doi:10.3969/j.issn.1000-436x.2013.06.019

面向深度分组检测的高速数据分组解析结构

董永吉[1]，郭云飞[1,2]，黄万伟[1]，黄慧群[1]
（1. 国家数字交换系统工程技术研究中心，河南 郑州 450002；2. 解放军理工大学 指挥信息学院，江苏 南京 210007）

摘 要：提出了一种面向深度分组检测的高速数据分组解析结构 BiPPCS（bidirectional packet parsing architecture for content security）。结构采用内容萃取树描述协议的耦合关系从而提高了数据分组解析的灵活性；利用硬件双向并行流水线提升了数据分组解析的处理速率；通过使用节点映射算法来均衡各级流水线上的节点数目优化存储空间；分析和仿真显示 BiPPCS 在处理速率、空间利用率等方面能取得较好的均衡。

关键词：数据分组解析；深度分组检测；二叉 trie 树；网络安全；可重构；NetFPGA
中图分类号：TP393 **文献标识码**：B **文章编号**：1000-436X(2013)06-0156-09

Deep packet inspection oriented high speed packet parsing architecture

DONG Yong-ji[1], GUO Yun-fei[1,2], HUANG Wan-wei[1], HUANG Hui-qun[1]
(1. National Digital Switching System Engineering Technological R&D Center, Zhengzhou 450002, China;
2. College of Command Information Systems, the PLA University of Science & Technology, Nanjing 210007, China)

Abstract: A deep packet inspection oriented high speed packet parsing architecture called BiPPCS (bidirectional packet parsing architecture for content security) was proposed. Firstly, the content extraction tree was used to describe the coupling of the protocol relationship to improve flexibility of the packet parsing. Secondly, hardware bi-directional parallel pipeline was used to enhance the processing rate of the packet parsing. Thirdly, a node mapping algorithm was used to balance the number of nodes on all pipeline stages to optimize the storage space. Analysis and simulation show that BiPPCS gets balance among the rate processing, resource consumption and other aspects.

Key words: packet parsing; deep packet inspection; binary trie; network security; reconfiguration; NetFPGA

1 引言

随着互联网技术的发展，各种网络蠕虫、僵尸网络和计算机病毒等新型攻击不断涌现，严重威胁和破坏互联网络的安全[1]，深度分组检测(DPI, deep packet inspection)[2]作为网络入侵检测与防御系统的核心，对网络安全及网络监控越来越重要[3]。数据分组解析为网络安全检测提供数据分组头部及数据分组有效载荷信息，是深度分组检测重要的基础和前提[4]。灵活精准的解析数据分组不仅能有效地提高检测效率，而且还能提升检测结果的准确性，为可重构基础网络的安全和管控提供良好的支撑。

深度分组检测设备通常部署在高速网络的关键路径，在对海量高速的数据分组进行协议解析的基础上，针对协议分组头及内容负载进行检查。由于基于软件的数据分组解析结构实现方式难以适应网络高速处理，所以当前高速数据分组解析结构主要基于硬件实现。根据硬件实现所依据硬件平台的差异，结构可以分为：基于 ASIC(application specific integrated circuits)[5]、基于网络处理器(network processor)[6]和基于 FPGA(field programmable gate arrays)[7]这 3 类方式。ASIC 芯片功能结

收稿日期：2012-05-16；**修回日期**：2012-09-05
基金项目：国家重点基础研究发展计划（“973”计划）基金资助项目（2012CB315901）；国家高技术研究发展计划（“863”计划）基金资助项目（2011AA01A103）；国家科技支撑计划基金资助项目（2011BAH19B01）
Foundation Items：The National Basic Research Program of China(973 Program)(2012CB315901)；The National High Technology R&D Program of China(863 Program)(2011AA01A103)；The National Key Technology R&D Program of China (2011BAH19B01)

构稳定，因而具备较快的处理速度，但限于 ASIC 芯片设计完成后不能再升级改动，因而协议解析处理不具备可扩展的能力，无法支持新的协议和技术，同时 ASIC 芯片设计和生产的周期较长，致使新技术部署远滞后于网络研究[8]；基于网络处理器的方法在达到 100 Gbit/s 处理速率的同时，也具备较好的灵活扩展能力，但由于 NP 不适合进行正则表达式[9]匹配运算，导致通常需要搭配专用芯片（如 ASIC 或 FPGA）来辅助完成内容安全的检测；相对于前 2 种方式，FPGA 因具有并行性好，易配置和高效性的特点，在深度数据分组检测中得到广泛的应用，文献[10]基于可扩展标记语言产生了一个协议解析的状态机，具备较好的协议扩展能力的同时可以达到 20 Gbit/s 协议解析处理速率，但在处理协议跳转较多时，处理速率会急剧下降；Kangaroo 结构[11]基于 TCAM 和 HASH 联合查表的方法，能够灵活线速地解析 40 Gbit/s 数据分组，但由于采用了 TCAM，导致结构功率较大；文献[12]提出了数据分组解析语言 Packet Parsing，可以生成一个在线实时更新协议解析能力的处理器，达到 400 Gbit/s 的处理能力，但由于结构所消耗的硬件资源较多，限制了其应用的场景。

为提高深度分组检测过程中的匹配效率，本文结合双向流水线设计和二叉 trie 树查表的思想，提出了一种面向深度分组检测的协议解析结构 BiPPCS，通过在双向流水线上为每个接口定制内容萃取树（CE-trie, content extraction-trie），实现协议解析的高性能和可扩展，并采用 NetFPGA-10G 实验平台[13]对结构的可行性进行了仿真验证。

2　BiPPCS 结构介绍

BiPPCS 由 CE-trie 树和双向流水线 2 部分构成，其中，CE-trie 树用于描述结构支持解析协议的能力，双向流水线用于承载并查找 CE-trie 树的节点，两者最终由映射算法进行关联映射，实现 2 个部分的耦合，进而完成结构的协议解析功能，如图 1 所示。

2.1　内容萃取树

CE-trie 树用来描述应用于深度分组检测的协议解析的层次包含关系，以解决传统硬件解析结构中协议解析可扩展性差的问题。CE-trie 树基于二叉 trie 树将待查找的关键词构造成检索树的原理，对协议栈中网络层和传输层协议的数据分组头部以深度分组检测为目的进行解析和萃取，将数据分组头部的相应协议字段转化成为待查找的关键词，结合二叉 trie 树的组织方式，将协议解析过程转换成一个有序的查找序列，根据查找 CE-trie 树的实现对数据分组内容负载的识别和协议信息的萃取，并通过将 CE-trie 树节点统一化的标准格式，实现不同协议节点之间的松耦合，可以根据业务需求自定义解析的能力，提升了解析的灵活性和可扩展性。

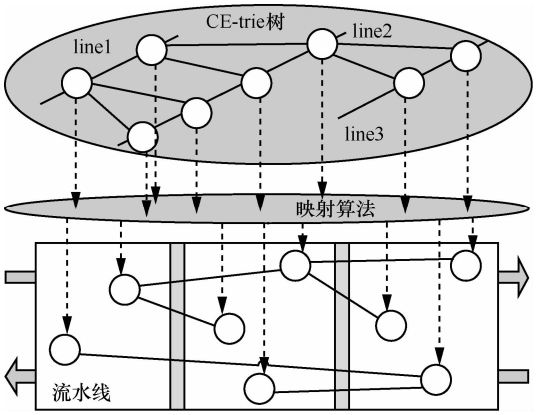


图 1　BiPPCS 结构示意

下面给出 CE-trie 树的定义。

定义 1　节点深度是该节点到根节点的最大距离，根节点的深度为 1，其余节点的深度为双亲节点深度的最大值加 1；CE-trie 树中最大的节点深度称为 CE-trie 树的深度。

定义 2　包含协议的定位规则的非叶子节点称为协议判定节点，包含协议的萃取信息的非叶子节点称为协议萃取节点。

定义 3　叶子节点只能包含解析的结果，其中包含正常解析内容的称为实叶子节点，反之包含无法解析的称为虚叶子节点。

定义 4　若二叉 trie 树满足如下的特征。

条件 1：每个非叶子节点最多只能判定一个协议的定位规则，而每个协议的识别判定可由多个定位规则联合组成。

条件 2：每个非叶子节点最多只能萃取一个协议所包含的相关字段。

条件 3：由于协议萃取节点不存在规则的判定，所以规定协议萃取节点只有左子树。

则称该二叉 trie 树为 CE-trie 树。

CE-trie 树结构中每个协议判定节点都包含一个协议判定的规则方法和 2 个指针，协议判定的规则方法用于协议的识别，而 2 个指针分别包括子节

点的位置及萃取下一个节点的比特偏移信息；协议萃取节点中包含要萃取的信息的指示；实叶子节点用于指示解析后负载的起始偏移；虚叶子节点用于指示协议无法识别。

图 2 中分别描述了 2 棵协议树 802.3 SNAP→IPv4→UDP&TCP 和 Ethernet→MPLS→MPLS→MPLS→IPv4→UDP 的 CE-trie 树表示。其中，P11 为 ETH II；P0、P1 为 802.3；P13、P14、P15 为 MPLS；P2 为 IPv4；P7 为 TCP；P4、P17 为 UDP 的协议判定节点；P3、P16 为 IPv4，P5、P18 为 UDP，P8 为 TCP 的协议萃取节点；P9 为 TCP，P6、P19 为 UDP 协议的实叶子节点；P10 和 P12 为虚叶子节点。

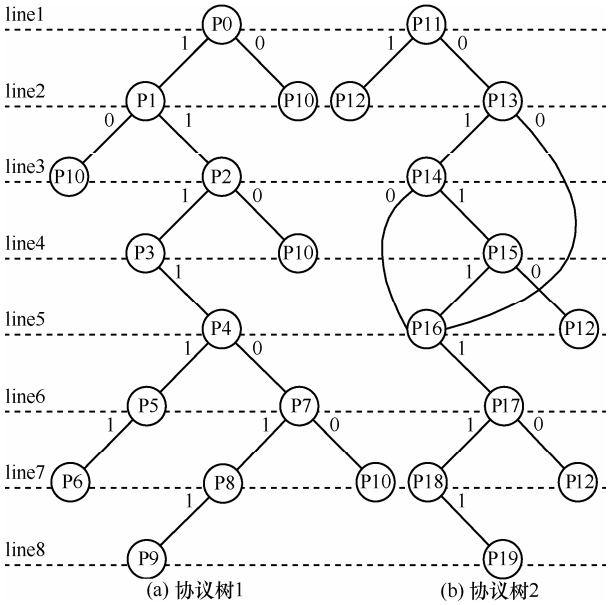


图 2 CE-trie 树结构示例

定义 5 解析路径为从根节点到叶子节点的一个节点序列，序列中的节点满足前一节点是后一节点的父节点的关系。

令 $P=\{P_1,P_2,\cdots,P_N\}$ 作为 CE-trie 树中协议字段的解析路径集合，其中，N 为 CE-trie 树中叶子节点个数，$b_{i,k}$ 代表解析路径 P_i 中第 k 个节点的指针指示（左子树为 0，右子树为 1），n_i 为解析路径 P_i 上的节点个数，则 CE-trie 树中解析路径 P_i 可以表示为

$$v(P_i)=\sum_{k=1}^{n_i} b_{i,k}2^k \tag{1}$$

由上述的定义可知，CE-trie 树具有如下的性质。

性质 1 若 2 个解析路径 P_i 和 P_j 分别有：$v(P_i)=\sum_{k=1}^{n_i} b_{i,k}2^{-k}$ 和 $v(P_j)=\sum_{k=1}^{n_j} b_{j,k}2^{-k}$。如果 $v(P_i)=v(P_j)$，则必定有 $n_i=n_j$，且 $b_i=b_j$。其中，$b_i=[b_{i,1}, b_{i,2},\cdots, b_{i,n_i}]$，$b_j=[b_{j,1},b_{j,2},\cdots,b_{j,n_j}]$。

证明 已知 2 个 P_i 和 P_j 的长度 n_i 和 n_j 有 2 种关系，$n_i=n_j$ 或 $n_i\neq n_j$。首先证明 $n_i=n_j$ 情况下命题成立。

①当 $n_i=n_j$ 时，由 $v(P_i)=v(P_j)$ 可知

$$\sum_{k=1}^{n_i} b_{i,k}2^{-k}=\sum_{k=1}^{n_j} b_{j,k}2^{-k}$$

$$b_{i,1}2^{-1}+\sum_{k=2}^{n_i} b_{i,k}2^{-k}=b_{j,1}2^{-1}+\sum_{k=2}^{n_i} b_{j,k}2^{-k}$$

$$(b_{j,1}-b_{i,1})2^{-1}=\sum_{k=2}^{n_i}(b_{i,k}-b_{j,k})2^{-k}$$

若有正整数 N,M，且 $N<M$，则存在 $2^{-N}>2^{-N}-2^{-M}=\sum_{k=N+1}^{M}2^k\geqslant\sum_{k=N+1}^{M}\left|b_{i,k}\right|2^k$，$\left|b_{i,k}\right|\in\{0,1\}$。而 $b_{i,k}-b_{j,k}\in\{-1,0,1\}$，$\left|b_{i,1}-b_{j,1}\right|\in\{0,1\}$，由于 2^{-1} 与 $\sum_{k=2}^{n_i}(b_{i,k}-b_{j,k})2^{-k}$ 值域不同，因而当且仅当 $\sum_{k=1}^{n_i}\left|b_{i,k}-b_{j,k}\right|=0$ 时，等式 $2^{-1}+\sum_{k=2}^{n_i}(b_{i,k}-b_{j,k})2^{-k}=0$ 成立，所以得到 $b_i=b_j$。

②当 $n_i\neq n_j$ 时，假设 $n_i>n_j$

$$\begin{aligned}v(P_i)-v(P_j)&=\sum_{k=1}^{n_i}b_{i,k}2^{-k}-\sum_{k=1}^{n_j}b_{j,k}2^{-k}\\&=\sum_{k=1}^{n_j}b_{i,k}2^{-k}+\sum_{k=n_j+1}^{n_i}b_{i,k}2^{-k}-\sum_{k=1}^{n_j}b_{j,k}2^{-k}\\&=\sum_{k=1}^{n_j}(b_{i,k}-b_{j,k})2^{-k}+\sum_{k=n_j+1}^{n_i}b_{i,k}2^{-k}=0\end{aligned}$$

进而得到

$$\sum_{k=1}^{n_j}(b_{j,k}-b_{i,k})2^{-k}=\sum_{k=n_j+1}^{n_i}b_{i,k}2^{-k}$$

$$\left|\sum_{k=1}^{n_j}(b_{j,k}-b_{i,k})2^{-k}\right|=\left|\sum_{k=n_j+1}^{n_i}b_{i,k}2^{-k}\right|$$

$$\sum_{k=1}^{n_j}\left|b_{j,k}-b_{i,k}\right|2^{-k}=\sum_{k=n_j+1}^{n_i}\left|b_{i,k}\right|2^{-k}$$

由于 $2^{-n_j} > \sum_{k=n_j+1}^{n_i} |b_{i,k}| 2^k$，所以当且仅当 $\sum_{k=1}^{n_j} |b_{i,k} - b_{j,k}| = 0$ 且 $\sum_{k=n_j+1}^{n_i} b_{i,k} = 0$ 时，等式成立。而在等式成立的时候，解析路径 b_j 被包含在解析路径 b_i 中，但是每个解析路径都是从根节点到叶节点的一个路径，所以不应存在包含关系。故条件 $\sum_{k=1}^{n_j} |b_{i,k} - b_{j,k}| = 0$ 且 $\sum_{k=n_j+1}^{n_i} b_{i,k} = 0$ 无法同时成立，所以等式无法成立，进而得到在 $n_i \neq n_j$ 时，不存在 $v(P_i)=v(P_j)$ 的情况。

综上，当 $v(P_i) = v(P_j)$ 时，则必定 $n_i = n_j$，且 $b_i = b_j$。因而可以推知，CE-trie 树中任一个解析路径都是唯一的。

性质 2　深度为 L 的 CE-trie 树的节点数 $L \leqslant S_L \leqslant 2^L - 1$。

证明　用数学归纳法证明。

归纳基础：当 L=1 时，$1 = 2^1 - 1 = 1$，则当深度为 1 时，节点只有 1 个节点，因为深度为 1 的 CE-trie 树有且只有一个根节点，所以命题成立。

归纳假设：假设对所有的 $j(1 \leqslant j < L)$ 命题成立，即 $j \leqslant S_j \leqslant 2^j - 1$，证明 $j = L$ 时命题亦成立。

归纳步骤：根据归纳假设，深度为 $L-1$ 的 CE-trie 树的节点数 S_{L-1} 满足 $L-1 \leqslant S_{L-1} \leqslant 2^{L-1} - 1$。由条件 2 可知，深度为 L 的 CE-trie 树节点数至多比深度为 $L-1$ 的点数多 2^{L-1} 个，故即 $j = L$ 时，深度为 L 的 CE-trie 最多有节点 $2^{L-1} - 1 + 2^{L-1} = 2^L - 1$ 个，满足 $S_L \leqslant 2^L - 1$；由定义 1 可知，CE-trie 树深度为 L 意味着至少有一个节点的父节点的深度是 $L-1$，故 $L - 1 + 1 = L \leqslant S_L$，综上可知，当 $j = L$ 的时候，满足 $L \leqslant S_L \leqslant 2^L - 1$，故命题成立。

推论 1　节点数 S_L 一定的 CE-trie 树,深度满足 $\text{lb}(S_L+1) \leqslant L \leqslant S_L$。

证明　由性质 2 可知，$L \leqslant S_L$ 且 $S_L \leqslant 2^L - 1$，故 $S_L + 1 \leqslant 2^L$，进而 $\text{lb}(S_L+1) \leqslant L$，所以 $\text{lb}(S_L+1) \leqslant L \leqslant S_L$。

综上可知，若待解析的协议可以用 CE-trie 树表示，则生成的 CE-trie 树是唯一的，且节点数和深度都存在上下限，即实现 CE-trie 树节点占用的存储空间有限，表明 CE-trie 树是物理可实现的。其次，由 CE-trie 树的定义可知，CE-trie 树具有开放式的描述方式，可将协议解析关系由紧耦合转变为松耦合，极大扩展了协议描述的灵活性，方便新协议在解析过程中的添加，易于协议的 CE-trie 树的构建，加快了新协议的解析部署。

2.2　BiPPCS 双向流水结构

BiPPCS 的双向流水结构用于 CE-trie 树节点的存储，并通过流水查找 CE-trie 树节点的方式实现协议解析的功能，如图 3 所示，BiPPCS 的双向流水线结构主要由 3 个部分构成：节点的存储空间双通道 RAM、比较器和移位器。双通道的 RAM 具备在一个时钟周期内并发访问 2 个地址的能力，因而属于同一级的 2 条流水线可以利用不同通道访问同一块 RAM 进行节点查找，进而在每一级流水线上可以保证同时刻有 2 个数据分组进行查表操作。双通道特性的利用，使得数据分组解析算法在不增加 RAM 资源的前提下，提高了一倍的吞吐率。

比较器用于在数据分组头部萃取协议字段与 RAM 中存储的协议字段预定值进行逻辑比较，并根据逻辑判定的结果指定左/右子节点输出。

移位器用于实现数据分组信息的萃取，根据萃取信息的起始位置，并按照指定的长度，从起始位置提取连续的比特串作为结果输出。

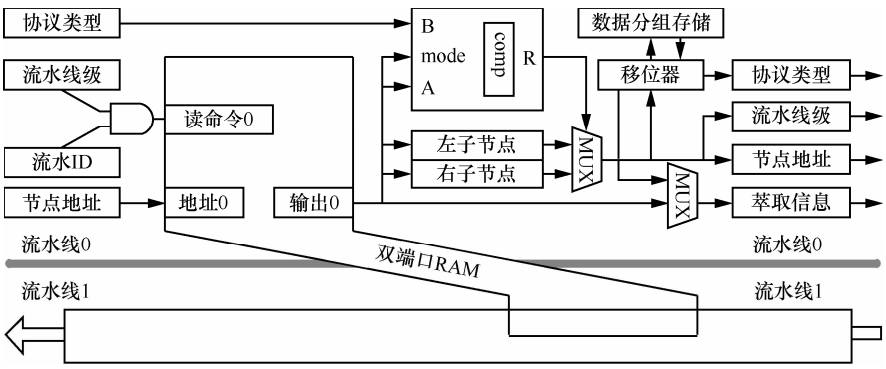


图 3　双向流水线结构

2.3 BiPPCS 工作原理

BiPPCS 结构为每个接口都建立一个定制的 CE-trie 树，因而每个接口都具备了灵活独立的解析功能，可以动态地满足深度分组检测对协议解析能力的需求。如图 4 所示，映射在流水线上以 P0 和 P11 为根节点的两棵 CE-trie 树，对应了 2 个独立的网络接口。每个到达结构的数据分组，依据接口的不同，从对应 CE-trie 树的根节点开始向叶子节点解析，若经历协议判定节点，则进行协议规则的判定，并选择匹配的子节点进行下一步查找；若经历协议萃取节点，则提取萃取信息，并选择左子节点进行下一步查找；若经历叶子节点，则输出最终匹配结果，并退出结构。

如果在流水线的最后一级匹配到叶子节点，则直接输出结果，若是在前几级流水线上匹配到叶子节点，则该数据分组仍需要等待未处理的流水线长的时间，用以保证数据分组的处理时延等长，进而达到结构对数据分组的保序处理。由于流水查找在每个阶段可以并行处理多个数据分组，一方面使每个时钟周期都能处理 2 个数据分组，另一方面降低了单个节点的复杂性，提高了数据分组解析的性能。

3 BiNMA 节点映射算法

在 BiPPCS 结构中，CE-trie 树节点需要被映射到双向流水线上来完成结构的构建。映射过程中，最简单的方式是按照 CE-trie 树中节点的深度，按照逐渐递增的顺序，依次将节点顺序分配到硬件流水级上，但是由 CE-trie 树的定义可知，CE-trie 树的中节点数目分布不规则，理论上深度低的节点数量相对要少于深度高的节点数，所以简单的映射方法会导致映射后存储空间被不均衡的占用，降低存储空间利用率，同时也会影响更新操作的效率，甚至影响系统整体的性能。针对 CE-trie 树映射过程中存储空间的均衡优化问题，提出双向映射的最优化的数学模型为

$$\min \max_{i=1,2,\cdots,H} G_i \tag{2}$$

其中，H 代表流水线的深度，G_i 代表了第 i 层流水线上的节点数

约束 1 父节点一定映射在映射方向上比子节点距根节点更近的流水线级。

约束 2

$$\sum_{i=1}^{H} G_i = \sum_{j=1}^{n} M_j \tag{3}$$

其中，n 代表接口对应的结构不同的 CE-trie 树的个数，M_j 代表第 j 个接口对应的 CE-trie 树的节点数量。

约束 3 CE-trie 树的根节点只能映射在映射方向上流水线级的第一级。

3.1 双向节点映射算法

针对简单的映射策略会导致流水线上节点存储的不均衡问题，提出了一种双向节点映射算法 BiNMA（bidirectional node mapping algorithm），算法将接口的 CE-trie 树初始化分配 2 个方向的映射等待队列中，并依次轮询 2 个等待队列，将 CE-trie 树的节点按照正反 2 个方向交叉的映射到流水线中，通过映射在正反 2 个方向上 CE-trie 树结构的差异互补，实现流水线上节点的均衡存储，具体流程步骤如图 5 所示。

1) 将 CE-tire 树集合 $\{T_1,T_2,\cdots,T_n\}$ 按照节点数目的多少降序排列得到排序后新的 CE-tire 树集合

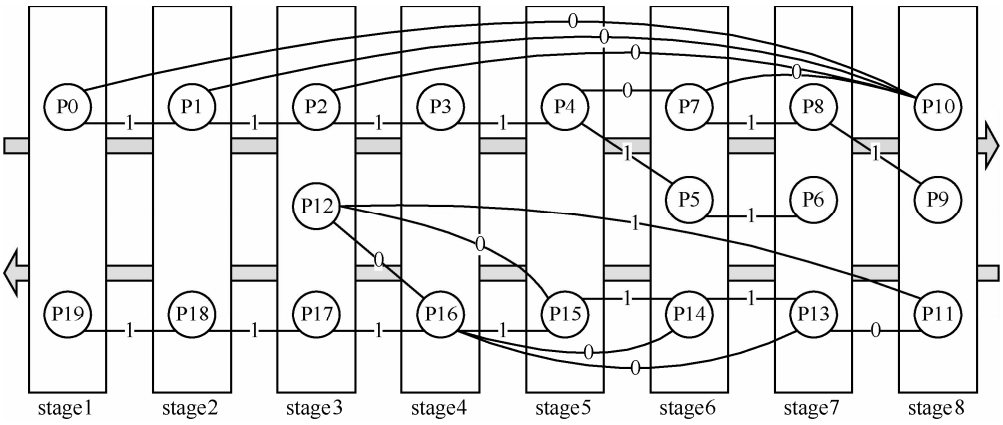


图 4 BiPPCS 流水查表

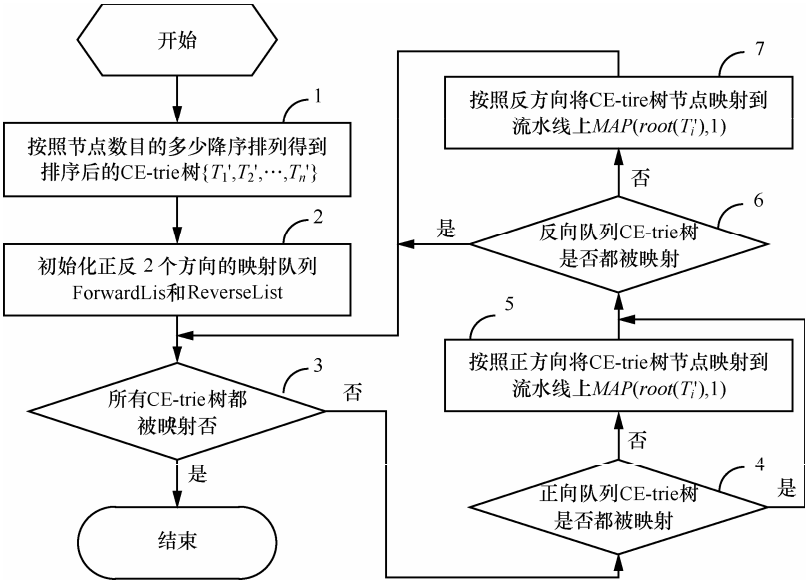


图 5　BiNMA 算法流程

$\{T_1^{'},T_2^{'},\cdots,T_n^{'}\}$，满足 $size(T_1^{'}) \geqslant size(T_2^{'}) \geqslant,\cdots,\geqslant size(T_n^{'})$ 的条件，其中，$size(\cdot)$的结果为 CE-trie 树的节点数目，n 代表接口对应结构不同 CE-trie 树个数。

2) 初始化正反 2 个方向的映射等待队列 ForwardList 和 ReverseList，图 6 给出了映射等待队列初始化流程：根据排序后的 CE-trie 树 $\{T_1',T_2',\cdots,T_n'\}$ 节点数的累加和，将 CE-trie 树均匀地分配到正反 2 个方向的映射等待队列当中，其中，ForwardList 为正向映射的等待队列，其映射方向为流水线级数的递增方向，ReverseList 为反向映射的等待队列，其映射方向为流水线级数的递减方向，G_F 和 G_R 分别为正向和反向映射的 CE-trie 树的节点总数。

```
1)   Initialize two lists: ForwardList←Φ and ReverseList←Φ;
2)   Initialize G_F and G_R:G_F←0 , G_R←0;
3)   for i=1 to n do
4)     if G_F≤G_R
5)       Assign T_i to ForwardList;
6)       G_F←G_F + size(T_i);
7)     else
8)       Assign T_i to ReverseList;
9)       G_R←G_R + size(T_i);
10)    end if
11)  end for
```

图 6　映射等待队列初始化流程

3) 以映射等待队列 ForwardList 和 ReverseList 是否同时为空为依据获得当前 CE-tire 树是否全部被映射，判断结果为真，表明所有 CE-tire 树都已经被成功的映射，则算法完成功能；否则转到步骤 4）。

4) 判断正向映射等待队列 ForwardList 是否同时为空，若为真表明所有正向等待队列中的 CE-tire 树都已经被成功的映射，则转到步骤 6），否则转步骤 5）进行正向等待队列中 CE-tire 树的映射。

5) 将当前正向映射等待队列 ForwardList 中的第一个 CE-trie 树 $\{T_i'|T_1',T_2',\cdots,T_n'\}$ 提取出，按照正向方向将其映射到流水线上，每个 CE-trie 树的节点映射顺序类似二叉树的前序遍历，依次为：①映射根节点；②前序遍历并映射左子树；③前序遍历并映射右子树。具体单个 CE-trie 树节点映射流程如图 7 所示。

```
MAP(CEtrie, CEdepth)
1)    if(CEtrie is not mapped) then
2)    if Condition(CEtrie)=TRUE or G_CEdepth≤G_ave then
3)        map CEtrie onto the CEdepth th pipeline;
4)        G_CEdepth ←G_CEdepth +1;
5)      else
6)    CEdepth ←CEdepth+1
7)        MAP(CEtrie, CEdepth);
8)      endif
9)    end if
10)   CEdepth ←CEdepth+1
11)   MAP(CEtrie→lchild, CEdepth);
12)   MAP(CEtrie→rchild, CEdepth);
```

图 7　单个 CE-trie 树节点映射流程

如图 7 所示，其中，Condition(·)为映射节点的条件判定函数，判断结果为真的节点必须映射到当前对应的流水线上，否则会影响后续的节点无法映射到流水线。G_{CEdepth} 代表应映射方向深度为 CEdepth 的流水线级上的已经映射的节点个数，由最优化的数学模型可知，$G_{\mathrm{ave}}=\left\lceil[\sum_{j=1}^{n}M_j-n]/\mathrm{H}\right\rceil$ 代表取整后的流水线级上节点数目理论平均值。

6) 查看反向映射等待队列 ReverseList 中的 CE-trie 树是否都被映射，若为真则转到步骤 3)，否则跳转至步骤 7)继续完成节点的映射。

7) 与步骤 5) 类似，将当前反向映射等待队列 ForwardList 中的第一个 CE-trie 树提取出，按照反向方向将其映射到流水线上，单个 CE-trie 树节点映射流程亦如图 7 所示。

3.2 节点更新方法

根据不同的网络业务的组网需求，BiPPCS 需要在流水线中动态调整 CE-trie 树来实现解析能力的灵活变更。如图 8 所示，BiPPCS 通过采用在流水线作业中通过插入 Write Bubble[14]的方法实现节点的实时更新。

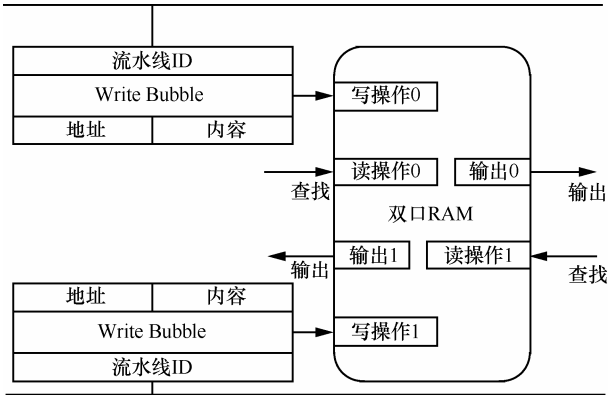


图 8 Write Bubble 更新方法

首先，需要离线算法计算得到更新的节点信息，其次，在每次进行更新时，根据流水级 ID 及节点地址匹配到流水线上需要更新的节点，并利用流水线上协议解析查找流水空隙，插入 Write Bubble 在流水线上完成更新。Write Bubble 的使用可以在不影响线速解析协议的情况下，快速更新实现节点信息。

4 仿真测试结果

为了验证 BiPPCS 结构的可行性，采用 NetFPGA-10 G 板卡上的 Xilinx Virtex-5 XCVTX240T FF1759-2 FPGA 进行仿真实验。该 FPGA 可用的逻辑资源为 18 720 可编程逻辑单元（configurable logic blocks）、2 400 kbit 分布式存储单元(distributed RAM)和 11 664k（324×36k）bit 块存储单元（block RAM），实验工具使用 Xilinx ISE 13.3 和 Synplify Pro E 2011-3 SP1 软件。

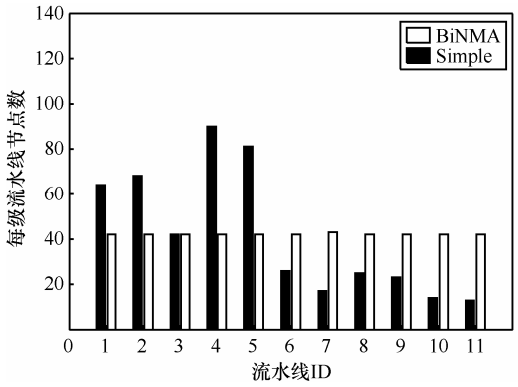
由于 NetFPGA-10G 板卡的接口类型被固定为 4×10 G，无法进行接口数量和链路速率上的升级扩展，为了验证 BiNMA 算法的有效性，在 FPGA 内部模拟搭建了一个 64 路接口的 BiPPCS 结构，并根据 Cisco 路由器支持的协议树，按照最大流水线深度为 14、13、12、11 这 4 种情况下，为每个接口均对应构建了独立的 CE-trie 树。

4.1 BiNMA 算法实验结果

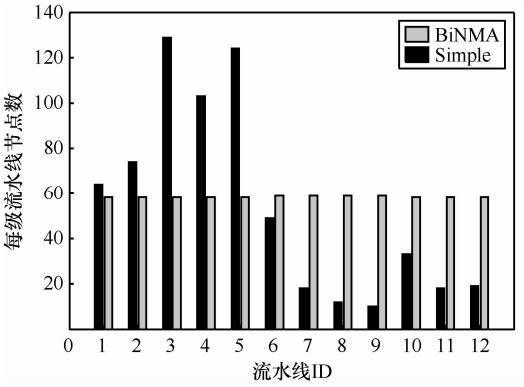
图 9 分别给出了 Simple 算法和 BiNMA 算法在流水线深度分别为 11、12、13 和 14 时，映射到流水线节点数目分布图。其中 Simple 算法采用节点深度直接对应映射的方式，将 CE-trie 树节点按照其深度对应映射到相同深度的流水线级。通过对比可以发现，由于 CE-trie 树形状的不规则，造成按照 Simple 算法映射到流水线级上的树节点分布极不均匀，不适用于高效的硬件流水查找操作，而相对于 Simple 算法，BiNMA 算法可以将 CE-trie 树节点均匀地映射到多级流水线上，提高了存储空间的使用率。

4.2 BiPPCS 性能实验结果

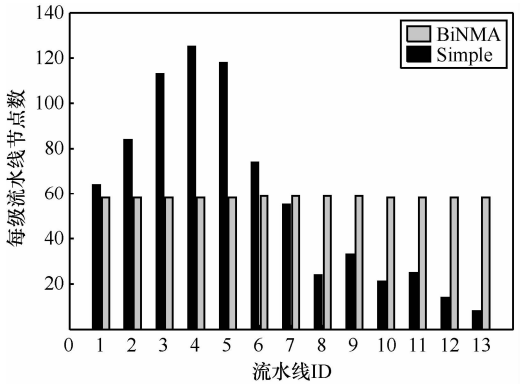
表 1 给出了基于 NetFPGA-10G 平台不同深度流水线的 BiPPCS 结构资源与性能对比，其中资源逻辑、BRAM 的占用和时钟频率来源于 ISE 工具的 Post place and route report 报告。观察可以发现，随着硬件流水线层数的递增，BiPPCS 结构中 BRAM 使用的个数逐渐递增，是由于硬件流水线采用 BRAM 资源搭建的，所以资源消耗会随着流水级数的增加而递增；而不同深度流水线对应的时钟频率会随着 BRAM 及逻辑资源的增多而减小，但即使是流水线深度达到 14 时，BiPPCS 也仅占用了不到 4%的片上逻辑资源，同时 BiPPCS 的时钟频率仍然可以达到 4.971 4 ns（201.15 MHz），由于本结构采用双通道 BRAM，可以支持 2 个流水查找过程并行进行，则对于最小的以太网帧（64 byte），吞吐量至少可以达到 201 MHz×64×8 bit×2 =206 Gbit/s。



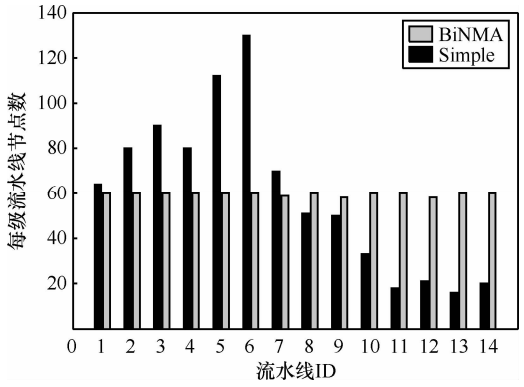
(a) 流水线深度为 11



(b) 流水线深度为 12



(c) 流水线深度为 13



(d) 流水线深度为 14

图 9　流水线节点数目对比

表 1　不同深度流水线的 BiPPCS 结构资源与性能对比

流水线深度	逻辑资源 (slice)	BRAM /36 kbit	时钟频率 /ns	吞吐量 /(Gbit·s^{-1})
6	10.4%	12	4.778 9	214
10	12.5%	20	4.914 6	208
14	14%	28	4.971 4	206

表 2 给出了 NetFPGA 实验平台下，当针对协议树 Ethernet→IPv4→TCP 解析时，PP、Kangaroo 和 BiPPCS 3 种协议解析结构在资源消耗和性能上的对比。在资源占用上，相对于 PP，BiPPCS 虽然占用了少量的 BRAM 资源，但节省 33.6%的片上逻辑资源；而相对于 Kangaroo，BiPPCS 在逻辑资源上多占用了 4%，但节约 12%的 BRAM 资源。在处理能力上，BiPPCS 的吞吐量远高于 Kangaroo 的 10 Gbit/s，但略低于 PP，由于 BiPPCS 结构为每个接口都建立专用的 CE-trie 树，所以不同于其他 2 种结构，可以为每个接口提供独立的解析能力。同时从资源占用上考虑，2 个并行的 BiPPCS 结构占用的资源仍远小于 PP 所占用的，但吞吐量却可以达到 428 Gbit/s，远远高于 PP 的协议处理能力。综上，BiPPCS 结构能在占用较小资源的情况下，提供较强的数据分组解析能力。

表 2　3 种解析结构资源和性能对比

结构	逻辑资源 (slice)	BRAM /36 kbit	吞吐量 /(Gbit·s^{-1})	接口独立的协议解析能力
BiPPCS	10.4%	3%	214	是
PP	44%	—	343	否
Kangaroo	6.4%	15%	10	否

5　结束语

本文针对于为高速网络深度分组检测灵活准确的萃取检测内容，提出了一种面向深度分组检测的数据分组解析结构 BiPPCS。通过结合双向流水线和 CE-trie 树达到高效和灵活的解析数据分组，采用平衡不同流水线级的节点数目以优化存储空间的占用，最后基于 NetFPGA-10G 实验平台仿真显示了 BiPPCS 在处理速率、空间利用率和资源消耗间取得了较好的均衡。

参考文献：

[1] 朱映映, 吴锦锋, 明仲.基于网络事件和深度协议分析的入侵检测研究[J]. 通信学报,2011,(8):171-178.
ZHU Y Y, WU J F, MING Z. Research on intrusion detection based on

network events and deep protocol analysis[J]. Journal on Communications,2011,32(8):171-178.

[2] LIAO M Y, LUO M Y, YANG C S, *et al*. Design and evaluation of deep packet inspection system: a case study[J]. IET Networks, 2012, 1(1):2-9.

[3] 朱国胜,余少华.基于 TCAM 的范围匹配方法——C-TCAM[J].通信学报,2012,33(1):31-37.
ZHU G S, YU S H. Range matching method based on TCAM: C-TCAM[J]. Journal on Communications,2012,33(1):31-37.

[4] PENG K, TANG S, DONG Q, *et al*. Chain-based DFA deflation for fast and scalable regular expression matching using TCAM[A]. Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems. Brooklyn[C]. NY, USA, 2011. 24-35.

[5] BHATNAGAR H. 高级 ASIC 芯片综合[M]. 北京: 清华大学出版社, 2007.
BHATNAGAR H. Advanced ASIC Chip Synthesis[M]. Beijing: Tsinghua Universit Press, 2007.

[6] AHMADI M, WONG S. Network processors: challenges and trends[A]. Proceedings of the 17th Annual Workshop on Circuits, Systems and Signal Processing[C]. The Netherlands, 2006. 223-232.

[7] Xilinx corporaton[EB/OL]. http://www.xilinx.com/,2012.

[8] ANWER M B, MOTIWALA M, TARIQ M B, *et al*. Switchblade: a platform for rapid deployment of network protocols on programmable hardware[A]. Proceedings of the ACM SIGCOMM Conference[C]. New Delhi, India, 2010. 183-194.

[9] BANDO M, SERTAC ARTAN N, CHAO H J. Scalable lookahead regular expression detection system for deep packet inspection[J]. IEEE/ACM Transactions on Networking, 2012.

[10] KOBIERSKY P, KORENEK J, POLCAK L. Packet header analysis and field extraction for multigigabit networks[A]. Proceedings of the IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems[C]. Liberec, Czech Republic, 2009. 96-101.

[11] KOZANITIS C, HUBER J, SINGH S, *et al*. Leaping multiple headers a single bound: wire-speed parsing using the Kangaroo system[A]. Proceedings of the 29th IEEE Conference on Computer Communications[C]. San Diego, CA, USA, 2010. 830-838.

[12] ATTIG M, GORDON B. 400 Gb/s programmable packet parsing on a single FPGA[A]. Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems[C]. Brooklyn, NY, USA, 2011. 12-23.

[13] KORCEK P, KOSAR V, ZADNIK M, *et al*. Hacking NetCOPE to run on NetFPGA-10G[A]. Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems[C]. Brooklyn, NY, USA, 2011. 217-218.

[14] BASU A, NARLIKAR G. Fast incremental updates for pipelined forwarding engines[A]. Proceedings of IEEE INFOCOM Conference[C]. San Francisco, CA, USA, 2003.64-74.

作者简介：

董永吉（1983-），男，吉林汪清人，国家数字交换系统工程技术研究中心博士生，主要研究方向为可重构基础网络、宽带信息网、网络安全等。

郭云飞（1964-），男，安徽桐城人，解放军理工大学教授，主要研究方向为可重构基础网络、宽带信息网等。

黄万伟（1979-），男，江苏盐城人，博士，主要研究方向为可重构计算。

黄慧群（1973-），女，江苏海门人，国家数字交换系统工程技术研究中心博士生，主要研究方向为路由交换与分发技术。