(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.[7]
G06F 13/14

(11) 공개번호 특2001-0040424
(43) 공개일자 2001년05월15일

(21) 출원번호 10-2000-7008178
(22) 출원일자 2000년07월22일
번역문제출일자 2000년07월22일
(86) 국제출원번호 PCT/US1999/01234 (87) 국제공개번호 WO 1999/38084
(86) 국제출원출원일자 1999년01월21일 (87) 국제공개일자 1999년07월29일
(81) 지정국 AP ARIPO특허 : 케냐 레소토 말라위 수단 스와질랜드 우간다 시에라리온 가나 감비아 짐바브웨

EA 유라시아특허 : 아르메니아 아제르바이잔 벨라루스 키르기즈 카자흐스탄 몰도바 러시아 타지키스탄 투르크메니스탄

EP 유럽특허 : 오스트리아 벨기에 스위스 리히텐슈타인 독일 덴마크 스페인 프랑스 영국 그리스 아일랜드 이탈리아 룩셈부르크 모나코 네덜란드 포르투칼 스웨덴 핀랜드 사이프러스

OA OAPI특허 : 부르키나파소 베넹 중앙아프리카 콩고 코트디브와르 카메룬 가봉 기네 말리 모리타니 니제르 세네갈 차드 토고 기네비쏘

국내특허 : 알바니아 아르메니아 오스트리아 오스트레일리아 아제르바이잔 보스니아-헤르체고비나 바베이도스 불가리아 브라질 벨라루스 캐나다 스위스 리히텐슈타인 중국 쿠바 체코 독일 덴마크 에스토니아 스페인 핀랜드 영국 그루지야 헝가리 이스라엘 아이슬란드 일본 케냐 키르기즈 북한 대한민국 카자흐스탄 세인트루시아 스리랑카 라이베리아 레소토 리투아니아 룩셈부르크 라트비아 몰도바 마다가스카르 마케도니아 몽고 말라위 멕시코 노르웨이 뉴질랜드 슬로베니아 슬로바키아 타지키스탄 투르크메니스탄 터어키 트리니다드토바고 우크라이나 우간다 우즈베키스탄 베트남 폴란드 포르투칼 루마니아 러시아 수단 스웨덴 싱가포르 아랍에미리트 안티구아바부다 코스타리카 도미니카연방 알제리 모로코 탄자니아 남아프리카 벨리즈 모잠비크

(30) 우선권주장 60/072,128 1998년01월22일 미국(US)
(71) 출원인 인텔로지스

미국, 유타주 84020, 드래퍼, 사우스비지니스 파크드라이브, 12257, 슈트 108

(72) 발명자 웨일버크앨런

미국,유타주84093,샌디,플라타웨이,1746이

리토마스앤

미국,유타주84660,스패니쉬포크,850사우스,1051이

(74) 대리인 황이남, 박형준

심사청구 : 없음

(54) 공통 데이타교환 게이트웨이의 방법 및 장치

요약

하나 이상의 네트워크 프로토콜 및 제어 프로토콜 사이에 데이터를 전달하는 공통 게이트웨이(104)를 설명한다. 각종 프로토콜이 동일한 물리적 네트워크 매체(100) 혹은 개별 네트워크(100,130)에 공존할 수 있다. 게이트웨이(104)를 사용해서 엔드유저는 전통적인 비호환성 네트워크를 공통 접근 중앙접근식 "슈퍼네트워크"로 전환시킬 수 있다. 게이트웨이(104)는 중앙집중식 노드 데이터베이스를 제공하고 연속프로토콜 및 규칙엔진 대상지원 클라스 라이브러리 인터페이스를 지원한다. 중앙집중식 노드 데이터베이스에 대한 고신뢰성 접근 방식은 대기서버 노드에 의한 폴트 허용에 의해 개선된다. 또한 게이트웨이(104)는 전선을 통한 다양한 데이터흐름을 분배시키는 능력도 갖는다. 라우팅 처리기는 TCP/IP 같은 연속데이타 네트워크 서비스를 가상허용한다.

대표도

도1

명세서

기술분야

본 발명은 컴퓨터 네트워크 프로토콜 게이트웨이 특히 전력공급선(즉, 전원) 네트워크장치에 적용되는 게이트웨이에 관계한 것이다.

배경기술

컴퓨터 특히 퍼스널컴퓨터의 광범위한 기능성은 컴퓨터 네트워크 수의 급속한 증가를 가져왔다. 2개 이상의 컴퓨터를 네트워크 연결하여 정보, 자료, 프린터 등을 공유할 수 있다. 2개 이상의 퍼스널컴퓨터 및 프린터를 연결하여 네트워크를 구성하는 것은 원리상 간단한 작업이다. 컴퓨터와 프린터는 케이블로 간단히 연결되며 필요한 소프트웨어를 각 컴퓨터 상에 인스톨 하기만 하면 된다. 상기 네트워크 분야의 용어에서, 케이블은 네트워크 매체, 컴퓨터와 프린터는 네트워크 노드 라고 한다. 네트워크 노드는 전송제어프로토콜, 인터넷 프로토콜(TCP/IP) 같은 하나이상의 프로토콜을 이용하여 서로 "이야기"한다.

게이트웨이는 한 프로토콜에서 다른 프로토콜로 프로토콜을 번역하는 기능을 하며 따라서 다른 프로토콜을 사용하는 2개의 네트워크를 상호접속시킬 수 있다. 예를들어, 프로디지 네트워크 서비스는 인터넷 전용 이-메일 포맷 및 인터넷 이-메일을 번역하는 게이트웨이를 갖는다.

표준네트워크 프로토콜은 전형적으로 각 네트워크 노드가 실질상의 처리 및 저장용량을 갖는 "스마트" 장치 라는 가정하에 설계된 것이다. 예를 들어 일반 퍼스널컴퓨터(PC)는 거의 모든 네트워크 프로토콜을 처리하기에 충분한 처리 및 저장용량 보다는 더 크다. 그러나, 일반 프린터는 상기의 처리 및 저장용량을 갖지 않는 "단순한" 장치이다. 프린터를 네트워크에 연결할 수 있게 해주는 네크워크 프린터 어댑터를 생산하는 회사들도 있다. 프린터 어댑터는 완전구성 PC와 유사한 처리 및 저장용량을 제공하는 싱글보드 컴퓨터이다. 네트워크 프린터 어댑터는 따라서 "단순한" 장치를 "스마트" 장치로 전환시켜준다. 그러나 이러한 역할을 하는 네트워크 프린터 어댑터는 가격이 아주 비싸기 때문에 가정용 및 소기업 환경에서는 대부분 사용하기에 적합치 않다. 더우기, 프린터 어댑터는 다른 비-PC 장치를 네트워크에 연결하기에는 그리 적합치 않다. 예를들어, 사용자는 현관외등, 경보장치, 전화장치 등의 단순 장치를 컴퓨터 네트워크에 연결하고 싶어하는 경우가 많다. 단순 장치를 스마트 장치로 전환시키기 위한 네트워크 어댑터 카드는 가격이 비싸다.

스마트 장치에 이용되는 프로토콜은 일반적으로 "네트워크 프로토콜" 이라고 한다. 단순 장치에 사용되는 프로토콜은 "제어 프로토콜" 이다. 네트워크 프로토콜은 제어 프로토콜과 용량 및 복잡성 측면에서 매우 상이하다. 이러한 차이점 때문에 네트워크 프로토콜 간에 데이타를 전송하도록 설계된 게이트웨이는 제어 프로토콜간 데이타 전송작업에 부적합한 것이 보통이다. 더 큰 난제는 네트워크 프로토콜 및 제어 프로토콜 간의 데이타 전송작업이다.

기존의 가정 관리/자동화 제품들은 중앙집중식 클라이언트/서버 모델이 아닌 집단-집단형 제어 프로토콜을 사용하는 경향이었다. 이것은 제품의 실용성을 크게 제한하는 것으로써, 그 이유는 각 노드가 상황정보 및 규칙을 지역적으로 저장하도는데 필요하기 때문이다. 네트워크 구성은 사용의 용이성이 부족하거나 중앙집중식 사용사 인터페이스 요소 때문에 어려워지는 경우가 많다. 또한, 경쟁사 제품간 상호운용성 (예, X-10 및 CEBus)은 사실상 불가능하다.

발명의 상세한 설명

본 발명은, 상기 및 기타의 문제들을 하나 이상의 네트워크 프로토콜 및 하나이상의 제어 프로토콜 간에 데이타를 전송할 수 있는 저렴하고 사용이 간편하며, 융통성, 신뢰성 및 기밀성있는 게이트웨이 아키텍춰를 제공함으로써 해결한다. 다양한 제품들이 동일한 물리적 네트워크 매체 상에서 공존할 수가 있는 것이다. 게이트웨이는 또한 네트워크 프로토콜을 선택된 프로토콜에 통과시키고, 중앙집중식 제어도 가능하게 해준다.

따라서, 게이트웨이는 특히 가정 및 소기업 환경에서 데이타 및 제어 네트워크의 최종 사용자에게 다수의 잇점을 제공해준다. 게이트웨이를 통해, 최종 사용자는 쉽고 편리하게 전통적인 독립식, 비호환식 네트워크를 하나의 공용접근식 중앙통제형 "수퍼 네트워크" 속에 통합시킬 수 있으며, 이 네트워크는 컴퓨터, 프린터, 경보장치, 가전제품, 전등, 전화 등을 서로 연결시킨다.

게이트웨이는 TCP/IP 같은 연결 프로토콜을 지원하는 중앙집중식 노드 데이타베이스, 규칙 엔진, 목적지향성 라이브러리 인터페이스 등을 제공한다. 게이트웨이는 인터넷 브라우저 같이 사용이 간편한 공동의 그래픽형 사용자 인터페이스를 사용하여 광역 가정/소기업 자동화 및 제어기능을 제공한다. 중앙집중식 노드 데이타베이스에 대한 접근은 대기 서버에서 제공된 시스템 오류 허용에 의해 개선된다.

전원 네트워크와 접속 사용할 때, 게이트웨이는 전원에 대해 각종 데이타흐름를 배급하는 기능을 제공한다. 예를 들어, 케이블모뎀 혹은 기타의 다양한 형태로 고속 인터넷 접속하는 네트워크 사용자는 가정/회사 어느 곳에서나 기존의 배선 이외에 다른 별도의 배선이 필요없이 인터넷 교통을 배급할 수 있다. 게이트웨이가 제공하는 라우팅 조정기는 실질적으로 연속데이타 네트워크 서비스 및 현재 공용중인 프로토콜을 전원 전체에 라우팅될 수 있도록 해준다.

본 발명의 장점 및 특징은 이 분야의 통상의 지식을 가진 자라면 첨부도면과 함께 더 자세히 기술되는 하기의 내용으로부터 더욱 용이하게 이해할 수 있다.

도면의 간단한 설명

도 1은 퍼스널 컴퓨터와 같은 스마트 노드 및 외부보안등과 같은 단순 노드를 갖춘 네트워크의 흐름도,

도 2는 7층 OSI 네트워크 모델의 흐름도,

도 3은 공동 게이트웨이 아키텍춰의 흐름도,

도 4는 서버 작동 알고리즘을 보여주는 플로우차트,

도 5는 규칙의 요소들을 보여주는 데이타차트,

도 6은 스마트 장치의 PLX 네트워크 모델의 흐름도,

도 7은 단순 장치의 PLX 네트워크 모델의 흐름도,

도 8은 매체접근 알고리즘을 보여주는 플로우차트,

도 9A는 작동 네트워크 서버 스피팅(spitting) 알고리즘을 보여주는 플로우차트,

도 9B는 클라이언트 스피팅 알고리즘을 보여주는 플로우차트,

도 10은 작동 네트워크 서버 폴링(polling) 알고리즘을 보여주는 플로우차트,

도 11은 PLX 논리그룹 분리(LoGI)패킷의 영역을 보여주는 흐름도,

도 12는 PLX 미가공데이타 패킷의 영역을 보여주는 흐름도,

도 13은 PLX 토큰패킷의 영역을 보여주는 흐름도,

도 14는 PLX 직접수령(DACK) 패킷의 영역을 보여주는 흐름도,

도 15은 PLX 차단구성 삽입패킷(LIPG)의 영역을 보여주는 흐름도,

도 16은 PLX 직접구성 삽입패킷(LIPD)의 영역을 보여주는 흐름도,

도 17은 PLX 내부 호스트패킷의 영역을 보여주는 흐름도,

도 18은 PLX 공용언어(CAL) 요청패킷을 보여주는 흐름도,

도 19는 PLX CAL 응답패킷을 보여주는 흐름도,

도 20은 PLX 단일채널 전송상태 패킷을 보여주는 흐름도,

도 21은 PLX 다중채널 전송상태 패킷을 보여주는 흐름도,

도 22는 PLX 패킷 타이밍을 보여주는 타이밍 다이어그램.

실시예

상기 도면에서 3자리 숫자의 첫번째 자리는 해당요소를 처음 도시한 도면의 도면부호를 표시한다. 4자리의 참조부호에서는 처음 2자리가 도면의 부호를 가리킨다. 도 1은 공동 게이트웨이와 함께 사용하기 적합한 컴퓨터 네트워크 장치를 도시한다. 도 1의 장치는 2개의 다른 물리적 네트워크를 보여준다. 제1 물리적 네트워크는 네트워크 매체(100) (케이블로 표시됨)를 이용한다. 스마트 노드(퍼스널 컴퓨터(103)로 표시됨)는 커넥터(102)에 의해 네트워크 매체(100)에 연결된다. 프린터(110), 컴퓨터(104) 및 보안등 장치(118) 역시 네트워크 매체(100)에 연결된다. 조명 장치(118)는 연산 전력공급이나 저장량이 아주 작은 "단순" 노드의 한 예이다. 컴퓨터(104)는 또한 공유교환 전화 네트워크(PSTN)로 도시된 제2 네트워크(130)에 연결된다.

전형적인 네트워크 환경에서, 네트워크 매체(100)는 다양한 데이타 프로토콜을 위한 데이타소통을 실행하도록 구성된다. 따라서, 도 1의 실시예에서 보는 바와 같이 컴퓨터(103)는 TCP/IP 프로토콜을 이용하여 컴퓨터(104)와 통신할 수 있으며 컴퓨터(104)는 부록A에 표시한 설명부분과 같이 제어 프로토콜, 즉 전원 교환(PLX) 프로토콜을 이용하여 조명장치(118)와 소통한다. 부록 A는 미국특허출원 제09/211950호("전원 교환 프로토콜을 위한 방법 및 장치")의 일부를 포함한다.

공통 게이트웨이는 예컨대 컴퓨터(104) 상의 소프트웨어 프로그램 역할을 한다. 공통 게이트웨이는 다른 네트워크 프로토콜 사이 및 다른 물리적 네트워크 사이를 연결한다. 예를들어, 도 1에서 보는 바와 같이 컴퓨터(104)에서 소프트웨어 프로그램 역할을 하는 공통 게이트웨이는 TCP/IP 프로토콜을 PLX 프로토콜에 연결하여 컴퓨터(103)로 하여금 조명장치(118)과 소통하도록 한다.

컴퓨터(104) 상에서 실행되는 공통 게이트웨이 소프트웨어는 또한 개별 물리적 네트워크 사이에 데이타를 전송하고 이에의해 개별 물리적 네트워크 상에서 장치간에 소통이 이루어지도록 한다. 도 1에서, 공통 게이트웨이는 네트워크(130) 및 조명장치(118) 사이에 데이타를 전송한다.

공통 게이트웨이는 적층형(layered) 네트워크 프로토콜과 호환한다. 스마트 노드(컴퓨터(103) 및 (104) 등의)를 위해 구성된 대부분의 네트워크는 개방장치 인터페이스(OSI) 협회에서 개발한 네트워크 아키텍춰 모델에 기초한다. OSI 아키텍춰는 통신장치 속의 개별 하드웨어 및 소프트웨어층, 층간의 상호연계성 및 각 층의 특별한 수행기능 등을 개괄적으로 나타낸 네트워크를 규정한다.

도 2는 최저층에서 최고층까지 7개의 층으로 구성된 OSI 아키텍춰를 도시한다. 이는 물리적 층(201), 데이타링크층(202), 네트워크층(203), 전달층(204), 실행층(205), 표시층(206) 및 응용층(207)으로 구성된다. 각 층은 자기 아래층을 이용하며 자기 윗층으로 서비스를 제공한다. 층 자체가 서브층으로 구성된 경우도 있다. 이 층은 특별한 네트워크 프로토콜을 운영하는 2개 이상의 통신장치 또는 컴퓨터의 소프트

웨어 및/또는 하드웨어 환경이다. 네트워크 접속은 각각이 다른 층 혹은 레벨로 된 1군의 다소 독립된 프로토콜로 생각할 수 있다. 최저층은 다른 노드에서 하드웨어 사이의 직접 노드-노드 통신을 통제하고 최고층은 사용자 어플리케이션으로 구성된다. 각 층은 바로 아랫층을 이용하며 반대로 바로 윗층에 서비스를 제공한다. 하나의 호스트 상의 각 네트워크 요소 하드웨어 혹은 소프트웨어는 이것의 층에 적절한 프로토콜을 이용하여 또다른 노드 상의 대응되는 요소(이것의 "집단")과 통신한다. 이러한 적층 프로토콜은 집단-집단 프로토콜이 라고도 한다.

적층 프로토콜의 장점은 한 층에서 다른 층으로 정보를 전달하는 방법이 프로토콜의 일부로 구체적으로 구현되고, 또한 하나의 프로토콜층 내의 변화는 다른 프로토콜층에 영향을 미치지 않도록 방지된다. 이 때문에 통신장치의 설계 및 관리 작업이 간단해진다.

물리적 층(201)은 OSI 적층 모델 내의 최저층이다. 이것은 매체접근 제어부(MAC)의 일부를 포함한 네트워크의 전기 및 기계적 접속부를 포함한다. 매체접근 제어부(MAC)는 데이타 전송매체(100)(즉, 네트워크 케이블)을 제어 및 이것에 접근하는 것에 관계한다. 물리적 층(201)은 데이타링크층(202)에 의해 사용된다.

데이타링크층(202)은 OSI모델 내의 제2 최저층이다. 데이타링크층(202)은 데이타를 프레임으로 분할하고(이것은 물리적 층(201)에 전달된다), 수령확인 프레임을 수용한다. 데이타링크층(202)은 오류검사 및 올바르게 수령되지 않은 프레임의 재전송을 수행한다. 데이터링크층(202)은 네트워크층(203)에 오류없는 가상채널을 제공한다. 데이터링크층(202)은 일반적으로 상부 서브층, 논리링크 제어부(LLC), 또한 매체접근 제어부(MAC)를 포함한 하부 서브층으로 분할된다.

네트워크층(203)는 OSI 7개층 모델 내에 있는 제3 최저층이다. 네트워크층(203)은 데이터링크층(202)을 경유하여 전송자에서 수신자로 향하는 데이터패킷의 경로를 결정하며 전송층(204)에 의해 이용된다. 가장 일반적인 네트워크층 프로토콜을 IP 이다.

전송층(204) (혹은 "호스트-호스트 층")은 OSI모델 내에 있는 중간층이다. 전송층(204)은 가상의 무오류, 점-점 접속을 제공하기 위해 네트워크층(203)을 이용하는 방법을 결정하며 이에 따라 제1 노드가 제2 노드에 메시지를 전송할 수 있고 오류가 없는 메시지가 정확한 순서로 도달하게 된다. 전송층(204)은 노드간의 접속을 형성하거나 소멸시킨다.

세션층(session layer)(205)는 OSI 모델 내에 있는 제3 최고 프로토콜층이다. 세션층(206)은 텍스트압축, 코드 혹은 포맷 전환 등 노드간 차이를 완화시키기 위한 기능을 수행한다. 표시층(206)은 응용층의 비호환형 프로세스를 세션층을 통해 소통시켜준다.

응용층(207)은 OSI 모델의 최고층이다. 응용층(207)은 사용자측의 네트워크 관찰(예, 포맷팅 전제메일 메시지)에 관련있다. 표시층(206)은 네트워크에 사용되는 포맷과 무관한 데이터를 응용층(207)에 제공한다. 응용층 프로토콜은 예를 들어, 텔넷, 파일전송프로토콜(FTP), 단순 네트워크 관리프로토콜 (SNMP), 단순 메일전송 프로토콜(SMTP), 인터넷 제어메시지 프로코로(ICMP), 넷웨어 코어 프로토콜(NCP), 라우팅 정보 프로토콜(RIP), 서비스광고 프로토콜(SAP), 트리바이얼 전송 프로토콜(TFTP) 및 시스템오류 허용 프로토콜(SFTP) 등이 있다.

도 3은 공통 게이트웨이(300)의 아키텍춰 및 게이트를 어플리케이션, 하드웨어 드라이버 등 기타 소프트웨어 요소와 상호작용시키는 방법을 보여준다. 어플리케이션(302)는 1군의 게이트웨이 파운데이션 클래스(304)를 이용하여 게이트웨이(300)와 통신한다. 게이트웨이 파운데이션 클래스(304)는 데이터베이스 접근모듈(306), 사건처리기(310), 디스패쳐(320) 및 실시간 운영장치 서비스(RTOS: 311)와 통신한다. 데이터베이스 접근모듈(306)은 정보저장소(노드 데이터베이스: 308)를 포함한다. 노드 데이터베이스(308)는 데이터베이스, 링크리스트, 테이블, 지도, 컨테이너 등으로 구성될 수 있다. 데이터베이스 접근모듈(306)은 또한 사건처리기(310) 및 페이로드/프로토콜 처리기(312)와도 통신한다. 페이로드/프로토콜 처리기는 미가공데이타 처리기(316), 흐름데이터 처리기(317) 및 CAL 제어처리기(318) 등의 하나이상의 페이로드/프로토콜 처리기를 포함한다. 사건처리기(310)는 또한 규칙엔진(314) 및 디스패처(320)과 소통한다. 디스패처(320)는 RTOS(311), 규칙엔진(314), PLX장치 드라이버(322) 및 연속장치 드라이버(326)과 소통한다. PLX장치 드라이버는 PLX 제어장치 드라이버(323)(단순 장치의 경우) 및 PLX 데이터장치 드라이버(324)(스마트 장치의 경우)를 포함한다.

게이트웨이(300)는 한 프로토콜에서 다른 프로토콜로의 프로토콜 번역을 실행하여 다른 프로토콜을 사용하는 2개의 네트워크를 상호연결시킬 수 있다. 네트워크는 실제의 물리적 네트워크 및 가상 네트워크(즉, 소프트웨어 상에서만 존재하는)일 수 있다. 게이트웨이(300)는 연속프로토콜 및 1차(필요한 경우) 프로토콜(즉, 전원 프로토콜) 사이에 인터페이스를 제공한다. 연속 프로토콜은 기존의 프로토콜에 국한되지 않는다. 즉, 연속 프로토콜은 1차 프로토콜 이외의 다른 프로토콜에도 적용된다. 연속장치 드라이버는 예를들어, X-10드라이버(329) 및 CEBus 드라이버(332)를 포함한다. 각 연속장치 드라이버는 또한 선택적으로 연속장치 드라이버를 게이트웨이에 적용시킬 연결기(shim)을 포함한다. 한 예에서, 1차 프로토콜은 전원 프로토콜이나 게이트웨이(300)는 여기에 한정되지 않는다. 따라서, 1차 프로토콜은 TCP/IP, PIX, ADSL 등을 포함한 프로토콜일 수 있으며 기타의 여기에서 특별히 언급하지 않은 프로토콜도 포함한다.

이더넷 드라이버(362) 및 ADSL 드라이버(364) 등의 흐름연속장치 드라이버(362)에 있어서, 게이트웨이는 연속스택(352)를 제공한다. 연속스택(352)은 TCP/IP 스택(353) 및 SPX/IPX 스택(354) 등의 OSI 프로토콜 스택을 지원한다. 연속스택(352)은 흐름연속장치 드라이버(362) 및 라우팅 처리기(355)와 통신한다. 라우팅 처리기(355)는 흐름연속장치 드라이버(362), 연락스택(352) 및 페이로드/프로토콜 처리기(312)와 통신한다. 연속스택(352)는 또한 연속 어플리케이션(350)와도 통신한다.

게이트웨이(300)는 다양한 네트워크 간의 접속점 역할을 하며 제한없이 이더넷, 디지탈통신망(DSL 및 ADSL), 전원 교환기(PLX), X-10 및 CEBus 등을 포함한 제어기 및 통신 네트워크 양측을 지원한다.

게이트웨이(300)는 2가지 기본기능 즉, 노드관리 및 데이터 라우팅 기능을 갖는다. 노드관리기로서 게이트웨이(300)는 네트워크 노드상태 정보를 발견, 목록화, 검색, 저장 및 가공하는 역할을 한다. 노드 데이터베이스(309)는 일반 공용언어(CAL) 사양에 기초한다. CEBus는 EIA-600 으로 정의되며 버스 장치들을 제어하기 위한 공업표준 제어언어이다. EIP-600 은 가정용 LAN에 사용할 공용언어를 위한 기본구조를 제공한다. 일반CAL은 EIA-721 시리즈 표준(EIA-721.1, EIA-721.2, EIA-721.3 및 EIA-721.4) 으로 정의된다. CEBus 공업협회(CIC)에서 상기 일반CAL 이용에 관한 "문법적"규칙을 정의하여 상기 기본구조를 보완하는 가정용 플러그&플레이(HPP) 사양을 마련하였다.

HPP 사양은 가정의 조건에 맞는 형태로 실행할 수 있는 제품 및 장치의 실행특성 일체에 대해 상세히 설명하고 있다. 예를들어, 상기 사항에서는 "사용자가 외출중" 혹은 "집에서 잘 때" 등의 다양한 조건에서 각 시설들이 보안장치 조작, 실내등 소등 혹은 온도 조정 등의 적절한 기능을 실행할 수 있도록 하는 것에 대해 상세히 기술하였다. HPP 사양은 또한 가정관리를 위한 윈도우95급 어플리케이션을 개발하는 것에 대한 정보도 수록하였다.

EIA-600에서 정의된 공용언어는 광범위한 산업분야(즉, 오락, 컴퓨터, 냉/온방, 주방가전제품 등)에서 생산되는 가정용 LAN 제품들의 통신을 위한 기본구조를 제공한다. 각 산업부문은 제품이 사용할 언어에 관련하여 "응용문맥" (즉, 문법적 규칙)를 정한다. CIC는 광범위한 산업분야가 "조화적인" 응용문맥을 개발 할 수 있도록 돕는 지원구조 역할을 위하여 개발한 것이다. CIC의 HPP는 상호운용이 가능한 CAL형 제품들을 가정용 LAN시장에 공급하는 산업부문의 조화적 응용문맥의 개요이다.

CEBus/일반CAL 사양을 참조한다.

게이트웨이(300)은 또한 노드상태 변화를 관찰할 규칙엔진(314)을 제공하며 공통규칙 정의언어(RDL)론에 따라 정의된 규칙에 의거하여 상태변화에 대응한다.

또한 게이트웨이(300)는 데이터-흐름 및 라우팅 작업을 디스패치 제어블록(DCBs)를 이용하여 처리한다. 라우팅 처리기(355)는 PLX, 프록시서버, 및 네트워크 어드레스번역 등에 대한 TCP/IP 터널링 등의 기능을 위해 제공된다. 게이트웨이 파운데이션 클래스(304)는 사용자 인터페이스 및 시스템관리 어플리케이션(302)을 이용하여 사용할 목적지향 API 라이브러리를 제공한다. 게이트웨이(300)는 또한 연결, 비-공통망 APIs(MFC, TLI, 소켓, 가정API 등)을 이용하여 네트워크 노드와 통신할 응용부문을 위해 별도의 지원을 한다. 이것은 네트워크에 연결망 데이터(이더넷, ADSL 등)의 데이터흐름을 분명히하고 자바 및 웹 브라우저를 사용하는 게이트웨이(300)가 관리하는 노드의 제어역할도 포함한다.

노드 데이터베이스(308)는 게이트웨이(300)가 저장한 노드 데이터의 저장소이다. 노드데이타는 각 클라이언트 노드, 사건 대기행렬, 클라이언트 노드 결합 및 규칙 등으로부터 얻은 노드구성 프로파일을 포함한다. 게이트웨이(300)의 다른 요소들은 일련의 접근방법(306)을 통해 노드 데이터베이스(308)에 접근한다.

사건처리기(310)와 규칙엔진(314)은 클라이언트노드에서 발생하는 상황변화 결과로 일어나는 활동을 관리한다. 규칙엔진(314)은 하나의 상황변화에 관련된 규칙을 해석하고 규칙평가의 결과로 생길 수 있는 통지 혹은 CAL 전송요청에 대한 스케줄설정도 관리한다. 규칙엔진(314)은 노드 데이터베이스(308) 및 사건처리기(310)과 함께 실행된다. 사건 처리기(310)는 사건 대기행렬을 처리하고 사건통지를 실행한다. 이들 사건통지는 게이트웨이(300)의 다른 요소들에 대한 내부통지 및 상기 통지를 수신하도록 등록한 어플리케이션(302)에 대한 외부통지도 포함한다. 게이트웨이(300)과 어플리케이션(302)의 상호기능은 게이트웨이 파운데이션 클래스(304)를 통해 실행된다.

디스패치(320)는 게이트웨이(300)의 실시간 운영 즉, 전송/수신 대기행렬 처리, 스래드(thread) 초기화 및 관리 등을 위한 것이다. 관리에 수반되는 게이트웨이의 역할은 응용 CAL전송요청, 미가공 흐름데이타 요청, 수신 사건, 및 기타의 가정관리 작업을 포함한다. 디스패처(320)는 드라이버(322),(326)에 의해 실행중인 네트워크 제어 하드웨어에 대한 상세한 다양한 종류의 네트워크 노드에 필요한 지원을 하는 일반 인터페이스를 상기 드라이버(322),(326)에 제공한다.

장치 드라이버(322),(326)는 직접 네트워크 통신 하드웨어(USB 혹은 병렬형 포트 PLX노드, X-10 연결기, CEBus 연결기 등)과 통신하며 데이터링크층(202) 기능 (즉, MAC 헤더의 발생/해석, 저수준 타이밍 등)을 처리한다. 직접 실행중인 하드웨어와 연락하는 드라이버(322),(326)의 포트는 보통 플랫폼형으로서 드라이버와 하드웨어간의 각종 접속망(즉, 공통 시리얼버스(USB), 병렬형 포트, 파이어-와이어 포트, PCI포트, ISA슬롯, 시리얼포트 등)을 지원하기 위한 것이다. 게이트웨이(300)는 연속 X-10 및 CEB노드로부터 수신된 제어데이타를 게이트(300)에 의해 이용되는 포맷으로 전환 처리할 연속요소(328),(330)를 제공한다.

페이로드/프로토콜 프로세서(312) 내의 페이로드/프로토콜 처리기(316)-(318)는 수신된 패킷데이타를 분석 및 처리한다. 데이터 처리기(316)-(318)는 라우팅 처리기(355)과 함께 작동한다. 라우팅 처리기(355)가 원료데이타 패킷헤더를 분석한다. 또한 이 처리기는 다양한 연속프로토콜 스택(352)과 연속흐름 드라이버(362) 간의 데이터흐름의 방향을 조정하는 어드레스 및 라우팅표를 자문한다.

다양한 플랫폼 및 운영장치에 대한 코드이식성을 유지하기 위해, 플랫폼형 코드를 RTOS(211)에 의해 제공된 플랫폼 요약층에서 분리한다. 통상의 실시간 서비스는 스래드 스케줄조정, 메모리관리, 인터럽트 관리, 클록, 동기화, 우선스케쥴 조정, 초기화 등을 포함한다.

게이트웨이(300) 내의 다수 요소는 직접 혹은 간접으로 노드 데이터베이스(308)과 상호작용한다. 데이터베이스의 초기탑재 및 관리 작업은 1차로 디스패처(320), 드라이버(322),(326) 및 페이로드/패킷처리기(312)에 의해처리된다. 사용자 인터페이스 및 관리 어플리케이션은 게이트 파운데이션 클래스(304)를 통해 노드 데이터베이스(308)에 접근하거나 이를 변형시킨다.

노드 데이터베이스(308)는 실시간 접근이 급속히 및 효과적으로 실행되도록 하는 구조이다.

속도/메모리-사용의 절충은 접근속도에 대한 메모리사용 효율을 위한 것이 우선적으로 적용된다. 노드 데이터베이스(308)의 일부를 비삭제 저장하는 경우도 있다.

노드 데이터베이스(308)는 클라이언트 노드 구성 및 노드 상황정보를 포함한다. 이것은 문맥, 목적대상 및 각 클라이언트 노드에 연계된 일시변수(IV)정보를 포함한다. 이 정보는 네트워크명령 및 네트워크요청을 이용하여 노드발견시 얻을 수 있다. 이 정보 중에서 동적IV는 영구저장 형태로 저장된다. 대부분의 대상(노드제어나 문맥제어는 포함되지 않음)에서 이것은 노드상태를 규정하는 현재값IVs을 포함한다.

노드 데이터베이스(308)는 게이트웨이형 정보도 일부포함한다. 각 노드에 대한 노드문맥과 별도로, 노드 데이터베이스(308)는 노드관리를 위해 필요한 대상을 포함한 데이터 서버문맥도 포함한다. 데이터 서버 문맥은 게이트웨이(300)에 의해 확인된 데이터 노드 어드레스 어레이를 갖는 네트워크 노드 리스트 대상을 포함한다.

노드 데이터베이스(308)는 일반적인 접근방법(306)을 통해 접근한다. 대부분의 접근은 네트워크 메시지를 통해 직간접적으로 이루어지기 때문에, 데이터베이스 접근 방법(306)은 일반적인 네트워크 Get 및 Set 방법을 이용하는 변수값 수득 및 설정에 관계한다. 이 방법은 값수득(GetValue), 어레이수득(GetArray), 값설정(SetValue), 어레이설정(SetArray), SetON 및 SetOFF 등을 포함한다. 실행되는 변수의 데이터 형태에 따라 적절한 방법을 선택한다. 예를들어 CAL네트워크는 4가지 IV데이타 형으로 부울값(GetValue, SetValue, SetON, SetOFF), 숫자(GetValue, SetValue), 문자열(GetValue, SetValue) 및 데이터(GetArray, SetValue) 등이 있다.

대부분, 이들 방법은 데이터가 실행되는 유사한 입력키 및 다수의 입력인수를 이용한다. 각 방법은 필요정보(있을 경우) 및 상태로 돌아간다. 예를 들어, CAL IVs에서 실행되는 방법의 일반 프로토형은 다음과 유사하다.

ENUM STATUS 〈method〉(노드ID, 문맥ID, 대상ID, args....*returnVal)

여기서, 노드I, 문맥ID, 대상ID, 및 IVID 등의 핵심키이다. 비-CAL 대상에서는 다른 핵심키 들이 사용된다.

특별한 변수에 대해 설정조작하는 경우, 사건 통지는 상황 변화사건이 일어난 것을 나타내는 사건 처리기(310)를 통해 규칙엔지(314)에 제공된다. 그후 규칙엔진(314)은 변경된 임시변수에 대해 규칙/기록-조건을 해석하고 조치가 적절하면 사건 처리기(310)를 통해 사건을 스케줄화 한다. 이결과, 네트워크가 디스패처(320)을 통한 요청을 수령하거나 혹은 파운데이션 클라스(304)를 통해 더 높은 수준의 어플리케이션(302) 쪽으로 통지한다.

상술한 바와 같이, GET 및 SET에 대해 접근요청이 있을 경우 노드IV, 4개의 정보편(혹은 핵심키)이 제공된다. 이것은 노드ID, 문맥ID, 대상ID 및 IVID을 포함한다.

노드ID는 설정시 기록된 노드 어드레스이다(생산시 혹은 실행과정에서 동적으로 발생할 수 있다). PLX 네트워크에서, 이것은 4바이트 영역이며 0 x 00000001 및 0 x fffffffef 사이의 값을 갖는다. 이 범위보다 큰 값의 어드레스는 방송 혹은 기타 PLX형 사용자를 위해 예정된 것이다. 비-PLX 네트워크의 경우 일부 어드레스 매핑/번역이 필요하다. 라우팅 처리기(355)는 어드레스 번역기록을 수행한다.

문맥ID는 2바이트 길이로서, 바이트가 크면 0xA0 - 0xDE의 값을 갖는 선택형 문맥수이거나 사용할 수 없을 경우는 00이 된다. 바이트가 작으면 0x00 - 0x9E(일반 CAL 사양에서 규정된 것임) 범위의 문맥 클래스 값이다.

목적ID는 1바이트 목적수치값으로 0x01 - 0x3E 범위이다.

IVID는 아스키 문자(문자들)로서 임시변수를 주어진 문맥 혹은 목적형으로 분류하는 것이다.

노드는 문맥 집합체로 파악한다. 각 문맥은 목적집합으로 되며 각 목적집합은 IVs집합으로 된다. 데이터베이스는 상술한 4가지 정보를 담고 IV에 접근하도록 구성되며 다음의 알고리즘을 수반한다.

1) 노드 리스트에 각 노드가 문맥기록 어레이에 표시되는 노드ID 지도를 입력한다(일반적으로 해쉉 알고리즘을 이용함). 문맥ID는 원하는 문맥기록을 수득하기 위한 직접적인 인덱스로 사용할 수 있도록 하기 위해 특별히 규정하지 않으며 직선형태가 된다. 노드는 보통 수개(2, 3개)의 문맥을 갖는다. 각 문맥기록은 대상기록의 어레이(혹은 링크된 리스트)에 대한 표시자를 포함한다

2) 대상ID는 원하는 대상기록을 얻기 위한 직접적인 인덱스로 사용할 수 있도록 하는 포맷으로 되어있다.

3) IVID에 기초하여 실행되도록 IV를 로케이트 하기 위하여 선형검색을 실행한다. 이것은 대부분의 클라이언트 노드에서 노드 데이터베이스(308)에 소수의 IV가 저장된 경우만 효과적이다. 대부분의 GET/SET 요청은 현재의 IV 값에 관계한다. 현재값이 항상 IV리스트에서 제일먼저 저장되는 경우, 선형검색의 연장은 거의 필요치 않다.

네트워크 노드 리스트를 집합화하기 위해 게이트웨이(300)에서 사용하는 발견방법은 노드 종류에 따라 달라진다. 일반적인 CAL 사용할 때의 PLX 노드에 사용되는 알고리즘은 다음과 같다.

1) CAL "Ping" 요청이 네트워크에 전달된다(이것은 초기화 과정 뿐만 아니라 주기적으로 실행된다). 이 요청의 포맷은 CAL 명령/요청의 형태와 유사하며 다음의 변수를 사용한다.

문맥 클래스 = 0 (공통문맥)

대상수 = 1 (노드 제어대상)

방법=50h=PING_REQUEST (일반CAL 사양에서는 정의되지 않음)

IV=〈임의의 IV OK〉

이 요청은 직접 노드에 전송되거나 모든 노드에 방송되는 표준CAL 요청과는 상이하다. Ping 요청은 수신기로부터의 CAL포맷 응답은 요구하지 않는다.

2) CAL Ping요청을 수신하는 노드는 전송노드에 CAL Ping 응답을 반송한다. 이 패킷은 "응답" 이라고 표현했으나, 이 패킷 구문은 전통적인 CAL 응답패킷과 상이하다. Ping요청을 방송 형태로 전송할 수 있도록 이 응답은 달라야 한다. 응답패킷의 포맷(실제로 CAL응답이 아닌 CAL 명령/요청을 수행하는)은 방법코드가 50h 대신 51h인 경우를 제외하고 Ping요청과 유사하다.

3) 게이트웨이(300)가 네트워크 노드 리스트에 없는 노드로부터 Ping 응답을 수신한 경우, 게이트웨이(300)는 이 노드를 리스트에 추가하고 또한 노드 데이터베이스(308)에 노드의 IV정보를 첨가한다.

4) 선택적으로, 게이트웨이(300)은 일정시간동안 듣지 못한 경우 CAL 요청을 노드리스트 상의 임의의 노드에 전송한다. 노드가 응답이 없으면 리스트에서 삭제한다.

5) 또한 클라이언트 노드가 켜지거나 리셋되면 Ping 응답패킷에도 전송한다.

연속제어 노드(X-10 혹은 CEBus 노드 같은)에서, 이 연속사양에 따라 노드 발견과정을 달리할 수 있다. 게이트웨이(300)는 연결기(328),(330)의 조력하에 이들 노드를 관리한다. 연결기(328),(330)는 상술한 CAL노드 발견방법 및 비-PLX 드라이버에서 사용된 연결방법에 따른다. Ping 요청에 대응할 때, 연결기는 연속제어 네트워크 상에서 노드 발견을 실행하는데 필요한 유사한 요청(혹은 요청의 집합)으로 상기 Ping 요청을 전환시킨다. 노드가 발견되면, 연결기(328)(330)는 Ping 응답패킷을 발생하고 게이트웨이(300)의 상부층까지 이것을 통과시킨다. 게이트웨이(300)의 나머지 부분에서, 이들 연속제어 노드는 PLX 노드이다. 연결기(328),(330)은 다음에서 더 상세히 설명한다.

게이트웨이(300)는 사용자 인터페이스 및 관리 어플리케이션(302)이 파운데이션 클래스(304)를 통해 노드 데이터베이스(308)에 접근하도록 해준다. 여기서, 게이트웨이(300)는 응용서버 역할을 한다. 어플리케이션(302)으로 노드 데이터베이스(308)의 접근성을 개선하기 위해 게이트웨이(300)는 복수의 응용서버를 지원하며 여기서 서버중 하나가 실행응용서버이고 나머지는 대기서버 역할을 한다.

각 서버는 동일한 노드 데이터베이스(308)를 저장하며 네트워크 교통 청취, 네트워크 노드 상태변화 관찰 및 노드 데이터베이스(308)의 갱신 등에 따라 정보를 각 노드 데이터베이스(308) 정보에 (동기적으로) 다른 노드와 함께 추가 유지시킨다. 그러나, 실행응용서버는 실제로는 상태변화 관련규칙을 평가하고 필요한 사건 통지를 실행하는 노드에 불과하다. 이 서버가 어떤 이유 때문에 운용불가 상태로 될 경우 대기서버가 이것을 감지하고 "작동하기" 시작한다. 응용서버노드(402)의 실행과정은 도 4에 보는 바와 같이 동기화 블록(404)에서 시작되며 노드(402)는 노드 데이터베이스(308)를 갱신한다. 노드 데이터베이스(308) 갱신후, 이 과정은 결정블록(406)으로 진행한다. 결정블록(406)에서 노드는 실행서버 상태일 경우 이 프로세스는 실행서버블록(408)로 진행하며 그렇지 않을 경우 대기서버블록(420)으로 진행한다. 실행서버블록(408)이 끝나면, 결정블록(412)로 진행한다. 결정블록(412)에서, 서버취침요청이 검출될 경우 지정 대기블록(410)으로 진행한다. 그렇지 않으면, 결정블록(414)으로 진행한다. 결정블록(414)에서, 클라이언트의 요청이 검출되면 다시 응답블럭(416)으로 진행하고 그렇지 않으면 동기화블록(404)로 복귀한다. 지정 대기블록(410)이나 대응블록(416)이 종결되면 이 프로세스는 동기화블록(404)로 다시 복귀한다.

대기서버블록(420)의 종료후 프로세스는 결정블록(422)로 간다. 결정블록(422)에서, 수령하지 않은 클라이언트 요청이 검출되면, 지정된 실행블록(424)으로 진행한다. 그렇지 않으면, 동기화블록(404)으로 복귀한다. 지정된 실행블록(424)의 종료후 다시 통지블록(426)으로 진행하여 여기서 본 발명의 노드가 실행됨을 다른 응용서버형 노드에게 통지한다. 블록(426)의 종료후, 동기화블록(404)로 복귀한다.

규칙엔진(314) 및 사건처리기(310)는 상태변화가 클라이언트 노드에서 발생할 때 실행된다. 이러한 상태변화는 SET 법을 실행할 때 발생하므로 각 SET법은 변화된 IV에 관계하는 규칙엔진을 사건처리기(310)를 통해 통지한다. 규칙엔진(314)는 다시 상기 변화한 IV에 관계된 다른 엔진이 없는지 체크한 후 사건통지가 필요함을 결정한다. 상기의 규칙은 파운데이션 클래스(304)를 이용하여 다양한 노드를 사용자 인터페이스를 통해 서로 연결하면서 사용자가 만들 수 있다. 디폴트 규칙 역시 사용자의 한정범위를 넘지않는 노드 및 일반적인 운용방법에 적용할 수 있다.

규칙은 간단하게 혹은 복잡하게 만들 수 있다. 간단한 규칙은 클라이언트 노드 간에 1-1 또는 1-다수의 결합관계를 형성하는 것이다. 1-1 결합관계의 예를 들면 "조명스위치 A가 켜지면 전구 B에 불이 들어온다" 는 것이 있으며, 1-다수의 결합관계는 "조명스위치 A가 켜지면, 전구A, B, C 및 D에 불이 들어온다" 는 예를 들 수 있다. 복잡한 규칙은 다수-다수의 결합관계 즉, "조명스위치 A가 켜지고 사용자가 보안을 해제하면 전구 A, B, C 및 D에 불이 들어온다" 등을 말한다.

게이트웨이(300)는 공통규칙정의언어(RDL)로 알려진 규칙정의에 필요한 범용성 구문을 제공한다. 도 5는 규칙의 요소 및 어플리케이션(502)과 규칙(504) 간의 상호작용을 보여주는 구조에 관한 것이다. 도 5에서 보는 바와 같이, 어플리케이션(502)은 하나 이상의 규칙 특히 규칙(504)을 포함하는 규칙을 생성한다. 이 규칙(504)은 조건절(508)인 "..만일"이라는 문장을 포함한다. 이 조건절(506)은 하나 이상의 상수(506), 하나 이상의 연산자(512) 및 하나 이상의 IVs를 포함한다. 다시 상기 규칙(504)의 "..그렇다면" 이라는 문구는 어플리케이션(502)로 복귀될 통지(514)를 구체화한 것이다. 각 연산자는 연산자식별(ID)를 포함한다. 각 IV는 ID, 값 및 트리거상태를 각각 하나씩 포함한다. 트리거상태는 엣지 트리거 및 엣지외 트리거를 포함한다. 각 통지는 통지ID 및 메시지ID를 포함한다.

규칙은 사건과 조치를 포함한다. 조치는 응용되는 스트링 어레이이다. 사건은 해당상태가 네트워크상에

나타나는지를 설명하는 부울표시이다. 사건스트링은 하기의 문법에 따라 구체화 된다. 문법은 비제한적으로 정수_상수, 문자_상수, 플로팅_상수 및 스트링으로 쓴다. 각 포맷에 대해 용어는 통일한다. '^' 혹은 '#' 표시뒤의 첫번째 정수_상수는 노드 어드레스를 6진수로 표시한 것이다. 두번째(예외) 정수_상수는 CAL 문맥번호를 6진수로 표시한 것이다. 3번째 정수_상수는 CAL 문맥 클래스를 6진수로 표시한 것이다. 4번째 정수_상수는 CAL 대상번호를 6진수로 표시한 것이다. 한편 문자_상수는 CAL 임시변수(IV)이다. 식별표시 앞의 ^ 기호는 "엣지-트리거" 혹은 "최종 변경" iv를 표시한다. '#' 기호는 iv가 "레벨" 혹은 "기존값" 임을 나타낸다. 하나의 규칙은 1개의 "엣지-트리거 상태" 식별표시만 한다.

규칙문법

OR 논리식 이면 ACTION 논리식으로

OR 논리식

  AND_ 논리식

  OR_ 논리식 ‖ AND_ 논리식

AND 논리식

  동일_ 논리식

  AND_ 논리식 & & 동일_ 논리식

동일_ 논리식

  관계_ 논리식

  동일_ 논리식 = = 관계_논리식

  동일_논리식 ! = 관계_논리식

관계_논리식:

  덧셈법_논리식

  관계_ 논리식 〈 덧셈법_논리식

  관계_논리식 〉 덧셈법_논리식

  관계_논리식 〈 = 덧셈법_논리식

  관계_논리식 〉 = 덧셈법_논리식

덧셈법_논리식

  곱셈법_ 논리식

  덧셈법_논리식 + 곱셈법_논리식

  덧셈법_논리식 - 곱셈법_논리식

곱셈법_ 논리식

  기본_논리식

  곱셈법_논리식 * 기본_논리식

  곱셈법_논리식 | 기본_논리식

  곱셈법_논리식 % 기본_논리식

기본_논리식

  식별자

  상수

  문자열

  (OR_논리식)

식별자 : ^ 정수_상수. 정수_상수.선택-선택정수_상수. 정수_상수. 문자_상수

  # 정수_상수. 정수_상수.선택-선택정수_상수. 정수_상수. 문자_상수

상수:

  정수_상수

  문자_상수

  플로팅_상수

ACTION 논리식

이_규칙에_대해_기록된_어플리케이션을_통지한다

노드에_ PLX_패킷을_전송한다.

연속구문(즉, RDL 이외의 구문)을 이용하는 실행규칙을 포함한 비-PLX 노드에 있어서, 규칙엔진(314) (연결기(328),(330)의 도움을 받아)은 연속구문을 RDL구문으로 전환한다.

디스패처(320)는 게이트웨이(300) 내부에서 일어나는 연산순서와 흐름에 대해 방향성을 제공한다. 처리기 스레드는 (1)수신 페이로드 스레드, (2)미가공데이타(흐름) 전송스레드, (3)CAL 전송스레드, 및 (4) 후순서 유휴 스레드(선택) 등을 순서대로 포함한다.

상기와 같은 처리기 스레드는 클라이언트 노드에 패킷을 보낸다. 장치 드라이버 전송 루틴을 호출하기 전 이들 스레드는 먼저 TxFreeCount 신호기에서 대기(슬립)하며 여기서 이 드라이브가 전송요청을 처리할수 있음을 표시한다. 또한 CAL 전송스레드는 "지연"모드에 있는 클라이언트에서 CAL패킷이 전송되지 않음을 확인시켜 준다. 지연모드를 다음에서 설명한다.

디스패처(320)는 또한 처리기 스레드에 의해 조치를 받는 대기행렬을 관리한다. 처리기 스레드 및 다른 게이트웨이(300) 요소는 디스패치(320)를 호출하여 대기행렬에 패킷을 추가하거나 삭제하도록 한다. 디스패처(320)는 CAL 요청/응답 패킷과 미가공데이타 흐름의 단편들을 설명하는데 디스패치 제어블록(DCBs)를 사용한다.

DCB 구조는 다음과 같이 (C/C + + 구문)으로 정의한다.

```
typedef struct sDCB
{
void         *link
UINT8    *buffer
UNIT32    destDevAddress
UNIT32    srcDevAddress
UNIT16    timeStamp
UNIT16    reservedl
UNIT8     destSocket
UNIT8     srcSocker
UNIT8     sequenceNo
UNIT8     bufferSize
UNIT8     controlInfo
UNIT8  reserved 2[5]
}
tDCB, * PDCB
```

link영역은 DCB 소유자가 필요시 사용할 수 있다. 통상 DCB 대기행렬 관리를 위해 이용된다. buffer 및 bufferSize 영역은 DCB에서 표시한 패킷/단편 데이터를 표시 및 정의한다. destDevAddress 및 srcDevAddress는 DCB와 관련하여 결정 및 소스 노드를 확인한다. DestSocker srcSocket은 또한 노드 속의 결정/소스 어플리케이션을 추후 식별하는데 사용한다. timeStamp 영역은 게이트웨이(300)가 각종 타임아웃 조건을 처리하는데 도움을 얻기 위해 사용한다. sequenceNo는 미가공데이타 패킷흐름에 앞서서 패킷을 요청하기 위해 이용된다. controlinfo 영역은 각종 비트영역 및 특징을 갖거나 DCB 및 관계 데이터패킷의 처리가 요구된 플랙을 저장하는 곳이다. 이들 비트는 데이터 암호화, 확인 및 데이터 흐름 등의 특징을 제어한다. reserved1 및 reserved2 필드는 게이트웨이(300)에 의해 내부이용 및 외부어플리케이션에 의해 이용된다.

수신 페이로드 스레드는 디스패처(320)에 의해 시작되는 가장 큰 우선순위를 갖는 페이로드이며 장치 드라이버(322),(325)가 수신패킷을 대기할 때마다 실행한다. 장치 드라이버(322),(325)의 인터럽트/폴 서비스경로는 이 스레드를 강제중단시킬 수 있다. 이 스레드는 수신 대기행렬로부터 패킷을 삭제하고 이것을 추후 프로세스를 위해 페이로드/프로토콜 처리기(312)로 이동시키는 역할을 한다. 스레드는 수신 대기행렬에 아무것도 남지 않을 때까지 계속 실행되며 그후 다른 수신 대기행렬이 생길 때까지 슬립상태로 된다.

미가공데이타 전송 스레드는 2가지 조건이 참일 때 실행되며, 이 조건은 (1)미가공데이타/흐름 전송요청이 어플리케이션에 의해 대기될 때, 및 (2)우선순위가 큰 스레드/인터럽트 처리기가 실행을 끝낼 때이다. 미가공전송 스레드는 전송 대기행렬의 다음 순서를 수득하여 이것을 적절한 장치 드라이버 전송경로로 보낸다. 전송 대기행렬에 더 이상의 기록이 남지 않을 때까지 혹은 스레드의 일부가 도달할 때까지 이것을 계속 실행한다. 미가공데이타 전송스레드는 그후 새 전송물이 계획될 때까지 슬립상태로 된다.

CAL 전송스레드는 미가공데이타 스레드와 비슷하다. 이들의 큰 차이점은 CAL 전송스레드가 다른 전송 대기행렬을 취급한다는 점이다. CAL 전송스레드는 CAL 요청사건에 대해 연산한다. CAL 요청사건은 통상 어플리케이션(304)에서 나온 요청결과로서 계획된다. 이들 어플리케이션 요청은 디스패처(320)에 의해 CAL

전송 대기행렬에 존재한다. CAL 전송물은 원격노드 혹은 노드 데이터베이스(308)에 저장된 상태정보를 읽기(GET) 및 쓰기(SET)위한 것이다. CAL 전송물은 또한 규칙엔진(314)에 의해 해석된 규칙에서 정의된 사건을 트리거하는 내부 및 외부 상태변화에 의해 생성될 수도 있다.

유휴 스레드는 장치의 저우선순위 상태에서 실행되며 선택적이다. 사용할 때, 유휴 스레드는 고우선순위 스레드가 실행될 때 비제한적으로 안전하게 중지할 수 있는 저우선순위 작업을 취급한다. 저우선순위 작업은 일반적으로, (1)메모리관리 및 불필요정보 수거작업, (2)무대응 노드를 검출 및 노화처리하는 감시/Ping패킷의 전송, 및 (3)캐쉬 외(영속형)저장본과 캐쉬 데이터베이스 정보의 동기화 처리 등을 행한다. 유휴 스레드를 지원할 때, 디스패처(320)는 다른 모듈에 대한 인터페이스를 제공하며, 따라서 유휴스레드가 호출한 저우선순위 작업을 계획할 수 있다.

장치 드라이버(322),(326)는 디스패처(320)로부터 포괄요청을 수신하고(디스패처 제어블럭 DCB 가 표시한) 이들 요청을 해당 네트워크노드에 적합한 I/O 요청으로 전환시킨다.

장치 드라이버(322),(326)는 또한 네트워크 하드웨어에서 수신된 패킷을 처리하고 DCB를 형성하여 이것을 추후 프로세스를 위해 디스패처(320)까지 전달한다. 디스패처 스레드는 장치 드라이버(322),(326)와 인터페이스 하는 게이트웨이의 유일한 요소이다. 장치 드라이버(322),(326)에서 디스패처(320)에 제공한 인터페이스는 DCB를 통과하는 DriverSend 인터페이스를 포함한다. 매체추출요소(MAC) 모듈은 MAC 헤더를 형성하고 네트워크 제어 하드웨어에 대한 정확한 패킷-전송을 개시한다. 드라이버 인터럽트/폴링 경로 대기행렬은 페이로드 수신 대기행렬의 사건을 수신하고 추후 수신 페이로드 스레드에 의해 이것이 처리된다.

하기의 설명은 장치 드라이버가 PLX형 노드인 경우를 가정한 것이다. 그러나, 상부층에 제공된 인터페이스는 다른 드라이버를 게이트웨이가 지원할 수 있는 일반적인 것이다. 연결기(328),(330)은 X-10 및 자연 CEBus 노드 같은 PLX형 노드가 경계없이 게이트웨이(300)의 지원을 받을 수 있도록 해준다.

드라이버, 예컨대 PLX 제어 드라이버(323)을 로드할 때, 드라이버 초기화 모듈은 드라이버에 의해 구동되는 네트워크 인터페이스 하드웨어를 작동상태로 만든다. 초기화가 종결되면, 드라이버(323)와 하드웨어는 패킷의 전송/수신 준비상태이다. 드라이버 초기화의 일부로서 수신-처리기 ISR/폴 경로의 설정, 하드웨어 자원의 결정/유지 및 기타 가정용 스레드의 작동개시 등도 포함된다. 통상 타이머 스레드는 전송종결 혹은 정지상태의 장치 드라이버/하드웨어 문제를 체크하기 위한 것이다.

전송시, 디스패처(320)는 DCB를 이용하여 드라이버(323)에 피킷/단편 데이터 어드레스, 제어정보를 공급한다. 드라이버(323)는 적절한 MAC헤더를 생성 및 전송을 개시한다. 동시처리 가능한 전송물의 수는 초기화 과정에서 읽을 수 있는 하드웨어/펌웨어 의존값 (여기서는 TxFreeCount 라고 함)이다.

전송데이타를 네트워크 제어기에 복사한 후, DriverSend 경로는 전송 타임아웃 타이머를 개시한다. 이때, 응답개시 전송종결을 동기식으로 수신하는 경우 (DriverSend 경로가 복귀되지 전의 전송에 따라)를 제외하고, 네트워크 하드웨어가 전송종결상태를 가리키기 전까지 전송은 "계속" 상태가 된다. 전송이 계속 상태이면, TxFreeCount 로 한정한, 네트워크 제어기가 버퍼영역을 활용할 수 있는 한도내에서 계속 송신이 진행된다. 진행중인 송신수가 TxFreeCount와 동일해지면 DriverSend 경로가 로크되고(더이상 송신이 일어나지 않는 것을 의미), ISR/폴-경로가 전송 완료상태 혹은 전송 타임아웃상태의 수신에 따라 상기 로크를 해제할 때까지 계속 로크상태를 유지한다.

드라이버 인터럽트/폴 서비스 경로(혹은 ISR 처리기)는 장치에서 가장 먼저인 우선순위를 가지며(문맥의 인터럽트), 수신된 패킷의 MAC 헤더를 해석/스트립핑 및 적절한 페이로드 처리기에 의해 추후 처리되도록 디스패치(320)로 패킷을 대기행렬화 시킨다. 수신패킷은 다음 중 하나의 카테고리에 들어간다. (1)내부(혹은 지역)제어/상황패킷, (2)CAL 명령/요청, (3)CAL 즉시응답, (4)CAL 지연응답 혹은 (5)미가공흐름 데이터.

내부제어/상태 패킷은 기본적으로 전송완료 혹은 오류상태를 표시한다. ISR 처리기 내의 전송당태 패킷의 처리는 이 상태가 응용되는 전송패킷 형태(즉, CAL/미가공, 지역/원격)에 따라 다소 변할 수 있다.

미가공 단편 및 CAL패킷에서 전송이 성공적으로 완료된 상태라면, ISR 경로는 TxFreeCount를 조정하여 다음 송신을 가능하게 만든다. 타임아웃 오류시에는,송신이 성공할 때 혹은 재시도 최대가능횟수에 도달할 때까지 CAL 전송이 수차 재시도된다.

미가공 데이터 단편의 전송 타임아웃은 CAL 전송시와 비슷한 방식이지만, 특별한 경우(미가공 데이터의 흐름 성격 및 수신측에서 예상되는 조건 등에 따라) 별도의 재시도처리가 필요할 수도 있다. 미가공 흐름 단편은 데이터 흐름을 조정하는데 도움이 될 패킷시퀀스법을 이용한다. 미가공흐름 수신 어플리케이션의 조건에 따라서, 타임아웃처리 코드는 타임아웃의 전송 뿐만 아니라 진행중인 전송이 성공적인지의 여부에 관계없이 다른 진행중인 미가공 전송도 재시도한다(시퀀스 수가 큰 경우). 이것은 수신 어플리케이션 혹은 라우팅 처리기가 시퀀스에 선행하기 위해 패킷을 수신할 수 있도록 해준다. 수신 어플리케이션/라우터는 이것을 전달하여 비정상 시퀀스를 처리하는데 이용된다.

CAL 요청/응답 및 미가공 흐름 패킷은 ISR 처리기에 의해 대기행렬화 하며 추후 수신 페이로드 스레드 및 적절한 페이로드/프로토콜 처리기에 의해 추후 처리된다.

특수처리기는, CAL 패킷과 함께, CAL 응답이 진행중인 원격노드에 새로운 CAL요청이 송신된 바가 없음을 확인시켜 준다.

연속 제어네트워크 장치드라이버(X-10드라이버(328) 및 CEBus 드라이버(332)에 있어서, 연결기(323),(330)은 게이트웨이(300)와 함께 비-PLX 드라이버가 운영되도록 하는 인터페이스를 제공한다. 전송시, 연결기(328),(330)는 디스패처(320)이 송신한 CAL 패킷을 검사하고 이 패킷을, 실행중인 연속 드라이브/장치에 의해 인식되는 등가의 포맷으로 전환한다. 수신시, 연결기(328),(330)는 수신된 연

속데이타를 CAL포맷으로 전환하고 DCB를 구축하고 DCB를 디스패처(320)까지 전달한다.

송신 및 수신과정에서, 연결기(328)(330)는 4-바이트 DCB 어드레스를 실행중인 연속드라이버의 어드레스 포맷으로 전환하여 어드레스 번역을 실행한다.

수신패킷은 CAL 제어메시지 혹은 미가공 흐름데이타(프린터 데이터, 터널연결된 이더넷/ADSL 데이터 등을 포함)를 함유할 수 있다. 이들 페이로드는 각각 적절한 페이로드/프로토콜 프로세서(312)에 의해 제공된 적절한 페이로드/프로토콜 처리기에 의해 처리된다. CAL 처리기(318)는 제어네트워크 노드에서 수신한 CAL 요청/응답 제어패킷을 분석 및 처리하는 역할을 하는 CAL 해석기를 포함한다. 페이로드 처리기 스레드는 추후처리를 위한 CAL 처리기에 클라이언트의 CAL 패킷을 전달한다. CAL 해석기는 CAL 메시지를 다룰 다른 게이트웨이 요소에 기본 CAL 분석경로를 제공한다.

디스패처(320)가 CAL 처리기(318)를 호출하면, 표시기를 통과하여 CAL 메시지, 패킷의 기원이 되는 클라이언트의 어드레스, 또한 실행될 클라이언트의 어드레스를 거친다. 이 두 어드레스는 패킷이 어플리케이션(302)에서 기원하는 경우를 제외하고 서로 동일하다.

CAL 메시지는 명령(통상 요청으로 해석됨) 혹은 응답 메시지로 구분된다. 클라이언트 노드의 CAL 명령은 통상 게이트웨이(300)에게 "현 상태변수를 새 값으로 변경"하는 것의 영향을 알려주는 상태변경 통지이다. 어플리케이션(304)에 의해 제어되는 등 특별한 기능의 클라이언트는 또한 사용자의 요청에 따라서, 다른 클라이언트의 상태(임시)변수를 Get 혹은 Set 하도록 CAL명령을 송신한다.

단일명령 CAL 패킷은 다음과 같다.

〈구문ID〉 〈대상#〉 〈방법/마크로ID〉[〈IV〉[〈인수〉]]

CAL 해석기는 메시지를 성분으로 분할하고, 방법식별기를 적절한 데이터베이스 접근방법으로 매핑시키며 이것을 ID 및 인수변수로 호출한다. 데이터베이스 접근방법(306)은 요청연산을 실행하고, IV가 변경되면 규칙엔진(314)에 통지한다. 규칙엔진(314)은 변경된 IV에 응용되는 규칙을 평가하고 확실하면 사건처리기(310)를 통해 사건통지/조처를 스케줄화 한다.

CAL 응답패킷은 클라이언트 노드가 CAL명령에 응답하기 위해 이용된다. CAL 응답은 〈상태 토큰〉[〈복귀데이타〉] 형싱이며 〈상태 토큰〉 이란[종결, 취소, 오류]의 응답 형태를 나타내는 1바이트 표시기이고 〈복귀〉 데이타는 명령메시지 결과로 복귀되는 데이터를 말한다.

이들 패킷 중 하나를 수신하면 최소 CAL 명령/요청에 연계된다. 이 연계는 클라이언트가 요청에 즉시 응답할 때 간단하다. 응답이 지연되면 이 연계가 다소 복잡해진다. 연계를 가능하게 하는 것은 사전요청에 대한 응답을 송신했을 때까지 클라이언트가 네트워크에 CAL 요청패킷을 송신하지 않을 수 있다는 점이다. 게이트웨이(300)는 "지연"모드에 있는 클라이언트에게는 요청을 송신하지 않는다. 따라서, 게이트웨이(300)는 각 클라이언트마다 이들에게 송신된 최후 요청(혹은 명령)을 저장할 수 있다. 클라이언트 응답을 수신하면, 이 정보는 CAL 처리기(318)로 하여금 복귀된 정보로 무엇을 할 것인지 결정하도록 해준다.

미가공 데이터 처리기(316)는 미-CAL 수신데이타를 처리한다. 게이트웨이(300)는 다수의 어플리케이션에서 기원하는 복수의 순행 데이터 흐름을 지원할 수 있다. 미가공 데이터 처리기(316)가 미가공 데이터 흐름을 미분하는 방식은, 1개의 단일소켓이 각 미가공 데이터 흐름과 연계되는 경우, 소켓영역을 수반한다. 미가공 데이터 흐름에는 이더넷/ADSL 네트워크패킷, 프린터 데이터, 음성흐름 등을 포함한다. 미가공 데이터 처리기(316)는 소켓번호를 검사하는 것 이외는 미가공 데이터의 분석을 거의 하지 않는다. 패킷 데이터 및 연계된 DCB는 그후 상기 소켓번호에 대응되는 라우팅 처리기(355)에 통과된다.

라우팅 처리기(355)는 게이트웨이(300)가 연속데이타 네트워크(이더넷, ADSL, 토큰링 등)으로부터 원하는 네트워크로 네트워크 데이터교통의 방향을 재지정할 수 있게 해준다. 예를들어, 연속 네트워크로부터 PLX 전원 네트워크 같은 네트워크로 네트워크 데이터교통의 방향을 재지정하는데 게이트웨이(300)을 사용하면 다음에 제한없이 다수의 장점을 갖는다. (1)기존의 이더넷 네트워크를 연장하여 별도의 이더넷 케이블 없이 전원 노드를 포함할수 있다. (2)광역대 데이터를 전원을 통해 배급할 수 있다. (3)전원 네트워크 클라이언트에 대해 프록시서버를 활성화할 수 있다. (4)연속프로토콜 스택(TCP/IP, PX/IPX, NewBEUI 등)을 전원을 통해 터널링 연결할 수 있다.

라우팅 처리기(355)에서 실행하는 중요한 작업은 어드레스 해석이다. 초기화 과정에서, 라우팅 처리기(355)는 게이트웨이(300)로부터 기원하는 전송 DCB 내에 srcDevAddress로 사용되는 4-바이트 어드레스를 얻는다. 라우팅 처리기(355)는 이 어드레스를 연속스택 및/혹은 처리기와 소통할 드라이버에 적합한 형태로 변역한다.

다음에서는 라우팅 처리기(355)가 연속네트워크 데이터 패킷을 다른 프로토콜에서(즉, PLX 프로토콜) DCB를 사용하여 터널화하는 방법을 설명한다. 터널화는 제1 프로토콜의 패킷을 제2 프로토콜의 패킷속에 가두거나 이것을 랩화하는 것이다. 최종상태에 도달하면, 랩을 벗기고 제1 프로토콜로부터 최초의 패킷을 노출시킨다.

초기화 과정에서, 라우팅 처리기(355)는 위에서 설명한 바와 같이 어드레스 매칭/번역표를 셋업한다. 라우팅 처리기(355)는 또한 DCB 당 지원되는 최대 패킷/단편 크기도 표시한다. 라우팅 처리기(355)는 이 정보를 디스패처(320)로부터 수득한다. 다양한 라우팅 처리기(355)는 소켓번호롤 구별한다. 공지된 소켓 어드레스를 초기화 과정에서 단독으로 사용하거나 동적으로 취득할 수 있다.

라우터 처리기(355)가 송신요청을 연속스택으로부터 혹은 전원 장치에 연결된 드라이버로부터 수득하면, 이 처리기는 송신데이타를 단편으로 분할하여 지원가능한 최대 DCB데이타 크기를 넘지않도록 만든다. DCB가 그후 생성되고 시퀀스번호를 각 단편에 부여한다. 제1단편은 항상 제0 시퀀스이며 마지막 단편은 지정된 고비트 시퀀스를 갖는다. 선택적으로, 라우팅 처리기(355)는 이것의 기능성이 다른 레벨에 적절

치 않은 경우 패킷에 검사합을 추가할 수 있다. 이 DCB는 그후미가공 송신 대기행렬 상에 대기하도록 디스패처(320)에 전달 된다. 이것은 미가공 송신 스레드를 자극하여 DCB를 전원 상에 전송하기에 적절한 장치드라이버에 전달한다.

장치드라이버에 의해 각 전송이 완료되면 Tx 상태가 DCB에 표시되고 DCB는 라우팅 처리기(355)에 복귀된다. DCB가 모두 복귀되면, 처리기(355)는 연속프로토콜에 의해 요청되는 송신완료 연산을 실행한다. 타임아웃/재시도 문제는 디스패처(320), 장치드라이버 및 기타 다수 경우는 연속스택이 해결하므로, 처리기(355)는 별도의 재시도연산을 실행할 필요가 없다.

수신처리는 라우팅 처리기(355)가 다수의 송신기로부터 나온 송신단편을 함께 수신하므로 약간더 복잡하다. 또한 경우에 따라 송신단편이 잘못 수신되거나 누락되는 경우도 있다. 라우팅 처리기(355)는 이러한 모든 가능성을 처리할 수 있는 수신모듈을 갖는다. 일반적으로 시작 DCB/단편이 페이로드/프로토콜 처리기(316)에 의해 하나의 처리기(355)에 전달되면 DCB/단편은 송신 어드레스를 위해 정해진 수신 대기행렬 상에 위치한다. 각 DCB를 수신하면 수신기는 최종 단편 시퀀스 혹은 수신 타임아웃을 체크한다.

모든 단편이 수신 및 적분검사를 통과하면 라우팅 처리기(355)는 수신사건을 표시하기 위해 연속프로토콜/드라이버에 요구되는 단계들을 실행한다. 이것은 보통 패킷을 재배열하고 관계있는 수신데이타를 연속모듈(352)의 버퍼에 복사할 수도 있다. 각 수신 DCB가 처리되면 DCB는 자유DCB 목록에 복귀된다.

게이트웨이 파운데이션 클래스(304)는 대상지향 개념에 기본한다(즉, 자바, C++, 스몰토크 등). 또한 노드 데이터베이스(306), 규칙엔진(314), 사건처리기(310), 기타 게이트웨이(300)로부터 제공되는 서비스에 저장된 노드정보에 접근 및 관리할수 있는 다양한 어플리케이션 방법을 제공한다. 최종사용자 어플리케이션(302)는 광범위한 장점을 사용자에게 제공할 수 있다. 예를들어, 게이트웨이 파운데이션 클래스(304)는 단독응용 및 자바브라우저 애플릿이 노드 데이터베이스(308)에 표시된 노드를 열거, 관측 및 제어할 수 있게 해준다. 이 어플리케이션/애플릿은 규칙정의언어를 이용하여 규칙을 간단히 혹은 복잡하게 정의하여 노드간의 다양한 결합을 한정할 수 있다.

상술한 내용은 본 발명의 특별한 예에 한정한 것이나 당해분야의 기술자라면 다음의 부록A에 따른 본 발명의 범위와 사상을 벗어나지 않는 범위에서 다양한 변화와 수정이 가능함을 이해할 것이다.

부록A

PLX 프로토콜은 복수의 스마트 및 단순 노드가 통상의 데이터/제어채널을 통해 통신할 수 있도록 해주는 저렴하고 사용하기 간편하며, 융통성, 신뢰성 및 다양한 크기변화성의 아키텍춰/프로토콜이다. 네트워크형 프로토콜은 네트워크상의 어떤 노드가 자신을 실행네트워크 서버로 변하게 해준다. 실행네트워크 서버는 라인업 카드에 기초하여 클라이언트 노드를 폴(poll) 시킨다. 이 아키텍춰는 실제데이타의 전송을 위한 대역폭을 유지하면서 충돌은 감소시킨다. 이 프로토콜은 제어 및 데이터 네트워킹요구를 모두 지원한다. 데이터 혹은 동기형 데이터의 흐름지원은 타임슬롯을 네트워크에 할당하고 2개의 지능노드가 실행네트워크 서버의 중재를 받아 직접 서로 소통할 수 있도록 하면 가능하다. 또한 실행네트워크 서버는 대량의 데이터교통이 주네트워크의 운영과 무관하게 유동할 수 있도록 각 데이터채널을 할당할 수도 있다. 네트워크 노드는 실행네트워크 서버가 동적베이스에서 바뀔 수 있게 동작하며 슬립상테의 네트워크에 전송요청을 개시하는 제1 노드에 의해 결정된다. 클라이언트 노드는 어드레스 분리식을 이용하여 동적폴링처리를 통해 어드레스된다.

PLX 프로토콜을 포함하는 아키텍춰는 네트워크매체로서 빌딩내 기존의 전원을 사용하는 네트워크에 특히 적합하다. 기존의 전원을 사용하여 데이터를 전송하므로 사용자가 네트워크용 케이블을 필요로하지 않는다.

PLX 프로토콜은 네트워크 노드에 로버스트, 결정력 및 매체접근성 등을 제공한다. 노드는 어드레스 분리식을 이용한 동적 폴링에 의해 어드레스된다. 가시형 데이터채널은 진단, 인수통과 및 일반적인 데이터통과 어플리케이션에 사용할 수 있다.

한 예로, PLX 프로토콜은 세계적으로 동일한 식별코드, 노그프로파일, 및 32-비트 가상 어드레스력을 제공한다. 이 때문에 플러그-n-플레이형 네트워크와도 호환가능하다.

또다른 예로, PLX 아키텍춰는 피어링(peering), 복수서버, 단순한 구조, 기밀성, 데이터그램 검출력, 복수데이타 포맷, 및 우선순위 선정식 등의 장점을 갖는다. CRC 및 검산합같은 오류검출, 데이터 적분력 등도 PLX의 다른 측면 중 일부이다. PLX 아키텍춰는 스마트 노드 및 단순 노드에 설치하여 간단한 제어부터 복잡한 데이터 흐름까지 광범위한 데이터 트랜잭션을 제공한다.

그밖의 예로서, PLX는 상태장치 논리 혹은 마이크로 컨트롤러에도 사용할 수 있다. 스트림라인 저-말단 노드(단순 노드)는 PLX 전체기능의 서브셋을 사용할 수 있도록 실행할 수 있다. 중감범위의 노드 즉, 인터콤/감시장치, 프린터, 마우스 및 기타 데이터 중앙집중노드 역시 PLX 아키텍춰 내에서 응용가능성을 발견할 수 있는 응용분야다.

PLX 프로토콜은 데이터링크층, 네트워크층 및 전송층의 운용규칙을 정의한다. 한 예에서, PLX는 데이터링크층의 매체접근제어(MAC) 부분을 포함한다. MAC 프로토콜은 물리적매질이 각 노드에 접근하는 방법 및 시기를 통제하는 규칙의 집합이다. 한 예에서 MAC 프로토콜은 전력선과의 충돌을 감소시킬 동적 중앙집중분포형 토큰통과 아키텍춰를 사용한다.

PLX 아키텍춰는 네트워크상의 노드가 자신을 실행네트워크 서버로 전환시켜 토큰에 대한 임의요청에 대응할 수 있도록 해준다. 노드가 작동하지 않으면 이들은 "슬립(sleep)" 모드로 들어가고 기타 불필요한 "폴링"교통이 사라진다. 이 아키텍춰는 실제데이타의 전송을 위한 정확한 대역폭을 유지하면서 충돌을 피할 수 있다.

PLX 아키텍춰는 대부분, 제어 및 데이터 네트워크화 목적을 모두 만족시키는 패킷을 지원할 클라이언트/

서버 네트워크화 아키텍춰이다. 데이터 또는 동기화데이타의 흐름성은 배선에 타임슬롯을 할당하고 또한 2개의 지능노드가 실행네트워크 서버에 의해 할당된 경우 직접 서로 소통하도록 해주면 지원할 수 있다. 실행네트워크 서버는 또한 대량의 데이터교통이 주네트워크의 운용과 관계없이 유동할 수 있도록 각 데이터채널을 할당할 수 있다. 실행네트워크 서버 역할을 하는 네트워크 노드는 동적기초에서 변경될 수 있고 대체로 슬립상태의 네트워크에 전송요청을 개시하는 제1 노드에 의해 결정된다. 이와 별도로, 실행네트워크 서버는 응용서버와 무관하게 선택한다. 응용서버는 고정노드 위치를 갖는다. 실행네트워크 서버는 임의의 서버형 노드일 수 있다.

한 구체예에서, PLX는 비실행(슬리핑) 네트워크 매체 상에 초기접급을 위한 데이터그램 감지 알고리즘과 또한 실행 네트워크에 삽입할 목적을 통과하는 중앙집중식 토큰을 포함하는 복합 매체접근기능을 제공한다. 이것은 비충돌성, 토큰통과 환경에서 다중접속하는데 효과적이고 결정론 측면에서도 유리하다. 한 예에서, PLX SMS 초기 매체접근성을 결정하기위한 데이터그램의 존재하에 사용한다.

한 예에서, PLX는 장치의 실행노드에 토큰을 통과시키기만 하는 중앙집중형 동적폴링 알고리즘을 사용하여 네트워크 상의 교통을 감소시킨다. 노드가 비실행이 되면, 이것은 폴링리스트에서 삭제된다. 선택형 폴링 프로세스는 이것을 "버스상의 방출"로 알려진 과정을 통해 폴링목록에 추가되는 노드의 기능력에 근거한다.

방출프로세스는 실시간 실행중에 폴링목록에 추가사키는 기능이다. 방출프로세스는 다중노드 응답이 단일장치 응답처럼 보이게 해준다. 이 장치응답 때문에 서버노드(폴링중인 노드)는 폴링목록에 추가되어야 할 특정 노드를 분리할 수 있다.

폴링목록으로부터 실시간 실행중의 삭제처리는 노화프로세스에 의해 실행된다. 비실행노드는 예정된 기간이 지난후 토큰을 사용할 수 없을 경우 그때그때 폴링목록에서 제거(삭제)된다. 한 예에서, 노화 프로세스는 노드가 토큰요청에 응답할 수 없을 경우 실행된다.

한 예에서, 폴링목록은 매체의 대역폭 용량에 기초하여 일정크기(노드수)를 설정한다. 저우선순위 데이터(조명장치의 제어데이타 등)를 갖는 노드는 고우선순위 데이터(흐름형 오디오/비디오 데이터 등)의 노드를 위한 목록을 담은 공간을 만들기 위해 폴링목록에서 삭제한다.

또다른 예에서, PLX 아키텍춰의 매체접근제어(MAC)층은 스페어 버퍼 및 BUSY 응답 핸드셰이크를이용하여 자체스로틀 메카니즘을 제공한다. 자체 스로틀은 MAC헤더 및 각 노드에 MAC헤더의 복사본을 수용하기에 충분한 수신영역을 공급하여 달성할 수 있다. 노드가 앞서의 패킷요청에 완전히 다 사용된 경우라도, 사용된 노드 역시 요청에 응답할 수 있는 것은 BUSY 응답 때문이다. BUSY 응답은 전송노드에게 패킷 버스트 혹은 시퀀스에 사용되고 있음을 알리고 각 수신노드의 용량에 따라 시스템에 공간을 제공한다.

파워업일 때 노드 자동알림 특징은 원격 데이터베이스 서버의 재동기화 기능을 제공할 수 있다. 새 노드의 파워업일 때, 새 노드는 매체에 새로 도달한 존재를 알려준다.

어떤 예에서, PLX는 바람직한 서버의 선택 및 킥-스타트 알고리즘을 제공한다. PLX 가 클라이언트/서버형 아키텍춰이므로 단일노드는 임의의 매체접근을 위해 선택하는 것이 일반적이다. 일반적인 전원 네트워크에서, 모든 노드는 반드시 동일하게 생성되는 것이 아니다. 따라서, PLX의 한 예의 경우, 바람직한 "실행네트워크 서버" 역할을 하도록 가장 중앙에 위치한(즉, 브레이커 패널에 가까운) 노드를 선택할 수 있다. 바람직한 서버가 비실행인 경우, 원격노드가 바람직한 서버로 실행할 수 있다. 간단한 웨이크업 알고리즘으로 비실행 상태의 바람직한 서버를 다시 실행시킬 수 있다.

처음에는, 클라이언트/서버 모델에 매체를 접근시키기 위해 노드는 토큰을 취득한다. 클라이언트 노드에 토큰이 제공되면 일정한 시간동안 매체를 실행시킨다. 이 시간동안 서버의 수반여부와 관계없이 장치의 노드와 직접 통신할 수 있다. 이 기간이 끝나면 매체접근제어는 서버노드로 철수한다. 따라서 매체임의성은 클라이언트/서버 형식으로 처음 수행되며 그 뒤 집합-집합 시간슬롯으로 이어진다.

한 예에서, PLX는 동적매체 임의서버를 포함한다. 메체에 대한 접근을 임의화하는 서버는 활성도에 기초하여 동적으로 부여된다. 동적부여권은 전송 패킷을 갖춘 제1 노드가 장치의 "비실행" 상태를 인식하고 바람직한 서버(존재하는 경우)에 대해 수차례의 웨이크업 절차를 시도한 후에 실행네트워크 서버의 역할을 예상한다. PLX 네트워크 상에 서버형 노드를 설치하면 실행네트워크 서버가 될 수 있다.

한 예에서, 본 네트워크 프로토콜은 전력선 매체에 흐름형 데이터를 송신 및 수신한다. 또다른 예에서, 흐름형 데이터는 디지탈 음성데이타를 포함할 수 있다. 또는 디지탈 영상데이타를 포함하기도 한다.

또다른 예에서, 네트워크 프로토콜은 전원 매체에 디지탈 PBX형 기능 및/또는 디지탈 인터콤 기능을 제공하는데 사용한다. 네트워크 프로토콜을 사용하여 광대역 디지탈 네트워크 서비스(DSL, 케이블, ISDN 등)를 가정에 있는 기존의 전원을 통해 모든 가정으로 연장시킬 수 있다.

네트워크 프로토콜은 3개 이상의 네트워크교통 즉, 제어교통, 데이터교통 및 흐름형 데이터교통(흐름형 멀티미디어 데이터)을 동시에 처리 및 관리할 수 있다. 네트워크 프로토콜은 해당 노드의 네트워크 요구(음성장치의 결정론 요구 등)에 의존하여 확실한 접근시간을 달성할 수 있는 우선순위 선정식을 제공한다.

PLX OSI모델

도 2에서 도시한 상위 5개 OSI층 (203)-(207) 각각은 네트워크 어플리케이션에 중요한 오버헤드를 추가한다. 도 3에서 도시한 바와 같이, PLX는 비교적 얇은 어플리케이션층(607) 소위 공용언어(CAL) 및 비교적 얇은 전송/네트워크층(603)을 사용하여 하부 데이터링크층(602) 및 물리층(601)을 보충한다. 각 층(601)-(603) 및 (607)은 통상 PLX 보충노드에 존재한다. 도 3에서 보는 바와 같이, PLX 데이터 네트워크 노드(스마트 노드)는 어플리케이션층(207), 네트워크층(203) 및 전송층(204)의 일반적인 OSI 네트워크기능(즉, TCP/IP, IPX, 윈도우, 넷웨어 등)을 포함한다. PLX 보충노드는 소량의 제어정보를 함유하고

이 정보는 층(601)-(603) 및 (607)에서 구현된 바와 같이 PLX스택만 사용하는 PLX 노드 사이를 통과한다.

PLX 물리적 층

PLX 물리적 층(601)은 네트워크 하드웨어, 네트워크 케이블과 물리적 인터페이스하는 하드웨어 세부를 관리하며 보통 실제의 하드웨어 자체를 포함한다. 물리적층은 모듈화 기술, 사용주파수, 출력 등의 애트리뷰트를 포함한다. 한 예에서, PLX는 하기와 같은 디지탈 전원(DPL)을 사용한다.

PLX 데이터링크층

PLX 데이터링크층(602)은 매체(100)와 인터페이스하는 세부, 즉 어드레싱 용량, 매체 임의중재식, 상호간격, 백오프 알고리즘 등을 관할한다. 데이터링크층 (602)는 일반적으로 자원/수신처 어드레스, 길이 및 오류검출/보상 데이터 즉, 사이클 중복검사(CRC) 혹은 검산합 데이터 등을 함유한 헤더를 포함한다.

PLX 네트워크층

PLX 네트워크/전송층(603)은 인터넷층 이라고도 하며, 네트워크 내부의 한 장소에서 다른 곳으로 데이터 패킷을 라우팅 하는 역할을 한다. PLX내에서, 네트워크층(603)은 장치, 각가의 노드, 소켓 및 MAC 헤더 영역 내부의 네트워크 어드레스영역을 이용하여 관리한다.

PLX 전송층

PLX 네트워크/전송층(603)은 이것 위에 있는 어플리케리션층(607)을 위한 2개의 호스트 사이에서 데이터 유동을 제공한다. 전송층(603)은 또한 시퀀스번호 및/또는 요청/응답 형태 수령확인정보를 담고 있다. PLX 내에서 전송층(603)은 OSI 전송층(203)과 비교할 때 더 작은 눈금 및 흐름화 되어있어 어플리케이션을 제어할 수 있다. 전송층(603)은 요청/응답 핸드셰이킹 알고리즘, 재시도 알고리즘, 타임아웃 알고리즘 등을 제공한다. PLX는 네트워크/전송층(603) 거의 전체를 MAC헤더의 제어영역 내에 제공한다.

PLX 어플리케이션층

PLX 어플리케이션(607)은 어플리케이션의 세부를 관할하며, 전송 정도에 따라 핸드셰이킹 프로토콜 및/또는 요청/응답 프로토콜을 사용하여 패킷을 전달한다. 중복영역의 상당부분이 OSI층의 프로토콜 속에 존재한다. 이 중복은 더 많은 중복으로 전환되고 더많은 공간을 사용하며 추가의 처리력을 필요한다. PLX 프로토콜에서, OSI 영역은 많이 필요치 않으며 따라서 누락된다.

각종 OSI 프로토콜에 포함된 다양한 요소의 검사에서 데이터링크층(602)은 상부 3개층 없이 대부분의 필터링처리를 행할 수 있음을 알았다. 이 필터링은 데이터링크층(602)이, 동일한 통신채널을 두고 경쟁하는 다수의 노드(즉, 동일한 네트워크선을 두고 경쟁하는 다중 네트워크 카드)와 같이, 하드웨어 문제를 고려한 하드웨어 논리에 제한되지 않기 때문에 유리한 점이다. 한 예에서, 특정 네트워크 노드를 위한 네트워크 하드웨어는 특정 네트워크 노드에 대해 수신되는 데이터 패킷을 제외한 모든 것을 필터한다.

DPL을 위한 2개의 프로토콜

2개의 프로토콜을 디지탈선(DPL)에 사용할 목적으로 PLX이 바람직하게 정의한다. 하위 프로토콜 및 상위 프로토콜.

하위 프로토콜 정의. 하위 프로토콜은 데이터링크층(602)의 정의 및 비교적 적은 네트워크 및 전송기능을 갖는 매체(100)으로부터의 고패킷 필터링, 송신 및 수신방식을 제공한다.

상위 프로토콜 정의. PLX 노드는 소량의 제어정보를 담고 있다. 각 PLX노드 특별한 노드 애트리뷰트를 제어하기 위해 공용층(607)을 이용한다. 이것은 PLX장치를 노드종류에 관계없이 특징화할 수 있게 해준다. 어플리케이션층(607)은 하드웨어 헤더를 벗겨낸 뒤 제어정보를 판독하거나 분석한다.

물리적 층: 디지탈선(DPL) 사양

PLX 프로토콜은 광전송, 광섬유전송, 무선주파수 전송방식, 트위스트쌍 전송방식, 공축전송방식, 위성장치, 디지탈선(DPL)장치 등의 다양한 종류의 네트워크 매체(데이타전송방식)와 함께 사용할 수 있는 범용 프로토콜이다.

DPL방식는 또한 전원중첩 전송방식(즉, 건물에 사용되는 표준 110볼트 교류(VAC) 전원)이라도고 하며 디지탈데이타를 전송하는 것이다. 한 예에서, PLX 프로토콜은 단일 저속채널(350-1000 kbps), 약 5.6MHz의 저속 중첩주파수, 약 80dB 동적범위이상, 저대역폭 용도 (속도에 따라 다르나 약 1MHz 정도) 등을 갖는 DPL 과 접속하여 사용한다.

또다른 예에서, PLX 프로토콜은 다중고속채널(총 4-8mbps), 최고 30MHz 이상의 고속 중첩주파수 및 약 80dB 동적범위 이상을 갖는 DPL에 관계하여 사용된다.

통상의 DPL 장치에서, 전송중첩은 데이터 전의 적어도 20마이크로초 동안 실행되며 또한 전송기의 실행불능 시간은, 수신기가 중첩을 검출할 수 없을 때까지, 15마이크로초 이상이 될 수 있다.

하위 프로토콜층: PLX 사양

PLX 프로토콜은 간단한 제어에서 복잡한 데이터 흐름형 네트워크까지 어플리케이션에 따라 크기조정할 수 있다. 한 예에서, PLX 프로토콜은 일반 공용언어(CAL) 사양의 장점 대부분을 이용할 수 있도록 개조된다. CEBus는 EIA-600으로 정의하며 버스장치의 제어를 위한 공업표준 제어언어이다. EIA-600는 가정LAN 내에서 사용할 공용언어를 위한 기본구조를 제공한다. 일반CAL은 EIA-721 시리즈로 정의하며 이 시리즈는 EIA-721.1, EIA-721.2, EIA-721.3 및 EIA-721.4를 포함한다. CEBus 공업협회(CIC)는 언어를 이용할 "문법적" 규칙을 정의하여 상기 기본구조에 부가되는 가정 플러그&플레이(HPP) 사양을 선정했다.

HPP 사양은 가정의 상태에 관하여 조처를 취할 수 있도록 가정내의 제품과 장치를 위한 특징들을 일괄적으로 세부설명하고 있다. 예를들어, 이 사양서는 "사용자가 외출중일 때" 혹흔 "집에서 자고있을 때" 등 실내에서의 다양한 상태를 분류하고 가정의 설비들에 대해 보안장치, 실내등 소등, 혹은 온도조절 등 적절한 조치를 취할 수 있도록 한 것이다. HPP 사양은 또한 가정관리가 가능하도록 윈도우 95형 어플리케이션의 개발에 관한 정보를 포함한다.

EIA-600내의 공용언어는 광범위한 산업분야(즉, 오락, 컴퓨터, 냉/온방, 주방설비 등)에서 생산되는 가정용 LAN 제품의 통신을 위한 기본구조를 제공한다.

각 산업부문은 "응용구문" (즉, 문법적 규칙)을 정의하여 이에 의거하여 제품이 언어를 이용할 수 있도록 했다. CIC는 광범위한 산업분야에서 개발한 "조화적" 응용구문을 돕는 지원구조 역할을 하도록 만든 것이다. CIC의 HPP는 가정 LAN시장에 CAL형 상호운용성 제품을 제공하는 상기의 산업분야에 적합한 조화적인 응용구문의 요약이다. CEBu/일반 CAL사양은 참조를 위해 수록했다.

매체 접근의 개괄

PLX은 중앙집중식 토큰 통과방식의 데이터그램 감지 다중접속 프로토콜 혹은 DSMA/CTP로 특징지을 수 있다. 다중집합체가 동일한 매체(100)에 접근할 수 있기 때문에 PLX는 데이터를 상기 매체(100)에 탑재하려 할 때 사용할 각 노드에 대한 공통의 규칙을 설명한다.

PLX는 다수의 프로토콜로부터 취한 특징들을 통합하여 단일의 효율적 결정론적인 환경을 창출했다. PLX은 데이터그램 검출기능을 제공한다. 각 PLX 노드는 매체의 교통을 감지하고 또한 매체(100)가 현재 휴지상태인 경우 자기주장할 수 있게 해준다. 충돌회피기능은 토큰 통과식 메카니즘을 통해 제공된다. PLX는 매체에 대한 접근을 관할하는 단일 중앙 임의노드를 선택하는 방법을 포함한다. 중앙노드(실행서버)는 토큰이 실행장치에 존재하는 것을 확인한다. PLX는 선택형 동적 폴링 기능을 통해 설계의 단순화, 실행용이성, 무충돌형 접근, 계통적 수용성 및 토큰의 후속철회, 또한 신뢰성있는 데이터 (요청/응답)전달을 위한 수령확인 시퀀스 등을 제공한다.

PLX는 노드가 "비실행"상태일 때 매체(100)를 "조용한" 상태로 만들 수 있다. 보통 PLX는 "실행" 노드만 매체(100)와 통신한다. 또한 PLX는 프러그-n-플레이 기능에 대해 세계적으로 통용되는 어드레싱 방식 및 매체(100)에 대한 다중노드 경쟁을 격리하는 알고리즘을 제공한다.

PLX는 흐름형 어플리케이션에 대해 시간결정론 혹은 정확한 시간슬롯과 또한 신속신행 시간의 경우 짧은 셀길이(패킷길이)를 제공한다.

PLX 는 다중지원, 핫 스와핑, 보완, 제어 및 관리패킷 등을 제공한다.

그 밖에도 PLX는 상위 프로토콜에 다양한 제어 네트워크 특징을 제공한다. 따라서, 매체접근 방법학은 다양한 학문적 특징과 장점을 활용하여 달성된 것이다.

매체접근 방법학

매체접근 방법학은 매체(100)에 대한 접근이득에 수반하는 규칙의 아우트라인이다. 매체(100)에 대한 접근이득의 PLX 방법은 다음의 3가지 사건을 포함한다.

1. 데이터그램 검출 혹은 "청취"

2. 버스에서의 방출, 및

3. 중앙집중식 토큰 통과

노드는 실행네트워크 서버노드 혹은 클라이언트노드인 장치에 존재하는 토큰에 대해 특징적이다. PLX 장치에서, 매체(100)에 대한 초기접근은 실행성에 대한 청취 및 실행네트워크 서버임을 주장함으로써 이루어지며, 궁극적으로는 이 서버에 의해 체계적인 중앙집중식 토큰 통과가 실행된다.

도 5는 노드가 매체(100)와 "소통"하는 임의의 상태에 대한 PLX에 의한 매체접근 알고리즘을 도시하는 플로우차트이다. 도 5의 플로우차트는 파워업 및 알림 프로세스 블록(801)에서 시작하며, 여기서 각 노드는 파워업 상태에서 매체(100)에 자신의 존재를 알린다. 알림이 끝나면, 프로세스는 결정블록(802)로 진행한다. 노드는 전송(Tx) 대기명령을 수신할 때까지 결정블록(802)에서 정지(휴지)하며 이때는 결정블록(803)으로 진행한다. 결정블록(803)에서 노드가 라인업 카드상에 있지않고 실행서버이라면 다시 데이터그램 검출블록(804)으로 진행하고, 그렇지 않다면 결정블록(816)으로 간다. 결정블록(816)에서, 노드가 토큰을 수신했다면 전송패킷블록(814)로 가고, 그렇지 않다면 타임아웃 결정블록(810)으로 진행한다. 결정블록(810)에서, 타임아웃이 일어나지 않았다면, 이 과정은 결정블록(816)으로 가고, 그렇지 않다면 데이터그램 검출블록(804)으로 진행한다. 전송패킷블록(814)에서 전송패킷을 송신하고 폴링블록(815)로 진행한다. 폴링블록(815)에서, 도 7에 도시된 바와 같이 실행네트워크 서버는 실행노드를 폴링시키거나 혹은 노드가 클라이언트라면 복귀한다. 폴링블록(815)이 끝나면 결정블록(802)으로 진행한다.

데이터그램 검출블록(804)에서, 노드는 일정기간 동안 매체(100)를 청취하고 그후 결정블록(805)으로 진행한다. 이 블록(804)의 청취기간 중 매체가 각성하면 LIP 요청 결정블록(806)으로 진행하고 그렇지 않다면 블록(812) 으로 진행한다. 이 블록(812)에서, 노드는 "웨이크업" 패킷을 송신하고 결정블록(814)으로 간다. 결정블록(814)에서, 3개의 웨이크업 패킷이 아무런 응답을 얻지 못한채 송신되면 자기주장블록(813)으로 진행한다. 그렇지 않으면 데이터그램 검출블록(804)으로 진행한다. 자기주장블록(813)에서, 노드는 자기를 실행네트워크 서버노드로 주장학고 이 프로세스는 전송패킷블록(814)로 진행한다.

LIP요청 결정블록(806)에서, LIP요청 여부를 체크한다. LIP요청이 없다면, 타임아웃 결정블록(809)으로 진행하고 그렇지 않다면 처리블록(807)으로 진행한다. 타임아웃 결정블록(809)에서 특별한 패킷 타임아

웃 기간이 지났는지 체크한다. 기간이 지났다면 결정블록(802)으로 복귀하고 그렇지 않으면 LIP요청 결정블록(806)으로 복귀한다.

처리블록(807)에서 노드는 버스상에 방출한 후, 결정블록(808)으로 진행한다. 결정블록(808)에서, 노드가 드래프트 되었는지 체크한다. 노드가 드래프트된다면 수신토큰 결정블록(816)으로 진행하고 그렇지않다면 LIP요청 결정블록(806)으로 복귀한다.

블록(802),(803),(810) 및 (814)-(816)은 중앙집중식 토큰 통과 알고리즘의 일부이다. 블록(804),(805) 및 (811)-(813)은 데이터그램 검출(청취) 알고리즘의 일부이다. 블록(806)-(809)는 버스 방출 알고리즘의 일부이다.

도 5에서 도시한 바와 같이, 매체(100)에 대한 초기접근은 매체(100)가 "자는" 혹은 "깨어있는" 상태인지에 따라 2가지 방식 중 하나에 의해 달성된다. 매체(100)가 슬리핑 상태이면 접근대상 노드는 자기가 실행서버라는 자기주장을 하게 될 것이다. 매체(100)가 실행중이라면(즉, 실행네트워크 서버 역할을 한다면) 접근대상 클라이언트 노드는 실행네트워크 서버에게 접근을 요청할 것이다. 실행네트워크 서버는 접그을 요청한 클라이언트 노드의 라인업카드를 관리한다. 클라이언트 노드는 "버스상의 방출" 프로세스를 통해 라인업에 들어갈 것을 요청한다.

일반적으로, 서버형 노드는 실행네트워크 서버라고 자기주장할 수 있으나 해당노드 내에 서버형 애트리뷰트를 갖는 것이 필요조건은 아니다.

실행네트워크 서버를 선택하면, 폴링처리된 실행노드 목록을 담은 "라인업 카드"를 생성 및 관리할 수 있어야한다. 모든 실행노드가 (노화 프로세스를 통해) 비실행 상태가 되었을 때, 실행네트워크 서버는 실행서버로서의 현재 상태를 철회하고 매체(100)는 다시 휴지(슬리핑)상태로 된다. 보통 실행네트워크 서버는 매체(100)에 전송할 것을 가진 노드에 의해 자체지정된다.

실행노드는 노드가 일정시간동안 침묵할 경우 라인업 카드로부터 삭제된다. 실행노드는 또한 고우선순위 데이터의 노드가 라인업 카드에 접근해야할 경우 라인업 카드에서 삭제된다. 라인업카드는 최대 슬롯수를 갖는다. 다시 말하면, 라인업 카드는 라인업 카드에 들어갈 수 있는 최대수의 노드를 갖는다. 슬롯수는 보통 매체(100)에 이용되는 대역폭 및 각종 네트워크 노드에 필요한 대역폭으로 결정된다. N이 라인업카드의 최대수인 경우 또한 t는 특정 실행노드가 토큰을 유지할 수 있는 최대시간량(밀리초단위)인 경우, 실행노드는 약 N*t 밀리초 동안 토큰을 수득할 것이다. 따라서 라인업 카드는 활성노드가 정규의 예측가능한 기초에서 폴링처리된다는 결정론을 제공한다.

예를들어, 흐름형 비디오 데이터가 오디오보다 고우선순위이다. N 흐름형 비디오 노드가 라인업 카드에 들어있는 경우 라인업 카드 상에 입력을 요청하는 흐름형 오디오 노드는 거부당한다. 흐름형 오디오 노드는 그러나, 라인업 카드에 입력을 요청할 때마다 토큰을 받게된다. 이것은 라인업 라인의 애트리뷰트 중 하나를 예로 든 것이다. 라인업 카드에 열거된 노드는 자동폴링되며 따라서 토큰 요청없이 정규기초상에서 토큰을 수득하게 될 것이다. 라인업 카드 상에 열거되지 않은 노드는 토큰 요청 혹은 라인업 카드에 들어갈 것을 요청한 후에만 토큰을 수용한다

특정한 네트워크 노드가 제공하는 데이터의 우선순위는 하기의 노드 프로파일 대상에 관련하여 기술되는 네트워크_클래스에서 결정된다. 특정 노드를 위한 네트워크_클래스 역시 노드 어드레스의 최고 4비트(장치_종류 영역)에서 발견된다.

노드 세마포어

각 PLX 노드는 장치의 현재상태 및 시스템 내의 노드수반을 반영하는 2개의 지역 세마포어를 관리한다. 이 세마포어는 노드가 청취 프로세스를 초기화할 것인지 결정하도록 조력한다. 일반적으로, 매체(100)에 대한 접근을 수득하므로 노드는 이들 2개의 세마포어를 관리한다(노드가 전송될 것을 가진 경우).

제1 세마포어는 "장치상태"를 반영한다. 이 장치상태는 매체(100)가 실행되는지의 여부에 따라(즉, 패킷을 매체(100)에서 볼수 있는지) "깨어" 있거나 "잠든" 상태이다.

제2 세마포어는 "지역노드상태"이다. 지역노드상태는 노드의 가능한 3가지 상태 중 하나를 반영하며 다음과 같다. (1)노드는 실행네트워크 서버노드이다. (2)노드는 실행 클라이언트 노드이다. 또 (3)노드는 비실행 클라이언트 노드이다. 지역노드상태는 노드가 청취 알고리즘을 초기화할 것인지, 노드가 라인업 카드에 있는 지(폴리되었는지), 또는 노드가 실행서버인지를 결정한다.

"장치상태" 세마포어

각 노드는 장치가 깨어있거나 잠든 상태를 각각 결정한다. 이 결정은 매체(100) 상의 라인업 삽입 요청 패킷(LIP)의 존재에 기초한다. 노드가 LIP 패킷을 보면, 장치상태 세마포어는 깨어난다. 일정시간이 지난 후 LIP패킷을 볼 수 없으면 노드는 장치를 잠든 상태로 고정시킨다. 이것은 실행네트워크 서버기 존재하면 주기적으로 LIP 패킷을 전송하여 클라이언트 노드가 계속 깨어있게 한다는 뜻이다.

노드는 이 세마포어를 사용하여 이것이 매체(100)를 청취해야할 것인지 결정한다. 장치상태가 잠든 경우에만 청취 프로세스를 통해 매체(100)를 주장할 노드가 필요하다.

"라인노드 상태" 세마포어

활성네트워크 서버는 클라이언트 노드에 의한 최종전송뒤 1-10초동안 라인업 카드에 현존하는 클라이언트 노드에 토큰(폴)을 지속 분배할 것이다. 이 때, 활성네트워크 서버는 노드가 통과전송하고 라인업카드의 클라이언트 노드를 "노화" 시키는 것을 결정한다. 클라이언트 노드가 현재 토큰을 수신하면, 실행되는 것으로 간주한다. 비실행 클라이언트는 실행네트워크 서버에 의해 라인업 카드 속으로 삽입된 후 매체(100) 상에만 전송할 수 있다. 하기의 표 A1는 가능한 노드 세마포어상태이며 각 상태가 매체에 대한 전송 측면에서 의미하는 뜻이다.

표 A1

(새 전송에 대기한 노드의 후속실행)

장치상태	노드상태	다음 전송실행
깨어있음	실행	라인업 카드 온: 토큰 대기
깨어있음	비실행	라인업 카드 오프: 버스상의 방출
잠든상태	실행	나쁜상태: 청취후 서버주장
잠든상태	비실행	청취후 서버주장

데이터검출 혹은 "청취"

장치상태 세마포어는 노드의 청취개시 여부를 결정하는 기본변수이다. 또한 노드가 자신이 활성네트워크임을 자기주장할 것인지 또는 클라이언트로서 순응할 것인지를 결정하는 기본변수이기도 하다. 일반적으로, 청취는 슬리핑 장치에 대한 초기전송 이전에 실행된다. 임의의 노드를 매체(100)상에 전송한다면 활성네트워크 서버는 이미 LIP패킷을 전송하고 토큰분배를 임의화하기 위해 선택되었고 따라서 장치가 깨어난다.

노드가 매체(100)상에 송신 예정인 패킷을 가진 것을 결정하면, 장치상태 세마포어가 깨고 노드는 청취 프로세스를 통과하여 다음단계를 결정하고 초기프로세스 시기의 충돌을 최소화한다. 이것은 두 개의 노드가 매체(100)를 위해 경쟁할 수 있는 PLX 네트워크 상에 소정시간동안만 존재하며 충돌상태를 눈에 보이지 않게 할 수도 있다. 따라서, 로버스트 백업 알고리즘을 제공한다.

청취시 어드레스에는 두가지 경우가 있다. (1)노드에 전원을 공급했고 "알림" 혹은 "CAL-Ping" 패킷을 전송하여 현재의 장치에 추가할 것을 알려야 하는 경우, (2)노드가 비실행상태였고 장치를 깨우는 시도를 하는 경우이다.

어떤 경우나 서버가 청취시 검출되면 노드는 즉시 LIP패킷을 검색시작한다. LIP패킷은 노드를 실행네트워크 서버 라인업 카드에 삽입할 것이고 그후 토큰통과 및 노드전송으로 진행된다.

초기 "청취/Ping" 알림

노드에 전원을 공급하자마자 방송 CAL-Ping패킷을 전송하여 장치상의 존재를 알린다. 이것은 항상 "풀(pull)" 시키기 위해 노력하는 대신 정보를 "푸시(push)"하여 자동발견 메카니즘이 더욱 로버스트 처리할 수 있도록 해준다. 전원공급 받은노드가 장치에 대한 아무런 정보를 가지고 있지 않기 때문에, 이것의 청취 알고리즘은 정상 웨이크업 프로세스와 약간 다르다.

초기 청취는 CAL-ping 패킷을 방송하기전 800ms 정도 계속될 수 있다. 이것은 소정시간동안 교통에 대한 정확한 청취 후, 존재할 경우 이 노드를 폴링처리할 기회를 서버에게 제공하기 위해 상기 시간동안 방송 웨이크업 패킷을 3회 전송한다. 이 시퀀스는 3회 반복하며 끝나면 CAL-ping패킷을 전체 노드에게 방송하여 장치상에 성공적으로 입력되었음을 확인시킨다. 청취/ping 프로세스의 시퀀스는 다음 같이 셰도-코드로 주어진다.

1)

a) 125ms 미만의 임의 시간동안 매체(100)을 청취한다(LIP패킷을 관찰).

b) 방송 웨이크업 패킷을 600us 간격으로 3회 전송한다.

c) 125ms 시간이 다 끝날 때까지 계속 청취한다.

2)

a) 125ms 미만의 임의 시간동안 매체(100)을 청취한다(LIP패킷을 관찰).

b) 방송 웨이크업 패킷을 600us 간격으로 3회 전송한다.

c) 125ms 시간이 다 끝날 때까지 계속 청취한다.

3)

a) 125ms 미만의 임의 시간동안 매체(100)을 "청취"한다(LIP패킷을 관찰).

b) 방송 웨이크업 패킷을 600us 간격으로 3회 전송한다.

c) 125ms 시간이 다 끝날 때까지 계속 "청취"한다.

4) 활성네트워크 서버로 주장하고 방송 "CAL-ping" 패킷을 전송하여 존재를 확인한다.

5) 실행네트워크 서버에 대한 주장을 철회한다.

상기 청취/ping 프로세스는 노드에 전원이 들어온 후 즉시 실행되며, 그후의 잠복기는 중요하지 않다. 실시간 웨이크업 프로세스는 더 자주 일어나며 잠복기는 짧을수록 좋다.

실시간 "청취/웨이크업" 시퀀스

노드에 전원을 공급하고 장치에 존재를 알리면 실시간 모드가 진행되기 시작한다. 실시간 모드 진행중에 노드가 패킷을 슬리핑 상태의 장치에 전송해야할 경우, 유사한 사건시퀀스를 통해 바람직한 서버를 깨우려고 한다. 바람직한 서버가 없고 더 이상 실행네트워크 서버가 존재하지 않을 경우, 노드는 자신을 실

행네트워크 서버로 주장하여 클라이언트 노드를 폴링하기 시작한다. 청취/웨이크업 알고리즘에 대한 셰도-코드 청취는 하기와 같다. 이 알고리즘 이외에, 응답시간을 신속히 하기 위해 노드는 매체(100)를 별도로 모니터하고 또한 장치상태를 반영할 지역노드 세마포어를 사용할 수 있다. 지역노드 세마포어는 웨이크업 패킷과 함께 사용하여 이 프로세스에 관계된 잠복시간을 축소시킨다.

1)

a) 125ms 미만의 임의 시간동안 매체(100)을 청취한다(LIP패킷을 관찰).

b) 방송 웨이크업 패킷을 600us 간격으로 3회 전송한다.

c) 125ms 시간이 다 끝날 때까지 계속 "청취"한다.

2)

a) 125ms 미만의 임의 시간동안 매체(100)을 청취한다(LIP패킷을 관찰).

b) 방송 웨이크업 패킷을 600us 간격으로 3회 전송한다.

c) 125ms 시간이 다 끝날 때까지 계속 청취한다.

3)

a) 125ms 미만의 임의 시간동안 매체(100)을 "청취"한다(LIP패킷을 관찰).

b) 방송 웨이크업 패킷을 600us 간격으로 3회 전송한다.

c) 125ms 시간이 다 끝날 때까지 계속 "청취"한다.

4) 활성네트워크 서버로 주장하고 다음 패킷을 전송하여 존재를 확인한다.

5) 실행네트워크 서버에 대한 주장을 철회한다.

버스에 대한 방출

"방출" 과정은 장치의 노드가 전송대기 패킷을 가질 때 장치가 깨어날 때(활성네트워크 서버가 존재하고 현재 토큰을 분배하는 경우) 일어난다. 활성네트워크 서버는 매체(100) 위에 접근허용 권한을 가진 노드뿐이다. 실행 네트워크 서버의 라인업 카드는 비실행 클라이언트 노드가 매체(100)에 대한 접근을 이득할 수 있는 메카니즘이다. 노드는 방출하여 실행네트워크 서버의 라인업 카드에 수록할 수 있게 한다.

일반 실시간 운용에서, 네트워크는 하나 이상의 상태, 즉 잠자는 상태와 깨어있는 상태를 나타낼 것이다. 방출 프로세스는 네트워크가 현재 어떤 상태인지에 따라 다소 차이가 있다.

잠자는 상태 및 깨어있는 상태

네트워크는 실행네트워크 서버가 현재 서비스(전송패킷)를 요청하는 노드가 없음을 결정할 때 잠자는 상태로 진행하며, 그결과 토큰 전송을 중단한다. 네트워크를 오프하기 전 실행네트워크 서버는 소정기간동안 시리즈형 차단 LIP(LIPG) 요청 패킷을 전송한다. 시리즈 LIPG 요청패킷이 아무런 응답도 소거할 경우, 실행네트워크 서버가 비활성화 되며 네트워크는 슬립 상태로 진행한다. 정상적인 주장의 처리를 통해 네트워크에 전송을 요청하는 노드가 입력되고 상술한 알고리즘을 청취한다.

깨어있는 상태는 하나이상의 원격노드와 함께 실제교환정보인 지정된 네트워크 상에 노드를 기호화한다. 깨어있는 상태에서 매체접근은 활성 네트워크 서버와 이것의 라인업 카드에 의해 제어된다. 충돌은 현재 라인업 카드에 있는 노드에 대해 토큰통과방식을 이용하고 또한 이 라인업 카드에 불러오려는 노드에 대해 방출시키면 감소된다.

"버스상의 방출" 시퀀스

버스상의 방출 시퀀스는 실행네트워크 서버가 LIPG패킷을 주기적으로 전송하도록 해준다. 슬리핑 상태의 클라이언트 노드는 LIPG패킷에 응답한다. 응답이 나타나면, 실행네트워크 서버는 모든 노드에 장애없는 LIPD 요청을 전송하여 토큰이 필요한 노드의 어드레스에 단일응답이 있기를 희망한다. 1개이상의 노드가 토큰에 대해 경쟁하면, 응답은 나타나지 않으며 따라서 실행네트워크 서버는 노드격리 시퀀스로 진행한다.

도 6A 및 6B는 실행네트워크 서버 및 클라이언트 노드에 대한 버스상의 방출 프로세스를 각각 도시한다. 도 6A에서, 노드가 실행네트워크 서버이면 실행네트워크 서버에 대한 버스상의 방출 프로세스는 출발블럭(901)에서 출발한다. 이 프로세스는 출발블럭(901)에서 폴링블럭(902)로 진행한다. 폴링블럭(902)에서, 실행네트워크서버는 라인업 카드에 현재 있는 모든 클라이언트 노드를 폴링한다. 폴링이 끝나면, 전송블럭(903)으로 진행한다. 전송블록(903)에서, 실행서버 노드는 장애없는 LIPG요청을 전송하고 계속해서 결정블록(904)으로 진행한다. 전송블록(904)에서, 실행서버는 LoGI 응답을 발견한다. LoGI 응답을 수신하면 이 프로세스는 처리블록(905)로 진행하거나 그렇지 않으면 폴링블록(902)로 복귀한다.

처리블록(905)에서, 실행서버는 장애없는 LIPD 요청을 전송하고 다시 결정블록(906)으로 진행된다. 결정블록(906)에서, 실행서버는 직접ACK(DACK) 응답을 관찰한다. 단일DACK 응답을 수신하면 처리블록(907)로 진행한다. 다중 DACK응답을 수신하거나, 아예 DACK응답이 없을 경우 노드격리블록(910)으로 진행한다. 처리블록(907)에서 DACK응답을 송신한 클라이언트 노드를 라인업 카드에 추가하고 이 과정은 폴링블록(902)으로 돌려보낸다.

처리블록(910)(노드격리 알고리즘이 시작됨)에서 이 프로세서는 LIPG 마스크를 초기화하고 처리블록(911)으로 진행된다. 처리블록(911)에서, 마스크는 갱신(즉, 마스크의 다음 비트는 중단된다)되고 이 프로세스는 전송블록(912)으로 진행한다. 전송블록(912)에서, 차단 LIPG요청이 송신되고

결정블록(913)으로 진행한다. 결정블록(913)에서, LoGI응답을 나타낸다. LoGI 응답을 수신한 경우 결정블록(915)으로 진행하며 그렇지않으면 처리블록(914)으로 간다. 처리블록(914)에서 처리블록(911)에 가장 최근 고정된 차단기 비트는 고정상태에서 풀려 결정블록(915)으로 진행한다.

결정블록(915)에서, 차단기의 모든 비트가 고정되었다면 처리블록(918)으로 진행한다. 그렇지 않으면 처리블록(911)으로 복귀한다. 처리블록(916)에서, 실행네트워크 서버는 차단된 LIPG 요청을 전송하고 결정블록 (917)으로 진행한다. 결정블록(917)에서 DACK 응답을 수신하면 처리블록(907)으로, 그렇지 않으면 폴링블록(902)으로 진행한다.

처리블록(903)-(907)은 서버 방출 초기시퀀스의 일부이다. 처리블록(910)-(917)은 서버방출 노드격리 시퀀스의 일부이다.

도 6B는 클라이언트 방출 알고리즘을 도시한 플로우차트로, 클라이언트가 실행네트워크 상에 있는 출발블록(981)에서 시작한다. 출발블록(981)에서, 전송상태를 검사할 결정블록(982)으로 진행한다. 전송상태는 "대기"이며 결정블록(983)으로 진행한다. 그렇지 않으면 유휴블록(988)(유휴블록은 결정블록(982)에 복귀한다)으로 진행한다.

결정블록(983)에서 노드가 장치의 토큰을 수신한 경우 전송블록(989)으로 진행하고, 그렇지 않으면 결정블록(984)으로 진행한다. 전송블록(989)에서, 노드는 데이터패킷을 전송하고 결정블록(982)으로 진행한다. 결정블록(984)에서, 노드가 LIPD 요청을 수신하면 처리블록(990)으로 진행한다. 그렇지 않으면 결정블록(986)으로 진행한다. 결정블록(986)에서 장치의 슬립상태 혹은 타임아웃을 체크한다. 타임아웃 혹은 슬립상태이면 처리블록(987)로 진행하고 여기서 현재의 노드가 자신을 실행서버롤 주장한다.

처리블록(990)에서 LIPD의 차단물은 현재 노드의 주소와 비교하고 결정블록(991)으로 진행한다. 결정블록(991)에서, 마스크가 노드와 부합하면 응답블록(992)으로 가고, 그렇지 않으면 결정블록(982)으로 간다. 응답블록(992)에서 노드는 네트워크 서버(LoGI 혹은 DACK)에 적절히 응답하고 이 프로세스는 결정블록(982)으로 복귀한다.

그룹 LIP(LIPG) 대기행렬

네트워크가 깨어있는 동안, 실행네트워크 서버는 주기적으로 그룹LIP 대기행렬을 방송한다. 그룹LIP(LIPG) 대기행렬은 논리그룹 격리(LoGI)응답을 다수의 노드에게 요청한다. 이 메카니즘은 무충돌 메카니즘의 분주한 네트워크 기간에 라인업 카드 속에 삽입될 기회를 클라이언트 노드에 제공한다. LoGI 패킷의 미는 다중동시노드가 이 패킷종류를 전송하고(동일시간내에 있다고 가정하여) 그결과 1개의 LoGI 패킷이 나온다는 것이다. 따라서, 다중LoGI 응답은 수신노드에서 관찰되는 단일 LoGI 이다.

초기LIP 시퀀스패킷은 누군가 네트워크 상에서 LIP 시퀀스를 출발시켜 라인업 카드속으로 삽입하길 원할 경우를 결정하기 위해 송신된 차단없는 그룹 LIP(LIPG)이다. LoGI 응답이 관찰되면, 기회는 단일노드가 삽입을 원하는 경우 뿐이며 따라서 차단되지 않은 직접LIP(LIPD)패킷을 그 다음에 송신한다. 직접응답을 관찰할 수 없으면 후속의 LIPG패킷이 차단된 어드레스에 그룹패킷으로 송신된다. 이것은 라인업카드 속에 삽입될 특별한 노드를 분리하기에 필요한 것으로서, 노동력이 필요한 비효율적인 격리메카니즘이다. 이것은 계통적으로 비트마스크, 즉 단일비트의 원격노드(32비트) 어드레스를 1회에 분리할 수 있는 비트마스크를 계통적으로 전송하여 달성한다. 분리메카니즘은 2개 이상의 충돌노드가 동시에 토큰을 요청하는 경우 실행해야한다.

직접 LID(LIPD) 대기행렬

직접 LID(LIPD) 대기행렬은 LoGI 응답을 LIPG 대기행렬의 결과로 송신된다. LIPD 대기행렬의 목적은 마스크처리 되지 않은 LIPD 요청을 모든 노드에게 전송하여 LIP 프로세스를 편리하게 진행하고 단 하나의 노드만 응답한다(가장 일반적인 시간조건). LIPD 패킷은 정상 DACK응답으로 응답하며, 응답노드의 어드레스를 포함한다. 단일노드가 응답할 경우, 응답이 관찰되며 노드어드레스는 라인업 카드에 적절히 추가된다. 그러나, LIPD요청을 관찰할 수 없을 경우(다중노드가 동시에 응답하는 것 때문에) 실행 네트워크 서버는 정상격리 알고리즘을 통해 지속격리되고, 또한 LIPG패킷을 사용하여 경쟁노드 중 단 하나만을 선택하여 "라인업 카드"에 삽입하는 작업을 계속한다.

따라서 LIPD 패킷은 격리 프로세스를 편리하게 수행하기 위해서 사용하며 단 1개의 노드가 요구에 응답한다.

노드 격리 시퀀스

노드가 초기 LIPG에 응답하지만 어떤 이유로해서 단일응답은 LIPD 대기행렬 중에 발견되지 않은 경우 실행 네트워크서버는 자동으로 노드격리를 수행한다. 격리 시퀀스는 LIPG 패킷을 사용하며 이것이 LoGI 응답을 요구한다. 이것은 실행네트워크 서버에 의해 다중동시응답을 확인시켜 준다.

"실행 네트워크서버"는 패킷을 제1 어드레스(가장 덜 중요한) 비트세트와 함께 패킷을 전송하여 시퀀스를 개시하는 것이다. 전송이 필요한 노드는 특정 어드레스 비트가 자신과 일치할 때만 이 패킷에 응답한다. 이 알고리즘은 최초의 마스크와 비교함에 따른 단순한 "AND" 이다. 2개의 값이 일치하면, 패킷은 LoGI과 함께 응답을 받는다.

실행 네트워크서버는 그후 다음 패킷을 앞서의 일치된 마스크와 함께 접촉이 없는 상태로 전송한다. 다시 노드는 전체비트 시퀀스가 일치하는 경우 응답할 것이다. 응답하는 노드가 하나도 없는 경우 실행네트워크 서버는 현재의 비트를 모두 삭제하고 패킷을 재시도한다. 이것은 32비트가 모두 확인되고 일치상태를 발견할 때까지 계속된다. 이 시점에서, 특별히 식별된 노드를 실행네트워크 서버의 라인업 카드에 첨가한다.

중앙집중식 토큰 통과(폴링)

장치가 깨어있으면, 매체(100)에 접근할 수 있는 결정론적 시간슬롯을 라인업 카드(방출과정을 거쳐) 상에 포함된 각 노드에 제공하는 것이 바람직하다. 또한 노드에 대해 바쁜 매체(100) 상에 전송할 기회를 제공하는 것도 바람직하다. 이더넷은 상술한 장점이 부족하며 반대로 토큰링은 이 두 장점을 모두 갖는다.

토큰링은 각 노드에게 이웃 어드레스 즉 상부흐름과 하부흐름을 알도록 또한 토큰의 지속적인 존재/회전을 요청하는 단점을 가진다. 기존의 토큰링 네트워크의 오버헤드 요건는 PLX와 상응하는 단순노드와 호환되지 않는다. 또한 전원형 네트워크의 ad hoc 네트워크 요건은 정확한 토큰회전 등에 맞지 않는 단점도 있다. PLX 는 동적 라인업 카드를 구비한 중앙집중식 토큰 통과(CTP)메카니즘을 소개한다.

CTP에서, 실행네트워크 노드는 토큰의 존재, 토큰이 필요한 노드가 이것을 수득하였는지, 슬립 상태의 노드가 깨어나 토큰을 수신하는지, 또한 토큰을 결정론적 방식으로 분배하는지에 대해 확실히 응답한다. CTP하에서, 실행서버를 제외한 모든 노드를 클라이언트라고 정의한다. 실행네트워크 서버의 역할은 상술한 데이터그램의 검출 혹은 청취 프로세스를 통해 자체지정된다. 이 서버의 역할은 매체(100) 상의 비실행 예정기간이 지난후에 철수하는 것이다. 한 예에서, 실행서버의 역할은 약 5초간의 비실행이 있은 후 철수하는 것이다. 장치의 실행과정에서, 실행네트워크 서버는 라인업 카드내의 각 클라이언트 노드를 폴링하는 역할 및 새로운 노드에게 방출프로세스를 통해 라인업 카드 속으로 자신을 삽입시킬 수 있는 기회를 주는 역할을 한다.

도 7은 네트워크서버 폴링 알고리즘을 도시하는 플로우차트이고, 이것은 노드가 실행서버가 되는 출발블록(1001)과 함께 시작한다. 출발블록(1001)에서 결정블록(1002)까지 프로세스가 진행되며 이 결정블록에서 주기적인 LIP패킷의 전송을 요구하는지 결정한다. LIP패킷이 필요하면 처리블록(1010)으로 진행하고 그렇지 않으면 처리블록(1003)으로 진행한다. 처리블록(1010)에서, 노드는 도 6A에 관계된 실행서버 방출 프로세스를 실행한다. 이 블록(1010)이 종결되면 처리블록(1003)으로 진행한다.

처리블록(1003)에서, 라인업 카드의 다음 입력을 얻고 결정블록(1004)으로 진행한다. 이 블록(1004)에서 모든 입력치가 처리되었다면(즉, 클라이언트 노드가 말할 수 있다면) 처리블록(1011)로 간다. 그렇지 않으면 처리블록(1005)로 진행한다. 처리블록(1011)에서, 토큰은 실행서버에 제공되어(실행서버가 말할 수 있고) 처리블록(1005)로 진행한다.

처리블록(1005)에서 토큰이 라인업 카드로부터 얻은 다음 노드에 제공되면 결정블록(1007)로 간다. 결정블록(1007)에서, 응답 타임아웃이 발생하면 처리블록(1012)로 가고 그렇지 않으면, 결정블록(1007)로 진행한다. 결정블록 (1007)에서 클라이언트 노드가 토큰을 사용하지 않았다면 처리블록(1012)로 진행한다. 처리블록(1012)에서 실행노드의 숫자를 줄이고 결정블록(1008)로 진행한다.

결정블록(1008)에서 모든 노드가 비실행이면 처리블록(1009)로 가고 그렇지않으면 결정블록(1002)로 간다. 처리블록(1009)에서 실행서버는 비실행 클라이언트 노드에 복귀한다.

패킷종류와 포맷

PLX 네트워크의 패킷은 패킷 목적에 따라 다른 형태의 포맷을 취할 수 있다. 다른 포맷들은 3가지 개별 카테고리로 구분되는 것이 일반적이다.

첫번째 포맷은 복수의 노드가 동일한 응답패킷을 간섭 혹은 탈모듈화의 문제 없이 동시에 전송/수신할 수 있도록 하는 것이다. 이것을 논리그룹격리(LoGI)패킷이라고 하며 방송/재방송 및 수령확인 등에 기본적으로 사용된다.

또다른 2종류의 패킷은 단일노드가 주어진 시간에 와이어에 통신할 때 사용되는 미가공 데이터 페이로드 패킷 및 명령페이로드 패킷이다. 미가공 데이터 페이로드 패킷은 어플리케이션에 합당한 정보를 전송/수신할 것을 요구하는 어플리케이션에서 사용한다. 호스트노드에서 나온 패킷은 미가공 데이터 페이로드 패킷 및 기타 CAL 패킷이다.

PLX 명령 페이로드 패킷은 매체 접근 및 유동을 관리한다. PLX 명령캐핏은 어댑터의 펌웨어 및 하드웨어 내에서 시작 및 종결된다. 또한 호스트노드를 통과하지 않는다. PLX 명령패킷은 원활한 토큰흐름, 수령확인, 라인업 삽입 등을 촉진하고 모든 PLX네트워크에 부여된 고유한 특징이다.

논리그룹 격리(LoGI) 응답패킷

제1형은 노드가 그룹요청(다중동시응답의 가능성이 있는 요청)을 네트워크에 송신할 때 이용된다. PLX가 충돌감소 혹은 무충돌환경을 원하므로 충돌을 검출하기는 어렵다. 따라서, 동시응답이 바람직하다. LoGI 패킷(1100)은 도 8에서 보는 바와 같이 2바이트 NULL영역을 포함하며 다수의 2바이트 "1"영역이 이어지고 다시 마지막으로 2바이트 NULL영역이 연결된다. 이러한 패킷형에서의 데이터는 숨기기에 적합하며 단일노드에 대한 그룹응답을 격리를 돕는 목적을 달성한다.

LoGI 패킷은 차단된 LIPG 요청에 선행한다. 하나 이상의 노드가 마스크 어드레스와 일치할 수 있으며 다중동시응답도 발생한다. LIPG패킷은 이하에 설명한다.

LoGI 패킷은 또한 특별한 패킷 내에 존재하는 시리즈를 길이로 연장하여 매우 간단한 데이터를 함유할 수도 있다. 연장된 패킷은 다양한 응답을 표시하기 위해 시간변위와 함께 사용한다. 방송패킷은 바쁜 응답을 하나 이상의 노드로 동시방식에 따라 표시할수 있도록 하는데 이 특징을 이용한다.

페이로드 패킷

제2 형태는 네트워크 주변에 페이로드를 이동시키는데 이용한다. 이것은 보통 네트워크에서 사용하기에 바람직한 데이터 정보를 전송 및 수신하는데 유리하다.

페이로드 패킷은 또다시 2가지 형태로 나뉘는데 수신감시범위 및 수신이 예측되는 응답 종류를 가리키는

것이다. 이들은 그룹어드레스(방송패킷) 및 직접어드레스 패킷 종류이다. 그룹어드레스 패킷은 LoGI 응답패킷만 수용할 수 있으나, 직접어드레스 패킷은 단일 응답만 예상되므로 직접 수령확인 혹은 DACK패킷을 수용한다.

페이로드 패킷 종류는 다시 두가지의 개별 카테고리로 분할되며 페이로드 사용을 패킷 내에서 결정하는 것이다. 이들은 미가공데이타 패킷 및 PLX 명령패킷이다.

미가공데이타 패킷

미가공데이타 패킷(1200)의 포맷은 도 9에서 도시되었으며 예비영역(1201), 길이영역(1202), 길이영역(1203) 및 ctrl 영역(1204), 수신어드레스 영역(1205), 소스 어드레스 영역(1206), 시퀀스영역(1207), 감시영역(1208), DSK영역(1209), SSk영역(1210), 페이로드영역(1211), 및 CRC영역(1212)를 포함한다. 미가공데이타 패킷(1200)은 실행서버 노드 혹은 클라이언트 노드에 의해 송신된다. 길이영역(1202),(1203) 및 ctrl영역(1204), 수신어드레스 영역(1205), 소스어드레스 영역(1206), 시퀀스영역(1207), 그리고 감시영역(1208), DSk영역(1209) 및 SSk영역(1210)은 MAC헤더(1215)의 요소들이다. 페이로드영역(1211)은 적절한 페이로드 처리기로 분석할 어플리케이션층 정보를 포함한다. 호스트 PC 및 CAL 번역기는 페이로드 처리기의 예이다. 한 예에서, 미가공데이타 패킷(1200)은 3바이트 예비역(1201), 13-15바이트 MAC헤더(1215), 최대 255바이트까지의 페이로드부(1211) 및 2바이트 CRC(1212) 등을 갖는다.

PLX(외부)명령 패킷

PLX명령 패킷은 간단한 패킷 시퀀스를 통해 통신하기 위해 2개의 노드를 위한 수단을 제공하여 매체(100)의 데이터 온오프 흐름을 돕는데 이용한다. PLX명령패킷의 변형은 다음과 같다.

토큰패킷: PLX 토큰패킷(1300)의 포맷은 도 10에 도시되어 있으며, 예비영역(1201), 길이영역(1202), 길이영역(1203) 및 ctrl 영역(1204), 수신어드레스 영역(1205), 및 CRC영역(1212)를 포함한다. 길이영역(1202),(1203) 및 ctrl 영역(1204)는 각각 (6진수)값인 0 ×05, 0 ×05, 및 0 ×17을 갖는다.

토큰패킷(1300)은 직접어드레스 노드에 송신되며 어느쪽 페이로드 패킷을 요청한다. 주의할 필요가 없는 노드는 간단히 DACK(상태영역을 0 ×03으로 설정)이고, 이것은 이들을 다루거나 토큰사용을 하지 않는다는 뜻이다.

클라이언트 노드는 실행네트워크에 전송하기 앞서서 (LIP 프로세스를 통해)토큰을 불러낸다. 노드가 토큰을 계속 사용하는 한 실행네트워크 서버는 계속 토큰을 전달하게 된다. 그러나 클라이언트 노드가 "토큰 사용할 수 없음"으로 반복응답할 경우 이 서버는 노드를 노화시켜 라인업으로부터 삭제하게 될 것이다.

토큰패킷은 일반적인 MAC헤더(마이너스 소스어드레스) 및 CRC를 함유하나, 데이터영역은 사용하지 않는다(데이타영역의 크기가 0). 토큰은 어드레스가 0xfffffffe로 고정된 "실행 네트워크서버"에서만 얻을 수 있으며 소스어드레스는 필요치 않다.

직접 수령확인(DACK) 패킷: PLX 토큰패킷(1400)의 포맷은 도 11에 도시하였으며 예비영역(1201), 길이영역(1202), 길이영역(1203) 및 ctrl 영역(1204), 수신어드레스 영역(1205), 및 CRC영역(1212)를 포함한다. 길이영역(1202),(1203) 및 ctrl 영역(1204)는 각각 (6진수)값인 0 ×06, 0 ×06, 및 0 ×07을 갖는다.

DACK 패킷은 패킷 혹은 패킷 시퀀스의 유효수신을 수령확인하기 위한 수신노드에 의해 송신된다. DACK 패킷은 직접 어드레스되는 메시지 패킷으로부터만 복귀된다(LIPD패킷 제외).

DACK 패킷은 네트워크의 2개 노드 사이의 일반적인 핸드셰이킹 시퀀스를 종결하는데 사용되며 따라서 다음의 3개 노드를 수반한다. (1)실행 네트워크서버, (2)노드요청, 및 (3)노드응답(현재의 요청을 수신하는 경우 철회/응답노드 역시 "실행 네트워크서버"가 될 수 있다). DACK 상태영역은 패킷수신노드형에 따라 바뀔 수 있다(실행네트워크 서버 혹은 클라이언트). 요청노드에 송신복귀한 DACK 패킷(응답노드에 의해)은 요청노드에 대한 제어를 철회하여 패킷흐름을 지속시키고, "실행네트워크 서버"로 송신복귀한 DACK 패킷(응답노드에 의해)은 "실행네트워크 서버"에 대한 제어를 철회하여 패킷 흐름의 종결을 확인한다. 요청노드는 응답 혹은 DACK 패킷이 수신되지 않는 경우 패킷을 재요청하도록 되어있다.

DACK 패킷은 일반적인 MAC 헤더 및 CRC 또한 1바이트 페이로드를 함유한다. 상태영역은 수신된 패킷에 대한 정보를 함유하며 이영역으로 복귀된다. 상태영역(1401)의 값은 하기의 표 A2와 같이 열거할 수 있다.

표 A2

(DACK 상태영역(1401)의 값)

DACK	노드	설명
0 ×0	전체	
0 ×1	전체	수신 버퍼 용량부족(실패)
0 ×2	서버	실패(다중채널 응답)
0 ×3	서버	토큰이 노드에 의해 사용됨
0 ×4	전체	토큰이 노드에 의해 사용되지 않음
0 ×9	전체	"웨이크업" 요청에 대한 토큰 응답

0 ×a	전체	프린터 시퀀스 숫자 오류
0 ×b	전체	프린터 오프라인 오류
0 ×c	전체	프린터 일반 오류
0 ×d	전체	프린터 용지공급 오류
0 ×e	전체	프린터 알 수 없는 오류
0 ×f	전체	성공

이들 정보는 실제매체(100) 자체에 전송된 것이며 호스트 노드까지 전달된 상태는 아니다. 호스트까지 전달된 상태정보에 대한 상세내용은 내부PLX 패킷에 대한 장을 참고한다.

라인업 삽입 패킷(LIPD 및 LIPG): 도 12에서 PLX LIPG 패킷(1500)에 대해 도시하며 이것은 예비영역(1201), 길이영역(1202), 길이영역(1203) 및 ctrl 영역(1204), 수신어드레스 영역(1205), 마스크영역(1501) 및 CRC영역(1212)를 포함한다. 길이영역(1202),(1203) 및 ctrl 영역(1204)는 각각 (6진수) 값인 0 ×09, 0 ×09, 및 0 ×23을 갖는다.

도 13은 PLX LIPD 패킷(1600)의 포맷을 도시하며 이것은 예비영역(1201), 길이영역(1202), 길이영역(1203) 및 ctrl 영역(1204), 수신어드레스 영역(1205), NULL(1601) 및 CRC영역(1212)를 포함한다. 길이영역(1202),(1203) 및 ctrl 영역(1204)는 각각 (6진수)값인 0 ×09, 0 ×09, 및 0 ×23을 갖는다.

라인업 삽입 패킷(LIP)은 주기적으로 "실행 네트워크서버"로부터 송신받아 기존의 라인업 카드에 새로운 것을 추가할 수 있다. 이것은 2가지의 개별 패킷에 의해 달성되며 모두 청취노드에 대해 방송한다. 제1 패킷인 LIPG패킷(1500)은 LIP 마스크영역(1501)을 포함한다. 마스크(1501)는 LoGI 응답에 대응하기 전에 원격 어드레스를 일치시켜야 한다. 제2 패킷인 LIPD 패킷(1600)은 소스어드레스 (라인업 카드에 삽입될)를 함유하는 DACK 패킷에 응답노드가 대응함으로써 삽입과정을 신속히 하는데 이용한다.

따라서, LIPG패킷은 마스크되고 LIP 마스크영역의 상응하는 비트 시퀀스를 갖는다. 노드는 LoGI 패킷으로 LIPG 패킷에 응답한다. 마찬가지로, LIPD 패킷은 마스크 해제되며 이는 라인업 카드에 들어갈 노드(이미 라인업 카드에 들어간 노드는 제외)는 DACK으로 응답한다.

페이로드 패킷 프레임 포맷

다음은 페이로드형 패킷내에 존재할 수 있는 각 영역에 관계한다. 이것은 미가공 데이터 및 PLX 명령패킷 형태 양측에 대해 참(ture)이다.

예비/개시 시퀀스는 패킷의 포맷에는 포함되지 않으나 캐리어 검출, 하드웨어를 인커밍 패킷에 동기화하고, 또한 현재의 패킷 내에 후속 바이트의 비트시간 혹은 라인속도를 측정하는데 사용되는 예정된 비트 패턴이다. 예정부의 길이는 유효 캐리어의 존재를 설정하고 동기화하는데 필요한 최소한의 비트시간으로 정해진다. 예정부(1201)의 비트패턴은 다음과 같다.

값	시퀀스
0 ×aa	1st sync byte
0 ×31	2nd sync byte
0 ×nn	속도/3rd sync byte

속도(또는 3rd sync) 바이트는 인커밍 데이터 (길이바이트(1202)에서 시작)의 속도를 결정하며 다음과 같이 요약된다.

값	속도
0 ×55	저속 - 650k
0 ×dd	중간속도 - 1000k
0 ×99	고속 - 1.19m
0 ×11	유지

마지막으로, 예정부 뒤에는 2개의 중복된 길이바이트(1202),(1203)이 있으며 이들은 패킷의 길이를 설명한다. 이들 바이트는 새로운 속도에 적용된다.

길이영역

길이영역(1202),(1203)은 인커밍 패킷의 크기를 표시한다. 길이영역(1202),(1203)은 하드웨어 (캐리어 검출신호가 없는)에 의해 사용되며 유효패킷 수신을 결정한다. 패킷길이에 도달하면 CRC영역(1212)의 유효성을 시험한다. PLX 패킷의 길이는 총 256 바이트로 한정하는 것이 적절하다(예비영역(1201) 및 CRC영역(1212)를 제외). 길이는 MAC헤더(1215)(제어부, 어드레스 등), 선택영역 및 페이로드영역(1211) 등을 포함한다.

길이영역은 2배로 중복되어(1202, 1203) 인커밍 데이터 흐름의 유효성을 확인시킨다(예비영역의 연장부 역할을 한다). 길이영역(1202),(1203)은 패킷수신에 앞서서 서로 일치시켜야 한다(예비정합).

제어영역

상술한 바와 같이, 페이로드 패킷은 다음 2가지 주요형태 중 하나이다: PLX 명령 패킷 혹은 미가공 데이터 패킷.

PLX 명령 패킷은 다시 2가지 서브형태로 구별된다: 외부 및 내부 PLX 명령. 내부 PLX 명령 패킷은 지역 접속부(USB, 1584, 시리얼 등)를 통한 하드웨어 및 호스트 노드 드라이버 간의 핸드셰이크이다. 외부 PLX 명령 패킷은 매체(100) 접근을 조정할 전원 매체(100) 자체 상의 핸드셰이크 패킷이다.

제어영역(1204)는 표 A3에서의 정의에 대해 각 비트로 표시한 패킷의 종류에 따라 다르다.

표 A3

(제어영역(1204)의 비트)

비트	PLX(EXT)	PLX(INT)	무가공(NON-PLX)
0	PACKET TYPE(1)	PACKET TYPE(1)	PACKET TYPE(0)
1	PLX_SUBTYPE(1)	PLX_SUBTYPE)0)	RAW_ACK_TYPE0
2	PLX_ACK_TYPE	예약(0)	RAW_ACK_TYPE1
3	예약(0)	예약(0)	CIPHER
4	EXT_SYBTYPE	INT_SUBTYPE	SOCKET
5	EXT_SYBTYPE	INT_SYBTYPE	예약(0)
6	EXT_SYBTYPE	INT_SYBTYPE	PID
7	EXT_SYBTYPE	INT-SYBTYPE	예약(0)

패킷종류

패킷종류 비트는 해당패킷이 PLX 혹은 미가공데이타 혹은 비-PLX 인지를 구별하는 것이다. PLX 프로토콜 요청은 마이크로컨트롤러 펌웨어에 의한 대부분의 경우가 다르게 처리되고, 무가공 데이터 패킷은 별도의 어플리케이션이나 호스트 소프트웨어로 처리되므로, 제어영역에서 분류하는 것이 빠르다. 미가공 데이터 패킷은 보통 적절한 응용소프트웨어에 전달되는 미가공 페이로드 정보를 갖는다. 이러한 경우의 예외는 마이크로컨트롤러의 변환기 일부 및 호스트장치의 일부를 포함하는 CAL패킷이다.

비트 0 패킷종류

1 PLX 명령패킷 = 1

0 미가공 데이터 패킷 = 0

PLX 서브패킷 종류

패킷명령은 두가지 형태중 하나이다. 제1형은 다른 노드에 의해 와이어로부터 오는 요청이고, 제2형은 호스트로부터 오는 요청으로 이것은 와이어 상에 전달되지 않는다. 마이크로컨트롤러 펌웨어는 이들 2가지 종류에 대응함에 있어서 구별되고 서로 완전히 별개이므로 이 비트가 생성된다.

비트 1 PLX 서브패킷종류

1 외부 PLX 명령패킷 = 1

0 내부 PLX 명령패킷 = 0

PLX ACK 종류

토큰 및 DACK 명령패킷은 매체(100)에 대한 접근권리를 전달하고, "실행네트워크 서버"가 다른 노드로 매체(100)의 제어를 임시로 해제하는 경우 시퀀스를 종결하는데 이용된다. 응답 종류는 LoGI 혹은 DACK 형태가 있다. 이 비트는 노드를 활용하는 종류의 응답을 결정한다.

비트 2 PLX ACK 종류

1 DACK을 이용한 응답 = 1

0 LoGI을 이용한 응답 = 0

PLX 서브패킷 외부 종류

PLX 사양은 중앙집중(서버 임의 토큰) 토큰 통과장치 내의 2개의 노드간에 무선형 수령확인 및 확인안된 데이터 전달서비스를 제공한다. 이 비트는 통신을 실행하기 위한 것이다.

실행 네트워크서버는 클라이언트가 전송을 시작하기 전 매체(100) 상에 직접 토큰을 제공한다. 클라이언트 노드는 실행서버 노드에 복귀하는 DACK 응답 패킷으로 매체(100)에 대한 접근권을 종결한다. 실행서버는 클라이언트 노드를 폴링할 때 실행노드의 라인업 카드를 유지한다. 라인업카드 상에 보내기 위해, 클라이언트 노드는 직접 LIP 요청(LIPD) 혹은 그룹LIP요청(LIPG)에 대해 응답한다.

라인업 카드에서 노드가 폴링될 경우 또한 이들은 페이로드 정보의 패킷을 수령확인 혹은 미확인 포맷으로 송수신할 수 있다. 다음은 매체(100)에 허용되는 테이블 Aof 유효 PLX 서브패킷 외부형태이다.

비트(7,6,5,4) 바이트 값 PacketSub-Type

0 0 0 0	0 ×07	DACK
0 0 0 1	0 ×17	토큰
0 0 1 0	0 ×27	LIPD
기타...	0 ×23	LIPG
		예약

주의: DACK/GACK 이 예정된 간격조건내에 요청노드에 의해 수신되지 않을 경우, 전송(요청 혹은 응답)노드는 요청(응답)을 재시도한다.

PLX 서브패킷 내부 종류

PLX 사양은 프로토콜의 일부가 PC 같은 호스트노드에 존재할 수 있게 한다. 주기적으로 호스트노드는 물리적으로 연결된 부착노드의 정보에 접근할 필요가 있다. 이것은 부착노드 때문에 내부 PLX 요청이라하고 원격노드에 전송될 와이어 상에 위치하지 않아도 좋다. 다음은 가능한 내부 PLX 서브타입이다.

비트(7,6,5,4)	바이트 값	PacketSub-Type
1 1 1 1	0 ×f1	ERROR 핸드셰이크
0 0 0 1	0 ×11	CAL 요청
0 0 1 0	0 ×21	CAL 응답
0 0 1 1	0 ×31	Tx 상태
1 1 x x		예약

내부 서브종류는 호스트에서 송신되며 하드웨어가 사용한다. 적절한 응답을 호스트노드에 반송한다. 내부패킷은 매체(100) 상에 전송되지 않는다. 따라서, 이 패킷종류는 페이로드 패킷부에 의해 한정되지 않으며 PLX (내부) 호스트패킷하에서 한정된다.

미가공 ACK 종류

미가공 ACK 종류는 현재의 미가공 데이터 패킷에 따른 응답종류이다. 응답종류는 4가지 형태이다. 버스트(응답없음), 더블LoGI, LoGI 및 DACK 이다.

버스트형은 자체설명과 같고, 패킷을 차례로 송신한다. 이 시퀀스의 마지막 패킷은 다른 수령확인종류(버스트 시퀀스를 종결하기 위한 응답 사용)를 갖는다.

더블LoGI 시퀀스는 방송 요청 혹은 그룹을 송신할 수 있는 것이다. 노드가 패킷을 버퍼처리할 수 없을 경우, 제1 간격 내에 응답하며 패킷을 정확히 수신하여 분석하고 지연된 간격 동안에 응답한다.

LoGI 응답은 단일노드를 향하며 응답에 가장 효과적인 메카니즘이다. LoGI 패킷의 길이가 가장 효율적인 대역폭이나 응답에 대해 많은 정보를 담을수 없다는 단점이 있다.

DACK 응답은 특정노드를 지향하나 LoGI형보다 응답내에 더 많은 정보를 담을 수 있다.

비트(2,1)	패킷 서브타입
0 0	버스트
0 1	더블LoGI
1 0	LoGI
1 1	DACK

Cipher

사이퍼비트는 권한바이트에서 출발하여 패킷 내용물을 암호화 할 수 있게 해준다. 하나의 암호방식은 256 비트 Diffie-Hellman 핸드셰이크를 키이변환을 위해 사용하고 그후, 비밀 32-비트 어레이를 매체(100)에 비밀송신 하는 것이다. 후속의 트랜잭션으로 암호화 어레이를 통신보안을 위해 사용할 수 있다.

비트3: 사이퍼

현재 패킷은 암호화 됨 = 1

현재 패킷은 암호화 안됨 = 0

소켓

PLX 미가공데이타 페이로드 패킷은 다음의 영역크기로 되어 있다.

영역	길이
예비영역(1201)	3 바이트
길이(1202),(903)	2 바이트 중복
제어부(1204)	1 바이트

수신어드레스(1205) 4 바이트

소스어드레스(1206) 4 바이트

페이로드(1211) 0-255 바이트

CRC(1212) 2 바이트

동일노드에 복수 어플리케이션이 존재하면 메카니즘을 사용하고 이에 의해 패킷이 특정 노드 어드레스 내의 적절한 어플리케이션에 경로화될 수 있다. 제1 바이트는 수신 소켓 어드레스이며 제2 바이트는 소스 소켓 어드레스이다. 따라서, 이 비트를 지정하면 MAC 헤더크기는 2만큼 늘게된다. 이 영역은 사용할 때 권한바이트영역 뒤에 따르며 다음처럼 지정된다.

비트4 소켓

1 소켓영역을 포함

0 소켓영역을 불포함

프로토콜 ID(PID)

각 패킷은 상위프로토콜 즉 IPX, TCP/IP 혹은 CAL로 분석할 수 있는 정보를 담고 있다. PLX는 이들 패킷을 캡슐화하는 전달부로 사용하여 네트워크에 패킷을 송/수신한다. 보통 상위분석 경로는 호스트장치에 존재한다. 그러나 하드웨어는 CAL 분석기능의 최소 단위만 담으면 된다. 따라서, 하드웨어는 CAL 요청을 분석하고 기타 다른 요청은 페이로드 처리경로까지 전달한다. 프로토콜 정보 중에는 하드웨어에 로케이트 살수 있는 경우도 있으며 (ROM, FLASH메모리 등) 어떤 것은 호스트 노드가 분석하기도 한다. 이 비트는 하드웨어 프로토콜 처리기가 패킷의 분석을 개시하는데 필요한지를 결정한다.

비트 6 프로토콜 ID(PID)

1 프로토콜 ID 존재 (Micro Parse)

0 프로토콜 ID 없음 (미가공 - Host Parse)

미가공 패킷은 제1 데이터 바이트가 프로토콜형 바이트코드가 아니라 프로토콜 헤더 자체의 제1 바이트라는 걸 의미한다. PID 분석형 패킷은 어떤 프로토콜이 패킷을 분석할 것인지 결정하기 위해 제1 바이트 코드를 해독한다.

다음은 PID 비트 설정에 필요한 옵션이다. 제1 데이터 바이트는 현재의 패킷을 분석하는데 필요한 프로토콜 형태를 나타낸다.

바이트값	정의	종류
0 ×ff	예약	n/a
0 ×fe	COMPLETE Packet	CebusResp
0 ×fd	FALSE Packet	CebusResp
0 ×fc	ERROR Packet	CebusResp
0 ×df - 0 ×fb	예약	n/a
0 ×a0 - 0 ×de	Context Numbers(CAL)	CebusResp
0 ×9f	예약	CebusResp
0 ×00 - 0 ×9e	Context Calss(CAL)	CebusResp

수신 어드레스 영역

수신어드레스(1205)는 현재 패킷의 수신노드를 함유한다.

노드가 요청을 갖거나 다른 노드의 요청에 응답할 때, 응답패킷이 수신되는 노드의 어드레스를 수신어드레스 영역(1205)에 표시한다. 노드가 실행서버나 데이터베이스 서버에 통신할 수 있을 경우 수신어드레스 영역(1205)에 이 주소가 담긴다. 아니면, 수신어드레스는 요청패킷의 소스어드레스 영역(1206)에서 취한다.

PLX 어드레스는 잘 알려져 있다. 이 공지된 PLX 어드레스는 다음과 같다.

어드레스	설명
0 ×00000000 - 0 ×fffffef	유효단일노드 어드레스
0 ×fffffff0 - 0 ×fffffffc	예약
0 ×fffffffd	응용서버 노드 어드레스
0 ×fffffffe	실행 네트워크 서버 노드 어드레스
0 ×ffffffff	방송 노드 어드레스

소스 어드레스 영역

소스어드레스(1206)은 현재 패킷의 노드 어드레스를 담고 있다. 노드가 요청을 갖거나 다른 노드의 요청

에 응답할 때, 소스어드레스 영역(1206) 속으로 자체의 노드 어드레스를 탑재한다. 노드어드레스는 노드 종류와 함께 8비트 GUID의 일부를 활용하며 4비트 노드어드레스를 형성한다. 최소 7개의 니블을 GUID로부터 사용하며 노드종류는 노드어드레스의 대부분의 니블(제8 니블)을 덮어쓴다.

예:

IF,...

GUID = 0 x 0123456789ABCDEF

AND Node Type = 0.03

THEN..

Source Address = 0 x 39ABCDEF

End If

시퀀스번호 영역

시퀀스 영역(1207)은 매체(100) 상의 전송을 위한 소형 패킷으로 분할된 데이터 패킷 혹은 시퀀스를 재생 혹은 재배열하는 기능을 갖는 호스트 어플리케이션을 제공한다. 중복시퀀스수를 제거하고 수신안된 시퀀스수를 재송신할 수 있다. 시퀀스처리는 더 큰 데이터흐름을 위해 데이터를 통합하는 것이다. 시퀀스영역(1207)내의 값은 어플리케이션에 따라 다르며 필요시 변경목적으로 사용할 수 있다.

권한영역

권한영역(1208)은 완전수신에 앞서 패킷의 유효화를 위한 것이다. 권한영역(1208)은 보통 암호화 어레이의 처음 2바이트를 독점화하여 시드한다. 따라서 보안장치내의 모든 노드는 동일한 권한값으로 시드처리되며 확인절차를 거치게 된다. 권한영역은 통합성 강화를 위해 역시 암호화처리된다.

페이로드 영역

데이터 페이로드영역(1211)은 수신정보에 정보를 표시하는 역할을 한다. 페이로드 데이터의 제1 바이트는 바이트코드를 담고 있으며 이것은 내용분석 방법을 결정한다. 데이터의 제1 바이트는 앞서의 미가공 바이트와 함께 사용된다.

사이클 중복검사(CRC) 영역

이 영역(1212)은 전송된 패킷내 오류를 검출하는 신뢰성있는 방법을 제공한다. 끝낸 뒤에 재평가하며 확인값과 비교한다. 이 검사를 통과하지 못한 패킷은 삭제한다.

CRC 알고리즘은 규정외 오버헤드(소프트웨어 및 하드웨어에서)가 없이 확실성 있는 값을 제공할 수 있도록 효과적이고 가능한 간단한 것을 선택한다. 전송 및 수신 패킷 양측에 있어서 실행중 CRC연산을 할 수 있을 만큼 빠른 CRC 알고리즘을 제공하는 것이 좋다. 실행중 연산(전체패킷을 기다리지 않고 비트 혹은 바이트 수신시, CRC가 갱신된다)는 위임사항은 아니나 장치 전체의 실행기능에 도움을 준다.

한 예로, G(X)는 $G(X) = x^{16}+x^{15}+x^{11}+x^{8}+x^{6}+x^{5}+x^{4}+x^{3}+x+1$ 로 표시된다.

PLX(내부) 호스트 패킷

PLX 내부 호스트 페킷은 매체(100)에 도달하지 않는다. 따라서 패킷설명도 훨씬 간단해 진다. 예비영역(1201)이나 중복 길이영역(1202),(903), 어드레싱영역(1205),(906) 도 필요없다. 또한 CRC영역(1212)도 필요없다. 도 14에서 PLX 내부호스트 패킷의 포맷을 보여준다. 여기에는 길이영역(1701), 제어영역(1702) 및 데이터영역(1703)이 포함된다. 데이터 영역(1703)은 제어영역이 한정한 것을 모두 담고 있다. 앞서의 제어영역에 대한 정의에서와 같이 (PLX 내부 호스트 패킷도 마찬가지로 응용된다) 하드웨어 및 교통흐름을 촉진하는 호스트 노드 사이를 통과흔 다수의 패킷이 존재한다. 다음의 이들 각 패킷의 정의이다.

CAL 요청 패킷

도 15는 CAL 요청 패킷(1800)의 포맷을 보여주며 이것은 길이영역(1701), 제어영역(1702) 및 CAL 데이터영역(1803)을 포함한다. 제어영역(1702)는 0 x 11의 값을 갖는다.

CAL 요청 패킷(1800)은 하드웨어 상에 있는 CAL정보를 보상하기 위해 호스트가 하드웨어 노드에 송신한다. PLX노드는 하드웨어/ASIC 로부터 분리된 호스트 프로세서의 응용코드를 가질 수 있기 때문에, CAL정보 역시 이들 2가지 개별 프로세서에 제공될 수 있다. 따라서, 호스트 프로세서는 주기적으로 CAL 정보를 부착노드로부터 모은다.

CAL 응답 패킷

도 16에서 CAL 응답 패킷(1900)의 포맷을 도시하며 여기에는 길이영역(1701), 제어영역(1702) 및 CAL응답영역(1903)이 포함된다. 제어영역(1702)은 0 ×21의 값을 갖는다.

상술한 것과 마찬가지의 이유로, CAL 응답 패킷은 하으웨어 노드에서 부착된 호스트노드로 송신된다. 이 응답패킷(1900)은 사전 CAL요청 패킷(1800)에 응답하여 송신되는 것이다.

Tx 상태패킷 (단일채널, 속도)

도 17은 단일채널 CAL 응답패킷(2000)의 포맷을 도시하며 여기에는 길이영역(1701), 제어영역(1702) 및 데이타영역(1903)이 포함된다. 제어영역(1702)은 0 ×21의 값을 갖는다. 도 18은 다중채널 CAL

응답패킷(2100)의 포맷을 도시하며 여기에는 길이영역(1701), 제어영역(1702) 및 데이타영역(2103)이 포함된다. 제어영역(1702)은 0 ×31의 값을 갖는다.

2가지의 Tx상태패킷 종류가 있다. 제1형은 단일채널, 단일속도 응용 및 제어바이트 값 0 ×21을 사용하는 것이고, 제2형은 다중채널, 다중속도 및 0 ×31의 제어바이트 값을 갖는다.

단일 채널, 단일속도는 2개의 Tx버퍼만 이용가능하고, 이 2개의 Tx버퍼는 주기적으로 내부 PLX 핸드세이크를 통해 호스트노드에 복귀한다. 이들 Tx상태 패킷의 목적은 호스트노드에서 하드웨어로 전달되는 외부 전송사건에 관한 루프를 닫는 것이다. DACK에 복귀되는 동일한 값은 이 전송사건에 관한 정보에 있어서 호스트로 전달되나 대부분 DACK 가 외부 PLX사건이고, 이 경우 DACK값은 호스트노드로 전달되지 않는다. 호스트노드가 전송요청을 지향할 때 DACK 값이 호스트노드로 전달된다.

따라서, PLX는 다음같이 DACK 상태값을 중복사용한다.

매체에서 나타낸 DACK 상태영역값

0 ×0 = 수신버퍼 용량부족(실패)

0 ×2 = 노드가 사용한 토큰 (호스트에 전달안됨)

0 ×3 = 노드가 사용하지 않은 토큰 (호스트에 전달안됨)

0 ×4 = "웨이크업"요청에 대한 토큰 응답(호스트에 전달안됨)

0 ×9 = 프린터 시퀀스수 오류

0 ×a = 프린터 플러그풀림 오류

0 ×b = 프린터 오프라인 오퓨

0 ×c = 프린터 일반오류

0 ×d = 프린터 용지없음 오류

0 ×e = 프린터 알 수 없는 오류

0 ×f = 성공

값 0 ×9 내지 0 ×e 는 프린터 노드로부터의 DACK응답이다. 프린터 응갑값은 바뀌지 않은 호스트노드로 복귀한다.

값 0 ×f 은 성공적인 DACK응답이며, 호스트가 요청할 때 이값도 바뀌지 않은 호스트노드로 복귀한다.

값 0 ×2 내지 0 ×4 는 PLX 명령패킷에 대한 DACK응답값이며 호스트노드로 전달되지 않는다.

유일한 다른 점은 0 ×0 이다. 이것은 와이어 상에서 수신노드가 바쁜상태이고 패킷을 수용할 수 없다는 걸 의미한다. 하드웨어는 이 상황을 인식하고 패킷을 일저횟수(덜 바쁜 경우라면 더욱 자주)만큼 재시도한다. 수신노드가 장기간 바쁜 상태에 있을 경우 패킷을 결국에는 무시하고 "실패 - 0 ×f" 응답 상태를 호스트노드에 전달하게 된다. 호스트에 입력된 0 ×0 값은 아무런 의미가 없다. 이것은 종결되지 않은 전송사건의 디폴트갓이며 호스트가 0이 아닌 상태가 될 때까지 대기하게 된다. 0 ×1 값은 와이어상으로 복귀하지 않는다. 노드가 오류데이타의 패킷을 수신하면, 이것은 패킷에 간단히 응답하지 못하고 재전송하기위해 전송노드가 필요하게 된다. 0 ×1 값은 전송패킷이 타임아웃 되었을 때, 또한 재시도 최대횟수를 넘겼을 때에만 호스트로 복귀한다.

다음은 정상적으로 호스트노드에 복귀한 Tx 상태값의 보기이다(이들 값은 모두 DACK 응답값과 다르다는 점에 유념한다).

Tx 상태 데이타필드값

0 ×0 = Tx 버퍼를 위한 Tx 상태 없음

0 ×1 = 실패 (Tx 타임아웃 혹은 버퍼 용량부족 수신)

0 ×9 = 프린터 시퀀스수 오류

0 ×a = 프린터 플러그풀림 오류

0 ×b = 프린터 오프라인 오퓨

0 ×c = 프린터 일반오류

0 ×d = 프린터 용지없음 오류

0 ×e = 프린터 알 수 없는 오류

0 ×f = 성공

이것은 다음의 DACK 정보가 내부 Tx 상태 패킷을 통해 호스트노드에 전달되지 않음을 의미한다.

호스트에 전달되지 않은 Tx 상태 정보

0 ×0 = 수신버퍼 용량부족(실패)

0 ×2 = 노드가 사용한 토큰 (호스트에 전달안됨)

0 ×3 = 노드가 사용하지 않은 토큰 (호스트에 전달안됨)

0 ×4 = "웨이크업"요청에 대한 토큰 응답(호스트에 전달안됨)

Tx 상태 바이트는 다시 2개의 부분, 즉 각각을 니블 와이드라고 하는 부분으로 분할되어 2개의 Tx버퍼 상태를 표시한다. 각각의 의미를 갖는 Tx 상태영역의 값은 다음과 같다.

Tx 상태값의 예

0 ×0f = 1차 Tx 버퍼의 송신 성공

0 ×f0 = 2차 Tx 버퍼의 송신 성공

0 ×ff = 양측 Tx 버퍼의 송신 성공

0 ×1f = 2차 Tx 버퍼의 송신 실패, 1차 Tx 버퍼의 송신 성공

기타.....

Tx 상태패킷 (다중채널, 속도)

Tx 상태 패킷의 제2형태는 다중채널, 다중속도 방식이다. 단일채널 Tx 상태 패킷에 관한 앞서의 설명과 DACK의 값에 관한 내용은 마찬가지이다. 차이는 다중채널/속도 Tx 상태패킷에 들어가는 데이터정보의 양에 있다. 패킷은 기본적으로 각 채널에 존재하는 앞서 정의된 단일상태 바이트를 함유한다. 그결과 각 바이트가 2개의 Tx 버퍼를 갖춘 단일채널을 표시하는 특징의 복수 바이트 데이터를 얻게된다.

패킷타이밍, 간격 및 재시도

매체(100)에 전송되는 모든 패킷은 타이밍 요건을 준수해야한다. 이 요건은 장치가 원할하게 운용되고 충돌현상이 없도록 하는 규칙이다. 규칙의 조건을 정확히 따라야 한다.

정상적인 조작에서 "실행네트워크 서버"는 매체(100)에 접근할 모든 실행노드와 임의연결된다. 다음의 가정은 매체(100)에 나타난 실행상태에 적용한 것이다. 매체(100)에서의 비실행은 각 노드가 슬립상태이고 "실행네트워크 서버"를 주장하기 전까지는 정상적인 "청취"가 실행되어야 한다.

또한 PLX 장치는 수령확인된 핸드셰이크 시퀀스를 특징으로한다. 수령확인 패킷은 특정한 시간간격 내에서 복귀된다. 토큰패킷은 다른 확인패킷(DACK, LoGI, 혹은 더블LoGI)을 제외하고 전송하기전에 필요한 조건이다. 실행서버는 토큰 혹은 LIP패킷을 전송할 수 있는 권한을 가진 유일한 노드이다. 클라이언트 노드는 페이로드 및 수령확인 패킷만을 전송한다.

정규 패킷 타이밍

도 19는 패킷 타이밍과 간격을 보여는 타이밍 다이어그램이다. 패킷시간은 제1 기준시간(2202)과 제2 기준시간(2204)에 관하여 정의한다. 제2 기준시간은 평균 패킷 간격(I/Gap) 50us(마이크로초) 정도로 제1 기준시간 뒤를 따른다.

상기의 다이어그램에서 약 650kbps 으로 실행되는 장치에 대해 상기의 다이어그램을 예측하였다. 간격 타이밍을 제외한 모든 값은 다음의 표 A4에 나와 있으며 여기서 윗첨자 1은 제1 기준(2202)을, 윗첨자 2는 제2 기준(2204)을 나타낸다.

표 A4

(패킷 타이밍)

	350kbps	700kbps	1.2mbps	1.4mbps
Min/Gap[1]	15㎲	15㎲	15㎲	15㎲
Avg/Gap[1]	50㎲	50㎲	50㎲	50㎲
Preamble	130㎲	65㎲	38㎲	33㎲
loGI 패킷[2]	140㎲			
DLoGI 패킷[2]	185㎲			
DACK 패킷[2]	335㎲			
TxRetry LoGI[1]	205㎲			
TxRetry DACK[1]				
TxRetry DloGI[1]				
Inter-Token[1]				

정상조건에서, 일반적인 패킷 타이밍은 패킷을 수신하는 노드가 예정 시간 내에 응답해야 한다. 이 응답 시간은 LoGI/더블LoGI 수령확인 패킷을 제외한 모든 패킷에 중요하다. 따라서 패킷 타이밍에 대한 2가지 경우는 (1)LoG/더블응답 및 (2) 기타 다른 응답이다.

기타 패킷 타이밍

노드는 예정시간 내에 패킷을 페이로드 패킷이 기원하는 곳으로부터 노드로 복귀전송한다. 예외로, 버스트 패킷 및 수령확인 패킷은 응답 패킷을 필요로하지 않는다. 응답패킷은 DACK 패킷, LoGI패킷, 혹은 페이로드패킷 이다.

응답패킷은 도 19에서 보는 것 같이 간격요건에 부합한다. 최소 응답시간은 15마이크로초 이상이며 최대 응답시간은 50 마이크로초를 초과하지 않아야 한다.

전송노드가 앞서의 전송을 확인수령하지 못한 경우, 전달의 신뢰도를 높이기 위해 재시도과정을 시작해야 한다. 재시도과정은 가장 긴 수령확인 예정 시퀀스 혹은 DACK 패킷과 가장 긴 시간간격을 합한 시간 혹은 650kbps에서 700마이크로초 정도를 넘은후 시작된다.

노드 특이정보

각 노드는 특정정보량을 갖도록로 구성되며 특정한 노드를 특징으로 한다. PLX 노드는 완전한 기능을 위해 최소량의 정보만 필요로한다.

특이성 식별, 어드레스성 및 세계공통식별(GUID)

PLX 노드를 전기장치에 연결하는 즉시 작동이 시작된다. 각 노드는 일련번호 순으로 나타나고 가장 작은 28피트가 노드의 실행 어드레스로 이용된다. 이것은 세계공통기준과 다른 것이나 어드레스 충돌하는 2개의 노드를 발견할 가능성은 1/268,000,000,000 정도로 작다. 실시간 어드레스는 감소하나 플러그&플레이 성능 및 사용의 간편성이 향상되며 시스템도 단순화 된다(노드를 공장에서 사전설정하기 때문이다).

공통구문 및 노드 프로파일 대상

CEBus/일반 CAL 보조노드는 공통구문 및 관계변수(IV)를 가진 노드제어 대상을 갖춘다. CEBus/일반 CAL 정의 보고조건 및 노드 어드레싱로부터 PLX이 벗어난다. (양측은 CEBus/일반 CAL 집단-집단 아키텍춰에 반대인 PLX 클라이언트/서버 아키텍춰에 관련된다). 따라서, PLX는 공통구문/노드 제어 대상을 다소 다른 IV정의에 따른 노드 프로파일 대상으로 재정의한다. 다시, PLX 보조노드는 노드프로프일 대상과 관련된 변수를 포함한다.

각 노드는 예정 애트리뷰트군을 포함하여 공통의 애트리뷰트를 관할하며, 가진 노드그룹에 상기 노드를 제공한다. 각 노드에 대한 노드 프로파일 대상 정보는 노드에 비휘발성 메모리로 하드코드 된다. 정보는 요청에 따라 서버에 송신한다. 노드 프로파일 대상을 변수의 목록으로 만들었다. 각 PLX 노드는 적어도 공동구문 (0x00), 노드프로프일대상(0x01) 및 특정번수 (IV)를 포함하며 이는 하기의 표 A5와 같다 (R/W는 읽기/쓰기).

표 A5

IV	R/W	종류	명칭	설명
o	R/W	d	구문_리스트	특정노드에 의해 지원되는 모든 구문목록을 함유
W	R/W	b	전원	특정노드에 대한 공용전원의 제어
s	R	d	시리얼_번호	장치GUID인 8바이트로된 생산자 제품번호(통상의 식별자)(18바이트)
n	R	c	생산자 명	생산자 제품사양명 (18바이트MAX)
m	R	c	생산자 모델	생산자 모델명(18 바이트 MAX)
c	R	n	제품 분류	일반 CAL사양(2아스키 바이트)
v	R	c	보충 레벨	특정장치 CAL/PLX지원 현재레벨 확인스트링 (4아스키 바이트)
h	R/W	d	에어리어어드레스	라우팅 및 네트워크 식별목적으로 사용(1바이트) 이 IV는 항상 판독가능함(네트워크 명칭을 따라)
a	R/W	d	유닛 어드레스	직접 어드레스 패킷용 노드ID (4바이트)
t	R	d	네트워크_클래스	장치의 네트워크 정의 및 장치MAC 어드레스의 중요니블을 덮어쓴다 하기는 네트워크종류에 관한 우선순위 및 관계값.0x01 비디오장치I0x02 비디오장치II0x03 오디오장치I0x04 오디오장치II0x05 예약0x06 보안장치 0x07 유틸리티 관측장치0x08 HVAC 장치0x09 조명장치0x0a 가전제품0x0b 데이터 네트워크장치0.x0c 예약0x0d 예약 0x0e 예약0x0f 공용장치

f	R	d	버퍼링	수신버퍼 크기(바이트)
x	R	c	제품	제품 개정레벨 (4아스키 바이트)
b	R/W	d	동적_마스크	동적노드 기능은 단일비트로 특징화한다Bit 0 예비모드1 = 가능0 = 불능Bit 1: MAC 서버1 = MAC 서버0 = MAC 서버아님 Bit 2: 규칙/데이타베이스 서버1 = 데이터베이스 서버0 = 데이터베이스 서버 아님Bit 3:장치 비실행/실행1 = 현재 실행0 = 현재 비실행
u	R	d	정지_마스크	정지노드 기능은 단일비트로 특징화한다Bit 0 원격기능1 = 가능0 = 불능Bit 1: 자동화 1 = 권한A(NV Mem요청)0 = 권한없음 Bit 2: 복합방법 지원1 = 복합방법 지원0 = 지원없음Bit 3: Diffie/Hellman Max Key 크기 1 = 812 비트 0 = 256 비트

y	R	d	통계	노드의 계수기능에 수행되는 통계표 Aof 포맷:바이트 0 ; 테이블 A버전바이트 1: 비트 바스크 카운터
r	R/W	d	리셋	사례변수에 대한 노드의 리셋, 0x52 R값을 쓴다.
l	R/W	b	슬립	노드를 오프라인 으로 혹은 수동제어
G	R/W	d	그룹 어드레스 리스트	모든 그룹 어드레스의 길이선행 리스트는 이 노드의 지원을 받음.처음 16비트 값이 그룹 어드레스의 번호임
jikgd	R/W	d	권한	권한ID 값이 구성 및 초기화시 노드에 전달. XOR 사이퍼 어레이가 Diffie-Hellman 에 사용됨비-XOR 사이퍼 어레이가 Diffie-Hellman 에 사용됨공공키공공 발생기 임의 숫자
q	R/W	c	네트워크 명	사용자가 이해할 수 있는 명칭으로 노드를 특정 네트워크에 부여
e	R/W	c	제품명	노드에 논리명 부여
p	R/W	c	제품 위치	사용자가 이해할 수 있는 명칭으로 노드를 특정위치에 설치
	R/W	d	시스템 id 목록	해당환경 도메인 에서의 할당장치ID 의 리스트
	R/W	c	래스트 로그	최종 로그 사건

표 A6은 "어플리케이션 서버"에 의해 저장, 관리 및 유지되는 클라이언트 IV를 열거한 것이며 상기 서버의 데이터베이스 내에 있다. 따라서 클라이언트는 상기 IV에 관한 정보를 보관 및 제공하는 것에 대해서는 신경 쓸 필요가 없다.

또한 마스터케이스에 관한 공통구문 부분만이 규칙대상(0 ×03) 이 되며 CAL에 의해 한정된 데이터 메모리 대상 및 다른 목적을 위해 한정된 특정IV을 이용한다. 하기는 특정한 대상이다.

표 A6

규칙대상은 규칙리스트에 들어있는 규칙들을 추가(계승), 삭제(계승상실), 및 관측(getArray)하는 방법을 원격노드에 제공할 수 있게 한다.

공통구문을 제공하면 네트워크는 노드리스트를 함유할 수 있다. 노드는 공통구문 리스트를 함유할 수 있다. 노드는 각구문의 대상리스트를 가질 수 있다. 대상리스트가 주어지면, 노드는 특정변수를 가질 수 있다. 이러한 리스트 대부분은 일반 CAL 사양에 나와 있다(네트워크 및 노드 리스트 제외).

필요시, 노드는 특정구조에 대해 도시한 노드 프로파일의 특정부분에 응답한다. 노드 프로파일은 고려대상인 네트워크 내에서 특별한 자동구성 노드에 의미를 부여한다. 중복노드는 고유식별될 수 있도록 다른 레벨의 구성을 제공할 수 있다.

보안

보안은 2단계 과정으로 실현된다. 1차로 전원에 연결된 각 노드는 즉시 공공망 속에 설치된다. 공공망은 모든 노드에 디폴트 네트워크를 제공하며 이들 노드는 다른 공공노드에 노출되고 이들의 권한ID가 NULL에 부여된다. 노드가 핵심교환 프로세스를 통과하면 권한ID가 암호화 어레이에 의해 추적된 값으로 변경된다. 각 노드가 이 개인용/보안 네트워크에 부여되기 때문에 32비트 암호화 어레이를 갖고 이를 통

해 후속 패킷을 암호처리 혹은 해독처리한다. 이것은 256비트 키이를 이용하여 Diffie-Hellman이라고 하는 키이변환기술을 통해 달성된다. 효과적인 지수함수 알고리즘을 이용하면 키이변환의 값들을 연산처리하는데 드는 시간을 절약할 수 있다. 암호화 어레이를 네트워크의 각 노드 메모리에 저장하면 바로 암호화 및 해독작업이 실행된다. 한 실시예에서, 암호/해독 작업은 단독 혹은 피드백을 이용한 흐름-사이퍼링 기술을 이용한다. 다른 알고리즘은 예컨대 DES, RC4, MD5 등이 사용될 수도 있다.

기타의 특징

기록조건의 사양

기록조건은 CAL 보다는 PLX에 의해 더 까다롭게 취급되므로 처리규칙에 관한 PLX 방법을 하기에서 나타내었다. 정확한 CAL 기록조건의 방법상 고유의 한계성을 다수 어드레스하기 위해 변화시킨 부분이 있다. 이 차이점은 아래의 표A7과 같다.

CEbus CAL	PLX
1대상 1 규칙	1대상 복수규칙
1대상 1 실행	1대상 복수 실행(IV)
단순규칙 전용	단순 및 복잡한 규칙
규칙의 불변성	유연한 규칙적용

표A7, 일반 CAL 대비 PLX의 장점

PLX 규칙은 일반CAL하에서 분포된 규칙과 달리 서버에 관한 것이므로, PLX 측이 단일 및 강력한 엔진을 통해 규칙을 이용하는 방법이 더욱 강력하다. PLX 클라이언트 노드 각각이 서버에 자신의 IV 변화를 기록한다. IV 변화에 대해서 믿을 수 있다. 서버가 IV 변경을 확인하면 변경된 특정 대상/IV 컴비네이션을 관찰하고 규칙목록을 확인한 후 각 규칙의 유효성을 검사한다. 따라서, 각 대상은 다음의 두가지 IV를 갖도록 사전설정한다. 이들 각 규칙은 하기와 같이 특정대상 및 이것과 관계된 IV에 대해 생성된 것이다

IV	R/W	종류	이름	기능
R	R/W	d	규칙_어레이	마스터 규칙 리스트에 포인터 어레이 인덱스를 함유(규칙대상). 각 입력물은 이 대상내의 IV가 수정될 경우 검사받을 규칙을 명시한다.
P	R/W	n	규칙_번호	규칙_어레이 내의 규칙번호

실제기록_헤더 및 기록_어드레스, 기록_조건 및 선행_변수값 등은 각각 어레이에 표시된 규칙에 들어있다. 호출경로는 이들 포인터(혹은 인덱스)를 규칙엔진에 전달하고 규칙엔진은 마스터 규칙 리스트에서 필요한 정보를 분석한다.

비휘발성 메모리 사용

각 노드는 노드 프로파일 정보를 고정메모리 위치 즉 ROM에 함유한다. 또한 노드는 비휘발성 메모리내에 권한키 같은 정보를 저장할 수도 있으나, 이것은 선택사항이다. 또한 PLX 보조노드가 반드시 필요한 것도 아니다. 다른 선택성 매모리 조건은 라우팅 정보 및 기타 동적기준표를 포함한다.

클라이언트 변경 통지

클라이언트 노드는 어플리케이션 서버의 노드에 상태변화조건을 기록한다. 이것은 어플리케이션 서버가 클라이언트에게 상태변화를 알릴 경우 클라이언트는 어플리케이션 서버에게 상태변경을 보고한다는 것을 뜻한다. 이것은 어플리케이션 서버 데이터베이스가 실제의 클라이언트 노드의 변수와 동기화하지 않는 문제들을 축소시킨다.

이것은 어플리케이션 서버가 클라이언트 변수의 변화에 관계된 기록조건 및 규칙을 담고 있기 때문에 필요하다. 클라이언트는 이 점에서 지능이 떨어지므로 적절한 변화를 어플리케이션 서버에게 통지해야한다.

어플리케이션 서버는, 클라이언트 노드의 유효성을 수신한 후에 클라이언트가 상태를 변경시켰음을 "어플리케이션 서버"에게 통지하기 전까지는, 데이터베이스 변수를 특정 클라이어언트 노드에 갱신하지 않는다.

(57) 청구의 범위

청구항 1

컴퓨터 네트워크 상의 복수의 노드가 하나이상의 데이터 프로토콜을 이용하여 통신할 수 있도록 구성된 게이트웨이에 있어서,

상기의 하나이상의 데이터 프로토콜은 매체 프로토콜을 이용하여 네트워크 매체에 전송되고,

게이트웨이는 상기 복수의 노드와 통신하기 위한 응용프로그램형 인터페이스를 제공하고, 이 게이트웨이가,

네트워크상에 노드에 관한 정보플 포함하는 내부노드 데이터베이스, 및

실행모드 및 대기모드를 제공하도록 구성되고, 이때의 실행모드는 내부노드 데이터베이스를 유지하고 이 베이스와 접근할 수 있도록 구성되며, 대기모드는 상기 내부노드 데이터베이스를 외부노드 데이터베이스의 거울상 복제물로 유지하도록 구성된 소프트웨어 모듈을 포함하는 것을 특징으로 하는 게이트웨이.

청구항 2

제1항에 있어서, 상기 내부노드 데이터베이스는 클라이언트의 상태변화에 대해 취하는 실행조처를 구체화하는 규칙을 또한 포함하는 것을 특징으로 하는 게이트웨이.

청구항 3

제2항에 있어서, 상기 규칙은 간단한 규칙인 것을 특징으로 하는 게이트웨이.

청구항 4

제2항에 있어서, 상기 규칙은 복잡한 규칙인 것을 특징으로 하는 게이트웨이.

청구항 5

제2항에 있어서, 상기 규칙을 해석하기 위하여 구성된 규칙엔진을 또한 포함하는 것을 특징으로 하는 게이트웨이.

청구항 6

제2항에 있어서, 상기 규칙을 규칙정의언어로 변환하기 위해 구성된 연결기를 또한 포함하는 것을 특징으로 하는 게이트웨이.

청구항 7

제2항에 있어서, 상기의 상태변화는 클라이언트 노드의 사례변수의 변화인 것을 특징으로 하는 게이트웨이.

청구항 8

제1항에 있어서, 상기 내부노드 데이터베이스는 핑(ping)요청을 기록하여 갱신되는 것을 특징으로 하는 게이트웨이.

청구항 9

제1항에 있어서, 상기 소프트웨어 모듈은 수령확인되지 않은 클라이언트 요청을 검출한 때 상기 실행모드로 변환하도록 구성되는 것을 특징으로 하는 게계이트웨이.

청구항 10

제1항에 있어서, 제1 프로토콜이 제2 프로토콜에 터널링하도록 구성되는 것을 특징으로 하는 게이트웨이.

청구항 11

제10항에 있어서, 상기 매체는 전원이고 상기 매체 프로토콜은 전원 프로토콜인 것을 특징으로 하는 게이트웨이.

청구항 12

제1항에 있어서, 상기 매체는 전원이고 상기 매체 프로토콜은 PLX 프로토콜인 것을 특징으로 하는 게이트웨이.

청구항 13

제7항에 있어서, 상기 클라이언트 노드의 사례변수에서 변화가 발생할 때 사용자 어플리케이션에 통지하도록 구성된 사건처리기를 또한 포함하는 것을 특징으로 하는 게이트웨이.

청구항 14

제1항에 있어서, 대상지향 응용프로그램형 인터페이스를 또한 포함하는 것을 특징으로 하는 게이트웨이.

청구항 15

제14항에 있어서, 상기 내부노드 데이터베이스의 정보에 대해 사용자 인터페이스를 제공하도록 구성된 인터넷 브라우저를 또한 포함하는 것을 특징으로 하는 게이트웨이.

청구항 16

제15항에 있어서, 상기 사용자 인터페이스는 사용자가 전원 네트워크 상에서 노드를 제어할 수 있도록 구성되는 것을 특징으로 하는 게이트웨이.

청구항 17

전원 네트워크 매체, 및

상기 전원 네트워크 매체로부터의 미가공 데이터 정보를 디스패치 제어블록의 이용을 통해 사용자 어플리케이션에 라우팅하는 게이트웨이 수단으로 구성되는 컴퓨터 네트워크.

청구항 18

노드 데이터베이스,

디스패치 제어블록을 생성 및 해석하기 위한 디스패처 수단,

네트워크 인터페이스 어댑터를 제어할 장치 드라이버 수단, 및

상기 장치 드라이버 수단에 장치 제어블록을 적용하기 위한 연결수단으로 구성되는 게이트웨이.

청구항 19

네트워크 상의 노드간 통신하기 위한 바람직한 프로토콜을 사용하는 방법에 있어서,

상기 노드에 대한 정보를 담은 노드 데이터베이스를 생성하고,

상기 노드 데이터베이스를 유지할 실행 게이트웨이 노드를 정의하고, 이 게이트 웨이 노드가 상기 노드 데이터베이스에 접근하는 하나 이상의 접근방법을 제공하며,

하나 이상의 대기서버 노드에 상기 노드데이타를 거울상 복제하는 단계로 구성된 것을 특징으로 하는 프로토콜 사용방법.

청구항 20

제19항에 있어서, 상태변화가 클라이언트 노드에 발생할 때 취하는 실행조치를 구체화한 실행규칙을 해석 및 실행하는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

청구항 21

제20항에 있어서, 상기 규칙은 규칙엔진에 의해 해석되는 것을 특징으로 하는 프로토콜 사용방법.

청구항 22

제20항에 있어서, 상태변화가 발생할 때 사건통지를 생성하는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

청구항 23

제22항에 있어서, 상기 통지가 디스패처에게 제공되는 것을 특징으로 하는 프로토콜 사용방법.

청구항 24

제20항에 있어서, 수신된 데이터를 규칙정의언어로 전환하는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

청구항 25

제20항에 있어서, 상기 상태변화는 클라이언트 노드의 사례변수의 변화인 것을 특징으로 하는 프로토콜 사용방법.

청구항 26

제19항에 있어서, 핑요청을 등록하고 핑요청에 대한 응답을 청취하고, 상기 응듭을 노드 데이터페이브 갱신에 사용하는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

청구항 27

제19항에 있어서, 실행서버가 비실행 상태로 된 후 상기 대기서버 노드 중 하나를 실행활성화하는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

청구항 28

제19항에 있어서, 상기 바람직한 프로토콜의 래퍼(wrapper) 패킷속으로 제1 프로토콜의 미가공패킷을 캡슐화하고 상기 미가공패킷을 바람직한 프로토콜에 터널링 통과시키는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

청구항 29

제19항에 있어서, 상기 매체는 전원이고, 상기 매체 프로토콜은 전원 프로토콜인 것을 특징으로 하는 프로토콜 사용방법.

청구항 30

제19항에 있어서, 상기 매체는 전원이고, 상기 매체 프로토콜은 PLX 프로토콜인 것을 특징으로 하는 프로토콜 사용방법.

청구항 31

제19항에 있어서, 변화가 클라이언트 노드의 사례변수에서 발생한 때 사용자 어플리케이션에 통지하는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

청구항 32

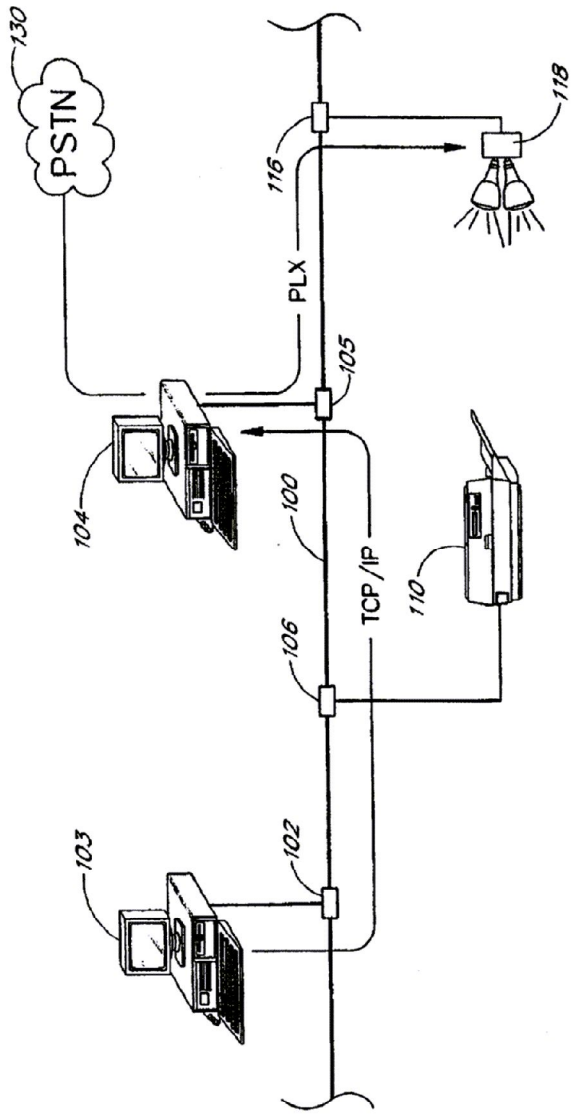
제19항에 있어서, 상기 노드 데이터베이스의 정보를 보기 위해 인터넷 브라우저를 사용하는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

청구항 33

제19항에 있어서, 전원 네트워크의 노드를 제어하기 위해 인터넷 브라우저를 사용하는 단계를 또한 포함하는 것을 특징으로 하는 프로토콜 사용방법.

도면

도면1



도면2


일반 네트워크에 연결된 PC
(데이타 장치)
207
어플리케이션층
206
표시층
205
비밀층
204
전송층
203
네트워크층
202
데이터링크층
201
물리적 층

도면3

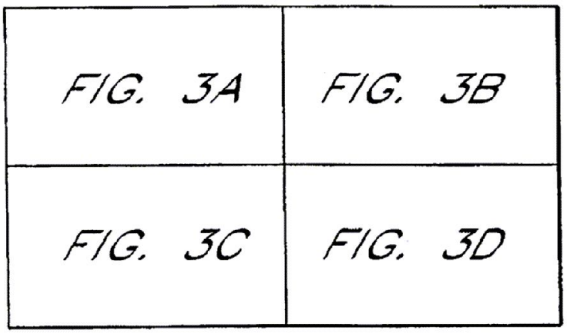
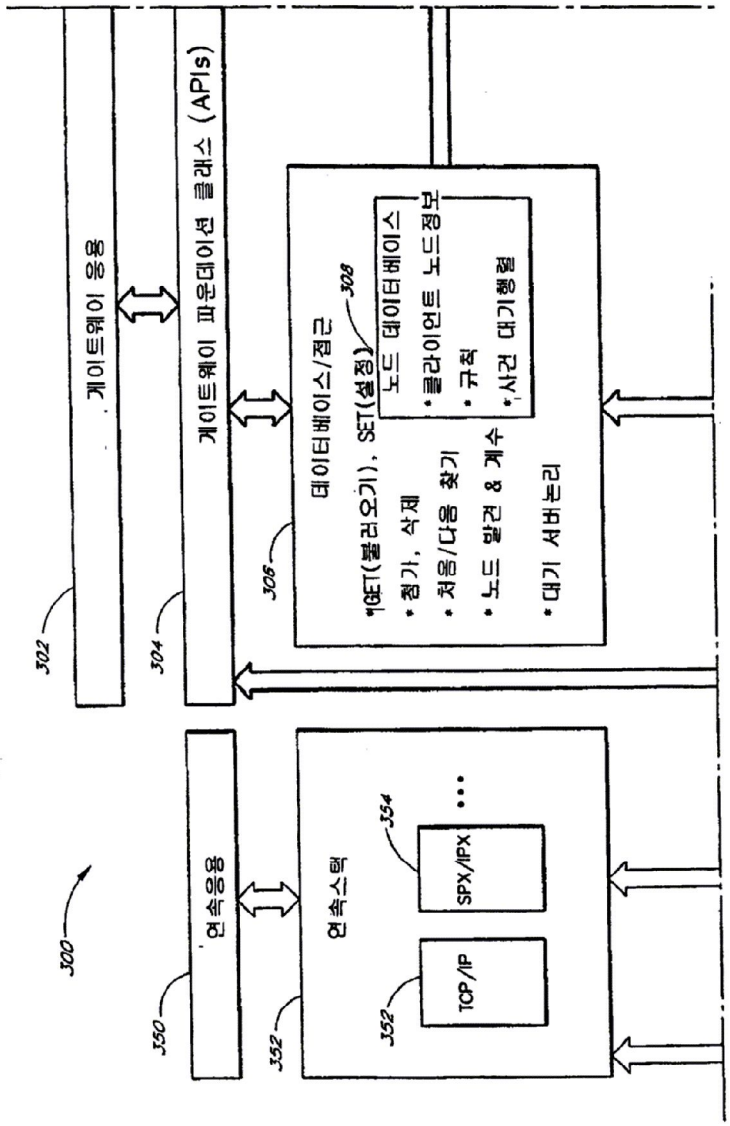


FIG. 3A
FIG. 3B
FIG. 3C
FIG. 3D

도면3a


게이트웨이 응용
게이트웨이 파운데이션 클래스 (APIs)
데이터베이스/접근
GET(불러오기), SET(설정)
• 찾기, 삭제
• 처음/다음 찾기
• 노드 발견 & 계수
• 대기 서버논리
노드 데이터베이스
• 클라이언트 노드정보
• 규칙
• 사건 대기행렬
연속응용
연속스택
TCP/IP
SPX/IPX
...
300
302
304
305
308
330
332
334
352