应用服务器消息分发组件的设计与实现①

孙 倩 朱晓民 马 拓

(北京邮电大学 网络与交换技术国家重点实验室 北京 100876; 东信北邮信息技术有限公司 北京 100191)

摘 要: 首先介绍应用服务器 EBAS(EB Application Server)的概况并分析其存在的弊病，然后据此设计消息分发组件，详细介绍了消息分发组件的结构和处理逻辑，包括消息分组件与外部实体的消息交互方法、维护外部实体的状态的准则、区别外部实体是重连还是重启的策略。添加消息分发组件的 EBAS 可靠性更高，支持在线无损升级，更加适合大容量、高并发的业务模型。

关键词: 消息分发; 心跳机制; 应用服务器; 无损升级; 注册

Design and Implementation of EBAS Message Distribution Component

SUN Qian, ZHU Xiao-Min, MA Tuo

(State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; EBUPT Information Technology Co. Ltd., Beijing 100191, China)

Abstract: This paper introduces the application server EBAS (EB Application Server) and makes an analysis of the shortcomings of its existence. Then, it designs a module of distributing the message components accordingly, to achieve distribution and load sharing of messages. It expounds the structure of news distribution and the processing logic, including the message distribution components interaction with external entities, the rules to safeguard the state guidelines for external entities and the strategy to distinguish restart from reconnect of external entities. EBAS with message distribution component is more reliable, supportive of online non-destructive upgrade and more suitable for the service model with large capacity and high concurrency.

Keywords: message delivery; heartbeat; application server; non-destructive upgrade; register

1 应用服务器

EBAS(EB Application Server)是东信北邮下一代核心业务平台的重要组成部分,是一个基于事件的应用中间件。它主要的任务是把来自于外部的事件消息分发到具体的应用中去。它是一个完全开放的业务平台，通过名为 XJoin 的事件驱动的业务框架提供一套简单易用的业务开发规范，不同开发商所开发的符合 XJoin 业务开发规范的业务都能够在其上部署运行。它是一种独立于网络的业务平台，通过部署资源适配器，可以支持多种外部资源。

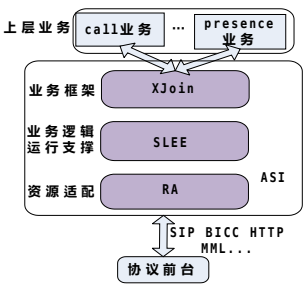

图 1 EBAS 系统结构

① 基金项目:国家杰出青年科学基金(60525110);国家重点基础研究发展规划(973)(2007CB307100;2007CB307103);国家自然科学基金(60902051);中央高校基本科研业务资助(BUPT2009RC0505);电子信息产业发展基金项目
收稿时间:2009-12-06;收到修改稿时间:2010-01-15

如图 1 所示，EBAS(以下简称 AS)包含资源适配 RA(Resource Adapter)组件，业务逻辑运行支撑 SLEE(Service Logic Execution Environment)组件，业务框架 Xjoin[1,2]。增值业务的业务逻辑基于 Xjoin 提供的接口开发并运行在 ASI(AS Instance)上。

AS 接收协议前台发来的消息(SIP,BICC, HTTP, MML 等)，处理并返回处理结果[3]。

在原有架构下，每一个前台连接一个 AS，协议前台把消息直接发送到当前连接的 AS 进行处理。

这种架构存在着两个缺点：第一，如果 AS 进程出现异常而宕掉，对于业务来说会处于一种不可用状态，此时，与此 AS 相连的前台收到的消息均无法处理，造成呼损，这种现象在电信级应用中是不可容忍的；第二，前台的处理能力大于 AS，AS 与协议前台一一对应配置，会造成协议前台永远无法最大效率的发挥，当业务量扩大到一定数量级时 AS 会遇到处理能力的瓶颈，此时前台的处理能力并没有充分发挥。

2　消息分发组件的设计

消息分发组件 Dispatcher 是一个消息中间件，负责应用服务器中所有协议消息的接入与转发，对 AS 实例进行消息的负荷分担以及接入控制，处于 ASI(AS Instance)与协议前台之间。它可以接收外部命令控制，有选择的屏蔽某个 AS 实例的消息接入，从而实现业务无损升级的功能。

消息分发组件 Dispatcher 是一个消息中间件，负责应用服务器中所有协议消息的接入与转发，对 AS 实例进行消息的负荷分担以及接入控制，处于 ASI(AS Instance)与协议前台之间。它可以接收外部命令控制，有选择的屏蔽某个 AS 实例的消息接入，从而实现业务无损升级的功能。

增加协议分发模块后的运行视图如图 2 所示。消息从各协议前台直接发送给 Dispatcher，由该模块决定将该消息转发到哪个 AS。一个 Dispatcher 下可以挂多个 AS，各 AS 可以分布在同一个服务器的多个进程中，也可以分布在多个服务器中[4]。

在这种架构中，如果一个 AS 宕掉，只会损失此 AS 当前处理的呼叫，对于整体业务来说新的呼叫仍旧可以路由到其他 AS 中，因此还是可用状态。同时，业务整体处理能力也会成倍提高。

除此之外，这种架构还能够提供业务的无损升级功能。可以用一种平滑的方式来停止其中的一个 AS：继续保持该 AS 中已有呼叫，不再路由给该 AS 新的呼叫，直到呼叫数为零后关闭，同时启动一个新版本的 AS 来接收新的呼叫。按照同样的方式依次处理余下的旧版本 AS 即可完成无损升级[5]。

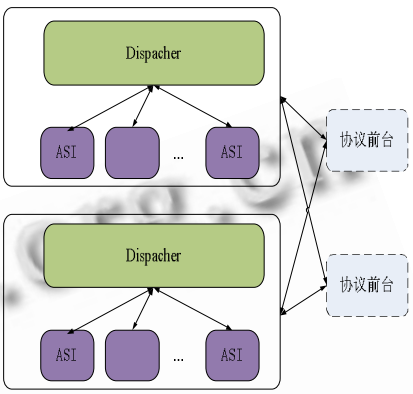

图 2　添加消息分发组件的 AS 运行视图

3　消息分发组件的实现

Dispatcher 中主要包含 Router，RouteMapInfo，Register，Listener 四大功能模块。其中，Router 用于路由消息；RouteMapInfo 中存储路由记录项，路由映射关系；Register 用于向 Dispatcher 中注册外部连接的设备信息；Listener 用于监控外部设备的连接状态。图 3 给出了 Dispatcher 结构视图。

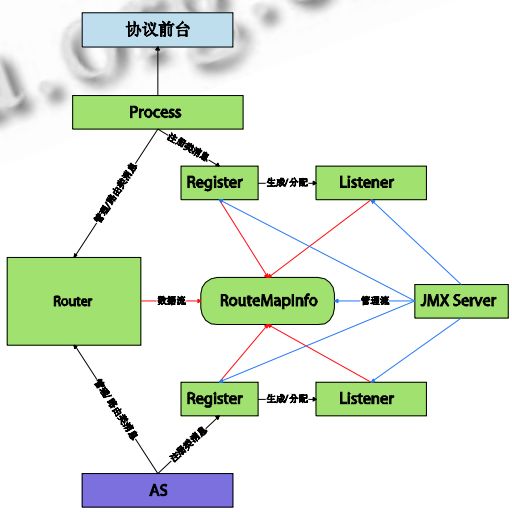

图 3　Dispatcher 内部结构

下面主要介绍 RouteMapInfo，Register，Listener 三个模块。

3.1 Register

注册模块 Register 功能是区分各个外部实体的。外部实体初次和 Dispatcher 通信需要首先注册，Register 模块处理注册消息，以此标识外部实体。和 Dispatcher 通信的外部实体有前台和后台两种，这里的后台设备是指多个 AS，下文同。

3.1.1 前台注册消息交互

当一个前台设备与 Dispatcher 建立连接的时候，会向 Dispatcher 发送注册消息，注册消息中携带前台的唯一标识。Dispatcher 根据唯一标识，来标识一个前台。

某些协议的前台不会主动发送注册消息，所以在有此类协议新的 Socket 连入时，Dispatcher 自动分配一个号作为新连入前台的标识。

3.1.2 后台注册消息交互

当有一个后台设备与 Dispatcher 建立连接的时候，Dispatcher 会主动向后台设备发送注册消息，注册消息体中含有 Dispatcher 分配给后台设备的唯一标识号。后台设备在接收到注册消息后，检查自己是否已经有 Dispatcher 分配的标识号，如果没有，则将 Dispatcher 分配的标识号作为自己的标识存储起来。

后台向 Dispatcher 回送注册消息的响应消息，消息体中含有后台设备的标识号以及后台设备所关联的信息。Dispatcher 接收到注册响应消息后，取出设备标识和以及关联信息。将设备标识号与发送的注册消息中分配的设备标识号进行比较，如果相等，则代表后台设备是一个新启动的设备；如果不相等，则代表后台设备不是一个新启动的设备，新建立的 Socket 很可能是之前断开的 Socket 的重连。

通过上述方法，Dispatcher 可以区分出一个后台设备是重启还是重连。

3.1.3 设备标识及状态维护

在 Dispatcher 中，对每个设备都有两种状态标识：Socket 状态和 Manual 状态。

Socket 状态标识了 Dispatcher 与该设备连接的链路状态，有 active，pending 以及 stop 状态，前两种状态分别对应 activeMap，pendingMap。根据所在 Register 中 Map 表的不同来区分不同设备的状态。

Active 代表与该设备连接的链路正常，可以正常的收发消息。存放在 activeMap 中的设备认为是链路正常的设备。

Pending 代表与该设备连接的链路异常，不能进行消息的收发。但是 Dispatcher 不认为对端设备已经宕机，等待对端设备进行重连，将设备标识存放在 pendingMap 中。

Stop 代表对端设备宕机或不存在。在 activeMap 以及 pendingMap 中不存在的设备都认为是 Stop 的设备。

3.2 RouteMapInfo

当处理某一会话时，Dispatcher 需要记录此会话的路由信息；当会话结束时，Dipatcher 需要删除次路由。RouteMapInfo 模块主要用来存储路由信息。路由表的维护是一项重要的工作。

在 Dispatcher 中，有两类路由表：有定时器路由表和无定时器路由表。路由表的基本存储结构为 Map<Integer, FeIdentityWrap>，其中键为会话号，值为对应前台，后台的标识。

3.2.1 有定时器的路由表

这种路由表中，每个存入消息都会启动一个对应的定时器，删除该路由项时会取消对应的定时器。如果定时器超时，则会删除路由表中的对应信息。

这种机制的路由表，每个路由项都是有定时器维护的。不会由于某个设备连接断开而造成对应路由项不能删除。但是这种定时器机制比较消耗系统定时器资源。消息比较多的时候，不适合使用这种定时器机制的路由表。

3.2.2 没有定时器的路由表

该路由表中，对于路由项不使用定时器管理。这样，如果某个设备连接断开了，则会使得在路由表中相应的路由项无法删除，造成内存泄露。

对于这种路由表，在存储时，除了基本的存储结构外，还需要附加的存储结构：即设备标识与所使用会话号列表的对应映射关系 Map<String, List<Integer>>。每次插入或删除一个路由项时，要对设备与会话号列表的映射关系也进行相应的操作。这样，在某个设备连接断开时，Dispatcher 能够根据映射关系表很快的得到与该设备相关联的所有会话，删除路由表中对应会话号以及通知相应设备释放资源。

3.2.3 资源清除

有定时器的路由表可以根据定时器自动清除资源。Dispatcher 中，对于没有定时器的路由表启动全局定时器，在指定时间对路由表中路由项的时间进行检查，如果时间大于指定门限的话，就清除该资源；在判断出某个前台设备断开连接时，只需要向后台设备发送连接断开消息通知后台进行处理，不发送针对每个会话号的会话结束消息[5]。

3.3 Listener

Listener 模块用于监控 Dispatcher 与外部设备的连接情况，每一个与 Dispatcher 通信的外部实体(前台和 ASI)都有一个 Listener 与其对应。Listener 主要有如下 3 点功能。

3.3.1 心跳机制

Dispatcher 通过心跳机制检测前台和后台的系统状态。Listener 启动发送心跳定时器，在指定时间间隔发送心跳消息；启动接收心跳定时器，接收对端发送的心跳消息。

如果发送心跳失败或是在指定时间内没有收到对端设备的心跳，则认为对端设备连接断开。当判定对端设备连接断开时，在 Register 中，将对端设备的状态设置为 pending。此状态下，从外部接收到新会话的消息不再路由到该设备。从外部接受到的已存在会话的结束消息会缓存在 Listener 中。

3.3.2 判断重启还是重连

由于前台重启后，标识仍然不变。所以设计时，在 Dispatcher 中通过时间来判断前台是重连还是重启。但是实际应用中发现前台重启的时间非常端，导致通过时间无法判断出前台的状态。需要借助使用其它的判断条件。

现在与 Dispatcher 连接的前台主要有四种：SIP，BICC，SMP，CMPP。

SIP 前台重启后，会发送 clear 消息来通知对端。所以通过这个消息可以判断出是重连还是重启。

CMPP 前台不涉及到会话资源，并且没有唯一的标识号。也就是说，在 Dispatcher 中，每次有 CMPP 前台连接，都认为是一个新的接入。

SMP 前台标识唯一，并且没有消息指定是否重启。但是在 Dispatcher 的 MML 路由表中，每个路由项都对应有一个定时器，并且没有会话资源的对应。这样，可以认为每次 SMP 前台都是一个新连入的，处理应该没有问题。

BICC 前台的标识唯一，并且没有消息指定是否重启。而且在 BICC 模块中，有会话资源的保存，并且没有对每个会话都启动定时器。所以，如果认为每次 BICC 前台都是一个新连入的，在 Socket 闪断时，会造成会话损失。

3.3.3 设备重连处理

当判断出对端设备连接不正常后，Listener 启动一个 Disconnect 定时器。

在该定时器时间内如果连接断开的设备又重新连接，则认为该设备只是 Socket 断开，没有重新启动，已经存在的会话可以继续正常处理。这时，将设备的状态设置为 active，并且将 Listener 中缓存的结束消息发送给重新连接的设备。

在 Disconnect 定时器时间内如果连接断开的设备没有重新连接，则认为该设备已经宕机。这时，将 Map 中保存的与该设备相关的资源全部删除掉，并且在删除的时候，通知相关的其它设备。

4 总结

本文首先概述应用服务器 EBAS 并介绍其系统结构，分析 AS 存在的 2 个弊病。一是当 AS 异常无法处理呼叫时，会产生大量呼损，导致系统可靠性低；二是 AS 与业务前台一对一分配无法充分发挥前台的能力。然后据此在 EBAS 上设计添加消息分发组件 Dispatcher，并详细介绍主要模块的功能及处理策略。

添加消息分发组件的 EBAS 能够灵活地与业务前台和 ASI 组网，可靠性更高，并能够支持在线无损升级，更加适合大容量、高并发的业务模型，已经应用于多媒体彩铃业务系统、并在上海移动研究院完成了业务的功能与性能测试，达到了理想预期。

参考文献

1 闫鹏,廖建新,林秀琴,曹予飞,荀兆勇.基于 JAIN SLEE 的业务逻辑执行环境的设计与实现.电信工程技术与标准化,2006,19(12):78－79.

2 李永平,邹华,陈俊亮.基于下一代网络的 SLEE APIs 的研究.高技术通讯,2003,13(7):1－5.

3 阮稳,王纯,杨军,罗诚,王金柱,廖建新.利用 EB-NCSP 平台构建 TD-MRBT 业务. 电信工程技术与标准化,2008,21(4):81－84.

4 吴乃星,廖建新,杨孟辉,朱晓民.基于软交换的异构集群媒体服务器中一种 LRV 负载均衡算法.电子学报,2005,33(10):1745－1750.

5 李亚波,苏森,陈俊亮.下一代智能化通信网中应用服务器的设计架构.高技术通讯, 2003,13(4):12－17.

6 Herceg S, Weber M, Macan T, KATE-KOM d.o.o., Zagreb, Croatia. Challenges in Implementing an SIP-Based Application Server. 9th International Conference on Telecommunications- ConTEL 2007.