doi:10.3969/j.issn.1000-436x.2014.04.021

基于并行处理的一种新型有效的网络架构 BSN-MOT

李江昀[1,2]，孙丽婷[1,2]

(1. 北京科技大学 自动化学院，北京 100083；2. 北京科技大学 钢铁流程先进控制教育部重点实验室，北京 100083)

摘 要：结合 BSN 及 MOT 架构的双重优势，提出一种新型的双层架构体系 BSN-MOT，并研究了其上的拓扑性质及在并行处理中应用的基本通信及应用等操作算法。算法包括行、列树广播、单向广播、数据求和、矩阵乘积、最短路径路由及多项式求根。最后，通过与其他 2 种有效的树形双层网络架构 MMT 及 OMULT 比较说明，基于 BSN-MOT 架构的通信算法要比其他 2 种网络有着更小的时间复杂度，且 BSN-MOT 是一种更具有竞争力的体系结构形式。

关键词：Biswapped 网络；拓扑性质；行树广播；列树广播；矩阵乘积；多项式求根

中图分类号：TP393.01 **文献标识码**：A **文章编号**：1000-436X(2014)04-0182-09

New efficient network architecture BSN-MOT for parallel processing

LI Jiang-yun[1,2], SUN Li-ting[1,2]

(1. School of Automation, University of Science and Technology Beijing, Beijing 100083,China;
2. Key Laboratory of Advanced Control of Iron and Steel Process (Ministry of Education),
University of Science and Technology Beijing, Beijing 100083,China)

Abstract: BSN-MOT as a two-tier architecture that takes the advantages of both the BSN and the MOT was presented. Topological property and many communication and application algorithms are investigated. The communication algorithms include row-tree and column-tree broadcast, one to all broadcast, data sum, matrix multiplication, shortest path rooting and polynomial root finding. In contrast with other two similar tree-based two-tier architectures MMT and OMULT, the results show that the algorithms that run on BSN-MOT are much faster, and BSN-MOT is more competitive.

Key words: Biswapped network; topological property; row-tree broadcast; column-tree broadcast; matrix multiplication; polynomial root finding

1 引言

Biswapped 网络(BSN)是一种双层架构的互联网络，它提供了一种构建可扩展性、模块化、可继承性、容错性等大规模的并行计算机体系结构形式[1]。基于 BSN 的多处理器系统架构，是由 $2N$ 个规模为 N 的处理器组组合而成，即含有 $2N^2$ 个处理器。另一方面，作为非常有效的互联网络拓扑结构，MOT(mesh of tree)拥有 2 个非常理想的性能，即小直径及高的对分宽度，并且在只考虑速度的情况下，MOT 被认为是最快速的互联网络，在解决如分组路由、排序、前置计算、矩阵乘法运算、最短路径、最近邻居等重要算法问题时，运行在 MOT 上的时间复杂度只达到 $O(\log^{\sqrt{N}})$ 或 $O(\log^{N})$ 。

本文结合 BSN 及 MOT 的双重拓扑优势，提出一种新型网络架构 BSN-MOT，并对新架构的模型及基本拓扑性质进行了具体分析；同时，本文研究了运行在该架构上的基本通信及应用等并行算法，并与其他 2 种流行的树形双层架构在算法时间复杂度上进行了比较，结果显示 BSN-MOT 更为快速。

其实，许多其他的树形双层架构都被研究给出，如 MCT(mesh-connected tree)[2]、MMT(mult-mesh of tree)[3]、OMULT(optical multi-trees)[4]等。在文献[2]中，Kemal 证明了 MCT 是比超立方体更简单，且成本更低的架构体系。Jana[3]给出了 MMT，分析了其拓扑性质及架构有效性，并理论验证了其并行算法的快速性。 Islam[4]提出了 OMULT，并与 OTIS-Mesh 的比较表明，OMULT 更具有优势。然

收稿日期：2014-01-06；修回日期：2014-03-17

而，根据 BSN 的网络连通性及极大容错性，BSN-MOT 将会比其他双层树形架构更具有竞争力的架构体系，研究其拓扑性质对网络的分析及网络的设计具有重要的意义。

同时，BSN-MOT 作为大规模的互联网络，且根据其优秀拓扑性质，研究其上的并行算法可推动现代并行计算机的实际应用。实际上，近些年，针对大规模交换互联网络的并行算法研究确实也成为了热点。例如，针对 OTIS(optical transpose interconnection system)，Sahni 等提出了建立在 OTIS-Mesh 上的矩阵算法[5]，以及基于 OTIS-Hypercube 和 OTIS-Mesh 上的 BPC 排列算法[6,7]；Jana 等给出了基于 OTIS-Mesh 的多项式插值算法[8]；Rajasekaran 等研究了 OTIS-Mesh 上的选择及路由等有效算法[9]；而诸如图形处理[10]、数据求和[11]、k-k 排序[12]等 OTIS 算法也已给出。对于 BSN 架构，Wei wenhong、Sun liting 等也研究了其上的矩阵算法[13,14]，在文献[15]中，Ye 等提出基于 BSN-Hypercube 的数据广播等算法。本文同样研究了在并行处理中基于 BSN-MOT 的基本通信及应用操作算法，这些算法可以应用至重要的开发程序，如图像处理、矩阵代数、图论等。

2　基本概念和性质

本文定义 BSN-MOT 是由 $2n^2$ 个同构的 $n\times n$ (定义 $n=\sqrt{N}$)MOT 组成，即在 BSN-MOT 多处理器系统中，BSN-MOT 含有 $2n^4$ 个处理器，它是由 $2n^2$ 个点不相交的同构 MOT 组成，每个处理器的索引都记为(i, g, p)，其中，i 代表部索引，二维布局 $g(x, y)$是组索引，代表了组的位置，$p(x, y)$是处理器的本地索引，则索引 (i,g_x,g_y,p_x,p_y) 代表了处于第 i 部的 $g(x, y)$组的 $p(x, y)$处理器。其中，同一组内的处理器链接是组内链接，而不同部的处理器链接则是组间链接，即第 i 部组 g 中的处理器 p 是与第 $\bar{i}$ 部组 p 中的处理器 g 连接，也就是 $(i,g_x,g_y,\ p_x,p_y)$ 与 $(\bar{i},p_x,p_y,g_x,g_y)$ 连接。图 1 给出了含有 32 个处理器的 BSN-MOT。在图中，小方框代表处理器，大方框则代表处理器组。小方框中的数据对(i, j)指明每组 MOT 中的本地索引；而大方框外的数据对(a, b)则是每部中的组索引。

2.1　基本定义

定义 1　(组内链接)。MOT 中的组内链接包含了行树链接及列树链接[16]。

在 MOT 的每行中，处理器都形成了一颗行树，根节点索引记为 $(i,g_x,g_y,p_x,1)$，且在每部中，对任意 g_x，g_y，$1\leqslant g_x,g_y\leqslant n$，$P(i,g_x,g_y,p_x,p_y)$ 都直接与 $P(i,g_x,g_y,p_x,2p_y)$ 及 $P(i,g_x,g_y,p_x,2p_y+1)$（若存在）连接，其中，$1\leqslant p_x,p_y\leqslant n$。

相似地，在 MOT 的每列中，处理器都形成了一棵列树，其根节点为 $P(i,g_x,g_y,1,p_y)$，且对 g_x，g_y，$1\leqslant g_x,g_y\leqslant n$，$P(i,g_x,g_y,p_x,p_y)$ 都直接与 $P(i,g_x,g_y,2p_x,p_y)$ 及 $P(i,g_x,g_y,2p_x+1,p_y)$ (若存在)连接，其中，$1\leqslant p_x,p_y\leqslant n$。

图 2 给出了 BSN-MOT 中单组 6×6 MOT 示例。

定义 2　(组间链接)。组间链接定义如下：处理器 $P(i,g_x,g_y,p_x,p_y)$ 直接与 $P(\bar{i},p_x,p_y,g_x,g_y)$ 连接。

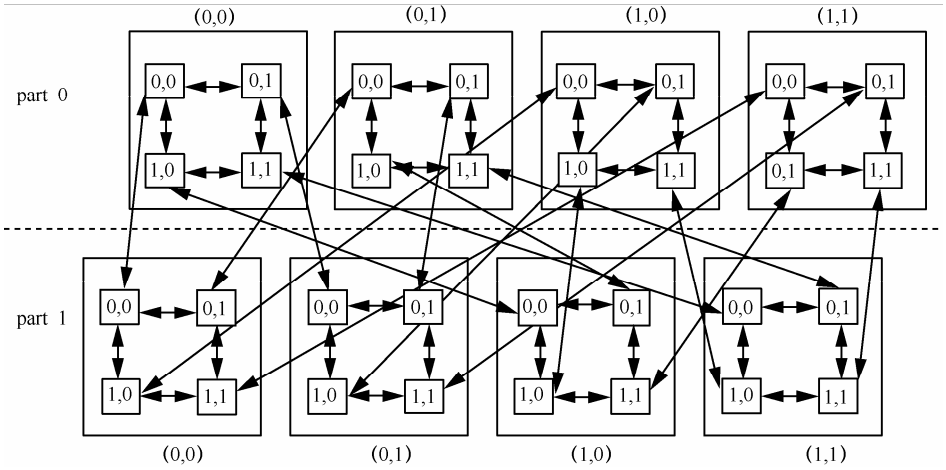


图 1　32 处理器的 BSN-MOT

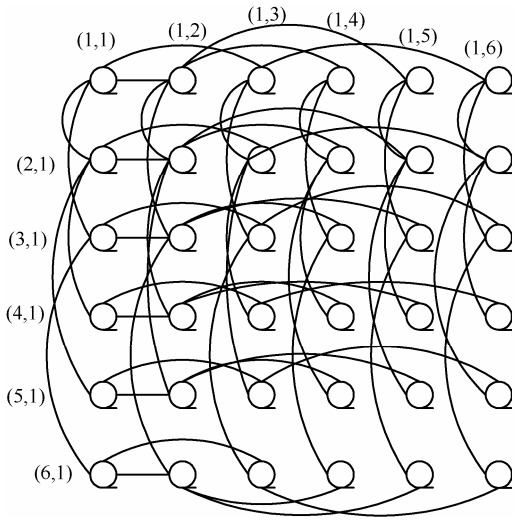


图 2 BSN-MOT$_6$ 中部 0 中的组 $g(0,0)$

2.2 拓扑性质

2.2.1 最短路径

假设 $d(u,v)$ 表示 MOT 中处理器 P_1 到 P_2 的最短路径长度；$P_1(i_1,g_{x1},g_{y1},p_{x1},p_{y1})$ 与 $P_2(i_2,g_{x2},g_{y2},p_{x2},p_{y2})$ 是 BSN-MOT 中的任意两不同点，则 P_1 到 P_2 的最短路径分下列 3 种情况构造。

1) 路径只存在组内移动。当且仅当 $i_1=i_2$，$g(x_1,y_1)=g(x_2,y_2)$，且 $p(x_1,y_1)\neq p(x_2,y_2)$，即 P_1 与 P_2 是同部内同组中的不同两点，则在 MOT 中，从 P_1 到 P_2 的一条最短路径为 $p_1\to s_1\Rightarrow s_2\to p_2$，其中 s_1 属于与 p_1 相邻的处理器集合，s_2 属于与 p_2 相邻的处理器集合。

2) 路径包含偶数个组间移动。当且仅当 $i_1=i_2$，$g(x_1,y_1)\neq g(x_2,y_2)$，且 $p(x_1,y_1)\neq p(x_2,y_2)$，即 P_1 与 P_2 是同部内不同组的两点。如果组间移动的数量超过 2 个，则可压缩得到从 P_1 到 P_2 的一条最短路径 $p_1\to s_1\Rightarrow s_2\to p_2$，即

$$(i_1,g_{x1},g_{y1},p_{x1},p_{y1})\Rightarrow(i_1,g_{x1},g_{y1},p_{x2},p_{y2})\to$$
$$(\overline{i_1},p_{x2},p_{y2},g_{x1},g_{y1})\Rightarrow$$
$$(\overline{i_1},p_{x2},p_{y2},g_{x2},g_{y2})\to(i_1,g_{x2},g_{y2},p_{x2},p_{y2})$$

其中，双箭头代表组内移动，单箭头表示组间移动。

3) 路径包含奇数个组间移动。当且仅当 $i_1\neq i_2$，即 P_1 与 P_2 属于不同部的两点。在这种情况下，最短路径必包含一个组间移动，其最短路径为 $p_1\to s_1\Rightarrow s_2\to p_2$，即

$$(i_1,g_{x1},g_{y1},p_{x1},p_{y1})\Rightarrow(i_1,g_{x1},g_{y1},g_{x2},g_{y2})\to$$
$$(i_2,g_{x2},g_{y2},g_{x1},g_{y1})\Rightarrow(i_2,g_{x2},g_{y2},p_{x2},p_{y2})$$

注意的是，如果 $p(x_1,y_1)=g(x_2,y_2)$，则 $(i_1,g_{x1},g_{y1},p_{x1},p_{y1})\Rightarrow(i_1,g_{x1},g_{y1},g_{x2},g_{y2})$ 为一条空路径；若 $g(x_1,y_1)=p(x_2,y_2)$，则 $(i_2,g_{x2},g_{y2},g_{x1},g_{y1})\Rightarrow(i_2,g_{x2},g_{y2},p_{x2},p_{y2})$ 同为空路径。

从以上情况可以看出，情况 1)的最短路径长度为 $d(p(x_1,y_1),p(x_2,y_2))$，情况 2)和情况 3)的最短路径长度则分别为 $d(p(x_1,y_1),p(x_2,y_2))+d(g(x_1,y_1),g(x_2,y_2))+2$ 及 $d(p(x_1,y_1),g(x_2,y_2))+d(g(x_1,y_1),p(x_2,y_2))+1$。

2.2.2 直径

定理 1 BSN-MOT 的直径为 $8\log^n+2$。

证明 因 BSN-MOT 的每组都是一 $n\times n$ MOT，而直径是指网络图中最大的离径，则 BSN-MOT 每组的直径为 $4\log^n$。假设 P_1 与 P_2 各自代表 BSN-MOT 中任意不同的源节点及目标节点。如果 $i_1\neq i_2$，则两节点可通过一组间链接来连接。如此，可从源节点过渡到一中间节点，即 $(i_1,g_{x1},g_{y1},g_{x2},g_{y2})$，其路由距离即为 $4\log^n$。而从 $(i_1,g_{x1},g_{y1},g_{x2},g_{y2})$ 出发，可以通过一组间移动到达 $(i_2,g_{x2},g_{y2},g_{x1},g_{y1})$，之后通过另一组内移动，这样就可到达目标节点 P_2。通过计算，从源节点到目标节点的最大离径即为 $8\log^n+1$。然而，通过另一种假想，如果 $i_1=i_2$，则整个路由过程会通过 2 个组间移动和 2 个组内移动来完成，依照这种情况，两节点的最大离径即为 $8\log^n+2$，因此，BSN-MOT 的直径即为 $8\log^n+2$。

2.2.3 容错直径

假设 BSN-MOT 的任意源节点与目标节点分别由 P_1 及 P_2 来表示。

倘若 $i_1\neq i_2$，$g(x_1,y_1)\neq g(x_2,y_2)$，则如果容错节点是在源节点所在的同行或同列，那么 BSN-MOT 的直径将变为 $8\log^n+1$，其路径即是 2.2.1 节提及的第 3 种情况；然而，如果容错节点是其路径上的中间节点 $(i_1,g_{x1},g_{y1},g_{x2},g_{y2})$，其路由路径将变为

$$(i_1,g_{x1},g_{y1},p_{x1},p_{y1})\to(i_2,p_{x1},p_{y1},g_{x1},g_{y1})\Rightarrow$$
$$(i_2,p_{x1},p_{y1},p_{x2},p_{y2})\to(i_1,p_{x2},p_{y2},p_{x1},p_{y1})\Rightarrow$$
$$(i_1,p_{x2},p_{y2},g_{x2},g_{y2})\to(i_2,g_{x2},g_{y2},p_{x2},p_{y2})$$

BSN-MOT 的直径变为 $8\log^n+3$。

倘若 $i_1=i_2$，$g(x_1,y_1)\neq g(x_2,y_2)$，而容错节点是在源节点所在的同行或同列时，BSN-MOT 的直

径仍保持为 $8\log^n+2$，其路由路径即是 2.2.1 节提及的第 2 种情况；相似地，如果容错节点是其路径上的中间节点 $(i_1,g_{x1},g_{y1},p_{x2},p_{y2})$，则路由将绕过此节点，其路由路径变为

$$(i_1,g_{x1},g_{y1},p_{x1},p_{y1})\to(i_2,p_{x1},p_{y1},g_{x1},g_{y1})\Rightarrow$$
$$(i_2,p_{x1},p_{y1},p_{x2},p_{y2})\to(i_1,p_{x2},p_{y2},p_{x1},p_{y1})\Rightarrow$$
$$(i_1,p_{x2},p_{y2},g_{x2},g_{y2})\to(i_2,g_{x2},g_{y2},p_{x2},p_{y2})$$

此时，BSN-MOT 的直径变为 $8\log^n+3$。

2.2.4 宽直径

宽直径也是衡量网络容错性的一个重要度量标准，而描述宽直径就需要提到容器的概念。假设 G 是连通图，u 及 v 是它的节点，则图 G 中从 u 到 v 的一组并行路被称为一(u,v)容器，容器的宽度为其中并行路的条数，容器的长度为其中最长路的长度。某两点间宽度为 d 的所有容器的最小长度称为这两点间的 d-宽距离，所有点对间的 d-宽距离的最大值称作是图 G 的 d-宽直径[1]。然而求一般图的宽直径是 NP 完全难问题，但是参考文献[1]的式(1)~式(8)，本文可推理得到 BSN-MOT 的宽直径上限不超过 $3d(MOT)+6$，即 $12\log^n+6$。

3　通信操作算法

本节将提出并行处理中基于 BSN-MOT 架构的各类基本的通信及应用操作算法，如广播算法、数据求和、矩阵乘积、排序、最短路径路由及多项式求根等，然后通过算法复杂度分析及与其他 2 种流行树形网络 MMT 及 OMULT 的比较来证明 BSN-MOT 是非常快速、有效的网络架构。

3.1　行树广播(V_x, A)

数据广播是并行计算中最基本的通信操作之一。在本算法中，数据 v_x 将被广播至二维 BSN-MOT 的第 x 行。此算法描述如下。

step1　赋值 $v_{(g_x-1)n+p_x}$ 至不同部的首列组的首列处理器，则寄存器 $A(i,g_x,1,p_x,1)$ 都有了赋值。

step2　通过行树链接，广播寄存器 A 内的值至同行的其他处理器，如此，第 1 列组的所有处理器都有了数据值。

step3　对整个网络通过组间链接执行组间移动。这样，不同部的第 1 列处理器都拥有了一数值。

step4　广播处理器中的数值至同行的其他处理器。

step5　再次执行组间移动。

step6　在 step1 中广播处理器中的数值至同行的其他处理器。

step7　算法结束。最后，所有 x 行的处理其都有了数值 v_x。

图 3 给出了 BSN-MOT$_3$ 经过行树广播之后的赋值示例。

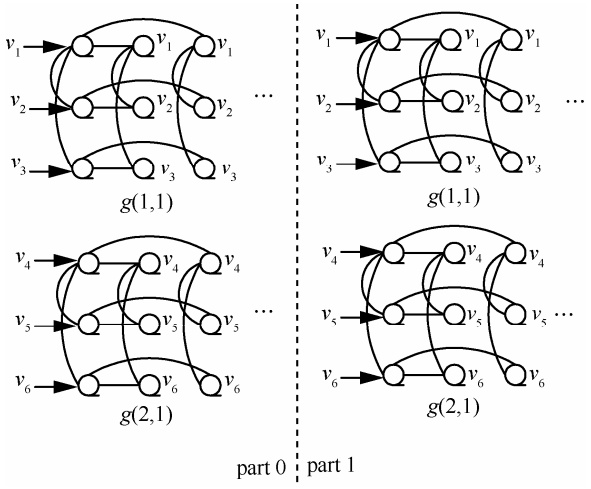


图 3　3×3 BSN-MOT 中经过行树广播的赋值示例

时间复杂度：从算法可以看到 step3 及 step5 各执行一次组间移动，step2 通过广播处理器中的数据至本地同行的其他处理器而消耗了 $\log^n$ 次组内移动，step4 与 step2 过程相似，同样需要 $\log^n$ 次组内移动，由此算法，可得出行树广播的时间复杂度为 $2\log^n$ 次组内移动及 2 次的组间移动。

注意：本文在分析时间复杂度时将以组间移动及组内移动为度量标准，并且，组间移动及组内移动是分开描述的。执行组间移动比执行组内移动需要消耗更大的网络带宽，且组间链接与组内链接在数据传输及等待时间上也是有差异的。

3.2　列树广播(V_y, A)

在列树广播算法中，数据 v_y 将被广播至 BSN-MOT 的第 y 列。算法描述如下。

step1　赋值 $v_{(g_y-1)n+p_y}$ 至不同部的第 1 行组的第 1 行处理器，则寄存器 $B(i,1,g_y,1,p_y)$ 都有了赋值。

step2　通过列树链接，广播寄存器 B 内的值至同列的其他处理器，如此，第 1 行组的所有处理器都有了相应的值。

step3　对整个网络通过组间链接执行组间移动。这样，不同部的所有第 1 行处理器都拥有了一

数值。

step4 广播处理器中的数值至同列的其他处理器。

step5 再次执行组间移动。

step6 在 step1 中广播处理器中的数值至同列的其他处理器。

step7 算法结束。最后，所有 y 列的处理器都有了数值 v_y。

图 4 给出了 BSN-MOT$_3$ 在经过列树广播之后的赋值示例。

时间复杂度：与行树广播算法分析相似，此算法的时间复杂度为 $2\log^n$ 次组内移动及 2 次的组间移动。

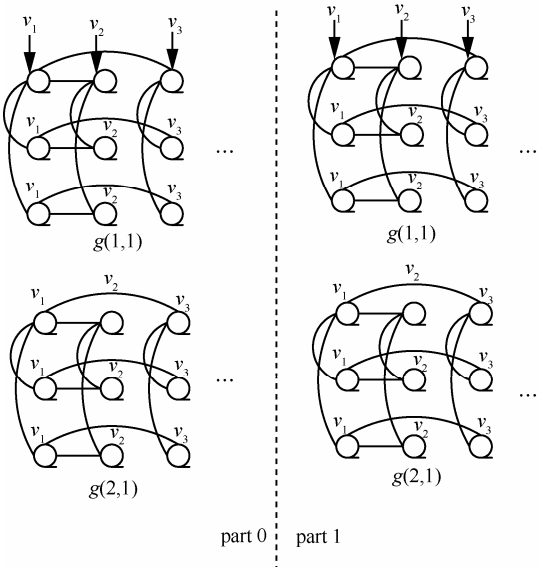


图 4 3×3 BSN-MOT 中经过列树广播的赋值示例

3.3 单向广播

在此算法中，数据首先被初始放置在一处理器 (i,g_x,g_y,p_x,p_y) 中，算法执行后，数据将被广播至其他 $2n^4-1$ 个处理器。算法如下。

step1 通过行树链接及列树链接，广播 (i,g_x,g_y,p_x,p_y) 中的数据至本地组的其他处理器。

step2 对非空组执行一次组间移动。

step3 在 $\bar{i}$ 中，广播本地组中的数据至组内其他处理器。

step4 通过组间链接，广播 $\bar{i}$ 内的数据至 i 部。

step5 算法结束。

时间复杂度：执行完 step2，部 $\bar{i}$ 中的每组内的一处理器都含有了数据副本，而经过 step4，数据在整个 BSN-MOT 中得到广播。在此算法中，step1 及 step3 各自需要 $2\log^n$ 次组内移动，而 step2 及 step4 则各需要一次组间移动，算法的时间复杂度为 $4\log^n$ 次组内移动及 2 次组间移动。

3.4 数据求和

在此算法中，每个处理器都含有一数据值，最后，$2n^4$ 个数据的和值被放置在处理器 (0,1,1,1,1) 中。算法描述如下。

step1 通过行树及列树链接，对每部中的每组执行本地求和，之后，将和值转至处理器 $A(i,g_x,g_y,1,1)$。

step2 对含有本地和值的处理器执行组间移动。

step3 分别在处理器 $A(0,1,1,p_x,p_y)$ 及 $A(1,1,1,p_x,p_y)$ 上执行本地求和算法，之后，在 i 部，将和值移动至 $A(i,1,1,1,1)$。

step4 将处理器 $A(1,1,1,1,1)$ 上的和值移动至 $A(0,1,1,1,1)$，然后将 $A(0,1,1,1,1)$ 上的两和值相加。

step5 算法结束。$2n^4$ 个处理器中的数据的总和值即被存放在处理器 (0,1,1,1,1)。

时间复杂度：执行完 step2，每部中组 $g(1,1)$ 的每个处理器都含有一本地和值，在 step4 中，总的和值被存放在了处理器 $A(0,1,1,1,1)$。此算法的时间复杂度即为 $4\log^n$ 次组内移动及 2 次组间移动。

3.5 矩阵乘法

算法初始，首先利用 GRM(group row mapping)[5]方式将 2 个 $N(N=n^2)$ 阶矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 映射到含有 $2N^2$ 个处理器的 BSN-MOT 中，矩阵 $\boldsymbol{A}$ 被映射到 BSN-MOT 的 0 部，矩阵 $\boldsymbol{B}$ 被映射到 1 部。图 5 给出了如何利用 GRM 将矩阵映射到 BSN-MOT 的分图示例。在算法中，本文实现 BSN-MOT 上矩阵 $\boldsymbol{A}$ 及 $\boldsymbol{B}$ 的相乘，其乘积即为 $C_{ij}=\sum_{m=0}^{N-1}A_{im}B_{mj}$。其中，矩阵第 i 行 j 列的元素被映射到组 i 的第 j 个处理器。算法如下。

step1 将部 1 中的 $\boldsymbol{B}$ 值通过组间链接移至 0 部。

step2 在部 0 中，每个处理器都通过行树链接及列树链接将本地组中的 $\boldsymbol{A}$、$\boldsymbol{B}$ 值集结。

step3 将集结的 $\boldsymbol{B}$ 值沿组间链接移至部 1 中。

step4 将集结的 $\boldsymbol{A}$ 值也从处理器 $(0,i,j)$ 移至 $(1,i,j)$。

step5　在部 1 中，每个处理器都计算其内的 ***A*** 值与 ***B*** 值的对应内积。

step6　算法结束。乘积矩阵在部 1 中生成。

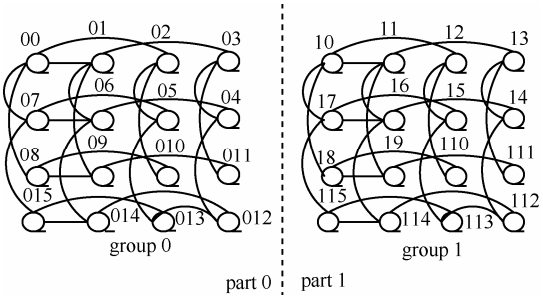


图 5　GRM 矩阵元素映射至 BSN-MOT$_4$ 的分布示例

由上述算法得知，算法开始时，矩阵元素 A_{ij} 及 B_{ij} 被分别放置在处理器 $(0,i,j)$ 及 $(1,i,j)$ 中。执行完 step1，$(0,i,j)$ 含有 A_{ij} 及 B_{ij}；在 step2 中，每组中的每个处理器都含有矩阵 ***A*** 的第 i 行值及 ***B*** 的第 j 列值；执行完 step3，部 1 中每组的处理器都含有 ***B*** 的第 j 列值。而在 step5，乘积矩阵的元素 C_{ij} 在部 1 中得出。

时间复杂度：值得说明的是，在 step2 中，本地组中的所有 ***A*** 值与 ***B*** 值都被集结，此步需要 $4\log^n$ 次组内移动。最后，此算法的时间复杂度即为 2 次的组间移动及 $4\log^n+2$ 次的组内移动。

注意：可以利用此映射方式及 BSN-MOT 计算两向量的乘积，如列向量×行向量，行向量×列向量，或行向量×矩阵以及矩阵×列向量。

3.6　排序

在此，本文对 n^2 个数据元素 $z_0,z_1,\cdots,z_{n^2-1}$ 进行排序。假设 BSN-MOT 的每个处理器都含有 3 个寄存器 A、B、C，寄存器 A 及 B 用来存储数据元素，C 用来存储数据排名。首先，寄存器 A 的初始赋值如下：$(i,g_{xy},A_{00})\leftarrow z_{(nx+y)}$，其中，$0\leqslant x,y\leqslant n-1$。算法如下。

step1　对任意 x 及 y，$0\leqslant x,y\leqslant n-1$，广播 (i,g_{xy},A_{00}) 中的值至本地组的其他处理器。

step2　对任意 x 及 y，$0\leqslant x,y\leqslant n-1$，将 (i,g_{xy},A_{kl}) 赋值给 (i,g_{xy},B_{kl})。

step3　对任意 x,y,k,l，$0\leqslant x,y,k,l\leqslant n-1$，对 (i,g_{xy},B_{kl}) 中的数据执行组间移动。

step4　对任意 x,y,k,l，$0\leqslant x,y,k,l\leqslant n-1$，如果 $(i,g_{xy},A_{kl})\geqslant(i,g_{xy},B_{kl})$ 则 $(i,g_{xy},C_{kl})=1$；反之，$(i,g_{xy},C_{kl})=0$。

step5　对寄存器 C 中的数据执行本地组的求和，之后和值寄存在处理器 (i,g_{xy},C_{00})。

step6　对任意 x 及 y，$0\leqslant x,y\leqslant n-1$，广播处理器 (i,g_{xy},C_{00}) 的数据至本组内的其他处理器。

step7　对任意 x,y,k,l，$0\leqslant x,y,k,l\leqslant n-1$，如果 $(i,g_{xy},C_{kl})=nk+l+1$，对 (i,g_{xy},A_{kl}) 及 (i,g_{xy},C_{kl}) 执行组间移动，随后各自在本地组内进行广播。

step8　对任意 x,y,k,l，$0\leqslant x,y,k,l\leqslant n-1$，对 (i,g_{xy},A_{kl}) 执行组间移动。

step9　算法结束。

图 6 给出了数据集{60，45，65，55，23，18，59，78，3}在 BSN-MOT$_3$ 上执行完排序算法后的分布示例。

时间复杂度：可以看出，在 step3、step7 及 step8 中各执行了一次组间移动，而 step1、step4、step5 及 step7 则各自执行了 $2\log^n$ 次组内移动。此算法的时间复杂度则为 $8\log^n$ 次组内移动及 3 次的组间移动。

3.7　最短路径路由

本节提出一种应用在 MOT 上的基于 SPR[2]的新的最短路径路由算法，称之为 E-SPR，随后将 E-SPR 应用在整个 BSN-MOT 中。

首先，将 E-SPR 应用在 MOT 中。本文定义 (x,y) 为顶点 u 的二维二进制表示，其中，“x”表示 u 的行索引，“y”表示 u 的列索引。对整个 MOT，其根节点被标志为 $(1,1)$。对任意的内部节点 u，它的左孩子被标记为 $2u$，右孩子被标记为 $2u+1$。接下来便给出从节点 u 到节点 v 的最短路径路由算法，其中，u 用 (x_1,y_1) 表示，v 用 (x_2,y_2) 表示。在路由过程中，首先遍历行链接的二进制树，直到满足 $x_1=x_2$，接下来，便开始搜寻列链接的二进制树，直到 $y_1=y_2$。现在，先以一个行链接的二进制搜索为例：如果 x_1 不是 x_2 的前缀，则将 x_1 移至它的父节点，并将父节点索引标为 x_1；然而，如果 x_1 是 x_2 的前缀，则将此前缀从 x_2 消除，之后观察 x_2 剩余比特的最左边的一位，如果此比特是 1，则将 x_1 移至它的右孩子，并且将右孩子的二进制索引标为 x_1；若此比特是 0，则将 x_1 移至它的左孩子，并将左孩子的索引标为 x_1。图 7 就给出了 MOT$_6$ 中从节点 u(10,1) 到目标节点 v(110,101) 的最短路由路径。

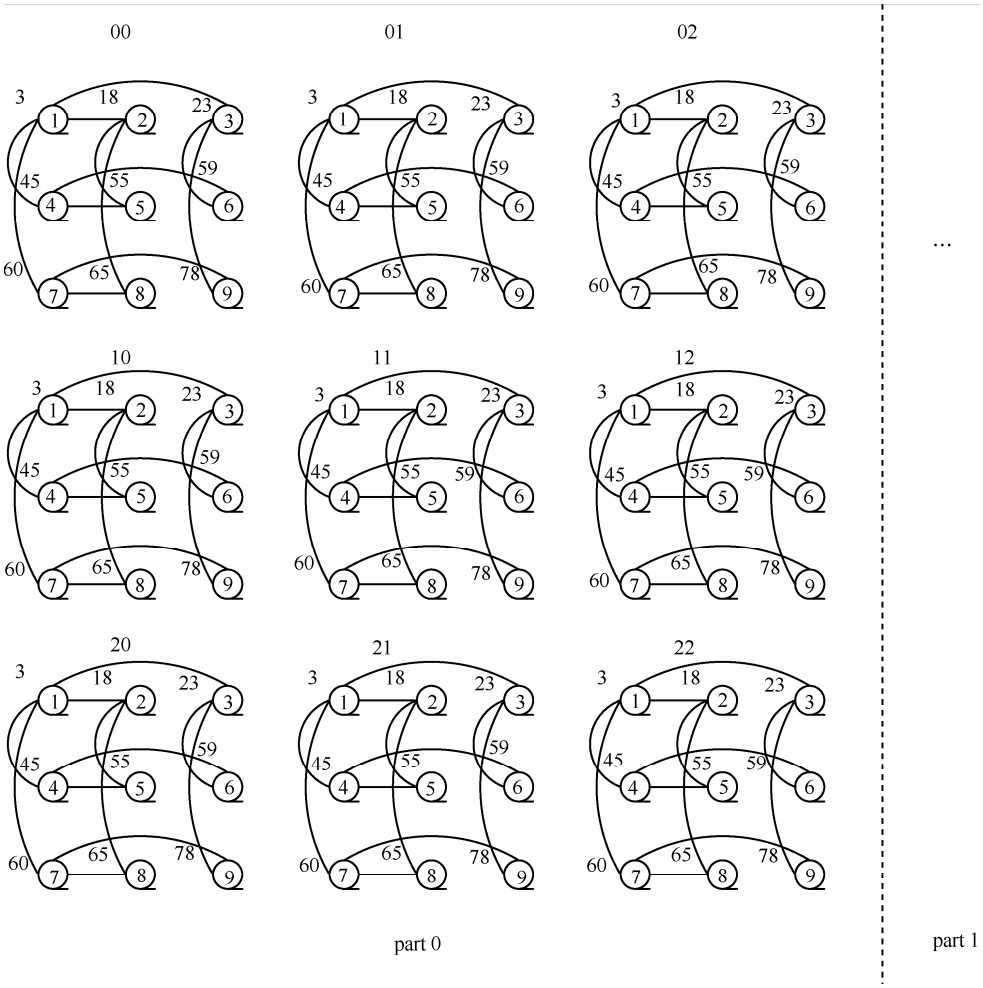


图 6　排序之后寄存器 A 及 C 内的数值示例

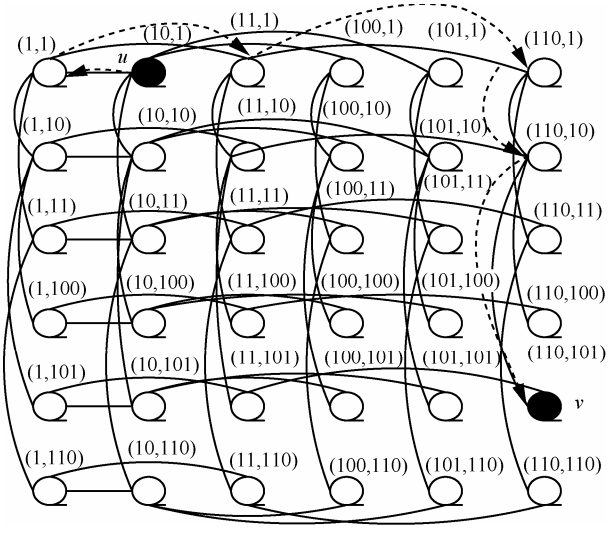


图 7　E-SPR: MOT_6 中从 u 到 v 的最短路由路径示例

下一步，本文运用 E-SPR 至 BSN-MOT。假使源节点及目标节点分别为 $(i_1, g_{x1}, g_{y1}, p_{x1}, p_{y1})$ 及 $(i_2, g_{x2}, g_{y2}, p_{x2}, p_{y2})$，则 E-SPR 算法如下。

step1　在部 i_1 中应用 E-SPR 至组 $g(x_1, y_1)$，直到寻到节点 $(i_1, g_{x1}, g_{y1}, g_{x2}, g_{y2})$。

step2　执行组间移动，即可寻到节点 $(i_2, g_{x2}, g_{y2}, g_{x1}, g_{y1})$。

step3　在部 i_2 中再次应用 E-SPR 至组 $g(x_2, y_2)$，则目标节点 $(i_2, g_{x2}, g_{y2}, p_{x2}, p_{y2})$ 即可遍历到。

step4　算法结束。

时间复杂度：此算法的时间复杂度为 $8\log^n$ 次组内移动及 1 次的组间移动，但是，倘若源节点及目标节点是处于同一部中，则算法的时间复杂度为 $8\log^n$ 次的组内移动及 2 次的组间移动。

3.8　多项式求根

多项式求根是许多实践应用中重要的计算步骤之一，如在并行计算的负载均衡策略中[17]，多项

式求根在基于多项式的迭代模型方法中是非常关键的一步。

首先，N 次多项式可表示如下

$$P_N(x)=a_0x^N+a_1x^{N-1}+a_2x^{N-2}+\ldots+a_{N-1}x+a_N \quad (1)$$

则根据 Durand-Kerner 迭代模型，其根的表示形式为[18]

$$x_i^{(k+1)}=x_i^{(k)}-\frac{P_N(x_i^{(k)})}{\prod_{j=1,j\neq i}^{N}(x_i^{(k)}-x_j^{(k)})},\quad i=1,2,\cdots,N \quad (2)$$

根据上面的公式可以看出，$x_i^{(k)}$ 表示根的第 k 个近似值。Jana[8]已经提出了基于 OTIS-Mesh 的多项式求根算法，在此，提出基于 BSN-MOT 的多项式求根算法。首先，假设每个处理器都含有 4 个寄存器，即 A、B、C、D。

算法执行如下。

step1　对任意的 x，k，$1\leqslant x,k\leqslant\sqrt{N}$

$$A(i,g_x,1,p_x,1)\leftarrow z_{\sqrt{N}(x-1)+k}$$

step2　通过行树链接，将寄存器 A 内的值广播至本地组的其他同行的处理器。

step3　对任意的 y，l，$1\leqslant y,l\leqslant\sqrt{N}$

$$B(i,1,g_y,1,p_y)\leftarrow z_{\sqrt{N}(l-1)+y}$$

step4　通过列树链接，将寄存器 B 内的值广播至本地组的其他同列处理器。

step5　对 A 及 B 各执行组间移动。

step6　在每组中，广播 A 值至本地组的同行处理器，同时，广播 B 值至本地组的同列处理器。

step7　对 BSN-MOT 的每个处理器：

如果 A 中的值不等于 B 中的值，

则 $A\leftarrow A-B$；

反之，$A\leftarrow 1$。

step8　在每部中的每组，将同行的所有 A 值相乘，乘积存储在 $A(i,g_x,g_y,p_x,1)$。

step9　对 A 值执行组间移动。

step10　不同于 step8，将部 0 中第 1 列组中的所有处于同一行的 A 值相乘，其乘积放在 $A(0,g_x,1,p_x,1)$。

step11　对任意的 $x,k=1\text{ to }\sqrt{N}$

$$C(0,g_x,1,p_x,1)\leftarrow z_{\sqrt{N}(x-1)+k}$$

$$D(0,g_x,1,p_x,1)\leftarrow r_{\sqrt{N}(x-1)+k}$$

step12　对任意的 $x,k=1\text{ to }\sqrt{N}$

$$A(0,g_x,1,p_x,1)\leftarrow C(0,g_x,1,p_x,1)-\frac{D(0,g_x,1,p_x,1)}{A(0,g_x,1,p_x,1)}$$

时间复杂度：此算法的时间复杂度即为 $6\log^n$ 次组内移动及 2 次的组间移动。

3.9　BSN-MOT 与 MMT 及 OMULT 的性能比较

本节将 BSN-MOT 与另外 2 种流行的双层树形网络即 MMT 及 OMULT 进行比较来说明 BSN-MOT 的快速有效性(在文献[5]中，Jana 已经分析了 MMT 的拓扑性质及架构的有效性；Islam[4]也证明了 OMULT 的性能优势)。表 1 列出了 3 种架构下的各算法的时间复杂度比较。从表中数据可以看出，BSN-MOT 是比 MMT 及 OMULT 更快速、有效的网络架构，而算法有效的关键点就是利用了 BSN 特殊的结构体系，以致降低了其通信流量。

4　结束语

基于 BSN 的一系列的理想特性，诸如扩展性，继承因子网络的正则性、点传递性、Cayley 图、哈密尔顿圈及极大容错性等，在并行计算中研究其上的基本通信操作算法是非常有意义的。本文提出一种新型的双层架构 BSN-MOT，并研究了其上的一些拓扑性质及基本的通信及应用操作算法，如行、列树广播、数据求和、矩阵相乘、排序及多项式求根等，并理论分析了每种算法的时间复杂度，且通过列表研究给出了 BSN-MOT 与同为基于树的双重架构 MMT 及 OMULT 的各算法的时间复杂度比较。

表 1　**BSN-MOT、MMT 和 OMULT 的算法时间复杂度比较**

操作	MMT		OMULT			BSN-MOT		
	数据规模	组内链接	数据规模	组内链接	组间链接	数据规模	组内链接	组间链接
行\列树广播	n^2	$2\log^n+3$	n	$2\log^n$	3	n^2	$2\log^n$	2
单向广播	1	$7\log^n+2$	1	$6\log^n$	2	1	$4\log^n$	2
数据求和	n^4	$4\log^n+11$	n^3	$6\log^n$	2	$2n^4$	$4\log^n$	2
矩阵乘法	$n^2\times n^2$	$O(\log^n)+n+1$	$n\times n$	$6\log^n$	8	$n^2\times n^2$	$4\log^n+2$	2
稀疏排序	n^2	$4\log^n+o(n)$	n	$5\log^n$	3	n^2	$8\log^n$	3

从数据结果可以看出，BSN-MOT 是比 MMT 及 OMULT 更快速、有效的网络架构。而未来的工作将基于 BSN-MOT 的各操作算法是否可以延伸至其他 BSN 进行展开研究。

参考文献：

[1] 陈卫东,肖文俊. Biswapped 网络(BSN)的拓扑性质研究:点对称性和极大容错性[J].计算机学报,2010,33(5): 822-832.
CHEN W D, XIAO W J. Topological properties of biswapped networks (BSN): node symmetry and maximal fault tolerance[J]. Chinese Journal of Computers, 2010, 33(5): 822-832.

[2] KEMAL K, FERNANDEZ A. Mesh-connected trees: a bridge between grids and meshes of trees[J]. IEEE Transactions on Parallel and Distributed Systems, 1996, 7(12):1281-1291.

[3] JANA P K. Multi-mesh of trees with its parallel algorithms[J]. Journal of Systems Architecture, 2004, 50(4):193-206.

[4] ISLAM R, AFROZ N, BANDYOPADHYAY S, *et al*. Computational geometry on optical multi-trees (OMULT) computer system[A]. CCCG 2005[C]. 2005.150-154.

[5] WANG C F, SAHNI S. Matrix multiplication on the OTIS-Mesh optoelectronic computer[J]. IEEE Transactions on Computers, 2001, 50(7):635-646.

[6] SAHNI S, WANG C. BPC permutations on the OTIS-hypercube optoelectronic computer[J]. Informatica (Ljubljana), 1998,22(3): 263-269.

[7] SAHNI S, WANG C. BPC permutations on the OTIS-mesh optoelectronic computer[A]. Proceedings of the 1997 4th International Conference on Massively Parallel Processing Using Optical Interonnections, MPPOI'97[C]. Montreal, Can,1997.

[8] JANA P K. Polynomial interpolation and polynomial root finding on OTIS-mesh[J]. Parallel Computing, 2006,32(4):301-312.

[9] RAJASEKARAN S, SAHNI S. Randomized routing, selection, and sorting on the OTIS-mesh[J]. IEEE Transactions on Parallel and Distributed Systems, 1998, 9(9):833-840.

[10] WANG C, SAHNI S. Image processing on the OTIS-mesh optoelectronic computer[J]. IEEE Transactions on Parallel and Distributed Systems, 2000,11(2):97-109.

[11] WANG C F, SAHNI S. Basic operations on the OTIS-mesh optoelectronic computer[J]. IEEE Transactions on Parallel and Distributed Systems, 1998,9(12):1226-1236.

[12] OSTERLOH A. Sorting on the OTIS-mesh[A]. Proceedings of the International Parallel Processing Symposium, IPPS[C]. 2000.269-274.

[13] WEI W, XIAO W. Matrix multiplication on the biswapped-mesh network[A]. SNPD 2007: 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing[C]. Qingdao, China, 2007.

[14] SUN L, TONG C, SU H. New mapping scheme of matrix algorithms on the Biswapped network[J]. Journal of Computational Information Systems, 2013,9(13):5371-5378.

[15] YE H, XIAO W, WU J. Broadcasting on the BSN-hypercube network[A]. 2009 2nd International Conference on Information and Computing Science, ICIC 2009[C]. Manchester, United Kingdom, 2009.

[16] CHEN W, CHEN G, HSU D F. Generalized diameters of the mesh of trees[J]. Theory of Computing Systems, 2004,37(4):547-556.

[17] DIEKMANN R, FROMMER A, MONIEN B. Efficient schemes for nearest neighbor load balancing[J]. Parallel Computing, 1999, 25(7): 789-812.

[18] FREEMAN T L. Calculating polynomial zeros on a local memory parallel computer[J]. Parallel Computing, 1989,12(3):351-358.

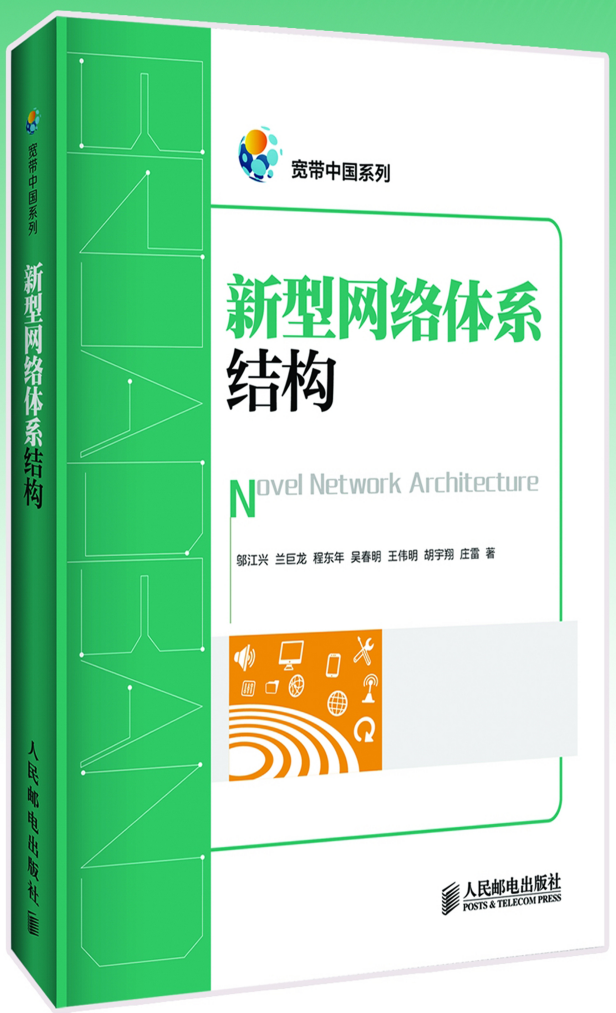
作者简介：

李江昀（1977-），男，山西太原人，北京科技大学副教授，主要研究方向为工业数据通信与控制网络、网络控制系统。

孙丽婷（1985-），女，山东邹城人，北京科技大学博士生，主要研究方向为集群优化、并行计算及负载均衡。

新型网络体系结构

Novel Network Architecture



互联网是人类20世纪最伟大的基础性科技发明之一。作为信息传播的新载体、科技创新的新手段，互联网的普及和发展改变了人类的生活和生产方式，引发了前所未有的信息革命和产业革命。互联网业已成为与国民经济和社会发展高度相关的重大信息基础设施，其发展水平是衡量国家综合实力的重要标志之一。

然而，当今互联网受初始设计理念的限制以及发展历史特点的影响，基础网络层面结构简单僵化、功能单一，与丰富多样的上下层功能之间存在巨大的反差，不具备增强其能力的基本智能，难以从根本上应对亟待解决的挑战性问题。因此，各国科学家提出了大量新型网络体系结构，希望解决当前互联网面临的问题。在这个新旧交替的关键时期，梳理和研究现有的新型网络体系结构，从国家、商业和技术的角度来看都显得愈发重要和迫切。

本书在介绍网络体系结构概念的基础上，对现有典型网络体系结构的基本原理以及国内外新型网络体系结构的研究现状进行了全面而系统的介绍。基于对网络体系结构的整体认识，最后凝练并提出了对未来中国宽带之路的思考。

本书取材新颖、内容翔实、实用性强，反映了国内外新型网络体系结构研究的现状与未来，适合于从事通信、计算机网络体系设计与研究的广大工程技术人员阅读，也可作为大专院校通信、计算机等专业和相关培训班的教材或教学参考书。

人民邮电出版社

书号：ISBN 978-7-115-34230-0

联系电话：010-81055487