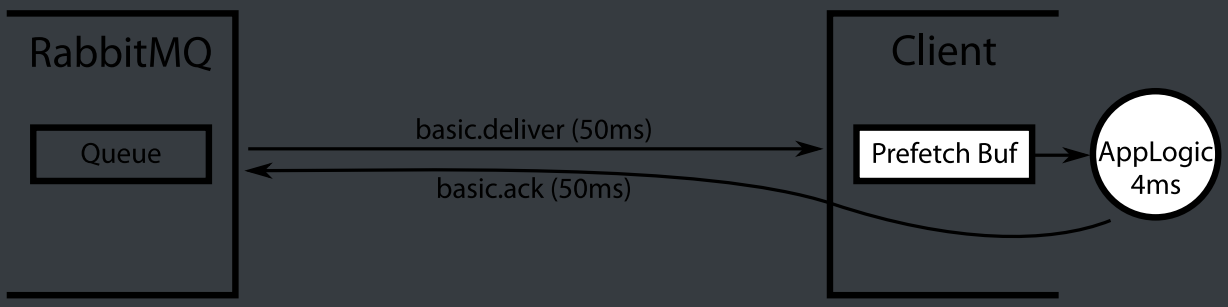
QOS

如果不设置QoS，那么RabbitMQ将会尽最大能力将消息推送到客户端，由于客户端将大量消息缓存在内存里，客户端的内存会急剧膨胀。这时，你去访问Rabbit的时候，会发现队列空着，可是实际上可能有成千上万的未确认的消息囤积在客户端，等待着客户端程序的处理。如果你新增一个消费者，队列里也不会有消息留下去发送给新的消费者。消息已经被缓存在已存在的客户端那里，就算已经有可用的客户端可以处理消息了，那依旧会持续一段时间。而这是待优化的地方。

默认的QoS `prefetch` 缓冲区是无限的，而这会造成一些不好的行为和性能损失。可是你到底应该把QoS `prefetch` 缓冲区大小设置为多少？我们的目标是让所有客户端都能充分工作，但是要尽量地减小客户端的缓冲区大小，以便更多的消息驻留在Rabbit的队列里，从而可以供新的消费者使用，或者在消费者空闲的时候发送给他们。

假定Rabbit需要50ms才能从队列里获取到消息并将其放到网络上然后到达消费者，而消费者则需要4ms实现去处理这些消息。一旦消费者处理完消息，它将会发送ack指令返回给Rabbit，Rabbit还需要50ms才能发送和处理该消息。所以我们实际上需要花费总共104ms。如果我们设置QoS `prefetch` 缓冲区大小为1条消息，那么Rabbit在这个过程完毕之前就不会发送下一条消息。因此客户端将会在104ms内忙碌4ms，或者说总时间的3.8%。然而我们想要它每时每刻（100%）都在忙碌着。



如果我们把客户端上每条消息的往返时间除以处理时间，我们将会得到$104/4 = 26$。如果每条消息需要花费4ms去处理，那么缓冲区的所有消息将会花费$26 * 4 = 104$ms。第一个4ms是客户端处理第一条消息。然后客户端发送一个ack并且去处理缓冲区里的下一条消息。这个ack花费50ms到达Rabbit。然后Rabbit发送一个新的消息给客户端，而这又会花费50ms，所以通过这104ms，客户端处理完了它所缓存的消息，而新的消息也已经到来了等待着客户端去处理。因此客户端将会一直处于忙碌状态。设置一个大的QoS `prefetch` 也并不会让它变得更快，但是我们将缓冲区的大小和消息在客户端的延迟降到最低：客户端对消息的缓冲时间不超过他们所需的时间，从而使客户端一直工作。事实上，客户端能在下一条消息到来之前就把缓冲区耗尽，因此缓冲区实际上一直为空。

上面的方案完全没毛病，但是前提是处理时间不变化和网络不波动。可是考虑到实际情况，如果突然网速降为原先的一半，那么缓冲区将不足以支持客户端，就会使客户端闲置，等待新的消息到来，因为客户端处理消息的速度比Rabbit发送消息的速度快。

为了解决这个问题，我们可能会把 `prefetch` 设置为原来的两倍。如果我们将缓冲区从26增大到51，而客户端处理消息的速度不变依旧为4ms，我们现在缓冲区的消息就需要$51 * 4 = 204$ms去处理，4ms用来处理第一个消息，留下200ms去发送ack和接受新的消息。因此我们现在可以应付网速降半的问题了。

可是如果网络表现正常，那么将我们的缓冲区大小设置为原来的两倍就意味着每一条消息将会在客户端缓冲区等一小会，就不会立即被处理了。而且，从满缓冲区到下一条新的消息到来，需要100ms，而这期间缓冲区还剩下25条消息，新的消息到来会追加到缓冲区的后面，这时缓冲区将会一直存在25条消息。此时，Rabbit发送消息的延迟从50ms增加到150ms。

因此，我们可以看到，增加缓冲区大小会提高网络恶化时的性能，但是会增加网络正常时的延迟。

同样，如果客户端处理消息的时间从4ms增加到40ms会发生什么呢？如果Rabbit的队列长度一定，由于现在消费的速度降低为原来的十分之一，现在队列将会迅速膨胀。你可能会想着通过增加更多的消费者来解决积压的消息，可是已经有一些消息被缓存在已存在而客户端的缓冲区里了。假设原来的缓冲区大小为26条消息，客户端需要花费40ms去处理第一条消息，而往返一次的时间客户端却只能处理$100/40 = 2.5$条消息，而不是原先的25条消息。因此，此时缓冲区还有$25 - 3 = 22$条消息。新来的消息不会被立即处理，它将会被排在第23个位置，而前面22条消息将会花费$22 * 40 = 880$ms。给Rabbit到客户端的延迟只有50ms，额外的880ms延迟就是95%等待时间了$(880/(880 + 50) = 0.946)$。

甚至更糟糕的是，如果我还把缓冲区扩大到51来防止网络恶化了，那么会发生什么？第一条消息被处理后，仍有50条消息留在客户端的缓冲区，100ms后（假设网络正常），一条新消息到来，缓冲区还有$50 - 3 = 47$条消息，这条新的消息会被排在第48位，它前面的消息需要花费$47 * 40 = 1880$ms。而Rabbit到客户端的延迟为50ms，这意味着客户端要负责97%的延迟（$1880/(1880 + 50) = 0.974$）。这可能是难以接受的：数据可能只有及时处理才是有效的，而不是在客户端接到消息将近2秒后再去处理。如果其他的客户端正在闲置，那么他们将啥也不能做：一旦消息发送到客户端，那么这个客户端将为这条消息负责，知道客户端ack或者reject了这条消息。一旦消息被发送，客户端就不能从其他客户端处拿到这条消息了。你想要的是客户端保持忙碌，但是让客户端尽量少的缓冲消息，从而使消息不会被客户端的缓冲区延迟，因此新的客户端可以快速地从队列里获取消息。

所以，缓冲区太小会导致客户端闲置，如果网络正常的话，缓冲区太大会导致消息有额外的延迟，如果客户端人突然处理消息慢了，那么消息的延迟就会大量高于平常。所以你可能想要的是一个可变大小的缓冲区。主动队列管理算法（*Active Queue Management*）尝试丢弃或拒绝消息，以避免消息长时间处于缓冲区中。