Super Personal Computer의 動向

—Stephen A. Wood 美 Massachusetts 工科大學 敎授—

1. 序 言

Perscom에 대한 특히 앞으로의 動向에 대해 언급하고자 한다.

첫째, 말하고자 하는 순서로 Perscom의 歷史的인 根幹 및 그 背景으로부터 시작하여 다음으로 基本的이며 더욱이나 중요한 문제로 앞으로의 컴퓨터가 발전해 나아가는 가운데서 어떤 문제가 제기될 것인가에 대해 말하고자 한다. 그리고 그 다음 이들 문제에 對處하기 위한 현재 행해지고 있는 硏究作業중에서도 高레벨의 Approach로써 특히 두 개의 프로젝트를 설명하려고 한다.

双方이 다같이 MIT의 프로젝트이지만 그 하나는 뉴 머신, 뉴 퍼스널 컴퓨터의 프로젝트이며 또 하나는 TRIX라고 하는 프로젝트로 퍼스널 컴퓨터의 Operating System을 위한 프로젝트이다.

끝으로 Super Personal Computer의 現狀과 앞으로의 動向 등에 관해 말하고자 한다. 아는 바와 같이 퍼스널 컴퓨터는 반드시 體系的인 계획에 바탕을 두고 發展해 온 것은 아니며 오히려 어떤 技術이 그 段階에서 入手 가능한 技術을 이용하는가 하는 形態로 발전해 왔다.

퍼스컴의 메이커는 언제나 여러 가지 壓力을 받고 있어 새로운 技術, 새로운 言語, 새로운 IC, 새로운 Software가 나오면 어쨌든 그것을 사용하지 않을 수 없다. 즉 檢討할 여유도 없이 또한 長期的인 展望이나 目標를 세울 수도 없이 그 때의 技術에 쫓겨 오고 있었다고 할 수 있다.

그와 같은 발전의 根幹 그 자체는 예를 들면 Main Frame으로 사용되고 있는 Patch System의 技術 등에 바탕을 두고 있었으나 이제부터의 分野는 아직 完成된 것으로는 되어 있지 않고 있다. 또한 이 밖에 Time Sharing 技術, 그리고 미니 컴퓨터, Local Network, 오락용 컴퓨터 등의 技術도 根幹이 되고 있다.

그 段階에서 눈에 보이는 短期的인 목표에 영향을 받아 왔으며 가장 두드러진 목표로는 旣存의 大型 컴퓨터를 들 수 있다.

여기에서 우리들이 自問自答하지 않으면 안되는 것은 大型 컴퓨터를 模倣하는 것이 퍼스컴의 目的인가 하는 것이다. 즉 퍼스컴은 大型의 Main Frame Computer의 模倣이 되어도 좋은 것인가. Time Sharing System이나 그 밖의 情報處理 시스템 등을 模倣하는 것만으로 좋을 것인가 하는 것을 질문하지 않을 수 없다. 現實的으로 퍼스컴이 사용되는 狀況이란 것은 大型의 Main Frame Computer가 이용되는 狀況과는 전혀 다른 것이다. 그러므로 우리들이 未來를 檢討할 때에는 Main Frame과 Perscom의 發展의 方向 및 그들의 根幹을 전혀 別個의 것으로 보아야 하지 않을까 생각된다.

2. Communication과 Computation

컴퓨터, 특히 퍼스컴의 앞으로 演算(Comput-

ation)用의 機能 즉 Main Frame의 Patch 處理나 Time Sharing System 등의 演算用의 機能은 아니며 Communication Resource로 보아야 하지 않을까 생각된다. 理由는 몇가지 있으나 그 가운데 하나로 演算機能의 地理的인 分散을 들 수 있다.

즉, 퍼스컴의 出現으로 지금까지 集中化되었던 것이 分散化된 것이다. 옛날의 形態를 벗어나 小型이 도처에 分散設置되어 시스템으로서 導入되게 되었다.

원래의 목표는 小型의 컴퓨터 全部로 集中化된 大型으로 代替케 하는 즉, 同等의 機能을 遂行케 하자는 것이었다. 그러므로 이 地理的인 分布라는 것이 첫째 이유이다. 다음으로 퍼스컴이 地理的으로 分散된다는 環境에서는 컴퓨터의 管理도 分散되게 된다. 즉 예를 들면 담당자나 經營者 등이 있는 큰 會社에서 IBM이나 DEC, NEC를 購入할 것인가 하는 決定이 구분되어진다. 最高經營者가 아닌 오히려 各部 또는 各課라는 담당 部署가 政策 決定이나 機種 選定을 하게 된다. 따라서 政策 決定 또는 決定을 내리는 Mechanism그 자체도 分散化된다는 것이다. 다만 결점으로는 그 決定 등이 整合性이 없다는 것이다.

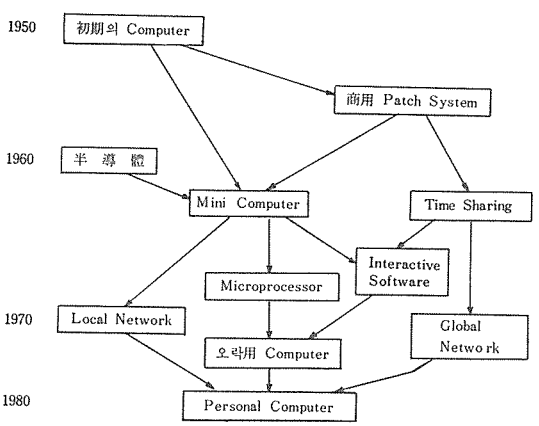

그림 1.

끝으로 중요한 理由는 퍼스컴은 通常의 경우 最終使用者가 직접 사용한다는 것이다. Main Frame의 大型 컴퓨터의 경우 典型的인 例로 周圍에 Programming, Operating Interface등의 專門 팀이 있어 그들은 컴퓨터의 여러 가지 特異한 側面 등을 熟知하고 있다. 즉, Main Frame Computer의 경우 사용자와 컴퓨터間에 專門家가 있게 된다. 그러면서도 퍼스컴의 경우에는 Main Frame과는 달리 實際의 사용자가 接觸하는 「實際 사용자」라는 것은 컴퓨터의 專門家는 아니다. 그와 같은 人間이 컴퓨터를 購入하여 직접 取扱하지 않으면 안 된다는 다른 점이 있다.

그리고 Communication과 Computer의 문제에 있어서는 두 개의 區分이 있다. Man Machine Communication 즉, 人間과 機械의 Interface와 Machine Machine Commanication 즉 機械間의 Communication이다. 이 Communication의 두 개의 分野를 묶어서 취급하려고 한다. 물론 각각의 分野에서의 問題 解決 方法은 여러 가지 형태로 나타날 것이지만 基本的으로는 共通點이 있다고 생각된다.

Man Machine Communication에 관한 단적인 예를 들어 본다면 예컨대 어떤 사용자가 Application Program을 컴퓨터에 入力시켰다고 하자. 그리고 자기의 Application을 適用 시키려고 하지만 컴퓨터의 취급 方法에 따라서는 予期치 않았던 문제가 나오므로 해서 그結果, 다소 컴퓨터로서는 適切했는지 모르지만 사용자에 있어서는 그다지 쓸모 없는 어떻게 할 수도 없는 Response가 되어 돌아오게 될는지도 모른다. 이것은 Man Machine Communication 의 문제에 있어서의 단순하면서도 極端的인 예이다.

Machine Machine Communication의 문제에 있어 그와 類似한 예가 있다. 바로 2値 情報에 대해 두 개의 다른 메이커의 機種을 想定했을 경우 긴 데이터 가운데서는 情報의 byte의 配列取扱이 다를 때가 있다. 예를 들면 IBM과 DEC의 機械에서는 典型的인 예로 低레벨의 데이터로는 전혀 互換性이 없는 形態를 취하고 있다는 것이다. 그러므로 機械間의 Communication도 반드시 成立되지 않는다.

Man Machine Communication, Machine Machine Communication 双方에 있어서는 역시 가장 強力하고 가장 適力한 Communication M-

odel을 作成치 않으면 안 된다. 즉 人間과의 Interface의 문제이며 最終 사용자에 있어서는 이해하기 쉽고,쉽게 가까이 할 수 있고,많은 경우 그날 그날의 일과 관계가 깊은 것이 좋은 모델인 것이다.

Interface가 나쁘다는 것은 사용자가 여러 가지를 이해할 수 있도록 강요하는 것이다. 컴퓨터의 모든 機能이나 컴퓨터의 構造 등을 이해하지 못하면 사용할 수 없는 것들을 말하는 것이다. 즉 컴퓨터의 構造 그 자체를 모르면 사용할 수 없게 되는 모델은 바람직하지 못하다는 것이다.

좋은 Interface의 例로는 Spread Sheet가 있다. 具體的으로는 有名한 VISICALC을 들 수 있다. 그리고 Display Window에 대한 例는 뒤에서 說明하기로 한다. 끝으로 Human Interface의 理想이라고도 말할 수 있는 것은 自然言語이다. 만약 컴퓨터 시스템을 이용하는 데에 있어 人間과 말하는 것과 같은 Communication의 形態로 컴퓨터와의 對話가 가능하다면 그것이 바로 理想的인 Interface이다.

Xerox社의 파로알트研究所가 개발한 모델은 대단한 인기를 얻고 있어 Man Machine Interface의 좋은 例이다. 最終 사용자가 사용하는 모델로 책상 위를 想定하면 된다. 이것은 책상 위에 여러 가지 종이가 놓여 있어 필요한 종이를 아래에서 꺼내서 제일 위에 놓는 方法에 따라 그 format을 결정한 것이다. 즉 사용자가 日常業務에 當面하고 있는 狀況을 모방하여 컴퓨터의 format을 결정하는 예로 매우 좋은 예이다.

Machine Machine Communication의 關鍵은 강력하고 간략한 Communication Model을 作成하는 것이다.

일반적으로 最終 사용자는 매우 높은 레벨의 Object의 表現方法을 가지고 있다. 여기에 대해서는 뒤에서 말하기로 한다.

Communication Model은 협소한 Application 분야에 限定되는 것은 아니고 相異한 분야에도 적용될 수 있는 즉, Communication의 制限外의 範圍를 갖는 것이 중요하다.

또한 Communication의 Model은 충분히 柔軟性을 가져야 한다. 즉 予期치 못했던 發展, 또한 展望 등이 기대될 경우 予想外의 요구에도 對應할 수 있는 Flexibility를 가져야 한다는 것이다.

Machine Machine Communication에 대한 몇 개의 예를 들기로 한다. 이것들은 實用化되고 있어 여러 가지 Application 분야에서 볼 수 있으나 아직 충분히 檢討되고 있지 않은 未踏의 분야라는 점도 殘存하고 있어 좋고 나쁘다는 結論은 이제부터라는 것이다.

먼저 Machine Machine Communication의 低레벨에 있어서의 예로는 프로그램을 보내거나 받거나 하는 Binary Bit列이 있다. 또 하나의 레벨은 메시지를 傳達 하는 Message Passing이다. 이것은 Program Module에 分割하여 서로 메시지를 보내는 것이다. 그리고 Shared Memory라는 것도 演算 분야에서는 잘 이용되고 있다. 이것은 複數의 프로그램 Module이 메모리에 접근하는 것이 可能한 것이다. 또 하나는 프로시저콜이다. 이것은 리모트 프로그램 모델간의 Communication에 사용할 수 있는 것이다.

3. 사용자의 要求

이 段階에서 지적하지 않을 수 없는 것은 業界의 專門家 중에서도 高레벨의 Communication 모델에서 어떤 레벨이 正確한가는 반드시 意見이 一致하고 있지 않다는 것이다. 技術 등에 관해서 여러 가지 調査나 研究가 행해지고 있으며 여러 분야에서 여러 가지 見解가 있다는 것이며 아직 Consensus는 얻지 못하고 있다. 이

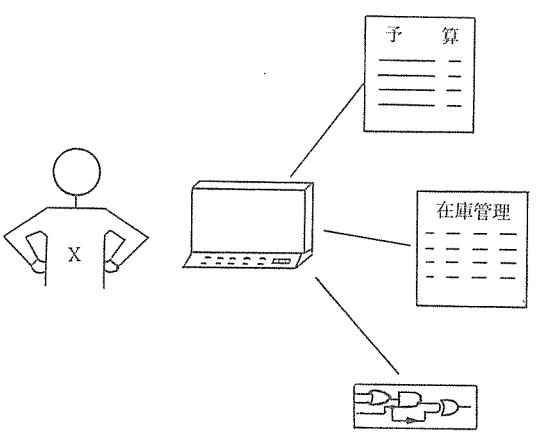

그림 2.

Man Machine Communication, Machine Machine Communication의 문제에 관해서 어떤 컴퓨터 시스템, 사용자를 想定하여 말을 진행시켜 보려고 한다'.

그림2는 사용자의 例로 X로 하고 規模가 큰 會社의 技術部의 課長이라고 假定해 본다.

課長이라는 職責으로 여러 분야의 사람들과 접촉이 있다. 會計, 經理와도 제휴해야 하고 또 在庫 관계, 購買 관계의 사람들과도 여러 가지 관계가 있다. 그리고 技術的 문제와도 관계되고 있어 엔지니어 팀이 그의 밑에서 일하고 있다는 것도 생각해야 한다. 또한 CAD 등의 시스템을 사용하고 있는 것도 생각할 수 있다. 물론 그는 컴퓨터의 여러 가지 Application Program이 이들의 각 분야에 있다는 것은 잘 알고 있는 일이다.

會計 분야에서는 專門 Package Software가 있다. 또 購買 부문에도 獨自의 소프트웨어가 있다. Display 앞에 技術者가 앉아서 CAD 시스템을 이용하여 自動化된 프로세스로 設計를 하고 있을는지도 모른다. 그래서 각 Application Program은 각각 꽤 섬세하게 시스템에 적합한 Interface를 가지도록 設計되어 있을 것이다.

예컨대 엔지니어링 패케이지, CAD의 패케이지의 경우에는 사용자는 IC 또는 게이트 레지스터, Capacity 등에 대해서는 잘 알고 있으므로 그것은 컴퓨터 言語로써 채택할 필요는 없다.

또한 經理 부문의 소프트웨어를 취급하는 사람은 원이나 Dollar에 대해서는 專門知識을 갖고 있어 Spread Sheet를 이용한 오퍼레이션만 알면 컴퓨터 言語를 사용할 필요가 없다.

다만 많은 사람들이 현재 직면하고 있는 문제는 任意로 끌어 모은 경우이다. 즉 각각 別個의 소프트웨어를 모아서 사용하려는 경우, 어떤 소프트웨어는 어떤 會社에 의해 設計되고 또 다른 소프트웨어는 또 다른 會社에 의해 設計되고 있으며 세번째의 것은 또 다른 會社에서 設計된 것이라면 거기에는 Communication은 전혀 없어 X氏가 세 개 전부를 사용하고 싶어도 잘 안 된다.

이 Sub Faction 즉, 개개의 Application Program이 別個의 機械로 運用되어 더우기 그의 會社의 다른 課나 部에 地理的으로도 分散해서 놓여졌을 경우에는 이 문제는 점점 어렵게 된다. 거기에서의 문제는 Communication에 歸結된다. X氏가 在庫管理 部署와의 접촉, 또 CAD, 그리고 經理와도 접촉이 있을 경우에는 매우 복잡하게 되어 Remote Machine이 필요하게 된다.

리모트 커뮤니케이션과 수반하여 새로 제기되는 문제로는 첫째로 低레벨인 경우에는 Bit 列의 表現 등이 틀리게 된다. 다음으로 매우 중요한 부분은 Local Reference의 문제가 있다. 이것은 一般的으로 말한다면 컴퓨터 가운데서 흐르고 있고 각 프로그램은 개개의 사용자에 있어 主가 된다. 그리고 Local의 환경에 어떤 연관을 갖게 된다. 채널의 Name이라든가 Input, Output의 명칭이라든가 또는 좀 더 複雜化된 連結性을 가지고 있을는지도 모른다. 그래서 컴퓨터의 Lease는 Local에 의해서도 콘트롤할 수 없는 문제도 있다.

예를 들면 X氏가 어떤 機械, 예컨대 그의 部下職員인 엔지니어가 가지고 있는 컴퓨터와 연결시키려 하여도 그 컴퓨터의 電源이 切斷되거나 또는 거기에서 다른 게임을 하고 있을 경우이다.

여기에서 X氏가 上司의 命에 따라 報告書를 쓰게 되었다고 하자.

예를 들면 新製品의 總賣出에 대한 어떤 書式의 報告書를 제출하게 되는 경우에는 經理 관계의 데이터도 필요하며 CAD部로부터의 設計上의 데이터도 필요하며 또한 在庫面에서의 데이터도 필요하게 된다.

X氏의 方法으로는 여기에서 리빙 도큐멘트로 불리는 方法을 생각할 수 있다. 즉, 종이에 쓴다든가 하는 受動的인 方法이 아닌 컴퓨터를 이용하여 經理나 技術部와 링크된 것보다 더 복잡한 報告書를 작성하려고 한다. 예를 들면 필요한 部品의 價格表와 링크된 것에 대해 또한 在庫 부문과의 링크를 가지려고 한다.

매일 이 報告書를 보면서 변경할 필요가 있을는지도 모른다. 각 부문에 연결되어 있으므로 自動的으로 변경되어 그 결과, Master Report를 보는 것만으로 이쪽에서 변경하지 않아

도 된다. 그러므로 Sub Section으로부터 데이터의 변경이 있으면 리포트의 作成에 그것이 自動的으로 반영되므로 언제나 그것이 리빙 도큐멘트가 된다.

이 새로운 製品이 목표를 달성할 것인가 아닌가, 예를 들면 販賣 목표를 달성할 것인가 하는 것을 계속적으로 上司에 보고하려는 생각에서 X氏가 이 리빙 도큐멘트를 作成하려고 한다.

X氏는 자신이 언제나 사용하고 있는 機械와 日常的으로 접촉하고 있는 機械와의 低레벨에서의 링크를 가지고 있다는 것이다. X氏는 자신의 部下技術者나 在庫部門과도 커뮤니케이트하여 이 링크에 자신의 컴퓨터로부터 Access될 수 있게 하고 있다.

그런데 여기에 X氏의 上司인 W氏가 있다고 하자. X氏는 W氏에 대해 자신이 作成한 리빙 도큐멘트를 제출하게 된다. W氏는 리빙 도큐멘트로 新製品의 動向을 파악할 수 있다. 그래서 W氏는 이번에는 이 리빙 도큐멘트를 자신의 컴퓨터에 複寫하려고 한다. 그것은 매우 간단하다. Bit 패턴을 X氏의 컴퓨터에 옮기면 된다.

그러나 아쉽게도 현재의 技術 수준에서는 이것은 잘 되지 않는다. 왜 잘 되지 않는가 하면 이쪽의 機械 사이의 링크나 다른 低레벨에서의 機械에의 Access는 X氏의 컴퓨터로는 可能하나 그것이 W氏의 컴퓨터에 옮긴 경우에는 意味가 없어지게 된다. 따라서 여러모로 링크를 伸張시켜야 한다.

그러면 지금까지를 간단히 정리하여 보기로 한다. 技術的인 문제에는 두 개의 분야가 있으나 다같이 중요하다고 생각된다. 앞에서의 X氏의 例를 채택하여 說明한다면 첫째 문제로는 整合性에 관한 문제이다. 프로그램 또는 사용자의 環境 속에 서서히 침투해 오고 있는 여러 가지의 문제가 있으나 그 가운데 어떤 系統的인 方法으로 이 문제를 整理하지 않으면 안 된다. 여기에서의 목표는 相違한 環境下에서도 같은 內容을 가진 프로그램이면 같은 형태의 Interface를 갖지 않으면 안 된다는 것이다.

하나의 具體的인 예를 든다면 하나의 서비스에는 다같이 同一한 Interface가 필요하다는 것이다. 여기에서 중요한 것은 이 整合性을 달성하기 위해서는 Communication을 위한 매우 知的 수준이 높은 컴퓨터를 導入해야 한다는 것이다. 더우기 標準化를 채택한다면 더욱 좋다.

두번째 문제로는 이것은 解決이 그렇게 容易한 것은 아니지만 파워에 관한 것이다. 매우 복잡한 예컨대 앞에서 說明한 바 있는 리빙 도큐멘트와 같은 것을 高레벨에서 취급해 보자는 것이다. 여기에서 여러 가지의 이유는 있겠으나 가장 중요한 것은 Reference의 커뮤니케이션이다.

여기에서 말하고 싶은 것은 이 能力은 어떤 象徵的인 심볼을 연관짓는다는 것과 어떤 環境에서 또 다른 環境에 이 심볼을 옮긴다는 것이다. 그리고 그 심볼이 意味하는 것, 또는 象徵하고 있는 것은 다른 環境에 옮겨서도 충분히 通用되지 않으면 안 된다는 것이다. 이것이 Machine Machine Communication의 하나의 문제이다.

4. 標準化 水準과 問題點

이제부터 解決의 可能性을 밝히고자 한다. 즉 이 문제를 整理하자면 어떻게 하는 것이 좋은가 技術的인 側面에서 본다면 매우 쉬운 문제이나 여러 가지의 政治的인 障害도 있으므로 그렇게 쉽게는 볼 수 없다는 것이다.

가장 쉬운 것은 廣範圍한 레벨에서의 標準化를 달성한다는 것이다. 여기에는 標準化를 달성시켜 전부가 그에 따르도록 하면 된다는 標準化가 용이한 분야이기도 하다. 이와 같은 解決이 간단한 문제로서는 매우 低레벨에서의 表現 즉, 네트워크 프로젝트나 Byte의 配列 순서가 있다. 그리고 Back Plan Bus의 Architecture가 있으나 이것은 컴퓨터의 메카니즘으로는 아주 基本的인 것이다. 또 命令 세트를 공통되게 한다는 것도 그렇다. 技術的으로 보면 이같은 標準化는 쉬운 것으로 볼 수 있으나 아직 進展된 상태가 아니므로 많은 사람들은 극히 悲觀的이다. 가까운 장래에 이것이 標準化될 것이라는 展望은 거의 없다.

다음으로 더욱 어려운 문제로는 汎用 프로그램 言語나 프로그램의 System Interface, Human Interface 등 몇 개를 들 수 있다. 왜 어려운가

具體的인 예로 英語의 文法을 생각해 보면 좋을 것이다. 이것은 英語를 하는 사람들에게는 어떤 種類의 매우 편리한 標準化이지만 하나 하나의 말을 標準化할 필요는 없다. 一貫性 문제의 전부는 아니지만 그 일부는 標準化에 의해 해결된다.

5. 標準化의 한 手段 : Personal Computer

여기에서는 低레벨의 標準化에 대해 간단히 說明하려고 한다. 과거 6년간 MIT에서 추진해 온 NU로 불리는 퍼스널 컴퓨터 프로젝트에 대해 말하고자 한다.

NU가 意圖한 이 Architecture의 目標는 廣範圍한 需要者에게 強力한 파워나 Flexibility를 提供하는 것이다. 그러면서도 NU는 基本的으로는 現世代의 技術과 반드시 連結되는 것은 아니다.

NU는 그 Architecture를 꼭 定義하지 않아도 된다. 즉 標準化라는 側面에서 말한다면 基本的으로 Back Plan Bus를 採用하고 있으나 이것은 基本的인 Communication Mechanism이며 단지 하드웨어 Module間에서의 Communication에 사용되는 것이다. 그러므로 NU에서는 하드웨어 Module間의 Communication을 標準化하는 것만으로 하드웨어 Module間에서 Communication되고 있는 實際의 알맹이를 標準化할 필요는 없다.

이같은 새로운 퍼스컴은 Intel이나 Motorola, Zirog 등에서 여러 가지 種類의 Processor를 사용하고 있으며 리스프머신社에 의해 製作된 部品도 사용하고 있다.

그림 3은 NU 머신의 典型的인 Configuration이나 Basic이나 Communication mechanism은 予期되는 대단히 廣範圍한 Configuration을 지원할 수 있다.

6. 言語 레벨의 標準化

여기에서 또 하나의 매력적인 標準化로 汎用型 프로그램의 言語標準化가 있다. 예를 들면 世界 各國이 같은 프로그램 言語를 사용하는 決定을 내리면 各國이 같은 데이터를 사용하게 된다. 그것은 그 言語에 있어 상당한 부문에까지 데이터 形成까지도 標準化되므로 그 레벨의 標準化는 대단히 매력적이며 로컬 베이스의 사람들 예컨대 작은 組織의 사람들은 言語의 標準化를 희망하고 있다.

그러나 상당수의 많은 사람들은 이 레벨의 標準化를 받아들이지 않기로 결정하고 있다. 이것은 言語의 레벨이 대단히 낮기 때문이다. 예컨대 C言語는 시스템 프로그램 言語이지만 충분히 高레벨의 標準化를 提供할 수 없게 된다.

言語 標準化에 관한 또 하나의 문제는 수많은 非技術的인 장해가 있다는 것이다. 소프트웨어의 각양각색의 Source가 각양각색의 言語로 쓰여져 있기 때문이다.

다음의 문제는 技術的인 側面에서는 대단히 매력적인 것이지만 Communication의 環境下에서 네트워크를 통해 高레벨의 Object를 취급, Reference의 頻度도 대단히 높은 경우에 관한 것이다. 이 tag 附데이터 즉 自己認識型 데이터는 그 價値는 물론 어떤 型의 것인지도 認識할 수 있는 것이다.

이 自己認識型의 데이터를 사용하므로써 프로그램 또는 프로그램에 대해 로컬 스탠더드를 설치할 수 있다. 그리고 그것을 데이터와 함께 보낼 수 있다. 받는 쪽은 데이터와 스탠더드의 양쪽을 받아서 意味가 있는 것으로 한다.

標準化에 있어 대단히 매력이 있는 것으로는 美國의 여러 메이커가 사용하고 있는 LISP 머신이 있다.

Reference Data의 Communication에 관해서 技術的으로 아직 충분한 해결을 보지 못한 문제도 확실히 있다. 매우 大量으로 복잡한 데이터를 相異한 環境에 보내는 경우 相異한 데이터의 構造를 모두 認知하고 그것을 다른 環境에 보내지 않으면 안 된다는 것이다. 附隨된 모든 것을 보내지 않으면 안 된다. 여기에서 基本的인 문제는 tag附데이터를 사용하는 시스템은 低레벨에서는 Global Space를 필요로 한다는 것이다. 흔히 데이터의 Address Space는 그 데이터를 認識하기 위해 사용되는 1對1의 관계이다. 그러나 리모트環境下에서는 相異한 Separate Space를 취급하고 있으므로 Globol N-

ame Space와의 相關性이 없다. 즉, 하나의 機械에서의 Address Space를 다른 環境에 보내 다른 機械가 거기에서 意味를 채취하는 것은 期待할 수 없다. 그것은 다른 機械가 전혀 獨立된 Address Space 體系를 가지고 있기 때문이다. 이런 것들이 우리들이 直面하고 있는 문제이며 이에 대한 여러 가지 解決策이 提起되고 있어 여기에 대해서 이제부터 言及하고자 한다.

7. Protocol과 커뮤니케이션

이제부터의 방향으로서는 Implementation 이 아닌 서비스를 提供하는 즉, 수동적인 입장에서의 데이터의 構造를 考察하는 것이 아닌 메시지를 보내는 方向인 것이다. 給與支拂의 경우를 말하면 이름은 10 Bit, 月給은 5 Bit 필요하지만 그와 같은 給料明細書와 같은 받는 쪽의 데이터가 아닌 예를 들면 X氏의 月給을 보내 달라고 하면 X氏의 月給에 해당하는 Digit 列이 되돌아온다는 것이다. 그러나 Input, Output Behaviour와 그 서비스間에는 커다란 壁이 있지 않을까 생각된다.

이 戰略이 제대로 된다면 형태로는 Portable Reference 즉, 統計 또는 Reference를 傳達할 수 있게 된다. 다시 말하면 Object를 表示할 경우 모든 데이터를 함께 보내지 않아도 外部環境에서 意味를 나타낼 수 있는 데이터만을 보내는 것만으로도 족하며 그것이 서비스 提供이 된다는 것이다.

이것을 간단히 描寫한 것이 그림 4 이다. 이 그림에서의 구름과 같은 것이 서비스이며 각각의 화살표가 Communication bus이다. 예를 들면 低次元에서의 문제로는 文書의 作成 프로그램을 생각해 보자. 文書 作成 과정에서 오늘의 날짜를 入力치 않으면 안 된다는 指示가 있었을 경우 그 메시지를 日附의 서비스에 대해 Communication bus에 가까이 하면서 알려 달라고 그 서비스를 요구하게 되는 것이다. 그렇게 해서 서비스의 提供이 이루어지게 된다. 또 Output의 서비스는 Output를 위한 Device, 예컨대 디스플레이라든가 Laser Printer 등과 관계를 유지하고 그 서비스를 表示한다.

Communication의 모델 그 자체를 대체적으로 다룬다면 TRIX 시스템으로 볼 수 있다. 이것은 MIT가 퍼스컴을 위해 開發한 것이다. 이 모델의 레벨에서는 Communication의 패턴을 標準化하는 즉, Implementation의 標準化가 아닌 Communication을 標準化하는 것이다.

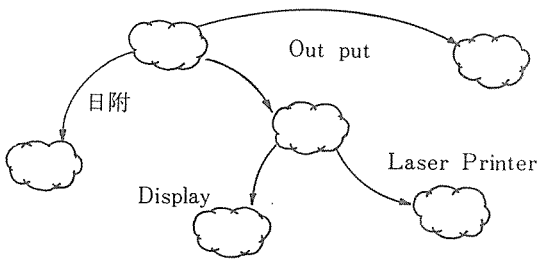

그림 4. TRIX : MIT가 開發한 OS

이 TRIX 프로젝트는 Communication 및 Communication bus에 焦點을 맞추고 있다. 본래 이 프로젝트가 開發된 것은 需要者 레벨에서의 一貫性, 각 需要者의 환경에서 의미있는 一貫性을 갖게 하는 즉, 상호 접속된 많은 퍼스컴이 서로 의미 있는 형태로 一貫性을 가진 Communictaion이 되게 한다는 것이 目的이다.

TRIX의 背后特徵은 兩者의 利點을 채택하려는 것이다. 예컨대 하나의 機械로부터 다른 機械에 Reference 없이 데이터를 移轉케 하는 것이 바람직한 것으로 이것은 하이 레벨 Communicaton에 있어서는 대단히 중요하다고 앞에서도 說明한 바와 같다. 그러나 이에 덧붙여서 다시 言語의 선택이나 프로그래밍의 方法 등을 지나치게 規制하지 않으려고 한다.

그러므로 TRIX는 다른 側面은 標準化를 원하지 않고 있다. 즉 言語의 선택 등의 標準化는 하지 않고 Communication Mechanism 등 有效適切하게 필요한 것은 標準化하고 있다. 다시 말하면 Object의 Reference, 다른 機械로부터 다른 機械에 移轉이 가능케 하는 형태의 標準化는 진행시키고 있다. Application Program 에서는 지금 段階에서는 아직 완전한 서비스를 제공하지는 못하고 있으나 그와 같은 서비스를 제공하기 위한 進化의 디딤돌이 될 것으로 생각된다.

그림 5는 TRIX의 基本的인 原理지만 이 그림은 두 가지의 要素에 의해 構成되고 있다. 이 그림의 작은 구름(雲)을 우리들은 Domain이라고 호칭하고 있으나 이에 대해서는 뒤에서 說明하기로 한다. 이를 連結하고 있는 線은 Domain間의 Communication의 經路이다. TRIX의 基本的인 原則은 唯一의 Communication은 이들 經路를 통하지 않으면 안 된다는 것이다. 즉 감추어진 Communication bus는 表面에 나타난 것뿐이라는 것이다.

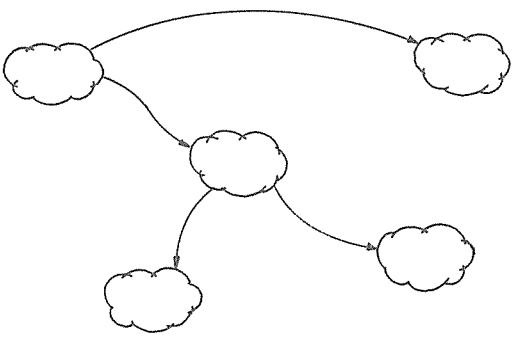

그림 5.

이들 Domain은 각각 어떤 서비스와 관련되고 있다. 많은 경우 低레벨의 서비스 즉, File이 행하고 있는 서비스로 TRIX에서는 普通의 오퍼레이션 서비스는 하나의 Domain이 하는 것으로 된다. 어떤 Domain은 特定 經路에 의해 File에 Access되어 쓰여지기를 요구한다. 이 요구에 따라 쓰여진 데이터에 대해 읽으라는 요구가 있을 경우 같은 經路를 반대 방향으로 되돌아가게 된다.

그리고 Cannel이라고 하는 고찰 방법인데 TRIX의 경우에는 매우 간단한 것이며 通常的인 OS와 같은 복잡성은 없다. Cannel이 서포트하고 있는 것은 Domain과 Sled와 핸들로 불리는 세 가지이다. Domain은 通常的인 OS의 프로세스의 Static 임파어런먼트 즉, 보호되고 있는 하나의 領域을 가지고 있다. 그러므로 앞의 그림에서는 구름(雲)이라는 형태로 나타나고 있다.

다음으로 Sled이나 이것은 예컨대 TRIX의 경우에는 레지스터나 Stake 등의 Active Processing Agent이다. 어느 段階에서도 그 Sled는 어딘가의 Domain에 들어 있어 Domain의 Static Context를 提供하는 관계에 있다. 하나의 Domain 중에서 몇 개의 Sled가 각각의 機能을 遂行하므로써 하나의 Address Space 중에서 데이터나 다른 서비스를 共有하게 되는 상황이 있을 수 있는 것이다. Sled는 언제나 하나의 Domain 속에 있어 그 Domain에 로컬한 요구를 하고 있다. 그 Domain에 한해 그 Domain Code의 支配下에 있게 된다.

끝으로 핸들은 이들의 Communication bus의 로컬한 명칭으로 각각의 Domain은 Address Space에 덧붙여 몇 개의 로컬 네임을 핸들을 위해 가지고 있다. 즉 어떤 Domain은 두 개의 Communication bus를 가지고 있어 각기 핸들을 가지고 있다. 각각의 핸들은 어디로 그 Communication이 향해져 있는가를 지시하고 있다.

그러면 TRIX의 原始的인 측면, 즉 基本原則 등에 대해 간단하게 檢討해 보자. TRIX의 Communication의 基本的인 要素는 종래의 Message Passing Communication과 共通點이 있으며 또 Remote Processor Call과도 類似性을 가지고 있다. 다만 Communication의 Mechanism은 Request와 Reply로 불리는 Active한 Pare를 필요로 하고 있다.

그림 6과 같이 두 개의 Domain이 있다고 하자. 그 사이에 Communication Bus가 있다. Communication Bus는 左側의 Domain에 로컬한 이름, 즉 h라는 이름이라고 하자. 그렇게 되면 리퀘스트라고 하는 오퍼레이션이 있을 때 이 Domain은 핸들 h로 리퀘스트를 나오게 하며 그리고 이 Communication bus를 認識하여 메시지를 전달한다는 것이다.

한편 右側의 Domain은 리퀘스트를 받은 段階에서 처리되어 본래의 Domain에 答을 보낸다. 다만 이것이 實施되는 Mechanism은 特定 Sled가 左側 Domain으로 리퀘스트를 實行하여 그 결과, Sled는 전혀 다른 프로그램 Context에 移動하게 된다. 즉 핸들h에 의해 認識된 Domain에 의해 움직이게 된다.

그리고 그 리퀘스트의 Target Domain 으로 Sled는 實行한다. 그러나 그 段階에서 다음의 메시지, 즉 Reply를 實行해야 되므로 Sled는 本來의 Domain으로 되돌아와 본래의 位置에서

作業을 시작한다.

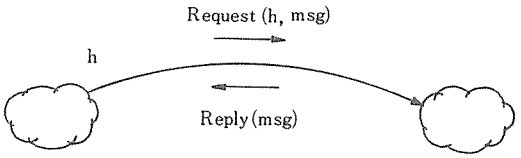

그림 6.

그러므로 리퀘스트와 Reply의 mechanism 은 Remote Processor Call과 거의 흡사하다.

새로운 Domain에 移行하므로써 새로운 메시지가 實行되게 된다. 移行된 Domain을 위해 하나의 서어비스를 提供한다. Reply가 完了되어 그 서비스가 끝났을 때에 Sled는 左側의 Domain에 되돌아와 본래의 呼出을 하게 된다. 이와 같은 交換에 의해 Data Structure의 메시지가 右로부터 左로 전달된다. 좀 더 詳細히 이 메시지를 觀察하면 메시지 그 자체는 情報와 함께 Response를 받은 Domain이 要求하고 있는 것이 무엇인가를 具體的으로 指示를 하게 된다.

이 標準化는 TRIX에는 共通되고 있다. 그러므로 이 標準化는 어떤 핸들일 경우에도 理解되어야 한다. 그러므로 네트워크를 통해서 메시지가 전달되었을 경우에는 어떤 經路에 의해서도 그 메시지의 내용을 이해할 수 있어야 한다. 그와 같은 의미에서는 네트워크가 理解할 수 있도록 TRIX의 내부 標準化가 필요하다.

또 하나의 要素는 우리들이 소위 Data Window라고 呼稱하고 있는 것이다. 이 Data Window는 基本的으로는 Address Space가 있는 일부분을 認識하게 된다.

Data Window는 基本的으로는 大量의 데이터를 Remote Context 間에서 効率的으로 전달하는 Mechanism이다.

Domain이 있는 Address Space의 Data Window를 두번째의 Domain, 세번째의 Domain으로 연이어 전달한다. 그리고 最終的으로는 전달된 大量의 데이터가 直接的으로 Data Window가 認識하고 있는 本來의 Domain으로 되돌아간다.

TRIX 메시지에 있어 보호되는 가장 중요한 情報는 핸들이다. 왜 중요한가 하면 Object Reference를 하나의 환경에서부터 外部의 환경에 移行시키는 TRIX의 기본적인 특징이기 때문이다. 다시 말하면 메시지를 어떤 Domain으로부터 다른 Domain으로 보내는 경우에는 핸들은 로컬 네임에 의해 두번째의 Domain에 보내고 그리고 다시 目的하는 세번째의 Communication Domain에 情報를 순차적으로 보내게 되는 형태가 된다. 그림 7과 같이 左側의 Domain h는 右側의 Domain과 連結되어 있으나 핸들에 의해 아래의 Domain과도 연결되게 된다.

TRIX의 경우 Reference Communication Handle에 의해 그림 7의 左側 Domain이 메시지를 右側의 Domain에 보낼 수 있게 된다. 그러나 그 메시지 가운데에 핸들의 Copy g를 세번째의 Domain에도 보낸다. 즉 左에서부터 右의 Domain에 메시지가 보내지면 그 핸들은 시스템에 의해 自動的으로 右側의 Domain의 로컬 Language에 번역된다. 그리고 다시 세번째의 Domain에의 이쪽으로부터의 연락 經路가 열리게 된다.

그러므로 이에 따라 Communication bus 그 자체를 받아들여 그것을 데이터 Object로 취급할 수 있게 된다. 그리고 外部의 전연 異質的인 환경에서 취급할 수 있게 된다. 이에 따라 Communication bus의 네트워크 즉, 핸들과 Domain 등의 네트워크는 Sled의 活躍結果 대단히 能動的으로 進化하게 된다.

여기에서 다시 強調하고 싶은 것은 이 TRIX의 Cannel이 提供하고 있는 Mechanism은 Domain, Sled, Handle의 세개에 한한다.

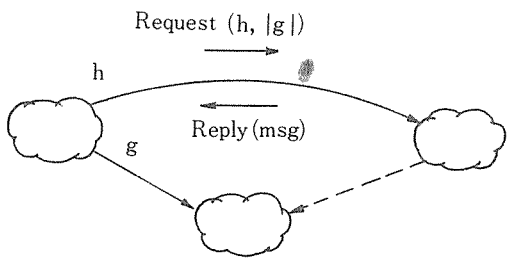

그림 7.

TRIX의 基本的인 모델은 이들의 서비스를 特別한 케이스로 취급하고 있다. 즉 Domain이 提供하는 서비스 가운데서 特殊한 것으로 取扱하고 있다. 간단히 說明하면 예컨대 File과 關聯되는 서비스를 提供하고 있는 경우를 생각할 수 있다. 또 디바이스의 역할을 하는 Domain은 Input, Output 디바이스에 直接 Access가 可

能한 코드도 가지고 있다. 그러므로 Input, Output Request이면 File의 읽고 쓰기 리퀘스트와 같은 형태를 취할 수 있다. Directory의 역할을 하는 Domain도 있다.

Directory라 함은 간단히 말하자면 네임을 다른 서비스와 연결시키는 서비스이다.

Interprocess Communication 즉, 두 개의 Active Process 사이를 연결하는 핸들이다. 여기에서 중심적인 것은 File System이다. 왜 File System이 중심적인가 하면 네임의 Semantics와 매니지먼트를 콘트롤하기 때문이다. 在來型의 OS와는 달리 TRIX의 경우에는 File을 네이밍하는 機能이 없다. 즉 File에 대해 전연 知識을 갖지 않고 있다. 그러면서도 또한 直接으로 Domain에 Refer할 수 있는 機能도 없다. 다시 말하면 TRIX의 核이 되는 Cannel Program으로서는 하나밖에 네임을 필요로 하지 않는다. 핸들이라는 것은 Communication bus 의 로컬네임이지만 TRIX의 생각으로는 Domain의 네임, 또는 Directory, File의 네임 등은 전연 갖지 않고 있다. 低레벨에서는 핸들 만으로 File System을 機能케 하자면 핸들에 대해 高레벨에서의 네임을 부여하고 있다. 이 Mechanism은 Domain의 여러가지 분류 가운데서 Directory의 機能을 遂行케 하는 Domain을 통해서 하고 있다. Directory라 함은 일반적으로 말하자면 TRIX의 경우에는 네임을 連結시키는 것과 같은 서비스로 Character String를 개개의 핸들과 연결시키기 위한 서비스이다. 그러므로 그림 8의 경우 左의 Domain이 Directory 서비스를 提供하는 것으로 假定해 보자.

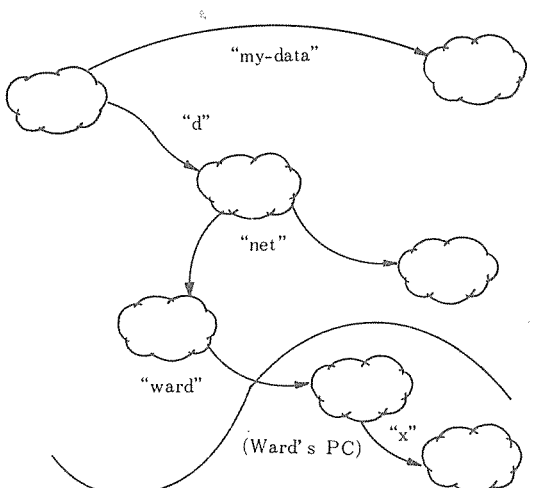

그림 8.

즉, 그 內部의 構造로 카탈로그 및 象徵的인 리스트를 갖고 있다는 것이다. 그 가운데 예컨대 "X"라든가 "Y"라는 이름이 있어 그것이 각각의 핸들의 이름이라고 假定한다. Directory 서비스에 Access가 행해지면 예컨대 "X"를Look Up하라는 메시지가 보내지게 된다. 이것은 Proceeder Form으로 쓰여지게 되지만 그 메시지는 그 Directory Domain이 연결되어 있는 핸들을 원위치로 되돌려 주는 作業을 하게 된다.

이 리퀘스트는 먼저 Bus「d」를 통해서 別個의 Domain에 들어 간다. 이 메시지는「x」를 룩업하라는 것이나 그렇게 되면 이 핸들을「x」와連結한다. 그리고 그 핸들, 또는 그 핸들의 Copy를 본래의 Domain에 보낸다.

TRIX의 Structured Name을 받아들일 경우 TRIX는 여러 가지 機能을 갖는다. 예를 들면 Directory의 機能을 가진 Domain에「x/y」를 Look Up하라는 메시지를 보낸다. 이것은Unix Syntax에서는 x 중의 y라는 의미이지만 이 Structure Name으로부터 最初의 部分인「x」를 分離시킨다. 그리고 그 로컬의 環境을 여러모로 살핀다. 그 결과로서「x」로 불리는 핸들이 別個의 Domain을 認識하게 된다. 이것이 Directory Protocol을 따르고 있는 別個의 Domain을 認識한다. 여기에서 또 다른 後半의「y」의 部分만을 불러들이는 Look Up 메시지를 하게 된다.

「x」에 의해 認識된 Domain에서는「y」의 핸들을 통해서 필요한 데이터를 뽑아 내어 본래의 Domain에 되돌려 보낸다. 간단하고 自然的인 방법이지만 階層化되고 또는 계통적으로 整理된 네임間의 여러가지 交換에는 대단히 便利하다. 그러나 여러 가지 콤포넌트가 많은 Structured Name의 Object Reference의 경우에는 그 네임의 각 요소에 대한 解釋이 필요하게 된다. 또한 로컬한 環境에서의 解釋이 필요하게 된다. 이 分散된 Semantics의 아이디어를 확대해 가면 高레벨네임의 整合性에도 연관되게 된다.

라고 말하는 것은 이에 대처할 충분하고도 強力한 機種이 아직 出現하지 못했기 때문이다. 다시 말하면 標準化를 위해서는 汎用的이며 매우 強力한 機種이 필요하게 되는 것이지만 여러 가지 System Interface의 문제 등 高레벨에서의 문제점이 있기 때문이다.

퍼스널 컴퓨터의 技術 수준은 標準化가 進展되면 충분히 向上될 것이지만 다만 어떻게 標準化를 推進할 것인가 하는 것이 문제가 될 수 밖에 없다. 또한 한편에서는 標準化로 반드시 모든 문제가 解決된다고는 생각지 않는 見解도 있다. 標準化는 어느 정도까지 一貫性을 유지시키는 것으로 족하며 결점도 있다. 예를 들면 標準化는 予期되는 문제만 커버할 수 있다는 것이다. 즉 1980년의 水準에서 予期되는 문제에만 對處할 수 있었으며 여기에는 두 가지 결점이 있다.

먼저 標準이 作成된 時點에서는 予期할 수 없는 수준에서의 表現上의 문제가 앞으로 나타나게 될 可能性이 있다. 그 다음으로는 일단 이 標準化가 確定되면 技術的으로 水準이 높은 標準化의 機會를 빼앗아 갈 수 있는 可能性이 있다는 것이다. 그러나 標準化 側面에서는 말하자면 妥協的인 方案도 있다. 예를 들면 Communication 계획의 基本的인 레벨의 標準化를 더욱 詳細한 下位 레벨에서의 標準化가 아닌 더욱 一般化한 형태의 標準化를 進行시킨다는 것이다.

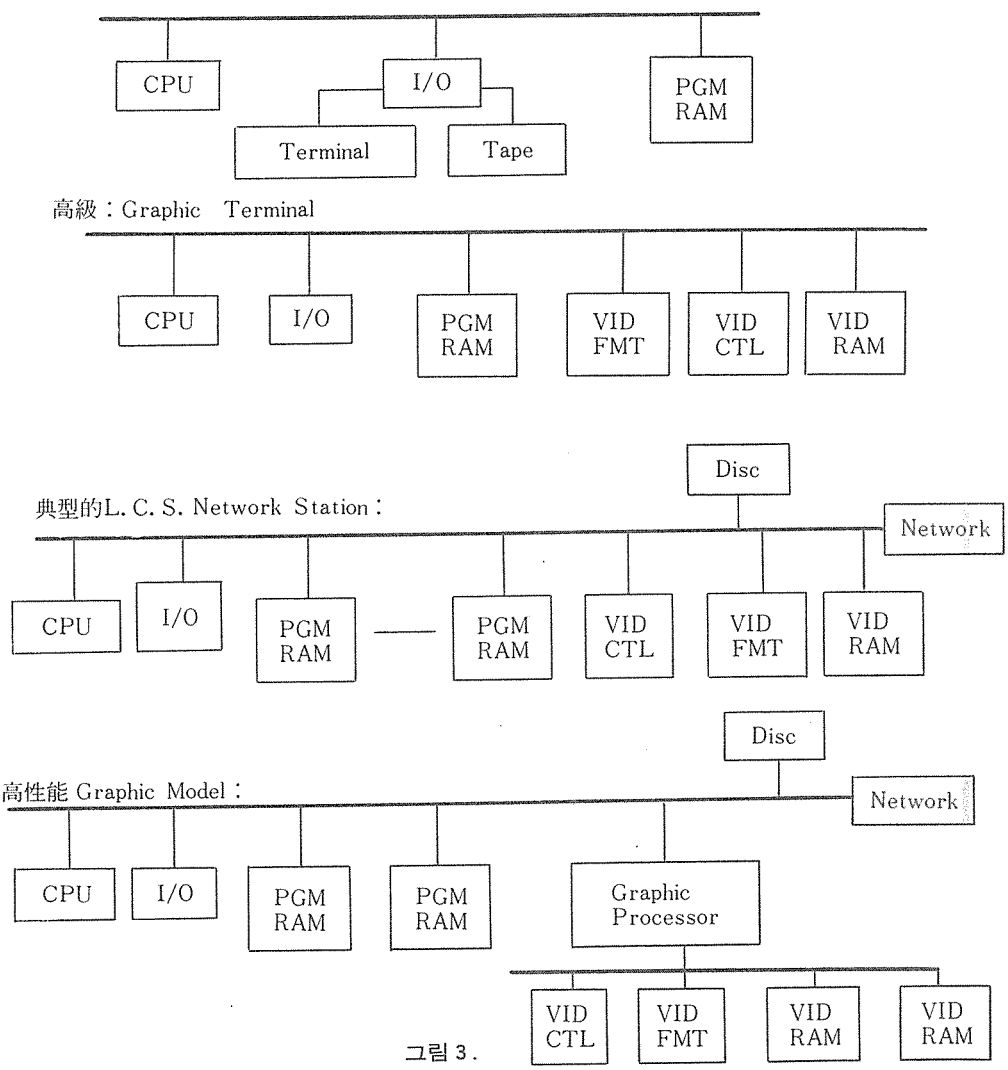

그림 3.

그림 9는 앞의 그림 8과 같이 左上이 디렉토리 서비스를 가지고 있는 Domain이다. 단 이 경우 디렉토리 가운데의 네임의 하나는 네트워크 서비스를 가지고 있는 Domajn에의 핸들일는지 모른다. 즉 「net ward/x」라는 네임의 Entry가 있어 이것은 앞에서와 같은 Mechanism을 사용하여 해결된다.

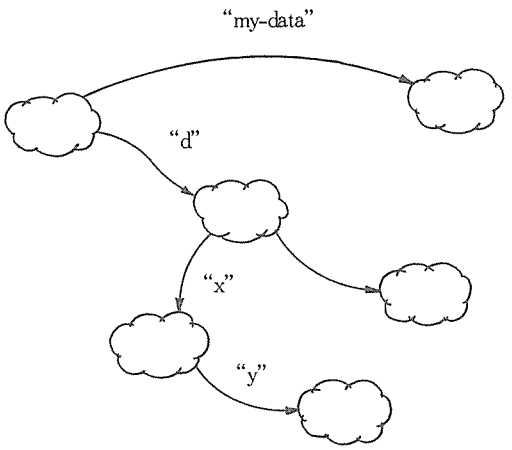

그림 9.

Look Up 메시지를 디렉토리 Domain에 보낸다. 그 Domain으로부터 이번에는 「net」를 룩업한다. 그러면 그 Domain쪽으로 핸들이 보내진다. 그러면서도 「net」의 경우에는 그 Domain은 반드시 디렉토리는 아니다. 이것은 能動的인 형태에서의 네트워크 서비스이며 로컬에리어 네크워크에의 Interface를 가지고 있다.

「Ward/x」는 이 로컬한 환경에서의 해석으로 「Ward」로 불리는 遠隔地의 機械를 룩업하여야 한다. Ward의 機械로 「x」를 찾아내게 된다. Ward의 機械가 그 메시지를 받아 解釋하고 핸들의 형태로 「x」의 Domain에 보내 그것을 본래의 Domain에 되돌려 보낸다. 그러므로 TRIX의 네임의 Semantics는 自然的이며 간단하게 일관되어 있다.

任意의 크기로 單一의 整合性으로 Address Space로 모든 일이 滿足하게 된다. 그리고 리모트 머신과 Communication bus는 로카라이저되어 있다. 즉 Network Domain과 이들의 다른 遠隔地에 있는 機械와의 사이가 緊密히 連結되게 된다. 그러므로 여러 가지 Communication, 技術, 또는 다른 여러 種類의 Protocol과도 연결시킬 수 있다.

그러면 두 세 가지 性能에 대해 言及하려고 한다. TRIX의 最新의 상황, 實施面에서의 문제를 들어 왔으나 우리들의 經驗을 살려 構造의 改善과 性能上의 배려도 했다. 특히 이전의 TRIX 시스템에서는 매우 간단한 方法으로 메시지의 交換을 했다. 이것은 앞에서도 말한 바와 같이 Semantics를 사용했을 때는 좋으나 Sled나 핸들을 사용했을 경우의 오버헤드가 높아지는 경향이 있다.

어떤 Communication에서도 이 OS의 Scheduling Mechanism 때문에 예컨대 File System과 같은 복잡한 서비스로 複數의 需要者나 복수의 프로그램을 동시에 교환하는 서비스를 提供해야할 경우에는 이 Domain에는 Multi Processing Mechanism을 가지고 있어야 한다. 그러나 OS側으로 보면 이것은 필요없는 二重의 일이 되고 만다. Synchronous 메시지 패싱의 경우에는 大量의 Over head가 일어난다.

Domain과 Sled의 構造에 있어서는 TRIX의 경우에는 여러 가지 實行上의 배려가 있은 다음 設計되고 있다. 例를 들면 單一 Stak를 Sled를 위해 사용할 수 있다. 單一Stake은 예컨대 리퀘스트 또는 Reply라고 하는 Communication에 사용할 수 있다. 즉 리퀘스트와 Reply의 오버 헤드의 量이라고 할 수 있으나 Proceeder Call의 오버 헤드의 量과 대체적으로 같다.

우리들이 최근 여러 가지로 研究하고 있는 것은 하드웨어의 Architecture를 TRIX의 모델에 맞추어 만들려고 하는 것이다. 이에 따라 TRIX에 의한 Communication의 効率化를 企圖하고 있다.

8. 展望과 問題點

끝으로 앞으로 수년간의 進化에 대해 說明하려고 한다. 현재의 技術 수준에서 可能한 改善點은 低레벨에서의 能力에 관해서는 앞으로 수년간 내로 크게 改良될 것으로 생각된다. 演算處理에서도 또 Input, Output 디바이스에서도 音聲認識 또는 畵像認識과도 같은 技術도 可能해질 것이다. 또한 비디오 디스크 또는 大量의 File 裝置 등도 發達될 것이다.

〈P. 90으로 계속〉

能하고 機器의 小形化의 必要에 合致됨. 이 콘덴서는 形狀도 抵抗器(칩形)와 合해져 있기 때문에 基板裝着의 自動化에 寄與가 큼. 이 콘덴서는 그 構造나 使用材料가 세라믹과 金屬뿐이기 때문에 使用溫度가 높고 使用溫度範囲도 더우기 넓음. 前述한 回路基板 직접부착의 단순한 構造는 耐振性에 있어서도 우수하기 때문에 이 콘덴서가 지금까지와 다른 환경에서 要求되는 경우에도 對應하기 쉽게 새로운 用途 分野에의 擴大에도 관련됨. 이 콘덴서는 지금까지 기술한 바와 같이 今後도 用途의 擴大, 多樣化의 可能性을 갖고 있으나 그것은 이 콘덴서가 크기 特性 등에서 多品種으로될 염려가 있어 今後에 이 規格의 운용을 통하여 標準化를 꾀하는 것이 製造者 및 使用者 양측을 위하여 필요할 것임. 그리고 이 콘덴서의 擴大되는 多樣化 用途를 위하여 參考로 使用上에 있어서 一般的으로 注意할 事項을 다음에 열거함.

(1) 이 콘덴서는 端子電極이 노출되어 있어 電極이 高溫, 多濕, 硫黃分, 塩素 가스 등을 포함한 분위기에 방치하면 酸化하여 납땜부착성이 약해지는 경우가 있음. 따라서 이 콘덴서는 위에 留意한 환경조건에서는 될 수 있는대로 短期間內에 使用됨이 바람직함.

(2) 이 콘덴서는 予熱 不充分 상태에서 그대로 납땜부착 등의 熱處理를 행하면 접촉성 저항 또는 기공 등으로 인하여 絶緣劣化 또는 絶緣不良으로 될 염려가 있음. 이것을 防止하려면 100℃ 以內로 되는 溫度差의 予熱이 바람직함.

(3) 이 콘덴서는 最近 프린트 基板에 직접 부착하여 使用하는 경우가 많아지게 되었음. 이 경우에는 基板의 材質에 따라서는 부착후에 휘어져 세라믹이 갈라지거나 또는 미접착을 誘發하는 경우가 있어 絶緣劣化 또는 絶緣不良으로 될 경우가 있음. 따라서 이러한 염려가 있는 경우는 부착후의 基板의 휨을 적게하기 위하여 基板의 材質選定과 回路 패턴의 設計 및 부착후의 取扱 등에 適切한 留意가 必要함.

(4) 이 콘덴서는 부착후의 洗淨不充分에 따라 殘留후럭스가 固着한 대로 되어 있으면 絶緣劣化의 原因으로 될 경우가 있음.
따라서 충분한 洗淨을 행하기 위하여 適切한 洗淨液의 選擇과 洗淨條件의 選定이 必要함.

(5) 이 콘덴서는 高溫이나 長時間의 납땜부착은 端子의 납땜덩어리가 생기기 쉬움. 이것을 防止하려면 最低限의 溫度로 短時間의 납땜부착이 좋음. 또한 銀을 넣은 납땜의 使用이 바람직함.

(6) 이 콘덴서는 基板裝着後 等에 使用하는 코팅樹脂의 選擇에 충분한 配慮가 바람직함. 適合하지 않은 樹脂를 使用하면 코팅후에 高溫, 多濕, 溫度싸이클 기타 環境條件에 對應할 수 있는 高信賴性의 것을 얻기 힘들게 되는 경우도 있음.

〈P. 82로부터〉

이것은 퍼스컴의 今後 10年間의 發展 과정에서 매우 중요할 것이지만 단 그렇게까지 革新的인 의미는 갖지 않을는지도 모른다. 다시 말하면 새로운 技術에 의해 가능해진 여러 가지 기능 또는 能力만으로는 새로운 모델, 새로운 技術, 거기에 Man Machine Communication, 또는 機械間의 Communication에 있어서의 전혀 새로운 모델을 創出하기까지는 이르지 못했다는 것이다.

끝으로 지금까지의 內容 가운데서 중요한 것만을 整理해 보기로 한다.

첫째로 퍼스컴은 演算機能을 가진 media로서가 아닌 이제부터는 Communication media로 보아야 한다는 것이다. 이 점을 強調하고 싶다.

다음으로 여러 가지로 문제가 있으나 곧 提起되어야 할 문제가 標準化이다. 그러면서도 이 문제는 여러 가지 行政的인 面에서의 制約을 排除하여 적절하게 計劃을 實行해 나아간다면 거의가 解決될 것이다. 또 Reference의 傳達, 外部 機關과의 Communication의 교환 등에서는 아직 技術上의 여러 가지 어려운 문제가 있다.

또 低레벨에서의 技術的인 側面에서는 現狀 문제를 해결해야 할 경우도 있으나 낡은 Approach를 그대로 계속할 경우에는 革新的인 Approach, 또는 문제의 解決 方向을 提示하지는 못할 것이다. 또한 基本的인 시스템에서의 문제점으로는 더욱 強力한 Communication모델이 더욱 광범위하게 보급될 경우에는 解決될 것으로 생각된다.