국민대학교
KOOKMIN UNIVERSITY

패킷 레벨 중복제거

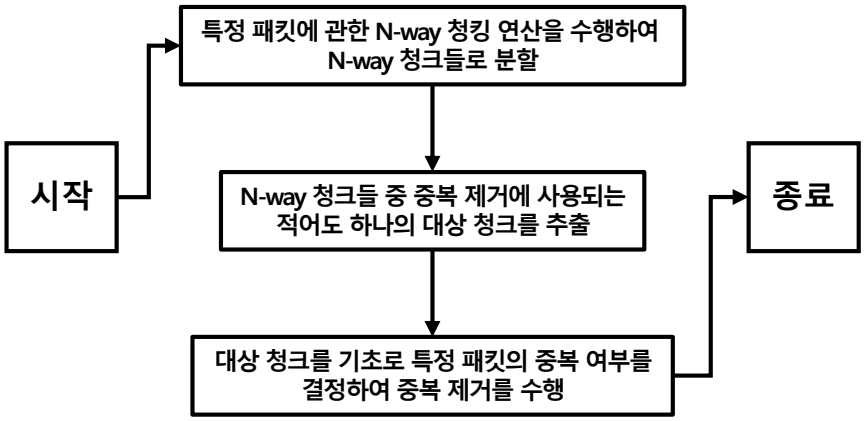
- 이름 : 윤명근
- 소속 : 소프트웨어융합대학
- 연구분야 : 소프트웨어

트래픽 처리 효율이 증가된
패킷레벨 중복 제거 장치

기술개요

- 본 기술은 패킷레벨에서 패킷의 중복을 제거 하여 네트워크 대역폭을 절약 하는 기술이다.
- 본 기술은 패킷 크기에 관계없이 미리 정의된 수의 청크들로 분할 할 수 있다.

*청크 : 규칙 기반 시스템에서 단일 단위로서 저장·검색되는 사실의 집합체 (IT용어대사전)



기술성

- 데이터 중복 제거를 통한 데이터 스토리지 공간 구축
- 웹 서버를 연결하는 링크 평균 대역폭의 감소로 인한 제품 경쟁 우위
- 3-way 청킹을 활용한 트래픽 처리 속도 증가

대표청구항

- 특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 N-way 청크들(chunks)로 분할하는 청크 분할부

  상기 N-way 청크들(chunks) 중 중복 제거에 사용되는 적어도 하나의 대상 청크(chunk)를 추출하는 청크 추출부 및 상기 적어도 하나의 대상 청크(chunk)를 기초로 상기 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행하는 중복 제거 처리부를 포함하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치

지식재산권

- 경량 복잡도 기반의 패킷레벨 중복 제거 장치 및 방법, 이를 저장하는 기록매체 (10-2017-0144878)

(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년09월27일
(11) 등록번호 10-2026125
(24) 등록일자 2019년09월23일

(51) 국제특허분류(Int. Cl.)
H04L 12/823 (2013.01) *H04L 12/805* (2013.01)
(52) CPC특허분류
H04L 47/32 (2013.01)
H04L 47/36 (2013.01)
(21) 출원번호 10-2017-0144878
(22) 출원일자 2017년11월01일
심사청구일자 2017년11월01일
(65) 공개번호 10-2019-0049244
(43) 공개일자 2019년05월09일
(56) 선행기술조사문헌
KR101465891 B1*
KR1020150064593 A*
US20160259572 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
국민대학교산학협력단
서울특별시 성북구 정릉로 77 (정릉동, 국민대학교)
(72) 발명자
윤명근
서울특별시 양천구 목동서로 397, B동 1003호(대림아크로빌)
정지만
서울특별시 은평구 백련산로4길 14, 402호(응암동, 청록아파트)
(74) 대리인
정부연

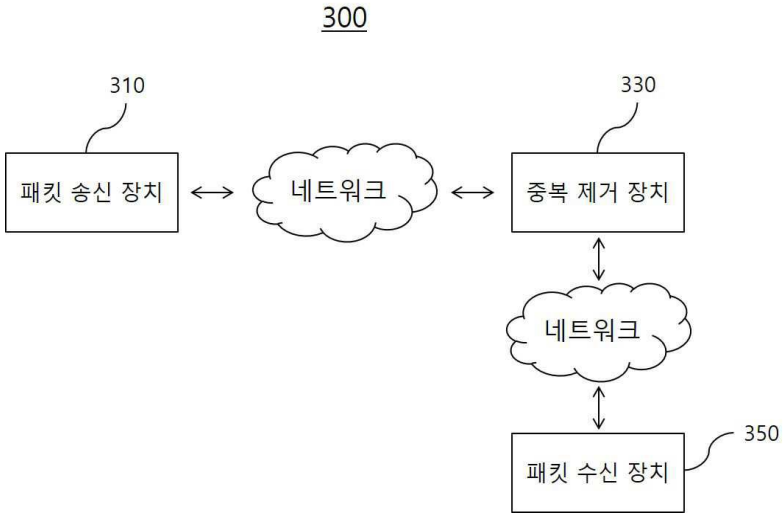
전체 청구항 수 : 총 12 항 심사관 : 김대성

(54) 발명의 명칭 **경량 복잡도 기반의 패킷레벨 중복 제거 장치 및 방법, 이를 저장하는 기록매체**

(57) 요 약

본 발명은 경량 복잡도 기반의 패킷레벨(packet-level) 중복제거 장치 및 방법에 관한 것으로, 특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 N-way 청크들(chunks)로 분할하는 청크 분할부, 상기 N-way 청크들(chunks) 중 중복 제거에 사용되는 적어도 하나의 대상 청크(chunk)를 추출하는 청크 추출부 및 상기 적어도 하나의 대상 청크(chunk)를 기초로 상기 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행하는 중복제거 처리부를 포함한다. 따라서, 본 발명은 패킷레벨(packet-level)에서 패킷의 중복 부분을 제거하여 네트워크 대역폭을 절약할 수 있는 효과를 가진다.

대 표 도 - 도3



이 발명을 지원한 국가연구개발사업
과제고유번호 1711054703
부처명 과학기술정보통신부
연구관리전문기관 한국연구재단
연구사업명 개인기초연구(미래부)
연구과제명 차세대 보안 모니터링을 위한 자가 조절형 스트리밍 알고리즘
기 여 율 1/1
주관기관 국민대학교
연구기간 2017.06.01 ~ 2018.03.31

명 세 서

청구범위

청구항 1

특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 적어도 제1 청크(chunk), 제2 청크 및 제3 청크로 구성된 N-way 청크들(chunks)로 분할하는 청크 분할부;

상기 N-way 청크들(chunks) 중 중복 제거에 사용되는 적어도 하나의 대상 청크(chunk)를 추출하는 청크 추출부; 및

상기 적어도 하나의 대상 청크(chunk)를 기초로 상기 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행하는 중복제거 처리부를 포함하되,

상기 청크 분할부는 상기 패킷의 처음부터 시작하는 전방향 가변 청킹(variable-size chunking) 연산을 수행하여 상기 제1 청크를 생성하고, 상기 전방향 가변 청킹과 병렬적으로 진행되고 상기 패킷의 끝부터 시작하는 역방향 가변 청킹 연산을 수행하여 상기 제3 청크를 생성하며, 상기 제1 및 제3 청크들을 제거한 상기 패킷의 나머지 부분을 상기 제2 청크로 결정하는 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치.

청구항 2

삭제

청구항 3

삭제

청구항 4

제1항에 있어서, 상기 청크 추출부는

상기 제2 청크(chunk)를 상기 적어도 하나의 대상 청크(chunk)로서 결정하는 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치.

청구항 5

제1항에 있어서, 상기 청크 추출부는

상기 N-way 청크들(chunks) 중 전방향 가변 청크(chunk)와 후방향 가변 청크(chunk)를 제외한 내부 청크(chunk)를 기초로 상기 적어도 하나의 대상 청크(chunk)를 결정하는 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치.

청구항 6

제1항에 있어서, 상기 중복제거 처리부는

상기 적어도 하나의 대상 청크(chunk)를 중복 제거 해시 함수(hash function)의 입력으로 제공하여 상기 특정 패킷의 중복 여부를 결정하는 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치.

청구항 7

제6항에 있어서, 상기 중복제거 처리부는

상기 중복 제거 해시 함수(hash function)의 출력으로 상기 적어도 하나의 대상 청크(chunk)에 대한 핑거프린트(fingerprint)를 산출하고, 상기 산출된 핑거프린트(fingerprint)를 중복 제거 해시 테이블(hash table)에서 검색하여 상기 특정 패킷의 중복여부를 결정하는 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치.

청구항 8

제7항에 있어서, 상기 중복제거 처리부는

상기 특정 패킷이 중복인 경우에는 상기 적어도 하나의 대상 청크(chunk)를 상기 해시 테이블(hash table)의 중복 제거 해시 인덱스(index)로 대체하는 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치.

청구항 9

패킷레벨(packet-level) 중복 제거 장치에서 수행되는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 방법에 있어서,

(a) 특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 적어도 제1 청크(chunk), 제2 청크 및 제3 청크로 구성된 N-way 청크들(chunks)로 분할하는 단계;

(b) 상기 N-way 청크들(chunks) 중 중복 제거를 위해 적어도 하나의 대상 청크(chunk)를 추출하는 단계; 및

(c) 상기 적어도 하나의 대상 청크(chunk)를 기초로 상기 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행하는 단계를 포함하되,

상기 (c) 단계는 상기 패킷의 처음부터 시작하는 전방향 가변 청킹(variable-size chunking) 연산을 수행하여 상기 제1 청크를 생성하고, 상기 전방향 가변 청킹과 병렬적으로 진행되고 상기 패킷의 끝부터 시작하는 역방향 가변 청킹 연산을 수행하여 상기 제3 청크를 생성하며, 상기 제1 및 제3 청크들을 제거한 상기 패킷의 나머지 부분을 상기 제2 청크로 결정하는 단계인 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 방법.

청구항 10

삭제

청구항 11

삭제

청구항 12

제9항에 있어서, 상기 (b) 단계는

상기 제2 청크(chunk)를 상기 적어도 하나의 대상 청크(chunk)로서 결정하는 단계인 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 방법.

청구항 13

제9항에 있어서, 상기 (c) 단계는

상기 적어도 하나의 대상 청크(chunk)를 중복 제거 해시 함수(hash function)의 입력으로 제공하여 상기 특정 패킷의 중복 여부를 결정하는 단계인 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제

거 방법.

청구항 14

제13항에 있어서, 상기 (c) 단계는

상기 중복 제거 해시 함수(hash function)의 출력으로 상기 적어도 하나의 대상 청크(chunk)에 대한 핑거프린트(fingerprint)를 산출하고, 상기 산출된 핑거프린트(fingerprint)를 중복 제거 해시 테이블(hash table)에서 검색하여 상기 특정 패킷의 중복여부를 결정하는 단계인 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 방법.

청구항 15

제14항에 있어서, 상기 (c) 단계는

상기 특정 패킷이 중복인 경우에는 상기 적어도 하나의 대상 청크(chunk)를 상기 해시 테이블(hash table)의 중복 제거 해시 인덱스(index)로 대체하는 단계인 것을 특징으로 하는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 방법.

청구항 16

특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 적어도 제1 청크(chunk), 제2 청크 및 제3 청크로 구성된 N-way 청크들(chunks)로 분할하는 단계;

상기 N-way 청크들(chunks) 중 중복 제거를 위해 적어도 하나의 대상 청크(chunk)를 추출하는 단계; 및

상기 적어도 하나의 대상 청크(chunk)를 기초로 상기 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행하는 단계를 포함하되,

상기 중복 제거를 수행하는 단계는 상기 패킷의 처음부터 시작하는 전방향 가변 청킹(variable-size chunking) 연산을 수행하여 상기 제1 청크를 생성하고, 상기 전방향 가변 청킹과 병렬적으로 진행되고 상기 패킷의 끝부터 시작하는 역방향 가변 청킹 연산을 수행하여 상기 제3 청크를 생성하며, 상기 제1 및 제3 청크들을 제거한 상기 패킷의 나머지 부분을 상기 제2 청크로 결정하는 단계인 것을 특징으로 하는 방법을 수행하는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 패킷레벨(packet-level) 중복 제거 기술에 관한 것으로, 보다 상세하게는, 패킷레벨(packet-level)에서 패킷의 중복 부분을 제거하여 네트워크 대역폭을 절약할 수 있는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치 및 방법에 관한 것이다.

배 경 기 술

[0003] 반복되고 중복된 데이터를 제거하는 중복 제거(deduplication) 기술은 스토리지 공간 및 네트워크 대역폭을 절약하는데 널리 사용된다. 스토리지 시스템은 동일하거나 유사한 파일의 원본 복사본을 오직 하나만 저장하고 각 복제본에 대해서는 인덱스를 할당한다. 패킷레벨(packet-level) 중복 제거 기술은 기업 및 대학 액세스 링크(access link) 및 사용량이 많은 웹 서버(web server)를 연결하는 링크에 대해 평균 대역폭을 15~60% 절감할 수 있다.

[0004] 한국 등록특허공보 제10-1465891(2014.11.20)호는 무선 네트워크에서 트래픽 중복 제거 방법 및 장치에 관한 것으로, 더욱 상세하게는 트래픽 중복 제거 방법은 패킷을 수신하는 단계, 상기 패킷을 분석하여 중복 청크를 검

색하는 단계, 상기 중복 청크가 존재하는 경우, 상기 중복 청크 및 대상 단말의 채널 상태를 이용하여 상기 대상 단말의 오버히어링 확률을 계산하는 단계, 상기 계산된 오버히어링 확률에 따라 상기 중복 청크를 제거하여 상기 패킷을 인코딩하는 단계, 및 상기 인코딩된 패킷을 상기 대상 단말로 전달하는 단계를 포함한다.

[0005] 한국 공개특허공보 제10-2015-0023896(2015.03.05)호는 중복 제거 미들박스들의 스케일링에 관한 것으로, 더욱 상세하게는 프로세서 및 상기 프로세서에 통신 가능하게 연결된 메모리를 포함하는 데이터 프로세싱 모듈로서, 수신된 패킷의 패킷 클래스에 기초하여, 복수의 중복 제거(RE) 프로세싱 기능들 중 어떤 것을 상기 수신된 패킷에 대해 수행할지를 결정하도록 구성되는, 상기 데이터 프로세싱 모듈을 포함한다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 한국 등록특허공보 제10-1465891(2014.11.20)호

(특허문헌 0002) 한국 공개특허공보 제10-2015-0023896(2015.03.05)호

발명의 내용

해결하려는 과제

[0008] 본 발명의 일 실시예는, 패킷레벨(packet-level)에서 패킷의 중복 부분을 제거하여 네트워크 대역폭을 절약할 수 있는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치 및 방법를 제공하고자 한다.

[0009] 본 발명의 일 실시예는, 패킷 크기에 관계없이 모든 패킷을 미리 정의된 수의 청크(chunk)들로 분할할 수 있는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치 및 방법를 제공하고자 한다.

[0010] 본 발명의 일 실시예는, 청킹(chunking), 핑거프린팅(fingerprinting) 및 해시 테이블(hash table)의 최적 조합을 선택하여 패킷레벨(packet-level) 중복 제거를 효과적으로 수행할 수 있는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치 및 방법를 제공하고자 한다.

과제의 해결 수단

[0012] 실시예들 중에서, 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치는 특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 N-way 청크들(chunks)로 분할하는 청크 분할부, 상기 N-way 청크들(chnuks) 중 중복 제거에 사용되는 적어도 하나의 대상 청크(chunk)를 추출하는 청크 추출부 및 상기 적어도 하나의 대상 청크(chunk)를 기초로 상기 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행하는 중복제거 처리부를 포함한다.

[0013] 상기 청크 분할부는 상기 특정 패킷을 적어도 제1 청크(chunk), 제2 청크(chunk) 및 제3 청크(chunk)로 구성된 청크들(chunks)로 분할할 수 있다.

[0014] 상기 청크 분할부는 상기 패킷에 관해 전방향 가변 청킹(variable-size chunking) 연산을 수행하여 상기 제1 청크(chunk)를 생성하고 역방향 가변 청킹(variable-size chunking) 연산을 수행하여 상기 제3 청크(chunk)를 생성하여, 상기 제2 청크(chunk)를 결정할 수 있다.

[0015] 상기 청크 추출부는 상기 제2 청크(chunk)를 상기 적어도 하나의 대상 청크(chunk)로서 결정할 수 있다.

[0016] 상기 청크 추출부는 상기 N-way 청크들(chunks) 중 전방향 가변 청크(chunk)와 후방향 가변 청크(chunk)를 제외한 내부 청크(chunk)를 기초로 상기 적어도 하나의 대상 청크(chunk)를 결정할 수 있다.

[0017] 상기 중복제거 처리부는 상기 적어도 하나의 대상 청크(chunk)를 중복 제거 해시 함수(hash function)의 입력으로 제공하여 상기 특정 패킷의 중복 여부를 결정할 수 있다.

[0018] 상기 중복제거 처리부는 상기 중복 제거 해시 함수(hash function)의 출력으로 상기 적어도 하나의 대상 청크(chunk)에 대한 핑거프린트(fingerprint)를 산출하고, 상기 산출된 핑거프린트(fingerprint)를 중복 제거 해시 테이블(hash table)에서 검색하여 상기 특정 패킷의 중복여부를 결정할 수 있다.

[0019] 상기 중복제거 처리부는 상기 특정 패킷이 중복인 경우에는 상기 적어도 하나의 대상 청크(chunk)를 상기 해시

테이블(hash table)의 중복 제거 해시 인덱스(index)로 대체할 수 있다.

[0020] 실시예들 중에서, 경량 복잡도 기반의 패킷레벨 중복 제거 방법은 (a) 특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 N-way 청크들(chunks)로 분할하는 단계, (b) 상기 N-way 청크들(chunks) 중 중복 제거를 위해 적어도 하나의 대상 청크(chunk)를 추출하는 단계 및 (c) 상기 적어도 하나의 대상 청크(chunk)를 기초로 상기 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행하는 단계를 포함한다.

[0021] 상기 (a) 단계는 상기 특정 패킷을 적어도 제1 청크(chunk), 제2 청크(chunk) 및 제3 청크(chunk)로 구성된 청크들(chunk)로 분할할 수 있다.

[0022] 상기 (a) 단계는 상기 패킷에 관해 전방향 가변 청킹(variable-size chunking) 연산을 수행하여 상기 제1 청크(chunk)를 생성하고 역방향 가변 청킹(variable-size chunking) 연산을 수행하여 상기 제3 청크(chunk)를 생성하여, 상기 제2 청크(chunk)를 결정할 수 있다.

[0023] 상기 (b) 단계는 상기 제2 청크(chunk)를 상기 적어도 하나의 대상 청크(chunk)로서 결정할 수 있다.

[0024] 상기 (c) 단계는 상기 적어도 하나의 대상 청크(chunk)를 중복 제거 해시 함수(hash function)의 입력으로 제공하여 상기 특정 패킷의 중복 여부를 결정할 수 있다.

[0025] 상기 (c) 단계는 상기 중복 제거 해시 함수(hash function)의 출력으로 상기 적어도 하나의 대상 청크(chunk)에 대한 핑거프린트(fingerprint)를 산출하고, 상기 산출된 핑거프린트(fingerprint)를 중복 제거 해시 테이블(hash table)에서 검색하여 상기 특정 패킷의 중복여부를 결정할 수 있다.

[0026] 상기 (c) 단계는 상기 특정 패킷이 중복인 경우에는 상기 적어도 하나의 대상 청크(chunk)를 상기 해시 테이블(hash table)의 중복 제거 해시 인덱스(index)로 대체할 수 있다.

[0027] 실시예들 중에서, 컴퓨터로 읽을 수 있는 기록매체는 특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 N-way 청크들(chunks)로 분할하는 단계, 상기 N-way 청크들(chunks) 중 중복 제거를 위해 적어도 하나의 대상 청크(chunk)를 추출하는 단계 및 상기 적어도 하나의 대상 청크(chunk)를 기초로 상기 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행하는 단계를 포함하는 방법을 수행하는 프로그램을 기록한다.

발명의 효과

[0029] 개시된 기술은 다음의 효과를 가질 수 있다. 다만, 특정 실시예가 다음의 효과를 전부 포함하여야 한다거나 다음의 효과만을 포함하여야 한다는 의미는 아니므로, 개시된 기술의 권리범위는 이에 의하여 제한되는 것으로 이해되어서는 아니 될 것이다.

[0030] 본 발명의 일 실시예에 따른 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치 및 방법은 패킷 크기에 관계없이 모든 패킷을 미리 정의된 수의 청크들(chunks)로 분할할 수 있다.

[0031] 본 발명의 일 실시예에 따른 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치 및 방법은 청킹(chunking), 핑거프린팅(fingerprint) 및 해시 테이블(hash table)의 최적 조합을 선택하여 패킷레벨(packet-level) 중복 제거를 효과적으로 수행할 수 있다.

도면의 간단한 설명

[0033] 도 1은 고정 크기 청킹(fixed-size chunking) 알고리즘을 설명하는 예시도이다.

도 2는 가변 크기 청킹(variable-size chunking) 알고리즘을 설명하는 예시도이다.

도 3은 본 발명의 일 실시예에 따른 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 시스템을 설명하는 도면이다.

도 4는 도 3에 있는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치를 나타내는 블록도이다.

도 5는 도 3에 있는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치에서 수행되는 패킷레벨(packet-level) 중복 제거 과정의 일 실시예를 설명하는 순서도이다.

도 6은 본 발명의 일 실시예에 따른 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치에서 수행되는 N-way 청킹 연산을 설명하는 예시도이다.

도 7은 고정 크기 청킹(fixed-size chunking), 가변 크기 청킹(variable-size chunking) 및 3-way 청킹

(chunking)을 사용하여 수행한 비교 실험 결과를 보여주는 도면이다.

도 8은 청킹(chunking), 핑거프린팅(fingerprint) 및 해시 테이블(hash table)에 관한 여러 조합을 사용하여 3-way 청킹(chunking) 기법을 수행한 비교 실험 결과를 보여주는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0034] 본 발명에 관한 설명은 구조적 내지 기능적 설명을 위한 실시예에 불과하므로, 본 발명의 권리범위는 본문에 설명된 실시예에 의하여 제한되는 것으로 해석되어서는 아니 된다. 즉, 실시예는 다양한 변경이 가능하고 여러 가지 형태를 가질 수 있으므로 본 발명의 권리범위는 기술적 사상을 실현할 수 있는 균등물들을 포함하는 것으로 이해되어야 한다. 또한, 본 발명에서 제시된 목적 또는 효과는 특정 실시예가 이를 전부 포함하여야 한다거나 그러한 효과만을 포함하여야 한다는 의미는 아니므로, 본 발명의 권리범위는 이에 의하여 제한되는 것으로 이해되어서는 아니 될 것이다.

[0035] 한편, 본 출원에서 서술되는 용어의 의미는 다음과 같이 이해되어야 할 것이다.

[0036] "제1", "제2" 등의 용어는 하나의 구성요소를 다른 구성요소로부터 구별하기 위한 것으로, 이들 용어들에 의해 권리범위가 한정되어서는 아니 된다. 예를 들어, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다.

[0037] 어떤 구성요소가 다른 구성요소에 "연결되어"있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결될 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어"있다고 언급된 때에는 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다. 한편, 구성요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0038] 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한 복수의 표현을 포함하는 것으로 이해되어야 하고, "포함하다"또는 "가지다" 등의 용어는 실시된 특징, 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이며, 하나 또는 그 이상의 다른 특징이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0039] 각 단계들에 있어 식별부호(예를 들어, a, b, c 등)는 설명의 편의를 위하여 사용되는 것으로 식별부호는 각 단계들의 순서를 설명하는 것이 아니며, 각 단계들은 문맥상 명백하게 특정 순서를 기재하지 않는 이상 명기된 순서와 다르게 일어날 수 있다. 즉, 각 단계들은 명기된 순서와 동일하게 일어날 수도 있고 실질적으로 동시에 수행될 수도 있으며 반대의 순서대로 수행될 수도 있다.

[0040] 본 발명은 컴퓨터가 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현될 수 있고, 컴퓨터가 읽을 수 있는 기록 매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록 매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피 디스크, 광 데이터 저장 장치 등이 있다.

[0041] 여기서 사용되는 모든 용어들은 다르게 정의되지 않는 한, 본 발명이 속하는 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한 이상적이거나 과도하게 형식적인 의미를 지니는 것으로 해석될 수 없다.

[0043] 패킷(Packet)은 정보 기술에서 패킷 방식의 컴퓨터 네트워크가 전달하는 데이터의 형식화된 블록이다. 패킷을 지원하지 않는 컴퓨터 통신 연결은 단순히 바이트, 문자열, 비트를 독립적으로 연속하여 데이터를 전송한다. 데이터가 패킷으로 형식이 바뀔 때, 네트워크는 장문 메시지를 더 효과적이고 신뢰성 있게 보낼 수 있다. 패킷은 데이터의 한 단위라고 할 수 있고, 헤더(Header)와 페이로드(Payload)로 구성될 수 있다. 페이로드(Payload)는 패킷 송신 장치에서 전송하는 사용자 데이터의 일 부분에 해당할 수 있고, 패킷의 내부에 존재할 수 있다. 패킷 헤더(Packet Header)는 페이로드를 포함하는 패킷의 헤더에 해당할 수 있다.

[0044] 패킷레벨(packet-level) 중복 제거의 기본 개념은 청크(chunk)라고 하는 패킷의 반복된 하위 부분을 찾아 크기를 원래 청크(chunk)보다 작은 인덱스 값으로 대체하는 것이다. 따라서, 각 패킷은 먼저 청크(chunk)로 분할되어야 한다. 그런 다음 청크(chunk)는 가능한 많은 인덱스로 대체되어야 한다. 문제는 패킷들 사이에서 반복적인 하위 부분을 효율적으로 찾는 방법이다. 최첨단 방식은 모든 패킷을 겹치지 않는 하위 파트로 나누고 인덱스 값

으로 대체될 중복 청크(chunk)를 식별하는 청킹(chunking) 알고리즘에 의존하고 있다. 문제는 모든 바이트가 특정 양의 계산을 필요로 하기 때문에 청크(chunk)가 CPU를 많이 사용한다는 것이다. 모든 청킹(chunking) 알고리즘은 O(n)의 시간 복잡도를 가지고, 여기서 n은 패킷의 바이트 수이다. 최상의 패킷레벨(packet-level) 중복 제거 기법의 최대 처리량은 CPU코어 당 2Gbps에 불과하다.

[0046] 현재까지 많은 청킹(chunking) 알고리즘이 연구되고 제안되어 왔으며, 고정 크기 청킹(fixed-size chunking)과 가변 크기 청킹(variable-size chunking)의 두 그룹으로 분류 할 수 있다. 도 1은 고정 크기 청킹(fixed-size chunking) 알고리즘을 설명하는 예시도이고, 도 2는 가변 크기 청킹(variable-size chunking) 알고리즘을 설명하는 예시도이다. 도 1 및 도 2를 참조하면, packet1과 packet2로 표시된 두 개의 유사한 패킷이 표시되어 있고, 청크(chunk)는 c_i로 표시되어 있다. 두 번째 패킷 앞에 하나의 추가 바이트('b')가 있다.

[0047] 도 1에서, 고정 크기 청킹(fixed-size chunking)은 각 패킷을 동일한 크기의 여러 청크(chunk)로 분할한다. 고정 크기 청킹(fixed-size chunking)은 매우 간단하고, 알고리즘은 빠르게 실행될 수 있다. 그러나, 고정 크기 청킹(fixed-size chunking)은 패킷의 시작 부분에 적은 수의 바이트가 삽입되거나 삭제될 때 제대로 작동하지 않는 문제(경계 시프트 문제라 한다.)를 가지고 있다. 예를 들어, 두 번째 패킷에서 하나의 추가 문자('b')는 모든 청크(chunk)를 첫 번째 패킷과 다르게 하기 때문에 도 1의 고정 크기 청킹(fixed-size chunking)은 중복 청크(chunk)를 생성하지 않는다. 고정 크기 청킹(fixed-size chunking)은 윈도우(window)의 내용을 고려하지 않는다. 예를 들어, 윈도우 크기를 w라고 할 경우, 도 1에서 고정 크기 청킹(fixed-size chunking)은 w = 3을 사용한다.

[0048] 도 2에서, 가변 크기 청킹(variable-size chunking)은 패킷을 내용 기반 방식으로 분할하여 경계 시프트(boundary shift) 문제를 해결할 수 있다. 가변 크기 청킹(variable-size chunking)은 슬라이딩 윈도우(sliding window)를 전체 패킷에 통과시키면서 각 윈도우 내부에서 체크섬(checksum)을 계산할 수 있다. Rabin의 롤링 해시 함수(Rabin's rolling hash function)는 체크섬(checksum)을 계산하는데 가장 널리 사용되는 알고리즘이다. 체크섬(checksum)이 특정 조건을 만족하는 경우, 예를 들어, 체크섬(checksum)이 3개의 연속하는 '0'비트로 시작하면, 청크(chunk) 사이에 구분 기호가 있는 것으로 판단할 수 있다. 도 2에서, a_0a_1, a_5a_6 및 a_7a_8의 세 문자열이 구분 기호 조건을 충족시키는 것을 나타낸다. 윈도우 크기를 w라고 할 경우, 도 2에서 가변 크기 청킹(variable-size chunking)은 w = 2를 사용한다.

[0049] N-way 청킹(chunking)은 모든 패킷에 대해 고정된 N개의 청크(chunk)로 분할하는 청킹(chunking) 기법에 해당한다. 고정된 개수의 청크(chunk)들로 분할하는 N-way 청킹(chunking)은 시간 복잡도를 일정한 수준으로 유지할 수 있다. 특히, 모든 패킷을 세개의 청크(chunk)로 분할할 수 있는 3-way 청킹(chunking)은 상수시간(constant time)의 시간 복잡도를 가질 수 있다. 경량 복잡도는 종래의 청킹(chunking) 알고리즘이 O(n)의 시간 복잡도를 가지는 것에 비해 N-way 청킹(chunking)은 보다 향상된 시간 복잡도를 가진다는 의미를 포함하고 있다.

[0051] 도 3은 본 발명의 일 실시예에 따른 경량 복잡도 기반의 패킷레벨 중복 제거 시스템을 설명하는 도면이다.

[0052] 도 3을 참조하면, 경량 복잡도 기반의 패킷레벨 중복 제거 시스템(300)은 패킷 송신 장치(310), 경량 복잡도 기반의 패킷레벨 중복 제거 장치(이하, 패킷레벨 중복 제거 장치라 한다.)(330) 및 패킷 수신 장치(350)를 포함할 수 있다.

[0053] 패킷 송신 장치(310)는 패킷을 전송하는 컴퓨팅 장치에 해당하고, 스마트폰, 노트북 또는 컴퓨터로 구현될 수 있으며, 반드시 이에 한정되지 않고, 태블릿 PC 등 다양한 디바이스로도 구현될 수 있다. 패킷 송신 장치(310)는 패킷레벨 중복 제거 장치(330)와 네트워크를 통해 연결될 수 있고, 실시간으로 데이터를 주고 받을 수 있다.

[0054] 패킷레벨 중복 제거 장치(330)는 패킷 송신 장치(310)로부터 패킷을 수신하여 패킷의 중복된 부분을 제거하고 패킷 수신 장치(350)로 중복 제거된 패킷을 전송하는 컴퓨터 또는 프로그램에 해당하는 서버로 구현될 수 있다. 패킷레벨 중복 제거 장치(330)는 패킷 송신 장치(310) 및 패킷 수신 장치(350)와 네트워크를 통해 연결될 수 있고, 실시간으로 데이터를 주고 받을 수 있다.

[0055] 패킷 수신 장치(350)는 패킷을 수신하는 컴퓨팅 장치에 해당하고, 스마트폰, 노트북 또는 컴퓨터로 구현될 수 있으며, 반드시 이에 한정되지 않고, 태블릿 PC 등 다양한 디바이스로도 구현될 수 있다. 패킷 송신 장치(310)는 패킷레벨 중복 제거 장치(330)와 네트워크를 통해 연결될 수 있고, 실시간으로 데이터를 주고 받을 수 있다.

[0056] 일 실시예에서, 패킷 송신 장치(310) 및 패킷 수신 장치(350)는 패킷을 송수신할 수 있는 하나의 장치에 포함되어 구현될 수 있다. 일 실시예에서, 패킷레벨 중복 제거 장치(330)는 패킷 송신 장치(310) 또는 패킷 수신 장치

(350)에 포함되어 구현될 수 있다. 패킷 송신 장치(310)가 패킷레벨 중복 제거 장치(330)를 포함하여 구현될 경우 패킷 송신 장치(310)는 패킷 수신 장치(350)로 패킷을 전송하기 전에 패킷의 중복을 제거할 수 있고, 패킷 수신 장치(350)는 중복 제거된 패킷을 수신하여 원래의 패킷 내용을 복원할 수 있다.

[0058] 도 4는 도 3에 있는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치를 나타내는 블록도이다.

[0059] 도 4를 참조하면, 패킷레벨 중복 제거 장치(330)는 청크 분할부(410), 청크 추출부(430), 중복제거 처리부(450) 및 제어부(470)를 포함할 수 있다.

[0060] 청크 분할부(410)는 특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 N-way 청크들(chunks)로 분할할 수 있다. 예를 들어 청크 분할부(410)는 각각의 패킷을 3개, 4개 또는 5개 등 미리 정의된 고정 개수만큼 패킷을 분할할 수 있고, 고정된 개수에 따라 3-way, 4-way 또는 5-way 청킹(chunking) 연산에 해당할 수 있다. 청크 분할부(410)는 고정 개수의 패킷으로 분할하는 방법을 사용함으로써 알고리즘 복잡성을 일정한 수준으로 유지할 수 있다.

[0061] 일 실시예에서, 청크 분할부(410)는 특정 패킷을 적어도 제1 청크(chunk), 제2 청크(chunk) 및 제3 청크(chunk)로 구성된 청크들(chunks)로 분할할 수 있다. 실제 인터넷 트래픽을 사용한 실험에서 처리 속도와 중복 제거 효과를 모두 달성하기 위해서는 각 패킷을 3개의 청크(chunk)로 분할하는 3-way 청킹(chunking)이 적합한 것으로 나타났다.

[0062] 청크 분할부(410)는 특정 패킷을 분할하여 고정된 개수만큼의 청크들(chunks)을 생성할 수 있으며, 특정 패킷의 처음부터 차례대로 분할된 패킷들을 제1, 제2 내지 제n 청크(chunk)로 지정할 수 있다. 예를 들어, 특정 패킷을 3개의 청크들(chunks)로 분할하는 3-way 청킹(chunking)을 이용하면 청크 분할부(410)는 패킷의 처음부터 차례대로 분할된 3-way 청크들(chunks)을 제1 청크(chunk), 제2 청크(chunk) 및 제3 청크(chunk)로 지정할 수 있다.

[0063] 일 실시예에서, 청크 분할부(410)는 패킷에 관해 전방향 가변 청킹(variable-size chunking) 연산을 수행하여 제1 청크(chunk)를 생성하고 역방향 가변 청킹(variable-size chunking) 연산을 수행하여 제3 청크(chunk)를 생성하여 제2 청크(chunk)를 결정할 수 있다. 보다 구체적으로, 청크 분할부(410)는 패킷의 처음부터 시작하여 가변 청킹(variable-size) 연산을 수행할 수 있다. 청크 분할부(410)는 슬라이딩 윈도우(sliding window)를 통과시키면서 각 윈도우 내부에서 체크섬(checksum)을 계산하고, 체크섬(checksum)이 특정 조건을 만족하는 청크(chunk) 사이의 첫 번째 구분기호를 발견하면 해당 구분기호를 기준으로 패킷을 분할하여 제1 청크(chunk)를 생성할 수 있다.

[0064] 청크 분할부(410)는 제1 청크(chunk)를 분할하는 과정을 특정 패킷의 끝부터 시작하여 역방향으로 동일하게 진행할 수 있다. 청크 분할부(410)는 슬라이딩 윈도우(sliding window)가 패킷의 마지막 w바이트(여기에서, w는 슬라이딩 윈도우의 크기)로 시작하여 역방향으로 이동하면서 체크섬(checksum)을 계산하고, 첫 번째 구분기호가 발견되면 패킷의 끝부분부터 해당 구분 기호를 기준으로 패킷을 분할하여 제3 청크(chunk)를 생성할 수 있다.

[0065] 청크 분할부(410)는 특정 패킷의 처음과 끝부분에서 동시에 가변 청킹(variable-size chunking) 연산을 수행하여 제1 청크(chunk) 및 제3 청크(chunk)를 생성할 수 있고, 특정 패킷에서 제1 청크(chunk) 및 제3 청크(chunk)를 분할한 나머지 부분을 제2 청크(chunk)로 결정할 수 있다. 청크 분할부(410)는 가변 청킹(variable-size chunking) 연산을 통해 제1 및 제3 청크(chunk)를 병렬적으로 생성한 후 제1 및 제3 청크(chunk)를 제거한 패킷의 나머지 부분을 제2 청크(chunk)로 결정함으로써 상수시간의 시간 복잡도를 갖는 3-way 청킹(chunking) 연산을 수행할 수 있다.

[0066] 일 실시예에서, 청크 분할부(410)는 제1 및 제2 청크(chunk)를 생성하는데 임의의 가변 청킹(variable-size chunking) 알고리즘 중 동일한 청킹(chunking) 알고리즘을 사용할 수 있고, 제1 및 제2 청크(chunk) 각각에 대해 별개의 가변 청킹(variable-size chunking) 알고리즘을 사용할 수 있다.

[0067] 청크 추출부(430)는 N-way 청크들(chunks) 중 중복 제거에 사용되는 적어도 하나의 대상 청크(chunk)를 추출할 수 있다. 청크 추출부(430)는 청크 분할부(410)에 의해 분할된 고정 개수의 N-way 청크들(chunks) 중에서 특정 청크(chunk)를 선택하여 대상 청크(chunk)를 추출할 수 있다.

[0068] 일 실시예에서, 청크 추출부(430)는 제1, 제2 및 제3 청크(chunk) 중 제2 청크(chunk)를 중복 제거를 수행할 적어도 하나의 대상 청크(chunk)로서 결정할 수 있다. 예를 들어, 청크 추출부(430)는 청크 분할부(410)를 통해 3-way 청킹(chunking) 연산을 수행하여 생성된 제1, 제2 및 제3 청크(chunk) 중 제2 청크(chunk)를 적어도 하

나의 대상 청크(chunk)로 결정할 수 있다.

[0069] 일 실시예에서, 청크 추출부(430)는 N-way 청크들(chunks) 중 전방향 가변 청크(chunk)와 후방향 가변 청크(chunk)를 제외한 내부 청크(chunk)를 기초로 적어도 하나의 대상 청크(chunk)를 결정할 수 있다. 청크 추출부(430)는 특정 패킷의 처음과 끝 부분에서 시작하여 임의의 가변 청킹(variable-size chunking) 연산을 병렬적으로 수행할 수 있고, 상수시간 내에 처음과 마지막 N-way 청크(chunk)를 분할할 수 있다. 청크 추출부(430)는 처음과 마지막 N-way 청크(chunk)를 제외한 나머지 N-way 청크들(chunks)에 해당하는 내부 청크(chunk)를 기초로 적어도 하나의 대상 청크(chunk)를 결정할 수 있다. 예를 들어, 3-way 청킹(chunking) 연산을 수행하여 3개의 청크(chunk)로 분할한 경우 청크 추출부(430)는 가운데 청크(chunk)에 해당하는 제2 청크(chunk)를 대상 청크(chunk)로 결정할 수 있다.

[0070] 중복제거 처리부(450)는 청크 추출부(430)에 의해 추출된 적어도 하나의 대상 청크(chunk)를 기초로 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행할 수 있다. 일 실시예에서, 중복제거 처리부(450)는 적어도 하나의 대상 청크(chunk)를 중복 제거 해시 함수(hash function)의 입력으로 제공하여 특정 패킷의 중복 여부를 결정할 수 있다. 보다 구체적으로, 중복제거 처리부(450)는 대상 청크(chunk)를 중복 제거 해시 함수(hash function)의 입력으로 하여 대상 청크(chunk)에 대한 해시(hash) 값을 얻을 수 있다. 예를 들어, 중복제거 처리부(450)는 중복 제거 해시 함수(hash function)로 MD5 또는 SipHash를 사용할 수 있다. 여기에서, SipHash는 짧은 메시지와 패킷의 무결성을 검사하기 위해 고안된 고속 의사 난수 생성 함수(fast pseudo random number generating function)이다. 중복제거 처리부(450)는 대상 청크(chunk)에 대한 해시(hash) 값을 이용하여 중복 여부를 결정할 수 있다.

[0071] 일 실시예에서, 중복제거 처리부(450)는 중복 제거 해시 함수(hash function)의 출력으로 적어도 하나의 대상 청크(chunk)에 대한 핑거프린트(fingerprint)를 산출하고, 산출된 핑거프린트(fingerprint)를 중복 제거 해시 테이블(hash table)에서 검색하여 특정 패킷의 중복여부를 결정할 수 있다. 여기에서, 핑거프린트(fingerprint)는 대상청크를 해시함수의 입력으로 하여 산출된 해시 값에 해당할 수 있다 예를 들어, 중복제거 처리부(450)는 MD5 해시를 사용하여 대상 청크에 대한 128비트 크기의 핑거프린트(fingerprint)를 얻을 수 있다. 중복제거 처리부(450)는 대상 청크(chunk)에 대한 핑거프린트(fingerprint)가 중복 제거 해시 테이블(hash table) 내에 존재하는지를 검색하여 특정 패킷의 중복여부를 결정할 수 있다.

[0072] 중복제거 처리부(450)는 대상 청크(chunk)에 대한 핑거프린트(fingerprint)가 중복 제거 해시 테이블(hash table) 내에 존재하는 경우 특정 패킷이 중복된 것으로 결정할 수 있다. 중복제거 처리부(450)는 대상 청크(chunk)에 대한 핑거프린트(fingerprint)가 중복 제거 해시 테이블(hash table) 내에 존재하지 않는 경우 특정 패킷이 중복되지 않은 것으로 결정할 수 있고, 해당 패킷의 대상 청크(chunk)의 원본 데이터와 해시 값을 중복 제거 해시 테이블(hash table)에 저장할 수 있다.

[0073] 일 실시예에서, 중복제거 처리부(450)는 대상 청크(chunk)에 대한 핑거프린트(fingerprint)를 기초로 중복 제거 해시 테이블(hash table)에서의 충돌(collision) 발생 여부에 따라 특정 패킷의 중복 여부를 결정할 수 있다. 예를 들어, 중복제거 처리부(450)는 핑거프린트(fingerprint)가 중복 제거 해시 테이블(hash table) 내에서 충돌을 발생시키면 해당 청크(chunk)가 중복인 것으로 결정할 수 있다.

[0074] 일 실시예에서, 중복제거 처리부(450)는 특정 패킷이 중복인 경우에는 적어도 하나의 대상 청크(chunk)를 해시 테이블(hash table)의 중복 제거 해시 인덱스로 대체할 수 있다. 보다 구체적으로, 중복제거 처리부(450)는 특정 패킷이 중복인 경우에는 중복 제거 해시 테이블(hash table)에 저장된 대상 청크(chunk)와 동일한 데이터에 대한 중복 제거 해시 인덱스를 대상 청크(chunk)와 교체할 수 있다. 중복 제거 장치(330)는 특정 패킷의 대상 청크(chunk) 부분을 중복 제거 해시 인덱스(index)로 대체할 수 있고, 네트워크를 통해 패킷 수신 장치(350)에 전송할 수 있다.

[0076] 일 실시예에서, 중복제거 처리부(450)는 청킹(chunking) 연산, 중복 제거 해시 함수(hash function) 및 중복 제거 해시 테이블(hash table)의 여러 조합을 이용하여 특정 패킷의 중복 제거를 수행할 수 있다. 예를 들어, 중복제거 처리부(450) AE 청킹 알고리즘(Asymmetric Extremum content defined chunking algorithm), SipHash 및 충돌 허용 해시 테이블(collision tolerant hash table) 중 적어도 하나를 사용하여 대상 청크(chunk)에 대한 중복을 제거할 수 있다. 일 실시예에서, 중복제거 처리부(250)는 Rabin의 롤링 해시(rolling hash) 대신 AE 청킹(chunking) 알고리즘을 사용할 수 있고, MD5 대신 SipHash를 사용할 수 있다. 일 실시예에서, 중복제거 처리부(250)는 LL(Linked List) 해시 테이블(hash table) 대신 CT(Collision Tolerant) 해시 테이블(hash table)을 사용할 수 있다.

[0077] 제어부(470)는 패킷레벨 중복 제거 장치(330)의 전체적인 동작을 제어하고, 청크 분할부(410), 청크 추출부(430) 및 중복제거 처리부(450) 간의 제어 흐름 및 데이터 흐름을 관리할 수 있다.

[0079] 도 5는 도 3에 있는 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치에서 수행되는 패킷레벨(packet-level) 중복 제거 과정의 일 실시예를 설명하는 순서도이다.

[0080] 도 5를 참조하면, 패킷레벨 중복 제거 장치(330)는 청크 분할부(410)를 통해 특정 패킷에 관한 N-way 청킹(chunking) 연산을 수행하여 N-way 청크들(chunks)로 분할할 수 있다(단계 S510). 패킷레벨 중복 제거 장치(330)는 청크 추출부(430)를 통해 N-way 청크들(chunks) 중 중복 제거에 사용되는 적어도 하나의 대상 청크(chunk)를 추출할 수 있다(단계 S530). 패킷레벨 중복 제거 장치(330)는 중복제거 처리부(450)를 통해 적어도 하나의 대상 청크(chunk)를 기초로 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행할 수 있다(단계 S550).

[0082] 도 6은 본 발명의 일 실시예에 따른 경량 복잡도 기반의 패킷레벨(packet-level) 중복 제거 장치에서 수행되는 N-way 청킹(chunking) 연산을 설명하는 예시도이다.

[0083] 도 6을 참조하면, 슬라이딩 윈도우(sliding window)의 크기가 2인 경우 3-way 청킹(chunking)이 첫 번째 및 마지막 청크(chunk)를 찾는 방식을 나타낸다. a_0a_1 및 a_7a_8의 문자열은 첫 번째 및 마지막 청크(chunk)에 대해 두 개의 구분 기호를 동적으로 만듭니다. packet1과 packet2는 모두 동일한 중간 청크(chunk) c_2를 생성합니다. 따라서 3웨이 청킹(chunking)은 패킷 시작 부분에서 작은 바이트 수의 변경으로 인해 발생하는 경계 시프트(boundary shift) 문제를 해결할 수 있습니다.

[0085] 도 7은 고정 크기 청킹(fixed-size chunking), 가변 크기 청킹(variable-size chunking) 및 3-way 청킹(chunking)을 사용하여 수행한 비교 실험 결과를 보여주는 도면이다.

[0086] 도 7을 참조하면, 3-way 청킹(chunking)은 고정 크기 청킹(Fixed-size Chunking) 및 가변 크기 청킹(Variable-size Chunking)과 비교되었다. Rabin의 롤링 해시(rolling hash)는 청크(chunk) 사이의 구분자를 찾기 위해 체크섬(checksum)을 계산하는데 사용되었으며, MD5 해시는 임의의 길이의 메시지를 입력받아 128비트의 고정 길이를 가진 출력 값을 생성하고, 각 청크(chunk)의 지문(fingerprint)을 계산하는데 사용되었다. MD5코드는 OpenSSL v1.0.1.e에서 가져왔다. 연결리스트(Linked List)는 해시 테이블(hash table) 충돌을 관리하는데 사용되었다. 세 가지 청킹(chunking) 알고리즘을 각각 "가변 크기 Rabin + LL + MD5", "고정 크기 + LL + MD5"및 "3웨이 Rabin + LL + MD5"로 표시한다. 여기에서, LL은 연결리스트 해시 테이블(hash table)을 나타낸다.

[0087] 도 7에서, "가변 크기 Rabin + LL + MD5"와 "고정 크기 + LL + MD5"의 비교는 가변 크기 청킹(variable-size chunking)이 더 큰 중복을 제거하는 반면 고정 크기 청킹(fixed-size chunking)이 더 빠르게 실행된다. 3-way 청킹(chunking)이 고정 크기 청킹(fixed-size chunking)보다 빠르게 실행되고 DER이 가변 크기 청킹(variable-size chunking)의 DER만큼 높음을 확인할 수 있다. 여기에서, DER은 중복제거 비율에 해당하고 패킷에서 제거된 부분의 크기와 원본 패킷의 크기의 비율로 산출될 수 있다. 첫 번째 및 마지막 청크(chunk)의 크기가 약간 증가하므로 평균 청크(chunk) 크기에 따라 3-way 청킹(chunking)의 처리 속도가 약간 감소한다. 이 크기는 고정 크기 청킹(fixed-size chunking)에서 직접 결정될 수 있지만 평균 청크(chunk) 크기는 슬라이딩 윈도우(sliding window) 크기와 가변 크기 청킹(variable-size chunking)에서 구분 기호를 찾을 확률을 보정하여 조정할 수 있다. 3-way 청킹(chunking)과는 달리, "가변 크기 Rabin + LL + MD5" 및 "고정 크기 + LL + MD5"의 처리 속도는 크기가 커질수록 더 적은 수의 청크(chunk) 및 MD5 작업을 필요로 하기 때문에 청크(chunk) 크기가 커질수록 빨라진다.

[0089] 도 8은 청킹(chunking), 핑거프린팅(fingerprinting) 및 해시 테이블(hash table)에 관한 여러 조합을 사용하여 3-way 청킹(chunking) 기법을 수행한 비교 실험 결과를 보여주는 도면이다.

[0090] 도 8을 참조하면, 청킹(chunking), 핑거프린팅(fingerprinting) 및 해시 테이블(hash table) 생성의 최적 조합을 찾을 수 있다. 최첨단 기술조차도 CPU 코어 당 최대 처리량이 2Gbps에 불과하다. 이 실험에서는 패킷수준(packet-level) 중복 제거를 위한 최선의 방법을 찾을 수 있다. 또한 3-way 청킹(chunking)을 최적화하여 6Gbps 이상의 처리량을 달성할 수 있다. 이를 위해 청킹(chunking), 핑거프린팅(fingerprinting) 및 해시 테이블(hash table) 생성 각각에 대해 대체 알고리즘을 시도한다. Rabin의 롤링 해시(rolling hash)는 AE 청킹(chunking) 알고리즘으로 대체된다. 최근의 연구에서 MD5는 낮은 처리량으로 패킷레벨(packet-level) 핑거프린트(fingerprint)에 적합하지 않기 때문에 짧은 메시지와 패킷의 무결성을 검사하기 위해 최근에 고안된 고속 의사 난수 생성 함수인 SipHash로 대체했다. 또한 연결리스트 해시 테이블(Linked List hash table)을 충돌 허용

해시 테이블(Collision Tolerant hash table)로 대체했다.

[0091] 도 8에서, 4가지 다른 방식의 3-way 청킹(chunking)이 구현되었다. 첫 번째 버전은 Rabin의 롤링 해시(rolling hash)를 사용하여 세 개의 청크(chunk)를 얻을 수 있고, 연결리스트 해시 테이블(LL hash table) 및 MD5 핑거프린팅(fingerprinting)을 사용한다. 두 번째 버전은 AE 청킹(chunking), 연결리스트 해시 테이블(LL hash table) 및 MD5 핑거프린팅(fingerprinting)을 사용한다. 세 번째 버전은 AE 청킹(chunking), 연결리스트 해시 테이블(LL hash table) 및 SipHash를 사용한다. 마지막 버전은 AE 청킹(chunking), 충돌 허용 해시 테이블(CT hash table) 및 SipHash의 조합으로 구현된다. 이 네 가지 버전은 중복 제거 성능이 점진적으로 향상되는 방법을 보여준다.

[0093] 상기에서는 본 발명의 바람직한 실시예를 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구의 범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

부호의 설명

[0095] 300: 경량 복잡도 기반의 패킷레벨 중복 제거 시스템

310: 패킷 송신 장치 330: 중복 제거 장치

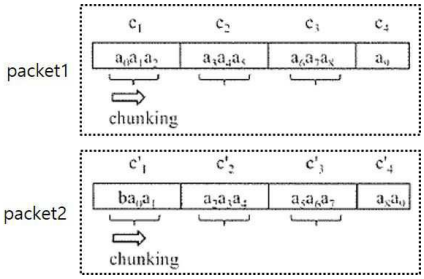
350: 패킷 수신 장치

410: 청크 분할부 430: 청크 추출부

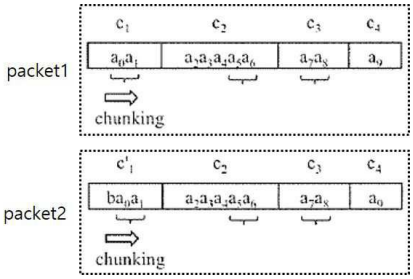
450: 중복제거 처리부 470: 제어부

도면

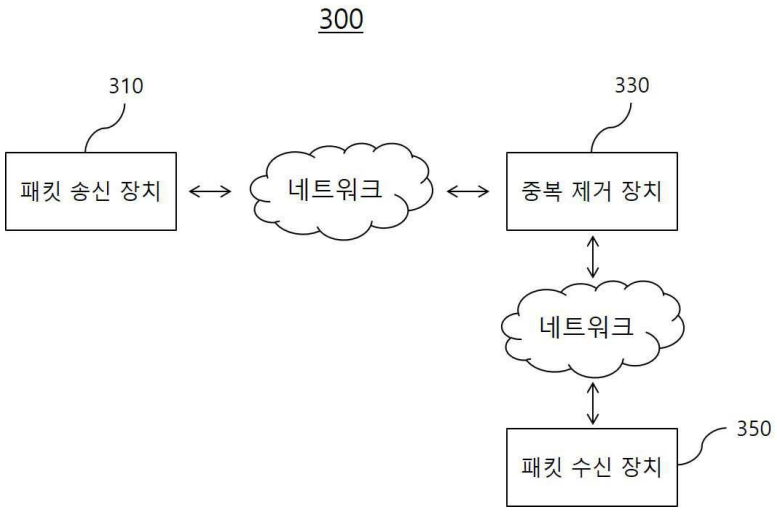
도면1



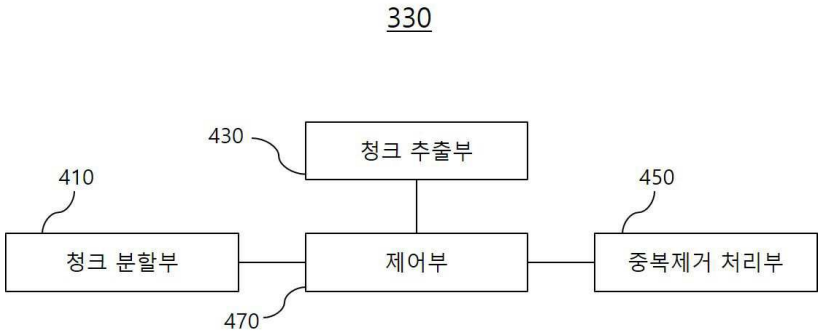
도면2



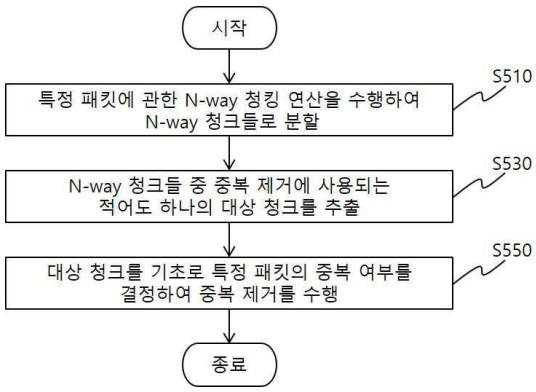
도면3


300
310
330
패킷 송신 장치
네트워크
중복 제거 장치
네트워크
350
패킷 수신 장치

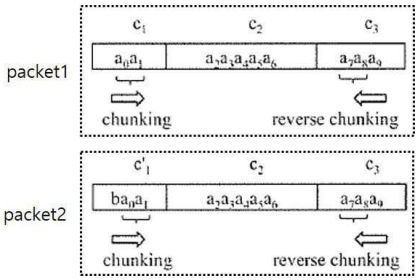
도면4


330
430
청크 추출부
410
450
청크 분할부
제어부
중복제거 처리부
470

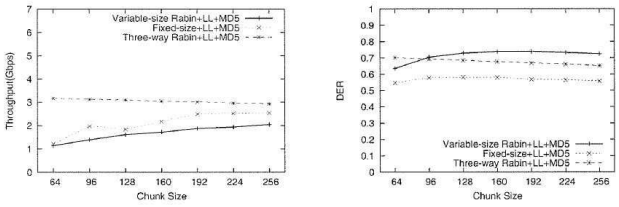
도면5


시작
특정 패킷에 관한 N-way 청킹 연산을 수행하여 N-way 청크들로 분할
S510
N-way 청크들 중 중복 제거에 사용되는 적어도 하나의 대상 청크를 추출
S530
대상 청크를 기초로 특정 패킷의 중복 여부를 결정하여 중복 제거를 수행
S550
종료

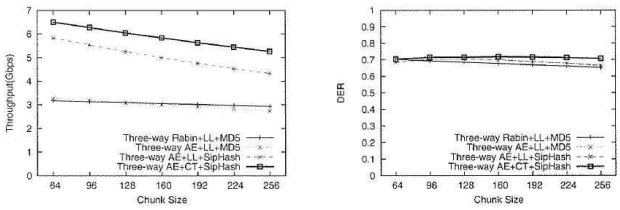
도면6


packet1
packet2
chunking
reverse chunking
chunking
reverse chunking

도면7


Variable-size Rabin+LL+MD5
Fixed-size+LL+MD5
Three-way Rabin+LL+MD5
Throughput(Gbps)
Chunk Size
DER
Chunk Size

도면8


Three-way Rabin+LL+MD5
Three-way AE+LL+MD5
Three-way AE+LL+SipHash
Three-way AE+CT+SipHash
Throughput(Gbps)
Chunk Size
DER
Chunk Size