In [250]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import statsmodels.api

import re
```

Funcoes para ler os arquivos

In [227]:

```
#Le os arquivos do teste informado por parametro. Recebe como argumento o numero do tes
te assim como o tipo do teste :
#"parimpar ou SW"
def read_files(num_teste, tipo_teste):

    path1 = "../iperftestes/Saidas/Teste" + str(num_teste) + "/info_" + tipo_teste + ".
txt"
    path2 = "../iperftestes/Saidas/Teste" + str(num_teste) + "/iperfs.txt"
    path3 = "../iperftestes/Saidas/Teste" + str(num_teste) + "/teste" + str(num_teste)
+ ".csv"

    df1 = pd.read_csv(path1, sep=" ")
    df2 = pd.read_csv(path2, sep="\t")
    df3 = pd.read_csv(path3)

    return df1, df2, df3



#Converte o arquivo ping?W para um dataframe de uma coluna so com o ping ! Retorna o da
taframe gerado
def convert_ping(nome_arquivo, num_teste):

    path = "../iperftestes/Saidas/Teste" + str(num_teste) + "/" + nome_arquivo + ".txt"
#caminho do arquivo

    with open(path) as stream:#Abre o arquivo

        times = [] #lista que armazenara todos os pings para coloca-los posteriormente
 em um series->dataframe

        next(stream)  # Ignora a primeira linha do arquivo de entrada

        for line in stream: #Pega todas as linhas
            string = line.split(" ") #String eh uma lista com cada palavra da linha

            if(line != "Request timed out\n"): #Se a linha tiver o ping

                for i in range(0,len(string)): #Para cada palavra na lista string, proc
ura "time"
                    if(re.match(r'time', string[i])): #Pega somente os valores numerico
s da palavra
                        time = re.findall(r'\d.+', string[i])
                        times.append(float(time[0])) #Transforma-os para float e coloca
-os na lista times

        series = pd.Series(times) #Transforma a lista em uma Series
        ds = pd.DataFrame(series, columns=['Ping']) # E por fim em um dataframe

    return ds
```

Método ParImpar (Análises)

In [262]:

```
df1, df2, df3 = read_files(10, "parimpar")
df4 = convert_ping("pingHW", 10)
df5 = convert_ping("pingSW", 10)
```

In [229]:

```
df1.head()
```

Out[229]:

	Tempo	Switch	RegrasInstaladas	RegrasAceitas	RegrasBloqueadas	BytesEnviados
0	13	SW	28	28	0	18721584
1	13	HW	22	22	0	15307488
2	23	HW	42	42	0	89062848
3	23	SW	48	48	0	104408136
4	33	HW	63	63	0	241986528

In [230]:

```
df2.head()
```

Out[230]:

	Inicio(seg)	Duracao(seg)	Banda(Kbps)	PCli	PServ
0	0.251040	78.275461	40.053267	18501	14001
1	0.314698	76.136369	1318.216400	18502	14002
2	0.361013	23.815972	2730.020501	18503	14003
3	1.127516	13.392282	2208.581846	18504	14004
4	1.546171	5.375085	1371.062812	18505	14005

In [231]:

```
df3.head()
```

Out[231]:

	Timestamp	IpOri	PortaOri	IpDest	PortaDest	?	Tempo	BytesEnv	Banda(bps)
0	20191016105343	10.1.0.2	18505	10.1.0.1	14005	3	0.0-5.4	945210	1403890
1	20191016105343	10.1.0.1	14005	10.1.0.2	18505	3	0.0-5.4	942270	1405350
2	20191016105350	10.1.0.2	18504	10.1.0.1	14004	3	0.0-13.4	3786720	2260984
3	20191016105350	10.1.0.1	14004	10.1.0.2	18504	3	0.0-13.4	3785250	2261171
4	20191016105352	10.1.0.2	18526	10.1.0.1	14026	3	0.0-9.1	4712820	4145133

In [232]:

```
df4.head()
```

Out[232]:

	Ping
0	1.86
1	0.87
2	0.82
3	0.75
4	0.68

In [233]:

```
df5.head()
```

Out[233]:

	Ping
0	2.46
1	1.26
2	0.58
3	0.57
4	0.58

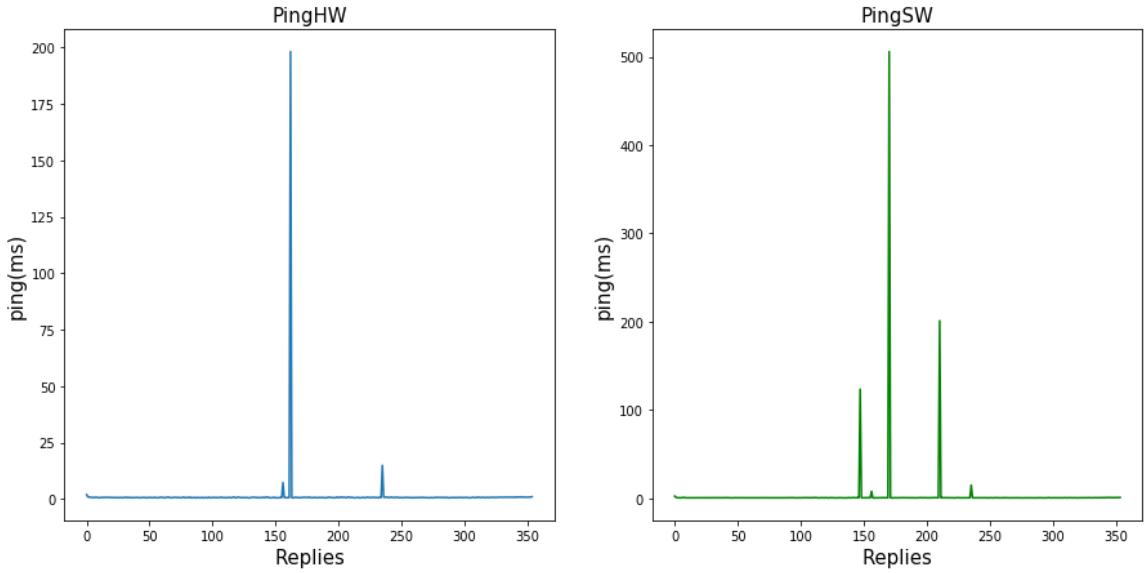
Pings (Referentes a HW e SW)

In [245]:

```
y = df4.Ping.to_numpy()
y2 = df5.Ping.to_numpy()

fig, (ax1, ax2) = plt.subplots(1,2,figsize = (15,7))

ax1.plot(y)
ax1.set_title("PingHW", fontsize=15)
ax1.set_xlabel('Replies', fontsize=15)
ax1.set_ylabel('ping(ms)', fontsize=15)
ax2.plot(y2, color='green', linestyle='-')
ax2.set_title("PingSW", fontsize=15)
ax2.set_xlabel('Replies', fontsize=15)
ax2.set_ylabel('ping(ms)', fontsize=15)
```

Out[245]:

```
Text(0, 0.5, 'ping(ms)')
```

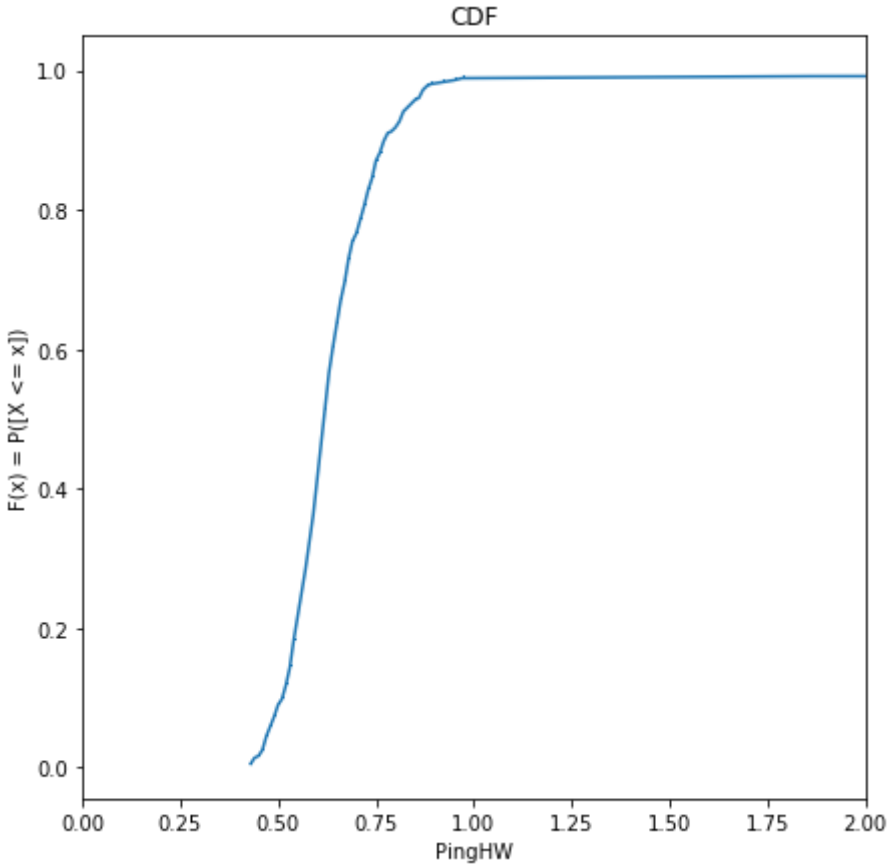


In [285]:

```
x1 = np.sort(df4['Ping'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x1)

y1 = func(x1)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('PingHW')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.set_xlim(0,2)
ax1.plot(x1, y1, marker = ',')
```

Out[285]:

```
[<matplotlib.lines.Line2D at 0x1dba9e7df98>]
```

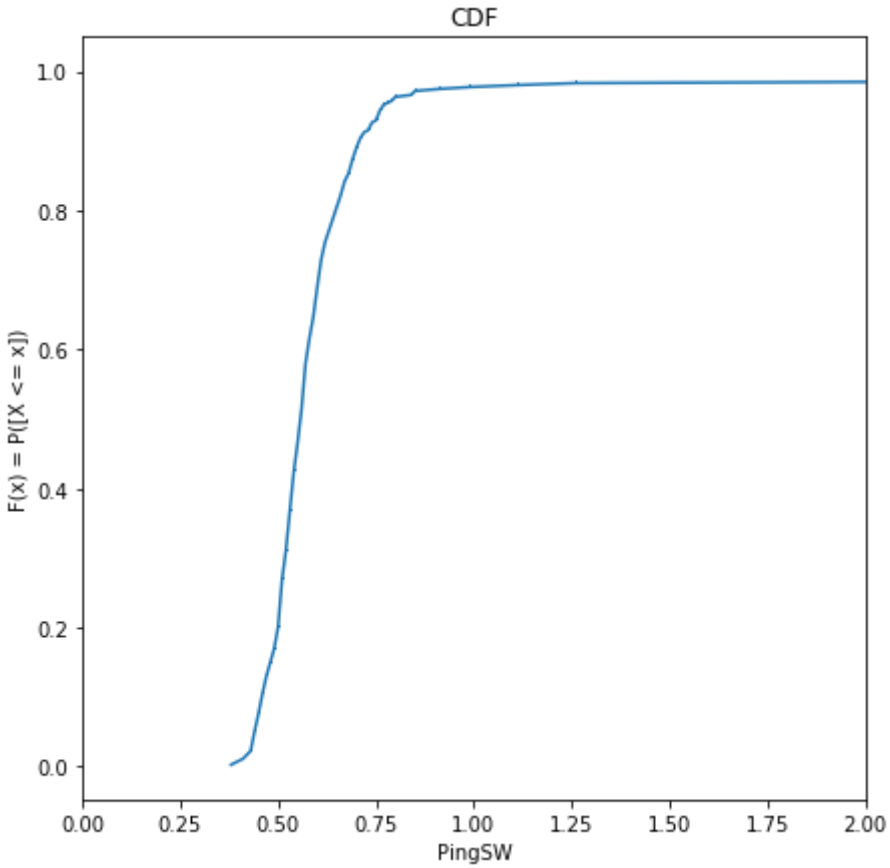


In [286]:

```
x2 = np.sort(df5['Ping'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x2)
y2 = func(x2)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('PingSW')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.set_xlim(0,2)
ax1.plot(x2, y2, marker = ',')
```

Out[286]:

```
[<matplotlib.lines.Line2D at 0x1dbab332ac8>]
```

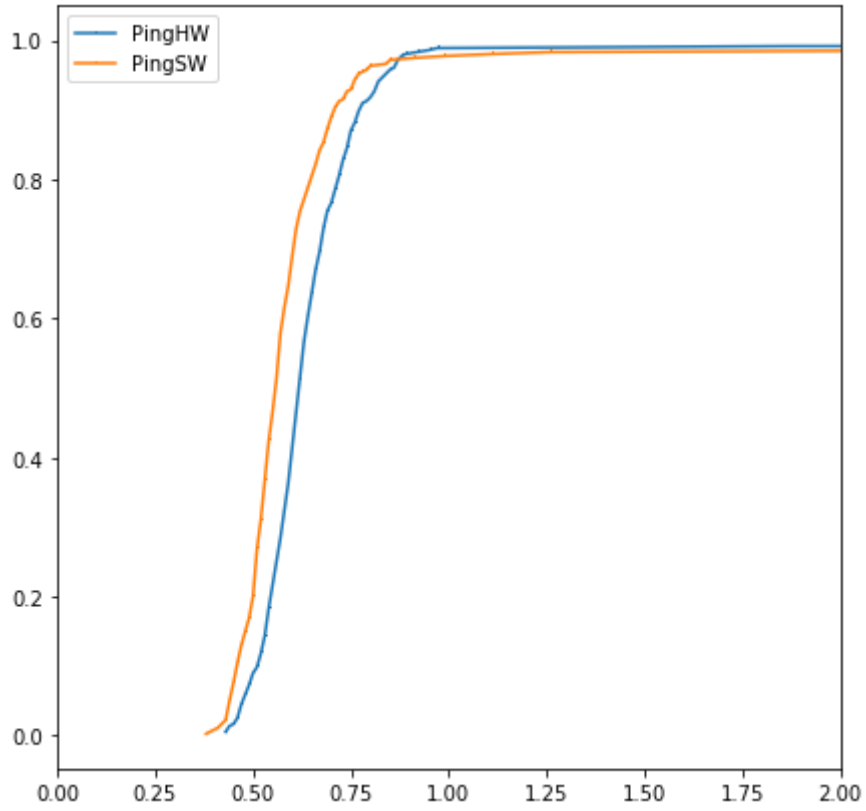


In [292]:

```
fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.plot(x1, y1, marker = ',', label='PingHW')
ax1.plot(x2, y2, marker = ',', label='PingSW')
ax1.set_xlim(0,2)
ax1.legend()
```

Out[292]:

```
<matplotlib.legend.Legend at 0x1dbac584be0>
```



Jitter

In [301]:

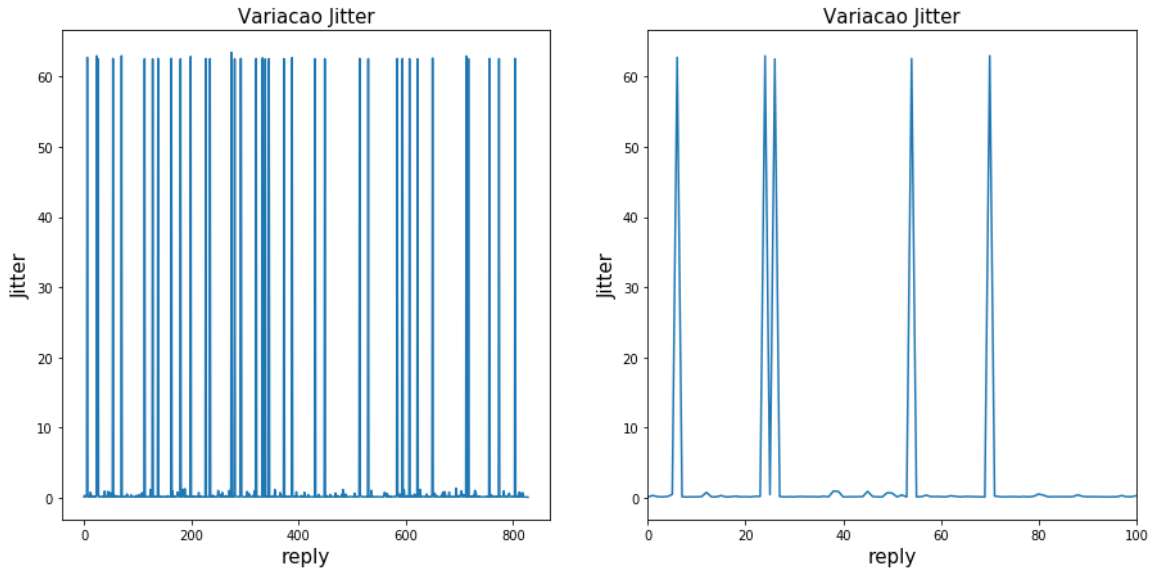
```
dfServer = df3[df3.IpOri == '10.1.0.1']
```

In [318]:

```
y = dfServer.Jitter.to_numpy()

fig, (ax1, ax2) = plt.subplots(1,2,figsize = (15,7))

ax1.plot(y)
ax1.set_title("Variacao Jitter", fontsize=15)
ax1.set_xlabel('reply', fontsize=15)
ax1.set_ylabel('Jitter', fontsize=15)

ax2.plot(y)
ax2.set_title("Variacao Jitter", fontsize=15)
ax2.set_xlabel('reply', fontsize=15)
ax2.set_ylabel('Jitter', fontsize=15)
ax2.set_xlim(0, 100)
```

Out[318]:

```
(0, 100)
```

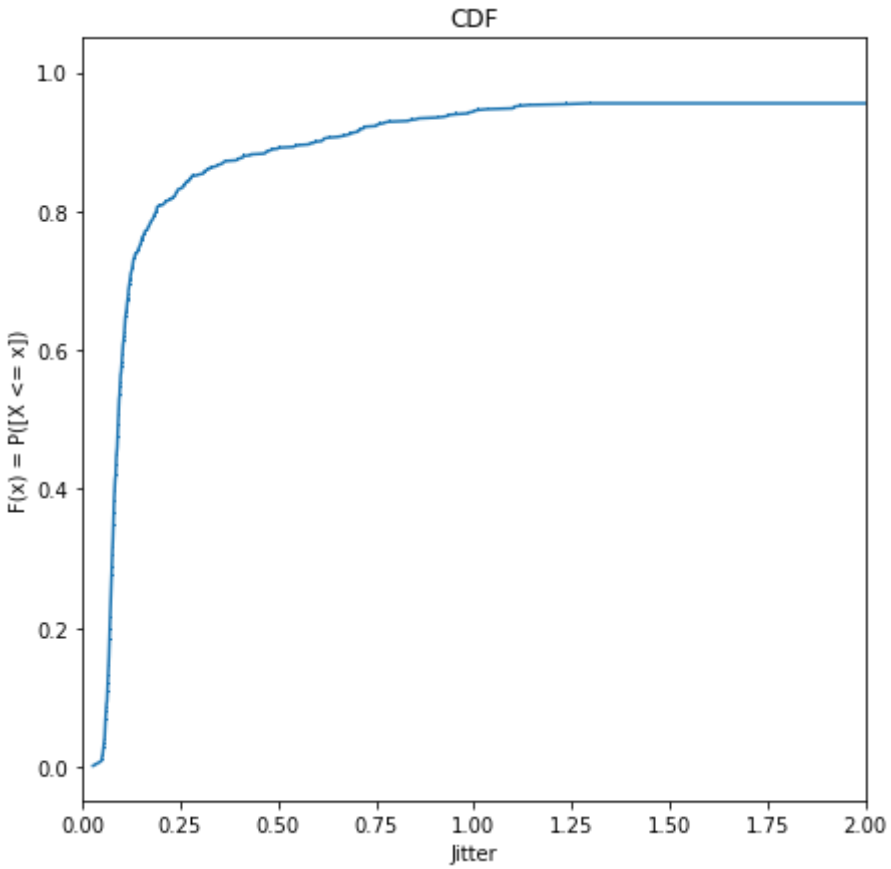


In [322]:

```
x1 = np.sort(dfServer['Jitter'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x1)

y1 = func(x1)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('Jitter')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.set_xlim(0,2)
ax1.plot(x1, y1, marker = ',')
```

Out[322]:

```
[<matplotlib.lines.Line2D at 0x1dbaf076f60>]
```



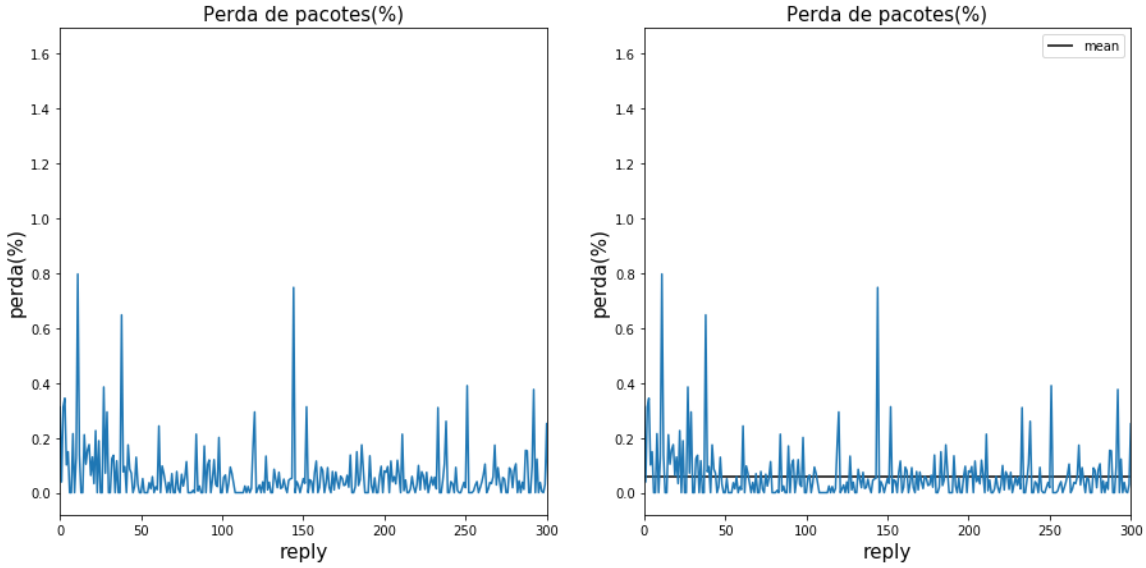
% Perda de Pacotes

In [344]:

```
y = dfServer["%Perda"].to_numpy()

fig, (ax1, ax2) = plt.subplots(1,2,figsize = (15,7))

ax1.plot(y)
ax1.set_title("Perda de pacotes(%)", fontsize=15)
ax1.set_xlabel('reply', fontsize=15)
ax1.set_ylabel('perda(%)', fontsize=15)
ax1.set_xlim(0, 300)

ax2.hlines(dfServer["%Perda"].mean(), 0 ,300, label='mean')
ax2.plot(y)
ax2.set_title("Perda de pacotes(%)", fontsize=15)
ax2.set_xlabel('reply', fontsize=15)
ax2.set_ylabel('perda(%)', fontsize=15)
ax2.set_xlim(0, 300)
ax2.legend()
```

Out[344]:

```
<matplotlib.legend.Legend at 0x1dbb03cdba8>
```

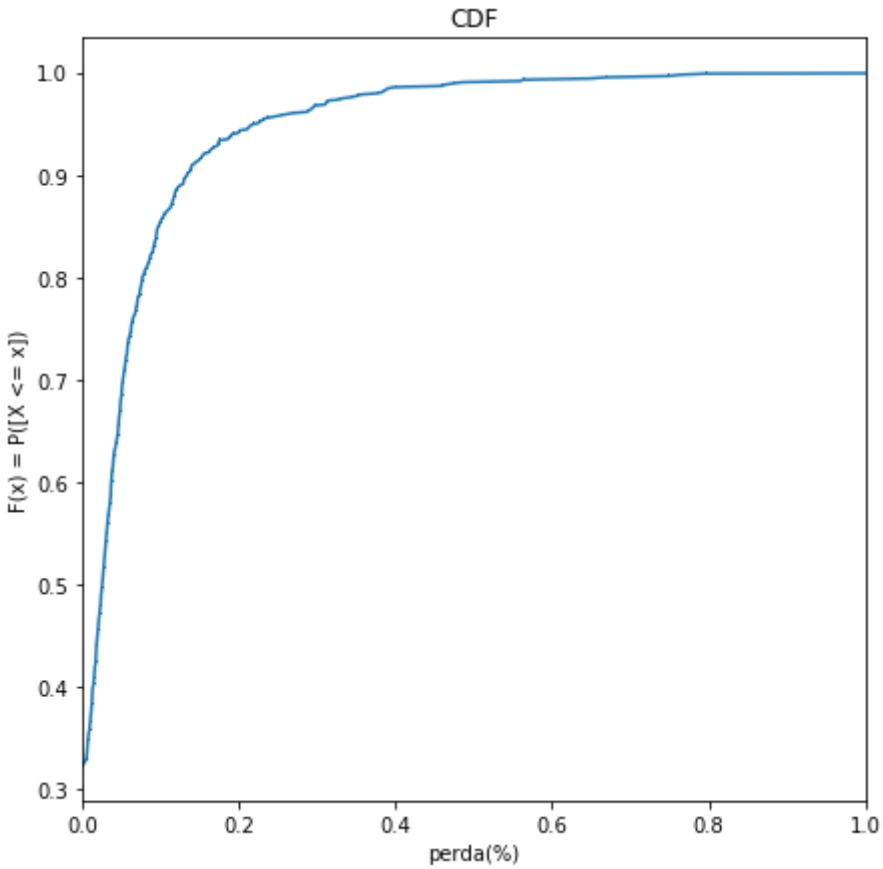


In [346]:

```
x1 = np.sort(dfServer['%Perda'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x1)

y1 = func(x1)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('perda(%)')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.set_xlim(0,1)
ax1.plot(x1, y1, marker = ',')
```

Out[346]:

```
[<matplotlib.lines.Line2D at 0x1dbb04d7550>]
```



Método SW (Análises)

In [347]:

```
df1, df2, df3 = read_files(9, "sw")
df4 = convert_ping("pingHW", 9)
df5 = convert_ping("pingSW", 9)
```

In [348]:

```
df1.head()
```

Out[348]:

	Tempo	Switch	RegrasInstaladas	RegrasAceitas	VezesBloqueado	BytesEnviados
0	11	HW	2	2	0	0
1	11	SW	19	19	0	5515776
2	16	HW	17	17	0	15525216
3	16	SW	50	50	0	5515776
4	21	HW	17	17	0	47727792

In [349]:

```
df2.head()
```

Out[349]:

	Inicio(seg)	Duracao(seg)	Banda(Kbps)	PCli	PServ
0	0.251040	78.275461	40.053267	18501	14001
1	0.314698	76.136369	1318.216400	18502	14002
2	0.361013	23.815972	2730.020501	18503	14003
3	1.127516	13.392282	2208.581846	18504	14004
4	1.546171	5.375085	1371.062812	18505	14005

In [350]:

```
df3.head()
```

Out[350]:

	Timestamp	IpOri	PortaOri	IpDest	PortaDest	?	Tempo	BytesEnv	Banda(bps)
0	20191016104657	10.1.0.2	18505	10.1.0.1	14005	3	0.0-5.4	945210	1403891
1	20191016104658	10.1.0.1	14005	10.1.0.2	18505	3	0.0-5.4	940800	1402749
2	20191016104705	10.1.0.2	18504	10.1.0.1	14004	3	0.0-13.4	3786720	2260984
3	20191016104705	10.1.0.1	14004	10.1.0.2	18504	3	0.0-13.4	3782310	2261699
4	20191016104706	10.1.0.2	18526	10.1.0.1	14026	3	0.0-9.1	4712820	4145131

In [351]:

```
df4.head()
```

Out[351]:

	Ping
0	55.30
1	1.09
2	0.69
3	0.79
4	0.79

In [352]:

```
df5.head()
```

Out[352]:

	Ping
0	55.11
1	1.19
2	0.65
3	0.54
4	0.65

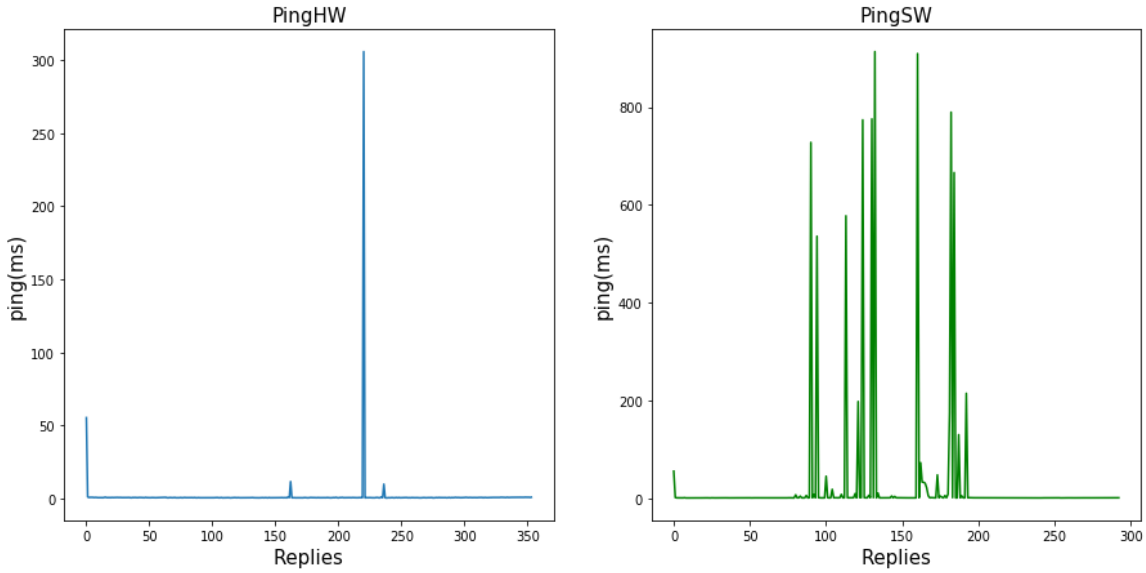
Pings (Referentes a HW e SW)

In [353]:

```
y = df4.Ping.to_numpy()
y2 = df5.Ping.to_numpy()

fig, (ax1, ax2) = plt.subplots(1,2,figsize = (15,7))

ax1.plot(y)
ax1.set_title("PingHW", fontsize=15)
ax1.set_xlabel('Replies', fontsize=15)
ax1.set_ylabel('ping(ms)', fontsize=15)
ax2.plot(y2, color='green', linestyle='-')
ax2.set_title("PingSW", fontsize=15)
ax2.set_xlabel('Replies', fontsize=15)
ax2.set_ylabel('ping(ms)', fontsize=15)
```

Out[353]:

```
Text(0, 0.5, 'ping(ms)')
```

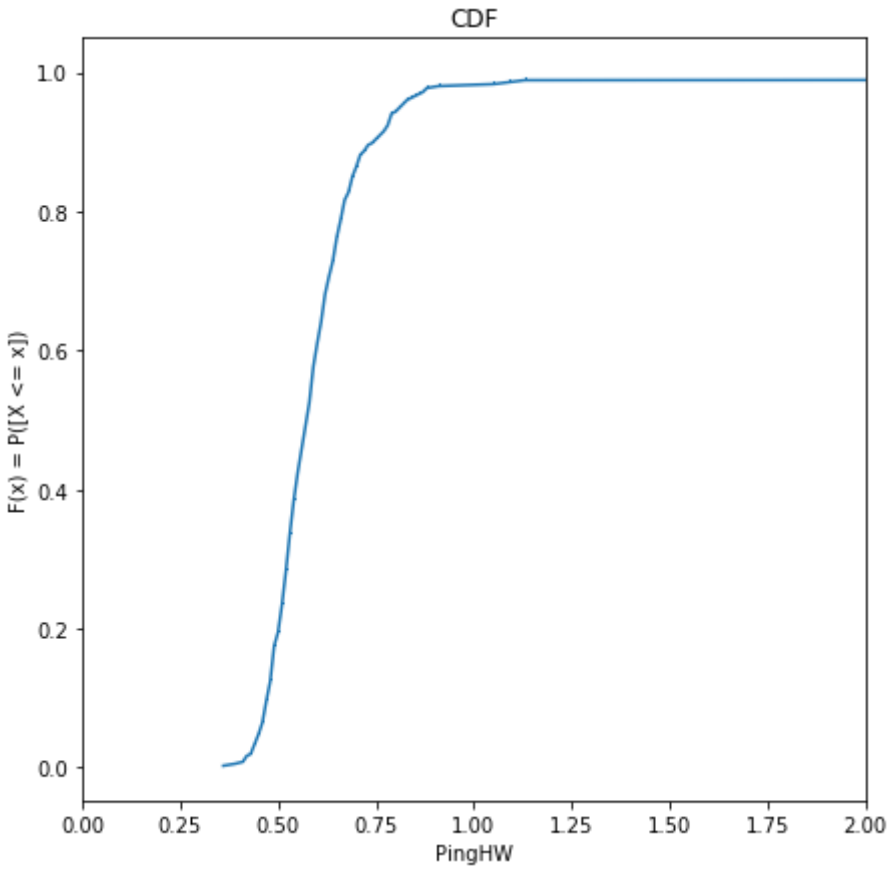


In [354]:

```
x1 = np.sort(df4['Ping'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x1)

y1 = func(x1)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('PingHW')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.set_xlim(0,2)
ax1.plot(x1, y1, marker = ',')
```

Out[354]:

```
[<matplotlib.lines.Line2D at 0x1dbb0368898>]
```

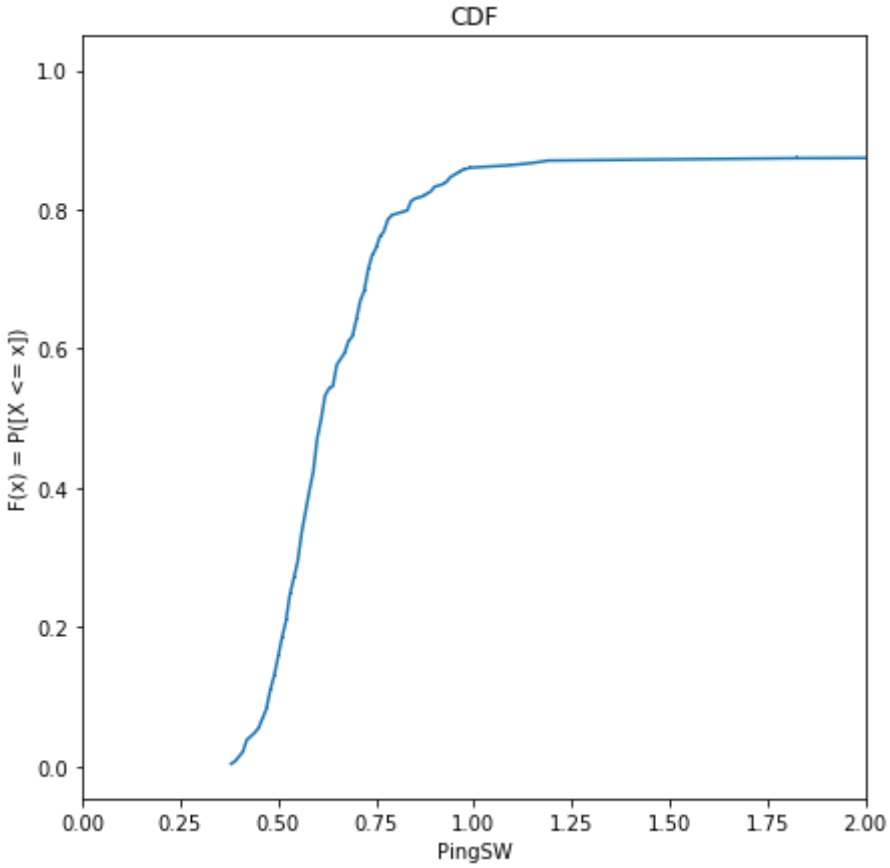


In [355]:

```
x2 = np.sort(df5['Ping'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x2)
y2 = func(x2)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('PingSW')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.set_xlim(0,2)
ax1.plot(x2, y2, marker = ',')
```

Out[355]:

```
[<matplotlib.lines.Line2D at 0x1dbafd9aa20>]
```

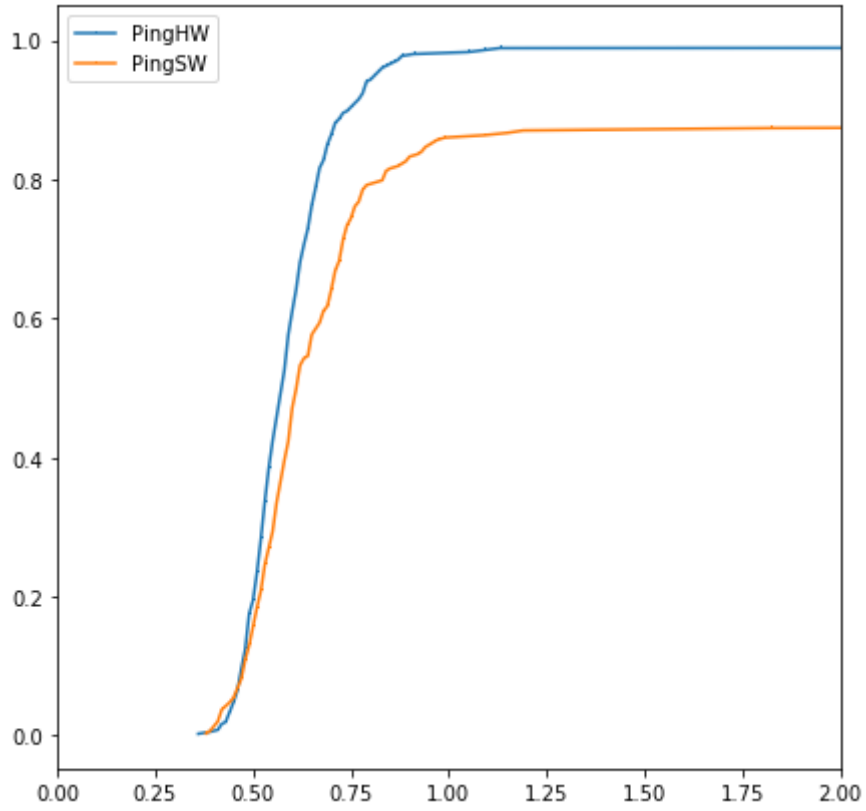


In [356]:

```
fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.plot(x1, y1, marker = ',', label='PingHW')
ax1.plot(x2, y2, marker = ',', label='PingSW')
ax1.set_xlim(0,2)
ax1.legend()
```

Out[356]:

```
<matplotlib.legend.Legend at 0x1dbafd00908>
```



Jitter

In [357]:

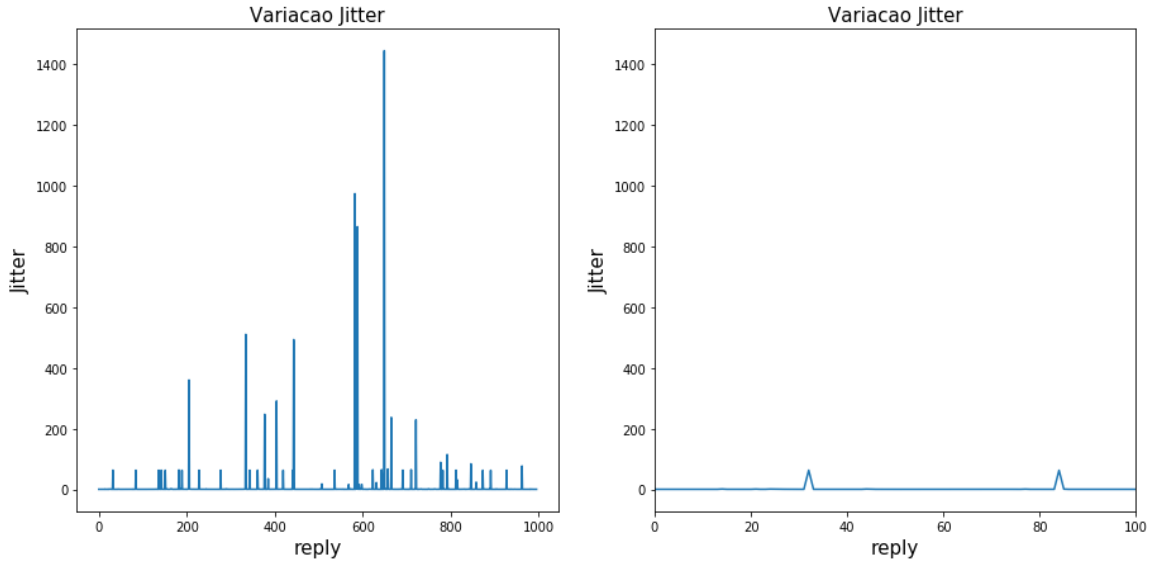
```
dfServer = df3[df3.IpOri == '10.1.0.1']
```

In [358]:

```
y = dfServer.Jitter.to_numpy()

fig, (ax1, ax2) = plt.subplots(1,2,figsize = (15,7))

ax1.plot(y)
ax1.set_title("Variacao Jitter", fontsize=15)
ax1.set_xlabel('reply', fontsize=15)
ax1.set_ylabel('Jitter', fontsize=15)

ax2.plot(y)
ax2.set_title("Variacao Jitter", fontsize=15)
ax2.set_xlabel('reply', fontsize=15)
ax2.set_ylabel('Jitter', fontsize=15)
ax2.set_xlim(0, 100)
```

Out[358]:

```
(0, 100)
```

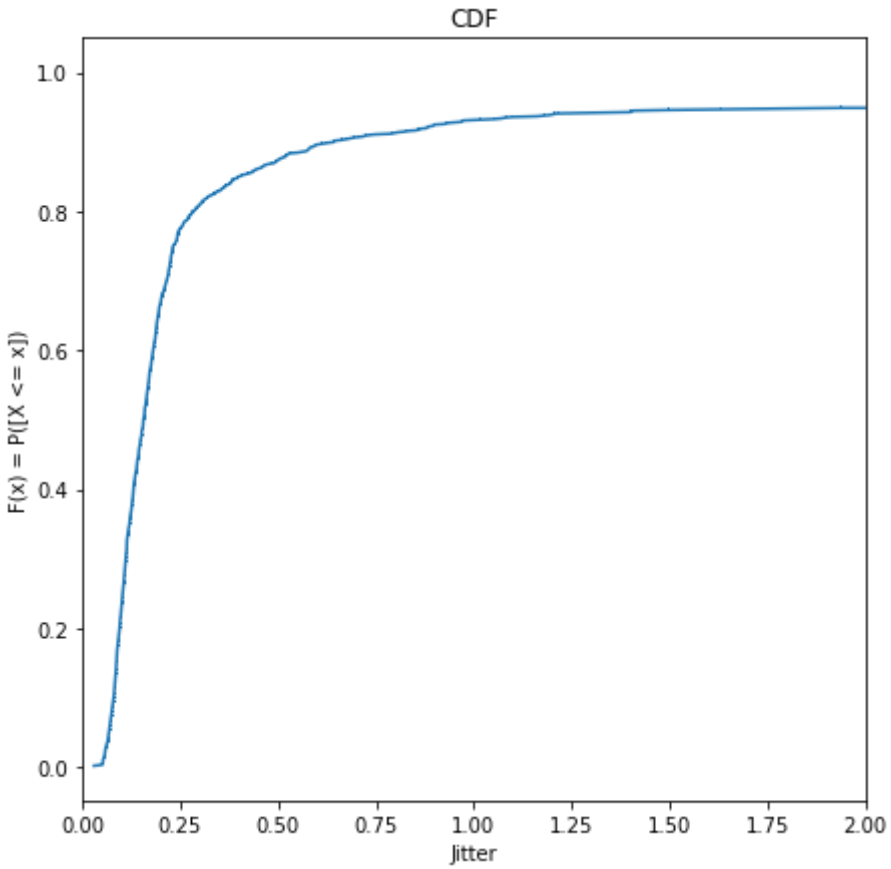


In [359]:

```
x1 = np.sort(dfServer['Jitter'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x1)

y1 = func(x1)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('Jitter')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.set_xlim(0,2)
ax1.plot(x1, y1, marker = ',')
```

Out[359]:

```
[<matplotlib.lines.Line2D at 0x1dbaf1ba5c0>]
```



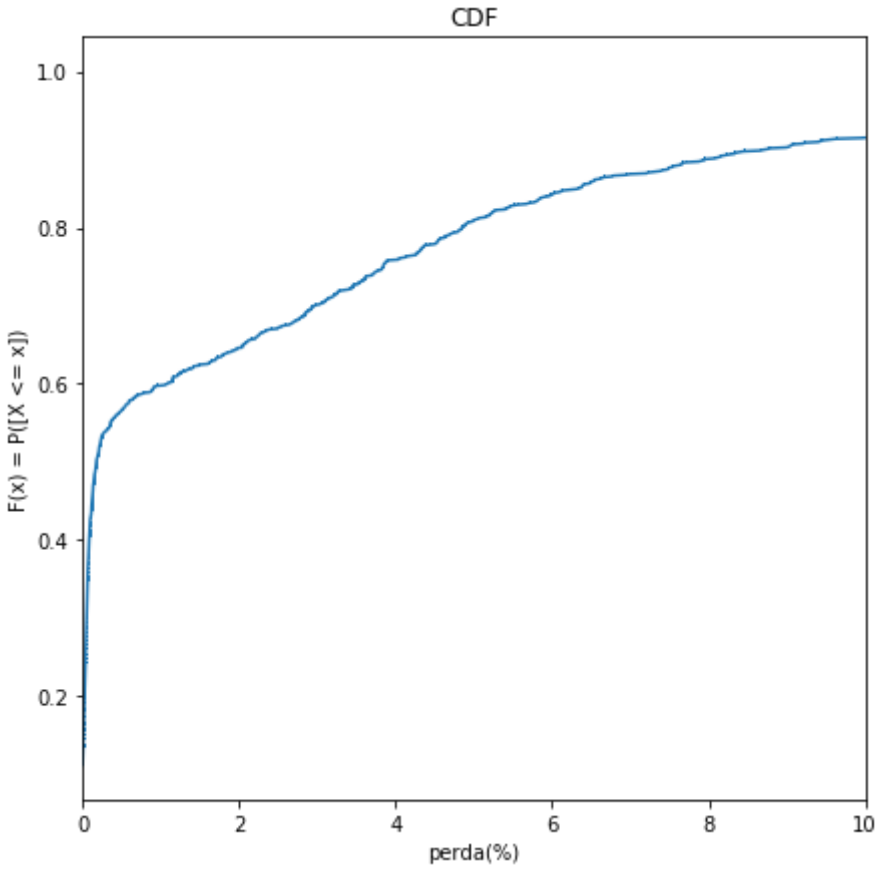
% Perda de Pacotes

In [363]:

```
x1 = np.sort(dfServer['%Perda'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x1)

y1 = func(x1)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('perda(%)')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.set_xlim(0,10)
ax1.plot(x1, y1, marker = ',')




x2 = np.sort(dfServerParImpar['%Perda'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x2)
y2 = func(x2)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('perda(%)')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.plot(x1, y1, marker = ',', label='MetodoSW')
ax1.plot(x2, y2, marker = ',', label='MetodoParImpar')
ax1.set_xlim(0,0.5)
ax1.legend()
```

Out[363]:

```
[<matplotlib.lines.Line2D at 0x1dba90a06d8>]
```

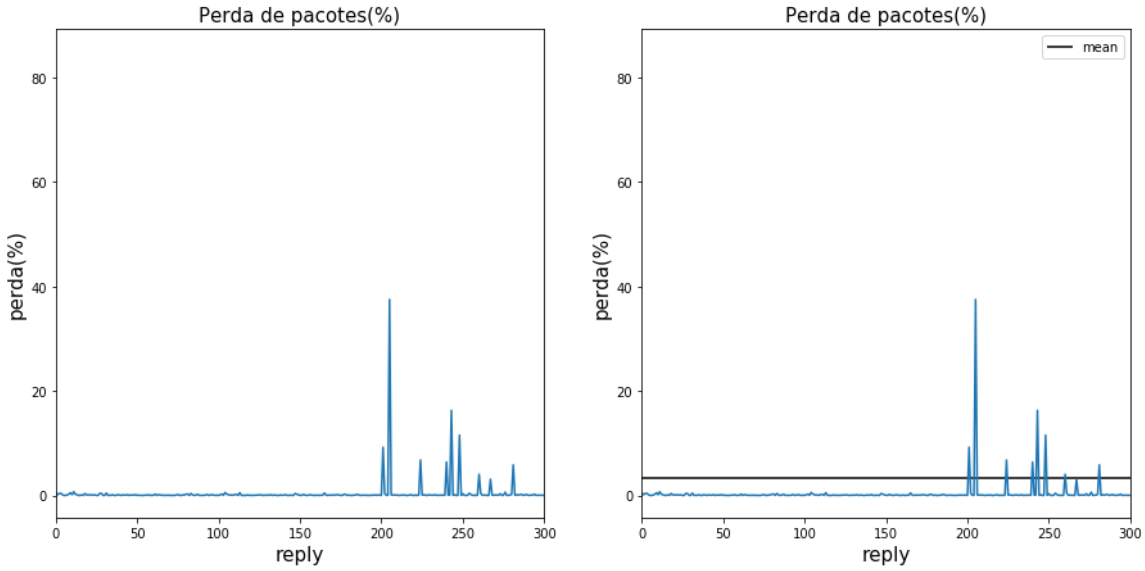


In [360]:

```
y = dfServer["%Perda"].to_numpy()

fig, (ax1, ax2) = plt.subplots(1,2,figsize = (15,7))

ax1.plot(y)
ax1.set_title("Perda de pacotes(%)", fontsize=15)
ax1.set_xlabel('reply', fontsize=15)
ax1.set_ylabel('perda(%)', fontsize=15)
ax1.set_xlim(0, 300)

ax2.hlines(dfServer["%Perda"].mean(), 0 ,300, label='mean')
ax2.plot(y)
ax2.set_title("Perda de pacotes(%)", fontsize=15)
ax2.set_xlabel('reply', fontsize=15)
ax2.set_ylabel('perda(%)', fontsize=15)
ax2.set_xlim(0, 300)
ax2.legend()
```

Out[360]:

```
<matplotlib.legend.Legend at 0x1dba88f5fd0>
```



Comparações entre os dois métodos

In [379]:

```
df1, df2, df3 = read_files(9, "sw")
df4, df5, df6 = read_files(10,"parimpar")
```

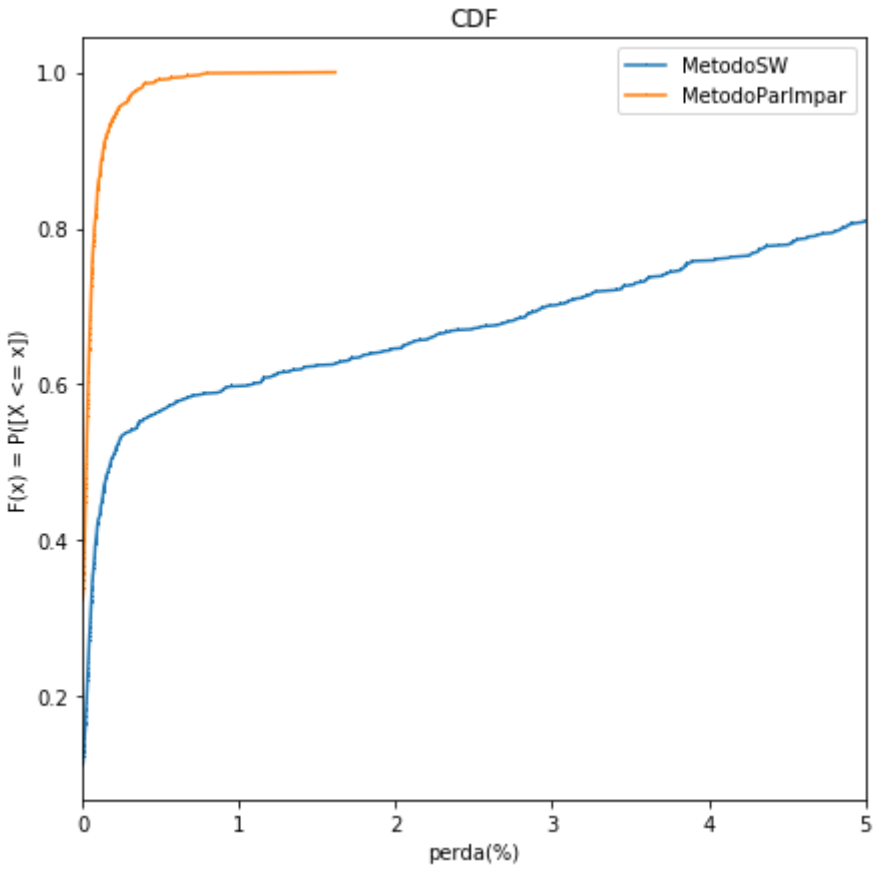
% de Perda de Pacotes

In [387]:

```
dfServerSW = df3[df3.IpOri == '10.1.0.1']
dfServerParImpar = df6[df6.IpOri == '10.1.0.1']

x1 = np.sort(dfServerSW['%Perda'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x1)
y1 = func(x1)

x2 = np.sort(dfServerParImpar['%Perda'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x2)
y2 = func(x2)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('perda(%)')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.plot(x1, y1, marker = ',', label='MetodoSW')
ax1.plot(x2, y2, marker = ',', label='MetodoParImpar')
ax1.set_xlim(0,5)
ax1.legend()
```

Out[387]:

```
<matplotlib.legend.Legend at 0x1dbafc08f98>
```



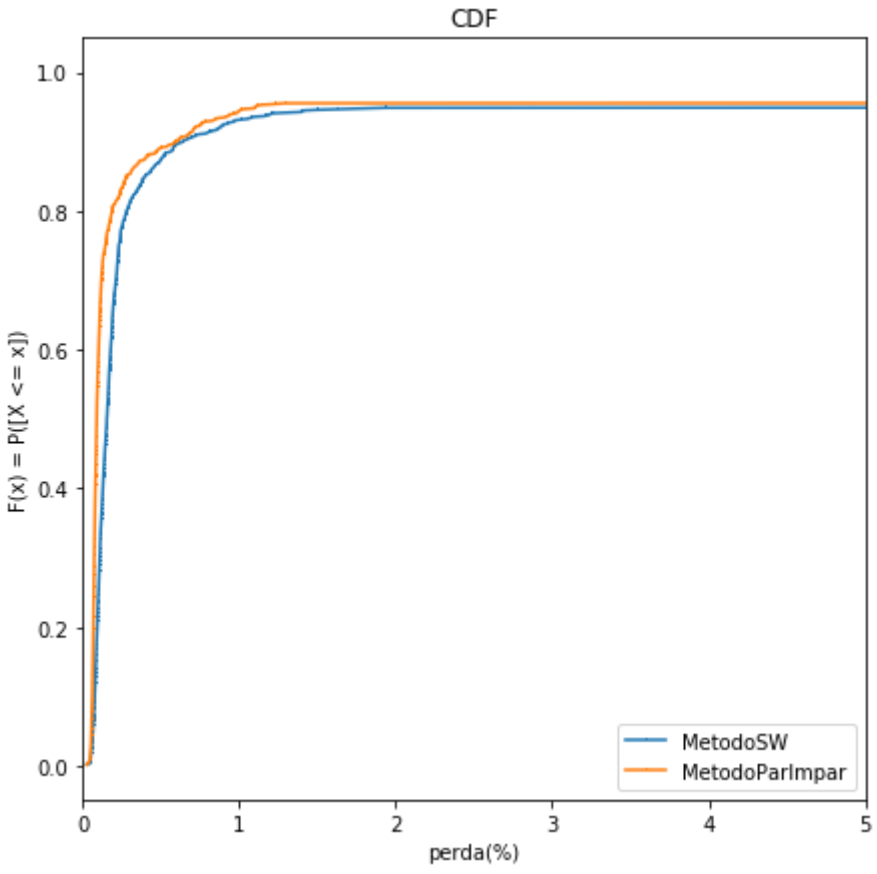
Jitter

In [388]:

```
x1 = np.sort(dfServerSW['Jitter'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x1)
y1 = func(x1)

x2 = np.sort(dfServerParImpar['Jitter'])

func = statsmodels.api.distributions.empirical_distribution.ECDF(x2)
y2 = func(x2)

fig, ax1 = plt.subplots(1,1,figsize = (7,7))

ax1.set_title('CDF')
ax1.set_xlabel('perda(%)')
ax1.set_ylabel('F(x) = P([X <= x])')
ax1.plot(x1, y1, marker = ',', label='MetodoSW')
ax1.plot(x2, y2, marker = ',', label='MetodoParImpar')
ax1.set_xlim(0,5)
ax1.legend()
```

Out[388]:

```
<matplotlib.legend.Legend at 0x1dbacc359b0>
```



In [ ]: