CRC 查询表及其并行矩阵生成方法

梁海华 盘丽娜 赵秀兰 李克清
(常熟理工学院计算机科学与工程学院 常熟 215500)

摘 要 循环冗余校验码(CRC)被广泛应用于通信领域。直接按位计算 CRC 校验值的方法难以满足高速链路要求,查询表方法与并行处理在很大程度上可以突破处理速度上的瓶颈。对查询表与并行处理矩阵之间的关系进行探讨,给出任意阶次生成多项式及任意处理位宽的查询表并行矩阵生成方法,并衍生出块处理方法。对表查询方法、并行矩阵查询方法及块处理方法的性能进行了比较分析,结果表明,并行处理位宽 w 增加,处理时间减少,并行矩阵方法在存储空间上表现更优;块处理通过减小校验序列长度,运算速度显著提高。

关键词 循环冗余校验码,位计算,查询表,并行处理矩阵,并行处理位宽,块处理
中图法分类号 TP301.6 **文献标识码** A

CRC Lookup-table and its Implementation by Parallel Matrix

LIANG Hai-hua PAN Li-na ZHAO Xiu-lan LI ke-qing
(Department of Computer Science and Technology, Changshu Institute of Technology, Changshu 215500, China)

Abstract Cyclic redundancy check (CRC) has already been used in the fields of communication widely. However, straightforward CRC implementation based on the bit cannot meet the requirements of high-speed link. By the lookup-table or parallel algorithm, the bottle-neck of speed can be resolved on a large scale. The relation between lookup-table and parallel matrix was investigated, from which lookup-table responding to polynomial of any order and any bit width processing can be given and deriving procedure of blocks. Comparison analysis on the performance of lookup-table, parallel matrix and blocks shows that less time is consumed while increasing width, and parallel matrix has better performance in requirement of memory space, and by reducing length of checking sequence, computational speed of blocks is increased significantly.

Keywords CRC, Bit calculation, Lookup-table, Parallel matrix, Parallel width, Blocks

1 引言

目前,在计算机通信领域,由于数据在传输或存储过程中会受到各种干扰而产生误码,因此需要进行数据校验来确保数据的完整性,循环冗余校验(CRC)是一种常用的方法[1]。CRC 编码是一种线性编码方式,通过线性移位寄存器(LFSR)来实现,该方法在计算机网络安全传输方面也被广泛使用[2]。CRC 校验值计算分两种:传统的按位计算 CRC 校验值和并行处理算法[3]。按位运算无法满足现有高速链路的需求,基于矩阵处理的并行 CRC 硬件电路被已经广泛采用[4,5],而软件方法实现更多地基于查询表方法(表驱动算法)。文献[6]中只讨论了并行处理位度小于等于生成多项式最高次幂的情况;文献[4]给出了并行处理长度大于生成多项式最高次幂的公式推导,但并未给出查询表与并行矩阵之间的关系。本文就任意并行处理位度的并行矩阵处理与查询表方法之间的关系进行探讨;分析 CRC 并行处理位宽 w 对算法性能的影响;对查询表方法、并行矩阵方法及块处理进行比较分析。

2 按位计算、查询表算法

首先给出假设:需要处理 $n+1$ 位数据,生成多项式的最高次幂是 m(长度为 $m+1$),校验序列长度为 m,并行处理长度为 w。

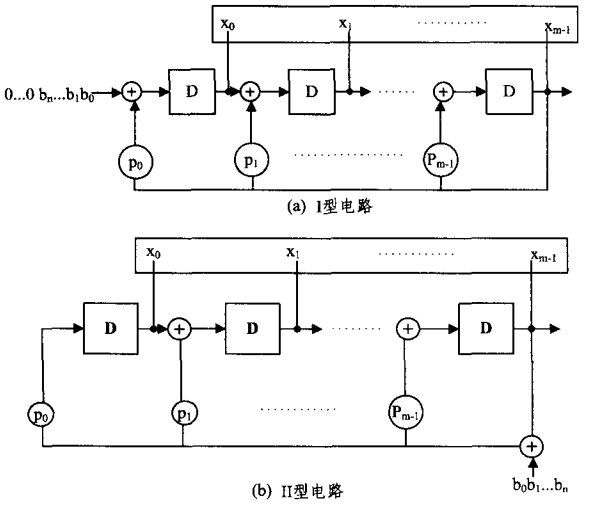

图 1 CRC 校验值按位生成电路

Peterson 等人于 1961 年给出了两种按位 CRC 计算电路的实现方法[7](见图 1),I 型电路较直观,但需要在输入数据

本文受苏州市 2010 年科技计划项目(SYND201002),常熟理工学院青年教师科研启动项目(KYZ2010136Z)资助。
梁海华(1981—),男,硕士,讲师,主要研究方向为计算机数据通信、网络安全,E-mail:edward_3me@163.com。

位后填充 m 位零；II 型电路处理无需填充零。本文重点以 I 型电路为基础进行探讨，$(x_0, x_1 \cdots, x_{m-1})$ 为寄存器的状态值，$(b_0, b_1 \cdots, b_n)$ 为输入序列，$(p_0, p_1 \cdots, p_{m-1}, p_m)$ 为生成多项式的低次项到高次项的系数。

根据文献[3]，查询表方法(表驱动算法)如图 1(a)、图 2 所示，描述如下：通过将每次处理位数(bit)由 1 扩大为 w，来提高 CRC 计算的运算速度；w 位二进制数位于 buff 的高位，每次先检查状态寄存器 buff 的最高位是否为 1：如果为 1，则将 $(p_m, p_{m-1} \cdots, p_1, p_0)$ 与 buff 进行异或操作；如果为 0，则不做任何操作，然后将 buff 左移一位。经过 w 次处理，新 buff 为原 buff 的查询值，原 buff 中 w 位数对后续二进制数或寄存器状态值的影响最终被保存在新 buff 中，2^w 种二进制序列与对应的影响值形成查询表 LUT(Look-up Table)，最终流程如图 2 所示。代码如下：

```
%w 位并行处理结果(共 2^w 种)保存在寄存器 buff 的高 m 位即为 LTU
While(w! =0)   If(buff>>(length(buff)-1))! =0
      % XOR 运算左对齐，POLY 为(p_{m-1}…p_0)
  buff =(buff <<1) XOR POLY
  else buff =<<1;
  w--;
End
```

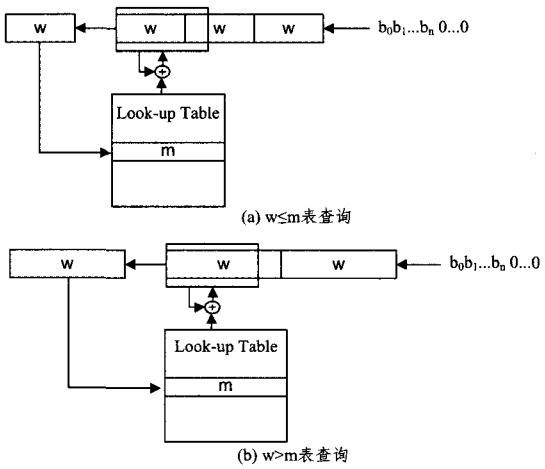

图 2　I 型电路表查询的处理流程

3　I 型电路并行矩阵算法

如图 1(a)所示，写出串行运算的系统状态方程组 $\begin{cases} x_{m-1}(t+1)=x_{m-1}(t)\otimes p_{m-1}\oplus x_{m-2}(t) \\ \cdots \\ x_1(t+1)=x_{m-1}(t)\otimes p_1 \oplus x_0(t) \\ x_0(t+1)=x_{m-1}(t)\otimes p_0 \oplus b(t) \end{cases}$，可得：

$$\begin{bmatrix} x_{m-1}(t+1) \\ x_{m-2}(t+1) \\ \cdots \\ x_1(t+1) \\ x_0(t+1) \end{bmatrix} = \begin{bmatrix} p_{m-1} & 1 & 0 & \cdots & 0 \\ p_{m-2} & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_1 & 0 & 0 & \cdots & 1 \\ p_0 & 0 & 0 & \cdots & 0 \end{bmatrix} \otimes \begin{bmatrix} x_{m-1}(t) \\ x_{m-2}(t) \\ \cdots \\ x_1(t) \\ x_0(t) \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ b(t) \end{bmatrix}$$

(⊕ 为异或、⊗ 为与)，设 $F=\begin{bmatrix} p_{m-1} & 1 & 0 & \cdots & 0 \\ p_{m-2} & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_1 & 0 & 0 & \cdots & 1 \\ p_0 & 0 & 0 & \cdots & 0 \end{bmatrix}$ 得 $X(t+1)=F\otimes X(t)\oplus \begin{bmatrix} 0 \\ \cdots \\ 0 \\ b(t) \end{bmatrix}$，

同理 $X(t+2)=F\otimes X(t+1)\oplus \begin{bmatrix} 0 \\ \cdots \\ 0 \\ b(t+1) \end{bmatrix}$，将 $X(t+1)$ 表达式代入可得：

$$X(t+2)=F\otimes\left(F\otimes\left[(t)\oplus\begin{bmatrix} 0 \\ \cdots \\ 0 \\ b(t) \end{bmatrix}\right]\oplus\begin{bmatrix} 0 \\ \cdots \\ 0 \\ b(t+1) \end{bmatrix}\right)$$

$$=F^2\otimes X(t)\oplus\begin{bmatrix} 0 \\ \cdots \\ b(t) \\ b(t+1) \end{bmatrix}$$

假设 w 时刻对应的状态为 $X(t+w)=F^w\otimes X(t)\oplus[0 \cdots b(t) \cdots b(t+w-1)]^T$，

$$X(t+w+1)=F\otimes X(t+w)\oplus\begin{bmatrix} 0 \\ \cdots \\ 0 \\ b(t+w) \end{bmatrix}$$

$$=F\otimes\left(F^w\otimes X(t)\oplus\begin{bmatrix} 0 \\ \cdots \\ b(t) \\ \cdots \\ b(t+w-1) \end{bmatrix}\right)\oplus\begin{bmatrix} 0 \\ \cdots \\ 0 \\ b(t+w) \end{bmatrix}$$

$$=F^{w+1}\otimes X(t)\oplus\begin{bmatrix} 0 \\ \cdots \\ b(t) \\ \cdots \\ b(t+w-1) \\ 0 \end{bmatrix}\oplus\begin{bmatrix} 0 \\ \cdots \\ 0 \\ b(t+w) \end{bmatrix}$$

$$=F^{w+1}\otimes X(t)\oplus\begin{bmatrix} 0 \\ \cdots \\ b(t) \\ \cdots \\ b(t+w) \end{bmatrix}$$

以上归纳法可得如下结论：$X'=F^w\otimes X\oplus D$(此处⊗为异或、与构成的矩阵操作，$X=(x_{m-1}\cdots, x_1, x_0)^T$ 表示系统的初始状态，$D=[0\cdots0|b_0\cdots b_{w-1}]^T$ 为 w 位输入，X' 表示 w 时刻的系统状态，$w\leqslant m$)，该表达式为 w 位串行运算等价的并行矩阵方法。该结论与文献[6]中一致。

令 $P'=[p_{m-1}\cdots p_1 p_0]^T$，$F^w=[F^{w-1}\otimes P'\cdots F\otimes P'P'\,|\,\frac{I_{m-w}}{0}]$，其中，$F^w$ 为 m 维的方阵，$F^{w-1}\otimes P'\cdots F\otimes P'P'$ 对应 $m\times w$ 维的矩阵，I_{m-w} 为 $m-w$ 维的单位阵，0 为 $w\times(m-w)$ 维的零矩阵。

4　查询表与并行矩阵算法

当 $w\leqslant m$ 时，将系统的当前状态 X 拆为两个部分，即：

$$X=\begin{bmatrix}x_{m-1}\\x_{m-2}\\\cdots\\\cdots\\x_1\\x_0\end{bmatrix}=\begin{bmatrix}x_{m-1}\\\cdots\\x_{m-w}\\0\\\cdots\\0\end{bmatrix}\oplus\begin{bmatrix}0\\\cdots\\0\\x_{m-w-1}\\\cdots\\x_0\end{bmatrix}$$

可得：

$$X'=F^w\otimes X\oplus D$$

$$=F^w\otimes\left(\begin{bmatrix}x_{m-1}\\\cdots\\x_{m-w}\\0\\\cdots\\0\end{bmatrix}\oplus\begin{bmatrix}0\\\cdots\\0\\x_{m-w-1}\\\cdots\\x_0\end{bmatrix}\right)\oplus\begin{bmatrix}0\\\cdots\\0\\b_0\\\cdots\\b_{w-1}\end{bmatrix}$$

$$=F^w\otimes\begin{bmatrix}x_{m-1}\\\cdots\\x_{m-w}\\0\\\cdots\\0\end{bmatrix}\oplus\left(\left[F^{w-1}\otimes P'\cdots F\otimes P'P'\,\middle|\,\frac{I_{m-w}}{0}\right]\otimes\right.$$

$$\left.\begin{bmatrix}0\\\cdots\\0\\x_{m-w-1}\\\cdots\\x_0\end{bmatrix}\oplus\begin{bmatrix}0\\\cdots\\0\\b_0\\\cdots\\b_{w-1}\end{bmatrix}\right)$$

$$=F^w\otimes\begin{bmatrix}x_{m-1}\\\cdots\\x_{m-w}\\0\\\cdots\\0\end{bmatrix}\oplus\left(\begin{bmatrix}x_{m-w-1}\\\cdots\\x_0\\0\\\cdots\\0\end{bmatrix}\oplus\begin{bmatrix}0\\\cdots\\0\\b_0\\\cdots\\b_{w-1}\end{bmatrix}\right)$$

$$=\left(F^w\otimes\begin{bmatrix}x_{m-1}\\\cdots\\x_{m-w}\\0\\\cdots\\0\end{bmatrix}\right)\oplus\begin{bmatrix}x_{m-w-1}\\\cdots\\x_0\\b_0\\\cdots\\b_{w-1}\end{bmatrix}$$

该表达式对应图 1(a)、图 2(a)所示的表查询方法。设 $H^w=F^w\otimes[x_{m-1}\cdots x_{m-w}\ 0\cdots 0]^{\mathrm{T}}$ 为$(x_{m-1}\cdots x_{m-w})$对下一寄存器状态值的影响，进一步简化：

$$H^w=F^w\otimes\begin{bmatrix}x_{m-1}\\\cdots\\x_{m-w}\\0\\\cdots\\0\end{bmatrix}$$

$$=[F^{w-1}\otimes P'\cdots F\otimes P'P'\,|\,\frac{I_{m-w}}{0}]\otimes\begin{bmatrix}x_{m-1}\\\cdots\\x_{m-w}\\0\\\cdots\\0\end{bmatrix}$$

$$=[F^{w-1}\otimes P'\cdots F\otimes P'P']\otimes\begin{bmatrix}x_{m-1}\\\cdots\\x_{m-w}\end{bmatrix}$$

$$=\Gamma^w\otimes X_w$$

即为查询表的并行矩阵生成方法，$\Gamma^w=(F^{w-1}\otimes P'\cdots F\otimes P'P']$为查询表并行生成矩阵。

当并行处理长度 $w>m$ 时，矩阵 F^w 为 m 维的方阵，无法处理长度超过 m 的输入。设 $km<w\leqslant(k+1)m$(k 为自然数)，可以先处理 $w-km$ 位输入 $B_{w-km}=[b_0\cdots b_{w-km-1}]^{\mathrm{T}}$，再处理剩下的 k 个 m 位输入 $B_{m_1}=[b_{w-km}\cdots b_{w-km+(m-1)}]^{\mathrm{T}}$，…，$B_{m_k}=[b_{w-m}\cdots b_{w-1}]^{\mathrm{T}}$，获得对应的查询表。

$k=1$ 时并行矩阵处理流程如图 3 所示。

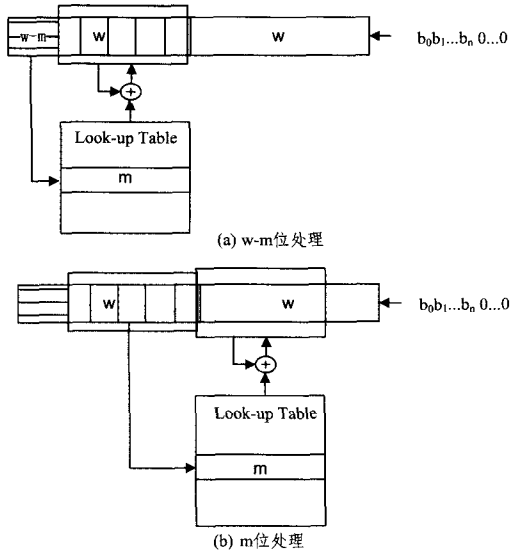

图 3　$m<w\leqslant 2m$ 时，I 型电路查询表对应并行矩阵生成流程

$$H^w=F^m\otimes((\cdots\otimes(F^m\otimes(\Gamma^{w-km}\otimes B_{w-km}\oplus B_{m_1})\oplus\cdots))\oplus B_{m_k})$$

$$=F^m\otimes((\cdots\otimes(F^m\otimes(F^{w-km}\otimes\begin{bmatrix}B_{w-km}\\0_{(k+1)m-w}\end{bmatrix}\oplus B_{m_1})\oplus\cdots))\oplus B_{m_k})$$

$$=(F^w\otimes\begin{bmatrix}B_{w-km}\\0_{(k+1)m-w}\end{bmatrix})\oplus(F^{km}\otimes B_{m_1})\cdots\oplus(F^m\otimes B_{m_k})$$

$$=[F^{w-1}\otimes P'\cdots F^{w-m}\otimes P']\otimes\begin{bmatrix}b_0\\\cdots\\b_{w-km-1}\\0_{(k+1)m-w}\end{bmatrix}\oplus([F^{km}\cdots F^m]\otimes\begin{bmatrix}B_{m_1}\\\cdots\\B_{m_k}\end{bmatrix})$$

$$=[F^{w-1}\otimes P'\cdots F^{km}\otimes P']\otimes\begin{bmatrix}b_0\\\cdots\\b_{w-km-1}\end{bmatrix}\oplus[F^{km-1}\otimes P'\cdots F\otimes P'\otimes P'\otimes P']\otimes\begin{bmatrix}b_{w-km}\\\cdots\\b_{w-1}\end{bmatrix}$$

$$=[F^{w-1}\otimes P'\cdots F^{km}\otimes P'\mid F^{km-1}\otimes P'\cdots F\otimes P'\otimes P'\otimes P']\otimes\begin{bmatrix}b_0\\\cdots\\b_{w-km-1}\\b_{w-km}\\\cdots\\b_{w-1}\end{bmatrix}$$

$$=[F^{w-1}\otimes P'\cdots F\otimes P'P']\otimes\begin{bmatrix}b_0\\\cdots\\b_{w-1}\end{bmatrix}$$

$$=\Gamma^w\otimes B_w$$

由此可知，$w>m$ 时，查询表 H^w 的生成方法与 $w\leqslant m$ 的方式一致，处理流程如图 2(b)所示。所以，I 型电路的任意并行位宽 w 的查询表都由矩阵 $\Gamma^w=[F^{w-1}\otimes P'\cdots F\otimes P'P']$ 生成。

另外，根据并行矩阵 Γ^w 可以直接画出对应的逻辑电路，便于硬件设计[9,10]。

5 $w>2^m-1$ 块处理

当 w 足够大时，会出现迭代现象，可以进行块处理。令 I 型电路输入为全零序列，则 I 型电路变为 LFSR 电路(见图 4)。

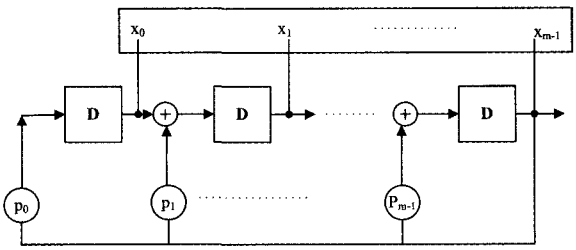

图 4 I 型电路对应 LFSR 电路

当生成多项式为本原多项式时，LFSR 电路经过 $T=2^m-1$ 次跳变回到初始状态(若初始状态为全零，则一直保持全零状态)[8]。由公式 $X'=F^w\otimes X\oplus D(w\leqslant m)$ 可得 $X=F^{T-km}\otimes(\cdots\otimes(F^m\otimes(F^m\otimes X\oplus 0_m)\oplus 0_m)\oplus\cdots)\oplus 0_m=F^T\otimes X$，$F^T\otimes P'=P'$ 得证。

当 $w=n>T$ 时，以 I 电路为例，通过一次查表获得校验值 $r=H^w+D_m=H^w$，其中 D_m 为 m 维零向量，设 $kT<w<(k+1)^T$：

$$H^w=\Gamma^w\otimes B_w=[F^{w-1}\otimes P'\cdots F\otimes P'P']\otimes\begin{bmatrix}b_0\\\cdots\\b_{w-1}\end{bmatrix}$$

$$=[F^{w-1}\otimes P'\cdots F^{kT}\otimes P'\mid\cdots\mid F^{2T-1}\otimes P'\cdots F^T\otimes P'\mid F^{T-1}\otimes P'\cdots P']\otimes\begin{bmatrix}b_0\\\cdots\\b_{w-kT-1}\\b_{w-kT}\\\cdots\\b_{w-1}\end{bmatrix}$$

$$=([F^{(k+1)T-1}\otimes P'\cdots\mid F^{w-1}\otimes P'\cdots F^{kT}\otimes P']\otimes\begin{bmatrix}0_{(k+1)T-w}\\b_0\\\cdots\\b_{w-kT-1}\end{bmatrix})\oplus\cdots\oplus([F^{T-1}\otimes P'\cdots F\otimes P'P']\otimes\begin{bmatrix}b_{w-T}\\\cdots\\b_{w-1}\end{bmatrix})$$

$$=[F^{T-1}\otimes P'\cdots F\otimes P'P']\otimes(\begin{bmatrix}0_{(k+1)T-w}\\b_0\\\cdots\\b_{w-kT-1}\end{bmatrix}\oplus\cdots\oplus\begin{bmatrix}b_{w-T}\\\cdots\\b_{w-1}\end{bmatrix})$$

$$=[F^{T-1}\otimes P'\cdots F\otimes P'P']\otimes X_T$$

由上式可知，通过将输入序列分成 T 长度的块(不足则高位补 $(k+1)T-w$ 位零，所有的块异或以后得 X_T)，只需查一次表即可获得校验值。但该方法并不适用于 m 值较大的本原多项式，例如 CRC32，需要 $2^{32}\times 32\approx 128$Gbit 的存储空间存储查询表，此时，可将 X_T 作为新的输入，选择合适的并行处理带宽 w' 进行计算。

结束语 根据图 2 的计算流程，通过 MATLAB 实验，将 $(2^{24}-1)b$ 伪随机序列作为输入，求生成多项式 CRC16 的校验值 r。不同并行处理带宽 $w=1,3,5,7,9,13,15,16,17,18,19$ 对应查询表方法、并行矩阵方法与块处理后并行矩阵方法的性能分析如图 5 所示，其中表生成耗时、表存储空间、表查询耗时曲线表示查询表方法性能，矩阵存储空间、矩阵查询耗时曲线表示并行矩阵方法性能，块处理耗时曲线表示块处理方法性能。观察发现上述 3 种方法，无论 $w\leqslant m$ 或者 $w>m$ 时，w 增大，需要实时处理的表查询处理、并行矩阵计算、块处理后并行矩阵计算的耗时均下降，表明并行处理带宽的加大，能够显著提高 3 种 CRC 计算方法的运算速度；但是查询表方法对应的查询表存储空间 $(2^w\times 32b)$ 及查询表生成耗时随 w 增大而快速增加，而并行矩阵存储空间 $(w\times 32b)$ 增长缓慢，矩阵生成耗时趋于零(故未在图中画出)；由于 $2^{24}>2^{16}$ 满足块处理条件，因此将处理数据长度由 $(2^{24}-1)b$ 缩短为 $(2^{16}-1)b$，再进行不同位宽 w 的并行矩阵方法查询，处理速度得到显著提升，由图 5 中块处理耗时曲线与矩阵查询耗时比较可知，图 5 中表查询耗时与矩阵查询耗时接近，但硬件环境中需要根据实际情况进一步分析。

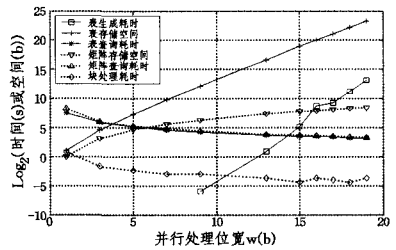

图 5 相同输入序列 3 类运算方法的性能分析

本文根据 I 型 CRC 校验值按位生成电路及 I 型电路表查询的处理流程推导出了查询表与并行处理矩阵之间的关系，并证明任意阶次生成多项式及任意处理位宽的查询表都可以通过生成矩阵 Γ^w 获得。通过实验发现，并行处理带宽 w 的增加能够有效地提高 CRC 处理速度，并行矩阵方法在时间与空间两个方面均表现良好；当处理位宽 w 足够大时，可以通过块处理来降低校验序列长度，再通过一次查表或进行多次 w 位的并行处理获得校验值，从而显著提高并行处理速度。

参 考 文 献

[1] Tanenbaum A S. Computer Networks(4th Edition) [M]. Prentice Hall, 2003: 192-200

[2] 斯廷森. 密码学原理与实践(第三版) [M]. 冯登国，等译. 北京：电子工业出版社，2009

[3] Williams R N. A PAINLESS GUIDE TO CRC ERROR DE-

TECTION ALGORITHMS [EB/OL]. http://www.ross.net/crc/download/crc_v3.txt,2010-04
[4] 毕占坤,张羿猛,黄芝平,等. 基于逻辑设计的高速 CRC 并行算法研究及其 FPGA 实现[J]. 兵工学报,2007,28(12):2243-2249
[5] 刘畅,章建雄,王玉艳. RapidIO 控制器的 CRC 模块设计[J]. 计算机工程,2011,37(15):238-239
[6] Campobello G, Patane G, Russo M. Parallel CRC realization [J]. IEEE Transactions on Computers,2003,52(10):1312
[7] Peterson W W, Brown D T. Cyclic Codes for Error Detection [J]. Proceedings of the IRE,1961,49(1):228-235
[8] Wikimedia Foundation, Inc. Linear feedback shift register [EB/OL]. http://en.wikipedia.org/wiki/Linear_feedback_shift_register,2011-06-30
[9] Shieh M D, Sheu M H, Chen C H H. A Systematic Approach for Parallel CRC Computations [J]. Journal of Information Science and Engineering,2001,17:445-461
[10] 阳璞琼,何怡刚,谭阳红,等. 超高频 RFID 系统 CRC 电路设计[J]. 电路与系统学报,2009,14(2):18-21

(上接第 135 页)

重要指标。图 6 描绘了在 Chord,HASN,以及优化的 HASN 中,AMTD 与 N(节点数)的关系。我们可以看到,虽然每个路由算法中的 AMTD 演示了与 N 的一种 O(log$N$)关系,但是 HASN 和优化的 HASN 的性能比 Chord 的好。这是因为 HASN 把临近网络的性质考虑在内。

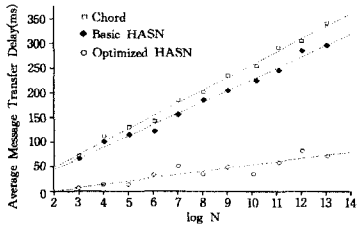

图 6 AMTD 与节点数 N 的关系

2)节点加入时发送消息的数目

图 7 示出了节点加入时发送消息的数目与 N 的关系。

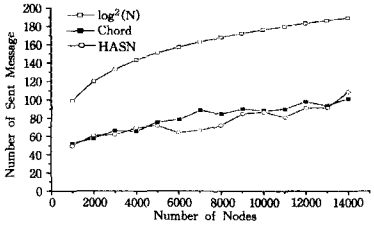

图 7 节点加入时发送消息的数目与 N 的关系

3)节点离开时发送消息的数目

节点离开时发送消息的数目,也是评价一个自组织网络性能的重要指标。图 8 描绘了 Chord 和 HASN 中节点离开时发送消息的数目与 N 的关系。我们可以看到,Chord 和 HASN 在用路由算法演示 N 与 O(log2N)的关系时,彼此的性能是相似的。

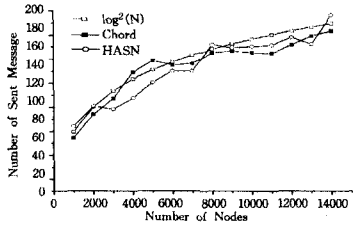

图 8 节点离开时发送消息的数目与 N 的关系

结束语 在自组织网络研究领域,移动特设网络(MANET)始终是一个热点,吸引了许多科学家的关注。与此相反,很少有人关注基于自组织网络的网络领域。HASN 是一个基于互联网基础架构的自组织网络路由模型。基于路由算法的 DHT,它继承了结构化 P2P 覆盖网络的所有优点并保证了网络的可扩展性。通过引入集聚分组机制和层次结构,HASN 充分利用互联网的网络邻近特性和网络局部特性改善了路由性能。

进一步的研究将侧重于把 P2P 计算模式延伸到 MANET。我们的目标是通过路由协议在一个逻辑上的命名空间进行操作和通过 MANET 路由协议在物理上的名称空间进行操作,从而提出一种新的多跳路由协议以传统的形式进行无缝集成。

参 考 文 献

[1] Èapkun S, Buttyán L, Hubaux J P. Self-organized public-key management for ad hoc networks[J]. IEEE Transactions on Mobile Computing,2003,2(1)
[2] Perkins C, Belding-Royer E M, Das S R. Ad hoc on-demand distance vector (AODV) Routing[Z]. Network Working Group RFC 3561, July 2003
[3] Parameswaran M, Susarla A. P2P Networking: An Information-Sharing Alternative[R]. IEEE Computing Practices, July 2001
[4] C—NET NEWS. Napster among fastest-growing Net technologies[EB/OL]. http://news.com.com/2100-1023-246648.html. October 2000
[5] Clarke I, Sandberg O, Wiley B, et al. Freenet: A distributed anonymous information storage and retrieval system [R]. Workshop on Design Issues in Anonymity and Unobservability, July 2000
[6] Gnutella[EB/OL]. http://gnutella.wego.com
[7] Ratnasamy S, Francis P, Handley M, et al. A Scalable Content-Addressable Network[C]// Annual conference of the Special Interest Group on Data Communication (SIGCOMM 2001). August 2001
[8] Stoica I, Morris R, Karger D, et al. Chord: A scalable peer-to-peer lookup service for Internet applications[C]// Annual conference of the Special Interest Group on Data Communication (SIGCOMM 2001). August 2001
[9] Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems[C]// International Conference on Distributed Systems Platforms (Middleware 2001). November 2001
[10] Zhao Ben, Kubiatowicz J, Joseph A. Tapestry: An infrastructure for fault-tolerant wide-area location and routing[R]. UCB/CSD-01-1141. Computer Science Division, U. C. Berkeley, April 2001
[11] Jain S, Mahajan R, Wetherall D. A Study of the Performance Potential of DHT-based Overlays[C]// 4th USENIX Symposium on Internet Technologies and Systems (USITS 2003). March 2003