Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2012, Article ID 935653, 15 pages
doi:10.1155/2012/935653

Research Article

Device Characteristics and Capabilities Discovery for Multimedia Content

Mohd Faisal Ibrahim,[1] Saadiah Yahya,[1] and Mohd Nasir Taib[2]

[1] *Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Malaysia*
[2] *Faculty of Electrical Engineering, Universiti Teknologi MARA, 40450 Shah Alam, Malaysia*

Correspondence should be addressed to Mohd Faisal Ibrahim, faisal@tmsk.uitm.edu.my

Received 5 January 2012; Revised 20 April 2012; Accepted 21 April 2012

Academic Editor: Yueh M. Huang

The rapid growth of web and mobile technologies has allowed people to access multimedia content from a wide range of heterogeneous client devices that have different characteristics and capabilities. In order to deliver the best presentation of content requested, the web system must possess a mechanism that is able to accurately discover the characteristics and capabilities of a client's device. Existing content negotiation techniques mainly focus on static profiling approach without considering the combination of static and dynamic approaches that are capable of overcoming the device scalability issues. In view of these issues, we propose a hybrid approach for recognizing devices and their capabilities using token-based method. By proposing such solution, we can provide flexible, extensible, and scalable method that provides more accurate information in resolving the content negotiation issues. To validate the effectiveness of the proposed method, we construct a laboratory and field studies to investigate its performance and accuracy. The experimental results show that the proposed hybrid approach has better performances in several aspects compared to the static profiling.

1. Introduction

The concept of universal multimedia access (UMA) has created a remarkable need to access multimedia content not only from personal computers (PCs) but also from mobile devices. Due to the growing number of new mobile devices, providing content in a usable format for UMA is challenging and still difficult to accomplish. Moreover, client access through mobile device has several limitations in terms of bandwidth, battery capacity, screen resolution, processing power, capabilities, and communication costs.

In order to make content accessible on both PCs and mobile devices in UMA, a flexible content negotiation strategy is required for providing different representations or contents of the same resource to requested client [1]. The web system should automatically detect the devices' capabilities, and hardware and software constraints and provide a suitable content based on the client characteristics, capabilities, and preferences.

There are several standards and mechanisms that have been proposed for this matter in content negotiation such as hypertext transfer protocol (HTTP) header mechanisms, resource description framework (RDF) profiles that consists of composite capabilities/preferences profile (CC/PP) [2], user agent profile (UAProf) [3], and WURFL (Wireless Universal Resource FiLe) [4]. However, these standards and mechanisms offer limited possibilities in device identification [5–7]. Adapting the static profiling approach to identify a device, whereby depending solely on human intervention to constantly contribute updates on the device is simply impossible as more upgrades and new devices are introduced in the market day by day. Hence, a more scalable and flexible approach is required.

Most of the existing efforts for content negotiation propose guidelines that mainly address static [8–13] or dynamic [14–16] content negotiation approach. The advantage of using a static approach is that it can provide a quick way for identifying and detecting the client capabilities based on the

device profile repository. This information will be used by the content provider to fetch the required profile of a specific device. The main benefit of using a dynamic approach is that this approach is not bounded to the dependency of the device vendor in updating the device repository and therefore provides a scalable approach to the device identification and detection problem. Little consideration is given towards hybrid approach that capable of overcoming the device scalability issues. Moreover, providing content negotiation solutions addressed to UMA requires a wide range of approaches which can be separated into two processes: device identification and device capabilities detection. The fundamental objective of this paper is to provide a flexible, extensible, and scalable content negotiation approach by allowing information regarding new device features, file formats, and matching criteria to be added into the system. This solution is critical in device heterogeneity environment because different types of devices have different capabilities for multimedia processing [17].

The remainder of this paper is structured as follows. Section 2 presents the background required to understand the approach and summarizes related works. Section 3 presents the proposed method. Section 4 presents the experiments and results achieved. Section 5 discusses the findings. Finally, Section 6 concludes the paper and summarizes the future work.

2. Preliminaries

2.1. Device Characteristics and Capabilities Discovery. One of the key technical issues in developing content negotiation approach is the problem of how to provide accurate and comprehensive profiles of heterogeneous clients and how these can be used to identify the device capabilities, especially if new devices are available in the marketplace. The main problem of the heterogeneous mobile devices is its format or characteristic which is dissimilar for different manufacturers or vendors. A file format or feature that is supported on one mobile device may not be available on another device model. In addition, future device model might include a new file format which is not defined in the current device profiles. Therefore, the ability to recognize and discover device capabilities is important in content negotiation method.

Device characteristics and capabilities discovery can either be static or dynamic. The static approach uses profiles for describing delivery context. The dynamic approach, in contrast, relies only on the metadata attached to every HTTP request in the form of HTTP headers. There are a number of different implementation approaches or methods to discover information about the delivery context. Each of these methods has benefits and drawbacks depending on application requirements and can be classified as follows.

2.1.1. Dynamic Approach. The dynamic approach presents a method whereby device identification and capabilities are detected based on the user agent request header field. This field contains information about the user agent originating the request [18] and can be used to identify a user agent and client device. Several information such as the device model, device manufacturer, client device's operating system, as well as browser and Java capabilities can be found in the user agent header. However, the major issue related to the user agent header mechanisms is its nonstandardized format in transmitting the information [19] and the headers are quite limited for describing delivery context [20].

2.1.2. Static Approach

In this approach, information regarding device characteristics and capabilities will be stored and maintained manually into the database or reference repositories by the device manufacturers or developers. There are two types of device discovery method in static approach.

(i) Based on profile header. In this method, when a client sends a request to the server, it also sends a profile header which contains the URL and a set of descriptions of device capabilities. This indicates the server on where to find the device profile and extract the device description which is provided by the hardware or software vendor. This method has been employed in several standards such as Composite capabilities/preferences profiles (CC/PP) and user agent profile (UAProf). However, this approach has numerous shortcomings: the profiles may be invalid [6], not all mobile devices support CC/PP [21, 22] or UAProf [20, 23, 24], do not define how the servers or proxies should do transformations or customizations based on the device capability information [23], prohibit collections of profiles or components inside the same profile [25], and only focus on descriptions for WAP devices [26].

(ii) Based on user-agent string. This method recognizes devices and their capabilities based on their user agent string. When a server receives a request, it queries the database using user agent string to describe the device capabilities [27]. An example of the device capabilities discovery solution that applies this method is WURFL (Wireless Uniform Resource FiLe). WURFL is an open source profile repository which holds a configuration file containing information about the features of most mobile devices offered in the market. However, the reliance on this approach can easily lead to out-of-date information [28] as it is bounded to the dependency of the device vendor in updating its repository. Existing devices with add-ons and newly installed applications will not be detected if its new capabilities have not been updated in the repository [11]. Moreover, the identification of the device model alone and associated static parameters can sometimes prove insufficient for media adaptation [29].

Figure 1 illustrates how static approach works based on user agent string. When the user uses the device to connect to the server, the UAS from the device will be captured for the device identification process. The user agent matcher will

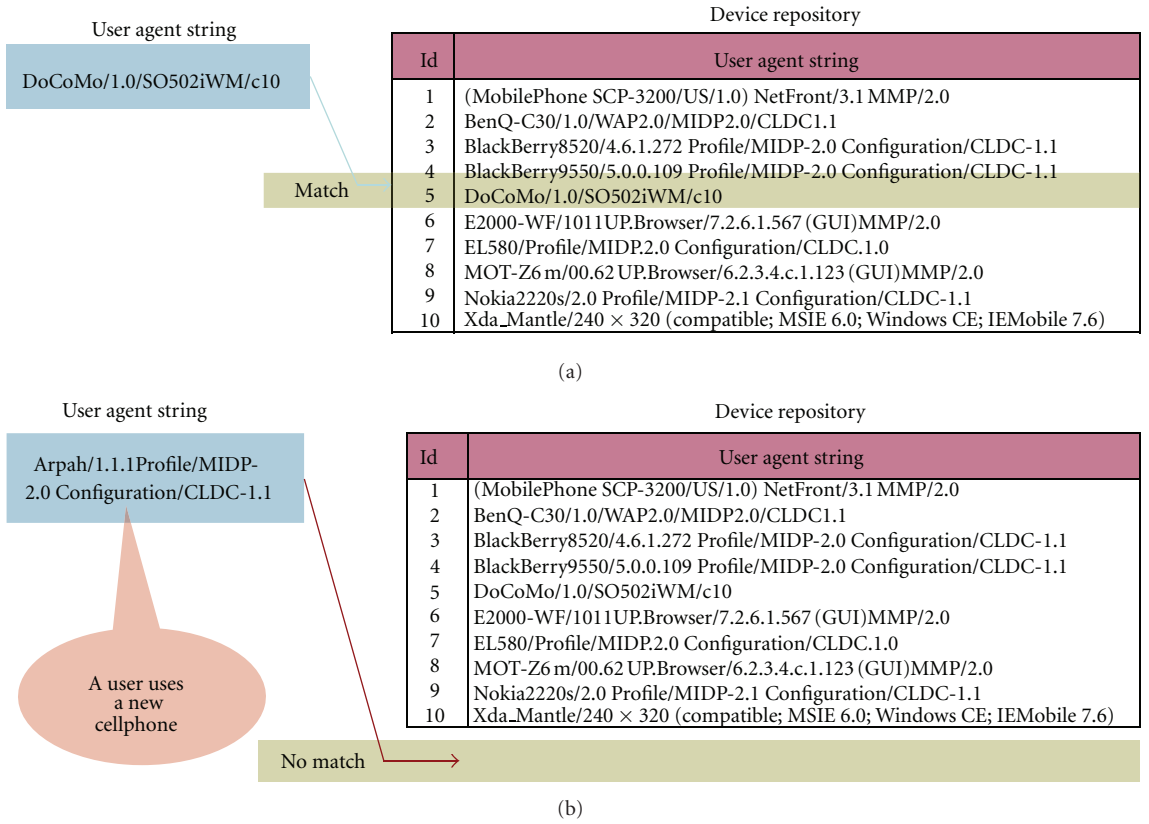


Figure 1: Illustration of the problem with static approach: (a) UAS exists in the device repository (b) UAS does not exist in the device repository.

then use the captured UAS to perform a one-to-one match with the UAS listed in the device repository. Sequentially, the matching process takes place going down the list, and if a match does exist the device will then be recognized as either a mobile device, personal computer, or other type of device. In this scenario, the UAS captured from the client device had an exact match with the 5th UAS residing in the device repository, and so subsequently conclude that this device is a mobile device (Figure 1(a)). This is an example of a device identification based on preexisting UAS profile defined in the device repository.

In the event that the user is using a new cell phone, a similar process of device identification takes place. However, when the exact one-to-one matching process ended with no match between the client device's UAS and the UAS in the device repository, the client device is not recognized and is classified as an unknown device (Figure 1(b)).

2.1.3. Hybrid Approach. Both static and dynamic methods have advantages and disadvantages. These two methods are complementary rather than mutually exclusive of each other. Therefore, a hybrid approach is expected to generate a better solution, at the cost of being able to recognize and discover device capabilities through the combination of both approaches.

2.2. Related Work. Numerous standards have been proposed to provide the descriptions about the device's capabilities using static or dynamic content negotiation techniques. For instance, W3C has proposed the CC/PP [30] and Wireless Application Protocol (WAP) Forum has recommended a similar approach called UAProf [31] which are based on RDF.

Beside these standards, there are extensive researches on content negotiation based on dynamic approaches. Mohan et al. [14] and Ma et al. [15] presented a method to discover client capabilities through HTTP request header and user advice. A user interface is provided for the user to submit information about user preferences and device capabilities if the information is not available from the request header.

To the best of our knowledge, a combination of static and dynamic approaches for device identification was originally introduced by Müller et al. [32]. They proposed a two-step approach for device identification: firstly by using CC/PP, and if not found, by analyzing the client's HTTP user agent based on the keywords or patterns. The definitions of keywords (e.g., S55 or Nokia7250) and of pattern (e.g., 240 × 320 or Windows) are used to assume that the client device is a mobile. The use of keyword has some limitations because information on a user agent string is not standardized and usually defined by the manufacturers or browser vendors. Furthermore, not many manufacturers or

browser vendors provide information on the device's screen resolution.

In recent work, Zhao and Okamoto [7] have proposed a device identification approach for mobile learning environment. Their approach is divided into two steps: (1) device capabilities extraction using WURFL and (2) device identification based on HTTP request header. They used several PHP commands like HTTP_*X_JPHONE_DISPLAY* and HTTP_*X_UP_DEVCAP_SCREENPIXELS* in the second step for device identification if the device features cannot be discovered from the first step.

Both Müller and Zhao approaches focused purely on device identification without considering the issues of device capabilities detection if the static approach (CC/PP or WURFL) is unable to detect the device features. Many of the current approaches, however, do not provide enough evidence that their dynamic approach is reliable in identifying the capabilities of different types of client devices. Furthermore, there is no discussion regarding the extra computational overhead required in computing the negotiation process for both static and dynamic approaches. To overcome this issue, the proposed solution should adopt a method for this overhead. In the solution, we further extended the utilization of hybrid approach by showing that dynamic technique can be used to discover device capabilities if the information regarding the HTTP accept or MIME headers are provided by device manufacturer.

3. The Proposed Approach

3.1. Token Matching. Token matching technique compares attributes in the captured UAS or MIME header with a predefined token to determine whether there is a match between these attributes and tokens. If there is a match between UAS and tokens, we can infer inductively that the device is a mobile device or a PC. If the attribute in the token can be corresponding pair wise with any attributes of MIME header, then we can conclude that the device is capable of supporting that particular format represented by the token attributes.

In this approach, when a user is using a new cell phone, all the fragmented tokens from the UAS will be matched with the matching criteria attributes (mobile browser (MOB), mobile manufacturer (MOM), mobile information (MOI), mobile operating system (MOS), and mobile network operator (MNO)). In this scenario, as depicted in Figure 2, the second attribute CLDC listed in the MOI table matched exactly with the fourth fragmented token (which is CLDC) in the UAS. This recognizes the device as a mobile device based on its detected attribute which is CLDC.

3.2. Problem Definition and Formulation. In this section, we give an overview of the proposed method for identifying and detecting client device. We will define token, user agent string, HTTP accept header, matching criteria, and describe our problem formulation.

Definition 1. A token T contains a sequence of characters or keywords in a user agent string UAS, HTTP accept, or MIME header which represents information regarding browser, operating system, java runtime environment, types of wireless devices, network operator, and file types that can be handled by the user agent.

Definition 2. A user agent string UAS contains a line of text that gives information about compatibility, device manufacturer, browser, operating system, screen resolution, Java capabilities, and so forth. HTTP accept or MIME header contains a text list of MIME media types that will be accepted by the user agent.

Definition 3. Matching criteria MC is a rule consisting of a pair (user agent string UAS, token T) or (HTTP accept/MIME header MIME, token T) where $\{t$: match(UAS, $T)\}$ or $\{t$: match(MIME, $T)\}$. The matching criteria process is repeated until a termination condition is satisfied.

Problem Formulation. Given a set of matching criteria $\mathrm{MC} = \{\mathrm{MC}_1, \mathrm{MC}_2, \ldots, \mathrm{MC}_n\}$ and token $T = \{T_1, T_2, \ldots, T_n\}$, find in a set of strings $\mathrm{UAS} = \{\mathrm{UAS}_1, \mathrm{UAS}_2, \ldots, \mathrm{UAS}_n\}$ or $\mathrm{MIME} = \{\mathrm{MIME}_1, \mathrm{MIME}_2, \ldots, \mathrm{MIME}_n\}$ all strings matching to T. Our problem is to find in UAS or MIME all tokens T_i that satisfy matching criteria MC_j where $i = j$.

3.3. Token Attributes Lookup Method for Device Identification and Capabilities Detection. This paper proposes a token attributes lookup method. This method will be invoked in two phases, the device identification phase, and device capabilities detection phase.

(1) *Device Identification.* In order to preserve flexibility and scalability, all premises and consequents are not hardcoded but instead are developed systematically in a database. This is to allow extensibility where any existing or new premises and consequents can be integrated into the system by simply adding attributes. The specification of device can be performed via token attributes matching that includes 4 steps as depicted in Figure 3.

Step 1. The user agent string (UAS) is captured when the client device sends a request to the server.

Step 2. Then, while receiving the UAS, the unwanted characters will be removed from the UAS and then will be fragmented into several token attributes and used together with the constructed matching criteria for matching purposes. In this process, all alphabetical characters in the UAS will be converted to lowercase. Then, all the unwanted symbols will be removed from the UAS and will be replaced with a space. The filtered user agent string will then be tokenized by using a space as the delimiter.

Step 3. Each matching criteria will be invoked sequentially to find the right match.

Step 4. Once there is a match between the user agent string and one of the token attributes, the matching process will stop and the device will be classified as either a mobile device or a PC. Otherwise, it will be identified as unknown device.

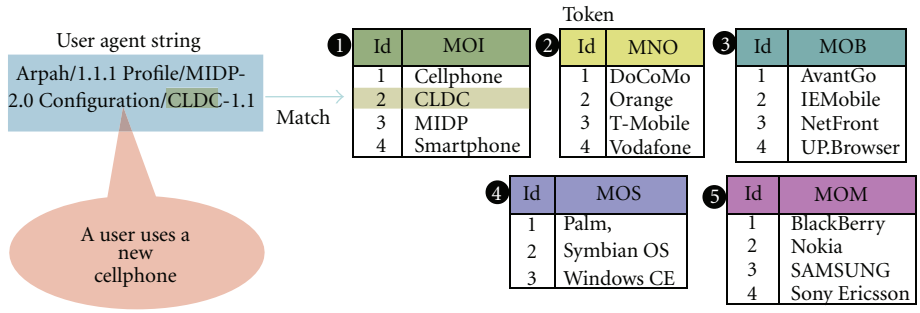


Figure 2: Illustration of how token matching approach manages to overcome static approach (user agent string) limitation.

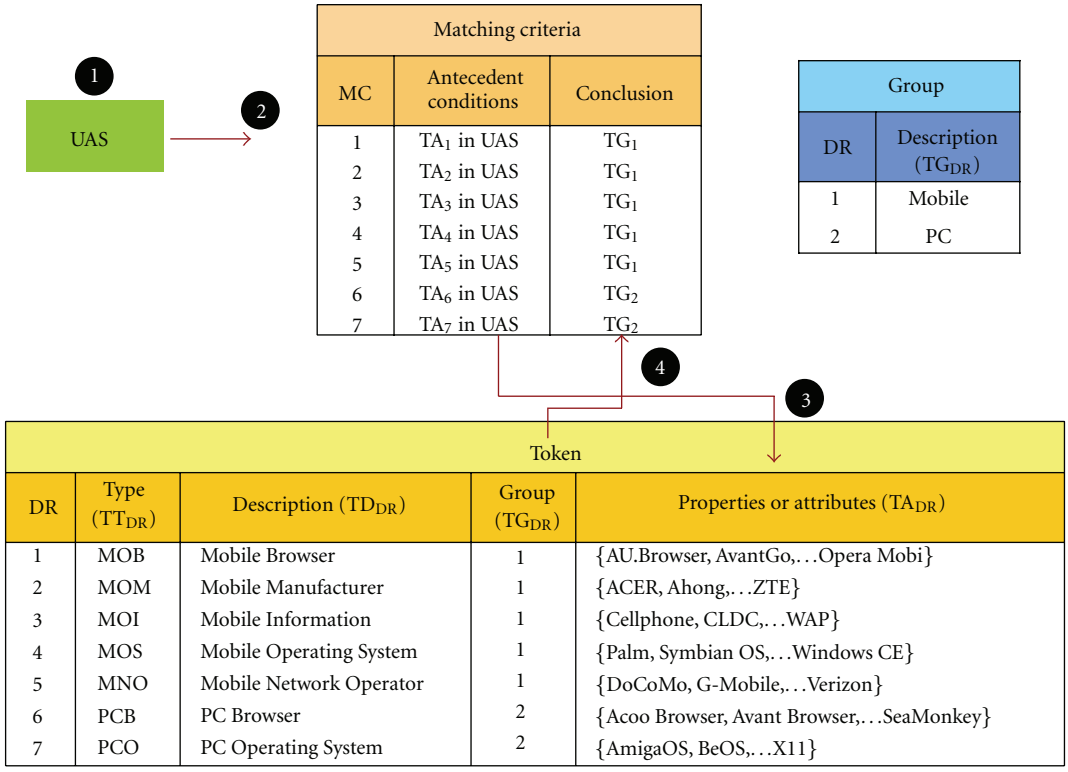


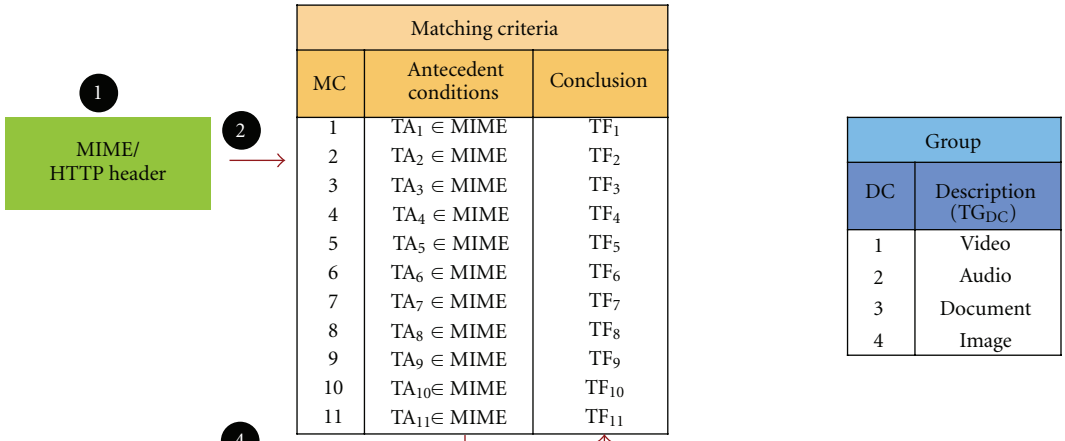
Figure 3: Matching criteria to detect type of device.

The matching criteria for device identification can be expressed as follows:

$$\mathrm{MC_{DR}}: \text{If } \mathrm{TA_{DR}} \in \mathrm{UAS} \text{ then } \mathrm{TG_{DR}}, \quad (1)$$

where DR = $1, \ldots, n$, $\mathrm{MC_{DR}}$ indicates the number of matching criteria for device identification, $\mathrm{TA_{DR}}$ contains the attributes of each token. For instance, token attributes $\mathrm{TA_1}$ = {AU.Browser, AvantGo, Opera}, $\mathrm{TA_2}$ = {ACER, Ahong$\cdots$ZTE}, $\mathrm{TA_3}$ = {Cellphone, CLDL, WAP}, and so on. $\mathrm{TG_{DR}}$ represents the group type of the device which is either mobile device or PC.

(2) *Device Capabilities Detection.* Similar with the previous concept in device identification mechanism, all token attributes and matching criteria are stored in the database for effortless system upgrades and modifications. Figure 4 illustrates the matching criteria for device capabilities specification. The only difference is that the information listed in the HTTP accept header and MIME types will be split into individual token attribute and will be used for device capabilities detection mechanism. Each token will be matched with the listed attributes sequentially to specify the device supported format. Once there is a match, the identified format will be stored in the database and the matching process will resume matching the token with other token attributes. As a device may have several supported formats, all its capabilities of supporting formats for video, audio, document, and image will be recognized at the end of the matching process.



Token

DC	Format (TF_{DC})	Description (TD_{DC})	Group (TG_{DC})	Attributes (TA_{DC})
1	3GP	3rd generation partnership project	1	Video/3gpp
2	MP4	MPEG-4 part 14	1	Video/mp4
3	WMV	Microsoft Windows media video	1	Video/x-ms-wmv
4	MP3	MPEG-1 audio layer 3	2	Audio/mpeg, audio/x-mpeg, audio/mp3, audio/x-mp3, audio/mpeg3, audio/x-mpeg3, audio/mpg, audio/x-mpg, audio/x-mpegaudio
5	WAV	Waveform audio format	2	Audio/wav, audio/x-wav, audio/wave, audio/x-pn-wav application/pdf, application/x-pdf,
6	PDF	Adobe portable document format	3	Application/acrobat, applications/vnd.pdf, text/pdf, text/x-pdf
7	DOC	Microsoft WORD	3	Application/doc, application/vnd.msword, application/vnd.ms-word, application/winword, application/word, application/x-msw6, application/x-msword
8	PPT	Microsoft PowerPoint	3	Application/vnd.ms-powerpoint
9	BMP	Bitmap	4	Image/bmp, image/x-bmp, image/x-bitmap, image/x-xbitmap, image/x-win-bitmap, image/x-windows-bmp, image/ms-bmp, image/x-ms-bmp, application/bmp, application/x-bmp, application/x-win-bitmap
10	GIF	Graphics interchange format		
11	JPEG	Joint photographic experts group	4 4	Image/gif, image/x-xbitmap, image/gi_ image/jpeg, image/jpg, image/jpe_ , image/pjpeg, image/vnd.swiftview-jpeg

Figure 4: Matching criteria to detect device capabilities.

The matching criteria for device capabilities detection can be expressed as follows:

$$MC_{DC}: \text{ If } TA_{DC} \in \text{MIME then } TF_{DC}, \quad (2)$$

where DC = 1, . . . , n, MC_{DC} indicates the number of matching criteria for specifying device capabilities, TA_{DC} contains the attributes of each token. For instance, token attributes TA_1 = {video/3gpp}, TA_2 = {video/mp4}, TA_3 = {video/mov, video/quicktime}, and so on. TF_{DC} represents the supported format.

3.4. Lexical Analyzer and Token Attributes Matching Algorithms. This paper proposes a token attributes lookup method. This method will be invoked in two phases, the device identification phase and device capabilities detection phase.

```
ALGORITHM Lexical Analyzer (String, Searched_Character)
// Input: A string of HTTP user agent and MIME type
// Output: Filtered UAS or MIME
(1)  n ← String · Length
(2)  String_Lower ← StrToLower(String)
(3)  Filtered_String ← " "
(4)  for i ← 1 to n do
(5)     if (String_Lower[i] ≠ Searched_Character)
(6)       Filtered_String ← Filtered_String + String_Lower[i]
(7)       i ← i + 1
(8)  return Lexical Analyzer
```

Algorithm 1: Lexical analyzer algorithm.

```
ALGORITHM TokenAttributesMatch (UMS, TA[1..MC, 1..TT])
// Input: A string of HTTP user agent, HTTP Accept and Mime
headers, a string TA[1..MC, 1..TT] of required pattern (Called
Token Attributes)
// Output: Boolean value of TokenAttributesMatch (True or False)
(1)  TokenAttributesMatch ← false
(2)  if (UAS = TA[i,j])
(3)     TokenAttributesMatch ← true
(4)  return TokenAttributesMatch
```

Algorithm 2: Token attributes matching algorithm.

The *Lexical Analyzer* algorithm in Algorithm 1 will scan the UAS or MIME type character by character. Once the nonalphanumeric character or space is identified during the scanning process, the entire characters before the space will be extracted as one token. The first two lines of the algorithm are to get the character length of the UAS or MIME and then convert the whole UAS or MIME into small letters. The condition statement in line 5 is to detect whether there is a nonalphanumeric character or a space in the UAS or MIME which will be removed by the algorithm.

In this subsection, we also describe the details of token attributes matching algorithm as shown in Algorithm 2. The input parameters to the algorithm are HTTP user agent strings, MIME and HTTP accept headers. Step 1 ensures that *TokenAttributesMatch* is always set to false before the pair wise matching process start. As mentioned earlier, each captured string will be matched with all token attributes TA$[i, j]$ as defined in the matching criteria *mc*. *TokenAttributesMatch* will be set to true if there is a match between the captured user agent string, HTTP accept or MIME headers *UMS* with token attributes TA$[i, j]$ (Steps 2–3). This algorithm will be used in the device identification and capabilities detection algorithms.

(1) *Device Identification Algorithm.* This algorithm is used to recognize type of device (see Algorithm 3) and will be invoked each time there is a client request to the server. Thus, in Step 3, the captured user agent string UAS and all token attributes TA$[i, j]$ are forwarded to *TokenAttributesMatch* algorithm for matching purposes in each iteration of the for-loop (Steps 1–2), until all rules in outer loop MC have been executed. If pair wise matching function *TokenAttributesMatch* (UAS, TA$[i, j]$) returns a true value (Step 4), then TG$[i]$ will be set to the token group value which is 1 for mobile and 2 for PC.

The main purpose of the *Device_Identification* algorithm is to identify the user agent string UAS characteristics. If these characteristics are determined to represent a mobile device, the algorithm will set TG$[i]$ to 1 so that the server can provide suitable user interface to be displayed efficiently on a small screen. This can be done by simplifying the style of the page and limiting the number of features such as lengthy text, unimportant information, or heavy images. As shown in Figure 5(b), we present an example for the adapted result generated by the device identification algorithm. These figures display the login and navigation pages for our e-learning websites when it is being accessed from mobile device. For mobile display, the screen will be split into a series of screens, which are linked with each other. If the website is accessed from a PC, a comprehensive user interface with complete information will be shown on the user's PC as demonstrated in Figure 5(a).

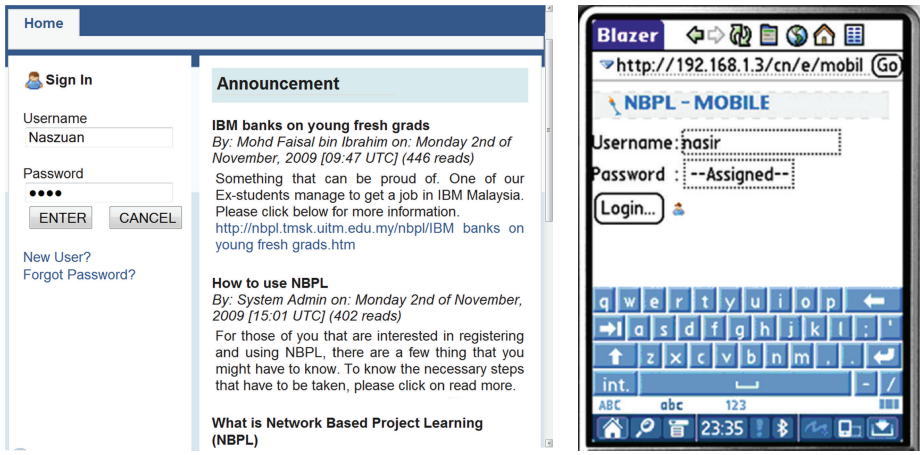
Here, we use an example to explain how the 4 steps depicted in Figure 3 and algorithm in Algorithms 1, 2 and 3 works. When the user uses their device to connect to the server, the UAS from the device will be captured for the device identification process. The UAS will be used by the token matcher to match the properties or attributes of predefined matching criteria of MOI, MNO, MOB, MOS,

```
ALGORITHM Device_Identification
// Output: Integer value of TG[i]
(1)  Non-Alphanumeric ←[! , @, #, $, %, , &,* , (, ), >, <, {, }, -]
(2)  Lexical Analyzer (UAS, Non-Alphanumeric)
(3)  for i ← 1 to mc
(4)    for j ← 1 to tt
(5)      if TokenAttributesMatch (UAS,TA[i,j])
(6)        TG[i] ← Token group value
(7)  return
```

Algorithm 3: Device identification algorithm.

```
ALGORITHM Device_Capabilities_Detection
// Output: String value of TF[i]
(1)  Lexical Analyzer (MIME, " ")
(2)  for i ← 1 to mc
(3)    for j ← 1 to tt
(4)      If TokenAttributesMatch (MIME,TA[i,j])
(5)        TF[i] ←Token format string
(6)  return
```

Algorithm 4: Device capabilities detection algorithm.



(a) (b)

Figure 5: Screenshots of e-learning websites when accessed through (a) PC and (b) mobile device.

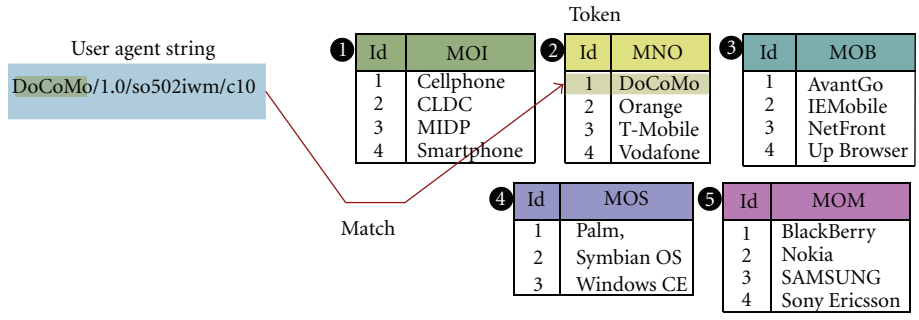


Figure 6: Device identification using token matching approach.

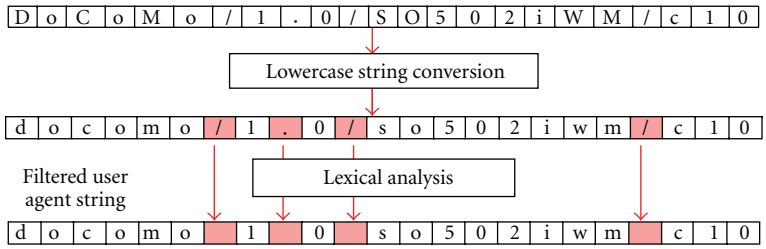
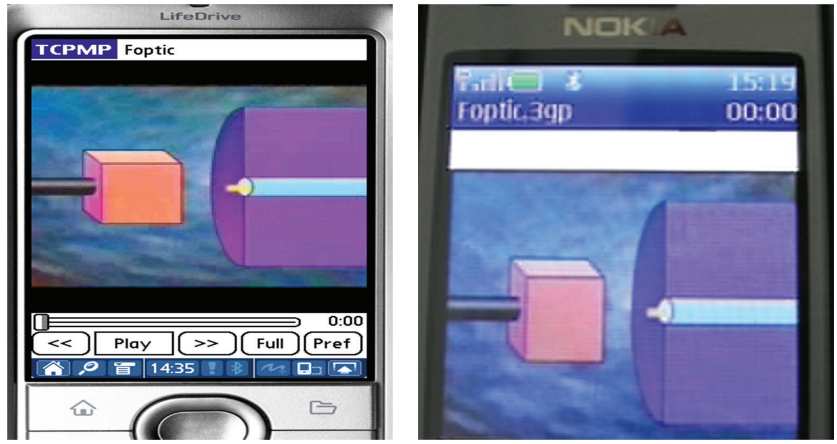


Figure 7: Lexical analysis by removing unwanted symbols.



(a) (b)

Figure 8: A video being displayed on a mobile device: (a) without video transcoding operation taking place, (b) after the video has been converted into 3GP format.

and MOM as described in Figure 6 to each fragmented tokens from the UAS (DoCoMo,1, 0, SO502iWM, c10). The lexical analyzer algorithm will convert all alphabetical characters in the user agent string to lowercase and replaces all the unwanted symbols with a space. This process is illustrated in Figure 7.

In this example, the first token that will be compared is docomo. The token attributes matcher (*TokenAttributes-Match* algorithm) goes from attribute to attribute trying to find a match. When it finds a match, it quits and concludes the device type. If no match is found, the token attributes matcher will continue to look for a match in the next respective criteria table. In this scenario (Figure 6), the attribute listed first in the second criteria table MNO is matched with its exact first fragmented token in the UAS, recognizing the device as a mobile device based on the name of its network operator, docomo.

(2) *Device Capabilities Detection Algorithm.* Algorithm 4 shows the device capabilities detection algorithm which is used to detect device capabilities. This algorithm works exactly like device identification algorithm except that MIME will be used instead of UAS. If there is a match in Step 3, token format TF[j] will be assigned with a string value such as 3GP, MP4, MOV, among others.

When the user download a video, the system analyzes the characteristics of the user's device (from the device identification algorithm) and then determine whether the device can display the content or not based on the device capabilities detection algorithm. If the device can display the video in its current state (AVI format), the original video will be retrieved (Figure 8(a)). If the device cannot display the AVI format, the video will be adapted to another format which is 3GP, before being displayed for the user (Figure 8(b)).

3.5. Overview and Architecture of DICAL. The hybrid approach and all the algorithms proposed here is part of the design of device identification and capabilities detection using token attributes lookup method (DICAL). It consists of a device database and two processing components: (1) device identification module and (2) device capabilities detection module. The device database contains the information regarding token attributes, matching criteria and token group profiles. When a client sends a HTTP request, an HTTP user agent string handler in DICAL captures the request and forwards it to the token attributes matching component. This component compares the captured user agent string with all the device identification tokens. After detecting type of device, the negotiation process continues with the device capabilities detection by checking the presence of the requested header in the cache. If there is a match, the server will get the device capabilities information from the cache. If there is no match, the server will then

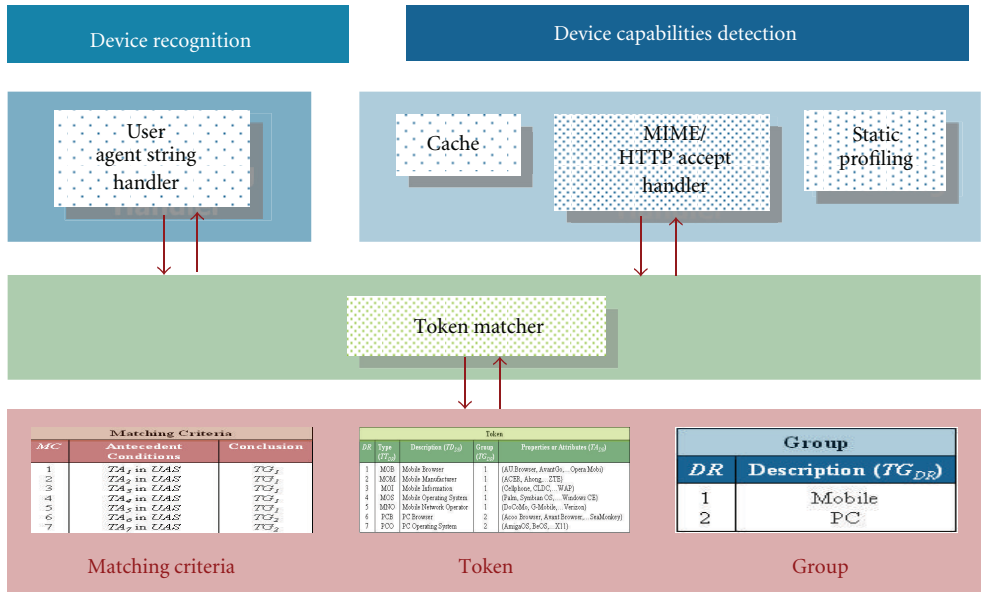


Figure 9: DICAL architecture.

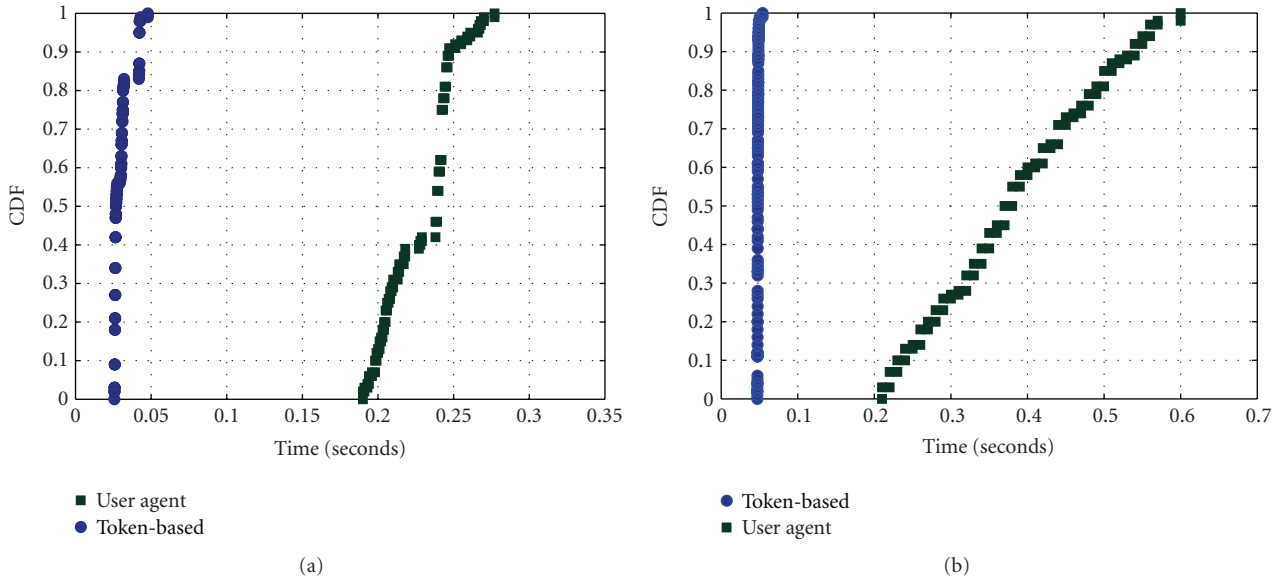


Figure 10: CDF of the query processing time for (a) device identification (b) device capabilities detection.

attempt to identify the device capabilities based on HTTP accept and MIME headers. If the device capabilities are still undetectable, the server will try to find it in the static device profiling repository. A simplified architecture of DICAL is depicted in Figure 9. We briefly describe the function of each component as follows.

HTTP *user agent request header (UA) handler* is responsible for retrieving the user agent string request headers that contain a line of text for identifying a client device. Most of the time, the device model and manufacturer can be identified from the user agent header which can also contain information such as the client device's operating system, browser and java capabilities. These headers will be used in the token attributes matching process for device identification.

Cache is employed to reduce token attributes matching processing time by eliminating subsequent token attributes lookup for identifying and describing the characteristics and capabilities of a client. DICAL keeps a cache of recently recognized user agent strings together with their capabilities

and retrieves such strings immediately when requested again. Two different schemes of caching are being employed: (a) server-side caching: caches the device information at the server side and (b) client-side caching: this caching mechanism allows the server to store information regarding the device capabilities at the client side.

MIME/HTTP *accept handler* is accountable for capturing multipurpose internet mail extensions (MIME) type or HTTP accept headers. These headers are passed to the token matching process for on-the-fly device capabilities detection. In case of the browser does not return any MIME values, the handler will specify the capabilities through HTTP accept header.

Static profiling is used to detect device capabilities from a predescribe set of attributes for a particular mobile devices. This method will be used if the device does not provide any MIME type or HTTP accept headers information.

Token attributes matcher (TAM) will be used for: (1) device identification and (2) device capabilities detection. For device identification, UA handler will forward the captured UAS to TAM. TAM will then match the UAS with all the token attributes based on the predefined matching criteria rules and conditions. For capabilities detection, HTTP accept or MIME header from MIME/HTTP accept handler is forwarded to TAM where the same process of token attributes matching (as device identification) is performed.

4. Experimental Results

We have performed several experiments to validate our proposed method. The first aims at evaluating the query processing time of token and user agent-based device identification and capabilities detection. Token-based technique compares attributes in the captured UAS or MIME header with a predefined token to determine whether there is a match between these attributes and tokens. If there is a match between UAS and token attributes, we can infer inductively that the device is a mobile device or a PC. If the attribute in the token can be corresponding pair wise with any attributes of MIME header, then we can conclude that the device is capable of supporting that particular format represented by the token attributes. User agent technique, on the other hand, performs one-to-one exact matching of the captured UAS with a device database.

The second and third experiments are to measure the accuracy of each method in identifying device type and specifying device capabilities. In the fourth experiment, we look into how much time it takes for a client to access the server and the time it takes to recognize and detect the device capabilities.

4.1. Evaluation Metrics. In the device identification experiment, we analyzed the query processing time and accuracy of the hybrid method with 235 different combinations of user agent strings in order to provide a comprehensive testing with different types of user agent syntax declaration. The query processing time here is the time from the moment the user agent string is being processed to the moment when it is being recognized. We measured the processing time for device identification and capabilities detection by obtaining the time right before the process's start (T_{start}) and just after its completion (T_{finish}), and then we computed the difference between the two ($T_{\text{finish}} - T_{\text{start}}$). In query processing time, for each experiment, we ran both token-based and user agent approaches 100 times. The client access latency time is defined as the total time a client request (T_{access}) spends waiting for the server to identify the client ($T_{\text{identification}}$) and specify the device capabilities ($T_{\text{capabilities}}$). The client access latency can be defined as

$$\text{CAL}_{\text{time}} = T_{\text{access}} + T_{\text{identification}} + T_{\text{capabilities}}. \quad (3)$$

In the accuracy evaluation for device capabilities detection, we selected 13 multimedia file formats as presented in Table 1. For each file format, the correctly detected device capabilities CDC_j from each respondent j are then compared against the supported files format SFF_j from each respondent j's mobile device to see if they are the same. The measure of accuracy DC_{Acc} is described in (4):

$$\text{DC}_{\text{Acc}} = \frac{1}{\text{FF}_i} \sum_{j=1}^{\text{FF}_i} \text{CDC}_j/\text{SFF}_j. \quad (4)$$

Table 1: Multimedia file formats for experiment.

Multimedia content	i	File format (FF)
Video	1	3GP
	2	MP4
	3	RM
	4	WMV
Document	5	PDF
	6	DOC
	7	PPT
Image	8	BMP
	9	GIF
	10	JPEG
	11	PNG
Audio	12	MP3
	13	WAV

4.2. Experimental Results

(1) *Token-Based versus User Agent-Query Processing Time.* We measured the costs of device identification and capabilities detection operations for both token-based and user agent methods by submitting a sequential stream of HTTP requests on the machines used for the experiments. Figures 10(a) and 10(b) plot the cumulative distribution function (CDF) of query processing time for each method. In this experiment the time for token-based queries lies in the range from 0.02 to 0.04 seconds for device identification and 0.04 to 0.05 seconds for device capabilities detection. We observed that the user agent approach requires more query processing time than the token-based approach. The reason is as follows.

Table 2: The result of device identification accuracy.

Approach	Number of user agent string detected	Accuracy
Token-based	235	100%
User agent	223	94.89%

User agent matcher relies on user agent HTTP header and compares this string with a database of known devices' profiles. Repeatedly attempting to match all the user agent strings in the device repository would be time consuming. In contrast, token-based use a limited pre-defined attributes or tokens for matching purposes. Moreover, there are a large number of variables in the WURFL descriptions which makes the negotiation process very complex. In the token-based solution, the flexible matching criteria process speeds up and reduce the complexity of the whole content negotiation method, thus making it faster than user agent approach.

(2) *Device Identification Accuracy.* The accuracy for device identification is presented in this section. Table 2 shows the accuracy corresponds to each method. In the experiments, we performed, hybrid approach was found to be able to recognize device more accurately than the user agent approach. The results clearly demonstrate that the idea of providing flexible and scalable approaches was an effective solution and resulted in significantly more accurate results. Token-based outperforms user agent approach due to the fact that this method does not depending on the matching process of user-agent strings with a device profiling database. Relying on such a solution, however, means the corresponding database has to be continually updated to remain current. This is because token-based captures and interprets the HTTP user agent string on-the-fly as a request is made, thus it does not need to know anything else about the device and no device profiling database needs to be maintained.

(3) *Device Capabilities Detection Accuracy.* From the results of comparative study, it was clear that a token-based technique has several advantages over the currently used user agent string approach. Hence, we decided to further validate the performance of token-based approach by deploying it in the hybrid method on an outdoor testbed. While the first testbed was setup in a controlled laboratory environment, the second testbed is deployed in a real-world application. We performed the study on 23 participants who owned a mobile phone with a mobile browser. Table 3 shows various types of mobile devices used by participants for this experiment. We provided a briefing session before the testing started. All participants will need to access our website and specify their device characteristics and capabilities explicitly through online form. In order to validate the user input, we perform a cross-check test based on user agent string and MIME headers. Upon completion, participants will submit the form which contains the information about their device capabilities to the server. A screen shot showing a portion of the form can be seen in Figure 11.

Table 3: Types of mobile devices used by respondent.

Respondent	Device	Respondent	Device
1	iPhone 3G	13	iPodTouch
2	Nokia 6120 Classic	14	LG-KG300
3	Samsung-S8300T	15	Nokia C3
4	Palm Tungsten T5	16	Nokia E71
5	HP iPAQ HX4700	17	LG-KU380
6	Nokia 5310 Xpressmusic	18	Nokia N80
7	HP iPAQ rw6800	19	Nokia N79
8	Blackberry 9700	20	O2 Xda Flame
9	HTC HD Mini T5555	21	Nokia N73
10	SonyEricsson W890i	22	LG-KM900
11	Samsung-SGH-i780	23	Nokia N95
12	SonyEricsson C510		

Table 4: Description of test devices.

Model	CPU (MHz)	RAM (MB)	Accept/MIME headers (Bytes)
Palm Tungsten T5	416	55	718
HP iPAQ h6300	200	64	3
HP iPAQ rw6800	416	64	3
iPhone 3G	412	128	209
HP iPAQ h4350	400	64	3

The accuracy of hybrid and static approaches is presented in Figure 12. The experimental results show that the hybrid approach is capable of determining device capabilities for most test cases. Based on the experiment, the hybrid method achieves high accuracy (more than 90%) for 3GP, MP4, PDF, BMP, JPEG, GIF, MP3, and WAV. It is interesting to see that this method is still manage to achieve more than 55% accuracy for DOC and PPT even though the information regarding these 2 file formats are not provided in Tera-WURFL.

Without cache, (2) with cache. We also disabled all web browsers caching in this experiment. The detail characteristics of each test device are presented in Table 4. The server machine is a Windows 2003 Server AMD Duron processor running at 491 MHz with a 368 MB RAM, and the router is a LinkSys 802.11b access point. To simulate the low-speed connections, the data rate of the link between the client and server is limited to 10 kbps for uplink and 20 kbps for downlink using a program called Netlimeter [33]. (GPRS networks have data rates of 10–20 kbps in uplink and 10–40 kbps in downlink [34]).

Figure 13 shows the result of client access latency. It can be seen from the graph that when the device information is cached, the client access latency will be reduced. As we have expected, the server-side caching can achieve a higher level of latency reduction, since only a smaller amount of data needs to be transferred between client and server. Interestingly,

Figure 11: Portion of the device characteristics and capabilities form.

this reduction is further decreased to 0.0006 seconds when the client-side caching is being deployed. This is because the client-side caching improves client access latency by avoiding round trips to the server and allowing the device information to be retrieved from the client device itself. This experiment verifies that although the process of requesting, retrieving, and determining device capabilities features can significantly increase the client access latency, this issue can be overcame by implementing caching mechanism at either the client-side or server-side.

5. Discussion

Overall, the hybrid approach performed better than static approach in all cases. It achieved an average accuracy of 87.02% and is better than the static approach by 28.54%. This indicates that by combining token-based with static and dynamic approaches would lead to more accurate device identification and capabilities detection. The dependency on the static information from the device profiling database is inadequate for content negotiation. The hybrid method has been designed to handle new devices that have been recently introduced in the market. When this method interprets a request from a device that is not available in static profiling, it uses the user-agent string, MIME and HTTP accept headers to identify and specify the device features and capabilities. In this paper, two different schemes of caching are introduced by allowing frequently accessed device information to be fetched from cache in order to improve client access latency. The improvement in access latency is important especially in the case of resource-poor mobile environments.

6. Conclusion

With the rising number of the use of mobile devices under the high-resource constraint environments, there is a sturdy need to deploy reliable device identification and capabilities detection solutions. This solution is critical because without reliable technique, content adaptation process based on the properties of the client terminal is ineffective. In this paper, we have presented architecture for determining device characteristics and capabilities. A hybrid content negotiation technique called DICAL has been developed that is capable of identifying and specifying the characteristics and capabilities of client devices. Various capabilities detection and caching approaches have been designed and implemented and their effectiveness have also been evaluated in determining the actual capabilities of the terminal. In this paper, token attributes matching, device identification, and capabilities detection algorithms have been developed to determine the device type and capabilities.

The hybrid method was constructed upon the idea of combining dynamic and static device capabilities detection methods. The main objective is to provide a flexible, extensible, and scalable approach. The flexibility and extensibility aspects are achieved by allowing users to add new content types or new user agent string features or to update rules in the matching criteria repository. The comparative experimental results highlight the importance of using token attributes matcher by eliminating the need of using the entire user agent strings for device identification and capabilities detection. Besides reducing the processing overhead it also achieves better results in terms of accuracy compared to the user agent approach.

Future research activity will aim at providing adapted content not only according to the device capabilities limitations but also network resources constraints. The dynamic properties of wireless networks and their availability should also be considered when developing and implementing UMA so that appropriate content can be delivered based on the current bandwidth condition.

We also plan to expand the device discovery approach by incorporating and deploying UAProf in the static profiling

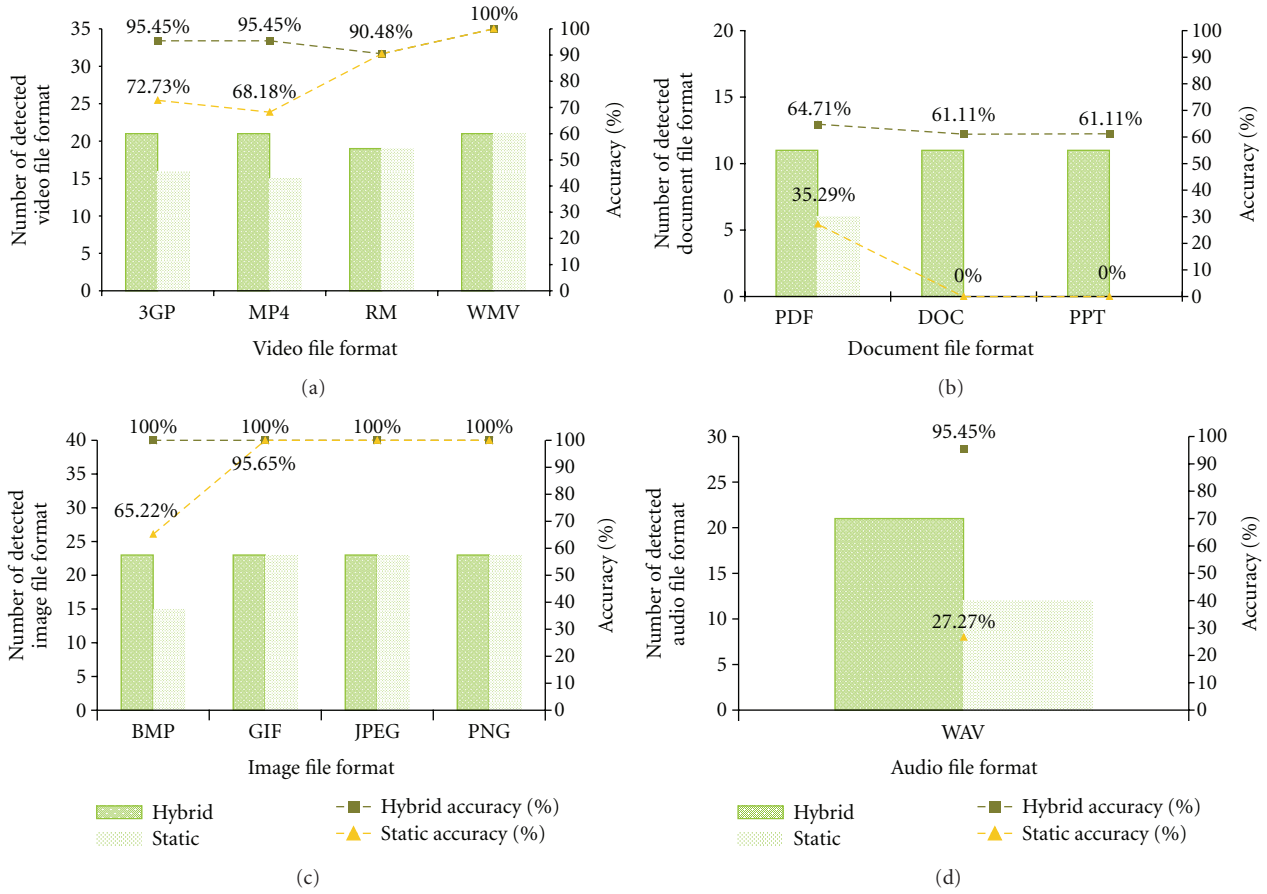


Figure 12: The accuracy of the device capabilities detection for (a) video, (b) document, (c) image, and (d) audio.

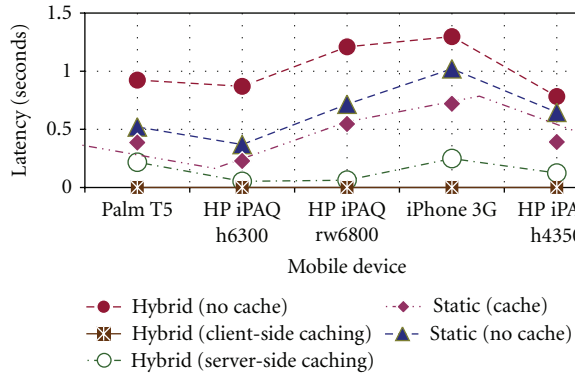


Figure 13: Client access latency.

solution to further increase the potential of recognizing and determining device capabilities in order to meet the needs of the heterogeneous device access. This effort will become increasingly important especially when the number of wirelessly connected devices is expected to grow.

References

[1] S. J. H. Yang, J. Zhang, R. C. S. Chen, and N. W. Y. Shao, "A unit of information-based content adaptation method for improving web content accessibility in the mobile internet," *ETRI Journal*, vol. 29, no. 6, pp. 793–807, 2007.

[2] F. Reynolds, J. Hjelm, S. Dawkins, and S. Singhal, Composite Capabilities/Preference Profiles (CC/PP): A User Side Framework for Content Negotiation, W3C Note, 1999, http://www.w3.org/TR/NOTE-CCPP.

[3] WAP Forum, "User agent profile specification, WAP forum," Tech. Rep. WAP-248, October 2001.

[4] L. Passani, Wireless Universal Resource File Library (WURFL), 2009, http://wurfl.sourceforge.net.

[5] X. Sanchez-Loro, V. Beitran, J. Casademont, and M. Catalan, "Ubiquitous web access: collaborative optimization and dynamic content negotiation," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 3, no. 3, 2008.

[6] S. Manoharan, "Dynamic content management and delivery for mobile devices," in *Proceedings of the International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM '07)*, pp. 63–67, November 2007.

[7] X. Zhao and T. Okamoto, "A device-independent system architecture for adaptive mobile learning," in *Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies (ICALT '08)*, pp. 23–25, July 2008.

[8] J. He, T. Gao, W. Hao, I. L. Yen, and F. Bastani, "A flexible content adaptation system using a rule-based approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 127–140, 2007.

[9] P. Salomoni, S. Mirri, S. Ferretti, and M. Roccetti, "A multimedia broker to support accessible and mobile learning through learning objects adaptation," *ACM Transactions on Internet Technology*, vol. 8, no. 2, pp. 9–23, 2008.

[10] S. Nepal and U. Srinivasan, "DAVE: a system for quality driven adaptive video delivery," in *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pp. 223–230, 2003.

[11] R. Eisinger, M. G. Manzato, and R. Goularte, "Devices descriptions for context-based content adaptation," in *Proceedings of the 3rd Latin American Web Congress (LA-WEB '05)*, pp. 121–129, November 2005.

[12] A. M. Haneef and A. Ganz, "ANMoLe—an adaptive multimedia content delivery middleware architecture for heterogeneous mobile multi-device neighborhoods," *Multimedia Tools and Applications*, vol. 22, no. 2, pp. 171–186, 2004.

[13] J. L. Hsiao, H. P. Hung, and M. S. Chen, "Versatile transcoding proxy for internet content adaptation," *IEEE Transactions on Multimedia*, vol. 10, no. 4, pp. 646–658, 2008.

[14] R. Mohan, J. R. Smith, and C. S. Li, "Adapting multimedia internet content for universal access," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 104–114, 1999.

[15] W. Y. Ma, I. Bedner, G. Chang, A. Kuchinsky, and H. Zhang, "Framework for adaptive content delivery in heterogeneous network environments," in *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, pp. 86–100, January 2000.

[16] C. Canali, V. Cardellini, and R. Lancellotti, "Content adaptation architectures based on squid proxy server," *World Wide Web*, vol. 9, no. 1, pp. 63–92, 2006.

[17] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011.

[18] R. Fielding, J. Gettys, J. Mogul et al., *RFC 2616: Hypertext Transfer Protocol—HTTP/1.1*, W3C Network Working Group, World Wide Web Consortium, 1999.

[19] H. Ohto and J. Hjelm, CC/PP Exchange Protocol Based on HTTP Extension Framework, W3C Working Draft, June 1999, http://www.w3.org/TR/NOTE-CCPPexchange.html.

[20] F. Reynolds, "Adapting content," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 6–8, 2008.

[21] M.H. Butler, "CC/PP and UAProf: issues, improvements and future directions," Tech. Rep. HPL-2002-35, Information Infrastructure Laboratory, HP Laboratories, Hewlett Packard, 2002.

[22] C. Papachristos and E. Markatos, "A CC/PP aware apache web server," in *Proceedings of the 7th CabertNet Radicals Workshop*, 2002.

[23] R. Hexel and E. Widjono, "WETA: bringing together Mobility and the Web," in *Proceedings of the International Conference on WWW/Internet*, pp. 477–484, 2004.

[24] T. Laakko and T. Hiltunen, "Adapting web content to mobile user agents," *IEEE Internet Computing*, vol. 9, no. 2, pp. 46–53, 2005.

[25] X. Sanchez-Loro, V. Beltran, J. Casademont, and M. Catalan, "Ubiquitous web access and application layer optimization: dynamic content negotiation over cellular links," in *Proceedings of the 3rd International Conference on Grid and Pervasive Computing Symposia/Workshops (GPC '08)*, pp. 269–274, May 2008.

[26] M. Hinz and Z. Fiala, "Context modeling for device and location-aware mobile web applications," in *Proceedings of the 3rd International Conference on Pervasive Computing*, 2005.

[27] M. Butler, L. Tran, E. Izdepski et al., Composite Capability/Preference Profiles (CC/PP) Processing Specification, W3C Public Draft, World Wide Web Consortium, 2002.

[28] B. Jankowska, "Approaches for device-independent content delivery to mobile devices," in *Wirtschaftsinformatik 2005, eEconomy, eGovernment, eSociety*, O. K. Ferstl, E. J. Sinz, S. Eckert, and T. Isselhorst, Eds., Physica, Heidelberg, Germany, 2005.

[29] B. Shen, W. T. Tan, and F. Huve, "Dynamic video transcoding in mobile environments," *IEEE Multimedia*, vol. 15, no. 1, pp. 42–51, 2008.

[30] Wide Web Consortium World, Composite Capability and Preference Profile, http://www.w3.org/Mobile/CCPP/.

[31] WAP Forum, User Agent Profile, http://www.wapforum.org/what/technical/SPEC-UAProf-19991110.pdf.

[32] J. Müller, T. Lenhart, D. Henrici, M. Hillenbrand, and P. Müller, "Developing web applications for mobile devices," in *Proceedings of the 1st International Conference on Distributed Frameworks for Multimedia Applications (DFMA '05)*, pp. 346–350, February 2005.

[33] Locktime Software, 2008, http://www.netlimiter.com/.

[34] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, "TCP over second (2.5G) and third (3G) generation wireless networks," The Internet Society RFC3481, 2003.