December 1, 2008

RT0827
Mobile 10 pages

Research Report

Agile Content Delivery Scheduling for Large-scale Context-aware Services

Yasuharu Katsuno, Michiharu Kudo, and Takayuki Kushida

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Agile Content Delivery Scheduling for Large–scale Context–aware Services

Yasuharu Katsuno, Michiharu Kudo, and Takayuki Kushida
IBM Research Tokyo Research Laboratory
1623–14, Shimotsuruma, Yamato-shi, 242–8502, Japan.
+81–46–215-{4541, 4642, 4937}
{katsuno, kudo, kushida}@jp.ibm.com

Abstract

A context-aware service can recognize users' contexts (such as location, interests, or status) and deliver appropriate content prepared by content providers to appropriate users at the appropriate times according to each user's context. Many prototype systems for context-aware services have been proposed and they deliver content to users whenever the delivery conditions attached to the content are matched with a user's context. However, these systems do not work well as the number of pieces of delivered content increases, resulting in strong dissatisfaction by both the users and the content providers.

In this paper, we propose Agile Content Delivery Scheduling (ACDS) to balance both the content provider's needs and the user's, using a balanced distance-based action selection algorithm. We evaluated the performance of ACDS in a simulation and showed that ACDS is twice better than previous work under some conditions.

Keywords *Context-aware Service, Mobile Computing, Human Factors*

1 INTRODUCTION

1.1 Background

Recently, there has been considerable interest in services that can recognize users' contexts (such as location, interests, or status) and deliver appropriate content to appropriate users at the appropriate times [1]. We call this type of service a *context-aware service.*

Figure 1 shows an example of a context-aware service for use in an urban area. The content provider, a sports shop owner in New York, plans to deliver three types of sales advertisements for sales on May 13th and 14th: a standard advertisement, an advertisement with an attached discount coupon, and an advertisement with an attached special discount coupon. First, the owner categorizes the target customers into four context-based customer groups: $Customer_{red}$ for potential customers who live in New York and who have an interest in sports, $Customer_{blue}$ for the regular customers of the shop members, $Customer_{green}$ for customers who walk near the shop and who have an interest in sports, and $Customer_{purple}$ for the regular customers of the shop members who pass near the shop. Next, the owner registers the advertisements into a delivery system along with the intended delivery plans. Finally, the delivery system sends the advertisements based on the delivery plans.

- May 6th (a week before the sale):

  A regular advertisement is delivered to the $Customer_{red}$ group, and the advertisement with a discount coupon is sent to the $Customer_{blue}$ group.

- May 13th (first day of the sale):

  An advertisement with a discount coupon is delivered to the $Customer_{green}$ group, and an advertisement with a special discount coupon is sent to the $Customer_{purple}$ group.

- May 14th (final day of the sale):

  An advertisement with a special discount coupon is delivered to the $Customer_{purple}$ and the $Customer_{green}$ groups.

Many prototype systems which provides such context-aware services have been proposed for use in cities [2][3][4][5], university campuses [6], tourist sites [7], shops [8], offices [9], and hospitals [10]. Most of them deliver content to users whenever the delivery conditions asscoiated with the contents matches the user's context [11]. However, as the use of such content increases, naive delivery approaches can trigger large numbers of spam-like deliveres and potentially annoy or even anger potential customers [12]. A user only wants to receive reasonable amounts of actually valuable content for each user, because the number of contents that he/she is willing to see is limited [13][14]. Massive numbers of deliveries cause users to stop looking at the delivered content. In constract, a highly limited delivery approach delivers little content and makes content providers unsatisfied, because they want to deliver their content to many users who satisfy the delivery conditions so that many target users will see their content [13]. In general, it seems impossible to completely satisfy both the content provider's desires and the user's, because the desires of both sides are incompatible. A delivery approach is needed that balances both the content provider's desires and the user's.

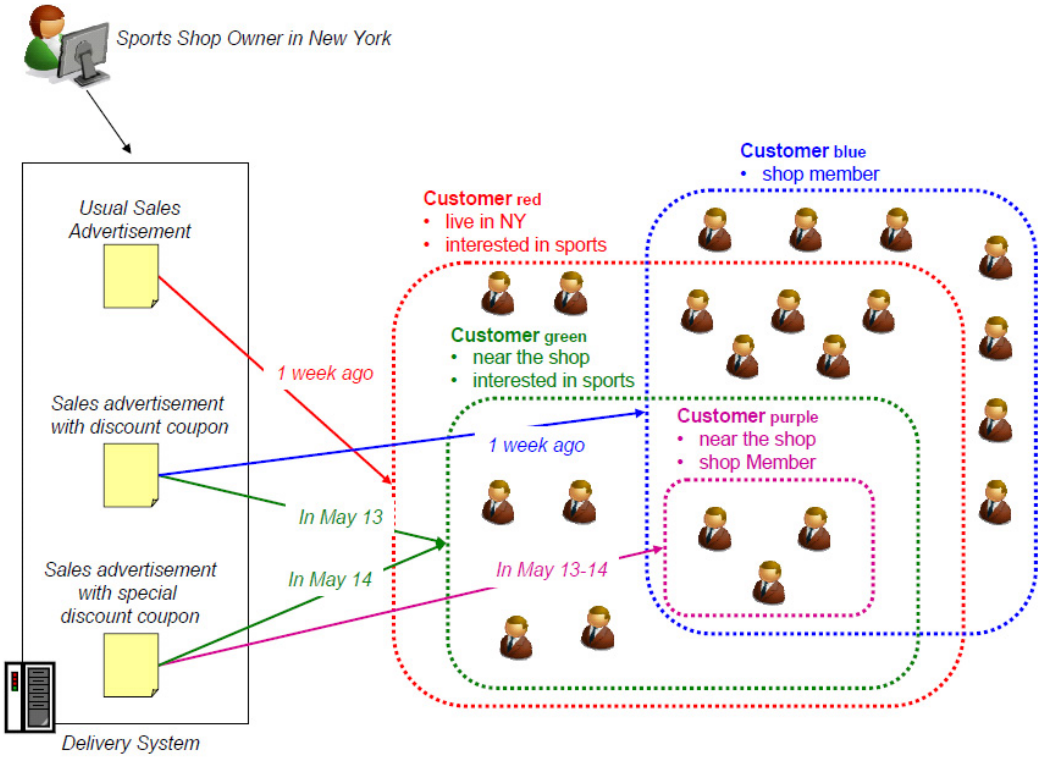

Figure 1: An Example of Context-aware Service

1.2 Our Contribution

In this paper, we propose Agile Content Delivery Scheduling (ACDS) for scalable context-aware services. ACDS runs on the delivery system and takes appropriate actions for the candidates to balance both the content provider's desires and the user's, by using a balanced distance-based action selection algorithm. We design ACDS, evaluate the performance against the present naive scheduling in a simulation, and show that our approach is effective.

This paper is structured as follows: We describe the design in Section 2. Section 3 evaluates our proposed scheduling in a simulation. Related work is discussed in Section 4. Finally, we conclude in Section 5 with a summary of our results and consider issues that still remain to be addressed.

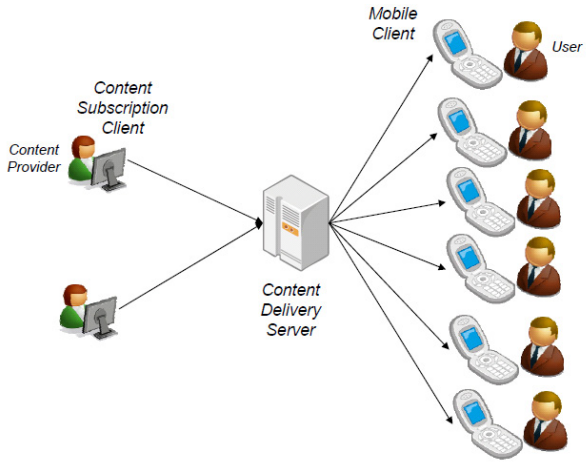

Figure 2: Context-aware Service System

2 DESIGN

2.1 Architecture

Figure 2 presents a our hypothetical system, a context-aware service system, based on [15]. It consists of two types of clients, a mobile client and a content subscription client, and one server, a content delivery server.

2.1.1 Mobile Client

Each user carries a mobile device on which a mobile client program is running. Each mobile client has the following attributes for content delivery.

- *user location:* Each mobile client is position-aware by using localization technologies (such as GPS [16], WiFi [17], Bluetooth [18], or GSM [19]), and is equipped with a wireless network interface that can communicate with a content delivery server. Each mobile client periodically acquires location information and sends it to the content delivery server.
- *user preferences:* The user preferences are in a list of personal information, such as interests (e.g. for *shopping*, *dining*, *cafe*) and statuses (e.g. *at the office*, *at home*, *on the train*). Each user can select preferences when subscribing for content-aware services.

2.1.2 Content Subscription Client

A content provider creates the content and sends it to the content delivery server that uses a content subscription client to request delivery to users. The content is referenced as a URL string that links to the content provider's website. The following attributes must be specified with the content as for the delivery conditions.

- *delivery area:* A delivery area is a square region. The content cannot be delivered when the *user location* is not within its *delivery area.*
- *delivery time period:* The content is delivered only in the delivery time period. For example, if the delivery time period is defined as 9:00 am to 6:00 pm on August 22 for some content, the content is a delivery candidate only during that time period.
- *delivery target preferences:* Some preferences of the users are specified for efficient deliveries. The number of matched attributes affects the content delivery scheduling described below.

2.1.3 Content Delivery Server

A content delivery server manages both content and users and has the role of delivering content to users according to the content delivery scheduler. The content delivery server delivers the content using such services as SMS (Short Messaging Service), MMS (Multimedia Messaging Service), and Internet mail services, as in [20].

2.2 Agile Content Delivery Scheduling

2.2.1 Overview

Our proposed content delivery scheduler, ACDS, runs on a content delivery server (Figure 3). There is a content queue for each mobile device to store delivery candidate content.

ACDS can select from four types of actions for each pair of a piece of content and a user: content aggregation, content reordering, content discarding, and content delivery.

- content aggregation

  All of the following conditions must be satisfied before a piece of content is enqueued into the content queue for a user.

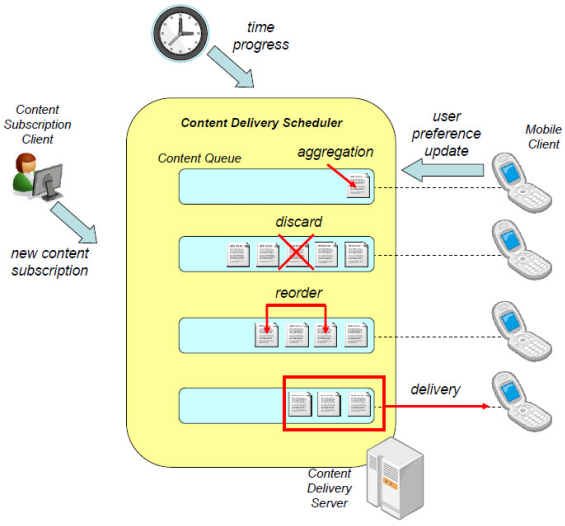

Figure 3: Content Delivery Scheduler

  - The size of the queue is less than the maximum size.
  - The current time is within the delivery time period of the content.
  - The user location of the mobile client is within the delivery area of the content.
  - The user preferences of the mobile client match the delivery target preferences of the content.

- content discarding

  A piece content in a content queue is discarded when any of the following conditions is fulfilled.

  - The current time is later than the delivery time period of the content.
  - The user location of the mobile client is outside of the delivery area of the content.
  - The user preferences of the mobile client do not match the delivery target preferences of the content.

- content reordering

  A piece of content in a content queue can be reordered according to the content priority score for each user. The content priority score is calculated for each user according to the number of matched preferences.

- content delivery

  Some content from the content queue is packed into a message and the message is delivered to the mobile device. Each piece of delivered content must satisfy all of following conditions when the content is delivered.

  - The current time passed more than a delivery interval time from the last delivered time.
  - The current time is within the delivery time period of the content.

– The user location of the mobile client is within the delivery area of the content.

– The user preferences of the mobile client match the delivery target preferences of the content.

A content reordering and a content discarding are always performed whenever their conditions are satisfied, because they are useful for both the content provider and the user. In contrast, content aggregation and content delivery should consider balancing the desires of both the content providers and the users.

There are three types of events that can trigger actions: a new content subscription event, a context update event, or a time progress event. ACDS attempts to select appropriate actions from the action candidates when one of these events occurs.

- New content subscription event

  A new content subscription event occurs when a content subscription client adds a content to a content delivery server. Action candidates are content aggregation, content reordering, or content delivery for each pair of content and user.

- Context update event

  A context update event occurs when a mobile client reports user location changes or preferences updates to a content delivery server. Action candidates are content aggregation, content reordering, content discarding, or content delivery for each pair of content and user.

- Time progress event

  A time progress event periodically occurs at time intervals set by the content delivery server administrator. The action candidate is content discarding for each pair of content and user.

Figure 4 shows an example of the scheduling. There are three content queues in the content delivery scheduler on the content delivery server, $ContentQueue_1$, $ContentQueue_2$, and $ContentQueue_3$ for $MobileDevice_1$, $MobileDevice_2$, and $MobileDevice_3$, respectively. A new content subscription event is reported to a content delivery scheduler when a content subscription client sends $Content$, then the scheduler takes actions. First, for $MobileClient_1$, the scheduler enqueues $Content$ into $ContentQueue_1$ (*content aggregation*), packs three pieces of content into MMS message, and delivers it to $MobileClient_1$ (*content delivery*). Next, for $MobileClient_2$, the scheduler enqueues $Content$ into $ContentQueue_2$ (*content aggregation*), and swaps it with the second content (*content reordering*). Finally, for $MobileClient_3$, the scheduler dequeues the third content (*content discarding*).

2.2.2 Balanced Distance-based Action Selection

ACDS must take appropriate actions for both content providers and users in action candidates for content aggregations and deliveries. For example, in Figure 4, there are 6 action candidates, a content aggregation for $MobileClient_1$, a content delivery for $MobileClient_1$, a content aggregation for $MobileClient_2$, a content delivery for $MobileClient_2$, a content aggregation for $MobileClient_3$, and a content delivery for $MobileClient_3$, The scheduler takes 3 actions with the candidates, a content aggregation for $MobileClient_1$, a content delivery for $MobileClient_1$, and a content aggregation for $MobileClient_2$.

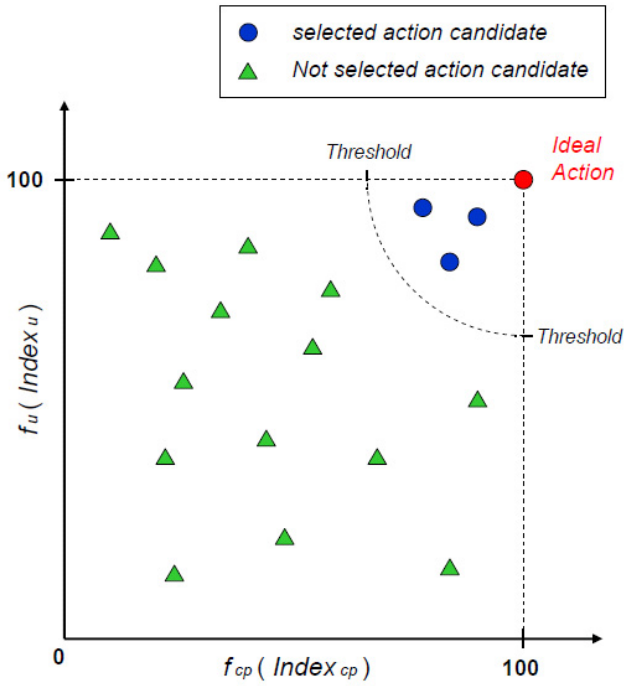

Figure 5: Balanced Distance-based Action Selection

We propose a balanced distance-based action selection algorithm for taking appropriate actions with action candidates (Figure 5).

We define two types of index values, $Index_{cp}$ and $Index_u$, that measure the desires of a content provider and a user respectively, based on system parameters, such as the number of delivered pieces of contents, the size of a content queue, the content priority score, and the number of matching preferences. Each action candidate can be expressed as a pair of an $Index_{cp}$ and $Index_u$. Next, each index value is normalized into the range of 0 to 100 with the functions f_{cp} and f_u, respectively. We can compare desires of the content providers and users directly with this normalization procedure. An ideal action which fully satisfies the desires of both the content provider and the user is scored as $(Index_{cp}, Index_u) = (100, 100)$. We calculate the *Balanced Distance* of each action candidate, which shows the distance between the action candidate and the ideal action.

$$BalancedDistance = \sqrt{(100 - f_{cp}(Index_{cp}))^2 + (100 - f_u(Index_u))^2}$$

The balanced distance for each action candidate is compared with *Threshold*, and a candidate whose balanced distance is no more than *Threshold* is selected.

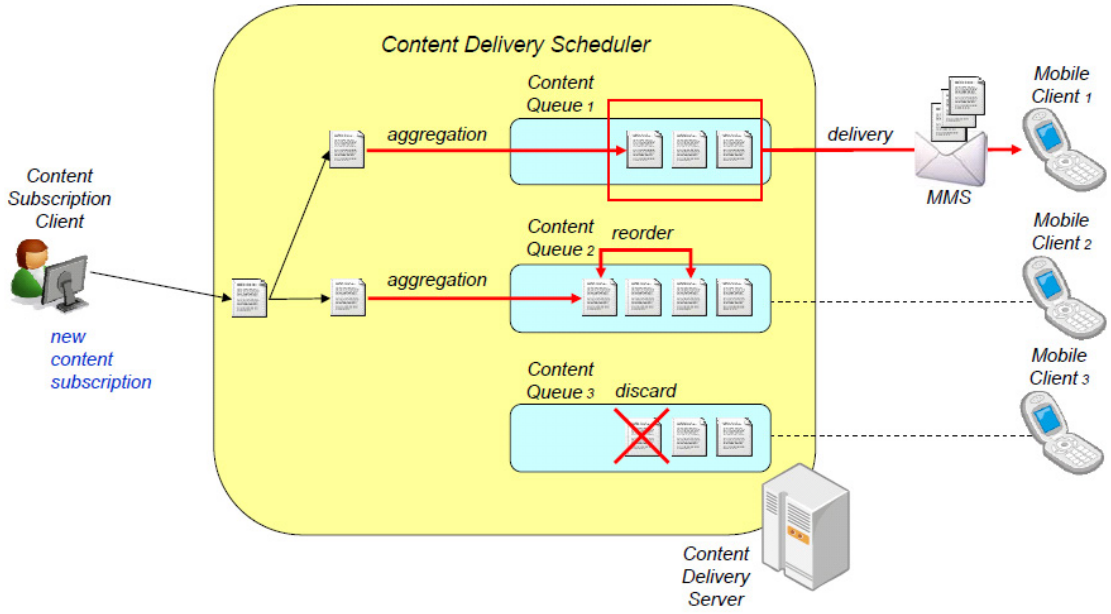

Figure 4: An Example of the Scheduling

With our balanced distance-based action selection algorithm, suitable actions for both the content provider and the user can be selected.

3 EVALUATION

In this section, we evaluate the performance of ACDS using a simulation.

3.1 Simulation Environment

3.1.1 Parameters

There are four types of simulation parameters: domain, user, content, and delivery.

- domain

  The simulation domain that users and contents are assigned to is fixed as a square whose height and width are $Space_{domain}$ meters. Each simulation period is $Period_{domain}$ minutes, and a simulation time passes every $Interval_{domain}$ minutes. We assume that the event for content delivery is only a context update event for user location update.

- users

  $Number_{user}$ users are generated and randomly placed in the domain. Each user randomly chooses at least one preference from the $NumberPreference_{user}$ preferences for user preferences. The users walk around the domain according to the Smooth Random Mobility Model [21], which makes the movement traces of mobile people more realistic than typical approaches that use stochastic principles for direction and speed control. The walking speed of the user changes randomly in the range of $\frac{1}{2}AverageSpeed_{user}$ to $\frac{3}{2}AverageSpeed_{user}$ m/s every $Interval_{user}$ minutes. The user location is reported every $Interval_{user}$ minutes, so a context update event occurs every $Interval_{user}$ minutes.

- content

  There randomly assigned $Number_{content}$ pieces of content in the domain at the start. At least one preference is selected from the $NumberPreference_{content}$ preferences as a delivery target preference. The delivery space of the content is set as a square whose size is $Space_{content}$ meters. The delivery time period for the content is also fixed at $Period_{content}$ minutes. A content priority score is calculated as $NumberofOverlappedPreference \times 10$.

- delivery

  A maximum of $MaxNumber_{delivery}$ pieces of content in a content queue whose maximum size is $MaxSize_{queue}$ can be delivered to a user every $Interval_{delivery}$ minutes.

Table 1 shows the values of the simulation parameters.

3.1.2 Scheduler

Three types of content delivery schedulers were used for comparison purposes on the same simulation domain, two ACDS schedulers whose threshold values are 15 and 25, respectively, and a naive scheduler that takes as many action candidates as possible without selecting from the candidates.

We used the following indexes to select appropriate action candidates for content aggregations and deliveries.

- $Index_{cp}$ for content aggregation: the size of the content queue

Table 1: Simulation Parameters

Parameter	Default	Range
$Space_{domain}$	10,000	NA
$Period_{domain}$	480	NA
$Interval_{domain}$	10	NA
$Number_{user}$	1,000	500 to 5000
$AverageSpeed_{user}$	40	NA
$Interval_{user}$	10	NA
$NumberPreference_{user}$	10	NA
$Number_{content}$	1,000	500 to 5,000
$Space_{content}$	2,000	NA
$Period_{content}$	180	NA
$NumberPreference_{content}$	10	NA
$MaxNumber_{delivery}$	3	1 to 10
$MaxSize_{queue}$	50	NA
$Interval_{delivery}$	20	NA

- $Index_{cp}$ for content delivery: the number of delivered pieces of content
- $Index_u$ for content aggregation: the content priority score of a pieces of enqueued content
- $Index_u$ for content delivery: the average of the priority scores of the delivered content

Each index is normalized by using normalized functions:

- f_{cp} for content aggregation

$$\begin{cases} 100, & \text{for } Index_{cp} \leq MaxSize_{queue} \\ 0, & \text{for } Index_{cp} > MaxSize_{queue} \end{cases}$$

- f_{cp} for content delivery

$$\begin{cases} 100, & \text{for } Index_{cp} \geq MaxNumber_{delivery}+1 \\ 90, & \text{for } Index_{cp} = MaxNumber_{delivery} \\ 80, & \text{for } Index_{cp} = MaxNumber_{delivery} - 1 \\ 75, & \text{for } Index_{cp} = MaxNumber_{delivery} - 2 \\ 0, & \text{for } Index_{cp} = 0 \end{cases}$$

- f_u for content aggregation and delivery

$$\begin{cases} 90, & \text{for } Index_u \geq 30 \\ 80, & \text{for } Index_u \geq 20 \\ 70, & \text{for } Index_u \geq 10 \\ 50, & \text{for } Index_u < 10 \end{cases}$$

As a performance index in the simulation, we used the number of expected clicks from the content, similar with the concept of Predicting Clicks [22]. The number of expected clicks from the content, $NumberOfExpectedClicksFromContent$, is calculated as follows:

First, we assume that the expected click ratio of $User_j$ for the received $Content_i$, $RatioOfContent_iforUser_j$, is the same as the content priority score for the user, calculated as $NumberOfMatchedPreference \times 10$. The average of the expected click ratios for $Content_i$ for all of the received users, $AverageRatiosOfContent_i$, is calculated as

$$AverageRatiosOfContent_i = \frac{\sum_{j=1}^{NumberOfReceivedUsers} RatioOfContent_iforUser_j}{NumberOfReceivedUsers}$$

Next, the number of expected clicks for $Content_i$, $NumberOfExpectedClicksFromContent_i$, is calculated as

$$NumberofClicksFromContent_i = NumberOfDelivered_i \times \frac{AverageRatiosOfContent_i}{100}$$

Finally, the number of expected clicks for the content, $NumberOfExpectedClicksFromContent$, is calculated as

$$NumberOfExpectedClicksFromContent = \sum_{i=1}^{NumberOfContent} NumberOfClicksFromContent_i$$

3.2 Simulation Results and Discussion

3.2.1 The Number of Pieces of Content

First, we fixed the number of users $Number_{user}$ at 1,000 and the maximum number of delivered pieces of content $MaxNumber_{delivery}$ at 3, and varied the number of pieces of content $Number_{content}$ from 500 to 5,000. The results are shown in Figures 6, 7, and 8.

Figure 6 shows that the most advantageous scheduler is the ACDC scheduler with the threshold of 15 when the number of pieces of content is 500 up to about 3,000 pieces, and the ACDS scheduler with the threshold of 25 from 3,000 to 5,000. From these results, we found that the proposed ACDS scheduler is more effective than the naive scheduler, but the threshold value should be carefully chosen (The naive scheduler is actually better than the ACDS scheduler if the threshold is 25 when the number of pieces of content is from 500 to about 1,000).

The results can be explained with reference to Figures 7 and 8. Figure 7 shows that the average of click ratios of all of the schedulers is nearly stable regardless of the amount of content. Figure 8 shows that the number of pieces of delivered content tends to increase along with the amount of content and then stabilizes at a certain time. The ACDS scheduler with threshold of 15 continues to increase from 500 to 5,000. The ACDS scheduler with threshold of 25 increases when the number is 500 to 2,000 is stable beyond 2,000. The naive scheduler is stable from 500 to 5,000. The figures indicate that we have to dynamically switch the threshold values of the ACDS scheduler along with changes in the number of pieces of delivered content.

3.2.2 The Number of Users

Next we fixed the number of pieces of content $Number_{content}$ at 1,000 and the maximum number of pieces of delivered content $MaxNumber_{delivery}$ at 3, and varied the number of users

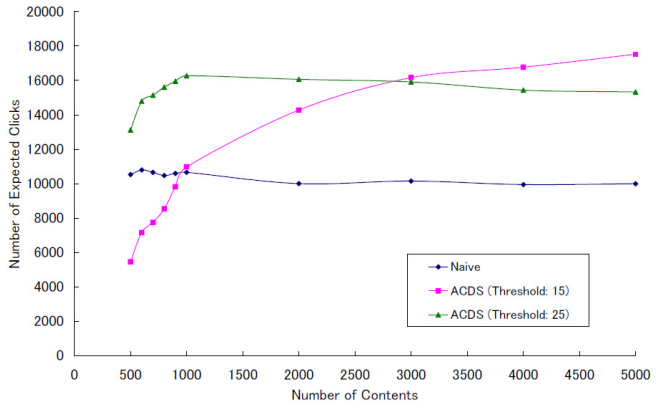

Figure 6: The Number of Expected Clicks vs. the Number of Pieces of Content

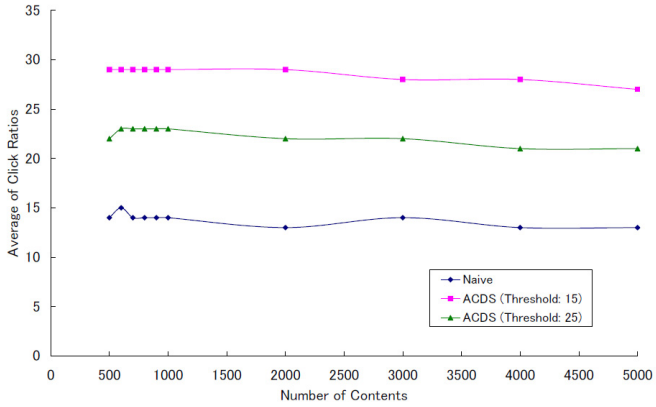

Figure 7: Average Click Ratio vs. the Number of Pieces of Content

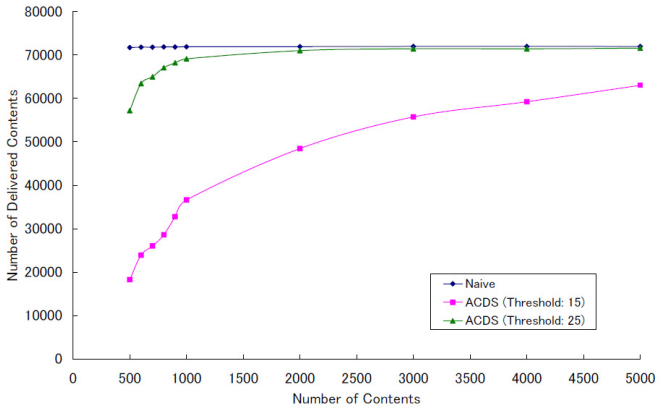

Figure 8: The Number of Pieces of Delivered Content vs. the Number of Pieces of Content

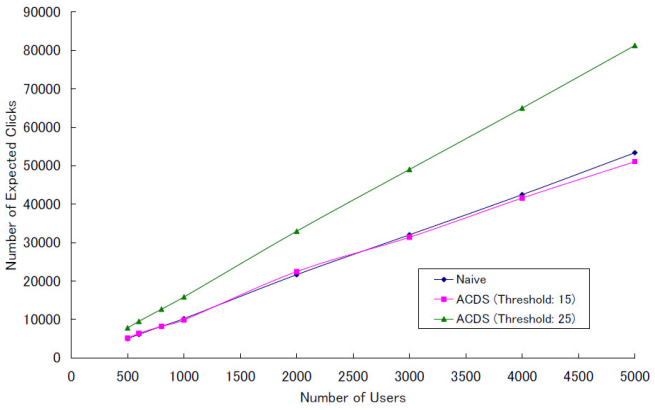

Figure 9: The Number of Expected Clicks vs. the Number of Users

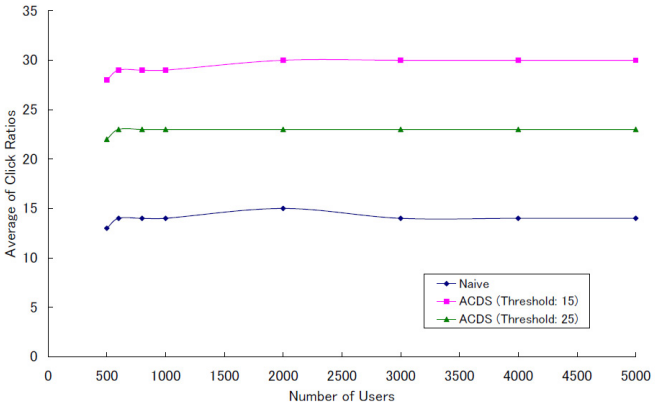

Figure 10: Average Click Ratios vs. the Number of Users

$Number_{user}$ from 500 to 5,000. The results are shown in Figures 9, 10, and 11.

Figure 9 shows that the most advantageous scheduler is the ACDS scheduler with the threshold of 15 regardless of the number of users. From this result, we found that the ACDS scheduler with the appropriate threshold value is the most effective one, but the threshold value should be selected carefully. An ACDS scheduler with the threshold of 25 is almost the same as the naive scheduler.

We can explain these results with Figures 10 and 11. Figure 10 shows that the average of the click ratio of all of the schedulers is nearly stable regardless of the number of pieces of content, even when the number of pieces of content is varied. Figure 11 shows that the number of pieces of delivered content continues to increase with the number of users and the order does not change regardless of the number of users. The figures indicate that we have to select an appropriate threshold value for the ACDS scheduler considering the amount of delivered content.

3.2.3 The Maximum Number of Delivered Contents

Finally, we fixed the number of pieces of content $Number_{content}$ at 1,000 and the number of users $Number_{user}$ at 1,000, and varied the maximum number of pieces of delivered content $MaxNumber_{delivery}$ from 500 to 5,000. These

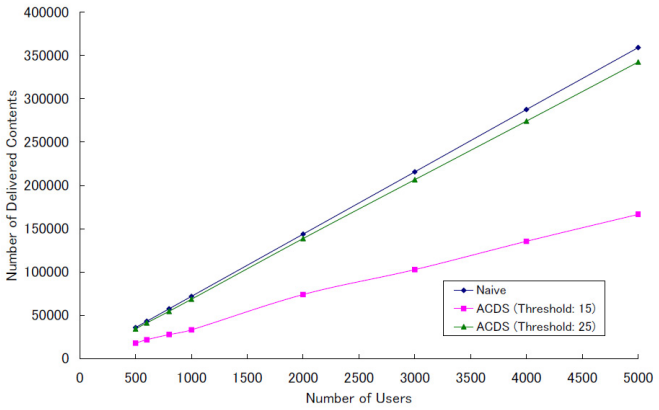

Figure 11: The Number of Pieces of Delivered Content vs. the Number of Users

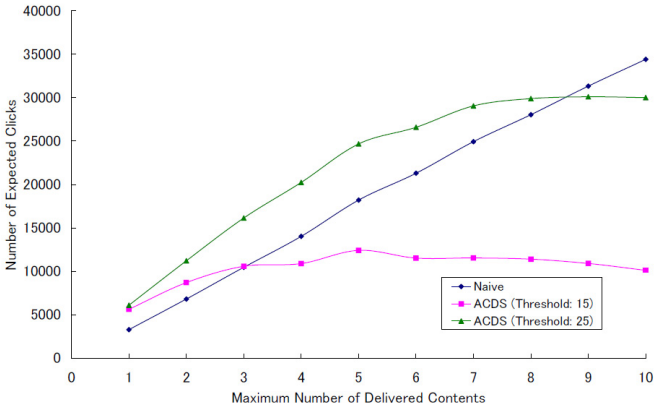

Figure 12: The Number of Expected Clicks vs. the Maximum Number of Pieces of Delivered Content

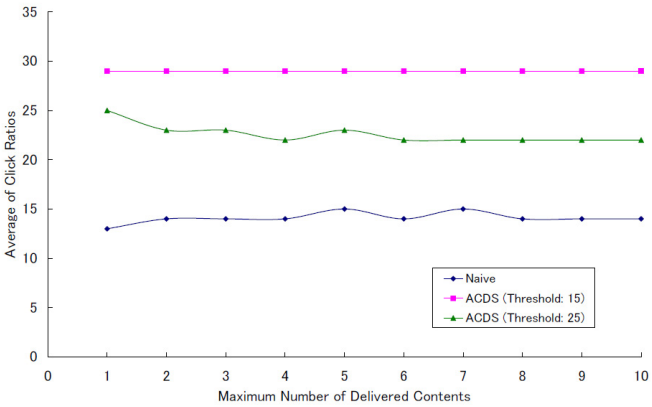

Figure 13: Average of Click Ratio vs. the the Maximum Number of Pieces of Delivered Content

results are shown in Figures 12, 13, and 14.

Figure 12 shows that the best scheduler is the ACDC scheduler with the threshold of 25 when the maximum number of pieces of delivered content is 1 to 8, and the naive scheduler when from 8 to 10. From these results, we found that the ACDS scheduler with suitable threshold value is the best, but the threshold value should be carefully selected

The results can be explained with Figures 13 and 14. Figure 13 shows that the average of the click ratio of all of the schedulers is nearly stable regardless of the maximum number of pieces of delivered content, as in the previous cases. Figure 14 shows that the number of pieces of delivered content tends to increase with the maximum number of pieces of delivered content and becoming stable for the ACDS schedulers while the number of pieces of delivered content continues to increase with the maximum number of pieces of delivered content. The ACDS scheduler with the threshold of 15 increases when the maximum number of pieces of delivered content is from 1 to 3, and is stable beyond 3. The ACDS scheduler with the threshold of 25 increases when the number is from 1 to 7 and is stable beyond 7. These results indicate that we have to increase the threshold value of the ACDS scheduler when the number of pieces of delivered content starts to become stable.

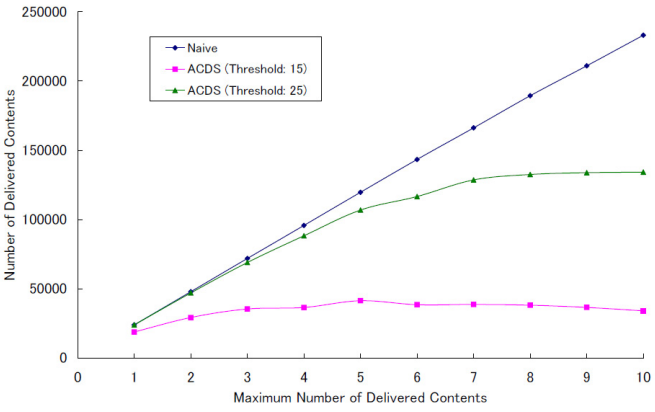

Figure 14: The Number of Pieces of Delivered Content vs. the Maximum Number of Pieces of Delivered Content

4 SUMMARY

The ACDS (Agile Content Delivery Scheduling) that we propose in this paper provides context-aware content delivery scheduling for large-scale context-aware services with a balanced distance-based action selection algorithm, which takes appropriate actions from the action candidates to properly balance both the content provider's desires and the user's. We evaluated the delivery performance by comparing ACDS with naive scheduling in simulations and showed that ACDS is twice better than the naive scheduling under some conditions.

In this paper, our ACDS was evaluated in a simulation environment. It would be useful to deploy ACDS in real environments and evaluate the performance. It would also be useful to add in the factor of remaining time for the content delivery into the balance distance based action selection algorithm, because the remaining time should be one of key factors when selecting appropriate actions.

5 ACKNOWLEDGMENTS

We wish to thank our colleagues at IBM who helped us with their insights in earlier discussions during our collaborative work. We also would like to thank Shannon Jacobs for checking the wording of this paper.

References

[1] Jenna Burrell, Geri K Gay, Kiyo Kubo, and Nick Farina. Context-aware computing: A test case. In *Proceedings of the 4th International Conference on Ubiquitous Computing (Ubicomp 2002), 2002.*

[2] Ying Cai and Toby Xu. Design, analysis, and implementation of a large-scale real-time location-based information sharing system. In *Proceedings of the 6th ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys 2008), 2008.*

[3] Shravan Gaonkar, Jack Li, Romit Roy Choudhury, Landon Cox, and Al Schmidt. Micro-blog: Sharing and querying content through mobile phones and social participation. In *Proceedings of the 6th ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys 2008), 2008.*

[4] Per Persson and Petra Fagerberg. Geonotes: A real-use study of a public location-aware community system. In *SICS Technical Report T2002-27, 2002.*

[5] Yasuharu Katsuno and Ryohji Honda. Magical device: A small device that makes it easy to build real-world navigation systems. In *Proceedings of the IEEE International Conference on Systems Man and Cybernetics (SMC 2000), 2000.*

[6] William G. Griswold, Patricia Shanahan, Steven W. Brown, Robert T. Boyer, Matt Ratto, R. Benjamin Shapiro, and Tan Minh Truong. Activecampus - experiments in community-oriented ubiquitous computing. In *IEEE Computer, Volume 37, Number 10, 2004.*

[7] Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. Experiences of developing and deploying a context-aware tourist guide: the guide project. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000), 2000.*

[8] Abhaya Asthana, Mark Cravatts, and Paul Krzyzanowski. An indoor wireless system for personalized shopping assistance. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, 1994.*

[9] Hao Yan and Ted Selker. Context-aware office assistant. In *Proceedings of the 5th International Conference on Intelligent User Interfaces (IUI 2000), 2000.*

[10] Jakob E. Bardram, Thomas R. Hansen, Martin Mogensen, and Mads Soegaard. Experiences from real-world deployment of context-aware technologies in a hospital environment. In *Proceedings of the 8th International Conference on Ubiquitous Computing (Ubicomp 2006), 2006.*

[11] Deepayan Chakrabarti, Deepak Agarwal, and Vanja Josifovski. Contextual advertising by combining relevance with click feedback. In *Proceedings of the 17th International World Wide Web Conference (WWW 2008), 2008.*

[12] Rebecca Bulander, Michael Decker, Gunther Schiefer, and Bernhard Kolmel. Comparison of different approaches for mobile advertising. In *Proceedings of the 2nd IEEE International Workshop on Mobile Commerce and Services (WMCS '05), 2005.*

[13] Dimitris Drossos and George M. Giaglis. Mobile advertising effectiveness: an exploratory study. In *Proceedings of the International Conference on Mobile Business (ICMB 2006), 2006.*

[14] Chingning Wang, Ping Zhang, Risook Choi, and Michael DEredita. Understanding consumers attitude toward advertising. In *Proceedings of the 8th Americas Conference on Information Systems, 2002.*

[15] William G. Griswold, Robert Boyer, Steven W. Brown, and Tan Minh Truong. A component architecture for an extensible, highly integrated context-aware computing infrastructure. In *Proceedings of International Conference on Software Engineering (ICSE 2003), 2003.*

[16] Motilal Agrawal and Kurt Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006), 2006.*

[17] Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca, and John Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *Proceedings of the 3rd ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys 2005), 2005.*

[18] Yasuharu Katsuno, Toru Aihara, Akihiko Mizutani, and Ken Tamagawa. Location detection by bluetooth wireless communication. In *Computer Software, Volume 20, Number 5, September, 2003.*

[19] Alex Varshavsky, Eyal de Lara, Mike Chen, Dirk Haehnel, Jeffrey Hightower, Anthony LaMarca, Jon Froehlich, Fred Potter, Timothy Sohn, and Karen Tang. Are gsm phones the solution for localization? In *Proceedings of the 7th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2006), 2006.*

[20] Patrick Barwise and Colin Strong. Permission-based mobile advertising. In *Journal of Interactive Marketing, Volume 16, Number 1, 2002.*

[21] Christian Bettstetter. Mobility modeling in wireless networks: Categorization, smooth movement, and border effects. In *Mobile Computing and Communications Review, Volume 5, Number 3, 2001*.

[22] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of the 16th International World Wide Web Conference (WWW 2007), 2007.*