```
import asyncio
import socket

HOST = 'vcm-32603.vm.duke.edu'
PORT = 51300
ID = 'yz605'

connected_ports = []
secondary_connections = {}
listening_ports = []
listening_sockets = {}

async def handle_primary_connection(reader, writer):
    print("Connected to server on primary connection")
    writer.write(f"id {ID}\n".encode('ascii'))
    await writer.drain()

    data = await reader.readline()
    port = int(data.decode('ascii').strip().split()[1])

    try:
        # Start two coroutines in parallel: handle_secondary_connection
and handle_status_messages
        await asyncio.gather(handle_secondary_connection(port,
connected_ports), handle_status_messages(reader))
    except asyncio.CancelledError:
        # If one of the coroutines is cancelled, close the writer and
wait for it to be closed
        writer.close()
        await writer.wait_closed()

    print(f"The connected ports are: {connected_ports}")
    print(f"The listening ports are: {listening_ports}")
    for s in listening_sockets.values():
        s.close()


async def handle_secondary_connection(port, connected_ports):
    if port in connected_ports:
        reader, writer = secondary_connections[port]
    else:
        reader, writer = await asyncio.open_connection(HOST, port)
        connected_ports.append(port)
        secondary_connections[port] = reader, writer

    print(f"Connected to server on secondary connection to port {port}")
    writer.write(f"id {ID}\n".encode('ascii'))
    await writer.drain()

    data = await reader.readline()
    if not data:
```

```
        writer.close()
        await writer.wait_closed()
        return

    message = data.decode('ascii').strip()
    if message.startswith("query"):
        new_port = int(message.split()[1])
        await handle_secondary_connection(new_port, connected_ports)
    elif message.startswith("listen"):
        listen_port = int(message.split()[1])
        if listen_port not in listening_sockets:
            listening_ports.append(listen_port)
            listening_sockets[listen_port] = await
asyncio.start_server(handle_listen_connection, port=listen_port)
            print(f"> listening on port {listen_port}: 1. starting
establish")
        else:
            print(f"> Already listening on port {listen_port}")


async def handle_listen_connection(reader, writer):
    print(f"> listening on port {writer.get_extra_info('sockname')[1]}:
2. server established")
    writer.write(f"id {ID}\n".encode('ascii'))
    await writer.drain()

    while True:
        data = await reader.readline()
        if not data:
            break

        message = data.decode('ascii').strip()
        print(f"> listening on port {writer.get_extra_info('sockname')
[1]}: 3. received message {message}")
        query_port = int(message.split()[1])
        await handle_secondary_connection(query_port, connected_ports)


async def handle_status_messages(reader):
    while True:
        # Wait for a status message from the server
        data = await reader.readline()
        if not data:
            break

        # Print the message and cancel all other coroutines
        message = data.decode('ascii').strip()
        print(f"Received status message: {message}")
        for task in asyncio.all_tasks():
            task.cancel()


async def main():
```

```
    try:
        # Open the initial connection to the server
        reader, writer = await asyncio.open_connection(HOST, PORT)
        await handle_primary_connection(reader, writer)
    except (ConnectionRefusedError, socket.gaierror):
        print("Failed to connect to server")


if __name__ == '__main__':
    asyncio.run(main())
```

Above is the code I wrote for this task. But the result is not correct. There are something that I cannot figure out:

- The following error raised at the `data = await reader.readline()` statement: RuntimeError: readuntil() called while another coroutine is already waiting for incoming data

Can you please help me with this? Thank you so much!