开源TTE规划软件(XZ-Plan)技术白皮书

V 1.0

湖南华芯通网络科技有限公司

2021年10月

目录

1 前言

时间触发以太网（Time Triggered Ethernet, TTE）在标准以太网基础上赋予确定性、可靠性传输能力的关键技术，在航空航天、车载网络等领域有非常广泛的应用前景。TTE 系统的设计包括数据平面分组交换和控制平面规划控制两部分。数据平面负责提供可配置的确定性分组转发能力。控制平面负责根据应用场景需求计算可行的时隙路径分配策略并配置到数据平面。因此，控制平面作为整个系统中“大脑”，是 TTE 系统满足应用需求的关键，而控制平面规划的需要解决的核心问题是如何高效合理的为关键流量分配时隙资源。

为支持多样化的规划算法设计、实现与验证，我们提出了开源 TTE 流量规划框架（Traffic Planning Framework: TPF）,旨在提供一套通用的 TTE 流量规划算法设计平台，定义标准的 TTE 交换平面抽象、规划算法外部接口规范，以及设备实现特定的工具集合。遵循 TPF 实现的规划算法可以方便地在开源的芯准 TTE 网络或其它 TTE 网络中使用和验证。

本白皮书分为四部分，第一部分为前言，介绍了 TPF 的开发背景；第二部分介绍流量规划框架 TPF 的具体内容；第三部分介绍采用 TPF 框架实现的 TTE 规划软件 XZ-Plan，其中实现了一种基于宏时隙的规划算法；第四部分介绍芯准 TTE 演示验证环境，第三方规划结果可以在该环境中测试与验证。

2 TTE 流量规划框架 TPF

TPF 旨在为 TTE 流量规划算法的开发提供通用环境，支持规划算法的测试与验证。TPF 本身与设备实现无关，遵循 TPF 框架开发的规划算法可以在不同的 TTE 网络中应用。

2.1 TPF 整体架构

TPF 整体架构如图 1 所示，主要分为三大部分：TTE 交换平面抽象、规划算法输入输出规范和 TPF 工具集合。其中，前两部分与设备实现无关，TPF 工具与设备实现相关。

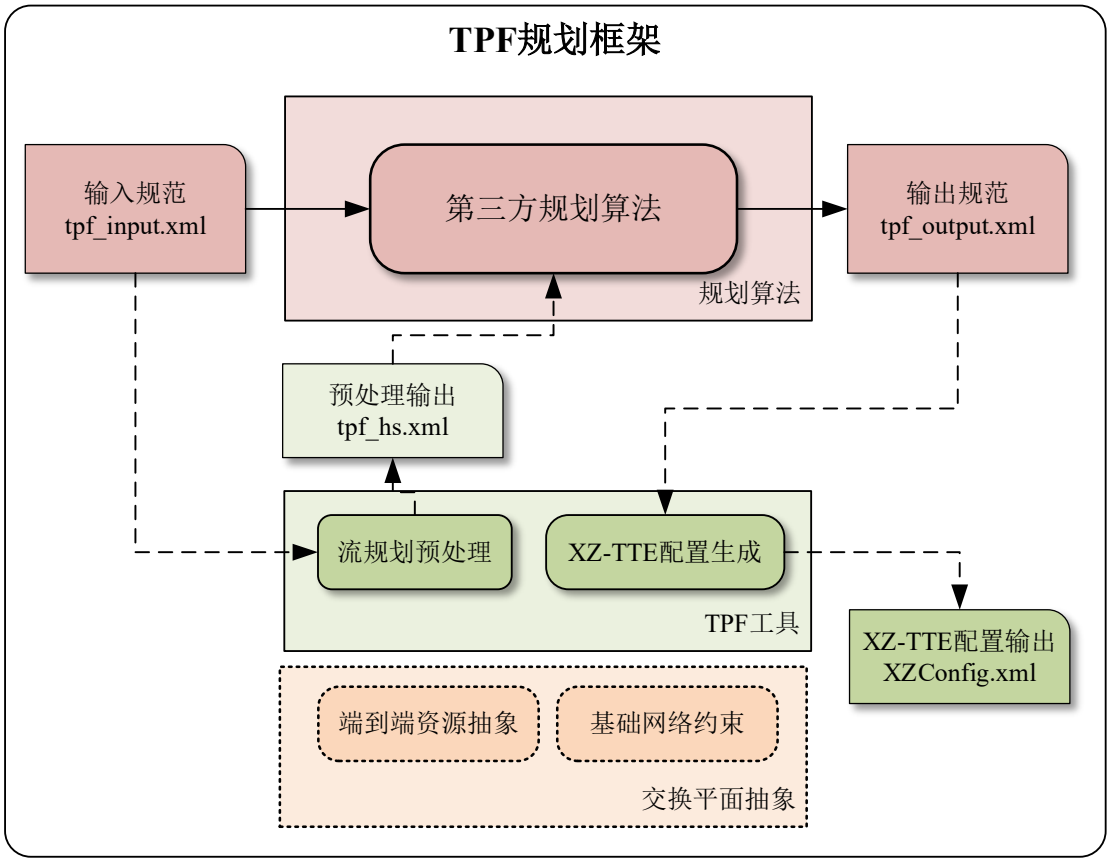


图 1 TPF 规划框架整体结构

（1）TTE 交换平面抽象

TTE 交换平面抽象提炼了 TTE 网络中规划算法的配置对象，将上层规划调度工具与底层数据平面设备设计实现解耦，是规划算法设计的基础。

（2）输入输出规范

输入输出规范定义了规划算法的输入和输出内容与格式，采用松耦合的 XML 文件形式。

（3）TPF 工具集

TPF 工具集包括了辅助算法实现和算法应用的各种工具软件，与 TTE 网络设备的实现相关。目前，开源 TPF 中提供两个工具：支持实现基于宏时隙的规划算法的流规划预处理工具，和支持将规划结果配置到 XZ-TTE 网络的 XZ-TTE 配置生成工具。

开发者可根据目标 TTE 网络的具体实现设计自己的工具。

2.2 TTE 交换平面抽象

TTE 交换平面抽象将上层规划调度与底层设备实现解耦，定义了规划算法的配置对象。上层规划调度软件以交换平面抽象为基础，实现网络资源分配，生成设备实现无关的流量规划结果。流量规划结果到底层设备配置的映射需要设备特定的适配层完成，例如 TPF 工具中提供的 XZ-TTE 配置生成工具。

TTE 网络包含三种流量类型：TT 流、RC 流和 BE 流。其中，BE 流无需规划，优先级最高的 TT 流在时间维度上对传输确定性要求高，其交换平面抽象最为复杂，因此，这里主要介绍针对 TT 流的端到端交换平面抽象，后续将补充对 RC 流交换平面的抽象。

TTE 交换平面抽象由资源抽象和约束条件组成。

2.2.1 资源抽象

资源抽象是对 TTE 交换平面的端系统和交换机对上层提供的可配置资源进行的描述，主要包括 TTE 网络节点上实现时间确定性数据传输需要的接收控制表和发送控制表。

（1）TT 接收控制表

TT 接收控制表定义网络节点何时、从哪个端口、接收哪条 TT 流。表项内容如表 1 所示：

表 1 TT 接收控制表

表项	说明
SlotID	时隙编号
PortID	端口号
StreamID	TT 流标识

（2）TT 发送控制表

TT 发送控制表定义网络节点何时、从哪些端口、发送哪条 TT 流。表项内容如表 2 所示：

表 2 TT 发送控制表

表项	说明
SlotID	时隙编号
StreamID	TT 流标识
PortMap	输出端口集合

2.2.2 约束条件

这里约束条件只包括网络约束条件，应用相关的约束条件在输入流规范中定义。TT 流规划时的网络约束条件如表 3 所示。

表 3 网络约束条件

约束	含义
无冲突约束	同一链路上不能有多条 TT 流的分组占用同一时间槽
路径依赖约束	同一条 TT 流的分组在上游交换机的发送时隙要小于下游交换机的发送时隙
Membound 约束	同一条 TT 流的分组在单个交换机上等待的时间槽数量要小于 Membound
端到端延迟约束	TT 流的分组的端到端延迟要在接收延迟窗口之内

2.3 输入输出规范

2.3.1 输入规范

输入信息规范定义规划算法需要的各种输入信息的内容和格式。输入信息内容包括网络拓扑结构、应用流特征、网络资源约束、以及应用约束四部分。输入信息格式采用 XML 文件形式，参见附录 A.1。

（1）网络拓扑结构

网络拓扑结构描述 TTE 网络的节点、链路和虚链路信息，如表 4 所示。

表 4　网络拓扑结构

类型	名称	描述
端节点	端系统标识	端系统标识号
	端口数目	端系统的端口数目
	端口标识	端系统的所有端口标识号
交换节点	交换机标识	交换机标识号
	端口数目	交换机的所有端口号
	端口标识	交换机的所有端口标识号
链路	链路标识	链路标识号
	上游节点标识	链路起始端的节点标识
	上游节点输出端口标识	链路起始端的输出端口标识
	下游节点标识	链路结束端的节点标识
	下游节点输入端口标识	链路结束端的输入端口标识
虚链路	链路序列	组成虚链路的链路序列集合

（2）应用流特征

应用流特征描述应用发出的 TT 流的属性，如表 5 所示。

表 5　应用流特征

TT 流特征	描述
流标识	用户对流量的标识
源端标识	帧的发送端系统标识
目的端标识	帧的接收端系统标识
周期	帧生成的时间周期
消息长度	帧的长度
发送窗口	帧从源端发送的时间窗口范围
接收窗口	帧到达目的端的时间窗口范围

（3）网络资源约束

表 6　网络资源约束

资源约束	描述
网络 MTU 值	网络中帧长度的最大值约束
端系统缓冲区规格	端系统中接收缓冲区和发送缓冲区的数量
交换机缓冲区规格	交换机中交换缓冲区的数量
链路带宽	网络链路带宽大小

（4）应用约束

应用约束与具体的应用场景需求相关，由上层用户进行定义。例如 PLC 闭环处理过程中，传感器到 PLC 的流 Flow A 和 PLC 到执行器的流 Flow B 之间有依赖关系，Flow B 的发送时间要晚于 Flow A 到达 PLC 的时间。

为便于扩展，应用约束采用<key, value>形式描述，由规划算法根据应用需求进行解释。

表 7　应用约束

应用约束	描述
key	约束名称
value	约束取值

2.3.2　输出规范

输出规范定义规划算法产生的规划结果的内容与格式。输出内容包括网络各个节点上的接收控制表和发送控制表。其中接收控制表位于接收端和交换机的接收方向，发送控制表位于发送端和交换机的发

送方向。输出格式采用 XML 文件形式，参见附录 A.2。

（1）接收控制表

接收端和交换机接收方向的接收控制表内容如表 8 所示。

表 8 接收控制表

表项	说明
节点 ID	节点标识号
流 ID	将要接收 TT 流的标识号
帧序列 ID	流在循环周期内的序列号
接收端口号	帧接收的端口号
宏时隙 ID	帧接收的宏时隙编号

（2）发送控制表

发送端和交换机发送方向的发送控制表内容如表 9 所示。

表 9 发送控制表

表项	说明
节点 ID	节点标识号
流 ID	将要发送 TT 流的标识号
帧序列 ID	流在循环周期内的序列号
输出端口号	帧输出的端口号
宏时隙 ID	帧输出的宏时隙编号

2.4 TPF 工具

TPF 工具是与网络实现相关的功能集合，由 TTE 网络开发者提供，用于支撑规划算法实现和规划结果配置。

目前，我们提供 XZ-TTE 网络相关的两个 TPF 工具：流规划预处理，和 XZ-TTE 配置生成。

2.4.1 流规划预处理

XZ-TTE 网络采用宏时隙机制实现 TT 流与 RC 流的解耦，极大地降低了混合流量规划调度复杂度，并使得 RC 帧有易于计算的确定性传输延迟上限。

XZ-TTE 流规划预处理的主要功能是将通用的应用流规范转换为基于宏时隙的流规范，方便后续规划算法的实现。

2.4.1.1 宏时隙简介

时隙机制是对链路带宽进行切片，实现不同类型流量的分时复用。而宏时隙（Huge Slot）的基本思想是对链路时隙进行“分类切片”，首先将链路时隙分为若干个“宏时隙”，每个宏时隙内部再分为 4 个固定长度的时段，包括 TT 时段、RC 时段以及两个保护带时段，如图 2 所示。

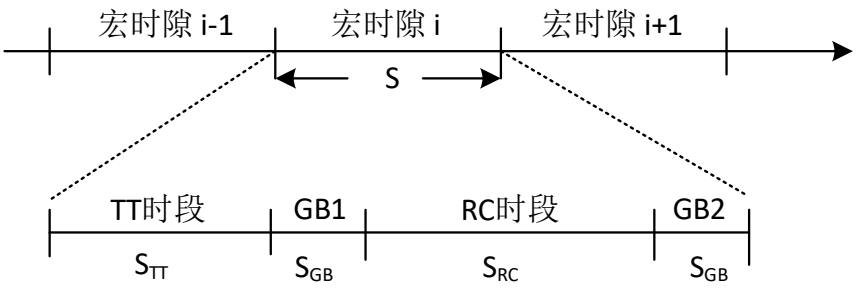


图 2 宏时隙组成

在离线规划时隙资源时，TT 时段用于传输 TT 帧和 BE 帧，如果离线规划没有为 TT 时段分配传输 TT 帧，或者 TT 帧传输结束，则

以传输 BE 帧。RC 时段用于传输 RC 帧和 BE 帧，在 RC 时段中如果 RC 队列为空，则可以传输 BE 帧。

保护带 GB2 用于确保在 TT 时段开始时，链路为空闲状态，即链路已经结束上一个宏时隙中所有数据的传输。保护带 GB1 用于确保在 RC 时段开始时，链路为空闲状态，即 TT 帧和 BE 帧的传输不会影响 RC 帧在 RC 时段开始时立刻发起传输。

需要注意的是，在每个 TT 时段中，只调度输出在上个宏时隙或更早到达交换机的 TT 帧，不会调度输出本宏时隙到达的 TT 帧。而在 RC 时段中，可以调度本时隙到达并就绪可调度输出的 RC 帧。

2.4.1.2 基于宏时隙的流规范

TPF 输入规范中定义了通用的应用流特征（表 5），可以形式化描述如下：

$$f_i = \{f_i.uid, f_i.src, f_i.dst, f_i.period, f_i.size, f_i.[Tx_{min}, Tx_{max}], f_i.[Rx_{min}, Rx_{max}]\}$$

其中$f_i.uid$为用户流标识，$f_i.src$为流发送端，$f_i.dst$为流接收端（组播时为集合），$f_i.period$为流周期，$f_i.size$为帧长度，$f_i.[Tx_{min}, Tx_{max}]$分别为发送窗口的时间下限和上限（发送窗口指分组从网卡发送到网络中的时间范围），$f_i.[Rx_{min}, Rx_{max}]$分别为接收窗口的时间下限和上限（这里接收窗口指应用从接收端网卡接收分组的时间范围）。

TT 流规划中通常会使用“**循环周期**”：循环周期为所有流周期的最小公倍数，循环周期内每条流会产生 1 个或多个分组。网络传输过程以循环周期为单位进行流量的重复发送，因此在规划过程中只需要为循环周期内的所有分组分配合适的时隙资源即可。

宏时隙流规范与通用流规范主要有两个区别：一是采用宏时隙为时间单位，二是将流在循环周期中的多条流实例描述为单独的流。形式化定义如下：

$$f_i = \{f_i.uid, f_i.seq, f_i.path\_schedwin, f_i.period, f_i.size\}$$

其中$f_i.uid$为用户流标识，$f_i.seq$为流在循环周期内的序列号，$f_i.path_schedwin$为流传输路径信息以及传输路径上每个节点可调度发送时间窗口，$f_i.period$为流周期，$f_i.size$为帧长度。

$f_i.path_schedwin$描述如下，分为发送端，交换机和接收端 3 部分：

$$f_i.path\_schedwin = \begin{cases} T_{id}, Outport, [schedTx_{min}, schedTx_{max}] & \text{发送端} \\ S_{id}, Inport, Outport, [schedTx_{min}, schedTx_{max}] & \text{交换机} \\ R_{id}, Inport, [schedRx_{min}, schedRx_{max}] & \text{接收端} \end{cases}$$

在发送端，T_{id}为发送端标识，$Outport$为输出端口，$[schedTx_{min}, schedTx_{max}]$为发送端可调度发送窗口的下限和上限。在交换机上，$S_{id}$为交换机标识，$Inport$为输入端口，$Outport$为输出端口，$[schedTx_{min}, schedTx_{max}]$为交换机可调度发送窗口的下限和上限。在接收端，$R_{id}$为接收端标识，$Inport$为输入端口，$[schedRx_{min}, schedRx_{max}]$为接收端可调度接收窗口的下限和上限。

2.4.1.3 预处理流程

应用流规范转换为宏时隙流规范的主要工作包括五部分：一是虚链路号分配；二是路径计算；三是生成循环周期内所有流实例；四是发送和接收时间窗口转换；五是时间窗口修正，如图 3 所示

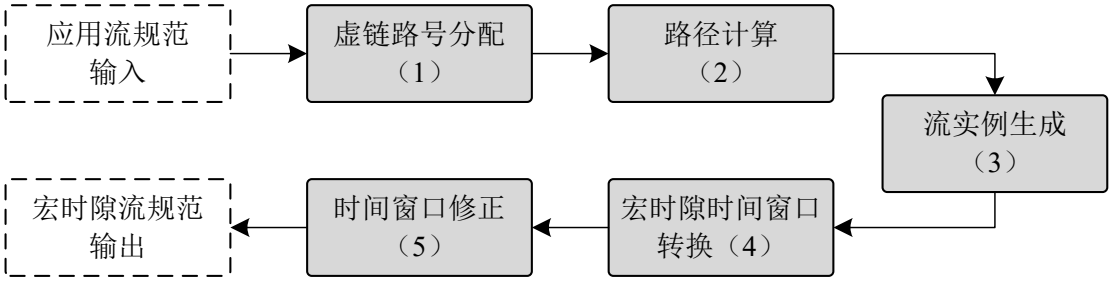


图 3　流规划预处理流程

（1）虚链路号分配

应用流规范中流标识为用户对流量的标识，网络层使用虚链路号对流量进行标识，在分组交换中基于虚链路号实现快速查表。虚链路号是每条流量的全局唯一标识，分配方式可以有多种，只要不产生冲突即可。

（2）路径计算

路径计算生成每条流从源端到目的端所经过的交换机和输入输出端口信息。

路径信息形式化描述如下：

$$f_i.path = \begin{cases} T_{id}, Outport & \text{发送端} \\ S_{id}, Inport, Outport & \text{交换机集合} \\ R_{id}, Inport & \text{接收端} \end{cases}$$

其中$T_{id}, Outport$为发送端标识和输出端口，$S_{id}, Inport, Outport$为经过的交换机标识以及交换机输入端口和输出端口，$R_{id}, Inport$为接收端标识和输入端口。

（3）循环周期内所有流实例生成

规划算法以帧为粒度进行时隙分配，因此需要得到循环周期内所有的帧信息。由于循环周期是所有流周期的最小公倍数，因此每条流

在循环周期内产生的帧数量以及每个帧的发送和接收窗口都可计算，具体计算方法说明如下：

由于循环周期是所有流周期的最小公倍数，因此每条流在循环周期内产生的帧数量$f_i.num$为：

$$f_i.num = \frac{cycle_{period}}{f_i.period} \tag{1}$$

其中$cycle_period$为循环周期大小，$f_i.period$为流的周期。

帧在循环周期内的序列号取值范围$f_i.seq$为：

$$f_i.seq = [0, f_i.num - 1]$$

对应的发送窗口的取值范围$f_i.Tx_{win}$为：

$$f_i.Tx_{win} = [Tx_{min} + f_i.seq * f_i.period, Tx_{max} + f_i.seq * f_i.period] \tag{2}$$

对应的发送窗口的取值范围$f_i.Rx_{win}$为：

$$f_i.Rx_{win} = [Rx_{min} + f_i.seq * f_i.period, Rx_{max} + f_i.seq * f_i.period] \tag{3}$$

（4）宏时隙发送和接收时间窗口转换

宏时隙发送和接收时间窗口转换主要功能是将以时间为单位的窗口转换为以宏时隙为单位的窗口。宏时隙的大小为所有流周期的公约数。时间窗口转换主要包括分为开始时间转换和结束时间转换。进行宏时隙时间窗口转换的原则是：转换后 TT 帧的可用时间范围不能大于转换前的时间范围。

宏时隙时间窗口的开始时间和结束时间转换公式如下：

- 开始时间转换

$$hs_{min} = \left\lceil \frac{t_{min}}{HSLen} \right\rceil \tag{4}$$

其中$HSLen$是宏时隙的长度。

- 结束时间转换

$$hs_{max}=\begin{cases}\dfrac{t_{max}}{HSLen}-1 & if\ t_{max}\ mod\ HSLen=0\\ \left\lfloor\dfrac{t_{max}}{HSLen}\right\rfloor-1 & if\ t_{max}\ mod\ HSLen<TTLen\\ \left\lceil\dfrac{t_{max}}{HSLen}\right\rceil-1 & if\ t_{max}\ mod\ HSLen\geq TTLen\end{cases} \quad (5)$$

其中，$HSLen$是宏时隙的长度，$TTLen$是宏时隙中 TT 时段的长度。

举例说明：设发送窗口为$[0,0.5ms]$，宏时隙长度为 200μs，TT 时段长度为 30μs，转换后以宏时隙为单位的发送窗口为 [0,2]。这里宏时隙 2（对应的时间范围为$[0.4ms,0.6ms)$）中的 TT 时段（对应时间为 0.4ms 到 0.43ms）满足发送时间需求。

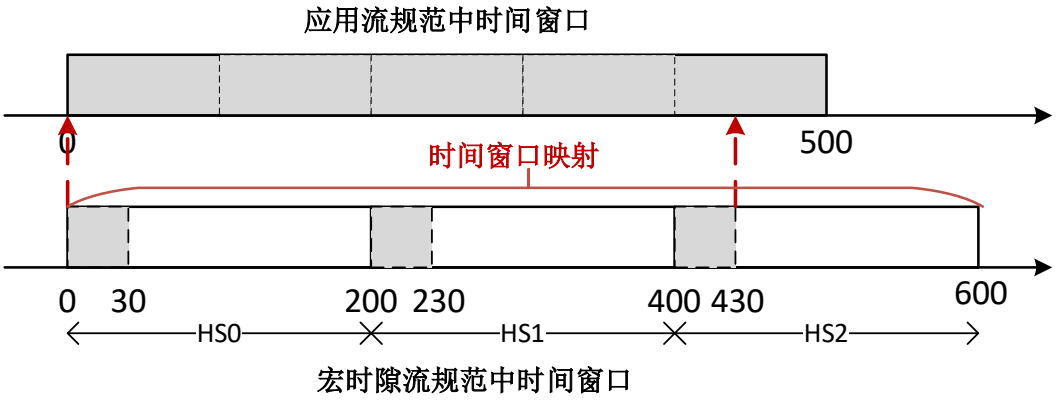


图 4　宏时隙时间窗口转换示例

（5）时间窗口修正

时间窗口修正的主要原因是由于交换机和接收端中缓冲区数量有限，帧从发送端发出后，其到达接收端应用的延迟有范围限制。时间窗口修正主要分为接收端可调度接收窗口修正、发送端可调度发送窗口修正和交换机可调度发送窗口修正。

时间窗口修正的计算方法如下：

在交换机端，上游节点在时隙i发出数据，下游节点可以在同一个

时隙i接收到该帧，可以在第$[i+1，\ i+Buf_{sw}]$个时隙发出，其中Buf_{sw}为交换机接口缓存数目。在接收端，上游节点在时隙i发出数据，接收端可以在同一个时隙i接收到该帧，可以在第$[i+1，\ i+Buf_{rx}]$个时隙提交给应用，Buf_{rx}为接收端接口缓存数目。

由于每个交换机对帧的延时范围是$[1, Buf_{sw}]$，接收端对帧的延时范围是$[1, Buf_{rx}]$，因此每条流的交换路径确定后，其拓扑对延时的约束也可以确定。设流f经过交换机个数为x，那么拓扑约束的最小延时ND_{min}和最大延时为ND_{max}满足：

$$ND_{min}(f) = x + 1$$

$$ND_{max}(f) = x \times Buf_{sw} + Buf_{rx}$$

- 接收端可调度接收窗口修正

由于网络延时约束，流发送窗口中发出的分组可能永远不会落到接收窗口中某些时隙上，因此需要对接收窗口大小进行修正。

流的可调度接收窗口$[schedRx_{min}, schedRx_{max}]$计算方法如下：

$$schedRx_{min} = \max(Rx_{min}, Tx_{min} + ND_{min}) \tag{6}$$

$$schedRx_{max} = \min(Rx_{max}, Tx_{max} + ND_{max}) \tag{7}$$

其中，Tx_{max}、Tx_{min}、Rx_{max}和Rx_{min}分别是根据流规范定义转换得到的最大最小发送时隙和接收时隙。

可调度接收窗口是接收窗口的子集，在规划中使用可调度接收窗口代替接收窗口，可以降低在输出接口调度的冲突概率，提高调度成功率。

- 发送端可调度发送窗口修正

同样，由于网络延时约束，从发送窗口中的某些时隙发出的帧永

远不会落在接收窗口内，因此需要对发送窗口进行修正，得到可调度的发送窗口$[schedTx_{min}, schedTx_{max}]$，计算公式如下：

$$schedTx_{min} = \max(Tx_{min}, schedRx_{min} - ND_{max})$$

$$schedTx_{max} = \min(Tx_{max}, schedRx_{max} - ND_{min})$$

代入公式(6)和(7)后，可得：

$$schedTx_{min} = \max(Tx_{min}, Rx_{min} - ND_{max}) \quad (8)$$

$$schedTx_{max} = \min(Tx_{max}, Rx_{max} - ND_{min}) \quad (9)$$

可调度发送窗口是发送窗口的子集，在规划中使用可调度发送窗口代替发送窗口，同样可以降低在输出接口调度的冲突概率，提高调度成功率。

- 交换机可调度发送窗口计算

根据流在源端的可调度发送窗口和接收端的可调度接收窗口，每条流可从源端开始，依次计算在中间交换节点输出接口的可调度发送窗口。

设流f_i的帧在网卡/交换机的输出接口可调度发送窗口为$(x1, y1)$，则在其下游交换机节点S_j的可调度发送窗口$(x2, y2)$应满足：

$$x2 = max(x1 + 1, f_i.schedRX_{min} - RSN(f_i, S_j) \times Buf_{sw} - Buf_{rx}) \quad (10)$$

$$y2 = \min(x2 + Buf_{sw} - 1,\ f_i.schedRX_{max} - RSN(f_i, S_j) - 1) \quad (11)$$

其中 RSN 指剩余交换机数量（Remaining Switch Number），其定义如下：$RSN(f_i, S_j)$是指流f_i在转发路径上位于交换机节点S_j之后的交换机数量。

$x2$在取值时需要考虑两点约束，一是至少等于$x1 + 1$，保证延迟确定性，二是避免下游交换机和接收端接口都缓存最长时间（延时为

$RSN(f,S) \times Buf_{sw} + Buf_{rx}$），分组的到达时间也小于可调度接收时间窗口的下限。

$y2$在取值时需要考虑两点约束，一是由于缓冲区数量限制，最大值为$x2 + Buf_{sw} - 1$；二是避免下游交换机和接收端接口都缓存最短时间（延时为$RSN(f,S) + 1$），分组的到达时间也大于可调度接收时间窗口的上限。

2.4.1.4 预处理输入输出

流规划预处理的输入为 TPF 输入规范 XML 文件（tpf_input.xml），输出为转换为宏时隙描述后的输入规范 XML 文件（tpf_hs.xml），可选地作为基于宏时隙规划算法的输入。

tpf_hs.xml 文件格式参见附录 A.3。

2.4.2 XZ-TTE 配置生成

XZ-TTE 配置生成将规划算法产生的规划结果（tpf_output.xml）转换为 XZ-TTE 网络设备配置文件，可直接配置到 XZ-TTE 网络进行测试验证。XZ-TTE 网络设备配置文件包括发送端配置表、交换机配置表和接收端配置表，转换后的 XZ-TTE 网络配置文件格式如附录 A.4 所示。

（1）发送端配置表

发送端配置表包含流映射表、注入控制表和发送控制表。

表 10　流映射表

表项	说明
节点 ID	节点标识号
节点 MAC	节点 MAC 地址
流特征	流分类的特征，例如流五元组信息
虚链路 ID	承载该条流的虚链路号

表 11　注入控制表

表项	说明
节点 ID	节点标识号
节点 MAC	节点 MAC 地址
虚链路 ID	承载该条流的虚链路号
注入端口号	帧注入的端口号（位于主机侧）
缓存区 ID	帧存放的 buffer 地址编号

表 12　发送控制表

表项	说明
节点 ID	节点标识号
节点 MAC	节点 MAC 地址
缓冲区 ID	帧存放的 buffer 地址编号
输出端口号	帧输出的端口号
宏时隙 ID	帧输出的宏时隙编号

（2）交换机配置表

交换机配置表包含接收控制表和发送控制表，其中接收控制表如表 13 所示，发送控制表如表 12 所示。

表 13 接收控制表

表项	说明
节点 ID	节点标识号
节点 MAC	节点 MAC 地址
输入端口号	帧输入的端口号
宏时隙 ID	帧输入的宏时隙编号
虚链路 ID	承载该条流的虚链路号
缓冲区 ID	帧存放的 buffer 地址编号

（3）接收端配置表

接收端配置表包含接收控制表、提交控制表和流逆映射表，其中接收控制表如表 13 所示，提交控制表如表 14 所示，流逆映射表如表 15 所示。

表 14 提交控制表

表项	说明
节点 ID	节点标识号
节点 MAC	接收端 MAC 地址
缓冲区 ID	表示帧存放的 buffer 地址编号
提交端口号	表示帧提交的端口号（端节点主机侧端口）
宏时隙 ID	表示帧提交的宏时隙编号

表 15　流逆映射表

表项	说明
节点 ID	接收端的节点标识号
节点 MAC	接收端 MAC 地址
虚链路 ID	帧的虚链路号信息
目的 MAC 地址	帧的目的 MAC 地址信息

3 基于 TPF 的 TTE 规划软件 XZ-Plan

3.1 XZ-Plan 软件架构

3.1.1 整体架构

XZ-Plan 规划软件基于 TPF 框架实现，提供人性化的图形配置和展示界面，软件实现了基于最大冲突接口的时隙分配算法，同时软件支持导入第三方规划算法的规划结果进行图形化展示和生成网络配置文件。软件架构如图 4 示，主要包含前端 UI 和后端规划逻辑两个部分。

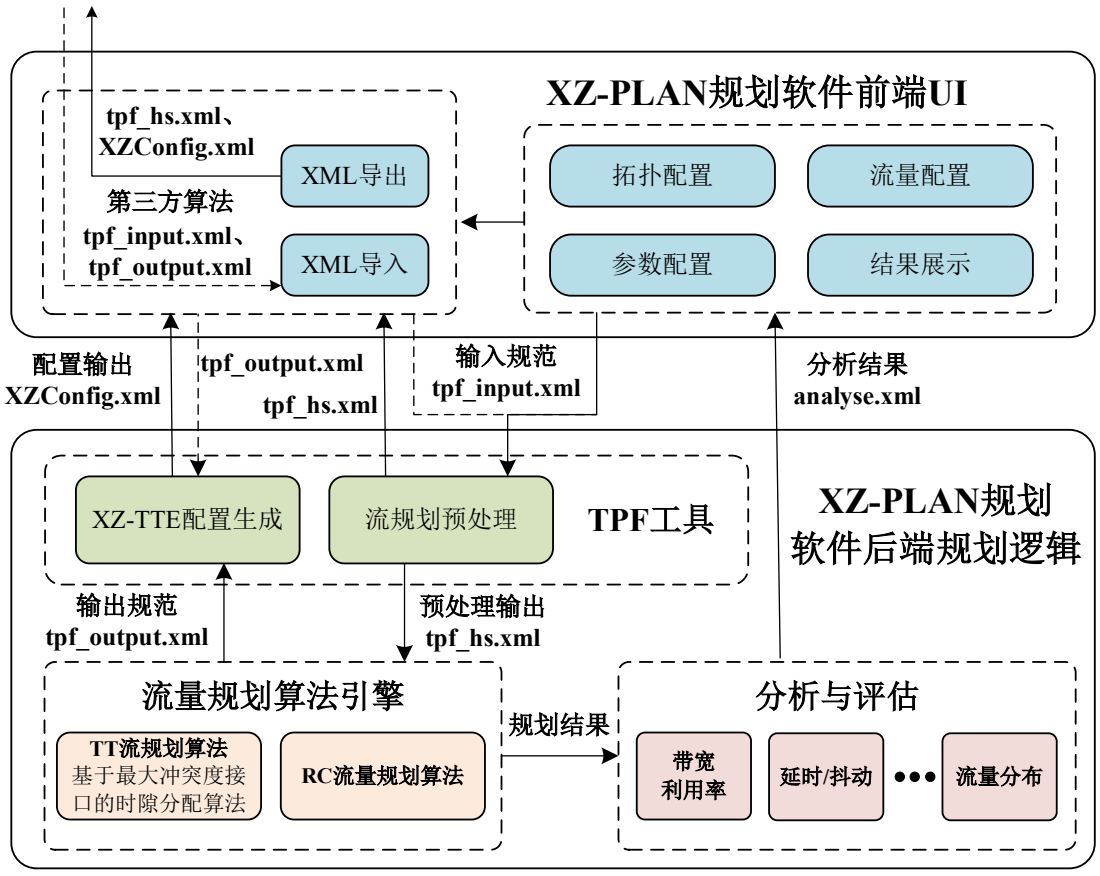


图 5 XZ-Plan 软件整体架构

前端 UI 包括拓扑配置、流量配置、参数配置、结果展示和 XML 文件导入导出模块。后端规划逻辑包括 TPF 工具模块、流量规划算法引擎模块和分析与评估模块。

3.1.2 核心功能

（1）UI 界面

UI 界面提供图形化接口，用于注入输入信息和展示规划结果信息。UI 界面包括三部分功能：一是提供图形化接口方便用户构建网络拓扑，进行流量设计，以及配置网络约束条件等参数；二是对规划算法的规划结果进行图形化展示，方便用户评估和分析；三是提供 XML 文件的导入导出功能，其中导入文件包括第三方软件生成的输入规范 XML 文件（tpf_input.xml）和第三方规划算法输出的规划结果 XML 文件（tpf_output.xml）；导出文件包括流规划预处理工具输出的 XML 文件（tpf_hs.xml）和 XZ-TTE 配置生成工具生成的 XML 文件（XZConfig.xml）。

（2）TPF 工具

TPF 工具包含基于宏时隙的流规划预处理工具和 XZ-TTE 配置生成工具。流规划预处理工具的主要功能是将通用的应用流规范转换为基于宏时隙的流规范，输入为通过 UI 界面输入的 TPF 输入规范 XML 文件（tpf_input.xml），输出为宏时隙描述的流规范 XML 文件（tpf_hs.xml）。XZ-TTE 配置生成工具的主要功能是将规划算法引擎或者第三方规划算法产生的规划结果转换为 XZ-TTE 网络设备配置

文件。

（3）流量规划算法引擎

规划算法引擎由 TT 流量规划模块和 RC 流量规划模块组成。TT 流量规划模块采用自研的基于最大冲突接口的启发式时隙分配算法实现，具体在 3.2 节介绍。RC 流量规划模块主要实现 RC 流量的资源规划。

（4）分析与评估

分析与评估模块的主要功能是对规划算法产生的规划结果进行静态分析，对带宽利用率、TT/RC 帧的端到端延时和抖动、流量分布情况等进行评估，生成评估结果供用户参考。

3.2 基于最大冲突接口的时隙分配算法

3.2.1 算法思想

基于最大冲突接口的时隙分配算法设计的主要根据是输出接口中 TT 流的传输时隙不能存在冲突的约束。因此时隙冲突最多的输出接口是网络中的热点和阻碍流量成功调度规划的瓶颈，因此优先解决时隙冲突最多的输出接口的资源分配问题对于提升规划的效率非常关键。

本算法的思路分为以下三部分：（1）找到网络中时隙竞争最激烈的输出接口，对该接口的时隙进行分配；（2）将经过该接口的流分割成子流，然后再找最大冲突接口，并进行该接口的时隙分配；（3）迭代前面的步骤，直到所有流在网络中所有接口都没有冲突时，随机为这些子流在无冲突的接口上分配时隙，算法结束。

3.2.2 算法处理流程

算法处理的主要流程包括以下 7 部分，如图 6 所示。

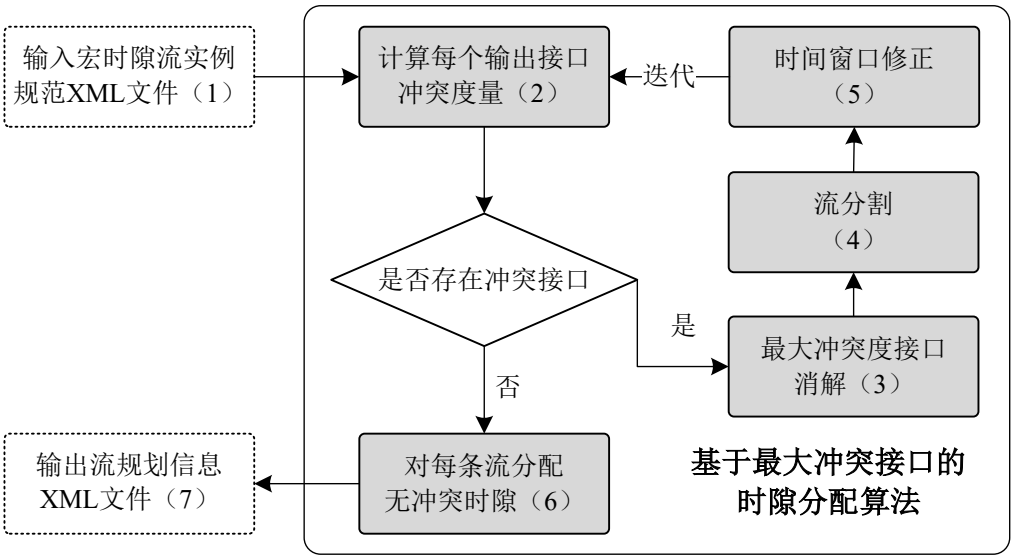


图 6 算法处理流程

（1）输入宏时隙流实例规范 XML 文件

算法的输入为 2.4.1 节中 XZ-TTE 流规划预处理工具的预处理结果，即将通用的应用流规范经过流量规划预处理模块处理后生成的宏时隙流实例规范，生成后的宏时隙流实例规范用 XML 文件进行描述，XML 文件格式参见附录 A.3。

（2）接口冲突度计算

输出接口的冲突度目前定义为该输出接口相互重叠的发送窗口个数，但不局限于这种量化方式。规划软件同时为用户提供编程接口支持用户对冲突度进行量化。接口冲突度计算的主要任务是计算每个输出端口的冲突度。

（3）最大冲突度接口消解

冲突接口的消解就是在冲突接口上，为不同的流求解一个不冲突的发送时隙。发送时隙的分配策略可以有多种，这里支持用户自定义设计。例如，最早开始优先的分配策略：从 0 开始循环，每次增 1，直到循环周期内时隙的最大值，依次为遇到窗口分配最小可行的时隙。

（4）流分割

正常情况下每条流的发送端都是 talker 网卡，接收端都是一个或多个 listener 的网卡。但在输出接口的冲突消解后，可能将一条子流分割成多条在调度时间上无关的“子子流”。因此每条“子子流”都有其不同的发送端和接收端。“子子流”的发送端称为发送锚，接收端称为接收锚。

（5）时间窗口修正

流经过分割和重新标定发送/接收锚后，需要对流可调度发送窗口进行修正。最大冲突接口经过冲突消解后，流在该接口上的发送时间和接收时间已经“锚定”，因此需要重新修正其他节点的可调度窗口，即把每个用发送/接收锚确定的“子子流”看成独立的流，重新调用 2.4.1 节的时间窗口修正公式，再次计算这些“子子流”的可调度窗口。

（6）对每条流分配无冲突时隙

每次迭代执行上述流程会消解一个冲突的发送接口，因此最多N次（N为发送接口数目）就可以保证网络中不再有冲突的发送接口。只要所有接口都没有冲突，可以在可调度窗口中任意选择发送时隙。在没有任何冲突时，可对每条流进行无冲突发送时隙分配。例如采用

最早时隙优先的无冲突发送时刻分配。

（7）输出流规划信息 XML 文件

算法的输出为 2.3.2 节描述的输出规范，即为每条流实例分配时隙资源后所生成规划后流调度信息，生成后的流调度信息用 XML 文件进行描述，XML 文件格式参见附录 A.2。

算法实现伪代码如图 7 所示：

Algorithm 1 基于最大冲突接口的时隙分配算法

输入：F, Nb
输出：流调度信息集合S
initial：初始化核心数据结构;
function TIMEWINDOWREVISE(F'', Nb)
  for each $f_i \in F'$ **do**
    for each $node \in f_i$ **do**
      if $node ==$ "$talker$" **then**
        $talker.[schTx_{min}, shcTx_{max}]$=$ReviseTxWindow(f_i, Nb)$
      else if $node ==$ "$bridge$" **then**
        $bridge.[schTx_{min}, shcTx_{max}]$=$ReviseTxWindow(f_i, Nb)$
      else$listener.[schRx_{min}, shcRx_{max}]$=$ReviseRxWindow(f_i, Nb)$
      end if
    end for
  end for
end function
function PLAN(F)
  PortNum = getportnum(F)
  repeat
    for $i = 0; i < PortNum; i++$ **do**
      $degree, port = max(computeconflict(F))$
      if $degree$ =!0 **then**
        $F' = ConflictResolve(port, F)$
        $F'' = FlowSlicing(F')$
        $F = TimeWindowRevise(F'', Nb)$
      end if
    end for
  until ($degree = 0$;break;)
  for each $f_i \in F$ **do**
    for each $node \in f_i$ **do**
      if $node ==$ "$talker$" **then**
        $talker.txslot = GetTalkerSlot(f_i, node)$
      else if $node ==$ "$bridge$" **then**
        $bridge.rxslot, bridge.txslot$=$GetBridgeSlot(f_i, node)$
      else$listener.rxslot$=$GetListenerSlot(f_i, node)$
      end if
    end for
  end for
end function

图 7　基于最大冲突接口的时隙分配算法

4 XZ-Plan 运行和验证环境

为了验证基于宏时隙机制的 XZ-Plan 规划软件的有效性，本文使用芯准 TTE 系列设备构建了支持宏时隙机制的 TTE 网络，作为控制平面 XZ-Plan 规划软件的运行和验证环境。

4.1 验证环境介绍

验证环境由 5 个芯准 TTE 交换机、3 个芯准 TTE 适配器、1 个芯准测试仪、1 台芯准 TTE 控制器终端、1 台 XZ-Plan 规划软件终端、1 台芯准 TTE 监控分析终端和 3 台普通 PC 终端组成，具体连接拓扑如图 8 所示。

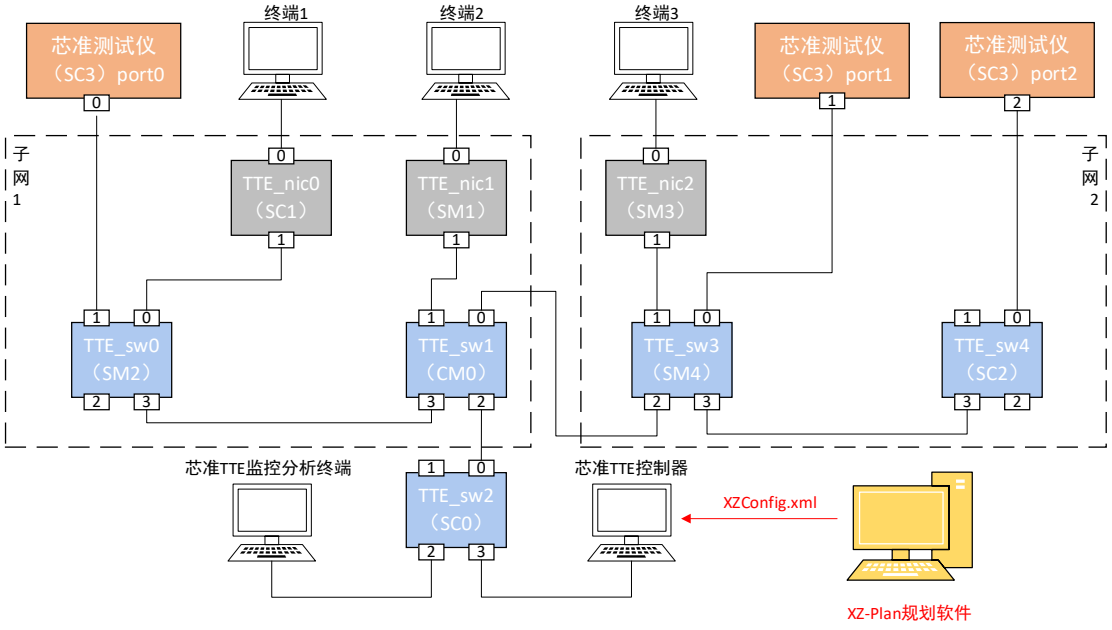


图 8　验证环境网络拓扑

演示环境中各组件功能描述如表 16 所示：

表 16　演示环境组件功能描述

组件名称	实现方式	功能
芯准 TTE 交换机	硬件	承载混合业务（实时/非实时）交换，流量类型支持 TT/RC/BE
芯准 TTE 适配器	硬件	实现用户终端的适配，通过适配器将用户终端接入到 TTE 网络中，支持 TT/RC/BE
XZ-Plan 规划软件	软件	实现对 TT/RC 业务流量的离线调度规划
芯准 TTE 控制器	软件	实现对整个 TTE 网络的集中管理和配置
芯准 TTE 监控分析软件	软件	实现对 TTE 网络在线/离线状态监控，支撑关键参数评估和分析
芯准测试仪	软/硬件	辅助用户完成 TTE 系统的测试和评估

演示环境实物图如图 9 所示：

图 9　演示环境实物图

4.2 验证环境组件

4.2.1 芯准 TTE 交换机

芯准 TTE 交换机主要功能与特性如表 17 所示：

表 17 芯准 TTE 交换机主要功能与特性

编号	主要特性
1	符合 AS6802 时钟同步协议规范
2	支持 SM、CM、SC 三种时钟同步角色
3	提供 4 个标准的千兆以太网接口，兼容 IEEE 802.3 标准以太网传输，支持时间触发消息（TT）、速率约束流量（RC 流）和尽力传输流量（BE 流）
4	支持时钟同步精度小于 100ns
5	支持 512 条虚拟链路（VL）
6	硬件对外支持故障注入接口，支持用户通过软件模拟故障注入
7	支持对外输出时钟同步脉冲，并提供接口可供示波器、测试仪等第三方设备进行时钟同步精度测量

4.2.1 芯准 TTE 适配器

芯准 TTE 适配器主要功能与特性如表 18 所示：

表 18　芯准 TTE 适配器主要功能与特性

编号	主要功能与特性
1	符合 AS6802 时钟同步协议规范，支持时钟同步精度小于 100ns
2	支持 SM、CM、SC 三种时钟同步角色
3	提供标准的千兆以太网接口，兼容 IEEE 802.3 标准以太网传输，支持时间触发消息（TT）、速率约束流量（RC 流）和尽力传输流量（BE 流）
4	支持网络侧不少于两个冗余端口
5	支持 512 条虚拟链路（VL）
6	硬件对外支持故障注入接口，支持用户通过软件模拟故障注入
7	支持对外输出时钟同步脉冲，并提供接口可供示波器、测试仪等第三方设备进行时钟同步精度测量

4.2.2　XZ-Plan 规划软件

XZ-Plan 规划软件主要功能与特性如表 19 所示：

表 19　XZ-Plan 软件主要功能与特性

编号	主要功能与特性
1	内嵌 TT 规划算法和 RC 规划算法，支持 TT 和 RC 混合流调度
2	提供图形化界面，支持用户通过图标拖拽方式构建网络拓扑
3	内嵌多种分析和评估插件，支持对带宽利用率、端到端延时和抖动、流量分布等关键参数进行静态分析和评估
4	支持对 TT 规划结果进行图形化展示
5	内嵌多种 TPF 工具，支持将规划算法规划结果直接转换为 XZ-TTE 网络配置文件
6	采用模块化、松耦合架构，提供用户开发接口，用户基于该接口可进行自定义的 TDP 工具和分析插件开发
7	支持将 XML 格式的第三方规划结果导入和 XZ-TTE 配置文件导出

4.2.3 芯准 TTE 控制器

芯准 TTE 控制器主要功能与特性如表 20 所示：

表 20 芯准 TTE 控制器主要功能与特性

编号	主要功能和特性
1	采用集中式架构，实现对整个 TTE 网络的集中配置和管理
2	支持 AS6802 时钟同步配置和管理，包括同步域划分、同步角色分配、时钟同步参数配置、同步状态监测等功能
3	支持在线和离线两种方式，实现 TT 调度表、设备参数、网络参数的配置加载
4	支持 netconf、EMP 等配置协议

4.2.4 芯准 TTE 监控分析软件

芯准 TTE 监控分析软件主要功能与特性如表 21 所示：

表 21 芯准 TTE 监控分析软件主要功能与特性

编号	主要特性
1	支持对 TTE 网络运行状态进行在线/离线监控，支持对 TT、RC、BE 带宽进行统计展示
2	支持数据收集、存储、分析、实时监控和流量重建
3	提供可视化的时钟同步状态机回溯功能，可供用户事后追溯 TTE 网络中任意节点的时钟同步状态机信息，并将各个节点的同步信息进行关联，辅助关键参数的分析与评估
4	提供可视化的 TT 流量回溯功能，可供用户进行 TT 流量重建，支持与离线规划结果进行对比分析，评估网络运行状态
5	支持故障注入功能，支持关键参数的分析和评估，支持分析 AS6802 多种故障模型，例如静默失效、非一致遗漏失效等

4.2.5 芯准测试仪

芯准测试仪主要功能与特性如表 22 所示：

表 22 芯准测试仪主要功能与特性

编号	主要功能与特性
1	支持 AS6802 时钟同步精度测试，支持被动授时同步和主动参与同步两种模式
2	支持 AS6802 协议一致性测试
3	支持流量调度特性测试，包括端到端延时测试、单跳交换机转发延时测试、TT 帧传输时序测试、TT 帧调度精度测试
4	支持 TTE 系统网络压力测试，可按照用户需求（指定速率、指定流量类型）进行 TT/RC/BE 混合传输测试
5	支持多种流量类型： 支持按需构造指定特征的 TT 流量，支持 TT 流总数目 16K； 支持按需构造指定特征的 RC 流量，支持 RC 流总数目 4K； 支持指定负载的 BE 流构造，支持 BE 流总数目 4K。

附录 A 输入输出 XML 文件定义

A.1 TPF 输入规范

TPF 输入规范 XML 文件 tpf_input.xml 格式如表 23 所示：

表 23 tpf_input.xml 文件定义

```
<tpf_input xmlns="urn:ietf:params:xml:ns:yang:tpf_input">
 <topology> <!—网络拓扑与网络资源约束-->
   <hosts>
       <host>
           <hostname>"A"</hostname>
           <nodeID>1</nodeID><!--节点ID，共64位，用于唯一地标识节点-->
           <opts>
               <MAC>"00:00:45:01:00:00"</MAC>
               <MTU>1500</MTU>
               <buffernum>4</buffernum>
           </opts>
       </host>
       <host>
           <hostname>"B"</hostname>
           <nodeID>2</nodeID>
           <opts>
               <MAC>"00:00:45:02:00:00"</MAC>
               <MTU>1500</MTU>
               <buffernum>4</buffernum>
           </opts>
       </host>
       <host>
           <hostname>"C"</hostname>
           <nodeID>3</nodeID>
           <opts>
               <MAC>"00:00:45:03:00:00"</MAC>
               <MTU>1500</MTU>
               <buffernum>4</buffernum>
           </opts>
       </host>
       <host>
           <hostname>"D"</hostname>
           <nodeID>4</nodeID>
```

```
            <opts>
                <MAC>"00:00:45:04:00:00"</MAC>
                <MTU>1500</MTU>
                <buffernum>4</buffernum>
            </opts>
        </host>
    </hosts>
    <switches>
        <switch>
            <switchname>"S1"</hostname>
            <nodeID>5</nodeID>
            <opts>
                <MAC>"00:00:66:01:00:00"</MAC>
                <portnum>4</portmum>
                <MTU>1500</MTU>
                <buffernum>4</buffernum><!--每个端口的缓冲区数量-->
            </opts>
        </switch>
        <switch>
            <switchname>"S2"</hostname>
            <nodeID>6</nodeID>
            <opts>
                <MAC>"00:00:66:01:00:01"</MAC>
                <portnum>4</portmum>
                <MTU>1500</MTU>
                <buffernum>4</buffernum><!--每个端口的缓冲区数量-->
            </opts>
        </switch>
    </switches>
    <links>
        <link>
            <srcnode>"A"</srcnode>
            <dstnode>"S1"</dstnode>
            <srcport>0</srcport>
            <dstport>0</dstport>
            <opts>
                <bandwidth>1000</bandwidth><!--链路带宽，单位为Mbps/s-
->
            </opts>
        </link>
        <link>
            <srcnode>"B"</srcnode>
            <dstnode>"S1"</dstnode>
            <srcport>0</srcport>
```

```
            <dstport>1</dstport>
            <opts>
                <bandwidth>1000</bandwidth><!--链路带宽，单位为Mbps/s-
->
            </opts>
        </link>
        <link>
            <srcnode>"C"</srcnode>
            <dstnode>"S2"</dstnode>
            <srcport>0</srcport>
            <dstport>2</dstport>
            <opts>
                <bandwidth>1000</bandwidth><!--链路带宽，单位为Mbps/s-
->
            </opts>
        </link>
        <link>
            <srcnode>"D"</srcnode>
            <dstnode>"S2"</dstnode>
            <srcport>0</srcport>
            <dstport>1</dstport>
            <opts>
                <bandwidth>1000</bandwidth><!--链路带宽，单位为Mbps/s-
->
            </opts>
        </link>
        <link>
            <srcnode>"S1"</srcnode>
            <dstnode>"S2"</dstnode>
            <srcport>2</srcport>
            <dstport>0</dstport>
            <opts>
                <bandwidth>1000</bandwidth><!--链路带宽，单位为Mbps/s-
->
            </opts>
        </link>
    </links>
</topology>
<virtuallinks><!—虚链路-->
    <virtuallink>
        <vlid>100<vlid>
        <datapath>"A,S1,S2,C"</datapath>
        <datapath>"A,S1,S2,D"</datapath>
    </virtuallink>
```

```
    <virtuallink>
        <vlid>101<vlid>
        <datapath>”B,S1,S2,D"</datapath>
    </virtuallink>
    <virtuallink>
        <vlid>102<vlid>
        <datapath>C,S2,D</datapath>
    </virtuallink>
    <virtuallink>
        <vlid>103<vlid>
        <datapath>”C,S2,D"</datapath>
        <datapath>”C,S2,S1,B"</datapath>
    </virtuallink>
</virtuallinks>
<flows><!—流特征-->
    <flow>
        <type>"TT"</type>
        <flowID>1</flowID>
        <period>2000</period>
        <size>1500</size>
        <vlid>100<vlid>
        <talker><!--源端标识-->
            <name>"A"</name>
            <txwin>[0,500]</txwin><!—发送时间窗口，单位为微秒-->
        </talker>
        <listeners><!--目的端标识，可能存在多个目的端-->
            <listener>
                <name>"C"</nodename>
                <rxwin>[1000,1100]</rxwin>
            </listener>
            <listener>
                <name>"D"</name>
                <rxwin>[1000,1100]</rxwin>
            </listener>
        </listeners>
    </flow>
</flows>
<constraints><!--应用约束，根据实际应用定义-->
    <constraint>
        <key>自定义值</key>
        <value>自定义值</value>
    </constraint>
</constraints>
</tpf_input>
```

A.2 TPF 输出规范

TPF 输出规范 XML 文件 tpf_output.xml 格式如表 24 所示：

表 24 tpf_output.xml 文件定义

```
<tpf_output xmlns="urn:ietf:params:xml:ns:yang:tpf_output">
<flows>
    <flow>
        <Instance>
            <flowID>1</flowID><!--流标识号-->
            <seq>0</seq><!--流在循环周期中的流实例标识，从0开始编号-->
            <path_slot><!--表示当前流实例传输路径上的节点集合-->
                <talker>
                    <nodeID>1</nodeID><!--节点标识-->
                    <outport>0</outport><!--发送端节点的发送端口标识-->
                    <txslotID>0</txslotID><!--发送端节点的发送时隙编号-->
                </talker>
                <bridges>
                    <bridge>
                        <nodeID>5</nodeID>
                        <inport>0</inport><!--交换机节点的输入端口标识-->
                        <rxslotID>0</rxslotID><!--交换机节点输入端口的接收
时隙编号-->
                        <outport>2</outport><!--交换机节点的输出端口标识-
->
                        <txslotID>1</txslotID><!--交换机节点输出端口的发送
时隙编号-->
                    </bridge>
                    <bridge>
                        <nodeID>6</nodeID>
                        <inport>0</inport>
                        <rxslotID>1</rxslotID>
                        <outport>2</outport>
                        <txslotID>5</txslotID>
                    </bridge>
                    <bridge>
                        <nodeID>6</nodeID>
                        <inport>0</inport>
                        <rxslotID>1</rxslotID>
                        <outport>1</outport>
                        <txslotID>5</txslotID>
                    </bridge>
```

```
            </bridges>
            <listeners>
                <listener>
                    <nodeID>3</nodeID>
                    <inport>0</inport><!--接收端节点的接收端口标识-->
                    <rxslotID>5</rxslotID><!--接收端节点的接收时隙编号-->
                </listener>
                <listener>
                    <nodeID>4</nodeID>
                    <inport>0</inport><!--接收端节点的接收端口标识-->
                    <rxslotID>5</rxslotID><!--接收端节点的接收时隙编号-->
                </listener>
            </listeners>
            </path_slot>
        </Instance>
    </flow>
</flows>
</tpf_output>
```

A.3 流规划预处理输出规范

流规划预处理工具输出规范 XML 文件 tpf_hs.xml 格式如表 25 所示，其中给出了一条流的描述：

表 25 tpf_hs.xml 文件定义

```
<tpf_hs xmlns="urn:ietf:params:xml:ns:yang:tpf_hs">
<flows>
    <flow>
        <Instance>
            <flowID>1</flowID><!—流标识号-->
            <vlid>100</vlid><!--帧实例的虚链路号-->
            <seq>0</seq><!—流在循环周期中的流实例标识，从0开始编号-->
            <period>10</period><!--帧周期，单位是宏时隙-->
            <size>1500</size><!—帧长度-->
            <path_schedwin><!—表示当前流实例传输路径上的节点集合-->
                <path>
                    <talker>
                        <nodeID>1</nodeID><!—节点标识-->
                        <outport>0</outport><!—发送端节点的发送端口标识-->
                        <schedtxwin>[0,2]</schedtxwin><!—发送端节点的可
调度发送时间窗口，以宏时隙为单位-->
                    </talker>
```

```
                    <bridge>
                        <nodeID>5</nodeID>
                        <inport>0</inport><!--交换机节点输入端口标识-->
                        <outport>2</outport><!—交换机节点输出端口标识-->
                        <schedtxwin>[1,4]</schedtxwin><!—交换机节点输出
端口的可调度发送时间窗口，以宏时隙为单位-->
                    </bridge>
                    <bridge>
                        <nodeID>6</nodeID>
                        <inport>0</inport>
                        <outport>2</outport>
                        <schedtxwin>[5,5]</schedtxwin>
                    </bridge>
                    <listener>
                        <nodeID>3</nodeID>
                        <inport>0</inport><!--接收端节点接收端口标识-->
                        <schedrxwin>[0,2]</schedrxwin><!—接收端节点的可
调度接收时间窗口，以宏时隙为单位-->
                    </listener>
                </path>
                <path>
                    <talker>
                        <nodeID>1</nodeID>
                        <outport>0</outport>
                        <schedtxwin>[0,2]</schedtxwin>
                    </talker>
                    <bridge>
                        <nodeID>5</nodeID>
                        <inport>0</inport>
                        <outport>2</outport>
                        <schedtxwin>[1,4]</schedtxwin>
                    </bridge>
                    <bridge>
                        <nodeID>6</nodeID>
                        <inport>0</inport>
                        <outport>1</outport>
                        <schedtxwin>[5,5]</schedtxwin>
                    </bridge>
                    <listener>
                        <nodeID>4</nodeID>
                        <inport>0</inport>
                        <schedrxwin>[5,5]</schedrxwin>
                    </listener>
                </path>
```

```
            </path_schedwin>
        </Instance>
    </flow>
</flows>
</tpf_hs>
```

A.4 XZ-TTE 配置生成输出规范

XZ-TTE 配置生成工具输出规范 XZConfig.xml 如表 26 所示：

表 26 XZConfig.xml 文件定义

```
<XZConfig xmlns="urn:ietf:params:xml:ns:yang:XZConfig">
  <node><!--发送端节点示例-->
    <nodeID>0</nodeID><!—节点标识-->
    <node_mac>"00:00:45:01:00:00"</node_mac><!—发送端节点MAC地址-->
    <tt_map_stable><!--流映射表-->
        <entry>
            <srcip>"202.197.162.10”</srcip>
            <dstip>"202.197.162.49"</dstip>
            <srcport>34500</srcport>
            <dstport>4949</dstport>
            <protocol>17</protocol>
            <vlid>100</vlid>
        </entry>
    </tt_map_stable>
    <tt_inject_stable><!--注入控制表-->
        <entry>
            <identity>0</identity>
            <inport>0</inport>
            <vlid>100</vlid>
            <bufferAddr>100</bufferAddr>
        </entry>
    </tt_inject_stable>
    <tt_send_stable><!--发送控制表-->
        <entry>
            <identity>0</identity>
            <outport>1</outport>
            <slotID>3</slotID>
            <bufferAddr>3</bufferAddr>
        </entry>
    </tt_send_stable>
  </node>
  <node><!--交换机节点示例-->
```

```
    <nodeID>2</nodeID><!—节点标识-->
    <node_mac>”00:00:45:01:01:00“</node_mac><!—交换机节点MAC地址-->
     <tt_recv_stable><!--接收控制表-->
        <entry>
            <identity>0</identity>
            <inport>1</inport>
            <slotID>0</slotID>
            <vlid>100</vlid>
            <bufferAddr>5</bufferAddr>
        </entry>
    </tt_recv_stable>
    <tt_send_stable><!--发送控制表-->
        <entry>
            <identity>0</identity>
            <outport>2</outport>
            <slotID>1</slotID>
            <bufferAddr>5</bufferAddr>
        </entry>
    </tt_send_stable>
  </node>
    <node><!--接收端节点示例-->
    <nodeID>3</nodeID><!--节点标识-->
    <node_mac>”0:00:45:55:01:00“</node_mac><!—接收端节点MAC地址-->
     <tt_recv_stable><!--接收控制表-->
        <entry>
            <identity>0</identity>
            <inport>1</inport>
            <slotID>1</slotID>
            <vlid>100</vlid>
            <bufferAddr>5</bufferAddr>
        </entry>
    </tt_recv_stable>
    <tt_submit_stable><!--提交控制表-->
        <entry>
            <identity>1</identity>
            <outport>0</outport>
            <slotID>2</slotID>
            <bufferAddr>5</bufferAddr>
        </entry>
    </tt_submit_stable>
    <tt_remap_stable><!--流逆映射表-->
        <entry>
            <vlid>100</vlid>
            <dstmac>”00:00:45:33:01:00“</dstmac>
```

```
        </entry>
    </tt_remap_stable>
  </node>
</XZConfig>
```

附录 B 算法示例

本节通过一个例子来说明基于最大冲突接口的时隙分配算法的实现过程。

B.1 网络拓扑与应用流规范

假设网络拓扑如图 10 所示，包含 A,B,C,D 四个节点和两个交换机 S1 和 S2。

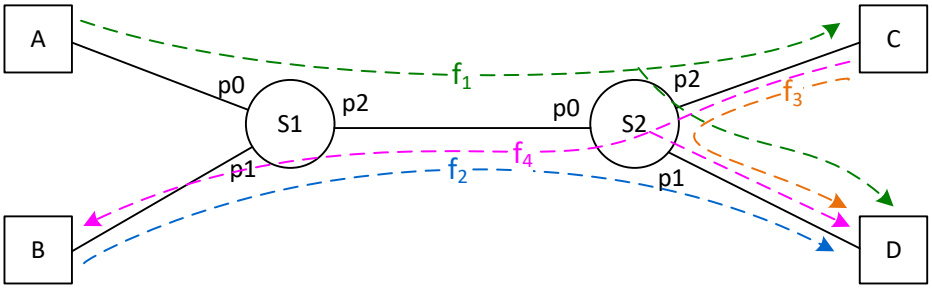


图 10 网络拓扑示例

应用流规范如表 27 所示：

表 27 应用流规范部分参数示例

流标识	源/目的端标识	周期	长度（字节）	发送窗口	接收窗口
f_1	A→{C, D}	2ms	1500	[0, 0.5ms]	C=[1ms,1.1ms] D=[1ms ,1.1ms]
f_2	B→D	2ms	64	[0.5ms, 1ms]	[1.4ms, 1.8ms]
f_3	C→D	3ms	256	[1ms, 1.5ms]	小于 3ms
f_4	C→{D, B}	4ms	512	[0ms, 2ms]	B=[2ms,3ms] D=[2ms, 3ms]

B.2 算法输入

算法的输入为 2.4.1 节流规划预处理工具输出的 tpf_hs.xml 文件。以$f_{1,0}$为例，内容包括流f_1的流标识、虚链路 ID、流在循环周期中的帧序列号、流周期、流长度、以及流实例的路径集合，其中路径集合包括发送端节点、中间交换机节点集合和接收端节点集合。

表 28 流$f_{1,0}$算法输入示例

表项	值	说明
flowID	1	流标识
vlid	100	虚链路 ID
seq	0	流在循环周期内的序列号
period	10	流周期
size	1500	流长度，字节
path_schedwin	/	路径集合，发送端节点见表 29

表 29 流$f_{1,0}$发送端节点示例

表项	值	说明
nodeID	1	发送端节点 ID 标识号
outport	0	发送端节点输出端口号
schedtxwin	[0,2]	发送端节点可调度发送时间窗口,以宏时隙为单位，

B.3 接口冲突度计算

示例中的应用流规范，经过流规划预处理后输出得到宏时隙的长

度为 200us，其中 TT 时段为 30us，调度周期为 60 个宏时隙（从 0 开始编号，即为[0,59]），以及所有 19 条流实例在每个输出接口的可调度发送时间窗口，如表 30 所示：

表 30 输出接口冲突示例

输出接口	可调度时间窗口	冲突度
A	f_1: [0,2], [10,12], [20,22], [30,32], [40,42], [50,52]	0
B	f_2: [3,4], [13,14], [23,24], [33,34], [43,44], [53,54]	0
C	f_3: [5,7], [20,22], [35,37], [50,52], f_4: [6,9], [26,29], [46,49]	2
D	——	0
S1.p0	——	0
S1.p1	f_4: [10,14], [30,34], [50,54]	0
S1.p2	f_1: [1,4], [11,14], [21,24], [31,34], [41,44], [51,54] f_2: [4,7], [14,17], [24,27], [34,37], [44,47], [54,57]	12
S2.p0	f_4: [7,13], [27,33], [47,53]	0
S2.p1	f_1: [5,5], [15,15], [25,25], [35,35], [45,45], [55,55] f_2: [7,8], [17,18], [27,28], [37,38], [47,48], [57,58] f_3: [6,11], [21,26], [36,41], [51,56]	8
S2.p2	f_1: [5,5], [15,15], [25,25], [35,35], [45,45], [55,55] f_4: [10,13], [30,33], [50,53]	0

定义 1：冲突度，输出接口的**冲突度**为该输出接口相互重叠的发送窗口个数。

定义 2：最大冲突接口，指输出接口中冲突度最大的接口。

示例中，S1.p2 冲突度为 12，为最大冲突接口。

调度问题实质上就是解决每个输出接口上的冲突问题。如果所有

输出接口的冲突度全为 0，则调度成功结束，此时，每条流都可以在每个接口上在其可调度发送窗口内随机分配时隙，而不会产生任何冲突。

B.4 冲突接口的消解

冲突接口的消解就是在冲突接口上，为不同的流求解一个不冲突的确定发送时间。冲突消解策略有多种方式，例如根据某种策略对流分优先级，优先分配优先级高的发送时隙，本文档采用一种最简单有效的冲突消解方法，即最早开始优先的分配策略。

最早开始优先的分配方法是从 0 开始循环，每次递增 1，直到调度周期内时隙的最大值（示例中为 59），依次为遇到窗口分配最小可行的时隙。根据上述方法，可为每条流分配确定的时间，如$f_{1,0}=1$，$f_{2,0}=3$等。

如果遇到两个窗口开始时间相同，但结束时间不同，优先为结束时间早的分配时隙；

如果遇到两个窗口开始时间相同，结束时间也相同，则为流 ID 小的优先分配时隙。

如果为某个发送窗口分配时隙失败（所有可行的时隙已经分配出去），则调度算法失败。

以 S1.p2 接口为例，基于最早开始优先的分配策略得到的流分配顺序和分配的发送时隙如表 31 所示：

表 31　最早开始优先分配策略示例

分配顺序	流 ID	窗口范围	分配时隙 ID
1	$f_{1,0}$	[1,4]	1
2	$f_{2,0}$	[4,7]	4
3	$f_{1,1}$	[11,14]	11
4	$f_{2,1}$	[14,17]	14
5	$f_{1,2}$	[21,24]	21
6	$f_{2,2}$	[24,27]	24
7	$f_{1,3}$	[31,34]	31
8	$f_{2,3}$	[34,37]	34
9	$f_{1,4}$	[41,44]	41
10	$f_{2,4}$	[43,47]	44
11	$f_{1,5}$	[51,54]	51
12	$f_{2,5}$	[54,57]	54

B.5 流的分割

正常情况下每条流的发送端都是 talker 网卡，接收端都是一个或多个 listener 的网卡。但在输出接口的冲突消解后，可能将一条子流分割成多条在调度时间上无关的“子子流”。因此每条“子子流”都有其不同的发送端和接收端。“子子流”的发送端称为发送锚，接收端称为接收锚。

由于接口 S1.p2 被判断为最大冲突接口，首先进行冲突消解，冲突消解后确定流$f_{1,0}$在该接口发送时隙为 1。由于该时间确定，从调度角度看，流 f1-0 被分成 3 条无关的子流。一是 A 到 S1.p2，二是 S2.p2 到 C，三是 S2.p1 到 D。

子流分割后，重新标定发送锚和接收锚如图 11 所示：

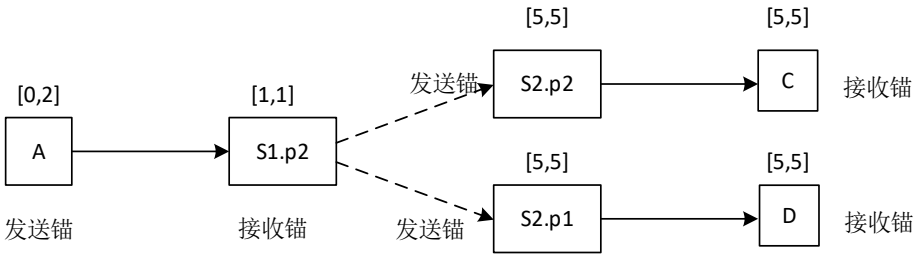


图 11　基于 S1.P2 接口分割后的流$f_{1,0}$示例

B.6 流状态修正

流经过分割和重新标定发送/接收锚后，需要对流状态（可调度发送窗口）进行修正。示例中最大冲突接口 S1.p2 经过冲突消解后，S1.p2 的发送时间，以及 S2 从 S1 的接收时间已经“锚定”，因此需要重新修正其他节点的可调度窗口，即把每个用发送/接收锚确定的“子子流”看成独立的流，重新调用 2.4.1 节的时间窗口修正公式，再次计算这些“子子流”的可调度窗口。

修正后的流可调度发送窗口如表 32 所示：

表 32　修正后的可调度发送窗口表

输出接口	$f_{1,0}(C/D)$	$f_{2,0}$	$f_{3,0}$	$f_{4,0}(B/D)$
A	[0,0]（原为[0,2]）			
B		[3,3]（原为[3,4]）		
C			[5,7]	[0,9]
D				
S1.p0				
S1.p1				[10,14]（B）
S1.p2	[1,1]（锚点）	[4,4]（锚点）		
S2.p0				[7,13]
S2.p1	[5,5]	[7,7]（原为[7,8]）	[6,11]	[10,13]（D）
S2.p2	[5,5]			

（1）第二轮接口冲突度计算

根据修正的发送窗口表，可对接口冲突度重新计算，第二轮接口冲突度计算结果如表 33 所示：

表 33　第二轮冲突计算结果

输出接口	可调度发送窗口	冲突度
A	f_1：[0,0]，[10,10]，[20,20]，[30,30]，[40,40]，[50,50]	0
B	f_2 :[3,3]，[13,13]，[23,23]，[33,33]，[43,43]，[53,53]	0
C	f_3: [5,7], [20,22], [35,37], [50,52], f_4: [6,9], [26,29], [46,49]	2
D		0
S1.p0		0
S1.p1	f_4: [10,14], [30,34], [50,54]	0
S1.p2	——	
S2.p0	f_4: [7,13], [27,33], [47,53]	0
S2.p1	f_1：[5,5]，[15,15]，[25,25]，[35,35]，[45,45]，[55,55], f_2：[7,7]，[17,17]，[27,27]，[37,37]，[47,47]，[57,57], f_3: [6,11], [21,26], [36,41], [51,56]	8
S2.p2	f_1 :[5,5]，[15,15]，[25,25]，[35,35]，[45,45]，[55,55], f_4: [10,13], [30,33], [50,53]	0

经过第二轮冲突计算，示例中，S1.p1 冲突度为 8，为最大冲突接口。

（2）第二轮接口冲突消解

根据最早开始优先分配策略，对 S1.p1 接口进行冲突消解，第二

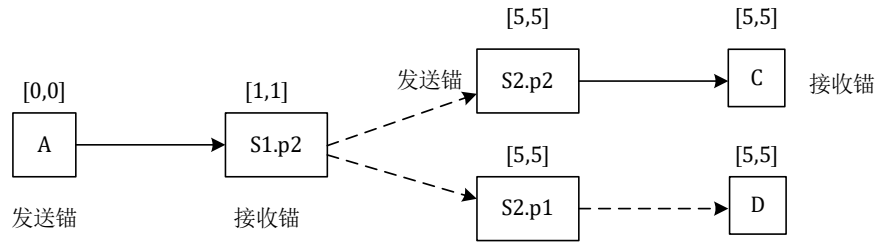
轮冲突消解结果如表 34 所示：

表 34 第二轮冲突消解结果（S1.p1）

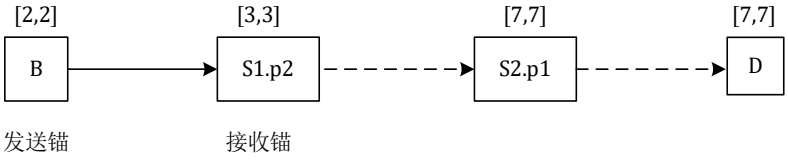
分配顺序	流 ID	窗口范围	分配时隙 ID
1	$f_{1,0}$	[5,5]	5
2	$f_{3,0}$	[6,11]	6
3	$f_{2,0}$	[7,7]	7
4	$f_{1,1}$	[15,15]	15
5	$f_{2,1}$	[17,17]	17
6	$f_{3,1}$	[21,26]	21
7	$f_{1,2}$	[25,25]	25
8	$f_{2,2}$	[27,27]-	27
9	$f_{1,3}$	[35,35]	35
10	$f_{3,2}$	[36,41]-	36
11	$f_{2,3}$	[37,37]	37
12	$f_{1,4}$	[45,45]	45
13	$f_{2,4}$	[47,47]	47
14	$f_{3,3}$	[51,56]	51
15	$f_{1,5}$	[55,55]	55
16	$f_{2,5}$	[57,57]	57

（3）第二轮流分割

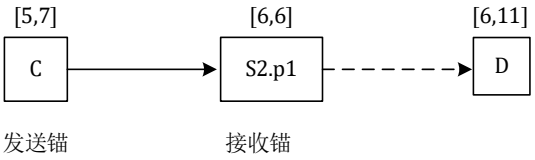
基于 S2.p1 接口的冲突消解结果，对 f1，f2 和 f3 流进行再分割，分割结果如图 12 所示：



（a）流$f_{1,0}$



（b）流$f_{2,0}$



（c）流$f_{3,0}$

图 12 基于 S2.p1 接口流分割

（4）第二轮流状态修正

第二轮修正后的流可调度发送窗口如表 35 所示：

表 35　第二轮修正后的流可调度发送窗口表

输出接口	$f_{1,0}\,(C/D)$	$f_{2,0}$	$f_{3,0}$	$f_{4,0}(B/D)$
A	[0,0]			
B		[3,3]		
C			[5,5]	[6,9]
D				
S1.p0				
S1.p1				[10,14]（B）
S1.p2（冲突消解）	[1,1]	[4,4]		
S2.p0				[7,13]
S2.p1（冲突消解）	[5,5]	[7,7]	[6,6]	[10,13]（D）
S2.p2	[5,5]			

（5）第三轮接口冲突度计算

根据修正的发送窗口，可对接口冲突度重新计算，计算结果如表 36 所示：

表 36　第三轮冲突计算结果

输出接口	可调度发送窗口	冲突度
A	f_1: [0,0], [10,10], [20,20], [30,30], [40,40], [50,50]	0
B	f_2: [3,3], [13,13], [23,23], [33,33], [43,43], [53,53]	0
C	f_3: [5,5], [20,20], [35,35], [50,50], f_4: [6,9], [26,29], [46,49]	0
D		0
S1.p0		0
S1.p1	f_4: [10,14], [30,34], [50,54]	0
S1.p2	——	
S2.p0	f_4: [7,13], [27,33], [47,53]	0
S2.p1	——	
S2.p2	f_1: [5,5], [15,15], [25,25], [35,35], [45,45], [55,55], f_4: [10,13], [30,33], [50,53]	0

第三轮冲突计算发现没有接口有冲突。

（6）无冲突时隙分配

在没有任何冲突时，可对每条流进行最早时隙优先的无冲突发送时刻分配。

以f_4的三条子流$f_{4,0}$、$f_{4,1}$和$f_{4,2}$为例。

（1）这三条子流在 C 发出的可调度窗口为[6,9],[26,29],[46,49],按照最早时隙优先原则，确定发送时隙为 6,26,46;

（2）在连接终端 D 的 S2. P2 接口，可调度窗口为:[10,13], [30,33], [50,53]，按照最早时隙优先原则，确定发送时隙为 10，30，50；

（3）在连接 S1 的 S2.p0 接口，可调度窗口为[7,13], [27,33], [47,53]，按照最早时隙优先原则，确定发送时隙为 7，27，47；

（4）在连接 B 的 S1.p1 接口，可调度窗口为[10,14], [30,34], [50,54]，按照最早时隙优先原则，确定发送时隙为 10，30，50；

上述流程说明，只要所有接口都没有冲突，可以在可调度窗口中任意选择发送时隙既可。无冲突时隙分配可完成对所有流在所有输出接口发送时隙最终的确定。

B.7 算法输出

算法的输出为 tpf_output.xml 文件。以$f_{1,0}$为例，内容包括流f_1的流标识、流在循环周期中的帧序列号、以及流实例的路径集合，其中路径集合包括发送端节点、中间交换机节点集合和接收端节点集合。

表 37　流算法输出示例

表项	值	说明
flowID	1	流标识
seq	0	流在循环周期内的序列号
path_path	/	路径集合，发送端节点见表 38

表 38　流$f_{1,0}$发送端节点示例

表项	值	说明
nodeID	1	发送端节点 ID 标识号
outport	0	发送端节点输出端口号
txslotID	0	发送端节点发送宏时隙编号

湖南华芯通网络科技有限公司

公司官网：www.c2comm.cn

联系电话：15616265487

邮箱：products@c2comm.cn