In [1]:

```
import pandas as pd
from sklearn import preprocessing
```

In [2]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

In [3]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:

```
from sklearn.decomposition import PCA
```

In [5]:

```
train_df = pd.read_csv("wat-all.csv")
```

In [6]:

```
train_df.head()
```

Out[6]:

	time	router	outport	inport	packet_address	packet_type	flit_id	flit_type	vnet	vc	src_ni
0	7	0	2	0	0x1dc0	0	0	3	2	8	0
1	7	1	1	0	0xecf40	0	0	3	2	8	1
2	7	0	2	0	0x1dc0	0	0	3	2	8	0
3	11	5	1	3	0xecf40	0	0	3	2	8	1
4	11	1	2	4	0x1dc0	0	0	3	2	8	0

In [7]:

```
train_X = train_df.drop(columns=['packet_address','time','target'])
```

In [8]:

```
train_X
```

Out[8]:

	router	outport	inport	packet_type	flit_id	flit_type	vnet	vc	src_ni	src_router	dst_n
0	0	2	0	0	0	3	2	8	0	0	2
1	1	1	0	0	0	3	2	8	1	1	2
2	0	2	0	0	0	3	2	8	0	0	2
3	5	1	3	0	0	3	2	8	1	1	2
4	1	2	4	0	0	3	2	8	0	0	2
...	...	...	...	...	...	...	...	...	...	...	.
504035	8	3	2	1	2	1	4	16	27	11	
504036	8	3	2	1	3	1	4	16	27	11	
504037	9	4	2	1	4	2	4	16	27	11	
504038	4	3	1	1	0	0	4	16	27	11	
504039	4	3	1	1	1	1	4	16	27	11	

504040 rows × 14 columns

In [9]:

```
x = train_X.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
train_X = pd.DataFrame(x_scaled)
train_X
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	
0	0.000000	0.50	0.00	0.000000	0.00	1.000000	0.0	0.000000	0.000000	0.000000	0.741
1	0.066667	0.25	0.00	0.000000	0.00	1.000000	0.0	0.000000	0.032258	0.066667	0.935
2	0.000000	0.50	0.00	0.000000	0.00	1.000000	0.0	0.000000	0.000000	0.000000	0.741
3	0.333333	0.25	0.75	0.000000	0.00	1.000000	0.0	0.000000	0.032258	0.066667	0.935
4	0.066667	0.50	1.00	0.000000	0.00	1.000000	0.0	0.000000	0.000000	0.000000	0.741
...	...	...	...	...	...	...	...	...	...	...	
504035	0.533333	0.75	0.50	0.333333	0.50	0.333333	1.0	0.727273	0.870968	0.733333	0.000
504036	0.533333	0.75	0.50	0.333333	0.75	0.333333	1.0	0.727273	0.870968	0.733333	0.000
504037	0.600000	1.00	0.50	0.333333	1.00	0.666667	1.0	0.727273	0.870968	0.733333	0.000
504038	0.266667	0.75	0.25	0.333333	0.00	0.000000	1.0	0.727273	0.870968	0.733333	0.000
504039	0.266667	0.75	0.25	0.333333	0.25	0.333333	1.0	0.727273	0.870968	0.733333	0.000

504040 rows × 14 columns

In [10]:

```
corr_df = pd.concat([train_X, train_df[['target']]], axis = 1)
corr_df.corr()
```

Out[10]:

	0	1	2	3	4	5	6	7
0	1.000000	0.311228	0.047617	-0.042331	2.395434e-03	-0.003780	0.178998	0.169018
1	0.311228	1.000000	-0.359564	-0.108256	6.052237e-03	-0.009565	0.482558	0.516584
2	0.047617	-0.359564	1.000000	0.111575	-6.101837e-03	0.009632	-0.488682	-0.578135
3	-0.042331	-0.108256	0.111575	1.000000	8.386949e-03	-0.013272	-0.395071	-0.430033
4	0.002395	0.006052	-0.006102	0.008387	1.000000e+00	0.079060	0.107364	0.083671
5	-0.003780	-0.009565	0.009632	-0.013272	7.906039e-02	1.000000	-0.169743	-0.132280
6	0.178998	0.482558	-0.488682	-0.395071	1.073640e-01	-0.169743	1.000000	0.977135
7	0.169018	0.516584	-0.578135	-0.430033	8.367094e-02	-0.132280	0.977135	1.000000
8	0.297251	0.509907	-0.490140	-0.211123	1.244433e-02	-0.019663	0.930521	0.902311
9	0.425807	0.459022	-0.403628	-0.175386	1.037869e-02	-0.016404	0.768846	0.735741
10	-0.110032	-0.485092	0.515266	0.210071	-1.247589e-02	0.019717	-0.931435	-0.914002
11	0.016156	-0.401589	0.464305	0.173351	-1.048132e-02	0.016574	-0.773016	-0.765335
12	0.025157	-0.016820	0.016370	0.045874	-2.262283e-05	-0.000024	-0.041155	-0.045096
13	-0.039097	0.010323	0.000941	-0.010380	6.820495e-07	0.000011	0.009316	0.011594
target	0.002485	0.003407	0.007999	0.007953	-3.565070e-06	0.000019	-0.007137	-0.008493

In [11]:

```
train_Y = train_df['target']
```

In [12]:

```
train_Y
```

Out[12]:

```
0         1
1         0
2         0
3         0
4         0
         ..
504035    1
504036    1
504037    1
504038    1
504039    1
Name: target, Length: 504040, dtype: int64
```

In [13]:

```
model = Sequential()
```

In [14]:

```
n_cols = train_X.shape[1]
n_cols
```

Out[14]:

```
14
```

In [15]:

```
model.add(Dense(32, activation='relu', input_shape=(n_cols,)))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [16]:

```
model.compile(optimizer='sgd', loss='mean_squared_error', metrics=['accuracy'])
```

In [19]:

```
early_stopping_monitor = EarlyStopping(patience=20)
```

In [20]:

```
model.fit(train_X, train_Y, epochs=30, validation_split=0.2, callbacks=[early_stopping_mon
itor])
```

```
Train on 403232 samples, validate on 100808 samples
Epoch 1/30
403232/403232 [==============================] - 32s 79us/step - loss: 0.1968
- accuracy: 0.7014 - val_loss: 0.2724 - val_accuracy: 0.6364
Epoch 2/30
403232/403232 [==============================] - 29s 73us/step - loss: 0.1866
- accuracy: 0.7189 - val_loss: 0.2878 - val_accuracy: 0.6350
Epoch 3/30
403232/403232 [==============================] - 33s 81us/step - loss: 0.1657
- accuracy: 0.7659 - val_loss: 0.3117 - val_accuracy: 0.6365
Epoch 4/30
403232/403232 [==============================] - 32s 79us/step - loss: 0.1481
- accuracy: 0.7999 - val_loss: 0.3226 - val_accuracy: 0.6462
Epoch 5/30
403232/403232 [==============================] - 32s 80us/step - loss: 0.1396
- accuracy: 0.8094 - val_loss: 0.3271 - val_accuracy: 0.6473
Epoch 6/30
403232/403232 [==============================] - 33s 82us/step - loss: 0.1330
- accuracy: 0.8145 - val_loss: 0.3299 - val_accuracy: 0.6400
Epoch 7/30
403232/403232 [==============================] - 32s 81us/step - loss: 0.1268
- accuracy: 0.8212 - val_loss: 0.3229 - val_accuracy: 0.6442
Epoch 8/30
403232/403232 [==============================] - 30s 75us/step - loss: 0.1218
- accuracy: 0.8286 - val_loss: 0.3295 - val_accuracy: 0.6291
Epoch 9/30
403232/403232 [==============================] - 30s 74us/step - loss: 0.1180
- accuracy: 0.8362 - val_loss: 0.3193 - val_accuracy: 0.6362
Epoch 10/30
403232/403232 [==============================] - 32s 80us/step - loss: 0.1152
- accuracy: 0.8428 - val_loss: 0.3069 - val_accuracy: 0.6494
Epoch 11/30
403232/403232 [==============================] - 30s 74us/step - loss: 0.1129
- accuracy: 0.8474 - val_loss: 0.3122 - val_accuracy: 0.6461
Epoch 12/30
403232/403232 [==============================] - 32s 80us/step - loss: 0.1113
- accuracy: 0.8510 - val_loss: 0.3158 - val_accuracy: 0.6403
Epoch 13/30
403232/403232 [==============================] - 35s 87us/step - loss: 0.1100
- accuracy: 0.8536 - val_loss: 0.3159 - val_accuracy: 0.6318
Epoch 14/30
403232/403232 [==============================] - 39s 97us/step - loss: 0.1088
- accuracy: 0.8562 - val_loss: 0.3191 - val_accuracy: 0.6340
Epoch 15/30
403232/403232 [==============================] - 39s 96us/step - loss: 0.1079
- accuracy: 0.8577 - val_loss: 0.3314 - val_accuracy: 0.6100
Epoch 16/30
403232/403232 [==============================] - 35s 87us/step - loss: 0.1071
- accuracy: 0.8588 - val_loss: 0.3206 - val_accuracy: 0.5975
Epoch 17/30
403232/403232 [==============================] - 38s 94us/step - loss: 0.1064
- accuracy: 0.8602 - val_loss: 0.3134 - val_accuracy: 0.6117
Epoch 18/30
403232/403232 [==============================] - 38s 94us/step - loss: 0.1056
- accuracy: 0.8623 - val_loss: 0.3121 - val_accuracy: 0.6122
Epoch 19/30
403232/403232 [==============================] - 33s 82us/step - loss: 0.1050
```

```
- accuracy: 0.8628 - val_loss: 0.3070 - val_accuracy: 0.6190
Epoch 20/30
403232/403232 [==============================] - 34s 84us/step - loss: 0.1041
- accuracy: 0.8649 - val_loss: 0.3065 - val_accuracy: 0.6323
Epoch 21/30
403232/403232 [==============================] - 35s 87us/step - loss: 0.1033
- accuracy: 0.8667 - val_loss: 0.3164 - val_accuracy: 0.6183
```

Out[20]:

```
<keras.callbacks.callbacks.History at 0x1e4bbb9f108>
```

In [21]:

```
pred = model.predict(train_X)
```

In [22]:

```
for i in range(100):
    print("%s, %s" % (pred[i], train_Y[i]))
```

```
[0.96757007], 1
[6.503344e-05], 0
[0.00134527], 0
[3.0323137e-05], 0
[0.00108236], 0
[0.96757007], 1
[0.00103919], 0
[2.867967e-05], 0
[0.96757007], 1
[3.2732438e-05], 0
[0.00127532], 0
[0.96757007], 1
[0.96757007], 1
[0.00209097], 0
[9.48787e-06], 0
[7.352932e-06], 0
[9.022571e-06], 0
[1.14067025e-05], 0
[6.413268e-06], 0
[0.00476331], 0
[0.96757007], 1
[0.96757007], 1
[4.6144555e-06], 0
[0.00491383], 0
[0.96757007], 1
[0.00571129], 0
[6.5895533e-06], 0
[8.580115e-06], 0
[9.410068e-06], 0
[0.00766705], 0
[0.96757007], 1
[0.00412699], 0
[5.8366554e-06], 0
[0.96757007], 1
[0.00424717], 0
[4.402364e-06], 0
[0.96757007], 1
[5.997057e-06], 0
[0.00433523], 0
[0.96757007], 1
[8.564059e-06], 0
[0.00442514], 0
[6.7707356e-06], 0
[0.00846835], 0
[0.96757007], 1
[0.96757007], 1
[0.00416923], 0
[0.96757007], 1
[5.426958e-06], 0
[0.00429059], 0
[3.4649554e-06], 0
[0.96757007], 1
[3.4520262e-06], 0
[0.00437955], 0
[0.96757007], 1
[0.96757007], 1
[0.96757007], 1
```

```
[4.9130117e-06], 0
[0.0044704], 0
[0.00455396], 0
[6.1619485e-06], 0
[0.96757007], 1
[0.00417187], 0
[0.00429333], 0
[0.96757007], 1
[0.00438237], 0
[0.96757007], 1
[3.5350074e-06], 0
[0.0046005], 0
[0.00447322], 0
[0.96757007], 1
[0.96757007], 1
[0.96757007], 1
[0.00463658], 0
[0.00469225], 0
[0.96757007], 1
[0.96757007], 1
[0.00478956], 0
[0.96757007], 1
[0.96757007], 1
[0.00460343], 0
[0.00488884], 0
[0.96757007], 1
[0.00503102], 0
[0.00059647], 0
[0.00048022], 0
[0.00052362], 0
[2.0777521e-05], 0
[0.68491536], 1
[8.531151e-06], 0
[0.4921332], 1
[0.00076693], 0
[1.1650405e-05], 0
[0.36676726], 1
[0.00138224], 0
[0.00112295], 0
[0.00134702], 0
[0.00149425], 0
[0.00087948], 0
[0.554235], 1
```

In [23]:

```
pca = PCA(n_components = 3)
```

In [24]:

```
pca.fit(train_X)
```

Out[24]:

```
PCA(copy=True, iterated_power='auto', n_components=3, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

In [25]:

```
principal_components = pca.transform(train_X)
principal_components
```

Out[25]:

```
array([[-0.70976983, -0.30256686,  0.5625034 ],
       [-0.92900455, -0.27575105,  0.59820052],
       [-0.71039771, -0.30538929,  0.51702793],
       ...,
       [ 1.09518458,  0.52966508, -0.66684601],
       [ 1.06930523, -0.50242002, -0.17903966],
       [ 1.06586005, -0.20939815, -0.23565379]])
```

In [26]:

```
pca.explained_variance_ratio_
```

Out[26]:

```
array([0.52095204, 0.07945799, 0.0733298 ])
```

In [27]:

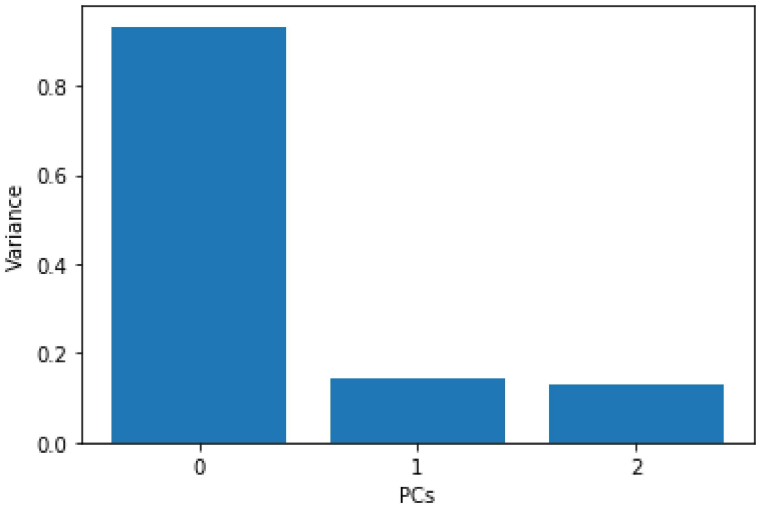
```
features = range(pca.n_components_)
```

In [28]:

```
plt.bar(features, pca.explained_variance_)
plt.xticks(features)
plt.xlabel("PCs")
plt.ylabel("Variance")
```

Out[28]:

```
Text(0, 0.5, 'Variance')
```



In [29]:

```
principal_df = pd.DataFrame(data = principal_components , columns = ['pc 1', 'pc 2','pc 3'
])
principal_df
```

Out[29]:

	pc 1	pc 2	pc 3
0	-0.709770	-0.302567	0.562503
1	-0.929005	-0.275751	0.598201
2	-0.710398	-0.305389	0.517028
3	-1.075973	-0.277781	0.275460
4	-0.926192	-0.287472	0.244392
...	...	...	...
504035	1.033983	0.020426	-0.465089
504036	1.038888	0.263618	-0.489906
504037	1.095185	0.529665	-0.666846
504038	1.069305	-0.502420	-0.179040
504039	1.065860	-0.209398	-0.235654

504040 rows × 3 columns

In [30]:

```
final_df = pd.concat([principal_df, train_df[['target']]], axis = 1)
final_df
```

Out[30]:

	pc 1	pc 2	pc 3	target
0	-0.709770	-0.302567	0.562503	1
1	-0.929005	-0.275751	0.598201	0
2	-0.710398	-0.305389	0.517028	0
3	-1.075973	-0.277781	0.275460	0
4	-0.926192	-0.287472	0.244392	0
...	...	...	...	...
504035	1.033983	0.020426	-0.465089	1
504036	1.038888	0.263618	-0.489906	1
504037	1.095185	0.529665	-0.666846	1
504038	1.069305	-0.502420	-0.179040	1
504039	1.065860	-0.209398	-0.235654	1

504040 rows × 4 columns

In [31]:

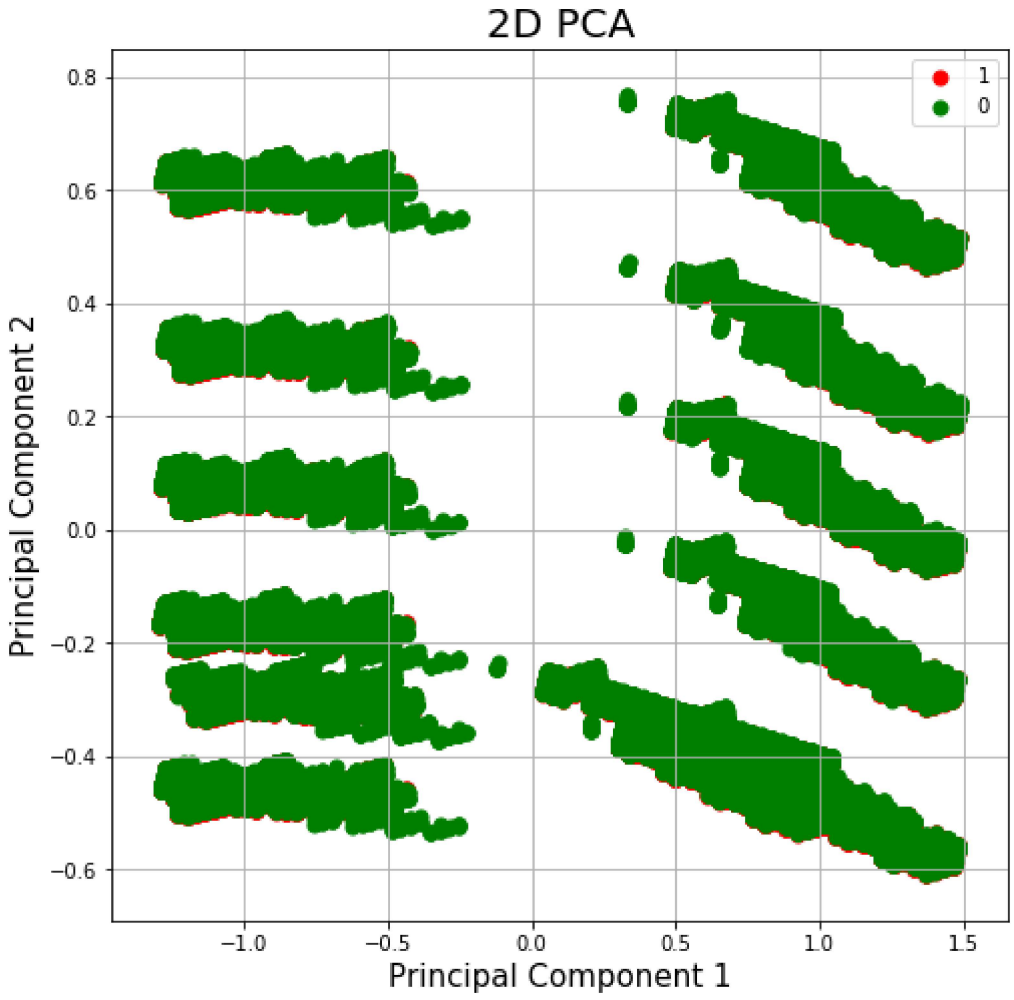
```
final_df.corr()
```

Out[31]:

	pc 1	pc 2	pc 3	target
pc 1	1.000000e+00	4.194285e-15	1.103954e-16	-0.006784
pc 2	4.194285e-15	1.000000e+00	-3.214468e-15	-0.004627
pc 3	1.103954e-16	-3.214468e-15	1.000000e+00	-0.069337
target	-6.784181e-03	-4.626786e-03	-6.933698e-02	1.000000

In [32]:

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2D PCA', fontsize = 20)
targets = [1, 0]
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = final_df['target'] == target
    ax.scatter(final_df.loc[indicesToKeep, 'pc 1']
               , final_df.loc[indicesToKeep, 'pc 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```

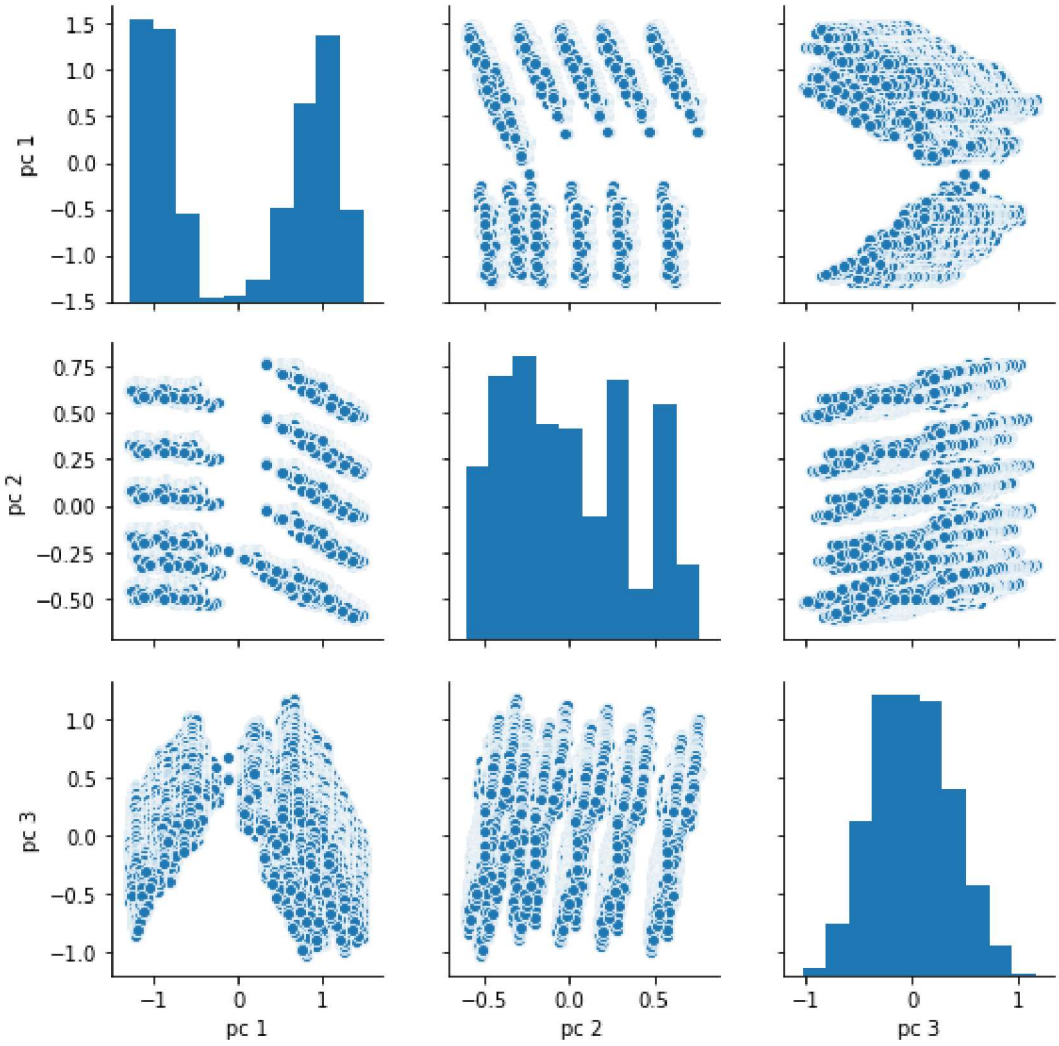


In [33]:

```
sns.pairplot(final_df.loc[:,final_df.dtypes == 'float64'])
```

Out[33]:

```
<seaborn.axisgrid.PairGrid at 0x1e4bcd90b08>
```



In [34]:

```
corr_df[corr_df.duplicated()].shape
```

Out[34]:

```
(0, 15)
```

In [35]:

```
dup_df = train_df.drop(columns=['packet_address_id','packet_address','time'])
dup_df
```

Out[35]:

	router	outport	inport	packet_type	flit_id	flit_type	vnet	vc	src_ni	src_router	dst_n
0	0	2	0	0	0	3	2	8	0	0	2
1	1	1	0	0	0	3	2	8	1	1	2
2	0	2	0	0	0	3	2	8	0	0	2
3	5	1	3	0	0	3	2	8	1	1	2
4	1	2	4	0	0	3	2	8	0	0	2
...	...	...	...	...	...	...	...	...	...	...	.
504035	8	3	2	1	2	1	4	16	27	11	
504036	8	3	2	1	3	1	4	16	27	11	
504037	9	4	2	1	4	2	4	16	27	11	
504038	4	3	1	1	0	0	4	16	27	11	
504039	4	3	1	1	1	1	4	16	27	11	

504040 rows × 14 columns

In [36]:

```
dup_df[dup_df.duplicated()].shape
```

Out[36]:

```
(0, 14)
```

In [37]:

```
dup_df[dup_df.duplicated()].count()
```

Out[37]:

```
router         0
outport        0
inport         0
packet_type    0
flit_id        0
flit_type      0
vnet           0
vc             0
src_ni         0
src_router     0
dst_ni         0
dst_router     0
enq_time       0
target         0
dtype: int64
```