广西科学院学报
Journal of Guangxi Academy of Sciences

2007,23(4):279~282
Vol. 23,No. 4 November 2007

嵌入式网关中分布式通信模块设计
Design of Distributed Communication Module in Embedded Gateway

于宏霞,黄志春
YU Hong-xia,HUANG Zhi-chun

(广州军区综合训练基地,广西桂林 541002)
(Comprehensive Training Base of Guangzhou Theater,Guilin,Guangxi,541002,China)

摘要:以一个嵌入式多功能网关项目为背景,设计一个分布式通信模块。该模块以库的形式为上层应用提供一层简单、高效的消息通信接口,较好地解决了多个CPU上进程间通信的位置透明性,最大限度提高IP应用的性能,较好地向上层应用屏蔽了主从CPU上操作系统的异构性。

关键词:分布式系统　嵌入式系统　进程间通信　ICS
中图法分类号:TN195　　**文献标识码**:A　　**文章编号**:1002-7378(2007)04-0279-04

Abstract: This paper designs a distributed communication module based on an embedded multifunctional gateway. By a simple and high efficient library API, this module supports transparent communication between process running on multiple CPU, improves the performance of IP applications and hides the diffirence of OS between main and secondary CPU for upper applications in a more excellent way.

Key words: distributed system, embedded system, internal process communication, ICS

随着嵌入式系统的发展,人们对嵌入式系统的性能提出了越来越高的要求。许多嵌入式系统已经从单处理机体系结构转向分布式多处理器的体系结构。在嵌入式领域中许多嵌入式应用设备,为了提高性能,在多处理器体系装备有更多的具有自治能力的处理器节点,每个节点都有属于自己的CPU、内存和输入输出设备(如串口及网络接口设备等),每个节点上都运行自己的嵌入式内核。这些节点上的应用设备紧密相联,互相协调,由下层嵌入式操作系统提供分布式处理功能,形成具有分布式体系结构的嵌入式系统。

操作系统[1]对分布式处理的支持通常有两种方式,一种分布式操作系统中,整个系统只有1个单一的映象,实现复杂,需要实现任务的动态分配和迁移机制[2,3]。另一种方式在现有的单处理器操作系统内核的基础上进行扩展,使其能部分支持分布式处理[4]。这种方式的优点是开发周期短,系统易于维护,且性能较好。对于嵌入式系统而言,往往并不是依靠单一系统映象的分布式操作系统的支持,而是需要多节点之间的分布式通信机制。

实现具有分布式体系结构的嵌入式系统的关键是分布式通信机制的设计和实现。本文将以一个嵌入式多功能能网关项目为背景,讨论面向网络应用的基于消息传递的分布式通信机制的设计。

1 嵌入式多功能网关的体系结构设计

1.1 硬件体系结构

系统采用两个CPU构成,一个主CPU,一个从CPU,可扩展为多个从CPU,如图1所示。主CPU端没有网络接口,从CPU端包括两个网络接口卡,一个LAN口,一个WAN口,相应备有两个MAC地址。主CPU是系统的控制核心,负责控制和配置功能,及部分复杂的数据报转发;从CPU主要负责数据报转发。在主CPU上运行Montavista Linux内

收稿日期:2007-07-30
作者简介:于宏霞(1974-),女,讲师,主要从事嵌入式系统、分布式系统研究。

核[5]，而从 CPU 运行一个基于自行开发的简单任务构成的嵌入式实时微内核。CUP 之间采用 PCI 总线相接。

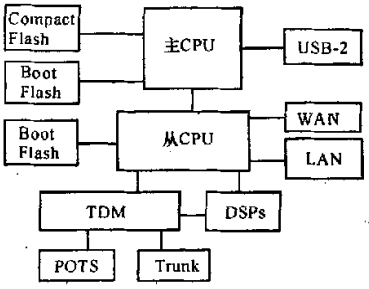

图1 嵌入式多功能网关的硬件体系结构

1.2 软件体系结构

由于各处理器节点之间不存在共享内存，因此分布式通信机制的设计使用消息传递的方法。采用图2所示的分布式操作系统的模型，对运行在从 CPU 上的微内核和主 CPU 上的 Montavista Linux 内核进行扩展，提供分布式消息通信机制。

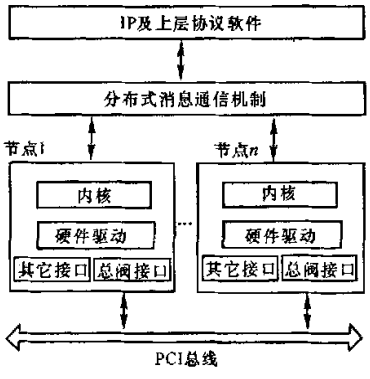

图2 分布式消息通信机制层次结构

2 ICS 模块

分布式消息通信机制架构于各节点内核之上，用于实现分布式消息通信机制的模块称为 ICS 模块。ICS 模块的设计目标是针对具有分布式体系结构的嵌入式多功能网关实现适合于网络报文处理高效、透明的消息通信。

多功能网关系统，具有分布式的体系结构，主要面向 IP 应用，需要处理各种网络报文。由于在硬件设计上，只有从 CPU 上具有网络接口，因此需要 ICS 通信模块提供的主要功能包括：(1)主 CPU 上的应用程序可以透明的正常使用网络。(2)主 CPU 上的应用程序可以通过 ICS 进行交互。(3)主 CPU 上的应用程序可以和从 CPU 上的任务进行通信。(4)从 CPU 上的应用程序可以通过 ICS 进行通信。在性能方面，ICS 通信模块的性能是整个系统性能的决定性因素之一。ICS 模块的设计要做到：(1)在多个 CPU 上实现进程间通信的位置透明性。(2)在实现通常的任务间消息通信的同时，实现面向 IP 应用的消息通信，最大限度的提高 IP 应用性能。(3) ICS 模块向上层应用设备屏蔽主从 CPU 上操作系统的异构。

2.1 ICS 体系结构

ICS(图3)分为两个部分：ICS Master 和 ICS Local。其中 ICS Master 主要由 ICS Shared lib(共享库)，ICS Virtual Device(虚拟设备)，ICS core 组成

ICS Master 和 ICS Local 向上层应用设备提供的 API 是统一的。每个 ICS 客户(指使用 ICS 进行通信的主 CPU 端应用和从 CPU 端任务)在系统中都拥有一个全局唯一的应用程序标识 AID，都可以把 AID 作为源地址和目的地址向其它 ICS 客户发送 ICS 消息，而不用关心消息目标程序的具体位置。

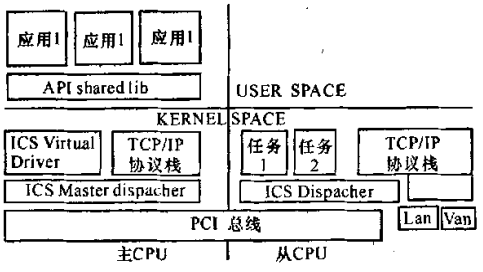

图3 ICS 系统结构

API shared lib 为上层应用程序提供统一的接口；ICS Virtual Driver 向 API shared lib 提供一个标准的 unix 文件操作函数集；ICS Master dispatcher 表示在主 CPU 这端的 ICS dispatcher；ICS Local dispatcher 表示从 CPU 上的 ICS dispatcher。

在从 CPU 中，ICS 完全是一组可供调用的 API；在主 CPU 中，ICS 由一个最底层的 ICS Master dispatcher 和一个负责应用层 API lib 交互的虚拟设备驱动程序组成。其中 ICS dispatcher 在 ICS 中将具体负责输入消息的派遣和发送消息的队列组织。ICS Virtual Driver 将面向应用程序提供服务，并且为应用程序组织一个接收消息的队列。当应用程序试图调用 API lib 获取消息时，API lib 将调用相应的设备文件操作，然后返回在 ICS Virtual Driver 中关于此应用程序队列中的报文。应用程序在获得 ICS 的服务之前，必须向 ICS Virtual Driver 进行登记，登记的内容包含了应用程序期望 ICS 采用的队列管理模型和应用程序相应的 AID 及其它

应用程序可配置的参数。

2.2 ICS 的消息结构设计实现

把主 CPU 上的所有应用程序都放在用户空间运行。同时,规定系统中的任意 2 个应用程序之间的通信都必须通过 ICS 提供的统一接口进行。系统中 2 个应用程序之间交换的消息包括普通的以太网报文和系统内部的消息报文。如果消息通信协议创建在网络协议之上,多层的协议处理会降低效率。

2.2.1 ICS 消息结构设计

系统内部无论是哪类报文,我们都将它封装成报文格式,即所谓的 SEENET 结构。SEENET 结构是自行定义的一种与标准以太网报文结构平行的消息结构。当处理主从 CPU 之间的以太网报文和内部消息报文时,也必须在原报文之前加装一个 SEENET 头(见图4)。

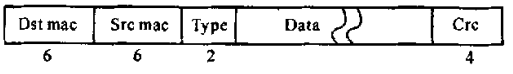

图4 SEENET 格式

图4中,Type 字段描述消息的类型,data 字段包含各种类型的自定义数据。

SEENET 消息结构以 RFC894 中描述的标准以太网报文为基础进行改装,对 2个字节的 Type 字段和 6 个字节的 MAC 地址字段进行了修改。SEENET 中的源和目的 MAC 地址分别为系统中相应的内部 MAC 地址。

当从 CPU 收到 SEENET 结构的消息时,就可知道消息从哪里来,到哪里去,并将此消息发送给相应 CPU 进行处理。

系统本身,包含两个网卡 MAC 地址,一个用于 WAN 端,一个用于 LAN 端。除此之外的 MAC 地址用作相应的系统内部 MAC。它由两部分组成,ACDE48和 Fxxxx,其中前3个字节为系统标识,用以区别系统内部 MAC 地址与实际网卡 MAC 地址。后3个字节为系统中相应内部 MAC 地址。

为了正确区分普通以太网报文和内部消息报文,对 SEENET 中的 Type 字段定义0xff01为普通以太网报文;0xff02为内部的消息报文。为了方便讨论消息的处理流程,设 ACP[123]表示在主 CPU 上运行的应用程序;TPP[123]表示在从 CPU 上运行的任务;COMM 表示负责内部通信的 PCI 设备;COMM local 表示从 CPU 上的这端,COMM master 表示主 CPU 上的这一端。

2.2.1.1 普通以太网报文

普通以太网报文(图5)的大概处理步骤如下。

	DST mac address(Hi)
DST mac address(Lo)	
DST mac address(Hi)	
SRC mac address(Lo)	Type(0xff01)
NIC index	
Original Ethernet packet	

图5 SEENET 结构封装的普通以太网报文

(1)从 CPU 发往主 CPU 的普通以太网报文

首先,从 CPU 上的 IP 协议栈能够识别这些报文将发往主 CPU 进行更进一步的处理。这些报文将通过 ICS local 额外封装一个 SEENET 头部并将其挂在 COMM local 输出队列上;其次,当这些报文通过 COMM 到达主 CPU 后,ICS master dispatcher 将根据 SEENET 结构来识别这是一个普通的 IP 报文,ICS Master 将相应的 SEENET 头部剥掉之后,将其送入主 CPU 上的相应的虚拟网卡的输入队列。再由主 CPU 上的 TCP/IP 协议栈对其进行相应的处理。

通过这种结构,我们可以断定这个以太网报文来自哪个物理网卡(从 CPU 往主 CPU 发送数据)或者必须由哪个网卡发送报文(主 CPU 通过从 CPU 发送网络报文)。

(2)主 CPU 发往从 CPU 的普通以太网报文

报文在穿过主 CPU 上的 TCP/IP 协议栈之后通过虚拟网卡的发送程序到达 ICS master dispatcher 层,ICS Master 将根据具体的报文信息为此报文封装一个 SEENET 结构,并将之送入 COMM master 的输出队列;到达从 CPU 后,ICS local 直接将此报文送入 IP 协议栈的处理队列。

2.2.1.2 内部通信报文

这里内部通信报文指的是除了普通以太网报文之外的,用于主 CPU 和从 CPU 之间协作的内部消息通信报文。定义如图6的消息结构。

					Dst mac address(Hi)
Dst mac address(Lo)					
Src mac address(Hi)					
Src mac address(Lo)					Type(0xff02)
DST AID					
SRC AID					
N	E	RESI	PRIO	RES2(8bits)	DATALEN
APPLICATION DATA					

图6 SEENET 结构封装的内部通信报文

图6中,N 表示 NACK bit,当消息无法解析时,

此位决定是否向发送者返回错误控制报文；E 表示 ERROR bit，当此位置为1时，表示此报文是一个错误控制报文；PRIO：Priority 字段（3bits）；RES1：保留字段（3bits）；RES2：保留字段；DST/SRC AID：源/目的 Application ID；DATALEN：应用程序相关的数据的长度。

由 ACP1到 ACP2的消息是 ACP1通过用户空间的接口库将消息发送到 ICS virtual driver，ICS driver 调用 ICS master dispatcher 相应的接口函数，接口函数根据 DST AID 得知这个消息发往本地，再根据消息中的 DST AID 和 PRIO，将此消息挂入 ACP2 的输入队列。

由 ACP1到 TPP1的消息与 ACP1至 ACP2的处理流程一样，ICS master dispatcher 收到相应的报文之后，根据 DST AID 可知此消息将发往从 CPU，再根据 PRIO 将其挂入 COMM master 的相应输出队列。当 ICS local 收到此报文后，将其送入从 CPU 的全局消息队列。

由 TPP1到 ACP1的消息是 TPP1直接调用 ICS local 的接口函数，ICS local 将此消息挂在 COMM local 的输出队列上。此消息到达 ICS master dispatcher 后，将根据 AID 挂入 ACP1的输入队列。

由 TPP1到 TPP2的消息是直接调用 ICS Local 的发送函数，将消息直接送入全局的接收队列即可。

2.2.2 AID 设计和实现

设计的进程间消息通信使用的 AID 是“CPU 号＋进程号”的两段格式。在分布式语音网关系统中，大部分的应用程序都将分为两个部分，一部分驻留在主 CPU 上，作为一个进程存在，称为 Master；另一部分驻留在从 CPU 上，作为任务存在，称为 Local。他们都将获得单独的 AID。并且，AID 的分配应当反映出这种关系，即提供一种手段，实现 Master 和 Local 的 AID 的相互映射。

AID 是一个32位的无符号整数，其具体分配如图7所示。

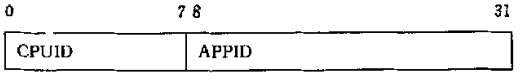

图7 AID 的组成

前面1个字节是 CPU 的编号，后面3个字节是具体应用程序编号。CPU 的编号和当前程序运行在哪个 CPU 相关，假定主 CPU 的编号为0，从 CPU1的编号为1，某应用的 APPID 为 1，则此应用 主 CPU 上 Master 进程的 AID 为 0x00000001，从 CPU1上 Local 任务的 AID 为0x01000001。

如果 AID 的 CPU 编号为0xff，则表示向多个 CPU 的广播，对应消息会发向各个 CPU 的对应进程。例如，目标 AID＝0xff000001 表示消息将发向所有 CPU 上 APPID 为1的进程。从根本上来说，AID 和 MID 实际上是一对一的关系。只是 AID 是动态分配的。进程控制模块动态维护此 MID 到 AID 的映射表，任一 ICS 客户可以向此进程控制模块查询获得此 MID－AID 动态映射关系，利用 AID 向对应目标 ICS 客户发送 ICS 消息，从而实现分布式消息通信的位置透明。

3 结束语

对于分布式消息通信模块，消息结构设计是关键。AID 的设计较好地解决了多个 CPU 上进程间通信的位置透明性问题。定义的 SEENET 结构是与网络报文在结构上平行的报文结构，而不是架构在网络报文之上，充分考虑到系统面向 IP 应用的特点的同时最大限度的提高了 IP 应用的性能。面向应用层的统一接口设计则较好的向上层应用屏蔽了主从 CPU 上操作系统的异构性。在消息结构设计上较好的解决了分布式消息通信的几个关键性问题。

参考文献：

[1] ANDREW S TANENBAUM. 现代操作系统[M]. 陈向群，译. 北京：机械工业出版社，1999.

[2] HIDEYUKI TOKUDA，CLIFFORD W MERCER. ARTS：a distributed real-time kernel [J]. Operating Systems Review，1989，23(3)：29-53.

[3] WU JIE. 分布式系统设计[M]. 高传善，译. 北京：机械工业出版社，2001.

[4] ERNESTO F V MARTINS，ANTONIO NUNES DA CRUZ. An operating system extension for a multiprocessor [J]. Journal of Systems Architecture，1998，45(5)：341-361.

[5] 李善平，刘文峰，王焕龙. Linux 与嵌入式系统[M]. 北京：清华大学出版社，2003.

（责任编辑：尹 闯）