计算机科学2002Vol. 29№. 3

CSC 模型的协议与语言*)

The Protocol and Languages for CSC Model

齐德昱　白　云　林伟建

(华南理工大学计算机系　广州510641)

Abstract　CSC model, advanced by the author in paper [9,10], is a new model towards the reform and extension of the traditional Client/Server models. It is mainly to enlarge and deepen the frameworks of C/S models in horizontal aspects, making it support a variety of interoperation modes including 1-n, n-1 and n-n. In this paper, we present CSC's main composition such as the message transfer protocol XITP(eXtensible Interface Transfer Protocol), data access model XAM(XML document Access Model), and a serials of XML based markup languages that express the interoperation and data in XITP messages.

Keywords　Distributed computing, Software Engineering

1. 引言

目前的网络计算模型大多基于客户/服务器模式，这种模式实质上是将任务串行地划分为两段，由客户和服务器分别执行。这种方式只适合一些简单的分布式计算，对于较复杂的计算，由于客户之间的通信不能透明地越过服务器，因此很不方便。最近有人提出三层服务模式。这种模式实质上是将任务串行地划分为三段，由三方顺序执行，开始的一方(记为 A)称为客户，A 的下一方(记为 B)是 A 的服务器，B 的下一方(记为 C)是 B 的服务器。请求发送的次序为 A-B-C，回送结果的次序是 C-B-A。照此思想，可有更多层的服务模式。三层(或更多层)服务模式更加细化了任务，适合一些大型的数据库应用系统，如数据仓库(DW:Data Warehouse)。

但是，不论是几层服务模式，其任务的划分都是串行的，对于一些复杂的需多方并发协作的任务，实现起来都很不方便。究其原因，主要是客户之间不能实现透明的通信。而导致这种情况的根本原因又在于客户/服务器模式的框架方面。

客户/服务器模式应用较多的是垂直框架，如各种客户/服务器模式的数据库管理系统、Internet 标准服务(Web, FTP, E-mail 等)，它们一般只能面向一些专门的问题。使用较为广泛的水平框架有 CORBA 和 COM+，它们同时也是分布对象技术，虽然能使各种模式交互，但主要是面向两层或多层的客户/服务器模式，对其他一些复杂的复合模式，如1-n 模式、n-1模式及 n-n 模式，框架化显得太弱了。客户/服务器/客户[9,10](CSC:Client/Server/Client)模型就是我们根据这些情况提出的。

2. CSC 模型逻辑结构

CSC 模型的物理结构仍同普通的 C/S 结构，但它们的 C 和 S 的功能和职责都有很大不同。首先，CSC 模型中的服务器的主要职能是接收客户的服务请求，按接收到的服务请求进行相应的工作或将服务请求委托给其他方处理，将处理结果回送到服务请求者或另外的客户，将来不及处理的请求或先行请求未完成的请求暂存起来、将各种消息(电话类、寻呼类、平信类、挂号信类、广播类、点播类等)进行相应的转发等。其次，在 CSC 中，C 与 S 的关系都高度模型/框架化。C 和 S 通过专门的 XITP 协议互操作，CSC 的客户内部也通过引入消息代理 XMA 实现框架化。图1给出了 CSC 模型的逻辑构成。

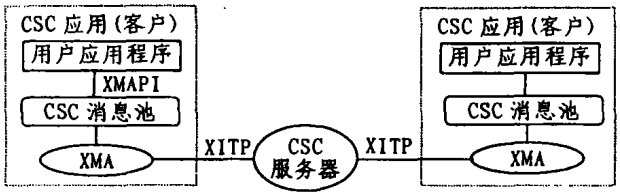

图1　CSC 模型逻辑结构

CSC 服务器充当 CSC 客户间的互操作的连接器。在逻辑上它直接与各 CSC 客户的消息代理 XMA 连接，负责转发 XMA 送来的消息。同时提供通信所涉及的各种服务，如名字服务、安全服务、时间服务、生命期服务、数据转换服务、并发控制服务等。

每个 CSC 客户都由两大部分构成：应用程序和 XITP 消息代理 XMA(XITP Message Agent)。XMA 与 CSC 服务器连接，负责接收来自 CSC 服务器的数据，或将用户程序的数据传送到 CSC 服务器，它也起到了隔离用户程序与 CSC 服务器的作用。

XMA 与 CSC 服务器及其它 XMA 的交互是通过 XITP 协议进行，而与用户应用程序的交互是通过共享消息池的方式进行的。应用程序通过 XIOM 消息 API(XMAPI)访问消息池，而 XMA 对消息池的访问作为 XMA 的组成成份。

XMA 是独立于用户程序的标准化的进程或线程，用户程序不需要直接与它互操作。对同种运行平台的用户程序，可以具有完全相同的 XMA。若 XMA 用 Java 实现，则可跨平台移动。从 XIOM 应用程序员角度看，他们所需进行的工作是创建并激活一个 XMA，然后在需要的时候调用 XMAPI 访问消息池。XIOM 并不对 XMA 和 XMAPI 的实现做硬性规定，只要能保证相应的功能/服务规范并能共享消息池即可。

用户应用程序除需使用 XMAPI 访问消息池外，还可根据需要使用 XML 文档存取模型 XAM(XML Document Access Model)的 API(XAAPI)和 XFML(eXtensible Form Markup Language)表单解释器。XAAPI 可对 XML 文档进行各种访问，而 XFML 解释器用于解释并控制用 XFML 编写的表单，从而实现可视化的互操作。

*)本课题为广东省重点科技攻关项目。齐德昱　博士，教授，主要研究兴趣：网络计算模型，大型软件开发，CIMS 集成等。

CSC 是继单机模型、终端/主机模型、客户/服务器模型后的另一种新模型。CSC 除具有传统客户/服务器模型的特点外，还可实现多客户间的透明的并发通信，解决了客户/服务器模型和多层服务模型的不足。

CSC 模型核心思想是，通信的各方不直接交互，而是通过公共的通信服务器进行交互。CSC 的引入，至少有如下几点影响：①支持多种形式的协作方式，而不仅仅是 C/S 方式，在 CSC 模型下，通信的各方可具有多种通信关系，如热连接、温连接、冷连接。②简化应用级的通信协议的使用。在 CSC 下，通信的双方可以互不认识。③解决胖瘦问题。目前，人们正在研究客户/服务器的胖瘦问题（任何分配客户与服务器的负荷），这里的 CSC 模型对该问题也预示着很好解决。④CSC 模型支持通信的双方以高度独立自治的方式工作，而不象客户服务器模型那样，将通信的双方紧紧地捆绑在一起。这种模型的引入，为实现软插件的应用代码透明奠定了基础。此外，CSC 模型的思想，还特别适合实现用户透明地建立虚拟服务器、增强/改造服务器的功能，我们在这里暂且称这种服务器为开放式服务器。这种服务器本身也是值得专门研究的。

3. XMA 及消息池结构

3.1 XAM

XITP 消息代理 XMA（XITP Message Agent）是 XIOM 客户的消息传递代理，同时也是 XITP 协议的实体。它与 CSC 服务器通过 XITP 协议传输 XIOM 客户之间的互操作接口。具体讲，XMA 将输出消息缓冲区中的消息发送到 CSC 服务器，并接收 CSC 服务器的消息，将其存放在输入消息缓冲区中。XMA 功能结构见图2，其各部分说明如下：

缓冲区监听器 BL（Buffer listener），动态检查消息池中输出缓冲区 IDOB（队列）中是否有由用户程序传输来的接口数据（XITP 消息），若有则取出消息，调用消息处理器处理消息。

XITP 输出消息处理器 OMP（Outgoing message processor），负责处理欲外发的 XITP 消息，并调用消息发送器 XSender 发送到 CSC 服务器。注意，外来消息则由用户程序从输入缓冲区 IDIB 取出，并调用 XITP 输入消息处理器 IMP（Incoming Message Processor）进行处理。

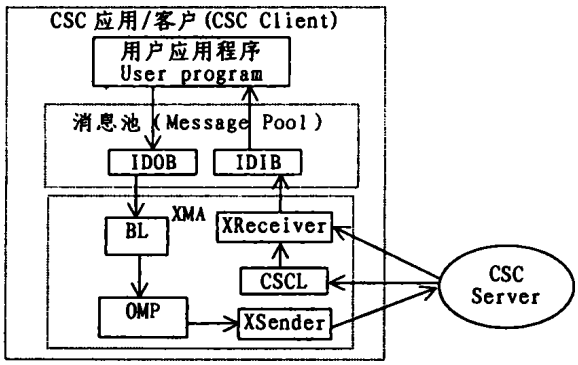

图2 XMA 功能结构

XITP 协议数据包监听器 CSCL（CSC server listener），其主要任务是动态检测来自 CSC 服务器的 XITP 协议数据包。若检测到，则通过 XITP 消息接收器 XReceive 将数据存入输入缓冲区 IDIB。

XITP 消息接收器（XReceive），主要负责从 CSC 服务器接收 XITP 消息。

XITP 消息发送器（XSender），主要负责将 XITP 消息发送到 CSC 服务器。

3.2 接口数据缓冲池

接口数据缓冲池是用户程序与 XMA 间的数据共享区。用户程序与 XMA 通过它进行通信与数据交换。接口数据缓冲池分输入缓冲区 IDIB（Interface Data Input Buffer）和输出缓冲区 IDOB（Interface Data Output Buffer）（输入输出的叫法相对于用户程序）。用户发往其它客户的数据先存放在 IDOB，来自于其它客户的数据则先到达 IDIB。

IDIB 和 IDOB 的存取由相应的 API 进行。具体实现时，XMA、用户程序及 IDIB/IDOB 是否在同一地址空间，有多种选择。例如，三者可以处在同一地址空间，或 XMA 与 IDOB/IDIB 在同一地址空间，或用户程序与 IDOB/IDIB 在同一地址空间。具体选择时应权衡运行效率和实现的方便性。

4. XITP 协议

我们提出的可扩展接口传输协议 XITP（eXtensible Interface Transfer Protocol）在 CSC 模型中占重要地位，它是各 CSC 客户进行互操作的接口语言。

XITP 主要用于描述 XIOM 客户之间的互操作动作与数据，以可扩展的标记语言 XML（eXtensibleMarkupLanguage）为描述工具，支持通信的各方以函数式语言的形式，交换结构化对话信息和数据信息。

XITP 是对等会话协议，不象 HTTP 那样是单纯的“请求－应答”模式，而是支持更广泛的平等对话模式。参加通信的双方，可以连续不断地以平等地位进行对话，没有严格的服务器与客户之分。因此，XITP 也是一种动态点对点通信协议。

XITP 具有三方面的可扩展性：形式可扩展、数据可扩展和互操作可扩展。XITP 的消息本身是 XML 文档，这不仅可使消息用统一的方法分析，而且导致了消息语法/形式的可扩展性。XITP 的数据是 XML 文档，由于 XML 具有很强的数据描述能力，故 XITP 除可传递一些非结构化的数据（多媒体数据）外，也能传递结构化的数据，如关系型文本数据库、层次型文本数据库等，这是 XITP 的数据可扩展性。XITP 的互操作主要用 XOIML 和 XFML 描述，前者是 XAM 的标记语言，对 XML 文档有强大的操作描述能力；后者是可视化交互表单的标记语言，描述可视化交互行为。它们与 XITP 的结合，使 XITP 的功能是可编程的，也可使 XITP 消息以可视化方式传递，这是 XITP 的互操作可扩展性。XITP 具有完善的安全保密体制，支持数据保密传输（单密钥和 RSA 密钥）、数据发送的签发、数据接收的签收。

XITP 消息是规范的 XML 文档，我们通过引入下列基于 XML 的标记语言描述消息。

XITP 构造标记语言 XCML（XITP Construction Markup Language）：用于标记 XITP 消息的构造（组成结构），是对 XITP 消息的总描述。

XML 对象访问标记语言 XOIML（XML Object Invocation Markup Language）是 XML 文档存取模型 XAM 对 XML 的映射，即 XAM 的 XML 形式，用于 XML 文档间或同一文档内各部分间的互操作。用 XOIML 编写的文档本身就是一个规范的 XML 文档。这部分内容统一在“XML 文档存取模型 XAM”一文中介绍。

可扩展的表单标记语言 XFML（eXtensibleFormMarkup Language）是基于 XML 的表单描述语言，它用 XML 描述用于构成屏幕对话界面的各种元素（成份），这种描述经相应的解释器解释后，就成为可视化的交互界面。这部分内容统一在

"可扩展的表单标记语言 XFML"一文中介绍。

XML 多媒体标记语言XMML(XMLMultimediaMarkup Language):用于在 XML 文档中携带多媒体数据。

XML 安全标记语言 XEML(XML Encrypt Markup Language):用于对 XML 文档的加/解密指示、数字签名指示。这部分内容统一在"XIOM 的安全保密"一文中介绍。

5. XITP 消息概述

XITP 分响应消息和请求消息两大类。每个 XITP 消息必须有 XITP 消息头,下面先给出 XITP 消息头的定义。

```
XITPMsgX ::= XITPMsg | HTTPMsgX
HTTPMsgX ::='〈HTTP' HTTPVersionDcl NameDcl IdDcl '〉'
   〈HTTP Message〉
   '〈/HTTP〉'
XITPMsg ::= '〈XITP' XITPVersionDcl NameDcl IdDcl '〉'
   SenderElem
   ReceiverElem
   RequestMsg | ResponseMsg
   '〈/XITP〉'
HTTPVersionDcl ::= 'Version=' XIOMVersionString
XIOMVersionString ::= 〈A quotation delimited string of digits and
   dots that represents the XITP message's version 〉
SenderElem ::= '〈Sender SenderNameDcl? SenderAddressDcl '/〉'
ReceiverElem ::= '〈Receiver ReceiverNameDcl? ReceiverAd-
   dressDcl '/〉'
SenderNameDcl ::= 'Name=' NameString
ReceiverNameDcl ::= 'Name=' NameString
SenderAddressDcl ::= 'Address=' AddressString
ReceiverAddressDcl ::= 'Address=' AddressString
AddressString ::= 〈A character string delimited by a quotation mark
   that represents the address of a XIOM registered user〉
NameString ::= 〈A character string delimited by a quotation mark
   that represents the name of a XIOM registered user〉
NameDcl ::= 'Name=' NameString
IdStringDcl ::= 'id=' NameString
```

这里给出了 XITP 的起始格式,适合于各种 XITP 消息。任何 XITP 消息都以 XITP 元素为根,消息的各具体部分都包含在 XITP 元素中。一个 XITP 扩展消息,或是扩展 HTTP 消息(XTTPMsgX),一条包含在 HTTP 元素中的 HTTP 消息(HTTPMsg),或者是一条纯 XITP 消息(XITPMsg)。

响应消息用于向请求者回送应答消息和数据。当请求者发出请求消息后,被请求者收到并处理请求消息后回送此消息。由于 XITP 允许有反请求,故该消息不局限于对请求消息的响应。响应消息的基本形式为:

```
ResponseMsg::='〈Response' ConnectionDcl? ResponseTypeDcl' 〉'
   ExecuteStatusElem
   (ResponseDataElem | RequestMsg)?
   '〈/Response)'
ConnectionDcl ::= 'Connection=' ConnectionType
ConnectionType ::= 'Keep' | 'Unkeep'
ResponseTypeDcl ::= 'ResponseTo=' XITPMsgType
XITPMsgType ::= 'GET' | 'STATUS' | 'INSERT' | 'DELETE'
   | 'GETCERFITICATE'
   | 'SENDCERFITICATE'
   | 'GETSIGNATURE'
   | 'SENDSIGNATURE'
   | 'RESPONSE'
ResponseDataElem ::= '〈ResponseData)'
   XMLElem
   CertificateElem?
   SignatureElem?
   '〈/ResponseData)
XMLElem ::= 〈Any XML element that can include XIML markups
   except for XCML 〉
SignatureElem ::= SignatureMarkupElem | ReceiptElem
```

请求消息由请求者发送给被请求者,用于描述请求操作,分数据请求消息与安全请求消息两类。它的基本形式为:

```
RequestMsg ::= '〈Request' ConnectionDcl? '〉'
   ExecutionConditionElem ?
   CertificateElem?
   ActionElem?
   '〈/Request〉'
ExecutionConditionElem ::= LogicExpr
ActionElem ::= '〈Action' ReturnDcl? GetStatusDcl? '〉'
   ActionBody
   '〈/Action〉'
ActionBody ::= XOIMLActionElem
   | GetCertificateElem
   | GetSignatureElem
XOIMLActionElem ::= '〈XOIMLAction'
   SendSignatureDcl? GetSignatureDcl '〉'
   XOIMLElem
   '〈/XOIMLAction〉'
XOIMLElem ::= 〈 XOIML element 〉
GetCertificateElem ::= '〈' 'GetCertificate' '/〉'
GetSignatureElem::='〈GetSignature'SignatureTypeDclObje-
ctElemDcl '/〉'
SignatureTypeDcl ::= 'Type=' Signature' | 'Receipt'
IDValue ::= 〈A string representing a XML element〉
```

6. XITP 的互操作语言

XITP 的协议实体是用基于 XML 的标记语言描述的互操作指令和数据,这些标记语言主要包括 XOIML、XAM、XFML、XMM、XEML 等。

XAM(XML document Access Model)是 XML 文档存取模型,通过引入树结构的表式视图,使树结构与相应的串行化结构有平滑而灵活的关联,对树结构的存取有独到的作用。

XOIML(XML object invocation markup language)是对象访问标记语言,用于描述对对象的访问/调用,采用函数式语言的特点,将有效地解决用 XML 标记对象访问的问题。XOIML 对 XAM 的应用,使对 XML 文档的存取的描述 XML 化,从而用在 XITP 中表达复杂的操作。XITP 与 XOIML 的结合,使得 XITP 直接具有强大的对象访问能力,特别是 XOIML 对 XIOM 的文档存取模型的具体应用,使得 XITP 对 XML 文档具有强大而灵活的操作描述能力,这种描述能力事实上是函数式程序设计语言的子集,这点是 XITP 的操作可扩展性。

XFML(Extensible form markup language)是可视化交互表单标记语言。XITP 与 XFML 的结合,使 XITP 具有了可视化交互的能力,从而可用于电子商务的可视化交互。

XEML(XML Encrypt Markup Language)是可扩展的安全保密标记语言,用于在 XITP 消息中指示 XML 文档的加/解密指示、数字签名。

限于篇幅,这些语言的定义不在这里给出。

参 考 文 献

1 OMG. The common object request broker: architecture and specification. http://www.omg.org,1998
2 Maffeis S. Piranha: a CORBA tool for high availability. Computer,1997,30(4)
3 Pnerta A R. A model-based interface development environment. Computer,1997,14(3)
4 Rossak W,et al. A generic model for software architectures. IEEE Software,1997,14(4)
5 Negro A,et al. Efficient distributed selection with bounded messages. IEEE Transactions on Parallel and distributed systems, 1997,8(4)
6 Kelling C,Henz J, Hommel G. Modeling and evaluation of a distributed controller architecture by stochastic Petri nets. International Journal of Mini and Microcomputers,1996,18(3)
7 Chu W W,et al. KmeD: a knowledge-based multimedia distributed database system. Information Systems,1995,20(2)
8 Schneberger S L. Distributed computing environment: effects on software maintenance difficulty. The Journal of Systems and Software,1997,37(2)
9 齐德昱.客户/服务器/客户模型与 DAOIS.小型微型计算机系统,1999,20(8):567～569
10 齐德昱,等.客户/服务器/客户模型的一种扩展.计算机工程与应用,1999,35(3):8～9
11 齐德昱.信息系统的 LOODS 抽象模型.计算机研究与发展,1996,33(11):849～853
12 Qi Deyu(齐德昱). LOODS: a new learning-based object-oriented system development environment. In: APSEC/ICSC'97 conf. proc. published by the IEEE Computer Society Press