云南大学学报 (自然科学版), 2004, 26 (2): 127~131 CN 53-1045/N ISSN 0258-7971
Journal of Yunnan University

多播路由协议 PIM-SM 的扩展 Petri 网形式分析*

陆正福, 王 敏, 王国栋
(云南大学 数学系, 云南 昆明 650091)

摘要: PIM-SM 是一类重要的多播路由协议. 在对 PIM-SM 详细的机制分析的基础上用扩展的库所/变迁网对其进行形式化的描述, 为其计算机仿真和具体实现奠定了形式化和自动化的基础.

关键词: IP 多播路由; PIM-SM 协议; 扩展 Petri 网; 稀疏模式
中图分类号: TP 393; TP 302 文献标识码: A 文章编号: 0258-7971(2004)02-0127-05

随着 IP 网络上的多播应用如网上金融、网络会议、视频点播、分布协同工作、分布计算、分布式仿真等的出现, IP 多播变得越来越重要. IP 多播以其节约带宽、并行接收、节省发方资源、减轻收方负荷等优势成为一类重要的高性能网络技术, 成为多种网络计算应用的基础支持技术. 多播路由是 IP 多播的重要组成部分, 已经出现了多种多播路由协议, 其中 PIM 是针对多播路由的可扩展性问题提出的, 获得了主流厂商如 Cisco 等公司的支持. 根据接收方的分布情况, PIM 又分为"稀疏模式"(即 PIM-SM) 和"密集模式"(即 PIM-DM). 其中 PIM-SM 沿着 2 个方向处于发展之中, 其一是 Bi-PIM, 其二则是 PIM-SM 自身的继续完善[1].

本文的研究对象是 PIM-SM, PIM-SM 的协议规范(RFC2362) 已发展到 2.0 版本[2], 并处于继续发展[1] 之中.

IP 多播领域的协议是带有复杂行为的异步并发的有限状态系统, 需要借助一定的形式化工具进行分析, 文献[3~5] 利用 EPTN 分别分析了多播组动态管理协议 IGMP 和前面提及的 Bi-PIM. 本文在文献[1, 2] 的非形式化分析的基础上, 沿用文献[3~5] 的方法, 对 PIM-SM 进行了详细的机制分析, 并利用 EPTN 对路由器收发有关消息的接口进行了形式化描述. 这些工作对协议的详细分析、计算机仿真和具体实现都具有重要的基础性意义.

1 基本定义与概述

1.1 EPTN 的基本定义

定义 1 设四元组 $\sum = (S, T; F, M_0)$ 为 P/T 网, 将其变迁部分 T 进行行为扩充, 加入行为集 A, 得五元组 $\sum' = (S, T; F, M_0, A)$, 称该五元组为扩充行为的 P/T 网, 简称为 EPTN.

在描述协议时, 一般需要对 Petri 网进行扩展[6], 在一些特殊情况下可以不需要扩展. PIM-SM 的 Petri 网描述采用了定义给出的 EPTN. 它包含 5 个基本要素: 库所(以圆圈和库所名称表示)、标识(以圆圈中的黑点表示)、变迁(以竖线和变迁名称表示)、行为(以变迁名称后括号中的内容表示)、流关系.

1.2 PIM-SM 的总体概述 在 PIM-SM 中, 路由器利用加入和删除消息显式加入和离开多播组, 每个多播组有一个确定的会合点 RP (rendezvous point), 这个 RP 是通过自举报文(bootstrap) 机制或静态配置设定的[1,2].

主机向所在物理子网发送 IGMP[3] 成员关系报告请求加入多播组 G, 当指定路由器 DR(当主机所在物理子网由不止一个路由器与互联网连接时, 由指定路由器负责转发多播信息) 收到报告时就其向上游路由器发送 Join/Prune 消息, Join/

* 收稿日期: 2003-08-14
基金项目: 云南省自然科学基金资助项目(2002F0012M); 云南省教育厅科研基金资助项目(0111155); 云南省省校合作项目(19-7).

作者简介: 陆正福(1965-), 男, 安徽人, 副教授, 主要从事网络信息安全、信息编码与网络计算方面的研究.

Prune 消息被逐跳地发往组 G 的 RP, 消息所经过的路由器创建一个(*, G) 状态, 最终形成一棵以 RP 为根的共享树.

源端第 1 次发送数据时, 由其 DR 将分组封装后单播给 RP, RP 对分组进行检测后向源端发送 (S, G) Join/ Prune 消息, 从而建立一条 RP 到源端的有源路, 源端便沿此有源路向 RP 发送数据, 然后由 RP 将收到的分组沿共享树转发给多播组成员.

如果源端发送数据的速率超出某个设定的阈值, 该源端将会加入一棵以它为根的最短路径树, 并从共享树上剪枝.

2 PIM- SM 的具体机制分析及形式化描述

本部分将从共享树加入、源注册、最短路径树切换 3 个方面分析 PIM- SM 的具体机制并建立 EPTN 模型.

2.1 共享树加入 PIM- SM 协议中包含 2 种组播转发树: 以 RP 为根的共享树和以源为根的有源树(又称最短路径树 SPT). 加入组的主机一般先加入共享树, 我们通过图 1 所示的拓扑结构说明共享树的建立过程.

图 1 中, 接收站点 1 和接收站点 2 是欲加入多播组 G 的 2 个端主机. 假设接收站点 1 先于接收站点 2 发出加入请求. 共享树的具体建立过程如下:

接收站点 1 通过 IGMP 成员关系报告加入组 G, R4 在组播路由表中建立(*, G) 状态条目(* 代表所有发送方), 并向 RP 发送 Join 消息, 传送路径上经过的每个路由器(如 R2) 都可以查看 Join 消息并在转发表中创建(*, G) 条目, R2 将 Join 消息到达的接口作为输出接口 oif (outgoing interface), 然后 R2 决定通过哪个接口将 Join 消息转发给 RP, 此接口便成为 R2 的输入接口 iif (incoming interface). 当消息到达 RP 时, 就形成了共享树的一个分支(如图 1 中从 RP 到 R4 的粗黑线所标示).

当接收站点 2 发出 IGMP 成员关系报告加入组 G, 路由器 R5 发送 Join 消息给 R2, 此时 R2 只简单地将新的 oif 加入到这个组的转发条目中(不需将其转发给 RP), 这样新的分支被添加到树上, 最终形成一棵以 RP 为根的共享树(粗黑线表示).

通过以上对共享树加入的机制分析及非形式化描述, 可以得到接收(*, G) 加入/ 删除消息的接口上的 EPTN 描述(图 2).

在图 2 中, 库所集合 $S = \{NI, J, PP\}$, 其中 NI 表示该接口无(*, G) 状态且定时器没有在运行; J 表示该接口已有(*, G) 状态, 且 ET (过时定时器 Expiry Timer) 在运行; PP 表示该接口已收到(*, G) 删除消息, 并在等待看是否有下游路由器的消息使此删除无效, PPT (删除悬挂定时器 PrunePending Timer)、ET 在运行. 变迁集 $T = \{r_J, r_P, t_{PPT}, t_{ET}\}$, 其中 r_J: 收到(*, G) 加入消息; r_P: 收到(*, G) 删除消息; t_{PPT}: PPT 超时; t_{ET}: ET 超时. 一个变迁发生时, 往往伴随着一些行为, 其行为集合 $A = \{start_{ET}, restart_{ET}, cancel_{PPT}, start_{PPT}, s_{echo}\}$, 其中 start 表示启动相应的定时器; restart 表示重启相应的定时器; cancel 表示取消定时器; s_{echo} 是向与此接口相连的子网发送 Prune(*, G) 消息, 其目的是确保子网内的路由器都能收到 Prune (*, G) 消息.

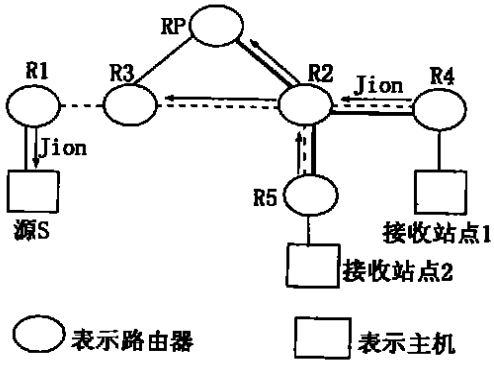


图 1 PIM- SM 共享树加入

Fig. 1 Creation of PIM-SM shared tree

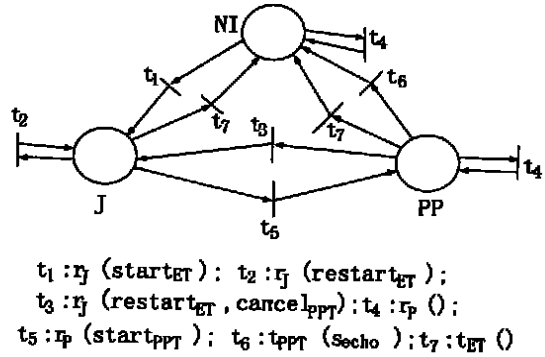
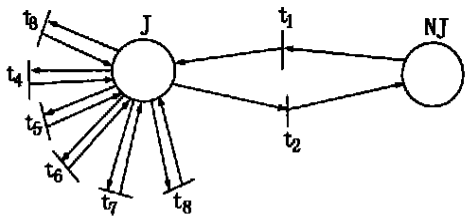


图 2 接收(*, G) 加入/ 删除消息的接口的 EPTN 描述

Fig. 2 EPTN description for receiving (*, G) Join/ Prune messages

类似地, 可得到发送(* , G) 加入/删除消息的接口上的 EPTN 描述(图 3).



t_1 : IF(s_J, set_{JT}); t_2 : NotIF(s_P, $cancel_{JT}$);
t_3 : t(s_J, set_{JT}); t_4 : J(I_{JT});
t_5 : P(D_{JT}); t_6 : c_{RPF} (s_J, set_{JT});
t_7 : c_{MRIB}(S_JToNew, S_PToOld, set_{JT}); t_8 : c_{GenID} (D_{JT})

图 3　发送(* , G) 加入/删除消息的接口的 **EPTN** 描述

Fig. 3 EPTN description for sending (* , G) Join/ Prune messages

在图 3 中, 库所集合 S = { NJ, J}, 其中 NJ 表示该接口未加入共享树, 且 JT (Join Timer) 没有在运行; J 表示该接口已加入共享树, 且 JT 在运行. 变迁集 T = { IF, NotIF, t, J, P, c_{RPF}, c_{MRIB}, c_{GenID}}, 其中 IF: 此路由器的下游至少有 1 个接口; NotIF: 此路由器的下游无接口; t: 定时器超时; J: 看到其他路由器向上游发送同样的加入(* , G) 消息; P: 看到其他路由器向上游发送删除(* , G) 消息; c_{RPF}: 向着 RP 的下一跳路由器改变了; c_{MRIB}: 多播路由信息库中对于向着 RP 的下一跳路由器的信息改变了; c_{GenID}: 向着 RP 的下一跳路由器的 Generation ID 改变了, 也就是说这个路由器失去了状态, 因此状态被刷新. 其伴随的行为集 A = { s_J, set_{JT}, s_p, $cancel_{JT}$, I_{JT}, D_{JT}, S_JToNew, S_pToOld}, 其中 s_J: 发送加入消息; set_{JT}: 设置 JT 的值; s_p: 发送删除消息; $cancel_{JT}$: 取消 JT; I_{JT}: 增加 JT 的值; D_{JT}: 减少 JT 的值; S_JToNew : 向新的下一跳路由器发送加入消息; S_pToOld: 向原来的下一跳路由器发送删除消息.

2.2　源注册　PIM - SM 使用一棵单向的共享树, 组播信息只能沿树向下流动, 因此组播源在首次发送数据时必须设法使它们的信息到达 RP, 以便信息能沿共享树流出, 该过程称为源注册.

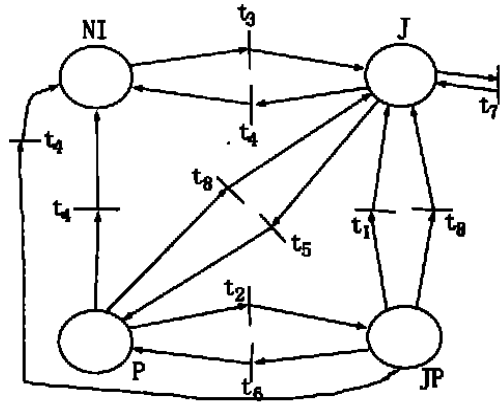
源注册的具体过程为: 由源 S 发出的数据包到达与之相连的路由器, 该路由器找不到匹配的状态条目, 于是它将数据包封装在 Register 消息内并单播给组 G 的 RP. 当 RP 收到 Register 消息后, 它先解封此消息, 对组播信息包进行检测, 根据检测结果的不同可以分出如下情况:

如果组 G 的共享树已存在, RP 就沿共享树向下转发分组, 然后创建一个新的组播路由条目(S, G), 并周期性地向源端发送 Join/ Prune 消息, 源端和 RP 之间的路由器利用这些消息建立并维护(S, G) 状态, 从而在 RP 与 S 之间建立一条有源路.

如果没有组共享树, RP 只是简单地丢弃组播信息包, 并不向源发送加入消息.

如果 RP 收到一个 Register 消息, 但 RP 没有与任何接收者相连, 或 RP 已加入有源树并正在接收数据包, 则当它收到一个 Register 消息后就发 Register- stop 消息, 让源 S 停止向其发送数据.

通过以上对源注册的过程分析及非形式化描述, 可以得到(S, G) 注册消息在 DR 部分的 EPTN 描述(图 4).



t_1 : t_{RST} (add); t_2 : t_{RST} (set_{RST});
t_3 : $register_T$ (add); t_4 : $register_F$ (rm);
t_5 : r_{RS} (rm, set_{RST}); t_6 : r_{RS} (set_{RST});
t_7 : c_{RP} (upd); t_8 : c_{RP} (add, $cancel_{RST}$)

图 4　(S, G)注册消息在 **DR** 部分的 **EPTN** 描述

Fig. 4 EPTN description of (S, G) register at a DR

在图 4 中, 其库所集合 S = { NI, J, JP, P}, 其中 NI 表示该接口无任何消息, 没有定时器在运行; J 表示此接口可通过注册通道 Register Tunnel (将封装后的消息发向 RP 的接口, 是一个虚拟接口) 与 RP 通信, 没有定时器在运行; JP 表示 RT 已被移除, 而 DR 在等待看是否它会被重新添加, 且 RegisterStop 消息定时器(RST) 在运行; P 表示 RT 已被移除, 且 RST 在运行. 变迁集 T = { t_{RST}, $register_T$, $register_F$, r_{RS}, c_{RP}}, 其中 t_{RST}: RST 超时, $register_T$ 和 $register_F$ 分别表示是否满足源注册的条件

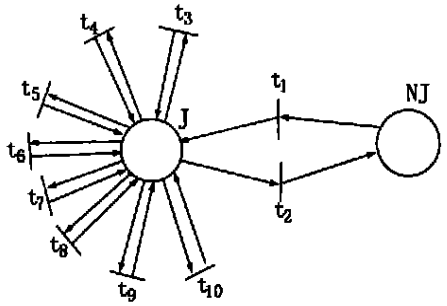
(即此路由器是否为指定路由器 DR 以及 S 是否为直连的源端), r_{RS}: 收到 RegisterStop 消息, c_{RP}: RP 改变了. 其行为集合 $A = \{add, set_{RST}, rm, upd, cancel_{RST}\}$, 其中 add: 添加 RT, set_{RST}: 设置 RST 的值, rm: 移除 RT, upd: 更新 RT, $cancel_{RST}$: 取消 RST.

2.3 最短路径树切换 对一个特定的源, PIM-SM 能够把末跳 DR(即直接与成员主机相连的 DR)从共享树切换到 SPT. 触发这一转变的标准有多种, 其中一个是利用数据速率来实现. 如果源端发送数据的速率超出了某个设定的阈值, 最后一跳 DR 就加入 SPT.

仍以图 1 所示的实例来说明, 共享树的下游节点(如 R4、R5)向源端发送(S, G) - Join/Prune 消息, 此消息沿一条最短路径传向源端, 所经的路由器创建(S, G) 状态, 并在各自的组播转发表中增加相应的(S, G) 条目的输入和输出接口, 从而生成了以源端 S 为根的树(见虚线部分).

此时, R4, R5 能通过共享树和 SPT 接收源 S 的信息, 这将导致带宽的浪费, 因此需要告诉 RP 剪枝来自共享树的(S, G)组播信息, 这种特定的剪枝称为(S, G) RP 位剪枝. R4, R5 沿通往 RP 的共享树向上发送(S, G) RP 位剪枝消息, 收到该消息后, RP 更新其组播转发状态, 它向 S 发送(S, G) RP 位剪枝消息, 将(S, G) SPT 从共享树上剪枝.

通过以上分析可以得到接收(S, G) 加入/删除消息的接口上的 EPTN 图(图 5).



$t_1: IF(s_J, set_{JT}); t_2: NotIF(s_P, cancel_{JT});$

$t_3: t(s_J, set_{JT}); t_4: J_{(S,G)}(I_{JT});$

$t_5: P_{(S,G)}(D_{JT}); t_6: c_{RPF}(S_J, set_{JF});$

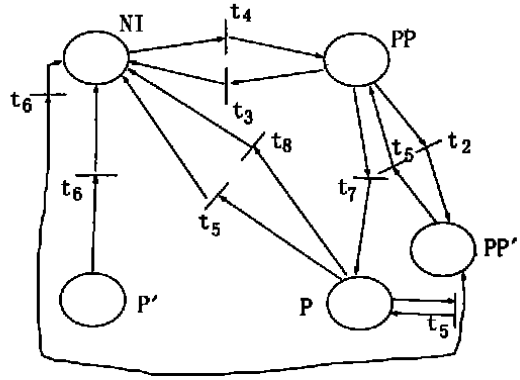
$t_7: c_{MRIB}(S_JToNew, S_PToOld, set_{JT});$

$t_8: c_{GenID}(D_{JT}); t_9: P_{(S,G,rpt)}(D_{JT}); t_{10}: P_{(*,G)}(D_{JT})$

图 5 发送(S, G)加入/删除消息的接口的 **EPTN** 描述

Fig. 5 EPTN description for sending (S, G) Join/Prune messages

该图与发送(*, G) 加入/删除消息的接口上的 EPTN 图相似. 其库所集 $S = \{NJ, J\}$, 其中 NJ 表示该接口未加入最短路径树, 且没有定时器在运行; J 表示该接口已加入最短路径树, 且加入定时器 JT 在运行. 变迁集 $T = \{IF, NotIF, t, J_{(S,G)}, P_{(S,G)}, P_{(S,G,rpt)}, P_{(*,G)}, c_{RPF}, c_{MRIB}, c_{GenID}\}$, 其中 IF, NotIF, t, c_{RPF}, c_{MRIB}, c_{GenID} 解释同前, $J_{(S,G)}$: 看到其他路由器向上游发送同样的加入(S, G) 消息, $P_{(S,G)}$, $P_{(S,G,rpt)}$, $P_{(*,G)}$: 看到其他路由器向上游发送相应的删除消息. 其伴随的行为集 $A = \{s_J, set_{JT}, set_{SPT}, s_p, cancel_{JT}, I_{JT}, D_{JT}, S_JToNew, S_pToOld\}$, 其中 set_{SPT}: 将(S, G) 中的 SPT 位设为 0, 其余解释同前.



$t_1: r_{J(*,G)}(); t_2: r_{J(*,G)}{}_{PPT}(cancel_{PPT});$

$t_3: r_{J(s,G,rpt)}(cancel_{PPT}, cancel_{ET});$

$t_4: r_{P(s,G,rpt)}(start_{PPT}, start_{ET});$

$t_5: r_{P(s,G,rpt)}(restart_{ET});$

$t_6: E(cancel_{PPT}, cancel_{ET}); t_7: t_{PPT}();$

$t_8\ t_{ET}(cancel_{PPT}, cancel_{ET})$

图 6 接收(S, G, rpt) 加入/删除消息的接口上的 **EPTN** 描述

Fig. 6 EPTN description for receiving (S, G, rpt) Join/Prune messages

类似地, 可得到接收(S, G, rpt) 加入/删除消息的接口上的 EPTN 描述. 其库所集 $S = \{NI, P, PP, P', PP'\}$, NI, PP 的解释与接收(*, G) 加入/删除消息的接口上的状态的解释相同; P 表示该接口已从共享树上被剪枝, 即只能通过最短路径树转发, 且 ET 在运行; P′ 是到 P 状态的过渡状态, 如果收到(*, G) 加入消息, 将先进入 P′ 状态, 如果后来收到(S, G, rpt) 删除消息就转化为 P 状态, 如果没收到就转化为 NI 状态, ET 在运行; PP′ 状态是到 PP

状态的过渡状态, 且 ET 在运行. 其变迁集 $T = \{r_{J(*,G)}, r_{J(S,G,rpt)}, r_{P(S,G,rpt)}, E, t_{PPT}, t_{ET}\}$, 其中 E: 最后没有收到删除消息, 其余解释同前. 事件集 $A = \{start_{PPT}, start_{ET}, restart_{ET}, cancel_{PPT}, cancel_{ET}\}$.

3 结　论

本文对多播路由协议 PIM – SM 进行了详细的机制分析, 并利用扩展的库所/变迁网给出了该协议中的消息到来接口上及发送消息的接口上的形式化分析. 将各个 EPTN 模型加以合成, 可以得到 PIM – SM 的完整的形式化描述. 这些研究工作对于深入理解 PIM – SM、分析改进协议、计算机仿真和具体实现协议都起着不可或缺的作用.

参考文献:

[1] ENNER B, HANDLEY M, HOLBROOK H, et al. Protocol independent multicast-sparse mode (PIM – SM): protocol specification (revised) [S], Work In Progress, < draft – ietf – pim – sm – v2 – new – 07. txt >, March, 2003.

[2] ESTRIN D, FARINACCI D, HELMY A, et al. Protocol independent multicast-sparse mode(PIM – SM): protocol specification[S]. IETF RFC 2362, 1998.

[3] 陆正福, 于光德, 李亚东, 等. 基于 Petri 网的多播组动态管理协议 IGMP 的形式化分析[J]. 计算机应用, 2002, 22(7): 12—14.

[4] 陆正福, 王　敏. Bi– PIM 协议的原理分析及其 Petri 网模型[J]. 计算机工程, 2004, 30(2): 124—126.

[5] 陆正福, 于光德, 李亚东, 等. 多播组动态管理协议 IGMP 的形式化分析[J]. 云南大学学报(自然科学版), 2002, 24(4): 256—261.

[6] 袁崇义. Petri 网原理[M]. 北京: 电子工业出版社, 1998.

Formal analysis of multicast routing protocol PIM-SM based on extended Petri net

LU Zheng-fu, WANG Min, WANG Guo-dong

(Department of Mathematics, Yunnan University, Kunming 650091, China)

Abstract: PIM-SM is an important class of multicast routing protocol. It is presented a formal protocol description of PIM-SM with extended place/transition net on the base of detailed mechanism analysis, thus lay a formalization and automation foundation for computer simulation and implementation of the PIM-SM.

Key words: IP multicast routing; PIM-SM protocol; extended Petri net; sparse mode