



Department of Computer Science and Engineering (Data Science)

Subject: Big Data Engineering (DJ19DSL604)

AY: 2022-23

Experiment 3

(Messaging Services)

Bhuvi Ghosh
60009210191

Aim: Implement messaging system using AMPS

Theory:

Communication, in general, is the process of transferring data from a source to a destination by using any of the available modes such as audio, video, text or even signals, etc. This communication may be straightforward between one sender and one receiver or it may include multiple senders and receivers. On the basis of the number of senders and receivers involved, a communication can either be "Point-to-Point" or "Multi-point".

Point-to-Point Communication:

In telecommunications, a point-to-point connection is a communications link between two communication endpoints or nodes. A telephone call is an example of this, in which two phones are linked, and what one caller says can only be heard by the other.

A "point-to-multipoint" or broadcast link, on the other hand, allows multiple nodes to receive information sent by a single node. Leased lines and microwave radio relays are also examples of point-to-point connections.

In a point-to-point communication, there will be a transmitter and a receiver connected together with a suitable connection. The capacity of the connecting channel remains unchanged throughout the communication.



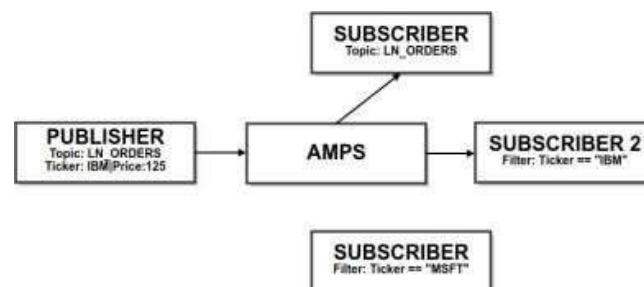
Department of Computer Science and Engineering (Data Science) Multi-point Communication

In telecommunications, point-to-multipoint communication (P2MP, PTMP, or PMP) is a form of one-to-many communication that allows numerous routes to be established from a single site to several locations.

The usage of gigahertz radio frequencies for point-to-multipoint telecommunications is common in wireless Internet and IP telephony. P2MP systems have been created with and without a return channel from the numerous receivers. The system employs a kind of time-division multiplexing to enable the return channel traffic, which is transmitted from a central antenna to numerous receiving antennas.

AMPS from 60East Technologies is a fast messaging engine that supports both publish-subscribe messaging and queuing. It is not just a simple messaging engine it is also packed with some powerful feature like high availability, historical replay, aggregation and analytics, content filtering and continuous query, last value caching, focus tracking, and more.

AMPS PUB-SUB Model



This is same as any other pub-sub model wherein a publisher publishes the message to a topic and subscriber subscribes to a topic and reads the message from it. But amps also provide some powerfully features like we can do **content filtering** where we can subscribe to a topic with a filter and with the help of that we can get only the filtered messages from AMPS server.

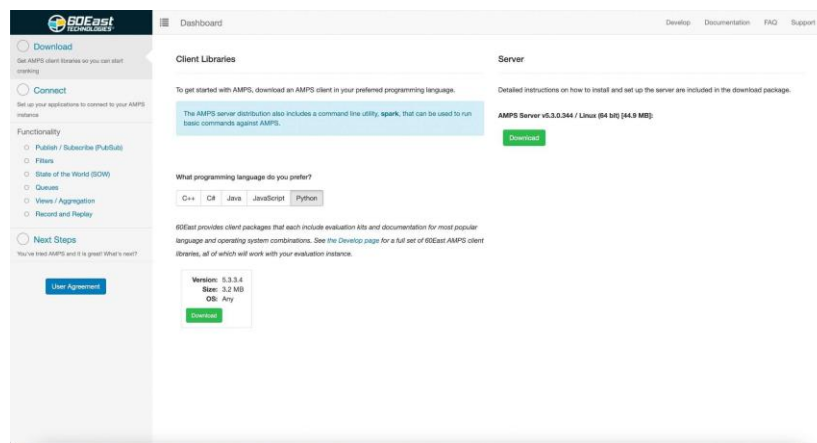
Setting up AMPS:

- Visit <https://www.crankuptheamps.com/evaluate/>

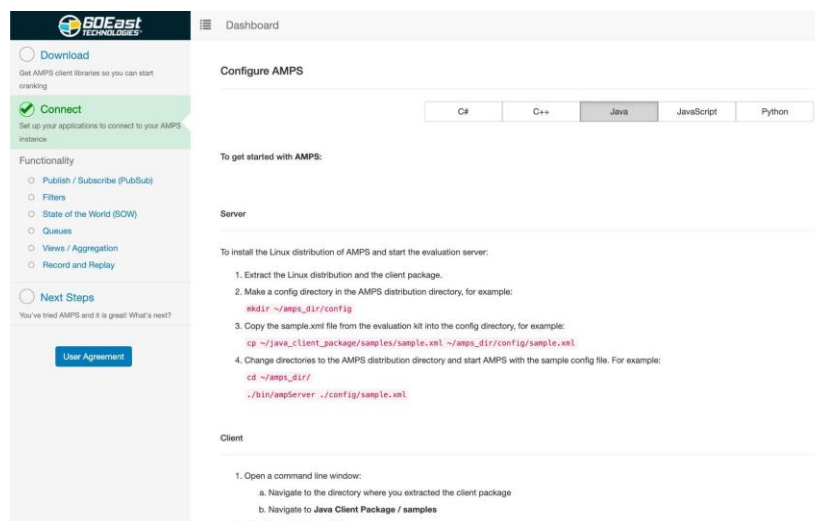


Department of Computer Science and Engineering (Data Science)

- Sign up with any email ID
- You'll receive an evaluation kit of AMPS in your email.
- Download the AMPS Server
- Download the Client Libraries (as per programming language preference)



- After downloading the server binaries and client, click on Connect



- On the right side pane, you'll see the steps to configure AMPS Server and Client.
- Once configured, head over to the Functionality tab (below Connect)



Department of Computer Science and Engineering (Data Science)

- Test your AMPS configuration by publishing / subscribing to a topic.

Publish Messages / Subscribe for Messages

AMPS Publish and Subscribe

Low-latency publish and subscribe messaging is at the heart of AMPS. The other features of AMPS build on this foundation.

Subscribe to a topic

This sample shows how to subscribe to a topic. Notice that, with simple publish/subscribe messaging, messages will only be received on the topic when messages are published.

```
1 from AMPS import *
2
3 # Construct a client object
4 c = Client("subscriber")
5
6 # Connect and login
7 c.connect("tcp://127.0.0.1:9007/amps/json")
8 c.login()
9
10 # Subscribe
```

AMPS is now ready to be explored.

Lab Assignment:

1. Setup AMPS server in the lab and all students will connect to it.
2. Write basic JSON messages (having flat structures), publish, and consume using AMPS.
3. Write nested JSON messages (having 5-6 levels of hierarchy), publish and subscribe using AMPS.
4. Apply Content Filtering, subscribing to relevant messages only.



Subscriber.py:

```
client.py x publish.py Welcome
client.py > ...
1  from AMPS import Client
2
3  # Construct a client object
4  c = Client("subscriber")
5
6  # Connect and logon
7  c.connect("tcp://10.120.107.156:9092")
8  c.logon()
9
10 # Subscribe
11 for m in c.subscribe("test"):
12     print ("Received message: %s" % (m.get_data()))
```

Publisher.py

```
client.py publish.py x Welcome
publish.py > ...
1  from AMPS import Client
2
3  # Construct a client object
4  c = Client("publisher")
5
6  # Connect and logon
7  c.connect("tcp://10.120.107.156:9092/amps/json")
8  c.logon()
9
10 # Publish
11 c.publish("test", '{ "message" : "Hello, world!" }')
```

AMPSConsoleSubscriber.py & Publisher.py

```
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ python3 AMPSConsoleSubscriber.py &
[1] 10137
csds-student@mum0923cpu0962:~/Downloads/amps-python-client-5.3.4.1/sample$ python3 AMPSConsolePublisher.py &
[2] 10469
csds-student@mum0923cpu0962:~/Downloads/amps-python-client-5.3.4.1/samples$ { "hi" : "Hello, world!"}
csds-student@mum0923cpu0962:~/Downloads/amps-python-client-5.3.4.1/samples$
```

```
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ python3 AMPSSOWConsolePublisher.py &
[1] 10540
csds-student@mum0923cpu0962:~/Downloads/amps-python-client-5.3.4.1/samples$ python3 AMPSSOWConsoleSubscriber.py &
[2] 10559
[1] Done
python3 AMPSSOWConsolePublisher.py
csds-student@mum0923cpu0962:~/Downloads/amps-python-client-5.3.4.1/samples$ Receiving messages from the SOW.
{ "text":"Hello, world!", "messageNumber": 0 }
{ "text":"Hello, world!", "messageNumber": 1 }
{ "text":"Hello, world!", "messageNumber": 2 }
{ "text":"Hello, world!", "messageNumber": 3 }
{ "text":"Hello, world!", "messageNumber": 4 }
{ "text":"Hello, world!", "messageNumber": 4 }
{ "text":"This is new information", "messageNumber":5 }
{ "text":"Hello, world!", "messageNumber": 6 }
{ "text":"Hello, world!", "messageNumber": 7 }
{ "text":"Hello, world!", "messageNumber": 8 }
{ "text":"Hello, world!", "messageNumber": 9 }
Done receiving messages from the SOW.
csds-student@mum0923cpu0962:~/Downloads/amps-python-client-5.3.4.1/samples$
```

AMPSSOWConsoleSubscriber.py & Publisher.py

AMPSSOWandSubscribeConsoleSubscriber.py & Publisher.py:

```
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples

csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ python3 AMPSSOWandSubscribeConsoleSubscriber.py &
[1] 10632
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ Receiving messages from the SOW.
(sow) : b'{"text":"Hello, world!", "messageNumber": 0 }'
(sow) : b'{"text":"Hello, world!", "messageNumber": 1 }'
(sow) : b'{"text":"Hello, world!", "messageNumber": 2 }'
(sow) : b'{"text":"Hello, world!", "messageNumber": 3 }'
(sow) : b'{"text":"Hello, world!", "messageNumber": 4 }'
(sow) : b'{"text":"This is new information", "messageNumber":5 }'
(sow) : b'{"text":"Hello, world!", "messageNumber": 6 }'
(sow) : b'{"text":"Hello, world!", "messageNumber": 7 }'
(sow) : b'{"text":"Hello, world!", "messageNumber": 8 }'
(sow) : b'{"text":"Hello, world!", "messageNumber": 9 }'
Done receiving messages from the SOW.
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ python AMPSSOWConsolePublisher.py &
[2] 10749
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ Command 'python' not found, did you mean:
command 'python3' from deb python3
command 'python' from deb python-is-python3
^C
[2]+  Exit 127                  python AMPSSOWConsolePublisher.py
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ python3 AMPSSOWConsolePublisher.py &
[2] 10765
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ (publish) : b'{"text":"Hello, world!", "messageNumber": 0 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 1 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 2 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 3 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 4 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 5 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 6 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 7 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 8 }'
(publish) : b'{"text":"Hello, world!", "messageNumber": 9 }'
(publish) : b'{"text":"This is new information", "messageNumber":5 }'
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$
```

```
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples

csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ python3 AMPSSOWConsoleSubscriber.py &
[1] 11829
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$ Receiving messages from the SOW.
{ "text":"Hello, world!", "messageNumber": 0 }
{ "text":"Hello, world!", "messageNumber": 1 }
{ "text":"Hello, world!", "messageNumber": 2 }
{ "text":"Hello, world!", "messageNumber": 3 }
{ "text":"Hello, world!", "messageNumber": 4 }
{ "text":"This is new information", "messageNumber":5 }
{ "text":"Hello, world!", "messageNumber": 6 }
{ "text":"Hello, world!", "messageNumber": 7 }
{ "text":"Hello, world!", "messageNumber": 8 }
{ "text":"Hello, world!", "messageNumber": 9 }
Done receiving messages from the SOW.
csds-student@mum0923cpu0962: ~/Downloads/amps-python-client-5.3.4.1/samples$
```

Filter.py:

```
Open  Filter.py
~/Downloads/amps-python-client-5.3.4.1/samples

1 import socket
2 import json
3
4 class Client:
5     def __init__(self, name):
6         self.name = name
7         self.socket = None
8
9     def connect(self, host, port):
10        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11        self.socket.connect((host, port))
12
13    def logon(self):
14        # Implement logon process if needed
15        pass
16
17    def subscribe(self, topic, filter):
18        # Subscribe command
19        subscribe_command = {
20            "command": "subscribe",
21            "topic": topic,
22            "filter": filter
23        }
24        self.send_message(subscribe_command)
25
26        # Receive messages
27        while True:
28            data = self.socket.recv(4096)
29            if not data:
30                break
31            messages = json.loads(data.decode())
32            for message in messages:
33                yield message
34
35    def send_message(self, message):
36        serialized_message = json.dumps(message)
37        self.socket.sendall(serialized_message.encode())
38
39 if __name__ == "__main__":
40     # Construct a client object
41     c = Client("client-name")
42
43     # Connect to server
44     c.connect("127.0.0.1", 9007)
45
46     # Logon (if required)
47     c.logon()
48
49     # Subscribe with a filter
50     for m in c.subscribe("test", "/details/items/description LIKE 'kitten' "):
51         print("Received message for topic: %s : %s" % (m.get("topic"), m.get("data")))
```

```
csds-student@num0923cpu0962:~/Downloads/amps-python-client-5.3.4.1/samples$ python3 Filter.py
csds-student@num0923cpu0962:~/Downloads/amps-python-client-5.3.4.1/samples$
```




Apply Content Filtering, subscribing to relevant messages only.

The screenshot shows a Visual Studio Code window with a Python script named `subscriber.py` and its terminal output. The script connects to an AMPQ broker, publishes a JSON message, and subscribes to a queue named `test`. The terminal output shows the received message for the `test` topic, which is a JSON object containing invoice details and items.

```
subscriber.py
1  #!/usr/bin/env python
2  # coding: utf-8
3  import pika
4  import json
5
6  # Connect to the broker
7  c = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
8  c.login()
9
10 # Create a channel
11 channel = c.channel()
12
13 # Publish JSON data
14 data = {
15     "invoice": 53498,
16     "customerID": 1983,
17     "details": {
18         "discountCode": "1347",
19         "items": [
20             {
21                 "sku": 3317,
22                 "qty": 468,
23                 "description": "action figure : purple : fox"
24             },
25             {
26                 "sku": 5098,
27                 "qty": 283,
28                 "description": "shoes : beige : kitten"
29             }
30         ]
31     }
32 }
33
34 # Publish
35 channel.publish("test", json.dumps(data))
```

```
subscriber@ubuntu20:~/Downloads/AMPS_Lab/amps-python-client-5.3.3.4$ python3 subscriber.py
Received message for topic: test : {"invoice": 53498, "customerID": 1983, "details": {"discountCode": "1347", "items": [{"sku": 3317, "qty": 468, "description": "action figure : purple : fox"}, {"sku": 5098, "qty": 283, "description": "shoes : beige : kitten"}]}}
Received message for topic: test : {"invoice": 53498, "customerID": 1983, "details": {"discountCode": "1347", "items": [{"sku": 3317, "qty": 468, "description": "action figure : purple : fox"}, {"sku": 5098, "qty": 283, "description": "shoes : beige : kitten"}]}}
```