병렬 프로토콜 구현을 위한 다중 프로세스 모델의 설계

최 선 완[†] · 정 광 수[††]

요 약

본 논문은 병렬 프로토콜 구현을 위해서 (1)채널통신 모델, (2)포크-조인 모델, (3)사건조회 모델이라 부르는 3 가지 유형의 다중 프로세스 모델을 제시한다. 각 모델에 대한 병렬화 사양을 위해서 병렬 프로그래밍 언어인 Par. C System을 사용한다. 제안한 다중 프로세스 모델의 성능을 측정하기 위하여 인터넷 프로토콜 스택의 Internet Protocol (IP)을 Transputer상에서 구현한다. IP 프로토콜 기능은 송신측과 수신측으로 분리하고 양측의 병렬화는 Multiple Instruction Single Data (MISD) 구조를 이용한다. 제안한 모델들은 다양한 실행시간 과부하에 대하여 성능 평가와 비교 분석을 한다. 즉, 채널통신 모델에서는 채널을 경유한 사건 송신, 포크-조인 모델에서는 프로세스 생성, 그리고 사건조회 모델에서는 프로세스간 문맥전환시에 발생하는 과부하를 송신측과 수신측에 대하여 성능을 분석한다. 송신측의 성능 측정 결과, 사건조회 모델이 채널통신 모델과 포크-조인 모델과 비교하여 77%와 9%의 빠른 처리 시간을 보였다. 수신측에서는 포크-조인 모델이 채널통신 모델과 사건조회 모델과 비교하여 55%와 107%의 빠른 처리 시간을 보였다.

Design of Multiprocess Models for Parallel Protocol Implementation

Sunwan Choi[†] · Kwangsue Chung[††]

ABSTRACT

This paper presents three multiprocess models for parallel protocol implementation, that is, (1)channel communication model, (2)fork-join model, and (3)event polling model. For the specification of parallelism for each model, a parallel programming language, Par. C System, is used. To measure the performance of multiprocess models, we implemented the Internet Protocol Suite(IPS) Internet Protocol (IP) for each model by writing the parallel language on the Transputer. After decomposing the IP functions into two parts, that is, the sending side and the receiving side, the parallelism in both sides is exploited in the form of Multiple Instruction Single Data (MISD). Three models are evaluated and compared on the basis of various run-time overheads, such as an event sending via channels in the parallel channel communication model, process creating in the fork-join model and context switching in the event polling model, at the sending side and the receiving side. The event polling model has lower processing delays as about 77% and 9% in comparison with the channel communication model and the fork-join model at the sending side, respectively. At the receiving side, the fork-join model has lower processing delays as about 55% and 107% in comparison with the channel communication model and the event polling model, respectively.

※본 논문은 96년도 안양대학교 학술연구비 지원을 받아 연구되었음.

† 정 회 원:안양대학교 정보통신공학과
†† 정 회 원:광운대학교 전자공학부

논문접수:1997년 5월 10일, 심사완료:1997년 9월 18일

1. 서 론

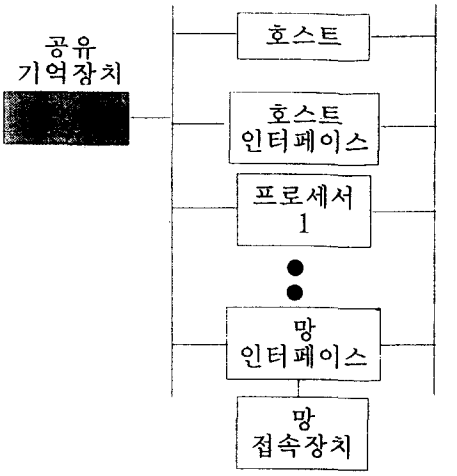
ATM과 같은 고속통신기술의 발전에 따라 통신에서의 전통적인 병목지점이 사라질 예정이다. 그 결과 통신 프로토콜의 처리 속도가 성능향상에 제약 요소가 되고 있다. 이를 해결하기 위해서 병렬 처리를 이용한 다중 프로세서 구조가 다양하게 설계되고 구현되어 왔다 [1-16]. 다중 프로세서상에서 병렬화를 추구하는 것은 성능 향상과 신뢰성 구현에 좋은 방법이다.

다중 프로세서 구조에서, 별도의 프로세스들 또는 프로토콜 기능들이 병렬로 수행된다. 그러나 병렬화 단위는 프로세서간 동기화, 프로세서간 통신, 스케쥴링 과부하 뿐만 아니라 프로세서의 수에 의해서 제약된다. 현재 8개의 프로세서가 병렬 프로토콜 구현에 사용되고 있다. 그 결과, 대부분의 경우에 여러 프로토콜 기능들은 다중 프로세서 플랫폼상에 한개의 프로세서에서 여러 프로세스들을 병렬로 구현하여 수행시키고 있다. 그러나 다중 프로세스의 사용은 프로세스 생성, 프로세스 사이에 접근제어, 문맥전환(context switching), 통신 과부하와 같은 이유로 실행시간(run-time) 과부하를 발생시킨다 [17]. 이러한 실행시간 과부하는 고성능 프로토콜 구현을 위한 병렬 시스템에 치명적이다. 그 결과 효과적인 다중 프로세스 모델을 찾는 것이 병렬 프로토콜 구현시에 성능을 향상 시키는데 매우 중요하다. Schmidt와 Suda [18]는 프로세스 구조를 프로토콜 스택으로 대응시켰다.

본 논문에서는 병렬 프로토콜 구현을 위한 다중 프로세스 모델로서 3가지 유형을 제시하고 실행시간 과부하를 측정하여 성능을 평가한다. 이들 모델의 성능을 측정하기 위해서 IP (Internet Protocol) [19]을 Transputer상에서 구현하고 병렬 언어로서 Par.C System [20]을 사용한다.

2. 다중 프로세스 구조

기억장치 할당 방법에 따라 다중 프로세서 시스템들은 공유 기억장치 시스템과 분산 기억장치 시스템으로 분류한다. 고속통신 구조를 위해서는 기억장치 복제를 제거한 공유 기억장치 시스템이 분산 기억장치 시스템보다 효과적이다. (그림 1)은 통신 시스템에 대한 전형적인 다중 프로세서 구조로서 프로세서, 연결(link), 기억장치, 망 접속장치로 구성된다. 각 프로세서는 프로토콜 기능들과 인터페이스, 기억장치 관리, 시계와 같은 기능을 수행한다.



(그림 1) 통신 시스템을 위한 다중 프로세서 구조
(Fig. 1) Typical multiprocessor architecture for communication systems

다중 프로세서 구조에서 조차도 이용 가능한 프로세서의 부족으로 인하여 한 개의 프로세서에서 프로토콜 기능들을 병렬로 수행하기를 요구한다. 따라서 다중 프로세서 환경에서 통신 시스템에 대한 효과적인 다중 프로세스 모델을 개발하는 것이 매우 중요하다. 다중 프로세스 기법을 이용하여 통신 프로토콜을 구현하기 위해서는 먼저 통신 프로토콜에서 병렬처리 모델을 찾는 것이 중요하다.

2.1 통신 프로토콜에서의 병렬화

병렬처리는 이미 컴퓨터 구조, 수치해석등의 다른 과학 분야에서 널리 사용되어 왔으며 이를 통신 프로토콜에 적용할때도 기존 방법의 영역을 벗어나지는 않을 것이다. 지금까지 병렬 프로토콜 구현에 적용한 다중 프로세서 구조는 다음 3가지로 분류할 수 있다.

첫번째는 일반적으로 병렬처리 모델을 기술할때 표준으로 사용되는 Flynn 모델 [21]에 적용한 경우이다.

- Single Instruction Single Data (SISD) 구조 : 병렬처

리 모델이 아님.

- Single Instruction Multiple Data (SIMD) 구조:[2][11]
- Multiple Instruction Single Data (MISD) 구조:[3][14]
- Multiple Instruction Multiple Data (MIMD) 구조:[1][4][5]

두번째로, 통신 프로토콜을 수행할 때 프로세서 또는 프로세스가 처리하는 단위에 따라 분류한 경우이다.

- 계층 단위:한개의 계층을 한개의 프로세서에 할당하는 방법 [5][9][10]
- 계층내의 함수 단위:한 프로토콜내에 함수들에 대하여 각기 다른 프로세서에 할당하는 방법 [1][3][7][8].
- 접속 단위:다른 데이터 접속들을 다른 프로세서에 할당하는 방법 [16].
- Protocol Data Unit(PDU) 또는 패킷 단위:패킷마다 다른 프로세서에 할당하는 방법 [2][11].

세번째로, 다중 프로세서에 작업을 할당하는 형태에 따라 수직구조(또는 시간구조) [6][12]와 수평구조(또는 공간구조) [3][14]로 분류하는 방법이다. 수직구조는 기존 통신 프로토콜 스택의 계층 구조와 같이 상·하위 계층들 사이에 순서 제약을 갖는 모델이며, 수평구조는 모든 프로세서들 사이에 순서 제약없이 동시에 수행되는 모델이다.

그러나, 이들 세가지 분류법은 기존의 병렬처리 모델에 기초를 둔 것이므로 그 구조는 같다고 할 수 있고 이들 병렬처리 모델을 다중 프로세스 환경에서 적용하기 위해서 적합한 모델을 찾아야 한다. 〈표 1〉은 이를 정리한 내용이다.

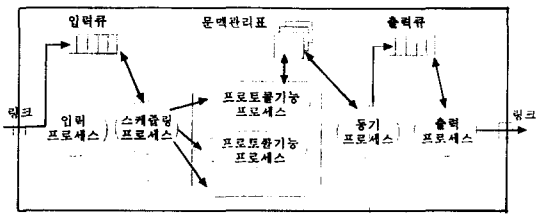
〈표 1〉 병렬 프로토콜 처리 유형.
〈Table 1〉 Type of parallel protocol processing

유형	병렬 단위	기타
SIMD	패킷	공간 병렬
MISD	기능(function) 또는 계층	공간 병렬
파이프라인	기능 또는 계층	시간 병렬
복합	패킷, 기능, 계층	복합

2.2 다중 프로세스 모델

2.2.1 다중 프로세스 구조

일반적으로 한개의 프로세서를 갖는 호스트에서 통신 소프트웨어를 처리하는 방법은 통신 관련 사건이 도착하였는가를 계속 검사하고, 사건이 도착하면 관련 통신 프로토콜을 한개의 프로세스 형태로 생성시키는 방법이다. 만일 특정 통신 프로토콜에 대하여 병렬화를 추구하기 위하여 다중 프로세스 모델에 적용할 때 병렬화가 가능한 각 기능은 별도의 프로세스에 할당되며 (그림 2)와 같은 구조를 갖는다.



(그림 2) 통신 프로토콜을 위한 다중 프로세스 모델
(Fig. 2) A multiprocess model for communication protocol

- 입력큐(*input queue, IQ*):인접 프로세서로 부터 도착한 데이터를 저장하는 공유 기억장치이다.
- 출력큐(*output queue, OQ*):인접 프로세서에게 전달할 데이터를 저장하고 있는 공유 기억장치이다.
- 문맥관리표(*context management table, CMT*):각 프로토콜기능 프로세스들이 사용하는 정보를 저장하고 있는 공유 기억장치이다.
- 입력 프로세스(*input process, IP*):인접 프로세서로 부터 도착한 데이터를 입력큐에 저장하는 프로세스이다.
- 프로토콜기능 프로세스(*protocol function process, PFP*):프로토콜의 특정 기능을 수행하기 위해 생성된 프로세스이다.
- 동기 프로세스(*synchronization process*):프로세스들간의 동기화 기능을 제공한다. 도착된 데이터를 모든 프로세스에게 동시에 전달하기 위한 프로세스인 스케줄링 프로세스(*scheduling process, SP*)와 각 프로세스의 수행 결과를 검사하기 위한 동기 프로세스(*SYP*)로 구분된다.

- 출력 프로세스(*output process, OP*):인접 프로세서에게 결과를 전송하는 프로세스이다.

2.2.2 사건처리 과정

한 사건이 도착하면, 한 프로세서는 다음과 같은 5 단계의 과정으로 진행된다.

1)1단계

입력 프로세스(IP)는 인접 프로세서로 부터 사건을 기다린다. 그 프로세서로 부터 특정 사건(event)이 도착하면 입력 프로세스는 입력큐(IQ)에 즉시 저장한다.

2)2단계

MISD 구조에서는 입력큐에 도착한 사건은 프로토콜기능 프로세스(PFP)들에게 동시에 전달된다. 그러나 SIMD 구조에서는 그 사건은 라운드-로빈 스케쥴링 정책에 의해서 한개의 프로토콜기능 프로세스에게 전달된다. 입력 프로세스와 프로토콜기능 프로세스 사이에 사건을 전달하는 방법으로 다음 3 가지가 있다.

- 통신 채널 방법:스케쥴링 프로세스가 도착한 사건을 특정 채널을 통해서 전달하는 방법이다.
- 인자 전달 방법:스케쥴링 프로세스가 도착한 사건을 부함수 호출과 같이 인자(argument)로써 프로토콜기능 프로세스에게 전달하는 방법이다.
- 사건 조회 방법:스케쥴링 프로세스는 존재하지 않으며, 프로토콜기능 프로세스가 입력큐에 사건이 도착했는가를 계속 조회하는 방법이다. 즉, 프로토콜기능 프로세스는 사건이 도착하자마자 입력큐로 부터 사건을 읽고 각 프로토콜 기능을 수행한다.

3)3단계

프로토콜기능 프로세스에 의해서 프로토콜기능을 수행하는 단계이며 다음 두가지 방법이 가능하다.

- 프로세서가 처음에 초기화 될 때 모든 프로토콜기능 프로세스들을 생성한다. 각 프로토콜기능 프로세스는 동기 프로세스로부터 사건을 전달받아서 관련 프로토콜을 수행하거나 사건 도착 여부를 스스로 검사하여 프로토콜을 수행한다.
- 사건이 도착할 때마다 프로토콜기능 프로세스들이 생성되며 프로토콜 기능을 끝마치면 자동 소멸된다.

4)4단계

동기 프로세스는 프로토콜기능 프로세스로 부터 출력큐에 저장한 결과를 검사하고 그 결과를 다음 프로세서에게 전달할 것인가를 결정한다.

5)5단계

그 결과는 출력 프로세스에 의해서 다음 프로세서에게 전달된다.

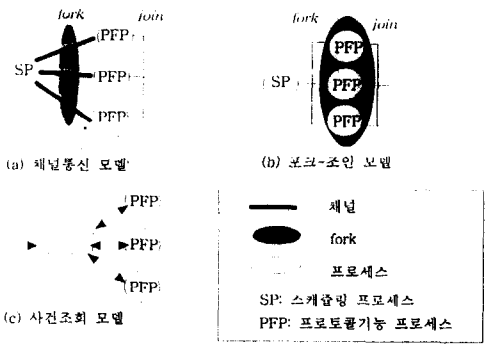
2.3 프로세스 구조 모델

2단계와 3단계 과정에서 〈표 2〉와 같이 6 가지의 다중 프로세스 모델이 가능하다. 그러나, 이들 모델이 모두 적용 가능한 것은 아니다. 예를 들면, 프로세서가 초기화될 때 프로토콜기능 프로세스가 생성되지 않는다면, 입력큐에 도착한 사건은 특정 채널을 통해서 프로토콜기능 프로세스에게 전달될 수 없다. 〈표 2〉에서 O로 표시된 모델만이 다중 프로세스 구조에서 실현 가능한 경우이다.

〈표 2〉 통신 시스템에 대한 다중 프로세스 모델
〈Table 2〉 Multiprocess models for communication systems

모델 번호	사건 전달 방법	프로토콜기능 프로세스 생성 시기	적용 가능 여부
1	채널	프로세서 초기화시	O
2	채널	사건 도착시	X
3	인자 전달	프로세서 초기화시	X
4	인자 전달	사건 도착시	O
5	사건 조회	프로세서 초기화시	O
6	사건 조회	사건 도착시	X

본 연구에서는 이들 세가지 모델을 구현하고 성능을 분석하며, 모델 번호 1을 "채널통신 모델(channel communication model)", 모델 번호 4를 "포크-조인 모델(fork-join model)", 모델 5를 "사건조회 모델(event polling model)"이라 명한다. (그림 3)은 각 구현 모델에 대한 다중 프로세스 구조이다.



(그림 3) 3가지 유형의 다중 프로세스 모델
(Fig. 3) Three types of multiprocess model

3. IP의 병렬화

본 논문에서는 인터넷 프로토콜 스택중에서 Internet Protocol(IP) [19]을 다중 프로세스 모델에 적용한다. IP 기능들은 송신측과 수신측으로 나뉘어지고 각각은 별도의 타스크로 동작한다. 송신측에서, IP 데이터그램구성(datagram consctruction) 기능과 라우팅(routing) 기능은 동일 데이터를 이용한 병렬 처리가 가능하다. 반면에 데이터그램전달(forward) 기능은 IP 데이터그램구성 기능과 라우팅 기능의 결과를 이용하기 때문에 파이프라인 구조로 구성한다. 수신측에서는 라우팅 기능, IP 데이터그램 헤더처리(header analysis) 기능, 생존시간계산(TTL computation) 기능, 헤더체섬(header checksum) 기능들이 동일 데이터를 이용하여 동시에 수행된다. 반면에 재결합(reassemlbly)과 IP 데이터그램분해(datagram decomposition) 기능은 파이프라인 구조로 구성되어 다른 기능들의 결과를 이용해야 한다.

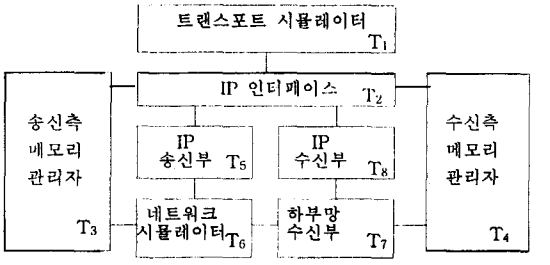
4. 구 현

4.1 구현 환경

다중 프로세서 플랫폼상에서 통신 프로토콜의 고속화를 위해서는 (1) 병렬 언어, (2) 효과적인 문맥 전환과 프로세스 생성 방법, (3) 효율적인 다중 프로세서 위상을 지원해야 한다. 본 논문에서는 다양한 병렬 언어를 지원하고, 문맥 전환과 프로세스 생성 시간이 1μs 보다 작은 Transputer[22] 상에서 IP를 구현하였다. Transputer는 임의의 다중 프로세서 구조를 생성시킬 수 있다. 사용된 Transputer는 Parsytec SuperCluster [23, 24] 이며, 사용된 프로그래밍 환경은 Par. C System [20] 이다. Par.C System은 Helios 병렬 운영체제 [25] 상에서 수행되며 병렬 언어와 임의의 다중 프로세서 망을 구성할 수 있다. Par.C System은 C 언어의 확장으로 다음과 같은 병렬 처리 환경을 제공한다. "Par" 문은 다중 프로세스를 동시에 생성하고, "select"와 "guard"/"guarlink" 문은 가드 명령을 지원한다. "SendLink"(또는 "RecvLink") 문은 Transputer들 사이에 데이터를 송신(또는 수신)한다. 프로세스간 데이터 전송은 "_In"과 "_Out" 문을 이용한다. "Channel" 자료 유형은 통신 경로를 제공한다.

4.2 구현 구조

Transputer 네트워크상에서 송신측과 수신측 양측에 3가지 유형의 다중 프로세스 모델을 평가하기 위해서 IP 기능을 구현하였다. (그림 4)는 8개의 Transputer를 사용한 구현 구조이다. Tn은 Transputer 네트워크상에 n번째의 노드를 나타낸다. IP에 대한 프로토콜 기능들이 송신측 Transputer와 수신측 Transputer에서 다중 프로세스로 수행된다. 다른 Transputer들은 시험 자료 생성, 계층간 인터페이스, 하부망 시뮬레이션, 기억장치 관리와 같은 시험 환경을 위해서 사용된다.



(그림 4) Transputer상의 구현 구조
(Fig. 4) Implementation structure on the Transputer

(그림 4)에서 트랜스포트 시뮬레이터에 대한 Transputer T1은 IP를 동작시키기 위한 시험명령 모듈을 수행한다. 트랜스포트 시뮬레이터는 IP 명령을 T2 (인

터페이스)로 보내고 순수 데이터는 T3 (송신측 메모리 관리자)로 보낸다. T3와 T4 (수신측 메모리 관리자)는 Transputer 간에 데이터 복제를 피하기 위해서 전역(global) 기억장치로 사용한다. T5 (IP 송신부)는 송신측의 IP 기능을 수행하고 T6 (IP 수신부)는 수신측 IP 기능을 수행한다. 구현한 Transputer 네트워크는 직접 망을 제어할 수 없기 때문에 국부망처럼 동작하도록 송신측에 네트워크 시뮬레이터를 T6에 구현하였고 수신측은 하부망 수신부 (T7)가 그 기능을 수행한다.

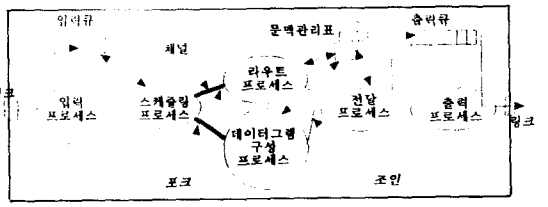
4.3 송신측의 다중 프로세스 구현

4.2장에서 전체적인 구현 구조를 기술하였다. 본 장에서는 다중 프로세스 모델에 따른 IP의 구현을 다룬다. 앞에서 기술한 바와 같이 IP는 송신측과 수신측으로 분리한다. 각각의 프로토콜 기능들은 MISD 구조에 적용한다. MISD 구조에서 모든 프로토콜 기능들은 통일 IP 데이터그램을 이용하여 동시에 수행한다. 그러나 어떤 기능들은 여전히 파이프라인 구조에서 순차적으로 처리된다. 본 장에서는 (그림 4)의 Transputer T5 상에 구현된 IP의 송신측에 대한 3가지 다중 프로세스 모델을 기술한다.

4.3.1 채널통신 모델

(그림 5)는 MISD 구조에 기반한 IP의 송신측에 대한 채널 통신 모델을 보여준다. Transputer 네트워크가 초기화될 때 입력 프로세스, 스케줄링 프로세스, 전달(Forward) 프로세스, 데이터그램구성 프로세스, 라우트 프로세스를 모두 생성한다.

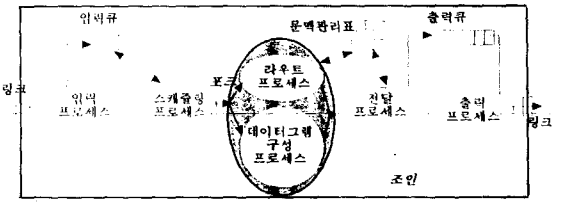
사건이 도착하면 입력 프로세스는 그 사건을 입력큐에 저장하고 다른 사건을 기다린다. 스케줄링 프로세스는 입력큐로 부터 사건을 별도의 채널을 통해서 데이터그램구성 프로세스와 라우트 프로세스로 동시에 보낸다. 두 프로세스는 자신의 프로토콜 기능을 수행하고, 그 결과를 문맥관리표에 저장한다. 전달 프로세스는 동기화 기능을 담당한다. 출력 프로세스는 그 결과를 출력큐로 부터 다른 Transputer에게 전달한다.



(그림 5) 송신측 IP의 채널통신 모델
(Fig. 5) Channel communication model for the sending side of IP

4.3.2 포크-조인 모델

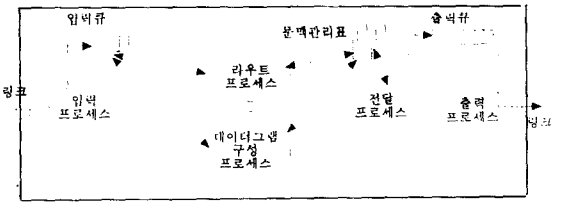
(그림 6)은 송신측 IP에 대한 MISD 구조에서의 포크-조인 모델에 대한 다중 프로세스 모델이다. 데이터그램구성 프로세스와 라우트 프로세스는 Transputer 네트워크가 초기화될 때에는 생성되지 않고, 사건이 입력큐에 도착할 때마다 스케줄링 프로세스에 의해서 새롭게 생성된다. 이때 두 프로세스에게 도착 사건의 전달은 call-by-reference 방법을 이용한다.



(그림 6) 송신측 IP의 포크-조인 모델
(Fig. 6) Fork-join model for the sending side of IP

4.3.3 사건조회 모델

(그림 7)은 송신측 IP의 사건조회 모델에 대한 MISD 구현을 보여준다. Transputer 네트워크가 초기화될 때 모든 프로세스들을 생성하며, 스케줄링 프로세스

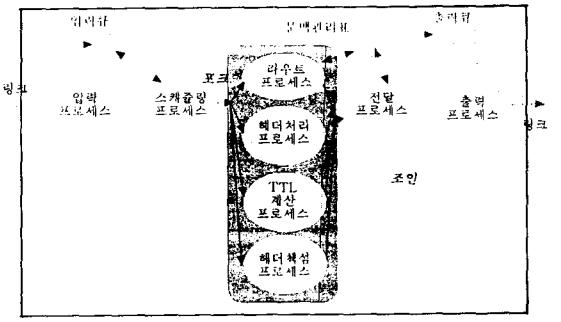


(그림 7) 송신측 IP의 사건조회 모델
(Fig. 7) Event polling model for the sending side of IP

는 필요없다. 즉, 데이터그램구성 프로세스와 라우트 프로세스가 입력큐를 직접 검사함으로써 사건 도착 여부를 판단한다.

4.4 수신측 다중 프로세스 구현

(그림 8)은 수신측 IP의 포크-조인 모델에 대한 다중 프로세스 구현 구조이다. 본 논문에서 수신측에 대한 3가지 다중 프로세스 구현은 송신측과 그 구현 구조가 같기 때문에 생략한다. 단지 프로토콜 기능 자체적인 것과 프로토콜 기능들의 수가 증가한 것이 송신측과 다른 점이다. 수신측 IP는 라우트 프로세스, 헤더처리 프로세스, TTL계산 프로세스, 헤더책섬 프로세스를 동시에 수행한다. 이 프로세스들은 동일한 IP 헤더를 이용한다.



(그림 8) 수신측 IP의 포크-조인 모델
(Fig. 8) Fork-join model for the receiving side of IP

5. 성능 평가

성능 측정은 시스템 호출 명령인 clock( )을 이용하였다. clock( ) 함수는 이 함수가 호출된 싯점부터 현재까지 진행된 프로세서의 시간을 알려준다. 즉, clock( )의 결과는 내부 Transputer clock 값이며 각 clcok 단위는 "tick" 이다. Transputer는 우선순위에 따라 2종류의 processor clock을 갖는다. 즉, 시간 측정을 위해서 clock( ) 값을 CLOCK_PER_SEC로 나누는데 이때 높은 우선순위는 1,000,000으로 나누며, 그 결과 1 tick은 1 μs가 된다. 반면에 낮은 순위는 15,625로 나누어 1 tick은 64 μs가 된다.

IP의 성능 측정시에 라우트 프로세스는 라우팅 테이블의 크기에 따라 크게 좌우된다. 본 논문은 성능에 영향을 미치지 않도록 모든 구현에 동일한 라우팅 테이블을 이용하였다.

〈표 3〉은 송신측 IP의 MISD 구조에 대한 평균 처리 시간이다. 채널통신 모델에서 채널을 경유한 사건 전송이 성능 저하 요소임을 알 수 있다. 포크-조인 모델에서는 프로세스 생성이 과부하 요소이며, 추가적으로 서로 다른 처리 시간을 갖는 프로세스들이 포크 후에 조인 할 때 많은 시간을 기다린다. 반면에 사건 조회 모델은 스케쥴링과 조인에 대한 과부하는 없다. 그러나 프로세스간 문맥전환이 성능에 상당한 영향을 미침을 알 수 있다. 그럼에도 불구하고 송신측에서는 사건조회 모델이 가장 좋은 성능을 얻었다.

송신측에서 프라그맨테이션이 발생하면 헤더 수정과 체섬 수정에 따른 약 30%의 추가적인 시간이 소요되었다.

〈표 3〉 송신측의 처리 시간 (단위 tick, 1 tick = 64 μs)
〈Table 3〉 Processing delay at the sending side (1 tick = 64 μs)

모델 / 기능	채널통신 모델	포크-조인 모델	이벤트 폴링 모델
사건도착 감지	1	1	2
첫 번째 프로세스 시작	64	33	0
라우트 프로세스	2	2	2
문맥전환	1	1	31
데이터그램구성 프로세스	8	8	8
조인 (join)	1	2	0
전체 시간	77	47	43

〈표 4〉는 수신측의 처리 시간을 보여준다. 송신측과는 달리 사건조회 모델이 문맥전환 과부하에 의해서 최악의 성능을 보였다. 즉, 프로토콜 기능들이 많아짐에 따라서 프로세스간의 문맥전환이 자주 발생하게 되며 이에 따른 과부하는 전체 성능 저하의 대부분을 차지함을 알 수 있다. 포크-조인 모델과 채널통신 모델은 프로토콜 기능 프로세스들 중 하나의 프로세스가 사건을 받을 때 까지 성능 저하를 야기하였다.

〈표 4〉 수신측 처리 시간 (단위 tick, 1 tick = 64 μs)
〈Table 4〉 Processing delay at the receiving side (1 tick = 64 μs)

모델 / 기능	채널통신 모델	포크-조인 모델	이벤트 폴링 모델
사건도착 감지	1	1	2
첫 번째 프로세스 시작	64	33	0
라우트 프로세스	2	2	2
문맥전환	1	1	31
헤더처리 프로세스	4	4	4
문맥전환	1	1	28
헤더책섬 프로세스	7	7	7
문맥전환	1	1	26
TTL계산 프로세스	2	2	2
조인 (join)	1	2	0
전체 시간	84	54	112

〈표 3〉과 〈표 4〉의 결과, 포크-조인 모델이 다음 두 가지에 기인하여 가장 좋은 모델로 판단된다.

- 채널통신 모델은 채널을 경유하는데 따른 추가적인 프로세스간 통신 과부하가 발생한다.
- 사건조회 모델은 기존의 컴퓨터에서 통신 기능에 적용하여 왔다. 이 모델이 다중 프로세스 모델에 적용될 때 문맥전환에 따라서 좋은 성능을 얻을 수 없다.

6. 결 론

고속통신망 기술이 발전에 따라 전송 문제에서 프로토콜 처리 부분으로 병목 지점이 옮겨가고 있다. 성능을 향상 시키기 위한 많은 연구가 있었으며, 그 중에서 다중 프로세서를 이용한 병렬 프로토콜 구현이 한 분야이다. 즉, 프로토콜에 있는 병렬성을 추출하여 처리 능력을 최대화하였다. 그러나 병렬처리는 프로세서의 제약 뿐만 아니라 프로세서간 동기화, 프로세서간 통신, 스케쥴링 과부하에 의해서 제약되어 왔다. 그 결과 다중 프로세서 플랫폼에서도 어떤 프로토콜 기능들은 한 개의 프로세서에서 병렬로 구현되어 왔다. 따라서 통신 프로토콜 구현시 다중 프로세스 모델에 대한 성능 저하 요인을 발견하는 것이 중요한 문제이다.

본 논문에서는 병렬 프로토콜 구현을 위해서 채널통신 모델, 포크-조인 모델, 사건조회 모델을 제안하였다. 또한 MISD 구조를 따르는 각 모델을 평가하기 위해서 IP를 구현하였다. 다중 프로세스 구현에 대한 성능 측정을 위해서 8개의 Transputer를 다중 프로세서 플랫폼으로 사용하였다. 구현구조는 IP를 송신측과 수신측으로 분리한 후에 각 Transputer에 할당하였고, 성능은 각 Transputer에 구현한 다중 프로세스 모델의 실행시간 과부하를 측정하였다.

관찰 결과 MISD 구조에서 포크-조인 모델이 가장 안정적인 결과를 얻었다. 특히 다중 프로세스 모델에서 문맥전환 시간이 다중 프로세스 생성 시간 보다 상당한 과부하 요인임을 발견하였다.

향후, 보다 복잡하고 다양한 병렬화가 적용될 수 있는 TCP (Transmission Control Protocol)를 구현할 예정이며, 또한 인터넷 데몬인 "INETD"와 일반 인터넷 응용 프로토콜에 이를 적용할 예정이다.

참 고 문 헌

[1] M. Zitterbart, "A Multiprocessor Architecture for High Speed Network Interconnection," INFOCOM'89, pp. 212-218, 1989.

[2] N. Jain, M. Schwartz, and T. R. Bashkov, "Transport Protocol Processing at Gbps Rates," SIGCOMM'90, pp. 188-199, Sept. 1990.

[3] Z. Haas, "A Protocol Structure for High-speed Communication ove- Broadband ISDN," IEEE Network Magazine, Vol. 5, No. 1, pp. 67-70, 1991.

[4] M. Zitterbart, "High-speed Transport Components," IEEE Network Magazine, Vol. 5, No. 1, pp. 54-63, 1991.

[5] T. Braun and M. Zitterbart, "A Transputer based OSI-gateway for LAN-interconnection across ISDN," 16th Annual Conference on Local Computer Network, pp. 158-165, Oct. 1991.

[6] T. Braun and M. Zitterbart, "A Parallel Implementation of XTP on Transputers," 16th Annual Conference on Local Computer Network,

pp. 321-329, Oct. 1991.

[7] C. Diot and V. Roca, "XTP/TRM Implementation on a Transputer Network," 16th Annual Conference on Local Computer Networks, pp. 310-320, Oct. 1991.

[8] G. Koufopavlou, A. N. Tantawy, and M. Zitterbart, "Analysis of TCP/IP for High Performance Parallel Implementations," 17th Annual Conference on Local Computer Network, pp. 576-585, Sept. 1992.

[9] K. Maly and et. al, "Parallel TCP/IP for Multiprocessor Workstations," 4th IFIP Conference on HPN, pp. C1-1-C1-16, Dec. 1992.

[10] E. Rutsche and M. Kaiserswerth, "TCP/IP on the Parallel Protocol Engine," 4th IFIP Conference on HPN, pp. C2-1-C2-16, Dec. 1992.

[11] R. Ito, T. Y. Takeuchi and G. W. Neufeld, "A Multiprocessor Approach for Meeting the Processing Requirements for OSI," IEEE JSAC, Vol. 11, No. 2, pp. 220-227, 1993.

[12] M. Kaiserswerth, "The Parallel Protocol Engine," IEEE/ACM Trans. on Networking, Vol. 1, No. 6, pp. 650-663, 1993.

[13] S. Fisher and W. Effelsberg, "Efficient Configuration of Protocol Software for Multiprocessor," IFIP Conference on HPN VI, pp. 195-211, Sep. 1995.

[14] S. Choi and K. Chon, "Partial ASN. 1/BER Decoding Scheme in a Multiprocessor Environment," Computer Communications 9, pp. 152-159, 1996.

[15] S. Choi and K. Chon, "Implementing OSI Protocol Stack in a Multiprocessor Environment," IEICE Transactions on Communications, vol. E79-B, No. 1, pp. 28-36, Jan. 1996.

[16] H. Abu-Amara and et. al, "PSi: A Silicon Compiler for Very Fast Protocol Processing," IFIP Workshop on Protocols for HSN, pp. 181-196, May 1989.

[17] M. Pancake, "Multithreaded Languages for Scientific and Technical Computing," Proc. of the IEEE, Vol. 81, No. 2, pp. 288-304, 1993.

[18] D. C. Schmidt and T. Suda, "Transport System Architecture Services for High-Performance Communications Systems," IEEE JSAC, Vol. 11, No. 4, pp. 489-526, May 1993.

[19] J. Postel, ed., 'Internet Protocol,' RFC 793, USC/Information Sciences Institute, Sep. 1981.

[20] Parsec Developments, 'Par.C System: User's Manual and Library Reference,' 1990.

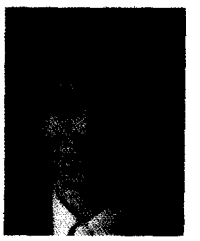
[21] D. J. Flynn, "Very High Speed Computing Systems," Proceedings of the IEEE, Dec. 1996.

[22] INMOS Ltd., 'The Transputer Family,' 1987.

[23] Parsytec GmbH, 'SuperCluster Technical Documentation, Installation, Expansion and Maintenance Manual, Revision 1.2,' 1989.

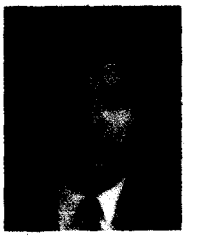
[24] Parsytec GmbH, 'Parsytec Toolset Supplement Guide,' 1991.

[25] Perihelion Software Ltd., 'The Helios Parallel Operating System,' 1991.



최 선 완

1984년 홍익대학교 전자계산학과 졸업(학사)
1986년 한국과학기술원 전산학과 졸업(공학석사)
1996년 한국과학기술원 전산학과 졸업(공학박사)
1986년~1996년 한국전자통신연구원 선임연구원
1996년~현재 안양대학교 정보통신공학과 전임강사
관심분야: 인터넷 멀티미디어 통신 프로토콜, 인터넷 전화, 위성통신



정 광 수

1981년 한양대학교 전자공학과(학사)
1983년 한국과학기술원 전기 및 전자공학과(공학석사)
1991년 미국 University of Florida 전기공학과(공학박사)
1983년~1993년 한국전자통신연구원 선임연구원
1991년~1992년 한국과학기술원 대우교수
1993년~현재 광운대학교 전자공학부 부교수
관심분야: 컴퓨터 통신, 멀티미디어 정보부호화 및 통신, 분산처리