# 함정 전투체계 네트워크의 실시간성 검증툴의 설계 및 구현

신희재[1], 권영준[2], 이재호[3], 유재학[4], 이재민[5], 김동성*

금오공과대학교 IT융복합공학과[1,2,5,*], 한화시스템[3,4]

{shinheejae[1], enuj_young[2], ljmpaul[5], dskim*}@kumoh.ac.kr, {jh8028.lee[3], jahak90[4]}@hanwha.com

# Design and Implementation of Real-time Verification Tool for Naval Ship Combat Network

Hee-Jae Shin[1], Young-June Kwon[2], Jae-Ho Lee[3], Jae-Hak Yu[4], Jae-Min Lee[5], Dong-Seong Kim*

Kumoh National Institute of Technology Dept. of IT Convergence Eng.[1,2,5,*], Hanwha System[3,4]

요 약

함정 전투체계는 다양한 체제들이 혼합되어 운용되고 있으며, 이는 함정 네트워크를 통해 연결되어 있다. 함정 전투체계 내에는 영상분배장치(RTVDU, Radar Tv Video Distributor Unit)와 같은 실시간성이 중요한 체계가 다수 있다. 이 중 영상 관련 애플리케이션의 품질은 지연시간뿐 아니라 지연시간의 변이에 의해서도 영향을 받는다. 따라서 함정 네트워크의 실시간 통신 성능 평가를 목적으로, 직접적인 네트워크 지연시간인 Latency와 지연시간의 변화량인 Jitter를 구하는 것이 적합하다. 본 논문에서는, NS3 로그 분석 도구로써 노드에 대한 송수신 Throughput 및 Goodput에 더해 시나리오에 대한 Latency와 Jitter를 출력하는 도구를 제안하고, 해당 지표를 통해 함정 네트워크의 실시간 통신 성능 분석이 가능하게 한다.

## Ⅰ. 서 론

함정 전투체계는 전투체계, C4I (Command, Control, Communications, and Intelligence) 체계, 기관제어체계, 통신체계, 행정체계 다양한 체제들이 혼합되어 운용되고 있다. 이 중 전투체계는 표적 정보 및 센서 정보 등 많은 데이터가 실시간으로 처리된다[1]. 이러한 하부 체계들은 다수의 센서와 장비를 취급하고 있는데, 현재 함정전투체계는 전시 상황에서의 즉각적 대처를 위해 하부 체계구조 및 서브 시스템의 복잡성과 데이터양이 증가하고 있다. 이러한 복잡성, 다양성을 고려하여 데이터 실시간성을 만족하기 위한 엣지 컴퓨팅 기반 프레임워크를 제안하는 등 다양한 연구가 진행되고 있다[2]. 새롭게 제안된 네트워크 구조는 실제 함정에 적용하기 전, 목표 성능 달성 여부를 적합한 지표를 통해 평가해 볼 필요가 있다. 이를 위해 함정 전투체계 내에 어떠한 실시간 애플리케이션이 있는지 확인하였으며, 센서정보처리장치, 표적정보처리장치, 영상분배장치(RTVDU, Radar Tv Video Distributor Unit) 와 같은 영상 애플리케이션 등이 있음을 확인하였다[1,3]. 이러한 영상 관련 애플리케이션의 경우 원본 데이터의 재현도가 서비스 품질(QoS, Quality of Service)에 영향을 미치기 때문에, 직접적인 Latency(지연시간)뿐 아니라 Jitter(지연변이)도 평가할 필요가 있다[4]. 즉, 영상 서비스는 규칙적인 패킷 스트림을 제공해야 하는 실시간 서비스이므로, 패킷 손실을 유발하는 버퍼 오버플로나 애플리케이션이 일정 시간 동안 패킷을 수신하지 않을 때 발생하는 버퍼 언더플로를 방지하기 위해 Jitter를 예측하는 것이 중요하다[5]. 따라서 본 논문에서는 직접적인 실시간성 평가를 위한 지표인 Latency와 실시간 네트워크의 패킷 도착시간 일관성 등을 평가하기 위한 지표인 Jitter를 분석하는 도구를 제안한다. 또한 이에 대하여 패킷 레벨 시뮬레이션이 가능한 네트워크 시뮬레이터인 NS3를 사용하여 패킷 크기를 고정한 시뮬레이션의 Jitter 분석을 통해 패킷의 크기와 관련 없는 네트워크 자체의 성능을 함께 분석하고자 한다.

## Ⅱ. 함정 전투체계 네트워크의 실시간성 검증툴 설계

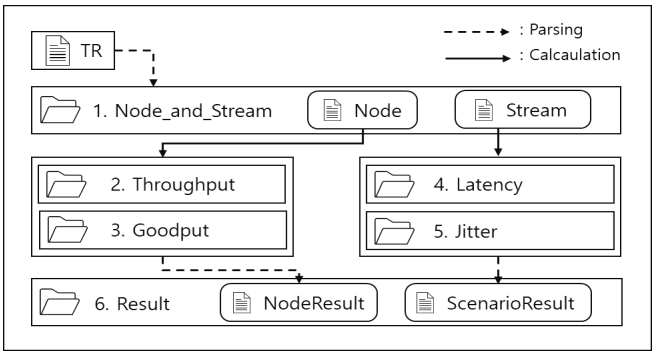


그림 1 프로그램 동작 흐름도

본 논문에서는 NS3에 의해 출력된 TR(Trace) 파일에서 데이터를 추출하여, 노드와 시나리오를 기준으로 정리된 2개의 파일을 최종적으로 출력한다. 노드의 송수신 Throughput과 Goodput 지표는 함정 네트워크 평가 지표로써 제시된다[6]. 따라서 시나리오에 대한 Latency와 Jitter를 추가로 출력하도록 설계하였다. 그림 1은 TR 파일에서 필요한 정보를 추출한 후 평가 지표들을 계산하는 흐름을 나타낸다. Throughput과 Goodput은 노드를 기준으로 추출되는 항목이며, TR 파일에서 각각 Length와 Payload 값을 가져와 처리한다. 또한 TR 파일에서는 기본적으로 노드의 Queue에 들어가는 동작, Queue에서 나오는 동작, 노드에 도착하는 동작을 각각 기호 +,-,r로 구별하여, 노드의 Throughput과 Goodput에 송수신 항목을 쉽게 분리하여 정리할 수 있다. Latency와 Jitter는 시나리오를 기준으로 추출되는 항목이며, TR 파일의 Source IP와 Destination IP가 모두 동일한 출력들에 대하여 같은 시나리오에 속하는 것으로 판단하게 된다. 그 후, 해당 출력 결과에 같이 출력된 줄의 시간 값을 이용하여 Latency와 Jitter를 구한다. 위의 총 4개의 세부 정보는 단위시간 별로 값이 구해져 있고, 최종 파일로 정리될 때 필요에 따라 추가적인 입력과 출력도 추가할 수 있다.

## Ⅲ. 함정 전투체계 네트워크의 실시간성 검증툴 구현

Latency와 Jitter를 구하기 위해서는 Stream을 불러와야 한다. 이때 파일을 쉽게 구별하고 호출하기 위해 파일명에 전처리가 이루어진다. Stream은 Source IP와 Destination IP를 통해 구별되지만, TR 파일 내에서도 번호가 따로 부여되어 있지 않기 때문에 IP 순서에 따라 0부터 차례대로 'Stream_숫자' 형식으로 임의로 이름을 붙였다. 따라서 Latency와 Jitter를 구할 때, 0번 시나리오부터 호출하여 구할 수 있다. 각 번호의 Stream 폴더 내부의 세부 항목인 송신 항목과 수신 항목을 호출하고 시간 부분을 추출 및 처리하여 Latency와 Jitter를 구하게 된다. Latency는 간단히 시간 차이를 단위 시간 행렬에 더해가며 구하였고, Jitter는 Algorithm 1과 같이 Latency의 변화량을 이용하여 구하였다.

**Algorithm1 :** Calculate Jitter

```
Input : StremList, Lots of SendFile, Lots of ReceiveFile
Output : Lots of JitterListFile, AllJitterListFile
    procedure
        Create & Reset Jitter Folder
        howmuch ← CountLines(StreamList 'Stream' Directory)
        for i ← 0 to howmuch − 1 do
            If not exists i-th SendFiles then continue end If
            If not exists i-th ReceiveFiles then continue end If
            count_s ← CountLines(i-th SendFile txt file in 'Stream/+' Directory)
            count_r ← CountLines(i-th ReceiveFile txt file in 'Stream/r' Directory)
            N ← Round(count_r/count_s)
            Initialize PrevLatency ← None
            Initialize JitterList ← 0-Array
            for each line in SendFile do
                SendTime = Time element of SendFile
                ReceiveTime = Time element of ReceiveFile
                if N>1 then ReceiveTime ← Average of N ReceiveTime
                Latency ← (ReceiveTime- SendTime)
                if PrevLatency is not None then
                    index ← floor(Time element of SendFile)
                    JitterList[index] ← (|Latency − PrevLatency| − JitterList[index])/16.0
                end if
                Update PrevLatency ← Latency
            end for
            WriteEachTxtFile(JitterList, i, JitterListFile)
        end for
        WriteEachFileToOne(JitterListFile, AllJitterListFile)
    end procedure
```

Jitter 계산 방식에는 여러 가지가 있다. 본 논문에서는 RTP(Real-time Transport Protocol) 표준인 RFC 3550을 따라 이득 매개변수 1/16을 사용하여 Jitter를 계산한다. 특정 시점의 Latency와 그 이전의 Latency 차로써의 기본 Jitter에, 그 절댓값과 그 이전 Jitter의 차를 구하고 1/16을 곱하며 값을 계속 더해간다. 이 과정을 단위 시간 별로 분리하여 배열로써 저장하여, 이를 통해 시나리오 결과에 단위 시간별 Jitter를 출력하였다. 추가로, N 변수는 Multicast에 의해 수신 결과가 송신 결과의 배수로 존재하는 경우를 처리하기 위해 코드에 포함되어 있다. N이 1보다 크면 수신 결과에서 N개의 시간을 추출하여 평균을 먼저 취한 후 Latency 계산에 활용한다.

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Type | Node name | Time | Send throughput | Send goodput | Received through | Received goodput |
| 2 | Switch | switch0_0 | 0 | 2373765 | 2230041 | 2373765 | 2230041 |
| 3 | Switch | switch0_0 | 1 | 2402645 | 2253769 | 2402645 | 2253769 |
| 4 | Switch | switch0_0 | 2 | 2384116 | 2241764 | 2384116 | 2241764 |
| 5 | Switch | switch0_0 | 3 | 2399104 | 2252384 | 2399104 | 2252384 |
| 6 | Switch | switch0_0 | 4 | 2411720 | 2265448 | 2411720 | 2265448 |
| 7 | Switch | switch0_0 | 5 | 2434333 | 2287697 | 2434333 | 2287697 |
| 8 | Switch | switch0_0 | 6 | 2425557 | 2279817 | 2425557 | 2279817 |

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Type | Scenario num | Source IP | Dest IP | Time | Latency | Jitter |
| 2 | Scenario | scenario0 | 10.4.31.2 | 10.4.56.2 | 0 | 0.00000359964787914 | 0.00000406034107748 |
| 3 | Scenario | scenario0 | 10.4.31.2 | 10.4.56.2 | 1 | 0.00000366298663568 | 0.00000396705538481 |
| 4 | Scenario | scenario0 | 10.4.31.2 | 10.4.56.2 | 2 | 0.00000364343786295 | 0.00000383089999473 |
| 5 | Scenario | scenario0 | 10.4.31.2 | 10.4.56.2 | 3 | 0.00000361650203370 | 0.00000381927908954 |
| 6 | Scenario | scenario0 | 10.4.31.2 | 10.4.56.2 | 4 | 0.00000361672473868 | 0.00000437704913571 |
| 7 | Scenario | scenario0 | 10.4.31.2 | 10.4.56.2 | 5 | 0.00000367112144102 | 0.00000448959078599 |
| 8 | Scenario | scenario0 | 10.4.31.2 | 10.4.56.2 | 6 | 0.00000361904761905 | 0.00000443646045912 |

그림 2 노드 및 시나리오에 대한 출력 결과

그림 2의 상단부는 노드에 대한 결과이며, 하단부는 시나리오에 대한 결과이다. 시뮬레이션 동작 시간은 10초로 노드 및 시나리오 결과 출력에 대해 공통으로 시간 항목이 추가되어 있다. 따라서 개별 노드 또는 시나리오 요소들은 각자에 대해 10개의 시간대별 결과를 묶음으로 출력하게 된다. 좌측의 Type 부분은 추후 출력된 데이터를 쉽게 정리하여 보기 위하여 노드와 시나리오에 대해 타입 명을 명시한 것이다. 시나리오 결과 파일에는 시나리오임을 간단히 표기하고, 노드 결과에 대해서는 노드 번호와 해당 노드에 대한 이름 데이터를 추가로 입력받아서 각 노드의 이름과 해당 노드의 스위치 여부를 구별하였다. 시나리오는 구별 기준인 Source IP와 Destination IP도 함께 출력한다.

## Ⅳ. 결론

본 논문은 함정 전투체계를 구성하는 노드, 트래픽, 네트워크에 대한 평가 지표로써 노드에 대한 송수신 Troughput 및 Goodput을 구하고 실시간성 평가 지표로써 네트워크 Latency와 Jitter를 구하였다. 시간대별로 분류하지 않은 단일 출력으로도 평가하기도 하나, 각 노드의 통신량과 각 시나리오의 네트워크 상태 변화를 단위 시간 별로 확인할 수 있게 구현하였다. 또한 시간대별 값에 대한 그래프를 출력하거나 이를 위한 파일 호출 기능을 설계할 때의 편리성을 위해 이름과 및 타입을 함께 출력하였다.

향후에는 특정 노드 간의 링크, 시나리오 내의 특정 경로 및 링크에 대한 네트워크 사용량 등 세부적인 부분에 대한 정보를 분석할 수 있는 기능을 추가할 계획이다. 이를 통해 여러 경로를 가지는 멀티캐스트 시나리오에 대해서도 구체적인 경로를 찾아 분석할 예정이다.


## ACKNOWLEDGMENT

본 논문은 2023년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업이며(2018R1A6A1A03024003, 50%), 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 지역지능화혁신인재양성사업임 (IITP-2024-2020-0-01612, 50%)


## 참 고 문 헌


[1] S. W. Oh, "An Integrated Architecture for Control and Monitoring Systems on Naval Surface Combatants", Journal of the KIMST, vol. 21, pp. 103-114, Jan. 2018

[2] G. S. Kim, J. M. Lee, D. S. Kim, and J. W. Kim, "Edge Computing Based Framework for Next Generation Naval Ship Combat System", Journal of the KICS, vol. 47, pp. 2078-2085, Dec. 2022

[3] S. M. Kown, and S. M. Jung, "Virtualization based high efficiency naval combat management system design and performance analysis", Journal of KSCI vol. 23, pp. 9-15, Nov. 2018

[4] L. Zhang, L. Zheng, and K. S. Ngee, "Effect of delay and delay jitter on voice/video over IP" Jounrnal of the Computer Communications, vol. 25, pp. 863-873, Jun. 2002

[5] H. Dahmouni, A. Girard, and B. Sansò "Analytical jitter model for IP network planning and design" 2009 First International Conference on Communications and Networking, Hammamet, Tunisia, pp. 1-7, Nov. 2009

[6] M. H. Jang, J. H. Lee, S. H. Lee, J. M. Lee, and D. S. Kim, "Design and Implementation of Simulator for Performance Evaluation of Naval Combat Network", KICS Summer Conference 2023, pp. 1272-1273, Jun. 2023