大规模发布/订阅系统中的可靠性模型*)

董 飚[1] 陈金辉[2] 孙亚民[1]
(南京理工大学计算机科学与技术学院 南京 210094)[1]
(南京信息工程大学信息与控制学院 南京 210044)[2]

摘 要 大多数 Internet 上的大规模发布/订阅系统,其覆盖网络是不可靠的,系统的可靠性和处理故障的能力是一个挑战性的问题。基于轨迹序列和线性时态逻辑定义系统的可靠性条件,是路由算法可靠性分析的基础。设计了崩溃/恢复模式的路由协议,维持一致的、共享的系统状态,有效地处理具有局部性、临时性的路由器故障和链路故障。
关键词 可靠性,轨迹序列,崩溃/恢复,路由协议

Modeling of Reliability in Large Scale Publish/Subscribe Systems

DONG Biao[1] CHEN Jin-hui[2] SUN Ya-min[1]
(School of Computer Science & Technology, Nanjing University of Science & Technology, Nanjing 210094, China)[1]
(School of Information & Control, Nanjing University of Information Science & Technology, Nanjing 210044, China)[2]

Abstract In most large scale Publish/Subscribe systems implemented on top of the internet, the overlay is not reliable. One of the major challenges of pub/sub systems is their reliability and their ability to cope with failures in the system. A formal specication of reliability condition in large scale publish/subscribe systems is described. The specication uses sequential traces and is based on the syntax of linear temporal logic, it also serves as the basis for proving the correctness of the routing algorithms. The Crash/Recover scheme has been designed to cope efficiently and locally with temporary router or link failures, that is, to maintain a consistent shared state in the system in spite of transient failures.
Keywords Reliability, Trace sequence, Crash/recover, Routing protocol

1 引言

Internet 技术的广泛应用和移动计算、网格计算以及普适计算平台的快速发展,要求分布式系统能够满足大规模、分散控制和动态改变的要求。新型面向大规模的发布/订阅系统,能够使得信息交互的双方在时间、空间和控制流三个方面都被完全解耦,是网络中内容路由技术的最佳载体。近年来,为提高发布/订阅系统的可靠性,在路由算法的容错能力、当故障发生时如何重配系统以使其恢复到正常状态等方面作了较多研究[1-3]。这些研究大都基于代理网络的无环图结构,但是关于容错能力的研究成果还不能令人满意。

IBM Gryphon[4,5]通过在每一个结点上维护一个并行查找树来高效地确定事件的路由,没有研究如何修改查找树,保证可靠传递。Siena[6,7]维护一个有关订阅集合的路由表,没有明确陈述系统的容错问题。Jedi[8,9]实现动态拓扑重构,目的在于减轻负载。Rebeca[10]是一个整合多个路由策略的原型,实现了基于租约的自稳定系统,要求有规律地刷新租约,阻碍了系统的大规模性。Onyx[11]使用 YFiler[12]作为内容路由,没有提出可靠性方案。与上述系统相比,本文的主要特点:

(1)基于状态轨迹和线性的时态逻辑,定义了订阅/发布系统的稳定性;

(2)给出基于崩溃/恢复模式的路由算法,并对算法的稳定性进行分析。

2 基本概念和定义

发布/订阅系统如图 1 所示。事件通知服务充当发布者和订阅者间的媒介,由一组互连的路由器组成。相邻的路由器之间通过接口(或称链路)相连,和发布者相连的路由器称为发布结点,和订阅者相连的路由器称为订阅结点,其它路由器称为内部路由结点。发布者从它的发布结点发布事件,订阅结点把通告传递给订阅者。订阅者调用 register()和 cancel()来注册和取消一个订阅,发布者调用 publish()创建一个事件后,由事件通知服务向所有订阅者传播事件。很多订阅/发布系统的带有通知消息,通过增加 Adv()、Unadv()两个操作,通知消息能被整合进订阅/发布系统,订阅结点或内部路由结点通过 Adv()向其上游路由器表明订阅意图,通过 Unadv()取消订阅。

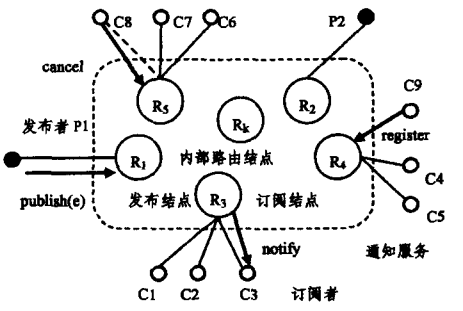

图 1 发布/订阅系统模型

*)本研究得到南京信息工程大学科研基金资助项目(Y640)资助。董 飚 博士生,主要研究方向为中间件技术;陈金辉 副教授,主要研究方向为系统分析与集成;孙亚民 教授,博士生导师,主要研究方向为网络工程。

文中 C 为订阅者的集合，E 为事件的集合，R 为所有路由器的集合。对于任意的 R_i，在 R_i 上存在唯一一个过滤器 F_i，F 是系统中所有过滤器的集合。

定义 1（映射 Filter） $(F,E)\rightarrow$Boolean，一个事件 E_i 匹配一个过滤器 F_j，当且仅当 Filter(F_j,e_i)=true，SatAll(F_j) 是 F_j 匹配的所有事件的集合。

定义 2 发布/订阅系统的系统状态是一个三元组 S(ActSub(C_i)，PubNoti(P_i)，ActAdv(C_i))，其中：

(1) ActSub(C_i)：C_i 的活动订阅结点集，即 C_i 已经注册了并且没有取消订阅条件的所有订阅结点集合；

(2) PubNoti(P_i)：P_i 已经发布的事件集合；

(3) ActAdv(C_i)：C_i 的活动通告结点集，即已经发出 C_i 的通知并且没有取消该通知的所有订阅结点集合。

定义 3 发布/订阅系统中的系统操作由表 1 中的操作组成，记为 OP。

表 1 系统操作

操作名	含义
Publish(P_i,E_j)	P_i 发布 E_j
Notify(C_i,E_j)	把 E_j 通知给 C_i
Register(C_i,R_j)	C_i 在 R_j 上注册
Cancel(C_i,R_j)	C_i 在 R_j 上注销
Adv(C_i,R_j) /Adv(R_i,R_j)	C_i/R_i 向 R_j 发通知消息
UnAdv(C_i,R_j)/UnAdv(R_i,R_j)	C_i/R_i 向 R_j 发取消通知消息

定义 4 轨迹序列是一个二元组(S_i,Op_i)组成的有序序列，记为 σ。轨迹序列描述了系统的特定行为。

定义 5 设 $\sigma=s_1,op_1,s_2,op_2,\cdots$是一个轨迹序列，对于 $\forall op_i\in OP$，在 σ 上关于 op_i 的谓词记为 OPP_i，有 $OPP_i(\sigma)=$true$\Leftrightarrow op_i=op_1$。

定义 6 由谓词 OPP、量词 $\exists\ \forall$、逻辑操作符 $\wedge\ \vee\ \sim\Rightarrow$、模态算子$\Box\Diamond\bigcirc$组成的合法表达式，称为发布/订阅系统中的公式，记为 ψ。

模态算子$\Box$解释为“永远、总是”，$\Diamond$解释为“有时、最终”，$\bigcirc$解释为“下一次”[13]。

定义 7 ψ是一公式，$\sigma=s_1,op_1,s_2,op_2,\cdots$，模态算子的语义：

(1) $\Diamond\psi=$true$\Leftrightarrow\exists\, i,\psi(s_i,op_i,s_{i+1},op_{i+1},\cdots)=$true；

(2) $\Box\psi=$true$\Leftrightarrow\forall\, i,\psi(s_i,op_i,s_{i+1},op_{i+1},\cdots)=$true；

(3) $\bigcirc\psi=$true$\Leftrightarrow(s_2,op_2,s_3,op_3,\cdots)=$true。

定义 8 订阅/发布系统的可靠性条件，记为 Ω_{as}：

$$\Diamond\Box[\text{Notify}(C_i,E_j)\Rightarrow[\bigcirc\Box\sim\text{Notify}(C_i,E_j)]\wedge[\exists P_k,E_j\in\text{PubNoti}(P_k)]]\wedge[\exists F_i\in\text{ActSub}(C_i),E_j\in\text{SatAll}(F_i)]]$$

在 Ω_{as} 中，系统故障被认为是一个短暂的，只能改变系统状态，而不改变系统行为。例如，路由器或链路的故障在短时间内能得到恢复，不需要任何外界干预。Ω_{as} 假设故障不连续发生，该条件保证从任何状态开始，系统最终将继续安全地运行。

定义 9 带通知的订阅/发布系统的可靠性条件，记为 Ω_{adv_as}：

$$\Diamond\Box[\text{Notify}(C_i,E_j)\Rightarrow[\bigcirc\Box\sim\text{Notify}(C_i,E_j)]\wedge[\exists P_k,E_j\in\text{PubNoti}(P_k)]]\wedge[\exists F_i\in\text{ActSub}(C_i),E_j\in\text{SatAll}(F_i)]]\wedge(\text{Publish}(P_k,E_j)\wedge[\forall F_i\in\text{ActAdv}(C_i),E_j\notin\text{SatAll}(F_i)]\Rightarrow[\Box\sim\text{Notify}(C_i,E_j)])]$$

3 崩溃/恢复模式的路由协议

崩溃/恢复模式被设计用来有效地处理局部性的、临时性的路由器或链路故障，它假设：路由器或链路的故障在短时间内能得到恢复。崩溃/恢复模式实现了可靠的订阅通知。在发生故障期间，发布者、订阅者仍然能发布、订阅事件，即故障是透明的，在故障被恢复后，系统达到一致的状态，好像没有发生故障一样。

设路由器 R 有 n 个下游接口 $I_{rdowni}(1\leqslant i\leqslant n)$，$D_i$ 是 R 的下游路由器，与 I_{rdowni} 相连；U 是 R 的上游路由器，I_{rup} 是 R 的上游接口。D_i 每一次发送一个通知和一个 R，D_i 之间唯一的序号到 R。设 hr_i 是 R 从 D_i 接受的通知消息的最高序号，即 R 已经从 D_i 接受了所有序号 $S_n\leqslant hr_i$ 的消息；$adv(S_j)$ 是 D_i 发送到 R 的通知消息 S_j。hs 表示 R 发送到 U 的通知消息的最高序号，$adv_{out}(S_j)$表示 R 发送到 U 的通知消息 S_j。

定义 10 R 上的恢复数据库 Rec 样式 $RS(routing_table,hr,hs,adv_{out})$是一个四元组，其中：

(2) $routing\_table$ 是路由表；

(2) $hr=\{hr_i\mid 1\leqslant i\leqslant n\}$；

(3) $adv_{out}=\{adv_{out}(S_j)\mid 0\leqslant j\leqslant k\}$，$k$ 为 R 已发送给 U，但未经 U 确认的通知消息数。

图 2 为路由器 R 的事件模型，R 处理 ADV，ACK，BACK 和 FROM_FAILURE 四种事件。对这四种事件的处理见算法 1-4。

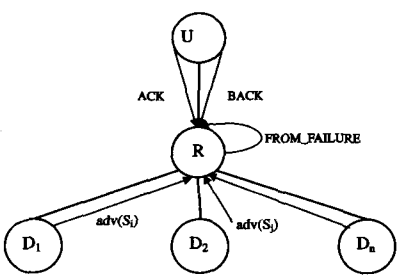

图 2 路由器结点的事件模型

算法 1 接收通知消息的处理程序

```
//R 接收从 Irdowni来的通知消息 adv(Sn)
OnReceiving(adv(Sn),Irdowni)
{if (0< Sn≤hri)
    //advout(Sn)是重复消息
    SendMsg(ACK,Sn,Irdowni);
  else if (Sn= hri+1)
    {hs++;
     //修改主存路由表
     Update(Main_routing_table);
     //生成通知消息 advout(hs)
     Assemble(advout(hs));
     hri++;
     //修改主存 advout表
     Append(Main_ advout,advout(hs));
     //修改 Rec,这是原子操作
     Backup(Rec. routing_table,Rec. advout);
     //向 Irdowni发对 adv(Sn)的确认
     SendMsg(ACK,Sn,Irdowni);
     //向 Iup发通知消息 advout(hs)
     SendMsg(Advout,hs,Iup);}
}
```

算法 2 接收确认消息的处理程序

```
//R 从 Irup接收序号为 Sn 的确认
OnReceiving(ACK,Sn,Irup)
{if advout(Sn)∈Main_ advout
    {Delete(Main_ advout(Sn));
    //原子操作,从 Rec 中删除 advout(Sn)
```

(下转第 148 页)

Wiley & Sons,1991

[2] Gruber T R. Towards Principles for the Design of Ontologies Used for Knowledge Sharing // Poli R, Guarino N, eds. International Workshop on Formal Ontology. Padova, Italy, 1993

[3] Kashyap V, Sheth A P. Semantic and schematic similarities between database objects: a context-based approach. VLDB Journal, 1996: 176-304

[4] Lehmann F, Cohn A G. The EGG/YOLK Reliability Hierarchy: Semantic Data Integration Using Sorts with Prototypes // CIKM'94. 1994: 272-279

[5] Levenshtein I V. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Cybernetics and Control Theory, 1996, 10(8): 707-710

[6] Noy N F, Musen M A. An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support // 16th National Conference on Artificial Intelligence (AAAI-99), Workshop on Ontology Management. Orlando, FL, 1999

[7] 刘惟一, 田雯. 数据模型. 科学出版社, 2001

[8] 史忠植. 高级人工智能(第二版). 科学出版社, 2006

[9] Weinstein P, Birmingham W. Comparing concepts in differentiated ontologies // Proc. of AW-99. 1999

(上接第 115 页)

```
    Backup(Rec. adv_out);}
}
```

算法 3 接收 BACK 消息的处理程序

```
//R 从 I_rup 接收 BACK 消息
OnReceiving(BACK, I_rup)
{//向 I_up 发送 Rec. adv_out
  SendMsg(Advout, Adv_out, I_up);
}
```

算法 4 接收故障恢复消息的处理程序

```
OnReceiving(FROM_FAILURE)
{//修改主存路由表
 Restore(Main_routing_table);
 //修改主存 adv_out 表
 Restore (Main_ adv_out);
 //向 I_up 发送 Rec. adv_out
 SendMsg(Advout, Adv_out, I_up);
 //R 向所有 I_rdowni 发 BACK 消息
 For(i=1,n,i++) SendMsg(BACK, I_rdowni);
}
```

定理 1 基于崩溃/恢复模式的路由协议，满足可靠性条件 Ω_{adv_as}。证明从略。

崩溃/恢复模式依靠这几个机制来处理暂时故障：恢复数据库是稳定的，当路由器发生故障后，它能恢复到故障前的状态；使用 TCP 协议确保文档和订阅能可靠、有序地传输；Rec. adv_{out} 维护一组路由器及其上游路由器间的确认信息，在上游路由器发生故障期间，路由器上保存改变量，当上游路由器恢复后，该路由器能前滚到和当前订阅一致的状态；序列号被嵌入在所有的消息中。

结束语 本文系统地、形式化地研究了订阅/发布系统中的可靠性问题。基于轨迹序列和线性时态逻辑定义了订阅/发布系统和带通知的订阅/发布系统的可靠性条件。设计了崩溃/恢复模式的路由协议，该路由协议满足带通知的订阅/发布系统的可靠性条件要求，能有效地处理具有局部性、临时性的路由器故障和链路故障。本文对订阅/发布系统中可靠性条件的研究成果可用于 Internet 上大规模发布/订阅系统路由算法的可靠性分析。

参 考 文 献

[1] Costa P, Migliavacca M, Picco GP, et al. Epidemic algorithms for reliable content-based publish-subscribe: An evaluation // Proc. of the ICDCS 2004. Tokyo: IEEE Computer Society, 2004: 552-561

[2] Bhola S. Topology changes in a reliable publish/subscribe system. Technical Report, RC23354. Yorktown Heights: IBM Thomas J. Watson Research Center, 2004

[3] Chand R, Felber PA. A scalable protocol for content-based routing in overlay networks // Proc. of the 2nd IEEE Int'l Symp. on Network Computing and Applications. Cambridge: IEEE CS Press, 2003: 123-130

[4] Bhola S, Strom R E, Bagchi S, et al. Auerbach. Exactly-Once delivery in a content-based publish-subscribe system // Lala J, ed. Proc. of the Int'l Conf. on Dependable Systems and Networks (DSN 2002). Washington: IEEE Computer Society Press, 2002: 7-16

[5] Opyrchal L, Prakash A. Secure distribution of events in content-based publish subscribe systems // 10th USENIX Security Symposium. August 2001

[6] Carzaniga A, Wolf A L. Forwarding in a content-based network // Proceedings of ACM SIGCOMM 2003. Karlsruhe, Germany, 2003: 163-174

[7] Carzaniga A, Rosenblum D S, Wolf A L. Content-based addressing and routing: A general model and its application. Technical Report CU-CS-902-00. Department of Computer Science, University of Colorado, January 2000

[8] Cugola G, Nitto E D, Fugetta A. The JEDI event-based infrastructure and its application to the development of the opss wfms. IEEE Transactions on Software Engineering, 2001: 27 (9): 827-850

[9] Bricconi G, Tracanella E, Nitto E D, et al. Analyzing the behavior of event dispatching systems through simulation // HiPC '00: Proceedings of the 7th International Conference on High Performance Computing. London, UK, Springer Verlag, 2000: 131-140

[10] Mühl G. Large-Scale Content-Based Publish/ Subscribe Systems. PhD thesis. Darmstadt University of Technology, 2002

[11] Diao Y, Rizvi S, Franklin M J. Towards an Internet-Scale XML Dissemination Service // Proceedings of VLDB 2004. Toronto, Canada, August 2004

[12] Diao Y, Franklin M J. High-Performance XML Filtering: An Overview of YFilter. IEEE Data Engineering Bulletin, March 2003

[13] Pnueli A. The temporal semantics of concurrent programs. Theoretical Computer Science, 1981(13): 45-60