Assignment 1

forty-two (redux)

Due Date: Sunday, February 18, 2018 @ 11:55pm

ECE 4564 - Network Application Design

Learning Objectives

Client Sockets

- socket()
- connect()
- send()
- recv()

Server Sockets

- bind()
- listen()
- accept()

Encapsulation/Decapsulation

- Encryption/Decryption
- Checksums

Twitter API

WolframAlpha API

Text-to-Speech Translation

Sockets

Sockets

- Low level API for opening a connection to another device and exchanging data
- Socket protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)
- Protocols act as a transport mechanism: bits travel from sender to receiver
- Higher level applications need to establish what data to transmit and how data is ordered

TCP is a reliable mechanism

- TCP will ensure that the bits sent from sender to receiver arrive in order and without error or TCP will notify the user of a problem

UDP is a best effort service

- Messages may not get to destination and the sender won't be warned of a lost message

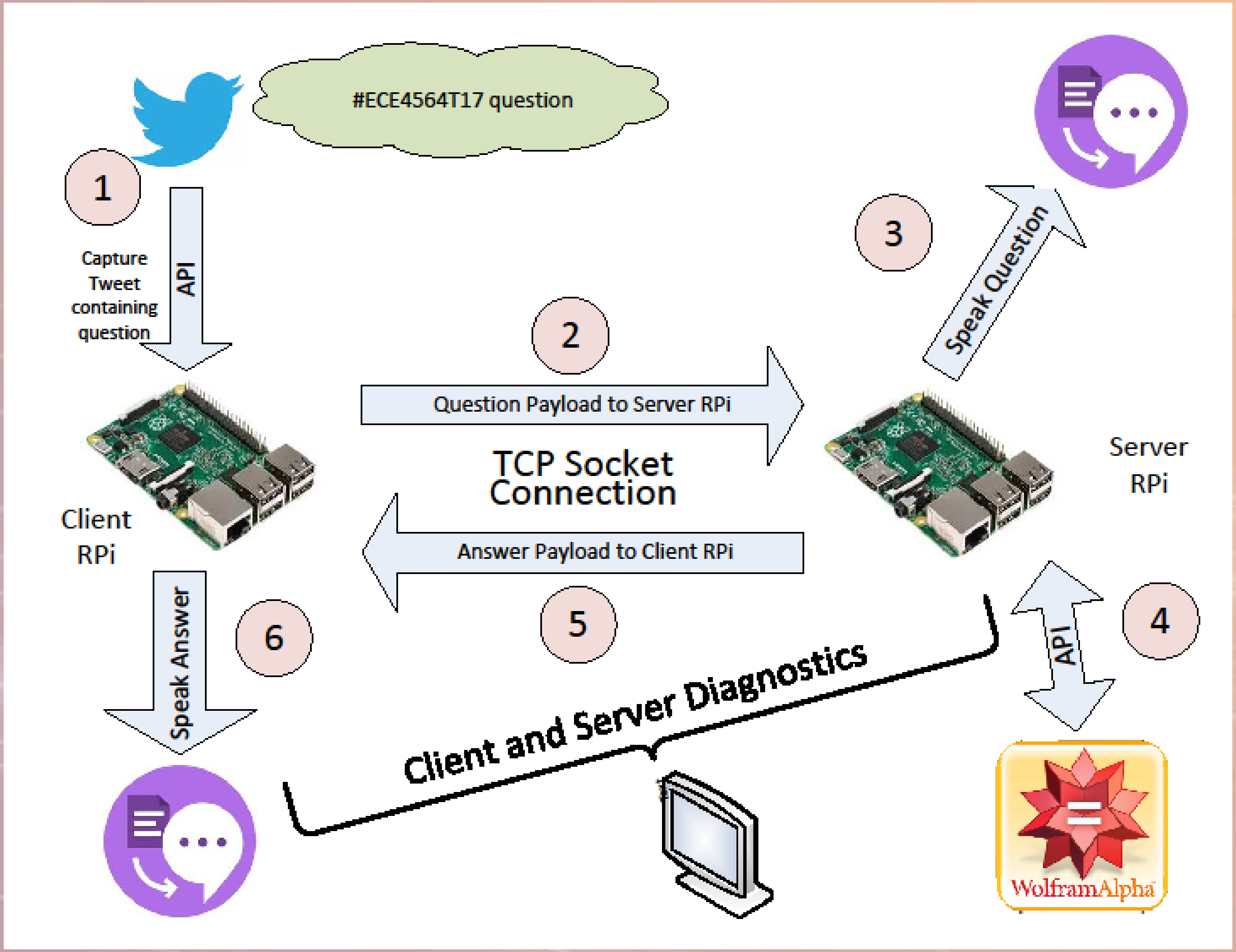
Assignment Overview

Assignment 1 is a text-based question and answer system using WolframAlpha's computational knowledge engine. Questions to your Q&A system are expressed as Twitter Tweets. The question and resulting answer are "spoken" using text-to-speech (TTS) translation.

The system uses two Rpi's following the client/server model discussed in class. The server is iterative and connection-oriented. Communication between client and server is handled via stream-oriented sockets.

The client Rpi captures the Tweet containing the question. The client builds and sends a question "payload" to the server Rpi via sockets. The server speaks the question and sends the question to the WolframAlpha engine and receives the answer. The server builds and sends an answer "payload" back to the client Rpi. The client speaks the answer and displays the answer on the attached monitor

System Overview


#ECE4564T17 question
1
API
Capture Tweet containing question
2
Question Payload to Server RPi
TCP Socket Connection
Answer Payload to Client RPi
5
3
Speak Question
Server RPi
Client RPi
4
API
6
Speak Answer
Client and Server Diagnostics
WolframAlpha

Client Requirements

Client Rpi

- Initiated on command line with parameter of server IP addr
- Captures a Twitter status object (Tweet) containing question
- Parses question from the Tweet
- Encrypts question using a Python cryptography library
- Compute checksum of encrypted question
- Build question payload (see payload slide)
- Sends question payload to server Rpi via socket interface
- Waits for answer payload
- Receives and deconstructs answer payload
  - Verify checksum
  - Decrypt answer
- “Speaks” the answer
- Displays answer on monitor

Client Initialization

```
python3 client.py -s <SERVER_IP> -p <SERVER_PORT> -z
<SOCKET_SIZE> -t “<HASHTAG>”
```

Example:

```
python3 client.py -s 192.168.1.128 -p 5803 -z 1024 -t
“#ECE4564T83”
```

Client Parameter File

Twitter API/Dev and any other API/Dev keys MUST NOT be hard coded into the source code of your client.py file. These keys must be located in a separate file and either read from or imported into client.py.

The easiest way is to create a .py file with variable names assigned to the API/Dev keys and import that file into client.py.

Status Object Format

Format of “question” Tweet:
#ECE4564TXX <question>

Example:
#ECE4564T18 What is the answer to life?

Dude, where's my car? #ECE4564T07

Where in the world #ECE4564T15 is Carmen Sandiego?

Server Requirements

Server Rpi

- Waits for question payload from client
- Receives and deconstructs question payload
  - Verify checksum
  - Decrypt question
- Sends question to WolframAlpha engine via API call
- Receives answer from WolframAlpha engine
- Builds answer payload
  - Encrypts answer
  - Generates checksum on encrypted answer
  - Assembles payload
- “Sends answer payload to client via socket call

Server Initialization

```
python3 server.py -p <SERVER_PORT> -b <BACKLOG_SIZE> -z <SOCKET_SIZE>
```

Example:

```
python3 server.py -p 5803 -b 5 -z 1024
```

Server Parameter File

Wolframalpha API/Dev and any other API/Dev keys MUST NOT be hard coded into the source code of your server.py file. These keys must be located in a separate file and either read from or imported into server.py.

The easiest way is to create a .py file with variable names assigned to the API/Dev keys and import that file into server.py.

Payloads

Question Payload (Python tuple):

- Encrypt/Decrypt key
- Question text (encrypted)
- MD5 hash of encrypted question text

Answer Payload (Python tuple):

- Answer text (encrypted)
- MD5 hash of encrypted answer text

Twitter

- Requires a Twiter account
- Applications accessing Twitter require an API key. This key is obtained by first signing up for a development account here and here. The API key can be created once logged into your account.
- Beware of API rate limits
- Implement as Streaming API
- Suggest implementing with Tweepy

WolframAlpha

Applications accessing the WolframAlpha engine require an API key. This key is obtained by first signing up for a development account here. The API key can be created once logged into your account.

Note: Access to this service is limited to 2000 non-commercial API calls per month.

Given this, each team member should set up their own development account and generate an API key. This provides 3 keys per team allowing 6000 service queries. Please to not intentionally try to exceed these limits.

Text-To-Speech

Use whatever you can find to make the raspberry pi speak.

Suggestions:

espeak
gtts

Design to a Specification

1. Encrypt/Decrypt question – symmetric encryption - Fernet
   - Plaintext = “How old is Virginia Tech?”
   - Key = b”WW5i4SzGYVA59GPst99iv_SurYdZLu9bo0d6STTUfVs=“
   - Ciphertext = b”gAAAAABX1a1StViHonShR_w75DyO_ahpQp91-g8zEpU4WpKdGQb6Lw7oROqJY7LDV_MkHhAioZ5a8BH1V-wuvJjg4YGMK6YtNWBZSe2QW00O-qZVHFc6g-8=“
2. Checksum – MD5 hash of ciphertext (the encrypted question)
   - “79cfdb00e9c61d82606c73a772584217”
3. Represent socket payload as a Python tuple where element order is:
   - Symmetric key
   - Encrypted question
   - Checksum of encrypted question
4. Pickle payload when sending over socket connection

Grading and Validation

Refer to GTA documentation posted to Canvas

Python Style

Follow style guide PEP0008 when writing and commenting your code

https://www.python.org/dev/peps/pep-0008/

Coding for all assignments in Python 3

What You Turn In

All assignments must be submitted through Canvas, no later than the due date of 2018 Feb 18 @ 23:55

Your assignment should be a single tar gz (tgz extension) which contains the following:

- All source code for this assignment
  - Python code running on client and server Rpi's
    - Client code identified as "client.py"
    - Server code identified as "server.py".
    - Client api/dev key file as "clientKeys.py"
    - Server api/dev key file as "serverKeys.py"

- Report (PDF file)

Assignment References

The official documentation from Python

- The Official Python Documentation. https://docs.python.org/3.4/index.html

A short and quick introduction to Python when programming on a Raspberry π (Ch. 2-6)

- Bradbury, Alex, and Ben Everard. Learning Python with Raspberry Pi. John Wiley & Sons, 2014.

A Python reference guide for programmers who understand software engineering/computer science concepts such as object-oriented programming*

- Martelli, Alex. Python in a Nutshell. 2nd Ed. " O'Reilly Media, Inc.", 2006. http://proquest.safaribooksonline.com/book/programming/python/0596100469

A more comprehensive book on Python that includes examples*

- Lutz, Mark. Programming python. 4th Ed. " O'Reilly Media, Inc.", 2010. http://proquest.safaribooksonline.com/book/programming/python/9781449398712

Assignment References

"Beej's Guide to Network Programming Using Internet Sockets" (PDF)

"Foundations of Python Network Programming", 2nd Ed. (Full text – VT Library)

The official WolframAlpha API
- http://products.wolframalpha.com/api/

The official Twitter Developers documentation
- https://dev.twitter.com/

A Twitter API
- Tweepy

Streaming Tweets from Twitter
- Tutorial

Assignment References

Text To Speech

- GTTS
- espeak

Academic Integrity

- For this assignment, it is expected that a team's work is their own
- The code you turn in must be your own (i.e. you need to have written your assignment)
- You are allowed to copy and paste example code from other websites, but you must include a comment in your code that attributes the website you copied the code from (i.e. original author's name and URL to the original code)
- You can discuss the assignment with other teams
- However, you cannot just tell another team the answer to a particular problem

Final Thoughts

In many cases, engineers are expected to just make things work given a particular design constraint (e.g. software package to use or are limited to a particular hardware platform).

You will likely run into similar situations in this class while designing and implementing your assignments and project.

When you're stuck, try searching online for a solution. Many times others have tried something similar and documented their experiences for others to learn and benefit from

Do not publically post answers to assignments, or your code until after the assignment due date.

Contact your instructor or GTA as soon as you encounter a problem you're unable to solve. Don't wait to begin right before the assignment is due.