Proceedings of the 11th
KACC, October 1996

공장 자동화를 위한 TCP/IP 네트웍 에서의 실시간 통신에 관한 연구
Evaluation of Realtime communication over TCP/IP network for industrial automation

°윤영찬*, 박재현*

*인하대학교 자동화공학과(Tel: (032)860-7713; Fax:(032)873-4386; E-mail: {treeman,jhyun}@rcsl.inha.ac.kr)

Abstract While Ethernet and TCP/IP are the most widly used protocol, for Real-time system, it is not applicable because it doesn't guarantee the deterministic transmission time. Furthermore, the TCP acknowledgement scheme and sliding window algorithm enforce to collide packets. Although various Collision-Free CSMA protocol was presented, it is very difficult to implement in well known OS(UNIX, WindowsNT) because we have to modify network kernel. This paper presents another transmission protocol based on modified UDP. The colliding probability can be minimized by avoiding successive packet transmission and decreasing competetion duration. The proposed algorithm can be used for the soft real-time industrial automation network.

Keywords Ethernet, CSMA/CD, TCP/IP, UDP, Real-time communication, Industrial automation

1. 서론

생산 자동화 설비를 구축함에 있어서 기기들간의 정보 교환을 위한 실시간 네트웍은 중요한 구성 요소중의 하나이다. 상호 호환성이 없는 이기종간의 의사 소통을 위해서 MAP(Manufacturing Automation Protocol) 등의 CIM을 지향하는 실시간 네트웍에 대한 연구가 활기를 띠고 있기는 하지만 비용문제 때문에 충분한 투자 능력이 있는 기업에서만 실제 생산 설비에 제한적으로 적용하고있는 형편이다. 반면 ethernet은 근거리 통신망에서 가장 널리 사용되는 방식으로 설치비용이 token ring이나 token bus에 비해 저렴하고 관리가 쉽다는 장점으로 사용자가 큰 폭으로 증가하고 있고, TCP/IP는 인터넷의 기하급수적인 수요 증가로 그 중요성이 크게 부각되고 있다. 그러나 적은 비용과 수요의 증가에도 불구하고 생산 자동화 설비에 TCP/IP가 적용되기 힘든 이유는 TCP/IP에서는 데이터 전송 시간보다는 데이터 전송의 정확성과 효율을 중요시하기 때문이다. 그러나 실시간 네트웍에서는 데이터 전송의 정확성 뿐만 아니라 데이터 전송 시간 역시 중요한 요소이기 때문에 데이터 전송 시간에 관한 정확한 검증없이 TCP/IP를 적용해서는 안된다. 이러한 문제점에도 불구하고 적은 비용으로 실시간 네트웍을 구축하고 싶어하는 기업의 욕구가 증가함에따라 본 논문에서는 자동화 설비에 ethernet과 TCP/IP를 적용할 수 있는 새로운 방법을 고찰한다.

IEEE 802.3 에서는 CSMA/CD Access Method 와 물리층 규격을 정의하고 있다. 1985년 이후로 만들어진 ethernet시스템은 모두 이 규격을 준수하고 있고 1994년까지 4천만 개의 노드가 연결될 정도로 가장 널리 사용되는 LAN 시스템이 되었다. 이 처럼 전 세계적으로 널리 보급 되어있는 시스템임에도 불구하고 자동화 생산 설비에 적용하기 힘든 이유는 접속 되어있는 모든 노드에게 미디어를 사용 할 수 있는 기회를 동일하게 주기 때문이다. ethernet에서 사용하는 1-persistent CSMA/CD 프로토콜은 패킷 충돌(collision)에 대한 아무런 보완이 마련되어 있지 않기 때문에 선로상에 과도한 부하가 걸리는 경우 실시간 네트웍이 불가능한 것이 현실이다. 이는 순간적으로 동시에 미디어를 사용하려는 시도로 인한 패킷 충돌(collision)을 야기하고 네트웍 사용량이 조금만 늘어나도 효율이 급격히 떨어지며 반복되는 패킷 충돌(collision)이 16회 이상 지속되면 패킷 전송을 포기하는등 실시간 네트웍 구현을 어렵게 하는 원인이 되고있다. ethernet의 이러한 문제점을 해결 하려는 많은 연구 결과가 발표 되었고 대표적인 것으로는 TDMA(Time Division Multiple Access), BRAM(Broadcast Recognizing Access Method) 등을 들 수있다[1]. 특히 BRAM은 패킷 충돌의 가능성을 완벽하게 제거하고 모든 노드에게 균등한 기회를 제공하며 필요에 따라서 우선순위 기능을 제공할 수 도있는 프로토콜이고 조금 변형된 형태의 MBRAM이 실제 구현되기도 하였다[7]. 그러나 이러한 프로토콜들은 MAC(Medium Access Control) layer수준에서 구현되어야 하기 때문에 ethnernet 장치의 종류에 따라 OS의 네트웍 커널이 수정 되어야하고 네트웍에 접속된 모든 컴퓨터가 동일한 프로토콜을 사용하게끔 장비들을 설치해야하는등 실제 구현하는데 있어서 많은 어려움이 따른다. ethernet 뿐만 아니라 TCP(Transmission Control Protocol)[5]도 실시간 네트웍 구현에 큰 장애가 되고있다. TCP의 sliding window[2][8] 알고리즘은 평균적으로 네트웍 부하가 적은 상황에서도 특정 시간에 과도한 부하를 발생시켜 패킷 전송을 지연시킨다. 그뿐 아니라 패킷의 전송 제어가 커널에서 이루어지기 때문에 유실된 패킷에 대한 재전송 시간을 조정 할 수가 없게되어 패킷의 실시간 전송을 더욱 어렵게한다.

따라서 본 논문에서 새롭게 제시되는 프로토콜은 MAC 수준의 패킷 제어를 대상으로 하지 않고 OSI 7 Layer[10]중 network layer[4]의 상위 레이어, 그중에서도 connectionless 프로토콜인 UDP(User Datagram Protocol) [3]를 이용하여 sliding window와 재전송 시간조절 등과 같이 실시간 통신에 장애가 되는 TCP의 단점을 보안함으로써 공장 자동화 네트웍으로의 적용 가능성을 제시해본다. 본 논문은 2장에서 TCP/IP 프로토콜[2][8]의 문제점에 대해서 다루고 3장에서 새로운 그 문제점을 해결하기 위한 새로운 알고리즘을 제시한다.

2. 실시간 네트웍 구현시 TCP/IP의 문제점

2.1 *CSMA/CD* 분석

하나의 물리적인 통신 회선을 많은 사용자가 공유할 수있게 하려는 노력은 아주 오래전부터 있어왔다. 그 노력의 결과로 ALOHA, CSMA, CSMA/CD, TDMA, MSAP, BRAM 등의 많은 프로토콜이 개발 되었다. 새로운 프로토콜의 성능을 분석하기 위해서 새로운 분석법들을 발표했는데 그중에서 앞으로 네트웍 효율을 언급할 때 사용될 1-persistent CSMA/CD 의 효율에 대한 분석은 Hideaki Takagi 와 Leonard Kleinrock에 의해서 제안 되었다[9]. 그 결과가

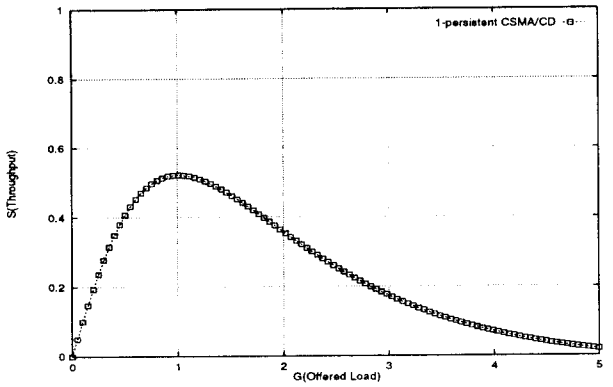

그림 1: 1-persistent CSMA/CD 의 처리량
Fig. 1 Throughput of 1-persistent CSMA/CD.

그림 1에 도시되어있다. 그림에서 가로축은 패킷 전송을 시도하는 비율(G)이고 세로축은 실제 전송된 패킷의 비율(S)을 나타낸다.

2.2 *Sliding Window*의 문제점

TCP/IP에서 사용중인 sliding window의 문제점을 파악하기 위해서 10개의 노드가 접속되어있는 네트웍에서 TCP/IP 프로토콜을 사용한다고 가정해본다. 기존의 TCP/IP 프로토콜을 사용하는 경우에는 그림 2의 A에 짙은색 블록으로 표시된 것 처럼 tcp 세그먼트를 전송하는 노드는 sliding window의 패킷들과 세그먼트에 대한 응답의 연속으로 선로를 최대한 이용하려 할것이다. 이런경우 각 노드에서 선로를 사용하려하는 packet arrival rate(λ)는 1이 되기 때문에 선로에 동시에 데이터를 보내려는 노드가 n개 있는 경우 선로에 유입되는 패킷의 전체적인 arrival rate λ=n 이 된다. 그림 1에서 알 수 있듯이 λ가 1보다 크게 되면 사용 효율은 급격히 감소하게 되고, 그 구간에서는 원하는 시간에 데이터를 전송 할 수없게 되어 실시간 통신에 치명적인 문제가 된다. 예를 들어서 10개의 노드중 우연히 4개의 노드가 데이터를 전송하는 구간이 겹치게된다면 그림 1에서 G=4(λ=1,노드의 갯수=4) 일때 전송되는 패킷의 전송 효율이 7%밖에 안된다는 것을 알 수있다. 여기서 전송 효율이 7%밖에 안되는 경우가 얼마나 자주 발생하는지를 계산 할 필요가 있다. 선로의 평균 패킷 전송 시도 비율을 1 이라하고 4개의 노드가 같은 크기의 sliding window 블록을 전송한다고 가정했을때 다음의 Poisson 분포에서

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{\lambda t}$$

블록 4개가 겹칠 확률 A(t)는 다음과 같다[6].

$$A(t) = 1 - e^{-\lambda t} - \lambda t e^{-\lambda t} - (\lambda t)^2 \frac{e^{-\lambda t}}{2} - (\lambda t)^3 \frac{e^{-\lambda t}}{6}$$

위식에서 λt=1 인 경우에 대해서 계산하면 0.264 가 나온다. 즉, 위와같이 평균 패킷 전송 비율이 1 인 선로에 접속된 4개의 노드가 보낸 sliding window 블록중 26.4%가 동시에 겹치게 된다. 결국 TCP/IP 프로토콜을 실시간 통신에 사용하려 할때 가장 큰 장애가 되는 것이 sliding window 알고리즘임을 알 수있다.

3. Proposed algorithm

3.1 분산 전송 제어

본 논문에서 제시하는 프로토콜의 가장 큰 특징은 네트웍에 연결된 노드들간의 짧은 시간 동안 생길 수있는 경쟁을 예방하는 것으로,각

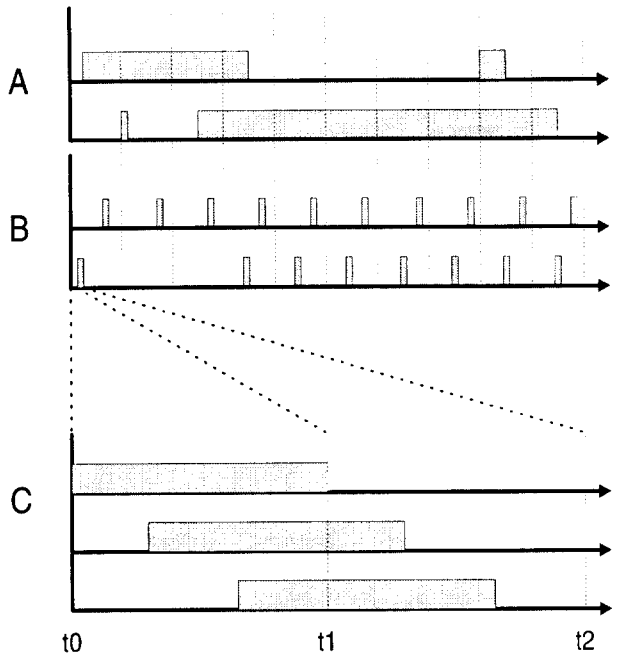

그림 2: sliding window 알고리즘과 패킷 분산방식의 비교
Fig. 2 Comparison of sliding window algorithm and packet distribution.

노드는 데이터를 최대한 분산시켜서 전송 함으로써 sliding window 알고리즘 때문에 생기는 과도한 부하를 방지할 수있다. 그림 2의 B는 전송형태를 그림으로 표시한 것이고 C는 t0-t1 구간에서 패킷이 전송되고 있을때 또 다른 두개의 노드에서 패킷 전송을 시도하여 충돌(collision)이 일어나는 경우를 표시한 것이다. CSMA/CD에서 충돌(collision)은 대부분 하나의 패킷이 전송되는 중간에 두개 이상의 노드가 전송 시도를 하는 경우에 발생한다. 간단히 (a,b) 2개의 노드가 전송 시도를 하는경우를 생각해보자. 각 노드는 현재 선로상에 캐리어가 있는 것을 확인한 후 선로가 휴지 상태가 되기를 기다게된다. 선로상에 캐리어가 없어지면 노드 a,b는 선로의 휴지 상태를 감지 하자마자 데이터 전송을 시도하게되는데, 아주 짧은 시간동안 a가 먼저 데이터를 전송하였다 하더라도 케이블의 저항과 축전지 성분때문에 propagation delay가 발생하기 때문에 b에서는 a가 발생시킨 캐리어를 감지 하지 못하고 자신의 데이터를 전송 하게된다. 결국 이 순간 패킷 충돌(collision)이 발생하게 되고 두개의 노드는 임의에 시간 이후에 다시 전송을 시도하게된다. 이처럼 충돌 방식은 비슷하지만 분산 전송인 경우 Poisson 분포가 적용 될 수 없다. 패킷 전송 시작 시간은 Poisson 분포로 표현될 수있지만 연속되는 패킷 전송은 첫번째 전송 시간에 영향을 받아서 memoryless 특성을 만족시키지 못하기 때문이다. 또, 패킷 전송 간격이 정해져 있기 때문에 하나의 노드에서 패킷을 전송하는 간격이 exponetial 분포를 갖지 않고 불연속으로 분포하기 때문에 더더욱 Poisson분포와 거리가 멀다고 할 수있다. 이러한 이유 때문에 의도적인 패킷 분산 전송의 경우에는 Hideaki Takagi 와 Leonard Kleinrock의 1-persistent CSMA/CD 효율 분석 결과를 적용하기는 힘들다. 그렇지만, 앞서 언급한 TCP sliding window블록에서처럼 블록의 겹침으로 인한 선로의 과도한 부하 때문에 패킷 전송이 심하게 지연되는 상황은 발생하지 않는다는 것은 쉽게 알 수있다. 그림 3에 분산 전송을 위한 알고리즘을 제시하였다. 알고리즘에서 알 수있듯이 단일 패킷으로 전송가능한 데이터와 분산 전송이 필요한 데이터는 다음 장에서 언급될 응답 계획과 scheduling을 위해서 다르게 처리되어야만 한다.

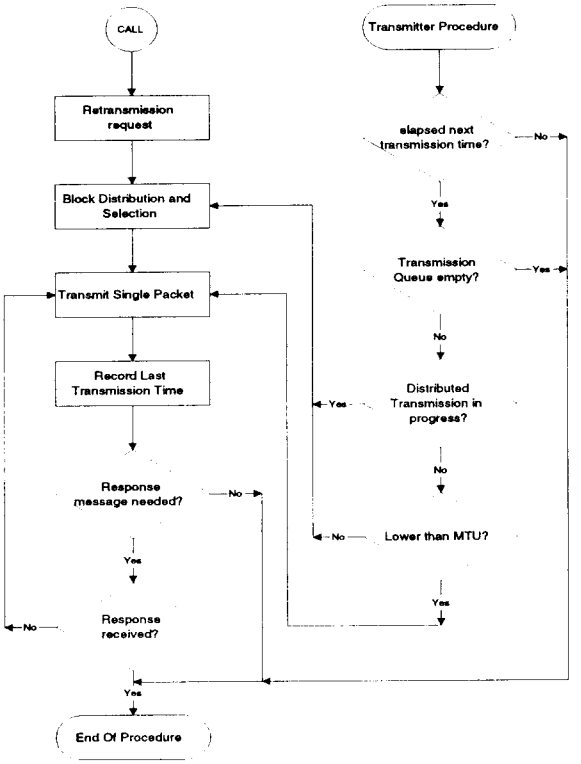

그림 3: 분산 전송 알고리즘
Fig. 3 Distributed Transmission Algorithm

3.2 응답 계획

TCP 에서는 최근에 보낸 패킷이 수신되었음을 알리는 응답을 송신측에 보내주고 일정 시간동안 보낸 데이터에 대한 응답이 없을 경우에는 패킷을 재 전송하는데 이 재전송 시간은 사용자가 바꿀 수 없고 대상 호스트의 접속 상태에 따라서 변할 수 있게 되어있다. 실시간 네트웍에서 재전송 시간은 충분이 짧아야하고, 한계 시간 결정은 전체 시스템의 특징에 따라 변경 되어야 한다. sliding window 알고리즘의 경우 연속된 세그먼트의 모든 패킷에 대해서 응답을 보내는 것은 아니다. 처음에 sliding window가 제시된 이유는 WAN에서의 전송 효율을 높이기 위한 것이었기때문에 연속적인 세그먼트에 대해서 중간중간 확인의 의미로 응답을 보내는 형식을 지원하게끔 정해 놓았다. 보통 응답 신호의 데이터 길이는 40바이트로 20바이트는 IP header , 나머지 20바이트는 TCP header로 구성된다. 결국 응답 신호로써 전송 가능한 가장 작은 크기의 패킷을 전송하는 것이다. CSMA/CD에서는 패킷의 길이가 짧을 수록 패킷 충돌(collision)이 발생할 가능성이 높아지며 특히, sliding window 블럭과 함께 전송되는 응답 신호열은 선로의 효율을 더욱 낮아지게한다. 따라서 대량의 데이터 전송시 응답 신호 전송은 최대한 억제 해야만한다. 단일 패킷으로 송 수신 가능한 데이터의 경우에는 패킷마다 반드시 응답이 있어야 하겠지만 대량의 데이터 전송시에는 최초의 패킷과 최종 패킷에 대한 응답만 전송하고 나머지 패킷에 대해서는 재전송 신호를 보냄으로써 응답 메세지를 줄일 수있다. TCP 는 기본적으로 Internet을 대상으로 하기 때문에 세그먼트 재 전송에 관한 요구가 없지만 분산 전송 제어는 자동화 시스템이 구축된 근거리 통신망을 대상으로 하기때문에 에러가 발생한 세그먼트에 대한 재전송 요구를 해야만한다. 보통 수신된 패킷에서 에러가 발생한 경우는 극히 드물기 때문에 이러한 방법은 선로 부하를 대폭 줄일 수있으나 대량의 데이터를 수신할때 에러가 발생할지도 모르는 패킷을 감안하여

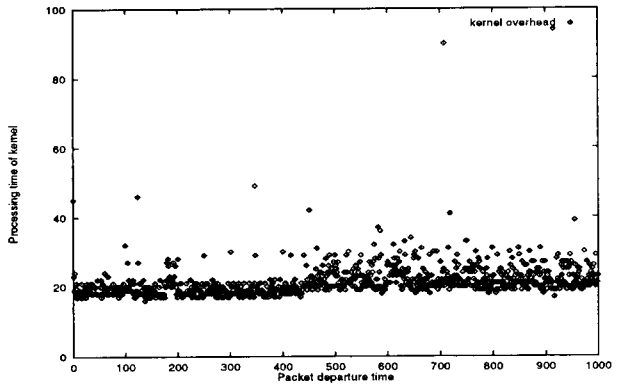

그림 4: UDP 패킷을 처리하는 커널에서의 overhead 분산
Fig. 4 Kernel overhead of UDP packet

프로그램을 만들게 되면 구조가 다소 복잡해지지만 BSD의 mbuf가 좋은 해결 책이 될 수있다.

Multi-user 운영체제에서 네트웍 관련 데이터를 처리할때 다양한 크기의 데이터를 효율적으로 관리해야하는 문제에 직면하게된다. 대표적인 버퍼 관리 문제는 IP datagram의 reassembly 과정에서 발생한다. 수신되고있는 데이터의 최종적인 크기를 알 수없고 전송 순서도 정해져있지가 않기 때문에 다양한 크기의 데이터를 처리 할 수있는 버퍼의 구현이 반드시 필요한 부분이기도하다. 이 문제를 해결하기 위한 많은 방법들이 있는데, 그중에서 Berkeley UNIX에서 사용되는 mbuf가 잘 알려져있다. mbuf는 다양한 크기의 데이터를 다루기 위해서 linked-list 형태의 버퍼이다. mbuf를 구성하는 linked-list 의 블록들은 고정된 크기를 가지고있고, 사용자가 이 블럭의 갯수를 변화시켜서 능동적인 메모리 관리를 할 수있게 해주는 구조로 되어있다. 보통 각 블록의 크기는 128byte에서 1Kbyte 이며 필요에 따라서 비교적 큰 데이터를 수용하기 위해 2Kbyte의 2차 버퍼를 만들어 사용하기도한다. 위에 설명한 알고리즘을 구현할때는 패킷의 크기와 전체 데이터의 크기를 알고 있기 때문에 mbuf를 그대로 사용하게되면 효율이 떨어지므로 header의 크기를 줄이고 버퍼 블록의 크기를 ethernet의 MTU(Maximum Transfer Unit)인 1500(header 제외한 데이터 크기는 1460byte)byte에 맞추어 버퍼를 설계했다.

3.3 *Scheduling*

Transport layer수준에서 datagram을 조작 하는 방법으로는 MAC 수준의 스케쥴링이 불가능하다. Network layer 와 Datalink layer를 거치는 동안에 생기는 불규칙한 시간 지연 때문이다. 그림 4은 SunOS 5.4에서 시스템 부하를 최소화 시킨 상태에서 커널의 시간별 UDP 패킷의 응답 시간의 분포이다. 그림으로 알 수있듯이 응답 시간이 불규칙 하기 때문에 스케쥴링이 안된다. 스케쥴링을 실행하는 프로세스 자체가 스케쥴링되기 때문에 패킷 충돌(collision) 방지를 목적으로 하는 transport layer 수준의 스케쥴링은 전혀 의미가 없다. 단지 선로 사용을 최대한 분산 시키려는 노력이 의미가 있을 뿐이다. 본 논문에서 제안하는 프로토콜은 선로상에 과도한 부하가 걸린 상태에서 우선 순위가 높은 데이터를 정해진 시간 안에 전송할 수있는 방법을 제시 할 수가 없는데 그 이유는 실제 패킷 전송이 MAC 에서 이루어지기 때문이다. 우선 생각해야 할 사항은 선로를 이용하는 노드들의 최대 전송 시도 비율이 1을 넘어서는 안된다는 것이다. 네트웍 상황에 따라 그 값은 달라지겠지만 가능한한 최소한의 크기로 결정되는 것이 좋다. 네트웍 부하가 적을 수록 패킷의 전송 지연이 급격히 줄어들기 때문이다. 네트웍을 담당하는 프로세

스가 시작되기 전에 각 노드는 자신의 최대 전송 속도를 결정 하고 최근에 전송한 시간을 기록해 두어야한다.

4. 결론

우리는 이 논문을 통해서 실시간 네트웍을 구현하는데 TCP/IP가 적합하지 않는 이유에 대해서 알아보았고 분산 전송 제어라는 새로운 방법을 도입하여 기존의 자동화 시스템의 실시간 네트웍의 가능성을 제시하였다. 분산 전송 제어는 선로의 휴지기간에도 항상 일정한 데이터 전송률을 유지하도록하여 일반적인 근거리 통신망 보다는 공장 자동화를 위한 실시간 통신을 목적으로 설계되었다. 그러나 제안된 분산 전송 제어 방식은 패킷 전송 스케쥴링이 필요하므로 UNIX와 같은 범용 운영체제 보다는 실시간 운영체제등을 이용한다면 비교적 정밀한 스케쥴링을 통해 분산 전송 제어의 성능을 극대화 시킬 수있을 것이다

5. 참고서적

[1] I. Chlamtac, W. R. Franta, and K. D. Levin, "Bram: The broadcast recognizing access method," *IEEE Transactions on Communications*, vol. 27, no. 8, pp. 1183–1189, August 1979.

[2] D. Comer, *Internetworking with TCP/IP*, Prentice Hall, Inc, Englewood Cliffs, New Jersey, 1995.

[3] *User Datagram Protocol*, RFC 791, 1980.

[4] *Internet Protocol*, RFC 791, 1990.

[5] *Transmission Control Protocol*, RFC 793, 1990.

[6] R. Rom and M. Sidi, *Multiple Access Protocols*, Springer-Verlag, New York, 1990.

[7] R. P. Signorile, "Mbram-a priority protocol for pc based local area networks," *IEEE Network*, vol. 2, no. 4, pp. 55–59, July 1988.

[8] W. R. Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley Publishing Company, Massachusetts, 1994.

[9] H. TAKAGI and L. KLEINROCK, "Throughput analysis for persistent csma systems," *IEEE Transaction on communications*, vol. 33, no. 7, pp. 627–637, 1985.

[10] A. S. Tanenbaum, *Computer Networks*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1989.