Flink 网络流控及反压剖析

张俊 · OPPO / 大数据平台研发负责人

Flink traning course 进阶篇 – 2019年06月13日

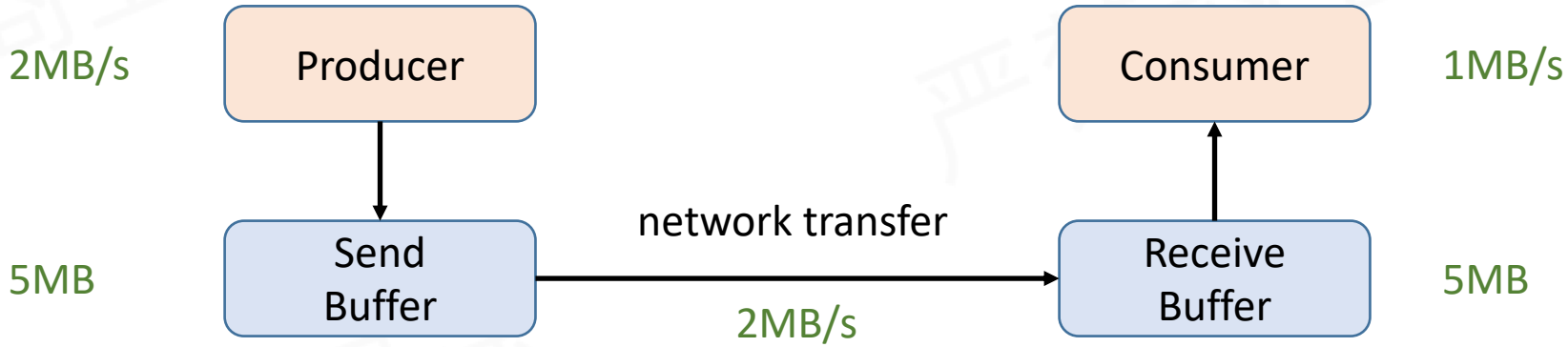
CONTENT

目录 >>

01

网络流控的概念与背景

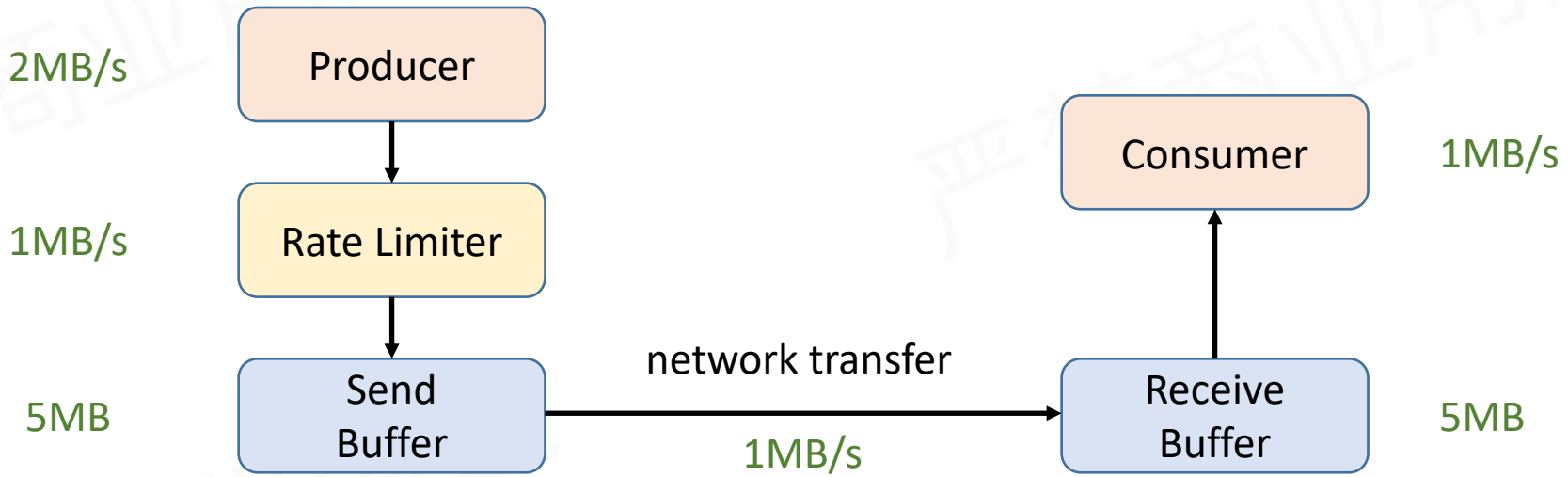
为什么需要网络流控



5秒钟后，将面临如下两种情况之一：

- ✓ bounded receive buffer：consumer丢弃新到达的数据
- ✓ unbounded receive buffer：buffer持续扩张，耗尽consumer内存

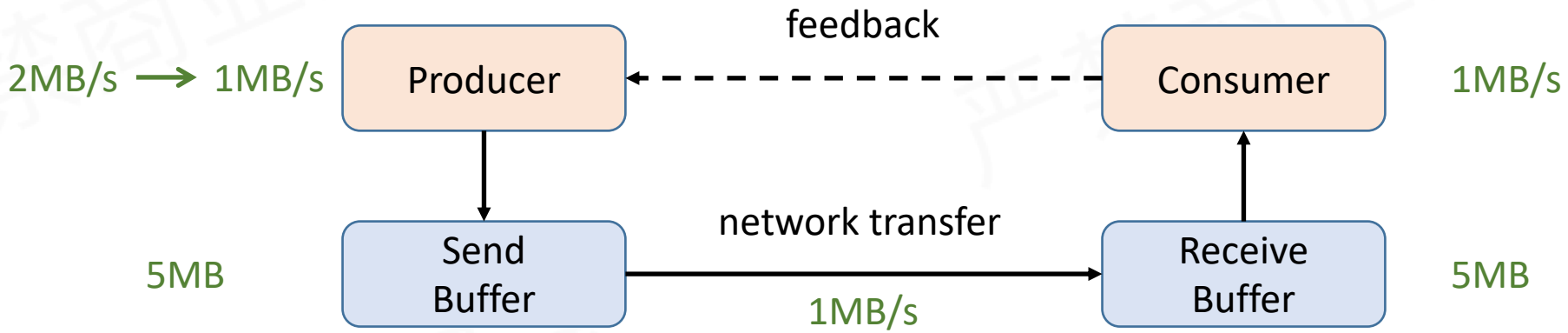
网络流控的实现：静态限速



静态限流有两点限制：

- ✓ 通常无法事先预估consumer端能承受的最大速率
- ✓ consumer承受能力通常会动态地波动

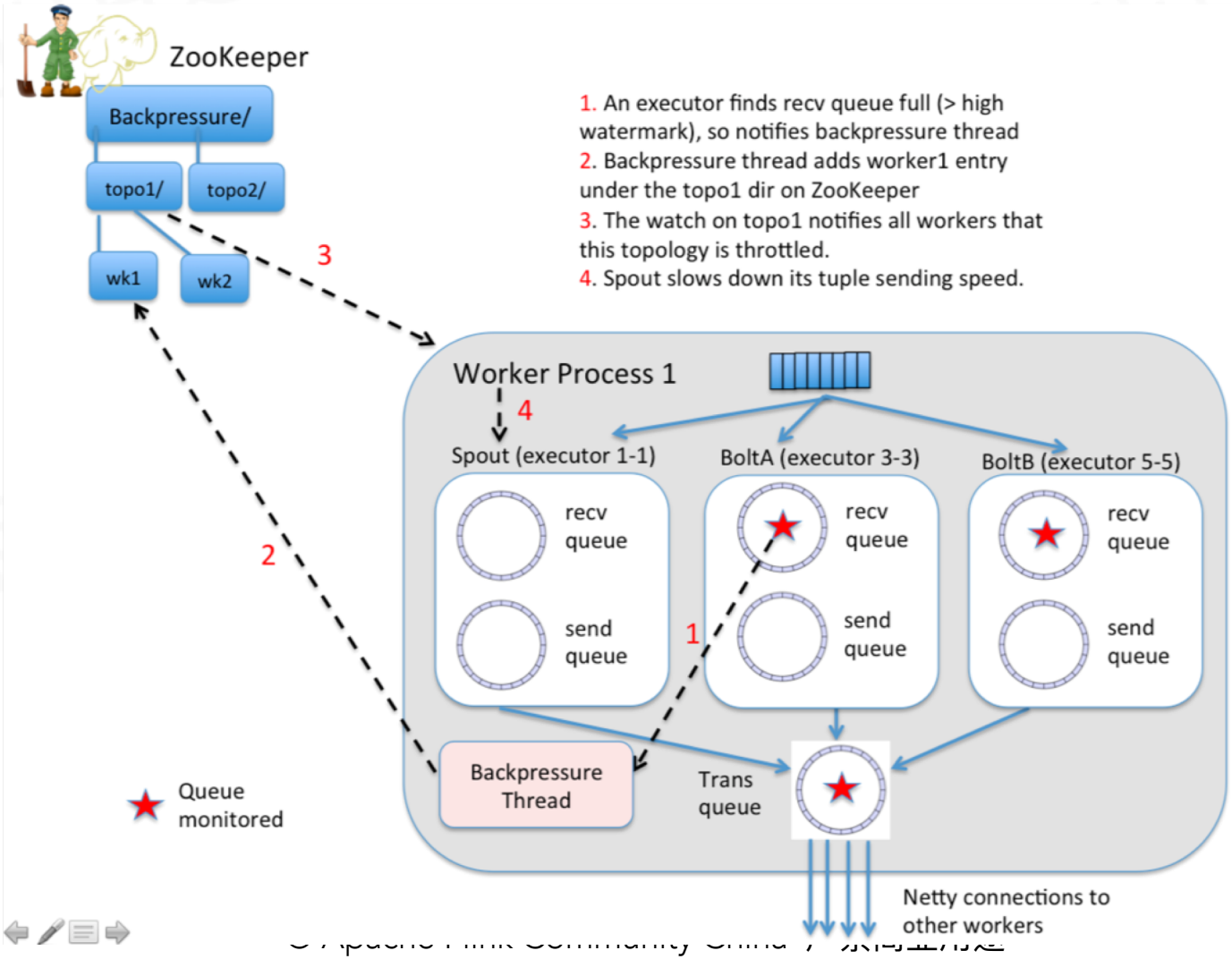
网络流控的实现：动态反馈/自动反压



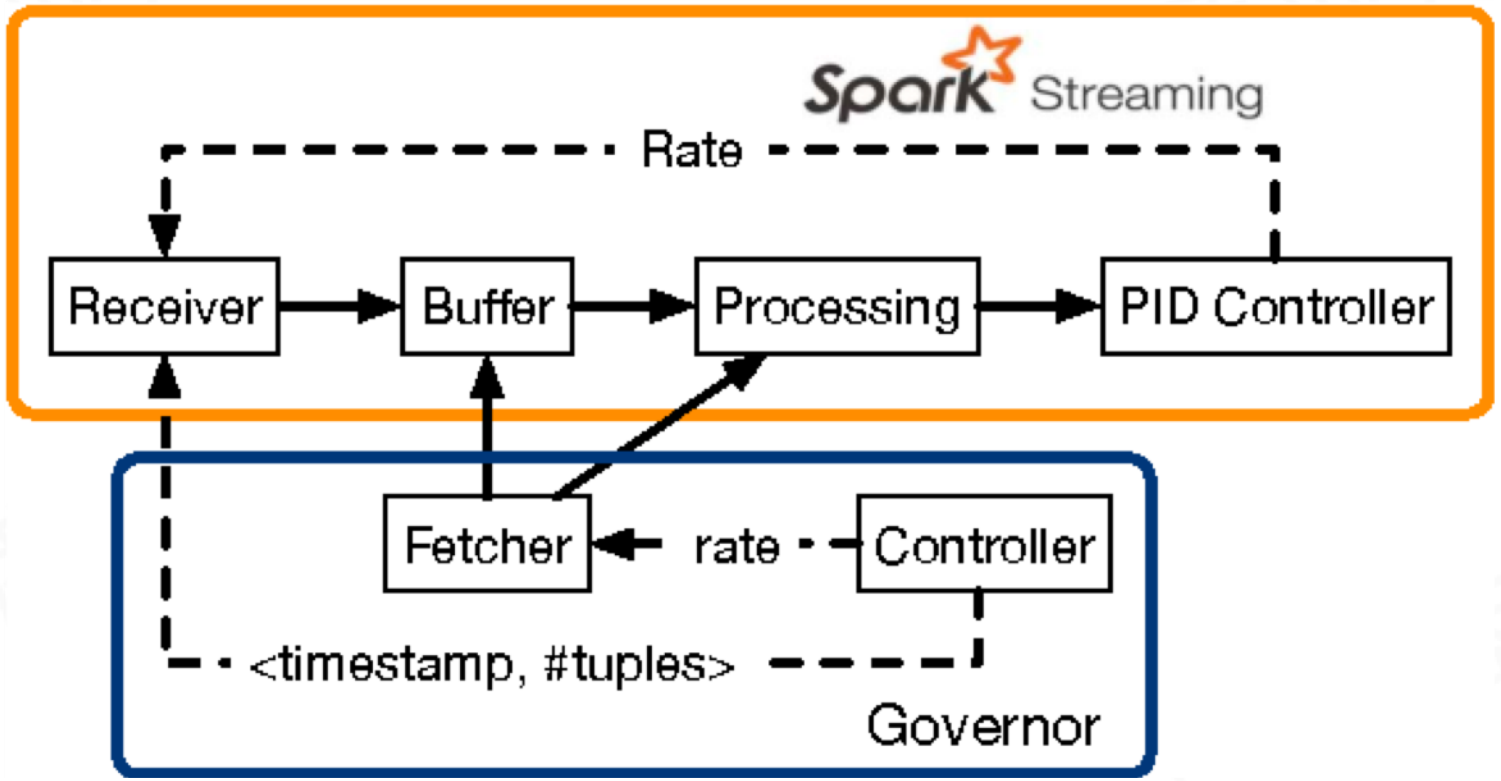
动态反馈分两种，广义上的反压机制都涵盖：

- ✓ 负反馈：接收速率小于发送速率时发生
- ✓ 正反馈：发送速率小于接收速率时发生

案例一：Storm反压


ZooKeeper
Backpressure/
topo1/
topo2/
wk1
wk2
3
2
1. An executor finds recv queue full (> high watermark), so notifies backpressure thread
2. Backpressure thread adds worker1 entry under the topo1 dir on ZooKeeper
3. The watch on topo1 notifies all workers that this topology is throttled.
4. Spout slows down its tuple sending speed.
Worker Process 1
4
Spout (executor 1-1)
BoltA (executor 3-3)
BoltB (executor 5-5)
recv queue
send queue
recv queue
send queue
recv queue
send queue
1
Backpressure Thread
Trans queue
Queue monitored
Netty connections to other workers

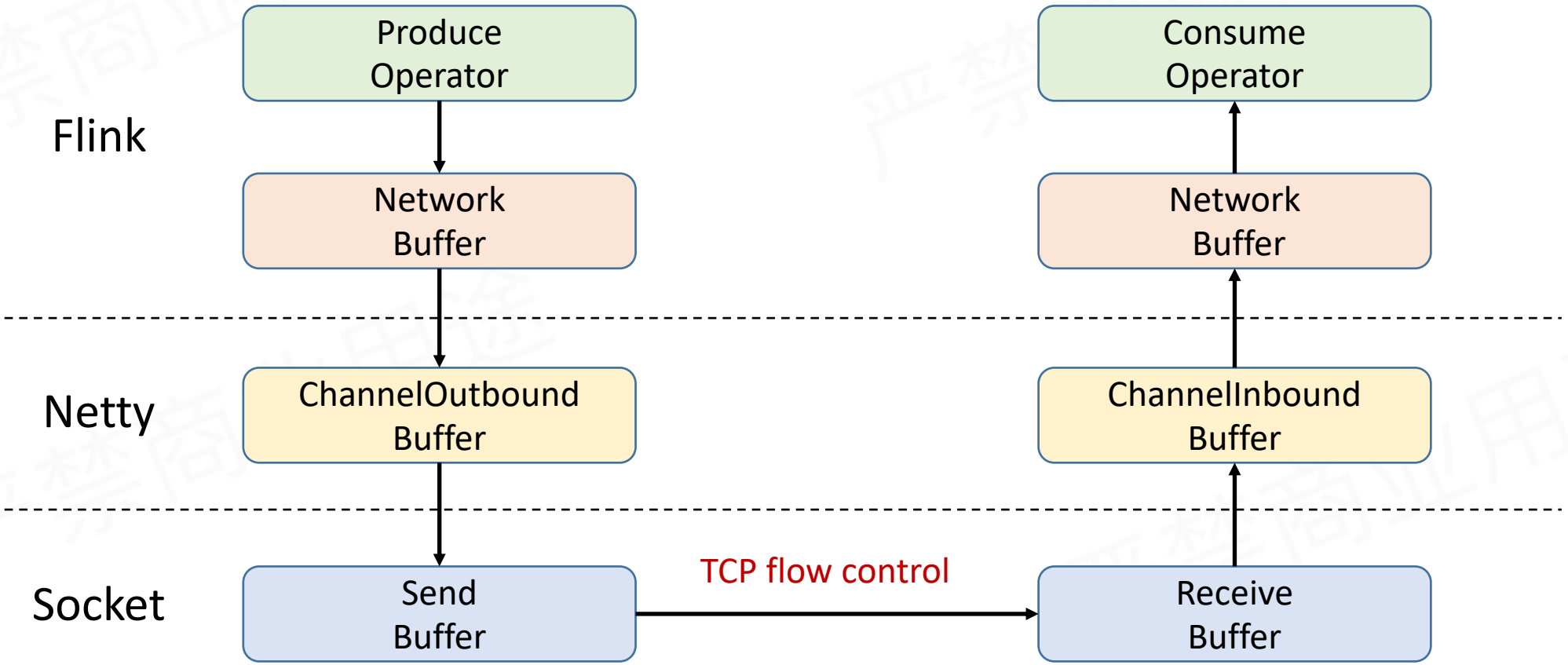
案例二：Spark Streaming反压



疑问

为什么Flink(before V1.5)里没有类似的feedback机制？

Flink网络传输的数据流向



先给结论

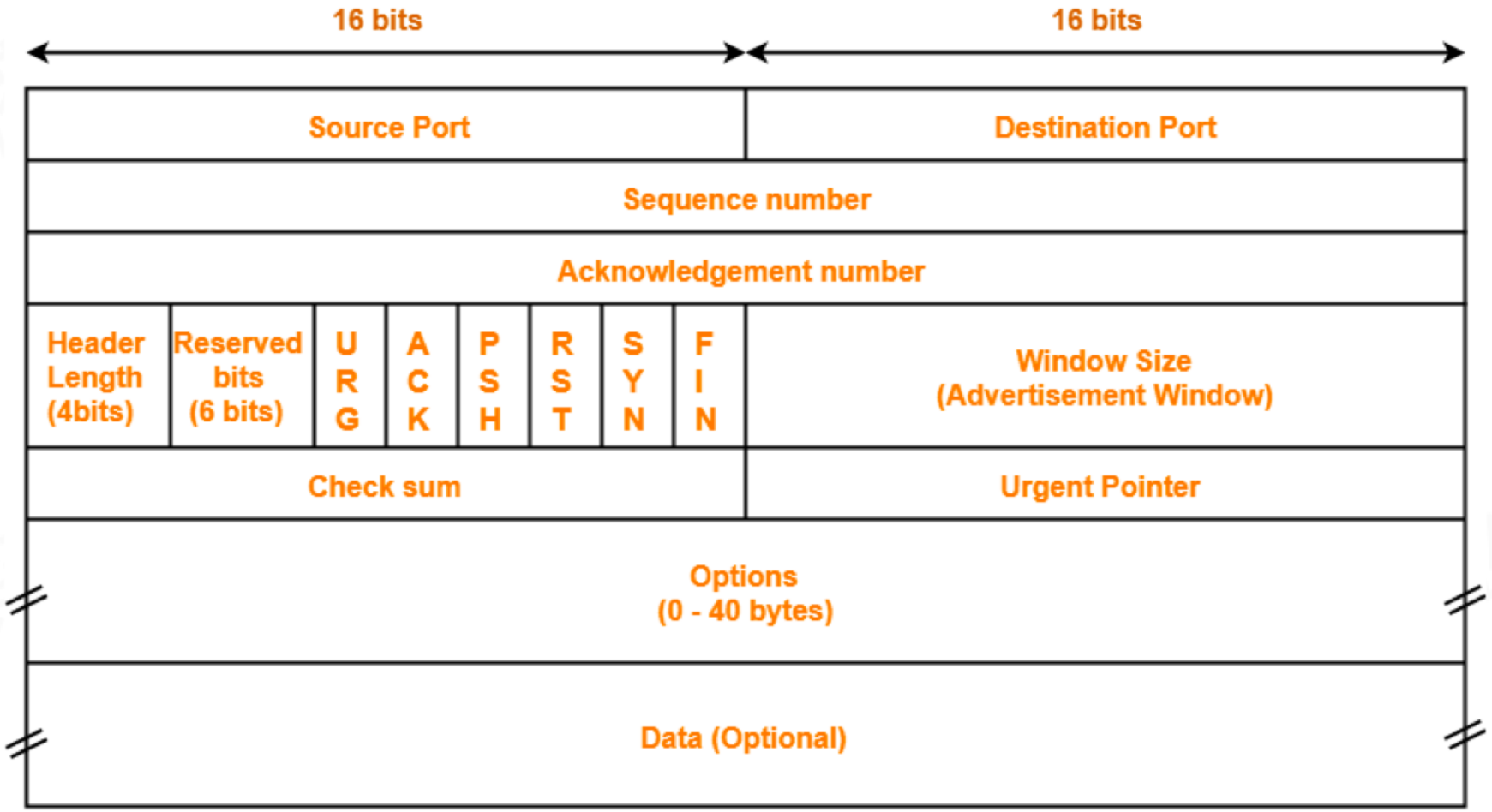
为什么Flink(before V1.5)里没有类似的feedback机制？

答：TCP天然具备feedback流控机制，Flink基于它来实现反压

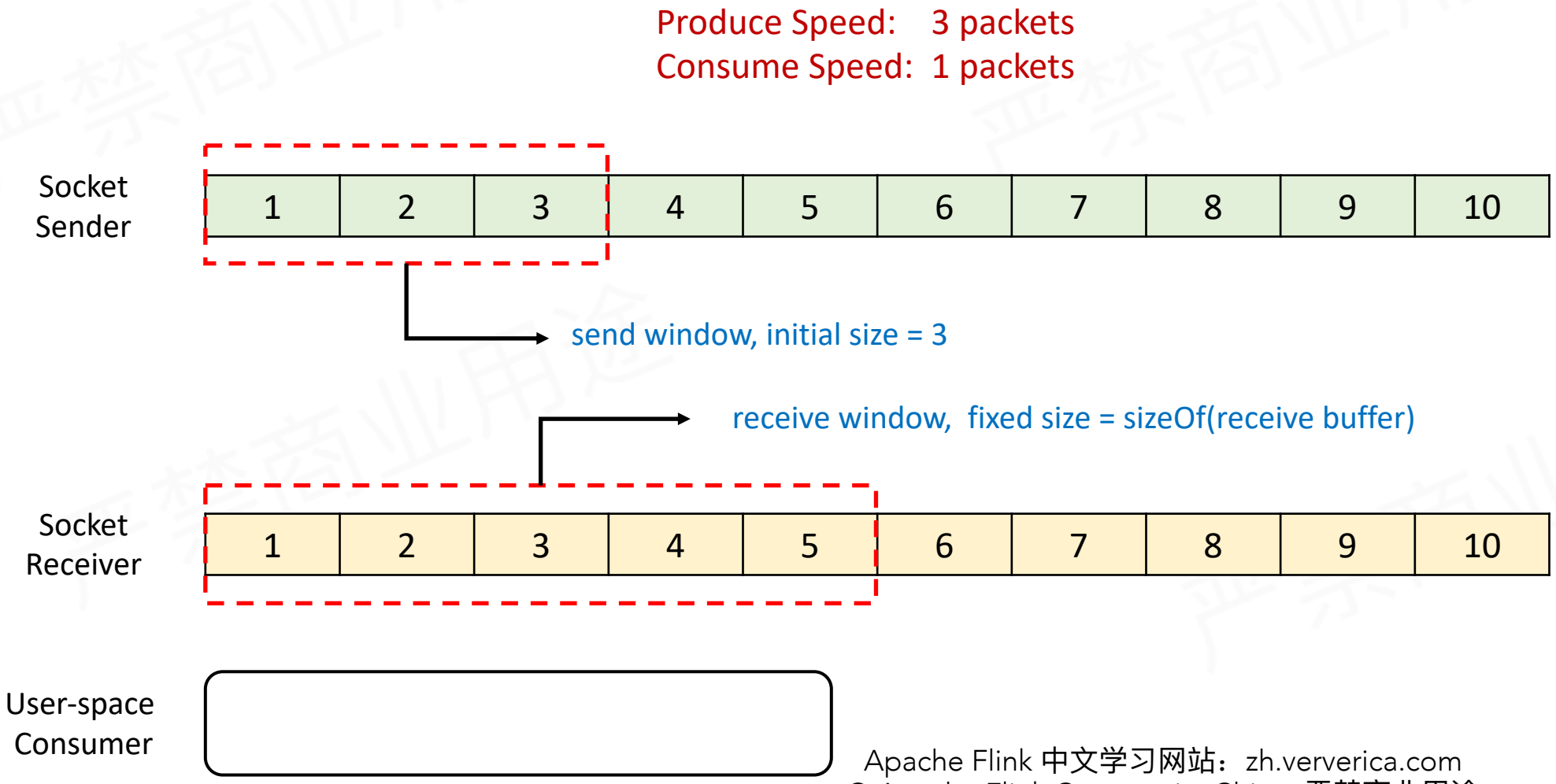
02

TCP流控机制

TCP Packet

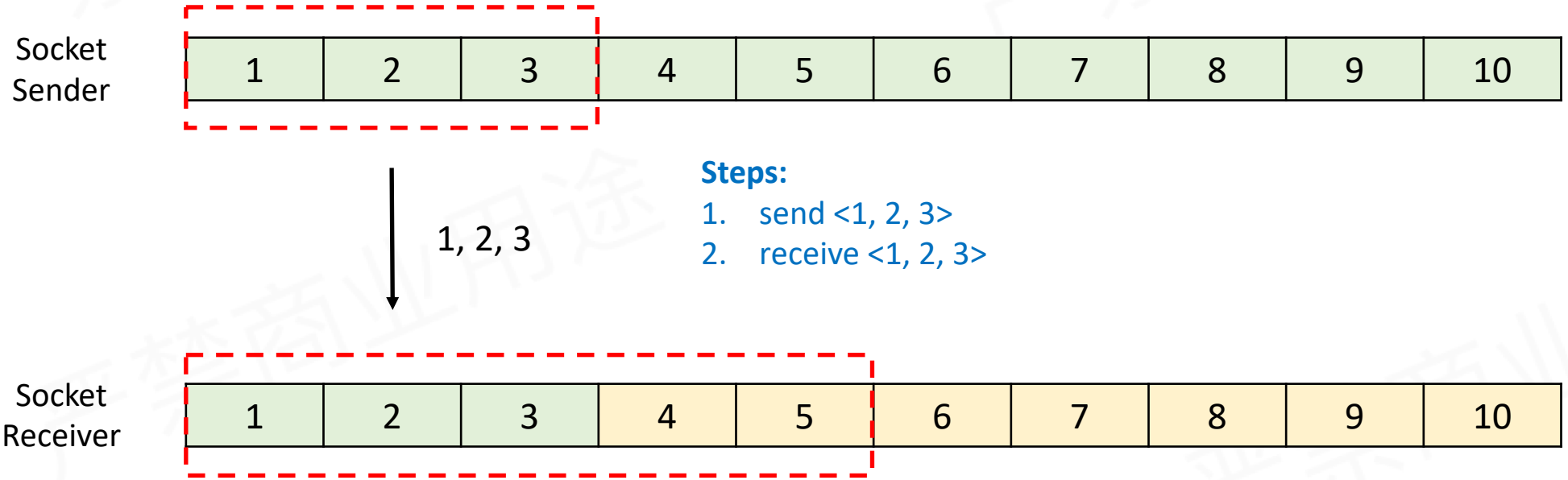


TCP流控：滑动窗口

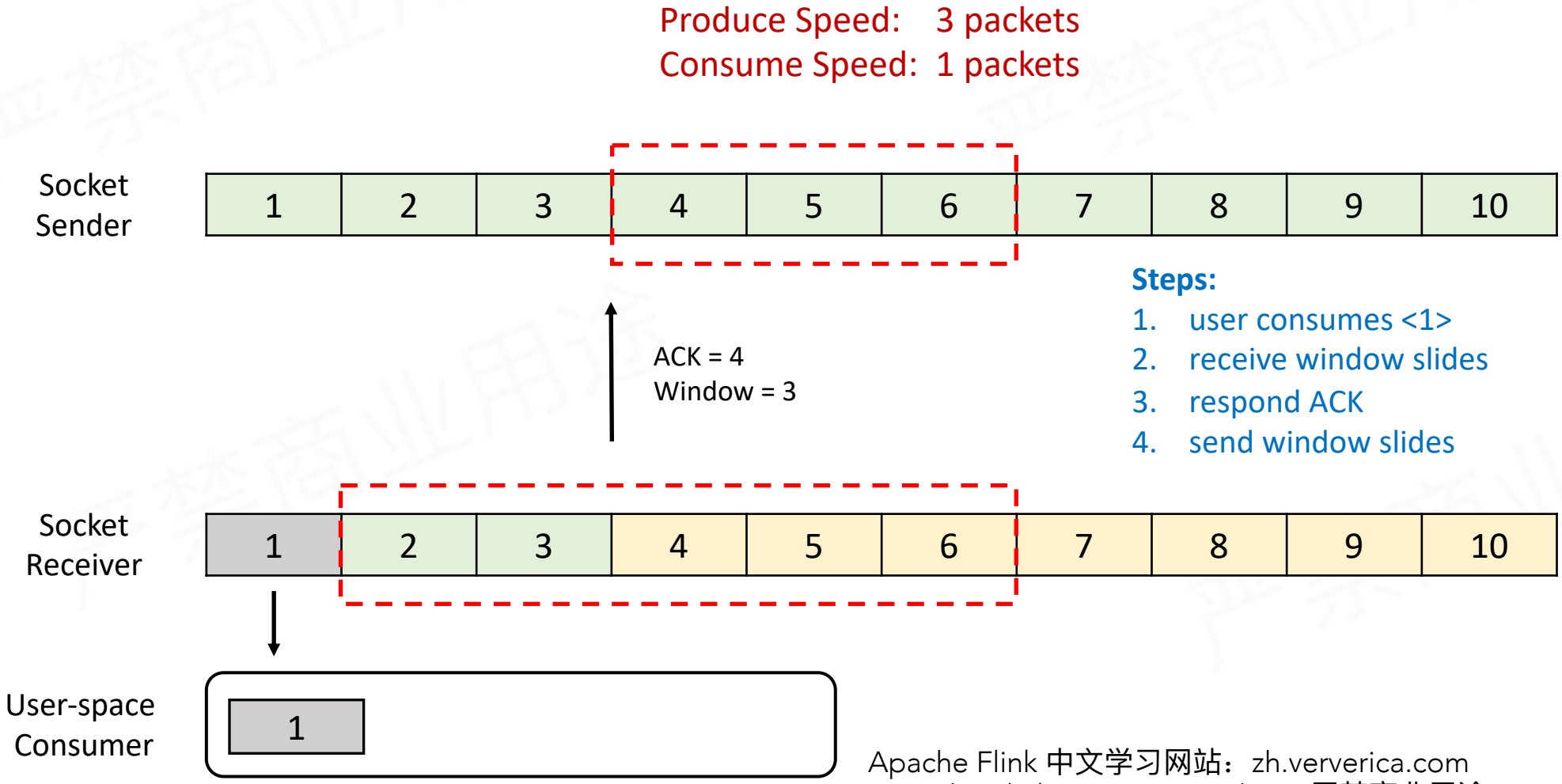


TCP流控：滑动窗口

Produce Speed: 3 packets
Consume Speed: 1 packets

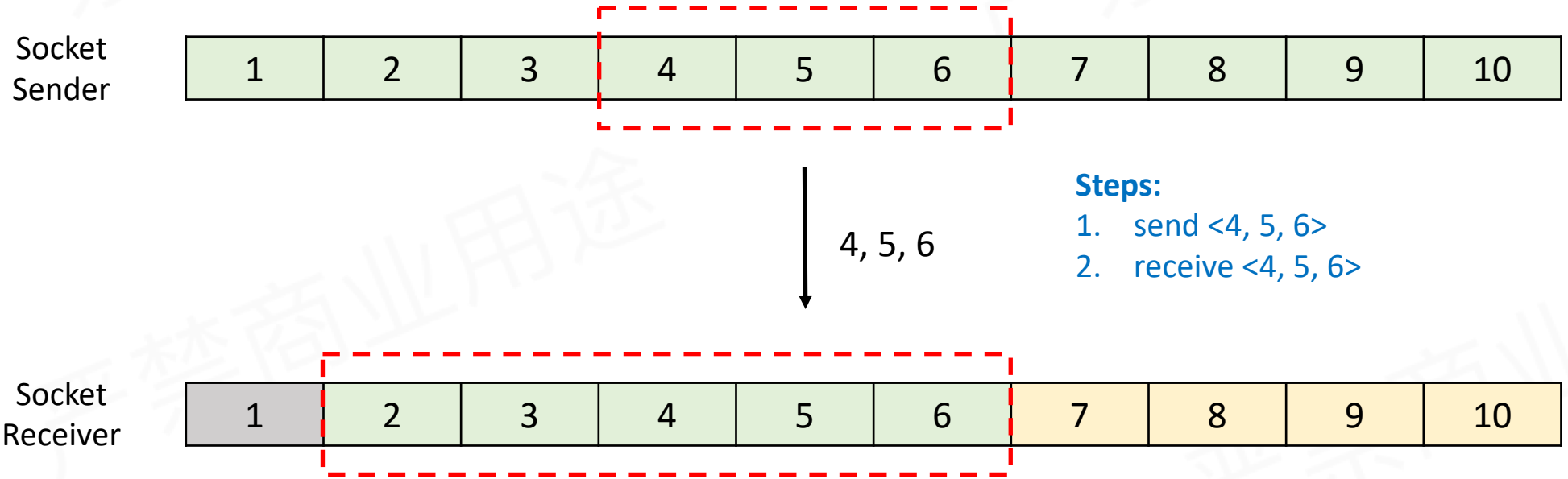


TCP流控：滑动窗口

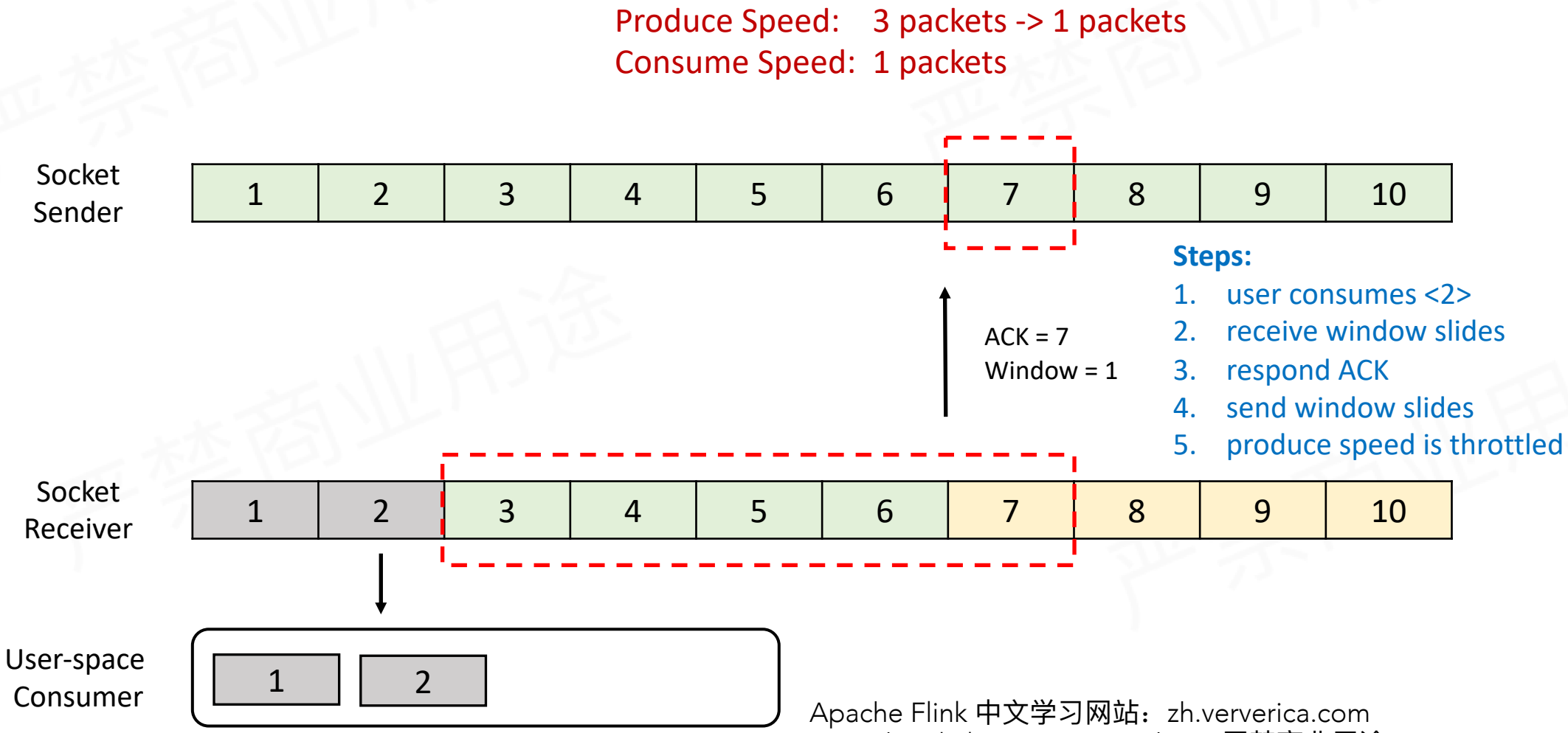


TCP流控：滑动窗口

Produce Speed: 3 packets
Consume Speed: 1 packets

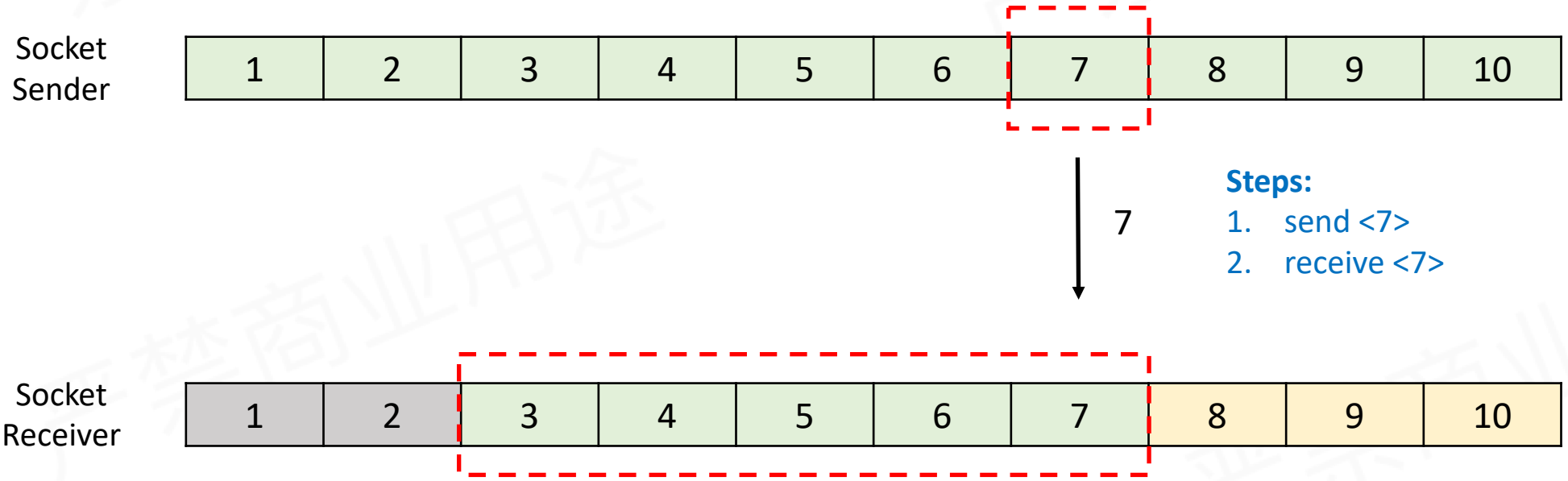


TCP流控：滑动窗口



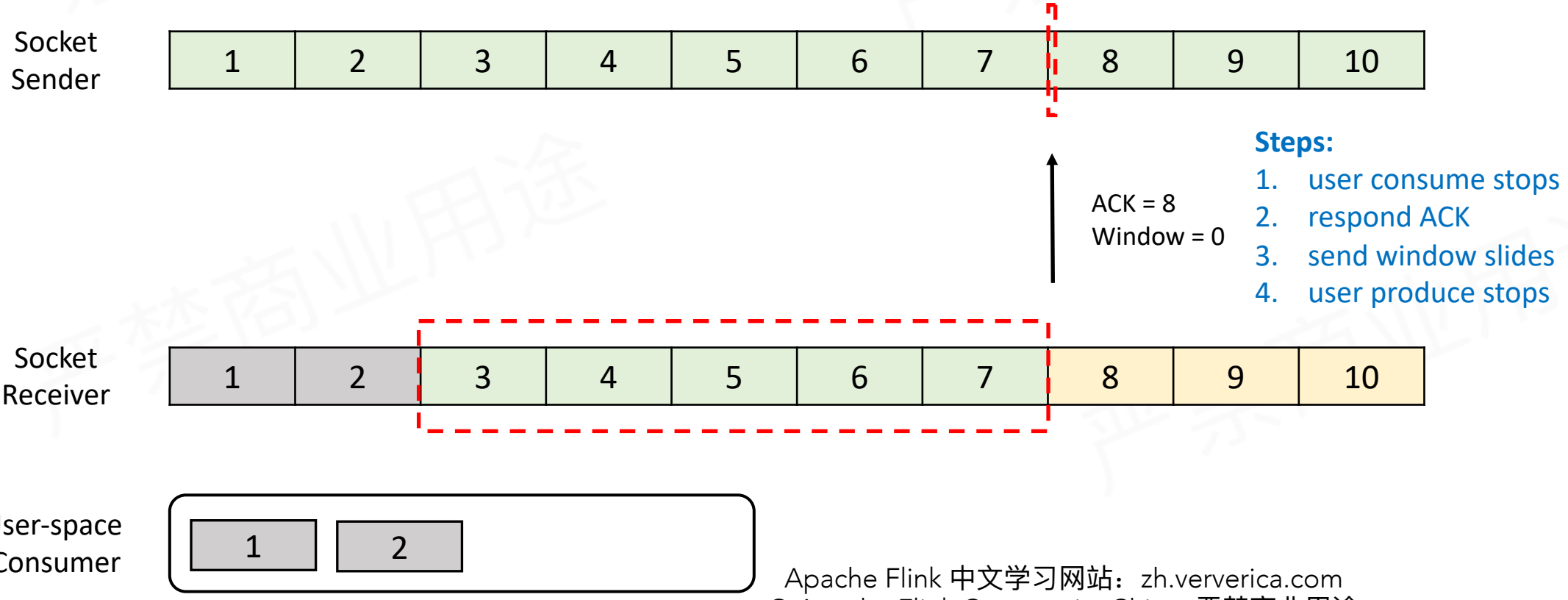
TCP流控：滑动窗口

Produce Speed: 1 packets
Consume Speed: 1 packets



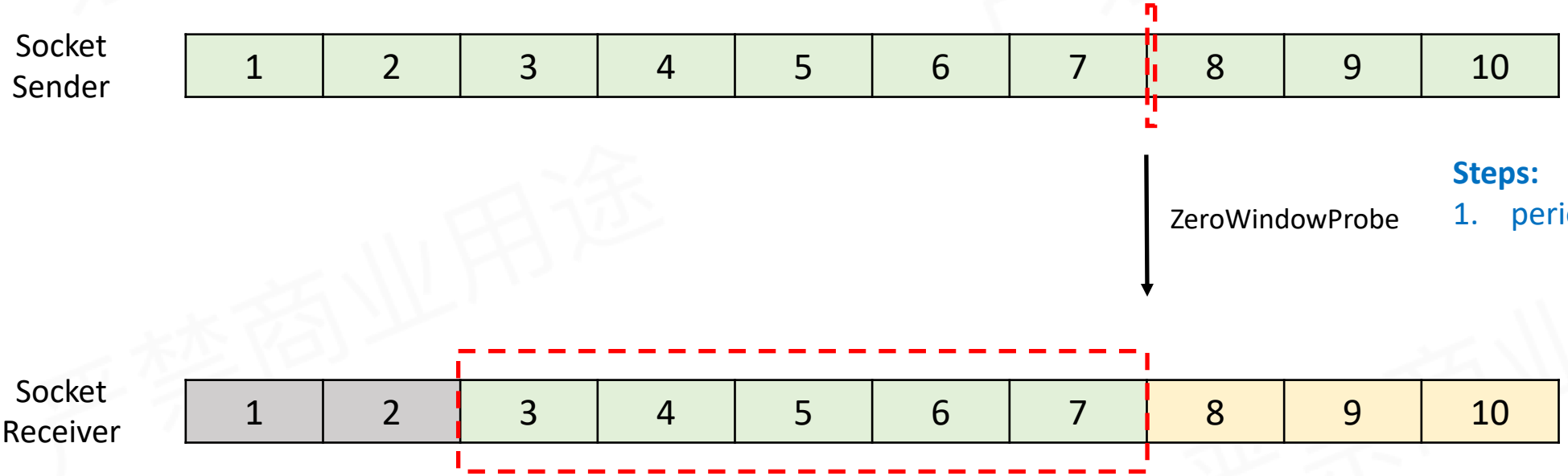
TCP流控：滑动窗口

Produce Speed: 1 packets -> 0 packets
Consume Speed: 1 packets -> 0 packets



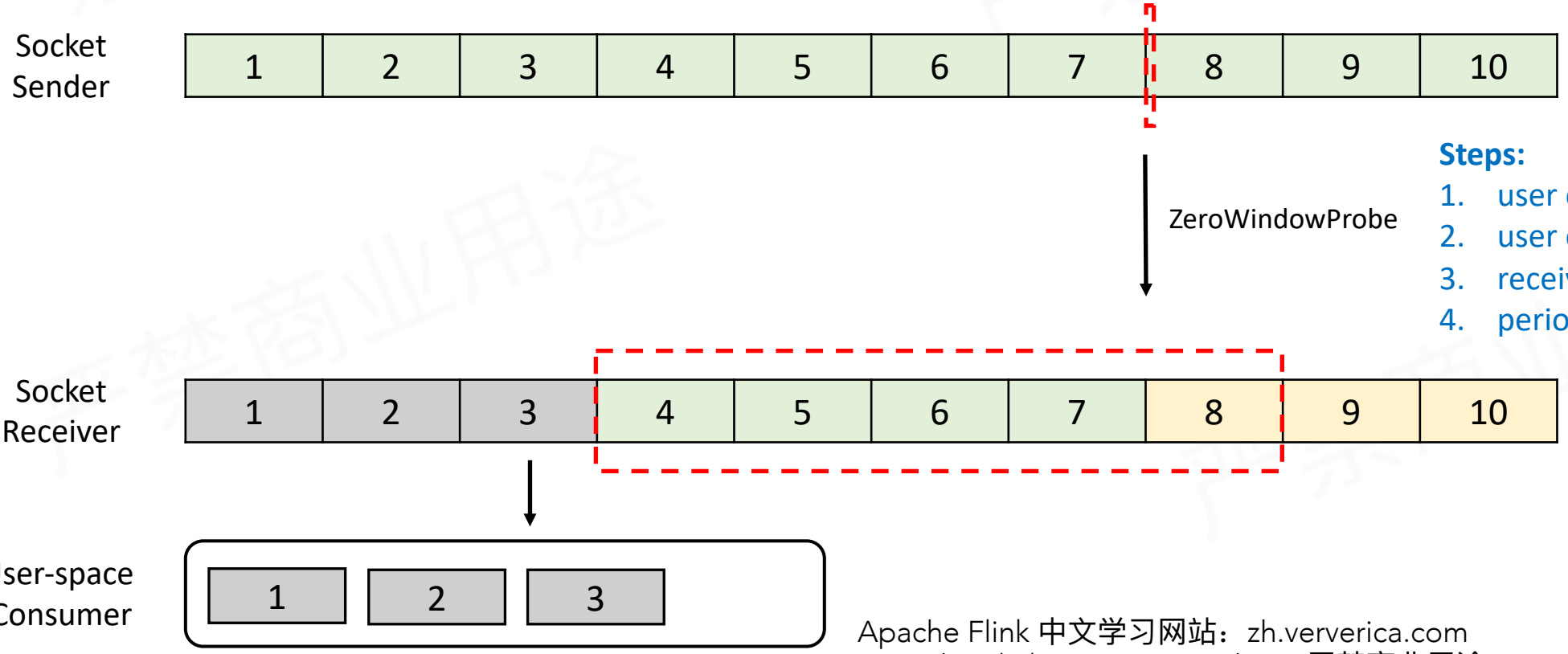
TCP流控：滑动窗口

Produce Speed: 0 packets
Consume Speed: 0 packets



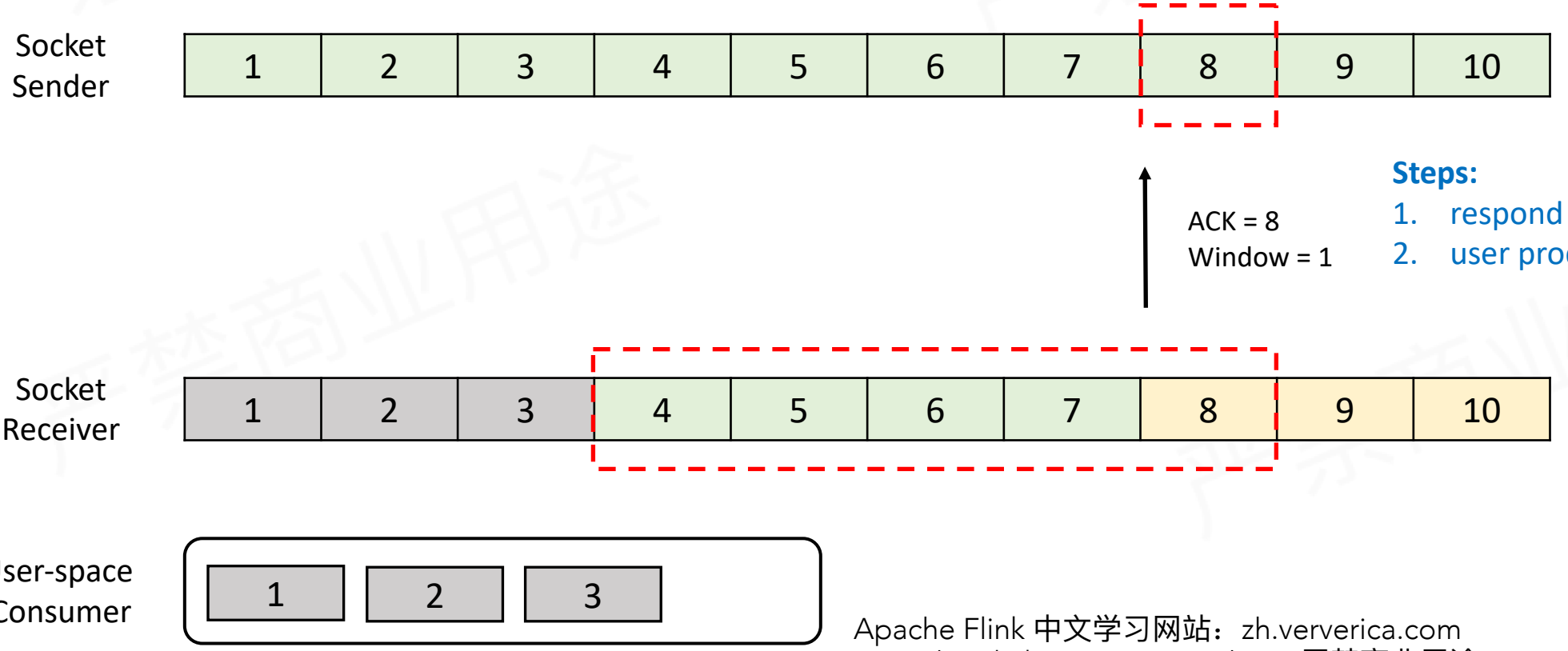
TCP流控：滑动窗口

Produce Speed: 0 packets
Consume Speed: 0 packets -> 1 packets



TCP流控：滑动窗口

Produce Speed: 0 packets -> 1 packets
Consume Speed: 1 packets



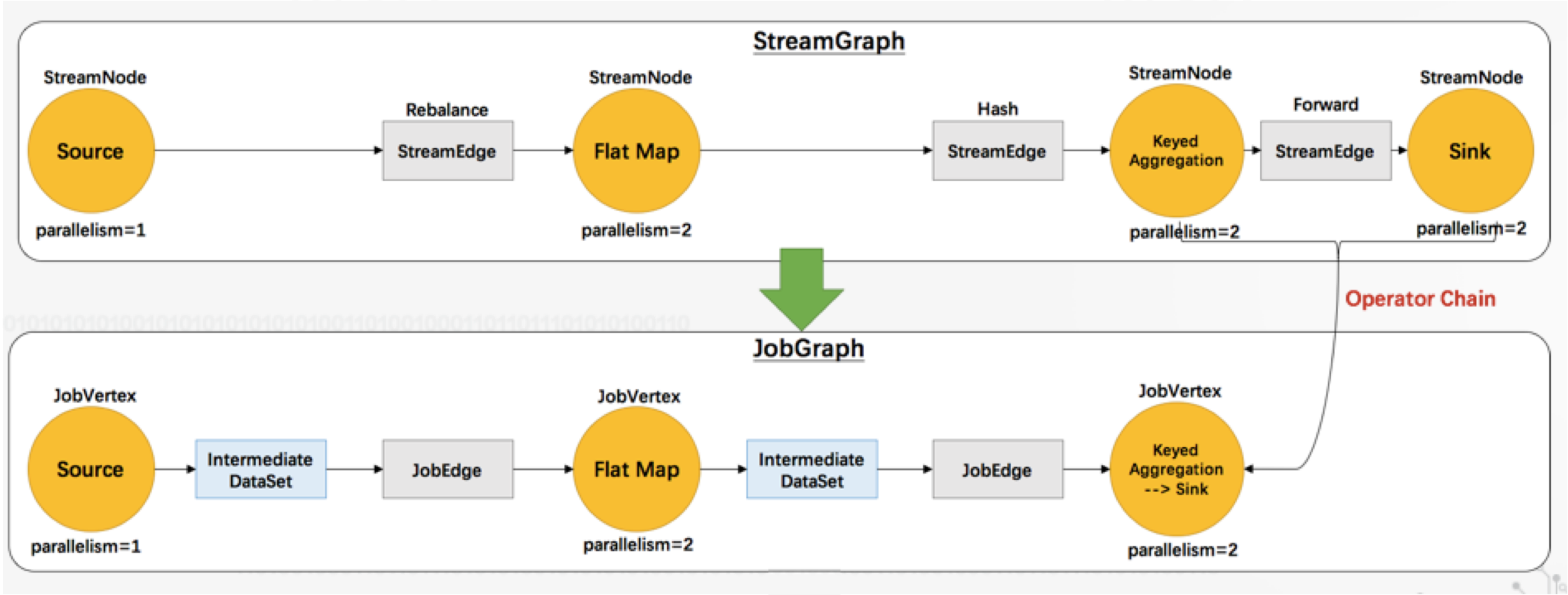
03

Flink TCP-based反压机制(before V1.5)

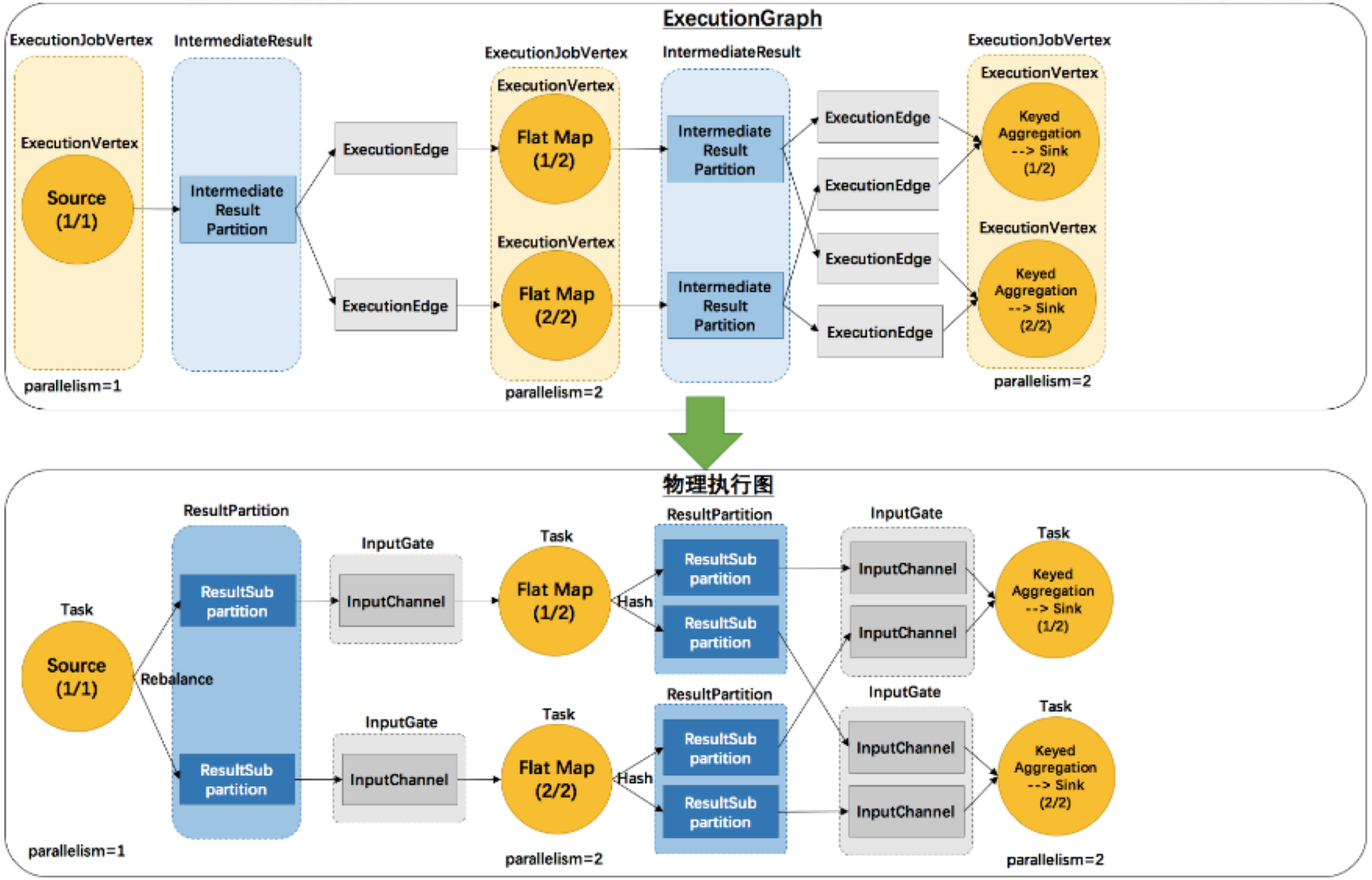
示例：WindowWordCount

```
public static void main(String[] args) throws Exception {

    StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

    DataStream<Tuple2<String, Integer>> dataStream = env
            .socketTextStream("localhost", 9999)
            .flatMap(new Splitter())
            .keyBy(0)
            .timeWindow(Time.seconds(5))
            .sum(1);

    dataStream.print();

    env.execute("Window WordCount");
}
```

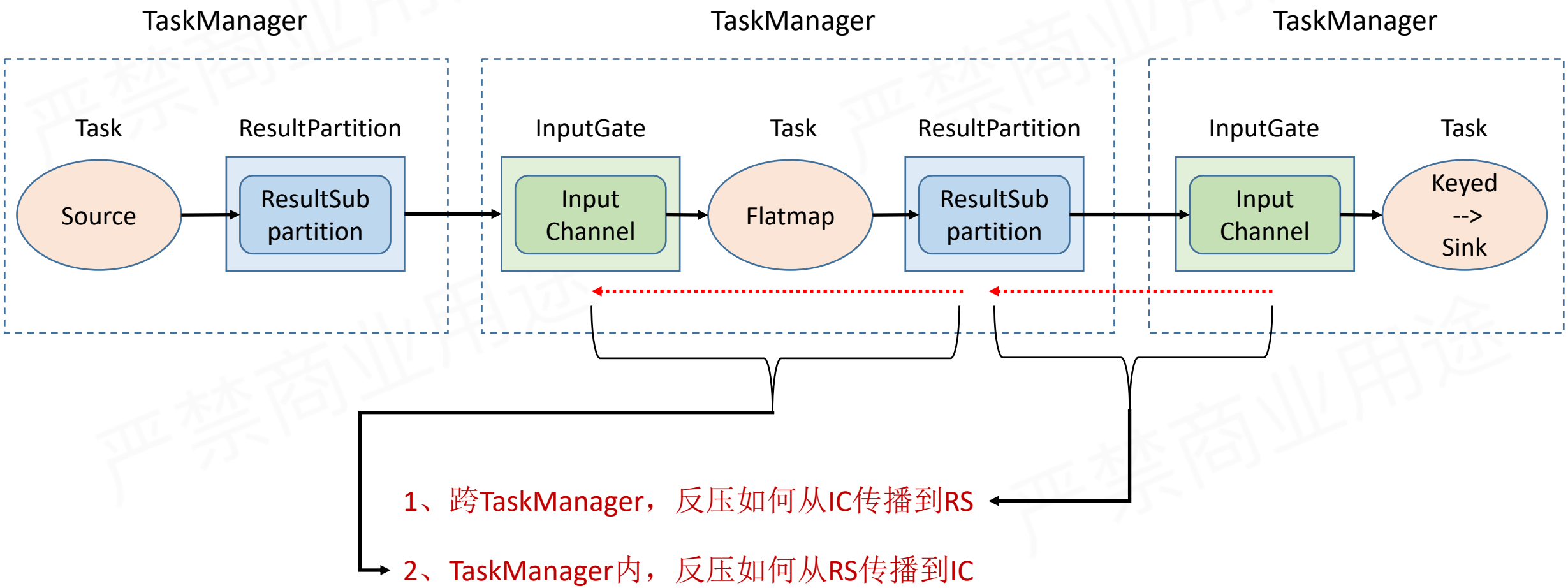
编译阶段：生成JobGraph



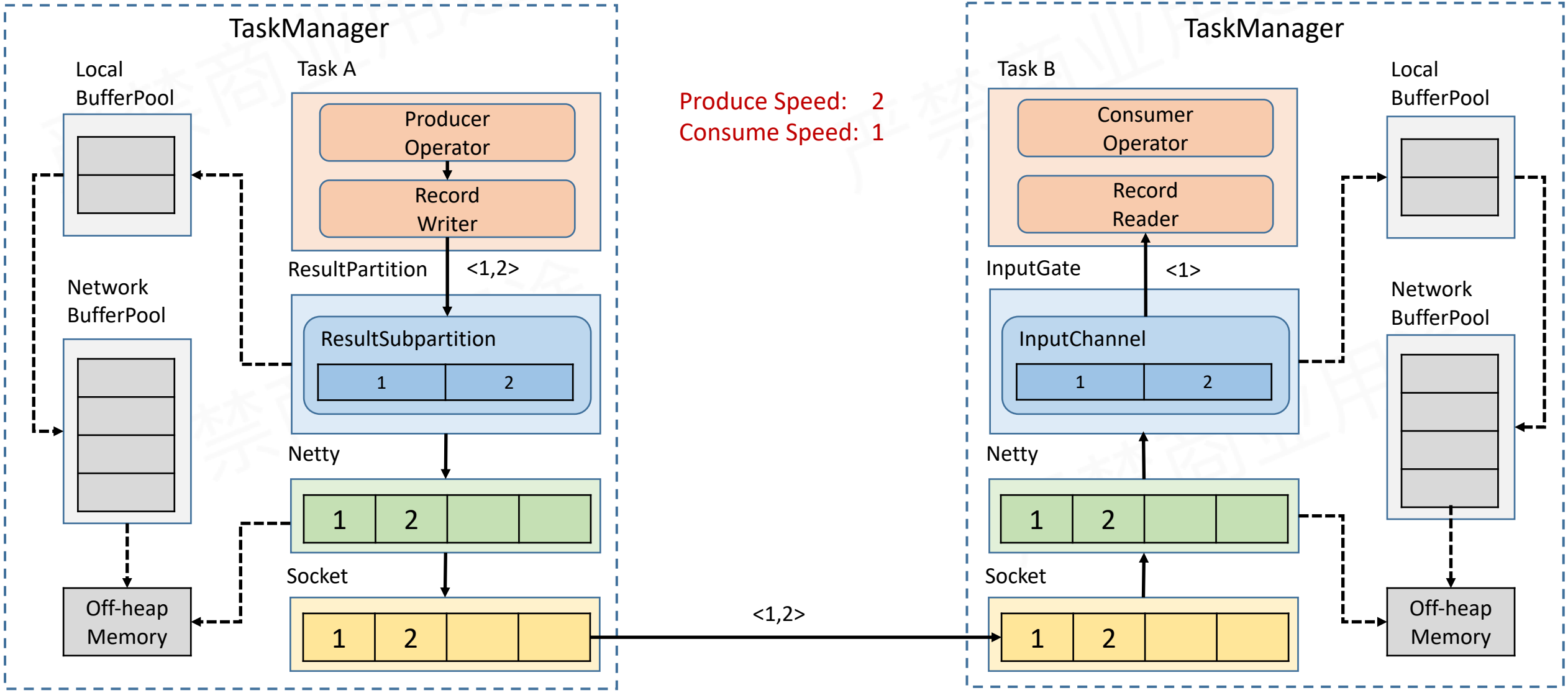
运行阶段：调度ExecutionGraph



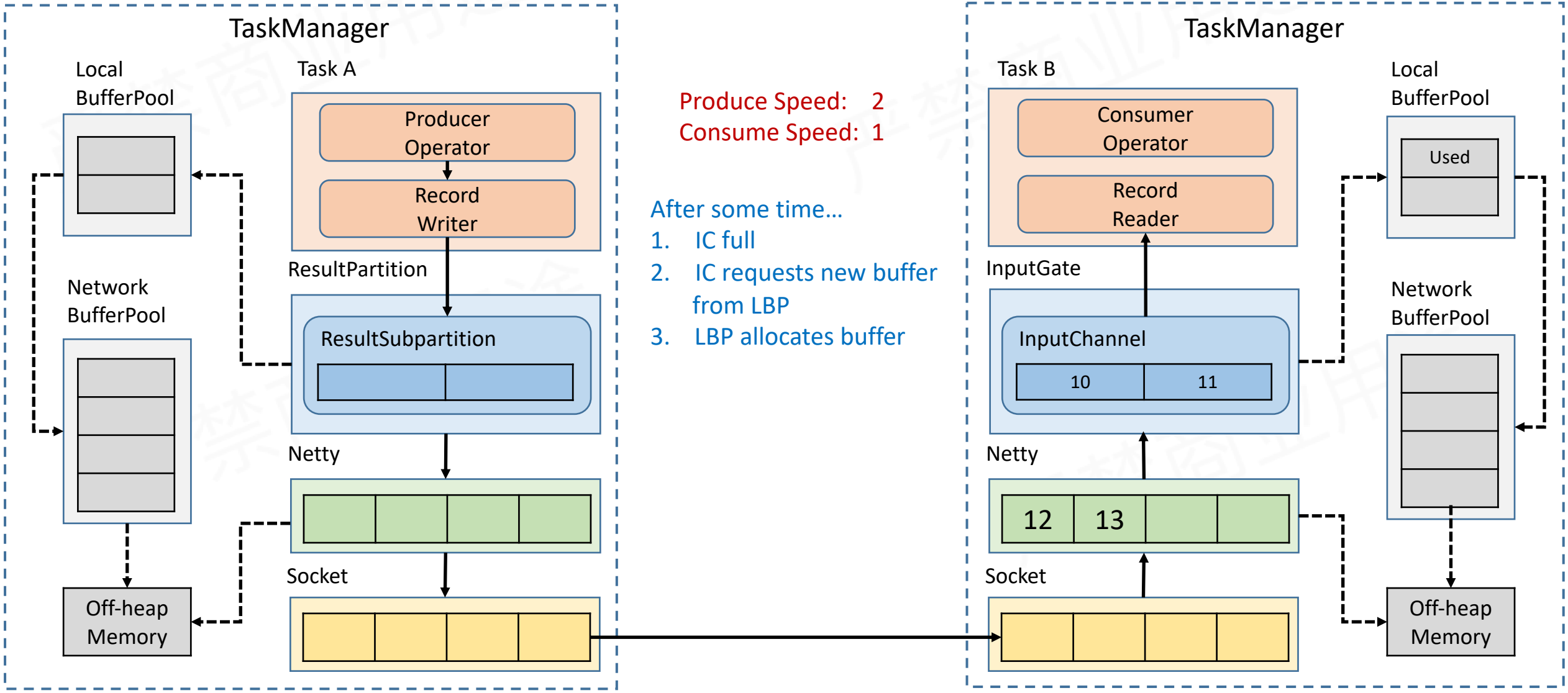
问题拆解：反压传播两个阶段



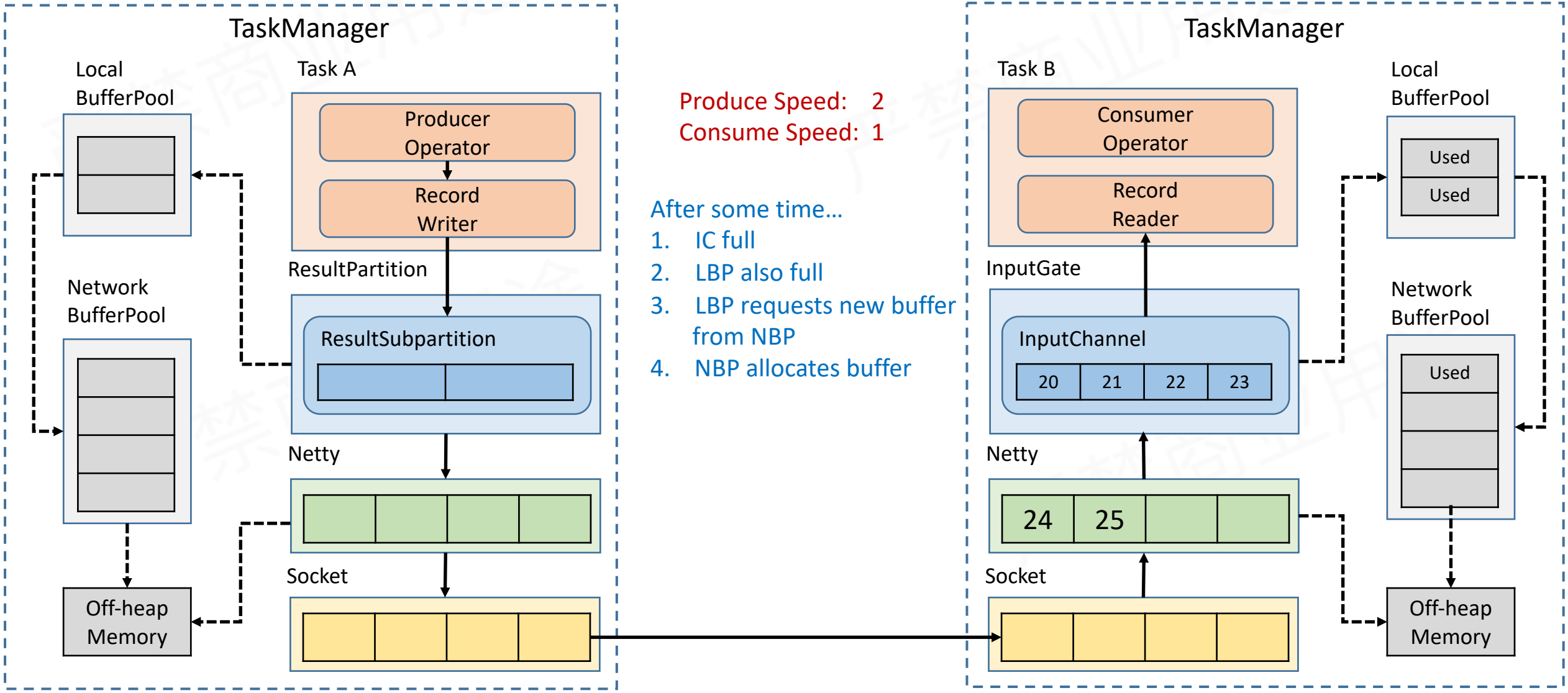
跨TaskManager数据传输



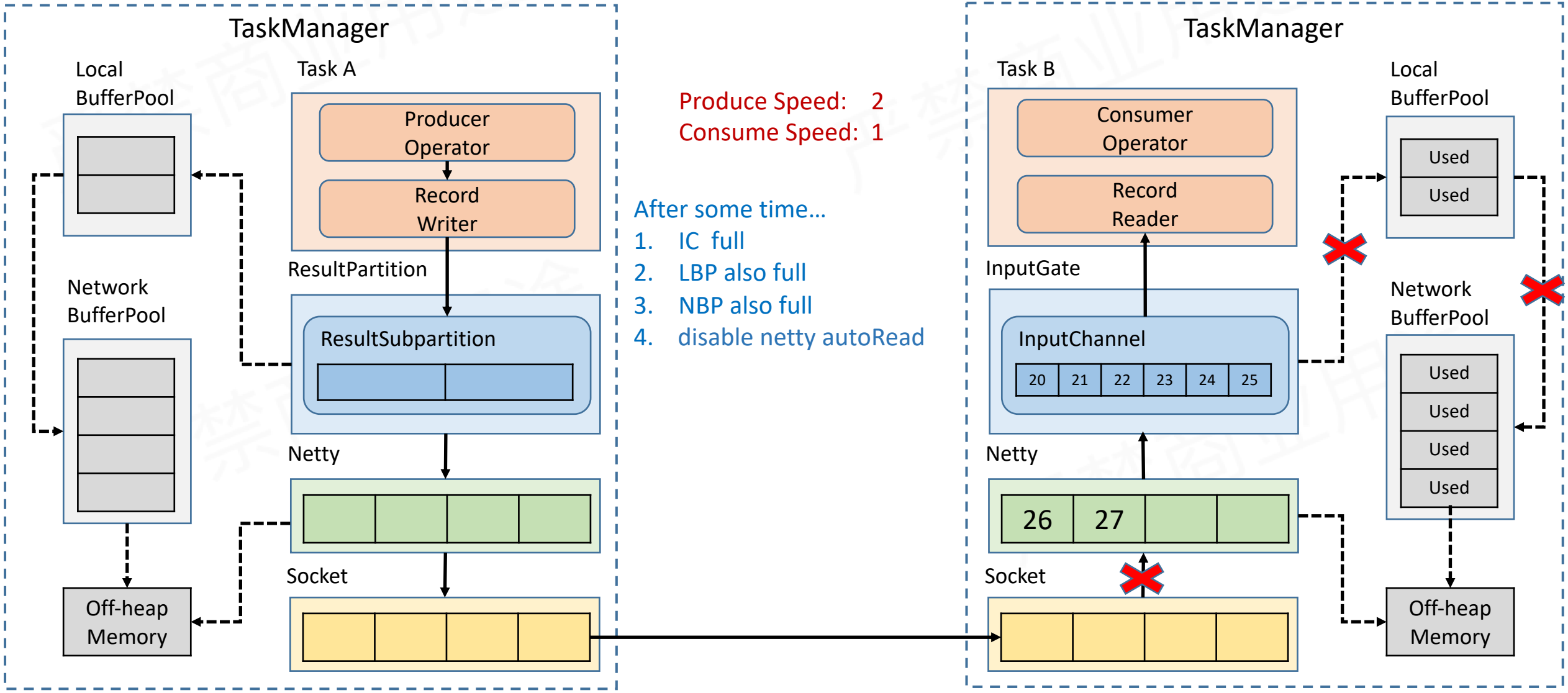
跨TaskManager反压过程



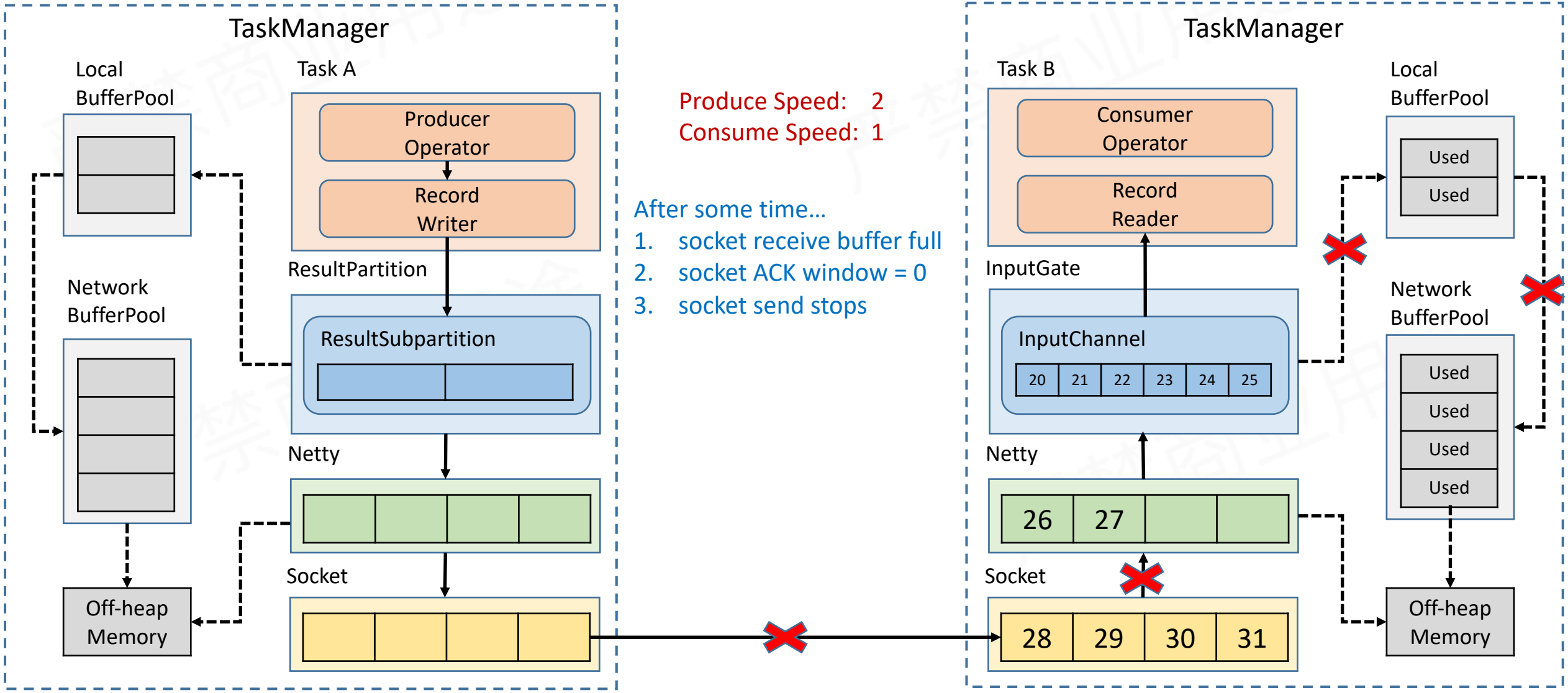
跨TaskManager反压过程



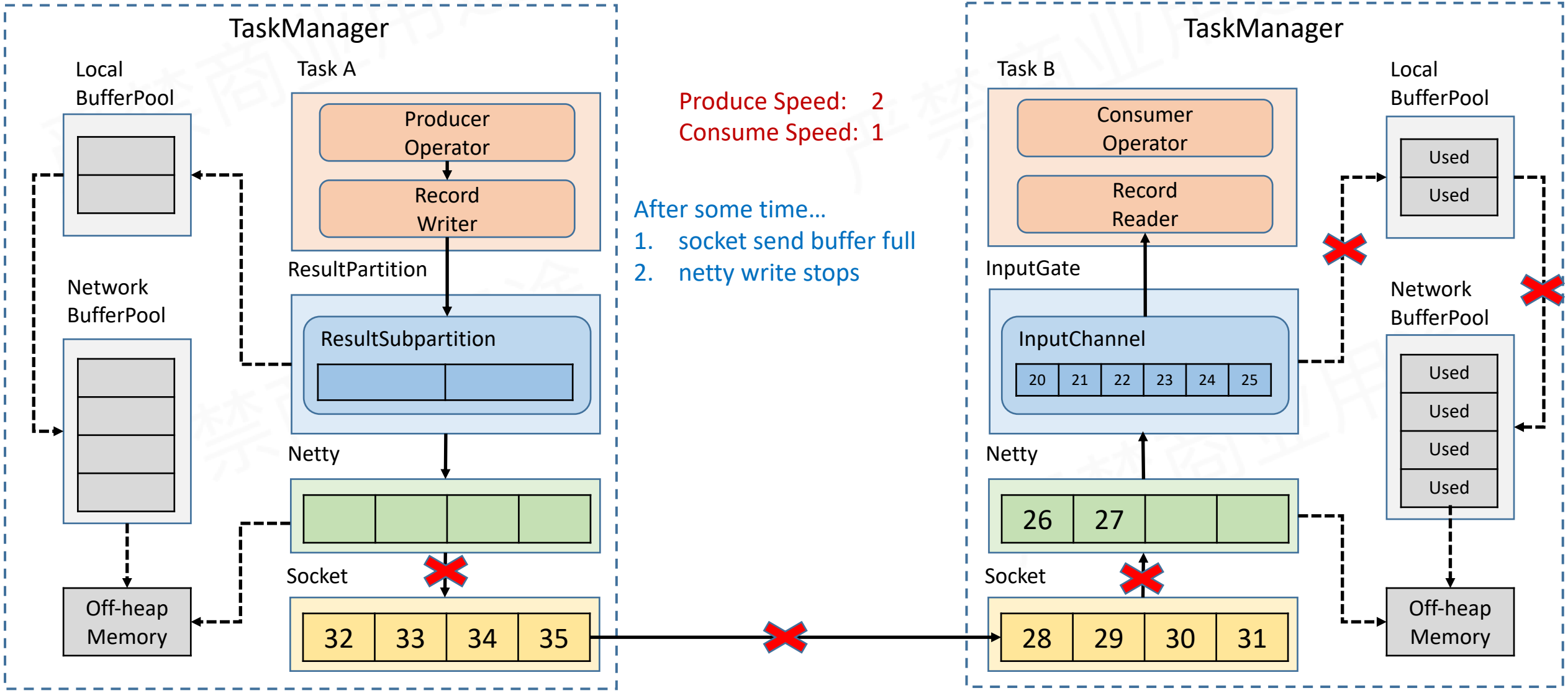
跨TaskManager反压过程



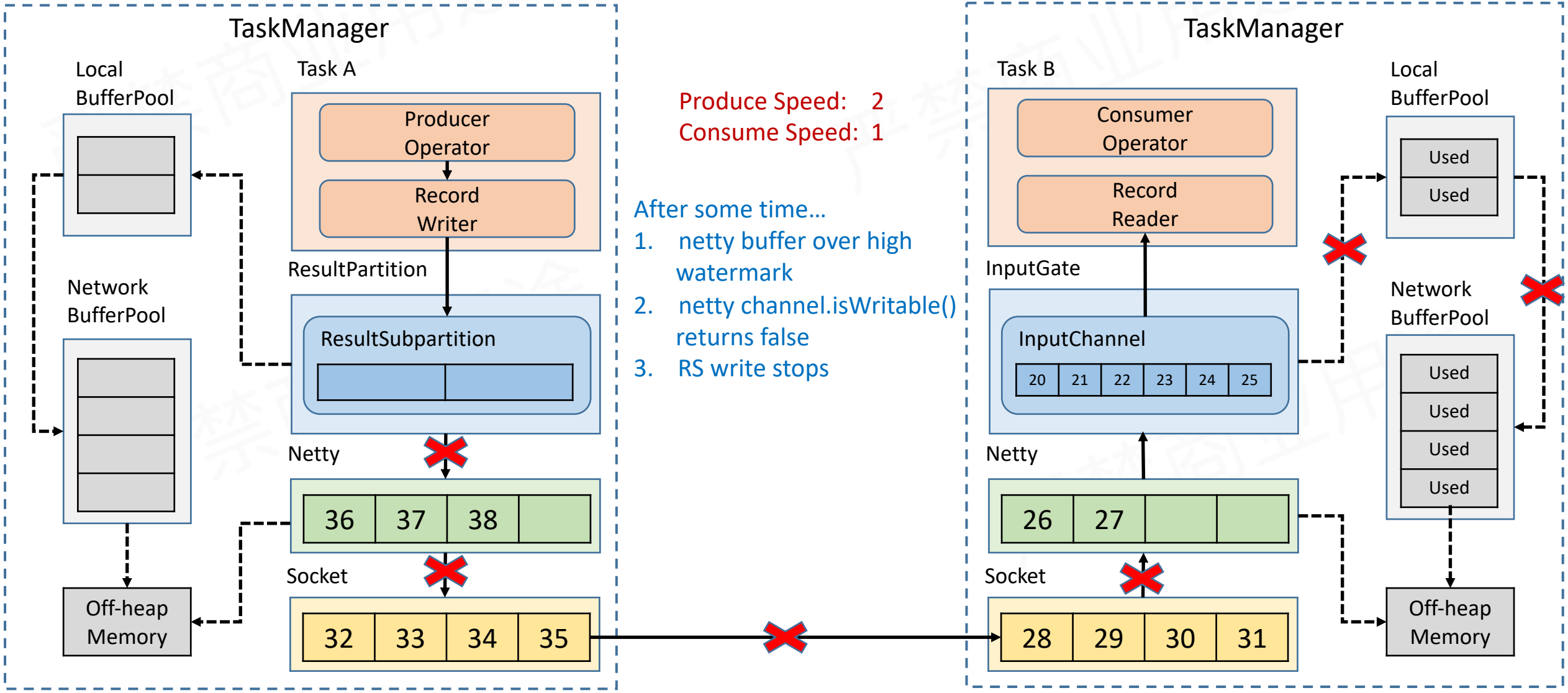
跨TaskManager反压过程



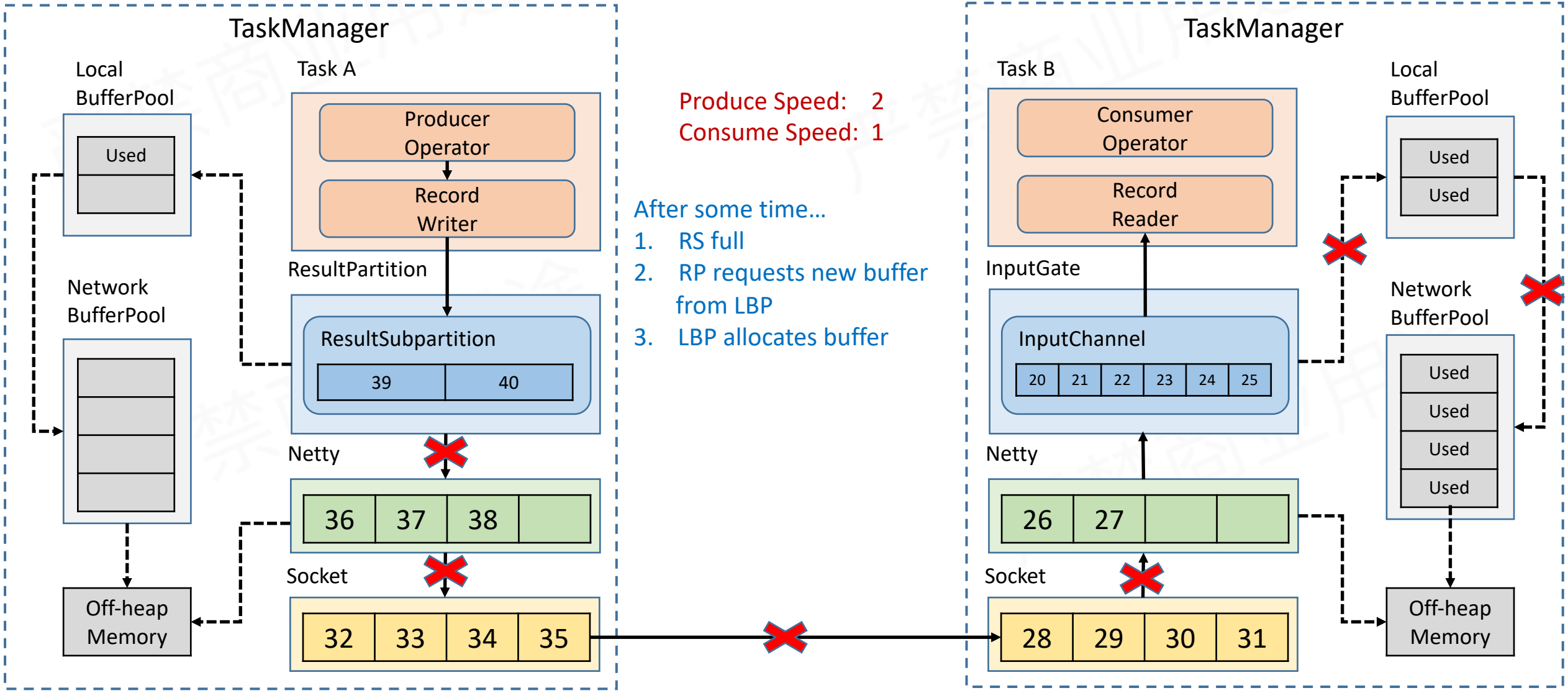
跨TaskManager反压过程



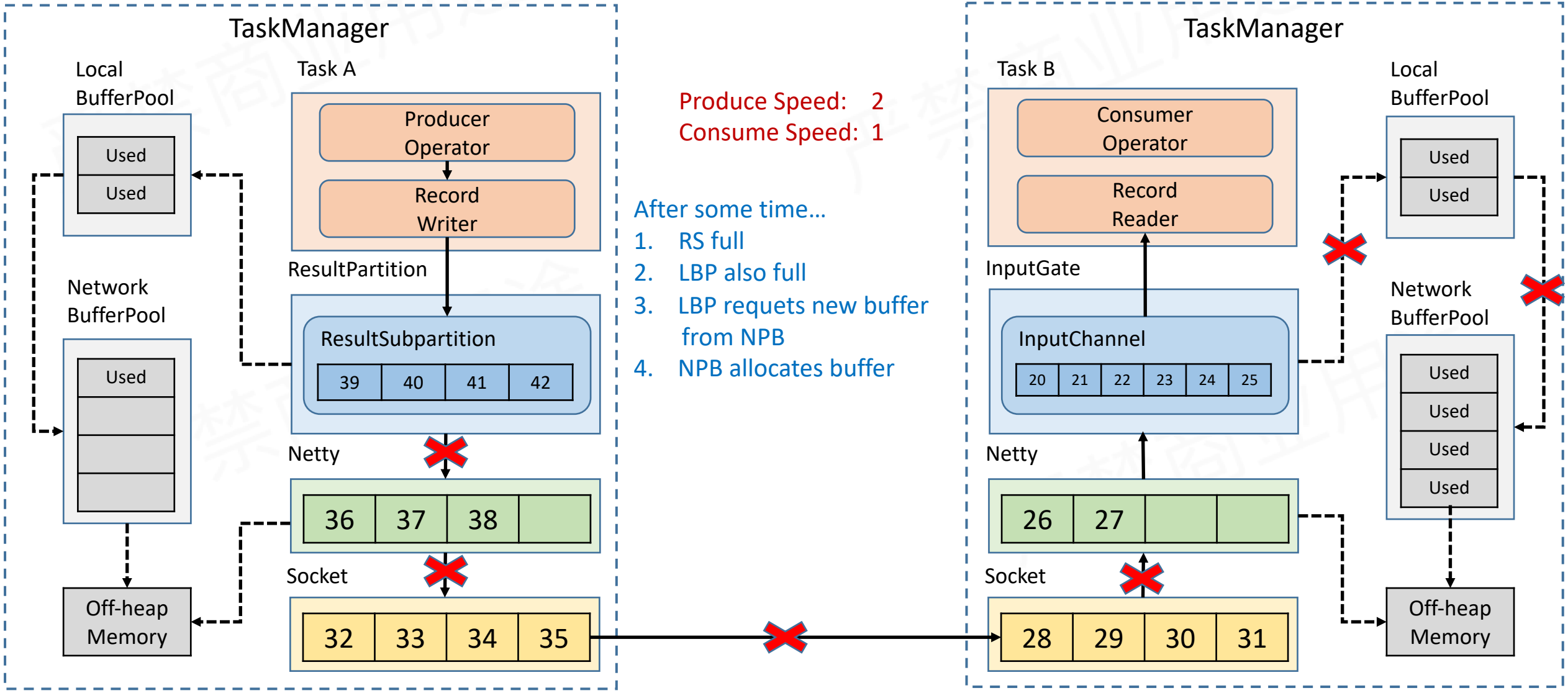
跨TaskManager反压过程



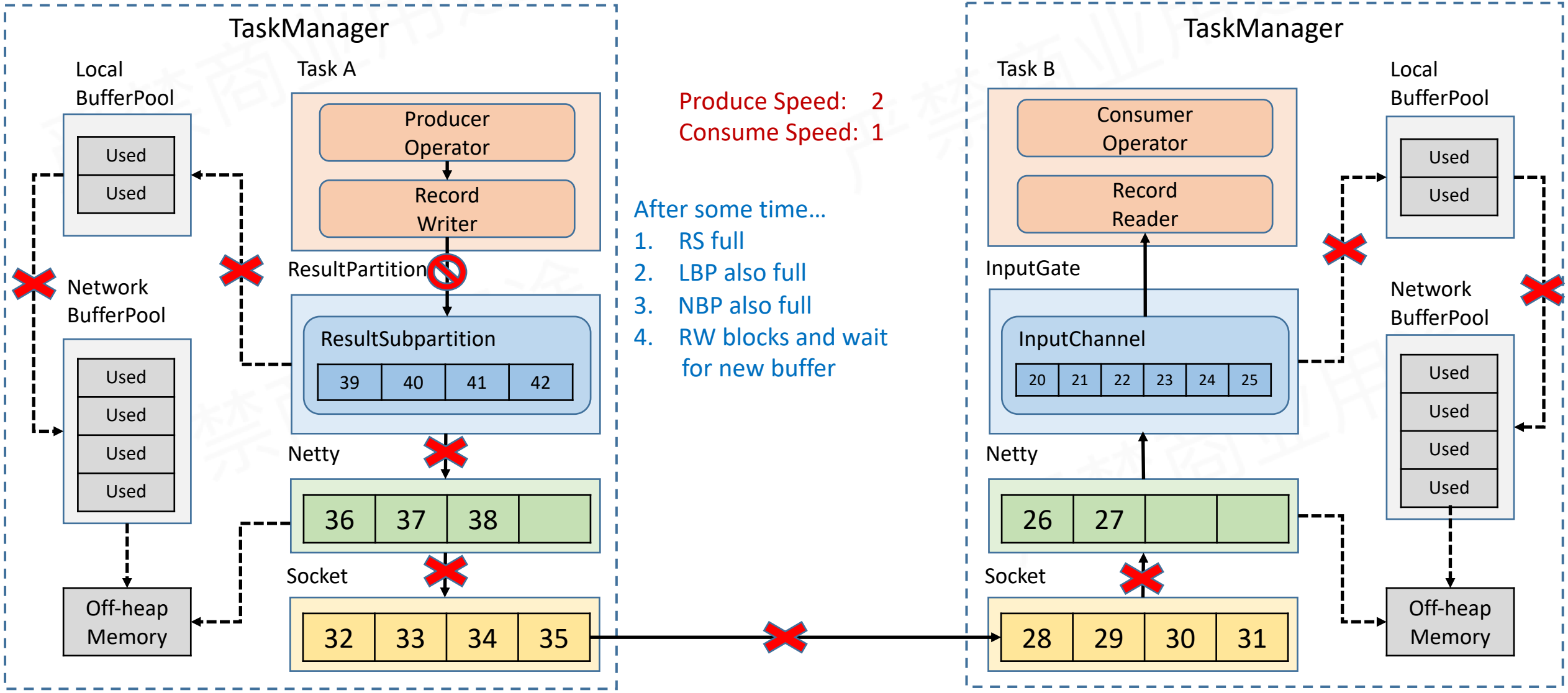
跨TaskManager反压过程



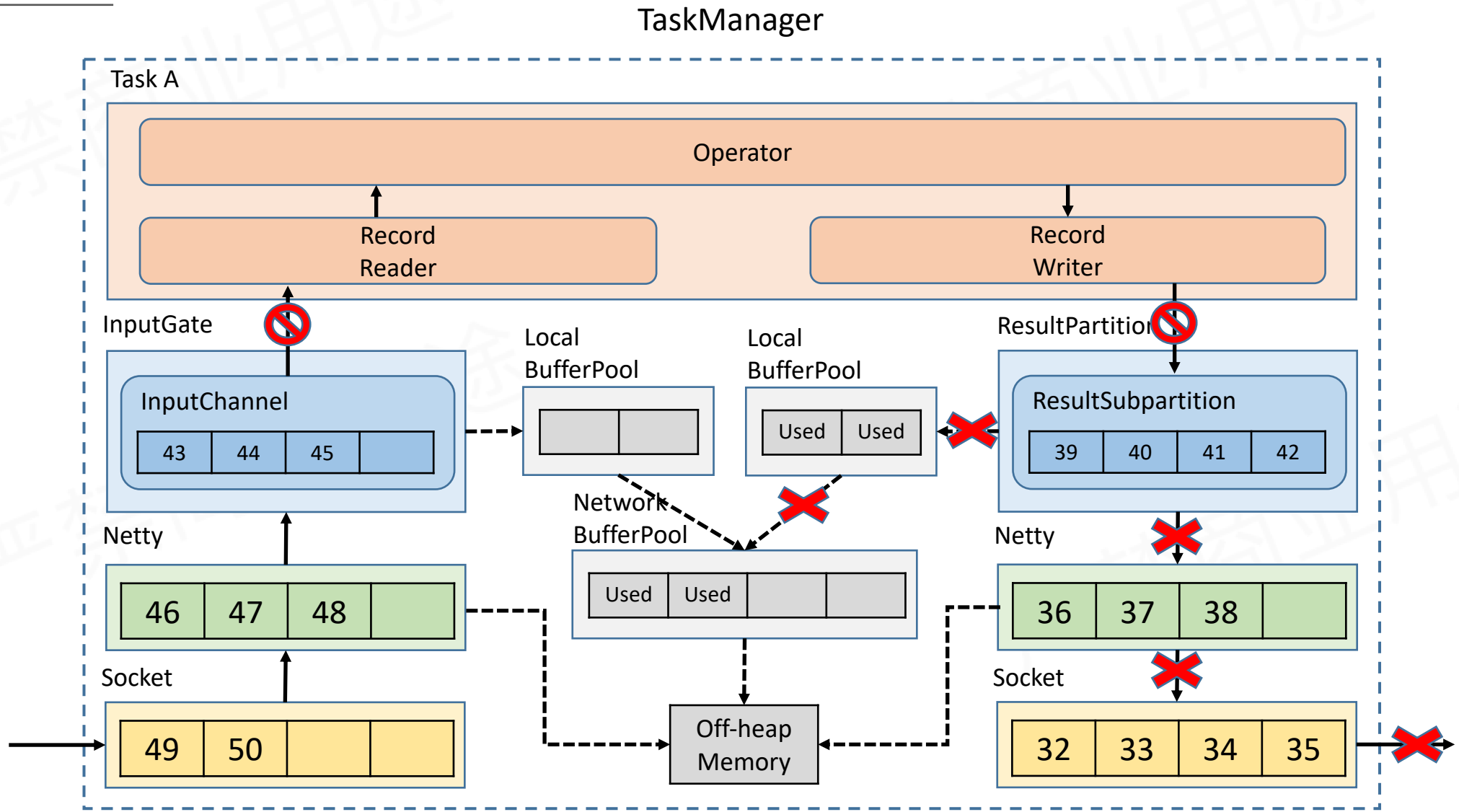
跨TaskManager反压过程



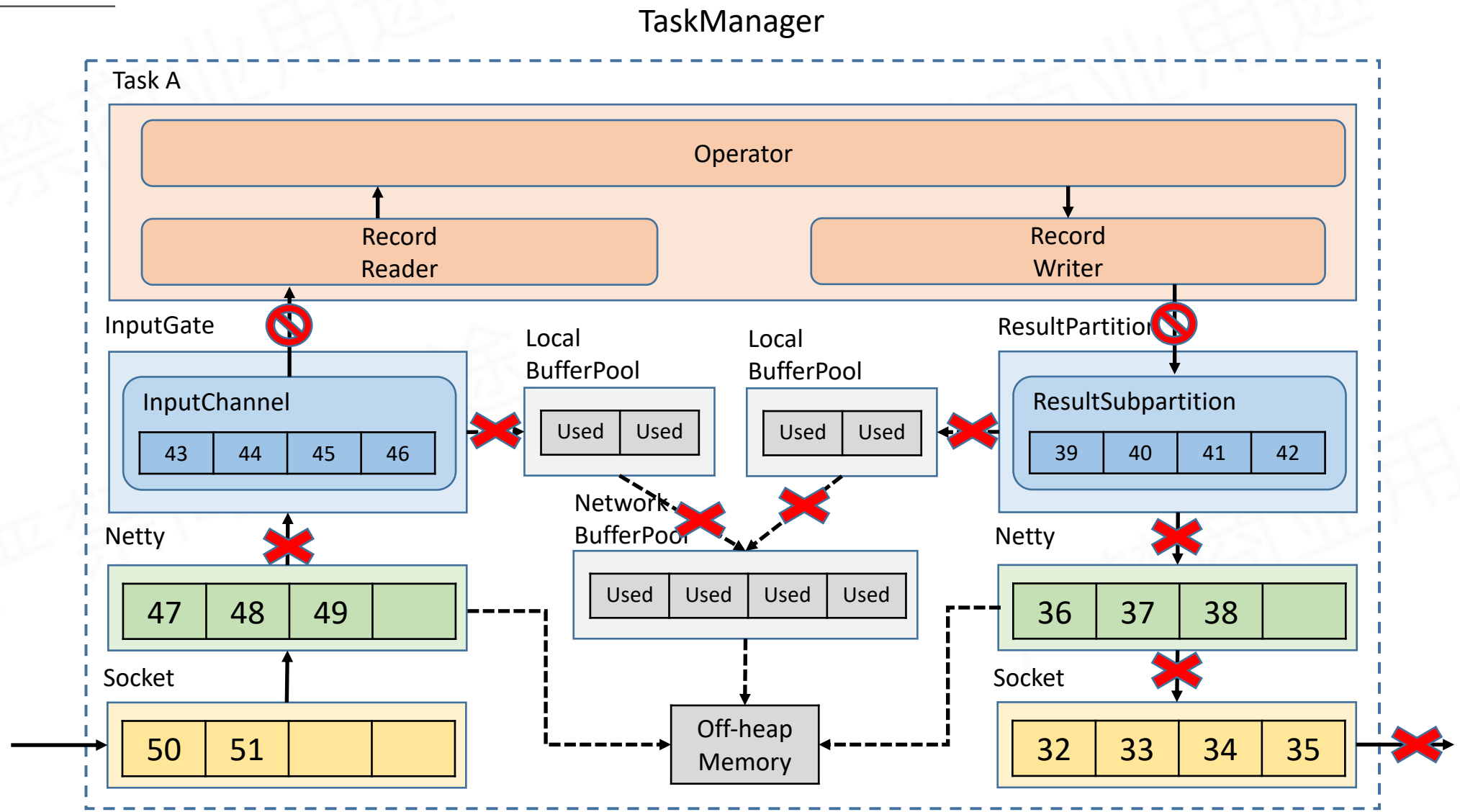
跨TaskManager反压过程



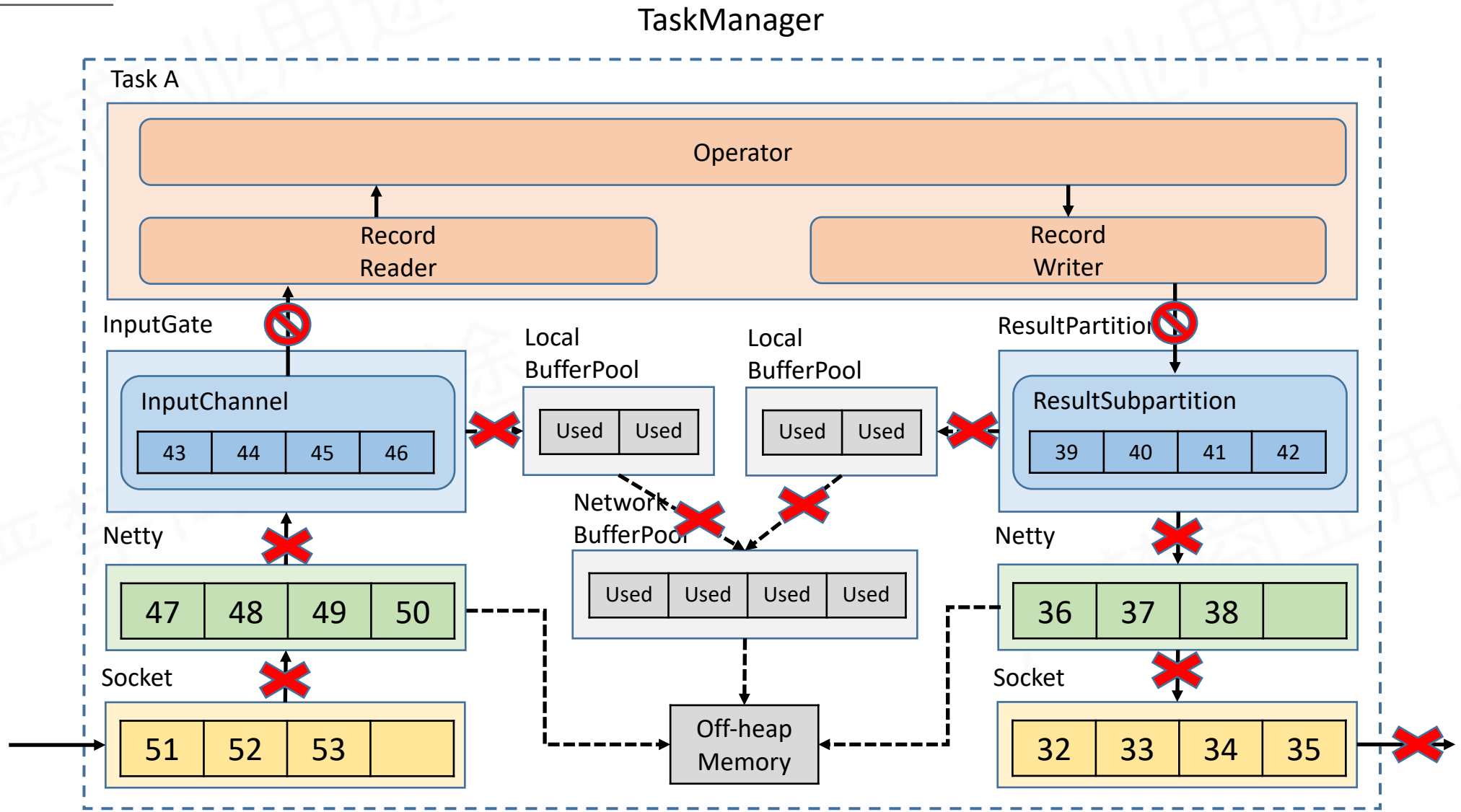
TaskManager内反压过程



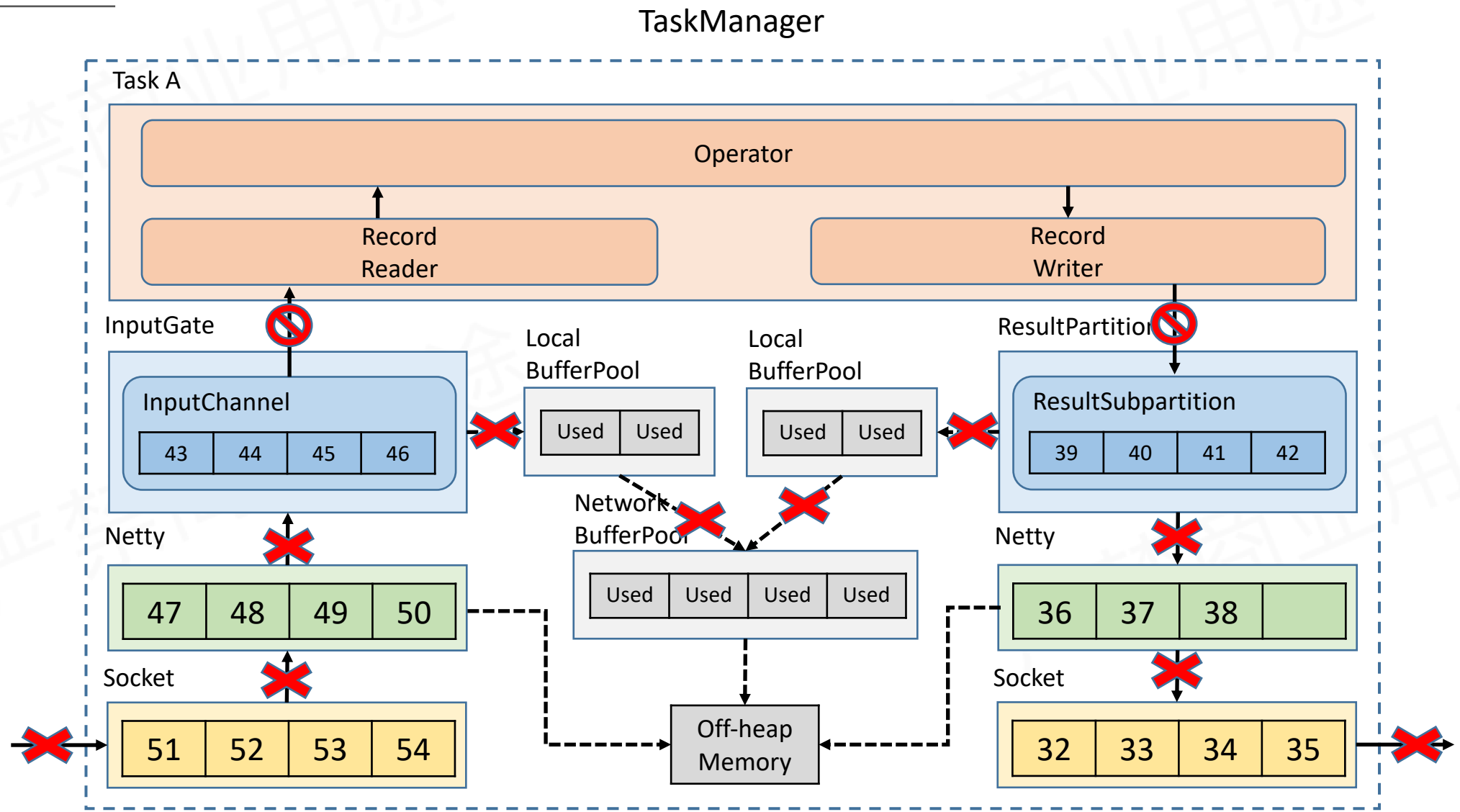
TaskManager内反压过程



TaskManager内反压过程



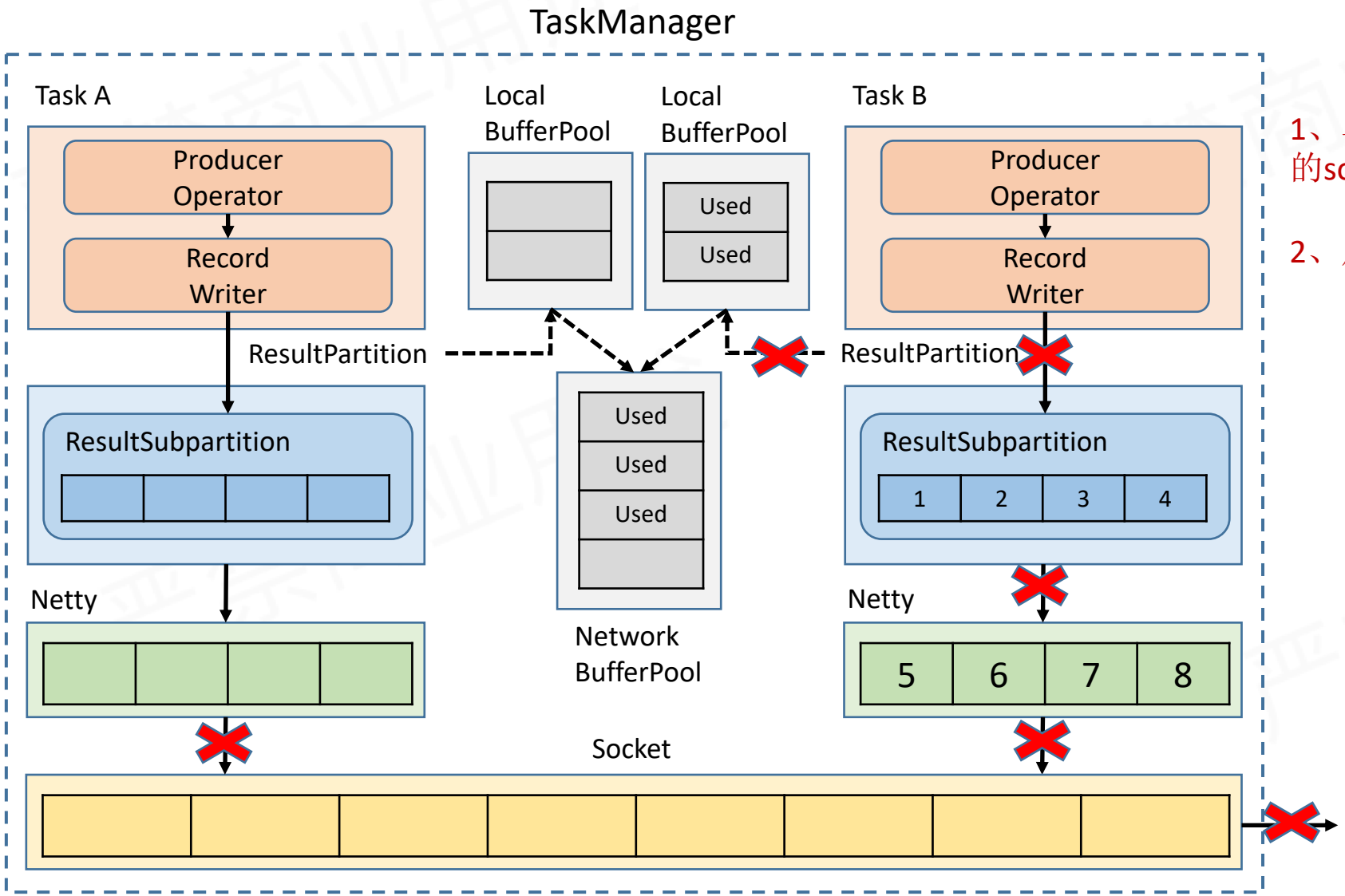
TaskManager内反压过程



04

Flink Credit-based反压机制(since V1.5)

TCP-based反压的弊端



1、单个Task导致的反压，会阻断整个TM-TM的socket，连checkpoint barrier也无法发出

2、反压传播路径太长，导致生效延迟比较大

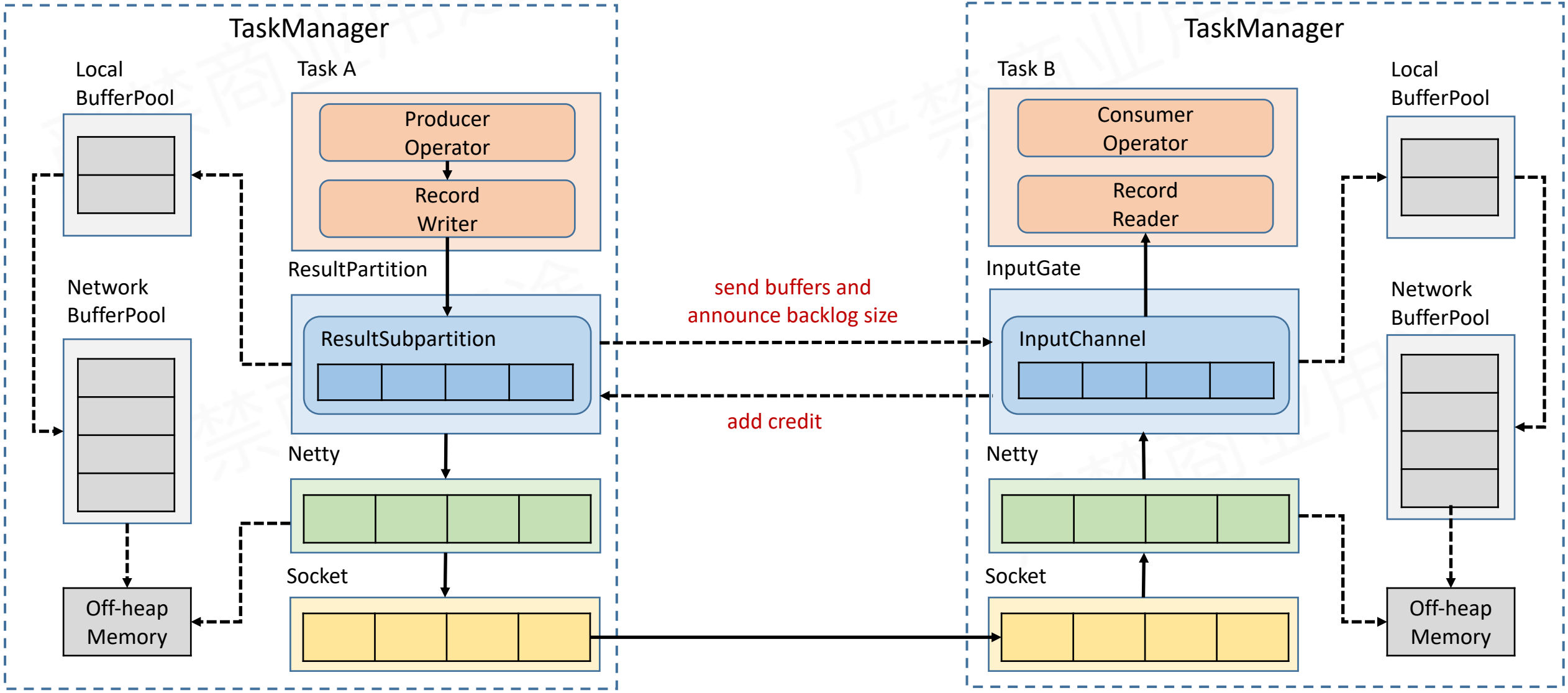
引入Credit-based反压

在Flink层面实现类似TCP流控的feedback机制，

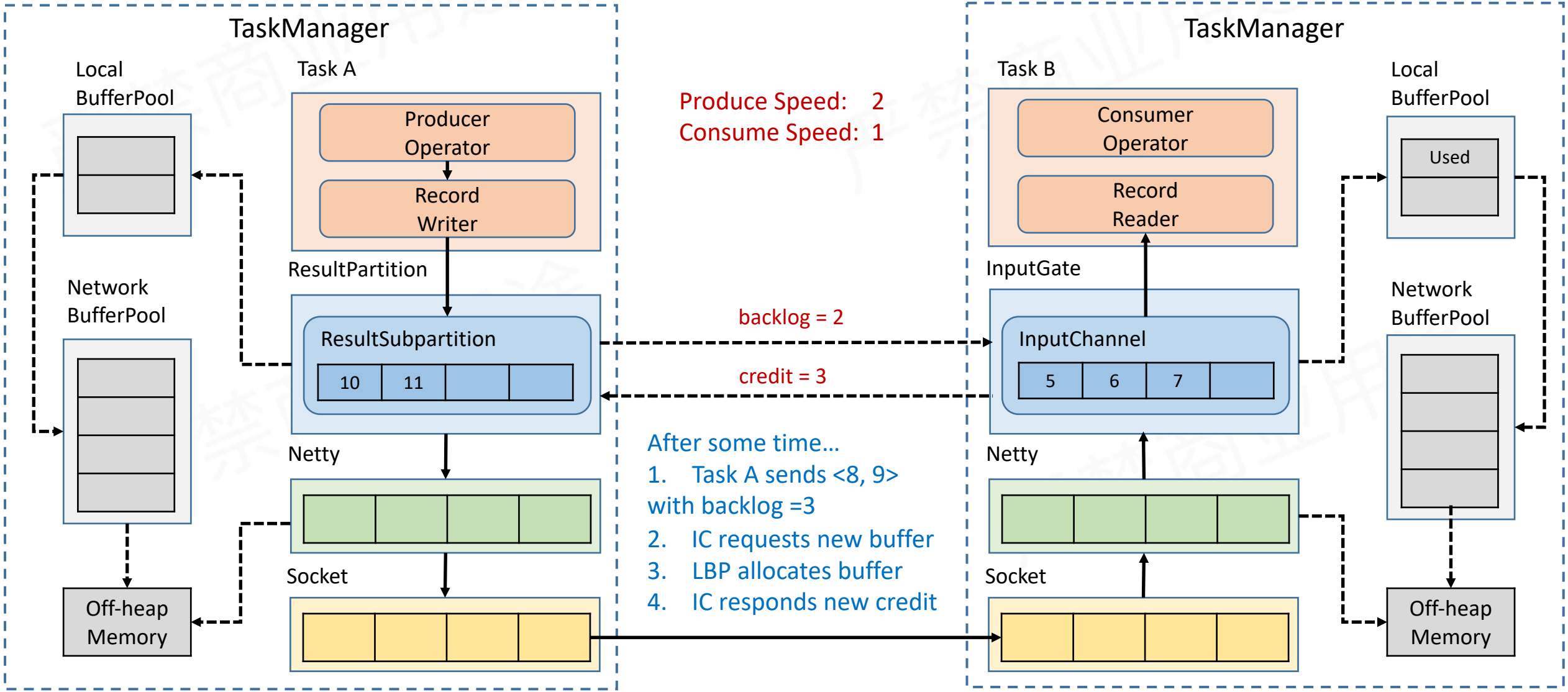
credit可类比为TCP window

Apache Flink

Credit-based反压

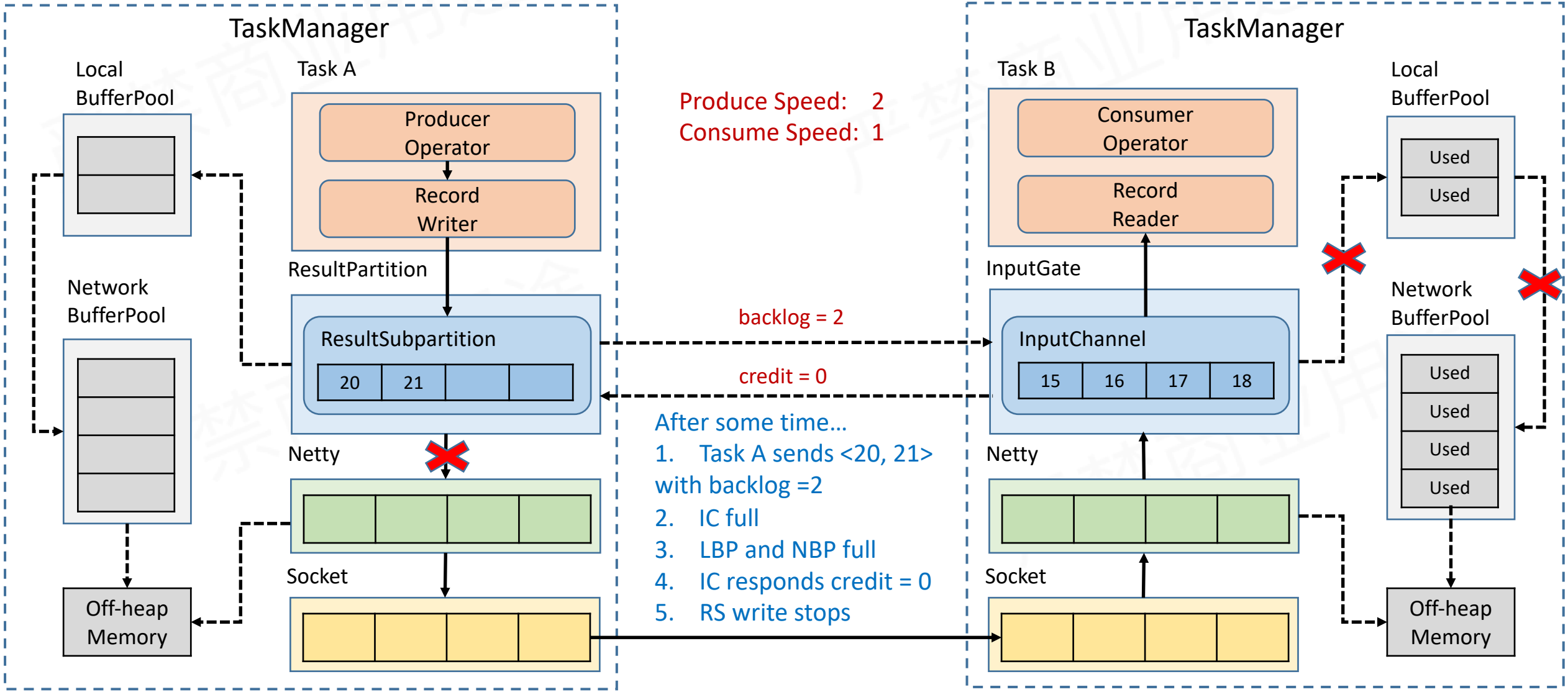


Credit-based反压过程



Apache Flink

Credit-based反压过程



05

总结与思考

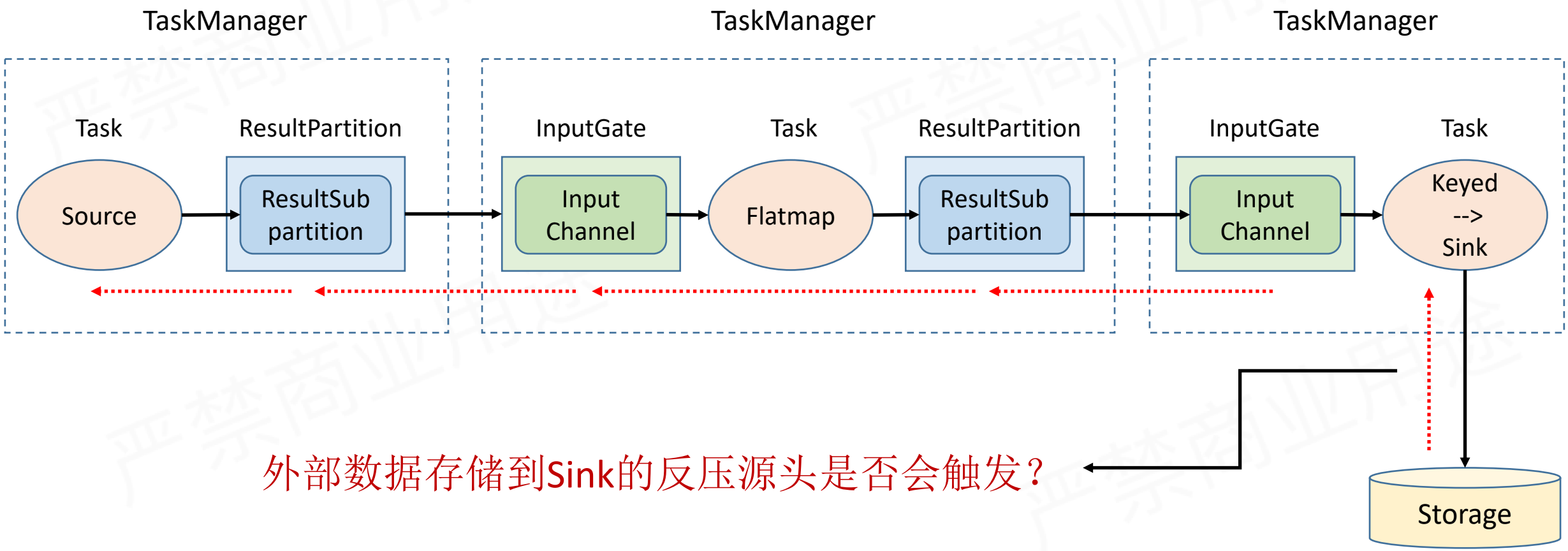
总结

- 网络流控是为了在上下游速度不匹配的情况下，如何防止下游出现过载
- 网络流控有静态限速和动态反压两种手段
- Flink 1.5以前是基于TCP流控+bounded buffer来实现反压
- Flink 1.5之后实现了自己托管的credit-based流控机制，在应用层模拟TCP流控的机制

思考

有了动态反压，静态限速是不是完全没有作用了？

反压不一定会触发



THANKS

【2群】Apache Flink C...

扫一扫群二维码，立刻加入该群。