

Cindy Tai 31462986 Section 002
Case 1: Name Found

The image shows a VS Code editor with a Python script named `dnsclient.py` and its terminal output.

Python Script (`dnsclient.py`):

```
1 #!/usr/bin/env python3
2
3 import socket
4 import struct
5
6 # Send DNS request to server
7
8 def send_request(question):
9     # Create a socket
10     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11
12     # Connect to the server
13     clientSocket.connect((address, port))
14
15     # Create a message
16     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
17
18     # Send the message
19     clientSocket.sendto(request, (address, port))
20
21     # Attempt to receive DNS response
22     try:
23         dResponse, dAddress = clientSocket.recvfrom(1024)
24
25         response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
26
27         rcode = int(response[1])
28         msgID = int(response[2])
29         qLen = int(response[3])
30         aLen = int(response[4])
31         question2 = response[5].decode()
32         answer = response[6].decode()
33
34         # Print the response
35         print("Received Response from " + dAddress + ", " + str(len(dResponse)) + " bytes")
36         print("Return Code: " + str(rcode))
37         print("Message ID: " + str(msgID))
38         print("Question Length: " + str(qLen) + " bytes")
39         print("Answer Length: " + str(aLen) + " bytes")
40         print("Question: " + question2 + "\n")
41         print("Answer: " + answer + "\n")
42
43     except socket.timeout:
44         print("Timeout")
45
46 # Main function
47 def main():
48     # Create a socket
49     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
50
51     # Connect to the server
52     clientSocket.connect((address, port))
53
54     # Create a message
55     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
56
57     # Send the message
58     clientSocket.sendto(request, (address, port))
59
60     # Attempt to receive DNS response
61     try:
62         dResponse, dAddress = clientSocket.recvfrom(1024)
63
64         response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
65
66         rcode = int(response[1])
67         msgID = int(response[2])
68         qLen = int(response[3])
69         aLen = int(response[4])
70         question2 = response[5].decode()
71         answer = response[6].decode()
72
73         # Print the response
74         print("Received Response from " + dAddress + ", " + str(len(dResponse)) + " bytes")
75         print("Return Code: " + str(rcode))
76         print("Message ID: " + str(msgID))
77         print("Question Length: " + str(qLen) + " bytes")
78         print("Answer Length: " + str(aLen) + " bytes")
79         print("Question: " + question2 + "\n")
80         print("Answer: " + answer + "\n")
81
82     except socket.timeout:
83         print("Timeout")
84
85 # Main function
86 def main():
87     # Create a socket
88     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
89
90     # Connect to the server
91     clientSocket.connect((address, port))
92
93     # Create a message
94     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
95
96     # Send the message
97     clientSocket.sendto(request, (address, port))
98
99     # Attempt to receive DNS response
100    try:
101        dResponse, dAddress = clientSocket.recvfrom(1024)
102
103        response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
104
105        rcode = int(response[1])
106        msgID = int(response[2])
107        qLen = int(response[3])
108        aLen = int(response[4])
109        question2 = response[5].decode()
110        answer = response[6].decode()
111
112        # Print the response
113        print("Received Response from " + dAddress + ", " + str(len(dResponse)) + " bytes")
114        print("Return Code: " + str(rcode))
115        print("Message ID: " + str(msgID))
116        print("Question Length: " + str(qLen) + " bytes")
117        print("Answer Length: " + str(aLen) + " bytes")
118        print("Question: " + question2 + "\n")
119        print("Answer: " + answer + "\n")
120
121    except socket.timeout:
122        print("Timeout")
123
124 # Main function
125 def main():
126     # Create a socket
127     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
128
129     # Connect to the server
130     clientSocket.connect((address, port))
131
132     # Create a message
133     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
134
135     # Send the message
136     clientSocket.sendto(request, (address, port))
137
138     # Attempt to receive DNS response
139     try:
140         dResponse, dAddress = clientSocket.recvfrom(1024)
141
142         response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
143
144         rcode = int(response[1])
145         msgID = int(response[2])
146         qLen = int(response[3])
147         aLen = int(response[4])
148         question2 = response[5].decode()
149         answer = response[6].decode()
150
151         # Print the response
152         print("Received Response from " + dAddress + ", " + str(len(dResponse)) + " bytes")
153         print("Return Code: " + str(rcode))
154         print("Message ID: " + str(msgID))
155         print("Question Length: " + str(qLen) + " bytes")
156         print("Answer Length: " + str(aLen) + " bytes")
157         print("Question: " + question2 + "\n")
158         print("Answer: " + answer + "\n")
159
160     except socket.timeout:
161         print("Timeout")
162
163 # Main function
164 def main():
165     # Create a socket
166     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
167
168     # Connect to the server
169     clientSocket.connect((address, port))
170
171     # Create a message
172     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
173
174     # Send the message
175     clientSocket.sendto(request, (address, port))
176
177     # Attempt to receive DNS response
178     try:
179         dResponse, dAddress = clientSocket.recvfrom(1024)
180
181         response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
182
183         rcode = int(response[1])
184         msgID = int(response[2])
185         qLen = int(response[3])
186         aLen = int(response[4])
187         question2 = response[5].decode()
188         answer = response[6].decode()
189
190         # Print the response
191         print("Received Response from " + dAddress + ", " + str(len(dResponse)) + " bytes")
192         print("Return Code: " + str(rcode))
193         print("Message ID: " + str(msgID))
194         print("Question Length: " + str(qLen) + " bytes")
195         print("Answer Length: " + str(aLen) + " bytes")
196         print("Question: " + question2 + "\n")
197         print("Answer: " + answer + "\n")
198
199     except socket.timeout:
200         print("Timeout")
201
202 # Main function
203 def main():
204     # Create a socket
205     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
206
207     # Connect to the server
208     clientSocket.connect((address, port))
209
210     # Create a message
211     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
212
213     # Send the message
214     clientSocket.sendto(request, (address, port))
215
216     # Attempt to receive DNS response
217     try:
218         dResponse, dAddress = clientSocket.recvfrom(1024)
219
220         response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
221
222         rcode = int(response[1])
223         msgID = int(response[2])
224         qLen = int(response[3])
225         aLen = int(response[4])
226         question2 = response[5].decode()
227         answer = response[6].decode()
228
229         # Print the response
230         print("Received Response from " + dAddress + ", " + str(len(dResponse)) + " bytes")
231         print("Return Code: " + str(rcode))
232         print("Message ID: " + str(msgID))
233         print("Question Length: " + str(qLen) + " bytes")
234         print("Answer Length: " + str(aLen) + " bytes")
235         print("Question: " + question2 + "\n")
236         print("Answer: " + answer + "\n")
237
238     except socket.timeout:
239         print("Timeout")
240
241 # Main function
242 def main():
243     # Create a socket
244     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
245
246     # Connect to the server
247     clientSocket.connect((address, port))
248
249     # Create a message
250     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
251
252     # Send the message
253     clientSocket.sendto(request, (address, port))
254
255     # Attempt to receive DNS response
256     try:
257         dResponse, dAddress = clientSocket.recvfrom(1024)
258
259         response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
260
261         rcode = int(response[1])
262         msgID = int(response[2])
263         qLen = int(response[3])
264         aLen = int(response[4])
265         question2 = response[5].decode()
266         answer = response[6].decode()
267
268         # Print the response
269         print("Received Response from " + dAddress + ", " + str(len(dResponse)) + " bytes")
270         print("Return Code: " + str(rcode))
271         print("Message ID: " + str(msgID))
272         print("Question Length: " + str(qLen) + " bytes")
273         print("Answer Length: " + str(aLen) + " bytes")
274         print("Question: " + question2 + "\n")
275         print("Answer: " + answer + "\n")
276
277     except socket.timeout:
278         print("Timeout")
279
280 # Main function
281 def main():
282     # Create a socket
283     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
284
285     # Connect to the server
286     clientSocket.connect((address, port))
287
288     # Create a message
289     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
290
291     # Send the message
292     clientSocket.sendto(request, (address, port))
293
294     # Attempt to receive DNS response
295     try:
296         dResponse, dAddress = clientSocket.recvfrom(1024)
297
298         response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
299
300         rcode = int(response[1])
301         msgID = int(response[2])
302         qLen = int(response[3])
303         aLen = int(response[4])
304         question2 = response[5].decode()
305         answer = response[6].decode()
306
307         # Print the response
308         print("Received Response from " + dAddress + ", " + str(len(dResponse)) + " bytes")
309         print("Return Code: " + str(rcode))
310         print("Message ID: " + str(msgID))
311         print("Question Length: " + str(qLen) + " bytes")
312         print("Answer Length: " + str(aLen) + " bytes")
313         print("Question: " + question2 + "\n")
314         print("Answer: " + answer + "\n")
315
316     except socket.timeout:
317         print("Timeout")
318
319 # Main function
320 def main():
321     # Create a socket
322     clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
323
324     # Connect to the server
325     clientSocket.connect((address, port))
326
327     # Create a message
328     request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))

```

Case 2: Name Not Found

```
dnsclient.py x  dnserver.py
dnsclient.py > ...
31 print("Question Length: " + str(len(question.encode())) + " bytes")
32 print("Answer Length: 0 bytes")
33 print("Question: " + question + "\n")
34
35 # Send DNS request to server
36 request = struct.pack('>hhhhHds' % (len(question.encode()), 1, 0, msgID, len(question), 0, question.encode()))
37 clientSocket.settimeout(1)
38 clientSocket.sendto(request, (address, port))
39 numTries += 1
40
41 # Attempt to receive DNS response
42 try:
43     dResponse, dAddress = clientSocket.recvfrom(1024)
44     response = struct.unpack('>hhhhHds' % (len(question.encode()), (len(dResponse) - len(question.encode()) - 12)), dResponse)
45     rcode = int(response[1])
46     msgID = int(response[2])
47     qlen = int(response[3])
48     alen = int(response[4])
49     question2 = response[5].decode()
50     answer = response[6].decode()
51
52 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: py, bash
ct890@LAPTOP-41P1T1GR MINGW64 /d/Users/ct890/documents/NJIT/2Spring2020/CS356/DNS
Program
$ py dnserver.py 127.0.0.1 12000
The server is ready to receive on port: 12000

ct890@LAPTOP-41P1T1GR MINGW64 /d/Users/ct890/documents/NJIT/2Spring2020/CS356/DNS
Program
$ py dnsclient.py 127.0.0.1 12000 host-not-exist.student.test A IN
Sending Request to 127.0.0.1, 12000:
Message ID: 1
Question Length: 32 bytes
Answer Length: 0 bytes
Question: host-not-exist.student.test A IN

Received Response from 127.0.0.1, 12000:
Return Code: 1 (Name does not exist)
Message ID: 1
Question Length: 32 bytes
Answer Length: 0 bytes
Question: host-not-exist.student.test A IN

ct890@LAPTOP-41P1T1GR MINGW64 /d/Users/ct890/documents/NJIT/2Spring2020/CS356/DNS
```

Case 3: No Server Response

```
dnscilent.py dnsserver.py X
dnsserver.py > ...
28 rCode = 1
29 answer = ""
30 records = []
31 ansFound = False
32
33 file = open("dns-master.txt", "r")
34 for line in file:
35     if 'A IN' in line:
36         records.append(line.rstrip())
37
38 for record in records:
39     if question in record:
40         answer = record
41         ansFound = True
42
43 if (ansFound == True):
44     rcode = 0
45
46 response = struct.pack('>hhhh{}s{}s'.format(len(question.encode()), len(answer.encode()), 2, rCode, msgID, len(question), len(answer), question.encode(), answer.encode()))
47 #serverSocket.sendto(response, address)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: py, bash

```
ct890@LAPTOP-41P1T1GR MINGW64 /d/Users/ct890/documents/NOIT/2Spring2020/CS356/DNS
Program
$ py dnsserver.py 127.0.0.1 12000
The server is ready to receive on port: 12000
[]

ct890@LAPTOP-41P1T1GR MINGW64 /d/Users/ct890/documents/NOIT/2Spring2020/CS356/DNS
Program
$ py dnsclient.py 127.0.0.1 12000 host3.student.test A IN
Sending Request to 127.0.0.1, 12000:
Message ID: 99
Question Length: 23 bytes
Answer Length: 0 bytes
Question: host3.student.test A IN

Request timed out ...
Sending Request to 127.0.0.1, 12000:
Request timed out ...
Sending Request to 127.0.0.1, 12000:
Request timed out ...
Exiting Program.
```