Project 4
    Question 3 (Figure 3)

    Create a music collaboration network visualization using Plotly.
a. Select 10 singers/musicians. For each musician, identify at least two collaborators and the songs they collaborated on. The more artists you include in your visualization the better.
    For example, Drake collaborated with Rihanna on “Work” and “What’s My Name?”, with Lil Wayne on “She Will”, etc.

b.Create a network visualization of the collaborations.
    i. Each node represents a musician with the name of the musician displayed.
    ii.Each edge represents a collaboration between two musicians. The name of the song should be displayed next to the edge.
    iii.If more than two musicians collaborated in a project, each musician should be connected to every other musician.
    iv. If two musicians collaborate more than once, create multiple edges between them.
    v. Pictures are optional.
    vi. Here is an example of Jazz music collaboration network visualization: https://linkedjazz.org/network/

c. Create THREE visualizations with three different layout algorithms.
    i. Some layout may not look good. It’s OK. The goal is to let you experiment with different layouts and learn how to adjust layout parameters.

d.You can choose the style of the visualization.

e. You decide how to handle the data. You may hard code the data in the Python program or create a spreadsheet and load it into your program.
    i. If you use a spreadsheet, make sure you submit the spreadsheet with your code and PDF file.

f. You can find musicians, songs, and their collaborators at https://www.billboard.com/charts/artist-100. Click on an artist and look at his/her Chart History. Or use your own source of information.

g. Write your code in Jupyter Notebook. Submit the Jupyter Notebook and a PDF file with the figure.

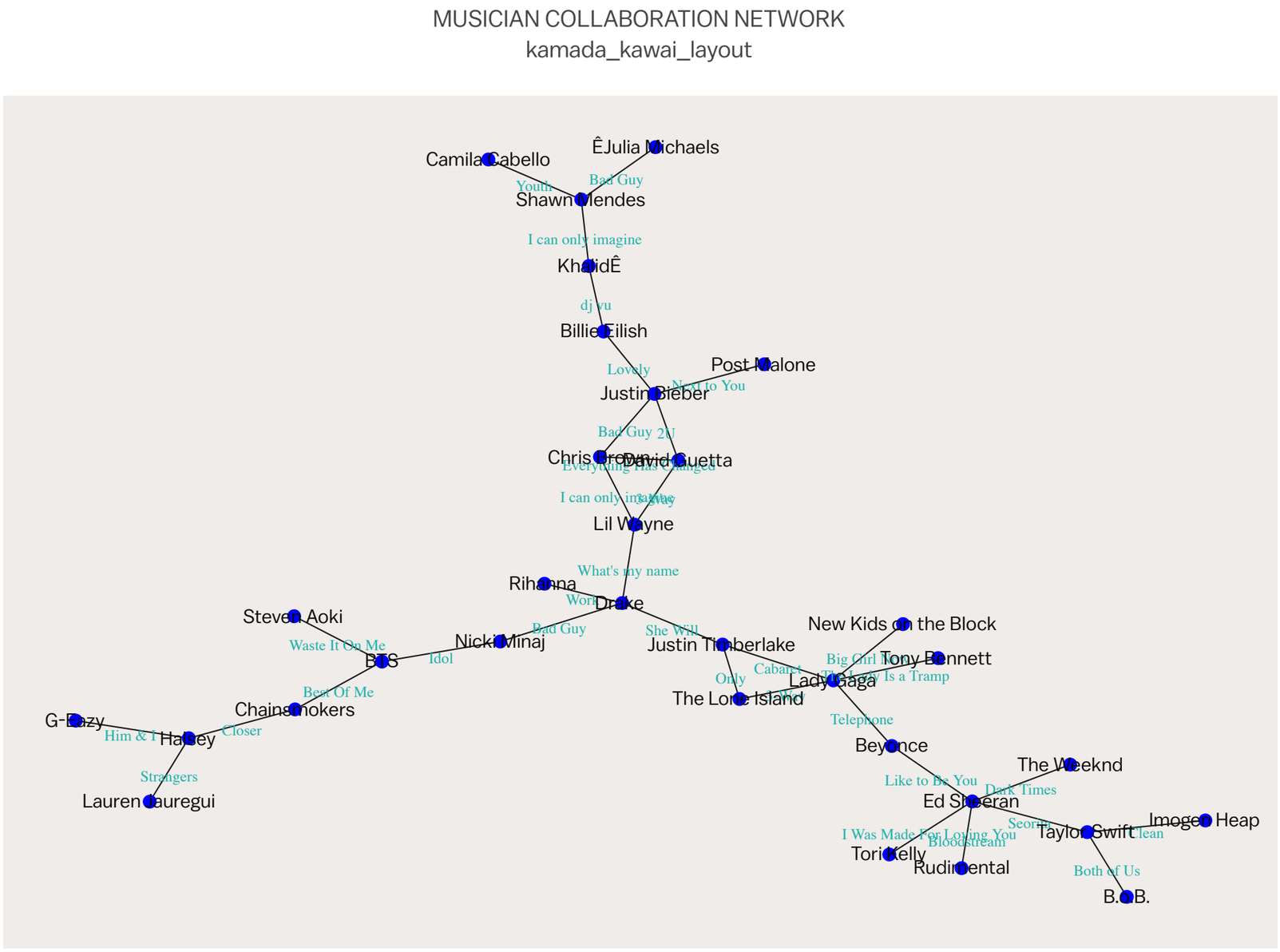
In [ ]:
```
import plotly.plotly as py
import networkx as nx
import pandas as pd

df = pd.read_csv('~lisun/GSU/2019 Summer/Data Visualization/Project4/Question3/MusicianAndCollebrationList.csv',\
                 encoding = "ISO-8859-1")
df = df.dropna()

G=nx.Graph()  #  G is an empty Graph

my_nodes=pd.concat([df['ID_Musician'],df['ID_Collaboration']]).unique()
G.add_nodes_from(my_nodes)

my_edges=list(zip(df['ID_Musician'],df['ID_Collaboration']))
my_edges
G.add_edges_from(my_edges, labels = df['Song_Name'])

def make_fig(pos, df, title):
    labels=pd.concat([df['Musician'],df['Collaboration']]).unique()

    Xn=[pos[k][0] for k in pos.keys()]
    Yn=[pos[k][1] for k in pos.keys()]


    trace_nodes=dict(type='scatter',
                 x=Xn,
                 y=Yn,
                 mode='markers',
```

```
                marker=dict(size=10, color='rgb(0,0,255)'),
                text=labels,
                hoverinfo='text')

    Xe=[]
    Ye=[]
    Xm=[]
    Ym=[]
    for e in G.edges():
        Xe.extend([pos[e[0]][0], pos[e[1]][0], None])
        Ye.extend([pos[e[0]][1], pos[e[1]][1], None])
        mid_edge = (pos[e[0]]+pos[e[1]])/2   # calculate the midpoint of each edge
        Xm.append(mid_edge[0])
        Ym.append(mid_edge[1])


    trace_edges=dict(type='scatter',
                     mode='lines',
                     x=Xe,
                     y=Ye,
                     line=dict(width=1, color='rgb(25,25,25)'),
                     hoverinfo= 'none'
                    )

    trace_edge_text=dict(
                    type='scatter',
                     mode='text',
                     x=Xm,
                     y=Ym,
                     text =df['Song_Name'],
                     textfont=dict(
                        family="sans serif",
                        size=11,
                        color="LightSeaGreen"
                    ),
                    textposition='bottom center',
                    hoverinfo='text'

    )

    axis=dict(showline=False, # hide axis line, grid, ticklabels and  title
              zeroline=False,
              showgrid=False,
              showticklabels=False,
              title=''
              )


    layout=dict(title= title,
                font= dict(family='Balto'),
                width=1000,
                height=800,
                autosize=False,
                showlegend=False,
                xaxis=axis,
                yaxis=axis,
                margin=dict(
                l=40,
                r=40,
                b=85,
                t=100,
                pad=0,

        ),
        hovermode='closest',
```

```
        plot_bgcolor='#efecea', #set background color
        )


    fig = dict(data=[trace_edges, trace_nodes,trace_edge_text], layout=layout)

    def make_annotations(pos, anno_text, font_size=14, font_color='rgb(10,10,10)'):
        L=len(pos)
        if len(anno_text)!=L:
            raise ValueError('The lists pos and text must have the same len')
        annotations = []
        for k, key in zip(range(L), pos.keys()):
            annotations.append(dict(text=anno_text[k],
                                    x=pos[key][0],
                                    y=pos[key][1],
                                    xref='x1', yref='y1',
                                    font=dict(color= font_color, size=font_size),
                                    showarrow=False)
                               )
        return annotations

    fig['layout'].update(annotations=make_annotations(pos, labels))

    return fig
```

In [8]:
```
##############################
####  kamada_kawai_layout ####
##############################

pos=nx.kamada_kawai_layout(G)
title = 'MUSICIAN COLLABORATION NETWORK' + '<br>' + 'kamada_kawai_layout'
fig = make_fig(pos, df, title)
py.iplot(fig, file = 'MusicianCollaborationNetwork')
```

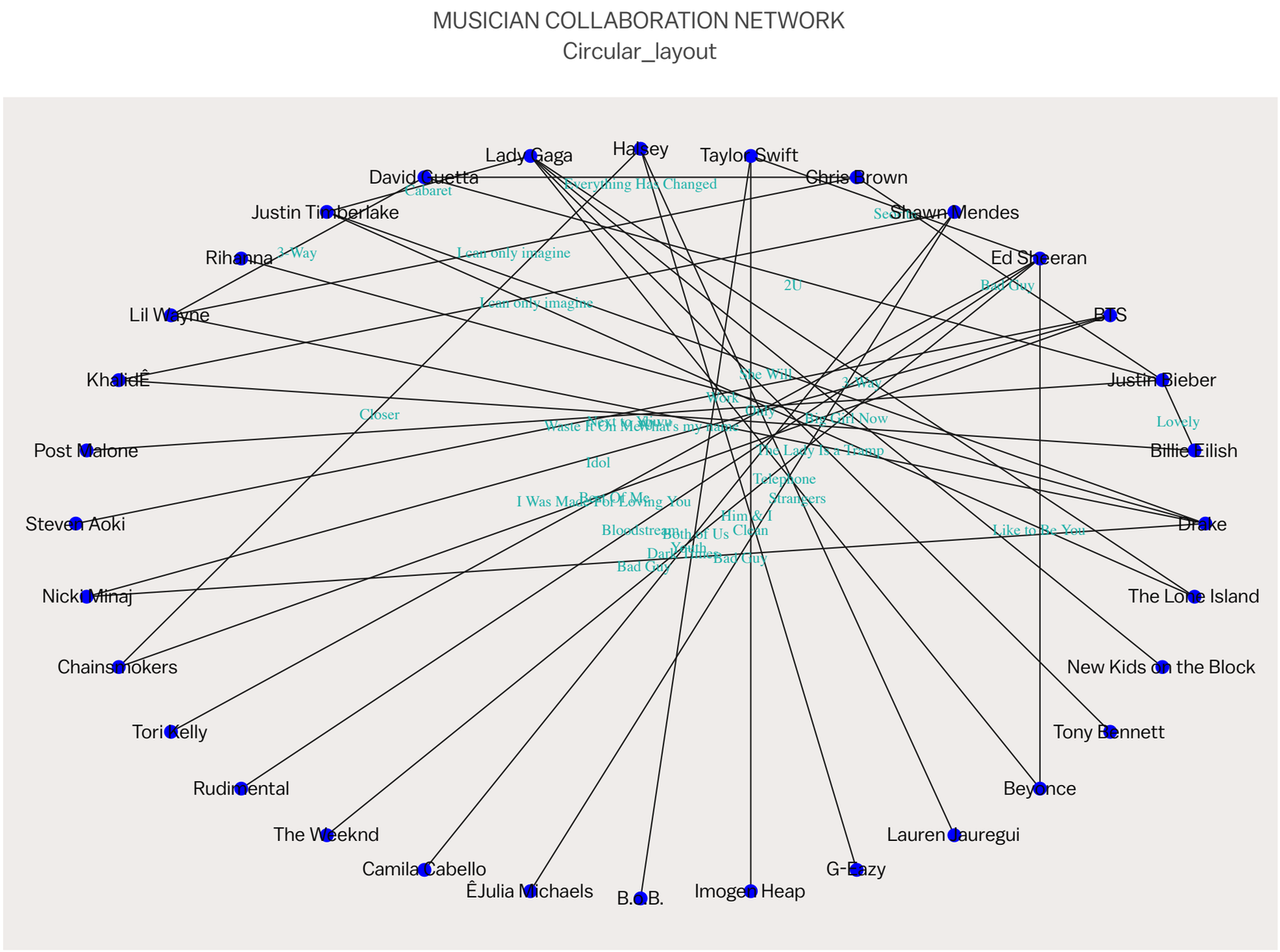
Out[8]:



EDIT CHART

In [16]:
```
##############################
####  Circular_layout ####
##############################

pos=nx.circular_layout(G)
title = 'MUSICIAN COLLABORATION NETWORK' + '<br>' + 'Circular_layout'
fig = make_fig(pos, df, title)
py.iplot(fig, file = 'MusicianCollaborationNetwork')
```

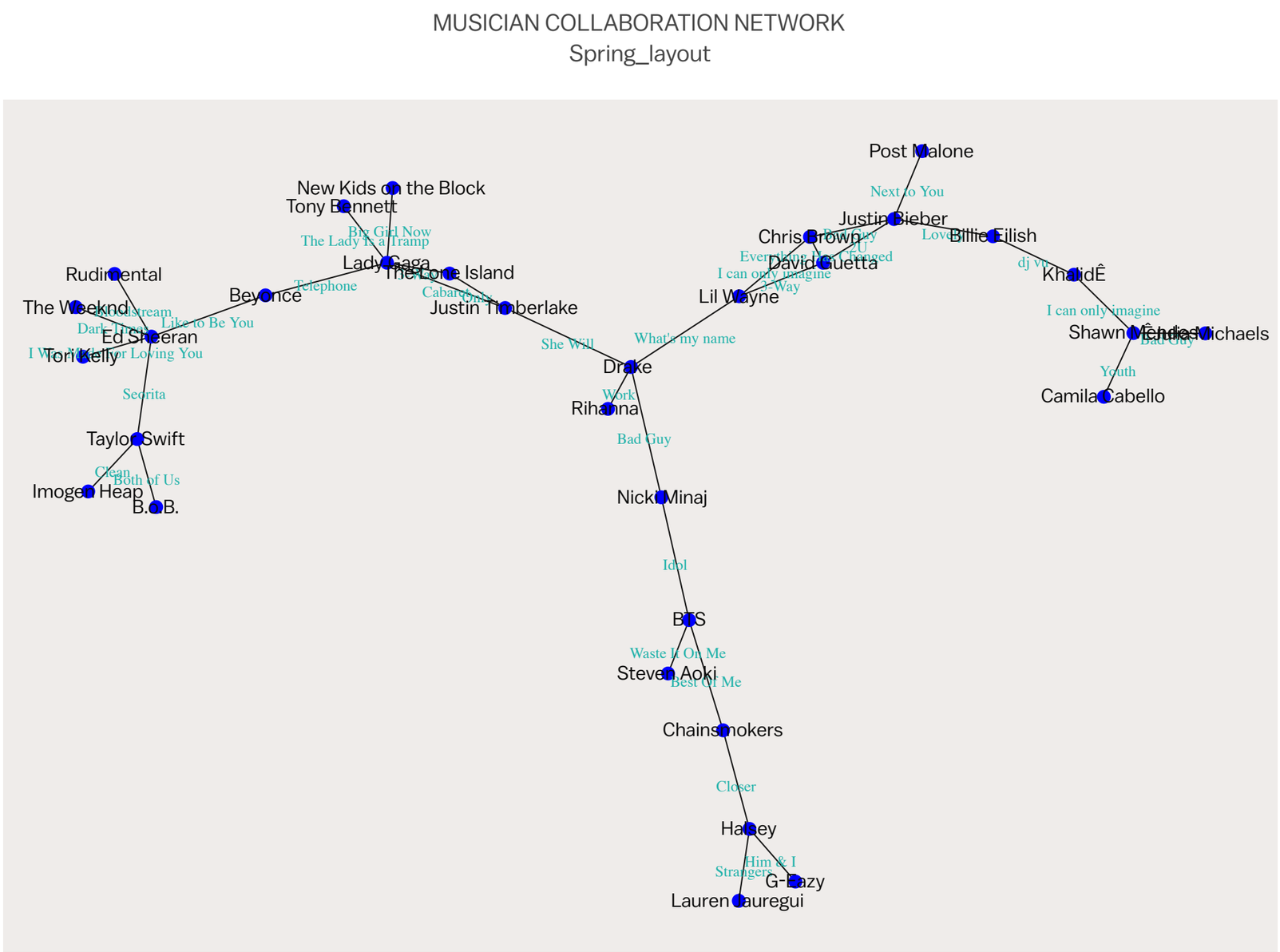
Out[16]:



EDIT CHART

In [20]:
```
######################
####  Spring_layout ####
######################

pos=nx.spring_layout(G)
title = 'MUSICIAN COLLABORATION NETWORK' + '<br>' + 'Spring_layout'
fig = make_fig(pos, df, title)
py.iplot(fig, file = 'MusicianCollaborationNetwork')
```

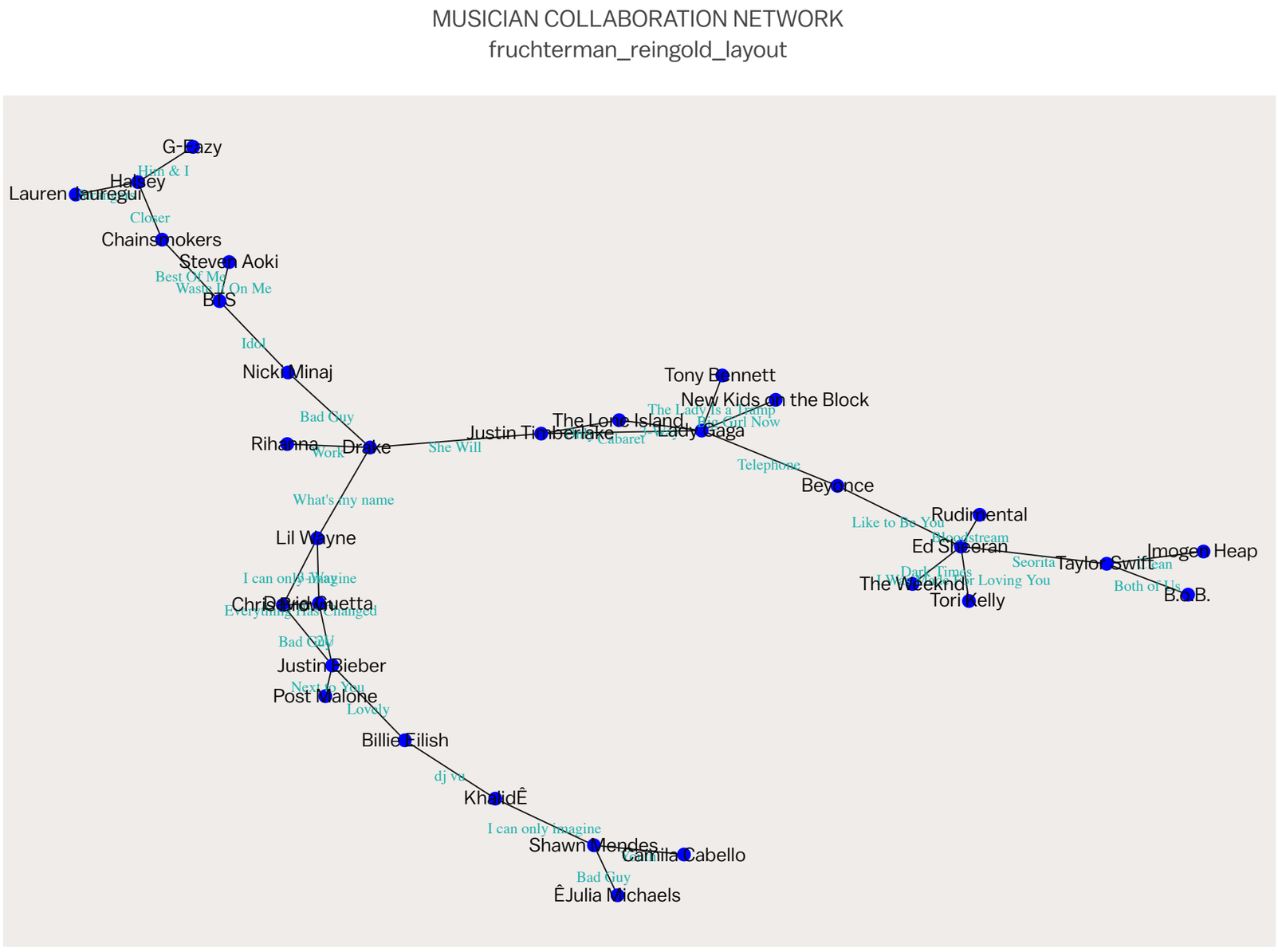
Out[20]:



EDIT CHART

In [23]:
```
####################################
#### fruchterman_reingold_layout #####
####################################

pos=nx.fruchterman_reingold_layout(G)
title = 'MUSICIAN COLLABORATION NETWORK' + '<br>' + 'fruchterman_reingold_layout'
fig = make_fig(pos, df, title)
py.iplot(fig, file = 'MusicianCollaborationNetwork')
```

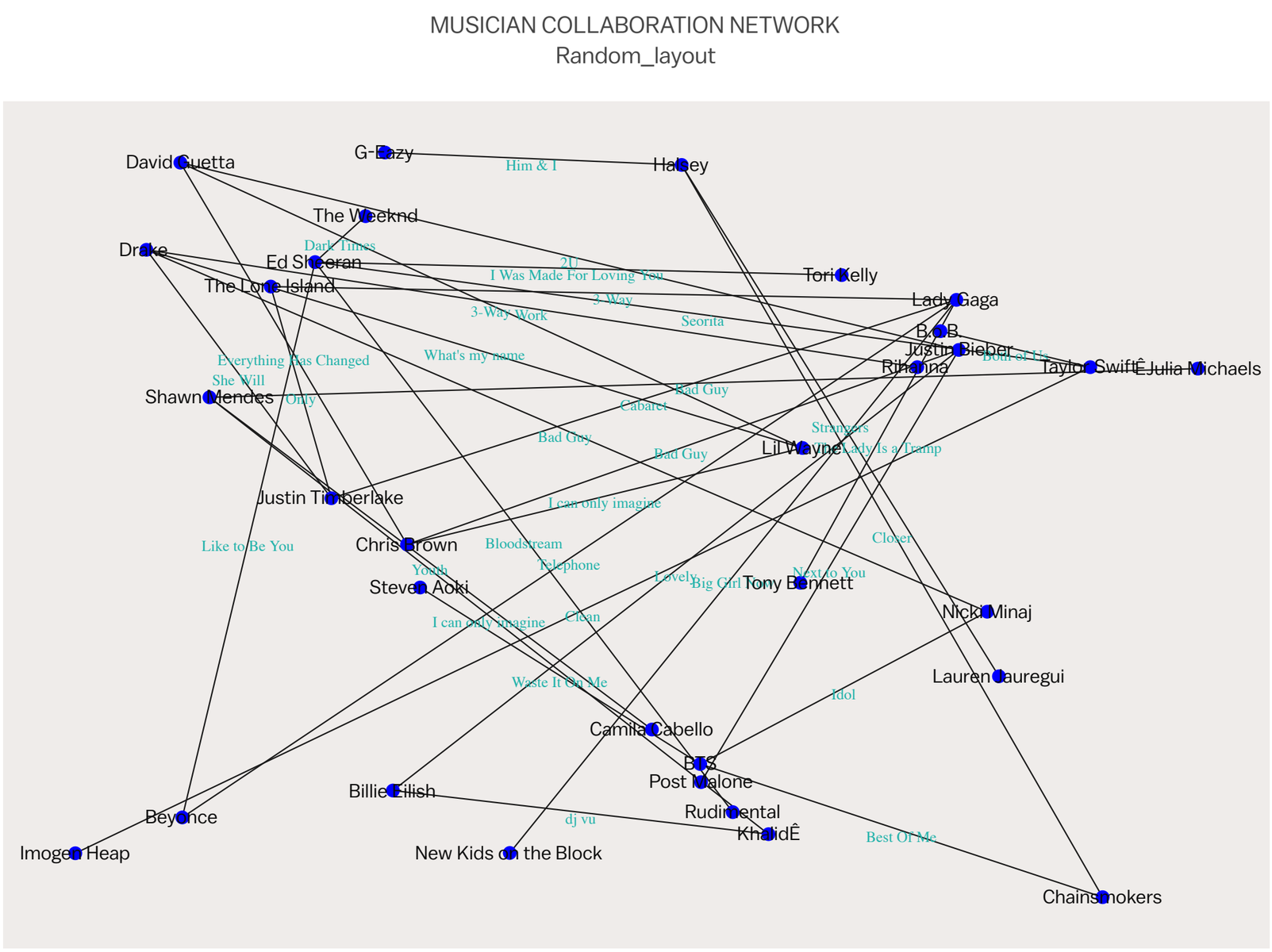
Out[23]:



EDIT CHART

In [18]:
```
######################
####  random_layout ####
######################

pos=nx.random_layout(G)
title = 'MUSICIAN COLLABORATION NETWORK' + '<br>' + 'Random_layout'
fig = make_fig(pos, df, title)
py.iplot(fig, file = 'MusicianCollaborationNetwork')
```

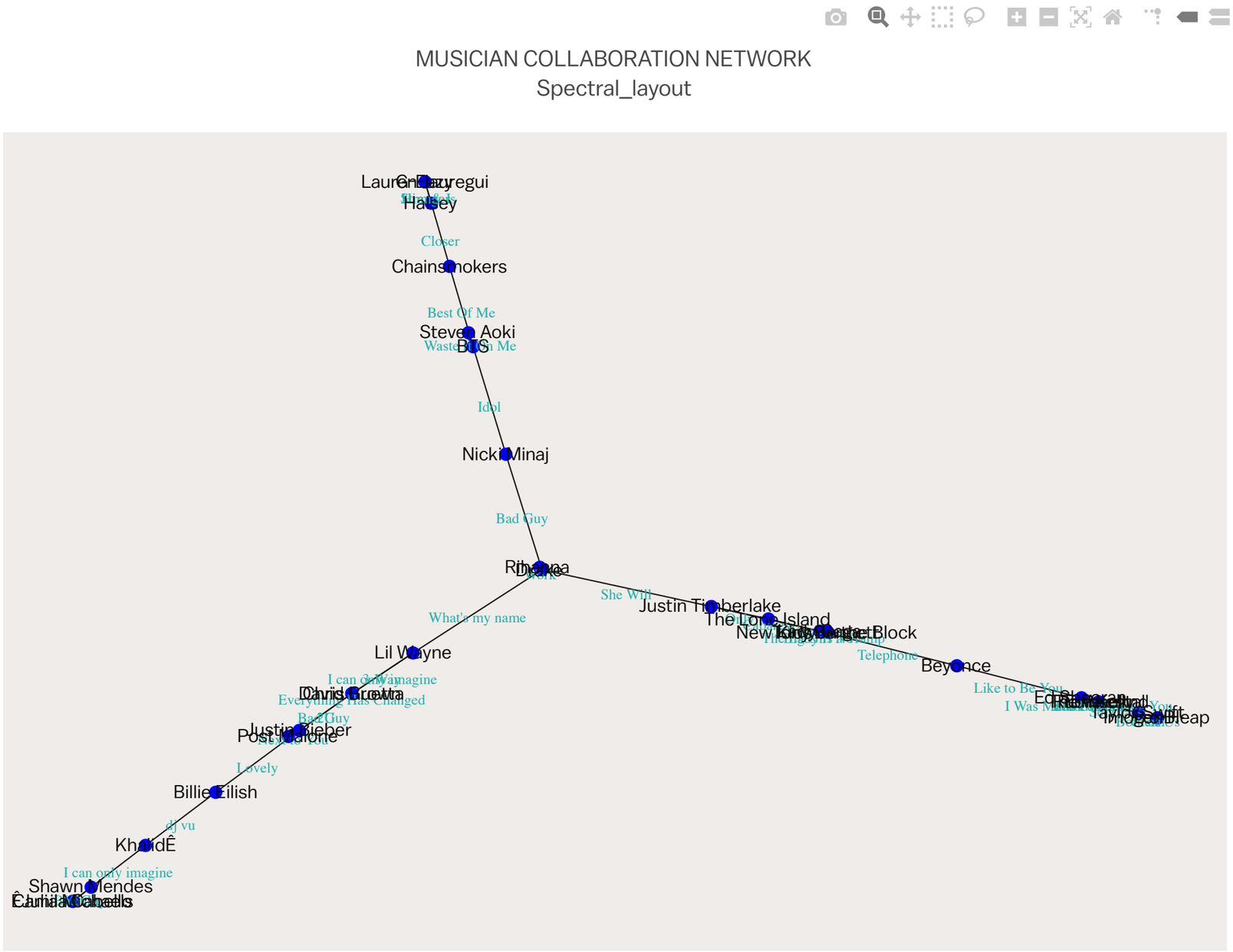
Out[18]:



EDIT CHART

In [22]:
```
#########################
####  Spectral_layout ####
#########################

pos=nx.spectral_layout(G)
title = 'MUSICIAN COLLABORATION NETWORK' + '<br>' + 'Spectral_layout'
fig = make_fig(pos, df, title)
py.iplot(fig, file = 'MusicianCollaborationNetwork')
```

Out[22]:



EDIT CHART

In [ ]:
```
```