(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0071832
(43) 공개일자 2016년06월22일

(51) 국제특허분류(Int. Cl.)
H04W 28/10 (2009.01) *H04W 72/10* (2009.01)
H04W 88/02 (2009.01)
(21) 출원번호 10-2014-0179500
(22) 출원일자 2014년12월12일
심사청구일자 없음

(71) 출원인
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
이웅희
서울특별시 송파구 토성로3길 21, 101호
김황남
서울특별시 성동구 행당로8길 8, 104동 602호
(뒷면에 계속)
(74) 대리인
윤동열

전체 청구항 수 : 총 26 항

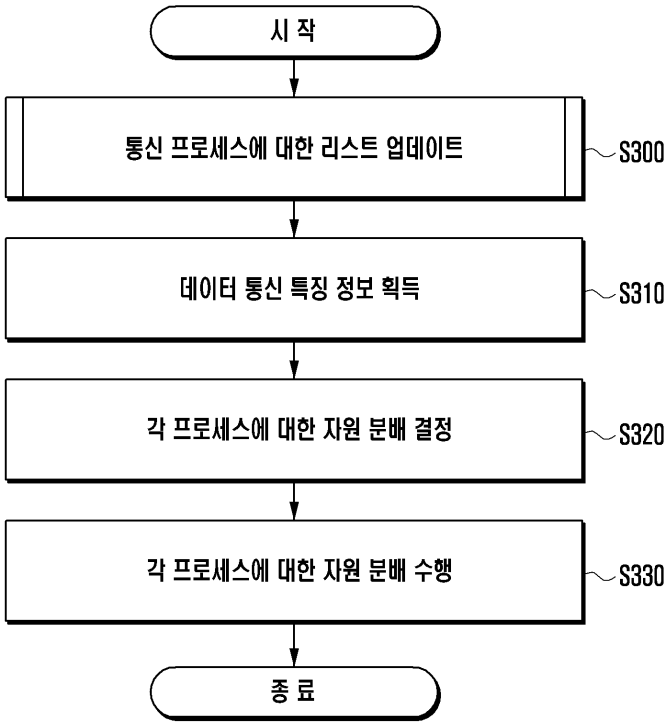
(54) 발명의 명칭 **무선 통신 시스템에서 자원 분배 방법 및 장치**

(57) 요 약

본 발명은 무선 통신 시스템에서 자원 분배 방법에 대한 발명으로서, 보다 구체적으로 단말에서 사용자에 의해 구동되는 프로세스 별로 통신 자원을 분배하는 방법 및 장치에 관한 발명이다. 본 발명의 일 실시 예에 따르는 무선 통신 시스템에서 단말의 자원 분배 방법은, 적어도 하나 이상의 프로세스에 관련된 정보에 기반하여, 구동

(뒷면에 계속)

대 표 도 - 도3



프로세스에 대한 목록을 업데이트하는 단계; 상기 프로세스에 관련된 정보에 포함된 우선 순위 정보에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 프로세스 별 자원 량을 결정하는 단계; 및 상기 결정된 자원 량에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 적어도 하나 이상의 프로세스에 자원 분배를 수행하는 단계;를 포함한다. 본 발명의 실시 예에 따르면, 단말의 각 프로세스의 단위 시간 당 데이터 처리량을 조정하여, 우선순위가 높은 프로세스의 통신 품질을 보장함으로써 사용자의 체감 통신 품질(quality of experience)을 높일 수 있다.

(72) 발명자

이철호

경기도 수원시 영통구 영통로 232, 821동 1505호

김민

서울특별시 금천구 독산로68가길 20, 나동 203호

박용석

서울특별시 서초구 신반포로 133 201동 1001호

김현순

경기도 성남시 분당구 양현로166번길 20, 908동 702호

이준엽

서울특별시 서초구 사임당로10길 39, 103동 403호

명 세 서

청구범위

청구항 1

무선 통신 시스템에서 단말의 자원 분배 방법에 있어서,

적어도 하나 이상의 프로세스에 관련된 정보에 기반하여, 구동 프로세스에 대한 목록을 업데이트하는 단계;

상기 프로세스에 관련된 정보에 포함된 우선 순위 정보에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 프로세스 별 자원 량을 결정하는 단계; 및

상기 결정된 자원 량에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 적어도 하나 이상의 프로세스에 자원 분배를 수행하는 단계;를 포함하는 것을 특징으로 하는 방법.

청구항 2

제1항에 있어서, 상기 자원 분배를 수행하는 단계는,

각 프로세스에 대한 버퍼 크기 또는 알림(advertised) 윈도우 크기 중 적어도 하나를 조정하는 것을 특징으로 하는 방법.

청구항 3

제2항에 있어서, 상기 알림(advertised) 윈도우는,

상기 단말이 정보 송신 장치에게 알리는 수신 가능한 데이터 양의 정보를 포함하는 것을 특징으로 하는 방법.

청구항 4

제2항에 있어서, 상기 자원 분배를 수행하는 단계는,

정보 송신 장치에게 전송하는 ACK(acknowledge) 메시지의 개수를 조정하는 것을 특징으로 하는 방법.

청구항 5

제1항에 있어서, 상기 프로세스에 관련된 정보는,

프로세스 식별자, 상기 프로세스의 프로토콜 정보 등을 포함하는 것을 특징으로 하는 방법.

청구항 6

제4항에 있어서, 상기 ACK(acknowledge) 메시지의 개수를 조정하는 것은 상기 프로세스에 대한 목록에 포함된 프로세스 별 우선 순위 정보에 기반한 것을 특징으로 하는 방법.

청구항 7

제2항에 있어서, 상기 자원 분배를 수행하는 단계는,

프로세스에 할당된 자원 량을 증가시키는 경우, 버퍼 크기를 수신 윈도우 크기보다 선행하여 증가시키는 단계;

프로세스에 할당된 자원 량을 감소시키는 경우, 수신 윈도우 크기를 버퍼 크기보다 선행하여 감소시키는 단계;를 포함하는 것을 특징으로 하는 방법.

청구항 8

제1항에 있어서, 상기 미리 저장된 프로세스 별 자원 필요량은,

프로세스의 데이터 통신 양의 평균 값 또는 데이터 통신 패턴 중 어느 하나에 기반하여 결정되는 것을 특징으로 하는 방법.

청구항 9

제1항에 있어서, 상기 각 프로세스 별 우선 순위는,

현재 구동되는 프로세스인지 여부에 따라 결정되는 것을 특징으로 하는 방법.

청구항 10

제1항에 있어서, 상기 프로세스 별 분배할 자원 량을 결정하는 단계는,

상기 프로세스에 대한 목록에 포함된 우선 순위가 가장 큰 프로세스의 미리 저장된 자원 필요량을 결정하는 단계;

상기 결정된 자원 필요량보다 가용 자원이 더 많은 경우, 상기 우선 순위가 가장 큰 프로세스에 자원 필요량에 대응되는 자원 량을 할당하는 단계;

상기 프로세스에 대한 목록에서 상기 우선 순위가 가장 큰 프로세스를 제외한 나머지 프로세스에 상기 자원 필요량에 기반하여 가중치 정보를 결정하는 단계;

상기 우선 순위가 가장 큰 프로세스를 제외한 나머지 프로세스에 상기 자원 필요량 및 가중치 정보에 기반하여 자원 량을 할당하는 단계;를 포함하는 것을 특징으로 하는 방법.

청구항 11

제10항에 있어서, 상기 프로세스 별 자원 량을 결정하는 단계는,

상기 결정된 우선 순위가 가장 큰 프로세스의 미리 저장된 자원 필요량보다 가용 자원이 더 적은 경우, 상기 우선 순위가 가장 큰 프로세스를 제외한 나머지 프로세스에 단위 패킷을 수신하는데 필요한 자원 량을 할당하는 단계;

상기 가용 자원에서 우선 순위가 가장 큰 프로세스를 제외한 나머지 프로세스에 할당된 자원 량을 제외한 자원 량을 상기 우선 순위가 가장 큰 프로세스에 할당하는 단계;를 포함하는 것을 특징으로 하는 방법.

청구항 12

제5항에 있어서, 상기 프로세스에 대한 목록을 업데이트하는 단계는,

생성된 프로세스와 관련된 정보에 포함된 식별자를 추출하는 단계;

상기 추출한 식별자를 이용하여 상기 프로세스가 프로세스에 대한 목록에 포함되는지 결정하는 단계;

상기 프로세스 목록에 포함되어 있지 않은 경우, 상기 프로세스를 상기 프로세스에 대한 목록에 저장하는 단계;를 포함하는 것을 특징으로 하는 방법.

청구항 13

제1항에 있어서, 상기 프로세스 별 자원 량을 결정하는 단계는,

미리 저장된 프로세스 별 자원 필요량 및 상기 우선 순위 정보에 기반한 것을 특징으로 하는 방법.

청구항 14

무선 통신 시스템에서 자원을 분배하는 단말에 있어서,

정보 송수신 장치와 신호를 송수신하는 통신부;

적어도 하나 이상의 프로세스에 관련된 정보에 기반하여, 구동 프로세스에 대한 목록을 업데이트하고, 상기 프로세스에 관련된 정보에 포함된 우선 순위 정보에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 프로세스 별 자원 량을 결정하고, 상기 결정된 자원 량에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 적어도 하나 이상의 프로세스에 자원 분배를 수행하는 것을 제어하는 제어부;를 포함하는 것을 특징으로 하는 단말.

청구항 15

제14항에 있어서, 상기 자원 분배를 수행하는 것은

각 프로세스에 대한 버퍼 크기 또는 알림(advertised) 윈도우 크기 중 적어도 하나를 조정하는 것을 특징으로 하는 단말.

청구항 16

제15항에 있어서, 상기 알림(advertised) 윈도우는,

상기 단말이 정보 송신 장치에게 알리는 수신 가능한 데이터 양의 정보를 포함하는 것을 특징으로 하는 단말.

청구항 17

제15항에 있어서, 상기 자원 분배를 수행하는 것은,

정보 송신 장치에게 전송하는 ACK(acknowledge) 메시지의 개수를 조정하는 것을 특징으로 하는 단말.

청구항 18

제14항에 있어서, 상기 프로세스에 관련된 정보는,

프로세스 식별자, 상기 프로세스의 프로토콜 정보 등을 포함하는 것을 특징으로 하는 방법.

청구항 19

제17항에 있어서, 상기 ACK(acknowledge) 메시지의 개수를 조정하는 것은 상기 프로세스에 대한 목록에 포함된 프로세스 별 우선 순위 정보에 기반한 것을 특징으로 하는 단말.

청구항 20

제15항에 있어서, 상기 자원 분배를 수행하는 것은,

프로세스에 할당된 자원 량을 증가시키는 경우, 버퍼 크기를 수신 윈도우 크기보다 선행하여 증가시키고, 프로세스에 할당된 자원 량을 감소시키는 경우, 수신 윈도우 크기를 버퍼 크기보다 선행하여 감소시키는 것을 포함하는 것을 특징으로 하는 단말.

청구항 21

제14항에 있어서, 상기 미리 저장된 프로세스 별 자원 필요량은,

프로세스의 데이터 통신 양의 평균 값 또는 데이터 통신 패턴 중 적어도 어느 하나에 기반하여 결정되는 것을 특징으로 하는 단말.

청구항 22

제14항에 있어서, 상기 각 프로세스 별 우선 순위는,

현재 구동되는 프로세스인지 여부에 따라 결정되는 것을 특징으로 하는 단말.

청구항 23

제14항에 있어서, 상기 프로세스 별 분배할 자원 량을 결정하는 것은,

상기 프로세스에 대한 목록에 포함된 우선 순위가 가장 큰 프로세스의 미리 저장된 자원 필요량을 결정하고, 상기 결정된 자원 필요량보다 가용 자원이 더 많은 경우, 상기 우선 순위가 가장 큰 프로세스에 자원 필요량에 대응되는 자원 량을 할당하고, 상기 프로세스에 대한 목록에서 상기 우선 순위가 가장 큰 프로세스를 제외한 나머지 프로세스에 상기 자원 필요량에 기반하여 가중치 정보를 결정하고, 상기 우선 순위가 가장 큰 프로세스를 제외한 나머지 프로세스에 상기 자원 필요량 및 가중치 정보에 기반하여 자원 량을 할당하는 것을 포함하는 것을 특징으로 하는 단말.

청구항 24

제23항에 있어서, 상기 프로세스 별 자원 량을 결정하는 것은,

상기 결정된 우선 순위가 가장 큰 프로세스의 미리 저장된 자원 필요량보다 가용 자원이 더 적은 경우, 상기 우선 순위가 가장 큰 프로세스를 제외한 나머지 프로세스에 단위 패킷을 수신하는데 필요한 자원 량을 할당하고, 상기 가용 자원에서 우선 순위가 가장 큰 프로세스를 제외한 나머지 프로세스에 할당된 자원 량을 제외한 자원 량을 상기 우선 순위가 가장 큰 프로세스에 할당하는 것을 포함하는 것을 특징으로 하는 단말.

청구항 25

제18항에 있어서, 상기 프로세스에 대한 목록을 업데이트하는 것은,

생성된 프로세스의 식별자 및 데이터 통신 패턴 정보를 추출하고, 상기 추출한 식별자를 이용하여 상기 프로세스가 프로세스에 대한 목록에 포함되는지 결정하고, 상기 프로세스 목록에 포함되어 있지 않은 경우, 상기 프로세스를 상기 프로세스에 대한 목록에 저장하는 것을 포함하는 것을 특징으로 하는 단말.

청구항 26

제14항에 있어서, 상기 프로세스 별 자원 량을 결정하는 것은,

미리 저장된 프로세스 별 자원 필요량 및 상기 우선 순위 정보에 기반한 것을 특징으로 하는 단말.

발명의 설명

기 술 분 야

[0001] 본 발명은 무선 통신 시스템에서 자원 분배 방법에 대한 발명으로서, 보다 구체적으로 단말에서 사용자에 의해 구동되는 프로세스 별로 통신 자원을 분배하는 방법 및 장치에 관한 발명이다.

배 경 기 술

[0002] 현대의 무선 통신 기기의 통신 성향을 보면 각종 알림, 백업, 데이터 전송 등등의 작업을 동시에 수행하고 있다. 일반적인 뉴스, 날씨 정보를 받아오면서 웹서핑을 하거나, 소프트웨어 업데이트를 하면서 동영상을 시청한다거나 하는 식으로 다중 통신이 활발해 지고 있다. 이러한 상황에서 지금처럼 각 연결마다 차별 없는 통신방식을 사용하면, 일부 통신 대역폭이 충분하지 않거나, 기기의 성능이 통신 량을 따라가지 못하는 경우에 사용자가 좋은 품질의 서비스를 이용할 수 없게 된다.

[0003] 실시간 서비스를 제외한 대부분의 인터넷 서비스는 burst traffic의 성향을 가지고 있으므로, 두 가지 이상의 서비스가 동시에 발생할 경우, 상대적으로 속도가 낮아질 수밖에 없으며, 이러한 현상이 자주 발생하게 되면 서비스를 이용중인 사용자가 서비스 품질의 저하를 직접적으로 체감하게 된다. 이러한 자원이 부족한 경우에는 효과적으로 전송자원을 분배함으로써 적어도 사용자가 현재 사용중인 어플리케이션의 전송속도를 보장할 수 있어야 하지만, 기본적인 통신 규약에는 이러한 우선순위를 고려하는 부분이 포함되어 있지 않다.

[0004] 또한 종래의 기술들은 전송 품질 향상을 위해 전송속도를 늘리는 방식은 고려하지 않았으므로, 우선순위가 높은 어플리케이션이 충분한 throughput(단위 시간당 데이터 처리량)을 확보하지 않았을 때와 같은 상황에서 직접적으로 그 어플리케이션을 조정할 수 없다. Throughput 증가 효과는 최소한의 통신자원을 확보하는데 사용할 수 있다는 점에서 중요하다. 예를 들어, 인터넷 전화의 경우 일시적으로라도 통화가 끊어지면 안되므로 지속적인 통신이 필요한데, 이러한 지속적인 통신 유지를 위한 최소자원을 확보하기 위한 우선순위에 따른 throughput 증가 기술은 현재 적용되어있지 않으며 결국 한정된 throughput을 우선순위에 관계없이 단순히 속도가 빠른 어플리케이션이 점유하게 되어 전송속도의 불균형을 심화시키게 된다.

[0005] 뿐만 아니라 각 소켓마다 기본 할당량의 메모리가 제공되는데, 이때 각각의 소켓에 모두 기본버퍼가 제공되어 버리면 상당한 용량의 메모리를 점유하게 된다. 스마트폰 뿐만 아니라 웨어러블 디바이스와 사물인터넷(internet of things, IoT) 기기들과 같이 상대적으로 작은 메모리를 가지는 기기의 경우 효율적인 메모리 관리가 이루어지지 못하고 있었다.

[0006] 도 1은 각 프로세스에 따른 평균 버퍼 사용률을 예시한 도면이다.

[0007] 상기 도 1에서, 각 프로세스는 기본값으로 할당되는 버퍼 크기를 의미하는 '기본 버퍼 크기'에 비해 상당히 낮은 양의 버퍼를 사용하고 있음을 볼 수 있고, 현재 시스템에서 버퍼가 낭비되고 있는 문제점을 보여준다.

발명의 내용

해결하려는 과제

[0008] 본 발명은 상술한 문제점을 해결하기 위하여 제안된 것으로, 보다 구체적으로 상기 무선 통신 시스템에서 자원을 분배하는 방법 및 장치는 단말 내의 각 프로세스의 우선 순위를 고려하여 차별적으로 단위 시간 당 데이터 처리량(throughput)을 조정하는 방법 및 장치를 제안하고자 한다.

과제의 해결 수단

[0009] 상술한 과제를 달성하기 위하여, 본 발명의 일 실시 예에 따르는 무선 통신 시스템에서 단말의 자원 분배 방법은, 적어도 하나 이상의 프로세스에 관련된 정보에 기반하여, 구동 프로세스에 대한 목록을 업데이트하는 단계; 상기 프로세스에 관련된 정보에 포함된 우선 순위 정보에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 프로세스 별 자원 량을 결정하는 단계; 및 상기 결정된 자원 량에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 적어도 하나 이상의 프로세스에 자원 분배를 수행하는 단계;를 포함한다.

[0010] 또한 상기와 같은 문제점을 해결하기 위한 본 발명의 일 실시 예에 따르는 무선 통신 시스템에서 자원을 분배하는 단말은, 정보 송수신 장치와 신호를 송수신하는 통신부; 적어도 하나 이상의 프로세스에 관련된 정보에 기반하여, 구동 프로세스에 대한 목록을 업데이트하고, 상기 프로세스에 관련된 정보에 포함된 우선 순위 정보에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 프로세스 별 자원 량을 결정하고, 상기 결정된 자원 량에 기반하여 상기 구동 프로세스에 대한 목록에 포함된 적어도 하나 이상의 프로세스에 자원 분배를 수행하는 것을 제어하는 제어부;를 포함한다.

발명의 효과

[0011] 본 발명의 실시 예에 따르면, 단말의 각 프로세스의 단위 시간 당 데이터 처리량을 조정하여, 우선순위가 높은 프로세스의 통신 품질을 보장함으로써 사용자의 체감 통신 품질(quality of experience)을 높일 수 있다. 또한, 상기 단위 시간 당 데이터 처리량을 조정함에 있어 버퍼, 알림(advertised) 윈도우 크기 및 지연된 ACK 최대 개수(delayed acknowledge max count)를 조정하여 우선순위가 높은 프로세스의 전송속도를 향상시키며 동시에 과도한 메모리사용을 방지할 수 있다.

도면의 간단한 설명

[0012] 도 1은 각 프로세스에 따른 평균 버퍼 사용률을 예시한 도면이다.

도 2는 본 발명의 실시 예에 따른 단말의 내부 구조를 도시하는 블록도이다.

도 3은 본 발명의 실시 예에 따른 단말이 구동되는 각 프로세스에 자원을 분배하는 과정을 도시한 순서도이다.

도 4는 본 발명의 실시 예에 따른 기존의 소켓 구조체에 추가되는 각 프로세스에 대한 정보를 포함하는 새로운 구조체의 구성을 예시하는 도면이다.

도 5는 본 발명의 실시 예에 따른 도 3의 S300단계를 구체화한 도면이다.

도 6은 본 발명의 실시 예에 따른 구동되는 프로세스에 대응하는 어플리케이션들에 대한 자원 분배를 예시한 도면이다.

도 7은 본 발명의 실시 예에 따른 각 프로세스별로 단위 시간당 데이터 처리량을 조정한 것을 설명하는 도면이다.

도 8은 본 발명의 실시 예에 따른 단말이 각 프로세스별로 자원 분배를 결정하는 것을 설명하는 도면이다.

도 9는 본 발명의 실시 예에 따른 버퍼 크기 조정에 따른 단위 시간당 데이터 처리량의 변화를 예시하는 도면이다.

도 10은 본 발명의 실시 예에 따른 지연된 ACK 최대 개수(delayed ACK max count)의 조정에 따른 단위 시간당 데이터 처리량의 변화를 예시하는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0013] 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0014] 본 명세서에서 실시 예를 설명함에 있어서 본 발명이 속하는 기술 분야에 익히 알려져 있고 본 발명과 직접적으로 관련이 없는 기술 내용에 대해서는 설명을 생략한다. 이는 불필요한 설명을 생략함으로써 본 발명의 요지를 흐리지 않고 더욱 명확히 전달하기 위함이다.

[0015] 마찬가지 이유로 첨부 도면에 있어서 일부 구성요소는 과장되거나 생략되거나 개략적으로 도시되었다. 또한, 각 구성요소의 크기는 실제 크기를 전적으로 반영하는 것이 아니다. 각 도면에서 동일한 또는 대응하는 구성요소에는 동일한 참조 번호를 부여하였다.

[0016] 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술되어 있는 실시 예들을 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시 예들에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있으며, 단지 본 실시 예들은 본 발명의 개시가 완전하도록 하고, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이며, 본 발명은 청구항의 범주에 의해 정의될 뿐이다. 명세서 전체에 걸쳐 동일 참조 부호는 동일 구성 요소를 지칭한다.

[0017] 이 때, 처리 흐름도 도면들의 각 블록과 흐름도 도면들의 조합들은 컴퓨터 프로그램 인스트럭션들에 의해 수행될 수 있음을 이해할 수 있을 것이다. 이들 컴퓨터 프로그램 인스트럭션들은 범용 컴퓨터, 특수용 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비의 프로세서에 탑재될 수 있으므로, 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비의 프로세서를 통해 수행되는 그 인스트럭션들이 흐름도 블록(들)에서 설명된 기능들을 수행하는 수단을 생성하게 된다. 이들 컴퓨터 프로그램 인스트럭션들은 특정 방식으로 기능을 구현하기 위해 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비를 지향할 수 있는 컴퓨터 이용 가능 또는 컴퓨터 판독 가능 메모리에 저장되는 것도 가능하므로, 그 컴퓨터 이용가능 또는 컴퓨터 판독 가능 메모리에 저장된 인스트럭션들은 흐름도 블록(들)에서 설명된 기능을 수행하는 인스트럭션 수단을 내포하는 제조 품목을 생산하는 것도 가능하다. 컴퓨터 프로그램 인스트럭션들은 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비 상에 탑재되는 것도 가능하므로, 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비 상에서 일련의 동작 단계들이 수행되어 컴퓨터로 실행되는 프로세스를 생성해서 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비를 수행하는 인스트럭션들은 흐름도 블록(들)에서 설명된 기능들을 실행하기 위한 단계들을 제공하는 것도 가능하다.

[0018] 또한, 각 블록은 특정된 논리적 기능(들)을 실행하기 위한 하나 이상의 실행 가능한 인스트럭션들을 포함하는 모듈, 세그먼트 또는 코드의 일부를 나타낼 수 있다. 또, 몇 가지 대체 실행 예들에서는 블록들에서 언급된 기능들이 순서를 벗어나서 발생하는 것도 가능함을 주목해야 한다. 예컨대, 잇달아 도시되어 있는 두 개의 블록들은 사실 실질적으로 동시에 수행되는 것도 가능하고 또는 그 블록들이 때때로 해당하는 기능에 따라 역순으로 수행되는 것도 가능하다.

[0019] 이 때, 본 실시 예에서 사용되는 '~부'라는 용어는 소프트웨어 또는 FPGA또는 ASIC과 같은 하드웨어 구성요소를 의미하며, '~부'는 어떤 역할들을 수행한다. 그렇지만 '~부'는 소프트웨어 또는 하드웨어에 한정되는 의미는 아니다. '~부'는 어드레싱할 수 있는 저장 매체에 있도록 구성될 수도 있고 하나 또는 그 이상의 프로세서들을 재생시키도록 구성될 수도 있다. 따라서, 일 예로서 '~부'는 소프트웨어 구성요소들, 객체지향 소프트웨어 구성요소들, 클래스 구성요소들 및 태스크 구성요소들과 같은 구성요소들과, 프로세스들, 함수들, 속성들, 프로시저들, 서브루틴들, 프로그램 코드의 세그먼트들, 드라이버들, 펌웨어, 마이크로코드, 회로, 데이터, 데이터베이스, 데이터 구조들, 테이블들, 어레이들, 및 변수들을 포함한다. 구성요소들과 '~부'들 안에서 제공되는 기능은 더 작은 수의 구성요소들 및 '~부'들로 결합되거나 추가적인 구성요소들과 '~부'들로 더 분리될 수 있다. 뿐만 아니라, 구성요소들 및 '~부'들은 디바이스 또는 보안 멀티미디어카드 내의 하나 또는 그 이상의 CPU들을 재생시키도록 구현될 수도 있다.

[0020] 전송 제어 프로토콜(transmission control protocol, TCP)은 신뢰성이 보장되는 전송 프로토콜로서, 패킷을 보낼 때마다 전송한 데이터에 대해서 수신자로부터 acknowledge(ACK)를 받음으로써 현재 데이터가 제대로 전송되었는지를 판단하는 전송 방식이다. 상기 전송 제어 프로토콜은 상기와 같은 과정을 통하여 잃어버린 패킷을 재전송을 통해 보충하며, 시작과 끝을 서로 알 수 있기 때문에 전송하기로 한 데이터의 확실한 전달을 보장한다. 또한 알림(advertised) 윈도우 크기의 증감을 통해서 회선의 상태나 통신 품질에 따라서 전송량을 늘이거나 줄일 수 있기 때문에 상황에 유동적인 대응도 가능하다.

[0021] 사용자 데이터 그램 프로토콜(user datagram protocol, UDP)는 Datagram이라 알려진 단위로 정보를 전송하는 프로토콜로서, 파일의 전달여부나 신뢰성은 보장되지 않는 단순히 데이터를 전송하는 방식이다. 따라서 송신자가 얼만큼 보낼지, 보낸 데이터를 받아지는지도 알 수 없고, 수신자 입장에서는 데이터그램 도착 순서가 바뀌거나, 누락되기도 한다. 하지만 해당 부분은 어플리케이션 단계에서 데이터 누락 여부를 검증하여 정보를 송신자에게 알려주도록 조작할 수도 있으며, 구조가 단순한 만큼 네트워크 부하가 적게 걸리고, 속도가 빠르다는 장점이 있다.

[0022] 버퍼는 통신을 할 때 전송하거나, 전송 받은 데이터가 일시적으로 저장되는 공간을 의미한다. TCP 버퍼 튜닝(buffer tuning)은 각 연결에 필요한 TCP 버퍼 크기를 결정하여 조정하는 기법으로 네트워크 관리자가 직접 튜닝하는 수동튜닝과, 관리자의 개입 없이 튜닝 데몬이나 TCP의 튜닝 알고리즘을 통해 동작하는 자동 튜닝으로 분류된다. 또한 자동튜닝은 튜닝 시점에 따라 연결 설정 시에만 튜닝을 실시하는 정적 튜닝(static tuning)과 네트워크 상황의 변화에 따라 지속적으로 동작하는 동적 튜닝(dynamic tuning)으로 나누어진다. 이러한 튜닝은 TCP의 통신에 필요한 윈도우의 크기를 만족시킬 수 있도록 충분한 양을 할당하거나, 줄임으로써 들어오는 패킷의 크기를 조정하는 기법을 말한다.

[0023] 윈도우는 송신자가 네트워크로 내보내는 데이터 양을 조정하기 위한 개념을 말하며, 통신상태에 따라 윈도우의 크기가 변동하게 된다. 통신을 위해서 소켓이 열리게 되면, 해당 소켓에 기본적인 버퍼량을 할당하게 되며, 그 이후 min, max값의 범위 내에서 필요한 만큼을 계산해서 새로운 버퍼량을 할당한다. 수신자의 수신 윈도우는 TCP에서 수신자가 한번에 받을 수 있는 최대 크기를 의미하며, 상기 수신 윈도우 크기를 송신자에게 알림(advertisement)으로써 수신자가 받을 수 있는 데이터 양을 송신자에게 알리게 되는데 이 때의 데이터 크기를 알림 윈도우(advertised window)라고 한다. 상기 버퍼는 상기 수신 윈도우의 상한선을 의미하게 된다.

[0024] 프로세스는 컴퓨터 분야에서 실행중인 프로그램을 의미한다. 상기 프로그램은어플리케이션을 포함할 수 있으며, 본 발명에서 프로그램과 어플리케이션이라는 용어는 혼용해서 사용할 수 있다. 소켓은 상기 프로세스가 데이터 송신 장치와 통신하기 위하여 생성되는 것을 의미한다. 상기 소켓은 상기 데이터 송신 장치와 통신하기 위한 채널을 포함할 수 있다. 상기 프로세스에 대응하는 소켓의 구조체에는 상기 프로세스에 관련된 정보들이 포함될 수 있다.

[0025] 도 2 는 본 발명의 실시 예에 따른 단말의 내부 구조를 도시하는 블록도이다.

[0026] 도 2를 참고하면, 자원을 분배하는 단말은 통신부(200), 및 제어부(210)로 구성되어 있으며, 상기 제어부(210)는 소켓 제거부(220), 네트워크 사용정보 추출부(225), 소켓 정보 초기화부(227), 통신 프로세스 관리부(230), 소켓 윈도우 관리부(240), 소켓 버퍼 관리부(245)를 포함할 수 있다. 상기 제어부의 구성은 본 발명의 실시 예에 불과하며 상기 구성 이외에 본 발명의 목적을 달성할 수 있는 구성을 포함할 수 있다. 또한 상기 제어부(210)에 포함된 구성들이 수행하는 동작은 모두 상기 제어부(210)가 수행할 수 있다.

[0027] 상기 통신부(200)는 송신 장치와 필요한 신호를 전송 또는 수신할 수 있다. 상기 통신부(200)는 무선 통신부를 포함할 수 있다. 상기 통신부(200)는 상기 송신 장치로부터 데이터의 전송 시에 상기 데이터의 전송을 확인하는 ACK을 상기 송신 장치에게 전송할 수 있다.

[0028] 상기 제어부(210)에 포함된 소켓 정보 초기화부(227)는 소켓이 생성될 때 소켓을 분석하고 본 발명에서 사용하는 기본정보를 소켓 구조체 안에 입력할 수 있다. 상기 소켓 정보 초기화부(227)는 상기 프로세스에 대한 정보를 쉽게 관리할 수 있도록 새로운 구조체를 만들어 기존 소켓 구조체 안에 저장할 수 있다.

[0029] 상기 제어부(210)에 포함된 네트워크 사용 정보 추출부는 시스템의 데이터 통신 패턴을 계속해서 추적하면서 각 프로세스의 데이터 통신 양과 패턴을 기록하게 된다. 네트워크 사용정보 추출부(225)에 의해 기록된 프로세스의 데이터 통신 특징 정보는 시스템이 종료와 함께 삭제되는 정보가 아니라 시스템이 종료되더라도 계속해서 남아있게 되며, 사용자가 프로세스를 사용할수록 더 정확한 데이터 통신 특징 정보를 추출할 수 있게 된다. 이렇게 추출된 프로세스의 데이터 통신 특징 정보는 통신 프로세스 관리부(230)로 전달되게 된다.

[0030] 상기 제어부(210)에 포함된 소켓 제거부(220)는 커널에서 소켓이 삭제되는 부분에 위치하여 소켓 정보 초기화부(227)를 거쳐 생성된 소켓이 시스템에서 제거 될 경우 제거되는 소켓을 확인하고 이 정보를 통신 프로세스 관리부(230)에게 전달하는 역할을 한다. 그렇게 함으로써 통신 프로세스 관리부(230)가 가지고 있는 통신 프로세스 목록이 항상 최신의 정보를 가질 수 있도록 하며, 통신 프로세스 관리부(230)가 통신 프로세스들의 자원을 관리하고 할당하는데 오판을 하지 않도록 한다.

[0031] 상기 제어부(210)에 포함된 통신 프로세스 관리부(230)는 소켓 정보 초기화부(227), 소켓 제거부(220), 네트워크 사용 정보 추출부(225)로부터 제공받은 정보를 활용하여 시스템에서 데이터 통신을 실행중인 사용자 프로세스를 모니터링하고 적절하게 자원을 분배하는 역할을 한다. 또한 상기 통신 프로세스 관리부(230)는 자원 분배를 결정한 후, 각 프로세스별 자원을 지속적으로 조정하며, 각 소켓의 delayed ACK max count 값을 설정한다. 통신 프로세스 관리부(230)에 의해 결정된 자원 분배 계획은 소켓 윈도우 관리부(240)와 소켓 버퍼 관리부(245)에게 전달되게 된다.

[0032] 상기 제어부(210)에 포함된 소켓 윈도우 관리부(240)와 소켓 버퍼 관리부(245)는 통신 프로세스 관리부(230)에 의해 전달받은 자원 분배 계획에 따라 해당 소켓의 윈도우 크기와 소켓 버퍼 크기를 각각 조정하게 된다. 본 발명에서, 소켓에 자원을 좀더 할당할 경우 소켓 버퍼 관리부(245)에 의해 버퍼가 먼저 조정된 이후 소켓 윈도우 관리부(240)에 의해 윈도우 크기가 조정되고, 반대로 소켓에 할당된 자원을 줄일 경우 소켓 윈도우 관리부(240)에 의해 먼저 윈도우 크기가 조정된 이후 소켓 버퍼 관리부(245)에 의해 버퍼가 조정되게 된다. 이처럼 상황에 따른 작업 흐름의 변경을 통해 수신 받은 데이터가 유실되는 문제를 방지하게 된다.

[0033] 도 3은 본 발명의 실시 예에 따른 단말의 제어부가 구동되는 각 프로세스에 자원을 분배하는 과정을 도시한 순서도이다.

[0034] 상기 단말은 S300단계에서 단말에서 구동되고 있는 통신 프로세스에 대한 목록을 업데이트할 수 있다. 보다 구체적으로, 새롭게 생성된 소켓에 대한 프로세스를 상기 통신 프로세스에 대한 목록에 추가하고, 제거된 소켓에 대한 프로세스를 상기 통신 프로세스에 대한 목록에서 제거하여, 현재 어떤 프로세스가 통신을 하고 있는지 모니터링할 수 있다. 즉, 상기 통신 프로세스에 대한 목록은 구동 프로세스에 대한 목록을 나타낼 수 있으며, 본 발명에서 상기 통신 프로세스에 대한 목록과 상기 구동 프로세스에 대한 목록은 혼용하여 사용이 가능하다. 상기 단말은 프로세스가 소켓을 생성하는 경우에, 본 발명에서 사용되는 정보들을 소켓의 구조체 안에 저장할 수 있다. 상기 단말은 상기 소켓의 구조체 안에 저장된 정보를 이용하여 통신 프로세스에 대한 목록을 업데이트할 수 있다. 상기 기존의 소켓 구조체에 추가되는 각 프로세스에 대한 정보를 포함하는 새로운 구조체의 구성은 아래의 도 4에서 예시하고 있다.

[0035] 또한, 상기 단말은 상기 통신 프로세스에 대한 목록이 생성되어 있지 않은 경우, 상기 단말은 S310단계에서 각 프로세스에 대한 '데이터 통신 특징 정보'를 획득할 수 있다. 즉, 상기 단말은 시스템에서 구동되는 각 프로세스 별 '데이트 통신 특징 정보'를 기록할 수 있는데, 상기 데이터 통신 특징 정보는 각 프로세스의 데이터 통신 양과 패턴을 포함할 수 있다. 이때 기록된 데이터 통신 양은, 예를 들어, 평균 값을 계산하여 각 프로세스 별 자원 분배를 결정하는데 이용될 수 있다. 상기 평균 값을 계산하는 것은 일 실시 예에 해당하며, 그 밖에 본 발명의 목적을 달성하기 위해 사용되는 방법도 본 발명에 포함될 수 있다.

[0036] 상기 단말은 S320단계에서 각 프로세스에 대한 자원 분배를 결정할 수 있다. 제한된 자원의 양과 현재 사용자가 사용하는 프로세스, 그 외의 프로세스들의 특성을 고려하여 자원을 적절하게 분배하는 것에서, 본 발명에서는 우선순위가 높은 프로세스의 품질이 가장 우선되어야 한다. 또한, 우선순위가 낮은 프로세스들에게 분배된 리소스가 부족하여 프로세싱이 종료되어서는 안되며, 프로세싱이 가능할 수 있는 자원이 분배되어야 한다. 이 때, 우선순위가 높은 프로세스가 필요로 하는 통신자원보다 사용할 수 있는 통신 자원이 더 많을 경우와 우선순위가 높은 프로세스가 필요로 하는 통신자원보다 사용할 수 있는 통신 자원이 더 적을 경우를 구별하여 자원 분배를 결정할 수 있다. 상기와 같은 자원 분배를 위하여 상기한 통신 프로세스에 대한 목록, 각 프로세스의 우선 순위 정보 및 각 프로세스의 데이터 통신 특징 정보에 기반하여 자원 분배를 결정할 수 있다. 보다 구체적인 자원 분배를 결정하는 방법은 도 6에서 설명한다.

[0037] 또한, 상기 단말은 S330단계에서 각 프로세스에 대한 자원 분배를 수행할 수 있다. 보다 구체적으로 상기 단말은 버퍼 크기와 알림(advertised) 윈도우 크기를 조정함으로써, 각 프로세스 별 단위 시간당 데이터 처리량을 조정하여 자원 분배를 수행할 수 있다. 또한 상기 단말은 상기 지연된 ACK 최대 개수를 조정함으로써, 송신자의 송신 윈도우 크기의 증감 속도를 조정할 수 있다. 즉 상기 S320단계에서 우선 순위 등을 고려하여 결정된 프로세스 별 자원 분배에 따라서, 우선순위가 높거나 낮은 프로세스의 전송 속도 상승과 저하를 동시에 컨트롤 할 수 있다. 보다 구체적으로 각 프로세스에 대한 자원 분배를 수행하는 방법은 도 9에서 설명한다.

[0038] 도 4는 소켓 구조체 안에 저장할 수 있는 프로세스 관련 정보에 대한 의사코드를 예시하는 도면이다.

[0039] 단말 안에서 프로세스가 소켓을 생성하는 경우, 상기 프로세스 관련 정보를 소켓 구조체 안에 저장하게 된다. 보다 구체적으로 상기 소켓 구조체 안에 저장할 수 있는 정보는 프로세스 식별자, 부모 프로세스 식별자, 우선

순위, 프로토콜 정보 및 지연된 ACK 최대 개수 등의 정보를 포함할 수 있다. 본 발명에서는 가장 핵심적인 정보만을 언급하고 있지만, 분배 목적이나 방식이 달라짐에 따라 추가적으로 필요한 정보가 있을 경우 새로운 정보를 간단히 추가하는 것 만으로도 새로이 적용 가능하므로 본 발명에서 언급한 정보 외에 추가적인 다양한 정보가 사용될 수 있으며, 본 발명은 이를 포함한다. 기존 소켓 구조체 안에 추가되는 상기와 같은 정보를 포함하는 새로운 구조체를 'socket_information'이라고 정의하며, 아래의 도 4 에서 예시하고 있다.

[0040] 프로세스 식별자 (Process Identifier, PID)는 상기 프로세스를 구분하는 특정한 숫자이다. 운영 체제에서 신규한 프로세스가 생성되게 되면 상기 프로세스에 PID가 할당될 수 있다. 예를 들어, 유닉스와 같은 운영체제에서 fork 시스템 콜에 의해 새로운 프로세스가 생성되게 되며, 이때 PID가 할당되게 된다.

[0041] 부모 프로세스 식별자(Parent Process Identifier, PPID)는 상기 프로세스들을 계층으로 나눌 때 상위 계층에 부여되는 식별자를 의미한다. 즉, 프로세스에는 부모프로세스라는 상위 계층과 자식프로세스라는 하위 계층으로 나뉘며, 부모프로세스는 새로운 프로세스를 생성하고 이 프로세스를 자식프로세스로 포함할 수 있다. 단말이 동작하는 경우 단순히 사용자가 사용하는 프로세스를 위한 프로세스뿐 아니라 시스템을 구성하고 동작시키는 수많은 프로세스들이 함께 동작하게 된다. 이러한 많은 프로세스 중, 본 발명에서는 사용자가 사용하는 프로세스들에게 초점을 맞추고 있으며, 다른 시스템 프로세스들과 구분을 하고 프로세스의 특징을 좀더 정확히 알기 위해 PPID 정보가 사용된다.

[0042] 지연된 ACK 최대 개수(Delayed ACK Max Count)는 상기 단말이 데이터 송신 장치에게 ACK를 보내는 주기를 의미한다. 즉 보내는 ACK의 개수를 일부러 줄여서 보내는 지연된 ACK(delyed ACK) 방법을 이용하기 위하여, ACK가 상기 지연된 ACK 최대 개수(Delayed ACK Max Count)만큼 모였을 때 보낸다. 만약 Delayed ACK Max Count를 2로 설정했다면 ACK가 2개 모일 때까지 기다리고, ACK가 모이는 즉시 전송을 한다. 만일 기다리던 중에 설정해둔 timeout을 넘기게 되면 대기 중이던 ACK을 즉시 전송한다. 송신자는 ACK을 받은 개수가 아닌 마지막으로 받은 ACK의 번호를 보고 얼마만큼 패킷이 성공적으로 전달되었는지를 판단하기 때문에 수신자가 보내는 ACK의 개수가 줄어들어도 문제가 없다. 하지만 ACK의 개수가 줄어들기 때문에 송신자의 송신 윈도우 크기가 빠르게 증가하지 못하게 된다. 본 발명에서는 이러한 특징을 활용하여, 각 프로세스의 우선 순위에 따라 각 프로세스들에게 각각 다른 지연된 ACK 최대 개수를 설정해 줌으로써 우선 순위가 높은 프로세스의 QoS를 향상시킨다. 따라서 각 소켓에 설정된 delayed ACK max count값을 쉽게 확인할 수 있도록, 통신 프로세스 관리부(230)에 의해 결정된 소켓 별 delayed ACK max count 값이 소켓 구조체 안의 새롭게 추가된 구조체 'socket_information' 안에 기록된다.

[0043] 우선 순위 (priority)는 프로세스의 중요도를 나타낸다. 운영체제는 프로세스를 관리하면서 상기 우선 순위 정보를 사용하여 현재 사용자에 의해 사용중인 프로세스의 경우 높은 중요도를 두고, 사용한지 오래된 프로세스의 경우 낮은 우선순위를 두어 메모리 관리가 필요할 경우 낮은 중요도의 프로세스부터 제거하게 된다. 이러한 프로세스 별 우선 순위 값을 이용하여 현재 사용중인 프로세스를 확인할 수 있다.

[0044] 프로토콜 정보(protocol)은 생성되는 소켓의 특성을 나타내는 정보이다. 예를 들어, 소켓을 생성할 때 SOCK_STREAM을 인풋으로 입력할 경우 TCP 소켓이 생성되게 되며, SOCK_DGRAM을 인풋으로 입력할 경우 UDP 소켓이 생성되게 되는데, 상기 소켓의 특성 정보를 포함할 수 있다.

[0045] 통신 프로세스 리스트(process_list)는 단말의 제어부(210)에 포함되는 통신 프로세스 관리부(230)가 관리하는 통신 프로세스에 대한 목록의 주소를 의미하며, 프로세스가 소켓을 통해 언제든 통신 프로세스 목록에 접근할 수 있도록 상기 새롭게 추가된 구조체에 통신 프로세스 목록의 주소를 포함할 수 있다.

[0046] 상기 단말은 생성된 소켓을 분석하여 소켓을 생성한 프로세스의 PID와 PPID를 찾아내며, 이렇게 찾아낸 PID를 활용하여 그 프로세스의 priority 값을 찾아내게 된다. 또한 이 소켓이 TCP 통신을 하는지 UDP 통신을 하는지를 분석하여 프로토콜 정보를 알아내게 된다. 이렇게 알아낸 프로세스의 PID, PPID, priority, 프로토콜 정보들을 소켓 구조체 안에 있는 socket_information 구조체에 추가하게 된다. 지연된 ACK 최대 개수(delayed ACK max count)는 추후에 통신 프로세스 관리부(230)에 의해 설정되는 값이므로, 초기에는 기본 값인 2로 설정을 한다. 마지막으로 소켓을 통해 언제든 통신 프로세스 목록에 접근할 수 있도록 통신 프로세스 관리부(230)가 관리하는 통신 프로세스 목록의 주소를 socket_information에 추가함으로써 소켓 정보 초기화 과정이 마무리 된다. 소켓 정보 초기화부(227)는 위에서 명시한대로 소켓을 초기화하며 초기화된 소켓을 통신 프로세스 관리부(230)로 전달하게 된다.

[0047] 도 5는 본 발명의 실시 예에 따른 도 3의 S300단계를 구체화한 도면이다.

[0048] 상기 단말은 S500단계에서 소켓이 새롭게 생성된 경우 신규 소켓 정보를 수신할 수 있다. 상기 단말은 상기 신규 소켓의 구조체에 포함된 PPID와 PID 정보를 활용하여 전달 받은 소켓이 사용자 프로세스인지 확인한다. 상기 신규 소켓이 사용자 프로세스라고 확인된 경우, 상기 단말은 S510단계에서 상기 소켓의 프로세스가 이미 통신 프로세스에 대한 목록에 등록된 프로세스인지 결정할 수 있다. 소켓의 프로토콜 정보를 확인하고 해당 프로토콜에 맞는 통신 프로세스 목록을 확인하여 이 프로세스가 이미 등록이 되었는지를 결정할 수 있다. 상기 단말은 만약, 상기 소켓의 프로세스가 이미 통신 프로세스에 대한 목록에 등록된 프로세스인 경우에, 상기 S300의 업데이트를 종료하고 S310단계로 복귀한다.

[0049] 만약, 상기 소켓의 프로세스가 이미 통신 프로세스에 대한 목록에 등록되지 않은 프로세스인 경우, S520단계에서 상기 신규 소켓의 프로세스를 통신 프로세스 목록에 등록하게 된다. 이 때 상기 단말은 상기 소켓의 구조체 내의 우선 순위 정보를 추출하여 저장할 수 있다.

[0050] 또한 상기 단말은 S530단계에서 제거된 소켓의 정보를 수신할 수 있다. 이 때 상기 단말은 제거된 소켓이 사용자 프로세스에 대응하는 소켓인지 확인하고, 맞다면 S540단계에서 상기 제거된 소켓의 프로세스가 이미 통신 프로세스에 대한 목록에 등록된 프로세스인지 결정할 수 있다. 만약, 상기 제거된 소켓의 프로세스가 통신 프로세스에 대한 목록에 등록된 프로세스인 경우, 상기 단말은 S550단계에서 소켓의 정보를 통신 프로세스 목록에서 검색하여 해당되는 프로세스 정보를 삭제할 수 있다. 만약, 상기 제거된 소켓의 프로세스가 통신 프로세스에 대한 목록에 등록된 프로세스가 아닌 경우, 상기 단말은 상기 S300의 업데이트를 종료하고 S310단계로 복귀한다. 이처럼, 상기 단말은 통신 프로세스에 대한 목록을 지속적으로 업데이트하여, 현재 어떤 프로세스가 통신을 하고 있는지를 모니터링 할 수 있다. 상기 통신 프로세스에 대한 목록을 통하여 상기 단말은 포어 그라운드로 실행되는 프로세스가 무엇인지 확인할 수 있으며, 이로써 프로세스들의 우선 순위를 결정할 수 있다.

[0051] 아래에서는, 상기 단말의 각 프로세스 별 자원 분배를 결정하는 방법을 구체화하여 설명한다.

[0052] 상기 단말에서 프로세스가 동시에 실행 중일 때 사용자가 사용하는 프로세스에 가장 큰 우선순위를 두어 좀더 많은 통신 자원을 할당하게 되어, 사용자의 서비스 품질(quality of service, QoS)를 향상시키는 방향으로 자원 분배를 결정할 수 있다. 각 프로세스 별 우선 순위를 결정하는 것은 어떠한 성능을 우선시 하냐에 따라 다르게 결정될 수 있으며, 가장 기본적인 우선순위 결정 방식의 예로, 현재 디스플레이에 보여지는 포어그라운드 프로세스의 우선순위를 가장 높게 두고 백그라운드로 동작하는 프로세스들의 우선순위를 낮게 둘 수 있다.

[0053] 또한 상기 단말은 각 프로세스 별 우선 순위를 상기 각 프로세스에 대응하는 소켓의 구조체 내의 우선 순위 정보를 추출하여 저장할 수 있다. 또는, 상기 단말은 상기 각 프로세스의 데이터 통신 특징 정보를 이용하여 각 프로세서의 우선 순위를 결정할 수 있다. 이와 같은 우선순위를 결정하는 과정을 통하여 상기 단말은 통신 연결을 기반으로 포어 그라운드에서 구동하는 프로세스가 있더라도 통신이 없는 순간 등 세부적인 정보를 획득하여 우선 순위를 조정할 수 있는 효과가 있다.

[0054] 도 6은 본 발명의 실시 예에 따른 구동되는 프로세스에 대응하는 어플리케이션들에 대한 자원 분배를 예시한 도면이다. 즉 도 6은 현재 구동중인 포어 그라운드에 있는 프로세스를 기준으로 우선 순위를 결정한 예이며, 상기 예 외에도 우선순위를 결정하는 다양한 방식이 선택될 수 있으며, 본 발명은 이를 포함한다. 즉 현재 포어그라운드에서 작업중인 어플리케이션(600)에 최우선으로 자원이 분배되며, 나머지 백그라운드에서 작업중인 어플리케이션(610)들은 차선적으로 자원이 분배되게 된다.

[0055] 도 7은 본 발명의 실시 예에 따른 각 프로세스별로 단위 시간당 데이터 처리량을 조정한 것을 설명하는 도면이다.

[0056] 보다 구체적으로, 도 7은 각 프로세스의 우선 순위에 기반하여 각 프로세스 별단위 시간당 데이터 처리량을 조정하기 전(700)과 후(710)를 비교한 도면이다. 각 프로세스의 우선 순위에 기반하여 각 프로세스에 자원 분배를 하기 전(700)에는 각 프로세스의 데이터 처리량이 차별화되지 않는다. 그러나, 상기 프로세스 별 우선 순위를 고려하여 자원 분배를 한 경우(710)은 각 프로세스 별 단위 시간당 데이터 처리량이 차별화된 것을 확인할 수 있다. 즉, 현재 포어 그라운드에 있어서 우선 순위가 높게 결정된 어플리케이션(720)의 경우에는 단위 시간당 데이터 처리량이 가장 높게 조정되었고, 상기 어플리케이션(720)보다 낮은 우선 순위로 결정된 어플리케이션(730, 740, 750)의 경우 단위 시간당 데이터 처리량이 낮게 조정되었다. 백그라운드 프로세스(730, 740, 750)들에게 할당되는 통신 자원을 포어그라운드 프로세스에게 좀더 집중함으로써 현재 사용자가 사용중인 프로세스의 통신을 좀더 원활하게 하여 QoS를 향상시키는 적용 예를 볼 수 있다.

[0057] 이하에서는 상기 도 7과 같이 각 프로세스 별 우선 순위에 따라 단위 시간 당 데이터 처리량을 조정하기 위한

자원 분배 방법을 설명한다. 상기 단말이 각 프로세스 별 상기 우선 순위 및 데이터 통신 특성 정보에 따라 자원을 분배할 때 아래와 같은 두 가지 상황을 고려할 수 있다.

[0058] 상황 1 : 우선 순위가 가장 높은 프로세스가 필요로 하는 통신 자원보다 가용 통신 자원이 더 많을 경우.

[0059] 상황 2 : 우선 순위가 가장 높은 프로세스가 필요로 하는 통신 자원보다 가용 통신 자원이 더 적을 경우.

[0060] 도 8은 본 발명의 실시 예에 따른 단말이 각 프로세스별로 자원 분배를 결정하는 것을 설명하는 도면이다.

[0061] 보다 구체적으로, 도 8은 상기 상황 1의 경우에 대하여 설명한다. 상황 1의 경우, 우선 순위가 가장 높은 프로세스가 요구로 하는 양의 통신자원을 그 프로세스에게 분배 해주더라도 가용 통신 자원이 남게 된다. 따라서 다른 우선순위가 낮은 프로세스들간 적절한 자원 분배 방법이 필요하게 된다. 도 8에서와 같이, i+1개의 프로세스가 하나의 단말에서 구동이 되어 있다고 가정하면, 1개의 포어그라운드 프로세스와 함께 i개의 백그라운드 프로세스들이 존재할 것이다.

[0062] F는 포어그라운드 프로세스(800)를 의미하며, 포어그라운드 프로세스가 필요로 하는 자원의 양을 f라고 한다. Bi는 i번째 백그라운드 프로세스(810, 820, 830)를 의미하며 Bi가 필요로 하는 자원의 양을 bi라고 한다. f와 bi는 각 프로세서 별 데이터 특성 정보에 의하여 결정된다. 즉, 상기 데이터 통신 특징 정보는 각 프로세스의 데이터 통신 양과 패턴을 포함할 수 있다. 이때 기록된 데이터 통신 양은, 예를 들어, 평균 값을 계산하여 각 프로세스 별 자원 분배를 결정하는데 이용될 수 있다. 상기 평균 값을 계산하는 것은 일 실시예에 해당하며, 그 밖에 본 발명의 목적을 달성하기 위해 사용되는 방법도 본 발명에 포함될 수 있다. 상기 백그라운드 프로세스(810, 820, 830)의 bi가 클수록 Bi는 자원을 상대적으로 많이 요구하는 프로세스이므로 F와의 경쟁이 자주 발생할 가능성이 크다. 반대로 bi가 작다면 F와의 자원을 경쟁할 가능성이 작다고 할 수 있다.

[0063] 본 발명에서 각 백그라운드 프로세스(810, 820, 830)에 자원을 분배하는 방법은 각 프로세스의 가중치 정보를 이용할 수 있다. 상기 각 프로세스 가중치를 결정하는 방법으로, 어떠한 요소를 중시하냐에 따라 다양한 방법이 적용될 수 있다. 가장 기본적으로 프로세스가 필요로 하는 자원의 양에 따라 결정될 수 있다. 위에서 언급하였듯 아래의 예 이외의 가중치를 결정하는 다양한 방법이 적용될 수 있으며 본 발명은 이를 포함한다. 각 백그라운드 프로세스(810, 820, 830)프로세스에 분배될 자원의 가중치 Φi는 아래의 수학식 1에 의하여 결정된다.

[0064] [수학식 1]

[0065] $\Phi i = 1 - (bi / (f + bi)) = f / (f + bi)$

[0066] 즉 bi가 f에 비해 많은 자원을 요구한다면 더 작은 가중치 Φi를 가지고, 요구하는 자원보다 작은 비율의 자원을 분배 받을 것이며 반대로 bi가 작다면 더 큰 가중치 Φi를 가지게 되며 Bi가 요구하는 양에 가까운 자원을 분배 받을 것이다.

[0067] 추가적으로, 포어그라운드 프로세스에 대한 더욱 큰 우선순위를 두기 위해서 위의 식에 scaling parameter α를 도입할 수 있으며, α가 커질수록 포어그라운드 프로세스에 대한 비중이 더 커진다고 할 수 있다. 가중치를 결정하는데, 위에서 설명한 예에서 제시된 방법 외에 다양한 방법이 적용될 수 있듯, 본 명세서에 제시된 scaling parameter α 외에도 다양한 추가적인 옵션이나 파라미터들이 고려되고 추가될 수 있으며, 본 발명은 이를 포함한다. 상기 α를 도입한 각 백그라운드 프로세스(810, 820, 830)프로세스에 분배될 자원의 가중치 Φi는 아래의 수학식 2에 의하여 결정된다.

[0068] [수학식 2]

[0069] $\Phi i = 1 - (\alpha bi /(f + \alpha bi)) = f / (f + \alpha bi)$

[0070] 이러한 Φi 는 일정 시간 t마다 업데이트되는 통신 프로세스에 대한 목록, 각 프로세스 별 우선 순위 정보, 또는 각 프로세스의 데이터 통신 특징 정보에 기반하여 다시 정해질 수 있다. 또한, 통신을 수행하는 프로세스들이 변경될 때 역시 새로 계산되어 갱신된다. 이렇게 정해진 각 프로세스의 가중치 Φi 와 필요로 하는 자원의 양 bi가 곱해져서 각 프로세스에 할당될 통신 자원의 양이 결정되게 된다. 본 명세서에서는 위와 같은 방법만을 언급하고 있지만, 시스템 개발자의 의도에 따라 가중치를 주는 다양한 방식이 적용될 수 있으며 본 발명은 이를 포함한다.

[0071] 이하에서는, 상황 2의 경우, 각 프로세스 별 자원 분배를 결정하는 방법에 대하여 설명한다. 이 경우에는, 우선순위가 높은 프로세스만 통신 자원을 사용하더라도 사용자의 QoS가 낮은 상황이다. 이러한 상황에서 우선순위가 낮은 프로세스들에게 많은 통신 자원을 할당할 경우 사용자의 QoS는 더욱 저하되며, 사용자가 우선순위가 높은

프로세스를 이용하는데 많은 지장이 생기게 된다. 따라서 이 경우에는 우선순위가 높은 프로세스에게 최대한의 통신 자원을 할당하여 높은 우선순위의 프로세스 성능을 최대한 보장하며, 다른 프로세스들에게는 통신이 유지되는 최소한의 자원만을 할당한다. 상기 통신이 유지되는 최소한의 자원만을 할당하는 것은, 예를 들어, 각 프로세스가 단위 패킷을 수신할 수 있는 정도의 자원을 할당하는 것을 포함할 수 있다. 이러한 동작방식은 우선순위가 낮은 프로세스들의 통신을 끊는 것이 아니라 유지시키며, 후에 우선순위가 높은 프로세스의 통신 자원 요구량이 작아졌을 때 우선순위가 낮은 프로세스들의 통신을 원활하게 하여 우선순위가 낮은 프로세스들의 데이터 통신작업을 뒤로 미루는 효과가 있다.

[0072] 본 발명은 종래의 TCP의 버퍼분배에는 포함되어 있지 않은 프로세스의 우선순위에 따른 자원의 분배 방식을 채택하였다. 이로 인하여 특히 전송자원을 우선순위가 높은 프로세스에 집중해서 사용자의 체감 서비스 품질을 높일 수 있다. 우선순위가 높은 프로세스에 집중하는 것은 사용자가 현재 사용중인 서비스에 우선권을 주게 되고, 이는 사용자의 입장에서는 최상의 만족도를 가질 수 있게 한다. 또한 이 기법에서의 자원 분배방식은 프로세스의 우선순위가 낮아도 최소한의 통신을 보장해 주며 시스템 전체의 통신 속도 측면에서도 큰 저하가 일어나지 않는다는 장점이 있다.

[0073] 이하에서는, 상기 단말이 상기 결정된 프로세스별 자원 분배 량에 기반하여, 각 프로세스에 대한 자원 분배를 수행하는 동작을 설명한다. 즉, 상기와 같이 프로세스 별 우선 순위, 필요 자원 량에 기반하여 계산한 가중치를 이용하여 도출한 프로세스 별 자원 량에 따라 각 프로세스 별 자원 분배를 수행한다. 이 경우, 우선 순위가 높은 프로세스에는 더 많은 자원을 분배하게 되고, 우선 순위가 낮은 프로세스에는 더 적은 자원을 분배하게 된다.

[0074] 각 프로세스에 대한 자원 분배를 수행하는 것은 프로세스에 대응하는 소켓의 성능을 조정하는 것을 의미한다. 보다 구체적으로 상기 단말은 버퍼 크기와 알림(advertised) 윈도우 크기를 조정함으로써, 각 프로세스 별 단위 시간당 데이터 처리량을 조정하여 자원 분배를 수행할 수 있다. 도 9는 상기 버퍼 크기를 조정함으로써, 각 프로세스 별 단위 시간당 데이터 처리량을 조정하는 것을 예시한 도면이다. 보다 구체적으로, 프로세스의 버퍼 크기가 증가할수록 프로세스의 단위 시간당 데이터 처리량이 증가하는 것을 알 수 있다.

[0075] 각 프로세스의 수신 윈도우 크기는 TCP에서 수신자가 한번에 받을 수 있는 최대 크기를 의미하며, 수신 윈도우 크기를 송신자에게 알림(advertisement)으로써 수신자가 받을 수 있는 데이터 양을 송신자에게 알리게 되는데 이 때의 데이터 크기를 알림(advertised) 윈도우라고 한다. 버퍼는 소켓에 할당된 버퍼의 양을 의미하며, 상기 수신 윈도우의 상한선을 의미하게 된다. 또한 상기 단말은 상기 지연된 ACK 최대 개수를 조정함으로써, 송신자의 송신 윈도우 크기의 증감 속도를 조정할 수 있다.

[0076] 상기 버퍼 크기의 조정과 알림 윈도우 크기의 조정은 모두 각 프로세스 별 통신 성능을 조정할 수 있으나 그 조정 방법에 있어서, 아래와 같은 차이점이 있다.

[0077] 먼저 알림 윈도우 크기의 조정은 알림(advertised) 윈도우가 실질적으로 데이터를 받는 양을 의미하기 때문에, 윈도우의 증가속도를 향상시켜 순간적으로 받을 수 있는 데이터 속도를 향상시키는 등 데이터 수신 속도를 조정할 수 있다. 반면에, 버퍼 크기의 조정은 UDP에서도 사용이 가능하며, 실질적으로 소켓에 할당되는 자원이기 때문에 메모리 효율성에 영향을 준다. 특히, 기존 시스템의 UDP의 속도 조절은 수신자가 일체 관여할 수도 없고, 송신자가 통신상황을 알 수도 없다. 그러나 본 발명을 통해서 어플리케이션의 종류나 개발상태와 무관하게 UDP의 컨트롤이 가능해진다는 장점이 있다. 이러한 UDP 버퍼 컨트롤 기술은 수신부에서 자의적으로 조절할 수 있는 바, 기기 내 임의의 UDP 연결로 인해 다른 연결에 영향을 미치는 현상을 완화할 수 있으므로, 그 활용의 폭이 높은 효과가 있다.

[0078] 따라서, 상기 두 방법은 상호보완적 관계가 될 수 있으며, 본 발명에서는 두 방법을 상황에 따라 적절히 사용함으로써 더 효과적으로 자원 분배를 수행할 수 있다.

[0079] 본 발명에서 각 프로세스 별로 자원 분배를 수행하는 경우는 아래와 같이 두가지 경우가 있다.

[0080] 자원 분배를 수행하는 방법 1 : 각 프로세스에 할당된 자원을 감소시키는 경우.

[0081] 자원 분배를 수행하는 방법 2 : 각 프로세스에 할당된 자원을 증가시키는 경우.

[0082] 먼저 자원 분배를 수행하는 방법 1인 각 프로세스에 할당된 자원을 감소시키는 경우, 알림(advertised) 윈도우 크기와 버퍼 크기모두 감소하게 된다. 따라서 이 경우에는 알림(advertised) 윈도우 크기를 먼저 감소시킨 뒤에 버퍼 크기를 감소시키게 된다. 만약 버퍼 크기를 먼저 감소시킨다면 송신자는 버퍼 크기가 감소되기 전에 계산

된 알림(advertised) 윈도우를 기준으로 데이터를 전송하기 때문에 수신자는 송신자가 보낸 데이터의 일부를 받지 못하게 된다. 이러한 문제를 방지하기 위해 위와 같은 순서로 알림(advertised) 윈도우 크기와 버퍼 크기를 감소시키게 된다.

[0083] 다음으로 자원 분배를 수행하는 방법 2인, 각 프로세스에 할당된 자원을 증가시키는 경우, 방법 1의 경우와 반대로 할당된 버퍼의 크기를 먼저 증가시킨 후에 알림(advertised) 윈도우 크기를 증가하게 된다. 그렇게 하지 않을 경우 송신자는 수신자가 받을 수 있는 메모리 크기 이상의 데이터를 전송하게 되며 일부 데이터가 누락되는 상황이 발생하게 된다. 따라서 할당 버퍼 크기 증가 후 알림(advertised) 윈도우 크기를 증가시킴으로써 이러한 문제를 방지하게 된다.

[0084] 상기 언급한 대로 버퍼는 알림(advertised) 윈도우의 상한선을 제한하며, 알림(advertised) 윈도우는 버퍼로 제한된 범위 안에서 프로토콜에 의해 크기가 변동된다. 이 기법은 송신자 측의 수정이 불필요 하며 수신자 측의 버퍼 크기와 알림(advertised) 윈도우 크기를 함께 조정함으로써 새로운 하드웨어나 소프트웨어의 추가 없이 커널 수정만으로 구현할 수 있다. 이렇듯 알고리즘 적용에 제약이 거의 없으므로, 범용성에서 상당한 강점을 가지고 있다.

[0085] 또한, 버퍼는 절대적인 값을 할당하는 반면에 알림(advertised) 윈도우 값은 절대값으로 설정하는 방법 외에도, 증가속도나 감소속도를 조정하는 방법이 적용될 수 있다. 예를 들면, 우선순위가 높은 프로세스의 알림(advertised) 윈도우 증가속도를 더 빠르게 하거나 우선순위가 낮은 프로세스의 알림(advertised) 윈도우 감소속도를 더 빠르게 하는 방식으로 적용이 가능하다.

[0086] 아래에서는 지연된 ACK 최대 개수를 조정함으로써, 송신자의 송신 윈도우의 증가 속도나 감소 속도를 조정하는 방법에 대하여 상세히 설명한다.

[0087] 지연된 ACK(Delayed ACK)는 ACK의 전송 횟수를 줄임으로써 실질적인 데이터 전송의 비율을 향상시켜 전송 효율을 높이는 목적으로 사용된다. 하지만 ACK의 개수가 줄어들기 때문에 송신자의 윈도우가 빠르게 증가하지 못하게 된다. 본 발명에서는 이러한 특징을 활용하여, 각 프로세스의 우선 순위에 따라 각 프로세스들에게 각각 다른 지연된 ACK 최대 개수를 설정해 줌으로써 우선 순위가 높은 프로세스의 QoS를 향상시킨다. 즉 우선 순위가 높은 프로세스의 경우, 지연된 ACK 최대 개수를 낮게 설정해주어 ACK를 보내는 주기를 짧게 설정하여, 상기 송신자의 송신 윈도우의 변화 속도를 향상시킬 수 있다. 만약 우선 순위가 높은 프로세스에 대하여 할당된 자원량을 증가시키고자 할 때, 상기 지연된 ACK 최대 개수를 낮춰 주어, 상기 송신자의 송신 윈도우 크기의 증가 속도를 향상시킬 수 있다. 즉, 실질적으로 보내는 ACK 개수를 조절함으로써 상기 송신자의 송신 윈도우 크기의 증가 속도를 조절할 수 있다. 송신자의 송신 윈도우 크기가 증가하는 경우, 결과적으로 프로세스의 소켓 성능이 향상되어, 프로세스의 단위 시간 당 데이터 처리량이 증가하게 된다. 아래의 도 10은 상기 설명한 결과를 설명하는 도면이다.

[0088] 도 10은 본 발명의 실시 예에 따른 지연된 ACK 최대 개수(delayed ACK max count)의 조정에 따른 단위 시간당 데이터 처리량의 변화를 예시하는 도면이다.

[0089] 보다 구체적으로, 도 10a는 본 발명에서 사용되는 지연된 ACK 최대 개수의 조정 효과를 확인하기 위해 수행한 실험 환경을 보여준다. 네트워크 시뮬레이터 3를 사용하여 두 개의 서버(1000, 1005)에서 AP를 거쳐 하나의 휴대 기기로 TCP 통신을 이용한 하향 링크를 전송하게 하였으며, 지연된 ACK 최대 개수(delayed ACK max count)를 다르게 주어, 그 효과를 확인하였다. 두 소켓 중 서버 1(1000)에 대응하는 소켓 1의 경우 지연된 ACK 최대 개수(delayed ACK max count)를 기본 값인 2로 고정하였으며, 소켓 2의 지연된 ACK 최대 개수(delayed ACK max count)만을 변경하였다. 초반 윈도우 증가에 대한 영향을 확인할 수 있도록 총 실험시간은 10초로 하였으며, 도 10b는 그 결과를 보여주는 그래프이다.

[0090] 도 10b에 따르면 지연된 ACK 최대 개수(delayed ACK max count) 값이 커질수록 소켓 2의 단위 시간당 데이터 처리량은 낮아짐과 함께 소켓 1의 단위 시간당 데이터 처리량은 높아짐을 볼 수 있다. 이는 지연된 ACK 최대 개수(delayed ACK max count) 값을 조정함으로써 송신자의 송신 윈도우 크기의 증가 속도를 조정할 수 있음을 보여주며, 본 발명에 사용된 기술의 타당성을 입증하는 결과이다.

[0091] 이하에서는 본 발명에 따른 효과 및 적용될 수 있는 영역에 대하여 설명한다.

[0092] 본 발명을 활용하면 프로세스에 따라 적절한 버퍼 크기를 할당함으로써 버퍼 활용 효율을 향상시킬 수 있다. 일반적인 홈페이지의 연결에는 20개에 달하는 소켓 생성이 이루어지고, 이는 메모리를 상당히 소비하게 된다. 본 기술을 통해 프로세스별 메모리 사용 효율을 극대화 할 수 있고, 특히 웨어러블 디바이스에서 메모리 확보가 어

렵다는 점에서, 중요하게 활용될 수 있다. 실제로 웨어러블 디바이스에 대체로 쓰이는 소형화된 메모리는 256MB(Mega Byte) 수준이거나 현 메인 스트림 기기의 1/8 수준에 불과하다. 비록 간략화 된 모바일 페이지에서도 PC버전 홈페이지와 크게 다르지 않은 수준인 15개 내외의 TCP 또는 UDP연결이 생성된다는 점을 감안하면, 본 기술의 버퍼 컨트롤을 통한 효율적인 메모리 사용이 웨어러블 디바이스를 포함한 차세대 기기에서 상당한 효율을 발휘할 수 있다.

[0093] Multipath TCP(MPTCP)는 현 모바일 기기의 성능 향상과 함께, Wi-Fi와 무선 통신 셀망을 동시에 사용하는 식의 다양한 통신방법을 동시에 활용하는 기법이다. 차세대 모바일 통신에서는 주력으로 사용될 예정인 기술이다. 이 MPTCP는 모든 서브 플로우(sub flow)의 알림 윈도우와 버퍼량을 서로 공유한다는 특징이 있다. 이는 결국 MPTCP 연결을 설정하더라도 어플리케이션 단계에서는 소켓이 하나만 열린 것처럼 보이며, 다양한 서브 플로우 연결이 생성되어도 버퍼 컨트롤을 통해 모든 서브 플로우를 한번에 컨트롤이 가능하다는 것을 뜻한다. 따라서 본 기술은 MPTCP에서도 충분히 효과를 거둘 수 있으며, 별다른 프로토콜 변경을 요구하지 않으므로 이러한 후속 프로토콜 모델이 나와도 사용 가능하다.

[0094] 또한, 통신서버가 혼잡(congestion)이 충분히 일어날만한 상황인 경우, 혹은 단일 AP에 여러 기기가 연결되어 있는 경우에는 본 기술을 통해 빠른 자원요구와 ACK응답을 유발하면서 AP의 한정된 대역폭(bandwidth)를 점유하는 경쟁에서 우위를 점할 수 있게 해준다. 이는 해당 기술이 적용되지 않은 타 기기와의 경쟁에서 이길 가능성이 높아짐을 뜻하며 본 기술의 자원 점유경쟁 측면에서의 효과는 다음과 같이 볼 수 있다.

[0095] 또한, 한 개의 AP에 여러 기기가 연결되어 경쟁을 할 경우, 본 발명이 적용된 기기에서는 우선순위가 높은 프로세스의 할당 버퍼와 수신 윈도우를 빠르게 증가시키기 때문에 타 기기보다 빠른 자원점유를 통해 QoS의 유지가 가능하다.

[0096] 또한, 특정 자원을 놓고 서로 다른 기기나 프로세스 등이 경쟁을 할 경우, 어느 한쪽에서 지속적으로 자원을 가져가거나 점유하여서 다른 한쪽이 계속해서 자원을 얻지 못하는 현상이 나타날 수 있는데 이를 starvation 문제라 부르고 있다. 해당 상황이 발생할 경우에 이득을 보는 대상은 지속적인 이득을 가져갈 수 있으나, 한번 손해보기 시작한 대상은 지속적인 불이익을 당할 수 있고, 경쟁 자체를 시작하기 어렵게 만들게 된다. 만일 starvation문제가 발생하게 되면 동일한 기술이 적용된 기기들간의 경쟁에서도 후속으로 통신을 실행한 쪽이 지속적인 손해를 보는 결과를 가져올 수 있다. 본 기술방식은 일방적인 점유가 아닌, 상대적으로 통신자원을 가져갈 확률과 요구량을 높이는 방식이기 때문에, 다른 링크의 통신을 방해한다거나 끊어버리는 현상을 일으키지 않으므로, 이러한 starvation문제를 일으키지 않기 때문에 부작용 없이 타 기기와의 경쟁에서도 우위를 점하여 QoS를 높이는 것이 가능하다는 특징을 보여준다.

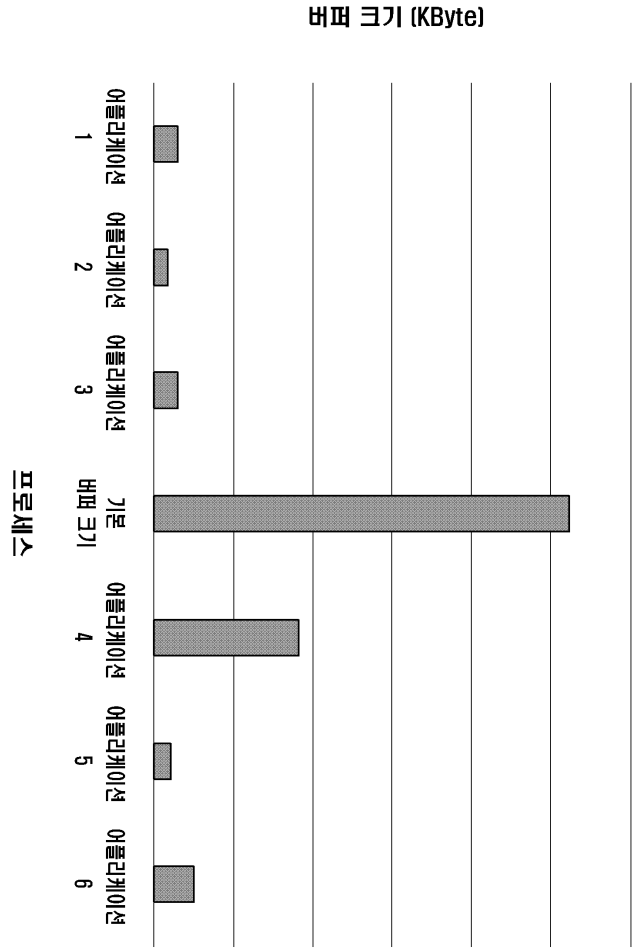
[0097] 본 발명은 가장 널리 사용되는 프로토콜인 TCP와 UDP에 집중되어 설명되었으나 메모리나 윈도우는 대부분의 통신 프로토콜에서 사용되는 개념이며 다른 많은 프로토콜들이 TCP와 UDP를 기반으로 만들어졌다. 따라서 본 발명은 본 명세서에서 언급한 TCP와 UDP에 국한되어 적용되는 것이 아닌, 간이 전자 우편 전송 프로토콜(Simple Mail Transfer Protocol, SMTP), 하이퍼 본문 전송 규약(HyperText Transfer Protocol, HTTP), 인터넷 제어 메시지 프로토콜(Internet Control Message Protocol, ICMP), 실시간 전송 프로토콜(Real-time Transport Protocol, RTP), 데이터그램 혼잡 제어 프로토콜(Datagram Congestion Control Protocol, DCCP), 주소 결정 프로토콜 (Address Resolution Protocol, ARP), 역순 주소 결정 프로토콜(Reverse Address Resolution Protocol, RARP), 라우팅 정보 규약(Routing Information Protocol; RIP), 내부 게이트웨이 라우팅 프로토콜(Interior Gateway Routing Protocol, IGRP), Post Office Protocol version 3(POP3) 등의 다양한 통신 프로토콜에 본 발명이 응용 적용될 수 있으며 본 발명은 이를 포함한다. 또한 본 발명은 버퍼를 활용하기 때문에 수송 계층을 거치지 않는 프로토콜(예: 인터넷 그룹 관리 프로토콜(Internet Group Management Protocol, IGMP), 최단 경로 우선 프로토콜(Open Shortest Path First, OSPF), 인터넷 제어 메시지 프로토콜(Internet Control Message Protocol, ICMP) 등)이나 어플리케이션을 사용하는 시스템에도 적용 가능하다. 따라서 본 발명의 적용 가능성은 매우 넓으며, 본 발명은 본 명세서에서 제시한 예 외의 다양한 적용 가능성을 포함한다.

[0098] 한편, 본 명세서와 도면에는 본 발명의 바람직한 실시 예에 대하여 개시하였으며, 비록 특정 용어들이 사용되었으나, 이는 단지 본 발명의 기술 내용을 쉽게 설명하고 발명의 이해를 돕기 위한 일반적인 의미에서 사용된 것이지, 본 발명의 범위를 한정하고자 하는 것은 아니다. 여기에 개시된 실시 예 외에도 본 발명의 기술적 사상에 바탕을 둔 다른 변형 예들이 실시 가능하다는 것은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 자명한 것이다.

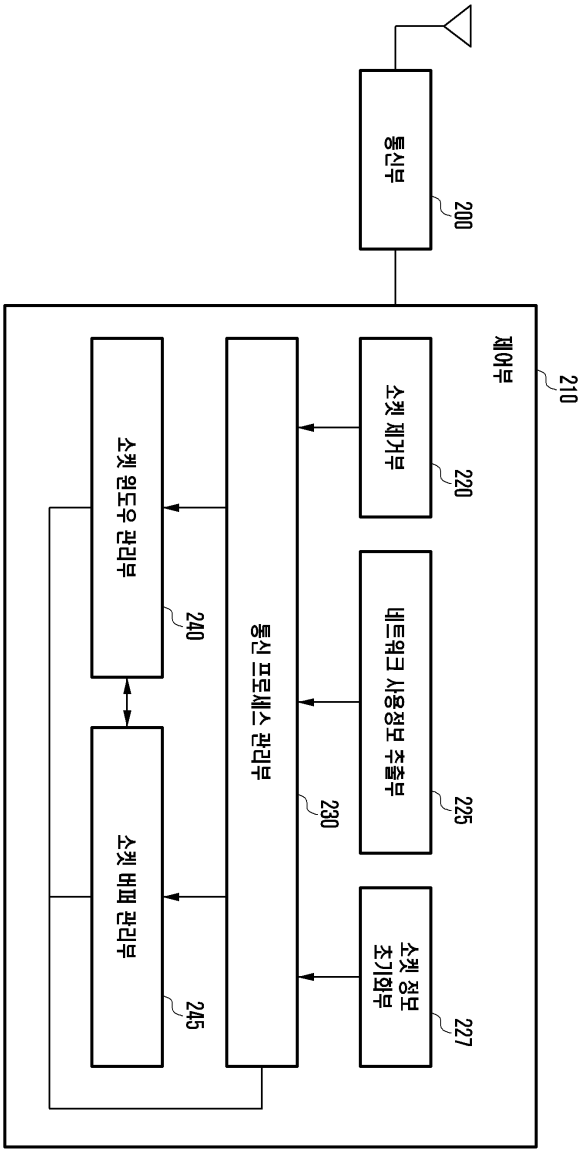
[0099]

도면

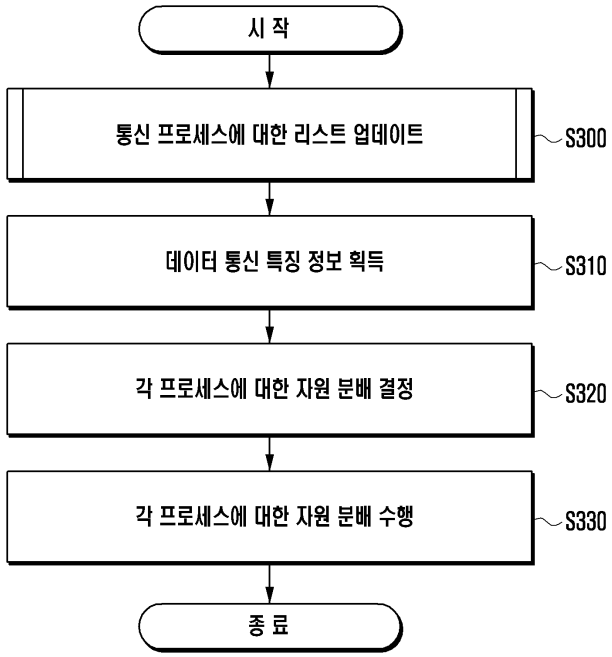
도면1


버퍼 크기 (KByte)
프로세스
어플리케이션 1
어플리케이션 2
어플리케이션 3
버퍼 크기
어플리케이션 4
어플리케이션 5
어플리케이션 6

도면2


통신부
200
제어부
210
소켓 제거부
220
네트워크 사용정보 추출부
225
소켓 정보 초기화부
227
통신 프로세스 관리부
230
소켓 윈도우 관리부
240
소켓 버퍼 관리부
245

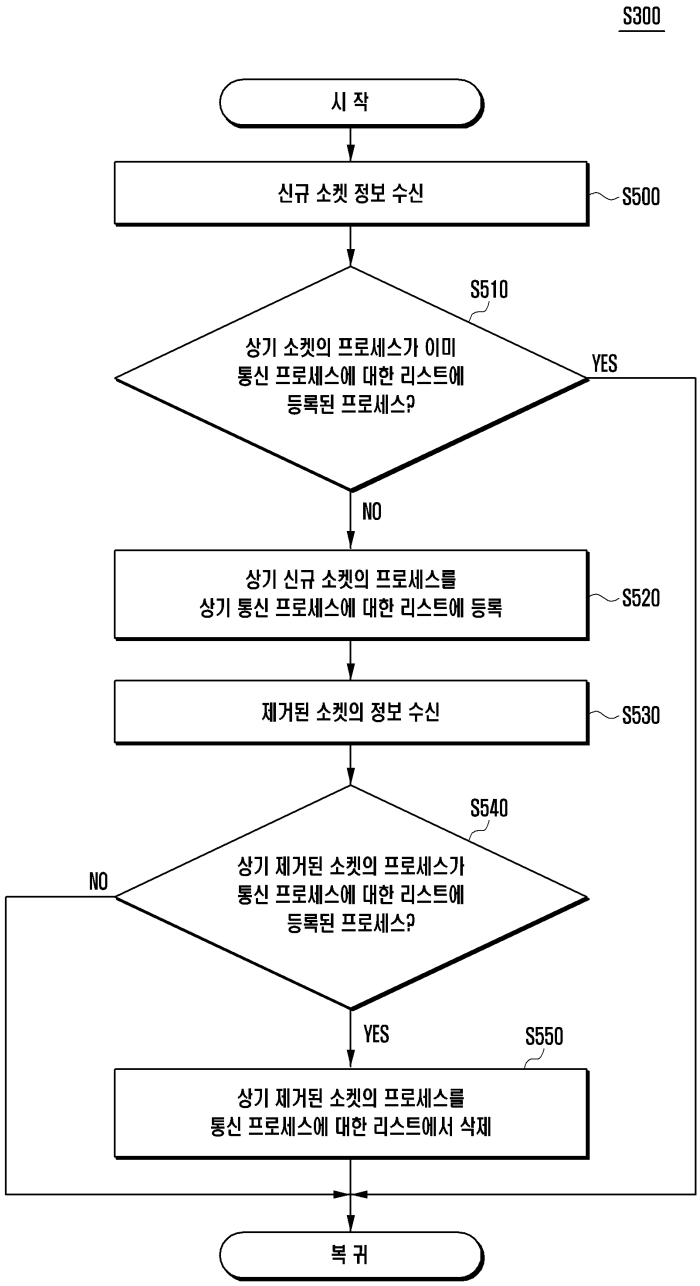
도면3



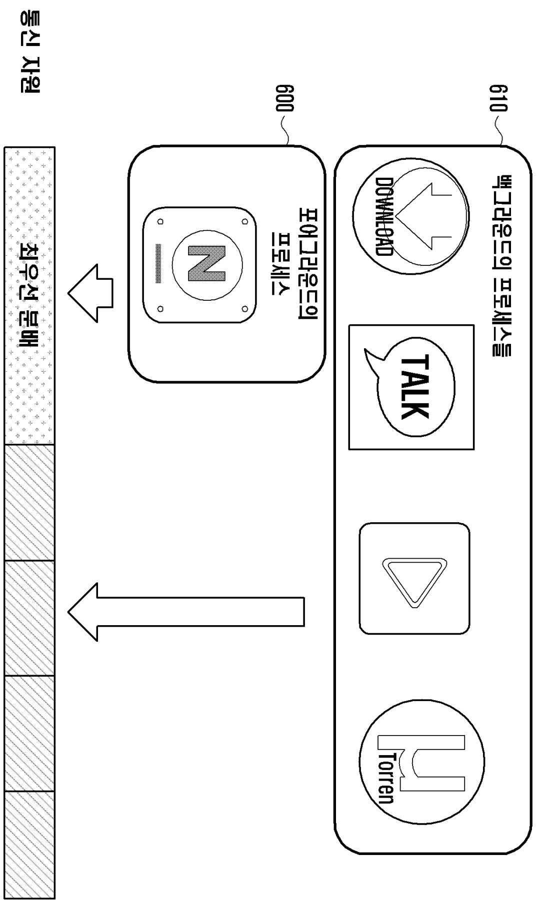
도면4

```
struct socket_information{
          pid;
          ppid;
          delayedAckMaxCount;
          priority;
          protocol;
          process_list;
}
```

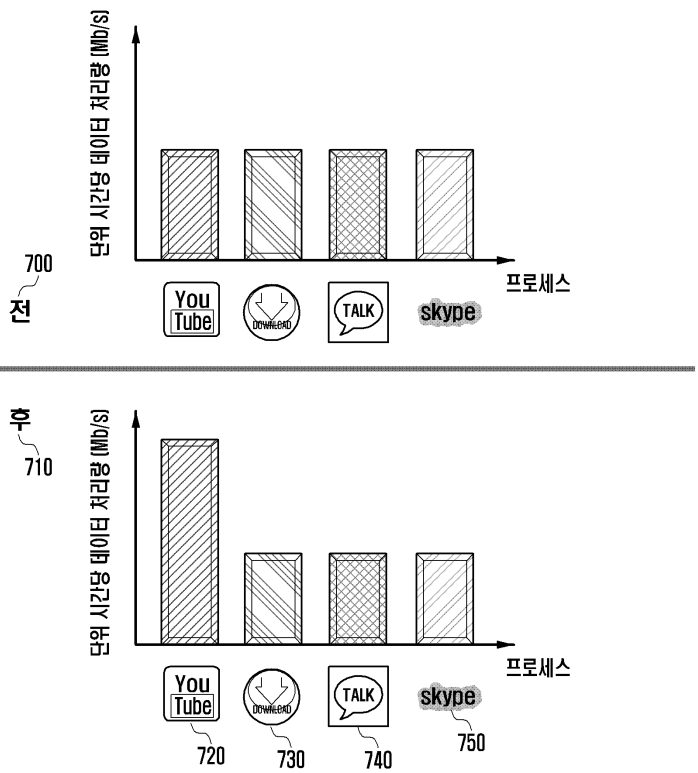
도면5


S300
시 작
신규 소켓 정보 수신
S500
S510
상기 소켓의 프로세스가 이미
통신 프로세스에 대한 리스트에
등록된 프로세스?
YES
NO
상기 신규 소켓의 프로세스를
상기 통신 프로세스에 대한 리스트에 등록
S520
제거된 소켓의 정보 수신
S530
S540
상기 제거된 소켓의 프로세스가
통신 프로세스에 대한 리스트에
등록된 프로세스?
NO
YES
S550
상기 제거된 소켓의 프로세스를
통신 프로세스에 대한 리스트에서 삭제
복 귀

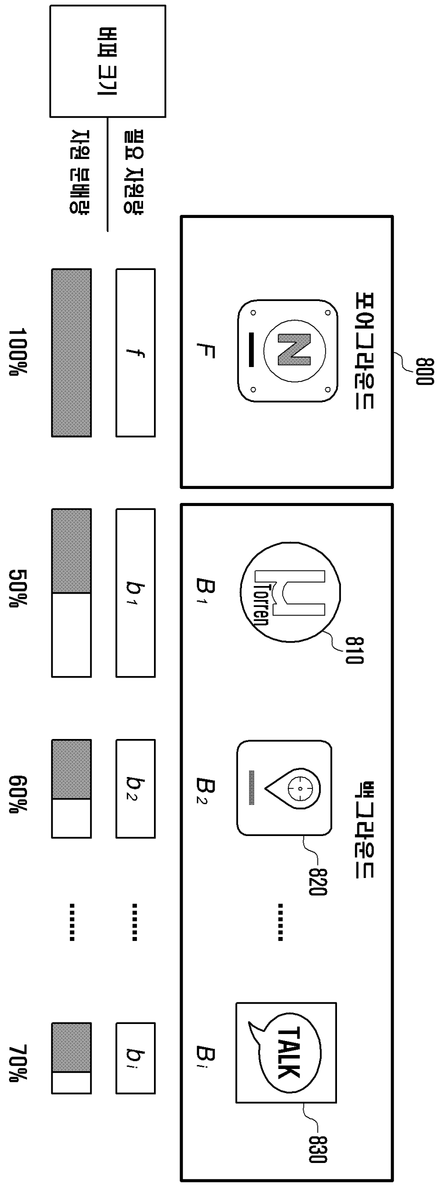
도면6


통신 자원
600
610
포어그라운드의 프로세스
백그라운드의 프로세스들
최우선 분배
DOWNLOAD
TALK
Torren

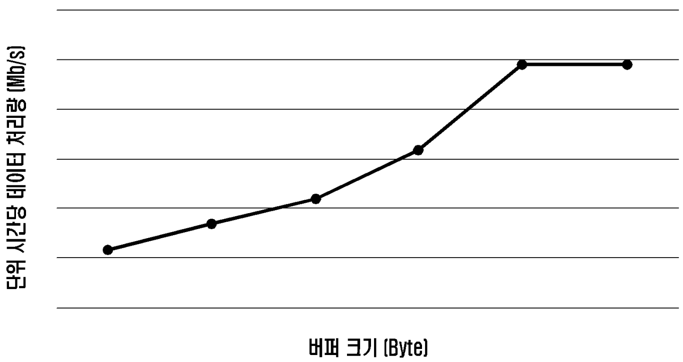
도면7


700
전
710
후
단위 시간당 데이터 처리량 (Mb/s)
프로세스
You Tube
DOWNLOAD
TALK
skype
720
730
740
750

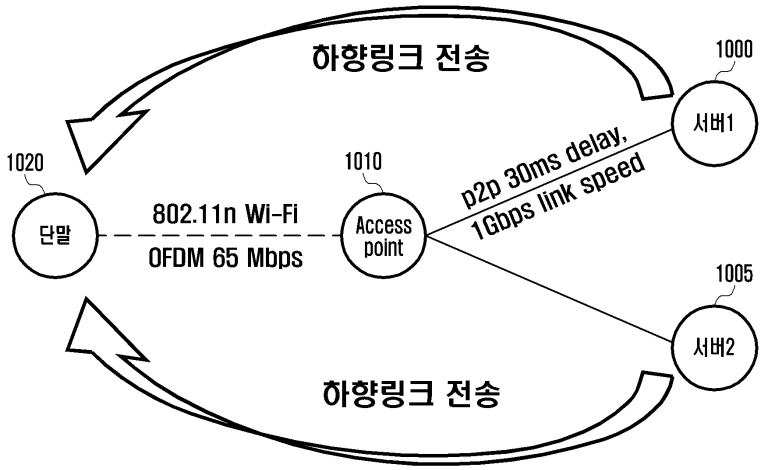
도면8


버퍼 크기
필요 자원량
자원 분배량
포어그라운드
800
F
f
100%
백그라운드
810
820
830
B_1
B_2
B_i
b_1
b_2
b_i
50%
60%
70%
Torrent
TALK

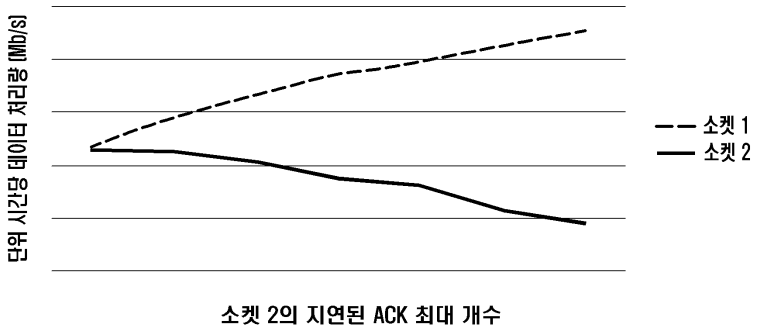
도면9


단위 시간당 데이터 처리량 (Mb/s)
버퍼 크기 (Byte)

도면10a


하향링크 전송
1000
서버1
1020
단말
802.11n Wi-Fi
OFDM 65 Mbps
1010
Access point
p2p 30ms delay,
1Gbps link speed
1005
서버2
하향링크 전송

도면10b


단위 시간당 데이터 처리량 (Mb/s)
소켓 1
소켓 2
소켓 2의 지연된 ACK 최대 개수