(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0049388
(43) 공개일자 2025년04월11일

(51) 국제특허분류(Int. Cl.)
H04L 47/28 (2022.01) *H04L 1/1607* (2023.01)
H04L 47/32 (2022.01) *H04L 47/62* (2022.01)
H04L 47/70 (2022.01) *H04L 69/28* (2022.01)
H04L 69/326 (2022.01)
(52) CPC특허분류
H04L 47/28 (2022.05)
H04L 1/1607 (2023.01)
(21) 출원번호 10-2025-7008267
(22) 출원일자(국제) 2023년08월17일
심사청구일자 2025년03월12일
(85) 번역문제출일자 2025년03월12일
(86) 국제출원번호 PCT/US2023/030492
(87) 국제공개번호 WO 2024/039794
국제공개일자 2024년02월22일
(30) 우선권주장
63/373,016 2022년08월19일 미국(US)
63/503,349 2023년05월19일 미국(US)

(71) 출원인
테슬라, 인크.
미국 텍사스 78725 오스틴 테슬라 로드 1
(72) 발명자
퀸넬, 에릭 씨.
미국 텍사스 78725 오스틴 테슬라 로드 1 테슬라, 인크. 내
윌리엄스, 더글러스 알.
미국 텍사스 78725 오스틴 테슬라 로드 1 테슬라, 인크. 내
(뒷면에 계속)
(74) 대리인
특허법인 무한

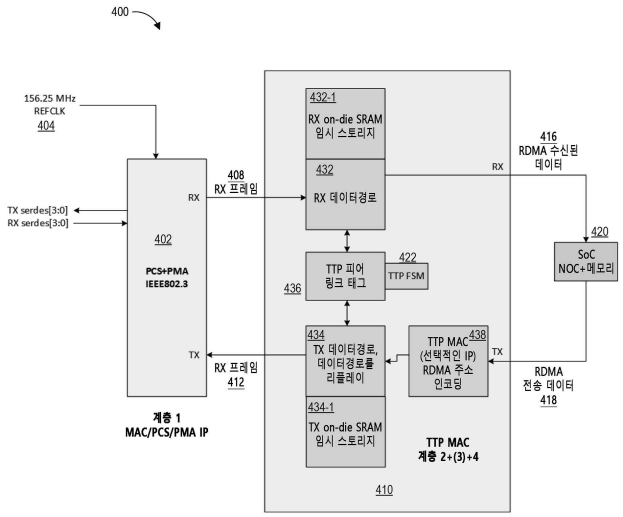
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **이더넷을 위한 링크 타이머**

(57) 요 약

본 개시는 소프트웨어-제어 메커니즘들의 도움 없이 전송 계층을 이용하여 이더넷-기반 네트워크에서 통신하는 시스템들 및 방법들에 관한 것이다. 일부 실시예들에서, 제1 노드는 전송 계층 하드웨어 전용 이더넷 프로토콜에 따라 전송된 패킷들을 리플레이하도록 결정하는 하드웨어 링크 타이머를 포함한다. 하드웨어 링크 타이머는 제1 노드에 의해 확립된 하나 이상의 링크와 연관된 타이밍 및 상태 정보를 저장하는 선입선출(FIFO) 메모리를 포함할 수 있다. 하드웨어 링크 타이머는 시간 주기에 따라 틱하는 하나 이상의 링크와 연관된 타이머를 더 포함할 수 있다.

대 표 도 - 도4



(52) CPC특허분류
H04L 47/32 (2022.05)
H04L 47/6225 (2022.05)
H04L 47/826 (2013.01)
H04L 69/28 (2022.05)
H04L 69/326 (2022.05)
(72) 발명자
셩, 크리스토퍼
미국 텍사스 78725 오스틴 테슬라 로드 1 테슬라, 인크. 내
나바로 허타도, 헤라르도
미국 텍사스 78725 오스틴 테슬라 로드 1 테슬라, 인크. 내

명 세 서

청구범위

청구항 1

이더넷 기반 네트워크에서 패킷들을 전송하는 제1 노드에 있어서,

상기 제1 노드는:

하나 이상의 프로세서를 포함하고,

상기 하나 이상의 프로세서는,

복수의 링크들과 연관된 타이밍 및 상태 정보를 저장하는 선입선출(FIFO) 메모리 - 상기 제1 노드는 이더넷 프로토콜을 이용하여 상기 복수의 링크들을 통해 하나 이상의 다른 노드에 패킷들을 전송함 -;

시간 주기에 따라 틱하는 타이머 - 상기 타이머는 상기 복수의 링크들과 연관됨 -; 및

논리 회로를 포함하고,

상기 논리 회로는:

상기 타이머의 각각의 틱들에 기초하여 상기 FIFO 메모리의 엔트리들에 액세스하고; 및

상기 복수의 링크들의 제1 링크와 연관된 상기 타이밍 및 상태 정보에 기초하여, 상기 제1 링크와 연관된 적어도 하나의 패킷을 리플레이하도록 결정하고,

상기 이더넷 프로토콜은 손실성인 것인,

제1 노드.

청구항 2

제1항에 있어서,

상기 논리 회로는,

라운드-로빈 방식으로 상기 FIFO 메모리의 상기 엔트리들에 액세스하는,

제1 노드.

청구항 3

제1항에 있어서,

상기 타이머는,

상기 FIFO 메모리의 상기 엔트리들과 연관된 활성 링크들의 개수에 기초하여 상기 시간 주기를 조정하고,

상기 활성 링크들은,

상기 복수의 링크들에 포함되는,

제1 노드.

청구항 4

제1항에 있어서,

상기 논리 회로는,

상기 복수의 링크들의 제2 링크와 연관된 상기 타이밍 및 상태 정보에 기초하여, 상기 제2 링크와 연관된 패킷들을 폐기하도록 결정하는,

제1 노드.

청구항 5

제4항에 있어서,

상기 제2 링크와 연관된 상기 패킷들은,

상기 제1 노드의 로컬 스토리지에 저장되고,

상기 논리 회로는,

상기 로컬 스토리지로 하여금 상기 제2 링크와 연관된 상기 패킷을 폐기하기로 결정한 것에 응답하여 상기 제2 링크와 연관된 상기 패킷들을 삭제하도록 하는,

제1 노드.

청구항 6

제1항에 있어서,

상기 논리 회로는,

상기 복수의 링크들의 제2 링크와 연관된 상기 타이밍 및 상태 정보에 기초하여, 상기 제2 링크를 닫도록 결정하는,

제1 노드.

청구항 7

제1항에 있어서,

상기 복수의 링크들의 상기 제1 링크와 연관된 상기 타이밍 및 상태 정보는,

패킷들을 리플레이하기 위한 임계 기간 동안 상기 제1 노드에 의해 상기 제1 링크와 연관된 상기 적어도 하나의 패킷을 수신했다는 확인응답이 수신되지 않았음을 나타내는,

제1 노드.

청구항 8

이더넷 기반 통신을 위한 제1 노드에 있어서,

상기 제1 노드는:

전송 계층 하드웨어 전용 이더넷 프로토콜을 구현하는 하나 이상의 프로세서를 포함하고,

상기 전송 계층 하드웨어 전용 이더넷 프로토콜은 손실성이고,

상기 하나 이상의 프로세서는,

상기 전송 계층 하드웨어 전용 이더넷 프로토콜에 따라 전송된 패킷들을 리플레이하도록 결정하는 하드웨어 링크 타이머를 포함하는,

제1 노드.

청구항 9

제8항에 있어서,

상기 제1 노드는,

상기 전송 계층 하드웨어 전용 이더넷 프로토콜에 따라 제1 링크를 통해 제1 복수의 패킷들을 전송하고 제2 링크를 통해 제2 복수의 패킷들을 전송하고,

상기 하드웨어 링크 타이머는:

선입선출(FIFO) 메모리의 제1 엔트리에 상기 제1 링크와 연관된 타이밍 및 상태 정보를 저장하고, 상기 FIFO 메모리의 제2 엔트리에 상기 제2 링크와 연관된 타이밍 및 상태 정보를 저장하는 상기 FIFO 메모리를 포함하는,

제1 노드.

청구항 10

제9항에 있어서,

상기 하드웨어 링크 타이머는,

시간 주기에 따라 틱하는 다수의 링크들과 연관된 타이머를 포함하고,

상기 하드웨어 링크 타이머는,

상기 타이머의 라운드-로빈 방식의 틱들로 상기 FIFO 메모리의 엔트리들에 액세스하고,

상기 엔트리들은,

상기 제1 엔트리 및 상기 제2 엔트리를 포함하는,

제1 노드.

청구항 11

제10항에 있어서,

상기 하드웨어 링크 타이머는,

상기 FIFO 메모리의 엔트리들과 연관된 활성 링크들의 개수에 기초하여 상기 시간 주기를 조정하고,

상기 활성 링크들은,

상기 제1 링크 및 상기 제2 링크를 포함하는,

제1 노드.

청구항 12

제10항에 있어서,

상기 하드웨어 링크 타이머는:

상기 FIFO 메모리의 상기 제1 엔트리에 저장된 상기 제1 링크와 연관된 상기 타이밍 및 상태 정보에 기초하여, 상기 제1 복수의 패킷들 중 적어도 일부를 리플레이하도록 결정하고; 및

상기 FIFO 메모리의 상기 제2 엔트리에 저장된 상기 제2 링크와 연관된 상기 타이밍 및 상태 정보에 기초하여, 상기 제2 복수의 패킷들을 폐기하도록 결정하는,

제1 노드.

청구항 13

제12항에 있어서,

상기 제2 복수의 패킷들은,

상기 제1 노드의 로컬 스토리지에 저장되고,

상기 하드웨어 링크 타이머는,

상기 로컬 스토리지로 하여금 상기 제2 복수의 패킷들을 폐기하기로 결정한 것에 응답하여 상기 제2 복수의 패킷들을 삭제하도록 하는,

제1 노드.

청구항 14

제12항에 있어서,

상기 제1 링크와 연관된 상기 타이밍 및 상태 정보는,

패킷들을 리플레이하기 위한 임계 기간 동안 상기 제1 노드에 의해 상기 제1 복수의 패킷들 중 하나를 수신했다는 확인응답이 수신되지 않았음을 나타내는,

제1 노드.

청구항 15

이더넷 기반 네트워크의 제1 노드에서 구현된 컴퓨터-구현 방법에 있어서,

상기 컴퓨터-구현 방법은:

상기 제1 노드의 선입선출(FIFO) 메모리에 복수의 링크들과 연관된 타이밍 및 상태 정보를 저장하는 단계 - 상기 제1 노드는 이더넷 프로토콜을 이용하여 상기 복수의 링크들을 통해 하나 이상의 다른 노드에 패킷들을 전송함 -;

하드웨어 타이머의 각각의 틱들에 기초하여 상기 FIFO 메모리의 엔트리들에 액세스하는 단계; 및

상기 복수의 링크들의 제1 링크와 연관된 상기 타이밍 및 상태 정보에 기초하여 상기 제1 링크와 연관된 적어도 하나의 패킷을 리플레이하도록 결정하는 단계를 포함하고,

상기 이더넷 프로토콜은 손실성인 것인,

컴퓨터-구현 방법.

청구항 16

제15항에 있어서,

상기 FIFO 메모리의 상기 엔트리들은,

라운드-로빈 방식으로 액세스되는,

컴퓨터-구현 방법.

청구항 17

제15항에 있어서,

상기 FIFO 메모리의 상기 엔트리들과 연관된 활성 링크들의 개수에 기초하여 상기 하드웨어 타이머의 시간 주기를 조정하는 단계를 더 포함하고,

상기 활성 링크들은,

상기 복수의 링크들에 포함되는,

컴퓨터-구현 방법.

청구항 18

제15항에 있어서,

상기 복수의 링크들의 제2 링크와 연관된 상기 타이밍 및 상태 정보에 기초하여, 상기 제2 링크와 연관된 패킷들을 폐기하도록 결정하는 단계를 더 포함하는,

컴퓨터-구현 방법.

청구항 19

제15항에 있어서,

상기 제1 링크와 연관된 상기 적어도 하나의 패킷이 리플레이되도록 하는 단계를 더 포함하는.

컴퓨터-구현 방법.

청구항 20

제15항에 있어서,

상기 복수의 링크들의 상기 제1 링크와 연관된 상기 타이밍 및 상태 정보는,

패킷들을 리플레이하기 위한 임계 기간 동안 상기 제1 노드에 의해 상기 제1 링크와 연관된 상기 적어도 하나의 패킷을 수신했다는 확인응답이 수신되지 않았음을 나타내는,

컴퓨터-구현 방법.

발명의 설명

기 술 분 야

[0001] 본 출원은 2022년 8월 19일에 출원된 "TRANSPORT PROTOCOL FOR ETHERNET"라는 명칭의 미국 가특허 출원 번호 63/373,016 건의 정규 출원이고, 이에 대한 우선권을 주장하며, 해당 미국 가특허 출원 번호 63/373,016 건의 기술적 개시는 본 명세서에 참조로 전체적으로 모든 목적들을 위해 포함된다. 본 출원은 2023년 5월 19일에 출원된 "TRANSPORT PROTOCOL FOR ETHERNET"라는 명칭의 미국 가특허 출원 번호 63/503,349건의 정규 출원이고, 이에 대한 우선권을 주장하며, 해당 미국 가특허 출원 번호 63/503,349 건의 기술적 개시는 본 명세서에 참조로 전체적으로 모든 목적들을 위해 포함된다.

[0002] 본 개시는 네트워크들을 통한 통신들을 용이하게 하는 시스템들 및 방법들에 관한 것이다. 보다 구체적으로, 본 개시의 실시예들은 이더넷 기반 네트워크(Ethernet based network)들을 통한 통신을 위한 하드웨어를 이용하여 구현 가능한 흐름 제어 프로토콜(flow control protocol)들에 관한 것이다.

배 경 기 술

[0003] 전기전자기술자협회(Institute of Electrical and Electronics Engineers; IEEE)는, 일반적으로 이더넷으로 알려진 IEEE 802.3 표준을 포함하는 IEEE 802라고 알려진 근거리 통신망(local area network; LAN)들에 대한 다양한 표준들을 제공하였다. IEEE 802.3 이더넷 표준에는 물리적 매체 인터페이스(physical media interface)들(이더넷 케이블(Ethernet cable)들, 광섬유(fiber optic)들, 백플레인(backplane)들 등)에 대한 사양(specification)들이 있지만, 통신 흐름 제어들에 대한 사양은 없다. TCP/IP, RoCE, 또는 InfiniBand와 같은 프로토콜들은 패브릭 흐름 제어(fabric flow control)들을 가속화할 수 있다. TCP/IP 프로토콜들은 일반적으로 밀리초 단위(order)의 지연 시간(latency)들을 가지는 반면, RoCE 또는 InfiniBand는 시스템을 지나치게 제한할 수 있는 무손실(lossless) 및 스케일링 사양(scaling specification)들을 갖는다.

[0004] 고성능 컴퓨팅(High-performance computing; HPC) 및 인공지능(artificial intelligence; AI) 훈련 데이터 센터(training data center)들이 보편화됨에 따라, 고 대역폭, 낮은 레이턴시, 스케일에 대한 손실 복원력(lossy resilience for scale), 분산 제어(distributed control) 및 가능한 한 적은 소프트웨어 오버헤드(software overhead)를 갖춘 통신 네트워크 패브릭들이 요구된다. 따라서, 기존(existing) 이더넷 기반 네트워크들보다 더 낮은 레이턴시를 달성하는 동시에, 중앙 처리 장치(central processing unit; CPU)의 개입(involvement)이 거의 없거나 전혀 없이 손실성의(lossy) 이더넷 기반 네트워크들을 통해 동작 가능한 네트워크 흐름 제어 프로토콜들을 개발하는 것이 바람직할 수 있다.

발명의 내용

과제의 해결 수단

[0005] 본 개시의 시스템들, 방법들 및 장치들은 각각 여러 가지 혁신적인 실시예들을 가지고, 그 중 어느 하나가 본 명세서에 개시된 바람직한 속성들에 대해 단독으로 책임지는 것은 아니다. 본 명세서에 설명된 주제의 하나 이상의 구현에 대한 상세한 설명들은 첨부된 도면들 및 아래의 설명에 제시되어 있다.

[0006] 일부 측면들에서, 본 명세서에서 설명된 기술들은 이더넷 기반 통신(Ethernet based communication)을 위한 제1 노드(node)와 관련되고, 제1 노드는: 전송 계층 하드웨어 전용 이더넷 프로토콜(transport layer hardware only Ethernet protocol)을 구현하는 하나 이상의 프로세서를 포함한다.

[0007] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 이더넷 프로토콜(Ethernet protocol)은 손실성(lossy)이다.

[0008] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하나 이상의 프로세서는, 제1 링크를 통해 제2 노드로 전송된 패킷(packet)들을 리플레이(replay)하기 위한 하드웨어 리플레이 아키텍처(hardware replay architecture)를 구현하도록 더 구성되고, 패킷들은, 제1 노드의 로컬 스토리지(local storage)에 저장되고, 리플레이하기 위한 패킷들의 순서(order)는, 링크된-리스트(linked-list)에서 지정된다.

[0009] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제1 노드는, 단일 자릿수 마이크로초 레이턴시(single digit microsecond latency)로 패킷을 제2 노드에 전송한다.

[0010] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하나 이상의 프로세서는 상태 머신(state machine)을 구현하도록 더 구성되고, 상태 머신은: 제1 노드와 제2 노드 간에 링크가 열려 있는 열린 상태(open state)에서 동작하고; 열린 상태에서 중간 닫힌 상태(intermediate close state)로 전환하고; 및 제2 노드로부터 닫힌 확인응답(close acknowledgement)을 수신한 것에 응답하여 링크를 닫기 위해 중간 닫힌 상태에서 닫힌 상태(close state)로 전환한다.

[0011] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 이더넷 포트(Ethernet port)를 더 포함한다.

[0012] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하나 이상의 프로세서는, 선입선출(first-in-first-out; FIFO) 메모리에 저장된 링크와 연관된 타이밍(timing) 및 상태 정보(status information)에 기초하여 제1 노드와 제2 노드 간의 링크에서 패킷을 리플레이하도록 결정하고, FIFO 메모리의 엔트리(entry)들은, 복수의 링크들과 연관된 하드웨어 링크 타이머(hardware link timer)의 틱(tick)들에 따라 액세스된다.

[0013] 일부 측면들에서, 본 명세서에서 설명된 기술들은 이더넷 기반 통신을 위한 제1 노드와 관련되고, 제1 노드는: 계층 2 하드웨어 전용 이더넷 프로토콜(layer 2 hardware only Ethernet protocol)을 구현하는 하나 이상의 프로세서를 포함한다.

[0014] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하나 이상의 프로세서는, 제1 링크를 통해 제2 노드로 전송된 패킷들을 리플레이하는 하드웨어 전용 아키텍처(hardware only architecture)를 포함한다.

[0015] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하나 이상의 프로세서는, 다수의 링크(multiple link)들과 연관된 타이머(timer)의 틱들에 기초하여 액세스되는 선입선출(FIFO) 메모리에 저장된 링크와 연관된 타이밍 및 상태 정보에 기초하여 제1 노드와 연관된 링크를 통해 패킷을 리플레이하도록 결정한다.

[0016] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제1 노드는, 이더넷 기반 네트워크에서 제2 노드와의 링크를 열고 닫도록 구성되고, 제1 노드는: 상태 머신 하드웨어를 포함하고; 상태 머신 하드웨어는: 제1 노드와 제2 노드 간에 링크가 열려 있는 열린 상태에서 동작하고; 열린 상태에서 중간 닫힌 상태로 전환하고; 제2 노드로부터 닫힌 확인응답을 수신한 것에 응답하여 링크를 닫기 위해 중간 닫힌 상태에서 닫힌 상태로 전환하고, 제1 노드는, 손실성의 네트워크(lossy network)에서 동작한다.

[0017] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 상태 머신 하드웨어는, 하드웨어에서만 전송 계층에 대한 흐름 제어 프로토콜을 구현한다.

[0018] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 흐름 제어 프로토콜과 연관된 레이턴시는, 10마이크로초 미만이다.

[0019] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 상태 머신 하드웨어는: 닫힌 상태에서 중간 열린 상태로 전환하고; 및 중간 열린 상태에서 열린 상태로 전환한다.

[0020] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 상태 머신 하드웨어는, 제2 노드에 링크를 닫으라는 요청(request)을 전송하거나 제2 노드로부터 링크를 닫으라는 요청을 수신한 것에 응답하여 열린 상태에서 중간 닫힌 상태로 전환한다.

[0021] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 상태 머신 하드웨어는, 제2 노드에 링

크를 닫으라는 확인응답(acknowledgement)을 전송한 것에 응답하여 중간 닫힌 상태에서 닫힌 상태로 전환한다.

[0022] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 상태 머신 하드웨어는, 시간의 주기 동안 대기하는 것 없이 중간 닫힌 상태에서 닫힌 상태로 전환한다.

[0023] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 열린 상태에서, 제1 노드는, 제2 노드로부터 패킷의 비-확인응답(non-acknowledgement)을 수신할 때까지 또는 패킷의 비-확인응답을 수신하는 것 없이 미리 결정된 타임아웃 기간(predetermined timeout period)이 만료될 때까지 패킷을 재전송하지 않는다.

[0024] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 열린 상태에서, 제1 노드는, 최대 N개의 패킷들을 중단(pause) 없이 전송하고, N은 제1 노드에 할당된 물리적 메모리(physical memory)의 크기에 의해 제한된다.

[0025] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제1 노드는 다수의 링크들과 연관된 하드웨어 링크 타이머; 및 하드웨어에서만 패킷들을 리플레이하는 하드웨어 리플레이 아키텍처를 더 포함한다.

[0026] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제1 노드는: 이더넷 프로토콜을 이용하여 제1 링크를 통해 제2 노드로 전송되는 패킷들을 리플레이하는 하드웨어 리플레이 아키텍처를 포함하고, 하드웨어 리플레이 아키텍처는: 패킷들을 포함하는 링크된-리스트를 저장하는 로컬 스토리지 - 링크된-리스트는 제2 노드로 전송하기 위한 패킷들의 순서를 유지함 -; 및 논리 회로(logic circuitry)를 포함하고, 논리 회로는: (a) 제2 노드로부터 제1 패킷의 비-확인응답(non-acknowledgement)의 수신 또는 (b) 제1 패킷과 연관된 타임아웃(timeout) 중 적어도 하나에 응답하여 패킷들의 제1 패킷을 리플레이하도록 결정하고; 및 제2 노드로부터의 제2 패킷의 확인응답의 수신에 응답하여 패킷들의 제2 패킷을 폐기(retire)하고, 이더넷 프로토콜은 손실성이다.

[0027] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 논리 회로는, 복수의 파이프라인 스테이지(pipelined stage)들을 포함하고, 논리 회로는, 복수의 파이프라인 스테이지들의 제1 파이프라인 스테이지에서 제1 노드와 제2 노드 간의 제2 링크가 아닌 제1 링크와 연관된 데이터를 처리하도록 결정한다.

[0028] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 논리 회로는, 복수의 파이프라인 스테이지들의 제2 파이프라인 스테이지에서 제1 패킷을 리플레이하도록 결정한다.

[0029] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 논리 회로는, 복수의 파이프라인 스테이지들의 제2 파이프라인 스테이지에서, 링크된-리스트에 의해 유지되는 패킷들의 순서에 기초하여 패킷들의 제3 패킷 및 패킷들의 제1 패킷을 리플레이하도록 결정한다.

[0030] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 논리 회로는, 링크 포인터(link pointer)에 기초하여 제2 링크가 아닌 제1 링크와 연관된 데이터를 처리하도록 결정하고, 논리 회로는, 복수의 파이프라인 스테이지들의 제3 파이프라인 스테이지에서 제2 링크를 가리키도록 링크 포인터를 업데이트한다.

[0031] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제1 노드 및 제2 노드는, 이더넷 기반 네트워크에 있고, 제1 노드는, 이더넷 스위치(Ethernet switch)를 통해 제2 노드와 통신한다.

[0032] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제1 노드는, 네트워크 인터페이스 프로세서(network interface processor; NIP) 및 고-대역폭 메모리(high-bandwidth memory; HBM)를 포함하고, HBM의 대역폭은, 적어도 1기가바이트(gigabyte)이다.

[0033] 일부 측면들에서, 본 명세서에서 설명된 기술들은 이더넷 기반 통신을 위한 제1 노드와 관련되고, 제1 노드는, 전송 계층 하드웨어 전용 이더넷 프로토콜을 구현하는 하나 이상의 프로세서를 포함하고, 전송 계층 하드웨어 전용 이더넷 프로토콜은 손실성이고, 하나 이상의 프로세서는, 전송 계층 하드웨어 전용 이더넷 프로토콜에 따라 전송된 패킷을 리플레이하는 하드웨어 리플레이 아키텍처를 포함한다.

[0034] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 리플레이 아키텍처는, 전송 계층 하드웨어 전용 이더넷 프로토콜에 따라 전송된 패킷들을 저장하는 로컬 스토리지를 포함한다.

[0035] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 리플레이 아키텍처는, 로컬 스토리지에 저장되고 다른 노드로 전송하기 위한 패킷들의 순서를 추적하도록 구성된 링크된-리스트를 포함하고, 링크된-리스트의 각각의 요소(element)는, 로컬 스토리지에 저장된 패킷들 각각에 대응한다.

[0036] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 리플레이 아키텍처는, 링크된-리스트에 대응하는 순서로 패킷들을 전송한다.

[0037] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 리플레이 아키텍처는: 링크된-리스트의 제1 요소를 가리키는 제1 포인터 - 제1 포인터는 링크된-리스트의 제1 요소에 대응하는 패킷들의 제1 패킷을 리플레이하지 않음을 나타냄 -; 및 링크된-리스트의 제2 요소를 가리키는 제2 포인터 - 제2 포인터는 링크된-리스트의 제2 요소에 대응하는 패킷들의 제2 패킷을 리플레이함을 나타냄 -를 저장한다.

[0038] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 리플레이 아키텍처는, 전송하기 위한 패킷들의 순서에 따라 제2 패킷 및 제2 패킷 다음의 하나 이상의 패킷을 리플레이한다.

[0039] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 리플레이 아키텍처는, 로컬 스토리지로 하여금 전송하기 위한 패킷들의 순서에 따라 제1 패킷 및 제2 패킷에 앞선 하나 이상의 패킷을 삭제(discard)하도록 한다.

[0040] 일부 측면들에서, 본 명세서에서 설명된 기술들은 이더넷 프로토콜을 이용하여 제1 링크를 통해 제2 노드로 전송된 패킷들을 리플레이하기 위한 제1 노드에서 구현되는 컴퓨터-구현 방법(computer-implemented method)과 관련되고, 컴퓨터-구현 방법은: 패킷들을 포함하는 링크된-리스트를 저장하는 단계 - 링크된-리스트는 제2 노드로 전송하기 위한 패킷들의 순서를 유지함 -; (a) 제2 노드로부터의 제1 패킷의 비-확인응답의 수신 또는 (b) 제1 패킷과 연관된 타임아웃 중 적어도 하나에 응답하여 패킷들의 제1 패킷을 리플레이하도록 결정하는 단계; 및 제2 노드로부터의 제2 패킷의 확인응답의 수신에 응답하여 패킷들의 제2 패킷을 폐기하는 단계를 포함하고, 이더넷 프로토콜은 손실성이다.

[0041] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 제1 노드는, 복수의 파이프라인 스테이지들을 포함하는 하드웨어 리플레이 아키텍처를 포함하고, 및 하드웨어 리플레이 아키텍처는, 복수의 파이프라인 스테이지들의 제1 파이프라인 스테이지에서 제2 링크가 아닌 제1 링크와 연관된 데이터를 처리하도록 결정한다.

[0042] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 하드웨어 리플레이 아키텍처는, 복수의 파이프라인 스테이지들의 제2 파이프라인 스테이지에서 제1 패킷을 리플레이하도록 결정한다.

[0043] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 하드웨어 리플레이 아키텍처는, 복수의 파이프라인 스테이지들의 제2 파이프라인 스테이지에서, 링크된-리스트에 의해 유지되는 패킷들의 순서에 기초하여 패킷들의 제3 패킷 및 패킷들의 제1 패킷을 리플레이하도록 결정한다.

[0044] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 제1 노드 및 제2 노드는, 이더넷 기반 네트워크에 있고, 제1 노드는, 이더넷 스위치를 통해 제2 노드와 통신한다.

[0045] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 제1 노드는, 네트워크 인터페이스 프로세서(NIP) 및 고-대역폭 메모리(HBM)를 포함하고, HBM의 대역폭은, 적어도 1기가바이트이다.

[0046] 일부 측면들에서, 본 명세서에서 설명된 기술들은 이더넷 기반 네트워크에서 패킷들을 전송하는 제1 노드와 관련되고, 제1 노드는: 하나 이상의 프로세서를 포함하고, 하나 이상의 프로세서는, 복수의 링크들과 연관된 타이밍 및 상태 정보를 저장하는 선입선출(FIFO) 메모리 - 제1 노드는 이더넷 프로토콜을 이용하여 복수의 링크들을 통해 하나 이상의 다른 노드에 패킷들을 전송함 -; 시간 주기(time period)에 따라 틱하는 타이머 - 타이머는 복수의 링크들과 연관됨 -; 및 논리 회로를 포함하고, 논리 회로는: 타이머의 각각의 틱들에 기초하여 FIFO 메모리의 엔트리들에 액세스하고; 및 복수의 링크들의 제1 링크와 연관된 타이밍 및 상태 정보에 기초하여, 제1 링크와 연관된 적어도 하나의 패킷을 리플레이하도록 결정하고, 이더넷 프로토콜은 손실성이다.

[0047] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 논리 회로는, 라운드-로빈 방식(round-robin manner)으로 FIFO 메모리의 엔트리들에 액세스한다.

[0048] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 타이머는, FIFO 메모리의 엔트리들과 연관된 활성 링크(active link)들의 개수에 기초하여 시간 주기를 조정하고, 활성 링크들은, 복수의 링크들에 포함된다.

[0049] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 논리 회로는, 복수의 링크들의 제2 링

크와 연관된 타이밍 및 상태 정보에 기초하여, 제2 링크와 연관된 패킷들을 폐기하도록 결정한다.

[0050] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제2 링크와 연관된 패킷들은, 제1 노드의 로컬 스토리지에 저장되고, 논리 회로는, 로컬 스토리지로 하여금 제2 링크와 연관된 패킷을 폐기하기로 결정한 것에 응답하여 제2 링크와 연관된 패킷들을 삭제하도록 한다.

[0051] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 논리 회로는, 복수의 링크들의 제2 링크와 연관된 타이밍 및 상태 정보에 기초하여, 제2 링크를 닫도록 결정한다.

[0052] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 복수의 링크들의 제1 링크와 연관된 타이밍 및 상태 정보는, 패킷들을 리플레이하기 위한 임계 기간(threshold duration) 동안 제1 노드에 의해 제1 링크와 연관된 적어도 하나의 패킷을 수신했다는 확인응답이 수신되지 않았음을 나타낸다.

[0053] 일부 측면들에서, 본 명세서에서 설명된 기술들은 이더넷 기반 통신을 위한 제1 노드와 관련되고, 제1 노드는: 전송 계층 하드웨어 전용 이더넷 프로토콜을 구현하는 하나 이상의 프로세서를 포함하고, 전송 계층 하드웨어 전용 이더넷 프로토콜은 손실성이고, 하나 이상의 프로세서는, 전송 계층 하드웨어 전용 이더넷 프로토콜에 따라 전송된 패킷들을 리플레이하도록 결정하는 하드웨어 링크 타이머를 포함한다.

[0054] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제1 노드는, 전송 계층 하드웨어 전용 이더넷 프로토콜에 따라 제1 링크를 통해 제1 복수의 패킷들을 전송하고 제2 링크를 통해 제2 복수의 패킷들을 전송하고, 하드웨어 링크 타이머는: FIFO 메모리의 제1 엔트리에 제1 링크와 연관된 타이밍 및 상태 정보를 저장하고, FIFO 메모리의 제2 엔트리에 제2 링크와 연관된 타이밍 및 상태 정보를 저장하는 선입선출(FIFO) 메모리를 포함한다.

[0055] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 링크 타이머는, 시간 주기에 따라 틱하는 다수의 링크들과 연관된 타이머를 포함하고, 하드웨어 링크 타이머는, 타이머의 라운드-로빈 방식의 틱들로 FIFO 메모리의 엔트리들에 액세스하고, 엔트리들은, 제1 엔트리 및 제2 엔트리를 포함한다.

[0056] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 링크 타이머는, FIFO 메모리의 엔트리들과 연관된 활성 링크들의 개수에 기초하여 시간 주기를 조정하고, 활성 링크들은, 제1 링크 및 제2 링크를 포함한다.

[0057] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 하드웨어 링크 타이머는: FIFO 메모리의 제1 엔트리에 저장된 제1 링크와 연관된 타이밍 및 상태 정보에 기초하여, 제1 복수의 패킷들 중 적어도 일부를 리플레이하도록 결정하고; 및 FIFO 메모리의 제2 엔트리에 저장된 제2 링크와 연관된 타이밍 및 상태 정보에 기초하여, 제2 복수의 패킷들을 폐기하도록 결정한다.

[0058] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제2 복수의 패킷들은, 제1 노드의 로컬 스토리지에 저장되고, 하드웨어 링크 타이머는, 로컬 스토리지로 하여금 제2 복수의 패킷들을 폐기하기로 결정한 것에 응답하여 제2 복수의 패킷들을 삭제하도록 한다.

[0059] 일부 측면들에서, 본 명세서에서 설명된 기술들은 제1 노드와 관련되고, 제1 링크와 연관된 타이밍 및 상태 정보는, 패킷들을 리플레이하기 위한 임계 기간 동안 제1 노드에 의해 제1 복수의 패킷들 중 하나를 수신했다는 확인응답이 수신되지 않았음을 나타낸다.

[0060] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 컴퓨터-구현 방법은: 제1 노드의 선입선출(FIFO) 메모리에 복수의 링크들과 연관된 타이밍 및 상태 정보를 저장하는 단계 - 제1 노드는 이더넷 프로토콜을 이용하여 복수의 링크들을 통해 하나 이상의 다른 노드에 패킷들을 전송함 -; 하드웨어 타이머의 각각의 틱들에 기초하여 FIFO 메모리의 엔트리들에 액세스하는 단계; 및 복수의 링크들의 제1 링크와 연관된 타이밍 및 상태 정보에 기초하여 제1 링크와 연관된 적어도 하나의 패킷을 리플레이하도록 결정하는 단계를 포함하고, 이더넷 프로토콜은 손실성이다.

[0061] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, FIFO 메모리의 엔트리들은, 라운드-로빈 방식으로 액세스된다.

[0062] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, FIFO 메모리의 엔트리들과 연관된 활성 링크들의 개수에 기초하여 하드웨어 타이머의 시간 주기를 조정하는 단계를 더 포함하고, 활성 링크들은, 복수의 링크들에 포함된다.

[0063] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 복수의 링크들의 제2 링크와 연관된 타이밍 및 상태 정보에 기초하여, 제2 링크와 연관된 패킷들을 폐기하도록 결정하는 단계를 더 포함한다.

[0064] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 제1 링크와 연관된 적어도 하나의 패킷이 리플레이되도록 하는 단계를 더 포함한다.

[0065] 일부 측면들에서, 본 명세서에서 설명된 기술들은 컴퓨터-구현 방법과 관련되고, 복수의 링크들의 제1 링크와 연관된 타이밍 및 상태 정보는, 패킷들을 리플레이하기 위한 임계 기간 동안 제1 노드에 의해 제1 링크와 연관된 적어도 하나의 패킷을 수신했다는 확인응답이 수신되지 않았음을 나타낸다.

[0066] 일부 측면들에서, 본 명세서에 설명된 기술들은 위에서 설명되고 논의된 모든 실시예들과 관련된다.

도면의 간단한 설명

[0067] 도면들 전체에서 참조 번호들은 참조 요소들 간의 대응 관계를 나타내기 위해 재이용된다. 도면은 본 명세서에 설명된 주제의 예들을 설명하기 위해 제공된 것이지, 그 범위를 제한하기 위해 제공된 것이 아니다.

본 개시의 실시예들은 첨부 도면들을 참조하여 설명되고, 도면에서 유사한 참조 문자들은 유사한 요소들을 참조한다.

도 1a-도 1b는 개방형 시스템 상호 연결(Open System Interconnection; OSI) 모델의 다양한 계층들에서 동작하는 예시적인 프로토콜들을 도시하는 표들이다.

도 2는 본 개시의 실시예들에 따른 테슬라 전송 프로토콜(Tesla Transport Protocol; TTP)을 구현하는 노드들 간의 링크들을 열고 닫기 위한 예시적인 상태 머신을 도시한다.

도 3a-도 3b는 본 개시의 실시예들에 따른 TTP를 구현하는 두 장치들 간의 패킷들의 전송 및 수신을 도시한 예시적인 타이밍 다이어그램들이다.

도 4는 본 개시의 실시예들에 따른 TTP를 구현하는 노드의 예시적인 개략적인 블록도를 도시한다.

도 5는 본 개시의 실시예들에 따른 TTP에 따라 전송되거나 또는 수신되는 패킷들에 대한 예시적인 헤더를 도시한다.

도 6은 본 개시의 실시예들이 구현될 수 있는 예시적인 네트워크 및 컴퓨팅 환경을 도시한다.

도 7a-도 7b는 본 개시의 일부 실시예들에 따른 다양한 타입들의 TTP 패킷들의 연산 코드(opcode)를 도시한다.

도 8은 본 개시의 일부 실시예들에 따른, TTP와 같은 손실성의 프로토콜(lossy protocol)에 따른 전송된 및/또는 수신된 패킷들을 리플레이하기 위한 패킷들을 저장하기 위한 예시적인 물리적 스토리지(physical storage)를 도시한다.

도 9는 본 개시의 일부 실시예들에 따른 패킷들의 전송 및 리플레이를 위한 전송의 순서를 추적하고 유지하기 위한 예시적인 데이터 구조(예를 들어, 링크된 리스트)를 도시한다.

도 10은 본 개시의 일부 실시예들에 따른 다수의 링크들을 통해 전송된 패킷들을 리플레이하기 위한 하드웨어 리플레이 아키텍처의 적어도 일부의 예시적인 블록도를 도시한다.

도 11은 본 개시의 일부 실시예들에 따른 소프트웨어의 도움 없이 패킷들을 리플레이하기 위한 타임아웃 체크 메커니즘(timeout checks mechanism)들을 구현하는 하드웨어 링크 타이머의 예시적인 블록도를 도시한다.

도 12는 본 개시의 일부 실시예들에 따른 노드로부터 전송되는 패킷들을 리플레이하기 위한 예시적인 루틴을 도시한다.

도 13은 노드와 연관된 하나 이상의 링크를 리플레이할지를 결정하기 위한 예시적인 루틴을 도시한다.

발명을 실시하기 위한 구체적인 내용

[0068] 특정한 실시예들에 대한 다음의 상세한 설명들은 특정한 실시예들의 다양한 설명들을 제시한다. 그러나, 본 명세서에 설명된 혁신들은 예를 들어 청구범위에 정의되고 적용되는 바와 같이 다양한 방식들로 구현될 수 있다. 본 설명에서, 도면들에 대한 참조가 이루어지고, 도면들에서 유사한 참조 번호들 및/또는 용어들은 동일하거나

기능적으로 유사한 요소들을 나타낼 수 있다. 도면들에 예시된 요소들은 반드시 일정한 축척으로 그려지는 것은 아니라는 것이 이해될 것이다. 또한, 특정한 실시예들은 도면에 예시된 것보다 더 많은 요소들 및/또는 도면에 예시된 요소들의 서브세트를 포함할 수 있다는 것이 이해될 것이다. 또한, 일부 실시예들은 둘 이상의 도면들로부터의 특징들의 임의의 적합한 조합을 포함할 수 있다. 제목들은 편의를 위해 제공된 것일 뿐, 특허청구범위의 범위 또는 의미에 영향을 미치지 않는다.

[0069] 일반적으로 설명하면, 본 개시의 하나 이상의 측면은 네트워크 트래픽 흐름(network traffic flow)을 제어하기 위해 하드웨어 메커니즘들(예를 들어, 소프트웨어의 도움 없이)을 이용하는 시스템들 및 방법들에 대응한다. 보다 구체적으로, 본 개시의 일부 실시예들은, 단일 자릿수 마이크로초 내의 레이턴시와 같은 낮은 레이턴시를 달성하기 위해, 하드웨어 회로를 통해 구현 가능하고 이더넷 표준들과 호환이 가능한 흐름 제어 프로토콜을 개시한다. 일부 실시예들에서, 단일 자릿수의 마이크로초 레이턴시는 네트워크 노드 간 통신 링크의 개폐를 간소화하기 위해 하드웨어 제어 상태 머신을 이용하여 최소한 부분적으로 달성된다. 또한, 개시된 흐름 제어 프로토콜(예를 들어, 테슬라 전송 프로토콜(TTP))은 확립된(established) 링크를 통해 전송/재전송되는 패킷들의 수 및/또는 하드웨어-제어 상태 머신(hardware-controlled state machine)의 다음 상태(next state)로 전환하기 전의 대기 기간(waiting period)들을 제한할 수 있다. 이는 통신의 낮은 레이턴시를 달성하는 것에 기여한다. 유리하게는, 본 명세서에 개시된 흐름 제어 프로토콜은 OSI(Open System Interconnection) 모델의 최대 계층 4(전송 계층)의 순수한(pure) 하드웨어 구현을 가능하게 한다.

[0070] 본 개시의 일부 측면들은 하드웨어에서만 실행되도록 설계된 흐름 제어와 관련된다. 이러한 흐름 제어는 소프트웨어 흐름 제어들 또는 중앙 처리 장치(CPU)/커널(kernel)의 개입 없이 구현될 수 있다. 이는 레이턴시를 물리적으로 제한하거나 우선 순위를 정하는 IEEE 802.3 이더넷 기능을 허용한다. 예를 들어, 단일 자릿수 마이크로초 레이턴시가 달성될 수 있다.

[0071] 이더넷을 통한 테슬라 전송 프로토콜(TTP)은 OSI 모델의 전송 계층까지 구현할 수 있는 하드웨어 전용 이더넷 흐름 제어 프로토콜이다. 계층 2(L2) 이더넷 흐름 제어는 하드웨어로만 구현될 수 있다. 계층 3 및/또는 계층 4 이더넷 흐름 제어도 하드웨어로만 구현될 수 있다. 링크 제어, 타이머들, 혼잡(congestion) 및 리플레이 기능은 하드웨어로 구현될 수 있다. TTP는 네트워크 인터페이스 프로세서들 및 네트워크 인터페이스 카드(Network Interface Card)들에 구현될 수 있다. TTP는 전체 I/O 배칭 구성(full I/O batching configuration)을 가능하게 할 수 있다. TTP는 손실성의 프로토콜이다. 손실성의 프로토콜에서, 손실된 데이터가 복구될 수 있다. 예를 들어, 손실성의 프로토콜에서 임의의 손실되거나 또는 손상된 패킷들이 리플레이(예를 들어, 재전송)될 수 있고 수신이 확인될 때까지 복구될 수 있다.

[0072] 본 개시에서 L2 헤더, 상태 머신 및 연산 코드들은 링크들의 N대N 세트(N-to-N set)에서 손실된 패킷들로부터 복구할 수 있는 하드웨어 전용 프로토콜(예를 들어, TTP)을 정의할 수 있다.

[0073] 또한, 본 개시의 일부 실시예들은 TTP와 같은 손실성의 프로토콜에 따라 전송된 및/또는 수신된 패킷들을 리플레이할 수 있는 하드웨어 리플레이 아키텍처(예를 들어, 마이크로 아키텍처)를 개시한다. 위에서 언급된 바와 같이, TTP(또는 TTPoE)는 하드웨어 전용 이더넷 흐름 제어 프로토콜이다. TTP는 HPC 및/또는 AI 훈련 시스템들을 위한 매우(extreme) 낮은 레이턴시(예를 들어, 단일 자릿수 마이크로초) 패브릭들의 구현을 용이하게 할 수 있다. 소프트웨어-제어 메커니즘(software-controlled mechanism)들의 도움 없이 손실성의 이더넷 흐름 제어 프로토콜을 구현하기 위해, 본 개시의 일부 측면들은 패킷들을 버퍼링하고, 홀드(hold)하고, 확인하고, 및/또는 리플레이할 수 있는 하드웨어 리플레이 아키텍처를 설명하고, 이를 통해 손실되거나 손상된 패킷들은 수신이 확인될 때까지 리플레이되고 복구될 수 있다.

[0074] 하드웨어 전용 리소스(hardware only resource)들을 가지는 TTP와 같은 손실성의 이더넷 프로토콜에 따라 전송된 및/또는 수신된 패킷을 리플레이하기 위해, 개시된 하드웨어 리플레이 아키텍처의 일부 실시예들은, 특히 리플레이가 발생할 때 전송된 패킷들의 순서를 유지하고 상이한 링크들에서 전송된 및/또는 수신된 패킷들을 저장하기 위해 데이터 구조 및 물리적 스토리지를 이용한다. 일부 실시예들에서, 물리적 스토리지는 하나 이상의 링크와 연관된 패킷들을 저장, 버퍼링 또는 홀드하는 임의의 타입의 로컬 스토리지 또는 캐시(cache)(예를 들어, 낮은-수준의 캐시들)일 수 있다. 물리적 스토리지는 크기가 메가바이트(megabyte; MB) 또는 킬로바이트(kilobyte; KB)의 단위와 같이 제한될 수 있다. 일부 실시예들에서, 데이터 구조는 하나 이상의 링크된 리스트를 포함할 수 있고, 각각의 링크된 리스트는 제1 통신 노드와 제2 통신 노드 간에 확립된 링크에 대해 전송된 패킷들의 순서를 기록 및/또는 추적할 수 있다. 유리하게는, 다양한 링크들에 대한 패킷 순서를 추적하는 링크된-리스트들 및 크기가 제한된 물리적 스토리지를 이용하는 하드웨어 리플레이 아키텍처를 이용하여 손실성의

프로토콜에 대한 리플레이 메커니즘을 구현하는 것은, 통신 노드가, 제한된 하드웨어 리소스들(예를 들어, 가상 처리 또는 스토리지 리소스들이 이용 가능하지 않은 경우)에 따라 TTP를 준수하여 동작하도록 한다.

[0075] 또한, 본 개시의 일부 실시예들은 소프트웨어-제어 메커니즘들의 도움 없이 타임아웃 체크들을 구현하는 하드웨어 링크 타이머와 관련된다. 각 링크별(per-link basis)로 타임아웃들을 추적하기 위해 다수의 타이머들을 이용하는 대신, 본 개시의 일부 측면들은 선입선출(FIFO) 메모리와의 협력(coordination)을 통해 다수의 링크들에 걸친 타임아웃들을 추적할 수 있는 단일 타이머를 이용하는 하드웨어 링크 타이머를 설명한다. 보다 구체적으로, FIFO 메모리의 엔트리는 링크의 상태 및/또는 타이머 정보를 저장할 수 있고, 하드웨어 링크 타이머는, 링크와 연관된 패킷들이 삭제될 수 있는지 또는 보존되어야 하는지를 결정하기 위해 라운드-로빈 방식으로 FIFO 메모리의 엔트리들에 액세스할 수 있다. 하드웨어 링크 타이머가 링크와 관련된 패킷들이 삭제될 수 있다고 결정하는 경우, 제한된 하드웨어 리소스들에 따라 또 다른 링크와 연관된 패킷들을 저장하기 위해 더 많은 공간이 이용 가능할 수 있다. 하드웨어 링크 타이머가 링크와 연관된 하나 이상의 패킷이 보존되어야 한다고 결정한 경우, 링크와 연관된 보존된 패킷(들)을 통해 하드웨어 링크 타이머를 호스팅하는 통신 노드가 보존된 패킷(들)을 리플레이하도록 할 수 있다.

[0076] 이더넷은 유선 통신을 위한 확립된 표준 기술이다. 최근 몇 년 동안, 이더넷은 자동차 산업에서도 다양한 차량용 애플리케이션들에 이용되고 있다. 일반적으로, 이더넷 통신과 연관된 레이턴시는 수백 마이크로초에서부터 수 밀리초 이상의 범위를 갖는다. 물리적 한계들(예를 들어, 통신 매체를 통한 신호 이동 속도) 외에도, 이더넷을 통한 데이터 흐름을 제어하는 연관된 프로토콜들의 복잡성(complexity)은 일반적으로 레이턴시에 또 다른 병목 현상(bottleneck)을 나타낸다. 예를 들어, 전송 제어 프로토콜(Transport Control Protocol; TCP) 또는 사용자 데이터그램 프로토콜(User Datagram Protocol; UDP)을 따르려면, 일반적으로 소프트웨어 제어 관리(software-controlled management)가 바람직할 수 있다. 소프트웨어 제어(software-controlled) 또는 소프트웨어-지원 네트워크 흐름 제어 관리(software-assisted network flow control management)는 통신과 연관된 레이턴시를 증가시키는 경향이 있다.

[0077] 그러나 레이턴시에 대한 이러한 제한들로 인해 이더넷 기술은 시스템 성능과 효율성을 개선하기 위해 1마이크로초 이내의 레이턴시가 바람직할 수 있는 고성능 컴퓨팅(HPC) 및 인공 지능(AI) 훈련 데이터 센터들과 같은 애플리케이션들에 덜 적합할 수 있다. 통합된 이더넷(Converged Ethernet)을 통한 원격 직접 메모리 액세스(Remote Direct Memory Access; RDMA)(RoCE) 또는 이더넷을 통한 인피니밴드(InfiniBand over Ethernet; IBoE)와 같은 프로토콜들이 레이턴시를 감소시키는 데 도움이 될 수 있지만, 시스템 설계 복잡성 또는 비용이 더 커질 수 있다. 예를 들어, RoCE 또는 InfiniBand는 구현하기 어려울 수 있는 무손실 네트워크(lossless network) 및 스케일링 사양들을 갖는다. RoCE 또는 InfiniBand를 구현하는 것은 상당한 소프트웨어 제어 오버헤드를 발생시키거나 대역폭 제한 중앙 집중식 토큰 제어 메커니즘(bandwidth-limited centralized token control mechanism)들을 수반한다. 또한, RoCE 또는 InfiniBand를 구현하는 시스템은 중단이 잦을(pause-heavy) 수 있다(예를 들어, 자주 중단(pause)됨).

[0078] 위의 문제들 중 적어도 일부를 해결하기 위해, 본 개시의 일부 실시예들은 이더넷 기반 네트워크들 또는 피어-투-피어(peer-to-peer; P2P) 네트워크들에서 동작 가능한 흐름 제어 프로토콜(예를 들어, 테슬라 전송 프로토콜(TTP))을 개시한다. 흐름 제어 프로토콜은, 통신의 레이턴시를 단일 자릿수 마이크로초 이내로 줄이기 위해 소프트웨어-제어 메커니즘들의 도움 없이 하드웨어를 통해 완전히(fully) 구현 가능하다. 흐름 제어 프로토콜은 범용 프로세서들 또는 컴퓨터-판독 가능한 인스트럭션(computer-readable instruction)들 또는 운영 체제들을 실행하는 중앙 처리 장치와 같은 소프트웨어 리소스들의 개입 없이도 구현될 수 있다. 또한, 흐름 제어 프로토콜에 내장된 일부 메커니즘들(예를 들어, 중단 전에 전송될 수 있는 패킷들의 수 제한, 동시에 확립될 수 있는 링크들의 수 제한, 하드웨어-제어 상태 머신 또는 TTP에 따라 전송된 또는 수신된 패킷들에 대한 제안된 헤더 포맷(proposed header format))을 이용하면, 흐름 제어 프로토콜을 구현하기 위해 가상화된 리소스(virtualized resource)들(예를 들어, 가상화된 프로세서들 또는 메모리)이 필요하지 않는다.

[0079] 일부 실시예들에서, 상태 머신은 노드들 간의 통신 링크를 열고 닫기 위해 상이한 상태들 사이의 전환들을 원활하게 한다. 상태 머신은 소프트웨어, 펌웨어, 드라이버 또는 다른 타입들의 프로그래밍 가능한 인스트럭션들의 개입 없이도 하드웨어에 의해 유지되고 구현될 수 있다. 따라서, 이더넷 기반 네트워크들에 적용 가능한 전송 제어 프로토콜(TCP)과 같은 소프트웨어 서포트(software support)를 이용하는 다른 프로토콜들의 구현들에 비해 상태 머신의 상이한 상태들 사이의 전환이 가속될 수 있다.

[0080] 일부 실시예들에서, TTP에 따라 전송된 및 수신된 패킷들에 대한 헤더(예를 들어, TTP 헤더)는 개방형 시스템간

상호 접속(Open System Interconnection; OSI) 모델의 계층 2에서부터 계층 4까지의 동작들을 서포트(support)한다. 헤더는 기존 이더넷 기반 네트워크 장치들 또는 인프라(infrastructure)에 의해 인식 가능한(recognizable) 필드(field)들을 포함할 수 있다. 따라서, TTP와 기존 이더넷 표준들의 호환성(compatibility)이 보존될 수 있다. 유리하게는, 이를 통해 기존 인프라 및/또는 공급망(supply chain)들을 경제적으로 이용할 수 있고, 시스템 설계 옵션들이 더 많아지고, 시스템-수준의 재이용(reuse) 또는 중복성(redundancy)을 달성할 수 있다.

[0081] 위에서 언급된 바와 같이, 노드는 소프트웨어-제어 메커니즘들의 도움 없이 하드웨어 전용 리소스들을 이용하여 TTP를 구현하거나 TTP에 따라 동작할 수 있다(예를 들어, TTP를 이용하여 또 다른 노드와 통신). 순수한 하드웨어 리소스를 이용하여 TPP에 따라 동작하기 위해, 노드는 하드웨어 리플레이 아키텍처를 이용하여 전송에서 손실될 수 있는 패킷들을 리플레이할 수 있다. 일부 실시예들에서, 하드웨어 리플레이 아키텍처는 하나 이상의 링크에서 전송된 및/또는 수신된 패킷들을 저장하기 위한 하나 이상의 캐시와 같은 로컬 스토리지를 포함할 수 있고, 하나 이상의 링크 각각은 TTP에 따라 열리거나 닫힐 수 있다. 거의 무제한의 처리 능력(processing power) 및 스토리지 용량(storage capacity)을 가지는 가상화된 리소스들이 일반적으로 소프트웨어-제어 네트워크 흐름 제어 관리를 통해 이용 가능한 TCP 또는 UDP와 같은 프로토콜들과 달리, TTP에 따라 동작하는 노드 내의 하드웨어 리플레이 아키텍처에 의해 이용되는 캐시(예를 들어, 낮은-수준의 캐시)의 크기는 제한될 수 있다. 예를 들어, 캐시의 크기는 256KB와 같이 메가바이트(MB) 또는 킬로바이트(KB)의 단위일 수 있다. 제한된 로컬 스토리지에 따른 TTP와 같은 손실성의 통신 프로토콜에 따라 확립된 하나 이상의 링크를 통해 서로 통신하기 위해, 하나 이상의 링크와 연관된 패킷들이 적절히 관리(예를 들어, 보존 또는 삭제)되어, 다른 패킷들이 캐시의 오버플로(overflow)를 방지하기 위해 삭제되는 동안 일부 패킷들은 리플레이를 위해 보존되도록 한다.

[0082] 일부 예들에서, TTP에 따라 확립된 링크를 이용하여 N개의 패킷들을 제2 노드로 전송하는 제1 노드는 N개의 패킷들을 저장하기 위해 캐시를 이용할 수 있고, N은 캐시의 크기에 의해 제한될 수 있는 임의의 양의 정수(positive integer)이다. TTP 및/또는 네트워크 컨디션(network condition)들로부터의 제약(constraint)들이 허락하는 한, 제1 노드는 N개의 패킷들 중 일부 또는 전부를 제2 노드로 계속해서 전송할 수 있다. 리플레이 패킷들을 수용(accommodate)하기 위해, 캐시는 제2 노드로부터 패킷을 수신한 것의 확인응답이 수신될 때까지 이미 전송된 패킷을 계속 저장할 수 있다. 패킷을 수신한 것의 확인응답이 수신되는 경우, 캐시는 제1 노드와 제2 노드 또는 다른 노드들 간의 링크 또는 다른 링크들을 통해 전송될 패킷들을 저장하기 위한 공간을 만들기 위해 패킷을 삭제할 수 있다. 대조적으로, 패킷의 비-확인응답이 수신된 경우(예를 들어, 제2 노드가 제1 노드에 패킷이 수신되지 않음을 알리는 경우) 또는 제2 노드로부터 패킷을 수신한 것의 확인응답 또는 비-확인응답을 수신하는 것 없이 타임아웃이 발생한 경우, 제1 노드는 패킷을 리플레이할 수 있다(예를 들어, 패킷을 제2 노드로 재전송). 패킷을 리플레이하는 것과 관련하여, 제1 노드는 수신의 확인응답을 수신한 다른 패킷들을 삭제할 수 있다.

[0083] 일부 예들에서, 패킷들을 전송하고 리플레이하는 순서가 동일할 수 있다. 예를 들어, 제1 노드는 특정한 순서로 N개의 패킷들을 전송할 수 있다(예를 들어, 1번째 패킷, 2번째 패킷 내지 N번째 패킷). 5번째 패킷이 리플레이되는 경우(예를 들어, 제1 노드가 제2노드로부터 5번째 패킷에 대한 비-확인응답을 수신한 것에 응답하여, 5번째 패킷을 수신한 것의 확인응답 또는 비-확인응답을 수신하는 것 없이 타임아웃이 발생한 것에 응답하여) 및 1번째 패킷~4번째 패킷에 대한 확인응답이 수신된 경우, 캐시는 1번째 패킷~4번째 패킷을 삭제할 수 있지만 5번째 패킷은 삭제하지 않으므로 노드는 5번째 패킷을 리플레이할 수 있다. 추가적으로 및/또는 선택적으로, 5번째 패킷을 리플레이하는 경우, 제1 노드는 5번째 패킷 이후에 전송된 패킷들을 이전에 전송된 것과 동일한 순서로 리플레이할 수 있다(N > 5라고 가정).

[0084] 일부 예들에서, N개의 패킷들 중 일부 또는 전부의 제1 전송과 그 이후의 임의의 리플레이 간의 순서를 유지하기 위해 제1 노드의 하드웨어 리플레이 아키텍처는 캐시와 협력하여 링크된-리스트를 이용할 수 있다. 링크된-리스트는 N개의 요소들을 포함할 수 있고, 각각의 요소는 N개의 패킷들 각각 및 다음 패킷에 대응하는 다음 요소에 대한 레퍼런스(reference)를 포함한다. N개의 패킷들을 전송 및/또는 리플레이하는 경우, 패킷이 리플레이를 위해 보관되어야 할지 또는 (예를 들어, 스토리지 리소스들을 보존(conserve)하기 위해) 삭제되어도 되는지를 결정하기 위해 하드웨어 리플레이 아키텍처는 링크된-리스트의 하나 이상의 요소를 가리키는 하나 이상의 포인터를 더 이용할 수 있다. N이 9인 경우(예를 들어, 제1 노드에서부터 제2 노드로 전송된 9개의 패킷)를 예로 들면, 링크된-리스트에서, 1번째 요소는 1번째 패킷 및 1번째 레퍼런스를 포함할 수 있고, 1번째 레퍼런스는 2번째 요소를 가리키고; 2번째 요소는 2번째 패킷 및 2번째 레퍼런스를 포함할 수 있고, 2번째 레퍼런스는 제3 요소를 가리키고; 및 8번째 요소는 8번째 패킷 및 8번째 레퍼런스를 포함할 수 있고, 8번째 레퍼런스는 9번째

요소를 가리키고; 및 9번째 요소는 9번째 패킷을 포함할 수 있다. 하드웨어 리플레이 아키텍처는 3개의 요소들을 가리키는 3개의 포인터들을 유지하고 업데이트할 수 있다. 노드가 1번째 패킷~9번째 패킷을 전송했고, 제2 노드로부터 1번째 패킷~7번째 패킷을 수신했지만 8번째 패킷~9번째 패킷은 수신하지 않은 확인응답을 수신하였다고 가정하면, 제1 포인터는 링크된-리스트의 1번째 요소를 가리킬 수 있고, 제2 포인터는 링크된-리스트의 8번째 요소를 가리킬 수 있고, 제3 포인터는 링크된-리스트의 9번째 요소를 가리킬 수 있다. 따라서, 하드웨어 리플레이 아키텍처는 캐시로 하여금 3개의 포인터들에 기초하여 패킷들을 리플레이하고 패킷들을 삭제하도록 할 수 있다. 보다 구체적으로, 캐시는 제2 포인터가 가리키는 패킷(예를 들어, 8번째 패킷)부터 제3 포인터가 가리키는 패킷(예를 들어, 9번째 패킷)을 리플레이할 수 있고, 나머지 패킷들(예를 들어, 제2 포인터가 가리키는 패킷 전의 제1 포인터가 가리키는 패킷)을 삭제할 수 있다. 추가적으로 선택적으로, 일부 또는 전체 하드웨어 리플레이 아키텍처는 노드의 처리량(throughput)을 증가시키기 위해 파이프라인 방식(pipelined manner)으로 동작할 수 있다. 유리하게는, 리플레이 기능을 구현하기 위해 캐시 및 링크된-리스트들을 이용하는 것은, 소프트웨어 제어 메커니즘들의 도움 없이 제한된 하드웨어 리소스들에 따른 TTP를 이용하여 제1 노드가 제2 노드와 통신할 수 있도록 한다.

[0085] 위에서 언급된 바와 같이, TTP 프로토콜에 따라 동작하는 노드는 소프트웨어의 도움 없이 패킷들을 리플레이하기 위한 타임아웃 체크 메커니즘들을 구현하는 하드웨어 링크 타이머를 포함할 수 있다. 다수의 타이머들(예를 들어, 하나의 링크에 대한 타이머)을 이용하여 다수의 링크들을 통한 타임아웃들을 추적하기 위해 일반적으로 소프트웨어가 이용되는 다른 이더넷 프로토콜들(예를 들어, TCP 또는 UDP)과 달리, 하드웨어 링크 타이머는, 노드가 어떤 패킷(들)을 어떤 링크(들)를 통해 전송되도록 하고 리플레이할지, 리플레이가 필요한 경우 제한된 하드웨어 리소스들(예를 들어, 가상 및/또는 물리적 주소 공간의 큰 리소스 풀(large resource pool)들 및 컴퓨팅 리소스들이 이용 가능하지 않은 경우)에 따라 언제 리플레이할지 결정하도록 할 수 있다. 일부 실시예들에서, 하드웨어 링크 타이머는 노드와 연관된 확립된 링크들(예를 들어, 활성 링크들)에 대한 타이밍 체크를(timing check) 주기적으로 수행할 수 있다. 하드웨어 링크 타이머는 활성 링크들 각각과 연관된 타이밍 및 상태 정보를 저장하고 라운드-로빈 방식으로 활성 링크들 각각과 연관된 타이밍 및 상태를 체크할 수 있는 선입선출(FIFO) 메모리를 포함할 수 있다. 하드웨어 링크 타이머는, 다수의 활성 링크들 및/또는 패킷들이 다수의 활성 링크들 및/또는 패킷들 각각과 연관된 타이밍 및 상태 정보를 판독할 수 있는 시점(point in time)들을 스케줄링 하기 위해 단일 프로그래밍 가능한 타이머(single programmable timer)를 이용할 수 있다. 판독 타이밍 및 상태 정보는, 링크와 연관된 패킷들을 리플레이할지, 또는 추가 정보 조회(further information look up)를 통해 패킷들을 삭제할지를 결정하기 위해 이용될 수 있다.

[0086] 일부 예들에서, FIFO 메모리는 제1 노드 및 다른 노드(들) 간에 확립된 하나 이상의 링크와 연관된 타이밍 정보를 저장할 수 있다. 예를 들어, 제1 노드는, 제1 노드 및 하나 이상의 다른 노드 간에 확립된 M개의 링크들과 연관된 타이밍 정보를 저장하기 위해 FIFO 메모리를 이용하는 하드웨어 링크 타이머를 포함할 수 있고, 여기서 M은 1보다 큰 양의 정수이다. 각각의 타이머가 대응하는 링크의 타이밍 정보를 추적하는 M개의 타이머들을 이용하는 대신, 하드웨어 링크 타이머는 라운드-로빈(예를 들어, 순환(circular)) 방식으로 FIFO 메모리에 액세스하는 것을 통해 M개의 링크들 각각에 대한 타이밍 정보를 추적 및/또는 업데이트하기 위해 단일 타이머(예를 들어, 프로그래밍 가능한 시간 주기 동안 한 번 틱하는 타이머)를 이용할 수 있다. 구체적으로, 하드웨어 링크 타이머는 단일 타이머가 한 번 틱할 때 FIFO 메모리의 엔트리들에 한 번에 하나씩 액세스할 수 있고, FIFO 메모리의 각각의 액세스된 엔트리들은 M개의 링크들 중 하나에 대응한다. 일부 실시예들에서, 각각의 틱의 시간 주기가 달라질 수 있고 수백 마이크로초에서 단일 자릿수 마이크로초 사이일 수 있다. 예를 들어, 틱의 시간 주기는 최대 100마이크로초일 수 있고, 최소 1마이크로초일 수도 있다. 또한, 하드웨어 링크 타이머는 FIFO 메모리의 엔트리들에 의해 나타난 링크들의 수(예를 들어, M)에 기초하여 틱의 시간 주기를 조정할 수 있다. 예를 들어, M이 증가하면(예를 들어, FIFO 메모리의 엔트리들에 의해 나타나는 링크들이 더 많아짐) 틱의 시간 주기는 감소할 수 있고, M이 감소하면(예를 들어, FIFO 메모리 엔트리들에 의해 나타나는 링크들이 더 적어짐) 틱의 시간 주기는 증가할 수 있다. 따라서, FIFO 메모리의 엔트리들에 의해 나타난 링크들의 개수에 대한 틱의 시간 주기가 비례하지 않게(disproportionally) 변경되는 경우, 링크의 상태 및/또는 타이밍 정보가 체크되는 시간 간격(time interval)은 변경되지 않을 수 있다.

[0087] 일부 예들에서, M개의 링크들 중 하나와 연관된 타이밍 및/또는 상태 정보는 링크가 얼마나 오래, 전송된 패킷들을 수신한 것의 확인응답을 수신하지 못했는지를 나타낼 수 있다. 제1 노드가 링크를 통해 제2 노드로 N개의 패킷들을 전송했다고 가정하면, FIFO 메모리의 하나의 엔트리는, 틱의 특정한 시간 주기에 따라 라운드-로빈 방식을 통해 액세스될 때, N개의 패킷들 중 임의의 것을 수신한 것의 확인응답이 미리 결정된 기간 동안 수신되지 않았다는 것을 나타내는 타이밍 및/또는 상태 정보를 저장할 수 있다. FIFO 메모리의 엔트리에 액세스하면, 하

드웨어 링크 타이머는, N개의 패킷들을 리플레이하기 위해 제1 노드의 로컬 스토리지(예를 들어, 낮은-수준의 캐시)에 저장되어 있을 수 있는 N개의 패킷들을 조회(look up)하기 위해, 엔트리에 저장된 타이밍 및/또는 상태 정보를 이용할 수 있다. 대안적으로, M개 링크들 중 하나와 연관된 타이밍 및/또는 상태 정보가, 링크가 닫힐 수 있음을 나타내기 위해 FIFO 메모리의 하나의 엔트리에 저장될 수 있다(예를 들어, 제1 노드에 의해 전송된 모든 패킷들이 제2 노드에 의해 수신됨). FIFO 메모리의 엔트리에 액세스하면, 하드웨어 링크 타이머는, FIFO 메모리의 엔트리에 저장된 타이밍 및/또는 상태 정보가, 링크가 닫힐 수 있음을 나타내기 때문에 패킷들을 삭제하고 제1 노드의 로컬 스토리지에 여전히 저장되어 있을 수 있는 패킷들을 조회하기 위해, 엔트리에 저장된 타이밍 및/또는 상태 정보를 이용할 수 있다. 유리하게는, 조정 가능한 기간(adjustable period)들에 따라 틱하는 다수의 링크들 및/또는 패킷들에 대한 단일 타이머 및 다수의 링크들의 타이밍 및/또는 상태 정보를 저장하는 FIFO 메모리를 이용하는 것에 의해, 제1 노드는, 제한된 컴퓨팅 및 스토리지 리소스들에 따라 동작하도록 활성 링크들에 의한 이용을 위해 비활성 링크(inactive link)들(예를 들어 닫힌 링크(closed link)들)에 의해 차지된 하드웨어 리소스들을 해제(release)하고 낮은 레이턴시를 달성하기 위해, 적절한 타이밍에 패킷들을 리플레이할 수 있다.

[0088] 예시적인 실시예들 및 특징들의 조합에 따른 다양한 측면들이 설명될 것이지만, 관련 기술 분야의 당업자는 이러한 예들 및 특징들의 조합이 본질적으로 예시적인 것이고, 제한적인 것으로 해석되어서는 안 된다는 것을 이해할 것이다. 더 구체적으로, 본 출원의 측면들은, 상이한 맥락들에 따라 다양한 타입들의 네트워크들 및 통신 프로토콜들에 적용될 수 있다. 또한, 네트워크 흐름들을 제어하기 위한 회로 블록도들 또는 상태 머신의 구체적인 아키텍처들이 설명되더라도, 이러한 예시적인 회로 블록도들 또는 상태 머신 또는 아키텍처는 제한적인 것으로 해석되어서는 안 된다. 따라서, 관련 기술 분야의 당업자는, 본 출원의 측면들이 반드시 임의의 특정한 타입들의 네트워크들, 네트워크 인프라 또는 네트워크들의 노드들 간의 예시적인 상호작용들에 대한 적용으로 제한되지 않는다는 것을 이해할 것이다.

[0089] **테슬라 전송 프로토콜(TTP)**

[0090] 도 1a-도 1b는 (7개의 계층들을 가지는) OSI 모델과 각각의 계층과 연관된 예시적인 프로토콜들을 도시하는 표들이다. 도 1a는 OSI 모델의 계층 4(예를 들어, 전송 계층)에서 동작하는 TCP 및 UDP 프로토콜들을 포함하는 예시적인 프로토콜들을 도시한다. 도 1b는 OSI 모델의 계층 4에서 동작하는 테슬라 전송 프로토콜(TTP)을 포함하는 예시적인 프로토콜들을 도시한다.

[0091] 도 1a에 도시된 바와 같이, 계층 4에서 동작하는 TCP 또는 UDP 외에도 TCP 또는 UDP와 함께 동작하는 다른 예시적인 프로토콜들 또는 애플리케이션들은: 계층 7에서 동작하는 하이퍼텍스트 전송 프로토콜(Hypertext Transfer Protocol; HTTP), 텔레타입 네트워크(Teletype Network; Telnet), 파일 전송 프로토콜(File Transfer Protocol; FTP); 계층 6에서 동작하는 공동 사진 전문가 그룹(Joint Photographic Experts Group; JPEG), 이동식 네트워크 그래픽(Portable Network Graphics; PNG), 동영상 전문가 그룹(Moving Picture Experts Groups; MPEG); 계층 5에서 동작하는 네트워크 파일 시스템(Network File System; NFS) 및 구조적 쿼리 언어(Structured Query Language; SQL); 계층 3에서 동작하는 인터넷 프로토콜 버전 4(Internet Protocol version 4; IPv4)/인터넷 프로토콜 버전 6(Internet Protocol version 6; IPv6); 등을 포함할 수 있다. 계층 4에서 동작하는 TCP 또는 UDP의 경우, 계층 4의 구현은 일반적으로 도 1a에 도시된 바와 같이 소프트웨어가 필요하다.

[0092] 도 1b에 도시된 바와 같이, 계층 4에서 동작하는 TTP 외에도, TTP와 함께 동작하는 다른 예시적인 프로토콜들 또는 애플리케이션들은: 계층 7에서 동작하는 파이토치(Pytorch); 계층 6에서 동작하는 FFMPEG, 고효율 비디오 코딩(High Efficiency Video Coding; HEVC), YUV; 계층 5에서 동작하는 RDMA; 계층 3에서 동작하는 IPv4/IPv6; 등을 포함할 수 있다. 도 1a와 대조적으로, 계층 4에서 동작하는 TTP의 경우, OSI 모델의 계층 1~계층 4의 구현들은 도 1b에 도시된 바와 같이 소프트웨어의 개입 없이 하드웨어로만 수행될 수 있다. 유리하게는, 도 1b에 도시된 바와 같이 TTP에 기초한 OSI 모델의 계층 1~계층 4를 통한 순수한 하드웨어 구현은 도 1a에 도시된 바와 같은 구현과 비교했을 때 이더넷 기반 네트워크들을 통한 통신의 레이턴시를 단축시킬 수 있다.

[0093] 도 2는 본 개시의 실시예들에 따른 TTP를 구현하는 노드들 간의 링크들을 열고 닫기 위한 예시적인 상태 머신 200을 도시한다. 상태 머신 200은 네트워크 인터페이스 프로세서 또는 네트워크 인터페이스 카드에 의해 구현될 수 있다. 이더넷 링크를 통해 통신하는 각각의 노드의 노드들 간에는 각각의 이더넷 링크에 대해 하나의 상태 머신 200이 있을 수 있다. 예를 들어, 네트워크 인터페이스 프로세서가 5개의 TTP 링크들을 통해 5개의 네트워크 인터페이스 카드들과 통신할 수 있는 경우, 네트워크 인터페이스 프로세서는 각각의 링크에 대해 하나의 인스턴스(instance)를 가지는 상태 머신 200의 5개의 인스턴스들을 포함할 수 있다. 이 예에서, 5개의 네트워크

인터페이스 카드들 각각은 네트워크 인터페이스 프로세서와 통신하기 위한 상태 머신 200의 하나의 인스턴스를 가질 수 있다. 일부 실시예들에서, 상태 머신 200을 이용하여 서로 통신하는 노드들은 피어-투-피어 네트워크를 형성할 수 있다.

[0094] 도 2에 도시된 바와 같이, 상태 머신 200은 닫힌 상태(closed state) 202, 열린 수신된 상태(open received state) 204, 열린 전송된 상태(open sent state) 206, 열린 상태 208, 닫힌 수신된 상태(close received state) 210 및 닫힌 전송된 상태(close sent state) 212를 포함한다. 상태 머신 200은, 상태 머신 200을 유지하는 제1 노드와 통신 링크가 확립되도록 하는 제2 노드 사이에 현재 열려 있는 통신 링크가 없음을 나타낼 수 있는 닫힌 상태 202에서 시작할 수 있다. 또한, 상태 머신 200의 개별 사본(individual copy)은 본 개시에서 개시되는 테슬라 전송 프로토콜(TTP)에 기초하여 동작하는 노드에 의해 유지, 업데이트 및 전환될 수 있다. 또한, TTP에 기초하여 동작하는 노드가 다수의 노드들과 동시에(concurrently) 또는 시간적으로 겹치게(overlapping in time) 통신하는 경우, 노드는 각각의 링크들에 대해 다수의 및 독립적인 상태 머신들 200을 보유(retain)할 수 있다.

[0095] 그러면 상태 머신 200은 제1 노드가 제2 노드에 통신 링크를 확립하기 위한 요청을 전송하는지, 또는 제2 노드로부터 통신 링크를 확립하기 위한 요청을 수신하는지에 따라 상이하게 전환할 수 있다. 제1 노드가 제2 노드에 통신 링크를 열라는 요청을 전송하는 경우, 상태 머신 200은 닫힌 상태 202에서 열린 전송된 상태 206로 전환할 수 있다. 반면, 제1 노드가 제2 노드로부터 통신 링크를 열라는 요청을 수신하는 경우, 상태 머신 200은 닫힌 상태 202에서 열린 수신된 상태 204로 전환할 수 있다.

[0096] 열린 전송된 상태 206에 있는 동안, 상태 머신 200은 열린 전송된 상태 206에 머물거나 다양한 기준에 따라 닫힌 상태 202로 다시 전환하거나 열린 상태 208로 전환할 수 있다. 제1 노드가 제2 노드로부터 open-nack(예를 들어, 링크를 열라는 요청을 거부하는 메시지)을 수신하는 경우, 상태 머신 200은 열린 전송된 상태 206에서 다시 닫힌 상태 202로 전환할 수 있다. 반면, 제1 노드가 제2 노드로부터 open-ack(링크를 열라는 요청을 수락하는 메시지)을 수신하는 경우, 상태 머신 200은 열린 전송된 상태 206에서 열린 상태 208로 전환할 수 있다. 대안적으로, 제1 노드가 특정한 시간 주기 내에 제2 노드로부터 open-nack 또는 open-ack를 수신하지 못한 경우, 제1 노드가 타임-아웃될 수 있고, 그러면 제1 노드는 제2 노드에 통신 링크를 열라는 요청을 재전송하고 열린 전송된 상태 206에 머무를 수 있다.

[0097] 앞서 언급된 바와 같이, 닫힌 상태 202에 있는 동안, 제1 노드가 제2 노드로부터 통신 링크를 열라는 요청을 수신하는 경우, 상태 머신 200은 닫힌 상태 202에서 열린 수신된 상태 204로 전환할 수 있다. 수신된 상태 204에서, 상태 머신 200은 제1 노드가 제2 노드로부터 링크를 열라는 요청을 수락하는지 또는 거부하는지에 따라 상이하게 전환할 수 있다. 예를 들어, 제1 노드는 제2 노드에 open-nack(예를 들어, 링크를 열라는 요청을 거부)을 전송하기로 선택할 수 있다. 이러한 상황에서, 상태 머신 200은 닫힌 상태 202로 다시 전환할 수 있고, 제1 노드는 제2 노드 또는 다른 노드들로부터 링크를 열라는 요청을 더 전송하거나 수신할 수 있다. 대안적으로, 열린 수신된 상태 204에서, 제1 노드는 제2 노드에 open-ack을 전송할 수 있고 그런 다음 열린 상태 208로 전환할 수 있다.

[0098] 열린 상태 208에서 제1 노드 및 제2 노드는 확립된 통신 링크를 통해 서로 패킷들을 전송 및 수신할 수 있다. 이 링크는 유선 이더넷 링크(wired Ethernet link)일 수 있다. 제1 노드는 일부 컨디션이 발생할 때까지 열린 상태 208에 머무를 수 있다. 일부 실시예들에서, 상태 머신 200은, 제1 노드 및 제2 노드가, 열린 상태 208에 있는 동안 패킷들을 전송 및 수신하도록 하는 통신 링크를 닫으라는 요청을 수신하는 것에 응답하여 열린 상태 208에서 닫힌 수신된 상태 210로 전환할 수 있다. 대안적으로, 상태 머신 200은, 제1 노드가 제2 노드에 통신 링크를 닫으라는 요청을 전송한 것에 응답하여 열린 상태 208에서 닫힌 전송된 상태 212로 전환할 수 있다. 통신 링크를 닫으라는 요청들 외에도, 통신 링크가 임계 시간 이상 유휴 상태(idle)인 경우, 상태 머신 200은 열린 상태 208에서 닫힌 수신된 상태 210 또는 닫힌 전송된 상태 212로 전환할 수 있다.

[0099] 닫힌 수신된 상태 210에 있는 동안, 제1 노드가 제2 노드에 close-ack(예를 들어, 링크를 닫으라는 요청을 확인하거나 수락하는 메시지)를 전송하는 경우, 상태 머신 200은 닫힌 상태 202로 다시 전환할 수 있다. 그렇지 않으면, 제1 노드가 제2 노드에 close-nack(예를 들어, 링크를 닫으라는 요청을 거부하거나 확인하지 않는 메시지)을 전송하는 경우, 상태 머신 200은 닫힌 수신된 상태 210에 머물 수 있다.

[0100] 닫힌 전송된 상태 212에 있는 동안, 제1 노드가 제2 노드로부터 close-ack(예를 들어, 링크를 닫으라는 요청을 확인하거나 수락하는 메시지)을 수신하는 경우, 상태 머신 200은 닫힌 상태 202로 다시 전환할 수 있다. 그렇지 않으면, 제1 노드가 제2 노드로부터 전송된 close-nack(예를 들어, 링크를 닫으라는 요청을 거부하거나 확인하

지 않는 메시지)을 수신하는 경우, 상태 머신 200은 닫힌 전송된 상태 212에 머무를 수 있다. 닫힌 전송된 상태 212에서, 제1 노드는 타임아웃 임계(timeout threshold) 내에 제2 노드로부터 응답을 받지 못하는 경우, 제2 노드에 통신 링크를 닫으라는 요청을 다시 전송할 수 있다.

[0101] 일부 실시예들에서, 상태 머신 200은 소프트웨어, 펌웨어, 드라이버 또는 다른 타입들의 프로그래밍 가능한 인스트럭션들의 개입 없이 하드웨어에 의해 유지되고 구현될 수 있다. 따라서, 이더넷 기반 네트워크들에 적용 가능한 전송 제어 프로토콜(TCP)과 같은 소프트웨어 서포트를 포함하는 다른 프로토콜들의 구현들에 비해 상태 머신 200의 상이한 상태들 사이의 전환이 가속될 수 있다.

[0102] 일부 실시예들에서, 전송 패킷(transmitting packet)들을 전송 대기 상태로 유지하고 전송 대기열(transmission queue)에 저장하는 대신, 제1 노드는 전송 대기열에서 패킷들을 전송하는 것을 즉시 중단하고 닫힌 수신된 상태 210에 있는 동안 제2 노드로부터 링크를 닫으라는 요청을 수신한 것에 응답하여 제2 노드에 close-ack을 전송한다. 유리하게는, 링크를 닫으라는 요청을 수신한 후 무한한 시간 동안 패킷들을 전송하는 것을 계속하지 않으면 제1 노드가 짧은 전환 주기(transition period)와 적은 시간적 불확실성(uncertainty in time)으로 열린 상태 208에서 닫힌 상태 202로 다시 전환하도록 할 수 있다.

[0103] 또한, 열린 상태 208 동안 제1 노드 또는 제2 노드에 의해 계속해서 전송될 수 있는 패킷들의 수는 제한될 수 있다. 예를 들어, 열린 상태 208에 있는 동안, 제1 노드는 패킷들을 전송하는 것을 중단하기 전에 N개의 패킷들만 연속적으로 전송할 수 있고 N은 1에서부터 천 이상까지의 양의 정수일 수 있다. 숫자 N은 물리적 메모리에 의해 제한될 수 있다. 일부 실시예들에서, N은 제1 노드에서 이용 가능한 물리적 메모리(예를 들어, 동적 랜덤 액세스 메모리(dynamic random access memory) 또는 이와 유사한 것)의 크기에 의해 제한되거나 제약을 받을 수 있다. 구체적으로, N은 제1 노드 또는 제2 노드와 연관된 물리적 메모리의 크기에 비례할 수 있다. 예를 들어, 제1 노드에 1기가바이트(GB)의 물리적 메모리가 할당된 경우, N은 최대 100만까지 될 수 있다. 일부 실시예들에서, N은 수만 또는 수십만 이내일 수 있다. 열린 상태 208 동안, 패킷을 교환하기 위한 물리적 메모리 양(amount)이 추적될 수 있다. 유리하게는, 제1 노드 또는 제2 노드에 의해 계속해서 전송될 수 있는 패킷들의 수를 제한하는 것은 상태 머신 200을 구현하기 위한 컴퓨팅 및 스토리지 리소스들을 감소시킬 수 있다. 일반적으로 가상화를 통해 무제한의 소프트웨어 및 하드웨어 리소스들(예를 들어, 가상화된 메모리 또는 처리 리소스들)을 이용 가능성(availability)을 가정하는 프로토콜들(예를 들어, TCP)과 달리, 전송되는 패킷들의 수를 제한하는 것은 TTP가 보다 제한된 계산 및 스토리지 리소스들에 따라 동작하도록 한다.

[0104] 일부 실시예들에서, 제1 노드 또는 제2 노드는 다른 노드에 close-ack을 수신하거나 전송한 후 링크를 닫기 위해 더 이상 대기하지 않는다. 예를 들어, 닫힌 전송된 상태 212에 있는 동안, 제1 노드는 제2 노드로부터 전송된 close-ack을 수신한 것에 응답하여 즉시 닫힌 상태 202로 전환할 수 있다. 제2 노드에 전송할 추가 패킷들이 있는지 모니터링하기 위해 미리 결정된 시간 또는 무작위 기간을 대기하는 대신, 제1 노드는 더 짧은 시간 안에 닫힌 전송된 상태 212에서 닫힌 상태 202로 다시 전환할 수 있다. 유리하게는, 이것은 상태 머신 200의 상태들 사이의 전환하는 것과 연관된 정확도를 증가시키고 레이턴시를 단축하므로, TTP는 TCP와 같은 프로토콜들보다 레이턴시가 낮은 통신을 용이하게 한다.

[0105] 도 3a-도 3b는 본 개시의 실시예들에 따른 TTP를 구현하는 두 장치들 간의 패킷들의 전송 및 수신을 도시하는 예시적인 타이밍 다이어그램들을 도시한다. 도 3a는 장치 A에서부터 장치 B로 전송된 패킷들이 전혀 손실되지 않는 시나리오를 도시하고, 도 3b는 장치 A에서부터 장치 B로 전송된 패킷들 중 일부가 손실되는 또 다른 시나리오를 도시한다. 도 3a-도 3b는 상태 머신 200과 함께 이해될 수 있다. 장치 A 및 장치 B는 TTP를 통해 통신하는 두 개의 예시적인 노드들이다.

[0106] 도 3a에 도시된 바와 같이, 장치 A는 닫힌 상태 202에 있는 동안 패킷 ID = 0인 TTP_OPEN을 장치 B로 전송할 수 있다. (1)에서 TTP_OPEN을 장치 B로 전송한 후, 장치 A에 의해 유지되는 상태 머신은 닫힌 상태 202에서 열린 전송된 상태 206로 전환할 수 있다. 또한, (1)에서 장치 A로부터 TTP_OPEN을 수신한 후, 장치 B에 의해 유지되는 상태 머신은 닫힌 상태 202에서 열린 수신된 상태 204로 전환할 수 있다.

[0107] 그런 다음, (2)에서 장치 B로부터 TTP_OPEN_ACK를 수신한 후, 장치 A에 의해 유지되는 상태 머신은 열린 전송된 상태 206에서 열린 상태 208로 전환할 수 있다. 또한, (2)에서 TTP_OPEN_ACK를 장치 A에 전송한 후, 장치 B에 의해 유지되는 상태 머신은 열린 수신된 상태 204에서 열린 상태 208로 전환할 수 있다.

[0108] (3)에서, 열린 상태 208에 있는 동안, 장치 A는 장치 B로부터 임의의 응답을 수신하기 전에 4개의 패킷들(예를 들어, TTP_PAYLOAD ID = 1~4)을 장치 B에 계속해서 또는 연속적으로 전송할 수 있다. 일부 실시예들에서, 장치

A가 장치 B로부터 임의의 응답을 수신하기 전에 장치 B에 전송할 수 있는 패킷들의 수는 제한된다. 장치 A로부터 수신된 패킷들에 응답하여, (4)에서, 장치 B는 장치 A에 의해 전송된 4개의 패킷들의 수신을 확인하는 4개의 패킷들(예를 들어, TTP_ACK ID = 1~4)을 전송할 수 있다.

[0109] (5)에서 장치 A는 TTP_CLOSE(패킷 ID = 5)를 장치 B로 전송한다. TTP_CLOSE를 전송한 후, 장치 A에 의해 유지되는 상태 머신은 열린 상태 208에서 닫힌 전송된 상태 212로 전환할 수 있다. 장치 A로부터 TTP_CLOSE를 수신한 것에 응답하여, 장치 B에 의해 유지되는 상태 머신은 열린 상태 208에서 닫힌 수신된 상태 210로 전환할 수 있다.

[0110] 그 후, (6)에서, 장치 B는 TTP_CLOSE_ACK(패킷 ID = 5)를 장치 A로 전송할 수 있다. TTP_CLOSE_ACK를 장치 A로 전송한 후, 장치 B에 의해 유지되는 상태 머신은 닫힌 수신된 상태 210에서 닫힌 상태 202로 다시 전환할 수 있다. 장치 B로부터 TTP_CLOSE_ACK를 수신한 후, 장치 A에 의해 유지되는 상태 머신은 닫힌 전송된 상태 212에서 다시 닫힌 상태 202로 전환할 수 있다. 따라서, 장치 A와 장치 B 간의 링크/연결은 닫힐 수 있다.

[0111] 도 3b는 본 개시에서 개시된 흐름 제어 프로토콜(예를 들어, TTP)와 연관된 "손실성의" 흐름 제어 특징("lossy" flow control feature)을 도시하고, 손실성은 손실되거나 손상된 패킷들이 비-확인응답을 수신한 후에 재전송된다는 것을 나타낼 수 있다.

[0112] 도 3b에 도시된 바와 같이, 장치 A는 닫힌 상태 202에 있는 동안 패킷 ID = 0인 TTP_OPEN을 장치 B로 전송할 수 있다. (1)에서 TTP_OPEN을 장치 B로 전송한 후, 장치 A에 의해 유지되는 상태 머신은 닫힌 상태 202에서 열린 전송된 상태 206으로 전환할 수 있다. 또한, (1)에서 장치 A로부터 TTP_OPEN을 수신한 후, 장치 B에 의해 유지되는 상태 머신은 닫힌 상태 202에서 열린 수신된 상태 204로 전환할 수 있다.

[0113] 그런 다음, (2)에서 장치 B로부터 TTP_OPEN_ACK를 수신한 후, 장치 A에 의해 유지되는 상태 머신은 열린 전송된 상태 206에서 열린 상태 208로 전환할 수 있다. 또한, (2)에서 TTP_OPEN_ACK를 장치 A에 전송한 후, 장치 B에 의해 유지되는 상태 머신은 열린 수신된 상태 204에서 열린 상태 208로 전환할 수 있다.

[0114] (3)에서, 열린 상태 208에 있는 동안, 장치 A는 장치 B로부터 임의의 응답을 수신하기 전에 4개의 패킷들(예를 들어, TTP_PAYLOAD ID = 1~4)을 장치 B에 계속해서 또는 연속적으로 전송할 수 있다. 그러나, 일부 네트워크 컨디션들로 인해, 장치 B는 패킷의 일부(예를 들어, TTP_PAYLOAD ID = 3)를 수신하지 못할 수 있다. 따라서, (4)에서, 장치 B는 장치 A에 의해 전송된 두 개의 패킷들(ID = 1 ~ 2)의 수신을 확인하지만 TTP_PAYLOAD ID = 3인 패킷은 수신되지 않음을 알리는 세 개의 패킷들(예를 들어, TTP_ACK ID = 1 ~ 2, TTP_NACK ID = 3)을 전송할 수 있다.

[0115] (5)에서, 장치 B로부터 패킷(예를 들어, TTP_NACK ID = 3)을 수신한 후, 장치 A는 두 개의 패킷들(예를 들어, TTP_PAYLOAD ID = 3~4)을 장치 B로 재전송한다. 특히, 패킷(예를 들어, TTP_NACK ID = 3)을 수신한 후 두 개의 패킷들의 재전송은 TTP의 "손실성의" 특징을 반영한다. 일부 실시예들에서, 장치 A는 타임-아웃이 발생한 후(예를 들어, 로컬 카운터(local counter)가 특정한 값을 초과하는 경우) 패킷들 중 일부를 재전송할 수 있다. 유리하게는, "손실성의" 특징은, TTP가 장치 A와 장치 B 간에 피어-투-피어 연결(peer-to-peer linking)의 존재 때문에 네트워크 흐름들을 제한 없이 제어 또는 스케일링할 수 있도록 하고, 일부 트래픽의 손실이 예상되는 큰(large) 시스템에서 TTP가 링크별 복구(link-specific recovery)를 달성하도록 한다.

[0116] (6)에서, 두 개의 패킷들(예를 들어, TTP_PAYLOAD ID = 3~4)을 수신한 후, 장치 B는 재전송된 패킷들(예를 들어, TTP_PAYLOAD ID = 3~4)의 수신을 확인하기 위해 두 개의 패킷들(예를 들어, TTP_ACK ID = 3~4)을 장치 A로 전송할 수 있다.

[0117] (7)에서, 장치 A는 장치 A와 장치 B 간의 링크를 닫기 위해 패킷(예를 들어, TTP_CLOSE ID = 5)을 장치 B로 전송할 수 있다. 또한, (7)에서, 장치 A에 의해 유지되는 상태 머신은 열린 상태 208에서 닫힌 전송된 상태 212로 전환할 수 있고, 장치 B에 의해 유지되는 상태 머신은 열린 상태 208에서 닫힌 수신된 상태 210로 전환할 수 있다.

[0118] (8)에서, 장치 B는, 링크를 닫는 것을 확인하고 동의하기 위해 패킷(예를 들어, TTP_CLOSE_ACK ID = 5)을 장치 A로 전송할 수 있다. 장치 B에 의해 유지되는 상태 머신은 닫힌 수신된 상태 210에서 닫힌 상태 202로 다시 전환할 수 있다. 장치 B로부터 패킷(예를 들어, TTP_CLOSE_ACK ID = 5)을 수신한 것에 응답하여, 장치 A에 의해 유지되는 상태 머신은 닫힌 전송된 상태 212에서 닫힌 상태 202로 다시 전환할 수 있다.

[0119] 일부 실시예들에서, 장치 A 및/또는 장치 B는 링크를 협상하는 프로세스(process of negotiating a link)가 완

료될 때까지 열린 상태 208로 전환하지 않거나 데이터 패킷들을 전송하지 않거나 수신하지 않을 수 있다. 예를 들어, 장치 A는 장치 B로부터 TTP_OPEN_ACK를 수신할 때까지 장치 B로 데이터 패킷들을 전송하지 않거나 장치 B로부터 데이터 패킷들을 수락하지 않을 수 있다. 이러한 구현들에서, 장치 A와 장치 B 간의 링크를 닫을 때, 특히 장치 A와 장치 B 간의 이전 링크가 닫힌 직후에 장치 A 또는 장치 B로부터 TTP_OPEN이 전송되는 경우에, 타임아웃 기간을 부과할 필요가 없을 수 있다.

[0120] 도 4는 본 개시의 실시예들에 따른 TTP를 구현하는 노드 400의 예시적인 블록도를 도시한다. 도 4에 도시된 바와 같이, 노드 400는 전송(TX) 경로 및 수신(RX) 경로를 포함할 수 있다. 도 4에 도시된 바와 같이, 노드 400의 프런트-엔드(front-end)는 OSI 모델의 계층 1(예를 들어, 물리적 계층)을 통한 통신들을 처리하는 물리적 코딩 서브계층(Physical Coding Sublayer; PCS) + 물리적 매체 연결(Physical Medium Attachment; PMA) 블록 402을 포함한다. 일부 실시예들에서, PCS + PMA 블록 402은 156.25MHz의 주파수(frequency)를 가지는 레퍼런스 클록(reference clock) 404에 기초하여 동작한다. 다른 실시예들에서, PCS + PMA 블록 402은 상이한 클록 주파수들에서 동작할 수 있다. PCS + PMA 블록 402은 이더넷 또는 IEEE 802.3 표준들과 호환될 수 있다. RX 경로의 데이터를 처리하기 위한 동작에서, PCS + PMA 블록 402은 RX serdes [3:0]를 입력들로서 수신하고 RX serdes [3:0]를 출력들(예를 들어, RX 프레임 408)로 다시 배열하여 TTP 매체 접근 제어(Medium Access Control; MAC) 블록 410에 의해 처리되도록 한다. TX 경로의 데이터를 처리하기 위한 동작에서, PCS + PMA 블록 402은 TTP MAC 블록 410으로부터 TX 프레임 412을 입력들로서 수신하고 데이터 포맷들을 재정렬하여 TX serdes [3:0]를 출력한다.

[0121] RX 경로에서, TTP MAC 블록 410은 RX 프레임 408을 입력들로서 수신하고 RDMA 수신된 데이터 416를 시스템-온-칩(System-on-chip; SoC) 420으로 출력한다. TX 경로에서, TTP MAC 블록 410은 SoC 420으로부터 RDMA 전송 데이터(RDMA send data) 418를 수신하고 TX 프레임 412을 PCS + PMA 블록 402에 출력한다. 도 4에 도시된 바와 같이, TTP MAC 블록 410은 OSI 모델의 계층 2~4의 동작들을 처리할 수 있다. TTP MAC 블록 410은 TTP 유한 상태 머신(finite state machine; FSM) 422을 포함할 수 있다. TTP FSM 422는 도 2에 도시된 바와 같이 상태 머신 200을 유지하고 업데이트할 수 있다. 위에서 논의된 바와 같이, 노드 400가 하나 이상의 다른 노드와 확립된 각각의 통신 링크에 대해, TTP FSM 422는 각각의 통신 링크와 연관된 흐름을 제어하기 위해 대응하는 상태 머신(예를 들어, 상태 머신 200)을 유지하고 업데이트할 수 있다.

[0122] 일부 실시예들에서, PCS + PMA 블록 402 및 TTP MAC 블록 410은 애플리케이션 특정 집적 회로(Application Specific Integrated Circuit; ASIC) 또는 필드 프로그래밍 가능한 게이트 어레이(Field Programmable Gate Array; FPGA)의 형태와 같은 하드웨어에 의해 구현될 수 있다. 따라서, PCS + PMA 블록 402 및 TTP MAC 블록 410은 소프트웨어/펌웨어/드라이버의 도움 또는 개입 없이 동작할 수 있다. 유리하게는, PCS + PMA 블록 402 및 TTP MAC 블록 410은 계층 1~계층 4의 통신과 연관된 레이턴시를 감소시키기 위해 소프트웨어 도움 없이 OSI 모델의 계층 1~계층 4의 통신들을 처리할 수 있다.

[0123] 도 5는 TTP에 따라 전송 또는 수신되는 패킷들에 대한 예시적인 헤더 500를 도시한다. 도 5에 도시된 바와 같이, 예시적인 헤더 500는 64바이트를 갖는다. 제1 16바이트는 이더넷 계층 2(예를 들어, 데이터 링크 계층) 및 가상 근거리 통신망(virtual local area network; VLAN) 동작을 위한 헤더를 포함한다. 제2 16바이트는 ETHTYPE을 포함하고 그 뒤에 선택적인 계층 3 인터넷 프로토콜(Internet Protocol; IP) 헤더가 온다. TTP에 기초한 계층 2 동작을 서포트하기 위해 ETHTYPE이 특정한 값(예를 들어, 0x9AC6)으로 설정될 수 있다. ETHTYPE이 특정한 값으로 설정되는 경우, 헤더 500는 헤더 500를 처리하는 네트워크 장치에 헤더 500가 TTP에 기초하여 포맷되었다는 신호를 보낼 수 있다. 제3 16바이트는 계층 3 (IP) 동작 및 UDP에 따른 계층 4 동작을 위한 선택적인 필드들을 포함한다. 제3 16바이트 및 제4 16바이트의 단부(end)는 TTP에 따른 계층 4 동작을 위한 필드들이다. TTP는 이더넷을 통한 TTP(TTP over Ethernet; TTPoE)라고도 지칭될 수 있다. 도 5에서 TTP가 TTPoE로 레이블링된다(labeled).

[0124] 바람직하게는, 예시적인 헤더 500는 TTP가 OSI 모델의 적어도 계층 2~계층 4으로부터의 이더넷 기반 네트워크를 통해 동작들을 서포트하도록 한다. 특히, 기존 이더넷 스위치들 및 하드웨어는 TTP와 연관된 동작들을 서포트할 수 있다.

[0125] 도 6은 본 개시의 실시예들이 구현될 수 있는 예시적인 네트워크 및 컴퓨팅 환경 600을 도시한다. 예시적인 네트워크 및 컴퓨팅 환경 600은 고성능 컴퓨팅 또는 인공지능 훈련 데이터 센터들에 대해 이용될 수 있다. 일 예로서, 네트워크 및 컴퓨팅 환경 600은 차량(예를 들어, 자동차)용 자율 주행 시스템에 의한 이용을 위한 데이터를 생성하기 위한 신경망 훈련(neural network training)에 이용될 수 있다. 도 6에 도시된 바와 같이, 예시적

인 네트워크 및 컴퓨팅 환경 600은 이더넷 스위치 608, 호스트들 602A~602E, 고속 주변장치 연결 인터페이스(Peripheral Component Interconnect Express; PCIe) 호스트들 604A~604N, 컴퓨팅 타일(computing tile)들 606A~606N을 포함한다. 도 6에는 호스트들 602A~602E이 5개 있지만, 5개보다 많거나 적은 임의의 적합한 수의 호스트들이 구현될 수 있다. 또한, PCIe 호스트들의 수 및 컴퓨팅 타일들의 수는 임의의 적합한 양의 정수일 수 있다.

[0126] 호스트들 602A~602E 각각은 네트워크 인터페이스 카드(NIC), 중앙 처리 장치(CPU), 동적 랜덤 액세스 메모리(DRAM)를 포함한다. CPU로서 예시되었지만, 일부 실시예들에서, CPU는 임의의 타입의 싱글-코어, 싱글-스레드, 멀티-코어 또는 멀티-스레드 프로세서, 마이크로프로세서, 디지털 신호 프로세서(digital signal processor; DSP), 마이크로콘트롤러 또는 기타 프로세서 또는 처리/제어 회로로서 구현될 수 있다. DRAM으로서 예시되었지만, 일부 실시예들에서, DRAM은 대안적으로 또는 추가적으로 정적 랜덤 액세스 메모리(static random access memory; SRAM), 동기식 DRAM(synchronous DRAM; SDRAM), 이중 데이터 속도 동기식 동적 랜덤 액세스 메모리(double data rate synchronous dynamic random access memory; DDR SDRAM)와 같은 임의의 타입의 휘발성(volatile) 또는 비-휘발성(non-volatile) 메모리 또는 데이터 스토리지로서 구현될 수 있다. DRAM은 호스트들 602A~602E의 동작 동안 이용되는, 운영 체제들, 애플리케이션 프로그램들, 라이브러리들, 드라이버 등을 포함하는 다양한 데이터 및 프로그램 코드를 저장할 수 있다.

[0127] 일부 실시예들에서, NIC는 이더넷 스위치 608와 통신하기 위해 TTP를 구현할 수 있다. 각각의 NIC와 네트워크 인터페이스 프로세서(NIP) 간에 확립된 링크를 이더넷 스위치 608을 통해 관리하기 위해 흐름 제어 프로토콜로서 TTP를 이용하여 이더넷 스위치 608와 통신할 수 있다. 일부 실시예들에서, NIC는 도 4의 PCS + PMA 블록 402 및 TTP MAC 블록 410을 포함할 수 있다. 일부 실시예들에서, NIC는 소프트웨어/펌웨어의 도움 없이 TTP를 구현할 수 있다.

[0128] 도 6에 도시된 바와 같이, PCIe 호스트들 604A~604N 각각은 네트워크 인터페이스 프로세서(NIP) 및 고-대역폭 메모리(HBM)를 포함할 수 있다. 일부 실시예들에서, HBM에 의해 서포트되는 대역폭은 컴퓨팅당 32기가바이트(GB)일 수 있다. PCIe 호스트들 604A~604N 각각은 컴퓨팅 타일들 606A~606N 각각과 통신할 수 있다. 컴퓨팅 타일들 606A~606N 각각은 스토리지, 입력/출력, 계산 리소스들을 포함할 수 있다. 컴퓨팅 타일 606A은 고성능 컴퓨팅을 위한 프로세서들의 어레이를 가지는 웨이퍼(wafer)의 시스템을 포함할 수 있다. 특정한 애플리케이션들에서, 컴퓨팅 타일들 606A~606N 각각이 초당 9페타 부동 소수점 연산(peta floating point operations per second; PFLOPS)을 수행하거나, 정적 랜덤 액세스 메모리(SRAM)를 이용하여 11기가바이트(GB)의 크기의 데이터를 저장하거나, 초당 36테라바이트(TB) 대역폭에서 입력/출력 동작들을 용이하게 할 수 있다.

[0129] 일부 실시예들에서, 호스트들 602A~602E의 NIC들 각각은 PCIe 호스트들 604A~604N의 NIP들 각각과 통신 링크를 열고 닫을 수 있다. 구체적으로, 하나의 NIC 및 하나의 NIP는 도 2의 상태 머신 200을 구현하는 것에 의해 서로의 통신 링크를 열고 닫을 수 있다. 통신 링크를 열고 닫기 위해 NIC와 NIP는, 바람직한 동작들을 수행하기 위해, 도 7a-도 7b의 연산 코드들을 포함하는 패킷들을 이용할 수 있다. 예를 들어, NIP와의 링크를 열기 위해, NIC는 통신 링크를 열 것을 요청하기 위해 연산 코드 TTP_OPEN(도 7a에 도시)를 포함하는 패킷을 NIP에 전송할 수 있다. 연산 코드 TTP_OPEN를 포함하는 패킷을 수신한 후, NIP는 도 2의 닫힌 상태 202에서 열린 수신된 상태 204로 전환할 수 있다. 도 7a에 도시된 연산 코드 TTP_OPEN_ACK를 포함하는 패킷을 전송한 후, NIP는 도 2에 도시된 바와 같이 열린 수신된 상태 204에서 열린 상태 208로 전환할 수 있다. 일부 실시예들에서, 통신 링크가 확립되면(예를 들어, NIC 및 NIP가 모두 열린 상태 208에 있는 경우), NIC 및 NIP는 도 5의 헤더 500를 이용하여 서로 패킷들을 전송하거나 수신할 수 있다. 즉, NIC와 NIP 간에 전송되거나 수신되는 패킷들 각각은 도 5의 헤더 500를 포함할 수 있다.

[0130] 도 6에 나타난 바와 같이, 호스트들 602A~602E 각각, PCIe 호스트들 604A~604N 각각, 컴퓨팅 타일들 606A~606N 각각 또는 이더넷 스위치 608 간의 통신 및 데이터 교환은 TTP에 기초하여 수행될 수 있다. 위에서 설명된 기술들을 이용하여 TTP를 통해 달성된 더 짧은 레이턴시(TCP와 비교했을 때)으로 인해, 도 6의 다양한 요소들 사이의 고-대역폭 및 고속-통신이 달성될 수 있다. 일부 실시예들에서, 도 6에 도시된 NIP들의 적어도 일부 또는 NIC들의 적어도 일부는 도 4의 노드 400와 유사하거나 동일하게 구현될 수 있다. 도 6 전체에 걸쳐 예시되어 있지는 않지만, 일부 실시예들에서 NIC들 및 NIP들 각각은 패킷들이 수신될 수 있고 전송될 수 있는 포트 610를 포함할 수 있다. 일부 실시예들에서, 포트 610는 이더넷 포트이다.

[0131] 도 7a-도 7b는 본 개시의 실시예들에 따른 다양한 타입들의 TTP 패킷들의 연산 코드들을 도시한다. 도 7a 및 도 7b에 도시된 TTP 패킷들은 네트워크들의 노드들 간의 링크를 닫고 열기 위해 도 2, 도 3a 및 도 3b에서 이용된

다. TTP 패킷들은 도 6의 컴퓨팅 환경과 네트워크의 노드들 간에서 교환될 수 있다. 도 7a 및 도 7b에 도시된 TTP 패킷들은 도 2, 도 3a 및 도 3b와 함께 더 잘 이해될 수 있다.

[0132] **패킷 리플레이 하드웨어 아키텍처**

[0133] TTP를 이용하여 패킷들을 전송 및/또는 수신하는 노드 400의 예시적인 블록도를 예시하는 도 4를 다시 참조하여 리플레이 하드웨어 아키텍처가 설명될 것이다. 위에서 언급된 바와 같이, 노드 400는, 계층 1~계층 4의 통신과 연관된 레이턴시를 감소시키기 위해 소프트웨어의 도움 없이 OSI 모델의 계층 1~계층 4로부터의 통신들을 처리하는 TTP FSM 422를 포함하는 TTP 매체 접근 제어(MAC) 블록 410 및 물리적 코딩 서브계층(PCS) + 물리적 매체 연결(PMA) 블록 402과 같은 블록들을 포함할 수 있다. 또한, 노드 400의 TTP 매체 접근 제어(MAC) 블록 410은, 적어도 TTP(피어 링크) 태그 블록 436, RX 데이터경로(Datapath) 432, RX 스토리지 432-1(예를 들어, SRAM에 있음), TX 데이터경로 434, TX 스토리지 434-1(예를 들어, SRAM에 있음)을 포함하는 하드웨어 리플레이 아키텍처를 포함할 수 있다. 하드웨어 리플레이 아키텍처는 TTP와 같은 손실성의 프로토콜에 따라 전송 동안 손실된 패킷들을 리플레이할 수 있다. 선택적으로, 노드 400의 TTP 매체 접근 제어(MAC) 블록 410은 시스템-온-칩(SoC) 420으로부터 RDMA 전송 데이터 418를 수신하고 인코딩할 수 있는 TTP MAC RDMA 주소 인코딩 블록 438을 더 포함할 수 있다.

[0134] 일부 실시예들에서, 패킷들을 리플레이하기 위한 노드 400의 하드웨어 리플레이 아키텍처는 적어도 TTP 태그 블록 436, RX 데이터경로 432, RX 스토리지 432-1, TX 스토리지 434-1 및 TX 데이터경로 434의 회로를 포함할 수 있다. 위에서 논의된 바와 같이, 하드웨어 리플레이 아키텍처는 다양한 링크에서 전송된 및/또는 수신된 패킷들을 저장하고, 특히 리플레이가 발생할 때 전송된 패킷들의 순서를 유지하기 위해, 물리적 스토리지 및 데이터 구조를 이용할 수 있다. 일부 실시예들에서, 하드웨어 리플레이 아키텍처에 의해 이용되는 물리적 스토리지는 하나 이상의 링크와 연관된 패킷들을 저장, 버퍼링 및/또는 홀드할 수 있는 임의의 적합한 타입의 로컬 스토리지 또는 캐시(예를 들어, 낮은-수준의 캐시들)일 수 있다. 물리적 스토리지는 크기가 메가바이트(megabyte; MB) 또는 킬로바이트(kilobyte; KB)의 단위와 같이 제한될 수 있다. 일부 예들에서, 물리적 스토리지는 TX 데이터경로 434의 일부로서 배치되거나, 보다 구체적으로는 TX 스토리지 434-1의 일부로서 배치될 수 있다. 물리적 스토리지는 RX 데이터경로 432의 일부로서 배치되거나, 보다 구체적으로는 RX 스토리지 432-1의 일부로서 배치될 수 있다. 예를 들어, 물리적 스토리지는 RX 스토리지 432-1 및 TX 스토리지 434-1일 수 있고, RX 데이터경로 432 및 TX 데이터경로 434 각각과 연관된 하드웨어 리플레이 아키텍처에 의해 이용되는 RX 스토리지 432-1 및 TX 스토리지 434-1의 크기는 256KB일 수 있다. 다른 예들에서, 물리적 스토리지는 TTP 태그 블록 436의 일부로서, 및 TTP 태그 블록 436 내에 (예를 들어, TTP 태그 블록 436 내에 배치된 로컬 스토리지로서) 배치될 수 있다.

[0135] 노드 400의 TTP 매체 접근 제어(MAC) 블록 410 내의 하드웨어 리플레이 아키텍처에 의해 임의의 다른 적합한 크기의 물리적 스토리지가 채택될 수 있다는 점에 유의해야 한다. 일부 실시예들에서, 하드웨어 리플레이 아키텍처에 의해 이용되는 데이터 구조(예를 들어, TTP 태그 블록 436 내)는 하나 이상의 링크된 리스트를 포함할 수 있고, 각각의 링크된 리스트는 제1 통신 노드와 제2 통신 노드 간에 확립된 대응하는 링크에 대해 전송된 패킷들의 순서를 기록 및/또는 추적할 수 있다. 일부 실시예들에서, TTP 태그 블록 436은, 다수의 링크들을 통해 전송된 패킷들을 리플레이하기 위해 저장된 패킷들을 유지 및 관리하기 위해, 물리적 스토리지(예를 들어, RX 스토리지 432-1 및 TX 스토리지 434-1)와 함께 링크된 리스트들을 이용할 수 있다.

[0136] 도 8 및 도 9는, 본 개시의 일부 실시예들에 따른 패킷들을 리플레이 또는 재전송하기 위해 TTP를 구현하는 이더넷-기반 네트워크에서 노드(예를 들어, 노드 400 또는 도 3b의 장치 A)에 의해 이용되는 예시적인 물리적 스토리지 및 데이터 구조(예를 들어, TX 링크된 리스트 952)를 예시한다. 도 8 및 도 9는, 장치 A가 패킷(TTP_PAYLOAD ID = 3)이 수신되지 않았음을 알리는 비-확인응답 패킷(예를 들어, TTP_NACK ID = 3)을 수신한 것에 응답하여 두 개의 패킷들(예를 들어, TTP_PAYLOAD ID = 3~4)을 리플레이하는 것을 도시하는 도 3b를 참조하여 이해될 수 있다.

[0137] 도 8을 참조하면, 도 3b의 장치 A는, 물리적 스토리지, 예를 들어 패킷 물리적 캐시 802의 전송 및/또는 리플레이를 위해, 패킷 1(예를 들어, 도 3b의 패킷 TTP_PAYLOAD ID = 1), 패킷 2(예를 들어, 도 3b의 패킷 TTP_PAYLOAD ID = 2), 패킷 3(예를 들어, 도 3b의 패킷 TTP_PAYLOAD ID = 3), 패킷 4(예를 들어, 도 3b의 패킷 TTP_PAYLOAD ID = 4), 패킷 5(예를 들어, 도 3b의 패킷 TTP_CLOSE ID = 5)를 저장할 수 있다. 위에서 언급된 바와 같이, 패킷 물리적 캐시 802는 TX 스토리지 434-1일 수 있고 및/또는 TTP 태그 블록 436 내에 배치된 물리적 스토리지일 수 있다. 일부 실시예들에서, 패킷 물리적 캐시 802는 두 개의 스토리지 공간들(패킷 물리적 태그 804 및 패킷 물리적 데이터 806)을 가질 수 있다. 패킷들(예를 들어, 패킷 1~패킷 5) 각각에 대해, 패킷 물

리적 태그 804는 패킷을 저장하는 패킷 물리적 데이터의 물리적 주소를 가리키는 물리적 주소 포인터(physical address pointer)를 포함할 수 있다. 예를 들어, 패킷 물리적 태그 804의 엔트리에 저장된 패킷 4와 연관된 물리적 주소 포인터 808는 패킷 4(예를 들어, 도 3b의 패킷 TTP_PAYLOAD ID = 4)가 저장된 패킷 물리적 데이터 806의 엔트리를 가리킬 수 있다. 도 8에 도시된 바와 같이, 장치 A는 패킷 1, 패킷 2, 패킷 3, 패킷 4 및 패킷 5를 순서 820로 전송할 수 있다(예를 들어, 패킷 1을 먼저 전송하고 패킷 5를 마지막으로 전송). 그러나 장치 A는 순서 820에 기초하여 패킷 물리적 데이터 806에 패킷 1~패킷 5를 저장하지 않을 수 있다. 구체적으로, 장치 A가 패킷 3을 패킷 4 및 패킷 5보다 먼저 전송하더라도, 패킷 3을 저장하는 패킷 물리적 데이터 806의 주소 810는 패킷 4 및 패킷 5를 각각 저장하는 패킷 물리적 데이터 806의 주소 812 및 주소 814 뒤에 올 수 있다.

[0138] 도 9는, 이전 전송과 리플레이 간의 패킷 전송의 순서를 유지하기 위해 도 3b의 노드 400 및/또는 장치 A에 의해 이용되는 TX 링크된 리스트 952를 예시한다. TX 링크된 리스트 952는 노드 400의 TTP 태그 블록 436의 일부일 수 있다. 도 8을 논의하는 위에서 언급된 바와 같이, 도 3b의 장치 A는, 패킷 1~패킷 5이 전송되는 순서 820를 반영하지 않는 패킷 물리적 데이터 806의 다양한 주소들에서 패킷 1~패킷 5을 저장할 수 있다. 그럼에도 불구하고, 장치 A는, 패킷 1~패킷 5을 전송하는 것의 순서를 추적하고 유지하기 위해, TX 링크된 리스트 952를 이용할 수 있다. 도 9에 도시된 바와 같이, TX 링크된 리스트 952는 5개의 요소들 960, 962, 964, 968, 970을 포함하고, 각각의 요소는 패킷 1~패킷 5 중 하나에 대응하거나 패킷 1~패킷 5 중 하나와 연관된다. 도 9는 TX 링크된 리스트 952가 패킷 1~패킷 5을 전송하는 것의 순서 820를 추적하고 유지하는 것을 예시한다. 예를 들어, TX 링크된 리스트 952에서, 패킷 3에 대응하는 요소 964는 패킷 4에 대응하는 요소 968 앞에 와서 이를 가리키고, 패킷 4에 대응하는 요소 968는 패킷 5에 대응하는 요소 970 앞에 와서 이를 가리킨다. 따라서, TX 링크된 리스트 952를 이용하는 것에 의해, 장치 A는 이전 전송 및 리플레이 동안 패킷 전송의 순서를 유지할 수 있고, 리플레이는 도 3b에서 TTP_PAYLOAD ID = 3인 패킷이 장치 B에 의해 수신되지 않았다는 것을 알리는 TTP_NACK ID = 3 패킷을 수신하는 것에 응답하여 트리거될 수 있다. 리플레이는 본 명세서에 개시된 임의의 적합한 원리들 및 장점들에 따른 타임아웃 또는 비-확인응답에 대응하여 트리거될 수 있다.

[0139] 도 9에 도시된 바와 같이, 도 3b의 장치 A는, 어떤 패킷(들)을 리플레이할지 결정하기 위해, 메모리에 저장된 하나 이상의 포인터 972, 974 및 976을 더 이용할 수 있다. 도 3b의 (3) 및 (4)에 도시된 바와 같이, 장치 A는 4개의 패킷들(예를 들어, TTP_PAYLOAD ID = 1~4)을 전송하고 3개의 패킷들(예를 들어, TTP_ACK ID = 1~2, TTP_NACK ID = 3)을 수신하여, 장치 A에 의해 전송된 2개의 패킷들(ID = 1~2)에 대한 수신을 확인하지만 TTP_PAYLOAD ID = 3인 패킷이 수신되지 않았음을 알린다. 이에 응답하여, 장치 A는, 장치 A가 패킷 3으로부터 시작하여 패킷들을 리플레이해야 함을 나타내기 위해, 패킷 3에 대응하는 요소 964를 가리키도록 포인터 972를 설정할 수 있다. 장치 A는 패킷 5뿐만 아니라 패킷 4도 리플레이해야 함을 나타내기 위해 패킷 4에 대응하는 요소 968를 가리키도록 포인터 974를 설정할 수도 있다. 장치는, 패킷 3 및 패킷 4를 리플레이한 후 장치 A가 패킷 5를 전송할 수 있음을 나타내기 위해 패킷 5에 대응하는 요소 970을 가리키도록 포인터 976을 더 설정할 수 있다. 또한, 장치 A는, 장치 A에 의해 전송된 또는 수신된 패킷들을 저장하기 위한 더 많은 스토리지 공간을 확보(free up)하기 위해 패킷 1 및 패킷 2가 패킷 물리적 데이터 806 및 패킷 물리적 태그 804의 주소들(도 8에 미도시됨)로부터 제거될 수 있다는 것을 나타내기 위해, TX 링크된 리스트 952의 요소 960 및 요소 962를 null로 설정할 수 있다.

[0140] 그 후, TX 링크된 리스트 952, 포인터 972 및 포인터 974에 기초하여 장치 A는 도 3b의 (5)에 예시된 바와 같이 패킷 3 및 패킷 4를 리플레이할 수 있다. 그러면, 장치 A는 도 3b의 (6)에 도시된 바와 같이 패킷 3 및 패킷 4을 수신한 것의 확인응답을 수신할 수 있다. 이에 대응하여, 장치 A는, TX 링크된 리스트 952에 기초하여, 패킷 1~패킷 5의 전송 및 리플레이를 완료하기 위해 TX 링크된 리스트 952의 요소 970에 대응하는 패킷 5(예를 들어, 도 3b의 패킷 TTP_CLOSE ID = 5)을 전송할 수 있다. 추가적으로 또는 선택적으로, 장치 A는 TX 링크된 리스트 952의 요소들에 대응되는 모든 패킷들이 전송되고 리플레이된 후 패킷 1~패킷 5에 의해 차지된 스토리지 공간을 해제할 수 있다. 일부 실시예들에서, 장치 A는, 자유 리스트 엔트리(free list entry) 832 및 자유 리스트 엔트리 834를 각각 특정한 값으로 설정하는 것에 의해, 패킷 물리적 태그 804의 주소들 및 패킷 물리적 데이터 806의 주소들이 해제되고 다른 패킷들에 대응하는 다른 링크된 리스트(들)과 함께 이용할 수 있도록 확보되었음을 나타낼 수 있다.

[0141] 도 10은 본 개시의 일부 실시예들에 따른 도 4의 TTP 태그 블록 436의 예시적인 블록도를 도시하고, TTP 태그 블록 436은 다수의 링크들을 통해 전송되는 패킷들을 리플레이하기 위한 하드웨어 리플레이 아키텍처의 일부이다. 도 10에 도시된 바와 같이, TTP 태그 블록 436은 파이프라인 스테이지들 1002, 1004, 1006 및 1008에서 각각 동작하는 TX 링크된-리스트 1020 및 논리 회로 1012, 1014, 1016 및 1018을 저장하는 메모리를 포함할 수 있

다. 논리 회로 1012, 1014, 1016 및 1018는 임의의 적합한 물리적 회로에 의해 구현될 수 있다. 일부 예들에서, 논리 회로 1012, 1014, 1016 및 1018 중 일부 또는 전부는 애플리케이션 특정 집적 회로(ASIC)의 형태와 같은 전용 회로(dedicated circuitry)에 의해 구현될 수 있다. 일부 예들에서, 논리 회로 1012, 1014, 1016 및 1018 중 일부 또는 전부는, 필드 프로그래밍 가능한 게이트 어레이(FPGA) 또는 디지털 신호 프로세서(DSP)와 같은 범용 처리 회로(general purpose processing circuitry) 또는 프로그래밍 가능한 논리 게이트들에 의해 구현될 수 있다. 동작에서, TX 링크된-리스트 1020는 도 9의 TX 링크된 리스트 952와 유사하게 기능할 수 있다. 일부 실시예들에서, TX 링크된-리스트 1020는 패킷 1022, 패킷 1024 및 패킷 1026을 포함하는 N개의 패킷들의 순서를 추적하고, 노드 400는 TX 링크된-리스트 1020에 의해 추적된 N개의 패킷들을 특정한 링크를 통해 전송할 수 있다. TTP 태그 블록 436은 패킷 1022, 패킷 1024 및 패킷 1026을 각각 가리키는 포인터 1032, 포인터 1034 및 포인터 1036를 더 포함한다. TTP 태그 블록 436은 포인터 1032, 포인터 1034 및 포인터 1036를 적합한 스토리지 요소(도 10에 미도시됨)에 저장할 수 있다. 특정한 애플리케이션들에서, TX 링크된-리스트 1020의 패킷 1022, 패킷 1024 및 패킷 1026을 포함하는 N개의 패킷들은 노드 400의 TX 데이터경로 434의 TX 스토리지 434-1와 같은 물리적 스토리지에 저장될 수 있다. 이러한 애플리케이션들에서, TX 링크된-리스트 1020는 패킷들 1022, 1024, 1026에 대한 포인터들을 포함할 수 있다. 다른 애플리케이션들에서, 패킷 1022, 패킷 1024 및 패킷 1026을 포함하는 N개의 패킷들은 TTP 태그 블록 436 내의 물리적 스토리지에 저장된 TX 링크된-리스트 1020의 일부일 수 있다.

[0142] 일부 실시예들에서, 노드 400는 TTP에 따라 확립된 링크를 이용하여 제2 노드로 전송된 N개의 패킷들(패킷 1022, 패킷 1024 및 패킷 1026 포함)을 TX 스토리지 434-1(또는 노드 400의 다른 물리적 스토리지)에 저장할 수 있고, N은 TX 스토리지 434-1의 크기에 의해 제한될 수 있는 양의 정수이다. 노드 400는, TTP 및/또는 네트워크 컨디션들로부터의 제약들이 허락하는 한 N개의 패킷들 중 일부 또는 전부를 제2 노드로 계속해서 전송할 수 있다. 패킷 1022, 패킷 1024 및 패킷 1026을 포함하는 N개의 패킷들을 리플레이하는 것을 수용하기 위해, TX 스토리지 434-1는 제2 노드로부터 하나 이상의 패킷을 수신한 것의 확인응답이 수신될 때까지 이미 전송된 하나 이상의 패킷(예를 들어, 패킷 1022)을 계속 저장할 수 있다. 패킷은 이전에 전송된 패킷들의 수신이 확인될 때까지 저장될 수 있다. 패킷을 수신한 것의 확인응답이 수신되는 경우, TX 스토리지 434-1는 노드 400과 제2 노드 및/또는 하나 이상의 다른 노드 간의 링크 또는 다른 링크를 통해 전송될 패킷들을 저장하기 위한 공간을 만들기 위해 패킷을 삭제할 수 있다. 대조적으로, 패킷에 대한 비-확인응답이 수신된 경우(예를 들어, 제2 노드가 노드 400에 패킷이 수신되지 않음을 알리는 경우) 또는 제2 노드로부터 패킷을 수신한 것의 확인응답 또는 비-확인응답을 수신하는 것 없이 타임아웃이 발생한 경우, 노드 400는 TX 스토리지 434-1에 여전히 저장되어 있는 패킷을 리플레이(예를 들어, 패킷을 제2 노드로 재전송)할 수 있다. 패킷을 리플레이하는 것과 관련하여, 노드 400는 수신의 확인응답을 수신한 다른 패킷들을 삭제할 수 있다. 일부 실시예들에서, TX 링크된-리스트 1020는, 패킷 1022, 패킷 1024 및 패킷 1026을 포함하는 N개의 패킷들 중 일부 또는 전부의 이전 전송과 그 이후의 임의의 리플레이 간의 순서를 유지하기 위해, TX 스토리지 434-1와 협력할 수 있다. 도 10에 도시된 바와 같이, TX 링크된-리스트 1020는 N개의 요소들을 포함하고, 각각의 요소는 N개의 패킷들 각각에 대응하거나 N개의 패킷들 각각을 포함하고, 다음 패킷에 대응하는 다음 요소에 대한 레퍼런스를 포함한다.

[0143] N개의 패킷들을 전송 및/또는 리플레이하는 경우, 패킷이 리플레이를 위해 보관되어야 할지 또는 TX 스토리지 434-1에 의해 스토리지 리소스들을 보존하기 위해 삭제되어도 되는지를 결정하기 위해 TTP 태그 블록 436은 TX 링크된-리스트 1020의 세 개의 요소들을 각각 가리키는 포인터 1032, 포인터 1034 및 포인터 1036를 더 이용할 수 있다. N이 9인 경우(예를 들어, 노드 400에서부터 제2 노드로 전송된 9개의 패킷들)를 예로 들면, TX 링크된-리스트 1020에서, 1번째 요소는 1번째 패킷(예를 들어, 패킷 1022) 및 1번째 레퍼런스에 대응하고, 1번째 레퍼런스는 2번째 요소를 가리키고; 2번째 요소는 2번째 패킷 및 2번째 레퍼런스에 대응하고, 2번째 레퍼런스는 제3 요소를 가리키고; 및 8번째 요소는 8번째 패킷(예를 들어, 패킷 1024) 및 8번째 레퍼런스에 대응하고, 8번째 레퍼런스는 9번째 요소를 가리키고; 및 9번째 요소는 9번째 패킷(예를 들어, 패킷 1026)에 대응한다. TTP 태그 블록 436은 각각 1번째 요소(예를 들어, 패킷 1022), 8번째 요소(예를 들어, 패킷 1024) 및 9번째 요소(예를 들어, 패킷 1026)를 가리키는 세 개의 포인터들 1032, 1034 및 1036을 유지하고 업데이트할 수 있다.

[0144] 노드 400가 1번째 패킷~9번째 패킷을 전송했고, 제2 노드로부터 1번째 패킷~7번째 패킷을 수신했지만 8번째 패킷~9번째 패킷은 수신하지 않은 확인응답을 수신하였다고 더 가정하면, 포인터 1032는 TX 링크된-리스트 1020의 1번째 요소(예를 들어, 패킷 1022)를 가리키고, 포인터 1034는 TX 링크된-리스트 1020의 8번째 요소(예를 들어, 패킷 1024)를 가리키고, 포인터 1036는 TX 링크된-리스트 1020의 9번째 요소(예를 들어, 패킷 1026)를 가리킨다. 따라서, TTP 태그 블록 436은 TX 스토리지 434-1로 하여금 포인터들 1032, 1034 및 1036에 기초하여 N개의 패킷들 중 일부 또는 전부를 리플레이하고 패킷 1022, 패킷 1024 및 패킷 1026을 포함하는 N개의 패킷들

중 일부 또는 전부를 삭제하도록 할 수 있다. 보다 구체적으로, TX 스토리지 434-1는 포인터 1034가 가리키는 패킷 1024부터 포인터 1036가 가리키는 패킷 1026을 리플레이할 수 있다(이 경우, 패킷 1024 및 패킷 1026만 리플레이된다). TX 스토리지 434-1는 나머지 패킷들(예를 들어, 포인터 1032가 가리키는 패킷 1022 및 패킷 1024 전의 이전에 전송된 다른 패킷들, 이 경우, 패킷 1022을 포함한 7개의 패킷들이 삭제될 수 있음)을 더 삭제할 수 있다.

[0145] 도 10에 도시된 바와 같이, TTP 태그 블록 436(예를 들어, 논리 회로 1012, 1014, 1016 및 1018)의 일부 또는 전부는 노드 400의 처리량을 증가시키기 위해 파이프라인 방식으로 동작할 수 있다. 논리 회로 1012, 1014, 1016 및 1018는, 패킷들이 리플레이되어야 하는지 또는 패킷들을 저장하는 노드 400의 TX 스토리지 434-1 또는 다른 물리적 스토리지로부터 삭제/폐기되어야 하는지를 결정하기 위해, TX 링크된-리스트 1020과 함께 동작할 수 있다. 도 10에 도시된 바와 같이, 논리 회로 1012, 1014, 1016 및 1018는 TTP 태그 블록 436이 동작하는 클록에 따라 각각의 파이프라인 스테이지들에서 동작할 수 있다. 구체적으로, 논리 회로 1012는 초기 파이프라인 스테이지(initial pipelined stage) 1002("Q0"로 레이블링됨)에서 동작하고, 논리 회로 1014는 제1 파이프라인 스테이지 1004("Q1"로 레이블링됨)에서 동작하고, 논리 회로 1016는 제2 파이프라인 스테이지 1006("Q2"로 레이블링됨)에서 동작하고, 논리 회로 1018는 제3 파이프라인 스테이지 1008("Q3"로 레이블링됨)에서 동작한다.

[0146] 동작에서, 논리 회로 1012는 TTP 링크 태그 파이프라인에서 처리할 데이터 스트림(data stream)들 중 하나를 선택할 수 있다. 초기 파이프라인 스테이지 1002에 도시된 바와 같이, 논리 회로 1012는, 제어 신호(예를 들어, "Pick")에 기초하여, TTP 링크 태그 파이프라인에서 처리하기 위해 전송 스트림(transmitting stream)("TX QUEUE"), 수신 스트림(receiving stream)("RX QUEUE") 또는 확인 스트림(acknowledging stream)("ACK QUEUE") 중 하나를 선택할 수 있다. TTP 링크 태그 파이프라인에서, 논리 회로는 선택된 데이터 스트림의 하나 이상의 패킷을 리플레이할지, 또는 선택된 데이터 스트림의 하나 이상의 패킷을 폐기할지를 결정한다. TTP 링크 태그 파이프라인은 TTP 태그 파이프라인이 리플레이하기로 결정한 또 다른 패킷 이후에 전송된 패킷의 확인응답을 거부하기로 결정할 수도 있다.

[0147] 논리 회로 1012가 패킷들을 리플레이하는 것을 준비하기 위해 전송 스트림을 선택한다고 가정하면, 제1 파이프라인 스테이지 1004에서 논리 회로 1014는 리플레이를 위해 어떤 링크를 평가할지 결정한다. 여기에는 링크들과 연관된 태그들을 판독하는 것이 포함될 수 있다. 도 10에 도시된 바와 같이, 논리 회로 1014는 가능한 리플레이를 위해 두 개의 링크들(예를 들어, "MOOSEs" 및 "CATs") 중 하나를 선택할 수 있고, 각각의 링크는 동일한 엔드포인트(endpoint) 또는 상이한 엔드포인트들 간에 확립될 수 있다. 예를 들어, 링크 "MOOSEs" 및 링크 "CATs"는 모두 노드 400와 제2 노드 간에 확립될 수 있다; 대안적으로, 링크 "MOOSEs"는 노드 400와 제2 노드 간에 확립되고 링크 "CATs"는 노드 400와 제3 노드 간에 확립될 수 있다. 논리 회로 1014는 선택된 링크를 가리키는 링크 포인터에 기초하여 리플레이하기 위한 링크(예를 들어, "CATs")를 선택할 수 있다.

[0148] 그런 다음, 제2 파이프라인 스테이지 1006에서, 논리 회로 1016는 링크 "CATs"를 통해 전송된 패킷(들) 중 어떤 것이 리플레이되는지 또는 폐기되는지를 결정할 수 있다. 일부 실시예들에서, 논리 회로 1016는 링크 "CATs"를 통해 전송된 패킷들 중 일부를 리플레이하도록 결정하고, 수신의 확인응답 또는 비-확인응답이 수신되었는지에 기초하여 다른 패킷들이 폐기될 수 있다. 예를 들어, 논리 회로 1016는 패킷 1024의 비-확인응답의 수신이 수신되거나 패킷 1024의 확인응답이 타임아웃을 트리거하는 시간 주기 동안 수신되지 않을 경우 패킷 1024을 리플레이하도록 결정할 수 있다. 대조적으로, 논리 회로 1016는 패킷 1022의 확인응답의 수신에 응답하여 패킷 1022을 폐기하기로 결정할 수 있다. 추가적으로 및/또는 선택적으로, 논리 회로 1016는 TX 링크된-리스트 1020에 기초하여 링크 "CATs"를 통해 전송된 다른 패킷들을 리플레이 및/또는 폐기하도록 더 결정할 수 있다. 예를 들어, 패킷 1026이 패킷 1024 이후에 전송되었음을 나타내는 TX 링크된-리스트 1020에 의해 지정된 링크 "CATs"를 통해 전송된 패킷들의 순서에 기초하여, 논리 회로 1016는 패킷 1024의 비-확인응답 수신에 응답하여 패킷 1024을 리플레이하는 것과 함께 패킷 1026을 리플레이하도록 결정할 수 있다. 논리 회로 1016는 추가로, 패킷 1022과 패킷 1024 간에 전송된 패킷들의 확인응답들이 수신되었다고 가정하여, TX 스토리지 434-1에서 이용 가능한 스토리지 공간을 더 확보하기 위해, 패킷 1022과 패킷 1024 간에 전송된 패킷들을 TX 스토리지 434-1로 하여금 폐기하도록 할 수 있다. 제2 파이프라인 스테이지 1006에서, 패킷에 대한 확인응답은 이전에 전송된 패킷을 리플레이할지 결정하는 것과 관련하여 거부될 수 있다. 패킷을 폐기한다는 것은 패킷 대신 다른 데이터를 메모리에 쓰거나 메모리로부터 패킷을 삭제하는 것을 포함할 수 있다.

[0149] 그 후, 제3 파이프라인 스테이지 1008에서, 논리 회로 1018는 링크 "CATs"를 가리키는 링크 포인터를 또 다른 링크(예를 들어, 링크 "MOOSEs")를 가리키도록 업데이트할 수 있다. 따라서, 파이프라인 동작의 다음 라운드(following round)에서, 논리 회로 1012, 1014, 1016 및 1018는 링크 "MOOSEs"를 통해 전송된 패킷들을 포함하

거나, 참조하거나, 대응하는 다른 TX 링크된-리스트(도 10에 미도시됨)에 기초하여 링크 "MOOSEs"와 연관된 패킷(들)을 리플레이할지를 결정하기 위해 동작할 수 있다. 유리하게는, 리플레이 기능을 구현하기 위해 TX 스토리지 434-1 및 TX 링크된-리스트 1020를 이용하는 것은, 노드 400가 소프트웨어 제어 메커니즘들의 도움 없이 제한된 하드웨어 리소스들에 따른 TTP를 이용하여 제2 노드와 통신하도록 한다.

[0150] **하드웨어 링크 타이머**

[0151] 도 11은 소프트웨어의 도움 없이 패킷들을 리플레이하기 위한 타임아웃 체크 메커니즘들을 구현하는 하드웨어 링크 타이머 1100의 예시적인 블록도를 예시한다. 일부 실시예들에서, 하드웨어 링크 타이머 1100는 도 4의 노드 400의 일부일 수 있다. 하드웨어 링크 타이머 1100의 일부 또는 전부는 도 4의 TTP 태그 블록 436 내에 배치될 수 있다. 위에서 언급된 바와 같이, 다수의 타이머들(예를 들어, 하나의 링크에 대한 하나의 타이머)을 이용하여 다수의 링크들을 통한 타임아웃들을 추적하기 위해 일반적으로 소프트웨어가 이용되는 다른 이더넷 프로토콜들(예를 들어, TCP 또는 UDP)과 달리, 하드웨어 링크 타이머 1100는, 노드 400가 어떤 패킷(들)을 어떤 링크(들)를 통해 전송되도록 하고 리플레이할지, 리플레이가 필요한 경우 제한된 하드웨어 리소스들(예를 들어, 가상 및/또는 물리적 주소 공간의 큰 리소스 풀들 및 컴퓨팅 리소스들이 이용 가능하지 않은 경우)에 따라 언제 리플레이할지 결정하도록 할 수 있다. 일부 실시예들에서, 하드웨어 링크 타이머 1100는 TTP에 따라 하나 이상의 다른 노드와 통신하기 위해 노드 400에 의해 이용되는 확립된 링크들(예를 들어, 활성 링크들)에 대한 타이밍 체크를 주기적으로 수행할 수 있다.

[0152] 도 11에 도시된 바와 같이, 하드웨어 링크 타이머 1100는 선입선출(FIFO) 메모리 1104, 타이머 1102 및 논리 회로 1120, 1112, 1114, 1116 및 1118를 포함할 수 있고, 논리 회로 1112, 1114, 1116 및 1118는 패킷들을 리플레이하기 위한 TTP 태그 블록 436의 일부일 수 있다. FIFO 메모리 1104는 활성 링크들 각각과 연관된 타이밍 및 상태 정보를 저장할 수 있다. 하드웨어 링크 타이머 1100는 라운드-로빈 방식으로 FIFO 메모리 1104에 저장된 활성 링크들 각각과 연관된 타이밍 및 상태를 체크할 수 있다. 보다 구체적으로, 하드웨어 링크 타이머 1100는 FIFO 메모리 1104의 제1 엔트리에 저장된 제1 링크와 연관된 타이밍 및 상태 정보를 FIFO 메모리 1104의 N번째 엔트리에 저장된 N번째 링크와 연관된 타이밍 및 상태 정보에 대한 체크로 시작할 수 있고, 그런 다음 FIFO 메모리 1104의 제1 엔트리에 저장된 제1 링크와 연관된 타이밍 및 상태 정보를 다시 체크할 수 있다. 하드웨어 링크 타이머 1100는 다수의 활성 링크들 및/또는 패킷들과 연관된 타이밍 및 상태 정보를 판독하는 시점들을 스케줄링하기 위해 타이머 1102를 이용할 수 있다. 판독된 타이밍 및 상태 정보는 링크와 연관된 패킷들을 리플레이할지, 또는 추가 정보 조회를 통해 패킷들을 폐기 및/또는 삭제할지를 결정하기 위해 이용될 수 있다. 도 4의 노드 400는 도 11에 도시된 것과 유사한 하나 이상의 하드웨어 링크 타이머를 포함할 수 있고, 각각의 하드웨어 링크 타이머는 복수의 링크들과 연관된 타임아웃이 있는지를 결정할 수 있다.

[0153] 일부 실시예들에서, FIFO 메모리 1104는 노드 400와 다른 노드(들) 간에 확립된 하나 이상의 링크와 연관된 타이밍 정보를 저장할 수 있다. 예를 들어, 노드 400는, 노드 400 및 하나 이상의 다른 노드 간에 확립된 M개의 링크들과 연관된 타이밍 정보를 저장하기 위해 FIFO 메모리 1104를 이용하는 하드웨어 링크 타이머 1100를 포함할 수 있고, 여기서 M은 1보다 큰 양의 정수이다. 각각의 타이머가 대응하는 링크의 타이밍 정보를 추적하는 M개의 타이머들을 이용하는 대신, 하드웨어 링크 타이머 1100는 라운드-로빈(예를 들어, 순환) 방식으로 FIFO 메모리 1104에 액세스하는 것을 통해 M개의 링크들 각각에 대한 타이밍 정보를 추적 및/또는 업데이트하기 위해 타이머 1102(예를 들어, 프로그래밍 가능한 시간 주기 동안 한 번 틱하는 하드웨어 클록)를 이용할 수 있다. 구체적으로, 하드웨어 링크 타이머 1100는 타이머 1102가 한 번 틱할 때 라운드-로빈 방식으로 FIFO 메모리 1104의 엔트리들에 한 번에 하나씩 액세스할 수 있고, FIFO 메모리 1104의 각각의 액세스된 엔트리들은 M개의 링크들 중 하나에 대응한다.

[0154] 일부 실시예들에서, 타이머 1102의 각각의 틱의 시간 주기가 달라질 수 있고 수백 마이크로초에서 단일 자릿수 마이크로초 사이일 수 있다. 예를 들어, 타이머 1102의 틱의 시간 주기는 최대 100마이크로초일 수 있고, 최소 1마이크로초일 수도 있다. 또한, 하드웨어 링크 타이머 1100는 FIFO 메모리 1104의 엔트리들에 의해 나타난 링크들의 수(예를 들어, M)에 기초하여 타이머 1102의 틱의 시간 주기를 조정할 수 있다. 예를 들어, M이 증가하면(예를 들어, FIFO 메모리 1104의 엔트리들에 의해 나타나는 링크들이 더 많아짐), 타이머 1102의 틱의 시간 주기는 감소할 수 있고, M이 감소하면(예를 들어, FIFO 메모리 1104의 엔트리들에 의해 나타나는 링크들이 더 적어짐), 타이머 1102의 틱의 시간 주기는 증가할 수 있다. 따라서, FIFO 메모리 1104의 엔트리들에 의해 나타난 링크들의 개수에 대한 타이머 1102의 틱의 시간 주기가 비례하지 않게 변경되는 경우, 링크의 상태 및/또는 타이밍 정보가 체크되는 시간 간격은 변경되지 않을 수 있다.

[0155] 일부 실시예들에서, M개의 링크들 중 하나와 연관된 타이밍 및/또는 상태 정보는 링크가 얼마나 오래, 전송된 패킷들을 수신한 것의 확인응답을 수신하지 못했는지를 나타낼 수 있다. 노드 400가 링크를 통해 제2 노드로 N개의 패킷들을 전송했다고 가정하면, FIFO 메모리 1104의 하나의 엔트리는, 타이머 1102의 틱의 특정한 시간 주기에 따라 라운드-로빈 방식을 통해 액세스될 때, N개의 패킷들 중 임의의 것을 수신한 것의 확인응답이 미리 결정된 기간(예를 들어, 20마이크로초, 50마이크로초, 100마이크로초, 200마이크로초, 300마이크로초, 400마이크로초, 500마이크로초 및/또는 그 사이의 모든 기간) 동안 수신되지 않았다는 것을 나타내는 타이밍 및/또는 상태 정보를 저장할 수 있다. 하드웨어 링크 타이머 1100는 FIFO 메모리 1104의 엔트리에 접근하면 논리 회로 1120, 1112, 1114, 1116 및 1118을 이용하여 엔트리에 저장된 타이밍 및/또는 상태 정보를 확인하고 노드 400의 로컬 스토리지(예를 들어, TX 스토리지 434-1 또는 기타 로컬 스토리지)에 저장되어 있을 수 있는 N개의 패킷을 조회하여 N개의 패킷을 리플레이할 수 있다.

[0156] 대안적으로, M개 링크들 중 하나와 연관된 타이밍 및/또는 상태 정보가, 링크가 닫힐 수 있음을 나타내기 위해 FIFO 메모리 1104의 하나의 엔트리에 저장될 수 있다(예를 들어, 제1 노드에 의해 전송된 모든 패킷들이 제2 노드에 의해 수신됨). FIFO 메모리 1104의 엔트리에 액세스 하면, 하드웨어 링크 타이머 1100는, FIFO 메모리 1104의 엔트리에 저장된 타이밍 및/또는 상태 정보가, 링크를 닫힐 수 있음을 나타내기 때문에 패킷들을 삭제하고 노드 400의 로컬 스토리지(예를 들어, TX 스토리지 434-1)에 여전히 저장되어 있을 수 있는 패킷들을 조회하기 위해 및 엔트리에 저장된 타이밍 및/또는 상태 정보를 체크하기 위해 논리 회로 1120, 1112, 1114, 1116 및 1118를 이용할 수 있다. 유리하게는, 조정 가능한 기간(adjustable period)들에 따라 틱하는 단일 타이머(예를 들어, 타이머 1102) 및 다수의 링크들의 타이밍 및/또는 상태 정보를 저장하는 FIFO 메모리 1104를 이용하는 것에 의해, 노드 400는, 제한된 컴퓨팅 및 스토리지 리소스들에 따라 동작하도록 활성 링크들에 의한 이용을 위해 비활성 링크들(예를 들어 닫힌 링크들)에 의해 차지된 하드웨어 리소스들을 해제하고 낮은 레이턴시를 달성하기 위해, 적절한 타이밍에 패킷들을 리플레이할 수 있다.

[0157] 도 11에 도시된 바와 같이, 논리 회로 1120, 1112, 1114, 1116 및 1118는 도 10에 예시된 논리 회로 1012, 1014, 1016 및 1018와 유사하게 상이한 파이프라인 스테이지들에서 동작할 수 있다. 도 11에 도시된 바와 같이, 논리 회로 1120, 1112, 1114, 1116 및 1118는, 하나 이상의 링크를 통해 전송된 패킷들이 언제 리플레이되어야 하는지 또는 TX 스토리지 434-1와 같은 로컬 스토리지로부터 언제 폐기/삭제될 수 있는지 또는 하나 이상의 링크가 닫힐 수 있는지를 결정하기 위해, 타이머 1102 및 FIFO 메모리 1104와 함께 동작할 수 있다. 도 11에 도시된 바와 같이, 논리 회로 1120, 1112, 1114, 1116 및 1118는 하드웨어 링크 타이머 1100가 동작하는 클록에 따라 각각의 파이프라인 스테이지들에서 동작할 수 있다. 구체적으로, 논리 회로 1120 및 1112는 초기 파이프라인 스테이지 ("Q0"로 레이블링됨)에서 동작할 수 있고, 논리 회로 1114는 제1 파이프라인 스테이지 ("Q1"로 레이블링됨)에서 동작할 수 있고, 논리 회로 1116는 제2 파이프라인 스테이지 ("Q2"로 레이블링됨)에서 동작할 수 있고, 논리 회로 1118는 제3 파이프라인 스테이지 ("Q3"로 레이블링됨)에서 동작할 수 있다.

[0158] 동작에서, 초기 파이프라인 스테이지 Q0에서, 논리 회로 1120는 논리 회로 1112에 대한 타이밍 및 상태 정보 조회(예를 들어, TIMER 링크 조회(TIMER Link Lookup))에 이용될 타이밍 및 상태 정보를 선택할 수 있다. 도 11에 도시된 바와 같이, 타이밍 및 상태 정보는 FIFO 메모리 1104로부터의 엔트리(예를 들어, 다른 모든 엔트리들보다 일찍 FIFO 메모리 1104에 들어오는 가장 오래된 엔트리) 또는 다른 소스들(예를 들어, 대안적인 우선 링크 조회 정보(alternative priority link lookup information)로부터 나올 수 있다. 도 11에 도시된 바와 같이, 초기 파이프라인 스테이지 Q0에서 FIFO 메모리 1104의 "링크 A"와 연관된 타이밍 및 상태 정보는 "TX Traffic" 또는 "RX Traffic" 대신 "TIMER Link Lookup"를 선택하는 제어 신호(예를 들어, "Pick")에 기초하여 논리 회로 1112에 의해 선택된다. "TX Traffic"은 노드 400에 의해 확립된 링크(예를 들어, "링크 B")를 통해 전송된 패킷들에 대응할 수 있고, "RX Traffic"은 노드 400에 의해 확립된 다른 링크(예를 들어, "링크 D")를 통해 수신된 패킷에 대응할 수 있다.

[0159] 제1 파이프라인 스테이지 Q1에서 논리 회로 1114는 초기 파이프라인 스테이지 Q0로부터 수신한 타이밍 및 상태 정보에 기초하여 어떤 링크가 쿼리되는지를 결정한다. 도 11에 도시된 바와 같이, 논리 회로 1114는 "링크 A"가 리플레이되어야 하는지 또는 닫힐 수 있는지를 나중에 결정하기 위해 쿼리되고 있음을 결정한다. 그런 다음, 제2 파이프라인 스테이지 Q2에서, 논리 회로 1116는 FIFO 메모리 1104로부터 액세스된 "링크 A"와 연관된 타이밍 및 상태 정보에 기초하여 "링크 A"가 닫힐 수 있는지를 결정한다. "링크 A"와 연관된 타이밍 및 상태 정보가, "링크 A"가 닫힐 수 있음을 보여주는 경우, 논리 회로 1116는 "링크 A"와 연관된 패킷들이 로컬 스토리지(예를 들어, TX 스토리지 434-1)로부터 폐기/삭제되도록 트리거할 수 있다. "링크 A"와 연관된 타이밍 및 상태 정보가, "링크 A"가 여전히 활성/열려 있음을 보여주는 경우, 하드웨어 링크 타이머 1100의 동작은 제3 파이프

라인 스테이지 Q3로 진행되고, 논리 회로 1118는 "링크 A"를 통해 전송된 패킷들을 리플레이할지 또는 "링크 A"와 연관된 타이밍 및 상태 정보를 어떻게 업데이트할지를 결정한다.

[0160] 제3 파이프라인 스테이지 Q3에서 논리 회로 1118는 FIFO 메모리 110로부터 액세스되는 "링크 A"와 연관된 상태 및 타이밍 정보에 기초하여 "링크 A"와 연관된 적어도 일부 패킷들을 리플레이하도록 결정할 수 있다. 예를 들어, "링크 A"와 연관된 상태 및 타이밍 정보는, "TIMER BIT"가 (예를 들어, 논리 1로) 설정되는 경우 "링크 A"와 연관된 패킷들 중 적어도 하나의 패킷을 수신한 것의 확인응답이 패킷들을 리플레이하기 위한 임계 기간 동안 노드 400에 의해 수신되지 않았음을 나타낼 수 있는 TIMER BIT"를 포함할 수 있다. 일부 실시예들에서, 임계 기간은 조정 가능할 수 있고, 20마이크로초, 50마이크로초, 100마이크로초, 200마이크로초, 300마이크로초, 400마이크로초, 500마이크로초 및/또는 그 사이의 임의의 적합한 기간이 될 수 있다. 임계 기간은 20마이크로초에서부터 500마이크로초까지 가능하다. 일부 실시예들에서, "링크 A"(및/또는 다른 링크들)와 연관된 "TIMER BIT"는 "링크 A"가 FIFO 메모리 1104로부터 쿼리된 횟수 및 타이머 1102의 시간 주기에 기초하여 설정될 수 있다.

[0161] "TIMER BIT"가 어설트(assert)되면, 논리 회로 1118는 "링크 A"와 연관된 패킷들이 리플레이되도록 할 수 있다. 어설트되는 "TIMER BIT"는 하나 이상의 패킷과 연관된 타임아웃이 발생했음을 나타낼 수 있다(예를 들어, 확인응답 또는 비-확인응답을 수신하는 것 없이 임계 기간에 도달함). 또한, 논리 회로 1118는 "링크 A"의 리플레이에 응답하여 FIFO 메모리 1104에 저장된 "링크 A"와 연관된 타이밍 및 상태 정보를 업데이트할 수 있다. 예를 들어, 논리 회로 1118는 "TIMER BIT"를 초기화할(clear) 수 있다(예를 들어, "TIMER BIT"를 논리 1에서 논리 0으로 설정). 반면에, "링크 A"와 연관된 상태 및 타이밍 정보가 "링크 A"에서 하나 이상의 패킷을 리플레이하지 않도록 나타내는 경우(예를 들어, 도 11에서 논리 0에 대응하는, "TIMER BIT"가 어설트되지 않음), 논리 회로 1118는 "링크 A"가 리플레이되도록 하지 않을 수 있다. 이러한 상황에서, 논리 회로 1118는 "링크 A"와 연관된 타이밍 및 상태 정보가 다음에 쿼리될 경우 "링크 A"가 리플레이되어야 함을 나타내면, "TIMER BIT"를 논리 1로 더 설정할 수 있다.

[0162] **리플레이 및 링크 타이밍의 예시적인 방법**

[0163] 이제 도 12를 살펴보면, 도 3b의 장치 A 또는 노드 400과 같은 노드로부터 전송된 패킷들을 리플레이하기 위한 예시적인 패킷 리플레이 절차(packet replay procedure) 1200가 설명되어 있다. 패킷 리플레이 절차 1200는 예를 들어, 절차 1200은 구성요소(component)들 또는 TTP 태그 블록 436에 의해 구현될 수 있다. 절차 1200는 블록 1202에서 시작하고, TTP 태그 블록 436은 이더넷 프로토콜을 이용하여 노드 400에서부터 제2 노드로 제1 링크를 통해 전송되는 패킷들을 포함하는 링크된-리스트를 저장할 수 있다. 예를 들어, 링크된-리스트는, 제2 노드에 전송하기 위한 패킷들 1022, 1024 및 1026의 순서를 유지하기 위해 패킷들 1022, 1024 및 1026을 포함하거나 참조하는 TX 링크된-리스트 1020일 수 있다.

[0164] 블록 1204에서, TTP 태그 블록 436은 (a) 제2 노드로부터 제1 패킷의 비-확인응답의 수신 또는 (b) 제1 패킷과 연관된 타임아웃 중 적어도 하나에 응답하여 패킷들의 제1 패킷을 리플레이하도록 결정할 수 있다. TTP 태그 블록 436은, (a) 제2 노드로부터 패킷 1024의 비-확인응답을 수신한 경우 또는 (b) 패킷 1024의 확인응답이 임계 시간 이상 수신되지 않았음을 나타내는 패킷 1024과 연관된 타임아웃에 응답하여 패킷 1024를 리플레이하도록 결정할 수 있다.

[0165] 블록 1206에서, TTP 태그 블록 436은 제2 노드로부터 제2 패킷의 확인응답의 수신에 응답하여 패킷들 중 제2 패킷을 폐기할 수 있다. 예를 들어, TTP 태그 블록 436은 제2 노드로부터 패킷 1022의 확인응답의 수신에 응답하여 패킷 1022을 폐기할 수 있다.

[0166] 도 13은, 도 3b의 장치 A 또는 노드 400과 같은, 노드와 연관된 하나 이상의 링크를 리플레이할지를 결정하기 위한 예시적인 링크 타임아웃 절차 1300를 예시한다. 링크 타임아웃 절차 1300는 예를 들어, 도 11의 하드웨어 링크 타이머 1100 또는 노드 400에 의해 구현될 수 있다. 절차 1300는 블록 1302에서 시작하고, 하드웨어 링크 타이머 1100 또는 노드 400는 FIFO 메모리의 복수의 링크들과 연관된 타이밍 및 상태 정보를 저장하고, 노드 400는 이더넷 프로토콜을 이용하여 하나 이상의 다른 노드로 복수의 링크들을 통해 패킷들을 전송한다. 예를 들어, 하드웨어 링크 타이머 1100는 FIFO 메모리 1104의 복수의 링크들과 연관된 타이밍 및 상태 정보를 저장할 수 있다.

[0167] 블록 1304에서, 하드웨어 링크 타이머 1100 또는 노드 400는 하드웨어 링크 타이머 1100 또는 노드 400 내에 배치된 하드웨어 타이머의 각각의 틱들에 기초하여 FIFO 메모리의 엔트리들에 액세스할 수 있다. 예를 들어, 하드웨어 링크 타이머 1100는 타이머 1102의 각각의 틱들에 기초하여 FIFO 메모리 1104의 엔트리들에 액세스할 수

있다.

[0168] 블록 1306에서, 하드웨어 링크 타이머 1100 또는 노드 400는 복수의 링크들의 제1 링크와 연관된 타이밍 및 상태 정보에 기초하여, 제1 링크와 연관된 적어도 하나의 패킷을 리플레이하도록 결정할 수 있다. 예를 들어, 하드웨어 링크 타이머 1100는 "링크 A"와 연관된 타이밍 및 상태 정보에 기초하여 "링크 A"와 연관되거나 "링크 A"를 통해 전송된 적어도 하나의 패킷을 리플레이하도록 결정할 수 있다.

[0169] **결론**

[0170] 전술한 개시는 본 개시를 개시된 정확한 형태들 또는 특정한 이용 분야들로 제한하려는 의도가 아니다. 이와 같이, 본 개시에 명시적으로 설명되거나 암시되든 간에, 본 개시에 대한 다양한 대체 실시예들 및/또는 수정(modification)들이 본 개시에 비추어 가능하다는 것이 고려된다. 본 개시의 실시예들을 설명하였으므로, 당업자는 본 개시의 범위를 벗어나지 않고 형태 및 세부 사항에 변경이 이루어질 수 있음을 인식할 것이다. 따라서 본 개시는 청구범위에 의해서만 제한된다.

[0171] 본 명세서에 설명된 임의의 특정한 실시예에 따라 반드시 모든 목적들 또는 이점들이 달성되지 않을 수 있음이 이해되어야 한다. 따라서, 예를 들어 당업자는 일부 예들이 본 명세서에 교시되거나 제안될 수 있는 다른 목적들 또는 이점들을 반드시 달성하지 않고도 본 명세서에 교시된 바와 같은 하나의 이점 또는 이점들의 그룹을 달성하거나 최적화하는 방식으로 구현되거나 수행될 수 있다는 것을 인식할 것이다.

[0172] 본 명세서에서 설명된 모든 프로세스들은 컴퓨터들 또는 프로세서들을 포함하는 컴퓨팅 시스템에서 실행되는 소프트웨어 코드 모듈들에 구현되고, 이를 통해 완전히 자동화될 수 있다. 코드 모듈들은 모든 타입의 비-일시적 컴퓨터-판독 가능한 매체 또는 기타 컴퓨터 스토리지 장치에 저장될 수 있다. 일부 또는 모든 방법들은 특수한 컴퓨터 하드웨어에 구현될 수 있다.

[0173] 본 명세서에서 설명된 것들 외에도 많은 다른 변형들이 이 개시로부터 명백해질 것이다. 예를 들어, 예시에 따라, 본 명세서에서 설명된 임의의 알고리즘들의 일부 행위들, 이벤트들 또는 기능들은 다양한 순서로 수행될 수 있고, 추가, 병합되거나, 완전히 생략될 수 있다(예를 들어, 설명된 모든 행위들 또는 이벤트들이 알고리즘들의 실행을 위해 필요한 것은 아니다). 또한, 일부 예들에서, 행위들 또는 이벤트들이 순차적으로가 아니라, 예를 들어 멀티-스레드 처리, 인터럽트 처리, 다수의 프로세서들 또는 프로세서 코어들, 또는 다른 병렬 아키텍처들을 통해 동시에 수행될 수 있다. 또한, 상이한 작업들 또는 프로세스들은 함께 기능할 수 있는 상이한 기계(machine)들 및/또는 컴퓨팅 시스템들에 의해 수행될 수 있다.

[0174] 본 명세서에 개시된 예들과 관련하여 설명된 다양한 예시적인 논리 블록들 및 모듈들은 처리 장치 또는 프로세서, 디지털 신호 프로세서(DSP), 애플리케이션 특정 집적 회로(ASIC), 필드 프로그래밍 가능한 게이트 어레이(FPGA) 또는 기타 프로그래밍 가능한 논리 장치, 개별 게이트 또는 트랜지스터 논리, 개별 하드웨어 구성요소들 또는 본 명세서에서 설명된 기능들을 수행하도록 설계된 이들의 임의의 조합과 같은 기계에 의해 구현되거나 수행될 수 있다. 프로세서는 마이크로프로세서일 수 있지만, 대안적으로 프로세서는 콘트롤러, 마이크로콘트롤러 또는 상태 머신, 이들의 조합 또는 이와 유사한 것이 될 수 있다. 프로세서는 컴퓨터-실행 가능한 인스트럭션들을 처리하기 위한 전기 회로를 포함할 수 있다. 일부 예들에서, 프로세서는 컴퓨터-실행 가능한 인스트럭션들을 처리하지 않고 논리 연산들을 수행하는 FPGA 또는 다른 프로그래밍 가능한 장치를 포함한다. 프로세서는 또한, 예를 들어, DSP와 마이크로프로세서의 조합, 복수의 마이크로프로세서들, DSP 코어와 결합된 마이크로프로세서들 또는 이와 유사한 다른 구성의 컴퓨팅 장치들의 조합으로서 구현될 수 있다. 본 명세서에서 주로 디지털 기술과 관련하여 설명되었지만, 프로세서는 주로 아날로그 구성요소들을 포함할 수도 있다. 컴퓨팅 환경은 마이크로프로세서에 기초한 컴퓨터 시스템, 메인프레임 컴퓨터, 디지털 신호 프로세서, 휴대용 컴퓨팅 장치, 장치 콘트롤러 또는 기기 내 계산 엔진을 포함하되 이에 국한되지 않는 임의의 타입의 컴퓨터 시스템을 포함할 수 있다.

[0175] 본 명세서에 개시된 실시예들과 관련하여 설명된 방법, 프로세스, 루틴 또는 알고리즘의 요소들은 하드웨어에 직접 구현되거나, 프로세서 장치에 의해 실행되는 소프트웨어 모듈에 구현되거나, 두 가지의 조합으로 구현될 수 있다. 소프트웨어 모듈은 RAM 메모리, 플래시 메모리, ROM 메모리, EPROM 메모리, EEPROM 메모리, 레지스터들, 하드 디스크, 이동식 디스크, CD-ROM 또는 임의의 기타 비-일시적 컴퓨터-판독 가능한 스토리지 매체에 저장될 수 있다. 예시적인 스토리지 매체는 프로세서 장치에 결합될 수 있어 프로세서 장치는 스토리지 매체로부터 정보를 판독하고, 스토리지 매체에 정보를 쓸 수 있다. 또는, 스토리지 매체가 프로세서 장치에 포함될 수 있다. 프로세서 장치 및 스토리지 매체는 ASIC에 저장될 수 있다. ASIC은 사용자 단말에 상주할 수 있다. 또는,

프로세서 장치 및 스토리지 매체는 사용자 단말의 개별 구성요소들로서 상주할 수 있다.

[0176] 본 명세서에 기술되거나 본 개시의 도면들에 도시된 프로세스들은 이벤트에 응답하여 시작될 수 있고, 예를 들어 미리 결정된 스케줄 또는 동적으로 결정된 스케줄에 따라, 사용자 또는 시스템 관리자가 요청할 때 필요에 따라, 또는 다른 이벤트에 응답하여 시작될 수 있다. 이러한 프로세스가 시작되면, 하나 이상의 비-일시적 컴퓨터 판독 가능한 매체(예를 들어, 하드 드라이브, 플래시 메모리, 이동식 매체 등)에 저장된 실행 가능한 프로그램 인스트럭션들의 세트가 서버 또는 다른 컴퓨팅 장치의 메모리(예를 들어, RAM)에 로드될 수 있다. 실행 가능한 인스트럭션들은 컴퓨팅 장치의 하드웨어 기반 컴퓨터 프로세서에 의해 실행될 수 있다. 일부 실시예들에서, 이러한 프로세스 또는 그 일부들은 다수의 컴퓨팅 장치들 및/또는 다수의 프로세서들에서 직렬 또는 병렬로 구현될 수 있다.

[0177] "할 수 있다(can)", "할 수 있다(could)", "일 수 있다(might)" 또는 "일 수 있다(may)"와 같은 조건부 표현(conditional language)은 특별히 달리 명시되지 않는 한, 일반적으로 이용되는 맥락 내에서 일부 예들이 일부 특징들, 요소들 및/또는 단계들을 포함하고 다른 예들을 포함하지 않는다는 것을 전달하기 위해 이해된다. 따라서, 이러한 조건부 표현은, 일반적으로 특징들, 요소들 및/또는 단계들이 어떤 방식으로든 예시들을 위한 것이거나 예시들이 사용자 입력 또는 프롬프트의 유무에 관계없이 이러한 기능들, 요소들 및/또는 단계들이 반드시 포함되어야 하는지를 결정하기 위한 논리를 반드시 포함한다는 것을 암시하는 것은 아니다.

[0178] "X, Y 또는 Z 중 적어도 하나"와 같은 구문과 같은 분리형 표현(disjunctive language)은 특별히 달리 명시되지 않는 한, 일반적으로 항목, 용어 등이 X, Y 또는 Z이거나 이들의 조합(예를 들어, X, Y 및/또는 Z)일 수 있음을 나타내기 위해 이용되는 맥락과 함께 이해된다. 따라서, 이러한 분리형 표현은 일반적으로 일부 예들이 각각 X 중 하나 이상, Y 중 하나 이상 또는 Z 중 하나 이상이 존재해야 한다는 것을 의미하지 않고, 그렇게 암시해서도 안 된다.

[0179] 본 명세서에 설명되어 있거나 첨부된 도면들에 나타난 흐름도들의 모든 프로세스 설명들, 요소들 또는 블록들은 프로세스에서 특정한 논리적 기능들 또는 요소들을 구현하기 위한 실행 가능한 인스트럭션들을 포함하는 모듈, 들 세그먼트들 또는 코드의 일부들을 나타낼 가능성이 있다고 이해되어야 한다. 본 명세서에 설명된 예시들의 범위 내에 대체 예시들이 포함되는데, 그러한 대체 예시들에서는 요소들 또는 기능들이 삭제되거나, 도시되거나 논의된 순서와 다르게 실행될 수 있고, 기능에 따라 실질적으로 동시에 또는 역순으로 실행될 수도 있다(당업자가 이해할 수 있는 바임).

[0180] 전술한 예들에 대해 많은 변형들 및 수정들이 이루어질 수 있고, 그 요소들은 다른 허용 가능한 예들 중 하나로 이해되어야 한다는 것이 강조되어야 한다. 이러한 수정들 및 변형들은 본 개시의 범위 내에 포함되도록 의도된다.

[0181] 본 명세서에 설명되어 있거나 첨부된 도면들에 나타난 흐름도들의 모든 프로세스 설명들, 요소들 또는 블록들은 프로세스에서 특정한 논리적 기능들 또는 요소들을 구현하기 위한 실행 가능한 인스트럭션들을 포함하는 모듈들, 세그먼트들 또는 코드의 일부들을 나타낼 가능성이 있다고 이해되어야 한다. 본 명세서에서 설명된 예들의 범위 내에는 대체 구현들이 포함되고, 그러한 대체 구현들에서는 요소들 또는 기능들이 삭제되거나, 도시되거나 논의된 순서와 다르게 실행될 수 있고, 기능에 따라 실질적으로 동시에 또는 역순으로 실행될 수도 있고, 이는 당업자에 의해 이해될 것이다.

[0182] 달리 명확하게 언급되지 않는 한, "하나(a 또는 an)" 또는 "일"(a 또는 an)"과 같은 관사들은 일반적으로 하나 이상의 설명된 항목을 포함하는 것으로 해석되어야 한다. 따라서 "~ 하는 일 장치(a device configured to)"와 같은 구문들은 하나 이상의 언급된 장치를 포함하는 것으로 의도된다. 이러한 하나 이상의 언급된 장치는 또한 언급된 항목들을 수행하도록 집합적으로 구성될 수도 있다. 예를 들어, "항목 A, 항목 B 및 항목 C를 수행하도록 구성된 프로세서"에는 항목 A를 수행하는 제1 프로세서가 항목 B 및 항목 C를 수행하는 제2 프로세서와 함께 동작하는 것이 포함될 수 있다.

도면

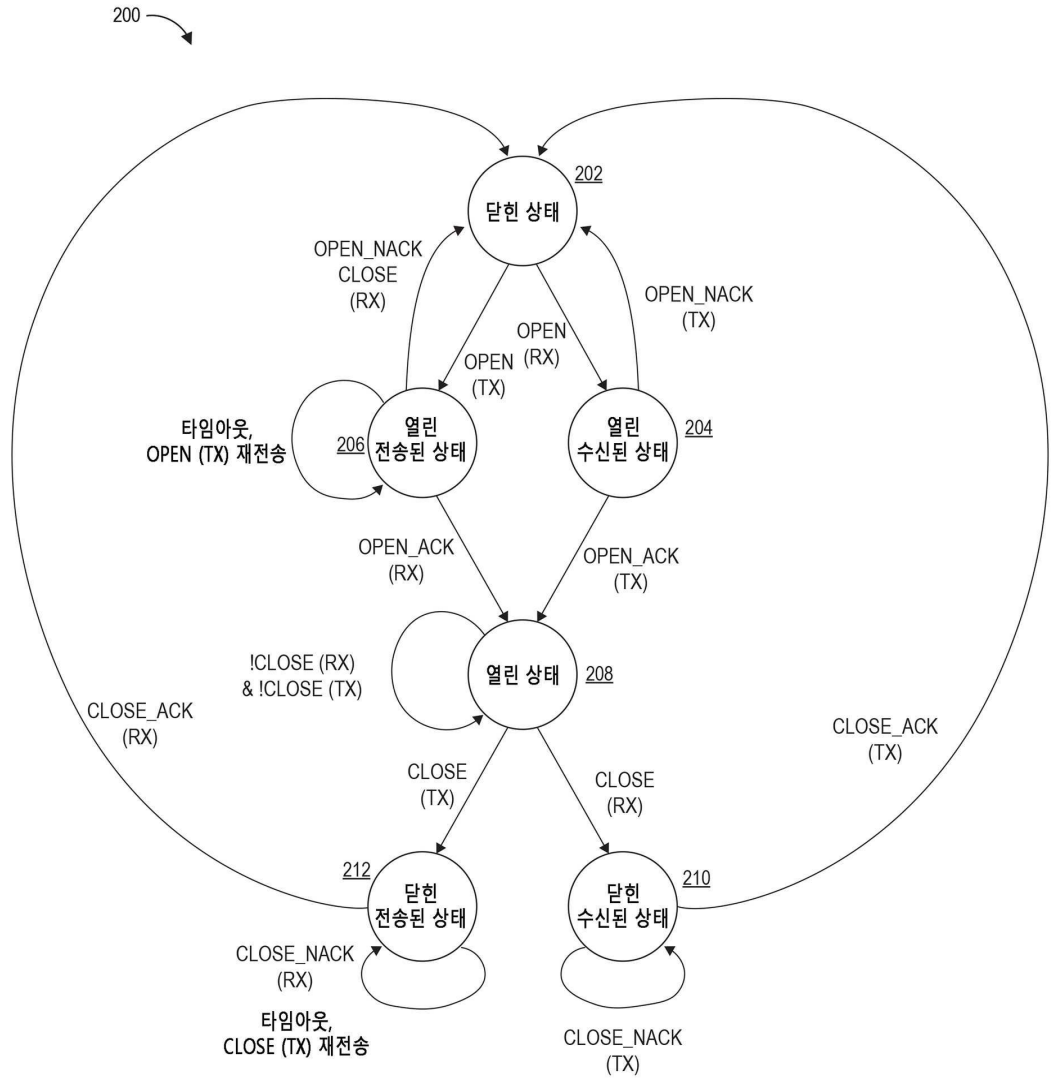
도면1a

OSI 계층	예시적인 프로토콜들 (TCP/IP)	TCP/IP 구현
계층 7 애플리케이션	HTTP, Telnet, FTP	소프트웨어
계층 6 표현	JPEG, PNG, MPEG	
계층 5 세션	NFS, SQL	
계층 4 전송	TCP, UDP	
계층 3 네트워크	IPv4/IPv6	
계층 2 데이터 링크	이더넷 프레임들, MAC 주소들, VLAN	하드웨어
계층 1 물리	데이터 인코딩, 물리적 사양들	

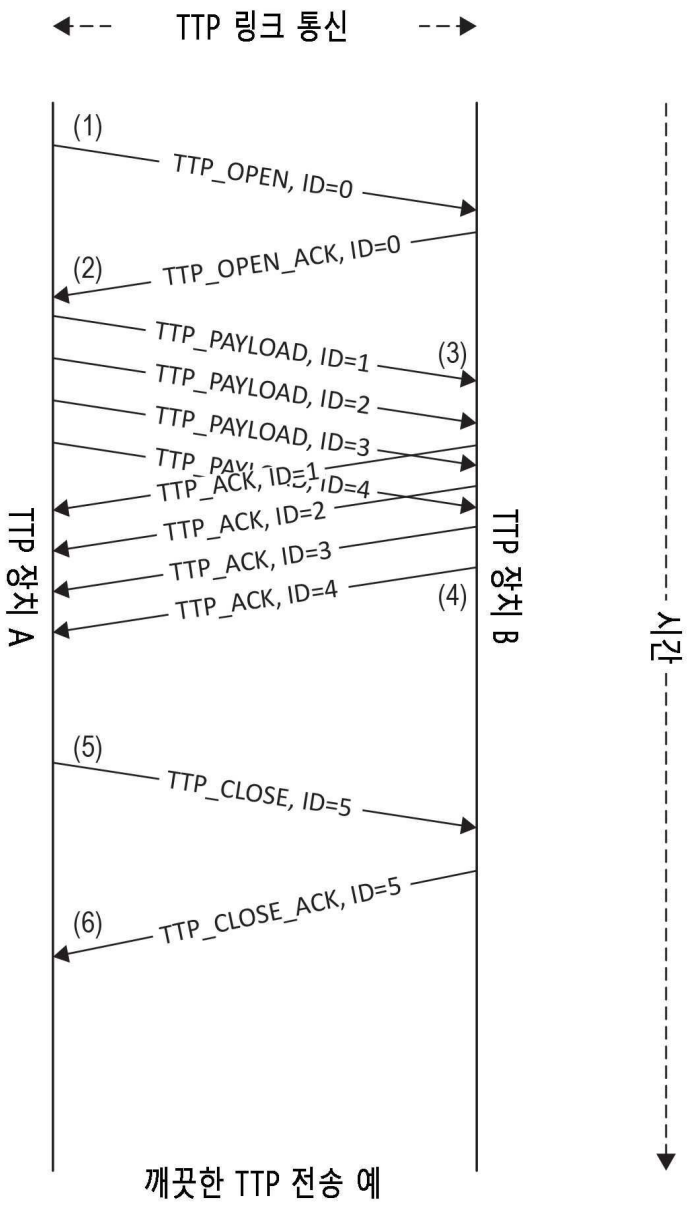
도면1b

OSI 계층	예시적인 프로토콜들	TCP/IP 구현
계층 7 애플리케이션	파이토치	소프트웨어
계층 6 표현	FFMPEG, HEVC, YUV	
계층 5 세션	RDMA	
계층 4 전송	TTP	하드웨어
계층 3 (선택적) 네트워크	IPv4/IPv6 (선택적)	
계층 2 데이터 링크	이더넷 프레임들, MAC 주소들, VLAN	
계층 1 물리	데이터 인코딩, 물리적 사양들	

도면2

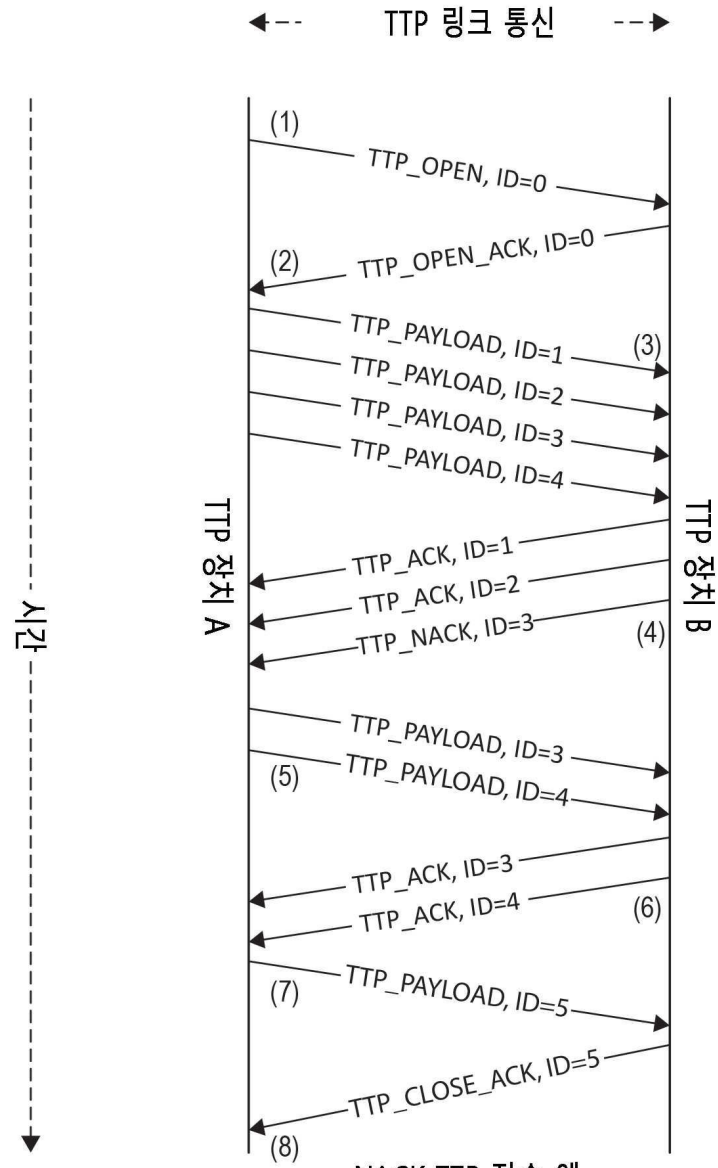

200
202
닫힌 상태
OPEN_NACK
CLOSE
(RX)
OPEN_NACK
(TX)
OPEN
(RX)
OPEN
(TX)
타임아웃,
OPEN (TX) 재전송
206
열린
전송된 상태
열린
수신된 상태
204
OPEN_ACK
(RX)
OPEN_ACK
(TX)
!CLOSE (RX)
& !CLOSE (TX)
열린 상태
208
CLOSE_ACK
(RX)
CLOSE_ACK
(TX)
CLOSE
(TX)
CLOSE
(RX)
212
닫힌
전송된 상태
닫힌
수신된 상태
210
CLOSE_NACK
(RX)
타임아웃,
CLOSE (TX) 재전송
CLOSE_NACK
(TX)

도면3a



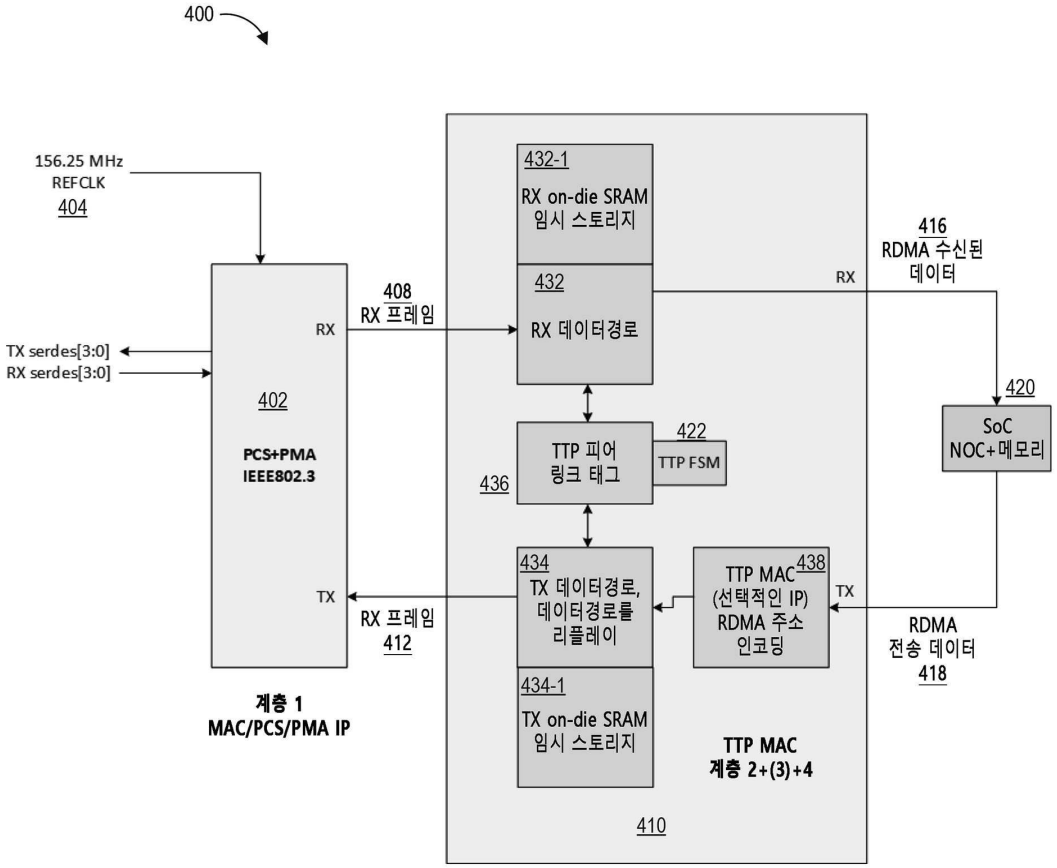
깨끗한 TTP 전송 예

도면3b

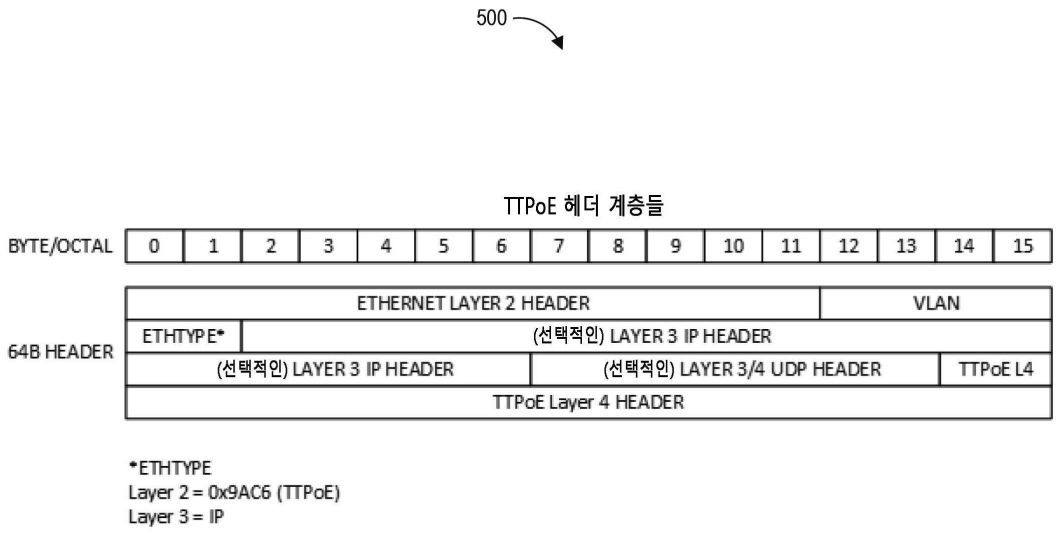


NACK TTP 전송 예.
TTP_PAYLOAD, ID=3은 손실되거나 순서대로가 아님

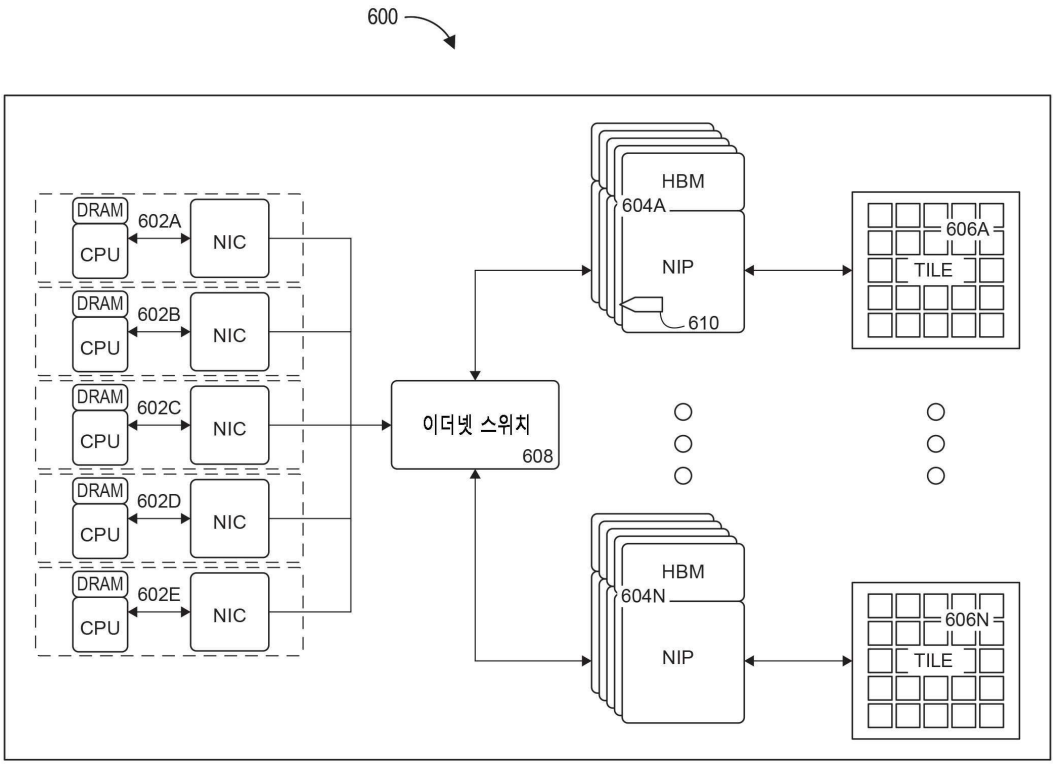
도면4


400
156.25 MHz
REFCLK
404
TX serdes[3:0]
RX serdes[3:0]
402
PCS+PMA
IEEE802.3
RX
TX
408
RX 프레임
RX 프레임
412
계층 1
MAC/PCS/PMA IP
432-1
RX on-die SRAM
임시 스토리지
432
RX 데이터경로
TTP 피어
링크 태그
436
422
TTP FSM
434
TX 데이터경로,
데이터경로를
리플레이
434-1
TX on-die SRAM
임시 스토리지
TTP MAC
(선택적인 IP)
RDMA 주소
인코딩
438
TTP MAC
계층 2+(3)+4
410
416
RDMA 수신된
데이터
RX
TX
420
SoC
NOC+메모리
RDMA
전송 데이터
418

도면5


500
TTPoE 헤더 계층들
BYTE/OCTAL
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
64B HEADER
ETHERNET LAYER 2 HEADER
VLAN
ETHTYPE*
(선택적인) LAYER 3 IP HEADER
(선택적인) LAYER 3 IP HEADER
(선택적인) LAYER 3/4 UDP HEADER
TTPoE L4
TTPoE Layer 4 HEADER
*ETHTYPE
Layer 2 = 0x9AC6 (TTPoE)
Layer 3 = IP

도면6


600
DRAM
CPU
602A
NIC
602B
602C
602D
602E
이더넷 스위치
608
HBM
604A
NIP
610
606A
TILE
604N
606N

도면7a

연산 코드들:

1. OPENING A LINK IN TPP

TTP_OPEN

설명: 두 MAC 엔드포인트들 간에 새로운 TTP 링크를 열도록 요청

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_OPEN	VC[1:0], 6'h00	OPEN pkt ID (1번째 데이터는 ID+1)	N/A

TTP_OPEN_ACK

설명: 링크가 OPEN으로 간주된다는 TTP_OPEN 요청에 대한 확인 응답; 로컬 시작 ID를 설정

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_OPEN_ACK	VC[1:0], 6'h01	1번째 로컬 데이터 ID	TTP_OPEN TxID

TTP_OPEN_NACK

설명: TTP_OPEN 요청에 대한 비-확인 응답. 링크는 여전히 CLOSED로 간주됨

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_OPEN_ACK	VC[1:0], 6'h02	N/A	TTP_OPEN TxID

2. CLOSING A LINK IN TPP

TTP_CLOSE

설명: 두 MAC 엔드포인트들 간에 새로운 TTP 링크를 닫도록 요청; 로컬 최종 ID를 설정

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_CLOSE	VC[1:0], 6'h03	마지막 TX DATA ID + 1	Last seen RX DATA (speculative)

TTP_CLOSE_ACK

설명: TTP_CLOSE 요청에 대한 확인 응답. 링크는 이제 CLOSED로 간주됨

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_OPEN_ACK	VC[1:0], 6'h04	N/A	TTP_CLOSE TxID

도면7b

TTP_CLOSE_NACK
설명: TTP_CLOSE 요청에 대한 비-확인 응답. 링크는 여전히 닫혀있지만,
미해결 패킷들은 아직 해결되지 않음; 로컬 닫음 ID를 설정

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_CLOSE_NACK	VC[1:0], 6'h05	(설정) 마지막 TX ID	마지막 RX ID

1

3. CONTROL, DATA MOVEMENT, AND RESPONSES IN TTP
TTP_PAYLOAD
설명: TTP 헤더 패킷과 함께 페이로드로서 데이터 패킷들 또는 컴퓨팅 시스템 제어를 전송;
DATA 패킷들은 VC 2에서 전송되고 1비트~16비트(64B-1KB)임; REQ_HI 패킷들은 TTP 헤더 비트 내의 VC 1에서 전송됨;
REQ_LO 패킷들은 TTP 헤더 비트 내의 VC 0에서 전송됨

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]	LEN[15:6]
TTP_PAYLOAD	VC[1:0], 6'h06	PAYLOAD TxID	N/A	VC2: #data beats + 1 VC1: 1 VC0: 1

TTP_ACK
설명: 패킷이 수신되고 올바르게 처리되었다는 TTP_PAYLOAD에 대한 확인 응답;
로컬 엔드포인트는 ID에서 더 이상 리플레이들이 발생하지 않을 수 있으므로 리소스들을 할당 해제함

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_ACK	VC[1:0], 6'h07	N/A	TTP_PAYLOAD TxID

TTP_NACK
설명: 패킷 전송(및 임의의 후속 패킷들)이 실패했다는 TTP_PAYLOAD 요청에 대한 비-확인 응답;
RxID는 초기 실패의 ID를 나타냄; 실패한 ID가 수신될 때까지 링크에서 더 젊은 패킷들을 무시

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_NACK	VC[1:0], 6'h08	N/A	실패한 TTP_PAYLOAD TxID

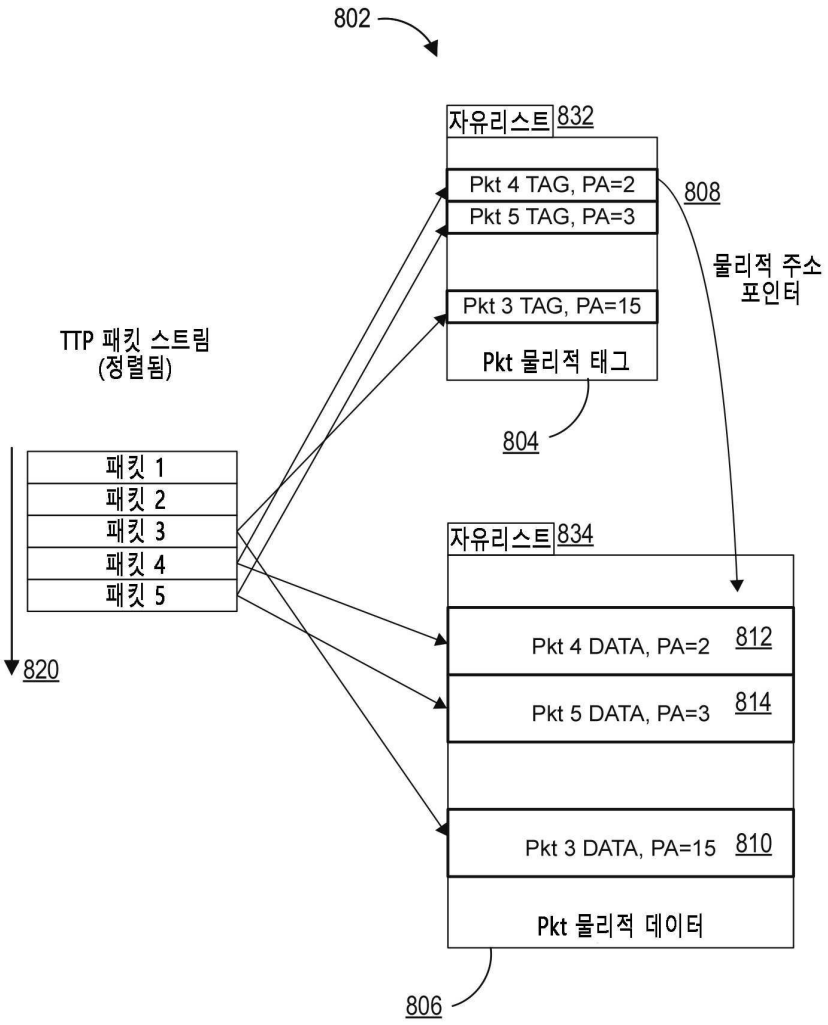
TTP_NACK_FULL
설명: 로컬 수신기의 리소스들이 가득 찼다는 TTP_PAYLOAD 요청에 대한 비-확인 응답; TTP_NACK에 대한 별칭

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_NACK_FULL	VC[1:0], 6'h09	N/A	N/A

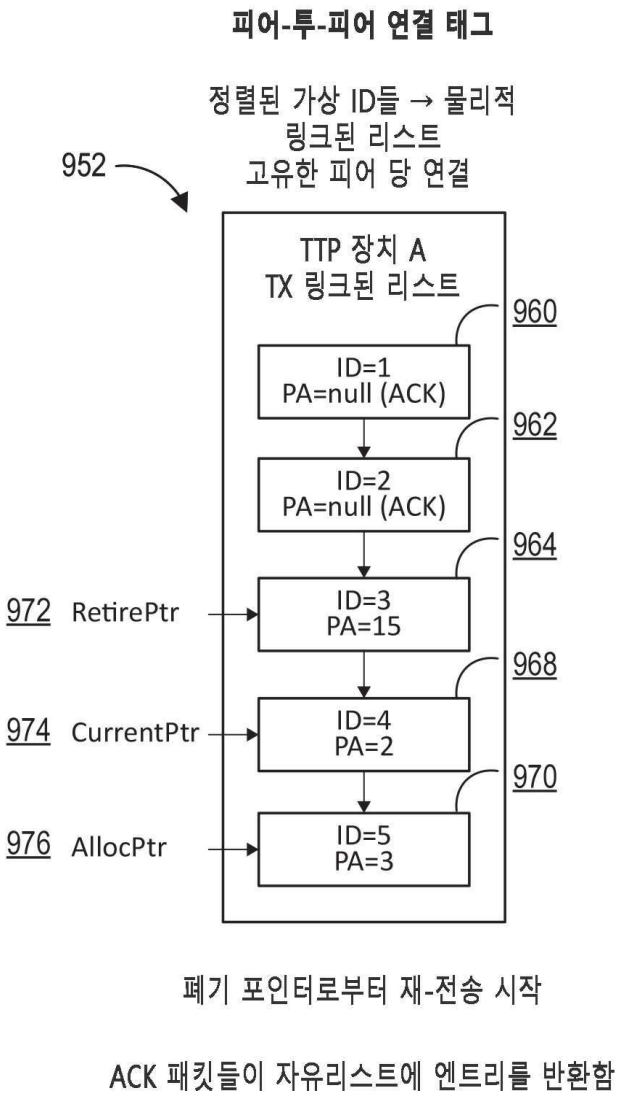
TTP_NACK_NOLINK
설명: 링크가 닫혔다는 TTP_PAYLOAD 요청에 대한 비-확인 응답

연산 코드 니모닉	인코딩[7:0]	TxID[31:0]	RxID[31:0]
TTP_NACK_NOLINK	VC[1:0], 6'h0A	N/A	TTP_PAYLOAD TxID

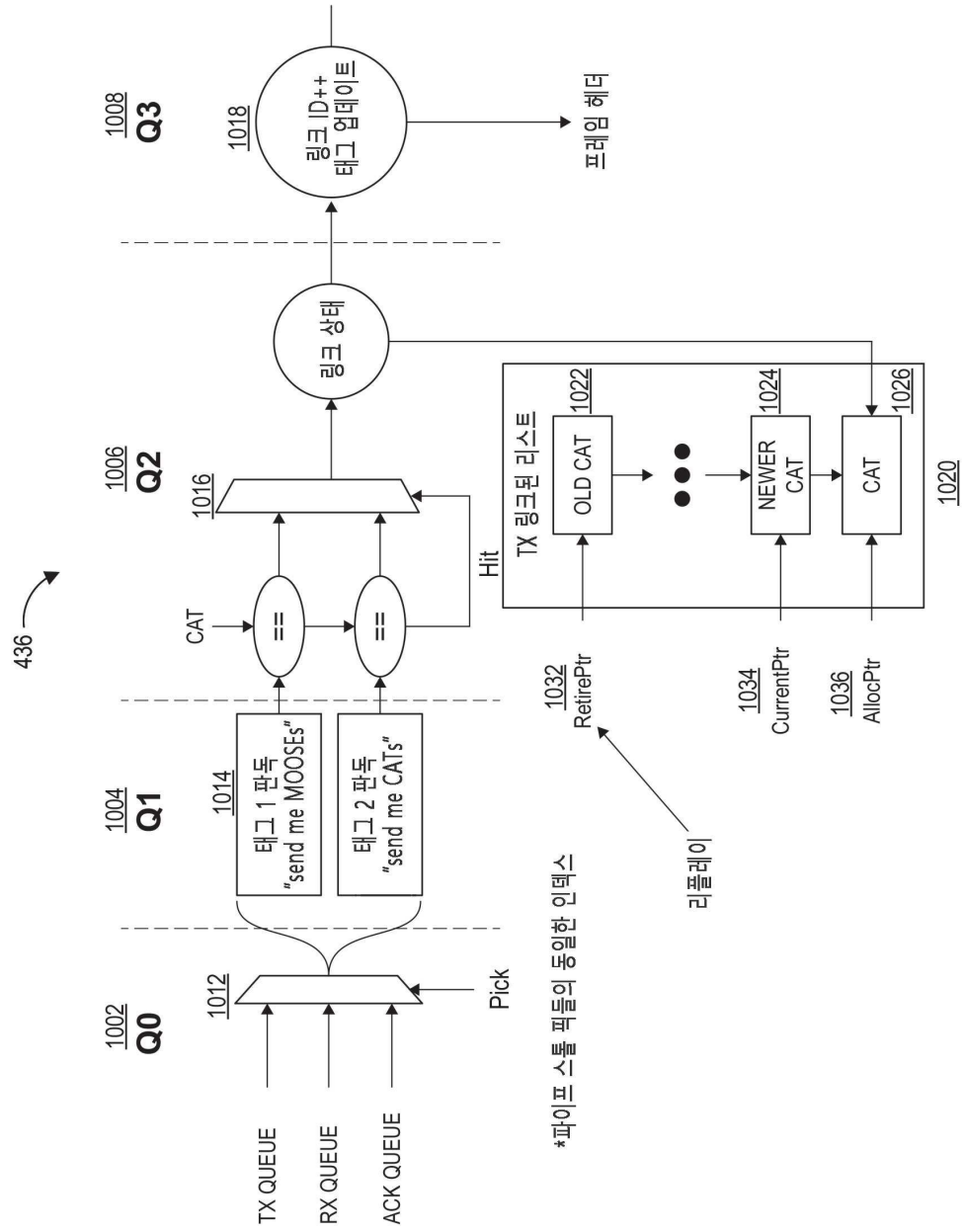
도면8


802
자유리스트 832
Pkt 4 TAG, PA=2
Pkt 5 TAG, PA=3
Pkt 3 TAG, PA=15
Pkt 물리적 태그
804
808
물리적 주소
포인터
TTP 패킷 스트림
(정렬됨)
패킷 1
패킷 2
패킷 3
패킷 4
패킷 5
820
자유리스트 834
Pkt 4 DATA, PA=2 812
Pkt 5 DATA, PA=3 814
Pkt 3 DATA, PA=15 810
Pkt 물리적 데이터
806

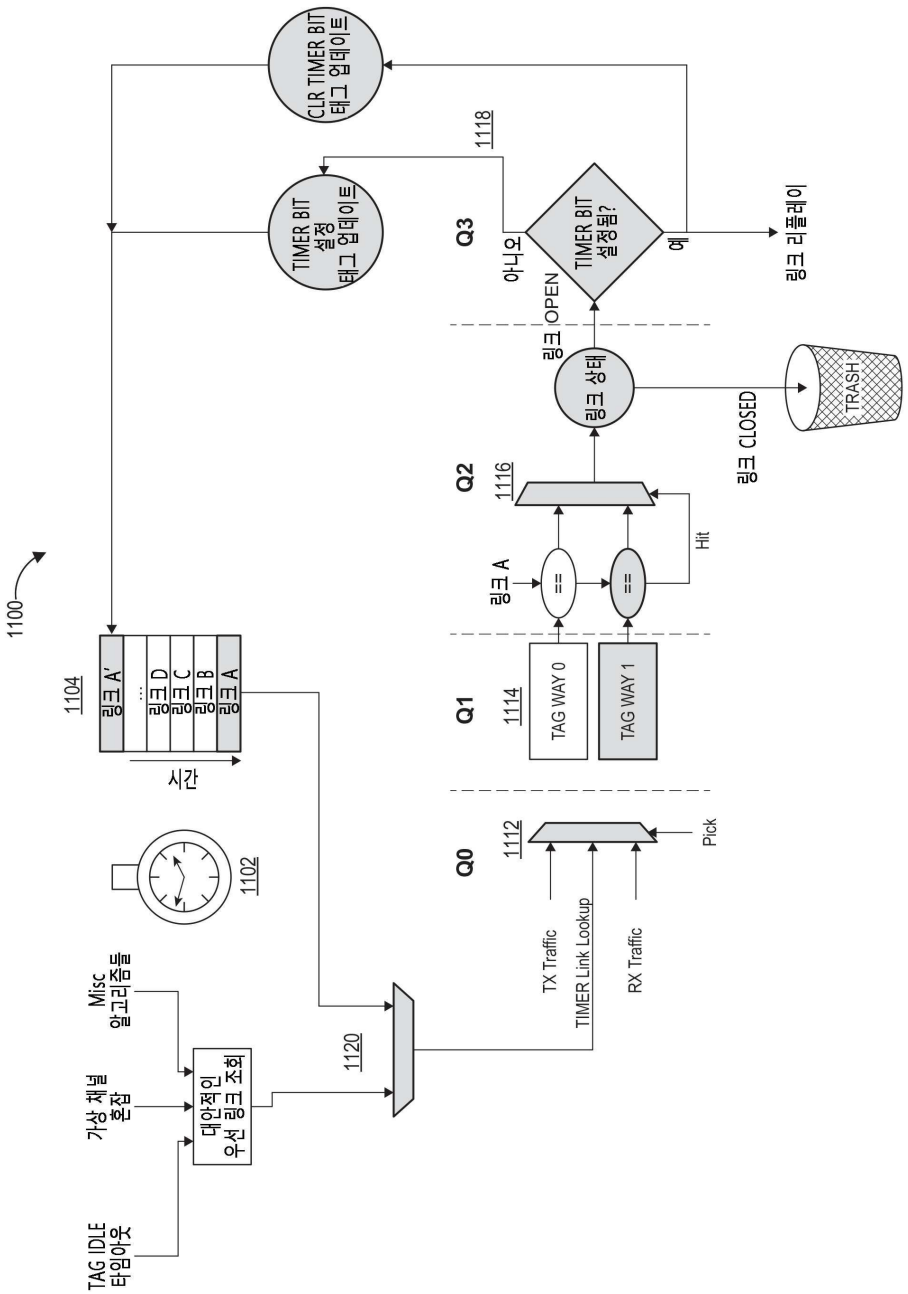
도면9


피어-투-피어 연결 태그
정렬된 가상 ID들 → 물리적
링크된 리스트
고유한 피어 당 연결
952
TTP 장치 A
TX 링크된 리스트
960
ID=1
PA=null (ACK)
962
ID=2
PA=null (ACK)
964
ID=3
PA=15
968
ID=4
PA=2
970
ID=5
PA=3
972 RetirePtr
974 CurrentPtr
976 AllocPtr
폐기 포인터로부터 재-전송 시작
ACK 패킷들이 자유리스트에 엔트리를 반환함

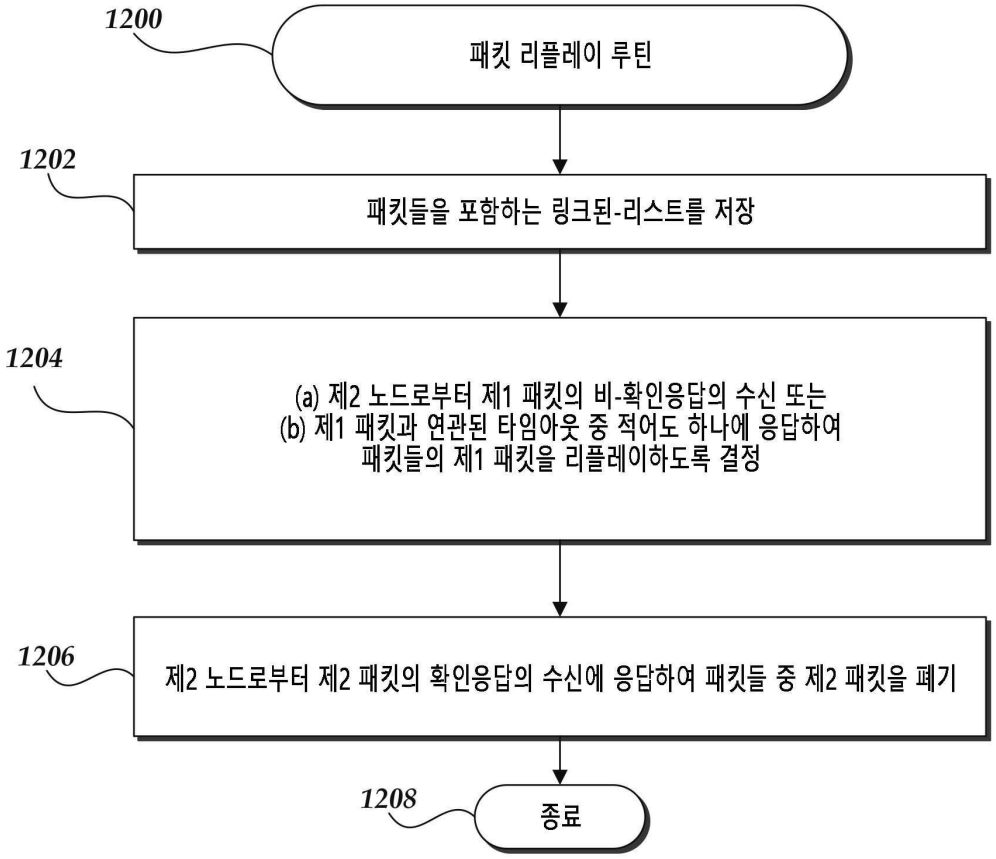
도면10


436
1002
Q0
1004
Q1
1006
Q2
1008
Q3
TX QUEUE
RX QUEUE
ACK QUEUE
1012
Pick
1014
태그 1 판독
"send me MOOSEs"
태그 2 판독
"send me CATs"
CAT
==
==
1016
Hit
링크 상태
1018
링크 ID++
태그 업데이트
프레임 헤더
TX 링크된 리스트
1020
1022
OLD CAT
1024
NEWER CAT
1026
CAT
1032
RetirePtr
1034
CurrentPtr
1036
AllocPtr
리플레이
*파이프 스톨 픽들의 동일한 인덱스

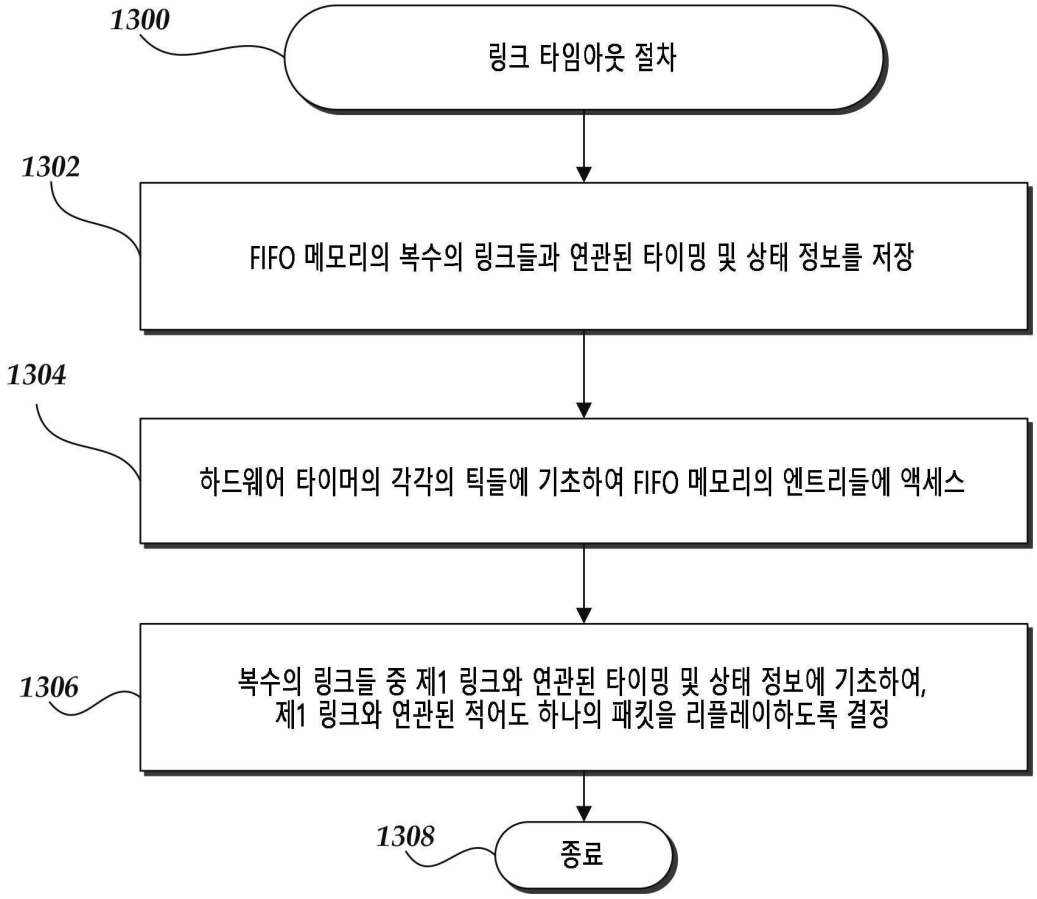
도면11


1100
1104
링크 A'
...
링크 D
링크 C
링크 B
링크 A
시간
1102
Misc 알고리즘들
가상 채널 혼잡
TAG IDLE 타임아웃
대안적인 우선 링크 조회
1120
Q0
1112
Pick
TX Traffic
TIMER Link Lookup
RX Traffic
Q1
1114
TAG WAY 0
TAG WAY 1
링크 A
==
==
Hit
Q2
1116
링크 상태
링크 CLOSED
TRASH
링크 OPEN
Q3
1118
TIMER BIT 설정됨?
아니오
예
링크 리플레이
TIMER BIT 설정 태그 업데이트
CLR TIMER BIT 태그 업데이트

도면12


1200
패킷 리플레이 루틴
1202
패킷들을 포함하는 링크된-리스트를 저장
1204
(a) 제2 노드로부터 제1 패킷의 비-확인응답의 수신 또는
(b) 제1 패킷과 연관된 타임아웃 중 적어도 하나에 응답하여
패킷들의 제1 패킷을 리플레이하도록 결정
1206
제2 노드로부터 제2 패킷의 확인응답의 수신에 응답하여 패킷들 중 제2 패킷을 폐기
1208
종료

도면13


1300
링크 타임아웃 절차
1302
FIFO 메모리의 복수의 링크들과 연관된 타이밍 및 상태 정보를 저장
1304
하드웨어 타이머의 각각의 틱들에 기초하여 FIFO 메모리의 엔트리들에 액세스
1306
복수의 링크들 중 제1 링크와 연관된 타이밍 및 상태 정보에 기초하여,
제1 링크와 연관된 적어도 하나의 패킷을 리플레이하도록 결정
1308
종료