文章编号:1671-4598(2019)06-0147-06 DOI:10.16526/j.cnki.11—4762/tp.2019.06.032 中图分类号:V557.1 文献标识码:A

# 基于动态窗口的虚拟信道通用调度算法

饶爱水，李永刚，周锦标，汪 毅，李清梅
（中国卫星海上测控部，江苏 江阴 214431）

**摘要**：针对虚拟信道调度算法的通用设计问题，设计了独占式轮转和顺序式轮转两种全同步调度算法、抢占式优先和非抢占式优先两种全异步调度算法、以及独占式混合和顺序式混合两种同步/异步混合调度算法，进一步实现了基于动态窗口的虚拟信道通用调度算法，通用算法采用双层调度模型，通过参数配置可实现8种调度策略；实践表明，通用算法既能满足同步数据固定时隙要求，又能适应异步数据动态调整要求，还能满足应急数据及时发送要求，窗口边界和信道边界可动态调整，减少了信道资源浪费，具有广泛通用性和良好适应性。

**关键词**：同步；异步；独占式；顺序式；抢占式；调度策略

# Virtual Channels General Scheduling Algorithm Based on Dynamic Windows

Rao Aishui, Li Yonggang, Zhou Jinbiao, Wang Yi, Li Qingmei
(Satellite Maritime Tracking and Control Department of China, Jiangyin 214431, China)

**Abstract**: Aiming at the problem of general algorithm with virtual channels scheduling, two synchronous scheduling algorithms with exclusive cycle and sequential cycle are designed, two asynchronous scheduling algorithms with preemptive priority and non-preemptive priority are designed, two synchronous and asynchronous mixed scheduling algorithms with exclusive mixed and sequential mixed are designed. Further more, a virtual channels general scheduling algorithm based on dynamic windows is proposed, it is composed of two layers of scheduling models, eight kinds of scheduling strategies are developed by configured parameters. Experiments show that the proposed algorithms satisfy the fixed time slots with synchronous data , the dynamic time slots with asynchronous data, and the peremptorily sending with urgent data. The dynamic boundaries of windows and channels are produced , the waste channels resource is decreased. The proposed algorithms are extensive generality and well adaptability.

**Keywords**: synchronous; asynchronous; exclusive; sequential; preemptive; scheduling strategy

## 0 引言

虚拟信道调度算法在国内外被广泛研究。文献[1]将数据帧生成模块与虚拟信道复用模块作为一个整体考虑，提出了自适应帧生成算法与虚拟信道调度算法相结合的多路复用方法，在任一时刻均计算信道传输紧迫度，其异步信道采用抢占式调度策略；文献[2]提出了一种基于虚拟信道/帧分离估算紧迫度的动态调度算法，将信源数据分成同步数据和异步数据，在异步数据的调度中，构造虚拟信道传输紧迫度函数分配传输时隙，其同步信道和异步信道的时隙不可互转；文献[3]设计了一种边界可移动的VIP/同步/异步混合复用方式，其同步信道采用独占式调度策略，异步信道采用抢占式调度策略，在同步信道时隙数据无效时，将调度异步信道。

目前针对虚拟信道调度一个或几个特性而设计的调度算法[4-7]，局限于某些应用场合，或者着眼于提高整体调度性能[8-10]，无法得到普遍应用；信道时隙计算、信道优先级调整内嵌到算法中，限制了调度算法在更大范围内通用。

**收稿日期**:2018-10-11； **修回日期**:2018-11-12。

**作者简介**:饶爱水(1979-)，男，江西临川人，大学本科，高级工程师，主要从事航天测控方向的研究。

本文着眼于设计广泛通用的虚拟信道调度算法，把信道时隙计算和信道优先级调整从算法中剥离，信道时隙和信道优先级以参数形式传入算法中；对全同步调度算法、全异步调度算法和同步/异步混合调度算法进行了研究，设计了独占式轮转和顺序式轮转两种全同步调度算法，抢占式优先和非抢占式优先两种全异步调度算法，以及独占式混合和顺序式混合两种同步/异步混合调度算法，在此基础上考虑应急信道数据发送要求，实现了基于动态窗口的虚拟信道通用调度算法。

## 1 全同步调度算法

设虚拟信道 $VS_1 \sim VS_n$ 为同步信道，占用时隙分别为 $Ts_1 \sim Ts_n$，遥测帧周期为 $T$，显然 $Tsi$ 应当为 $T$ 的整数倍，同步信道总时隙 $Ts$ 为 $Ts_1 \sim Ts_n$ 之和。采用时间轮转调度算法实现虚拟信道同步调度，时间片长度为 $T$。按照调度策略，分为独占式轮转调度算法和顺序式轮转调度算法，算法描述如下：

1）根据虚拟信道占用时隙和遥测帧周期，计算信道轮转次数 $Ni$ $(1 \leqslant i \leqslant n)$：$Ni = Tsi/T$。

2）系统维护一张虚拟信道调度表，如表1所示，“序号”为调度顺序，“发送计数”$Ci$ 为本轮周期内剩余调度次

数，初始化为 $Ni$，$Ri$ 为信道数据是否有效。

表 1 时间轮转调度算法中的虚拟信道调度表

| 序号 | 虚拟信道 | 发送计数 $Ci$ | 数据有效标志 $Ri$ |
|---|---|---|---|
| $V_1$ | $VS_1$ | $N_1$ | 是 |
| 2 | $VS_2$ | $N_2$ | 是 |
| …… | …… | …… | …… |
| $n$ | $VS_n$ | $N_n$ | 是 |

3）独占式轮转调度：当前信道调度完毕后，再调度下一个信道。即当遥测帧中断到来时，系统按照如下步骤搜索虚拟信道调度表：

①系统从前往后搜索。如果 $Ri$ 为真且 $Ci$ 大于 0，则调度信道 $i$，对应的“发送计数” $Ci$ 减 1；接下来 $Ni-1$ 周期内，均调度信道 $i$，直至 $Ci$ 计数值为 0 时调度信道 $i+1$；如果 $Ci$ 为 0，该信道不参与调度；

②当最后一个信道 $n$ 调度完毕后，所有信道 $Ci$ 恢复初始值 $Ni$，此时可以重新获取信道时隙，然后重复步骤①，直至系统退出。

4）顺序式轮转调度算法与独占式轮转调度算法略有不同，当前信道不是一次性调度完毕，而是仅调度 1 次，下次调度下一个信道。即当遥测帧中断到来时，系统按照如下步骤搜索虚拟信道调度表：

①系统依次从前往后搜索。如果 $Ci$ 大于 0，则调度信道 $i$，$Ci$ 减 1，下一次调度信道 $i+1$；如果 $Ci$ 为 0，则该信道不参与调度，直接调度信道 $i+1$；

②当最后一个信道 $n$ 被调度后，返回调度表起始处，重复步骤①，直至所有信道均被调度完毕，即 $Ci$ 均为 0，则所有虚拟信道的 $Ci$ 恢复为初始值 $Ni$，此时可以重新获取信道时隙，重复步骤①，直至系统退出。

设计了获取同步信道调度函数 int GetSyncVC（int * Ci，int * Ri，int n，int &nNextVC，BOOL bSeq）实现该算法，函数流程如图 1 所示。其中 $n$ 为同步信道个数，nNextVC 为下一次可能被调度的虚拟信道（初始化为 0），bSeq 为 TRUE 表示顺序式轮转调度，bSeq 为 FALSE 表示独占式轮转调度，返回值表示当前被调度的虚拟信道（−1 表示本轮信道调度完毕）。

①在调度某信道时，如果发现该信道数据无效，则自动调度下一信道，以避免发送填充数据；

②在顺序式调度模式下，当搜索到最后一个信道且未发现有需要调度数据时，需返回调度表头部重新进行搜索，以调度其他信道尚未调度完毕的数据；

③在顺序式调度模式下，下一次调度下一个信道；而在独占式调度模式下，当本信道调度完毕后，下一次才调度下一个信道。

同步信道占用时隙随卫星运行状态变化，全同步调度算法提供了时隙调整接口，GetSyncVC（）返回−1 时重新计算初始值 $Ni$，$Ci$ 恢复为初始值 $Ni$。

假设 4 个同步信道 $VS_1$、$VS_2$、$VS_3$、$VS_4$，占用时隙为

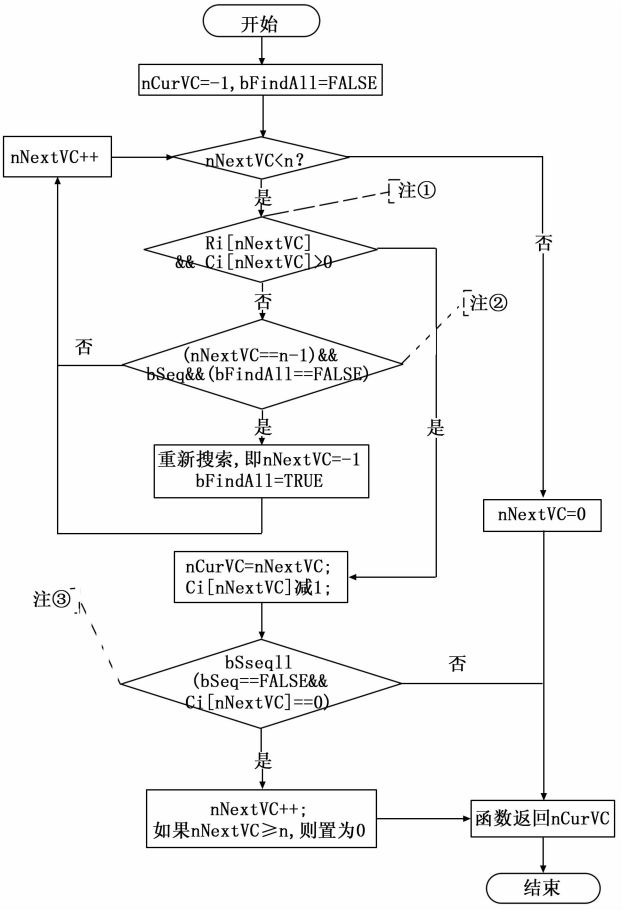



图 1 同步信道调度函数流程图

8 s、4 s、2 s、1 s，帧周期为 1 s，轮转次数为 8、4、2、1。总时隙为 15 s，一个调度周期需调度 15 次。信道数据均有效，顺序式同步调度和独占式同步调度的虚拟信道如表 2 第 2 列和第 5 列所示；第 3 列和第 6 列表示 $VS_2$ 数据无效时调度顺序，加粗部分为第二轮调度；第 4 列和第 7 列表示 $VS_2$ 数据在第 10 次有效时的调度顺序。当某信道数据无效时，该算法可调度后续数据有效的信道；当信道数据恢复时，可确保该信道数据得到调度。而顺序式同步调度相比独占式同步调度，能够提高信道调度及时性。

## 2 全异步调度算法

设虚拟信道 $VA_1 \sim VA_n$ 为异步信道，占用时隙分别为 $Ta_1 \sim Ta_n$，优先级依次为 $P_1 \sim P_n$，按照优先级排序，即 $P_1 \geqslant P_2 \geqslant \cdots\cdots \geqslant P_n$。遥测帧周期为 $T$，显然 $Tai$ 应当为 $T$ 的整数倍，异步信道总时隙 $Ta$ 为 $Ta_1 \sim Ta_n$ 之和，该时间称为异步信道调度周期。

按照调度策略不同，分为非抢占式优先级调度算法和抢占式优先级调度算法，算法描述如下：

1）根据虚拟信道占用时隙和遥测帧周期，计算信道发送次数 $Si$（$1 \leqslant i \leqslant n$）：$Si = Tai/T$。

2）系统维护一张信道调度表，按照优先级从高到低排列，如表 2 所示，“发送计数” $Ci$ 为本轮周期内剩余调度次

表 2　顺序式同步调度和独占式同步调度顺序

| 顺序 | 顺序式同步调度 | | | 独占式同步调度 | | |
|---|---|---|---|---|---|---|
| | 有效 | Vs2 | | 有效 | $Vs_2$ 无效 | |
| | | 无效 | 10 有效 | | 无效 | 10 有效 |
| 1 | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ |
| 2 | $VS_2$ | $VS_3$ | $VS_3$ | $VS_1$ | $VS_1$ | $VS_1$ |
| 3 | $VS_3$ | $VS_4$ | $VS_4$ | $VS_1$ | $VS_1$ | $VS_1$ |
| 4 | $VS_4$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ |
| 5 | $VS_1$ | $VS_3$ | $VS_3$ | $VS_1$ | $VS_1$ | $VS_1$ |
| 6 | $VS_2$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ |
| 7 | $VS_3$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ |
| 8 | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ | $VS_1$ |
| 9 | $VS_2$ | $VS_1$ | $VS_1$ | $VS_2$ | $VS_3$ | $VS_3$ |
| 10 | $VS_1$ | $VS_1$ | $VS_2$ | $VS_2$ | $VS_3$ | $VS_3$ |
| 11 | $VS_2$ | $VS_1$ | $VS_1$ | $VS_2$ | $VS_4$ | $VS_4$ |
| 12 | $VS_1$ | $VS_1$ | $VS_2$ | $VS_2$ | $VS_1$ | $VS_2$ |
| 13 | $VS_1$ | $VS_3$ | $VS_1$ | $VS_3$ | $VS_1$ | $VS_2$ |
| 14 | $VS_1$ | $VS_4$ | $VS_2$ | $VS_3$ | $VS_1$ | $VS_2$ |
| 15 | $VS_1$ | $VS_1$ | $VS_2$ | $VS_4$ | $VS_1$ | $VS_2$ |

数，初始化为 $Si$，$Ri$ 为信道数据是否有效，初始化时信道调度表如表 3 所示。

表 3　优先级调度算法中的虚拟信道调度表

| 序号 | 虚拟信道 | 发送计数 $Ci$ | 信道优先级 $Pi$ | 数据有效标志 $Ri$ |
|---|---|---|---|---|
| 1 | $VA_1$ | $S_1$ | $P_1$ | 是 |
| 2 | $VA_2$ | $S_2$ | $P_2$ | 是 |
| …… | …… | …… | …… | …… |
| $n$ | $VA_n$ | $Sn$ | $Pn$ | 是 |

3）非抢占式优先级调度算法中，系统按照如下步骤搜索虚拟信道调度表：

①当遥测帧周期到来时，如果 $VA_1$ 信道 $Ri$ 为真，在接下来 $S_1$ 个周期内，均发送 $VA_1$ 数据，发送完毕后，$Ri$ 设置为假，令 $i=2$；如果 $VA_1$ 信道 $Ri$ 为假，令 $i=2$；

②当遥测帧周期到来时，如果 $VAi$ 信道 $Ri$ 为真，在接下来的 $Si$ 个周期内，均发送 $VAi$ 的数据，发送完毕后，$Ri$ 设置为假。如果 $VCi$ 信道 $Ri$ 为假，令 $i++$，如果 $i\leqslant n$，重复步骤②，否则转③；

③此时最后一个信道 $n$ 调度完毕，所有信道 $Ci$ 恢复初始值 $Si$，在此可以调整信道优先级，然后重复步骤①，直至系统退出。

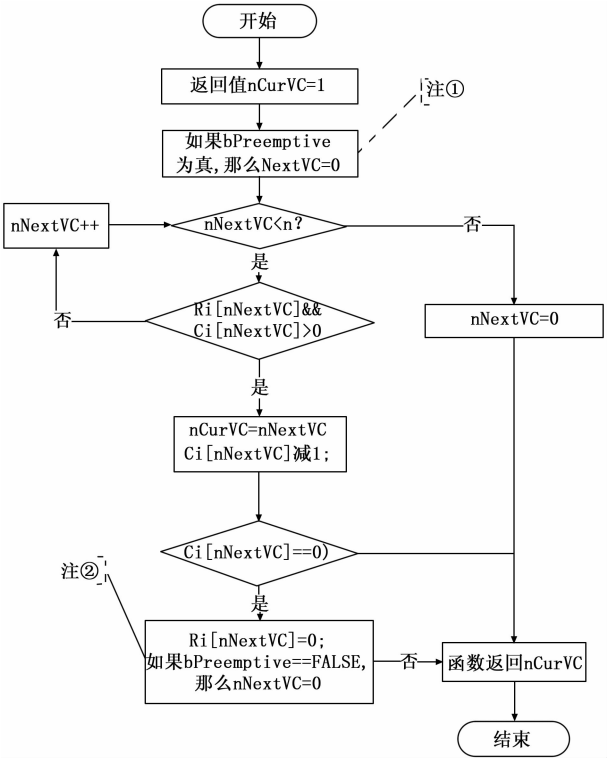
4）抢占式优先级调度算法与非抢占式优先级调度算法略有不同，即低优先级信道占用时间片时，高优先级信道可抢占其时间片，概述如下：

①与非抢占式优先级调度算法步骤①相同；

②当遥测帧周期到来时，如果 $VAi$ 信道 $Ri$ 为真，发送一帧 $VAi$ 信道数据，并令 $Ci$ 减 1（如果 $Ci$ 等于 0，设置 $VAi$ 的 $Ri$ 为否），并转步骤①；如果 $VAi$ 信道 $Ri$ 为假，令 $i++$，如果 $i\leqslant n$，重复步骤②，否则转③；

③与非抢占式优先级调度算法步骤③相同。

设计了获取异步信道调度函数 int GetAsynVC（int * Ci，int * Ri，int n，int &nNextVC，BOOL bPreemp）实现该算法，流程图如图 2 所示。bPreemp 为 TRUE 表示抢占式优先级调度，bPreemp 为 FALSE 表示非抢占式优先级调度，返回值表示当前被调度的虚拟信道（－1 表示本轮信道调度完毕）。




①在抢占式优先级调度模式下，高优先级信道优先调度，即使当前信道数据未调度完毕，也从第一个信道开始调度；②当信道调度完毕后，数据准备就绪标志置为假。此时如果是非抢占式优先级调度，还需返回调度表头重新进行调度。

图 2　异步信道调度函数流程图

异步信道占用时隙及其优先级可随卫星运行状态变化，全异步调度算法提供了信道时隙调整接口以及优先级调整接口，在一轮调度完成，即函数 GetAsynVC（）返回－1 时调整，此时按照优先级重新排序生成 $VA_1\sim VA_n$，重新计算初始值 $Si$，$Ci$ 计数恢复为初始值 $Si$，此后系统将按新的时隙和优先级进行调度。此外，系统在每次调用函数 GetAsynVC（）前，均可调整优先级。

假设有 4 个异步信道 VA1、VA2、VA3、VA4，占用时隙分别为 8 s、4 s、2 s、1 s，帧周期为 1 s，发送次数分别为 8、4、2、1，优先级设置为 4、3、2、1，各信道数据均准备完毕。抢占式优先级调度和非抢占式优先级调度过程中，搜索的信道如表 4 所示。

表 4 抢占式和非抢占式优先级调度中搜索的信道

| 顺序 | 调度信道 | 抢占式搜索的信道 | 非抢占式搜索的信道 |
|---|---|---|---|
| 1 | VA1 | VA1 | VA1 |
| 2 | VA1 | VA1 | VA1 |
| 3 | VA1 | VA1 | VA1 |
| 4 | VA1 | VA1 | VA1 |
| 5 | VA1 | VA1 | VA1 |
| 6 | VA1 | VA1 | VA1 |
| 7 | VA1 | VA1 | VA1 |
| 8 | VA1 | VA1 | VA1 |
| 9 | VA2 | VA1、VA2 | VA1、VA2 |
| 10 | VA2 | VA1、VA2 | VA2 |
| 11 | VA2 | VA1、VA2 | VA2 |
| 12 | VA2 | VA1、VA2 | VA2 |
| 13 | VA3 | VA1、VA2、VA3 | VA1、VA2、VA3 |
| 14 | VA3 | VA1、VA2、VA3 | VA3 |
| 15 | VA4 | VA1、VA2、VA3、VA4 | VA1、VA2、VA3、VA4 |

表 4 显示抢占式和非抢占式最终调度信道相同，但实际搜索信道不同。表 5 第 2 列和第 3 列显示了 VA1 信道在第 5 次调度时数据无效、在第 7 次调度时恢复、在第 9 次调度时数据无效、在第 11 次调度时恢复的调度顺序，调度开始时所有数据均有效；表 5 第 3 列和第 4 列显示了所有数据均有效时，在第 5 次调度时，VA3 优先级提高为 5 时的调度顺序，抢占式调度第 5 次开始调度 VA3，而非抢占式调度第 9 次才开始调度 VA3。

表 5 数据无效对信道调度的影响

| 顺序 | VA1 在第 5、9 次无效，第 7、11 次恢复 | | VA3 在第 5 次时提高优先级为 5 | |
|---|---|---|---|---|
| | 抢占式 | 非抢占 | 抢占式 | 非抢占 |
| 1 | VA1 | VA1 | VA1 | VA1 |
| 2 | VA1 | VA1 | VA1 | VA1 |
| 3 | VA1 | VA1 | VA1 | VA1 |
| 4 | VA1 | VA1 | VA1 | VA1 |
| 5 | VA2 | VA2 | VA3 | VA1 |
| 6 | VA2 | VA2 | VA3 | VA1 |
| 7 | VA1 | VA2 | VA1 | VA1 |
| 8 | VA1 | VA2 | VA1 | VA1 |
| 9 | VA2 | VA3 | VA1 | VA3 |
| 10 | VA2 | VA3 | VA1 | VA3 |
| 11 | VA1 | VA1 | VA2 | VA2 |
| 12 | VA1 | VA1 | VA2 | VA2 |
| 13 | VA3 | VA1 | VA2 | VA2 |
| 14 | VA3 | VA1 | VA2 | VA2 |
| 15 | VA4 | VA4 | VA4 | VA4 |

## 3 同步/异步混合调度算法

设虚拟信道 $VS_1 \sim VS_m$ 为同步信道（统称为 VS，即同步窗口），$VA_1 \sim VA_n$ 为异步信道（统称为 VA，即异步窗口），同步信道占用时隙为 $Ts$，异步信道占用时隙为 $Ta$。设计了固定比率同步/异步混合调度算法，按照调度策略分为独占式混合算法和顺序式混合算法。算法描述如下：

1）根据窗口占用时隙和遥测帧周期，计算窗口打开次数 $Wi$（$1 \leqslant i \leqslant 2$）：$W1 = Ts/T$，$W2 = Ta/T$。

2）系统维护一张窗口调度表，如表 6 所示，“序号”表示调度顺序，“发送计数” $Ci$ 表示本轮周期内该窗口还剩余的调度次数，且 $Ci$ 被初始化为 $Wi$，“数据有效标志” $Vi$ 表示该窗口数据是否有效，$V_1$ 初始化为同步信道 $Ri$ 值之和，$V_2$ 初始化为异步信道 $Ri$ 值之和，初始化时窗口调度表如表 6 所示。

表 6 同步/异步混合窗口调度表

| 序号 | 窗口代号 | 窗口名称 | 发送计数 $Ci$ | 数据有效标志 $Vi$ |
|---|---|---|---|---|
| 1 | VS | 同步窗口 | W1 | 是 |
| 2 | VA | 异步窗口 | W2 | 是 |

3）独占式混合算法中，首先调度 $VS$ 窗口，即采用全同步调度算法调度 $VS_1 \sim VS_m$，调度 $W_1$ 个周期后，在接下来 $W_2$ 个周期内，调度 $VA$ 窗口，即采用全异步调度算法调度 $VA_1 \sim VA_m$。共调度（$W_1 + W_2$）个周期后，一次混合调度完毕后，重复步骤 3），直至系统退出。

4）顺序式混合算法中，依次调度 $VS$ 窗口、$VA$ 窗口、$VS$ 窗口、$VA$ 窗口。即采用全同步调度算法调度 $VS_1 \sim VS_m$，调度 1 个周期以后，在接下来的 1 个周期内，采用全异步调度算法调度 $VA_1 \sim VA_m$，然后调度 $VS_1 \sim VS_m$、$VA_1 \sim VA_m$。共调度（$W_1 + W_2$）个周期以后，一次混合调度完毕，重复步骤 4），直至系统退出。

混合信道调度算法流程图如图 3 所示。其中窗口调度采用 GetSyncVC 函数实现，即设 $Ci[] = \{W_1, W_2\}$，$n = 2$，nNextMixVc=0，bMix 表示顺序式混合（TRUE）还是独占式混合（FALSE）：

int nW= GetSyncVC（Ci，2，nNextMixVc，bMix）；

$nW$ 为 0，表示调度同步信道；$nW$ 为 1，表示调度异步信道；$nW$ 为 −1，表示本轮调度结束，开启新一轮调度，此时重新计算初始值 $S$ 和 $A$，$Ci$ 计数恢复为初始值，此后系统按新的时隙进行调度。

注：①拟调度窗口发送计数有效而数据无效时，GetSyncVC 函数将跳过该窗口调度下一个窗口，之前需计算窗口数据有效标志；

②在一轮混合调度完毕后，调整窗口打开次数；

③在一轮异步信道调度完毕后，调整信道发送次数及优先级；

④在一轮同步信道调度完毕后，调整信道轮转次数。

同步/异步混合调度算法采用两级调度机制，第一级在同步窗口和异步窗口之间调度，第二级采用全同步或全异步调度。显然，混合调度算法根据调度方式可分为 8 种调度模式，如表 7 所示。

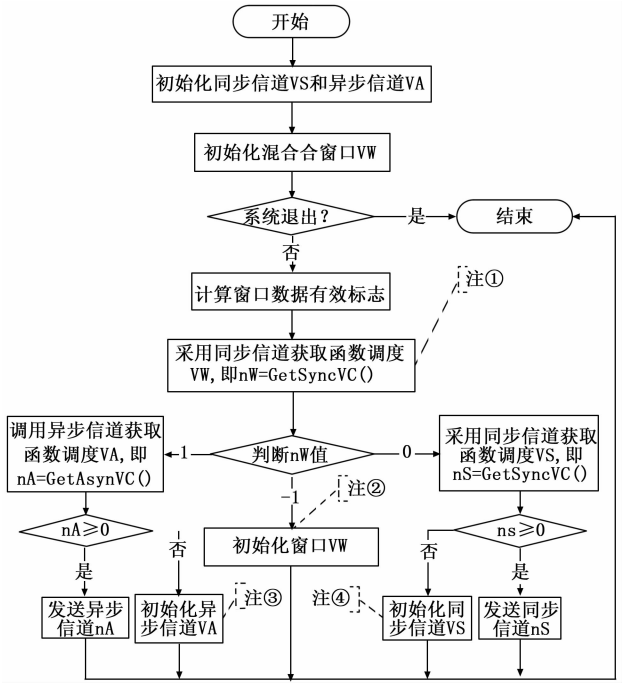



图 3　混合信道调度算法流程图

表 7　混合调度算法的多种调度模式

| 模式编号 | 窗口调度 | 同步信道调度 | 异步信道调度 |
|---|---|---|---|
| 1 | 顺序式 | 顺序式 | 抢占式 |
| 2 | 顺序式 | 顺序式 | 非抢占式 |
| 3 | 顺序式 | 独占式 | 抢占式 |
| 4 | 顺序式 | 独占式 | 非抢占式 |
| 5 | 独占式 | 顺序式 | 抢占式 |
| 6 | 独占式 | 顺序式 | 非抢占式 |
| 7 | 独占式 | 独占式 | 抢占式 |
| 8 | 独占式 | 独占式 | 非抢占式 |

## 4　基于动态窗口的通用调度算法

该算法建立在全同步调度算法、全异步调度算法以及同步/异步混合调度算法之上，既能满足同步数据固定时隙要求，又能适应异步数据动态调整要求，还能满足应急数据及时发送要求，并减少信道资源浪费。算法描述如下：

1）设计了如表 8 所示的混合调度表，相比同步/异步混合调度表，增加了应急窗口 $VE$，初始化时，同步窗口、异步窗口比率为 $W_1$：$W_2$，应急窗口发送次数默认为 0，当需要发送应急信道数据时才计算其值。

表 8　基于动态窗口的混合调度表

| 序号 | 窗口代号 | 窗口名称 | 发送计数 $Ci$ | 数据有效标志 $Vi$ |
|---|---|---|---|---|
| 1 | $VE$ | 应急窗口 | $W_0$ | 否 |
| 2 | $VS$ | 同步窗口 | $W_1$ | 是 |
| 3 | $VA$ | 异步窗口 | $W_2$ | 是 |

2）设计了动态窗口调度函数 int GetDynamicW（int $Ci$[3]，int &nNextVC，BOOL bSeq）实现窗口调度，算法流程如图 4 所示。其中：$Ci$ [] 为窗口剩余调度次数（初始化为 $Wi$）；nNextVC 为下次可能被调度窗口（初始化为 0）；bSeq 为 TRUE 表示顺序式轮转调度，bSeq 为 FALSE 表示独占式轮转调度；返回值－1 表示本次调度轮空，0 表示应急窗口有效，1 表示同步窗口有效，2 表示异步窗口有效。

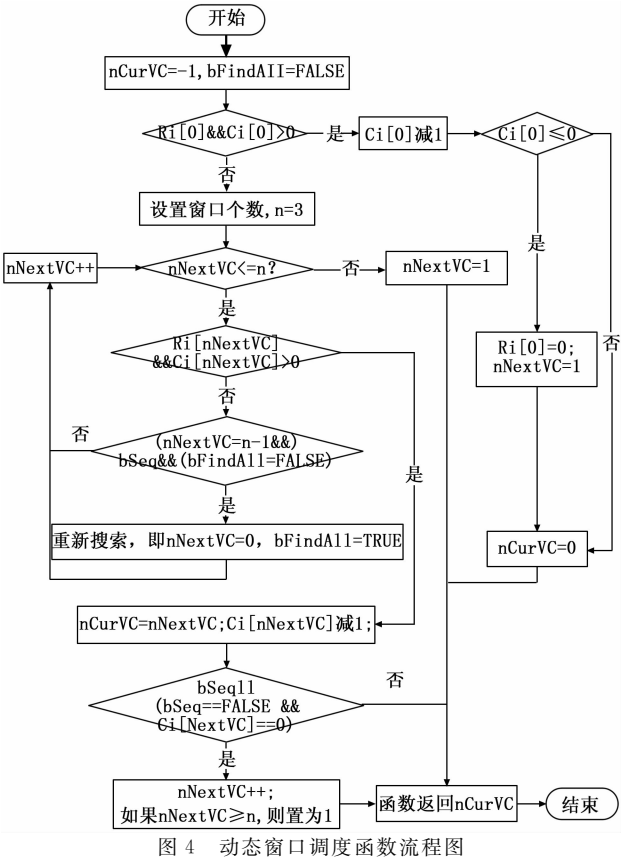



图 4　动态窗口调度函数流程图

3）在动态窗口调度函数 GetDynamicW（）、同步信道调度函数 GetSyncVC（）、异步信道调度函数 GetAsynVC（）基础上，实现基于动态窗口的混合调度算法，流程图与图 3 类似，不同之处在于窗口调度采用 GetDynamicW（）函数，其返回值为 0 时调度应急窗口，1 时调度同步窗口、2 时调度异步窗口。

基于动态窗口的通用调度算法中，当 $W_0=0$ 时，即实现了第 3 节的同步/异步混合调度算法；当 $W_0=W_1=0$ 时，即实现了第 2 节的全异步调度算法；当 $W_0=W_2=0$ 时，即实现了第 1 节的全同步调度算法。

## 5　实验结果与分析

采用 Visual C＋＋6.0 实现了基于动态窗口的通用调度算法，程序界面图 5 所示。用户通过设置窗口调度方式、同步信道调度方式、异步调度方式定制信道调度方式，通过设置窗口调度比率控制同步调度、异步调度以及应急调度的所占用时隙。

第一个列表框显示程序中使用的同步信道和异步信道中各虚拟信道的初始次数和优先级，其中同步信道共 4 个，即 $VS_1$～$VS_4$，初始次数分别为 8、4、2、1，异步信道共 4

个，即 $VA_1 \sim VA_4$，初始次数分别为 8、4、2、1，优先级分别为 4、3、2、1。

第二个列表框显示全同步调度顺序式和独占式的调度结果，全同步调度通过控制窗口调度比率实现，即同步窗口调度次数为 15，异步窗口调度次数为 0，应急窗口调度次数为 0；第三个列表框显示全异步调度抢占式和非抢占式的调度结果，窗口调度比率设置为同步窗口调度次数为 0，异步窗口调度次数为 15，应急窗口调度次数为 0，列表框最后一列中，显示了异步信道调度时信道搜索情况；第四个列表框显示混合式调度顺序式的调度结果，窗口调度比率设置为同步窗口调度次数为 15，异步窗口调度次数为 15，应急窗口调度次数为 0。

第五个列表框显示动态调度结果，初始时窗口调度比率设置为同步窗口调度次数为 15，异步窗口调度次数为 15，应急窗口调度次数为 0；为模拟应急窗口调度，首先设置应急窗口调度次数为正值，然后点击“应急调度”按钮，模拟程序运行过程中实时接收应急数据的情况。列表框中显示，同步调度和异步调度交叉进行，因为窗口调度方式选择了“顺序式”；同步信道中 $VS_1 \sim VS_4$ 为独占运行，异步信道中 $VA_1 \sim VA_4$ 为非抢占式运行；当用户点击应急调度后，列表框中显示应急调度被及时插入到调度队列中，调度 2 次后又恢复正常调度。

上述信道占用时隙、信道优先级、信道调度模式等信息可通过参数传入调度程序，这些参数既可由嵌入卫星的调度程序实时计算生成，也可由地面运控软件发送遥控数据注入的方式生成；前一种方式与时隙计算、优先级计算内嵌到算法中的执行效率一致，后一种方式虽然降低了系统执行效率，但增加了地面人工控制接口，有利于应急情况处置。实际应用中，两种方式可结合使用。

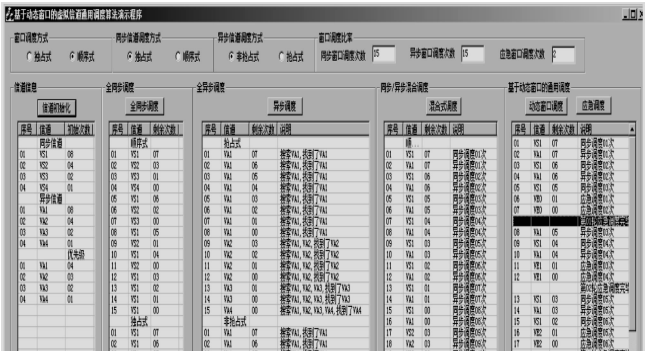


图 5 基于动态窗口的通用调度算法的实现

## 6 结论

在全同步调度算法、全异步调度算法、同步/异步混合调度算法基础上，实现的基于动态窗口的虚拟信道通用调度算法具有如下几个特点：

1）算法通用性强。通过参数配置，可实现全同步调度算法、全异步调度算法、同步/异步混合调度算法以及兼顾应急信道的混合调度算法，并具有 8 种可选的调度策略；

2）算法窗口利用率和信道利用率高。在同步调度和异步调度中，当拟调度信道发送计数有效而数据无效时，算法将跳过该信道调度下一个信道，实现了信道时隙自动接续；在同步/异步混合调度以及兼顾应急信道的混合调度中，当拟调度窗口发送计数有效而数据无效时，算法将跳过该窗口调度下一个窗口，实现了窗口时隙自动接续；

3）算法动态特性强。初始化时、每一轮调度完毕后，算法获取窗口占用时隙、信道占用时隙以及信道优先级，每次异步信道调度时也获取信道优先级，调度策略的参数化也提高了算法的动态调整性能。

本文提出的算法，抛开了传统算法把时隙计算、优先级计算和调度策略内嵌到算法中的设计思路，信道占用时隙、信道优先级、信道调度模式、调度策略等均以参数形式传入，提高了算法通用性和适用性。

**参考文献：**


[1] 田 野，那 鑫，夏 莹，等. AOS 自适应帧生成算法的性能分析与应用研究 [J]. 宇航学报，2012，33 (2)：242－248.

[2] 刘庆利，潘成胜，王国仁，等. AOS 虚拟信道/帧分离估算的调度算法 [J]. 系统仿真学报，2013，25 (1)：87－93.

[3] 别玉霞，潘成胜，蔡春妍. AOS 虚拟信道复用技术研究与仿真 [J]. 宇航学报，2011，32 (1)：193－198.

[4] 王向晖，王同桓，李宁宁，等. 一种 AOS 遥测源包多路调度算法 [J]. 航天器工程，2011，20 (5)：83－87.

[5] 毕明雪，潘成胜，赵运张，等. AOS 自适应帧长传输系统的仿真研究 [J]. 系统仿真学报，2011，23 (2)：358－362.

[6] 田 野，潘成胜，张子敬，等. AOS 协议中自适应帧生成算法的研究 [J]. 宇航学报，2011，32 (5)：1171－1178.

[7] Alrashed S，Alhiyafi J，Shafi A，et al. An efficient schedu-leability condition for non－preemptive real－time systems at common scheduling points [J]. The Journal of Supercomput-ing，2016，72 (12)：4651－4661.

[8] Tian Y，Wang R N，Jiang Y Q，et al. A novel multiple－chan-nels scheduling algorithm based on timeslot optimization in the advanced orbiting systems [J]. Multimedia Tools and Applica-tions，2017，76 (3)：4523－4551.

[9] Gavryushkin A，Khoussainov B，Kokho M，et al. dynamic al-gorithms for multimachine interval scheduling through analysis of idle intervals [J}. Algorithmica，2016，76 (4)：1160－1180.

[10] Zhao Y C，Zeng H B. An efficient schedulability analysis for optimizing systems with adaptive mixed－criticality scheduling [J]. Real－Time Systems，2017，53 (4)：467－525.