コールバックに基づく World Wide Web アーキテクチャの提案

池田直樹[†] 新城靖[†] 板野肯三[†]
佐藤聡[†] 中井央[†]

現在の World Wide Web アーキテクチャは，クライアント・サーバ・モデルに基づいており，利用者が操作する Web ブラウザがサービスを提供するサーバへ要求メッセージを送信している．このアーキテクチャでは，個人情報や継続的に変化する情報の扱いに問題がある．この論文は，これらの問題を解決するために，新しい World Wide Web アーキテクチャを提案している．新しい Web アーキテクチャでは，従来の呼び出し，すなわち，サービス受取者がサービス提供者へ要求メッセージを送信する方法だけでなく，それに加えてコールバック，すなわち，サービス提供者がサービス受取者に要求メッセージを送信する方法を用いる．新しい Web アーキテクチャでは，利用者は個人情報を Web フォームに繰り返し入力する必要がなくなる．サービス受取者は，コールバックされた時にサービス提供者の身元を証明書により確認することができる．サービス提供者は，個人情報をコンピュータに保持する必要がなくなる．Comet や WebSocket とは異なり，サービス提供者は，多数のサービス受取者を扱う時でも結合を維持する必要がなくなる．この新しいアーキテクチャに基づき，著者らは，サービス受取者で個人情報を管理する仕組みである Personal Information Safe (PIS) を実装している．この論文では，PIS を実装するためのプロトコルをコールバック関数のリストとして示している．PIS は，フレームワーク Ruby on Rails を用いて実装が進められている．

Proposal of a new architecture of the World Wide Web based on calling back

Naoki IKEDA,[†] Yasushi SHINJO,[†] Kozo ITANO,[†]
Akira SATO[†] and Hisashi NAKAI[†]

The current architecture of the World Wide Web is based on the client-server model, and web browsers operated by users send request messages to servers that provide services. This architecture inherently have problems about treatment of personal information and continuously chaining information. To address these problems, this paper proposes a new architecture of the World Wide Web. This new Web architecture uses not only regular calling, in which service recipients send request messages to service providers, but also calling back, in which service providers send request messages to service recipients. In the new architecture, users do not have to fill in repetitive Web forms with their personal information. Service recipients can verify the identities of service providers with providers' certificates. Service providers do not have to keep personal information in their computers. Unlike in Comet and WebSocket, service providers do not have to keep connections to service recipients. Based on this new architecture, the authors are implementing Personal Information Safe (PIS), which is a part of service recipients to manage personal information. This paper shows the protocol to realize PIS as a list of callback functions. The authors are implementing PIS under the framework Ruby on Rails.

1. はじめに

現在の World Wide Web は，クライアント・サーバ・モデルに基づき構築されている．図 1(a) のように，クライアント・サーバ・モデルでは，まず利用者が操作する Web ブラウザがクライアントとしてサービス提供者側でパッシブに動作しているサーバへ要求メッセージを送る．それに対してサーバが応答メッセージをクライアントに送る．この形態は，サービス提供者側に静的なコンテンツがあり，それを利用者が選択して表示するという目的に非常によく適合している．しかし，別の局面では次のような問題が生じる．

- 利用者は，様々なサイトで提供されているサービ

† 筑波大学
University of Tsukuba

スを受けるために住所氏名など個人情報を繰り返し入力する必要がある．
- サーバ側に蓄積された個人情報がクラッカの攻撃目標になる．
- フィッシング・サイトに個人情報を送信してしまう危険性がある．
- サーバ側にある刻々と変化するデータを自然な形で効率的に提供することができない．

これらの問題は，現在の World Wide Web がクライアント・サーバ・モデルに基づき構築されていることに起因している．本研究では，現在の World Wide Web のアーキテクチャを見直すことでこれらの問題を解決することを提案する．

この目的を達成するために，本研究では，サービス提供者側 (従来のサーバ) からサービス受取者側 (従来のクライアント) を呼び出す（コールバックする）という方法を従来のクライアント・サーバ・モデルに追加する (図 1(b))．コールバックという考え方自体は，ローカルプログラミングの世界では古くから存在し，現在でもよく利用される．たとえば，ウインドウ・システムのプログラミングでは利用者のボタン操作やキー操作をウィジェットが受け取るために使われている．また，コールバックにおけるサービス提供者側がサービス受取者側を呼び出す仕組みは，クライアント・サーバ・モデルにおいてサービス受取者側がサービス提供者側に要求メッセージを送る仕組みと似ている．そのため，現在の World Wide Web に大きな変更を加えることもなく導入できると見込まれる．本研究では，コールバックを自然な形で World Wide Web に導入することで上記の問題を解決することを目指す．この論文では，上記の問題の中で特に個人情報保護に関わる部分を深く論じる．個人情報は取り扱いに注意を要する情報である．個人情報の漏洩事件も繰り返し発生しており，銀行サイトを狙ったパスワードへの攻撃等に対する個人情報の保護は社会的に解決すべき重要な問題となっている．この論文では，本研究で提案するアーキテクチャにより個人情報をより安全に扱えるようになることを示す．

2. 関 連 研 究

従来の World Wide Web では，サーバ側にある刻々と変化するデータを利用者に届ける際に，定期的なポーリングがよく使われている．しかしながらこの方法はクライアント数が増えるとサーバの負荷が増大し，無駄なネットワーク帯域を消費することになる．この問題を解決するために，HTTP (HyperText Transfer Protocol) の枠組みを変更するものとして Reverse HTTP が提案された[5]．Reverse HTTP では，クライアントからサーバへ結合 (connection) を作成するまでは HTTP と同じであるが，その後，サーバとクライアントの役割を入れ替え，サーバ側から HTTP で要求メッセージを送信できる．Comet は，HTTP の枠組みを変更せずにこの問題を解決するために提案された技法の総称である[9]．Comet では，長期ポーリング (long polling) という手法が使われる．長期ポーリングでは，クライアントからサーバに要求メッセージを送信するが，サーバは応答メッセージをすぐには送信しない．サーバ側からクライアントへデータが変化したことを通知したい時に，サーバは応答メッセージを送信する．この応答メッセージの中に，要求的なコマンドを含ませることもできる．WebSocket は，Web ブラウザで動作する JavaScript のプログラムと Web サーバの間で Socket API (Application Program Interface) と類似の方法で通信をするための API，および，プロトコルを定めたものである[1)3)]．WebSocket は，サーバ側からクライアントへメッセージをプッシュするために利用することもできる．規格では，Web ブラウザ側はクライアントとしてしか動作しないので，サーバからの結合確立要求を受け付けることができない．

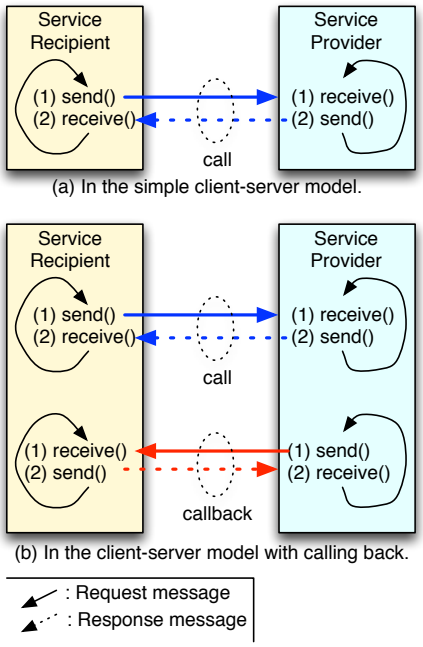

図 1 単純なクライアント・サーバ・モデルおよびコールバックを追加したクライアント・サーバ・モデル
Fig. 1 The simple client-server model and the client-server model with calling back

Reverse HTTP，Comet，および WebSocket の問題は，第 1 に，サーバ側ですぐには通信に用いない結合を維持するための資源を消費してしまうことである．結合が切れてしまうと，サーバ側からの通信は不可能になる．第 2 に，サーバが複数のプロセスから構成されていた場合，最初に接続したサーバとしか通信を行うことができないことである．そのため，最初のサーバが通信を中継しなければならないことがある．第 3 の問題は，データを必要とする主体が必要な時にアクティブに結合を作成するという自然なプログラムが記述できなくなっていることである．本研究では，これらの問題を解決する方法を提案する．

OperaUnite では Web ブラウザ Opera でサーバの機能を提供し，JavaScript で記述した Web アプリケーションを動作させることができる[11]．Web アプリケーションは，他の利用者が実行している Web ブラウザだけでなく，サービスを提供しているサーバからの要求にも応えることができる．本研究でも当初 OperaUnite を利用してコールバックを実装することを試みた．しかしながら，実装を進めているうちに，OperaUnite には，アクセスしてきたサーバを認証する機能やライブラリがないという問題があることが分かった．そこで，本研究では，OperaUnite を使っての実装は断念し，独自に実装することにした．また，OperaUnite は，必ず Web ブラウザと同じコンピュータで Web アプリケーションが動作する．本研究で提案するコールバックでは，ルータやデータセンタ等，Web ブラウザが動作しているコンピュータとは別のコンピュータに対してコールバックがなされることも許す．

クロスサイトスクリプティング攻撃への対策としては，サーバ側のプログラムを解析するものやクライアント側のパーソナル・ファイアウォールで検出するものなど，様々な手法が提案されている[8][4]．本研究では，サービス受取者とサービス提供者の両方で，コールバックという仕組みを導入している点が異なる．

Locker Project は，PC (Personal Computer)，デバイスおよびクラウド等に散らばった個人が所有しているデータを自分自身で管理可能にすることを目指している[6]．そして，そのようなデータを使ったアプリケーションを開発するためのフレームワークを提供している．個人が所有しているデータとしては，友人関係，フィットネスや健康に関するデータ，買物の記録等を想定している．ソーシャル・アプリケーションならば，自分のデータだけでなく他人のデータも許された範囲で取得できる．プロセス間の通信のために，TeleHash と呼ばれる，peer-to-peer 型の publish/subscribe モデルに基づくメッセージ交換プロトコルも開発している[7]．Locker Project は，現在初期段階であり，実用的なアプリケーションはまだ存在しない．

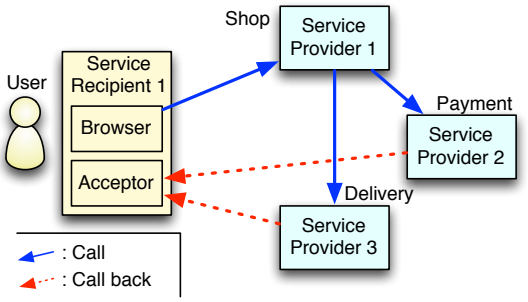

図 2 コールバックに基づく World Wide Web におけるサービス受取者とサービス提供者
Fig. 2 A service recipient and service providers in the World Wide Web based on calling back.

本研究で個人情報を所有者自身が管理しようとしている点は，Locker Project と共通している．Locker Project と比較して本研究の特徴は，従来の Web におけるサービス提供者とサービス受取者の関係を変えていない点にある．したがって，現在のオンラインでの買物等という適用範囲では，本研究で提案するアーキテクチャの適合性が高い．

3. World Wide Web へのコールバックの導入

従来の World Wide Web は，クライアント・サーバ・モデルに基づき，クライアントである Web ブラウザと Web サーバから構成される．クライアント・サーバ・モデルでは，まず，Web ブラウザがクライアントとして Web サーバに要求メッセージを送る．そして，Web サーバは，それに対してクライアントに応答メッセージを返す．

これに対して，コールバックに基づく World Wide Web では，次の要素から構成される (**図 2**)．

サービス受取者 (service recipients) 従来の Web における Web ブラウザの機能を持つ．それに加えて，サービス提供者から要求メッセージを受け付け，応答メッセージを返す機能を持つ．サービス受取者は，利用者の PC で実行される．

サービス提供者 (service providers) 従来の Web における Web サーバの機能を持つ．それに加えて，必要なタイミングでサービス受取者へ要求メッセージを送信し，応答メッセージを受け取る．サービス提供者は，サービスを提供する組織が用

意したコンピュータ（クラウドを含む）で実行される．

受取者代理人 (recipient agents) サービス受取者に成り代り，サービス提供者から要求メッセージを受け付け，応答メッセージを返す．詳しくは，3.3 節で述べる．

コールバックに基づく World Wide Web では，クライアント・サーバ・モデルに基づき次の 2 つのメッセージ送受信パタンが使われる．

コール 従来の Web と同様に，サービス受取者が要求メッセージを送信し，サービス提供者が応答メッセージを返す．

コールバック サービス提供者が要求メッセージを送信し，サービス受取者，または，受取者代理人が応答メッセージを返す．

サービス受取者は，次の 2 つのモジュールから構成される．

ブラウザ部 従来の Web における Web ブラウザと同じ．

受容部 (acceptor) コールバックの要求メッセージを受け取り，応答メッセージを返す．

サービス提供者は，単一のプロセスではなく，複数のプロセスの集合体からなることが一般的である．たとえば，図 2 のように買物を行う場合，商品を陳列する店 (Shop)，決済を行う会社 (Payment)，配達を行う業者 (Delivery) のプロセスが連携することになる．コールバックに基づく Web では，サービス提供者が複数のプロセスから構成された場合，それぞれのプロセスがコールバックを行う．サービス受取者は，コールバックがなされた時に，コールバック元を検査する．コールバック元が信頼でき，かつ，身に覚えがある時だけコールバックに応える．コールバック元が信頼できない時や，信頼できても身に覚えがないコールバックがなされた時には，応えない．また，サービス提供者が複数のプロセスから構成された場合，コールバック元に応じて提供する情報の範囲を変えることもできる．たとえば，買物を行う場合，クレジットカード番号は決済を行う会社のサービス提供者だけに，自宅住所は，配達を行う業者のサービス提供者だけに応答することもできる．

コールバックに基づく World Wide Web では，コールに用いるプロトコルとコールバックに用いるプロトコルを定義する必要がある．コールのプロトコルとしては，現在の HTTP ，および，HTTPS (HTTP over Secure Sockets Layer (SSL)/ Transport Layer Security (TLS)) をそのまま用いる．これにより，現在の Web ブラウザと Web サーバの主要部分を再利用することができる．コールバックのプロトコルとしては，HTTPS 上に遠隔手続き呼出し (Remote Procedure Call) の方法を定義して用いる．遠隔手続き呼出しの方法としは，現在広く使われているものとして SOAP，XML-RPC (XML Remote Procedure Call)，および，REST (REpresentational State Transfer) がある．これらの中で，本研究では，手続き呼出しとしては，REST を用いることにする．REST を用いた理由は，SOAP や XML-RPC よりも利用方法が簡単であり，ライブラリやツールを再利用することもできるからである．

サービス提供者から要求メッセージに従ってサービス受取者の受容部で実行される手続きを**コールバック関数**と呼ぶ．サービス受取者は，サービス提供者にブラウザ部を通じて次のような情報を渡す．

- 受容部の URL．外部から受容部をアクセスするために用いるホスト名とポート番号を含む．
- 受容部の SSL サーバ証明書．これにより，サービス提供者は間違いなく目的の受容部にアクセスできる．
- コールバック関数の一覧とそれらのインタフェース (名前，引数の型，結果の型など)．
- セッション管理，および，アクセス制御のためのトークン．詳しくは，3.2 節で述べる．

3.1 サービス提供者の認証とホワイトリスト

コールバックに基づく Web では，無闇にコールバックに応答しないようにコールバック元のサービス提供者が信頼できるものかどうかを判定する必要がある．

まず，サービス提供者が本物であるかどうかを確かめる必要がある．本研究では，サービス提供者が本物であるかどうかを確かめるため，そのサービス提供者が持つ SSL サーバ証明書を用いる．

次に，利用者は，正しい SSL サーバ証明書を提示したサービス提供者のうち，自分が信頼しているサービス提供者にだけコールバックを許すように設定する必要がある．コールバックの可否は，ACL (Access Control List) に記述する．この ACL は，利用者が信頼しているサービス提供者を格納したホワイトリストとして実装する．コールバック元のサービス提供者が複数のプロセスから構成されていた場合，正面に見えるサービス提供者だけでなく，後背のサービス提供者もホワイトリストに登録する．

ホワイトリストを維持するために，本研究では，サービス提供者の信用を調査して維持するような機関を設置して行うことにする．このような信用調査は，SSL

の EV 証明書 (Extended Validation Certificate) の発行と同等の手順で実現可能であると考えている[10]．あるいは，利用者が最初に正面のサービス提供者をアクセスする時に，自身のホワイトリストから使いたい後背のサービス提供者のリストを渡す方法も考えられる．たとえば，図 2 に示した買物の例では，サービス受取者が利用可能な決済会社と配送業者のサービス提供者のリストを店のサービス提供者に渡す方法が考えられる．店は，それらのリストの中から店自身が信頼しているものを選び，取引を行う．

3.2 トークンを用いたセッション管理とアクセス制御

コールバック元のサービス提供者が本物であってもそのコールバックが利用者の意図したものかどうかを判断する必要がある．これは，利用者になりすました攻撃者によるサービス提供者への要求メッセージにより発生した，利用者の身に覚えのないコールバックによる被害を防ぐためである．

本研究では，コールバックが利用者の意図したものかどうかを判断するために，トークンを利用する．トークンは，サービス受取者の受容部によって生成された乱数であり，**表 1** に示された情報と関連付けられている．トークンはサービス受取者のブラウザ部により受容部の URL，受容部の SSL サーバ証明書，および，利用可能なコールバック関数の一覧と共にサービス提供者に送信される．サービス提供者は，受容部にコールバックを行う際，トークンを渡す．受容部は，サービス提供者から受け取ったトークンの乱数をキーとして，表 1 に示した情報を取り出す．取り出せた場合，トークンは有効であり，それに関連付けられた情報に定められた有効期限，利用可能な回数および許可された手続きのリストに従ってコールバック関数を実行させる．取り出せなかった場合，トークンは無効であり，コールバック関数を実行させない．

トークンには，表 1 に示した情報の他に，セッションごとに独自の情報を結びつけて保存してもよい．

3.3 受取者代理人

サービス受取者は，利用者の PC で実行される．PC には，携帯電話等，従来の Web でブラウザが動作していたものを含む．利用者の PC は，電源を切っていたり，一時的に障害が発生している場合にコールバックに対応することができない．アプリケーションによっては，携帯電話のような非力なハードウェアやネットワークでは不十分なこともある．

表 1 トークンに関連付けられた受容部に保存される情報
Table 1 Pieces of information stored in a recipient agent with a token.

項目	説明
有効期限	トークンが使用できる期限
使用回数	トークンが使用できる回数
許可	実行可能なコールバック関数の並び

そこで，サービス受取者のうち，受容部だけを切り離して常に実行されているコンピュータで実行する．これを，本研究では，**受取者代理人 (recipient agent)** と呼ぶ．

受取者代理人を実行する場所については，次のものが考えられる (**図 3**)．

- 家庭用ルータ
- データセンタ

多くの家庭では利用者が PC の電源を切るなどしてオフラインになっていてもルータの電源は切らないことが多い．したがって，受取者代理人をルータで実行すると，受取者代理人がオフラインになることは少ないという利点がある．個人情報を狙う攻撃者の観点から見ると，家庭用ルータを用いる方法では攻撃対象が分散することになる．このことは，以下で述べるデータセンタを用いる方法と比較して，1 回の攻撃成功による被害を少なくできる．

データセンタで実行される受取者代理人は，個人情報やその他のプライベートな情報を預かる銀行のような存在になる．データセンタは，家庭用ルータよりもオフラインになる可能性が低く，常時コールバックにより必要な情報を取得する機会が得られるという利点がある．また，利用者の PC と比べて障害の発生する可能性も低いと思われる．ただし，利用者とは離れているので，非対話的な処理にしか利用することができない．セキュリティに関しては，現在のインターネット上の銀行サイトなどで使われている技術を応用することで十分な強度を得られると思われる．

データセンタで実行される受取者代理人は，PC 上で動作する受容部と同様に，ホワイトリストやトークンを用いてコールバック元が信頼でき，かつ，身に覚えがあるコールバックにだけ応答する．さらに，データセンタで動作していることを生かし，他の利用者の受容部へのアクセスパタンも使いながら総合的に攻撃に対処することもできる．たとえば，利用者 A の受容部にサービス提供者 1 から好ましくないコールバックがなされた時，受取者代理人は，サービス提供者 1 から 利用者 A の受容部へのコールバックを禁止するだけでなく，他の利用者の受容部へのコールバックを禁止することができる．このような働きは，実世界の銀行が個人の口座からの自動引き落としを許す先を審

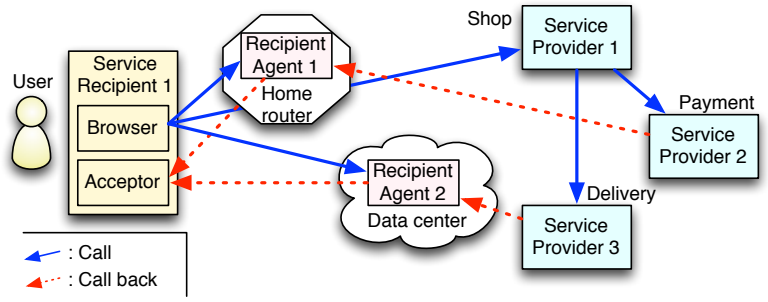

図 3 コールバックに基づく World Wide Web における受取者代理人
Fig. 3 Recipient agents in the in the World Wide Web based on calling back.

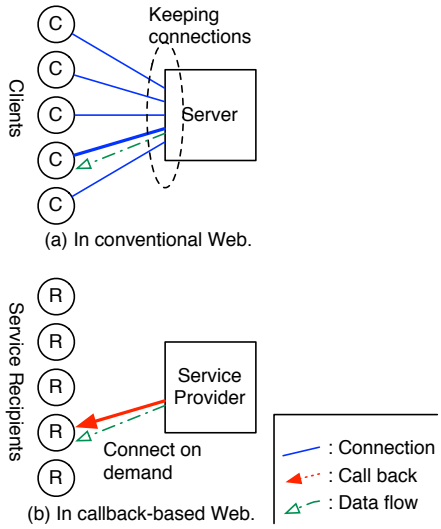

図 4 従来の Web とコールバックに基づく Web における結合の管理
Fig. 4 Managing connections in the conventional Web and the callback-based Web.

査することと似ている．

受取者代理人は，2 つのインタフェースを持つ．1 つは，サービス提供者からコールバックを受け付けるインタフェースである．もう 1 つは，利用者からコールバックで提供すべき情報を受け付けるインタフェースである．後者のインタフェースでは，受取者代理人は，利用者に分かりやすく保存している情報をブラウズさせる．また，コールバックのログを提示する．受取者代理人は，定期的に利用者の PC で動作している受容部をコールバックし，一貫性を維持する．

3.4 サービス提供者で変化する情報の扱い

1 章，および，2 章で述べたように，従来の Web にはサーバ側にある刻々と変化するデータを自然な形で効率的に提供することができないという問題がある．この節では，それらの問題とコールバックに基づく Web におけるその解決方法について述べる．個人情報に関する問題とその解決方法については，4 章で述べる．

(1) 結合を維持するための資源を消費する問題

2 章で述べたように，従来の Web では，Comet および WebSocket でサーバ側にある変化するデータを扱おうとすると，結合を維持するための資源を消費する．その様子を **図 4**(a) に示す．サーバは，クライアントの数だけ TCP/IP の結合を維持しなければならない．この問題は，クライアント数が 1 万 (10K) を超えた時に問題が顕在化する C10k 問題の原因の 1 つになっている[9]．

この問題は，コールバックに基づく Web では解消される．サービス提供者は，しばらく通信に利用しない結合を維持する必要はない．図 4(b) に示すように，ある特定のサービス受取者にメッセージを送信したい時に自らアクティブに結合を作成することができる．これを実現するプログラムも，自然な形で記述することができる．

(2) 最初に接続したサーバとしか通信を行うことができない問題

2 章で述べたように，従来の Web では，サーバが複数のプロセスから構成されていたとしても，クライアントは最初に接続したサーバとしか通信を行うことができない．たとえば，**図 5**(a) は，4 台のサーバから構成されている．クライアントは，Server 1 に接続している．Server 2 から 4 は，Server 1 を通じてデータを送っている．この Server 1 による中継が問題を引き起こすことがある．

この問題は，コールバックに基づく Web では解消される．コールバックに基づく Web では，後背のサービス提供者からサービス受取者に中継なしでデータを届けることができる．たとえば，従来の Web で図 5(a) のように中継が必要となる場合でも，コールバックに

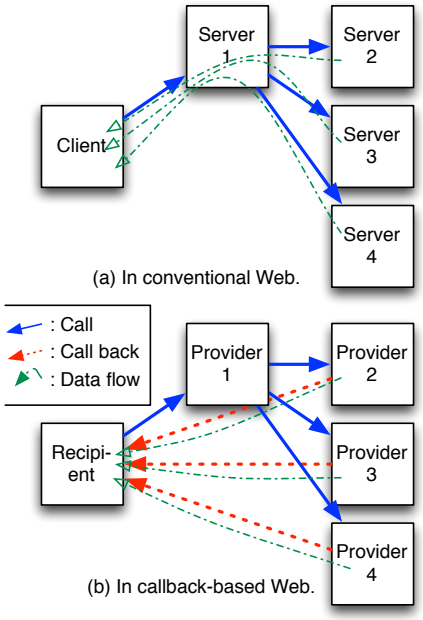

図 5 従来の Web とコールバックに基づく Web における協調サーバ間のデータフロー
Fig. 5 Data flows among collaborative servers in the conventional Web and the callback-based Web.

基づく Web では 図 5(b) のように直接結合を作成してデータを届けることができる．

3.5 既存の Comet，および，WebSocket の実装の改良

現在までに Comet および WebSocket に基づき多くのアプリケーションが開発されている．本研究では，そのようなアプリケーションをコールバックに基づいて書き直したいと考えている．これにより，3.4 節で述べた問題が解消され，自然な形でプログラムが書けるようになり，効率も良くなる．

3.6 コールバックのための結合確立要求を通過させる家庭用ルータ

家庭用ルータの内側 PC で受容部を動作させる場合，家庭用ルータは，外部からの結合確立要求を通過させる必要がある．現在普及している家庭用ルータの設定では，NAT (Network Address Translation) が使われ，そのような外部からの要求を遮断していることが多い．UPnP (Universal Plug and Play) 規格の 1 つ IGD プロトコル (Internet Gateway Device Protocol) を利用すれば，そのようなルータであっても外部からの結合確立要求を受け付けることができる[2]．UPnP IGD プロトコルに対応した家庭用ルータは市場に普及している．受容部は，この機能を利用してコールバックを受け付けることができる．

なお，3.3 節で述べたように，受容部を家庭用ルータ自身やデータセンタで走らせれば，UPnP IGD プロトコルに対応した家庭用ルータは不要である．

4. コールバックに基づく Web における個人情報保護の強化

この章では，個人情報に関して現在の World Wide Web の問題とコールバックに基づく Web におけるその解決方法について述べる．

コールバックに基づく Web では，サービス受取者，または，受取者代理人に受容部を設置する．個人情報を保存する受容部を，個人情報の金庫という意味で **Personal Information Safe** (以下 **PIS**) と呼ぶ．サービス提供者は必要なときに PIS にコールバックを行う．コールバックを受けた PIS はそのサーバが信頼できるサーバかどうかを判断する．信頼できるサーバであった場合は，個人情報をサービス提供者に取得させる．

4.1 従来の Web における個人情報に関する問題とそのコールバックによる解決

(1) 個人情報繰り返し入力の問題とその解決策

従来の Web では，利用者は，買物などをする際に個人情報をフォームに入力して送信しなければならないという問題がある☆．

コールバックに基づく Web を用いると，複数のサービス提供者が個人情報を必要としたとしても，各サーバが直接 PIS へコールバックを行うことで個人情報を取得しにくる．そのため，利用者による個人情報の入力は，PIS への入力の 1 回で済み，サービス提供者ごとに逐一個人情報をフォームで送信する必要がなくなる．名前やクレジットカードの有効期限を変更した時も，更新作業は 1 回で済む．

(2) サービス提供者での個人情報の中継と蓄積の問題とその解決策

従来の Web では，サービス提供者が複数のプロセスから構成されている場合，個人情報を必要としていないプロセスでも個人情報を中継する場合がある．また，利用者の利便性を上げるために個人情報をサービス提供者に蓄積することもある．このように個人情報を中継したり蓄積したりするコンピュータに脆弱なものがあった場合は，そこから個人情報が流出する危険性がある．このようなコンピュータの管理者には大きな負担がかかる．サービス提供者に個人情報を蓄積する場合，利用者にはログイン名とパスワードを管理す

☆ 個人情報をサーバに保存する場合には，(2) の問題が生じる．

ることが求められる．

このような問題は，コールバックを使って解決することができる．まず，サービス提供者が複数の連携しているプロセスから構成されている場合，個人情報が必要なプロセスが PIS に直接コールバックを行うため，個人情報を利用しないプロセスを経由することがなくなる．また，処理の途中で個人情報が必要になったらその度にコールバックにより取得することができるので，個人情報を蓄積する必要がなくなる．これにより，コンピュータの管理者の負荷が大きく軽減される．利用者もログイン名とパスワードを管理する手間から開放される．

コールバックを利用すると，サービス提供者のプロセスに脆弱性があり攻撃者に乗っ取られた場合でも被害を最小にすることができる．プロセスが乗っ取られた場合，攻撃者は個人情報を不正に取得できる．しかし，攻撃者が取得できるのは，乗っ取りを行っている最中に行われたトランザクションに関するものだけである．その他の利用者の個人情報が漏れることはない．

(3) 悪意のある外部サーバによる問題とその解決策

従来の Web におけるフィッシングでは，攻撃者が偽者の Web サイトを用意し，そこに利用者が個人情報を入力することを待つ．利用者は，URL やプロパティ等を参照し，Web サイトが本物であることを確認することで被害を防げる場合がある．しかし，利用者がそれらを目で確認することは負担が大きく，また，利用者が間違うこともあるため，フィッシングを完全に防ぐのは難しい．

コールバックに基づく Web では，利用者は直接 Web サイトに個人情報を入力することがなくなる．したがって，フィッシングにより表示された偽者の Web サイトに個人情報を入力することもなくなる．信頼できないサイトからコールバックがなされたとしても，3.1 節で述べたようにホワイトリストにより自動的に拒否できる．

サービス提供者の Web サイトに脆弱性があり，クロスサイトスクリプティング攻撃によりセッションが乗っ取られた場合，もともと利用者の受容部の URL とトークンが不正に攻撃者の手に渡ることがある．攻撃者がそれらの URL とトークンを利用して外部のサイトからコールバックを行っても，受容部はホワイトリストにより拒否できる．攻撃者が man-in-the-middle 攻撃によりコールバックを中継した場合も同様である．サービス提供者が直接受容部へコールバックを行った場合には，それは許可され，個人情報が返される．この個人情報は，サービス提供者のプログラムが利用してすぐに破棄すれば，Web ブラウザで動作している攻撃者のスクリプトには渡らない．

4.2 個人情報に関するコールバック関数

表 2 に PIS が提供するコールバック関数の主要部分を示す．サービス提供者はサービス受取者から与えられたコールバック関数の一覧からコールバック関数を選択し，実行させることで，個人情報を取得することができる．コールバック関数のうち，引数のあるものは引数で指定した言語に変換された個人情報を返す．サービス提供者は，決済や配達等の利用者の個人情報が必要な処理を行うときにいつでも，許された回数だけコールバックを行うことができ，その処理を終えたら個人情報を破棄することもできる．

PIS は，コールバック元のサービス提供者によって関数の実行の可否を変えることもできる．たとえば，買物においては配達を行う業者のサービス提供者が `getName` を実行した時には許可し，店のサービス提供者が実行した時には，拒否することもできる．

PIS は，コールバック元のサービス提供者によって関数の結果を変えることも考えられる．たとえば，買物において配達を行う業者のサービス提供者が `getAddress` を実行した時には詳細な住所を返し，店のサービス提供者が実行した時には，送料が計算できる程度の粗いものを返すこともできる．

4.3 Ruby on Rail を用いた PIS の実装

現在，Ruby on Rails(以下，RoR と呼ぶ) を利用して PIS の実装を進めている．RoR とは，Ruby で Web アプリケーションを記述できるフレームワークである．開

表 2 個人情報に関する主要なコールバック関数
Table 2 Primary callback functions related with personal information.

関数名	引数	返り値
`getEmail Address`	なし	メール アドレス
`getName`	言語を指定	名前
`getCountry`	言語を指定	国
`getState`	言語を指定	県，州
`getCity`	言語を指定	市
`getAddress`	言語を指定	番地
`getZipCode`	なし	住所
`getGender`	なし	性別
`getBirthday`	なし	誕生日
`getTelephone Number`	なし	電話番号
`getCredit Number`	なし	クレジットカードの 番号
`getCredit Expire`	なし	クレジットカードの 有効期限
`getCredit SecurityCode`	なし	クレジットカードの セキュリティコード

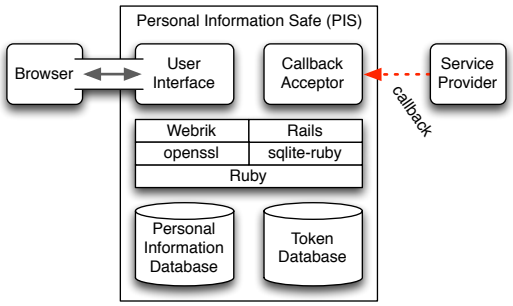

図 6 Ruby 言語による Personal Information Safe (PIS) の実装
Fig. 6 Implementation of Personal Information Safe (PIS) in the Ruby language.

発環境として，RoR 3.2.1，言語処理系 Ruby 1.9.3p0，ライブラリ Webrick，Openssl，および，Sqlite3-ruby を使用している．現在までに，Openssl を使い，サービス提供者からコールバックが行われた際は，SSL サーバ証明書を検証する部分は動作している．現在，トークンによるアクセス制御を実装している．

図 6 に実装している PIS の概要を示す．PIS は，2 つのインタフェースを持っている．1 つは，利用者インタフェース (User Interface) であり，利用者が個人情報を保護したり，ブラウザ部と協調してトークンを生成したりする．もう 1 つのインタフェース (Callback Acceptor) は，サービス提供者からのコールバック関数の呼出しを受け付ける．

表 3 PIS が保持する個人情報を保存するためのデータベースの属性
Table 3 Database attributes to store personal information in PIS.

属性	型	説明
name	string	利用者の名前
gender	string	利用者の性別
country	string	利用者の国
state	string	利用者の県，州
city	string	利用者の市
address	string	利用者の番地
birthday	date	利用者の誕生日
telephone_number	string	利用者の電話番号
credit_number	string	利用者のクレジットカードの番号
credit_expire	date	利用者のクレジットカードの有効期限
credit_security_code	string	利用者のクレジットカードセキュリティコード

表 4 PIS が保持するトークンに関連付けられた情報を保持するデータベースの属性
Table 4 Database attributes to token-related information in PIS.

属性	型	説明
rand	string	トークンの乱数
expire	date	トークンの有効期限
count	int	トークンの使用可能な回数
function_list	string	実行可能な関数

PIS は，内部に 2 つのデータベースを持っている．1 つは，個人情報を保持するデータベースである．その属性を**表 3** に示す．利用者はこのデータベースの項目について，任意の値を入力できる．1 人の利用者も，複数の個人情報を保存してサービス提供者ごとに切り替えて利用することもできる．このデータベースの user_id は，その切替えに用いる識別子である．

PIS が持っているもう 1 つのデータベースは，トークンに関連付けられた情報を保存するためのものである．その属性を**表 4** に示す．これは，表 1 に示したものを拡張したものになっている．rand はトークンの乱数を示す．外部のサーバからアクセスがあった際に，そのパラメータからトークンの乱数を取り出し，その乱数に一致する rand の値を持つデータに従ってアクセス制御を行う．expire には，トークンの有効期限を示す日付が格納される．count にはトークンが使用可能な回数が格納される．function_list には，実行可能なコールバック関数の名前の並びが格納される．

利用者インタフェースは，RoR のコントローラとして実装している．利用者インタフェースの主要な機能を以下に示す．

- 個人情報を保持するデータベースの更新
- トークンの生成とそれに関連付けられた情報の保存
- ホワイトリストの管理

コールバック関数の呼出しを受け付けるインタフェースも，RoR のコントローラとして実装している．サービス提供者からのコールバックは，次のような URL でなされる．

/token=*rand*&*func*1&*func*2&...

コールバックがなされた時，このコントローラは，次のような処理を行う．

(1) Openssl ライブラリより得られる情報からサービス提供者のホスト名を取り出し，ホワイトリストに含まれていることを確認する．
(2) RoR の要求オブジェクトからトークンの乱数を取り出す．それをキーとして，トークンのデータベースから関連付けられた情報を取り出す．
(3) トークンの有効期限や使用回数を確認する．
(4) 結果を保持するための空のハッシュ表を確保する．
(5) RoR の要求オブジェクトからコールバック関

数の名前を取り出し，それが許可されているかを確認する．
(6) 引数を取り出し，コールバック関数を実行する．
(7) コールバック関数の実行結果をハッシュ表に加える．
(8) (5) から (7) をコールバック関数の分だけ繰り返す．
(9) トークンの使用回数を減らす．
(10) 結果のハッシュ表をサービス提供者に返す．

5. おわりに

この論文では，コールバックに基づく World Wide Web アーキテクチャを提案した．既存の World Wide Web アーキテクチャは，クライアント・サーバ・モデルに基づき利用者が操作する Web ブラウザがサービスを提供するサーバへ要求メッセージを送信している．このアーキテクチャでは，個人情報や継続的に変化する情報の扱いに問題がある．これに対してこの論文で提案した Web アーキテクチャでは，従来のサービス受取者側からサービス提供者側を呼び出すことに加えてコールバック，すなわち，サービス提供者側からサービス受取者側を呼び出す仕組みを用いる．

新しい Web アーキテクチャでは，サービス提供者は必要な時にコールバックにより個人情報を取得できるので，利用者は個人情報を Web フォームに繰り返し入力する必要がなくなる．さらに，サービス提供者は個人情報をコンピュータに保持する必要がなくなる．サービス受取者は，コールバックされた時にサービス提供者の身元を証明書により確認することができるので，フィッシング攻撃への防御が強化される．Comet や WebSocket とは異なり，サービス提供者は，多数のサービス受取者を扱う時でも結合を維持する必要がなくなる．

現在我々はコールバックを活用して個人情報の保護を強化するための仕組みである PIS (Personal Information Safe) を実装している．この論文では，サービス提供者と PIS の通信プロトコルをコールバック関数のリストとして示した．また，フレームワーク Ruby on Rails を用いた PIS の実装についても述べた．

今後の課題は，PIS を完成させ，評価することである．また，Comet および WebSocket で生じる結合を維持するための資源を消費する問題を実際のシステムで解決したいと考えている．

参 考 文 献

1) Fette, I. and Melnikov, A.: The WebSocket Protocol, RFC 6455 (2011).
2) Forum, T. U.: Internet Gateway Device (IGD) V 2.0 (2010). http://upnp.org/specs/gw/igd2/.
3) Hickson, I (ed.): The WebSocket API, W3C Working Draft (2011). http://www.w3.org/TR/2011/WD-websockets-20110929/.
4) Kirda, E., Kruegel, C., Vigna, G. and Jovanovic, N.: Noxes: a client-side solution for mitigating cross-site scripting attacks, *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, ACM, pp.330–337 (2006).
5) Lentczner, M. and Preston, D.: ReverseHTTP, Internet Engineering Task Force Internet-Draft (2009). http://tools.ietf.org/html/draft-lentczner-rhttp-00.
6) Miller, J.: Locker Arch in a Nutshell (2011). http://blog.lockerproject.org/2011/06/16/locker-arch-in-a-nutshell/.
7) Miller, J.: TeleHash Draft Protocol Spec (2011). http://telehash.org/proto.html.
8) Minamide, Y.: Static Approximation of Dynamically Generated Web Pages, *Proceedings of the 14th international conference on World Wide Web*, WWW '05, ACM, pp. 432–441 (2005).
9) Russell, A.: Comet: Low Latency Data For Browsers, Internet blog (2006). http://alex.dojotoolkit.org/wp-content/LowLatencyData.pdf.
10) The CA / Browser Forum: Guidelines for the Issuance and Management of Extended Validation Certificates (2010). http://cabforum.org/Guidelines_v1_3.pdf.
11) Tømmerholt, H. S. and Davis, D.: Opera Unite developer's primer revisited, Dev.Opera (2009). http://dev.opera.com/articles/view/opera-unite-developer-primer-revisited/.