社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

サービス期間を考慮したホームネットワークサービス競合検出・解消システムの実装

池上　弘祐† 　吉村　悠平† 　井垣　　宏† 　中村　匡秀†

† 神戸大学　〒 657–8501 神戸市灘区六甲台町 1–1
E-mail: †{ikegami,yoshimura,igaki,masa-n}@cs.kobe-u.ac.jp

あらまし　ホームネットワークシステムにおいて，複数の家電を連携制御して便利な付加価値サービスを実現する家電連携サービスがある．しかし複数の連携サービスを同時に使用すると，サービス競合が発生し，ユーザーの利便性を大きく損ねてしまう．この問題に対し我々の研究グループでは，サービスの実行期間を考慮してサービス競合をオンラインで動的に検出・解消する手法を提案している．本論文ではこの手法を実装し，我々が開発している実際のホームネットワークシステム上にサービス競合検出・解消システムを構築する．また，実装したシステムを用いて，実用ホームネットワークサービスに対するサービス競合検出・解消実験を行い，有効性の評価を行う．
キーワード　ホームネットワークシステム，サービス競合，競合検出・解消，性能評価

Considering Online Feature Interaction Detection and Resolution for Integrated Services in Home Network System

Kousuke IKEGAMI†, Yuhei YOSHIMURA†, Hiroshi IGAKI†, and Masahide NAKAMURA†

† Kobe University　rokkoudaityou 1–1, nada-ku, Kobe, Hyogo, 657–8501 Japan
E-mail: †{ikegami,yoshimura,igaki,masa-n}@cs.kobe-u.ac.jp

Abstract　The integrated services of the home network system orchestrate multiple networked home appliances to achieve value-added and comfortable services for home users. However, if multiple services are used at the same time, functional conflicts may occur among the services, which significantly declines the quality of service. This is known as *feature interactions (FIs)*. To cope with the problem, we have previously proposed a method that detects and resolves the FIs for the integrated services during runtime. Based on the method, in this paper we implement an FI detection and resolution system on the actual home network system. To show the effectiveness, we conduct an experiment that detects and resolves FIs among practical home network services.
Key words　home network system, feature interactions, interaction detection and resolution, performance

1. はじめに

家庭内の家電機器やセンサをネットワークに接続し，付加価値サービスを実現する**ホームネットワークシステム** *(HNS)* の研究・開発が盛んである．HNS では，テレビ・DVD プレイヤー・エアコン・照明・カーテン等といった家電や，温度計・照度計・騒音計といったセンサをネットワークに接続し，家庭内外から遠隔制御したり，連携制御することで，利用者により便利で快適なサービスを提供する．各家電やセンサの持つサービスの機能は，API(Application Program Interface) としてネットワークに公開され，利用者は，様々な外部アプリケーションからネットワーク越しに各家電やセンサを操作・利用する事が可能である．これらの API を利用して複数の家電やセンサを連携・協調させ，付加価値の高い**連携サービス**を実現することができる．以下に連携サービスの例を示す．

- **シアターサービス**：テレビ・DVD レコーダー・カーテン・照明を連携制御することで，快適に DVD の視聴を行うことが出来るサービス．ユーザーがサービスを要求すると，自動的に照明が暗くなり，カーテンが閉まり，テレビの電源が付き，DVD が再生される．
- **セキュリティ監視サービス**：DVD レコーダー・ＷＥＢカメラを連携制御することで，ＷＥＢカメラの電源を入れ，DVD レコーダーでその映像を一定時間録画するサービス．
- **おかえりサービス**：照明・カーテンを連携制御することで，ユーザーが帰宅した際に照明を付け，カーテンを閉めるサービス．

- **おでかけサービス**：HNS 内の全ての機器を連携制御することで，ユーザーが外出する際に関連した全ての家電の電源を切るサービス．

HNS 上で複数の連携サービスが実行される際，異なるサービスが同一の家電リソースを利用しようとする場合，干渉・衝突が発生することがある．これを**サービス競合** [2] [3] と呼ぶ．サービス競合が発生した場合，連携サービスはユーザーの意図した通りには動作せず，快適性，利便性といった HNS の品質を低下させてしまう．そのためサービス競合を**検出**して**解消**する事が求められる．

上記の連携サービス例において，サービス競合を説明する．ユーザーＡがシアターサービスを実行中に，ユーザーＢがセキュリティ監視サービスを実行した場合を考える．このときユーザーAが DVD を視聴しているにも関わらず，DVD レコーダーはユーザーＢのセキュリティ監視サービスにより，録画を開始してしまう．これは，DVD レコーダーにおいて２つの連携サービスの要求が衝突してしまったことによるサービス競合である．

我々の研究グループは，先行研究 [4] において，HNS におけるサービス競合の動的な検出手法を提案している．この手法では，サービスに対する優先順位を導入し，競合が発生した際に優先度の低いサービスを中止することにより競合解消を図っている．しかしながら，競合検出の厳密なタイミングや，優先度が低いサービスの公平性や実行補償が考慮されていなかった．これらの問題を解決すべく，現在我々は [4] の方法を拡張し，サービスの具体的な有効期間である**サービス期間**，サービスの本質的な機能を保証する **必須メソッド**，低優先度のサービスを一旦中断し後で再開させる**中断・再開機構**を採り入れた，新たな競合検出・解消手法を [5] で提案している．本稿の目的は，その理論をシステムとして実装し，有効性の評価を行うことである．具体的には，我々が以前に開発している実際の HNS [1] 上に，オンラインサービス競合検出・解消システムを構築する．また，実装したシステムを用いて，実用ホームネットワークサービスに対するサービス競合検出・解消実験を行い，有効性の評価を行う．

2. サービス期間を考慮したホームネットワークサービス競合検出解消手法 [5]

2.1 HNS モデルとサービス競合

HNS におけるサービス競合問題を形式化するため，我々は [2] [4] において，オブジェクト指向に基づく HNS のモデル化手法を提案している．この手法では，HNS における各家電機器を，状態 (**プロパティ**) と操作 (**メソッド**) を持つオブジェクトとしてモデル化する．各メソッドは，事前条件と事後条件でモデル化され，それぞれプロパティ上の論理式で表される．図 1 および図 2 はそれぞれプロパティ，メソッドのモデルを表している．この例では，たとえばカーテンは OpenLevel という 1 つのプロパティを持ち，0〜100 の値をとることが定義されている．また，open(), close() という 2 つのメソッドを持ち，open() が実行された後 (事後) には，OpenLevel が 100 になる事が定義されている．

機器	状態	値
DVD_RECORDER	Power	{true,false}
	Input	{0,1}
	Action	{play,stop,recording}
TV	Power	{true,false}
	Input	{0,1,2,3,4}
VideoCamera	Power	{true,false}
CURTAIN	OpenLevel	{0,100}
LIGHT	Power	{true,false}
	Brightness	{0,10}
SPEAKER	Power	{true,false}
	Mode	{2ch,5.1ch}

図 1　機器状態 (プロパティ) モデル

機器	操作(メソッド)	前条件	後条件
DVD_RECORDER	on		Power = true
	off		Power = false
	changeInput(tInput num)	Power = true	Input = num
	play	Power = true	Action = play
	stop	Power = true	Action = stop
	recording	Power = true	Action = recording
TV	on		Power = true
	off		Power = false
	changeInput(tInput num)	Power = true	Input = num
VideoCamera	on		Power = true
	off		Power = false
CURTAIN	open		Openlevel = 100
	close		Openlevel = 0
LIGHT	on		Power = true
	off		Power = false
	setBrightness(tBrightness num)	Power = true	Brightness = num
SPERKER	on		Power = true
	off		Power = false
	changeMode(tMode mode)	Power = true	Mode = mode

図 2　機器操作 (メソッド) モデル

	機器名	メソッド名	パラメータ	優先度	メソッドタイプ	レジューム
シアターサービス	DVD_RECORDER	on		50	必須	再開可
(BeginEnd型)	DVD_RECORDER	changeInput	0	50	必須	再開可
	DVD_RECORDER	play		50	必須	再開可
	TV	on		50	必須	再開可
	TV	changeInput	1	50	必須	再開可
	SPEAKER	changeMode	5.1 ch	50	必須	再開可
	LIGHT	setBrightness	1	50	通常	再開可
	CURTAIN	close		50	通常	再開可
セキュリティ監視サービス	DVD_RECORDER	on		100	必須	再開可
(Timer型)	DVD_RECORDER	changeInput	1	100	必須	再開可
	DVD_RECORDER	recording		100	必須	再開可
	VideoCamera	on		100	必須	再開可
	TV	on		100	通常	再開可
	TV	changeInput	2	100	通常	再開可
おかえりサービス	LIGHT	setBrightness	10	70	通常	再開不可
(Timer型)	CURTAIN	close		70	通常	再開不可
お出かけサービス	LIGHT	off		80	通常	再開不可
(Instant型)	CURTAIN	close		80	通常	再開不可
	DVD_REVORDER	off		80	通常	再開不可
	VideoCamera	off		80	通常	再開不可
	TV	off		80	通常	再開不可
	SPERKER	off		80	通常	再開不可

図 3　連携サービスモデル

また，複数の家電のメソッドの実行系列によって，連携サービスの**シナリオ**を定義する．図 3 の連携サービスモデルにおいて，一行一行がメソッドであり，機器への操作を表している．例えば，シアターサービスの実行系列を見ると，1. で述べたシナリオが機器操作の系列で表されていることがわかる．

さらに，サービス s, s' が**競合する**とは，以下の条件が成り立つことであると定義する：s のあるメソッド m と s_2 のあるメソッド m' が存在して，

- m の事後条件と m' の事後条件が両立しない．または，
- m の事前条件と m' の事後条件が両立しない．

1. で述べたシアターサービスとセキュリティ監視サービスの競合は，シアターの DVD_RECORDER.play() とセキュリティの DVD_RECORDER.recording() とが相容れない後条件を持つため生じる．

サービス競合を**解消**するひとつのアプローチとして，**サービス優先度**が知られている．各サービス s に予め優先度 $pri(s)$ を

与えておく．s と s' が競合した際に，もし $pri(s) > pri(s')$ ならば，s を優先して実行し，s' を中止またはキャンセルする．

2.2 オンラインサービス競合検出・解消手法

[5] において我々は，前述のサービス競合をサービス実行時にオンラインで検出，解消する手法を提案している．オンライン競合検出・解消とは，新規に実行する連携サービスと現在実行中の連携サービスを比べ，それらの間に競合が発生するか調べ（検出），可能であれば競合を回避する操作（解消）を行うことである．この実現にあたり，我々は**サービス期間**，**必須メソッド**，**サービスの中断・再開**という 3 つの概念を導入している．

2.2.1 サービス期間

サービス競合を実行時に厳密に検出するために，全ての連携サービスシナリオを，サービスの有効期間という観点から，以下の三種類に分類する．

BeginEnd **型：** ユーザーがサービスを開始したのち，明示的にサービスを終了するまで有効なサービス．

Timer **型：** ユーザーがサービスを開始したのち，一定時間後まで有効なサービス．

Instant **型：** ユーザーがサービスを開始した後，すぐに有効期間が切れるサービス．

BeginEnd 型，Timer 型は，サービスの有効期間中であれば，新しく実行される連携サービスとの競合検出の対象になる．Instant 型は有効期間がないため，新しく実行される連携サービスとの競合検出の対象とはならない．

図 3. サービスシナリオモデルにおいて，シアターサービスはユーザーが明示的にサービスを終了する BeginEnd 型サービスである．セキュリティ監視サービスは，サービスが開始されてから一定時間の録画を行う Timer 型サービスであり，おかえりサービスもユーザーが帰宅してから一定時間，帰宅時に最適な環境を提供する Timer 型サービスである．おでかけサービスは実行時に家電の電源を切りさえできればいいため，Instant 型サービスである．

2.2.2 必須メソッド/通常メソッド

連携サービスシナリオに含まれるメソッドのうち，そのメソッドを欠くとサービスの主目的が達成できなくなるメソッドを**必須メソッド**と定義し，それ以外のメソッドを**通常メソッド**と定義する．あるサービス s のメソッド m が**実行可能**であるとは，「m と競合するメソッド m_a が実行中でない，または，実行中でも m の優先度が高い $pri(m) > pri(m_a)$」と定義する．連携サービス Snew を新しく実行する際，Snew が**実行可能**であるとは「Snew の全ての必須メソッドが実行可能であるとき」と定義する．つまり，高優先度の競合サービス Sa がすでに実行中であり Snew に含まれる必須メソッドが一つでも実行できない場合には，Snew を実行しないことにする．また，実行中の連携サービス Sa に含まれる必須メソッドが，Snew の実行により実行不能になった場合，Sa 全体を終了することにする．一方，通常メソッドが競合した場合には，サービス機能を縮退して実行することができる．

図 3 のサービスシナリオモデルを用いて説明する．シアターサービスにとって，DVD プレイヤーを再生する操作であるメソッド DVD．play() は欠くことのできない操作である．シアターサービスを実行しようとするとき，より優先度の高いセキュリティ監視サービスが実行中であった場合，シアターサービスのメソッド DVD.play() はセキュリティ監視サービスのメソッド DVD.record() と競合し，DVD.record() が優先される．このとき，シアターサービスの他のメソッドが実行可能であったとしても，必須メソッド DVD.play() が実行不可能であるために，シアターサービス全体が実行されない．

シアターサービスを実行しようとするとき，おかえりサービスが実行中であった場合，競合するのはシアターサービスのメソッド LIGHT. setBrightness(1) である．これは照明を暗くするメソッドであり，欠けてもシアターサービスにとって大きな差し支えはないので，通常メソッドとなっている．この場合，シアターサービスは実行され，LIGHT.setBrightness(1) を除くシアターサービスのメソッドが実行される．

2.2.3 サービスの中断/再開

実行中であるサービス及びメソッドが他のサービスとの競合で終了された時，メソッドが再開可能なものであれば，終了はせずに中断状態となる．中断状態のメソッドは，再び自らが実行可能な環境になったとき，再開する．中断状態のメソッドは，実行中のメソッドと同様に競合検出の対象となるが，競合解消において，中断状態が継続する限り，他の実行中であるメソッドと新規実行されるメソッドに影響を与えない．

2.3 サービス競合検出・解消の流れ

サービス競合検出・解消の流れは以下のものとなる．

開始条件： 新規に連携サービス S_{new} が実行される．

Step 1: S_{new} と実行中サービスとの競合を検出する．

Step 2: S_{new} が実行可能であるか検証する．

Step 3: 実行中・中断中メソッドを更新する．

Step 4: S_{new} を実行する．

Step 5: 再開可能性のあるメソッドの再開処理を行う．

Step 6: 終了・中断するサービスの終了処理を行う．

Step1 では，連携サービスモデルを利用し，S_{new} の全メソッドと実行中サービスの全メソッドを比較，**競合する全てのメソッドのペア**を検出する．Step2 では，検出結果を用いて，S_{new} の全必須メソッドが実行可能かどうか検査する．実行不能な場合，S_{new} をキャンセル，実行可能な場合は Step3 に進む．Step3 では，S_{new} の実行可能メソッドおよび S_{new} の実行により中断・終了させられる実行中メソッドを調べ，サービスの状態を変更する．Step4 では，S_{new} の実行可能なメソッドを実行し，各機器の制御を行うと共に，S_{new} をサービス期間の型に応じて，実行中として管理する．Step5 では，競合解消によるメソッドの終了に伴って生じた再開可能性のあるメソッドを，再開処理にかける．Step6 では，終了処理が必要なサービスの機器制御を行う．終了処理とは，サービスが終了または中断した際に，機器環境をサービス実行前の状態に復帰させるようあらかじめ設定してあるメソッド群である．

3. オンラインサービス競合解消システム

2. の手法に基づきサービス競合解消システムを実装した．

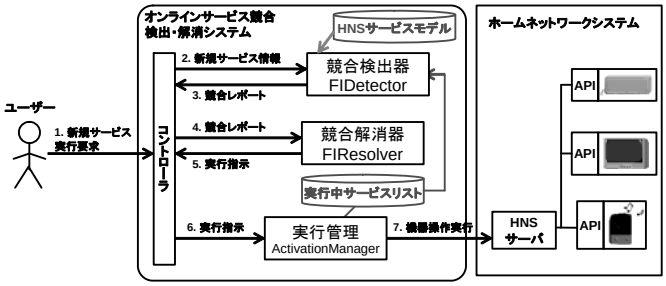

図 4　オンラインサービス競合解消システム

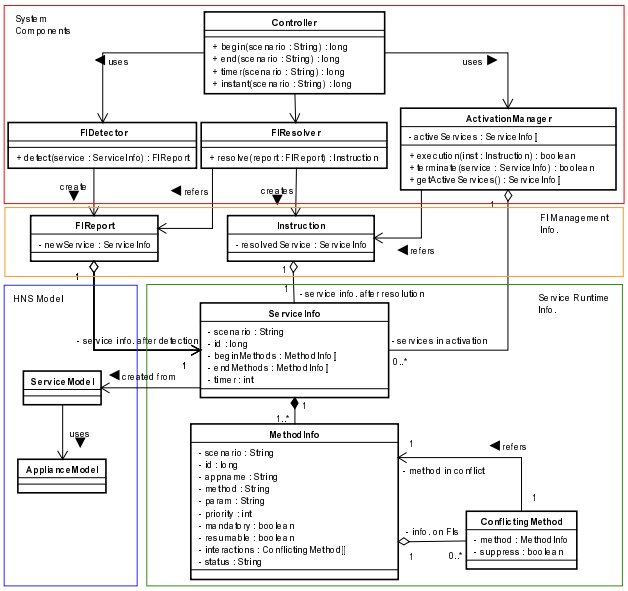

図 5　オンラインサービス競合解消システム　クラス図

3.1　アーキテクチャ

図 4 に示すとおり，オンラインサービス競合解消システムは，ユーザーと HNS との中間に位置し，サービス競合の検出・解消を図る．システムは，ユーザから HNS への連携サービス実行要求を途中で受け取り，サービス競合の検出を行う．競合がなければ，HNS に対して要求されたサービスを実行する．競合が発生すれば解消を行い，必要があればサービスの中断・再開等を行う．また，システムは全てのサービスのサービス期間を管理している．

実装したシステムは，**コントローラ**，**競合検出器**，**競合解消器**，**実行管理マネージャ**の 4 つのコンポーネントより構成される．以下の節でそれぞれの説明を行う．

3.2　コントローラ

コントローラは，ユーザーからの連携サービス起動要求，終了要求を受け付ける．サービスの起動については，サービス期間に応じた 3 つの起動インターフェース (begin(), timer(), instant()) を公開している (図 5 クラス図参照)．

ユーザーが各種連携サービスの実行を要求すると，コントローラは，連携サービスの定義の書かれた外部 XML ファイルより，連携サービスオブジェクトを表す ServiceInfo クラスのインスタンス S_{new} を生成する．生成された S_{new} は，サービスが実行すべきメソッドを表す MethodInfo のリストを保持している．コントローラは，S_{new} を競合検出器に渡し，S_{new} のメソッドと，競合する実行中および中断中の別サービスのメソッドとの間の競合を検出する．検出した競合の情報が追加された S_{new} が，検出結果としてコントローラに返る．次に，コントローラは，競合解消器に検出結果を渡し，メソッドの実行可能性を判定する．各メソッドの実行可能性の情報を追加された S_{new} が，解消結果としてコントローラに返る．最後に，コントローラは実行管理マネージャに解消結果を渡し，機器の実行を図る．

ユーザーから実行中又は中断中の連携サービスの終了要求を受けた場合，コントローラは実行管理マネージャにサービスの終了を伝達する．

3.3　競合検出器

競合検出器は，新規実行されるサービスと，実行中，中断中であるサービスとの間の競合を検出するコンポーネントである．図 5 において，FIDetector クラスが競合検出器にあたる．

競合検出器は，コントローラより新規実行されるサービスの ServiceInfo インスタンス S_{new} を受け取ると，実行管理マネージャから実行中および中断中である ServiceInfo リストを呼び出す．そして，S_{new} と，実行中および中断中である全ての ServiceInfo の競合を検出する．競合の検出には，2.1 にて述べた競合検出のアルゴリズムを用いる．検出に用いる前条件・後条件は，外部 XML ファイルが保持しているものを読み出す．新規実行される MethodInfo m と実行中および中断中である MethodInfo m' の間に競合が検出された場合，m の持つ競合リストに m' との競合情報を追加する．図 5 において，ConflictingMethod クラスが競合情報にあたる．競合検出器は，検出した競合の情報が追加された S_{new} を，競合検出結果を表す FIReport クラスに入れ，コントローラに返す．

3.4　競合解消器

競合解消器は，競合検出結果を受け取り，実行させたいメソッドの実行可能性を判断し，競合解消結果として返すコンポーネントである．図 5 において，FIResolver クラスが競合解消器にあたる．

コントローラより，競合検出結果 FIReport を受け取ると，競合解消器は S_{new} 中の MethodInfo m と，m の持つ競合情報内の競合相手メソッド MethodInfo m' のどちらが優先されるのかを，2.1 で述べたサービス優先度に基づいて判定する．判定結果は，m の持つ競合情報内に保存される．

次に，m が実行可能であるかどうかを判定する．実行可能性の判定には，2.2.2 で述べたアルゴリズムを用いる．m の持つ競合リストにおいて，”m より優先度が高い，且つ実行中である競合メソッド”が存在しない場合，m の status を”実行可能”に変更する．逆に，存在した場合は，status を”実行不可能”に変更する．また，必須メソッド m_{man} が”実行不可能”と判定された場合は，S_{new} に含まれる全ての MethodInfo に”実行不可能”であると情報を追加し，実行可能性の判定を終える．競合解消器は，実行可能性の情報が追加された S_{new} を，競合解消結果を表す Instruction クラスに入れ，コントローラに返す．

3.5 実行管理マネージャ

実行管理マネージャは，競合解消結果に従って，競合が完全に解消された機器操作要求を HNS に伝達するコンポーネントである．実行，または中断しているサービスのサービス期間を管理している．図 5 において，ActivationManager クラスが実行管理マネージャにあたる．

実行管理マネージャは，それまでに実行されており，サービス期間が終了していない連携サービスモデル ServiceInfo のリストを保持している．以下，このリストをサービスリストと呼ぶ．サービスリストに登録されている ServiceInfo に含まれる MethodInfo は，”実行中”か”中断中”かのどちらかの status を持つ．

ユーザーが新規サービスの実行を要求した際，実行管理マネージャはコントローラより，競合解消結果 Instruction を受け取る．実行管理マネージャはまず，S_{new} に，status が”実行可能”である MethodInfo m_{act} が含まれているかを調べる．含まれていなかった場合は，S_{new} が実行できないものと判断し，何もせずに終了する．含まれていた場合は，m_{act} が表す機器操作要求を HNS に伝達する．その後，m_{act} の競合リストに存在する競合相手メソッドの競合リストに，m_{act} との競合情報を追加する．また，S_{new} が BeginEnd 型か Timer 型であるならば，m_{act} の status を”実行中”にし，S_{new} をサービスリストに登録する．

次に，m_{act} が実行されることによって，終了/中断する MethodInfo m_{end} を抽出する．m_{act} の持つ競合リスト内で，m_{act} より優先度が低く，且つ status が”実行中”である MethodInfo が終了/中断されると判断する．m_{end} が再開可能なメソッドであれば，m_{end}のstatus を”中断中”に，再開不可能であれば，m_{end} を”終了”に変更する．このとき，m_{end} が必須メソッドであれば，同じサービスの他の MethodInfo も同様に終了・または中断させる．

終了/中断するメソッド m_{end} の持つ競合リストにおいて，m_{end} より優先度が低い MethodInfo m_{re} は，m_{end} が終了または中断することにより再開する可能性がある．そこで，終了/中断するメソッドの集合 $m_{end}[]$ から抽出した $m_{re}[]$ について，サービス優先度の高いものから順に再開できるか確かめる．m_{re} の持つ競合リストにおいて，”m_{re} より優先度が高く，且つ”実行中”である MethodInfo”が存在しなければ，m_{re} は再開可能といえる．よって，m_{re} の status を”実行中”に変更し，m_{re} が表す機器操作要求を HNS に伝達する．また，m_{act} を実行した際と同様に，m_{re} が終了させるメソッド m_{end} の状態を”中断中”，もしくは”終了”に変更し，それが必須メソッドであった場合，同じサービスの他の MethodInfo も同様に終了・中断させる．また，それらが終了することによって再開する可能性のある MethodInfo を，$m_{re}[]$ に追加する．すべての m_{re} について再開できるかを確かめ終えたならば，保持する MethodInfo の status が全て”終了”の ServiceInfo s_{end} について，サービスリストから ServiceInfo を除く．また，サービスリスト中の他の ServiceInfo より，s_{end} に関連した競合情報を消す．s_{end} に終了処理が設定されていれば，これを競合検出，解消にかけた上で実行する．

最後に，S_{new} が Timer 型であれば，一定時間後に実行管理マネージャに S_{new} の終了要求を出すスレッドを立てる．S_{new} が Instant 型であれば，S_{new} を終了要求を出す．

コントローラ，もしくは Timer サービスによって生成されたスレッドから，サービスの終了要求があった場合，実行管理マネージャは，終了要求のあったサービスの”実行中”である MethodInfo を m_{end} として，status を”終了”に変更する．そして，サービスの新規実行時と同様に，m_{end} が終了することによるメソッドの再開処理，終了処理を実行する．

3.6 実 装 環 境

オンラインサービス競合解消システムは以下の技術を用いて実装した．

- サーバー:950MB RAM 2.00GHz WinXP Pro
- Tomcat 5.5
- Apache Axis2
- Java JDK5

開発した競合検出・解消システムは，我々の研究グループで以前より構築している実際の HNS(CS27-HNS と呼ぶ) [1] に接続され，実サービスの競合検出・解消が可能となっている．

4. 評　　価

4.1 オンライン競合検出・解消実験

開発したシステムを用いて，図 3 に示す 4 つの連携サービスの任意のペア間のサービス競合検出・解消実験を行った．4 つのサービスは CS27-HNS 内の実機器を用いたサービスとして実装されている．任意のサービスペア s_1, s_2 に対し，これらを $s_1 \rightarrow s_2$ および $s_2 \rightarrow s_1$ の 2 通りの順序で実行し，サービス競合の検出・解消を行う．

表 1 にサービス競合の検出結果，および，システムが競合検出・解消，実行に要した実行時間を示す．検出された競合それぞれの解消結果は以下の通りである．

DVD シアター→セキュリティ監視： DVD レコーダにおいて，入力切替および録画 (セキュリティ)・再生 (DVD シアター) の競合が発生．セキュリティサービスの優先度が高いため，録画が優先される．DVD シアターは一時中断し，セキュリティ終了後再開された．

DVD シアター→おかえり： 照明において，明るくする要求 (おかえり)・暗くする要求 (DVD シアター) の競合，が発生．おかえりサービスの優先度が高いため，DVD シアターの暗くする要求は中断され，照明が明るくなる．おかえりが終了後，暗くする要求が再開し，照明が暗くなった．

DVD シアター→お出かけ： ．DVD レコーダにおいて電源を切る要求 (お出かけ)・電源を入れる要求 (DVD シアター) の競合，TV において電源を切る要求 (お出かけ)・電源を入れる要求 (DVD シアター) の競合，照明において照明を消す要求 (お出かけ)・暗くする要求 (DVD シアター) の競合がそれぞれ発生．お出かけサービスの優先度が高いため，お出かけが実行され，DVD シアターが中断するが，お出かけはすぐに終了し，DVD シアターが再開した．

表 1　検出された競合メソッドペア，及び，競合の検出/解消に要した時間

		検出された競合メソッドペア														
		競合検出時間			競合解消時間			実行管理時間								
		後から実行するサービス														
		DVDシアター			セキュリティ監視			おかえり			お出かけ					
先に実行しているサービス	DVDシアター				DVD_RECORDER.changeInput(0),DVD_RECORDER.changeInput(1) DVD_RECORDER.play(),DVD_RECORDER.recording()			LIGHT.setBrightness(1),LIGHT.setBrightness(10)			DVD_RECORDER.on(),DVD_RECORDER.off() TV.on(),TV.off() LIGHT.setBrightness(1),LIGHT.off()					
	実行時間				78[ms]	1[ms]以下	62[ms]	62[ms]	1[ms]以下	1[ms]以下	141[ms]	1[ms]以下	62[ms]			
	セキュリティ監視	DVD_RECORDER.changeInput(0),DVD_RECORDER.changeInput(1) DVD_RECORDER.play(),DVD_RECORDER.recording()									DVD_RECORDER.off(),DVD_RECORDER.on() VideoCamera.off(),VideoCamera.on() TV.off(),TV.on()					
	実行時間	109[ms]	1[ms]以下	1[ms]以下				47[ms]	1[ms]以下	1[ms]以下	125[ms]	1[ms]以下	16[ms]			
	おかえり	LIGHT.setBrightness(1),LIGHT.setBrightness(10)									LIGHT.setBrightness(10),LIGHT.off()					
	実行時間	78[ms]	1[ms]以下	1[ms]以下	47[ms]	1[ms]以下	1[ms]以下				94[ms]	1[ms]以下	15[ms]			
	おでかけ															
	実行時間	15[ms]	1[ms]以下	16[ms]	16[ms]	1[ms]以下	15[ms]	16[ms]	1[ms]以下	16[ms]						

セキュリティ監視→ DVD シアター： DVD レコーダにおいて，再生 (DVD シアター)・入力切替および録画 (セキュリティ) の競合が発生．セキュリティ監視の優先度が高いため，DVD シアターは実行されなかった．

セキュリティ監視→お出かけ： DVD レコーダ,TV, ビデオカメラにおいて，電源を切る要求 (お出かけ)・電源を入れる要求 (セキュリティ監視) の競合が発生．セキュリティサービスの優先度が高いため，お出かけサービスの競合のなかったメソッドだけが実行された..

おかえり→ DVD シアター： カーテンにおいて，閉じる要求 (DVD シアター) と開く要求 (おかえりサービス) の競合が発生．おかえりサービスの優先度が高いため，カーテンを開く要求を除いた DVD シアターが実行された．

おかえり→お出かけ： 照明において，照明を消す要求 (お出かけ)・照明を明るくする要求 (おかえり) の競合が発生．優先度の高いお出かけが実行された．

[5] に基づき，サービス期間，必須メソッドと通常メソッド，サービスの中断再開を考慮にいれた競合解消が行われていることが確認できた．

4.2 性能評価

表 1 の実行時間は，オンラインサービス競合解消システムの各コンポーネント (競合検出器，競合解消器，実行管理マネージャ) が処理に要した時間である．コンポーネント別に処理時間を見ると，実行中サービスが存在する場合の競合検出に比較的多くの時間が掛かっていることがわかる．これは，実行中サービスの全メソッドと新規実行するサービスの全メソッドの全ての組み合わせに対して競合検出を行うためであり，サービスが含むメソッドが多いほど多くの時間がかかる．また，サービスの新規実行に伴って実行中のサービスの中断（または終了)，中断中のサービスの再開が起こる場合の実行管理にも多少の時間がかかっていることがわかる．

およそどのサービス組み合わせにおいても，合計 100ms〜200ms でサービス競合の検出・解消・実行が行えている．実際のサービス実行が秒オーダーの実行時間を要する (例えば DVD シアターは約 40 秒) ことを考えると，十分実用に耐えうる短い時間で競合検出・解消処理が行えているといえる．

5. ま と め

本論文では，サービス期間を考慮したホームネットワークサービス競合検出解消手法 [5] をシステムとして実装し，実サービスに対して適用，評価を行った．その結果，開発したシステムは，予想した全てのサービス競合を検出し，意図したとおりの競合解消が実行された．また，性能面からも実際の HNS での運用に十分適用可能であることが確認できた．

今後，開発したシステムをより多くの実サービスに対して適用し，また，実ユーザによる評価も加えて，競合検出・解消手法の妥当性検証を行っていく予定である．また，他の競合解消方法を容易にシステムに組み込めるような改良を加えていきたい．

謝辞　この研究の一部は，科学技術研究費 (若手研究 B 18700062，20700027)，および，日本学術振興会日仏交流促進事業（SAKURA プログラム），パナソニック電工株式会社の助成を受けて行われている．

文　献

[1] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. ichi Matsumoto. Constructing home network systems and integrated services using legacy home appliances and web services. *International Journal of Web Services Research*, 5(1):82–98, January 2008.

[2] M. Nakamura, H. Igaki, and K. Matsumoto. Feature interactions in integrated services of networked home appliances -an object-oriented approach-. In *Proc. Int'l. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI'05)*, pages 236–251, 2005.

[3] M. Wilson, M. Kolberg, and E. H. Magill. Considering side effects in service interactions in home automation - an online approach. In *Proc. Int'l. Conf. on Feature Interactions in Software and Communication Systems (ICFI'07)*, pages 172–187, 2007.

[4] 吉村 悠平, 井垣 宏, 中村 匡秀,"ホームネットワークシステムにおけるサービス競合の動的検出・解消システムの設計と実装",IEICE Technical Report IN 2008-32(2008-7)

[5] 吉村 悠平, 井垣 宏, 中村 匡秀,"ホームネットワークシステムにおける家電連携サービスのための競合解消方式の考察", 情報ネットワーク研究会 (IN) '09