〈基盤技術〉

Polimatica:ポリシー自動プロビジョニング・グリッドの実装 ーダイナミックコラボレーションの基盤技術としてー

Polimatica:An Implementation of Policy Automated Provisioning Grid — Foundation of Dynamic Collaboration —

古城　隆* 前野義晴** 妹尾義樹**
Takashi Kojo Yoshiharu Maeno Yoshiki Seo

要　旨

ダイナミックリソースアロケーションは，ダイナミックコラボレーションを実現するバックエンドシステムに必要不可欠な特性の1つです。本稿では，まずそのためのシステムの要件を取り上げ，ポリシーによる自律化がキーテクノロジーであることを示します。次に，複雑な大規模分散システムのダイナミックリソースブローカリングによる拡張性に優れたモデルを提案します。NECは，このモデルの基本機能を実装し，モデルの実現性の証明に成功しました。

Dynamic resource allocation is one of the critical characteristics required for the back-end systems implementing Dynamic Collaboration. In this paper, we first discuss the requirements on such systems and point out that the policy automation is the key technology. Then, scalable models for dynamic resource brokering of complex large distributed systems are discussed. We implemented basic functions of the models and successfully demonstrated feasibility of the model.

1．はじめに

ダイナミックコラボレーションは，様々な種類のアプリケーションを統合的にサポートします。モバイルアプリケーションやWebポータルのようなユーザアクセス指向の対話型アプリケーションもあれば，ストリームビデオアプリケーションのように大量のネットワークトラヒックが必要なアプリケーションもあります。また，データベース指向のアプリケーションや，集中的なコンピュータ処理が必要なアプリケーションもあります。ダイナミックコラボレーションシステムを実現するには，バックエンドシステムを中核としてフロントエンドやネットワークインフラまでを含めて様々な特性が必要とされます。

このようなアプリケーションは，ユーザアクセスの急増を予測できない，システム負荷がまったく確率的に変動するアプリケーションもあれば，1日，1週間または1ヵ月などの周期的なシステム負荷パターンを持つアプリケーションもあります。それぞれのアプリケーションには，性能やセキュリティなどの要件をもとにした独自のハードウェアとソフトウェアの構成が必要です。システム負荷は常に変化するので，各アプリケーションのシステム構成に最大負荷量を想定するのは現実的ではありません。システムリソース全体の活用を考えると，システム負荷の変化に即したシステム構成とリソース割り当ての動的な変更が強く求められますが，これには通常，人による操作や，時間のかかる難しい作業が多く含まれます。従来の人による操作を媒介としたシステム管理速度では，常に変化する環境に合わせることはできません。

プロビジョニング・グリッドは，アプリケーションに必要なすべてのソフトウェアの配備も含めたシステム構成変更に関する操作をテンプレートを用いてワークフロー化する技術です。ビジネスプロセスのリエンジニアリングにあわせてシステム構成変更を即座に実行できる利点があります。この技術は，多くの面で管理間接費を削減し，運用コスト全体を削減します。また，システム構成の変更にかかるターンアラウンドタイム（プロビジョニング時間）を抜本的に短縮します。多くのベンダーが実用化をめざしているグリッドテクノロジーは複数組織間の共同作業環境におけるITリソースの共有に重点を置くのに対し，本稿で取り上げるプロビジョニング・グリッドは，ポリシー・ベースの動的リソース割り当て，リソースの共有・活用の自律化に重点をおいた新しいグリッドテクノロジーです。プロビジョニング・グリッドにより，システム環境の変化に即して，ハードウェアとソフトウェアの構成変更を完全に自律化でき

＊ システム基盤ソフトウェア開発本部
System Platform Software Development Division

＊＊ インターネットシステム研究所
Internet Systems Research Laboratories

ます。また，常に変化するシステム実行環境向けの各アプリケーションに最適な環境を提供するので，アプリケーションはユーザが独自に要求するサービスレベルに対応でき，システム全体も最大リソース使用量，つまり非常に高い稼働率を実現できます。

本稿では，プロビジョニング・グリッドに必要な特性，システムのベースとなる内部モデル，NECによる実装状況について説明します。

2. 要　件

本章では，プロビジョニング・グリッドの様々な要件を取り上げます。

(1) 管理可能なリソース

システム構成とリソース割り当てを確実に最大限最適化するためには，グリッドの管理可能なリソースに，ハードウェアからアプリケーションソフトウェアまでの様々な層のシステムリソースを含める必要があります。最下層では，様々な種類のサーバハードウェア，記憶装置，ファイアウォール，ロードバランサ，およびネットワークスイッチの割り当てを管理する必要があります。また，割り当てられたサーバには，適切な種類，バージョンのOSをインストールしなければなりません。Webサーバ，アプリケーションサーバまたはDBMS（データベース管理システム）などのミドルウェアも配備が必要なアプリケーションです。アプリケーションを実行するには，アプリケーションプログラムや初期データ，コンテンツまたはデータベースも，振られたシステム構成内にセットアップしなければなりません。

これらの構成物はすべて，管理可能なリソースとみなされます。

(2) リソースの仮想化

リソースには管理用のインタフェースが必要です。プロビジョニング・グリッドとの関係において，インタフェースは，プロビジョニング・グリッドのミドルウェアが，他のグリッドサービス（**図1**）を扱う場合と同じようにリソースを管理できるように，管理用のサービスポートを公開するWebサービスによって仮想化されます。

仮想化によって異種の構成要素を容易に取り込むことが可能になります。組織間に分散した大規模なシステムでは，同質のハードウェア，OSまたはミドルウェアを想定することはできません。サービスの標準化については，複数の団体で議論されています。一番最近の試みとしては，OASIS OpenでのWebサービス分散管理（Web Service Distributed Management）があります。

(3) 自律性とポリシー

アプリケーションには，リソース割り当てと動作の面で自律性とある程度の独立性が必要です。これらの特性をビジネスプロセスとポリシーによって事前定義し，アプリケーションがシステム環境の変化にかかわらず必要な作業を適切に実行できるようにする必要があります。

ビジネスプロセスとは，管理のための複数のフェーズを実行する一連のアクティビティのことです。プロセスには，無限周期的な活動と，イベントによって起動される有限の活動があります。

ポリシーは，イベントとアクションのペアです。イベントは，ユーザアクセス，システム負荷の変化，あるいはシステム内の様々な挙動です。アクションは，イベントを適切に処理するビジネスプロセスとして定義することができます。

(4) 拡張性

ダイナミックコラボレーションは，広範囲のグループ，組織または企業にまたがっています。コラボレーションをサポートするバックエンドシステムは，アプリケーションのサービスレベル要件を満たすために，簡単に拡張できなければなりません。拡張性の指標には，物理的・地理的な分散，組織の論理的な数，ユーザ，組織，様々なアプリケーションの規模の拡張性などが含まれます。

システムが多数のアプリケーションに拡張されると，システムの動作や性質も急激に複雑になります。通常，システム設計者がすべての可能なシステム動作を予測し，あらゆる場面でシステム全体を適切に管理するプロセスおよびポリシーを定義するというのは，非現実的です。グリッドは，問題を妥当な単位に分解するための適切な手段を提供しますが，システム設計者がシステムのローカルの動作と性質だけを処理し，妥当な複雑性の問題に切り分けられるように，プロセスとポリシーをローカルレベルまで分解する必要があります。

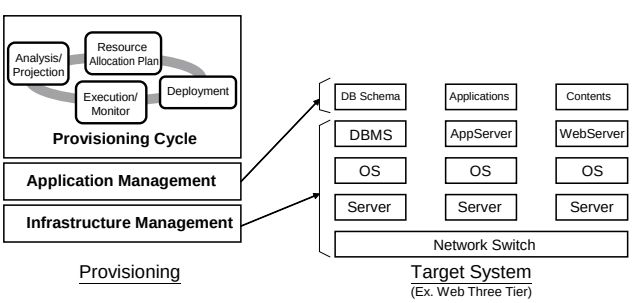

図1　プロビジョニング層
Fig.1　Provisioning layers.

3. モデル

本章では，プロビジョニング・グリッドのベースとなる様々なモデルを取り上げます。

(1) 仮想組織

VO（Virtual Organization）は，実際の組織が共有できる，アプリケーションのための独立した仮想作業領域です（**図2**）。このモデルは，OGSA（Open Grid Services Architecture）で検討されています。VOは，セキュリティ領域でもあり，リソース割り当て領域でもあります。セキュリティ領域としては，外部から組織内部のリソースへのアクセスは，完全に禁

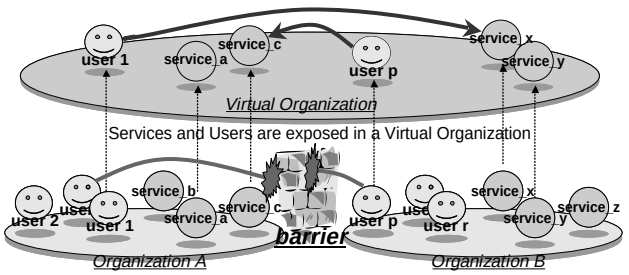

図2　実存組織と仮想組織
Fig.2　Real and virtual organizations.

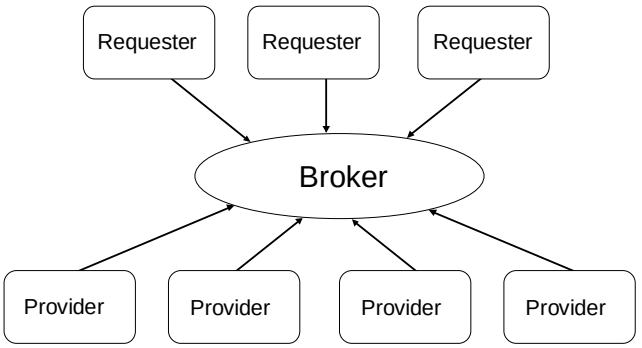

図3　シングル・ブローカー・モデル
Fig.3　Single broker model.

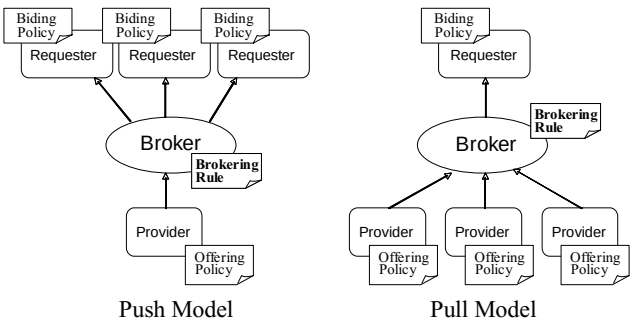

図4　プッシュ/プルモデル
Fig.4　Push and pull models.

止されるか，またはVOのポリシーによって管理されます。

リソース割り当て領域としてのVOは，実存組織のリソースプールから割り当てられたリソースを所有します。リソースは，必要に応じて，適切なメカニズムとポリシーを通して，VOのアプリケーションに割り当てられます。

(2) ネットワーク・モデル

ネットワークデバイスの仮想化は，この種のシステムでのリソース・プロビジョニングをなるべく複雑にしないために重要です。VOは，ネットワーク仮想化に拡張できます。プロビジョニング・グリッドは，従来のネットワークデバイスの集合の上にネットワークデバイスの設定の詳細すべてを隠すVOモデルを実装します。プロビジョニング・グリッドのVOモデルには，既存のネットワークデバイスの集合に対する一連の操作設定の詳細を隠蔽したネットワークが含まれます。

VOは，独立したネットワークパーティションを持っており，パーティションは，指定された帯域幅で別のVOまたは公共のインターネットスペースといった外部に接続されています。パーティションは，ネットワークセキュリティ要件に応じて，複数の方法で実装可能です。プロビジョニング・グリッドVOは，パーティションの外側からの仮想上，より厳格な独立を実現するパーティションで区切られたレイヤ2上に実装されます。

単純なモデルでは，パーティション内部のリソースは，互いに平等にアクセスできます。しかし現実的なセキュリティ要件を考えると，リソースのクラスタが固有の接続性を持つ，より高度なモデルを実装する必要があります。たとえば，3層のWeb構成などが考えられます。

NECは，構成の動的な作成および管理のために，テンプレートモデルを導入しました。3層のWeb構成は，システムが多くの適切な種類のサーバリソースを必要に応じて割り当てることができるテンプレートです。

(3) リソースブローカリングモデル

リソースブローカは，必要に応じてROのリソースをVOに動的に割り当てるメカニズムです。システムがすべてのROからすべてのVOへのリソースブローカリングをすべて処理するシングルセントラルブローカリングメカニズムの場合，システム拡張に対応できるブローカリングポリシーは非常に複雑になり，十分な設計は，完全に不可能ではないとしても極めて困難になります（**図3**）。

設計範囲をローカルレベルの各ROおよびVOに分解し，問題を簡単にするために，NECはリソースブローカリングのプッシュ/プルモデル（**図4**）を導入しました。プッシュブローカリングモデルは，リソースプロバイダ中心のモデルです。プロバイダによって提供されたリソースは，リソースのコンシューマ間で分配されます。グリッドシステムでは，プロバイダはROで，コンシューマはVOです。ブローカにはブローカリングポリシーがあり，リソースプロバイダにはリソース提供ポリシーがあり，コンシューマには要求ポリシーがあります。

プルモデルは，プッシュモデルの逆のモデルです。コンシューマによって要求されたリソースは，複数のプロバイダからの利用可能なリソースから選択されます。プッシュモデルの場合同様，各構成要素はそれぞれポリシーを持っています。

複数のリクエスタとプロバイダ向けのブローカリングモデルは，プッシュ/プルモデルブローカを組み合わせて構築します（**図5**）。このモデルの長所は，ポリシーがVOまたはRO内に限定され，システム設計者が特定のVOまたはROのポリシー設計に集中できることです。また，完全なピアーツーピア・アーキテクチャのため集中的なメカニズム

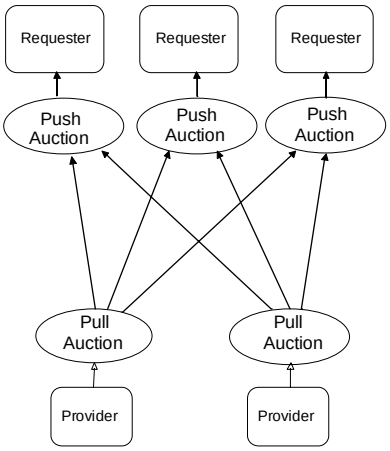

図5　プッシュ/プルモデルの組合せ
Fig.5　Combination of push and pull models.

が必要なく，多くのVOを作成する大規模システムの拡張性面でのボトルネックも回避できます。

(4) ポリシーモデル

ポリシーモデルも，グリッド実装の鍵となるモデルです。一般的なモデルや特定のアプリケーションのモデルなど様々なモデルが検討されています。イベントとアクションのポリシーモデルは，単純ですが強力な一般ポリシーモデルです。このモデルは，イベントと対応する実行アクションをセットで定義します。

4. 実　装

4.1　機能の構成

第3章のモデルをもとにプロビジョニング・グリッド・ミドルウェアPolimaticaとプロトタイプシステムが開発されました。**図6**は，システムの機能構成を示しています。システムは3つの層によって構成されています。VO管理層は，VOを実装し，管理します。リソースプロバイダ層は，ROが所有するリソースを管理します。エージェント層は，物理デバイスとグリッド・ミドルウェアとをインタフェース接続します。

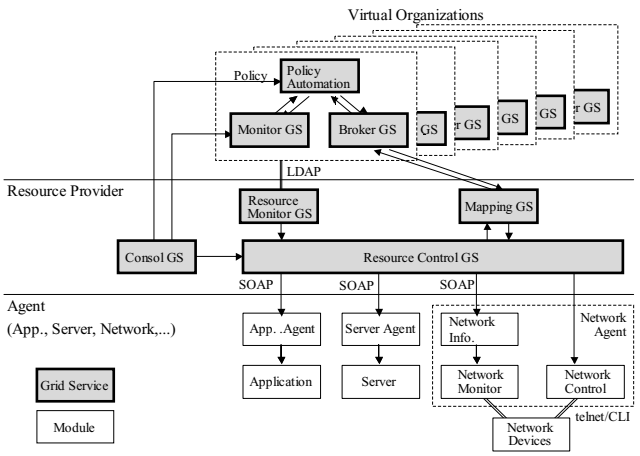

図6　システムの機能構成
Fig.6　Functional configuration of the system.

VO管理層は，3つのグリッドサービスで構成されます。ポリシー自動化GSは，VOの自律リソース管理を実装する最上位のプロセスとポリシーです。監視GSは，システムコンポーネントの負荷を監視し，コンポーネントの負荷レベルが変わると，ポリシー自動化GSにイベントを通知します。ブローカGSは，システムリソースのリクエスタ側ブローカです。現在のシステムは，1つのROを管理し，プルモデルブローカリングのみを実装します。

リソースプロバイダ層は，リソース監視GS，リソース制御GS，およびマッピングGSで構成されています。マッピングGSは，ネットワークデバイスの操作をVOのネットワークパーティションに仮想化します。ブローカが，ネットワークパーティション（VO）でリソースの追加または削除を要求すると，マッピングGSはリクエストを解釈し，ネットワークデバイス向けの一連のコマンドに変換します。

4.2　ポリシーエンジン

ポリシーエンジンは，ポリシー自律化GSの中心機能です。このエンジンは，イベント条件とアクションの定義のペアであるイベント/アクションタイプのポリシーを実装します。ポリシーは，1つまたは複数のイベント条件が満たされ，対応するアクションが要求されると開始されます。

イベントには，優先順位が割り当てられています。エンジンがポリシーの優先順位をコントロールしているので，イベントが前のアクションのエラーによって発生した場合，復旧ポリシーがアクションを無効にし，開始します。

4.3　リソース監視

リソース監視GSは，リソースの状態を監視します。リソース監視GSは，それぞれの管理する構成物に対応するリソース制御GSから状態情報を定期的に収集します。構成物は，物理デバイスまたはソフトウェアです。VOリソースモニタおよびROリソースモニタは，MDS（Monitoring and Discovery Service）に実装されます。MDSは，グリッドリソース向けディレクトリサービスを提供するグローバスツールキット（Globus Tool Kit）が提供するサービスです。VOモニタのMDSは，VOのリソース情報をキャッシュし，ROモニタのMDSは，ROのリソース情報を保存します。VOベースのセキュリティを実装するために，VOモニタは，VOのリソースまたはプールの空きリソースにしかアクセスできません。

4.4　リソースプール

リソースプールもリソース監視GSによって管理されます。GSは，VOの照会に対して使用可能なRO内のリソースをすべて表示します。現在の実装では，リソース照会は順番に並べられ，リソースはVOに先着順に割り当てられます。

4.5　技術基盤

プロビジョニング・グリッド・ミドルウェアPolimatica

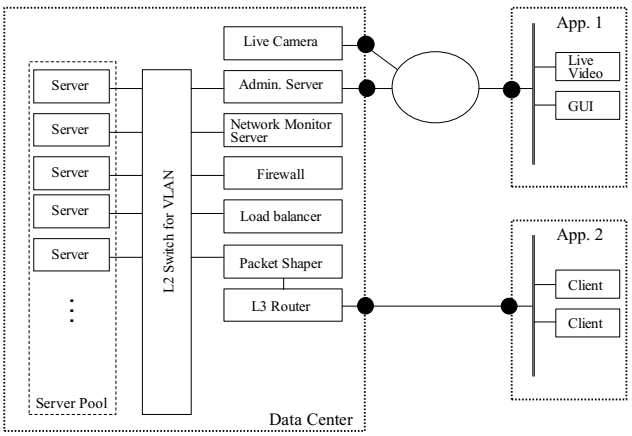

図7　デモシステムの物理構成
Fig.7　Physical configuration of the demo system

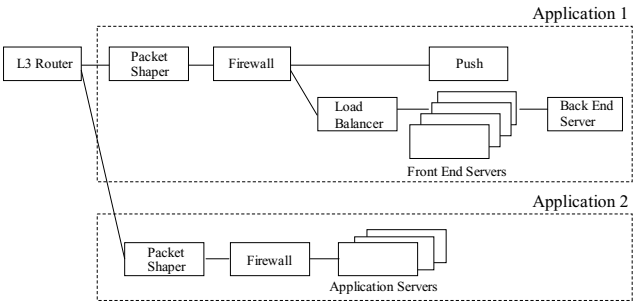

図8　デモシステムの論理ネットワーク構成
Fig.8　Logical network configuration of the demo system.

は，Globus Tool Kit 3で開発され，グリッドサービスは，OGSIサービスとして実行されます。MDSに関しては当時利用可能だったGT2ベースのMDSを使用しました。

4.6　コンセプト実証用デモアプリケーション

プロビジョニング・グリッドのコンセプトを実証するために，2つのアプリケーションをグリッドミドルウェア上に実装しました。最初のアプリケーションは，ビデオストリーム提供アプリケーションで，クライアントからリクエストが来ると，多数のサーバおよびサーバとクライアント間の帯域幅を使います。次は，ビジネスプロセスを管理するアプリケーションで，このアプリケーションは，ネットワーク帯域幅を必要としません。

デモシステムの物理的および論理的なネットワーク構成を**図7**と**図8**に示します。システムは，パケットシェーパ，ファイアウォール，ロードバランサなどのネットワークデバイスを持つ2つのVOと，サーバを管理します。システムは，アプリケーション向けテンプレート構成を管理しながら，そのアプリケーションに合った数のサーバと帯域幅を割り当てます。

5．むすび

本試作実験によりプロビジョニング・グリッドの設計構想が動的に変化する環境で効果的に機能することは，十分証明できました。しかし，現在のプロトタイプの実装は限定的で，多くの機能はまだ改善する必要があります。現在，システムは，製品レベルの実現に向けて強化が続けられています。現在解決しようとしている将来的な問題の一部を以下に示します。

(1) リソースブローカリングポリシー

現在のプロトタイプでは，簡単なリソースブローカリングを処理するだけの最小限のポリシーしか実装していません。リソースがプールで使い果たされ，複数のVOからのリソースリクエストを調整する際への考慮がさらに必要です。複数のVOからのリクエストの優先順位を判断するために，より高度なポリシーが必要です。

(2) 複数のRO

現在の実装は，1つのROしかサポートしていません。将来的な大規模グリッドの配備では，複数のROのサポートが必要不可欠です。これには，本稿で述べたブローカリングモデルの完全な実装が必要です。

(3) ダイナミックVOのライフサイクル

現在の実装は，スタティックVOを想定しています。ダイナミックコラボレーションのより現実的な環境に対応するためには，VOを必要に応じて動的に作成および削除する必要があります。

(4) リソース予約

現在の実装は，すぐ使用するリソースアロケーションを想定しており，リソース予約はサポートしていません。

将来利用されるリソース予約をサポートするために予約管理テーブルの実装とテーブルに対する一貫性のある操作を保証するリソースブローカリングの拡張性が必要になります。

(5) 配備

現在の実装は，使用前にすべての必要なソフトウェアが適切にインストールされていることを想定しています。大規模なリソースアロケーションでは，リソースの予約や割り当てに即したオンザフライ配備のサポートが必要となるでしょう。配備では，アプリケーション，ミドルウェア，およびOSの層をサポートする必要があります。アプリケーションの配備には，アプリケーションソフトウェア同様，すべてのコンテンツデータ，データベーススキーマなどのレジストリ情報も含める必要があります。OS，DBMS，アプリケーションサーバ，Webサーバなどの基盤ソフトウェアの配備も必要です。

参考文献

1) I. Foster, D. Gannon and H. Kishimoto, “The Open Grid Services Architecture,” Global Grid Forum OGSA working group, http://forge.gridforum.org/projects/ogsa-wg.

2) S. Tuecke, K. Czajkowski, et al., “Open Grid Services Infrastructure (OGSI) Version 1.0,” Global Grid Forum OGSI working group, http://forge.gridforum.org/projects/ogsi-wg.

3) R. Wolski, J. S. Plank, et al., “Analyzing Market-based Resource Allocation Strategies for the Computational Grid,” The International Journal of High Performance Computing Applications, vol.15, no.3, pp.258-281, 2001.
4) R. Buyya, H. Stockinger, et al., “Economic Models for Management of Resources in Peer-to-Peer and Grid Computing,” Proceedings of the SPIE International Conference on Commercial Applications for High-Performance Computing, Denver, 2001.
5) G. Wasson and M. Humphrey, “Towards Explicit Policy Management in Virtual Organizations, “Proceedings of the IEEE 4th International Workshop on Policies for Distributed Systems and Networks, Lake Como, 2003.
6) “Utility data center: architecture,” http://h30046.www3.hp.com/solutions/architecture.html.

筆者紹介

Takashi Kojo
古城　隆（こじょう　たかし）　1977年，NEC入社。現在，ソリューション開発研究本部システム基盤ソフトウェア開発本部統括マネージャー。

Yoshiharu Maeno
前野　義晴（まえの　よしはる）　1993年，NEC入社。現在，インターネットシステム研究所主任。

Yoshiki Seo
妹尾　義樹（せお　よしき）　1986年，NEC入社。現在，インターネットシステム研究所研究部長。