

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.

H04N 7/18 (2006.01)

H04L 29/12 (2006.01)



[12] 发明专利说明书

专利号 ZL 200710024639.3

[45] 授权公告日 2009年12月2日

[11] 授权公告号 CN 100566407C

[22] 申请日 2007.6.26

[21] 申请号 200710024639.3

[73] 专利权人 南京联创网络科技有限公司

地址 211100 江苏省南京市江宁区土山路
68号

[72] 发明人 俞海腾 赵峰 汪海林 徐进

杨富强 汪敏 史亮

[56] 参考文献

US7116679B1 2006.10.3

US20040095913A1 2004.5.20

WO2005069559A1 2005.7.28

审查员 张璇

[74] 专利代理机构 南京天翼专利代理有限责任公司

代理人 汤志武 王鹏翔

权利要求书2页 说明书7页 附图5页

[54] 发明名称

多种视频交换路由方式的瘦资源视频绑定策略方法

[57] 摘要

多种视频交换路由方式的瘦资源视频绑定策略，在用户使用前端设备(PU)和视频分发服务器(VTDU)设备资源的时候动态计算合适的视频转发分发路由，即由VTDU进行单视频点的多路转发，从而实现视频流的合理传输，以实现下述策略：用户(CU)和前端设备(PU)的绑定：设定PU直接向CU发送视频流；PU和视频分发服务器(VTDU)的绑定：设定VTDU只分发来自指定PU的视频流；CU和VTDU的绑定：设定特定的VTDU只能为指定的客户提供服务，以便保证该用户拥有足够的资源实现稳定的服务；CU和NRU的绑定：限定特定的NRU只为指定的客户提供服务；PU和NRU的绑定：设定特定的NRU只为指定的PU提供录像服务。

1、多种视频交换路由方式的瘦资源视频绑定的方法，其特征是在用户使用前端设备 PU 和视频分发服务器 VTDU 设备资源的时候：

- 1.1) 用户通过浏览器提交申请视频的请求，
- 1.2) 进行 udp 打洞，获取用户的对外 net 地址和端口私网穿透的技术，
- 1.3) 申请指令提交给 web 服务器，
- 1.4) 由 servlet 调用远程调度 EJB 接口，
- 1.5) 进行视频的业务处理，
- 1.6) 调度 EJB 计算最优路由算法，建立路由，
- 1.7) 执行打通路由操作，申请并传输视频；

动态计算合适的视频转发分发路由，即由 VTDU 进行单视频点的多路转发，从而实现视频流的合理传输，以实现下述策略：

- 1) 用户 CU 和前端设备 PU 的绑定：设定指定的 PU 直接向用户 CU 发送视频流；
- 2) PU 和视频分发服务器 VTDU 的绑定：设定特定的 VTDU 只分发来自指定 PU 的视频流；
- 3) 用户 CU 和 VTDU 的绑定：设定特定的 VTDU 只能为指定的客户提供服务，以便保证该用户拥有足够的资源实现稳定的服务；
- 4) 用户 CU 和 NRU 的绑定：限定特定的 NRU 只为指定的客户提供服务，以加强录像资料的机密性、保证获得独立的存储空间、保证拥有足够的网络资源；
- 5) PU 和 NRU 的绑定：设定特定的 NRU 只为指定的 PU 提供录像服务；

平台管理各个设备资源的动态路由表，选路策略由下表中指定的设备参数决定：

设备	策略	参数名	描述
PU	PU 的最大视频连接数	pu_max_out	≥ 0 , 0 表示禁止直接访问 PU, 若 PU 可用连接数 >2 则优先采用与 PU 直连的方式；必须留一个可用连接用于将视频发往 VTDU；
	PU 是否与 VTDU 绑定	pu_vtdu_bound	若绑定则优先将 PU 的视频发送到此 VTDU；单个 PU 可与多个 VTDU 绑定；
VTDU	VTDU 的最大视频流入连接数	vtdu_max_in	≥ 1
	VTDU 同一路视频同时被转发的路数	vtdu_max_out	≥ 1
	VTDU 负载分担模式是否启用	vtdu_lb_enabled	若启用则优先采用 VTDU 级联方式建立路由，若不启用则优先耗尽单个 VTDU 的容量
CU	CU 是否与 VTDU 绑定	cu_vtdu_bound	若绑定则优先使用此 VTDU 作为 CU 的视频源；单个 CU 可与多个 VTDU 绑定；
	CU 是否优先直连 PU	cu_pu_bound	若是则优先直连 PU

选择起点流程：选择路由表中选择第一个视频分发服务器 VTDU ，选择的流程如下述：开始：1、CU 与 VTDU 是否绑定？是则进入 3 否则进入 2；2、在所有 VTDU 中查找最合适的？是则成功，否则失败；3、查找最合适的绑定 VTDU？是则成功，否则进入 4；4、查找最合适的未绑定 VTDU 是则失败，否则成功；

选择最合适节点流程：最合适的 VTDU 查找流程时找不到再到未绑定 VTDU 服务器中进行选择最合适的；如果 PU 没有和任何的 VTDU 进行绑定，只需要在全部的 VTDU 中进行选择最合适的流程；11、查找有视频源的 VTDU？是则成功，否则进入 12；12、VTDU 负载是否均衡？是则进入 13，否则进入 14；13、查找最空闲的 VTDU？是则成功，否则失败；14、查找最忙的 VTDU 是则成功，否则失败；

终点选择流程，即选择路由表中选择第一个视频分发服务器 VTDU ，选择的流程：首先判断用户申请的视频服务器 PU 有没有和此 VTDU 进行直接绑定，如果用户和某 VTDU 进行了绑定，此 PU 的其所有的视频流的分发就需要用此类服务器进行分发：如 PU 同时绑定了多个视频分发服务器，就需要在其中进行选择最合适的；

选择有源节点流程： 在选择了起点和终点之后，如果起点和终点都没有用户所需要的视频源，则需要在所有的 VTDU 中进行选择有源的节点，在 PU 节点列表 Collection puColl 保存了所有 PU 视频的视频转发服务器的端口列表，根据负载均衡原理，在一组 VTDU 中查找输出最空闲的有源端口，并返回已知端口；

如果在此 PU 的节点列表中没有相关视频源，则需要到前端编码器进行新的视频申请，申请成功后，更新 puColl 中的数据结构，并在呼叫列表 callColl 中添加此次呼叫的信息，从而下次如果此次呼叫没有被中断，再有不同用户访问该视频点的图像时，只需通过此 VTDU 进行视频转发，不需要再到前端 PU 进行视频的申请。

多种视频交换路由方式的瘦资源视频绑定策略方法

一、技术领域

本发明涉及在对视频监控系统中视频资源的管理方法。

二、背景技术

当前，在传统网络视频监控领域，大多数是客户端直接和视频源连接，这种方式严重限制了监控视频的使用者数目，无法适应大规模应用的需求。

为此，人们开始使用软交换技术来解决这个问题。视频流的软交换技术一定程度上解决了视频源的网络带宽限制，允许更多的用户同时查看同一视频流。定义 VTDU 视频分发与转发单元来实现视频流的软交换功能。VTDU 主要完成视频信号的转发和分发，避免客户端直接访问前端摄像机，降低网络流量。

1. 转发：视频可通过流媒体服务器被转发，访问方只要访问流媒体服务器并告知要访问的前端设备，流媒体服务器可代为取到视频流并转发给该访问方。

2. 分发：一路视频通过 VTDU 可以被复制成多路送给不同的访问方。媒体交换层完成整个运营级视频监控系统的中心平台功能，它在运营支撑层的控制和管理下实现音视频的传送和存储。媒体交换层由 CMU、VTDU、NRU 以及其他相关的配套设备组成。部分或全部设备承载在现有运营商数据网之上。媒体交换层设备可向用户接入层的 PU 请求音视频，接受用户接入层 CU 的音视频传输请求，并与用户接入层协同建立传输通道。VTDU 作为中心平台的媒体处理单元，负责音视频的请求、接收和分发，实现多级级联和分布式部署。一路视频通过 VTDU 可以被复制成多路送给不同的访问方。每个 VTDU 仅接受本域 CMU 的管辖，同时可为其它域提供媒体访问接口。但是这种单一的视频流软交换方式对于电信级网络视频监控仍然存在许多不足，无法满足对不同客户提供差异化服务、根据网络环境的不同和实际业务时间覆盖情况，分别提供不同转发策略的应用需求。

四、发明内容

本发明目的是：针对上述问题，本发明提出多种视频交换路由方式的瘦资源视频绑定策略的方法，解决直连方式无法同时满路多名用户实时视频以及传统流路由转发方式过于单一，导致资源分配不合理而出现的资源浪费的问题，该发明是针对电信级运营商的视频绑定策略。

本发明的技术方案是多种视频交换路由方式的瘦资源视频绑定策略，在用户使用前端设备（PU）和视频分发服务器（VTDU）设备资源的时候动态计算合适的视频转发分发路由，即由 VTDU 进行单视频点的的多路转发，从而实现视频流的合理传输，以实现下述策略：

1) 用户 (CU) 和前端设备 (PU) 的绑定: 设定指定的 PU 直接向 CU 发送视频流;

2) PU 和视频分发服务器 (VTDU) 的绑定: 设定特定的 VTDU 只分发来自指定 PU 的视频流;

3) CU 和 VTDU 的绑定: 设定特定的 VTDU 只能为指定的客户提供服务, 以便保证该用户拥有足够的资源实现稳定的服务;

4) CU 和 NRU 的绑定: 限定特定的数字图像存储空间网络录像服务器 (NRU), 只为指定的客户提供服务, 以加强录像资料的机密性、保证获得独立的存储空间、保证拥有足够的网络资源;

5) PU 和 NRU 的绑定: 可以设定特定的 NRU 只为指定的 PU 提供录像服务。

另外, 考虑负载均衡的要求, 保证网络资源的充分利用和高可靠性,

联创视频监控系统平台提供上述策略的配置界面, 并在用户使用以上设备资源的时候动态计算合适的视频转发分发路由, 以实现上述策略。

通过灵活多变的视频交换路由策略, 并通过 VTDU 进行单视频点的多路转发, 从而实现视频流的合理传输, 最大限度的减少骨干网络的带宽负担, 是占用网络资源非常小的瘦资源模式。

本发明的中心平台(包括 CMU、VTDU、NRU)以及支撑系统(包括 AAA、ULS 和 NMS) 与前端设备单元 PU、客户端用户 CU 各部分均接入承载 IP 网络工作。

平台管理各个设备资源的动态路由表, 选路策略由下表中指定的设备参数决定:

表 1

设备	策略	参数名	描述
PU	PU 的最大视频连接数	pu_max_out	≥ 0 , 0 表示禁止直接访问 PU, 若 PU 可用连接数 > 2 则优先采用与 PU 直连的方式。必须留一个可用连接用于将视频发往 VTDU。
	PU 是否与 VTDU 绑定	pu_vtdu_bound	若绑定则优先将 PU 的视频发送到此 VTDU。单个 PU 可与多个 VTDU 绑定。
VTDU	VTDU 的最大视频流入连接数	vtdu_max_in	≥ 1
	VTDU 同一路视频同时被转发的路数	vtdu_max_out	≥ 1
	VTDU 负载分担模式是否启用	vtdu_lb_enabled	若启用则优先采用 VTDU 级联方式建立路由, 若不启用则优先耗尽单个 VTDU 的容量
CU	CU 是否与 VTDU 绑定	cu_vtdu_bound	若绑定则优先使用此 VTDU 作为 CU 的视频源。单个 CU 可与多个 VTDU 绑定。
	CU 是否优先直连 PU	cu_pu_bound	若是则优先直连 PU

流程为：

- 1.1 用户通过浏览器提交申请视频的请求
- 1.2 进行 udp 打洞，获取用户的对外 net 地址和端口私网穿透的技术；
- 1.3 申请指令提交给 web 服务器
- 1.4 由 servlet 调用远程调度 EJB 接口
- 1.5 进行视频的业务处理
- 1.6 调度 EJB 计算最优路由算法，建立路由
- 1.7 执行打通路由操作，申请并传输视频。

选择起点流程的方法是：选择路由表中选择第一个视频分发服务器(VTDU)，选择的流程如下述：

开始：1、CU 与 VTDU 是否绑定？是则进入 3 否则进入 2；2、在所有 VTDU 中查找最合适的？是则成功，否则失败；3、查找最合适的绑定 VTDU？是则成功，否则进入 4；4、查找最合适的未绑定 VTDU？是则失败，否则成功；

选择最合适节点流程：最合适的 VTDU 查找流程时找不到再到未绑定 VTDU 服务器中进行选择最合适的；

如果 PU 没有和任何的 VTDU 进行绑定，只需要在全部的 VTDU 中进行选择最合适的。11、查找有视频源的 VTDU？是则成功，否则进入 12；12、VTDU 负载是否均衡？是则进入 13，否则进入 14；13、查找最空闲的 VTDU？是则成功，否则失败；14、查找最忙的 VTDU？是则成功，否则失败；

终点选择流程，即选择路由表中选择第一个视频分发服务器(VTDU)，选择的流程是：首先判断用户申请的视频服务器(PU)有没有和此 VTDU 进行直接绑定，如果用户和某 VTDU 进行了绑定，此 PU 的其所有的视频流的分发就需要用此类服务器进行分发。如 PU 同时绑定了多个视频分发服务器，就需要在其中进行选择最合适的。

选择有源节点流程：

在选择了起点和终点之后，如果起点和终点都没有用户所需要的视频源，则需要所有的 VTDU 中进行选择有源的节点，在 PU 节点列表 Collection puColl 保存了所有 PU 视频的视频转发服务器的端口列表，根据负载均衡原理，在一组 VTDU 中查找输出最空闲的有源端口，并返回已知端口； 如果在此 PU 的节点列表中没有相关视频源，则需要到前端编码器进行新的视频申请，申请成功后，更新 puColl 中的数据结构，并在 callColl 中添加此次呼叫的信息，从而下次如果此次呼叫没有被中断，再有不同用户访问该视频点的图像时，只需通过此 VTDU 进行视频转发，不需要再到前端 PU 进行视频的申请。

本发明的有益效果是：

- 通过对 PU 的最大视频连接数的判定，最大的合理利用 PU 的解码能力，充分保留了传统直连方式的优点，同时保留一路 PU 的视频连接数供视频流转发服务器

转发视频流，可以避免直连方式对同时浏览实时视频的户数有上限的弊端，受限于 PU 的解码能力，直连的方式单通道可以同时解码的能力一般不超过六路实时视频，而该发明完全可以使单通道同时拥有的浏览用户量达到成百甚至上千。

● 通过对最合适视频流转发服务器，有源视频流转发服务器，最空闲源视频流转发服务器，最忙碌源视频流转发服务器的查找和选择的多种视频交换路由方式避免了传统视频流转发服务器路由转发策略单一，资源分配不合理的缺点。传统视频流转发服务器路由转发策略是基于单个转发服务器负载量达到峰值的情况下选择其它服务器，在资源浪费的同时，由于单台服务器过高的负载量，使视频流的稳定性和实时器也有一定程序的下降，如果基于小企业，接入 PU 数量不是太大的情况下，比如说几十至上面个 PU，该方式可以比较稳定的运行，但如果 PU 接入的数量成千乃至上万时，如电信级运营商，该方式就可能在短时间内因太过于庞大的业务流量而导致整个系统崩溃。

● 在各个不同的域之间使用跨域路由策略，可以指定特定的视频流转发服务器作为接口，在漫游时优先使用此 VTU 作为视频的入口或出口，并合理的使用并控制资源，使之真正的达到电信运营级要求。

四、附图说明

图 1 本域实时视频浏览进行详细说明，

图 2 申请视频指令提交给 EJB 进行路由计算流程图

图 3 路由表的结构

图 4 选择的流程

图 5 VTU 查找流程

图 6 选择的流程

图 7 寻找最空闲的 VTU 进行转发的寻找结构示意图

五、具体实施方式

下面结合具体实施例，进一步阐述本发明。

现以本域实时视频浏览进行详细说明，如图 1：

1. 流程说明

1.1 用户通过浏览器提交申请视频的请求

1.2 进行 udp 打洞，获取用户的对外 net 地址和端口

1.3 申请指令提交给 web 服务器

1.4 由 servlet 调用远程调度 EJB 接口

1.5 进行视频的业务处理

1.6 调度 EJB 计算最优路由算法，建立路由

1.7 执行打通路由操作，申请并传输视

2. 实现原理

由于用户的 pc 很可能是内网地址，由于在如今公网 IP 地址如此紧张的今天，通过内网访问 Internet 变得非常的普通，所以需要私网穿透的技术。通过第三方进行 UDP 打洞操作，获取用户的对外 net 地址和端口。假设 client A 要向 client B 对话,但是 A 不知道 B 的地址,即使知道根据 NAT 的原理这个对话在第一次会被拒绝,因为 client B 的 NAT 认为这是一个从没有过的外部发来的请求.这个时候,A 如果发现自己没有保存 B 的地址,或者说发送给 B 的会话请求失败了,它会要求 server 端让 B 向 A 打一个洞,这个 B->A 的会话意义在于它使 NAT B 认为 A 的地址/端口是可以通过的地址/端口,这样 A 再向 B 发送对话的时候就不会再被 NAT B 拒绝了。这样来实现打洞操作。

申请视频指令提交给 EJB 进行路由计算，计算流程图如图 2，分别根据是否要 PU 直连，离用户最近的一台视频转发服务器上面是否有视频流，就可以判断是否进行此视频的重新申请，如果有则只需要该视频转发服务器进行转发，这样就可以满足多用户同时查看同一路用户的需求，这样可以最大限度的保证，编码器只有一路视频发往视频转发服务器，而转发服务器可以将其视频流转发给任意多的终端用户。

紧接着查询终点，并查看其转发服务器上面有没有视频源，如有此用户所需要的视频源，则直接进行转发，如没有视频源，则需要所有系统转发服务器中进行查找是否有此视频源的服务器，有则利用此服务器进行转发，如没有发现则到前端编码器进行视频的重新申请，使得视频流按已生成的视频分发路由的顺序进行视频传输。

本地路由的策略如表 1:

3. 详细说明

3.1 数据结构介绍

使用来进行调度数据的存储，路由表的结构如图 3 所示，

用 PU 节点列表 Collection puColl 来保存所有 PU 视频的视频转发服务器的端口列表，每个列表单元为 PuNode，其数据结构是：

```
String subPuId,
Collection ports, //包含 PortNode
Collection directCalls,
Collection totalCalls //pu 的所有呼叫
```

而 PortNode 的数据结构为：

```
PortInfo portInfo; PortNode parent;
Collection directCalls; Collection totalCalls; Collection children;
```

使用呼叫列表 callColl 来保存所有视频呼叫的列表，每个呼叫列表的节点内容为 CallInfo,而 CallInfo 中包含了每次视频申请的具体属性，其中包含了此次呼叫生成的路由表信息 path，每个 path 节点内容为 PortInfo，结构为：

```
protected int portNo;
protected String objId; protected String objIp;
```



```
protected String srcObjId;      protected int outPort;
```

使用 vtduColl 来保存所有视频转发服务器的信息，每个节点包括本 VTDU 上所有视频接收端口列表，及其可用状态，数据结构为：

```
protected String vtduId;
```

```
protected int state = StateType.RESPONDING;
```

```
protected Collection ports = null; // 本 VTDU 上所有视频接收端口
```

使用 nruColl 来保存所有视频转发服务器的信息，每个节点包括本 VTDU 上所有视频接收端口列表，用于播放的客户端列表，及其可用状态，数据结构为：

```
protected String nruId;
```

```
protected int state = StateType.RESPONDING;
```

```
protected Collection portsRecv = null; // 用于录像的本地接收端口列表
```

```
protected Collection portsSend = null; // 用于播放的客户端列表
```

3.2 流程说明

3.2.1 起点选择流程

起点选择流程，即选择路由表中选择第一个视频分发服务器(VTDU)，选择的流程如图 4 所示。图 4 中，1、CU 与 VTDU 绑定？是则进入 2 否则进入 3；2、在所有 VTDU 中查找最合适的？是则成功，否则失败；3、查找最合适的绑定 VTDU？是则成功，否则进入 4；4、查找最合适的未绑定 VTDU？是则失败，否则成功；

首先判断用户有没有和此视频分发服务器进行直接绑定，如果用户和某视频分发服务器进行了绑定，其所有的视频流的分发就需要用此服务器进行分发。如用户同时绑定了多个视频分发服务器，就需要在其中进行选择最合适的，最合适的 VTDU 查找流程如图 5，找不到再到未绑定 VTDU 服务器中进行选择；

如果用户没有和任何的 VTDU 进行绑定，只需要在全部的 VTDU 中进行选择最合适的。

3.2.2 终点选择流程

终点选择流程，即选择路由表中选择第一个视频分发服务器(VTDU)，选择的流程如图 6 所示。

首先判断用户申请的服务器(PU)有没有和此 VTDU 进行直接绑定，如果用户和某 VTDU 进行了绑定，此 PU 的其所有的视频流的分发就需要用此类服务器进行分发。如 PU 同时绑定了多个视频分发服务器，就需要在其中进行选择最合适的，最合适的 VTDU 查找流程如图 5，找不到再到未绑定 VTDU 服务器中进行选择最合适的；

如果 PU 没有和任何的 VTDU 进行绑定，只需要在全部的 VTDU 中进行选择最合适的。

3.2.3 选择有源节点

在选择了起点和终点之后，如果起点和终点都没有用户所需要的视频源，则需要所有的 VTDU 中进行有源的节点，由于在 PU 节点列表 Collection puColl 保存了所有 PU 视频的视频转发服务器的端口列表，根据负载均衡原理，在一组 VTDU 中查找输出最空闲的有源端口，并返回已知端口。

如果在此 PU 的节点列表中没有相关视频源，则需要到前端编码器进行新的视频申请，申请成功后，更新 puColl 中的数据结构，并在 callColl 中添加此次呼叫的信息，从而下次如果此次呼叫没有被中断，再有不同用户访问该视频点的图像时，只需通过此 VTDU 进行视频转发，不需要再到前端 PU 进行视频的申请，从而增加视频服务器的压力。

3.2.4 选择最合适节点

在同时有多个 VTDU 可以选择的情况下，需要选择最合适的节点，其选择的步骤如下：

首先需要查找有视频源的 VTDU 节点，用户所查看的前端编码器(PU)的视频源能够从此 VTDU 进行转发，有就可以直接返回成功；如果没有则判断是否需要进行负载均衡，如需要，则寻找最空闲的 VTDU 进行转发，寻找结构示意图如图 7，寻找最空闲的 VTDU 分为输入最闲和输出最闲两种，并且算法很相似，现就输入最闲的 VTDU 的选择是首先设置一个最大的整数值 V，通过对所有符合条件的 VTDU 的设备进行遍历，根据每个设备的输入视频路数和当前的最大整数进行比较，如果此设备的视频输入数 S 小于此整数 V，则把 S 赋给 V，每次遍历设备的时候都会进行类似的比较，如果输入数大于 V，则继续下一个 VTDU，直到所有的 VTDU 被遍历完，此时的 V 就是这组 VTDU 设备中，视频输入数最少的，此 VTDU 也就是输入最空闲的。

如不需要进行负责均衡，则寻找最繁忙的 VTDU 进行转发，结构图和图 7 类似。

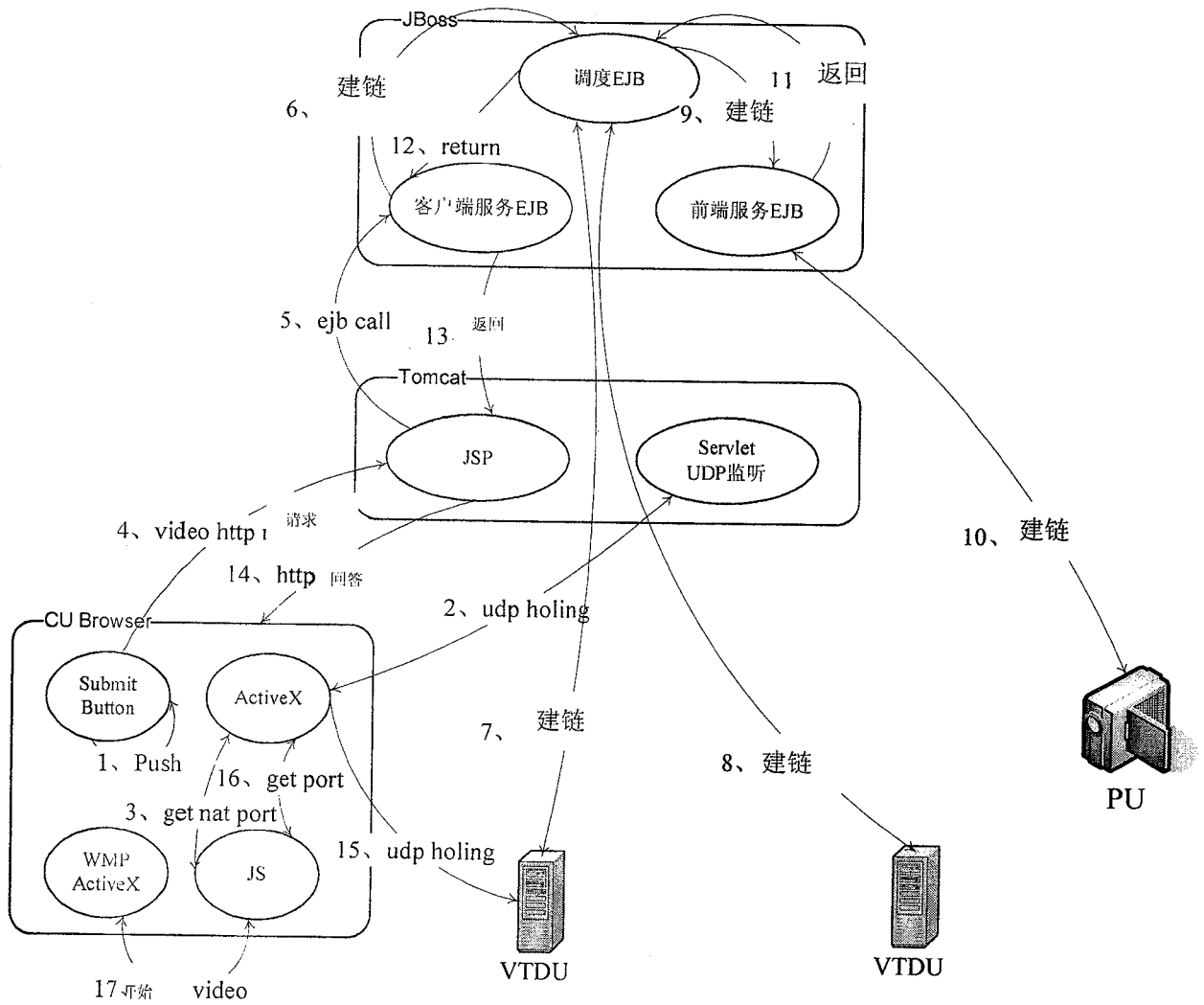


图 1

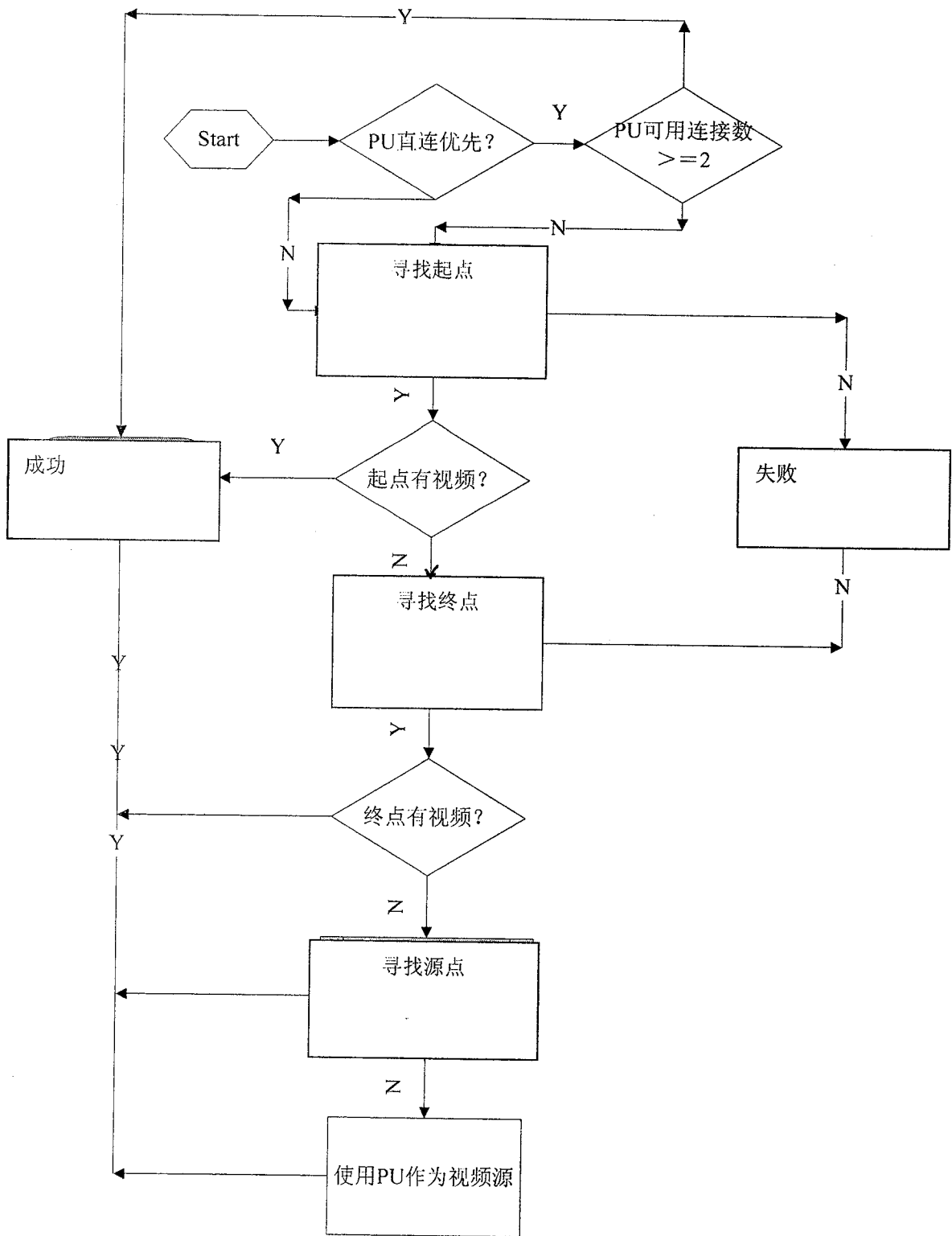


图 2

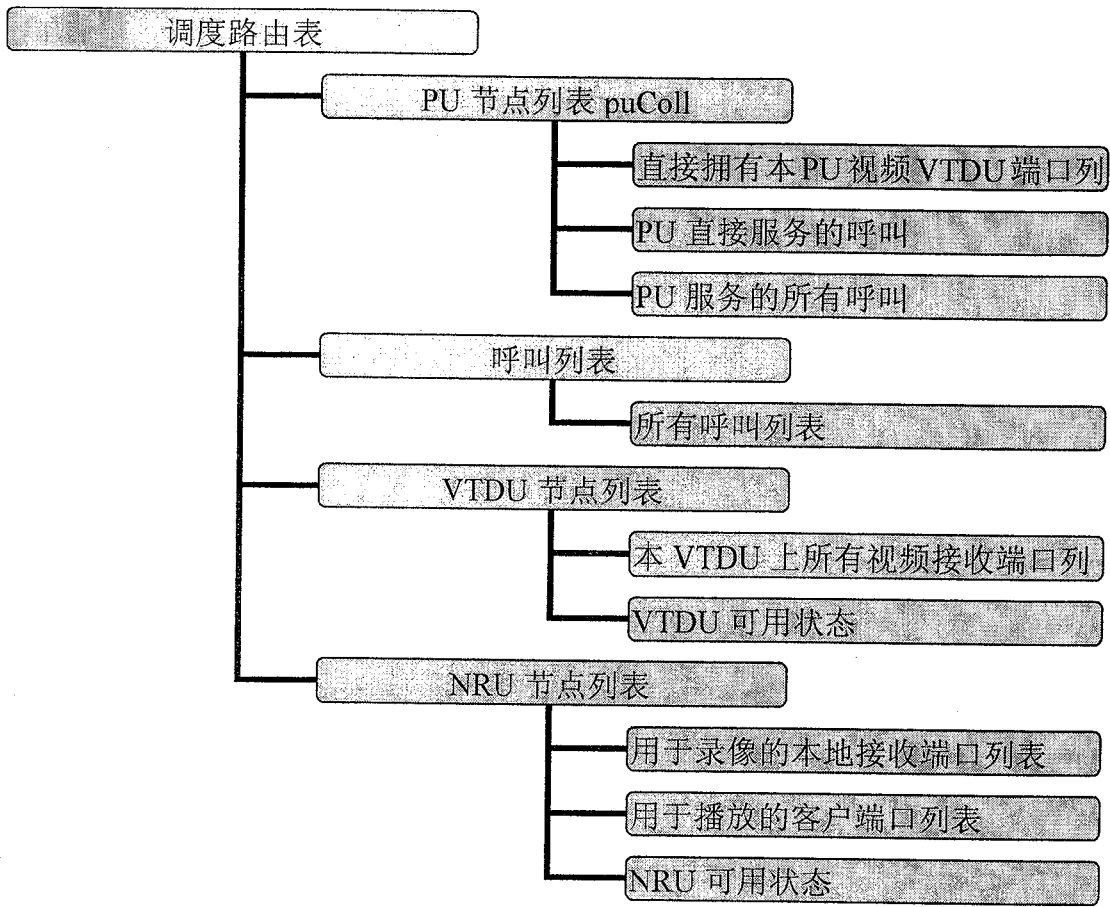


图 3

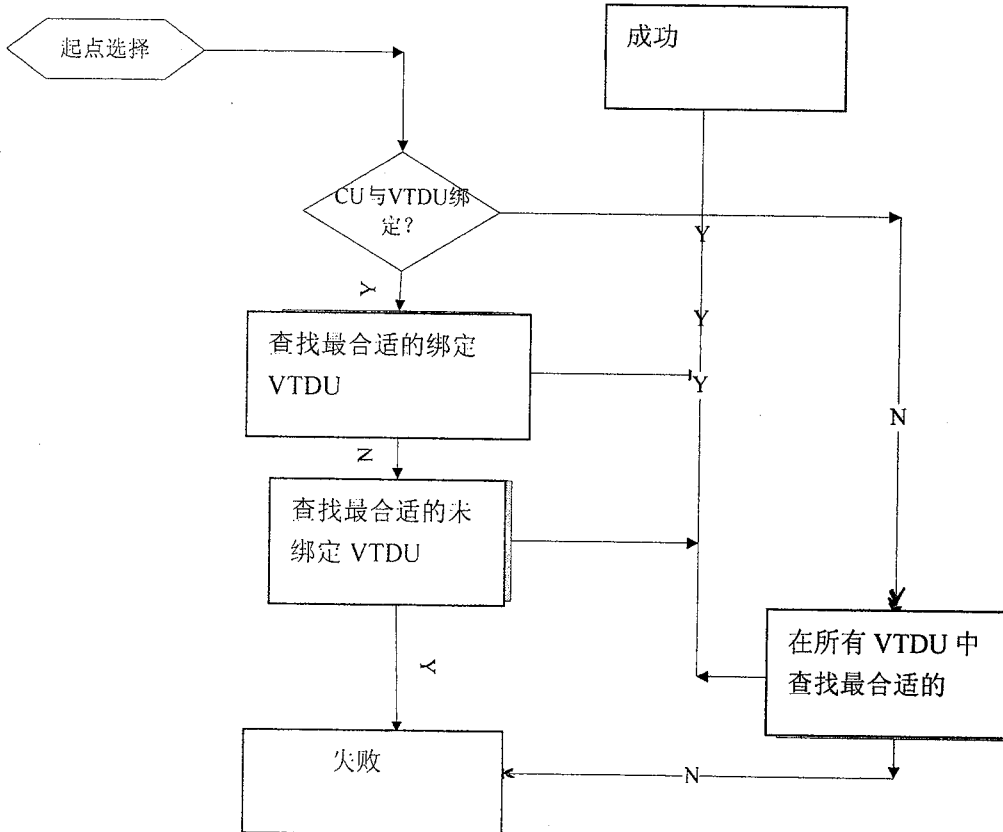


图 4

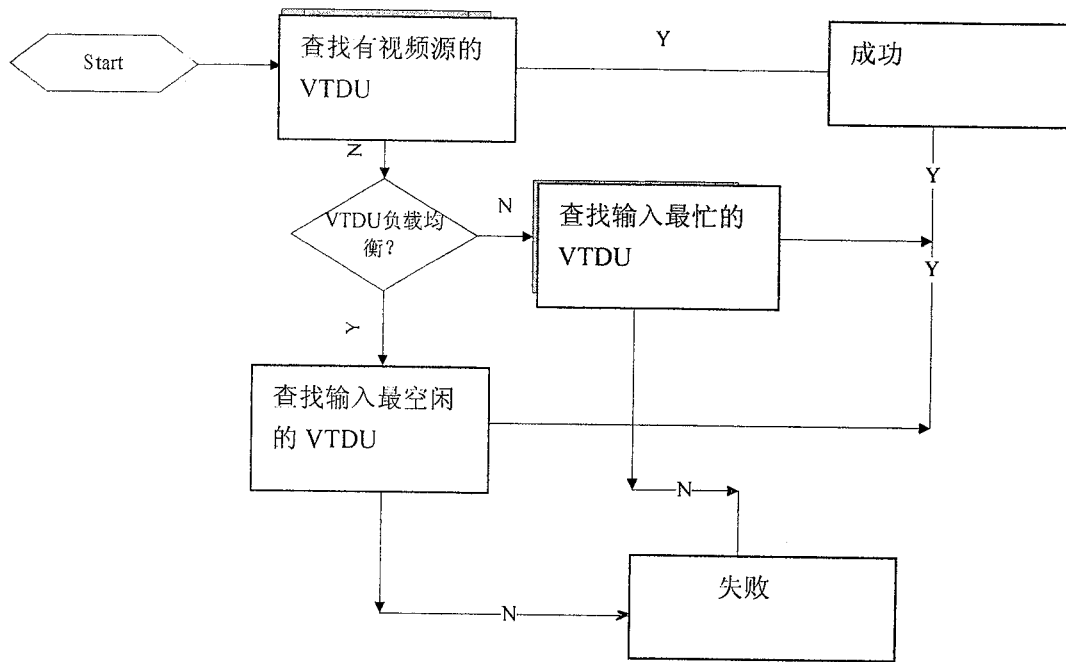


图 5

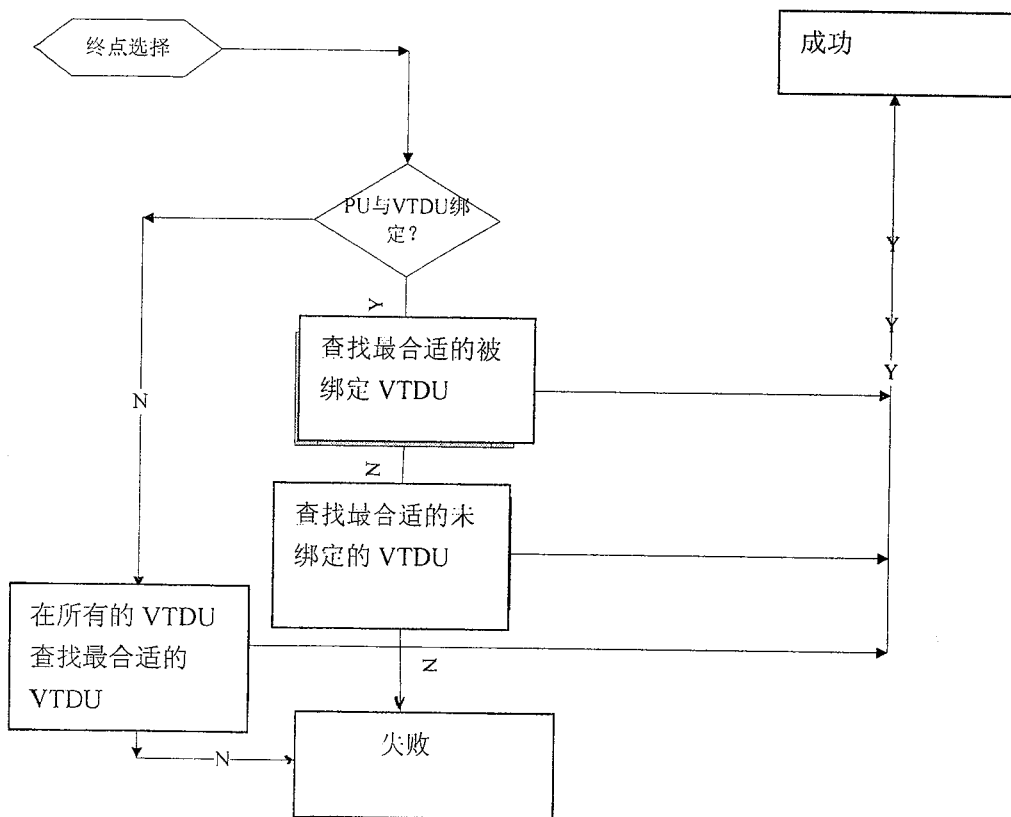


图 6

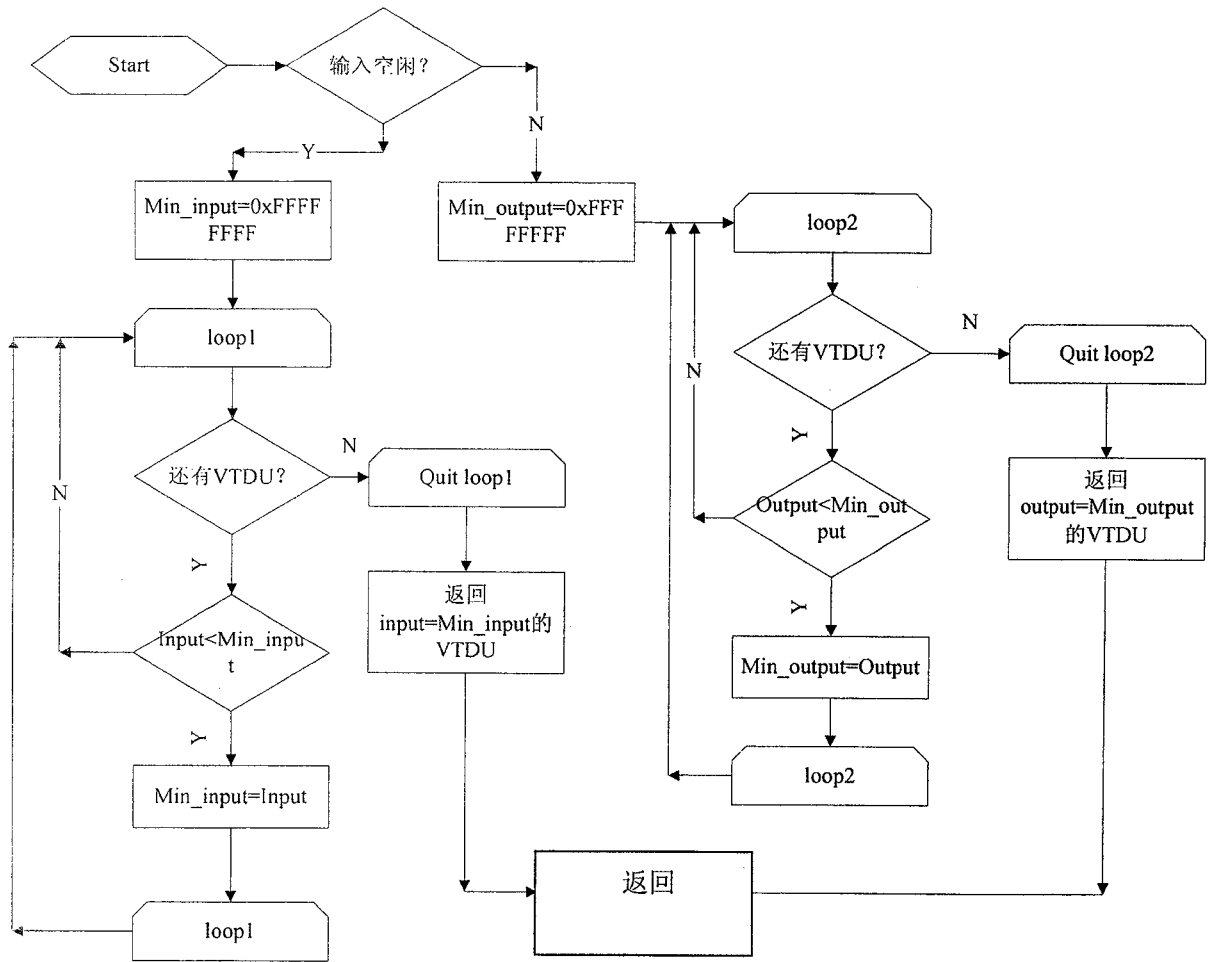


图 7