异构网络协议转换器构造技术研究*)

赵新有　古天龙　董荣胜　李凤英
(桂林电子工业学院计算机系　桂林541004)

Research of Protocol Converter Between Heterogeneous Networks

ZHAO Xin-You　GU Tian-Long　DONG Rong-Sheng　LI Feng-Ying
(Department of Computer, Guilin University of Electronic Technology, Guilin 541004)
E-mail: longwind@gliet.edu.cn

Abstract　A protocol converter can solve the problem of communication between heterogeneous and distributed computer networks. This paper proposes a novel approach to construct a converter for these protocols. This approach includes following steps: getting reduced protocol model by merging states and shielding insignificant message; constructing protocol converter by reduced protocols; dividing merged states and getting a protocol converter. This approach can improve computational complexity.

Keywords　Communication finite state machine, Protocol conversion, Heterogeneous networks

1. 引言

异构网络互联的一个难点就是协议的不匹配，它导致不同网络体系下的用户难以直接相互通信。为了解决此困难，在协议间插入转换器当作通信中介，与协议实体一起形成网络系统，转换器依事先定义好的次序分发报文到另外一个协议实体，实现协议的转换。Okumura[1]提出了采用单个通信有限状态机(CFSM)作为一个转换种子，用它来描述两个协议间的交互操作关系，该方法能自动构造一个转换器，但转换种子是依设计者给出，设计者应用不同的转换种子可能得出不同的转换器，在转换器不能构造时，不知是协议本身问题，还是由于转换种子的问题。尽管它有这些缺点，但与其它协议转换算法相比，它仍然是很有潜力的、高效的协议转换算法，在自底向上转换算法中具有重要作用[2,3,6]。然而，与其它采用通信有限状态机(CFSM)模型描述一样，计算复杂度[4,5]仍是人们关注的问题之一。为此，本文对此问题作了进一步的研究，提出了一种新的协议转换构造方法，此方法可改善构造过程中的计算复杂度。

2. 协议的形式描述

形式描述是协议设计的一个关键环节。在目前已开发的形式描述技术(FDT)中，通信有限状态机(CFSM)[1,2,6]由于具有简单性、直观性和高度抽象性，且易于自动实现，被广泛应用于协议的形式描述、验证中[1~3]。

定义1　通信有限状态机(CFSM) A 是一个四元组，$A=(S_A,\Sigma_A,s_{0A},\delta_A)$，在此 S_A 是状态集；Σ_A 是报文集，包含发送报文与接收报文，即 $\Sigma_A=\{+e$ 或 $-e \mid e\in\Sigma_A\}$，式中 $+e(-e)$ 表示接收(发送)消息 e(在此+/−仅仅为了分析的方便，不会增加报文的数量)；$s_0\in S_A$，是 CFSM 的初始状态；δ_A 是部分转移函数，即 $\delta_A: S_A\times\Sigma_A\to S_A$ 的一个子集，在状态 $s\in S_A$ 对应的一个消息 $e\in\Sigma_A$，记为 $\delta_A(s,e)$，它表示 CFSM A 在状态 $s\in S_A$ 处执行消息 $e\in\Sigma_A$ 后所达的一个状态。

定义2　简单 P 协议是一个二元组，$P=[P_0,P_1]$，其中二者分别表示协议的发送实体与接收实体，它们都采用 CFSM $(P_i=(S_{Pi},\Sigma_{P_i},s_{0P_i},\delta_{P_i}),i=0,1)$ 模型来描述。

在此模型中，通信实体 P_0 与 P_1 通过全双工通信通道 Channel(采用 FIFO 机制)进行通信。发送实体 P_0 发送报文置于信道的尾部，而接收实体 P_1 从信道的首部取出报文并分发此报文，然后从信道中删除它，进而实现报文的转移。由此可知信道 Channel 用来保存那些已经发出，但仍然没有到达接收地的那些报文包[3]。

3. 转换器的构造过程

本文给出一种方法构造协议转换器，首先，采用状态归并算法简化协议的 CFSM 模型；然后采用 Okumura 算法构造转换器；最后采用状态拆分算法取得一个完整的转换器。假设两个目标协议 $P=[P_0,P_1]$ 与 $Q=[Q_0,Q_1]$，它们功能相似，但在细节上不同。由于协议不匹配，实体 P_0 与 Q_1 无法直接进行通信。为使实体 P_0 与 Q_1 相互通信，需在二者之间插入转换器 C 形成新的协议系统 $D=(P_0,C,Q_1)$，在此 C 充当一个解释器，它从一个协议的实体接收消息，并解释它们，然后把这些解释过的报文分发到另外的协议实体，达到相互通信的目的。此时指向初始协议实体 P_1 与 Q_0 的消息将直接指向协议系统 D 的实体 C，所以采用协议实体 P_1 与 Q_0 来构造转换器，在此采用转换种子 CFSM $M=(S_M,\Sigma_M,s_{0M},\delta_M)$ 描述实体 P_1 与 Q_0 部分消息之间的转移关系，其中 $\Sigma_M\subset\Sigma_{P_1}\cup\Sigma_{Q_0}$[1,2,6]。

在引入算法之前，为描述方便，下面给出一些约定。如果转换协议实体中消息在转换种子 M 中出现，则称此消息为有意义的消息，若此消息未在转换种子 M 中出现，称为是无意义的消息。若在 CFSM 中有 $\delta_A(s,e)=s_i(s,s_i\in S_A,e\in\Sigma_A)$，则

*)本课题得到国防预研基金项目及广西自然科学基金项目的资助。赵新有　硕士生，研究方向为网络协议、形式化技术。古天龙　教授，博士生导师，研究方向主要为形式化技术，符号模型检验，协议工程。董荣胜　副教授，研究方向主要为形式化技术，计算机科学与技术方法论。李凤英　助教，研究方向主要为协议工程。

s_i 是 s 的后继状态，所有 s 的后继状态表示为 $succ(s)=\{s_i|\delta_A(s,e)=s_1;s,s_i\in S_A,e\in\Sigma_A\}$；若有 $\delta_A(s_i,e)=s(s,s_i\in S_A,e\in\Sigma_A)$，则 s_i 是 s 的前趋状态，所有 s 的前趋状态表示为 $pred(s)=\{s_i|\delta_A(s_i,e)=s;s,s_i\in S_A,e\in\Sigma_A\}$。

算法1 状态归并算法

利用此算法可以简化协议的 CFSM 模型。屏蔽掉协议中相对于转换种子 M 无意义的消息，然后合成连接无意义消息的状态为一个状态，达到简化的目的。算法描述如下：

Step1 输入协议实体 F 的 CFSM 模型及转换种子 M 的 CFSM 模型

Step2 初始化 F' 为 F

Step3 对 F 中的一对状态 s_1 与 s_2（二者可以是合成状态也可以是简单状态），若二者之间的所有转移都是无意义的消息（相对转换种子 M），则进行以下的操作取得协议实体简化 CFSM F'

A）合并 s_1 与 s_2 为一个合成状态 (s_1,s_2)，并且删除 F' 中与二者关联的所有转移

B）对每一个 $s\in[succ(s_1)-s_2]$（或者 $[succ(s_2)-s_1]$），要在 F' 增加一处从 (s_1,s_2) 到 s 的转移，此转移用 F 中从 s_1（或 s_2）到 s 的转移来标识

C）对每一个 $s\in[pred(s_1)-s_2]$（或者 $[pred(s_2)-s_1]$），要在 F' 增加一处从 s 到 (s_1,s_2) 的转移，此转移用 F 中从 s_1（或 s_2）到 s 的转移来标识

step4 若 $F\neq F'$，置 F 为 F'，并转到 step3 继续执行；否则结束算法，输出 CFSMF'。

输出的 CFSM F' 即为最初输入 CFSM F 的简化模型。由于 F' 是对 F 中的状态进行归并所产生，每运行一次状态机中的状态数量将减少1，所以最终 F' 中的状态比初始 CFSM F 中的少，即 $|F|>=|F'|$（在此 $|F|$ 表示 CFSM F 中的状态数量）。

利用以上算法取得协议实体的简化 CFSM 模型后，用 Okumura 算法进行转换器（在此用 C' 表示）的构造。由于 Okumura 算法在构造转换器之前未对协议实体进行处理，所以她的方法计算复杂程度为 $O(|P_1|*|Q_0|*|M|)$（在此 P_1、Q_0 分别表示协议实体、M 表示转换种子）[2,6]；在本文中，由于构造之前对协议实体的模型进行了简化，计算复杂性问题得以改善，此方法的计算复杂度为 $O(|P'_1|*|Q'_0|*|M|)$（其中 P'_1，Q'_0 表示对实体 P_1、Q_0 归并后的实体）。

算法2 状态拆分算法

拆分算法是把简化过程中的简化状态拆分出来，于是被屏蔽掉的无意义消息也得到恢复。通过此算法，转换器 C' 扩展为一个完整的转换器 C。被合成的状态恢复到最初的状态，此状态称为合成状态的子状态，合成状态又称子状态的父状态。算法描述如下：

step1 对 C' 中的合成状态 $(x,s_1,s_2,\cdots,s_n,t_1,t_2,\cdots,t_m)$，在此 $x\in S_M,\forall\ i,s_i\in S_{P_1},\forall\ j,s_j\in S_{Q_0}$，把此状态分解成 mn 个简单状态，即是 $(x,s_1,t_1),(x,s_2,t_2),\cdots,(x,s_j,t_i),\cdots,(x,s_n,t_m)$。

step2 （在同一个父状态分解出的状态之间创建转移）

如果在 Q_0 中存在从 t_i 到 t_j 的转移 e，则对所有的 s_k，在 C 中用 e 连接即在 C 转移函数中加入 $\delta_C((x,s_k,t_i),e)=(x,s_k,t_j)$

如果在 P_1 中存在从 s_i 到 s_j 的转移 e，则对所有的 t_k，在 C 中用 e 连接即在 C 转移函数中加入 $\delta_c((x,s_i,t_k),e)=(x,s_j,t_k)$

step3 （在不同父状态分解出的子状态之间创建转移）

如果在 C 中有 $\delta_C((x,s_i,s_j,t_i,t_j),e)=(x',s_i,s'_j,t_i,t'_j)$，并且在 P_1 中存在 $\delta_{P_1}(s_k,e)=s'_k$，则在中增加转移函数 $\delta_C((x,s_k,t_i),e)=(x',s'_k,t_i)$，否则若在 Q_0 中存在 $\delta_{t_0}(t_1,e)=t'_i$，则在 C 中增加转移函数 $\delta_C((x,s_k,t_i),e)=(x',s_k,t'_i)$

4. 应用

为了说明本文方法的有效性，在此我们采用 Poll-End 协议（表示为 $P=(P_0,P_1)$）与 *Ack-Nack* 协议（表示为 $Q=(Q_0,Q_1)$）作为例子进行分析（如图1所示）。

Poll-End 协议的接收实体首先发送 *Poll* 命令请求创建连接，连接建立后，协议发送实体开始发送数据 *Data*，数据发送完毕后由协议发送实体发出 *End* 消息结束数据的传输；Ack-Nack 协议发送实体直接发送数据包 *Msg*，当收到 *Ack* 消息后，继续发送下一个数据包，当收到 *Nack* 消息，将重发原数据包，在接收端，当收到 *Msg* 时，以 *Ack* 消息进行响应，若收到一个错误的数据包 *Msg'* 时，它将返回一个出错响应消息 *Nack*。尽管二者都提供数据传输服务，但通信机制却不同，若要使 Poll-End 协议接收实体 P_0 与 Ack-Nack 的发送实体 Q_1 能相互通信，此时出现协议的不匹配，需要进行协议转换。下面给出转换器的构造过程：

转换种子 M 的构造。由于对 *Ack* 报文处理与否，协议实体 P_1 与 Q_0 之间将会产生不同的转换种子，在此用与 Okumura 相同的种子 M（如图3中的(a)中所示），此 CFSM M 说明，若通道的容量是无限时，此时根本不需要管理 *Ack* 的发送顺序，所以在此根本不需要处理 *Ack* 消息。

用归并算法取得协议实体 P_1 与 Q_0 简化 CFSM 模型。对图3(a)的转换种子 M，$\Sigma_M=\{+Msg,-Data\}$，在 P_1 中，状态0与1是用 $+Poll$ 与 $-End$ 来连接的，消息 $+Poll$ 与 $-End$ 相对于种子 M 是无意义的，因此状态0与1在 P'_1 可以合并为一个状态(0,1)，由于 P'_1 不可能再被简化，所以此时 P'_1 是一个简化的 CFSM。相似地对 Q_0，图2中的 Q_0 是 Q_0 的简化 CFSM。在简化的 CFSM 中，包含有初始状态的合成状态将成为简化协议的初始状态。在此(0,1)、(0,2)分别是 P'_1、Q'_0 的初始状态，在此用一个箭头来标识。

应用 Okumura 的算法构造出合成转换器。利用 P'_1 与 Q'_0，我们可以通过 Okumura 的第一步得到 $U=P'_1//Q'_0$；然后利用 U 与 M 构造出 $T=M*U$；经过 Okumura 中的第三步，删除出错的状态及它们之间的转移，构造出合成协议转换器 C'。三个 CFSM 分别如图3中的(b)，(c)，(d)所示。由于在图3中的转换器 C' 的三个状态全是合成状态，所以要进行分解。

通过对转换器 C' 合成状态的分解，得到转换器 C。C' 中的状态 $(a,0,1,0',2')$ 由 $P1$ 中的两个状态0、1和 Q_0 中的状态0'、2'组成，它被分为四个孩子结点：$(a,1,2')$，$(a,0,2')$，$(a,1,0')$，其中 $(a,0,0')$ 是 C 的初始状态。相似地 $(a,0,1,1')$ 分解成两个孩子状态 $(a,1,1')$ 与 $(a,0,1')$，$(b,0,1,1')$ 分解成两个孩子状态 $(b,1,1')$ 与 $(b,0,1')$。利用算法加入转移边：对同一个合成结点分解出的结点，直接利用最初协议的状态的转移进行加入。例如，对状态 $(a,1,2')$，$(a,0,2')$，我们将依转移函数 $\delta_{P_1}(0,+Poll)=1$ 执行，所以在 C 中将加入转移函数 $\delta_C((a,0,2'),+Poll)=(a,1,2')$；对不同合成结点分解出的结点，不可能把每一个合成结点的所有状态相互连接。例如，对合成状

态$(a,0,1,1')$与$(b,0,1,1')$在C'中用一个-Data连接，但不可能对它们的所有分解状态都用-Data来连接，仅仅有状态$(a,1,1')$与$(b,1,1')$用-Data连接，因为在P_1中仅有状态1到状态1的转移为-Data。

经过以上的构造，完整的协议转换器可以取得，如图4所示。由此例可知，在构造前对实体P_1与Q_0的状态进行归并，达到简化的目的，使构造过程中的状态空间降到了$1*2*2=4$（如图3中(c)），使计算复杂度得到了改善。

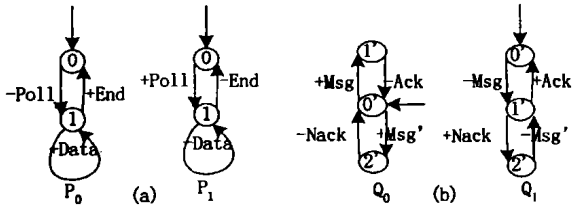

图1 (a)简单的Poll-End协议(P)
(b)简单Ack-Nack协议(Q)

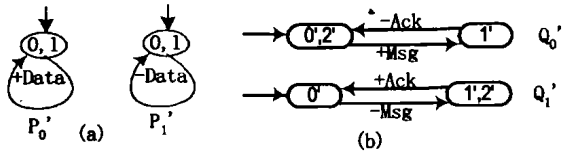

图2 (a)简化的Poll-End协议(P)
(b)简化Ack-Nac协议(Q)

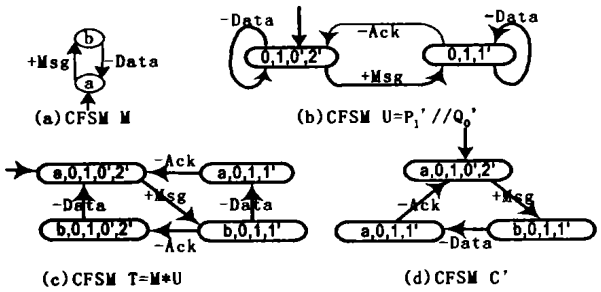

图3 Okumura转换过程

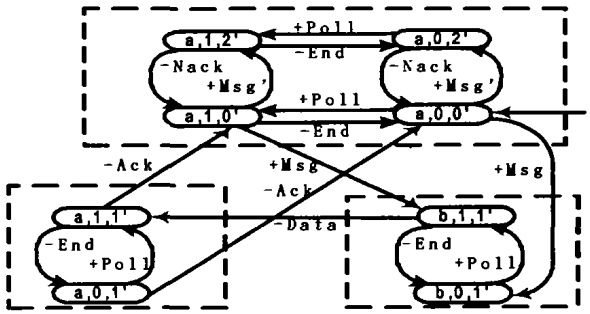

图4 经过分解C'得到转换器C

结束语 在两个异构网络之间进行通信，协议转换是一种有效的解决方法。在本文中，给出一种新的构造转换器方法。此方法在构造转换器之前对初始协议的CFSM模型进行了简化，协议实体中的状态、消息数量得到减少，使转换器构造过程中所用的状态空间得到降低，从而改善了构造过程中的计算复杂度。

参考文献

1 Okumura K. A formal protocol conversion method. In: Proc. of the ACM SIGCOMM Conf. on Communications Architecture & Protocols, 1986. 30～38

2 Saleh K, Jaragh M. Synthesis of protocol converters: annotated bibliography. Computer Standards &Interface 1998, 19: 105～117

3 Hallal H, Negulescu R, Petrenko A. Design of divergence-free protocol converters using supervisory control techniques. In: The 7th IEEE Intl. Conf. on Electronics, Circuits and Systems, 2000, 2(1): 705～708

4 Jaragh M, Saleh K. Synthesis of communications protocol converters using the timed Petri net model. Journal of Systems and Software, 1999, 47(1): 53～69

5 Huang C M, Lai H Y, Huang D T. A reduced incremental ECFSM-based protocol verification. In: The 17th Annual Intl. Conf. on Computer Software and Applications, 1993. 166～172

6 Calvert K L, Lam S S. Formal methods for protocol conversions. IEEE Journal on Selected Areas in Communications, 1990, 8(1): 127～148

（上接第27页）

4 Gong L, Needham R, Yahalom R. Reasoning about belief in cryptographic protocols. In: IEEE Computer Society Symposium in Security and Privacy. IEEE Computer Society Press, May 1990. 234～248

5 Abadi M, Tuttle M R. A Semantics for a Logic of Authencation (Extended Abstract). In: Proc. of the 10th Annual ACM Symposium on Principles of Distributed Computing, Aug. 1991. 201～216

6 Syverson P F, van Oorschot P C. On Unifying Some Cryptographic Protocol Logics. In: Proc. of the 1994 IEEE Computer Security Symposium on Security and Privacy, IEEE Computer Society Press, 1994. 14～28

7 Dolev D, Even S, Karp R. On the Security of Public Key Protocols. IEEE Transactions on Information Theory, 1983, 29(2): 198～208

8 Woo T W, C, Lam S S. A Semantic Model for Authentication Protocols. In: Proc. IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, May 1993. 178～194

9 Woo T Y C, Lam S S. Verifying Authentication Protocols: Methodology and Example. In: Proc. Intl. Conf. on Network Protocols, Oct. 1993

10 Ryan P, Schneider S. Modelling and analysis of Security protocols. Addison-Wesley Press, 2001

11 Abadi M, Gordon A D. A Calculus for Cryptographic Protocols: The Spi Calculus. Information and Computation: [SRC Research Report 149(January 1998)]. in preliminary form as Technical Report 414, University of Cambridge Computer Laboratory, Jan. 1997

12 Mao W. An Augmentation of BAN-Like Logics

13 Mao W, Boyd C. Towards Formal Analysis of Security Protocols. In: Proc. of the Computer Security Foundations Workshop VI, IEEE Computer Society Press, P. 147～158

14 Lowe G. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. In: Proc. of TACAS (Tools and algorithms for the Construction and Analysis of Sysmstems), Spring Verlag, 1996, 1055: 147～166

15 Lowe G. Towards a Completeness Result for Model Checking of Security Protocols. June 1999

16 Lowe G. Casper: A Compiler for the Analysis of Security Protocols. In: Proc. of 10th IEEE Computer Security Foundations Workshop, 1997. Also in Journal of Computer Security, 1998, 6: 53～84

17 Denning D E, Sacco G M. Timestamps in key distribution protocols. Communications of the ACM, 1981, 24(8): 533～536