



[12] 发明专利说明书

专利号 ZL 02829885.3

[45] 授权公告日 2009 年 4 月 8 日

[11] 授权公告号 CN 100477627C

[22] 申请日 2002.11.13 [21] 申请号 02829885.3

[86] 国际申请 PCT/CN2002/000807 2002.11.13

[87] 国际公布 WO2004/045154 英 2004.5.27

[85] 进入国家阶段日期 2005.5.13

[73] 专利权人 英特尔公司

地址 美国加利福尼亚

共同专利权人 英特尔(中国)有限公司

[72] 发明人 何量

[56] 参考文献

US6101472A 2000.8.8

CN1372668A 2002.10.2

US5819220A 1998.10.6

审查员 孙淑蓉

[74] 专利代理机构 北京润平知识产权代理有限公司

代理人 周建秋 王凤桐

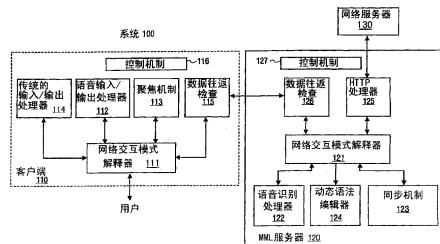
权利要求书 6 页 说明书 38 页 附图 7 页

[54] 发明名称

通过无线网络的多模式网络交互

[57] 摘要

公开了一种系统、设备和方法，用于接收在客户端设备的用户输入，解释所述用户输入以识别对多个网络交互模式的至少其中一个的选择，部分地基于所述用户输入和所述网络交互模式产生相应的客户端请求；并通过网络向服务器发送所述客户端请求。



1. 一种多模式网络交互方法，该方法包括：

接收在客户端设备的用户输入；

解释所述用户输入以确定对多个网络交互模式的至少其中一个的选择；

基于所述用户输入和所述网络交互模式产生相应的客户端请求；以及

通过网络向服务器发送所述客户端请求，

其中，所述解释所述用户输入以确定对多个网络交互模式的至少其中一个的选择包括：当所述用户输入触发谈话按钮时，则确定所选择的网络交互模式为语音交互模式。

2. 如权利要求 1 所要求的方法，还包括：

在确定所选择的网络交互模式为语音交互模式之后，根据所述用户输入确定聚焦的显示元素，所述客户端请求还基于所确定的聚焦的显示元素。

3. 如权利要求 2 所要求的方法，还包括：

在确定聚焦的显示元素之后，发送所确定的聚焦的显示元素的标识符到所述服务器。

4. 如权利要求 2 所要求的方法，其中所聚焦的显示元素是超链接。

5. 如权利要求 2 所要求的方法，其中所聚焦的显示元素是表单中的区域。

6. 如权利要求 1 所要求的方法，还包括：

在确定所选择的网络交互模式为语音交互模式之后，从所述用户输入中提取语音特征，所述客户端请求还基于所提取的语音特征。

7. 如权利要求 6 所要求的方法，还包括：

在提取语音特征之后，向所述服务器发送所提取的语音特征。

8. 如权利要求 1 所要求的方法，还包括：

在通过网络向服务器发送所述客户端请求之前，发送对话消息到所述服务器以初始化与所述服务器的连接。

9. 如权利要求 8 所要求的方法，其中所述对话消息包括所述客户端设备的 IP 地址，所述客户端设备的设备类型，所述用户的语音特征，所述用户使用的语言，以及所述客户端设备请求的缺省的识别精确度。

10. 如权利要求 1 所要求的方法，还包括：

在通过网络向服务器发送所述客户端请求之前，发送传输消息到所述服务器以与所述服务器交换传输参数。

11. 如权利要求 1 所要求的方法，还包括：

当通过所述用户输入触发谈话按钮时，发送聚焦消息到所述服务器以通知聚焦的显示元素的标识符和当前页的 URL。

12. 如权利要求 11 所要求的方法，还包括：

在确定所选择的网络交互模式为语音交互模式之后，发送所提取的语音特征到所述服务器。

13. 如权利要求 1 所要求的方法，还包括：

当未聚焦消息发生时执行与发生未聚焦消息的情况相对应的任务。

14. 如权利要求 1 所要求的方法，在通过网络向服务器发送所述客户端请求之后，所述方法还包括：

发送退出消息到所述服务器以终止与所述服务器的对话。

15. 如权利要求 1 所要求的方法，其中使用多模式标记语言来实现所述方法。

16. 一种多模式网络交互方法，包括：

在服务器通过网络从客户端设备接收客户端请求；

解释所述客户端请求以确定对多个网络交互模式中的至少其中一个的选择，至少一个网络交互模式是语音交互模式；以及

如果选择所述语音交互模式，则

接收聚焦的显示元素的标识符，

基于所聚焦的显示元素，建立语音识别的正确语法，

执行语音识别，和

根据所述语音识别的结果执行特定任务。

17. 如权利要求 16 所要求的方法，其中所聚焦的显示元素是超链接。

18. 如权利要求 16 所要求的方法，其中所聚焦的显示元素是表单中的区域。

19. 如权利要求 16 所要求的方法，还包括：

如果语音识别结果匹配，则通过所述网络发送表示语音与所述聚焦显示元素中的输入选择相匹配的匹配事件到所述客户端设备。

20. 如权利要求 16 所要求的方法，还包括：

如果语音识别结果不匹配，则通过所述网络发送表示语音与所述聚焦显示元素中的输入选择不相匹配的不匹配事件到所述客户端设备。

21. 如权利要求 16 所要求的方法，还包括：

在接收客户端请求之前，从所述客户端设备接收用于与所述客户端设备交换传输参数的传输消息。

22. 一种客户端设备，包括：

用户输入接收器；

解释器，从由所述用户输入接收器接收的用户输入来确定对多个网络交互模式的至少其中一个的选择，至少一个网络交互模式是语音交互模式，并且当用户输入触发谈话按钮时，则确定所选择的网络交互模式为语音交互模式；

客户端请求发生器，基于所述用户输入和所述网络交互模式产生客户端请求，以及通过网络发送所述客户端请求到服务器。

23. 如权利要求 22 所要求的客户端设备，其中所述客户端请求发生器还在确定所选择的网络交互模式为语音交互模式之后，根据所述用户输入确定聚焦的显示元素，所述客户端请求还基于所确定的聚焦的显示元素。

24. 如权利要求 23 所要求的客户端设备，其中所述客户端请求发生器还发送所确定的聚焦的显示元素的标识符到所述服务器。

25. 如权利要求 23 所要求的客户端设备，所述解释器包括网络交互模式解释器，用于接收和解释用户输入以确定网络交互模式。

26. 一种服务器设备，包括：

客户端请求接收器，通过网络从客户端设备接收客户端请求；
解释器，从由所述客户端请求接收器接收的客户端请求来确定对多个网络交互模式的至少其中一个的选择，至少一个网络交互模式是语音交互模式；

语音处理器，如果选择所述语音交互模式，就处理在所述客户端请求中接收的语音，所述语音处理器使用聚焦的显示元素的标识符，并基于所述标识符来获得语音和显示元素之间的同步关系，并基于语音元素和显示元素之间的同步关系来建立正确的语音识别语法，所述语音处理器执行语音识别，并按照所述语音识别的结果执行特定任务。

27. 如权利要求 26 所要求的服务器设备，其中所聚焦的显示元素是超链接。

28. 如权利要求 26 所要求的服务器设备，其中所聚焦的显示元素是表单中的区域。

29. 如权利要求 26 所要求的服务器设备，所述解释器为网络交互模式解释器，用于接收和解释来自所述客户端请求以确定网络交互模式。

30. 一种多模式网络交互系统，包括：

客户端设备，具有用户输入接收器，客户端解释器，用于从由所述用户输入接收器接收的用户输入确定对多个网络交互模式的至少其中一个的选择，至少一个网络交互模式是语音交互模式，以及客户端请求发生器，用于基于所述用户输入和所述网络交互模式产生相应的客户端请求，并通过网络

发送所述客户端请求到服务器；和

服务器，具有通过所述网络从所述客户端设备接收所述客户端请求的客户端请求接收器，用于从由所述客户端请求接收器接收的客户端请求以确定对多个网络交互模式的至少其中一个的选择的服务器端解释器，其中至少一个所述网络交互模式是语音交互模式，以及语音处理器用于如果选择所述语音交互模式就处理在所述客户端请求中接收的语音，所述语音处理器使用聚焦的显示元素的标识符，并基于所述标识符来获得语音和显示元素之间的同步关系，并基于语音元素和显示元素之间的同步关系来建立用于语音识别的正确语法，所述语音处理器执行语音识别，并按照所述语音识别的结果执行特定任务。

31. 如权利要求 30 所要求的系统，其中所述客户端请求发生器还确定聚焦的显示元素，所述客户端请求还基于所确定的聚焦的显示元素。

32. 如权利要求 31 所要求的系统， 其中所述客户端请求发生器还发送所确定的聚焦的显示元素的标识符到所述服务器。

33. 如权利要求 30 所要求的系统， 其中所聚焦的显示元素是超链接。

34. 如权利要求 30 所要求的系统， 其中所聚焦的显示元素是表单中的区域。

通过无线网络的多模式网络交互

技术领域

本发明涉及通过在无线通信设备和互联网应用程序之间的无线网络的网络交互。特别地，本发明涉及通过无线网络的多模式网络交互，所述多模式网络交互使用户以多种方式与互联网应用程序进行交互。

背景技术

用于个人通信需要的无线通信设备日益普遍。这些设备包括例如，移动电话、数字传呼机、“掌上型”电脑、个人信息管理器（PIMS），和其它小的主要是手持的通信和计算设备。无线通信设备在其特征上已经相当成熟，并且现在不但支持基本的点对点通信功能如电话呼叫，而且支持更加高级的通信功能，例如电子邮件、传真接收和发送、互联网的访问和全球网的浏览等等。

通常，传统的无线通信设备具有管理手机的各种功能和与基站的无线电通信连接的软件。管理所有电话功能的软件一般被称为电话堆栈(telephone stack)。管理输出和输入，例如按下键盘和屏幕显示的软件，称为用户界面或人机界面或“MMI”。

美国专利 NO. 6,317,781 公开了一种基于人机界面的标记语言。所述人机界面为无线通信设备的各种无线电通信功能提供了用户界面，所述无线电通信功能包括电话拨号、电话呼应回答、创建消息、发送消息、接收消息和创建配置设置，所述无线电通信功能由众所周知的标记语言例如 HTML 来定义，并通过所述无线通信设备所执行的浏览器程序而访问。所述特征使对互联网和全球网的内容，例如网页的直接访问与所述设备的无线电通信功能

直接结合，并且允许网页内容与其它数据类型无缝结合，因为通过所述用户界面向用户给出的所有数据都通过基于标记语言的网页给出。这种基于人机界面的标记语言使用户能够直接与互联网应用程序交互。

然而，与传统的台式或笔记本电脑不同，无线通信设备具有非常有限的输入能力。台式或笔记本电脑具有基于点击设备的指针，例如电脑鼠标、跟踪球、操纵杆等等，以及整个键盘。这样能够通过点击和拖动滚动条、点击超文本链接以及在例如 HTML 表单的不同表单区域之间键盘切换来实现网页内容导航。相反，无线通信设备具有非常有限的输入能力，一般为上下键，以及一至三个软键盘。因此，即使具有基于人机界面的标记语言，无线通信设备的用户也不能使用传统的技术与互联网应用程序交互。尽管现有技术中存在一些形式的语音识别，但现有技术系统没有实现多模式网络交互，所述多模式网络交互将使用户以多种方式通过无线网络完成网络交互。

附图说明

结合附图将更充分地理解本发明的特征，在附图中：

图 1 表示应用了本发明实施例的网络环境。

图 2 表示根据本发明一个实施例的用于通过无线网络的网络交互的系统 100。

图 3 和图 4 表示对一组超链接或表单的聚焦。

图 5-6 表示所述 MML 事件机制。

图 7 表示系统消息和 MML 事件的基本流程图。

图 8 表示在本发明一个实施例的系统中所使用的 MML 元素块(Element Block) 的细节。

具体实施方式

在以下的详细说明中，为实现本发明的清楚理解而阐述了大量细节。然而，需要本领域普通技术人员理解的是，本发明并不限于这些详细的细节。

本发明的多个实施例通过提供一种通过无线网络的多模式网络(web)交互的系统和方法克服了传统的用于无线通信的人机界面的局限性。本发明的多模式网络交互将使用户以多种方式与互联网应用程序交互，例如包括：

输入：键盘、小键盘、鼠标、指示笔、语音；

输出：纯文本、图形、动态视频、音频、合成语音。

这些模式的每一个都能独立或合并使用。在以下更详细描述的实施例中，本发明使用多模式标记语言（MML）。

在一个实施例中，本发明提供一种用于通过无线网络的网络交互方法。在所述实施例中，客户端系统接收用户输入，解释所述用户输入以确定几个网络交互模式中的至少一个，产生相应的客户端请求并发送所述客户端请求。所述服务器接收并解释所述客户端请求以执行特定的检索工作，并将结果发送到所述客户端系统。

在一个实施例中，通过使用多模式标记语言（MML）实现本发明，所述多模式标记语言（MML）包括 DSR（分布语音识别）机制、聚焦机制、同步机制和控制机制，其中所述聚焦机制用于确定聚焦于哪一个活动显示以及所聚焦的显示元素的标识符。所述同步机制用于获得语音元素和显示元素之间的同步关系以建立用于处理用户的语音输入的相应语音元素的语法。所述控制机制控制客户端机和服务器之间的交互。根据这种设备，所述多模式网络交互流程如下例所示：

用户： 使用超链接指向或点击，提交表单（form）（传统的网络交互）或按下“谈话按钮”并输入语音（语音交互）。

客户端：接收并解释所述用户输入。在传统的网络交互的情况下，所

述客户端向所述服务器发送新页面的请求，或提交表单。在语音交互的情况下，所述客户端确定聚焦于哪一个活动显示元素以及所聚焦的显示元素的标识符、捕获语音、提取语音特征，并将所述语音特征、所聚焦的显示元素的标识符和例如当前页 URL 的其它信息发送到所述服务器。客户端等待相应的服务器应答。

服务器：接收和解释来自所述客户端的请求。在传统的网络交互的情况下，所述服务器从高速缓存或网页服务器中检索新的网页并将所述网页发送至客户端。在语音交互的情况下，所述服务器接收所聚焦的显示元素的标识符，以建立基于显示元素和语音元素的同步性的正确语法。随后执行语音识别。根据语音识别的结果，所述服务器将执行特定的工作并向客户端发送事件或新的网页。然后，所述服务器等待来自所述客户端的新请求。

客户端：下载所述新的网页或处理事件。

这里所描述的本发明的多个实施例提供了使用分布语音识别（DSR）技术来实现多模式网络交互的方法。所述方法使几种交互模式中的每一个都能单独地或合并使用。

作为本发明的另一好处，由于具有所述聚焦机制和同步机制，本发明将使所述语音识别技术可行地使用于所述网页的检索信息上，改善语音识别的精确性，减少语音识别所需的计算资源，并实现实时的语音识别。

作为本发明的另一好处，由于基于多模式标记语言的实现，本发明的方法可以在各成员之间共享，所述多模式标记语言是通过增加语音特征的 XML 的扩展。所述方法可用于帮助互联网服务提供商（ISP）容易地建立用于多模式网络交互的服务器平台。所述方法可用于帮助互联网内容服务提供商（ICP）来容易地创建具有多模式网络交互特征的应用程序。特别地，多模式标记语言（MML）能够以至少两个方案用在所述网络上建立语音应用程序：

多模式应用程序可以通过将 MML 增加到用于语音模型的可视 XML 页面上来设计；

通过使用用于 DTMF 输入的 MML 特征，可以编写在不需要可视显示的情况下如电话的纯语音应用程序。

这允许内容开发者重新使用用于处理用户输入的代码。所述应用程序逻辑在各方案中保持相同：基础的应用程序不需要知道信息是通过语音还是其他输入方法获得的。

现在参照图 1，表示可以应用本发明实施例的网络环境的示意图。如图 1 所示，客户端 10 可以通过互联网 5 从网络服务器 12 中访问文档，特别是通过全球网（“网络”）。如所熟知的，所述全球网是对位于全世界无数通过互联网逻辑联系着的计算机上的有格式的超文本页面的集合。客户端 10 可以是个人电脑或各种移动计算设备 14，例如个人数字助理或无线电话。个人数字助理或 PDA 是通常所知的手持计算机，其可用于存储各种个人信息，包括但不仅限于联系信息、日历信息等等。

这种信息可以从其它计算机系统下载，或可以由 PDA 的指示笔和压敏屏幕输入。PDA 的例子是 3Com 公司的 PalmTM 计算机和微软 CETM 计算机，其各自都可以从多种厂家得到。操作例如无绳电话、双模无绳电话和 PDA 的移动计算设备或者操作便携式膝上型计算机的用户发出访问因特网的控制命令。所述控制命令可以包括数字的编码数据、DTMF 或语音命令。这些控制命令通常被发送至网关 18。所述网关 18 处理来自所述移动计算设备 14 的控制命令（包括执行语音识别）并向所述网络服务器 12 发送请求响应于所述请求，所述网络服务器 12 向所述网关 18 发送文档。然后，所述网关 18 合并来自所述文档的显示内容并向所述客户端 14 发送所述显示内容。

根据用于通过无线网络的网络交互的本发明实施例，所述客户端 14 解释所述用户输入以确定网络交互模式，基于所述交互模式确定结果产生和发

送所述客户端 14 请求；以及多模式标记语言（MML）服务器（网关）18 解释所述客户端 14 请求以执行特定的检索工作。所述网络交互模式可以是传统的输入/输出（例如：键盘、小键盘、鼠标以及指示笔/纯文本、图形和动态视频）或语音输入/音频（合成语音）输出。这些实施例使用户以多种方式浏览所述全球网。特别地，用户可以单独地或同时地通过传统的输入/输出和语音输入/输出与互联网应用程序交互。

在以下部分，我们将根据本发明的一个实施例描述通过无线网络的用于网络交互的系统。我们所给的参考设计是 MML 的一个实现，所述 MML 通过增加语音特征以增强 XHTML 模块，扩展了 XHTML Basic。XHTML Basic 的目的是提供可以在成员之间共享的 XHTML 文档类型。因此， XHTML 基本语言文档可以在最大量的网络客户端中使用，例如移动电话、PDA 和智能电话。这就是基于 XHTML Basic 实现 MML 的理由。

在一个实施例中的 XHTML Basic 模块：

结构模块；文本模块；超文本模块；列表模块；基本表单模块（Basic Forms Module）；基本表格模块（Basic Tables Module）；图像模块；目标模块；元信息模块（Metainformation Module）；链接模块和基本模块（Link Module and Base Module）。

其它 XHTML 模块可以提供更多特征：

脚本模块：支持客户端脚本。

风格模块（Style Module）：支持内联样式表单。

参照图 2，表示根据本发明一个实施例的用于通过无线网络的网络交互的系统 100 的示意图。在图 2 中，仅表示与本发明相关的部分，以免模糊本

发明。如图 2 所示，所述客户端 110 包括：网络交互模式解释器 111、语音输入/输出处理器 112、聚焦机制 113、传统输入/输出处理器 114、数据往返检查 115 和控制机制 116。所述 MML 服务器 120 包括：网络交互模式解释器 121、语音识别处理器 122、同步机制 123、动态语法编辑器 124、HTTP 处理器 125、数据往返检查 126 和控制机制 127。

在所述系统 100 中，在客户端 110 中，网络交互模式解释器 111 接收和解释用户输入以确定网络交互模式。所述网络交互模式解释器 111 还协助所述客户端 110 的内容解释。在传统的网络交互情况下，传统的输入/输出处理器 114 处理用户输入，然后数据往返检查 115 向所述服务器 120 发出新网页或提交表单的请求。在语音交互情况下，语音输入/输出处理器 112 捕获并提取语音特征，聚焦机制 113 确定聚焦于哪个活动显示元素上以及所聚焦的显示元素的标识符。然后数据往返检查 115 将所提取的语音特征、所聚焦的显示元素的标识符和例如当前页的 URL 的其它信息发送至所述 MML 服务器。在 MML 服务器 120 中：网络交互模式解释器 121 接收和解释来自所述客户端 110 的请求以确定所述网络交互模式。所述网络交互模式解释器 121 还帮助所述服务器 120 中的内容解释。在传统的网络交互情况下，HTTP 处理器 125 从高速缓存或网络服务器 130 中检索新的网页或表单。在语音交互情况下，同步机制 123 基于所接收的标识符获得在语音元素和显示元素之间的同步关系，动态语法编辑器 124 基于在语音元素和显示元素之间的同步关系编写正确的语法。语音识别处理器 122 基于由动态语法编辑器 124 编写的正确语法完成语音识别。按照所述识别结果，HTTP 处理器 125 从高速缓存或网页服务器 130 中检索新的网页或表单。然后，数据往返检查 126 基于检索结果向所述客户端 110 发送应答。所述控制机制 116 和 127 用于控制所述客户端和服务器之间的交互。

以下是根据实施例对使用了具有聚焦机制、同步机制和控制机制的

MML 的本发明一个实施例的详细说明。

聚焦机制

在多模式网络交互中，除传统的输入法之外，语音输入可以成为一种新的输入源。当使用语音交互时，在客户端检测语音并提取语音特征，并且语音识别在所述服务器中完成。我们通常注意到，用户一般使用以下类型的常规显示元素执行输入：

超链接： 用户可以选择稳定的超链接。

表单： 用户可以观看和/或修改包含例如股票价格、货币兑换、航班等信息的电子表单。

考虑到当前语音识别技术的局限性，在本发明的多模式网络交互中，提供聚焦机制来把用户注意力集中到用户执行语音输入的活动的显示元素上。显示元素是通过高亮度显示或者另外特别着色的显示元素来聚焦的，用户的语音输入将应用于所述显示元素上。当将所聚焦的显示元素的标识符（ID）发送到所述服务器时，所述服务器可以基于显示元素和语音元素之间相应的关系执行语音识别。因此，代替传统的具有很大词汇的录入，一个实施例的所需词汇数据库基于超链接、电子表单和其它用户将在其上执行语音输入的显示元素。同时，在服务器端，可以基于所述显示元素和语音元素的同步性而动态地建立正确的语法。因此，将提高语音识别的精确性，减少语音识别的计算负荷，以及真正实现实时的语音识别。

图 3 和图 4 的 MMI 可以帮助理解对实施例中的活动显示元素的聚焦处理。图 3 表示在一组超链接上的聚焦，以及图 4 表示在表单上的聚焦。

在传统的 XHTML 规范中，不允许在表单之外增加按钮。由于我们的方法不改变所述 XHTML 规范，在实施例中可以采用“可编程硬件按钮”来聚焦一组超链接。采用具有“对我说（Talk To Me）”标题的软件按钮来聚焦所

述电子表单显示元素。对本领域普通技术人员来说显而易见的是，其它输入方式同样可以用于特殊显示元素的聚焦。

当在显示屏上显示“界面卡（card）”或文档的网页时，没有开始聚焦显示元素。使用所述“可编程硬件按钮”或“对我说按钮”，用户可以通过语音方法来执行网络交互。如果用户点击所述“可编程硬件按钮”或“对我说按钮”，按钮所属的显示元素被聚焦。然后，可能的情况如下：

用户谈话

一旦用户引起在特定显示元素上的聚焦，来自用户的语音被接收和存储或者与同所聚焦的显示元素相联系的可得的输入选择相比较。如果所存储的语音与特定的输入选择足够接近，则产生“匹配”事件并显示新的界面卡或网页。

所述新的界面卡或网页符合所匹配的输入选择。如果所存储的语音不能与特定的输入选择匹配，则产生“不匹配”事件，显示音频或文字提示，显示元素仍被聚焦。

用户也可以使用传统的方式引起对特定显示元素的聚焦，例如指向输入区，例如在表单中的框符。在这种情况下，当前聚焦的显示元素变为不聚焦，因为选择了不同的显示元素。

用户还可以指向超文本链接，该超文本链接一起显示新的界面卡或网页。如果用户指向另一个“对我说按钮”，先前的显示元素变为不聚焦，而上一次点击所属的显示元素变成聚焦。

如果用户在长于预定配置长度的时间中不进行任何操作，所聚焦的显示元素可以变为不聚焦。

同步机制

当用户希望通过语音方法在显示元素上输入时，相应的语音元素的语法将加载到所述服务器以处理用户的语音输入。因此，用于所述语音元素和显示元素的同步或构造设置是必需的。以下是实现所述结果的两个实施例。

一种基本的语音元素具有包括用于在网页上同一时间语音交互的所有入口文字的语法。

一个基本的语音元素必须有且仅有如下一个相应的如下显示元素：

一组超链接

一个表单

一个可识别的单个或成组的显示元素。

因此，需要执行绑定功能将语音元素绑定到相应的显示元素上。在一个实施例中，“绑定”属性由< mml: link>、< mml: sform>和< mml: input>定义。所述绑定属性包含一个显示元素和相应的语音元素对的信息。

下面给出了用于超链接类型显示元素的所述绑定的源代码示例。

```
<mml: card >
    <a id="stock" > stock </a >
    <a id="flight" > flight </a >
    <a id="weather" > weather </a >
    <mml : speech >
        <mml : recog >
            <mml : group >
                <mml grammar src="grammar. gram" />
                <mml : link value="#stock-gram" bind="stock"/>
                <mml : link value="#flight-gram" bind="flight"/>
                <mml : link value="#weather-gram" bind="weather"/>
```

```
</mml : group >  
</mml : recog >  
</mml : speech >  
</mml : card >
```

下面给出了用于在电子表单中绑定的源代码示例，例如飞行航班信息表单。

```
<mml : card title="flight inquiry" >  
  <script language="javascript" >  
    function talk1 ()  
    {  
      var sr= new DSR. FrontEnd ;  
      sr. start ("form-flight") ;  
    }  
  </script >  
  <p> flightquery : </p>  
  
< form id="form-flight" action="flightquery. asp" method="post" >  
  <p> date : < input type="text" id ="date1" name="date01" />  
  company (optional) : < input type="text" id ="company1" name="company01" />  
  </p>  
  <p> startfrom : < input type="text" id ="start-from" name="start" />  
    arrivingat : < input type="text" id ="arriving-at" name="end" />  
  </p>  
  <p>< input type="submit" value="submit" />< input type="Reset"  
  value="Reset" />  
    < input type="button" value="Talk To Me" onclick="talk1()" />  
  </p>
```

```
</form>

<mml:speech>
  <mml:recog>

    <mml:sform id="sform-flight" bind="form-flight">
      <mml:grammar src="flight-query.gram"/>
      <mml:input id="sdate" value="#date" bind="date1"/>
      <mml:input id="scompany" value="#company" bind="company1"/>
      <mml:input id="sstart" value="#start" bind="start-from"/>
      <mml:input id="send" value="#end" bind="arriving-at"/>
      <mml:onevent type="match">
        <mml:do target="flight-prompt" type="activation"/>
      </mml:onevent>
    </mml:sform>
    <mml:onevent type="nomatch">
      <mml:do target="promptl" type="activation"/>
    </mml:onevent>
  </mml:recog>
</mml:speech>
</mml:card>
```

客户端-服务器控制机制

当执行多模式交互时，为了向用户代理和服务器指示已经发生一些动作，需要较好地定义在客户端或服务器端产生的系统消息或者在客户端或服务器端产生的其它事件。

在本发明的实施例中，设计客户端-服务器控制机制以提供用于系统消息和 MML 事件的定义的机制，所述系统消息和 MML 事件是控制所述客户端和服务器之间的交互所需要的。

表 1 包括系统消息和 MML 事件的典型设置。

	系 统 消 息	事 件	客户 端与服 务器 之 间 的 通 信
错误 (服务器)	√		√
传输 (服务器)	√		√
传输 (客户端)	√		√
就绪 (服务器)	√		√
对话 (客户端)	√		√
退出 (客户端)	√		√
聚焦* (客户 端)	√		√
未聚焦* (客户 端)	√		
匹配 (服务器)		√	√
不匹配 (服务 器)		√	√
加载 (客户端)		√	
卸载 (客户端)		√	

表 1 控制信息表

系统消息：

所述系统消息用于客户端和服务器交换系统信息。一些类型的系统消息由所述客户端触发并发送给所述服务器。其它的由所述服务器触发并发送给所述客户端。

在一个实施例中，在客户端触发的系统消息包括如下：

<1> 对话消息

当所述客户端初始化到服务器的连接时发送所述对话消息。在发送所述对话消息之后，应接收来自所述服务器的就绪消息或错误消息。以下是所述

对话消息的例子：

```

< message type="session" >
  < ip >< /ip >          <!-- 请求客户端的 IP 地址-->
  < type >< /type >        <!-- 客户端的设备类型-->
  < voice >< /voice >      <!-- 用户的语音特征-->
    <!-- 如男人、女人、老年男子、老年妇女、孩子-->
  < language >< /language >   <!-- 用户所用语言-->
  < accuracy >< /accuracy >  <!-- 客户端请求的缺省识别精确度-->
< /message >

```

< 2 > 传输消息

在客户端与所述服务器建立对话之后发送所述传输消息（客户端）。在发送所述传输消息（客户端）之后，应从服务器接收传输消息（服务器）或错误消息。以下是传输消息的例子：

```

< message type="transmission" >
  < session >< /session >          <!-- 本次对话的标识符-->
  < crc >  < /crc >            <!-- 客户端要求的 crc 信息-->
  < QoS >< /QoS >           <!-- 客户端要求的 QoS 信息 -->
  < bandwith >< /bandwith >     <!-- 客户端要求的带宽-->
< /message >

```

< 3 > 聚焦消息

聚焦和未聚焦消息是特定的客户端系统消息。

当用户点击、或按下、或触发所述“谈话按钮”（这里“谈话按钮”意味着“硬件可编程按钮”和“对我说按钮”）时聚焦发生。当聚焦发生时，所述客户端将执行以下任务：

- a. 打开话筒和执行前端检测
- b. 当捕获到实际语音的起点时，执行前端语音处理。相应的聚焦显示元素的标识符和其它基本信息（例如当前页的 URL）与所述第一语音特性数据包一起发送到服务器。
- c. 当所述第一语音特性数据包到达所述服务器时，相应的语法将加载到识别器中并执行语音识别。

以下是发送到所述服务器的聚焦消息的例子：

```
< message type="OnFocus" >
< session >< session >      <!--本次对话的标识符-->
< id >< /id >            <!--聚焦显示元素的标识符-->
< url >< url >          <!--当前客户端加载的文档的 URL-->
< /message >
```

建议聚焦消息随语音特性一起发送而不是单独发送。其理由是为了在这些情况中优化和减少不必要的通信和服务器负载：

“当用户在开始一次对话之前在同一界面卡或同一页面上切换和点击两个不同的“谈话按钮”时，所述客户端将向服务器发送不必要的聚焦消息并且将使服务器不必要的创建语法。”

但是软件供应商可以选择实现单独发送聚焦消息。

<4> 未聚焦消息

当未聚焦发生时，所述客户端将执行关闭话筒的任务。在以下情况中，未聚焦发生：

- a. 用户点击、按下或另外触发所聚焦的显示元素的“谈话按钮”。

b. 用户在例如表单框符等的输入区内使用点到点的传统方式。

在以下情况中，未聚焦不发生，

a. 当在界面卡或页面中存在聚焦的显示元素时，用户点击、按下或另外触发所述未聚焦的显示元素的“谈话按钮”。

<5> 退出消息

当所述客户端退出对话时发送所述退出消息。以下举例：

```
<message type="exit">
  <session></session>      <!--本次对话的标识符-->
</message>
```

服务器触发的系统消息

<1> 就绪消息

当客户端首先发送对话消息且服务器准备好工作时，由所述服务器发送所述就绪消息。以下举例：

```
<message type="ready">
  <session></session>      <!-- 由服务器随后创建的对话的标识符-->
                           <!-- 已经接收对话消息-->
  <ip></ip>                <!--响应服务器的 IP 地址-->
  <voice>
    <support>T</support>  <!-- T 或者 F, 服务器支持或者不支持在对话消息中
                           客户端请求的语音特征-->
    <server></server>    <!-- 服务器正在使用的语音特征-->
  </voice>
  <language>
```

< support > T < /support > <!-- T 或者 F, 服务器支持或者不支持客户端在对话消息中所请求的语言-->

< server >< /server > <!-- 服务器正在使用的语言-->
< /language >
< accuracy >
< support > T < /support > <!-- T 或者 F, 服务器支持或者不支持客户端在对话消息中请求的缺省识别精确度-->
< server >< /server > <!-- 服务器正在使用的缺省识别精确度-->
< /accuracy >
< /message >

< 2 > 传输消息

当客户端首先发送传输消息或网络状态变化时，服务器发送所述传输消息。所述消息用于通知客户端其应使用的传输参数。以下举例：

```
< message type="transmission" >
    < session >      < /session >   <!-- 本次对话的标识符-->
    < crc >           < /crc >       <!-- crc 信息-->
    < QoS >          < /QoS >       <!-- QoS 信息-->
    < bandwidth >    < /bandwidth >  <!--客户端应使用的带宽-->
< /message >
```

< 3 > 错误消息

所述错误信息由所述服务器发送。如果服务器在处理客户端请求时产生一些错误，所述服务器将向所述客户端发送错误信息。以下举例：

```
< message type="error" >
```

```
< session >    < /session >          <!--本次对话的标识符-->
< errorcode > 500   < /errorcode >    <!--错误的编码数量-->
< errorinfo >    < /errorinfo >     <!--错误的文本信息-->
< /message >
```

MML 事件

所述 MML 事件的目的是为处理不同的处理事件提供灵活的界面结构。

MML 事件可以根据事件源分类为客户端产生的事件和服务器产生的事件。

并且所述事件可能需要在所述客户端和服务器之间通信。

在所述 MML 定义中，事件处理指令的元素是<mml: onevent>。有四种类型的事件：

由服务器端触发的事件

<1> 匹配事件

当语音处理结果匹配时，如果页面开发者将所述处理指令添加到所述 MML 页面的“不匹配”事件的处理程序中，

```
< mml: card >
< form id="form01" action="other.mml" method="post" >
    < input id="text1" type="text" name="add" value="上海" />
< /form >
...
< mml: speech >
    < mml: prompt id="promptServer" type="tts" >
        The place you want to go < mml: getvalue from="stext1" at="server" />
    < /mml: prompt >
    < mml: recog >
        < mml: sform id="sform01" bind="form01" >
```

```
< mml: grammar src="Add. gram"/>
< mml: input id="stext1" value="#add" bind="text1"/>
< mml: onevent type="match">
    < mml: do target="promptServer" type="activation"/>
</mml: onevent>
</mml: sform>
</mml: recog>
</mml: speech>
</mml: card>
```

所述事件发送至所述客户端如下例：

```
< event type="match">
    < do target="promptServer">
        < input id="stext1" value="place"/>      <!--根据识别结果-->
</event>
```

如果页面开发者没有处理所述“匹配”，则不向客户端发送事件。

<2> 不匹配事件

当语音处理结果不匹配时，如果所述页面开发者将所述处理指令添加到所述 MML 页面的“不匹配”事件的处理程序中，

```
< mml: onevent type="nomatch">
    < mml: do target="prompt1" type="activate"/>
</mml: onevent>
```

所述事件发送至所述客户端如下例：

```
<event type="nomatch">
  <do target ="prompt1"/>      <!-- 由页面开发者指定-->
</event>
```

如果页面开发者没有处理所述“匹配”，则事件发送至所述客户端，如下：

```
<event type="nomatch"/>
```

由客户触发的事件

<1> 加载事件

当某个显示元素加载时所述“加载”事件发生。所述事件类型仅在向所述“客户端”发送所述触发器属性时有效。页面开发者可以将所述处理指令添加到所述 MML 页面的“加载”事件的处理程序中：

```
<mml:onevent type="onload">
  <mml:do target="prompt1" type="activate"/>
</mml:onevent>
```

“加载”事件不需要发送至所述服务器。

<2> 卸载事件

当某个显示元素卸载时所述“卸载”事件发生。所述事件类型仅在向所述“客户端”发送触发器属性时有效。页面开发者可以将所述处理指令添加到所述 MML 页面的“加载”事件的处理程序中。

```
<mml:onevent type="unload">
```

```
<mml:do target="promptl" type="activate"/>  
</mml:onevent>
```

“加载”事件不需要发送给所述服务器。

MML 事件一致性

所述一个实施例的 MML 事件机制是传统的 XML 事件机制的扩展。如图 5 所示，在所述传统的事件处理中有两个阶段：“捕获”和“起泡”（见 XML 事件一致性）。

为简化所述事件机制、提高效率和更简易地实施，我们提出了一个实施例的 MML 简易事件机制。如图 6 所示，在所述简易事件机制中，既不需要“捕获”也不需要“起泡”阶段。在所述实施例的 MML 事件机制中，观察器节点必须是所述事件处理程序<mml:onevent>的父节点。由一个节点触发的事件是只能由子<mml:onevent>事件处理节点处理的。其它<mml:onevent>节点将不中止所述事件。并且，忽略所述<mml:onevent>的阶段属性。

图 5 和 6 表示两个事件机制。虚线父节点（图 5 中的节点 510 和图 6 中的节点 610）将中止所述事件。虚线节点的子节点<mml:onevent>（图 5 中的节点 520 和图 6 中的节点 620）将有机会去处理所述确定事件。

在所述实施例中的 MML 事件具有与主语言（XHTML）统一的事件界面，但是独立于所述主语言的传统事件。页面开发者通过加入如观察器节点或目标节点的子节点的<mml:onevent>标记在所述 MML 网页中写入事件。

图 7 表示在本发明一个实施例中使用的系统消息和 MML 事件的处理过程的基本流程图。所述过程可以划分为以下部分，所述每个部分都执行所列步骤，如图 7 所示。

1) 连接：

步骤 1：对话消息从客户端发送到服务器

步骤 2：就绪消息从服务器发送到客户端

步骤 3：传输消息（传输参数）从客户端发送到服务器

步骤 4：传输消息（传输参数）从服务器发送到客户端

当在上述四个步骤中发生不匹配时，将从所述服务器向所述客户端发送错误信息。

2) 语音交互：

步骤 1：特征流与聚焦消息从所述客户端发送到所述服务器

步骤 2：将发生几个情况：

结果匹配：

如果所述实施包括在所述 MML 网页中任意的事件处理，所述事件将发送给客户端。

如果链接到新文件，所述新文档将发送给客户端。

如果链接到相同文档中的新界面卡或页面，所述事件与所述界面卡或页标识符将发送给客户端。

结果不匹配：

如果所述实施包括在所述 MML 网页中的任意事件处理，所述不匹配事件与事件处理信息将发送给客户端。

如果所述实施不包括在所述 MML 网页中的任意事件处理，所述不匹配事件与空的信息将发送给客户端。

3) 传统的交互

URL 请求

新文档将发送给客户端

4) 退出对话：

如果客户端退出，所述退出消息将发给服务器

如上所述，本发明的多种实施例提供了由 MML 实现的聚焦机制、同步机制和控制机制。MML 通过添加语音特性处理扩展了所述 XHTML 基本语言。图 8 表示在本发明一个实施例中使用的 MML 元素块的细节。当由所述多模式服务器接收到内容文档时，一部分 MML 元素块将发送给客户端。在图 8 的虚线 810 内表示发送给所述客户端的所述 MML 元素块的设置。所述整个文档将保持在所述多模式服务器中。

以下是对每个 MML 元素的详细说明。

元素	属性	最小内容模式
Html		(head, (body (mml:card+)))
mml:card	id(ID), title(CDATA), style(CDATA)	(mml:onevent *, (Heading Block List)*, mml:speech?)
mml:speech	id(ID)	(mml:prompt *, mml:recog ?)
mml:recog	id(ID)	(mml:group ?, mml:sform*, mml:onevent*)
mml:group	id(ID), mode(speech dtmf), accuracy(CDATA)	(mml:grammar, mml:link+, mml:onevent*)
mml:link	id(ID), value(CDATA), bind(IDREF)	EMPTY
mml:sform	id(ID), mode(speech dtmf), accuracy(CDATA), bind(IDREF)	(mml:grammar, mml:input+, mml:onevent*)
mml:input	id(ID), value(CDATA), bind(IDREF)	EMPTY
mml:grammar	id(ID), src(CDATA),	PCDATA

mml:prompt	<code>id(ID), type(text tts recorded), src(CDATA), loop(once loop), interval(CDATA)</code>	(PCDATA mml:getvalue)*
mml:onevent	<code>id(ID), type(match nomatch onload unload), phase(default capture), propagate(continue stop), defaultaction(perform cancel)</code>	(mml:do)*
mml:do	<code>id(ID), target(IDREF), href(CDATA), action(activate reset)</code>	EMPTY
Mml:getvalue	<code>id(ID), from(IDREF), at(client server)</code>	EMPTY

仍然参照图 8, 以下更详细地描述了在一个实施例中的每个 MML 元素。

作为举例, MML 元素使用以“mml:”前缀为标识的命名空间。

所述<card>元素:

元素	属性	最小内容模式
mml:card	<code>id(ID), title(CDATA), style(CDATA)</code>	(mml:onevent *, (Heading Block List)*, mml:speech?)

其功能为将整个文档划分为一些界面卡或页面（段）。所述客户端装置将每次显示一个界面卡。这是对小型的显示设备和无线传输的最优化。多个界面卡元素可能出现在一个文档中。每个界面卡元素代表独立的显示或与所述用户的交互。

所述`<mml: card>`是且仅是一个与所述内容显示和文档结构有关系的 MML 的元素。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述任意的 *title* 属性规定了当加载和显示所关联界面卡时将显示在所述用户代理的标题栏上的字符串。

所述任意的 *style* 属性规定了所述 XHTML 内联样式。所述样式的作用范围是整个界面卡。但是这可能由一些 XHTML 子元素覆盖，所述 XHTML 子元素可以定义它们自己的内联（inline）样式。

所述`<speech>`元素：

元素	属性	最小内容模式
<code>mml:speech</code>	<code>id(ID)</code>	(<code>mml:prompt*</code> , <code>mml:recog?</code>)

所述`<mml: speech>`元素是整个语音相关元素的容器。所述`<mml: speech>`的子元素可以是`<mml: recog>`和/或`<mml: prompt>`。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述`<recog>`元素：

元素	属性	最小内容模式
<code>mml:recog</code>	<code>id(ID)</code>	(<code>mml:group ?</code> , <code>mml:sform*</code> , <code>mml:coevent*</code>)

所述< mml: recog>元素是语音识别元素的容器。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述<group>元素：

元素	属性	最小内容模式
mml:group	<i>id</i> (ID), <i>mode</i> (speech dtmf), <i>accuracy</i> (CDATA)	(mml:grammar, mml:link+, mml:coevent*)

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述任意的 *mode* 属性规定了所述语音识别模式。支持两个模式：

1. “speech”（缺省值）

“speech” 模式是缺省的语音识别模式。

2. “dtmf”

“dtmf”模式用于接收电话的 dtmf 信号。(这个模式将支持传统的电话)。

所述任意的 *accuracy* 属性规定了所述网页开发者所能接收的语音识别的最低精确度。支持以下风格：

1. “accept”（缺省值）

语音识别器设置所述识别输出数值是否可以接受。

2. “xx”（例如 “60”）

如果从识别器接收的输出数值等于或大于 “xx” %，所述识别结果将认为是“匹配”并将触发所述“匹配”事件。否则，所述结果将认为是“不匹配”。将触发所述“不匹配”事件。

所述< link>元素：

元素	属性	最小内容模式
mml:link	id(ID),value(CDATA), bind(IDREF)	EMPTY

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所要求的 *value* 属性规定了所述< mml: link>元素对应于所述语法的哪一部分。

所要求的 *bind* 属性规定了将与哪个 XHTML 超链接（例如<a>）绑定。

所述<sform>元素：

元素	属性	最小内容模式
mml:sform	id(ID),mode(speech dtmf), accuracy(CDATA), bind(IDREF)	(mml:grammar, mml:input+, mml:onevent*)

所述< mml: sform >元素起语音输入形式的作用。所述元素应当与 XHTML <form>元素绑定。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述任意的 *mode* 属性规定了所述语音识别模式。支持两个模式：

1. “speech” (缺省值)

“speech” 模式是缺省的语音识别模式。

2. “dtmf”

“dtmf” 模式用于接收电话 dtmf 信号。(这个模式支持传统的电话)。

所述任意的 *accuracy* 属性规定了页面开发者所能接受的语音识别的最低精确度。支持以下风格：

3. “accept” (缺省值)

语音识别器设置所述识别输出数值是否可以接受。

4. “xx” (例如“60”)

如果从识别器接收的输出数值等于或大于“xx”%，所述识别结果将被认为是“匹配”并将触发“匹配”事件。否则，所述结果将被认为是“不匹配”。将触发“不匹配”事件。

所述<input>元素：

元素	属性	最小内容模式
mml :input	id(ID), value(CDATA), bind(IDREF)	EMPTY

所述< mml:input >元素起语音输入数据占位符的作用。所述< mml:input >元素应当与 XHTML <input>绑定。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述任意的 *value* 属性规定了语音识别结果的哪个部分应当分配给绑定的 XHTML <input> 标记。如果没有设置这个属性，整个语音识别结果将分配给绑定的 XHTML <input> 标记。

所要求的 *bind* 属性规定了将与<form>中的哪个 XHTML <input> 绑定。

所述<grammar>元素：

元素	属性	最小内容模式
mml :grammar	id(ID), src(CDATA)	PCDATA

所述< mml :grammar > 规定了用于语音识别的语法。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述任意的 *src* 属性规定了所述语法文档的 URL。

如果没有设置这个属性，所述语法内容应当在< mml :grammar>内容中。

所述<prompt>元素：

元素	属性	最小内容模式
mml:prompt	id(ID), type(text tts recorded), src(CDATA), loop(once loop), interval(CDATA)	(PCDATA mml:getvalue)*

所述< mml:prompt>规定了提示消息。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述任意的 *type* 属性规定了所述提示类型。现在在一个实施例中支持三种类型：

1. “tts” (缺省值)

“tts” 规定了所述语音输出是合成的语音。

2. “recorded”

“recorded” 规定了所述语音输出是预先录制的音频。

3. “text”

“text” 规定了用户代理将在消息框符中输出内容。

如果这个属性设置为 “text”，那么客户端的用户代理将忽略所述 “loop” 和 “interval” 属性。

如果客户端的用户代理没有 TTS 引擎，则所述客户端的用户代理可以覆盖从 “tts” 到 “text” 的这种 “type” 属性。

所述任意的 *src* 属性规定了所述提示输出文档的 URL。

如果没有设置这个属性，所述提示内容应当在< mml :prompt>的内容中。

所述任意的 *loop* 属性规定了所述语音输出将进行多少次。在一个实施例中支持两种模式：

1. “once” (缺省值)

“once” 表示没有循环。

2. “loop”

“loop” 表示所述语音输出将被循环地播放，直到有效的范围改变。

所述任意的 *interval* 属性规定了在语音输出的两个循环之间的间隔时间。仅当所述 *loop* 属性设置为 “loop” 时需要设置它。

格式：

“xxx” (例如 “5000”)

用户代理将在语音输出的两个循环之间等待 “xxx” 毫秒。

所述<onevent>元素：

元素	属性	最小内容模式
mml:onevent	id(ID), type(match nomatch onload unload), trigger(client server), phase(default capture), propagate(continue stop), defaultaction (perform cancel)	(mml:do)*

所述< mml : onevent >元素用于中止某些事件。

所述用户代理（客户端和服务器二者）必须忽略任何规定了不响应于用于即时封装（enclosing）的元素的合法事件的类型的< mml : onevent >元素。例如：所述服务器必须忽略在< mml : sform >元素中的< mml : onevent type = “onload” >。

所述 *type* 属性指示事件的名称。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所需要的 *type* 属性规定了将要处理的事件类型。

在一个实施例中支持以下事件类型：

1. “匹配”

当接受所述语音识别的结果时“匹配”事件发生。

仅当触发器属性设置为“服务器”时，这个事件类型才有效。

2. “不匹配”

当不能接受语音识别的结果时“不匹配”事件发生。仅当触发器属性设置为“服务器”时，这个事件类型才有效。

3. “加载”

当加载特定显示元素时“加载”事件发生。仅当触发器属性设置为“客户端”时，这个事件类型才有效。

4. “卸载”

当卸载特定显示元素时“卸载”事件发生。仅当触发器属性设置为“客户端”时，这个事件类型才有效。

“匹配”和“不匹配”事件类型仅用于语音相关的元素。

“加载”和“卸载”事件类型仅用于显示元素。

所需要的 *trigger* 属性规定了在客户端或服务器端所期望发生的事件。

1. “客户端”(缺省值)

所述事件应发生在客户端。

2. “服务器”

所述事件期望发生在服务器端。

所述任意的 *phase* 属性规定了由所期望的事件激发所述< mml :onevent > 的时间。如果用户代理（包括客户端和服务器）支持 MML 简易内容事件一致性，将忽略这个属性。

1. “缺省”(缺省值)

< mml :onevent > 将在起泡阶段对目标元素中止事件。

2. “捕获”

< mml :onevent > 将在捕获阶段中止事件。

所述任意的 *propagate* 属性规定了所述中止事件是否应当继续执行 (XML 事件一致性)。如果所述用户代理（包括客户端和服务器）支持 MML 简易内容事件一致性，将忽略这个属性。

在一个实施例中支持下列模式：

1. “继续”(缺省值)

所述中止事件将继续执行。

2. “停止”

所述中止事件将停止执行。

所述任意的 *defaultaction* 属性规定在所述事件< mml :onevent > 之后是否应当执行对所述事件(如果有的话)的缺省作用。

例如：

在所述< mml : sform > 上的“匹配”事件的缺省作用将提交表单。< mml : sform > 上的“不匹配”事件的缺省作用将重新设置为相应的<form>并给出“不匹配”消息。

在实施例中支持以下模式：

1. “执行”（缺省值）

执行缺省动作（除非被其它方式取消，比如脚本或另一个`<mml:onevent>`）。

2. “取消”

取消缺省动作。

所述`<do>`元素：

元素	属性	最小内容模式
<code>Mml:do</code>	<code>Id(ID),target(IDREF), href(CDATA), action (activate reset)</code>	<code>EMPTY</code>

所述`<mml:do>`元素总是`<mml:onevent>`元素的子元素。当所述`<mml:onevent>`元素中止所期望的事件时，将启用由所包含的`<mml:do>`元素规定的行为。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述任意的 *target* 属性规定了将要启用的目标元素的标识符。

所述任意的 *bref* 属性规定了与行为相关联的 URL 或脚本。如果设置所述 *target* 属性，将忽略这个属性。

所述任意的 *action* 属性规定了目标或 URL 上将要启用的动作类型。

1. “启动”（缺省值）

将启动目标元素或 URL。最后的行为取决于目标元素类型。例如：如果目标是超链接（HYPERLINK），用户代理将遍历（traverse）它。如果目标

是表单，将提交它。

2. “重新设置”

所述目标元素将被设置为初始状态。

所述`<getvalue>`元素：

元素	属性	最小内容模型
<code>mml: getvalue</code>	<code>id(ID),from(IDREF),at(client server)</code>	<code>EMPTY</code>

所述`< mml: getvalue >` 元素是`< mml : prompt >`的子元素。其用于从`<form>`或`<sform>`数据占位符中获得内容。

所述任意的 *id* 属性规定了元素在整个文档范围内的唯一标识符。

所述需要的 *from* 属性规定了所述数据占位符的标识符。

所述`< mml: getvalue >` 获得客户端元素的值。

在这种情况下,所述 *from* 属性应当设置成`<form>`的数据占位符。

所述 *at* 属性规定将所需要分配的值分配至客户端还是服务器:

1. “客户端”(缺省值)

所述`<mml:getvalue>` 得到客户端元素值。

在这种情况下, *from* 属性应该被设置为`<form>` 的数据占位符。

2. “服务器”

所述`<mml:getvalue>` 得到服务器端数据值。

在这种情况下, *from* 属性应该被设置为`<sform>` 的数据占位符。

例如：

a. `< mml: getvalue at= "client">`

```

<mml:card>
  <mml:onevent type="onload">
    <mml:do target="promptclient" type="activation">
      </mml:do>
    <form id="form01" action="other.mml" method="post">
      <input id="text1" type="text" name="ADD" value="ShangHai"/>
    </form>
    ...
    <mml:speech>
      <mml:prompt id="promptclient" type="tts">
        the place you want to go, for example <mml:getvalue
        from="text1" at="client"/>
      </mml:prompt>
      ...
    </mml:speech>
  </mml:onevent>
</mml:card>

```

处理流程如下：

- 所述客户端用户代理加载这种界面卡。
- 将触发所述“加载”事件以及然后将启动<mml:prompt>。
- 然后将处理<mml:getvalue>。文本框“text1”的值将由<mml:getvalue>检索。在这里所述文本框的初始值是“ShangHai”。
- 最后，客户端将与用户交谈：“你想去的地方，例如：上海”。

b. <mml:getvalue at="server"> :

```

<mml:card>
  <form id="form01" action="other.mml" method="post">
    <input id="text1" type="text" name="add" value="ShangHai"/>
  </form>

```

```

< mml: speech >
  < mml : prompt id="promptServer" type="tts" >
    The place you want to go < mml: getvalue from="stext1"
  at="server"/ >
  < /mml : prompt >
  < mml: recog >
    < mml: sform id="sform01" bind="form01" >
      < mml: grammar src="Add. gram" />
      < mml : input id="stext1" value="#add" bind="text1" />
      < mml : onevent type="match" >
        < mml: do target="promptServer"
        type="activation" />
      < /mml: onevent >
    < /mml:sform >
    < /mml: recog >
  < /mml: speech >
  < /mml: card >

```

处理流程如下：

- 用户输入语音并执行语音识别。
- 如果可以接受所述语音识别的语音输出值，则触发“匹配”事件。
- 在提交表单之前，服务器将首先处理“匹配”事件处理程序(<mml:onevent>)。然后,事件消息将发送到客户端，如下：

```

< event type="match" >
  < do target="promptServer" >
  < input id="stext1" value="地点" />
  < !-'根据识别结果-- >

```

< /event >

- 然后,当客户端处理< mml:prompt >和< mml : getvalueat = " server " >时,从服务器接收的值将分配给< mml: getvalue >元素。
- 最后,客户端将与用户交谈。所述交谈内容与语音识别的结果有关。

根据本发明的一个实施例,下文描述了通过无线网络上的多模式网络交互的系统中客户端机和服务器交互的流程。

与传统的网络交互和电话交互不同,本发明的系统支持多模式网络交互。因为主要语音识别处理工作将由服务器控制,因此多模式网页将在客户端和服务器两者上解释。以下是使用本发明实施例的客户端和服务器交互的简单流程。

< User> :选择超链接,提交表单(传统的网络交互)或按下“谈话按钮”和输入发言(语音交互)。

< Client > :在传统的网络交互的情况下,所述客户端向服务器发出新页面或提交表单的请求。在语音交互的情况下,所述客户端确定将聚焦于哪个活动的显示元素和所聚焦的显示元素的标识符、捕获语音、提取语音特性,以及向服务器发送所聚焦的显示元素的标识符、所提取的语音特性和例如当前页 URL 的其它信息。然后,所述客户端等待应答。

< Client > :在传统的网络交互的情况下,服务器从高速缓存或网页服务器检索新的页面并将其发送给客户端。在语音识别的情况下,所述服务器接收所聚焦的显示元素的标识符并创建正确的语法。然后,将执行语音识别。根据所述语音识别的结果,所述服务器将执行特定工作并向客户端发送事件或新的页面。然后,所述服务器等待来自客户端的新请求。

< Client > :客户端将加载新的页面或处理事件。

由此，公开了由 MML 实现的具有聚焦机制、同步机制和控制机制的创造性的多模式网络交互方法。以下阐述的权利要求保护的范围不限于与在此所提供的本发明的各种实施例的详细说明有关的描述的细节。

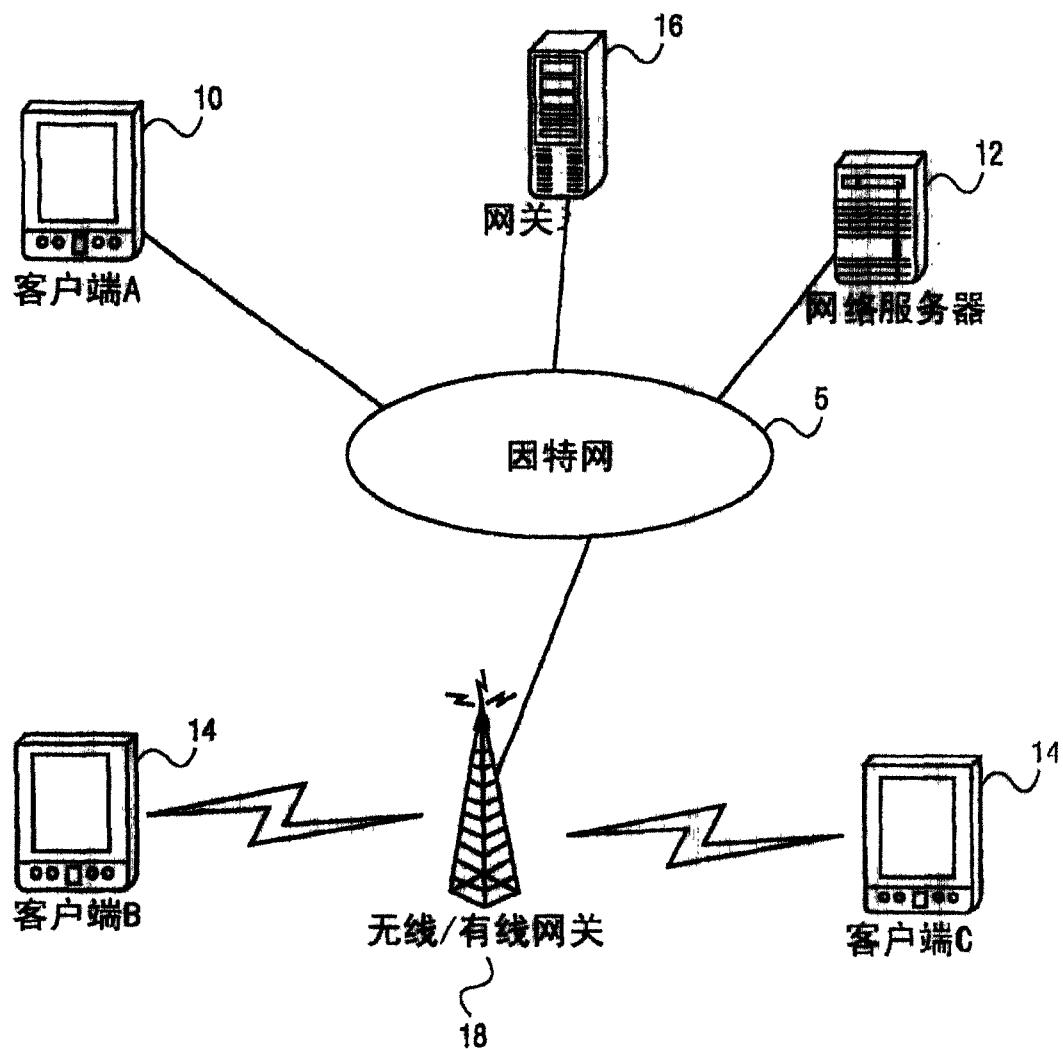


图 1

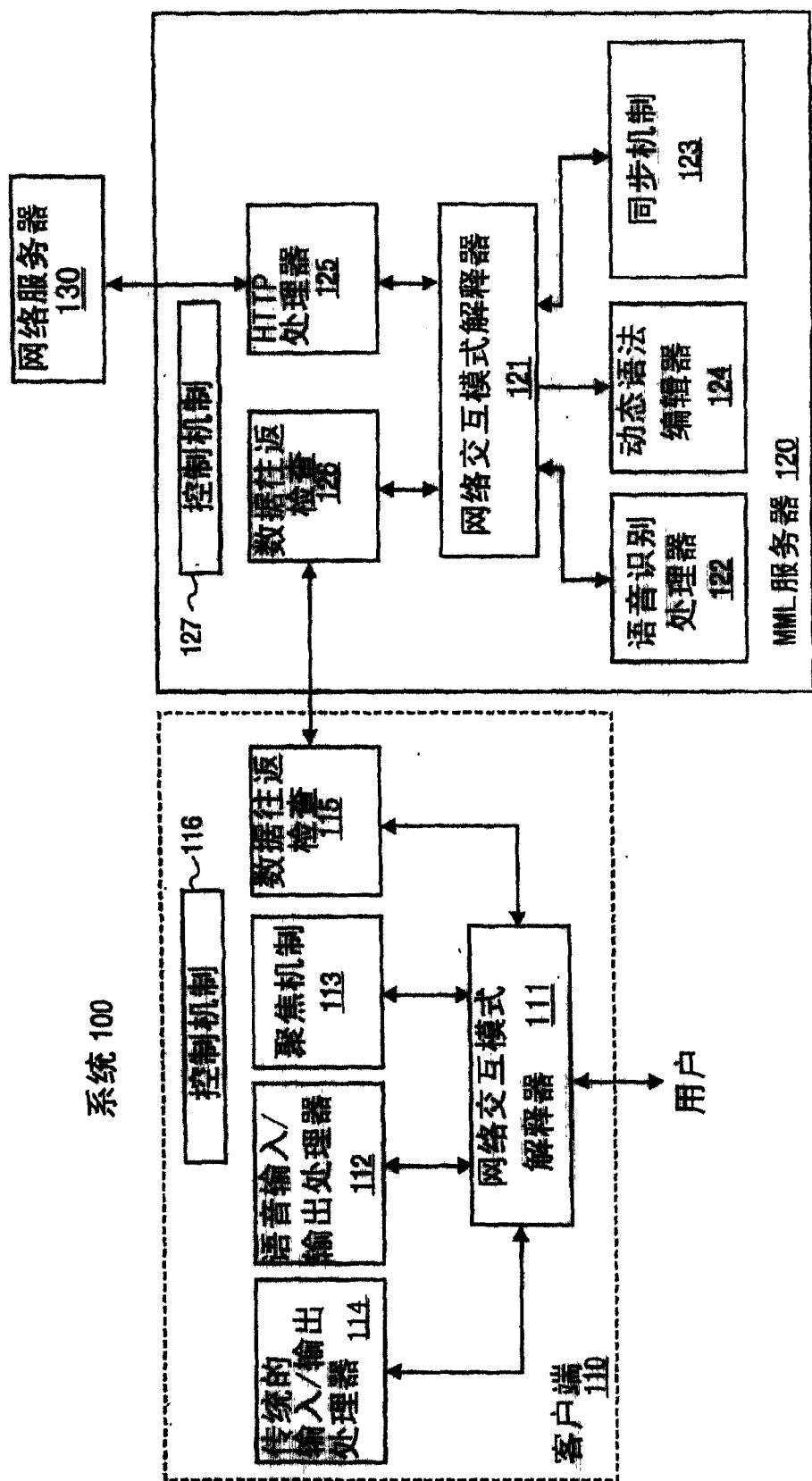


图2

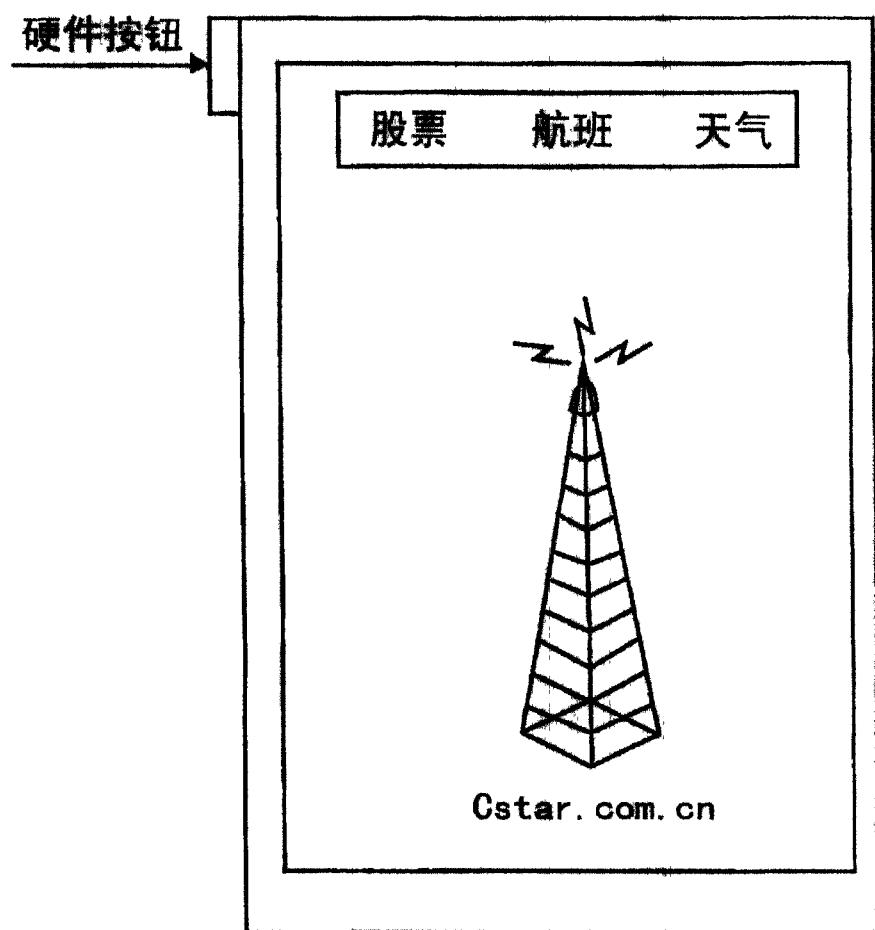


图 3

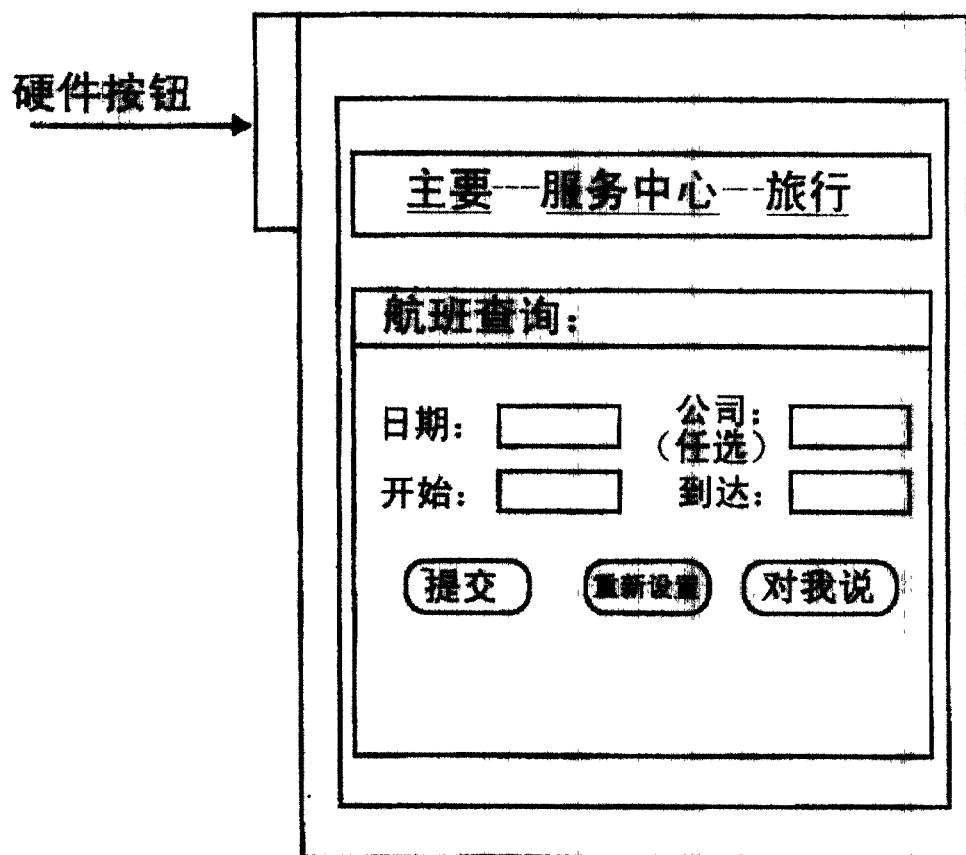


图4

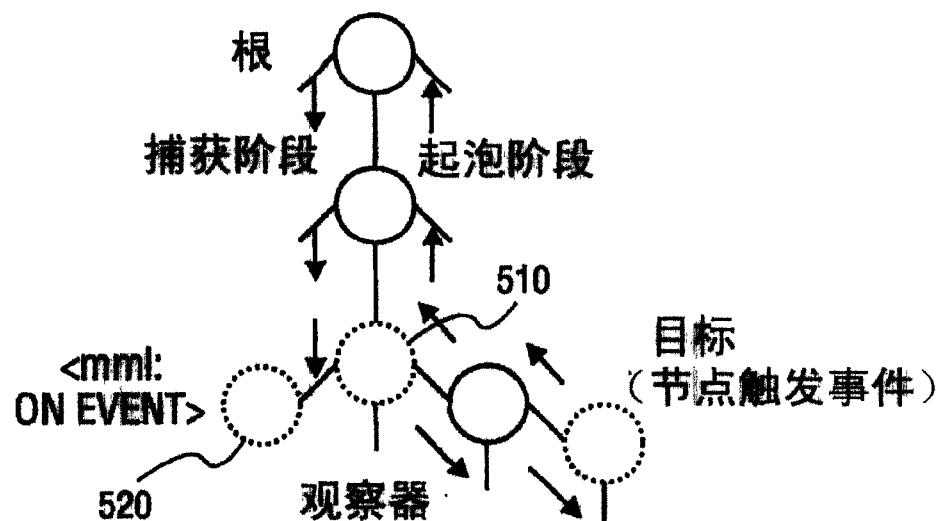


图5

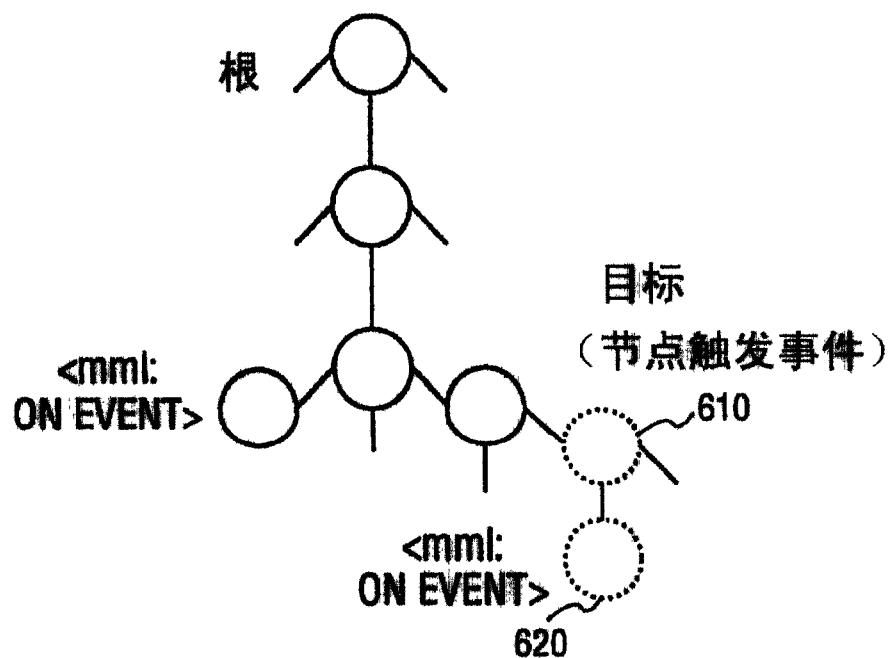


图6

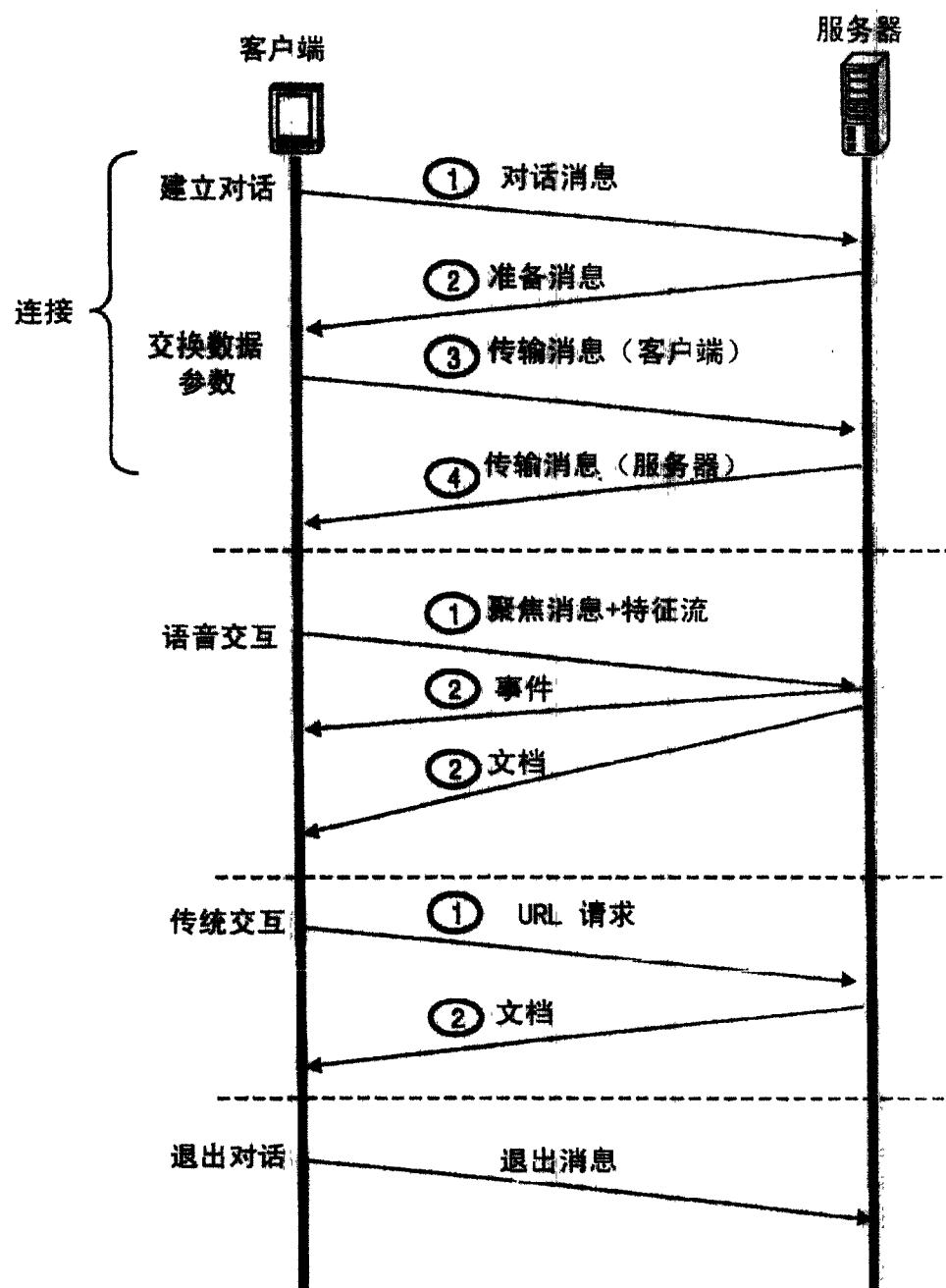


图7

