高技术通讯 2009 年 第 19 卷 第 3 期:253 ~ 258

doi:10.3772/j.issn.1002-0470.2009.03.007

重配置系统中基于组件的协议栈管理①

迟 骋② 冯志勇

(北京邮电大学泛网无线通信教育部重点实验室 北京 100876)

摘 要 考虑到未来移动通信设备能够支持多种接入标准和协议栈,而且协议栈重配置能力将会成为一种促进异构系统融合的关键驱动技术,本文研究了协议栈重配置系统的协议组件的管理,并基于组件化的协议框架,提出了一套管理协议组件的方法。为了使得协议栈能够针对不同的通信情况对自身进行配置调整,从而达到为每个无线业务定制最佳协议栈的目标,提出了一种针对协议组件的评估机制和一种在网络协议组件库中寻找最佳组件来替代当前系统中运行效果不佳的协议组件的协议栈优化方法。上述机制和方法的最终目标是通过灵活的协议栈配置为用户提供最佳的业务体验。

关键词 协议组件,重配置,组件选择,组件评估,自主通信

0 引 言

在最近十年中,移动通信技术取得了巨大发展,衍生出多样的无线接入技术,其中包括蜂窝移动通信系统,如全球移动通信系统(GSM),通用无线分组业务(GPRS)/增强型数据速率 GSM 演进技术(EDGE)/GSM/EDGE 无线接入网(GERAN),通用移动通信系统(UMTS),CDMA2000;无线局域网系统,如 HIPERLAN/2,IEEE802.11a/b/g 等;广播系统,如数字音频广播(DAB),数字视频广播(DVB);以及短距离通信系统(蓝牙)。据预测,未来的移动通信设备将能支持多种接入标准和协议栈,重配置能力将会成为一种关键的驱动技术来促进异构系统融合,从而支持多标准的终端设备和不同网间的联合无线资源管理[1]。被称为超三代(B3G)或第四代(4G)的下一代无线移动通信系统,将会基于一种异构的网络架构,包含多种无线接入技术。这些无线接入技术在特点和优势上相互补充。B3G 时代将会引入网络的重配置能力,在业务周期内,动态灵活地调整无线协议栈,以更好地满足不断变化的业务需求。

总的来说,主要有两个方面导致了对协议栈重配置能力的需求。一方面,终端需要切换到一种完全不同的接入网络中。这时,终端应当下载本机中缺少的驱动软件,然后进行协议的重配置来保证无缝的连接。另一方面,在同一个接入网中,当需要提升业务质量时,也有可能触发协议栈的重配置。后者是本文关注的重点。

有许多因素影响了通信系统中协议栈的性能。文献[2-4]关注于通过调整协议的参数或配置来使协议栈适应当前的链路情况,比如丢包率、接入时延等。另一方面,文献[5]也讨论了多样的业务服务质量(QoS)需求对协议栈设计的影响。为了构建一个通用的协议平台,Alonistioti 及 Patouni 等提出了一种具有一般性的协议框架和一些相关的协议重配置机制[1,6],但其没有包含详细的协议组件管理办法。文献[12]提出了一种简单的组件选择算法,但是它仅仅考虑了每个组件的业务质量函数和时延函数,并没有涉及对正在运行组件的评估过程和整个系统的瓶颈发现机制,这样可能会导致低效的甚至是无意义的系统优化行为。

为了实现在特定的无线环境下为每个无线业务定制最佳的协议栈,我们使用 Alonistioti 等人提出的通用协议框架作为参考模型,提出了一整套协议组件的管理办法。其核心思想是在当前链路条件和业务需求的前提下,发现协议栈中运行不佳的协议组件,然后寻求合适的协议组件对其进行替换,从而达到提升协议栈性能的目标。我们假定,具有相同功能的协议组件可以存在多个版本,在设计上,不同的版本对不同的通信条件和业务具有不同的实现细节

① 863 计划(2006AA01Z276)和国家自然科学基金(60632030)资助项目。
② 男,1983 年生,硕士;研究方向:端到端可重配置系统中的体系框架设计;联系人,E-mail: chicheng613@gmail.com
(收稿日期:2007-12-07)

或参数配置。这种设计理念,使得协议栈能够针对不同的通信情况对自身进行配置调整,从而达到为每个无线业务制定最佳的协议栈这一目标。而寻求“最佳”的途径,正是本文着重考虑的问题。本文的前面一部分致力于对协议组件的评估。图论中的关键路径法被用来发现影响系统性能的协议组件。后一部分关注对备选协议组件的选择过程。以上提出的这些机制既可以在业务时间内周期性地触发,也可以由业务 QoS 的衰减来触发。

1 基于组件的协议通用框架

文献[1]和文献[6]的贡献在于将软件重配置的成果和一些前沿的概念(比如自主通信系统)结合起来,提出了解决下一代通信系统中自配置协议栈问题的新方法。这些研究工作关注于系统设计方面的自动化、自适应和自配置的问题。特别是提出了一个自配置的协议栈方案并对其进行了一定程度的验证(图 1)。这个方案包括一个通用的协议栈模型和一些相关的运行机制。通用协议栈模型的设计主要遵循以下的两点核心思想:

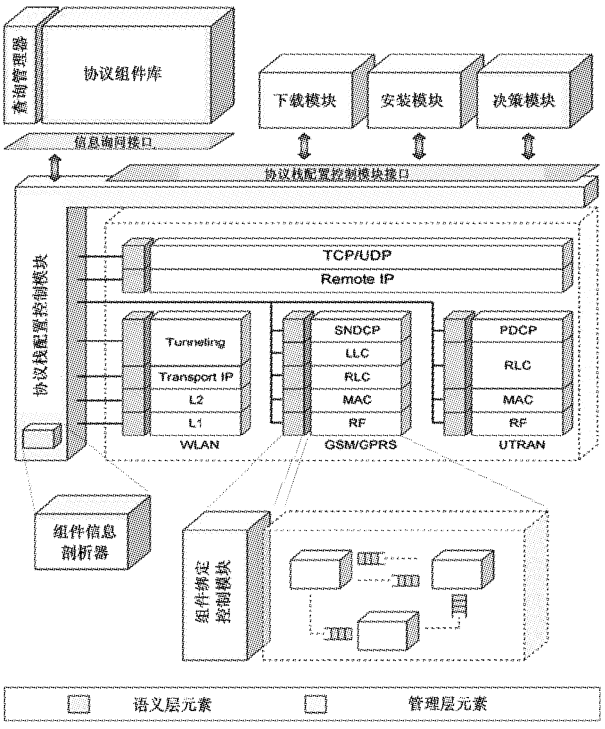

图 1 用于可重配置协议栈的通用功能模型[1]

(1) 一个协议栈由若干离散的协议层组成。层与层之间的通信可以通过标准定义接口来完成,比如服务接入点(service access point,SAP)接口。也可以使用基于队列的通信方案。

(2) 一个协议层由若干的协议组件构成。

以上介绍的框架模型主要是用来解决以下的一些问题:把一个协议组件动态绑定到现有协议栈中;在系统运行过程中实现协议组件的替换。通用协议框架由图 1 所示,相关的模块定义和功能介绍在文献[1]中有详细描述。

该框架中模块间的相互作用关系可分为三个不同的阶段:首先,发现可用的协议栈或协议组件单元;其次,通过空中接口下载并安装需要的软件包和软件库;最后,实现协议组件的动态绑定或替换,来保证协议栈的完整性。

2 协议栈评估方案

协议栈重配置的一个重要思想是在协议栈工作过程中实现协议组件的动态下载和替换。因此,协议栈的评估就理所当然地成为协议栈管理和优化的第一个重要步骤。评估是否合理准确不仅影响了优化对象的选择,而且可能进一步影响了系统优化后的稳定性。系统评估的目标是要对每个处于激活状态的协议组件进行性能度量,从而找到整个协议栈中对系统性能起到瓶颈作用的组件所在。继而,对系统性能衰减最强烈的组件应当依据当前的业务 QoS 要求和链路质量考虑适当的优化行为(比如组件替换)。而且,组件的易测性也应当在组件设计之初予以考虑和保证。

2.1 组件的易测性要求

因为组件是构成系统的单元,所以,开发高性能的组件对于基于组件的协议栈就显得尤为重要。为了开发高性能的组件,我们必须关注组件的易测性,确保组件不仅能在设计阶段被设计者测试,而且能够在运行阶段被某类效用函数自动地评估。文献[7]通过 5 个因素对易测性的具体内容给予了定义,它们分别是易读性、可观测性、可控性、可跟踪性、可支持测试性。

易读性主要是指组件在设计和实现时应便于用户(包括人和系统)了解组件的性能和规范,从而使用户能够轻松地定义对组件进行验证的相关测试和标准。

可观测性是指组件的设计应为观察组件的功能和组件测试时的行为提供便利。

可控性是指组件在运行中能够被管理实体有效地控制的操作。

可跟踪性是指组件在设计时应定义一些性能指标,通过这些指标,可以观察到组件的不同行为特征。

可支持测试性主要是指组件在验证阶段能够支持测试的能力。

2.2　针对单一组件的评估技术

针对整个协议栈的评估,前人已经做了许多工作。但是,很少研究关注于针对一个协议组件的评估。鉴于基于组件的协议栈框架的特点和考虑到后续的优化行为,针对每个组件的单独评估就不可避免。

我们提出的方案主要包括一个核心的性能模型(performance model)和被测的协议组件。性能评估的最终目标是要为每个激活的组件生成一个系统性能阻滞因子。由这些因子构成的集合将被用于下一个系统瓶颈发现的过程,该内容将在本节的第三部分详细阐述。图 2 说明了组件评估机制的具体结构。

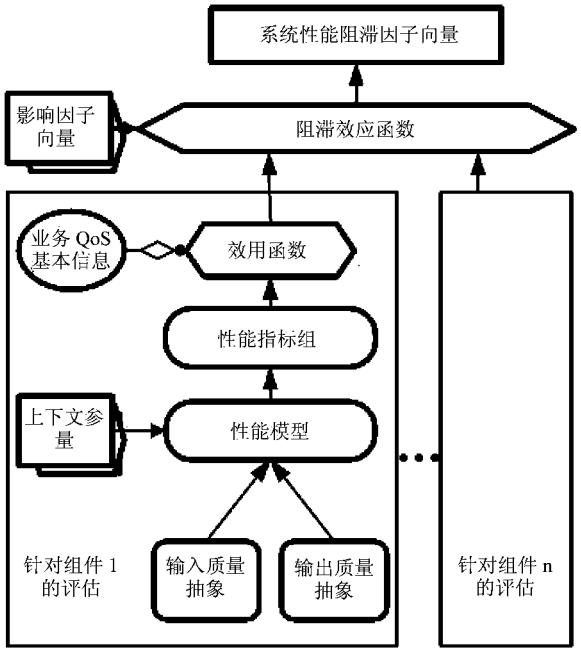

图 2　组件评估机制结构

Freedman[8]将软件模块看成一个具有输入和输出的黑盒子。对模块性能的评估可以通过观测输入和输出序列实现。我们的方案参考了他的基本思想,通过这种方式,可简化性能模型的工作复杂度。输入质量抽象和输出质量抽象是两组参数集合,它们分别描述了一个协议组件输入和输出的性能特征。上下文参量表征了系统当前的配置信息,这些信息说明了系统的状态和被测组件的工作环境参数。它连同输入输出质量抽象一起构成了性能模型的主要输入。性能模型可以被定义为一个现有的参数化模型,但是具备了一些更复杂的功能或算法。我们认为这些功能和算法是和被测组件的类型紧密相关的。因为不同类型的组件可能具有完全不同的功能,从而导致了完全不同的评估方法。性能模型通过处理三个输入,导出一个性能指标组。业务 QoS 基本信息被用来构造一个效用函数,效用函数的构造需要和组件的功能类型紧密相关。然后,根据性能模型产生的性能指标组,效用函数为该组件在当前运行状况下计算一个单一的组件效用值。

如图 2 所示,所有组件的效用值被传送给阻滞效应函数。影响因子向量表明了各个组件在当前系统中,当前业务和配置条件下,对整个系统影响力度的大小。每个组件的影响因子应当是运行业务和系统配置的函数。阻滞效应函数最终会为每个组件产生一个单一的系统性能阻滞因子值。最简单的阻滞效应函数可如下实现:

$$\beta_i = \alpha_i \times \frac{\sum_{j=1}^{n} \mu_j - \mu_i}{\mu_i} \tag{1}$$

其中,β_i 是组件 i 的系统性能阻滞因子值;α_i 是组件 i 的影响因子,它是根据当前系统的配置情况,对 β_i 的生成起到修正作用;μ_i 是组件 i 单一的效用值。n 代表了在当前系统中,总共处于激活状态的组件的数目。可以看出,效用值越低且协议栈中影响力度越大的组件,β 值也就越大,这表示它对系统的负面效应越大。我们假定 β 的理想值为 0,它意味着,该组件在当前系统中完美运行,不需要任何的优化或升级。

2.3　系统瓶颈的发现

在系统性能阻滞因子计算完成之后,我们将要深入到系统性能瓶颈的发现技术上来。此外,在这一步中确定的系统瓶颈将会成为系统优化的目标,跟随着一系列组件下载和替换的动作。我们受到了现实生活中项目管理理论的启发,关键路径法被用到这里来检测哪些组件对系统性能起到了关键的抑制作用。而且,关键路径法还能够就哪些组件应当得到优化的问题给出建议。因为优化行为需要带来系统开销,我们认为优化并不是无限自由的行为。所以,我们需要找到系统中最应优化的组件,给予它们最高的优先级,让它们的性能得到提升。

关键路径法是项目管理中的一个非常重要的工具。它适用于处理由许多小的活动组成的工程。如

果有些活动需要在其他活动完成后才能开始,那么这样的一个项目将会变成一个由众多小活动组成的复杂的项目网。关键路径法能够帮助项目管理者确定以下问题:(1) 需要多少时间这个复杂的项目才能完成。(2) 哪些活动是重要的。活动重要意味着完成的时间要首先得到保证,如果被拖延,那整个项目也会被拖延。

如果项目管理者还为每个活动定义了一些开销信息,如加速每个活动各需要多大的开销,那么,关键路径法还能够解决:(1) 你是否应该尝试加速这个项目的进程。(2) 以最小的开销加速这个项目的方法是什么。

在我们的策略中,关键路径法被用来分析协议拓扑中每个组件的性能对整个系统的影响。这里所说的协议拓扑是指基于组件的协议栈中组件和组件之间网状的结构关系。不同于传统意义上的关键路径法,我们在这里关注的不是活动的持续时间,而是每个组件的性能指标(在这里我们使用系统性能阻滞因子)。图 3 给出了系统性能阻滞因子关系的一个范例,借助于图 3 可方便地进行过程描述。具体的步骤如下:

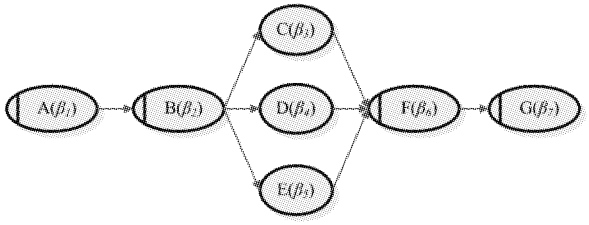

图 3 系统性能阻滞因子关系范例

(1) 首先为每个处于激活状态的组件计算系统性能阻滞因子,这一步已经在前面的章节有详细的描述。

(2) 每个处于激活状态的组件被视为图 3 中的一个点,将该点的系统性能阻滞因子与该点绑定。

(3) 依据组件和组件之间的逻辑关系构建系统性能阻滞因子关系图。我们将在逻辑上有相连关系的组件之间的关系分为两类:功能并联类和功能串连类。如果两个组件是功能串连的,那么它们分别对系统起阻滞作用,它们两个总的阻滞因子是它们各自阻滞因子之和。如果两个组件是功能并联的,那么它们中性能最差的一个对系统起到决定的阻滞作用,在这种情况下,它们两个总的阻滞因子是它们中阻滞因子值较大的那个。我们可以参照图 3的范例,组件 A 和组件 B 的总的系统性能阻滞因子为 $(\beta_1 + \beta_2)$,组件 C、组件 D、组件 E 的总的系统性能阻滞因子为 $\mathrm{Max}\{\beta_3, \beta_4, \beta_5\}$。

(4) 运用经典的关键路径算法来确定组件拓扑中的关键路径。在图 3 中,我们假定 $\beta_3 < \beta_4 < \beta_5$,那么,该图的关键路径就应该为 A($\beta_1$), B($\beta_2$), E($\beta_5$), F($\beta_6$), G($\beta_7$)。

由上述办法得到的关键路径具有以下特性:

(1) 处于关键路径上的组件决定了整个协议栈的性能。这部分组件的系统性能阻滞因子的代数和表征了整个协议系统的总的阻滞效应的大小。

(2) 处于关键路径上的任一个组件性能的退化都会导致整个系统性能的退化。

(3) 减小关键路径上的任一个组件的系统性能阻滞因子值都会使整个系统获得一定程度的性能增益。

(4) 改变处于关键路径上组件的系统性能阻滞因子值可能导致关键路径本身的变化。

(5) 整个协议拓扑中可能存在多条关键路径,它们的系统性能阻滞因子的代数和相等。

总的来说,处于关键路径上的组件的性能应首先予以保障。在优化时应该给予更高的优先级。而且,我们对关键路径上组件的优化空间也进行了研究和定义。

我们将处于关键路径上的组件分为两类。一类是“确定处于关键路径”上的点;一类是“可能处于关键路径”上的点。对于确定处于关键路径上的点,不论它的系统性能阻滞因子值为多少,它都确定处在关键路径上。图 3 中组件 A,B,F,G 就属于这一类。对于可能处于关键路径上的点,它们是否在关键路径上取决于它们的系统性能阻滞因子的值,组件 C,D,E 就属于这一类。我们针对这两种组件类型,对它们的理论优化空间做了分别定义。

对于确定处于关键路径上的点,它的优化空间应该为:

$$S_i = \frac{\beta_i}{\sum_{j \in C} \beta_j} \tag{2}$$

其中, S_i 是组件 i 的优化空间值。C 是由所有处在关键路径上的组件构成的集合。对于可能处于关键路径上的点,它的优化空间应该为:

$$S_i = \frac{\beta_i - \beta_i'}{\sum_{j \in C} \beta_j} \tag{3}$$

其中 β_i' 是一个门限,当组件 i 的系统性能阻滞因子低于它时,组件 i 将不再被包括在关键路径当中。

需要说明的是,β_i' 的具体数值是由图论中的关键路径算法和次关键路径算法决定的。由于该算法比较复杂,详细的推导过程可以参见图论的相关资料,在本文中不再赘述。

这里定义的优化空间表征了超越 S_i 的范围,针对单一的组件 i 的优化就会失去意义。这个结论对于上述的处于图中的两类组件均适用。另外,优化空间仅仅给组件的优化提升范围做了量化。因此,具体处于关键路径上的组件哪些需要被优化或替换,还要考虑备选组件的可用性,和对备选组件性能的预测结果。

3　备选组件的选择

在系统评估之后,依据业务的 QoS 需求,可能要进行一些系统优化来保证基本的通信。在我们的机制中谈到的优化主要是指通过下载新的更适合当前业务和链路条件的组件来替代旧组件的过程。当然,新组建应该能够给系统带来需要的性能增益。本文提出的组件选择机制包括两个阶段,第一个阶段是基于组件说明规范的筛选,第二个阶段是通过测试进行的试验选择。组件选择的流程如图 4 所示。

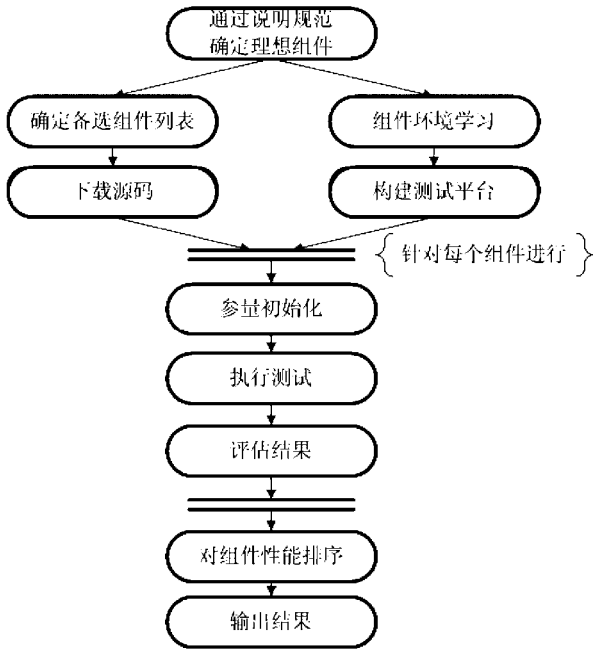

图 4　组件选择流程

整个流程开始于针对组件说明规范的分析。在文献[1]中,Metadata 被用来存储组件的基本信息,这些信息包括一个唯一的组件 ID 号、组件的版本、源码的路径和组件的绑定信息。但是,在我们的机制中,对于组件筛选来说,这些信息是远远不够的。为了能在筛选中明确组件运行的方式、环境以及组件运行对周边组件的需求,我们还需要更多的信息。详尽的说明规范需要具备以下内容:

(1) 运行规范:这部分信息至少应当包括组件具有功能的规范说明。不仅接口部件的约束条件需要说明,而且组件中部件和部件的约束条件也需要说明。

(2) 运行环境:这里需要描述在何种场合,以何种方式运行组件。其中"何种场合"至少应当对组件运行的环境做出基本估计和描述。"何种方式"至少包括该组件的期待用途,比如它在整个组件系统中扮演的角色。

(3) 非功能属性:这部分需要描述组件的性能,安全性和稳定性。

(4) 需要的接口和资源:这部分描述了组件进行正常运作时需要的功能性和资源,比如存储单元的大小和处理器的速度。组件规范中一个容易被严重忽视的部分在这里进行了说明,它是组件的独立性问题。

基于上述的信息,再依据当前系统运行的业务和链路情况,组件的筛选过程被执行。理想情况下,被称为规范说明匹配[9]的过程,应当自动地在最大尺度上执行。然后,依据筛选的结果,将可能具备优良性能的组件以备选列表的形式列出。这些备选组件的源码也在此时得到下载。同时,在终端方面,移动终端正在自主地根据当前环境构建一个测试平台,测试平台的搭建应在最大程度上模拟当前的业务情况和链路水平。上述工作完成后,应依次针对每个备选组件进行如下工作:初始化组件参数,启动测试平台,结果评估分析。使用同一个平台对具有相同功能的组件进行评估其结果的合理性是可以得到保证的。最后,对比每个组件的评估结果,将它们的性能进行排序,然后输出报告。

在这些工作完成之后,组件替换和动态绑定的工作将会后续进行,其内容不在本文讨论之列。

4　结　论

本篇论文基于组件化的协议框架,提出了一种新的针对组件的评估方法。关键路径法被引入可便于自主地发现系统瓶颈。我们同样对组件选择的机制进行了讨论。本论文提出的组件管理方案的目标是依据当前的网络链路情况为每个业务定制专用的

协议栈。这样做的好处是能够依靠协议栈的重配置能力,为用户提供更好的业务体验。

参考文献

[1] Alonistioti N, Patouni E, Gazis V. Generic architecture and mechanisms for protocol reconfiguration. *Mobile Networks and Applications*, 2006,11: 917-934

[2] Akan O B, Akyildiz I F. ATL: an adaptive transport layer siute for next generation wireless internet. *IEEE Journal on Selected Areas in Communication*, 2004,22(5):802-817

[3] Bertozzi D, Raghunathan A, Benini L, et al. Transport protocol optimization for energy efficient wireless embedded systems. IEEE, 2003

[4] Patouni E, Lilis A, Beaujean C, et al. Protocol reconfiguration schemes for policy-based equipment management. IEEE, 2006

[5] Ranjesh G, Jaganathan, Underwood K D, et al. A configurable network protocol for cluster based communications using modular hardware primitives on an intelligent NIC. In: ACM/IEEE SC2003 Conference, 2003

[6] Patouni E, Alonistioti N. Towards self-configuring protocols for reconfigurable systems. In: Proceedings of the 13th National Summiton Information and Communication Technologies, Tashkent, 2006

[7] Gao J, Tsao J, Wu Y. Testing and Quality Assurance for Component-Based Software. MA: Artech House, 2003

[8] Freedman R S. Testability of software components. *IEEE Transactions on Software Engineering*, 1991,17(6):553-563

[9] Zaremski A M, Wing J M. Specification matching of software components. *ACM Trans Software Engineering Methodology*, 1997,6(4):333-369

[10] Han J. An approach to software component specification. In: Proceedings of 1999 International Workshop on Component Based Software Engineering, Erfurt, Germany, 1999

[11] Szyperski C. Component Software-Beyond Object-Oriented Programming. 2nd edition. London: AddisonWesley, 2002

[12] IST-2005-027714 Project E2R II Deliverable 2.2. System architecture, draft reconfiguration management and control network architecture, and analysis of support protocols and mechanisms, Project E2R II Deliverable 2.2, 2007

Management for component-based protocol stack in reconfigurable systems

Chi Cheng, Feng Zhiyong

(Key Laboratory of Universal Wireless Communications , Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract

The future telecommunication equipment is able to support multi-ply radio resource technologies and protocol stacks. Moreover, the reconfiguration ability of protocol stack will be the key technology which effectively accelerates the convergence of heterogeneous wireless systems. Therefore, the paper investigates the protocol component management in reconfigurable systems based on the component-based protocol stack architecture. In order to make the protocol stack have the ability to reconfigure itself according to different communication environments so as to achieve the goal that the optimal protocol stack can be configured for each wireless service, the paper proposes an evaluation mechanism and an optimization scheme that finds the best protocol components from the network-side library to replace ill-suited ones running in the current system. The major advantage of the proposed protocol management schemes is that consumers may enjoy the best service due to the protocol reconfigurability.

Key words: protocol component, reconfiguration, component selection, component evaluation, autonomic communications