Packet Scheduling in Multipath TCP:
Fundamentals, Lessons, and Opportunities

0. Abstract

- MPTCP는 Peer간의 모든 IP Address에 대한 Multiple Transmission Control을 제공한다. MPTCP의 Subflow들의 Path는 Bottleneck link condition을 거치기 때문에, 송신 패킷을 예약하기 위한 최적의 Subflow를 선택하는 것이 Multipath의 성능에 중요한 역할을 한다. 좋은 스케줄링 결정은 Throughput을 크게 향상시킬 수 있지만, 잘못된 의사 결정은 사용자가 사용 가능한 Subflow의 Aggregating Capacity를 이용하는 것을 방해한다. 이러한 우려를 해결하기 위해, 지난 몇 년 동안 다양한 목표와 응용 프로그램을 달성하기 위한 몇 가지 스케줄링 솔루션이 제안되었다. 본 문서에서는 일관된 시험 방법론에서 기본사항 뿐만 아니라 Multipath 성능에 대한 상세한 분석을 제공하는 MPTCP 패킷 스케줄링 튜토리얼을 제공한다. Multipath 네트워크 설정에서는 서로 다른 Bottleneck Condition 및 Multipath Congestion Control에서의 단일기준과 다중기준 스케줄러를 비교한다. 몇 가지 성능 문제를 밝히는 실험 분석에서, 우리는 학습한 교훈과 Multipath Throughput 개선에 관한 연구 기회를 중심으로 논의한다. 이러한 방식으로 이 논문은 MPTCP에서 패킷 스케줄링을 이론만 아니라 실제적 포괄적으로 분석한다.

1. Introduction

- MPTCP Connection의 Subflow는 경로의 조건에 따라 성능이 달라진다. 경로 조건(Path Condition)은 Packet Loss Rate, Queue Delay, Throughput Capacity 등과 같은 Bottleneck Link 상태에 따라 결정된다.

- MPTCP의 Packet Scheduler는 각 패킷을 보낼 가장 적합한 Subflow를 선택해야 한다. "Best Subflow" 개념은 Packet Scheduler의 목적에 따라 달라지며, 따라서 Scheduling 결정은 Multipath 성능 향상에 중요한 역할을 한다.

- MPTCP 리눅스 커널 구현(www.multipath-tcp.org) 릴리스 이후 Multipath Packet Scheduling에 대한 연구가 수행되어 많은 스케줄링 솔루션이 서로 다른 목표와 응용 프로그램을 달성하기 위한 서로 다른 접근 방식으로 제안되었다. MPTCP 구현과 Congestion Control에 대한 Tutorial이 존재한다. 그러나 새로운 스케줄러를 제안하기 위한 연구 노력에도 불구하고 Review Studies(Article)에서 패킷 스케줄링은 생략되어 있다. 따라서 이 글에서는 Throughput향상의 관점에서 이 주제를 활용하고 기초(Fundamentals), 교훈(Lessons Learned), 기회(Opportunities) 이 세 가지 주요 측면에 기여한다.

- 처음에는 MPTCP 패킷 스케줄링 대한 Review를 제공한다. 리눅스 커널에서의 MPTCP 구현, 패킷 스케줄링 방법, 주요 스케줄링 문제가 무엇이며, 주요 기존 스케줄러는 두 개의 일반 범주(단일 및 다중 조건 스케줄러)로 그룹화 된다.

- 이 후 MPTCP의 여러 스케줄러의 성능을 분석하기 위해 최대 5개의 분리된 Bottleneck Link가 있는 MPTCP 리눅스 커널 구현으로 Multipath 네트워크를 설정하였으며, 여기에 Multipath 성능에 가장 큰 영향을 미치는 정량적(수치조정) 및 정성적(상황조정) 요소를 고려한 여러 실험을 수행하였다. 여덟 가지 Bottleneck Condition을 설계하고 네 가지 Congestion Control, 다섯 가지의 Packet Scheduler를 통해 총 160개의 서로 다른 실험을 수행하였다.

- 서로 다른 Bottleneck Condition, Congestion Control과 함께 Packet Scheduler에서 Subflow의 네트워크 상태 변수를 고려할 때, 다중기준 스케줄러는 단일기준의 기본 MPTCP 스케줄러(Lowest Latency : LL)보다 처리량을 최대 130% 더 향상 시켰다.

2. Fundamentals on MPTCP Packet Scheduling

- 이 섹션은 기존 MPTCP 리눅스 커널 구현에 대한 패킷 스케줄링과 주요 스케줄링 문제에 초점을 둔 MPTCP의 배경을 제공한다.
  해당 섹션에서의 MPTCP라는 단어는 MPTCP Linux Kernel Implementation로 대체된다.

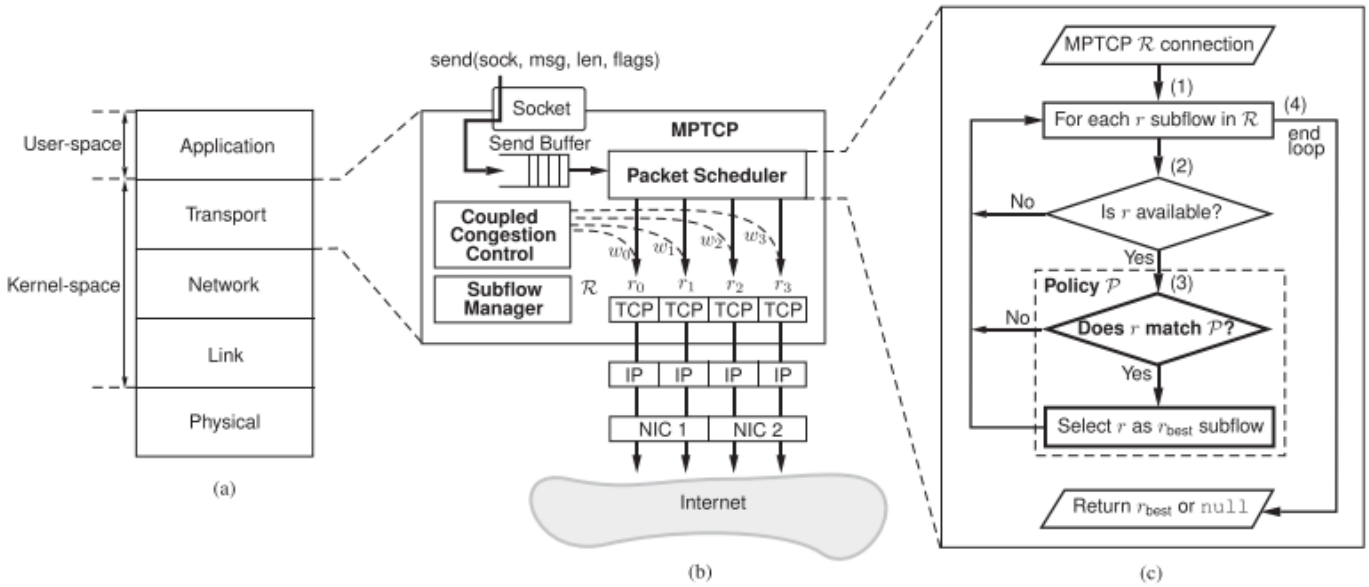

그림 1. 두 개의 NIC가 장착된 Multihomed End-System에 관한 MPTCP 리눅스의 일반적인 견해
(a) TCP/IP 스택, (b) MPTCP 아키텍처, (c) 모듈 식 패킷 스케줄링 프레임워크

- 그림 1(a)
  - MPTCP는 리눅스 운영 체제의 커널 공간에 있는 TCP/IP 프로토콜 스택의 Transport Layer에 위치한다.

- 그림 1(b)
  - MPTCP 내부에는 Connection R이 Application Layer에서 볼 수 없는 투명한 Subflow를 가진다.
    Socket은 App으로부터 MPTCP 서비스를 숨기는 고유한 인터페이스이므로 App Program을 변경하지 않고 유지한다.
  - Subflow Manager : 두 MPTCP End-System 사이에서 MPTCP Signalling(MP_CAPABLE, MP_JOIN, DSS, ADD_ADDR 등)을 통해 MPTCP Connection R의 하위 Subflow r_i를 제어한다.
  - Packet Scheduler : 송신 버퍼의 사용자 데이터를 패킷으로 나누고 각 패킷을 전송하기 위한 최상의 상태인 Subflow를 선택한다.
  - Coupled Congestion Control : 전체 연결과 여러 속성을 고려하여 Non-MPTCP Flow와의 Bottleneck을 공유함으로써 각 Subflow의 Congestion Window를 조정한다.

- 그림 1(c)
  - MPTCP는 모듈 식 패킷 스케줄링 프레임워크를 제공하며, 사용자가 다른 스케줄링 정책을 기반으로 QoS를 경험할 수 있도록한다.
  - (1) Socket 버퍼에서 패킷을 전송할 때 연결 R에 속하는 각 Subflow r이 Output 된다.
    (2) 첫 번째 검증은 Subflow 가용성으로, r_i의TCP 3-way Handshake가 완료되고 Congeistion 공간에 송신 패킷을 수용할 수 있는 공간이 있는지 검증한다.
    (3),(4) 검증된 r인 경우 스케줄링 정책 P에 따라 r은 송신 패킷을 전송하기에 가장 적합한 Subflow(r_{best})로 선택된다.

- MPTCP의 디폴트 스케줄링 정책은 Lowest Latency(LL)이다.
  - 이 정책은 lowest smoothed round-trip time(sRTT)로 최상의 Subflow를 선택한다.
  - RTT란 특정 시퀀스 번호로 패킷을 보낸 뒤 해당 시퀀스 번호를 포함하는 ACK 패킷을 수신하는 사이에 송신자가 측정한 경과 시간이다.

- LL 기반 정책은 head-of-line(HOL) blocking과 Bufferbloat를 완화하는데 도움이 된다.
  - HOL Blocking : 패킷 교환 네트워크에서 같은 큐에 있는 패킷이 첫번째의 패킷에 의해 지연될 때 발생하는 성능 저하 현상
  - Bufferbloat : 패킷의 과도한 버퍼링으로 인해 생기는 패킷 교환 네트워크의 높은 Latency의 원인

- 그러나 LL은 전송 속도, 패킷 손실과 같은 다른 경로 조건에 관계없이 최상의 Subflow를 결정하는 단일 기준으로 Path Delay(RTT)에만 의존한다.
  다음 그림 2를 통해 HOL과 Bufferbloat와 다른 Bottleneck 현상 두 가지 시나리오를 확인할 수 있다.

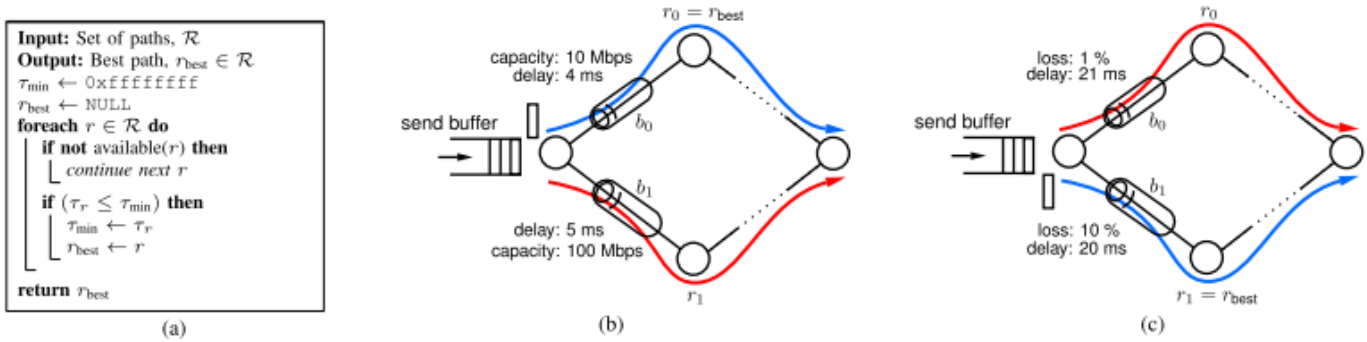

그림 2. 기본 MPTCP 스케줄러, 두 경로가 서로 다른 특징을 갖는 시나리오에서 스케줄링 결정의 예
(a) Lowest Latency. (b) rbest가 작은 delay, 작은 capacity 예. (c) rbest가 작은 delay, 높은 loss.

- 그림 2의 (b) 시나리오에서 두 path 모두 delay가 4ms와 5ms로 작지만, r0은 10Mbps의 capacity traffic이고 r1은 100Mbps의 traffic이다. 그러나, LL은 rbest를 RTT로만 결정함으로 r0를 통해서만 패킷을 전송한다.

- 그림 2의 (c) 시나리오에서 LL 정책으로 인해 두 path 중 delay가 1ms 작지만 loss가 높은 r1이 전송 경로로 쓰여진다. 해당 r1 경로를 통해 패킷을 전송을 유지한다면 결국에는 Congestion 및 Loss 상태를 유발하게 된다.

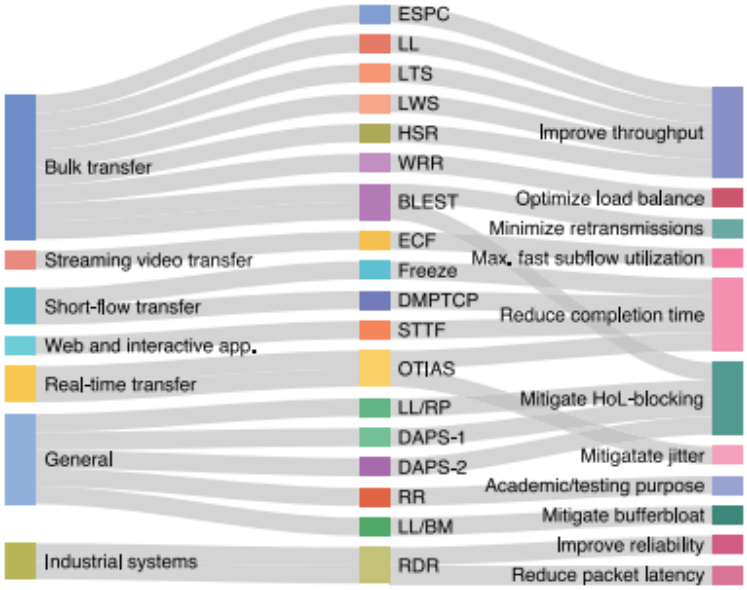

그림 3. Application과 Scheduler간의 전반적인 Goals
왼쪽 App, 중간 Scheduler, 오른쪽 Goal

- 그림 3은 MPTCP의 여러 스케줄러들이 어떠한 서비스의 목적으로 설계되었는지 나타낸 그림이다.
  - WRR : Optimize Load Balance 스케줄러
  - BLEST : HoL Blocking 완화 스케줄러
  - ECF : Streaming Application을 위해 가장 빠른 Subflow의 활용을 극대화한 스케줄러
  - LL/RP, DAPS-1, DAPS-2 : 특별한 Application을 위한 것이 아닌 HoL 차단만을 목표로 하는 스케줄러
  - LL/BM : Bufferbloat 문제를 완화시키기 위한 스케줄러
  - RR : 테스트 목적 스케줄러
  - OTIAS : jitter와 Transfer Latency를 줄이는 것을 목표로 하는 스케줄러
  - FPS, DMPCTP : 짧은 전송 시간(Ex. 마우스 움직임 등)이 필요한 Application을 위한 스케줄러
  - STTF : 웹 및 대화형 Application의 전송 시간을 줄이는 것을 목표로 하는 스케줄러
  - RDR : 제어 및 데이터 수집(SCADA)와 같은 산업 시스템에서 패킷 전송 지연 시간을 줄임으로써 신뢰성 향상이 목적인 스케줄러

Ref.	Scheduler	Algorithm	Approach
[4]	LL	Lowest-latency first.	Default MPTCP scheduler that chooses the available subflow of lowest sRTT.
[4]	RR	Round-robin.	Alternates the use of available subflows in round-robin fashion.
[5]	DAPS-1	Forward delay aware.	Schedules a chunk of packets based on their sequence numbers and the path delays for in-order receiving.
[6]	DAPS-2	Maximum receive buffer blocking time.	Same approach of DAPS-1.
[7]	ESPC	Estimated subflow path capacity.	Estimates subflow capacity and keeps it longer at the saturation point, while considering the path congestion.
[8]	LL/RP	LL with retransmission and penalization.	LL extension to retransmit HoL packets and penalize subflows that cause HoL-blocking.
[8]	LL/BM	LL with bufferbloat mitigation.	LL extension to identify subflows in bufferbloat condition bounding their congestion windows.
[9]	OTIAS	Out-of-order transmission for in-order arrival.	Estimates the packet arrival time over all available subflows, and chooses the ones of earliest time.
[10]	RDR	Redundant Data Reinjection.	LL extension to send the packet by replicating it throughout the other existing subflows.
[11]	FPS	Temporary freezing of slow subflows.	LL extension to freeze temporarily slower subflows and prioritize the fastest ones.
[12]	BLEST	Blocking estimation.	Estimates whether a subflow will cause HoL-blocking and dynamically adapts scheduling to prevent blocking.
[13]	ECF	Earliest completion first.	Identifies whether scheduling a slow subflow will cause faster subflows to become idle.
[14]	LTS	Lowest time/space first.	Chooses subflows of lowest ratio of latency and window space whenever possible, otherwise, reduces to LL.
[14]	LWS	Largest window space first.	Chooses subflows of largest congestion window space whenever possible, otherwise, reduces to LL.
[14]	HSR	Highest send rate first.	Chooses subflows of highest send rate.
[15]	WRR	Weighted round-robin.	RR extension with subflow weight to provide optimal load-balance.
[16]	DMPTCP	Application workload aware.	Finds the set of the subflows that can complete the applications transmission within RTT.
[17]	STTF	Shortest transmission time first.	Estimates transfer time of a data segment over the subflows and selects the one of the shortest time.

표 1. MPTCP 리눅스 커널에 제안된 다양한 패킷 스케줄링 솔루션 요약

- 표 1은 그림 3에서 설명한 스케줄러 들이 어떠한 정책을 바탕으로 패킷 스케줄링을 결정하는지 나타낸 표이다.

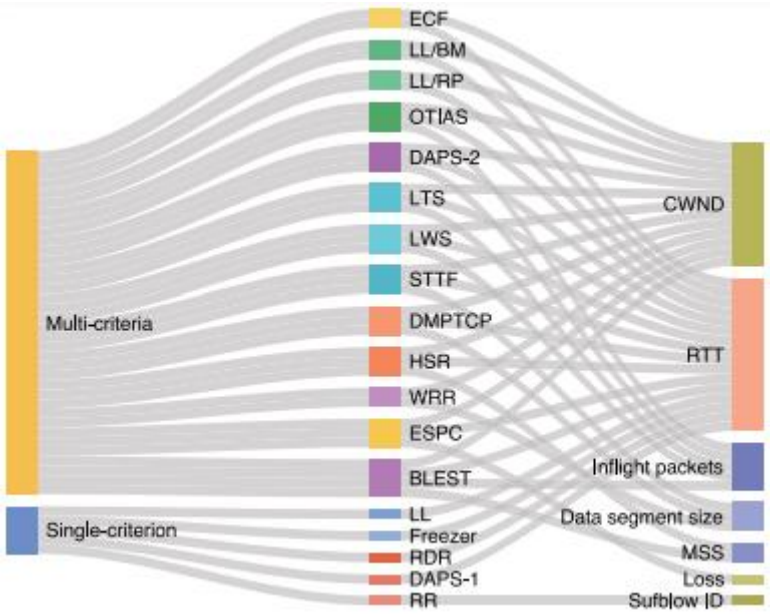

그림 4. Two general categories(왼쪽), Schedulers(중간), and theirs input metrics(오른쪽)

- 다음 그림 4는 단일기준 및 다중기준을 사용하는 MPTCP Packet Scheduler를 나타낸 그림이고, 해당 그림의 가장 오른쪽에는 스케줄러의 최상의 Subflow 결정에 사용되는 네트워크 정보 Input Metric이다.

3. Applying the Experimental Methodology to Evaluate MPTCP Packet Schedulers

- 이 섹션에서는 MPTCP 스케줄러 실험 환경에 대해서 설명한다.

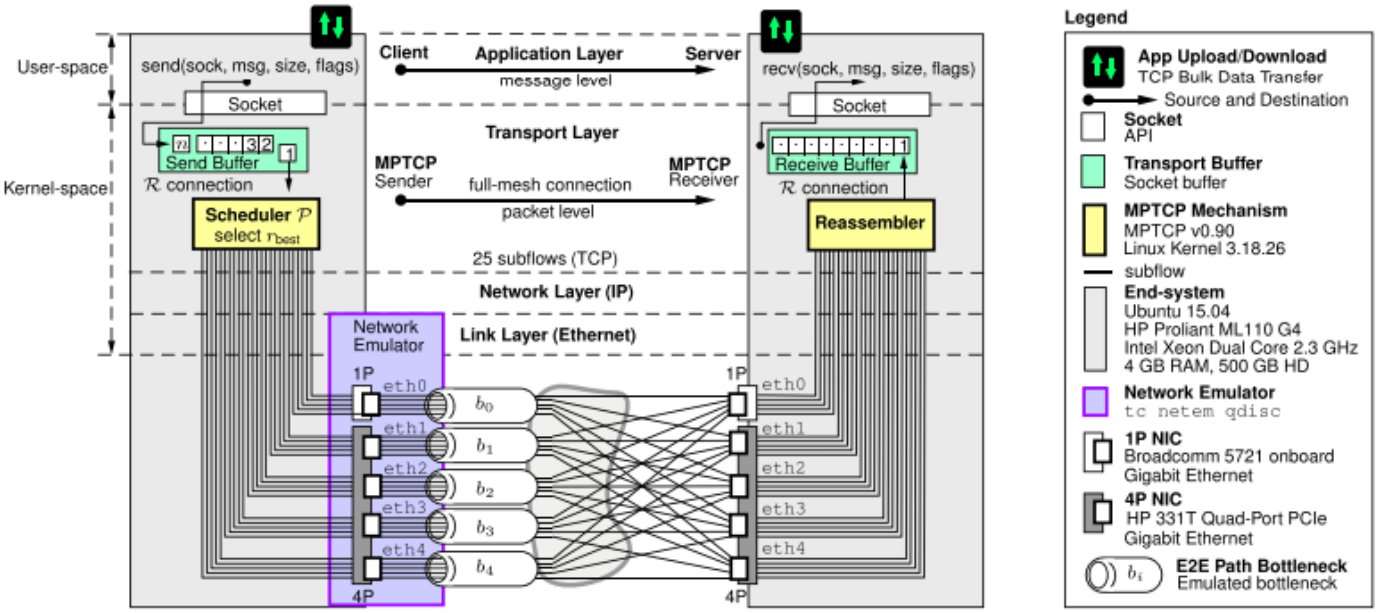

그림 5. Multipath Network Setup and the system architecture used in the experiments.

- 그림 5의 구성요소들에 대한 설명은 오른쪽 박스에 있다.

- End-System간의 eth 인터페이스 쌍은 서로 연결되고 고유한 IP 네트워크 주소가 지정된다.
  MPTCP Subflow Manger는 5개의 분리된 Bottleneck을 통해 Client-Server 노드 간 25개의 서로 다른 Subflow를 full-mesh로 설정한다.
  Bottleneck Emulation은 ***tc, qdisc, netem*** 리눅스 툴을 사용하여 Client에서 수행된다.

	Factors	Domain of Values	Description
Quantitative	Capacity (Mb/s)	10, 25, 50, 75, 100	Bottleneck capacities of broadband access in both wireless (WiFi and LTE) and wired (datacenters and FTTH) networks.
	Packet Loss (%)	0, 1, 2, 6, 12	While losses in wired Internet is very low ($\leq 0.1\%$) [47], higher rates ($\leq 10\%$) are experienced in wireless links [50]. We observed losses higher than 12% prevent MPTCP to distinguish subflows.
	Queue Delay (ms)	5, 10, 20, 40, 80	Bottleneck queue delays that allow RTTs to vary up to 160 ms, in accordance with MPTCP latency measured in both fixed [49] and mobile networks [48].
Qualitative	Subflow Management	Full-mesh (default)	The current MPTCP subflow manager creates a complete mesh of subflows from all the possible pairs of source and destination IP addresses between two MPTCP-enabled nodes.
	Congestion Control	LIA (default)	*Linked-Increases Algorithm* [52] is the default MPTCP congestion control. It dynamically adjusts the subflows' aggressiveness to improve throughput at least as a regular SP-TCP on the best path, to do no harm to non-MPTCP flows when sharing bottlenecks, and balance congestion among subflows.
		OLIA	*Opportunistic Linked-Increases Algorithm* [53] provides responsiveness to react to network changes, and an optimal congestion balance among subflows. To move traffic from fully utilized paths to the ones of free capacity, congestion windows increase faster for best subflows (with higher transmission rates) and slower for most used subflows (of larger congestion windows).
		BALIA	*Balanced Linked Adaptation Algorithm* [54] adjusts the subflows' congestion windows to reach oscillation levels of equilibrium between responsiveness and fairness. On the best available path or in single-path connections, the congestion control is reduced to the regular TCP.
		wVegas	*Weighted Vegas* [31] estimates queue delays of bottlenecks to infer congestion. It adjusts proactively subflows' congestion widows in two steps: increase/decrease based on the subflow goodput; upon variation on the estimated queue delays, apply a backoff factor to decrease.
	Scheduling Policy	RR, LL (default)	Baselines of single-criterion schedulers available across the versions of MPTCP. While RR is possibly the worst scheduler, LL is the best one to overcome.
		HSR, LWS, LTS	Baselines of multi-criteria schedulers we proposed in [14].

표 2. 실험에서 선택될 요인 및 영역

- 실험은 현재 MPTCP 버전에서 사용할 수 있는 공식적인 Subflow Manager, Congestion Control, Scheduling Policy를 사용한다.

- 실험 적용 요인
  - 양적 = 정량적 요인 (수치변화)
    : Capacity (Mb/s), Packet Loss (%), Queue Delay (ms)
  - 질적 = 정성적 요인 (상황변화)
    : Path Manager(full-mesh), Congestion Control(LIA, OLIA, BALIA, wVegas), Scheduling Policy(RR, LL, HSR, LWS, LTS)

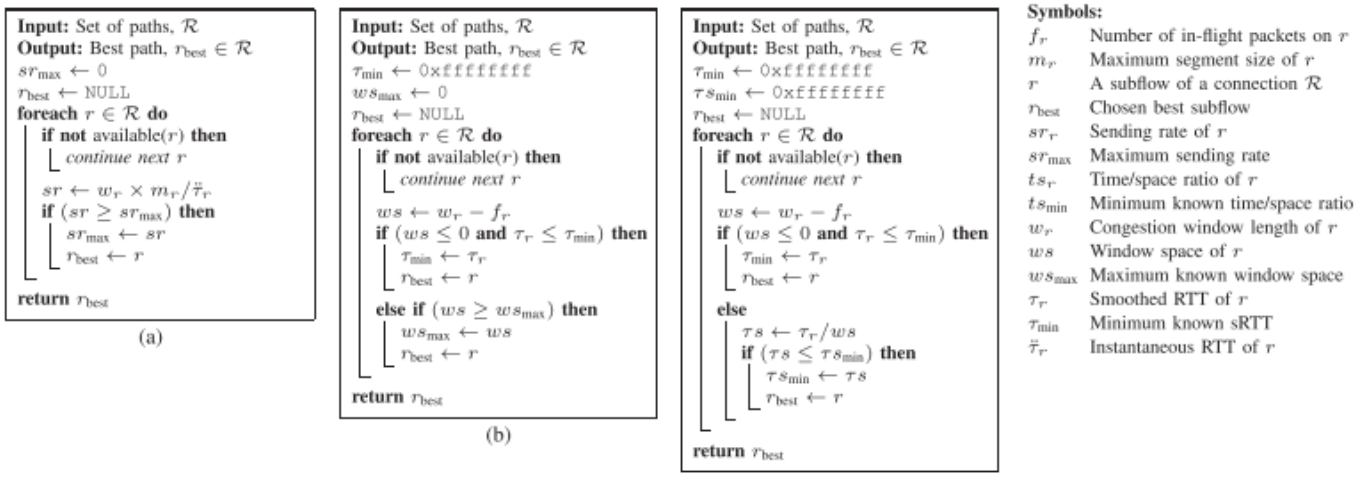

그림 6. 다중 기준의 MPTCP Packet Scheduling Policy Algorithm

(a) Highest Sending Rate - HSR, (b) Largest Window Size - LWS, (c) Lowest Time/Space – LTS

- (a) Highest Sending Rate - HSR
  - 더 큰 Congestion Window인 wr와 더 낮은 Instantaneous RTT(Tr, 수신된 패킷에 대한 즉각적인 RTT)는 충돌하지 않는 Bottleneck을 나타낼 수 있기 때문에, HSR은 더 높은 출력 속도의 Subflow가 패킷 전송에 적합할 수 있다는 것이다.

- (b) Largest Window Size - LWS
  - Congestion Window에 송신 패킷을 가장 많이 수용할 수 있는 Subflow 선택
  - Subflow Window인 ws는 Congestion Window의 wr와 전송 중인 패킷(in-flight packets) 수 fr의 차입니다.
  - LWS는 대용량 병목에서 실행되는 Subflow는 네트워크에서 패킷이 더 빨리 offload 되기 때문에 정체 창에 더 많은 공간을 갖는 경향이 있다고 판단한다. 따라서, Connection(R)의 subflow(r)는 wr 창에 보류 중인 승인 패킷을 더 적게 갖는 경향이 있다.

- (c) Lowest Time/Space – LTS
  - Subflow Window인 ws와 Delay Time인 Tr 이 가장 낮은 비율인 Subflow 선택
  - LTS는 LWS와 LL을 병합하여 더 높은 용량(capacity)의 Subflow은 Window가 더 크고 정체가 덜한 병목 지점에서 실행되는 Subflow는 지연 시간이 짧기 때문이라고 판단한다.

	Experiment e_0			Experiment e_1			Experiment e_2			Experiment e_3		
Bottle	⓪	⓪	⓪	⓪	⓪	❶	⓪	❶	⓪	⓪	❶	❶
-neck	Loss	Capacity	Delay	Loss	Capacity	Delay	Loss	Capacity	Delay	Loss	Capacity	Delay
b_0	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	5 ms
b_1	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	10 ms	0 %	25 Mb/s	5 ms	0 %	25 Mb/s	10 ms
b_2	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	20 ms	0 %	50 Mb/s	5 ms	0 %	50 Mb/s	20 ms
b_3	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	40 ms	0 %	75 Mb/s	5 ms	0 %	75 Mb/s	40 ms
b_4	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	80 ms	0 %	100 Mb/s	5 ms	0 %	100 Mb/s	80 ms

	Experiment e_4			Experiment e_5			Experiment e_6			Experiment e_7		
Bottle	❶	⓪	⓪	❶	⓪	❶	❶	❶	⓪	❶	❶	❶
-neck	Loss	Capacity	Delay	Loss	Capacity	Delay	Loss	Capacity	Delay	Loss	Capacity	Delay
b_0	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	5 ms	0 %	10 Mb/s	5 ms
b_1	1 %	10 Mb/s	5 ms	1 %	10 Mb/s	10 ms	1 %	25 Mb/s	5 ms	1 %	25 Mb/s	10 ms
b_2	2 %	10 Mb/s	5 ms	2 %	10 Mb/s	20 ms	2 %	50 Mb/s	5 ms	2 %	50 Mb/s	20 ms
b_3	6 %	10 Mb/s	5 ms	6 %	10 Mb/s	40 ms	6 %	75 Mb/s	5 ms	6 %	75 Mb/s	40 ms
b_4	12 %	10 Mb/s	5 ms	12 %	10 Mb/s	80 ms	12 %	100 Mb/s	5 ms	12 %	100 Mb/s	80 ms

표 3. 실험 설계

- 표 3은 실험 시 정량적 요인의 수치변화를 나타낸 표이다.
  즉, 8번의 실험을 다음 표 3과 같이 진행하며 표의 각 수치를 그림 5의 Bottleneck bi에 적용시킨다.

- 5개의 Packet Scheduler(RR, LL, HSR, LWS, LTS), 4개의 Congestion Control(LIA, OLIA, BALIA, wVagas), 1개의 Subflow Manager(full-mesh)로 구성되므로 20(5x4)개의 서로 다른 솔루션 세트를 얻었다. 또한, 각 솔루션에는 표 3과 같은 8번의 실험이 진행될 것으로 총 160개(20x8)의 다른 실험이 진행된다.

- 다른 프로토콜이 Traffic 상에 존재하지 않는다는 전제하에 실험을 진행한다.
  또한, 실험 시 Client는 3MB씩 500개의 메시지를 지속적으로 보냈으며, 패킷 분석시 ***tcpdump2***로 Traffic을 캡처하였습니다. 그리고 ***tshark3***로 Multipath Traffic을 분석하였다.

4. Lessons Learned from Performance Analysis

- 이 섹션에서는 실험 결과를 그림 7, 8, 9를 통해 나타낸다.
  또한, 그림 9는 다중기준 스케줄러(LTS, LWS, HSR)과 단일기준 스케줄러(LL) 사이의 상대적인 성능을 비교한다.

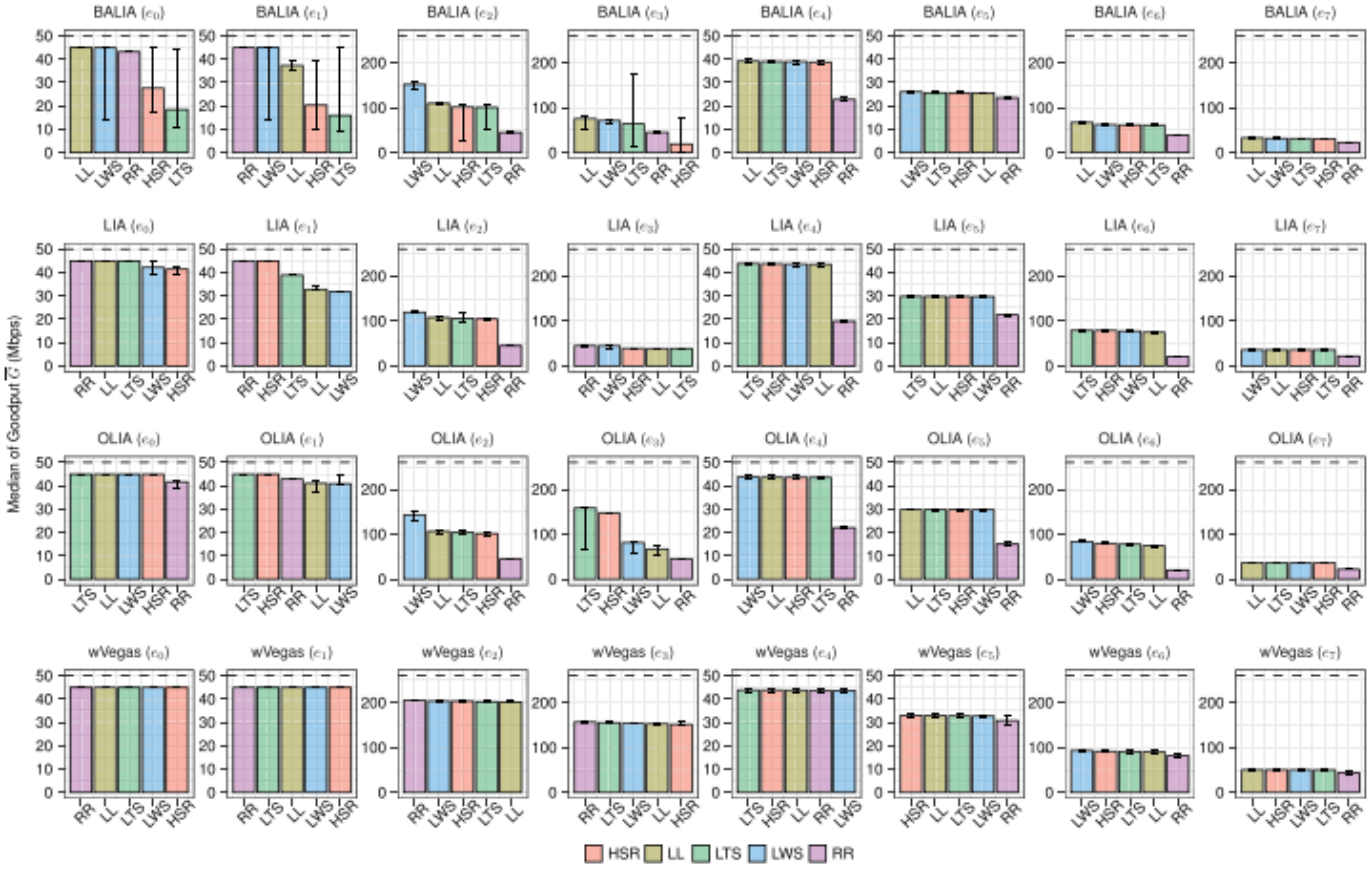

그림 7. 160번 실험의 goodput(단위 시간당 손실되지 않고 전달이 완료된 데이터 량) 결과

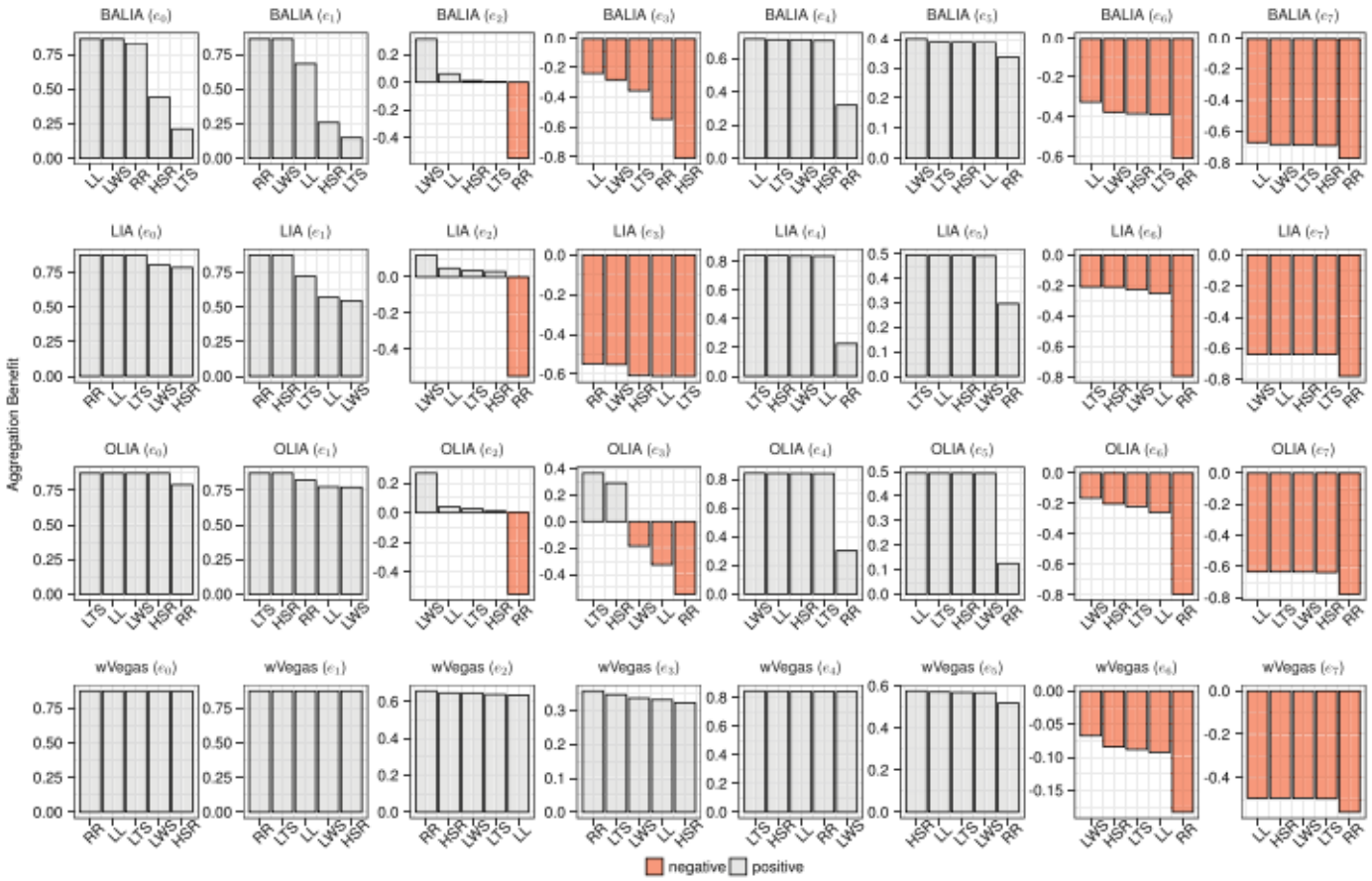

그림 8. 160번 실험의 Aggregation Benefit 결과

A. 스케줄링이 Loss 및 Delay 병목 현상에 영향을 미치지 않음
   - 그림 7의 실험 e0, e1, e2, e3에서 병목현상의 loss가 0일 때, 즉 Congestion 상태에서만 손실이 발생했을 경우 스케줄링 결정은 MPTCP Connection의 성능에 상당한 영향을 미친다.
   - 그림 7의 실험 e4, e5, e6, e7에서 병목현상의 loss가 늘어날 때에서는 RR을 제외하고는 스케줄러 간에 유의미한 차이가 없다.
   - 우리는 높은 Delay와 함께 높은 Loss Rate는 스케줄링 결정 및 혼잡 제어에 영향을 미치지 않는 것을 확인할 수 있다.

B. 때때로 동일한 Load Sharing이 가치가 있는 경우
   - RR showed that having no criterion to distinguish subflows leads to performance degradation in most of experiments, mainly when bottlenecks have heterogeneous capacity and/or loss rates (Experiments e2, e3, e4, e5, e6, and e7). When no subflow is prioritized, RR avoids biased feeding with load balance but induces head-of-line blocking in heterogeneous paths. However, in two situations, there are aggregation benefits higher than 0.75,regardless of the congestion control, as shown in Fig. 8. First, in bottlenecks completely homogeneous (e0), when distributing a packet workload equally among subflows of same characteristics. Second, in delay-divergent bottlenecks (e1), RR makes slower subflows to be utilized as much as faster ones. With RR, we observed that head-of-line blocking is much more harmful under bottlenecks of heterogeneous capacities and/or losses than heterogeneous delays.

C. Lowest-Latency Subflow는 여러 종류의 병목 현상에 적합하지 않음
   - 실험 e1에서 LL이 가장 최상의 성능을 얻을 것으로 예상하였다.
   - 하지만, RR을 포함한 다른 스케줄러가 BALIA, LIA, OLIA에서 더 나은 성능을 보였다.
   - LL의 편향된 특정 Subflow 공급은 느린 Subflow를 덜 활용하면서 빠른 Subflow의 정체 창을 가득 채우는 경향이 있게 한다. 따라서, Congestion에 반응할 때 느린 Bottleneck의 용량은 활용되지 않는 반면, 느린 Subflow보다 빠른 Subflow에 대해 Window 감소(Congestion 상태에서)가 훨씬 더 공격적이다.

D. Greater the Heterogeneity of Paths' Capacities, the Lower the Multipath Performance
   - In the name of friendliness with non-MPTCP flows sharing bottlenecks, the congestion control algorithms have rather conservative behavior, which leads to adverse performance results. Even when the schedulers prioritize the best subflows, the aggressiveness of the subflows is controlled to limit the congestion window more than the necessary. This can be verified in the results of Experiments

e2 and e3, where lossless bottlenecks have heterogeneous capacities, whose total multipath aggregate capacity is 200 Mb/s. In detriment to the performance, results show that various congestion controls boycott subflows in the name of a false friendliness that brings no benefit—note that only MPTCP subflows share bottleneck links, with no contention with non-MPTCP flows. In LIA, a unique aggressive factor limits the increase of all the subflows' congestion windows, preventing the windows from exceeding the one belonging to the subflow on the best path. This makes multipath performance never greater than a regular SP-TCP connection in the best available path. However, we observe similar behavior in OLIA and BALIA, with $A \approx 0$ in Experiment e2 in Fig. 8.

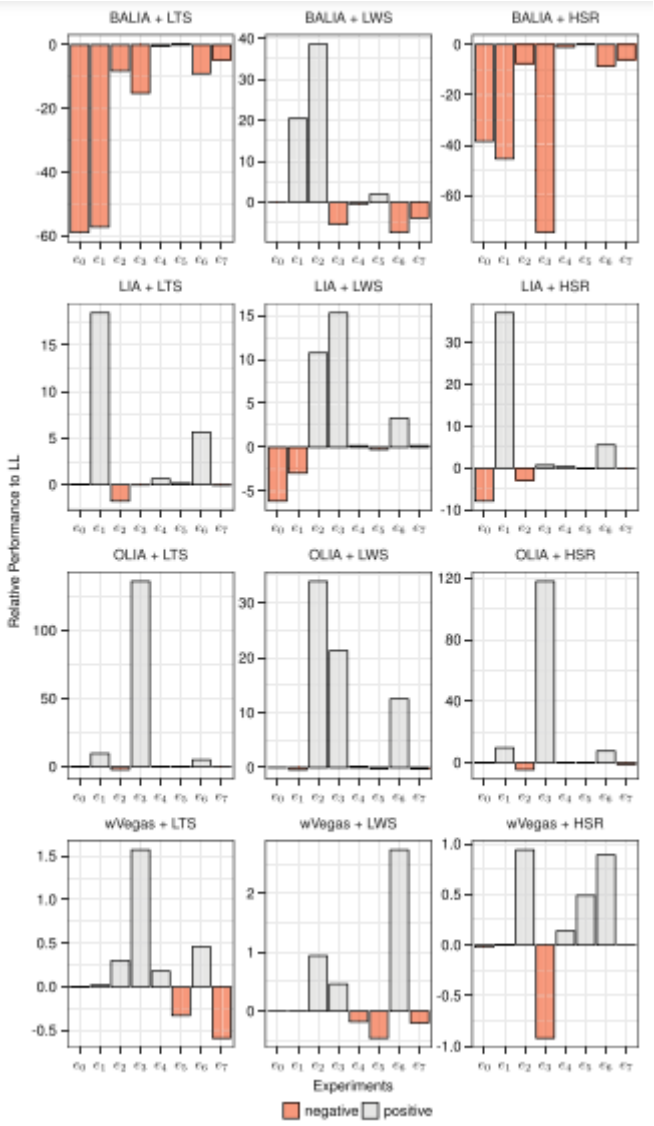

그림 9. 다중 기준 스케줄러(LTS, LWS, HSR)와 단일 기준 스케줄러(LL)의 기준선 사이의 상대적 성능.
Fig 9. Relative performance between multicriteria schedulers and the baseline scheduler of single criterion.

E. 다중 기준이 단일 기준보다 더 나은 Scheduling을 제공할 수 있음
- 그림 9에서 볼 수 있듯이 스케줄링 결정을 내리기 위해 여러 기준을 결합할 때 더 나은 결과를 얻을 수 있다.
- 대부분의 실험과 혼잡 제어에서 스케줄러 LTS, LWS, HSR은 LL 보다 높은 양호한 출력으로 더 높은 성능을 제공한다.
- 해당 결과를 통해 Multipath의 Path Performance가 항상 Path Delay에 반비례하는 것아 아닌 것을 알 수 있다.

- BALIA (Loss Rate에 영향을 받는 것 같았지만, 통계치를 보니 그렇게 큰 성능차이가 있진 않아 보임..)
  - Packet Loss 발생 시
  - 높은 Goodput의 Subflow에서 패킷 ACK 확인 시 덜 적극적(drastic)으로 Subflow Window가 증가
    but, Congestion Window는 더 급격한 감소(factor of 1/2)를 나타낸다.
  - 낮은 Goodput의 Subflow는 덜 급격한 Window 감소(factor of 3/4)와
    but, 더 급격한 윈도우 증가(factor of > 1)를 가진다.
  - 이러한 전략은 높은 Goodput과 낮은 Goodput의 Subflow 사이의 성능 평균화를 위해 시도하는 것이다.

- LIA (??)

- 동일한 Aggressiveness factor가 모든 Subflow를 제어한다.
- Aggressiveness는 Multipath Aggregated Goodput이 정규 SP-TCP를 최상의 경로로 제공하는 방법으로 조정.
- 다중기준 스케줄러들은 LIA에 따른 대부분의 실험에서 LL보다 나은 것을 확인할 수 있다.

- OLIA (Traffic의 Capacity에 영향을 많이 받음, Capacity가 클수록 Congestion Window의 크기가 기하 급수적으로 증가함으로)
  - Congestion Window를 다음 3개의 집합으로 제어한다.
    1. 더 큰 Congestion Window (W)
    2. 두번의 손실 중 더 많은 Byte를 보낸 것 (B)
    3. 더 많은 Bytes를 보냈지만 더 큰 Congestion Window는 없음 (C)
  - OLIA는 Congestion Window의 크기가 C에서 더 빠르고 W에서 더 느리도록 각 Subflow의 Aggressiveness 계수를 조정한다.
  - 결과적으로, W에서 충분히 활용되는 Subflow에서 C에 여유 용량이 있는 경로로 되돌아간다.

- wVegas
  - 스케줄러 성능에 미치는 영향이 크지 않은 Congestion Control Protocol
  - 다중 기준 스케줄러는 LL의 1.5%와 -1% 사이의 상대적인 성능을 제공한다.
  - 여기서 wVegas는 네트워크 정체를 추론하기 위해 패킷 손실 이벤트에 의존하지 않기 때문에 가장 효율적인 정체 제어라는 것을 관찰할 수 있다.
  - 대신, 높은 민감도 전략에 따라 Subflow의 Congestion Window를 증가 및 감소시킨다. 즉, Bottleneck 대기열 Delay를 인식하고 이에 따라 Congestion Window를 능동적으로 조절한다. 그 결과, 대부분의 정체된 Subflow의 트래픽을 경로간에 잘 분산시켜 성능을 향상시킨다.

5. Research Opportunities

A. Bottleneck 인식을 지원하는 End System
  - End System은 End to End 경로의 Bottleneck Link 상태를 인식할 수 없다. 이 문제를 극복하기 위해, End System의 Transport Control Protocol은 더 나은 결정을 내리기 위해 Bottleneck 현상을 추정해야 한다. "명시적 Congestion Notification"은 경로가 혼잡한 경우 Network core가 End System에 명시적으로 알릴 수 있도록 하는 가장 잘 알려진 접근 방식이다. 이는 Bottleneck 라우터에서 패킷을 폐기할 필요 없이 수행되지만, Congestion Notification은 분리되어 거의 정보를 전달하지 못한다. End System은 큐 딜레이, 라우터 버퍼 용량, 전송 속도, 패킷 에러와 손실, 라우터를 공유하는 Flow의 수와 같은 경로상 Bottleneck 현상의 트래픽 Metric을 In-band(Network 장비에 직접 Connect) 방식으로 검색할 수 없다. 네트워크가 Bottleneck Link 상태에 대한 경험적 데이터를 제공한다면 TCP와 MPTCP 모두 현재의 네트워크 Condition을 인지할 수 있을 것이고, 따라서 정확한 전송 결정을 내릴 수 있을 것이다. 이러한 의미에서 "Bottleneck-인식(Aware) End System"을 가능하게 하는 것은 효율적인 전송 솔루션의 공간을 넓히기 위한 관심 연구 기회임을 보여준다.

B. Packet Meta Scheduling
  - Ad-hoc MPTCP Packet Scheduler는 다양한 네트워크 조건 및 사용자 App에서 더 나은 성능을 제공하기 위해 제안되었습니다. 이 Article에서는 패킷 스케줄링의 단일기준 및 다중기준 접근 방식에 대해 본 문서에서 설계된 실험을 기반으로 스케줄러가 각 Multipath Congestion Control에 대해 다른 것보다 더 잘 수행할 수 있는 특정 병목 조건을 확인했다. 이는 그림 7, 8, 9에서 확인할 수 있다. 어떻게든 Bottleneck Link의 경험적 데이터를 캡처하고 분석할 수 있다면, End System의 경로가 어느 조건인지 밝혀내고 그러한 조건에 효율적으로 반응하기 위해 가장 적합한 스케줄러를 선택할 수 있다. 이 방향에서, 관심 연구 기회는 사용 가능한 솔루션에 최적의 후보 솔루션을 선택하기 위한 Meta Scheduler를 설계하는 것이다.
  - 즉, Meta Scheduler란 여러 Scheduler를 동시에 제어하는 감독과 같은 역할을 하는 Scheduler임을 뜻한다. 즉, 특정 Path가 특정 Bottleneck Condition이라면 해당 Condition에 맞는 Scheduler를 Meta Scheduler가 조정하여서 Throughput을 향상시키는 방법.

6. Conclusion

- 패킷 스케줄링은 MPTCP의 핵심 구성요소이다. 올바른 스케줄링 정책은 최종 시스템 처리량을 향상시킬 수 있는 반면, 잘못된 스케줄링 결정은 사용 가능한 네트워크 자원을 이용하지 않아 비효율적이다. 본 논문에서는 일관된 다중 경로 실험 방법론에 의해 지원되는 MPTCP 패킷 스케줄링에 대한 자세한 검토를 제공하였다. 이러한 방법론을 통해 단일기준과 다중기준이라는 두 가지 패킷 스케줄링 클래스에서 솔루션을 평가할 수 있었다. MPTCP Linux 커널 구현과 함께 배포한 네트워크 에뮬레이션 설정에 본 논문에서 설계한 실험을 수행하였다. MPTCP가 실제 시나리오에서 직면하는 현실적인 경로 조건을 재현한 4개의 기존 MPTCP 혼잡 제어를 사용하여 성능 결과를 얻었다. 종합적인 성과 분석을 바탕으로, 우리는 새로운 교훈과 연구 기회에 대해 논의하였다.

참 고 문 헌

- B. Y. L. Kimura, D. C. S. F. Lima and A. A. F. Loureiro, "Packet Scheduling in Multipath TCP: Fundamentals, Lessons, and Opportunities," in IEEE Systems Journal, vol. 15, no. 1, pp. 1445-1457, March 2021, doi: 10.1109/JSYST.2020.2965471.