

큐의 우선순위에 근거한 흐름제어방식

이 광 준[†] · 손 지연[†] · 손 창 원[†]

요 약

본 논문에서는 우선순위에 기초를 둔 새로운 흐름제어 방식을 제안한다. 이 방식에 따르면 각 통신경로는 유지하고자 하는 적정 버퍼의 크기를 설정하고, 전송중 이를 초과하지 않도록 해당 통신경로의 전송 우선순위를 조절한다. 대용량의 통신대역폭을 요구하는 통신경로는 작은 크기의 적정 버퍼를 설정한다. 적정 버퍼의 크기와 전송 우선순위가 반비례하게 정의하면, 통신경로의 전송 우선순위는 높게 설정되므로, 각 노드는 설정된 통신경로를 지나는 패킷에 높은 처리 기회를 제공하게 되고, 이를 통하여 버퍼의 크기를 적정 수준이하로 감소시키는 작용을 한다. 또한 대용량의 통신대역폭을 요구하는 통신경로에서는 경로상에 있는 노드에 적은 수의 패킷을 유지하게 함으로 짧은 전송지연을 갖게 된다. 각 통신경로의 버퍼 크기는 통신경로가 연결된 동안 일정한 시간 간격으로 재조정될 수 있다. 사용하고 있는 통신망 대역폭이 통신경로의 최종 목적지 응용 프로그램이 필요로 하는 데이터 흐름보다 적은 경우, 그 노드는 큐의 크기를 줄임으로써 대역폭을 보다 크게 확보할 수 있다. 통신망간 연동 환경에서는 나은 통신망 대역폭을 균질한 노드로부터 확보할 수 있으며, 따라서 네이터의 응답이 전통적인 end-to-end 흐름제어 기법보다 빨라진다. 고속의 통신망 환경에서는 제안된 흐름제어 기법이 보다 효율적인 성능을 보인다.

A Flow Control Scheme based on Queue Priority

Goang-Jun Lee[†] · Ji-Yeon son[†] · Chang-Won Son[†]

ABSTRACT

In this paper, a flow control mechanism is proposed which is based on the priority control between communication path of a node. In this scheme, demanding length of a data queue for any path is pre-defined, then each node in that path is forced to maintains buffer size under the limit by controlling priority level of the path. The communication path which requires higher bandwidth sets its demanding queue length smaller. By providing relationship between the priority of a path and length of its queue, the high bandwidth requesting path has a better chance to get high bandwidth by defining the smaller demanding queue size. And also, by forcing a path which has high flow rate to maintain small queue size in the path of the communication, the scheme keep the transmission delay of the path small. The size of the demanding queue of a path is regularly adjusted to meet the applications requirement, and the load status of the network during the lifetime of the communication. The priority control based on the demanding queue size is also provided in the intermediate nodes as well as the end nodes. By that the flow control can provide a quicker result than end-to-end flow control, it provides better performance advantage especially for the high speed network.

[†] 정회원: 한국전자통신연구소 컴퓨터연구단

논문접수: 1995년 10월 17일, 심사완료: 1996년 10월 23일

1. 개 요

전통적인 흐름제어는 통신경로 또는 세션 개개의 특성에 속하였다. 따라서 흐름제어는 특정 통신경로의 양 종단간의 흐름을 일정하게 하는 데 관심을 주로 가졌다. 이러한 하나의 예는 윈도우 기법을 이용하여 버퍼의 오버플로우를 방지하는 것이다[1]. 윈도우에 기반을 둔 흐름제어 기법에서는, 통신하고자 하는 호스트간에 연결설정시 윈도우라 불리는 서로 동의한 일정한 크기의 버퍼를 확보하며, 확보된 버퍼를 가능한 한 효율적으로 이용하여 데이터의 전송을 수행한다. 이 기법의 단순성으로 인하여 TCP를 포함한 많은 통신 프로토콜에서 윈도우 기법을 이용하고 있다[3, 4]. 그러나 이러한 기법을 자세히 살펴보면, 데이터의 수신측에서 버퍼의 부족 현상이 발생하면 수신측의 버퍼 상태를 송신측에 알려서 수신측에서 충분한 버퍼를 확보할 때까지 송신측에 데이터의 전송을 일시 중단하도록 하는 “back pressure mechanism”에 기초한 방식이다. 이러한 방식의 목적은 버퍼를 보다 효율적으로 이용하는 것이며, 이를 위해서 각 호스트의 버퍼 상태를 상호 교환하여야 한다. 이러한 방법은 통신 경로상에 많은 호스트가 존재할 수 있는 Internet과 같은 multi-hop 통신 환경에서는 부적절할 수 있다.

중간 노드에서 흐름제어 기능을 제공하기 위해서 여러가지 방법이 제안되고 있다. 이러한 방식의 예로 써, Internet 환경에서 제안된 예약 프로토콜의 일종인 RSVP가 있다[2]. 예약 프로토콜은 각 통신 노드의 통신 자원인 대역폭, 데이터 전송 비율 등을 미리 할당하는 방법이다. 그러나 이러한 기법은 몇 가지 부담을 가지고 있다. 첫째, 각 통신경로의 데이터 흐름을 조절하기 위하여 개개의 데이터 패킷을 스케줄링을 하여야 한다. 둘째, 이러한 방법은 통신자원의 낭비를 초래한다. 예를 들면, 대량의 데이터를 간헐적으로 전송하는 경우 통신 경로상에 있는 노드들은 송신자를 데이터를 전송하지 않음에도 불구하고 기법상의 결점으로 인하여 통신자원을 계속 점유하여 통신자원의 낭비를 초래한다. 따라서 통신망의 모든 데이터 흐름에 대하여 통신자원을 예약하는 것은 바람직하지 않다.

또 다른 문제점은 중간 노드에서 데이터를 축적방

식에 의하여 전달하기 때문에 예기치 못한 데이터의 전달 지연이 발생한다. 일반적으로 중간 노드들은 통신망의 폭주로 인한 데이터의 손실을 방지하기 위하여 필요이상의 버퍼를 준비한다. 그러나 통신망의 속도가 빨라짐에 따라 과도한 버퍼링에 의한 지연은 데이터의 전달 지연을 예측 불가능하게 하는 문제를 야기시킨다. 따라서 손실된 데이터의 재전송을 위한 timeout 값의 결정을 어렵게 한다. 따라서 윈도우를 기반으로 한 end-to-end 흐름제어는 비효율적이라고 할 수 있다. 이러한 문제를 해결하기 위하여 다양한 논문에서 hop-by-hop 기법을 이용한 방법을 제안하고 있다[5, 6, 7, 8]. 이러한 hop-by-hop 기법은 버퍼 사용률과 같은 각 노드의 정보를 기반으로 한 전송률을 제어 기법에 의존한다. 각 노드의 처리 능력이 증가하더라도 각 노드에서 non-timer 방식의 효율적인 흐름제어가 제공되지 않으면 패킷의 전송속도를 제어하는 것이 매우 비싼 비용을 치르게 된다.

본 논문에서는 각 노드 통신경로에 우선순위를 두어 흐름제어를 수행하는 흐름제어 기법을 제안하였다. 우선순위는 각각의 통신경로에 따른 원하는 큐의 크기를 근거로 하였다. 예약기법과는 달리 제안된 기법은 어떠한 자원도 미리 확보하지 않으며, 각 통신경로가 존재하는 한 동적으로 각 통신경로의 전송 우선순위를 규정하여 이에 따라 데이터 전송을 수행한다. 본 논문은 다음과 같이 구성되었다. 제2절에서는 제안된 흐름제어 기법의 기본적인 개념을 설명한다. 개념에 따른 자세한 흐름제어 기법은 제3절에서 설명한다. 그리고 제4절에서는 제안된 기법과 기존의 윈도우를 기반으로 한 흐름제어 기법과 효율성을 비교하기 위하여 시뮬레이션하고 그 결과를 나타내었다.

2. 큐의 우선순위에 근거한 흐름제어

통신에서는 각 노드의 통신경로를 효율적으로 사용하기 위한 공평성 문제가 매우 중요하다. 이러한 문제를 해결하기 위해서 다양한 방법이 제안되었다. 이러한 기법중의 하나는 통신망의 자원을 미리 예약하는 예약기법이 있다. 그러나 이러한 기법은 통신망 자원을 관리하기가 어려울 뿐만 아니라 통신망 자원을 낭비하게 되는 측면도 있다. 본 논문에서는 통신망 자원의 하나인 통신망 통신경로를 효율적으로 사

용할 수 있는 새로운 기법을 제안한다. 제안된 기법은 각 노드에서 통신경로에 할당할 수 있는 큐의 크기를 기초로 하여 각 통신경로에 우선순위를 두는 방법이다.

통신경로가 개설되면, 그 통신경로에 따르는 원하는 큐의 크기를 개설된 통신경로에 속하는 모든 노드에게 알리게 된다. 각 노드들은 이러한 큐의 크기를 기초로 하여 데이터 패킷의 처리 순위를 결정하기 위해 각 통신경로의 우선순위를 결정한다. 각 통신경로의 우선순위는 통신경로에 따른 큐의 크기와 반비례 한다. 즉, 큐의 크기가 가장 큰 경우 우선순위가 가장 낮고, 큐의 크기가 가장 작은 경우 우선순위가 가장 높다. 따라서 우선순위가 가장 높은 통신경로에 데이터 패킷을 처리할 기회가 가장 많이 오게 된다. 이러한 기법은 예약기법과는 달리 통신망 자원을 사전에 예약할 필요가 전혀 없으며, 단지 각 통신경로에 따르는 큐의 크기를 통신경로 개설시 제시한 한계이내로 유지하려는 노력만 필요하다.

통신경로에 따른 큐의 크기는 시간에 따라 변할 수 있다. 즉 통신경로에서의 전송속도, 노드에 존재하는 통신경로의 수, 다른 여러 통신경로로부터의 데이터 패킷의 수 등에 따라 큐의 크기가 영향을 받는다. 이로 인하여 큐의 크기가 초기에 주어진 한계값보다 커질 수 있다. 이러한 경우가 발생하면 그 노드에서는 큐의 크기를 한계값이하로 줄이기 위하여 조치를 취하게 되는 테, 여러 가지 방법이 있다. 첫째, 노드는 큐의 크기를 줄이기 위하여 통신경로의 우선순위를 보다 높여 준다. 둘째, 노드는 큐의 크기를 줄이기 위하여 데이터의 송신 노드에게 데이터 전송속도를 낮추도록 요청한다. 세째, 노드는 그 노드에 속하는 전체 통신경로의 수를 제한한다. 이러한 각각의 방법은 출력제어, 입력제어, 혼가제어방식이라 칭할 수 있으며, 각각의 장단점을 가지고 있다. 예를 들면, 출력제어 방식은 이미 폭주상태에 있는 노드에서는 모든 통신경로들이 궁극적으로 보다 높은 우선순위를 요청하게 되므로 적절한 해결책이 되지 못한다. 입력제어 방식 또한 주어진 통신경로에 입력되는 데이터의 비율을 제한하여 결국 대역폭을 줄이게 되므로 바람직하지 않다. 새로운 통신경로의 개설을 제한하거나, 기존의 통신경로를 단절시키는 방법은 통신망의 관점에서 볼때, 균등한 통신망 자원의 이용 측면에서 심

각한 문제를 초래할 수 있다. 따라서 특정한 방법을 통한 해결보다는 여러 방법을 적절히 활용하여 보다 나은 결과를 얻도록 해야 한다.

3. 흐름제어 방식

이 절에서는 제안된 흐름제어 방식의 동작에 관해서 설명한다. 흐름제어를 수행하기 위해서 각 통신경로의 진행 상태에 따라 4개의 프로세스가 존재한다. 이들은 각각 “Path Establishment Process”, “Packet Scheduling Process”, “Flow Control Process”, “Path Termination Process”라 부른다.

3.1 Path Establishment Process

새로운 통신경로가 설정되면 하나의 새로운 데이터구조가 생성된다. (그림 1)에 나타난 바와 같이 새로운 데이터구조는 적어도 다음과 같은 부분을 포함한다. 즉 “Queue Limit”, “Current Queue Length”, “Packet Queue Pointer”이다. “Queue Limit” 필드는 데이터 패킷을 위해 할당된 큐의 한계 크기를 알기 위해서 필요하다. 노드에 현재 전송을 위해서 대기하고 있는 패킷의 수는 “Current Queue Length” 필드에 의해 유지 관리된다. “Packet Queue Pointer” 필드는 각 통신경로로 전송되기 위해서 대기하고 있는 첫번째 데이터의 위치를 가리킨다. 통신경로가 개설되어 데이터가 전송되기 전에는 각 통신경로의 정확한 큐의 크기를 계산하는 것은 어렵다. 따라서, 통신경로 개설 단계에서는 초기치로써 임의의 큐 크기를 설정하게 된다. 각 용용 프로그램이 통신망의 대역폭의 필요에 따라 프로그램의 요구에 맞는 초기 큐 크기를 갖게 된다. 큐의 크기는 통신경로를 통해 데이터가 전송됨에 따라 그 크기가 조정된다. 통신경로 개설시 최초로 전송되는 데이터 패킷에는 통신경로의 초기 큐 크기를 요청하는 “QUE_REQUEST”라는 정보를 포함한다. 통신경로상의 임의의 중간 노드는 이 정보를 통신경로 정보의 일부로 유지한다. 이 “QUE_REQUEST” 정보는 큐의 크기를 한계 크기 이하로 유지하기 위하여 통신경로상의 각 노드들이 사용한다. 모든 개설된 통신경로로부터 수집된 “QUE_REQUEST” 정보를 각 노드들은 추후에 사용하기 위하여 표로써 분류하고 유지 관리한다. 동일한 “QUE_REQUEST”

값을 갖는 통신경로는 동일한 우선순위를 갖는 집단으로 분류한다. “QUE_REQUEST” 값이 가장 작은 집단의 통신경로의 우선순위가 가장 높게 책정된다. 이 값은 통신경로 데이터 구조중 “Queue Limit” 필드에 저장된다. 각 노드는 개설된 통신경로중에서 이러한 우선순위에 따라서 다음에 전송할 패킷을 결정한다.

Queue Limit
Current Queue Length
Packet Queue Pointer

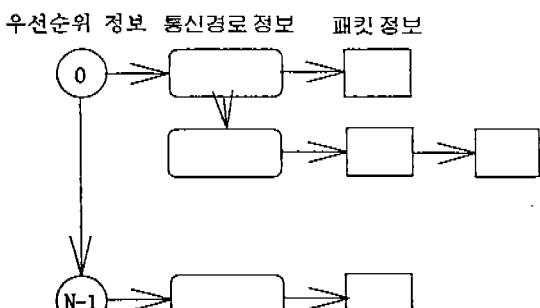
(그림 1) 통신경로 정보 데이터 구조
(Fig. 1) Path Information Data Structure

3.2 Packet Scheduling Process

축적방식(Store-and-Forward)에서는 도달되는 각 패킷은 큐에 임시로 저장된다. 그림 2에 나타난 것처럼 패킷은 각 통신경로에 대한 연계리스트의 형태로 저장된다. 우선순위에 따라 집단으로 나누어 전 통신경로는 한번에 한 통신경로씩 round-robin 방식으로 우선순위에 따라 데이터를 전송하게 된다. 우선순위가 반영되는 방법은 다음 몇가지를 포함하여 여러 가지가 있을 수 있다.

1) 각 통신경로가 서비스될 때, 우선순위에 따라 서로 다른 수의 통신경로를 스케줄러가 선택한다. 예를 들면, 우선순위 1인 집단에서는 통신경로를 5개 선택하고 우선순위가 5인 집단에서는 1개의 통신경로를 선택하는 방법.

2) 스케줄러가 모든 우선순위 집단에서 각각 1개의



(그림 2) 우선순위와 통신경로정보의 구조
(Fig. 2) Queue Priority & Path Information Structure

통신경로를 선택하지만, 통신경로의 우선순위에 따라 전송되는 데이터 패킷의 수를 다르게 하는 방법.

이들 방법에서 보다 높은 우선순위를 갖는 통신경로에게 전송 대역폭을 보다 낮은 우선순위를 갖는 통신경로보다 더 할당하게 된다. 이러한 방법에 의해 결국은 큐의 크기를 한계치 이하로 유지하게 된다.

3.3 Flow Control Process

통신경로의 현재 큐크기가 미리 협상된 값을 초과하게 되는 원인은 크게 2가지로 나누어 생각할 수 있다. 먼저, 특정한 통신경로로 데이터 패킷이 너무 빨리 도착하는 경우, 그리고 통신망 노드의 다른 통신경로로부터의 부하가 너무 많은 경우이다. 임의의 통신경로의 큐크기가 협상된 값을 초과하는 원인을 판정하는 것은 쉬운 문제가 아니지만, 이러한 문제를 해결하기 위한 방법을 몇가지 생각해 볼 수 있다. 첫 번째, 큐에 과도한 데이터를 가진 통신경로가 많지 않은 경우는 문제가 해결될 때까지 해당 통신경로들의 우선순위를 높여 주는 것이다. 큐에 과도한 데이터를 가진 통신경로가 많은 경우에는 해당 통신경로의 근원지 통신경로에 큐의 크기를 재조정하여 데이터 전송속도를 늦추게 요청하는 방법이다.

통신경로의 과부하 여부는 2개의 변수 “LINKS”와 “EXCEED_LINKS”에 따라 결정된다. “LINKS” 변수는 노드에 존재하는 통신경로의 갯수를 나타내며, “EXCEED_LINKS” 변수는 미리 협상된 큐크기보다 현재 큐의 크기가 큰 통신경로의 갯수를 나타낸다. “Flow Control Process”는 주기적으로 각 통신경로의 상태와 이를 변수를 조사한다. “EXCEED_LINKS”와 “LINKS”的 비율이 미리 기술된 값보다 크면 그 노드가 현재 폭주상태(Congestion)에 있다고 가정한다. 폭주상태는 두 변수의 비율에 따라 과도한 폭주상태와 적절한 폭주상태로 구분한다. 노드의 현재 사용 가능한 버퍼의 크기가 폭주상태의 정도를 판단하는 다른 요소가 될 수 있다. 노드의 상태가 과도한 폭주상태인 경우, 새로운 통신경로가 개설되는 것을 제한함으로써 폭주방지절차를 시작한다. 이러한 접근제어(Access control)를 함으로써 노드에 더이상의 폭주상태가 발생하는 것을 방지할 수 있다.

노드의 폭주상태가 과도하지 않은 경우 각 통신경로에 대한 큐관리는 다음과 같다. over-queue 상태가

길어질 경우, 노드가 폭주상태가 아닌 경우에는 데이터의 송신측에 데이터 전송 비율을 낮추도록 요청한다. 그리고 데이터의 폭주상태가 심각하지 않은 경우에는 통신경로의 우선순위를 높여준다. 이렇게 함으로써 그 통신경로의 전송속도가 빨라져서 결과적으로 큐의 길이가 이전에 설정된 큐의 한계치이하에 도달하게 된다. 송신측의 데이터 전송 비율을 낮추는 방법은 문제점을 해결할 수는 있지만, 송신측에서 원하는 방법은 아니다. 또한 흐름제어를 수행할 수 있는 방법을 제공하지 않는 한, 데이터의 전송율을 낮추는 것이 쉽지 않다. 보다 단순한 방법으로 통신경로의 우선순위를 보다 낮게 설정함으로써 동일한 목적을 달성하는 방법이 있다. 마찬가지로 보다 큰 데이터 전송율이 필요한 경우, 통신경로의 우선순위를 높이기 위한 제어메시지를 보냄으로써 해결할 수 있다. 우선순위를 높이는 것은 큐의 크기를 줄이는 것을 의미한다. 이러한 기법은 응용 프로그램을 위해서 데이터의 흐름을 제어하기 위하여, 데이터의 종착지에서도 사용할 수 있다. 이러한 흐름제어 기법을 multihop 통신망 환경에서 사용함으로써 종단간의 흐름제어 방식보다 빠른 응답시간을 통신망이 제공할 수 있다.

3.4 Path Termination Process

통신경로의 종료 방법에는 “의도된 종료(Intended termination)” 및 “의도하지 않은 종료(Unintended termination)”가 있다. 의도된 종료는 데이터의 송신측이, 더이상의 데이터 전송을 하지 않으려는 경우이며, 의도되지 않은 종료인 경우는 그러한 경우 이외의 여러가지 다른 이유, 예를 들면 일정기간 동안 데이터의 전송이 없는 경우에 발생한다. 부적절하게 통신경로가 종료된 경우, 이를 회복하는 비용이 매우 크기 때문에, 이를 방지하기 위한 적절한 time-out값을 설정하는 것이 쉽지 않다. 통신경로의 비용은 통신경로를 유지하기 위한 비용과 데이터 전송을 위해서 항목을 찾기 위한 비용으로 이루어 지며, 이러한 값은 허용 가능한 범위에서는 크게 선택할 수 있다. 그러나, 이 값이 데이터 송신측에서의 값보다 크게 되면, 통신경로가 비정상적으로 종료되고 따라서 다시 통신경로를 개설하게 된다. 그러나 이 값이 너무 작게 되면 송신측은 종료된 통신경로를 통하여 데이터

타 전송을 시도하는 결과가 되고 만다. 이러한 경우 적절한 정보가 없이 통신경로를 새로이 개설하여야 하므로 문제가 있다.

4. Simulation

제안된 흐름제어 기법의 효율성을 평가하기 위하여 단순하고도 유연한 통신용 시뮬레이터를 개발하였다. 시뮬레이터의 구성은 다음과 같다.

- input parser
- simulation engine
- simulated node block
- output data generator

Parser는 문서편집기로 작성된 시뮬레이션 모델을 기술한 화일을 입력으로 받아 들인다. Output Data Generator는 시뮬레이션 결과를 그림으로 출력하기 위한 데이터를 생성하여 화일에 저장한다. 또한 그 결과를 X-windows 환경에서 수행되는 유필리티인 gnuplot이 이용할 수 있는 명령어 화일을 생성하여, gnuplot 명령어를 이용하여 X-windows 환경에서 시각적으로 시스템의 동작상황을 나타낼 수 있다. Simulation Engine은 각 프로세스의 시작 시간에 맞추어 프로세스가 동작될 수 있도록 한다. 그리고 노드 복록은 다양한 형태의 노드와 통신경로로 구성된다. 현재 가능한 노드 형태는 송신자, 수신자, 중간 노드가 있다. 노드의 특성은 현재 hard-coded 형태로 구성이 되므로, 시뮬레이션하기 위한 통신망을 구성하는 경우, 데이터의 생성, 소비, 전송비율과 같은 파라미터의 변경만 허용한다. 현재까지 시뮬레이터가 지원하는 흐름제어의 형식은 두가지이다. 고정된 윈도우 크기를 갖는 Sliding window 방식과 본 논문에서 제안된 큐의 크기를 기반으로 한 흐름제어 방식이다. 시뮬레이션하고자 하는 흐름제어 기법은 시뮬레이션의 시작 시 명령어의 파라미터로 선택할 수 있다. 시뮬레이션의 기본단위는 통신용 통신경로이다. 통신경로에는 송신자, 수신자, 통신경로를 구성하는 중간 노드가 포함된다. 다음에 단일 통신경로로 구성되는 간단한 통신 모델을 작성하기 위한 입력화일의 예를 나타낸다. 시뮬레이션 모델이 다중통신경로를 갖는 경우, 다중

통신경로를 참조함으로써 각 노드는 여러개의 통신 경로를 가질 수 있다. 다음은 하나의 통신경로를 갖는 시뮬레이션 모델을 구성하는 시뮬레이션 입력화 일의 예이다.

```
# simple network
# first argument is type of node
# second argument is transmission interval
s_1 = add_node(TYPE_SENDER, 10);
r_1 = add_node(TYPE_RECEIVER, 10);
i_1 = add_node(TYPE_ROUTER, 10);
h = hop_list(i_1);
# s_1 is sender node, r_1 is receiver node
# third argument is the link's transmission start time
# of data stream
# fourth argument is queue limit (window size for
# window based flow control)
# fifth and sixth arguments are generation and con-
# sumption interval
```

the last argument is the list of intermediate nodes in
the link

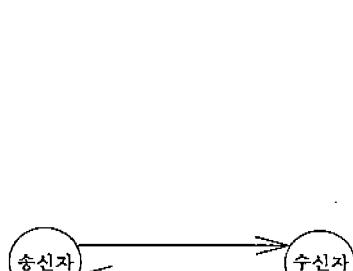
p_1 = create_path(s_1, r_1, 100, 10, 5, 5, h);

4.1 Network Models

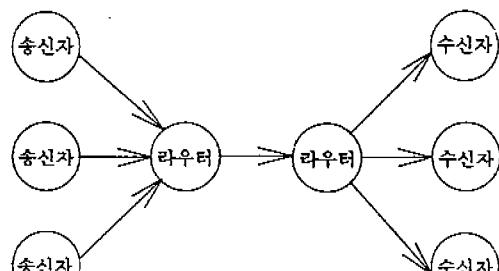
본 논문에서 제안된 흐름제어 기법을 평가하기 위하여 몇 개의 통신 모델을 구성하였다. (그림 3. a)는 가장 단순한 형태로써, 하나의 송신자와 하나의 수신자로 구성되며, 다중통신경로를 설정할 수 있다. 각 통신경로는 시뮬레이션 기간에 서로 다른 통신 설정 시간을 갖는다. 통신 모델이 중간 노드를 갖게 되면 (그림 3. b)의 형태가 된다. 또 다른 모델은 2개의 송신자, 2개의 수신자, 2개의 중간노드를 갖는 형태로 (그림 3. c)에 나타나 있다.

4.2 Simulation Results

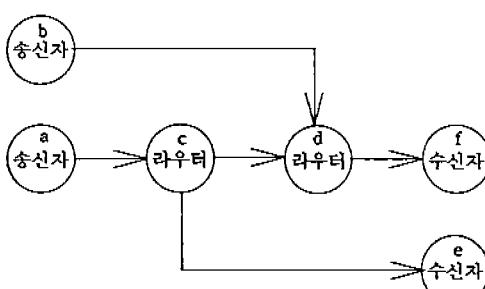
시뮬레이션에서 시간의 기본 단위는 “tick”이라 불린다. 데이터 패킷의 생성 및 소비 비율은 tick count 값에 따른 데이터 패킷간의 도착시간을 반영한다. 각



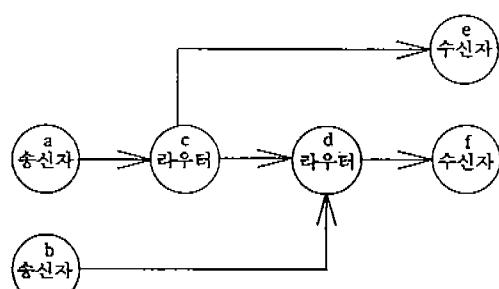
(a) 송수신자가 각 1인 경우



(b) 라우터가 있는 경우



(c) 보다 복잡한 경우



(d) 보다 복잡한 경우

(그림 3) 시뮬레이션 모델

(Fig. 3) Simulation Models

각의 시뮬레이션 결과는 global clock이 80,000 tick을 경과한 후의 결과이다. 데이터 패킷은 무작위로 도착 하며, 각종 사건이 발생한다. 패킷의 생성, 소비, 전송 간격이 이에 속한다. 시뮬레이션을 단순하게 하기 위해서 각 통신경로의 단위 길이당 패킷 전송 지연은 일정하다고 가정한다. 시뮬레이션에서 중요한 값을 갖는 변수는 다음과 같다.

- a. 송신측의 패킷 생성 간격(α)
- b. 수신측의 패킷 소비 간격(β)
- c. 송신측의 패킷 전송 간격(γ)
- d. 중간 노드의 패킷 전송 간격(μ)
- e. 통신경로의 단위 길이당 지연(η)

(그림3)의 모델에서 흐름제어 기법으로 원도우 방식과 제안된 방식에 따른, 송신자와 수신자간의 전달된 패킷 및 각 패킷의 평균 전송 지연을 비교하였다. 각 모델에서 원도우의 크기와 초기의 큐크기를 10으로 선택하였다.

첫 모델인 (그림 3. a)는 하나의 송신자와 하나의 수신자로 구성되어 있으며, 3개의 통신 통신경로를 사용한다. 송수신자간에는 단일의 전송회선을 가지고 있으며, 3개의 통신경로는 전송회선을 공유하고 있다. 모든 송신노드의 패킷 생성간격 α 는 20 tick이고, 수신노드의 패킷 소비간격 β 는 10 tick이다. 전송회선의 전송 용량은 10 tick을 설정하여, 평균적으로 한 패킷의 전송에는 10 tick의 시간이 소요된다. 3개의 통신경로는 시차를 두고 전송을 시작한다. 첫번째 통신경로는 system clock이 0 일때, 두번째와 세번째 통신경로는 각각 10,000 및 30,000 tick이 경과한 후 데이터 전송을 시작한다. 이에 의하면 초기에는 패킷의 생성간격이 통신경로의 용량을 초과하지 않으나, system clock이 10,000 및 30,000 tick을 경과하면서 점차 요구하는 정보의 전달량이 통신경로의 용량을 초과하고 있다. 이때 원도우에 의한 흐름제어 방식과, 버퍼의 크기에 의한 흐름제어 방식이 동작하게 된다. 이때 통신경로의 지연 시간을 2, 10, 50 tick의 3가지 값을 사용하여 측정하였다. 표에 나타난 값은 system clock이 80,000 tick을 경과한 후 수신노드에 전송된 총 패킷의 수를 의미한다. 시뮬레이션의 결과는 (표 1)과 같으며, 두 기법에서 커다란 차이를 발견할 수

없다.

〈표 1〉 간단한 모델
〈Table 1〉 Simple Model

	Packet Count	
Path Delay	Window Based	Queue Based
2	10159	10113
10	10166	10110
50	10168	10103

두번째 모델인 (그림 3. b)에서는 3개의 송신노드와 3개의 수신노드들이 2개의 중간노드들을 경유하여 각각 하나의 통신경로를 구성한다. 각 송수신노드는 1개의 통신경로를 설정한다. 모든 시뮬레이션 파라미터는 첫 모델과 동일하게 설정하였다. 또한 첫 모델과 동일하게 각각 송수신노드에 있는 3 개의 통신경로는 시차를 두고 전송을 시작한다. 첫번째 통신경로(송신노드 1)는 system clock이 0 일때, 두번째(송신노드 2)와 세번째 통신경로(송신노드 3)는 각각 10,000 및 30,000 tick이 경과한 후 데이터 전송을 시작한다. 〈표 2〉의 각 “path delay”的 (1), (2), (3)은 각 통신경로를 구별한다. 이 시뮬레이션의 목적은 중간 노드가 과부하일 때 각 통신경로에 미치는 영향을 원도우에 의한 흐름제어 방식과, 버퍼의 크기에 의한 흐름제어 방식을 사용한 경우를 서로 비교하는 것이다. 〈표 2〉의 결과에 의하면 전송지연 η 이 2와 10인 경우에는 두 기법간의 별다른 차이를 발견할 수 없으나, 전송

〈표 2〉 다중전송자인 경우
〈Table 2〉 Multiple Senders

	Packet Count	
Path Delay	Window Based	Queue Based
2	(1)	4064
	(2)	3567
	(3)	2576
10	(1)	4125
	(2)	3600
	(3)	2534
50	(1)	2403
	(2)	2106
	(3)	1506

지연 η 이 더 길어져 50이 되면, 본 논문에서 제안된 방식이 각 통신경로에서 보다 나은 성능을 보임을 알 수 있다.

세 번째 모델인 (그림 3. c)에서는 보다 복잡한 통신망을 설정하였으며, 각각 2개의 송신노드, 수신노드, 중간노드들로 구성되어 있다. 송신노드 a와 수신노드 f간에 통신경로가 개설되었으며 이들 노드간에는 중간노드 c, d가 존재한다. 또 다른 통신경로는 송신노드 b와 수신노드 f간에 개설되었으며, 여기에는 중간노드 d가 존재한다. 마지막 통신경로는 송신노드 a와 수신노드 e간에 개설되었으며, 중간노드 c가 존재한다. <표 3>의 결과는 system clock이 80,000 tick 경과한 후 수신노드 f(1)와 수신노드 e(2)에 전달된 패킷의 수이다. 이 결과는 두 번째 모델과 유사함을 알 수 있다.

<표 3> 복잡한 모델
<Table 3> Complex System

Path Delay	Packet Count	
	Window Based	Queue Based
2 (1)	6725	6741
	3558	3614
10 (1)	6662	6661
	3656	3571
50 (1)	4470	6606
	2932	3593

마지막 모델인 (그림 3. d)에서는 서로 다른 패킷 생성 간격을 가진 송신자들이 존재하는 경우에 각각의 방식을 비교하고자 한다. 이 모델에서는 2개의 통신경로가 사용되었으며, 한 통신경로는 송신노드 a와 수신노드 f사이에 설정되었고, 이들간에는 중간노드 c, d가 존재한다. 또 다른 통신경로는 송신노드 b와 수신노드 e사이에 설정되었고 이들 간에는 중간노드 d가 있다. 송신노드 a는 패킷 생성 간격이 20 tick이며, 송신노드 b는 10 tick인 경우이다. <표 4>에서는 수신노드에 system clock이 80,000 tick 경과한 후 전달된 총 패킷의 수를 나타낸다. 수신노드 e(1)에 전달된 패킷들은 송신노드 a로부터 전송된 패킷이며, 수신노드 f(2)에 전달된 패킷은 송신노드 b로부터 전송된 패킷이다.

<표 4> 복잡한 모델
<Table 4> Complex System

Path Delay	Packet Count	
	Window Based	Queue Based
2 (1)	7664	8156
	2708	2754
10 (1)	7011	8150
	2693	2739
50 (1)	2463	8029
	2618	2665

5. 결 론

본 논문에서는 통신경로 관리에 우선순위 개념을 도입하여 각 통신망 노드에서 통신망 전체의 정보를 이용한 흐름제어 기법을 제안하였다. 각 통신경로의 우선순위는 통신경로에서 필요로 하는 큐의 크기를 기반으로하여 결정된다. 제안된 기법에서는 보다 큰 대역폭을 요구하는 통신경로일수록 큐의 크기가 작아지도록 하였다. 이렇게 함으로써 통신경로는 전송 매체를 통하여 전송되기를 기다리는 데이터가 보다 적어진다. 결과적으로 각 데이터 패킷은 우선순위가 낮은 통신경로의 데이터보다 낮은 전송 지연 시간을 나타낸다. 제안된 기법의 효율성을 평가하기 위하여 시뮬레이션 모델을 개발하였다. 이러한 방법을 통하여 기존의 윈도우를 기반으로 한 흐름제어 기법을 채용한 모델과 그 결과를 비교하였다. 결과에서도 알 수 있듯이 각 노드간의 전송 지연이 증가할수록 제안된 기법이 월등한 성능을 보임을 알 수 있다.

참 고 문 헌

- [1] Charles A. Eldridge, "Rate Controls in Standard Transport Layer Protocols," ACM SIG COMM, vol. 22, No. 3, pp. 106-120, July 1992.
- [2] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, Daniel Zappala, "RSVP: A New Resource ReSerVation Protocol," IEEE Network, pp. 8-17, September 1993.
- [3] Defense Advanced Research Projects Agency, "Transmission Control Protocol," RFC-793, Sep-

tember, 1981.

- [4] V. Jacobson, "Congestion Avoidance and Control," Proceedings of ACM Sigcom, Computer Communication Review, vol. 18, No. 4, pp. 314-329, 1988.
- [5] Partho P. Mishra, Hemant Kanakia, "A Hop by Hop Rate-based Congestion Control Scheme," SIG-COMM '92, Baltimore, Maryland, pp. 112-123.
- [6] G. Ramamurty & B. Sengupta, "A Predictive Hop-by-Hop Congestion Control Policy for High Speed Networks," IEEE INFOCOM '93, San Francisco, CA., March 1993.
- [7] M. Schroeder et al. "Autonet:A High-speed, self-configuring local area network using point-to-point links," IEEE Journal on Selected Areas in Communications, vol. 9, No. 8, pp. 1318-1335, October 1991.
- [8] Jiseung Nam, C. W. Son, K. W. Rim, and T. G. Kim, "Performance Evaluation of Congestion Control on Interworking Conventional LANs with B-ISDN," Proceeding of the SCSC'94, 4838 Ronson Court, Suite L, San Diego, California, pp. 175-180, July 1994.



이 광 준

1988년 울산대학교 전자계산학
과 졸업(학사)

1988년~현재 한국전자통신연구
소 연구원

관심분야: 컴퓨터통신 및 분산처리



손 지 연

1991년 숙명여자대학교 전산학
과 졸업(학사)

1991년~현재 한국전자통신연구
소 연구원

관심분야: 고속/멀티미디어 통신
프로토콜



손 창 원

1980년 아주대학교 전자공학과
졸업(학사)

1983년 University of New Ma-
xico 컴퓨터공학과 졸업
(석사)

1988년 University of Arizona
컴퓨터공학과 졸업(박사)

1994년~현재 한국전자통신연구소 선임연구원

관심분야: 컴퓨터통신, 분산처리