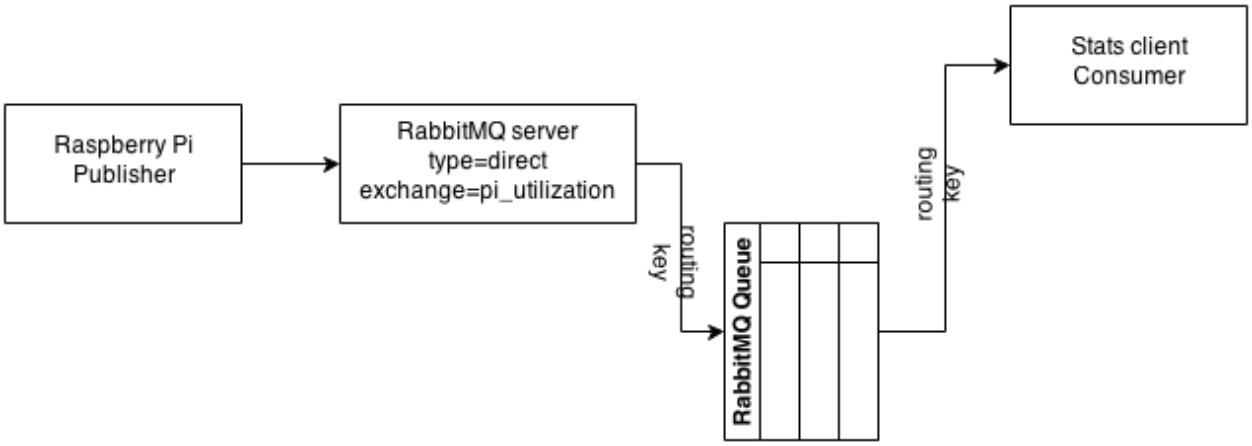
Matt Hazinski
Nathan McCloskey

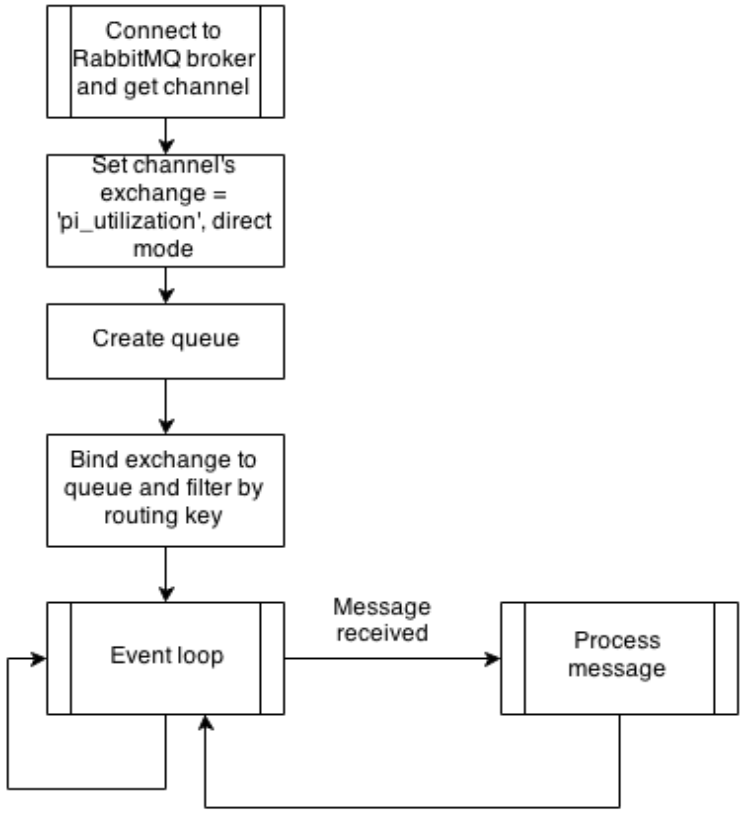
System overview



The system consists of a RabbitMQ message broker and separate publisher and consumer applications. An event loop runs on the server to continuously acquire usage statistics and publish them to the message broker in JSON format. The publisher sends messages to an exchange named 'pi_utilization' with direct queue-exchange mapping and a routing key specified by the utilization server.

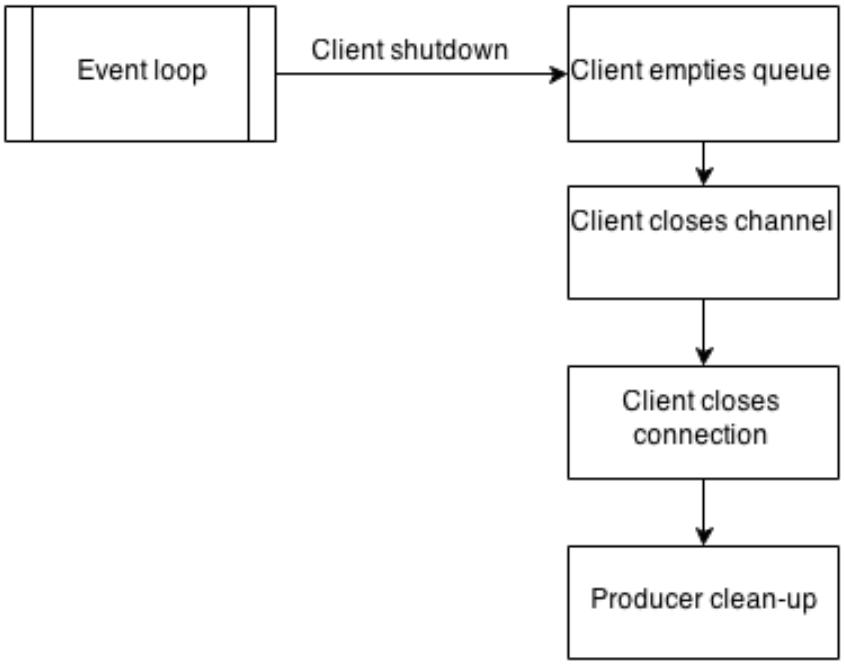
The statistics client connects to the broker and creates a queue directly bound to the pi_utilization exchange which contains all messages tagged with the routing key specified on the command line. As messages arrive to the consumer, they are processed by deserializing the JSON, calculating statistics, and printing to stdout.

Client connection process



When the client is started, it connects to the RabbitMQ message broker using the credentials, virtualhost, and hostname specified on the command line. A channel instance is created and the pi_utilization exchange is specified in direct mode. A queue is then created and bound to the exchange. Messages tagged with the routing key arrive in the queue, and are processed as they arrive.

Client disconnection process



When a shutdown request is received (e.g. by a Ctrl-C), a cleanup process is initiated. First, the RabbitMQ queue is emptied if any messages remain in it. The client closes the channel through pika's API, which notifies the RabbitMQ broker so that a clean shutdown can occur; the producer is notified of this and can stop sending messages as they will no longer be relevant if read at a later time. Finally, the consumer closes its connection to the broker.