基于可编程移动软设备的主动网络执行环境*

陆月明, 钱德沛, 徐 斌, 王 磊

(西安交通大学 计算机科学与技术系 新型计算机研究所,陕西 西安 710049)
E-mail: yueming@xjtu.edu.cn; depeiq@xjtu.edu.cn
http://www.xjtu.edu.cn

摘要: 执行环境是主动网络中各种应用、移动代码的编制、管理和运行的基础,提出了一种基于可编程移动软设备的主动网络执行环境,该执行环境中的可编程单元称为可移动软设备,在功能上按协议子树划分,通过执行环境中的多路分解器,网络中的主动包和被动包都能找到相应的处理代码,在执行环境里,主动节点的功能易于扩充,大量的 API 方便用户编程,协议或应用的调试和部署简单.

关 键 词: 主动网络;执行环境;软设备;协议树;可编程单元
中图法分类号: TP393 文献标识码: A

网络计算的性能依赖于网络设备的性能和协议的性能,网络设备的性能有了大幅度的提高,但网络协议的发展速度很迟缓.主动网是一种中间节点可编程,根据功能需要能动态扩充的新型网络结构,方便网络协议的调试和部署[1].佐治亚理工学院的 Bhattacharjee,Calvert 等认为传统网络存在一些不可知情况(如拥塞发生时间和地点等)值得加强,传统网络的一些协议值得优化[2].在 DARPA(defense advanced research projects agency)主动网络团体的努力下,文献[3~5]分别进行了主动网络的研究.

1984 年,Saltzer 和 Reed 等人的端对端观点(end-to-end argument)[6],提出了网络计算或功能的放置问题,对层次协议的设计起到了很大的作用.主动网络提出了一种反传统的中间节点可编程网络,网络计算或功能的放置有了很大的改变,计算可以动态部署到执行环境中执行.本文分析了主动网节点上执行环境的放置,可编程单元划分,在主动网络原型 Softnet 上,提出了一个新型的主动网络执行环境.Softnet 的目的是支持新协议、多协议的部署和多种数据交换方式的开展,多协议的部署指 Softnet 可以同时支持两种不同性质的协议(如 IPX 和 IP)在同一物理网和节点中部署,Softnet 还支持数据报和虚电路两种数据交换方式,在这些功能的实现和设计中,执行环境起到了关键的作用.

1 执行环境的放置

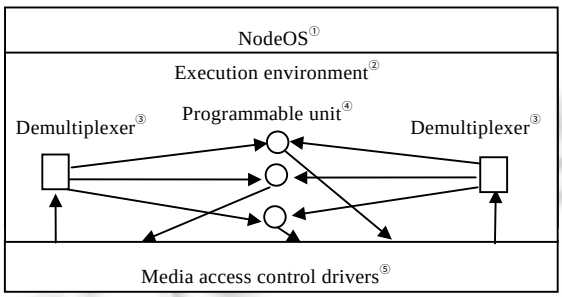
在主动网络中,执行环境存在的位置即为网络计算需要加强的地方,执行环境放置于路由协议部分,则该部分可以得到动态更新和扩充.美国 GTE 公司 BBN 技术部的 Parttridge,Strayer 等人认为增加执行环境到网络的 N 层将加强 N 层以上的各层性能[7],主动网络“Smart Packets”项目在应用层的网管代理中增加了执行环境,以管理和诊断网络,获取传统网络难以得到的信息[7].在路由协议和 IP 转发模块中增加执行环境,可以动态支持 VPN(virtual personal network)、防火墙、移动 IP 等,文献[4,5,8]等对主动网络的执行环境及放置问题进行了研

* 收稿日期: 2000-03-30; 修改日期: 2000-08-01
基金项目: 国家自然科学基金资助项目(69973036);国家 863 高科技发展计划资助项目(863-317-01-10-99);国家重点基础研究发展规划 973 资助项目(G1999032710)

作者简介: 陆月明(1969－),男,江苏苏州人,博士,主要研究领域为主动网络,网络安全,网络管理;钱德沛(1963－),男,浙江海宁人,教授,博士生导师,主要研究领域为计算机体系结构,计算机网络;徐斌(1974－),男,重庆人,博士生,主要研究领域为主动网络,网络管理;王磊(1977－),男,河南开封人,硕士生,主要研究领域为计算机网络安全.

究,其中 MIT“ANTS”项目把执行环境设计在 UDP 以上,所以不能对传统的 IP 和 MAC(media access control)协议层进行控制,哥伦比亚大学的“Netscript”项目把所有的协议设计成几个单元(box)的连接,在多协议传输的实现方面难度很大,Softnet 的执行环境的目的是实现所有的协议层可编程,并同时能支持多协议传输和多种数据交换方式.

Softnet 的执行环境如图 1 所示,它被放置于主动节点(active node)[3]MAC 层上,以加强网络的各上层协议,执行环境中运行可编程单元(programmable unit,简称 PU),可编程单元是用户或应用程序一次性编程的最小单位,执行环境和 MAC 层之间是节点操作系统(NodeOS)[3],节点操作系统处理资源的管理和调度.



①节点操作系统,②执行环境,③多路分解器,④可编程单元,⑤MAC 驱动程序.

Fig.1 The placement of Softnet's execution environment

图 1 Softnet 的执行环境位置

2 可编程单元的划分

2.1 按网络协议层次划分可编程单元

图 2 是网络协议的分层结构图,圆圈代表协议,虚线代表协议的层次划分,从下到上形成一棵协议树[9],图中协议树用 a[b[d],c[e,f]]表示.按协议层次划分是把协议树按层分成相应的可编程单元 PU,PU 分上下两个出口和两个入口,类似于传统网络计算或功能的放置,一次性可编程的最大范围是本层内的所有协议,可编程单元之间的二元关系为

$$R=\{(PU_i, PU_{i+1}) \mid i\in \mathrm{N}\}.$$

这里,N 为自然数,PU_i 代表第 i 个可编程单元.对包流的处理过程形成两条路径,连接 $n(n\in \mathrm{N})$个 PU 的上出口和下入口得到路径

$$P_{\mathrm{up}}=(\langle PU_0, PU_1\rangle,\langle PU_1, PU_2\rangle,\langle PU_2, PU_3\rangle,\ldots,\langle PU_{n-1}, PU_n\rangle);$$

连接下出口和上入口得到路径

$$P_{\mathrm{down}}=(\langle PU_{n-1}, PU_n\rangle,\langle PU_{n-2}, PU_{n-1}\rangle,\ldots,\langle PU_1, PU_2\rangle,\langle PU_0, PU_1\rangle).$$

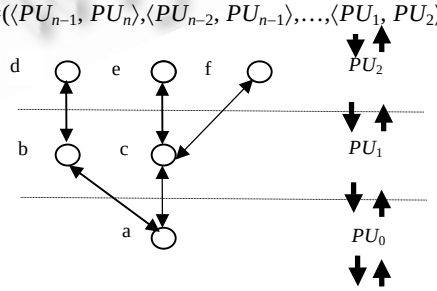


Fig.2 The network protocol tree and PU partition and connection

图 2 网络协议树及 PU 划分与连接

依据 P_{up},P_{down} 执行环境动态插入和连接可编程单元,对协议的处理是 PU 的嵌套过程,包头是对协议树的串行化描述,包的处理即为对协议树的处理,对从根结点来的包,向上处理嵌套为 $PU_2(PU_1(PU_0))$.

2.2 按协议划分可编程单元

计算机网络协议具有相对的独立性,把协议模块设置为一个 *PU* 要比按协议层次划分 *PU* 在粒度上更细,在编程上灵活性更大.图 3 为按协议划分的 *PU* 的连接图,圆圈为 *PU*,分两个入口和两个出口,虚线为连接线,*PU* 和协议是一一对应的,呈树型.对如图 2 所示的子树 a[b[d]]对应的 *PU* 子树为 $PU_0[PU_1[PU_3]]$,上出口和下入口是一对多连接.

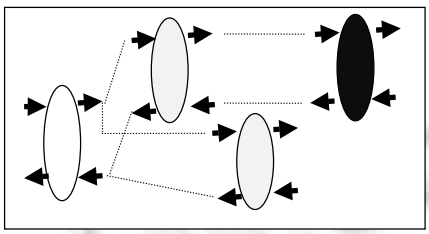


Fig.3 PU connection

图 3 PU 连接图

按这种划分方式,单个 PU 可加强目前网络设备中的部分协议,如网管、路由、可靠传输等协议.

2.3 按协议子树划分可编程处理单元

按协议的处理路径把图 1 分成 a[b[d]],a[c[e]],a[c[f]]三棵子树,I_a 代表协议 a 的标识,在包中,协议标识和标识位置是不可分的,用 A_i 代表第 *i* 层协议标识位置值,$\{A_1=I_a,A_2=I_b,A_3=I_d\}$代表子树 a[b[d]]的协议子树标识(protocol-subtree identification,简称 PI).

协议子树的特点和划分原则如下:

(1) 协议子树之间没有公共的叶子节点;

(2) 协议子树有共同的根结点,根结点为协议树的根结点;

(3) 协议子树可以有共同的分枝点;

(4) 协议子树可以有两个以上的 PI.

这里,每棵子树定义为一个 *PU*,a[c[e]]和 a[c[f]]都是协议子树 a[c[e,f]]的子树,所以 a[c[e,f]]可以有 $\{A_1=I_a,A_2=I_c,A_3=I_e\}$和$\{A_1=I_a,A_2=I_c,A_3=I_f\}$两个 *PI*,但 a[c[e,f]]和 a[c[e]]不能分为两个 *PU* 而在同一个执行环境里存在.

定义 **1**. 设协议子树标识 $P=\{PI_0,PI_1,\ldots,PI_n\}$,*PU* 标识 $D=\{DI_0,DI_1,\ldots,DI_m\}$ (*n*,*m* 为自然数,对任意一个 $PI_i\in P$,存在唯一的一个 $DI_j\in D$ 与 PI_i 对应;对 $DI_j\in D$,除 PI_i 之外,可以存在 $PI_k\in P$ ($PI_k\neq PI_i$)与 DI_j 对应.

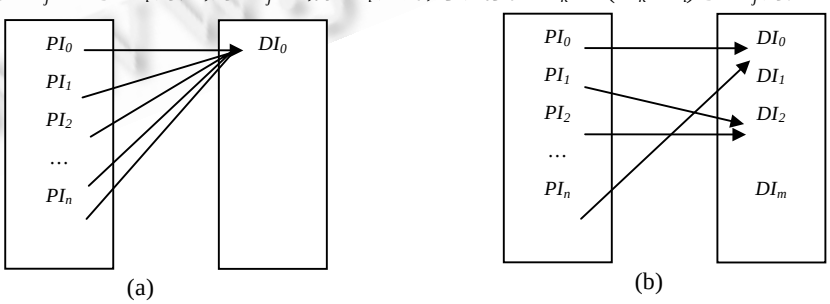


Fig.4 The relationship between protocol sub-tree IDs and PU IDs

图 4 协议子树标识和 PU 标识的对应关系

图 4 是协议子树标识和 PU 标识的对应关系,图 4(a)是传统网络协议协议子树标识和 PU 标识的对应关系,即所有的协议子树标识和一个 PU 标识 DI_0 对应.经过协议子树的划分,可以得到图 4(b)的对应关系,即多对一关系,我们按 PU 标识对图 4(b)进行分类得到

$$\{PI_0,PI_n\},\{PI_1,PI_2\},\ldots,$$

由于多对一的关系,每类之间没有公共元素.

图 5 是 PU 在主动节点上根据图 4(b)对数据包流的处理,其中 3 个圆圈分别对应着 PU_0,PU_1,PU_2,每个 PU 处理对应协议子树的包流,互不交叉.

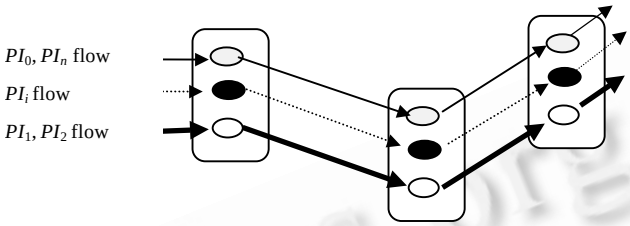


Fig.5 PUs process packet flows based on PIs

图 5 PU 按 PI 处理包流

2.4 可扩充性

主动网络的最大特点是动态可扩充性,假设 PU_1 的功能为$\{f_1,f_2\}$,PU_2 功能为$\{f_4,f_5\}$,如果扩充的新服务功能需要用到第 1 层中功能 f_1 和 f_3,第 2 层中功能 f_6,采用层次划分的扩展方法为

$$PU_1:\{f_1, f_2\} \longrightarrow \{f_1, f_2, f_3\};PU_2:\{f_4, f_5\} \longrightarrow \{f_4, f_5, f_6\},$$

需要更新 PU_1 和 PU_2,更新意味着一次 PU 的删除和增加过程,在运行系统中删除 PU 将导致系统的中断,特别是采用此类网络来测试新协议时是不可取的,按协议划分 PU 也存在同样的问题.采用按协议子树划分 PU 的扩展方法为

$$PU_1:\{f_1, f_2\}, \quad PU_2:\{f_4, f_5\}, \quad PU_3:\{f_1, f_3, f_6\},$$

按协议子树划分PU的系统,当扩充功能时,即使在每层都需增加功能,只需附加一个PU,这种扩展法不影响原来系统.

3 执行环境中的可编程移动软设备

3.1 移动软设备

在 Softnet 中,采用了按协议子树划分 PU 方式,由于 PU 中包含了从根结点到叶子结点的各层协议,可以独立成为一个网络设备,并且是可编程移动的,为了和传统网络设备区别,这里称为可编程移动软设备(softdevice).

主动网络把代码移动到分布的数据中执行,而不是把数据带给代码[9].移动代码可以放入一个主动包中,在经过的执行环境中执行,如 MIT 的 ANTS 项目,也可以预埋代码让经过的主动包执行,如 SwitchWare 项目中的 active extension,Softnet 采用在执行环境中预埋 Softdevice 的方式.

传统网络设备的功能可分成两部分,一部分是数据包流的处理功能,称为带内服务(in-band services),如 IP 包的转发;还有一部分是网络设备维护和服务功能,成为带外服务(out-band services),如路由、网管代理等.在 Softdevice 中定义如下 Java 语言描述的 3 种功能:

```
public abstract class Agent extends Object{
    public void entry(int from_interface, byte[] packet, int packet_length){};
    public void handleActivepkt(int from_interface, byte[] packet, int packet_length){};
    public void handlePassivepkt(int from_interface, byte[] packet, int packet_length){};
}
```

entry 函数提供各种带外服务,同时启动线程维护 Softdeivce,带内服务中,*handleActivepkt* 函数处理主动包流,*handlePassivepkt* 函数处理被动包流,任何一个 Softdevice 的编制都是重载上述 3 个函数.

3.2 Softdevice的寻址

Softdevice 的寻址是依赖于执行环境中的一个称为 Demux 的多路分解器来寻址的,Softdevice 代码的信息摘要值(代码 16 位 MD5 值)[11]称为设备标识 DI.所有的 Softdevice 都放入字典中,用 DI 唯一标识,为了兼顾到在这里被定义为纯数据的传统协议,采用协议标识 PI 来寻址正确处理该包的 Softdevice.首先在 Demux 中像表 1 一样注册 PI 和处理该协议的 DI,协议标识 PI 中的 home 代表包中 IP 地址为本节点中的 IP 地址,Demux 根据传统包的 PI 查询到 Softdevice 的 DI,最后从字典中来得到 Softdevice.对主动包,包内必须包含 Softdevice 的 DI,从而直接从字典中得到 Softdevice,所以不需 PI.

PI 和 DI 是多对一的关系,表一中 IP forwarding 和 IP filter 两个 Softdevice 是不相容的,它们只能在注册表中存在其中的一个,但多个协议可以由一个 PU 来处理,传统网络设备是一个特例,即所有的协议标识和一个 DI 对应.

Table 1 Softdevice registration table in Demux

表 **1** Demux 中 Softdevice 注册表

Softdevice①	Protocol identification②	Device identification③
ARP (address resolution protocol)④	08,06:home	7b,4c,73,1c,44,eb,9d,22,93,04,fa,e6,ea,1d,96,8b
RIP (routing information protocol)⑤	08,00:17:520:home	90,2c,76,d9,55,61,34,59,01,0a,46,8e,a0,bd,3f,51
IP forwarding	08,00	cf,be,a8,ab,40,ed,20,a7,c8,e6,e6,da,d5,84,f4,e8
IP Filter	08,00	49,14,67,14,64,93,fc,cf,30,d2,74,b8,24,11,2d,f0
Active-packet Handler1	–	c2,ec,88,cf,8c,7c,d1,33,d6,37,78,e6,d6,e1,37,c5

①软设备体,②协议标识,③设备标识(代码的 16 位 MD5 值),④地址解决协议,⑤路由信息协议.

4 代码的引用

4.1 代码的分发和下载

移动代码是主动网络的血液,Softdevice 中的代码可以来自于应用程序或用户采用主动包分发的代码,由执行环境组装和加载代码.Softdevice 也可以采用动态连接的代码,执行环境在 Demux 中注册了 Softdevice 的 DI 和 URL,当某一包需要该设备处理时,Demux 要求执行环境从 URL 中采用协议(如 HTTP,FTP 等)下载该 Softdevice 代码,或主动包要求执行环境下载离节点最近的服务器上 Softdevice 代码来处理,这和 Java 的动态加载相符,又和主动网中找代码处理数据的方式一致.表 2 是 Softdevice 的 Java 代码被分发的时间和 Softdevice 处理包的性能.

Table 2 Softdevice Java code dispatching time and performance

表 **2** Softdevice 的 Java 代码分发时间和性能

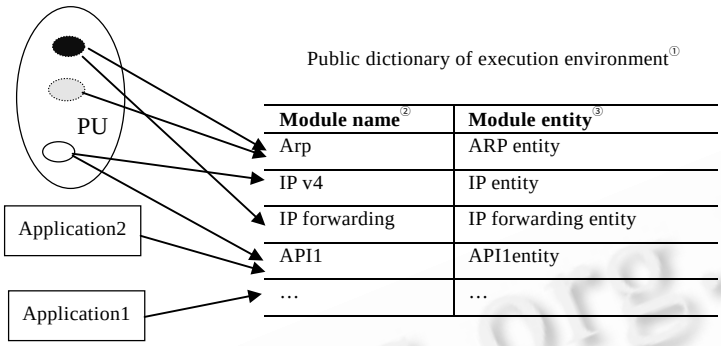
Softdevice①	Code quantity② (Byte)	Distribution duration③ (ms)	Softdevice's processing speed④ (Mpps)	Node's processing speed⑤(Kpps)
ARP	1,224	120,5	4.1	3.8
RIP	12,048	181,0	3.2	3.2
IP forwarding	14,246	193,4	2.8	2.9
Active-packet Handler	13,564	184,5	2.4	2.8

①软设备,②代码量,③分发时间,④Sofdevice 的包处理速度,⑤节点的包处理速度.

4.2 执行环境API和共享模块的管理

执行环境中有 Softdevice 和系统 API(application programming interface)两个字典,Softdevice 字典在 Demux 中,API 字典由系统管理,使用 API 的好处是,一方面节约编程工作量,另一方面可以节省内存空间,防止代码重复

和内存泄露.图 6 是代码使用 API 字典的过程,使用共享模块的过程,表中左栏是模块索引,右边是模块实体.执行环境是各种应用的运行平台,系统的 API 和常用的共享模块需要向执行环境注册,执行环境以它们的名字作为索引把它们放入字典,其它模块按名字存取字典中的方法和数据,有利于用户或应用程序编程.



①执行环境的公共字典,②模块名,③模块实体.

Fig.6 Applications and PUs call APIs and shared modules

图 6 应用程序和 PU 使用 API 与共享模块

下面是按名引用 IPv4 模块的一段 Java 代码:

IPv4 ipv4=(*IPv*4)*API.Dictionary.get*("*IPv*4");

If (*ipv*4!=*null*)*ipv*4.*forward*(*from_interface*, *packet*, *packet_len*).

4.3 应用程序

应用程序是处理应用层协议的 PU,采用传统协议的应用程序首先注册 PI 到 Demux 中,包括协议类型、端口等,如果需要多个协议支持,可以注册多个 PI,也可根据程序动态注册,,Demux 把满足要求的包送交应用程序处理.采用主动协议的应用程序由主动包来标识查找.在执行环境中,可以存在多个应用程序嵌入到系统中同时执行,经过多线程接受用户请求,Demux 通过下面的 Java 代码提交数据包到应用程序:

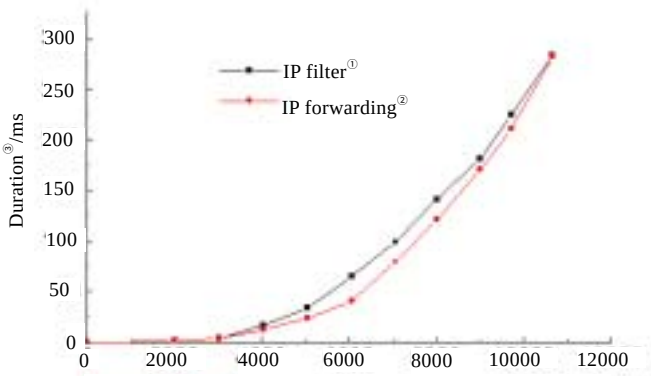
Agent application = (*Agent*) *AGENT.Dictionary.Get*(*DI*);

application.HandlePassivepkt (*from_interface*, *packet*, *packet_len*).

5 相关工作

课题组采用 Java 语言和一些本地方法实现了执行环境,形成了较多的 API,软设备采用移动代理实现,主动节点的动态过滤、移动 IP 等动能的扩充都非常容易实现,扩展的移动 IP 协议采用了 Softnet 中虚电路交换方式,避免了三角通信,提高了效率.Softnet 软件包括 MAC 驱动程序、节点操作系统(Windows/Windows NT)、执行环境、外壳(shell)、移动代理和应用程序.图 7 是带 Softdevice 的主动节点(200Mhz 主频)在 10Mbps Ethernet 上的时延情况,图 8 是 Softdevice 在主动节点中的丢包情况,从图可知,当周遍网络达到 6000pps 时,主动节点的性能有明显的降低,受环境影响很大,另一方面采用部分本地方法的 IP forwarding 的模块性能要好一点.

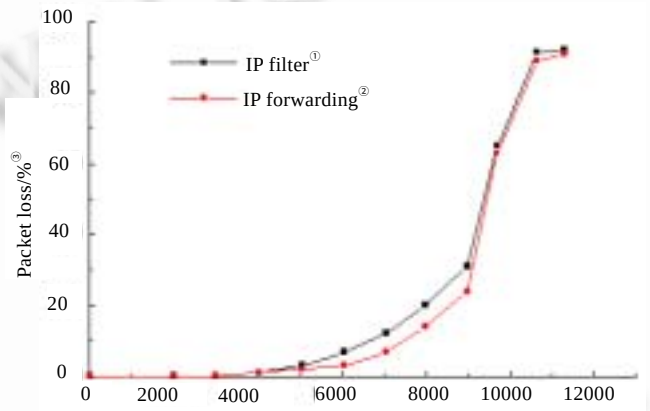
在 Java 代码效率还不高的前提下,提高主动节点性能的两个条件是值得考虑的,一是常用功能、执行环境提供的 API 尽量采用本地方法,应采用高性能代码编制,有条件可以采用硬件实现;二是在 Java 字节码执行前进行编译.



①IP 过滤器,②IP 转发器,③时延,④通过主动的通信量.

Fig.7 Variation in duration for Softdevice in active node to process one packet

图 7 Softdevice 在主动节点中处理包的时间间隔的动态变化



①IP 过滤器,②IP 转发器,③丢包率,④通过主动的通信量.

Fig.8 Variation in packet loss incurred by Softdevice in the active node

图 8 Softdevice 在主动节点中丢包动态变化情况

6 结 论

主动网络不但需要代码的移动,更重要的是良好的执行环境,在主动网络 Softnet 的执行环境里,主动节点的功能是易于扩充的,用户对主动节点的编程是方便的,用户通过 API 能自制软设备,分发并控制设备,隔离现有正常应用,有利于调试,试验新协议新应用,为协议的快速部署和推广提供基础.当然可以看到,主动网络的灵活性也付出了很多性能的代价,还需要进一步完善.可喜的是,目前主动网络已在智能网管理中使用已获得成功,网络操作系统开发者已开始支持移动代码.

致谢 在论文形成和相关课题研究中,得到了西安交通大学新型计算机研究所的伍卫国老师、博士生张文杰、博士生栾钟治等人的帮助,在此一并表示感谢.

References:

[1] Calvert, S. K., Zegura, E. An architecture for active networking, 1997. ftp://ftp.tns.lcs.mit.edu/pub/papers/ieeecomms97.ps.gz.

[2] Tennenhouse, D.L., Wettherall, D.J. Towards an active network architecture. ACM SIGCOMM Computer Communication Review, 1996,26(2):5~18.

[3] Tennenhouse, D.L., Smith, J.M., Sincoskie, W.D., *et al*. A survey of active network research. IEEE Communication Magazine, 1997,35(1):80~86.

[4] Calderon, M., Sedano, M., Azcorra, A., *et al*. Active network support for multicast. IEEE Network, 1998,12(3):46~52.

[5] Alexander, D.S., Arbaugh, W.A., Hicks, M.W., *et al*. The switchware active network architecture. IEEE Network, 1998,12(3): 12~18.

[6] Saltzer, H., Reed, D., Clark, D. End-to-End arguments in system design. ACM Transactions on Computer System, 1984,2(4): 277~288.

[7] Bhattacharjee, S., Calvert, K., Zegura, E. Active networking and end-to-end argument, 1997. http:// www.ccgetech.edu/projects /canes/ papers/ane2e.ps.gz.

[8] Yemini, Y., Silva, S.D. Towards programmable network, 1996. http://www.cs.columbia.edu/dcc/netscript/Publications /dsom96.ps.

[9] Norman, C.H, Lary, L.P. The x-kernel: an architecture for implementing network protocols. IEEE Transactions on Software Engineering, 1991,17(1):64~75.

[10] Reed, D.P., Saltzer, J.H., Clark, D.D., *et al*. Commentaries on active networking and end-to-end arguments. IEEE Network, 1998, 12(3):66~71.

[11] Rivest, R. The MD5 message-digest algorithm, 1996. ftp://ftp.xanet.edu.cn/internet/rfc/rfc0000/ rfc1312.txt, 1992.

Execution Environments for Active Network Based on Programmable Mobile Softdevices *

LU Yue-ming, QIAN De-pei, XU Bin, WANG Lei

(*Neo-Computer Institute, Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an* 710049, *China*)

E-mail: ueming@xjtu.edu.cn; depeiq@xjtu.edu.cn

http://www.xjtu.edu.cn

Abstract: Execution environments (EEs) are infrastructures on which applications and mobile code are programmed, managed and executed. A new type of EE for active network based on programmable mobile softdevices is presented in this paper. Programmable units, called softdevices in the EEs, are formed according to functions of protocol sub-trees. By means of demux (demultiplexer), passive packets and active packets can find and execute corresponding codes. In these EEs, functions of active nodes are extended flexibly, user programming is facilitated by rich APIs, and protocols or applications are tested and deployed dynamically.

Key words: active network; execution environment; softdevice; protocol sub-tree; programmable unit

* Received March 30, 2000; accepted August 1, 2000

Supported by the National Natural Science Foundation of China under Grant No.69973036; the National High Technology Development 863 Program of China under Grant No.863-317-01-10-99; the National Grand Fundamental Research 973 Program of China under Grant No.G1999032710