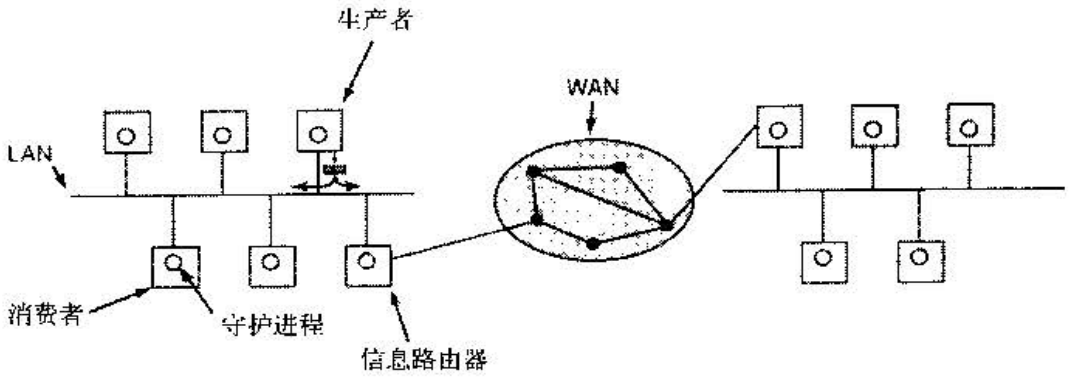

图8-41 发布/订阅的体系结构

3. Jini

50多年来，计算始终是以CPU为中心的，一台计算机就是一个独立的装置，包括一个CPU、一些基本存储器、并总是有诸如硬盘等这样一些大容量的存储器。Sun公司的Jini（基因拼写的变形）则是企图改变这种计算模型的一个尝试，这种模型可以描述为以网络为中心（Waldo，1999）。

在Jini世界中有大量自包含的Jini设备，其中的每一个设备都为其他的设备提供了一种或多种服务。可以把Jini设备插入到网络中，并且立即开始提供和使用服务，这并不需要复杂的安装过程。请注意，这些设备是被插入到网络中，而不是如同传统那样插入到计算机中。一个Jini设备可以是一台传统的计算机，但也可以是一台打印机、掌上电脑、蜂窝电话、电视机、立体音响或其他带有CPU、一些存储器以及一个（可能是无线）网络连接的设备。Jini系统是Jini设备的一个松散联邦，Jini设备可以依照自己的意愿进入和离开该联邦，不存在集权式的管理。

当一个Jini设备想加入Jini联邦时，它在本地局域网上广播一个包，或者在本地无线蜂窝网上询问是否存在查询服务（lookup service）。用于寻找查询服务的协议是发现协议（discovery protocol）以及若干Jini硬线协议中的某一个。（另一种寻找方法是，新的Jini设备可以等待直到有一个周期性的查询服务公告经过，但是我们不会在这里讨论这种机制）。

当查询服务看到有一个新的设备想注册时，它用一段可以用来完成注册的代码作为回答。由于Jini是纯的Java系统，被发送的代码是JVM（Java虚拟机语言）形式的，所有的Jini设备必定能运行它，通常是以解释方式运行。接着，新设备运行该代码，代码同查询服务联系并且在某个固定的时间段中进行注册。在该时间段失效之前，如果有意愿，该设备就可以注册。这一机制意味着，一个Jini 设备可以通过关机的方式离开系统，有关该设备的曾经存在的状态很快就会被遗忘掉，不需要任何集中性的管理。注册一定的时间间隔的做法，称为取得一项租约（lease）。

请注意，由于用于注册设备的代码是通过下载进入设备的，因此注册用的代码会随着系统演化而被修改掉，不过系统的演进并不会影响设备的硬件和软件。事实上，设备甚至不用明白什么是注册协议。设备所需明白的只是整个注册过程中的一段，即注册的设备提供的一些属性和代理代码，这样其他设备稍后将会使用这些属性和代理代码，以便访问该设备。

寻找某个特定服务的设备和用户可以请求查询服务是否知道这样的一个特定服务存在。在该请求中可以包含设备在注册时使用的属性。如果请求成功，在该设备注册时所提供的代理就会被送回给请求者，并且加以运行以联络有关设备。这样，设备或用户就可以同其他的设备对话，而无须知道对方在哪里，甚至也无须知道对话所用的协议是何种协议。

Jini 客户机和服务（硬件或软件设备）使用JavaSpace进行通信和同步，这方式实际是模仿Linda 的元组空间，但存在一些重要的差别。每个JavaSpace由一些强类型的记录项组成。这些记录项与Linda的元组类似，不过它们是强类型的，而Linda的元组则是无类型的。在每个记录项中包含一些域，每个域中有一个基本Java类型。例如，一个雇员类型的记录项可以包括一个字符串（用于姓名）、一个整数（用于部门）、第二个整数（用于电话分机号）以及一个布尔值（用于全时工作）。

在JavaSpace中只定义了四个方法（尽管其中的两个方法还有一个变种）：