한국군사과학기술학회지 제16권 제5호, pp. 659~665, 2013년 10월

학술논문 전자 · 통신 부문

함정전투시스템 미들웨어 IPCS의 성능 개선

Performance Improvement of IPCS : A Middleware for Warship Combat Systems

류 원 재* 신 수 용* 허 성 길** 최 윤 석**
Won-Jae Ryu Soo-Young Shin Seong-gil Heo Yoon-Seok Choi

ABSTRACT

IPCS(Inter Process Communication Service) is a real-time communication middleware designed for warship combat battle systems based on publisher-subscriber communication model. Because IPCS was originally designed to operated under 100 Mbps network environment, increasing network speed into Gigabit environment does not linearly increase the throughput of IPCS. To solve this problem, we anaylized IPCS structure and optimized IPCS into Gigabit-Ethernet environment. We found parameters to improve IPCS based on UDP and Token-ring structure. By improving, IPCS has reliability and higher throughput than TCP although IPCS is based on UDP.

Keywords : Real Time(실시간), Communication Middleware(통신 미들웨어), Warship Combat System(전투 함정 시스템), Throughput(처리량)

1. 서 론

함정전투시스템은 다양한 적의 위협에 대해 정확한 탐지 및 대응 능력과 동시 다발적인 전투상황 하에서 함 탑재센서와 외부로부터 획득된 정보를 종합 처리하여 최적의 전투능력을 제공할 수 있도록 지휘 및 무장을 통제하는 기능이 통합된 체계이다. 즉 전투체계는 공중, 수상 및 수중으로 부터 적의 위협을 조기에 탐지 및 추적하는 센서체계, 함내 외부 센서로부터 수집된 표적정보를 실시간으로 처리 및 종합하여 지휘관의 의사결정을 지원하는 전투관리체계(CMS : Combat Management System), 그리고 교전을 수행하는 다양한 무기체계 등으로 구성된다[1]. 이러한 함정 전투 시스템은 교환기, 인터컴 단말기, 네트워크 관리 장치 등으로 구성되며, 단말기간의 일대일 통신, 회의망의 구성 등을 통한 그룹통신 및 다양한 통신관련 장비와의 연동을 통해 연동장비의 원격운용 등을 할 수 있다[2,3].

IPCS는 이러한 전투 함정 시스템에서 사용되는 실시간 통신 미들웨어(Inter Process Communication Service)로, 삼성탈레스에서 UDP를 기반으로 발행자-구독자 통신 서비스를 구현한 것이다[4].

통신 미들웨어는 클라이언트, 서버 사이의 통신에 대한 코딩이 쓸데없이 복잡해지지 않도록, 구현이 간단하고 한 방식을 일관되게 제공한다. 또한 서버 프로그램이 살아있는지 여부를 감시하는 프로세스 관리자

† 2013년 4월 22일 접수~2013년 8월 16일 게재승인
* 금오공과대학교(Kumoh National Institute of Technology)
** 삼성탈레스(Samsung Thales)
책임저자 : 신수용(wdragon@kumoh.ac.kr)

로서의 역할을 하며, 서버의 규모가 커지는 경우 서버 프로세서의 현황을 집중해서 관리 가능하다. 또한 복잡한 개별 장비의 사양이나 인터페이스 방법을 모르더라도 쉽게 연결이 가능하고, 미들웨어의 계층 구조에 따라 데이터를 일정 기간 보관하고 필요에 따라 외부 어플리케이션에 이를 제공할 수 있다. 또한 시스템의 추가 및 수정, 비즈니스 프로세스의 추가 및 변경 시에도 전체 시스템에 위험을 주지 않고 용이한 처리가 가능한 장점들이 있다[5].

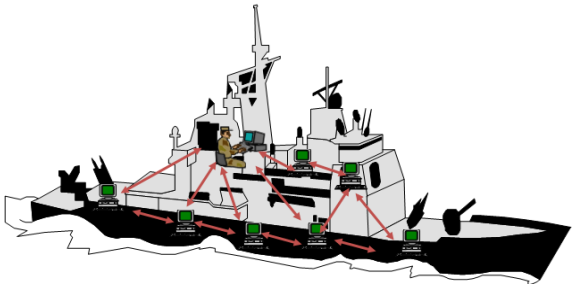

Fig. 1. 함정 전투 시스템 개념도

IPCS는 네트워크 상에서 높은 성능의 메시지 서비스(High Performance Messaging Service)를 제공하여 응용 프로그램 사이에서 신뢰성 있는 통신을 가능하게 한다. 통신 실패나 네트워크 분할에 대하여 뛰어난 감지 능력을 가지고 있으며 메시지에 대한 주문 서비스 및 전송을 보장한다.

IPCS는 발행자-구독자의 통신 모델을 기반으로 구현되었다. 발행자-구독자는 메시지 기반의 비연결 지향 서비스로, 서비스에 대한 정보에 관계없이 데이터에 대한 접근을 가능하게 해주어 실시간에 적합한 통신 환경을 제공한다. 또한 일대일 방식과 달리 한번에 일대다 전송이 가능하다[6].

IPCS에서는 네트워크 상에 있는 각 노드와 응용 프로그램을 멤버십 서비스를 통하여 관리하고, 단체 통신을 통하여 메시지가 효율적으로 전달할 수 있게 해준다[7].

IPCS는 기본 프로토콜인 UDP 기반으로 설계된 미들웨어지만 TCP와 같이 신뢰성을 제공할 수 있도록 설계되었다.

그러나 기존의 IPCS의 성능은 100메가급의 네트워크 환경을 바탕으로 설계되었기에 현재 IPCS를 적용하려하는 무인잠수정의 네트워크 환경은 기가급에는 적합하지 못하다. 따라서 100메가급 네트워크 환경에서 설계된 IPCS가 올바른 성능을 끌어낼 수 없다는 문제점을 가지고 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 기가급 네트워크에서의 IPCS 성능 개선을 수행하고자 한다.

Fig. 2. IPCS를 사용하는 무인잠수정

다음 장에서는 IPCS의 구조와 특징에 대해서 서술하고, 3장에서는 IPCS가 현재 사용되는 기가급 네트워크에서 낮은 성능을 보이는 문제점을 설명하고, 4장에서는 서술된 내용을 바탕으로 성능 평가 구현 방법을 위한 실험 정의 및 실험범위를 정하고 실험결과를 다룰 것이며 5장에서 결론으로 끝을 맺는다.

2. IPCS의 구조 및 특징

IPCS는 네트워크 메시징 미들웨어로 응용 프로그램을 컴퓨터 구조, 운영 체제, 네트워크 등과 분리시키는 역할을 한다. 응용 프로그램이 소켓과 TCP/IP같은 하위 레벨의 프로토콜을 사용하지 않고 데이터를 주고받을 수 있게 함으로서 분산 시스템의 구성을 쉽게 할 수 있도록 해준다. 이런 점은 사용자가 응용 프로그램을 용이하게 구현할 수 있고, 특별한 응용 코드의 수정 없이도 이기종간의 통신을 가능하게 해준다[8].

IPCS는 크게 대몬과 사용자로 구분되며, 이 두 요소는 동일한 노드에 존재한다.

가. 대몬[9]

패킷의 송수신을 담당하고 자신에게 접속한 사용자를 관리하며, 대몬들과의 상호작용을 담당하는 일종의 서버 역할을 한다. 다른 대몬과의 통신 및 토큰 제어

를 담당하는 내부-대몬 인터페이스와 사용자와의 통신을 담당하는 사용자 인터페이스가 있다.

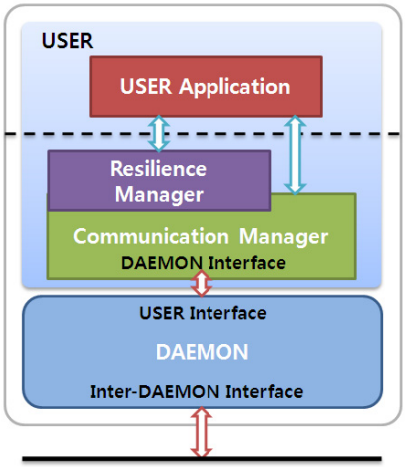

Fig. 3. IPCS 구성 요소

1) 대몬의 구조

Fig. 4에서 보여주듯이 대몬의 내부는 기능적으로 5개의 요소로 나눌 수 있다. 세션 관리(Session Management)와 그룹 관리(Group Management)는 대몬에 접속한 사용자에 대한 정보를 저장하고 그룹 단위로 관리한다. 패킷 제어는 다운큐(Down-Queue), 업큐(Up_Queue), 패킷데이터베이스(Packets_DB)를 사용하여 대몬간에 전송되는 패킷을 관리하며, 내부-대몬 인터페이스와 사용자 인터페이스는 각각 다른 대몬이나 사용자들과의 통신을 담당한다.

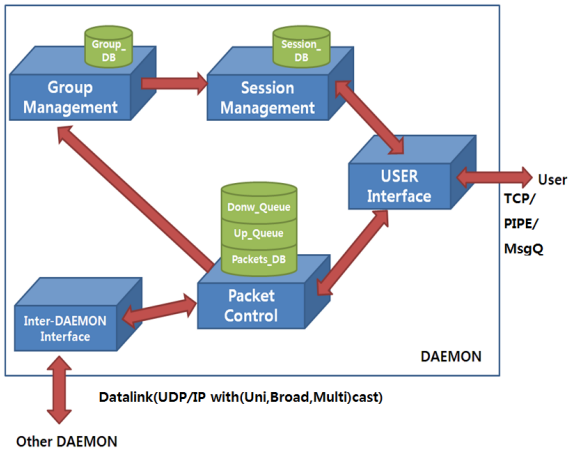

Fig. 4. 대몬 구조

2) 대몬의 서비스

IPCS 대몬은 IPCS망을 안정되게 관리하고, 패킷에 대한 신뢰성을 보장하기 위하여 멤버십 서비스 및 가상 토큰 링 프로토콜 그리고 패킷 손실에 대한 복구 서비스를 제공한다.

또한 IPCS는 신뢰성 있는 브로드캐스팅 및 멀티캐스팅을 지원함으로써 분산환경에서 응용 프로그램간 신뢰성 있는 데이터의 전송을 보장한다.

- 토큰[10] : IPCS에서는 가상 토큰 링 프로토콜을 적용하여 신뢰성 있는 통신을 보장한다. 가상 토큰 링 프로토콜은 대몬들 사이에 토큰을 차례대로 순환시키고, 토큰을 가진 대몬만이 패킷을 IPCS망 내에 브로드캐스팅할 수 있다. 대몬은 토큰 정보를 통하여 패킷의 송수신 완료 여부, 대몬의 실패 또는 재 시작 여부, 패킷의 재전송 여부 등을 알 수 있게 된다.

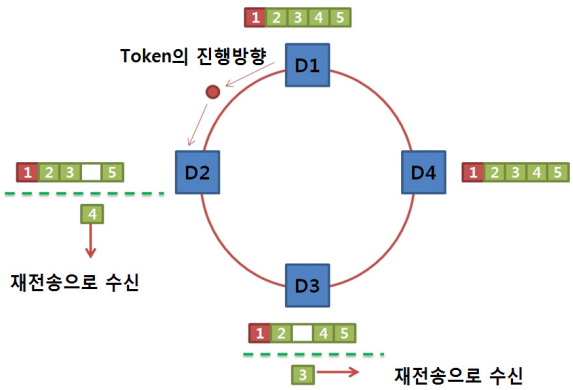

Fig. 5. Retransmission flow

- 재전송 플로우 : IPCS에서는 대몬 간에 패킷 손실에 대하여 재전송을 해준다. 다음은 IPCS의 재전송 동작에 대한 동작 예시이다.
  ① D1이 토큰을 소유했을 때 패킷 1를 브로드캐스팅하고 D2로 토큰을 전송한다. 다른 대몬들은 모두 패킷 1을 정상적으로 수신하여 패킷데이터베이스에 저장하였다. 패킷데이터베이스에 저장할 때는 패킷의 열 번호에 따라 저장된다.
  ② 토큰을 수신한 D2는 패킷 2와 패킷 3을 브로드캐스팅하고 D3으로 토큰을 전송한다. D3을 제외한 다른 대몬들은 모두 패킷 2와 3을 정상적으로 수신하여 패킷데이터베이스에 저장하였

지만 D3은 패킷 3을 정상적으로 수신하지 못하였다.

③ 토큰을 수신한 D3은 패킷 4를 브로드캐스팅하고 자신이 패킷 3을 수신하지 못하였기 때문에 패킷 3에 대한 재전송 요청이 필요하다. D2를 제외한 다른 대몬들은 패킷 4를 정상적으로 수신하여 패킷 데이터베이스에 저장하였지만 D2는 패킷 4를 수신하지 못하였다. 패킷 3에 대한 재전송 요청 정보가 포함된 토큰을 D4로 전송한다.

④ 토큰을 수신한 D4는 재전송 요청이 있으므로 자신의 패킷 데이터베이스를 확인하여 패킷 3을 D3으로만 전송한다. 그리고 자신의 새로운 패킷 5를 브로드캐스팅한다. 모든 대몬은 패킷 5를 수신하여 패킷 데이터베이스에 저장하였다. 토큰을 D1로 전송한다. 만약 D4자신도 패킷 3을 수신하지 못하였다면 이러한 정보를 토큰에 포함시켜 D1로 전송하고 D1은 패킷 3을 수신하지 못한 대몬이 한 개 이상이라는 것을 알고 패킷 3을 모든 대몬으로 브로드캐스팅할 것이다(이미 패킷 3을 수신한 대몬들은 이 패킷을 무시한다.).

나. 사용자

사용자는 통신 관리자(Communication Manager)와 이중화 관리자(Resilience Manager) 그리고 사용자 어플리케이션으로 구성되어 있다. 여기서 사용자 어플리케이션은 IPCS API를 사용해서 사용자가 구현한 응용프로그램을 말한다. 사용자는 대몬 인터페이스를 통하여 대몬과 통신한다.

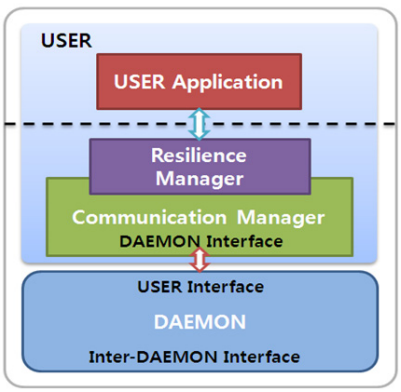

Fig. 6. USER의 구성

- 통신 관리자 : 대몬과의 통신을 담당하며 TCP/IP, PIPE, MsgQ 등을 사용하여 대몬과 통신한다. 통신 관리자는 내부적으로 두 개의 쓰레드가 작동한다. 수신 프로세스는 메시지를 수신하여 이중화 관리자나 사용자 어플리케이션으로 전달하는 기능을 수행한다. 그리고 생존 프로세스(Alive Process)는 사용자 어플리케이션이 정상적으로 동작하고 있음을 주기적으로 대몬에게 보고하는 기능을 수행한다.
- 이중화 관리자 : 사용자 어플리케이션은 IPCS망 내에서 이중화 구성 요소 및 비이중화 구성 요소로 동작 할 수 있다. 사용자 어플리케이션이 이중화 구성요소로 동작 할 때 이중화 관리자는 활성/대기 모드 결정, 이중화 관련 데이터의 처리, 활성과 대기 모드간 이중화 데이터 동기화, 서버 다운 발생 시 활성모드에서 처리를 완료하지 못한 이중화 데이터에 대한 처리완료 등을 담당한다.
- 사용자 어플리케이션 : IPCS에서 제공하는 API를 이용하여 개발자가 구현한 응용프로그램이다.
- 대몬 인터페이스 : 사용자와의 통신을 담당하며 대몬을 거쳐 다른 사용자들과 통신할 수 있게 해준다. IPCS에서 사용자가 대몬과 통신하는 방법에는 3가지가 존재하는데 Socket, Windows pipe, MsgQ이다. Socket은 IPCS의 시스템 환경에 무관하게 사용할 수 있고 또한 대몬과 사용자가 동일한 장치에 존재하거나 원격 장치에 존재하거나 모두 사용 가능하다. 반면에 Windows pipe는 Windows환경에서 MsgQ는 VxWorks환경에서만 사용할 수 있고 반드시 대몬과 사용자가 동일 장치에 존재하여야 한다.

3. Gigabit Ethernet하에서의 IPCS 성능 개선 필요성

IPCS는 UDP 기반의 미들웨어로써 기본 프로토콜에 비해 구현 및 사용이 간단하다는 장점을 가지고 있다. 또한 UDP가 가지지 못한 신뢰성을 가졌기 때문에 TCP와 같이 100% 전송률을 보장한다. 하지만 기가급 이더넷에서의 실험 결과 기본 프로토콜 TCP에 비하여 처리량이 떨어지는 단점을 가지고 있다. 1:1 상황에서 메시지 크기가 500바이트일 때만 IPCS가 TCP와 같은 결과를 가지고 그 외의 경우에는 IPCS의 성능이 전반적으로 낮았다. 이는 IPCS는 기존 100메가급 이더넷 환경에서 설계되어 있기 때문이다. 나머지 경우

에서는 IPCS가 TCP에 비해서 낮은 처리량을 가지는 것을 알 수 있다. 따라서 본 논문에서는 IPCS가 TCP보다 나은 성능을 가질 수 있도록, 성능 개선 파라미터 값을 찾는 것에 중점을 두었다.

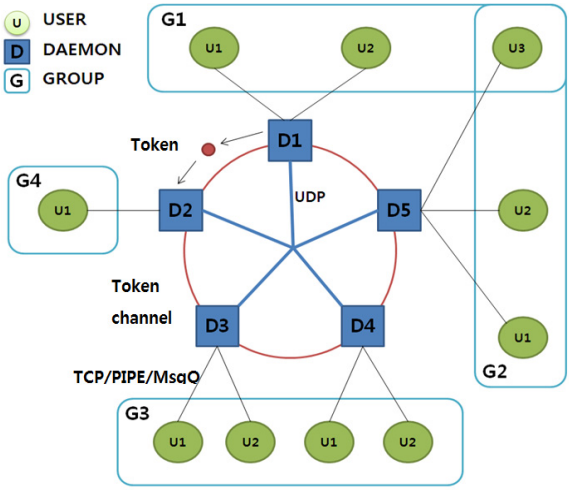

Fig. 7. IPCS 시스템 구성

4. 실 험

가. 실험 방법

메시지 크기에 따른 5가지 케이스(100, 500, 1000 bytes, 5000, 10000바이트)와 노드수에 따른 4가지 케이스(1:1, 1:2, 1:4, 1:8)로 총 20가지의 경우로 테스트 하였다.

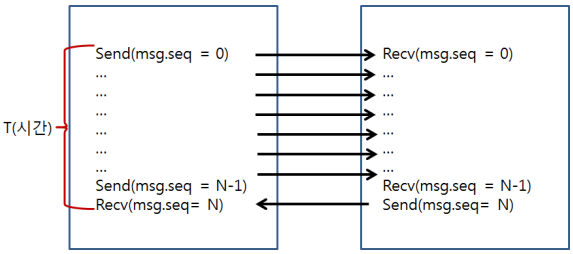

Fig. 8. 실험방법

반복수는 총 80메가비트가 되도록 하였다(예를 들어, 100바이트는 800비트이므로 100000번, 500바이트는 20000번 반복 전송). 이를 또 다시 30번 반복하여 평균 값을 통해 처리량을 측정하였다.

Table 1. 테스트 노드 환경

Intel® coreTM i7 CPU 870@ 2.93Ghz
RAM 4.00GB
Windows 7 professional K 32bit.
1Gbit Ethernet.

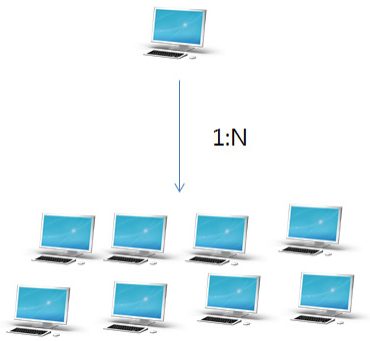

Fig. 9. 테스트 구성도
(1개의 발행 : N개의 구독)

또한 IPCS와 기본 프로토콜을 비교를 하였다. IPCS는 UDP 기반의 미들웨어이다. 이 때문에 UDP와 비교가 좀 더 IPCS의 성능을 확인하기에 좋겠지만, UDP는 신뢰성을 보장하지 않기 때문에 테스트에 적합하지 못하다. 따라서 다른 기본 프로토콜인 TCP와의 비교를 통해 IPCS의 성능을 측정하도록 하였다.

IPCS의 성능을 개선시키기 위하여 FastCopyReceive Threshold, FastSendDatagramThreshold, 전체 윈도우, 개인 윈도우라는 4가지 파라미터를 선택하였다. FastCopy ReceiveThreshold, FastSendDatagramThreshold는 UDP 관련 파라미터이다. IPCS는 UDP 기반 미들웨어이기 때문에 IPCS의 성능에 영향을 미친다. 또한 전체 윈도우 및 개인 윈도우는 IPCS 네트워크 상에 각 데몬들이 전송할 수 있는 메시지의 양을 나타낸다. 따라서 멀티캐스팅이나 브로드캐스팅인 경우 IPCS 성능에 영향을 미친다.

나. 실험 결과

기가급 이더넷에 적합한 파라미터 값을 찾기 위해 총 4가지 파라미터를 수정했다. 전체윈도우 크기, 개인 윈도우 크기, FastCopyReceiveThreshold, FastSend DatagramThreshold 등을 수정하였다.

Fig. 10. 실제 테스트 베드

전체 윈도우는 전체 노드에서 한번에 전송될 수 있는 메시지의 수이고, 개인 윈도우는 한 노드당 최대 전송 가능한 메시지의 수이다.

Table 2. 파라미터

파라미터	기본값	수정값
Window	60	300
Personal window	15	100
FastCopyReceiveThreshold	1024	1500
FastSendDatagramThreshold	1024	1500

전체 윈도우, 개인윈도우 크기를 조정하였을 때, 노드 수가 늘어나는 경우 처리량이 떨어지는 현상을 최소화 할 수 있었다. 기존 값은 100메가급 환경에 맞게 설정되어 있어서, 최대한 보낼 수 있는 메시지에 훨씬 못 미치게 한계 값이 되어 있었기 때문에 노드 수에 따른 처리량의 감소가 있었다. 전체 윈도우 크기가 300보다 작으면, IPCS 네트워크 상에 존재하는 메시지들의 수가 적어서, 최대한의 성능을 끌어내지 못한다. 또한 300보다 클 시에는 네트워크 상의 메시지가 한 번에 처리 가능한 메시지 양보다 많아서 처리량이 떨어진다. 또한 UDP 관련 레지스트리(FastCopyReceive Threshold, FastSendDatagramThreshold)[11]를 수정하였을 때, 메시지 크기가 1024이상 되었을 때 처리량이 떨어지는 현상을 없앨 수 있었다. IPCS는 패킷이 커지면 1440바이트씩 잘라서 전송한다. 하지만 UDP관련 레지스트리 값들이 1024바이트로 되어 있기 때문에 1440바이트는 빠른 I/O[12]가 아닌 대기 상태가 많은 느린 I/O 처리과정을 거치게 되어 속도가 떨어진다. 따라서 1500바이트로 수정하였을 때 IPCS의 메시지 단위 1440바이트를 빠른 I/O로 전송 가능해 진다. 그러므로 크기가 커져도 IPCS의 성능이 떨어지지 않았다.

5. 결 론

IPCS는 전투 함정 시스템에서 사용되는 실시간 통신 미들웨어로 발행자-구독자 통신 모델을 기반으로 구현되었고, UDP 기반으로 설계된 미들웨어로, TCP와 같이 신뢰성을 제공하는 것이 특징이다.

본 논문에서는 기존의 IPCS가 100메가급 환경에 서 설계되었기 때문에 기가급 환경에 적용시 발생하는 성능 저하 문제를 인식하고, IPCS가 기가급 환경에서 향상된 성능을 발휘할 수 있도록 성능 개선을 하는 작업을 수행하였다. 즉 전체 윈도우 크기 및 개인 윈도우 크기와 FastCopyReceiveThreshold, FastSendDatagram Threshold를 수정하여 성능을 개선 시켰다. 또한 개선된 성능 평가를 수행하기 위해 실제 기가급 통신망 환경을 구축하고 구축된 테스트베드상에서 IPCS의 성능을 기본 프로토콜인 TCP와 비교하여 측정 및 평가하였다.

이를 통해 기가급 통신망 환경에서의 성능개선이 이루어진 IPCS가 기존의 100메가급을 바탕으로 설계된 IPCS와 기본 프로토콜인 TCP보다 더 나은 성능을 내는 것을 관찰할 수 있었다. 때문에 고화질의 비디오 데이터 등과 같은 경우에도 보다 안정적으로 전송 가능하다. 따라서 신뢰성을 보장, 발행자-구독자 구조의 장점을 지닌 동시에 TCP보다 높은 성능을 가진 IPCS는 향후 함정 전투 체계의 통신망 시스템으로 활용도가 높다고 할 수 있다.

후 기

이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임. (No.2012R1A1A1009442).

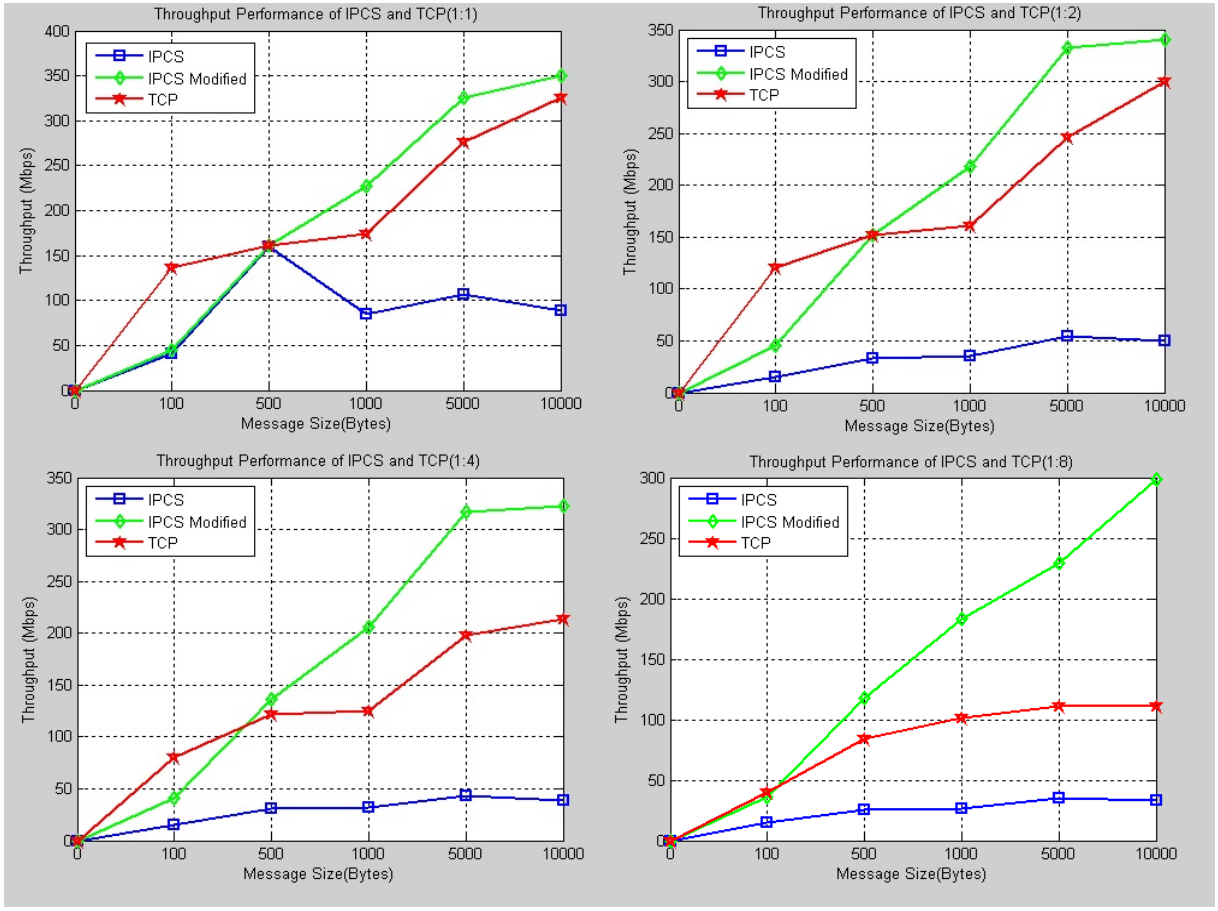

Fig. 11. IPCS, 수정된 IPCS 및 TCP 실험 결과

References

[1] 고순주, "네트워크 중심전을 대비한 전투체계 개발 및 발전방향", 대한전자공학회, 전자공학회지, 제37권 제11호, pp. 27~38, 2010년 11월.

[2] 송경섭, 임영규, 김동성, "함정 전투시스템 정보망을 위한 최적 전송 및 분석 기법", 대한전자공학회 하계종합학술대회, pp. 1668~1671, 2012년 6월.

[3] 이채동, 신우섭, 김석찬, "함정용 통합통신체계의 적용현황 및 발전방향", 한국마린엔지니어링학회지 제34권 제1호, 2010년 1월.

[4] Jimmy Kjallman, "Attachment to a Native Publish/Subscribe Network", Publisher-subscribe Internet Routing Paradigm, July 2009.

[5] 전형국, 이수형, 김원태, 김경태, 박승민, "DDS 미들웨어 표준 기술 동향", 정보통신산업진흥원 주간기술동향 통권, 2010년 7월 28일

[6] http://msdn.microsoft.com/en-us/library/ff649664.aspx

[7] 김영은, 최윤석, 허성길, "IPCS User's Guide", 삼성 탈레스, 2011년 12월 23일.

[8] 석진원, 유인태, 나원식, "실시간 분산 시뮬레이션 시스템을 위한 통신 미들웨어의 구현 및 평가", 한국지식정보기술학회 논문지 제5권 제5호, 2010년 10월.

[9] http://wiki.kldp.org/wiki.php/daemon

[10] http://en.wikipedia.org/wiki/Token_ring

[11] http://www.microsoft.com/korea/technet/network/tcpip2k.mspx

[12] Khealin, "Cache Manager와 File System Driver와의 관계", Devguru 칼럼, 2005년.