

□ 기술해설 □

클라이언트/서버 환경 구축과 활용2

명지대학교 이 기 현*

● 목	차 ●
3.6 분산처리 환경	5.3 원격 절차 호출
3.7 분산시스템 설계	5.4 망 파일 시스템(Network File System)
4. 개방 접속 환경	5.5 X 윈도우 시스템
4.1 망 연결(Networking)	6. 정 리
4.2 망 동향	6.1 클라이언트/서버의 장점
4.3 기존 망의 문제점	6.2 클라이언트/서버 컴퓨팅의 단점
4.4 개방 시스템	6.3 기업내 경영참여자의 각 그룹별 관점
5. 클라이언트/서버 활용	6.4 클라이언트/서버로부터 오는 기업의 기회
5.1 트랜잭션 처리	7. 결 언
5.2 온라인 트랜잭션 처리	

3.6 분산처리 환경

분산처리(Distributed Processing)라 함은 여러 지역에 분산된 프로그램들이나 다른 지역에 있는 프로그램을 마치 같은 지역에 있는 프로그램처럼 처리함을 말한다. 반면에 분산 데이터베이스 관리(Distributed Database Management)라 함은 여러 지역에 분산된 자료들이나 다른 지역에 있는 자료를 마치 동일지역에 있는 자료처럼 관리함을 의미한다.

분산처리 구조의 전제조건은 분산된 조직과 업무, 근거리 통신망(Local Area Network : LAN), 도시지역 통신망(Metropolitan Area Network : MAN), 클라이언트/서버 모형, 시스템 통합(System Integration : SI) 등을 들 수 있다.

다시 말하여 클라이언트/서버는 응용과 자료를 여러개의 조각으로 나누고 이를 프로그래밍이 가능한 여러개의 워크스테이션에 나누어 분산 수행하게 함으로써 그 성격상 분산처리 환경을

만들며 또한 그 환경 내에서 분산의 기능을 갖는다.

클라이언트/서버에서 분산모델의 성능은

- 분산 표현 - 원격 자료관리
- 원격 표현 - 분산 자료관리
- 분산 기능

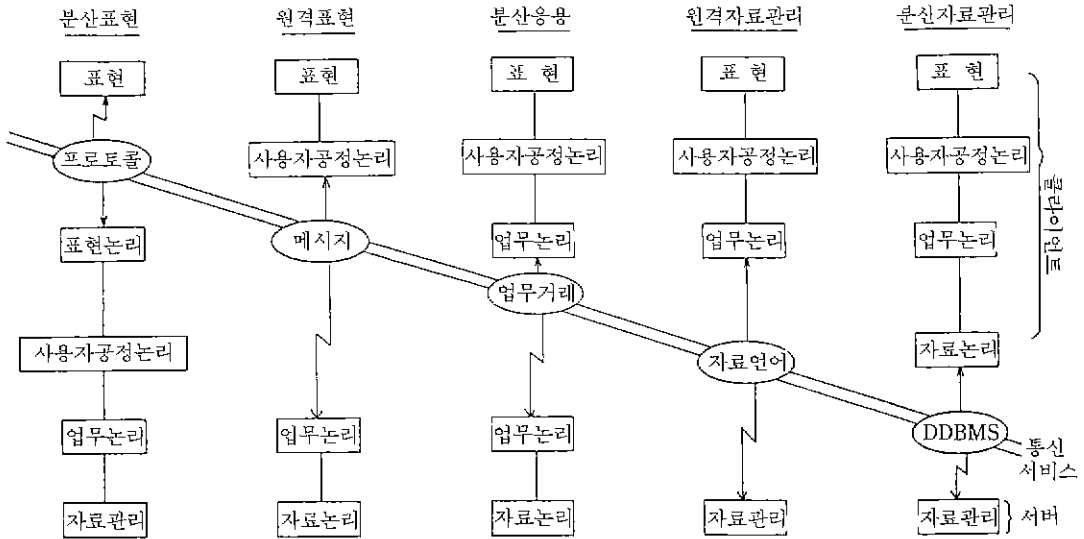
등으로 구분하여 나타낼 수 있다.

정보시스템 설계에서 클라이언트/서버 환경에 적용하기 위한 업무활동의 절차 요소(Procedure Elements)에 대한 분할 및 분산의 일반적인 구분은

- 표현논리(Presentation Logic)
- 업무논리(Business Logic)
- 데이터베이스 논리(Database Logic)

의 세 논리이며, 이중 데이터베이스 논리는 실제로 자료관리(Data Management)나 자료접근(Data Access)이 된다. 표현논리는 실제로 시스템이 최종사용자들에게 어떤 "표현"을 제공하거나 반대로 최종사용자들이 시스템을 통하여 어떤 "표현"을 하고자 하는 논리이다(그림 3-14 참조).

* 중신회원



〈주〉 - : 단일 플랫폼 내의 통신 자료 : 가트너그룹
 <-> : 여러 플랫폼 간의 통신

그림 3-14 클라이언트/서버 분산 모델

분산처리 시스템은 다음과 같은 관점과 목적에서 다루어 진다.

3.6.1 분산시스템을 보는 세가지 방향

가. 공유의 정도

- 완전 독립
- 데이터의 공유(프로그램은 각각 중복되어 보관)
- 데이터와 프로그램의 공유

나. 데이터 사용 방법

- 사용자의 사용방식이 고정적인 경우
 - 사용자의 사용방식이 가변적인 경우
- 다. 데이터 사용방법에 대한 설계자의 인지도
- 정보가 없는 경우
 - 필요한 모든 정보를 가진 경우

3.6.2 분산의 목적

가. 편재적 처리

- 자주 참조하는 자료들은 가까이 둬으로써 통신비 절감 효과를 가져오며, 응용을 간단화 할 수 있다.

나. 분산자료의 유용성과 신뢰성

- 자료를 중복해서 여러곳에서 액세스 함
 - 여럿중에 하나만 액세스할 수 있으면 됨.
- 다. 부하(load) 분산

- 여러시스템을 사용해서 분산

라. 저장 비용과 유용성

- 각 사이트의 저장용량이나 비용에 맞게 분산

3.7 분산시스템 설계

분산시스템의 설계와 관련하여 설계의 방식, 설계의 쟁점, 분산 이유, 분산의 단위 그리고 분산의 단점 등을 간단히 살펴본다.

3.7.1 세가지 설계 방식

가. 하향식 설계 방식

아무것도 없는 상태에서 설계를 시작할 경우 적합하다.

나. 상향식 설계 방식

주어진 데이터베이스들을 통합할 경우 적합하며, 특히 서로다른 DBMS(Heterogeneous DBMS)일 경우에 유리하다. 통합은 각 데이터

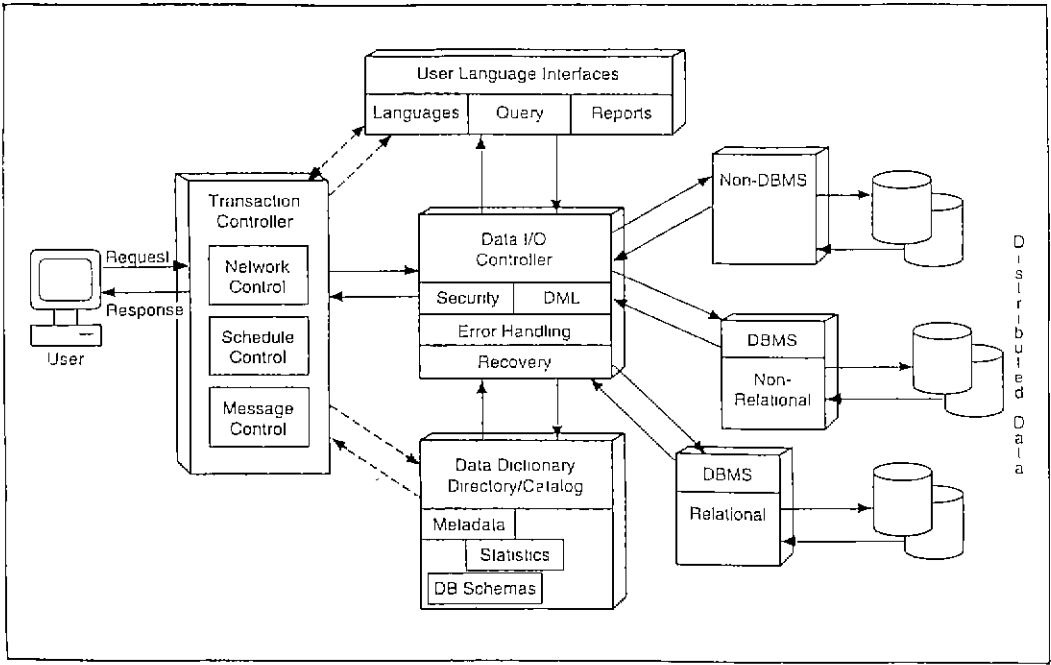


그림 3-15 ANSI 분산구조[17]

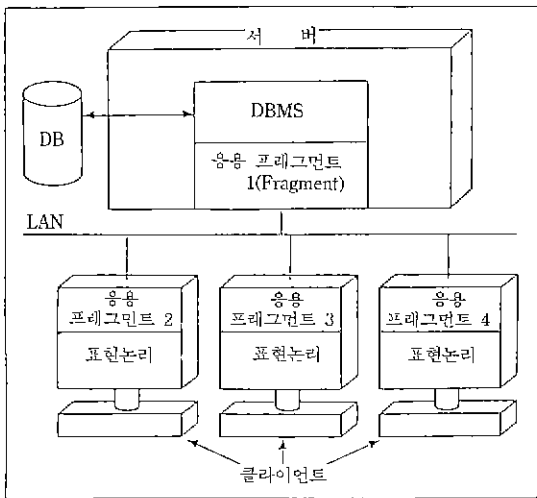


그림 3-16 분산처리[18]

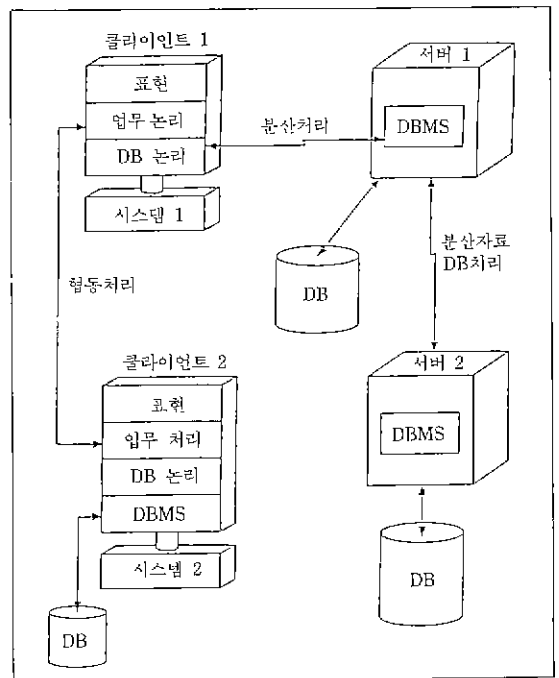


그림 3-17 C/S 분산과 협동처리[19]

베이스의 개념적 스키마에서 시작한다.

다. 하이브리드(Hybrid) 방식

주어진 서로 다른 데이터베이스들을 통합할 경우 적합하며, 특히 서로다른 DBMS일 경우에 유리하다. 아주 복잡해서 상향식 통합(bottom-up integration)이 힘들 때 사용한다.

3.7.2 분산 설계의 쟁점

— 왜 분산하는가?

- 어떻게 분산하는가?
- 얼마만큼 분산하는가?
- 분산의 정당성을 검증할 수 있는가?
- 어떻게 배치하는가?
- 분산과 배치에 필요한 정보는 무엇인가?

3.7.3 분산 하는 이유

시스템의 성능 향상과 큰 데이터베이스를 복수의 작은 시스템에서 처리할 수 있기 때문이다.

3.7.4 분산의 단위

보통 릴레이선의 전체가 아닌 일부분이 분산의 단위가 되며, 특히 응용은 릴레이선의 일부를 대상으로 한다. 만일 릴레이선을 단위로 분산할 경우 모든 사이트에 중복 보관한다면, 필요없는 중복이 되고, 한곳에만 보관한다면 교통량 증가 문제가 발생한다. 릴레이선의 일부가 분산 단위로 처리될 때 복수의 트랜잭션이 동시 수행가능 (intraquery concurrency) 하다.

3.7.5 분산의 단점

하나 이상의 분산을 필요로 하는 응용에 불리하며 자료의 무결성 제약조건의 검증이 어렵다.

3.7.6 분산 시스템 설계시 유의점

분산 시스템을 설계 할 때의 유의점으로는 다음과 같은 사항들을 고려할 필요가 있다.

가. 투명성(Transparency)

분산 시스템을 설계함에 있어서 표 3-5에서 보는 바와 같이 서로 다른 5가지 투명성들을 고려할 필요가 있다.

나. 유연성(Flexibility)

유연성을 확대시키는 방법으로는 운영체제(OS)에 망연결 기능과 통합된 원격 서비스 기능을 부가하여 성능향상을 꾀하든가(Monolithic Kernel), 사용자 수준 서버(User-Level Server)로 운영체제가 서비스하는(Micro Kernel) 방법 등이 있으나 후자가 보다 더 유연성을 가지며 대부분의 분산시스템은 이 방법으로 설계하고 있다.

다. 신뢰성(Reliability)

분산 시스템의 궁극적인 목적 가운데 하나는

표 3-5 시스템에서의 투명성[24]

투명성 구분	의 미
위치 (Location)	사용자는 사용하고자 하는 자원들이 어디에 있는지 알 필요가 없다.
이식 (Migration)	자원들은 그들의 이름 변경 없이 식할 수 있어야 한다.
중복 (Replication)	사용자는 얼마나 많은 복제가 있는지 알 필요가 없다.
동시성 (Concurrency)	다수의 사용자들은 자동적으로 자원을 공유할 수 있다.
병행성 (parallelism)	사용자가 알 필요없이 행위(Activities)가 병행적으로 발생할 수 있다.

분산되어 있는 처리기능을 통하여 단일 프로세서 시스템에 비하여 더 큰 신뢰성을 얻는데 있으며, 신뢰성에 영향을 미치는 요소로는 가용성(Availability), 보안성(Security), 결합허용(Fault Tolerante) 등을 들 수 있다.

라. 성능(Performance)

분산 시스템은 단일 시스템에 비하여 통신 속도가 느리므로 상당부분 통신속도에 의존하는 연산속도를 향상시키기 위하여는 병행처리 기술을 사용해야 한다.

마. 확장성(Scalability)

시스템 설계자들은 거대한 분산 시스템에서는 집중화된 구성요소(Components)라든가 표(Tables), 알고리즘(Algorithms) 등을 피해야 한다. 집중화는 시스템을 경직되게 만들 염려가 있고, 약간의 고장으로라도 전체가 마비될 수 있으며 결과적으로 융통성이 결여되기 때문이다.

바. 분산관리(Distribution Management)

분산환경하에서 나타나는 여러가지 시스템 관리상의 문제, 특히 보완(Supplement)과 예비조치(back-up) 등에 대한 상세한 계획의 수립이 요구된다.

4. 개방 접속 환경

4.1 망 연결(NETWORKING)

클라이언트/서버는 망 연결에 의하여 클라이

외부적 요소로는 생산현장과 지점 등의 사용자들과 거래처 고객 등을 들 수 있고, 이의 관련 매체요소로서 Email, EDI, Dial-up, SNMP, ISDN, B-ISDN 등을 들 수 있다.

한편 망을 효과적으로 관리하기 위하여는 NMS(Network Management System)가 필요하며, 망 연결에서 요구되는 사항들은 다음과 같다.

- 분산된 장소나 원격지 시스템의 상호접속
- 투명성(Transparency)
- 높은 신뢰성, 안정성
- 기밀 보안성
- 고속통신 기능
- 멀티미디어 관련 기능
- 유연한 확장성, 호환성
- 무정지(無停止) 연속 가동성
- 결합허용(Fault-Tolerant) 기능
- 망관리 시스템(NMS)
- 표준화된 개방형
- 구축의 용이성, 경제성
- 유지보수 및 관리의 용이성, 경제성

4.2 망 동향

그 동안 발전을 거듭해온 통신부분은 80년대에 들어와서 전자, 컴퓨터 부문의 급속한 발전의 영향으로 비약적인 발전을 이룩했다. 이를 몇가지 사항에서 비교해 보면 표 4-1과 같다.

4.3 기존 망의 문제점

표 4-1 망 동향

80년대 이전	80년대 이후
모뎀 (analog-to-digital)	ISDN 인터페이스 (all-digital world)
업체별 네트워크 프로토콜	OSI, TCP/IP (open standard)
부분적으로 분산된 망	완전히 분산된 망
16 Mbps 이하	100 Mbps 이상
망을 통하여 텍스트 자료 전송	텍스트 이외의 다양한 정보 전송

기존의 망들은 각 메이커 중심의 하드웨어나 소프트웨어에 종속되므로 투명성이 미흡하며, 다른 망과의 접속이 곤란한 경우가 많았다. 결국 각 시스템은 독립적으로 존재하게 되므로 자료와 프로그램을 공유하는데는 그만큼 제한적일 수밖에 없었다. 각 업체간에서는 시스템에 관한 정보의 교류가 단절된 상태이고, 각 시스템별로 주요부분의 전문가가 항상 필요했다. 또한 각 기업은 시스템에 대한 투자에 비하여 기대에 부응하지 못하므로 만족하지 못하는 경우가 많았다.

4.4 개방 시스템

클라이언트/서버 환경 구축에서 가장 중요한 것 가운데 하나가 통신망 구축이다. 구축된 통신망은 시스템의 내부와 접속되어야 하며 외부의 다른 시스템이나 다른 망과 연결되어야 한다.

다른 시스템이나 다른 망과 연결시키기 위하여는 서로 불편없이 연결되어야 하고, 서로간의 통신에 아무런 문제도 발생하지 않아야 하며 고품질의 통신으로 신뢰성이 보장되어야 한다. 이를 위하여 “개방형 시스템간 상호접속”(Open System Interconnection : OSI)이라는 개념이 등장하게 되었다.

이는 통신의 개념을 7층구조(7 Layer Architecture)로 정의하고, 통신하고자 하는 두 주체(사람, 컴퓨터, 단말기 등)는 같은 층으로서 동일층(peer-to-peer)끼리는 그 층의 “프로토콜”로, 인접한 상하계층간에는 하층으로부터의 “서비스”를 통하여 통신이 이루어진다.

모든 시스템이 OSI규약에 따른다면 결국 하나의 표준으로 통일되고 시스템간의 통신은 투명적(또는 투과적 : Transparent)인 기능이 향상될 것이다.

그러나 OSI는 너무 그 대상이 방대하고 세계적인 표준으로서 완성되기에는 시간이 많이 필요하므로 시스템 생산자들은 보다 간편한 SNMP(Simple Network Management Protocol)를 사용하고 있으며, 이의 대표적인 예로써 TCP/IP를 들 수 있다.

다음 그림 4-2는 OSI 참조 모델(Reference

Model)을 나타낸 것이며, 그림 4-3은 각층의 구조와 기능을 알기 쉽게 나타낸 것이다. 그림 4-

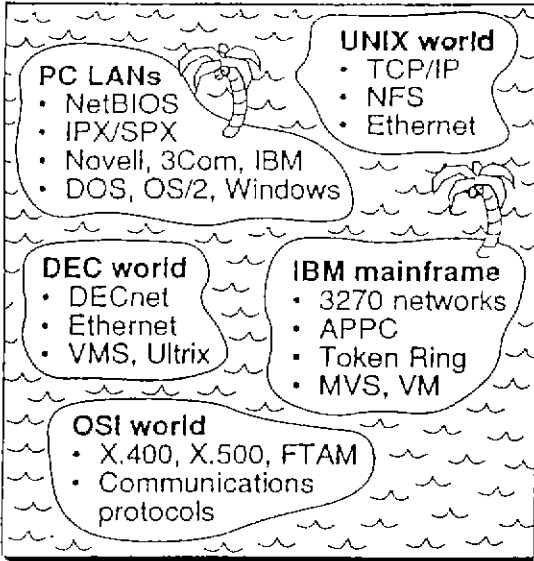


그림 4-1 독립적으로 존재하는 시스템들

4는 OSI와 TCP/IP를 대응시켜 그 기능이 서로 같은 층을 보인 것이다.

그림 4-5는 실제의 이해를 돕기 위하여 Microsoft사의 접속 프로토콜을 예시한 것이며, 그림 4-6은 각층의 해당 프로토콜의 예를 들었고, 그림 4-7은 망 표준들의 계층구조를 비교 예시하였다.

그림 4-8과 그림 4-9는 TCP/IP 표준과 OSI 표준을 이용한 다양한 기기종간의 연결을 예로서 보인 것이다.

5. 클라이언트/서버 활용

5.1 트랜잭션 처리

트랜잭션 처리(Transaction Processing : TP)의 기본 동작은 클라이언트가 서버에게 “요청(Request)”를 보내면 이 “요청”은 “요청대기(Request Queue)”에 들어가게 되는데 이것이 하나의 트랜잭션 단위가 되고, 다음 동작으로 서버가 “요청대기”로부터 “요청 접수”를 받고

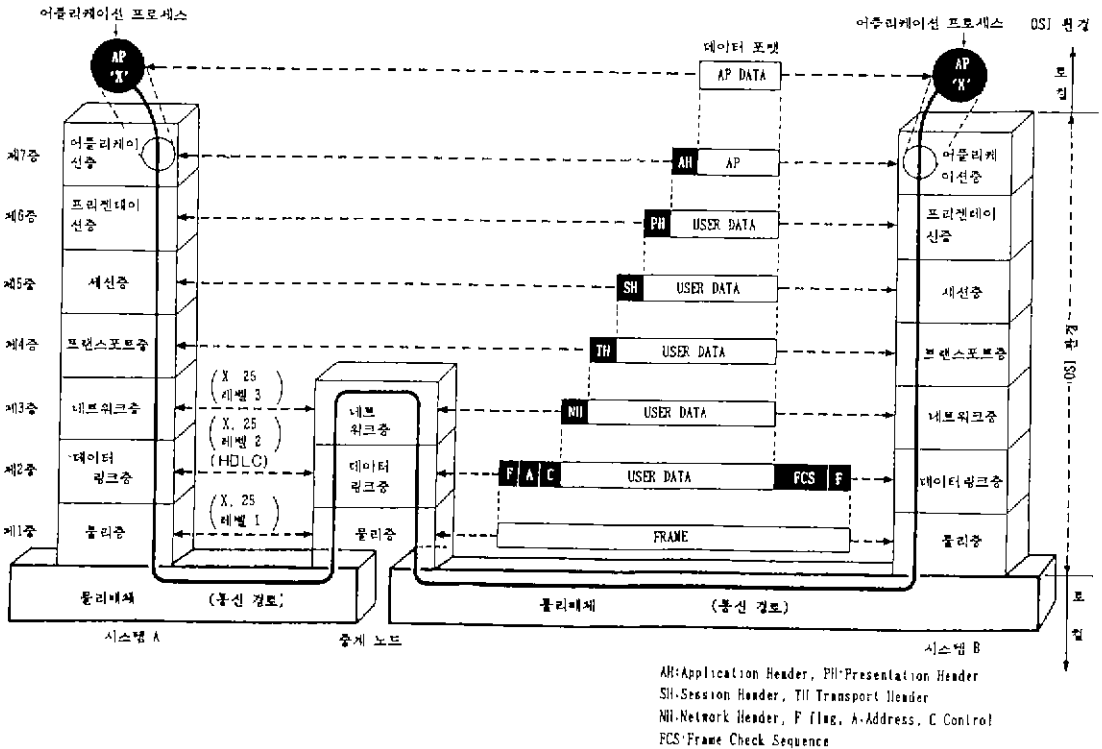


그림 4-2 OSI 참조 모델[25]

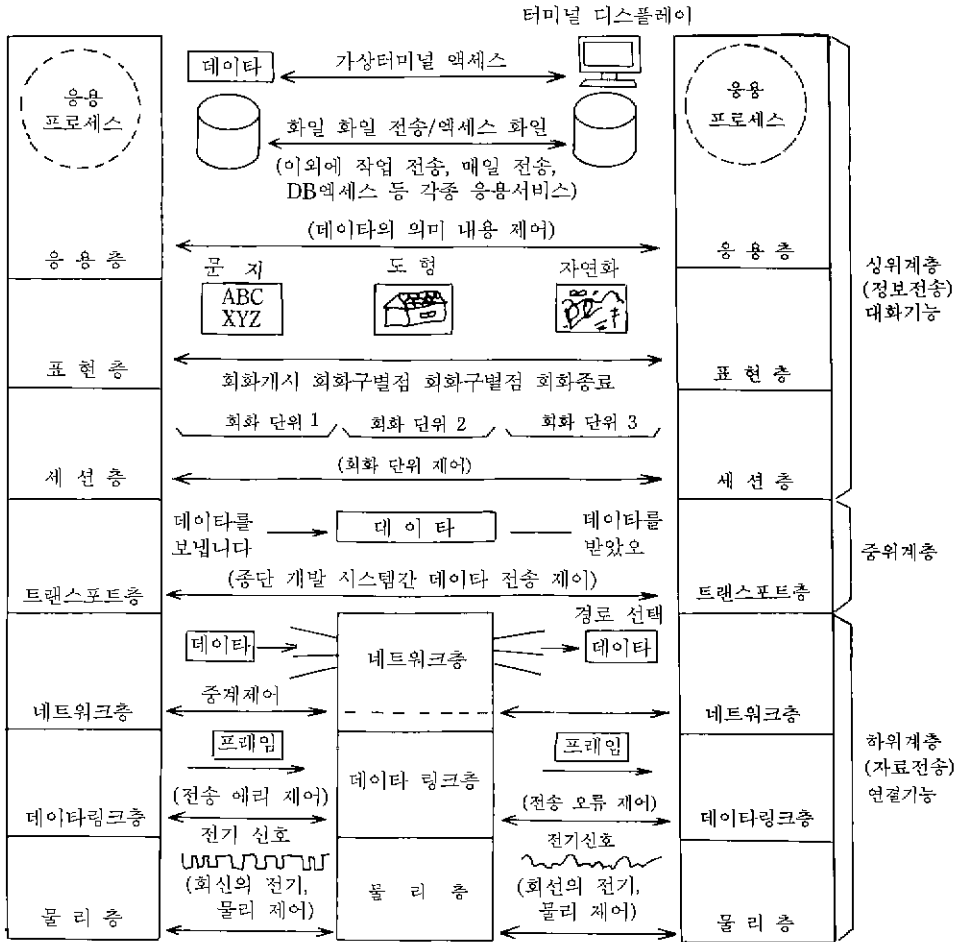


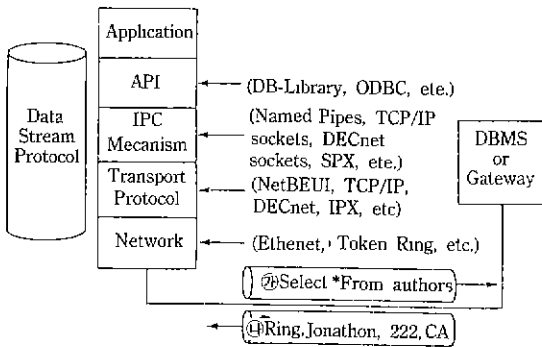
그림 4-3 OSI 참조 모델 7층의 기능[26]

OSI 참조모델		TCP/IP와 각종 서비스		
7층	응용층	SMTP, FTP, TELNET, TCP, RLOGIN (ARPA 서비스) (Berkeley 서비스) 퍼스널 커넥션 NFS		
6	표현층			
5	세션층	소켓 인터페이스	NETBIOS	
4	전송층	TCP	UDP	
3	네트워크층	IP (ICMP) (ARP)		
2	데이터링크층	이더네트	X 25	기타
1	물리층	동축선	전회회선	기타

[주] 약어

- OSI(Open Systems Interconnection)
- SMTP(Simple Mail Transfer Protocol)
- FTP(File Transfer Protocol)
- TELNET(미 국방성(DOD)가 정의한 기성단말 서비스)
- rcp(rcremote copy)
- rlogin(remote login)
- ARPA(Advanced Research Projects Agency)
- NFS(Network File System)
- NETBIOS(Network Basic Input Output System)
- TCP(User Datagram Protocol)
- UDP(User Datagram Protocol)
- IP(Internet Protocol)
- ICMP(Internet Control Message Protocol)

그림 4-4 OSI와 TCP/IP 통신



[주]

- API : Application Program Interface
- IPC : Internet Protocol Control
- TCP/IP : Transmission Control Protocol/Internet Protocol
- SPX : Sequenced Packet Exchange
- NetBEUI : Netbios Extended User Interface
- IPX : Internet Packet Exchange
- S : 자료검색을 위한 명령의 예
- R : 응답(검색된 자료)의 예

그림 4-5 클라이언트/서버 인터페이스 구조 예[27]

“응답대기(Response Queue)”에 “응답”을 넣게 되는데 이것이 다른 하나의 트랜잭션 단위이다.

그리고 마지막 동작으로 “응답대기”로부터 클라이언트가 “응답접수(Receive Response)”를 받게 되는데 이것이 또다른 트랜잭션단위가 된다.

그림 5-2는 통합제어(Integrated Control)가 트랜잭션 서비스에 의하여 수행되고 있음을 보여준다. 여기서 “트랜잭션 프로그램”은 2개의 데이터베이스와 원격 서버, 그리고 약간의 응용 종속 문맥(Application-dependent Context)을 유지하며 윈도우 환경을 가진 워크스테이션에 관련하고 있음을 보여준다.

그리고 이들의 상호작용은 자원할당 관리 인터페이스와 절차(Procedure)들로 구성된 프로그램 즉, TP 모니터에 의하여 효율적으로 수행 된다.

OSI Layer	Example ISO Protocols			
Application	ISO 9040/9041 VT	ISO 8831-8832 JTM	ISO 8571/8572 FTAM	ISO 9595/9596 CMIP
Presentation	ISO 8823/CCITT X 226 Connection-Oriented Presentation Protocol			
Session	ISO 8327/CCITT X.225 Connection-Oriented Session Protocol			
Transport	ISO 8073/CCITT X 224 Connection-Oriented Transport Protocol			
Network	ISO 8473 Connectionless Network Service		ISO 8208/CCIT X.25 Packet Level Protocol	
Data Link	ISO 8802-2			
	ISO 9314-2 FDDI	ISO 8802-3 CSMA/CD BUS	ISO 8802-4 TOKEN BUS	ISO 8802-5 TOKEN RING
			ISO7776 CCITT X 25 LAP/LAPB	ISO 7809 HDLC
Physical	Options from EIA, CCITT, IEEE, etc			

그림 4-6 OSI 각 계층의 해당 프로토콜 예

OSI	TCP/IP	SNA	HSN
Application	Application	Application	Application
Presentation		Presentaion	
Session		TCP	Data Flow Control
Transport	Transaction Control		
Network	IP	Path Control	FDDI, DQDB, or HIPPI
Data Link	Subnet	Data Link	
Physical		Physical	

- [주] • XTP (eXpress Trasfer Protocol)
 • VMTP (Versatile Message Transfer Protocol)
 • NETBLT (Network Block Transport Protocol)
- FDDI (Fiber Distributed Data Interface)
 - DQDB (Distributed Queue Dual Bus)
 - HIPPI (High Pararell processing Interface)

그림 4-7 망 표준들의 계층구조

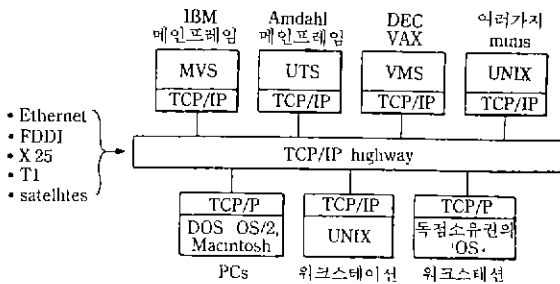


그림 4-8 TCP/IP 표준을 이용한 이기종간 연결 예

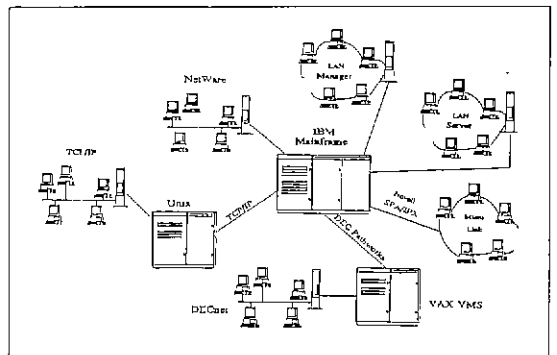


그림 4-10 망 표준들을 이용한 이기종간 시스템 구성 예

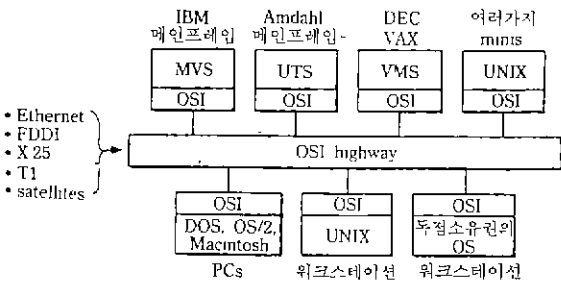


그림 4-9 OSI 표준을 이용한 이기종간 연결 예

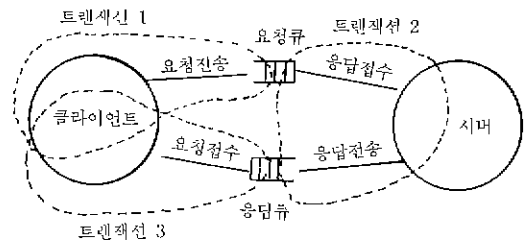


그림 5-1 트랜잭션 단위 처리[28]

TP 모니터 사용시의 잇점으로는

- ① 복수의 단말기 지원
- ② 트랜잭션 단위 처리량 향상
- ③ 장애율 감소
- ④ 응용 관리비 및 개발비 절감

트랜잭션 처리는 다음과 같이 ACID특징을

만족해야 한다.

- 원자성(Atomicity) : 트랜잭션 수행결과가 DB에 부분적으로 반영되어서는 안됨.
- 일관성(Consistency) : 트랜잭션 수행 후에

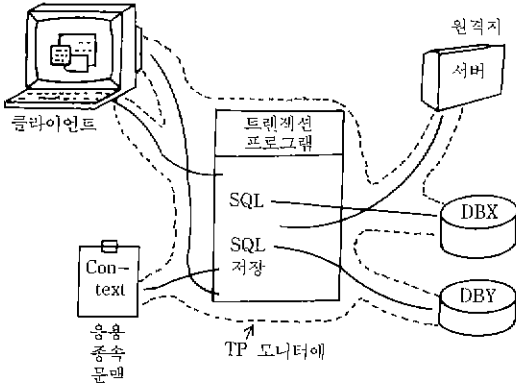


그림 5-2 트랜잭션 서비스에 의한 통합제어 수행[29]

표 5-1 클라이언트/서버 처리방식과 다른 처리방식들과의 비교

처리의 기본 5형태 비교항목	일괄처리	서분할	신시간	C/S	트랜잭션중심
지도	개별서 살다	개별적 걸나	개별적 매우 짧다	공유적 걸다	공유적 짧다
지속시간	보통	보통	매우 짧다	보통	매우 높다
신뢰성 보장	없음	없음	있음	있음(?)	ACID 4)
일관성보장	규칙적	규칙적	임의적	임의적	임의적
작업형	10'	10'	10'	10'	10'
사업자정보/목표	가상프로 세서	가상프로 세서	단순기능	단순요청	단순 요는
서비스요구	생산성	응답시간	응답시간	생산성과 응답시간	생산성과 응답시간
성능비주	유용성 이기단위	보통 삼(Job)	보통 사용자	높음 없음(?)	높음 요청자

4) Atomicity(원자성), Consistency(일관성), Isolation(불간섭성), Durability(지속성)

※위의 비교는 자원분배, 작업단위의 크기, 자료사용형태, 성능과 유용성 및 보안 등급에 의한 것임.

하는 트랜잭션 수행 요청을 신속히 처리하는 것을 말한다.

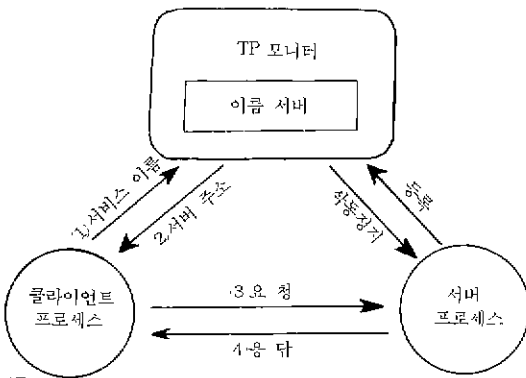
온라인 트랜잭션 처리는 다음과 같은 특성을 갖는다.

- 빠른 응답시간 : 2,3초내
- 입력의 예측 가능
- 많은 수의 짧은 대화 : 수십~수백 트랜잭션/초
- 방대한 공유 DB
- 동일작업, 처리의 빈번한 요구
- 다양한 요구사항과 완전한 처리
- 평균 트랜잭션 크기(1 트랜잭션 당)
 - 메시지 교환 : 2~20개 정도
 - CPU명령 수행 수 : 1만~100만개 정도
 - 디스크 접근 : 0~30/회 정도

OLPT의 대표적인 예로는 은행의 온라인 처리와 자동현금인출기(Automated Teller Machine : ATM) 및 비행기나 열차의 좌석예약(Reservation) 등을 들 수 있으며 그 밖에 자동현금이체, 신용조회, 무역 및 증권거래 시스템들을 들 수 있다.

최근에는 출판, 편집 지원 시스템, 자동차 등록, 판매시점정보관리(Point of Sale : POS) 등에도 활용되고, 전화국의 전화 호출교환 제어시스템이나 통신망 관리(온라인 감시, 진단, 보수 등)와 병원관리 시스템 등에도 OLTP의 개념이 널리 이용되고 있다.

5.3 원격 절차 호출



*TP모니터 : 트랜잭션 응용 프로그래밍이 트랜잭션 ACID 특성을 쉽게 구현할 수 있도록 도움을 주는 프로그램(자원할당관리 인터페이스와 절차(procedure)들로 구성)

그림 5-3 C/S 구조에서의 TP 모니터

도 DB는 계속 일관성을 유지해야 함.

- 불간섭(Isolation) : 복수의 트랜잭션이 동시 수행시 상호 간섭하지 말아야 함.
- 지속성(Durability) : 트랜잭션의 정상적 종료후 어떤 장애 발생에도 그 정상적 효과가 지속되어야 함.

5.2 온라인 트랜잭션 처리

온라인 트랜잭션 처리(On-Line Transaction Processing : OLTP)란 온라인 사용자의 요구를 만족시키고 편익을 증대시키기 위하여 파일이나 데이터베이스에 저장되어 있는 정보에 단말기를 사용하여 직접접근하는 온라인 정보시스템에 대한 응용을 말한다. 또한 온라인 사용자가 요구

원격 절차 호출(Remote Procedure Call : RPC)은 클라이언트가 대행코드(Stub Code)를 통하여 인자(Argument)를 요청, 메시지에 실어서 서버에게 보내면 서버쪽의 피호출절차(Called Procedure)에 의해 처리된 인자가 서버 대행(Server Stub)을 통하여 응답 메시지에 실려 결과로서 수신되는 통신방식이다.

- 그림 5-5중 점선위의 근거리 프로세스 호출(LPC) 내에서 호출 절차는 그 자신의 주소공간에서 절차를 수행한다.
- 점선아래 부분의 원격 절차 호출(RPC)내에서 클라이언트와 서버는 둘로 분리된 절차처럼 수행한다. 그들은 동일기계상에서 수행될 필요가 없다.
- 이 두 프로세스는 클라이언트와 서버 각각에 대한 대행(Stubs)을 통하여 통신한다. 이 대행은 일련의 망 RPC기능 호출내에서 근거리 절차 호출과 대응(map)하는 기능을 가진 코드의 한 부분이다.

***대행(Stub):**

- 자동적으로 생성
- 주 프로그램과 원격 절차 사이의 통신을 관리
- RPC Runtime Library와 함께 Network Transparency 제공
- 클라이언트 대행은 지역 절차를 대신
- 서버대행은 호출 프로그램처럼 행동(Dispatcher 기능)

원격 절차 호출(RPC)은 다음과 같은 특성으로

표현할 수 있다.

즉, 클라이언트/서버 모델의 전형적인 예중의 하나로 그 내부에 대행(stub) 코드를 갖고, 이를 통하여 클라이언트/서버간의 원격 절차 호출 기능을 수행한다. 또한 RPC Runtime Library를 갖고 있으며, RPC는 하나의 Interface Definition Language라 할 수 있고 이를 위한 Interface Definition Language 컴파일러를 갖고 있다. TLI (Transport Layer Interface)나 Socket보다 사용하기 쉬우며 망 투명성(Network Transparency)이 우수하다. 이식성이 높은 소프트웨어로서 Downsizing 도구로 활용되고 있다.

5.4 망 파일 시스템(Network File System)

망 파일 시스템(NFS)은 1984 선마이크로 시스템 회사가 개발한 Client/Server 모델 기반의 분산파일 시스템으로서 LAN상의 이기종 컴퓨터간의 분산파일 서비스를 제공한다.

최초로 4.3 BSD UNIX와 호환된 SUN O.S에 설치되었으며 O.S로부터의 독립성을 유지하기 위하여 RPC기법을 사용한다. NFS 서비스는 원격 절차 라이브러리이며, NFS에서 클라이언트가 서버에게 보내는 원격 절차 호출 메시지는 호출을 성공적으로 완료하기 위한 모든 정보가 포함 되어 있는 반면에 서버는 클라이언트에 관한 상태정보를 전혀 가지고 있지 않은 무상태성(無狀態性 : Stateless)서버이다.

NFS는 다음과 같이 두가지 작업을 수행한다.

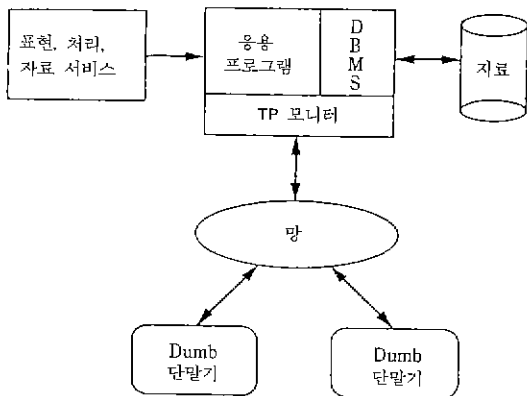


그림 5-4 호스트 OLTP 구조

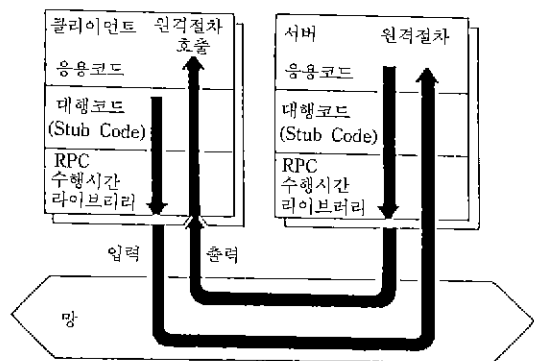


그림 5-4 RPC 기법[30]

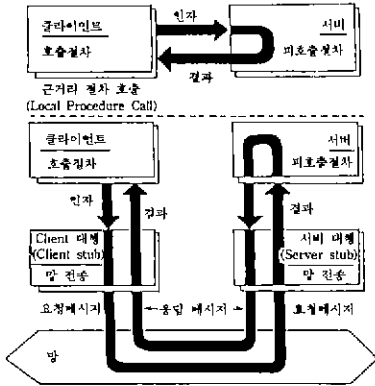


그림 5-5 근거리/원격 호출 통[31]

- Export
NFS 서버 자신의 지역 파일을 타 호스트들이 원격지에서 마운트(Mount) 할 수 있도록 개방하는 것
- Mount
어떤 파일 시스템이 다른 파일 시스템에 있는 디렉토리에 접속하는 것

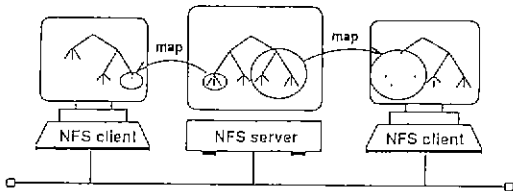
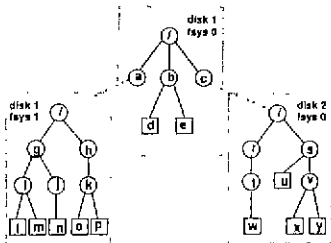


그림 5-6 NFS의 기본적인 원리[32]



- 디스크 1에 있는 파일 0의 디렉토리 a에 디스크 1의 파일 1이 마운트 되고,
- 디스크 1에 있는 파일 0의 디렉토리 c에 디스크 2의 파일 0이 마운트 됨
- Root 파일을 제외한 모든 파일은 임의의 디렉토리에 마운트 가능함.

그림 5-7 유닉스의 화일 시스템의 Mount[33]

5.5 X 윈도우 시스템

5.5.1 X 윈도우의 역사 및 배경

가. Athena project

1984년 미국 MIT와 DEC사의 공통 프로젝트로서 망에 연결된 그래픽 워크스테이션을 이용하여 원격 교육을 시도한 연구 개발이었으며, 다양한 vendor의 워크스테이션으로 구성되어야 하므로 공통의 윈도우 시스템 필요성이 대두되었다.

나. X 윈도우

개발자는 Robert Scheifler로 Stanford 대학의 W 윈도우 시스템의 영향을 받았다.

망을 통한 전송과 하드웨어나 운영체제에 독립적으로 그래픽을 제공하며 현재, X11 Release 5를 제공하고 있다.

5.5.2 X 윈도우 시스템의 특징

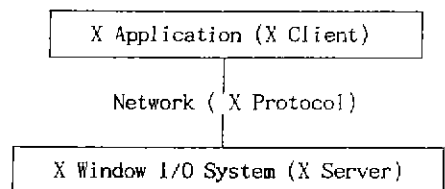
크기 조절이 가능한 계층 구조의 윈도우를 제공하며, 하드웨어나 운영체제에 독립적인 고성능 2차원 그래픽을 제공한다. 또한 소스 코드 공개 및 이식성이 뛰어나다.

망을 통한 연결이 사용자에게 투명한 환경을 제공하며, X 프로토콜은 다양한 종류의 프로그래밍 언어나 운영 체제를 사용하여 구현할 수 있다.

비동기 통신 프로토콜을 사용하며, UNIX International의 Open Look나 OSF의 Motif 등의 표준 사용자 인터페이스를 제공한다.

5.5.3 X의 구조

X 서버, X 클라이언트, X 프로토콜이 구조의 주요소이다.



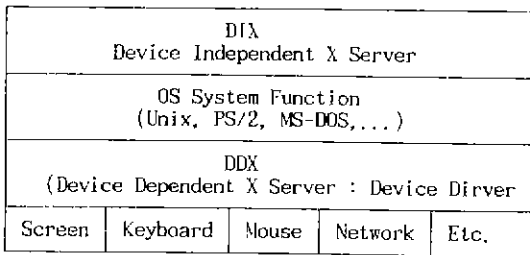
5.5.4 X 서버

X 서버는 화면, 키보드, 마우스를 관리하는

소프트웨어를 제공하며, 사용자의 키보드, 마우스 조정에 대하여 화면을 제어한다. 또한 서버는 클라이언트가 보낸 정보 요청에 대답하며, 오류 보고를 한다.

5.5.5 X 서버의 구조

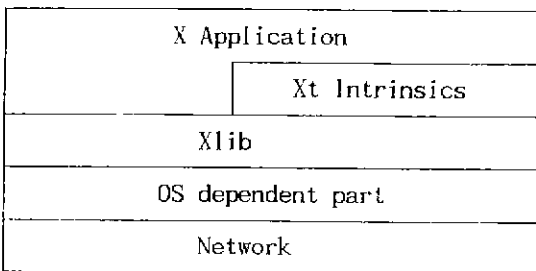
X서버는 시스템의 상위층의 장치에 독립적인 DIX와 이에 인접한 하위층으로서 OS 기능층과 여기에 인접하는 하위층으로서 장치의존적 서버인 DDX가 있고, 여기에 사용자의 스크린, 키보드, 마우스, 망 등이 접속되는 구조를 갖고 있다.



5.5.6 X 클라이언트

X 클라이언트는 화면표시 기능과 키보드 및 마우스로부터 입력을 받고 그리기 요청과 정보 요청을 서버에게 보낸다.

X 서버와 클라이언트는 동일한 기계나 망 상의 다른 기계에서 실행 가능하다.



5.5.7 X 클라이언트의 개발

C 및 Lisp 라이브러리가 제공되는 클라이언트 프로그래밍 라이브러리를 사용한다.

가. Xlib :

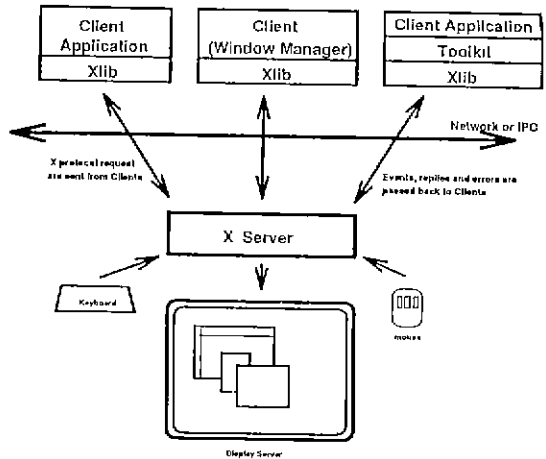
X 프로토콜에 대한 저급의 프로시저 인터페이스로서 MIT에서 제공했으며, C 언어로 된 클라이언트 프로그래밍 라이브러리로 BSD soc-

ket을 사용한다.

나. Xt Intrinsics :

객체지향 기법을 이용한 툴킷으로 사용자 인터페이스 요소인 Widget기능을 갖는다.

5.5.8 C 클라이언트 프로그래밍 라이브러리

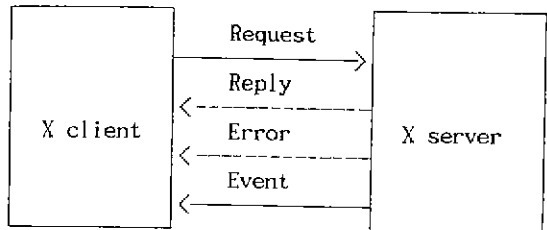


5.5.9 X 프로토콜

비동기 양방향의 바이트 스트링 전송 서비스를 기반으로 TCP/IP나 DECnet을 하부 네트워크 프로토콜로 사용한다. 하드웨어 운영체제에 상관 없는 독립적인 자료 구조를 갖는다.

5.5.10 X 프로토콜의 패킷 유형

X 프로토콜의 패킷 유형에는 request, reply, event,error의 4가지 이며, 각각의 기능은 다음과 같다.



가. Request

-클라이언트가 생성하여 서버에 보냄

-종류 :

- Line Draw
- Color Map 변경

- Window Size Query
- 기타

나. Reply

- 클라이언트의 request에 대한 응답
- request with reply에 응답

다. Event

- 디바이스 동작이나 앞 요청에 대한 side-effect에 대한 정보 포함

라. Error

- 이벤트와 비슷하거나 클라이언트가 다르게 처리
- 클라이언트 측 라이브러리에 있는 오류 처리
- 루틴에 보내짐

5.5.11 X의 미래

멀티미디어를 지원하고, 다중언어 처리를 지원하며, 3차원 그래픽 지원과 벡터폰트 지원이 가능할 것이다.

6. 정 리

앞에서 5장에 걸쳐서 살펴본 클라이언트/서버에 대한 고찰을 통하여 그 장단점과 경영참여자의 관점과 기업에 가져다 주는 긍정적 측면의 기회를 간략히 간추려 보면 다음과 같다.

6.1 클라이언트/서버의 장점

- 자료 공유성의 제고
- 종합 서비스
- 다양한 플랫폼(platforms)들 간의 자원공유
- 자료의 상호 교환 가능성과 상호 사용 가능성 제고
- 풍부한 응용들, 분산처리
- 개발자에게 기술적, 투명성(Transparency of Technology)을 보장
- 쉬운 사용, 최종 사용자 요구 만족
- 확장성, 유연성
- 저렴한 비용

6.2 클라이언트/서버 컴퓨팅의 단점

- 응용프로그램을 C/S로 나누는 도구가 상대적으로 적다.
- 단일 시스템보다 망의 부하가 클 경우도 있다.
- GUI는 그래픽 처리위주이므로 응용의 응답 시간이 늦어질 수 있다.
- 분산환경지원용 DB가 충분하지 않다.

6.3 기업내 경영참여자의 각 그룹별 관점

가. 경영진

- 전략적, 경쟁적 이점 제공
- Reengineering 업무 처리
- 전체 컴퓨팅 비용절감

나. 지식 작업자/의사결정자

- 분산자료에의 신속한 접근
- 관점에 의한 사용자 인터페이스
- 런비용 절감

다. MIS스텝

- 신속한 응용 개발
- H/W의 효율적 사용
- 병행개발
- 축소된 스텝으로 증가된 출력
- 응용의 유연성

6.4 클라이언트/서버로부터 오는 기업의 기회

- 비용절감
- 전략적 위치선정
- 경쟁에서 유리한 입장
- 새로운 서비스
- 최종 사용자의 만족
- 조직의 재설계
- DB S/W가 여러 시스템에 산재할 수록 자료의 무결성 문제발생 가능
- 시스템 보안과 자료의 정확성 문제 우려

7. 결 언

세계적인 경기침체와 선진국의 기업경영 여건의 악화는 획기적인 기업 생존 전략의 수립을 재촉하였고, 최근에는 이를 뒷받침하는 리엔지

니어링(Reengineering), 벤치마킹(Benchmarking), 다운사이징(Downsizing) 등의 최첨단 경영이론들이 등장하였다.

여기서 컴퓨터와 통신부분의 급속한 발달은 신경영체제로의 전환에 새로운 구현 도구로써 그 역할이 증대되었고, 구체적으로 고기능의 다양한 지능단말기들을 통신망에 연결함으로써 각종 응용의 분산화와 최종사용자 중심의 시스템이 구현되어 메인프레임의 퇴조가 가시화 되었다.

클라이언트/서버 시스템은 실제로 비용절감 효과가 뚜렷하며, 기업환경 변화에 유연하게 대처할 수 있는 바람직한 방법이며, 피할 수 없는 추세이다. 결과적으로 기계중심적 시스템에서 인간중심적 정보화를 실현할 수 있는, 현재로서는 이상적인 기업정보화 방법이다.

그러나 클라이언트/서버 환경에서도 호스트 컴퓨터(Host computer)는 필요하며, 다만 그 기능과 규모가 다르게 되는 것 뿐이다. 특히 대량의 자료 처리를 위한 응용 서버로서 알맞다. 또한 GUI나 CASE 도구(Tool)를 사용하면 개발과 이용에 효과적인 것은 분명하나 즉시 효과를 보는 것은 아니다. 도구들은 도구로서 그치지 않고 때로는 방법을 유추할 수 있는 수단이 되기도 하므로 사용자의 자세가 중요하다.

클라이언트/서버 시스템의 생산성을 향상시키기 위하여는 GUI, 4GL, 객체지향(Object-Oriented), 멀티미디어 등의 새로운 도구와 방법들을 적극적으로 사용하고 여러가지 방면에 활용하도록 노력하는 자세가 필요하다. 현재의 여건을 감안하여 볼 때 장차 클라이언트/서버의 환경은 다음과 같은 특성을 갖될 것으로 예견된다.

- 이기종(Heterogeneous)간의 상호작용
- 분산형태의 확대
- 지능적 기능의 증대
- 객체지향적(Object-oriented) 프로그래밍의 보편화
- 멀티미디어 기술의 증대
- 사용자들의 고성능 요구
- 중간매체(Middleware)의 사용 용이

다운사이징이나 라이트사이징은 보다 비즈니스적인 면을 강조한 용어로서 조직이나 기업내 처리과정을 어떻게 간소화하고 최적화하느냐의

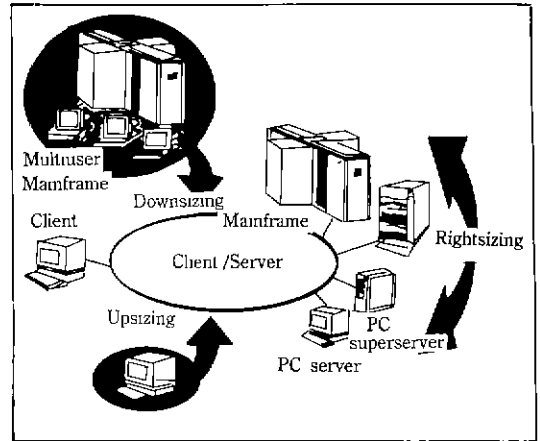


그림 6-1 Downsizing, Upsizing, Rightsizing

문제이다.

지금까지 살펴본 내용들은 총론적, 개괄적인 측면에서 기술된 것이다. 각 기업이나 기관에서는 각각의 입장에서 그 환경과 여건을 고려하여 각론적으로 접근해야 할 것이다.

참고문헌

- [17] ~[22] Alex Berson, *Client/Server Architecture*, McGraw-Hill, PP. 36~383, 1992.
- [23] C. J. Date, *An Introduction to Database System*, Vol. 1. 5th Ed., Addison Wesley, Inc., PP. 620~627, 1980.
- [24] Andrew S. Tanenbaum, *Modern Operating System*, Prentice-Hill, Inc, PP. 362~393, 1992.
- [25] 小野欽司外 6人, *OSI 프로토콜繪とき讀本*, オム社, P. 34, 1990.
- [26] 조국현외, *컴퓨터 네트워크[OSI]*, 정익사, P. 36, 1989.
- [27] Microsoft社 "Designing Client-Server Application for Enterprise Database Connectivity", Technical note, Microsoft Co., Vol. 3, No. 14, PP 3~5, Jun. 1992.
- [28] [29] 문송천, "Client/Server 컴퓨팅 환경의 국내구축 방법", ITB '93, SDS, PP. 10~14, 1993.
- [30] John Shirley, *Guide to Writing DCE Applications*, O'Reiley & Associates, PP. 1~20, 1992.
- [31] John Bloomer, *Power Programming with RPC*, O'Reiley & Associates, PP. 1~5, 1991.

[32] [33], 서울대학교 하계강좌 자료, Client/Server
구축 방법론(TCP/IP), 자연과학종합연구소, 19
93.

이 기 현



- 1965 성균관대학교 경상대학
경제학과 졸업
- 1972 성균관대학교 경영대학
원 정보처리전공(경영학
석사)
- 1976 총무처 행정전산계획실
전산처리관
- 1981 대한손해보험협회 전산
실장
- 1986 광운대학교 대학원 전산
학과(이학석사)

- 1993 광운대학교 대학원 전산학과(이학박사)
 - 1982 ~ 현재 명지대학교 공과대학 전자계산학과 교수
동 대학교 전자계산소장
대신통상(주) 고문
한국정보과학회 학회지편집위원회 부위원장
한국정보처리응용학회 부회장
IEEE 정회원
- 관심 분야: 컴퓨터네트워크, 소프트웨어 공학, 운영체제

'94 인공지능연구회 춘계학술발표대회 논문모집 안내

- 행사일자 : 1994년 5월 21일(토)
- 행사장소 : 아주대학교 제 2이공관
- 논문마감 : 1994년 4월 29일(금)
- 연 락 처 : 아주대학교 컴퓨터공학과

김 민 구 교수

TEL. (0331) 219-2437

FAX. (0331) 213-8284