```
    void informSinkFailed(Sink failedSink);
}
```

当 Sink 处理器启动时，用传递过来的 Sink 的列表，初始化 Sink 选择器且调用 `setSinks` 方法。在配置文件中该列表以相同的顺序指定。每次 Sink 处理事件时，就调用 `createSinkIterator` 方法。该方法必须返回一个迭代器，这个迭代器以 Sink 必须要求拉取数据入 Sink 的顺序，返回 Sink。

一旦 Sink 成功处理事件且返回成功，那么现在的迭代器就被废弃，再次调用该方法用来获取新的迭代器，这可能会返回不同顺序的 Sink。当 Sink 发送事件失败时（由抛出的异常所指示），那么调用 `informSinkFailed` 方法。如果需要，这可以用来将该 Sink 暂时加入黑名单。 162

为了创建一个自定义 Sink 选择器，必须在你的 *pom.xml* 文件的依赖部分包含 `flume-ng-core` 工件，像例 3-6 展示的那样。

Failover Sink 处理器

图 6-2 中展示的相同问题可以用一种略微不同的方式来解决。load-balancing sink 处理器的问题是，因为每个 Sink 组决定哪个 Sink 在大量的 Agent 上是活跃的，所以可能第二层的 Agent 将不能接收到相同总量的数据，尽管当使用循环顺序时，它们应该是平均的。然而，像前面描述的一样，可以配置 Sink 组使用硬连接方式的写操作，直到失败实际发生。大多数时候，通过允许 Sink 组写入一致的数据到相同的 Sink，可以预测多少数据被写入到每个 Agent。这种情况可以使用 *failover sink* 处理器来实现。

failover sink 处理器从 Sink 组中以优先级的顺序选择 Sink。拥有最高优先级的 Sink 先写数据直到它失败（在 RPC Sink 的情况下，Sink 的失败甚至可以是下游 Agent 的死亡），然后选择组中其他 Sink 中拥有最高优先级的 Sink。只有当前 Sink 写入数据失败时，才会选择另一个不同的 Sink 写数据。这能确保当没有失败时，每台机器上只有一个 Sink 写入到第二层的所有 Agent，只有当失败时某些机器才会看到更多传入的数据。

但是除非现在的 Sink 失败，failover 机制不会选择一个新的 Sink。这意味着，即使已经失败的最高优先级的 Agent 重新上线，failover sink 处理器也不会让写入该 Agent 的 Sink 激活，直到目前活跃的 Sink 遇到一个错误。图 6-3 展示了 failover sink 处理器的工作流。