Proposal and Implementation of Pseudo Push Using Network Subsystem and Task Execution for PC

Kazuaki Nimura†, Hidenobu Ito†, Yousuke Nakamura†, Akira Shiba†, and Nobutsugu Fujino†

†Fujitsu Laboratories Ltd.
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, Japan
{kazuaki.nimura, itou.hidenobu, nkmr, shiba.akira, fujino}@ jp.fujitsu.com

Abstract -For the progress of Cloud computing, receiving a service from the Cloud 24/7 will improve the experience of PCs (personal computers) like that of smartphones today. At the same time, an issue of electric power comes to attention widely and increasingly. By putting a PC in lower power mode more actively, it can increase the sleep mode timing and can improve the electrical issue. However, this should not bring any degradation of the usability; otherwise, it cannot be accustomed user to the policy. In this paper, we propose a system that can send a task from the Cloud and execute the task at a PC whenever it is necessary, even though the PC is in sleep mode. Therefore, it is easy to live with the sleep state of PC because it will open up the task automatically, if needed. Hence, we prototypes the system that use a web application as a task and a pseudo push as a notification. Then, we evaluate it in a real field. As a result, the proposed system is found to be capable of executing the task in a timely manner. In addition, we confirm that the pseudo push can maintain the communication using little additional power consumption.

Keywords: PC, Push, HTML5, Application cache, Power consumption.

1 INTRODUCTION

Like a smartphone, it is desirable that PC remains connected to the network and receive services at all times. To receive network services, the PC needs to be powered on even when a user does not use it, such as during a meeting or at nighttime [1]. To reduce the power consumption more actively, a tool is provided by PC companies, which can set the power policy easily and put the PC in sleep mode for a short period when there is no activity. However, actively putting the PC in sleep mode may have an impact on business performance because it cannot access network among them. In addition to put the PC in sleep mode more actively, any degradation of user experience is not desired.

In this paper, we propose an architecture that can receive and execute a task whenever needed. The architecture consists of four functions: task provision, task notification, task preparation, and task execution. Task provision and task notification are functions included in the Cloud side. Task notification is realized with pseudo push by introducing a subsystem that works even while the PC is in sleep mode. It can help in informing a user regarding the incoming task from the Cloud.

We then prototype the system as the basis of the usage that smartphone sends a task which consists of a photo viewer programmed as a Web application and photo data. Here, we focus on the Web application because the Cloud technology has been widely used recently and many applications are provided as Web applications, such as HTML5 (HyperText Markup Language version 5) [2]. By using HTML5 as an application, it can provide a highly functional web application that is comparable with the native application and can provide the functionality of a locally executable task. As the execution environment, we use the Google Chrome browser [3] and Chrome OS [4] because of its improved compatibility with HTML5.

Moreover, we evaluate it in real field environment. The result shows that the PC subsystem can deal with pseudo push, even when the PC host is in sleep mode, by using an incoming message. In addition, it can provide low-power downloading and execute the application without user access. Although we introduced an additional dedicated hardware to receive services, it proves that it is effective in power consumption.

The remainder of this paper is organized as follows. In section 2, we summarize related works. In section 3, we present our architecture. In section 4, we show the details on the prototype implementation. In section 5, we show the results from the system evaluation, and finally in section 6, we state our conclusions.

2 RELATED WORKS

In this section, we summarize the related works from the aspect of wake function as described in Table 1.

To wake through a LAN (Local Area Network), a magic packet needs to be issued as an awakener, like Local server. The controllable area is basically the local area.

- An architecture that can wake PC and continue network services for PC is described in [5]. This work copies the host system's properties to the subsystem in Sleep mode.

To wake through a WLAN (Wireless LAN), the PC needs to ask a proxy or an AP (Access Point) to issue a magic packet. The controllable area is basically limited in the local area.

- Apple Inc. has implemented proxying as a wake-on-demand feature on their wireless network-attached storage devices and computers [6].

To wake through a WWAN (Wireless Wide Area Network), an SMS (Short Message Service) gateway or a push gateway is used. It is needed to deal AT-command to receive the message.

- About notification service, Intel Anti-Theft uses SMS as the notification message [7].

- Android Cloud to Device Messaging Framework [8] for Android phones and tablets use persistent connection by TCP (Transmission Control Protocol) and wake through the push gateway.
- How to establish a WWAN connection from a host system even when the subsystem has a network connection is described in [9].

Table 1: Types of network and wake for PC

Network type	Controllable area	Awakener
LAN	Local	Local server
WLAN	Local	Independent proxy, Access Point
WWAN	Wide area	SMS gateway

As far as we know, no feasibility study has been made on PC wake using the push gateway. The concept is briefly explained in this paper; however, the main focus of our research is dealing with the tasks.

3 PROPOSED ARCHITECTURE

In this section, first we will explain the issue, and then show the requirement to solve the issue, in addition propose architecture to meet the requirements.

3.1 Problem

The problem we are focusing on in this study is there is no mechanism that can send a task whenever needed while keeping PC in a low-power state as long as possible. The term of "task" in this paper means small job such as registration to an event through a network.

3.2 Requirements

The requirements to solve the problem statement are as follows.

(i) To have a function that can accept and send a task from the Cloud whenever necessary.
This corresponds to the phrase of "send a task whenever needed" in the problem statement.

(ii) To have a function that can notify a task to PC regardless of the power state of the PC.
This corresponds to the phrase of "while keeping PC in a low-power state as long as possible" in the problem statement because if there is notifying capability, PC can be in sleep mode.

(iii) To have a function that can prepare the task without waking up the PC.
Usually a powered on PC consumes much power and if the task can be prepared without waking up the PC, it will contribute to reducing power consumption.
This requirement corresponds to the same statement described in (ii).

(iv) To have a function to execute a task without user interaction and show the task to user.
This corresponds to the same statement described in (i) because if a task does not execute in PC, the cloud cannot receive the acknowledgement of "send a task".

By complying with these, a task can be transferred to PC anytime and the problem can be solved.

3.3 Architecture

To meet each requirement, we propose an architecture that consists of following functions, which are also shown in Fig. 1.

(I) Task provision: To fulfill requirement (i), a function that can provision a task, which will be transmitted to the PC, is introduced. As described later, a requester registers a task to this function.

(II) Task notification: To fulfill requirement (ii), a function that can notify task existence by messaging even if the PC is in sleep mode is introduced. For this, a network connection needs to be available anytime. The adequate communication technology changes depending on the usage of PC. If a PC is required only local network connection, for example, because of the large size, a communication technology that covers the local area can be used. On the other hand, if we expect mobility to PC, wide coverage of the communication needs to be hired.

(III) Task preparation: To fulfill requirement (iii), a function that can do background preparation even if the PC is in sleep mode to contribute to reduction in power consumption is introduced. There are two types of background service. One is a task that is needed to wake PC, and consumed the task by PC. Another is a task that is not needed to wake PC, and it can transact all in the background.

(IV) Task execution: To fulfill requirement (iv), a function that can provide and execute a task that is received even if the PC is in sleep mode is introduced. To translate the incoming task to an executable task takes an important role. Once PC turns on, a task is executed automatically or showed the existence to display.

By using those, PC can receive a task whenever from the Cloud. Therefore, even persons that are inexperienced in PC can receive a service easily. Then, it can contribute to improve the user experience of PC.

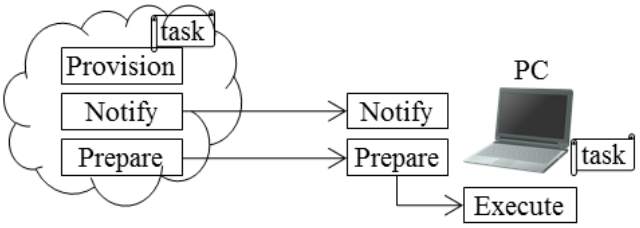


Figure 1: Proposed architecture.

4 PROTOTYPE IMPLEMENTATION

In this section, we will explain the prototype that realizes the proposed architecture and the task.

4.1 Prototype

Figure 2 shows the system structure of the prototype. The proposed system consists of the following hardware components: a requestor, task server, push gateway, host PC, and subsystem.

Requester:

There are two requesters. One is a requester that registers a task to the task server. Another is a requester that requests a push to the push gateway to send the task to the PC with the subsystem. It might be the requester is a corporate IT, for example. This component corresponds to the function (I) in the architecture.

Task server:

The task server carries out (I) task provision. It receives a task from the requestor, adds it to the repository, and exposes the repository included tasks to the Cloud.

Push gateway:

The push gateway carries out (II) task notification. It is designed that it can provide the service not only PC but also other devices such as smartphone and smart tablet. It consists of the following three elements that asynchronously work with each other.

- Push reception: A push message request is received from a pre-defined device such as smartphone, PC, or server.
- Push sender: It is responsible for sending a push message to the PC. It manages a persistent connection with the PC to deal pseudo push. If there is no connection between the push gateway and PC, for some reason, then it will resend the push message later.
- DB: It has information of PC identifications and requester identifications. When it receives a push request with a PC identification and requestor identification, the DB is checked whether the requester is preregistered and the PC identification is preregistered. If there are, it will serve.

Host PC:

The host PC cares the (IV) task execution, and it has the following elements:

- Browser synchronizer: It checks whether the web application or data exist in the SD memory and if there, correct through the memory. The timing of check is at the return from sleep or polling from the host system while it is in the power on state. A packaged application or data are reconstructed by this function. In addition, it injects the application or data to the browser cache. By doing this, the Web application or data can be used after the browser is initiated or reloaded.

Subsystem:

The PC subsystem carries out the counterpart of the (II) task notification and the (III) task preparation and it consists of the following elements:

- Push handler: It establishes a network connection with the push gateway. In addition, it receives a task notification. Besides, on, it analyzes an incoming push message. If it downloads or PC control requests, then the following functions are called.
- Downloader: If a push message has a download request, then it will download the Web application or data from the task server through the network to SD memory.
- PC control: If a push message has a PC control request, then it will issue a wake command to the PC with a condition. A condition might be nothing or wait until something happens. The example of a condition is whether the user is in front of the PC or not. If not, the request of wake will be pending, and if the user comes to the PC, then it will issue a wake to PC.

As mentioned above, these components can suffice the four pillars of the architecture. Therefore, by realizing the prototype, it can put the architecture into shape.

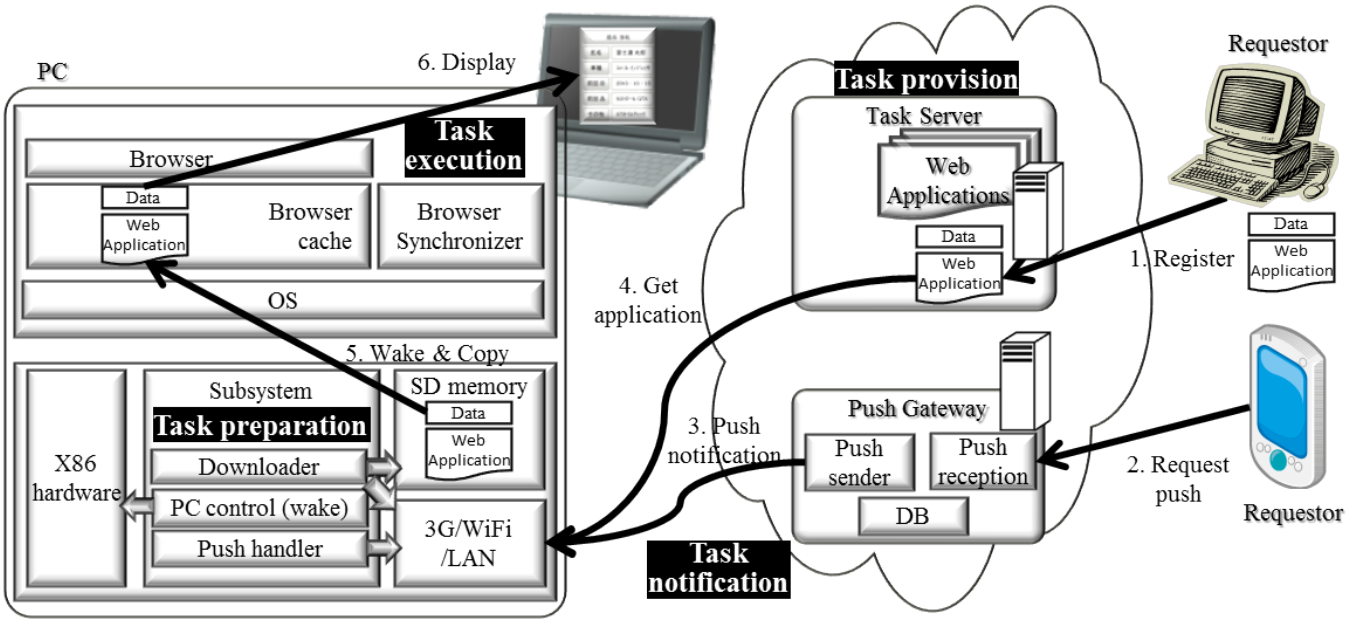


Figure 2: Prototype system.

4.2 Task and functions to deal task

To express a task, we use a HTML5 application and/or data. We will not exclude the local application though in accordance with the momentum of the Cloud computing, we focus on an HTML5 web application here. A HTML5-based Web application can be comparable with a native application because of its higher functionality than ever before. One of the curious features of HTML5 is the application cache. The application cache provides a capability to run the Web application in the local environment, i.e., even if the network is not available, you can still use the Web application. This is good for a person or corporate that needs to work in a restricted area, where the network connection is limited, for some reason, such as in a hospital. In this case, just before you get into that area, if you could receive the latest data from the Cloud service, you can use the application or data like a local application later.

Figure 3 shows an example of an HTML5 application. To use a locally accessible web application by HTML5, a list of resources needs to be specified below the line of "CACHE:" in a manifest file. By specifying it, you can use the application in a local situation.

When an application runs, basically, it first confirms the network connection at the runtime, and if there is a network connection, then it can act as conventional web application and access to the network as usual. However, if there is no network connection, it shall access to local application cache. Therefore, you can use the application anytime.

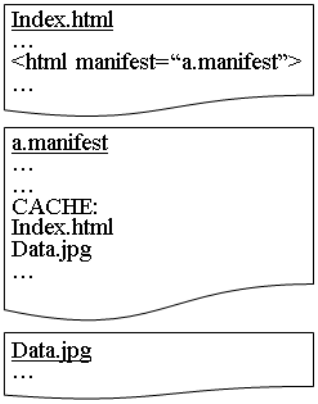


Figure 3: Web application with application cache

The detail of the four functions expressed in the architecture will be explained below.

Task provision:

Task provision provides a repository, which can store application and data for download.

Task notification:

Regarding the task notification for access to computing device anytime anywhere, there are two types of notification methods: polling and pushing.

- Polling: Polling is a method by which the client device accesses the server periodically. The disadvantage of polling is power consumption. An example of power consumption by a polling method has been shown in [10]. According to the explanation, when a TCP connection to the server is established, it consumes 5 to 8 mA. In addition, when the data are read, it consumes 115 mA, and when the data are written, it consumes 180 to200 mA. Then, a short polling consumes 0.5 mAh. If the information device polls every five minutes, it consumes 144 mAh. A typical battery for a smartphone gives 1500 mAh; therefore, about 10 % of the battery power will be consumed only by polling. Polling less frequently improves the efficiency, leading to less energy consumption. However, it loses the freshness.
- Push: Push is a way by which the server accesses the client when it wants to deliver some data. There are two types of push: true push and pseudo push.
  - True push: SMS or the type of messaging in Blackberry is categorized in this type [11]. SMS is a popular method for not only communicating between users but also initiating device management such as OMA DM (Open Mobile Alliance Device Management) [12]. OMA DM has a DM Server and a DM Client for the services. The DM Server can use a notification, and it is allowed to use an SMS-like message as a trigger to cause the DM Client to initiate a connection back to the DM Server. The problem of using SMS is that it can only accommodate a short size of information. For example, if you want to put a certificate in the SMS, it cannot accommodate that much information. E-mail notification is another method even though it has more than enough capability to control those computing devices.
  - Pseudo push: Pseudo push notification using persistent connection has been used to control smartphones or tablets. Basically, there is no limitation on the message size. Therefore, it can resolve the shortcomings of SMS.

We decided to use Pseudo push utilizing a TCP connection to notify a task. However, it has not been established for PCs yet. Then, we accommodated necessary capability needed for the client side in the subsystem, which works independently with the PC.

Task preparation:

Task preparation is a background function of introducing the subsystem, which means that it works independently with the PC.

Figure 4 shows the flow diagram for task preparation. First, the subsystem checks the attached network device and tries to connect through a valid network device. After it establishes a connection with the push gateway, it waits for the incoming push message. When a message comes, it analyzes the message whether it is valid message or not. If it is valid and the message is requested for download, then it will start downloading by accessing the given http address from the task server. If the message has a wake request, PC control function issues wake to the PC after the download is completed. It can be put an additional check before issuing a wake. For example, keep the wake request until the user approaches the PC, and if it detected by a sensor; it will issue wake to the PC.

Task execution:

Task execution works when PC is in on state. An added software function is the browser synchronizer. As shown in

Fig. 5, it checks the SD memory whether a valid application or data exist or not. If it exists, then it will reconstruct and inject the application or data to the browser cache and af terwards initiate the browser. Depending on the request in the push message, it initiates the browser. There are two types of view on the browser. One is displaying the application icons, and the user initiates an application implicitly. Another is running the application without user interaction.

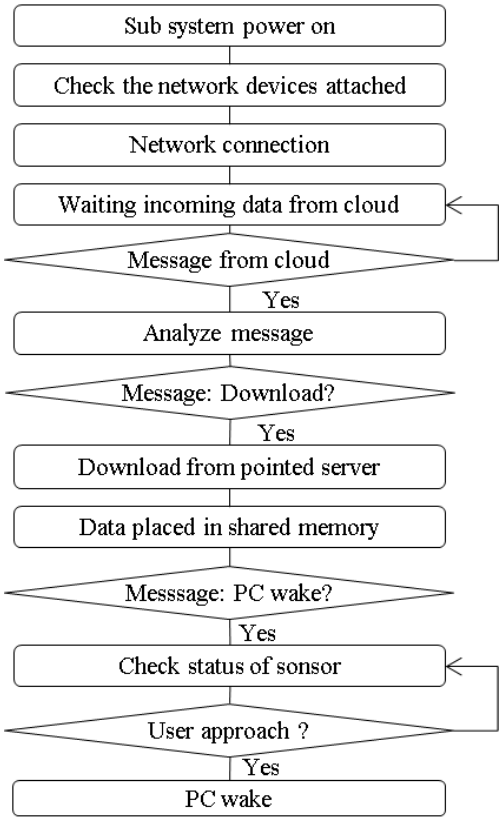


Figure 4: Flow of subsystem

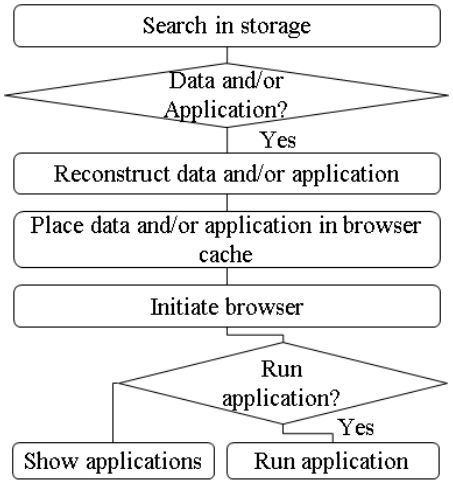


Figure 5: Flow of data collection and cache

5 SYSTEM EVALUATION

In this section, first, we show the hardware and software environments that are used for the evaluation, the result of the basic operation, and the measured power consumption.

5.1 Environment for System evaluation

Followings are the hardware and software configurations, which are used to evaluate the system.

Task Server:

- Hardware: A laptop computer is used as the task server and push gateway. It is a Fujitsu LIFEBOOK E780/A with an Intel Core i7 Processor 620M 2.66 GHz, 4 GB of main memory, and160-GByte hard drive.
- Software: A Web server is used as the task server. The public area of the server is used for placing an application and/or data. The requestor will place an application and/or data.

Push gateway:

- Hardware: One computer is used as the push gateway and task server. See above.
- Software: Linux is used as the OS. For the push reception, apache and Java Application Server (tomcat) are used. On the top of the Java Application Server, push reception is placed as a servlet. For the push sender, C-based program handles the communication with the PC. Between the two functions, those are communicating with a socket.

PC:

- Hardware: A laptop computer is used as the host system. It is a Fujitsu FMV-BIBLO NF/C80N, which had an Intel Core 2 Duo Processor P8600 2.4 GHz, 4 GB of main memory, and 80-GByte hard drive.
- Software: Three combinations are used as software environment as follows.
  - Microsoft Windows 7 as the OS and Google Chrome as the browser are used.
  - Linux as the OS and Google Chrome as the browser are used.
  - Google Chrome OS is used as the OS and the browser.

Chrome disk cache [13] is used as a place to put the Web application at all above.

Subsystem:

- Hardware: A Keil MCB2388 Evaluation Board is used as the subsystem. This has an ARM7 family-based processor. It also has a USB interface supporting USB devices and USB OTG/Host, SD/MMC memory card interface, and 10/100 Ethernet interface. The subsystem is connected to the host system through the USB device interface and to the network device through the USB host interface. In terms of electrical hardware specifications, the supply voltage of the board is 5.0 V and the typical current is 65 mA with the maximum current being 120 [mA] based on the specification sheet.
- Software: The board comes with RTX Real-Time Operating System, which allows programs that simultaneously perform multiple functions to be created. MDK-ARM Microcontroller Development Kit is a software-development environment that has a TCP Networking

Suite, USB device, USB host stacks, and other programming libraries.

Network devices:

- Hardware: Three network devices are used as follows.
  - A 3G communication device FOMA A2502 HIGH-SPEED is used as the network device. In terms of hardware specifications, the maximum downlink data rate is 7.2 Mbps and the maximum uplink data rate is 384 Kbps. It is bus powered through the USB port. The voltage is 5.0 V. The maximum current is 650 mA, the average current is 440.6 mA with the maximum standby current being 60 mA. The average standby current is 54.7 mA.
  - A LAN chip on the subsystem is used.
  - A WLAN device LANTRONIX WiPort, which supports IEEE802.11b/g, is used by connecting to the above LAN interface.
  -

5.2 Confirmation of basic action

We evaluated the system using the Google Chrome browser on Windows and Linux and Google Chrome OS. As a task, a Web application-based photo viewer and pictures are used. In the communication device, 3G, LAN, and WLAN devices are used.

Then, the following basic action is confirmed, as depicted in Fig. 6, based on the flow shown in Fig. 2.

Figure 6: Task send and execution.

(a) User takes picture using a smartphone. (b) User sends the data to the task server by e-mail. (c) The task server makes a package of data and an HTML5 photo viewer. In addition, the push gateway receives a push request from the phone, and pushes a message to the PC. (d) The subsystem receives the push message and downloads the application and data. Then, wake command is issued to the PC. (e) When the PC wakes, the application and data are injected to the application cache of the Google Chrome browser; finally, the web application and data are executed. At this moment, there is no network connection with host PC but subsystem. Therefore, it could confirm that the application is locally executed. Another simple basic action is confirmed as depicted in Fig. 7 and as follows.

(a') User put a URL which shows a link to a HTML5 game application of a tool on PC and the tool register the app to the task server. Then it asks a push to push gateway with a message for execute the application. (b') PC is in sleep. The subsystem receives the push message and downloads the application. (c') PC wakes. The application is unpackaged. Then it is injected to application cache of the browser. Finally, a user can play.

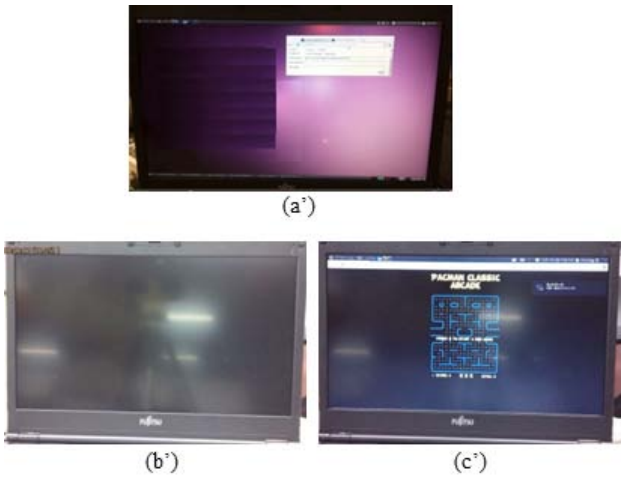


Figure 7: Task send and execution.

5.3 Measurement of power consumption

We will show the results of measurements of power consumption using the 3G module. First, we examine the persistent connection. Then, we compare the conventional PC and the PC in the proposed method. In addition, we will show the line of balance, which expresses how long the system should remain in sleep mode for reduced energy consumption.

How to put the PC in sleep mode is easy; the most used way might be timeout. It can be set in the operating system. If you specify a limitation time, and no activity has taken place at that time, the PC goes to sleep without any interaction. We have used this method to put the PC in sleep.

To express the power state, we refer the ACPI (Advanced Configuration & Power Interface) [14] that defines system and device power states. Some of them are as follows: S0 is in the system working state. S3 is a state in which the processors are not executing instructions and Dynamic RAM context is maintained. S4 is a state where the DRAM context is not maintained and all devices are in off state. S5 does not save any context. D0 is a state in which a device is in the operating state. D1 is an intermediate power state whose definition varies by the device. D3 is a state in which

the device is powered off. If we use the term of "sleep" in this paper, then it implies that the PC is mainly in S3 but not exclude S4 or S5. If we express network device in D1, then it means that the network is connected but there is no communication.

5.3.1. Persistent connection.

We discuss the power consumption of a persistent connection in this subsection. We use a TCP connection to notify a task. Therefore, the power consumption of the persistent connection is a curious factor in terms of feasibility. It is expected that by using an open TCP connection without any transmitting or receiving of data, the power consumption will be low based on the standard hardware capability.

First, we measured the current of the subsystem that is in D0 (On), as listed in Table 2. Somehow, it does not match with the specification described in 5.1, and the result of actual measurement is more than that of the specification. For reference, the voltage is taken as 5 V. The evaluation board used as the subsystem has an unwanted device for the evaluation, such as an LCD; therefore, it can be a small value if we use the embedded CPU alone.

Table 2: Subsystem

Subsystem status	Current [A]
D0 (On)	0.152

Then, we measured the current of the subsystem with the network device as shown in Fig. 8, and the average current values are listed in Table 3.

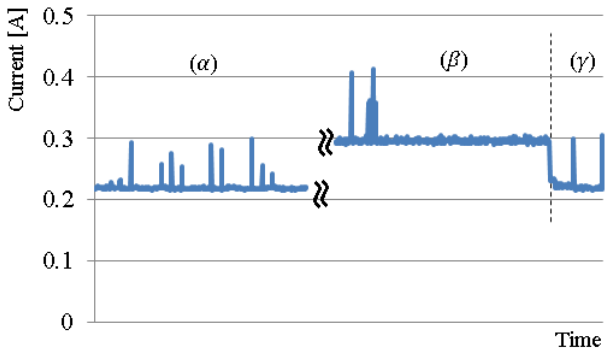


Figure 8: Measured current of subsystem with 3G.

The current changes until 60 sec and at 60 sec or later. As the device automatically transits to a low state when no explicit data are sent or received, the communication is maintained. The difference of current between the first row "plugged but not used for communication" and the third-row D1 in Table 3 is trivial. Even if the network is in communication ready, it does not consume a large amount of energy. This shows that a persistent connection can be fairly efficient in terms of power consumption. Comparing with the Table 2, difference between results given in Table 2 and the third-row D1 in Table 3 is 0.07 A, and it is found that the current is slightly high somehow when compared with the average standby current that is described in section 5.1. For reference, the voltage is taken as 5 V. This time, we use a product of a communication module for the PC, which started to sell in 2007. However, if we could use average type of communication device used in a cellular phone or M2M (Machine to Machine), the consumption will be around 1/8.

Table 3: Measured average current of subsystem with 3G

3G status	Subsystem status	Average current [A]
Plugged but not used for communication	D0	0.221…(α)
D0: Connected and idle (until 60 sec)	D0	0.296…(β)
D1: Connected and idle (60 sec or later)	D0	0.222…(γ)

5.3.2. Comparison of the conventional PC and proposed one.

We measured and compared the conventional PC and the PC used in the proposed method. Table 4 shows measurement result of average power consumption using the conventional PC with 3G. While the system is in S3, the absence of a network can considerably reduce the power consumption, which is well known.

Table 4: Without a subsystem: PC with 3G

PC status	3G status	Power [W]
S0: Idle	D0	26.45
S0: Http communication	D0	26.71
S3: Sleep w/o network	D3	2.030

Figure 9 shows the measurement results, and Table **5** lists the average power using the proposed PC with subsystem and 3G.

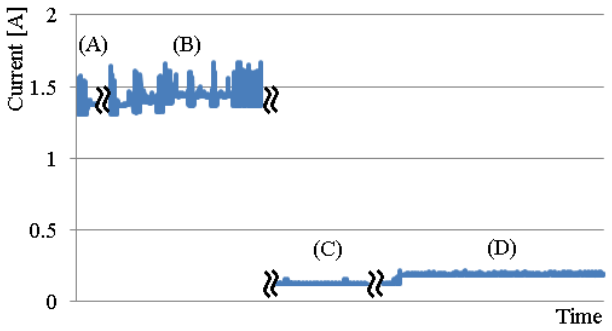


Figure 9: Measured current of proposed system.

The power consumption in S0 is almost the same. However, in S3, power consumption is slightly increased because of the communication capability. Figure 10 shows a border line derived using equation (10) in Appendix when T = 24 and based on the data given in Table 4 and Table 5. It shows whether the proposed system can provide the merit regarding the power consumption. The horizontal axis in the

figure indicates the utilization time of the conventional PC, and vertical axis shows the bifurcation point, which indicates whether the proposed system is paid off. The left-hand side of the dotted line shows the invalid area. If it could increase the sleep time by reducing the time to go to sleep, then it can be reached to the line. For example, suppose you to use 12 hours per day currently, reducing 0.85 hour is the point matches to the current power consumption. If we could use a power-efficient subsystem and network device as described in the previous subsection, the borderline will move to much easier portion.

Table 5: With a subsystem: PC with a subsystem and 3G

PC Status	Subsystem status	3G status	Power [W]
S0: Idle	D0	D0	25.8…(A)
S0: Http communication	D0	D0	26.8…(B)
S3: Sleep w/ network	D0: Idle	D0	2.41…(C)
	D0: Http receive		3.55…(D)

This is strict and may not be so adequate to evaluate the merit because a computing device that is not networked cannot receive any crucial and valid service. Therefore, it might be accepted without any concern by the consumer even when additional power consumption is involved.

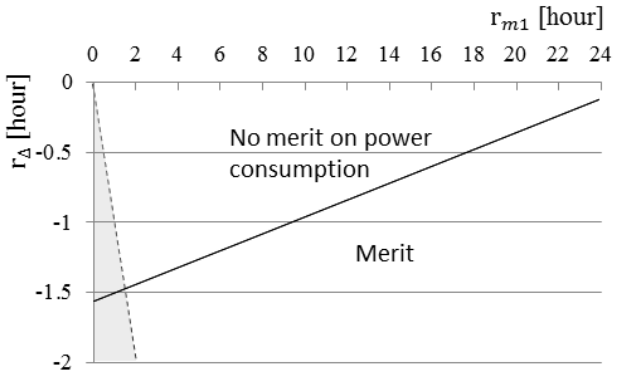


Figure 10: Line of balance of the proposed method.

6 CONCLUSION

We proposed an architecture where the system allows sending and executing of a task at any time in a PC. In our prototype implementation, we use a subsystem to keep network for dealing the pseudo push. In addition, we implemented a push gateway in the Cloud and a push handler and a browser synchronizer in PC.

The system could provide value that was not provided in current PC that:

- When a user wants to use the PC, the preparation is already completed.
- Task is presented to the user without user interaction.
- The power consumption can be reduced because during preparation, the PC can be in the sleep state.

In addition, we evaluated in real field using Google Chrome as the browser, Linux and Windows as the OS, and Google Chrome OS as the browser and the OS. For the network devices, we use LAN, WLAN, and 3G as the network devices.

Then, we confirmed that the system can send and execute tasks without user interaction even the PC is in sleep and pseudo push works efficiently in the power consumption point of view. From the measurements' data, the only small amount of additional power 0.001 [A] was required to deploy a connection for push.

With regard to the energy, as the subsystem is always powered, the lower the power consumption of the subsystem, more efficient is the system. The total power consumption per day would be comparable with conventional PC when if we could reduce the use of proposed PC 0.85 hour (suppose you to use 12 hours per day currently). However, the subsystem used in this study was not so efficient because of some glue logic. If we could use devices that eliminated wasteful logic or a new one, then the power consumption will be decreased.

In this study, we only focus on the PC. However, the same architecture can be used for embedded systems without any modification. In addition, the push gateway is designed to be an information device agnostic. Therefore, it can provide the push service to smartphone, smart tablet and other device as well.

As for the future research, remaining challenge might be how the system identifies the necessary task and the timing to push based on the activity of the users.

REFERENCES

[1] PC ENERGY REPORT 2009 UNITED STATES, UNITED KINGDOM, GERMANY, http://www.1e.com/energycampaign/downloads/PC_EnergyReport2009-US.pdf
[2] HTML5, http://dev.w3.org/html5/spec/Overview.html.
[3] Chromium an open-source browser project, sites.google.com/a/chromium.org/dev/Home.
[4] Chromium OS, http://www.chromium.org/chromium-os
[5] Yuvraj Agarwal, Steve Hodges, Ranveer Chandra, James Scott, Paramvir Bahl, and Rajesh Gupta, Somniloquy: Augmenting Network Interfaces to Reduce PC Energy.
[6] Apple Wake on demand, http://support.apple.com/kb/HT3774
[7] Introducing Intel Anti-Theft Technology version 3.0, http://antitheft.intel.com/Anti-Theft-30.aspx
[8] Android Cloud to Device Messaging Framework, code.google.com/android/c2dm/index.html.
[9] K. Nimura, H. Ito, Y. Nakamura, Z. Guo, K. Yasaki, and T. Miura, ``An Implementation of Transparent Network Subsystem for PC Manageability,'' RTCSA 2011, 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (2011).
[10] D. Ghosh, ``Building Push Applications for Android,'' Google I/O (2010).

[11] Simon Judge, What is Push?, http://www.mobilephonedevelopment.com/archives/832
[12] OMA Device Management Notification Initiated Session, www.openmobilealliance.com/Technical/release_program/docs/DM/V1_3-20101207-C/OMA-TS-DM_Notification-V1_3-20101207-C.pdf.
[13] Chromium disk cache, www.chromium.org/developers/design-documents/network-stack/disk-cache.
[14] Advanced Configuration & Power Interface, http://www.acpi.info/spec.htm

APPENDIX: FORMULATION OF POWER CONSUMPTION

In this section, we formulate the amount of electric power consumed by a conventional PC and the proposed PC, and then derive the equation of the line of balance. As described in Fig. 11, the proposed PC has an additional component called "subsystem," which is not present in the conventional PC. "Host system" is a main part of the PC. The subsystem affects the host system and the "network device." The network device is used for the communication. Each part consumes electric power.

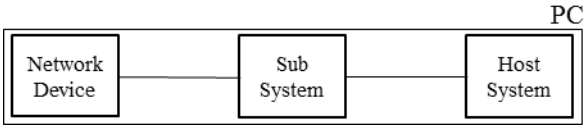


Figure 11: Proposed PC architecture.

Then, we formulate the power consumption as follows.

$$x_{on_{ave}} = \frac{\sum_{t=1}^{n_1}\left(v_c \cdot i_c(t) + v_h \cdot i_h(t)\right)}{n_1} \quad (1)$$

Here, $x_{on_{ave}}$ denotes the average electric power of a powered-on PC without a subsystem. n_1 denotes the number of measured points. t means the time at which the measurement is made. v_c denotes the voltage of the communication device. i_c denotes the measured current of the communication device at time t. v_h denotes the voltage of the host PC. i_h denotes the measured current of host PC at time t.

$$x_{slp_{ave}} = \frac{\sum_{t=1}^{n_2}\left(v_{h_{slp}}(t) \cdot i_{h_{slp}}(t)\right)}{n_2} \quad (2)$$

Where, $x_{slp_{ave}}$ denotes the average electric power of the sleeping PC without the subsystem. n_1 denotes the number of measured points. $v_{h_{slp}}$ denotes the voltage of the host PC in sleep mode. $i_{h_{slp}}$ denotes the measured current of host PC in sleep mode at time t.

$$r_{m1} + s_{m1} = \mathrm{T}\ [\mathrm{hours}] \quad (3)$$

Where, r_{m1} denotes the time of on state of the conventional PC in one day. s_{m1} denotes the time of sleep state in the one day of conventional PC. T denotes the total amount of time in a certain period, such as a day.

$$X_{conventional} = x_{on_{ave}} \times r_{m1} + x_{slp_{ave}} \times s_{m1} \quad (4)$$

$X_{conventional}$ denotes the total electric energy of the conventional PC at certain time T.

$$W_{on_{ave}} = \frac{\sum_{t=1}^{n_3}\left(v_c \cdot i_c(t) + v_s \cdot i_s(t) + v_h \cdot i_h(t)\right)}{n_3} \quad (5)$$

Where, $W_{on_{ave}}$ denotes the average electric power of a powered-on PC with a subsystem. v_s denotes the voltage of the subsystem. i_s denotes the measured current of subsystem at time t.

$$W_{net_{ave}} = \frac{\sum_{t=1}^{n_4}\left(v_c \cdot i_c(t) + v_s \cdot i_s(t)\right)}{n_4} \quad (6)$$

Where, $W_{net_{ave}}$ denotes the average electric power of the sleeping PC with a subsystem that is connected to the network.

$$r_{m2} + s_{m2} = \mathrm{T}\ [\mathrm{hours}] \quad (7)$$

Here, r_{m2} denotes the time of on state of the proposed PC in one day. s_{m2} denotes the time of sleep state of one day of proposed PC.

$$W_{proposed} = W_{on_{ave}} \times r_{m2} + W_{net_{ave}} \times s_{m2} \quad (8)$$

Where, $W_{proposed}$ denotes the total electric energy of the proposed PC in certain time T.

Obviously, adding the networked subsystem increases power consumption. Therefore, it is important to understand when it turns out to have merit. If we can put the PC in sleep mode for more time, then it can be paid.

$$r_\Delta = r_{m1} - r_{m2} = s_{m2} - s_{m1} \quad (9)$$

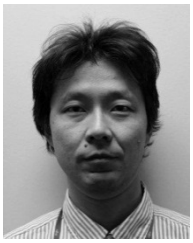
r_Δ denotes the time difference between how much spend the PC on at conventional PC and proposed method.

Consequently, the following formula can identify what time should the proposed system be kept in sleep mode than the conventional one if we wish to see the merit of the propose method.

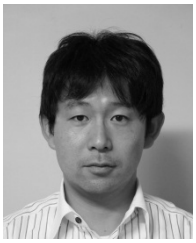
$$r_\Delta \leq \{x_{on_{ave}} \times r_{m1} + x_{slp_{ave}} \times (\mathrm{T} - r_{m1}) - W_{on_{ave}} \times r_{m1} - W_{net_{ave}}(T - r_{m1})\}/\{W_{on_{ave}} - W_{net_{ave}}\} \quad (10)$$

(Received February 27, 2012)
(Revised June 1, 2012)

Kazuaki Nimura received the BE and ME degrees in Graduate School of Information and Communication Engineering, Tokyo Denki University, Japan, in 1992 and 1994, respectively.
He joined Fujitsu Limited in 1994 and transferred to Fujitsu Laboratories Ltd, in 1997. His current research includes advanced technology of smart device and human centric computing.



Hidenobu Ito received the BE and ME degrees in Mathematical Sciences from University of Osaka Prefecture, Japan, in 1991 and 1993, respectively. He joined Fujitsu Laboratories Ltd in 1993. His current research includes mobile computing and human centric computing.



Yosuke Nakamura received the BE and ME degrees in Graduate School of Engineering, Yokohama National University, Japan, in 2000 and 2002, respectively. He joined Fujitsu Laboratories Ltd in 2002. His current research includes advanced technology of personal computer and human centric computing.

Akira Shiba received the B.S. and M.S. degrees in electronics engineering from Sophia University in 1980 and 1982, respectively. He joined Fujitsu Laboratories Ltd. in 1982. Since then he has been engaged medical electronics and mobile computing, and is currently a Research Manager of human centric computing technology.

Nobutsugu Fujino received the B.S. and M.E. degrees in electronics engineering from University of Osaka Prefecture in 1984 and 1986, respectively. He joined Fujitsu Laboratories Ltd. in 1986. Since then he has been engaged radio communication systems and mobile computing, and is currently a research manager of human centric computing and multi device interaction technology. His research interests include mobile and ubiquitous computing and network applications. He received IPSJ Industrial Achievement Award in 2003. He received Ph.D. degree in informatics from Shizuoka University in 2008.