Assignment 2

Little Brother

Due Date: Monday, February 27, 2017 @ 23:55

ECE 4564 - Network Application Design

Learning Objectives

Message Broker

- AMQP Protocol
- RabbitMQ

Publish/Subscribe

Callback Routines

Performance Monitoring

Data Persistence

- noSQL Database

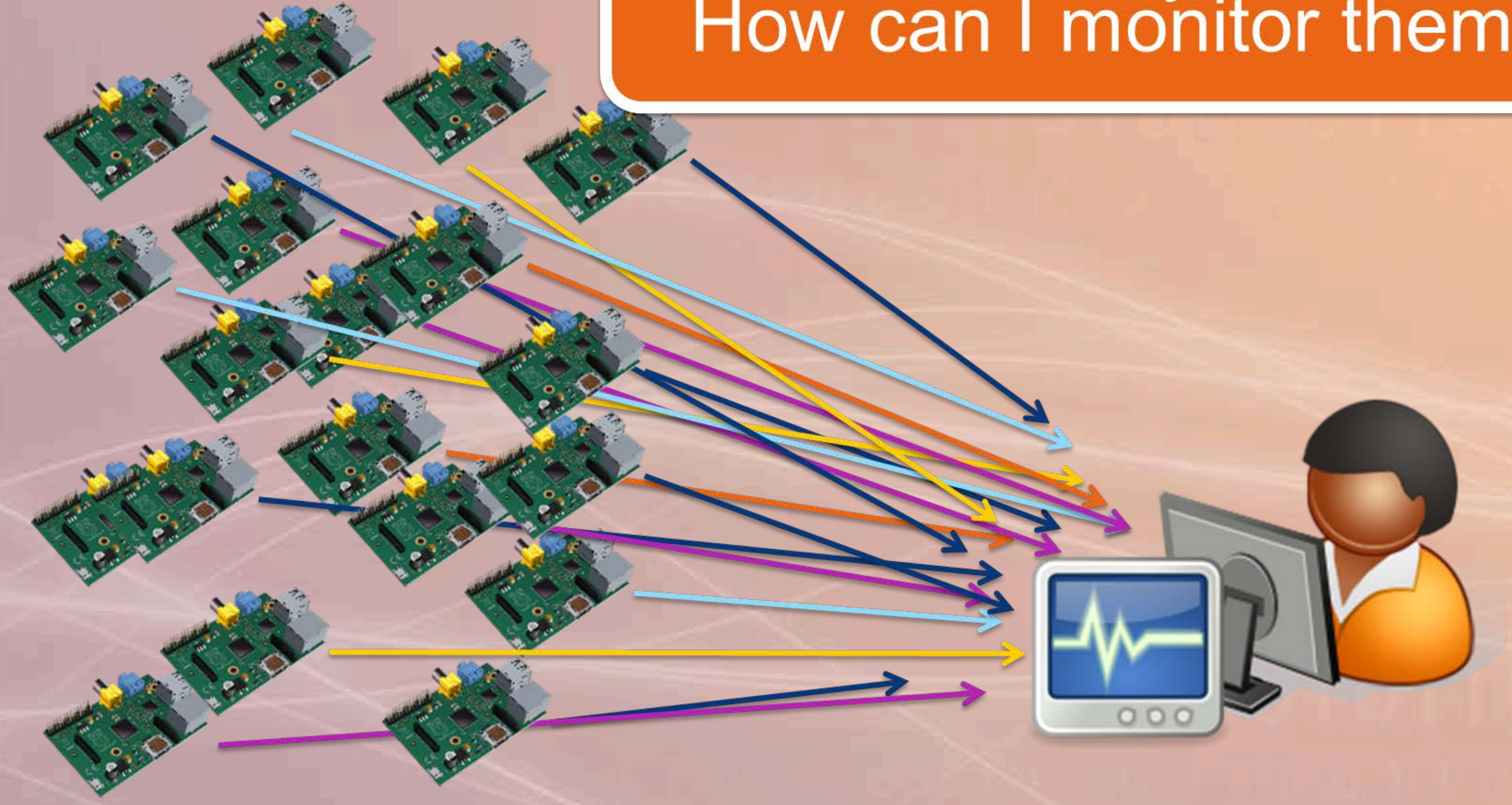
Serialization and Data Interchange

- JSON

Raspberry Pi GPIO

Remote Monitoring

I have an army of machines.
How can I monitor them all?



Assignment Overview

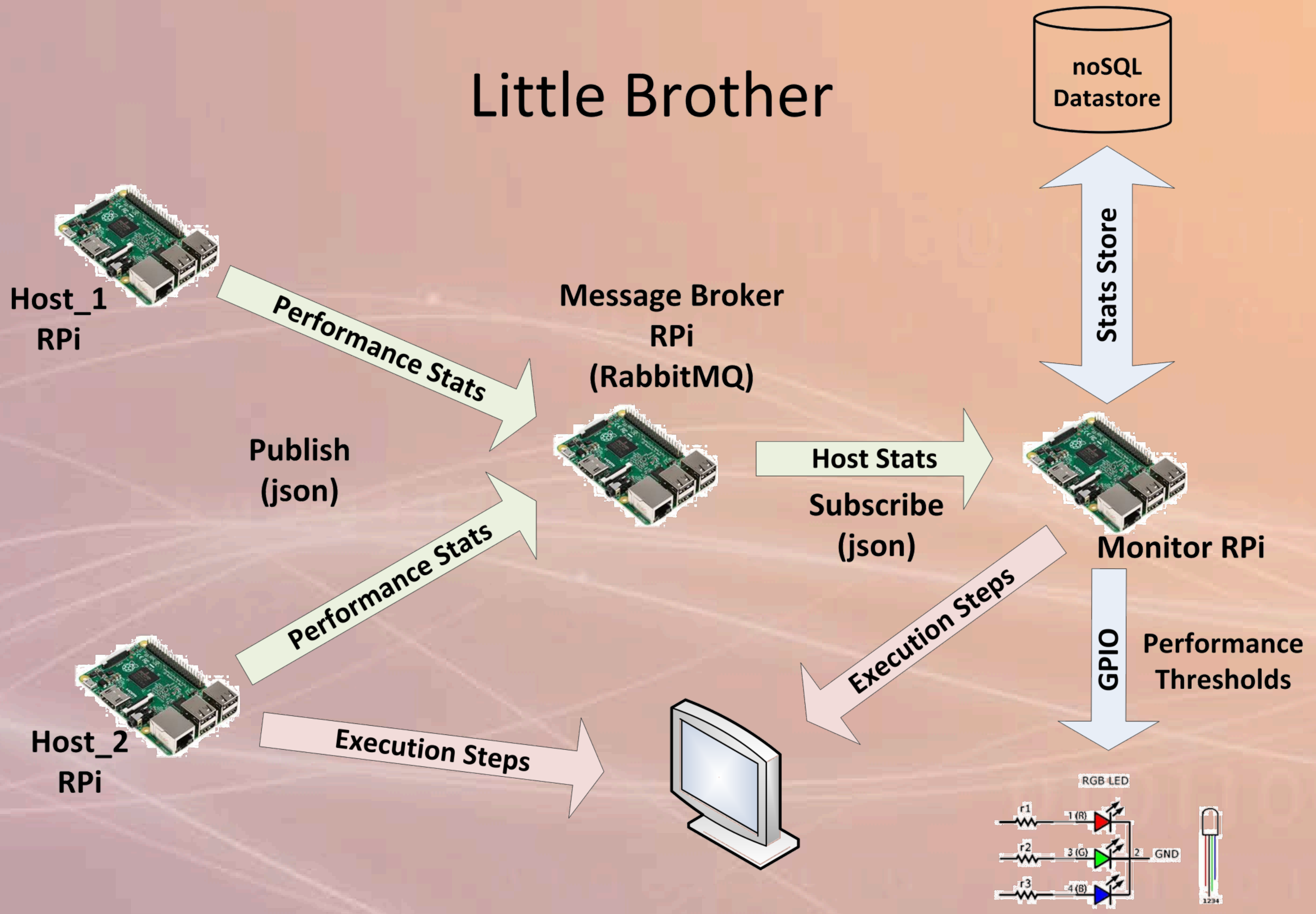
Assignment 2 is a remote host monitoring system inspired by the Big Brother Unix Network Monitoring and Notification System released in 1996.

Your system consists of four Rpi's:

- Two Host RPi's that generate performance data
- One Message Broker RPi running RabbitMQ that manages the Host RPi data
- One Monitor RPi that processes the performance data generated by the Host RPi's

Pub/Sub: Host RPi data is published to the Message Broker Rpi. The Monitor RPi subscribes to that performance data.

System Overview



RPi Functions

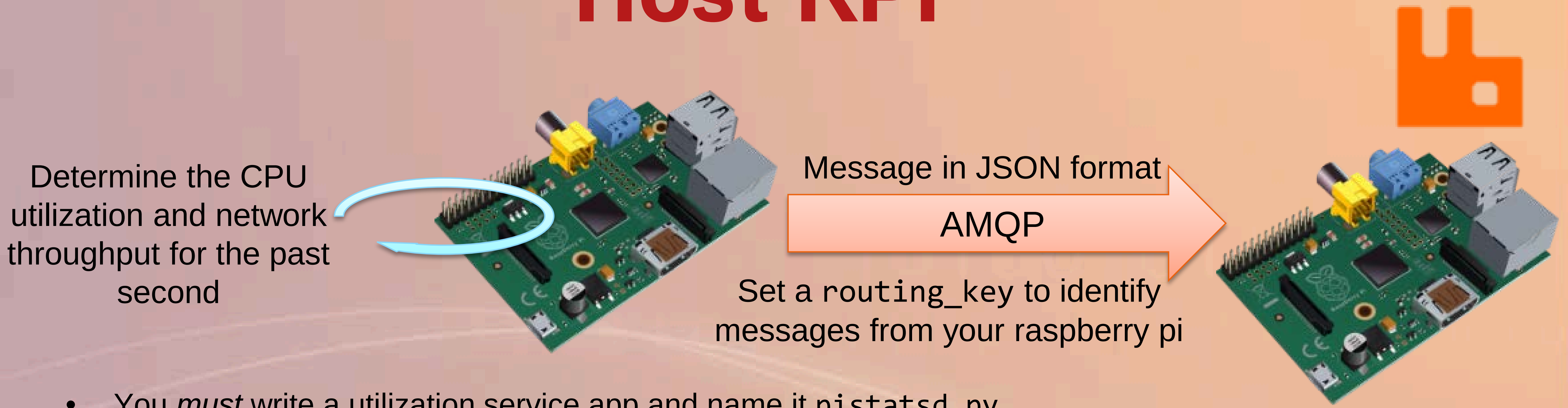
- Host RPi
  - CPU & Network load monitoring app
  - Reads CPU and network utilization information every second
  - Calculates CPU utilization and network throughput
  - Publishes calculated info to Message Exchange on the Broker

- Message Broker RPi (RabbitMQ)
  - Stores and routes messages
  - Manages message queues
  - Pushes messages to interested subscribers

RPi Functions

- Monitor RPi (server)
  - Subscribe to metrics information from a particular queue
  - Remotely monitor Host RPi resource usage
  - Store performance data in noSQL datastore
  - Compute basic stats on high and low utilization times
    - Compare current performance data against stored noSQL data
  - Display threshold alerts on RGB led attached to Montor RPi’s GPIO pins

Host RPi


Determine the CPU utilization and network throughput for the past second
Message in JSON format
AMQP
Set a routing_key to identify messages from your raspberry pi

- You *must* write a utilization service app and name it `pistatsd.py`
- The utilization service will calculate CPU utilization and network throughput every second, and *must* publish the data for the last second's utilization to an exchange named: `pi_utilization`
- The utilization message *must* be in JSON format
- Each message published by the utilization service *must* have a `routing_key`
  - The `routing_key` will be used by subscribers to filter for messages from A Host RPi
  - HINT: Since you'll be using a routing key to label your published messages, you'll want to use a `direct` exchange type in RabbitMQ.
  - Direct Exchange tutorial: https://www.rabbitmq.com/tutorials/tutorial-four-python.html
- Your app *must* accept a set of command line parameters that control the behavior of your app

Host Command Line Parameters

```
pistatsd –b message broker [–p virtual host] [–c login:password] –k routing key
```

`-b`

This is the IP address or named address of the message broker to connect to

`-p`

This is the virtual host to connect to on the message broker. If not specified, should default to the root virtual host (i.e. ‘/’)

`-c`

Use the given credentials when connecting to the message broker. If not specified, should default to a guest login.

`-k`

The routing key to use when publishing messages to the message broker

Calculate CPU Utilization

Reading CPU Usage

- The proc filesystem exposes the `/proc/uptime` file:
  - Contains two numbers:
    - System uptime
    - System idle time

Calculating CPU Usage

- Utilization = $1 - \frac{\Delta idle}{\Delta uptime}$
- $\frac{\Delta idle}{\Delta uptime} = \frac{idle_t - idle_{t-1}}{uptime_t - uptime_{t-1}}$
  - where *t* is the time when a sample was taken

Calculate Network Throughput

Reading network interface throughput

- The `proc` filesystem exposes the `/proc/net/dev` file:
  - Contains statistics for each interface, as a list of numbers:
    - Bytes sent/received

Calculating network interface throughput

- Throughput $= \frac{bytes_t - bytes_{t-1}}{t - (t-1)}$
  - where *t* is the time when a sample was taken

Message Format

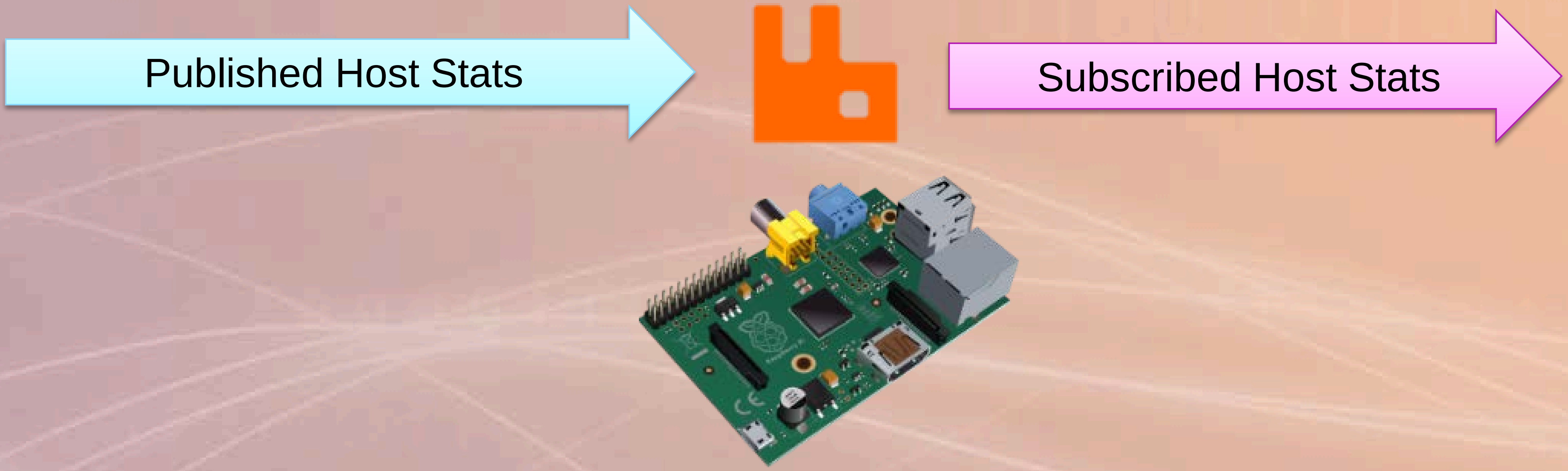
The utilization messages you publish to the server must be in JSON format with the following structure:

- Each message *must* be published as a single JSON object
- *Must* contain a `cpu` key with the CPU utilization as a *number* (i.e. a numeric type) between 0 and 1
- *Must* contain a `net` object where:
    - `net` *must* contain keys that correspond to the interface name (e.g. `lo`, `eth0`, and so on) of each network interface on your raspberry pi as given in `/proc/net/dev`
    - Each network interface key *must* contain a JSON object with the transmit (`tx`) rate and receive (`rx`) rate for the interface as numbers.
    - The `tx` and `rx` rates reported *must* be in units of bytes/second

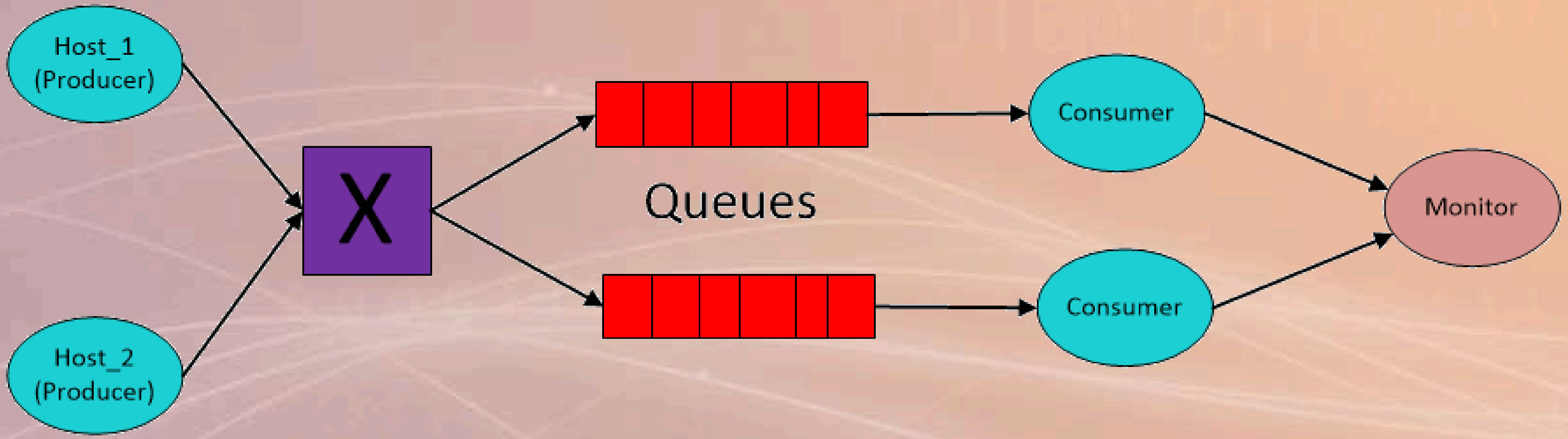
Utilization Message Example

```
{
    "net": {
        "lo": {
            "rx": 0,
            "tx": 0
        },
        "wlan0": {
            "rx": 708,
            "tx": 1192
        },
        "eth0": {
            "rx": 0,
            "tx": 0
        }
    },
    "cpu": 0.2771314211797171
}
```

Message Broker RPi



Direct Exchange



RabbitMQ

Create Users and Virtual Hosts

By default, RabbitMQ enables the guest account for connections from localhost. It's recommended that you create a new user.

- Create a new user (executed from the computer running RabbitMQ)

```
sudo rabbitmqctl add_user "name" "password"
```

Virtual Hosts are ways to separate different messaging apps that use one RabbitMQ server

- Create new virtual hosts:

```
sudo rabbitmqctl add_vhost "my_new_vhostname"
```

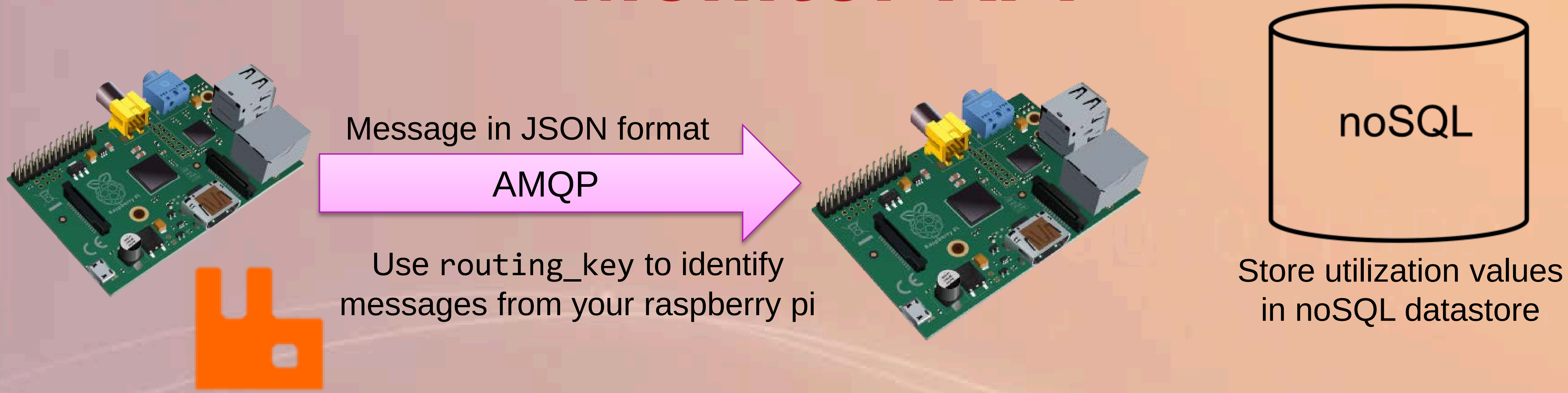
- Giving users permissions to access a virtual host. In this example, we give the user complete access to do *anything* in the virtual host:

```
sudo rabbitmqctl set_permissions -p "my_new_vhostname" ".*" ".*" ".*"
```

- Complete documentation for `rabbitmqctl` can be found online or by looking at the Linux Manual page

```
man rabbitmqctl
```

Monitor RPi



Use `routing_key` to identify messages from your raspberry pi

Store utilization values in noSQL datastore

- You *must* write a utilization service app and name it `pistatsview.py`
- This app will subscribe to an exchange named: `pi_utilization` using a `routing_key` to tell the exchange which Host RPi to receive messages from
- Your app *must* accept a set of command line parameters that control the behavior of your app
- For each message your stats client receives:
  - Print the current utilization values for each category to `stdout`
  - Print the highest value observed for each category to `stdout`
  - Print the lowest value observed for each category to `stdout`
- The highest and lowest values do *not* need to be persistent (i.e. the values should be the highest and lowest seen for the current run of your Monitor RPi app)

Monitor Command Line Parameters

```
pistatsview –b message broker [–p virtual host] [–c login:password] –k routing key
```

`-b`

This is the IP address or named address of the message broker to connect to

`-p`

This is the virtual host to connect to on the message broker. If not specified, should default to the root virtual host (i.e. '/')

`-c`

Use the given credentials when connecting to the message broker. If not specified, should default to a guest login.

`-k`

The routing key to use for filtering when subscribing to the `pi_utilization` exchange on the message broker

Hi / Lo and Threshold Values

- Store performance data in noSQL datastore as it comes in.

- Compute basic stats on high and low utilization times
  - Compare current performance data against stored noSQL data

  - Display threshold alerts on RGB led attached to Montor RPi’s GPIO pins

Sample Monitor RPi Output

```
Host_1:
cpu: 0.543244 [Hi: 0.99, Lo: 0.3171314211797171]
lo: rx=0 B/s [Hi: 0 B/s, Lo: 0 B/s], tx=0 B/s [Hi: 0 B/s, Lo: 0 B/s]
eth0: rx=0 B/s [Hi: 0 B/s, Lo: 0 B/s], tx=0 B/s [Hi: 0 B/s, Lo: 0 B/s]
wlan0: rx=708 B/s [Hi: 1048576 B/s, Lo: 0 B/s], tx=1192 B/s [Hi: 769885 B/s, Lo: 0 B/s]

Host_2:
cpu: 0.3648725132 [Hi: 0.99, Lo: 0.2771314211797171]
lo: rx=0 B/s [Hi: 0 B/s, Lo: 0 B/s], tx=0 B/s [Hi: 0 B/s, Lo: 0 B/s]
eth0: rx=0 B/s [Hi: 0 B/s, Lo: 0 B/s], tx=0 B/s [Hi: 0 B/s, Lo: 0 B/s]
wlan0: rx=12548 B/s [Hi: 1948576 B/s, Lo: 0 B/s], tx=778 B/s [Hi: 769885 B/s, Lo: 0 B/s]

Host_2:
cpu: 0.290135487 [Hi: 0.99, Lo: 0.2771314211797171]
lo: rx=0 B/s [Hi: 0 B/s, Lo: 0 B/s], tx=0 B/s [Hi: 0 B/s, Lo: 0 B/s]
eth0: rx=0 B/s [Hi: 0 B/s, Lo: 0 B/s], tx=0 B/s [Hi: 0 B/s, Lo: 0 B/s]
wlan0: rx=2098576 B/s [Hi: 2098576 B/s, Lo: 0 B/s], tx=42 B/s [Hi: 769885 B/s, Lo: 0 B/s]
```

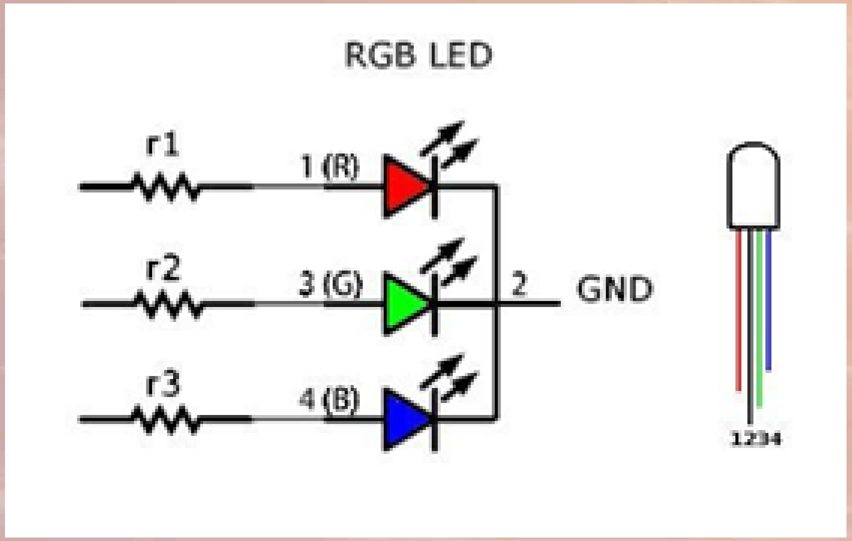
Threshold Indicator

Indicates CPU utilization thresholds.

Green : <25%

Yellow : >25% - <50%

Red : >50%



Diffused – 10mm – COM-11120

Datasheet Tutorial

Grading

- Report: 10%
  - 1 to 2 pages
  - Single-spaced
  - Submit as PDF

- Validation with GTAs: 90%
  - Measurements of performance stats : 10 points
  - JSON formatting : 10 points (5 points per link)
  - RabbitMQ : 30 points
  - NoSQL DB operations :
    - Performance statistics storage : 10 points
    - Hi/Lo calculations : 10 points
  - Threshold indications LED : 15 points
  - All operations over eduroam : 5 points

Grading

- Code must include error handling
- Must have enough print statements indicating steps i.e. print the current network load at the server etc.

During validation:

- Must show performance stats from both Host RPi's
- Must clear noSQL database prior to team validation
- Must issue queries showing proper noSQL DB behavior

Report

You must document the design, and outcomes in a brief written report. Your report should contain the following items.

- At the top of the first page of your report, include: your names (as recorded by the university); your email addresses; your team number; and the assignment name (e.g., “ECE 4564, Assignment 2”). Do not include your Virginia Tech ID number or your social security number.
- The body of the report must contain the following sections. Use section numbers and headings to organize your report.
  - Section 1 – Objectives: Provide a description of the design objectives and general approach to the design. Include a system diagram showing your system’s end-to-end function.
  - Section 2 – Team member responsibilities
  - Section 3 – Conclusions: Discuss the outcome of your design and any problems encountered and resolutions; what you learned by doing this project; and any experiences that were particularly good or bad.

Python Style

Follow style guide PEP0008 when writing and commenting your code

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

What You Turn In

All assignments must be submitted through Canvas, no later than the due date of Feb 27 2017 @ 23:55

Your assignment should be a single zip or tarball (i.e. tar.gz, tar.bz) which contains the following:

- All source code you wrote for this assignment
  - Python code running on Host and Monitor RPi’s
    - pistatsd.py
    - pistatsview.py
  - A ReadMe file describing client and server initialization procedures and any extra libraries used.
- Report (PDF file)

Assignment References

RabbitMQ Tutorials

- RabbitMQ - Direct Exchange Tutorial

Python Docs

- Pymongo

MongoDB

Share other references on the Canvas discussion group

The Validation Process

1. Each team will get 15 minutes slot for validation. This timing deadline has to be followed strictly. This includes the time for setting up your hardware and presenting your assignment. Sign-up for slots will be available on Canvas soon.

2. Each team member **must** be present during the validation. Inform Dr. Plymale and GTA’s prior to your validation time by email in case you won’t be able to make it. Team members absent from validation will receive a 10 point reduction in score.

3. Each team should bring their hardware kits and setup for validation (Raspberry Pi, power adapter & related hardware for that HW). Kindly come 10 minutes before your appointed time slot.

4. During validation, you will be provided with sets of monitors, keyboards, mouse, HDMI cables and a LAN cable if needed.

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 example code until after the assignment due date.

Contact your instructor or GTA’s as soon as you encounter a problem you’re unable to solve.

Don’t wait until right before the assignment is due.