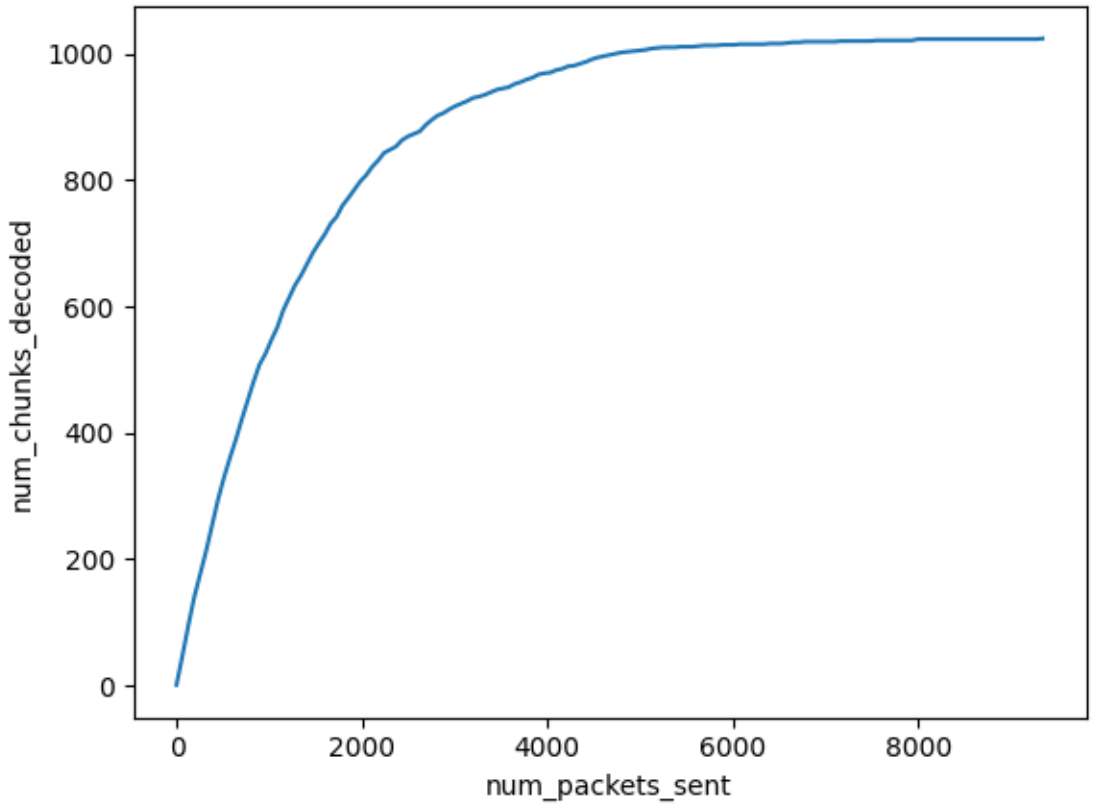
0.0.1 Task 2.2 (manually graded)

Plot the number of chunks decoded as a function of the number of packets you send. The `chunks_decoded` array should be helpful here.

In [64]:
```
sz=64
num_chunks_decoded = single_decoded
num_packets_sent = list(range(0,single_sent+sz,sz))
plt.plot(num_packets_sent, num_chunks_decoded)
plt.xlabel("num_packets_sent")
plt.ylabel("num_chunks_decoded")
plt.show()
```



0.0.2 Task 2.3 (manually graded)

Looking at the graph, we see that it gets harder and harder to find the rest as we decode more and more chunks. Does this remind you of a well known theoretical problem? Which problem is it?

Hint: Try out some small examples!

Type your answer here, replacing this text.

This reminds me of the coupon collector problem. The kid wants to collect N distinct chunks to have the racoon image. Each received packet gives information on 1 random chunk. As you collect more and more new chunks, the probability that a packet contains a new chunks greatly decreases.

0.0.3 Task 2.5 (manually graded)

In the cell below, try to send the raccoon over a channel with erasure probability 0.2 using the `double` degree distribution (don't worry about intermediate plots this time).

Comment on what happens when you try the `double` degree distribution. Can you tell why this happens?

Type your answer here, replacing this text.

In the double degree distribution, each packet holds the xor of exactly 2 distinct chunks everytime. You will unfortunately never be able to deduce the actual value of any single chunk. You must introduce the probability that some of the packets send only 1 chunk. This is a cyclical problem: suppose you know a^b, b^c, and a^c. Without knowing the value of any of a, b, or c, you won't be able to get any progress!

```
In [66]: # YOUR CODE HERE
         eps = 0.2
         ch = Channel(eps)
         tx = Transmitter(chunks, ch, 'double')
         rx = Receiver(len(chunks), ch)

         double_sent, images, double_decoded = send(tx,rx)

         print("The number of packets sent: {}".format(double_sent))

         # n_of_figures = len(images)
         # fig = plt.figure(figsize=(8, 3*n_of_figures))

         # for i in range(n_of_figures):
         #     fig.add_subplot(n_of_figures,1,i+1)
         #     plt.imshow(images[i], cmap = cm.Greys_r)
         # plt.show()
```

```
Ending transmission because too many packets have been sent. This may be caused by a bug in receive_pac
The number of packets sent: 20481
```

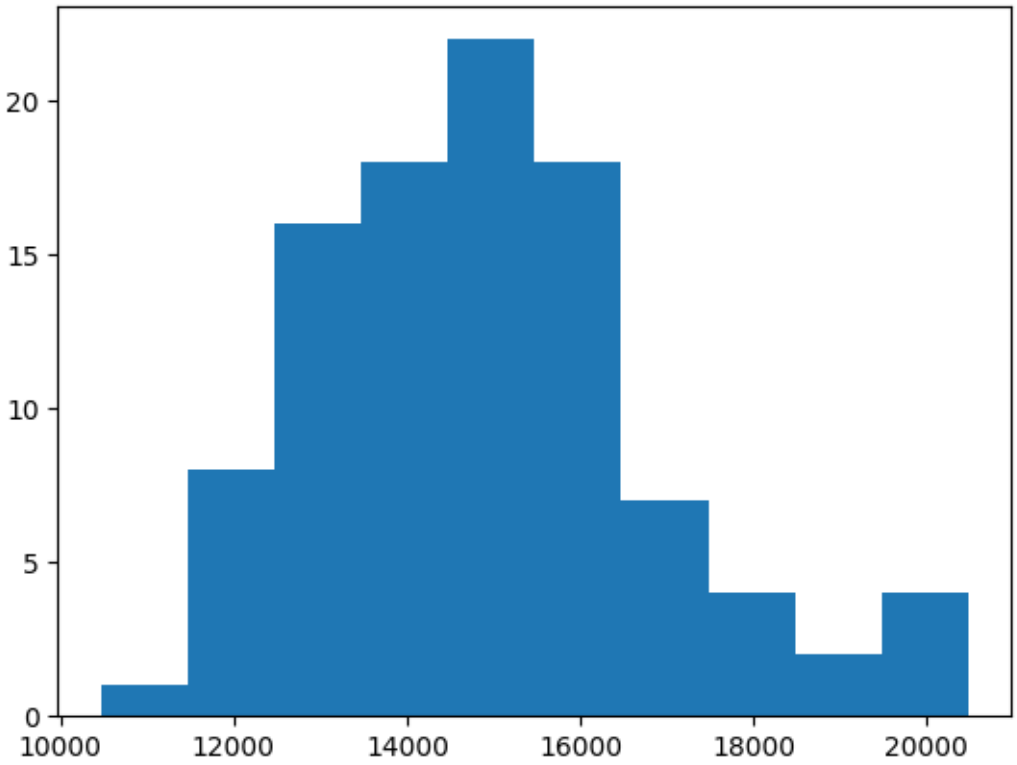
0.0.4 Task 3.1 (manually graded)

Implement the `uniform` degree distribution in the `Transmitter` class. The `uniform` degree distribution will randomly pick a degree d uniformly at random from 1 to 5 (why might it be a good idea to limit the degree to 5?). If we have fewer than 5 chunks, we will instead pick d uniformly at random from 1 to the number of chunks.

Next, use the `uniform` degree distribution to send the raccoon over a channel with erasure probability 0.2 over multiple trials. For each trial, record the number of packets sent for the image to be decoded. Then, plot this as a histogram.

In [67]:
```
num_trials = 100  # do not change this
eps = 0.2
ch = Channel(eps)
tx = Transmitter(chunks, ch, 'uniform')

packets_required = []

for _ in range(num_trials):
    rx = Receiver(len(chunks), ch)
    uniform_sent, images, uniform_decoded = send(tx,rx)
    packets_required.append(uniform_sent)


# Plot the packets required as a histogram
plt.hist(packets_required)
plt.show()
```

```
Ending transmission because too many packets have been sent. This may be caused by a bug in receive_pacl
Ending transmission because too many packets have been sent. This may be caused by a bug in receive_pacl
Ending transmission because too many packets have been sent. This may be caused by a bug in receive_pacl
```



You can run the cell below to check your implementation.

```
In [68]: grader.check("q3.1")
```

```
Out[68]: q3.1 results: All test cases passed!
```

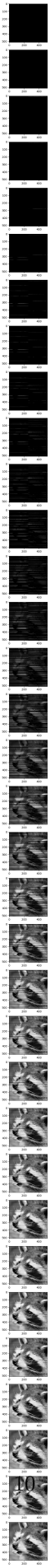
0.0.5 Task 3.3 (manually graded)

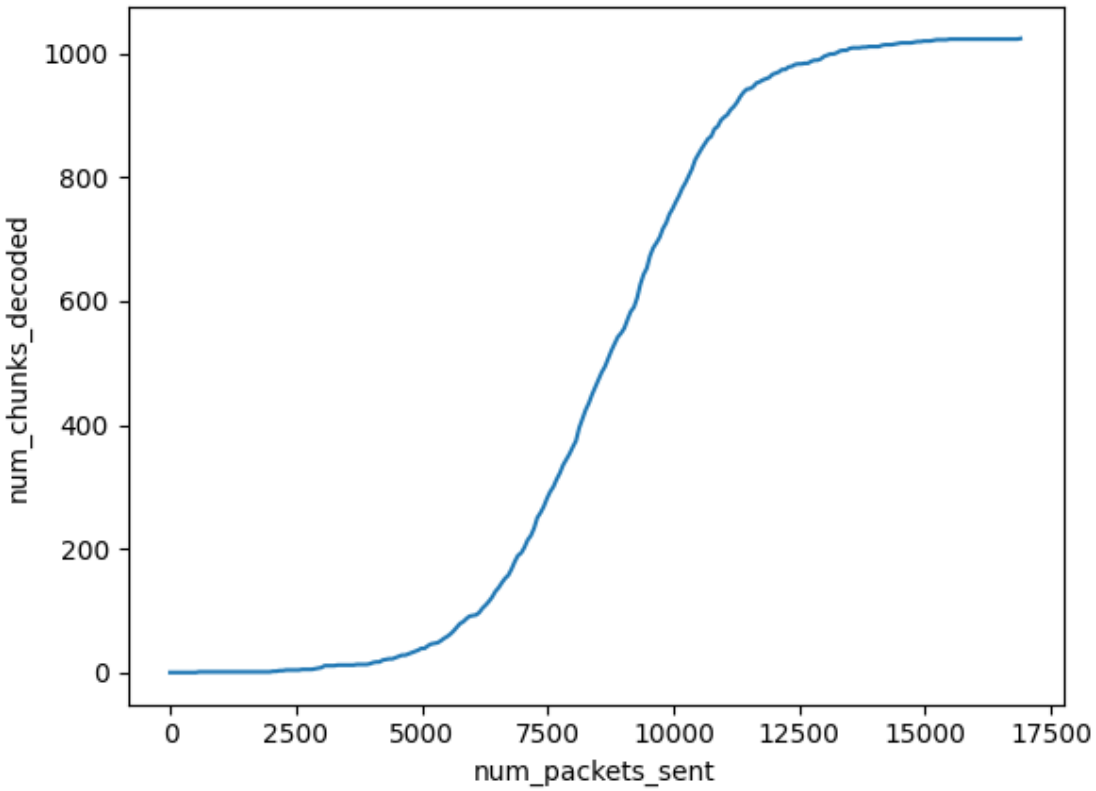
Using the `ideal_soliton` degree distribution, send the image over a channel with erasure probability 0.2. Plot the number of packets decoded against the number of packets transmitted.

Type your answer here, replacing this text.

In [70]:
```
eps = 0.2
ch = Channel(eps)
tx = Transmitter(chunks, ch, 'ideal_soliton')
rx = Receiver(len(chunks), ch)

sol_sent, images, sol_decoded = send(tx,rx)

print("The number of packets sent: {}".format(sol_sent))

n_of_figures = len(images)
fig = plt.figure(figsize=(8, 3*n_of_figures))

for i in range(n_of_figures):
    fig.add_subplot(n_of_figures,1,i+1)
    plt.imshow(images[i], cmap = cm.Greys_r)
plt.show()

sz=64
num_chunks_decoded = sol_decoded
num_packets_sent = list(range(0,sol_sent+sz,sz))
plt.plot(num_packets_sent, num_chunks_decoded)
plt.xlabel("num_packets_sent")
plt.ylabel("num_chunks_decoded")
plt.show()
```

```
The number of packets sent: 16874
```




num_chunks_decoded
num_packets_sent
0
200
400
600
800
1000
0
2500
5000
7500
10000
12500
15000
17500