算法 1: flvTag 切片发送

输入: 字节数组 flv_tag

```
while next flv_tag do
    if len(flv_tag) <= MAX_RTP_PAYLOAD_LEN then
        rp ← Rtp 包(initialization)
        rp.marker ← true
        rp.payload ← flv_tag
        发送 rp
    else
        slice_num ← len(flv_tag)%MAX_RTP_PAYLOAD_LEN + 1
        for i=0 to slice_num do
            rp ← Rtp 包(initialization)
            if !last slice then
                rp.marker ← false
            else
                rp.marker ← true
            end if
            rp.payload ← flv_tag[i*MAX_RTP_PAYLOAD_LEN,
            (i+1)*MAX_RTP_PAYLOAD_LEN]
            发送 rp
        end for
    end if
end while
```

算法 2: Rtp 排序与重传

输入: 新的 Rtp 数据包 rp

输出: 有序的 Rtp 数据包 pkt

```
while next rp do
    firstSeq ← rtpQueue[0].Seq;
    rtpQueue[rp.Seq - firstSeq] ← rp;
    for len(rtpQueue) > 等待区大小 do
        p ← rtpQueue[0];
        if p 为空 then
            重传 p;
        end if
        输出 p
    for rtpQueue[0]不为空 do
        p ← rtpQueue[0];
        输出 p
    end for
end while
```

算法 3: Rtp 重组 flvTag

输入: 有序的 Rtp 数据包 rp
输出: flv Tag 字节数组 flvTag

```
while next rp do
    marker ← rp.marker;
    pos ← 0
    if marker 为 0 then
        if rp 为初始分片 then
            payload ← rp.payload;
            TagSize ← 解析 playload;
            flvTag ← 长度 TagSize 的字节数组(initialize);
            flvTag[0:len(payload)] ← payload;
            pos ← len(payload)
        else
            flvTag[pos:len(payload)] ← payload;
            pos ← pos + len(payload)
        end if
    else
        flvTag[pos:len(payload)] ← payload;
        pos ← 0
        return flvTag
    end if
end while
```