Zonal Architecture의 토폴로지에 따른 가용 컴퓨팅 시간 분석

배성원, 박한영, 장용재, 고강현, 최지웅*

대구경북과학기술원

swbae@dgist.ac.kr, prkhnyng@dgist.ac.kr, yj46.jang@dgist.ac.kr, ericko98@dgist.ac.kr, jwchoi@dgist.ac.kr*

Analysis of Available Computing Time According to Topology of Zonal Architecture

Sung won Bae, Hanyoung Park, Yongjae Jang, Kanghyun Ko and Ji-Woong Choi*

Daegu Gyeongbuk Institute of Science and Technology

요 약

본 논문은 Ethernet Backbone을 기반으로 토폴로지가 서로 다른 두 가지 차량용 전기전자 아키텍처를 OMNeT++로 구성해보고, 시뮬레이션을 통해 토폴로지 구성에 따라 LKAS를 수행하기 위한 센서(전방 카메라)와 액추에이터(MDPS) 사이의 End to End latency를 분석하고 데이터 처리를 위한 가용 컴퓨팅 시간을 비교해보았다.

Ⅰ. 서 론

자율 주행 기술이 발전함에 따라 카메라, 레이더, LiDAR 등 자율 주행에 필요한 여러 센서나 그와 관련된 ECU의 수가 늘어나고 있다[1]. 이에 따라 차량의 배선, 비용 및 복잡성 문제가 커지게 되었고, 이 문제를 해결하기 위해 여러 OEM과 TIER Supplier의 개발 방향은 Ethernet Backbone을 기반으로 하는 차량용 전기전자 아키텍처의 구성으로 변화하고 있다. 이렇게 Ethernet Backbone을 기반으로 하는 차량용 전기전자 아키텍처를 Zonal Architecture라고 하며, 차량의 데이터를 처리하는 고성능 프로세싱 유닛(이하 VC : Vehicle Computer)을 중심으로 지역별로 나눈 Switch와 고속의 Ethernet으로 연결되도록 하는 형태를 말한다.

이전에는 Zonal Architecture의 Ethernet Backbone을 Ring 토폴로지로 구성하여 LKAS와 AEB를 수행하는 Case에서 End to End latency를 시뮬레이션해보고, 어떤 어플리케이션이 더 많은 컴퓨팅 시간을 확보할 수 있는지 확인해보았다. 이전 연구가 두 가지 어플리케이션을 한가지 토폴로지에서 확보할 수 있는 컴퓨팅 시간을 확인해본 것이었다면, 본 논문의 연구는 같은 어플리케이션에서 토폴로지의 구성에 따라 가용한 컴퓨팅 시간이 어떻게 달라지는지를 확인하기 위해 진행되었다. 따라서 Ethernet Backbone을 2가지 토폴로지로 구성해보고 LKAS를 수행하는 측면에서의 지연시간을 분석하기 위해 전방 카메라와 MDPS 사이를 시뮬레이션하여 VC가 확보할 수 있는 프로세싱 여유 시간을 비교 분석해보았다.

Ⅱ. 본론

본 논문에서는 차량용 전기전자 아키텍처를 Ring 토폴로지와 Star 토폴로지로 구성해보았다. 구성한 두 가지 아키텍처의 공통사항은 다음과 같다. 차량 중앙부에 VC를 배치하고, 조수석 지역부터 Zone 1으로 시작하여 반시계 방향으로 Zone 2, 3, 4를 나누어 지역을 4개로 구분하였다. 각 Switch에는 자율주행 5단계를 가정하여 구성된 Camera, Lidar, Radar 등의 센서 및 액추에이터를 연결하였다[2]. VC와 Switch 사이의 Ethernet Backbone은 10Gbps로, Switch와 센서 사이의 인터페이스는 저속의 Ethernet과 CAN으로 구성하였다.

본 논문에서 살펴볼 Zonal Architecture의 가용한 컴퓨팅 시간은 다음과 같다. 센서부터 액추에이터까지 데이터가 생성되는 시점부터 처리되는 모든 과정까지의 시간을 End to End latency라고 하고, 이는 수식 1과 같이 IVN latency, Computing latency 그리고 Actuation latency의 합이 된다. 이 중에서 Computing latency가 본 논문에서 확인해볼 가용 컴퓨팅 시간이고, Actuation latency는 10ms[3]로 고정하였다.

$$L_{E2E} = L_{IVN} + L_{Computing} + L_{Actuation} \quad \cdots (1)$$

수식 1. End to End Latency의 구성식

$$L_{IVN} = L_{Tx} + L_{Queue} + L_{Sw} + L_{Gateway} \quad \cdots (2)$$

수식 2. IVN latency의 구성식

IVN latency는 다시 수식 2와 같이 Transmission latency와 Queuing latency 그리고 Switching latency와 Gateway latency의 합이 된다. 여기서 Switch를 통과하면서 발생하는 Switching latency는 7us[4]로, Gateway latency는 500us[5]로 발생한다고 가정했고, 1000Mbps Ethernet 카메라로 25Mbit의 데이터를 송신한다고 가정했을 때, 토폴로지 구성으로 인한 Transmission latency와 Queuing latency의 변화에 따른 가용 컴퓨팅 시간을 분석해보았다.

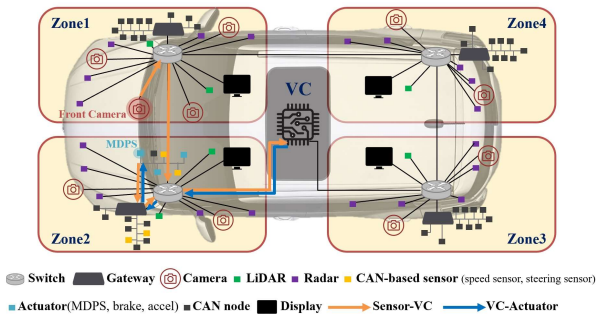

그림 1. Ring 토폴로지로 구성한 Zonal Architecture

그림 1은 Zonal Architecture를 Ring 토폴로지로 구성한 모식도이다. VC와 연결되는 Switch는 Zone 2와 Zone 3로 Zone 1과 Zone 4의 스위치는 VC와 데이터를 교환하기 위해 각각 Zone 2와 Zone 3의 Switch를 지나게 된다. 전방 카메라의 데이터는 차례대로 Zone 1의 Switch, Zone 2의 Switch, VC로 입력되고, 처리되어 다시 Zone 2의 Switch에서 gateway

되어 MDPS의 출력으로 이어진다.

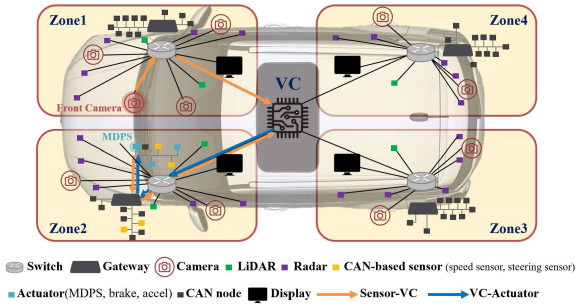

그림 2. Star 토폴로지로 구성한 Zonal Architecture

그림 2는 Zonal Architecture를 Star 토폴로지로 구성한 모식도이다. 전방 카메라의 데이터는 차례대로 Zone 1의 Switch, VC로 입력되고, 처리되어 Zone 2의 Switch에서 gateway 되어 MDPS의 출력으로 이어진다.

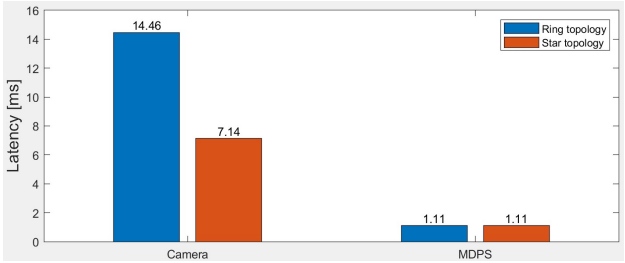

그림 3. 토폴로지에 따른 IVN latency 비교

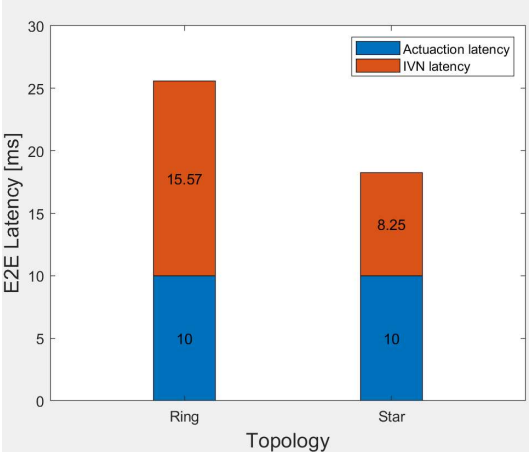

그림 4. 토폴로지별 E2E Latency 비교

그림 3과 같이 Camera 데이터의 IVN latency가 Ring 토폴로지에서 Star 토폴로지에 비해 큰 것을 확인할 수 있고, 이는 많은 센서와 연결된 2개의 switch를 지나면서 발생한 Transmission latency와 Queuing latency의 차이라고 할 수 있다. 동일한 관점에서 MDPS의 데이터는 토폴로지의 변경에도 같은 홉을 지나기 때문에 값이 바뀌지 않는 것을 확인할 수 있다.

자율주행 시스템은 100ms의 지연시간 내에 현재 교통 상황을 처리해야 한다는 논문을 참조[6]하면, 그림 4의 결과와 같이 Ring 토폴로지는 약 74.43ms, Star 토폴로지는 약 81.75ms가 Computing latency를 위한 여유 시간으로 확보되는 것을 확인하였다.

Ⅲ. 결론

본 논문에서는 OMNeT++ 시뮬레이션을 통해 차량의 전기전자 아키텍처에서 Transmission latency와 Queuing latency의 영향을 알아보았고, 이를 통해 Ring 토폴로지보다 Star 토폴로지에서 더 많은 컴퓨팅 시간을 확보할 수 있는 것을 확인해보았다. Ring 토폴로지가 아닌 Star 토폴로지로 구성하는 것은 switch의 고장에 따라 해당 Zone에 연결된 모든 센서 및 액추에이터를 사용할 수 없게 된다는 측면에서 risk가 크고, VC에 더 많은 인터페이스가 요구된다는 점에서 더 높은 cost가 예상되는 단점이 존재한다. 다만, 실험의 결과를 바탕으로 컴퓨팅 시간 측면에서는 Ring 토폴로지로 구성하는 것보다 Star 토폴로지로 구성하는 것이 유리하다는 것을 확인할 수 있었다. 본 논문에서는 가정한 값들이 많고 시뮬레이션을 통해 비교해보는 것에 그쳤으나, 실차의 환경에서는 다른 요인들이 영향이 존재할 것이므로 다음에는 이에 관해 연구할 예정이다.

ACKNOWLEDGMENT

이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임. (No. 2022-0-01053, 다중 통신기술 네트워크 로드밸런싱 기술개발)

참 고 문 헌

[1] Vector Inc., “Advances towards A Compact In-vehicle Ethernet-, Camera-, Radar- & LIDAR-measurement for High-bandwidth Driver Assistance Systems.” Vector India Conference, 2019. [Online]. Available:https://assets.vector.com/cms/content/events/2019/VH/VIC2019/5_Advances_towards_a_compact_invehicle_Ethernet_Camera_Radar_LIDAR.pdf.

[2] Cha, H. J., Jeong, W. H., and Kim, J. C. “Control-Scheduling Codesign Exploiting Trade-Off between Task Periods and Deadlines.” Mobile Information Systems, 2016, 2016, pp. 11.

[3] Maity, B., Yi, S., et al. “Chauffeur: Benchmark Suite for Design and End-to-End Analysis of Self-Driving Vehicles on Embedded Systems.”, ACM Transactions on Embedded Computing Systems (TECS), 20(5s), 2021, pp. 1-22.

[4] Santos, A. D., Soares, B., et al. “Characterization of Substation Process Bus Network Delays.” IEEE Transactions on Industrial Informatics, 14(5), 2018, pp. 2085-2094.

[5] Kern, A., Reinhard, D., et al. “Gateway Strategies for Embedding of Automotive Can-Frames into Ethernet-Packets and Vice Versa.” International Conference on Architecture of Computing Systems, 6566, 2011, pp. 259-270.

[6] Lin, S. C., Zhang, Y., et al. “The Architectural Implications of Autonomous Driving: Constraints and Acceleration.” Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, 2018, pp. 751-766.