An Adaptive Approach to Multimedia Adaptation in Context-Aware Pervasive Systems

Abdelghafar Ayadi
Department of Mathematics and Computer Sciences, University of Oum El Bouaghi, Algeria | ICOSI Laboratory, University of Abbes Laghrour Khenchela, Algeria
ayadi.abdelghaffar@univ-khenchela.dz (corresponding author)

Asma Saighi
Department of Mathematics and Computer Sciences, University of Oum El Bouaghi, Algeria | Artificial Intelligence and Autonomous Things Laboratory, University of Oum El Bouaghi, Algeria
saighi.asma@univ-oeb.dz

Zakaria Laboudi
Department of Networks and Telecommunications, University of Oum El Bouaghi, Algeria
laboudi.zakaria@univ-oeb.dz

Received: 8 December 2024 | Revised: 12 January 2025 | Accepted: 29 January 2025

 DOI: https://doi.org/10.48084/etasr.9873

ABSTRACT

In pervasive systems, contextual constraints can hinder the smooth execution of multimedia documents. Several adaptation approaches have been proposed that operate as client-side, server-side, proxy-based or peer-to-peer applications. Most of these approaches rely on centralized mechanisms that adapt content at a single point. While centralized mechanisms offer several advantages, they also introduce significant drawbacks related to resource overhead. In this work, we propose a novel hybrid approach for multimedia document adaptation that efficiently combines multiple adaptation categories, such as client-side, server-side, proxy-based, and peer-to-peer, to alleviate some of their limitations. The main objective is to improve the quality of service in adaptation processes. The proposed process can operate on different devices, enabling self-reconfiguration by seamlessly switching from one adaptation category to another, depending on the computation context of the users, i.e., the device capabilities. To validate our proposal, we focus on three key aspects: first, real-world prototyping applied to the adaptation of multimedia content for e-learning materials; second, an assessment of performance in terms of response time, considering the number of adaptation requests and the available computing resources of each device; and third, a comparison with related work. The results show that as the number of adaptation requests increases, our approach outperforms traditional methods that rely on a single location. Specifically, it achieves response time reductions of up to 53.3% over the client/proxy/server hybrid scheme, 66.6% over the client/proxy hybrid scheme, and 74.3% over the client-only scheme. These findings are highly satisfying and encouraging, demonstrating that distributed computation can effectively optimize resource consumption, reduce response time, and maintain high scalability and robustness.

Keywords-multimedia document adaptation; pervasive systems; client-side adaptation; server-side adaptation; proxy-based adaptation; peer-to-peer adaptation; computation resources

I. INTRODUCTION

A. Scope of the Study

Multimedia documents are composed of various media objects such as video, images, audio, and text. This content is used in various domains including distance learning, e-health, and tourism. The diversity of devices has contributed to making such documents accessible anytime, anyhow and from anywhere. However, as the context changes over time, certain restrictions may arise that could affect the proper execution of these documents. One way to deal with such situations is to adapt the multimedia content to the current constraints of the users. This refers to the process of modifying the content and structure of documents while preserving their semantics using transformation, transmoding, and transcoding operations, based on contextual information [1]. For example, if the user is on

public transportation, video sounds in multimedia documents should be muted and replaced with subtitles instead.

Context refers to any information that characterizes the situation of entities (people, places, or objects) that is relevant to how users and applications interact. It includes details about users, applications, and the environment. Typically, context information covers four broad facets: physical context (e.g., ambient temperature and noise), computing context (e.g., communication connectivity, device capabilities, runtime), temporal context (e.g., date, schedule), and user-related context (e.g., location, profile). Context information is typically structured and organized using specific models (e.g., key-value, ontology-based modeling).

Several approaches to multimedia adaptation in pervasive environments have already been proposed [1-21]. Typically, the adaptation process follows a generic three-layer architecture [22-24]. First, users' context data are sensed by physical, logical, and virtual sensors. Then, these data are analyzed to identify the constraints that make the context non-compliant with the original content features (e.g., low battery level with high quality videos). Finally, adaptation actions are inferred and executed to deliver adapted documents.

B. Related Works

Much research has been conducted on the adaptation of multimedia documents in context-aware pervasive systems [2-21]. These approaches differ mainly in how context is sensed and modeled, and how they respond to contextual changes. Based on where the decision making and adaptation actions take place, adaptation processes are classified into four main categories: client-side, server-side, proxy-based, and peer-to-peer adaptation, each of which has its own advantages and disadvantages [1]. The choice of a particular category depends on various aspects, especially the device characteristics in terms of runtime and computing resources.

- Server-side adaptation: Adaptation is performed according to the client-server paradigm, where the device playing the multimedia content sends adaptation requests to a server [2, 6]. In this option, the server helps client devices maintain performance by efficiently using its computing resources. However, this can lead to severe overload problems that negatively affect response time.
- Client-side adaptation: Client devices are assumed to be capable of performing the adaptation operation themselves [5, 13]. This feature gives some systems more responsiveness but most are likely to struggle due to the limited device resources such as battery, CPU, and memory.
- Proxy-based adaptation: Adaptation is facilitated by a proxy that acts as an intermediary between the server and the clients [8, 19]. The proxy offloads computationally intensive tasks from both the client and the server by using its own processing power. However, this is likely to result in too much communication to negotiate the tasks at hand.
- Peer-to-peer adaptation: The devices playing multimedia content can communicate with each other as well as with other platforms, to perform adaptation tasks [9, 11]. This scheme leverages the peer-to-peer model to provide more service options. Nevertheless, the number of connected nodes may affect the availability of services.

The analysis of existing adaptation approaches reveals that:

- These methods are mainly built around the client-server model due to the simplicity and availability of computation resources. However, the effective use of the other models can also be a good alternative to address many situations such as fault tolerance, service availability, and workload balancing.
- These methods tend to concentrate the adaptation module in a single location, typically on servers. As a result, the workload is centralized, resulting in increased response time when handling adaptation requests, especially as the volume of requests increases.

C. Study Contribution and Motivation

To mitigate some of the drawbacks of the above adaptation categories, we propose a novel hybrid approach to multimedia document adaptation that exploits the computational resources of each adaptation module, including client, proxy, server and peer. The key idea is to perform the adaptation process according to an optimal choice of where to adapt the documents, depending on the constraints imposed by the current computation context, such as runtime and device computation resources. For example, consider a mobile user using a smartphone with a low battery. If multimedia adaptation is required, it may be preferable to run the adaptation process on a proxy, provided that it is underloaded. If the proxy is overloaded, it should be better to rely on either a server or another peer with better computing conditions. To this end, we rely on an effective hybridization that combines principles from different categories of adaptation approaches. The proposal is expected to ensure two essential elements:

1. Adaptability: Pervasive systems target both multi-device and cross-device interactions, as users often interact with multiple devices throughout the day, whether in synchronous, parallel, or independent use. Thus, it is essential that the adaptation process is flexible enough to operate on different types of devices such as smartphones, laptops, servers, etc., depending on the computational context of the users (runtime and computation resources).
2. Polymorphism: A well-defined software architecture enables polymorphism, which supports two features:
    - Self-reconfiguration: The system's ability to automatically adjust its behavior in response to changing conditions and requirements, without external intervention, to optimize performance. This ensures continuous operation without manual reconfiguration.
    - Hybrid scheme execution: The system's ability to seamlessly switch between different adaptation techniques such as client, server, proxy, and peer when the current solution reaches its limits. It depends on factors such as available resources, system load, or

device capabilities. This also implies that the process can be distributed across multiple locations, while maintaining optimal functionality and efficiency as conditions change.

D. Research Methodology

The proposed model is initially tested and validated using a small-scale real prototype to demonstrate the feasibility of our approach. This prototype focuses on the adaptation of html pages for e-learning, providing a representative perspective on the practical implications of our solution. Such a system is designed to support many active users (learners), creating a large network of collaborative peers, which highlights the scalability and practical relevance of our approach. However, since it is very challenging to provide the necessary hardware resources to scale for a large number of users, the overall performance is measured through simulation. The tests are conducted through various case studies, comparing the response times of client-side, proxy-based, and server-side adaptation processes with the hybrid adaptation approach proposed in this paper. The performance is measured through case analysis, evaluating the scenarios that the adaptation process is likely to encounter. The experimental results indicate that our proposal could achieve significant improvements over related works, particularly in the case of the hybrid scheme. Overall, the outcomes obtained are promising and satisfactory, as they could achieve improvements over existing adaptation approaches based on different criteria.

II. MATERIALS AND METHODS

A. Model Design

In this section, we describe our proposed hybrid approach to multimedia document adaptation, the general architecture of which is shown in Figure 1. The proposed architecture consists of three main components, structured in layers. This partitioning is done so that each layer performs a subset of well-defined tasks, including the sensing, thinking, and acting elements discussed in [22-24], in addition to intercomponent communication:

- Client component: This component plays the role of both client and peer. A client component can perform an entire adaptation process on its own if it has sufficient computing resources and adaptation services.
- Server component: This component plays the role of a server, which has the capacity to perform thinking tasks and invoke adaptation services if it has enough adaptation services and less workload.
- Proxy component: With less computing power than a server, a proxy can also perform thinking tasks and invoke adaptation services if it has enough adaptation services and less workload.

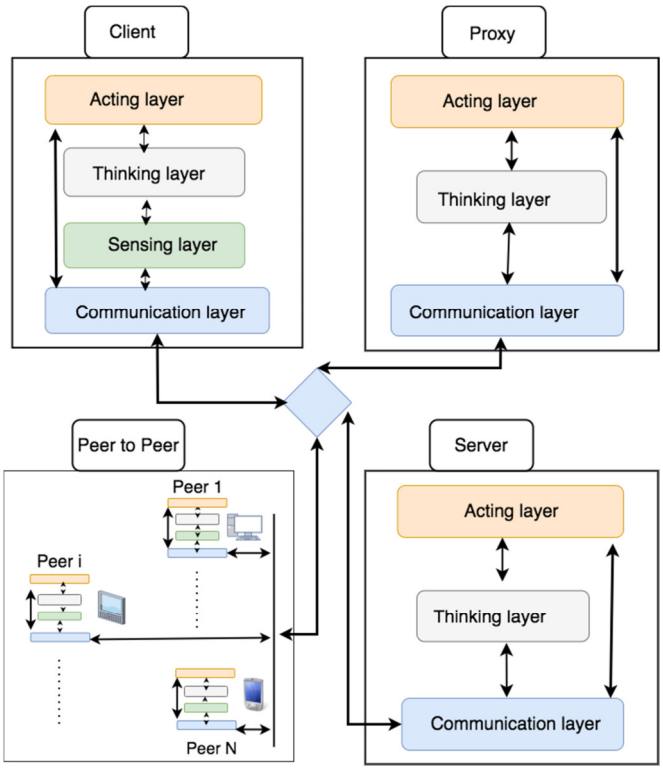

Fig. 1. The general architecture of the proposed adaptation approach.

Table I summarizes the roles of each layer. Depending on the availability of computing resources, workload, and adaptation services, the adaptation process is performed according to four scenarios: client-side adaptation, client/proxy adaptation, client/proxy/server adaptation, client/proxy/server/peer-to-peer adaptation, as shown in the flowchart in Figure 2. When the context changes, the adaptation process is triggered. The sensing layer within the client component sends sensed data (global context) to the thinking layer, which analyzes it and forwards the results to the acting layer. If the client can adapt the document itself (i.e. it has the computing capacity and the necessary adaptation services), the acting layer executes all adaptation actions. Otherwise, if the client cannot execute all or part of the adaptation process, the acting layer informs the communication layer to send an adaptation request to the proxy. Likewise, depending on its current state in terms of workload and available adaptation services, the proxy may fully or partially satisfy the client's request or even ask the server to intervene. In the case where the server can only perform part of the actions due to overload, the proxy delegates the rest of the adaptation task to other peers. In this case, its acting layer asks the facilitator to determine the list of the most appropriate peers and then informs its communication layer to send this list back to the client. Consequently, the acting layer of the client component informs its communication layer to send the adaptation request to the selected peers in a sequential manner until it finds peers that meet the request. If the adaptation process is unable to find peers that meet the adaptation requirements, the system initiates a client-side procedure by requesting a lower-quality version of the document, which may

involve the proxy or server. This approach is a resource degradation strategy that favors process robustness over performance. According to this execution scheme, the adaptation process exhibits self-adaptation in such a way that each adaptation request is seamlessly executed as a polymorphic process, ranging from client-side, proxy-based, and server-side adaptation to an effective hybridization of all these techniques.

TABLE I. SUMMARIZED DESCRIPTION OF THE LAYERS USED IN THE PROPOSED ARCHITECTURE

Layer	Leyer's role
Sensing	Sensing changes in context values
Thinking	Inference of adaptation actions
Acting	1. Adaptation plan generation and service execution 2. Peers subscription 3. Selecting peers to execute adaptation actions
Communication	Communication interface between the components

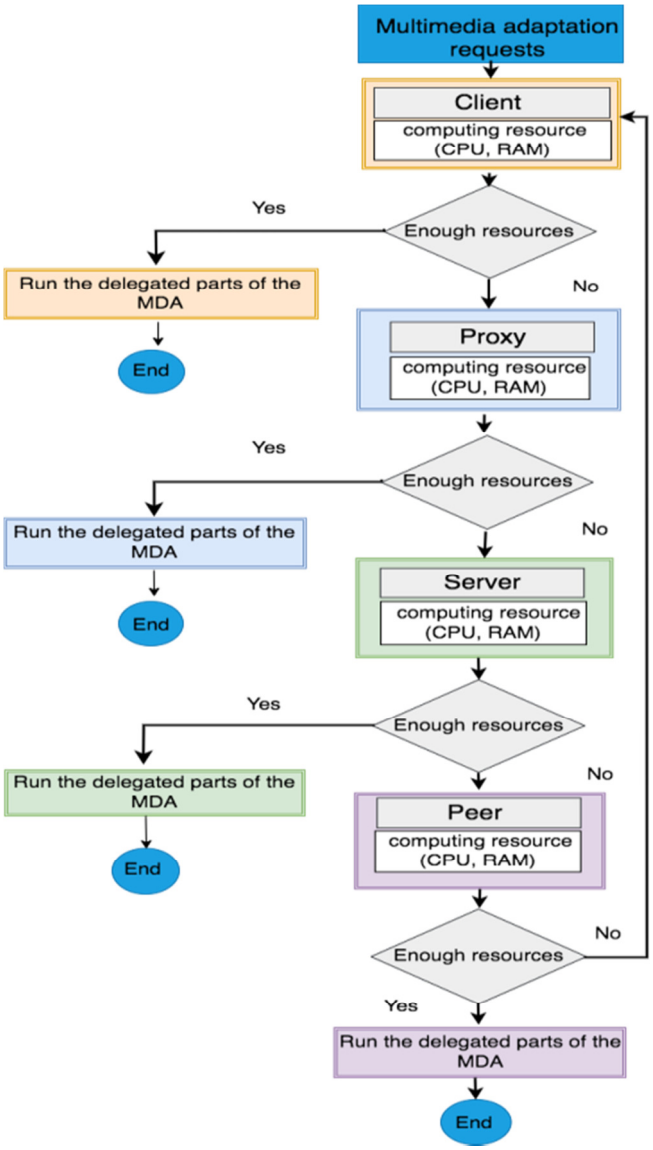

Fig. 2. Model design of the proposed adaptation process.

B. Model Formulation

To achieve the stated objectives, formulas used in grid computing studies are adopted to simulate the determination of client, proxy, server, and peer states [19, 25, 26]. The current computing context of each system is calculated based on the current resource and the length of the queue according to (1):

$$Current_{computing\ context} = Current_{resource} + L_{queue} \quad (1)$$

The current resource is determined by the ratio of the current CPU and RAM to the work assigned to the CPU and RAM according to (2):

$$Current_{resource} = \frac{Current_{CPU} + Current_{RAM}}{W_{CPU} + W_{RAM}} \quad (2)$$

The current CPU is determined using the work assigned to the CPU, the CPU load, and the ratio of the CPU speed to the minimum level required for the CPU according to (3):

$$Current_{CPU} = W_{CPU}(1 - CPU_{load})\frac{CPU_{speed}}{CPU_{min}} \quad (3)$$

The current RAM is determined using the allocated RAM work, the RAM usage, and the ratio of the RAM size to the minimum required amount of RAM according to (4):

$$Current_{RAM} = W_{RAM}(1 - RAM_{usage})\frac{RAM_{size}}{RAM_{min}} \quad (4)$$

The queue length of each system is determined by multiplying the packet arrival time with the average standby time according to (5):

$$L_{queue} = R_{arrival-time} \times T_{standby} \quad (5)$$

where:

- $Current_{resource}$ is the resource consumption of a client, proxy, server, and peers.
- L_{queue} is the length of the queue representing the work currently in progress.
- W_{CPU} is the work assigned to the CPU.
- CPU_{load} is the current CPU load.
- CPU_{speed} is the operational speed of the CPU.
- CPU_{min} is the minimum level required for the CPU to perform the requested task.
- W_{RAM} is the work assigned to the RAM.
- RAM_{usage} is the current RAM usage.
- RAM_{size} is the RAM size.
- RAM_{min} is the minimum amount of RAM required to perform the requested task.
- $R_{arrival-time}$ is the packet arrival time rate.
- $T_{standby}$ is the average standby time.

The proposed hybrid approach considers the current performance of the server, proxy, client, and connected peers.

The CPU_{load} and RAM_{usage} are randomly generated for each computing resource. These parameters play a key role in optimizing multimedia adaptation processes. As discussed in [17], the quality of service in adaptation processes is evaluated using two parameters: cost and benefit. Cost refers to non-functional features such as response time and reliability, whereas benefit refers to functional aspects such as image resolution and video speed. These parameters help select the best adaptation path (i.e., service selection) based on user preferences (e.g., resolution), available hardware (e.g., battery level, CPU, and memory), and adaptation service quality features (e.g., service availability, response time). For example, a user might prioritize high-resolution video, while the system considers bandwidth and battery constraints. Most adaptation approaches focus on functionality, but often overlook non-functional aspects. Therefore, the parameters set in our simulation are very representative, as they help improve the adaptation process by considering these non-functional factors too improve the overall quality of service while satisfying the functional requirements.

C. Model Validation

In this section, we address the validation aspect of our proposed model. The pseudocode that represents the general structure of our algorithm is outlined below:

```
Begin
max_request_adaptation = N;
for device(client, proxy, server, peer)
   parameter initialization
   (cpu_load, w_ram,w_cpu, ram_usage,
   cpu_min, cpu_speed, ram_size, ram_min,
   l_queue);
end for
i=0;
while i< max_request_adaptation
if (client.computing_resources are
available)
   compute adaptation_request(client);
else
   if (proxy.computing_resources are
   available)
      compute adaptation_request(proxy);
   else
      if (server.computing resources are
      available)
         compute adaptation_request(server);
      else
         j= randomint(0,num_peer -1)
         compute adaptation_request(peer_j);
      end if
   end if
end if
i=i+1;
end while
```

For each device (client, proxy, server, peer) the pseudocode representing the computing context is outlined below:

```
Begin
function adaptation_request(selected_
device)
selected_device.cpu_load += random(val);
selected_device.ram_usage += random(val);
selected_device.current_cpu = w_cpu × (1-
selected_device.cpu_load)×(selected_device
.cpu_speed/selected_device.cpu_min);
selected_device.current_ram = w_ram × (1-
selected_device.ram_usage)×(selected_devic
e.ram_size/selected_device.ram_min);
selected_device_current_computing_context
= selected_device.current_resource +
selected_device.Lqueue ;
end function
end
```

The implementation of our model has been done using the Google Colab environment, considering the process shown in Figure 2 through simulation tests. The values of the parameters are chosen according to Table II. In particular, the work attributed to the W_{CPU} parameter is assigned a slightly higher value (6) compared to the W_{RAM} parameter (4), reflecting its greater influence on the response time. These values are actually reused from [19]. However, it is important to note that these values are not fixed; they can be adjusted as needed, since they merely indicate the relative importance of CPU load and RAM usage in the performance of the adaptation process.

TABLE II. RESOURCE FACTORS FOR EACH CASE

Parameter	Client	Proxy	Server	Peer
CPU_{speed} (Ghz)	1.6	2.8	5.8	1.6
RAM_{size} (MB)	1024	4096	8192	1024
CPU_{min} (Ghz)	1.6	2.1	5.1	1.6
RAM_{min} (MB)	512	512	512	512
CPU_{load}	rand[30,60]	rand[50,70]	rand[50,80]	rand[20,50]
RAM_{usage}	rand[30,60]	rand[50,70]	rand[50,80]	rand[20,50]
$R_{arrival-time}$ (s)	2	2	2	2
$T_{standby}$ (s)	10	10	10	10
W_{CPU}	6	6	6	6
W_{RAM}	4	4	4	4

Depending on the computing resource and the waiting list of each system, the location where the adaptation will take place is determined. For each adaptation request, we calculate and compare the new value of the current computing context of each system according to (1). A portion of the simulation results in Google Colab for 300 adaptation requests is shown in Figure 3. The results show the distribution of computations among the client, proxy, server and peers, exactly according to our proposed adaptation algorithm.

III. RESULTS AND DISCUSSION

To evaluate the proposed model, three evaluation schemes are considered, feasibility demonstration, performance measurement, and comparison with related work.

```
peer adaptation 7
adaptation 291 enters waiting list at 603.00
Start handling adaptation 291  at 603.00
End handling adaptation 231  at 603.40
Start handling adaptation 234  at 603.40
End handling adaptation 283  at 603.53
proxy-side adaptation
adaptation 292 enters waiting list at 604.00
End handling adaptation 289  at 604.49
peer adaptation 14
adaptation 293 enters waiting list at 605.00
Start handling adaptation 293  at 605.00
End handling adaptation 235  at 605.09
Start handling adaptation 239  at 605.09
End handling adaptation 288  at 605.75
client-side adaptation
adaptation 294 enters waiting list at 606.00
End handling adaptation 239  at 608.14
Start handling adaptation 243  at 608.14
End handling adaptation 234  at 608.62
Start handling adaptation 240  at 608.62
client-side adaptation
adaptation 295 enters waiting list at 609.00
End handling adaptation 236  at 610.86
Start handling adaptation 237  at 610.86
proxy-side adaptation
adaptation 296 enters waiting list at 612.00
server-side adaptation
adaptation 297 enters waiting list at 613.00
End handling adaptation 291  at 613.02
peer adaptation 9
adaptation 298 enters waiting list at 615.00
Start handling adaptation 298  at 615.00
End handling adaptation 293  at 615.02
peer adaptation 15
adaptation 299 enters waiting list at 618.00
Start handling adaptation 299  at 618.00
peer adaptation 9
adaptation 300 enters waiting list at 619.00
Start handling adaptation 300  at 619.00
End handling adaptation 240  at 619.34
Start handling adaptation 244  at 619.34
End handling adaptation 298  at 620.64
End handling adaptation 244  at 626.24
```

Fig. 3. Simulation results in Google Colab.

A. *Implementation of a Prototype for the Proposed Approach*

To validate our proposal, we rely on a prototype that provides an environment equipped with e-learning components, inspired by real scenarios. The system allows the adaptation of multimedia content within the learning materials used by the learners. This serves two purposes: first, to demonstrate the feasibility of our method, and second, to provide context on the practical implications of the proposed solution. This system allows the creation of a large network of collaborative peers (learners), which highlights the scalability and practical relevance. Regarding the learning materials (courses) in our prototype, their structure is inspired by the work presented in [15], which defines four levels of course organization: course, section, concept, and content. For the implementation, we use XML-based languages to describe, represent, and visualize the structured learning objects, as they are flexible and well suited for data storage and exchange. Table III presents a subset of the adaptation actions we have adopted; these rules are inspired by those proposed in [1]. The hardware configuration includes two HP Z640 stations (20-core Xeon CPU, 64 GB RAM) that act as proxy and server. In addition, we rely on three client devices that act as peers for the execution of the learning materials. Their characteristics are as follows: a tablet (2.0 GHz CPU, 2GB RAM), a laptop (2.7 GHz CPU, 8 GB RAM), and a desktop PC (2.7 GHz CPU, 16 GB RAM). These devices were connected through a Local Area Network (LAN) to allow them to communicate in different ways. The adaptation actions were performed using dedicated APIs, and an Apache server was used to generate the adapted content.

TABLE III. RULES FOR INFERRING ADAPTATION ACTIONS

Rule	Contextual constraints	Actions
R1	if (location=lab, bus, public place)	Speech to text
R2	if (unsuitable brightness)	Adjust contrast
R3	if (time is night)	Summarize content
R4	if (battery level is low)	Summarize content Adjust contrast Change media object format

1) *Illustrative Scenarios*

Consider a college student who works part-time as a delivery driver to manage his financial needs. As a result, he relies on online and distance learning. The courses consist of materials that include text, images, audio, and video. Let us assume that the learner is on his way to the college. On the bus, he consults his courses on his tablet. The inferred contextual constraints and adaptation rules are given in Table IV.

TABLE IV. CONTEXTUAL CONSTRAINTS AND CORRESPONDING ADAPTATION RULES

Context element	Context element value	Adaptation rule
Location	Bus	R1
Battery level	25% (low)	R4

Table V summarizes the original and adapted content of a typical learning material object, where:

- The proxy converts audio objects to text.
- The server reduces the video quality and adds subtitles.
- The peer, a desktop PC, reduces images quality.
- The client summarizes text and adjusts the contrast.

TABLE V. ORIGINAL AND ADAPTED LEARNING CONTENT

Media objects	Number of objects in the original document	Adaptation location	Number of objects in the adapted document
Image	2	Peer	2
Text	3	Client	6
Video	1	Server	1
Audio	3	Proxy	0

The goal is to make the adapted content as close to the contextual constraints as possible, relying on a hybrid adaptation scheme. The above scenario highlights the

importance of understanding the context to infer constraints and efficiently perform useful actions accordingly. This ensures the continuity of the service is ensured even at lower quality levels. It is worth noting that our prototype can be extended to support different types of contextual elements, such as ambient weather conditions and noise, thus enabling further adaptation rules.

B. Performance Analysis

Since providing the necessary hardware resources to scale to a large number of users is very challenging, the overall performance is measured by simulation. Performance measurement is based on case analysis, which evaluates the response time scenarios that the adaptation process is likely to encounter.

- Best case: The scenario in which we expect the adaptation process to perform very well.
- Worst case: The scenario in which we expect the adaptation process to perform poorly.

These two cases depend on three main factors: the available computing resources, the number of adaptation requests, and the network conditions regarding the latency. The more computing resources are available, the smaller the number of adaptation requests, and the more ideal the network conditions, the faster the response time is expected to be. Conversely, the more limited the computing resources, the more important the number of adaptation requests, the more basic the network conditions, and the more the response time is expected to increase. For this purpose, parameter values were chosen to simulate a large number of adaptation requests while varying the availability of computing resources and network conditions. Thus:

- By setting the number of peers to 20%, we assume that only a few peers are willing to contribute to the realization of the adaptation requests. This helps to demonstrate the effectiveness of our proposal.
- By adding a random value $V_{lat-bdwidth}$ to the adaptation request time, we simulate the impact of bandwidth and latency on the adaptation process and show how it behaves under varying network conditions. Latency and bandwidth, which are influenced by factors such as transmission delays and protocol overhead, mainly affect the network responsiveness and thus the adaptation response time.
- By setting the time between two adaptation requests to 2 s and the average waiting time to 10 s, we assume that the number of adaptation requests is important, so that the workload is very high (i.e., there is always work to be done on the waiting lists).

The workload is studied according to five scenarios:

- In scenario 1 (the blue curve in Figure 4), the adaptation process is handled at a single point. The entire workload is delegated to the client as long as it has sufficient resources. The response time for satisfying all the adaptation requests is very high. The obtained result is nearly 2919 s for 300 adaptation requests.
- In scenario 2 (the red curve in Figure 4), the client lacks sufficient resources to perform the entire adaptation process. Therefore, it handles part of the adaptation actions while the proxy performs the rest, since it has enough computing resources. This workload sharing results in a decrease in response time. For 300 adaptation requests, the response time is nearly 2246 s. As a result, neither the server nor the peers participate in this adaptation process.
- In scenario 3 (the gray curve in Figure 4), the adaptation process is shared between the client, proxy, and server. This is because neither the client nor the proxy have enough computing resources to perform all of the adaptation actions themselves, necessitating the involvement of the server to handle the remaining portion. As the number of adaptation requests increases to 300, the response time decreases to nearly 1602 s.
- In scenario 4 (light yellow curve in Figure 4), the peers are involved in the adaptation process. The adaptation workload is distributed among the client, proxy, server, and peers due to the lack of sufficient computing resources in the early stages of the adaptation process. As a result, the response time is significantly reduced. For 300 adaptation requests, the response time is nearly 748 s due to the parallelism resulting from the workload distribution.
- In scenario 5, the evaluation involves replicating scenarios 2, 3, and 4 while considering varying network conditions (latency and bandwidth). Figure 5 shows the response time for 300 adaptation requests.

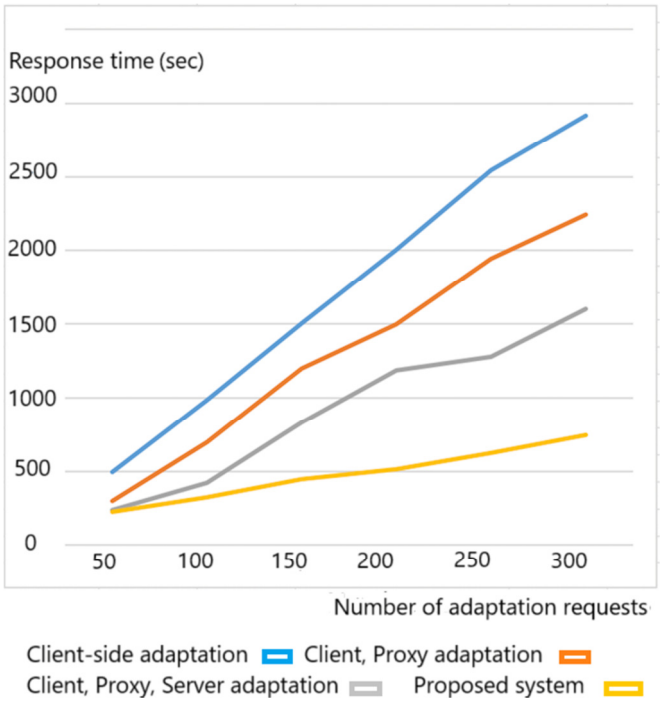

Fig. 4. Response time comparison graph.

By increasing the number of adaptation requests and limiting the availability of computing resources, our proposed

system is expected to outperform existing adaptation schemes, client-side, server-side, proxy-based, and peer-to-peer, when applied separately. For instance, one can observe the discussed scenarios illustrated in Figure 4. The more the adaptation process is distributed (i.e., not centralized in a single location), the faster the response time becomes. In fact, most existing approaches do not take into account the available computing resources and the number of adaptation requests within their processes, which makes our proposal more efficient. The response times recorded in these scenarios show that as the number of adaptation requests increases to 300, our approach outperforms traditional methods. Specifically, it achieves response time reductions of up to 53.3% over the client/proxy/server hybrid scheme, 66.6% over the client/proxy hybrid scheme, and 74.3% over the client-only scheme. It should be noted that testing different parameter values had no difference or impact on the results obtained. Varying network conditions naturally lead to a negative impact on response time in adverse situations. Nevertheless, our proposal performs better than the other adaptation schemes.

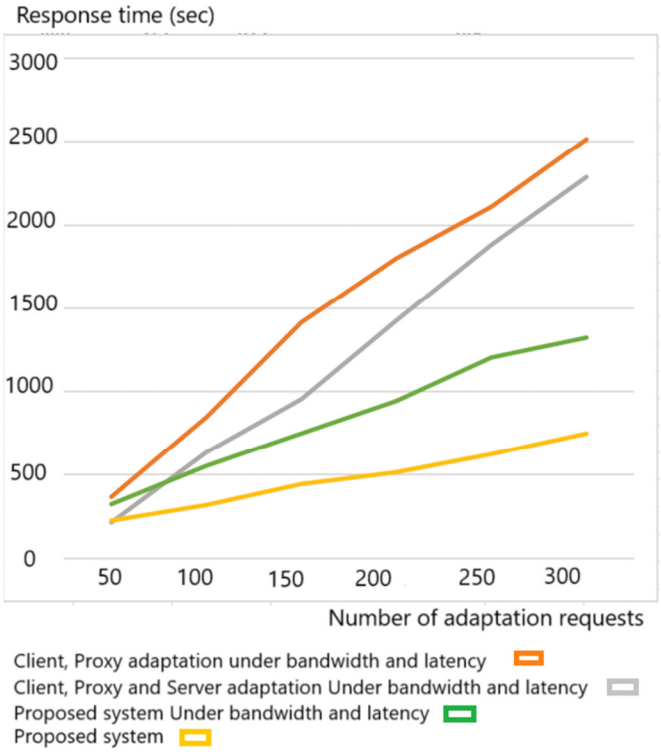

Fig. 5. Response time comparison graph.

C. Comparison of the Proposed Approach with Related Work

The third evaluation of our proposal involves a comparison with related work, as shown in Table IV, to highlight the strengths that distinguish our approach. It is important to note that the difficulty of reproducing the outputs of these works, in addition to the lack of system details, prevented us from conducting quantitative evaluations. For this purpose, we rely on a qualitative assessment through the following criteria:

- Approach category (C1): Refers to the adaptation approach to which a given research work belongs.
- Computation overload (C2): Refers to the system's ability to handle the computation cost in terms of resources required to perform the adaptation process.
- Adaptability (C3): Refers to the system's ability to execute the adaptation process across multiple devices.
- Polymorphism (C4): Refers to the system's ability to support self-reconfiguration and hybrid scheme execution of the adaptation process.
- Robustness (C5): Refers to the system's ability to continue to perform its functions effectively under varying network conditions (e.g., unexpected disruptions, technical failures, network congestion).
- Scalability (C6): Refers to the number of users the system can support.

TABLE VI. COMPARISON BETWEEN DIFFERENT MULTIMEDIA ADAPTATION APPROACHES

Reference	C1	C2	C3	C4	C5	C6
[2]	Server-side	No	No	No	No	Yes
[3]	Server-side	No	No	No	No	Yes
[5]	Client-side	No	No	No	No	No
[6]	Server-side	No	No	No	No	Yes
[7]	Server-side	No	No	No	No	Yes
[8]	Proxy-based	No	No	No	No	Yes
[9]	Peer-to-Peer	No	No	Yes	Yes	Yes
[11]	Peer-to-Peer	No	No	Yes	Yes	Yes
[12]	Server-side	No	No	No	No	Yes
[13]	Client-side	No	No	No	No	No
[14]	Server-side	No	No	No	No	Yes
[15]	Server-side	No	No	No	No	Yes
[16]	Server-side	No	No	No	No	Yes
[1]	Server-side	No	No	Yes	No	Yes
[18]	Server-side	No	No	No	No	Yes
[19]	Client-side Server-side Proxy-based	Yes	Yes	No	No	Yes
[20]	Server-side	No	No	No	No	Yes
[21]	Server-side	No	No	No	No	Yes
Our work	Client-side Server-side Proxy-based Peer-to-peer Hybrid	Yes	Yes	Yes	Yes	Yes

Examining Table VI, the following observations can be made:

1. Unlike related works that mainly focus on one adaptation approach or partial hybridization as in [9, 19], our approach potentially supports all adaptation categories in addition to their hybrid scheme. More specifically, the process depicted in Figure 2 can be easily adapted to separately execute client-side, proxy-based, server-side, or peer-to-peer schemes, providing various design choices, including traditional schemes.

2. Most related works do not address computation overhead, except for partially hybrid approaches, such as in [19], that rely only on client, proxy, and server (similar to scenario 3 in our case). Our approach efficiently handles computation overhead, leading to a significant reduction in response time. It also shows high adaptability, as it can be executed on any device with sufficient computational resources.
3. Most related works do not support polymorphism, except for hybrid approaches such as in [19], which marginally support this feature by involving only client, proxy, and server. Our approach effectively leverages polymorphism, particularly using peers, and significantly reduces the response time.
4. Except for peer-to-peer approaches as in [9], most adaptation approaches are either not robust or only marginally robust, as they rely on a single or few locations to perform adaptation [19]. Our approach is more robust because it relies on all parties, client, proxy, server, and peers.
5. Finally, except for client-side approaches as in [5], most adaptation approaches are relatively scalable, as they rely on proxies, servers, or even peers. In particular, our approach may excel because it involves all of these parties.

Overall, our hybrid approach outperforms related works when considering these criteria. This is due to the efficient exploitation of adaptation categories, especially the peer-to-peer approach, which provides a high degree of computation distribution, leading to an efficient improvement in overall performance. Moreover, compared to other adaptation approaches, our proposal stands out due to its general applicability across various adaptation categories and to a wide range of domains.

IV. CONCLUSION

In this paper, a new hybrid architecture for multimedia document adaptation has been proposed. The proposed model is based on distributed computing to perform the adaptation process at multiple levels: client-side, proxy-side, server-side, peer-to-peer as well as their hybridization. In order to evaluate the effectiveness of our proposal, the validation was performed by relying on a real prototype to show its feasibility and application implications, and on Google Colab to measure its performance, in addition to comparisons with related works. The tests were conducted across various scenarios, comparing the response times. The experimental results showed that our proposal could achieve significant improvements over related works, especially in the case of the hybrid scheme. Overall, the results obtained were promising as they could achieve improvements in terms of computation overload handling through resource utilization efficiency, adaptability, polymorphic features, robustness, and scalability. As a future project, we plan to integrate our multimedia adaptation model into a context-aware application linked to ubiquitous learning systems.

REFERENCES

[1] A. Saighi *et al.*, "On Using Multiple Disabilities Profiles to Adapt Multimedia Documents: A Novel Graph-Based Method," in *Research Anthology on Physical and Intellectual Disabilities in an Inclusive Society*, Information Resources Management Association, Ed. Hershey, PA, USA: IGI Global Scientific Publishing, 2022, pp. 173–201, https://doi.org/10.4018/978-1-6684-3542-7.ch010.

[2] A. Adel, S. Laborie, and P. Roose, "Automatic Adaptation of Multimedia Documents," *Procedia Computer Science*, vol. 19, pp. 992–997, Jun. 2013, https://doi.org/10.1016/j.procs.2013.06.138.

[3] A. Adel, R. Philippe, and L. Sébastien, "Multimedia Documents Adaptation Based on Semantic Multi-Partite Social Context-Aware Networks," *International Journal of Virtual Communities and Social Networking*, vol. 9, no. 3, pp. 44–59, 2017, https://doi.org/10.4018/IJVCSN.2017070104.

[4] F. Bettou and B. Boulkroun, "A Multi-Viewpoint Approach For Semantic Multimedia Documents Adaptation," in *Proceedings of the 9th World Congress on Electrical Engineering and Computer Systems and Sciences*, London, United Kingdom, 2023, https://doi.org/10.11159/cist23.107.

[5] Y. Belhadad, A. Refoufi, and P. Roose, "Spatial reasoning about multimedia document for a profile based adaptation," *Multimedia Tools and Applications*, vol. 77, no. 23, pp. 30437–30474, Dec. 2018, https://doi.org/10.1007/s11042-018-6080-8.

[6] F. Bettou and M. Boufaida, "An Adaptation Architecture Dedicated to Personalized Management of Multimedia Documents," *International Journal of Multimedia Data Engineering and Management*, vol. 8, no. 1, pp. 21–41, 2017, https://doi.org/10.4018/IJMDEM.2017010102.

[7] S. Sarwar, Z. U. Qayyum, R. García-Castro, M. Safyan, and R. F. Munir, "Ontology based E-learning framework: A personalized, adaptive and context aware model," *Multimedia Tools and Applications*, vol. 78, no. 24, pp. 34745–34771, Dec. 2019, https://doi.org/10.1007/s11042-019-08125-8.

[8] C. Dromzée, S. Laborie, and P. Roose, "A Semantic Generic Profile for Multimedia Document Adaptation," in *Intelligent Multimedia Technologies for Networking Applications: Techniques and Tools*, D. Kanellopoulos, Ed. Hershey, PA, USA: IGI Global Scientific Publishing, 2013, pp. 225–246, https://doi.org/10.4018/978-1-4666-2833-5.ch009.

[9] Q. P. Hai, S. Laborie, and P. Roose, "On-the-fly Multimedia Document Adaptation Architecture," *Procedia Computer Science*, vol. 10, pp. 1188–1193, Aug. 2012, https://doi.org/10.1016/j.procs.2012.06.171.

[10] E. G. Rincon-Flores *et al.*, "Improving the learning-teaching process through adaptive learning strategy," *Smart Learning Environments*, vol. 11, no. 1, Jun. 2024, Art. no. 27, https://doi.org/10.1186/s40561-024-00314-9.

[11] Z. Kazi-Aoul, I. Demeure, and J.-C. Moissinac, "PAAM: a web services oriented architecture for the adaptation of composed multimedia documents," in *International Conference on Parallel and Distributed Computing and Networks*, Innsbruck, Austria, 2008, pp. 164–169.

[12] H. Khallouki and M. Bahaj, "Multimedia documents adaptive platform using multi-agent system and mobile ubiquitous environment," in *2017 Intelligent Systems and Computer Vision*, Fez, Morocco, 2017, pp. 1–5, https://doi.org/10.1109/ISACV.2017.8054915.

[13] S. Laborie, J. Euzenat, and N. Layaïda, "Semantic adaptation of multimedia documents," *Multimedia Tools and Applications*, vol. 55, no. 3, pp. 379–398, Dec. 2011, https://doi.org/10.1007/s11042-010-0552-9.

[14] A.-E. Maredj and N. Tonkin, "CSP-based adaptation of multimedia document composition," in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing*, Anaheim, CA, USA, 2015, pp. 232–235, https://doi.org/10.1109/ICOSC.2015.7050811.

[15] I. El Guabassi, Z. Bousalem, M. Al Achhab, I. Jellouli, and B. E. EL Mohajir, "Personalized adaptive content system for context-aware ubiquitous learning," *Procedia Computer Science*, vol. 127, pp. 444–453, Mar. 2018, https://doi.org/10.1016/j.procs.2018.01.142.

[16] S. Chimalakonda and K. V. Nori, "An ontology based modeling framework for design of educational technologies," *Smart Learning*

Environments, vol. 7, no. 1, Oct. 2020, Art. no. 28, https://doi.org/10.1186/s40561-020-00135-6.

[17] Z. Laboudi, A. Moudjari, A. Saighi, A. Draa, and S. Hadjadj, "An adaptive context-aware optimization framework for multimedia adaptation service selection," *Neural Computing and Applications*, vol. 34, no. 17, pp. 14239–14251, Sep. 2022, https://doi.org/10.1007/s00521-021-06644-w.

[18] S. Ennouamani, Z. Mahani, and L. Akharraz, "A context-aware mobile learning system for adapting learning content and format of presentation: design, validation and evaluation," *Education and Information Technologies*, vol. 25, no. 5, pp. 3919–3955, Sep. 2020, https://doi.org/10.1007/s10639-020-10149-9.

[19] J. Cho, S. Lee, and E. Lee, "Multi-agent Based Hybrid System for Dynamic Web-Content Adaptation," in *Intelligent Data Engineering and Automated Learning – IDEAL 2006: Proceedings of 7th International Conference*, Burgos, Spain, 2006, pp. 1215–1222, https://doi.org/10.1007/11875581_144.

[20] Y. Lin, S. Feng, F. Lin, W. Zeng, Y. Liu, and P. Wu, "Adaptive course recommendation in MOOCs," *Knowledge-Based Systems*, vol. 224, Jul. 2021, Art. no. 107085, https://doi.org/10.1016/j.knosys.2021.107085.

[21] Y. Yousaf, M. Shoaib, M. A. Hassan, and U. Habiba, "An intelligent content provider based on students learning style to increase their engagement level and performance," *Interactive Learning Environments*, vol. 31, no. 5, pp. 2737–2750, Jul. 2023, https://doi.org/10.1080/10494820.2021.1900875.

[22] P. N. Mahalle and P. S. Dhotre, *Context-Aware Pervasive Systems and Applications*. Singapore: Springer, 2020.

[23] E. M. Milic and D. Stojanovic, "EgoSENSE: A Framework for Context-Aware Mobile Applications Development," *Engineering, Technology & Applied Science Research*, vol. 7, no. 4, pp. 1791–1796, Aug. 2017, https://doi.org/10.48084/etasr.1203.

[24] S. R. Idate, T. S. Rao, and D. J. Mali, "Context-Based Aspect-Oriented Requirement Engineering Model," *Engineering, Technology & Applied Science Research*, vol. 13, no. 2, pp. 10460–10465, Apr. 2023, https://doi.org/10.48084/etasr.5699.

[25] M. Li and M. Baker, *The Grid: Core Technologies*, 1st ed. Hoboken, NJ, USA: Wiley, 2005.

[26] J. D. C. Little, "A Proof for the Queuing Formula: L = λW," *Operations Research*, vol. 9, no. 3, pp. 383–387, Jun. 1961, https://doi.org/10.1287/opre.9.3.383.