(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년12월23일
(11) 등록번호 10-1098744
(24) 등록일자 2011년12월19일

(51) Int. Cl.
H04L 12/24 (2006.01) *H04L 12/42* (2006.01)
(21) 출원번호 10-2005-0008699
(22) 출원일자 2005년01월31일
심사청구일자 2010년01월25일
(65) 공개번호 10-2006-0042903
(43) 공개일자 2006년05월15일
(30) 우선권주장
10/881,695 2004년06월30일 미국(US)
60/540,738 2004년01월30일 미국(US)
(56) 선행기술조사문헌
JP09205429 A*
KR1020020077446 A
KR1020030089604 A
KR1020010072079 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
조오우 리동
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내
키우 릴리
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내
(뒷면에 계속)
(74) 대리인
제일특허법인

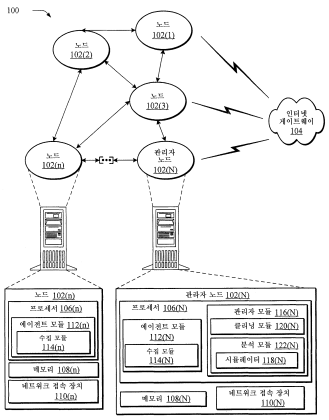
전체 청구항 수 : 총 24 항 심사관 : 유병철

(54) 장애의 탐지 및 진단

(57) 요 약

본 발명은 네트워크 중재 프레임워크를 제공한다. 일 구현예의 방법에서는, 네트워크를 기술하는 데이터를 네트워크 시뮬레이션에 공급하여 네트워크 시뮬레이션이 네트워크 성능에 대한 추정을 제공함으로써 네트워크의 동작에서의 불일치를 탐지하는 단계를 포함한다. 또한, 네트워크 성능의 추정이 네트워크의 관찰된 네트워크 성능과 다른지를 판단하는 단계를 포함한다. 불일치의 근본 원인은 네트워크 성능의 추정이 관찰된 네트워크 성능에 근사할 때까지 네트워크 시뮬레이션에 복수의 장애 중 하나 이상을 삽입시킴으로써 진단된다.

대 표 도 - 도1



(72) 발명자

배흘 파람비

미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내

라오 아난스 라자고팔라

미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 내

특허청구의 범위

청구항 1

네트워크에서의 장애(faults)의 탐지 및 진단을 위한 방법으로서,

네트워크 시뮬레이션(210)이 네트워크 성능의 추정을 제공하도록 상기 네트워크를 기술하는 데이터를 상기 네트워크 시뮬레이션(210)에 공급하는 단계(504)와, 상기 네트워크 성능의 추정이 상기 네트워크의 관찰된 네트워크 성능과 다른지를 판단하는 단계(508)에 의해, 네트워크의 동작에서의 불일치를 탐지하는 단계, 및

상기 네트워크 성능의 추정이 상기 관찰된 네트워크 성능에 근사할 때까지, 복수의 장애들 중 하나 이상을 상기 네트워크 시뮬레이션에 삽입시킴으로써 상기 불일치의 근본 원인(root cause)을 진단하는 단계(510, 512, 514, 516)

를 포함하는 방법.

청구항 2

제1항에 있어서,

상기 판단하는 단계는 상기 네트워크 성능의 추정이 상기 네트워크의 상기 관찰된 네트워크의 성능과 상응하는 임계치 초과만큼 다른지를 확인하는 단계를 포함하는 방법.

청구항 3

제1항에 있어서,

상기 진단하는 단계는,

추정된 상기 네트워크 성능과 상기 관찰된 네트워크 성능 간의 차를 장애 크기(fault magnitude)로 변환시키는 단계와,

상기 장애 크기가, 상기 네트워크 시뮬레이션을 상기 네트워크의 상기 관찰된 네트워크 성능에 근사하도록 움직이는지를 판단하는 단계

를 포함하는 방법.

청구항 4

제1항에 있어서,

상기 진단하는 단계는,

추정된 상기 네트워크 성능과 상기 관찰된 네트워크 성능 간의 차를 장애 크기로 변환시키는 단계와,

상기 장애 크기가 상응하는 임계치 미만인지를 판단하는 단계와,

상기 장애 크기가 상기 상응하는 임계치 미만인 경우, 상기 네트워크 시뮬레이션에 대해 입력으로서 사용되는 장애 세트로부터 대응하는 상기 장애를 제거시키는 단계

를 포함하는 방법.

청구항 5

제1항에 있어서,

상기 진단하는 단계는,

초기 장애 세트를 생성하기 위한 초기 진단을 행하는 단계와,

상기 네트워크 시뮬레이션에 의해 입력으로서 사용될 때, 상기 네트워크 시뮬레이션에 의해 제공된 상기 네트워크 성능의 추정이 상기 관찰된 네트워크 성능에 근사하게 하는 현재 장애 세트에 도달하도록 상기 초기 장애 세트를 반복적으로 정련하는(iteratively refining) 단계

를 포함하는 방법.

청구항 6

제1항에 있어서,

상기 네트워크는 다중-홉(multi-hop) 무선 네트워크인 방법.

청구항 7

제1항에 있어서,

상기 네트워크는 유선 네트워크인 방법.

청구항 8

제1항에 있어서,

상기 데이터는 상기 네트워크의 시작(originating) 노드와 대상(target) 노드 사이에서 상기 네트워크를 통해 데이터 패킷에 의해 취해진 일련의 홉(hop)으로 형성된 라우트(route)를 기술하는 방법.

청구항 9

제1항에 있어서,

상기 데이터는,

라우팅을 포함하는 네트워크 형상(topology),

트래픽 통계,

물리적 매체, 및

네트워크 성능으로 이루어진 그룹에서 선택된 트레이스(trace) 데이터를 포함하는 방법.

청구항 10

컴퓨터 실행가능 명령어를 포함하는 컴퓨터 판독가능 기록 매체로서, 상기 컴퓨터 실행가능 명령어는 컴퓨터에 의해 실행될 때 상기 컴퓨터로 하여금 제1항의 방법을 수행하게 하는 컴퓨터 판독가능 기록 매체.

청구항 11

서로 통신가능하게 결합되어 네트워크를 형성하는 복수의 노드(102)를 포함하는 상기 네트워크에서의 장애의 탐지 및 진단을 위한 시스템으로서,

하나 이상의 상기 노드는, 네트워크 세팅을 수집하고, 상기 네트워크를 통한 통신을 위해 상기 네트워크 세팅을 포함하는 통신을 형성하도록 실행가능한 에이전트 모듈(112(N))을 포함하고,

적어도 하나의 상기 노드는, 상기 통신을 수신하고, 네트워크 시뮬레이션이 네트워크 성능의 추정을 제공하도록 상기 통신으로부터 얻어진 상기 네트워크 세팅에 기반하여 상기 네트워크 시뮬레이션을 생성하고, 상기 네트워크의 네트워크 성능의 관찰과 상기 네트워크 성능의 추정을 비교함으로써(508) 네트워크 동작에서의 장애를 탐지(510, 512, 514, 516)하도록 실행가능한 관리자 모듈(116(N))을 포함하며,

상기 관리자 모듈은 또한, 상기 추정이 상기 관찰에 근사할 때까지, 복수의 상기 장애 중 하나 이상을 상기 네트워크 시뮬레이션에 삽입시킴으로써 장애를 진단하도록 실행가능한 시스템.

청구항 12

제11항에 있어서,

적어도 두 개의 상기 노드는 각자의 상기 관리자 모듈을 포함하는 시스템.

청구항 13

제11항에 있어서,

각각의 상기 노드는 각자의 상기 에이전트 모듈을 포함하는 시스템.

청구항 14

제11항에 있어서,

상기 관리자 모듈은 또한,

상기 탐지된 장애에 기반하여 시정 액션(corrective action)을 도출해 내고,

상기 에이전트 모듈이 상기 시정 액션을 수행하게 하도록 실행가능한 시스템.

청구항 15

제11항에 있어서,

상기 네트워크 동작에서의 장애는 상기 추정과 상기 관찰 간의 차가 상응하는 임계치를 초과하는지를 판단함으로써 탐지되는 시스템.

청구항 16

삭제

청구항 17

제11항에 있어서,

상기 관리자 모듈은 또한,

상기 추정과 상기 관찰 간의 차를 장애 크기로 변환하고,

상기 장애 크기가, 상기 네트워크 시뮬레이션에 의해 제공된 상기 추정을 상기 관찰에 근사하도록 움직이는지를 판단함으로써, 장애를 진단하도록 실행가능한 시스템.

청구항 18

제11항에 있어서,

상기 관리자 모듈은 또한,

상기 추정과 상기 관찰 간의 차를 장애 크기로 변환하고,

상기 장애 크기가 상응하는 임계치 미만인가를 판단하고,

상기 장애 크기가 상기 상응하는 임계치 미만인 경우, 상기 추정을 생성하기 위해, 생성된 상기 시뮬레이션에 의해 입력으로서 사용되는 장애 세트로부터 대응하는 상기 장애를 제거시킴으로써, 장애를 진단하도록 실행가능한 시스템.

청구항 19

제11항에 있어서,

상기 관리자 모듈은 또한,

초기 장애 세트를 생성하기 위한 초기 진단을 행하고,

상기 네트워크 시뮬레이션에 의해 입력으로서 사용될 때, 상기 추정이 상기 관찰에 근사하게 하는 현재 장애 세트에 도달하도록 상기 초기 장애 세트를 반복적으로 정련함으로써 상기 장애를 진단하도록 실행가능한 시스템.

청구항 20

제11항에 있어서,

상기 네트워크는 메쉬 네트워크인 시스템.

청구항 21

제11항에 있어서,

상기 네트워크 세팅은 상기 네트워크의 시작 노드와 대상 노드 사이에서 상기 네트워크를 통해 데이터 패킷에 의해 취해진 일련의 홉(hop)으로 형성된 라우트를 기술하는 시스템.

청구항 22

네트워크에서의 장애의 탐지 및 진단을 위한 수단을 갖는 상기 네트워크 내의 노드(102)로서,

데이터 패킷을 라우팅하는 복수의 수단을 갖는 네트워크의 동작을 관리하기 위한 수단(102(N))을 포함하며,

각각의 라우팅 수단은 다른 라우팅 수단과 통신가능하게 결합되며,

상기 관리 수단은,

상기 네트워크로부터 얻어진 네트워크 세팅을 상기 네트워크를 시뮬레이팅하기 위한 수단(210)에 제공하고,

상기 시뮬레이팅 수단으로부터 상기 네트워크의 네트워크 성능을 추정하는 출력을 수신하고,

상기 출력을 상기 네트워크의 네트워크 성능의 관찰과 비교함으로써(508), 장애를 탐지하기 위한 수단(510, 512, 514, 516)을 더 포함하며,

상기 탐지는, 상기 출력이 상기 네트워크의 네트워크 성능의 관찰에 근사할 때까지, 복수의 장애들 중 하나 이상을 상기 시뮬레이팅 수단에 삽입시킴으로써 상기 네트워크의 동작에서의 불일치의 근본 원인을 진단하는 것을 포함하는 노드.

청구항 23

제22항에 있어서,

상기 관리 수단은 상기 장애를 진단하기 위한 수단을 더 포함하는 노드.

청구항 24

제22항에 있어서,

상기 네트워크는 메쉬 네트워크인 노드.

청구항 25

제22항에 있어서,

상기 관리 수단은 탐지된 상기 장애에 기초하여 시정 액션(corrective action)을 도출해 내기 위한 수단을 더 포함하는 노드.

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

명 세 서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

[0021] 이 출원은 발명의 명칭이 "다중-홉 무선 네트워크에서의 장애 탐지, 분리, 및 진단"인 2004년 1월 30일자로 출원된 미국 가출원 명세서 제60/540,738호의 우선권을 주장한다.

[0022] 본 발명은 일반적으로 유선 네트워크 및 무선 네트워크에 관한 것으로, 특히 네트워크에서의 장애를 탐지하고 진단하기 위한 네트워크 중재 프레임워크에 관한 것이다.

[0023] 네트워크 관리가 비록 다중-홉(muti-hop) 무선 네트워크의 성공적인 개발에 있어서 핵심 사항이지만, 산업 및 연구 공동체에서는 제한적으로 대처해 왔다. 네트워크를 중재한다는 것은 그 네트워크의 "건강 상태(health)"를 유지 관리하며 순탄하고 지속적인 동작을 보장하는 역할을 수행하는 네트워크 관리의 한 양상이다. 네트워크의 중재는, 네트워크가 유선이든 무선이든 간에, 서로 다른 네트워크 엔티티 사이, 서로 다른 장애 등 사이에

서 봉착되어지는 대화에 의해 복잡해 진다.

[0024] 다중-홉 무선 네트워크에 대한 중재는 여러 추가의 요인에 의해 더욱 복잡해 질 수 있다. 예컨대, 전형적인 다중-홉 무선 네트워크는 일반적으로 신호 전파 파동으로 인한 링크 에러를 일으키기 쉽다. 이런 신호 전파 파동은 여러 요인에 의해 유발될 수 있는데, 예를 들면, 환경적인 조건의 변동이다. 이러한 파동으로 인해, 네트워크 형상(topology)이 동적이고 예측불허가 된다. 또한, 노드의 이동성도 이들 요인을 더욱 악화시키는 데, 이는 노드가 네트워크에 연결되어 있는 동안 여러 위치에 위치될 수 있어, 네트워크가 갖고 있는 동적이고 예측불허의 본질이 증가되기 때문이다. 또한, 다중-홉 무선 네트워크의 성능은 일반적으로 자원(예컨대, 대역폭, 배터리 전력 등)의 부족으로 인해 제한되는데, 이러한 자원 부족은 네트워크가 허용할 수 있는 관리 트래픽 오버헤드의 양을 제한시킨다. 게다가, 무선 네트워크는 악의있는 상대방으로부터의 링크 공격에 취약할 수 있다. 예컨대, 이런 공격자들은 오류 정보를 삽입시킬 수 있는 네트워크 관리 노력을 파괴시키거나 방해할 수 있다.

[0025] 네트워크 중재를 수행함에 있어 통상적으로 사용되어진 전통적인 발견적 학습법 및 이론적 기술로는, 통상적으로 "실제" 환경에서 구현되는 네트워크의 행동을 포착하지 못한다. 예를 들면, 네트워크 행동은 서로 간의 노드 대화 뿐 아니라, 노드 근방에 위치한 외부 잡음원에 의해서도 지배될 수 있다. 전통적인 발견적 학습법 또는 이론적인 기술로는 네트워크의 서로 다른 구성 요소들 간의 대화를 그 주위 환경에서 적절하게 다룰수 없으므로, 이러한 네트워크의 행동을 포착하지 못한다.

[0026] 따라서, 개선된 장애 탐지 및 진단을 제공하는 네트워크 중재를 위한 프레임워크가 필요로 된다.

발명이 이루고자 하는 기술적 과제

[0027] 네트워크 중재 프레임워크에 대해 기술하고자 한다. 이 프레임워크는 실제 네트워크의 시뮬레이션을 채용하여 그 실제 네트워크의 동작에서의 장애를 검출하고 진단할 수 있다. 예컨대, 네트워크 시뮬레이션은 실제 네트워크의 동작을 기술하는 데이터에 의해 구동될 수 있다. 실제로, 네트워크 시뮬레이션의 구동 시에 사용도록 수집된 미처리 데이터는 여러 가지의 이유로 인해, 예컨대, 하드웨어, 소프트웨어, 및/또는 네트워크 에러로 인한 에러들을 포함할 수 있다. 네트워크 시뮬레이션을 구동하는 데 사용되는 데이터의 일관성을 보장하기 위해, 미처리 데이터를 클리닝시킬 수 있다. 예를 들면, 네트워크 내의 각 노드는 네트워크 시뮬레이션의 구동 시에 사용하기 위한 데이터를 제공할 수 있다. 특정 노드에 의해 제공된 데이터는 그 특정 노드의 동작 뿐 아니라, 하나 이상의 이웃하는 노드들의 동작도 기술할 수 있다. 그러므로, 네트워크의 노드로부터 얻어진 데이터는 중복될 수 있다. 이로써, 중복된 데이터를 서로 비교하여 불일치되는 것들을 식별하며, 이러한 불일치된 것을 데이터 평균화, 불일치 데이터의 제거 등을 통한 여러 방법으로 조정될 수 있다.

[0028] 그 후, 네트워크 시뮬레이션은 이런 데이터에 기초하여 네트워크 성능을 추정할 수 있다. 추정된 네트워크의 성능을 실제 네트워크 성능의 관찰된 네트워크의 성능과 비교하여 실제 네트워크가 예기된 바와 같은 수행을 행하는 가를 탐지한다. 만약 예기된 바와 같은 수행을 행하지 않으면, 실제 네트워크의 동작에 장애가 있는 것으로 탐지한다. 환언하자면, 네트워크 시뮬레이션에 의해 나타난 추정된 네트워크 성능과 실제 네트워크에 의해 나타난 관찰된 네트워크 성능 간의 차를 활용함으로써 실제 네트워크에서의 장애 발생을 탐지할 수 있다. 다음에, 네트워크 시뮬레이션은 네트워크 시뮬레이션의 네트워크 성능이 실제 네트워크의 네트워크 성능에 근사하게 될 때까지 하나 이상의 장애를 네트워크 시뮬레이션 내로 선택적으로 삽입시킴으로써 장애 진단에 사용될 수 있다.

[0029] 일단 네트워크 성능을 근사화시키는 하나 이상의 장애 세트가 식별되면, 장애를 식별하기 위한 하나 이상의 변경이 식별되어 구현될 수 있다. 예를 들면, 시뮬레이티드된 네트워크에 대해 변경을 행하여 그 변경이 장애를 정정하고/정정하거나 네트워크 성능을 개선시킬지를 테스트하는 상황 분석(what-if analysis)을 수행하기 위해 네트워크 시뮬레이션을 사용할 수 있다. 이와 같이, 네트워크 시뮬레이션은 장애를 정정하고/정정하거나 네트워크 성능을 개선시키도록 행해진 변경과 같은, 네트워크에 대해 행해질 수 있는 여러 변경에 의한 네트워크 성능 영향에 대해 정량적(quantitative) 피드백을 제공할 수 있다.

[0030] **개관**

[0031] 유선 네트워크 및/또는 무선 네트워크에서 사용되어 네트워크의 동작을 효율적이고 신뢰성있게 유지 관리하는 네트워크 중재 프레임워크가 기술된다. 본원에서 기술된 프레임워크는 온 라인 트레이스 구동식(online trace-driven) 네트워크 시뮬레이션을 채용하여 장애를 탐지하여 그 장애에 대한 근본 원인 분석을 행한다. 이 네트

워크 시뮬레이션은 "실제" 네트워크로부터 네트워크 성능 데이터를 얻을 수 있다는 점에서 "온 라인"이다.

[0032] 본 발명의 프레임워크는 패킷 탈락, 링크 혼잡, 매체 액세스 제어(MAC)의 잘못된 행동, 외부 잡음 등으로 인한 장애와 같은, 다양한 종류의 성능 문제(즉, 장애)의 진단에 적용될 수 있다. 또한, 본 발명의 프레임워크는 네트워크 성능을 개선시키기 위해 다른 네트워크 구성을 평가하는 데도 사용될 수 있다. 이하의 기술에서는 예시적인 무선 네트워크에서의 프래임워크에 대해 기술하고 있지만, 프레임워크는 유선 네트워크에도 사용될 수 있다.

[0033] **예시적 환경**

[0034] 상술한 바와 같이, 네트워크 관리에 대해 산업 및 연구 공동체에서는 제한적으로 대처해 왔다. 네트워크 관리의 구현은 네트워크 기능에 대한 지속적인 모니터링, 네트워크의 노드 및 링크에 관한 정보 수집, 보고된 데이터로부터 불일치되는 것 및 잡음의 제거, 데이터의 분석, 네트워크 신뢰성 및 성능을 개선시키는 적절한 액손(actions)에 대한 성능을 포함할 수 있다.

[0035] 네트워크를 중재한다는 것은 그 네트워크의 "건강 상태(health)"를 유지 관리하고 순탄하고 지속적인 동작을 보장하는 역할을 수행하는 네트워크 관리의 한 양상이다. 네트워크의 중재는, 네트워크가 유선이든 무선이든 간에, 서로 다른 네트워크 엔티티 사이, 서로 다른 장애 등 사이에서 봉착되어지는 대화에 의해 복잡해 진다.

[0036] 다중-홉 무선 네트워크에 대한 중재는 여러 추가의 요인에 의해 더욱 복잡해 질 수 있다. 예컨대, 전형적인 다중-홉 무선 네트워크는 일반적으로 신호 전파 파동으로 인한 링크 에러를 일으키기 쉽다. 이런 신호 전파 파동은 여러 요인에 의해 유발될 수 있는데, 예를 들면, 환경적인 조건의 변동이다. 이러한 파동으로 인해, 네트워크 형상(topology)이 동적이고 예측불허가 된다. 또한, 다중-홉 무선 네트워크의 성능은 일반적으로 자원(예컨대, 대역폭, 배터리 전력 등)의 부족으로 인해 제한되는데, 이러한 자원 부족은 네트워크가 허용할 수 있는 관리 트래픽 오버헤드의 양을 제한시킨다. 게다가, 무선 네트워크는 악의있는 상대방으로부터의 링크 공격에 취약할 수 있다.

[0037] 이들 복잡성을 해결하는 프레임워크가 기술된다. 이 프레임워크는 온 라인 트레이스 구동형 시뮬레이션을 사용하여 장애를 탐지하여 그 근본 원인을 분석한다. 이 시뮬레이션을 사용하여 장애를 일으킨 네트워크에 위치된 이벤트들을 재현함으로써 이들 장애를 식별하여 조정할 수 있다.

발명의 구성 및 작용

[0038] 도 1은 복수의 노드(102(1), 102(2), 103(3),…, 102(n),…, 102(N))를 갖는 네트워크(100)를 도시하는 예시적인 구현예에서의 환경을 도시한다. 도 1의 복수의 노드(102(1) 내지 102(N))는 장애 탐지 진단, 및 상황 분석을 위해 네트워크(100)의 시뮬레이션을 이용하는 예시 프레임워크를 구현한다. 이 프레임워크는 다양한 이로운 특성들을 갖는다. 첫 째, 이 프레임워크는 유연성을 갖는다는 것이다. 시뮬레이션은 고도로 커스터마이징가능하여 서로 다른 환경에서 구현되는 광범위한 종류의 네트워크에 적용될 수 있으므로, 시뮬레이터의 상부에 구축되는 장애 진단은 이러한 유연성을 상속받도록 구성될 수 있다. 둘 째, 시뮬레이션은 복잡한 여러 대화들을 포착할 수 있도록 하게 해 준다. 예를 들면, 네트워크 내에서, 네트워크와 환경 사이, 또한 네트워크의 동작 중에 발생하는 서로 다른 장애들 사이에서의 대화들을 포착할 수 있다. 그러므로, 프레임워크는 이러한 시뮬레이션의 이용을 통해 광범위한 종류의 장애들, 및 이들의 조합을 포함해서 이들 장애에 대한 체계적인 진단을 제공한다. 셋 째, 프레임워크는 시뮬레이션에서의 장애들을 시스템에서의 다른 장애들과 독립적으로 모델링함으로써 새로운 장애들을 탐지할 수 있는 기능을 프레임워크 내에 구축시킬 수 있다는 점에서 확장성을 갖는다. 새로운 장애와 프레임워크에서 모델링된 기존의 장애들 간에서의 대화는 시뮬레이션의 실행을 통해 암시적으로 포착된다. 넷 째, 시뮬레이터 내에서의 네트워크의 재현은 상황 분석을 용이하게 하며, 이는 네트워크에 대해 행해질 수 있는 변경의 성능 영향에 대한 정량적 피드백을 제공한다. 예컨대, 네트워크의 동작 장애를 정정하기 위한 시정 액션(corrective action)이 취해질 수 있으며, 네트워크의 성능 등을 향상시키기 위한 변경이 행해질 수 있다.

[0039] 프레임워크는 여러 종류의 기존의 네트워크 시뮬레이터 중 하나 이상을 사용하여 QUALNET(QUALNET는 미국 캘리포니아주 로스앤젤레스 소재의 Scalable Network Technologies, Inc.의 등록 상표임), OPNET MODELER(OPNET MODELER는 미국 워싱턴 D. C. 소재의 OPNET Technologies, Inc.의 등록 상표임) 등의 네트워크(100)를 시뮬레이팅할 수 있다. 시뮬레이터에 제공되는 트레이스들(traces)은 진단되는 네트워크로부터, 즉 "실제" 네트워크로부터 얻어진다. 실제 네트워크로부터의 트레이스를 사용함으로써, 당해 특정 네트워크의 하드웨어, 소프트웨어, 및 환경에서의 미묘한 차이(nuances)를 포착할 수 없는 포괄적인 이론 모델에 대한 프레임워크의 종속성이

제거되어, 프레임워크의 정확성을 향상시킨다.

[0040] 또한, 프레임워크는 근본 원인 분석을 수행하는 장애 진단 구성(scheme)을 채용할 수 있다. 예컨대, 이런 구성은 실제 네트워크의 예기된 성능에 대한 기준선으로서 온 라인 트레이스 구동식 시뮬레이터에 의해 보내진 추정된 네트워크 성능 데이터를 사용할 수 있다. 그런 다음, 예기된 성능으로부터의 편차를 이용하여 잠재적인 장애를 나타낼 수 있다. 또한, 이런 구성은 시뮬레이터에 한 세트의 후보 장애들을 선택적으로 삽입시켜 한 세트의 장애를 탐색하는 문제에 대한 장애 진단을 줄임으로써 근본-원인 분석을 행할 수 있다. 그러므로, 근본 원인은 삽입될 때 시뮬레이션에게 실제 네트워크의 관찰된 성능에 근사토록 하게 하는 장애에 근거하여 식별될 수 있다. 따라서, 프레임워크는 패킷 탈락, 링크 혼잡, 외부 잡음원, MAC 오동작 등의 장애들을 탐지하여 진단하는 탐색 알고리즘을 채용할 수 있다. 이들 장애들은 성능에 대해 비교적 길고 지속적으로 영향을 미칠 수 있으므로, 노드가 전력 또는 배터리 중단으로 인해 그 자신이 턴 오프될 때 등과 같은 고장에 의한 중단(fail-stop) 에러보다도 탐지함에 있어 더 곤란하다.

[0041] 이와 같이, 프레임워크는 네트워크에서 대체적이고 잠재적인 성능 향상 구성을 중재하고 테스트하는 분석 도구로서 시뮬레이션을 사용할 수 있다. 이하의 기술에서는, 시뮬레이터에 제공될 때 실제 네트워크 행동을 정확하게 묘사하는 네트워크 시뮬레이션을 제공하는 네트워크 트레이스들이 식별된다. 또한, 트레이스로부터 에러 데이터를 감축시키거나 제거시키는 기술을 기재하고 있으며, 이에 대한 보다 상세한 설명은 도 8 및 도 9를 참조하면서 기술될 것이다. 결과적으로 시뮬레이터에는 고 품질의 데이터가 공급된다. 추가로, 네트워크에서 다수의 장애들을 진단하는 데 효과적인 탐색 알고리즘이 기술되며, 이에 대한 보다 상세한 설명은 도 7에 관련하여 설명하기로 할 것이다. 시뮬레이터는 또한, 상황 분석을 수행하여 현재 네트워크에 대해 일어날 수 있는 액션들의 성능 이익을 정량화시키는 데 사용될 수 있으며, 이에 대한 보다 상세한 설명은 도 10 내지 도 13에 관련하여 설명하기로 할 것이다.

[0042] 중재용 프레임워크는 광범위한 종류의 네트워크 구성에서 사용될 수 있다. 한 예로서는 무선 메쉬 네트워크로서 도시된 도 1의 네트워크(100)가 있다. 메쉬 네트워크는 완전(full) 메쉬 형상 또는 부분(partial) 메쉬 형상 등의 다양한 구성을 가질 수 있다. 완전 메쉬 형상에서, 각 노드는 네트워크의 다른 각 노드에 직접 연결되어진다. 부분 메쉬 형상에서, 각 노드는 적어도 하나의 다른 노드에 연결되지만, 네트워크의 다른 각 노드에 반드시 연결될 필요는 없다.

[0043] 메쉬 네트워크는, 예를 들어, 이웃들이 자기 관리형 공동체 무선 메쉬 네트워크를 협력적으로 형성하는 인에이블링 기술로서 사용될 수 있다. 각 이웃들은 네트워크(100)의 복수의 노드(102(1) 내지 102(N)) 중 하나 이상을 제공할 수 있다. 이런 네트워크의 사용에 의해, 이웃들은, 예컨대, 비용면에서 효율적으로 인터넷 게이트웨이(104)를 공유할 수 있다.

[0044] 메쉬 네트워크가 이웃에서 사용되는 일예에서는, 복수의 노드(102(1) 내지 102(N))를 통신가능하도록 결합하는데 사용되는 라우터들은 가정(home) 내에 상주하며 전기 콘센트에 플러그 인된다. 그러므로, 이 예에서의 라우터들 각각은 제한된 이동성을 갖게 된다. 그러나, 이런 네트워크의 상대적인 안정성은 네트워크 중재를 보다 중요하게 하는 데, 이는 장애가 네트워크 성능에 지속적인 영향을 미칠 수 있기 때문이다. 그러나, 주목해야 할 것은, 이 예에서의 라우터 이동성의 부족으로 인해서도 네트워크 형상에서의 역동성(dynamism)은 감소되지 않는다는 점인 데, 이는 무선 링크가 환경 변화로 인해 액세스가능하거나 또는 액세스 불가능할 수 있기 때문이다. 다른 예에서, 메쉬 네트워크의 노드는 무선 통신 기능을 갖는, 개인 휴대 단말기(PDA), 태블릿 퍼스널 컴퓨터(PC), 랩톱 컴퓨터 등의 이동형 컴퓨팅 장치의 사용을 통해 이동성을 가질 수 있다.

[0045] 추가로, 공동체 메쉬 네트워크는 사용자들이 메쉬 네트워크에 가입하기 위한 장비를 구매하여 설치할 때 유기적으로 생성된다. 전통적인 메쉬 네트워크는 네트워크 관리에 책임있는 집중화된 엔티티가 결여되어 있다. 그러나, 본원에서 기술된 프레임워크를 통해 제공되는 자기 관리 기능 및 자기 치유 기능을 제공하여 각 노드(102(1) 내지 102(N))는 중재 기능을 구현할 수 있다. 예시된 구현예에서는, 관리 기능을 갖는 단일 노드가 제공되어 있다.

[0046] 도 1에 도시된 네트워크(100)에서, 각 노드는 프로세서, 메모리, 및 네트워크 접속 장치를 포함하는데, 예를 들면, 노드(102(n)는 프로세서(106(n)), 메모리(108(n)), 및 네트워크 접속 장치(110(n))를 포함하는 것으로 도시되어 있다. 프로세서(예를 들어, 프로세서(106(n), 106(N))는 이들을 구성하는 구성 성분 또는 내장된 처리 메커니즘에 의해 제한되지 않는다. 예를 들어, 프로세서는 반도체 및/또는 트랜지스터(예컨대, 전자 집적 회로(ICs))로 구성될 수 있다. 이런 맥락에서 보면, 프로세서-실행가능 명령어들은 전자적으로 실행가능한 명령어일 수 있다. 또는, 프로세서 및 노드를 위한 또는 프로세서 및 노드의 메커니즘은 양자 컴퓨팅, 광학적

컴퓨팅, 기계적 컴퓨팅(예컨대, 나노기술을 사용함) 등을 포함하지만, 이에만 국한되는 것은 아니다.

[0047] 메모리(예컨대, 메모리(108(n), 108(N))는 판독 전용 메모리(ROM), 랜덤 액세스 메모리(RAM) 등의 휘발성 및/또는 비휘발성 메모리의 형태로 컴퓨터 저장 매체를 포함한다. 또한, 메모리는 기타 이동가능형/이동 불가능형, 휘발성/비휘발성 컴퓨터 저장 매체를 포함할 수 있다. 메모리는 노드들의 컴퓨터 판독가능 명령어, 데이터 구조, 소프트웨어 컴포넌트, 및 가타 데이터를 저장한다.

[0048] 네트워크 접속 장치(예를 들어, 네트워크 접속 장치(110(n), 110(N))는 네트워크(100)에 노드들을 통신가능하게 결합시키는 여러 구성을 취할 수 있다. 예컨대, 근거리 통신망(LAN) 환경에서 사용할 경우, 노드(102(n))는 유선 및/또는 무선일 수 있는 네트워크 인터페이스인 어댑터를 통해 LAN에 통신가능하게 접속된다. 원격 통신망(WAN)에서 사용할 경우에는, 네트워크 접속 장치는 모뎀이나, 또는 디지탈 가입자 회선(DSL)을 통한 유선 접속, 위성을 갖춘 무선 접속 등의 통신을 확립하기 위한 기타 수단으로 구성될 수 있다. 도 1의 화살표에 의해 논리 접속을 도시하고 있다. 도 1에서 도시된 네트워크(100)는 무선 메쉬 네트워크로 도시되었지만, 인터넷, 인트라넷 등의 여러 유형의 다른 네트워크가 사용될 수 있다.

[0049] 노드(102(n), 102(N))는 소프트웨어 모듈로 이루어진 예시적인 관리 아키텍처를 도시하고 있다. 일반적으로, 본원에서 기술된 어떠한 기능이라도 소프트웨어, 펌웨어(예를 들어, 정해진 논리 회로), 수동 처리, 또는 이들 구현의 조합을 사용하여 구현될 수 있다. 본원 명세서에서 기술된 "모듈", "기능"(functionality)", 및 "논리(logic)"라는 용어는 일반적으로 소프트웨어, 펌웨어, 또는 소프트웨어와 펌웨어의 결합을 나타내는 것이다. 소프트웨어 구현의 경우, 모듈, 기능, 또는 논리는 하나 이상의 중앙 처리 장치(CPUs) 등의 프로세서 상에서 실행될 때 특정 타스크를 수행하는 프로그램 코드를 나타낸다. 이 프로그램 코드는 하나 이상의 컴퓨터 판독가능 메모리 장치에 저장될 수 있다. 이하 기술된 프레임워크의 특징은 플랫폼 독립형으로서, 이는 중재 기술이 각종의 프로세서를 갖는 시판중의 여러 컴퓨팅 플랫폼 상에서 구현될 수 있음을 의미한다.

[0050] 에이전트 모듈((112(n))은 네트워크(100)의 각 노드(102(n)) 상에서 실행되도록 제공된다. 에이전트 모듈((112(n))은 프로세서(106(n)) 상에서 실행되는 것으로 도시되어 있으며, 메모리(108(n))에 저장가능하다. 에이전트 모듈((112(n))은 데이터 수집 모듈(114(n))(이하, "수집 모듈"이라 함)을 포함하며, 이 모듈은 실행될 때 각 종 프로토콜 계층 및/또는 네트워크 접속 장치(110(n))로부터 데이터를 수집할 수 있다. 도 1에 예시된 네트워크(100)에서, 에이전트 모듈((112(n))은 관리 기능을 갖는 노드(102(N))(이하, 관리자 노드라 함)에 이들 데이터를 보고한다. 관리자 노드(102(N))는 이들 데이터에 대해 분석을 행하여(예를 들어, 입력으로서 이들 데이터를 수락하는 시뮬레이션의 구현을 통해), 네트워크를 중재하기 위한 적절한 조치를 취한다. 네트워크의 관리는 도 1의 네트워크(100)에 도시된 바와 같이 단일 노드 상에 관리자를 위치시켜 중앙 집중화시킬 수 있거나, 또는 네트워크의 복수 노드 각각이 관리 기능을 갖도록 분산시킬 수 있다.

[0051] 에이전트 모듈(112(n), 112(N))은 각각의 프로세서들(106(n), 106(N)) 상에서 실행될 때, 네트워크 행위에 대한 그들의 (로컬) 관점을 설명하는 데이터를 관리자 노드(120(N))에 수집 및 전달한다. 송신된 데이터의 예로는, 트래픽 통계, 수신된 여러가지 링크의 패킷 신호 강도, 각각의 링크에 대한 재전송 카운트 등이 있다.

[0052] 관리자 노드(120(N))는, 네트워크(100)의 장애를 해소하기 위한 에이전트(112(n), 112(N))로부터의 데이터를 처리하기 위해, 메모리(108(N))에 저장가능하고, 프로세서(106(N)) 상에서 실행가능한 관리자 모듈(116(N))을 포함한다. 관리자 모듈(116(N))은 네트워크(100)를 시뮬레이트하기 위해, 예를 들어, 프로세서(106(N)) 상에서 실행가능하고 메모리((108(N))에 저장가능한 네트워크 시뮬레이터(118(N); 이하 시뮬레이터라고 지칭함)를 포함한다.

[0053] 관리자 노드(102(N))에 의해 여러 에이전트들(112(n), 112(N))로부터 수신된 데이터는 네트워크(100)의 불일치된 관점을 일으킨다. 그러한 불일치는 형상 및 환경적 변화, 측정 에러, 예상 외의 행위를 하는 노드 등의 결과일 수 있다. 따라서, 관리자 노드(102(N))는 그러한 불일치를 해소하기 위해, 프로세서(106(N)) 상에서 실행가능한 데이터 클리닝 모듈(120(N); 이하 "클리닝 모듈"이라 지칭함)을 포함한다. 클리닝 모듈(120(N))로부터의 클리닝된 데이터 출력은 근본 원인 분석 모듈(122(N); 이하 "분석 모듈"이라고 지칭함)에 의한 처리에 제공되며, 이에 관하여는 다음의 도면과 관련하여 더 논의될 것이다. 관리 노드(102(N))는 에이전트 모듈(112(N)) 및 관리 모듈(116(N))을 포함하는 것으로 설명되고 있지만, 다른 구현에서, 관리자 노드(102(N))는 에이전트 모듈(112(N))을 포함하지 않는 전용 관리 노드이다. 또한, 상술된 바와 같이, 관리자 모듈(116(N))의 기능은 네트워크(100) 내의 하나 이상의 노드에 의해 제공될 수 있다.

[0054] 도 2는 도 1의 분석 모듈(122(N))을 더 상세하게 나타내는 예시적인 구현예(200)를 도시한다. 데이터 내의 불

일치가 도 1의 클리닝 모듈(120(N))에 의해 해소되었다면, 클리닝된 데이터는 추가 조사를 위해 분석 모듈(122(N))에 제공된다.

[0055] 분석 모듈(122(N))은 온라인 트레이스-구동식 시뮬레이션을 사용하여, 시뮬레이트된 네트워크 성능에 의해 나타난 바와 같이, 예상된 네트워크 성능으로부터의 불일치의 근본 원인을 결정한다. 다음의 논의에서, 예상된 네트워크 성능 및 시뮬레이트된 네트워크 성능은 상호교환가능하게 사용되어, 네트워크 시뮬레이션에 의해 제공되는 네트워크 성능을 나타낸다. 분석 모듈(122(N))은 트레이스 유틸리티로부터 획득된 클리닝된 데이터(202)를 사용하여, 소정의 네트워크 구성과 트래픽 패턴 하에서 온라인 시뮬레이션을 구동하고 예상된 성능을 구축할 수 있고, 그러한 데이터의 예는 도 2에서 링크 수신된 신호 강도(RSS; 204), 링크 위치(206) 및 라우팅 업데이트(208)로서 도시되어 있다.

[0056] 분석 모듈(122(N))은 시뮬레이터(118(N))의 실행을 통해 제공되는 네트워크 시뮬레이션(210)을 포함하는 것으로 도시된다. 네트워크 시뮬레이션(210)은 네트워크 특성의 시뮬레이션을 제공하는 하나 이상의 소프트웨어 모듈의 실행에 의해 제공될 수 있으며, 이 예는 도 2에서 간섭 주입 모듈(212), 트래픽 시뮬레이터 모듈(214) 및 형상 변화 모듈(216)로 도시된다. 간섭 주입 모듈(212)은 네트워크 시뮬레이션(210)에 외부 잡음 효과를 주입함으로써 외부 잡음 소스를 시뮬레이트하는 데 실행될 수 있다. 트래픽 시뮬레이터 모듈(214)은 네트워크 시뮬레이션(210)의 트래픽이 실제 네트워크의 트래픽에 근사하다는 것을 보장하기 위해 실행될 수 있다. 형상 변화 모듈(216)은 예를 들어, 네트워크 시뮬레이션(210) 내의 노드를 추가하고/추가하거나 제거함으로써 형상의 변화를 시뮬레이트하는 데 실행될 수 있다.

[0057] 분석 모듈(122(N))은 네트워크 시뮬레이션(210)에 의해 나타난 예상된 성능과 관찰된 성능을 비교함으로써, 도 1의 네트워크(100) 내의 장애를 탐지한다. 불일치가 관찰되면, 분석 모듈(122(N))은 시뮬레이트된 네트워크 성능과 관찰된 네트워크 성능 간의 최상의 매치를 산출하는 장애 디렉토리(218)에 저장된 하나 이상의 장애를 검색함으로써, 불일치에 대한 근본 원인을 결정한다.

[0058] 분석 모듈(122(N))은 예를 들어, 손실율, 처리율 및 잡음(220)을 설명하는 도 1의 에이전트 모듈(112(n))들 중 하나 이상의 모듈로부터 관찰된 데이터(220)를 수신할 수 있고, 이것은 도 2에서 "손실율, 처리율 및 잡음(220)"으로 도시되어 있다. 네트워크 시뮬레이션(210)은 예상된 손실율, 예상된 처리율 및 예상된 잡음을 설명하는 예상된 데이터(222)를 계산하고, 이것은 도 2에서 "예상된 손실율, 처리율 및 잡음(222)"으로 도시되어 있다. 관찰된 데이터(220)는 지연(224)을 통해 비교기(226)로 전달되어, 비교기(226)가 관찰된 데이터와 예상된 데이터(220 및 222)를 동시에 수신하도록 한다. 비교기(226)는 관찰된 데이터(220)가 예상된 데이터(222)를 초과하는 지를 판정한다. 초과한다면, 비교기(226)는 네트워크 관리자에게 전달하기 위해 에러 메시지(228)를 출력하고, 그 에러를 에러의 근본 원인을 결정하는 장애 디렉토리(218)에 전달한다.

[0059] 장애 디렉토리(218)로부터의 장애들 중 하나 이상을 선택함으로써 에러의 근본 원인이 식별된 후에, 분석 모듈(122(N))은 장애를 정정하기 위한 하나 이상의 대안적인 동작을 시뮬레이트할 수 있다. 대안적인 동작은 트래픽 시뮬레이터(214)와 형상 변화 모듈(216)에 의해 각각 제공되는 현재 트래픽 패턴과 네트워크 형상 하에서 시뮬레이트될 수 있다. 이 시뮬레이션에 기초하여, 분석 모듈(122(N))은 장애를 완화하고 네트워크의 전반적인 성능을 강화하는 하나 이상의 적합한 동작을 제시할 수 있으며, 이 예는 도 2의 링크 노드 장애(230)로서 도시된다. 예로서, 소프트웨어 또는 하드웨어가 장애가 있는 것으로 의심되는 경우 네트워크 운영자에게 통지할 수 있고, 불량 접속이 탐지되는 경우 전송-전력 조정을 통해 형상을 변경할 수 있고, 정체를 완화하기 위해 라우터가 비율 한계를 사용할 수 있는 등의 동작이 있다.

[0060] 온라인 진단을 위한 네트워크 시뮬레이션(210)의 사용은 통상적인 휴리스틱(heuristic) 또는 이론적인 진단 기법에 대해 여러가지 이점을 제공한다. 예를 들면, 네트워크 시뮬레이션(210)은 통상적인 휴리스틱 또는 이론적인 진단 기법에 대하여 네트워크의 행위에 향상된 통찰을 제공할 수 있다. 운영적 무선 네트워크는 예를 들어, 트래픽 흐름, 네트워킹 프로토콜, 신호 처리 알고리즘, 하드웨어, 무선 주파수(RF) 전달 등의 난해한 부분들을 갖는 복잡한 시스템이다. 또한, 네트워크의 부분들 모두 간에는 상호작용이 일어날 수 있다. 장애들 간의 상호작용은 "실제" 네트워크의 실제 행위에 대응하는 네트워크 시뮬레이션(210)을 일으키는 장애 디렉토리(218)로부터의 하나 이상의 장애들의 선택을 통해, 효과적으로 진단되고 처리될 수 있다.

[0061] 또한, 네트워크 행위는 서로 간의 노드 상호작용 뿐만 아니라, 노드 주변에 위치한 외부 잡음 소스에 의해 제어될 수 있다. 통상적인 휴리스틱 또는 이론적 기법은 그러한 네트워크의 행위를 캡처하지 않고, 네트워크의 상이한 컴포넌트들 간의 상호작용을 적절히 처리하지 않는다.

[0062] 예를 들어, 도 3에 도시된 7×3 그리드 형상 네트워크(300)를 생각해보자. 네트워크(300)에는 5개의 흐름이 도시되어 있고, 이 5개의 흐름은 F_1(302), F_2(304), F_3(306), F_4(308) 및 F_5(310)로 도시되어 있다. 이 실시예에서, 흐름들(302 내지 310) 각각은 통신하기 위해 유사한 양의 트래픽을 갖는다. 예를 들어, 흐름들(302 내지 310) 각각은 개별 애플리케이션으로부터 대체로 유사한 양의 데이터를 수신한다.

[0063] 또한, 이 예에서, 조정 노드는 서로를 "들을" 수 있고, 간섭 범위는 통신 범위의 두배이다. 노드 A(312)와 노드 O(314) 간의 트래픽은 예를 들어, 노드 C(316)와 노드 Q(318) 간의 트래픽을 간섭한다. 마찬가지로, 노드 G(320)와 노드 U(322) 간의 트래픽은 노드 E(324)와 노드 S(326) 간의 트래픽을 간섭한다. 그러나, G(320)와 U(322) 간의 트래픽, 및 노드 A(312)와 O(314) 간의 트래픽은 노드 D(328)와 R(330) 간의 트래픽을 간섭하지 않는다.

[0064] 다음 표는, 각각의 흐름이 11 Mbps 비율의 일정한 비트율(CBR)을 송신할 때의 흐름(302 내지 310)의 처리율의 예를 나타낸다.

[0065]

F_1	F_2	F_3	F_4	F_5
2.50 Mbps	0.23 Mbps	2.09 Mbps	0.17 Mbps	2.53 Mbps

[0066] 상기 테이블에서 나타나는 바와 같이, 흐름 F_3(306)은 흐름 F_2(304) 및 F_4(308)보다 더 높은 처리율을 수신한다. 다시 말하면, 흐름 F_3(306)은 네트워크(300)의 다른 흐름들보다 더 높은 영역의 대역폭을 소비한다는 것이다.

[0067] 통상적으로, 휴리스틱의 애플리케이션은 흐름 F_3(306)이 과도하게 큰 대역폭 공유를 수신한다는 결론에 도달할 수 있다. 그러나, 온라인 트레이스-구동식 시뮬레이션의 사용을 통해, 관리자 노드(102(N))는 이것이 정상 행위라고 결론지을 수 있다. 예를 들어, 네트워크 시뮬레이션은 링크 품질을 고려하여, 흐름 F_1(302)와 F_5(310)가 흐름 F_2(304)와 F_4(308)를 간섭한다고 판정할 수 있다. 따라서, 흐름 F_3(306)은 흐름 F_2(304)와 F_4(308)와는 다르게, 흐름 F_1(302)와 F_5(310)로부터의 간섭 부족으로 인해, 추가적인 대역폭을 제공받는다. 이러한 방식으로, 시뮬레이션은, 모든 흐름이 동일한 애플리케이션-레벨 송신율를 가질 수 있어도 관찰된 처리율이 예상된다는 것을 판정할 수 있다. 그러나, 단순한 휴리스틱에서는 노드 D(328)와 R(330)이 예상 외의 행동을 한다는 잘못된 결론이 나올 수 있다.

[0068] 네트워크 시뮬레이션은 분석 모듈(122(N))에 의해 사용되어, 현재 트래픽 흐름과 링크 품질이 주어진 네트워크로부터 "예상되는 것"을 인지함으로써 네트워크를 관리한다. 다시 말하면, 분석 모듈(122(N))은 네트워크 시뮬레이션에 의해 제공되는 추정에 기초하여 무엇이 정상 행위로 구성되는 지를 말할 수 있다. 상술한 예에서, F_3(306)은 네트워크(300) 내의 다른 흐름들보다 더 많은 네트워크(300) 대역폭 공유를 사용하고 있지만, 이 행위는 예상되기 때문에, 이 노드는 관리자 모듈에 의해 장애로서 플래깅되지 않을 것이다. 관찰된 행위가 예상된 행위에서 벗어나면, 관리자 모듈은 그 이탈의 근본 원인을 결정하기 위해 도 2의 장애 디렉토리(218)를 사용하는 장애 검색 알고리즘을 호출할 수 있다.

[0069] 또한, 특정 환경 및 구성 하에서 특정 유형의 네트워크에 전통적인 서명-기반 또는 규칙-기반 장애 진단 접근법을 적용하는 것이 가능할 수 있지만, 단순한 서명 또는 규칙은 일반적인 설정에서 장애 진단에 대한 내재적 복잡도를 캡처하는 데 불충분하다. 이와는 다르게, 시뮬레이터는 주문가능하고, 적절한 파라미터 설정으로, 상이한 환경에서 사용하기 위해 구성되는 큰 네트워크 클래스에 적용될 수 있다. 그러한 시뮬레이터의 상단에 구축된 장애 진단은 이러한 일반성을 상속한다.

[0070] 또한, 시뮬레이션-기반 접근법의 다른 장점은 상황 분석을 수행하는 기능이다. 즉, 시뮬레이터에서 설정을 변경하거나 특정 동작을 수행함으로써, 시뮬레이터는 가상의 시나리오에 대한 성능을 예측할 수 있다. 이 데이터에 기초하여, 관리자 모듈은 에이전트 모듈(예를 들면, 도 1의 에이전트 모듈(112(n)))에게 적절한 동작을 취하여 네트워크의 성능을 최적화하라고 지시할 수 있다. 상술된 바와 같이, 네트워크 내의 다수의 요인들의 상호작용으로 인해 교정 동작의 결과를 예상하는 것이 어려울 수 있기 때문에, 그러한 상황 분석은 가치가 있다. 예를 들어, 송신기 전력이 증가되어 링크 품질을 개선시킬 수 있지만, 그 증가는 또한, 네트워크 내의 다른 노드들에게 영향을 끼치는 추가적인 간섭을 생성할 수 있다.

[0071] **장애 탐지 및 진단**

[0072] 실제 네트워크의 기능에 근사한 시뮬레이터(예를 들면, 네트워크 시뮬레이션(210)) 내부 환경의 생성을 제공하는 시뮬레이션-기반 진단 접근법이 설명된다. 그러면 생성된 환경(즉, 네트워크 시뮬레이션)은 실제 네트워크의 예상된 행위를 결정할 뿐만 아니라, 실제 네트워크의 동작에서 불일치가 일어나는 때를 결정하는 데 사용될 수 있다. 이 불일치의 근본 원인을 찾기 위해서, 관리자 모듈이 장애 공간 상에서 검색하도록 실행되어, 어느 장애 또는 장애 집합이 실제 네트워크에서 관찰된 네트워크 성능에 근사한 네트워크 성능을 재생산할 수 있는지를 결정한다. 시뮬레이트된 네트워크는 네트워크 형상, 라우팅 행위, 실제 네트워크에서 관찰된 트래픽 패턴 등의 여러가지 네트워크 양상을 재생산할 수 있다.

[0073] 구축 블럭으로서 온라인 트레이스-구동식 시뮬레이션을 사용하여, 장애에 대한 근본-원인을 찾도록 실행가능한 진단 알고리즘이 설명된다. 예를 들어, 진단 알고리즘은 우선, 소정의 장애 집합 하에서 네트워크의 성능을 추정할 수 있다. 그 추정된 성능과 관찰된 성능 간의 차이에 기초하여, 진단 알고리즘은 임의의 관찰된 불일치를 재생산하기 위해 장애 공간을 검색한다. 일 구현예에서, 진단 알고리즘은 동일한 유형(예를 들어, 네트워크 형상)의 다수의 장애 뿐만 아니라, 다수의 유형의 장애들(예를 들어, 잡음 및 형상)의 존재를 진단할 수 있다.

[0074] 장애는 시뮬레이션을 구동하는 데 사용된 트레이스 데이터가 에러를 포함할 때에도 진단될 수 있다. 예를 들면, 도 1의 에이전트 모듈(112(n))에 의해 제공되는 데이터는 여러가지 이유로 측정 에러, 잘못된 정보, 노드(102(n)) 실행 시의 소프트웨어/하드웨어 에러, 네트워크 통신 에러 등과 같은 에러를 포함할 수 있다. 클리닝 모듈(120(n))은 관리자 노드(102(N))에 의해 트레이스로부터 오류 데이터를 감소시키거나 제거하도록 실행되어, 품질 트레이스 데이터가 시뮬레이션-기반의 장애 진단을 구동하는 데 사용되도록 한다. 클리닝 모듈(120(N)) 실행에 대하여는 도 8 내지 9를 참조하여 더 논의된다.

[0075] **트레이스-구동식 시뮬레이션**

[0076] 도 4는 도 2의 시뮬레이터(118(N))와 네트워크 시뮬레이션(210)을 포함하는 시스템(400)을 나타내는 예시적인 구현예를 도시한다. 실제 네트워크의 동작을 통해 획득된 트레이스 데이터는 시뮬레이터(118(N))가 실제 네트워크의 네트워크 동작을 적절히 표현하고 실제 네트워크 상의 소정의 장애 집합의 효과를 검증할 수 있게 한다. 여러가지 트레이스 데이터는 시뮬레이터로의 입력을 위해 수집될 수 있으며, 그 예는 다음에서 설명된다.

[0077] 네트워크 형상(402)

[0078] 네트워크 형상(402) 데이터는 어느 노드들이 현재 네트워크의 멤버인 지와 그 노드들 간의 대응하는 링크들과 같은 네트워크 형상을 설명한다. 네트워크 내의 각 노드는 예를 들어, 이웃하는 노드들, 및 노드의 하나 이상의 라우팅 테이블에서 참조된 노드들의 상태(예를 들면, 접속 또는 접속해제)에 대해 보고하도록 구성될 수 있다. 이러한 방식으로, 네트워크의 노드 멤버쉽은 도 1의 관리자 노드(102(N))에게 전달될 수 있다. 일 구현예에서는, 이웃 또는 라우트에서의 변화 만이 보고된다. 이 데이터는 라우트 시뮬레이션을 구동하는 데 사용될 수 있고, 이것은 도 4의 라우트 시뮬레이터와 관련하여 더 상세하게 설명된다.

[0079] 트래픽 통계(404)

[0080] 트래픽 통계(404) 데이터는 네트워크를 통해 전달되는 데이터의 양과, 그 데이터를 전달하는 특정 노드들을 설명하는 데 사용될 수 있다. 트래픽 통계(404)는 도 2의 트래픽 시뮬레이터 모듈(214)에 의해 입력으로서 사용될 수 있어, 네트워크 시뮬레이션(210)이 실제 네트워크의 트래픽 흐름에 근사한 트래픽 흐름을 갖도록 한다. 네트워크의 노드 각각은 이것의 바로 이웃들에게 송신하고 이 이웃들로부터 수신한 트래픽의 규모를 설명하는 하나 이상의 카운터를 보유할 수 있다. 이 데이터는 트래픽 시뮬레이션 모듈(214)에 의해 제공된 라우트 트래픽 시뮬레이션을 구동하는 데 사용되며, 이것은 또한, 도 4와 관련하여 더 상세하게 설명된다.

[0081] 물리적 매체(406)

[0082] 물리적 매체(406) 데이터는 네트워크를 구현하는 데 사용되는 물리적인 매체의 네트워크 성능에 미치는 효과를 설명할 수 있다. 예를 들면, 무선 네트워크에서, 각각의 노드는 그것의 잡음 레벨과, 이웃하는 노드들로부터의 무선 링크의 신호 강도를 보고할 수 있다. 일 구현예에서, 신호 강도의 변화는 시간 평균, 표준 편차 또는 다른 통계 집합을 통해 주기적으로 캡처된다.

[0083] 네트워크 동작(408)

[0084] 네트워크 동작(408) 데이터는 실제 네트워크의 네트워크 동작(408)을 설명한다. 상술된 바와 같이, 네트워크

동작 불일치를 탐지하기 위해, 관찰된 네트워크 동작은 네트워크 시뮬레이션으로부터 추정된 네트워크 동작 출력과 비교된다. 네트워크 동작은 링크 동작과 엔드-투-엔드(end-to-end) 동작을 둘다 포함할 수 있고, 이 둘은 패킷 손실율, 지연 및 처리율과 같은 여러가지 메트릭을 통해 측정될 수 있다. 다음의 설명은 링크 레벨 동작에 초점을 둔다.

[0085] 데이터 수집은 (1) 로컬 노드에서 원시 성능 데이터를 수집하는 단계, 및 (2) 수집된 데이터를 분석을 위한 수집점들로 분산시키는 단계인, 2개의 단계를 포함한다. 로컬 데이터 수집을 위해서는 원시 라우팅 프로토콜 및 패킷 침입자(paket sniffer)와 같은 여러가지 도구가 사용될 수 있다.

[0086] 일 구현예에서, 관리자 모듈로의 데이터의 분산이 네트워크 오버헤드를 일으키고 있지만, 네트워크 오버헤드는 적으며, 네트워크의 데이터 트래픽에 거의 영향을 주지 않는다. 또한, 네트워크 오버헤드는 압축, 델타 인코딩, 멀티캐스트, 시간 스케일의 적응 변화 및/또는 공간적 분산 범주 등을 사용함으로써 감소될 수 있다. 예를 들어, 네트워크의 정상 동작동안, 최소 데이터 집합이 수집되고 교환된다. 추가적인 데이터에 대한 요구가 증가하면(예를 들어, 수집되고 있는 정보가 불일치를 나타낼 경우), 관리자 모듈은 추가적인 정보를 요구하고, 증가된 모니터링을 필요로하는 노드들의 부분집합에 대한 데이터 수집의 빈도수를 증가시킬 수 있다.

[0087] **시뮬레이션 방법론**

[0088] 시뮬레이터에 의해 모델링되는 네트워크 특성은 트래픽 로드, 라우팅, 무선 신호, 장애 등의 여러가지 카테고리로 분류될 수 있다. 다음 섹션은 시뮬레이터가 대응 네트워크 특성을 시뮬레이트하게 하도록 사용되는 개별 모듈로서 이들 예시적인 카테고리들 각각의 시뮬레이션 예를 설명한다.

[0089] 트래픽 로드 시뮬레이터(410)

[0090] 시뮬레이터에 의해 생성된 네트워크 시뮬레이션은 실제 네트워크의 트래픽 패턴에 근사한 트래픽 패턴을 제공하도록 구성될 수 있다. 트래픽 로드 시뮬레이션 접근법의 일례는 엔드-투-엔드 애플리케이션 수요를 포함한다. 그러나, N-노드 네트워크는 잠재적으로 N^2개의 수요를 포함할 수 있다. 또한, 애플리케이션 수요의 의 이질성(heterogeneity), 및 전송 제어 프로토콜(TCP), 사용자 데이터그램 프로토콜(UDP), 빠른 전송 프로토콜(RTP) 등과 같은 상이한 전송 프로토콜의 사용이 주어지면, 엔드-투-엔드 애플리케이션 수요는 얻기 힘들 수 있다.

[0091] 일 구현예에서, 트래픽 로드 시뮬레이터(410) 모듈은 도 2의 트래픽 시뮬레이터 모듈(214)의 일부이고, 확장성을 위해, 그리고 엔드-투-엔드 애플리케이션 수요에 대한 필요를 피하기 위해 사용되는 링크-기반 트래픽 시뮬레이션을 제공한다. 구현 시, 링크-기반 트래픽 시뮬레이션은 각각의 링크의 애플리케이션-레벨 송신율를 조정하여, 실제 네트워크의 관찰된 링크-레벨 트래픽 카운트를 매치할 수 있다. 이러한 방식으로, 상위 계층들(예를 들면, 전송 계층, 애플리케이션 계층 등)이 추상화되고, 이것은 시뮬레이션이 패킷 사이즈 및 트래픽율에 집중하게 한다.

[0092] 애플리케이션-레벨 송신율만이 조정될 수 있고 매체 액세스 제어(MAC) 프로토콜이 처리되어야 하는 경우와 같이, 링크에 대한 송신율이 직접적으로 제어될 수 없을 때, 시뮬레이터에서 링크 단위로 송신율을 매치하는 것은 단순하지 않을 수 있다. 예를 들어, 링크의 애플리케이션 송신율이 1 Mbps로 설정될 때, (공중에서의) 실제 송신율은 MAC 계층의 백-오프(back-off)때문에 더 낮아질 수도 있고, MAC 레벨 재전송 때문에 더 높아질 수도 있다. 이 문제는 간섭에 의해 더 복잡해지고, 이것은 상이한 링크들에 대한 송신율들 간의 상호 의존도를 일으킨다.

[0093] 반복 검색 기법은 각각의 링크의 송신율을 결정함으로써 이 문제를 처리하는 데 이용될 수 있다. (ⅰ)배수 증가 및 배수 감소, (ii)가산 증가 및 가산 감소와 같은 여러 가지 반복 검색 기법이 사용될 수 있다. 예시적인 수도-코드(pseudo-code)를 사용하여 도시된 다음의 프로시저에서 나타나는 바와 같이, 각각의 링크는 개별적으로, 시뮬레이터 내의 현재 송신율과 실제 네트워크 내의 실제 송신율 간의 차이를 감소시키려고 시도한다.

[0094]
```
while (not converged and i < maxIterations)
```

[0095]
```
    i = i + 1
```

[0096]
```
    If (option = = multiplicative)
```

[0097]
```
        for each link (j)
```

[0098]
```
            prevRatio = targetMacSent(j)/simMacSent(J);
```

[0099] currRatio = (1 - α) + α* prevRatio;

[0100] simAppSent(J) = prevAppSent(j) * currRatio;

[0101] else // additive

[0102] for each link (j)

[0103] diff = targetMacSent(j) - prevMacSent(j);

[0104] simAppSent(j) = prevAppSent(j) + α* diff;

[0105] run simulation using simAppSent as input

[0106] determine simMacSent for all links from simulation results

[0107] conveyed = isConverge (simMacSent, targetMacSent)

[0108] 따라서, 상기 수도-코드는 배수-증가/배수-감소 또는 가산-증가/가산-감소를 이용한 애플리케이션-레벨 송신율에 대한 검색의 예를 나타낸다. 상기 예시적인 프로시저에서, α≤1(예를 들어, α=0.5)인 파라미터 α가 도입되어, 변동을 완충한다. 프로세스는 레이트가 목표 비율(targetMacSent로 표시)에 근사하거나 최대 수의 반복에 도달할 때까지 다시 반복한다.

[0109] 라우트 시뮬레이터(412)

[0110] 라우팅은 네트워크 성능, 특히, 멀티-홉 무선 네트워크에 있어서 중요한 역할을 한다. 하나의 라우트 시뮬레이션 접근법은 시뮬레이터 내부의 실제 네트워크에서 사용된 라우팅 프로토콜의 시뮬레이션을 포함시킨다. 실제 네트워크에서와 동일한 라우팅 행위을 재생산하기 위하여, 라우팅을 설정하도록 상세한 패킷 트레이스를 얻는다.

[0111] 패킷이 취하는 실제 라우트는 라우트 시뮬레이터(412) 모듈에 대한 입력으로서 사용될 수 있다. 라우트가 빈번하게 변동하지 않는 경우에는, 관리자가 패킷마다의 라우트를 수집하는 것 대신에, 라우팅 변경을 트레이스할 수 있다. 이를 위해, 라우트 시뮬레이터(412) 모듈은 트레이스-구동식일 수 있다. 예를 들어, 라우트 시뮬레이션 모듈은 QUALNET 시뮬레이터와 같은 시뮬레이터(118(N)) 내부에 구현될 수 있다 (QUALNET은 캘리포니아주 로스엔젤레스 소재의 Scalable Network Technologies, Inc.의 상표임). 라우트 시뮬레이션(412) 모듈은 라우팅 업데이트 및 대응 타임스탬프(time stamp)를 입력으로서 받고, 네트워크 시뮬레이션에서의 패킷이 실제 네트워크에서와 동일한 라우트를 따를 것을 보증한다.

[0112] 신호 강도 시뮬레이터(414)

[0113] 신호 강도는 무선 네트워크와 유선 네트워크 양자 모두의 성능에 영향을 미친다. 상이한 네트워크 접속 장치들(예를 들어, 무선 카드) 및 환경들 간의 편차로 인하여, 그러한 요인들을 전부 캡쳐하는 보편적인 전파 모델(propagation model)을 유도해내기 어려울 수 있다. 이러한 문제점을 해결하기 위하여, 신호 강도 시뮬레이터(414)는, 네트워크 접속 장치들 자체에서 취득된 것과 같은, 실제 네트워크에서의 신호 강도에 대한 실제 측정값으로 구동될 수 있다.

[0114] 장애 주입(Fault Injection)(416)

[0115] 프레임워크는 호스트에서의 패킷 탈락, 외부 잡음 소스, MAC 부당행위(misbehavior) 등과 같은 상이한 유형의 장애들을 시뮬레이터에 주입하도록 실행될 수 있는 장애 주입(416) 모듈을 포함할 수 있다. 이러한 방식에서, 분석 모듈은 해당 장애가 네트워크에 미치는 영향을 조사할 수 있다. 예를 들어, 호스트에서의 패킷 탈락은, 부당행위를 행한 노드가 하나 이상의 이웃 노드로부터 온 트래픽의 일부를 하드웨어/소프트웨어 에러, 버퍼 오버플로우, 부당한 탈락 등으로 인하여 누락시킬 때에 발생한다. 이러한 엔드호스트 패킷 탈락을 탐지하는 기능은, 관리자가 엔트호스트에 의해 유발된 손실과 네트워크에 의해 유발된 손실을 구별할 수 있게 해 준다는 점에서 유용하다.

[0116] 프레임워크는 장애 주입(416) 모듈의 실행을 통하여, 외부 잡음 소스를 네트워크에 주입하는 기능을 지원한다. 따라서, 프레임워크는, 네트워크 외부에 있으면서도 (즉, 노드에 의해 제공되는 것이 아니면서도) 네트워크에 영향을 미치는 잡음 소스의 영향을 복제하는 시뮬레이션을 제공할 수 있다.

[0117] MAC 부당행위는 장애를 갖는 노드가 MAC 에티켓을 따르지 않고 채널 대역폭에 대한 부당한 공유를 얻을 때에 발

생한다. 예를 들어, IEEE 802.11에서, 장애를 갖는 노드는 공격적으로 트래픽을 전송하기 위하여 보다 작은 컨텐션 윈도우(contention window, CW)를 선택할 수 있다.

[0118] 프레임워크는 시뮬레이트되는 네트워크에 높은 데이터 전송 부하를 가함으로써 링크 혼잡(link congestion)도 시뮬레이트할 수 있다. 다른 유형의 장애들과는 달리, 링크 혼잡은 각각의 노드로부터 수집된 트래픽 통계에 의하여 암시적으로 캡쳐된다. 그러므로, 트레이스-구동식인 시뮬레이션은, 링크 혼잡이 실제 네트워크에 미치는 영향을 직접적으로 평가할 수 있다. 장애 진단에 대해서는 다음 섹션에서 더 상세하게 설명될 것이다.

[0119] **장애 진단**

[0120] 고장 및 성능 문제에 대한 근본적인 원인은 도 2의 분석 모듈(122(N))의 실행을 통해 진단될 수 있다. 네트워크 시뮬레이션에 장애를 적용하면, 네트워크 불일치의 진단은, 시뮬레이션되는 네트워크에 주입될 때에, 시뮬레이션되는 네트워크에 의해 추정된 성능이 실제 네트워크의 관찰된 성능에 가까워지게 하는 장애들의 집합을 검색하도록 감소될 수 있다. 보다 형식적으로 표현하면, 소정의 네트워크 설정 NS에 대하여, FaultSet는 다음과 같이 찾아진다.

[0121] $SimPerf(NS; FaultSet) \approx RealPerf$

[0122] 여기에서, 네트워크 성능은 다양한 메트릭을 사용하여 정량화될 수 있는 함수값이다.

[0123] 장애에 대한 검색 공간은, 발견될 수 있는 상이한 장애들의 조합으로 인하여 수많은 검색 차원(searching dimension)을 포함할 수 있다. 일 구현예에서, 상이한 유형의 장애들이 일부 특정 네트워크 성능 메트릭들을 종종 변경시킨다는 것을 인식하여, 분석 모듈(122(N))은 보다 효율적인 검색을 위하여 최적화된다. 예를 들어, 호스트에서의 패킷 탈락은 일반적으로 링크 손실율에는 영향을 미치지만, 다른 네트워크 성능 메트릭들에는 영향을 미치지 않는다. 그러므로, 네트워크 성능 메트릭은 그 메트릭에 의해 나타나는 추정된 네트워크 성능과 관찰된 네트워크 성능 간의 차이를 인식함으로써 네트워크 성능을 진단하는 데에 사용될 수 있다.

[0124] 일 구현예에서는, 장애 진단을 위한 예측 모델을 제공할 필요가 없다. 그 대신에, 그 사실 이후에 네트워크에 무엇이 발생하는지를 시뮬레이션하는 것으로 충분하다. 예를 들어, 에이전트 모듈은 링크 상황 및 트래픽 패턴에 관한 정보를 관리자 모듈에 주기적으로 보고할 수 있다. 이러한 정보는 처리된 후에 시뮬레이터에 제공되어, 장애에 대한 가능성있는 근본 원인을 결정하는 데에 이용될 수 있는 네트워크 시뮬레이션을 생성하게 할 수 있다.

[0125] 초기 진단

[0126] 도 5는 서로 동일한 유형의 장애들이 처음 진단되는 구현예에서의 프로시저(500)를 나타낸 흐름도이다. 설명의 편의를 도모하기 위하여, 이하의 논의에서는 (1) 호스트에서의 패킷 탈락, (2) 외부 잡음, 및 (3) MAC 부당행위의 3가지 예시적인 유형의 장애를 다룬다. 그러나, 매우 다양한 기타의 장애들 및 그 조합들도 마찬가지로 다루어질 수 있음을 알아야 한다. 이하의 논의는 설명되는 시스템 및 장치를 이용하여 구현될 수 있는 프로시저를 포함한다. 각각의 프로시저의 양상은 하드웨어, 펌웨어, 소프트웨어 또는 그들의 조합으로 구현될 수 있다. 프로시저는 하나 이상의 장치에 의해 수행되는 동작들을 지정하는 블럭들의 집합으로서 나타나 있지만, 각각의 블럭에 의한 동작을 수행하기 위해 나타내어진 순서로만 제한되는 것은 아니다.

[0127] 전술한 바와 같이, 트레이스-구동식인 시뮬레이션은 실제 네트워크의 현재 네트워크 설정을 제공받으면, 해당 네트워크의 추정된 네트워크 성능을 구축하는 데에 사용될 수 있다. 장애의 유형은, 추정된 네트워크 성능과 관찰된 네트워크 성능 간의 차이에 기초하여, 도 6에 도시된 예와 같은 판정 트리를 사용하여 결정될 수 있다.

[0128] 다양한 요인으로 인해, 장애가 없는 경우에서조차도 추정된 네트워크 성능이 관찰된 네트워크 성능과 동일할 가능성은 낮다. 그러므로, 네트워크 성능에서의 불일치는 임계치를 이용하여 결정될 수 있다. 예를 들어, 추정된 네트워크 성능과 관찰된(즉, 실제의) 네트워크 성능 값간의 차이가 대응하는 임계치를 초과하는지의 여부에 따라 불일치가 결정될 수 있다. 임계치는 시뮬레이션된 네트워크 성능과 실제 네트워크 성능 간의 히스토릭한 차이를 관찰함으로써와 같이 다양한 방식으로 계산될 수 있다.

[0129] 도 6에 일례가 도시되어 있는 장애 분류 스킴은, 상이한 장애가 각각의 상이한 행위을 나타나게 한다는 것을 인식하여, 불일치를 유발한 장애의 유형을 결정하도록 구성된다. 각각의 장애에 의해 나타나는 행위가 중복될 수 있기는 하지만 (예를 들어, 잡음 소스와 호스트에서의 패킷 탈락 양자 모두가 탈락율을 증가시키고, 컨텐션 윈도우를 저하시키면 트래픽의 양이 증가하여 간섭 잡음도 증가됨), 우선 각각의 행위를 구별시키는 검사에 의하

여 장애를 카테고리화할 수 있다. 예를 들어, 외부 잡음 소스는 이웃 노드가 경험하는 잡음 레벨은 증가시키지만, 어떠한 노드의 전송률도 증가시키지 않는다. 그러므로, 외부 잡음 소스는 MAC 부당행위 및 호스트에서의 패킷 탈락으로부터 구별될 수 있다.

[0130] 이하에서는, 다시 도 5를 참고할 것이다. 이하의 논의는, 관련 도면에 대한 논의에 포함된 예시적인 수도 코드에서 사용되는 대안적인 표기를 설명하는 이탤릭체 텍스트의 삽입구를 포함한다. 블럭(502)에서, 분석 모듈은 도 2의 장애 디렉토리(218)에 있는 것과 같은 복수의 장애로부터 하나 이상의 장애를 선택한다. 프로시저(500)의 1차 반복에서, 정상적으로 동작하는 상황일 때, 즉 장애가 없을 때에는, 복수의 장애 중 어느 것도 예상되는 네트워크 성능을 유도하기 위하여 선택되지 않는다. 다른 구현예에서, 도 5의 프로시저(500)는 초기 진단을 수행하는 데에 이용되며, 반복되지 않는다. 즉, "원 패스(one pass)" 프로시저이다. 이러한 구현예에서, 블럭(502)은 프로시저(500)로부터 제거될 수 있고, 장애 집합은 공집합 ()으로서 제공된다.

[0131] 블럭(504)에서, 장애 집합(FS) 및 네트워크 설정(NS)은 네트워크 시뮬레이션에 입력으로서 제공된다. 신호 강도, 트래픽 통계, 라우팅 테이블 등과 같은 다양한 네트워크 설정이 제공될 수 있다.

[0132] 블럭(506)에서, 제공된 입력을 이용하여 네트워크 시뮬레이션을 실행함으로써, 예상 성능(SimPerf)이 예측된다. 결정 블럭(506)에서, 예상 성능(SimPerf)과 실제 성능(RealPerf) 간의 차이(Diff)가 임계치보다 큰지에 대한 판정이 이루어진다. 그 차이가 임계치보다 큰 경우(블럭(506)), 장애 유형(FT)이 결정된다(블럭(510)). 장애 유형의 결정에 대한 상세한 논의는 도 6과 관련하여 찾을 수 있을 것이다.

[0133] 장애 유형이 결정되고 나면, 관찰 네트워크 성능과 예상 네트워크 성능 간의 차이가 해당 특정 장애 유형에 대한 임계치를 초과하는 노드 및 링크의 집합을 검색함으로써(블럭(514)), 장애를 찾아낸다(블럭(512)). 장애 유형에 따라, 어떤 네트워크 성능 메트릭을 사용하여 성능 차이를 정량화할 것인지가 결정된다. 예를 들어, 예상 손실율과 관찰 손실율 간에 상당한 차이를 갖는 링크를 찾음으로써, 패킷 탈락이 식별될 수 있다.

[0134] 블럭(516)에서, 장애의 크기가 결정된다. 예를 들어, 함수("g()"로 표시됨)를 이용하여, 장애의 영향을 대응 크기에 맵핑할 수 있다. 예를 들어, 엔드호스트 패킷 탈락의 시나리오에서, 링크 손실율의 차이는 링크 상에서의 패킷 탈락율의 변화(장애의 크기)에 직접적으로 맵핑될 수 있으므로, g() 함수는 항등 함수이다. 외부 잡음 장애의 시나리오에서, g() 함수는 잡음 신호의 전파 함수이다. 블럭(510) 내지 블럭(516)은 각각의 링크 또는 노드에 대해 반복될 수 있다. 그리고, 대응 크기를 갖는 장애는 블럭(516)에서 장애 집합에 추가될 수 있다.

[0135] 이하는, 다음과 같이 나타나는 도 5의 프로시저(500)와 유사한 프로시저를 구현하기 위해 실행될 수 있는 예시적인 수도 코드를 나타낸 것이다.

[0136] Let NS denote the network settings (i.e., signal strength, traffic statistics, routing table)

[0137] Let RealPerf denote the real network performance

[0138] FaultSet = {}

[0139] Predict SimPerf by running simulation with input (NS; FaultSet)

[0140] if |Diff (SimPerf, RealPerf)| > threshold

[0141] determine the fault type ft using a decision tree for each link or node i

[0142] if ($|Diff_{ft}$ (SimPerf (i), RealPerf(i))| > threshold)

[0143] add fault(ft, i) with

[0144] magnitude(i) = g($Diff_{ft}$ (SimPerf (i), RealPerf (i))

[0145] 수도 코드는 장애가 발생했는지의 여부를 탐지하는 데에 이용될 수 있는 진단 알고리즘을 기술한다. 이하의 프로시저는 탐지된 장애의 유형을 결정하는 데에 이용될 수 있는 알고리즘의 일례이다.

[0146] 도 6은 장애의 유형을 결정하기 위하여 판정 트리가 이용되는 구현예에서의 프로시저(600)를 도시한 흐름도이다. 도 6에 도시된 프로시저(600)는 도 5의 블럭(510)에 대응할 수도 있고 대응하지 않을 수도 있다. 결정 블럭(602)에서, 시뮬레이트된 패킷 전송량(SimSent)에서 실제 패킷 전송량(RealSent)을 감산한 것의 절대값이 (ThreshSentDiff로 표시된)임계이보다 큰지의 여부가 판정된다. 그런 경우, 컨텐션 윈도우(CW)가 지나치게 낮게 설정되었음을 나타내는 장애가 전송된다(블럭(604)).

[0147] 블럭(602)의 임계치를 초과하지 않는 경우, 결정 블럭(606)에서는, 실제 네트워크 상에 나타나는 실제 잡음(RealNoise)과 시뮬레이션된 네트워크의 예상 잡음(SimNoise) 간에 불일치가 있는지의 여부(즉, 임계 잡음 차 ThreshNoiseDiff를 초과하는지의 여부)가 판정된다. 그런 경우, 잡음 장애가 결정된다(블럭(608)).

[0148] 잡음 임계치를 초과하지 않는 경우(블럭(606)), 결정 블럭(610)에서, 시뮬레이션된 패킷 손실(SimLoss)(즉, 예상 패킷 손실)과 실제 패킷 손실 간의 차가 임계치 손실 차이(ThreshLossDiff)보다 큰지의 여부가 판정된다. 그런 경우, 탈락 장애가 산출된다(블럭(612)). 그렇지 않은 경우, 노드는 정상적으로 동작하고 있는 것이다(블럭(614)). 매우 다양한 기타 장애 유형들 또한 유사한 방식으로 결정될 수 있음이 분명하다.

[0149] 도 7은 서로 상이한 유형을 갖는 장애들이 반복적 진단 알고리즘을 사용하여 진단되는 구현예에서의 프로시저(700)를 도시하는 흐름도이다. 일반적으로, 다양한 유형의 상호작용하는 장애 상호작용이 네트워크 내에서 마주칠 수 있다. 장애들이 동일한 유형인 경우에서도, 상호작용은 여전히 있을 수 있고, 그로 인해 원 패스 진단 알고리즘은 불충분하게 된다. 그러므로, 근본적인 원인을 찾기 위하여, 도 7에 도시된 것과 같은 반복적 진단 알고리즘이 구현될 수 있다. 알고리즘은 (ⅰ) 도 5의 프로시저(500)와 유사한 초기 진단 스테이지와 (ii) 반복적 상세화(refinement)의 2개의 스테이지를 포함한다.

[0150] 초기 진단 스테이지 동안, 초기 장애 집합을 유도하기 위하여 원 패스 진단 알고리즘이 적용된다. 제2 스테이지 동안, (ⅰ) 장애 집합에 이미 삽입되어 있는 장애들의 크기를 조정하고, (ii) 필요한 경우에 새로운 장애를 집합에 추가하는 것에 의하여, 장애 집합을 반복적으로 상세화된다. 프로시저(700)는, 장애 유형 및 위치가 변하지 않게 되거나, 장애의 크기가 최소량만큼만 변하게 되는 것 등과 같이, 장애 집합의 변화가 무시할 수 있을 정도로 될 때까지 반복된다.

[0151] 반복적 접근 방식은 장애의 크기를 찾는 데에도 사용될 수 있다. 하이레벨에서, 이러한 접근 방식은, 현재값과 목표값 간의 차이가 목표에 가깝게 이동하기 위한 피드백으로서 이용되었던 도 5의 링크 기반 시뮬레이션과 유사하다.

[0152] 블럭(702)에서, 예를 들어, 예상 네트워크 성능은 기존의 장애 집합 하에서 각각의 반복마다 추정된다. 예를 들어, 예상 네트워크 성능은 실제 네트워크로부터 얻어진 네트워크 설정을 사용한 네트워크의 시뮬레이션을 통해 추정될 수 있다. 네트워크 설정은 각 노드 상에서의 에이전트 모듈의 실행을 통하여 제공된다. 각각의 노드에 의하여 제공되는 네트워크 설정은 해당 노드의 로컬 네트워크 성능은 물론, 이웃 노드들의 네트워크 성능도 기술할 수 있다.

[0153] 블럭(704)에서, (기존 장애 집합에 기초한) 추정 네트워크 성능과 실제 성능 간의 차이가 계산된다. 예를 들어, 차이는 관리자 모듈의 실행을 통해 관리자 노드에 의해 계산될 수 있다. 관리자 모듈은, 실행 시에, 시뮬레이션된 네트워크로부터 얻어진 추정(즉, 예상) 네트워크 성능과, 복수의 에이전트로부터 얻어진 추가의 네트워크 설정들에 의해 나타나는 실제(즉, 관찰) 네트워크 성능을 비교한다.

[0154] 도 7의 프로시저(700)는 우선 도 5와 관련하여 설명된 프로시저(500)와 유사한 방식으로 초기 장애 진단을 수행한다. 결정 블럭(706)에서, 예를 들어, 계산된 차이가 대응 임계치보다 큰지의 여부가 판정된다. 그렇지 않은 경우, 장애 집합이 보고된다(블럭(708)). 이러한 경우, 계산된 차이가 임계치보다 크지 않기 때문에, 분석 모듈은 네트워크가 정상적으로 동작하고 있음을 알게 된다. 그러나, 계산된 차이가 대응 임계치보다 큰 경우(블럭(706)), 장애 유형이 결정된다(블럭(710)). 장애 유형은 다양한 방식으로 결정될 수 있고, 도 6에 그 일례가 나타나 있다.

[0155] 블럭(712)에서, 차이가 장애 크기의 변화로 해석되고, 장애 크기는 계산된 변화에 따라 조정된다(블럭(714)). 예를 들어, 도 5와 관련하여 전술한 함수 g()는 예상 네트워크 성능과 실제 네트워크 성능 간의 각각의 차이에 기초하여 각각의 장애마다의 장애 크기를 계산하는 데에 이용될 수 있다. 이러한 방식에서는, 장애들을 서로 비교하여, 어떤 장애가 관찰된 불일치에 대응하는 영향을 네트워크 성능에 미쳤는지를 결정한다. 일 구현예에서, 우선 최대 장애 크기가 불일치를 설명하고 그 불일치를 유발한 특정 장애를 식별하는 데에 사용된다. 다른 구현예에서, 계산된 차이에 근사한 차이를 유발한 장애를 찾아내기 위하여, 장애 크기들이 비교된다. 예를 들어, 복수의 장애 각각은, 예상 네트워크 성능과 실제 네트워크 성능 간에 각각의 차이를 가질 수 있다. 각각의 차이를 계산된 네트워크 성능 차이에 매치시킴으로써, 하나 이상의 장애가 선택될 수 있다. 블럭(716)에서, 대응 임계치 미만의 크기를 갖는 장애를 제거하여, 장애 집합을 최적화한다.

[0156] 결정 블럭(718)에서, 현재의 장애 집합을 이용한 네트워크의 예상 성능이 실제 네트워크 성능에 수렴하는지의 여부가 판정된다. 예를 들어, 분석 모듈은 장애 집합의 하나 이상의 이전의 반복 및 네트워크 시뮬레이션에서

의 결과적인 성능값을 기술하는 휴리스틱 데이터를 저장할 수 있다. 목표값(즉, 실제 네트워크 성능값)과 현재값(즉, 시뮬레이션된 네트워크 성능값) 간의 차이는, 분석 모듈이 네트워크 시뮬레이션을 실제 네트워크에 근사하도록 점진적으로 "이동"시키기 위한 피드백으로서 사용된다.

[0157] 예상 성능이 실제 네트워크 성능에 수렴하지 않는 경우(블럭(718)), 새로운 장애 후보가 장애 집합에 추가된다. 정확한 장애 크기를 찾는 것 외에도, 예를 들어 예상 네트워크 성능과 실제 네트워크 성능 간의 차이를 가장 잘 설명할 수 있는 새로운 장애 후보들을 선택함으로써, 장애 집합의 맴버쉽을 반복적으로 상세화할 수 있다(블럭(720)). 이들 새로운 장애들은 장애 집합에 추가된다(블럭(722)). 그러고 나서, 새로운 장애 후보를 포함하는 장애 집합을 네트워크 시뮬레이션에 대한 입력으로서 사용하여, 기존 장애 집합 하에서 예상 네트워크 성능을 추정한다(블럭(702)). 일 구현예에서, 프로시저(700)가 반복될 때마다 최대의 불일치를 설명할 수 있는 장애가 추가되어, 오탐율(false positive)을 제어한다. 그리고, 프로시저(700)는 시뮬레이션되는 네트워크의 예상 성능이 실제 네트워크의 실제 성능에 근사할 때까지 반복된다. 이러한 방식에서, 시뮬레이션된 네트워크는 장애들의 포함을 통하여, 실제 네트워크에 관찰된 네트워크 성능을 유발하는 장애들에 대한 정확한 설명을 제공하도록 진전될 수 있다.

[0158] 이하는, 도 7의 프로시저(700)를 제공하도록 실행될 수 있는 예시적인 수도 코드를 나타낸 것이다.

[0159]

```
1) Let NS denote the network settings
      (i.e., signal strength, traffic statistics, and routing tables)
   Let RealPerf denote the real network performance
2) FaultSet = {}
3) Predict SimPerf by running simulation with input (NS; FaultSet)
4) if |Diff (SimPerf, RealPerf)| > threshold
      go to (5)
   else
      go to (7)
5) Initial diagnosis: initialize FaultSet by applying the algorithm of FIG. 5
6) while (not converged)
      a) adjusting fault magnitude
         for each fault type ft in FaultSet (in the order of decision tree in FIG. 6)
           for each fault i in (FaultSet, ft)
              magnitude(i) -= g(Diff_ft (SimPerf(i), RealPerf (i)))
              if (|magnitude(i)| < threshold)
                 delete the fault (ft, i)
      b) adding new candidate faults if necessary
         forcach fault type ft (in the order of decision tree of FIG. 6)
           i) find a fault i s.t. it is not in FaultSet and has the largest |Diff_ft
              (SimPerf (i):RealPerf (i))|
           ii) if |Diff_ft (SimPerf(i), RealPerf(i))| > threshold)
              add (ft, i) to FaultSet with magnitude(i) = g(Diff_ft (SimPerf(i),
              RealPerf (i))
      c) simulate
7) Report FaultSet
```

[0160]

[0161] 이와 같이, 상기의 수도 코드는 다양한 유형의 장애들을 진단하도록 구성된 예시적인 진단 알고리즘을 설명한다.

[0162] **트레이스 데이터에서의 에러 제거**

[0163] 이전의 섹션에서는, 온라인 시뮬레이션을 구동하기 위하여 트레이스 데이터가 사용되는 장애 진단이 설명되었다. 실제로, 에이전트 모듈에 의해 수집된 원시 트레이스 데이터는, 각각의 노드 상에서 실행될 때, 상술한 것과 같이 하드웨어, 소프트웨어 및/또는 네트워크 에러로 인한 다양한 이유로 인해 에러를 포함할 수 있다. 그러므로, 복수의 에이전트로부터 수신된 "원시" 트레이스 데이터를 클리닝하여, 장애 진단을 위한 시뮬레이터(118(N))로의 입력으로서 클리닝된 트레이스 데이터를 제공하기 위하여, 클리닝 모듈(120(N))이 실행될 수 있다.

[0164] 도 8은, 도 1의 복수의 노드(102(1)-102(N))가 이웃 모니터링(neighbor monitoring)을 수행하도록 실행될 수 있는 에이전트 모듈을 포함하는 구현예에서의 네트워크(800)를 나타낸 도면이다. 네트워크 내의 각각의 노드에

서 실행되는 에이전트 모듈은, 복수의 노드(102(1)-102(N)) 각각이 자기 자신의 인입/인출 링크는 물론, 자신의 통신 범위 내에 있는 다른 링크에 대한 성능 및 트래픽 통계를 보고하는 기술인 이웃 모니터링을 수행한다. 이웃 모니터링은 다양한 방식으로 수행될 수 있다. 예를 들어, 제1 노드 상의 에이전트 모듈이 네트워크 내의 제2 노드를 검사하도록 실행되어, 제2 노드로부터의 네트워크 성능 데이터를 얻을 수 있다. 다른 예에서, 제1 노드는 네트워크 성능 데이터를 포함하는 브로드캐스트와 같은 제2 노드로부터의 통신을 수신한다. 또 다른 예에서, 제1 노드는 네트워크를 통한 통신을 위하여 제2 노드에 의해 송신된 데이터를 모니터링하여, 네트워크 성능을 모니터링한다. 예를 들어, 제1 노드는, 노드의 네트워크 접속 장치가 해당 특정 노드에 도달하는 각각의 데이터 패킷 전체를 인터셉트하고 판독할 수 있게 하는 "무차별(promiscuous)" 모드로 동작할 수 있다.

[0165] 이웃 모니터링 동안, 상이한 소스들(즉, 노드들)로부터의 복수의 보고가 각각의 링크에 제출될 수 있다. 예를 들어, 노드(102(3))는, 노드(102(1), 102(n))의 네트워크 성능뿐만 아니라, 노드(102(2))의 네트워크 성능을 기술하는 보고(802(2))를 노드(102(2))로부터 얻을 수 있다. 도 8의 보고의 참조 번호들에서 사용되는 삽입구는, 예를 들면 노드(102(2) 및 보고(802(2))와 같은 자신의 개별적인 너드와 보고와의 대응 관계를 나타내도록 선택되어 있다.

[0166] 노드(102(3))는 관리자 노드(102(N))로의 전달을 위해 형성된 보고(802(3)) 내에, 보고(802(2))(도 8에 음영으로 표시됨)로부터의 네트워크 성능 데이터를 포함한다. 보고(802(3))는, 노드(102(3))가 노드들(102(2), 102(1))을 모니터링하여 얻은 네트워크 성능 데이터를 또한 포함할 수 있다. 일 구현예에서, 에이전트 모듈의 실행을 통해 중복되는 정보를 제거함으로써 보고(802(3))가 최적화된다. 예를 들어, 노드(102(3))의 에이전트 모듈은, 각각의 보고(802(2), 802(3))에서 노드(102(2), 102(3))에 의해 일관되고 반복되는 정보를 제거하고, 데이터의 임의의 불일치를 기술하는 데이터는 남겨 놓는다. 마찬가지로, 노드(102(n))는 수집 모듈(114(n))을 실행시켜, 노드(102(2), 102(3))로부터 네트워크 성능 데이터를 얻을 수 있다. 네트워크 성능 데이터는 관리자 노드(102(N))로 전달하기 위한 보고(802(n))로서 구성된다.

[0167] 중복 보고들은 관리자 노드(102(N))에 의해 네트워크 성능에서 하나 이상의 불일치를 탐지하기 위해 사용될 수 있다. 예를 들어, 보고(802(2), 802(3))는 관리자 노드(102(N))에 의해 클리닝(cleaning) 모듈(120(N))의 실행을 통해 서로 비교되어 거기에 기재된 네트워크 성능 데이터의 불일치들을 발견할 수 있다. 불일치들은 일예가 다음 도면에서 기재된 다양한 방식들로 발견될 수 있다.

[0168] 도 9는 이웃 노드들을 기재하는 보고들이 네트워크에서 부당행위를 하는 노드들의 위치를 알아내기 위해 비교되는 구현의 예에서 프로시저(900)를 도시하는 흐름도이다. 이 구현에서, 프로시저(900)는 보고들의 불일치를 설명할 수 있는 노드들의 최소 집합으로서 부당행위의 노드들을 식별한다.

[0169] 도 9와 관련되어 설명된 프로시저(900)에서, 송신 노드 i는 방향성 링크 l에 대해 송신된 패킷들의 수 및 수신된 복수의 MAC-레벨 에크너리지먼트를 ($sent_i(l)$, $ack_i(l)$)로서 보고한다. 수신 노드 j는 링크에 수신되는 패킷들의 수를 $recv_j(l)$으로서 보고한다. 추가로, 송신 또는 수신 노드의 바로 이웃 k는 또한 링크에 송신되고 수신되는 패킷들의 수 및 MAC-레벨 에크너리지먼트들을 ($sent_k(l)$, $recv_k(l)$, $ack_k(l)$)으로서 보고한다. 보고들의 불일치는 다음 경우들 중의 하나로 정의된다.

[0170] 판정 블럭(902)에서, 링크의 목적지에 의해 보고된 바와 같이, 링크에 수신된 패킷들의 수가, 링크의 소스에 의해 보고된 바와 같이, 동일 링크에 전송된 패킷들의 수보다 매우 큰 지(임계치로 설명된 바와 같이)에 대해 판정이 이루어진다. 즉, 노드 i로부터 노드 j까지의 링크 l에 대해, 소정의 임계치 t에서, 다음 판정이 이루어진다:

[0171] $$recv_j(l)-sent_i(l)>t$$

[0172] 각각이 노드들에 의한 보고들의 통신이 통상적으로 동기화되지 않으므로, 임계치 t가 사용된다. 수신된 패킷들의 수가 전송된 패킷들의 수보다 매우 크면, 보고들에 불일치가 기재되는데, 이것은 블럭(912)에 관련하여 더 상세히 기술될 것이다. 각각의 노드들에 의한 수신되고 송신된 패킷 수들이 대응하면, 프로시저(900)는 블럭(904)으로 진행한다.

[0173] 결정 블럭(904)에서, 링크의 소스에 의해 보고된 바와 같이, 링크에 전송된 MAC-레벨 에크너리지먼트의 수가, 링크의 목적지에 의해 보고된 바와 같이, 그 링크에 수신된 패킷들의 수에 대응하는지에 대해 판정이 이루어진다. 다시 말하면, 노드 i로부터 노드 j로의 링크 l에 대해, 소정의 임계치 t에서, 다음이 판정된다:

[0174] $$|ack_i(l)-recv_j(l)|>t$$

[0175] 그러므로, 에크너리지먼트의 수가 수신된 패킷들의 수에 대응하지 (즉, 근사하지) 않으면(블럭(904)), 보고들에 불일치가 기재된다. 에크너리지먼트의 수와 수신된 패킷들의 수가 대응하면(블럭(904)), 프로시저(900)는 블럭(906)으로 진행한다.

[0176] 결정 블럭(906)에서, 링크의 목적지의 이웃에 의해 보고된 바와 같이, 링크에 수신된 패킷들의 수가, 링크의 소스에 의해 보고된 바와 같이, 동일 링크에 송신된 패킷들의 수보다 매우 큰 지에 대한 판정이 이루어진다. 즉, 노드 j의 이웃이 노드 k인 노드 i로부터 노드 j로의 링크 l에 대해, 소정의 임계치 t에서, 다음이 판정된다:

[0177] $$recv_k(l)-sent_i(l)>t$$

[0178] 그러므로, 수신된 패킷들의 수가 전송된 패킷들의 수와 대응하면(즉, 근사하면)(블럭(906)), 보고들에 불일치가 기재된다. 그렇지 않으면, 그 다음, 프로시저(900)는 블럭(908)으로 진행한다.

[0179] 판정 블럭(908)에서, 링크의 소스의 이웃에 의해 보고된 바와 같이, 링크에 전송된 패킷들의 수가, 그것의 소스에 의해 보고된 바와 같이, 동일 링크에 전송된 패킷들의 수보다 매우 큰 지에 대한 판정이 이루어진다. 다시 말하면, 노드 i로부터 j까지의 링크 l, i의 이웃 k 및 소정의 임계치 t에 대해, 다음이 판정된다:

[0180] $$sent_k(l)-sent_i(l)>t$$

[0181] 상술된 수학식에서와 같이, 송신된 패킷들의 수가 소스와 이웃 노드들에 의해 각각 표시된 바와 같이 전송된 패킷들의 수에 근사하면(블럭(908)), 보고들의 불일치가 기재된다. 그렇지 않으면, 보고들은 일관된다(블럭(910)).

[0182] 결정 블럭(912)에서, 노드들의 불일치 쌍이 이미 불일치 그래프에 포함되었는지에 대한 판정이 이루어진다. 그렇지 않았으면, 노드들은 불일치 그래프에 추가된다(블럭(914)). 노드들의 불일치 쌍이 이미 불일치 그래프에 존재하거나(블럭(912)) 불일치 그래프에 추가되었으면(블럭(914)), 불일치 그래프의 노드들 간의 에지가 추가된다(블럭(916)).

[0183] 각각의 불일치 쌍이 식별된 후에, 블럭(918)에서, 관찰된 불일치를 설명할 수 있는 노드들의 최소 집합(즉, 최소 수)이 불일치 그래프에서 발견된다. 예를 들어, 네트워크의 대부분의 노드들이 신뢰성이 있는 보고들을 송신한다는 가정이 이루어질 수 있다. 그러므로, 관찰된 불일치들을 설명할 수 있는 노드들의 최소 집합이 발견된다. 예를 들어, 이 동작은 식별된 정점들이 부당행위를 하는 노드들을 나타내는, 불일치 그래프를 수용하는 정점들의 최소 집합을 발견함으로써 수행될 수 있다.

[0184] 정점들의 최소 집합은 NP-하드(NP-hard)라고 알려진 최소 정점 수용 문제의 사용을 통해 발견될 수 있다. 에지가 남지않을 때까지, 현재 불일치 그래프로부터의 최대 에지 및 인접한 에지를 가지는 노드를 반복적으로 선택하고 제거하는 그리디 알고리즘(greedy algorithm)이 적용된다.

[0185] 보고들이 히스토리가 불일치 탐지의 정확성을 더 향상시키기 위해 사용될 수 있다. 예를 들어, 블럭(920)에서, 새로운 보고가 추가되어 이전 정보를 삭제하지 않고 불일치 그래프를 업데이트할 수 있다. 그 다음, 새로운 보고의 노드들의 불일치 쌍들은 프로시저(900)의 블럭(912) 내지 블럭(918)을 사용하여 프로세스될 수 있다. 예를 들어, 블럭(918)과 같은 그리디 알고리즘은 부당행위의 노드들을 식별하기 위해 재적용될 수 있다.

[0186] 상황 분석

[0187] 이전 섹션들에서, 만약 존재하면, 예상 네트워크 성능과 관찰되는 네트워크 성능 간의 차이를 일으킬 수 있는 장애들을 식별하기 위해 네트워크 시뮬레이션으로 장애들이 선택적으로 주입된다. 네트워크 시뮬레이션은 또한 "상" 분석을 수행하기 위해 사용되어 네트워크 동작을 향상시킬 수 있을 것이다. 상황 분석은, 실행될 때, 관리자 모듈이 가능성 있는 상이한 네트워크와 노드 구성의 네트워크 성능에 미치는 효과를 결정하도록 한다. 상황 분석의 결과는, 네트워크에서 선택된 노드들 상의 에이전트 모듈이 대응하는 각각의 노드를 구성하도록 하는 등의, 관리자 모듈이 네트워크를 효과적으로 관리자 모듈이 동작시키도록 하는 동작들의 집합이다.

[0188] 예를 들어, 상황 분석은 이전에 설명된 바와 같이 온라인 트레이스-구동식 시뮬레이션의 사용을 통해 수행될 수 있다. 시뮬레이터를(즉, 도 2의 시뮬레이터(118(N)) 구동시키기 위해 수집될 수 있는 트레이스의 예는 다음의 논의에서 식별된다. 예를 들어, 시뮬레이션은 실제 네트워크의 네트워크 시뮬레이션을 제공하기 위해 사용될 수 있다. 네트워크 시뮬레이션은 상이한 노드 및 네트워크 구성을 테스트하고, 어떤 구성이 기존의 트래픽 조

건에 대해 최상의 전체적 네트워크 성능을 산출하는지를 결정하기 위해 재구성될 수 있다. 그 다음, 관리자 모듈은 그 구성에 기초하여 네트워크의 특정 노드에 의한 구현을 위한 동작들의 집합을 결정할 수 있다.

[0189] 상황 분석을 위해 채택된 종래 기법들은 단순화된 네트워크 모델들을 사용하고, 분석적으로 예상되는 성능을 구하였다. 그러나, 온라인 트레이스-구동식 시뮬레이션은 시뮬레이터의 사용이 휴리스틱이나 이론적 기술 그자체에 의해 가능한 것보다 네트워크의 동작에 향상된 통찰을 제공한다는 점에서 이론적 분석에 보다 유리한 점들을 갖는다. 예를 들어, 동작적 무선 네트워크는 트래픽 흐름, 네트워크 프로토콜, 신호 처리 알고리즘, 하드웨어, RF 전파, 및 가장 중요하게는 각각의 이들 피스(piece)들 간의 상호작용을 포함하는 다수의 복잡한 이들 피스들을 구비하는 복잡한 시스템이다. 더욱이, 네트워크 행위는 서로의 범위 내의 노드들 간의 상호작용 및 주위의 잡음 소스들에 의해 좌우될 수 있다. 휴리스틱과 이론적 기술 모두 그런 네트워크들의 행위 및 상이한 컴포넌트들 간의 상호작용을 캡처링하지 않는다.

[0190] 도 10은 상황 분석이 온라인 트레이스-구동식 시뮬레이션에 기초하여 수행되는 구현의 예의 프로시저(1000)를 도시하는 흐름도이다. 상위 레벨에서, 프로시저(1000)는 우선 네트워크 시뮬레이션을 사용하여 실제 네트워크를 재생성한다. 그 다음, 실제 네트워크에 적용될 때, 네트워크의 수정의 결과들은 네트워크 시뮬레이션의 그런 변경들을 적용하여 결정되어 네트워크 성능의 함축성을 정량화함으로써 결정된다.

[0191] 블럭(1002)에서, 복수의 수정 중의 하나 이상이 관리자 모듈의 실행을 통해 선택된다. 수정은 다양한 방식으로 선택될 수 있다. 예를 들어, 수정은 네트워크 성능을 감소시키는 대신에 증가시키는 장애로서 관리자 모듈에 의해 간주될 수 있다. 그런 예에서의 수정은 도 2의 장애 디렉토리(218)에 저장될 수 있고, 유형에 기초하여 배치될 수 있다. 블럭(1004)에서, 분석 모듈은 실제 네트워크의 네트워크 설정 및 네트워크 시뮬레이션의 선택된 수정을 포함하는 수정 집합을 입력으로서 제공한다.

[0192] 블럭(1006)에서, 네트워크의 예상되는 성능은 그 입력에 기초하여 예측된다. 예를 들어, 시뮬레이터는 실제 네트워크의 네트워크 설정 및 수정 집합에 기초하여 네트워크 시뮬레이션을 생성할 수 있다. 그 다음, 이전에 설명된 바와 같이, 네트워크 시뮬레이션은 실제 네트워크의 수정의 결과들을 결정하기 위해 사용될 수 있다.

[0193] 판정 블럭(1008)에서, 네트워크 시뮬레이션의 예상되는 성능과 실제 네트워크의 실제 성능 간의 차이가 임계치보다 큰 지에 대한 판정이 이루어진다. 예를 들어, 네트워크 시뮬레이션은 도 1의 에이전트 모델들(122(n))로부터 얻어진 실제 네트워크 성능 값들과 비교되는 시뮬레이트된 네트워크 성능 값들의 출력을 제공할 수 있다. 예상되는 성능과 실제의 네트워크 성능 간의 차이가 임계치보다 작으면(블럭(1008)), 새로운 수정이 선택될 수 있고(블럭(1002)), 그 수정의 영향은 전술된 바와 같이 결정된다.

[0194] 실행될 때, 분석 모듈은 네트워크의 에이전트 모듈들에 의해 수행될 하나 이상의 동작들을 구함으로써 수정을 구현한다(블럭(1008)). 예를 들어, 분석 모듈은 대응하는 수정에 매핑되는 동작들의 디렉토리를 포함할 수 있다. 그 다음, 분석 모듈은 그 수정들에 기초하여 대응하는 동작들을 얻을 수 있다.

[0195] 블럭(1010)에서, 분석 모듈은 대응하는 에이전트 모듈들로의 통신을 위한 하나 이상의 동작들을 설명하는 통신을 형성한다. 그 다음, 대응하는 에이전트 모듈들은 네트워크의 각각의 노드들이 거기에 기재된 동작들을 구현하도록 할 수 있다. 그러므로, 관리자 및 에이전트 모듈들은 장애 탐지와 유사한 방식으로 온라인 트레이스-구동식의 시뮬레이션에 기초하여 상황 분석을 수행하기 위해 사용될 수 있다. 상황 분석은 장애 정정 및 네트워크 성능 향상을 위해 사용될 수 있다.

[0196] 다른 실시예에서, 시뮬레이션은, 상황 분석을 수행하는 반복적 접근법을 사용해서와 같이, 네트워크 성능을 향상시키기 위해 네트워크에 이루어질 수정을 결정하기 위해 사용된다. 이 접근법은 도 5와 도 7과 관련되어 기재되는 바와 같은 시뮬레이션과 유사하다. 그러므로, 반복 정제는 복수 수정 동작들이 필요될 때 사용될 수 있다.

[0197] 도 11는 네트워크의 수정이 손상된 흐름(damaging flow)의 진단에 기초하여 구하여지는 구현의 예의 프로시저(1100)를 도시하는 흐름도이다. 블럭(1102)에서, 관리자 모듈(예를 들어, 도 1 및 도 2의 관리자 모듈(116(N)))은 네트워크의 하나 이상의 흐름들이 그들의 대응하는 예상되는 목표 처리율 값들보다 더 낮은 체험하는 처리율 값들을 결정하기 위해 실행된다. 블럭(1104)에서, 관리자 모듈은 네트워크의 흐름들 중의 임의의 것이 "손상된 흐름"인지를 판정한다. 손상된 흐름은 그 존재가 네트워크 처리율에의 심각한 저하를 일으키는 장애 유형이며, 그 손상된 흐름이 그 자체로서는 괜찮을 수 있으나 다른 경쟁하는 흐름들과는 상호작용을 잘 하지 않다는 점에서 이전 장애들과는 상이하다.

[0198] 블럭(1106)에서, 예를 들어, 목표 엔드-투-엔드 수요 및 사용되는 라우팅 프로토콜을 기재하는 네트워크 설정이

수집된다. 프로시저(1100)는 어떻게 네트워크(즉, 링크 로드들 및 라우팅)가 네트워크 구성에의 변경들에 반응하는지를 조사하므로, 이들 네트워크 설정은 중재(troubleshooting)를 위해 사용되는 트레이스들로부터 상이할 수 있음을 유의한다.

[0199] 블럭(1108)에서, 통합 네트워크 처리율에 미치는 영향은 네트워크 시뮬레이션으로부터의 각각의 흐름의 삭제에, 한 번에 한 개씩, 기초하여 조사된다. 일 실시예에서, 손상된 흐름은 성능에 가장 중요한 전체적인 향상을 산출하는 삭제를 가지는 흐름으로서 식별된다. 예를 들어, 네트워크(1200)는 복수의 흐름(1202) 내지 흐름(1216)을 포함하는 도 12에 도시된다. 흐름 8(1216)(도 12에서 F_8로서 도시됨)은 예시된 네트워크(1200)의 다른 흐름(1202) 내지 흐름(1214)을 각각에 교차된다. 그러므로, 흐름 8(1208)을 삭제하면, 다른 흐름(1202) 내지 흐름(1214) 중의 임의의 것의 삭제하는 것과는 달리, 처리율면에서 가장 큰 증가를 일으킬 것이다. 다시 말하면, 흐름 8(1216)의 존재는 네트워크(1200)의 성능에 가장 큰 손상을 일으킨다. 이 방식으로, 네트워크 성능에 가장 큰 증가를 일으키는 네트워크(1200)의 수정(즉, 시스템의 다른 흐름들에 미치는 흐름 8(1216)의 영향의 제거 또는 감축)이 결정될 수 있다.

[0200] 블럭(1110)에서, 하나 이상의 동작들이 네트워크 성능을 향상시키도록 사용될 수 있는 수정에 기초하여 구하여진다. 동작들의 예들로는 흐름 8(1216)의 비율-제한, 리라우팅(rerouting), 및 형상 제어를 포함할 수 있다. 네트워크 시뮬레이션은 이들 동작들의 이점들을 관리자 모듈이 정확히 더 추정하도록 한다. 예를 들어, 다음 표는 교정 동작들의 예에 대한 예상되는 처리율을 나타낸다.

[0201]

동작	총 처리율(Mbps)
동작 없슴	1.064
흐름 8의 비율을 반으로 감소함	1.148
흐름 8을 그리드 경계를 통해 라우트함	1,217
전송 전력을 20dBM으로 증가시킴	0.990
전송 전력을 25dBm으로 증가시킴	1.661

[0202] 표에 나타난 바와 같이, 전송 전력의 25 dBm으로의 증가는, 이 동작이 목적지에 도착하기 위해 필요한 홉(hop)들의 수를 감소시키므로, 고려 사항 하의 4개의 동작들의 예들(및 한 개의 비동작) 중에 가장 높은 처리율을 산출한다. 이들 결과들에 기초하여, 관리자 모듈은 각각의 노드들에 하나 이상의 에이전트들이 전력을 증가시켜 네트워크 성능 문제를 경감시키는 통신을 형성한다.

[0203] 예시적인 프레임워크 구현

[0204] 기술된 프레임워크의 일 예는 윈도우즈 XP 플랫폼(윈도우즈 XP는 워싱턴주 레드몬드의 마이크로소프트사의 상표임) 상에 구현되었다. 예시적인 구현의 컴포넌트들, 설계 원칙들, 및 상기 것들의 특징들이 이 섹션에 기재된다.

[0205] 이 예에서의 예시적인 프레임워크는 2개의 분리된 컴포넌트들 즉, 에이전트 모듈 및 관리자 모듈을 포함한다. 도 1과 관련하여 상술한 바와 같이, 에이전트 모듈은 네트워크의 각각의 노드에서 실행되어 정기적으로 또는 요구할 때마다 로컬 데이터를 보고한다. 관리자 모듈은 에이전트 모듈로부터 관련 데이터를 수집하고, 도 2와 관련하여 기술된 바와 같이 포함된 분석 모듈의 실행을 통해서와 같이 데이터를 분석하기 위해 실행된다.

[0206] 예시적인 프레임워크는 간결성 및 확장성 설계 원칙들을 채택한다. 예를 들어, 모니터링과 관리를 위해 수집되고 전파되는 데이터는 (워싱턴주 레드몬드의 마이크로소프트사의 상표인) 윈도우즈에 지원되는 성능 카운터들로 캐스트(cast)될 수 있다. 성능 카운터들은 카테고리들에 의해 그룹화된 (이름, 값)쌍들로서 제공될 수 있다.

[0207] 기술된 프레임워크는 또한 확장가능하다. 모니터되는 데이터에의 추가는 성능 카운터들의 새로운 카테고리의 생성 및 정보가 변경시 성능 카운터 값들을 업데이트하는 모듈을 기록하는 것을 포함한다. TCP(Transmission Control Protocol), UDP(User Datagram Protocol), IP(Internet Protocol), 및 워크스테이션 원격 응용 프로그램 인터페이스(Workstation Remote Application Programming Interface;WRAPI)는 거의 추가 작업 없이 프레임워크에 포함될 수 있다.

[0208] 이들 성능 카운터의 값들은 판독 전용이거나 또는 기입가능할 수 있다. 기입가능 카운터는, 예를 들어, 위임받은 관리자 노드가 값들을 변경시켜 노드의 행동에 영향을 줌으로써 서로 다른 각 노드에서 실행되는 에이전트 모듈과 관리자 모듈의 통신을 통해 문제들을 해결하거나 원격으로 실험을 개시하는 방법을 제공한다.

[0209] 각 관리자 노드는 또한 네트워크 관리자와 대화하는 그래픽 사용자 인터페이스(GUI)(1400)을 구비하고 있으며, 이에 대한 일례가 도 14에 도시되어 있다. GUI에 의해 관리자는 네트워크의 가시화 뿐 아니라 관리자 모듈을 통해 관리 요청을 발행할 수 있다. GUI(1400)는 예시적 네트워크 테스트-베드(test-bed)에 대한 형상을 디스플레이한다. 이 예에서의 GUI(1400)는 에이전트가 23개 노드의 테스트-베드를 거쳐 배치된 관리자 윈도우를 도시한다. 관리자 모듈은 직접 얻어지거나 추론된 노드의 상대 좌표에 기반하여 네트워크 형상을 디스플레이할 수 있다. 또한, GUI(1400)에 의해, 관리자는 보다 상세한 정보를 위해 네트워크의 특정 부분을 줌-인(zoom-in)하여 특정 링크에 대한 네트워크 성능 데이터를 태블릿 포맷으로 디스플레이하는 링크를 클릭할 수 있다.

[0210] **결론**

[0211] 비록 본 발명을 구조적인 특징 및/또는 방법론 액트에 특정한 언어로 기술하였지만, 첨부된 청구범위에 기재된 본 발명의 사상 및 범주는 기술된 특정 특징 또는 액트에만 한정되는 것은 아니다. 오히려, 특정 특징들 및 액들은 본 발명의 일부 실시예로서 기재된 것에 불과한 것이다.

발명의 효과

[0212] 네트워크 중재 프레임워크에 대해 기술하고자 한다. 이 프레임워크는 실제 네트워크의 시뮬레이션을 채용하여 그 실제 네트워크의 동작에서의 장애를 검출하고 진단할 수 있다. 예컨대, 네트워크 시뮬레이션은 실제 네트워크의 동작을 기술하는 데이터에 의해 구동될 수 있다. 실제로, 네트워크 시뮬레이션의 구동 시에 사용도록 수집된 미처리 데이터는 여러 가지의 이유로 인해, 예컨대, 하드웨어, 소프트웨어, 및/또는 네트워크 에러로 인한 에러들을 포함할 수 있다. 네트워크 시뮬레이션을 구동하는 데 사용되는 데이터의 일관성을 보장하기 위해, 미처리 데이터를 클리닝시킬 수 있다. 예를 들면, 네트워크 내의 각 노드는 네트워크 시뮬레이션의 구동 시에 사용하기 위한 데이터를 제공할 수 있다. 특정 노드에 의해 제공된 데이터는 그 특정 노드의 동작 뿐 아니라, 하나 이상의 이웃하는 노드들의 동작도 기술할 수 있다. 그러므로, 네트워크의 노드로부터 얻어진 데이터는 중복될 수 있다. 이로써, 중복된 데이터를 서로 비교하여 불일치되는 것들을 식별하며, 이러한 불일치된 것을 데이터 평균화, 불일치 데이터의 제거 등을 통한 여러 방법으로 조정될 수 있다.

도면의 간단한 설명

[0001] 도 1은 복수의 노드를 갖는 네트워크을 도시하는 예시 구현예서의 환경을 도시.

[0002] 도 2는 도 1의 분석 모듈을 상세히 도시하는 예시 구현예를 도시.

[0003] 도 3은 7 ×3 그리드 형상을 갖는 네트워크를 도시.

[0004] 도 4는 도 2의 시뮬레이터 및 네트워크 시뮬레이션을 포함하는 시스템을 도시하는 예시 구현예를 도시.

[0005] 도 5는 서로 동일한 유형의 장애를 초기에 진단하는 예시 구현예에서의 절차를 도시하는 흐름도.

[0006] 도 6은 추정된 성능과 관찰된 성능 간의 차에 기초하여 장애의 유형을 판정하는 데 사용될 수 있는 예시 구현예에서의 판정 트리를 도시.

[0007] 도 7은 서로 다른 유형의 장애를 반복적인 진단 알고리즘을 사용하여 진단하는 예시 구현예에서의 절차를 도시하는 흐름도.

[0008] 도 8은 도 1의 복수의 노드가 인접 모니터링을 행하도록 실행가능한 에이전트 모듈을 포함하는 예시 구현예에서의 네트워크를 도시.

[0009] 도 9는 이웃하는 노드를 기술하는 레포트들을 비교하여 네트워크에서 잘못된 행동을 행하는 노드들의 위치를 찾아 내는 예시 구현예에서의 절차를 도시하는 흐름도.

[0010] 도 10은 온 라인 트레이스 구동식 시뮬레이션에 기초하여 상황 분석(what-if)을 수행하는 예시 구현예에서의 절차를 도시하는 흐름도.

[0011] 도 11은 네트워크에 대한 변경을 유해한 흐름의 진단에 기초하여 도출해 내는 예시 구현예에서의 절차를 도시하는 흐름도.

[0012] 도 12는 하나가 유해한 흐름인 복수의 흐름을 포함하는 네트워크를 도시.

[0013] 도 13은 네트워크 관리자에게 네트워크를 볼 수 있도록 하게 하여 그 네트워크에 대해 관리 요청을 발부하도록

하게 하는 관리자 노드에 의해 제공되는 그래픽 사용자 인터페이스(GUI)를 도시하는 예시 구현예를 도시.

[0014] <도면의 주요 부분에 대한 부호의 설명>

[0015] 106(n): 프로세서

[0016] 112(n): 에이전트 모듈

[0017] 108(n): 메모리

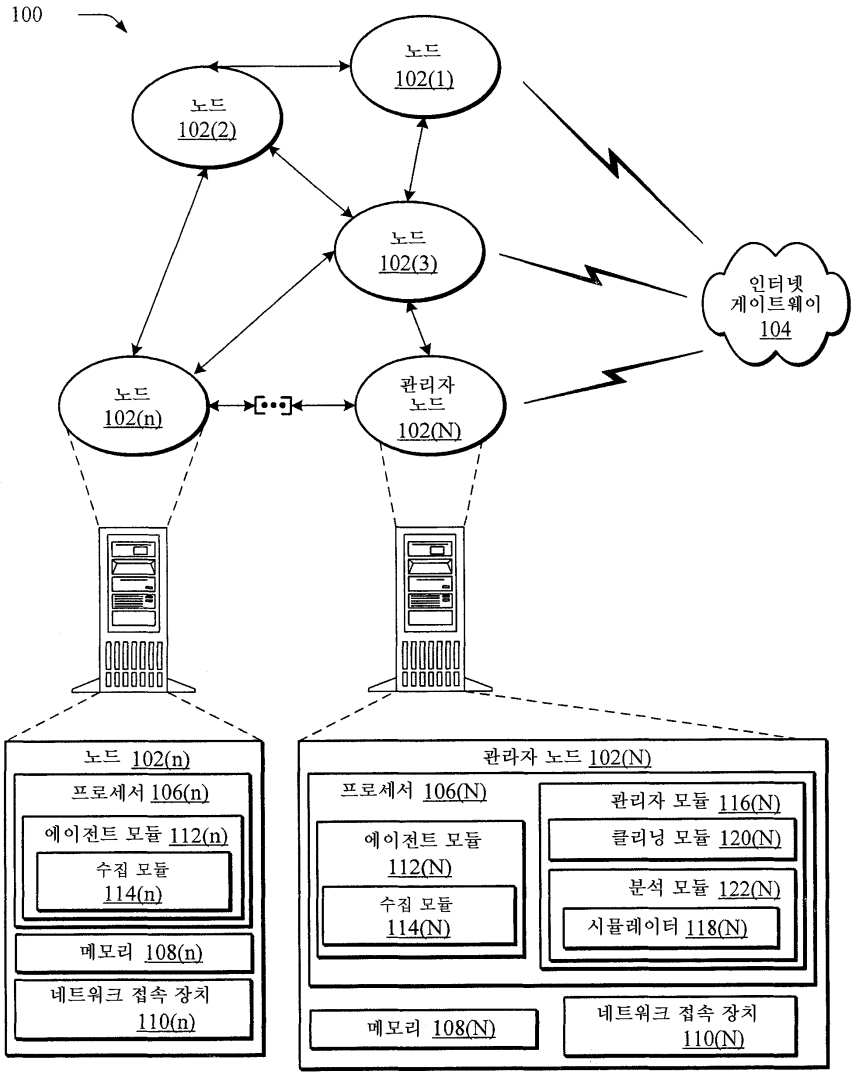
[0018] 118(N): 시뮬레이터

[0019] 122(N): 분석 모듈

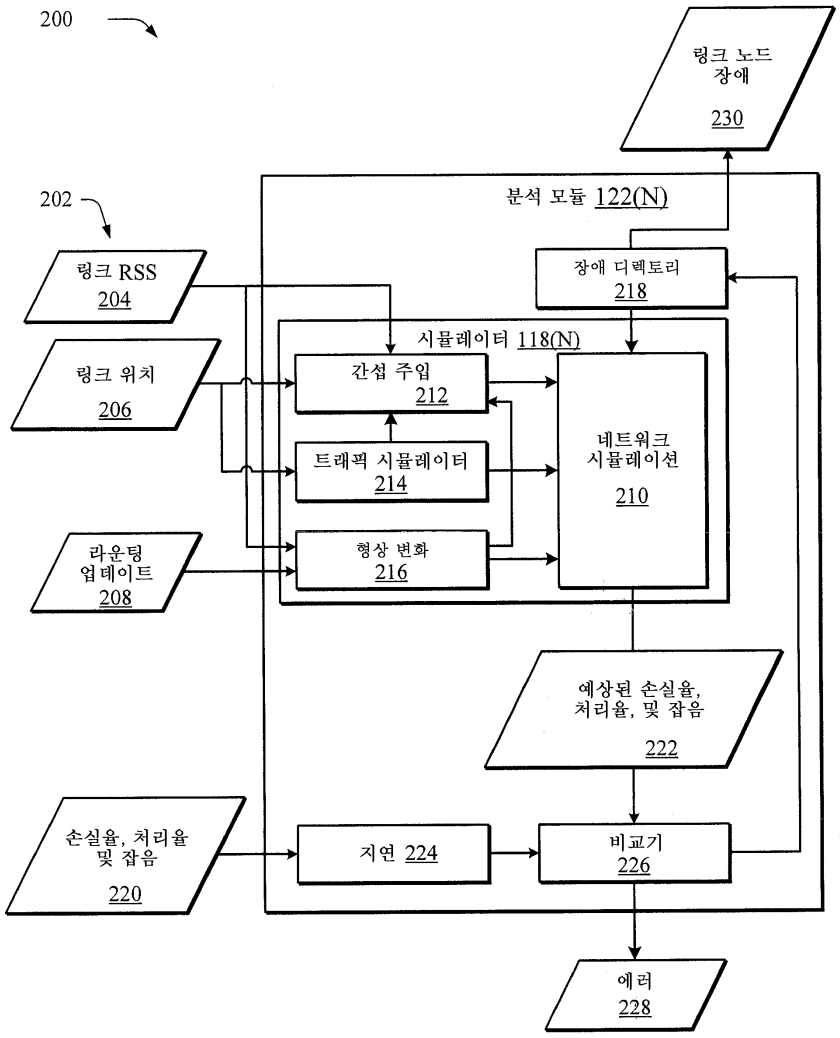
[0020] 404: 트래픽 통계

도면

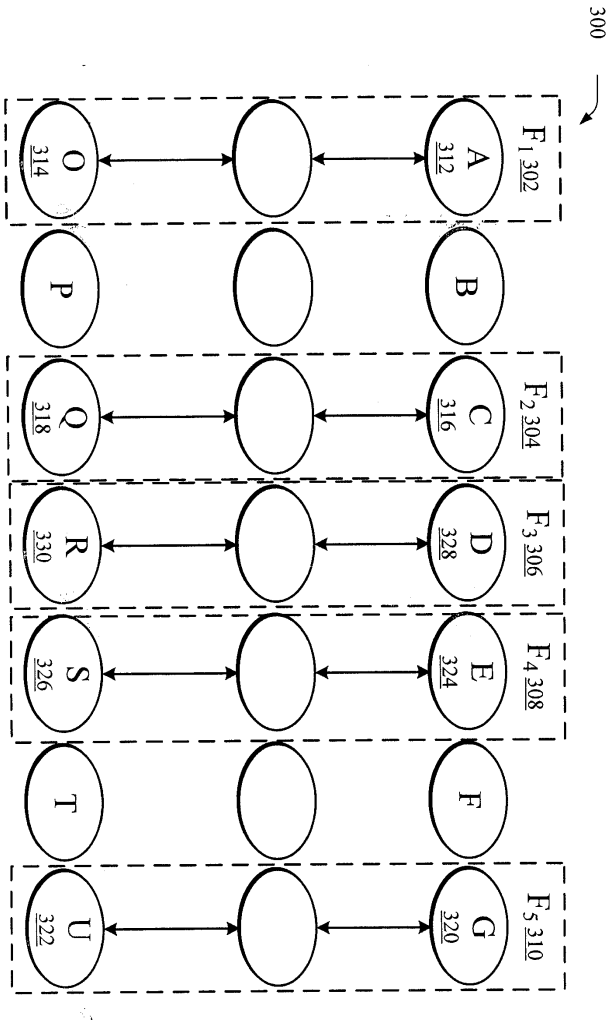
도면1



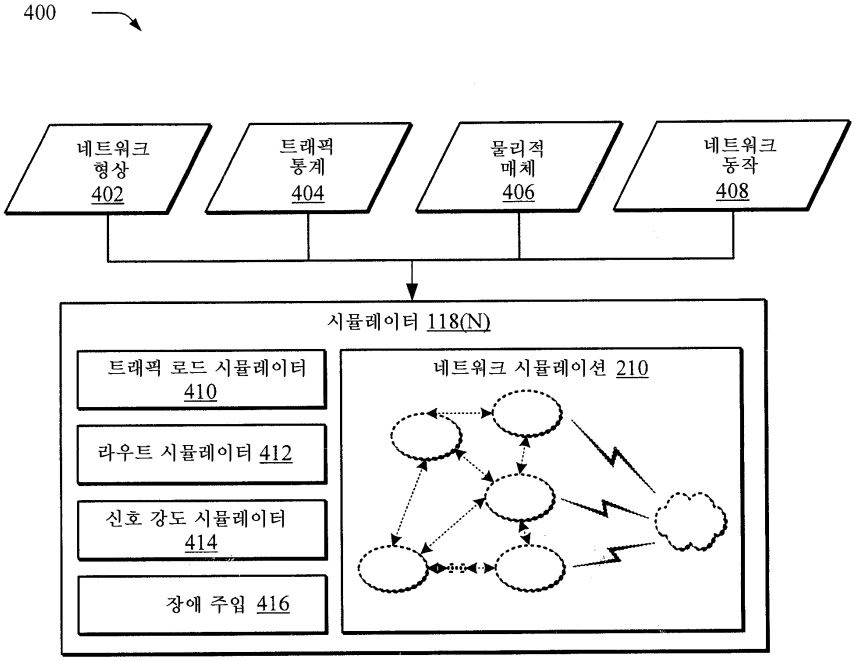
도면2


200
링크 노드 장애
230
202
분석 모듈 122(N)
링크 RSS
204
장애 디렉토리
218
시뮬레이터 118(N)
링크 위치
206
간섭 주입
212
네트워크 시뮬레이션
210
트래픽 시뮬레이터
214
라우팅 업데이트
208
형상 변화
216
예상된 손실율, 처리율, 및 잡음
222
손실율, 처리율 및 잡음
220
지연 224
비교기
226
에러
228

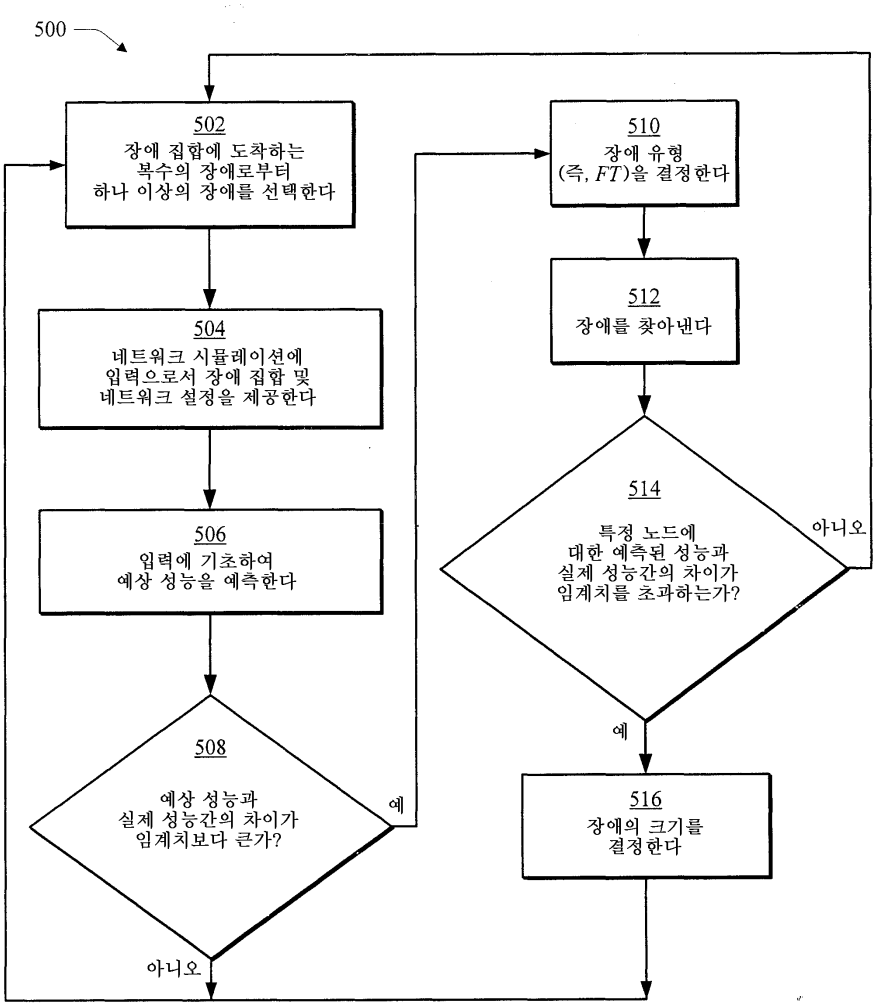
도면3


300
F1 302
A 312
O 314
B
P
F2 304
C 316
Q 318
F3 306
D 328
R 330
F4 308
E 324
S 326
F
T
F5 310
G 320
U 322

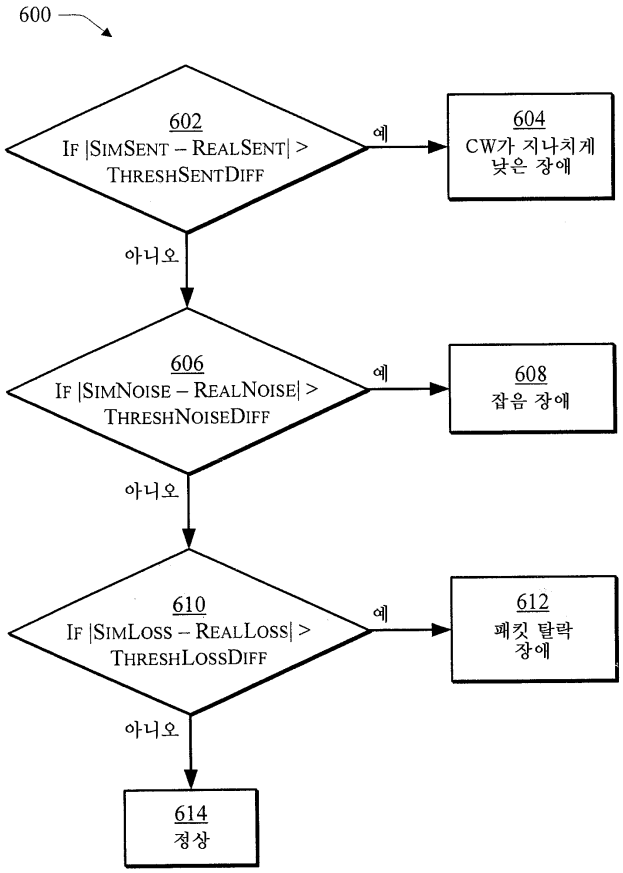
도면4


400
네트워크 형상 402
트래픽 통계 404
물리적 매체 406
네트워크 동작 408
시뮬레이터 118(N)
트래픽 로드 시뮬레이터 410
라우트 시뮬레이터 412
신호 강도 시뮬레이터 414
장애 주입 416
네트워크 시뮬레이션 210

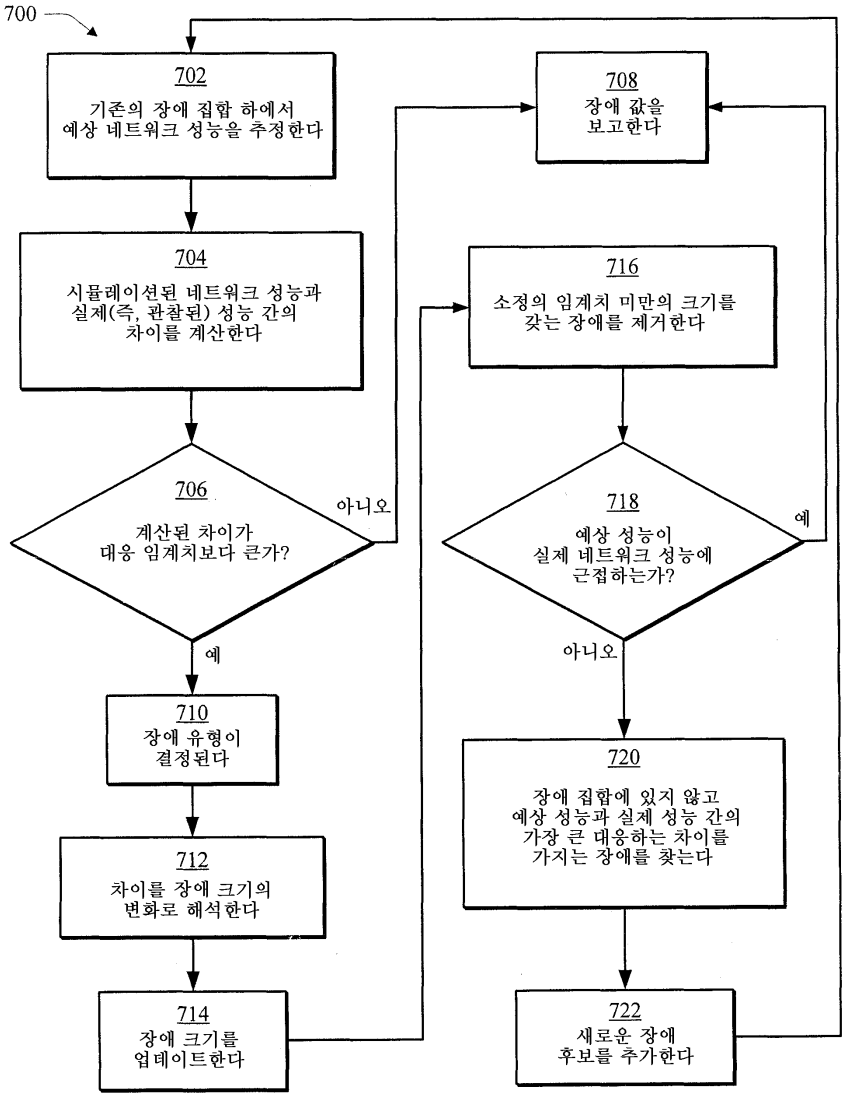
도면5


500
502
장애 집합에 도착하는
복수의 장애로부터
하나 이상의 장애를 선택한다
504
네트워크 시뮬레이션에
입력으로서 장애 집합 및
네트워크 설정을 제공한다
506
입력에 기초하여
예상 성능을 예측한다
508
예상 성능과
실제 성능간의 차이가
임계치보다 큰가?
예
아니오
510
장애 유형
(즉, FT)을 결정한다
512
장애를 찾아낸다
514
특정 노드에
대한 예측된 성능과
실제 성능간의 차이가
임계치를 초과하는가?
아니오
예
516
장애의 크기를
결정한다

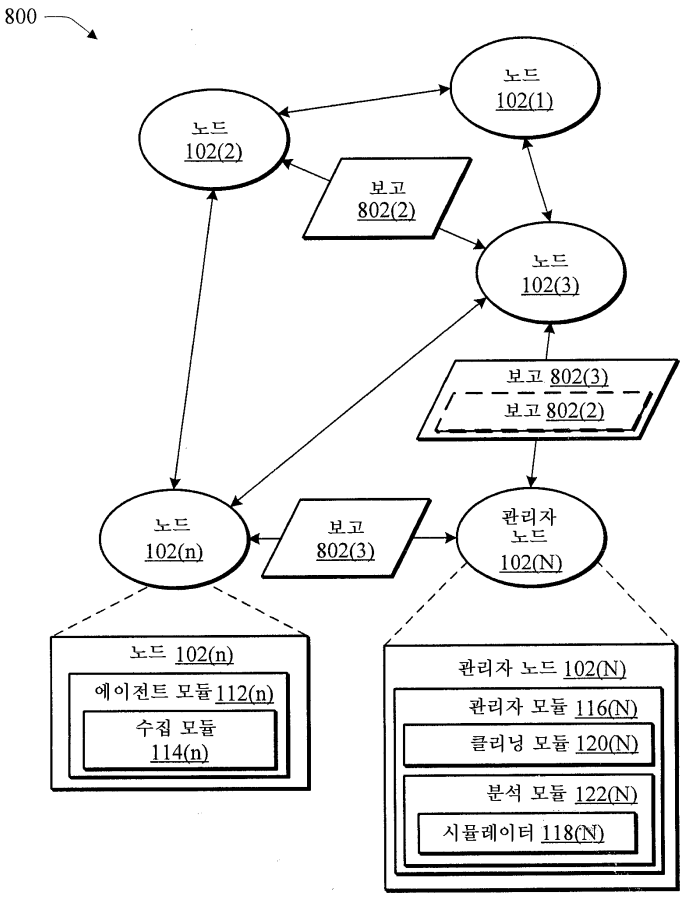
도면6


600
602
IF |SIMSENT – REALSENT| > THRESHSENTDIFF
예
604
CW가 지나치게 낮은 장애
아니오
606
IF |SIMNOISE – REALNOISE| > THRESHNOISEDIFF
예
608
잡음 장애
아니오
610
IF |SIMLOSS – REALLOSS| > THRESHLOSSDIFF
예
612
패킷 탈락 장애
아니오
614
정상

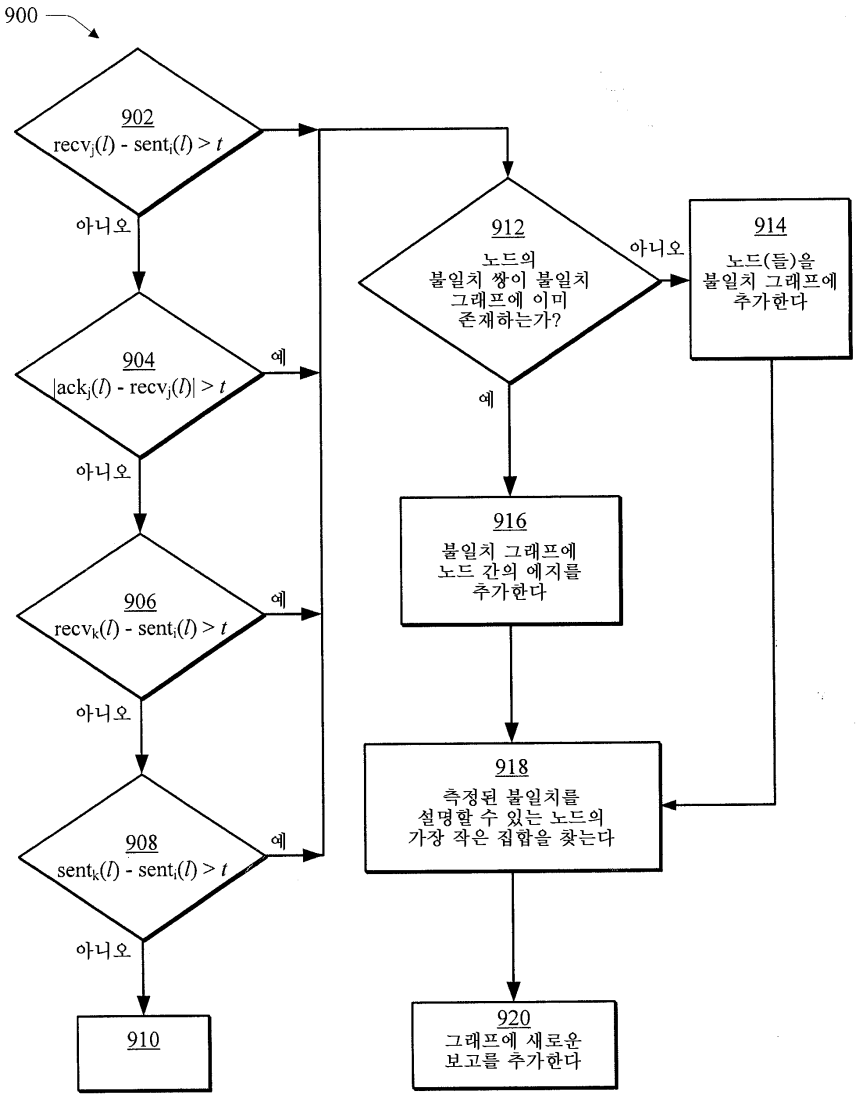
도면7


700
702
기존의 장애 집합 하에서
예상 네트워크 성능을 추정한다
704
시뮬레이션된 네트워크 성능과
실제(즉, 관찰된) 성능 간의
차이를 계산한다
706
계산된 차이가
대응 임계치보다 큰가?
아니오
예
708
장애 값을
보고한다
710
장애 유형이
결정된다
712
차이를 장애 크기의
변화로 해석한다
714
장애 크기를
업데이트한다
716
소정의 임계치 미만의 크기를
갖는 장애를 제거한다
718
예상 성능이
실제 네트워크 성능에
근접하는가?
예
아니오
720
장애 집합에 있지 않고
예상 성능과 실제 성능 간의
가장 큰 대응하는 차이를
가지는 장애를 찾는다
722
새로운 장애
후보를 추가한다

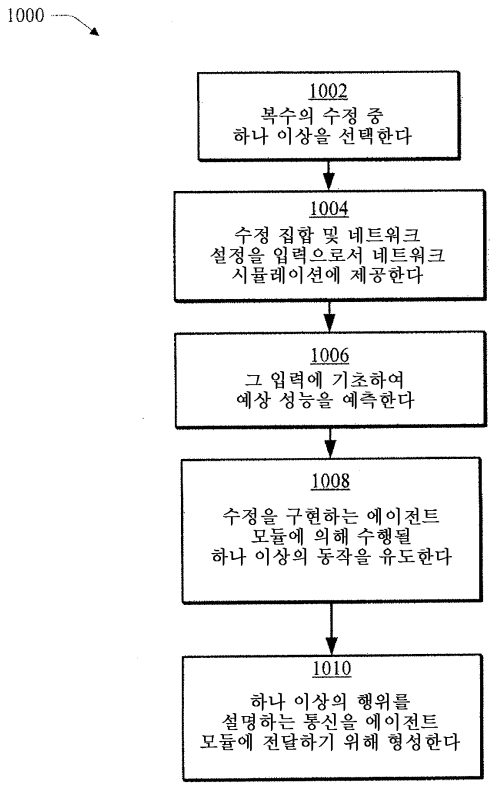
도면8


800
노드 102(1)
노드 102(2)
보고 802(2)
노드 102(3)
보고 802(3)
보고 802(2)
노드 102(n)
보고 802(3)
관리자 노드 102(N)
노드 102(n)
에이전트 모듈 112(n)
수집 모듈 114(n)
관리자 노드 102(N)
관리자 모듈 116(N)
클리닝 모듈 120(N)
분석 모듈 122(N)
시뮬레이터 118(N)

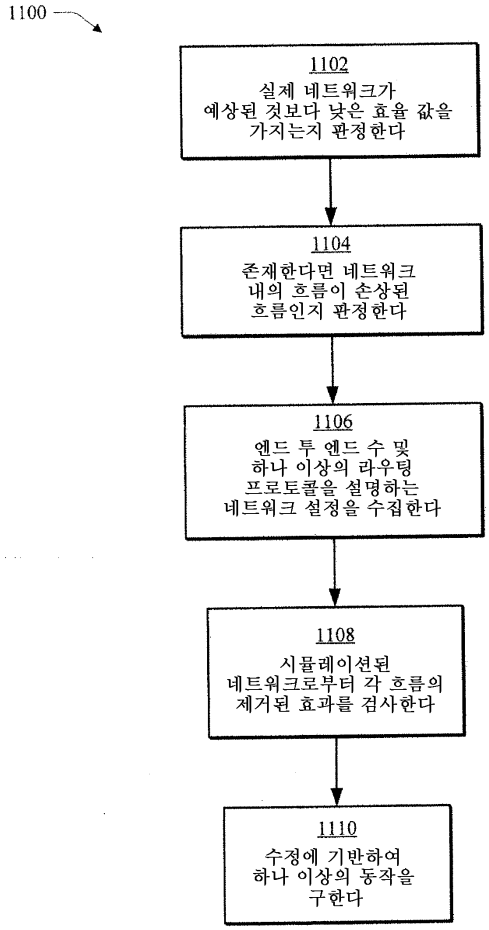
도면9


900
902
$recv_j(l) - sent_i(l) > t$
아니오
904
$|ack_j(l) - recv_j(l)| > t$
예
아니오
906
$recv_k(l) - sent_i(l) > t$
예
아니오
908
$sent_k(l) - sent_i(l) > t$
예
아니오
910
912
노드의 불일치 쌍이 불일치 그래프에 이미 존재하는가?
아니오
914
노드(들)을 불일치 그래프에 추가한다
예
916
불일치 그래프에 노드 간의 에지를 추가한다
918
측정된 불일치를 설명할 수 있는 노드의 가장 작은 집합을 찾는다
920
그래프에 새로운 보고를 추가한다

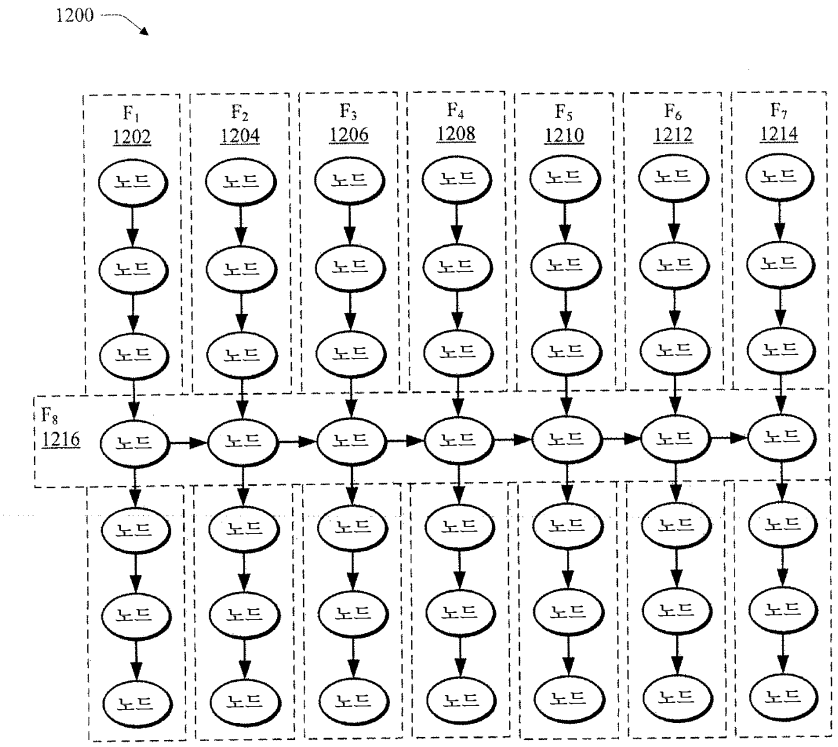
도면10


1000
1002
복수의 수정 중
하나 이상을 선택한다
1004
수정 집합 및 네트워크
설정을 입력으로서 네트워크
시뮬레이션에 제공한다
1006
그 입력에 기초하여
예상 성능을 예측한다
1008
수정을 구현하는 에이전트
모듈에 의해 수행될
하나 이상의 동작을 유도한다
1010
하나 이상의 행위를
설명하는 통신을 에이전트
모듈에 전달하기 위해 형성한다

도면11


1100
1102
실제 네트워크가
예상된 것보다 낮은 효율 값을
가지는지 판정한다
1104
존재한다면 네트워크
내의 흐름이 손상된
흐름인지 판정한다
1106
엔드 투 엔드 수 및
하나 이상의 라우팅
프로토콜을 설명하는
네트워크 설정을 수집한다
1108
시뮬레이션된
네트워크로부터 각 흐름의
제거된 효과를 검사한다
1110
수정에 기반하여
하나 이상의 동작을
구한다

도면12


1200
F1
1202
F2
1204
F3
1206
F4
1208
F5
1210
F6
1212
F7
1214
F8
1216
노드

도면13

