|  |
| --- |
| [键入公司名称] |
| 上期全挡行情 |
| my\_quote\_shfe\_my\_jr\_ex-spd\_lib |

|  |
| --- |
| wangying  2017/4/24 |

## 丢包测试用例

## 丢包场景

* 一个行情批次

1. 丢掉了一个行情批次中的买方向前面一些数据包
2. 丢掉一个行情批次中一些买方向数据中的中间数据包
3. 丢掉一个行情批次中买方向数据的后面一些数据包
4. 丢掉一个行情批次中中间一些买卖数据包
5. 丢掉一个行情批次中卖方向的开头一些数据包
6. 丢掉一个行情批次中卖方向数据中间一些数据包
7. 丢掉一个行情批次中卖方向数据后边一些数据包
8. 一个行情批次中所有卖方向数据丢包
9. 一个行情批次中部分买和所有卖方向数据丢包
10. 一个行情批次中所有买方向和部分卖数据丢包
11. 一个行情批次中所有买方向数据包丢失

* 跨行情批次

1. 跨批次丢失了中间部分的卖方向和部分买方向数据
2. 跨批次丢失了中间全部的卖方向和部分买方向数据
3. 跨批次丢失了中间部分的卖方向和全部买方向数据
4. 跨批次丢失了中间全部的卖方向和全部买方向数据
5. 跨批次丢失了中间部分的买卖方向和买方向数据
6. 跨批次丢失了中间部分的买卖方向和全部买方向数据
7. 跨批次丢失了中间部分的买卖方向和买卖方向数据
8. 跨批次丢失了中间部分的卖方向和买卖方向数据
9. A1709[120] 则后边还有数据
10. 前边有a1709[120] 但是丢包，后边收到a1709[30]，则无法判断丢包

///////////////////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////////////////////////

## 需求

上期全挡数据因是UDP传输，在实盘中，存在数据包丢包的情况，故需要对丢包情况做些处理，为了使策略获得更多的盘口信息，对于因丢包而损坏的数据，只要该数据有价值，则就需要发送给策略，故对损坏的数据执行如下处理：

* + 如果某个合约只有单方向数据，则丢掉该合约的该笔数据
  + 如果某个合约有2个方向的数据，但有数据丢失，但是只丢失了排在尾部的数据，则保留该数据，但是将总买/卖量数据都设置为0；但是如果丢失了头部的数据，则丢掉该合约的该笔数据

## 如何确保数据的完整性

前置条件：在一个批次完整数据队列中，买方向队列的所有合约在卖方向队列中也都存在（一字涨/跌停，可能因为没有买方数据，或者没有卖方向数据而被丢掉）

确保数据完整的要点：

* + - 开始接收时的起点必须是完整数据
    - 买方向缓存队列中的数据必须是完整的数据(也包含可用的损坏数据，但做了损坏标记)
    - 只有接收一个合约的完整数据才将该合约的买方向和卖方向的数据按顺序发出(也包含可用的损坏数据，但做了损坏标记)
    - 只有买方向和卖方向都有数据才认为是完整数据（这样会误丢涨停和跌停数据，但可以通过一档数据获取涨/跌停信息）

## 找到数据接收的起点

因为系统可能是中途启动，或有丢包发生在开市时，所以开始接收的数据可能是不完整的，为了找到系统接收完整数据的起点，我们需要执行如下步骤：

1. 接收到第一笔买方向的数据，将该数据记录为牺牲数据(victim)
2. 随后收到新的合约数据：
   * + - 如果没有丢包发生，则该合约数据作为正常数据的起点，并进入正常的数据接收流程
       - 如果有丢包发生，则清空victim，然后回到步骤i

注意：因全挡行情有多个UDP数据源，如上是针对一个数据源而言。

## 数据接收流程

当系统找到完整数据的起点后，则进入数据接收流程。因为上期全挡行情共有3个UDP数据源，故有3个数据缓冲区，分别缓存3个数据源的数据。

缓冲区的结构：

* 买方向队列

缓存买方向的数据

* 卖方向队列

存储卖方向的数据

注意：因全挡行情有多个UDP数据源，如上是针对一个数据源而言。

## 接收买方向数据

## 无丢包的接收处理

当接收的数据是买方向数据，且没有丢包发生，：

1. 如果victim等于空，则是正常接收处理

* 新数据合约大于等于买队列末尾数据合约，则且新数据合约将数据放到队列中；
* 新数据合约小于于买队列末尾数据合约(一字涨停或跨批次丢包)
  + 如果卖方向有数据，则发送合适的数据后清空买卖队列；
  + 如果卖方向没有数据，直接清空买卖队列
* 将新数据合约将数据放到队列中；

1. 如果victim不等于空，则是丢包后修正流程

* 如果新数据合约等于victim，则丢掉该笔数据；
* 如果新数据合约不等于victim
  + 如果新数据合约大于等于买队列队尾合约，则加到队列中
  + 如果新数据合约小于买队列队尾合约（跨批次丢包或一字涨停），则清空买队列后并将数据放到队列中；
  + 将victim清空

## 一个批次内丢包

如果接收到的数据是买方向数据，但是有丢包发生，则需要执行如下的处理过程：

1. 将该数据记为牺牲者数据(victim)
2. 如果买方向队列尾部那个合约的数据是不完整数据，则将其标记为：damaged

如何判断尾部合约的数据是否完整：如果最后的数据的count<120，则是完整数据；否则为不完整数据（因为数据的缓冲区长度：120）

1. 退出该流程

## 跨行情批次的丢包

当有丢包发生时，将新接收的数据的合约作为victim,然后：

* 新的数据是买方向数据，合约小于买方向队尾数据合约，则将买卖队列的数据都标记为：damaged，然后如果卖方向队列有数据，则将匹配的数据发送出去，之后，清除买/卖方向队列的所有数据

## 接收卖方向数据

## 无丢包处理流程

当收到卖方向数据后，且没有丢包发生，有丢包修正流程和正常接收流程

## 丢包后修正流程

如果victim不等于空，则进入修正流程

1. 如果新数据合约等于victim，则丢掉该数据，结束流程
   1. 如果新数据合约不等于victim，如果卖队列有数据，将可以发送的数据发送出去后，清空买卖队列
2. 将新数据放到队列中
3. 如果数据的count==120，则结束该流程，等待处理下一笔数据
4. 如果数据的count<120，则从头扫描买队列的数据
5. 如果合约大于买方向队首合约，则弹出该数据并丢掉该队首数据，并进入该父步骤
6. 如果合约等于队首合约，则弹出所有这个合约的数据，并发送，并退出处理流程，等待接收下一笔数据
7. 如果合约小于买方向队首合约，则弹出并丢弃卖方向队列中所有数据，并退出处理流程，等待接收下一笔数据
8. 将victim清空

## 正常接收流程

如果victim等于空，则进入正常接收流程：

1. 如果卖队列有数据，且新数据合约不等于队列尾部合约，则发送合适的数据后清空买卖队列
2. 将数据放到队列中
3. 如果数据的count==120，则结束该流程，等待处理下一笔数据
4. 如果数据的count<120，则从头扫描买队列的数据
5. 如果合约大于买方向队首合约，则弹出该数据并丢掉该队首数据，并进入该父步骤
6. 如果合约等于队首合约，则弹出所有这个合约的数据，并发送，并退出处理流程，等待接收下一笔数据

## 丢包处理流程

## 单个批次内丢包

当收到卖方向数据后，如果有丢包，则执行如下处理流程：

1. 将新数据的合约记为victim
2. 如果卖方向队列有数据，合约的数据是不完整数据，则将其标记为:damaged，并按 方法将买卖对列中的相应数据按顺序发送出去

如何判断尾部合约的数据是否完整：如果最后的数据的count<120，则是完整数据；否则为不完整数据（因为数据的缓冲区长度：120）；

1. 如果卖方向队列没有数据，则将买方向队列队尾不完整数据标记为：danaged

如何判断尾部合约的数据是否完整：如果最后的数据的count<120，则是完整数据；否则为不完整数据（因为数据的缓冲区长度：120）；

1. 结束该流程

## 跨批次丢包

丢包数据跨一个以上批次数据

当有丢包，且新的合约小于卖方向队尾数据合约，则清除买/卖方向队列的所有数据

* 新的数据是卖方向数据，合约小于卖方向队尾数据合约，则清除买/卖方向队列的所有数据

新的买方向数据，合约大于等于卖方向队尾数据合约，且卖方向队列有数据，则清除买/卖方向队列的所有数据

## 数据字典

Victim:当丢包时，要牺牲丢弃的数据的合约

为什么要丢掉牺牲着数据？因为当丢包发生时，接收的第一笔数据无法判断其是否完整（因为数据缓存长度120，有些数据是分几次发送的，而有些是单次的）

要思考的问题

Victim考虑临界点问题：如当接收买数据时，下一笔是卖；或者当接收卖时，下一笔是买

如果丢包连续发生

Victim不为空，则是丢包修正流程