分布式数据库管理系统POREL的进程通信系统CS

徐泽同

(中国科学院数学研究所)

THE COMMUNICATION SYSTEM CS OF THE DISTRIBUTED DATA BASE MANAGEMENT SYSTEM POREL

Xu Zetong

(*Institute of Mathematics, Academia Sinica*)

ABSTRACT

This paper gives an introduction to the communication system CS of the distributed data base management system POREL, it was implemented on the VAX/750 computer and the operating system UNIX in the application information institute of the vienna technical university in January, 1987 by writer.

摘 要

本文介绍了作者于1987年元月在维也纳技术大学应用信息研究所，用VAX/750机及UNIX操作系统实现的分布式数据库管理系统POREL的进程通讯系统CS。

§1. POREL简介

POREL是西德计算机科学家E.J.Neuhold领导下研制的分布式关系数据库管理系统。在1984年前的斯图加特时期，他们在PDP机RSX11M操作系统上作出了一个不可运行的原型系统。

1989年11月19日收到，1990年3月17日定稿。

1985 年元月因执行中国科学院科学数据库与维也纳技术大学的合同，我等五名访问学者去维也纳技术大学应用信息研究所Neuhold 工作处，用VAX/750 机UNIX 操作系统重实现POREL。经10 人年的努力，完成了对POREL 的用户服务，离网分析、涉网分析、关系基本机器的修改，完成了通信系统、事务管理的重写，使POREL 在模拟的X.25 网上得以联合运行，通过了一批分布式数据库定义、查询、修改及分布计算的试例，使POREL 达到了八万余行pascal+c 程序的规模，将POREL 作成了一个可运行的原型系统。

通信系统是POREL 的子系统，它负责POREL 的局部结点和远程结点上的各进程之间的信息和数据的传输。

§2. CS 的设计思想

POREL 的分布进程间的同步和通信是利用中间通信进程CS 来实现的。CS 要在局部进程和远程进程间可靠地传输任意长度的数据，并利用数据转换机制来保持所交换数据的语义。数据传输基于X.25 网，之所以选此是为了使CS 易于与现存的网络接口，从而使POREL 易于适应多种网络。CS 的体系结构遵从ISO 的OSI 结构，即公认的应用、表示、会话、传输、网络、链路及物理七层通信协议的开系统互连结构。

CS 的用户是POREL 的其它各分布进程，这些进程用CS 来实现分布式数据库管理系统的委托、并发、恢复、死锁等协议，我们没有限定上述协议的特性，因此我们必须使CS 有能力来克服应用层传来的各种错误。

为了使CS 的使用简单、清晰，各CS 服务必须由使用者启动，这样CS 不会中断别的进程，从而摆脱了难于分析及难于消除的不确定性。于是就引出了对称通信结构，在此结构中应用实体间的通信不能在两对应的CS 调用之前发生。

POREL 主要使用进程对进程通信，因而CS 仅支持进程对进程通信。当应用需要广播等功能时，应用者应把它映射为进程对进程的通信。

POREL 内用结点号NODE，进程名PNAME(或者进程类型PTN 与事务号TNO 的组合)，连接号CNO 三者来唯一标识一个应用实体，于是对偶(NODE1, PNAME1, CNO1; NODE2, PNAME2, CNO2) 唯一标识了POREL 内两个应用进程之间的一个连接。

§3. CS 的服务原语

CS 提供了一批简洁的使用原语，其形如:

XYZ, $\rightarrow P_1, \leftarrow P_2, \leftrightarrow P_3$

其中XYZ 是调用的操作码，P_i 是参数，$\rightarrow$ 表示输入参数，$\leftarrow$ 表示输出参数，$\leftrightarrow$ 表示输入输出参数。应用CS 的进程在使用CS 前必须调用CSBEGIN 原语来进入CS，通信结束之后必须调用CSEND 原语来退出CS。CSBEGIN 和CSEND 之间的部分才是CS 管理的部分；应用进程调用CONNECT 原语来建立希望与之通信的进程之间的连接。这要提供自身的进程标识，也要提供对方的进程标识，同时要指定一连接号，当两个彼此希望通信的进程都向对方发出连接要求，并且CS 接受这两个连接要求之后，这两进程之间才能建立连接号为CNO 的一连接。由调用DISCONNECT 原语来撤销。当连接着的两个进程相

互发出撤销连接之后，这一连接才会被撤销；调用TRANSMIT 原语可在CNO 指明的连接上发送电文，调用AWAIT 原语可在CNO 指明的连接上接收电文，调用AWAITL 原语可在CNOSET 指明的连接集合中任选地接收电文；调用TRANSMITF 原语可在CNO 指明的连接上发送文件，调用AWAITF 原语，可在CNO 指明的连接上接收文件；调用INFORMC 原语可查询连接状态，调用INFORMN 原语可查询结点状态。

§4. CS 的实现要点

4.1 现未实现表示层

因我们的工作现在还限于使用均质网，数据在传输过程中不需要改变其表示。但将来走向非均质网时则要补上表示转换功能。

4.2 会话层(*l5*) 实现了流量控制

CS 实现时将一些会话层的功能放进了CS 的接口CS1 中，其中包括流量控制功能。之所以需要流量控制，是为了防止CS 的用户在发送和接收不协调时将CS 堵塞。流量控制的原理如图1, 其含义是：发送进程(如PS) 调用CS 发送原语发送信息时，PS 的CS1 将要发送的信息登记在连接号CNO 指明的发送缓冲区中，再把发送缓冲区的控制信息记录在CNO 指明的发送队列中，进而传向本方CS 处理；目标CS 收到信息后，要在相应CNO 指明的接收缓冲区登记信息，同时把接收缓冲区的控制信息登记在CNO 指明的接收队列中，这样才完成了信息的发送；当接收进程(如PR) 调用CS 接收原语接收信息时，其CS1 查相应CNO 指明的接收队列，如查到则据其控制信息从接收缓冲区中取走信息，之后清除接收队列中的该项接收控制信息，同时向发送信息方送去“信息已被接收”的回答；当发送信息进程的CS1 收到对方送回的“信息已被接收”的回答后才会清除发送队列中的相应发送控制信息。因发送和接收队列有限，当发送队列满而又出现发送要求时，如此发送要求无等待条件，则失败返回；如有等待条件，则要等到接收方取走一信息而发送方在发送队列有空间时才继续发送。在局部进程通讯情况下，本方CS 与目标CS 合二为一，因此信息的存贮地址不变，只是在发送队列与接收队列间交换了控制信息，这样简化操作，因而提高了速度。

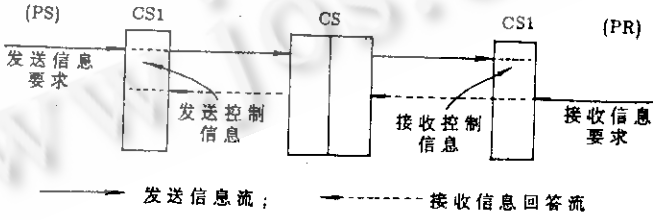


图1 流量控制示意图

4.3 传输层(*l4*) 实现了错误控制，建立连接，传输数据，撤销连接等协议

实现的CS 定义了下述传输服务原语：TCONRQ 建立连接要求；TCLRRQ 撤销连接要求；TSDURQ 传送电文要求；TFILRQ 传送文件要求；TSDUAC 接收电文回答；TFI-

LAC 接收文件回答。

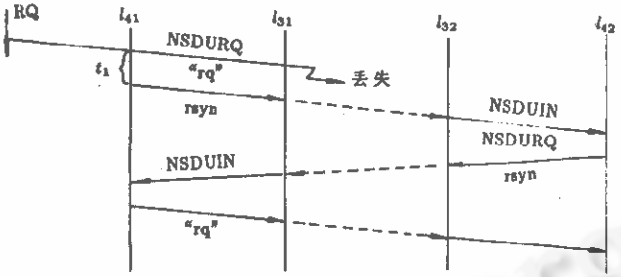
从传输层向会话层报告的服务原语：TCONAC1 初步接受建立连接；TCONAC2 接受建立连接；TCONRJ 拒绝建立连接；TCLRIN 远端撤销连接指示；TCLRAC 接受撤销连接；TSDUIN 收到电文指示；TFILIN 收到文件指示；TSDUAC 接收远端电文；TFILAC 接收远端文件。

4.3.1 错误控制

为了正确地发送协议数据单元(pdu)，我们采用了发送—确认机制。在发送单个信息时，发送确认为一对一；为了节省确认个数，可用多发送一单确认机制，即连着发送n 个pdu 后只要第n 个pdu 被确认，则认为n 个pdu 全部得确认，确认信息还可作为pdu 的寄生物来传送。

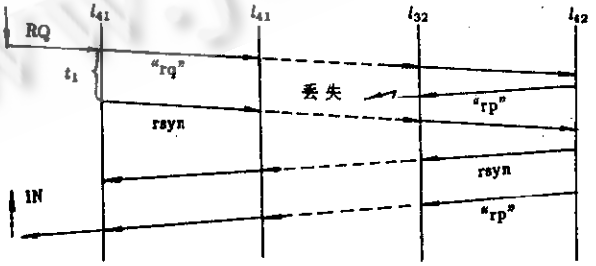
将所有pdu 顺序编号。当它被发送后如未收到确认则认为发送出错。于是发送方用同步信息问接收方“正确收到了哪号pdu?”，接收方用同步信息回答它所需的pdu 号。当发送的pdu 丢失，发送方重发它；当确认丢失，接收方在同步回答之后即送出确认。

因实现的网络层并不向传输层报告所传数据的丢失，于是发送方设置了时钟t_1，在正常情况下发送后t_1 时限内会收到确认；同理在接收方设置了时钟t_2，在送出确认之后，如其丢失，t_2 时限内会收到发送方来的同步信息。图2、3 分别指出了电文丢失和确认丢失情况下的纠错流程，在上述过程中所有pdu 存贮在与连接有关的队列中，但同步pdu 则插入队列之首位以保证首先处理错误后的再同步工作。



$l4i$、$l3i$ 为层4、层3 的i 号实体；↓RQ “要求”服务原语；“rq” “要求” pdu；rsyn 同步pdu；t_1 发送时限；→ 原语流程；···→网络传输；NSDURQ, NSDUIN 模拟的X.25 传输要求及传输指示。

图2　电文丢失纠错流程



“rp” 确认pdu；↑ IN “指示”服务原语

图3　确认丢失纠错流程

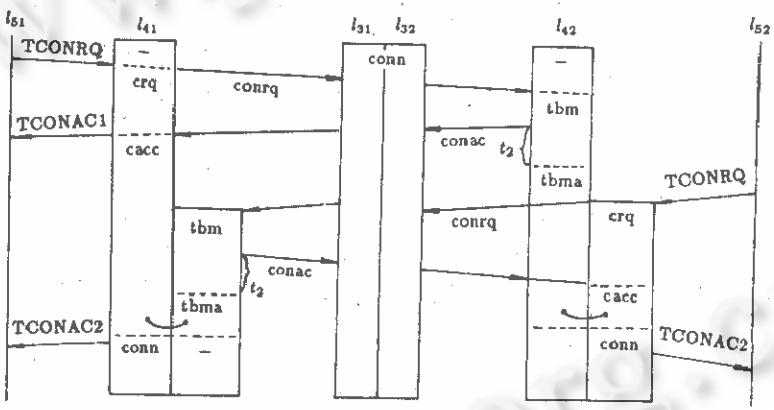
4.3.2 建立连接

传输层从一本地进程的CS1 收到建立连接要求(TCONRQ) 时，于是建立一个新的连接，其状态记为crq(connect-request), 并用NSDURQ 服务原语发送conrq-pdu。其后若收到接收方用conac-pdu 表示的接受时，则将连接状态改为cacc(connect-accept), 且以TCONAC1 服务原语将控制交到CS1; 若接收方以conrj-pdu 表示拒绝时，则撤销此连接，并以服务原语TCONRJ 将控制交到CS1。

在接收方收到远端建立连接要求conrq-pdu 时，则建立状态tbm(to-be-matched) 的新连接。当送回conac-pdu, 且渡过时限t_2 之后连接状态改为tbma(to-be-matched-accept); 若送回conrj-pdu 则撤销tbm 连接。

若在一传输实体中，存在状态tbma 和cacc 的两个连接，且它们地址一致，于是将cacc 的连接改为conn(connected), 撤销tbma 连接。

图4 展示了两个应用实体相继接受对方的连接要求而建立成连接的流程。图5 展示了两个应用实体重迭向对方发出建立连接要求而拒绝一个的流程。传输层同时也处理两个局部实体的连接要求，图6 展示了建立局部连接的情况，从图4 和图6 可以看出，远程连接和局部连接的建立流程有很多相似之处。



⌣ 表示两个连接有一致的地址

图4　两个连接实体间正常建立连接的流程

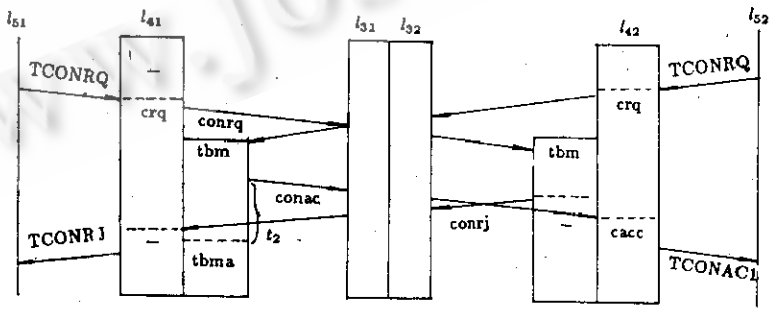


图5　拒绝重迭的连接要求流程

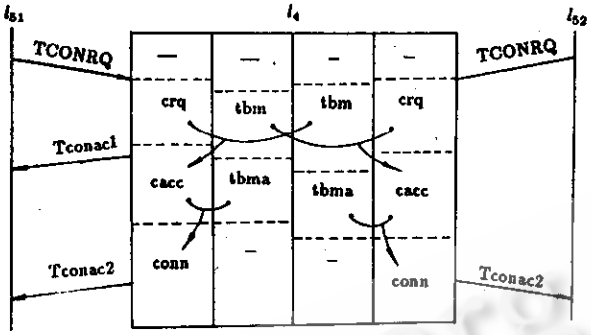


图6　正常的局部连接建立流程

4.3.3 数据传输

对话层以服务原语TSDURQ 要求传输电文；以TFILRQ 要求传输文件。传输层将电文放在一个data-pdu 中；将文件的每块放在一个data-pdu 中，且用fbegin-pdu 和fend-pdu 将它们括起来。远端传输实体收到电文时以服务原语TSDUIN 通知对话层；收到文件的各块后，即收到fend-pdu 之后用TFILIN 通知对话层。每收到一个pdu 则用ack-pdu 去证实。传输层暂存收到的电文或文件，以备应用实体的接收。当应用实体接收电文或文件时，则用服务原语TSDUAC 或TFILAC 通知传输层，而传输层则用acack-pdu 向电文或文件的发送者报告。

图7、8、9、10 描述了电文、文件的远程传输与接收。rf 为远程文件传输状态。

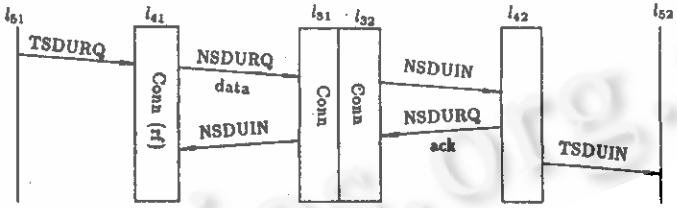


图7　电文的传输

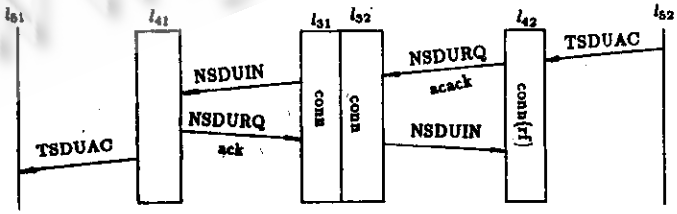


图8　电文的接收

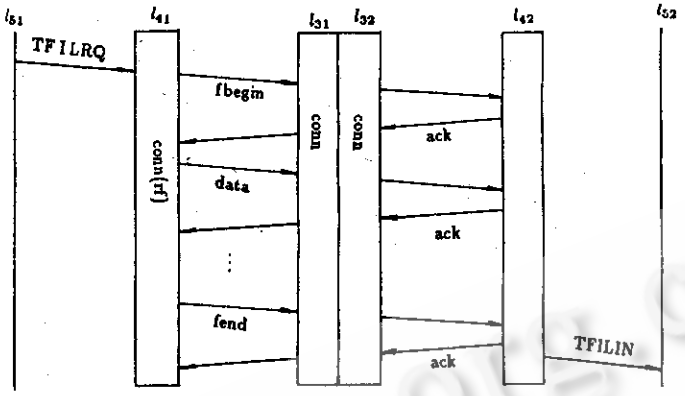


图9　文件的传输

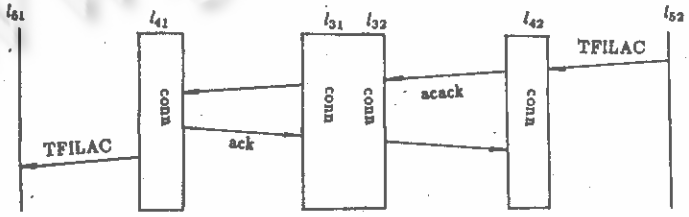


图10　文件的接收

图11, 12, 13, 14 展示了局部电文和文件的传输与接收。

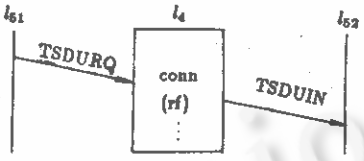


图11　局部电文传输

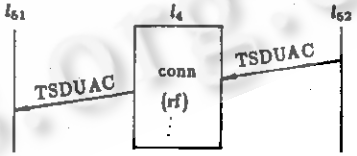


图12　局部电文接收

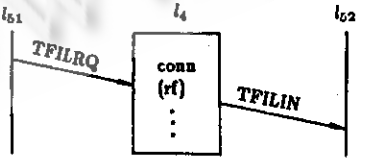


图13　局部文件传输

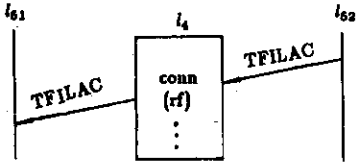


图14　局部文件接收

4.3.4 撤销连接

传输实体收到服务原语TCLRRQ, 则将连接状态从conn 改为drq(disconnect-request);

从rdac(remote-disconnect-accept) 改为drqac(disconnect-request-accept); 从rdrg(remote disconnect-request) 改为drqrdrq(disconnect-request-remote-disconnect-request), 并发送clrrq-pdu 到远端传输实体。一旦收到clrac-pdu, 则把连接状态从drq 改为dacc(disconnect-accept); 从drqac 改为一, 且撤销连接, 并送TCLRAC 到对话层。

远端来的撤销连接要求只能接受。因此收到这要求一方面用TCLRIN 向对话层报告, 一方面送回clrac-pdu 以表接受, 连接状态则从conn 改为rdrq; 或从dacc 改为drqac。

图15、16 展示了远程实体间撤销连接的两种情况。图17 展示了局部连接撤销的情况。

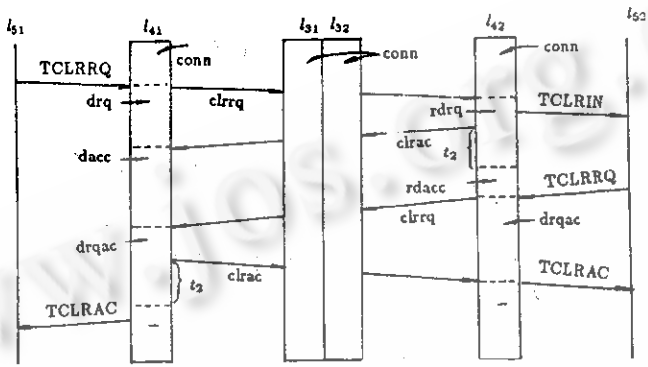


图15　正常的两个连接实体间撤销连接流程

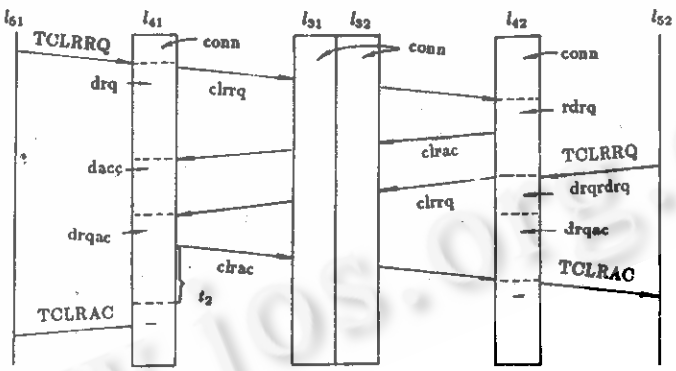


图16　重迭的两个连接实体撤销连接流程

4.4 X.25 网的模拟

在维也纳工作时, 因无可用的网络软件, 故在VAX/750 上模拟了X.25 网, 并在此上实现了CS, CS 有1 万1 千余行pascal+c 程序。因模拟的X.25 网与现在要进行的实用化研究相关不大, 故不再述。

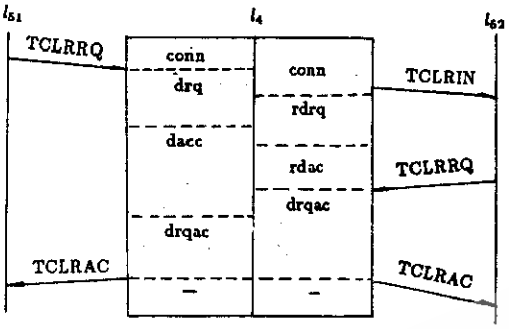


图17 撤销局部连接流程

参考文献

[1] ISO/TC97/SC16 Data Processing-Open Systems Interconnection-Basic Reference Model (December, 80).
[2] CCITT Recommadation X.25 (June, 79).
[3] POREL Design Specification, Technical Report (78.5) University of Stuttgart.
[4] E.J.Neuhold, Distributed Data Base Systems with Special Emphasis toward POREL (84).
[5] K.böhme, Prozesskommunikation in heterogenen rechnernetzen dargestellt am Beispiel des Verteilten Datenbank Systems POREL (82).
[6] Ada 导引/程序设计语言Ada 参考手册, 袁崇义、徐泽同译, 科学出版社(86.1)。

(上接第10 页)

因此由$C = C_1' + C_2'$ 知:

$$C = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -1 & 1 & -1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{array}$$

显然满足定理1 的条件(1) (2) (3) 的X_1 有两个: $X_1 = (1,1,0)$, $X_1' = (1,0,1)$, 因此由定理2 和定理3 知: 全局存在两个死锁$\{A,B\}$, $\{A,C\}$。这和实际情况相一致, 而由定理4 及定理5 知: 撤消进程A 可消除全部死锁, 从而保证系统正常工作。

这个例子进一步说明文中的方法是正确的。

感谢西德GMD 的K. Voss 博士提供了宝贵的资料, 感谢中科院数学所陆维明教授的帮助。

参考文献

[1] 吴启明, 事务管理系统的基础模型BMT, 中科院数学所硕士论文, 1988。
[2] 何炎详, 分布式OS 处理死锁的两种方法, 计算机工程与应用, 1988, NO. 7。
[3] J. L. Peterson, 《Operating System》, 1985.
[4] D. L. P et al., Comment on Deadlock Prevention Method, Communication of ACM, Vol. 15, No. 7, 1972.
[5] D. Stuart et al., A Gentralised Deadlock Detection Algorithm, ACM Trans on DB, VOl. 7, No. 2.
[6] W. Resig, Introduction to Petri Nets, 1985.
[7] T. Murata, State Equation, IEEE Trans, 1977.
[8] Yuan ChongYi, 《Synchonic Structure》, LNCS 222.
[9] K. Voss, 《Nets in DB》, LNCS 254.
[10] K. Voss, 《Nets in Office Autonation》, LNCS 254.
[11] 《高等代数》, 北京大学力学系。
[12] 张远达, 《线性代数》, 1983。