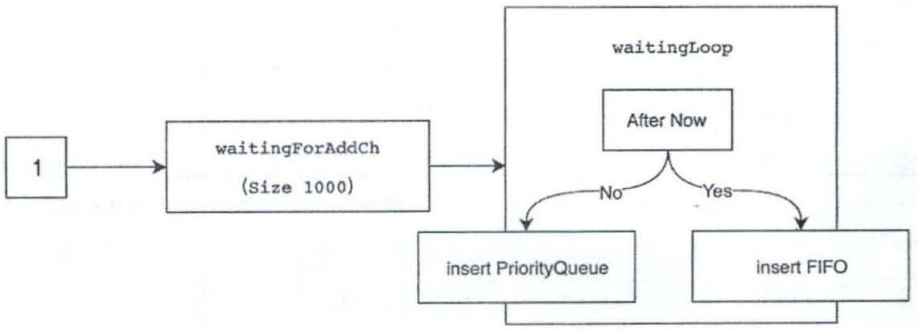

图 5-11　延迟队列运行原理

如图 5-11 所示，将元素 1 放入 waitingForAddCh 字段中，通过 waitingLoop 函数消费元素数据。当元素的延迟时间不大于当前时间时，说明还需要延迟将元素插入 FIFO 队列的时间，此时将该元素放入优先队列（waitForPriorityQueue）中。当元素的延迟时间大于当前时间时，则将该元素插入 FIFO 队列中。另外，还会遍历优先队列（waitForPriorityQueue）中的元素，按照上述逻辑验证时间。

5.4.3　限速队列

限速队列，基于延迟队列和 FIFO 队列接口封装，限速队列接口（RateLimitingInterface）在原有功能上增加了 AddRateLimited、Forget、NumRequeues 方法。限速队列的重点不在于 RateLimitingInterface 接口，而在于它提供的 4 种限速算法接口（RateLimiter）。其原理是，限速队列利用延迟队列的特性，延迟某个元素的插入时间，达到限速目的。RateLimiter 数据结构如下：

代码路径：vendor/k8s.io/client-go/util/workqueue/default_rate_limiters.go

```
type RateLimiter interface {
    When(item interface{}) time.Duration
    Forget(item interface{})
    NumRequeues(item interface{}) int
}
```

限速队列接口方法说明如下。

- **When**：获取指定元素应该等待的时间。
- **Forget**：释放指定元素，清空该元素的排队数。
- **NumRequeues**：获取指定元素的排队数。