《연구논문》 Korean Journal of Optics and Photonics, Vol. 27, No. 6, December 2016, pp. 212-217
DOI: https://doi.org/10.3807/KJOP.2016.27.6.212
ISSN : 1225-6285(Print)
ISSN : 2287-321X(Online)

Time-Deterministic Event Processing in Terabit Optical-Circuit-Packet Converged Switching Systems

Bup-Joong Kim[1†], Jeong-dong Ryoo[2], and Kyoungrok Cho[3]

[1]*ETRI, Daejeon, 34129, and Information and Communication Engineering*
[2]*ETRI and Korea University of Science and Technology*
[3]*Information and Communication Engineering, Chungbuk National University*

(Received September 27, 2016; Revised manuscript November 8, 2016; Accepted November 10, 2016)

In connection-oriented data-transport services, data loss can occur when the service experiences a problem on its end-to-end path. To promptly resolve this problem, the data-switching systems providing the service should quickly modify their internal configurations distributed among different places in each system. This is performed through a sequence of problem (event) recognition, sharing, and handling procedures among multiple control processors in the system. This paper proposes a method for event sharing and messaging between control processors, to improve the time determinacy of event processing. This method simplifies runtime event sharing and minimizes the time variability caused by the event data, resulting in a decrease in the latency time in processing global events. The proposed method lessens the latency time of global event processing by about 40%, compared to general methods, for 738 internal path changes.

Keywords: Event processing, Time Determinacy, Terabit Optical-circuit-packet Converged Switching System, Inter-Processor Messaging, Event provisioning
OCIS codes: (060.4259) Networks, packet-switched; (060.4261) Networks, protection and restoration

테라비트 광-회선-패킷 통합 스위칭 시스템에서 시간결정성 높은 이벤트 처리에 관한 연구

김법중[1†] · 류정동[2] · 조경록[3]

[1]한국전자통신연구원
㉾ 34129 대전광역시 유성구 가정로 218

[2]과학기술연합대학원대학교
㉾ 34113 대전광역시 유성구 가정로 217

[3]충북대학교 전자정보대학 정보통신공학부
㉾ 28644 충북 청주시 서원구 충대로 1

(2016년 9월 27일 받음, 2016년 11월 8일 수정본 받음, 2016년 11월 10일 게재 확정)

연결 지향적 데이터 경로 서비스는 운영중인 데이터 경로에 문제가 생겼을 경우, 이의 인지와 변경을 제한된 시간 내에 끝내야 한다. 서비스 중인 데이터 경로에 이상이 있을 경우 계속해서 데이터 유실이 발생하기 때문에, 경로 서비스를 제공하는 데이터 스위칭 시스템은 문제의 경로를 신속하게 정상 경로로 전환해야 한다. 시스템의 신속한 경로 전환을 위해서는 문제 (이벤트)의 인지와 공유와 처리 과정이 신속하게 끝나야 한다. 본 논문에서는 테라비트 광-회선-패킷 통합 스위칭 시스템에서 시간결정성 높은 이벤트 공유를 위한 제어프로세서 사이의 메시징 방법을 제안한다. 제안한 방법은 서비스 실행 중에 이벤트 공유를 단순화하고 이벤트 데이터에 의한 시간 가변성을 최소화하여, 글로벌 이벤트 처리의 지연 시간을 줄이고 시간결정성을 향상시킨다. 경로 이벤트 처리에 제안한 방법을 사용할 때, 738개의 경로 전환에서 일반적인 방법보다 최대 40% 의 지연 시간 감소 효과가 있다.

†E-mail: bjkim74@etri.re.kr
Color versions of one or more of the figures in this paper are available online.

Keywords: 이벤트 처리, 시간 결정성, 테라비트 광-회선-패킷 통합 스위칭 시스템, 제어프로세서 메시징, 이벤트 프로비전닝
OCIS codes: (060.4259) Networks, packet-switched; (060.4261) Networks, protection and restoration

I. 서 론

테라비트 광-회선-패킷 통합 스위칭 시스템은 초당 수 테라비트의 패킷 데이터 전송과 수 십만 개수의 회선 서비스를 제공하기 위해, 수 십 개의 고속 데이터 처리 엔진과 다수의 제어프로세서를 사용한다. 패킷 기반의 데이터 전송은 데이터 처리 엔진이 참조하는 포워딩 테이블의 데이터 경로 정보를 기반으로 한다. 따라서, 데이터 경로 전환과 같은 경로 이벤트를 처리하기 위해서는 해당 경로와 관련 있는 모든 데이터 처리 엔진의 경로 변경 작업이 필요하다. 이 작업은 데이터 처리 엔진의 관리 기능을 수행하는 제어프로세서들의 메시징과 공동 작업에 의해 실현된다.

회선 서비스와 연결 지향적 데이터 경로 서비스는 운영중인 데이터 경로에 문제가 생겼을 경우, 이의 인지와 변경을 제한된 시간 내에 끝내야 한다[1-4]. 특히, 데이터 경로 서비스의 경우 설정경로에 이상이 있는 동안 계속해서 데이터 유실이 발생하기 때문에, 경로복구가 빠를수록 데이터 손실과 서비스 품질의 열화를 최소화할 수 있다. 빠른 경로복구를 위해서는 문제 (이벤트)의 인지와 공유와 처리 과정이 신속하게 끝나야 하는데, 이중에서 이벤트 공유 과정이 상대적으로 복잡하고 제어프로세서의 자원소모가 많고 소프트웨어 처리가 많기 때문에 시간 개선의 여지가 많다.

본 논문에서는 테라비트 광-회선-패킷 통합 스위칭 시스템에서 시간결정성 높은 이벤트 공유를 위한 제어프로세서 사이의 메시징 방법을 제안한다. 제안한 방법을 22개의 제어프로세서가 운영하는 3.2 테라비트 광-회선-패킷 통합 스위칭 시스템에 적용하고, 실험을 통해 그 성능을 검증한다.

II. 테라비트 광-회선-패킷 통합 스위칭 시스템

그림 1은 본 연구에서 사용한 테라비트 광-회선-패킷 통합 스위칭 시스템 (데이터 스위칭 시스템)의 구성을 보인다.

데이터 스위칭 시스템은 크게 두 개의 서브 시스템 - 패킷-회선 스위칭 시스템과 광 스위칭 시스템으로 나눈다. 패킷-회선 스위칭 시스템은 이더넷, MPLS-TP (Multi-protocol Label Switching - Transport Profile), OTN (Optical Transport Networking) 같은 패킷 회선 데이터 처리 기술을 활용하여 데이터를 스위칭 한다. 광 스위칭 시스템은 패킷-회선 스위칭 시스템의 NNI (Network-Network Interface) 측 광 신호들을 파장 다중화시켜 단일 광 선로로 내보낸다. 광 스위칭 시스템은 수신된 광 신호에서 드롭(drop)되는 광 파장과 다른 노드로 전달되는 광 파장을 각각 스위칭 한다. 드롭되는 광 파장은 패킷-회선 스위칭 시스템으로 전달된다.

패킷-회선 스위칭 시스템의 라인카드는 포워딩 엔진을 이용하여 광모듈로 유입된 데이터 트래픽을 목적지 포트로 전달한다. 광 스위칭 시스템의 라인카드는 광 신호 증폭기와

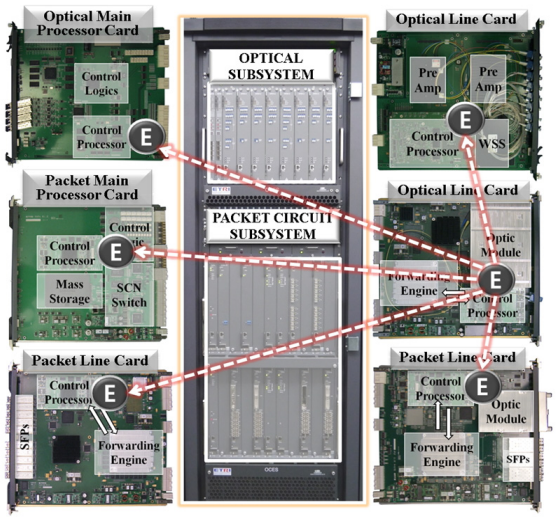

Fig. 1. Inter-control processor messaging for event sharing in a terabit optical-circuit-packet converged switching system.

분산 보상기를 이용하여 광 신호의 품질을 높이고, 파장선택 스위치 (WSS, Wavelength Selective Switch)를 이용하여 광 신호를 스위칭 한다.

시스템의 각 카드는 카드 고유의 기능을 실현하기 위한 카드기능 블록과 이의 제어 및 관리를 위한 제어프로세서를 포함한다. 시스템의 제어프로세서는 크게 두 가지 종류로 구분할 수 있는데 하나는 시스템 운영 관리 기능을 수행하는 메인 프로세서이고, 다른 하나는 각 기능카드 내부의 로컬 한 운영 관리 기능을 수행하는 로컬 프로세서이다.메인 프로세서는 이더넷 스위치로 시스템 제어 네트워크 (SCN, System Control Network)를 구성하여 제어프로세서 사이의 통신을 중계하고, 대용량 저장장치를 이용하여 소프트웨어와 로그 데이터를 저장한다. 메인 프로세서는 로컬 프로세서들을 조직하고 조정하는데, 두 서브 시스템 사이의 조정 기능은 패킷-회선 스위칭 시스템의 메인 프로세서가 한다.

제어프로세서는 자신이 관리하는 영역에서 이벤트를 수집하는데, 이벤트 종류에 따라 자체적으로 이벤트를 처리 하거나, 다른 제어 프로세서와 협업하여 이벤트를 처리한다. 다른 제어 프로세서와 협업이 필요한 글로벌 이벤트의 경우 이벤트 감지, 이벤트 공유, 이벤트 처리 과정을 거치게 된다. 이벤트 감지와 처리는 자신의 영역에서 자신의 스케쥴링으로 처리한다. 반면, 이벤트 공유는 물리적으로 떨어져 있고 다른 스케쥴링으로 돌고 있는 제어프로세서와 메시지를 주고 받아 처리한다. 제어프로세서 사이의 메시지 송수신은 시스템 제어 네트워크를 이용한다.

다음 장에서는 글로벌 이벤트 처리의 시간효율성과 시간결정성을 높이기 위한 제어프로세서 사이의 메시징 방법을 제안한다.

III. 시간 결정성 높은 이벤트 공유 방법

3.1. 일반적 제어프로세서 메시징

그림 2는 제어프로세서 사이에 메시지 혹은 명령을 전달할 때 주로 사용하는 RPC (Remote Procedure Call), RPC-like 방식의 데이터 전달 방법이다 (GIPM, General packet-type Inter-control-Processor Messaging)[5-8]. 송신측 제어프로세서는 이더넷 패킷 (64~1518 bytes)을 매개로 수신측 제어프로세서에 자신의 메시지를 전달한다.

메시지 헤더는 고정크기의 고정 포맷으로 호출 아이디 (Call ID), 함수 (Procedure), 매개 변수 사이즈 (Size of Arguments)를 전달한다. 메시지 페이로드는 매개 변수 (Arguments)를 전달하는데, 그 크기와 포맷은 가변적이다. 메시지 페이로드가 패킷의 페이로드보다 클 경우, 여러 패킷으로 나눠 보낸다.

그림 2의 GIPM에서 사용하는 메시지 패킷의 개수는

$$N_p = 1 + \lfloor (L_a - 1) / L_{maxu} \rfloor, \quad (1)$$

와 같다. 여기서, L_a 는 매개 변수의 전체 사이즈이고, L_{maxu} 은 하나의 패킷이 허용하는 페이로드의 최대 크기이다.

이 방식에서 이벤트 공유 및 처리 시간은

$$T_{total}(L_a) = T_{ep}(L_a) + T_{hdr}(64) + T_{pl}(L_{ep}) \quad (2)$$

와 같다. 여기서, $T_{ep}(L_a)$는 수신측 제어프로세서가 이벤트를 처리하는데 걸리는 시간이고, $T_{hdr}(64)$는 메시지 헤더 (64 바이트)를 송수신하는 데 걸리는 시간이고, $T_{pl}(L_{ep})$는 메시지 페이로드를 송수신하는 데 걸리는 시간이다. $T_{pl}(L_{ep})$에서 L_{ep} 는 $N_p \times L_{nh} + L_a$ 인데, L_{nh} 는 ETHERNET 헤더와 TCP/IP 헤더의 크기이다. $T_{pl}(L_{ep})$ 는

$$T_{pl}(L_{ep}) = (N_p - 1) \times T_{pl}(1518) + T_{pl}(L_{pmod}) \quad (3)$$

와 같다. 여기서, L_{pmod} 는 $L_a \le L_{maxu}$ 일 때 $(L_{nh} + L_a)$ 이고, $L_a > L_{maxu}$ 일 때 $\{L_{nh} + remainder\ of\ (L_a/L_{maxu})\}$ 이다.

위에 보인 것처럼, GIPM 방식에서는 매개 변수의 크기가 커지면, 이벤트 공유 시간도 함께 증가하게 된다. 매개 변수의 크기가 패킷의 페이로드 크기를 벗어나게 되면, 추가로 패킷을 사용하게 되는데, 이는 갑작스런 이벤트 공유 및 처리 시간의 증가로 이어지게 된다.

3.2. 시간 결정성 높은 제어프로세서 메시징

그림 3은 본 연구에서 제안하는 '대용량 데이터 스위칭 시스템에서 시간결정성 높은 제어프로세서 사이의 메시징 방법'(PIPM, Proposed Inter-control-Processor Messaging)을 보인다.

제어프로세서는 이벤트 서비스를 시작하기 전에 상대 제어프로세서들에게 이벤트 프로비전닝 데이터를 보낸다. 이벤트 프로비저닝 데이터를 수신한 제어프로세서는 디코딩 작업으로 이벤트 아이디, 로컬 함수, 매개 변수를 추출하고, 이벤트 테이블에 그 결과를 등록한다. 이 작업이 끝나면 제어프로세

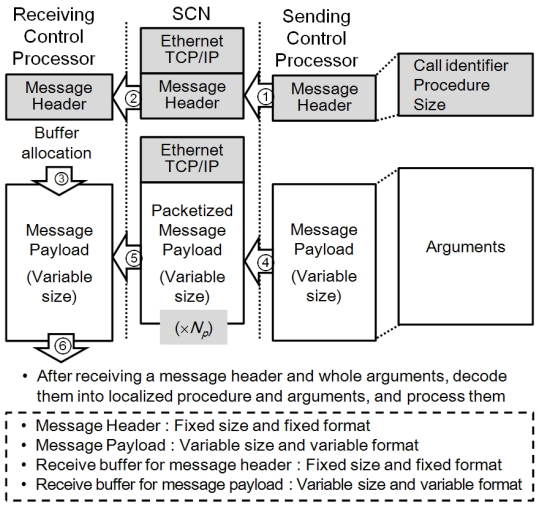

Fig. 2. General packet-type Inter-control-Processor Messaging (GIPM).

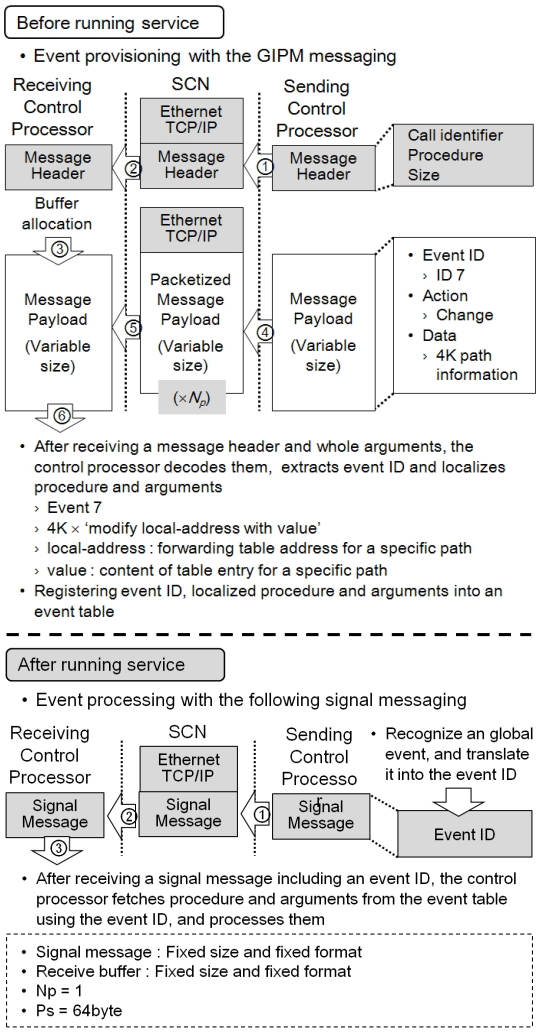

Fig. 3. Proposed Inter-control-Processor Messaging (PIPM).

서는 해당 이벤트 서비스를 시작한다.

이벤트 서비스 운영 중에 제어프로세서가 이벤트를 감지하게 되면, 이벤트에 상응하는 이벤트 아이디를 도출하고, 이를 시그널 메시지에 실어 상대 제어프로세서들에게 전달한다. 수신측 제어프로세서들은 시그널 메시지에서 이벤트 아이디를 확인하고, 이벤트 테이블에서 이벤트 아이디에 해당하는 로컬함수와 매개 변수를 불러와 실행시킨다. 이벤트 아이디는 특정 글로벌 이벤트의 상징으로써, 이벤트가 발생했을 때 수행할 함수와 그의 매개 변수를 지정한다. 이벤트 아이디는 시스템에서 모든 제어프로세서들이 이해할 수 있는 공통의 기호이다. 이벤트 아이디를 싣는 시그널 메시지는 그 크기와 포맷이 모든 이벤트에 대해 동일하다.

시그널 메시지는 최소 크기의 이더넷 패킷 (64 바이트)을 이용한다. 시그널 과정에서 사용하는 메시지 패킷의 수는 항상 1 이다. 즉, 이벤트 시그널을 송신하는 제어프로세서는 단지 하나의 메시지 패킷만을 보낸다.

이 PIPM 방식에서, 이벤트를 공유하고 처리하는데 걸리는 시간은

$$T_{total}(L_a) = T_{ep}(L_a) + \{ T_{tx}(64) + T_{rx}(64) \} \quad (4)$$

와 같다. 여기서, $T_{ep}(L_a)$ 는 이벤트를 처리하는데 걸리는 시간이고, $\{ T_{tx}(64) + T_{rx}(64) \}$ 는 이벤트를 공유하는데 걸리는 시간이다. 이는 이벤트 공유에 필요한 시간이 매개 변수 (이벤트 데이터)의 크기와 무관함을 보여준다. PIPM 에서 이벤트 데이터의 크기는 이벤트 처리 시간 $T_{ep}(L_a)$ 에만 영향을 준다.

시그널 메시지를 수신한 제어프로세서는 자신의 스케쥴링으로 자신의 영역 안에서 이벤트 처리를 수행한다. 이벤트 처리는 이벤트 공유보다 작업의 연속성을 유지하기가 용이하다. 이벤트 공유는 물리적으로 분리되어 있고 서로 다른 스케쥴링으로 돌고 있는 제어프로세서들이 메시지를 주고받아 처리한다. 메시지의 송수신은 제어프로세서에게 비동기적인 인터럽트를 발생시킨다. 인터럽트가 발생하면 프로세서는 코어자원 일부를 재배치하고, 캐쉬 데이터 일부 혹은 전부를 무효화 한다. 이는 제어프로세서에서 수행 중이던 시스템 태스크에 일시적인 인터럽트와 멈춤을 유발하게 된다. 전체 이벤트 처리 과정에 있어 이벤트 공유는 이벤트 처리보다 시간결정성에 더 큰 영향을 주게 된다.

본 연구의 PIPM 방식은 이벤트 서비스 운영 중에 이벤트가 발생했을 때, 이벤트 데이터의 이동과 버퍼링의 부담을 최소화하고, 이벤트 메시징과 관련된 인터럽트 수를 줄이고, 이벤트 공유 과정을 간략화 한다. PIPM 은 이벤트 공유 과정에서 이벤트 데이터의 영향을 최소화함으로써, 이벤트 처리의 시간효율성과 시간결정성을 높이고, 시스템의 이벤트 처리 성능을 향상시킨다.

IV. 실 험

그림 4는 대용량 데이터 스위칭 시스템에서 복수의 제어프로세서가 관여된 글로벌 이벤트 발생과 그 처리의 예를 보인다. 실험에서는 이전 장에서 보인 두 개의 제어프로세서 메시징을 그림 4의 이벤트 처리에 적용하고 그 성능을 측정하였다.

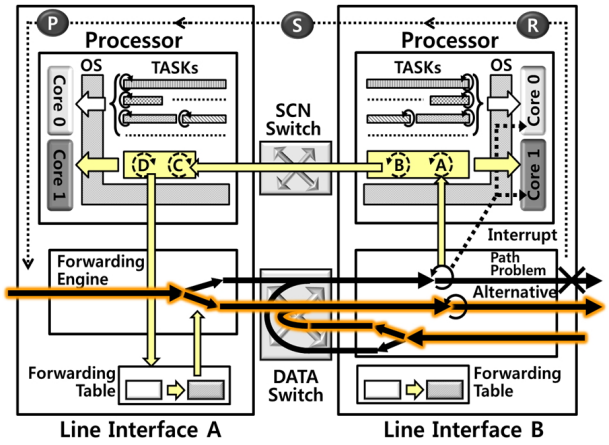

Fig. 4. Event recognition, sharing, and handling in a data switching system.

네트워크 패킷이 라인카드 A에 도착하면, 포워딩 엔진은 패킷의 네트워크 아이디 (이더넷 주소, VLAN ID, IP 주소)에 해당하는 경로 레이블을 포워딩 테이블에서 찾는다. 경로 레이블은 패킷의 논리적인 경로를 가리키고, 패킷의 서비스 타입, 서비스 클래스, 데이터 스위치 포트, 출력 포트를 지정한다. 출력 라인카드 B의 포워딩 엔진은 경로 레이블에 따라 패킷을 스케쥴링하고, 경로 레이블에 상응하는 출력 포트로 패킷을 내보낸다.

패킷 경로 각각의 유효성은 CCM (Continuity Check Messages)으로 점검한다[9]. CCM 을 통해 이벤트 감지 태스크 (A)가 경로 문제를 인지하면 (R), 문제의 경로를 이벤트 메시징 태스크 (B)에게 알린다. 이벤트 메시징 태스크 (B)는 시스템 제어 네트워크를 통해 상대 제어프로세서에 경로 변경 메시지를 송신 한다 (S). 경로 변경 이벤트 메시지를 수신한 이벤트 메시지 태스크 (C)는 경로 변경을 이벤트 처리 태스크 (D)에게 알린다. 이벤트 처리 태스크 (D)는 포워딩 테이블 내용을 변경하여 문제의 경로를 대체 경로로 바꾼다 (P).

여러 개의 경로를 포함하는 출력 포트에 문제가 발생하면, 인터럽트와 인터럽트 서비스 루틴을 통해 이벤트 감지 태스크 (A)가 문제의 포트 정보를 인지한다. 이벤트 감지 태스크 (A)는 문제의 포트 정보를 이벤트 메시징 태스크 (B)에 알려 상대 제어프로세서에 공지하도록 한다.

여러 출력 포트와 경로를 포함하는 출력 라인카드 B에 문제가 발생하면, 메인 프로세서의 이벤트 감지 태스크 (A)가 라인카드 상태 신호를 통해 문제를 인지하고, 이벤트 메시징 태스크 (B)에게 알린다. 이벤트 메시징 태스크 (B)는 문제의 카드 정보를 제어프로세서들에게 알린다.

위와 같이, 경로 변경이 필요한 이벤트 발생 시, 제어프로세서들은 이벤트 메시징을 통해 문제의 경로 정보를 공유하고, 각자의 영역에서 문제 경로를 대체 경로로 변경하기 위

한 로컬 작업을 수행한다.

PIPM 에서는 포트 문제와 카드 문제를 이벤트 아이디로 치환하고 각 카드의 이벤트 테이블에 등록한다. 이벤트 아이디는 포트 문제와 카드 문제를 처리하는 과정에서 제어프로세서 사이의 공통 기호로 사용된다. 이벤트 아이디와 함께, 문제가 발생했을 때 실행시킬 로컬 함수와 매개 변수를 이벤트 테이블에 등록한다. 특정 포트에 새로운 경로 서비스를 추가하기 위해서는 경로 서비스 시행 전에 이벤트 테이블에 이벤트 아이디, 로컬 함수, 매개 변수를 등록 혹은 갱신한다 (이벤트 프로비전닝). 추가되는 새로운 경로 서비스는 이벤트 프로비저닝 후에 시작된다.

포트에 문제가 발생하면, 상응하는 인터럽트가 활성화되어 인터럽트 서비스 루틴이 실행된다. 인터럽트 서비스 루틴은 포트 문제를 인지하고 이벤트 감지 태스크 (A)에게 알린다 (R). 이벤트 감지 태스크 (A)는 포트 문제를 이벤트 아이디로 변환하고 이벤트 메시징 태스크 (B)에게 알린다. 이벤트 메시징 태스크 (B)는 시그널 메시지에 이벤트 아이디를 실어 관련 제어프로세서들에게 보낸다 (S). 시그널 메시지를 수신한 이벤트 메시지 태스크 (C)는 이벤트 아이디를 이벤트 처리 태스크 (D)에게 알린다. 이벤트 처리 태스크 (D)는 이벤트 테이블에서 이벤트 아이디에 해당하는 로컬 함수와 매개 변수를 불러와 실행시킨다 (P). 이를 통해, 문제의 포트를 사용하는 경로들을 대체 경로로 변경한다.

GIPM 에서는 이벤트 메시징 태스크 (B)가 변경이 필요한 모든 경로 정보를 이벤트 메시지에 실어 상대 제어프로세서들에게 보낸다.

다음 단락은 PIPM 과 GIPM 을 시스템에 적용하고, 실험한 결과이다.

동일한 경로 변경 이벤트에 대해 처리 방식이 다른 이벤트 케이스를 정의하고, 각각의 이벤트 케이스 별로 PIPM 과 GIPM 의 이벤트 처리 성능을 측정하였다. 경로 변경 이벤트의 처리는 라인카드의 포워딩 테이블을 변경함으로써 가능하다. 포워딩 테이블은 제어프로세서의 메모리 그리고 PCI express 혹은 로컬 버스를 통해 접근 가능한 포워딩 디바이스에 위치한다. 실험에서는 포워딩 테이블의 위치와 접근 방식에 따라 3개의 이벤트 케이스를 정의하였다. 각각의 이벤트 케이스 별로 경로 정보를 포함하는 이벤트 데이터 크기 (경로 개수에 비례)를 다르게 하여 이벤트 공유와 처리에 소요되는 시간을 측정하였다. 이벤트 케이스 1 (Event 1)은 포워딩 테이블이 제어프로세서의 메모리에 있고, 그 내용을 변경함으로써 문제의 경로를 대체 경로로 변경하는 경우이다. 이벤트 케이스 2 (Event 2)는 PCI express 디바이스의 포워딩 테이블을 변경하여 경로를 변경하는 경우이다. 이벤트 케이스 3 (Event 3)은 로컬 버스 디바이스의 포워딩 테이블을 변경하여 경로를 변경하는 경우이다.

표 1은 PIPM 의 경로 변경 이벤트 처리를 위한 이벤트 테이블의 내용이다. PIPM 에서는 위에서 정의한 3가지 이벤트 케이스를 각각의 이벤트 아이디로 치환하고, 이벤트 아이디 별로 이벤트 처리에 필요한 로컬 함수와 매개 변수를 이벤트 테이블에 등록하였다 (이벤트 프로비전닝). 제어프로세서가 이벤트 테이블에 등록된 이벤트 아이디 (id)를 수신하면 (시그널 메시징), 이벤트 아이디에 해당하는 로컬 함수와 매개 변수를 이벤트 테이블에서 불러와 이벤트 처리 작업을 수행하였다.

이벤트 공유와 처리 시간 측정을 위해, IXIA 계측기[10]를 시스템 제어 네트워크에 연결하고, UDP 패킷에 이벤트 메시지를 실어 제어프로세서로 송신하였다. 제어프로세서는 이벤트 메시지 수신과 처리를 끝낸 후, 처리 완료 메시지를 IXIA 계측기로 보냈다. IXIA 계측기는 이벤트 메시지 송신에서 처리 완료 메시지 수신까지 걸린 시간 (지연 시간)을 측정하였다. 시스템에서 실험에 사용한 제어프로세서는 1.2 GHz 듀얼코어 RISC 프로세서이었고, 제어프로세서의 OS는 리눅스 (커널 2.6.35)이었다. 제어프로세서는 이벤트 메시지 수신과 처리 완료 메시지 송신을 위해 1 Gbps 이더넷 포트 하나를 사용하였다.

그림 5에 각 이벤트에 대한 GIPM 과 PIPM 의 지연 시간 측정 결과를 보인다. GIPM과 PIPM 모두 이벤트 데이터 (경로 개수)가 증가하면 지연 시간도 함께 증가하였는데, PIPM 이 GIPM 보다 그 증가 폭이 작았다. 특히, 경로 개수가 증가하여 이벤트 데이터가 UDP 패킷의 페이로드 크기 (1,472 바이트) 보다 커질 경우 (1,476 바이트), PIPM 의 지연 시간 증가는 미미한데 비해, GIPM 의 지연 시간 증가는 4 바이트 데이터 (경로 2개) 증가에 큰 폭의 변화를 보였다.

그림 6의 그래프는 이벤트 데이터 증가에 따른 PIPM 의

Table 1. Event table and measurement environment

id[#1]	Local prodedure[#2]	Local arguments[#3]	Measurement environment
1	modify mem data	address, data	mem[#4] : 800 Mbps, 64 bit
2	modify PCIE data	address, data	PCIE[#5] : 2.5 Gbps, 1 lane
3	modify LB data	address, data	LB[#6] : 37.5 MHz, 16 bit

#1: Event ID

#2: Event action to carry out target operation

#3: Auxiliary event data for event handling – address and data for target device

#4: DDR3 SDRAM memory

#5: PCI express device

#6: 16 bit local bus device

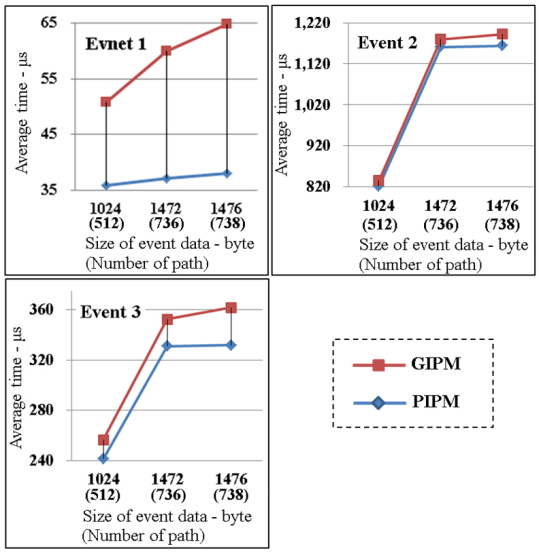

Fig. 5. Event messaging and handling time.

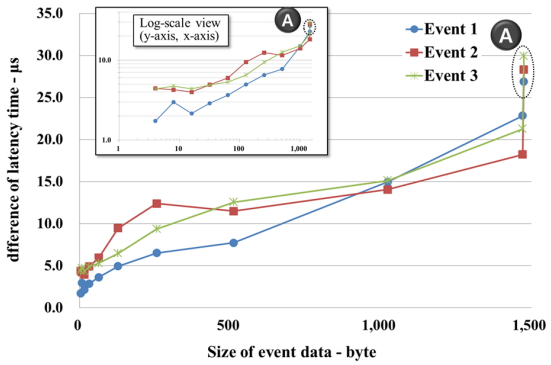

Fig. 6. Increasing rate of event messaging and handling time.

지연 시간과 GIPM 의 지연 시간 차이를 보여준다.

모든 이벤트 케이스에 대해 PIPM 의 지연 시간은 GIPM 의 지연 시간보다 적은 결과를 보였다. 이벤트 데이터의 증가에 따라 지연 시간 차이가 점차 증가되는 추세를 보이다가, 패킷을 추가로 사용해야 하는 지점 (A)에서 공통되게 갑작스런 증가 패턴을 보였다. 이는 패킷 하나만을 사용하여 이벤트를 공유하는 PIPM 과 다르게, GIPM 이 경로 증가에 따라 패킷을 추가로 사용하여 이벤트를 공유했기 때문이다. 이벤트 공유 과정에서 PIPM 은 경로 개수와 상관없이 고정 크기의 패킷 하나만을 송수신지만, GIPM 은 경로 개수가 증가하면 패킷의 크기를 키워야 하고, 여기에 더해 경로 개수가 패킷이 담을 수 있는 범위를 벗어나면 추가로 패킷을 송수신 해야 한다. 그 결과 PIPM 의 지연 시간이 GIPM 의 지연시간 보다 적게 나타나고, GIPM 에서 보이는 급작스런 지연 시간 변화 (A)가 PIPM 에서는 나타나지 않게 된다.

IV. 결 론

테라비트 광-회선-패킷 통합 스위칭 시스템에서 시간결정성 높은 이벤트 공유를 위한 제어프로세서 사이의 메시징 방법을 제안 하였다. 제안한 방법을 22개의 제어프로세서가 운영하는 3.2 테라비트 광-회선-패킷 통합 스위칭 시스템에 적용하고, 실험을 통해 그 효과를 검증하였다. 제안한 방법은 서비스 실행 중에 이벤트 공유를 단순화하고 이벤트 데이터에 의한 시간 가변성을 최소화하여, 글로벌 이벤트 처리의 지연 시간을 줄이고 시간결정성을 향상시켰다. 시스템의 경로 이벤트 처리에 제안한 방법을 사용했을 때, 738개의 경로 전환에서 GIPM 보다 최대 40% 의 지연 시간 감소 효과가 있었고, 경로 개수 증가에 따른 급작스런 지연 시간 증가가 없었다.

감사의 글

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였습니다. [B0101-16-0024, 차세대 광전달망 구축을 위한 테라급 광-회선-패킷 통합 스위칭 시스템 기술 개발]

References

1. M. Murakami and Y. Koike, “Highly Reliable and Large-Capacity Packet Transport Networks: Technologies, Perspectives, and Standardization,” Journal of Lightwave Technology, vol. 32, no. 4, pp. 805-816, February 2014.
2. J. W. Youn, J. Yu, and T. Yoo, “Performance evaluation of service-aware optical transport system,” ETRI J., vol. 32., no.2, pp. 241-247, 2010.
3. R. Sabella *et al.*, “Flexible packet-optical integration in the cloud age: Challenges and opportunities for network delayering,” IEEE Comm. Mag., vol. 52, no. 1, pp. 35-43, 2014.
4. S. Gringeri and T. Rarick, “Packet and TDM transport integration: how, when and why?” OFC/NFOEC, NWF4, 2006.
5. A. D. Birrell and B. J. Nelson, “Implementing Remote Procedure Calls,” ACM Transactions on Computer Systems, pp. 39-59, Feb. 1984.
6. R. Thurlow, “RPC: Remote Procedure Call Protocol Specification Version 2,” RFC 5531, May 2009.
7. Qureshi K. and Rashid H., “A Performance Evaluation of RPC, Java RMI, MPI, and PVM,” Malaysian Journal of Computer Science, vol. 18, no. 2, pp. 38-44, Dec. 2005.
8. L. Cassano *et al.*, “An inter-processor communication interface for data-flow centric heterogeneous embedded multiprocessor systems,” Design & Technology of Integrated Systems In Nanoscale Era (DTIS), pp. 1-6, May 2014.
9. J. Ryoo *et al.*, “OAM and its performance monitoring mechanisms for carrier Ethernet transport networks,” IEEE Commun. Mag., vol. 46, no. 3, pp. 97-103, Mar. 2008.
10. Ixia, “Ixia Packet Generator,” http://www.ixiacom.com/.