IEEE 802.1CB 내결함성 전송을 위한 K-Max-Disjoint Path 알고리즘

민준홍[학생]
중앙대학교 소프트웨어대학 컴퓨터공학부
Email: dmc93@cau.ac.kr

백정엽[지도교수]
중앙대학교 소프트웨어대학 소프트웨어학부
Email: jpaek@cau.ac.kr

***Abstract*—IEEE 802.1 Time-Sensitive Networking (TSN) Task Group은 기존의 이더넷을 시민감적 데이터를 다룰 수 있는 시간 결정적인 네트워크로 이용할 수 있는 표준을 제정하기 시작하였다. TSN 네트워크의 핵심인 도착의 보장을 위해 TSN Task Group은 802.1CB에서 redundancy를 통한 fault-tolerance의 달성 위하여 Frame Replication and Elimination를 제안하였으나 정작 복제한 프레임들을 보낼 경로를 어떻게 찾고 구성할 것인지에 대해서는 구체적인 구상이 되어있지 않다. 따라서 본 논문에서는 전송 신뢰성을 위한 다양한 경로를 찾기 위하여, 트래픽이 요구하는 중복경로 k개를 만족하면서 최대한 disjoint 중복경로의 집합들을 만들고 전체 트래픽에 대한 load-balancing을 고려하는 Dijkstra에 기반한 휴리스틱 알고리즘을 설계 및 실험한다. 우리의 실험은 본 논문의 휴리스틱 알고리즘이 기존의 알고리즘들에 비해 보다 오래 걸리지만 트래픽들이 요구하는 중복경로들의 생성과 전체 네트워크의 load-balancing에서 많은 개선을 이뤄낸다는 것을 보여준다. 특히 본 논문의 알고리즘에만 전송거리와 같은 추가 제약이 있을 때에도 앞선 두가지의 성능이 더 좋은 것으로 나타났다.**

I. Introduction

오늘날 4차 산업혁명이 도래함에 따라, 수많은 기술들이 발전 및 개발되고 있다. 이러한 흐름은 데이터의 생성 및 이동을 증가시켰고 이는 기존의 통신 네트워크에 새로운 도전이 되고 있다. 또한 현재 개발되거나 시행되고 있는 많은 4차 산업기술들은 상당한 요구조건과 제약사항들을 가지고 있는 경우가 많으며 이를 만족시키기 위해서는 현재 보편화된 이더넷 구조가 아닌 요구조건에 맞게 커스터마이징 된 네트워크 구조가 필요하였고 이는 추가적인 비용들을 야기하였다.

IEEE 802.1 Time-Sensitive Networking (TSN) Task Group [1]은 Std 802.1Q를 통하여 현재 보편화 되고 비용이 저렴한 이더넷 기반 기술을 이용한 해결책을 제안하였다. TSN은 높은 요구조건을 가져 커스터마이징 하여 제공되는 네트워크를 대체하고, 나아가 현재 활발히 개발되고 있는 자율주행차, 공장자동화 등의 기반 네트워크가 되고자 초저지연, 낮은 지터 그리고 무손실 전송 등을 목표로 두고 있다. 이와 같은 결정론적 QoS 네트워크를 이더넷 기반으로 개발하고자 TSN은 크게 시간동기화, 스케줄링과 트래픽 그리고 내결함성 전송을 위한 경로예약으로 나뉘게 된다. TSN의 시간동기화는 IEEE 802.1AS [2]를 기반으로 하고 있으며, 스케줄링과 트래픽의 경우에는 IEEE 802.1Qcc [3], 802.1Qbv [4], 802.1Qbu [5] 등과 같은 표준에서 제안되고 있다. 마지막으로 본 논문에서 다루고자 하는 Fig. 1과 같은 내결함성 전송을 위한 경로예약은 IEEE 802.1CB [6]과 IEEE 802.1Qca [7]에서 다루고 있다.

IEEE 802.1CB은 신뢰성 있는 전송을 제공하는 내결함성 전송을 위해 Frame Replication and Elimination for Reliability 이라는 주제로 중복된 프레임을 복제하여 복수의 경로를 통해 전송하고, 수신 시 중복된 프레임을 제거하는 방법을 제안하고 있다. 그러나 802.1CB은 범위를 프레임 복제 및 제거에만 한정 지었으며 복수의 경로를 어떻게 구하고 정할 것인지에 대한 것은 해당 표준의 범위 밖이라고 정의하였다. 802.1CB에서는 경로의 문제는 802.1Qca에 관한 내용을 참고하라고 언급하고 있으나, 802.1Qca은 오직 2개의 경로를 만드는 알고리즘을 제안하고 있으며 그 이상은 유저가 직접 알고리즘을 구성하여야 한다고 언급하였다. TSN의 신뢰성 있는 전송을 위해서는 2개가 아닌 그 이상의 중복경로가 필수적이며, 네트워크의 구성 및 크기에 따라 중복경로 개수 k는 매우 커질 수도 있다. 따라서 본 논문에서는 중복경로를 만드는 문제를 일반화하여, k개의 중복경로를 만드는 알고리즘을 제안하고자 한다.

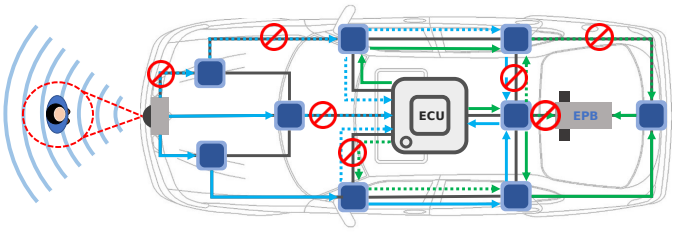


Fig. 1. 본 논문에서 제안한 알고리즘으로 구성한 내결함성 전송의 예

본 논문의 알고리즘에서 이루고자 하는 것은 1)k-max-disjoint paths 와 2)network load balancing이다. k-max-disjoint paths과 network load balancing 각각 모두 NP-Complete [8] 문제이기 때문에, 이를 위해 우리는 Dijkstra에 기반한 휴리스틱 알고리즘을 제안하며, 해당 알고리즘의 성능 비교를 위하여 기존에 중복경로에 관한 내용을 다루는 논문과 802.1Qca에서 언급한 알고리즘과의 비교를 진행하고, 종장에는 결과에 대한 분석 및 평가를 할 것이다.

II. Related work

중복경로 문제의 경우 TSN을 위해서 뿐만 아니라, 고전적인 그래프 문제로서 많은 논문에서 제안되었고 많은 알고리즘들이 있다. 그러나 이것들은 실제 TSN에서 사용하기엔 한계가 있다.

Loh et al.[9]는 최대 대역폭을 링크가 분리된 중복경로 세트를 만들기 위한 방법을 제안하고 기존의 다른 방법들과 비교를 하였다. 그러나 해당 논문은 단일 start-end에 대한 중복경로 생성을 제안하기 때문에 여러 start-end가 있는 실질

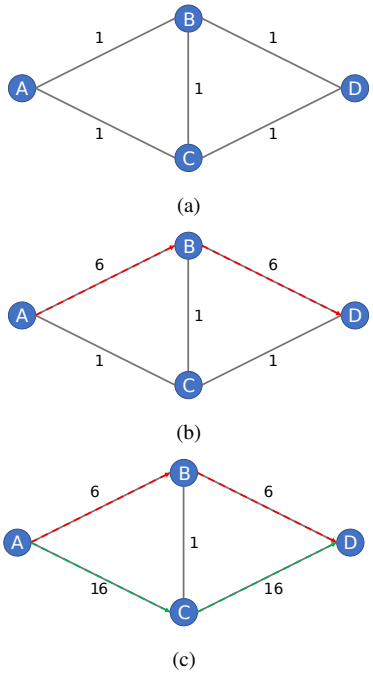


Fig. 2. (a) 네트워크 토폴로지 (b) 첫 번째 경로 (c) 두 번째 경로

```
Input:
TS: Traffic Set
G: Link Cost Graph
UL: Max Link Utilization (0 to 1)

def KMDA(TS, G, UL):
    for i in range(len(TS)):
        for j in range(TS.path_num):
            G_back_up = G
            for k in range(10):
                weight = sum_of_all_link_cost(G)
                route = Dijstra()
                update_link_utill()

                if not link_utill_exceed_limit:
                    if same_link_exist:
                        weight_random_link_in_path()
                    elif not satisfy requirement:
                        weight_random_link_in_path()
                    else:
                        add_route()
                        G = G_back_up
                        weight_all_link_in_path()
                else:
                    exceed_link_is_disable()
                    rollback_link_utill()

                if k == 14:
                    rollback_link_utill()
                    G = G_back_up
```

Listing 1. k-max-disjoint paths 알고리즘의 수도코드

적인 네트워크 환경에서는 load-balancing의 문제가 있으며, 완전히 분리된 경로만을 찾기 때문에 네트워크 topology의 구성에 따라 찾을 수 있는 중복경로의 수가 매우 제한된다.

Wang et al. [10]은 앞선 Loh et al.의 연구에서 load-balancing의 문제를 고려하여 좀더 현실적인 네트워크 경로 구성에 대한 방법을 제안하였으나, 이 역시 완전히 분리된 경로만을 찾는다. TSN의 경우 중복경로를 통한 신뢰성있는 전송이 최우선 목표임으로 완전히 분리된 경로가 아니어도 내결함성을 충족시켜줄 수 있다면 어느정도 링크를 공유하는 중복경로 세트를 제안할 필요가 있다.

Ojewale et al. [11]은 TSN을 위한 중복경로를 만들며, 경로 구성 시 load balancing 또한 고려하고 있다. 그러나 해당 논문의 알고리즘은 Equal link cost graph를 가정하고 있으며 이 또한 실제 네트워크 상황을 반영하기엔 한계가 있다.

III. Design

위 코드는 본 논문의 KMDA(k-max-disjoint path algorithm)의 로직이다. 기본적으로 경로는 Dijkstra 알고리즘을 이용해 찾으며[Code line 12] 그 다음 경로를 찾기 전에 링크들에 가중치를 주어 Graph의 cost를 변경시켜 준다. 가중치를 주는 방법은 아래 두 상황으로 나뉘게 된다.

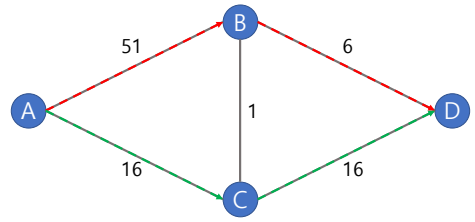
A. 합당한 경로가 생성되었을 경우

Fig. 2(a)와 같은 네트워크 토폴로지가 있을때 다익스트라 알고리즘을 이용하여 경로를 찾고, 해당 경로에 사용된 link의 전체 링크들의 코스트합[Code line 11]을 Fig. 2(b)와 같이 더해준다[Code line 23]. 이렇게 함으로서 이전 경로에 사용되었던 link의 코스트는 사용되지 않은 link들의 전체 합보다 크게 됨으로 다음 경로 선정에서 후순위로 밀리게 되며 Fig. 2(c)와 같은 최대한 disjoint한 경로를 찾게 된다.

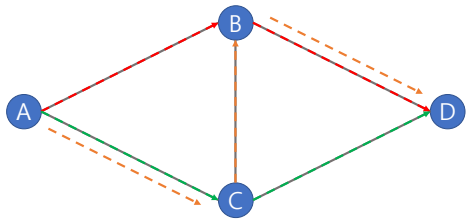
B. 합당한 경로가 생성되지 않았을 경우

Fig. 2(c)에서 다익스트라를 이용하여 다음 경로를 찾을 경우 Fig. 2(b)에서 이미 나온 경로가 생성이 되게 된다. 이는 합당하지 못한 경로가 생성 된 것이다. 이 경우 또한 찾은 경로에 사용된 link에다 전체 링크들의 코스트합을 더해준다. 그러나 다른 점은 사용된 각 링크에 가중치를 더해줄지 말지 링크마다 확률을 이용하여 결정한다. 이는 사용된 전체 링크에다 더해줄 경우 다음 경로선정이 특정한 경로 집합내에서만 번갈아 찾을 수 있기 때문이다. 예를 들면 Fig. 2(b)와 Fig. 2(c)같은 경로가 구성된 후, 다시 다익스트라를 이용하여 경로를 구성할 시 합당한 경로가 나오지 않더라도 해당 경로에 있는 모든 링크에 가중치를 더해주게 된다면 그 다음 경로 구성은 Fig. 2(b)가 나오며, 다시 경로 구성시엔 또 Fig. 2(c)가 나와서 결국 Fig. 3(b)의 경로 같은 것은 찾지 못하기 때문이다. 이를 위해 본 알고리즘은 확률을 이용하여 Fig. 3(a)과 같이 몇 링크에만 가중치를 주어 Fig. 3(b)같은 경로도 찾게 설계하였다.

이와 같이 우리는 경로 생성 성공과 실패의 case마다 가중치 정책을 다르게 책정하였으며, 또한 해당 가중치가 다음 경로를 찾을 때 적용되는지 여부 또한 다르게 구성하였다. 다음 경로를 찾을 때 적용되어야 할 가중치는 오직 성공적인 경로를 구성하는데 사용된 link로만 한정하며, 이를 위해 [Code line 22]에서와 같이 경로 구성 성공 시, 실패하면서 더했던 가중치를 모두 제거하게 된다. 이를 통해 실질적인 가중치는 실제로 사용된 링크에만 더해지게 되어 이는 전체 트래픽의 경로 구성 시에 load balancing이 고려되게 하여 링크 사용률이 고르게 분포하게 하여 허용된 링크 사용률이 넘어가는 것을 방지하고 링크 장애 발생시 여러 트래픽의 전송이 중단 될 수있는 critical link를 최소화 하고자 한다.



(a)



(b)

Fig. 3. (a) 실패시 확률을 통해 link A-B에만 weight 부과 (b) 세 번째 경로

```
def make_graph(n):
    G = np.zeros((n, n), dtype = float)
    F = float('inf')
    edge_num = 0
    list = [2500, 2500, 2500, 25000]
    for i in range(n+1):
    numlist.append(F)

    for i in range(n-1):
    while True:
        for j in range(i+1, n):
            G[i][j] = random.sample(list, 1)[0]
            if G[i][j] != F and G[i][j] != 0:
                edge_num += 1
        if (G[i][(G[i, :] != F)].sum()) > 0:
            break
```

Listing 2. Random topology를 만드는 코드

IV. Simulation and Evaluation

시뮬레이션을 수행한 네트워크 토폴로지는 위의 코드를 이용한 random topology를 이용하여 테스트하였으며 각 node의 개수 별로 100개의 random topology 대해 기존 알고리즘 Alg-1 [7], Alg-2 [9]와 함께 비교 시뮬레이션을 수행하였다.

네트워크의 링크의 종류는 1Gbps와 100Mbps인 두 가지의 종류가 있다고 가정 했으며 경로를 만드는 스케줄링 시간 단위는 250us로 가정하였다. 따라서 한 interval당 각 링크의 대역폭은 25000btye와 2500btye가 되게 된다. 각 링크의 코스트의 경우 interval당 대역폭의 값을 역으로 정렬하여 1Gbps의 코스트는 2500 100Mbps의 코스트는 25000이 되게 구성하였다. 또한 본 실험에서 이용하는 Input 트래픽을 제외한 기타 트래픽들의 전송 보장을 가정하기위해 링크의 사용률은 본래 대역폭의 최대 7할만 사용 가능하도록 제약하였다.

Table I 은 각 트래픽의 요구사항이다. 네트워크 토폴로지가 위의 코드에 따라 무작위로 구성됨으로 각 트래픽의 출발지와 도착지는 고정된 수로 가정하였으며 각 트래픽은 알고리즘을 동작하기전 Frame size순으로 정렬된다. 또한 Interval의 경우 계산의 편의성을 위하여 모두 동일하게 250us로 가정하였다. 만일 각 트래픽이 Interval이 모두 다르다면 스케줄링의 기간을 Input 트래픽 Interval의 최소공배수로 구성하면

TABLE I
Traffic Requirement

Traffic name	Frame size (byte)	Interval (us)	Max hop	Path num	Start	Destination
ST1	200	250	7	4	1	5
ST2	200	250	7	3	0	4
ST3	200	250	7	4	2	6
ST4	250	250	7	3	3	7
ST5	250	250	7	4	4	2
ST6	250	250	7	3	5	3

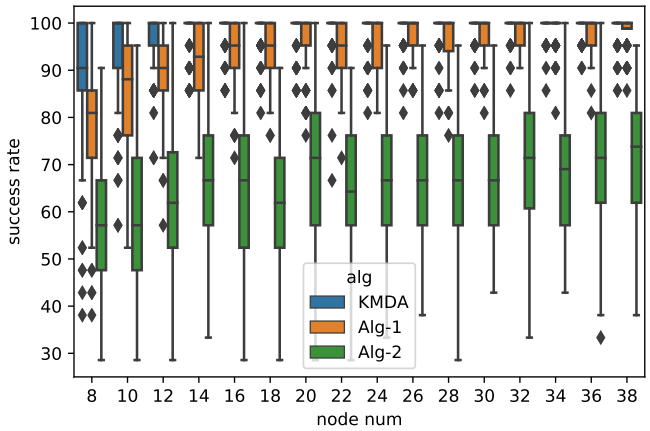


Fig. 4. 요구하는 경로의 개수에 대한 성공률

된다. 그리고 max hop을 통해 각 트래픽의 전송거리에 대한 제약사항을 두어 실제 TSN 네트워크에서 TSN 트래픽이 지정된 시간안에 전송되어야하는 것을 고려하였다.

Fig. 4는 각 노드의 개수별로 100개의 랜덤 네트워크 토폴로지를 구성하여 Input 트래픽들의 경로를 생성하는 것이다. KMDA는 본 논문의 알고리즘이며, Alg-1은 QCA [7]에서 언급되는 알고리즘 방법이며 Alg-2는 대역폭 최대화를 위한 disjoint한 중복경로를 만드는 논문 [9]의 알고리즘이다. Alg-1과 Alg-2모두 다익스트라 기반으로서, Alg-1은 경로 구성에 사용한 링크에 1000을 곱해주며, Alg-2는 한 트래픽의 경로 구성에 사용한 경로는 해당 트래픽의 중복경로를 만들 시 다시 못 사용 하게하는 알고리즘이다. Fig. 4의 결과를 보자면 KMDA는 Alg-1보다는 대략 5%에서 15%정도의 더 높은 경로 구성 성공률(생성경로의 개수 / 요구조건 경로의 개수)을 달성했으며 Alg-2 비교할 시 이는 40%로 올라간다. 여기서 주목해야할 점은 KMDA은 경로구성시 앞서 말한 max hop에 대한 제약 조건이 있지만 Alg-1과 Alg-2는 max hop에 대한 제약을 두지 않고 경로 생성을 하였는데도 불구하고 KMDA의 경로 생성 성공률이 훨씬 높다는 것이다. 또한 KMDA의 경로 생성의 실패이유는 우리가 몇몇 토폴로지에 대해 살펴본 결과 랜덤으로 구성한 토폴로지 구조에서 트래픽 요구조건에 맞는 경로가 실질적으로 없기 때문이었다.

Fig. 5는 각 시뮬레이션 경우마다 최대의 사용률을 가지는 링크의 사용률을 비교한 것이다. 먼저 Alg-2의 경우 최대 링크 사용률이 다른 두 알고리즘에 비해 낮지만, 이것은 Alg-2의 경로 생성 성공률이 매우 떨어지기 때문이다. KMDA와 Alg-1을 비교하자면 Fig. 4에서 보다시피 KMDA의 경로 생성 성공률이 더 높은데도 불구하고 KMDA의 최대 링크 사용률 링크의 값들이 대체로 더 낮다는 것을 볼 수 있으며,

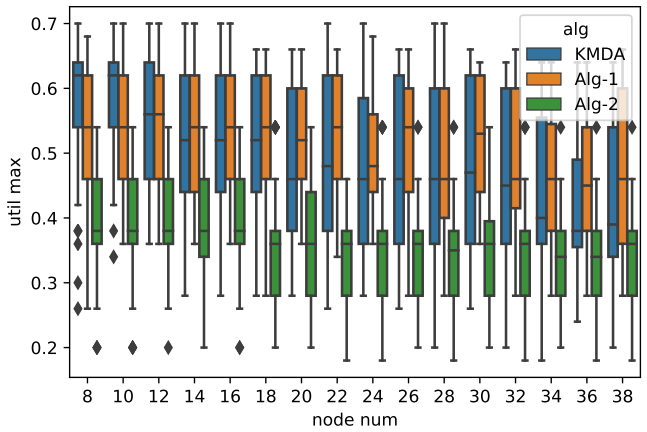


Fig. 5. 각 케이스별 최대 사용률 링크의 값

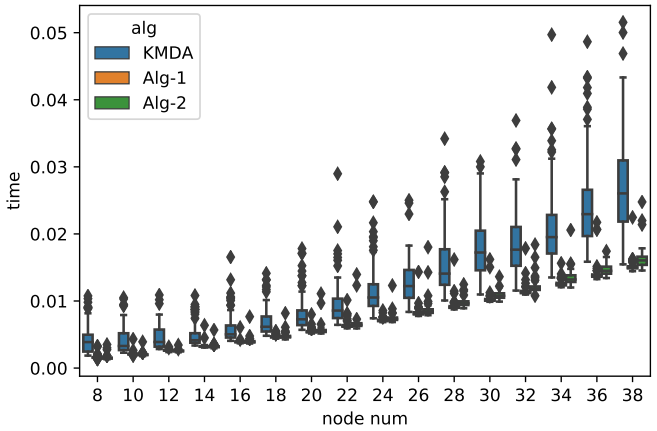


Fig. 6. 각 케이스별 경로 생성에 걸린 시간

이는 KMDA의 load balancing이 기존의 알고리즘들에 비해 잘되고 있다는 것을 보여준다. 다만 노드의 개수가 8과 10인 경우에는 KMDA의 값이 Alg-1에 비해 상당히 높은 것을 알 수 있는데, 이는 중복경로의 생성이 load balancing보다 우선시되기 때문에 노드의 개수가 8과 10같이 링크의 개수가 적어 경로 생성 성공률이 보다 떨어지는 경우에는 당연히 최대 링크 사용률이 높아질 수밖에 없다.

Fig. 6는 각 알고리즘이 경로 생성에 걸리는 시간을 비교한 것이다. KMDA는 나머지 두 알고리즘에 비해 보통 1.5배에서 3배정도의 시간이 걸리는 것을 확인할 수 있다. 그러나 소수의 경우들에서 KMDA의 작동시간이 상당히 커지는 것을 볼 수 있는데, 이는 해당 네트워크 토폴로지에 요구조건에 맞는 경로가 실제로 없기 때문에 KMDA가 계속 찾으려다 실패하기 때문이다. 이는 사용자가 경로 생성 실패 시 얼마나 재 시도를 할 것인지를 커스텀마이징을 할 수 있는데 본 실험에서는 Listing 1 코드의 line10과 같이 최대 10번의 재시도를 하게 구성하였다.

본 실험에서 KMDA 알고리즘은 트래픽 경로 생성에 대한 period 단위를 앞서 언급한바와 같이 Input 트래픽 Interval의 최소공배수로 설정하여 트래픽 경로를 구성한다. 이를 통해 한 period는 트래픽의 간섭에 대한 모든 경우의 수를 고려할 수 있으며 [12], 최악의 경우에도 해당 period 단위로는 트래픽 스케줄링이 가능하다. 따라서 본 KMDA 알고리즘이 생성하는 경로를 이용한 트래픽의 스케줄링은 언제나 가능하다.

V. Conclusion

본 논문에서 우리는 IEEE 802.1CB를 비롯한 IEEE Time Sensitive Networking 표준에서 제안하지 못한 신뢰성 있는 전송을 위한 효과적인 다중경로 구성 알고리즘을 위하여 Dijkstra에 기반한 휴리스틱 알고리즘을 제안하였다 또한 우리가 제안한 알고리즘이 얼마나 경로를 잘 생성하는지 보여주고 기존의 알고리즘과 성능비교를 하였으며, 전체 트래픽 경로 구성이 얼마나 load balancing 잘 고려되었는지도 보여주었다. 경로 구성 성공률과 load balancing을 위하여 기존 알고리즘에 비해 추가로 소모되는 시간과 그에 대한 효율성도 보여줬다. 또한 이렇게 생성한 경로를 이용한 트래픽 스케줄링이 언제나 가능하다는 것을 보여줬다.

우리는 앞으로 본 논문의 알고리즘을 통해 생성한 경로들을 이용하여, 실제 TSN의 작동을 위한 트래픽 전송 타이밍 스케줄링 구성을 어떻게 보다 효과적으로 할 것 인지에 대한 연구를 진행할 예정이다.

References

[1] “IEEE Time-Sensitive Networking Task Group,” [Online] http://www.ieee802.org/1/pages/tsn.html (last accessed Aug. 2020).

[2] “IEEE Std AS - Timing and Synchronization,” [Online] https://www.ieee802.org/1/pages/as.html(last accessed Nov. 2020), Mar. 2011.

[3] “IEEE Std Qcc- Stream Reservation Protocol (SRP) Enhancements and Performance Improvements,” [Online] https://ieeexplore.ieee.org/document/8514112(last accessed Nov. 2020), Jun. 2018.

[4] “IEEE Std Qbv - Enhancements for Scheduled Traffic,” [Online] https://ieee802.org/1/pages/bv.html(last accessed Nov. 2020), Mar 2016.

[5] “IEEE Std Qbu - Frame Preemption,” [Online] https://ieeexplore.ieee.org/document/7553415(last accessed Nov. 2020), Aug. 2016.

[6] “IEEE Std CB - Frame Replication and Elimination for Reliability,” [Online] https://ieeexplore.ieee.org/document/8091139(last accessed Nov. 2020), Sep. 2017.

[7] “IEEE Std Qca - Path Control and Reservation,” [Online] https://ieeexplore.ieee.org/document/7434544(last accessed Nov. 2020), Sep. 2015.

[8] A. Sen, Bin Hao, Bao Hong Shen, Ling Zhou, and S. Ganguly, “On maximum available bandwidth through disjoint paths,” in *HPSR. 2005 Workshop on High Performance Switching and Routing, 2005.*, 2005, pp. 34–38.

[9] R. C. Loh, S. Soh, and M. Lazarescu, “Maximizing bandwidth using disjoint paths,” in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 304–311.

[10] T. Wang, C. Q. Wu, Y. Wang, A. Hou, and H. Cao, “Multi-path routing for maximum bandwidth with k edge-disjoint paths,” in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2018, pp. 1178–1183.

[11] M. A. Ojewale and P. M. Yomsi, “Routing heuristics for load-balanced transmission in tsn-based networks,” *SIGBED Rev.*, vol. 16, no. 4, p. 20–25, Jan. 2020. [Online]. Available: https://doi.org/10.1145/3378408.3378411

[12] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, “Enabling fog computing for industrial automation through time-sensitive networking (tsn),” *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.