第 37 卷 第 8 期
Vol.37 No.8

计 算 机 工 程
Computer Engineering

2011 年 4 月
April 2011

• 网络与通信 •

文章编号：1000—3428(2011)08—0102—03 文献标识码：A 中图分类号：TP393

光纤通道流量控制的 OPNET 仿真模型设计

蔡昭权[1]，秦磊华[2]，罗 伟[1]，卢庆武[1]

(1. 惠州学院教育技术中心，广东 惠州 516007；2. 华中科技大学计算机科学与技术学院，武汉 430074)

摘 要：通过分析光纤通道流量控制协议的基本工作原理，提出一个基于 OPNET 的光纤通道流量控制协议仿真模型，给出发送端和接收端节点模型及进程模型的设计方案，并在 OPNET 环境下利用 C 语言实现该模型。仿真实验结果证明其可以正确地执行光纤通道流量控制协议，且模型简单直观、可扩展性好。

关键词：光纤通道；流量控制；进程模型；仿真模型；OPNET 仿真软件

Design of OPNET Simulation Model for Fibre Channel Flow Control

CAI Zhao-quan[1], QIN Lei-hua[2], LUO Wei[1], LU Qing-wu[1]

(1. Educational Technology Center, Huizhou University, Huizhou 516007, China;

2. School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

【Abstract】By analyzing the fundamental working tenet concerning the flow control agreement of the Fibre Channel(FC), this paper proposes an OPNET-based simulation model for controlling the flow of FC. The design program models the process nodes of sending and receiving. Simulation results show that the model proposed can abide by the flow control agreement of FC precisely with the advantages of simplicity, straghtforwardness and extensibility under the environment of OPNET and using C programming language as the development tool.

【Key words】Fibre Channel(FC); flow control; process model; simulation model; OPNET simulation software

DOI: 10.3969/j.issn.1000-3428.2011.08.035

1 概述

光纤通道存储区域网(Fibre Channel Storage Area Network, FC SAN)以其高性能、易扩展、高可靠等优点，已成为主流网络存储技术之一，并广泛应用于各类数据中心。随着容灾应用的普及，越来越多的 FC SAN 通过广域网互连，但距离的增加给 FC SAN 流量控制协议的性能带来较大的损失[1-3]。本文利用当前流行的网络仿真工具 OPNET[4]建立光纤通道流量控制协议仿真模型，为进一步研究光纤通道业务流量和业务性能提供参考。

2 FC 流量控制原理

FC 流量控制机制建立在信用(Credit)的基础上，是光纤通道协议 FC 第 2 层的功能之一[5]。Credit 值即接收缓冲区的数量，代表接收设备接收帧的能力，其初始值由通信双方在登录时协商。FC 协议中定义了增性和减性 2 种 Credit 管理方式，图 1 为基于 Credit 减性管理的 FC 流量控制模型。

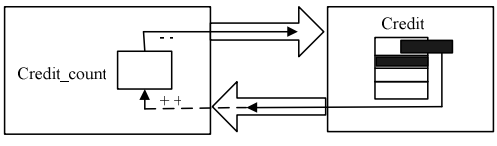

图 1 基于 Credit 减性管理的 FC 流量控制模型

发送方 FC 流量控制算法如下[6]：

(1)Credit_count=Credit;

(2)if Credit_count≥0 then send one FC Frame and Credit_count= Credit_count −1;

(3)if Credit_count>=0 then goto (2)

(4)Waiting for FC updating signal from receiver;

(5)Credit_count=Credit_count +1

(6)Goto (2)

接收方的流量控制算法很简单。当接收缓冲区不为空时，每从中取出一个 FC 帧，就向发送方反馈应答信号。

增性管理的 FC 流量控制工作原理与减性管理的 FC 流量控制类似，不同的是，Credit_count 的初值为 0，每发送一个 FC 帧，Credit_count 加 1，每收到一个接收方的反馈信息，Credit_count 减 1，当 Credit_count 的值等于 Credit 时，发送方将暂停新数据的发送。

基于 Credit 的流量控制协议是一种以信宿为基础的流量控制方法，当接收方没有向发送方分配和颁发信用时，发送方就不能发送任何数据，避免了由于接收缓冲区溢出导致的帧丢失，减小了对整个 FC 帧序列进行重传的概率，从而提高了链路的利用率和数据传输的性能。

3 基于 OPNET 的 FC 流量控制仿真模型设计

3.1 OPNET 的建模机制

OPNET 采用了包含网络、节点及进程的 3 层建模机制。其中，网络域为最上层，反映由多个设备构成的网络拓扑结构和应用连接关系；节点域为中间层，由相应的协议模块构成，用于实现网络层设备的功能并反映设备的特性；进程域为最底层，它以状态机的形式来描述节点域中的协议模块，反映协议具体功能的实现。3 层模型建模从低层到高层依次与实际的协议、设备和网络完全对应，全面反映网络的相关特性。因此，基于 OPNET 建模的关键是节点模型和进程模型的建立。

基金项目：广东省科技计划基金资助项目(2008B010200002)

作者简介：蔡昭权(1970－)，男，教授、硕士，主研方向：网络存储系统，数据库技术，信息安全；秦磊华，副教授、博士；罗 伟、卢庆武，高级工程师、硕士

收稿日期：2010-12-16 **E-mail：**cai@hzu.edu.cn

3.2　FC 流量控制节点模型的建立

由于最简单的 FC 流量控制模型只需要包含一个发送节点和一个接收节点[7]，因此本文建立的发送节点模型和接收节点模型分别如图 2 和图 3 所示。

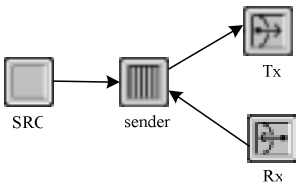

图 2　发送节点模型

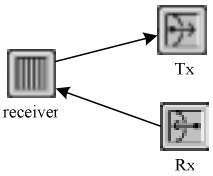

图 3　接收节点模型

图 2 中的发送节点模型由 4 个功能模块组成，其中，SRC 用于产生 FC 原始帧，该功能模块使用 OPNET 提供的进程模型 simple_source；sender 为发送节点的核心功能模块，用于实现 FC 协议发送端的算法；Tx 和 Rx 为 OPNET 提供的总线发送模块和接收模块，分别用于发送 FC 帧和接收来自接收端的 Credit 更新信号。

图 3 中的接收节点模型由 3 个功能模块组成，其核心是 receiver，用于实现 FC 协议接收端算法；Tx 和 Rx 分别与发送节点模型中的 Rx 和 Tx 连接，向发送端发送 Credit 更新信号和接收来自发送端的 FC 帧。

3.3　FC 流量控制进程模型的建立

3.3.1　sender 进程模型

实现 sender 功能模块的状态机如图 4 所示，由 5 个状态组成，除 idle 为非强制状态外，其余 4 个均为强制状态。该进程模型是在 OPNET 自带进程模型 abc_fifo 的基础上结合 FC 流量控制协议的发送端工作原理修改而成的。

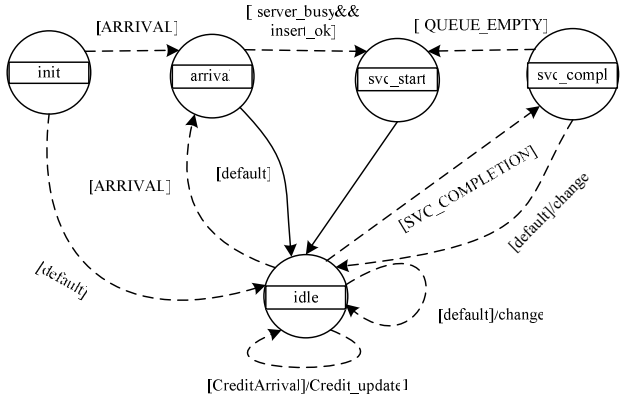

图 4　发送节点中 sender 模块状态机

init 状态：用于初始化状态变量，如设置 Credit 初始值，执行 Credit_count=Credit，并设置其他相关状态变量等。该状态有 2 个出口，当有新的 FC 帧到达时，转移到 arrival 状态，其他情况下转移到等待状态 idle。

arrival 状态：从接收端接收 FC 帧，并送入发送缓冲队列保存。该状态有 2 个可能的出口，当 FC 帧正确地送入发送队列时，转移到 svc_start 状态，其他情况下转移到等待状态 idle。

svc_start 状态：从发送队列取出 FC 帧，调整 FC 帧发送间隔。该状态只有一个出口——转移到等待状态 idle。

svc_compl 状态：发送 FC 帧，并执行 Credit 的管理策略修改 Credit_count。该状态有 2 个出口，发送方队列不为空时，转移到 svc_start 状态；其他情况下转移到等待状态 idle，并修改发送服务端的状态，由不空闲状态转换到空闲状态。

idle 状态：等待转移状态的产生。该状态在不同条件下可能会向包括自己在内的 3 个状态转移。其中，当有新的 FC 帧到达时，转移到 arrival 状态；当 Credit_count 非负(对 Credit 减性管理而言)且发送队列不为空时，转移到 svc_compl 状态；有 Credit 更新信号到达时，转移到自己，并修改 Credit_count 的值；其他情况下也转移到自己，并置服务器状态为空闲。

3.3.2　receiver 进程模型

实现 receiver 功能模块的状态机如图 5 所示，也由 5 个状态组成，除 idle 为非强制状态外，其余 4 个均为强制状态。该进程模型在 OPNET 自带进程模型 abc_fifo 的基础上结合 FC 流量控制协议中接收端的工作原理修改而成。

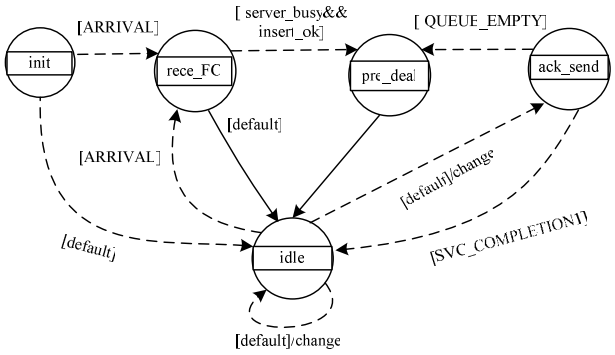

图 5　接收节点中 receiver 模块状态机

init 状态：用于初始化状态变量，如设置缓冲队列长度 queue_length=Credit。该状态转移到 rece_FC 和 idle 的条件与图 4 中的情况相同。

rece_FC 状态：接收 FC 帧并送至缓冲队列保存，每插入一个 FC 帧，queue_length 加 1。该状态有 2 个出口，当 FC 帧接收完毕时，转移到 pre_deal 状态，其他情况下转移到 idle 状态。

pre_deal 状态：从接收队列中取出 FC 帧，调整 FC 帧处理间隔。该状态只有一个出口——转移到 idle 状态。

ack_send 状态：处理 FC 帧，queue_length 减 1 并发送 ACK 更新信号。该状态有 2 个出口，当接收队列不为空时，转移到 pre_deal 状态；其他情况下转移到 idle，并置处理服务器为空闲状态。

idle 状态：不执行任何代码，只是等待转移状态的产生。该状态在不同条件下可能会向包括自己在内的 3 个状态转移。其中，当有新 FC 帧到达时，转移到 rece_FC；当接收队列不为空时，转移到 ack_send；其他情况下转移到自己，并置接收服务器状态为空闲。

3.4　转移条件及统计量定义

3.4.1　转移条件的定义

进程模型状态机中不同状态之间的变迁需要满足一定的条件，包括确定的条件和隐含的条件。如图 4 中当处在 arrival 状态时，若条件(!server_busy && insert_ok)满足，则转移到状态 svc_start，这里的条件是确定条件；在 arrival 状态时，在除(!server_busy && insert_ok)以外的其他条件下，则转移到状态 idle，称这类条件为隐含条件。

状态机中确定的转移条件在进程编辑器中定义，有 2 种方法：(1)在状态变量(State Variable, SV)或临时变量(Temporary Variable, TV)中定义相关的变量。(2)通过 HB(Header Block)定义[8]。如条件(!server_busy && insert_ok)中的 2 个变量 server_busy 和 insert_ok 分别在 SV 和 TV 中定义；而图 4 中的状态转移条件 arrival 是在 HB 中采用下列语句进行定义的：

#define ARRIVAL ((op_intrpt_type () = = OPC_INTRPT_STRM) && (op_intrpt_strm () = = SRC_IN_STRM)&& (credit_count >=0))

除定义转移条件外，在状态转移过程中还可以通过执行代码完成相应的工作，如图 4 中有一个从状态 idle 到自身的转移条件 CreditArrival/Credit_Update1，其中，CreditArrival 为转移条件；Credit_Update1 为函数名，表示执行该状态转移过程中要完成的操作。这个条件和执行函数也是在 HB 中定义的，定义过程如下：

#define CreditArrival ((op_intrpt_type () == OPC_INTRPT_STRM) &&(op_intrpt_strm () == RX_IN_STRM))

static void Credit_update(void);

Credit_update(void)的函数体在函数代码块 FB(Function Block)中定义。图 4 和图 5 中其他确定的状态转移条件及状态转移时执行的函数的定义与此基本相同，在此不再重复。

3.4.2 统计量的定义

网络仿真的过程就是对反映网络某方面特征的变量进行跟踪的过程，这些变量即统计量。统计量的定义包括 3 步：

(1)在进程编辑器的 interfaces 下拉菜单中进行定义，可根据应用的需要定义成局部统计量或全局统计量。

(2)在相应状态机的 init 状态中用 OPNET 的库函数 op_stat_reg 对统计量进行注册。如对统计量 credit_count 的注册操作为：

credit_count_t = op_stat_reg("credit_count",OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

(3)用 op_stat_scalar_write()函数在相应状态生成统计信息的记录。之后统计量会以图形的方式动态显示仿真过程中的值，供分析仿真结果使用。

3.5 代码实现

图 4 和图 5 中每个状态的功能是通过程序代码来实现的，每个状态被其名字分成上、下 2 个部分，进入状态时要实现的功能代码在上部分中编辑，称为入口代码；离开状态时要实现的功能代码在状态下部分中编辑，称为出口代码。对于强制状态而言，每次都要依次执行入口代码和出口代码，而对非强制状态而言，入口代码在进入该状态时执行，出口代码是否执行取决于是否发生状态转移。

图 5 是实现 FC 流量控制协议的核心，主要包含 5 个模块，其中，idle 状态中没有任何代码。其他 4 个状态中均设计了入口代码。

(1)receiver 模型中状态转移条件的定义

根据图 5，receiver 模型中主要用到 3 个状态转移条件，其定义如下：

#define QUEUE_EMPTY (op_q_empty ()) /*调用 OPNET 的库函数判断队列是否为空*/

#define SVC_COMPLETION ((op_intrpt_type () == OPC_INTRPT_SELF) &&(!QUEUE_EMPTY))

/*调用 opnet 中断类型函数和上一步定义的 QUEUE_EMPTY 函数，并将 2 个函数的返回值相与，作为从 idle 到 scv_compl 的转移条件*/

#define ARRIVAL (op_intrpt_type () ==OPC_INTRPT_STRM && op_intrpt_strm () == RX_IN_STRM)

/*调用 opnet 函数判断是否有新的数据包到达*/

(2)init 模块的算法实现

该模块主要是对程序中用到的变量进行初始化：

```
server_busy = 0; /*定义服务器的状态，为 0 表示服务器不忙*/
ch=200;/*定义 FC 协议 Credit 初始值*/
queue_length=0; /*定义接收队列的初始值为 0*/
insert_ok = 0; /*定义队列的状态，为 0 表示 FC 帧插入操作不正常，为 1 表示 FC 帧插入队列操作正常*/
queue_length_t= op_stat_reg ("queue_length", OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL); /*注册队列长度统计量*/
own_id = op_id_self ();
```

(3)rece_FC 模块实现

```
pkptr=op_pk_get(op_intrpt_strm()); /*从接收的流中获得 FC 帧*/
if (op_subq_pk_insert (0, pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
    { op_pk_destroy (pkptr);
    insert_ok = 0; } /*队列满，插入不成功*/
else{
  insert_ok = 1;
  queue_length=queue_length+1;
  op_stat_write (queue_length_t, queue_length); } /*插入成功*/
```

(4)pre_deal 模块实现

该模块实现数据接收的预处理：

```
pkptr = op_subq_pk_access (0, OPC_QPOS_HEAD);/*定义一个指向队列头的指针*/
pk_len=1000+op_dist_uniform(24);/*定义 FC 帧的长度*/
pk_svc_time = (double) pk_len / (service_rate + op_dist_uniform(2)); /*定义等待的时间*/
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);/*触发 FC 帧的处理*/
```

(5)ack_send 模块的算法实现

该模块主要实现的功能是从接收队列中取出一个 FC 帧进行处理，同时向发送端返回 Credit 更新信号：

```
pkptr1 = op_subq_pk_remove (0, OPC_QPOS_HEAD);
if (fc_count >=0 && fc_count< ch )
            { op_pk_send_forced (pkptr1, TX_OUT_STRM);
                fc_count=fc_count-1; };
server_busy = 0;
```

图 4 的模型实现方法比图 5 的模型更为简单。由上述实现代码可以看出，该模型代码的时间复杂性和空间复杂性都很低，算法具有良好的性能。

4 模型功能验证

利用本文建立的节点模型和进程模型建立了如图 6 所示的 FC 网络模型。然后分别在发送进程模型和接收进程模型中定义统计量 Credit_count 和 Queue_length，用于跟踪发送方信誉值和接收方缓冲队列长度即时变化情况。仿真结果分别如图 7 和图 8 所示。

图 6 FC 网络模型

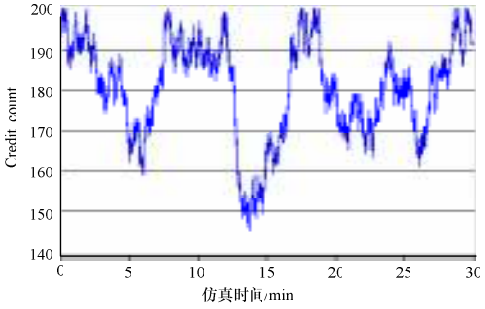

图 7 发送方信誉值的动态变化

(下转第 114 页)