In [1]:
```
import pandas as pd
import numpy as np
from scipy import stats
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from tqdm.notebook import tqdm
import re
pd.set_option('max_columns', None)
pd.options.display.max_colwidth = 100
from pandas.api.types import CategoricalDtype
```

In [2]:
```
df = pd.read_csv('data.csv', index_col=0)
df.shape
```

Out[2]:
```
(6643221, 4)
```

In this part, we aim to understand how different user agents (e.g. web browsers) are accessing the network and the volume of requests they are generating. We will begin by tallying the total number of requests made by each agent and group them accordingly. Then, we will calculate each group's contribution to the total requests and traffic across the network.

1 Number of agents in each range

In [3]:
```
df1 = df[['agent','timestamp']].groupby(['agent']).count()
df1 = df1.rename(columns={"timestamp": "count"})
df1 = df1.reset_index()
df1.head()
```

Out[3]:

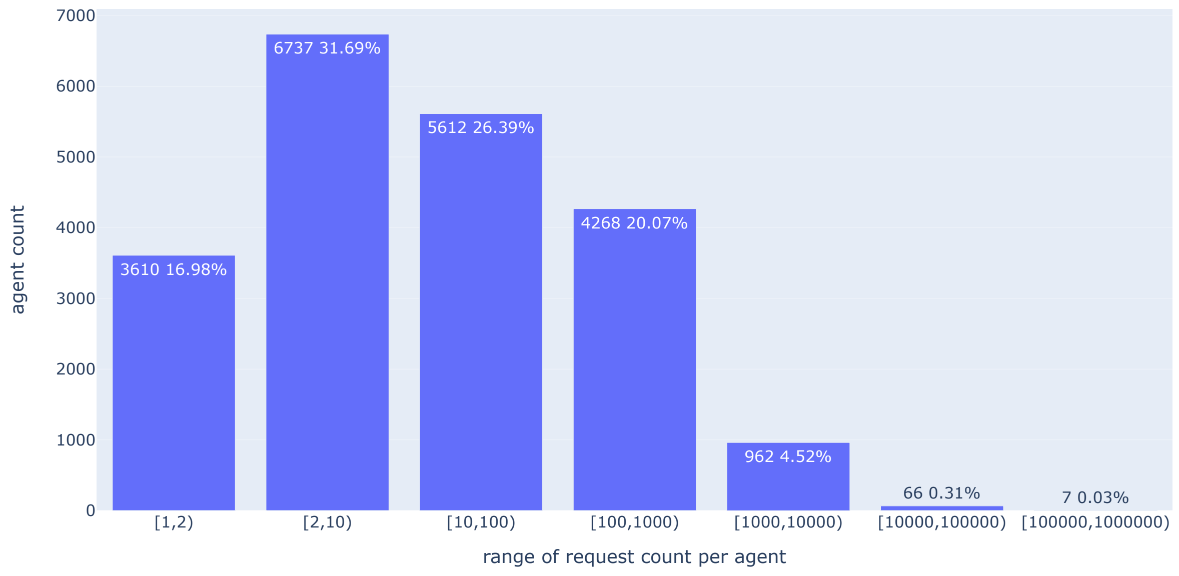
	agent	count
0	AVProMobileVideo/6.1.7.39280 (Linux;Android 10) ExoPlayerLib/2.15.0	1
1	AccompanyBot	22
2	ActionExtension/3 CFNetwork/1220.1 Darwin/20.3.0	5
3	AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/257872-0020)	101
4	AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/ElVU8BViYT0YUCNRKu1tWQNNxfpQUqz5a9U...	413

In [4]:
```
df2 = pd.DataFrame(columns = ['request_time', 'count'])

def addRow(df2, l, r):
    df_temp = df1[(df1['count'] >= l) & (df1['count'] < r)]
    c = df_temp.count()[0]
    df2 = df2.append({'request_time':'['+str(l)+','+str(r)+')', 'count':c}, ignore_index = True)
    return df2

df2 = addRow(df2, 1, 2)
df2 = addRow(df2, 2, 10)
df2 = addRow(df2, 10, 100)
df2 = addRow(df2, 100, 1000)
df2 = addRow(df2, 1000, 10000)
df2 = addRow(df2, 10000, 100000)
df2 = addRow(df2, 100000, 1000000)

df2.replace('[1,2)', '1')
df2.replace('[100000,1000000)', '[100000,+∞)')

total = df2['count'].sum()
df2['percentage'] = df2['count']/total
df2
```

Out[4]:

	request_time	count	percentage
0	[1,2)	3610	0.169786
1	[2,10)	6737	0.316856
2	[10,100)	5612	0.263945
3	[100,1000)	4268	0.200734
4	[1000,10000)	962	0.045245
5	[10000,100000)	66	0.003104
6	[100000,1000000)	7	0.000329

```
In [5]: fig = px.bar(df2, x='request_time', y='count', text=[str(x[0])+'\n{0:1.2f}%'.format(x[1]*100) for x in zip(df2['count'

        fig.update_xaxes(title='range of request count per agent')
        fig.update_yaxes(title='agent count')

        fig.show()
```



2 Number of requests and traffic for each group of agents

```
In [6]: df3_1 = df[['agent','timestamp']].groupby(['agent']).agg('count')
        df3_1 = df3_1.rename(columns={"timestamp": "count"})
        df3_1 = df3_1.reset_index()

        df3_2 = df[['agent','bytes_returned']].groupby(['agent']).agg('sum')
        df3_2 = df3_2.rename(columns={"bytes_returned": "size"})
        df3_2 = df3_2.reset_index()

        df3 = df3_1.set_index('agent').join(df3_2.set_index('agent'))
        df3 = df3.reset_index()

        df3.head()
```

Out[6]:

	agent	count	size
0	AVProMobileVideo/6.1.7.39280 (Linux;Android 10) ExoPlayerLib/2.15.0	1	6629429
1	AccompanyBot	22	244764
2	ActionExtension/3 CFNetwork/1220.1 Darwin/20.3.0	5	1586273
3	AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/257872-0020)	101	64108028
4	AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/ElVU8BViYT0YUCNRKu1tWQNNxfpQUqz5a9U...	413	525377961

```
In [7]: df3['type'] = ''

        def addType(l, r, name):
            df3.loc[(df3['count'] >= l) & (df3['count'] < r), 'type'] = name

        addType(1, 2, '1')
        addType(2, 10, '[2,10)')
        addType(10, 100, '[10,100)')
        addType(100, 1000, '[100,1,000)')
        addType(1000, 10000, '[1,000,10,000)')
        addType(10000, 100000, '[10,000,100,000)')
        addType(100000, 1000000, '[100,000,1,000,000)')

        df3.head()
```

Out[7]:

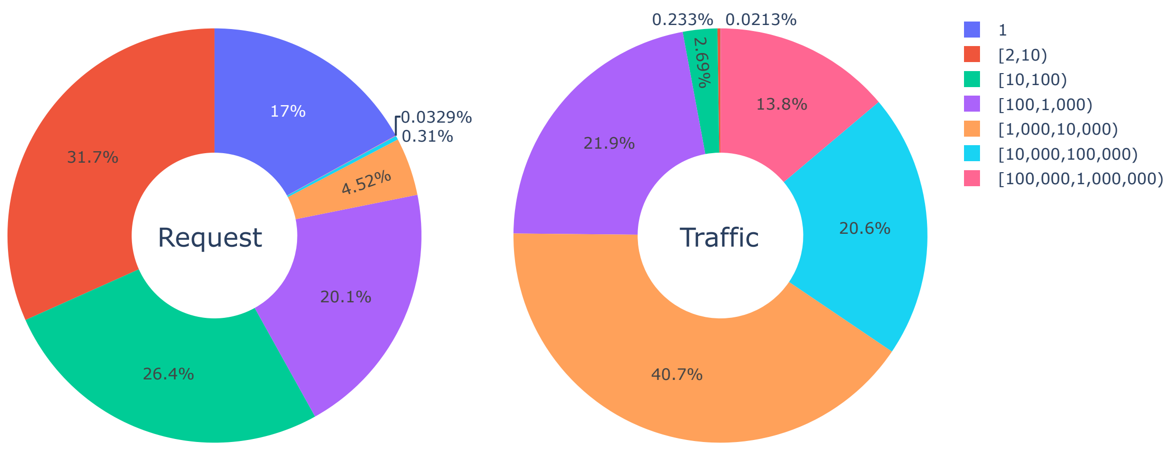
	agent	count	size	type
0	AVProMobileVideo/6.1.7.39280 (Linux;Android 10) ExoPlayerLib/2.15.0	1	6629429	1
1	AccompanyBot	22	244764	[10,100)
2	ActionExtension/3 CFNetwork/1220.1 Darwin/20.3.0	5	1586273	[2,10)
3	AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/257872-0020)	101	64108028	[100,1,000)
4	AirPlay/2.0 (App/30.172.0) MFi_AirPlay_Device (MFiModelGroup/ElVU8BViYT0YUCNRKu1tWQNNxfpQUqz5a9U...	413	525377961	[100,1,000)

```
In [8]: df4_1 = df3[['count', 'type']].groupby('type').agg('count')
        df4_1 = df4_1.reset_index()

        df4_2 = df3[['size', 'type']].groupby('type').agg('sum')
        df4_2 = df4_2.reset_index()

        df4 = df4_1.set_index('type').join(df4_2.set_index('type'))
        df4 = df4.reset_index()

        cat_order = CategoricalDtype(
            ['1', '[2,10)', '[10,100)', '[100,1,000)', '[1,000,10,000)', '[10,000,100,000)', '[100,000,1,000,000)'],
            ordered=True
        )
        df4['type'] = df4['type'].astype(cat_order)
        df4 = df4.sort_values('type')
        df4['size'] = df4['size']/pow(1024,3) #GB
        df4.head()
```

```
Out[8]:
              type  count         size
   0             1   3610     1.248206
   6        [2,10)   6737    13.677831
   3      [10,100)   5612   158.108663
   5   [100,1,000)   4268  1286.671503
   1 [1,000,10,000)   962  2389.261190
```

```
In [9]: # Create subplots: use 'domain' type for Pie subplot
        fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])

        fig.add_trace(go.Pie(
                labels=df4['type'],
                values=df4['count'],
                sort=False),1,1
            )
        fig.add_trace(go.Pie(
                labels=df4['type'],
                values=df4['size'],
                sort=False),1,2
            )


        # Use `hole` to create a donut-like pie chart
        fig.update_traces(hole=.4, hoverinfo="label+percent+name")

        fig.update_layout(
            title_text="Number of requests and traffic for each group of agents",
            # Add annotations in the center of the donut pies.
            annotations=[dict(text='Request', x=0.16, y=0.5, font_size=20, showarrow=False),
                         dict(text='Traffic', x=0.82, y=0.5, font_size=20, showarrow=False)])

        fig.show()
```

Number of requests and traffic for each group of agents


Request
Traffic
[2,10)
[10,100)
[100,1,000)
[1,000,10,000)
[10,000,100,000)
[100,000,1,000,000)