Modeling of an N_port’s FSM in Fibre Channel with OPENT*

LEI YANJING, MIAO KEJIAN, YIN XIAOYAN, ZHENG KUI, KANG JICHANG
School of Computer Science
Northwestern Polytechnical University
P.O. Box 400, No.127 West Youyi Road, Xi’an City
THE PEOPLE’S REPUBLIC OF CHINA

Abstract: - When an FC_port is powered-on, reset or encounters failures, a complex state machine will be performed to recover from these abnormal conditions. Design of an FC_port’s FSM is the entrance and jumping-off point of implementing the whole FC_port. Starting with an N_port, which is the most basic and common port in FC, the FSM mechanism is analyzed and modeled with OPNET. After studying the processes of four kinds of Primitive Sequence Protocols, an important conclusion can be deduced: whichever state an N_port may be, if it sends the Primitive Signals and Primitive Sequences just received, it can switch to the Active State automatically after at most six states. Following this rule, the design of an N_port’s FSM can be simplified greatly. The simulation results provide a reference for the design and implementation of an FC_port in real life.

Key-Words: - Fibre Channel, N_port, FSM, OPNET, Modeling, Primitive Sequence Protocols

1 Introduction

Today, high-speed network has three distinct development trends: network with the topology of switch, network with the medium of optical fiber and network with the speed of Gigabit [1]. Coincidently, American National Standard Institute (ANSI) has released a high-speed serial data transmission standard named Fibre Channel (FC). FC supports Fabric (switch) topology and optical fiber medium, and it provides 800Mbps~4Gbps transmission rate. FC combines the features of network and channel, and provides a highly reliable, low-cost and real-time scheme for high-speed network [2].

In FC, a port (named FC_port) is the most basic communication entity. All the devices attached to FC network must have at least a port which manages the transmission of data. When a port is powered-on or reset, a series of special transitions are needed to change from initial state to working state. During the transmission, if abnormal conditions such as link error, loss-of-signal and buffer overflow occur, some special protocols will be started up to recover from these failures. The special transitions and protocols above can be accomplished by Finite State Machine (FSM). FSM is the essential function of an FC_port, and its correctness determines whether the whole FC_port works properly or not. However, a port has many kinds of states, so the FSM is very complex. In this paper, by modeling and simulating the FSM of a port with OPNET and analyzing the results, we expect to reveal the working principle of FSM mechanism and the transition processes of four kinds of Primitive Sequence Protocols. Since the N_port is the simplest and most common port in FC, we use it as an example in the whole dissertation.

2 States and Switch Relationship of an N_port

To make an N_port work properly, a set of Primitive Signals and Primitive Sequences are defined in FC, namely IDLE, LR, LRR, NOS, OLS [3]. IDLE is a mandatory Primitive Signal while the others are mandatory Primitive Sequences. The meaning and usage of them are as follows:

(1) IDLE (Idle). It indicates that the port sending this signal is ready for transmission of sequences and frames and the current link has been synchronized.
(2) LR (Link Reset). It’s used for the Link Reset Protocol.
(3) LRR (Link Reset Response). It indicates that the current port has received LR.
(4) NOS (Not Operational). It indicates some failures are encountered in the current port.
(5) OLS (Offline). It indicates the port sending this sequence will be offline.

Based on these Primitive Signals and Primitive Sequences, nine states of an N_port are specified in FC as Table 1, and the switch relationship of these states is shown in Table 2 [4].

---

* This paper is supported by the National 863 Project of China (No. 2003AA001018) and the Aeronautical Science Foundation of China (No. 02F53031).
Table 1. The Definition of an N_port’s States

<table>
<thead>
<tr>
<th>The Whole States</th>
<th>Sub-States</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC (Active State)</td>
<td>AC (Active State)</td>
<td>Indicating that a port is in working state.</td>
</tr>
<tr>
<td>LR (Link Recovery)</td>
<td>LR1 (LR Transmit State)</td>
<td>Indicating that a port is sending LR.</td>
</tr>
<tr>
<td></td>
<td>LR2 (LR Receive State)</td>
<td>Indicating that a port has received LR.</td>
</tr>
<tr>
<td></td>
<td>LR3 (LRR Receive State)</td>
<td>Indicating that a port has received LRR.</td>
</tr>
<tr>
<td>LF (Link Failure)</td>
<td>LF1 (NOS Receive State)</td>
<td>Indicating that a port has received NOS.</td>
</tr>
<tr>
<td></td>
<td>LF2 (NOS Transmit State)</td>
<td>Indicating that a port is sending NOS.</td>
</tr>
<tr>
<td>OL (Offline)</td>
<td>OL1 (OLS Transmit State)</td>
<td>Indicating that a port is sending OLS.</td>
</tr>
<tr>
<td></td>
<td>OL2 (OLS Receive State)</td>
<td>Indicating that a port has received OLS.</td>
</tr>
<tr>
<td></td>
<td>OL3 (Wait for OLS State)</td>
<td>Indicating that a port is waiting for OLS.</td>
</tr>
</tbody>
</table>

Table 2. Switch Relationship of an N_port’s States

<table>
<thead>
<tr>
<th>Current State</th>
<th>AC</th>
<th>LR1</th>
<th>LR2</th>
<th>LR3</th>
<th>LF1</th>
<th>LF2</th>
<th>OL1</th>
<th>OL2</th>
<th>OL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>AC</td>
<td>LR1</td>
<td>LR2</td>
<td>LR3</td>
<td>LF1</td>
<td>LF2</td>
<td>OL1</td>
<td>OL2</td>
<td>OL3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primitive Sequence transmitted while in the states Above</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Event</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR Arrival</td>
<td>LR2</td>
</tr>
<tr>
<td>LRR Arrival</td>
<td>LR2</td>
</tr>
<tr>
<td>IDLES Arrival</td>
<td>AC, AC, AC</td>
</tr>
<tr>
<td>OLS Arrival</td>
<td>OLS</td>
</tr>
<tr>
<td>NOS Arrival</td>
<td>LRR</td>
</tr>
<tr>
<td>Loss-of-Signal</td>
<td>LRR, LRR</td>
</tr>
<tr>
<td>Loss-of-Sync&gt;Limit</td>
<td>OLS, OLS</td>
</tr>
<tr>
<td>Event time-out</td>
<td>N/A</td>
</tr>
<tr>
<td>Link time-out</td>
<td>LR1</td>
</tr>
</tbody>
</table>

According to Table 2, the state diagram of an N_port can be outlined as Fig.1. In Fig.1, we can find that there are nine states in each FC_port and every state has many switch branches, so the whole state diagram is quite complex. Analyzing Fig.1 directly is not only inefficient but also difficult to find some intrinsic rules. Fortunately, OPNET, an excellent simulation platform, is extremely adept in simulating some complex behaviors with brief and clear state machines. Moreover, OPNET has a strong function of playing animation and debugging. By observing the animation or analyzing debugging information, we can easily make out the whole working process and find some essential rules. Therefore, the N_port’s FSM is modeled and simulated on OPNET platform in this paper.

3 Modeling of an N_port’s FSM with OPNET

By far, OPNET is the most advanced development and application platform for network simulation in the world [5]. It provides a hierarchical modeling mechanism: process level, node level and network level. Process depicts the behaviors of an object, and it’s the carrier of algorithms and protocols. Process is accomplished by FSM in which the states and switch conditions can be described in C/C++ language. In addition, OPNET provides a kernel function library which contains more than 400 kernel functions. FSM, C/C++ language and the kernel function library constitute the programming core of OPNET, named Proto-C [6]. At node level, separate components are combined to form devices, and at network level, these devices can be connected to form network. According to the three-level modeling mechanism, the FSM of an N_port is modeled and simulated.

3.1 Modeling of Packets in an N_port’s FSM

During the state switch of an N_port’s FSM, only
Primitive Signals and Primitive Sequences are transmitted. So the packet is defined as a 4-byte data structure named FC_packet.

### 3.2 Modeling of Links in FC

The handshake between two N_ports needs a full-duplex link at the rate of 1.0625Gbps. The data format supported on this link must be FC_packet. In ideal conditions, there is no interference on the link, and the BER (Bit Error Rate) is zero. However, to simulate the actual conditions such as link failure and loss-of-signal, some link interference models and error correction models can be applied. These models can be realized conveniently by selecting the existing options provided by the OPNET kernel or redefining some new models yourself. Besides, point-to-point transmission delay model (dpt_txdel) and point-to-point promulgation delay model (dpt_prodel) can be set to calculate the delay-time. The link model defined above is named FC_link.

### 3.3 Modeling of the Process of an N_port

Process is used to describe complex behaviors such as communication protocols and queuing policies. The process model of an N_port’s FSM designed in this paper is shown as Fig.2 (c). In this process model, “wait” is an unforced state and the rest states are all forced states. When in an unforced state, the process will be blocked in the current state after executing Enter Executives, and the control of simulation is handed over to the OPNET kernel until next event’s arrival. But when in a forced state, after executing the Enter Executives and Exit Executives orderly, the process will switch to other states immediately.

When simulation begins, the N_port is in “init” state. After a quick initialization, the N_port will enter “wait” state and remain in this state. Hereafter, when triggered by a packet’s arrival, the packet will be abstracted in the Exit Executives according to the interrupt types and stream indexes. Then the “wait” can determine which state it will switch to according to the current state and the received packets. Switch conditions (To_AC~To_OL3) can be defined as Macros in Header Block. When changing from “wait” to another state, the current state of the N_port is modified firstly and then the corresponding
To LR3 can be defined with Macros in Header Block as follows:

# define LRR_ARRIVAL (op_intrpt_type is a interrupt of stream && packets on rcv stream are LRRs)
# define To_LR3 (LRR_ARRIVAL && Current_State = = AC / LR1 / LR2 / LR3 / OL2)

The behaviors of LR3 are defined in Enter Exects as follows:

If (PACKETS ARRIVAL) printf Current_State and the contents of received packets;
Current_State = LR3;            /* Modify the Current_State to LR3. */
Send_IDLES( );                     /* Creat a packet, init it as IDLE and send it continuously at least 3 times. */
Display ODB information:  current simulate time, node ID, packet ID and states transition.

As an example, the switch condition of To_LR3 and the behaviors of "LR3" are illustrated as Fig.3.

By comparing Fig.1 with Fig.2 (c), it's easy to find that the N_port's FSM model implemented with OPNET is much clearer and briefer than the ordinary model. What's more, since the working process of each state is similar, the codes of every state in Fig.2 (c) can be reused with quite a little modification.

3.4 Modeling of Nodes in FC

Node model is composed of three components: src, proc and rcv/xmt. src is used to generate packets. proc processes these packets, and its behaviors are realized by process model. rcv/xmt simulates the interfaces between the port and external devices, and their attributes (such as data rate and packet types supported) must be consistent with the ones of FC_link. Node model is shown as Fig.2 (b).

3.5 Modeling of Network

In this application, the network model is the full-duplex communication in point-to-point between two N_ports. In the case when delay is insignificant, the geographical places of these two N_ports can be ignored. Network model is shown as Fig.2 (a).

4 Analysis of Simulation

To deal with the abnormal conditions during the communication, four Primitive Sequence Protocols...
are defined in FC. These protocols are typical instances of an N_port’s FSM mechanism.

1) Link Initialization Protocol
When an N_port is powered-on, reset or has been offline and desires to come back online, it will perform Link Initialization Protocol. This protocol begins when the N_port enters OL1 State and ends when the N_port enters Active State. Suppose that two N_ports named A and B are ready to perform Link Initialization Protocol, their initial states should be set to OL1 in “init” in Fig.2 (c) and they both send OLS sequences continuously to each other. Then the two ports will remain in OL1 state and the processes will be blocked. Upon receiving OLS sequences from the other, these two N_ports will both enter OL2 state and then send LR sequences to each other. Similarly, while in OL2 state and receiving LR sequences, they will enter LR2 state and then send LRR sequences. While in LR2 state and receiving LRR sequences, they will enter LR3 state and then send IDLEs. While in LR3 state and receiving IDLE signals, these two N_ports both enter Active State. Upon entering Active State, the N_port can work properly. Hereto, the whole process of Link Initialization Protocol is finished. The transition process of this protocol is shown as Fig.4 (a).

2) Link Reset Protocol
When Connection Recovery, buffer overflow or link timeout are detected, an N_port should perform Link Reset Protocol. By executing this protocol, the N_port can resume operating normally. Since the processes of the rest three protocols are analyzed in the same way as the Link Initialization Protocol, we don’t describe them herein again. The transition process of Link Reset Protocol is shown as Fig.4 (b).

3) Link Failure Protocol
When failures such as Loss-of-Synchronization and Loss-of-Signal are detected, an N_port should perform Link Failure Protocol. By executing this protocol, an N_port can recover from these failures. The process of Link Failure is shown as Fig.4 (c).

4) Online-to-Offline Protocol
An N_port shall perform the Online-to-Offline Protocol to enter the Offline State from the Active State. After transmitting OLS continuously for at least 5ms, the N_port will be offline and then it may power down or turn off its transmitter. The process of Online-to-Offline Protocol is shown as Fig.4 (d).

In fact, by setting the “init” in Fig.2 (c) to be anyone of the nine states (AC~OL3), we can analyze the switch processes through the animation or ODB information in OPNET and find an interesting phenomenon: Whichever state an N_port may be, if it sends the Primitive Sequences and Primitive Signals just received unchangeably, this N_port can switch to the Active State automatically after at most six states. This rule can be easily proved by observing the processes of four Primitive Sequence Protocols in Fig.4. According to this rule, an N_port is able to be activated itself by just responding the opposite N_port with received sequences or signals without considering its current state. That is, by connecting the transmitter with the receiver logically when power-on, reset or encountering failures, an N_port can restore automatically. In this way, the design of an N_port’s FSM can be simplified greatly.

5 Conclusions
Fiber Channel is a novel and greatly prospective communication technology characterized by high-speed, good-expansibility and high-reliability.
When a port in FC is powered-on, reset or encounters failures, a complex state machine will be performed to recovery from these abnormal conditions. In this paper, the N_port’s FSM is modeled and simulated on OPNET platform, and an important conclusion is drawn by studying the processes of four kinds of Primitive Sequence Protocols. Following this conclusion, the design of an N_port’s FSM can be simplified greatly. In the next, we will study the FSM mechanism of L_port (port in Arbitrated Loop topology) and F_port (port in Fabric topology) in FC.

References: