コンテキストアウェアな端末クラウド協調制御

西原 康介† 辻 聡† 狩野 秀一† 酒井 淳嗣†

開発コスト削減や開発期間短縮のため，複数のサービス機能単位を組み合わせてサービス提供できる技術が注目されている．近年携帯端末の高機能化により，従来サーバで動作していた上記のようなサービスの一部も，携帯端末上で十分動作するようになっている．このため，端末及びサーバの限られたリソース利用を最適化には，端末やサーバ上で動作する上記機能単位の構成を連携して制御する必要がある．さらに，ユーザ個別にサービスをカスタマイズするため，機能単位の構成をユーザコンテキストに応じて端末側で最適化する必要がある．そこで本稿では，コンテキスト状況に応じて，端末側でサービスやリソース利用を最適化し，低コストでサービスを構築できる仕組みを提案する．提案手法を評価するため，端末で撮影した画像を共有するサービスを構築することで，エンドユーザに高度なカスタマイズ性やユーザビリティ，システム設計者に低コストなサービス構築や容易なリソース最適化を実現する手法を提供できることを確認した．

Context-Aware Coordinated Control over Mobile Devices and Servers

KOSUKE NISHIHARA†, AKIRA TSUJI†,
SHUICHI KARINO† and JUNJI SAKAI†

We present a context-aware scheme to develop services which control resource usages of mobile devices and servers in a coordinated fashion. Increasing service development cost and turnaround time require a simple scheme to build services with various combinations of service components. Limited resource of mobile devices and servers needs to be controlled jointly because their recent mobiles can operate some of the highly-loaded components which have been done on server side. Additionally, personalized customization on the basis of user context is demanded due to diversified user needs. We propose an effective scheme for low-cost service development which can control combination of the service components adaptively to optimize resource usages. We develop a trial service which can share pictures taken by multiple devices for evaluation of our proposal. The development process and the trial indicate our proposal can offer customizability, low-cost development environment and easy resource optimization.

1. はじめに

クラウド環境ではサービスを 1 から構築するのは困難になってきており，サービス機能単位を組み合わせて 1 つのサービスを提供できる技術が注目されている[1]. 近年，ユーザニーズの多様化に伴う個別のカスタマイズや，他社優位に立つためのスピード感が求められている．又，ユーザサイドに立ったサービスは利用ユーザ数の少ないニッチなものも多く，開発にコストを掛けられない．このため，検証済みの機能単位を用い開発期間を短縮し，多様化する要求に応えることができるサービスをコストを掛けずに構築することが求められている．上記の機能単位を本稿では Service Component (SC)と呼ぶ.

また，近年携帯端末が高性能になり，前記 SC のようなサービスの一部が端末上でも動作できるようになってきた．携帯電話やタブレットタイプの携帯端末でも，マルチコアプロセッサや GPU を搭載し高性能になってきている. 特にタブレットタイプの端末では，携帯電話に比べ HW 面で余裕があり，PC のような高性能な機種も存在する．このため，上記のような日常的に使われる端末上でも，従来サーバ上で提供されていたサービスが十分動作するようになっている.

そのような携帯端末上で動作する SC(端末 SC)やサーバ上で動作する SC(サーバ SC)の接続構成を，リソース最適化のため端末側で連携制御することが必要である．限られた端末及びサーバのリソース利用を最適化するには, SC が利用する端末とサーバのリソースを連携して制御しなければならない．従来サービスやリソースを制御する機能はサーバ側に存在していたが，ネットワークが途切れたときにもシームレスなサービスを提供したり，利用者増加による制御サーバ負荷の影響をなくしたりするためには，サービス制御の機構は端末側にあるほうが望ましい.

さらに，その SC の構成をユーザコンテキストに応じて最適化することが必要である．端末に複数のセンサデバイスが搭載されるようになり，端末に登録されているユーザ情報や，電池の残量やネットワーク接続状態などの端末状態だけでなく，いつどこで誰が何をしているかなどのユーザ状況を認識することも容易になってきている．これらのユーザコンテキスト情報に応じた制御により，ユーザ個別の要求により応えることができる.

しかし，ユーザコンテキスト状況に応じて，端末側でサービスやリソース利用を最適化し，低コストでサービスを構築できる仕組みは無かった．既存の WEB を利用して 1 つのサービスとして見せる研究もあるが[2-4]，まだ一般的ではなくできることは限られている．又，WEB サービスや SNS サービスなどのサービス構築フレームワークはいつくも存在するが[5-8]，すべて Web 上で構成されることが前提で，端末リソー

† NEC システム IP コア研究所
System IP Core Research Laboratories, NEC Corporation

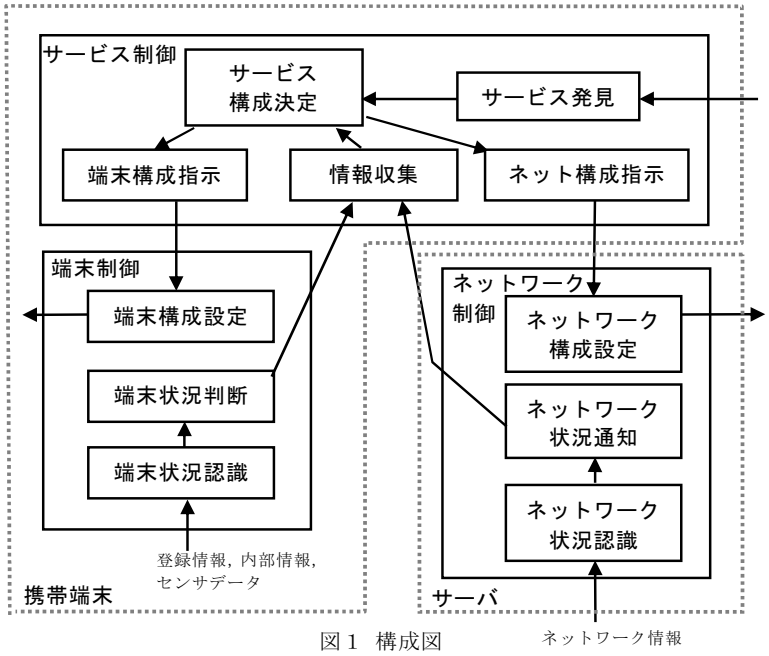

図 1　構成図
Figure 1 Block Diagram

スとサーバリソースの連携した最適化を行う手段は今まで存在しなかった．さらに，従来リソース制御はサーバ側で行っていたため，オフライン時にサービスが受けられなくなるだけでなく，被制御端末が多くなると制御が困難になっていた．コンテキストに応じたサービス構築フレームワークも提案されているが[9-11], 端末及びネットワークリソースの連携制御は考えられていない．

そこで本稿では、端末制御部，ネットワーク制御部，及び，サービス制御部から構成され，ユーザコンテキスト状況に応じて端末及びサーバのサービスやリソースを連携制御することで，上記課題を解決する仕組みを提案する．端末制御部は，ユーザコンテキストの認識，及び，端末 SC の構成やリソース利用の制御をする．ネットワーク制御部は，ネットワーク状況の認識，及び，ネットワークを制御することでサーバ SC 構成の制御をする．サービス制御部は，端末及びネットワークの状況に応じ，サービス構成や端末及びネットワークのリソース制御の決定をする．

上記の構成を持つことで，エンドユーザに高度なカスタマイズ性やユーザビリティを，システム設計者に低コストなサービス構築や容易なリソース最適化を実現する手法を提供することができる．ユーザコンテキスト情報やユーザ指示に応じて SC 構成を連携制御することで，高度なカスタマイズが可能になる．又，端末側でサービス制御を行うため，オフライン時のシームレスなサービス提供が実現できる．SC 単位でサービス構築することでサービス制御と SC の処理アルゴリズムが独立して開発でき，低コストなサービス構築環境を提供できる．さらには，端末及びサーバの連携制御によりリソースの最適化が容易になる．

2. 提案手法

ユーザコンテキスト状況に応じて端末及びサーバのサービスやリソースを連携制御する仕組みを説明する．本章では，始めに本手法の構成要素の説明をする．次にサービスを提供する場合に構成されるシステム，及び，SC を組み合わせたサービスを提供するときの制御処理フローの説明を行う．

2.1 提案手法構成

本手法は，端末制御部，ネットワーク制御部，及び，サービス制御部から構成される(図 1).

2.1.1 端末制御部

端末制御部は，ユーザコンテキストの認識，及び，端末 SC の構成やリソース利用の制御をする．端末制御部はさらに，端末状況認識部，端末状況判断，端末構成設定部から構成される．端末状況認識部が認識した情報を，端末状況判断部がサービス制御部に通知するか判断する．又，サービス制御部からの指示を受けた端末構成設定部が，端末上で動作する SC の構成を制御する．

端末状況認識部は，ユーザ情報やユーザ状況，端末状態などのユーザコンテキスト情報を認識する．ユーザ情報は端末に登録されているユーザの年齢，性別などの情報ある．ユーザ状況は，いつどこで誰が使っているかや，歩行，乗車などのユーザが何をしているかの情報であり，端末に備えられたセンサデバイスからのセンサデータを用いて認識する.端末状態は,電池の残量やネットワーク接続状態などの情報であり，端末システムへの問い合わせや端末システムからの通知により取得する．

端末状況判断部は，端末状況認識部で認識した情報をサービス制御部に通知するか判断し，判断条件とマッチした場合通知を行う．この判断は，予め作成したルール，又は，ユーザの実行時の指示に基づいて行う．ルールは，サービス提供者やシステムの設計者が予め用意するか，ユーザが新規に作成する．又，ユーザが実行時に既存のルールを修正することもできる．端末状況判断部の判断は端末状況認識部とは独立して設計される．このため，サービス提供者又はユーザは，同じ認識を行う精度が良い

別の認識部への置き換えや，新しい認識の追加が比較的自由にできる．この認識及び判断の仕組みは，我々が開発してきたコンテキストに応じて端末リソースを制御する，コンテキストアウェアリソース制御[12]の仕組みを利用している．

端末構成設定部は，SC 間で行われているプロセス間通信の接続先を静的又は動的に変更することで，端末 SC の接続構成を制御する．端末 SC は，端末上で動作するプロセスとして定義し，それらの間はプロセス間通信で接続した．各 SC は入力と出力を定義するだけでよく，各 SC 自身はどの SC から接続されているか，どこに接続しているかは関知しないようにデザインした．つまり，SC の設計は，SC 間の接続を気にせずに行うことができる．

SC の具体的な実装は Android の UI を持たないプロセスであるサービスが考えられ，その接続は AIDL が適当である[13,14]．AIDL での通信は，Bind という仕組みを用いた Android 特有のプロセス間通信である．aidl ファイルで入出力の定義を行い，Intent という接続先サービスの情報を持ったオブジェクトを Android システムに通知することで接続を確立する．aidl ファイルを公開することで，第三者が既存 SC と通信する SC を作成できる．

端末構成設定部は，接続元 SC が接続する際に Intent の接続先情報を書き換え，意図した SC へ接続させる．接続元 SC は，制御される前は最終 SC への接続に必要な情報を持った Intent やダミーの Intent を持っている．接続元 SC が起動されたときに，接続先情報を取得し接続を確立する．

2.1.2 ネットワーク制御部

ネットワーク制御部は，ネットワーク状況の認識，及び，ネットワークを制御することでサーバ SC 構成の制御をする．ネットワーク制御部はさらに，ネットワーク状況認識部，ネットワーク状況通知部，ネットワーク構成設定部から構成される．ネットワーク状況通知部がサービス制御部からの問い合わせに応じて，ネットワーク状況認識部がモニタした情報を返答する．又，サービス制御部からの指示を受けネットワーク構成設定部がサーバ上で動作する SC の構成を制御する．

ネットワーク状況認識部は，定期的にネットワークやサーバをモニタして情報を保持するか，問い合わせがあった時に情報を取得する．この情報を，サービス制御部の問い合わせに応じネットワーク状況通知部がサービス制御部に通知する．ネットワーク状況は，ネットワークの遅延や使用帯域情報，サーバの演算負荷情報などである．

ネットワーク構成設定部は，ネットワークの設定を変更することでサーバ SC の接続構成を制御する．サーバ SC は，サーバ上で動作するプロセスとして定義し，それらの間はソケットを用いた TCP/IP 通信で接続した．ネットワーク構成部は，ネットワークノードに対し，ポートの設定や IP アドレスの書き換えを行うための設定を行う．これによりパケットのフローが制御され，SC の接続構成が制御できる．端末 SC と同様に，各 SC 自身はどの SC から接続されているか，どこに接続しているかは関知しないようにデザインした．つまり，SC の設計は，サービス構成の制御を気にせずに行うことができる．SC のインタフェースは公開されることを前提としており，新しい SC はこのインタフェースにマッチするように作成される．現実装は，インタフェースがマッチしているもの同士の接続を想定しており，異なるインタフェースのもの同士のマッチングは今後の検討課題とする．

ネットワーク構成設定部の具体的な実装は，OpenFlow プロトコルでネットワークを制御する，OpenFlowController が考えられる．OpenFlow は，OpenFlow Switch Consortium により開発されているネットワーク制御技術である[15,16]．OpenFlowController は，OpenFlow で規定されるネットワークスイッチである OpenFlowSwitch のポートの設定や IP アドレスの書き換えを行いネットワークの制御を行う．

2.1.3 サービス制御部

サービス制御部は，端末及びネットワークの状況に応じ，サービス構成や端末/ネットワークのリソース制御の決定をする．サービス制御部はさらに，情報収集部，サービス発見部，サービス構成決定部，端末構成指示部，ネットワーク構成指示部から構成される．端末制御部及びネットワーク制御部からの情報を情報収集部が収集し，この情報を基にサービス構成決定部が，サービス発見部が発見した利用可能なサービスをどのように構成するか決定する．

サービス発見部は，利用可能な SC を保持しているデータベースに問い合わせるなどして発見する．このデータベースは，端末又はサーバ上に存在し，SC を端末又はサーバに導入する際に更新される．

情報収集部は，端末制御部やネットワーク制御部から通知があった時に情報を更新する．新規情報は，端末制御部からは新しくユーザコンテキストが認識されたときに通知される．ネットワーク制御部からは，情報収集部が定期的又は手動で問い合わせをした場合に通知される．

サービス構成決定部は，情報が更新された時に，予め作成したルール又はユーザの実行時の指示に応じてサービス構成の決定を行う．この判断も端末状況判断と同様に，サービス提供者やシステムの設計者が予め用意するか，ユーザが新規に作成する．又，ユーザが実行時に既存のルールを修正することもできる．

そして，各構成指示部は，サービス構成決定部の決定に基づき，各構成設定部に指示を出す．端末構成指示は，端末構成設定部へプロセス間通信の接続のフローを指示する．又，ネットワーク構成指示は，ネットワーク構成設定部へネットワークスイッチへの設定を指示する．

2.2 サービス制御処理フロー

サービスを提供するシステム構成を図２に示す．本システムは，携帯端末と各種サーバ，ネットワークスイッチから構成される．前記端末制御部及びサービス制御部は端

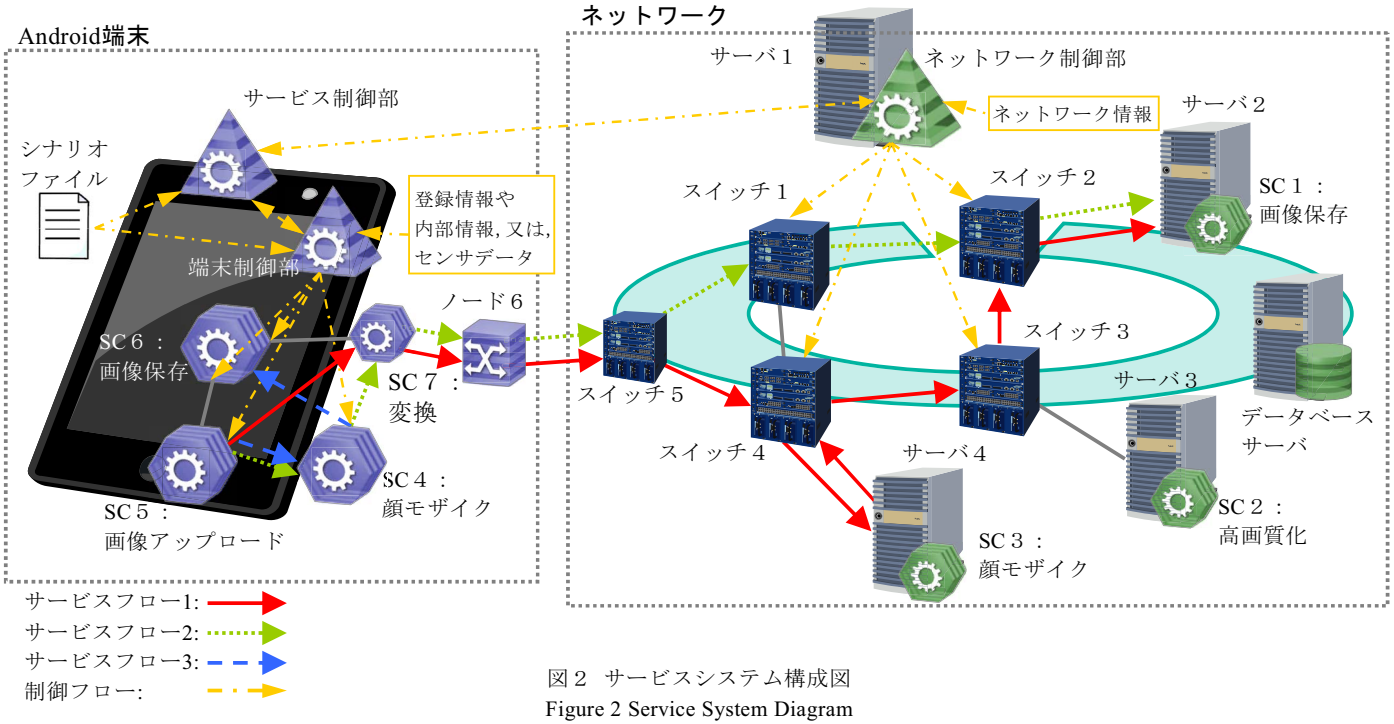

図 2　サービスシステム構成図
Figure 2 Service System Diagram

末上で動作し，ネットワーク制御部はネットワーク上のサーバで動作する．

本手法を用いたサービス制御の処理フローは，ユーザコンテキストを認識するコンテキスト認識フローと，サービスの構成を決定するサービス構成決定フローからなる．図 2 は，端末上の SC5 から始まりサーバ上の SC1 又は端末上の SC6 で終わるサービスの流れを示している．端末 SC は AIDL によるプロセス間通信を行い，サーバ SC は TCP/IP 通信を行う．これらの間は，プロセス間通信を TCP/IP 通信に変換する SC を通して行う．この変換 SC も Android サービスで実装し，他の端末 SC から AIDL で接続される．もちろん，1 つのサーバ上に複数のサーバ SC が動作しても良い．

まず，コンテキスト認識フローを説明する．端末状況認識部は登録情報や端末内部状態，入力されるセンサデータを用いてユーザコンテキストを認識する．ユーザコンテキストは，端末の登録情報や端末内部状態，入力されたセンサデータを解析して得られたユーザ状況などである．次に，端末状況判断部は，端末状況認識部で認識した結果を，予め作成したルール又はユーザの実行時の指示に応じて通知を行うか判断する．そして，ルールとマッチした場合やユーザ指示があった場合は，認識結果をサービス制御部に通知する．

次に，サービス構成決定フローを説明する．まず，情報収集部は，情報の更新を行う．情報は，端末状況判断部から通知があったときや，ネットワーク状況通知部から新しい情報が得られたときに更新される．情報が更新されると，サービス構成決定部はサービス構成の決定を行う．サービス構成決定部は，予め作成したルール又はユーザの実行時の指示に応じてサービス構成の決定を行う．利用可能な SC は，サービス発見部がデータベースに問い合わせて発見する．この決定に基づき，端末構成指示部は，端末構成設定部に端末構成設定の指示を出す．端末構成設定部は，端末構成指示

部からの指示に従い，端末 SC 間で行われているプロセス間通信の接続先を変更することで，SC の構成を設定する．同様に，ネットワーク構成指示部は，ネットワーク構成設定にネットワーク構成設定の指示を出す．ネットワーク構成設定部は，この指示に従い，ポート設定や IP アドレスの書き換えを行うための設定をネットワークノードに行う．

リソース利用の制御は，SC の実行場所を変更することで行う．同じ機能を持った SC が複数の異なるサーバや端末に存在し，サーバの演算負荷やネットワークの接続状況，端末の電池残量などに応じ，どの SC を利用するか選択することでリソースの利用を変更する．例えば，サーバ 1 上の SC1 とサーバ 2 上の SC2，端末上の SC3 が同じ機能を持っていたとする．サーバ 1 の負荷が大きくなった場合やサーバ 1 への経路に問題が生じた場合には，SC1 を SC2 に切り替える．また，端末の電池残量が少なくなった場合や圏外になった場合には，サーバ上の SC ではなく SC3 を使うなどである．

上記のように，端末やネットワークの情報を基に端末及びサーバの SC 構成を制御する構成をとることで、高度なカスタマイズ性や低コストなサービス構築，容易なリソース最適化を実現することができる．ユーザによる実行時の SC 選択に加え，ユーザコンテキストによる SC の自動選択で，エンドユーザに高度なカスタマイズ性を提供できる．SC 単位のサービス構築で開発コストが削減できるだけでなく，サービス制御と SC 処理が分離するため制御と独立した SC 開発が可能になる．さらには，端末やネットワークの状況に応じ SC の実行場所を変更することで，端末及びサーバの利用リソースの最適化が可能になる．

また，サービス制御部が端末上で動作する構成をとることで，オフライン時のシームレスなサービス提供や，サービス制御のスケーラビリティを実現できる．サービス制御部が端末上で動作するため，ネットワークが途切れたときでも端末内の SC でサービスを自動で再構成し，サービスを継続できる．又，サービスの制御をサーバで行うと接続端末数が増えた場合に処理負荷増加のためサービス構成決定処理が滞りサービス提供に問題が出るが，前記サービス制御部が端末上で動作するならばそのような問題は発生しないと期待できる．

3. 評価デモ

画像の共有サービスを提供するデモシステムを作成し，本提案手法の評価を行った．本サービスは，端末で撮影した画像をデータベースにアップロードするというものである．設定した場所でアップロードするときは，撮影画像に顔が映っているか判断し顔にモザイクをかけることができる．上記機能の提供のため，本サービスは，撮影した画像をアップロードする画像アップロード SC，顔を検出してモザイクをかける顔モザイク SC，入力画像をデータベースに保存する画像保存 SC から構成される．顔モザイク SC とデータ保存 SC は端末とサーバの両方に存在する．画像保存のためのデータベースサーバはネットワーク上に設置した．端末上のデータ保存 SC は，オフライン時には画像を一時的に保持するが，オンラインになるとデータベースに保存する．

サービス開発には，まず，各 SC 及び端末状態認識部を用意する．今回はすべて作成したが，既存の SC を流用することシステム開発が容易になる．端末状況認識部も同様に，独自開発に加え，既存の認識部流用することができる．本サービスに必要な認識は，GPS の位置データを一定距離ごとに取得するもの，電池残量及びネットワーク接続状況を Android システムの Intent のブロードキストの検出で認識するものである．これらの SC や認識は，サービスの制御や端末状況判断とは独立しており，サービス制御や判断に依存せずにアルゴリズム開発に注力できる．つまり，異なるサービスに適用する場合でも，基本的には修正することなく再利用することできる．

次に，端末状況判断及びサービス構成決定のルールを決める．端末状況の判断は，予めシナリオファイルに記載しておき，実行時に読み込むよう実装した．当該ファイルには，特定の位置に来た場合や，ネットワークの接続状態が変わった場合，電池残量が変わった場合に通知するように記述してある．サービス構成決定のルールも同じシナリオファイルに記載した．1) ネットワークに接続しているが電池残量が少ない場合には，顔モザイク及び画像保存のどちらの SC もサーバのものを利用する． 2) ネットワークに接続していて電池残量が十分な場合は，顔モザイク SC は端末のもの，画像保存 SC はサーバのものを利用する．一方，3) ネットワークに接続してないなら，どちらも端末のものを利用する．それぞれのサービス構成は，図２のフロー1，2，3 に対応する．上記のシナリオファイルはユーザが修正することができ，自分にあった制御が行える．

上記のサービスに新しい機能を追加するのも容易である．例えば，入力画像を高画質化する高画質化 SC とユーザの歩行状態を認識できる認識部が新しく利用できる場合を考える．この場合，新しい判断ルール及びサービス制御ルールを追加することで，歩いているときはアップロード画像に高画質化処理を適用するようなことが簡単にできる．このルールは，SC 導入時に自動で，又はユーザが手動で追加する．これらの追加する SC や認識は，アプリマーケットなどからダウンロードすることを想定している．

上記評価用サービスを実装し，評価実行を行った．図３に，実際にサービスを実装したマシン構成を示す．図２のサーバ 1〜4 は，マシン 1 内でバーチャルマシン(VM)として実装した．又，スイッチ 1〜4 はマシン 1 内で仮想 OpenFlow スイッチとして実装し，VM として実装した仮想サーバ 1〜4 やデータベースサーバ(マシン 2)，Android 端末間を接続する．VM を利用しても，実際のマシンと制御手法は変わらない．

実装及び評価実行を通し，以下の点を確認できた．i) 端末位置及びユーザの行動状態に応じ，ユーザが意図した時に撮影画像に顔モザイクや高画質化を施すことがきる．

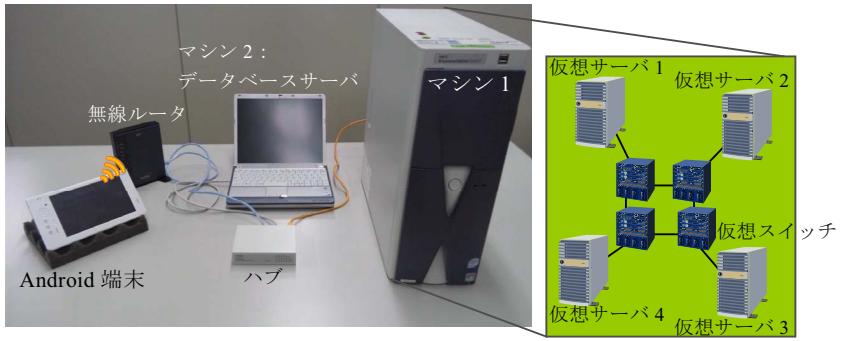

図 3　サービスを実装したマシン構成
Figure 3 Service-implemented Machines

ii) オフライン時にも端末内の SC を利用しシームレスなサービスが提供できる．iii) サービス制御に独立して各 SC が開発でき，サービス構築が容易にできる．iv) 電池残量やネットワーク接続状況で端末 SC かサーバ SC かを選択でき，少なくとも端末とサーバ間ではリソースの有効利用ができる．

4. おわりに

ユーザコンテキスト状況に応じて，端末側でサービスやリソース利用を最適化し，低コストでサービスを構築できる仕組みを開発した．端末やネットワークの情報を基に端末及びサーバの SC 構成を連携制御する構成をとることで、高度なカスタマイズ性や低コストなサービス構築，容易なリソース最適化を実現することができる．特に，サービス制御部が端末上で動作する構成をとることで，オフライン時のシームレスなサービス提供を実現できる．

評価サービス作成及び実行過程を通し，本手法の有効性を確認した．つまり，端末で認識した情報を基に SC の制御が出来ること，オフライン時でもサービスが継続できること，サービス制御と独立した SC 開発環境によりサービス構築が容易になること，端末状況に応じた端末及びサーバ連携制御でリソースの有効利用ができることである．

今後は，複数の携帯端末から制御要請があった場合のスケーラビリティの評価や，複数端末及びサーバのリソース利用を最適化する手法を検討したい．本稿では，ユーザ端末は 1 つであることを想定していた．通常のサービスは複数のユーザから利用され，相反する要求が生じる可能性もある．複数の端末がサービスを利用するときにリソース利用を最適化し，ユーザにより良い環境を提供できる方法を検討していく．

参考文献

1) 山登庸次，中野雄介，須永宏 ，“BPEL エンジンを用いたユーザカスタマイズサービス合成・切換技術の研究開発,”電子情報通信学会論文誌 B, Vol.J91-B, No.11, pp.1428-1439.
2) 森雅生，中藤哲也，廣川佐千男, “マッシュアップの軽量実装のための提案,” 電子情報通信学会 第 18 回データ工学ワークショップ(DEWS2007), C7-152, 2007.03.
3) 池田宗平，草野直樹，長嶺貴一，鎌田十三, “要求駆動によるマッシュアップと AJAX Widget による効率的な閲覧,” コンピュータソフトウェア 26(3), 20-33, 2009-07-28
4). 坂本寛幸，井垣 宏，中村匡秀, “センササービスのマッシュアップを実現するサービス指向基盤の提案,” IEICE technical report. Information networks 109(276), 23-28, 2009-11-05.
5) Apache Software Foundation, “The Apache Struts Web Application Framework,” http://struts.apache.org/1.2.9/
6) Slim3 WEB page, http://sites.google.com/site/slim3appengine/
7) Ruby on Rails WEB page, http://rubyonrails.org/
8) OpenSocial Foundation, OpenSocial WEB page, http://www.opensocial.org/
9) Adrian K. Clear, “A situation-oriented framework for programming context-aware applications,” in *Proceedings of the 10th ACM international conference on Ubiquitous computing (Ubicomp '08) Adjunct Programs*, Sep. 2008, pp.87-91.
10). David Dearman and Khai N. Truong, “BlueTone: a framework for interacting with public displays using dual-tone multi-frequency through bluetooth,” in *Proceedings of the 11th ACM international conference on Ubiquitous computing (Ubicomp '09)*, Sep. 2009, pp.97-100.
11) Brian Y. Lim and Anind K. Dey, “Toolkit to support intelligibility in context-aware applications,” in *Proceedings of the 12th ACM international conference on Ubiquitous computing (Ubicomp '10)*, Sep. 2010, pp.13-22.
12) Kosuke Nishihara, Kazuhisa Ishizaka and Junji Sakai, “Power Saving in Mobile Devices Using Context-Aware Resource Control,” in *Proceedings of 2010 First International Conference on Networking and Computing (ICNC)*, Nov. 2010, pp.17-19.
13) Android Developer WEB page, “Services,” http://developer.android.com/guide/topics/fundamentals/services.html
14) Android Developer WEB page, “Android Interface Definition Language (AIDL),” http://developer.android.com/guide/developing/tools/aidl.html
15) OpenFlow Switch Consortium, “OpenFlow: Enabling Innovation in Campus Networks,” http://www.openflow.org/
16) OpenFlow Switch Consortium, “OpenFlow Switch Specification,” http://www.openflow.org/