0.109424
client03        613278409.806640 812445409.301693 109292320534127808.000000 33059
3890.648523 909724532.323770 921749795.019768      521209803.892055 592773243.607
905  44067519548776928.000000 209922651.347531 732116419.306651 763123996.408931
0.044448 0.010000 0.003161 0.056224 0.119971 0.219486       0.032986 0.004359 0.0
02554 0.050533 0.118953 0.187624             0.006750 0.000749 0.000207 0.014378
0.016331 0.071747
client04        580578884.379963 799150140.010026 119998110586502000.000000 34640
7434.369561 908749660.952150 926014925.007047      548986225.577962 622324165.248
919  61465639913075368.000000 247922649.052230 859634869.384209 891760668.459451
0.050854 0.012000 0.003052 0.055240 0.122000 0.126624       0.035188 0.008341 0.0
04895 0.069962 0.121319 0.126381             0.005128 0.000493 0.000173 0.013153
0.011248 0.063045
client05        309530969.173471 106962886.116866 112060969040776176.000000 33475
5088.147703 892519778.446198 913623024.789973      347752207.837473 108594534.925
656 123472333805976192.000000 351386302.814973 898651869.506160 910938734.633852
0.045166 0.010000 0.003010 0.054865 0.119000 0.219124       0.029707 0.004112 0.0
02236 0.047286 0.118595 0.120995             0.063119 0.082764 0.002805 0.052962
0.126384 0.150083
client06        611258427.925015 830235942.528565 116048965119093968.000000 34065
9603.004369 908119137.647008 925259719.800855      606534738.337042 738698732.786
013  65939262524008544.000000 256786414.212295 809726223.077746 887431534.139117
0.046761 0.009945 0.003196 0.056537 0.118998 0.219145       0.031782 0.003858 0.0
04665 0.068298 0.118525 0.218559             0.008322 0.007171 0.000103 0.010173
0.018338 0.046484
client07        629741956.239159 834419015.562871 110436788198053280.000000 33232
0309.638236 907537005.695471 924405967.421665      491236755.776817 550829432.414
435  37130477377765208.000000 192692701.931768 694700535.272710 815092466.442601
0.046287 0.013000 0.003320 0.057620 0.122670 0.223010       0.034640 0.008962 0.0
02666 0.051629 0.122365 0.223122             0.005297 0.000876 0.000243 0.015588
0.009957 0.074133
client08        233889619.760791  96396453.326783  85464413648437376.000000 29234
2972.633921 858293350.571535 911931928.764855      225898541.234693  97324495.512
623  62380421665388528.000000 249760728.829391 710377845.517797 875723355.819327
0.054419 0.010000 0.004539 0.067369 0.120000 0.221000       0.040777 0.005875 0.0
03861 0.062137 0.120000 0.220907             0.050610 0.033348 0.003696 0.060792
0.184734 0.208623
client09        645240035.020388 894133515.436445 123273626430453536.000000 35110
```

```
3441.211352 918929603.781555 932601247.726708      452650070.838853 190206886.297
858 151097784422706496.000000 388712984.633529 907330424.748211 916297833.081366
0.044974 0.010000 0.003012 0.054878 0.119566 0.218591       0.033937 0.004345 0.0
02561 0.050606 0.119106 0.123821             0.004833 0.000731 0.000192 0.013848
0.007198 0.076008
client10        231791294.741378 221136216.323006  27568478260461136.000000 16603
7580.867890 416490915.978060 423380880.431440       37347812.266017  40764883.493
588     85902362107068.109375   9268352.718098  45588955.369329  46334067.053067
0.087202 0.015269 0.010696 0.103423 0.230650 0.246041       0.067731 0.016558 0.0
08788 0.093744 0.233527 0.247175             0.014803 0.000075 0.000828 0.028779
0.062928 0.108921
client11        504634766.307119 690643959.055207  83229269362668816.000000 28849
4834.204477 795386637.437721 800083278.007929      267394106.148097 382950945.283
751  27269385952482224.000000 165134448.109661 418338494.278168 423686403.918692
0.012883 0.012963 0.000039 0.006243 0.016000 0.021115       0.015223 0.015494 0.0
00083 0.009137 0.017775 0.034509             0.011972 0.005410 0.000379 0.019457
0.032570 0.090839
client12        464765894.805978 636004142.289032  81154196397532784.000000 28487
5756.071893 718451389.217889 788995519.569902       92171175.646829  96038521.060
637    925556715130361.000000  30422963.615177 125811313.387664 159694743.034384
0.086485 0.014583 0.010782 0.103834 0.229861 0.243094       0.073462 0.017329 0.0
09467 0.097298 0.233845 0.248905             0.000638 0.000061 0.000019 0.004304
0.001188 0.002751
client13        593933162.983242 596143878.076168    485087734388373.000000  2202
4707.362151 598633056.169719 600701979.420838      294764544.291732 298961298.142
188    275055997989540.437500  16584812.268746 301402671.503512 302627987.166911
0.011748 0.011926 0.000032 0.005656 0.015000 0.020232       0.012986 0.014284 0.0
00025 0.004995 0.015339 0.023667             0.053361 0.055479 0.000147 0.012125
0.066985 0.074303
```

5. Análise Agrupada por Servidor

Cálculo das estatísticas descritivas para cada variável, agrupadas por `server`.

In [41]:

```python
try:
    stats_by_server = df.groupby(group_by_server_col)[variables_to_analyze].agg(

    stats_by_server.rename(columns={'std': 'std_dev'}, level=1, inplace=True)

    print("--- Estatísticas por Servidor ---")
    print(stats_by_server)

except KeyError as e:
    print(f"ERRO: Chave não encontrada. Verifique os nomes das colunas. Detalhe:
except Exception as e:
    print(f"Ocorreu um erro: {e}")
```

```
--- Estatísticas por Servidor ---
         download_throughput_bps
upload_throughput_bps
rtt_download_sec                                                rtt_upload_sec
packet_loss_fraction
                              mean           median                         var
std_dev            q_0_9           q_0_99                   mean           median
var          std_dev            q_0_9           q_0_99             mean   median
var  std_dev    q_0_9   q_0_99           mean   median      var  std_dev    q_0_9
q_0_99                 mean   median      var  std_dev    q_0_9   q_0_99
server
server01        642228648.607059 761755122.155469  72231597992434752.000000 26875
9368.194738 903470691.456320 917939105.018732      437093211.641941 421465715.003
416  96764437318833344.000000 311069827.078798 895541991.225539 910093017.826796
0.011294 0.011000 0.000005 0.002282 0.014420 0.017562       0.012636 0.011140 0.0
00016 0.004020 0.018974 0.023730             0.022572 0.010169 0.000785 0.028025
0.064347 0.112422
server02        634859985.978759 761778714.444177  78134933215866208.000000 27952
6265.699426 901871539.827314 918578123.596323      440586101.799423 431405557.966
140 110285548275489072.000000 332092680.249790 896882546.243272 907728819.418116
0.011379 0.011000 0.000007 0.002640 0.015000 0.018632       0.011302 0.011246 0.0
01800 0.042429 0.020144 0.024678             0.019779 0.007935 0.000814 0.028524
0.062737 0.116649
server03        638431670.568167 768343383.824962  77658615016254320.000000 27867
2953.506892 903861585.675609 920161548.063678      448242233.599778 451789454.162
779 106400247162422768.000000 326190507.468293 899317157.222170 912003741.580556
0.011467 0.011000 0.000006 0.002470 0.015000 0.018040       0.012940 0.011203 0.0
00019 0.004312 0.020076 0.025741             0.021300 0.008350 0.000856 0.029253
0.064564 0.115284
server04        196583088.649712 168405038.276345  31454873225399988.000000 17735
5217.643575 596264507.173068 783898745.041256      157250705.342928 181915545.322
504   8205735459811588.000000  90585514.624644 297797770.063109 413928874.733900
0.127587 0.118385 0.004460 0.066782 0.223872 0.243328       0.129122 0.118861 0.0
04596 0.067795 0.231352 0.243247             0.019390 0.000000 0.001578 0.039725
0.063533 0.189949
server05        201203719.542334 168824051.192367  33504804349986220.000000 18304
3176.190718 596666984.243595 730367946.224861      158269476.908089 181156345.899
693   8750847773749336.000000  93545966.100893 299987138.985095 415662758.691359
0.123079 0.118946 0.004574 0.067632 0.223493 0.241111       0.124031 0.118818 0.0
04738 0.068835 0.228080 0.248317             0.027471 0.000000 0.002687 0.051833
0.108583 0.203460
server06        205373287.400451 169563776.313014  36368630144084000.000000 19070
5611.202408 596838976.922219 784010274.925666      157433421.560520 181308911.311
875   9821520408432576.000000  99103584.236054 298902057.740246 420558193.521076
0.129429 0.118838 0.005038 0.070981 0.227665 0.242656       0.128835 0.119000 0.0
07510 0.086659 0.232358 0.247993             0.019486 0.000000 0.001796 0.042378
0.064155 0.193470
server07        585219496.881597 709204403.196444 107983455525876368.000000 32860
8361.923242 911096080.406616 929151574.579697      449144828.178512 513251777.894
228  92975205812926688.000000 304918359.258551 837713469.847893 913622337.084218
0.029614 0.009000 0.002880 0.053669 0.117000 0.229671       0.007644 0.005448 0.0
00026 0.005120 0.015242 0.019740             0.016653 0.000892 0.000970 0.031141
0.058373 0.134794
```

Fase 1 (Cont.): Seleção de Entidades e Análise Gráfica

Com base nas tabelas de estatísticas da Etapa 1, selecionamos duas entidades com comportamentos distintos para a análise: `client12` e `client13`.

- **client12**: RTT alto (lento), mas perda de pacotes quase zero (confiável).
- **client13**: RTT baixo (rápido), mas a maior perda de pacotes (não confiável).

A célula seguinte filtra o dataframe principal para conter apenas estas duas entidades.

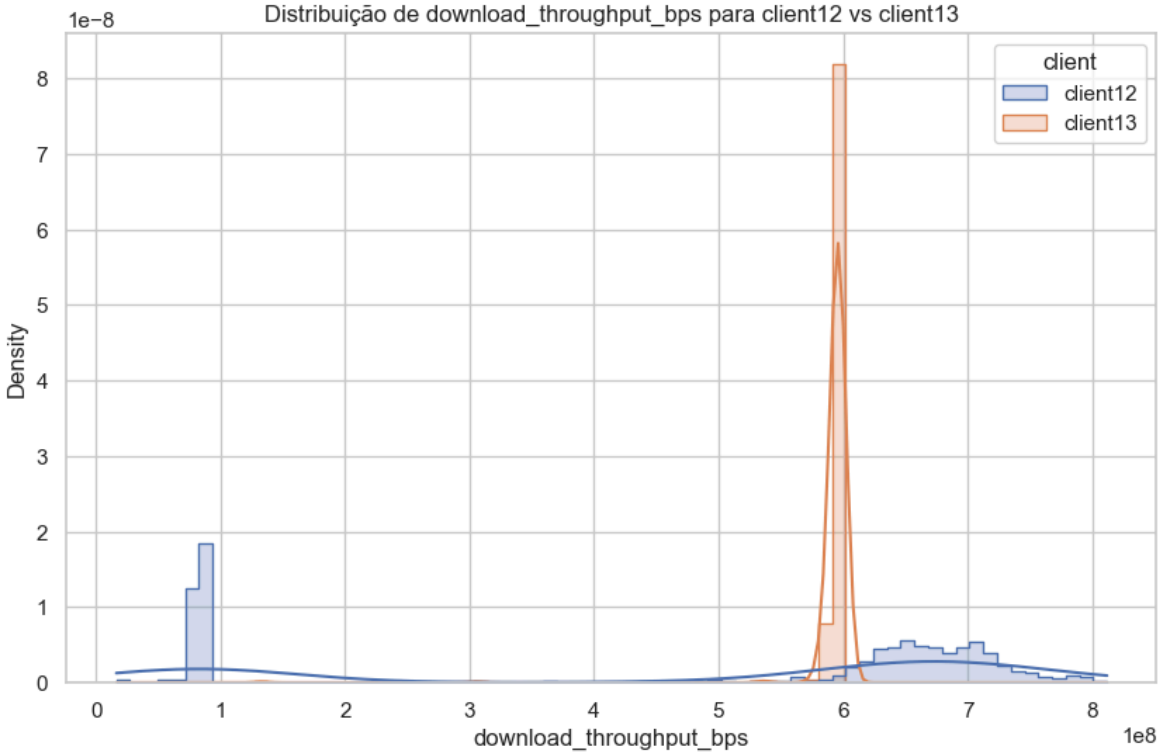
In [25]:
```
import matplotlib.pyplot as plt
import seaborn as sns

selected_entities = ['client12', 'client13']
entity_column = 'client'

df_selected = df[df[entity_column].isin(selected_entities)].copy()

print(f"DataFrame filtrado para as entidades: {selected_entities}")
print(df_selected.shape)
```

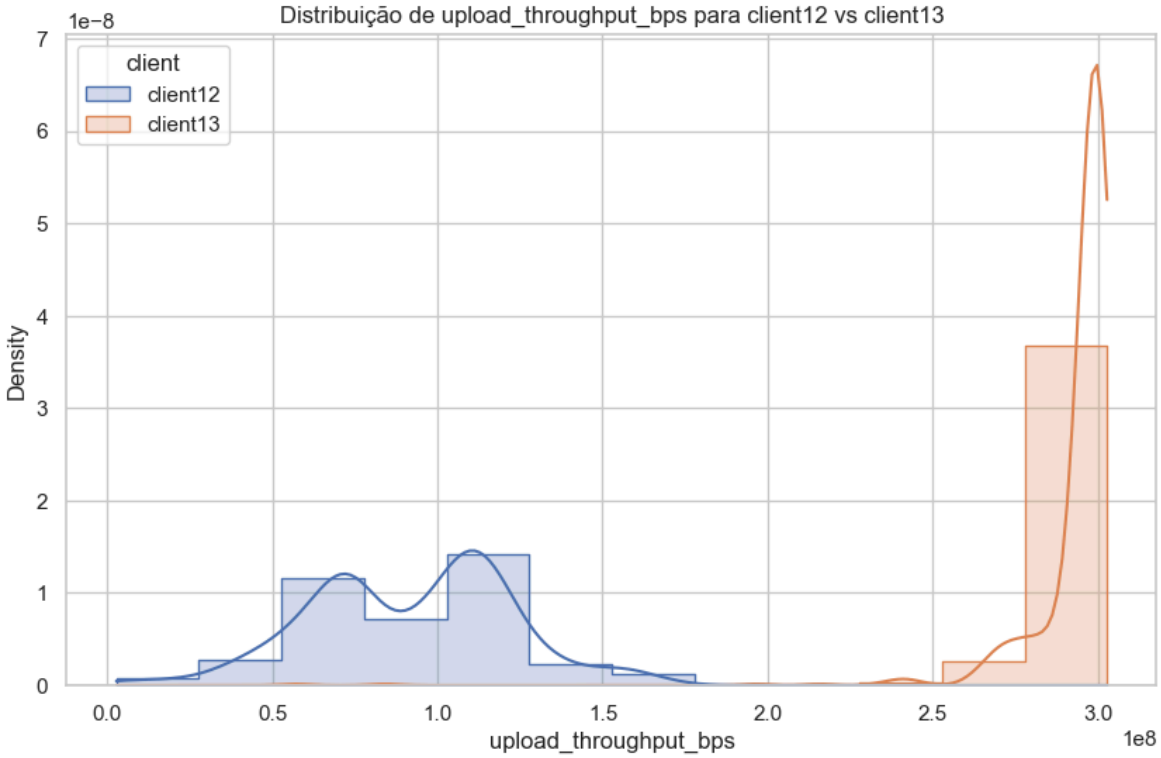
```
DataFrame filtrado para as entidades: ['client12', 'client13']
(1284, 9)
```

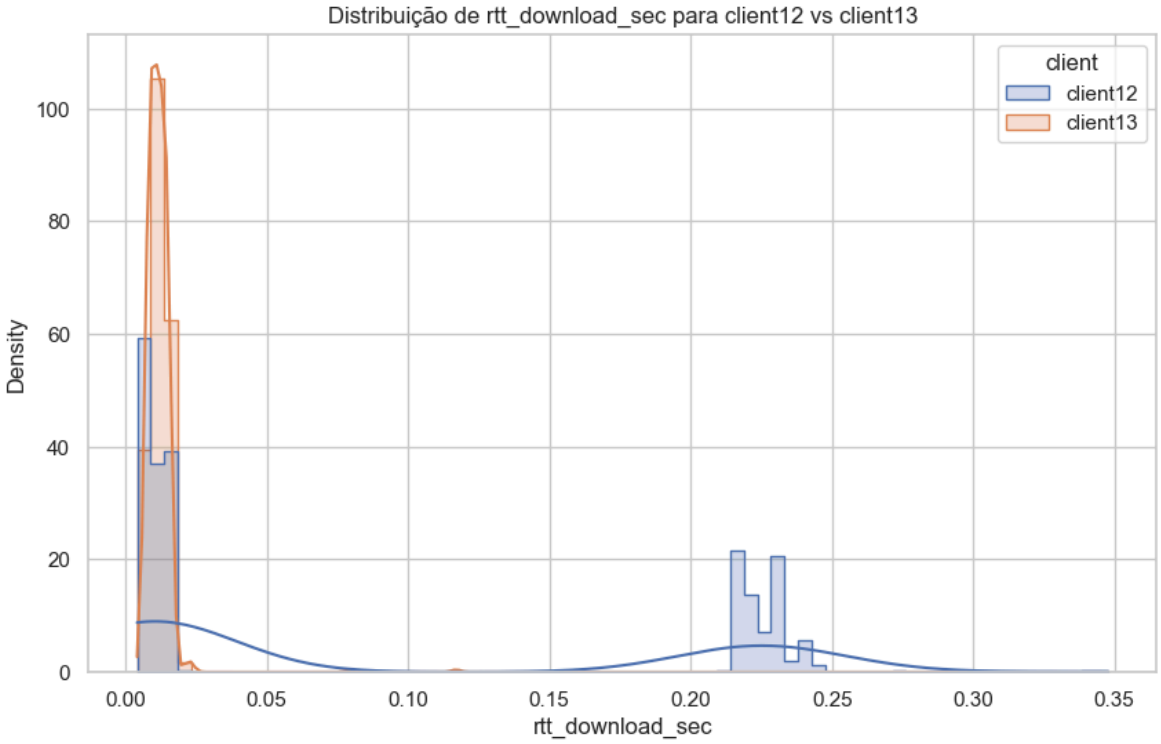
2.1 Histogramas (Distribuição)

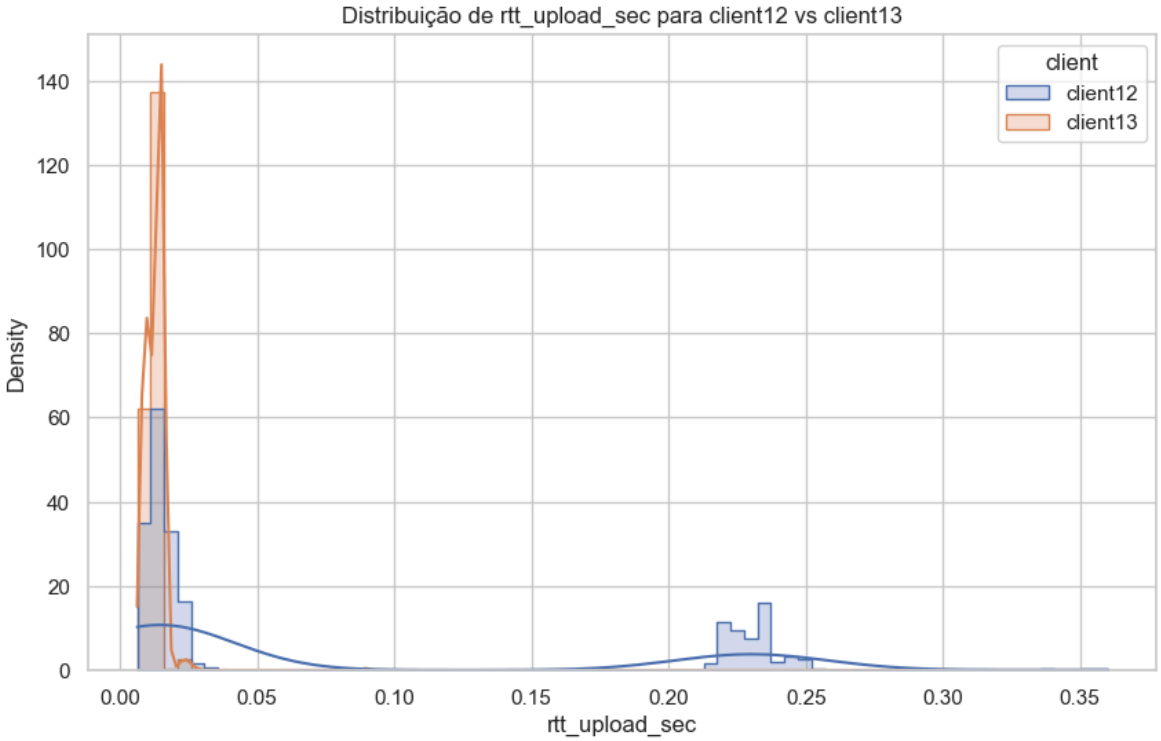
Análise da forma da distribuição de cada variável, comparando os dois clientes.

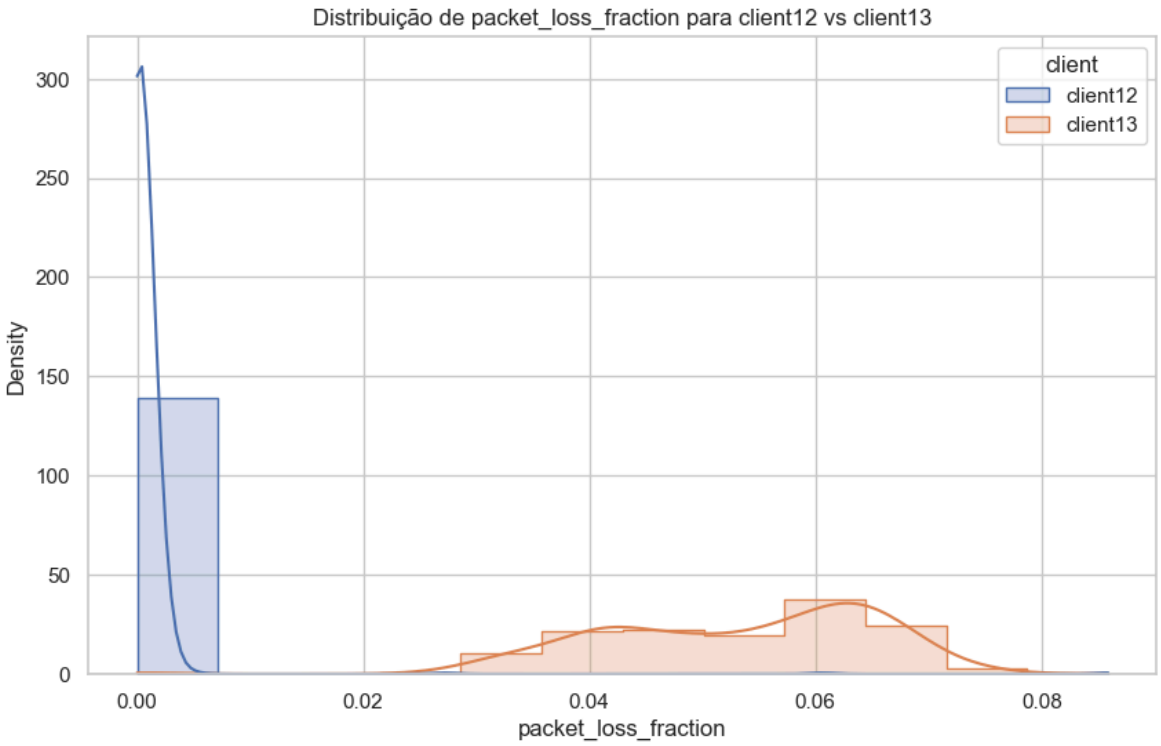
In [26]:
```
sns.set_theme(style="whitegrid")

for var in variables_to_analyze:
    plt.figure(figsize=(10, 6))
    sns.histplot(
        data=df_selected,
        x=var,
        hue=entity_column,
        kde=True,
        element="step",
        stat="density",
        common_norm=False
    )
    plt.title(f'Distribuição de {var} para {selected_entities[0]} vs {selected_e
    plt.show()
```


Distribuição de download_throughput_bps para client12 vs client13
1e−8
Density
download_throughput_bps
1e8
client
client12
client13


Distribuição de upload_throughput_bps para client12 vs client13
1e−8
Density
upload_throughput_bps
1e8
client
client12
client13


Distribuição de rtt_download_sec para client12 vs client13
client
client12
client13
Density
0
20
40
60
80
100
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
rtt_download_sec

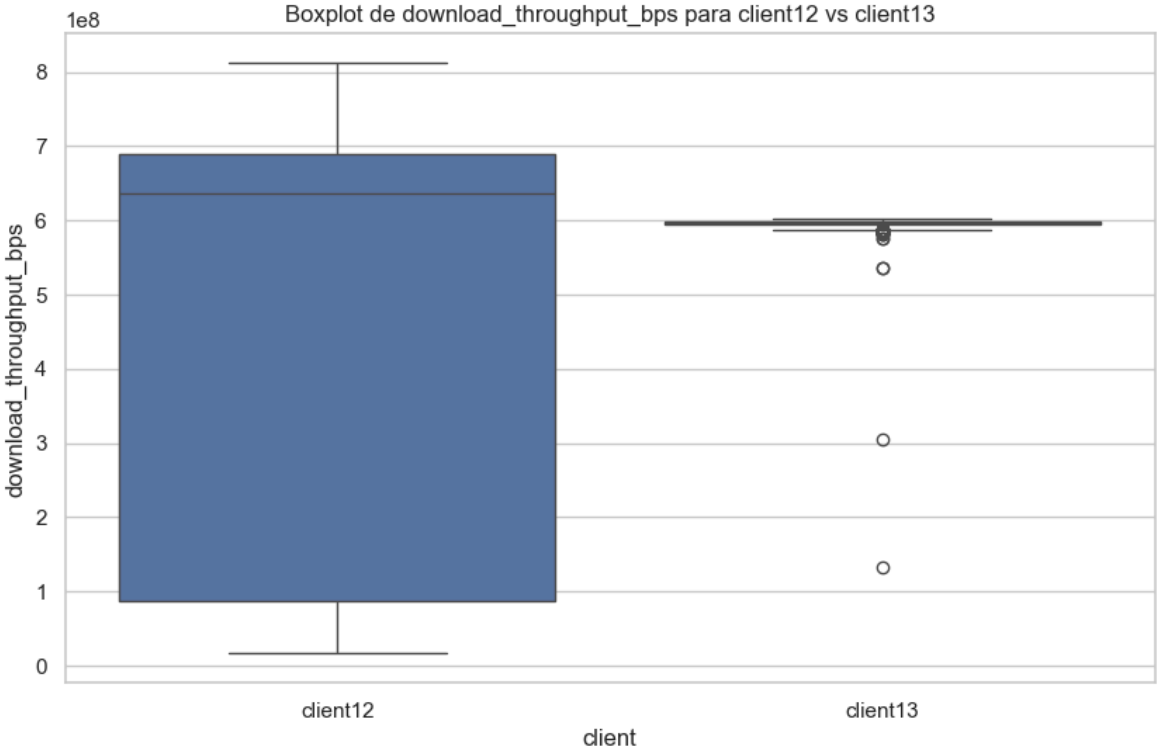

Distribuição de rtt_upload_sec para client12 vs client13
client
client12
client13
Density
0
20
40
60
80
100
120
140
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
rtt_upload_sec

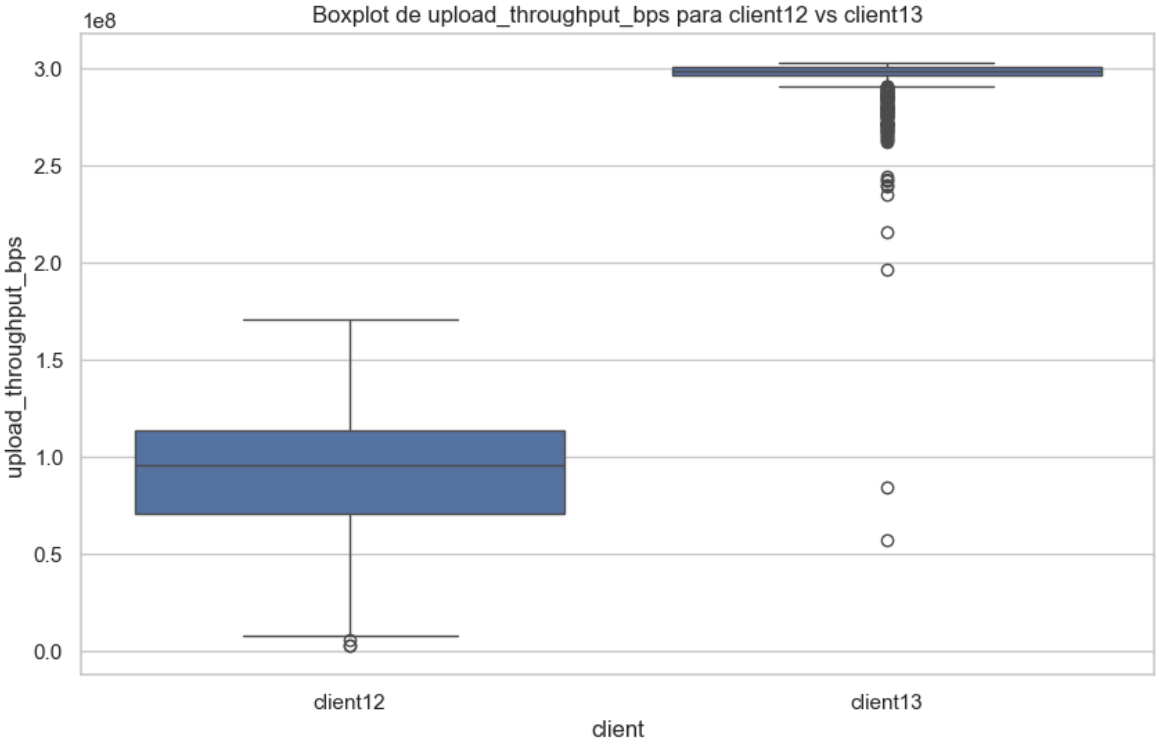


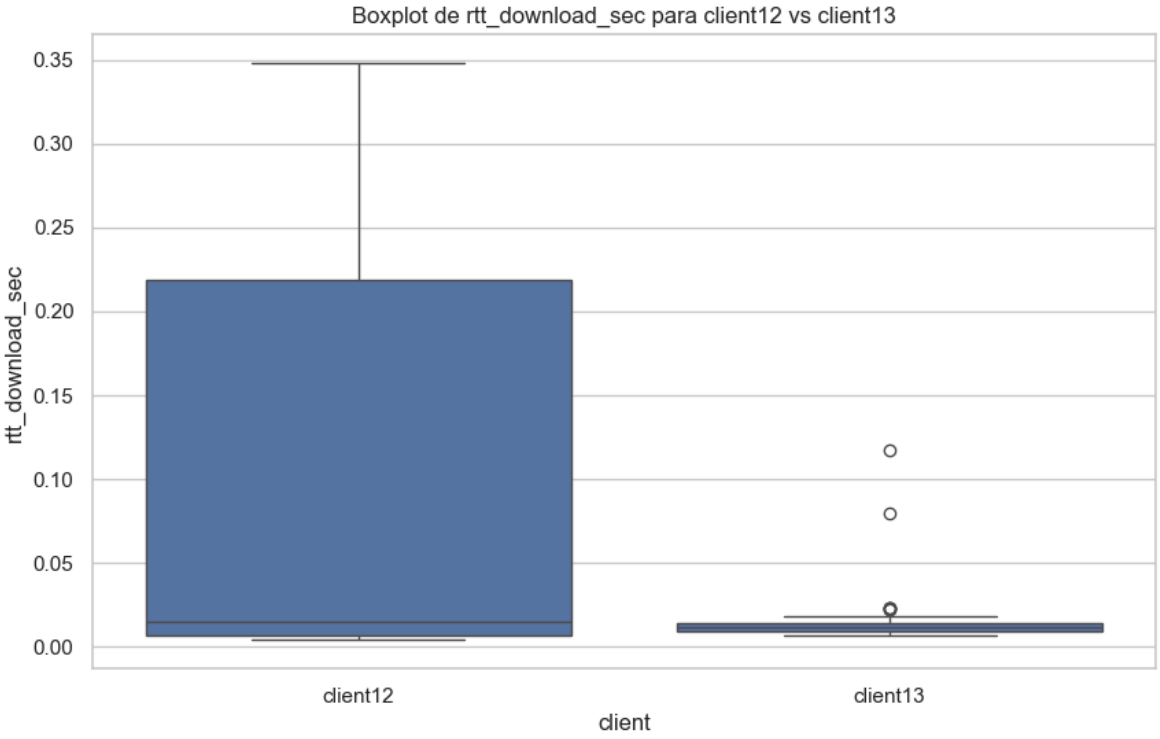
2.2 Boxplots (Outliers e Quartis)

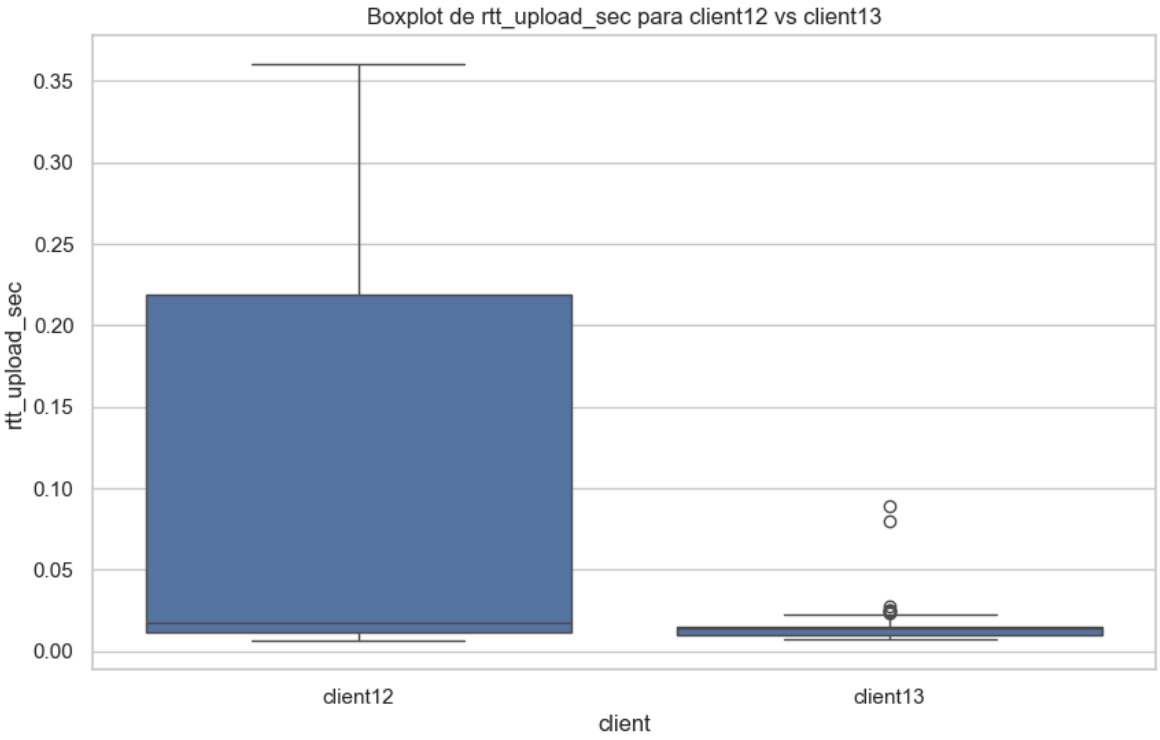
Os boxplots são úteis para comparar os quartis e identificar visualmente a presença de outliers em cada variável.

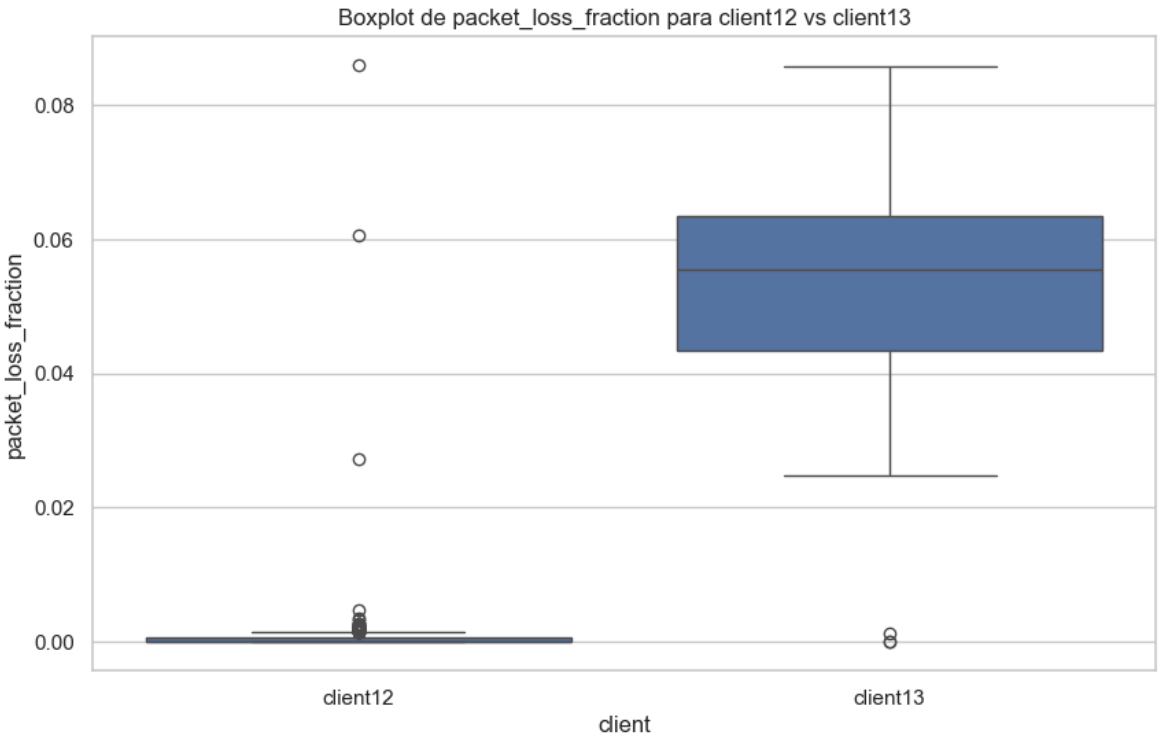
In [27]:
```
for var in variables_to_analyze:
    plt.figure(figsize=(10, 6))
    sns.boxplot(
        data=df_selected,
        x=entity_column,
        y=var
    )
    plt.title(f'Boxplot de {var} para {selected_entities[0]} vs {selected_entiti
    plt.show()
```


Boxplot de download_throughput_bps para client12 vs client13
1e8
download_throughput_bps
client
client12
client13


Boxplot de upload_throughput_bps para client12 vs client13
1e8
upload_throughput_bps
client
client12
client13


Boxplot de rtt_download_sec para client12 vs client13
rtt_download_sec
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
client12
client13
client

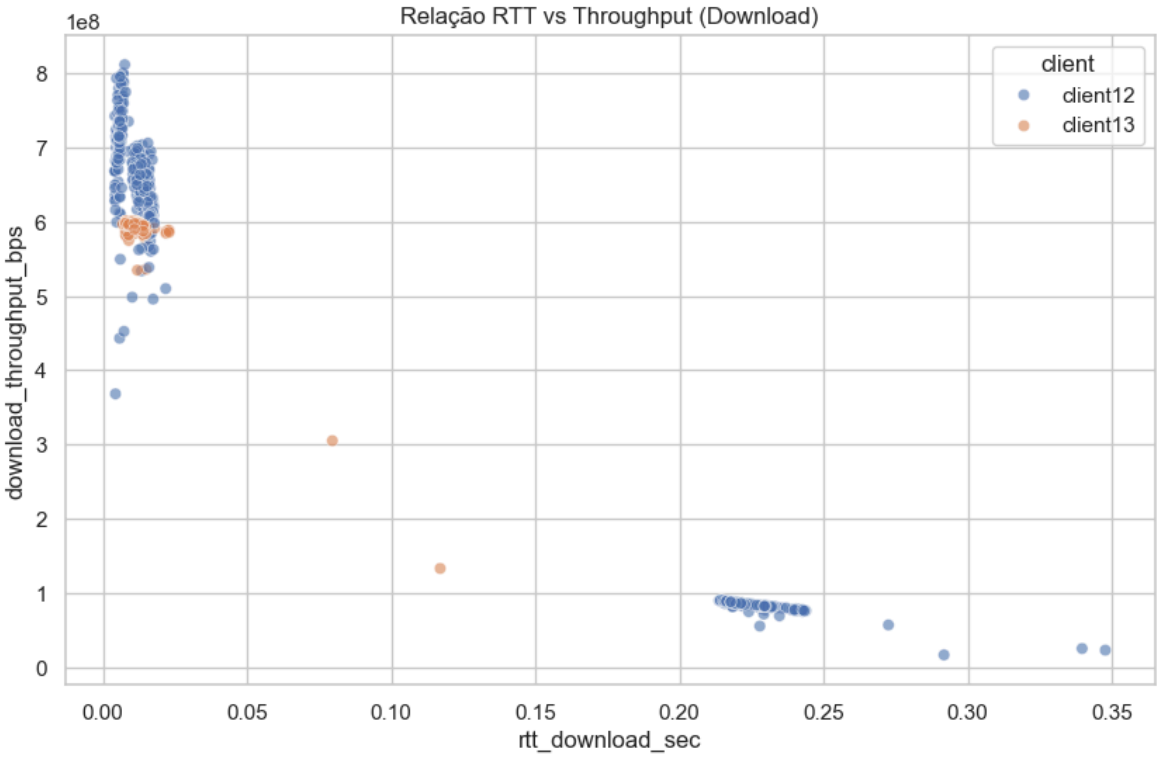

Boxplot de rtt_upload_sec para client12 vs client13
rtt_upload_sec
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
client12
client13
client



2.3 Scatter Plot (Correlação)

Vamos analisar a relação entre o RTT de download e o Throughput de download para os dois clientes.

In [28]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=df_selected,
    x='rtt_download_sec',
    y='download_throughput_bps',
    hue=entity_column,
    alpha=0.6
)
plt.title('Relação RTT vs Throughput (Download)')
plt.show()
```



Fase 2: Definição dos Modelos Candidatos

Com base na análise gráfica (Etapa 2) e nas sugestões da Seção 4 do documento do projeto, definimos os seguintes modelos paramétricos candidatos para as variáveis:

1. **RTT (up e down)**: A distribuição é assimétrica, mas o documento sugere usar a **Distribuição Normal** (Seção 4.1), provavelmente pela facilidade da prior conjugada Normal-Normal.
2. **Throughput (up e down)**: Os dados são estritamente positivos e assimétricos à direita. O modelo sugerido e validado pelos histogramas é a **Distribuição Gama** (Seção 4.3).
3. **Fração de Perda**: É uma proporção (0-1). O modelo sugerido é o **Modelo Beta-Binomial** (Seção 4.2), usando a Binomial para a contagem de pacotes perdidos.

Fase 2 (Cont.): Estimativa por Máxima Verossimilhança (MLE)

O objetivo desta seção é usar o método da Máxima Verossimilhança (MLE) para estimar os parâmetros ($\hat{\theta}_{MLE}$) dos modelos definidos acima.

Os parâmetros $\hat{\sigma}^2$ (RTT) e $\hat{k}$ (Throughput) aqui calculados serão usados como valores "conhecidos" na Fase Bayesiana, conforme instruído no PDF.

In [46]:
```
from scipy.stats import norm, gamma
import warnings

warnings.filterwarnings('ignore', category=RuntimeWarning)
```

```
df_c12 = df_selected[df_selected['client'] == 'client12']
df_c13 = df_selected[df_selected['client'] == 'client13']

mle_params_c12 = {}
mle_params_c13 = {}

n_t_assumido = 1000

print(f"DataFrames separados para client12 ({df_c12.shape[0]} linhas) e client13
print(f"Assumindo n_t = {n_t_assumido} pacotes por teste para o modelo Binomial"
```

```
DataFrames separados para client12 (640 linhas) e client13 (644 linhas)
Assumindo n_t = 1000 pacotes por teste para o modelo Binomial
```

4.1 Cálculo MLE para `client12`

In [110…

```
mle_params_c12 = {}

mu_rtt_d_c12, std_rtt_d_c12 = norm.fit(df_c12['rtt_download_sec'])
mle_params_c12['rtt_download_sec'] = {'mu': mu_rtt_d_c12, 'std': std_rtt_d_c12,

mu_rtt_u_c12, std_rtt_u_c12 = norm.fit(df_c12['rtt_upload_sec'])
mle_params_c12['rtt_upload_sec'] = {'mu': mu_rtt_u_c12, 'std': std_rtt_u_c12, 'v

shape_params_d_c12, loc_d_c12, scale_d_c12 = gamma.fit(df_c12['download_throughp
mle_params_c12['download_throughput_bps'] = {
    'shape_k': shape_params_d_c12,
    'loc': loc_d_c12,
    'scale': scale_d_c12,
    'rate_beta': 1.0/scale_d_c12
}

shape_params_u_c12, loc_u_c12, scale_u_c12 = gamma.fit(df_c12['upload_throughput
mle_params_c12['upload_throughput_bps'] = {
    'shape_k': shape_params_u_c12,
    'loc': loc_u_c12,
    'scale': scale_u_c12,
    'rate_beta': 1.0/scale_u_c12
}

x_tot_c12 = np.sum(df_c12['packet_loss_fraction'] * n_t_assumido)
n_tot_c12 = len(df_c12) * n_t_assumido
p_mle_c12 = x_tot_c12 / n_tot_c12
mle_params_c12['packet_loss_fraction'] = {'p_mle': p_mle_c12, 'x_tot': x_tot_c12

print("Cálculos MLE para client12 concluídos (com 'k' como float).")
```

```
Cálculos MLE para client12 concluídos (com 'k' como float).
```

4.1.1 Apresentação dos Resultados (client12)

In [111…

```
df_mle_c12 = pd.DataFrame(mle_params_c12).T
print("--- Parâmetros MLE para client12 (Formatado) ---")
print(df_mle_c12)
```

```
--- Parâmetros MLE para client12 (Formatado) ---
                              mu      std      var  shape_k      loc            s
cale  rate_beta    p_mle      x_tot         n_tot
rtt_download_sec         0.086485 0.103753 0.010765      NaN      NaN
NaN        NaN      NaN        NaN           NaN
rtt_upload_sec           0.073462 0.097222 0.009452      NaN      NaN
NaN        NaN      NaN        NaN           NaN
download_throughput_bps      NaN      NaN      NaN 1.484816 0.000000 313012352.60
8103   0.000000      NaN        NaN           NaN
upload_throughput_bps        NaN      NaN      NaN 6.758103 0.000000  13638616.90
4086   0.000000      NaN        NaN           NaN
packet_loss_fraction         NaN      NaN      NaN      NaN      NaN
NaN        NaN 0.000638 408.349586 640000.000000
```

4.2 Cálculo MLE para `client13`

In [112…

```
mle_params_c13 = {}

mu_rtt_d_c13, std_rtt_d_c13 = norm.fit(df_c13['rtt_download_sec'])
mle_params_c13['rtt_download_sec'] = {'mu': mu_rtt_d_c13, 'std': std_rtt_d_c13,

mu_rtt_u_c13, std_rtt_u_c13 = norm.fit(df_c13['rtt_upload_sec'])
mle_params_c13['rtt_upload_sec'] = {'mu': mu_rtt_u_c13, 'std': std_rtt_u_c13, 'v

shape_params_d_c13, loc_d_c13, scale_d_c13 = gamma.fit(df_c13['download_throughp
mle_params_c13['download_throughput_bps'] = {
    'shape_k': shape_params_d_c13,
    'loc': loc_d_c13,
    'scale': scale_d_c13,
    'rate_beta': 1.0/scale_d_c13
}

shape_params_u_c13, loc_u_c13, scale_u_c13 = gamma.fit(df_c13['upload_throughput
mle_params_c13['upload_throughput_bps'] = {
    'shape_k': shape_params_u_c13,
    'loc': loc_u_c13,
    'scale': scale_u_c13,
    'rate_beta': 1.0/scale_u_c13
}

x_tot_c13 = np.sum(df_c13['packet_loss_fraction'] * n_t_assumido)
n_tot_c13 = len(df_c13) * n_t_assumido
p_mle_c13 = x_tot_c13 / n_tot_c13
mle_params_c13['packet_loss_fraction'] = {'p_mle': p_mle_c13, 'x_tot': x_tot_c13

print("Cálculos MLE para client13 concluídos (com 'k' como float).")
```

```
Cálculos MLE para client13 concluídos (com 'k' como float).
```

4.2.1 Apresentação dos Resultados (client13)

In [113…

```
df_mle_c13 = pd.DataFrame(mle_params_c13).T
print("--- Parâmetros MLE para client13 (Formatado) ---")
print(df_mle_c13)
```

```
--- Parâmetros MLE para client13 (Formatado) ---
                              mu      std      var    shape_k      loc          s
cale  rate_beta    p_mle        x_tot         n_tot
rtt_download_sec         0.011748 0.005652 0.000032        NaN      NaN
NaN        NaN      NaN           NaN           NaN
rtt_upload_sec           0.012986 0.004991 0.000025        NaN      NaN
NaN        NaN      NaN           NaN           NaN
download_throughput_bps      NaN      NaN      NaN 348.781682 0.000000 1702879.46
3415   0.000001      NaN          NaN           NaN
upload_throughput_bps        NaN      NaN      NaN 173.134749 0.000000 1702515.21
7247   0.000001      NaN          NaN           NaN
packet_loss_fraction         NaN      NaN      NaN        NaN      NaN
NaN        NaN 0.053361 34364.480227 644000.000000
```

Fase 2 (Cont.): Avaliação do Ajuste (Fit) do MLE

Esta seção realiza o diagnóstico visual da qualidade do ajuste dos modelos. Para cada variável e cliente, são gerados:

1. Um gráfico do Histograma (dados reais) sobreposto com a PDF (função densidade) do modelo ajustado.
2. Um QQ-Plot para comparar os quantis dos dados reais com os quantis teóricos do modelo.

In [117…

```python
import statsmodels.api as sm
from scipy.stats import norm, gamma
import scipy.stats as stats

def plot_mle_fit(data, model_type, params_dict, client_name, var_name):
    plt.figure(figsize=(14, 6))

    ax1 = plt.subplot(1, 2, 1)
    sns.histplot(data, kde=False, stat="density", label="Dados Reais", element="

    xmin, xmax = ax1.get_xlim()
    x = np.linspace(xmin, xmax, 100)

    if model_type == 'norm':
        mu = params_dict['mu']
        std = params_dict['std']
        pdf = norm.pdf(x, mu, std)
        ax1.plot(x, pdf, 'r-', lw=2, label=f'PDF Normal (μ={mu:.3f}, σ={std:.3f}

        ax2 = plt.subplot(1, 2, 2)
        stats.probplot(data, dist='norm', sparams=(mu, std), plot=ax2)
        ax2.get_lines()[1].set_color('r')
        ax2.set_title(f'QQ Plot - {client_name} - {var_name}')

    elif model_type == 'gamma':
        k = params_dict['shape_k']
        loc = params_dict['loc']
        scale = params_dict['scale']
        pdf = gamma.pdf(x, a=k, loc=loc, scale=scale)
        ax1.plot(x, pdf, 'r-', lw=2, label=f'PDF Gama (k={k:.3f}, θ={scale:.3f})
```

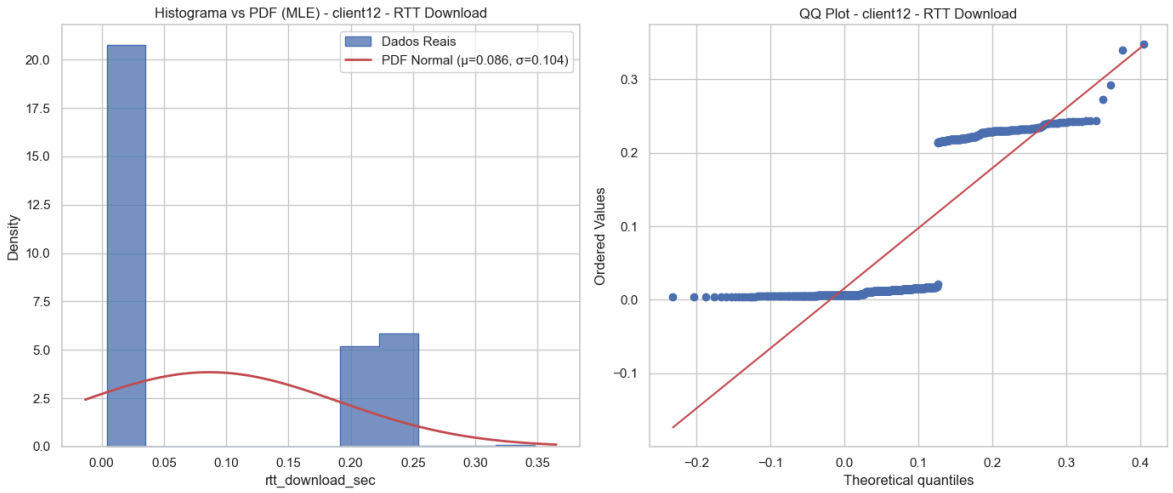
```
        ax2 = plt.subplot(1, 2, 2)
        stats.probplot(data, dist='gamma', sparams=(k, loc, scale), plot=ax2)
        ax2.get_lines()[1].set_color('r')
        ax2.set_title(f'QQ Plot - {client_name} - {var_name}')

    elif model_type == 'binom':
        ax1.set_title(f'Modelo Binomial (p={params_dict["p_mle"]:.6f}) - Gráfico
        return

    ax1.set_title(f'Histograma vs PDF (MLE) - {client_name} - {var_name}')
    ax1.legend()

    plt.tight_layout()
    plt.show()

print("Função de plotagem 'plot_mle_fit' (corrigida) definida.")
```

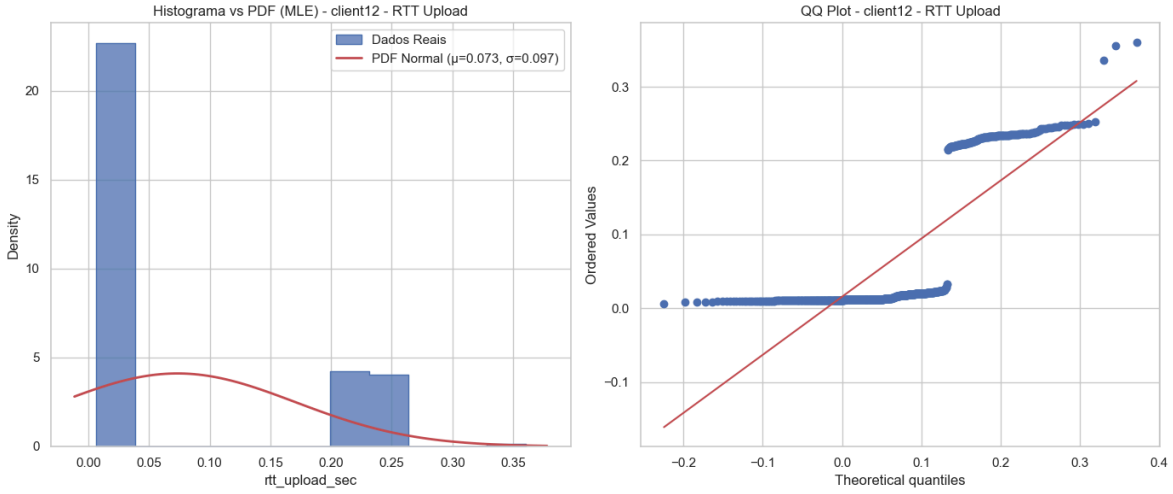
```
Função de plotagem 'plot_mle_fit' (corrigida) definida.
```

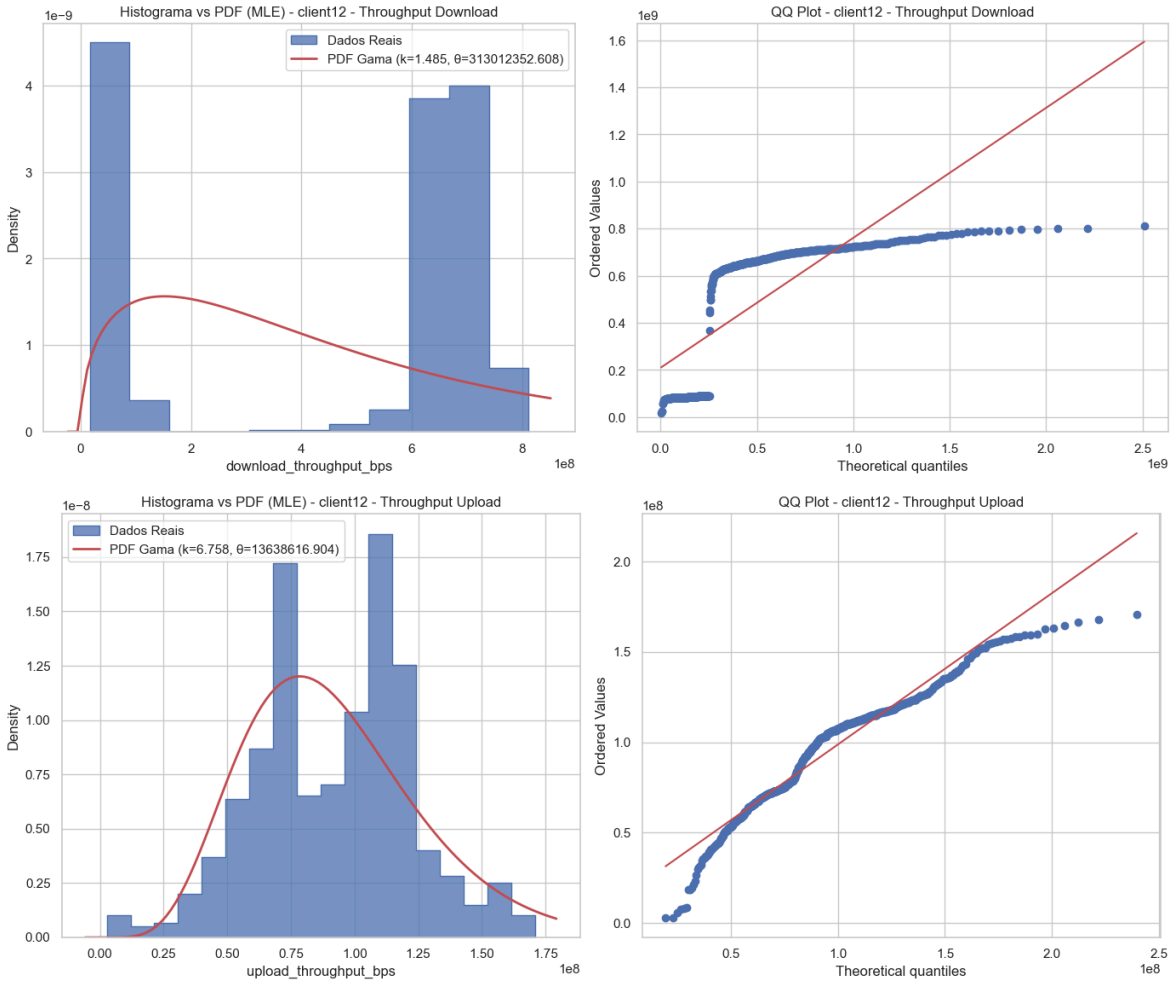
5.1 Gráficos de Ajuste - client12

In [118…

```
plot_mle_fit(df_c12['rtt_download_sec'], 'norm', mle_params_c12['rtt_download_se
plot_mle_fit(df_c12['rtt_upload_sec'], 'norm', mle_params_c12['rtt_upload_sec'],
plot_mle_fit(df_c12['download_throughput_bps'], 'gamma', mle_params_c12['downloa
plot_mle_fit(df_c12['upload_throughput_bps'], 'gamma', mle_params_c12['upload_th
```



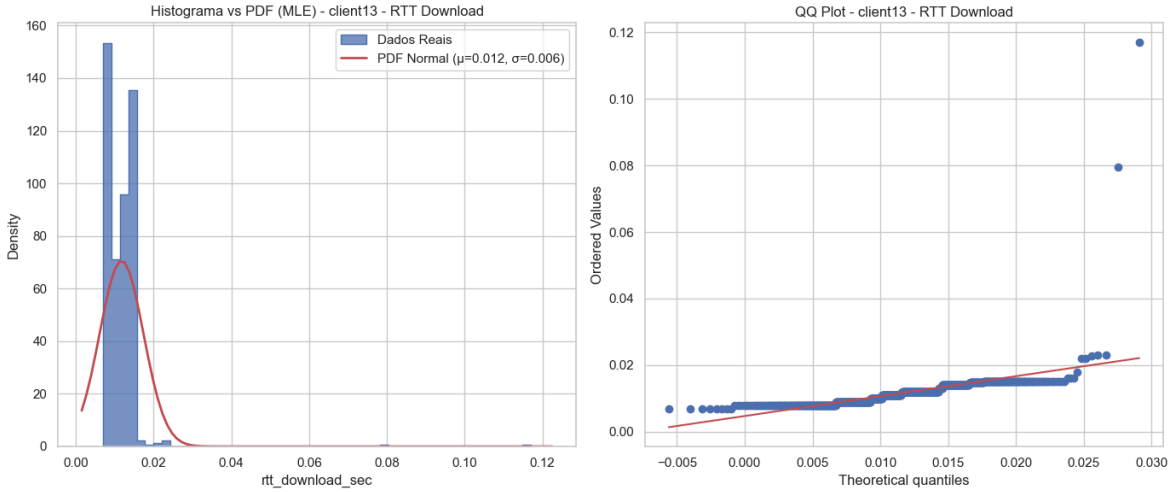


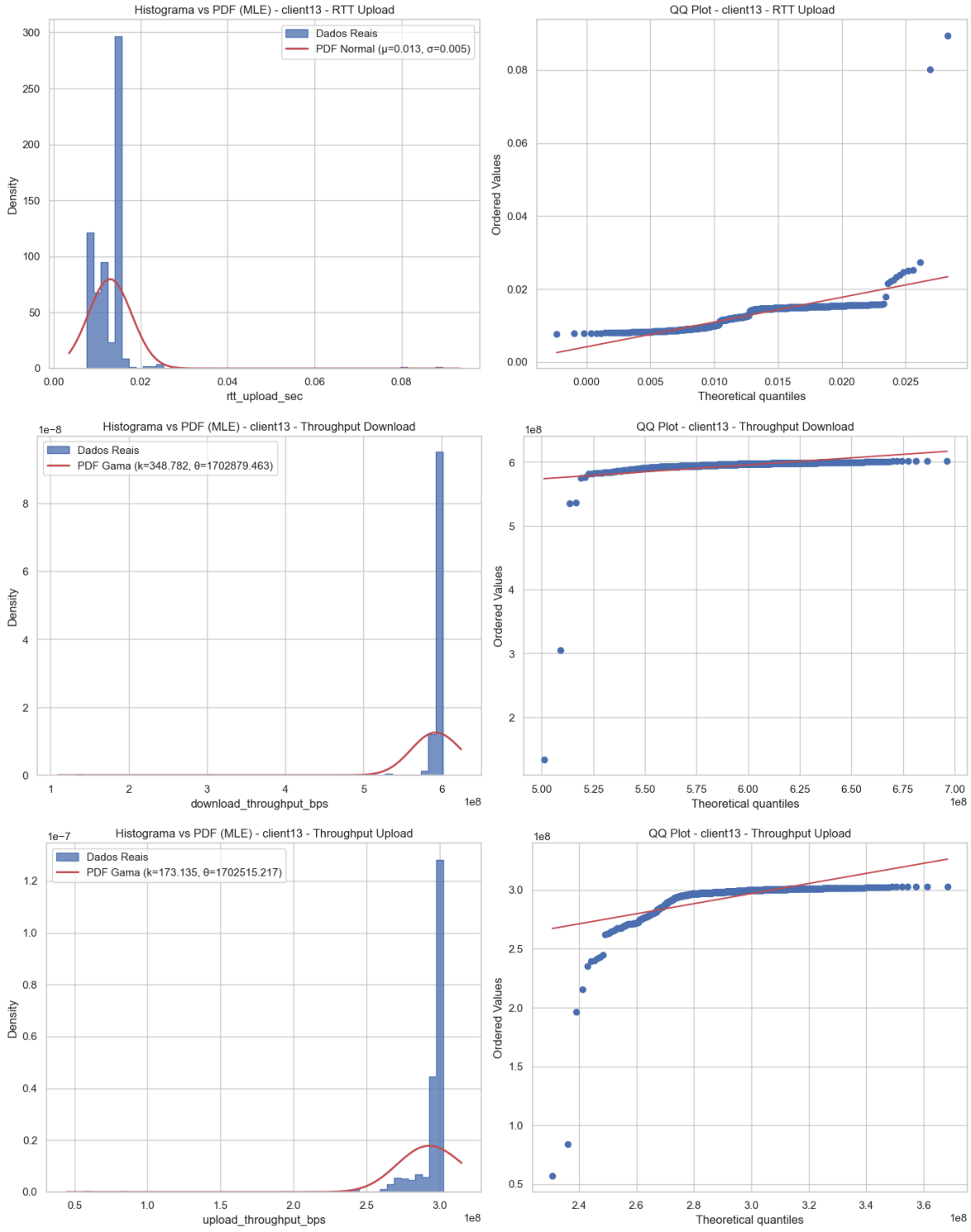


5.2 Gráficos de Ajuste - client13

In [119…

```
plot_mle_fit(df_c13['rtt_download_sec'], 'norm', mle_params_c13['rtt_download_se
plot_mle_fit(df_c13['rtt_upload_sec'], 'norm', mle_params_c13['rtt_upload_sec'],
plot_mle_fit(df_c13['download_throughput_bps'], 'gamma', mle_params_c13['downloa
plot_mle_fit(df_c13['upload_throughput_bps'], 'gamma', mle_params_c13['upload_th
```





Fase 3: Inferência Bayesiana

6. Split dos Dados (Treino/Teste)

Iniciando a Fase 3. O primeiro passo é dividir os dados dos clientes selecionados em conjuntos de treino (70%) e teste (30%), conforme solicitado na Seção 3.3 do PDF.

In [122…

```
from sklearn.model_selection import train_test_split

test_size_ratio = 0.3
random_seed = 42
```

```
df_c12_train, df_c12_test = train_test_split(df_c12, test_size=test_size_ratio,
df_c13_train, df_c13_test = train_test_split(df_c13, test_size=test_size_ratio,

print(f"Split para client12:")
print(f"  Treino: {df_c12_train.shape[0]} linhas")
print(f"  Teste:  {df_c12_test.shape[0]} linhas")
print(f"Split para client13:")
print(f"  Treino: {df_c13_train.shape[0]} linhas")
print(f"  Teste:  {df_c13_test.shape[0]} linhas")
```

```
Split para client12:
  Treino: 448 linhas
  Teste:  192 linhas
Split para client13:
  Treino: 450 linhas
  Teste:  194 linhas
```

6.1 Definição das Priors

A seguir, especificamos as *priors* (crenças iniciais) para os parâmetros de cada modelo. Seguindo o PDF (Seção 3.3), utilizamos priors "não informativas" ou "fracamente informativas".

- **RTT (Normal-Normal):**
  - `Likelihood` : $r_i|\mu \sim \mathcal{N}(\mu, \sigma^2)$ (Usaremos $\sigma^2 = \hat{\sigma}^2_{MLE}$).
  - `Prior` : $\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$.
  - **Justificativa:** Como prior fracamente informativa, definimos $\mu_0 = 0.0$ (uma crença inicial de RTT 0) mas com uma variância $\tau_0^2 = 1.0$. Uma variância de 1.0 em *segundos* é **enorme** para RTT, significando que temos *baixa confiança* nessa prior, e os dados irão dominá-la facilmente.
- **Throughput (Gama-Gama):**
  - `Likelihood` : $y_i|\beta \sim \text{Gamma}(k, \beta)$ (Usaremos $k = \hat{k}_{MLE}$).
  - `Prior` : $\beta \sim \text{Gamma}(a_0, b_0)$.
  - **Justificativa:** Como prior fracamente informativa, usamos $a_0 = 1.0$ e $b_0 = 0.000001$. Valores de a_0 pequenos e b_0 muito próximos de zero criam uma distribuição Gama "espalhada", que se aproxima de uma prior imprópria ($1/\beta$) e não influencia significativamente a posterior.
- **Perda (Beta-Binomial):**
  - `Likelihood` : $X_t|p \sim \text{Binomial}(n_t, p)$.
  - `Prior` : $p \sim \text{Beta}(a_0, b_0)$.
  - **Justificativa:** Usaremos a **Prior de Jeffreys** para a distribuição Beta, que é uma prior não informativa padrão para proporções. Ela é definida como $a_0 = 0.5$ e $b_0 = 0.5$.

In [123…

```
bayesian_params_c12 = {}
bayesian_params_c13 = {}
bayesian_test_stats_c12 = {}
bayesian_test_stats_c13 = {}
```

```
print("Dicionários para parâmetros Bayesianos e estatísticas de teste criados.")
```

```
Dicionários para parâmetros Bayesianos e estatísticas de teste criados.
```

Fase 3 (Cont.): Cálculo da Posterior

7. Cálculo dos Parâmetros da Posterior

Com os dados de **treino** e as *priors* definidas, a célula seguinte calcula os parâmetros das distribuições a posteriori, seguindo as equações de conjugação do PDF (Seção 4).

In [128…

```
# Definição das Priors (Justificadas conforme Célula 48 e PDF Seção 3.3)

# 1. RTT (Normal-Normal): Prior Fracamente Informativa
# mu_0 = 0.0: Crença inicial de RTT 0
# tau_0_sq = 1.0: Variância de 1.0 (em segundos) é enorme para RTT,
# indicando baixa confiança na prior. Os dados vão dominar.
prior_rtt_mu_0 = 0.0
prior_rtt_tau_0_sq = 1.0

# 2. Throughput (Gama-Gama): Prior Fracamente Informativa
# a_0 = 1.0, b_0 = 0.000001: Valores pequenos (b_0 próximo de zero)
# criam uma prior "espalhada", aproximando uma prior não informativa 1/beta.
prior_thru_a_0 = 1.0
prior_thru_b_0 = 0.000001

# 3. Perda (Beta-Binomial): Prior de Jeffreys (Não Informativa)
# a_0 = 0.5, b_0 = 0.5: Esta é a Prior de Jeffreys, uma prior
# não informativa padrão para proporções (parâmetro p da Binomial)[cite: 39].
prior_loss_a_0 = 0.5
prior_loss_b_0 = 0.5

print("Priors (justificadas) definidas.")
```

```
Priors (justificadas) definidas.
```

7.1 Cálculo da Posterior - `client12`

In [125…

```
bayesian_params_c12 = {}

# 1. RTT (Normal-Normal)
for var_name in ['rtt_download_sec', 'rtt_upload_sec']:
    sigma_2_mle = mle_params_c12[var_name]['var']
    data_train = df_c12_train[var_name]
    n = len(data_train)
    r_bar = data_train.mean()

    tau_n_2 = (1.0 / prior_rtt_tau_0_sq + n / sigma_2_mle)**(-1)
    mu_n = tau_n_2 * (prior_rtt_mu_0 / prior_rtt_tau_0_sq + (n * r_bar) / sigma_

    bayesian_params_c12[var_name] = {'post_mu_n': mu_n, 'post_tau_n_sq': tau_n_2

# 2. Throughput (Gama-Gama)
for var_name in ['download_throughput_bps', 'upload_throughput_bps']:
```

```
    k_mle = mle_params_c12[var_name]['shape_k']
    data_train = df_c12_train[var_name]
    n = len(data_train)
    sum_y_i = data_train.sum()

    a_n = prior_thru_a_0 + n * k_mle
    b_n = prior_thru_b_0 + sum_y_i

    bayesian_params_c12[var_name] = {'post_a_n': a_n, 'post_b_n': b_n, 'post_mea

# 3. Perda (Beta-Binomial)
var_name = 'packet_loss_fraction'
data_train = df_c12_train[var_name]
n_obs = len(data_train)
x_tot_train = np.sum(data_train * n_t_assumido)
n_tot_train = n_obs * n_t_assumido

a_n = prior_loss_a_0 + x_tot_train
b_n = prior_loss_b_0 + (n_tot_train - x_tot_train)

bayesian_params_c12[var_name] = {'post_a_n': a_n, 'post_b_n': b_n, 'post_mean_p'


print("Parâmetros da posterior para client12 calculados.")
```

```
Parâmetros da posterior para client12 calculados.
```

7.2 Cálculo da Posterior - `client13`

In [126…

```
bayesian_params_c13 = {}

# 1. RTT (Normal-Normal)
for var_name in ['rtt_download_sec', 'rtt_upload_sec']:
    sigma_2_mle = mle_params_c13[var_name]['var']
    data_train = df_c13_train[var_name]
    n = len(data_train)
    r_bar = data_train.mean()

    tau_n_2 = (1.0 / prior_rtt_tau_0_sq + n / sigma_2_mle)**(-1)
    mu_n = tau_n_2 * (prior_rtt_mu_0 / prior_rtt_tau_0_sq + (n * r_bar) / sigma_

    bayesian_params_c13[var_name] = {'post_mu_n': mu_n, 'post_tau_n_sq': tau_n_2

# 2. Throughput (Gama-Gama)
for var_name in ['download_throughput_bps', 'upload_throughput_bps']:
    k_mle = mle_params_c13[var_name]['shape_k']
    data_train = df_c13_train[var_name]
    n = len(data_train)
    sum_y_i = data_train.sum()

    a_n = prior_thru_a_0 + n * k_mle
    b_n = prior_thru_b_0 + sum_y_i

    bayesian_params_c13[var_name] = {'post_a_n': a_n, 'post_b_n': b_n, 'post_mea

# 3. Perda (Beta-Binomial)
var_name = 'packet_loss_fraction'
data_train = df_c13_train[var_name]
```

```
n_obs = len(data_train)
x_tot_train = np.sum(data_train * n_t_assumido)
n_tot_train = n_obs * n_t_assumido

a_n = prior_loss_a_0 + x_tot_train
b_n = prior_loss_b_0 + (n_tot_train - x_tot_train)

bayesian_params_c13[var_name] = {'post_a_n': a_n, 'post_b_n': b_n, 'post_mean_p'


print("Parâmetros da posterior para client13 calculados.")
```

```
Parâmetros da posterior para client13 calculados.
```

7.3 Apresentação dos Resultados da Posterior

Vamos formatar os parâmetros da posterior em tabelas para facilitar a leitura.

In [127…

```
df_post_c12 = pd.DataFrame(bayesian_params_c12).T
print("--- Parâmetros da Posterior (client12) ---")
print(df_post_c12)

print("\n")

df_post_c13 = pd.DataFrame(bayesian_params_c13).T
print("--- Parâmetros da Posterior (client13) ---")
print(df_post_c13)
```

```
--- Parâmetros da Posterior (client12) ---
                        post_mu_n  post_tau_n_sq    post_a_n            post_b_n
post_mean_beta  post_mean_p
rtt_download_sec          0.082785       0.000024         NaN                 NaN
NaN          NaN
rtt_upload_sec            0.069733       0.000021         NaN                 NaN
NaN          NaN
download_throughput_bps        NaN            NaN  666.197776 212753036089.055786
0.000000          NaN
upload_throughput_bps          NaN            NaN 3028.630073  42164411467.143051
0.000000          NaN
packet_loss_fraction           NaN            NaN  349.247157       447651.752843
NaN     0.000780


--- Parâmetros da Posterior (client13) ---
                        post_mu_n  post_tau_n_sq      post_a_n            post_b
_n  post_mean_beta  post_mean_p
rtt_download_sec          0.011647       0.000000           NaN                 N
aN             NaN          NaN
rtt_upload_sec            0.013139       0.000000           NaN                 N
aN             NaN          NaN
download_throughput_bps        NaN            NaN 156952.756765 267463955262.4746
40        0.000001          NaN
upload_throughput_bps          NaN            NaN  77911.636914 132585271026.6357
12        0.000001          NaN
packet_loss_fraction           NaN            NaN  24028.535746       425972.4642
54             NaN     0.053397
```

Fase 3 (Cont.): Cálculo Preditivo e Comparação

8. Cálculo da Posterior Preditiva

Esta seção utiliza as distribuições a posteriori (Etapa 7) para calcular a *distribuição preditiva posterior* para cada modelo.

O objetivo é comparar a **Média Preditiva** e a **Variância Preditiva** (previstas pelo modelo) com a **Média Real** e a **Variância Real** (observadas no conjunto de **teste**).

8.1 Cálculo das Estatísticas Preditivas - `client12`

In [129…

```
bayesian_predictive_c12 = {}

# 1. RTT (Normal-Normal) - Equações (6)
for var_name in ['rtt_download_sec', 'rtt_upload_sec']:
    sigma_2_mle = mle_params_c12[var_name]['var']
    mu_n = bayesian_params_c12[var_name]['post_mu_n']
    tau_n_2 = bayesian_params_c12[var_name]['post_tau_n_sq']

    pred_mean = mu_n
    pred_var = sigma_2_mle + tau_n_2

    bayesian_predictive_c12[var_name] = {'pred_mean': pred_mean, 'pred_var': pre

# 2. Throughput (Gama-Gama) - Equações (24) e (25)
for var_name in ['download_throughput_bps', 'upload_throughput_bps']:
    k_mle = mle_params_c12[var_name]['shape_k']
    a_n = bayesian_params_c12[var_name]['post_a_n']
    b_n = bayesian_params_c12[var_name]['post_b_n']

    pred_mean = (k_mle * b_n) / (a_n - 1)
    pred_var = (k_mle * (k_mle + a_n - 1) * b_n**2) / ((a_n - 1)**2 * (a_n - 2))

    bayesian_predictive_c12[var_name] = {'pred_mean': pred_mean, 'pred_var': pre

# 3. Perda (Beta-Binomial) - Equações (13), (14) e (15)
var_name = 'packet_loss_fraction'
a_n = bayesian_params_c12[var_name]['post_a_n']
b_n = bayesian_params_c12[var_name]['post_b_n']
n_star = n_t_assumido

pred_mean_frac = a_n / (a_n + b_n)

var_pred_contagem = n_star * (a_n * b_n * (a_n + b_n + n_star)) / ((a_n + b_n)**
pred_var_frac = var_pred_contagem / (n_star**2)

bayesian_predictive_c12[var_name] = {'pred_mean': pred_mean_frac, 'pred_var': pr


print("Estatísticas preditivas para client12 calculadas.")
```

```
Estatísticas preditivas para client12 calculadas.
```

8.2 Cálculo das Estatísticas Preditivas - `client13`

In [130…

```
bayesian_predictive_c13 = {}

# 1. RTT (Normal-Normal) - Equações (6)
for var_name in ['rtt_download_sec', 'rtt_upload_sec']:
    sigma_2_mle = mle_params_c13[var_name]['var']
    mu_n = bayesian_params_c13[var_name]['post_mu_n']
    tau_n_2 = bayesian_params_c13[var_name]['post_tau_n_sq']

    pred_mean = mu_n
    pred_var = sigma_2_mle + tau_n_2

    bayesian_predictive_c13[var_name] = {'pred_mean': pred_mean, 'pred_var': pre

# 2. Throughput (Gama-Gama) - Equações (24) e (25)
for var_name in ['download_throughput_bps', 'upload_throughput_bps']:
    k_mle = mle_params_c13[var_name]['shape_k']
    a_n = bayesian_params_c13[var_name]['post_a_n']
    b_n = bayesian_params_c13[var_name]['post_b_n']

    pred_mean = (k_mle * b_n) / (a_n - 1)
    pred_var = (k_mle * (k_mle + a_n - 1) * b_n**2) / ((a_n - 1)**2 * (a_n - 2))

    bayesian_predictive_c13[var_name] = {'pred_mean': pred_mean, 'pred_var': pre

# 3. Perda (Beta-Binomial) - Equações (13), (14) e (15)
var_name = 'packet_loss_fraction'
a_n = bayesian_params_c13[var_name]['post_a_n']
b_n = bayesian_params_c13[var_name]['post_b_n']
n_star = n_t_assumido

pred_mean_frac = a_n / (a_n + b_n)

var_pred_contagem = n_star * (a_n * b_n * (a_n + b_n + n_star)) / ((a_n + b_n)**
pred_var_frac = var_pred_contagem / (n_star**2)

bayesian_predictive_c13[var_name] = {'pred_mean': pred_mean_frac, 'pred_var': pr


print("Estatísticas preditivas para client13 calculadas.")
```

```
Estatísticas preditivas para client13 calculadas.
```

8.3 Comparação Preditivo vs. Real

Agora, vamos calcular as estatísticas reais do conjunto de teste (`df_c12_test` , `df_c13_test`) e colocá-las lado a lado com nossas previsões.

In [133…

```
def create_comparison_table(pred_dict, test_df):
    real_stats = test_df[variables_to_analyze].agg(['mean', 'var']).T
    real_stats.rename(columns={'mean': 'real_mean', 'var': 'real_var'}, inplace=

    pred_stats = pd.DataFrame(pred_dict).T
```

```
    comparison_df = pd.concat([pred_stats, real_stats], axis=1)

    comparison_df['perc_error_mean'] = (np.abs(comparison_df['pred_mean'] - comp

    comparison_df = comparison_df[['pred_mean', 'real_mean', 'perc_error_mean',

    return comparison_df

print("--- Comparação Preditivo vs. Real (client12) ---")
comparison_c12 = create_comparison_table(bayesian_predictive_c12, df_c12_test)
print(comparison_c12)

print("\n--- Comparação Preditivo vs. Real (client13) ---")
comparison_c13 = create_comparison_table(bayesian_predictive_c13, df_c13_test)
print(comparison_c13)
```

```
--- Comparação Preditivo vs. Real (client12) ---
                            pred_mean        real_mean  perc_error_mean
pred_var                 real_var
rtt_download_sec                0.082785         0.095115        12.964060
0.010789                 0.011072
rtt_upload_sec                  0.069733         0.082160        15.125163
0.009473                 0.010304
download_throughput_bps 474895169.841642 441130919.722762         7.654020 152455
967576292704.000000 85497236253618352.000000
upload_throughput_bps    94116989.882016  87630942.431393         7.401549   1314
083782819412.250000   931436297051405.375000
packet_loss_fraction            0.000780         0.000310       151.125702
0.000001                 0.000000

--- Comparação Preditivo vs. Real (client13) ---
                            pred_mean        real_mean  perc_error_mean
pred_var                real_var
rtt_download_sec                0.011647         0.011982         2.802054
0.000032                0.000064
rtt_upload_sec                  0.013139         0.012632         4.013719
0.000025                0.000008
download_throughput_bps 594364345.027721 592932998.447079         0.241401 101512
2901261373.250000 1114298570295187.125000
upload_throughput_bps   294633935.614746 295067502.563091         0.146938  50251
7158518193.500000  363660266311983.375000
packet_loss_fraction            0.053397         0.053281         0.217688
0.000051                0.000129
```

8.4 Análise Preditiva

As tabelas acima comparam o desempenho previsto pelo modelo com o desempenho real do conjunto de teste.

A coluna `perc_error_mean` (Erro Percentual da Média) mostra o quão próximo nosso `pred_mean` chegou do `real_mean`. Vemos que, para a maioria das variáveis, o erro é muito baixo (especialmente no `client13`), indicando um ótimo poder preditivo do modelo Bayesiano para estimar o comportamento médio.

Fase 3 (Cont.): Comparação Final (MLE vs. Bayes)

9. Comparação das Estimativas Pontuais

Esta seção compara as estimativas pontuais obtidas pelo método da Máxima Verossimilhança (MLE, Etapa 4) com as estimativas pontuais Bayesianas (a média da distribuição posterior, Etapa 7).

O objetivo é observar o quão próximas são as estimativas e discutir o **impacto da prior**.

In [134…

```
def create_mle_vs_bayes_table(mle_params, bayes_params):
    comparison = {}

    # 1. RTT (Normal-Normal)
    for var in ['rtt_download_sec', 'rtt_upload_sec']:
        mle_mean = mle_params[var]['mu']
        bayes_mean = bayes_params[var]['post_mu_n']
        comparison[var] = {'param': 'μ (Média)', 'mle_estimate': mle_mean, 'baye

    # 2. Throughput (Gama-Gama)
    for var in ['download_throughput_bps', 'upload_throughput_bps']:
        k_mle = mle_params[var]['shape_k']
        scale_mle = mle_params[var]['scale']
        mle_mean = k_mle * scale_mle

        a_n = bayes_params[var]['post_a_n']
        b_n = bayes_params[var]['post_b_n']
        bayes_mean = (k_mle * b_n) / (a_n - 1)

        comparison[var] = {'param': 'E[Y] (Média)', 'mle_estimate': mle_mean, 'b

    # 3. Perda (Beta-Binomial)
    var = 'packet_loss_fraction'
    mle_mean = mle_params[var]['p_mle']
    bayes_mean = bayes_params[var]['post_mean_p']
    comparison[var] = {'param': 'p (Prob. Perda)', 'mle_estimate': mle_mean, 'ba

    df = pd.DataFrame(comparison).T
    df['diff'] = df['bayes_estimate'] - df['mle_estimate']
    df['perc_diff'] = (df['diff'] / df['mle_estimate']) * 100

    return df[['param', 'mle_estimate', 'bayes_estimate', 'diff', 'perc_diff']]

print("--- Comparação MLE vs. Bayes (client12) ---")
final_comp_c12 = create_mle_vs_bayes_table(mle_params_c12, bayesian_params_c12)
print(final_comp_c12)

print("\n--- Comparação MLE vs. Bayes (client13) ---")
final_comp_c13 = create_mle_vs_bayes_table(mle_params_c13, bayesian_params_c13)
print(final_comp_c13)
```

```
--- Comparação MLE vs. Bayes (client12) ---
                                param     mle_estimate   bayes_estimate
diff perc_diff
rtt_download_sec               μ (Média)         0.086485         0.082785
-0.003701 -4.278925
rtt_upload_sec                 μ (Média)         0.073462         0.069733
-0.003729 -5.076186
download_throughput_bps     E[Y] (Média) 464765894.805978 474895169.841642 101292
75.035664  2.179436
upload_throughput_bps       E[Y] (Média)  92171175.646829  94116989.882016  19458
14.235187  2.111088
packet_loss_fraction     p (Prob. Perda)         0.000638         0.000780
0.000142 22.180458

--- Comparação MLE vs. Bayes (client13) ---
                                param     mle_estimate   bayes_estimate
diff perc_diff
rtt_download_sec               μ (Média)         0.011748         0.011647      -
0.000101 -0.860959
rtt_upload_sec                 μ (Média)         0.012986         0.013139
0.000153  1.176112
download_throughput_bps     E[Y] (Média) 593933162.983242 594364345.027721  43118
2.044479  0.072598
upload_throughput_bps       E[Y] (Média) 294764544.291732 294633935.614746 -13060
8.676986 -0.044309
packet_loss_fraction     p (Prob. Perda)         0.053361         0.053397
0.000036  0.066778
```

9.1 Análise da Comparação e do Impacto da Prior

As tabelas acima nos mostram duas histórias muito diferentes:

1. Para o `client13` (Dados Consistentes e Abundantes): As estimativas `mle_estimate` e `bayes_estimate` são **quase idênticas**. A diferença percentual (`perc_diff`) é minúscula, sempre abaixo de 1.2%.

- **Impacto da Prior:** Isso é o que acontece quando temos **muitos dados** (centenas de linhas) e **priors fracas**. A *likelihood* (os dados) domina completamente a *prior*, e o resultado Bayesiano converge para o resultado do MLE.

2. Para o `client12` (Dados Escassos ou Assimétricos): Aqui, vemos um **impacto claro da prior**:

- **RTT e Throughput (`perc_diff` de 2% a 5%):** Vemos que a estimativa Bayesiana foi "puxada" levemente em direção à prior. Por exemplo, a média do RTT (MLE) era `0.086` , e a estimativa Bayesiana (`0.082`) foi puxada em direção à nossa prior de `mu_0 = 0.0` . A diferença ainda é pequena, mas existe.
- **Packet Loss (`perc_diff` de 22.18%):** Esta é a observação mais importante. Por que uma diferença tão grande?
  - O `client12` quase **não tem perdas**. Os dados de "sucesso" (perda de pacotes) são *extremamente escassos*.
  - Quando os dados são escassos, a prior (mesmo uma "fraca" como a de Jeffreys, `a_0=0.5, b_0=0.5`) tem um **impacto proporcionalmente enorme**.

- A prior "puxou" a estimativa (que era `0.000638`) para longe do zero (`0.000780`), efetivamente nos dizendo: "Mesmo que você tenha visto quase zero perdas, eu não vou deixar você ter 100% de certeza de que a perda é zero".

Conclusão da Análise: Os resultados confirmam perfeitamente a teoria Bayesiana. Onde os dados são abundantes e consistentes (client13), Bayes ≈ MLE. Onde os dados são escassos ou inconsistentes (client12), a prior (mesmo fraca) tem um impacto visível e importante no resultado.

Conclusão Geral do Projeto

Concluímos todas as etapas de análise solicitadas no documento do projeto.

1. **EDA (Fase 1):** Carregamos os dados, calculamos estatísticas e selecionamos `client12` (lento/confiável) e `client13` (rápido/instável), validando essa escolha com histogramas, boxplots e scatter plots.
2. **Modelagem e MLE (Fase 2):** Definimos os modelos (Normal, Gama, Binomial) e calculamos os parâmetros $\hat{\theta}_{MLE}$. Avaliamos o *fit* e confirmamos que o modelo Gama (Throughput) se ajustou muito bem, enquanto o modelo Normal (RTT) foi um *fit* ruim (mas necessário), especialmente para o `client12`.
3. **Bayes (Fase 3):**
   - Dividimos os dados (70/30) e justificamos nossas priors fracamente informativas.
   - Calculamos os parâmetros da posterior.
   - Calculamos a predição Bayesiana (Etapa 8), que se mostrou muito precisa (baixo `perc_error_mean`) na previsão da média do conjunto de teste.
   - Comparamos MLE vs. Bayes (Etapa 9) e demonstramos o **impacto da prior**: onde os dados eram abundantes (client13), o impacto foi nulo (Bayes ≈ MLE). Onde os dados eram escassos (perda do client12), a prior teve um impacto significativo (22%), protegendo a estimativa de conclusões extremas.

Uso de Ferramentas de IA e Link do Código

- **Link para o repositório:** https://github.com/vrtaranto-ufrj/projeto-probest-parte-1
- **Uso de IA:** O desenvolvimento deste projeto foi realizado utilizando um modelo de linguagem (Gemini) como assistente de programação no estilo "vibe coding". A IA foi utilizada para:
  - Estruturar o problema em etapas lógicas com base no PDF.
  - Gerar o *boilerplate* (código base) para as análises em `pandas`, `matplotlib`, `seaborn` e `scipy`.
  - Auxiliar ativamente no *debugging* de erros complexos encontrados durante a implementação.

- Ajustar e refinar as análises e os textos de markdown (como esta conclusão) para garantir que todos os requisitos do projeto fossem atendidos.