第2章　関連研究

2.1　名前解決

2.1.1　名前

計算機から利用される資源を指し示す際に，各種の名前が使用される．これらの名前は，計算機内部で使用される低水準なものから，計算機のユーザが望む資源を指し示すために使用する高水準の名前まで多岐に渡る．

名前が低水準であればあるほど，計算機ハードウェアやオペレーティングシステムには扱いやすく，計算機のユーザには扱うことが困難なものとなる．このため，高水準の名前から低水準の名前へと変換を行うことで，両者の仲介を行うことが必須となる．この仲介を行うシステムを名前解決システムとよぶ．

名前解決システムは，ある値からそれに対応する他の値に変換する関数と考えることができる．すなわち，空間 A から空間 B に名前を対応づける名前解決関数 f は，

$$a \in A;\ b \in B;\ b = f(a)$$

となる．このときこの関数 f の性質によって名前解決はいろいろな特性を持つことになる．この名前解決を図示すると図 2.1 のようになる．

名前解決を行うとき，ある名前に対してその解決される実体がその解決を行う場所によって異なるとき，この名前解決システムは位置依存であるという．例えば，計算機Ｘにおける解決と，計算機Ｙにおける解決が異なるような場合である．これに対して，場所によらずに名前解決先が一意に定まる性質を位置独立という．また，名前の中にその名前を管理しているサーバや，その名前の指す実体のある場所など，位置に関する情報が含まれないという性質を位置透明性という．

2.1.2　ファイルシステムと名前

本節では，名前解決の代表的な例として，ファイルシステムにおける名前管理について述べる．

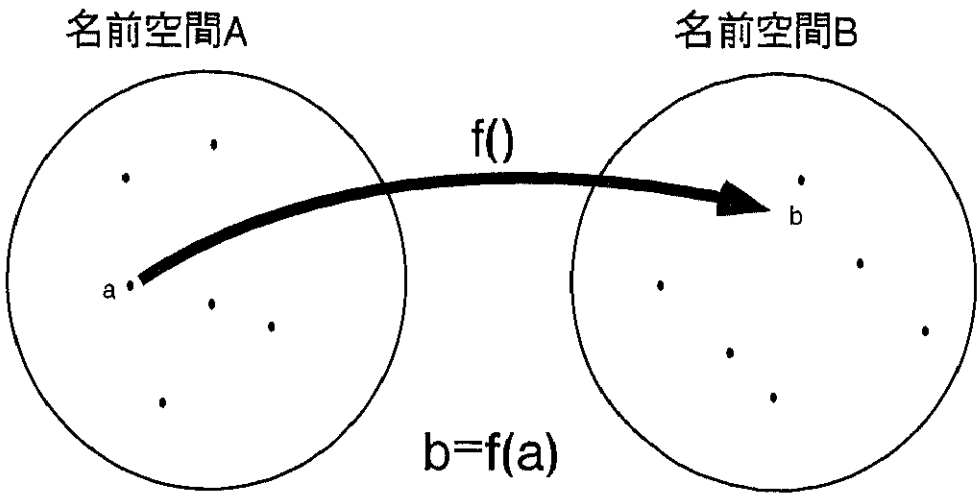

図 2.1: 名前解決関数

ファイルシステムにおいては，名前解決システムの役割は，ユーザがあたえたファイル名に対してその物理的な実体を対応づけることにある．Unix 等のファイルシステムは基本的に木構造を成している．この基本構造に対し，ファイルシステムではリンクという機能によって，同一の実体に対し複数の名前を付けることを可能としている．

Unix ファイルシステムにおけるリンクには 2 種類の形態がある．

ハードリンク (図 2.2) とは，同じ実体に対し単純に複数の名前を割り当てるものである．すなわち

$$a1, a2 \in A;\ \ b \in B;\ \ b = f(a1) = f(a2)$$

という名前解決関数となることを意味する．

シンボリックリンク (図 2.3) は，名前解決先の実体が示す内容を，さらに名前として扱うことで，同じ実体に複数の名前を付けるものである．これは，

$$a1, a2 \in A;\ \ b, b' \in B;\ \ b' = f(a2),\ \ b = f(a1) = f(g(b')) = f(g(f(a2)))$$

となることを意味する．$g(x)$ は，x の実体を名前解決システムが評価した値を示す．

分散ファイルシステムでは，複数の計算機の間でファイルシステムを共有する．この時，各計算機間でのファイルシステムに対する名前解決の差異によっていろいろな方式が考えられる．Sun の Network File System(NFS)[28] では，各計算機のファイルシステムに対し，ネットワーク上の他の計算機のファイルシステムを部分木として結合可能である．この操作をリモートマウントと呼ぶ．これは，名前解決関数の定義域の一部を他の関数で置き換える操作と考えることができる．すなわち，名前空間 A

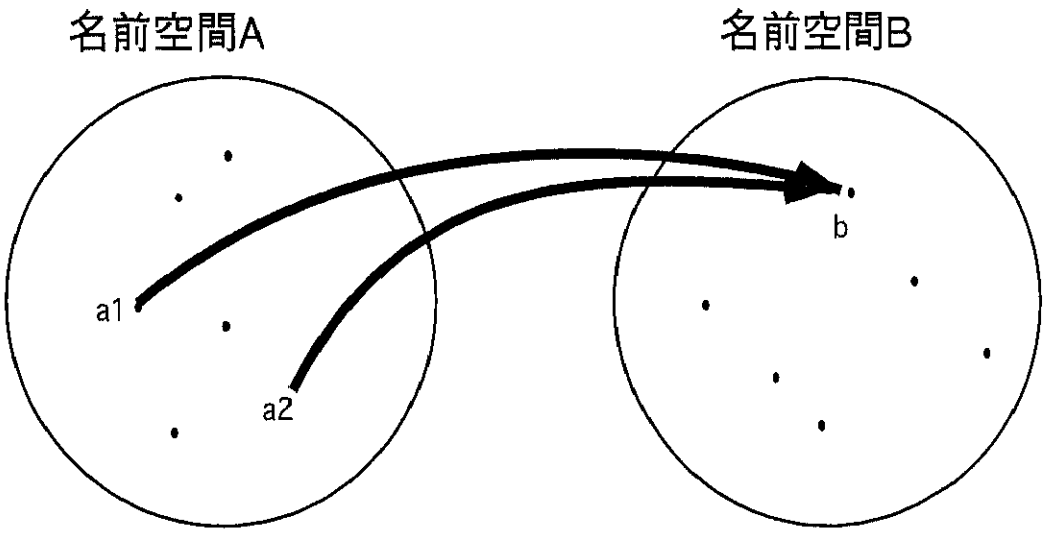

図 2.2: ハードリンク

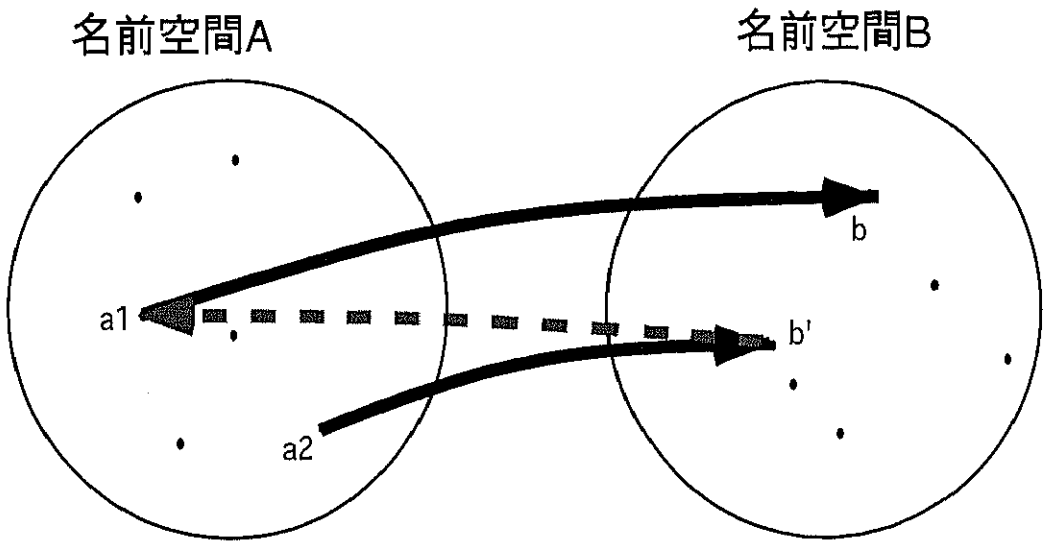

図 2.3: シンボリックリンク

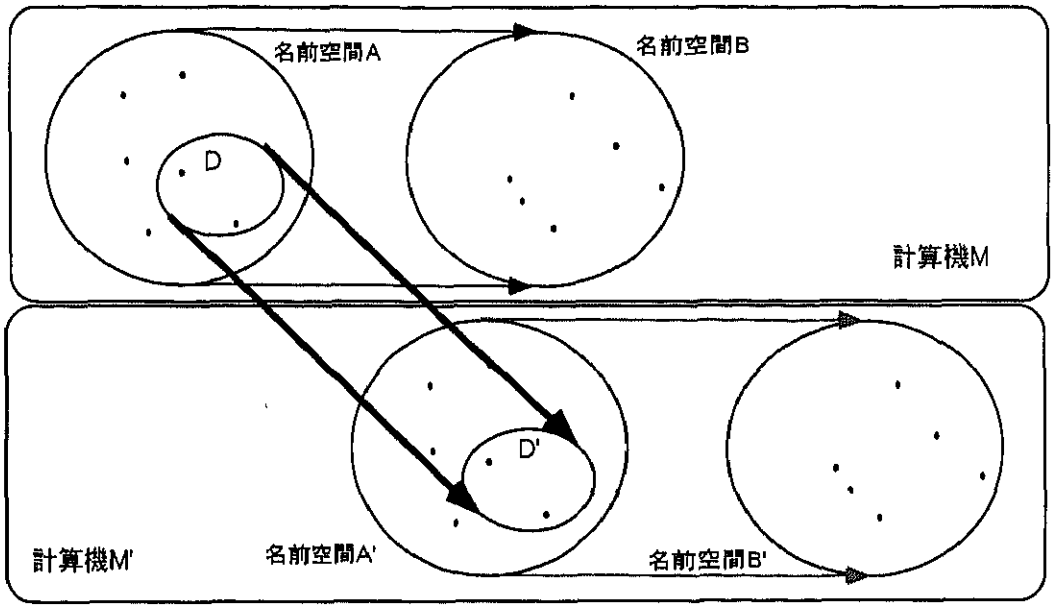

図 2.4: リモートマウント

中のある名前を prefix として持つ名前の集合 D に対し，他の計算機をアクセスする名前解決関数 f' を使用するものである (図 2.4).

リモートマウントを用いると，他の計算機に存在する資源を同じ計算機に存在する資源と同様に扱うことができる．この機構によって，分散環境に散在する計算機間で同一の名前空間のイメージを得ることが可能になっている．しかし，これらの空間のイメージの作成には，計算機管理者の多大な労力が必要となる．これは，各計算機において，どの計算機のどの部分木をマウントするか等の設定はすべて管理者によって行われるためである．

2.1.3 単一名前空間

このように，単一の名前空間を各計算機毎の管理によって維持することは非常に多大なコストがかかることになる．これに対し，名前空間を共有する全ての計算機，または全てのユーザに対し，全く同じ単独の名前空間が提供されるシステムを単一名前空間システムと呼ぶ．分散環境での協調作業においては，複数のユーザの間で情報を交換，あるいは共有する機能が必須である．単一名前空間は，分散環境での協調作業において，その交換・共有の機能のベースとなるものとして重要なものである．ユーザは，単一名前空間を介して協調処理を行うことになる．

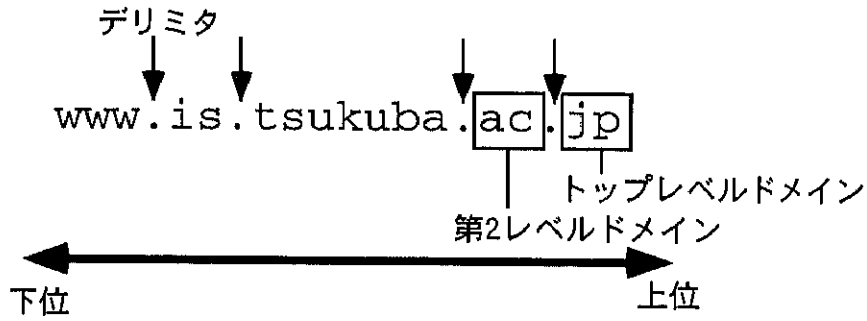

図 2.5: DNS における名前の一例

単一名前空間を単純に構成するためには，名前管理を行うサーバを一つおくことが考えられる．そして，そのサーバを介して全てのユーザが名前解決を行うことで，単一名前空間を構成することが可能である．しかし，この方法では全ての要求が一つのサーバに集中することになる．また，広域分散環境など，ユーザとサーバの間の通信が高コストとなる場合には，キャッシュを用いてサーバとのトラフィックを削減することや，サーバの複製を用意することで単独のサーバに負荷が集中することを防ぐことが重要である．

2.1.4 Domain Name System

具体的な名前空間システムの例として Domain Name System(DNS)[18, 19] を取り上げる．

DNS は，広域分散環境における名前解決システムである．その目的は，広域空間に散在する各種の資源をリモートマシンから指し示すことにある．DNS が提供する名前空間は，広域環境内で単一な名前空間であり，その空間は木構造をなしている．

名前空間は複数のサーバで分散管理されている．名前空間はゾーン(zone) と呼ばれる空間に分割されており，ひとつのゾーンの情報は，ひとつ以上のサーバに保持されている．複数のサーバにゾーン情報が置かれる場合，主となるサーバをプライマリ・マスタサーバ(primary master server) と呼び，プライマリ・マスタサーバから情報を受けとり，プライマリ・マスタサーバの複製として動作するものをスレーブサーバと呼ぶ．

DNS における名前は図 2.5 のような形をとる．“.”(ピリオド) は名前のデリミタであり，木構造をした名前空間の辺の部分に相当する．その間の英数字列は木の節，あるいは葉に相当する．最も右側にある名前が木構造の最上位，すなわちルートに存在

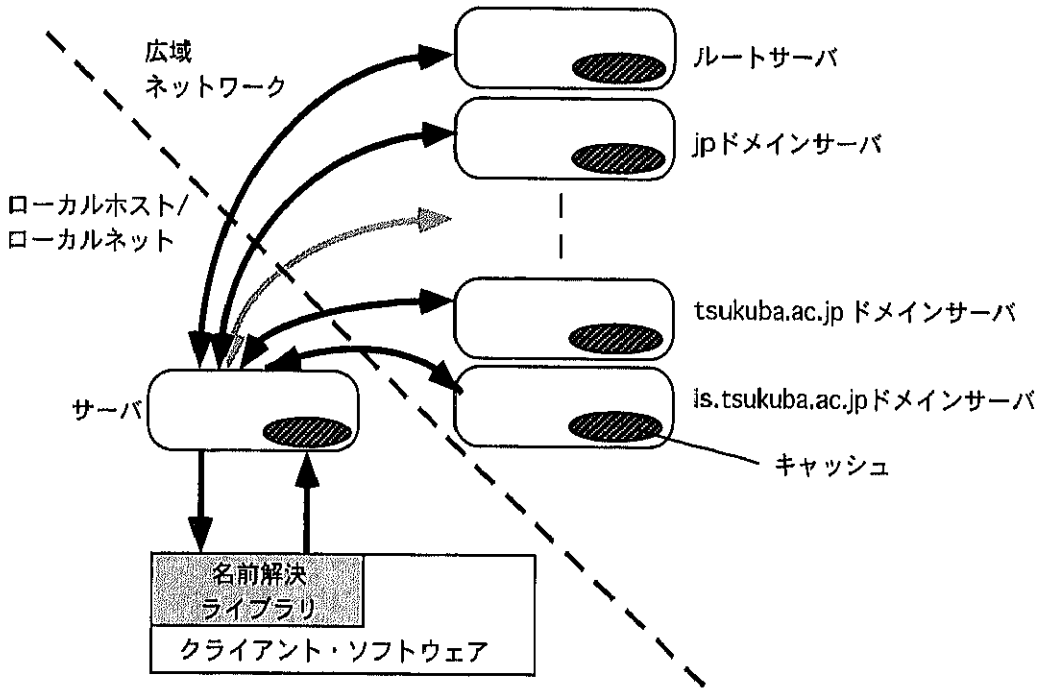

図 2.6: クライアントから見た DNS の名前解決システム

する名前であり，トップレベルドメイン名と呼ばれる．より左にある名前をサブドメインと呼ぶ．Unix ファイルシステムにおける木の上位・下位の関係と反対の関係になっている．

各ゾーンは，木構造をした単一名前空間の部分木に相当する．各ゾーン間の関係は，各ゾーンがそのサブドメインを管理しているゾーンのサーバアドレスをポインタとして保持することによって指定される．

名前解決は以下の手順で行われる．名前解決システムをクライアント側から見ると，図 2.6 のようになる．クライアントにリンクされた名前解決ライブラリは，指定された DNS サーバに対して名前解決要求を行う．クライアントからの要求を受けたサーバは，要求がそのサーバが管理している (プライマリ・マスタサーバかスレーブサーバとして動作している) ゾーン内の名前であるかを調べ，そうであるならば，要求に対する返答を行う．そうでない場合は，サーバ内のキャッシュを使って返答を行う．

要求がそのサーバで解決できない場合，サーバはルートサーバへ問い合わせを行う．これは，他のサーバがどのゾーンを管理しているかという情報をもっていないためである．結果として，ルートサーバには過大な負荷が集中することになる[1]．DNS サー

[1]例えば，ルートサーバの一つである F.root-servers.net は一日に 330,000,000 件以上の問い合わせを受けているという報告 [15] がある．

バにはforwarderとして名前解決の移譲先のサーバを設定する機能が存在する．しかし，この機能は，組織内の特定のサーバに問い合わせを集中させることを目的としたものであり，この値によってサーバをカスケーディングし，問い合わせを次第に遠方のサーバへと転送するということは設定できない [1].

DNSにおける名前の一貫性管理は，基本的に名前のキャッシュ可能期間の設定によって行われる．ゾーンを定義するファイルには，

- スレーブサーバがプライマリ・マスタサーバからゾーンデータをコピーする間隔
- コピーが何らかの理由で失敗した場合，リトライする間隔
- コピーした情報の有効期限

の定義が含まれている．プライマリ・マスタサーバとスレーブサーバの間の一貫性は，プライマリ・マスタサーバにおいてゾーンの情報の有効期間を定め，情報がスレーブにコピーされてからその指定された期間の間は有効であるとする．また，各名前にもそれぞれ有効期限が定められており，問い合わせをクライアントやサーバがキャッシュする場合は，この期限内のみキャッシュしておくことができる．これらの時間定数を小さくすることで常に最新の状態を保つようにすることや，大きくすることでキャッシュを有効に機能するようにするなどのことが可能になる．

しかし，プライマリ・マスタサーバでの変更が直ちに反映されないことは運営上問題が生じるため，現在ではDNS NOTIFY[32] とよばれる更新通知機構が実装されている．また，その際ゾーン全体の情報を転送する (AXFR) と，変更されていない部分の情報の転送が無駄となるため，Incremental Zone Transfer(IXFR) [23] という，ゾーンの変更部分のみを転送する手法が提案されている．

DNSは，ホスト名からIPアドレスへの変換をすることが主要な利用目的である．この変換を利用することで，簡単な負荷分散をすることが可能である．これは，DNS round-robinと呼ばれ，同じホスト名に対し複数のIPアドレスを割り当て，これらを順に使用することで簡単な負荷分散を行うものである．しかし，解決先IPの実際の負荷とは無関係に分散させるため，クライアントの要求する処理の負荷に大きな変動があるときには効率的に負荷分散を行うことはできない．

2.2 名前空間のカスタマイズ

単一な名前空間をユーザがカスタマイズ可能とするものとして，種々の手法が提案されている．本節ではそれらの関連研究と本研究との関連を述べる．

2.2.1 Prospero

Prospero[20, 22] は，Virtual System Model[21] に基づいた分散ファイルシステムである．Prospero はインターネット上の各種の資源をファイルシステムにマッピングするためのツールを提供する．

ツールには 2 つの種類のものが存在する．一つ目は，union link で，link の指す先のディレクトリの内容を，link のあるディレクトリにあるように見せかけるものである．これは，複数の情報源から情報を収集し，単一のディレクトリにまとめるために使用される．二つ目はフィルタである．これは，ディレクトリを引き数として受け取りディレクトリを返り値とする任意の関数を使用することが可能である．たとえば，ディレクトリ内に存在するファイルをそのプロパティに応じて仮想的に作成されたディレクトリに分配するなどの利用法がある．

これら 2 つの機構により，ユーザーは自由に独自の名前空間を構成することが可能である．

2.2.2 Active Names

Active Names[31] は，名前を移動可能なプログラムの列にマップすることで，名前の指す対象を自由にカスタマイズ可能とするものである．ただし，これらのカスタマイズはユーザではなく名前空間を作成する管理者が設定するものである．

Active Names は，サービスを提供するソケットの作成部分を "hijack" することでアプリケーションの名前解決に割り込む．一般のシステムでは，名前解決によって IP アドレスを得た後，その IP アドレスを用いてコネクションを作成する．Active Names は，これらを不可分のものとし，Active Names の名前解決の結果としてコネクションを返すように設計されている．

Active Names の名前解決により指定されたプログラムの列が，リクエストの流れるストリームに対して処理を行うことで，クライアントに対して資源のカスタマイズを可能としている．

2.2.3 Translucent file service

Translucent file service(TFS)[10] は，Sun Operating System (SunOS) 上に実装されたファイルシステムの一つである．TFS は，Unix ファイルシステムにおけるマウントの概念を拡張し，二つ以上のファイルシステムを重ね合わせる形でマウントすることを可能とするものである．

重ね合わせには優先順位が付けられており，手前のファイルシステムに存在するファイルが優先される．ファイルシステムへの書き込みを行う際には，一番手前に重ね合わされたファイルシステムへ書き込みが行われる．すなわち，もっとも手前のファイルシステムにおいて更新処理がなされる．また，優先順位の低いファイルシステムにあるファイルを仮想的に消去するためにwhite-outという機能をもつ．これは，優先順位の高いファイルシステムにwhite-outファイルを置くことで，優先順位の低いファイルシステムあるファイルを隠し，かつそれをユーザからは削除されたように見せかけるものである．

これらの重ね合わせの順序づけは，ファイルシステムをマウントする際に固定されるため，カスタマイザビリティは本論文で提案する多重名前空間にくらべ低いものとなっている．

2.2.4 3-D file system

3-D file system[16]は，Unixファイルシステムの名前空間を拡張するものである．3-D file systemでは，version fileという新たなファイル形式をを導入している．Version fileは，ディレクトリと同様に複数のファイルを格納可能であるが，ユーザがversion fileを参照すると，その中にあるファイルの一つが選択されて返される．

Version file内のファイルはそれぞれにバージョン付けがなされている．ユーザは，ファイル名のプレフィックスとバージョンを対にして指定することで，指定したプレフィックスをもつファイルに対して適切なバージョンを選択することが可能となる．

また3-D file systemには，TFSと同様にファイルシステムの一部分を重ね合わせる機能を持っている．

3-D file systemでは，本論文で提案する多重名前空間と同様のカスタマイズが可能である．提案方式では，重ね合わせを名前空間単位に限定することで名前空間の管理を単純化しつつ，資源名の指定のたびに名前空間の重ね合わせを変更することを可能とすることで使用時の柔軟性を達成している．

2.3 アクティブネットワーク

現在，世界で活発に行われているアクティブネットワーク研究は，2つに大別することが可能である．まず，第一にパケットにプログラムを組み込み，それがアクティブノードを通過する際に実行されるというものである．もう一方は，アクティブノード上のパケット処理プログラムを動的に配置可能とし，ルータなどの機能を拡張する

ものである．

前者の研究としては，Programming Language for Active Network(PLAN) [11] をパケット言語としたPLANet[13, 12] やActive Node Transfer System(ANTS)[38, 37]，後者の研究としてはNetScript[39, 40] を代表例としてあげることができる．これら2つの研究は，セキュリティや実行効率の観点，またその上で利用するアプリケーションによって，どちらのアプローチが適しているかが異なってくる．前者の研究においては，パケットごとにルータの振る舞いを変化させることが可能であるため，非常に柔軟性が高い反面，実行時のコスト削減やセキュリティに対する防御を厳密に行う必要が有る．同様に後者では柔軟性という点では劣っているが，アクティブノード内である程度固定されたプログラムが動くため，最適化やセキュリティに対しては有利である．

本論文の第4章で述べる名前解決の効率化方式は，名前解決ごとにプログラムを入れ替えるような柔軟性は必要としていない．本論文におけるプロトタイプシステムの実装では，あらかじめパケット処理のプログラムを各ノードに用意する方式をとっている．しかし，更なる柔軟性のためにプログラムを事前または実行直前に動的に途中経路上のアクティブノードに配置することも考えられる．そのような場合は後者のようなアクティブネットワークの実行時システム上に提案方式の名前解決機構を実装することになる．

2.3.1 PLANet

PLANetは，"purely active" なアクティブネットワーク・システムである．"Purely active" とは，パケットとルータの双方がactiveであること，すなわち，パケットとルータ双方がプログラマブルであることをいう．

パケットの機能拡張は，全てのパケットにプログラムを格納することで行う．パケットに格納されるプログラムは，PLANと呼ばれる特別なプログラミング言語で記述する．パケットを受け取ったサーバは，そのヘッダに含まれるPLANで記述されたプログラムにしたがって各種の処理を行う．

ルータは，拡張したOCamlを使って記述されており，機能の拡張は，ルータにおいてOCamlで書かれたプログラムを動的にロードすることで行われる．パケットのルーティングもこの処理に含まれるため，Internet Protocol(IP)[25] などの特定のインフラストラクチャを仮定することはない．

このように，プログラマブルな要素をパケットとルータに分離することにより，本節の冒頭で述べたトレードオフの妥協点を自由に変更することが可能である．つまり，

パケットのプログラマビリティを下げることでパケット処理の効率を向上することや，より高度なプログラミングを行うためにルータの機能を拡張することが可能である．

2.3.2 ANTS

ANTS では，従来のネットワークにおけるパケットをカプセルという概念に置き換えている．これは，オブジェクト指向におけるデータのカプセル化と同様のものと考えることが可能であり，パケットに対してそのパケットの処理方法を記述可能とするものである．

カプセル内にはコード自体は格納されず，コードへのリファレンスのみが格納される．カプセルによって指定されたコードは，最初にカプセルがノードに到着した際に，そのカプセルの送信元から動的にロードされる．これにより，同じコードを複数のカプセルにコピーするコストが削減される．

カプセルのプログラミングは Java を用いて行い，ANTS ではカプセル用の型，ノードの型，およびカプセルが使用する API などを提供している．

2.3.3 NetScript

NetScript は，ルータやスイッチなどの機器をプログラマブルにすることを目的とした研究である．

NetScript は，パケットのプログラマビリティよりむしろネットワークのプログラマビリティに重点を置いている．ネットワークをプログラムする際に使用する言語は NetScript 言語と呼ばれ，データフロー型の言語となっている．ネットワーク・プログラミングに特化した言語でプログラミングする点が NetScript の特徴である．ただし，低レベルな記述などは，従来のプログラミング言語 (C, C++, Java 等) を使用することが可能であり，NetScript はそれらのコンポーネントの glue 言語であるということもできる．

2.3.4 Active Messages

Active Messages[33] は，分散並列計算における通信コスト削減のために考案されたメッセージ通信の機構である．

Active Messages では，パケットのヘッダ部分に，そのメッセージを処理するハンドラのアドレスを記述しておく．メッセージを受け取ったネットワークデバイスは，パ

ケットをアドレスに指定された所定のハンドラに渡す．この時，一切のバッファリングを行わないことにより通信処理の高速化が可能となっている．

受信したメッセージによって計算機上の特定のプログラムが作動するという，メッセージ駆動型のアーキテクチャは，アクティブネットワークのものと相似している．しかし，Active Messages は分散並列計算に特化したものであり，セキュリティの問題や異機種性などは考慮の範囲外となっている．

2.4 広域ネットワーク上のキャッシング

2.4.1 Harvest

Harvest[6] は最も有名なインターネット上のオブジェクト・キャッシュシステムの 1 つである．Harvest は Internet Cache Protocol[36, 35] を用い，また，Time-To-Live ベースの一貫性管理を行っている．キャッシュの階層構造は，キャッシュを行うサーバ間の接続によって静的に構成される．本論文の第 4 章で述べる方式では，キャッシュの階層構造はサーバ・クライアントの通信経路上に存在するルータなどの機器上に実装される．このため，クライアントからのリクエストは，自然な形でそのルート上でキャッシングされる．

2.4.2 広域環境でのキャッシュ一貫性管理

キャッシュの一貫性管理では，大きくわけて 2 つの一貫性管理モデルがある．強い (strong) 一貫性管理モデルでは，元データに変更がなされた場合，その変更は直ちに全てのキャッシュに反映されるというものである．このとき，「直ちに」の定義によって様々なモデルが存在する．これに対し，弱い (weak) 一貫性管理モデルは，元データとキャッシュとの一貫性を緩めるものである．一般に，強い一貫性管理モデルは，弱い一貫性管理モデルよりその管理コストが大きくなる．反面，弱い一貫性管理では，既に古くなったデータをキャッシュが保持したままとなるため，これが致命的な処理の場合には弱い一貫性管理モデルを採用することはできない．

文献 [17] では，Web キャッシュにおいて強い一貫性管理モデルを導入した場合でも，弱い一貫性管理モデルを採用した場合に比べ，少し，あるいは全く追加のコストがかからないと述べている．そして同文献では，アップデートでなく無効化プロトコルを使用したほうが良いと述べている．

2.4.3 キャッシュを内蔵したルータ

Clarity Web Cache[30] はサーバ及びクライアントに透明に Web キャッシングを導入するものである．これは，Web キャッシュをルータ上に透明に構成するものである．このシステムが提供している機能に加え，本論文の第4章で述べる方式ではルート上でリクエストのフォワーディングを行うことや，名前の更新情報をノード間で互いにやりとりすることで一貫性を維持している．

Intel 社の Transparent Cache Switching[14] や Cisco Systems 社の Cisco Cache Engine[7] は，専用ルータ上に構成された Web キャッシングのメカニズムである．これは，ルータを通過する Web トラフィックを監視し，外部に出ていく Web リクエストのストリームを，あらかじめ指定されたアドレスに転送するというものである．指定されたアドレスには Web キャッシングを行うサーバが置かれている．この転送はルータによって透明に行われるため，ユーザは，この機能を意識することなくキャッシュを使用することができる．

2.4.4 Web キャッシングと名前解決キャッシュの関連

前節までで述べた研究や製品は，キャッシュを Web に適用している．Web キャッシングと名前のキャッシングの大きな差異は，データのサイズと更新の方法である．

Web のキャッシングでは，データのサイズは多岐にわたる．これは Web が小さなテキストデータから巨大なマルチメディアデータまでを扱うからである．これに対し，名前解決システムでは，名前解決時にクライアントとサーバの間でやりとりされるデータは小さく，それゆえシステムの性能はネットワークのバンド幅より伝播遅延に影響されやすい．さらに，キャッシュ機構には少ないメモリ領域で大量の名前をキャッシュすることが可能である．さらに，名前のキャッシングにおいては，Web キャッシュとは異なるキャッシュ置換アルゴリズムを考える必要がある．

Hypertext Transfer Protocol (HTTP)[9] は，Web サーバへアクセスする際に使用されるプロトコルである．しかし，Web 上のコンテンツを更新する際には HTTP は使用されないため，Web コンテンツの更新は単純に HTTP トラフィックを監視するだけでは行えない．本提案方式を Web に適用するためには，Web サーバはそのコンテンツに更新があるたびに，隣接するアクティブノードへ更新情報を伝えるようにする必要がある．

これに対し，同様にインターネット上の情報取得に使用される File Transfer Protocol (FTP)[27] は，ファイルのダウンロードだけでなくファイルのアップロードもサポート

している．すなわち，全てのファイルの交換をFTP上で行うのであれば，サーバ上のファイルのアップデートを途中のノードで検知することも可能であると考える．

2.5 まとめ

本章では，計算機で使用される名前とその利用法に関して概観した後，名前解決のカスタマイズに関する関連研究について述べた．また，クライアントおよびサーバに透明な途中経路上におけるキャッシングのための基礎技術としてのアクティブネットワークに関する研究について述べたのち，広域ネットワークにおける各種キャッシング・システムについて概観した．