分散コンポーネントによる即興的アプリケーション構成機構の実現

岩井将行[†] 中澤仁[†]
西尾信彦[†,†††] 徳田英幸[†,††]

今日，コンポーネント技術によって多様な分散システムが構築されている反面，それらのシステムは，高度な知識を有するシステム開発者やシステム設定者を必要としてきた．著者らは，ホームネットワークなどのシステム開発者の介在が困難な場面で，コンポーネント技術の利便性を生かし，一般的なユーザが即興的に分散システムの構築を可能とする分散ミドルウェア Dragon を開発した．Dragon は，各コンポーネントを単純化できる複数仲介者型間接通信モデルを採用し，経路判断機能の外部化，コンテンツ適応機能の構造化，信頼性のある 1 対多イベント配送機能などの特徴を持つ．本論文では，Dragon が 1 対多イベント配送において，優先度を区別したマルチプルユニキャストが正しく動作しているか，仲介者の直列的な配送時間，近接コンポーネントの発見などを評価し，適切な配送処理動作と妥当な動作時間で稼動可能であることを示す．Dragon を用いることで家庭内ネットワークだけでなく，工場などの時間制約をともなったイベント配送を要求される場でも即興的なシステム構築が可能となる．

A System for Improvised Composition of Component-based Applications

Masayuki Iwai,[†] Jin Nakazawa,[†] Nobuhiko Nishio[†,†††]
and Hideyuki Tokuda[†,††]

Distributed systems, these days, are developed by putting various software components together through networks. However, such systems require developers who are familiar with the software architecture and its components. In this paper, we present a design and implementation of a new middleware, Dragon, which is suitable for collaboration between distributed components within networks where there are no system developers, such as a home environment. Dragon allows us to construct and modify an event-driven distributed component system easily and extemporarily. Dragon's implementation is based on the Multiple Mediators Indirect Communication Model, which makes each component simple. One of the features is one-to-many communication with reliability using multiple-unicasting. In addition, Dragon incorporates soft real-time event delivery to enhance the predictability of the behavior within a constructed system, and the composition of distributed services for event management.

1. はじめに

近年，多くの分散システム開発の場面でコンポーネント技術が注目されている．コンポーネント技術は，ソフトウェアを部品化することで，すでに他の分散システム用に作成したコンポーネントを再度他の分散システムに流用できるため，開発コストを軽減できる利点がある．また，ソフトウェアコンポーネント技術は複雑な分散システムにおいて，障害が発生した個所をすばやく特定して，システム全体から切り離すことを可能にする．このような再利用性とモジュール性を備えたソフトウェアコンポーネント技術は，情報のミラーリング，分散スケジューラ，組織間意思決定プロセス支援，監視アプリケーション，システムインテグレーションなどの様々なアプリケーションドメインで急速に利用されつつある．

しかしこれらのコンポーネント技術は，現在のところシステム開発者を介して構成，運用されている場でしか利用できない．これは既存の分散システムが，システム開発者が利用者の用途を想定してコンポーネントの詳細をデザインし，組み合わせるコンポーネントを決定し，それらのコンポーネント間を接続する詳細なプロトコルを決定しているからである．そのため，

† 慶應義塾大学大学院政策・メディア研究科
Graduate School of Media and Governance, Keio University
†† 慶應義塾大学環境情報学部
Faculty of Environmental Information, Keio University
††† 科学技術振興事業団，さきがけ研究 21，協調と制御
PREST, Japan Science and Technology Corporation (JST)

利用者によるシステムの再構成は困難である．分散システムの仕様変更ごとにシステム開発者に変更を要請するため，経済的にもコストが生じる．

一方，家庭内ネットワークや小規模ネットワークをはじめとしてシステム開発者の介在が困難な場面が多く存在する．そのようなネットワーク環境では，コンポーネントやソフトウェア，および情報家電機器が独立して動作している．情報家電機器やソフトウェアが，分散システムの構成要素として，頻繁に追加あるいは変更される場合，それらの組合せをシステム開発者ではなく利用者自身が簡単に行える必要がある．しかし，高度なシステム開発の知識のない一般的な利用者でもコンポーネント間を接続して簡単に分散システムを構築できる機構は，提供されていない．

一般的な利用者が，使用目的の異なるコンポーネントを即興的に接続し，1 つの分散システムを構築するには，ミドルウェア側で近接するコンポーネントの発見，通信内容の相互適応，1 対多通信における配送のオーバフローの制御などを考慮しなければならない．また情報家電機器内の組込み機器などで動作させることを考慮すると，各コンポーネントが単純な機能で実装されており，負荷を与えないことが重要となる．組込み機器などは PC と異なり，資源が制限され，複雑なソフトウェアの挙動がコスト面で大きな問題になるからである．

本研究では，コンポーネントによる分散システムの即興的な構築を可能にする分散ミドルウェア Dragon を開発した．Dragon により，複雑な前提知識がない利用者にも，分散システムの即興的な構築が可能になる．これによってコンポーネント技術の利用分野はさらに広がり，家庭内ネットワークなどの分野での応用が期待できる．

本論文では，まず 2 章で即興的分散システム構築の要件を述べ，従来研究の問題点と Dragon イベント配送モデルにおける解決手法について述べる．3 章で同モデルを基に実時間のイベント配送機構を備えた分散ミドルウェアである **Dragon** の実装を示し，即興的分散システム構築に適したソフトリアルタイムイベント配送メカニズムの詳細を述べる．4 章では，即興的な分散システム構築を視覚的に支援する GUI ツールを含めた Dragon の応用例を述べ，他研究では行えない特徴を説明する．5 章では，Dragon の基本性能を測定し，家庭内ネットワークでの利用および即興的分散システム構築の要件を満たしているかを評価する．6 章で個々のコンポーネントの単純性とコンポーネント間のトポロジに着目し，分散コンポーネント通信の分類を行い関連研究との比較を述べる．最後に 7 章で本論文をまとめる．

2. 即興的分散システム構築モデル

本章では即興的分散システム構築の要件を述べ，従来研究の手法の問題点を指摘する．そのうえで，即興的分散システム構築に適した Dragon イベント配送モデルを提案し，その特徴を述べる．

2.1 用 語 定 義

本節では，即興的分散コンポーネント通信を議論するうえで重要となる用語を定義する．まず，コンポーネント間でやりとりされるメッセージ，データ，オブジェクト，および命令を総じて**イベント**として定義する．また，イベントがあるコンポーネントから別のコンポーネントへ伝わるとこを，**イベント配送**あるいは**イベント通知**と呼ぶ．

イベントを発生させる送り手コンポーネントを**イベント生産者**と呼ぶ．また，他のコンポーネントから発生したイベントを受け取り，その内容に基づいた処理を行うコンポーネントを，**イベント消費者**と呼ぶ．イベントを受け取り，その内容に基づき，新たなイベントを発生させるコンポーネントを**イベント仲介者**と呼ぶ．さらに，これらのコンポーネントをどのように接続させるのかを認識しているコンポーネントを**経路判断者**と呼ぶ．中でも第 3 者として経路判断を下しているコンポーネントを，**コマンダ**とする．

2.2 即興的分散システム構築の要件

システム管理者や開発者が介さない家庭内ネットワークなどの小規模ネットワークにおいて，コンポーネント間の即興的な接続を実現するためにミドルウェアが必要となる機能を示す．

近接コンポーネント認識機能 システム開発者がすべてのコンポーネントを認識している場合と異なり，利用者が即興的に分散システム構築を行う場合は，ミドルウェア側で近接のコンポーネントを発見する機構を提供していなければならない．特に，物理的に利用者に近接する場所や，マルチキャスト通信可能なネットワーク上にあるコンポーネントを発見することで，利用者に対する現状の提示や自動的なコンポーネント間接続が可能になる．近接するコンポーネントを認識する手法として，ディレクトリサービスをシステム内に設置する手法と，コンポーネントが相互に自分自身の存在を通知する手法がある．

コンポーネント障害検知機能 分散システムにおいては，利用者からの外的な要因で障害が起こる可能性があり，通信相手先コンポーネントに対する信頼性は

つねにない．そのためにコンポーネントの障害をすばやく検知する必要がある．

システム障害可能性検知機能 即興的に構築される分散システムにおいては，システム開発者によるテストやチューニングがない．そのため設定によっては，オーバフローなどによって分散アプリケーション全体が障害に陥る可能性があり，その障害をあらかじめ検知する機構をミドルウェアが備えている必要がある．

経路判断機能 初期段階でコンポーネント間の接続が決定されていない分散システムにおいて，近接のコンポーネントの中から自らがイベントを送信あるいは受信すべきコンポーネントを選び出す機能が必要となる．経路判断機能はイベント生産者がイベント消費者を選んでいる場合と，イベント消費者がイベント生産者を選んでいる場合がある．各コンポーネントで経路判断機能を持たず，この機構を外部化する 2.1 節で定義したコマンダによって操作させることも可能である．

コンテンツ適応機能 即興的にコンポーネントを接続する場合，生産者が発生するイベントと消費者の認識可能なイベントの内容が異なる可能性がある．その差異を吸収する機能をコンテンツ適応機能と呼ぶ．適応性があるコンポーネントとは，即興的に発見された通知先コンポーネントに合わせ，イベントのコンテンツを変換させるコンポーネントを指す．たとえば *button.pushed()* という状態を示したイベントから *light.on()* という命令の示すイベントへ変換するなどの処理をしなければならない．

1 対多イベント配送機能 複数のコンポーネントにより構築される分散システムにおいて，重要なイベントを生成する生産者が 1 つしかない場合などは，1 つの生産者から多数の消費者へイベントが配送されるアプリケーションを構築しなければならない．このような 1 つのコンポーネントが複数のコンポーネントへイベントを一度に配送しなければならない状態（**1 対多イベント配送**）は分散システムでは多く存在し，ミドルウェアではシステム全体がオーバフローすることを防ぎながら，1 対多イベント配送の信頼性を保証する必要がある．

2.3 従来研究の問題点

コンポーネント間の接続を行うミドルウェアは，CORBA Event Service[8]，JEcho[12]，JavaSpaces[9] など多く開発されてきた．それらの研究では，各コンポーネントにコンテンツ適応機能，経路判断機能を組み込むために複雑な処理を追加した．しかし，家庭内ネットワークなどの，生産者コンポーネントや消費者コンポーネントに複雑な処理を期待できない場合では，より単純なモデルで各コンポーネントを実装できることが望ましい．また，コンポーネントどうしを即興的に接続できる Carp@[2] においてもユーザ構築したシステムに対する障害可能性検知や信頼性を考慮していない．したがって，システム開発者を介さない家庭内ネットワークなどの場面で利用できない問題がある．

2.4 Dragon イベント配送モデルの特徴

著者らは，2.2 節に示す機能要件を満たし，生産者コンポーネントや消費者コンポーネントに負荷をかけない Dragon イベント配送モデルを提案する．Dragon イベント配送モデルは，仲介者の構造化，経路判断機能の外部化，信頼性のある 1 対多イベント配送機構という特徴を持つ．以下にそれらの 3 つの特徴を記す．

仲介者の構造化 Dragon イベント配送モデルでは，仲介者を構造化し，複数の分散仲介者を連結している．このため，各仲介者はより単純な機能を備えるだけでよく，組込み機器などの資源制約が多い環境に適している．たとえば，仲介者 A を，各部屋の温度計からの入力の平均値を出力するコンポーネントとして実装し，仲介者 B は，データがある閾値より大きければイベントを発生させ，エアコンを起動するという役割を持たせておく．エアコンの起動条件を平均値から最低値を変更させたい場合，仲介者 A を変化させればよく，仲介者 B を変更する必要はない．このように仲介者を部品化することで仲介者の再利用が可能になる．

経路判断機能の外部化 経路判断機能を 2.1 節で定義したコマンダとして外部化することにより，各コンポーネントの負荷を減らすだけでなく，分散システム全体の挙動を認識することが容易になる．たとえば，GUI により現在のコンポーネント間の構成を即座に変更したい場合，GUI は外部化したコマンダとの通信のみを行えばよい．本モデルでは，経路判断はコマンダに一元化するが，経路情報は各コンポーネントが独自に保持する．このため，コマンダが停止した場合でもシステム全体は停止することがない．

信頼性のある 1 対多イベント配送機構 1 対多のイベント配送手法としてブロードキャスト配送，マルチキャストグループ配送，マルチプルユニキャスト配送の 3 種類がある．ブロードキャスト配送は，ブロードキャストアドレスを用いてすべてのコンポーネントに一度に通知する方法であり，消費者に対するイベントの伝達を確認できないため信頼性はない．マルチキャストグループ配送は，**図 1** が示すように，消費者のグループごとにマルチキャストのチャネルを定義する．信頼性のあるマルチキャスト通信として，リライアブルマルチキャスト[6] などの研究がある．しかし現在，

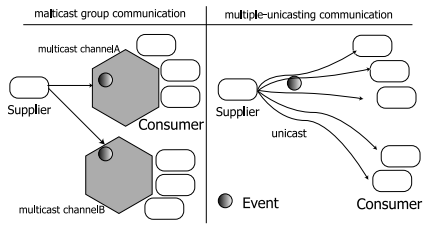


図 1 1 対多イベント配送を実現する手法
Fig. 1 One-to-many communication protocols.

複数のルータを介した現実的なヘテロジニアスなネットワーク環境では，リライアブルマルチキャストを利用しがたい．

逆に，個々のコンポーネントに逐次的に配送を行うマルチプルユニキャストの利用は，信頼性が必要となる場面に適しているが，個々に配送を行うため伝達時間に揺らぎが生じる．これを回避するために著者らは，優先度をつけたマルチプルユニキャストを新たに提案し，重要なイベント通知に関しては信頼性を持たせる．信頼性のある 1 対多イベント配送の実現により，生産工場などにおいて，機械間での信頼性を持ったイベント駆動の連携作業システムの構築が可能になる．

3. 分散コンポーネント基盤機構 Dragon

筆者らは，前章で述べたモデルを基に分散アプリケーションを即興的に構築可能なコンポーネント基盤機構 Dragon[5),13)]を Java2 上で実装した．

3.1 Dragon の実装方針

2.2 節の即興的分散システム構築を可能にする機能要件に従い，Dragon の実装方針を示す．

コンポーネント障害検知機能 分散処理技術 Jini1.1 の Lease 機能を利用する．各コンポーネントは，定期的に Lease 更新を行い，自分自身が起動状態にあることを Jini LookupService に通知する．Lease 更新がない場合は，コンポーネント障害と見なし，コマンダに即座に通知を行う．これにより，利用者も即座に分散システム内のコンポーネントの挙動を検知できる．

近接コンポーネント認識機能 Jini1.1のLookupService を利用する．各コンポーネントは，マルチキャスト通信可能な領域内であるか，あるいは LookupService アドレスをあらかじめ認識していることを前提としている．LookupService 自身の障害にそなえて，LookupService を 2 個以上同時に起動することも可能とする．

経路判断機能 経路判断は，Commander クラスに一元化する．このクラスの API を用いて，利用者がコンポーネント間を接続する GUI を構築できる．Commander クラスが障害により終了する場合に対処できるように，経路情報は各コンポーネントに分散して保持する．Commander は，起動されるたびに各コンポーネントから通知先のリストを取得する．

コンテンツ適応機能 イベントコンテンツ変換を専門に扱うクラスとしてフィルタオブジェクトを定義する．このフィルタオブジェクトを持った仲介者を複数接続して，生産者が発したイベントを消費者が利用するイベントに変換する．

信頼性のある 1 対多イベント配送機能 Dragon では，マルチプルユニキャストにより通信を行う．**ソフトリアルタイム配送** と呼ぶ優先的に行う配送と**ノンリアルタイム配送**と呼ぶ優先度の低い配送を区別する．ソフトリアルタイム配送は，ノンリアルタイム配送の周期が非常に短い場合や配送先が多い場合でも，イベントが優先的に伝達される．

システム障害可能性検知機能 システム内においてノンリアルタイム配送が大量に発生した場合に，イベントのオーバフローを検知し，ソフトリアルタイム配送の状況の変換を Commander クラスに収集する，各コンポーネントからは，通信先ごとの配送の平均時間と最悪時間などが伝えられ，利用者がすぐさま状況を認識することが可能となっている．また，デッドラインをミスするソフトリアルタイム配送を増やしすぎないようにソフトリアルタイムイベント配送登録の制約を行い，システム障害の可能性を避ける．

3.2 コンポーネントを構成するモジュール群

Dragon の各コンポーネントは，以下に説明するモジュールの組合せで構成する．各モジュールは，**表 1** のメソッドを持つ．*BasicService* クラスのインスタンスがこれらのモジュールを保持する．BasicService クラスは，自動生成されたコンポーネント固有の番号である *SerialID* を持つ．この ID は，ホスト名＋サービス名＋起動された時間＋ランダム定数より生成され，コンポーネントを一意に識別するために利用する．

イベント入力処理モジュール（EIM） 到着するイベントをハンドルするモジュールである．RMI のインタフェース *notify()*，*notifyRT()* の引数として，イベントオブジェクトが渡される．このように，オブジェクトの受け渡しのみに限定することでコンポーネント間通信の複雑化を避けている．イベントオブジェクトとしては，*java.io.Serializable* を実装したあらゆるオブジェクトおよび *java.rmi.MarshalledObject* を実装したリモート呼び出し可能なオブジェクトの 2 種類を配送可能である．受け取ったイベントに対して現在処理可能な状態であるかを判断し，以下に示すイベント

表 1 各モジュールが保持するメソッド一覧
Table 1 Methods of each module.

モジュール	メソッド	説明
EIM	*notify(Object evt)*	イベント通知に利用
EIM	*notifyRT(Object evt)*	リアルタイムイベント通知に利用
EFM	*updateFilter(FilterObject newfilter)*	フィルタを変更する
EFM	*int filterEvent(Object evt)*	フィルタを実行させるメソッド．返り値は拒否：0，通過：1，変更：2，待機：3
EFM	*Object getNewEvent()*	配送するイベントオブジェクトを取得
EOM	*registerRemoteEventListener()*	通常のイベント配送登録を行う
EOM	*registerRemoteEventListenerRT()*	リアルタイムイベント配送登録を行う
EOM	*cancelRemoteEventListener()*	イベント配送登録のキャンセルを行う
SM	*generateEvent(Object evt)*	イベントの生成時に呼び出す
AM	*abstract void action(Object evt)*	イベントを受け取った後の挙動を記述
CMM	connectComponents(c1,c2)	ノンリアルタイム配送接続
CMM	connectComponentsRT(c1,c2,deadline)	ソフトリアルタイム配送接続
CMM	disconnectComponents(c1,c2)	イベント配送解除
CMM	getSupplierComponents()	イベント生産者を取得
CMM	getConsumerComponents()	イベント消費者を取得
CMM	getMediatorComponents()	イベント仲介者を取得

フィルタモジュールおよびアクションモジュールに受け渡す．

イベントフィルタモジュール（EFM） イベントの内部をチェックし，以下のイベント出力処理モジュールへ通すか通さないかを規定する．本モジュール内のフィルタオブジェクトは，*updateFilter(FilterObject newfilter)* メソッドによって変更可能である．

イベント出力処理モジュール（EOM） 1 対多でイベントを出力するモジュールである．イベント配送登録を行う場合，まずコマンダが下流にあるコンポーネントの参照を配送依頼する上流コンポーネントの EOM に対して登録する．EOM は，配送先への参照を記録したテーブルを保持している．配送登録には，ソフトリアルタイム配送を要求する登録とノンリアルタイム配送登録の 2 種類がある．その登録に基づき，イベントを受け取った上流のコンポーネントは，登録されている配送先に対してイベント配送を行う．

センサモジュール（SM） センサモジュールは，主にセンサなどの入力機器の状態変化に対するイベントを非同期で発生させるメソッドを持つ．情報家電機器などのコンポーネントプログラマによって記述されるモジュールである．状態変化が起こると *generateEvent(evt)* メソッドを起動する．

アクションモジュール（AM） イベントを受け取ったあとの挙動を記述する部分である．ホストに依存している処理内容を，ActionModule の *action(evt)* メソッド内部に，実装クラスでコンポーネントプログラマが記述する．

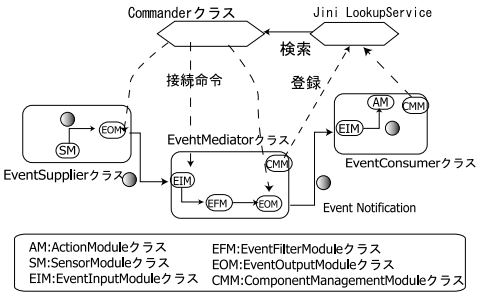


図 2 イベント配送登録とコンポーネント内部のモジュール
Fig. 2 Event propagation for distributed components.

3.3 イベント配送を担うコンポーネントタイプ

Dragon 内でのコンポーネントは，上述したモジュールの組合せとして定義される BasicService クラスが基本となる．著者らは，コンポーネントをよりシンプルで，プログラマが開発を行いやすくするために，**図 2** のように目的別にサブクラスを提供した．

EventSupplier クラス EOM を保持している生産者コンポーネントである．本クラスのインスタンスの生成時には，SensorModule（SM）をコンストラクタに引き渡す．EventSupplier クラスを用いたコンポーネントの実装例を付録 A.1 に記す．

EventMediator クラス 複数の仲介者を連結できることを前提とし，EOM，EIM のモジュールを持つ仲介者コンポーネントとなっている．EventMediator クラスのインスタンスの生成には EFM が必要になる．

EventConsumer クラス EIM イベントのイベント到着を待機し，イベントの内容に基づいた処理を行

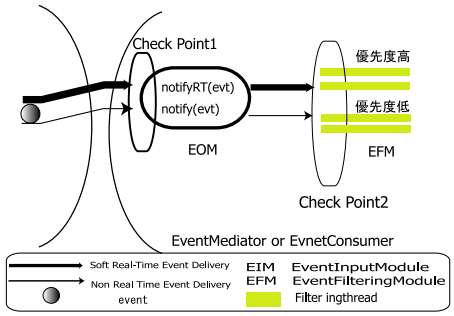


図 3 EIM 内部でのスレッド数の制御
Fig. 3 Filtering thread control of Event Input Module.

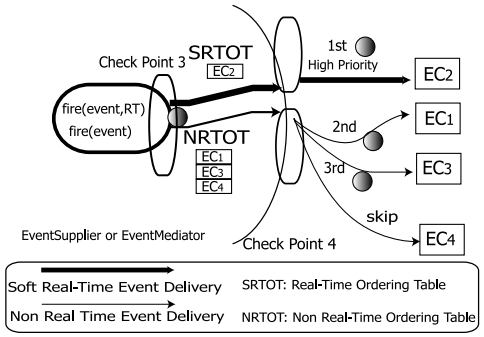


図 4 EOM での優先度区別したイベント配送
Fig. 4 Priority-based event delivering of Event Output Module.

う消費者コンポーネントである．処理内容は AM 内部で記述する．

3.4 コンポーネント間接続命令機構

コンポーネントを組み合わせて分散アプリケーションを構成するために Commander クラス（**CMM**）を提供している．同クラスは，Dragon イベント配送モデルでいうコマンドにあたる役割を持ち，コンポーネント間を接続させる命令を生成する．同クラスが持つメソッドを表 1 に示す．

3.5 優先度をつけたマルチプルユニキャスト配送

1 対多イベント配送をサポートし，マルチプルユニキャストを行いながら伝達時間に揺らぎを与えないために，優先度をつけたイベント配送の手法を実現する．ソフトリアルタイムイベント配送メカニズム[4]では，デッドラインまでに優先的にユニキャスト配送するものと配送を保証しないものとに分割する．

3.5.1 ソフトリアルタイム配送を受けた側の処理

ソフトリアルタイム配送のイベントを受けた EIM では，そのイベントに対する処理が優先的に実行されることをできる限り保証する．

図 3 中の CheckPoint1 では，イベント配送元のコンポーネントから *notify()* が起動される．起動と同時に現在までイベントを受け付けてからフィルタリングを終了していないスレッドの総数を数え，規定値を超えていなければ到着したイベントを EFM に渡す．ノンリアルタイム配送のイベントは，CheckPoint2 において，現時点でフィルタリングを行っているスレッド数を検査し，高優先度のスレッド数が規定値以下になるまで待機し，一定時間後にスレッド数が規定値以下にならない場合はスキップする．

3.5.2 ソフトリアルタイム配送を行う側の処理

EOM でのイベントの配送時には，**図 4** に示すようにイベント配送先の優先度区別を行っている．ノンリアルタイム配送は，ソフトリアルタイム配送を行っていないことを CheckPoint3 および CheckPoint4 で検査し，パスした場合に下流のコンポーネントへ配送を行う．仮にノンリアルタイムの配送がすべて終わらないうちに次のイベントが到着した場合は，ノンリアルタイム配送の登録をされている通知先をスキップし，ソフトリアルタイムのイベント配送を優先する．

3.5.3 ソフトリアルタイムイベント配送登録の制約

ノンリアルタイム配送は，リアルタイム配送に支障がない場合ベストエフォートで行われるが，すべてのコンポーネントがソフトリアルタイム配送を要求できるわけではない．ソフトリアルタイム配送を保証するために，新たなソフトリアルタイム配送キュー（$SRTOT$）への登録要求は，以下の項目を満たすときに実行される．$SRTOT$ には，通知先関連情報 $O_i(i = 1, 2, 3...n)$ が入っている．各 O_i には，通知先のスタブである $target$，デッドライン（ミリ秒），イベントを受け取ってからターゲットに配送を完了するまでの最悪配送時間 $worstcasetime$ が記入されている．つまり O_i は，$O_i = \{target, deadline, worstcasetime\}$ となる．ここで新たな $O_{new} = \{target, deadline, \phi\}$ なる配送先から要求があった場合，$O_{new}.deadline < O_i.deadline$ なるすべての i に対して $O_i.\ worstcasetime + durtime(O_{new}) < O_i.deadline$ を満たすならば登録を受け付け，i 番目として $SRTOT$ に挿入する．これによりデッドラインを保証でき，既存の登録を無効化するようなソフトリアルタイム配送が追加されることはない．配送する場合は，$SRTOT$ の i 番目から順に *notifyRT()* メソッドを呼び出す．

4. 構築可能な即興的分散アプリケーション

本章では，Dragon を用いて構築可能な分散アプリケーションの特徴と構築方法を説明し，Dragon の有用性を示す．

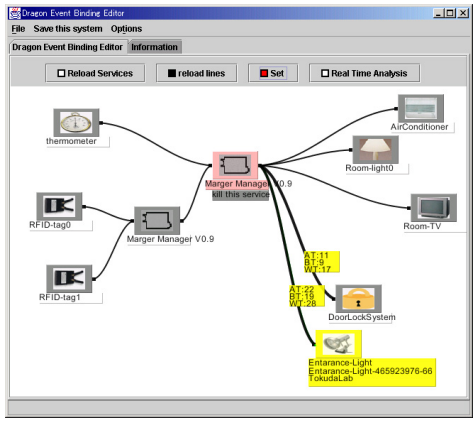


図 5 GUI の利用時のスクリーンショット
Fig. 5 A screen shot of Dragon Event Binding editor.

即興的構築を可能にする UI 開発の容易性

3.4 節で示した Commander クラスの API を用いて実装された Event Binding Editor を，**図 5** に示す．Event Binding Editor は，高度な知識を持ったシステム開発者が存在しない場面で即興的システム構築を可能にするためのツールである．ローカルコンポーネントのみを扱う視覚プログラミングモデルの Java Beans とは違い，Dragon での GUI は完全に分散されたコンポーネント間の接続の設定を行い，すぐさまその設定が分散システム全体に反映される．この GUI の基盤となる Commander クラスは，新たに起動されたコンポーネントや障害が起きたコンポーネントを認識する API も備えているため，新たにユーザインタフェースを開発することも容易である．

ユビキタス機器環境に適した単純なインタフェース

図 5 は，ネットワーク上で 3 つの生産者と 2 つの仲介者と 5 つのイベント消費者を起動し，GUI により接続命令を行った設定画面である．たとえばこの例は，家庭内のユーザ認識タグと玄関のライト，室内エアコン，室内 TV などをユーザが接続し，特定の人物が帰宅した場合に行う処理を設定した様子である．仕様や目的の異なる情報家電機器どうしを接続するため Dragon では *notify()* と *notifyRT()* という 2 つのメソッドのみを規定し，受け渡されるイベントで差異を吸収させている．図 5 では，アイコン間の線がコンポーネント間が接続されイベントが配送されていることを示している．ユーザはマウスで操作し，このアイコン間に線を引くという単純な作業でコンポーネント間の接続を行える．

分散アプリケーションにおける障害可能性の検知

ソフトリアルタイム配送を行った場合は，各線の上に過去 50 回のイベント配送に関する最短時間，平均時間，最悪時間が表示され，イベントのオーバフローを認識できる．これは 3.5.3 項に示したように，各コンポーネントが配送完了時間をつねに測定し履歴として保存しているためである．図 5 の例では，ドアキーと玄関ライトはすばやく配送される．他のコンポーネントへはノンリアルタイム配送の命令になっている．リアルタイム性を示すフィードバックのある設定は，Carp@などの研究では行えず，Dragon でのみ実現している．

5. Dragon の基本性能の評価

本章では 3.1 節で述べた近接コンポーネント認識機能，経路判断機能，コンテンツ適応機能，信頼性のある 1 対多イベント配送機能，システム障害可能性検知機能の実装方針が正しく動作していることを Dragon の基本性能を含めて以下で示す．測定環境には，100Base-T で相互接続された，**表 2** に示す機器を利用した．測定用コンポーネントには，1 回ごとに 8 バイトの付加情報を保持させたイベントの配送を行う．

5.1 コンポーネント発見完了と障害検知時間

経路判断機能を持つ Commander クラスは，近接コンポーネントの発見と，障害も把握も行わなければならない．そのために本評価では，コンポーネントが起動されてから Commander クラスに検出され GUI に反映されるまでの時間と，コンポーネントがダウンしてしまった場合の障害検知時間を測定した．**図 6** の横軸は各コンポーネントが Lease を行う周期を表し，縦軸に Commander クラスに検出されるまでの経過時間を表している．それぞれの Lease 時間でコンポーネントを動作させ 100 回の試行をし，その平均値を示している．コンポーネント障害検知機能は，Lease 周期の設定を短くすることで障害から 10 秒以下で反映されることが可能になり，家庭内などでは実用に耐えるといえる☆．発見完了時間は，ほとんど Lease に影響はなく，100 ミリ秒以内に Commander クラスに伝達され近接コンポーネント認識から経路判断が行える．

5.2 イベント仲介者群の直列な配送時間

Dragon では，コンテンツ適応機能を複数の構造的なイベント仲介者群で行っている．このためイベント仲介者が直列に接続された場合のオーバヘッドを測定し，コンテンツ適応機能の妨げにならない値であるかを調べる．マシン A とマシン B の間で起動しているイベント仲介者間を交互に接続し，イベント仲介者間

☆ ただし，Lease 周期を短くしすぎると Lease 更新にかかる通信コストは増加してしまう．

表 2 測定環境
Table 2 Platforms for evaluation.

項目	マシン A	マシン B	マシン C
CPU	Duron800M	UltraSPARCII248M×2	PentiumIII750M
主記憶	128 MB	512 MB	256 MB
OS	FreeBSD4.4	Solaris7	Windows2000
JDK	JDK1.3.1	HotSpot1.3.1	HotSpot1.3.0

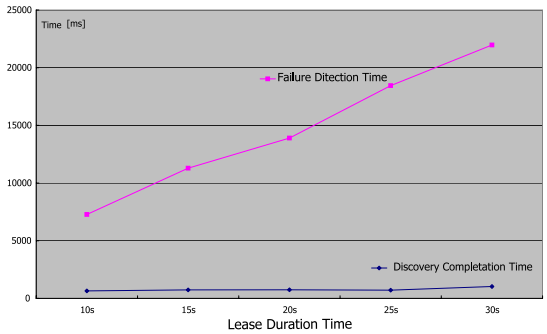


図 6 発見完了時間および障害検知時間
Fig. 6 Discovery completion time and failure detection time.

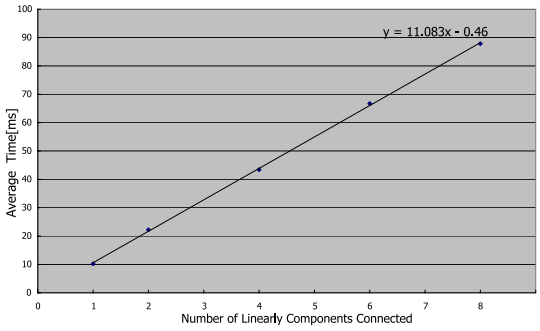


図 7 仲介者の直列配送の時間測定
Fig. 7 Linear delivering duration time.

を直列的に配送するために要する時間を測定する．イベント仲介者内部の処理は行わない．コンポーネントが配送を終えるまでの時間を正確に測定するために，イベント生産者とイベント消費者は同一のホスト上で実行し測定した．その結果，**図 7** に示す結果が得られた．全配送時間を y，直列につながっているコンポーネントの数を x とすると，式 $y = 11.083x - 0.46$ で近似できるため，直列に並ぶコンポーネントの数が多くなっても影響を受ける時間は，予測可能である．この結果，コンテンツ適応機能のオーバヘッドを 1 秒と許容すると，約 90 個のイベント仲介者を直列に配置可能であり，家庭ネットワークや小規模ネットワークでの実用に十分である．

5.3 イベント入力時およびフィルタ処理時におけるソフトリアルタイム性の評価

Dragon で信頼性のある 1 対多イベント配送機能が満たされていること評価するために，まずソフトリアルタイムイベント配送は，優先的にフィルタ処理が行われていることを示す．本測定には以下の正規分布に基づく記号を利用する．$NonRTN(p, d)$ は，分散 d ミリ秒，平均周期 p ミリ秒で上流のコンポーネントからイベントが到着している周期を表す．同様に $RTN(p, d)$ は，分散 d ミリ秒，平均周期 p ミリ秒で上流のコンポーネントからソフトリアルタイム配送でイベントが到着している周期を示している．

図 8 では，マシン C においてあるイベント仲介者に

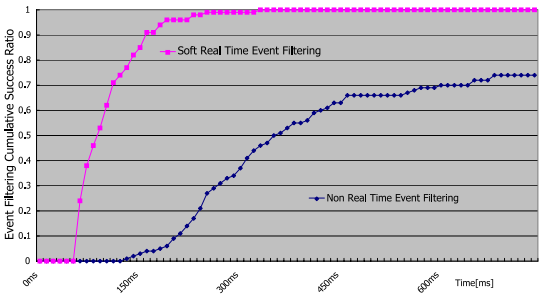


図 8 $NonRTN(80, 5)$, $RTN(80, 5)$ 周期のフィルタ成功率
Fig. 8 Success ratio and timing at $NonRTN(80, 5)$, $RTN(80, 5)$.

おいて周期 $NonRTN(80, 5)$ と周期 $RTN(80, 5)$ のイベントが到着している状態を示している．そのなかで，それぞれのイベントが入力からフィルタで処理終了されるまでの時間を横軸にとり，縦軸にそのイベントのフィルタ累積成功率を示す．フィルタの負荷として 0 から 15,000 の間から素数の数をカウントするダミー処理を行う．**図 9** は，ノンリアルタイム配送の周期を $NonRTN(20, 5)$ にし，負荷を高くした時点の状態を表している．図 8 と図 9 を比較すると，ノンリアルタイム配送の成功率が低下しているが，ソフトリアルタイム配送の処理には影響を与えていないことが累積成功確率曲線から分かり，イベント処理時に関して，ソフトリアルタイム配送が行われていることが分かる．これにより，信頼性のある時間見積りが可能であれば，最悪時間を測定した結果を基に行うシステ

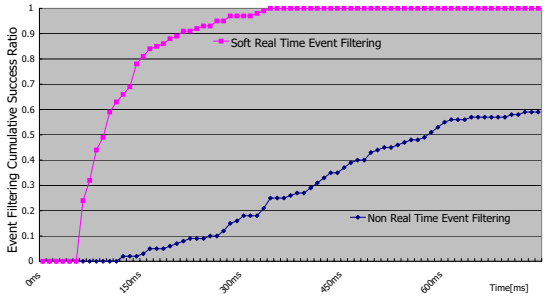


図 9 $NonRTN(20,5)$，$RTN(80,5)$ 周期のフィルタ成功率
Fig. 9 Success ratio and timing at $NonRTN(20,5)$, $RTN(80,5)$.

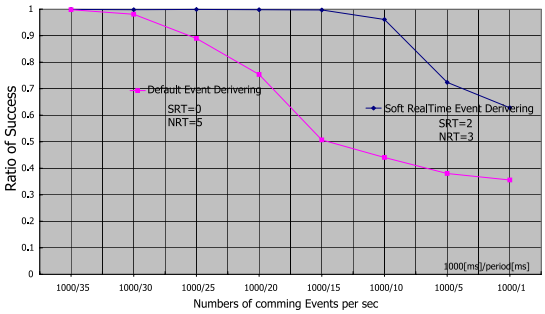


図 10 1 対多イベント配送の成功率
Fig. 10 Ratio of successfully delivered events.

ム障害可能性検知機能が有効になる．

5.4 イベント配送時のリアルタイム性の評価

イベント配送時にソフトリアルタイム配送機構がある場合とない場合での評価を行った．**図 10** では横軸に 1 秒間のイベント数をとり，縦軸にそのイベントがマシン B 上の 5 つの消費者コンポーネントに配送成功する率を測定した．ソフトリアルタイム配送では，5 つの消費者のうち 2 つをソフトリアルタイム配送に登録し，その他のものをノンリアルタイム配送として登録した．ノンリアルタイム配送の場合は，5 つの消費者にイベントを配送する要求を制御しないまま 1 対多イベント配送を行っている．通常，1 つのイベントの到着に対して 5 つのイベントが配送されるが，処理しきれないイベントは破棄されていく．図 9 では，通常の配送ではイベント配送の 1 秒あたりのイベント数（1/配送周期）が高まると成功率が下がっているが，ソフトリアルタイム配送は高い配送成功率を保っているため，信頼性のある 1 対多イベント配送が実現できていることが分かる．特に周期 15 ミリ秒での比較は，デフォルトのマルチプルユニキャストの 50%近く高い成功率を示し，優先度をつけたマルチプルユニキャストが正しく動作していることを示している．

6. 関連研究との比較

本章では，分散コンポーネント間通信を行うミドルウェアと Dragon を比較する．各コンポーネントが単純に実装されることは，組込みシステムや既存のコンポーネントを再利用する際に重要となるため，コンポーネント間のイベントの流れの信頼性と各コンポーネントの単純さを指標とし，関連研究をトポロジによって分類しながら比較を行う．

既存の分散ミドルウェアが提供する分散コンポーネント通信は，イベント生産者と消費者の間で直接送信されているものと，間接的に送信されているものに分類できる．さらに通信の主導者がどの種のコンポーネントであるかを基準に，**図 11** で示すように分類し，即興的な分散システム構成に対する適性を比較する☆．

6.1 直接通信トポロジ

直接通信トポロジとは，イベント生産者と消費者が直接通信を行うトポロジである．

全通知型直接通信モデル イベント生産者が，ブロードキャストによりイベントを配送させる手法である．全通知型直接通信モデルは，1 対多イベント配送としてブロードキャストを用いているため，生産者側は消費者側を考慮する必要がなく，経路判断を行うコンポーネントは存在しない．そのため，単一の目的コンポーネントにより構成される分散システムで信頼性が必要ない場合は，イベントの伝達が一度になり効率が良い．しかし，消費者はすべての生産者からのイベントに対して自らに関係があるかどうかを判断する必要がある．つまり，消費者側に高度な適応性がなければならない．多様な目的のコンポーネントが介在した場合，生産者の量的なスケーラビリティや，イベントの発生する周期が短い場合に消費者の処理が追いつけない．

生産者主導型直接通信モデル 各コンポーネントが互いの存在をあらかじめ認知していない場合に，ディレクトリサービスやブロードキャストアドレスを利用して互いの存在を認知し，生産者側からダイレクトに通信を行うモデルである．JEcho は，クライアントグループごとにチャネルを定義し，そのチャネルに基づいて高速にイベント通知を行う．また，独自の名前空間を持ち生産者とチャネルを紹介している．しかし直接的な通信トポロジのため，プログラマは互いのインタフェースを熟知していなければならない．また

☆ 本章で述べきれない，同期性も含めたさらに詳しい既存の分散ミドルウェアの考察は文献 14) で述べている．

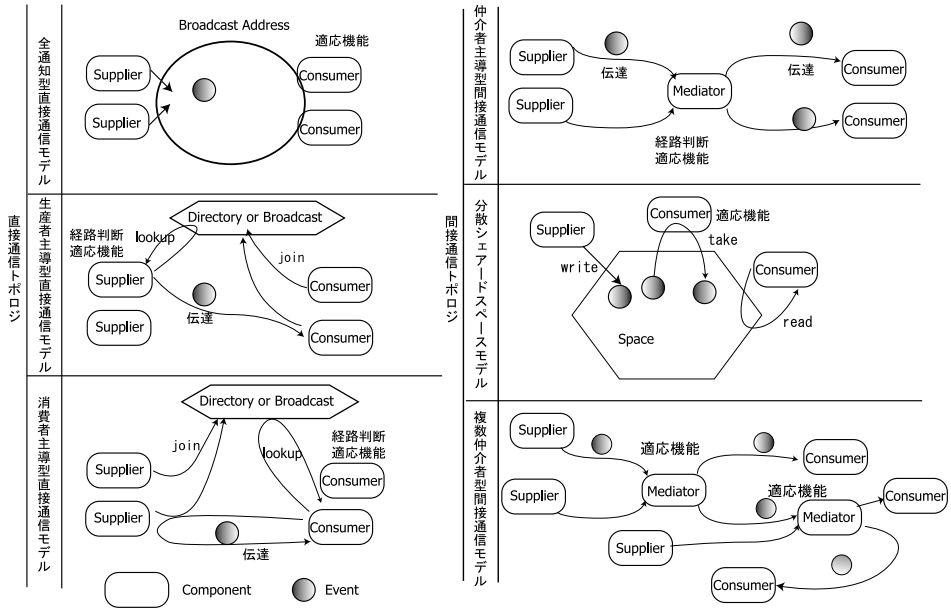


図 11 分散コンポーネント通信の分類
Fig. 11 Topology classification of communication for distributed components.

JECho のモデルでは，生産者が増加した場合，消費者がそのイベントを扱いきれない問題が起こる．一方 Dragon イベント配送モデルは，複数の仲介者を介した間接的な通信モデルであり，すべてのコンポーネントのインタフェースを知らずにイベント仲介者が変換を行える．

消費者主導型直接通信モデル イベント消費者が能動的に，ディレクトリサービスやブロードキャストアドレスを利用し生産者を認識する．その中から生産者を決定し配送要求を行う．Jini Distributed Event Model[10]，VNA[7]などがこのモデルにあてはまる．Jini の Distributed Event Model に対する設定用 GUI ツールを提供している Carp@は，バイトコードジェネレータを用いて，動的に Jini のサービスをラップさせるオブジェクトを生成する．このラッパーオブジェクトにより，GUI から操作可能な Carp@Bean と呼ばれるサービスとなる．*port* と *connector* を用いて Jini の詳細なメカニズムを認識することなくコンポーネント間のつながりを即興的に設定できる．しかし既存の Jini のサービスからは，イベントの仲介を行うサービスは生成されず，やはり既知の Jini サービス間どうしの単純な通信になってしまう．そのためヘテロジニアスな分散アプリケーションを構築することは難しい．Dragon では，内部を通るイベントオブジェクトを自由に変更可能であるので，Jini を基盤としないコンポーネントを含めた分散アプリケーションを構築できる．また，Dragon では，Carp@では提供されていない 1 対多のイベント配送の時間制約の提示と設定が可能であり，時間や配送の信頼性に厳密さを要するアプリケーションにも応用できる．

6.2 間接通信トポロジ

イベント生産者と消費者が直接通信を行わず，間接的に通信を行うトポロジである．

仲介者主導型間接通信モデル システム内で唯一特別なイベント仲介者を定義し，各コンポーネントはそのイベント仲介者を認知しておく．イベント仲介者は，すべてのコンポーネントを認知している．仲介者はイベントが送られると，すぐさまコンポーネントを選択し通知する．生産者と消費者は互いを認知している必要はなく，イベントの変換と引き渡しは，イベント仲介者が代理する．CORBA Event Service は，イベント消費者に対して，*push* メソッドのみを用いれば，仲介者主導型間接通信モデルの典型であるが，*pull* メソッドを備えているため一時的なオブジェクトの格納を行う役割を持たせることも可能である．また，CORBA Event Service は，*push* メソッド，*pull* メソッドインタフェースを持ち，コンポーネント開発者はこのインタフェースに基づいて仲介者を実装できる．しかし複数のコンポーネントがある中で，即興的に分散システムを構築したい場合，すべてのコンポーネントの組合せに必要な変換を想定し，中心的な仲介者である Event Service を作り込む必要がある．そのため多様なコン

表 3 コンポーネント間通信機構の評価
Table 3 Classification of distributed components communication mechanisms.

sモデル名（代表的な研究名）	経路判断	適応機能	1対多	送り手	仲介者	受け手	Directory
全通知型直接通信モデル	なし	受け手	必要	単純	なし	複雑	無し
送り手主導型直接通信モデル（JECho）	送り手	送り手	必要	複雑	なし	単純	有り
受け手主導型直接通信モデル（Carp@）	受け手	受け手	不要	単純	なし	複雑	有り
仲介者主導型間接通信モデル（CORBA）	仲介者	仲介者	必要	単純	複雑	単純	無し
シェアード スペースモデル（JavaSpaces）	なし	受け手	不要	単純	単純	複雑	無し
複数仲介者型間接通信モデル（SIENA）	各仲介者	仲介者群	必要	単純	多少複雑	単純	無し
Dragon イベント配送モデル（Dragon）	外部化	構造化仲介者群	信頼性有	単純	単純	単純	有り

ポーネントが存在し，各機器と低機能であることを前提とするホームネットワークなどの分散システム構築には向かない．

分散シェアード スペースモデル ネットワーク内部にシェアード スペースを定義し，処理内容を通知したり，処理結果をシェアード スペースに書き込む消費者主導のモデルである．スペース側は，オブジェクト置き場としての役割を果たし，受動的に振る舞う．JavaSpaces，Linda[3]，TSpaces[11]などがこの例である．JavaSpaces では *read()*，*write()*，*take()* という 3 つのオペレーションをスペースに対して行うことが可能である．引き渡されるオブジェクトは，値渡しでなければならない．クライアントが *take()* を行い，一度オブジェクトに対して作業を完了すると，*write()* によって再びスペースに戻す作業を行う．単純な計算処理を複数に分割し，分散処理させる場合などに適する．各コンポーネントは互いに認知する必要はないが，共通のフォーマットを認識している必要がある．クライアントが互いを認知することがない即興的な分散処理システム構築には有効であるが，複数のコンポーネントからなるヘテロジニアスな環境でオブジェクトを相互に理解することには不向きである．一方 Dragon は，構造的な仲介者群を用いているため，イベントのコンテンツの変換を行う EventMediator クラスが介在できる．相互性のないコンポーネント間の異なるイベントコンテンツによるプリケーションが構築できる．

複数仲介者型間接通信モデル SIENA[1]は，仲介者を構造化でき，本研究も採用している複数仲介者型間接通信モデルにあてはまる．SIENA は厳密に型付けされたイベントと *pattern* と呼ばれるイベントのマッチングルールにより，複数の仲介者にわたってイベントのルーティングを動的に行うことが可能である．一方 Dragon は，RMI によるオブジェクト渡しであればどのようなイベントでも配送可能である．Dragon は，個々のコンポーネントの機能を軽量化するという制約のもと，コンポーネント自らが自律的なルーティングは行わない．しかし，外部化した経路判断者である Commander クラスは，GUI でユーザに経路を設定させるだけでなく，独自のアルゴリズムによって経路判断を行うことも可能に設計してある☆．

6.3 考 察

以上の本章の通信トポロジによる分類をふまえた考察結果を**表 3** に示した．表中の**複雑**とは，経路判断あるいは適応機能をコンポーネント内部で行っていることを示す．**単純**とはその逆であり，コンポーネントが独自のロジックのみに集中している状態を表している．Dragon イベント配送モデルは，各コンポーネントを単純に実装できる特徴と信頼性のある 1 対多イベント配送があり，ホームネットワークにおいて，組込み機器向けの利用ができる．

7. おわりに

コンポーネントソフトウェアによる分散システムには，高度な知識を持った開発者が必要とされてきた．そのため近年の家庭内ネットワークなどのシステム開発者の介在が困難な場面で，コンポーネント技術は有効利用されていなかった．本論文では高度な知識を持ったシステム開発者が介さなくても，一般的なユーザが，即興的に分散システムを構成可能なコンポーネント接続ミドルウェア Dragon を提案した．Dragon は，経路判断機能の外部化，コンテンツ適応機能の構造化，信頼性のある 1 対多イベント配送機能などの特徴を持ち，構成要素の各コンポーネントが単純な機能を持つモデルである．本論文で Dragon の各モジュールの実装と，ユーザが即興的に分散システムを構築可能とする GUI ツールを示した．

評価において Dragon は，仲介者コンポーネントである EventMediator クラスを連結した配送のオーバヘッドを測定した結果，オーバヘッドが 1 秒以上となるには約 90 個の連結まで許容できることが示せたた

☆ 自律的なアプリケーションレベルルーティングは，Dragon の想定しているような比較的小規模のネットワークでは必要ない．しかし将来的には，Dragon をインターネットスケールに拡張させる研究に合わせて課題としていく．

め，小規模ネットワークで稼動可能といえる．さらにDragonは，20ミリ秒の周期という負荷の高いノンリアルタイムイベント配送がある中で，ソフトリアルタイム性を持たせたイベント処理は負荷が80ミリ秒のときとほぼ同一の累積成功確率曲線を示し，負荷の高い配送においても優先度処理が保たれていることを示した．周期15ミリ秒の1対多イベント配送においては，優先度を区別したマルチプルユニキャストが，デフォルトの50%以上も成功率が高い結果を得られた．そのためDragonは，家庭内ネットワークだけでなく時間制約のある工場などのシステムに対する応用も想定できる．

今後は，RMI以外の情報配送手法への適応機能，インターネットスケールでの自律的イベントルーティング機能，ストリーミングメディアコンテンツ対応機能などを追加し，応用範囲の広い分散アプリケーション構成機構への発展を予定している☆．

謝辞 本研究を進めるにあたって，ご協力いただいた慶応義塾大学徳田研究会の方々，ならびに．HOMEプロジェクトの方々に深く感謝いたします．

参考文献

1) Carzaniga, A., Rosenblum, D.S. and Wolf, A.L.: Achieving Expressiveness and Scalability in an Internet-Scale Event Notification Service, *19th ACM Symposium on Principles of Distributed Computing* (July 2000).
2) Fahrmair, M., Salzmann, C. and Schoenmakers, M.: A Reflection Based Tool for Observing Jini Services, *Reflection and Software Engineering*, pp.209–227 (June 2000).
3) Gelernter, D.: Generative Communication in Linda, *ACM Trans. Prog. Lang. Syst.*, Vol.7, No.1, pp.80–112 (1985).
4) Iwai, M., Nakazawa, J. and Tokuda, H.: Dragon: Soft Real-Time Event Delivering Architecture for Networked Sensors and Appliances, *The 7th International Conference on Real-Time Computing System and Applications*, pp.425–432 (Dec. 2000).
5) Iwai, M., Nakazawa, J. and Tokuda, H.: Flexible Distributed Event-Driven Programming Framework for Networked Appliances and Sensors, *Proc. 3rd International Symposium on Distributed Objects and Applications Short Papers*, pp.61–68 (Sept. 2001).
6) Paul, S., Sabnani, K.K., Lin, J.C. and Bhattacharyya, S.: Reliable Multicast Transport Protocol (RMTP), *IEEE Journal on Selected Areas in Communications*, Vol.15, No.3, pp.407–421 (1997).
7) Nakazawa, J., Tobe, Y. and Tokuda, H.: On Dynamic Service Integration in VNA Architecture, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.7, No.E84-A, pp.1610–1623 (July 2001).
8) Object Management Group: *The Common Object Request Broker Architecture and Specification 2.2ed CORBA Event Service* (Feb.1998).
9) Sun Microsystems Inc.: *JavaSpaces Service Specification*, version 1.1 (Oct. 2000). http://www.sun.com/jini/specs/js1_1.pdf.
10) Sun Microsystems Inc.: *Jini Architecture Specification* (Oct. 2000).
11) Wycko, P., McLaughry, S., Lehman, T. and Ford, D.: Ibm systems journal, TSpaces, *IBM Systems Journal*, Vol.37, No.3 (1998).
12) Zhou, D., Schwan, K., Eisenhauer, G. and Chen, Y.: JECho—Interactive High Performance Computing with Java Event Channels, *International Parallel and Distributed Processing Symposium*, (Apr. 2001).
13) 岩井将行，楠本晶彦，中澤 仁，徳田英幸：情報家電機器間の動的なイベントバインディング機構の構築，情報処理学会システムソフトウェアとオペレーティングシステム研究会，pp.39–46 (May 2000).
14) 岩井将行，中澤 仁，徳田英幸：広域および小規模分散コンポーネント間通信機構の評価．マルチメディア通信と分散処理第105回研究会，pp.19–24, 情報処理学会 (Nov. 2001).

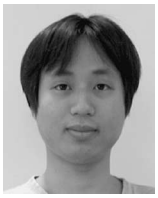
☆ 本ミドルウェアは，**http://www.ht.sfc.keio.ac.jp/~tailor/dragon**からソースコード，バイナリ，API，マニュアル，評価などが閲覧および取得可能である．

付　　録

A.1　イベント生産コンポーネントサンプルコード

```
public class SampleSupplierComponent{
  static int POLLING_TIME=100;
   //周期 100 ミリ秒でイベントを配信
   public static void main(String[] args){
      SensorModuleInnerImple sm
       = new SensorModuleInnerImple();
      ... 略. ..
      EventSupplier es
       = new EventSupplier(hostname,name,sm);
      es.exec();//LUS への登録開始
   }
  static class SensorModuleInnerImple
    extends SensorModule implements Runnable{
      SensorModuleInnerImple(){super();}
    public void run() {
      DToken tokenEvent =new DToken();
      while (true) {
        try {Thread.sleep(POLLING_TIME);
            tokenEvent.reverseToken();
            generateEvent(tokenEvent);
        }catch (InterruptedException ex){}
      }
    }
  }
}
```

(平成 13 年 12 月 20 日受付)
(平成 14 年 4 月 16 日採録)



岩井　将行（学生会員）

2002 年慶應義塾大学大学院政策・メディア研究科修士課程修了．現在同研究科博士課程在学中．分散ミドルウェア，分散イベント通信，情報家電協調制御等の研究に従事．

中澤　仁

1999 年慶應義塾大学大学院政策・メディア研究科修士課程修了．現在同研究科博士課程在学中．分散オブジェクト指向システム，オブジェクト移送ミドルウェア，ホームネットワーク等の研究に従事．

西尾　信彦（正会員）

1992 年東京大学大学院理学系研究科博士課程所定単位取得後退学．1993 年より慶應義塾大学に勤務，同大学より博士（政策・メディア），現在，政策・メディア研究科助教授．分散リアルタイムシステムに関する研究に従事．平成 6 年度山下記念研究賞受賞．

徳田　英幸（正会員）

慶應義塾大学より工学修士．カナダ，ウォータールー大学より Ph.D.（Computer Science）．現在，慶應義塾大学大学院政策・メディア研究科委員長．分散リアルタイムシステム，マルチメディアシステム，通信プロトコル，超並列・超分散システム，ユビキタスシステム等の研究に従事．IEEE，ACM，日本ソフトウェア科学会各会員．