(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0031891
(43) 공개일자 2025년03월07일

(51) 국제특허분류(Int. Cl.)
H04L 41/142 (2022.01) *H04L 43/0829* (2022.01)
H04L 43/0852 (2022.01) *H04L 43/0888* (2022.01)
(52) CPC특허분류
H04L 41/142 (2022.05)
H04L 43/0829 (2022.05)
(21) 출원번호 10-2023-0113905
(22) 출원일자 2023년08월29일
심사청구일자 2023년08월29일

(71) 출원인
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50
(72) 발명자
임영빈
울산광역시 울주군 언양읍 유니스트길 50
이봉원
울산광역시 울주군 언양읍 유니스트길 50
(74) 대리인
특허법인 무한

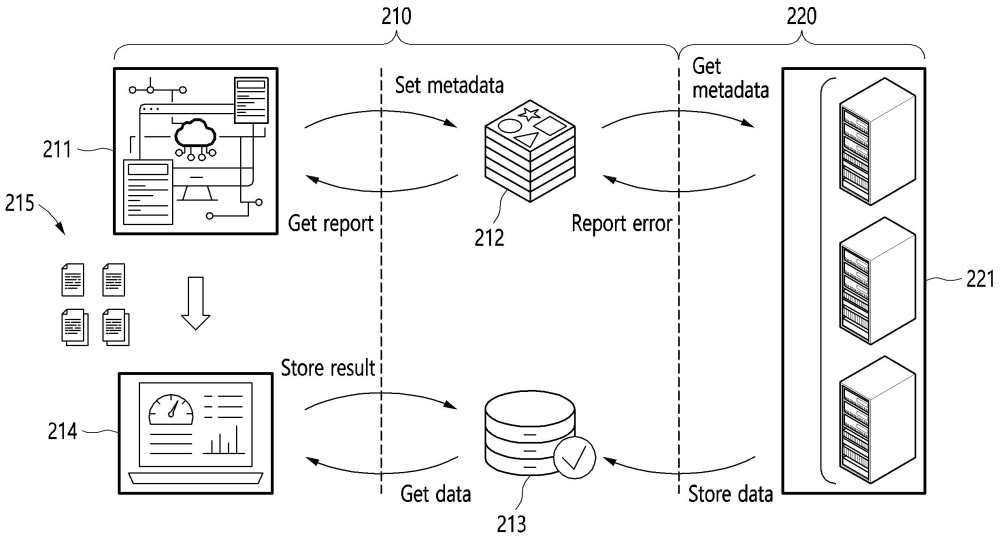
전체 청구항 수 : 총 24 항

(54) 발명의 명칭 **네트워크 성능 분석 방법 및 장치**

(57) 요 약

일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 각 서버에 필요한 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시키는 단계, 상기 설정에 기초하여 수집된 각 서버의 네트워크 관련 정보를 각 서버로부터 수신하는 단계, 상기 수신된 네트워크 관련 정보에 기초하여 서버 각각의 커널 영역 내 각 레이어(layer)의 누적 바이트를 계산하는 단계 및 상기 계산된 누적 바이트에 기초하여 네트워크 성능을 분석하는 단계를 포함할 수 있다.

대 표 도



(52) CPC특허분류
H04L 43/0852 (2022.05)
H04L 43/0888 (2022.05)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711184969
과제번호 2021R1F1A1061346
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 개인기초연구(과기정통부)
연구과제명 세부 성능 정보 측정을 통한 분산화, 가상화된 시스템의 성능 향상 연구
기 여 율 50/100
과제수행기관명 울산과학기술원
연구기간 2023.03.01 ~ 2024.02.29
이 발명을 지원한 국가연구개발사업
과제고유번호 1711193910
과제번호 2021-0-01817-003
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성
연구과제명 하이퍼-컴포저블 데이터센터를 위한 차세대 컴퓨팅 기술 개발
기 여 율 50/100
과제수행기관명 울산과학기술원
연구기간 2023.01.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

프로세서에 의해 수행되는 네트워크 성능 분석 방법에 있어서,

각 서버에 필요한 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시키는 단계;

상기 설정에 기초하여 수집된 각 서버의 네트워크 관련 정보를 각 서버로부터 수신하는 단계;

상기 수신된 네트워크 관련 정보에 기초하여 서버 각각의 커널 영역 내 각 레이어(layer)의 누적 바이트를 계산하는 단계; 및

상기 계산된 누적 바이트에 기초하여 네트워크 성능을 분석하는 단계

를 포함하는 네트워크 성능 분석 방법.

청구항 2

제1항에 있어서,

상기 누적 바이트를 계산하는 단계는,

상기 레이어 중 소켓(socket) 레이어에서 후킹(hooking) 함수의 리턴값에 기초하여 누적 바이트를 계산하는 단계; 및

상기 레이어 중 TCP 레이어, IP 레이어 및 드라이버 레이어에서 패킷의 TCP 헤더 내 시퀀스 번호(Sequence Number)에 기초하여 누적 바이트를 계산하는 단계

를 포함하는 네트워크 성능 분석 방법.

청구항 3

제2항에 있어서,

상기 시퀀스 번호에 기초하여 누적 바이트를 계산하는 단계는,

상기 TCP 레이어의 시작 시점의 시퀀스 번호를 초기 시퀀스 번호로 하여 상기 TCP 레이어, IP 레이어 및 드라이버 레이어의 각각의 계산 시점의 시퀀스 번호와의 차이에 기초하여 누적 바이트를 계산하는 단계

를 포함하는 네트워크 성능 분석 방법.

청구항 4

제1항에 있어서,

상기 네트워크 성능을 분석하는 단계는,

소켓 레이어의 누적 바이트를 기준으로 가장 근접한 위치의 TCP 레이어, IP 레이어 및 드라이버 레이어의 세그먼트(segment)의 누적 바이트를 매칭시켜 네트워크 성능을 분석하는 단계

를 포함하는 네트워크 성능 분석 방법.

청구항 5

제1항에 있어서,

각 서버에 옵저버(observer)를 설치하는 단계; 및

상기 옵저버를 이용하여 상기 각 서버의 커널 영역 내에 코드를 설치하는 단계

를 더 포함하는 네트워크 성능 분석 방법.

청구항 6

제1항에 있어서,

상기 각 서버의 프로그램에 적용시키는 단계는,

각 서버에 필요한 메타데이터를 생성하여 저장시키는 단계; 및

옵저버를 이용하여 상기 저장된 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시키는 단계

를 포함하는 네트워크 성능 분석 방법.

청구항 7

제1항에 있어서,

상기 수신하는 단계는,

상기 각 서버의 상태를 실시간 모니터링하도록 서버가 프로그램을 시작하는 경우, 주기적으로 확인해야 하는 키(key)를 사용하여 상기 네트워크 관련 정보를 수신하는 단계

를 포함하는 네트워크 성능 분석 방법.

청구항 8

제1항에 있어서,

상기 수신하는 단계는,

상기 각 서버의 주기에 따른 오버헤드 상태에 기초하여 상기 네트워크 관련 정보의 수신 간격을 조정하는 단계

를 더 포함하는 네트워크 성능 분석 방법.

청구항 9

제1항에 있어서,

상기 각 서버의 네트워크 관련 정보를 저장시키는 단계; 및

상기 분석된 네트워크 성능을 저장시키는 단계

를 더 포함하는 네트워크 성능 분석 방법.

청구항 10

제1항에 있어서,

상기 각 서버의 시간을 기준 시간으로 동기화시키는 단계

를 더 포함하는 네트워크 성능 분석 방법.

청구항 11

제10항에 있어서,

상기 동기화시키는 단계는,

패킷을 상기 각 서버로 송신한 시간, 상기 각 서버로부터 패킷을 수신한 시간 및 상기 각 서버의 시간 정보에 기초하여 동기화시키는 단계

를 포함하는 네트워크 성능 분석 방법.

청구항 12

제1항에 있어서,

상기 네트워크 성능은,

지연(Delay), 손실률(loss rate), 스루풋(Throughput) 및 시스템-관련(System-Related) 정보인,

네트워크 성능 분석 방법.

청구항 13

각 서버에 필요한 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시키고, 상기 설정에 기초하여 수집된 각 서버의 네트워크 관련 정보를 각 서버로부터 수신하고, 상기 수신된 네트워크 관련 정보에 기초하여 서버 각각의 커널 영역 내 각 레이어의 누적 바이트를 계산하며, 상기 계산된 누적 바이트에 기초하여 네트워크 성능을 분석하는 프로세서

를 포함하는 네트워크 성능 분석 장치.

청구항 14

제13항에 있어서,

상기 프로세서는,

상기 레이어 중 소켓 레이어에서 후킹 함수의 리턴값에 기초하여 누적 바이트를 계산하고, 상기 레이어 중 TCP 레이어, IP 레이어 및 드라이버 레이어에서 패킷의 TCP 헤더 내 시퀀스 번호에 기초하여 누적 바이트를 계산하는,

네트워크 성능 분석 장치.

청구항 15

제14에 있어서,

상기 프로세서는,

상기 TCP 레이어의 시작 시점의 시퀀스 번호를 초기 시퀀스 번호로 하여 상기 TCP 레이어, IP 레이어 및 드라이버 레이어의 각각의 계산 시점의 시퀀스 번호와의 차이에 기초하여 누적 바이트를 계산하는,

네트워크 성능 분석 장치.

청구항 16

제13에 있어서,

상기 프로세서는,

소켓 레이어의 누적 바이트를 기준으로 가장 근접한 위치의 TCP 레이어, IP 레이어 및 드라이버 레이어의 세그먼트의 누적 바이트를 매칭시켜 네트워크 성능을 분석하는,

네트워크 성능 분석 장치.

청구항 17

제13항에 있어서,

상기 프로세서는,

각 서버에 옵저버를 설치하고, 상기 옵저버를 이용하여 상기 각 서버의 커널 영역 내에 코드를 설치하는,

네트워크 성능 분석 장치.

청구항 18

제13항에 있어서,

상기 프로세서는,

각 서버에 필요한 메타데이터를 생성하여 저장시키고, 옵저버를 이용하여 상기 저장된 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시키는,

네트워크 성능 분석 장치.

청구항 19

제13항에 있어서,

상기 프로세서는,

상기 각 서버의 상태를 실시간 모니터링하도록 서버가 프로그램을 시작하는 경우, 주기적으로 확인해야 하는 키를 사용하여 상기 네트워크 관련 정보를 수신하는,

네트워크 성능 분석 장치.

청구항 20

제13항에 있어서,

상기 프로세서는,

상기 각 서버의 주기에 따른 오버헤드 상태에 기초하여 옵저버를 통한 상기 네트워크 관련 정보의 수집 간격을 조정하는,

네트워크 성능 분석 장치.

청구항 21

제13항에 있어서,

상기 프로세서는,

상기 각 서버의 네트워크 관련 정보를 저장시키고, 상기 분석된 네트워크 성능을 저장시키는,

네트워크 성능 분석 장치.

청구항 22

제13항에 있어서,

상기 프로세서는,

상기 각 서버의 시간을 기준 시간으로 동기화시키는,

네트워크 성능 분석 장치.

청구항 23

제22항에 있어서,

상기 프로세서는,

패킷을 상기 각 서버로 송신한 시간, 상기 각 서버로부터 패킷을 수신한 시간 및 상기 각 서버의 시간 정보에 기초하여 동기화시키는,

네트워크 성능 분석 장치.

청구항 24

제13항에 있어서,

상기 네트워크 성능은,

지연, 손실률, 스루풋 및 시스템-관련 정보인,

네트워크 성능 분석 장치.

발명의 설명

기 술 분 야

[0001] 이하, 대용량 시스템을 위한 실시간 네트워크 성능 분석 툴에 관한 기술이 제공된다.

배 경 기 술

[0003] 네트워크(Network)는 증강현실(Augmented Reality, AR), 가상현실(Virtual Reality, VR), 혼합현실(Mixed Reality, MR) 기반 기술, 원격 수업, 라이브 비디오 스트리밍, 멀티 유저 게임, 화상 회의, 산업 자동화, 자율 주행 차량(autonomous vehicle), 커넥티드 의료기기 등의 분야에서 활발히 사용되고 있어 그 중요성이 날로 증대되고 있다. 다양한 분야에서 사용되는 프로그램 및/또는 애플리케이션은 네트워크를 통해 대용량의 데이터가 송수신됨에 따라 원활한 기능을 수행하기 위하여 다양한 네트워크 설계 및 관리가 필요할 수 있다.

[0004] 네트워크 관리자는 네트워크를 설계 및 관리하기 위하여 우선적으로 네트워크의 품질과 성능을 평가하는 네트워크 성능 측정이 필요할 수 있다. 네트워크 성능은 대역폭(bandwidth), 지연(delay), 패킷 손실(packet loss), 지연 변동(jitter), 스루풋(throughput), 신뢰성(reliability) 등에 기초할 수 있다. 예를 들어, 네트워크 관리자는 대역폭을 측정하여 설계 시 필요한 대역폭을 계획할 수 있으며, 이미 대역폭을 형성한 경우, 한정된 대역폭 내에서 발생될 트래픽 부하에 대응하는 네트워크 인프라를 구성할 수 있다. 추가로, 네트워크 관리자는

전술한 네트워크 성능을 측정함에 따라 네트워크에서 발생하는 문제를 식별하여 해결하므로 네트워크 성능을 최적화할 수 있다.

[0005] 위에서 설명한 배경기술은 발명자가 본원의 개시 내용을 도출하는 과정에서 보유하거나 습득한 것으로서, 반드시 본 출원 전에 일반 공중에 공개된 공지기술이라고 할 수는 없다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 한국 공개 특허공보 10-2007-0111661(2007년 11월 22일 공고)에는 점대점 패킷 성능 분석 시스템 및 그 분석 방법이 제시된다.

(특허문헌 0002) 한국 공개 특허공보 10-2009-0094284(2009년09월04일 공고)에는 오버레이 네트워크 노드 간의 네트워크 성능을 측정하기 위한 방법, 장치 및 시스템 이 제시된다.

발명의 내용

해결하려는 과제

[0008] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 TCP 헤더의 시퀀스 번호에 기초한 각 레이어의 누적 바이트를 기반으로 하여 각 레이어 간의 패킷을 매칭시킬 수 있다.

[0009] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 커널 네트워크 내 각 레이어의 대표 함수를 후킹(hooking)하여 네트워크 관련 정보를 수집할 수 있다.

[0010] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 각각의 구성을 기능에 따라 나누어 처리하는 분산 시스템 아키텍처(architecture)를 설계할 수 있다.

[0011] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 옵저버를 통해 각 서버에 동일한 시스템의 코드를 설치하거나 동일한 설정을 적용시킬 수 있다.

[0012] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 패킷을 상기 각 서버로 송신한 시간, 상기 각 서버로부터 패킷을 수신한 시간 및 상기 각 서버의 시간 정보에 기초하여 동기화할 수 있다.

[0013] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 옵저버를 통한 네트워크 관련 정보의 수집 간격을 조정할 수 있다.

[0014] 다만, 기술적 과제는 상술한 기술적 과제들로 한정되는 것은 아니며, 또 다른 기술적 과제들이 존재할 수 있다.

과제의 해결 수단

[0016] 일 실시예에 따른 네트워크 성능 분석 방법은 각 서버에 필요한 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시키는 단계, 상기 설정에 기초하여 수집된 각 서버의 네트워크 관련 정보를 각 서버로부터 수신하는 단계, 상기 수신된 네트워크 관련 정보에 기초하여 서버 각각의 커널 영역 내 각 레이어(layer)의 누적 바이트를 계산하는 단계 및 상기 계산된 누적 바이트에 기초하여 네트워크 성능을 분석하는 단계를 포함할 수 있다.

[0017] 상기 누적 바이트를 계산하는 단계는, 상기 레이어 중 소켓(socket) 레이어에서 후킹(hooking) 함수의 리턴값에 기초하여 누적 바이트를 계산하는 단계, 및 상기 레이어 중 TCP 레이어, IP 레이어 및 드라이버 레이어에서 패킷의 TCP 헤더 내 시퀀스 번호(Sequence Number)에 기초하여 누적 바이트를 계산하는 단계를 포함할 수 있다.

[0018] 상기 시퀀스 번호에 기초하여 누적 바이트를 계산하는 단계는, 상기 TCP 레이어의 시작 시점의 시퀀스 번호를 초기 시퀀스 번호로 하여 상기 TCP 레이어, IP 레이어 및 드라이버 레이어의 각각의 계산 시점의 시퀀스 번호와의 차이에 기초하여 누적 바이트를 계산하는 단계를 포함할 수 있다.

[0019] 상기 네트워크 성능을 분석하는 단계는, 소켓 레이어의 누적 바이트를 기준으로 가장 근접한 위치의 TCP

레이어, IP 레이어 및 드라이버 레이어의 세그먼트(segment)의 누적 바이트를 매칭시켜 네트워크 성능을 분석하는 단계를 포함할 수 있다.

[0020] 상기 방법은 각 서버에 옵저버(observer)를 설치하는 단계, 및 상기 옵저버를 이용하여 상기 각 서버의 커널 영역 내에 코드를 설치하는 단계를 더 포함할 수 있다.

[0021] 상기 각 서버의 프로그램에 적용시키는 단계는, 각 서버에 필요한 메타데이터를 생성하여 저장시키는 단계 및 옵저버를 이용하여 상기 저장된 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시키는 단계를 포함할 수 있다.

[0022] 상기 수신하는 단계는, 상기 각 서버의 상태를 실시간 모니터링하도록 서버가 프로그램을 시작하는 경우, 주기적으로 확인해야 하는 키(key)를 사용하여 상기 네트워크 관련 정보를 수신하는 단계를 포함할 수 있다.

[0023] 상기 수신하는 단계는, 상기 각 서버의 주기에 따른 오버헤드 상태에 기초하여 옵저버를 통한 상기 네트워크 관련 정보의 수집 간격을 조정하는 단계를 더 포함할 수 있다.

[0024] 상기 방법은 상기 각 서버의 네트워크 관련 정보를 저장시키는 단계 및 상기 분석된 네트워크 성능을 저장시키는 단계를 더 포함할 수 있다.

[0025] 상기 방법은 각 서버의 시간을 기준 시간으로 동기화시키는 단계를 더 포함할 수 있다.

[0026] 상기 동기화시키는 단계는, 패킷을 상기 각 서버로 송신한 시간, 상기 각 서버로부터 패킷을 수신한 시간 및 상기 각 서버의 시간 정보에 기초하여 동기화시키는 단계를 포함할 수 있다.

[0027] 상기 네트워크 성능은, 지연(Delay), 손실률(loss rate), 스루풋(Throughput) 및 시스템-관련(System-Related) 정보일 수 있다.

[0028] 일 실시예에 따른 네트워크 성능 분석 장치는 각 서버에 필요한 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시키고, 상기 설정에 기초하여 수집된 각 서버의 네트워크 관련 정보를 각 서버로부터 수신하고, 상기 수신된 네트워크 관련 정보에 기초하여 서버 각각의 커널 영역 내 각 레이어의 누적 바이트를 계산하며, 상기 계산된 누적 바이트에 기초하여 네트워크 성능을 분석하는 프로세서를 포함할 수 있다.

[0029] 상기 프로세서는, 상기 레이어 중 소켓 레이어에서 후킹 함수의 리턴값에 기초하여 누적 바이트를 계산하고, 상기 레이어 중 TCP 레이어, IP 레이어 및 드라이버 레이어에서 패킷의 TCP 헤더 내 시퀀스 번호에 기초하여 누적 바이트를 계산할 수 있다.

[0030] 상기 프로세서는, 상기 TCP 레이어의 시작 시점의 시퀀스 번호를 초기 시퀀스 번호로 하여 상기 TCP 레이어, IP 레이어 및 드라이버 레이어의 각각의 계산 시점의 시퀀스 번호와의 차이에 기초하여 누적 바이트를 계산할 수 있다.

[0031] 상기 프로세서는, 소켓 레이어의 누적 바이트를 기준으로 가장 근접한 위치의 TCP 레이어, IP 레이어 및 드라이버 레이어의 세그먼트의 누적 바이트를 매칭시켜 네트워크 성능을 분석할 수 있다.

[0032] 상기 프로세서는, 각 서버에 옵저버를 설치하고, 상기 옵저버를 이용하여 상기 각 서버의 커널 영역 내에 코드를 설치할 수 있다.

[0033] 상기 프로세서는, 상기 각 서버에 필요한 메타데이터를 생성하여 저장시키고, 옵저버를 이용하여 상기 저장된 메타 데이터에 기초한 설정을 각 서버의 프로그램에 적용시킬 수 있다.

[0034] 상기 프로세서는, 상기 각 서버의 상태를 실시간 모니터링하도록 서버가 프로그램을 시작하는 경우, 주기적으로 확인해야 하는 키를 사용하여 상기 네트워크 관련 정보를 수신할 수 있다.

[0035] 상기 프로세서는, 상기 각 서버의 주기에 따른 오버헤드 상태에 기초하여 옵저버를 통한 상기 네트워크 관련 정보의 수집 간격을 조정할 수 있다.

[0036] 상기 프로세서는, 상기 각 서버의 네트워크 관련 정보를 저장시키고, 상기 분석된 네트워크 성능을 저장시킬 수 있다.

[0037] 상기 프로세서는, 상기 각 서버의 시간을 기준 시간으로 동기화시킬 수 있다.

[0038] 상기 프로세서는, 패킷을 상기 각 서버로 송신한 시간, 상기 각 서버로부터 패킷을 수신한 시간 및 상기 각 서버의 시간 정보에 기초하여 동기화시킬 수 있다.

[0039] 상기 네트워크 성능은, 지연, 손실률, 스루풋 및 시스템-관련 정보일 수 있다.

발명의 효과

[0041] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 TCP 헤더의 시퀀스 번호에 기초한 각 레이어의 누적 바이트를 기반으로 하여 각 레이어 간의 패킷을 매칭시킴에 따라 TCP 레이어, IP 레이어 및 드라이버 레이어에서 발생할 수 있는 패킷 재전송, 순서 문제 등을 예방함과 동시에 각 레이어의 이상여부에 대하여 정확하게 분석할 수 있다.

[0042] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 커널 네트워크 내 각 레이어의 대표 함수를 후킹하여 네트워크 관련 정보를 수집함에 따라 네트워크 패킷이 지나는 모든 경로에 대한 성능을 세부적으로 분석할 수 있다.

[0043] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 각각의 구성을 기능에 따라 나누어 처리하는 분산 시스템 아키텍처를 설계함에 따라 몇개의 명령어만으로도 네트워크 성능을 분석하는 툴을 자동으로 실행할 수 있으며, 대용량 시스템 구성이 용이할 수 있다.

[0044] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 옵저버를 통해 각 서버에 동일한 시스템의 코드를 설치하거나 동일한 설정을 적용시킴에 따라 코드 수정과 같은 특정 작업의 필요 없이 용이하게 네트워크 성능 측정이 가능할 수 있다.

[0045] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 패킷을 상기 각 서버로 송신한 시간, 상기 각 서버로부터 패킷을 수신한 시간 및 상기 각 서버의 시간 정보에 기초하여 동기화함에 따라 전체 서버의 시간을 동일하고 정확하게 동기화시킬 수 있다.

[0046] 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 옵저버를 통한 네트워크 관련 정보의 수집 간격을 조정함에 따라 네트워크 관련 정보 수집 시 서버의 오버 헤드를 감소시킬 수 있다.

도면의 간단한 설명

[0048] 도 1은 일 실시예에 따른 네트워크 성능 분석 시스템의 블록도를 도시한다.

도 2는 일 실시예에 따른 네트워크 성능 분석 시스템의 구조를 예시적으로 설명한다.

도 3은 일 실시예에 따른 네트워크 성능 분석 시스템의 구조에서 옵저버(observer)가 설치된 서버의 구조를 예시적으로 설명한다.

도 4는 일 실시예에 따른 네트워크 성능 분석 장치를 이용하여 각 레이어에 대한 누적 바이트를 계산하여 매칭시키는 방식을 예시적으로 설명한다.

도 5는 일 실시예에 따른 네트워크 성능 분석 장치를 이용하여 각 레이어에서 정보를 수집하는 포인트를 예시적으로 설명한다.

도 6은 일 실시예에 따른 네트워크 성능 분석 방법의 흐름도를 도시한다.

도 7은 일 실시예에 따른 네트워크 성능 분석 방법에서 수집 간격을 조정하는 방식을 예시적으로 설명한다.

도 8은 일 실시예에 따른 네트워크 성능 분석 방법에서 각 서버의 시간을 동기화시키는 방식을 예시적으로 설명한다.

발명을 실시하기 위한 구체적인 내용

[0049] 실시예들에 대한 특정한 구조적 또는 기능적 설명들은 단지 예시를 위한 목적으로 개시된 것으로서, 다양한 형태로 변경되어 구현될 수 있다. 따라서, 실제 구현되는 형태는 개시된 특정 실시예로만 한정되는 것이 아니며, 본 명세서의 범위는 실시예들로 설명한 기술적 사상에 포함되는 변경, 균등물, 또는 대체물을 포함한다.

[0050] 제1 또는 제2 등의 용어를 다양한 구성요소들을 설명하는데 사용될 수 있지만, 이런 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 해석되어야 한다. 예를 들어, 제1 구성요소는 제2 구성요소로 명

명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0051] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다.

[0052] 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 설명된 특징, 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함으로 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0053] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 해당 기술분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0054] 이하, 실시예들을 첨부된 도면들을 참조하여 상세하게 설명한다. 첨부 도면을 참조하여 설명함에 있어, 도면 부호에 관계없이 동일한 구성 요소는 동일한 참조 부호를 부여하고, 이에 대한 중복되는 설명은 생략하기로 한다.

[0055] 네트워크 성능 측정 툴은 핑(Ping), 트레이스라우트(Traceroute), Iperf, ELEMENT 등을 포함할 수 있다.

[0056] 핑 및 트레이스라우트는 ICMP(Internet Control Message Protocol) 프로토콜을 통해 지연 시간을 측정할 수 있고, Iperf는 서버, 클라이언트를 통해 스루풋을 측정할 수 있다. 핑, 트레이스라우트 및 Iperf는 전송 레이어(transport layer) 보다 하위 레이어들(lower layers)에 대한 네트워크 지연(network delay)을 측정할 수 있다. ELEMENT는 전송 레이어보다 상위 레이어인 애플리케이션 레이어(application layer) 및 전송 레이어에 대한 엔드호스트 지연(Endhost delay)을 측정할 수 있다.

[0057] 전술한 툴들을 포함한 네트워크 성능 측정 툴들은 전송 레이어보다 하위인 레이어들을 측정하거나 애플리케이션 레이어 및 전송 레이어를 측정하는 것과 같이 국한적인 구간에서의 성능만을 측정하고 있다. 또한, 대용량 시스템(Large-Scale System)에 대한 네트워크 성능 측정에 있어서도 많은 서버의 네트워크 성능을 실시간으로 분석하는 툴들이 존재하지만, 네트워크 경로의 모든 구간을 분석하지 못하는 문제가 역시 존재한다. 이에 대용량 시스템에 대한 네트워크의 모든 구간을 세부적으로 측정하기 위하여 전술한 문제점을 해결한 새로운 네트워크 성능 측정 툴의 개발이 요구되고 있다. 네트워크 성능은 대역폭(bandwidth), 손실률(loss rate), 지연(delay), 패킷 손실(packet loss), 지연 변동(jitter), 스루풋(throughput), 신뢰성(reliability) 등에 기초하여 판단될 수 있다. 특히, 이중에서 지연은 네트워크를 통해 데이터를 전송하는 데 걸리는 시간으로써, 네트워크 속도를 결정하는 주요한 원인이다. 이에 지연을 정확하게 측정하는 것은 네트워크 성능 측정에 있어서 매우 중요한 목적일 수 있다. 그러므로, 대용량 시스템에서 최적의 네트워크 성능을 유지하기 위해서는 네트워크의 모든 구간에 대하여 지연을 세부적으로 측정하는 네트워크 성능 측정 툴이 필요할 수 있다.

[0058] 또한, 전술한 툴들을 포함한 네트워크 성능 측정 툴들은 사용하기 위하여 코드 수정과 같은 특정 작업이 필수적으로 필요할 수 있다. 그러나, 이러한 작업은 네트워크 성능을 자동으로 측정하는데 어려움을 발생시키며, 대용량 시스템의 경우, 더 많은 작업량이 필요로 하므로 대용량 시스템의 규모 및 성능에 한계가 있을 수 있다.

[0059] 이러한 문제들을 해결하기 위하여 일 실시예에 따른 네트워크 성능 분석 방법 및 장치는 각각의 구성을 기능에 따라 나누어 처리하는 분산 시스템 아키텍처를 설계하고 시퀀스 번호에 기초한 각 레이어의 누적 바이트(예: 계산 시점까지 전송된 페이로드(payload)의 총 바이트 수)를 계산하여 레이어 간 매칭시키는 방식을 이용함에 따라 네트워크의 모든 구간의 성능을 자동으로 분석할 수 있다.

[0060] 도 1은 일 실시예에 따른 네트워크 성능 분석 시스템의 블록도를 도시한다.

[0061] 일 실시예에 따른 네트워크 성능 분석 시스템(100)은 네트워크 분석 장치(110) 및 서버(120)를 포함할 수 있다.

[0062] 네트워크 성능 분석 장치(110)는 프로세서(111), 메모리(112) 및 통신부(113)를 포함할 수 있다.

[0063] 프로세서(111)는 통신부(113)를 통해 수신된 데이터에 기초하여 서버(120)의 네트워크 성능을 측정할 수 있다. 우선, 프로세서(111)는 각 서버에 필요한 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시킬 수 있다. 메타데이터는 Redis, MySQL 접속 주소 등에 기초하여 생성될 수 있다. 메타데이터는 각 서버에서 네트워크 관

련 정보를 수집하는 주기, 수집하는 방법, 수집된 정보를 전송하는 경로 등을 지정하는 정보를 포함할 수 있다. 이에 각 서버는 메타데이터의 정보에 기초하여 프로그램이 네트워크 관련 정보에 대한 동작을 수행할 수 있도록 프로그램 설정을 구성할 수 있다. 프로세서(111)는 설정에 기초하여 수집된 각 서버의 네트워크 관련 정보를 각 서버로부터 수신할 수 있다. 프로세서(111)는 수신된 네트워크 관련 정보에 기초하여 서버 각각의 커널 영역(kernel space) 내 각 레이어의 누적 바이트를 계산하며, 계산된 누적 바이트에 기초하여 네트워크 성능을 분석할 수 있다. 네트워크 관련 정보는 누적 바이트를 계산하기 위한 각 레이어에서의 계산 시점까지 전송된 페이로드의 총 바이트 수 뿐만 아니라 각 레이어에서의 손실률, 스루풋 및 시스템-관련 정보를 산출하기 위한 정보를 포함할 수 있다.

[0064] 메모리(112)는 휘발성 메모리 장치 또는 불휘발성 메모리 장치로 구현될 수 있다. 다양한 실시예에 따르면, 프로세서(111)는 메모리(112)에 저장된 컴퓨터로 읽을 수 있는 코드(예를 들어, 소프트웨어) 및 프로세서(111)에 의해 유발된 인스트럭션(instruction)들을 실행할 수 있다. 프로세서(111)는 목적하는 동작들(desired operations)을 실행시키기 위한 물리적인 구조를 갖는 회로를 가지는 하드웨어로 구현된 데이터 처리 장치일 수 있다. 목적하는 동작들은 예를 들어, 프로그램에 포함된 코드(code) 또는 인스트럭션들(instructions)을 포함할 수 있다.

[0065] 통신부(113)는 유선 또는 무선으로 복수개로 구성된 서버(120)와 연결될 수 있다. 이를 통해 네트워크 성능 분석 장치(110)는 서버(120)에 데이터를 송신하거나 서버(120)로부터 데이터를 수신할 수 있다.

[0066] 서버(120)는 하나 이상의 클라이언트(client)와 연결되며 대용량 시스템 구조에 의해 복수개 구성될 수 있다. 서버(120)는 클라이언트와 데이터를 송수신하며 검색, 처리 및 저장 등의 기능을 수행할 수 있다. 서버(120)는 클라우드 서버(cloud server), 웹 서버(web server), 애플리케이션 서버(application server), 데이터베이스 서버(database server), 파일 서버(file server) 등일 수 있으며, 이에 한정되지 않는다.

[0067] 도 2는 일 실시예에 따른 네트워크 성능 분석 시스템의 구조를 예시적으로 설명한다.

[0068] 도 3은 일 실시예에 따른 네트워크 성능 분석 시스템의 구조에서 옵저버(observer)가 설치된 서버의 구조를 예시적으로 설명한다.

[0069] 네트워크 성능 분석 장치(210)(예: 네트워크 성능 분석 장치(110))는 프로세서(111)를 기능에 따라 예시적으로 구성을 나눌 수 있다. 프로세서(111)는 네트워크 성능 분석을 위한 하나 이상의 명령어를 실행함으로써, 매니저(manager)(211) 및 애널라이저(analyzer)(214)를 구현할 수 있다. 매니저(211) 및 애널라이저(214)는 소프트웨어 모듈 또는 프로그램으로서, 매니저(211) 및 애널라이저(214)의 동작은 프로세서(111)에 의해 수행될 수 있다. 또한, 네트워크 성능 분석 장치(210)는 메모리(112)를 기능에 따라 예시적으로 구성을 나눌 수 있다. 메모리(112)는 키-밸류 스토어(KVS, key-value store)(212) 및 데이터베이스(Database)(213)를 포함할 수 있다. 이는 네트워크 성능 분석 장치(210)는 각각의 구성을 기능에 따라 나누어 처리하는 분산 시스템 아키텍처(architecture)가 설계됨에 따라 몇개의 명령어만으로도 네트워크 성능을 분석하는 툴을 자동으로 실행할 수 있다. 분산 시스템 아키텍처는 매니저(211), 애널라이저(214), 키-밸류 스토어(212), 데이터베이스(213) 및 옵저버(221)에 한정되지 않고 그 구성이 증감되거나 감소될 수 있다.

[0070] 매니저(211)는 키-밸류 스토어(212)를 이용하여 각 서버(220)를 제어하고 관리할 수 있다. 이를 위해 매니저(211)는 키-밸류 스토어(212)를 통해 각 서버(220)에 옵저버(observer)(221)를 설치할 수 있다. 또한, 매니저(211)(예: 프로세서(111))는 설치된 옵저버(221)를 이용하여 각 서버의 커널 영역(320) 내에 코드(330)를 설치할 수 있다. 각 서버 내에는 설치된 코드에 기초하여 따라 네트워크 관련 정보를 수집하는 프로그램이 형성될 수 있다. 커널 영역은 네트워크 스택(Network Stack) 중 하나로 유저 영역(user space)에서 실행되는 애플리케이션을 통해 수신된 데이터를 처리하기 위해 내부 처리 구성에 전달하거나 처리된 데이터를 애플리케이션으로 전달할 수 있다. 커널 영역은 간략하게 소켓 레이어, TCP 레이어, IP 레이어 및 드라이버 레이어로 구성되어 순차적으로 각 레이어에 데이터가 통과하면서 전달될 수 있다. 예를 들어, 일부 레이어에서 데이터 전달 시 지연이 발생되면 전체적인 네트워크 성능이 하락될 수 있다. 이에 커널 영역 내 전체 레이어에 대한 네트워크 성능(예: 지연)을 확인하는 것이 중요할 수 있다.

[0071] 매니저(211)는 각 서버(220)에 필요한 메타데이터(Metadata)를 생성할 수 있다. 메타데이터는, 예를 들어, yaml 파일 형식으로된 Redis, MySQL 접속 주소 등에 기초하여 생성될 수 있다. 매니저(211)(예: 프로세서(111))는 각 서버에 필요한 메타데이터를 키-밸류 스토어(212)에 저장시킬 수 있다(set metadata). 매니저(211)(예: 프로세서(111))는 옵저버(221)를 이용하여 키-밸류 스토어(212)로부터 수신된 메타데이터에 기초한

설정을 각 서버의 프로그램에 적용시킬 수 있다(get metadata). 옵저버(221)는 각 서버의 종류 및 특성과 관계없이 동일한 시스템의 코드(330)를 설치하거나 동일한 설정을 적용시킬 수 있다. 이를 통해 매니저(211)는 코드 수정과 같은 특정 작업의 필요 없이 용이하게 네트워크 성능 측정이 가능할 수 있다.

[0072] 옵저버(221)는 설정이 적용된 프로그램을 통해 커널 영역(320) 내에 네트워크와 관련된 정보를 수집할 수 있다. 옵저버(221)는 수집된 네트워크 관련 정보를 링 버퍼(Ring buffer)를 사용하여 유저 영역(user space)(310)으로 전달할 수 있다(set data). 링 버퍼는 커널 영역과 유저 영역 사이에 공유 메모리를 만들어 데이터를 전달할 수 있다. 이에 이벤트(event) 발생(예: 데이터 전송) 시 커널 영역에서 링 버퍼에 데이터를 넣으면, 유저 영역에서 링 버퍼에 존재하는 데이터를 가져감에 따라 데이터가 전달될 수 있다. 옵저버(221)는 이러한 링 버퍼를 이용함에 따라 네트워크 관련 정보 전달 시 데이터 전달에 의해 발생되는 오버헤드가 감소되고 증가된 전달 속도를 이용할 수 있다. 옵저버(221)는 유저 영역(310)으로 전달된 네트워크 관련 정보를 콜백(Callback)을 통해 업데이트하여(update data) 데이터베이스에 저장되도록 전달할 수 있다.

[0073] 또한, 옵저버(221)는 네트워크 관련 정보 수집 시 오류(error)가 발생하면, 발생된 오류를 키-밸류 스토어(212)로 보고할 수 있다(report error). 키-밸류 스토어(212)는 보고된 오류를 매니저(211)에게 전달하고(get report), 매니저(211)는 보고된 오류를 확인할 수 있다.키-밸류 스토어(212)는 매니저(211)의 제어에 따라 각 서버(220)에 프로그램을 실행시키고 서버 관리에 필요한 정보를 전달할 수 있다. 키-밸류 스토어(212)는 키 기반의 값을 저장하는 데이터베이스 중에 하나이다. 키-밸류 스토어(212)는 서버(220)가 키(key)가 입력된 요청 패킷(request packet)을 통해 데이터를 요청하면, 입력된 키에 대응되는 값(value)을 응답 패킷(response packet)에 입력하여 서버(220)에게 제공할 수 있다. 이에 매니저(211)(예: 프로세서(111))는 각 서버(220)의 상태를 실시간 모니터링하도록 서버(220)가 프로그램을 시작하는 경우, 키-밸류 스토어(212)를 통해 주기적으로 확인해야 하는 키를 사용하여 네트워크 관련 정보를 수신할 수 있다.

[0074] 전술한 바와 같이, 매니저(211)(예: 프로세서(111))는 각 서버의 네트워크 관련 정보를 데이터베이스(213)에 저장할 수 있다(store data). 애널라이저(214)(예: 프로세서(111))는 누적 바이트를 계산하도록 데이터베이스(213)로부터 저장된 네트워크 관련 정보를 획득할 수 있다(get data). 이는 애널라이저(214)가 각 서버에서 추출한 복수의 데이터들을 사용하여 서버 내부 및 서버 간의 네트워크 성능을 분석하는 것을 의미할 수 있다. 애널라이저(214)(예: 프로세서(111))는 네트워크 관련 정보에 기초하여 네트워크 성능을 분석할 수 있다. 애널라이저(214)(예: 프로세서(111))는 분석된 네트워크 성능을 데이터베이스(213)에 저장할 수 있다(store result).

[0075] 데이터베이스(213)는 구조화된 데이터 구조를 사용할 수 있다. 이로 인해 데이터베이스(213)는 중복된 데이터 저장을 방지할 수 있고 체계적이면서 용이하게 데이터를 관리할 수 있다. 또한, 데이터베이스(213)는 인덱스(index) 기법을 활용하여 데이터베이스 안의 데이터를 빠르게 쿼리함으로써 빠른 처리 속도를 확보할 수 있다.

[0076] 애널라이저(214)는 각 네트워크 계층 간의 패킷을 시퀀스 번호 기반 누적 바이트 매칭 알고리즘을 사용하여 매칭함에 따라 네트워크 성능을 분석할 수 있다. 또한, 애널라이저(214)는 네트워크 성능 분석을 위하여 매니저(211)로부터 시간 정보(215)를 수신할 수 있다.

[0077] 도 4는 일 실시예에 따른 네트워크 성능 분석 장치를 이용하여 각 레이어에 대한 누적 바이트를 계산하여 매칭시키는 방식을 예시적으로 설명한다.

[0078] 애널라이저(214)(예: 프로세서(111))는 레이어 중 소켓(socket) 레이어에서 후킹(hooking) 함수의 리턴값에 기초하여 누적 바이트를 계산할 수 있다. 애널라이저(214)(예: 프로세서(111))는 레이어 중 TCP 레이어, IP 레이어 및 드라이버 레이어에서 패킷의 TCP 헤더 내 시퀀스 번호(Sequence Number)에 기초하여 누적 바이트를 계산할 수 있다. 소켓 레이어는 TCP 레이어, IP 레이어 및 드라이버 레이어와 달리 TCP 헤더가 없으므로 후킹 함수에 기초하여 누적 바이트를 계산할 수 있다.

[0079] 패킷의 TCP 헤더는 출발지 포트(Source Port), 목적지 포트(Destination Port), 시퀀스 번호, 승인 번호(Acknowledgement Number), 데이터 오프셋(Data Offset), 예약(Reserved), 플래그 비트(Flag bit), 윈도우 사이즈(Window size), 체크섬(Checksum), 긴급 포인터(Urgent Pointer) 및 옵션(Options) 필드를 포함할 수 있다. 여기서, 시퀀스 번호는 현재까지 전송한 페이로드(Payload)의 바이트 수를 더한 값(예: 처음부터 전송된 페이로드의 길이)을 나타낼 수 있다. 시퀀스 번호의 사용은 TCP 레이어, IP 레이어 및 드라이버 레이어에서 발생할 수 있는 패킷 재전송(retransmission), 순서(ordering) 문제 등을 예방할 수 있다.

[0080] 애널라이저(214)(예: 프로세서(111))는 TCP 레이어의 시작 시점의 시퀀스 번호를 초기 시퀀스 번호(init_seq_num)로 하여 TCP 레이어, IP 레이어 및 드라이버 레이어의 각각의 계산 시점의 시퀀스 번호

(cur_seq_num)와의 차이에 기초하여 누적 바이트(sent_bytes)를 계산할 수 있다. 애널라이저(214)(예: 프로세서(111))는 TCP 레이어의 초기 시퀀스 번호인 시작 시점의 시퀀스 번호를 저장시킴에 따라 각 레이어마다 계산 시점까지 전송한 누적 바이트를 계산할 수 있다. 예를 들어, TCP 레이어에서 세그먼트 1(segment 1)의 시퀀스 번호가 2100이고, 세그먼트 2(segment 2)의 시퀀스 번호가 2180이면, 누적 바이트는 TCP 레이어의 세그먼트 1과 세그먼트 2의 차이에 따라 80일 수 있다. 또한, TCP 레이어에서 세그먼트 1의 시퀀스 번호가 2100이고, IP 레이어에서 세그먼트 2의 시퀀스 번호가 2200이면, TCP 레이어의 세그먼트 1과 IP 레이어의 세그먼트 2의 차이에 따라 100일 수 있다.

[0081] TCP 레이어는 최대 전송 단위(MTU, Maximum Transmission Unit) 및 윈도우 사이즈에 따라 다양한 세그먼트 크기로 분할할 수 있다. 또한, 하위 레이어인 IP 레이어 및 드라이버 레이어는 TCP 레이어와 동일한 세그먼트 크기로 구성될 수 있다. 이는 소켓 레이어의 세그먼트 크기와 TCP 레이어, IP 레이어 및 드라이버 레이어의 세그먼트 크기가 각기 다름을 의미할 수 있다. 이에 애널라이저(214)(예: 프로세서(111))는 소켓 레이어에서의 누적 바이트를 기준으로 가장 근접한 위치의 TCP 레이어, IP 레이어 및 드라이버 레이어의 세그먼트의 누적 바이트를 매칭시켜 네트워크 성능을 분석할 수 있다. 예를 들어, 소켓 레이어의 세그먼트 1 내 (410) 지점에서의 누적 바이트 및 TCP 레이어의 세그먼트 1 내 (420)지점에서의 누적 바이트가 120이지만, IP 레이어의 세그먼트 1 내 (430)지점에서의 누적 바이트 및 드라이버 레이어의 세그먼트 1 내 (440)지점에서의 누적 바이트는 210인 경우, IP 레이어에서 지연이 발생된 것으로 분석할 수 있다. 네트워크 성능은 지연(Delay)에 한정되는 것이 아니며, 손실률(loss rate), 스루풋(Throughput) 및 시스템-관련(System-Related) 정보 또한 가능할 수 있다. 예를 들어, 손실률은 패킷의 전송에 기초하여 발생되므로, 애널라이저(214)(예: 프로세서(111))는 소켓 레이어를 제외한 TCP 레이어, IP 레이어 및 드라이버 레이어에 대하여 손실률을 계산할 수 있다. 이에 애널라이저(214)는 TCP 레이어, IP 레이어 및 드라이버 레이어의 시퀀스 번호에 기초하여 손실률을 계산할 수 있다. 패킷 손실의 발생은 패킷의 재전송 여부를 통해 확인할 수 있으므로, 애널라이저(214)는 시퀀스 번호를 통해 현재 패킷이 재전송하는 패킷인지를 확인할 수 있다.

[0082] 도 5는 일 실시예에 따른 네트워크 성능 분석 장치를 이용하여 각 레이어에서 정보를 수집하는 포인트를 예시적으로 설명한다.

[0083] 프로세서(111)는 옵저버(221)를 이용하여 각 레이어에서 정보를 수집할 수 있다. 옵저버(221)는 서버의 오버헤드를 줄이기 위해 각 레이어마다 대표 함수 하나를 선택하여 그 함수를 후킹(hooking)함으로써 네트워크 관련 정보를 수집할 수 있다. 이에 프로세서(111)는 네트워크 패킷이 지나는 모든 경로에 대한 성능을 세부적으로 분석할 수 있다.

[0084] 예를 들어, 소켓 레이어(0)에서는 데이터 전송에 사용되는 함수인 sock_sendmsg 또는 kernel_sendpage를 대표 함수로 사용할 수 있다. TCP 레이어(1)에서는 데이터 전송에 사용되는 함수인 tcp_transmit_skb를, IP 레이어(2)에서는 데이터 전송에 사용되는 함수인 ip_local_out를 대표 함수로 사용할 수 있다. 드라이버 레이어(DEVICE(3))에서는 네트워크 인터페이스를 통한 데이터 전송에 사용되는 함수인 dev_queue_xmit를 대표 함수로 사용할 수 있다.

[0085] 드라이버 레이어(DEVICE(3))에서 전송한 데이터는 또 다른 디바이스의 드라이버 레이어(DEVICE(7))에서 수신할 수 있다. 이 때 드라이버 레이어(DEVICE(7))는 네트워크 인터페이스를 통해 데이터를 수신하는 함수인 netif_receive_skb를 대표 함수를 사용할 수 있다. IP 레이어(6)에서는 데이터 수신에 사용되는 함수인 ip_local_deliver 함수를, TCP 레이어(5)에서는 데이터 수신에 사용되는 함수인 tcp_v4_rcv를 대표 함수로 사용할 수 있다. 대표 함수는 각 레이어의 동작 상태에 따라 변경될 수 있으며, 전술한 함수로 한정되지 않는다.

[0086] 이에 프로세서(111)는 DEVICE(3) 및 DEVICE(7) 내에서의 각각 소켓 레이어(0)와 TCP 레이어(1)(501), TCP 레이어(1)와 IP 레이어(2)(502), IP 레이어(2)와 드라이버 레이어(DEVICE(3))(503), 소켓 레이어(4)와 TCP 레이어(5)(505), TCP 레이어(5)와 IP 레이어(6)(506), IP 레이어(6)와 드라이버 레이어(DEVICE(7))(507) 및 드라이버 레이어(DEVICE(3))와 드라이버 레이어(DEVICE(7))(511) 사이에 연결된 네트워크 성능을 분석할 수 있다. 또한, 프로세서(111)는 호스트(Host) 내에서 브릿지(Bridge)를 사용하는 경우의 구간((504), (508)), 현재 서버가 버추얼 머신(Virtual Machine)인 경우에 버추얼 머신과 호스트 사이의 구간((509), (510)) 및 호스트 간의 구간(511)의 네트워크 성능을 분석할 수 있다.

[0087] 또한, 프로세서(111)는 각 서버의 주기에 따른 오버헤드 상태에 기초하여 옵저버(221)를 통한 네트워크 관련 정보의 수집 간격을 조정할 수 있고, 각 서버의 시간을 기준 시간으로 동기화시킬 수 있다. 프로세서(111)는 패킷을 각 서버로 송신한 시간, 각 서버로부터 패킷을 수신한 시간 및 각 서버의 시간 정보에 기초하여 동기화시

킬 수 있다.

[0088] 도 6은 일 실시예에 따른 네트워크 성능 분석 방법의 흐름도를 도시한다.

[0089] 단계(610)에서, 프로세서는 각 서버에 필요한 메타데이터에 기초한 설정을 각 서버의 프로그램에 적용시킬 수 있다.

[0090] 단계(620)에서, 프로세서는 설정에 기초하여 수집된 각 서버의 네트워크 관련 정보를 각 서버로부터 수신할 수 있다. 프로세서는 각 서버의 상태를 실시간 모니터링하도록 서버가 프로그램을 시작하는 경우, 주기적으로 확인해야 하는 키를 사용하여 네트워크 관련 정보를 수신할 수 있다. 또한, 프로세서는 각 서버의 주기에 따른 오버헤드 상태에 기초하여 옵저버를 통한 네트워크 관련 정보의 수집 간격을 조정할 수 있는데, 이는 도7에서 후술한다.

[0091] 단계(630)에서, 프로세서는 수신된 네트워크 관련 정보에 기초하여 서버 각각의 커널 영역 내 각 레이어의 누적 바이트를 계산할 수 있다. 프로세서는 레이어 중 소켓 레이어에서 후킹 함수의 리턴값에 기초하여 누적 바이트를 계산할 수 있다. 또한, 프로세서는 레이어 중 TCP 레이어, IP 레이어 및 드라이버 레이어에서 패킷의 TCP 헤더 내 시퀀스 번호에 기초하여 누적 바이트를 계산할 수 있다. 프로세서는 TCP 레이어의 시작 시점의 시퀀스 번호를 초기 시퀀스 번호로 하여 TCP 레이어, IP 레이어 및 드라이버 레이어의 각각의 계산 시점의 시퀀스 번호와의 차이에 기초하여 누적 바이트를 계산할 수 있다.

[0092] 단계(640)에서, 프로세서는 계산된 누적 바이트에 기초하여 네트워크 성능을 분석할 수 있다. 프로세서는 소켓 레이어의 누적 바이트를 기준으로 가장 근접한 위치의 TCP 레이어, IP 레이어 및 드라이버 레이어의 세그먼트의 누적 바이트를 매칭시켜 네트워크 성능을 분석할 수 있다. 네트워크 성능은 지연, 손실률, 스루풋 및 시스템-관련 정보(예: CPU, 메모리 사용 등)일 수 있다.

[0093] 단계(610) 이전에, 프로세서는 각 서버에 옵저버를 설치하고, 옵저버를 이용하여 각 서버의 커널 영역 내에 코드를 설치할 수 있다. 단계(610)에서, 프로세서는 각 서버에 필요한 메타데이터를 생성하여 키-밸류 스토어에 저장시킬 수 있다. 프로세서는 옵저버를 이용하여 키-밸류 스토어에 저장된 메타 데이터에 기초한 설정을 각 서버의 프로그램에 적용시킬 수 있다.

[0094] 또한, 단계(610) 이전에, 프로세서는 각 서버의 시간을 기준 시간으로 동기화시킬 수 있고, 이는 도8에서 후술한다.

[0095] 단계(620) 이후에, 프로세서는 각 서버의 네트워크 관련 정보를 데이터베이스에 저장시킬 수 있고, 단계(640) 이후에, 분석된 네트워크 성능을 데이터베이스에 저장시킬 수 있다.

[0096] 도 7은 일 실시예에 따른 네트워크 성능 분석 방법에서 수집 간격을 조정하는 방식을 예시적으로 설명한다.

[0097] 프로세서는 각 서버의 수집 주기에 따라 옵저버를 통해 네트워크 관련 정보를 수집하는 간격을 조정할 수 있다. 이는, 예를 들어, 제1 수집 주기에서는 소켓 레이어 기준 하나의 세그먼트가 수집되는 기본적인 간격으로 TCP 레이어, IP 레이어 및 드라이버 레이어에서는 두 개의 세그먼트가 수집되는 간격으로 설정될 수 있다. 이에 소켓 레이어의 세그먼트 1(segment 1) 내 (710)지점에서 네트워크 성능을 분석할 수 있다. 그러나, 수집된 정보에서 서버 내 오버헤드가 발생한 것으로 분석되거나 오버헤드가 발생할 것으로 예상되는 경우, 제2 수집 주기에서는 소켓 레이어 기준 두개의 세그먼트가 수집되는 간격으로 증가시켜 서버 내 오버 헤드 발생을 감소시킬 수 있다. 또한, TCP 레이어, IP 레이어 및 드라이버 레이어에서는 네 개의 세그먼트(720)가 수집되는 간격으로 조정될 수 있다. 이에 소켓 레이어의 세그먼트 2(segment 2) 내 (720)지점에서 네트워크 성능을 분석할 수 있다. 또한, 제3 수집 주기에서는 제2 수집 주기에서 수집된 정보에 기초하여 수집 간격을 늘리거나 줄어들도록 동적으로 조정할 수 있다.

[0098] 도 8은 일 실시예에 따른 네트워크 성능 분석 방법에서 각 서버의 시간을 동기화시키는 방식을 예시적으로 설명한다.

[0099] 프로세서의 매니저(810)는 패킷에 각 서버의 정보를 세팅하여 로컬 호스트(localhost) 인터페이스(예: lo)로 패킷을 전달할 수 있다. 로컬 호스트 인터페이스 내 설치된 코드에 기초한 프로그램(830)은 해당 패킷을 캡처 후 각 서버에 패킷이 전달될 수 있도록 아이피(ip), 맥 어드레스(mac address), 체크섬(checksum) 등을 계산하여 해당 패킷을 다른 인터페이스(예: eth1)로 리디렉션(redirect)할 수 있다. 이러한 패킷은 각 서버의 옵저버(820)로 포워딩(packet forwarding)되어 이를 수신한 각 서버 내 설치된 코드에 기초한 프로그램(830)은 해당 패킷을 캡처 후 자신의 서버 시간을 패킷에 저장한 후 해당 패킷을 인터페이스(예: eth1)로 리디렉션(redirec

t)할 수 있다. 해당 패킷은 프로세서의 매니저(810)로 전달될 수 있다(packet retrieve).

[0100] 프로세서의 매니저(810)는 전달된 패킷의 정보(예: 패킷을 각 서버로 송신한 시간, 각 서버로부터 패킷을 수신한 시간 및 각 서버의 시간 정보)에 기초하여 각 서버의 시간을 프로세서의 매니저(810)의 시간(예: 기준 시간)으로 동기화시킬 수 있다. 이에 매니저(810)는 네트워크 성능 측정 시에만 기준 시간으로 동기화된 각 서버의 시간을 애널라이저에 전달할 수 있다.

[0101] 프로세서의 매니저(810)는 각 서버로부터 패킷을 수신한 시간(MRT)에서 패킷을 각 서버로 송신한 시간(MST)을 제외한 평균((MRT-MST)/2)을 의미하는 단방향 지연(One_Way_Delay)을 산출할 수 있다. 그후, 매니저(810)는 각 서버에서 패킷을 수신한 시간(ST)에서 단방향 지연(One_Way_Delay)을 제외(ST-One_Way_Delay)한 매니저가 패킷을 보낸 시간에 대응하는 각 서버의 시간(SMST)을 산출할 수 있다. 매니저(810)는 패킷을 각 서버로 송신한 시간(MST)에서 매니저가 패킷을 보낸 시간에 대응하는 각 서버의 시간(SMST)을 제외(MST-SMST)한 시간 차(TD)를 산출할 수 있다. 매니저(810)는 데이터 내 각 서버에 대한 시간(St_i)에서 시간 차(TD)를 제외(St_i-TD)한 프로세서의 매니저(810)의 시간으로 변경된 각 서버의 시간(STOMT)를 산출할 수 있다.

[0102] 이를 통해 프로세서의 매니저(810)는 옵저버(820)를 통해 서버의 커널 네트워크 스택을 직접적으로 거치지 않고 우회함으로써 패킷 송수신으로 인해 발생하는 서버 내 오버헤드를 최소화하면서 각 서버의 시간을 정확하게 획득할 수 있다. 프로세서는 정확하게 획득된 각 서버의 시간에 기초하여 네트워크 성능을 분석함에 따라 높은 신뢰성을 가진 결과를 획득할 수 있다.

[0103] 이상에서 설명된 실시예들은 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치, 방법 및 구성요소는, 예를 들어, 프로세서, 컨트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0104] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 저장될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0105] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있으며 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0106] 위에서 설명한 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 또는 복수의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

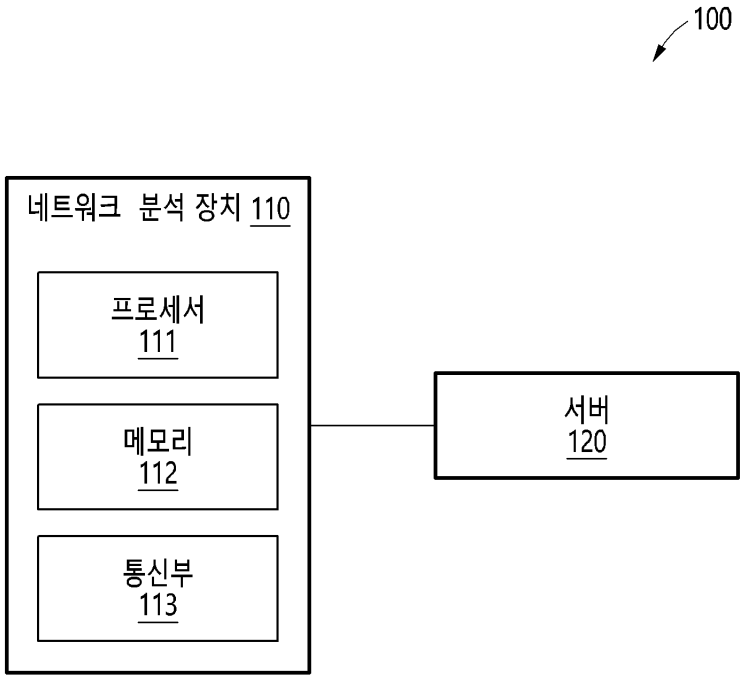
[0107] 이상과 같이 실시예들이 비록 한정된 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면

이를 기초로 다양한 기술적 수정 및 변형을 적용할 수 있다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

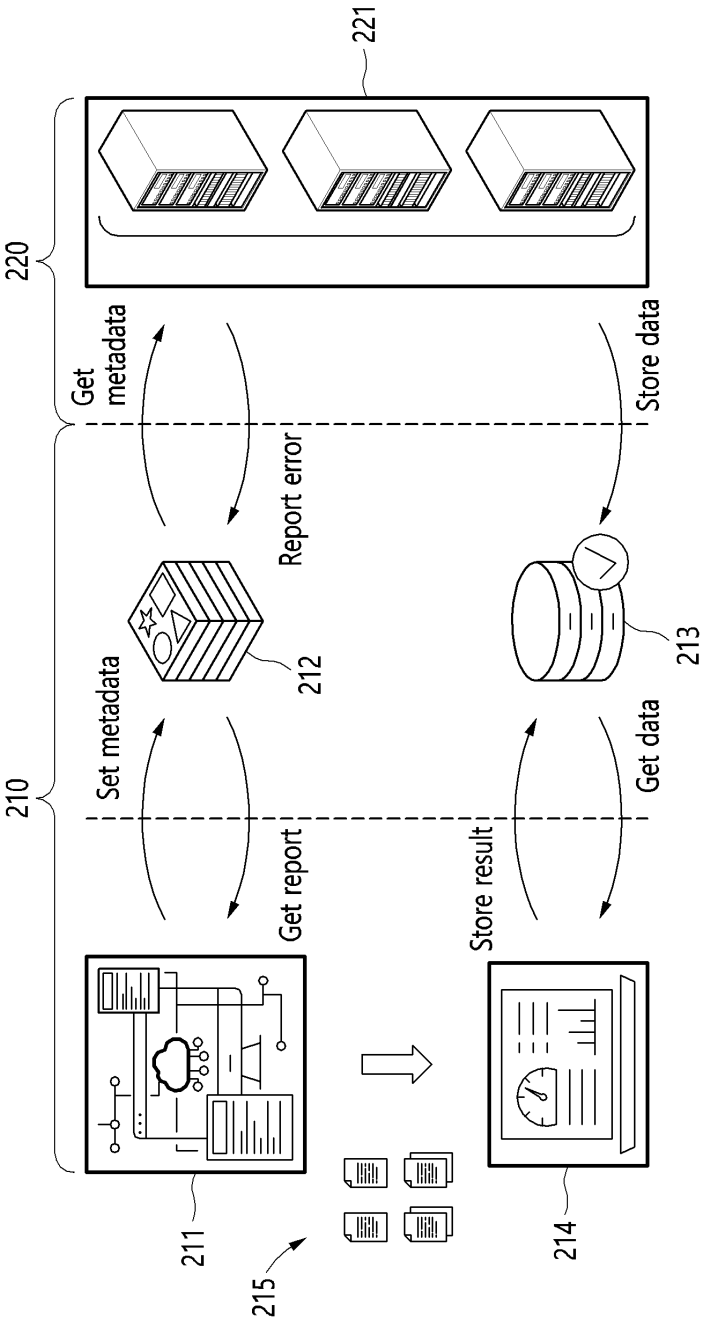
[0108] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

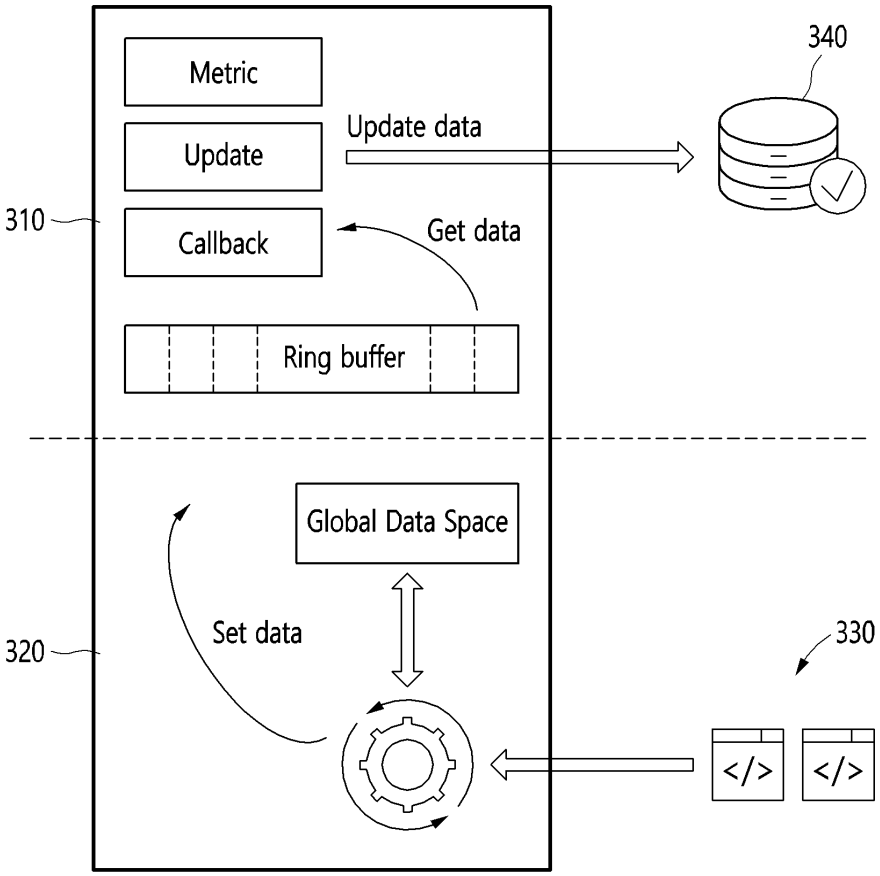
도면1



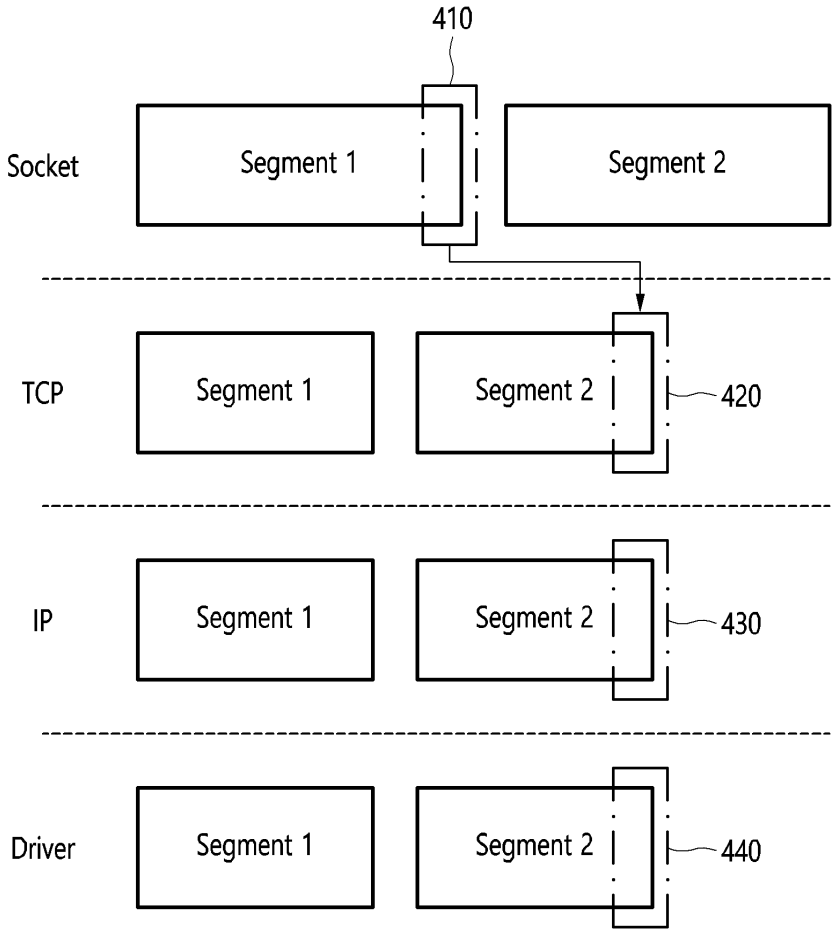
도면2


221
220
Get metadata
Store data
Report error
212
213
Set metadata
Get data
210
Get report
Store result
211
215
214

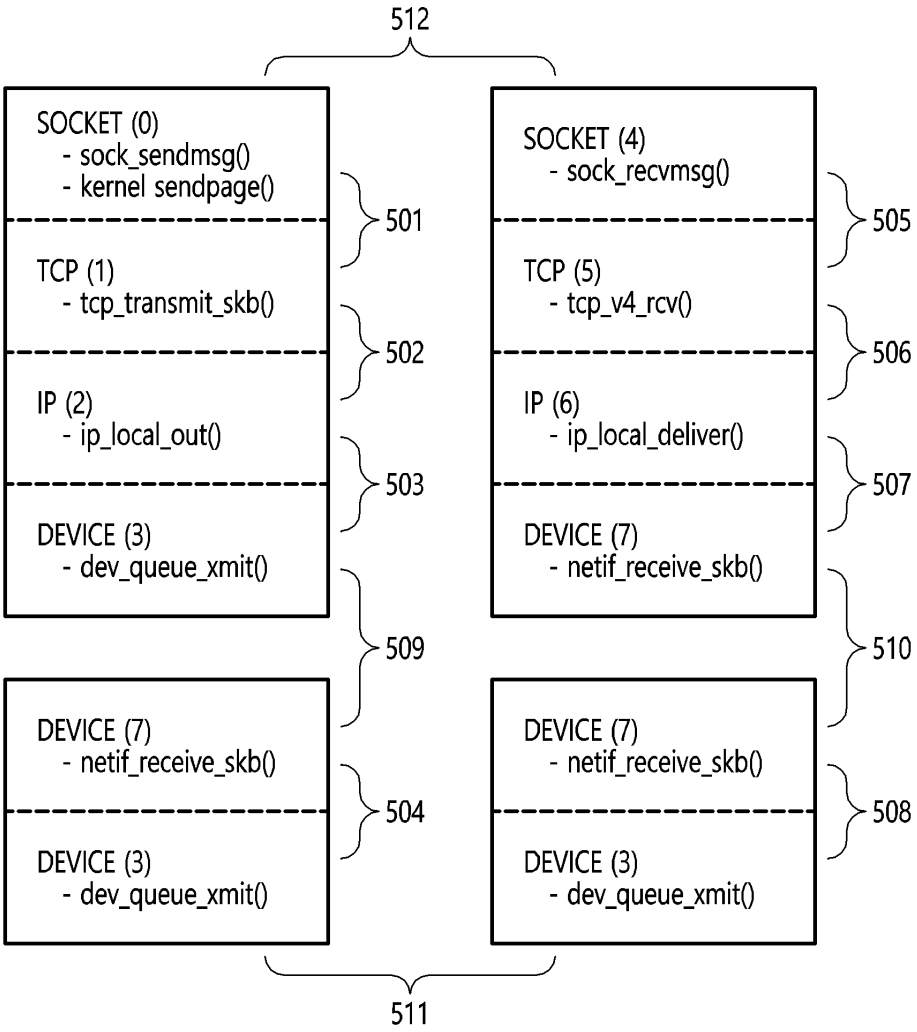
도면3


Metric
Update
Callback
Ring buffer
Update data
Get data
340
310
Global Data Space
Set data
320
330
</>
</>

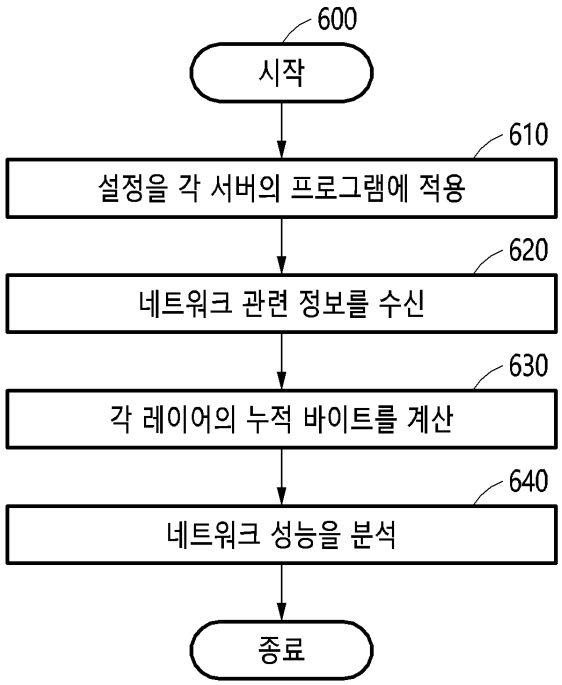
도면4


410
Socket
Segment 1
Segment 2
TCP
Segment 1
Segment 2
420
IP
Segment 1
Segment 2
430
Driver
Segment 1
Segment 2
440

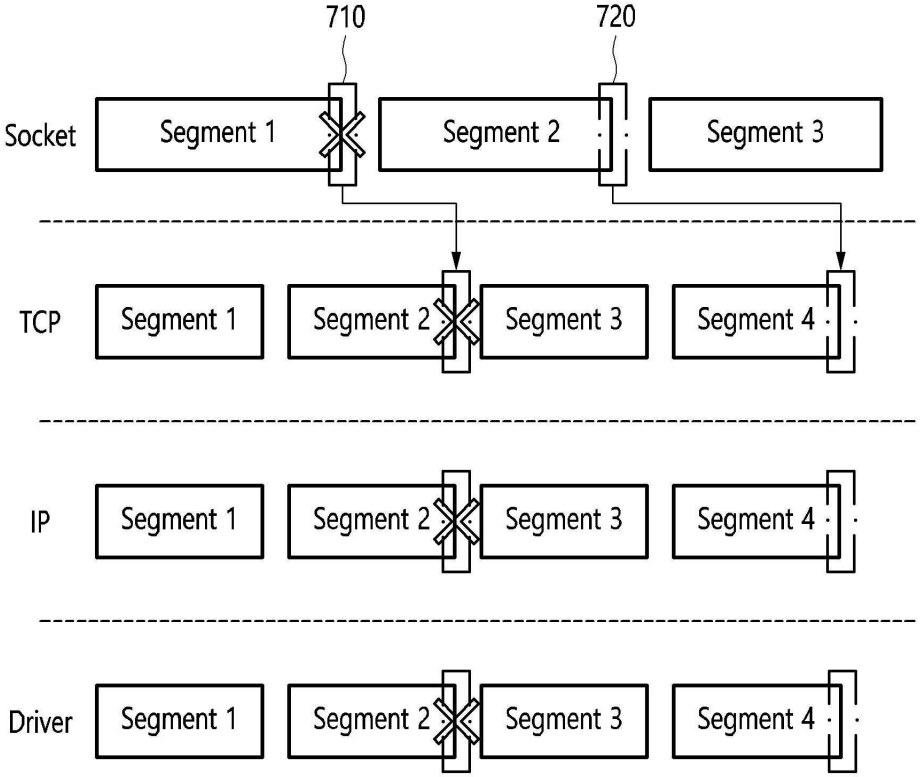
도면5


512
SOCKET (0)
- sock_sendmsg()
- kernel sendpage()
TCP (1)
- tcp_transmit_skb()
IP (2)
- ip_local_out()
DEVICE (3)
- dev_queue_xmit()
501
502
503
509
DEVICE (7)
- netif_receive_skb()
DEVICE (3)
- dev_queue_xmit()
504
511
SOCKET (4)
- sock_recvmsg()
TCP (5)
- tcp_v4_rcv()
IP (6)
- ip_local_deliver()
DEVICE (7)
- netif_receive_skb()
505
506
507
510
DEVICE (7)
- netif_receive_skb()
DEVICE (3)
- dev_queue_xmit()
508

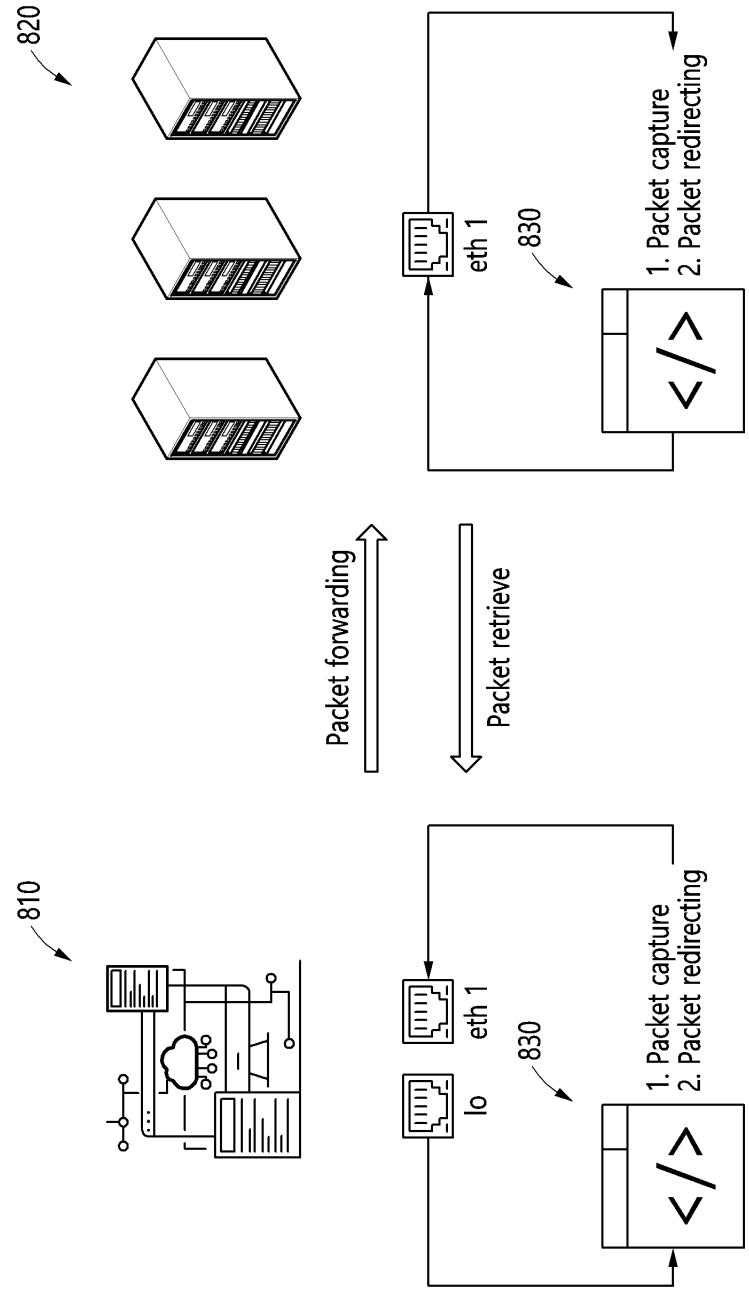
도면6


600
시작
610
설정을 각 서버의 프로그램에 적용
620
네트워크 관련 정보를 수신
630
각 레이어의 누적 바이트를 계산
640
네트워크 성능을 분석
종료

도면7


710
720
Socket
Segment 1
Segment 2
Segment 3
TCP
Segment 1
Segment 2
Segment 3
Segment 4
IP
Segment 1
Segment 2
Segment 3
Segment 4
Driver
Segment 1
Segment 2
Segment 3
Segment 4

도면8


820
810
eth 1
lo
830
1. Packet capture
2. Packet redirecting
Packet forwarding
Packet retrieve