社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－OS－107 (5)
2008／1／30

データフロー指向ネットワーク分散システムの提案

中村和敬　日比野靖[†1]

現状のネットワーク分散環境から考えると、ネットワーク分散システムはネットワーク抽象によってシステムを表現し、データフロー計算モデルをを採用するのが望ましい。また、ネットワークはルーティングとスイッチングを分離するべきである。本論文ではなぜそのようであるべきか、それがネットワーク分散システムの問題にどのような効果があるのかを論じ、その上でそのようなネットワーク分散システムの実装にどのような機構が必要かを考察する。

A proposal to Dataflow oriented network distributed system

NAKAMURA KAZUTAKA and HIBINO YASUSHI[†1]

From the view of contemporary network distributed computing environment, the network distributed system should be rerepresented by network abstraction and the data-flow computational model should be adopted. In addition, functions of routing and switching should be segregated. This paper discusses why the authors do so, what effect those manners have on the network distributed system problems, and what mechanisms are required for implementation of network distributed systems.

1. はじめに

広帯域ネットワークが普及と計算資源の低廉化によって、様々な機器にプロセッサが組み込まれるようになったが、未だユーザが自由自在に計算機を扱える程には計算機システムは発展していない。

これまで計算機システムは人間との交信しか出来ないスタンドアロンシステムが主流であり、主に組織内部でのデータ処理に用いられた。技術の革新に伴い、計算機とネットワークが普及することによって、様々な組織や個人が管理する計算機システム同士が、直接データをやり取りできる計算機間通信が普及し、計算機ネットワークが一般的な社会インフラとなる可能性を得た。しかしながら、現状のスタンドアロンシステムは、アーキテクチャやオペレーティングシステムの異なる計算機間での機能の結合を前提としていない。しかし、ネットワークで結合することを前提として設計されたアーキテクチャやオペレーティングシステムをもつ計算機であるならば、ネットワークで結合された地理的に分散した様々な機能を結合し、新しい機能を作成することが自在にできるようになるであろう。そのため産業界では近年、ネットワークで結合された計算機を連携させて、一個の計算機システムを構成する為のフレームワークを活用しようという動きが高まっている。

一方、ネットワーク分散システムに関する研究は古くから行われてきており[5]、そのため様々な要素技術が確立されている。しかしながらこれまでのネットワーク分散システムの設計思想は、スタンドアロンシステムの考え方やかつての計算機の性能や通信インフラの状況に多分に影響されており、現在のネットワーク分散環境にはそぐわない部分も多い。筆者らは現状と将来の状況の変化を見据えてネットワーク分散システムをどう設計すべきか、もう一度全体を考え直すべきであると考えた。

その結果、新しいネットワーク分散システムは、ネットワーク抽象によってシステムを表現し、データフロー計算モデルを[*1]を採用し、処理ノード間を接続するネットワークは、ルーティングとスイッチングを分離する方が望ましいとの結論に至った。このような、ネットワーク抽象とデータフロー計算モデルの採用は、ネットワーク分散システムを構成するハードウェア全体のアーキテクチャや利用形態に良く適合している。また、ネットワーク分散システム構築上の問題の内、特に接続性と処理の分散に関わる問題に良い効果があること

†1 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology

*1 計算モデルのことあり、プロセッサアーキテクチャではない

を示す。

本論文ではなぜそのようであるべきか、それがどのような効果があるのかを論じ、その上でそのようなネットワーク分散システムであるには、どのような実装が必要かを考察する。

2. ネットワーク分散システムの設計

2.1 計算機およびネットワーク分散システムの構成

ネットワーク分散システムとは如何なるものだろうか。まずはその構成を見てみよう。

まず、インターネットを見てみる。これはIPアドレスに基づいて、各ルータでルーティング (ネットワーク層の処理) されスイッチング (データリンク層の処理) されることにより、計算機間のネットワークを実現している。計算機はこの上でトランスポート層の処理であるTCP/UDPを利用し、一つの計算機が複数の通信チャネルを扱う事を可能とし、これによって様々なサーバをインターネットに公開している。インターネット上のみでみるなら、これらの通信チャネルを通じて連携するサーバ群によって、一つの機能が実現される。

一方、個々の計算機の内部の様子は、そのシステムソフトウェアのアーキテクチャによって異なるが、例としてマイクロカーネルOS環境の計算機を考えてみよう。このような計算機では、計算機システムの機能はプロセス間通信によって連携するプロセス群によって実現される。そうしたプロセス群の内、IPプロトコルにより外部との通信を行う機能を提供するものがあり、それによってプロセス間通信ネットワークに所属するプロセスがインターネットを通じた通信を利用できるようになっている。

計算機には、この他にも様々な内部バスによる通信や、関数呼出しによるモジュール間通信、共有メモリを用いた通信方法等が存在する。しかしこれらは各々性能特性が異なるものの、いずれも通信という観点からは、同種のものと捉えることができる。これらのネットワークにもやはり、外部ネットワークとの通信機能を提供するノード群が有り、それによって外部ネットワークとの通信を可能としている。

このように、すべての相互作用をネットワークとして見るならば、計算機システムは複数のネットワークから構成されるネットワークであり、個々のネットワークのノードは、自身の所属するノードとだけと通信できるノードと、外部ネットワークとの通信機能を提供する境界ノードとから成る。

このようなネットワークでは、自身の所属するノードとの通信機能と、外部ネットワークとの通信機能の両方を利用することによって全ネットワークにわたるノード間通信を行い、ネットワーク分散システムとしての機能を実現している。そして、これが一つの計算機に限定されている場合が、スタンドアロンシステムであり、複数の計算機で構成されている場合が、ネットワーク分散システムである。

2.2 制御と通信

以上のように、計算機システムはノード間の通信によって全体が制御され一つの大きな機能を実現している。このようにシステムが要素間の通信によって制御され動作するという考え方は、サイバネティクス[9]と呼ばれ、これは計算機システムに限らずシステムを理解し構築する為に有効な考え方である。

これまで計算機システムはプロセッサを如何に動作させるかという観点から作られていたためか、プロセッサを基底とした階層構造によってシステムを捉えるのが一般的である。しかしながらそもそも計算機システムには、プロセッサだけでなくグラフィックカードやネットワークインターフェースカード、ディスク等の複数のハードウェアが存在し、それらが相互に通信することによって一つの機能を実現しており、実際には階層とは無関係なエッジ(辺)とノード(節)による、ネットワーク構造となっている。階層構造はたまたまその接続関係が階層的であるというだけである。

これがネットワーク分散システムでは、処理ノードがネットワーク上に分散しているために、通信による制御という性質がより重要な地位を占めるようになるため、もはや階層構造では捉えきれない。もし無理矢理、階層構造を当てはめようとすれば、それは直接到達出来ないノードへ到達するための、実装上のオーバーヘッドという形で表れ、その計算機システムの複雑さやパフォーマンスの低下につながる。

したがって、新しいネットワーク分散システムではエッジとノードによる、ネットワーク構造によってシステムを捉えるべきである。

2.2.1 相互作用のコスト

このようなネットワーク分散システム上でアプリケーションを設計する時には何を考えなければならないだろうか。

設計とは、目的を実現する為に様々な手法の組合せた場合の利益とコストの均衡点を探索する行為である。計算機システムを通信によって制御がなされるシステムと捉えた場合、コストとはノードのコストと、ノード間の相互作用のコストである。具体的には、ノードのコストは計算時間や利用料金であり、ノード間の相

互作用のコストとは通信のコストであって、それは主に帯域と遅延である。

特にネットワーク分散システムでは通信遅延が無視できないものになる。と言うのも、プロセッサの高速化やネットワークの広帯域化はまだ改善の余地があるが、通信速度については光速の上限が有り、現在の科学はこれを突破する方法を見出せていないからである。光速の上限はスタンドアロンシステムでは通信の範囲が地理的に非常に狭い範囲に閉じていたので完全に無視してしまってよかったが、光速では地球を一秒間に7.5週しか出来ないので、地球規模のネットワーク分散システムでは非常に大きなコストとなる。

また従来計算機システムでは、物理的な地理属性を取り扱うことはなかったが、通信技術的に通信できなくても、物理的に接触できるなら、適切な手法によって相互作用に参加させることができる[7]。現在もユーザは地理的属性を取り扱うことが出来ない資源へのアクセスには、相互作用の為のコストを算出できないので困難にみまわれている。例えば、システム上からはプリンタが何処にあるのか判らないので、どのプリンタに出力してよいか判断できない。一方で知らずに遠方にあるプリンタに出力しても意味がない。また先に述べたように通信遅延は無視出来る程に小さくはならないので、社会インフラとしてのネットワーク分散システムは、常に現実社会の物理的地理属性に適合する形で構築する必要がある。移動体通信やロボット技術が進歩するに伴い、こういった通信コストを概算するのにも役立つ物理的地理属性はますます重要になると考えられる。

したがって、ネットワーク分散システムは物理的/論理的地理属性を取り扱えるようにすべきである。これにより相互作用のコストを取り扱う事ができる。また、正しく扱うことが出来るのであれば、隠蔽もまた容易である。

2.2.2 ネットワーク抽象

これまでのシステムの捉え方としては、ファイル抽象やオブジェクト指向がある。

ファイル抽象は、計算機資源をファイルという形で抽象化し、送信を書込み、受信を読込みという形で表現していた。しかしそのままでは一つの計算機資源は通信チャネルを一つしか持てず、表現力が足りない。そのためプロセス間通信をファイル抽象によって表現する場合は、ディレクトリにそのプロセスの複数のポートをファイルとして並べるなどといった形を取る。これはファイル抽象では一つのファイルは入出力ポートを一つづつしかとることができないからである。これに対して、ネットワークのトランスポート層では複数のポートが扱える。

また、ファイル抽象はファイルシステムにマップされる為、代表的なファイルシステムの階層型ディレクトリモデルに束縛される。階層型ディレクトリモデルでは、ネットワーク構造を表現出来ないので、リンクファイルの氾濫という結果を招く。

オブジェクト指向では、計算機資源をオブジェクトという形で抽象化するので、ファイル抽象で生じるような問題は減るものの、ネットワークという視点が欠けているために、通信のコストを取り扱うことが出来ない。また、データと処理をひとまとめにして考えてしまうために、実際に転送されるものが何かがあいまいになってしまうだろう。

したがって、ネットワーク分散システムには、すべての資源をネットワークのエッジ(辺)とノード(節)という形で表現するネットワーク抽象に依るインターフェースが最適であると考えられる。利用可能な複数ネットワークを、一つのネットワークに統合し、それを抽象ネットワークとして捉える。一つの抽象ネットワークを構成するノードは、別々のネットワークに所属するノードである場合もあるし、同一のネットワークに所属する場合もある。このような抽象ネットワークのノード間で通信させて新しい機能を実現する。逆に言えば、新しい機能を実現するために、抽象ネットワークを構成するのである。また、ある程度以上の複雑さをもつノードを表現する為に、OSIの7階層モデルの要求を尊重すべきであり、最低でもトランスポート層を取り扱えるべきである。

2.3 並列制御フロー

これまで計算機での支配的な構成は、単一プロセッサの計算機であったため、利用者からみれば単一制御フローを仮定していた。そのため、従来のシステムソフトウェアは、利用者に単一制御フローのイメージを維持しつつ、いかにしてプロセッサの多重化を行うかという観点から構築されていた。これが、プロセス概念に基づくタイムシェアリングシステムである。ネットワーク分散システムにおいても、単一制御フローの振舞を提供できるよう、シングルシステムイメージの構築に力が注がれていた。

しかしながら現実の計算機では、単一プロセッサの計算機であっても、グラフィックカード、ネットワークインターフェースカード、ディクスコントローラといった、独立に動作する複数のコンポーネントが存在し、それらが並列に動作・連繋することによって一つの機能を実現しており、並列制御フローのシステムと

なっている。また、現状のソフトウェアでも、プロセスやスレッドの相互作用を利用した構成がとられている。ハードウェアにおいてもプロセッサのクロック周波数が限界に達しつつある現在では、プロセッサのマルチコア化の流れが進行している。さらに、目的に特化した専用プロセッサが広く利用されるようになってきている。このようにネットワークに接続されていないスタンドアロン環境においても、システムの並列制御フローシステムとしての性格が明確になりつつある。そのため、並列計算モデルの言語も注目されつつある。

ネットワーク分散システムはそもそも、複数のプロセッサを持つ並列制御フローのシステムである。また、そもそもの分散システムの目的の一つが並列計算にあったのに、これを無理矢理単一プロセッサモデルに当てはめようとするのは、問題をややこしくするだけである。

血路として、現状の計算機システムを正しく捉え有効に活用するために、新しいネットワーク分散システムでは並列計算モデルを採用すべきである。

2.3.1 データフローに依る制御

有力なプログラミングパラダイムとして、関数プログラミングがある。処理(計算)はすべて関数によって行われる。制御動作の観点からこれを見ると、関数呼出し(コール)と復帰(リターン)である。ネットワーク上では、このコール・リターンは、リモート・プロシジャー・コールという形で実現され、システムの形態としては、クライアント・サーバ方式という形に変化して取り込まれている。しかしながら、この考え方もまた暗に制御フローが単一のものであると言う前提に基づいており、ネットワーク分散システムの並列制御フローシステムとしての性格を活用する際の妨げとなっている。

また、問題は、ネットワーク分散システムの活用の面だけでなく、性能面にも存在する。コール・リターンを行う関数プログラミングでは、暗に呼出し元に返り値を戻すという使い方を想定しているが、ネットワーク分散システムでは処理を行うノードが分散しているので、必ずしも次の処理を行うノードが、呼出元と同じネットワークノードにあるとは限らず、必ず呼出元に返り値を戻すコール・リターンでは、余計な通信を増やす原因となる。伝送速度(bps)にテクノロジによる上限があり、回線には光速という物理的限界がある以上、回線の記憶効果による通信遅延は大きなものとなるので、実用上の観点から無駄な通信を抑えることは必要である。

翻ってネットワーク分散システムをみてみると、ネットワークで結合されたシステムであるので、必然的に分散メモリである。したがって、引数渡しメカニズムは必然的に値渡しが基本となる必要がある。またどんな処理でも必要なデータが無ければ開始することは出来ないが、これはデータ駆動方式であることを意味する。

現状のシステムを正しく捉え有効に活用するために、新しいネットワーク分散システムでは、上記の理由から、計算モデルとして、データフロー計算モデルを採用するのが自然である。

これは並列計算モデルであり、値渡しである。さらにデータの流れによって制御を行うというデータフロー計算モデルでは、各々の処理単位で制御が独立している。このことは、ネットワークに分散した処理単位が通信によって処理を進めるというネットワーク分散システムの性格に良く適合する。例えば処理単位で制御が独立しているので、ノードをネットワーク上に適切に配置することが可能となり、通信コストを削減できる可能性があり、また障害時の影響を障害ノードのみに局限できる。ネットワーク抽象だけでは、その上でシステムを如何に構築するかという指針がまったくないが、データフロー計算モデルのデータ駆動原理という制限を導入することによって、システム構築が容易になると考えられる。

3. 複数のネットワークの統合

計算機システムは、複数のネットワークから構成されるネットワーク上での、様々な機能の相互作用として捉えられることがわかった。そこで次に、異なるプロトコルの複数のネットワーク統合する問題を考える。

複数のプロトコル間でのプロトコル変換の方法としては、1対1のプロトコル変換を相手となるプロトコルすべてに対して実装する方法がある。この原始的方法では、N種のプロトコルに対して、Nの2乗に比例する数のプロトコル変換器の実装が必要なり、またすべてのノードがこれを取り扱う事が出来る必要が有った。しかしながらこの手法は、新しい種類のノードを追加したり、他システムから取り込もうとした場合の障害となる。

一方で、なにがしかのグローバルなネットワークにマッピングする方法が考えられた。代表例がIPネットワークである。IPネットワークでは、IPアドレスとIPプロトコルをピポッドとしてすべてのネットワークを統合することを可能とした。しかし、その場合システム構成が常にそのグローバルなネットワークの状

態に依存し、柔軟性を大きく損なうことになる。またグローバルなネットワークはプロトコルが複雑であるので、内部バスなどのような小さく単純で高速な実装が求められるような領域で採用するのは難しい。

これらの問題は、OSI7 階層モデルに於けるデータリンク層とネットワーク層を不可分のものと考えている事が原因である。個々のノードにデータを転送するスイッチングはデータリンク層の役割であり、その際の経路を設定するルーティングはネットワーク層の役割である。あらかじめルーティングを済ませるネットワークであるならば、個々のノードは転送時にはデータリンクの取扱い方のみを知っておけばよい。またルーティングは通信を行うノード以外が行うこともできるが、その場合、ノードはネットワーク層のプロトコルを知る必要は一切無く、自身が所属しているネットワークに対してのみインターフェースを持てば良い。これはノードやネットワークの追加/削除に対して大きな柔軟性を提供する。

以上の考察から、新しいネットワーク分散システムでは、データリンク層とネットワーク層の機能をわけるべきである。

3.0.2 分散多重管理

今度は、利用形態の面からネットワーク分散システムを見てみよう。

スタンドアロンシステムに於いて、管理の形態は集中管理が一般的である。これはこれまで支配的な構成であった単一プロセッサシステムでは、プロセッサの特権命令を扱えるユーザにすべての権限が集中するので、その構成上当然とも言える。しかしネットワーク分散システムに於いては、管理の形態は一変し、また利用面においても大きな変化があると考えられる。

まず全体を俯瞰すると、ネットワーク分散システムは、ネットワークに接続された計算機の集合であるが、そもそもネットワークは異なる主体同士が通信する為のものであるので、この全体を管理する管理者を想定することは、その構造からして不可能であるということである。自ずと「個別に管理されている計算機の集合」という形でネットワーク分散システムは実現される。ここでネットワーク分散システムが単にネットワークに接続されたスタンドアロンシステムと大きく異なる点は、一人の利用者が管理者の異なる複数のサービスを利用すると言うことである。現実にも一人のユーザが自分の家の計算機システム、ネットワークプロバイダの計算機システム、会社の計算機システムといった形で管理者の異なる複数のサービスの提供を受けている。

しかし従来のスタンドアロンシステムでは、一人の管理者が複数の利用者にサービスを提供する事は考えられていたが、一人の利用者が複数の管理者から異なるサービスを提供される場合は考慮されていない。そのためこういった管理者の異なるサービスを統合利用する際の障害となっている。

と言うのも、現在、スタンドアロンシステムにおいて提供されている特別なサービスの多くは、ネットワーク分散システムに於いては、ユーザが自身の資源にアクセスを許可するだけで実現できるものだからである。これはスタンドアロン環境の UNIX がネットワーク環境に進出するに伴い、様々なネットワークアプリケーションプロトコルが開発された経緯を見れば明らかである。

しかし、管理者の異なる複数のサービスを利用して、一つの新しいサービスを提供しようとすると、単なるグループアクセスの許可などといった手法では実現でききない。サービスを構成する要素が、異なる管理者の権限に属するからである。これを実現するがめには、例えば、常に管理者から利用者に権限が渡し、それを権限を受け取った利用者は管理者として別の利用者に権限を渡すという形を取る必要があるからである。

したがって、新しいネットワーク分散システムでは、全てのユーザがサービスの管理者であると同時に利用者としての性格も持てる必要がある。

3.0.2.1 プライベートなネットワーク集合

一方、ネットワーク分散システムを一ユーザの視点から見るとどのように見えるだろうか。

ユーザはネットワーク分散システム全体を知ることはない。なぜなら、ネットワーク分散システムは複数のネットワークによって構成されるものの、ユーザが利用出来る境界ノードがユーザごとに異なり、また境界ノードの数が膨大であるために、ネットワーク分散システムの全体としてのネットワークを知ることはほぼ不可能である。個々のユーザは各々異なったネットワークの集合をネットワーク分散システムに於ける通信リソースとして利用できる。これはつまり、新しいネットワーク分散システムに於ける抽象ネットワークは、常にユーザ毎に異なるものであり、プライベートなものである事を意味する。一方で実際に通信機能を提供する基底ネットワークと境界ノードはパブリックなものである。

したがって、新しいネットワーク分散システムでは、ユーザごとに個別のアドレス空間を取り扱える必要がある。

3.0.3 ルーティングとスイッチングの分離

これらを鑑みるに、新しいネットワーク分散システムは、ルーティングとスイッチングを分離したネットワークによって接続性を実現すべきだとわかる。

実際に通信を行うために必要なのは個々のネットワークであるが、これは既に整備されており、本来すべてのノード間で伝送が可能である。したがって、これらのネットワークをデータリンク層として扱い、正しくルーティングを行えば、すべてのノード間の通信が実現できる。言い替えるならば、ネットワーク上の処理ノードの結合は、トランスポート層として扱い、処理ノード間はネットワークアドレスでルーティングされる。処理ノード間の経路は、データリンクとして捉える。すなわち、経路上の中継ノードでは、スイッチングのみを行いネットワーク層の処理であるルーティングは行わない。複数のネットワークから成る場合は、ネットワーク間を接続する境界ノードが必要となるが、ルーティングを事前に行っておくことにより、これらのノードもスイッチを行うだけでよい。

これにより、複数のネットワークの各々のネットワークに所属するノードを、一つの抽象ネットワークに統合し、相互に通信させて新しい機能を実現することが可能となる。

また、この方式ならばルータが複数存在する事が出来るが、これはつまり複数のアドレス空間が多重に存在しうると言うことを意味する。

以上の考察から、名前空間の柔軟な制御が可能となり、透過性を実現することが容易になるだけでなく、複数の異なるネットワークに所属するノードを、一つの同じネットワークに所属するノードの様に見せることができるようになる、バーチャルネットワーク機能や、複数のノードで構成されるサブネットワークを一つのノードとして扱うバーチャルノード機能を簡単に実現することが出来、これは権限管理/負荷分散に有用である。

ルータ/スイッチに認証/承認機構を埋め込むことによって、抽象ネットワークそれ自体に認証/承認の役割を負わせる事が出来るようになるので、分散管理も容易である。pそうでありながら各々のノードに合った適切な方法で通信させることが出来るので、性能の低下もないだろう。

3.1 ま と め

以上から、新しいネットワーク分散システムは、ネットワーク抽象によってシステムを表現し、計算モデルとしてにデータフロー計算モデルを採用し、ルーティングとスイッチングを分離したネットワークによって接続するシステムである事が望ましいとわかる。

4. ネットワーク分散システムの問題

ネットワーク分散システムを実現するには、いくつかの問題を乗り越えなければならない。本方式では、特に接続性と処理の分散に関わる問題に良い効果がある。ここではそれをまとめておく。

4.1 並列処理と透過性

まず移動/位置/複製透過性といった名前空間に纏わる透過性は、バーチャルネットワーク機能やバーチャルノード機能によって解決できる。

また並列/並行透過性といった処理に関わる透過性は、並列計算モデルを導入することによって解決されている。

処理の同期の問題については、データフロー計算モデルを採用し処理の開始と終了のタイミングを決めることによって、通信によって同期がとられるので解決されている。

またネットワーク分散システムでは、デッドロックを解決するのが難しいが、データフロー計算モデルではノードが正しく動作する限り、十分な長さのキューを設けてやることによって、一つの処理単位が複数のノードを占有することがない、すなわち、一つの処理単位は一つのノードを占有するだけであるので、デッドロックを避けることが可能である[2)]。このキューの長さはノードの多重度によって決まるので、事前に見積もることが出来る。

4.2 耐障害性

ネットワーク分散システムでは、不安定なネットワークを用いた場合、障害が発生することがスタンドアロンシステムよりはるかに多い。故に、耐障害性を考慮に入れる必要がある。この時、障害自体を防止するのは非常にコストがかかる。システムを構成するノードも信頼性を上げようとしているのだから、それを生かす形で対応するほうがよい。システムソフトウェアはノードの障害について考えるよりも、それらを連携させた場合の連鎖被害の軽減に力を入れるべきである。

本方式では、データフロー計算モデルを選択することにより、各々のノードの制御が独立するので、各々のノードの信頼性に応じた耐障害手法を採用でき、また障害を検知できるならば、障害の発生したノードのみで対応策を講じる事が出来る。このようにオーバーヘッドを考慮に入れて、最適な耐障害手法を採用する事が可能となる。

4.3 接 続 性

ネットワーク分散システムの第一の目的である計算

機の統合の為には、あらゆる計算機資源にネットワーク経由で接続できなければならない。本方式に基づくシステムでは、抽象ネットワークによってすべての資源間の接続が可能となっている。

4.3.1 セキュリティと権限

実装が正しくなければ安全でないが、実装の正しさの詳細はその実装方法によって異なるので、ここでは論じない。

計算機システムの権限管理は、基本的に要求を受理するか拒否するかという動作として表れる。したがって、認証/承認のメカニズムによってユーザが権限を持っているか確認するための仕組みが必ず必要である。ネットワーク分散システムの場合は、ノードの認証/承認とネットワークの認証/承認の二つの問題にわけられるが、ネットワークも複数のノードから成るシステムであるので、結局のところノードの認証/承認メカニズムが問題となる。

またネットワーク分散システムの場合は、ネットワークの名前空間の制御による権限管理手法も存在する。これはルータが行うもので、スイッチが適切な認証/承認メカニズム によって守られており、ノード毎に扱えるデータリンクを制限できるならば、実ノードの所在を隠蔽することが可能となり、権限管理と区分化の有効な手法となる。

一ヶ所のノードに権限情報を集約する方式の場合、負荷が際限無く大きくなる可能性があるが、本方式ではユーザごとに権限の管理を行い、また名前空間の制御に依る権限管理を取り入れることにより、これを分散ルーティングの考え方に基づいて分散出来るので、柔軟かつ大規模な権限管理を実現できる。

4.3.1.1 Firewall と NAT

現状を見てみると、IPv6 を支持する人々に代表される、接続性を提供するためにグローバルで広大なアドレス空間を用意して、それにすべてマッピングしてしまおうという主張をする人々がいる。

それに対してセキュリティ上の観点から懸念を唱える人々がおり、Firewall と NAT に依る防御を支持している。Firewall と NAT は接続性を制限するために両者は対立しているが、こういった問題はルーティングとスイッチングを分離した抽象ネットワークによってシステムを構築することによって解消する。

Firewall や NAT は区分化の発想であり、これはすべての安全保障に於ける基本的な手法である。Firewall はネットワークの権限確認であり、ネットワークの一部を隠蔽するものである。NAT も激しく攻撃されているが、実際の機能としてはマイクロカーネルに於ける IP プロトコルスタックと変わりはない。NAT の問題もそれがその場しのぎの対応策であり、きちんとしたサービスとして作られていない事による。

接続性の為にすべてがグローバルなアドレスにマップされている事が必要だという主張は間違いである。実世界において人々は相手の家を知らずともてつき合うことが出来る。これはつまり落ち合う場所を決めておけばよいという事で、NAT がきちんとサービスとして作られているならば問題とならない。

ルーティングとスイッチングが分離された抽象ネットワークの場合、ルーティングがユーザ毎に異なるので、境界ノードが承認によって要求毎に権限を管理することによって、区分化による安全保障が実現される。名前空間が分離されているので、権限のないユーザは境界ノードの内側を窺い知ることは出来ない。境界ノードがプロトコルスタックサービスとして実現されているので、必要に応じてグローバルなネットワークのポートをリスンする事もプログラムによって処理する事ができ、接続性に関する問題も解決される。この場合に問題となるのは、ポート番号によってサービスが固定されているウェルノウンポートという考え方と、TCP ポート数の少なさであろう。

4.4 その他の問題

本方式はこれらの問題は直接的に解決は出来ない。しかしより良い解決策を導く助けとなる可能性がある。

4.4.1 リソースマネジメント

リソースマネジメントの問題は、アロケーションとスケジューリングの問題にわけられる。このうち個々のノードに関するスケジューリングは従来のスタンドアロンシステムの問題とあまり違いはない。ノード全体に関するスケジューリングは、全体情報の収集が難しいので困難が予想される。今後の課題である。

リソースアロケーションは、ノードや実ネットワーク毎の特性を考慮に入れて抽象ネットワークの最適な位置にノードを配置する、あるいは最適な位置のノードを選択する問題である。サブネット (サブグラフ) のすべてのノードを検査するには、総当たりでは指数時間のアルゴリズムになってしまうので、よいアルゴリズムを考える必要がある。ネットワーク抽象によって通信コストが取り扱えるので、様々なプロセッサ配置アルゴリズム[5] を用いる事ができる可能性がある。

4.4.2 規模拡張性

ネットワーク分散システムへの要請の一つとして、規模の拡張性がある。これは多くの場合、負荷分散の問題である。この問題はシステムの目的のよって異なるものの、本論文の方式はいくつかの利点がある。

まずノード毎に処理が独立しているので、状態を持たないノードは単に並列化することによって負荷を分散できる。またネットワーク抽象によって、各処理の依存関係や転送量が明確になるので、よりよい分散方法を発見する手助けとなるだろう。

5. ネットワーク分散システムの構築

5.1 要素技術

ネットワーク分散システムには既に多くの要素技術が開発されている。

マイクロカーネルは、通信による制御によって計算機システムが実際に構築できる事を示し[6]、 またL4カーネルによって、性能も決してモノリシックカーネルに劣るものではないことが示された[4]。

データフロー計算モデルとアーキテクチャについては、計算モデル、プログラミングインターフェース、評価機構など様々な点からの考察や実装が行われている[2),8)]。

暗号技術は、特に公開鍵暗号の実現によって、セキュリティ技術を大きく前進させた。この成果は既に様々な形で計算機システムに取り入れられており、その有用性が示されている[1),3)]。

5.2 実　　装

新しいネットワーク分散システムの為に作るべきものは非常に多いが、まずは抽象ネットワークの実現が第一歩である。そのためにルータとスイッチのプロトコルを決めなければならない。これらはいずれも二つのアドレスのペアを引数にとるものとなるはずで、良く似たものに成るはずである。また、データフロー計算モデルを選択したことによって、ノードもまた必要があるなら、複数の入力/出力アドレスのペアを引数として受け取る事様な動作をするはずで、ルータ/スイッチの実装は、システム全体の雛型となる。

実装の際には様々な既存ネットワークをデータリンク層として使うはずなので、論理的なインターフェースとバイナリインターフェースを、きちんと区別して決める必要がある。

また、デッドロックを避けるために、リンクは寿命付で生成し、必要に応じてルータが延命させてやるといった手法を取る必要がある。

また、バーチャルネットワークとバーチャルノードは、ネットワークに依る権限管理の鍵となる重要な機能である。これらが正しく区分化に依る防御性能を発揮するには、スイッチがルータ毎に利用可能なリンクを管理出来る必要がある。これを直接設定するのは大変繁雑な仕事になると思われるので、継承ルール等によって自動的にルータを生成する仕組みが必要になる。

6. ま と め

本論文では、新しいネットワーク分散システムとして、ネットワーク抽象によってシステムを表現し、データフロー計算モデルを採用し、ルーティングとスイッチングを分離したネットワークによって接続する計算機システムを提案した。この計算機システムは、ネットワーク分散システム構築上の問題の内、特に接続性と処理の分散に関わる問題に良い効果がある可能性を示し、それがどのように実装されるかを考察した。

今後はこの考え方に基づいた基本サービスを実装し、その有効性を実証していく。

参 考 文 献

1) F.Cusack and M.Forssen. Generic message exchange authentication for the secure shell protocol (ssh), 2006.
2) Wesley M. Johnston, J. R. Paul Hanna, and Richard J. Millar. Advances in dataflow programming languages. *ACM Comput. Surv.*, Vol.36, No.1, pp. 1–34, 2004.
3) S. Lehtinen and Ed. C. Lonvick. The secure shell (ssh) protocol assigned numbers, 2006.
4) J. Liedtke. On micro-kernel construction. In *SOSP '95: Proceedings of the fifteenth ACM symposium on Operating systems principles*, pp. 237–250, New York, NY, USA, 1995. ACM.
5) Andrew S. Tanenbaum. *Distributed Operating Systems.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
6) AndrewS. Tanenbaum and AlbertS Woodhull. *Operating systems: design and implementation, 3/E.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.
7) D. Waitzman. Standard for the transmission of ip datagrams on avian carriers, 1990.
8) PaulG. Whiting and Robert S.V. Pascoe. A history of data-flow languages. *IEEE Ann. Hist. Comput.*, Vol.16, No.4, pp. 38–59, 1994.
9) Norbert Wiener. *Cybernetics, Second Edition: or the Control and Communication in the Animal and the Machine.* The MIT Press, 1965.