Cisco Packet Tracer

Packet Tracer 消息传递协议 (PTMP)

规格文档

修改历史

修订	日期	创始者	注释
1	2008年 1月10日	Michael Wang (miwang@cisco.com)	为公众而设
2	2008年 4月8日	Michael Wang (miwang@cisco.com)	更新了最新 PT （Packet Tracer） 版本的所有部分
3	2008年 7月3日	Michael Wang (miwang@cisco.com)	添加了 Keep-alive 部分
4	2020年 3月12日	Michael Wang (miwang@cisco.com)	添加了 ExApps 之间的多用户通信部分 添加了断开连接消息格式 更新了“连接协商消息保留”字段

目录

1. 引言

1.1. 目标

本文档介绍了数据包跟踪器消息传递协议 （PTMP）。 本文档还介绍了添加到数据包跟踪器的PTMP的规格。

1.2. 受众

本文档的范围适用于 Packet Tracer 外部应用程序 （ExApp） 开发人员。 它用于验证需求并描述详细的实现问题。

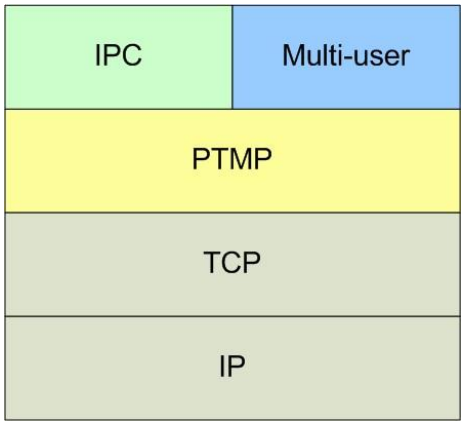
1.3. 摘要

本文档介绍支持Packet Tracer的IPC和多用户功能的PTMP功能，以及有关其架构设计的详细信息。

Packet Tracer 具有需要在不同 Packet Tracer 实例和/或其他应用程序之间进行网络通信的功能。此通信必须对其他组件统一且透明，以便它们可以轻松使用此通信。 PTMP通过连接协商、编码、加密、压缩和身份验证等方面实现此通信。

2. 架构

为了实现最佳网络利用率，PTMP 被可视化为一种通过 TCP/IP 工作的协议。这被设计为一种通用的消息传递协议。 使用 PTMP 的应用程序（如 PT 实例或其他应用程序）将称为 PTMP 应用程序。 在 PTMP 应用程序之间发送的消息不受 PTMP 控制。 利用 PTMP 的组件负责指定和遵循自己的消息传递格式和行为。 在多用户的情况下，这些消息可以是网络数据包，在IPC的情况下，这些消息可以是IPC呼叫。 通常，可以将 PTMP 视为应用程序可用于与 PT 实例通信的 TCP 覆盖。



确定的通信协议是 TCP。 其他协议，如UDP，也被考虑过。 但是，TCP是标准，在不同的平台和网络上提供了最大的多功能性和可靠性。

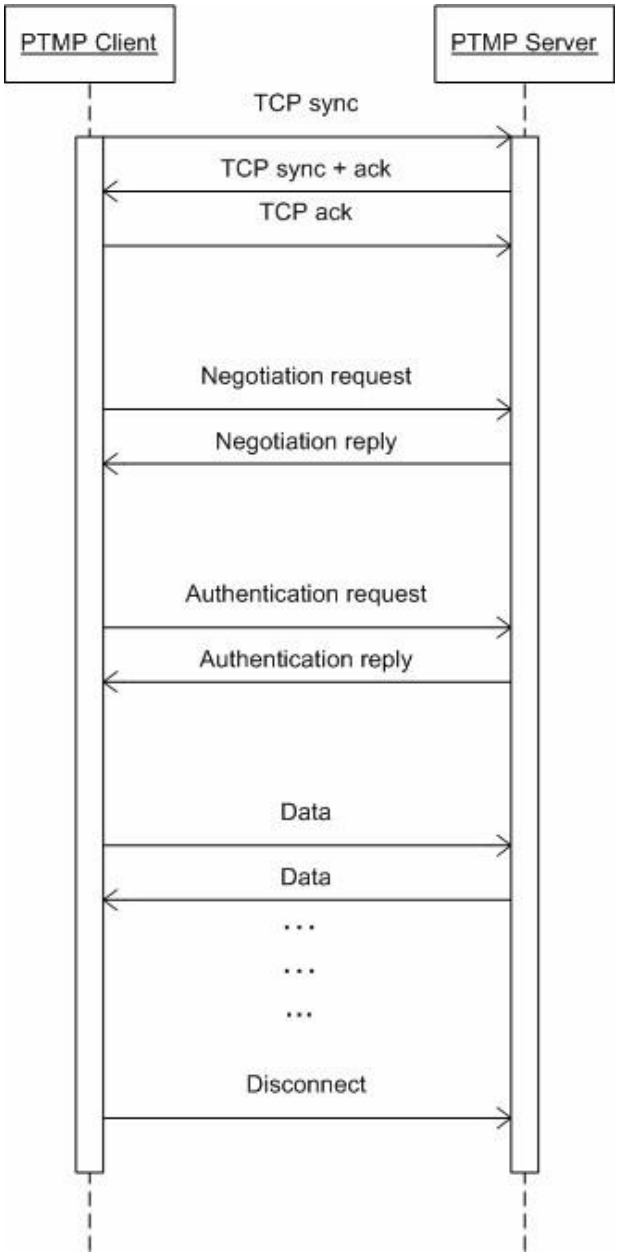
PTMP 应用程序的通信模型是基于 TCP 的客户端-服务器。 PTMP 应用程序可以开始侦听 TCP 端口，并允许其他实例连接到该端口。 多用户默认 TCP 端口为 38000，IPC 默认 TCP 端口为 39000。 这些端口号目前未在 IANA 注册的 TCP 编号中分配（http://www.iana.org/assignments/port-numbers）。使用 PTMP 的组件也可以更改端口。 其他 PTMP 应用程序可以通过提供 IP 地址和端口号连接到正在侦听的应用程序。

建立连接后，协商步骤用于确定两个 PTMP 应用程序将如何相互通信。 这包括身份验证、编码、加密和压缩方法的协商。 然后，使用身份验证步骤来验证客户端 PTMP 应用程序。

在两个 PTMP 应用程序之间建立 TCP 连接后，在协商和身份验证步骤之后，就没有服务器和客户端的概念了。 这两个应用程序具有相同的角色和功能。 现在，两个 PTMP 应用程序可以使用 PTMP 相互发送消息。

可以有多个连接到不同的 PTMP 应用程序。对于每个连接，都会建立一个专用的 TCP 会话。

当 PTMP 应用程序需要断开连接时，它会发送断开连接消息以正常断开连接。



3. 建模

3.1. 编码

在详细介绍该协议之前，我们必须解决 PTMP 中可用的编码以及每个连接期间选定的编码。为了向后兼容某些不支持二进制的开发平台，必须支持文本编码。 但是，二进制编码允许高效的转换和更短的消息，因此还需要支持二进制编码。

用于每个连接的编码在每个连接的开头协商。

PTMP 还指定了 PTMP 中本机支持的基本类型。 这些类型可以通过 PTMP 自动转换。 在这些基本类型之上构建的自定义类型也是可能的，但使用 PTMP 的组件负责指定和遵循这些自定义类型。

不同基本类型要遵循的编码格式如下。

名字	二进制编码	文本编码（均以 0 结尾）
byte	An 8-bit signed value	A signed value between -128 to 127
bool	An 8-bit value -- true and false	“true” or “false”
short	A 16-bit signed number	A signed number between -2^15 to 2^15 – 1
int	A 32-bit signed number	A signed number between -2^31 to 2^31 – 1
long	A 64-bit signed number	A signed number between -2^63 to 2^63 – 1
float	A single precision 32-bit	A decimal number
double	A double precision 64-bit	A decimal number
string	Variable-length Unicode characters terminated by \0	Variable-length Unicode characters terminated by \0
QString	Variable-length Qt Unicode characters terminated by \0	Variable-length Qt Unicode characters terminated by \0
IP address	A 32-bit value	An IP address in the x.x.x.x format
IPv6 address	A 128-bit value	An IPv6 address in the x:x:x:x:x:x:x:x format
MAC address	A 48-bit value	A MAC address in the xxxx.xxxx.xxxx format
uuid	A 128-bit value	A UUID in the {HHHHHHHH-HHHHHHHH-HHHH-HHHHHHHHHHHH} format

为每种数据类型分配最大宽度，以适应未来的要求和不同的编程语言。上述所有数据类型可能不是必需的。但为了完整起见，我们可以保留编码信息。 无符号类型不可用，因为 Java 等语言对无符号类型没有本机支持。

二进制编码指定每种值类型的长度或终止字符。 因此，分离值以进行读取是很简单的。 但是，文本编码中的值不会指定每种类型的长度。 终止字符 （0） 用于分隔读取值。

3.2. 通用消息格式

建立 TCP 连接后，将在 PTMP 应用程序之间发送消息。 这些消息不需要在一个 TCP 段中发送。 它可以是几乎无限长的（2^31 -1 = int 的最大数量），并在多个 TCP 段中发送。 但是，只有在收到整个消息时才会处理它们。 所有这些消息都必须遵循通用的消息格式，以便区分几种类型的消息并确定消息的结尾。 常用的长度类型值 （LTV） 用于所有 PTMP 消息。

字段如下：

- 长度 （int）： 消息中的字节数或字符数，不包括长度字段，但包括类型和值字段;此字段从不加密或压缩
- 类型 （int）： 指定消息的类型
- 值：可变长度

以下是“类型”字段中的不同值：

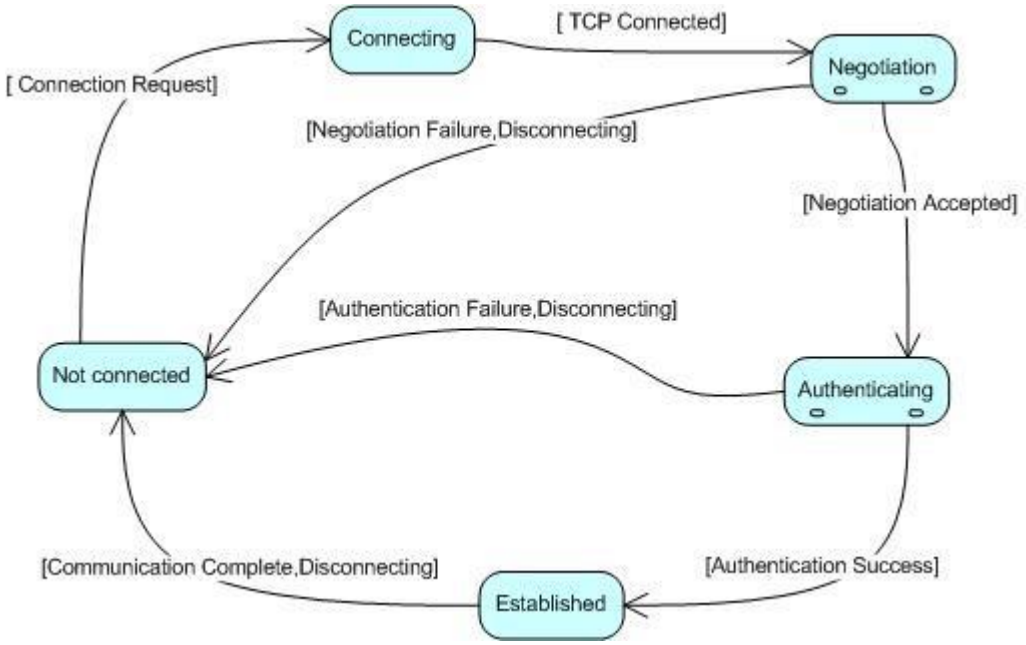
值	消息
0	Negotiation request
1	Negotiation response
2	Authentication request
3	Authentication challenge
4	Authentication response
5	Authentication status
6	Keep-alive
7	Disconnect
8	Communication
>= 100, < 200	IPC messages
>= 200, < 300	Multi-user messages

3.3. 状态图

PTMP 具有以下状态：

- 未连接：已创建但未启动 TCP 连接，或已断开连接
- 连接中：TCP 正在连接
- 谈判：交换连接信息
- 身份验证：交换用户名和密码
- 已建立：经过身份验证并完全建立

PTMP的简化状态图如下



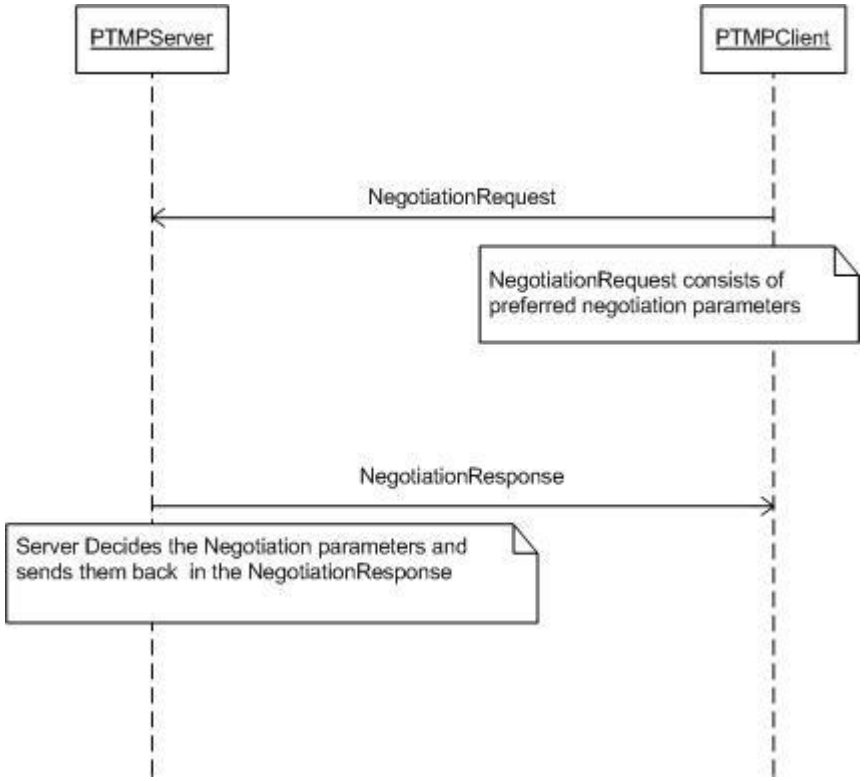
3.4. 连接协商

建立 TCP 连接后，两个 PTMP 应用程序的第一步是连接协商。 这是为了确定要在连接两端使用的一组通用属性。

PTMP 应用程序在协商期间使用以下消息格式交换信息：

- PTMP 标识符（字符串）： 常量标识符，“PTMP”
- PTMP 版本号 （int）： 1
- PTMP 应用程序 ID （uuid）： 发送此协商消息的应用程序的 UUID
- 编码 （int）： 1 = text，2 = binary
- 加密 （int）： 1 = none，2 = XOR
- 压缩 （int）： 1 = no，2 = zlib default
- 身份验证 （int）： 1 = clear text，2 = simple，4 = MD5
- 时间戳（字符串）： 发起连接的本地时间，格式为 YYYYMMDDHHMMSS
- 保持活动状态 （int）： 保持活动状态，以秒为单位
- Reserved（字符串）： 表示正在运行的 Packet Tracer 版本，格式为 :P TVER，<version>例如 :P TVER8.1.0.0000

客户端首先向服务器发送协商请求消息，指定所需的连接属性。 服务器在协商响应消息中使用已确定的连接属性进行回复。 整个协商过程采用文本编码。

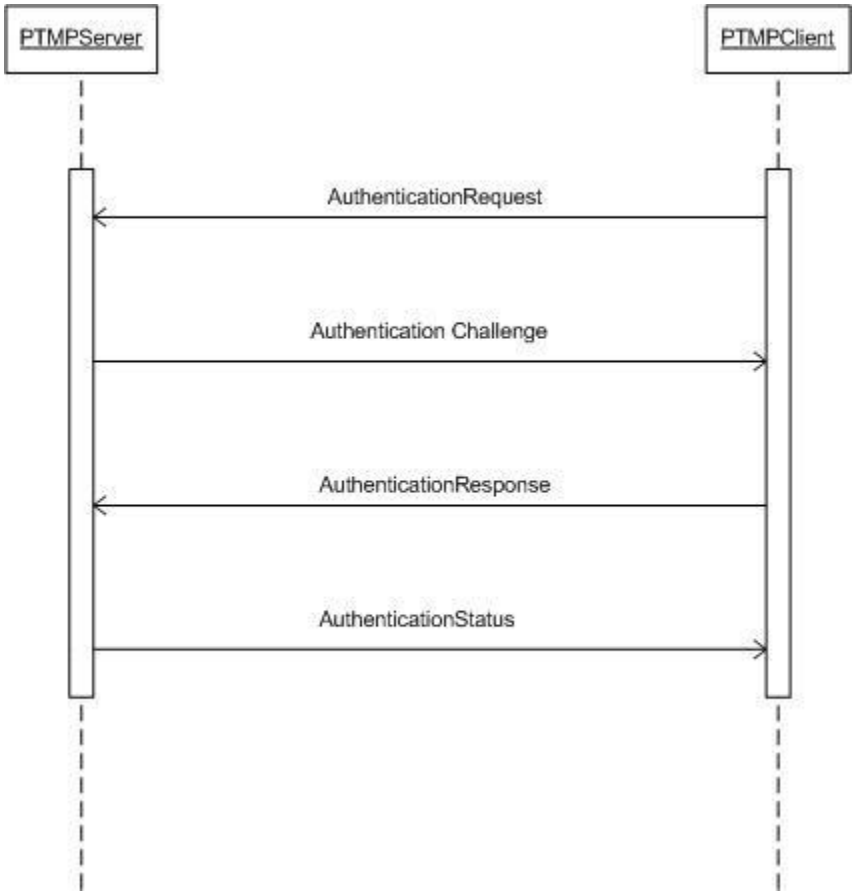


3.5. 身份验证

PTMP 遵循 CRAM（质询响应身份验证机制），这是一种用于身份验证的质询响应机制。

建立 TCP 连接和连接协商后，身份验证过程开始。假定使用 PTMP 的客户端和服务器应用程序都知道相同的凭据（用户名和密码，或 ID 和密钥）。这是成功进行身份验证的先决条件。

客户端首先发送身份验证请求消息。服务器通过发送身份验证质询消息来质询它。挑战是一个随机生成的字符串，由 32 个字符组成。 根据协商的身份验证方法，客户端可能需要通过应用哈希算法来计算“摘要”。 如果协商的身份验证方法是明文，则密码将以明文形式发回。 如果协商的身份验证方法是简单身份验证，则使用第 3.5.5 节中描述的简单身份验证方法来加密密码。 如果协商的身份验证方法是 MD5，则它将 MD5 与质询文本一起使用生成密码摘要。 摘要在身份验证响应消息中发送回。 服务器使用相同的身份验证方法验证摘要。 如果摘要匹配，服务器将向客户端发送回身份验证状态消息并结束身份验证过程。 如果摘要不匹配，则服务器会发回断开连接消息。



断开 TCP 会话后，将重新启动身份验证。

3.5.1. 认证请求消息格式

认证请求消息的字段如下：

- 用户名（字符串）：客户端的用户名或ID

3.5.2. 认证质询消息格式

认证质询消息的字段如下：

- 质询文本（字符串）

3.5.3. 认证响应消息格式

认证响应消息的字段如下：

- 用户名（字符串）：客户端的用户名或 ID
- 摘要文本（字符串）：使用协商的身份验证方法生成的密码摘要
- 自定义（字符串）：保留，当前未使用

3.5.4. 认证状态消息格式

认证状态消息的字段如下:

- 状态 （bool）： true = 成功，false = 失败

3.5.5. 断开消息格式

断开消息的字段如下:

- 原因（字符串）： 断开连接的原因;可以是空的

3.5.6. 简单认证方法

简单身份验证方法使用简单的哈希函数从给定的密码生成摘要。

```
函数 simple_hash（字符串密码）
{
        string hash;
        for (int i=0; i<password.length; i++)
                hash[i] = 158 – password[i];
        return hash;
}
```

3.6. 加密

机密性是 PTMP 中加密的唯一目的。 协商过程之后的每条消息都要加密，如果协商这样做的话。 加密方法是使用对称密钥对数据进行简单的异或（即服务器和客户端都使用相同的密钥进行加密和解密）。

加密密钥派生自 UUID 以及服务器和客户端的时间戳，顺序如下:

1. 服务器的 UUID 格式为 {HHHHHHHH-HHHH-HHHH-HHHH-HHHHHHHHHHHH}
2. 客户端的 UUID 格式为 {HHHHHHHH-HHHH-HHHH-HHHH-HHHHHHHHHHHH}
3. “PTMP”
4. 服务器的时间戳格式为 YYYYMMDDHHMMSS
5. 客户端的时间戳格式为 YYYYMMDDHHMMSS

此加密密钥将按以下方式用于对数据消息进行异或操作：

```
函数加密（数组数据、数组键）
{
        for (int i=0; i<data.length; i++)
                data[i] = data[i] ^ key[i % key.length];

}
```

加密功能也可以用作解密功能，以

3.7. 压缩

压缩方法是 zlib 的压缩。 请参见 http://www.zlib.net。 协商过程后的所有消息都将被压缩（如果协商这样做）。

3.8. 操作顺序

发送消息时，PTMP 遵循以下操作顺序：

1. 将带有类型和值字段的消息传递到 PTMP 中进行发送。
2. 如果状态正在验证或已建立：
    a. 如果协商压缩，则压缩消息
    b. 如果协商加密，则加密消息
3. 压缩和加密后预置邮件大小。

接收消息时，PTMP 遵循以下操作顺序：

1. 读取消息的大小。
2. 如果状态正在验证或已建立：
    a. 如果协商加密，则解密消息
    b. 如果协商压缩，则解压缩消息
3. 使用 PTMP 向组件发送消息

3.9. 保持活力

PTMP 中的 Keep-alive 机制用于检测与对等体的不知情断开连接。 保持活动期是在协商阶段协商的。 客户端发送所需的保持连接期，服务器将采用相同的数字。 该时间段以秒为单位，即保持活动消息分开发送的秒数。 如果协商的保持活动状态为零，则不会发送保持活动状态。 keep-alive 消息只是一个空的 PTMP 消息，其类型为 Keep-alive。 如果在保持活动状态期间的三次内未收到保持连接消息，则 PTMP 将认为对等方已断开连接，并通知 PTMP 应用程序。

3.10. ExApps之间通过多用户进行通信

在 Packet Tracer 中运行的 Packet Tracer 外部应用程序 （ExApp） 和脚本模块可以将消息发送到通过多用户远程连接的其他 ExApp 和脚本模块。 这允许在多个 Packet Tracer 实例上运行的同一 ExApp 或脚本模块之间进行同步。 此机制的详细信息超出了本文档的范围。 PTMP 仅指定通信消息的格式，而 Packet Tracer Multi-user 负责传递和处理消息。

通信消息字段如下:

- 源 ExApp 实例 ID （UUID）： 源 ExApp 或脚本模块的实例 ID
- 目标 ExApp 实例 ID （UUID）： 目标 ExApp 或脚本模块的实例 ID，如果在不知道 ExApp 实例 ID 的情况下发送到目标 ExApp ID，则为 null UUID
- 选项 （int）： 当前未使用
- 源 ExApp ID（字符串）： 源 ExApp 或脚本模块的 ID
- 目标 ExApp ID（字符串）： 目标 ExApp 或脚本模块的 ID
- 访问的 PT 实例数 （int）： 此消息访问的 PT 实例数;用于避免环路
- 已访问的 PT 实例 ID（列表<UUID>）： 此消息访问过的 PT 实例 ID 列表; 用于避免循环
- 消息（字符串）： 要传递到目标 ExApp 或脚本模块的消息